An hybrid method for fine-grained content based image retrieval

Romaric Pighetti

UNIVERSITÉ CÔTE D'AZUR

i3S sophia antipolis

DOCTORAL SCHOOL STIC

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

Research Unit: Laboriatoire I3S - UMR7271 - CNRS

P H D T H E S I S

to obtain the title of

PhD of Science

Specialty: Computer Science

of

UNIVERSITE COTE D'AZUR

Defended by

Romaric Pighetti

A Hybrid Method for Fine Grained Content Based Image Retrieval

Thesis Advisor: Frédéric Precioso

Defended On November the 28th 2016
In front of a Jury Composed of:

Carlos Artemio Coello Coello	-	Professor, CINVESTAV-IPN	Examinator
Cyril Fonlupt	-	Professor, University Littoral Côte d'Opale	Reviewer
Denis Pallez	-	Associate Professor, Université Côte d'Azur	Co-Advisor
Frédéric Precioso	-	Professor, Université Côte d'Azur	Advisor
Arnaud Revel	-	Professor, University of La Rochelle	Reviewer
Andrea G. B. Tettamanzi	-	Professor, Université Côte d'Azur	Examinator

This PhD thesis was prepared at:
Laboratoire d'Informatique, Signaux et Systèmes de Sophia-Antipolis (I3S)
UMR7271 - Université Côte d'Azur - CNRS
team SPARKS, group MinD
2000, route des Lucioles
Les Algorithmes - bât. Euclide B
06900 Sophia Antipolis - France

Une méthode hybride pour la classification d'images à grain fin

Résumé: La quantité d'images disponibles sur Internet ne fait que croître, engendrant un besoin d'algorithmes permettant de fouiller ces images et retrouver de l'information. Les systèmes de recherche d'images par le contenu ont été développés dans ce but. Mais les bases de données grandissant, de nouveaux défis sont apparus. Dans cette thèse, la classification à grain fin est étudiée en particulier. Elle consiste à séparer des images qui sont relativement semblables visuellement mais représentent différents concepts, et à regrouper des images qui sont différentes visuellement mais représentent le même concept. Il est montré dans un premier temps que les techniques classiques de recherche d'images par le contenu rencontrent des difficultés à effectuer cette tâche. Même les techniques utilisant les machines à vecteur de support (SVM), qui sont très performantes pour la classification, n'y parviennent pas complètement. Ces techniques n'explorent souvent pas assez l'espace de recherche pour résoudre ce problème. D'autres méthodes, comme les algorithmes évolutionnaires sont également étudiées pour leur capacité à identifier des zones intéressantes de l'espace de recherche en un temps raisonnable. Toutefois, leurs performances restent encore limitées. Par conséquent, l'apport de la thèse consiste à proposer un système hybride combinant un algorithme évolutionnaire et un SVM a finalement été développé. L'algorithme évolutionnaire est utilisé pour construire itérativement un ensemble d'apprentissage pour le SVM. Ce système est évalué avec succès sur la base de données Caltech-256 contenant environ 30,000 images réparties en 256 catégories.

Mots clés: Recherche d'images par le contenu, machine à vecteur de support, algorithmes évolutionnaires, classification fine

A Hybrid Method for Fine-Grained Content Based Image Retrieval

Abstract: Given the ever growing amount of visual content available on the Internet, the need for systems able to search through this content has grown. Content based image retrieval systems have been developed to address this need. But with the growing size of the databases, new challenges arise. In this thesis, the fine grained classification problem is studied in particular. It is first shown that existing techniques, and in particular the support vector machines which are one of the best image classification technique, have some difficulties in solving this problem. They often lack of exploration in their process. Then, evolutionary algorithms are considered to solve the problem, for their balance between exploration and exploitation. But their performances are not good enough either. Finally, a hybrid system combining an evolutionary algorithm and a support vector machine is proposed. This system uses the evolutionary algorithm to iteratively feed the support vector machine with training samples. The experiments conducted on Caltech-256, a state of the art database containing around 30,000 images, show very encouraging results.
Keywords: CBIR, Evolutionary Algorithm, SVM, Fine Grained Classification

Une méthode hybride pour la classification d'images à grain fin

Résumé substentiel:

Introduction

Avez-vous déjà cherché "pizza aux anchois" dans Google image ? Si vous le faites, il y a des chances qu'une image d'Adriana Karembeu apparaisse dans les premiers résultats. Ce n'est pas vraiment ce que l'on espérait trouver, et on ne peut pas dire qu'elle ressemble à une pizza aux anchois. Alors pourquoi se retrouve-t-elle dans les résultats de cette recherche ? Il se trouve que les algorithmes utilisés par Google classent les images en utilisant les informations contextuelles disponibles autour de l'image (texte, métadonnées). L'image d'Adriana Karembeu en question se trouve sur une page traitant exclusivement et extensivement de pizza aux anchois, donnant des recettes de pizzas aux anchois, proposant des vins pour accompagner les pizzas aux anchois etc. Cet exemple est une parfaite démonstration des limitations des systèmes de recherche basés sur les informations contextuelles : ils peuvent facilement être trompés par de fausses informations. C'est pourquoi des méthodes de recherches d'images se basant uniquement sur le contenu visuel des images ont été développées. C'est ce qu'on appelle la recherche d'images par le contenu.

L'intérêt pour ces méthodes n'a pas cessé de croître durant ces dernières années, en particulier à cause de la quantité de contenu visuel mis en ligne. Entre sa création en 2004 et Décembre 2014, Flickr a amasser 5,26 billions d'images. En 2015, 2 millions d'images étaient ajoutées sur Internet chaque jour en moyenne. Il est évident que pour un tel volume de contenu, chaque donnée ne peut être accompagnée d'informations contextuelles pertinentes. Être capable d'effectuer des recherches dans ces données en se basant uniquement sur le contenu visuel devient donc nécessaire.

Afin d'effectuer une recherche d'images par le contenu, deux éléments principaux sont nécessaires : une représentation des images extrayant les caractéristiques visuelles des images, et un algorithme de fouille de données permettant de traiter ces représentations pour trier les images. Le travail de cette thèse porte sur le développement d'un algorithme hybride de classification d'images pouvant utiliser n'importe quelle représentation d'images. L'idée de la classification d'images est d'être capable de séparer en différentes catégories (bus, voitures, montagnes, ...) les images. Une revue des techniques existantes dans le domaine de la recherche d'images par le contenu est faite dans le chapitre 1. Cette revue montre que les techniques existantes présentent certaines limitations les empêchant de résoudre correctement certains problèmes. En particulier, le problème de la classification à grain fin est introduit et sera celui auquel nous nous intéresserons. Les systèmes existant semblent manquer de capacités exploratoires pour résoudre ce problème, les algorithmes évolutionnaires sont donc étudiés comme une solution potentielle car ils présentent de bonnes capacités exploratoires. Ils sont par conséquent étudiés dans ce sens dans le chapitre 2. Ils se révèlent cependant avoir des performances limitées pour répondre

aux besoins de la classification d'images à grain fin. Une méthode hybride combinant les performances de classification des machines à vecteurs de support (l'un des meilleurs algorithmes de classification d'images) aux capacités exploratoires d'un algorithme évolutionnaire a donc été conçu. Tous les détails de ce système, ainsi qu'une implémentation de celui-ci, sont donnés dans le chapitre 3. Une étude des performances de ce système, présentée dans le chapitre 4, a été mené sur une base de données de référence pour la recherche d'images par le contenu (Caltech-256, qui contient envieront 30,000 images réparties en 256 catégories). Les résultats se sont révélés très encourageant sur les capacités du système proposé, ouvrant des perspectives de travaux complémentaires qui sont présentées à la fin du document après une conclusion résumant les travaux effectués.

Recherche d'Images par le Contenu

La recherche d'images par le contenu est utilisée pour effectuer différentes tâches (de la reconnaissance de visage, de la détection de copies, de la reconnaissance d'objets etc.). La classification d'image est plus particulièrement étudiée dans cette thèse. La structure globale d'un système de recherche d'images par le contenu est d'abord présentée. Les différents éléments formant un tel système sont ensuite détaillées, en présentant les méthodes les plus communément utilisées. Les techniques d'extraction de caractéristiques visuelles et de construction de représentations d'images sont présentées en premier, avant d'introduire les systèmes de classification les plus utilisés. Les systèmes de classification les plus utilisés sont ensuite exposés. Enfin, le défi que représente la classification d'images à grain fin est présenté, en soulevant les difficultés rencontrées par les systèmes existant pour le résoudre.

Structure

La structure générale d'un système de recherche d'images par le contenu est représenté sur la figure 1.1. Dans un premier temps, une représentation tenant compte des caractéristiques visuelles des images doit être extraite pour chaque image de la base de données. Cette étape, représentée en haut de la figure, est une étape préliminaire à tout système de recherche d'images par le contenu, quel que soit son but.

Ensuite, une ou plusieurs images requêtes sont fournies au système. La représentation de chacune de ces images est calculée, et elles sont ensuite fournies à un algorithme de fouille de données qui se charge, à partir de la représentation des requêtes, de fouiller les représentations des images de la base de données et de retourner un résultat (qui dépendra de la tâche à effectuer). Dans le cas de la classification, qui est plus particulièrement étudiée ici, un ensemble d'images est fourni comme requête avec l'indication de la catégorie à laquelle chaque image appartient. Ces images forment ce que l'on appelle l'ensemble d'apprentissage qui est utilisé par l'algorithme pour apprendre et être capable de déterminer à quelle catégorie appartient une image à partir de sa représentation. Les images utilisées dans l'ensemble d'apprentissage

sont issues d'un ensemble d'images pour lesquelles la classe est connue que l'on appellera l'ensemble des images d'entraînement disponibles. Nous allons maintenant détailler les différents éléments composant un système de recherche d'images par le contenu.

Chaîne de Traitement d'un Système de Recherche d'Images par le Contenu

La chaîne de traitement d'un système de recherche d'images par le contenu commence par la construction de la représentation des images. Pour ce faire, une première étape consiste à extraire des caractéristiques visuelles des images. Ensuite, ces caractéristiques sont combinées pour former une représentation des images. Afin de pouvoir comparer des images entre elles, et en particulier savoir si des images sont semblables ou non, des mesures de similarité sont développées. Enfin, les algorithmes les plus communs pour effectuer de la classification d'images par le contenu sont présentés.

Caractéristiques visuelles

Les caractéristiques visuelles sont directement extraites des informations contenues dans les pixels. Le but est d'extraire une information de plus haut niveau que ce que contient un seul pixel en étudiant des groupes de pixels. Plusieurs types de caractéristiques peuvent être extraites : la couleur, la texture [Haralick *et al.* 1973, Viola & Jones 2001, Lowe 1999] ou la forme [A. Foulonneau *et al.* 2009, Belongie *et al.* 2000, Belongie *et al.* 2001, Belongie *et al.* 2002] par exemple.

Ces caractéristiques sont calculées sur un seul pixel ou un petit groupe de pixels. Pour construire une représentation significative des images, il est nécessaire d'extraire ces caractéristiques en plusieurs points de chaque image. Deux méthodes d'extraction existent : l'extraction dense et l'extraction éparse.

L'extraction dense consiste à extraire un vecteur de caractéristiques pour chaque point sur une grille régulière. La finesse de la grille peut aller jusqu'à ce qu'un vecteur de caractéristiques soit extrait pour chaque pixel de l'image. Le nombre de vecteurs de caractéristiques extrait est donc très important.

L'extraction éparse consiste à détecter des points d'intérêt dans l'image et extraire un vecteur de caractéristiques uniquement pour ces points. Bien que générant moins de vecteurs de caractéristiques que l'extraction dense, Il y en a toujours beaucoup.

Étant donné le nombre de vecteurs de caractéristiques visuelles, il est nécessaire d'effectuer un traitement pour construire une représentation plus compacte des images. En effet, comparer des images représentées par des milliers de vecteurs est long, et avec les volumes d'images à traiter, cette représentation n'est pas raisonnable.

Représentation des Images

Le but d'une représentation d'images est d'être compacte, afin de faciliter son traitement et de permettre de traiter les volumes d'images des bases récentes, tout en rapprochant les images sémantiquement similaires. De nombreuses représentations d'images existent [Deselaers *et al.* 2004]. Le système présenté dans cette thèse est construit au-dessus des représentations d'images et de manières orthogonales à ses dernières, il peut donc utiliser n'importe quelle représentation d'images. Parmi les plus courantes on compte les dictionnaires visuels [Sivic & Zisserman 2003, van Gemert *et al.* 2008, Boureau *et al.* 2010] qui s'inspirent des représentations de documents en recherche de textes par exemple. Ou encore les modèles génératifs et score de Fisher [Perronnin *et al.* 2010] qui considèrent les descripteurs comme des variables aléatoires issues d'une loi probabiliste et représente une image par son écart à cette loi. Plus récemment, l'apprentissage profond a été utilisé pour créer des représentations d'images performantes [Zeiler *et al.* 2011, Bengio *et al.* 2013].

Ce ne sont ici que quelques représentations d'images possibles. Nous utilisons la représentation proposée par [Perronnin *et al.* 2010] dans nos expérimentations car elle présente de bonnes performances et nous a été fournie par les auteurs. Grâce à la générosité de [Perronnin *et al.* 2010], quelques mois de travail et de calculs ont pu être économisés.

Mesures de Similarité

La mesure de similarité utilisée dépend énormément de la représentation des images. Les distances les plus simples comme la distance l_1 (équation 1.2) et l_2 (équation 1.3) peuvent être utilisées. Mais celles-ci ne sont pas adaptées à toutes les représentations d'images. De nombreuses représentations, telles que les dictionnaires visuels, sont des histogrammes. Plusieurs mesures dédiées aux histogrammes sont donc utilisées. La forme quadratique [Hafner *et al.* 1995] (équation 1.4), ou la distance χ^2 (équation 1.5), qui a déjà été utilisée en recherche d'images par le contenu [Schiele & Crowley 1996, Ling & Jacobs 2007, Martin *et al.* 2004], en sont deux exemples.

Quelques fois, les sacs de caractéristiques sont utilisés comme description, il convient alors d'avoir des mesures pour les comparer. Beaucoup de distances existent [Grauman & Darell 2005, Charikar 2002] et elles sont souvent basées sur des notions de voisinage, si des vecteurs de caractéristiques de deux images sont voisins, les images se ressemblent.

Enfin, les classifieurs linéaires utilisent parfois l'astuce du noyau pour évaluer la similarité entre deux images. L'astuce du noyau consiste à projeter les vecteurs de représentation des images dans un espace de plus grande dimension, où la similarité peut être mesurée grâce à un produit scalaire. La projection doit se faire à l'aide d'une injection dans un espace de Hilbert alors appelé *espace de redescription*. L'espace dans lequel sont exprimées les descriptions d'image est lui appelé *espace d'entrée*. Le produit scalaire dans l'*espace de redescription* est alors une

fonction noyau. La particularité de ces fonctions est qu'elles sont aussi définies comme des fonctions semi-définies positive de l'*espace d'entrée*, c'est-à-dire respectant l'équation 1.8. N'importe quelle fonction définie positive est un noyau, et donc un produit scalaire dans un espace de plus grande dimension, même si celui-ci n'est pas connu. Il n'est alors pas nécessaire de connaître ni de calculer la fonction de transformation pour calculer le produit scalaire dans l'*espace de redescription*.

Les algorithmes de classification linéaire ne reposant que sur le produit scalaire peuvent alors utiliser de telles fonctions pour calculer facilement une séparation linéaire dans un espace de plus grande dimension. Des données non linéairement séparables dans l'*espace d'entrée* ont plus de chance d'être linéairement séparable dans l'*espace de redescription* comme le montre la figure 1.8. L'astuce du noyau permet donc d'améliorer les performances des classifieurs linéaires utilisant uniquement le produit scalaire.

Algorithmes de Classification d'Images Classiques

Les représentations d'images sont construites de façon à ce que les images sémantiquement proches soient aussi proches dans l'espace de description. C'est donc naturellement que les premières méthodes utilisées pour classer les images sont basées sur des voisinages. Ainsi, la recherche d'images par rapport à une requête est faite en recherchant les k plus proches voisins et la classification se fait en utilisant des algorithmes dérivés des k-moyennes [Chang *et al.* 2012].

Cependant, les représentations d'images ne sont pas parfaites et ne permettent pas de séparer toutes les images ne représentant pas le même concept, ou de rassembler toutes les images représentant le même concept. Alors que certains cherchent à améliorer la qualité des représentations d'images et des mesures de similarité, d'autres cherchent à adapter les systèmes de recherche à ces problèmes. Ainsi, les réseaux de neurones artificiels ont été utilisés récemment pour effectuer de la classification d'images, en utilisant principalement des représentations d'images créées avec de l'apprentissage profond [Wan *et al.* 2013]. Ces méthodes présentent de bons résultats, mais des techniques récentes utilisant des machines à vecteurs de support ont obtenu de meilleurs résultats en utilisant ce genre de représentations d'images [Tang 2013]. Les machines à vecteurs de support restent donc bien compétitives et sont l'une des meilleures techniques de classification d'images. Détaillons un peu leur fonctionnement pour mieux comprendre leurs limitations.

Machines à Vecteurs de Support

Les machines à vecteur de support sont des classifieurs binaires linéaires introduits en 1995 par [Cortes & Vapnik 1995]. Ils ont été utilisés dans la recherche d'images par le contenu à plusieurs reprises [Gorisse *et al.* 2010, Perronnin *et al.* 2010, Hoi *et al.* 2008] et ont été présentés comme une bonne méthode de classification de manière générale [Fernández-Delgado *et al.* 2014].

En tant que classifieurs binaires linéaires, les machines à vecteurs de support

cherchent un hyperplan séparant les données en deux classes. Mais, contrairement à la majorité des autres classifieurs binaires linéaires, les machines à vecteur de support ne se contentent pas de chercher n'importe quel hyperplan séparateur (figure 1.9), elles recherchent l'hyperplan séparateur à plus large marge (figure 1.10), c'est-à-dire présentant la plus grande distance avec les données les plus proches. Cet hyperplan est unique et présente de meilleures capacités de généralisation que les autres hyperplans.

La recherche de cet hyperplan est un problème quadratique dépendant du nombre de données n utilisées pour l'apprentissage, donnant une complexité finale de $\mathcal{O}(n^3)$. Seul le produit scalaire est utiliser dans le calcul de l'hyperplan, l'astuce du noyau peut donc être utilisée pour résoudre des problèmes de classification non linéaires dans l'espace des représentations d'images, ou *espace d'entrée*.

Enfin, les machines à vecteurs de support sont des classifieurs binaires, or de nombreux problèmes de classification d'images sont composés de plusieurs classes. Afin de résoudre des problèmes de classification multi-classes, plusieurs machines à vecteurs de support sont utilisées conjointement. Deux techniques principales se côtoient : le *un contre tous* et le *un contre un*.

Dans le cas du *un contre un*, une machine est apprise par couple de classe à identifier, elle est entraînée en utilisant uniquement les images issues de ces deux classes et se charge de séparer ces deux classes. Pour connaître la classe d'une nouvelle image, la décision de chaque machine est calculée, ajoutant un point à la classe décidée à chaque fois. La classe obtenant le meilleur score est la classe à laquelle l'image appartient.

Pour le *un contre tous*, une machine est apprise pour chaque classe à identifier, elle est apprise avec l'ensemble des images d'apprentissage, et doit séparer une classe en particulier de toutes les autres classes. Pour déterminer la classe d'une nouvelle image on calcule son score avec chaque machine, celle donnant le meilleur score décide de la classe de la nouvelle image.

Ces algorithmes ont été utilisés à plusieurs reprise en recherche d'images par le contenu, mais avec la taille grandissante des bases de d'images et par conséquent du nombre d'images d'apprentissage disponibles, les machines à vecteurs de support ne peuvent plus être apprises avec l'ensemble des images d'apprentissage disponibles. Le temps d'apprentissage serait trop long. Il est donc nécessaire d'utiliser un sous-ensemble des images disponibles pour l'apprentissage. Cet ensemble est construit soit de façon aléatoire, soit itérativement en utilisant des méthodes d'apprentissage actives. Pour les machines à vecteurs de support, les méthodes actives consistent le plus souvent à sélectionner les images les plus proches de l'hyperplan séparateur, c'est-à-dire les plus incertaines. Ces méthodes présentent de bonnes performances, mais ne permettent pas de résoudre tous les problèmes et peuvent être un frein aux performances des machines à vecteurs de support dans certains cas.

Défis

La classification à grain fin consiste à séparer dans différentes classes des images semblables mais ne représentant pas le même concept (figure 1.11), et vice-versa (figure 1.12). Ce problème apparaît de plus en plus avec l'accroissement du volume d'images disponibles sur Internet. Du fait de son apparition liée à l'accroissement du nombre d'images, ce problème est souvent lié au problème de passage à l'échelle, c'est-à-dire de la capacité du système à traiter un grand volume d'images. Dans ces conditions, il est nécessaire d'apprendre la machine à vecteurs de support avec seulement une sous-partie des données d'apprentissage disponibles.

Or les techniques utilisées pour les machines à vecteurs de support sont soit aléatoires, soit basées sur des recherches locales autour de la décision de la machine à vecteurs de support. Bien que les techniques de recherches locales soient le plus souvent efficaces, elles peuvent rencontrer des difficultés dans le cas de la classification à grain fin. En effet, dans ces conditions, les images ne sont pas toujours linéairement séparables dans l'*espace de redescription*. Les recherches locales peuvent alors permettre de retrouver une partie de la classe recherchée mais échouer complètement à retrouver l'ensemble de la classe. Un exemple est donné dans la figure 1.13, où les champignons sont recherchés. Le meilleur classifieur linéaire dans l'*espace de redescription* est représenté en (b) et ne fait une erreur que pour une image de champignon. En revanche, un classifieur appris itérativement en commençant avec quelques images de champignons orange et des images proches finirait avec une décision proche de celle représentée en (c) sur la figure, qui est loin de la décision optimale et échoue complètement à retrouver les champignons blancs. Pour résoudre ce problème, d'autres techniques de sélection des images d'apprentissage doivent être considérées. Les méthodes actuelles semblent manquer d'exploration. Nous nous intéressons donc aux algorithmes évolutionnaires qui possèdent une bonne capacité d'exploration dans un temps raisonnable Ils sont présentés dans le chapitre suivant, qui détaille en quoi l'exploration et les algorithmes évolutionnaires (AEs) peuvent aider à résoudre le problème de la classification à grain fin.

Le Manque d'Exploration Résolu par l'Utilisation d'Algorithmes Évolutionnaires

Les représentations d'images regroupent les images semblables et séparent les images dissemblables. Ainsi, quelle que soit la représentation d'images, on pourra trouver un cas où des images appartenant à la même classe ne sont pas au même endroit de l'espace de recherche, comme représenté sur la figure 2.1. Le processus d'un système d'apprentissage actif pour une machine à vecteurs de support utilisé sur cet exemple est présenté sur la figure 2.2. On voit qu'à l'issu de l'apprentissage, aucun champignon blanc n'a été retrouvé, ils étaient trop loin de la décision de la machine pour être ajoutés à l'ensemble d'apprentissage. Une méthode sélectionnant les éléments d'apprentissage de manière moins locale aurait permis d'identifier un champignon blanc et ainsi d'améliorer les performances de classification de la ma-

chine à vecteurs de support. De fait, ajouter de l'exploration au processus permettrait d'améliorer les performances du système. Les AE proposent de telles capacités exploratoires, nous avons donc décidé de les utiliser.

Algorithmes Évolutionnaires

Les AEs sont des algorithmes utilisant une population de solutions potentielles à un problème qui évolue itérativement pour atteindre les solutions optimales à ce problème. Le processus général d'un AE est présenté sur la figure 2.5. Il existe de nombreux type d'algorithmes évolutionnaires : les essaims de particules [Kennedy & Eberhart 1995], les stratégies d'évolution [Rechenberg 1973, Hansen *et al.* 1995], l'évolution différentielle [Storn & Price 1997, Fonlupt *et al.* 2011, Segura *et al.* 2015b, Segura *et al.* 2015a] ou les algorithmes de lucioles [Yang 2008]. Il existe de nombreux autres AEs, mais nous nous sommes intéressés tout particulièrement aux Algorithmes Génétiques (AGs). Leur processus général, qui est relativement simple, est présenté sur la figure 2.6. Voyons comment ces algorithmes peuvent nous aider à résoudre le problème de la classification d'images à grain fin.

Les Algorithmes Évolutionnaires pour Résoudre la Classification d'Images à Grain Fin

Un certain nombre de travaux ont utilisés des AEs pour traiter des problèmes de recherche d'images par le contenu. En particulier, la structure itérative des AEs permet de les utiliser facilement pour construire des méthodes interactives, dans lesquels l'utilisateur contribue à l'évaluation des solutions potentielles. Ainsi, des techniques basées sur les essaims de particules [Broilo & De Natale 2009] ou les AGs [Lai & Chen 2011] ont été développées. Ces systèmes sont le plus souvent testés sur de petites bases d'image (1000 images pour la base utilisée par [Lai & Chen 2011]), l'intervention de l'utilisateur limitant le nombre d'évaluations que l'on peut demander. Dans ces systèmes, les individus de la population sont exprimés sous forme de vecteurs ayant la même forme que la représentation des images. Cependant, ils ne représentent pas nécessairement un vecteur correspondant à une image existante. En effet, de par l'évolution des individus durant l'exécution de l'algorithme et parce que les images représentent un espace défini extensivement, tous les vecteurs ne correspondent pas à une image existante. Une opération d'assignation est nécessaire pour ramener les individus sur des images réelles. [Johnson 2012] traite de l'élaboration d'opérateurs de croisement et de mutation dans le cadre d'espaces définis extensivement, formalisant les idées déjà utilisées dans d'autres systèmes.

Ces approches cependant ne traitent pas de classification, et plus particulièrement de la classification d'images à grain fin que nous cherchons à résoudre. Voyons de quel type de problème relève la classification à grain fin, et quels AGs ont été développés pour résoudre ces problèmes.

Le Problème d'Optimisation Lié à la Classification à Grain Fin

Dans la classification à grain fin, des images appartenant à la même classe se retrouvent séparées en plusieurs endroits de l'espace de recherche, plus ou moins éloignés, comme représenté sur la figure 2.1 pour les champignons. Identifier une classe consiste à retrouver l'ensemble de ces images, c'est à dire retrouver l'ensemble des maxima de la fonction donnant la pertinence d'une image par rapport à la classe à identifier. Ces maxima sont obtenus pour différentes images, c'est-à-dire différents vecteurs de caractéristiques, parfois éloignés dans l'espace de représentation. C'est ce qu'on appelle un problème multi-modal.

Une grande majorité des AGs dédiés aux problèmes multi-modaux utilisent la multi-objectivisation pour résoudre ces problèmes. En effet, les problèmes multi-objectifs sont étudiés depuis longtemps dans le domaine, et leur résolution implique en général l'identification d'un ensemble diversifié de solutions. Les problèmes multi-modaux nécessitant eux aussi une diversité de solutions, tirer profit des avancées dans le domaine des algorithmes multi-objectifs semble être une bonne idée. De ce fait nous présentons les concepts liés aux algorithmes multi-objectifs avant d'introduire une étude comparative de trois AGs dédiés aux problèmes multi-modaux qui permet de mesurer leur intérêt dans le cadre de la classification d'images à grain fin.

Problèmes Multi-Objectifs Un problème multi-objectif est un problème d'optimisation composé de plusieurs fonctions qui doivent être optimisées conjointement. La définition 1 en donne une formalisation dans le cadre d'une minimisation, mais cela peut aisément se réécrire pour une maximisation. Dès lors que l'on a plusieurs critères à optimiser conjointement, il est difficile de dire si un individu est meilleur qu'un autre. En effet, il peut être meilleur pour l'un des critères mais pas pour l'autre par exemple. La dominance de Pareto est alors utilisée pour évaluer si un individu est meilleur qu'un autre, elle est définie par la définition 2. Cette relation ne permet pas de comparer tous les individus, en effet, si un individu est meilleur qu'un autre sur un critère et plus mauvais sur un autre, les deux individus sont dits incomparables et sont considérés comme étant équivalents.

Le but d'un algorithme de résolution de problèmes multi-objectifs est de trouver l'ensemble des éléments non-dominés de l'espace de recherche, appelé ensemble de Pareto optimal. L'image de cet ensemble par le problème multi-objectif F est appelé le front de Pareto optimal. Cet ensemble peut être infini, et contient dans la plupart des cas trop d'éléments pour qu'ils soient tous retrouvés par l'AE. Il est alors attendu de l'algorithme qu'il trouve un ensemble diversifié de solutions caractérisant au mieux le front de Pareto optimal.

Les algorithmes évolutionnaires ont besoin de trier les individus. La dominance de Pareto permet de faire un premier tri, appelé le tri par non-dominance. Les individus sont triés par rang, les individus non-dominés de la population forment le premier rang, ceux qui sont non-dominés dans les individus restant forment le deuxième rang et ainsi de suite. La figure 2.12 illustre un tel classement dans le cadre

d'un problème de minimisation à deux objectifs. Ensuite, pour définir un classement au sein d'un même rang, des mesures de diversité sont utilisées. Ainsi, la *crowding distance* [Deb *et al.* 2000] ou la mesure de l'hyper-volume [Auger *et al.* 2009] sont utilisées dans ce sens. Les individus participant le plus à la diversité du rang auquel ils appartiennent sont alors considérés comme meilleurs que les autres individus du même rang. Cela permet de garantir une certaine diversité dans les points identifiés sur le front de Pareto, ce qui est nécessaire à la bonne résolution d'un problème multi-objectif.

Beaucoup d'algorithmes ont été développés sur ces bases pour résoudre des problèmes multi-objectifs, un certain nombre sont listé dans [Zitzler *et al.* 2000]. NSGA-II [Deb *et al.* 2000] est un algorithme génétique classique dans le domaine du multi-objectif présentant de bonnes performances. Voyons maintenant comment les AEs sont utilisés pour résoudre les problèmes multi-modaux qui nous intéressent plus particulièrement.

Algorithmes Évolutionnaires pour l'Optimisation Multi-Modale La plupart des AEs dédiés aux problèmes multi-modaux utilisent la multi-objectivisation pour résoudre ces derniers. La multi-objectivisation consiste à transformer un problème simple objectif difficile à résoudre en un problème multi-objectif plus facile à résoudre et dont les solutions sont aussi solutions du problème initial. Ainsi, la résolution du problème multi-objectif permet d'identifier au moins un sous-ensemble des solutions du problème initial.

Les AEs ont tendances à converger vers une unique solution de manière générale, or la résolution d'un problème multi-modal nécessite d'identifier plusieurs solutions. En conséquence, la multi-objectivisation se fait la plupart du temps en ajoutant un objectif promouvant la diversité des individus dans l'espace de recherche afin d'être capable d'identifier plusieurs solutions au problème multi-modal au lieu d'une seule.

Trois AEs récents dédiés aux problèmes multi-modaux sont étudiés dans [Pighetti *et al.* 2015a]: PNA-NSGA-II [Bandaru & Deb 2013], MO-BiDE [Basak *et al.* 2013], MOMMOP [Wang *et al.* 2014]. Ces trois algorithmes utilisent la multi-objectivisation puis des algorithmes dédiés aux problèmes multi-objectifs qui ont été modifiés pour être plus adaptés à la résolution du problème multi-modal. Ils sont donc comparés à l'algorithme multi-objectif NSGA-II [Deb *et al.* 2000] classique utilisant les mêmes objectifs que PNA-NSGA-II pour étudier l'influence de ces adaptations sur les résultats finaux. Cette étude est menée sur les fonctions de référence proposées dans la compétition multi-modal de CEC2013 [Bandaru & Deb 2013]. Les résultats montrent que la version originale de NSGA-II n'est pas adaptée à la résolution des problème multi-modaux, même après multi-objectivisation. NSGA-II ne retrouve qu'un seul optima pour chacune des fonctions. Les adaptations apportées dans les algorithmes dédiés aux problèmes multi-modaux sont donc utiles et nécessaires pour résoudre correctement ces problèmes. Cependant, lorsque la dimension de l'espace de recherche augmente, les performances des trois algorithmes s'effondrent. Dans un espace à 10 dimensions

sur un problème liant plusieurs difficultés, les performances sont médiocres pour tous les algorithmes.

Les représentations d'images étant des vecteurs de grande dimensions (plus de 100 dimensions voire même quelques milliers de dimensions), il devient évident que ces algorithmes ne pourront pas résoudre le problème de la classification d'images à grain fin seuls. C'est pourquoi nous étudions les systèmes hybrides, combinant plusieurs algorithmes pour tirer parti des forces de chacun.

Systèmes Hybride pour la Recherche d'Images par le Contenu

Des systèmes hybrides ont été développés pour résoudre des problèmes de recherche d'images par le contenu ces dernières années. Nous nous intéressons plus particulièrement aux systèmes combinant des AEs avec des méthodes de recherche locales, et aux systèmes combinant des AEs avec des machines à vecteurs de support. Les machines à vecteurs de support ont de très bonnes performances de classification, et nous espérons que les combiner avec des AEs permettra de pallier le manque d'exploration caractéristique des méthodes qui y sont liées.

Algorithmes Évolutionnaires et Voisinage [Lakdashti & Ajorloo 2011] utilise un algorithme des k-moyennes modifié pour construire des règles de classement des images basées sur les retours utilisateurs. Les images sont classées en deux catégories, pertinentes et non pertinentes ; un algorithme de classification d'images a été exécuté au préalable pour avoir une première classification. L'algorithme des k-moyennes est alors utilisé sur chacune des classes identifiées séparément. Chaque groupe d'images identifié est caractérisé par l'hypercube l'englobant. Chaque hypercube est associé à une classe et défini par ses bornes dans chaque dimension, formant ainsi des règles d'appartenance. Ces hyper-cubes sont les individus d'un AG qui les fait évoluer en fonction des retours utilisateurs pour améliorer la précision des règles. Des tests sur une base de 10 004 images réparties en 28 catégories montrent que cette méthode obtient de meilleurs résultats que des machines à vecteurs de support.

Une combinaison d'un AG multi-objectif (NSGA-II) et de plusieurs méthodes de recherche locale est proposée par [Arevalillo-Herráez *et al.* 2013]. À partir d'une requête, des images sont présentées à l'utilisateur pour évaluation. Ces images sont annotées comme pertinentes ou non pertinentes. NSGA-II est ensuite utilisé avec autant d'objectifs que d'images annotées comme pertinentes, le but étant de minimiser la distance à chacune de ces images. À l'issue du processus d'évolution, qui est maintenu à une durée inférieure à 1 seconde, les individus de la population plus proche d'une image non-pertinent que d'une image pertinente sont supprimés. Les autres servent de point de départ pour une recherche locale, qui sert ensuite à classer les images de la base par ordre décroissant de pertinence. Testé sur des bases allant de 1 500 à 30 000 images, ce système s'est révélé plus performant que les méthodes de recherche locale seules.

Ces systèmes nous montrent qu'utiliser une méthode d'exploration en plus d'une

méthode de recherche locale améliore les performances dans le cadre de la recherche d'images par le contenu. Les machines à vecteurs de support sont parmi les meilleurs algorithmes de classification dans le domaine, et nous avons vu que la plupart des systèmes basés sur ces algorithmes manquent d'exploration. Les combiner à des algorithmes évolutionnaires semble alors être une bonne idée.

Algorithmes Évolutionnaires et Machine à Vecteurs de Support [Wang *et al.* 2005] utilise un AG interactif avec une machine à vecteurs de support pour effectuer de la recherche d'images. 12 images aléatoires sont présentées à l'utilisateur et forme la population initiale de l'AG, qui utilise les représentations d'images comme génome. L'utilisateur annote les images entre 0 et 1 selon leur pertinence par rapport à un sentiment qu'il recherche dans les images. Ces notes sont utilisées d'un côté comme fitness pour l'AG, et de l'autre les images annotées sont utilisées comme ensemble d'apprentissage pour une machine à vecteurs de support. L'AG et la machine à vecteurs de support sont exécutés en parallèle. Les 8 images de la base les plus proches d'un individu de l'AG sont sélectionnées pour faire partie de la prochaine itération avec les 3 images ayant le meilleur score donné par la machine à vecteur de support. Les 12 nouvelles images sont montrées à l'utilisateur et le processus recommence ainsi jusqu'à ce que 80% des images montrées à l'utilisateur soient pertinentes. Ainsi, la machine à vecteurs support est utilisée pour insérer de nouveaux individus dans la population et stimuler la convergence de l'AG. Ce système est testé sur une base de 2786 images. Les performances sont intéressantes mais des adaptations sont nécessaires, notamment au niveau de la recherche des images les plus proches des individus de l'AG qui peut prendre beaucoup de temps avec de grosses bases d'images.

[Shi *et al.* 2007] utilise une approche similaire pour de la recherche de visage spécifique dans une base d'images de visages. L'AG n'utilise qu'un seul objectif : minimiser la distance aux images annotées comme pertinentes par l'utilisateur.

[Yu *et al.* 2016] ont une méthode similaire mais les auteurs utilisent l'évolution différentielle plutôt qu'un AG. La fitness utilisée est alors une somme pondérée de la distance de l'individu à l'image requête et du score donnée par la machine à vecteurs de support (représentant une approximation de la note donnée par l'utilisateur). L'utilisation d'une somme pondérée plutôt que de deux objectifs séparés empêche la convergence vers un ensemble diversifié d'images. Cette méthode a été testée sur la même base de 1 000 images que [Lai & Chen 2011] qui utilise seulement un AG.

Récemment, [Kanimozhi & Latha 2015] ont utilisé les algorithmes des lucioles (*Firefly Algorithms*) avec des machines à vecteurs de support. Ils utilisent aussi un seul objectif, qui consiste en une somme d'un terme cherchant à s'approcher des images annotées comme pertinentes et d'un terme cherchant à s'éloigner des images non-pertinentes. La boucle d'évolution de l'AE est effectuée une seule fois entre deux annotations, ce qui signifie que l'algorithme évolue très peu entre deux interventions de l'utilisateur. Cela peut être problématique et engendrer une fatigue de l'utilisateur. De plus, l'utilisation d'un seul objectif entraîne la convergence de

l'AE vers une seule solution, ce qui peut devenir problématique pour résoudre le problème de la classification à grain fin.

L'utilisation d'AEs avec des machines à vecteurs de support semble être une bonne piste à la vue de ces travaux. Seulement, la plupart utilisent la machine à vecteurs de support pour améliorer les performances et la convergence de l'AE, et aucune de ces méthodes n'effectue une tâche de classification. Un premier système de recherche d'images utilisant un AG pour fournir des éléments d'apprentissage à une machine à vecteurs de support a été proposé [Pighetti *et al.* 2012] à partir des travaux de [Lai & Chen 2011]. Les deux éléments composant la fitness dans [Lai & Chen 2011] ont été séparés en deux objectifs distincts : l'un étant l'évaluation de l'utilisateur, l'autre la distance à l'image requête. À chaque itération, les meilleurs individus de la population de l'AG sont associés à l'image de la base ayant la représentation la plus proche. Ces images sont ajoutées à l'ensemble d'apprentissage de la machine à vecteurs de support après avoir été annotées par l'utilisateur. La machine à vecteurs de support est ensuite utilisée pour ordonner les images de la base et les présenter à l'utilisateur. Si l'utilisateur n'est pas satisfait, une autre boucle d'évolution est effectuée et le processus recommence. Ce système a obtenu de meilleures performances que celui de [Lai & Chen 2011] et nous a encouragé à développer le système hybride de classification dédié à la classification à grain fin présenté dans la suite.

Un Système Hybride pour la Classification d'Images à Grain Fin

Le système proposé est un système de classification basé sur une machine à vecteurs de support. Le nombre d'images disponible pour l'apprentissage de la machine étant trop grand, il faut en sélectionner un sous ensemble. Nous proposons d'effectuer cette sélection en utilisant un AE, profitant ainsi de leur capacité exploratoire pour identifier des zones de l'espace contenant des images de certaines classes qui n'auraient pas été identifiées autrement. La structure du système est présentée dans un premier temps, puis une implémentation du système est introduite.

Structure du Système

Le principe général du système proposé est d'utiliser un AE pour fournir des éléments d'apprentissage à une machine à vecteurs de support. La machine à vecteurs de support ainsi entraînée est alors utilisée pour classifier les images qui lui sont présentées dans les bonnes catégories. Les individus de l'AE ne représentant pas nécessairement une image existant dans la base d'images, une étape d'assignation des individus à des images est nécessaire afin de pouvoir ajouter des images réelles et annotées à l'ensemble d'apprentissage de la machine à vecteurs de support. Le schéma de fonctionnement de ce système est présenté sur la figure 3.1. La partie bleue représente l'algorithme évolutionnaire. La partie verte le processus de

récupération des annotations des images, qui consiste en l'assignation des individus à des images, la sélection des images à annoter et l'annotation des images et leur ajout à l'ensemble d'apprentissage. La partie grise représente la machine à vecteur de support.

Pour démarrer le processus, une image requête de chaque classe à identifier doit être fournie au système. Ces images sont issues de l'ensemble des images disponibles pour l'apprentissage. Elles servent à l'initialisation de la population de l'AE. La machine à vecteurs de support étant utilisée lors des étapes d'évaluation de l'AE, il est nécessaire qu'elle soit initialisée avant que l'AE ne puisse continuer son exécution. La population de l'AE est donc extraite pour être traitée par le processus d'annotation des images. Chaque individu de la population est associé à une image de l'ensemble des images d'apprentissage. Certaines de ces images sont choisies pour être ajoutées à l'ensemble d'apprentissage de la machine à vecteurs de support qui est alors entraînée. La machine à vecteurs de support est entraînée chaque fois que l'ensemble d'apprentissage est modifié.

Une fois la machine à vecteurs de support entraînée, l'algorithme évolutionnaire peut reprendre son court, utilisant la machine à vecteurs de support dans l'évaluation des individus afin d'identifier les zones de l'espace contenant les meilleures images à ajouter à l'ensemble d'apprentissage. Il est attendu de l'AE qu'il identifie un ensemble diversifié de zones de l'espace et qu'il explore l'espace pour identifier de nouvelles zones potentiellement intéressantes.

Après g générations, la population de l'AE est de nouveau extraite et traitée par le processus d'annotation des images. Ceci permet d'identifier de nouvelles images intéressantes à ajouter à l'ensemble d'apprentissage, d'en ajouter un certain nombre et ainsi de faire évoluer la décision de la machine à vecteurs de support. L'AE est mis en pause le temps que la machine à vecteurs de support soit entraînée, comme elle est utilisée pendant les étapes d'évaluation des individus, il est nécessaire que la machine soit entraînée avec la dernière version de l'ensemble d'apprentissage pour que l'AE évalue correctement les individus de la population. Comme la décision de la machine a changé, la fonction d'évaluation de l'AE change également, modifiant la position des optimaux dans l'espace, l'AE va donc converger vers de nouvelles zones permettant d'identifier de nouvelles images. Ce processus continue jusqu'à ce que l'ensemble d'apprentissage atteigne la taille souhaitée.

Implémentation

L'implémentation proposée utilise comme représentation d'images les vecteurs de Fisher basés sur des descripteurs SIFT extraits de façon dense proposés par [Perronnin *et al.* 2010]. Ce sont des vecteurs de 2048 dimensions normalisés dont la valeur de chaque composante est comprise entre 0 et 1.

Afin de pouvoir effectuer de la classification multi-classe, la technique du *un contre tous* a été utilisée pour la machine à vecteurs de support. Ainsi, une machine est apprise pour chaque classe et identifie la séparation entre cette classe et le reste des images. Ce format permet de facilement calculer la pertinence d'un individu par

rapport à une classe en particulier, ce qui est utilisé dans l'AE plus loin. La machine à vecteurs de support utilise un noyau linéaire et un paramètre de régularisation C fixé à 1.

Afin d'identifier un ensemble divers de solutions, il a été décidé d'utiliser un problème multi-objectifs. Un problème est défini pour chaque catégorie C_i à identifier et est composé de deux fonctions : la première est la distance de l'individu à l'image requête Q_i fournie pour cette catégorie qui doit être maximisée, la deuxième est la valeur absolu du score obtenu par l'individu avec la machine à vecteurs de support SVM_i qui doit être maximisée. Ces fonctions sont exprimées dans l'équation 3.3. La première fonction permet de s'éloigner de la requête et des images ajoutées à l'ensemble d'apprentissage après l'initialisation. La deuxième fonction encourage l'AE à explorer des zones loin de la décision de la machine à vecteurs de support, à la recherche d'éléments mal classés qui, une fois ajoutés à l'ensemble d'apprentissage, vont grandement améliorer la décision prise par la machine à vecteurs de support.

L'AE utilisé pour résoudre ce problème est NSGA-II. Autant de sous-populations indépendantes sont initialisées que de catégories à identifier. Chaque sous-population travaille sur un problème en particulier. C'est comme si une instance de NSGA-II était exécutée pour chacun des problèmes à résoudre. Un individu représentant une image dans l'AE, il a été décidé que le génome des individus aurait la même forme que la représentation d'une image : un vecteur de 2048 dimensions dont la valeur de chaque composante est comprise entre 0 et 1. Chaque sous-population contient 30 individus. L'initialisation de chaque sous-population P_i se fait de manière aléatoire en utilisant une loi normale centrée sur Q_i. Ceci permet d'identifier un certain nombre d'images de chaque classe situées autour de la requête associée. L'AG va ensuite diverger de cette endroit grâce à son premier objectif, explorant l'espace et trouvant de nouvelles images.

L'évaluation se fait grâce à la procédure de tri non-dominé et à la *crowding distance*, de façon usuelle pour NSGA-II. Le croisement utilisé est un croisement à deux points (voir figure 2.8) et la mutation une mutation par réinitialisation avec un taux de mutation de 0,3. La sélection pre-variation (voir figure 3.1 (h)) est un tournoi de taille 2 et la sélection post-variation (voir figure 3.1 (l)) est celle classiquement effectuée dans NSGA-II.

L'AG se voit allouer $g = 30$ générations entre deux extractions de sa population, pour qu'elle soit traitée par la procédure d'annotation des images. Lors de cette procédure, l'assignation de chaque individu à une image est effectuée à l'aide de l'algorithme de hachage localement sensible (Locality Sensitive Hashing - LSH). Cet algorithme permet d'effectuer une recherche rapide de plus proches voisins, permettant d'identifier rapidement une image à assigner à chaque individu. Tous les individus sont assignés à des images différentes les unes des autres. Les fonctions de hachage utilisées sont les projections aléatoires (voir définition 5), le nombre de fonctions de hachage par table k est fixé à 1 et le nombre de tables L fixé à 10. Pour répondre à des contraintes du protocole de test nous obligeant à avoir autant d'images de chaque classe dans l'ensemble d'apprentissage, les individus de chaque sous-population P_i sont assignés à des images de la catégorie C_i uniquement. Pour ce

faire, les images de chaque catégorie ont été mises dans une structure LSH séparée.

Une fois l'assignation effectuée, 15 images sont tirées aléatoirement de chaque ensemble d'images correspondant à chaque sous-population. Ces images sont ajoutées à l'ensemble d'apprentissage et la machine à vecteurs de support est apprise avec le nouvel ensemble d'apprentissage.

L'ensemble des paramètres sont rappelés dans le tableau 3.1.

Performances

Les performances du système proposé sont comparées aux performances de deux autres systèmes : l'un sélectionnant les images d'apprentissage aléatoirement et dont les résultats sont disponibles dans [Perronnin *et al.* 2010] ; l'autre étant inspiré de la méthode de sélection active proposée par [Tong & Chang 2001]. Comme nous comparons les résultats obtenus à ceux de [Perronnin *et al.* 2010], le protocole expérimental utilisé et décrit dans une première partie respecte ce qui est utilisé dans cette publication afin d'avoir des résultats comparables. Une deuxième partie présente les algorithmes comparés. Enfin les résultats obtenus sont présentés.

Protocole d'Expérimentation

[Perronnin *et al.* 2010] évalue les capacités de classification des machines à vecteurs de support sur la base Caltech-256 [Griffin *et al.* 2007] avec des représentations d'images qui sont des vecteurs de Fisher normalisés de 2048 dimensions. La même base d'images et la même description d'images sont utilisées ici, [Perronnin *et al.* 2010] nous ayant fourni les représentations d'images qu'ils ont utilisées.

Afin de comparer nos résultats à la méthode de sélection aléatoire qu'ils utilisent, la même mesure de performance doit être utilisée. Les performances de classification seront donc évaluées en terme d'exactitude moyenne (*average accuracy*) dont le calcul est présenté par l'équation 4.1.

Les machines à vecteurs de support doivent être apprise avec le même volume de données pour que leurs performances soient comparables. En respectant ce qui est fait dans [Perronnin *et al.* 2010], le système proposé et la méthode active sont donc évalués après que 15, 30, 45 et 60 images ont été ajoutées à l'ensemble d'apprentissage de la machine à vecteur de support. La base de données n'est pas séparée en un ensemble de test et un ensemble d'entraînement bien définis, les images d'apprentissage sont donc sélectionnées dans la base sans restriction. Les performances de classifications sont ensuite évaluées sur l'ensemble des images n'ayant pas servi à l'apprentissage.

La méthode d'apprentissage actif, expliquée plus en détail dans la prochaine section, est déterministe, une seule exécution suffit donc pour évaluer ses performances. En revanche, une moyenne sur plusieurs exécutions est nécessaire pour la sélection aléatoire et le système proposé car ils ne sont pas déterministes et leurs performances dépendent de certains processus aléatoires. Pour la sélection aléatoire,

les résultats proposés par [Perronnin *et al.* 2010] sont déjà moyennés. Pour notre système, l'expérience est répétée 10 fois et la moyenne des 10 résultats est présentée.

Algorithmes Comparés

Trois méthodes de sélection des images d'apprentissage sont testées : la sélection aléatoire faite par [Perronnin *et al.* 2010] qui sert de référence; l'implémentation du système proposé, présenté dans le chapitre précédent; une méthode de sélection active inspirée des propositions de [Tong & Chang 2001]. Nous présentons ici un peu plus en détail la méthode de sélection active, les deux autres méthodes sont déjà détaillées par ailleurs.

[Tong & Chang 2001] prouve que, étant donné une machine à vecteurs de support binaire possédant un ensemble d'apprentissage TS, étant donné un ensemble d'images disponibles pour l'apprentissage ATS, l'image de ATS apportant la meilleure espérance d'amélioration de la décision de la machine lorsqu'elle est ajoutée à TS est celle se situant le plus proche de la décision de la machine. Dans le contexte multi-classes étudié ici, il convient d'adapter ce principe. La méthode du *un contre tous* est utilisée pour créer une machine à vecteurs de support multi-classes. C'est-à-dire qu'une machine est apprise pour chaque classe à identifier, chaque machine séparant les images d'une classe en particulier des autres images de la base. Améliorer les performances individuelles de chaque machine devrait permettre d'améliorer les performances globales. En conséquence, l'image la plus proche de la décision de chaque machine peut être sélectionnée pour être ajoutée à l'ensemble d'apprentissage.

Afin de respecter le protocole expérimental, l'ensemble d'apprentissage doit contenir autant d'images de chaque catégorie. De plus, le système proposé ajoute les images par groupe de 15×256 à l'ensemble d'apprentissage, et les images sélectionnées pour améliorer la machine à vecteurs de support SVM_i sont sélectionnées dans la catégorie C_i uniquement. Pour avoir des conditions similaires tout en respectant l'idée de [Tong & Chang 2001], les images ajoutées à TS pour la machine SVM_i sont les 15 images issues de la catégorie C_i les plus proche de la décision de m_i. L'initialisation du système se fait en sélectionnant dans chaque catégorie les images les plus proches de la requête fournie.

Résultats

Les résultats présentés sur la figure 4.5 et dans le tableau 4.1 montrent que l'exactitude moyenne du système proposé est meilleure que celle des deux autres systèmes. Il est rassurant que les performances soient meilleures que la sélection aléatoire, cela prouve que la façon de sélectionner les images d'apprentissage proposée apporte un gain réel de performances, même si elle est plus coûteuse. Le gain est nul à l'initialisation car celle-ci est pseudo aléatoire pour le système proposé et aléatoire pour la sélection aléatoire. Il est donc normal que les performances soient proches à ce moment. En revanche, le gain se fait ressentir dès le premier appel

à la procédure d'annotation d'images après les 30 premières générations de l'AG, gagnant 1,23 points par rapport à la sélection aléatoire. Le gain grandit de plus en plus avec la taille de l'ensemble d'apprentissage, et donc le nombre de générations de l'AG, pour atteindre 3,45 points de mieux à la fin de l'exécution avec 60 images d'apprentissage par catégorie.

Le système de sélection actif proposé présente des performances inférieures à celles de la sélection aléatoire. La comparaison n'est donc pas nécessaire. Ces résultats peuvent être attribués au fait que la généralisation faite de la méthode proposée par [Tong & Chang 2001] en sélectionnant les 15 images les plus proches de la décision au lieu d'une n'est pas bonne. Cette méthode peut ne pas être adaptée aux problèmes multi-classes. De plus, les données ne sont pas nécessairement linéairement séparables dans l'espace de redescription, ce qui est un problème pour les méthodes actives en général.

La méthode proposée montre donc des résultats encourageant, ce qui incite à explorer les capacités de celle-ci plus en profondeur. Ceci dit, il convient d'abord de faire une petite étude de la complexité de ce système. En effet, il est important de savoir la quantité de calculs supplémentaires effectués par ce système par rapport aux autres. Il convient aussi de savoir si la complexité de ce système dépend fortement de la taille de la base ou non, afin de savoir si la classification de plus grosses bases d'images est envisageable.

Comparaison de la Complexité

Nous séparons l'analyse de la complexité en deux parties : une première partie analysant la complexité des processus préliminaires (extraction de la représentation des images, etc), une deuxième partie analysant la complexité de l'étape d'apprentissage.

Complexité des Étapes Préliminaires

Pour la sélection aléatoire ou le système de sélection active, la seule étape préliminaire nécessaire est la construction de la représentation de chaque image de la base. Le système proposé doit lui aussi extraire la représentation de chaque image, mais il doit de plus construire la structure de hachage pour le LSH. La complexité de la construction de cette structure dépend du nombre d'images disponibles pour l'apprentissage n_{ATS}, du nombre de tables dans la structure L et du nombre de fonctions de hachage par table k. Elle est exprimée par l'équation 4.2. Le parcours de l'ensemble des images est nécessaire pour extraire la représentation des images, si les paramètres du LSH ne nécessitent pas l'étude des représentations pour être fixés, le hachage peut se faire en même temps que l'extraction des descriptions, réduisant l'impact de cette étape. De plus, l'extraction des représentations d'images est beaucoup plus complexe que le hachage. En conséquence, bien qu'introduisant une étape supplémentaire dans les calculs préliminaires, l'augmentation de la complexité des calculs préliminaires dans le système proposé n'est pas très importante.

Complexité de la Phase d'Apprentissage

On note n_{TS} le nombre d'images dans l'ensemble d'apprentissage, n_{ATS} le nombre d'images disponibles pour l'apprentissage (pouvant être ajoutées à l'ensemble d'apprentissage), B le nombre de catégories dans la base d'images, SVM_{eval} la complexité liée à l'évaluation d'une image par une machine à vecteurs de support, $SVM_{train}(x)$ la complexité d'entraînement d'une machine à vecteurs de support avec x images dans l'ensemble d'apprentissage qui est de l'ordre de $\mathcal{O}(x^3)$. On considère que chaque catégorie de l'ensemble des images disponibles pour l'apprentissage contient en moyenne le même nombre d'images : $\frac{N_{ATS}}{B}$

Sélection Aléatoire La sélection de n_{TS} images aléatoirement parmi n_{ATS} ne dépend que du nombre d'images à sélectionner. Une fois les images choisies, il faut apprendre la machine à vecteurs de support avec celles-ci. Nous sommes dans un cas multi-classes, en utilisant la stratégie *un contre tous*, c'est donc B machines à vecteurs de support qui sont apprises avec les n_{TS} images sélectionnées. Cela donne la complexité exprimée dans l'équation 4.3.

Sélection Active La sélection active construit itérativement l'ensemble d'apprentissage pour la machine à vecteurs de support. À chaque itération, n_{add} images sont ajoutées à l'ensemble d'apprentissage. La machine à vecteurs de support utilise toujours la stratégie du *un contre tous*, on a donc B machines binaires. Pour chaque catégorie C_i, les $\frac{n_{add}}{B}$ images les plus proches de la décision de SVM_i sont ajoutées à l'ensemble d'apprentissage. Pour se faire, les images de chaque catégorie C_i doivent être évaluées par SVM_i puis triées en fonction de leur score. Chaque image de l'ensemble des images disponibles pour l'apprentissage est donc évaluée par une machine à vecteurs de support, ce qui a une complexité de $\mathcal{O}(n_{ATS} \times SVM_{eval})$, puis chaque catégorie est triée séparément, ce qui a une complexité de $\mathcal{O}\left(B\frac{n_{ATS}}{B}\log\left(\frac{n_{ATS}}{B}\right)\right)$.

Une fois les images sélectionnées, elles sont ajoutées à l'ensemble d'apprentissage de la machine à vecteurs de supports qui est alors apprise avec $i \times n_{add}$ images d'entraînement à la i$^{\text{ème}}$ itération. Comme la machine est composée de B machines binaires, l'entraînement a une complexité de $\mathcal{O}\left(B \times SVM_{train}(i \times n_{add})\right)$.

L'ensemble des opérations est alors répété autant de fois que nécessaire pour atteindre n_{TS} images dans l'ensemble d'apprentissage, c'est-à-dire $\frac{n_{TS}}{n_{add}}$ fois. Cela donne la complexité exprimée dans l'équation 4.4. On la simplifie en supposant que le nombre d'itérations est faible, et donc que la complexité de la somme des apprentissages de la machine à vecteurs de support peut être approximé par la complexité de l'apprentissage de la machine avec le plus gros ensemble d'apprentissage. n_{ATS} est aussi considéré comme étant très grand, le terme $\frac{n_{ATS}}{B}$ prédomine donc devant la complexité de l'évaluation d'un individu par une machine à vecteurs de support SVM_{eval} qui n'est que le calcul d'une fonction noyau. La complexité simplifiée est exprimée par l'équation 4.5.

Méthode Hybride Proposée La méthode hybride proposée utilise un AG pour aider à la sélection des images à ajouter à l'ensemble d'apprentissage. L'AG effectue g générations avant que le processus d'annotation des images ne soit exécuté. L'AG utilisé est NSGA-II. Les étapes de croisement et de mutation des individus ont une complexité négligeable devant l'évaluation et le tri des individus lors de l'étape de sélection post évaluation. La complexité du tri par non dominance est de $\mathcal{O}\left(m \times n_{ind}^2\right)$, avec n_{ind} le nombre d'individus à classer et m le nombre de fonctions de fitness. L'algorithme génétique est composé de B sous-populations indépendantes de n_{ind} individus dans notre cas. La complexité de g générations est donc de $\mathcal{O}\left(g \times B \times m \times n_{ind}^2\right)$.

Vient ensuite le processus d'évaluation des individus. Il consiste d'abord à assigner chaque individu à une image en utilisant la structure LSH. La recherche d'un voisin en utilisant LSH a une complexité de $\mathcal{O}(n_{ATS}^{\frac{1}{c}})$, c étant un paramètre de la structure LSH [Gorisse 2010]. Il y a autant de requête à effectuer que d'individus dans les différentes sous-populations de l'AG, ce qui nous donne une complexité pour cette étape de l'ordre de $\mathcal{O}\left(B \times n_{ind} \times n_{ATS}^{\frac{1}{c}}\right)$. La sélection des images à ajouter à l'ensemble d'apprentissage parmi les images obtenues se fait ensuite de manière aléatoire et l'annotation consiste simplement à lire la catégorie de l'image dans les données. Ces étapes ont donc une complexité négligeable.

Le dernier élément est l'apprentissage de la machine à vecteurs de support avec l'ensemble d'apprentissage disponible, c'est-à-dire $SVM_{train}(i \times n_{add})$ à l'itération i. Le tout mis ensemble, la complexité totale du système est exprimée par l'équation 4.6. La simplification est faite en utilisant les mêmes suppositions que pour la sélection active.

Comparaison des Complexités La sélection aléatoire est la méthode la plus simple et la moins coûteuse, le coût se résume au coût d'entraînement de la machine à vecteurs de support.

La sélection active nécessite d'évaluer et de trier l'ensemble des images disponibles pour l'apprentissage, afin de sélectionner les plus intéressantes. Cette méthode dépend donc linéairement du nombre d'images disponibles pour l'apprentissage. Les résultats obtenus par la méthode active présentée ne sont pas bons, cependant la majorité des méthodes actives parcourent l'ensemble des images disponibles pour l'apprentissage afin d'en sélectionner les meilleures. La complexité sera donc similaire et dépendra linéairement du nombre d'images disponibles pour l'apprentissage. Ceci peut être un frein à l'utilisation de ces méthodes pour de grosses bases de données.

Le système hybride proposé ne dépend du nombre d'images disponibles pour l'apprentissage que par le terme $n_{ATS}^{\frac{1}{c}}$, où c est un paramètre de la structure LSH. Cette dépendance peut donc être modulée, et comme $c > 1$, la dépendance est sous-linéaire. La complexité de ce système est de manière générale fortement liée aux paramètres de l'AG et de la structure LSH. Les paramètre du LSH doivent

être définis lors de la phase de calculs préliminaires, mais les paramètres de l'AG peuvent être changés juste avant l'apprentissage. Cela permet d'adapter la quantité de ressources investies dans l'apprentissage. De plus, comme la dépendance avec le nombre d'images disponibles pour l'apprentissage est sous-linéaire, ce système est intéressant pour traiter de grosses bases d'images.

Finalement, bien que nécessitant plus d'étapes de calcul que la sélection aléatoire ou active, le système présenté s'est révélé plus efficace que ces derniers dans les expérimentations faites sur Caltech-256. De plus, malgré ces étapes de calcul supplémentaires, la complexité du système dépend de manière sous-linéaire du nombre d'images disponibles pour l'apprentissage. Le reste de la complexité étant lié aux paramètres du LSH et de l'AG. Ce système parait donc intéressant et des études plus approfondies permettront de mieux cerner ses forces et ses faiblesses. Des propositions d'études sont faites dans le chapitre suivant.

Conclusion et Perspectives

Le volume d'images disponible sur Internet ne fait que croître. Il devient donc nécessaire d'avoir des systèmes permettant de fouiller et classer ces images sans recourir à d'autres informations que le contenu visuel. C'est le but des systèmes de recherche d'images par le contenu. Mais avec le volume croissant d'images, de nouveaux défis sont apparus, tel que la classification d'images à grain fin. Cette dernière consiste à être capable de séparer des images visuellement similaires mais ne représentant pas le même concept sémantique et vice versa. Les méthodes traitant la classification d'images par le contenu n'adressent pas complètement ce problème, souvent parce que ces méthodes se concentrent sur une recherche locale, oubliant l'exploration.

Suite à une étude d'algorithmes évolutionnaires (AE) dédiés au type du problème soulevé par la classification à grain fin, il s'est révélé que ces algorithmes ne peuvent pas être utilisés seuls pour résoudre ce problème car ils ne sont pas assez efficaces dans des espaces de grandes dimensions. Or les représentations d'images contiennent souvent plus d'un millier de dimensions. Nous avons alors proposé un système hybride utilisant un AE, connu pour ses capacités exploratoires, pour fournir des éléments d'apprentissage à une machine à vecteurs de support, l'un des plus performants algorithmes de classification. Une étape d'assignation des individus de l'AE à des images présentes dans l'ensemble d'images disponibles pour l'apprentissage est nécessaire car les individus de l'AE peuvent ne pas représenter une image existante. Cette étape est assurée par une structure de hachage localement sensible (LSH) permettant d'effectuer une recherche approximée rapide de plus proches voisins. Ce système s'est révélé plus performant que la sélection aléatoire et qu'un système de sélection active basé sur de la recherche locale sur la base Caltech-256 contenant environ 30 000 images réparties en 256 catégories. De plus, la complexité de ce système ne dépend pas fortement du volume d'images disponibles pour l'apprentissage, en faisant une méthode intéressante pour traiter de plus grosses bases d'images.

La complexité dépend en effet essentiellement des paramètres du LSH et de l'AE, permettant de régler la quantité de ressources investies dans l'entraînement. Ces résultats encourageants et ces propriétés intéressantes ouvrent des opportunités pour de plus amples analyses de ce système, dont certaines sont présentées ci-après.

Perspectives

Nous distinguons ici deux types de perspectives : celles liées à des expérimentations supplémentaires permettant de mieux cerner le comportement du système et celles concernant des évolutions possibles du système en rapport à des limitations ou restrictions que nous avons identifiées.

Expérimentations Complémentaires

Influence des Paramètres Il convient d'effectuer une analyse de l'influence des paramètres propres au système, à savoir : le nombre de générations g entre deux extractions de la population pour effectuer la procédure d'annotation des images; le nombre d'images x sélectionnées pour être ajoutées à l'ensemble d'apprentissage chaque fois que la procédure d'annotation est exécutée. Ce sont des paramètres introduits par l'architecture du système et non des paramètres d'algorithmes existants, il convient de savoir quel est leur impact sur les performances du système. Une étude est donc nécessaire.

Choix d'Implémentation Une seule implémentation du système a été faite, utilisant NSGA-II comme AE, LSH pour l'assignation des individus à une image et une machine à vecteurs de support à noyau linéaire utilisant la stratégie du *un contre tous* pour la classification. Ces algorithmes ainsi que leurs paramètres peuvent être changés. D'autres fonctions de hachage pourraient être utilisées dans le LSH, le nombre de tables L ou de fonctions de hachage par table k pourraient être modifiés voir adaptés en fonction de la base d'images. L'évolution différentielle (DE) ou la stratégie d'évolution d'adaptation de la matrice de covariance (CMA-ES) pourraient remplacer NSGA-II comme des alternatives d'AE étant plus adaptées à un environnement continue. Le type de machine à vecteurs de support et ses paramètres peuvent aussi être optimisés.

Autres Représentation d'Images et Bases d'Images plus Grandes Afin de confirmer que ce système s'adapte à n'importe quelle représentation d'images, il serait intéressant de le tester en utilisant les représentations d'images les plus récentes apprises à l'aide de l'apprentissage profond. Il est aussi intéressant d'étudier le comportement du système sur de plus grosses bases d'images. Notamment comment les différents paramètres doivent être fixés pour obtenir de bonnes performances. Si g est trop grand et x trop faible, le coût du système devient très important, rendant l'ensemble potentiellement inintéressant.

Fonctions de Fitness de l'Algorithme Évolutionnaire Deux fonctions d'évaluation ont été proposées pour évaluer les individus de l'AE. Le but de l'AE est d'explorer l'espace tout en identifiant les zones permettant d'améliorer au mieux la classification de la machine à vecteurs de support si des images de ces zones sont ajoutées à l'ensemble d'apprentissage. D'autres fonctions peuvent être mises en place pour remplacer celles qui sont proposées, en s'inspirant en partie de méthodes de sélection actives pour les machines à vecteurs de support multi-classes par exemple.

Ce ne sont ici que quelques expériences complémentaires qui pourraient être menées pour mieux cerner les points forts et les points faible de ce système et d'en améliorer l'implémentation. Voyons maintenant quelques points qui peuvent être améliorés dans le système lui-même.

Amélioration du Système

Tel qu'il est actuellement, le système ajoute le même nombre d'images provenant de chaque catégorie à l'ensemble d'apprentissage. Or chaque catégorie ne contient pas nécessairement le même nombre d'images et certaines catégories peuvent être plus difficiles à identifier que d'autres. De plus, afin de pouvoir respecter cette contrainte, plusieurs structures LSH ont dû être construites. Ce mécanisme, mis en place pour respecter les conditions d'expérimentation fixées par [Perronnin *et al.* 2010], n'est donc pas forcément adapté et doit être revu.

Le système requiert aussi une image requête provenant de chaque catégorie de la base de données qui doit être identifiée comme requête initiale. Ces exemples ne sont pas nécessairement disponibles, voir même le nombre de classes peut ne pas être connu. Le système peut être modifié pour permettre une initialisation avec des requêtes provenant au minimum de deux classes. Des classes supplémentaires peuvent être ensuite ajoutées. Dès qu'une image est annotée avec une nouvelle étiquette lors de la phase d'annotation, elle sert de requête pour cette nouvelle classe et une nouvelle sous-population est initialisée dans l'AE pour travailler sur cette catégorie.

D'autres limitations et améliorations sont certainement possibles. Mais cette thèse se termine ici, et elle laisse beaucoup de travaux à effectuer pour approfondir la compréhension du système introduit. Nous travaillons déjà à l'étude du comportement de ce système sur de plus grosses bases d'images en utilisant une représentation d'images issue de l'apprentissage profond.

Acknowledgments

First of all, thank you to both of my supervisors Denis Pallez and Frédéric Precioso without whom this document would have never exist. They supported me throughout the whole process to complete this thesis, helped me overcome the difficulties and were very supportive with regards to the personal problems I encountered during the thesis. They believed I could achieve great things all along and I hope what I have done is enough to fulfill at least part of their expectations.

Thank you Christian Brel, for all the help and support he gave me when I most needed it. I couldn't have went through the personal issues I faced so easily without his help, and he has always been supportive.

Thank you Magali Brel, for all the love and support you gave me through the two past years. Without you and Christian, I probably would have given up at some point.

Thank you to my family, they were always there to support me, in particular my mother, Corinne, and my sister, Anaïs, who helped me recover when I needed it. I always enjoy the time I spend with all my family and the vacation I had the chance to take during the thesis to visit them helped me overcoming the difficulties I faced.

Thank you Paul Sachez, a long time friend who always supported me and helped me overcome all the difficulties I encountered. Thank you to his parents too, who kindly accepted to host me for a few days when I really needed it.

Thank you Dimitri Martinho, another very long time friend who has always been supportive and came to spend some time with me. He has always been there when I needed and I hope I'm there for him too.

Thank you Anne-Marie, Jean-Paul, Ivan, Cyril, Christian Delettre, Stéphanie, Mélanie, Katty, Yoann, Sébastien, Simon, Sami, Clément, Cécile and all the other people from the SPARKS team I had the chance to talk with and spent time with. Speaking with all of you was a great pleasure, I truly enjoyed all the time I spent with each of you. The moments we spent together, either at lunch, somewhere in the afternoon, an evening around boardgames or dining at a restaurant, or even working together were great and helped me in completing this thesis.

Thank you Magali Richir, Régine Saelens and all the administrative staff from the I3S laboratory and the doctoral school STIC from Nice for their help and support. Doing all the administrative tasks from afar while being in recovery couldn't have been possible without your help. And thank you for the help and support you provided me all along the way.

Thank you to all the friends from our teamspeak server: Aerendil, Gnollset, Krusk, Shysu, Mimil, Kira, Ursakar, Mathosse to name just a few. I spent a lot of time playing and speaking with you all, which helped me a lot too. Also thank you Riot Games, Quickshot, Krepo and all the league of legends casters and analysts; playing the game and watching the pro scene was part of the distraction that helped me escape from the work for some time to better get back to it after.

Thank you Cyril Fonlupt and Arnaud Revel for accepting to review this work,

and thank you Carlos Artemio Coello Coello and Andrea G. B. Tettamanzi for accepting to be member of the jury of my PhD defense. I hope you will appreciate it.

I'd also like to thanks Florent Perronnin for giving us the image description he used in his work. Those would have taken several weeks to obtain without his kind help.

Finally thank you, the reader, for reading this. It means a lot to me, because it means that you find this work is worth investing some of your time, and considering the time we spent with my supervisors to create it, I hope you will find something interesting in it.

Contents

Acronyms

BoF Bag of Features. 12, 13, 17, 19–21

BoW Bag of Words. 13, 15, 17

CBIR Content Based Image Retrieval. xxxv, 3–6, 8, 10, 13, 16, 17, 19, 22–24, 26, 27, 30, 31, 41, 47–49, 57, 71–73, 75, 77, 99, 104, 106, 113, 115, 119

DE Differential Evolution. 37–40, 58, 62, 76, 78, 121

EA Evolutionary Algorithm. xxxv, 30, 31, 36–39, 41, 46–51, 55, 57, 58, 62, 71, 75, 77, 78, 82–92, 108, 119–121

GA Genetic Algorithm. 31, 36–38, 41–44, 46–49, 51, 54, 55, 58, 62, 73, 75–78, 81, 83, 87, 92–95, 99, 107, 112, 115, 117, 120–122

GMM Gaussian Mixture Model. 15

IGA Interactive Genetic Algorithm. 47, 48, 71, 73, 75, 76, 78

IR Image Retrieval. 3

LSH Locality Sensitive Hashing. 95–99, 113, 116, 117, 120, 122

ML Machine Learning. 5

MMOP Multi-Modal Optimization Problem. 49, 55, 57–60, 62–64, 66, 67, 69

MOP Multi-Objectives Optimization Problem. 49–55, 57–59, 62–65, 78

NSGA-II Non-dominated Sorting Genetic Algorithm II. 54–56, 60–62, 65–69, 75, 92, 93, 106, 115, 120

PoI Point of Interest. 11–13

PSO Particle Swarm Optimization. 37, 40, 47, 77

SIFT Scale Invariant Feature Transform. 9, 10, 12, 16, 48, 104

SVM Support Vector Machine. xxxvi, 7, 22–27, 29, 31, 32, 34–36, 47, 71, 73, 75–78, 81–94, 99, 104, 106–112, 114–117, 119–122

Introduction

When using the term "Pizza aux anchois" (which is the French for pizza with anchovies) as a query in Google image, you may stumble into an image of the famous French model Adriana Karembeu within the first results. She is not really looking like a pizza with anchovies and this results is quite far from what is expected as results for this query. This happens because the ranking of images is based on the contextual information available around the image. In this case, the image of the model is located on a carefully crafted page speaking only of pizza with anchovies, the wine you should drink with pizza with anchovies, different recipes of pizza with anchovies, etc. Because of this, it is ranked high within the results for the query "pizza with anchovies". This is a great example of one of the flaws of systems performing image search using meta-data and peripheral information only. They can be fooled by false data provided around the image. This is one of the reasons that lead to the development of content based image retrieval systems.

A content based image retrieval system performs data mining tasks (search, ranking, classification, ...) on images using the visual content of images only. Such systems cannot be fooled by false surrounding content as they are not using it. In addition, those systems are of great interest when dealing with images without contextual information or meta-data associated to it. This tends to happen a lot nowadays, with the number of images produced and uploaded on the Internet. Indeed, from its creation in 2004 to December 2014, 5.26 billions photos were posted on Flickr; and in 2015, the number of public photos uploaded to the Internet every day is estimated at 2 millions.

This thesis introduces a new hybrid framework dedicated to image classification in a content based image retrieval context. The proposed framework uses an evolutionary algorithm to select training samples for a support vector machine. To converge to such a system, the most popular techniques to address content based image retrieval are reviewed first. This review reveals some limitations of the existing techniques, preventing them to accurately address some problems. In particular, the fine grained classification problem introduced at the end of chapter 1. Evolutionary algorithms are then identified as a potential interesting solution to undertake the fine grained classification problem. Thus they are studied in the second chapter of this thesis. Their performances for the problem considered are not good enough to use them alone to solve it though. Thus using them together with another algorithm is considered, forming what is called a hybrid system. Existing hybrid systems in content based image retrieval are reviewed at the end of the second chapter. Using an evolutionary algorithm to provide training samples to a support vector machine seemed to be new. And a first implementation of such a method lead to promising results. Based on this preliminary work, an innovative hybrid framework is proposed and detailed in chapter 3. Experiments and performances of this framework on a standard benchmark database (Caltech-256, containing around 30,000 images)

are provided in chapter 4. A conclusion reviewing the work done together with the perspectives opened by this work are then provided at the end of the document.

Chapter 1

Content Based Image Retrieval

Contents

An Image Retrieval (IR) system is a system allowing to perform different data mining tasks on images from a database (classification, retrieval, copy detection, ...). IR systems may use any kind of data available (keywords, gps location, website from which the image is extracted, ...) to perform those tasks.

A CBIR system is an IR system that uses only the visual content of images as input data. This means that no keywords or any meta-data are used during the process, only information extracted from the pixels of the images is used.

This chapter is dedicated to the presentation of CBIR. An insight of the most common tasks performed using CBIR and the general structure of a CBIR system are provided in the first section. This is followed by an explanation on how information is extracted from images to build meaningful image representations. The algorithms commonly used in CBIR are then briefly presented. Some remarks about the limitations of those algorithms conclude this chapter.

1.1 Tasks and Framework

CBIR is the fact of mining information from images using their visual content only. To better understand CBIR systems, we will first describe the most popular tasks addressed using CBIR systems before explaining the general framework of a CBIR system.

1.1.1 Common Tasks in Content Based Image Retrieval

Face Detection Face detection consists in finding any human face present in an image. Faces can be of different sizes (foreground and/or background), from different angles (front, profile, ...) and of different types (skin color, eyes, ...). The CBIR system is expected to retrieve the faces and their bounding box in any image [Wong *et al.* 2001]. This is used in cameras to identify where people are when taking a picture for example.

Copy Detection In copy detection, given an image (called the query), the system must find any copy or derivative of this image in a database [Kim 2003]. Derivative means any geometric transformation, lightning change or even color modification. The system must be very discriminative for such a task, in order to strictly retrieve copies of the query and not just similar images. This is used to identify illegal copies of arts or images and copyright infringement for example.

Object Detection and Classification Detecting objects is the task of identifying what is an object in an image (not just grass, soil, ...) and defining its bounding box. The classification part then gives a label to the objects (bike, car, horse, ...) [Winn *et al.* 2005]. This task is applied on big databases to automatically detect and label objects in images. Manually labeling them being impossible due to the amount of images created every day. The result is a set of bounding boxes and labels for each image from the database.

Image Retrieval A user comes to the system with a query image, and he wants to find images similar to this query in a database. The system is then expected to rank the database so that the images the most relevant to the query are at the top and the least relevant to the query at the bottom of the ranking. This can be used for everyday search [Lai & Chen 2011], but can also be domain specific [Lakdashti & Ajorloo 2011].

Image Classification Here, given a database of images, the system must give one (or several) labels to each image of the database, thus classifying the images in several categories (planes, beaches, mountains, ...). In opposition to object detection and classification, images are here considered globally. This is used to automatically label images. Image labeling can be used for common objects like the categories previously cited for example [Gorisse *et al.* 2010].

Those are only some of the tasks addressed using CBIR systems. In this work, we will concentrate on image classification in particular. Now that the task has been identified, the workflow of a CBIR system will be introduced to have an insight of how this works.

1.1.2 General Framework

CBIR engines are complex systems putting together several elements as shown in Figure 1.1. First of all, they rely only on information contained in the pixels of the images. However, trying to retrieve images using individual pixels information only is not relevant. Indeed, those do not hold enough information on their own. The system is expected to find similar images at a semantic level, pixels do not provide such information and do not allow to access to such a level of abstraction easily. Therefore, the first task when creating a CBIR system is designing a representation of images, as shown at the top of Figure 1.1. The representation is extracted for each image in the database. It must hold meaningful information in a compact way. Reducing the size of the description reduces the cost and complexity of similarity computations, and thus the cost of the computation of the decision taken by the Machine Learning (ML) algorithm. This algorithm is referred to as the search engine in the middle of Figure 1.1. Therefore, the more compact the representation is, the faster the system will be. It is also important to have a measure of similarity associated to this representation in order to be able to compare images. Creating such representations and similarity measures is a field of research on its own, and an insight of the works in this field is given in the next section.

Once the representation of each image in the database is extracted, the search engine is learned. In practice, the search engine is a ML algorithm learning a model able to assign images into relevant classes. Most of the time, a part of the dataset is already annotated using expert information. It is then used to train the ML algorithm. However, several other techniques exist to feed a ML algorithm with training data, those are explained in the next subsection. And numerous ML algorithms can be used to learn a classification model. The major techniques used to address CBIR are introduced in section 1.2.4.

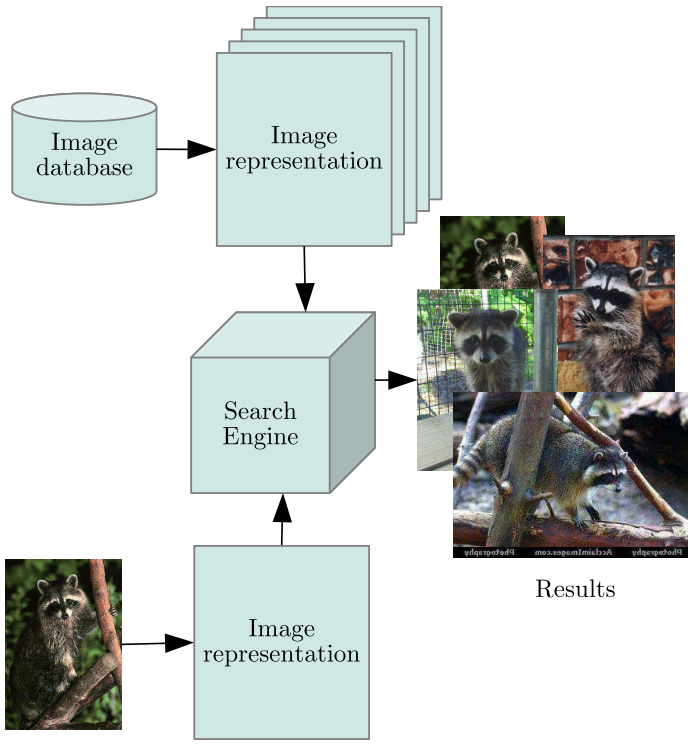
Then, the trained ML algorithm is applied to the representation of every image of the dataset, assigning them a label. The dataset is now completely classified, and the trained ML algorithm can be applied to classify any new image added to the dataset. It is done by first computing the representation of the images to be added, and then applying the ML algorithm to determine its class, as shown at the bottom of Figure 1.1.

This ends the description of the CBIR general workflow for classification. The most common the ways that can be used to feed a ML algorithm with training data, applied in a CBIR context in particular, are presented in the next subsection.

1.1.3 Learning Contexts

This thesis is focusing on the image classification task within CBIR. Therefore we are more interested in classification algorithms than other algorithms within the wide range of ML algorithms. Classification algorithms approximate a classification function f using examples (training samples forming a training set) to be able to predict the class of any image. The learning context then consists in how the

Figure 1.1: Global workflow of a CBIR system



information is provided to the classification algorithm, and what kind of information is given to the algorithm. We can separate those contexts into two main categories, depending on the quantity of information provided to the algorithm to learn from: unsupervised classification in which a set of data is provided to the algorithm without any other information; supervised classification in which a group of data is given to the algorithm together with their class. Those two methods and their applications in CBIR are briefly detailed below.

1.1.3.1 Unsupervised Classification

Unsupervised classification consists in, given a set of training samples and an expected number of classes, learning a classification function without any a priori knowledge about the classes of the training samples.

Some works are using unsupervised learning as part of CBIR systems, such as [Chen *et al.* 2005], which uses unsupervised clustering to organize the results of an image retrieval results. Image retrieval results are usually presented as a list of images ordered by their similarity to the query. In their work, unsupervised clustering is used to organize the set of results into several clusters. The list of clusters is presented to the user by showing one selected image of each cluster. The user can then explore the content of each cluster separately.

But the main use of unsupervised image classification is for very specific domains, such as medical images [Dinh *et al.* 2013], radar images [Cloude *et al.* 2002] or even satellite images [Xu *et al.* 2013]. In those domains, even experts may have difficulties to classify images, therefore, a priori information are difficult if not impossible to gather. Therefore, unsupervised learning makes sense. Because no a priori knowledge is provided, unsupervised learning usually separates the data based on the similarity measure, gathering together similar data, such as the k-Means algorithm [J. A. Hartigan 1979]. The results are quite good when data are easily separable based on the similarity only, but things get complicated when visually dissimilar data have to be put into the same class.

1.1.3.2 Supervised Classification

In supervised classification, the class of the training samples is known a priori (or easy to determine), and is used in the learning process to build the classification function. Let $\mathcal{ATS}$ be the set of available training samples with a priori knowledge of their classes, and $\mathcal{TS}$ be the training set used by the classification algorithm to learn the classification function. $\mathcal{TS}$ is a subset of $\mathcal{ATS}$. A lot of techniques are based on supervised learning to perform content based image classification. They can be separated by the way $\mathcal{TS}$ is built from $\mathcal{ATS}$.

Raw Supervised Classification In those methods, all elements of $\mathcal{TS}$ are selected at the same time and provided to the algorithm to build the classification function. The simplest version of this is to select all individuals of $\mathcal{ATS}$ to form $\mathcal{TS}$. This is the technique used by [Perronnin *et al.* 2010] (as well as the methods they compare to) when evaluating the performances of their improved Fisher vector representation. A SVM classifier is learned using all available training samples for each representation, and they are evaluated on the same test set in order to compare their performances. This learning scheme is very simple to set up, but has a major drawback: its computation time explodes with the number of available training data. Indeed, classification algorithms learning time mainly depends on the size of the training set, the complexity most of time being more than linear (SVM complexity is quadratic with respect to the size of the training set for example). With recent challenging datasets containing millions of images even for training, this method does not scale up.

To build the decision function, classification algorithms expect the training data to follow the same probabilistic model as the data itself. With millions of training samples available, it is expected that a random subset of a significant size of those data still follows the same probabilistic model. Thus, some works learned classification algorithms on a random subset of $\mathcal{ATS}$ [Balcázar *et al.* 2001]. While this in theory should not affect the performances, in reality, performances greatly depends on the content of the random selection, leading to unstable results. To enhance stability, some works performed several random selections to build a better and more stable final training set from $\mathcal{ATS}$, such as cross validation or n-fold

strategies. A ten folds strategy consists in randomly splitting $\mathcal{ATS}$ in 10 parts of the same size. Each part is used in turn as the test set to assess the performances of the classification function built using the 9 remaining parts as a training set. The best performing classification function is retained once every part has been used as training set once. Although this performs quite well, it still requires a lot of resources because of the several random selections and evaluations before getting the final classification function.

Active/Interactive Learning In order to avoid the multiple selection and evaluation process done in random selection schemes, new methods were developed, building $\mathcal{TS}$ incrementally from an empty or fairly small set. Those are called active or interactive learning. In active learning, only computers are involved while in interactive learning a user gives feedbacks about some images. Given a classification algorithm, a set $\mathcal{ATS}$ of available training samples, and a partially built training set $\mathcal{TS}$, those techniques focus on finding which data from $\mathcal{ATS}$, if added to $\mathcal{TS}$ with their labels, will improve the most the performances of the classification function built by the algorithm. In the case of active learning, all labels in $\mathcal{ATS}$ are known, and thus can be automatically provided to the algorithm. For interactive learning, $\mathcal{ATS}$ is often the whole database, for which labels are unknown, and labels are asked to the user when needed. Beside this main difference, they tend to both use the same underlying techniques for training samples selection. Those techniques shift the computation overhead of running multiple random selection to the careful selection of training samples, achieving better stability and performances. [Tong & Chang 2001] and [Gorisse *et al.* 2010] developed active learning schemes dedicated to CBIR for example.

Working in a general image classification context, the goal is to identify the class of an image among simple classes such as bears, mussels, cars etc. Those classes are easy to determine, and the database we are working with provide a set of training samples for which classes are known. Supervised classification is the method of choice in such a context. Now that the learning context has been identified, the different components of the CBIR processing chain are introduced in the following section.

1.2 CBIR Processing chain

The first component of a CBIR system is the description of the images. The gap between the raw visual content and the semantic similarity is called the *semantic gap*, and one of the major challenges of CBIR is to bridge that gap. Carefully design an image representation and its associated similarity measure is the foundation stone of that bridge. The goal here is to build a compact and meaningful representation of images together with a similarity allowing a quick comparison of images. To do so, visual features are extracted from the pixels' information as presented in subsection 1.2.1. Then, those features are combined so as to build a final representation of

images, which is detailed in subsection 1.2.2. Those image descriptions, as well as the visual features themselves, are usually real-valued vectors. Finally, a similarity measure must be defined on those description to provide a semantic level comparison of images. The most standard measures are presented in subsection 1.2.3.

1.2.1 Visual Features

Visual features are extracted from the pixels' information. More than the pixels themselves, they aim at grasping some visual information from a group of pixels or even from the entire image. Several kinds of information can be extracted this way, some are presented in the following paragraphs.

Color An image is represented by its color at every pixel. So this feature is in fact the image file itself. However, all color space are not equally relevant to the human perception. The standard RGB color space is adapted to displays, which use a mixture of red, green and blue to create any color. But this is one of the worst representations when it comes to human perception. Thus, color feature in the literature are most of the time based on other color spaces like YUV, HSV, etc. Those are considered to be closer to the human color perception. In addition, color features are often computed as the mean of a group of neighboring pixels, and not as individual pixel's color.

Texture Right after colors, textures come as the most considered feature in semantic visual representation. Whilst a lack of explicit definition, texture features implicitly define spatial structures emerging from pixels by analyzing pixel's intensity. The lack of explicit definition induce the existence of multiple texture features.

One kind of texture feature uses statistical models and information to represent textures. We can mention [Haralick *et al.* 1973] who uses the co-occurrence matrix for example.

Others are using convolution with filter banks to extract texture information. The most commonly used filter bank is the Gabor filters [Manjunath & Ma 1996] bank; but Haar wavelets [Viola & Jones 2001] can also be cited, mainly for the efficient representation of face they provide. The feature can be extracted for any pixel of the image by computing and concatenating the response of the pixel to each filter of the bank. Filters are often computed on patches of several pixels, the pixel at the center of the patch is then considered to be the one for which the texture feature is extracted.

Last classic texture features cited here are based on gradient orientation. Histogram of Oriented Gradient (HOG) [Dalal & Triggs 2005] or the Scale Invariant Feature Transform (Scale Invariant Feature Transform (SIFT)) [Lowe 1999] are good examples of such features. For each pixel, the HOG constructs a uniform grid around the pixel and extracts the gradient quantized over 8 different orientations (0°, 45°, 90°, ...) for each cell on the grid. This forms an histogram for each cell, those histograms are then concatenated to form the final description of the pixel. The

SIFT descriptor does the same 8 orientations extraction, using 4-by-4 pixels cells on a 16-by-16 cells grid around the pixel. Each cell's histogram magnitude is weighted with a Gaussian function with σ equal to half the width of the descriptor's grid size. Then, the 16 resulting histograms of 8 orientations are concatenated to obtain a 128-bins histogram, which is first normalized to unit length. This improves its invariance to affine changes in brightness. To cope with non-linear brightness changes, a threshold of 0.2 is further applied. Finally, the histogram is normalized again to obtain the final description. The SIFT descriptor has been widely used in the CBIR community, and *corresponds to the low-level visual description we are using in our experimentations.*

Shape Some techniques segment the image into several regions. Each region may contains several semantic objects [Carson *et al.* 2004, Fauqueur & Boujemaa 2004]. Shape features are most of the time based on the decomposition of the signal on a basis of functions like the Legendre polynomial basis [A. Foulonneau *et al.* 2009], but they can also be based on contours analysis [Belongie *et al.* 2000, Belongie *et al.* 2001, Belongie *et al.* 2002].

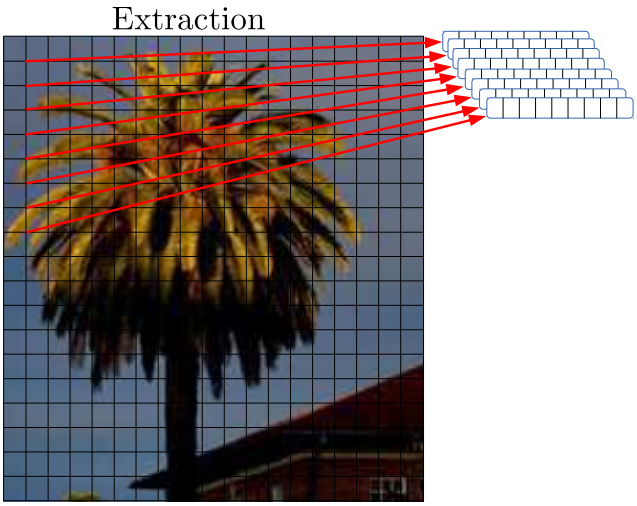
Shape features usually only identify several regions in the image. To build a proper description, they are paired with the previously presented color and/or texture features. Features are then extracted from each region, considering each region as a new image. Regions' features are considered as a set of independent groups of features representing each region.

Those are just some characteristics that can be extracted from pixels or group of pixels. To build a meaningful representation of an image, those features must be extracted on several pixels (or group of pixels) within the image. The extraction of visual features from several pixels in an image can be separated into two main categories: *dense* extraction and *sparse* extraction which are presented in the following subsections.

1.2.1.1 Dense Extraction

Dense extraction consists in computing a feature vector for every position on a regular grid (the finest grid gets nodes at each pixel) as shown in Figure 1.2. A large number of feature vectors is then associated to each image. Considering a representation of images made from the concatenation of those feature vectors, comparing images becomes complex and time consuming. It will even become impractical when dealing with large databases. In addition, as the feature vectors themselves do not hold semantically meaningful information, the similarity function would need to bring some generalization capabilities. Therefore, when using such an extraction, clever and compact image representations are further built from the feature vectors, so as to ease the computation of image similarity. Such representations are explained in Subsection 1.2.2.

Figure 1.2: An example of dense extraction of feature vectors on one image. A feature vector is extracted on each point of the grid.

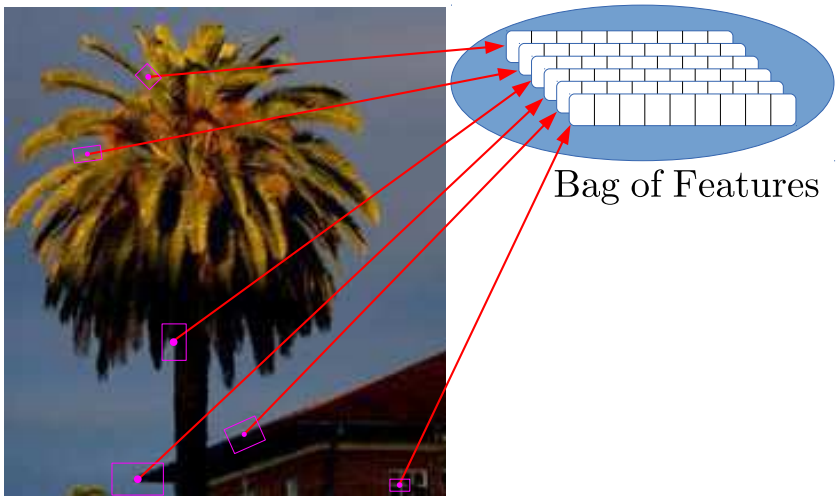


1.2.1.2 Sparse Extraction

As represented in Figure 1.3, sparse extraction mainly concentrates on identifying Points of Interest (PoIs) (represented in pink on the figure) in images so as to extract visual descriptions for those PoIs only. Identifying a PoI consists in precisely locating such a point in an image. PoIs are points with peculiar visual saliency, like a maximum of information, or being the point the less similar to its neighbors (two properties that can be identified using the autocorrelation matrix). Once identified, a description of the neighborhood of the point is extracted, usually using any visual feature such as the ones presented in Subsection 1.2.1 for example. PoIs detection and description must be repeatable: the same points and same descriptions must be extracted every time the process is run on a given image. PoIs are expected to be invariant to geometric, scale and luminosity transformations too. This enhances the semantic meaning of the image representation built on top of them. Indeed, if the same point is detected whatever its orientation and its lighting condition, it represents more the object than environmental conditions and thus better approximates the semantic meaning than a color feature randomly extracted on the image. For classification tasks, it is also desirable that PoIs generalize well, i.e. the same points are detected on the same kind of objects (points detected on a bus are also detected on other buses). This would enhance even more the semantic meaning of the description built from features extracted using such a scheme and would allow to use a less costly similarity measure to compare images as part of the semantic is held in the features.

Many PoIs have been developed through the years, a large number of them using gradient detection. One of the first most popular detector is the Harris cor-

Figure 1.3: Sparse extraction of feature vectors on one image. Pink dots are the detected PoIs and the pink rectangles are the neighborhood from which feature vectors are computed.



ner detector [Harris & Stephens 1988]. Unfortunately, it is not invariant to scale changes. Later, scale stable detectors were developed using a scale-space representation. The first detectors were based on the scale-normalized Laplacian of Gaussian (LoG) [Lindeberg 1994, Lindeberg 1998].

Because of the mathematical properties of the scale space, the LoG could be approximated using a Difference of Gaussian (DoG) operator, which was used to develop the SIFT detector [Lowe 2004]. It corresponds to extrema in the scale space, identified by local extrema of DoG in 3 adjacent scales. The SIFT detector integrates scale and orientation detection, allowing it to be invariant to scale and rotation changes. The SIFT detector must not be mistaken with the SIFT descriptor of subsection 1.2.1; the detector identifies a PoI while the descriptor builds a feature vector that represents the neighborhood of that point. Other detectors were developed using various scale space representations like the JET point detector [Schmid & Mohr 1997] or the Determinant of Hessian used in the Speeded Up Robust Features (SURF) descriptor [Bay *et al.* 2006].

A feature vector is extracted on each point detected. The pink rectangles in Figure 1.3 represent the areas on which descriptions are extracted for each point. It can be observed that those areas can be of variable size (stability to scale changes), variable orientation and even variable shape (for stability to affine changes). Although detectors usually come up with their own descriptors, fixing some of the detector's drawbacks, any pair of detector, descriptor can be used.

Sparse extraction provides a set of feature vectors per image called a Bag of Features (BoF). Each vector holding the representation of the neighborhood of a given PoI. The union of all described areas usually does not cover the whole image,

as shown in Figure 1.3 where the pink rectangles are far from covering the whole image.

Once the BoF is extracted from an image, one can simply consider the set of all these feature vectors as the image representation. The similarity would then be evaluated by measuring the similarity between every pair of points (made of one point from each image compared). However this kind of representation does usually not lead to the best performing one, and it is a quite heavy representation. Indeed, even if a feature vector is not extracted for every pixels, several hundreds of PoIs can be detected per image. Therefore, clever image representation have been built through the years to address this issue. Whether using dense or sparse extraction, several image representation can be built from the resulting set of feature vectors obtained from each image. The most standard are presented in the next subsection.

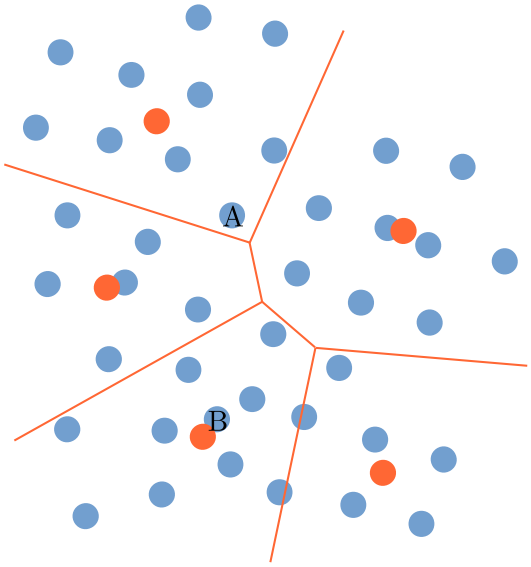
1.2.2 Image Representation

Given the number of feature vectors extracted from each image, comparing images using all those vectors is both complicated and costly. Therefore, given the number visual data added on the Internet everyday and the growing size of the databases, more compact representations are needed, so that comparing two images costs less. However, this compactness must not be done at the expense of the semantic relevance of the description. While a lot of image representations exist [Deselaers *et al.* 2004], since our system is built on top of the representation schema and independently from it, we present in the following a few of the most common and most recent works in the domain, starting with visual dictionary based approaches before introducing generative-discriminative models. Finally representations built using deep learning algorithms are presented also as they are the most recent works in the domain.

Visual dictionary-based approach Visual dictionary-based approaches are inspired by techniques used to represent documents in the text retrieval field [Yates & Neto 1999]. Those techniques first perform a vector quantization on the set of all feature vectors extracted from every image. Vector quantization consists in building a set of k quantization vectors representative of the distribution of the extracted feature vectors. An example is shown in Figure 1.4 where the blue dots are the feature vectors and the orange dots are the quantization vectors. Any vector of the space can then be represented by the closest quantization vector, the portion of space represented by each quantization vector is delimited by the orange lines on the figure. In CBIR, the quantization vectors are called *codewords*, and the set of *codewords* is referred to as the *codebook*. Then, feature vectors of each image are associated to the closest *codeword*, and an histogram of occurrence of each codeword is built for each image. Those occurrence histograms are then normalized into frequency histograms to form the final representation called the Bag of Words (BoW) [Sivic & Zisserman 2003].

Building such codebooks is costly and involves a lot of steps. Thus, when dealing with very large scale datasets, only part of the database is used to build the

Figure 1.4: An example of vector quantization, in blue feature vectors, in orange quantization vectors. Orange lines delimits the space represented by each quantization vector.



codebook. Yet choosing the size of this part to preserve good performances is not easy.

Another drawback of these strategies is the hard assignment of feature vectors to a codeword. All assignments do not have the same relevance and the same confidence. Some feature vectors are almost equidistant from several *codewords*, such as point A in Figure 1.4, and others are very close to the *codeword* they are associated to, such as point B in Figure 1.4. Hard assignment does not take this into account, assigning feature vectors to the closest codeword and only this one.

To answer this problem, methods have been developed to improve dictionary-based approaches. One of them is *soft assignment* [van Gemert *et al.* 2008] which weights the assignment of feature vectors to codewords depending on the distance to the codewords. Many other coding techniques have been created: sparse coding; pooling methods to summarize feature vectors in large neighborhood; hard and soft quantization among others. A nice comparison of those techniques is provided by [Boureau *et al.* 2010].

Generative models and fisher scores have also been used to create image representations and are presented thereafter. They can be considered as a generalization of the dictionary-based approaches and are equivalent under certain conditions.

Generative Models and Fisher Scores Given a set of feature vectors (from all the database, from only one category, ...), generative models consider each feature vector as an instance of a random variable drawn from a probabilistic model P

with a set of parameters λ. The parameters λ are set so that P best fits the set of feature vectors of the whole database. Given an image and its associated bag-of-features $\{B_i\}$, a representation of this image can be defined as the gradient of the log likelihood, given by Equation 1.1, also known as the Fisher score. This gradient represents more or less the directions in which the parameters λ should be moved such that the model P would better fit B_i. It can also be interpreted as an indication of how much the data deviate from the probabilistic model P. The dimension of the Fisher score vector only depends on the number of parameters λ, and not on the size of B_i. This means that the final representation of each image or object has the same dimension, whatever the number of feature vectors extracted from them (exactly as in the Bag-of-words schema). This greatly ease their comparison.

$$\nabla_\lambda log(P(B_i|\lambda)) \tag{1.1}$$

P is generally represented by a Gaussian Mixture Model (GMM), which parameters are approximated using the Expectation Maximization algorithm [Moon 1996]. GMMs parameters are usually cluster assignments (each data is assigned to a cluster) and means, as well as the class prior probabilities and cluster covariance. When restricting the GMM to isotropic clusters and even prior probabilities, it is equivalent to k-means clustering. Therefore, Fisher score based on GMM can be considered as an extension of BoW when this latter is based on k-means clustering (which corresponds to hard assignment).

However, GMMs are richer, they bring more parameters as well as the 1^{st} and 2^{nd} order statistics with the derivative with respect to the means and standard deviation (gradient with respect to the parameters). In addition, GMMs operate soft assignments by construction, each point has a probability to be assign to any Gaussian in the model and statistically prefer one Gaussian to the others.

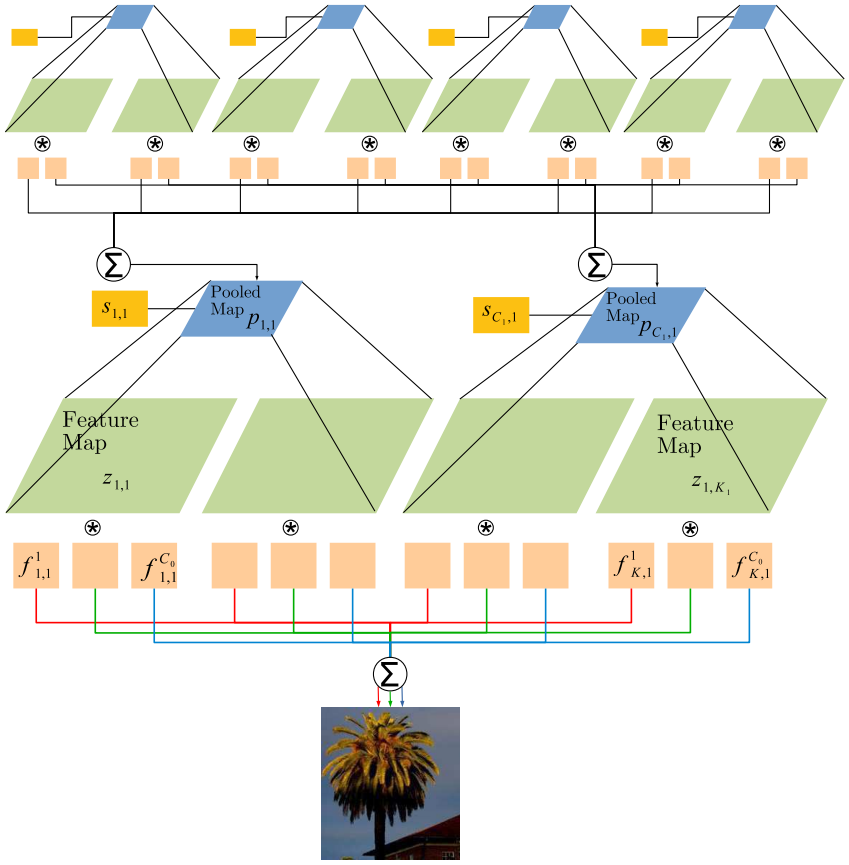
Those models have proven to be really good, and an improved version including normalization has been introduced with great success by [Perronnin *et al.* 2010] for classification tasks.

1.2.2.1 Deep Learning Based Representations

Most recent works consider building sparse representation using deep learning. Convolutional Neural Network (CNN) have been used with success for example, as in [Zeiler *et al.* 2011]. Convolution and pooling layers are alternated to form a multi-layer deep structure represented in Figure 1.5.

At each layer, filters for the convolution ($f^c_{i,l}$ in orange on the figure) are learned to minimize reconstruction errors for each image. The sum of convolutions of the features maps $z_{i,l}$ (in green on the figure) with a given channel filters $f^c_{i,l}$ gives the pooled map $P_{c,l-1}$. Therefore, feature maps (in green) are inferred from the filters and the pooled maps of the previous layer (or the channels of the input image for the first layer). Features maps are then pooled using max pooling. This selects the maximum value over a region of several feature maps, effectively reducing the size and number of maps. The indexes of the maxima are kept (in yellow on the figure)

Figure 1.5: 2 layers of the architecture proposed by [Zeiler *et al.* 2011]



for later reconstruction. Many layers of convolution and pooling are learned like this, containing much more feature maps than presented in the figure. Experiments have been run using a Spatial Pyramid Matching classification algorithm, which classically takes non-normalized SIFTs as input. The result of the first layer of the presented system is equivalent to SIFT representations. So, to take advantage of all layers, images are processed up to the last layer, inferring all feature maps. Then, the M maximum of activation on the last layer provide the final representation into this M dimensional vector. The reconstructed feature maps of the first layer forms the final representation. This representation outperformed all others classical image representations to which it was compared in [Zeiler *et al.* 2011].

This is only one type of representation based on deep learning, here using CNN. However, several other strategies based on deep learning have been built in the last years, [Bengio *et al.* 2013] presents a good state of the art of those, including but not limited to Restricted Boltzman Machines (RBMs) and auto-encoders. Most of those representations outperform other representations known so far in CBIR.

Visual dictionary, Fisher vectors and deeply learned representations are just a

few of the several image representations developed through the years. This thesis proposes a new learning framework which can be adapted to any image representation, so image representation is not the main concern here. *The representation proposed by [Perronnin et al. 2010] is the one that was chosen.* Its performances are good and, most importantly, the authors kindly provided us with this representation for the Caltech-256 database. This ensures that the exact same representation vectors are used and thus the results [Perronnin *et al.* 2010] obtained with this representation can serve as a baseline for the performances of the system presented in this thesis in chapter 3. Even though deeply learned features present better performances, recomputing them is time consuming and considering the difficulty to reproduce results from scratch of deep representation from the literature, those representations were left aside in order to test the core of the proposed framework first.

Choosing the image representation is not the last thing to do though. To be able to compare images, the representation must come with a similarity measure. The most standard measures are discussed in the next subsection.

1.2.3 Similarity Measures

The similarity measure to be used in CBIR will greatly depend on the image representation. A lot of image representations are based on histograms, such as the ones built using BoWs. Therefore, the problem of histogram comparison is briefly presented in the first part of this subsection. Sometimes, BoFs are used as image representation; most commonly when images have been segmented into several objects, then the bag of feature extracted for each object is kept as the description of the object. In such cases, we need to be able to compare BoFs, this is described in the second part of this subsection. Finally, powerful techniques uses the kernel trick to evaluate the distance between individuals in an higher dimension feature space. This technique is presented in the third part of this subsection.

1.2.3.1 Histogram Comparison

Histograms are usually real valued vectors in the image representation context. As such, they can be compared using usual distance functions such as the l_1 or l_2 distances recalled in Equations 1.2 and 1.3. Even if those are valid distance functions, the vectors under study are histograms, and more specific measures can be more suitable to compare them. The following paragraphs introduce these measures.

$$l_1(\mathbf{x}) = \sum |x_i| \tag{1.2}$$

$$l_2(\mathbf{x}) = \sqrt{\sum x_i^2} \tag{1.3}$$

Quadratic-Form distance The first distance to compare two histograms P and Q is the Quadratic-Form (QF) distance [Hafner *et al.* 1995]. It is defined by Equation 1.4, where A is the bin-similarity matrix. If A is positive-definite, QF is a metric. It can be thought of as a generalization of the l_2 distance. In fact, if A is the identity, QF is the l_2 distance. If A is positive-definite, it is the l_2 distance between the linear transformation of P and Q.

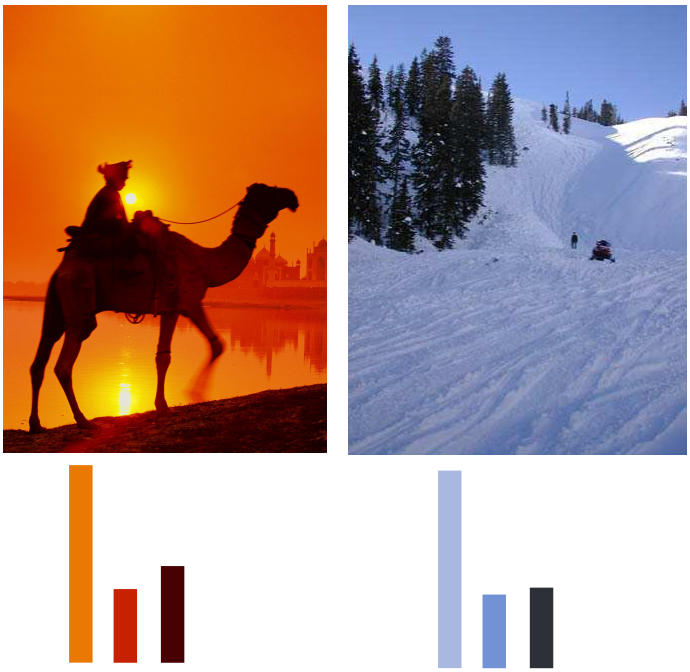
$$QF^A(P,Q) = \sqrt{(P-Q)^T A(P-Q)} \tag{1.4}$$

This measure, and the l_1 and l_2 measures, compare histograms by comparing bins at the same position between each other. This is called a bin to bin comparison. Such comparisons assume that the domain of the histograms, the feature vector represented by each bin, are aligned. This condition is most of the time not fulfilled, due to the per image quantization of feature vectors. Indeed, a sunset image contains a lot of red and yellow, whereas a snowy mountain under a blue sky contains mainly blues and whites. When comparing histograms extracted from such images after quantization with simple distance functions, they can be very close while representing completely different things. This is illustrated in Figure 1.6, where histograms of a color feature are considered. The color of a bin stands for the color it represents and its size for the amount of this color in the image. Because the most present colors in the image on the left are orange, yellow and dark brown, the three quantization vectors represent such colors. In the image on the right, the three most important colors are white-blue, blue and dark green, therefore quantization vectors represent those colors. The proportion of the three most present colors are fairly the same in both images, resulting in close histograms with respect to the l_1, l_2 or QF metrics, however those histograms represent completely different things. This is only one example in which bin to bin comparison fails, other phenomenons such as light changes could lead to wrong comparison results.

Bin to bin distances robustness can be tweaked by modifying the number of bins. The less bins in the histogram, the more robust the distance is to those alignment problems, but the less discriminative it is (it will be more difficult to separate classes of images correctly). The more bins, the less robust the distance is, but the more discriminative it is, and it will potentially lack generalization capabilities at some point, leading to the separation of images that should be put in the same class. Distances that take into account cross-bin relationship, which can be both robust and discriminative are thus often considered instead of bin to bin distances. The Chi-Squared (χ^2) distance is one of the most famous measure of this type and is explained below.

Chi-Squared distance The Chi-Squared (χ^2) is a famous histogram distance. It is based on the χ^2 statistic test and is computed using Equation 1.5. It considers relative differences instead of absolute differences, thus giving more importance to differences between small bins over differences between large bins. This is interesting for histograms because differences between small bins are often

Figure 1.6: Two images and an example of what a simple color histogram could be for each. Bin color stands for the color the bin is representing and bin size represents how much this color is present in the image. We can see that histograms are quite close in terms of bin values but bins are representing completely different colors. This is a typical case of the wrong domain alignment problem.



most important. This distance has been successfully used in many CBIR applications [Schiele & Crowley 1996, Ling & Jacobs 2007, Martin *et al.* 2004], outperforming the l_2 distance.

$$\chi^2(P,Q) = \frac{1}{2}\sum_i \frac{(P_i - Q_i)^2}{(P_i + Q_i)} \tag{1.5}$$

Those are just a two among many others available similarity measures dedicated to histograms. We won't detail more measures here, but the interested reader can refer to [Fauqueur 2003] for more measures dedicated to histograms. But what if the representation of images is not an histogram ? Sometimes, BoFs are used as image representation and thus similarity measures between BoFs are needed. Such measures have been developed by the community and some are presented in the next part of this subsection.

1.2.3.2 Bags of Features Similarity

Previous similarity measures addressed the case of data represented by a single vector. This considerably speed up the evaluation of the similarity by reducing the

number of vector comparisons. But reducing the image representation to a single vector considerably decreases the amount of information available; in particular, position information are often dropped. Therefore, BoFs are sometimes used as image representation directly, in particular in the detection of objects covering only a part of an image. In order to compare BoFs, each vector of each BoF must be compared to one another. Those comparisons must then be combined to form a final score. Only voting strategies are presented here as an example of computation of such a score, but several other strategies exist.

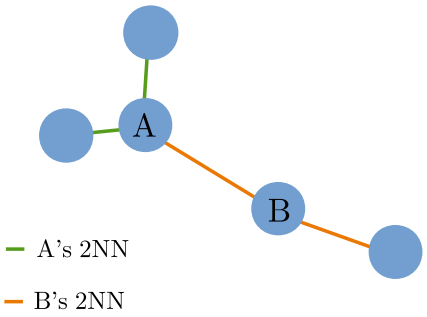
Voting Strategies Let $\mathcal{F}$ be the set of all features (extracted from all the images in the database). An image I_j is represented by a BoF $B_j = \{b_{sj}\}_s$. To compute the similarity between an image I_j and an image I_k with voting strategies, neighborhood of feature vectors is used. For each b_{si}, each $b_{sk} \in N(b_{si})$, where $N(b_{si})$ is a neighborhood of b_{si}, increments the similarity measure by one. Let $sim(x, y)$ be the similarity function, and 1_P the function equal to 1 if the predicate P is true and 0 otherwise. The computation of $sim(I_j, I_k)$ is then given by Equation 1.6.

$$sim(I_j, I_k) = \sum_s \sum_{s'} 1_{b_{s'k} \in N(b_{sj})} \tag{1.6}$$

Voting strategies are classically based on the R Nearest Neighbors (RNN) or k Nearest Neighbors (kNN) neighborhoods. $RNN(x)$ is the set of elements being at a distance less than R of x. Using such a neighborhood definition makes sim a symmetric similarity function. Indeed, if $dist(b_{si}, b_{s'k}) \leq R$, then $dist(b_{s'k}, b_{si}) \leq R$. So if $b_{si} \in RNN(b_{s'k})$ then $b_{s'k} \in RNN(b_{si})$. Under this context, I_j and I_k can be interchanged in Equation 1.6 without changing the result.

When considering kNN however, the similarity function is not symmetric. $kNN(x)$ are the k elements that are the closest to x. Figure 1.7 shows a simple example of the kNN asymmetry for $k = 2$, $A \in 2NN(B)$ but $B \notin 2NN(A)$.

Figure 1.7: kNN is not symmetric. Here if $k = 2$, $A \in 2NN(B)$ but $B \notin 2NN(A)$



Those voting strategies are fairly simple, and any feature vector within the neighborhood of another has the same contribution to the similarity measure. This is not

an issue for copy detection, in which every point has the same importance and high discrimination is expected. In a similarity search or general classification task however, some points better represent objects, and thus could receive more importance. In addition, the closest the vectors are, the better it is. Taking those information into account would help in achieving better semantic comparisons. Those observations lead to the construction of other metrics dedicated to BoFs comparison [Grauman & Darell 2005, Charikar 2002], but those are out of the scope of the introduction made here on the subject.

A last class of functions used to assess the similarity between image descriptions must be mentioned: the kernels functions. Those are dedicated to any image representation composed of one vector, histogram or not, and are widely used because of their properties. They are presented right below.

1.2.3.3 Kernels

Let $\mathcal{X}$ be the space in which the raw visual features are defined, also called the *input space*. There are two ways to define a kernel function. The first one is: if there exists an injection ϕ mapping any $\mathbf{x} \in \mathcal{X}$ to a vector $\phi(\mathbf{x})$ in an Hilbert space, called the *feature space*; then, the function k defined by Equation 1.7 is a kernel function. k is defined by a dot product in the *feature space*.

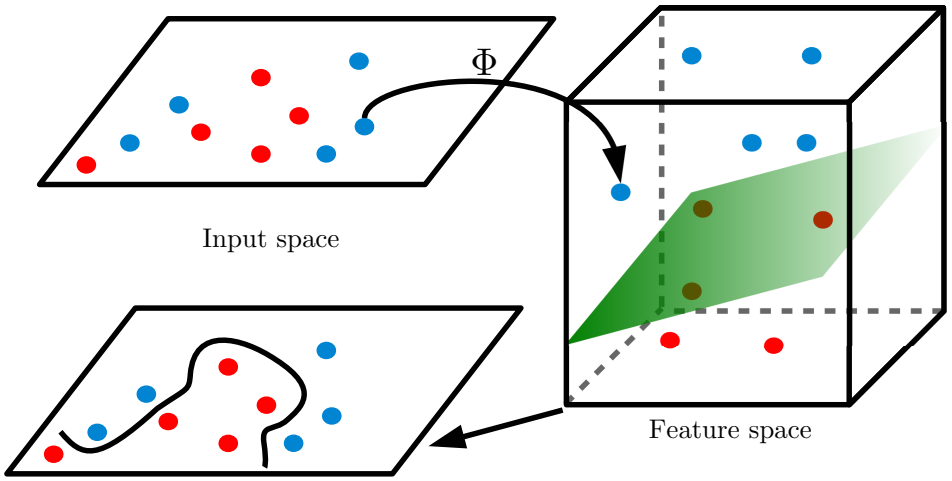
$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle \tag{1.7}$$

Linear classification algorithms most of the time uses the dot product to compare data and take their decision. As kernel functions are dot products in a *feature space*, they can be used instead of the dot product in the *input space* to build a decision using those linear classification algorithms. The linear separation is then built in the *feature space* instead of in the *input space*. This allows linear classifiers to operate on non linearly separable data in the *input space*. Indeed, as shown in Figure 1.8, where we want to separate the orange dots from the blue dots, non-linearly separable data in the *input space* space can become linearly separable in the *feature space*. The greater the dimension of the *feature space*, the higher the probability that the data can be linearly separable in it. This is the so called kernel trick. Then finding the linear separation in the *feature space* results in finding the non-linear separation in the *input space* [Cover 1965].

Building a function ϕ to then express the kernel as a dot product requires some computation overload (first computing the image by ϕ of both vectors before computing their dot product). Hopefully, kernel functions have a second equivalent definition: a function k on $\mathcal{X} \times \mathcal{X}$ is a kernel function if and only if it is a semi-definite positive function, i.e. it is symmetric and verifies Equation 1.8. Using this definition, a kernel function can be defined and computed without even knowing the function ϕ and the *feature space*.

$$\forall \{\mathbf{x_i}\}_{i=1\ldots n} \in \mathcal{X}, \forall \{\alpha_i\}_{i=1\ldots n} \in \mathbb{R}, \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0 \tag{1.8}$$

Figure 1.8: Non linearly separable data in the input space becomes linearly separable in the feature space. Finding the separating hyperplane in the *feature space* allows to find the non-linear separation in the input space.



This heavily reduces the cost to use such functions, and even allow the use of *feature space* with infinite dimensions depending on the kernel function. In addition, computing a kernel function is most of the time not very costly, such as for the popular polynomial kernel (Equation 1.9) or the Gaussian kernel (Equation 1.10) for example. Thanks to all those assets, kernels have been used a lot in classification tasks and in CBIR, as for example by [Perronnin *et al.* 2010] and in most of the work using SVMs as the classification algorithms. They are presented in the next subsection together with some of the most popular and most recent algorithms used to classify images in CBIR.

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^q \tag{1.9}$$

$$k(\mathbf{x}, \mathbf{y}) = \exp^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}} \tag{1.10}$$

1.2.4 Popular algorithms

Packed up with a proper representation of images and a similarity measure, the road to mining the images is wide open. This thesis focuses on image classification, therefore, this subsection concentrates on giving an insight of some popular algorithms in this context. The classical k-Nearest Neighbors and k Means algorithms are treated first as one of the first techniques used, then a brief state of the recent use of Artificial Neural Networks in CBIR is presented. Finally, a deep description of SVMs is given as they are the most popular algorithms in CBIR and are one of the best performing techniques for years in the domain.

1.2.4.1 K-Nearest Neighbors (kNN)

One of the first approaches to solving CBIR problems was to search for neighbors of images. A lot of effort has been put in the image description and similarity measure so that those elements holds some semantic level meaning. Because such effort has been put in them, those tools may be able to separate images correctly, then, looking at the closest images to a given example should return similar images. Those assumptions lead to the use of the kNN algorithm to perform image retrieval and the k-means algorithm to perform image classification [Chang *et al.* 2012]. kNN is an algorithm retrieving the k nearest neighbors of a points, those neighbors are the result of an image retrieval search when used in CBIR. k-means cuts the space into k partitions, finding the center of each partitions (the k means of the partitions), called the centroids. In image classification, once the k centroids have been found, any image is associated to the partition corresponding to the closest centroid. Those algorithms heavily rely on the similarity measure, and dissimilar images are automatically put in different categories. The community quickly realized that those methods were not performing that well in real situations. Indeed, image representing the same objects have sometimes dissimilar representations. From there, some people worked at improving the image representations and similarity measures, leading to numerous works, some of which have been presented in Section 1.2.2 and 1.2.3 for image representations and similarity measures respectively. Others thought that whatever the representation, it would never be able to put all relevant images to a given query close to each other every time, and therefore they worked on new learning methods to solve this problem. This lead to a lot of different algorithms to tackle several tasks in CBIR, two of those are presented below for image classification.

1.2.4.2 Artificial Neural Network (ANN)

Recently, ANN techniques emerged in the CBIR context, most of the time using image representations built with deep learning, such as [Wan *et al.* 2013]. This is done mainly to use the same kind of algorithm from the description extraction to the final results. Indeed, deep learning techniques are a kind of artificial neural network and some people believe that using neural networks to mine data which representation are built from neural networks would work better. Those systems have shown really good results [Wan *et al.* 2013], but latest techniques using SVMs as the search engine with image representations built with deep learning outperformed ANNs [Tang 2013]. This shows that SVMs are still up to the task and able to adapt to this new image representation, still performing better than other techniques. A lot has been told about them since the beginning of this thesis, it's now time to introduce and explain them since *they are one of the core elements of the system proposed in this thesis.*

Figure 1.9: Possible separating planes

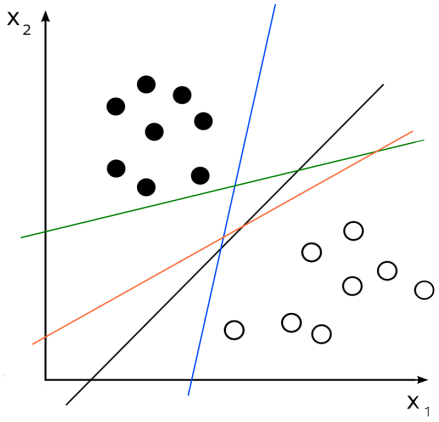
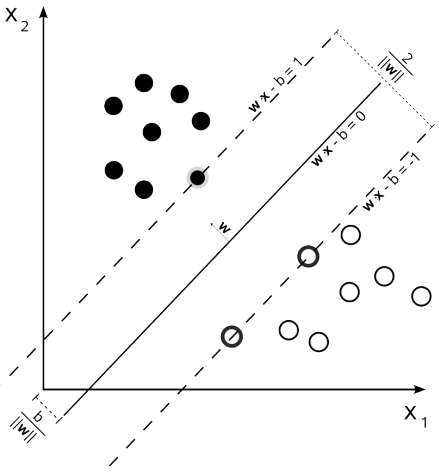


Figure 1.10: SVM margin illustration



1.2.4.3 Support Vectors Machines

Introduced in 1995 by [Cortes & Vapnik 1995], Support Vector Machines (SVMs) encountered an increasing interest in the CBIR community [Gorisse *et al.* 2010, Perronnin *et al.* 2010, Hoi *et al.* 2008], and have even been presented as a good classification method in general [Fernández-Delgado *et al.* 2014]. They are deeply explained in the following to better understand how they work and the challenges are presented in the next section.

SVM Basics and Problem Definition SVMs are originally a linear binary classification method. Linear classifiers separate classes of data using a linear separators, i.e. hyperplanes. While several hyperplanes can separate a given set of training data (the different lines in Figure 1.9 all separate the black dots from the white dots); SVMs compute the hyperplane which is the furthest away from the examples they are given to learn from (Figure 1.10). This hyperplane is unique and is called the maximum margin separator. Indeed, given the distribution of points in Figure 1.9, if we expect new points to follow the same distribution, the green line separator is more likely to misclassify black points than the maximum margin separator represented by the black line. So, even though the empirical loss (the portion of known examples that are misclassified) is the same, the maximum margin separator presents better generalization properties. Studies showed that this observation holds true in general and that the maximum margin separator minimizes the generalization loss, making SVMs a better generalization method compared to methods working on the empirical loss only to compute their separators.

To formalize the problem a bit, data to be classified are expressed in an input

space $\mathcal{I}$. The set of applicable classes is $\mathcal{Y} = \{-1, 1\}$ (binary classification). The linear classification function y is expressed by Equation 1.11. Equation 1.12 is then the equation of an hyperplane called the *separating hyperplane*.

$$\begin{aligned} h(\mathbf{x}) &= \mathbf{w}^T \cdot \mathbf{x} + w_0 \\ y(\mathbf{x}) &= sign(h(\mathbf{x})) \end{aligned} \tag{1.11}$$

$$h(\mathbf{x}) = 0 \tag{1.12}$$

SVMs build the separating hyperplane with the biggest margin. Given a set of labeled data $X = \{\mathbf{x_i}, y_i\} \in \mathcal{I} \times \{-1, 1\}$, the margin is given by Equation 1.13.

$$margin(h, X) = \min_{x \in X} \frac{h(\mathbf{x})}{\|\mathbf{w}\|} \tag{1.13}$$

$$\arg\max_{\mathbf{w}, w_0} \left(\min_{x \in X} \frac{h(\mathbf{x})}{\|\mathbf{w}\|} \right) \tag{1.14}$$

Maximizing the margin is then solving the maximization problem given by Equation 1.14 Without going in the details, this problem can be transformed into the dual problem exposed in Equation 1.15 using Lagrange multipliers and the Kuhn-Tucker's conditions. This is a quadratic optimization problem with respect to $|X|$ and several algorithms exist to solve it.

$$\arg\max_{\alpha} \left(W(\alpha) = \sum_{i=1}^{|X|} \alpha_i - \frac{1}{2} \sum_{i=1}^{|X|} \sum_{j=1}^{|X|} \alpha_i \alpha_j y_i y_j \langle \mathbf{x_i} \cdot \mathbf{x_j} \rangle \right) \tag{1.15}$$

It is important to note that only the vectors sitting at the edge of the margin are used to define the separating hyperplane. This is an important property of SVMs, and those vectors are called *support vectors*, hence the name of the method. This is a consequence of the Kuhn-Tucker's conditions used to transform the optimization problem in Equation 1.14. In Figure 1.10, the *support vectors* are represented with a thick black or gray border while other points are part of the training material but are not *support vectors*. It can be observed that there are a few *support vectors* among all the learning material given to an SVM, and this observation holds true in most cases.

This solves the SVM training problem. However, only problems linearly separable have been covered so far, the following shows how non linearly-separable problems are addressed using SVMs.

Classifying non Linearly Separable Data with SVMs If data are not linearly separable in their description space, the Kernel trick is used to transpose the data into a higher dimensional *feature space*. As presented previously in subsection 1.2.3.3, Mercer's kernels can be expressed as a dot product in a the *feature space*. As shown in Equation 1.15, the SVM optimization problem relies only on the

dot product. Therefore, considering a Mercer's kernel K, the SVM problem can be generalized as expressed in Equation 1.16.

$$\begin{aligned}\arg\max_{\alpha}\Bigg(W(\alpha) &= \sum_{i=1}^{|X|}\alpha_i - \frac{1}{2}\sum_{i=1}^{|X|}\sum_{j=1}^{|X|}\alpha_i\alpha_j y_i y_j (K(\mathbf{x_i},\mathbf{x_j})) \\ &= \sum_{i=1}^{|X|}\alpha_i - \frac{1}{2}\sum_{i=1}^{|X|}\sum_{j=1}^{|X|}\alpha_i\alpha_j y_i y_j \langle\Phi(\mathbf{x_i}),\Phi(\mathbf{x_j})\rangle\Bigg)\end{aligned} \tag{1.16}$$

Solving this general problem results in finding the separating hyperplane in the feature space, which results in a non-linear separation in the input, or description, space. The key feature here is that the transform Φ from the input space to the feature space does not need to be explicitly defined at any time. The computation of K, which is a relatively simple function, is sufficient to solve the SVM optimization problem and predict the class for any new presented data.

This allows to solve any problem that is linearly separable in the feature space. Though this represents only a few real problems, it sets the main theory behind SVMs and allow the comprehension of how SVMs work. Soft margin SVMs have then been developed to handle non-linearly separable problems in the feature space, allowing some training samples to be misclassified at a given cost in order to improve the overall classification performance.

So, SVMs are able to handle any binary classification problem it is presented to (with varying performances). CBIR can sometimes be considered as a binary classification problem. Mainly when trying to retrieve images relevant to a user's request, in which case images are classified as relevant or irrelevant to the request. However, when trying to classify a database or identifying objects, several possible classes are available. Or SVMs are binary classifiers by nature, the following paragraph shows how they can be used to perform multi-class classification.

Multi-Class Classification Using SVMs To handle the multi-class classification problem, several binary SVMs are used together. Two main methods exist: using one SVM for each pair of classes, and then using a max vote strategy to decide the class of a given sample; or using one SVM for each class, each SVM learning how to discriminate the given class from the rest, and using a best score strategy to decide the class of a given sample.

As stated before, the learning process of SVMs is a quadratic problem with respect to the number of training samples. This is multiplied by the number of SVMs to be learned when dealing with multi-class problems. Therefore, learning a SVM on a huge database is costly, but their performances with small training set sizes are pretty good [Shao & Lunetta 2012], so they can still be learned quite fast by using small training sets. In addition, thanks to the way SVMs compute their decision, only a few training samples are needed to achieve the performances obtained by learning with all the training samples available: the *support vectors*.

Active learning methods have been developed for SVMs based on this fact, trying to maximize the improvement brought by each training sample added to the training set. Their main ideas are presented below.

Active Learning for SVMs Active learning methods for SVMs try to get as close as possible to the *support vectors* when adding training samples to the training set, without explicitly computing them and at a lower cost than their explicit computation. [Tong & Chang 2001] proved that, given a training set, a set of available training samples and data linearly separable in the feature space, the best element to add to the training set is the closest to the current SVM decision. This element is also the most uncertain data, indeed, the closest to the decision, the most uncertain the decision is. Using this result allow to build an active learning scheme in which the closest elements from the SVM decision is added to the training set at each iteration. However, the SVM must be retrained every time one element is added to the training set before being able to chose the next. To speed up the process, it would be better to be able to add several learning samples at each iteration.

[Brinker 2003] showed that when adding several training samples at the same time, an angle diversity between the added samples should be enforced in addition to the closeness to the SVM decision to get a better improvement in the decision. This was used with success by [Gorisse *et al.* 2010] to iteratively select training samples for a SVM to address object category retrieval in an interactive framework for example.

Whilst the performances observed when using SVMs, the next section unveils how this training sample selection strategy can be holding SVMs back, preventing them from achieving potentially better results.

1.3 Challenges

SVMs are one of the best performing techniques nowadays in CBIR, whatever the image description used. Their performances with a small training set size are also fairly good, allowing its use in interactive and active learning strategies too. But actual active learning strategies suffer from one drawback, at least in CBIR, they concentrate on local search to find new training samples, preventing them to identify some misclassified data in some cases. The fine grained classification challenge in CBIR is an illustration of this limitation. It consists in classifying a database of images containing many classes among which some are very similar with respect to the visual description of their content while the semantic information they hold is different (Figure 1.11) and vice-versa (Figure 1.12). This problem arises when the amount of data and the number of classes increase. It has been exhibited in particular in the context of image classification for medium to large databases. Caltech-256 is one of those databases, and Figure 1.11 illustrates inter-class visual ambiguities and Figure 1.12 illustrates intra-class visual dissimilarities extracted from this database.

Figure 1.11: Inter-class ambiguities for categories: BEAR, GORILLA and CHIMP

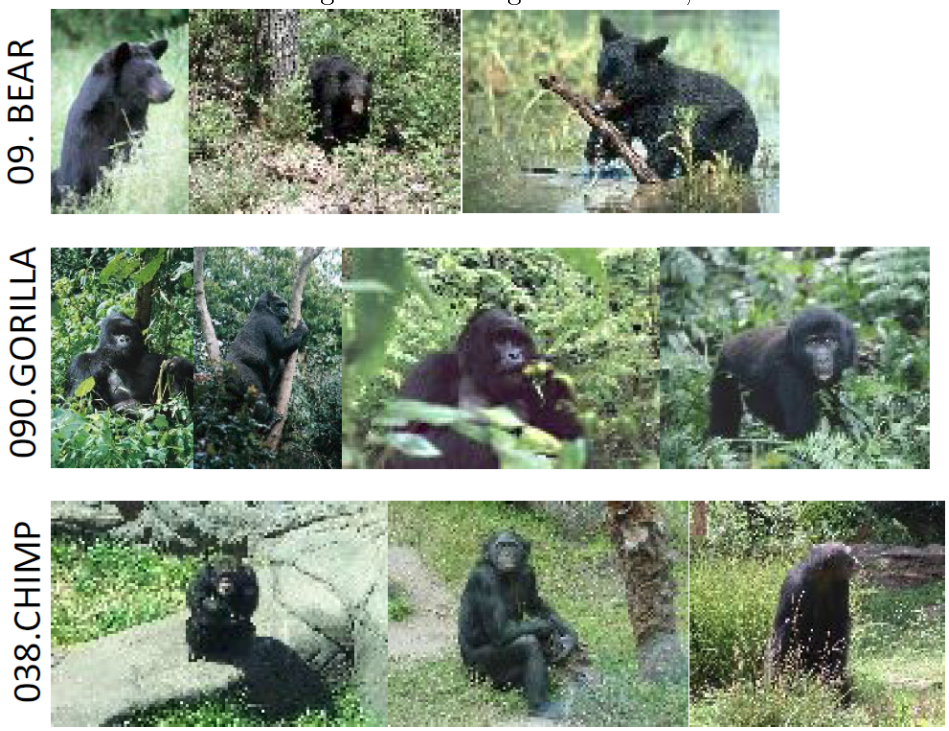


Figure 1.12: Intra-class ambiguities for the category: MUSSELS

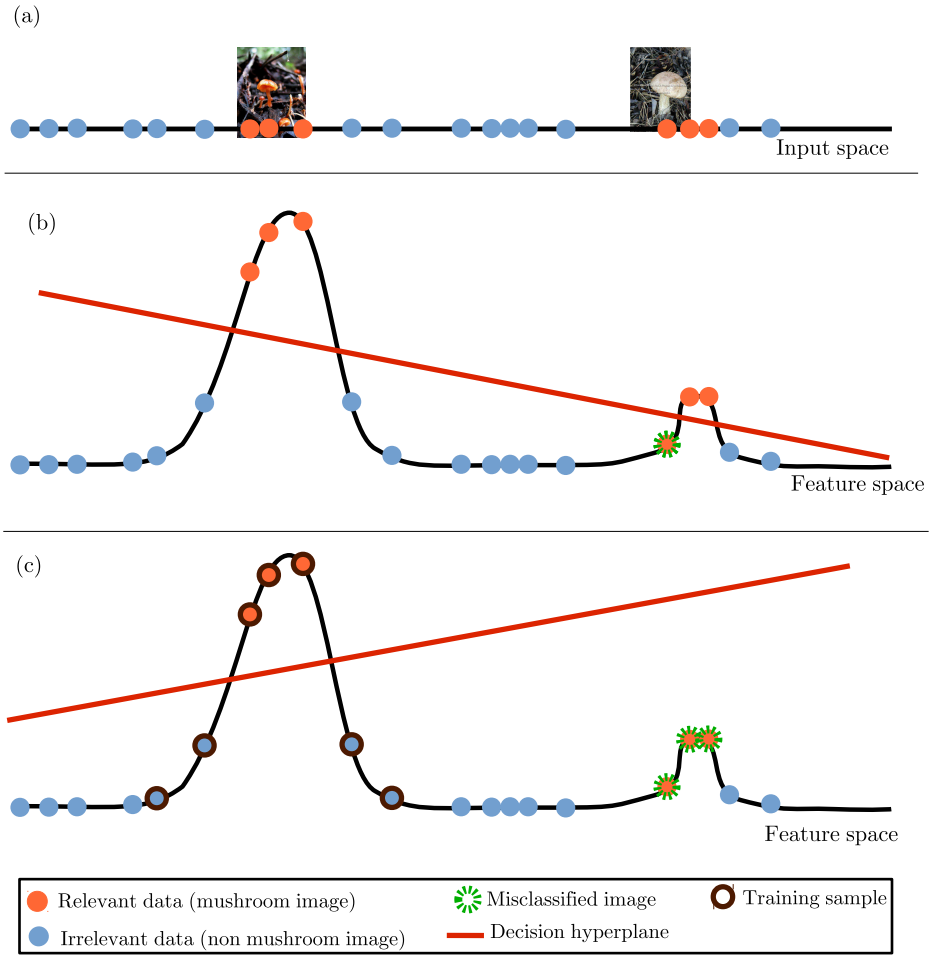
This problem is most of the time linked with the large scale problem (because it arises when the number of classes and images increases), making it worse. One way to deal with very large datasets and proposing low time consuming algorithms is to consider subsets of the available training samples during the learning process. The issue is then how to build those subsets ?

A lot of works are based on random subsets [Balcázar *et al.* 2001]. While this effectively solves the large scale problem, it does not guarantee the best results. In opposition to random strategies, a statistical and a spatial distribution analysis of the data may be used to build better subsets. In this case, computing geometrical information on the whole dataset is still very time consuming and is hard to apply for large datasets [Wang *et al.* 2010, He *et al.* 2010].

Other methods use active learning strategies, selecting iteratively the data from the training sample pool to be added to the SVM training set [Panda *et al.* 2006]. Those active learning techniques are most of the time based on adding images close to the SVM decision to the training set to improve the decision. This allows to build a decision function iteratively and once again solves part of the large scale problems but does not really solve the fine grained classification problem. Indeed, images may not be exactly linearly separable in the feature space, as illustrated by Figure 1.13(a). The blue dots are representing mushrooms while the orange dots are representing other images. In this context, the orange and white mushrooms are in different places of the one dimension description space, represented by a line on the figure, because their color description are different. But orange and white mushrooms are still part of the same class, so they must be put in the same class. Imagine a kernel function is used to project the data in a two dimensions feature space. Parts (b) and (c) of Figure 1.13 represents the images in such a feature space, the black curve representing the image of the description space by the kernel function. White and orange mushroom images can't be linearly separated from other images in this feature space, indeed, the best hyperplane separator (a line in 2 dimension) is represented in red on Figure 1.13(b) and it stills misclassify one white mushroom image. Imagine a SVM training process to separate mushroom images from other images starts with a few orange mushroom images and some other close images as training samples, as depicted on Figure 1.13(c). Then the first decision taken by the SVM, represented by a red line on the figure, misclassifies every white mushroom. In addition, white mushrooms are pretty far from that decision, meaning that known active learning schemes, which concentrate on images close to the decision, will take a long time before identifying them. Until white mushrooms are detected, the decision will not change much, because any added vector is correctly classified and is not in the margin so does not change the decision. This means that a big amount of iterations, and thus of training samples added to the training set, are needed to reach something close to the best separating hyperplane presented on Figure 1.13(b).

Given those observations, existing active learning techniques seem to be lacking something to address the fine grained classification problem: exploration. Exploration is the fact of searching areas of the search space that have not been looked for yet. Random selection answers this in some way, but it would be better to find more

Figure 1.13: (a) Mushroom images forms two separated groups in the description (or input) space. (b) The black line represent the image of the input space by a kernel function ϕ transforming the input space into the feature space. The best possible hyperplane separator in this 2D feature space is represented by the red line. Mushroom images are not linearly separable from other images in this feature space.(c) The hyperplane obtained in the 2D feature space learning a SVM from the thick bordered individuals only is represented by the red line.



clever ways to perform exploration that would allow to improve the performances more; EAs in particular have drawn some attention to perform the task. The next chapter presents why exploration, and EAs, could fit in addressing the fine grained classification problem; how evolutionary algorithms work; and how they were used in some existing works in CBIR.

Chapter 2

Toward the Use of Evolutionary Algorithms to Address the Lack of Exploration in Content Based Image Retrieval

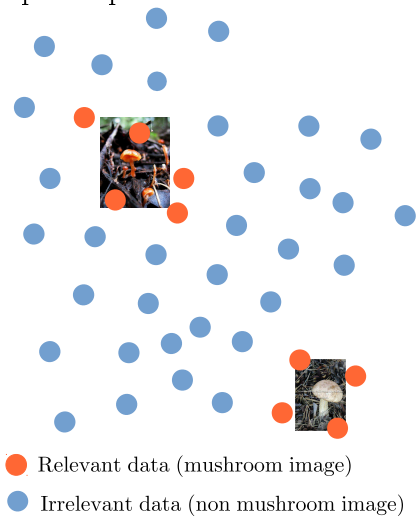
Contents

SVMs, presented in section 1.2.4.3, are one of the best algorithms when dealing with image classification in CBIR. However, with the fine grained classification problem, limitations of known methods based on SVMs arose, as presented in section 1.3. This thesis emphasizes on the use of exploration techniques to overcome some of the limitations of the existing methods. Exploration consists in searching new interesting elements in part of the space that has not been looked for yet. The first section of this chapter explains how exploration is expected to help solving some of the issues of existing techniques. The second section introduces EAs and Genetic Algorithms (GAs) as the exploration technique chosen to bring exploration. Section three then explains how CBIR has been percieved as an exploration problem, the problems that are expected to be solved using EAs and the limitations of EAs in addressing those problems. This chapter ends with an overview of hybrid systems developped to tackle CBIR problems, using both EAs and a local search techniques, which is often a SVM.

2.1 Motivation

2.1.1 Problem

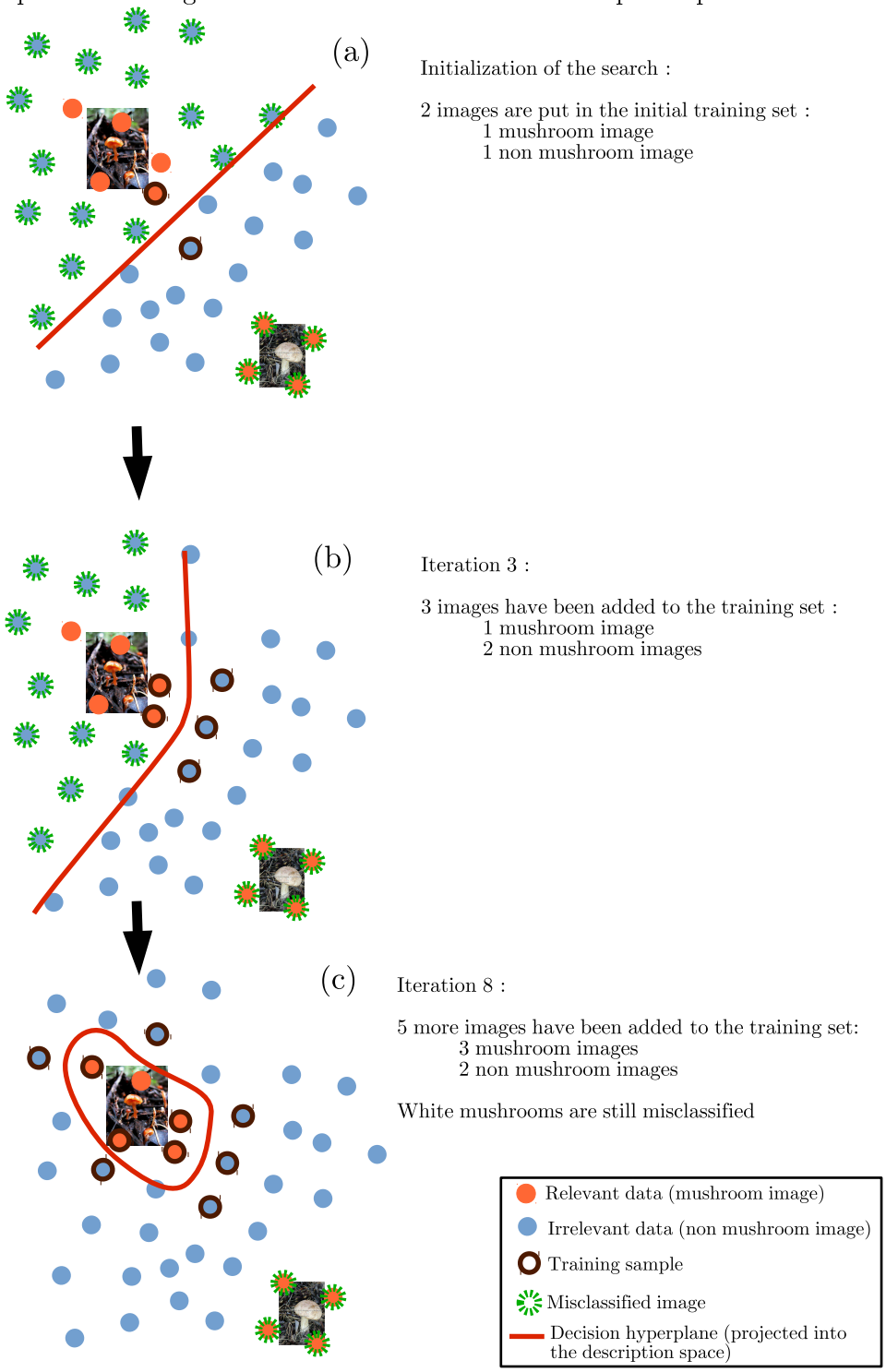
Figure 2.1: The mushroom images (represented by orange dots) are separated into two modes into the description space



Even though image representations and associated similarity measures are built to grasp part of the semantic of images, they are most concerned about the visual similarity of images. However, in a fine grained classification context, visually dissimilar images may belong to the same class of images. Therefore, images belonging to the same class may end up in completely different parts of the image description space. Figure 2.1 is an illustration of such a situation. The description space is a 2 dimensions space represented by the plane, orange dots are representing mushroom images and blue dots are representing other images. The database contains white and orange mushrooms, one picture of each is included in the figure. Because of their visual dissimilarity, in particular their color which is part of their description, we can see on the figure that white and orange mushrooms end up grouped in two disjoints parts of the search space. They form what is called two modes, two disjoint contiguous part of the space containing relevant images, mushroom images in this example. It is expected that they are classified in the same class though.

A SVM classifier is built using an active learning scheme to separate the database into two classes: mushrooms and non mushrooms images. Different steps of the learning process are represented on Figure 2.2, orange dots represents mushroom images, blue dot represents non mushroom images. A thick black border is added to the images that are part of the training set. The red curve represents the SVM decision. Any image, represented by an orange or blue dot, on the same side as training samples representing mushrooms (orange dots with thick black border) is classified as a mushroom by the SVM and any image on the other side is classified as

Figure 2.2: Steps of a learning process based on local search to identify training samples when images forms several modes in the description space



a non-mushroom image by the SVM. A fine dashed green border is added to images misclassified by the SVM to help identify them.

To start the learning process, one of the orange mushrooms together with one non mushroom image are randomly chosen to form the first training set for the SVM classifier, as depicted on Figure 2.2(a). The decision taken by the SVM using this two elements training set is fairly bad, orange mushrooms are well classified but a lot of non-mushrooms are classified as mushrooms too, and all the white mushrooms are classified as non-mushroom by the SVM. It is expected when giving only two learning samples to the SVM.

To improve the SVM decision, more training samples must be added to the training set. An active learning scheme following the guidelines provided by [Tong & Chang 2001] is used. [Tong & Chang 2001] proved that the image bringing the best expected improvement to the decision is the one the closest to the current SVM decision. Following this idea and adding three training samples one by one to the training set leads to Figure 2.2(b). Two non-mushroom and one orange mushroom images have been added to the training set, applying the selection proposed by [Tong & Chang 2001] three times in a row. The classification is improving as less non-mushroom images are misclassified than in Figure 2.2(a). Despite this improvement, none of the white mushrooms have been correctly classified yet, and the SVM decision is getting further from them than in Figure 2.2(a). As the decision is getting further from them, it is likely that white mushrooms will never be added to the training set, since only the closest image to the decision is added at each iteration. Thus it is likely that they will never be retrieved using this technique, or at least a lot of images must be added to the training set before they will be retrieved.

This is confirmed by continuing the experiment, Figure 2.2(c) represents what we could end up with if we continue adding the closest image to the SVM decision into the training set one by one. Here 5 more images have been added to the training set, two orange mushrooms and three non-mushroom images. The SVM decision has improved when compared to previous steps, now no non-mushroom image are classified as mushrooms. But none of the white mushrooms have been correctly classified. In order to identify white mushrooms following this strategy, images must be added to the training set until the closest image to the SVM decision is a white mushroom. This represents a lot of images to add to the training set.

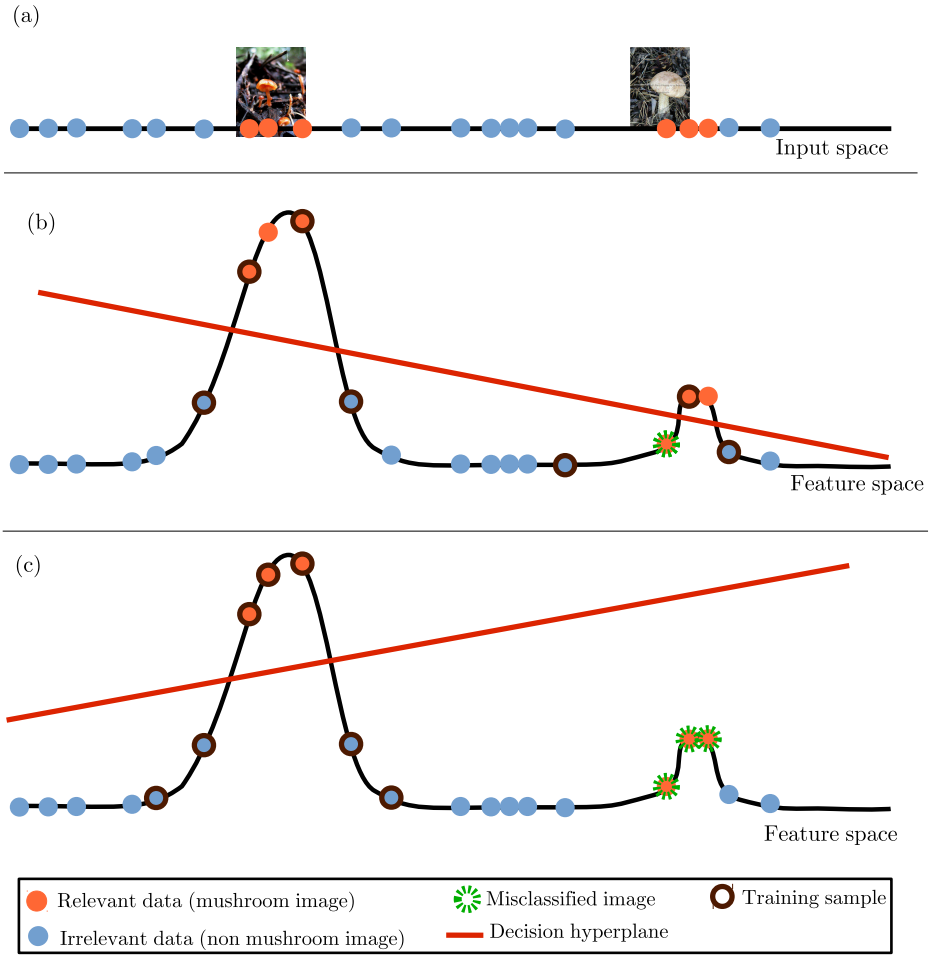
This issue comes from the fact that data may not be linearly separable in the feature space, which infringes one of the assumptions of [Tong & Chang 2001]. And in such a case, the best linear classifier to separate part of the relevant image from irrelevant ones may not be the best hyperplane to separate all the relevant images from other images.

Such a case is represented in Figure 2.3. The input space is considered to be a 1D space and the feature space is a 2D space. Mushroom images are represented in orange while other images are represented in blue. Figure 2.3(a) shows the images in the description space. In Figure 2.3(b) and (c), the black line represents the image of the 1D description space by a kernel function ϕ transforming it into a 2D feature

space. Depending on the training samples given to the SVM (circled in black on the figure), the SVM may identify only one mode of mushrooms (case (c)), or both modes of mushrooms (case (b)). In order to identify both modes, there must be images from the two modes in the training set.

For those reasons, concentrating the search in the vicinity of the SVM decision, which is an exploitation process, without further exploring may lead to miss some relevant data and potentially ending up with a sub-optimal, or even bad, classifier. However, before diving into further explanations about the exploration technique chosen to help solving this issue, a quick introduction to the differences between exploration and exploitation as well as their advantages and drawbacks is given below.

Figure 2.3: (a) Mushroom images forms two separated groups in the description (or input) space. (b) and (c) The black line represents the image of the input space by a kernel function ϕ transforming the input space into the feature space. If images from only one mode are in the training set (c), only this mode is identified while if images from the two modes (b) are in the training set, both modes are identified.



2.1.2 Exploitation Versus Exploration

In a data mining context, exploitation is the process of searching in the vicinity of previously identified data or area of a search space for better solutions to a problem. What is done by most active learning systems based on SVMs can be assimilated to exploitation, since most of them perform local search. [Tong & Chang 2001] and [Gorisse *et al.* 2010] focusing on the data the closest to the SVM decision are just two examples of such techniques.

Exploitation is often used as a refinement method, after the use of a global search method to identify the interesting areas in the search space. Indeed, exploitation is meant for local search, and thus fails at identifying any new areas or relevant data far from its starting search point. Classical active learning schemes for SVM use only exploitation because they consider the data to be linearly separable in the feature space. In this case, the promising area has been mathematically proven to be the area close to the SVM decision [Tong & Chang 2001]. However, data may not be linearly separable in the feature space.In such a case, identifying the promising areas before exploiting them requires exploration.

Exploration is the task of identifying promising areas by examining the whole search space. Exploration techniques are also expected to get out of the local optima exploitation can get stuck into. Even though necessary, this process is time and resource consuming. Therefore the exploration must be handled wisely, else the system could take a very long time to converge. In addition, exploration is dedicated to the scanning of the search space, and thus using it alone often results in investing a lot of time and resources to identify the best solution. Exploration is most of the time quite efficient at identifying loosely the vicinity of good solutions or relevant data, but is less effective when it comes to improving those approximations to reach the concrete best element in the area.

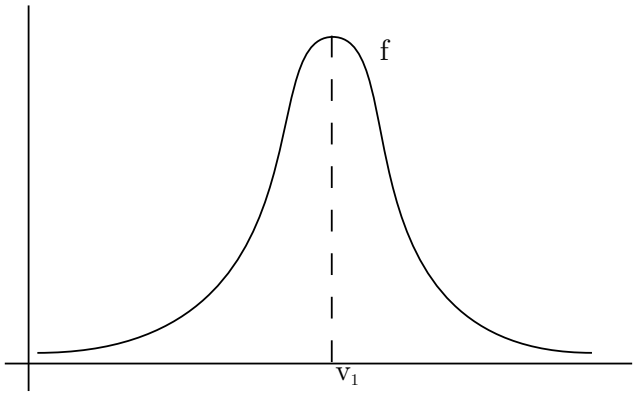
Therefore, a good technique should use both exploitation and exploration, balancing the use of both to reach the expected result as fast as possible. SVM active learning techniques have dropped this exploration part because of mathematical proofs, but those do not hold in every condition, and in particular in the fine grained classification challenge. Reintroducing an exploration component to restore the balance between exploration and exploitation seems important. In this thesis, EAs, and in particular GAs, have been chosen so as to bring back the exploration part dropped in most of the active learning schemes dedicated to SVMs. Those methods are known for their good balance between exploration and exploitation and thus sounded like good candidates. The next section introduces EAs and gives a detailed explanation of GAs. The following section then exposes their use in a multi-modal context such as the one encountered in the fine grained classification.

2.2 Evolutionary Algorithms

EAs are population-based meta-heuristic optimization algorithms. Optimization algorithms are dedicated to solving optimization problems. An optimization problem

consists in finding the parameter vectors for which the optimal value (could be a maximum or a minimum depending on the problem) of one or several functions are attained. For example, the optimization problem consisting in maximizing the function f represented in Figure 2.4 consists in finding v_1, for which f reaches its maximum. The function to optimize is often called a fitness function or an objective

Figure 2.4: A function f, reaching its maximum at v_1



in EAs.

EAs cover a wide range of algorithms, including but not limited to GAs [Goldberg 1989], Particle Swarm Optimization (PSO) [Kennedy & Eberhart 1995], Differential Evolution (DE) [Storn & Price 1997] or even firefly algorithm [Yang 2008].

Though quite different, those algorithms share a common structure we introduce in the first subsection.

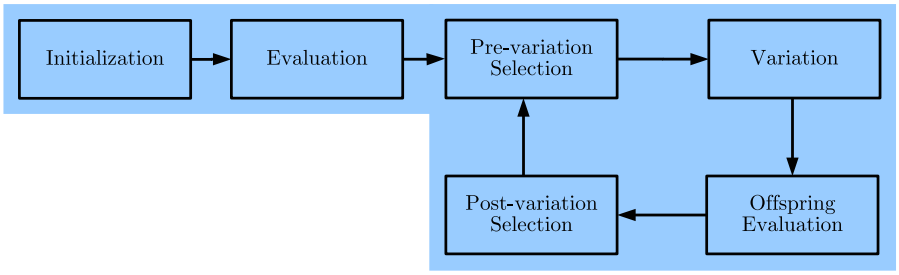
2.2.1 General Structure

As optimization algorithms, EAs are used to solve a given optimization problem P which have its solutions in a solution space, search space, or parameter space S. To solve an optimization problem, those algorithms build and evolve a set of potential solutions called a population. Each element of the population is called an individual. Individuals represent potential solutions to the optimization problem P and thus are elements of S. Each individual performances in solving P can be evaluated using a score function f called fitness function in EAs.

Given those definitions, Figure 2.5 shows the common framework to any EA. First the initialization takes place. The population is initialized, most of the time randomly. Uniform distribution over the search space is often used, in particular when no a priori knowledge about the problem is known. But biased initialization or specific initialization can be used too, taking advantage of a priori knowledge when available and initializing individuals in the vicinity of known interesting areas.

Once the population is initialized, each individual in the population is evaluated

Figure 2.5: Framework of an EA



using the fitness function before the evolutionary loop starts. The evolutionary loop is composed of a pre-variation selection process, a variation process, an evaluation process and a post-variation selection process.

The pre-variation selection process aims at selecting which individuals will be used during the variation process to generate new individuals. Those individuals are grouped as needed by the variation process. In the classical DE algorithm [Storn & Price 1997], each individual of the population is processed at least once through the variation process, as the parent vector also called the target vector in DE. For each parent, the pre-variation process randomly draws 3 other individuals for the evolution process of DE to run. This is a pretty special case of pre-variation selection, using each individual at least once. Most of the time, only part of the population is processed through the variation process to generate new individuals. The selection is usually based on the performance, i.e. the fitness value, of each individual, favoring the best individuals. Generally, every individual has a chance (however small it is) of being selected, even those with a very bad fitness value.s

The variation process then consists in modifying the selected individuals so as to create new individuals. Different variation methods exists, GAs for example use recombination and mutation as detailed in subsection 2.2.2, whereas DE uses differences between individuals and recombination. At the end of the variation process, we end up with two sets of individuals: the individuals from the population; and the generated individuals, called the offspring.

After the variation process, offspring are evaluated and the post variation selection process takes place. It consists in selecting which individuals, from the current population and the offspring, will form the new population. The simplest selection process just keep the offspring as the new population but several other options exist such as replacing some of the worse offspring by the best individuals from the current population for example.

Finally, the new population goes through the evolutionary loop again, which is commonly called a generation. New generations are run until a stopping condition is met. Stopping conditions will widely vary depending on the algorithm, but are often related to resource investment (elapsed time or number of evaluations for example), the quality of the best solution found so far, or a lack of improvements from one

generation to the next.

EAs are most of the time inspired by nature in a way or another [Siddique & Adeli 2015]. A lot of them were developed over the years, we will briefly present some of them.

Evolution Strategies Evolution strategies were introduced by [Rechenberg 1973], with the (1+1)-ES algorithm. It uses only one individual, the parent, and one offspring, and is based on two rules: apply small random changes to all variables of the parent simultaneously to create the offspring; if the offspring is better than the parent (with respect to the fitness function) take the offspring as the next parent, else retain the parent. Variants of evolution strategies have been developed through the years, the most commonly known being CMA-ES [Hansen *et al.* 1995].

Differential Evolution DE was introduced by [Storn & Price 1997]. Each individual of the population is used to generate at least one offspring. The generation of an offspring from a parent, called the target vector, I_{target} in the canonical DE algorithm presents two steps. First, the mutant vector (sometimes also called donor vector) I_m is computed using Equation 2.1. I_1, I_2, I_3 are three different individuals randomly drawn from the population, and F is a parameter of the method. I_{target} is then recombined with I_m using a binary crossover to form I_{trial}, the trial vector, which is the offspring. The binary crossover consists in, for each component of I_{trial}, selecting randomly from the value of I_{target} and I_m, giving a higher probability to the value from I_m. The post variation selection then compares I_{trial} and I_{target} fitnesses and adds only the best one to the next generation's population. This classical DE is named DE/rand/1/bin, which stands for differential evolution, randomly choosing I_1 (also called the base vector) I_2 and I_3, 1 difference computation and binary crossover.

$$I_m = I_1 + F * (I_2 - I_3) \tag{2.1}$$

This algorithm is best applied to real valued continuous problems. The application of a difference of vector to another vector enhance exploration, leading to the creation of a completely new vector. The mutation factor F must be set at a value less than 1 for the population to converge. Indeed, this will reduce the distance between individuals by moving the donor vector I_1 to a position closer than the distance between I_2 and I_3, effectively reducing the distance between individuals if repeated several times. The post selection process ensures that the best individuals are kept at any time, ensuring convergence toward the best solution.

Variants of DE exist using several vector differences (DE/rand/2/bin, $I_m = I_1 + F * (I_2 - I_3 + I_4 - I_5)$ with I_1 through I_5 randomly chosen), or introducing the best solution found so far or the best solution in the current population as the base vector (DE/best/1/bin, $I_m = I_{best} + F * (I_2 - I_3)$ with I_2 and I_3 randomly chosen) or in the difference computation (DE/rand to best/1/bin, $I_m = I_1 + F * (I_2 - I_3) +$

$F_2 * (I_{best} - I_1)$ with I_1, I_2 and I_3 randomly) for example. DE has shown pretty good results in several problems and have been studied a lot recently with different perspectives [Fonlupt *et al.* 2011, Segura *et al.* 2015b, Segura *et al.* 2015a].

Particle Swarm Optimization PSO [Kennedy & Eberhart 1995] is an optimization algorithm that was originally inspired by bird swarms behavior. Each individual, or particle, is given a speed and a position in the search space. Initially, individuals, or particles, are randomly positioned into the search space. The pre-variation selection process keeps every individual of the population (the swarm in this case) to generate the offspring. The variation process consists in updating the speed and position of each particle. The speed is updated using the current speed, the best position the particle has achieved so far, and the best particle in the vicinity of the studied particle, using an equation similar to Equation 2.2. V_{k+1} represents the new speed of the particle, V_k is the current speed of the particle, ω is an inertial parameter, X_k is the current position of the particle, P_i is the best position achieved by the current particle, P_v is the position of the best particle in the vicinity of the current particle, b_1 and b_2 are randomly drawn at each iteration and weight the influence of the vicinity and personal best known positions. We can see that this equation makes the speed evolves to point toward the best known areas of the search space. The position of the particle is then updated by applying its new speed to its actual position: $X_{k+1} = X_k + V_{k+1}$. The post-variation selection usually keeps only the newly generated particles. Using the best solution known by a particle and the best solution in its vicinity only may get a particle stuck to local optima. Indeed, a particle initialized near a local optima with no global optima near it will get stuck into this local optima. Its personal best position will always be near this optima and no better position exists in its vicinity by definition, so its speed will always point toward the local optima and the particle will be stuck. Therefore some variations of PSO also use the global best position found within the whole population to update the particles' speed. However, if the global best solutions found so far is a local optimum, this runs the risk of getting all particles converging and being stuck to this local optimum. Careful management of the speed update process must be made to keep a good balance between exploration and exploitation in this algorithm.

$$V_{k+1} = \omega * V_k + b_1 * (P_i - X_k) + b_2 * (P_v - X_k) \tag{2.2}$$

Firefly Algorithm Firefly algorithm [Yang 2008] mimics the attractive behavior of fireflies swarms, which depends on the light they generate. Each individual represents a firefly an has a position and an attractiveness. As for particle swarm optimization, the pre-variation selection process keeps all individuals. The intensity of light emitted by, and so the attractiveness of, each firefly depends on its fitness function. Each firefly's position is updated by moving it toward every firefly that has a greater light intensity using Equation 2.3. γ is an absorption coefficient that defines how much light intensity decreases with the distance. So the further away

fireflies are to each other, the less they influence their position. The $\alpha_t \varepsilon_t$ component is a small random component allowing some exploration. In the end, firefly algorithm is close to the particle swarm optimization, but uses all better individuals to update one individual's position instead of just two of them. It also incorporates a random term to add exploration capabilities to the algorithm.

$$\mathbf{x_i}^{t+1} = \mathbf{x_i}^{t} + \beta exp(-\gamma r_{i,j}^2) * (\mathbf{x_j}^{t} - \mathbf{x_i}^{t}) + \alpha_t \varepsilon_t \tag{2.3}$$

Many other EAs exist, yet we were particularly interested by GAs. Their balance between exploration and exploitation [Črepinšek *et al.* 2013] fits the needs we identified in the fine grained classification context. In addition, their modularity allows us to change any of their component at will to fit our problem. Indeed, each selection process can be tuned and the variation process of GAs is fairly modular and can be easily changed. GAs are explained in details in the next subsection.

2.2.2 Genetic Algorithms

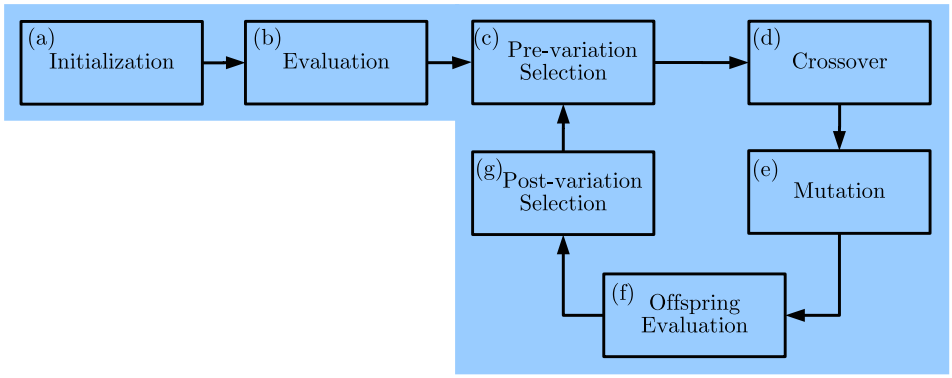
GAs are a specific kind of EAs inspired by Darwin's theory of evolution and the crossover and genes mutations encountered in nature.

Considering an optimization problem, each potential solution to the problem, or phenotype, is represented in the population of a GA by a set of characteristics that can be evolved. This set of characteristics is called the genotype, and is what the GA will evolve. As an example, in the CBIR context, we are searching images relevant to a given query. One way to consider the problem is by thinking of each image as a potential solution, they are then the phenotypes. The GA then needs a representation of those images, representation the GA will evolve and evaluate. One possible representation is to take one of the image representations presented in section 1.2.2. Those representations are real-valued vectors, forming the genotypes of the images in this example. Several other ways to characterize the problem and represent its potential solutions exist, this is just an example of the difference between phenotype and genotype. Genotypes are usually vectors, in the first GAs composed of boolean, but they can be made of real values or any kind of information. The process of determining the genotypes for a given problem is sometimes referred to as encoding the problem. Then a binary-coded problem is a problem for which genotypes are binary-valued vectors, a real-coded problem is a problem for which genotypes are real-valued vectors.

Given those genotypes representation of solutions, the GA then evolves following the global workflow presented in Figure 2.6, which is detailed thereafter.

Initialization The GA process starts with the initialization of a first population of genotypes. The initialization process, step (a) in Figure 2.6, is the same as the one of general EAs initialization. It most of the time uses a uniform random initialization over the search space, and uses problem specific information to bias the initialization toward interesting areas when they are known. Each individual

Figure 2.6: Framework of a GA



of this initial population is then evaluated before the core loop starts (step (b) in Figure 2.6).

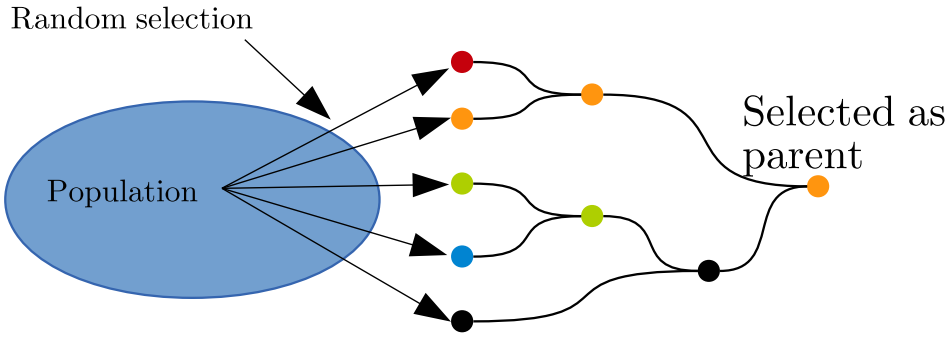
Steps (c) through (g) are the core evolution process, containing the classical pre-variation (step (c)) and post-variation (step (g)) selection processes, the variation process (step (d) and (e)) and the offspring evaluation (step (f)). Several techniques exist for each of those steps, and almost any combination of techniques works as long as the workflow is not changed. This leads to a myriad of possible algorithms, each composed of a different mixture of selections, crossover and mutation. The elements composing those four steps are sometimes called genetic operators in the field of GAs. The goal and some of the principle techniques of each step are presented in the following paragraphs.

Pre-variation Selection The selection of parents for the generation of offspring (step (c)) is usually done randomly, favoring the individuals of the population with the best fitnesses. However, every individual has a chance to be selected. This selection step follows Darwin's principle of evolution: survival of the fittest and is sometimes called mating selection. The fittest individuals, i.e. the ones with the best fitnesses, are the one having the greatest chance to survive and produce offspring. But some individuals that are not the best fitting solutions may manage to generate offspring too. It may be noticed that allowing some worse individuals to be selected can be considered as exploration in some sense. Indeed, those are scattered through the search space, and generating new individuals from them will lead to other new potentially interesting areas of the search space.

Several selection techniques exist that match the presented rules. One of them is the tournament selection represented in Figure 2.7. The idea is to randomly draw k individuals from the population using a uniform distribution (each individual has the same chance of being drawn). Those individuals then compete in a tournament, and the winner is chosen as a parent for the variation process.

Many other pre-variation selection techniques exist for GAs. Once the parents

Figure 2.7: Tournament Selection

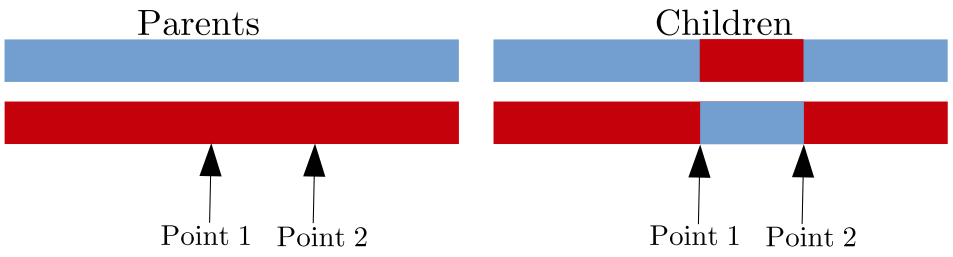


are selected, the variation process starts. This process is composed of two operators in a GA: first the crossover and then the mutation.

Crossover During the crossover (step (c) in Figure 2.7), new individuals are generated by combining several individuals from the selected parents. Most of the time, two individuals are used to generate one or two offspring. The idea of the crossover is to imitate individual's genome recombination encountered in nature during breeding.

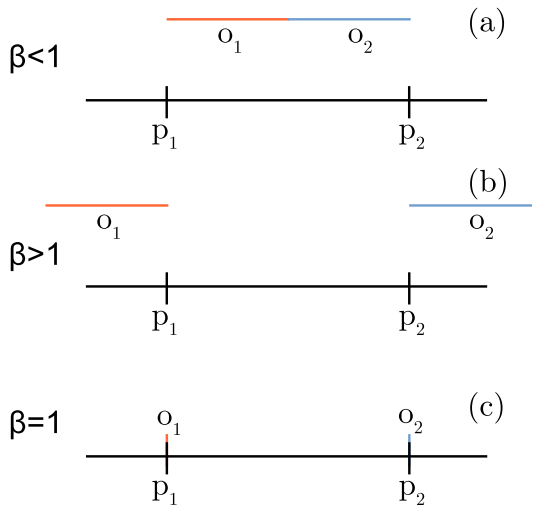
Figure 2.8 shows an example of crossover for GAs, the two points crossover. It applies to two parent vectors and forms two offspring. It chooses randomly, using a uniform distribution, two cutting points within the genotype vector, represented on the left part of the figure. Then, the part of the parent vectors that is between the cutting points is swapped between the two parents to form the offspring as shown on the right part of the figure.

Figure 2.8: Two Points Crossover



Another common crossover, the simulated binary crossover (SBX), consists in computing two offspring, $\mathbf{o}_1$ and $\mathbf{o}_2$ as a combination of two parents using Equation 2.4. This is particularly adapted to real valued genotypes. If $\beta < 1$, then children end up being between the parents (Figure 2.9(a)). If $\beta > 1$, then children end up outside the parent segment (Figure 2.9(b)). And if $\beta = 1$, the children are the parent swapped (Figure 2.9(c)).

Figure 2.9: SBX crossover



$$\begin{aligned} \mathbf{o}_1 &= \bar{\mathbf{x}} + \frac{1}{2} * \beta * |\mathbf{p}_1 - \mathbf{p}_2| \\ \mathbf{o}_2 &= \bar{\mathbf{x}} - \frac{1}{2} * \beta * |\mathbf{p}_1 - \mathbf{p}_2| \\ \bar{\mathbf{x}} &= \frac{1}{2} * (\mathbf{p}_1 + \mathbf{p}_2) \end{aligned} \tag{2.4}$$

Any recombination between individuals may be performed as the crossover operator, therefore, an infinite number of crossover operators exist. Those two are just some of the most common crossover operators, we won't present other crossover operators, we will instead detail the operators used in the algorithms we study when necessary. After the crossover, each offspring obtained is processed through the mutation operator (step (e) in Figure 2.6), which can modify it to form the final offspring.

Mutation Mutating an offspring consists in randomly modifying it. It mimics the genes mutation encountered in nature during breeding. This process enhance exploration, generating completely new individuals within the search space, potentially in unidentified interesting areas. Usually, mutation only slightly modify individuals, but it is better if it has a chance of generating any vector in the search space from any individual. Indeed, the capacity of the mutation operator to generate any point of the search from any other point is one of the conditions to the proof of convergence of GAs presented in [Zitzler *et al.* 2004]. The other condition is linked to the post-variation selection process and will be presented in the next paragraph.

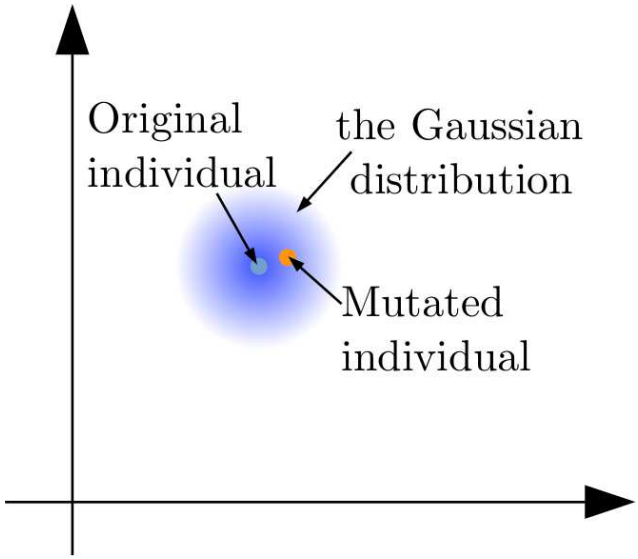
As for the crossover operator, an infinite number of mutation operators exist. Any process that modifies a given individual to create a new one can be considered as a mutation operator. Usually, it has a high chance of not modifying or just slightly

modifying the individual, and a really small chance of completely changing the individual. We will detail only two simple mutation scheme here to give an insight of some common practice. The mutation schemes used by the studied algorithm will be presented when needed.

The most simple mutation operator consists in giving a really small chance (usually less than 3%) for each individual to be reset. Which means that each component of its genome are drawn at random to form the mutated individual. This is called the reset mutation. Though effectively able to generate any element of the search space, this mutation doesn't take advantage of all the evolution process that was run before the mutations.

Another example of mutation which is softer is the Gaussian mutation. It is made for real valued genomes. Each component of the genome have a chance of being modified. The modification is done by drawing randomly a new value for the gene in a Gaussian distribution centered on the actual value of the gene with a small standard deviation. The blue gradient circle in Figure 2.10 illustrates the probability of the position of the mutated individual when using the Gaussian mutation on the original blue vector. A sample result of this mutation is represented by the orange point. As a Gaussian distribution is used to select the new values, the mutation

Figure 2.10: The Gaussian Mutation. The Gaussian distribution is illustrated by the blue color gradient, giving the probability for the mutated vector to be at each point of the space.



covers the whole space and any individual of the space can potentially be generated, even if the probability is very low for some of individuals. The probability is very high that the mutated individual falls close to the offspring position from which it is generated however, thanks to the Gaussian distribution.

Once every offspring has been processed through the mutation operator, and has potentially been modified, we have a set of newly generated offspring ready to be use as well as a current population. Individuals in the current population have already been evaluated, but the newly generated offspring have not. In order to perform the post-variation selection, we need every individual to be evaluated. Therefore, the evaluation of offspring is performed as shown in step (f) of Figure 2.6. The last step of the evolution loop is then the post-variation selection process (step (g) in Figure 2.6) which consists in selecting which individuals from the population and the offspring will form the population for the next generation.

Post-variation Selection Process GAs usually run with a constant population size. And anyway, we don't want to have a constantly growing population. Therefore, the algorithm must select which individual to keep in the current population and the offspring to form the new population. This selection process mimics the natural selection of nature, favoring the fittest individuals. It is sometimes referred to as the environmental selection in the context of GAs. Favoring the fittest individuals enhance the exploitation by keeping more individuals around interesting areas than in other places.

An almost infinite number of possibility exist to select which individuals will form the next population. The simplest one consists in keeping only the offspring. They are the youngest individuals, so they will replace individuals in the population as the individuals of the current population die, following the cycle of life. Although inspired by nature, this process does not guarantee the convergence of the algorithm.

The second necessary condition in the proof of convergence performed by [Zitzler *et al.* 2004] is that the environmental selection keeps the best solution from the current population and offspring as part of the population for the next generation. This is referred to as elitism, and can be applied to more than one individual (even if the convergence is guaranteed as long as at least the best individual is kept).

Most selection schemes use a combination of random selection and elitism. Some keep only a constant number of best individuals to put in the new population, others are drawn randomly from the current population and the offspring. Other schemes keep only the best individuals from the population and offspring to form the next population [Deb *et al.* 2000]. We won't give more environmental selection example here. However we will detail the methods used in the studied algorithms when needed.

Once the new population is created, a new generation starts. The new population is processed through the the generation loop (step (c) to (g) of Figure 2.6) to form the next generation, i.e. the population issued from the next generation loop. This process continues until a stopping criteria is reached, which are the same as those previously presented for EAs.

This ends the introduction to GAs. The variation process of this algorithm promotes the exploration, while the selection steps promote exploitation. The next

sections focuses on how those algorithms can be used to help addressing CBIR tasks and in particular the optimization problem identified in the fine grained classification problem.

2.3 Addressing Fine Grained Classification Using Evolutionary Algorithms

Even though SVMs are one of the best techniques in CBIR and in particular for image classification, several works are using EAs as a way to address CBIR tasks. In particular when building interactive systems, exploration can be key to identify the images relevant to the user. Then using EAs allows to get some exploration, while classical SVM techniques are most of the time lacking it as discussed in section 2.1 A quick review of some of those techniques is presented first below. Then, the particular optimization problem associated with fine grained classification is presented in section 2.3.2. Some GAs adapted to this kind of problems are detailed and studied to assess their relevance in the context of fine grained classification in CBIR. As it turned out that those algorithm do not have good enough performances to tackle the problem alone, combining them to the SVM to form a hybrid system is considered. Therefore, section 2.4 focuses on hybrid methods as a good option to address the flaws of the classical SVM based techniques. A quick survey of hybrid systems dedicated to CBIR is presented, concluding with a preliminary work that lead to the hybrid framework developed in the next chapter.

2.3.1 CBIR Addressed Using EAs Only

The iterative structure of EAs presented in the previous section makes it easy to build interactive systems from it. The fitness function (or part of it) is replaced by evaluations provided by the user on individuals of the population, the system then optimizes the user's preferences. Those methods including the user in the process are called interactive evolutionary algorithms [Takagi 2001]. Taking advantage of this, several works use EAs to perform interactive image retrieval. [Broilo & De Natale 2009] used PSO to do so for example. The user annotates some images as relevant or irrelevant to start the process. The system then evolves particles, representing potential image description, using the distance of particles to the representation of annotated images as the fitness function. The closer to the relevant and the further from the irrelevant the better. Evolved particles most of the time does not match any image from the database, therefore a matching technique is used, taking the image with the closest representation to each particle as the corresponding image. Newly retrieved images are then presented to the user for annotation. The process continues, iteratively building a set of relevant images. As the user evaluates images at each iteration, if the convergence is too slow user's fatigue might be an issue in this system.

Other works were based on Interactive Genetic Algorithm (IGA), such as

[Lai & Chen 2011] who present a new IGA based image retrieval system and also make a brief survey of the use of IGA in CBIR. The goal of this work is to interactively retrieve images from a query image provided by the user. In their work, each individual of the GA's population represents an image. The genome is the image representation vector composed of color and texture descriptors. Those are very simple descriptors and they don't bring as much high level information as SIFT descriptors presented in chapter 1 for example. No mutation was used in the GA used, however, as individuals generated after crossover did not match any image in the database, a matching technique is used so that individuals are always representing images. After the crossover, the image with the representation vector the closest to the generated individual is selected to be the offspring instead of the generated individual. The new individuals are then ranked to be presented to the user for evaluation before the next generation starts. The fitness function is a weighted sum of the user's evaluations given at each generation and a similarity function on the image representation, both having the same weight. Giving the same weight those two components is questionable, a multi-objective algorithm could have been used, or more importance could have been given to the user's evaluation. Indeed, user's preference is what is the most important, even though at the beginning of the search, not much information about it is available. [Lai & Chen 2011] presents the population to the user at each generation for feedbacks, and the process continues until the user is satisfied. The user's fatigue must be taken into account in such a system, with a population of 20 individuals and a 10 levels scale to give feedbacks, user shortly feels the fatigue and provide inaccurate feedbacks or simply stop using the system. Therefore ways to reduce user's involvement must be found. Despite the basic representation of images and user's fatigue, the results show that the use of the GA to explore the search space brought some benefit over a simple nearest neighbors search performed from the image description. Experiments were done on a small dataset with only 1000 images, so the improvements are expected to be even better on bigger datasets, in which exploration is even more important. This encourages the use of GAs to explore the search space.

In EAs dedicated to CBIR, individuals most of the time represent images. As shown in the two methods presented above, after the variation process, the generated individuals often does not represent any image from the dataset. This is due to the fact that search space, which is the set of images in the database, is extensionally defined. Meaning that the elements are known and listed. However, the space in which image descriptions are expressed is most of the time a continuous space, and in any way in an intentionally defined space. Most of the EAs are made to work in intentionally defined spaces and thus generate individuals in the global description space without taking into account the real search space, thus the need of matching techniques. [Johnson 2012] worked on the expression of crossover and mutation operators in an extensionally defined search space. A similarity function returning a set of close elements for any point in the underlying intentionally defined space must be provided. The authors come up with a mutation in the form of a matching, picking randomly a neighbor in the ones provided by the similarity

function, with a higher probability for the closest neighbors if this information is provided by the similarity measure. As for the crossover, they propose to either generate a proxy individual by using classical crossover operators and then use the similarity measure from the generated point, or to find an element similar to both parents at the same time. They have done some simple experiments on the mutation and crossover operators they defined on existing CBIR systems. But this work is mainly a formalization of what has been done in most existing works, including the ones presented above, to handle extensionally defined space. No new scheme to handle extensionally defined space is proposed and no deep analysis of the proposed operators is made. It is however a good formalization and a summary of the techniques used to handle extensionally defined spaces.

This gives an overview of several systems dedicated to CBIR and using only EAs to address it. In this thesis, the fine grained classification problem is the problem to be addressed in particular. The following subsection briefly describe and recall the optimization problem linked with fine grained classification before explaining and studying some GAs developed to tackle such problems.

2.3.2 Identified Optimization Problem

As shown in Section 2.1, one of the main characteristics of fine grained classification is that images belonging to the same class may be scattered in different modes over the search space. Then, identifying the images from a given class can be considered as a Multi-Modal Optimization Problem, i.e. an optimization problem having several solutions (several parameter vector with the same optimal fitness function value) scattered through the search space.

So as to solve Multi-Modal Optimization Problems, a lot of techniques are using multi-objectivization. Multi-objectivization consists in transforming a single objective problem into a Multi-Objectives Optimization Problem which solutions are also solutions to the initial problem. Then solving the Multi-Objectives Optimization Problem allows to find solutions to the initial problem. Doing so, it is expected that the resulting Multi-Objectives Optimization Problem is easier to solve than the initial problem, else the transformation is worthless.

Multi-Objectives Optimization Problems consist in finding the parameter vectors optimizing several fitness functions, called objectives, at the same time. Multi-objectivization is a common practice when solving Multi-Modal Optimization Problems because Multi-Objectives Optimization Problems are studied for quite some time and are often better solved than Multi-Modal Optimization Problems as of today. Most of the time, the Multi-Objectives Optimization Problem is built by adding an objective to enhance diversity alongside the objective composed of the multi-modal function to be optimized. However, the Multi-Objectives Optimization Problem is sometimes built differently.

Because multi-objectivization is used a lot when dealing with Multi-Modal Optimization Problems, and to better understand the GAs dedicated to Multi-Modal Optimization Problems presented in section 2.3.2.2, Multi-Objectives Optimization

Problems are presented in details first in the following section.

2.3.2.1 Multi-Objectives Optimization Problems

Multi-Objectives Optimization Problems consist in optimizing two or more functions, or objectives, at the same time. It can be formalized as shown in Definition 1 (here presented in a minimization context but can easily be generalized). Solving a Multi-Objectives Optimization Problem consists in finding all elements from $\mathcal{S}$ that provide the best trade-offs between the different objectives f_i. $\mathcal{S}$ is often referred to as the parameter space while $F(\mathcal{S})$ is the objective space.

Definition 1. Multi-Objective Optimization Problem
Let $F = [f_1, f_2, ..., f_n]$,
$minF(x), x \in \mathcal{S}$ is a multi-objective optimization problem.

When dealing with such problems, the comparison of individuals performances is harder. Indeed, when an individual a performs better that another individual b in all the objectives, then a is obviously better than b. But what if a performs better than b in only some objectives, and worse in others ? A comparison relation has been built to compare individuals in the objective space in the context of Multi-Objectives Optimization Problems: the Pareto dominance.

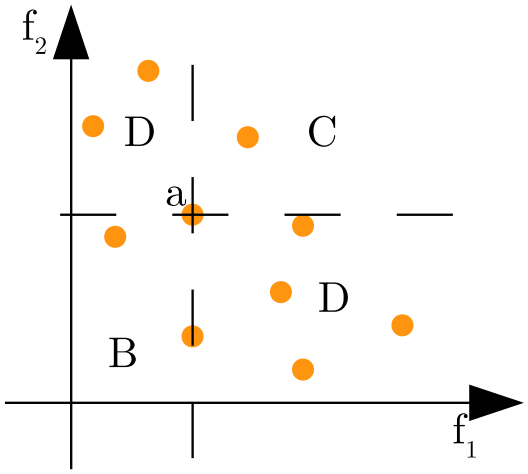
Pareto dominance The Pareto dominance is a partial order relation, detailed in Definition 2 in a minimization context. It basically says that an individual a is better than an individual b if and only if it is at least as good as b in all objectives and strictly better than b in at least one objective. If a is better than b in some objectives and worse in others, then a and b are considered to be not comparable with respect to the Pareto dominance. Based on this dominance relation, the solution to a Multi-Objectives Optimization Problem is the set PS_{opt} of non-dominated elements of $\mathcal{S}$, also called the optimal Pareto set. The image by F of this set $F(PS_{opt})$ is called the optimal Pareto front.

Definition 2. Dominance relation
For $(a, b) \in \mathcal{S}$, a dominates b in a minimization context, also noted $a \prec b$, if and only if:
$\forall i \in [1, n]$, $f_i(a) \leq f_i(b)$ and
$\exists i \in [1, n]$, $f_i(a) < f_i(b)$

Figure 2.11 shows how this dominance relation acts in the objective space for a two objectives minimization problem. The two axis are representing the two objective functions. The individual a dominates any individual in the area marked C, it is dominated by any individual in the area marked B. Any individual in the areas marked D are not comparable with a.

This dominance relation allows to compare individuals between each others, and even allows to rank them. This ranking is important as EAs need a way to know which individuals are better than other during the evolution process, and in particular for the selection steps. This ranking is called the non dominated sort.

Figure 2.11: Relation of individual a with other individuals in the objective space using the Pareto dominance in a 2 objectives minimization context. a dominates elements in area C, it is dominated by elements in area B and it is not comparable with elements in area D.

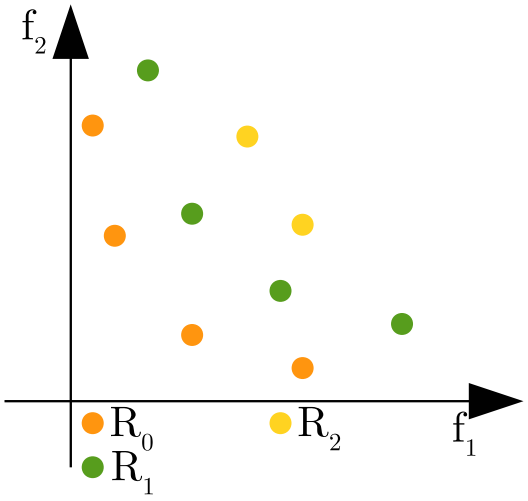


Non dominated sort As the Pareto dominance is a partial order relation, several individuals may not be comparable. Therefore, a population of individuals is ranked into subsets of incomparable individuals called fronts which are assigned a rank. Figure 2.12 represents the result of such a ranking in a minimization context with two objectives. The set of non-dominated individuals in the population forms the first front, R_0, they are colored in orange on the figure. They are the best individuals in the population, since no other individuals dominate them. They can't be compared between each other, so they are all assigned the same rank: 0. This front is also called the Pareto front of the population.

Those individuals are then removed from the population (just for the time of the ranking procedure), and the non-dominated individuals from the rest of the population forms the second front, R_1. They are colored in green on the figure and form the rank 1. The process continues until all individuals have been assigned to a front with an associated rank. This ranking process is called the non-dominated sort. It is often used by EAs dedicated to Multi-Objectives Optimization Problems to compute the probability for each individual to be selected during the two selection steps [Deb *et al.* 2000].

Though effectively ranking individuals, the non-dominated sort leaves a lot of individuals with the same rank, as shown in Figure 2.12. During the environmental selection, GAs often select some of the best individuals from the current population and the offspring to be part of the new population. Consider the non-dominated sort of Figure 2.12 as the ranking of a population and its offspring, how would you select the 3 best individuals ? They would obviously be selected within the 4 individuals of R_0, but how to choose between those ? A random selection is the

Figure 2.12: Assigning a rank to individuals in a 2 objectives minimization problem using the Pareto dominance.



easiest and simplest way to answer, but it is not the best way. When solving a Multi-Objectives Optimization Problem, it is expected that a diverse set of trade-offs between objectives is found; therefore, metrics have been designed to measure the diversity of trade-offs presented by a front, so as to rank individuals within a same front by their contribution to the diversity of solutions. Several measures exist, only two are presented here as examples: the crowding distance [Deb *et al.* 2000] and the hyper-volume [Auger *et al.* 2009].

Crowding distance Given a set of individuals R_i forming a front, the crowding distance is computed using Algorithm 1. f_m^{max} and f_m^{min} represent the maximum and minimum values for the m^{th} objective in R_i respectively; N_{obj} is the number of objectives in the problem. This measure gives an indication on how close in the objective space an individual is from its neighbors of the same front. Figure 2.13 shows some individuals from the same front together with their crowding distances. The detail of the computation is given for individual a. It is the sum over the objectives (loop on line 4 in Algorithm 1) of the distance between a's neighbors over the distance between the maximum and minimum value in the front for the current objective (operation on line 8 in Algorithm 1 repeated for each individual thanks to the loop on line 7). On the figure, a's neighbors are separated by a distance of 10 along the f_1 values axis while the best and worst individual for that fitness are separated by a distance of 20, which leads to $\frac{10}{20}$ for the first objective. To this is added the distance between its neighbors in the second objective, which is 5, over the distance between the best and worst individual on this front for the second objective, which is 19 and leads to $\frac{5}{19}$ for the second objective. Finally, the crowding distance for individual a is $\frac{10}{20} + \frac{5}{19}$. The crowding distance depends only on the distance

between the neighbors of the individual in the front in each objective. The closer the neighbors are to each others, the lesser the distance, and the more crowded the individual is. It can be observed when comparing the crowding distance values of the different individuals on the figure.

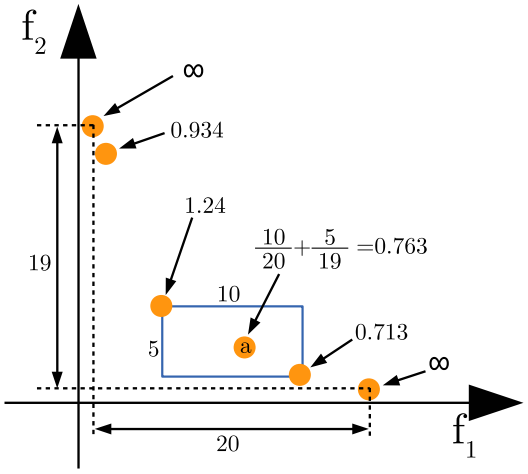
Individuals of the front are then ranked in decreasing order of their crowding distance, the best individual being the one with the highest crowding distance, and thus the less crowded one. Individuals representing the maximum or minimum value of at least one objective are assigned an infinite crowding distance. Indeed, those have no neighbor on one side, the neighbor is then considered to be at an infinite distance, which results in an overall infinite crowding distance value ($\frac{|\infty - constant|}{constant} + constant = \infty$). Therefore, they are the best individuals within a front. This is a good thing. Indeed, as solving a Multi-Objectives Optimization Problem consists in approximating the wider possible range of values from the optimal Pareto front, keeping extremal values is fundamental. If extrema are not kept, the range of known values for an objective decreases, effectively reducing the diversity of solutions found for that objective. Giving an infinite crowding distance to extrema sets them as the best individuals of their front and improves their chances of being selected.

Algorithm 1 Computation of the crowding distance for individual of a front R_i

1: $l = |R_i|$
2: **for** $ind = 0$ to $l - 1$ **do**
3: $\quad R_i[ind]_{distance} = 0$
4: **for** $m = 0$ to $N_{obj} - 1$ **do**
5: $\quad R_i = Sort(R_i, m)$ (Rank individuals in R_i in ascending order with respect to objective m)
6: $\quad R_i[0]_distance = R_i[l-1]_distance = \infty$
7: $\quad$ **for** $j = 1$ to $l - 2$ **do**
8: $\quad\quad R_i[j]_distance += \frac{R_i[j+1][m] - R_i[j-1][m]}{f_m^{max} - f_m^{min}}$

Even though performing quite well, the crowding distance does not always rank individuals as intuitively expected, an example is the second individual from the top in Figure 2.13. This individual has a crowding distance of 0.934, but it is really close to the first individual from the top, while individual a, which is further away from its closest neighbor has a crowding distance of only 0.763. So the later has a worse crowding distance than the former, which seems wrong in some way, it seems too close to its neighbor to bring that much diversity. This is because only the distance between the two neighbors is used in the computation, and not the distance of the individual with its neighbors. In addition, this measure takes into account diversity only, and not the performance of each individual. All individuals within the same rank are equivalent regarding the domination relation within the population, which is a set of individuals, but they may not be equivalent in the whole objective space. This is what has partially motivated the development of the hyper-volume measure presented below.

Figure 2.13: A set of individuals belonging to the same front and their crowding distance. This figure is in a two objectives minimization context. The crowding distance of each individual is reported next to it. The computation of the crowding distance of individual a is detailed.

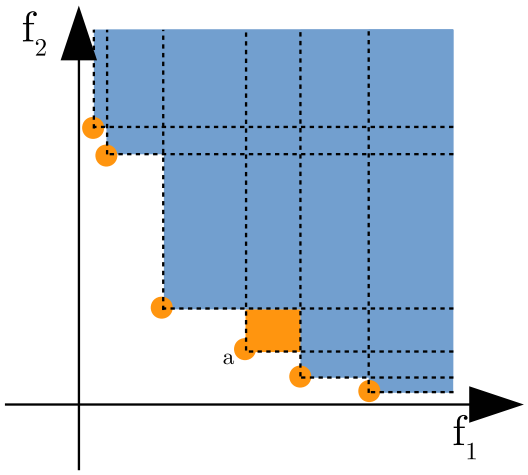


Hyper-volume As shown when explaining the Pareto dominance, each individual dominates a part of the objective space. A set of individuals together dominates a part of the objective space that is the union of the space dominated by each individual of the front. In Figure 2.14, the whole colored area (orange plus blue) is the area of the objective space dominated by the front composed of the orange points in a minimization context. The hyper-volume measure consists in measuring the contribution of an individual to the hyper-volume of space dominated by the front. For example, the hyper-volume of the individual a in Figure 2.14 is colored in orange. It is computed by making the difference between the hyper-volume dominated by the front (orange plus blue areas) and the hyper-volume dominated by the individuals of the front without a (blue area only). Although depicted for individual a only on the figure, this measure can be computed for each individual in the front. The higher the hyper-volume, the more an individual contributes to the domination of the objective space, so the most effective it is. Effective solutions are preferred for the next generations.

Individuals are then ranked in decreasing order of their hyper-volume value. If the reference point is taken at an infinite position, extremal points of the front are assigned infinite hyper-volume values, which is similar to what is done for extremal points in the crowding distance. This is beneficial for the evolution process as it has been shown in the previous paragraph.

Given those tools, several algorithms have been created to tackle Multi-Objectives Optimization Problems [Konak *et al.* 2006]. The focus is made here on Non-dominated Sorting Genetic Algorithm II (NSGA-II) [Deb *et al.* 2000], which is a well known and quite effective GA to perform the task. It retained our attention

Figure 2.14: Representation of the Hyper-volume measure for individual a. The Hyper-volume of a is colored in orange in this 2 objectives minimization problem. Other individuals Hyper-volume are not expressively represented here.



because of its simplicity and adaptiveness, most of its components can be tuned or changed without breaking the logic of the algorithm. In addition, it is quite stable and has made its proofs. It has also been the base algorithm for some methods dedicated to Multi-Modal Optimization Problem presented later in the next subsection, which is interesting for the fine grained classification problem.

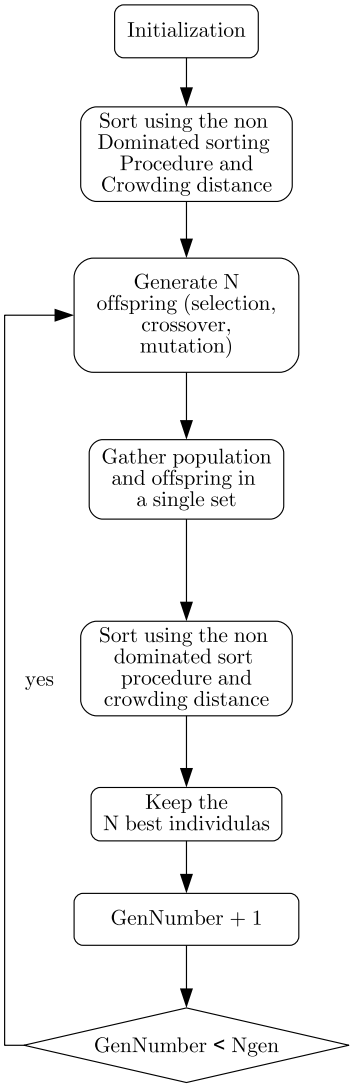
NSGA-II NSGA-II is a GA dedicated to Multi-Objectives Optimization Problems. Its workflow is presented in Figure 2.15. The initialization is straightforward, any kind of initialization can be used to create the first population of N individuals. As usual with EAs, without a priori information about the problem, a uniform random initialization is used.

The evaluation of individuals is done using the dominance relation and the crowding distance. To do so, individuals are ranked by front using the non dominated sort procedure, and individuals within the same front are ranked using their crowding distance. Positions in that ranking can be used as performance score by selection procedures.

Any mating selection, crossover, mutation can be used to generate the offspring. N offspring are generated during the variation process.

Then comes the most important part of this algorithm: the environmental selection. To perform this selection, the population and the offspring are put together in a $2N$ individuals set. They are then ranked using the non dominated sort procedure and the crowding distance. The N best individuals are kept to form the population for the next generation. Doing so, the best individuals are kept, ensuring convergence toward better solutions. And, by the use of the crowding distance, some diversity in the objective space is also ensured, thus better solving the Multi-

Figure 2.15: NSGA-II workflow



Objectives Optimization Problem by giving a diverse set of trade-offs between the objectives.

The environmental selection is the only step that can't be modified in NSGA-II's workflow, everything else can be tuned to fit any problem. As an example, the original algorithm was run on both continuously encoded and binary encoded problems, using two different kinds of crossovers and mutations. For real-coded problems, a simulated binary crossover and a polynomial mutation were used, while for binary-coded problems, one-point crossover and bitwise mutation were used. The mating selection process was a tournament selection in both cases. Crossover, mutations and selection schemes are explained in section 2.2.2. Any other kind of mating selection, mutation and crossover could have been used however. Changing

those operators will probably affect performances, but this flexibility is needed so that the algorithm can be adapted to fit any problem.

Other EAs are dedicated to Multi-Objectives Optimization Problems. Some may use external archives to keep track of the best identified solutions [Zitzler *et al.* 2001]. Others use niching techniques, i.e. identify and maintain multiple *niches*, multiple groups of individuals, that will potentially converge to different optimal solutions. Many algorithms exist to address Multi-Objectives Optimization Problems, some of which are reviewed by [Zitzler *et al.* 2000].

At the end of the run, the population of an algorithm dedicated to Multi-Objectives Optimization Problems is expected to characterize the optimal Pareto front, which represents the best trade-offs between the objectives. Because the population of the algorithm is fixed and contains a finite number of solutions, individuals are expected to spread along the optimal Pareto front to better characterize it. To do so, diversity measures are used during the evolution to enhance diversity in the objective space. Multi-Objectives Optimization Problems were presented because they are often used to build algorithms dedicated to Multi-Modal Optimization Problems. Multi-modal optimization is detailed in the next subsection.

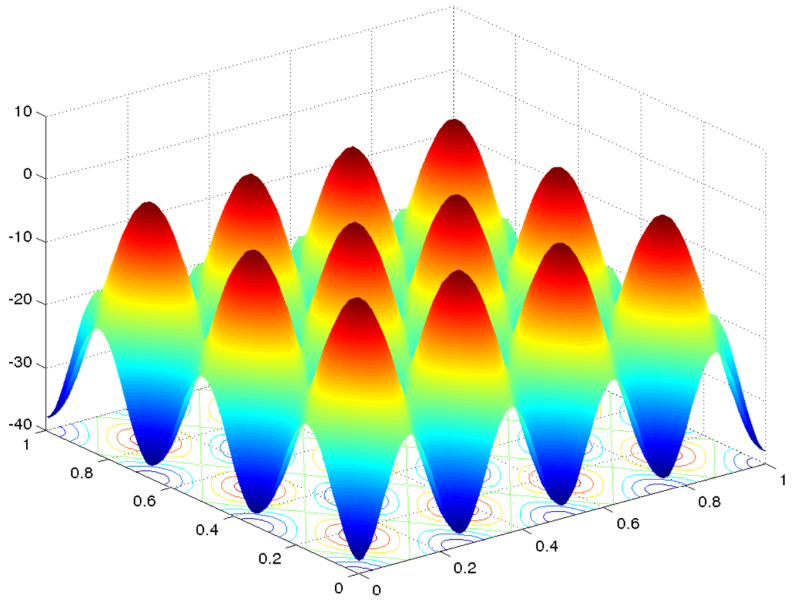
2.3.2.2 Evolutionary Algorithms for Multi-Modal Optimization Problems

A Multi-Modal Optimization Problem is a problem for which several satisfactory solutions exist, i.e. given a function f to optimize, several different parameter vectors lead to the optimal f value. For example, maximizing the function represented in Figure 2.16 is a multi-modal problem. Indeed, the maximal value of this function (dark red peaks) can be obtained with several different parameter vectors (different values of the parameters represented on the x and y axes). Solving a Multi-Modal Optimization Problem consists in finding as much satisfactory solutions as possible for the presented problem. Considering a function f from S to $\mathcal{R}$, maximizing f is a Multi-Modal Optimization Problem and solving it consists in finding the set of vectors V as defined in Equation 2.5. max_f is the maximum value reachable by f and ε is a precision parameter. A similar set can be defined for minimization problems.

$$V = \{\mathbf{v} \in S \text{ such that } max_f - \varepsilon < f(\mathbf{v})\} \tag{2.5}$$

The fine grained classification problem in CBIR can be considered as a Multi-Modal Optimization Problem. Indeed, images from the same category can be visually dissimilar, thus scattered in different part of the search space. All are solutions to the problem of maximizing the relevance to the category they belong to, which is then a Multi-Modal Optimization Problem. But outside the scope of CBIR, many real world problems tend to be multi-modal, and finding several solutions turned out to be useful in many situations. Has an example, having the choice between different solutions allows a decision maker to chose the solution that suits him the most, given some criteria that he couldn't include into the optimization process. When

Figure 2.16: A plot of the function modified rastringin from the CEC 2013 multi-modal problems benchmark



dealing with systems with evolving conditions (such as physic systems depending on temperature or pressure conditions), some solutions cannot be used all of the time. Having several solutions then allows to pick the best one at any time, without running the optimization process on every condition change. Those are just a few examples of the applications of Multi-Modal Optimization Problems, explaining the interest the community put into those problems.

Because EAs have been identified as good candidates to bring back exploration in the learning process dedicated to the fine grained classification in section 2.1, the effort put on those methods to address Multi-Modal Optimization Problems is particularly interesting. Several works have taken advantage of the diversity of solutions needed characterize the Pareto front in Multi-Objectives Optimization Problems to tackle Multi-Modal Optimization Problems using multi-objectivization.

So as to address the multi-modal nature of fine grained classification, three recent EAs dedicated to Multi-Modal Optimization Problems are presented and compared to see if they fit in the context. The first one is Parameterless Niching Assisted Non-dominated Sorting Genetic Algorithm-II (PNA-NSGA-II) [Bandaru & Deb 2013], a GA derived from the well known NSGA-II [Deb *et al.* 2000]. The two others are Multi-modal Optimization using Bi-objective Differential Evolution (MOBiDE) [Basak *et al.* 2013] and Multi-objective Optimization for locating Multiple optimal Solutions of Multi-modal Optimization Problems (MOMMOP) [Wang *et al.* 2014], and are both based on DE. In order to outline the improvement brought by the modifications done to the algorithms to ad-

Figure 2.17: NSGA-II workflow

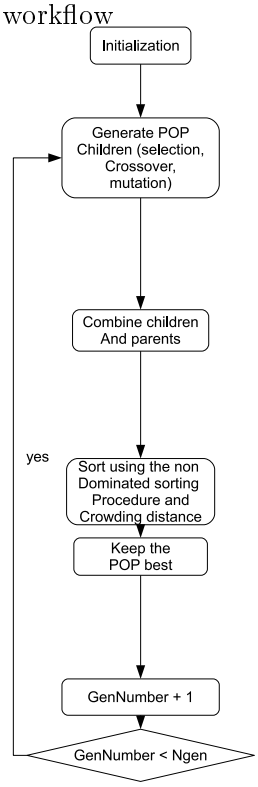


Figure 2.18: PNA-NSGA-II workflow

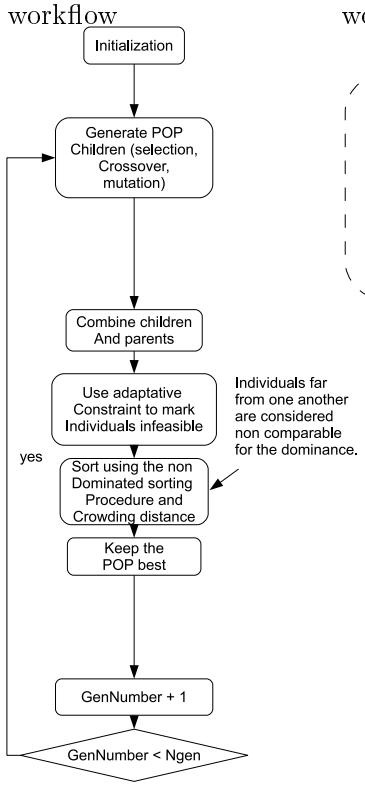


Figure 2.19: MOBiDE workflow

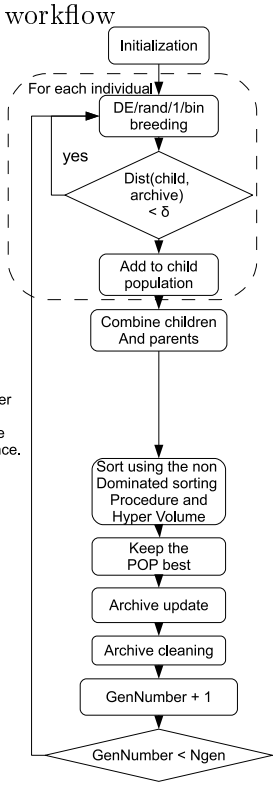
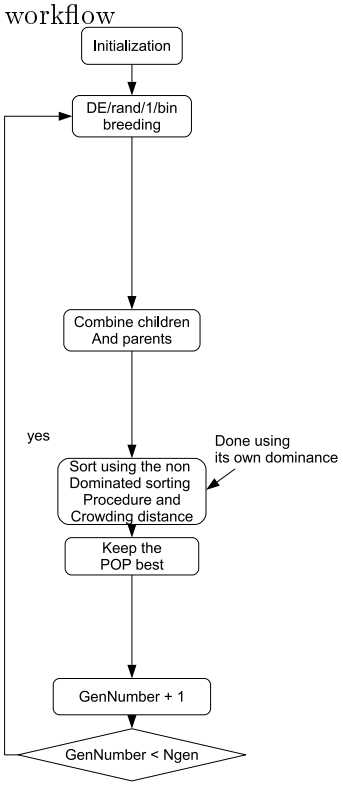


Figure 2.20: MOMMOP workflow



dress multi-modal problems, those three algorithms are compared to the standard NSGA-II [Deb *et al.* 2000], which is dedicated to Multi-Objectives Optimization Problems only as presented previously.

A side by side representation of those algorithms' workflow is presented in Figures 2.17 to 2.20. Those workflow are detailed in the following paragraphs. For the rest of this subsection, f is the multi-modal function under study, and N is the size of the population.

PNA-NSGA-II PNA-NSGA-II uses multi-objectivization to address the Multi-Modal Optimization Problem. Two objectives are set to form the Multi-Objectives Optimization Problem. The first and obvious objective function is f. As f is the function to be optimized, adding it as an objective ensures that parameter vectors giving its optimum value are retrieved.

The second objective is built to enhance diversity in the parameter space, so as to prevent the algorithm from converging to a unique area of the parameter space. This objective is presented in Equation 2.6. $\Omega(\mathbf{x})$ measures the cumulative distance of an individual to other individuals in the population. The smaller f_2 is, the further the individual is from other individuals in the population. With the minimization

of f_2 as an objective, individuals that are too close to each others have a bad score with respect to this objective.

$$f_2(\mathbf{x}) = \frac{1}{\sum_{j=1}^{N} ||\mathbf{x} - \mathbf{x_j}||^2} = \frac{1}{\Omega(\mathbf{x})} \tag{2.6}$$

A bi-objectives problem composed of the optimization of f and f_2 has now been defined. So the optimal parameter vectors are those that have a good f value, and are far from each other. Indeed, individuals close to each other have a high f_2 value, which is bad. So given the definition of the dominance relation, they are likely to dominated, unless they have a near optimal f value. As a consequence, they are likely to be discarded, unless they represent a good solution to the Multi-Modal Optimization Problem. And if an individual is far from others, it has a low f_2 value (which is good), and so it is willing to dominate other individuals unless its f value is really bad. As a consequence, it is likely to be kept for the next generation, effectively enhancing diversity because low f_2 value means far from other individuals. If an individual is both far from others and has a good f value, then it is a near perfect individual, and it dominates a huge part of the objective space. Thus it is discarded only if a lot of individuals with similar results are found, which is very unlikely and would mean more individuals presenting similar results than the population can contain have been found.

This is exactly what is expected to solve the multi-modal problem, individuals far from each other presenting good f values. Then solving this bi-objective problem is more likely to provide a good set of solutions to the Multi-Modal Optimization Problem.

To solve this bi-objectives problem, a dedicated multi-objective algorithm is used. PNA-NSGAII uses a modified version of NSGA-II that further enhance the diversity in the parameter space.

First of all, the dominance relation used in PNA-NSGA-II to compare individuals is a modified version of the dominance presented in section 2.3.2.1. In PNA-NSGA-II, if two individuals are too far from each others, they are considered not comparable, whatever their fitness values are. Then, the comparison of two individuals follows the modified version of the dominance relation presented in Definition 3.

Definition 3. PNA-NSGAII dominance

For $(\mathbf{a}, \mathbf{b}) \in \mathcal{D}$, $\mathbf{a}$ dominates $\mathbf{b}$ in a minimization context, noted $\mathbf{a} \prec_{mm} \mathbf{b}$, if and only if:
$proximate(\mathbf{a}, \mathbf{b})$ is true and
$\forall i \in [1, n]$, $f_i(\mathbf{a}) \leq f_i(\mathbf{b})$ and
$\exists i \in [1, n]$ / $f_i(\mathbf{a}) < f_i(\mathbf{b})$

The distance from which individuals are considered not comparable is determined by the proximate function presented in Equations 2.7, 2.8 and 2.9. m is the dimension of the problem, U_d and L_d are the upper and lower bounds of the search

space for dimension d and $\mathbf{x_d}$ stands for the d$^{\text{th}}$ component of vector $\mathbf{x}$.

$$T = \exp \frac{ln(N)}{m} \tag{2.7}$$

$$v_d = \frac{U_d - L_d}{T} \forall d \in 1, 2, ..., m \tag{2.8}$$

$$proximate(\mathbf{a}, \mathbf{b}) = 1 \text{ if } |(\mathbf{a} - \mathbf{b})_d| \leq v_d \tag{2.9}$$

This restriction makes individuals converge more locally than they would in the standard version of NSGA-II. Indeed, considering two individuals a and b such that $proximate(a, b) = 0$. a may dominates b, but because $proximate(a, b) = 0$ they are not comparable with respect to the new dominance relation and thus can fall into the same rank. Doing so, a and b have the same chance of being selected during the selection process. So this promotes local search, or exploitation. However this is not a problem since f_2 promotes exploration, so the balance between exploration and exploitation seems to be maintained.

The workflow of PNA-NSGA-II, which is presented in Figure 2.18, is detailed in the following. The initialization is done at random over the search space. Individuals are evaluated and then the mating selection and the variation process take place. The mating selection, crossover and mutation used in this algorithm are the same as in the original NSGA-II [Deb *et al.* 2000] for real-coded problems. However, every comparison between individuals is based on the new dominance relation.

Once N offspring have been generated, the current population and offspring are put together. Then, a new step is added when compared to the classic NSGA-II before the environmental selection can take place. Every individual that has not achieved a good enough f value is marked as infeasible. The ranking of individual is done without them, they are discarded. The threshold from which the f value is considered to be too small is adaptively determined and computed using Equations 2.10 to 2.12. Equation 2.10 is presented in a minimization context. ε is the expected precision for the retrieved optima, F^{best} is the best value of $f(\mathbf{x})$ found so far and $MaxGen$ is the maximum number of generations allowed for the evolution process.

$$\text{If } f(\mathbf{x}) > F^{best} + f_{gen} \times \varepsilon \text{ then } \mathbf{x} \text{ is infeasible.} \tag{2.10}$$

$$f_{gen} = a \times \exp^{b \times gen} \text{ with:} \tag{2.11}$$

$$a = \frac{10^{14}}{\exp^b}, b = \frac{\ln(2) - \ln(10^{14})}{MaxGen} \tag{2.12}$$

This adaptive constraint has been added to discard individuals that may have converged toward local optima due to the new dominance relation. At the beginning of the run, the exploration is more important and letting individuals converge towards both local and global optima is fine. It is possible because the adaptive constraint is really loose at the beginning of the run, almost all individuals respect the constraint and so are not discarded right away. However, as the runs progresses, individuals are expected to converge toward global optima only. Because of the new dominance

relation, individuals that have converged toward a local optima far from any other individuals then stay in the population. Thanks to the adaptive constraint, those are discarded as the end of the run approaches. Indeed, the constraints tightens when the end of the run gets close, and the f value needed to stay in the population becomes closer and closer to the best identified f value, making any local optima infeasible, so the individual corresponding to those are discarded. This results in a population converging toward global optima only.

Once the adaptive constraint has been applied, individuals are ranked as in the original NSGA-II algorithm using the non dominated sort procedure and the crowding distance. However, during the ranking process, the new dominance relation is used and individuals marked infeasible are put apart in an added last front grouping the worst individuals.

Finally, the N best individuals are kept to form the population for the next generation. Once the new population is built, another generation is run. The process ends when a given maximum number of generations has been made.

MOBiDE MOBiDE is based on DE, introduced in section 2.2.1. In order to address Multi-Modal Optimization Problem, MOBiDE uses multi-objectivization too. MOBiDE's first objective is the same as inPNA-NSGAII: f, the multi-modal function to be optimized. The second objective also focuses on the diversity of solutions. This objective is presented in Equation 2.13. This objective is almost the same as the second objective in PNA-NSGAII.

$$\begin{aligned} &\text{Max.} \left(f_3(\mathbf{x_d}) = \frac{\Omega(\mathbf{x})}{N} \right) \\ \iff &\text{Max.} \left(\frac{1}{N \times f_2(\mathbf{x})} \right) \end{aligned} \tag{2.13}$$

Indeed, maximizing the invert of f_2 multiplied by a constant or minimizing f_2 leads to the same result. The difference between PNA-NSGA-II and MOBiDE resides in the EA used to solve the Multi-Objectives Optimization Problem (DE for MOBiDE and a GA for PNA-NSGAII) and the adaptations made. MOBiDE's workflow is presented in Figure 2.19 and is detailed below.

The initialization process is done as usual: randomly unless a priori information about the problem are available.

MOBiDE's variation process is based on DE/rand/1/bin [Storn & Price 1997] to generate new individuals. The variation procedure of DE/rand/1/bin is presented in Algorithm 2. X_{i_1}, X_{i_2} and X_{i_3} are individuals and SBX is the simulated binary crossover procedure (presented in section 2.2.2). Each individual of the population is used to generate one offspring. The variation process is based on the application of differences of vectors for the mutation and a simulated binary crossover. MOBiDE adds a step before the addition of the generated individual to the set of offspring to ensure several different solutions are found. This algorithm uses an archive to keep track of the best individuals found through the evolution. Each time an individual is generated, if it is within a δ radius of any individual in the archive, it is immediately

Algorithm 2 DE/rand/1/bin Breeding procedure

1: **for** $i \in [1, N]$ **do**
2: chose i_1, i_2, i_3 randomly such that $i \neq i_1 \neq i_2 \neq i_3$
3: $V = X_{i_1} + F \times (X_{i_2} - X_{i_3})$
4: $U = SBX(V, X_i)$
5: add U to the offspring set.

discarded, and another individual is generated instead. This enforces the exploration of new areas of the search space. The influence of δ on the performances of MOBiDE is tested in the experiments presented after the description of the algorithms.

Once offspring have been generated, they are put together with individuals of the population to form a single set. Individuals in this set are ranked using the non dominated sort procedure. The hyper-volume criteria is used to rank individuals within the same front. The non-dominated sort procedure and the hyper-volume are presented in section 2.3.2.1.

Then comes the environmental selection which keeps only the N best individuals to form the new population.

Finally, before looping to a new generation, specific steps to MOBiDE are run: the archive update and archive cleaning processes. The archive is updated using the new population. Individuals that present a good enough f value (see Equation 2.14) are added to the archive. The required f value to be added to the archive is computed from the best f value found since the beginning of the run (F^{best}) and a fixed parameter α. α has been set to 0.1 in the experiments, as it is the case in the work presented in [Basak *et al.* 2013].

Add $\mathbf{x}$ to the archive if

$$\begin{cases} f(\mathbf{x}) < (1+\alpha) * F^{best} \text{ when } F^{best} > 0 \\ f(\mathbf{x}) < (1-\alpha) * F^{best} \text{ when } F^{best} < 0 \\ f(\mathbf{x}) < 0.001 \text{ when } F^{best} \approx 0 \end{cases} \tag{2.14}$$

The last step is to clean the archive by removing from it any individual that has a fitness below the average fitness of the individuals in the archive.

A new generation can now begin with the new population and the updated archive. The process stops after a given number of generations.

MOMMOP MOMMOP also uses multi-objectivization to solve Multi-Modal Optimization Problems. However, it is not using f as an objective as PNA-NSGAII and MOBiDE do. MOMMOP builds a brand new Multi-Objectives Optimization Problem, using f as part of the different objectives. In fact, a family of bi-objectives problems is defined by MOMMOP, this family is presented in Equation 2.15. x_i stands for the i$^{\text{th}}$ component of the vector $\mathbf{x}$, U_i and L_i for the upper and lower bound of the search space for i$^{\text{th}}$ dimension, F^{best} and F^{worst} for the best and worst known f values and D for the dimension of the parameter space. Whether f is to be

minimized or maximized, the objectives in BOP_i are to be minimized to optimize f, indeed, $\frac{|f(\mathbf{x})-F^{best}|}{|F^{worst}-F^{best}|}$ gets closer to 0 as $f(\mathbf{x})$ gets close to F^{best}, and the value of this term grows as $f(x)$ get further from F^{best}. Then $\frac{|f(\mathbf{x})-F^{best}|}{|F^{worst}-F^{best}|}$ is to be minimized to optimize f, in consequence, f_1 and f_2 are also to be minimized to optimize f.

In each bi-objectives problem, η is a parameter that grows with the number of function evaluations (FEs), i.e. the number of times an f value is computed. This gives more importance to the optimization of the Multi-Modal Optimization Problem f as the run progresses. Indeed, if η is equal to 0, the f value has no importance on an individual score. The biggest η becomes, the more important the f value term becomes in the equation, and thus the more the f value is prioritized.

$$BOP_i \begin{cases} \text{Min.} \left(x_i + \frac{|f(\mathbf{x})-F^{best}|}{|F^{worst}-F^{best}|} \times (U_i - L_i) \times \eta \right) \\ \text{Min.} \left(1 - x_i + \frac{|f(\mathbf{x})-F^{best}|}{|F^{worst}-F^{best}|} \times (U_i - L_i) \times \eta \right) \end{cases} \tag{2.15}$$

Authors of MOMMOP proved that the two objectives of this Multi-Objectives Optimization Problem are conflicting. Indeed, the first objective scales with x_i while the second scales with $1 - x_i$. And for both objectives, the second term is the same and is a normalized evaluation of f. So these two objectives are clearly conflicting, x_i and $1 - x_i$ can't be optimized at the same time, one increases while the other decreases. This ensures that the optimal Pareto front contains several different solutions. Authors of MOMMOP also proved that all Pareto optimal solutions to this Multi-Objectives Optimization Problem are solutions to the Multi-Modal Optimization Problem f. Without going into the details about that, this means that the set of solutions to BOP_i is a subset of the solutions to f. So solving BOP_i gives part of the solution to the Multi-Modal Optimization Problem.

Each Multi-Objectives Optimization Problem of the family is associated with one of the decision variable. Choosing only one particular variable and solving the Multi-Objectives Optimization Problem associated with it could result in missing some of the solutions to the Multi-Modal Optimization Problem. Indeed, several solutions to f may have the same value for x_i, it is the case for the function presented at the beginning of this chapter in Figure 2.16. Though corresponding to different parameter vectors, because they have the same value for x_i, those individuals have the same value for both objectives of BOP_i. Because of their equality in the objective space, those solutions have a really bad crowding distance, and thus are likely to be discarded during the environmental selection. To avoid this, MOMMOP is using simultaneously the bi-objective problems linked to each decision variable. To do so, an extension to the dominance relation has been defined and replaces the standard dominance in this algorithm. Equation 2.16 shows this new dominance relation.

$$\begin{aligned} &\mathbf{x} \prec \mathbf{y} \text{ if :} \\ &\forall i \in [1, D], \mathbf{x} \prec \mathbf{y} \text{ on } BOP_i \end{aligned} \tag{2.16}$$

Another condition, showed in Equation 2.17, has also been added to the dominance. The normalization is explained in Equation 2.18 and the distance used is the euclidean distance. This modification helps discarding individuals that are too close

$$\begin{aligned} &\mathbf{x} \prec \mathbf{y} \text{ if :} \\ &f(\mathbf{x}) \text{ is better than } f(\mathbf{y}) \wedge \\ &distance(norm.(\mathbf{x}), norm.(\mathbf{y})) < 0.01 \end{aligned} \tag{2.17}$$

$$\begin{aligned} &norm.(\mathbf{x}) = \mathbf{n} \text{ so that} \\ &\forall i \in [1, D], n_i = \frac{x_i - L_i}{U_i - L_i} \end{aligned} \tag{2.18}$$

to one another by making one dominate the other. Indeed, if two individuals are close to each other, the one with the greatest f value dominates the other. The dominated individual is assigned a lower rank during the non dominated sorting, and thus is most likely to be discarded during the environmental selection, while the other is more likely to be kept for the next generation.

The workflow of the algorithm used to solve the defined Multi-Objectives Optimization Problem using this new dominance is presented in Figure 2.20. It is the same workflow as NSGA-II, but it uses DE/rand/1/bin's variation process, as presented for MOBiDE, and its own dominance relation. So everything happens in the dominance relation and the definition of the Multi-Objectives Optimization Problem.

NSGA-II has already been presented in 2.3.2.1 so this ends the presentation of the algorithms. NSGA-II is here used as a baseline to see the improvements brought by the multi-modal adaptation made in other algorithms. It is used to solve the same Multi-Objectives Optimization Problem as defined in PNA-NSGA-II. The Multi-Objectives Optimization Problems defined by each algorithm are recalled in Table 2.1.

Table 2.1: objective functions used by the different algorithms

	NSGA-II	PNA-NSGA-II	MOBiDE	MOMMOP
Objective 1	Min. or Max.$(f(x))$			Min.$\left(x_i + \frac{\lvert f(\mathbf{x}) - F^{best} \rvert}{\lvert F^{worst} - F^{best} \rvert} \times (U_i - L_i) \times \eta\right)$
Objective 2	Min.$\left(f_2(\mathbf{x}) = \frac{1}{\sum_{j=1}^{N} \lvert\lvert \mathbf{x} - \mathbf{x_j} \rvert\rvert^2} = \frac{1}{\Omega(\mathbf{x})}\right)$		Max.$\left(f_3(\mathbf{x_d}) = \frac{\Omega(\mathbf{x})}{N} = \frac{1}{N \times f_2(\mathbf{x})}\right)$	Min.$\left(1 - x_i + \frac{\lvert f(\mathbf{x}) - F^{best} \rvert}{\lvert F^{worst} - F^{best} \rvert} \times (U_i - L_i) \times \eta\right)$

The comparison setup and the results are presented next, before concluding on the relevance of those algorithms in the context of fine grained classification for image retrieval.

Performance Analysis of Those Algorithms To compare those algorithms, a set of benchmark functions must be chosen. A large choice of functions is available [Jamil & Yang 2013]. To have a good insight of each algorithm strengths and weaknesses, it is essential to use functions that present different types of problems. A set of benchmark functions respecting this condition has been published for the CEC2013 multi-modal optimization competition [Li *et al.* 2013]. Along with the benchmark functions, this technical report also provides the maximum number of

Table 2.2: Benchmark parameters

Benchmark functions	CEC2013 [Li *et al.* 2013]
Number of runs	50
Population size	$100 * D$
Precision (ε)	0.001

function evaluations allowed for each function, some interesting figures to present as results when dealing with Multi-Modal Optimization Problems and some experimental recommendations. It is the chosen test bed for the studied algorithms.

As recommended in [Li *et al.* 2013], each experiment is run 50 times and average results are presented. The average results of the same experiment done with several algorithms are compared. To ensure that the differences observed between those means are relevant, a statistical test is required. The Kruskal-Wallis [Coolican 2004] test has been used to validate the results obtained. This test is the equivalent to the ANOVA test when one does not know if the results of the experiments follow a normal law. In addition, the Kruskal-Wallis test presents similar results to the ANOVA test when dealing with data following a normal law and having enough samples [Coolican 2004]. The number of samples studied in the presented experiments is high enough to assume this property is verified.

The peak ratio measure has been chosen as the tool for comparisons. This measure, presented in [Li *et al.* 2013], is the ratio of the number of optima retrieved over the total number of optima of the function. As solving a Multi-Modal Optimization Problem consists in retrieving as much optimal solution as possible, this measure seems adapted to compare algorithms performances. In all the conducted experiments, the expected precision for the retrieved optima is $\varepsilon = 0.001$, i.e. x is considered to be a solution if and only if $|f(x) - f_{optimal}| \leq 0.001$ where $f_{optimal}$ is the optimal expected value for f.

All the compared algorithms are population based. To be fair in our comparisons, the same population size is used in each algorithm. This size has been set to $100 * D$. This value is what has been used by the authors of [Bandaru & Deb 2013].

Those parameters are summed up in table 2.2.

In addition to those parameters common to all algorithms, some other parameters are specific to each algorithm. All the parameters are not detailed here, but this study has been published as a conference paper [Pighetti *et al.* 2015a] in which more details are available.

Thanks to our own implementations of PNA-NSGA-II and MOBiDE, the matlab sources of MOMMOP provided by the authors and the implementation of NSGA-II provided in ECJ[1], the results of each algorithm at each generation of the runs are accessible. So, in addition to the final results, the results at each generation can also be compared, and thus the convergence speed and convergence scheme differences between the algorithms can be revealed too. To ensure that those comparisons are

[1] `https://cs.gmu.edu/~eclab/projects/ecj/` visited on July the 27th, 2016

Table 2.3: Kruskal-Wallis results for algorithms comparison

Function name	Generations for which the Kruskal-Wallis test fails
Equal Maxima	0 to 1
Modified Rastringin All	0 to 2
CF3 3D	0 to 17
CF3 10D	0 to 31

relevant, the Kruskal-Wallis test is performed for each generation.

Tables 2.4 to 2.7 compare the evolution of the peak ratio obtained by MO-BiDE, PNA-NSGA-II, MOMMOP and NSGA-II for 4 functions from the CEC2013 benchmark [Bandaru & Deb 2013]. The results of the Kruskal-Wallis tests for this experiment are reported in Table 2.3. It lists, for each function, the generation numbers for which no conclusion can be drawn from the presented data. That is the generations for which the confidence in the results was less than 95%.

Table 2.3 shows that the Kruskal-Wallis test often fails on the first few generations. But comparisons are relevant for all presented functions from the 31^{st} generation, as the the Kruskal-Wallis test presents at least a 95% confidence that the difference observed are relevant from this generation.

In Tables 2.4 to 2.7, the original NSGA-II algorithm has a fairly fast convergence, but its results are poor in term of peak ratio. Indeed, when looking at the results in detail, it appears that NSGA-II is always retrieving exactly one optimum per run whereas other algorithms are retrieving more solutions for each function. This shows that the niching techniques and other mechanisms used in the dedicated multi-modal algorithms improve performances when solving Multi-Modal Optimization Problems. However, if only one solution is required, the choice of the raw NSGA-II seems to be the best. It converges faster or at the same speed than other presented algorithms and always finds one optimum to the function in the benchmark used.

The following concentrates on the comparison of the algorithms dedicated to Multi-Modal Optimization Problems. It can be observed that MOMMOP's convergence is not stable. It converges extremely fast for five uneven peak trap (Table 2.4) and Composition Function 3 (CF3) in 3 dimensions (Table 2.6), whereas its convergence is slow for modified rastringin (Table 2.5) and CF3 in 10 dimensions (Table 2.7). MOBiDE and PNA-NSGAII always present the same convergence scheme. MOBiDE tends to find a lot of solutions at the beginning of the run and then slowly improves. On the contrary, PNA-NSGAII has poor results at the beginning of the run but presents a big step in the retrieved results at around 75% of the run completion. This convergence scheme is probably due to the adaptive constraint that evolves exponentially as the run progresses. It confirms what has been said about this constraint when presenting MOMMOP: tightening it forces individuals to converge toward global optima as the run progresses.

Speaking of the peak ratio value at the end of the run, MOMMOP is ahead for five uneven peak trap, modified rastringin and CF3 in 10 dimensions. However,

Table 2.4: Evolution of the peak ratio of the different algorithms for the function five uneven peak trap

Generation	0	10	25	60	175	350	425	500
PNA-NSGA-II	0	0	0	0.11	0.66	**1**	**1**	0.98
MOBiDE	0	0	0	0.26	**1**	**1**	**1**	**1**
NSGA-II	0	0.05	0.5	0.5	0.5	0.5	0.5	0.5
MOMMOP	**0.97**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

Table 2.5: Evolution of the peak ratio of the different algorithms for the function modified rastringin

Generation	0	20	50	120	350	700	850	1000
PNA-NSGA-II	**0**	0.01	0.05	0.09	0.09	0.08	0.24	0.98
MOBiDE	**0**	0.05	**0.53**	**0.61**	0.73	0.8	0.82	0.83
NSGA-II	**0**	**0.12**	0.08	0.08	0.08	0.08	0.08	0.08
MOMMOP	**0**	0	0	0	**1**	**1**	**1**	**1**

Table 2.6: Evolution of the peak ratio of the different algorithms for the function CF3 in 3 dimension

Generation	0	26	66	159	466	933	1133	1333
PNA-NSGA-II	**0**	0	0	0	0.05	0.15	**0.28**	**0.43**
MOBiDE	**0**	0	0	0.01	0.16	**0.17**	0.18	0.19
NSGA-II	**0**	**0.09**	**0.17**	**0.17**	**0.17**	**0.17**	0.17	0.17
MOMMOP	**0**	**0.09**	**0.17**	**0.17**	**0.17**	**0.17**	0.17	0.17

Table 2.7: Evolution of the peak ratio of the different algorithms for the function CF3 in 10 dimension

Generation	0	8	20	48	140	280	340	400
PNA-NSGA-II	**0**	**0**	**0**	**0**	0	0	0.01	0.1
MOBiDE	**0**	**0**	**0**	**0**	0	0	0	0
NSGA-II	**0**	**0**	**0**	**0**	**0.17**	**0.17**	**0.17**	0.17
MOMMOP	**0**	**0**	**0**	**0**	0	0.05	0.06	**0.18**

Table 2.8: MOBiDE influence of δ, Kruskal-Wallis results

Function name	Generations for which the Kruskal-Wallis test fails
Vincent 3D	0 to 201
Modified Rastringin all	0 to 24
Five Uneven Peak Trap	0 to 22

MOBiDE and PNA-NSGAII are not far behind. PNA-NSGAII gets the best results when dealing with CF3 in 3 dimensions, retrieving more than twice as much optima as MOMMOP or MOBiDE. But this is the only case where this algorithm takes the lead. In other cases, it has lower or equal results than MOMMOP or MOBiDE and has a slower convergence speed. Therefore, MOMMOP and MOBiDE will most of the time be preferred.

When dealing with higher dimensional problems, like CF3 in 10 dimensions, MOMMOP stays stable in its results while MOBiDE completely falls off. So, in the end, MOMMOP seems to be a good choice when dealing with Multi-Modal Optimization Problems today. It converges sometimes really fast and sometimes it is fairly slow, but never slower than PNA-NSGAII. Its performances are better than or equal to those of MOBiDE and PNA-NSGAII and it is able to perform at least equally with NSGA-II when dealing with composition functions. However, MOBiDE's results are not too bad and its stable and quick convergence can be preferred in some cases, in particular when the number of function evaluations has to be reduced. However MOBiDE depends on a parameter δ that needs to be setup manually and for which not many information is provided. For the results presented above, this parameter has been set to $0.9e^{-6}$ for every function.

In order to better determine the influence of this parameter on MOBiDE's performances, a second experiment has been done. MOBiDE has been used with different values of δ to solve the benchmark functions. Tables 2.9 through 2.11 present the evolution of the peak ratio of MOBiDE with different δ values for 3 functions from the benchmark. As for the previous experiment, the Kruskal-Wallis tests results are reported in Table 2.8. The results of only 3 functions are shown here in Tables 2.9 to 2.11. Other functions didn't bring more informations.

The Kruskal-Wallis tests in Table 2.8 show that no trusted comparison can be observed on the first generations, up to the 201^{st} generation for Vincent in 3 dimensions. However most of the presented data can be used for comparisons safely as the Kruskal-Wallis test ensures a 95% confidence on the relevance of the differences observed after the 201^{st} generation.

Depending on the benchmark function, the best results are not obtained with the same value of δ. For Vincent3D, the best value of δ is by far $0.9e^{-3}$, whereas for FiveUnevenPeakTrap the best value of δ is by far $0.9e^{-5}$ or $0.9e^{-6}$. And for modified rastringin, the value of δ is of less importance. In conclusion, δ is a sensitive and crucial parameter of MOBiDE which must be chosen carefully depending on the problem. We tried to infer some heuristic to set this parameter to its optimal value depending on the problem but were not able to come up with a satisfying solution

Table 2.9: Evolution of the peak ratio with different values of delta for the function five uneven peak trap

Generation	0	10	25	60	175	350	425	500
MOBiDE $0.9e^{-3}$	**0**	**0**	0.01	0.01	0.04	0.05	0.05	0.05
MOBiDE $0.9e^{-4}$	**0**	**0**	0.06	0.23	0.27	0.28	0.28	0.28
MOBiDE $0.9e^{-5}$	**0**	**0**	0.16	**1**	**1**	**1**	**1**	**1**
MOBiDE $0.9e^{-6}$	**0**	**0**	**0.26**	**1**	**1**	**1**	**1**	**1**

Table 2.10: Evolution of the peak ratio with different values of delta for the function modified rastringin

Generation	0	20	50	120	350	700	850	1000
MOBiDE $0.9e^{-3}$	**0**	0.03	0.33	0.63	0.84	0.88	0.89	0.9
MOBiDE $0.9e^{-4}$	**0**	**0.07**	**0.55**	**0.72**	**0.88**	**0.95**	**0.97**	**0.98**
MOBiDE $0.9e^{-5}$	**0**	0.04	0.54	0.62	0.78	0.85	0.87	0.87
MOBiDE $0.9e^{-6}$	**0**	0.05	0.53	0.61	0.73	0.80	0.82	0.83

Table 2.11: Evolution of the peak ratio with different values of delta for the function Vincent 3D

Generation	0	26	66	159	466	933	1133	1333
MOBiDE $0.9e^{-3}$	**0**	0.01	0.19	0.30	0.42	0.55	0.60	0.63
MOBiDE $0.9e^{-4}$	**0**	0.02	0.20	0.29	0.33	0.38	0.39	0.40
MOBiDE $0.9e^{-5}$	**0**	0.02	0.19	0.29	0.32	0.34	0.35	0.36
MOBiDE $0.9e^{-6}$	**0**	**0.05**	**0.53**	**0.61**	**0.73**	**0.80**	**0.82**	**0.83**

yet. This makes MOBiDE a difficult method to tune, and its convergence speed strength is greatly balanced by the drawback of tuning δ for each specific problem.

Practical usage in the fine grained classification The results of those recent algorithms greatly differ depending on the function studied. Overall, the results are quite good when dealing with *simple* functions. However, when the dimensionality increases and the function are composed, results are quite bad. In the fine grained classification problem, the problem landscape is unknown, and thus the lack of reliability of those algorithms is a big drawback. In addition, image descriptions are often high dimension vectors, having over 1000 dimensions. In such a context, fairly bad results can be expected when using those algorithms given the drop observed in performances when getting over 10 dimension in the presented experiments.

Therefore, those methods seem unadapted to the CBIR context, or at least the fine grained classification challenge. Works using other EAs to tackle some CBIR tasks have been presented at the beginning of this section, but those are not adapted to the fine grained classification challenge. However, some methods combining EAs and other algorithms have been developed recently, addressing the CBIR problem. Those are called hybrid systems, and building such a system would allow to benefit from the performances of the existing techniques while introducing exploration using EAs. The next section is dedicated to the presentation of some hybrid systems dedicated to CBIR.

2.4 Hybrid Systems

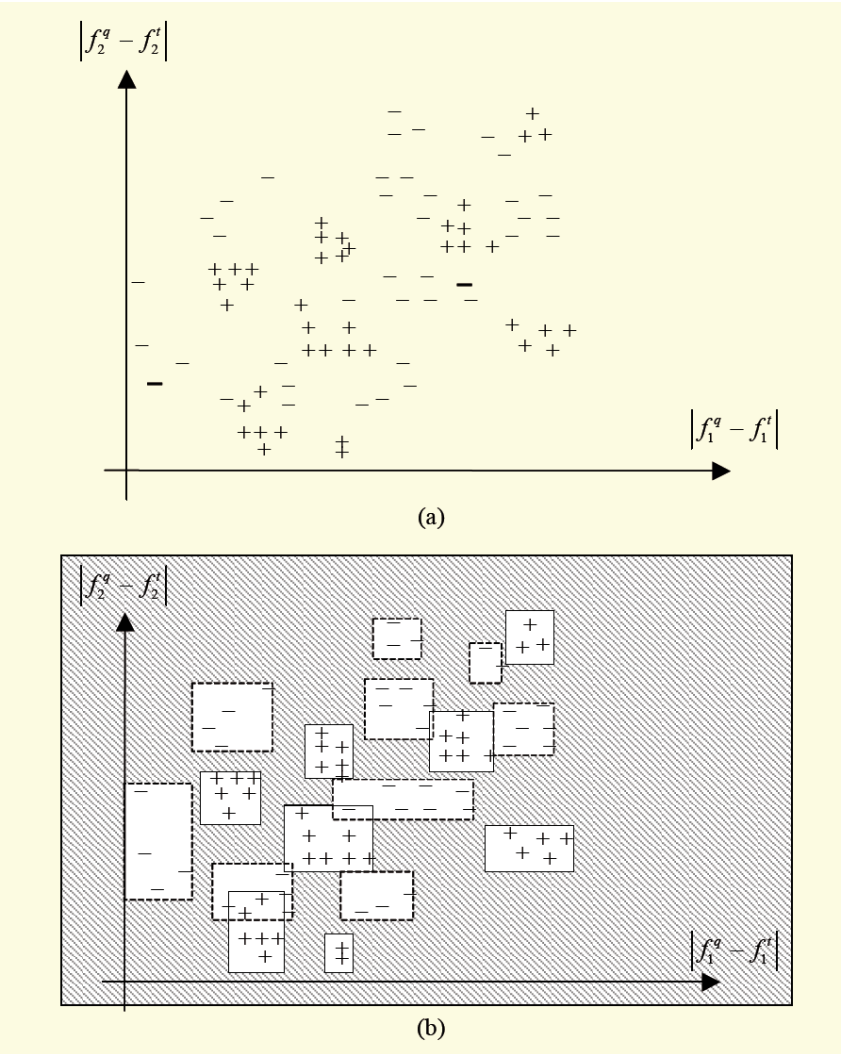
Hybrid systems are composed of two or more algorithms, trying to take advantage of every algorithm involved to achieve better performances. Several schemes exist to build such systems: algorithms may run in parallel, results from one algorithm could be used as the input or relevance for the other etc. The way algorithms are put together and communicate to form the hybrid system is as important as the algorithms forming the system themselves. Such hybrid systems have been used recently in CBIR, in particular to build interactive retrieval systems. Some of those systems are using EAs to perform exploration and global search, and then add a local search component, most of the time to enhance the convergence speed. Algorithms presented here are separated into two groups: those that are using some kind of nearest neighbor technique as the local search, and those using SVMs. This separation was made to stress out the use of SVMs in some hybrid techniques because they are one of the most popular and effective class of algorithms to address CBIR today, as discussed in section 1.2.4.

EAs and Nearest Neighbor Search [Lakdashti & Ajorloo 2011] enhances an existing CBIR technique using an IGA and k-means clustering to build a rule-based ranking system based on user's feedbacks. The user comes up with a request image,

and the underlying CBIR system is used to retrieve the best results and show them to the user for relevance feedback.

Feedbacks are separated into two categories: relevant and irrelevant images. An adaptive dedicated k-means clustering is run on each set (relevant and irrelevant set). For each cluster found, an hypercube in the description space is constructed based on the images contained in each cluster. This is represented in a 2 dimensional space on figure 2.21. Each hypercube is associated to the relevant or irrelevant class based on the set it was built on. The dedicated k-means clustering algorithm described in the paper aims at minimizing the overlapping volume between hypercubes of opposite classes.

Figure 2.21: relevant and irrelevant sets in a 2 dimensional space before (a) and after (b) clustering and hypercube computation. Image extracted from [Lakdashti & Ajorloo 2011].



Each hypercube represents a rule for the rule-based decision. For any image in the dataset, if it falls into at least one hypercube, the number of relevant hypercubes h_R and irrelevant hypercubes h_I it falls into is counted. The score of the image is then $\frac{h_R}{h_R+h_I}$. If it is not in an hypercube, its relevance is determined by the underlying CBIR system.

Hypercubes are also encoded as individuals for an IGA (the encoding procedure is based on the coordinates of the corners), and will be evolved to boost the performances of the rule-based decision. They are separated into two completely separate population that will evolve independently: a relevant population and an irrelevant population. To evolve the hypercubes, an evaluation of their quality is needed; this evaluation is computed from the user's feedbacks. Each image annotated as relevant by the user increases the score of hypercubes from the relevant population it falls into by ϕ_R and decreases the score of each hypercube from the irrelevant population by σ_I. Each image annotated as irrelevant by the user increases the score of hypercubes from the irrelevant population it falls into by ϕ_I and decreases the score of each hypercube from the relevant population by σ_R. Fitnesses are lower bounded by 0. If an individual reaches a fitness of 0, the associated rule is ignored during the ranking process.

Once every individual is evaluated, selection, crossover and mutation are performed. After this evolution process, individuals are evaluated again and the database is ranked using the new individuals before presenting some images retrieved as relevant and irrelevant to the user for annotation as well as the ranking of the database as the result of the search. After the user has annotated the images, the evolution process starts over from the evaluation, taking into account the newly annotated images.

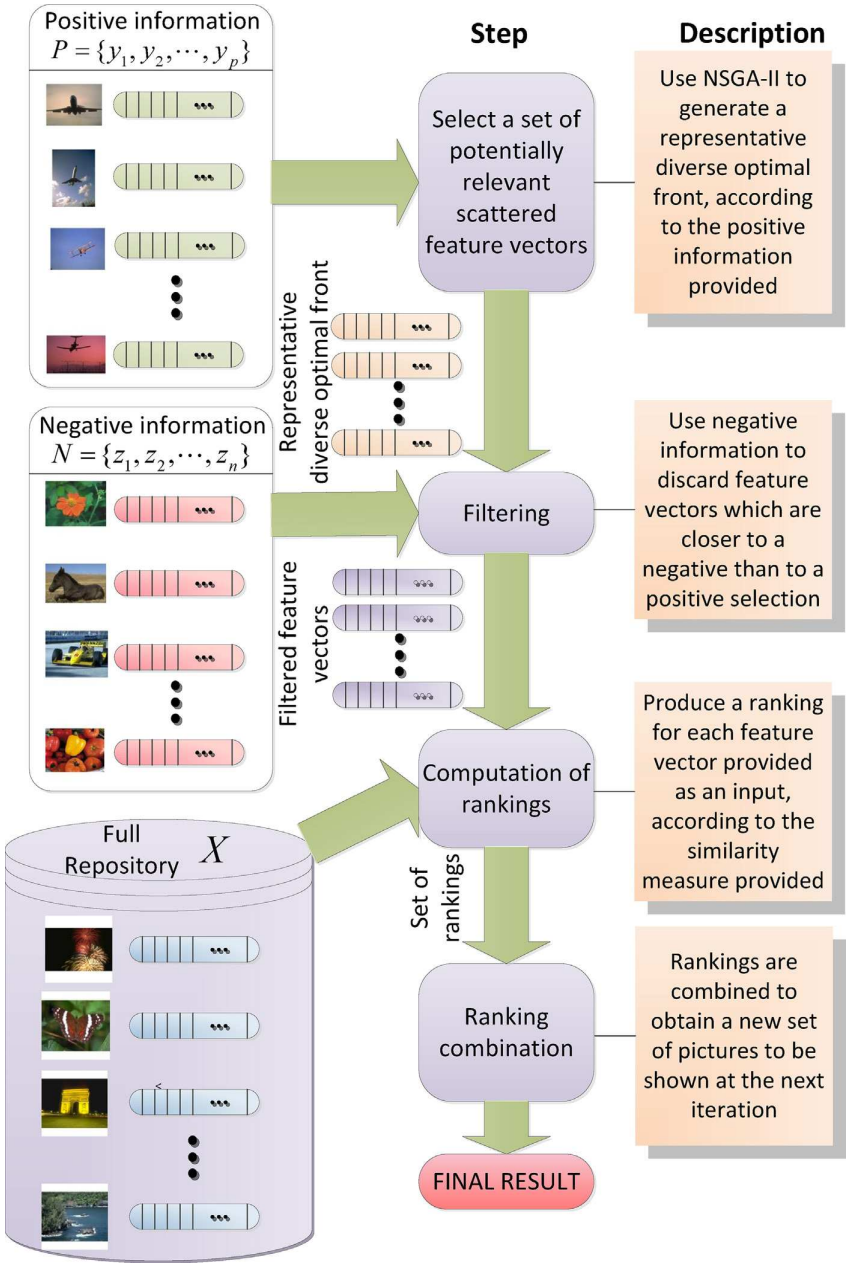
After a given number of user's evaluations, the population of the GA is reset by computing the dedicated k-means on the set of annotated individuals. This set has grown since the last computation of the k-means, therefore, the newly computed clustering is expected to better fit user's expectation. An entirely new populations for the relevant and irrelevant populations are generated from this new clustering. As the k-means performs a local search around each element of the set, this reset is expected to improve the convergence speed. Indeed, the new initialization may be closer to the expected result, and thus the GA may converge faster from the newly generated individuals. The process ends once the user is satisfied or decides to stop.

The results show that this scheme outperformed raw SVMs and others CBIR methods on a database composed of 10,004 images in 28 classes from the IRMA medical images database. The underlying CBIR algorithm used in the experiments is not specified, and though the system could be adapted to any CBIR system, the choice may have a big impact on the performances of the system. However, experiments conducted by the authors showed that this system evolves better and outperforms other methods to which it is compared.

A multi-objectives GA is associated to several local search methods to create an interactive CBIR system by [Arevalillo-Herráez *et al.* 2013]. Their idea is to explore the search space while still being close to the known relevant images and far from

the known irrelevant images. The workflow of the system they propose is presented in figure 2.22. First, given a query image, database images are ranked by similarity to the query and first results are shown to the user for feedbacks.

Figure 2.22: Workflow of the system proposed by [Arevalillo-Herráez *et al.* 2013]; extracted from the publication.



Once several feedbacks are available, they are separated as relevant ones R and irrelevant ones I. To explore the search space, the authors used a multi-objective

GA, NSGA-II. Individuals of the GA are vectors from the image representation space, even though they may not correspond to any image in the database. For each image R_i in R, the similarity of an individual with R_i is an objective of the GA, so there are as many objectives as images in R. Multi-objective GAs are usually built to handle less than 5 or 6 objectives, beyond this, problems are called many objectives [Li *et al.* 2015] and classical multi-objective algorithms are most of the time inefficient to solve them. Therefore, using each image positively annotated could be a big pitfall in this method.

The population is initialized randomly and evolves for a given number of generations (the number of generations is set to keep the evolutionary process time below 1 second by the authors). This generates a set of vectors distributed over the objective space and approximating the optimal Pareto front. Those vectors are filtered, discarding any vector closer to an element from I than an element from R. The remaining vectors are used as seed for a local search method (nearest neighbors, self organizing map, etc.), making as many ranking of the database as remaining vectors. Those ranking are visited iteratively to build the final ranking (the first image of each ranking, then the second image of each ranking, ...). The user can then provide new feedbacks and a new iteration starts from the initialization of the GA.

This hybrid system was applied with success to databases ranging from 1500 images to 30,000 images, outperforming the use of local search methods alone. However, the number of objectives of the GA rapidly grows with new positive annotations added at each generation. This makes the problem shift from the multi-objectives domain to the many objectives domain [Ishibuchi *et al.* 2008], which could be an issue for NSGA-II even though here the results were satisfying.

Those methods show that using exploration techniques in addition to local search brings some improvement when compared to the usage of only local search in CBIR. However, the local search techniques used are not the best known for CBIR. SVMs still have a lead in the domain, and the next paragraph presents some hybrid systems using them together with other algorithms.

EAs and SVMs [Wang *et al.* 2005] combines SVM and IGA to perform an emotion semantics image retrieval system. The process starts with 12 pseudo random images representing different emotions. Those images forms the initial population of the IGA, and are represented by a vector containing their visual description.

The user provides relevance information on those images, giving a grade between 0 and 1 to each image. On one side, those grades are used to build a training set for the SVM, each image with a grade above 0.5 is considered as relevant and others are considered irrelevant. On the other side, they are used as the fitness for the individuals of the IGA. The SVM learning process and the IGA evolution process (selection, crossover and mutation) are run in parallel. Then, given a similarity function between images and individuals resulting from the IGA evolution process, the 8 images the most similar to an individual are selected. The 3 images with

the best SVM score are also selected and added to the 8 others to form the new population. Those 12 images are shown to the user for relevance feedback and the process continues until at least 80% of the results are annotated as relevant by the user.

Using some of the best images found by the SVM to feed the IGA, authors expect to boost the convergence of the IGA towards user's preferred solutions. The classification is of less importance in this work, the focus being on the convergence of the IGA and the relevance of the 12 images retrieved by the system. In such a system, the training set will naturally contain more relevant than irrelevant images, due to the fact that the new individuals are the one with the best scores at each generation. The classification resulting from such a training set may have poor performances, SVMs having better performances when learning from quite balanced training sets. At each generation, images from the database must be ranked using the SVM and the similarity measure from the IGA. Therefore, the similarity of each image from the database to each individual is computed. Those ranking procedures are costly and may slow the system down. This system was tested on a database with 2786 images only, retrieving 12 images for the user. Adaptations are needed to tackle bigger datasets, because the ranking procedure might take too long to perform for an interactive system with bigger datasets. In addition, users may want to retrieve more than 12 images. This system cannot do this, we would need some way to rank the whole database and present this ranking to the user. Doing so will also remove the link between the number of individuals in the GA's population and the number of images returned as a result to the user. Finally, GAs are known to require several generations to converge, and here only one generation is executed between each relevance feedback. Even if the individuals introduced by the SVM may boost the convergence, dealing with bigger datasets may result in a system needing a lot of iterations to converge, resulting in fatigue for the user.

[Shi *et al.* 2007] used a very similar technique but applied to the identification of a specific face in a database of human faces. In their work the GA has only one objective: the minimization of the similarity with the image the user has annotated as the best image. This is a bit restrictive, and it will not be effective for multi-modal searches as it focuses on only one place of the search space.

[Yu *et al.* 2016] also uses a similar framework, but with interactive DE instead of the GA. They also use only one objective, which is a weighted sum of the similarity of the individual with the query image and a value based on the SVM score that represents the user's expectation. Using the SVM as an approximation of the user's expectation is a good idea. However, the use of a weighted sum instead of two objectives limits the number of solutions the algorithm can retrieve and does not allow to exploit different balance between the raw similarity measure and the measure based on the user's expectations. This scheme was used for interactive image search on the same database as [Lai & Chen 2011] containing 1000 images only.

[Picard *et al.* 2008, Picard *et al.* 2012] uses SVM in combination with ant algorithms to build a distributed content based interactive image retrieval system.

Images are retrieved from a network of hosts. A SVM is used to rank the images in each host, and the ant algorithm determine how many images are retrieved from each host. Hosts providing the most relevant images are preferred over the others. The retrieved images are shown to the user, who can annotate them as relevant or irrelevant. Given the annotation, the SVM is retrained and the weigths of the network for the ant algorithm are optimized before the process starts over.

Recently, firefly algorithms were used together with SVM by [Kanimozhi & Latha 2015] to perform interactive image retrieval too. Given an query image provided by the user, visual similarity is used to perform the first ranking. The user can then provide feedbacks on the images, which are then used to train a SVM. The N images having the best SVM score are used as the initial population for a firefly algorithm. Then, one iteration of the evolution process of the firefly algorithm is run, with the objective function f_{FF} presented in Equation 2.19. R and I are the set of images annotated as relevant and irrelevant respectively. EMD is the earth mover's distance [Rubner *et al.* 1998], a distance adapted to histogram comparison and sometimes used in CBIR. This fitness function is to be minimized, and it decreases the closest an individual is from images in R and the furthest it is from images in I.

$$f_{FF}(A) = \frac{1}{|R|}\sum_{i=1}^{|R|} EMD(A;R_i) + \frac{1}{\frac{1}{|I|}\sum_{i=1}^{|I|} EMD(A;I_i)} \tag{2.19}$$

Once this iteration is done, each individual from the population of the firefly algorithm is matched to an image of the database (as for other methods using EAs, the evolution process may have moved the individuals to points not corresponding to any image from the database). Those images are presented to the user, if he is not satisfied, he can annotate those images and the process will start over from the SVM learning process, taking into account all annotated images (the new and old ones).

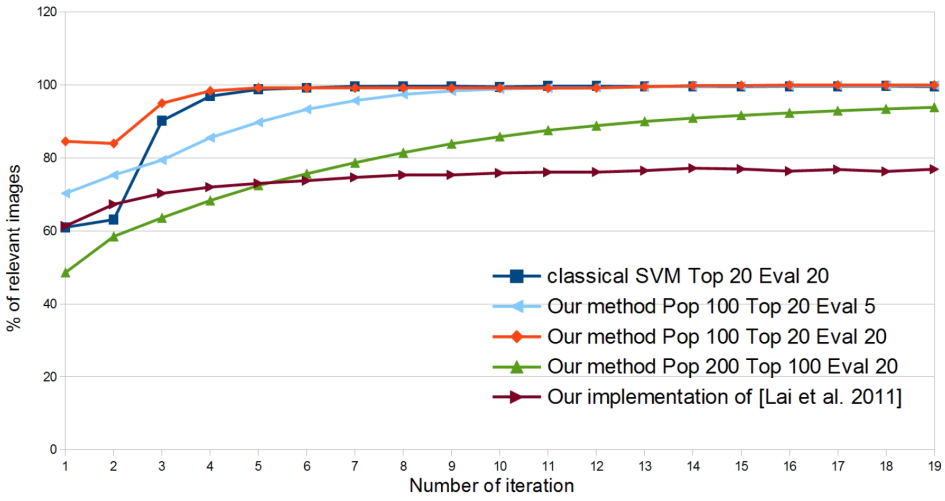
This system is compared to other hybrid systems, one based on PSO and one based on a GA using a similar framework, as well as a system based on SVM only. It outperformed other methods on three databases of 2500 images extracted from Corel, Caltech-101 and Pascal datasets. The evolution process of the EA is run only one time at each iteration in this framework, which is usually not enough for an EA to converge. In addition, using a single objective leads the population to converge to one point in the search space, which is contradictory with the exploration expected. As the datasets are small, the lack of exploration may not be punished that much, and as the convergence is not attained, the fact that the population converges to one unique point may not be a problem. However, when dealing with bigger datasets, those could become problematic.

Looking at those systems, using a GA and a SVM together to address the fine grained classification challenge seems to be a good idea. SVMs are chosen because of their great performances when dealing with image classification. The main goal of the developed framework is to associate them with an EA to provide the ex-

ploration part they are lacking so as to boost their performances. While several EAs have been used together with SVMs, GAs allow a great flexibility as almost every of their component (from the initialization to the way individuals are recombined to form offspring) can be tuned to fit a particular problem. This is not the case for DE or the firefly algorithm for example, in which more things, such as the way the variation process is performed for example, are predefined. Therefore, in other to be more flexible, GAs are chosen to be part of the system over other EAs. Then, to take full advantage of the SVMs performances, they are used as the classification algorithm in the proposed framework. The GA is there to help identifying the best training samples possible for the SVM, and adding them iteratively to its training set. This interaction is new, as previous works are most of the time using the SVM to boost the GA convergence, or using both algorithms in parallel. So as to test this idea, a first implementation was made based on the same image descriptions and the same dataset as [Lai & Chen 2011]. [Lai & Chen 2011] is presented in section 2.3.1. It performs interactive image retrieval from a query image using an IGA with a weighted sum of the user's feedback and the similarity to the query as the fitness function. The proposed system uses a bi-objective problem, separating the two components of the fitness into two objectives. NSGA-II is used to solve the resulting Multi-Objectives Optimization Problem. In combination with the GA, a SVM is used; learning from the GA's best individuals. At each generation, the GA's best individuals are mapped to their closest image from the dataset. The images to which they were mapped are added to the SVM training set after the user has given relevance feedback for them. The SVM is then trained with the augmented training set, and images in the database are ranked in decreasing order of their SVM score. This system presented better performances than the system proposed by [Lai & Chen 2011], as shown in figure 2.23 extracted from the paper. This was encouraging and was published as part of the ICPR 2012 conference [Pighetti *et al.* 2012].

This system is not dedicated to fine grained classification though, it does interactive image retrieval (which was done this way to be comparable to [Lai & Chen 2011]) while fine grained classification is a non-interactive classification task. In addition, the database on which it was tested contains only 1500 images (one image from each class is presented on figure 2.24), which is very small in the image retrieval community where image databases usually ranges from several tens of thousands (like Caltech-256 [Griffin *et al.* 2007]) to several millions (like imageNET [Russakovsky *et al.* 2015]) of images. The system also presents a lot of flaws. For example, the fact that relevance feedback is extracted at each generation of the GA is not optimal, it would be better to have the algorithm converge a bit before using its individuals. The maping system, which uses a exact nearest neighbor search in this first implementation, doesn't scale when the database grows too. Adaptations to the system must be made accordingly, and it must be tested on more relevant datasets. However, the performances obtained with this first experiment are promising, and the coupling between the GA and the SVM seems to be working fine. Therefore, this system was improved and adapted to the fine grained

Figure 2.23: Results obtained in [Pighetti *et al.* 2012]. *Our method* reffers to the method they developed.



classification challenge to form the hybrid framework presented in details in the next chapter.

Figure 2.24: Database used in [Lai & Chen 2011].

Chapter 3

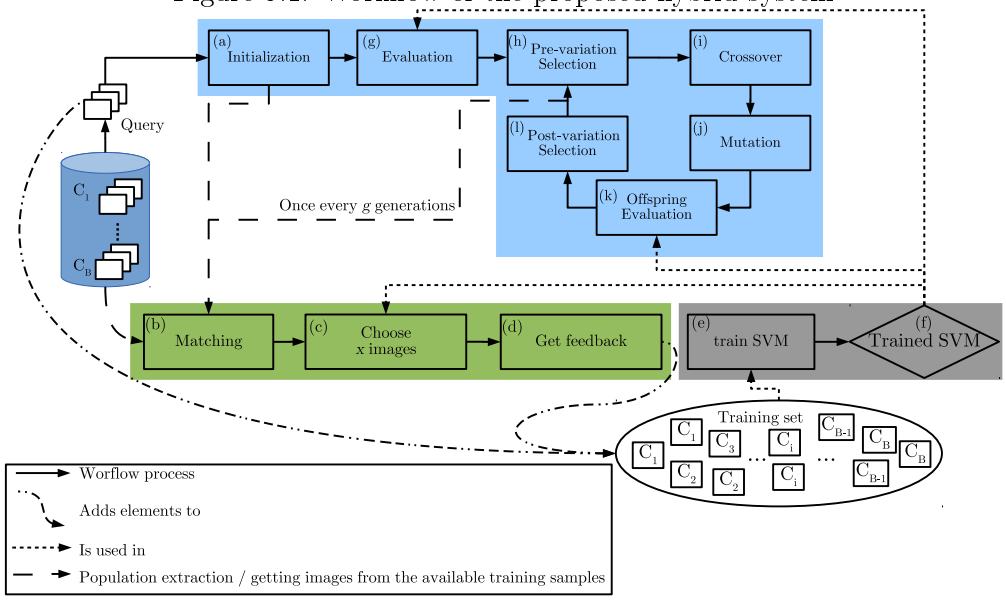
A Hybrid Framework to Address the Fine Grained Classification Challenge

Contents

A first novel hybrid system was built, briefly presented at the end of section 2.4. It uses a GA to provide training samples to an SVM, unlike other hybrid techniques which are most of the time using the SVM to boost the GA convergence, or using both algorithms in parallel. Despite presenting a lot of flaws, this system performances were promising. Therefore, it has been enhanced to form the hybrid framework dedicated to the fine grained classification presented in this chapter. This framework and the results obtained with the implementation we made of it have been published in [Pighetti *et al.* 2015b].

The first section of this chapter is dedicated to the presentation of the framework workflow. The different parts of the framework are introduced and the interactions between them are detailed. No implementation details, such as which specific algorithm or representations used for images or individuals, are given in this first section; it will be done in the second section of this chapter. The second section presents which algorithm is chosen for each part of our system, their parameters and the way images and individuals of the EA are represented.

Figure 3.1: Workflow of the proposed hybrid system



3.1 The System Workflow

This system is meant for classification, and thus learns a classifier for a database given a set of available training samples. The database is considered to be containing B distinct categories: $C_i, i \in [1 : B]$. The classifier must be able to classify any image from the database into one of those classes. It is learned using a SVM. The SVM cannot be learned from the whole set of available training samples in the ever growing datasets encountered nowadays. In this system, the training set is built iteratively with the help of an EA. The EA is used instead of classical active learning techniques for its exploration capabilities, the benefit of which has been introduced in Chapter 2.

The workflow of the proposed system is presented in Figure 3.1. The blue cylinder on the left represents the set of available training samples, each labeled with their class. The blob labeled training set at the bottom is the training set of the SVM, it contains a subset of the available training samples that is used to train the SVM.

The system is composed of three main components: the EA represented in blue at the top; the relevance feedback process represented in green at the bottom left; and the SVM represented in gray at the bottom right. Those components are interacting between each others in several ways:

- "add elements to" interaction: the query (top left of the figure) and the relevance feedback (step (d) on the figure) provide images to the training set (bottom right of the figure).
- "Is used in" interaction: it brings the training set (bottom right of the figure) to the SVM (gray part of the figure), and it brings the SVM score to the

evaluations steps of the EA (steps (g) and (k) on the figure) and it also brings the SVM score to the image selection (step (c) on the figure).

- "Population extration" interaction: the population of the EA is extracted at initialization (step (a) on the figure) and every g generation after the post-variation selection (step (l) on the figure) to be treated by the relevance feedback process (green part of the figure). It will match each individual to an image from the set of available training samples, thus extracting images from the available training samples.

To better understand how the system works, the execution of a run is explained step by step in the next few subsections, starting with the initialization. It is focusing on the framework general workflow and the interactions between the different part of the system. This means in particular that specific algorithm choices, parameters or evaluation functions for example are not covered in this section.

3.1.1 Initializing the System

To initialize the learning process, the system needs one query image from each category. So B images are drawn from the set of available training samples, one from each category, to form the query images. Those query images are added as the first samples to the training set of the SVM.

Then, the initialization process of the EA takes place. This process has access to the query images if needed. Given a query image from a category, it is very likely that several images in the vicinity of the query belong to the same class. Therefore a biased initialization could be considered to help identifying the core of the classes. However, an initialization focused too much on a specific area of the search space could hinder exploration. Thus this must be done carefully, as exploration is essential in this system.

The fine dotted lines going out from step (f) in Figure 3.1 show that the SVM is used during the evaluation step of the EA. However, the training set of the SVM contains only one image per category, which is too few even for a rough first classification. Therefore, before the GA loop starts, the relevance feedback process (steps (b) to (d) in Figure 3.1) is run, adding some images to the training set and growing it into a relevant first training set.

3.1.2 The Relevance Feedback Process

The population of the EA is extracted and processed through the relevance feedback process, represented in green on the figure. The goal of this step is to match each individual to an image among the set of available training samples and then select which images are added to the training set of the SVM.

This process is composed of 3 steps:

- matching individuals to images from the available training sample set (step (b) in Figure 3.1).

- selecting which images are selected to get relevance feedback from and to be added to the training set (step (c) in Figure 3.1).
- collecting the relevance feedback information and adding the images to the training set (step (d) in Figure 3.1).

Those steps are presented one at a time in the following.

3.1.2.1 Matching Individuals to Images From the Available Training Samples Set

Once the population has been extracted from the EA, it is first processed through the matching procedure (step (c) in Figure 3.1). The purpose of this step is to match each individual from the EA's population to an image from the set of available training samples. This is necessary because individuals' genome may not be the same as the image description, in which case a function needs to be defined to evaluate the similarity between an image description and a genome. And even if the genome has the same format as image description, due to the evolution process of the EA, individuals may not represent any image from the dataset. Only existing images can be used as training samples for the SVM. Therefore, individuals must be transformed to match images from the set of available training samples before they can be used as training samples.

The similarity measure provided with the image description is used to find the image with the closest representation to the individual genome. Some images cannot be used to represent individuals though. As the goal of the relevance feedback process is to add new images to the training set, images that are already in the training set are not eligible to be chosen to represent individuals. In addition, once an image has been chosen to represent an individual, it cannot be used to represent another individual during the same execution of the matching. This ensures that enough different images are present for the following selection step. If there are not enough images left in the set of available training samples, the remaining images are added to the training set, the SVM is learned from it and the learning process stops.

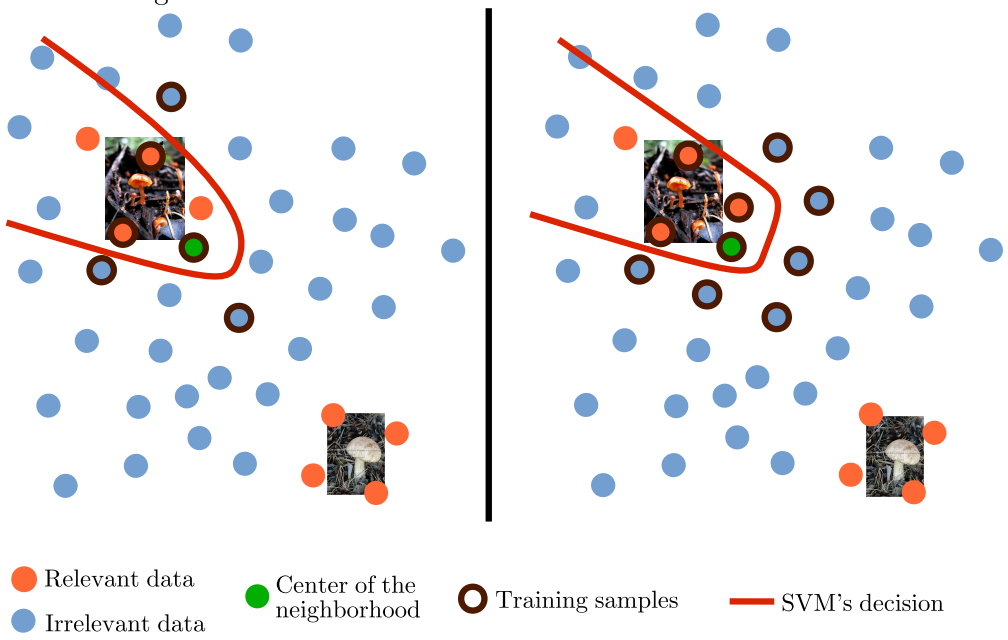
Given the size of the databases (and thus of the set of available training samples) and the number of individuals to match, the matching process must run as fast as possible to keep the system in a reasonable execution time. This must be kept in mind when implementing the system.

Once each individual has been matched to an image from the set of available training samples, the images to be added to the training set must be selected (step (c) in Figure 3.1).

3.1.2.2 Selecting Which Images to Add to the Training Set

The EA process is responsible for identifying a set of good candidate individuals to be added to the training set. Those are matched to images from the set of

Figure 3.2: Influence of adding several neighbors against adding just a few neighbors to the training set on the SVM's decision.



available training samples by the previous step. The EA is likely to contain a lot of individuals, and thus a lot of images are available at this step. Adding all of those images to the training set is not a good solution. Indeed, the EA is expected to have quite converged before the relevance feedback process is run, and even though it is expected to converge to a diverse set of individuals, it is likely that several individuals will lead to images close to each others. Adding several training samples close to each others will not improve the decision of a SVM much more than adding just a few of them however. This is illustrated by Figure 3.2, which presents two states. The green circle is an orange mushroom image that is the center of the neighborhood in which images are selected to be added to the training set. Thick borders indicates training samples, orange dots represent mushroom images and blue dots represent non mushroom images. The red thick curve represents the SVM decision built using the training samples. Green marks around dots stands for misclassified images. Both present almost the same decision and thus almost the same classification results. Then adding a lot of images from the same area of the search space does not improve the performances that much and increases the learning time.

Therefore, only x images are extracted to get relevance feedback from and be added to the training set. The EA is responsible for the exploration, at this stage the selection should focus on the images that will improve the SVM decision the most. This selection step may have access to the SVM to compute scores for images if necessary. The SVM is not available when running the relevance feedback process for the first time right after initialization of the EA however. Therefore, the first

selection may defer from the subsequent ones. The details about the selection scheme are left to the choice of the implementer; the details about the implementation made are presented in section 3.2.

Once the images have been selected, relevance feedback is collected for them, which is the next and last step of the relevance feedback process (step (d) in Figure 3.1).

3.1.2.3 Getting Relevance Feedback

The input of this step is a set of x images, and the goal is to provide relevance feedback for them. In a multi-class classification context, getting relevance feedback for an image is getting the class it belongs to. In an active learning context, this information is provided for each image in the available training samples set. As any image selected belongs to the set of available training samples, their classes are checked and they are added to the active current training set with their label.

The multi-class SVM is then trained, or retrained, using the new training set (step (e) in Figure 3.1). Once the training is done, the SVM score is available (step (f) in Figure 3.1) for classification or to help in the evaluation of individuals (steps (g) and (k) in Figure 3.1) and in the subsequent selections of the relevance feedback process (step (c) in Figure 3.1). As running the relevance feedback process changes the SVM score, and as this score can be used in the EA evaluation processes, it is important that the EA process does not enter any evaluation step while the relevance feedback process and the SVM training process are not done. Indeed, doing so would result in evaluating the individual with an outdated version of the SVM, which has not been trained with the latest training samples added to the training set. Therefore, the EA process is paused before any evaluation step while the relevance feedback and the SVM training processes are running. It is resumed once the SVM training is terminated, using the freshly trained SVM to evaluate the individuals. The EA must pause before any evaluation step if this process is running.

3.1.3 The Evolutionary Algorithm Process

Once the population and the SVM have been initialized, the EA process can start. The first step is to evaluate the individuals of the population (step (g) in Figure 3.1) generated during the initialization. This evaluation step is the same as the evaluation of offspring done later in the EA process (step (k) in Figure 3.1) and is presented first below.

3.1.3.1 Individual's Evaluation

The EA is expected to converge toward a diverse set of individuals representing interesting elements to add to the SVM training set to improve its classification score. The area of the search space to which the individuals will converge is determined

by the functions used to evaluate the individuals. Therefore, those need to be build carefully.

The goal would be to identify a diverse set of images from each category in the dataset. Several elements are available to craft the evaluation functions:

- The query image from each category
- The SVM learned so far
- Other individuals

The category of an individual is unknown, as individuals may not represent a real image from the available training samples; and matching it to an image every time an evaluation is needed would be too much time consuming. As the SVM score is available, it can be used as to approximate the individual's class if needed.

Any evaluation function can be built, keeping in mind that what is expected from the EA is to explore the search space and approach the best images to add to the training set to improve the SVM classification. Choices made in the implementation of the system are given in the next section. However, using a multi-objective evaluation is advised as the EA is expected to retrieve a diverse set of individuals and classic single objective EAs will often converge toward a unique solution. This convergence toward a unique point is bad as it would lead to a set of very close images after the matching process (step (b) in Figure 3.1). And as seen above, adding too many images close to each other to the SVM's training set increases the learning cost a lot for little or even no improvements of the classification performances.

Once the post-initialization evaluation process is done, the evolution loop starts.

3.1.3.2 The Evolution Loop

Any EA can be used in the proposed system. Figure 3.1 steps (h) to (l) shows the example using a GA. Although it can easily be adapted to any EA by replacing the evolution loop of the GA by the evolution loop of the desired EA. The evolution loop is not modified and can run any number of generations without the need of giving any feedbacks about the individuals of the population. This separation between the relevance feedback and the evolution loops leaves room for the EA to converge before its individuals are used to add new training samples to the SVM's training set. EAs need several generations for their individuals to start converging toward interesting solutions. The fact that the EA can run several generations before its individuals are used ensures that convergence has at least started to happen when individuals are used. This ensures that the individuals used in the relevance feedback process are not just random, but are relevant to the objectives of the EA.

To add new samples to the training set and improve the SVM's classification, the relevance feedback process needs to be executed at some point. This is done by extracting the population after the post-variation selection process every g generations and process it through the relevance feedback process. As stated before, the EA cannot go into any evaluation step until the relevance feedback process ends, adding

elements to the training set and retraining the SVM with the augmented training set. Once the SVM has been retrained, its classification function has changed because the training set contains new elements. The evaluation process of the EA uses the SVM to evaluate the individuals. This means that the evaluation function of the EA has changed, and thus the position of the optimal solutions have changed too. As a result, individuals of the EA will converge toward the new optimal solutions in the next g generations, thus exploring new areas of the search space. This will lead to new images retrieved during the next call to the relevance feedback. The process then continues until an end condition is met, or no more training samples are available to be added to the training set.

Finally, in this system the EA is expected to evolve toward a diverse set of individuals that represents interesting training samples for the SVM. Those individuals are periodically processed through the relevance feedback process, which adds images to the SVM training set. Modifying the training set changes the SVM's decision, and thus the position of the interesting training samples. As finding the interesting training samples is the goal of the EA, objectives of the EA change and thus individuals evolves toward new areas of the search space to identify them. By doing so, the EA is expected to explore the search space and thus better solve the fine grained classification problem.

Even though any EA could be used, the diversity in the solutions identified by it is key. Indeed, this diversity ensures that the individuals of the population are matched to a diverse set of images, and not a set of images close to each other, thus ensuring a better improvement of the SVM's decision. Thus the choice of the EA and its fitnesses must be done keeping that in mind when implementing the system.

This ends the description of the general workflow of the proposed system, the presentation of the different part involved in it and their interactions and their goals. The next sections exposes the details of the implementation we made of such a system.

3.2 Our Implementation Choices

Now that the general workflow of the system has been explained, it must be implemented to assess its performances. This section focuses on the presentation of the choices we made. Several things are to be defined when implementing such a system: the image representation, the individuals representation in the EA, the EA evaluation functions, which EA to use and its components, the algorithm used for the matching, the selection scheme in the relevance feedback process, the type of SVM used.

The image representation used for the SVM is presented first, followed by the presentation of the SVM used. Then comes all the details about the EA: the algorithm, the individuals' genome, and all other details about it. The details about the choices made for the components of the relevance feedback process (the image selection scheme and the method used for the matching process) are presented last.

3.2.1 Image Representation

Fisher Vectors computed on top of SIFT descriptors are used as the image representation. Those vectors were kindly provided to us by Florent Perronnin for the Caltech-256 database [Griffin *et al.* 2007]. They were used in their work [Perronnin *et al.* 2010] and are presenting quite good performances. Using those allows us to concentrate on the definition of the other parts of the system, and provides us with some results to compare our system to. Indeed, the focus in this thesis is not on the image representation but on the system used to perform the classification and its workflow. Any image representation could be used here. However, using a well known database with an image representation for which results are provided using other methods, allows us to better outline the performances of the proposed system.

The Fisher Vectors used are 2048 dimensions normalized vectors. The value of each component is comprised between 0 and 1.

The EA's evaluation process is using the SVM. Therefore, before getting to the details about the EA, the SVM scheme chosen to address the multi-class classification problem is presented.

3.2.2 The SVM

The SVM is used to learn the multi-class classifier from the training set, in the evaluation of the individuals of the EA and in the selection of images to be added to the training set in this system. As mentioned in Subsection 1.2.4.3, two main schemes exist to address the multi-class classification problem using SVMs: the *one versus one*, and the *one versus all* strategies. In both strategies, the algorithm is given a training set TS containing n images from every class together with their label. Let B be the number of categories in the database.

In the *one versus one* strategy, a binary SVM is learned for each pair of classes, using only the images of the two concerned classes as training samples. Using a *one versus one* strategy implies learning $\frac{B\times(B-1)}{2}$ binary SVMs with approximately $2\times\frac{n}{B}$ training samples for each, considering that the number of images provided for each class is approximately the same. A voting strategy is then used to determine to which class the presented image representation must be assigned.

The *one versus all* strategy however only uses B binary SVMs, one for each class. But every SVM is learned using all n images from TS. Indeed, for each SVM_i, images in the class C_i are used as the relevant images, and other images are used as irrelevant images. In the end, SVM_i is dedicated to identify the separation between images from C_i and other images of the database. Then, the decision is taken by comparing the SVM score obtained by an image representation vector with each SVM. The SVM with the best score tells the class to which the vector is associated.

The training complexity of both methods is similar, one is using a lot of SVMs with small training set sizes, and the other a small number of SVMs with bigger

training sets. But in the *one versus all* strategy, the relevance of a particular representation vector against C_i can be obtained by the score obtained by the vector with SVM_i. This is a lot easier than in the *one versus one* strategy where $B-1$ SVMs concern C_i, a combined result may be computed but it would involves computing the score of $B-1$ SVMs, which costs less than computing the score of only one as for the *one versus all* strategy. Because the SVM is used in several steps of the system, and because evaluation of the SVM with respect to a specific class is often needed, the *one versus all* strategy is chosen in our implementation.

The kernel used is a linear kernel, which is also used in [Perronnin *et al.* 2010] and has proven to be effective. The regularization parameter of the SVM is set to $C = 1$.

Now that the image representation and the SVM scheme have been chosen, the next subsection introduces the choice of the EA, the details of its components and its parameters.

3.2.3 The Evolutionary Algorithm

The first thing to define when building an EA is the representation of the individuals. Thus they are presented first, followed by a presentation of the construction of the population and the initialization process. Then, the evaluation functions as well as the algorithm used are presented. The generation loop and its components are then detailed.

3.2.3.1 Individual's Genome

The EA is expected to identify training samples for the SVM. Training samples are images. Those already have a representation, given in section 3.2.1. In order to not create another representation that would bring its own drawbacks, the genome of individuals in the EA have the same form as the image representation presented before. That is to say the individual's genome is a vector or 2048 floats, each of which is bound to $[0, 1]$.

3.2.3.2 Population and Initialization

The initialization process of the EA is responsible for the creation of the population and the first individuals that populates it. In our context, the goal of the EA is to converge toward the best training samples to add to the SVM's training set to improve its classification results. The *one versus all* strategy is used, meaning that one binary SVM per class is learned. Improving each SVM separately should result in an overall performance improvement. Therefore, it has been decided to separate the population of the EA into B (the number of categories to be identified in the database) independent subpopulations $P_i; i \in [1, B]$: one for each category to be identified in the database. The subpopulations are all initialized and evolved separately, as if there were B distinct EA running in parallel. The goal of each subpopulation P_i is to identify interesting training samples for the category C_i, i.e.

improving the classification of SVM_i in the *one versus all* SVM. All the subpopulations have the same number of individuals, which is set to 20 in this implementation.

At initialization, the EA is given one query image from each searched category as an input. Given the queries $Q_i, i \in [1, B]$, for each Q_i, there should be some images with a representation close to Q_i's representation that are part of the same category C_i as Q_i. This is the goal of image representations: getting similar images close to each other in the description space. Even though it cannot achieve this for all images, it is the problem raised by the fine grained classification, there are most of the time several images from the same classes in a given neighborhood. Thus initializing each subpopulation around the corresponding query allows to grasp part of each category and gives a good starting point for the SVM when extracting its first training set from the initial population. Indeed, such an initialization allows to identify quite well at least one mode, one area of the search space containing relevant images, for each class. In order to keep some diversity in the population however, each population P_i is initialized using a normal law centered on Q_i. This is necessary to allow the EA to explore the space more easily. If all the individuals are stuck in the same place of the search space, the genetic operators would take a lot of generations to explore the search space. Using randomness in the initialization ensures a bit of diversity and thus a better exploration of the search space from the first generations of the EA.

3.2.3.3 Objectives Functions

Two main objectives are identified: finding images that will quickly improve the SVM decision, and explore the search space to find new areas with images from each category to better address the fine grained classification. So as to fulfill those goals, two fitness functions were defined. The population is composed of B independent subpopulations, each dedicated to the improvement of the classification of one class. Therefore, each subpopulation evaluation functions are adapted to the class it is focusing on.

First, as the initialization is done mostly around the query images Q_i, this area is already exploited and we would like to explore other areas of the search space. So as to do so, for each population P_i, the l_2 distance to Q_i forms the first evaluation function which is to be maximized. This first objective is called f_1 and is computed using Equation 3.1, with *ind* the individual being evaluated. This ensures that individuals are getting away from this area of the search space and enhances exploration in every direction.

$$f_1(\mathbf{ind}) = l_2(\mathbf{ind}, Q_i) = \sqrt{\sum_j (\mathbf{ind}_j - Q_{i,j})^2} \text{ to be maximized} \tag{3.1}$$

As for the second objective function, for each population P_i, it is the the absolute value of the score obatined by the individual with SVM_i, which is to be maximized.

It is named f_2 and is computed using equation 3.2.

$$f_2(\mathbf{ind}) = |SVM_i(\mathbf{ind})| \text{ to be maximized} \tag{3.2}$$

This objective fulfills two goals. First, it helps identifying the core of the classes by identifying more relevant images on the relevant side of the relevant side of the SVM decision. Second, it can identify relevant images on the irrelevant side of the SVM's decision by exploring this space. Identifying such images is what is expected from the GA to help solving the fine grained classification challenge. Indeed, those are the misclassified images far from the SVM's decision that are often missed by other active learning techniques. They are probably part of a new mode of images from C_i, and identifying them allows the SVM to classify them correctly and enhance its performances.

3.2.3.4 Evolutionary Algorithm Used

The problem defined is a bi-objective problem. Any multi-objective EA could be used to solve it. NSGA-II is used in this implementation. It is presented in section 2.3.2.1. It has been chosen because it is known to have decent performances in a wide range of applications.

3.2.3.5 Generation Loop

The generation loop starts with the pre-variation selection process. Then, the variation takes place, composed of a crossover and a mutation, and finally there is the post-variation selection process. As NSGA-II is being used, the post-variation selection is determined by the algorithm and is unchanged. It is explained with NSGA-II in section 2.3.2.1 and is not discussed here. Each of the other components is detailed, one at a time and in the order of the generation loop, starting with the pre-variation selection process. A last paragraph is dedicated to the parameter g of the framework, fixing the number of generations run between two calls to the relevance feedback process.

Pre-Variation Selection In the proposed implementation, the simple but still efficient tournament selection is used. It consists in selecting randomly t individuals in the population and keeping only the best one. In this implementation, $t = 2$ has been chosen. The comparison of individuals is done using the standard Pareto dominance relation and the crowding distance. Once the parents have been selected, children are generated using the crossover operator presented next.

Crossover The two-points crossover presented in Subsection 2.2.2 is used to generate the offspring. Swapping part of the genome between individuals makes sense for images. Taking a simplified examples with the orange and white mushrooms presented previously, if the image representation is a concatenation of a color and a

shape descriptors, then mushrooms have similar shapes but different colors. Swapping the descriptors of a white mushroom with anything orange would allow to find the orange mushrooms. Even though more complicated, each component of the Fisher vectors is capturing some characteristics of the images, and images from the same class may have similar values for some components and completely different values for other components. Therefore, swapping part of the genome allows an interesting exploration of the search space for image retrieval.

The selection of the two points for the cut is done randomly using a uniform law.

Once the offspring have been created, they are processed through the mutation operator to form the definitive offspring.

Mutation The mutation operator is the simple reset mutation. In this mutation scheme, each gene has a small chance, called the mutation probability, of being randomly reset within the bounds of the search space using a uniform distribution. In this implementation, the mutation probability is set to 0.3.

Even though a uniform random reset seems to be a bit brutal for a mutation, it is quite adapted to the problem. Indeed, images from the same class may have quite different values in some components of their genome, using such a mutation scheme allows to greatly modify some of the components, thus allowing to find new interesting individuals.

In addition, after the GA has converged for g generations, training samples are added to the SVM's training set, thus modifying its decision and consequently modifying the second objective of the GA. This means that the solutions forming the optimal Pareto set have changed. Then this kind of reset mutation allows to move the population away from the areas they have converged to previously faster, enhancing exploration and allowing a faster convergence toward the new best areas of the search space.

In addition, this mutation scheme is able to generate any vector of the search space from any individual. Indeed, each component of a vector has a chance of being modified, and the new value is picked randomly within all the available values. Combined with the post-variation process imposed by NSGA-II that is using elitism, the GA convergence is ensured as proved by [Zitzler *et al.* 2004].

The post-variation selection process is the one used in the standard NSGA-II, so this ends the description of the evolutionary process components. The next paragraph concerns the setting of g, which even though not an intern parameter of the GA is tightly linked to it as it determines how much time the GA has to converge.

Number of Generation Between Relevance Feedback Process Calls The relevance process extracts the population (composed of B subpopulations) of the GA every g generations. In the implementation, $g = 30$ is used. It gives some time to the GA to converge, while not allocating too much resources to it. In addition,

a full convergence of the GA is not necessarily needed in the proposed framework. Indeed, individuals are mapped to existing images which representations are not exactly corresponding to the individuals' genome. So the image representation added to the training set is not exactly the individual retrieved by the GA, it is the closest neighbor to it. Thus finding the exact best position is not necessary, a good approximation leading to the same neighbor is enough in this context. 30 generations then seems like a good compromise between convergence and resource investment.

The last things to detail are the methods used to perform the different steps of the relevance feedback process, presented thereafter.

3.2.4 The Relevance Feedback Process

This process is composed of three steps: the matching process, a selection process and finally gathering the feedbacks before adding the images to the training set. In the classification context studied here, gathering the feedbacks only consists in reading the category to which an image belong in the set of available training samples. Then only two steps need to be detailed: the selection scheme used to select which image from the one obtained by mapping the population to images are chosen to be added to the training set; and how the individuals of the population are matched to images from the set of available training samples. The selection scheme is described first, followed by the matching process.

3.2.4.1 Selecting the Images to Be Added to the Training Set

The populations of the GA contain more individuals than images we want to add to the training set each time the relevance feedback process is called. As the GA is responsible for identifying interesting areas of the search space, every image within the populations might be interesting in some way. Indeed, the GA is converging toward a diverse set of individuals that should lead to a diverse set of images, putting pressure on the selection of the images to be added to the training doesn't seem necessary. In addition, choosing a criteria to chose the images would mean favoring exploration or exploitation in some way. And it is not the goal of this method, which try to combine both exploration and exploitation without favoring too much one or the other. Hence a uniform random selection is used to extract $\frac{x}{B}$ images from each population and form the set of x images that will be added to the training set. The same number of images is extracted from each matched population so that each SVM of the multi-class SVM are given the same number of training samples dedicated to its improvement. The number of images extracted from each mapped subpopulation and added to the SVM training set is chosen to comply with the experimental setup presented in the next chapter. It set to $x = B * 15$.

Before getting to this selection steps, individuals of the population must be matched to images from the available training samples set though. Below is the explanation on how this is achieved in the presented implementation of the system.

3.2.4.2 Matching Individuals to Images of the Available Training Samples Set

After several generations of the GA, individuals of the populations are most likely not corresponding to any image from the set of available training samples. To get back to images from this set, each individual is matched to an image having a representation close to the individual's genome. Searching for the exact nearest neighbor is a quite expensive task tough, and doing it would slow the system too much. Therefore, in order to lower the impact of the matching on the speed of the system, a fast approximate nearest neighbor search based on Locality Sensitive Hashing (LSH) is used instead. LSH are thus introduced below before the way they are used in the presented implementation is detailed.

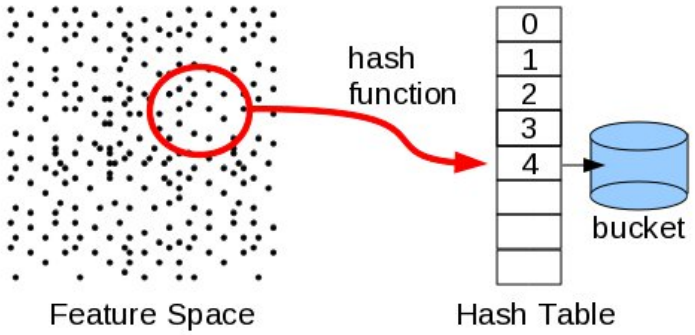
Locality Sensitive Hashing LSH [Gorisse *et al.* 2010] is a hashing technique aiming at giving a fast approximate solution to the $(R; cR)$ Nearest Neighbor problem, shorten as the $(R; cR)NN$ problem in the following. This problem consists in, given a set of data and a particular data d, finding at least one neighbor at a distance less than cR from d, $c > 1$, if there exists at least one data in the dataset at a distance less than R from d.

Hashing consists in using a hash function to transform a data into a fixed-length value, often an integer used as an index in computer science. Then, a table is constructed by storing each data to the slot with the index corresponding to its hash value. This eases the search of items, as instead of comparing the data itself, only the hash values are compared, which is much more easier to do. The domain of the hash function, i.e. the number of data to hash, is usually larger than its range, i.e. the number of hash values it can take. Those situation are called collisions, and because of this, each slot of the table dos not contain only one data, but it can contain several. Therefore, the slots of the table are often called buckets and the hash values bucket indexes. Collisions mean that several data are assigned the same hash, and thus when looking in the table, some data have to be fully compared to retrieve a particular data. Those are thus in general avoided because they slow down data search.

LSH takes full advantage of the collisions to provide a fast solution to the $(R; cR)NN$ problem. The idea is that if two data have a high chance of collision when they are close to each other, and a low chance of collision when they are far from each other, then each bucket contains a set of data relatively close to each other. Figure 3.3 shows this process graphically, where several data close to each others are all hashed into the 4^{th} bucket of the hash table. Then searching for neighbors for a specific query can be limited to searching in the bucket the query is hashed into, which reduces the cost of the search.

To perform such a hashing, dedicated hash functions must be used: locality sensitive hash functions. Locality sensitive functions ensure that, given a query data Q, the probability p_1, that a data A at a distance less than R from the query is put in the same bucket as the query, is higher than the probability p_2 that a data

Figure 3.3: The input space $\mathcal{D}$ is partitioned thanks to hash functions and data are assigned to buckets.



B at a distance higher than cR from the query, is put in the same bucket as the query. LSH uses several locality sensitive functions to build the structure needed to address the $(R; cR)NN$ problem. A family of locality sensitive function must be provided to the LSH algorithm to select hash functions from. The formal definition of a (R, cR, p_1, p_2)-sensitive family of functions is given in Definition 4.

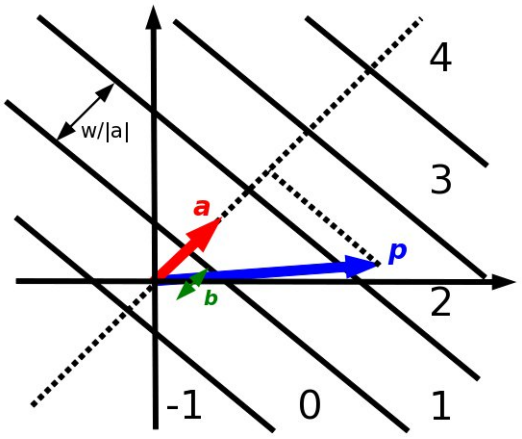
Definition 4 ((R, cR, p_1, p_2)-sensitive family of functions)**.** Let $\mathcal{D}$ be the space of data to be hashed, $(R, cR) \in \mathbb{R}^2$, $p_1 \in [0, 1]$ be the probability of true nearest neighbor detection, $p_2 \in [0, 1]$ be the probability of false nearest neighbor detection, $\mathcal{H}$ be a family of hashing functions and d be a similarity or distance function on $\mathcal{D}$. $\mathcal{H}$ is said to be (R, cR, p_1, p_2)-sensitive if, given a query data $Q \in \mathcal{D}$:

$$\forall h \in \mathcal{H}, \forall (A, B) \in \mathcal{D}^2$$
$$\text{If } d(A, Q) \leq R \text{ then } P_{\mathcal{H}}[h(A) = h(Q)] \geq p_1$$
$$\text{If } d(B, Q) \geq cR \text{ then } P_{\mathcal{H}}[h(B) = h(Q)] \leq p_2$$
$$\text{With } p_2 \leq p_1,\ R \leq cR \text{ and } c > 1.$$

Several family of locality sensitive functions have been presented in several works [Indyk & Motwani 1998, Gionis *et al.* 1999, Datar *et al.* 2004]. Random projections are the only one detailed here because they are the one used in the implementation of the presented system. The family of random projection functions is defined in Definition 5. Figure 3.4 shows a graphical example for one random projection in a 2 dimensions space. The space is separated into $\frac{W}{|\mathbf{a}|}$ width slices when using this function. $\mathbf{a}$ determines how the slices are oriented, they are perpendicular to it; and b determines how much the slices are shifted from the origin. The hash value of a data is obtained by looking the number of the slice it falls into. For example $\mathbf{p}$ falls into the slice 2, so its value for the represented random projection is 2. Random projections is a family of locality sensitive functions. When searching for the neighbors of a query point, only data having the same hash value, i.e. falling in the same slice as the query are studied. However the slices are infinite, so using only one hash function leaves a lot of space in which nearest neighbors are searched for.

Definition 5 (Random Projections family of functions)**.** Given $W \in \mathbb{R}$ the family of functions $H = \{ h_{\mathbf{a},b}$ from $\mathcal{D}$ to $\mathbb{Z}$ such that $\forall p \in \mathcal{D}, h_{\mathbf{a},b}(p) = \lfloor \frac{\mathbf{a}.\mathbf{p}+b}{W} \rfloor$, with $\mathbf{a}$ a random vector in $\mathcal{D}$, and b a random value in $[0, W[\}$ is a family of random projections functions.

Figure 3.4: One hash function (random projection) for $\mathcal{D} = \mathbb{R}^2$

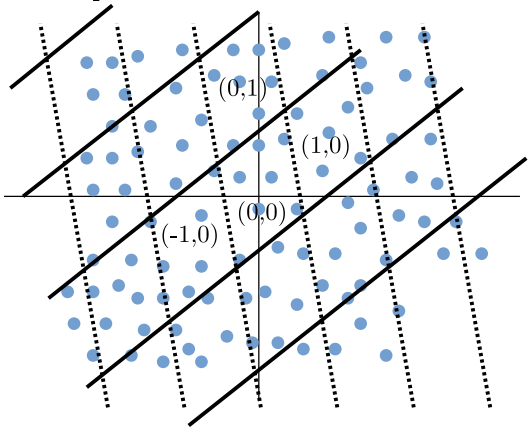


To reduce the amount of space, and thus the number of data points, evaluated when searching for neighbors, LSH is using several locality sensitive functions to form a more complex hash function that makes smaller slices of space. To do so, several functions from the family of locality sensitive functions are randomly drawn, and the hash value obtained in each of them are concatenated to form a hash vector. This hash vector is used as the hash value, instead of using the result of only one function. Figure 3.5 shows how the 2 dimensions space is cut when using two random projection functions. Plain lines represent the limit of the slices obtained with one random projection function and fine dotted lines the limit of the slices obtained with the other function. The space is then hashed into diamond shaped slices. The hash vectors of a few of those cells is written in their center. Every data point falling into the same cell are stored in the same bucket. Using several hash functions reduces the size of each cell or slice of space, and thus the number of elements in each bucket. The more hash functions are used to build the final hash function, the smaller the slices are and the less elements there is in each bucket. It allows to be more discriminative, separating in different buckets data points that were far away but in the same bucket when using only one function. This also speed up the search as less points are evaluated when looking for the neighbors of a query.

When searching for the neighbors of a query point at the edge of a slice however, the precision can be bad. Indeed, the nearest neighbors of this query can be at the edge of a neighboring slice, and thus falling in another bucket. Therefore, they will not be retrieved when searching for neighbors in the same bucket. This is why the neighbor search is approximate when using LSH.

Even though finding the nearest neighbor can't be guaranteed, the precision can be increased. To do so, instead of building only one hash table with k hash

Figure 3.5: $\mathcal{D} = \mathbb{R}^2$ partitioned with one table of two hash functions



functions randomly drawn from the locality sensitive family of functions, L tables are constructed, each with its own set of k randomly drawn hash functions. Even though functions are drawn from the same family, they are different for each table, and thus separate the space in different ways. Thus the borders of the slices of space are not the same, meaning the the bucket do not contain the same elements either. Then, when looking for the neighbors of a query point, the hash vector for the query point is computed for each of the L tables. The L corresponding buckets are then merged to form the set from which neighbors will be drawn. This is like repeating Figure 3.5 L times with different orientations of the slices each time, and then taking the the data points falling into the same diamond shaped slice as the query point in each of the figure as potential neighbors. Those potential neighbors are then ranked by their distances to the query point and form the result of the neighbors search. As the space is not cut in the same way in each table, some close data that were in different buckets in one table can be in the same bucket in another table. So although increasing the cost of the search, using more tables also increases the precision of the search.

At the end, the LSH structure is a set of L hash tables containing the buckets defined by hashing the data using k hash functions randomly drawn from the family of locality sensitive functions provided. Then, when searching for the neighbors of a query point, the hash vector of this query is computed for each of the L tables, and the L corresponding buckets are merged to form the pool of retrieved neighbors, which is ranked by the distance of the elements to the query point.

The standard implementation of LSH provides a solution to the $(R; cR)NN$ problem with a computation complexity of $O(d^{\rho})$ with $\rho < 1/c$ and d the dimension of $\mathcal{D}$ space once the structure has been built. Careful attention must be taken when choosing the number of tables L and the number of hash functions k. If those are not tuned correctly, one might have difficulties to find neighbors in some parts of the search space, because there are too few elements retrieved; or spend too much time computing the ranking of the neighbors there are too much elements in the

retrieved set of approximate nearest neighbors.

This ends the general presentation of LSH, the next paragraph shows how it was used in the proposed implementation of the CBIR system to match individuals from the GA's populations to images from the available training samples set.

Matching Individuals to images In our system, potential image representations issued from the GA must be matched to images from the set of available training samples with a close representation quickly. To do so, the set of available training samples is hashed using LSH with the random projections family of functions. The number of tables is set to $L = 10$ and the number of hash per table is set to $k = 1$. Then, to get the neighboring images of an individual, the set of neighbors retrieved by LSH is used.

For each individual **ind** from the GA, the set of neighboring images from the set of available training samples is obtained using the LSH structure built earlier with **ind** as query. The set of neighbors Ne is ordered in increasing distance to **ind**. Then, for each neighbor img_j, beginning with the closest to **ind**, if img_j is already part of the SVM's training set or has already been matched to another individual, img_j is discarded and the next neighbor is considered. Otherwise, img_j is chosen to represent **ind** and the next individual is processed. If every image in Ne is already assigned to an individual, then a random image is drawn from the available images in the set of available training samples (i.e. an image not in the training set and not already selected to represent another individual). The process ends when every individual in the populations has been assigned to a particular image. If there are not enough images left, all the remaining images are added to the training set and the SVM is learned with all the images in the set of available training samples as a training set. This is unlikely to happen though has the hole point of the method is to learn the SVM from a small subset of the available training samples.

3.2.5 Parameters summary

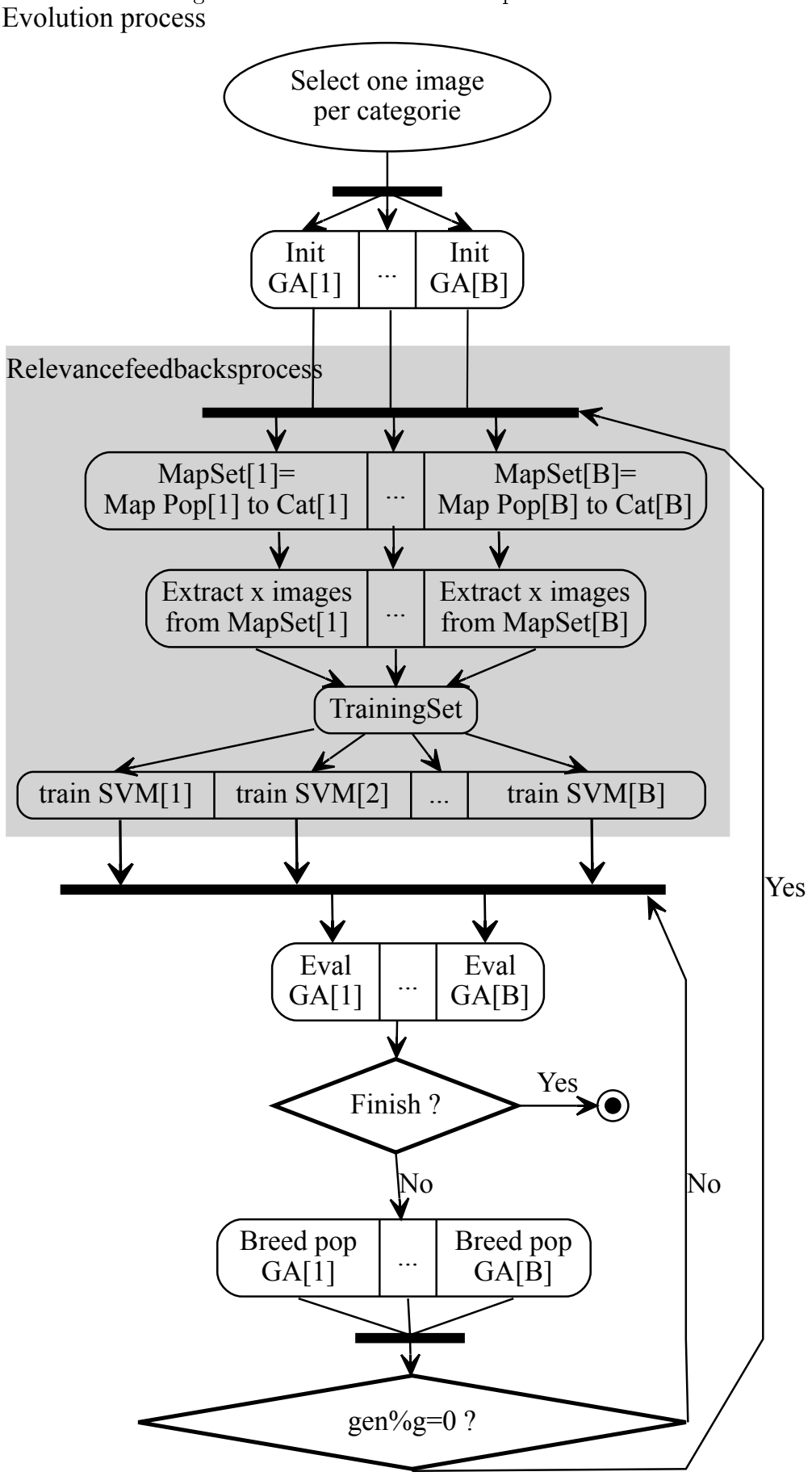
$$\begin{cases} f_1(\mathbf{x}) & = l_2(\mathbf{x}, Q_i) = \sqrt{\sum_j (\mathbf{x}_j - Q_{i,j})^2} \text{ to be maximized} \\ f_2(\mathbf{x}) & = |SVM_i(\mathbf{x})| \text{ to be maximized} \end{cases} \tag{3.3}$$

The parameters used for each part of the system are summed up in Table 3.1, and the objectives used in the GA in Equation 3.3. A flow chart of the implementation is also provided in figure 3.6. Horizontal black bars represent synchronization steps, in which every process before the bar must finish before the process under can start. This implementation has then been tested against other systems on the well known Caltech-256 database containing $B = 256$ categories. The details about the experimental conditions and the performances of this implementation of the framework are presented in the next chapter.

Table 3.1: Parameters used in the implementation of the system

Input	
Number of categories	$B = 256$
Queries	One per category $Q_i, i \in [1, B]$
Image description	
Type of description	SIFT based Fisher vectors
Format of the description	2048 dimensional real valued vectors normalized in $[0, 1]$
Evolutionary Algorithm	
Algorithm used	NSGA-II
Genome	2048 dimensional real valued vectors
Domain	$[0, 1]$
Population	B subpopulations of 20 individuals
Initialization	Normal law centered on Q_i
Pre variation selection	Tournament selection
Tournament size (t)	2
Crossover	Two points crossover
Mutation	Reset mutation
Mutation probability	0.3
g	30
LSH	
Hash Functions	Random Projections
L	10
k	1
Selection of images for feedbacks	
Selection scheme	Random in each matched subpopulation
x	$15 * B$ (15 per subpopulation)
SVM	
SVM scheme	*One versus all*
SVM library	liblinear
Kernel	Linear kernel
C	1

Figure 3.6: Flow chart of the implementation



Chapter 4

Performances

Contents

This chapter is dedicated to the evaluation of the performances of the implementation of the framework. The first section thus introduces the experimental setup, covering, among other things, the dataset used, the metrics, the maximum number of training samples in the training set. The performance analysis compares three algorithms: the proposed approach, a random selection and an active learning technique. Those are presented together with their parameters in the second section. The last section presents, analyses and compares the classification performances as well as the computation complexity of the three methods.

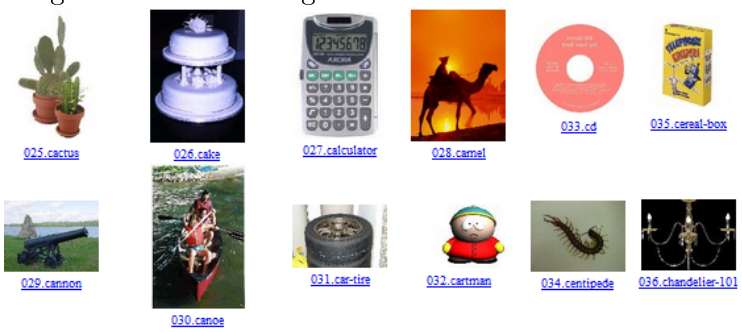
4.1 Experimental Setup

To assess the performances of the proposed system, it must be evaluated on an image database presenting the characteristics of fine grained classification for at least some of the classes to be identified. The database must be standard enough such that results for other systems are available. Our system should also be using the same image description as competitor systems, since what is evaluated is the learning scheme itself and not the image description. Therefore, the image database and the image description chosen for the test are presented first. Then, the measure used to compare the classification results is explained and, finally, the experimental process and how the final figures are computed are explained.

4.1.1 Image Database and Image Description

Several experimental image databases exist in the CBIR community, from the smallest containing as few as 1000 for the subset of the COREL dataset used in [Lai & Chen 2011] to the biggest containing millions of images such as IMAGENET [Russakovsky *et al.* 2015]. The proposed system is dedicated to image classification, and is designed to solve some of the issues encountered when dealing with fine grained classification in particular. To get a database including fine grained classification challenges, it has been decided to work on Caltech-256 [Griffin *et al.* 2007]. This is a medium sized database containing 30,607 images divided into 256 categories, some of which are represented in Figure 4.1. It has been used by [Perronnin *et al.* 2010] to experiment several image descriptions. In their comparison, the training set for the SVM is built by randomly selecting a subset of images from the dataset. Fine grained classification challenges are present in this database, such as the differentiation of gorillas, bears and chimp which are hard to differentiate as shown in Figure 4.2, or the identification and gathering of all mussels images in a same class while those are visually different from one another as shown in Figure 4.3. Those characteristics makes Caltech-256 a good starting point to assess the performances of the proposed system. Bigger and more challenging databases may be considered in the future if the system performs well.

Figure 4.1: Some categories from Caltech-256 database



As for the image description, authors of [Perronnin *et al.* 2010] kindly provided us with the improved fisher kernel vector they used. The image descriptions are 2048 dimensional vectors based on SIFT descriptors densely extracted on each image. Image description is not the focus of this thesis, and thus the computation of those descriptions is not detailed here. An introduction to image descriptions is provided in chapter 1. Interested reader may refer to [Perronnin *et al.* 2010] for an in depth explanation of the image descriptions used in the experiments. Using those image descriptions ensures us some quality on the image descriptions. It also allows us to compare our results directly to those obtained by [Perronnin *et al.* 2010] provided that we are using the same measures and experimental process to assess the quality of the classification. Those are detailed right below, starting with the measure used to assess the classification performances before introducing the experimental process.

Figure 4.2: Inter-class ambiguities for categories: BEAR, GORILLA and CHIMP

Figure 4.3: Intra-class ambiguities for the category: MUSSELS

4.1.2 Classification Performances Measurement

The most extensive way to assess the performances of a classifier is to compute, for each category C_i of the database, the number of images from that category that have been classified in each category C_j by the classifier. This information is usually stored into a matrix M called the confusion matrix. This is a square matrix containing both as many rows and as many columns as classes in the database. A cell $M_{i,j}$ contains the number of images from the i^{th} category that have been classified in the j^{th} category. Though providing extensive information about the classification, this matrix is large (256 rows by 256 columns for Caltech-256) and thus hard to display and read.

Several metrics computed from the confusion matrix exist and are usually used. They give a good insight of a classifier's performances and are much more easy to read and understand. Among them, the most used measures are the precision, the recall and the accuracy. To get results comparable to those of [Perronnin *et al.* 2010], the same measure must be used. The mean classification accuracy is the only measure used in [Perronnin *et al.* 2010]. Therefore the same is done in our experiments.

The average classification accuracy computation is given by Equation 4.1. It is the ratio of the sum of correctly classified images by the total number of images, which is also the ratio of the sum of the diagonal cells of the confusion matrix by the sum of all the cells of the confusion matrix. It is a single figure which represents the portion of images that have been classified correctly. It is fairly easy to understand: the bigger the value, the more images have been correctly classified, the better the system.

$$\text{Average Accuracy} = \frac{\text{Number of correctly classified images}}{\text{Number of images in the database}} = \frac{\sum_i M_{i,i}}{\sum_{i,j} M_{i,j}} \quad (4.1)$$

This performance measure is used to evaluate the classifier learned by the SVM. For the results of our experiments to be comparable to those of [Perronnin *et al.* 2010], the training set of the SVM must be built respecting some constraints. In addition, the proposed hybrid system includes the use of a stochastic algorithm: NSGA-II. Therefore, evaluating its performances on one execution only is not fair. To address this, an experimental process has been defined. It is explained in the next subsection.

4.1.3 Experimental Process

The experimental process followed in the experiments conducted here must be similar to the one used in [Perronnin *et al.* 2010] so that our results can be compared to their results. Several points of the experimental process are detailed in the following paragraphs, explaining the approach chosen to be comparable to [Perronnin *et al.* 2010]. The subsection ends with some general parameters of the experimental process such as the number of runs over which the performances are averaged.

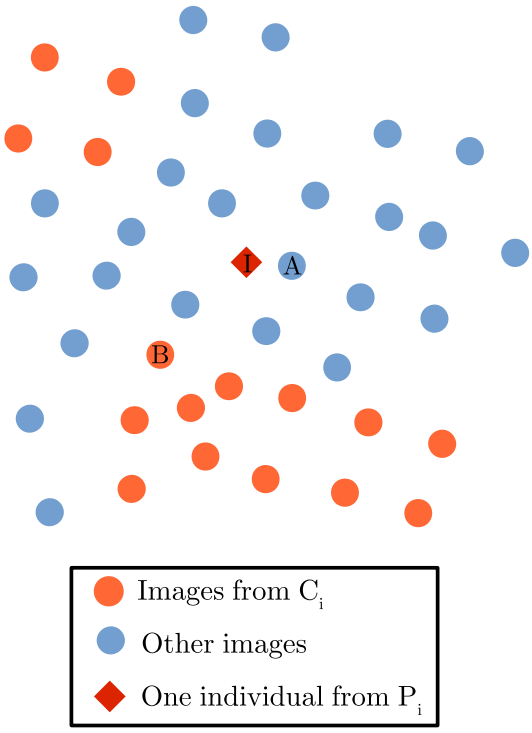
The Set of Available Training Samples The learning process consists in building a training set for the SVM. To do so, a set of available training samples is used, for which the category is known and from which training samples are drawn. This set must be separated from the testing set, which serves only for the evaluation of the system's performances. Most databases in the CBIR community are separated into two distinct sets.No such separation exists for Caltech-256. In their experiments, [Perronnin *et al.* 2010] consider that the whole database is the set of available training samples. Once the training is done, every image that is not part of the SVM training set is considered to be a testing image. The same scheme is adopted for the presented experiments to get experimental conditions as close as possible from theirs, so that comparisons are more meaningful. Thus, any image from the database may be added to the SVM training set, and when evaluating the classification performances of the SVM, only the images not in the training set are evaluated.

Number of Images in the Training Set To compare classification results with those of [Perronnin *et al.* 2010], the classifier must be learned on the same amount of data. Indeed, comparing the results of a classifier that has learned with ten times more information than another makes no sense. This means that the training sets must contain the same number of images. [Perronnin *et al.* 2010] builds the training set randomly, using 4 different sizes of training sets. The training sets are composed of 15, 30, 45, and 60 images per category respectively. The same training set sizes are used in the conducted experiments.

Portion of Each Category in the Training Set The training sets built by [Perronnin *et al.* 2010] contain exactly the same number of images from each category. It has been decided to stick to this structure when evaluating the proposed hybrid system. To ensure that the same number of images from each category is added to the training set in the proposed implementation, the matching process has been altered. In the proposed implementation, the GA's population is composed of as much independent subpopulations as categories of images in the database. The matching process is then modified such that individuals from population P_i can only be matched to images from the i^{th} category: C_i. The same number of images is then selected from each matched subpopulation. Although this allows to get the same number of images from each category in the training set, it interferes with the GA convergence. Figure 4.4 shows an individual I from P_i represented by a red diamond. The image from the dataset the closest to I is marked A and the image from C_i the closest to I is marked B. The GA has converged to I. However I cannot be used as a training sample because it is not representing an image from the dataset. Using A as a training sample is an alteration of the genome of the individual to which the GA has converged, but it is minimal and needed because only images from the dataset can be used as training samples. With the modified version of the matching, B is used instead of A, making the alteration to the genome bigger. Doing so, part of the work performed by the GA when converging to I is ignored, potentially degrading the final performances.

To get the 15, 30, 45 and 60 images per category steps, x, the number of images added to the training set during each relevance feedback process, is set to $15 * B$, B being the number of categories in the database, which is equal to 256 for caltech-256. There are B subpopulations. Individuals from P_i are all matched to C_i. The same number of images is extracted from each matched subpopulation. This ensures that 15 images from each category are added to the training set each time the relevance feedback process is called. Thus, after the first call to the relevance feedback the training set contains 15 images per category; after the second call it contains 30 images per category and so on until the 60 images per category are reached after the 4^{th} call to the relevance feedback process. The average classification accuracy of the SVM learned from each training set can then be computed on the images from the database that are not part of the training set. This gives the final performance measure which can be compared to the results obtained in [Perronnin *et al.* 2010]

Figure 4.4: Neighboring images of an individual. In red an individual from P_i, A is the closest neighboring image from the dataset and B the closest neighboring image from C_i.



safely.

Number of Runs To lesser the influence of the randomness on the final results, and ensures that the observed performances are not due to particularly good or bad random choices, the experiments are run 10 times. The average value of the average classification accuracy obtained over the 10 runs is presented as the final result.

4.2 Compared Algorithms

Three learning schemes are tested and compared in this performances analysis: a random selection scheme proposed by [Perronnin *et al.* 2010], an active learning scheme adapted from the proposition of [Tong & Chang 2001], and the implementation of the proposed system. The random selection doesn't put any effort in the way images are selected and serves as a baseline in this study. The active learning scheme iteratively builds the training set by evaluating each image from the available training samples before choosing which one are added to the training set. The proposed approach uses an EA instead of evaluating each image to find the bes training samples. In addition, active learning scheem dedicated to SVMs often concentrate on a specific part of the search space where the proposed method try

to bring diversity.

This section aims at presenting and giving the value of the parameters for the two methods that have not been detailed yet: the random selection and the active learning schemes. All the details about the proposed framework and the implementation used for the performance analysis are provided in the previous chapter, with a review of its parameters in Table 3.1.

The parameters of the random selection scheme issued from [Perronnin *et al.* 2010] are presented first. Other methods' parameters are set so that the differences in the results observed are due mainly to the learning scheme, and not to a change in the SVM strategy or kernel used for example.

4.2.1 Random Selection Scheme

The results obtained by [Perronnin *et al.* 2010] are used for this scheme. As the name self-explains it, images are randomnly drawn to form the SVM training set. The SVM is learned using the resulting training set and the average accuracy of the resulting classifier is computing on the images not used in the training set. The process is repeated 5 times and average results are presented to smooth the influence of random selections and ensures that the results are not due to particularly good or bad random choices.

The SVM used in the experiments is a linear SVM. The regularization parameter C is fixed by learning $n_{train} - 5$ images and validating with the5 remaining images from the training set. n_{train} represents the number of images in the training set, i.e. $15 * 256$, $30 * 256$, $45 * 256$ or $60 * 256$ in this study on Caltech-256 with the chosen number of images per category in the training set.

To ensure that performances variations are due to the way training samples are chosen and not to the classifier chosen, other methods should stick to SVM configuration and number of training set used defined here.

4.2.2 Active Learning Scheme

The active learning scheme is adapted from the work of [Tong & Chang 2001]. In this work, the authors state that for a binary SVM, the image improving the decision the most when added to the training set is the one the closest to the SVM's decision. Then to build the SVM training set, the distance of each image from the set of availble training samples to the SVM decision is computed, and the image presenting the smallest distance is added to the training set. The SVM is then retrained with this new training sample, and the process continues until enough training samples are added to the training set.

The context here is the multi-class classification though. In addition, the experimental process states that the classifiers are tested with training set composed of 15, 30, 45, and 60 images extracted from each category. The random selection scheme and the proposed method are selecting images by bunch of 15 in each category; so the active learning scheme is developed to do the same to respect this process.

To respect those constraints, the active learning scheme proposed by [Tong & Chang 2001] has been adapted in this study. A one versus all SVM strategy is used to address the multi-class classification. This leads to the creation of one binary SVM per category to be identified in the database. For each binary SVM SVM_i, the 15 images the closest to its decision from category C_i are added to the training set. Doing so, 15 images from each category are added to the training set, and the property of being the closest to the SVM training set proposed by [Tong & Chang 2001] is respected. Improving the decision of each binary SVM should improve the overall decision, as for the implementation of the proposed system. In addition, this system suffers from the same constraints as the developed system, i.e. training samples selected for the improvement of SVM_i are drawn from C_i only. Using the same constraint ensures that one system is not favored and thus the results are not biased. The first training set formed of 15 images per category is build by selecting images in the vicinity of the queries provided for each category.

For the parameters: to stick with the parameters of the random selection scheme a linear kernel is used. The regularization parameter C however is not tuned. It is set to $C = 1$. There are no more parameters, so this ends the presentation of this active learning technique inspired by [Tong & Chang 2001].

All the information needed to run the experiments are now available. The next section reports the classification performances obtained in the experiments and analyses those results, discussing the performances observed for the proposed system against two other systems. The computation complexity of each system is also briefly studied and a comparison between the complexity of each method is provided.

4.3 Performances Analysis

In this section, the performances of the three techniques mentioned in the last section are compared. The first part of this section presents and compares the classification results obtained by each of those methods following the experimental process described in section 4.1 and using the parameters presented in the previous section. A second part then compares the computational complexity of each of those methods.

4.3.1 Classification Performance Analysis

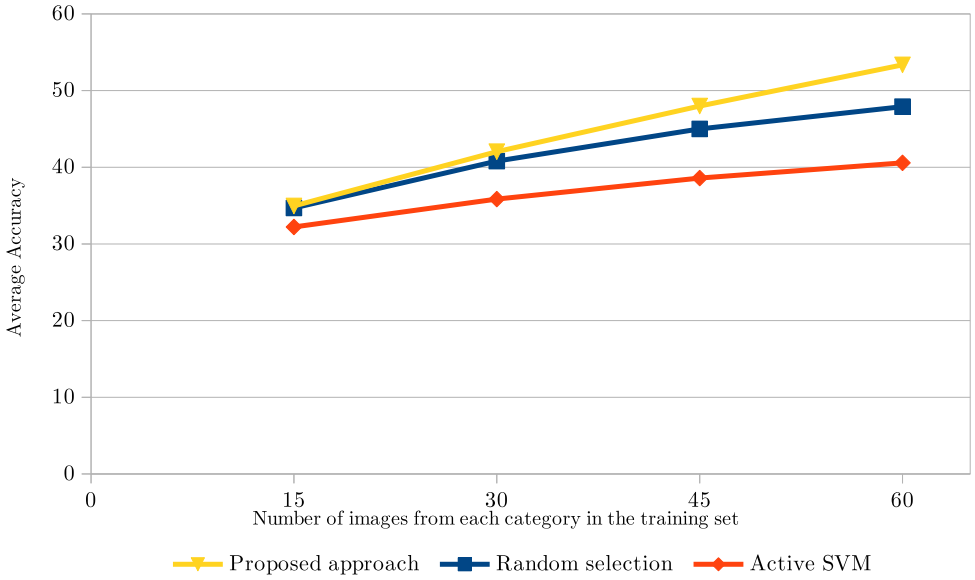
In order to evaluate the classifier learned using the system proposed in chapter 3, its implementation is evaluated against two other learning strategies: an active learning techniques based on [Tong & Chang 2001] presented in the previous section; and the random selection scheme used in [Perronnin *et al.* 2010].

The average accuracy obtained by each method is reported in table 4.1 and also graphically represented in Figure 4.5. The plot represents the number of images from each category in the training set on the x-axis and the average accuracy on the y-axis.

Table 4.1: Average accuracy of the proposed approach, a random selection scheme, and an active learning schemes on Caltech-256

Method	Number of images per category in the training set			
	15	30	45	60
Random selection from [Perronnin *et al.* 2010]	34.70	40.80	45.00	47.90
Active SVM based on [Tong & Chang 2001]	*32.22*	*35.85*	*38.59*	*40.58*
Proposed approach	**34.96**	**42.03**	**47.99**	**53.35**

Figure 4.5: Average Accuracy of the Different Methods with Respect to the Size of the Training Set



The three methods have similar results when using a training set containing 15 images per category. This is easily explained for the proposed approach, as the initialization is done using a random process. The process is biased toward the query image used for each category. However, the matching process, which matches individuals to the closest image using an approximate nearest neighbor search, alters the individuals before they are added to the training set. And the selection after the matching process uses a uniform random selection. Thus, even though a bit biased, the first training set is somehow randomly generation. This explains the results close to those obtained using a random selection scheme. The active SVM scheme is a bit behind in this initialization process performances. The loss is not that big and may be attributed to the fact that the initialization is done solely in the vicinity of the query images from each category. Indeed, as shown in section 3.1.2.2, using a lot of images close to each other does not improve the decision of a SVM that much when compared to using just a few images from the same vicinity. This can explain the lower performance for the active learning scheme at initialization.

When adding images to the training set, the proposed system takes the lead,

outperforming both other methods. The improvement is small at the beginning, the average accuracy of the proposed system is 2 points above the random selection's score when using 30 images per category in the training set. At this moment, the GA has run for 30 generations and the relevance feedback process has been executed once. This lead grows to attain a 6 points lead over the random selection scheme at the end of the run, with 60 images per category in the training set. The GA has then run for $3 * 30 = 90$ generations and the relevance feedback process has been executed 3 times, initialization not included. This shows that selecting the learning samples through the GA brings a substantial improvement in the average accuracy of the classification over a random selection, and thus that the proposed system's training sample selection is better than a random selection to address the classification task under study.

As for the active learning scheme, its performances are bad, performing even worse than the random selection. One of the flaws comes from the initialization process, which makes the system start at a lower performance. But the performance gap grows with the number of images added to the training set so the initialization can't be the only issue. The adaptation of the process proposed by [Tong & Chang 2001] may not be adapted, explaining the poor results obtained. [Tong & Chang 2001] is adding only one image at a time, retraining the SVM before adding another image to the training set. Here 15 images are added at the same time for each SVM forming the *one versus all* strategy. This is probably not the best thing to do, even though those are chosen as the closest to the SVM's decision. Indeed, even if it seems like a natural extension to the scheme proposed by [Tong & Chang 2001], nothing proves that adding more than the first closest image to the decision is the best thing to do. In addition, a multi-class classification problem is studied here, using several binary SVMs to solve it whereas the original active learning scheme was thought for only one binary classifier. Choosing the images the closest to each SVM's decision is done here to be as close as possible to the original active learning strategy, expecting that improving each of the binary classifier would improve the overall classification result. But this might again not be the best thing to do for multi-class classification active learning. Though the proposed adaptations to build the active learning scheme may not be adapted and lead to those poor results, it shows that selecting the training samples the closest to the SVM's decision, and not exploring the search space, does not lead to good results in real classification problems. And in particular, it is not adapted in the case of fine grained classification problems, as proposed in the Caltech-256 database. Random selection performs even better than such a strategy as shown by the average classification scores obtained in this study.

This experiment on Caltech-256, which is a medium-sized database presenting some fine grained classification challenges, shows that the proposed combination of a GA to select the training samples and a SVM to perform the classification presents interesting performances. In particular, this system performs better than random selection, showing its relevance, and better than an active learning scheme. It also shows that selecting images the closest to the SVM's decision is not necessarily the best solution, and sometimes exploration is needed to enhance the performances.

This encourages to investigate the proposed hybrid system more to find its limits. Performances are not the only important element though, and before going through more advanced tasks or bigger databases, an investigation of the computational complexity of such a method is needed to evaluate its potential to scale to larger databases. The next part of this section evaluates the complexity of the proposed system and compares it to the active learning scheme's complexity and random selection's complexity.

4.3.2 Computation Complexity of the Proposed System

The three methods compared here to perform classification in CBIR have different computation complexity. The gain obtained in the classification performance may not be worth the computation overhead needed to obtain it sometimes. And depending on the computation complexity, some methods may not be adapted for very large databases. Therefore, the computation complexity of each of the three compared methods is presented and compared.

Two phases are distinguished here when talking about the computation complexity of the methods. The first phase is a pre-processing phase, it consists in everything that is performed before the learning process starts. The second phase is the learning phase, which consists in everything that is done to learn the classifier.

4.3.2.1 Pre-processing Computation Complexity

The pre-processing phase consists mainly in the computation of the image description, which is the same for each of the methods used and so does not include any difference in the resource investment put in each method. This pre-processing step is even common to any CBIR system and its complexity depends on the image description built. Because of this, it is not a discriminant factor when choosing a method over another.

The random selection and the active learning scheme do not have any other pre-processing step. The proposed system however includes the computation of the LSH structure in the pre-processing phase. LSH pre-processing consists in hashing the set of available training samples into the LSH structure. Building the LSH structure only consists in computing the hash of each image in each table. The computation complexity of building the LSH structure is then given by Equation 4.2, with L the number of tables in the structure, k the number of hash functions per table and n_{ATS} the number of images in the available training samples set. Computing the value of a hash function is pretty simple and is considered as a unit operation. Computing the image description for each image in the database takes a lot more time than hashing the resulting vector however, and the hashing process can be done right after the image description extraction within the same loop as it does not need access to other image descriptions to compute the hash of one image description. So the computation overhead induced by the construction of the LSH structure is not that important in the end.

$$\mathcal{O}(n_{ATS} \cdot L \cdot k) \tag{4.2}$$

4.3.2.2 Learning Computation Complexity

The computation complexity of the learning phase of each algorithm is quite different. Therefore, the computation complexity of each algorithm is detailed separately before comparing them. But before doing any computation complexity analysis, a set of notation is defined. In the following: let n_{TS} be the number of learning samples used to train the classifier; let n_{ATS} be the number of images in the set of available training samples; let B be the number of category in the database, and let consider that images are equally spread among categories in the database; let SVM_{eval} be the computation complexity to process the evaluation of one vector by a SVM; and let $SVM_{train}(x)$ be the time needed to train a SVM with a training set containing x elements. It must be noted that $SVM_{train}(x) = \mathcal{O}(x^3)$ and SVM_{eval} time is negligible with respect to $SVM_{train}(x)$.

The computation complexity of the learning part of the three compared algorithm is detailed below, starting with the random selection scheme.

Random Selection Learning Process Computation Complexity The random selection is fairly simple, it requires n_{TS} random selections, and then learning the SVM from the selected images. A random selection can be considered as a unit operation; then selecting n_{TS} images is done in $\mathcal{O}(n_{TS})$. The complexity of learning the SVM with n_{TS} learning samples is to be added to this, leading to the computation complexity given in Equation 4.3 which is only the complexity of learning the SVM. The SVM is a *one versus all* strategy multi-class classification SVM. Thus it is formed of B binary SVMs, explaining the B factor in the complexity. The computation complexity of the active learning scheme is detailed next.

$$\mathcal{O}(n_{TS} + B \times SVM_{train}(n_{TS})) = \mathcal{O}(n_{TS} + B \times n_{TS}^3) = \mathcal{O}(B \times n_{TS}^3) \tag{4.3}$$

Active Learning Computation Complexity The active learning process considered adds images to the training set by batch of n_{add} images. For each category C_i, the $\frac{n_{add}}{B}$ closest images to SVM_i's decision are added to the training set. Identifying the closest images to SVM_i's decision requires the evaluation of each image in the set of available training samples by the SVM and their ranking. And this must be done for each category. In the end, each image from the set of available training samples is evaluated once, which leads to a computation complexity of $n_{ATS}SVM_{eval}$.

Then, the ranking is done for each category separately. As images are considered to be equally spread among the categories, each category is considered to have $\frac{n_{ATS}}{B}$ images. Sorting this number of elements has a complexity of $\mathcal{O}\left(\frac{n_{ATS}}{B} \log \frac{n_{ATS}}{B}\right)$. It is done for each of the B categories, leading to a computation complexity for the

ranking of $\mathcal{O}\left(B\frac{n_{ATS}}{B}\log\left(\frac{n_{ATS}}{B}\right)\right)$. Then selecting the first n_{add} images from each ranking has no cost.

Once images have been selected and added to the training set, the SVM is trained; and everything is repeated until n_{TS} images have been added to the training set. The complexity of learning the SVM depends on the number of iteration, let i be the iteration number. The SVM is a *one versus all* multi-class SVM, so it is composed of B binary SVMs. As the SVM is learned with $i \times n_{add}$ images, its learning computation complexity is $\mathcal{O}(B \times (i \times n_{add})^3)$.

The process is repeated $\frac{n_{TS}}{n_{add}}$ times to get enough training samples in the training set. The overall complexity is then given by Equation 4.4.

$$\mathcal{O}\left(\frac{n_{TS}}{n_{add}} \times n_{ATS} \times \left(SVM_{eval} + \log\left(\frac{n_{ATS}}{B}\right)\right) + B \times \sum_{i=1}^{\frac{n_{TS}}{n_{add}}} SVM_{train}(i \times n_{add})\right) \tag{4.4}$$

The simplification is done under the assumption that $\frac{n_{TS}}{n_{add}}$ is not too big, and therefore learning the SVM with the most training samples prevails in the sum. If there are a lot of iterations, then the sum becomes much more difficult to simplify. The SVM evaluation complexity is assumed to be negligible with respect to $\log\left(\frac{n_{ATS}}{B}\right)$ (which is most of the time true as it only consists in computing a kernel function which is pretty simple). Finally, the term in n_{ATS} is kept in addition to the n_{TS}^3 term because in the CBIR context, n_{ATS} is usually way much bigger than n_{TS}, making it important in the complexity analysis. This lead to the simplified form presented in Equation 4.5

$$\mathcal{O}\left(\frac{n_{TS}}{n_{add}} \times n_{ATS} \times \log\left(\frac{n_{ATS}}{B}\right) + n_{TS}^3\right) \tag{4.5}$$

Computation Complexity of the Proposed Hybrid Approach This hybrid system iteratively builds the training set for a SVM by extracting interesting training sample from a GA's population. The population of the GA is extracted after g generations. Each individual in it is matched to an image and then n_{add} images are selected to be added to the SVM's training set. The process is repeated until enough images have been added to the training set.

First, the complexity of running g generations of the GA is computed. The GA is made of B independent subpopulations of n_{ind} individuals. Each step in the evolution loop is pretty simple and has little to no impact on the complexity of the GA. The overall complexity of the evolution loop is then defined by the evaluation process which uses the non-dominated sort procedure of NSGA-II. It has a complexity of $\mathcal{O}\left(m \times n_{ind}^2\right)$ as given by [Deb *et al.* 2000], where m is the number of fitness functions or objectives. In the proposed implementation $m = 2$, and in general when using NSGA-II m is kept at a value less than 6. The evolution loop is run g times for each of the B subpopulations before the relevance feedback

process is run. This leads to a complexity of $\mathcal{O}\left(B \times g \times m \times n_{ind}^2\right)$ for running the g generations.

Then, the relevance feedback process is executed, first matching individuals to images using LSH. Each query to the LSH has a complexity of $\mathcal{O}(n_{ATS}^{\frac{1}{c}})$, c being a parameter of the LSH as defined in Section 3.2.4.2 respecting $c > 1$. Several images may be returned by an LSH query, but it assumed that the LSH parameters have been tuned during the pre-processing so that the number of images returned is low enough. Thus ranking them does not impact the complexity much. Each individual of each subpopulation is matched to an image, leading to an overall complexity for this step of $\mathcal{O}(B \times n_{ind} \times n_{ATS}^{\frac{1}{c}})$. Then, n_{add} images must be selected through the mapped subpopulations to be added to the SVM's training set. As this is done randomly, it has no impact on the complexity. The SVM is then learned with its new training set, and the process starts over until enough the training set contains enough training samples; i.e. $\frac{n_{TS}}{n_{add}}$ times.

The overall complexity of the system is then given by Equation 4.6 with $n_{pop} = B \times n_{ind}$. As for the active learning scheme, $\frac{n_{TS}}{n_{add}}$ is assumed to be small enough so that the complexity of learning the final SVM summarizes the complexity of learning all the SVMs successively. The following paragraph is dedicated to the comparison of those computation complexities.

$$\mathcal{O}\left(\frac{n_{TS}}{n_{add}} \times \left(B \times g \times m \times n_{ind}^2 + B \times n_{ind} \times n_{ATS}^{\frac{1}{c}}\right) + \left(B \times \sum_{i=1}^{\frac{n_{TS}}{n_{add}}} SVM_{train}(i \times n_{add})\right)\right)$$
$$= \mathcal{O}\left(\left(g \times m \times n_{pop}^2\right) + \left(n_{pop} \times n_{ATS}^{\frac{1}{c}}\right) + B \times n_{TS}^3\right) \quad (4.6)$$

Computation Complexity Analysis The random selection scheme is obviously the least costly scheme. It only costs the learning of the SVM.

The active learning scheme includes the cost of learning the SVM in its complexity, but it also adds terms related to evaluating and ranking the images in the set of available training set. In particular, the active learning scheme complexity linearly depends on the size of the set of available training set. Even though, as shown in subsection 4.3.1, this system is not performing well and is probably not adapted to the problem studied, any active learning system would need to scan the set of available training samples to pick the most interesting ones, therefore at least linearly depending on the size of the training set. This means that the complexity of such a learning process linearly grows with the size of the database considered, which can become a problem when dealing with huge datasets.

The proposed hybrid system's complexity depends on the size of the set of available training set with the term $n_{ATS}^{\frac{1}{c}}$ only, which is sub-linear as $c > 1$. This makes this method easier to scale with the size of the database treated. This is not the only thing the complexity depends on however. It is also tightly coupled to the

parameters of the GA and the other parameters of the SVM. The parameters of the LSH can be tuned at pre-processing time and those of the GA can be tuned before the execution of the learning process. Therefore, the major part of the complexity of this system can be adapted to the need, choosing to put more or less effort in finding the most interesting training sample. This makes this system very interesting to treat bigger dataset.

In conclusion, the implementation of the system proposed in chapter 3 showed better classification results than the random selection and the active learning schemes it was compared too. And even though it presents some computation overhead when compared to random selection or active learning schemes, the computation complexity of its learning process depends only sub-linearly with the size of the available training sample set. It depends mainly on the size of the GA population, making it easy to scale to address larger databases, and easy to tune to respect resource constraints if needed. This makes this system interesting to investigate further and improve. Some ideas for improvement or further testing are presented in the next chapter.

Conclusion and Perspectives

The amount of visual content available on the Internet is ever growing. Therefore, there is an immediate need of techniques adapted to search through this content. Content Based Image Retrieval techniques were developed to address this need, using visual descriptions only to classify the data. A CBIR system is usually composed of two separate steps: extracting a visual description from the images, and then using a data mining algorithm to search through the images. The emphasis in this thesis is on the later, and in particular image classification.

Support Vector Machine are among the best techniques for image classification. However, their learning process has a complexity of n_{TS}^3, n_{TS} being the number of training samples. Therefore, they cannot be learned using all the training material available on the big databases encountered today. A subset of all the images available for training must be selected to form the actual training set. Random selection is the easiest solution, but better alternatives may exist. Active learning techniques have been developed with quite success for example. Their goal is to iteratively build a training set by selecting the best image to add to the training set from the set of available training samples at each iteration. But with the size of the databases growing, new challenges arose.

This thesis focuses on one of those new challenges: the fine grained classification. It consists in being able to separate accurately images from classes presenting inter-classes visual similarities and intra-class visual dissimilarities. This problem turns out to be multi-modal by nature. Indeed, visual descriptions, such as the the SIFT descriptors which form the base for the image description used, are built to keep visually similar images close and visually dissimilar images far from each other. Therefore, intra-class visual dissimilarities lead to images from the same class to be scattered through the search space. And inter-class similarities lead images from different classes to be mixed in one place of the search space. Thus they form small groups of images from the same class separated by groups of images from other classes. Those groups of images belonging to the same class but being scattered through the search space are modes for this class. To solve this kind of problem, a mix of global and local search is needed. Or, the existing active learning techniques are lacking of global search to deal with such problems.

Evolutionary Algorithms are presenting interesting exploration capabilities and were thus considered to tackle the fine grained classification problem. But image descriptions are most of the time of high dimension, more than 1000 dimensional vectors. And it turned out that the performances of EAs dedicated to multi-modal problems are not that good when dealing with high dimensional problems.

This is why hybrid systems have been considered. The idea is to combine EAs and SVMs to take advantage of the strengths of both of them. A matching technique is needed however to link those two algorithms. Indeed, the genome of the individuals generated by the EA are not necessarily corresponding to the repre-

sentation of an image from the database, and the SVM can learn only from real images. Then this matching process is needed, and has been done using LSH in the proposed implementation. The system proposed in this thesis uses an EA to iteratively feed a SVM with training samples. The SVM then learns a classifier from the iteratively built training set. This hybrid system can be used with any image representation, and is expected to improve the results when compared with other techniques using the same representation. The performance analysis performed on Caltech-256 shows that the proposed approach outperforms random selection and an active learning scheme. Getting such results on this database emphasizes the potential of the method. In addition, the computation complexity of this system essentially depends on the parameter of the GA and the LSH. It is hardly linked to the size of the database, making it a good candidate to address bigger databases. Those encouraging results and properties opens up to further investigations about this system, which are presented thereafter.

Future Works

The study conducted on the implementation of the proposed study is a demonstration that the system is interesting to consider. Some complementary experiments are proposed in the first subsection to get a better insight of the influence of the parameters and the performances of the system on bigger databases.

In addition, even though already presenting good results, some improvements to the framework can still be done. Indeed, the proposed system still has some drawbacks, as requiring to know the number of classes a priori and giving a query image for each category at the beginning of the learning process for example. Some propositions of improvements addressing some of the flaws of the system are proposed in the second subsection.

Complementary Experiments

Several complementary experiments can be done on the proposed system to evaluate its performances in different situations and identify its strengths and weaknesses. Some of them, that we think are the most important, are given here.

Influence of the Framework's Parameters First, the influence of the different parameters of the framework could be studied. In particular, how the performances are affected when modifying g the number of generations between each relevance feedback call, and x the number of images added to the training set each time the relevance feedback process is called. Those parameters are inherent to the framework, and knowing how they affect the performances is important to understand the framework.

Implementation Choices It must be noted that only one implementation of the system has been proposed. It uses NSGA-II, LSH and a linear SVM with a one

versus all strategy to address the multi-class classification. Those algorithms, and their parametrization (which is different from g and x which are parameters of the framework) could be changed too. Other hash functions could be considered, the number of tables L and hash per table k could be fine tuned to the problem. DE or CMA-ES could be used as they are more adapted to real valued problems than GAs in general. However, a variation dedicated to multi-objective problems would be needed if the same fitness functions are kept. The different genetic operators of the GA could be changed or the influence of their parameters on the system studied. The regularization parameter C of the SVM could also be fine tuned, and perhaps a kernel different than the linear kernel could bring better results. All those implementation choices could be contested and different implementations can be tested into a comparative study so as to find the best set of algorithms.

Bigger Databases and Other Image Representation This system can be used with any image representation, then using it with newer and better image representation should improve the performances. In addition, the system presents interesting scalability potential, but it is important to see if the classification performances are still good when tackling bigger databases. When dealing with bigger databases, g and x would need to be set wisely so that the system learning process is not too long but still explores the search space enough. It is important to know how the system performs in such situations and if it is really adapted. Indeed, if g or the size of the GA population have to be set at a too high value, the time needed to run the system will become too high for it to be worth.

Objectives of the Evolutionary Algorithm A couple of objective functions have been proposed for the evaluation process of the EA. The goals being to explore the search space and retrieve interesting learning samples for the SVM. While the functions used have lead to the interesting results provided in chapter 4, it is worth trying other fitness functions that may identify better training samples. Using a fitness function derived from active learning schemes dedicated to SVMs in combination with a fitness function dedicated to exploration could be a good lead for example. Studies about active learning techniques dedicated to multi-class classification SVMs have to be done before however.

Those are just a few of the numerous experiments that could be conducted to provide a deeper analysis of the system as it stands. But doing those would already provide a good overview of this system's capabilities. In addition to this deep study of the system, some drawbacks of the framework have already been identified and could be fixed. The next subsection stresses out some of those together with some solution to investigate.

Framework Improvements

Several drawbacks have been identified in the proposed system, and fixing them is one of the first thing to do to improve the system.

First, as it is right now, the system adds the same number of images from each category to the training set. This is done to respect the experimental process used in [Perronnin *et al.* 2010]. However, doing may not be optimal. Indeed, the number of images in each category is not necessarily the same and categories are not equally easy (or hard) to identify. Therefore using the same number of images from each category to build the training set is not necessarily a good idea. Another problem of this selection strategy comes from the adaptation made to the implementation to respect this constraint. The population of the GA is separated into several small subpopulations, which is an implementation choice. But then, individuals of each subpopulation P_i are matched to images belonging to a single category C_i. This has been done only to ensure the same number of images are selected from each category to be added to the training set. The closest image from an individual in C_i is probably further away than the closest image from the same individual in the whole database. So this matching strategy hinders the work done by altering the genome of the individuals more than needed during the matching process. In addition, images from each category are hashed in different LSH structures to respect the constraint too, which may not be possible as the categories may not be known a priori. To cope with that, the constraint of adding the same number of images for each category can be released. Doing so, the matching process won't be constraint anymore and a single LSH structure can be computed for the whole database.

Another drawback of the system is the need of one query image per category to initialize the search. The categories may not be known a priori, in particular when dealing with interactive systems. In such a situation, providing one query image for each category is not relevant. The system could start learning with any number of images provided. Those are used as a first training set for the SVM and must represent at least two categories. Then, during the learning process, any new category can be introduced by labeling one image with a new label during the gathering of the relevance feedback. As the SVM is retrained after each addition of elements to the training set, it will take the new category into consideration immediately and for the rest of the learning process. If the GA uses independent subpopulations or fitnesses for each category, a new subpopulation can be dynamically initialized at the moment the first image from the new category is added. Doing so makes the system more dynamic and allows to treat problems where the number of categories is not known a priori.

The framework may suffer from other drawbacks that have not been identified yet, but will most likely emerged while investigating it more.

This ends this document I wish you enjoyed reading, and leaves plenty of room for following studies on the framework introduced here. We are already investigating about the performances of this framework on bigger databases using deep learning features which we wish will lead to interesting results.

Bibliography

[A. Foulonneau *et al.* 2009] A. Foulonneau, P. Charbonnier and F. Heitz. *Multi-reference shape priors for active contours.* Int. J. Comput. Vision, vol. 81, no. 1, pages 68–81, 2009. (Cited on pages xi and 10.)

[Arevalillo-Herráez *et al.* 2013] Miguel Arevalillo-Herráez, Francesc J. Ferri and Salvador Moreno-Picot. *A hybrid multi-objective optimization algorithm for content based image retrieval.* Applied Soft Computing, vol. 13, no. 11, pages 4358 – 4369, 2013. (Cited on pages xix, 73 and 74.)

[Auger *et al.* 2009] Anne Auger, Johannes Bader, Dimo Brockhoff and Eckart Zitzler. *Theory of the Hypervolume Indicator: Optimal μ-distributions and the Choice of the Reference Point.* In Proceedings of the Tenth ACM SIGEVO Workshop on Foundations of Genetic Algorithms, FOGA '09, pages 87–102, New York, NY, USA, 2009. ACM. (Cited on pages xviii and 52.)

[Balcázar *et al.* 2001] Jose Balcázar, Yang Dai and Osamu Watanabe. *A Random Sampling Technique for Training Support Vector Machines.* In Naoki Abe, Roni Khardon and Thomas Zeugmann, editors, Algorithmic Learning Theory, volume 2225 of *Lecture Notes in Computer Science*, pages 119–134. Springer Berlin Heidelberg, 2001. (Cited on pages 7 and 29.)

[Bandaru & Deb 2013] S. Bandaru and K. Deb. *A parameterless-niching-assisted bi-objective approach to multimodal optimization.* In Evolutionary Computation (CEC), 2013 IEEE Congress on, pages 95–102, June 2013. (Cited on pages xviii, 58, 66 and 67.)

[Basak *et al.* 2013] A. Basak, S. Das and K.C. Tan. *Multimodal Optimization Using a Biobjective Differential Evolution Algorithm Enhanced With Mean Distance-Based Selection.* Evolutionary Computation, IEEE Transactions on, vol. 17, no. 5, pages 666–685, October 2013. (Cited on pages xviii, 58 and 63.)

[Bay *et al.* 2006] Herbert Bay, Tinne Tuytelaars and Luc Gool. *SURF: Speeded Up Robust Features.* In Aleš Leonardis, Horst Bischof and Axel Pinz, editors, Computer Vision – ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I, pages 404–417. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. (Cited on page 12.)

[Belongie *et al.* 2000] S. Belongie, J. Malik and J. Puzicha. *Shape Context: A new descriptor for shape matching and object recognition.* In NIPS, 2000. (Cited on pages xi and 10.)

[Belongie *et al.* 2001] S. Belongie, J. Malik and J. Puzicha. *Matching Shapes.* In IEEE International Conference on Computer Vision, July 2001. (Cited on pages xi and 10.)

[Belongie *et al.* 2002] S. Belongie, J. Malik and J. Puzicha. *Shape Matching and Object Recognition Using Shape Contexts.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 24, pages 509–521, April 2002. (Cited on pages xi and 10.)

[Bengio *et al.* 2013] Y. Bengio, A. Courville and P. Vincent. *Representation Learning: A Review and New Perspectives.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 8, pages 1798–1828, August 2013. (Cited on pages xii and 16.)

[Boureau *et al.* 2010] Y. \-L. Boureau, F. Bach, Y. LeCun and J. Ponce. *Learning Mid-Level Features For Recognition.* In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), 2010. (Cited on pages xii and 14.)

[Brinker 2003] Klaus Brinker. *Incorporating diversity in active learning with support vector machines.* In In Proceedings of the 20th International Conference on Machine Learning, pages 59–66. AAAI Press, 2003. (Cited on page 27.)

[Broilo & De Natale 2009] M. Broilo and F.G.B. De Natale. *Evolutionary image retrieval.* In Image Processing (ICIP), 2009 16th IEEE International Conference on, pages 1845 –1848, November 2009. (Cited on pages xvi and 47.)

[Carson *et al.* 2004] C. Carson, S. Belongie, H. Greenspan and J. Malik. *Blobworld: Image segmentation using expectation-maximization and its application to image querying.* IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 24, no. 8, pages 1026–1038, 2004. (Cited on page 10.)

[Chang *et al.* 2012] Ray-I Chang, Shu-Yu Lin, Jan-Ming Ho, Chi-Wen Fann and Yu-Chung Wang. *A novel content based image retrieval system using K-means/KNN with feature extraction.* Comput. Sci. Inf. Syst., vol. 9, no. 4, pages 1645–1661, 2012. (Cited on pages xiii and 23.)

[Charikar 2002] M. S. Charikar. *Similarity estimation techniques from rounding algorithms.* In STOC, pages 380–388. ACM, 2002. (Cited on pages xii and 21.)

[Chen *et al.* 2005] Yixin Chen, J.Z. Wang and R. Krovetz. *CLUE: cluster-based retrieval of images by unsupervised learning.* Image Processing, IEEE Transactions on, vol. 14, no. 8, pages 1187–1201, August 2005. (Cited on page 6.)

[Cloude *et al.* 2002] S.R. Cloude, Eric Pottier and W.M. Boerner. *Unsupervised Image Classification Using the Entropy/Alpha/Anisotropy Method in Radar Polarimetry.* In 2002 JPL-AIRSAR Earth Science Apllications Workshop, Pasadena, United States, March 2002. (Cited on page 7.)

[Coolican 2004] H. Coolican. Research Methods and Statistics in Psychology. A Hodder Arnold Publication. Hodder & Stoughton, 2004. (Cited on page 66.)

[Cortes & Vapnik 1995] Corinna Cortes and Vladimir Vapnik. *Support-vector networks.* Machine Learning, vol. 20, no. 3, pages 273–297, 1995. (Cited on pages xiii and 24.)

[Cover 1965] Thomas M Cover. *Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition.* IEEE transactions on electronic computers, no. 3, pages 326–334, 1965. (Cited on page 21.)

[Dalal & Triggs 2005] N. Dalal and B. Triggs. *Histograms of oriented gradients for human detection.* In CVPR, 2005. (Cited on page 9.)

[Datar *et al.* 2004] M. Datar, N. Immorlica, P. Indyk and V.S. Mirrokni. *Locality-sensitive hashing scheme based on p-stable distributions.* SCG, pages 253–262, 2004. (Cited on page 96.)

[Deb *et al.* 2000] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap and T. Meyarivan. *A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II.* In Marc Schoenauer, Kalyanmoy Deb, Günther Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo and Hans-Paul Schwefel, editors, Parallel Problem Solving from Nature PPSN VI: 6th International Conference Paris, France, September 18–20, 2000 Proceedings, pages 849–858. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000. (Cited on pages xviii, 46, 51, 52, 54, 58, 59, 61 and 115.)

[Deselaers *et al.* 2004] Thomas Deselaers, Daniel Keysers and Hermann Ney. *Features for image retrieval: A quantitative comparison.* In Joint Pattern Recognition Symposium, pages 228–236. Springer, 2004. (Cited on pages xii and 13.)

[Dinh *et al.* 2013] Thien Anh Dinh, Tomi Silander, Bolan Su, Tianxia Gong, Boon Chuan Pang, C. C. Tchoyoson Lim, Cheng Kiang Lee, Chew Lim Tan and Tze-Yun Leong. *Unsupervised Medical Image Classification by Combining Case-Based Classifiers.* In MEDINFO 2013 - Proceedings of the 14th World Congress on Medical and Health Informatics, 20-13 August 2013, Copenhagen, Denmark, pages 739–743, 2013. (Cited on page 7.)

[Fauqueur & Boujemaa 2004] J. Fauqueur and N. Boujemaa. *Region-Based Image Retrieval: Fast Coarse Segmentation and Fine Color Description.* Journal of Visual Languages and Computing (JVLC), special issue on Visual Information Systems, vol. 15, pages 69–65, 2004. (Cited on page 10.)

[Fauqueur 2003] J. Fauqueur. *Contributions pour la Recherche d'Images par Composantes Visuelles.* PhD thesis, Université de Versailles, 2003. (Cited on page 19.)

[Fernández-Delgado *et al.* 2014] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro and Dinani Amorim. *Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?* Journal of Machine Learning Research, vol. 15, pages 3133–3181, 2014. (Cited on pages xiii and 24.)

[Fonlupt *et al.* 2011] Cyril Fonlupt, Denis Robilliard and Virginie Marion-Poty. *Linear imperative programming with Differential Evolution.* In Differential Evolution (SDE), 2011 IEEE Symposium on, pages 1–8. IEEE, 2011. (Cited on pages xvi and 40.)

[Gionis *et al.* 1999] A. Gionis, P. Indyk and R. Motwani. *Similarity search in high dimensions via hashing.* In International Conference on VLDB, pages 518–529, 1999. (Cited on page 96.)

[Goldberg 1989] David E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st édition, 1989. (Cited on page 37.)

[Gorisse *et al.* 2010] D. Gorisse, M. Cord and F. Precioso. *Scalable active learning strategy for object category retrieval.* In Image Processing (ICIP), 2010 17th IEEE International Conference on, pages 1013–1016, September 2010. (Cited on pages xiii, 4, 8, 24, 27, 36 and 95.)

[Gorisse 2010] David Gorisse. *Passage à l'échelle des méthodes de recherche sémantique dans les grandes bases d'images.* PhD thesis, Université de Cergy Pontoise, 2010. (Cited on page xxviii.)

[Grauman & Darell 2005] K. Grauman and T. Darell. *The pyramid match kernel : Discriminative classification with sets of image features.* In IEEE International Conference on Computer Vision (ICCV), Beijing, China, October 2005. (Cited on pages xii and 21.)

[Griffin *et al.* 2007] G. Griffin, A. Holub and P. Perona. *Caltech-256 Object Category Dataset.* Technical report 7694, California Institute of Technology, 2007. (Cited on pages xxiv, 78, 89 and 104.)

[Hafner *et al.* 1995] J. Hafner, H. Sawhney, W. Equitz, M. Flickner and W. Niblack. *Efficient color histogram indexing for quadratic form distance functions.* IEEE Trans. Pattern Anal. Mach. Intell, pages 729–736, 1995. (Cited on pages xii and 18.)

[Hansen *et al.* 1995] Nikolaus Hansen, Andreas Ostermeier and Andreas Gawelczyk. *On the Adaptation of Arbitrary Normal Mutation Distributions in Evolution Strategies: The Generating Set Adaptation.* pages 312–317. Morgan Kaufmann, 1995. (Cited on pages xvi and 39.)

[Haralick *et al.* 1973] R. M. Haralick, Dinstein and K. Shanmugam. *Textural features for image classification.* IEEE Transactions on Systems, Man, and

Cybernetics, vol. SMC-3, pages 610–621, November 1973. (Cited on pages xi and 9.)

[Harris & Stephens 1988] C. Harris and M. Stephens. *A combined corner and edge detector.* In Alvey Vision Conference, pages 147–151, 1988. (Cited on page 12.)

[He *et al.* 2010] Junfeng He, Wei Liu and Shih-Fu Chang. *Scalable Similarity Search with Optimized Kernel Hashing.* In ACM SIGKDD Conference, Washington, DC, USA, 2010. (Cited on page 29.)

[Hoi *et al.* 2008] Steven C. H. Hoi, Rong Jin, Jianke Zhu and Michael R. Lyu. *Semi-supervised SVM batch mode active learning for image retrieval.* IEEE CVPR, vol. 0, pages 1–7, 2008. (Cited on pages xiii and 24.)

[Indyk & Motwani 1998] P. Indyk and R. Motwani. *Approximate nearest neighbors: towards removing the curse of dimensionality.* ACM, pages 604–613, 1998. (Cited on page 96.)

[Ishibuchi *et al.* 2008] Hisao Ishibuchi, Noritaka Tsukamoto and Yusuke Nojima. *Evolutionary many-objective optimization: A short review.* In IEEE congress on evolutionary computation, pages 2419–2426, 2008. (Cited on page 75.)

[J. A. Hartigan 1979] M. A. Wong J. A. Hartigan. *Algorithm AS 136: A K-Means Clustering Algorithm.* Journal of the Royal Statistical Society. Series C (Applied Statistics), vol. 28, no. 1, pages 100–108, 1979. (Cited on page 7.)

[Jamil & Yang 2013] Momin Jamil and Xin-She Yang. *A Literature Survey of Benchmark Functions For Global Optimization Problems.* International Journal of Mathematical Modelling and Numerical Optimisation, vol. 4, no. 2, pages 150–194, 2013. (Cited on page 65.)

[Johnson 2012] C.G. Johnson. *Search-based evolutionary operators for extensionally-defined search spaces: Applications to image search.* In Evolutionary Computation (CEC), 2012 IEEE Congress on, pages 1–7, June 2012. (Cited on pages xvi and 48.)

[Kanimozhi & Latha 2015] T. Kanimozhi and K. Latha. *An integrated approach to region based image retrieval using firefly algorithm and support vector machine.* Neurocomputing, vol. 151, Part 3, no. 0, pages 1099 – 1111, 2015. (Cited on pages xx and 77.)

[Kennedy & Eberhart 1995] J. Kennedy and R. Eberhart. *Particle swarm optimization.* In Neural Networks, 1995. Proceedings., IEEE International Conference on, volume 4, pages 1942–1948 vol.4, November 1995. (Cited on pages xvi, 37 and 40.)

[Kim 2003] Changick Kim. *Content-based image copy detection.* Signal Processing: Image Communication, vol. 18, no. 3, pages 169 – 184, 2003. (Cited on page 4.)

[Konak *et al.* 2006] Abdullah Konak, David W. Coit and Alice E. Smith. *Multi-objective optimization using genetic algorithms: A tutorial.* Reliability Engineering & System Safety, vol. 91, no. 9, pages 992 – 1007, 2006. Special Issue - Genetic Algorithms and ReliabilitySpecial Issue - Genetic Algorithms and Reliability. (Cited on page 54.)

[Lai & Chen 2011] Chih-Chin Lai and Ying-Chuan Chen. *A User-Oriented Image Retrieval System Based on Interactive Genetic Algorithm.* Instrumentation and Measurement, IEEE Transactions on, vol. 60, no. 10, pages 3318 –3325, October 2011. (Cited on pages xvi, xx, xxi, 4, 48, 76, 78, 79 and 104.)

[Lakdashti & Ajorloo 2011] Abolfazl Lakdashti and Hossein Ajorloo. *Content-Based Image Retrieval Based on Relevance Feedback and Reinforcement Learning for Medical Images.* ETRI Journal, vol. 33, no. 2, pages 240–250, April 2011. (Cited on pages xix, 4, 71 and 72.)

[Li *et al.* 2013] Xiaodong Li, Andries Engelbrecht and Michael G. Epitropakis. *Benchmark Functions for CEC'2013 Special Session and Competition on Niching Methods for Multimodal Function Optimization.* Technical report, Evolutionary Computation and Machine Learning Group, RMIT University, Australia, 2013. (Cited on pages 65 and 66.)

[Li *et al.* 2015] Bingdong Li, Jinlong Li, Ke Tang and Xin Yao. *Many-Objective Evolutionary Algorithms: A Survey.* ACM Comput. Surv., vol. 48, no. 1, pages 13:1–13:35, September 2015. (Cited on page 75.)

[Lindeberg 1994] T. Lindeberg. *Scale-space theory: A basic tool for analysing structures at different scales.* Journal of Applied Statistics, (Supplement on Advances in Applied Statistics: Statistics and Images: 2), vol. 21, no. 2, pages 224–270, 1994. (Cited on page 12.)

[Lindeberg 1998] T. Lindeberg. *Feature detection with automatic scale selection.* International Journal of Computer Vision, vol. 30, no. 2, pages 77–116, 1998. (Cited on page 12.)

[Ling & Jacobs 2007] H. Ling and D. Jacobs. *Shape classification using the inner-distance.* IEEE PAMI, 2007. (Cited on pages xii and 19.)

[Lowe 1999] D. G. Lowe. *Object recognition from local scale-invariant features.* In Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on, volume 2, pages 1150–1157 vol.2, 1999. (Cited on pages xi and 9.)

[Lowe 2004] David G. Lowe. *Distinctive Image Features from Scale-Invariant Keypoints.* International Journal of Computer Vision, vol. 60, no. 2, pages 91–110, 2004. (Cited on page 12.)

[Manjunath & Ma 1996] B. S. Manjunath and W. Y. Ma. *Texture features for browsing and retrieval of image data.* IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI - Special issue on Digital Libraries), vol. 18, no. 8, pages 837–42, August 1996. (Cited on page 9.)

[Martin *et al.* 2004] D. Martin, C. Fowlkes and J. Malik. *Learning to detect natural image boundaries using local brightness, color, and texture cues.* IEEE PAMI, 2004. (Cited on pages xii and 19.)

[Moon 1996] T. K. Moon. *The expectation-maximization algorithm.* IEEE Signal Processing Magazine, vol. 13, no. 6, pages 47–60, November 1996. (Cited on page 15.)

[Panda *et al.* 2006] N. Panda, K. Goh and E. Y. Chang. *Active learning in very large databases.* MTAP, vol. 31(3), pages 249–267, 2006. (Cited on page 29.)

[Perronnin *et al.* 2010] Florent Perronnin, Jorge Sánchez and Thomas Mensink. *Improving the Fisher Kernel for Large-scale Image Classification.* In Proceedings of the 11th European Conference on Computer Vision: Part IV, ECCV'10, pages 143–156, Berlin, Heidelberg, 2010. Springer-Verlag. (Cited on pages xii, xiii, xxii, xxiv, xxv, xxxi, 7, 15, 17, 22, 24, 89, 90, 104, 105, 106, 107, 108, 109, 110, 111 and 122.)

[Picard *et al.* 2008] David Picard, Matthieu Cord and Arnaud Revel. *Image retrieval over networks: Active learning using ant algorithm.* IEEE Transactions on Multimedia, vol. 10, no. 7, pages 1356–1365, 2008. (Cited on page 76.)

[Picard *et al.* 2012] David Picard, Arnaud Revel and Matthieu Cord. *An application of swarm intelligence to distributed image retrieval.* Information Sciences, vol. 192, pages 71–81, 2012. (Cited on page 76.)

[Rechenberg 1973] I. Rechenberg. Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Problemata, 15. Frommann-Holzboog, 1973. (Cited on pages xvi and 39.)

[Črepinšek *et al.* 2013] Matej Črepinšek, Shih-Hsi Liu and Marjan Mernik. *Exploration and Exploitation in Evolutionary Algorithms: A Survey.* ACM Comput. Surv., vol. 45, no. 3, pages 35:1–35:33, July 2013. (Cited on page 41.)

[Rubner *et al.* 1998] Y. Rubner, C. Tomasi and L. J. Guibas. *A metric for distributions with applications to image databases.* In Computer Vision, 1998. Sixth International Conference on, pages 59–66, January 1998. (Cited on page 77.)

[Russakovsky *et al.* 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. *ImageNet Large Scale Visual Recognition Challenge.* IJCV, pages 1–42, April 2015. (Cited on pages 78 and 104.)

[Schiele & Crowley 1996] B. Schiele and J. L. Crowley. *Object recognition using multidimensional receptive field histograms.* LNCS, pages 610–619, 1996. (Cited on pages xii and 19.)

[Schmid & Mohr 1997] C. Schmid and R. Mohr. *Local greyvalue invariants for image retrieval.* IEEE PAMI, vol. 19, no. 5, pages 530–535, 1997. (Cited on page 12.)

[Segura *et al.* 2015a] Carlos Segura, Carlos A Coello Coello and Alfredo G Hernández-Díaz. *Improving the vector generation strategy of differential evolution for large-scale optimization.* Information Sciences, vol. 323, pages 106–129, 2015. (Cited on pages xvi and 40.)

[Segura *et al.* 2015b] Carlos Segura, Carlos A Coello Coello, Eduardo Segredo and Coromoto León. *On the adaptation of the mutation scale factor in differential evolution.* Optimization Letters, vol. 9, no. 1, pages 189–198, 2015. (Cited on pages xvi and 40.)

[Shao & Lunetta 2012] Yang Shao and Ross S. Lunetta. *Comparison of support vector machine, neural network, and {CART} algorithms for the land-cover classification using limited training data points.* {ISPRS} Journal of Photogrammetry and Remote Sensing, vol. 70, pages 78 – 87, 2012. (Cited on page 26.)

[Shi *et al.* 2007] Shuo Shi, Jiu-Zhong Li and Li Lin. *Face Image Retrieval Method Based on Improved IGA and SVM.* In De-Shuang Huang, Laurent Heutte and Marco Loog, editors, Advanced Intelligent Computing Theories and Applications. With Aspects of Theoretical and Methodological Issues: Third International Conference on Intelligent Computing, ICIC 2007 Qingdao, China, August 21-24, 2007 Proceedings, pages 767–774. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. (Cited on pages xx and 76.)

[Siddique & Adeli 2015] Nazmul Siddique and Hojjat Adeli. *Nature Inspired Computing: An Overview and Some Future Directions.* Cognitive Computation, vol. 7, no. 6, pages 706–714, 2015. (Cited on page 39.)

[Sivic & Zisserman 2003] J. Sivic and A. Zisserman. *Video Google: A text retrieval approach to object matching in videos.* In Ninth IEEE international conference on computer vision, 2003. Proceedings, volume 2, pages 1470–1477, 2003. (Cited on pages xii and 13.)

[Storn & Price 1997] Rainer Storn and Kenneth Price. *Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces*. Journal of Global Optimization, vol. 11, no. 4, pages 341–359, 1997. 10.1023/A:1008202821328. (Cited on pages xvi, 37, 38, 39 and 62.)

[Takagi 2001] H. Takagi. *Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation*. Proceedings of the IEEE, vol. 89, no. 9, pages 1275–1296, September 2001. (Cited on page 47.)

[Tang 2013] Yichuan Tang. *Deep learning using linear support vector machines*. arXiv preprint arXiv:1306.0239, 2013. (Cited on pages xiii and 23.)

[Tong & Chang 2001] S. Tong and E. Chang. *Support vector machine active learning for image retrieval*. In ACM MM, pages 107–118, 2001. (Cited on pages xxiv, xxv, xxvi, 8, 27, 34, 36, 108, 109, 110, 111 and 112.)

[van Gemert *et al.* 2008] J.C. van Gemert, J.M. Geusebroek, C.J. Veenman and A.W.M. Smeulders. *Kernel codebooks for scene categorization*. In Proc. ECCV, volume 5. Springer, 2008. (Cited on pages xii and 14.)

[Viola & Jones 2001] P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. In CVPR, 2001. (Cited on pages xi and 9.)

[Wan *et al.* 2013] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun and Rob Fergus. *Regularization of neural networks using dropconnect*. In Proceedings of the 30th International Conference on Machine Learning (ICML-13), pages 1058–1066, 2013. (Cited on pages xiii and 23.)

[Wang *et al.* 2005] Shang-Fei Wang, Xu-Fa Wang and Jia Xue. *An improved interactive genetic algorithm incorporating relevant feedback*. In Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on, volume 5, pages 2996 –3001 Vol. 5, August 2005. (Cited on pages xx and 75.)

[Wang *et al.* 2010] Jun Wang, Sanjiv Kumar and Shih-Fu Chang. *Semi-Supervised Hashing for Scalable Image Retrieval*. In IEEE CVPR, San Francisco, USA, 2010. (Cited on page 29.)

[Wang *et al.* 2014] Y. Wang, H.-X. Li, G.G. Yen and W. Song. *MOMMOP: Multiobjective Optimization for Locating Multiple Optimal Solutions of Multimodal Optimization Problems*. Cybernetics, IEEE Transactions on, vol. PP, no. 99, pages 1–1, 2014. (Cited on pages xviii and 58.)

[Winn *et al.* 2005] J. Winn, A. Criminisi and T. Minka. *Object categorization by learned universal visual dictionary*. In Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, volume 2, pages 1800–1807 Vol. 2, October 2005. (Cited on page 4.)

[Wong *et al.* 2001] Kwok-Wai Wong, Kin-Man Lam and Wan-Chi Siu. *An efficient algorithm for human face detection and facial feature extraction under different conditions.* Pattern Recognition, vol. 34, no. 10, pages 1993 – 2004, 2001. (Cited on page 4.)

[Xu *et al.* 2013] K. Xu, W. Yang, G. Liu and H. Sun. *Unsupervised Satellite Image Classification Using Markov Field Topic Model.* IEEE Geoscience and Remote Sensing Letters, vol. 10, no. 1, pages 130–134, January 2013. (Cited on page 7.)

[Yang 2008] Xin-She Yang. *Firefly algorithm.* Nature-inspired metaheuristic algorithms, vol. 20, pages 79–90, 2008. (Cited on pages xvi, 37 and 40.)

[Yates & Neto 1999] R.B. Yates and B.R. Neto. *Modern information retrieval.* ACM P, 1999. (Cited on page 13.)

[Yu *et al.* 2016] Fei Yu, Yuanxiang Li, Bo Wei and Li Kuang. *Interactive differential evolution for user-oriented image retrieval system.* Soft Computing, vol. 20, no. 2, pages 449–463, 2016. (Cited on pages xx and 76.)

[Zeiler *et al.* 2011] M. D. Zeiler, G. W. Taylor and R. Fergus. *Adaptive deconvolutional networks for mid and high level feature learning.* In Computer Vision (ICCV), 2011 IEEE International Conference on, pages 2018–2025, November 2011. (Cited on pages xii, 15 and 16.)

[Zitzler *et al.* 2000] Eckart Zitzler, Kalyanmoy Deb and Lothar Thiele. *Comparison of multiobjective evolutionary algorithms: Empirical results.* Evolutionary computation, vol. 8, no. 2, pages 173–195, 2000. (Cited on pages xviii and 57.)

[Zitzler *et al.* 2001] Eckart Zitzler, Marco Laumanns, Lothar Thiele and others. *SPEA2: Improving the strength Pareto evolutionary algorithm.* In Eurogen, volume 3242, pages 95–100, 2001. (Cited on page 57.)

[Zitzler *et al.* 2004] Eckart Zitzler, Marco Laumanns and Stefan Bleuler. *A Tutorial on Evolutionary Multiobjective Optimization.* In Xavier Gandibleux, Marc Sevaux, Kenneth Sörensen, Vincent T'kindt, M. Beckmann, H. P. Künzi, G. Fandel and W. Trockel, editors, Metaheuristics for Multiobjective Optimisation, volume 535 of *Lecture Notes in Economics and Mathematical Systems*, pages 3–37. Springer Berlin Heidelberg, 2004. 10.1007/978-3-642-17144-4_1. (Cited on pages 44, 46 and 93.)

Personal Publications

[Pighetti *et al.* 2012] R. Pighetti, D. Pallez and F. Precioso. *Hybdrid Content Based Image Retrieval combining multi-objective interactive genetic algorithm and SVM.* In Pattern Recognition (ICPR), 2012 21st International Conference on, pages 2849–2852, 2012. (Cited on pages xxi, 78 and 79.)

[Pighetti *et al.* 2015a] R. Pighetti, D. Pallez and F. Precioso. *Comparative Study of Recent Multimodal Evolutionary Algorithms.* In Computational Intelligence, 2015 IEEE Symposium Series on, pages 837–844, December 2015. (Cited on pages xviii and 66.)

[Pighetti *et al.* 2015b] R. Pighetti, D. Pallez and F. Precioso. *Improving SVM Training Sample Selection Using Multi-Objective Evolutionary Algorithm and LSH.* In Computational Intelligence, 2015 IEEE Symposium Series on, pages 1383–1390, December 2015. (Cited on page 81.)

Romaric Pighetti, December the 7th 2016 at Sophia Antipolis