République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Batna 2
Faculté des mathématiques et d'informatique

Thèse

En vue de l'obtention du diplôme de
Doctorat en sciences

Spécialité : Informatique

"Approches de classification des images à grande échelle sur des architectures massivement parallèles."

Présentée Par
Saliha MEZZOUDJ

Membres du jury :

Président :	Pr. Kamal Eddine MELKEMI	Université Batna 2
Rapporteur :	Dr. Ali BEHLOUL	Université Batna 2
Invité :	Pr. Rachid SEGHIR	Université Batna 2
Examinateurs :	Pr. Allaoua CHAOUI	Université Constantine 2
	Pr. Leila DJEROU	Université Biskra
	Dr. Abdelhamid DJEFFAL	Université Biskra

Dédicace

Je dédie cette thèse à :
-Toute ma famille et en particulier ma mère et mon père qui m'ont donné l'éducation sans laquelle je n'en serai pas là aujourd'hui.
-Mon mari qui m'a aidé et supporté, à ma chère fille Nada pour son amour continu.
-Mes chers frères et soeurs : Houssin, Salim, Hassna, Yassmina, qui n'ont cessé de m'encourager à continuer dans des moments de relâchement et d'hésitation.
-Mes ami(e)s et tout ceux qui m'ont aidé.

Remerciements

-Avant toute chose, je remercie Dieu Tout Puissant de m'avoir aidée et éclairée le chemin pour la réalisation de cette thèse.

-A Monsieur Ali BEHLOUL de m'avoir proposé ce sujet intéressant, m'ouvrant ainsi les portes sur un domaine de recherche assez vivant, qui m'a permis, grâce à sa confiance et son soutien précieux, de réaliser ce travail, et pour l'attention qu'il m'a apporté durant toute cette période ainsi que pour sa lecture et ses commentaires qui m'ont permis d'améliorer cette thèse.

-A Monsieur Rachid SEGHIR pour l'attention qu'il m'a apporté durant toute cette période ainsi que pour sa lecture et ses commentaires qui m'ont permis d'améliorer cette thèse.

-Aux responsables de Grid'5000 pour le partage de leurs clusters.

-Un grand remerciement à Monsieur le président de jury : Pr. Kamal Eddine MELKEMI qui m'a soutenu durant mes années d'études de magister en 2014, et qui m'a honoré dans ce jour par son évaluation de ma thèse de doctorat, et ses précieux conseils afin d'améliorer mon travail.

-Mes sincères remerciements vont également aux membres du jury pour avoir étudié et commenté avec pertinence mes travaux de recherche.

je remercie tous mes enseignants depuis le primaire jusqu'à l'université.

Je tiens aussi à remercier tous ceux qui ont, de près ou de loin, aidé à rendre ce travail possible, que ce soit par des idées ou par des encouragements.

Résumé

Actuellement, l'énorme volume de données structurées et non structurées provenant de sources diverses, produites en temps réel et dépassent la capacité de stockage d'une seule machine. En effet, ces données sont difficilement traitées avec les systèmes de recherche d'information classique. Ce challenge nous a motivés pour élaborer un nouveau système de recherche et de classification des images permettant le stockage, la gestion et le traitement des grandes quantités d'images (Big Data) et qui imposent une parallélisation des calculs pour obtenir des résultats en temps raisonnable et avec une précision optimale. Cette thèse est organisée en deux parties. La première partie représente un état de l'art sur le domaine de la vision par ordinateur et ces concepts de base, les technologies de Big Data spécialement les plateformes de stockage et de calcul, elles présentent aussi l'infrastructure distribuée nécessaire pour rendre ces plateformes faciles à utiliser. Dans la deuxième partie un nouveau système de recherche d'images par le contenu rapide basé sur la plateforme Spark (CBIR-S) est présenté comme une première proposition tout en ciblant les bases d'images à grande échelle (Big Data). Dans ce système nous utilisons le modèle distribué MapReduce sous la plateforme Spark pour traiter de grands volumes de données en parallèle en divisant le travail en un ensemble de tâches indépendantes, et afin d'accélérer le processus d'indexation nous avons utilisé également un système de stockage distribué, qui s'appelle Tachyon ; et afin d'accélérer le processus de recherche nous utilisons l'algorithme parallèle de K-plus proches voisins (k-NN) de classification basée sur le modèle MapReduce implémenté sous Apache Spark. De plus, nous exploitons la méthode de cache de spark. Notre approche proposée permet d'accélérer les temps d'indexation, et de classification, tout en gardant les précisions des méthodes. Les tests effectués sur une base d'images ImageCLEFphoto 2008 montrent les avantages de système proposé. Comme une deuxième proposition, nous introduisons une nouvelle méthode de classification des races dans des base d'images faces à grande échelle basée sur la méthode Local Binary Pattern (LBP) et la méthode de régression logistique (LR) sous la plateforme Spark pour traiter d'une manière parallèle une grande échelle des faces. L'évaluation de notre méthode proposée a été effectuée sur la combinaison de deux grandes bases d'images de visage CAS-PEAL (partie de pose) et Color FERET. Deux races ont été considérées pour ce travail, y compris les races asiatiques et non asiatiques. Par comparaison, notre méthode avec les classificateurs SVM linéaires, bayésiens naïfs (NB), forêts aléatoires (RF) et arbres de décision (DT) de Spark, nous avons obtenu une précision moyenne de 99,99%, les résultats montrent que notre méthode est très compétitive.

Mots clés : Big Data ; système de recherche d'images ; classification ; les technologies de Big Data ; k-NN parallèle ; spark, Tachon ; cache ; classification des races ; LR ; NB ; RF ; DT.

Abstract

Recently, the huge volumes of structured and unstructured data from a variety of sources, produced in real time and exceeding the storage of a single machine. Indeed, these data are difficult to process with traditional information retrieval systems. This challenge has motivated us to develop a new image retrieval and classification system for storage, management and processing of large quantities of images (Big Data) ; and which require parallelisation of calculations to obtain results in reasonable time and with optimal precision. This thesis is organized in two parts. The first part represents a state of the art in the field of computer vision and its basic concepts, then Big Data technologies, especially the storage and computing platforms, it presents also the distributed infrastructure needed to make these platforms easy to use. In the second part a new fast system of image search by the content based on the Spark platform (CBIR-S) is presented as a first proposal which targeting large-scale image bases (Big Data). In this system we use the distributed model Mapreduce under the platform Spark to process large volumes of data in parallel by dividing the work into one set of independent tasks, and in order to accelerate the process of indexing, we used a distributed storage system, which called Tachyon ; and in order to speed up the search process we use the parallel algorithm of K nearest neighbours (k-NN) classification based on the Mapreduce model implemented under Apache Spark. In addition, we exploit the spark cache method. Our proposed approach allows faster indexing and classification times, while keeping the accuracy of the methods. Tests carried out on Imageclefphoto 2008 database, show the benefits of the proposed system. As a second proposal, we introduce a new classification method of races in large-scale face image base focused on the Local Binary pattern method (LBP) and logistic regression method (LR) under the Spark platform to treat in a parallel way a large scale of faces. The evaluation of our proposed method was performed on the combination of two large image bases face CAS-PEAL (pose part) and Color FERET. The two breeds were considered for this work, including Asian and non-Asian races. By comparison, our method with linear SVM classifiers, naive Bayesian (NB), random forests Spark's (RF) and decision trees (DT), and we achieved average accuracy 99.99%, the results show that our method is very competitive.

keywords : Big Data ; image retrieval system ; classification ; Big Data technologies ; k-NN parallèle ; spark, Tachon ; cache ; race clasification ; LR ; NB ; RF ; DT.

ملخـــص

إن ما نواكبه اليوم من الحجم الهائل للبيانات المهيكلة وغير المهيكلة الناتجة عن مصادر مختلفة، والتي يتم إنتاجها بديمومة وباستمرار حتى أنه يتجاوز سعة تخزين الآلة الواحدة، حيث بات من الصعب اليوم معالجة هذه البيانات باستخدام أنظمة استرجاع المعلومات التقليدية. دفعنا هذا التحدي إلى تطوير نظام جديد للبحث عن الصور وتصنيفها، مما يسمح بتخزين وإدارة ومعالجة كميات كبيرة من الصور (البيانات الكبيرة) والتي تتطلب حسابات متوازية للحصول على نتائج في وقت معقول، وبدقة مثالية، خاصة مع تطوير البنى المتوازية ووفرتها بشكل متزايد وبتكاليف معقولة، كما هو الحال مع المعالجات المتعددة. هذا ما يبرر دوافعنا لتوجيه جهودنا البحثية إلى تصنيف الصور على نطاق واسع نحو استغلال مثل هذه البنى مع منصات البيانات الكبيرة.

لتحقيق غايتنا تم تنظيم هذه الرسالة في جزئين: الجزء الأول يمثل دراسة لأحدث التقنيات في مجال رؤية الكمبيوتر ومفاهيمه الأساسية، ويستعرض تقنيات البيانات الكبيرة وخاصة منصات التخزين والحوسبة، كما يبين البنى التحتية الموزعة اللازمة لجعل مثل هذه المنصات سهلة الاستخدام. أما في الجزء الثاني: يتم تقديم نظام جديد (CBIR-S) للبحث السريع عن الصور باستخدام منصة Spark كمساهمة أولى مع استهداف قواعد بيانات الصور الكبيرة (البيانات الكبيرة). في هذا النظام نستخدم نموذج MapReduce الموزع ضمن منصة Spark لمعالجة كميات كبيرة من البيانات بالتوازي عن طريق تقسيم العمل إلى مجموعة من المهام المستقلة من أجل تسريع عملية الفهرسة، كما استخدمنا أيضًا نظام Tachyon لتخزين موزع للمعلومات، ولتسريع عملية البحث نستخدم الخوارزمية المتوازية للشرفات المجاورة (k-NN) للتصنيف بناءً على نموذج MapReduce المطبق تحت منصة Apache Spark بالإضافة إلى ذلك، نستغل طريقة ذاكرة التخزين المؤقت لـ Spark، هذا ما يتيح لنظامنا تسريع عملية الفهرسة والتصنيف مع الاحتفاظ بخصائص الطرق المستعملة. تُظهر الاختبارات التي أجريت على قاعدة صور ImageCLEFphoto 2008 مزايا النظام المقترح. في المساهمة الثانية نقدم طريقة جديدة لتصنيف الأجناس حسب الوجوه على نطاق واسع على أساس الصورة بناءً على طريقة النمط الثنائي المحلي (LBP) وطريقة الانحدار اللوجستي (LR) ضمن منصة Spark لمعالجة نطاق كبير من الوجوه بالتوازي.

تم إجراء تقييما لمقترحاتنا على مزيج من قاعدتين كبيرتين من صور الوجوه (CAS-PEAL و Color FERET)، تم اعتبار جنسين لهذا العمل بالتحديد الجنس الآسيوية وغير الآسيوية. من خلال النتائج المتحصل عليها للنظامين المقترحين مقارنة مع SVM الخطية ومصنفات Bayesian البسيطة (NB) والغابات العشوائية (RF) ومصنفات شجرة القرارات (DT) من Spark، حصلنا على متوسط دقة 99.99 ٪، حيث تأكد النتائج المتحصل عليها أن طريقتنا جد فعالة.

الكلمات المفتـاحيـة : تصنيف; نظام استرجاع الصور, البيانات الكبيرة ,Tachyon; Cache k-NN parallèle Spark ;LR; NB. ,. الاجناس ; تقنيات تصنيف البيانات الكبيرة . RF; DT

Table des matières

Table des figures

Liste des tableaux

Liste des publications

— Mezzoudj Saliha, Behloul Ali, Rachid Seghir, and Yassmina Saadna. "A Parallel Content-Based Image Retrieval System Using Spark and Tachyon Frameworks." Journal of King Saud University-Computer and Information Sciences, Elsevier, 2019.

— Mezzoudj Saliha, Behloul Ali, Rachid Seghir."Towards Large-Scale Face-Based Race Classification on Spark framework". Multimedia Tools and Applications78(18), p26729-26746, Springer, 2019.

Chapitre 1

Introduction générale

Ce chapitre présente le cadre général de cette thèse. Premièrement, le contexte de travail de recherche est introduit et les plateformes de base nécessaire sont examinées. Ensuite, les objectifs visés par cette thèse et les principales contributions sont décrits.

1.1 le contexte de travail de recherche

La vision par ordinateur (appelée aussi vision artificielle ou vision numérique) est une branche de l'intelligence artificielle dont le principal but est de permettre à une machine d'analyser, traiter et comprendre une ou plusieurs images prises par un système d'acquisition (exemple : caméras, mobile,etc.) [5]. Elle sert à automatiser les tâches que le système visuel humain peut faire : la reconnaissance, l'analyse de mouvement, la reconstruction de scène, et la restauration d'image [5]. Dans notre thèse, nous nous intéressons à la tâche de reconnaissance. Actuellement, on trouve plusieurs applications spécialisées basées sur la reconnaissance, tel que les systèmes de recherche d'images par le contenu(CBIR) et de classification d'images. La classification d'images est une tâche importante dans le domaine de la vision par ordinateur, qui nécessite le développement des systèmes de classification robuste, pouvant améliorer les performances des systèmes de vision.

En effet, la plupart des systèmes de classification d'images comportent trois étapes :

— La première étape : c'est l'extraction de caractéristiques de bas niveau des images (extraction des descripteurs), dont l'utilisation de descripteurs de bas niveau de l'image est le cœur des systèmes de classification d'images actuelle.
— La deuxième étape : c'est l'apprentissage des classes d'images en utilisant une méthode de classification afin de construire un modèle.
— La troisième étape : l'objectif ici est de tester le modèle aboutit dans l'étape 2 afin de prédire la classe de la nouvelle instance à classifier.

Lors de la première étape d'extraction des descripteurs de bas niveau, le choix des descripteurs utilisés se fait selon le type de problème à résoudre, par exemple on trouve beaucoup de systèmes qui choisissent, souvent, le descripteur Local Binary Pattern (LBP) [6], puisque c'est l'un des descripteurs les plus adaptés en raison de sa qualité de résultats retournés pour la reconnaissance faciale. Pendant, la deuxième étape (l'apprentissage), le choix habituel lors de cette étape est l'utilisation de l'algorithme de classification Support Vector Machines (SVM) avec des noyaux linéaires ou non linéaires. La troisième étape est le test de modèle abouti. On trouve que la plupart de ces systèmes conventionnels sont évalués sur de petites bases des images qui tiennent sans problème en mémoire centrale, comme Caltech-101 [7], Caltech-256 [8] ou PASCAL VOC [9].

Récemment, l'accroissement des images produites dans différents domaines couplés au développement d'outils informatiques a permis l'acquisition et le stockage d'une quantité importante d'informations, ce qui offre de nouveaux concepts comme les Big Data, qui sont d'énormes volumes de données structurées et non structurées provenant de sources diverses, produites en temps réel et dépassent la capacité de stockage d'une seule machine. En effet, ces données sont difficilement traitées avec les systèmes de recherche d'information classique.

1.2 Problématique et motivation

Comme les appareils photo numériques deviennent plus abordables et répandus, les images numériques sont en croissance exponentielle sur internet, tels que ImageNet 1 [10] qui se compose de 14 197 122 images étiquetées pour 21 841 classes. En effet, cette énorme quantité d'images rend la tâche de classification d'images beaucoup plus complexe et difficile à effectuer, surtout que les méthodes de traitement et de stockage traditionnel ne parviennent pas toujours à confronter cette énorme quantité d'images.

Ce challenge nous a motivés pour élaborer un nouveau système de recherche et de classification des images permettant le stockage, la gestion et le traitement des grandes quantités d'images (Big Data), cela impose une parallélisation des calculs pour obtenir des résultats en temps raisonnable et avec une précision optimale.

Les machines massivement parallèles sont de plus en plus disponibles à des coûts de plus en plus abordables, comme le cas des multiprocesseurs. Ce qui justifie notre motivation d'orienter nos efforts de recherche en classification des images à grande échelle vers l'exploitation de telles architectures avec des nouvelles plateformes de Big Data qui utilisent les performances de ces machines. Cela soulève trois questions principales :

— Comment peut-on stocker cette énorme quantité d'images ?

— Comment peut-on assurer un petit temps de traitement et de stockage des images à grande échelle ?

— Quelles sont les méthodes de classification qui nous donnent une meilleure précision dans les bases d'images à grande échelle ?

1.3 Motivations et objectifs

Afin de mieux comprendre le développement des applications, les méthodes de traitement, et de stockage des bases d'images à grandes échelles , il est nécessaire de définir une analyse efficace des images. Nous allons concentrer sur la classification des images à grande échelle. Cependant, l'énorme quantité d'images rend le problème de classification très complexe.

Dans l'état de l'art, les plus prometteuses pour la classification des images sont basées sur les méthodes du noyau et les machines à vecteurs de support (SVM) [11], qui se sont révélés très efficaces et robustes pour résoudre de nombreuses questions de classification. Néanmoins, les techniques disponibles sont encore insuffisantes pour analyser les énormes quantités d'images, et d'autres méthodes sont nécessaires pour exploiter efficacement les plateformes de big data pour le développement des applications qui peuvent traiter et stocker cette base d'image à grand échelle d'une manière efficace et rapide.

Pour une ou plusieurs raisons, l'objectif nécessaire de cette thèse consiste à présenter dans un premier temps une synthèse des travaux récents qui touchent à la problématique d'indexation des images et de classification des images à grande échelle, ainsi qu'une étude des différentes architectures massivement parallèles. Par la suite, on se focalisera sur la proposition de nouvelles méthodes de classification qui tiennent en considération les spécificités des machines massivement parallèles dans le but d'aboutir à de meilleures performances possibles lors de classification de grandes bases d'images. Enfin, on va développer des méthodes de classification des images précises et rapides en utilisant des bases des images à grande échelle. Dans ce qui suit, nous examinerons le contexte de base nécessaire pour la classification des images à grande échelle (big data). Ensuite, nous énoncerons les objectifs de cette thèse.

— Développer de nouvelles approches pour la classification d'images à grande échelle.
— Appliquer les différentes méthodes d'apprentissage supervisé pour améliorer la précision des résultats de la classification des bases d'images à grand échelle.
— Proposer des nouvelles approches de classification dont le but est d'améliorer le temps de la classification des bases d'images à grand échelle.
— Valider les systèmes proposés avec des bases d'images à grande échelle (big data) et sous architectures parallèles et/ou distribuées afin d'assurer une rapidité de traitement.

Nous allons présenter notre solution pour le traitement efficace de tels ensembles de

données et les premiers résultats prometteurs par rapport aux approches de l'état de l'art sur différentes bases d'images à grande échelle.

1.4 Le contexte et les principales contributions de ce travail

Cette section résume les principaux axes abordés dans cette thèse et ses principales contributions. Nous nous concentrons particulièrement sur le problème de l'amélioration des performances de classification des bases d'images à grande échelle, soit en termes de précision des résultats ou en termes de temps de traitement et de stockage. Dans ce but, nous introduisons plusieurs nouvelles plateformes parallèles de traitement et de stockage des big data pour la classification en utilisant différentes méthodes de classification.

La contribution la plus importante de cette thèse est la proposition d'un nouveau système pour la classification des bases d'images à grande d'échelle, dont le but est de réduire la complexité des calculs et d'exploiter les performances des machines, ainsi que l'exploitation de différentes méthodes de classification d'images sous une nouvelle plateforme Spark afin d'accroître la précision de la classification et de tester notre système proposé sur trois bases d'images à grande échelle : ImageClef, ImageNet, Color Feret, et CAS-PEAL. Ces défis peuvent être résumés comme suit :

— Premièrement, nous avons profité dans nos développements des nouvelles technologies, telles que : Spark, HDFS, Tachyon, et le modèle de programmation Map Reduce afin de confronter à des bases d'images grande échelle (big data), et d'améliorer les performances de traitement et de stockage de Big Data.

— Deuxièmement, nous avons proposé un système de recherche d'images par le contenu rapide basé sur la plateforme Spark (CBIR-S) en ciblant les bases d'images à grande échelle (Big Data). Dans ce système nous utilisons MapReduce modèle distribué sous la plateforme Spark pour traiter des grands volumes de données en parallèle en divisant le travail en un ensemble de tâches indépendantes, dont le but est d'accélérer le processus d'indexation. Nous avons utilisé d'une part un système de stockage distribué, qui s'appelle Tachyon, et d'autre part nous utilisons l'algorithme parallèle de K-plus poches voisins (k-NN) de classification basée sur le modèle MapReduce implémenté sous Apache Spark pour accélérer le processus de recherche. En outre, la méthode de cache de spark a été adopté. Notre approche proposée permet d'accélérer les temps d'indexation, et de classification, tout en gardant les précisions des méthodes orignales. Les tests effectués sur une base d'images ImageCLEFphoto 2008 montrent les avantages du système proposé.

— Troisièmement, nous avons proposé une nouvelle méthode de classification des races, dans des bases d'images de visages à grande échelle basée,sur la méthode

Local Binary Pattern (LBP) et la méthode de régression logistique (LR) sous la plateforme Spark pour traiter d'une manière parallèle une grande échelle de visages. L'évaluation de notre méthode proposée a été effectuée sur la combinaison de deux grandes bases d'images de visage CAS-PEAL (partie de pose) et Color FERET. Deux races ont été prises pour ce travail, les races asiatiques et non asiatiques. Par comparaison à d'autres classificateurs tels que : SVM linéaires, bayésiens naïfs (NB), forêts aléatoires (RF) et arbres de décision (DT) de Spark, notre méthode a donné une meilleur précision moyenne de 99,99%. Le fait de comparer les résultats de notre méthode proposée à d'autres méthodes de l'état de l'art de classification des races, montre que notre méthode est très compétitive.

1.5 Structure de la thèse

Cette thèse est organisée en six (06) chapitres :

Chapitre 1 : présente un bref aperçu sur la classification des bases d'images à grande échelle. En outre, il introduit le fond, la motivation et les principales contributions originales de cette thèse.

Chapitre 2 : passe en revue le domaine de la vision par ordinateur et ses concepts de base, pour la compréhension du lecteur.

Chapitre 3 : présente d'une part l'état de l'art sur les technologies de Big Data, spécialement les plateformes de stockage et de calcul, et d'autre part l'infrastructure distribuée nécessaire pour rendre ces plateformes faciles à utiliser.

Chapitre 4 : présente l'architecture d'un système de recherche d'images par le contenu (CBIR) pour les bases des images à grande échelle (big data) sous la plateforme Spark. Afin d'accélérer la phase de recherche, nous avons adopté la méthode KNN parallèle afin de l'accélérer. Le système proposé est testé, en utilisant des bases d'images à grande échelle telle qu'ImageClef2008 [1] et ImageNet, sur un cluster d'un seul nœud et multi-nœud, constitué de cinq nœuds, sous grid5000 dans le cluster de Rennes [2].

Chapitre 5 : propose un système de classification de visages basé sur la race, en utilisant les bases d'images de visages à grand échelle sous la plateforme Spark. On faisons une comparaison entre notre système qui utilise la régression logistique à d'autres différents classificateurs tels que SVM linéaires, naïfs bayésiens (NB), forêts aléatoires (RF) et les arbres de décision (DT) sous Spark. L'évaluation de notre système proposé a été effectuée en combinant deux grandes bases d'images de faces CAS-PEAL (partie de pose) [12] et Color FERET [13].

Conclusion générale : Ce chapitre résume les principales contributions et conclu-

1. ImageCLEFphoto2008 : http ://www.imageclef.org/photodata
2. Grid5000 : Home :https ://www.grid5000.fr.

sions tirées de cette recherche. Il présente les plans futurs qui peuvent être mis en œuvre pour améliorer et élargir notre travail.

Chapitre 2

La vision par ordinateur et la classification

2.1 Introduction

Depuis sa naissance, l'intelligence artificielle cherche à simuler le raisonnement humain et construire des systèmes capables de trouver des solutions optimales à des problèmes complexes. Beaucoup de chercheurs de l'intelligence artificielle concentrent leurs études sur le domaine de l'apprentissage automatique (classification) qui permet à une machine de remplir des tâches qui sont difficiles de les remplir par des moyens conventionnels. L'apprentissage automatique est utilisé dans plusieurs domaines sociaux comme la détection de fraude, la conception de médicaments, les systèmes de recommandation, le filtrage de spams et la recherche web [14]. Il inclut plusieurs problématiques comme la régression, l'exploration des données (Data Mining) et la classification, ces dernières permettent de construire un modèle pour regrouper des objets à base des critères implicites ou explicites dans des groupes homogènes.

Ce chapitre a pour but de présenter un aperçu de la vision par ordinateur et la classification. On introduit d'abord, le domaine de vision par ordinateur, ensuite, on s'accentue sur les étapes principales d'un processus de reconnaissance d'objets, on finalise le chapitre par la citation de quelques méthodes de classification ainsi que les avantages et les inconvénients de chacune.

2.2 Qu'est-ce que la vision par ordinateur ?

La vision par ordinateur est un domaine interdisciplinaire qui traite la façon dont les ordinateurs peuvent être faits pour acquérir une compréhension de haute niveau à partir d'images ou de vidéos numériques. Autrement dit, la vision par ordinateur sert

à automatiser les tâches que le système visuel humain peut faire [15], [16].

Les tâches de vision par ordinateur comprennent les méthodes d'acquisition, de traitement, d'analyse et de compréhension d'images numériques et d'extraction de données de grandes dimensions du monde réel afin de produire des informations numériques ou symboliques, par exemple sous forme de décisions [17],[18].

2.2.1 Les domaines connexes à la vision par ordinateur

1- La Neurobiologie : Au cours du siècle dernier, il y a eu une étude approfondie des yeux, des neurones et des structures cérébrales consacrées au traitement des stimuli visuels chez les humains et divers animaux. Cela a conduit à une description grossière, mais compliquée, de la façon dont les systèmes de vision «réels» fonctionnent afin de résoudre certaines tâches liées à la vision. Ces résultats ont conduit à un sous-champ dans la vision par ordinateur où les systèmes artificiels sont conçus pour simuler le traitement et le comportement des systèmes biologiques, à différents niveaux de complexité. En outre, certaines méthodes basées sur l'apprentissage développée dans le cadre de la vision par ordinateur ont leur origine de la biologie, par exemple l'analyse et la classification des images et des caractéristiques basées sur les réseaux neuronaux et l'apprentissage profond.

2- L'infographie(Computer graphics) :

L'infographie produit des images à partir de modèles 3D, la vision par ordinateur produit souvent des modèles 3D à partir des images [19]. Il existe également une tendance vers une combinaison des deux disciplines, par exemple, comme exploré dans la réalité augmentée [1].

3- Le traitement/l'analyse d'images :

Ces deux domaines sont les plus étroitement liés à la vision par ordinateur, ils ont une tendance à se concentrer sur les images 2D, comment transformer une image en une autre, par exemple l'utilisation des opérations pixel comme le contraste, les opérations locales telles que l'extraction de contours ou le bruit, ou des transformations géométriques telles que la rotation de l'image [2].

4- La vision industrielle :

Un autre domaine plus proche à la vision par ordinateur, est la vision industrielle. Elle consiste à appliquer une gamme de technologies et de méthodes pour fournir une inspection automatique basée sur l'imagerie, un contrôle de processus et un guidage par robot [20] dans des applications industrielles. La vision artificielle a une tendance à se concentrer sur des applications dans la fabrication, par exemple, des robots basés

1. https ://www.hisour.com/fr/computer-vision-42799/
2. https ://www.hisour.com/fr/computer-vision-42799/

sur la vision et des systèmes d'inspection, de mesure ou de prélèvement basés sur la vision [21].

2.2.2 Les applications de vision par ordinateur

Les domaines de vision par ordinateur et de vision par machine se chevauchent de manière significative. La vision par ordinateur couvre la technologie de base de l'analyse d'image automatisée utilisée dans de nombreux domaines. La vision par machine désigne généralement un processus consistant à combiner l'analyse d'image automatisée avec d'autres méthodes et technologies afin de fournir une inspection automatisée et un guidage par robot dans les applications industrielles. Dans de nombreuses applications de vision par ordinateur, les ordinateurs sont pré-programmés pour résoudre une tâche particulière, mais les méthodes basées sur l'apprentissage deviennent de plus en plus courantes [3]. Aujourd'hui, la vision par ordinateur est utilisée dans une grande variété d'applications du monde réel, qui incluent [22] :

— La reconnaissance optique de caractères (OCR) : elle sert à lire les codes postaux manuscrits sur les lettres et la reconnaissance automatique des plaques d'immatriculation (ANPR) (voir Fig.2.1).

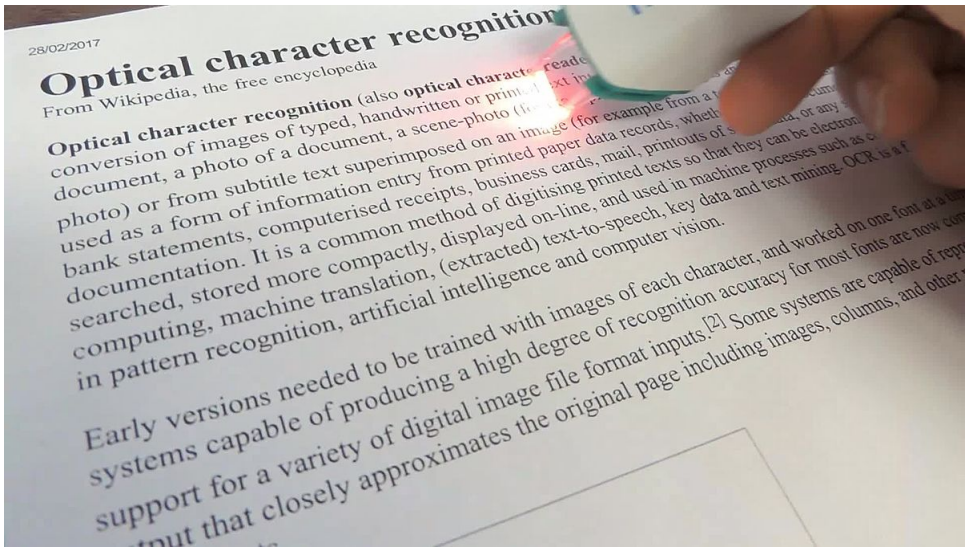

FIGURE 2.1 – Un exemple d'un système de reconnaissance optique de caractères [4]

— Les systèmes d'inspection des machines : elles permettent l'inspection rapide des pièces pour assurer la qualité en utilisant une vision stéréoscopique avec éclairage spécialisé afin de mesurer les tolérances sur les ailes d'avion ou les pièces de carrosserie ou la recherche de défauts dans les pièces moulées en acier en utilisant la vision par rayons X, (voir Fig. 2.2).

3. https ://www.hisour.com/fr/computer-vision-42799/

FIGURE 2.2 – Un exemple d'un système d'inspection de la machine
5

— La construction de modèles 3D (photogrammétrie) : la construction entièrement automatisée de modèles 3D à partir de photographies aériennes utilisées dans des systèmes tels que Bing Maps (voir Fig. 2.3).

— L'imagerie médicale : elle enregistre des images pré-opératoires et intra-opératoires ou effectue des études à long terme sur la morphologie cérébrale des personnes.

— La capture de mouvement (mocap) : il sert à utiliser des marqueurs rétro-réfléchissants vus à partir de plusieurs caméras ou d'autres techniques basées sur la vision pour capturer des acteurs pour l'animation par ordinateur.

— La surveillance des intrus, l'analyse de la circulation routière et la surveillance des piscines pour les victimes de noyade.

— La reconnaissance d'empreintes digitales et biométrie pour l'identification automatique d'accès ainsi que les applications médico-légales (voir Fig. 2.4).

2.3 Les tâches typiques de la vision par ordinateur

Chacun des domaines d'application décrits ci-dessus utilise une gamme de tâches de vision par ordinateur. Les tâches de vision par ordinateur comprennent des méthodes d'acquisition, de traitement, d'analyse, de compréhension d'images numériques et d'extraction de données de grandes dimensions du monde réel. L'objectif principal ici est de produire des informations numériques ou symboliques sous forme de décisions [17], [18].

À titre illustratif, citons quelques exemples de tâches de vision par ordinateur typique. [23] : la reconnaissance, l'analyse de mouvement, la reconstruction de scène, et la restauration d'image. En effet, dans ce chapitre nous allons concentrer notre attention sur les tâches de la reconnaissance.

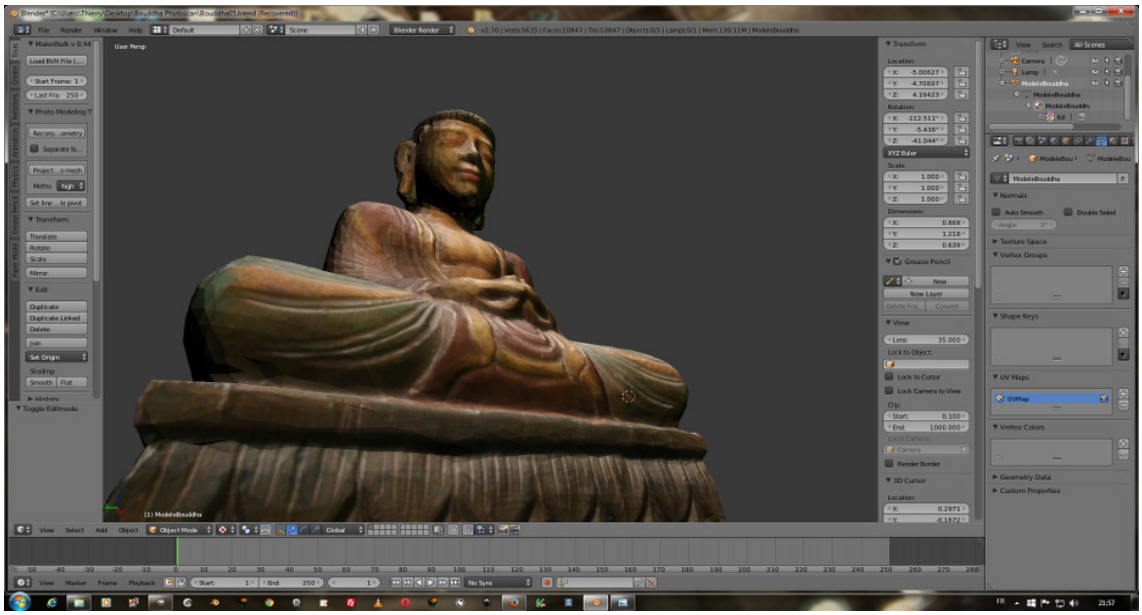

FIGURE 2.3 – Un exemple d'un système de modélisation 3D par photogrammétrie avec PhotoScan [7].

2.4 La reconnaissance

Parmi les problèmes de la vision par ordinateur est de déterminer si l'image contient un objet ou non, une caractéristique ou une activité spécifique. Dans ce qui suit, plusieurs tâches spécialisées basées sur la reconnaissance [23] :

— **La reconnaissance d'objet ou classification d'objet(object recognition) :** la reconnaissance d'objet consiste à déterminer l'identité d'un objet observé dans l'image à partir d'un ensemble d'objets ou classes connues (prédéfinies). Souvent, on suppose que l'objet observé a été détecté ou qu'il n'y a qu'un seul objet dans l'image [24].

— **L'identification (identification) :** c'est la reconnaissance d'une instance individuelle d'un objet. Les exemples incluent l'identification du visage ou de l'empreinte digitale d'une personne spécifique, l'identification des chiffres manuscrits, ou l'identification d'un véhicule spécifique [25].

— **La détection (detection) :** Dans la tâche de détection les images sont analysées pour une condition spécifique. La détection basée sur des calculs relativement simples et rapides est parfois utilisée pour trouver de plus petites régions d'image intéressantes qui peuvent être analysées plus en détail par des techniques plus exigeantes pour produire une interprétation correcte. Des exemples comprennent la détection de cellules ou de tissus anormaux possibles dans des images médicales ou la détection d'un véhicule dans un système de péage routier automa-

FIGURE 2.4 – Un exemple d'un système biométrique
[8]

tique [25].

— **La récupération d'images basée sur le contenu (Content-Based Image Retrieval CBIR) :** elle sert à rechercher sur une image requête dans un ensemble plus large d'images ayant un contenu spécifique. Le contenu peut être spécifié de différentes manières, par exemple en termes de similarité par rapport à une image cible (donnez-moi toutes les images similaires à l'image X) [26].

— **L'estimation de la position (pose estimation) :** estimation de la position ou de l'orientation d'un objet spécifique par rapport à la caméra. Un exemple d'application de cette technique consisterait à aider un bras de robot à récupérer des objets d'une bande transporteuse dans une situation de ligne d'assemblage ou à prélever des pièces d'un bac [27].

— **La reconnaissance optique de caractères (Object Caracter Recognition) :** identification de caractères dans des images de texte imprimé ou manuscrit, généralement en vue d'encoder le texte dans un format plus facilement modifiable ou indexable (par exemple ASCII) [28].

— **La reconnaissance faciale :** est un domaine de la vision par ordinateur consistant à reconnaître automatiquement une personne à partir d'une image de son visage [29].

Actuellement, les meilleurs algorithmes pour de telles tâches sont basés sur des réseaux de neurones convolutionnels. Une illustration de leurs capacités est donnée par le défi de reconnaissance visuelle à grande échelle d'ImageNet. Les performances des réseaux neuronaux convolutifs, sur les tests ImageNet, sont maintenant proches de celles des humains [30].

2.5 Les étapes principales d'un processus de reconnaissance d'objets :

Le processus de reconnaissance consiste en trois étapes principales après l'acquisition des données comme une étape initiale. En effet, les bases de données pour la reconnaissance de formes peuvent provenir de plusieurs sources, telles que les images de capteurs satellites, les images de capteurs au sol, les images médicales, etc. Une fois l'ensemble de bases de données acquis, elles sont pré-traitées dans une première étape qui sert à visant à améliorer les caractéristiques de l'objet. La deuxième étape c'est l'extraction de caractéristiques, dans laquelle l'ensemble de données est converti en un ensemble de vecteurs de caractéristiques pour représenter les données d'origine. Ces vecteurs sont utilisés comme entrées à la troisième étape qui est la classification. Cette dernière étape consiste à séparer les points de données en différentes classes en fonction d'un problème donné. Dans ce qui suit, un bref survol sur quelques méthodes d'extraction des caractéristiques et de classification pour la tâche de reconnaissance d'images [31] est présenté.

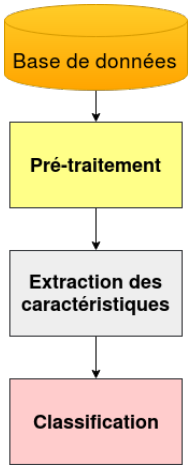

FIGURE 2.5 – Les étapes principales d'un processus de reconnaissance d'objets

2.5.1 Les techniques d'extraction des caractéristiques visuelles

L'extraction des caractéristiques des images est la base de tout système de vision par ordinateur qui fait de la reconnaissance. Ces caractéristiques peuvent contenir à la fois du texte (mots clés ; annotations,etc.), et des caractéristiques visuelles (couleur, texture, formes, visages, etc.). Nous allons nous intéresser aux techniques d'extraction de ces caractéristiques visuelles seulement. Et pour cela les caractéristiques visuelles (descripteurs) sont classées en deux catégories les descripteurs généraux et les descripteurs de domaine spécifiques[32], [31] :

2.5.1.1 Descripteurs générales

Ils contiennent des descripteurs de bas niveau qui donnent une description de la couleur, de la forme, des régions, des textures et du mouvement.

La couleur : La couleur est l'une des caractéristiques visuelles les plus utilisées dans les systèmes de reconnaissance faciale ou tout autre système pareil. Elle est relativement robuste aux complexités du fond et indépendante de la taille et de l'orientation de l'image. L'histogramme est la représentation la plus connue de la couleur, il désigne les fréquences d'apparition des intensités des trois canaux de couleur. De nombreuses autres représentations de cette caractéristique existent : on parle surtout des moments de couleurs. Le fondement mathématique de cette approche est que chaque distribution de couleur peut être caractérisée par ses moments de couleur. En outre, la plupart de l'information sur la couleur est concentrée sur les moments d'ordre inférieur qui sont respectivement : la moyenne, l'écart-type, l'asymétrie de couleur, la variance, la médiane, etc.

La texture : Une grande variété de descripteurs de texture ont été proposés dans la littérature. Ceux-ci étaient traditionnellement divisés en approche statistique, spectrale, structurelle et hybride [33]. Parmi les méthodes traditionnelles les plus populaires sont probablement celles basées sur des histogrammes, des filtres de Gabor [34], des matrices de cooccurrence [35] et des modèles (lbp) [36]. Ces descripteurs présentent différentes forces et faiblesses notamment en ce qui concerne leur invariance par rapport aux conditions d'acquisition.

La forme : Au cours des deux dernières décennies, les descripteurs de forme 2D ont été activement utilisés dans les moteurs de recherche 3D et les techniques de modélisation basées sur les esquisses. Certains des descripteurs de forme 2D les plus populaires sont l'espace d'échelle de courbure (CSS) [37], SIFT [38], SURF [39], et les contextes de forme [40]. En effet, dans la littérature, les descripteurs de forme 2D sont classés en deux grandes catégories : les contours et les régions. Les descripteurs de forme basés sur le contour extraient des entités de forme à partir du contour d'une forme uniquement. En revanche, les descripteurs de forme basés sur une région obtiennent des caractéristiques de forme à partir de la région entière d'une forme. En outre, des techniques hybrides ont été également proposées, combinant des techniques basées sur le contour et la région [41].

Mouvement : Le mouvement est lié au mouvement des objets dans la séquence et au mouvement de la caméra. Cette dernière information est fournie par le dispositif de capture, tandis que le reste est mis en œuvre au moyen du traitement d'images. L'ensemble de descripteurs est le suivant [32] : descripteur d'activité de mouvement (MAD), descripteur de mouvement de caméra (CMD), descripteur de trajectoire de mouvement (MTD), et descripteur de mouvement Warp et paramétrique (WMD et PMD).

Location : L'emplacement des éléments dans l'image est utilisé pour décrire les

éléments du domaine spatial. De plus, les éléments peuvent également être situés dans le domaine temporel [32] : Descripteur de localisateur de région (RLD), descripteur de localisateur temporel Spatio (STLD).

2.5.2 La classification

La classification des images est une étape importante dans le processus de reconnaissance d'images. En effet, de nombreuses techniques de classification des images ont été proposées jusqu'à ce jour. Elle est considérée comme l'une des types principaux de l'apprentissage automatique (machine learning). Diverses études ont été menées afin de choisir la meilleure technique de classification des images [42].

2.5.2.1 Qu'est-ce que l'apprentissage automatique ?

C'est l'un des sous-domaines de l'intelligence artificielle (AI) qui utilise une série de techniques pour laisser les ordinateurs apprendre, (c'est-à-dire, améliorer progressivement la performance de l'ordinateur sur une tâche spécifique) avec des données, sans être explicitement programmées. En effet, l'apprentissage automatique couvre un vaste champ de tâches. Ci-dessous les types d'apprentissage automatique décrites dans cette section [43] :

Apprentissage supervisé (classification) : Dans ce cas, les entrées sont étiquetées par un expert, et l'algorithme doit apprendre à partir des étiquettes de ces entrées afin de prédire la classe de chaque nouvelle entrée. Autrement dit, à partir d'un ensemble d'observations X et d'un autre ensemble de mesures Y, on cherche à estimer les dépendances entre X et Y.

Apprentissage non supervisé (clustering) : Dans ce cas, les entrées ne sont pas étiquetées, aucun expert n'est disponible, et l'algorithme doit prédire la classe de chaque entrée. L'objectif de ce type d'apprentissage est de décrire comment les données sont organisées et d'en extraire des sous-ensembles homogènes.

Apprentissage semi-supervisé : l'algorithme combine des exemples étiquetés et non étiquetés pour générer une fonction ou un classifieur approprié.

Apprendre à apprendre : où l'algorithme apprend son propre biais inductif basé sur une expérience précédente.
Dans notre thèse, nous allons nous concentrer sur l'apprentissage supervisé (classification).

2.5.2.2 Le processus de classification

L'apprentissage supervisé tente de trouver les relations entre l'ensemble des caractéristiques et étiquettes, c'est-à-dire extraire des connaissances et des propriétés à partir d'un ensemble de données étiqueté [43]. En effet, la tâche de classification est considérée comme une technique supervisée où chaque exemple appartient à une classe, et se compose de deux étapes la phase d'apprentissage et la phase de test [44] :

La phase d'apprentissage(training) : Dans la phase d'apprentissage(training), l'algorithme accède aux valeurs des deux attributs de prédicateur (valeurs d'attribut (caractéristique), classe) pour tous les exemples de l'ensemble d'apprentissage, et il utilise cette information pour construire un modèle de classification. Ce modèle représente la connaissance de classification, c'est une relation entre les valeurs d'attribut de prédiction et les classes, ce qui permet la prédiction de la classe d'un nouveau exemple donné.

La phase de test : Cette étape consiste à classifier des nouvelles instances ou instances inconnues. Elle consiste à estimer le taux d'erreur du modèle : la classe connue d'une instance « test » est comparée avec le résultat du modèle. Le taux d'erreur est défini par le pourcentage des tests incorrectement classés par le modèle.

En effet, l'un des principaux objectifs d'un algorithme de classification est de maximiser la précision prédictive obtenue par le modèle de classification lors de la classification des exemples dans l'ensemble de test.

2.5.3 Les méthodes de la classification

Les deux sections précédentes considèrent l'apprentissage automatique comme un outil ou un concept et discutent ses types, et ses idées de base. À partir de cette section, on introduit les techniques d'apprentissage automatique, notamment l'apprentissage supervisé. Il existe une variété de méthodes (algorithmes) de classification, voici une brève description de chaque méthode de classification :

2.5.3.1 k-Plus proche voisin (k-NN)

k-NN est l'un des algorithmes de classification les plus simples à tester et le plus rapide à mettre en œuvre. L'algorithme est basé sur un vote majoritaire des k-plus proches voisins, basé sur la distance Euclidienne dans l'espace de caractéristique pour prédire les classes de nouvelles données à partir de données étiquetées (données d'apprentissage), où k spécifie le nombre de voisins à utiliser.

Les exemples d'apprentissage sont des vecteurs dans un espace de caractéristiques multidimensionnel, chacun avec une étiquette de classe, cet algorithme ne nécessite

pas d'étape d'apprentissage pour être effectué mais peut être réglé pour déterminer la valeur optimale de k sur laquelle baser la classification. Les données d'apprentissage ont été mises à l'échelle de sorte qu'elles aient une moyenne de 0 et un écart type de 1 [45].

En effet, la phase d'apprentissage de l'algorithme k-NN consiste seulement à stocker les vecteurs de caractéristiques et les étiquettes de classe des échantillons d'apprentissage. Dans la phase de classification, k est une constante définie par l'utilisateur et un vecteur non étiqueté (une requête ou un point de test) est classé en affectant l'étiquette la plus fréquente parmi les k échantillons d'apprentissage les plus proches de ce point de requête. La distance Euclidienne est une mesure de distance couramment utilisée pour les variables continues. Pour les variables discrètes, telles que la classification de texte, une autre mesure peut être utilisée, telle que la mesure de chevauchement (ou distance de Hamming).

Avantages :

— Apprentissage rapide.

— Méthode facile à comprendre.

— Adapté aux domaines où chaque classe est représentée par plusieurs prototypes.

Inconvénients :

— Prédiction lente car il faut revoir tous les exemples à chaque fois.

— Méthode gourmande en place mémoire.

— Sensible aux attributs non pertinents et corrélés.

2.5.3.2 La régression logistique

La régression logistique est une alternative de la méthode de Fisher, l'analyse discriminante linéaire [46], [47]. Elle mesure la relation entre la variable dépendante catégorielle et une ou plusieurs variables indépendantes en estimant les probabilités à l'aide d'une fonction logistique, qui est la distribution logistique cumulative. Une fonction logistique ayant la forme suivante est utilisée [46] :

$$\theta = \frac{1}{1+e^{\theta}}, \theta = a + \sum_{i=1}^{n} b_i X_i \tag{2.1}$$

Où Xi représente l'ensemble des variables individuelles, bi est le coefficient de la variable ith, et θ est la probabilité d'un résultat favorable. Le résultat θ est une variable aléatoire de Bernoulli. Avantages et Inconvénients de la régression logistique [46] :

Avantages :

— Applicable aux situations non linéaires.

— La pondération d'interactions est possible.

Inconvénients :

— La régression logistique nécessite des échantillons de grande taille pour pouvoir prétendre un niveau acceptable de stabilité. Un nombre minimal de cinquante (50) observations par variables est en général nécessaire.

— Les catégories auxquelles appartiennent les variables indépendantes doivent être mutuellement exclusives et exhaustives, car un prédicateur ne peut pas appartenir aux deux groupes Y0 et Y1 à la fois.

2.5.3.3 Les séparateurs à vastes marges (SVM) (Support Vecteur Machines)

La méthode SVM est relativement moderne, elle est apparue en 1995 suite aux travaux de « Vapnik ». C'est une technique d'apprentissage supervisé basée sur la théorie de l'apprentissage statistique ou automatique. Le but d'un SVM est de déterminer si un élément appartient à une classe ou non. Nous disposons d'un ensemble de données et nous cherchons à séparer ces données en deux groupes. Le premier est l'ensemble de données appartenant à une classe, ces données sont étiquetées par (+) et un autre ensemble qui contient les éléments qui n'appartiennent pas à la classe (-) [48], [49].

En effet, dans les SVMs, en choisissant un hyperplan séparateur qui maximise la distance entre les classes. Les SVM sont capables de gérer les problèmes où les classes ne sont pas linéairement séparables en transformant les données en utilisant une fonction noyau telle que le noyau de la fonction de base radiale (RBF). Le noyau RBF est le choix le plus commun pour les tâches de classification [48].

Principe de l'algorithme SVM : Soit un ensemble d'apprentissage où chaque élément appartient à l'une des deux classes notées +1 et -1 et supposées linéairement séparables, l'algorithme recherche un hyperplan séparateur entre ces deux classes qui maximise la «marge».

— Hyperplan : h sous forme d'une fonction linéaire : h(x) = w.x + b.

— La marge : est définie comme la distance euclidienne entre la surface de séparation (hyperplan) et le point le plus proche de l'ensemble d'apprentissage.

Dans certains cas les données sont non linéairement séparables. Il n'existe donc pas un hyperplan séparateur. Deux options sont possibles pour pallier ce problème :

Définir une constante qui permet de tolérer une marge d'erreur (marge souple).Nous pouvons distinguer une constante pour les classes positives (Cp) et une constante pour les classes négatives (Cn). Les deux constantes sont en relation par un coefficient j. Soit $C_p = j\ddot{O}C_n$.

Chercher une autre séparation non linéaire : Pour faciliter les calculs de cette frontière non linéaire nous utilisons une fonction noyau qui nous permet de passer dans un autre espace vectoriel de plus grande dimension où une frontière linéaire existe. Parmi les avantages et les inconvénients des SVMs sont, on peut enumerer :

Avantages :

— Les SVMs sont très bons lorsque nous n'avons aucune idée des données.

— Les SVMs fonctionne bien avec des données même non structurées et semi-structurées telles que du texte, des images et des arbres.

— Le noyau est la vraie force de SVM. Avec une fonction de noyau appropriée, nous pouvons résoudre tout problème complexe.

Inconvénients :

— Choisir une «bonne» fonction du noyau n'est pas facile.

— Long temps d'apprentissage pour les grandes bases de données.

— Difficile à comprendre et à interpréter le modèle final.

2.5.3.4 La classification Naive Bayesienne (Naive Bayes)

Naive Bayesien a été étudié intensivement depuis les années 1950. Il a été introduit sous un nom différent dans la communauté de récupération de texte au début des années 1960 [50] . Le classificateur NB calcule les probabilités de classe sur la base de Bayes'rule avec des hypothèses d'indépendance fortes (naïves) entre les caractéristiques [50].

Abstraitement, naive Bayesien est un modèle de probabilité conditionnel. Sachant que le problème d'instance à classer, représentée par un vecteur x = $(x_1, ..., x_n)$ représentant n éléments (variables indépendantes), il affecte à cette instance des probabilités

$$p(C_k \mid x_1, ..., x_n) \tag{2.2}$$

pour chacun des K résultats ou classes possibles C_k [50]. En utilisant le théorème de Bayes, la probabilité conditionnelle peut être décomposée comme suit :

$$p(C_k \mid x) = \frac{p(C_k)p(x \mid C_k)}{p(x)} \tag{2.3}$$

En utilisant la terminologie de probabilité bayésienne, l'équation ci-dessus peut s'écrire comme suit :

$$posterior = \frac{prior * likelihood}{evidence} \tag{2.4}$$

Où $p(C_k)$ est la probabilité d'apparition des classes C_i, $p(x \mid C_k)$ est le terme de vraisemblance et p(x) est la probabilité d'apparition des primitives x_i. Ainsi, l'étape critique dans la classification bayésienne est de déterminer le terme de vraisemblance.

2.5.3.5 L'arbre de décision(Decision tree)

Un arbre de décision est une structure semblable à un organigramme, dans laquelle chaque nœud interne représente un "test" sur un attribut, chaque branche représente le résultat du test, et chaque nœud représente une classe (décision prise après le calcul de tous les attributs). Les chemins de la racine à la feuille représentent les règles de décision.

l'architecture d'un arbre de décision :

Un arbre de décision n'a que des noeuds éclatés (chemins de fractionnement) mais pas de noeuds puits (chemins convergents). Par conséquent, l'arbre de décision peut être linéarisé en règle de décision [51], où le résultat est le contenu du nœud feuille, et les conditions le long du chemin forment une conjonction dans la clause if. En général, les règles de décision ont la forme suivante :

si condition1 et condition2 et condition3 alors résultat.

Les règles de décision peuvent être générées en construisant des règles d'association avec la variable cible sur la droite. Ils peuvent également désigner des relations temporelles ou causales [51]. Avantages et inconvénients des arbres de décision :

Avantages :

— Ils sont simples à comprendre et à interpréter. Les gens sont capables de comprendre les modèles d'arbre de décision après une brève explication.
— Il a une valeur, même avec peu de données concrètes. Autoriser l'ajout de nouveaux scénarios possibles.
— Il aide à déterminer les valeurs les plus mauvaises, les meilleures et les plus attendues pour différents scénarios.
— Il peut combiner avec d'autres techniques de décision.

Inconvénients :

— Ils sont instables, ce qui signifie qu'une petite modification dans les données peut entraîner une modification importante de la structure de l'arbre décisionnel optimal.
— Ils sont souvent relativement inexacts. Beaucoup d'autres prédicteurs fonctionnent mieux avec des données similaires. Cela peut être résolu en remplaçant un seul arbre de décision par une forêt aléatoire d'arbres de décision, mais une forêt aléatoire n'est pas aussi facile à interpréter qu'un arbre de décision unique.
— Les calculs peuvent devenir très complexes, en particulier si de nombreuses valeurs sont incertaines et/ou si de nombreux résultats sont liés.

2.5.3.6 Les forêts aléatoires(Random Forest) :

Elles ont été formellement proposées en 2001 par Leo Breiman et Adèle Cutler. Une forêt aléatoire est un ensemble d'arbres de décision binaire dans lequel a été introduit de l'aléatoire [52]. L'algorithme des forêts d'arbres décisionnels effectue un apprentissage sur de multiples arbres de décision entraînés sur des sous-ensembles de données légèrement différents.

Parmi les avantages de l'algorithme des forêts aléatoires comparé à d'autres techniques de classification. Pour les applications dans les problèmes de classification, l'algorithme Random Forest évitera le problème de sur-apprentissage pour les tâches de classification et de régression. Il est en général plus efficace que les simples arbres de décision, et il a également été démontré que la vitesse de convergence des forêts aléatoires était plus élevée que celle des estimateurs ordinaires, mais elle possède l'inconvénient d'être plus difficilement interprétable [52].

2.6 Conclusion

Dans ce chapitre, nous avons présenté le domaine de vision par ordinateurs, la classification et ses différents domaines où elle peuvent intervenir. Dans le chapitre suivant nous allons présenter un background théorique sur les big data, les technologies de calcul et de stockage y relatifs, ainsi que l'infrastructure distribuée nécessaire à utiliser dans le domaine de vision par ordinateur et la classification.

Chapitre 3

Background théorique

3.1 Introduction

Actuellement, des trillions d'octets de données (Big Data) se génèrent par jour. Ces données proviennent de différentes sources telles que : les capteurs dédiés à collecter les informations climatiques, les messages sur les réseaux sociaux, les images numériques et des vidéos publiées en ligne.

Les Big Data se caractérisent par leur volumétrie (données massives) ; ils sont connus aussi par leur variété en termes de formats et de nouvelles structures, ainsi, qu'une exigence en termes de rapidité dans le traitement. À ce jour, il n'existe aucun logiciel capable de gérer plusieurs types et formes de données qui n'arrête pas d'accroitre très rapidement. De ce fait , la problématique du Big Data fait partie de notre vie quotidienne, et nécessite des solutions plus avancées pour gérer cette masse de données dans des délais très réduits.

Le calcul distribué concerne le traitement de grandes quantités de données. Ce traitement ne peut être réalisé par les paradigmes classiques de traitement de données, mais il nécessite l'utilisation des plateformes distribuées. Dans la littérature, il existe plusieurs solutions, pour l'implémentation de ce paradigme. Parmi ces solutions on trouve l'exemple Google qui a développé un modèle de programmation très fiable pour le traitement de Big Data : c'est le modèle MapReduce. Ce modèle est implémenté sur plusieurs plateformes comme la plateforme Hadoop. Cependant, cette dernière souffre de problèmes de la latence considérée comme le principal motif de développement d'une nouvelle alternative pour améliorer les performances du traitement, c'est la plateforme Spark qui est plus puissante, plus souple et rapide que Hadoop MapReduce.

Dans ce chapitre nous allons expliquer les notions de base de Big Data, les plateformes de traitement des Big Data, ainsi que le stockage.

3.2 Les notions de base des big Data

3.2.1 Définition

Le Big Data désigne un très grand volume de données souvent hétérogènes qui ont plusieurs formes et formats (texte, données de capteurs, son, vidéo, données sur le parcours, fichiers journaux, etc.), et comprenant des formats hétérogènes : données structurées, non structurées et semi-structurées. Le Big Data a une nature complexe qui nécessite des technologies puissantes et des algorithmes avancés pour son traitement et stockage. Ainsi, il ne peut être traité en utilisant des outils tels que les SGBD traditionnels [53]. La plupart des scientifiques et experts des données définissent le Big Data avec le concept des 3V comme suit [53] :

— Vélocité : Les données sont générées rapidement et doivent être traitées rapidement pour extraire des informations utiles et des informations pertinentes. Par exemple, Wallmart (une chaîne internationale de détaillants à prix réduits) génère plus de 2,5 petabyte(PB) de données toutes les heures à partir des transactions de ses clients. YouTube est un autre bon exemple qui illustre la vitesse rapide du Big Data.
— Variété : Les données volumineuses sont générées à partir de diverses sources distribuées dans plusieurs formats (vidéos, documents, commentaires, journaux, par exemple). Les grands ensembles de données comprennent des données structurées et non structurées, publiques ou privées, locales ou distantes, partagées ou confidentielles, complètes ou incomplètes, etc.
— Volume : il représente la quantité de données générées, stockées et exploitées. Le volume des données stockées aujourd'hui est en pleine explosion il est presque de 800.000 Péta-octets, Twitter générer plus de 7 téraoctets chaque jour de données, Facebook génère plus de 10 téraoctets et le volume de données dans 2020 peut atteindre 40 zêta-octets [54].

Par la suite, les trois dimensions initiales sont élargies par deux autres dimensions des données Big Data (on parle aussi des « 5 V du Big Data») :

— Véracité : La véracité (ou validité) des données correspond à la fiabilité et l'exactitude des données, et la confiance que ces Big Data inspirent aux décideurs. Si les utilisateurs de ces données doutent de leur qualité ou de leur pertinence, il devient difficile d'y investir davantage.
— Valeur : Ce dernier V joue un rôle primordial dans les Big Data, la démarche Big Data n'a de sens que pour atteindre des objectifs stratégiques de création de valeurs pour les clients et pour les entreprises dans tous les domaines.

Une des raisons de l'apparition du concept de Big Data est le besoin de réaliser le défi technique qui consiste à traiter de grands volumes d'information de plusieurs types (structurée, semi structuré et non structurée) générée à grande vitesse. Le Big Data s'appuie sur quatre sources de données [55] :

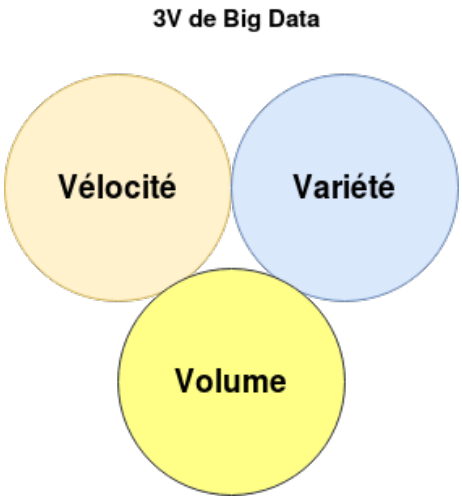

FIGURE 3.1 – Le modèle de Big Data 3V

— Les logs (journaux de connexion) issus du trafic sur le site officiel de l'entreprise : Ces sources de données, sont les chemins pris par les visiteurs pour parvenir sur le site : moteurs de recherche, annuaires, rebonds depuis d'autres sites, etc. Les entreprises d'aujourd'hui disposent d'une vitrine sur le Web au travers de son site officiel. Ce dernier génère du trafic qu'il est indispensable d'analyser, ainsi ces entreprises disposent des trackers sur les différentes pages afin de mesurer les chemins de navigation, ou encore les temps passés sur chaque page,.etc. Citons parmi les solutions d'analyse les plus connues : Google Analytics, Adobe Omniture, Coremetics.

— Les issus des médias sociaux «insights» : Une approche complémentaire, consiste à recueillir les commentaires aux publications et à y appliquer des algorithmes d'analyse de sentiment. Citons quelques pistes pour suivre nos différents comptes : Hootsuite, Radian6 ou encore les API mises à disposition et interrogées avec le complément Power Query pour Excel, IRaMuTeQ pour l'analyse de données textuelles.

— Les données comportementales (third party data) sont toutes des données récoltées sur les internautes via des formulaires ou des cookies. Au-delà des classiques informations d'identité (sexe, âge, CSP, etc), il est maintenant beaucoup plus efficace de mesurer les comportements (navigation, configuration matérielle, temps passé sur les pages, etc). Pour cela, il existe des acteurs spécialisés du Web qui nous aident à collecter de l'information sur nos clients ou prospects et à améliorer ainsi les campagnes de communication. Quelques acteurs du domaine de la third party data : Bluekai, Exelate, Weborama, Datalogix, etc.

— Les données ouvertes et réutilisables «L'open data» sont toutes les données ouvertes et réutilisables, L'open data permet de mettre en ligne les données ouvertes, de fiabiliser les données et de les rendre réutilisables et exploitables, où l'ouverture consiste à rendre une donnée publique : libre de droits, téléchar-

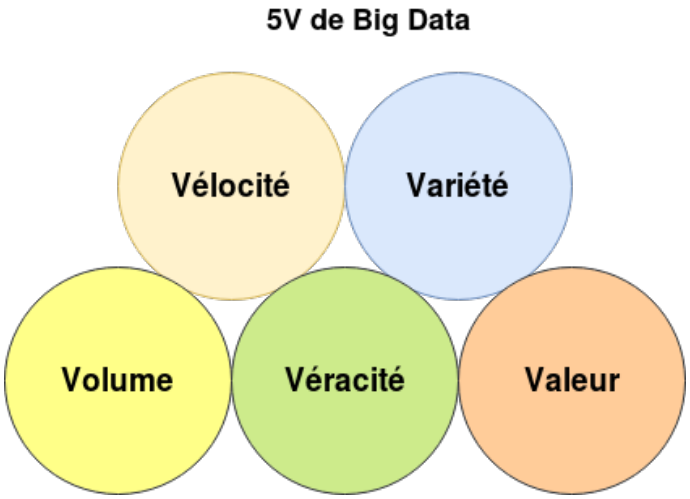

FIGURE 3.2 – Le modèle de Big Data 5V.

geable, réutilisable et gratuite. L'ouverture ne concerne pas les données à caractère privé, les informations sensibles et de sécurité, les documents protégés par des droits d'auteur, etc. Les données ouvertes et réutilisables ne sont pas encore légion même si une mission gouvernementale et très active sur le sujet manque de complétude, niveau de détail insuffisant, relative ancienneté sont les défauts actuels de nombreux jeux de données. Toutefois, c'est un champ d'investigation qu'il ne faut pas négliger, ne serait-ce que par son faible coût (celui du temps passé à chercher!) et son développement inéluctable.

3.2.2 Les technologies de traitement et de stockage de Big Data

Les Big Data requièrent de redéfinir les systèmes de stockage et de traitement de données, qui peuvent supporter ce volume de données. En effet, plusieurs technologies ont été proposées afin de représenter ces données, ces technologies prennent au moins un axe parmi les deux soit l'amélioration des capacités de stockage, soit l'amélioration de la puissance de calcul [53] :

— Amélioration de la puissance de calcul : le but de ces techniques est de permettre de faire le traitement sur un grand ensemble de données, avec un coût considérable et d'améliorer les performances de l'exécution comme le temps de traitement et la tolérance aux pannes. Avant l'apparition de la plateforme Hadoop on trouve plusieurs technologies comme le Could Computing, les architectures massivement parallèles MPP et les technologies In-Memory.

— Amélioration des capacités de stockage : l'amélioration du stockage vers des systèmes distribués, où un même fichier peut être réparti sur plusieurs disques durs,

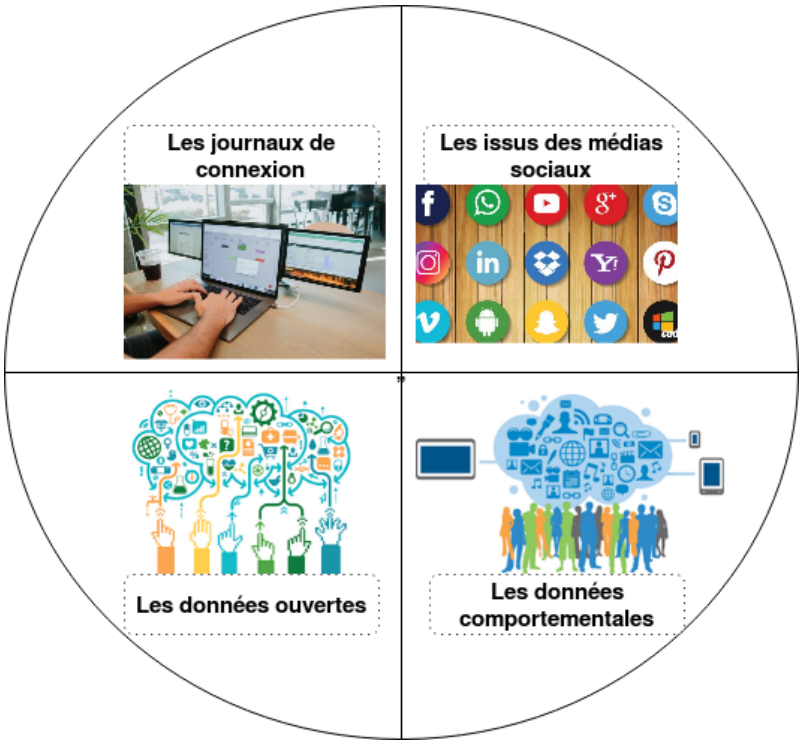

FIGURE 3.3 – Les quatre sources de Big Data.

cela permet d'augmenter les volumes de stockage en utilisant un matériel de base. Ces technologies de stockage évoluent toujours pour offrir des accès plus rapides aux données comme le NoSQL, HDFS de la plateforme Hadoop, HBase, le Cloud Computing,etc.

3.3 MapReduce

3.3.1 Pourquoi MapReduce?

Les systèmes d'entreprise traditionnels disposent normalement d'un serveur centralisé pour stocker et traiter les données. Le modèle traditionnel n'est certainement pas adapté au traitement de gros volumes de données évolutives et ne peut pas être géré par des serveurs de base de données standard. De plus, le système centralisé crée trop de goulot d'étranglement lors du traitement simultané de plusieurs fichiers. Google a résolu ce problème du goulot d'étranglement à l'aide de modèle MapReduce.

3.3.2 Définition de modèle MapReduce :

Il a été conçu dans les années 2000 par les ingénieurs de Google. C'est un modèle de programmation conçu pour traiter plusieurs téraoctets de données sur des milliers de nœuds de calcul dans un cluster [56]. MapReduce peut traiter des téraoctets et des pétaoctets de données plus rapidement et efficacement. Par conséquent, sa popularité a connu une croissance rapide pour diverses marques d'entreprises beaucoup de champs. Il fournit une plateforme très efficace pour l'exécution parallèle des applications, l'allocation de données dans des systèmes de bases de données distribuées, et communications réseau à tolérance de pannes [57]. L'objectif principal de MapReduce est de faciliter la parallélisation des données, leur distribution et l'équilibrage de la charge dans une bibliothèque simple [56].

3.3.3 L'architecture de modèle MapReduce

Google a créé MapReduce pour traiter de grandes quantités des données non structurées ou semi-structurées, tels que les documents et journaux de demandes de pages Web, sur de grands clusters de noeuds. Il a produit différents types de données telles que des indices inversés ou fréquences d'accès aux URL [58]. Le MapReduce comporte trois parties principales, y compris Master, la fonction de Map et de reduce. Un exemple de ce flux de données est présenté à la Fig. 3.4.

Le Master est responsable de la gestion des fonctions Map et Reduce et de la mise à leur disposition des données et des procédures, il organise la communication entre les mappeurs et les réducteurs. La fonction map est appliquée à chaque enregistrement d'entrée et produit une liste d'enregistrements intermédiaires. La fonction Réduire (également appelée Réducteur) est appliquée à chaque groupe d'enregistrements intermédiaires avec la même clé et génère une valeur. Par conséquent, le processus de MapReduce inclut les étapes suivantes :

— Les données d'entrée sont divisées en enregistrements.
— Les fonctions de Map traitent ces données et produisent des paires clé/valeur pour chaque enregistrement.
— Toutes les paires clé/valeur issues par la fonction Map sont fusionnées ensemble et regroupées par une clé, puis elles sont triées.
— Les résultats intermédiaires sont transmis à la fonction de réduction (Reduce), qui va produire le résultat final [59].

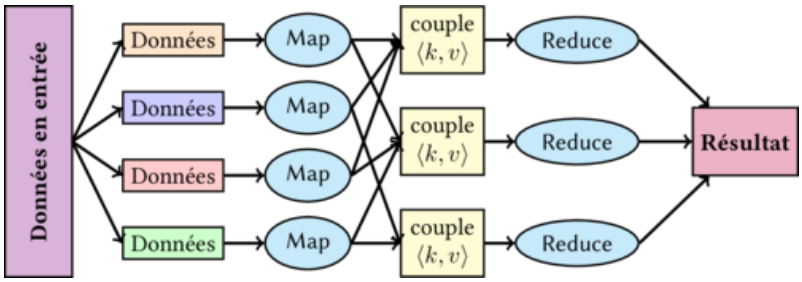

FIGURE 3.4 – Un exemple de flux de données dans l'architecture MapReduce [1]

3.3.4 Les applications de MapReduce

L'application MapReduce facilite l'exécution de nombreuses applications parallèles de données . MapReduce est le facteur principal dans de nombreuses applications importantes et peut améliorer le parallélisme du système. Il reçoit une attention considérable, pour les applications gourmandes en données et en temps de calculs sur des clusters de machines. Il est utilisé comme un outil de calcul distribué, efficace pour résoudre des problèmes variables, tels que la recherche, le regroupement, l'analyse de journaux, différents types d'opérations de jointure, la multiplication de matrices, la correspondance de modèles et l'analyse de réseaux sociaux. Il permet aux chercheurs d'étudier dans différents domaines [58].

MapReduce est utilisé dans de nombreuses applications de Big Data tels que [58] : l'exploration de messages courts, les algorithmes génétiques, k-means, algorithme de clustering, fragment d'ADN, système de transport intelligent, applications scientifiques dans le domaine de la santé, systèmes de classification à base de règles floues, environnements hétérogènes, la machine d'apprentissage extrême, forêt aléatoire, l'énergie proportionnelle, capteur mobile de données, web sémantique, etc.

3.4 Les plateForme de traitement des Big Data

3.4.1 La plateforme hadoop pour le calcul distribué de Big Data

Tout d'abord, Hadoop est un Framework libre, écrit en java, créé et distribué par la fondation Apache, et destiné au traitement de données volumineuses (de l'ordre des péta-octets et plus) ainsi qu'à leur gestion intensive. Inspirée de plusieurs publications techniques rédigées par le géant Google, son objectif est de fournir un système de stockage et de traitement de données distribué, évolutif et extensible. Il permet de traiter un grand nombre de types de données (y compris les données non structurées). On dit qu'il est organisé sur un mode non-relationnel, il est plus général que NoSQL, on peut par exemple stocker les données avec deux types de systèmes HDFS (Hadoop Distri-

buted File System) et HBase qui forment un système de gestion de bases de données orientées, colonnes projetées sur des serveurs distribués en clusters [3].

Hadoop parallélise le traitement des données à travers de nombreux nœuds faisant partie d'une grappe d'ordinateurs (clusters), ce qui permet d'accélérer les calculs et masquer la latence des opérations d'entrées et de sorties. Hadoop contient un système de fichiers distribué fiable qui garantit la tolérance aux pannes grâce à la réplication des données.

Hadoop contient deux composants de base, HDFS et MapReduce. Les deux sont liés au calcul distribué comme suit :

— HDFS est le responsable de stockage des données et MapReduce est le cœur de Hadoop qui effectue le traitement parallèle grâce aux deux fonctions Map et Reduce.

— Au niveau architectural, Hadoop contient deux types de nœuds (fig.3.5).

— Les serveurs ou bien les maitres : le NameNode, le NameNode secondaire et le JobTracker.

— Les esclaves qui sont distribués dans le cluster : le DataNode et le TaskTracker.

— L'exécution des tâches se fait par les TaskTrackers qui sont déployés sur chaque machine selon les instructions du JobTracker.

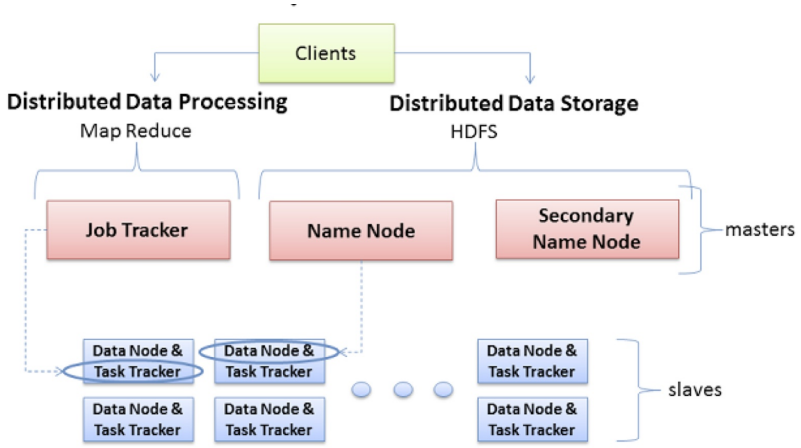

FIGURE 3.5 – L'architecture maître/esclave de la plateforme Hadoop [1]

3.4.2 La plateforme Spark pour le calcul distribué de Big Data

3.4.3 Motivation de Spark

Depuis sa création, Hadoop est devenu une technologie importante pour le Big Data. Une des principales raisons de ce succès est sa capacité à gérer d'énormes quan-

tités de données quel que soit leur type (structuré, semi structuré, non structuré). Toutefois, les utilisateurs ont été systématiquement plaignants du problème de la latence élevée avec Hadoop MapReduce indiquant que la réponse en mode batch pour toutes ces applications en temps réel est très douloureuse quand il s'agit de données de traitement et d'analyse.

3.4.4 Historique de Spark

Spark est un cluster de calcul rapide développé par les contributions de près de 250 développeurs de 50 entreprises AMPLab de l'Université de Berkeley. Spark a commencé en 2009 comme un projet de recherche dans le Berkeley Lab RAD, qui deviendra plus tard l'AMPLab. Les chercheurs en laboratoire avaient déjà travaillé sur Hadoop MapReduce, et ont observé que MapReduce était inefficace pour des emplois informatiques itératifs et interactifs. Ainsi, depuis le début, Spark a été conçu pour être rapide pour les requêtes interactives et les algorithmes itératifs, apportant des idées comme le support de stockage en mémoire et la récupération efficace de fautes.

Les documents de recherche ont été publiés à propos de Spark à des conférences universitaires et peu de temps après sa création en 2009, il était déjà de 10 à 100 fois plus vite que MapReduce pour certains emplois.

Certains des premiers utilisateurs de Spark étaient d'autres groupes à l'intérieur de l'Université de Berkeley, y compris des chercheurs, tel que le projet mobile du Millénaire, qui a utilisé Spark pour surveiller et prévoir les embouteillages dans la baie de San Francisco Machine Learning. Dans un temps très court, cependant, de nombreuses organisations externes ont commencé à utiliser Spark, et aujourd'hui, plus de 50 organisations listent eux-mêmes sur la Page Spark PoweredBy, et des dizaines parlent de leurs cas d'utilisation lors d'événements.

En 2011, l'AMPLab a commencé à développer des composants de haut niveau sur Spark, comme Shark et Spark streaming. Ceux-ci et d'autres composants sont parfois appelés comme Berkeley Data Analytics Stack (ODB). Spark a été en open source en Mars 2010, et il a été transféré à Apache Software Fondation en juin 2013, où il est maintenant un projet de haut niveau [60].

3.4.5 Définition

Apache Spark est un Framework open source de traitement, il est construit autour de la vitesse, la facilité d'utilisation et capacité de gérer des grands ensembles de données qui sont de nature diverse (données de texte, données de graphes, etc.), Spark étend le modèle MapReduce pour soutenir efficacement plusieurs types de calculs, y compris le traitement itératif, les requêtes interactives et le traitement de flux [60].

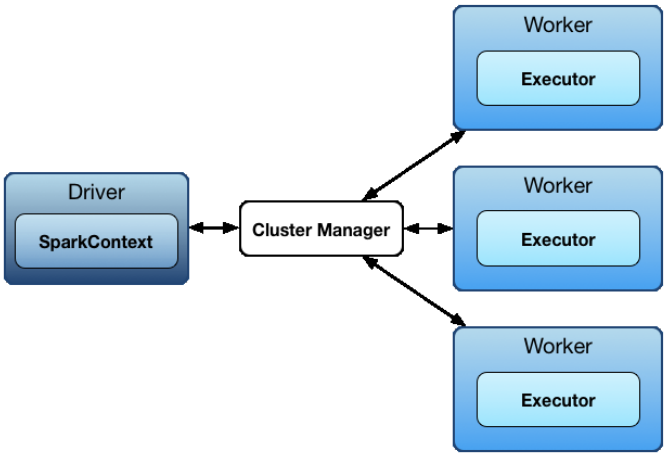

FIGURE 3.6 – Architecture de Spark [2]

3.4.6 Avantages de Spark par rapport à Hadoop MapReduce

Spark est un fort Framework pour les futures grandes applications de données qui peuvent nécessiter des requêtes de faible latence, calcul itératif et traitement en temps réel. Spark a beaucoup d'avantages par rapport au Framework Hadoop MapReduce parmi eux on trouve [61],[60] :

3.4.6.1 La rapidité

Spark est un environnement de calcul open source similaire à Hadoop, mais il a quelques différences utiles qui le rendent supérieur dans certaines charges de travail, il permet de charger l'ensemble de données en mémoire distribuée pour optimiser la charge de travail itératif et les requêtes interactives.

Spark peut exécuter les traitements de 10 à 100 fois plus rapidement que Hadoop MapReduce tout simplement en réduisant le nombre de lectures et écritures sur le disque.

3.4.6.2 Traitement itératif

Il existe beaucoup d'algorithme qui appliquent la même fonction à plusieurs étapes. Comme les algorithmes d'apprentissage, Hadoop MapReduce est basée sur un modèle de flux de données acyclique, c'est-à-dire que la sortie d'une tâche MapReduce précédente est l'entrée de la prochaine tâche MapReduce. Dans ce cas on perd beaucoup de temps dans l'opération d'E/S, alors dans Hadoop MapReduce entre deux opérations MapReduce, il existe une barrière de synchronisation et on a besoin de conserver les

données sur le disque à chaque fois [61].

Mais avec Spark, le concept de RDD (Resilient Distributed Datasets) permet d'enregistrer les données sur la mémoire et préserver le disque seulement pour les opérations de résultats. Ainsi il n'a pas tout une barrière de synchronisation qui éventuellement pourrait ralentir le processus. Alors Spark permet de réduire le nombre de lecture/écriture sur le disque, donc le temps principal est lié au traitement des données car il n'existe pas un temps de E/S.

3.4.6.3 Requêtes interactives

Pour le traitement dans les algorithmes d'extraction de données interactive où un utilisateur a besoin d'exécuter de multiples requêtes sur un même sous ensemble de données, Hadoop charge les mêmes données plusieurs fois à partir de disque selon le nombre de requêtes c'est-à-dire chaque requête à son propre lecteur de disque et son propre traitement MapReduce.

Mais Spark charge les données une seule fois, il stocke ces données dans la mémoire distribuée, ensuite il applique le traitement adéquat. Pour le traitement dans les algorithmes d'extraction de données interactive où un utilisateur a besoin d'exécuter de multiples requêtes sur un même sous ensemble de données, Hadoop charge les mêmes données plusieurs fois à partir de disque selon le nombre de requêtes c'est-à-dire chaque requête à son propre lecteur de disque et son propre traitement MapReduce.

3.4.6.4 Plus riche

Spark fournit des API concises et cohérentes à Scala, Java et Python et Prend en charge plusieurs fonctions (actions et transformations), contrairement à Hadoop, on trouve seulement les deux fonctions Map et Reduce et un seul langage Java.

3.4.6.5 Facilité d'utilisation

Spark vous permet d'écrire rapidement des applications en Java, Scala, ou Python avec des instructions simples et lisibles.

3.4.6.6 Généralité

Du côté de la généralité, Spark est conçu pour couvrir un large éventail de charges de travail qui nécessitent auparavent des systèmes distribués distincts, y compris les

applications de traitement en temps réel, les algorithmes itératifs, les requêtes interactives et le streaming. En soutenant ces charges de travail dans le même moteur, Spark est facile et peu coûteux de combiner les types de traitement différents, ce qui est souvent nécessaire dans les pipelines d'analyse de données de production.

3.4.6.7 Méthode streaming Real-Time de Spark à traiter des flux

En cas de Hadoop MapReduce il est juste possible de traiter un flot de données stockées, mais avec Apache Spark il est ainsi possible de modifier les données en temps réel grâce à Spark streaming [60].

3.4.6.8 Traitement graphique

Les développeurs peuvent maintenant aussi bien faire usage de Apache Spark pour le traitement graphique qui mappe les relations dans les données entre les diverses entités telles que les personnes et les objets [60].

3.4.6.9 Les algorithmes d'apprentissage

Spark est livré avec une bibliothèque d'apprentissage appelé MLlib, elle fournit plusieurs types d'algorithmes d'apprentissage, notamment la classification, la régression, le regroupement et le filtrage collaboratif, ainsi que l'appui des fonctionnalités telles que l'évaluation du modèle et l'importation de données [60]. Mais dans Hadoop il faut intégrer une bibliothèque d'apprentissage appelé Mahout.

3.4.6.10 Gestion rapide des données structurées

Spark SQL est le module de Spark pour travailler avec des données structurées. Spark SQL permet d'interroger des données structurées comme un ensemble de données distribuées (RDD) dans Spark, avec des API intégrées en Python, Scala et Java [3].

3.4.6.11 Généralité de stockage

Spark utilise le système de fichier HDFS pour le stockage de données. Il fonctionne aussi avec n'importe quelle source de données compatible avec Hadoop y compris, HBase, Cassandra, etc.

3. Spark Programming Guide - Spark 1.2.0 Documentation. [Online]. Available : http ://spark.apache.org/docs/1.2.0/programming-guide.html

3.4.6.12 Interactive

Offre une console interactive pour Scala et Python. Ce n'est pas encore disponible en Java.

3.4.7 Déploiement

Exécuter des traitements lourds sur un cluster, piloter les noeuds esclaves, leur distribuer les tâches équitablement, et arbitrer la quantité de CPU et de mémoire qui sera allouée à chacun des traitements, tel est le rôle d'un gestionnaire de cluster. Spark offre pour l'instant trois solutions pour cela : Spark standalone, YARN et Mesos. Livré avec Spark, Spark Standalone est le moyen le plus simple à mettre en place. Ce gestionnaire de cluster s'appuie sur Akka pour les échanges et sur Zookeeper pour garantir la haute disponibilité du noeud maître. Il dispose d'une console pour superviser les traitements, et d'un mécanisme pour collecter les logs des esclaves.

Autre possibilité, YARN le gestionnaire de cluster Hadoop, Spark peut s'exécuter dessus, et aux côtés de jobs Hadoop. Enfin, plus sophistiqué et plus généraliste, Mesos permet de configurer plus finement l'allocation des ressources (mémoire, CPU) aux différentes applications.

3.4.8 Composants de Spark

Parce que le moteur de base de Spark est à la fois rapide et polyvalent, il alimente de multiples composants de haut niveau spécialisés pour diverses charges de travail, tels que SQL ou l'apprentissage automatique. Ces composants sont conçus pour inter opérer étroitement, vous permettant de les combiner comme les bibliothèques dans un projet logiciel.

Spark Core : contient les fonctionnalités de base de Spark, y compris les composants pour la planification des tâches, gestion de la mémoire, la reprise après incident, interaction avec les systèmes de stockage, et plus. Spark Core est également l'API qui définit les ensembles de données distribués élastiques (RDD), qui sont les principales abstractions de programmation de Spark. RDD représentent une collection d'objets répartis sur plusieurs nœuds de calcul qui peuvent être manipulés en parallèle. Spark Core offre de nombreuses API pour la construction et la manipulation de ces collections.

Autre que Spark core API, il ya des bibliothèques supplémentaires qui font partie de l'écosystème de Spark et fournissent des capacités supplémentaires dans l'analyse des Big Data 3.7. Ces bibliothèques sont : Spark streaming, Spark SQL, Spark MLlib, Spark GraphX [60].

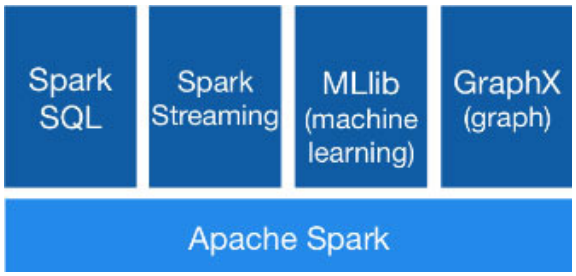

FIGURE 3.7 – Les composants de Spark

3.4.9 RDD Dataset Resilent Distribue

3.4.9.1 Définition

Un RDD est une collection d'objets partitionnés à travers un ensemble de machines, permettant aux programmeurs d'effectuer des calculs en mémoire sur grosses grappes d'une manière qui assure la tolérance aux pannes [4].

3.4.9.2 Caractéristiques

1. RDD atteindre la tolérance aux pannes à travers une notion de lignée : si une partition d'un RDD est perdue, le RDD a suffisamment d'informations pour reconstruire simplement cette partition. Cela supprime la nécessité de la réplication pour atteindre la tolérance aux pannes.

2. Il y a deux possibilités pour créer un RDD soit de référencer des données externes ou bien de paralléliser une collection existante. Spark permet de créer un RDD à partir de toute source de données acceptée par Hadoop (fichier local, HDFS, HBase, etc.).

3. Vous pouvez modifier un RDD avec une transformation, mais la transformation vous retourne une nouvelle RDD alors que le RDD original reste le même.

4. RDD prend en charge deux types d'opérations les transformations et les actions (voir figure 3.8) :

Transformation : les transformations ne renvoient pas une seule valeur, elles renvoient une nouvelle RDD. Rien n'est évalué lorsque vous appelez une fonction de transformation. L'évaluation des transformations est paresseuse, les opérations sont effectuées seulement quand un résultat doit être retourné.

Action : une opération qui évalue et renvoie une nouvelle valeur. Lorsqu'une fonc-

4. Spark Programming Guide - Spark 1.2.0 Documentation. [Online]. Available : http ://spark.apache.org/docs/1.2.0/programming-guide.html.

tion d'action est appelée sur un objet RDD, toutes les requêtes de traitement de données sont calculées à ce moment et la valeur de résultat est renvoyée [61].

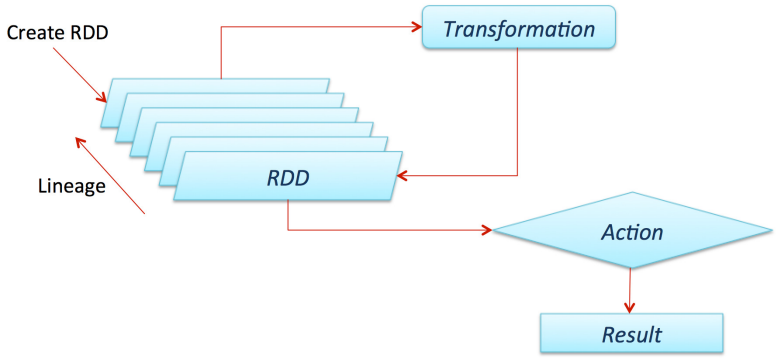

FIGURE 3.8 – Les actions et les transformations de Spark

3.5 Le stockage

Lors du traitement d'une grande quantité de données, les données d'entrée et les résultats doivent être stockés. En outre, la performance des applications à forte intensité de données dépend généralement de l'infrastructure matérielle et logicielle utilisée pour le stockage.

3.5.1 Le stockage classique

Ce type de système de stockage est parfois appelé stockage direct (DAS). Sur ces disques, les données sont stockées à l'aide d'un système de fichiers hiérarchique classique comme ext3 ou ReiserFS. Ces systèmes de fichiers sont généralement implémenté par un pilote du système d'exploitation comme une pièce sensible pour la sécurité, la performance et la fiabilité.

Ce type de stockage permet des opérations de lecture et d'écriture rapides puisque tout est effectué localement. Il est également simple à utiliser car il est utilisé avec n'importe quel système d'exploitation. Cependant, il n'existe aucun moyen simple d'échanger des données entre plusieurs nœuds.

3.5.2 Le stockage centralisé réseau

Une deuxième façon de stocker les données, c'est le stockage centralisé dans le réseau, habituellement appelé stockage attaché au réseau etworkAttached Storage (NAS).

Dans ce cas, un nœud a un ou plusieurs disques connectés et permet à d'autres noeuds de lire et d'écrire des fichiers via une interface standard et les servir par le biais du réseau. Le système de fichiers réseau (NFS) est principalement un protocole d'accès aux fichiers via le réseau. Bien que le serveur soit libre de mettre en œuvre n'importe quel moyen d'accès aux données réelles à fournir via le réseau, la plupart des implémentations dépendent simplement du fait que les données sont directement accessibles sur le serveur.

L'un des principaux avantages de ce type d'architecture est la facilité de partager des données entre plusieurs nœuds de calcul. Étant donné que les données sont stockées sur un serveur, elles sont facilement maintenues.

Cependant, l'un des principaux inconvénients est que les opérations simultanées de lecture et d'écriture doivent être supportées par un seul serveur. Il a également l'inconvénient de ne pas tolérer aux fautes par lui-même. Si le serveur de stockage tombe en panne, toutes les données ne sont pas disponibles.

3.5.3 Le stockage parallèle et distribué

Afin de surmonter les limites de stockage centralisé réseau, les données peuvent être réparties sur plusieurs nœuds de stockage. L'utilisation de plusieurs nœuds permet d'accéder à plusieurs fichiers en même temps sans conflit. Il permet également un meilleur débit pour lire le même fichier lorsqu'il existe des répliques sur plusieurs nœuds. Un système de fichiers distribués est habituellement conçu pour être meilleur que le stockage centralisé. De plus, en théorie, les systèmes de stockage distribués peuvent éviter le seul point de problème de tolérance à la panne.

Le système de stockage distribué possède souvent un noeud de point d'entrée unique qui reçoit toutes les demandes de lecture ou d'écriture de données. Comme son rôle est central et critique, son travail doit être réduit au minimum. Ce nœud maître ne permet généralement qu'une collaboration globale parmi tous les noeuds impliqués dans le système de stockage et peut stocker les méta-données (nom de fichier, taille de fichier, attributs d'accès, ...). Ainsi, lorsqu'une demande de lecture ou d'écriture d'un fichier est reçue, le client est redirigé vers un autre nœud qui va réellement traiter sa demande. Cependant, si les métadonnées peuvent être stockées sur le noeud maître, les données réelles sont toujours stockées sur d'autres nœuds et peuvent être reproduites.

L'inconvénient d'un stockage distribué est que pour obtenir les meilleures performances, l'application devrait tenir compte de la localité. En effet, même si l'on pensait que le comportement par défaut du système de stockage pourrait être assez bon, il est généralement préférable de lire ou d'écrire des données de / vers le nœud le plus proche du stockage du système à partir d'un noeud avec un coût de réseau élevé.

Un exemple de ce système de stockage est le système de fichiers distribués Hadoop

(HDFS) et Tachyon.

3.5.3.1 Architecture de HDFS

HDFS a une architecture du type maitre/esclave. Un cluster HDFS est constitué d'un maitre unique appelé NameNode qui gère l'espace de noms du système de fichiers et règle l'accès aux fichiers par les clients (ouvrir, fermer, renommer, etc.), ainsi qu'un ensemble de DataNodes pour gérer le stockage réel de données [59].

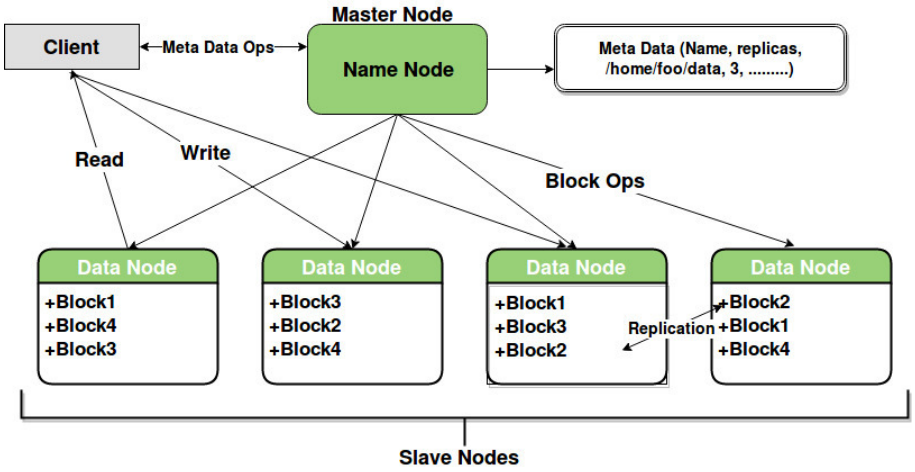

FIGURE 3.9 – L'architecture HDFS

3.5.3.2 Les composants de HDFS

HDFS est géré avec l'architecture maître-esclave, il inclut les composants suivants :

3.5.3.3 NameNode

Il n'y a qu'une seule instance de NameNode par cluster HDFS, c'est le maître du système HDFS. Il gère les blocs qui sont présents sur les DataNodes [59], cette tâche nécessite plusieurs fonctionnalités comme la gestion de l'espace de nommage, la gestion de l'arborescence du système de fichiers et le stockage des modifications des métadonnées (noms, permissions, etc.) des fichiers et répertoires [62], sans NameNode tous les fichiers peuvent être considérés comme perdus. Chaque fois qu'un client veut lire ou écrire une donnée, il communique avec le NameNode lequel répond à la demande en retournant l'emplacement des blocs de données ainsi que les informations du DataNode qui contient ces blocs. Les fonctions principales du NameNode sont :

— Contrôler l'exécution des opérations d'E/S de bas niveau des DataNodes.
— Choisir de nouveaux DataNodes pour de nouvelles réplications de blocs de données, en cas de défaillance d'un DataNode dans le processus d'écriture .
— Enregistrer les métadonnées de tous les fichiers enregistrés dans le cluster, comme par exemple l'emplacement, la taille des fichiers, les permissions, la hiérarchie, etc.
— Gérer le facteur de réplication de tous les blocs.

3.5.3.4 Name Node secondaire

Afin d' assurer la fiabilité de HDFS, on trouve le NameNode secondaire. Ce dernier, c'est le responsable de l'exécution périodique des points de contrôle. Donc si le NameNode est indisponible, peut-être remplaçait par une image instantanée stockée par les points de contrôle de NameNode secondaires [59]. Alors sa fonction principale est de lire en permanence tous les fichiers et métadonnées à partir de la RAM du NameNode et les écrits sur son disque pour rendre ces informations accessibles en cas d'échec dans le fonctionnement du NameNode.

3.5.3.5 Data Nodes

Ce sont des esclaves qui sont déployés sur chaque machine et fournissent le stockage réel. Ils sont chargés de servir les demandes de données de lecture et d'écriture pour les clients. Les DataNodes sont sous les ordres du NameNode et sont surnommés les Workers. Ils sont donc sollicités par les Name Nodes lors des opérations de lecture et d'écriture. En lecture, les Data Nodes vont transmettre au client les blocs correspondant au fichier demandé. En écriture, les Data Nodes vont retourner les données de chaque bloc, pour cela ils ont plusieurs fonctionnalités comme [62] :

— Envoyer régulièrement un rapport sur tous les blocs présents dans le cluster au NameNode.
— Un DataNode effectuer les opérations de lecture et d'écriture de bas-niveau.
— Il est responsable de la création de blocs et un canal de communication avec les autres DataNodes.
— Stocker chaque bloc de données HDFS dans des fichiers séparés dans son système de fichiers local.

3.5.3.6 Les opérations de HDFS

Comme la plupart des systèmes de fichiers classiques, HDFS contient un nombre important d'opérations comme la lecture, l'écriture, la suppression des fichiers, et d'autres

opérations pour créer et supprimer des répertoires. Le NameNode est le responsable de la gestion des métadonnées pour les fichiers et les répertoires, il expose un espace de noms qui permet de stocker les données sur un cluster de nœuds. Le NameNode gère toutes les opérations telles que (ouvrir, fermer, renommer ou déplacer un fichier ou un répertoire). Les DataNodes sont les responsables de servir les données de fichiers réels.

Une caractéristique importante dans l'HDFS : lorsqu'un client demande ou envoie des données, ces dernières ne passent pas physiquement à travers NameNode, ce serait un énorme goulot d'étranglement. Au lieu de cela, le client obtient tout simplement les métadonnées sur le fichier du NameNode et récupère les blocs de fichiers directement à partir des nœuds.

3.5.3.7 Le processus de lecture d'un fichier sur HDFS

Ce processus commence par l'interrogation entre le client et le NameNode. L'identité du client est d'abord validée en utilisant un mécanisme d'authentification ; ensuite le NameNode confirme si le client a le droit d'accès pour la lecture de ce fichier ; enfin pour chaque bloc de ficher le NameNode renvoie l'adresse du DataNode le plus proche possédant une copie du bloc (Figure 3.10) [2].

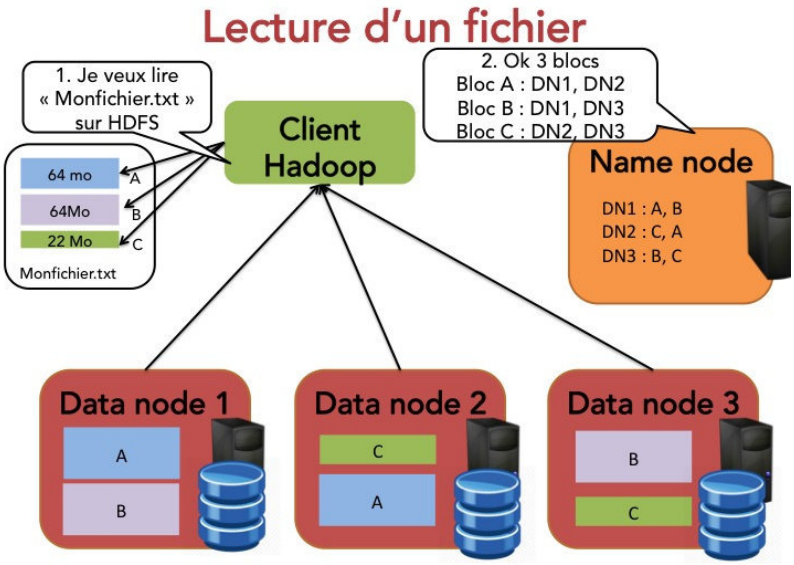

FIGURE 3.10 – Le processus de lecture d'un fichier sur HDFS [2]

3.5.3.8 Le processus d'écriture d'un fichier sur HDFS

Dans ce processus le client contacte le NameNode, ce dernier crée un fichier dans la hiérarchie du système de fichiers HDFS et informe le client sur l'identificateur de bloc

et l'emplacement des DataNodes. Le client utilise ces informations pour faire une écriture réelle dans les DataNodes concernés (le processus de réplication est fait) ; chaque DataNode informe le NameNode par la création d'un fichier réel sur son système de stockage local si l'opération est réussie (Figure 3.12) [2].

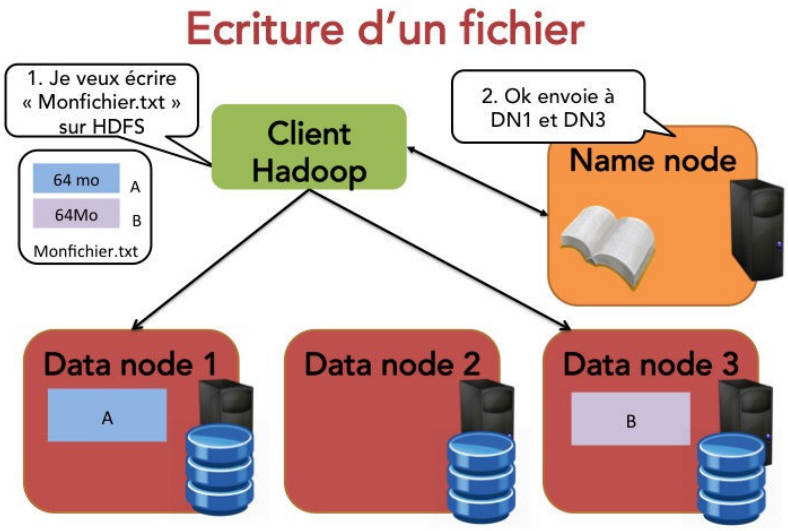

FIGURE 3.11 – Le processus d'écriture d'un fichier sur HDFS [5]

3.5.3.9 La gestion des blocs HDFS

Il faut comprendre comment les fichiers sont stockés physiquement dans le cluster. Dans Hadoop, chaque fichier est divisé en multiples blocs. Une taille typique de bloc est de 64 Mo, mais il n'est pas typique pour configurer la taille des blocs de 32 Mo ou 128 Mo. Les tailles de bloc peuvent être configurées par fichier dans le HDFS [62]. Si le fichier n'est pas un multiple exact de la taille du bloc, l'espace n'est pas gaspillée et le dernier bloc est juste inférieure à la taille totale du bloc.

— Un grand fichier sera divisé en plusieurs blocs. Chaque bloc est stocké sur un DataNode. Il est également répliqué pour s'assurer contre l'échec.

— HDFS permet également de définir le facteur de réplication d'un fichier. Par défaut le facteur de réplication d'un fichier est de trois, il améliore la tolérance aux pannes et réduit les défauts système et augmente la bande passante de lecture

3.5.4 Définition de Tachyon

Tachyon est un système de stockage distribué centré sur la mémoire, qui permet aux utilisateurs de partager des données à travers les plateformes et d'exécuter des

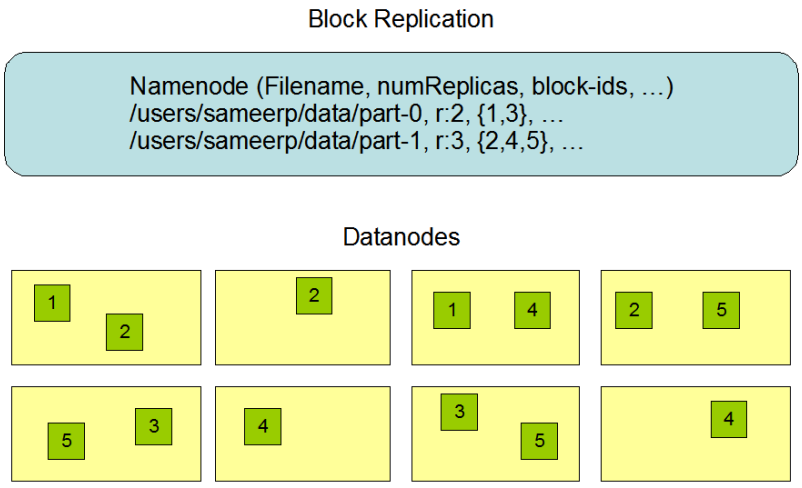

FIGURE 3.12 – Le processus de réplication de blocs [3]

actions de lecture/écriture à la vitesse de la mémoire à travers les plateformes de traitement en Cluster. Il permet aussi d'obtenir un débit d'écriture de 110x supérieur que dans la mémoire HDFS [63]. Afin d'assurer la tolérance aux pannes, la sortie perdue est récupérée en exécutant à nouveau les opérations qui ont créé la sortie, appelée lignage (lineage) [63]. Ainsi, l'option de lignage de Tachyon est considérée comme un défi majeur à Tachyon, et la couche de lignage fournit des E/S à haut débit et suit la séquence de travaux et le lignage de données dans la couche de stockage.

3.5.4.1 Architecture de Tachyon

Tachyon utilise une architecture master-slave standard similaire à HDFS (voir la figure 3.13), cette architecture s'appelle master-worker.

Le master gère les métadonnées et contient un gestionnaire de flux de travail, ce dernier interagit avec un gestionnaire de ressources de cluster pour allouer des ressources et recalculer. Tandis que, les workers gèrent les ressources locales et rapportent l'état au maître, et chaque worker utilise un RAMdisk pour stocker des fichiers mappés en mémoire.

3.5.4.2 Les composants de tachyon

La conception de Tachyon utilise un seul maître et plusieurs travailleurs. Tachyon peut être divisé en trois composantes, le maître, les travailleurs et les clients. Le maître et les travailleurs constituent ensemble les serveurs Tachyon, qui sont les composants qu'un administrateur système maintiendrait et gérerait. Les clients sont généralement les applications, telles que Spark ou MapReduce, ou les utilisateurs de la ligne de com-

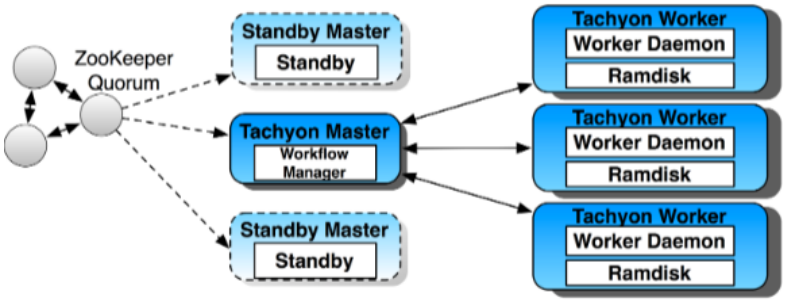

FIGURE 3.13 – L'archieture de Tachyon [6]

mande Tachyon. Donc, les utilisateurs de Tachyon ne doivent habituellement qu'interagir avec la partie client de Tachyon [63].

i. Maître (Master) : Tachyon peut être déployé dans l'un des deux modes principaux, un maître unique ou plusieurs maîtres. Le maître est principalement responsable de la gestion des métadonnées globale du système, par exemple, l'arborescence du système de fichiers. Les clients peuvent interagir avec le maître pour lire ou modifier ces métadonnées. En outre, tous les travailleurs interrogent périodiquement le maître pour maintenir leur participation au cluster. Le maître ne lance pas de communication avec les composants ; il interagit uniquement avec les composants en répondant aux demandes.

ii. Travailleurs (worker) : Les travailleurs de Tachyon sont responsables de la gestion des ressources locales allouées à Tachyon. Ces ressources pourraient être la mémoire locale, le SSD ou le disque dur et ils sont configurables par l'utilisateur. Les travailleurs de Tachyon stockent les données en tant que blocs et répondent aux demandes des clients de lire ou d'écrire des données en lisant ou en créant de nouveaux blocs. Toutefois, le travailleur n'est responsable que des données dans ces blocs.

iii. Client : Le client Tachyon offre aux utilisateurs une passerelle pour interagir avec les serveurs Tachyon. Il lance la communication avec le maître pour mener à bien les opérations de métadonnées et avec les travailleurs pour la lecture et l'écriture des données.

3.6 L'infrastructure distribuée

Toutes les plates-formes présentées précédemment ont besoin d'un logiciel pour les rendre faciles à utiliser comme un ensemble de noeuds cohérent. Cela peut se faire soit comme un système d'exploitation, soit comme un logiciel de gestion de niveau supérieur [64].

3.6.1 Les grappes Cluster

On peut définir un cluster comme un ensemble de noeuds connectés entre eux, où l'ensemble homogène de noeuds qui font le cluster peut-être utilisé de deux manières différentes. L'une des façons est d'utiliser un système d'exploitation de cluster qui fait apparaître le cluster en tant que système unique. L'autre façon est d'utiliser un gestionnaire de ressources qui permet d'attribuer un ensemble de noyaux (cores) ou de nœuds pendant un certain temps [64].

Une grappe de machine (Cluster en Anglais) désigne un ensemble d'ordinateurs, appelés nœuds, tous inter-connectés, dans le but de partager des ressources informatiques. Une grappe peut être constituée d'ordinateurs de bureau, de "racks" de machines constituées de composants standards ou de "lames" également constituées de composants standards afin d'optimiser l'espace physique. Une grappe est généralement composée de machines homogènes en termes d'architecture et de système d'exploitation. Elle ne regroupe que des machines appartenant au même domaine d'administration réseau et les nœuds communiquent entre eux en utilisant un réseau de communication rapide. Les différents nœuds d'une grappe possèdent souvent une configuration logicielle semblable [65].

3.6.2 Les grilles

Le terme "grille" désigne un ensemble beaucoup plus important de machines, hétérogènes, réparties sur différents domaines d'administration réseau . Les différentes entités composant une grille peuvent être réparties sur l'ensemble de la planète et communiquent entre elles en utilisant une grande diversité de réseaux allant de l'Internet à des réseaux privés très haut débit.

Une grille informatique est une infrastructure matérielle et logicielle qui fournit un accès consistant à des ressources informatiques [66]. Le but est ainsi de fédérer des ressources provenant de diverses organisations désirant collaborer en vue de faire bénéficier aux utilisateurs une capacité de calcul et de stockage qu'une seule machine ne peut fournir. Cependant, tout système informatique distribué ne peut posséder l'appellation de grille [67].

En effet, une grille est un système qui coordonne des ressources non soumises à un contrôle centralisé, qui utilise des protocoles et interfaces standards dans le but de délivrer une certaine qualité de service (en termes de temps de réponse ou bien de fiabilité par exemple). Plusieurs types de grilles peuvent être discernées selon l'utilisation recherchée :

— Grille d'information : la ressource partagée est la connaissance. L'Internet en est le meilleur exemple : un grand nombre de machines hétérogènes réparties sur toute la surface du globe autorisant un accès transparent à l'information.

— Grille de stockage : l'objectif de ces grilles est de mettre à disposition un grand nombre de ressources de stockage d'information afin de réaliser l'équivalent d'un "super disque dur" de plusieurs PetaBytes. Le projet DataGrid ou les réseaux x Kaaza ou Gnutella sont un bon exemple de grille de stockage.

— Grille de calcul : l'objectif de ces grilles est clairement d'agréger la puissance de traitement de chaque nœud de la grille afin d'offrir une puissance de calcul "illimitée" (exemple : grid5000).

3.6.3 Les nuages

Le concept de nuage ou "cloud" [68] est une évolution de la notion de grille. ll se focalise plus sur les fournisseurs des ressources qui mettront à disposition la puissance de calcul et de stockage avec leur propre technologie. La définition d'utilisateurs et de fournisseurs permet de créer une relation claire entre ces deux types d'entités. On pourra alors associer des garanties de puissance fournie ou de quantité de stockage par exemple. Trois modèles peuvent être utilisés sur un nuage [64] :

— IaaS (Infrastructure as a service) : le fournisseur met à disposition uniquement des ressources matérielles (machine, réseau, disque), l'utilisateur gère le système, les intergiciels et les applications.

— PaaS (Platform as a service) : l'utilisateur ne gère et ne contrôle ici que ses applications métier, le reste est délégué au fournisseur de service.

— SaaS (Software as a service) : c'est l'ultime niveau de transfert vers le fournisseur, l'utilisateur ne gère plus que ses données métier, il utilise les applications fournies par le fournisseur. Il s'agit d'utiliser le logiciel à la demande.

3.7 Conclusion

Dans ce chapitre, nous avons introduit les concepts de base de Big Data et ses technologies, permettant de dépasser des outils classiques, qui ne parviennent pas à traiter ces énormes volumes de données, et aller vers des outils modernes, qui font face à la grande taille de données.

Plusieurs outils ont été cités et introduits dans ce chapitre, à savoir les plateformes de stockage comme HDFS et Tachyon et les plateformes de traitement comme Hadoop MapReduce et Spark. La plateforme Spark est une alternative à MapReduce permet aux développeurs de simplifier la complexité du code MapReduce et écrire des programmes et des requêtes d'analyse de données en Java, Scala ou Python en utilisant le même système de stockage Hadoop HDFS ou Tachyon pour traiter d'énormes quantités de données « Big Data » et d'accélérer leurs temps de traitement, car elle supprime le temps de lecture/écriture de disque grâce à l'utilisation de structure de données distribué RDD, permettant aux programmes d'effectuer des calculs en mémoire.

Nous allons revenir sur ces plateformes dans les chapitres suivants afin d'accentuer sur leurs utilisations et spécificités dans le cas des bases d'images à grande échelle.

Chapitre 4

Un système parallèle de recherche d'images par le contenu basé sur les plateformes Spark et Tachyon

4.1 Introduction

Récemment, un accroissement extrêmement rapide des images multimédias générées sur Internet dans les différents domaines. Par conséquent, le dévelopement d'un système de recherche d'images par le contenu (CBIR) pour les bases images à grande échelle (big data) est devenu un grand défi, à cause de l'un des inconvénients associés aux systèmes CBIRs est le très longtemps d'exécution.
Dans ce cadre, nous proposons un système de recherche d'images par le contenu rapide basé sur la plateforme Spark (CBIR-S) tout en ciblant les bases d'images à grande échelle (Big Data). En effet, notre système est composé de deux étapes :
(i) L'étape d'indexation d'images, dans laquelle nous utilisons le modèle distribué MapReduce sous la plateforme Spark afin d'accélérer le processus d'indexation. Nous utilisons également un système de stockage distribué, appelé Tachyon ;
(ii) L'étape de recherche d'image, dans laquelle nous accélérons le processus de recherche en utilisant la méthode parallèle de K-plus poches voisins (KNN) de classification basée sur le modèle MapReduce implémenté sous Apache Spark. De plus, nous exploitons la méthode de cache du framework spark.
Nous allons montrer à travers une large série d'expériences, l'efficacité de notre approche en termes de temps de traitement.

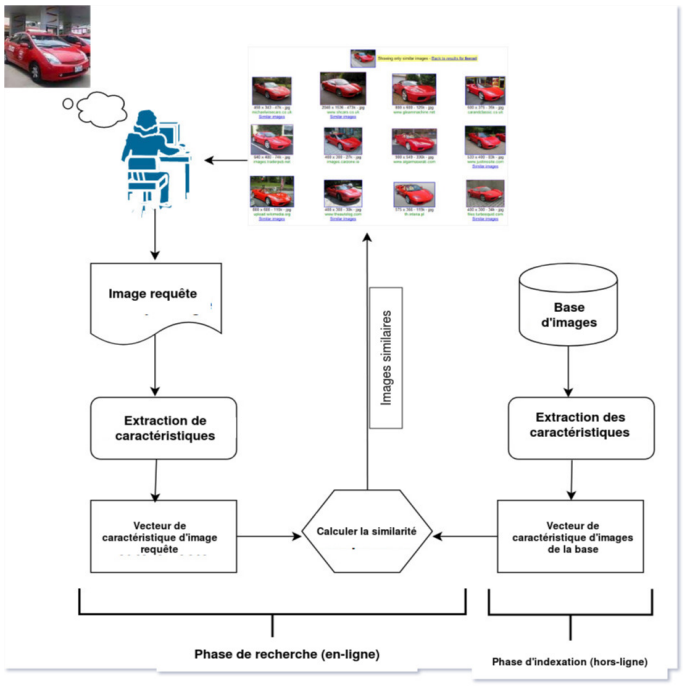

FIGURE 4.1 – L'architecture typique de système de recherche des images par le contenu.

4.2 Motivation

L'utilisation croissante des appareils mobiles, tels que les appareils photo et les téléphones intelligents ont entraîné une augmentation considérable de la quantité d'images collectées chaque jour. Par conséquent, la récupération et la gestion de ces grands volumes des images sont devenu un défi majeur dans le domaine de vision par ordinateur. L'une des solutions pour gérer efficacement les bases de données d'images, c'est un système de recherche d'images par le contenu (CBIR) [69].
En général, un système de recherche d'images par le contenu (CBIR) typique peut être décomposé en deux étapes : l'indexation et la recherche :
(i) l'étape d'indexation est chargée d'extraire les vecteurs caractéristiques appropriés des images et de les sauvegarder dans un support de stockage.
(ii) l'étape de recherche, dans laquelle le vecteur de caractéristiques d'une image de requête est calculé et comparé aux vecteurs de caractéristiques d'images de la base de données. Par conséquent, le système CBIR renvoie les images les plus proches à la requête de l'utilisateur [70]. Fig. 4.1 présente l'architecture d'un système de recherche d'images par le contenu.
En fait, le temps de calcul élevé causé par la complexité et la quantité colossale des descripteurs générés pendant l'étape d'indexation, et l'étape de recherche sont devenus des obstacles devant la construction d'un système CBIR [71] : Par conséquent, dans ce chapitre, nous proposons un système CBIR basé sur la plateforme Spark à réponse rapide pour la recherche d'images dans les bases d'images à grande échelle. Nous ap-

pellerons cette approche de recherche d'image basée sur le contenu en utilisant la plateforme Spark (CBIR-S), notre travail consiste en deux contributions :

— Afin d'accélérer l'étape d'indexation des bases d'images à grande échelle, nous utilisons d'abord le système de stockage distribué Tachyon au lieu du système de fichiers distribués Hadoop (HDFS) [72], ce dernier étant environ 110 fois plus rapide que HDFS [63]. De plus, nous parallélisons l'étape d'indexation à l'aide du modèle MapReduce sous Spark.

— Dans le module de recherche, nous utilisons la technique de recherche k-plus proche voisin (k-NN) [73], pour comparer le vecteur de caractéristiques de l'image de requête et les vecteurs de caractéristiques stockés dans la base d'images. Cependant, l'algorithme k-NN pose deux problèmes principaux lorsqu'il s'agit de traiter des données à grande échelle : le temps d'exécution et la consommation de mémoire [74]. Par conséquent, nous utilisons Apache Spark, qui fournit un environnement simple, transparent et efficace pour paralléliser l'algorithme k-NN comme un processus MapReduce itératif. Par conséquent, le point principal de notre suggestion est d'exploiter la méthode puissante cache/persist de Spark, qui permet de stocker en mémoire les distances calculées pour un calcul rapide.

Ce chapitre est organisé comme suit. Section 4.3 décrit les études antérieures en lien avec les problématiques de notre travail. Dans la section 4.4, nous présentons les technologies des big data utilisées dans notre approche et les informations de base sur l'algorithme k-NN. La section 4.5 présente notre système CBIR basé sur la plateforme Spark (CBIR-S) et la section 4.6 décrit les résultats expérimentaux. Enfin, des conclusions et des idées pour de futures extensions de ce travail sont données à la fin de ce chapitre.

4.3 Travaux antérieurs

Dans cette section, nous présentons brièvement quelques travaux en lien avec le développement des systèmes CBIR. Nous commençons par les premiers CBIRs systèmes aux plus récents, à partir de 1990 la technique CBIR a commencé [75]. Depuis cette période, un ensemble de systèmes CBIRs sont apparus, tels que QBIC [76], Virage [77], MARS [78], TinEye [1],etc. Cependant, avec l'augmentation de la production d'images produites tous les jours, les systèmes conventionnels de CBIR sont devenus si gourmands en consommation de temps, ce qui a conduit à la création de systèmes parallèles pour la recherche d'images dans les bases d'images à grande échelle. Dans la suite, nous donnons un bref aperçu sur certains travaux dans ce domaine.
Zhang et al. dans [79] présente une implémentation d'un système DIRS (Distributed Image Retrieval System) déployé sur un environnement informatique distribué Hadoop. À l'étape d'indexation du système DIRS, les auteurs adoptent la base de données

1. Idée Inc . : http : //www.tineye. com /

distribuée HBase comme une couche de stockage. En outre, ils ont utilisé le modèle informatique Hadoop de MapReduce pour améliorer les performances de l'étape de recherche dans les grandes bases d'images. Gu et al. [71] ont proposé un système CBIR basé sur les frameworks Hadoop, HBase et Lucene. Raju et al. [80] ont développé un système CBIR basé sur Hadoop MapReduce. Ils ont utilisé les motifs de tétra locaux (LTPr) [81] comme une méthode de vecteur de caractéristiques pour représenter les images dans le système CBIR. Luca et al. [82] ont proposé un nouveau système de récupération d'images basé sur Hadoop et par Spark pour les bases d'images à grande échelle. Dans leur travail, les auteurs utilisent Hadoop dans la phase d'indexation et Spark dans l'étape de recherche. Sakr et al. [83] ont utilisé une technique efficace Hadoop MapReduce pour rechercher les images les plus proches pour l'image de la requête. Duan et al. [84] ont proposé un système CBIR amélioré basé sur Apache Spark. Ils ont utilisé Avro Framework pour combiner les fichiers image, un fichier Avro pouvant rester en mémoire afin de permettre aux actions futures d'être beaucoup plus rapides. Dernièrement, Lagiewka et al. [85] ont introduit un système CBIR distribué dans des bases de données relationnelles. Leur système est composé d'un ensemble de machines, chacune utilisant le framework Apache Hadoop avec HDFS.

Outre les systèmes parallèles et distribués cités ci-dessus, de nombreuses autres méthodes ont été tentées par les recherches pour optimiser la méthode de calcul k-NN, en particulier dans le Big Data. Dong et al. dans [86], proposent la méthode actuelle NN-Descent, qui consiste en une construction simple et efficace de graphe K-NNG (K-Nearest Neighbor Graph) avec des mesures de similarité arbitraires pour les applications à grande échelle, où les structures de données doivent être réparties sur le réseau. Song et al. [87] ont introduit une méthode k-NN distribuée pour les données volumineuses en utilisant le modèle de programmation MapReduce. Maillo et al. [88] ont proposé un algorithme k-NN parallèle basé sur le modèle de programmation MapReduce pour la classification des données à grande échelle, notée MR-kNN. Les auteurs de [89] ont mis au point un algorithme personnalisé K-Nearest Neighbor utilisant Hadoop pour déployer des applications de manière parallèle sur un cluster.

4.4 Préliminaires

Dans cette section nous présentons les concepts indispensables pour le reste du chapitre. La section 4.4.1 présente le modèle de programmation MapReduce et le framework Spark, section 4.4.2 présente le système de stockage distribué appelé Tachyon. Enfin, Section 4.4.3 définit formellement l'algorithme KNN et ses points faibles en terme de temps et mémoire pour traiter les problèmes de big data.

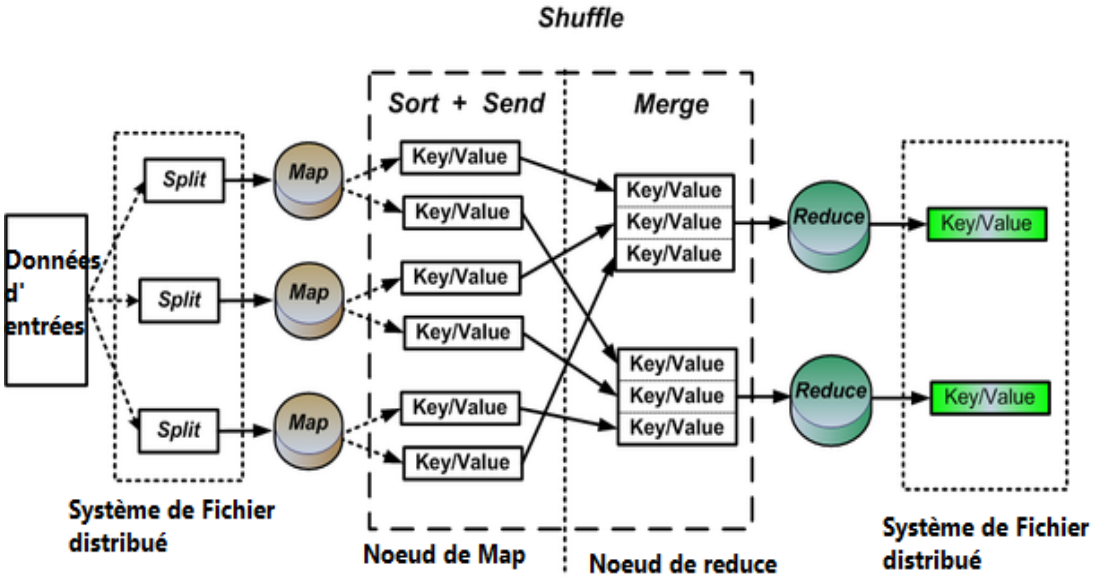

FIGURE 4.2 – L'architecture de modèle de programmation MapReduce [4]

4.4.1 Modèle de programmation MapReduce et le framework Spark

MapReduce est un modèle de programmation développé par Google pour fournir des améliorations significatives dans les applications de données à grande échelle dans les clusters [90]. Ce modèle de programmation peut rendre le schéma de parallélisme transparent aux utilisateurs et simplifier la programmation parallèle sur les clusters à grande échelle.
Généralement, les programmes implémentés en utilisant le modèle MapReduce requièrent la définition de deux éléments de base : les fonctions de mappers et de reducers.
Tout d'abord, la fonction Map a une paire (clé /valeur) en entrée et produit un ensemble de paires (clé/valeur) intermédiaires tel que : (k1, v1) → list (k2, v2).
Dans la phase de reducer, les paires (clé, valeur) sont utilisées comme données d'entrée pour le reducer. Ensuite, le reducer collecte toutes les paires (clé, valeur). Ainsi, les valeurs correspondant à la même clé sont additionnées comme suit :
(k2, liste (v2)) → list (v2). Ce processus peut être schématisé comme le montre dans la figure .4.2.

Apache Spark est l'une des plateformes les plus récentes, basée sur le modèle de programmation MapReduce pour contenir des données à grande échelle sur les clusters. Spark propose un ensemble d'outils, à savoir Spark SQL for SQL et le traitement de données structuré, Spark MLlib pour l'apprentissage automatique, GraphX pour le traitement graphique, et Spark Streaming [91]. Les unités de données centrales de Spark s'appellent Résidences Distribuées (RDD), elles sont une abstraction de la mémoire, qui regroupe un ensemble d'éléments dans lequel on peut appliquer un ensemble d'opérations pour obtenir un autre RDD (transformations) ou revenir en ar-

rière. En effet, les RDDs peuvent résider en mémoire, en disque ou en combinaison. Cependant, lorsque nous réutilisons des RDD plusieurs fois, une méthode puissante de cache/persistance [91] deviennent nécessaire dans le système mis en place pour éviter le re-calcul des RDD et effectuer moins de calculs.

4.4.2 Tachyon

Tachyon est un système de stockage distribué centré sur la mémoire, qui permet aux utilisateurs de partager des données à travers les frameworks et d'exécuter des actions de lecture/écriture à la vitesse de la mémoire en cluster [63]. De plus, Tachyon atteint un débit d'écriture 110 fois supérieur à celui de la mémoire HDFS [63]. En effet, Tachyon utilise une architecture maître-esclave standard similaire à Hadoop Distributed File System (HDFS) [72] (voir chapitre 3. section 3.5.4).

4.4.3 l'algorithme KNN et ses faiblesses pour traiter les big data

Cette section présente la méthode de recherche K-plus proches voisins (k-NN) :

Soit $Tr(d_i)$ l'ensemble de données d'apprentissage, où $i \in \{1,2,...,n\}$. $Ts(d_j)$ est l'ensemble de données de test et K le nombre de points les plus proches d'un $Ts(d_j)$, où $j \in \{1,2,...,m\}$, le processus de classification utilisant l'algorithme k-NN est composé de deux étapes :

— *L'étape d'apprentissage*, qui sert à enregistrer les vecteurs d'entités et les étiquettes de classe dans le système de stockage.

— *L'étape de classification*, qui permet de rechercher les k échantillons les plus proches dans le jeu de données $Tr(d_i)$. Il commence par calculer les distances entre $Ts(d_j)$ et tous les échantillons de $Tr(d_i)$ à l'aide d'une métrique de distance. Les données d'entraînement sont ensuite triées en fonction des distances calculées et les points les plus proches de $Ts(d_j)$ sont affectés à la classe la plus proche.

En fait, l'algorithme k-NN donne de bons résultats pour traiter un vaste ensemble de problèmes. Cependant, l'algorithme k-NN souffre encore de quelques problèmes pour traiter le big data, qui sont :
- Le temps d'exécution : l'algorithme k-NN calcule les distances entre l'ensemble de données d'apprentissage et chaque requête de test, puis il trie les distances calculées et prend les K plus proche voisins à la requête donnée. Ce processus est répété pour chaque requête de test. Ces calculs itératifs prennent beaucoup de temps.
-La consommation de mémoire : pour un tri rapide des données d'apprentissage en fonction de leurs distances, l'algorithme k-NN nécessite de sauvegarder les données d'apprentissage et les distances calculées en mémoire.

Ces faiblesses ont conduit à paralléliser l'algorithme k-NN et de distribuer ses calculs sur les nœuds du cluster [74].

4.5 L'architecture de l'approche proposée

Dans cette section, nous présentons notre système de récupération d'images basée sur le contenu utilisant Spark (CBIR-S), qui cible des bases d'images à grande échelle. Le nouveau système utilise le système de stockage distribué Tachyon pour un stockage rapide des vecteurs de caractéristiques d'images. Nous introduisons également un algorithme parallèle de k-NN avec la méthode cache pour effectuer la partie la plus fastidieuse du travail, qui est le calcul des distances entre le vecteur de caractéristique de requête et les vecteurs de caractéristique de base de données correspondants. Enfin, la structure exploite les fonctionnalités de Spark et MapReduce pour accélérer les étapes d'indexation et de recherche.

Dans notre système CBIR-S, nous connectons deux frameworks, Spark et Tachyon, où Spark est un framework basé sur MapReduce utilisé pour stocker des données à grande échelle sur des clusters, et Tachyon est un système de stockage distribué basé sur la mémoire, qui permet le partage des images et l'exécution des actions de lecture/écriture à la vitesse de la mémoire. Tachyon se déploie sur un cluster de mode standalone. Ensuite, pour connecter Tachyon à nos applications Spark, il suffit d'appeler les API de chargement/enregistrement de DataFrame et de RDD et de spécifier le chemin de l'URL comprit le protocole Tachyon. Notre système CBIR-S proposé est composé de deux modules :

1. *Le module d'indexation*, qui est une couche hors ligne comprenant deux composants dédiés à l'exécution des travaux qui ne nécessitent pas d'interaction avec l'utilisateur. Le module d'indexation proposé contribue à :
   - — Accélérer l'étape d'indexation pour les bases d'images à grande échelle en utilisant la parallélisation MapReduce sous Spark.
   - — Permettre de diminuer du temps nécessaire au stockage des vecteurs de caractéristiques d'images en utilisant le système de stockage Tachyon, ce qui améliore le débit de l'action d'écriture (sauvegarde) et permet d'atteindre un débit d'écriture 110 fois plus rapide que celui de HDFS [63].
2. *Le module de recherche*, qui est une couche en ligne, inclut les étapes dédiées à adresser l'interaction entre le vecteur de requête et les vecteurs de caractéristiques de la base de données (représentés dans le fichier d'entrée Tachyon créé lors du module précédent). Comme indiqué dans la Fig. 4.4. L'étape de recherche proposée contribue à réduire le temps consommé en utilisant l'algorithme parallèle k-NN sous Spark pour trouver l'image la plus proche à l'image requête. Dans ce module, nous exploitons la fonctionnalité de Spark, qui est la méthode cache/persist.

4.5.1 L'étape d'indexation

Comme le montre la figure 4.3, la structure de l'étape d'indexation comprend le module de représentation d'image et le système de stockage distribué Tachyon. Les objectifs de cette étape sont :

— Chargement des images d'entrée à partir du système de stockage Tachyon pour effectuer le traitement requis pour extraire les vecteurs de caractéristiques CS-LBP d'images [92].

— Sauvegarde des vecteurs extraits dans le système de stockage distribué Tachyon.

Pour atteindre ces objectifs, nous utilisons la méthode de traitement distribué, qui est le modèle de programmation MapReduce sous Spark.

4.5.1.1 Le module de représentation d'image

Le premier module de notre approche proposée est constitué de deux composants : le premier pour des prétraitements chargés de convertir l'image couleur en image en niveaux de gris et de redimensionner l'image, le second pour l'extraction des caractéristiques, qui fonctionne comme suit :
Les caractéristiques sont extraites par la méthode CS-LBP qui est conçue pour avoir une plus grande stabilité dans les régions d'image plates. Dans la méthode CS-LBP, les valeurs de pixel ne sont pas comparées au pixel central, mais au pixels opposés symétriquement par rapport au pixel central. De plus, la robustesse sur les régions d'images plates est obtenue en limitant les différences de niveau de gris avec une petite valeur T comme proposé dans [92] :

$$CS-LBP_{R,N,T}(x,y) = \sum_{i=0}^{(N/2)-1} s(n_i - n_{i+(N/2)})2^i \tag{4.1}$$

, où la valeur de s(x) est égale à :

$$s(x) = \begin{cases} 1 & x >= 0 \\ 0 & sinon \end{cases} \tag{4.2}$$

Dans CS-LBP, les valeurs de pixels ne sont pas comparées au pixel central mais aux N pixels opposés symétriquement, espacés sur un cercle de rayon R par rapport à celui du centre. Les valeurs de n_i et $n_{i+(N/2)}$ correspondent aux valeurs de gris des paires de pixels symétriques.

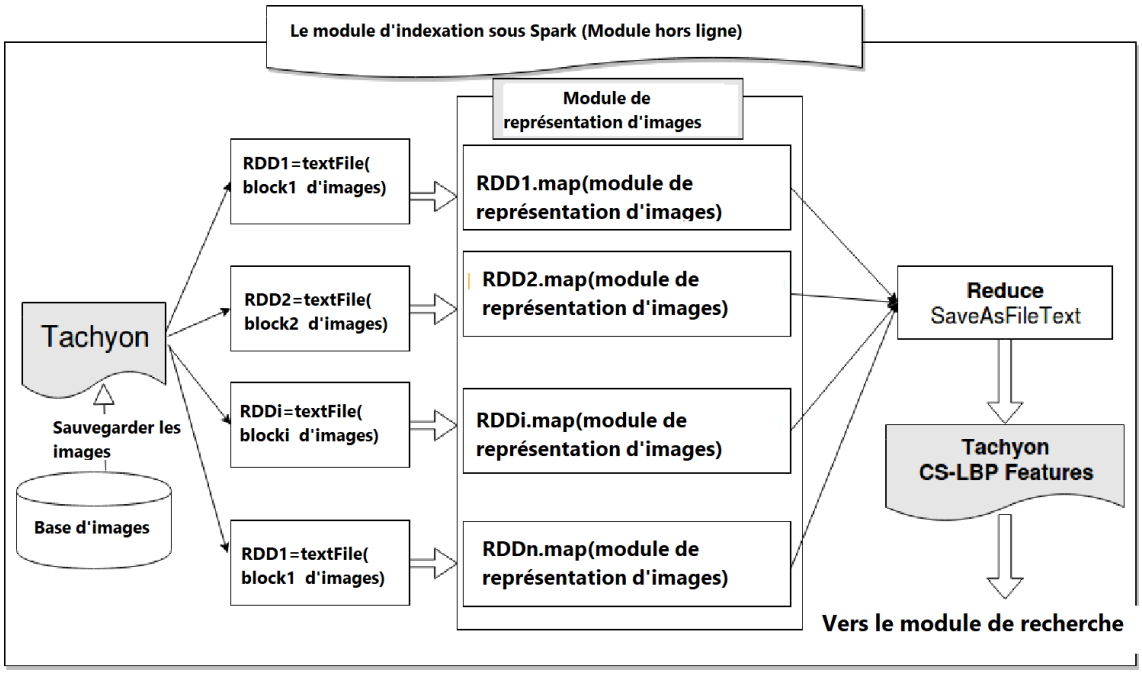

FIGURE 4.3 – L'architecture de l'étape d'indexation (phase off-line) du système CBIR-S : application de MapReduce dans le processus d'indexation d'image

4.5.1.2 MapReduce pour le module d'indexation dans Spark :

Nous avons utilisé le processus Spark MapReduce pour gérer l'indexation de sous-ensembles d'images, ils sont sauvegardés dans Tachyon. Ce processus permet à plusieurs map et reduce de diviser le calcul en deux phases principales :

— *La phase de map :* elle divise les images chargées à partir de Tachyon en blocs et calcule pour chaque bloc d'images les vecteurs caractéristiques.

— *La phase de reduce :* elle regroupe les vecteurs de caractéristique de chaque map et les sauvegarde dans Tachyon.

Le MapReduce job sous Spark est utilisé pour indexer les images en parallèle, comme suit :

Premièrement, nous chargeons les images du système Tachyon dans le RDDi d'images : $< ID\ image, T >$, où T est le chemin de l'image. Ensuite, dans la phase de map, nous map les images pour calculer leurs vecteurs de caractéristiques sur chaque bloc de RDDi. À la suite de ce map, nous obtenons un RDDi $< ID\ image, feature\ vecteur >$ des images. Enfin, la phase de reduce consiste à collecter les vecteurs d'entités fournis par les maps et à les enregistrer dans Tachyon ; en utilisant l'action de sauvegarde (save action).

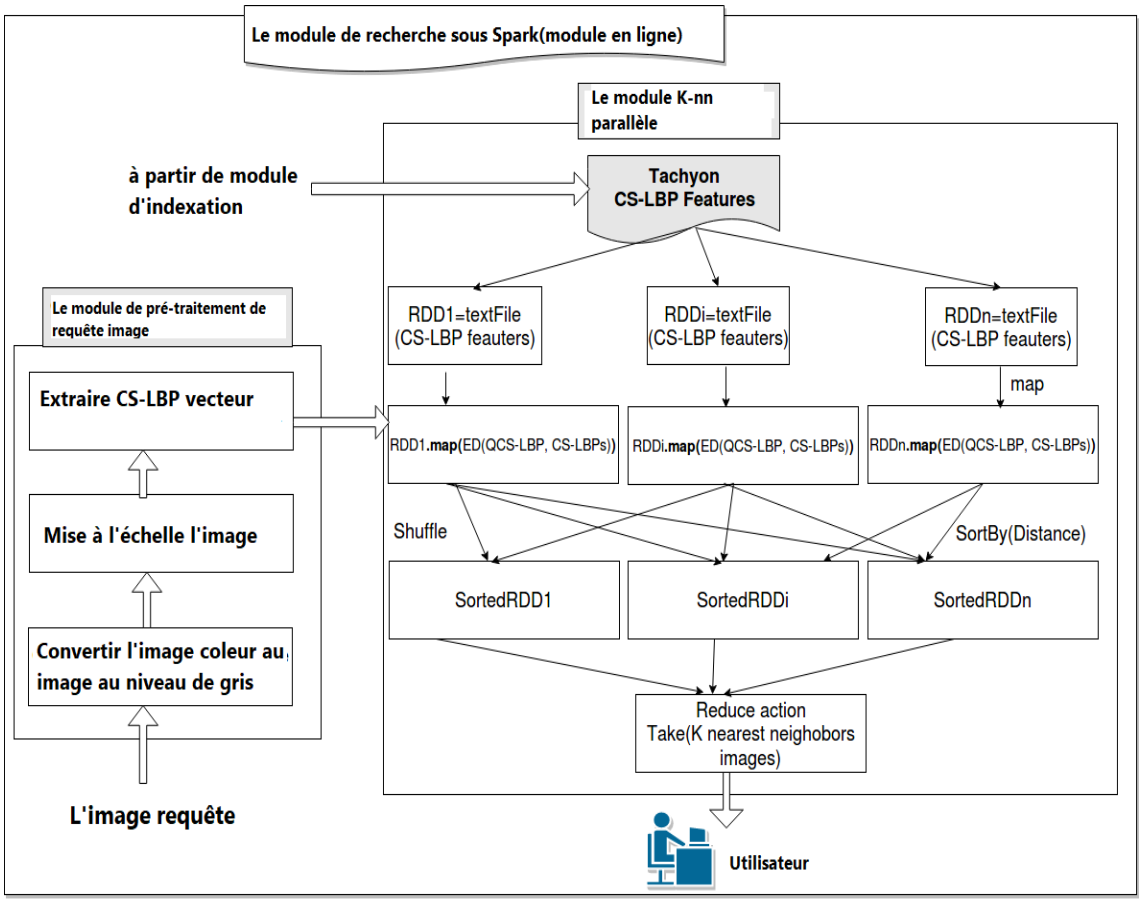

FIGURE 4.4 – L'architecture de l'étape de recherche (phase en ligne) du système CBIR-S : application de MapReduce dans le processus de recherche d'image

4.5.2 L'étape de recherche

L'étape de recherche en ligne consiste en deux étapes : le prétraitement d'image requête et la classification parallèle, comme indiqué dans la Fig. 4.4.

4.5.2.1 La phase de recherche en utilisant k-NN parallèle

Dans ce module, nous allons nous concentrer sur la réduction du temps de calcul de l'étape de recherche, en implémentant un module k-NN parallèle sous Spark.
En effet, le module k-NN parallèle a pour but de calculer la similarité entre le vecteur de caractéristique de l'image requête et les vecteurs de caractéristiques d'images d'apprentissage correspondants. Cette tâche est l'étape qui consomme plus de temps dans notre approche proposée. Afin de diminuer le temps de calcul de cette étape, nous avons développé une implémentation parallèle de k-NN pour l'étape de recherche sous la plateforme Spark en utilisant le modèle de programmation MapReduce et la

méthode cache/persist de spark.
L'algorithme 1 montre le pseudo-code de k-NN parallèle de la phase de recherche. Dans la section suivante, nous décrivons les instructions les plus significatives énumérées de (1 à 5). Tout d'abord, l'algorithme k-NN parallèle commence par charger les vecteurs de caractéristiques CS-LBPs d'apprentissage du système Tachyon dans RDDi à l'aide de l'instruction 1. Ensuite, l'instruction 2 map la distance euclidienne entre chaque vecteur d'apprentissage et le vecteur CS-LBP de requête. Après cela, l'instruction 3 trie les distances calculées. Enfin, l'algorithme parallèle k-NN renvoie les k images les plus proches de l'image de requête (instructions 4, 5). Dans cet algorithme, nous mettons en cache les distances calculées en mémoire, pour les utiliser par les actions de tri et de retour (instructions 1, 2).

4.5.2.2 MapReduce pour le k-NN parallèle sous Spark

Cette sous-section présente les travaux MapReduce sous Spark pour le k-NN parallèle de l'algorithme 1. L'algorithme 2 montre l'implémentation de l'algorithme 1 Parallel k-NN dans Apache Spark, qui est comme suit :

L'étape de map :
Dans l'étape de map, nous commençons par charger les vecteurs de caractéristiques CS-LBPs d'apprentissage du système Tachyon dans RDD (trainRDD) à l'aide de sc.textFile (instruction 10). Ensuite, l'instruction 11 crée un vecteur MLlib (vectorsRDD) pour chaque vecteur de caractéristiques CS-LBP. La première transformation zipWithIndex (instruction 12) attribue à chaque vecteur CS-LBP (vectorsRDD) un numéro de séquence. Ce numéro identifie le vecteur CS-LBP lorsqu'il est écrit sur le disque et traité.

Par la suite, nous chargeons les classes d'images du système Tachyon dans classRDD en utilisant l'instruction 14. La deuxième transformation zipWithIndex attribue à chaque classe d'image un numéro de séquence (instruction 15). L'instruction 17 joint les résultats de ces deux fichiers zipWithIndex dans classVector afin que nous puissions les afficher de manière plus lisible.

Algorithme 1 : Parallel k-NN algorithm.

Input : CS-LBPs les vecteurs de caractéristiques d'apprentissage CS-LBP, K : le nombre des K plus proche voisins images, QCS-LBP : vecteur de caractéristique de la rêquette.

Output : $y_1, y_2, \cdots, y_k$: K nplus proche voisins images à l'image requête

1 $RDDi \leftarrow textFile(CS\text{-}LBPs).cache$
2 $DistRDD \leftarrow RDDi.map(Distance(CS\text{-}LBPs, QS\text{-}LBP)).cache()$
3 $SortedRDDi \leftarrow DistRDD.sortBy(Distance)$
4 $SortedRDDi.take(K)$
5 $Return\ y_1, y_2, \cdots, y_k$

Enfin, nous calculons la distance euclidienne entre les vecteurs CS-LBPs de class-Vector et QCS-LBP (vecteur de caractéristique CS-LBP d'une image requête) à l'aide de la fonction euclideanDist définie ci-dessous (instruction 19). Ensuite, nous trions les distances calculées en utilisant l'action sortBy (instruction 21).

L'étape de reduce :

Dans cette étape, nous avons un RDD (résultat) qui contient un ensemble de paires (clé, valeur) pour chaque image. Ici, la clé est l'identificateur d'image numérique et la valeur est constituée d'un ensemble d'images et d'informations connexes. Les informations sur l'image que nous avons sont l'id de l'image, la classe et la distance du vecteur CS-LBP de l'image à partir de la requête QCS-LBP, dont laquelle nous trions les résultats en fonction de leurs distances par rapport à la requête QCS-LBP. Enfin, nous prenons les k voisins les plus proches de l'image de la requête à l'aide de l'instruction 23.

Méthode de cache : est une méthode d'optimisation pour les calculs Spark (itératifs et interactifs), prédéfinie par Spark, elle permet d'enregistrer les résultats partiels intermédiaires afin de pouvoir le réutiliser ultérieurement. Ces résultats intermédiaires en

tant que RDD sont donc conservés en mémoire (par défaut).

Algorithme 2 : Implémentation de l'algorithme parallèle k-NN sous Apache Spark.

```
1  begin
2  /*La création de SparkContext*/
3  val conf = new SparkConf().setAppName("Parallel knn")
4  .setMaster("spark://salihaInspiron3542:7077")
5  .set("spark.executor.memory", "800m")
6  .setJars(Seq("/usr/share/scala/project/target/scala2.10/parallel_knn_2.10-
     1.0.jar"))
7  List("target/scala2.10/parallel_knn_2.10−1.0.jar")
8  val sc = new SparkContext(conf)
9  /*L'étape de map*/
10 val trainRDD =
     sc.textFile("tachyon://localhost:19998/descriptors/CSLBPs").cache
11 val vectorsRDD = trainRDD.map(x =>
     Vectors.dense(x.split(',').map(_.toDouble)))
12 val withIndex = vectorsRDD.zipWithIndex
13 val indexvector = withIndex.map{case (k,v) => (v,k)}
14 val classRDD =
     sc.textFile("tachyon://localhost:19998/classes/image_class").cache
15 val withIndex1 = classRDD.zipWithIndex
16 val indexclass = withIndex1.map{ case (k,v) => (v,k)}
17 val classVector = (indexclass join indexvector)
18 val classDistRDD = classVector.map {case (id, (class,vector)) =>
19 val dist = euclideanDist(QCS_LBP, vector)
20 (id, class, dist)}.cache
21 val result = classDist.sortBy(_._3)
22 /*L'étape de reduce*/
23 result.take(K)
24  end;
```

4.6 Résultats expérimentaux

4.6.1 Environnement expérimental

(i) Mémoire installée (RAM) 8 Go, (ii) Processeur Intel Core i5-4210U Vitesse du processeur 1.70GHz * 4, (iii) Tachyon-0.5.0 (iv) Ubuntu 14.04 LTS (v) Spark 1.6.1 (vi) Oracle Java 7 (vii) Version du langage scala-2.10.4.deb et constructeur sbt pour compiler des projets Scala.
Deuxièmement, nous avons testé notre système proposé dans un cluster multi-nœuds,

constitué de cinq nœuds sur grid5000 dans le cluster de Rennes [2]. La configuration du cluster de l'environnement d'expérimentation est montré dans la table 4.1. Ici, nous avons utilisé une architecture maître/esclave (Spark Master et Spark Workers). Spark Master s'exécute dans (Node0) du cluster et Spark Workers s'exécute sur les nœuds esclaves du cluster (Node1, Node2, Node3, Node4).

4.6.2 Les performances de notre système sur un cluster à un seul nœud

4.6.2.1 Les performances de l'étape d'indexation

Les expériences sont réalisées pour l'indexation de 20.000 images en utilisant deux systèmes de stockage distribués, qui sont HDFS et Tachyon sur Spark. Le tableau 4.2 indique le temps total consommé pour extraire et enregistrer les vecteurs de caractéristiques dans les systèmes Tachyon et HDFS. Nous pouvons noter que, lorsque le jeu de données d'image est augmenté, Tachyon consomme moins de temps pour enregistrer les vecteurs d'entités par rapport au système HDFS. Par exemple, le temps d'indexation sur Tachyon est de 1 200 s avec 20.000 images, alors qu'il atteint 1.500 s sur HDFS.

La performance de la technique d'indexation proposée est comparée à d'autres méthodes de l'état de l'art, tels que Centralisée [79], DIRS[79], et la méthode de Nicolussi et Costantini [82]. Le tableau 4.3 montre que notre approche surpasse les autres méthodes de travail connexes en termes de temps d'indexation. Ce gain est obtenu en utilisant Tachyon et MapReduce pour accélérer le stockage et l'indexation des images. Cette expérience prouve la rapidité de l'approche proposée pour les bases d'images à grande échelle.

4.6.2.2 Les performances de l'étape de recherche

Le tableau 4.4 compare le temps moyen nécessaire pour rechercher des images en utilisant différentes tailles de la base. Nous comparons le temps de recherche en utilisant l'algorithme parallèle k-NN avec et sans la méthode de cache. Nous pouvons observer que dans les deux cas, le temps de recherche augmente proportionnellement à la taille des bases de données d'image. Cependant, ce temps croissant est moins important lorsque nous utilisons la méthode cache. Les temps de recherche moyens obtenus pour 20.000 images sont respectivement de 960 s et 1.980 s, avec et sans méthode cache. Le tableau 4.6 montre clairement la vitesse d'accélération obtenue lors de l'utilisation de l'algorithme parallèle k-NN sous Spark, en particulier avec la méthode de cache, par rapport à d'autres approches de l'état de l'art. Centralisée [79], DIRS [79], et Sakr et al. [83]. En effet, notre approche avec la méthode cache surpasse

2. Grid5000 : Home :https ://www.grid5000.fr.

TABLE 4.1 – La spécification de chaque nœud de cluster de l'environnement d'expérimentation sous le cluster rennes de grid5000

Nœud	Spécification	Role
Nœud0	2 CPUs, 4 cores/CPU	Spark Maître, Tachyon Maître
Nœud1	2 CPUs,4 cores/CPU	Spark esclave, Tachyon worker
Nœud2	2 CPUs,4 cores/CPU	Spark esclave, Tachyon worker
Nœud3	2 CPUs,12 cores/CPU	Spark esclave, Tachyon worker
Nœud	2 CPUs,12 cores/CPU	Spark esclave, Tachyon worker

TABLE 4.2 – La table de temps (s) consommé dans le processus d'indexation d'images selon le nombre de partition d'image.

Nbre d'images	HDFS	Tachyon
2.011	240 s	120 s
5.952	660 s	420 s
11.963	1.260 s	720 s
17.953	1.380 s	1.140 s
20.000	1.500 s	1.200 s

TABLE 4.3 – Comparaison des performances de différentes approches d'indexation.

Approche	Description	Temps (s)
La méthode centralisée [79]	La méthode séquentielle on 1 nœud	600.000
La méthode DIRS [79]	système HBase + MapReduce sous 9 nœuds Hadoop	200.000
La méthode de Luca C. et R. [82]	HDFS + MapReduce sous 1 nœud Hadoop	2.820
Notre méthode	Tachyon + MapReduce sous 1 nœud Spark	460

les meilleurs d'entre eux par 20%([83]). Notez que, dans cette expérience, Sakr et al. utilisent 9 nœuds alors que nous n'en utilisons qu'un seul.

4.6.2.3 L'effet de la méthode cache sur l'algorithme k-NN parallèle

Nous pouvons noter qu'à partir de la Table 4.5, et Fig.4.5 que l'utilisation de la méthode de cache pour sauvegarder le contenu de RDD en mémoire pour éviter le temps de recalculs des RDDs qui est intéressant pour réduire le temps de l'étape de recherche, en particulier, lorsque nous mettons en cache le contenu du DistRDD résultant dans la mémoire (voir l'algorithme 1). À partir de Map la fonction qui calcule les distances entre les vecteurs de CS-LBPs et le vecteur QCS-LBP, ce deuxième cache peut réduire considérablement le temps de la méthode k-NN parallèle, de 1860s avec un seul cache à 960s avec le deuxième cache.

4.6.3 Test de performance sur un cluster multi-nœuds

4.6.3.1 L'évaluation de l'étape d'indexation

La figure 4.6 montre le temps nécessaire pour indexer 20.000 images à l'aide de systèmes de stockage Tachyon et HDFS en utilisant 2, 3, 4 et 5 nœuds. On remarque que le temps d'indexation des images diminue car, d'une part, le processus d'indexation des images est réparti sur les différents nœuds, et, d'autre part, le chargement des images et la sauvegarde des descripteurs sont plus rapides dans Tachyon que dans HDFS.

L'indexation de 20.000 images prend 1.200 s sur le cluster à nœud unique, alors qu'elle ne prend que 720 s avec 5 nœuds (1 maître et 4 esclaves).

4.6.3.2 L'évaluation de l'étape de recherche

La figure 4.7 montre le temps moyen nécessaire pour rechercher une image de requête à l'aide de l'algorithme k-NN parallèle avec et sans la méthode cache dans le cluster à plusieurs nœuds distribué. Nous pouvons remarquer que les performances de l'étape de recherche sont meilleures lorsque nous utilisons l'algorithme k-NN parallèle sur un cluster à plusieurs nœuds, en particulier avec la méthode cache. De plus, l'écart de performances entre l'algorithme parallèle k-NN, avec et sans la méthode de cache, s'agrandit sur le cluster à plusieurs nœuds.

TABLE 4.4 – Comparaison du temps de calcul moyen de l'étape de recherche.

Taille de base de données d'images	k-NN Parallèle sans la méthode Cache	k-NN Parallèle avec la méthode Cache
2.011	180 s	120 s
5.952	540 s	240 s
11.963	1.140 s	540 s
17.953	1.740 s	840 s
20.000	1.980 s	960 s

TABLE 4.5 – L'effet de la méthode du cache sur la méthode k-NN parallèle parmi 20000 images, (voir l'algorithme 1, RDD1 : RDD est les vecteurs caractéristiques d'apprentissage CS-LBPs (RDDi), RDD2 : est le RDD des distances résultant du calcul de distance entre les vecteurs de caractéristiques CS-LBPs et QCS-LBP (DistRDD)

	Nombre de cache	Temps
Sans cache	0	1980s
RDD1	1	1860s
RDD1, RDD2	2	960s

TABLE 4.6 – Comparaison du temps consommé dans l'étape de recherche de cinq méthodes (en s) parmi 20.000 images

Approche	Description	Temps (s)
La méthode centralisée [79]	La méthode séquentielle sous 1 nœud	25.000
La méthode DIRS [79]	k-NN parallèle sous 9 nœuds de Hadoop	15.000
La méthode Sakr et al. [83]	La recherche parallèle sous 1 nœud de Hadoop	1.200
Notre méthode sans cache	k-NN parallèle sous 1 nœud Spark sans cache	1.980
Notre méthode avec cache	k-NN parallèle sous 1 nœud Spark avec cache	960

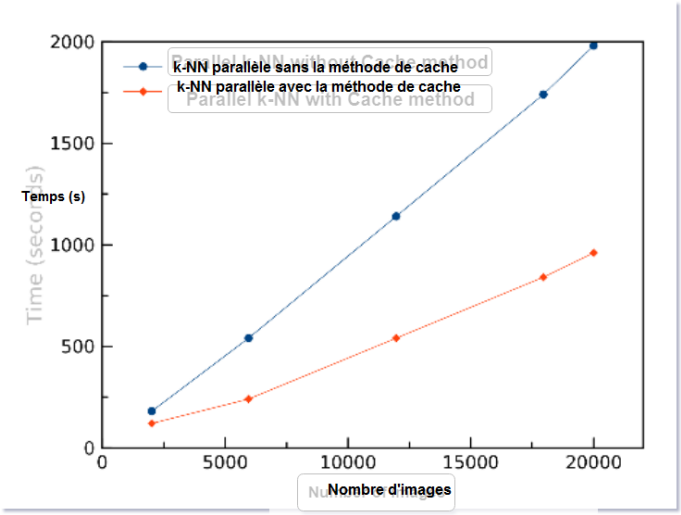

FIGURE 4.5 – Le graphe de temps (s) moyen consommé dans le processus de recherche d'images selon différentes tailles de base de données d'images.

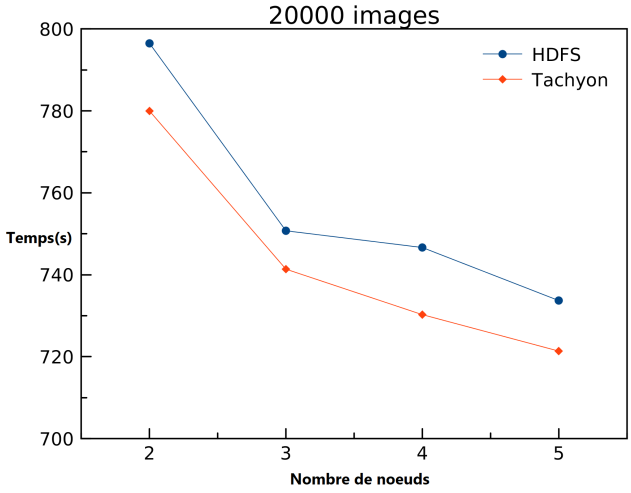

FIGURE 4.6 – Les performances d'indexation de 20.000 images pour différentes configurations de cluster.

4.6.4 L'évaluation de notre sytème en utilisant la base d'images ImageNet

Dans cette expérience, nous avons testé notre système CBIR-S en utilisant les bases de données ImageClef et ImageNet. Comme nous pouvons le constater dans les deux tableaux (4.7 et 4.8), les temps d'indexation et de recherche augmentent proportionnellement à la taille des bases de données d'images (ImageClef de 20.000 images et ImageNet de 1.461.406 images). Le temps d'exécution requis pour la méthode k-NN séquentielle est assez élevé. Cependant, en utilisant l'approche proposée, on obtient une réduction importante du temps de calcul consommé en utilisant Tachyon sur le cluster multi-nœuds.

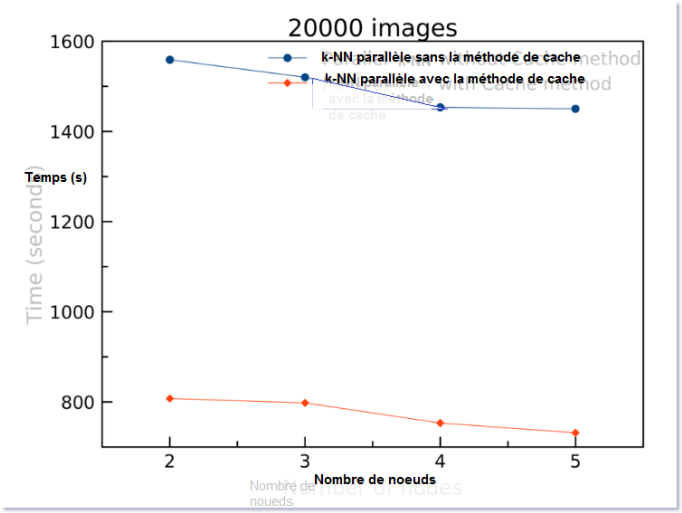

FIGURE 4.7 – Le graphe de temps consommé (en s) dans l'étape de recherche parmi 20000 images sur un cluster multi-nœuds, où 2, 3, 4, 5 : (un maître et les autres esclaves).

TABLE 4.7 – Les performances de l'étape d'indexation de notre système en utilisant les bases de données ImageClef et ImageNet.

La base de donnée	Nombre d'images	avec 1 nœud	avec 5 nœuds
ImageClef	20.000	1.200 s	720 s
ImageNet	1.461.406	51.283 s	32.000 s

TABLE 4.8 – Les performances de l'étape de recherche de notre système sur les bases de données ImageClef et ImageNet.

La base	Nbr d'images	K-NN séquentiel Maillo et al. [74]	K-NN parallèle avec 1 nœud	K-NN parallèle avec 5 nœuds
ImageClef	20.000	/	960 s	790 s
ImageNet	1.461.406	107.735 s	42.250 s	34.265 s

4.7 Conclusion

Dans ce chapitre, nous avons proposé un système CBIR alternatif distribué pour les bases de données d'images à grande échelle en utilisant le framework Spark (CBIR-S). Nous avons utilisé le système de stockage distribué Tachyon pour accélérer le processus de sauvegarde et de récupération des images. Nous avons constaté que Tachyon atteint un débit d'écriture en mémoire plus supérieur à celui de HDFS. De plus, nous avons amélioré le processus de récupération en utilisant l'algorithme k-NN parallèle , tout en exploitant la méthode de cache de Spark.

L'étude expérimentale réalisée sur un cluster à un seul noeud et sur un cluster à plusieurs noeuds de différentes configurations, a montré que le système CBIR-S proposé permet une exécution très compétitive par rapport à d'autres travaux de l'état de l'art. L'accélération obtenue sur le cluster à plusieurs noeuds est plus importante que celle obtenue sur le cluster à un seul noeud. En outre, l'algorithme de recherche k-NN parallèle a montré de bonnes performances, en particulier lorsqu'on utilise la méthode de cache de Spark pour réutiliser les distances calculées précédemment.

Comme perspective de ce travail, le système proposé pourrait être amélioré en utilisant des algorithmes plus efficaces en termes de précision sur des clusters et des ensembles de données plus grands.

Chapitre 5

La classification de faces à grande échelle basée sur la race (ethnicité) sous la plateforme Spark

5.1 Introduction

Récemment, l'identification de la race (ethnique) d'un individu joue un rôle principal dans les systèmes de reconnaissance faciale à grande échelle. Dans ce chapitre, nous proposons une nouvelle méthode de classification des races à grande échelle basée sur la méthode Local Binary Pattern (LBP) et la méthode de régression logistique (LR) sous la plateforme Spark. LBP est l'une des méthodes les plus efficaces dans la reconnaissance faciale [93] [6]. Par conséquent, dans notre système proposé, LBP est utilisée pour extraire les caractéristiques des images faciales. Par la suite, nous utilisons la régression logistique sous Spark comme classificateur pour améliorer la précision et l'accélération du système de classification de race. La méthode de classification de race est effectuée sous la plateforme Spark pour traiter d'une manière parallèle une grande échelle des faces.

L'évaluation de notre méthode proposée a été effectuée sur la combinaison de deux grandes bases d'images de visage CAS-PEAL (partie de pose) [12] et Color FERET [13]. Deux races ont été considérées pour ce travail, les races asiatiques et non asiatiques.

5.2 La reconnaissance des races

La reconnaissance de visage a attiré plus de recherches en raison du fait que la reconnaissance des personnes devient un défi avec les bases de données à grande échelle. En général, le visage humain dans une grande base d'images de visage peut

être classé en diverses manières telles que l'expression, l'âge, le sexe et la race. Parmi ceux-ci, la race (ethnique) est considérée comme l'élément invariant et le plus dominant qui ne peut pas être facilement caché comme le genre et l'âge même en camouflage [94]. De plus, la collecte de visages centrés sur l'âge et le genre non seulement compliquerait le problème, mais pourrait aussi générer des résultats incorrects [95].

En fait, le terme "race" est défini comme suit [1] : "La race est un système de classification, utilisé pour classer les êtres humains en grandes et distinctes populations ou groupes par caractéristiques phénotypiques héréditaires, l'ascendance géographique, l'apparence physique, l'origine ethnique et le statut social.", À partir de cette définition, une forte concurrence a été établie pour le développement du système de reconnaissance des races, en raison de son omniprésence dans de nombreuses applications telles que l'antiterrorisme et divers systèmes de vidéosurveillance qui ont largement émergé dans les lieux publics [95].

Généralement, un système de classification des races typiques commence par une étape de prétraitement pour normaliser la face d'entrée (par exemple, en ce qui concerne l'emplacement, la taille, l'orientation). Ensuite, l'indexation est appliquée pour extraire les entités discriminantes des images de visage. Enfin, un mécanisme de décision (classificateur robuste) est exécuté pour reconnaître la race du visage donné, le résultat de sortie présenté comme des étiquettes de course, comme la race asiatique/africaine [96]. Pour un système efficace de reconnaissance de la race basé sur la vision par ordinateur, les auteurs dans [94] ont proposé sept groupes raciaux les plus connus et les mieux acceptés tels que : Africain/Afro-américain, Caucasien, Asiatique de l'Est, Amérindien/Amérindien, Pacific Islander, Asian Indian, et Hispanic /Latino ", tout cela couvrent plus de 95% de la population mondiale (voir figure 5.1 pour une illustration).

Cependant, les systèmes conventionnels de classification de la race faciale se sont concentrés sur une base de données limitée d'images de visage, qui peuvent être affichées à l'écran. Par conséquent, ces applications doivent être généralisées et formées sur une base de données à grande échelle [94]. Pour cela, le principal défi consiste à concevoir un système efficace et précis de classification de la race faciale en utilisant une base des images à grande échelle [94], [97]. Par conséquent, dans ce chapitre, nous présentons le problème de la classification des races sur une base des images à grande échelle, en proposant une nouvelle approche de classification des races sous la plateforme Spark. Nous présentons nos principales contributions :

(1) Le traitement d'images de visage à grande échelle, en se concentrant sur une nouvelle technologie, qui s'appele Spark framework [98], pour faire un traitement parallèle d'une grande échelle de visages.

(2) L'introduction d'une combinaison de LBP en tant que vecteur de caractéristiques faciales et de régression logistique de Spark MLlib, et en tant que classifieur pour la classification des races faciales, où nous utilisons LBP, puisque la méthode LBP surpasse les autres méthodes de reconnaissance faciale telles que EBGM, PCA (Analyse

1. Race Wiki (Race classification humaine) : http ://en.wikipedia.org/wiki/

FIGURE 5.1 – Illustration de la distribution de la variance génétique des races humaines [4].

en Composantes Principales), LDA (Linear Discriminant Analysis) dans la précision de la classification [93] [6]. De plus, la régression logistique ne s'est pas limitée aux problèmes de probabilité mais a également été utilisée pour la classification.

(3) La comparaison de notre méthode de classification de race proposée pour classer les images faciales dans deux groupes de races (asiatiques et non-asiatiques), avec quatre autres classificateurs Spark MLlib [2], qui sont SVM linéaires, bayésiens naïfs (NB), forêts aléatoires (RF) et arbres de décision (DT) en termes de précision et temps de classification. En outre, la précision de la classification de notre approche proposée mesurée à différentes échelles par rapport aux quatre classificateurs.

(4) La comparaison de notre méthode de classification raciale à certaines méthodes récentes en termes de temps et d'exactitude, en utilisant deux bases d'images à grande échelle CAS-PEAL et Color FERET, qui contiennent 34214 images de visage de 2250 faces classes.

Le reste de ce chapitre est organisé comme suit. La section 5.3 décrit la littérature de la classification des races. Ensuite, Section 5.4 les algorithmes utilisés dans ce travail et notre méthode proposée dans Section 5.5. La section 5.6 décrit l'environnement de notre expérience et un ensemble de résultats expérimentaux sont rapportés. Enfin, des conclusions et des idées pour de futures extensions de ce travail sont données dans la section 5.7.

2. Classification et régression spark.mllib : https : // spark. apache.org/docs/1.6.1/mllib-classification-regression.html

5.3 État de l'art

Récemment, la recherche sur la reconnaissance des races basée sur le visage a largement émergé. En conséquence, les parties faciales pour la reconnaissance de race sont traitées de trois façons soit : (i) Utiliser l'image de la texture de l'iris pour la reconnaissance des races. (ii) Utiliser la région périoculaire, qui contient des informations raciales distinctives. (iii) la reconnaissance raciale basée sur le visage holistique.

Dans notre travail, nous nous sommes basée sur la manière la plus existante, qui est le visage holistique, en raison de nombreuses littératures se concentrant sur la classification de l'ethnicité basée sur le visage holistique. Nous commençons par les premiers travaux proposés par Gutta et al. [99], [100], [101] qui ont utilisé le réseau de neurones RBF avec l'arbre de décision pour la reconnaissance du race en utilisant la base d'images FERET, leur meilleure précision était de 94%. Après cela, le travail de Lu et Jain [102] ont développé un système de classification de race basé sur l'image de visage holistique, et l'analyse discriminante linéaire (LDA) pour la classification ethnique des deux races (Asian et Non-Asian). Les résultats expérimentaux ont atteint une précision de 96,3% sur l'ensemble de données mélangées manuellement de 2630 faces (97,7% comme un meilleur résultat). Ou et al. [103] ont introduit PCA-ICA en tant que caractéristique du visage, SVM comme un classifieur et en les fusionnant, ils ont atteint 82,5% sur la base de données FERET de 750 images de visage.

Dernièrement, Manesh et al. [104] ont proposé une classification basée sur un modèle d'ethnicité. Les auteurs ont utilisé le masque de ratio d'or modifié pour séparer automatiquement les régions de visage, les caractéristiques de Gabor extraites de chaque patch, dans la phase d'apprentissage, ils ont utilisé le classifieur SVM. Ils ont obtenu une précision de reconnaissance de 98% comme meilleur résultat sur des bases d'images combinées de 1691 visages de FERET et CAS-PEAL pour la classification asian/non-asian. Roomi et al. [105] ont étudié les caractéristiques discriminatoires, qui sont la caractéristique de la couleur de la peau, et d'autres caractéristiques secondaires comme le front et la lèvre du visage donné pour classer les races. Leurs résultats expérimentaux sur les bases de données Yale et FERET avec 250 classes ont montré une précision de 91,6% pour la classification caucasienne, asiatique, africaine et américaine. Une autre méthode globale utilise trois races (européenne, orientale, africaine), cette méthode a été proposée plus tard par Salah et al. [106], où ils ont exploité les performances de la fusion de motifs LBP basés sur des blocs et de la transformation en ondelettes de Haar pour extraire des entités. Ensuite, ils ont utilisé K-plus proches voisins (KNN) comme classifieur pour la classification de la race sur la base de données EGA de 746 faces, leur meilleure précision était de 96,6%. Chen and Ross [107] ont introduit un nouveau descripteur de modèle gabor à gradient local (LGGP). La classification de représentation collaborative (CRC) est adoptée comme méthode de classification. Les résultats de l'expérience sur les bases d'images MORPH et CAS-PEAL de 750 faces ont obtenu une moyenne de 98,7% comme une meilleure précision pour la classification des trois races (asiatique, caucasienne et africaine). Dans [108], Momin et Tapamo ont présenté

un modèle pour détecter les composants du visage comme les yeux, le nez et la bouche à partir de l'image du visage et ils ont extrait les principales caractéristiques faciales en appliquant des filtres de Gabor à chaque composant. Dans la phase d'entraînement, ils utilisaient les méthodes de classification K-means, Naïf Bayésien (NB), perceptron multicouche (MLP), SVM(Support Vector Machines) pour classer l'image du visage humain en différents groupes. 99,60% de précision a été atteinte pour le groupe asian et 99,23% pour le groupe non asian sur la fusion de trois bases de données de 1022 images.

En effet, toutes les méthodes citées ci-dessus ont été réalisées sur des bases d'images limitées avec un petit nombre d'images. Cependant, avec l'augmentation du nombre de visages humains au cours de ces dernières années, les chercheurs proposent de nouvelles méthodes pour surmonter la classification du race dans les bases de données de grande taille. Xi et al. [109] ont extrait les entités de faces à l'aide de l'analyse des entités dépendant de la classe du noyau (KCFA) et des méthodes de coloration du visage. Ils ont atteint la plus haute précision pour les Caucasiens, les Africains et les Asiatiques respectivement, 97%, 97% et 95% sur la base de donnée MBGC (20000 Caucasiens, 10000 Africains, 10000 Asiatiques). Une autre approche pour la classification des images à grande échelle fondée sur l'origine ethnique est étudiée par Han et al. [97] sur deux races (noir et blanc). Ils ont extrait des caractéristiques informatives démographiques en utilisant des entités biologiquement inspirées (BIF), le classificateur hiérarchique est utilisé pour prédire la race d'image de visage donnée. Pour évaluer l'efficacité de la méthode proposée, ils ont utilisé deux bases d'images de visage à grande échelle (MORPH2 de 78207 images) et (PCSO de 100012 images). Ils ont atteint une précision de (99,1%, 98,7%) respectivement. Dans [110] Anwar et Naeem ont proposé une nouvelle méthode pour la classification d'ethnicité, basée sur le réseau neuronal convolutif (CNN) pour extraire les caractéristiques, dans la phase d'apprentissage, ils ont utilisé SVM avec un noyau linéaire comme classificateur. Les résultats de précision obtenus sur trois races : asiatiques, afro-américaines et caucasiennes sont respectivement de 99,28%, 99,66 % et 99,05% sur une combinaison de dix différentes bases de données faciales 13394 images.

5.4 Méthodes connexes

5.4.1 La méthode Local Binary Pattern (LBP)

Généralement, l'opérateur LBP est devenu le plus utilisé dans la reconnaissance faciale [93] [6]. En fait, l'opérateur LBP a été introduit par Ojala et al. [111] pour le descripteur de texture, la méthodologie LBP démarre à partir d'une fenêtre 3 * 3, elle calcule les pixels centraux et voisins de l'image de la fenêtre. Par conséquent, le code LBP du pixel courant est alors produit en concaténant ces 8 valeurs pour former un code binaire. On obtient donc, comme pour une image à niveaux de gris, une image des va-

leurs LBP contenant des pixels dont l'intensité se situe entre 0 et 255 [111], son calcul est comme suit :

$$LBP_{P,R}(x_c, y_c) = \sum_{k=0}^{P-1} 2^k * S(g_k - g_c) \tag{5.1}$$

Où g_c et g_k sont respectivement des valeurs de niveau de gris du pixel central et p pixels environnants dans le voisinage de cercle de rayon R, et $S(g_k - g_c)$ est défini comme :

$$S(g_k - g_c) = \begin{cases} 1 & g_k - g_c >= 0 \\ 0 & g_k - g_c < 0 \end{cases} \tag{5.2}$$

5.4.2 Méthode de régression logistique (LR)

L'algorithme linéaire le plus commun pour la classification et la reconnaissance d'image de visage est SVM. Ici, dans notre approche proposée, nous avons utilisé l'algorithme LR avec BFGS de Spark MLlib pour la classification de race à grande échelle basée sur l'image, qui est un algorithme linéaire, L'algorithme est défini par l'équation (5.3) comme suit [5] :

$$f(w) = \lambda R(W) + \frac{1}{n} \sum_{i=1}^{n} L(W, x_i, y_i) \tag{5.3}$$

Où, $x_i \epsilon \Re^d$ sont les exemples de données d'apprentissage, et $y_i \epsilon \Re$ sont leurs étiquettes correspondantes, l'ensemble de données d'apprentissage est représenté par un RDD du point étiqueté dans MLlib, W est un poids, et λ est le paramètre de régularisation fixe ($\lambda >= 0$). Dans l'équation ci-dessus L (W;.) est la fonction de perte, qui est donnée par la perte logistique :

$$L(W, x, y) = \log(1 + \exp(-yW^T x)) \tag{5.4}$$

Ici, nous avons utilisé la classification binaire, donc l'algorithme produit un modèle LR binaire, dans lequel le modèle LR binaire prédit les nouvelles données d'échantillon en appliquant la fonction logistique :

$$f(z) = \frac{1}{1 + e^{-z}} \tag{5.5}$$

Où $z = W^T x$ par défaut, si f($W^T x$)>0.5, le résultat est positif, ou négatif sinon.

5. Linear Methods RDD-based API : https ://spark.apache.org/docs/2.2.0/mllib-linear-methods.html/logistic-regression

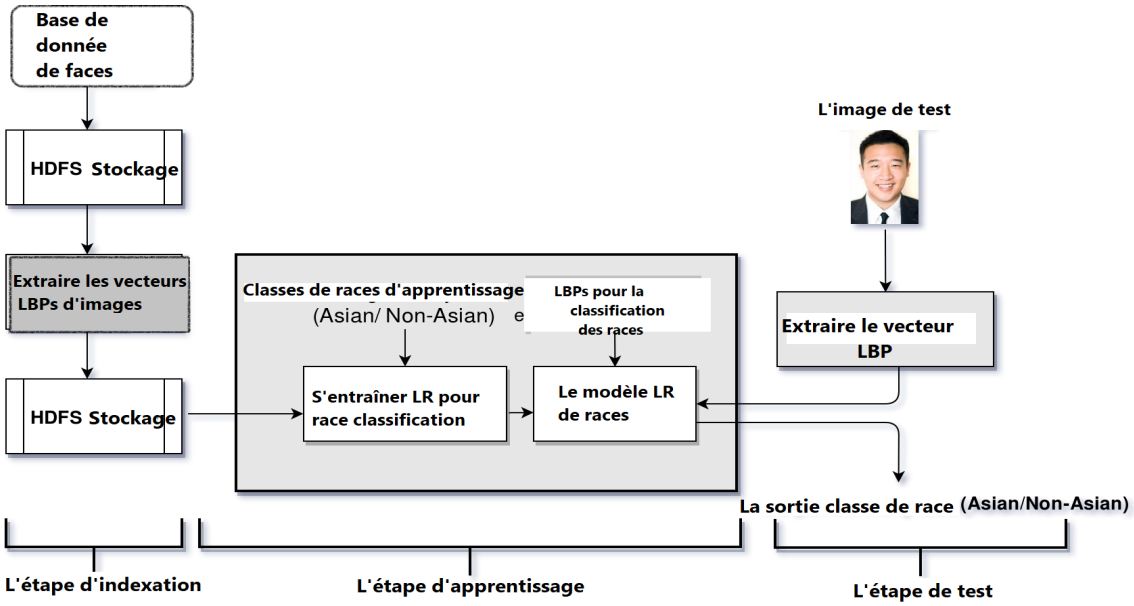

FIGURE 5.2 – Architecture de système de classification de race (ethnicité) basée sur le visage architecture sous Spark

5.5 Travail proposé

En combinant les avantages des méthodes LBP et de régression logistique présentées dans la section 5.4, nous proposons notre système de classification de race basée sur des images à grande échelle, qui utilise la méthode LBP pour extraire les caractéristiques des images de visage. Par la suite, nous utilisons la régression logistique comme classificateur sous le framework Spark pour traiter en parallèle un ensemble de données à grande échelle. De plus, nous utilisons Hadoop Distributed Storage System (HDFS) [112] pour sauvegarder une grande quantité d'images. La figure 5.2 présente l'architecture de notre système proposé.

5.5.1 Implémentation sous Spark

Le système parallèle de reconnaissance de visage basé sur la race proposée a été implémenté en utilisant Spark [98], qui permet de traiter en parallèle une énorme quantité de données. Un traitement parallèle est fourni par le modèle de programmation MapReduce [90]. Afin de stocker de grandes quantités d'images, nous utilisons le système HDFS. Le HDFS est adapté pour enregistrer les fonctionnalités LBP pour le traitement parallèle. MapReduce est un modèle de programmation utilisé pour paralléliser le traitement de grands ensembles de données, il est basé sur deux étapes Map et Reduce, qui se présentent comme suit :

- Map : <Key1, value1> − > list(<key2, value2>)

-Reduce : <key2, list(value2)> − > list(<key3, value3>).

En effet, dans notre système proposé, il y a trois étapes, , comme illustré à la figure 5.2. (i) l'étape d'indexation pour l'extraction de caractéristiques. (ii) l'étape d'apprentissage. (iii) l'étape de test.

- Initialement, le mapper convertit les images couleur en image en niveaux de gris, redimensionne les images et enregistre les images traitées dans le système HDFS, comme une étape de prétraitement. Dans l'étape d'indexation, le mapper extrait les vecteurs de caractéristiques LBP sur chaque bloc d'images de visage traités, qui sont chargés à partir de HDFS. Après cela, les vecteurs de caractéristiques LBP extraits sont transmis au reducer, ce qui enregistre plus tard les vecteurs de caractéristiques LBP dans HDFS, en utilisant l'action de sauvegarde (save).

Ensuite, dans l'étape d'apprentissage, le mapper charge les vecteurs d'apprentissage LBPs à partir du système de stockage HDFS et construit le modèle à partir des vecteurs de caractéristiques LBPs chargés en utilisant le classifieur de régression logistique de Spark MLlib, et le reducer enregistre le modèle de régression logistique.

Enfin, dans l'étape de test, le mapper charge le modèle de régression logistique, après avoir extrait un vecteur de caractéristiques LBP de la requête d'entrée donnée. En conséquence, le reducer prédit l'étiquette de la classe de visage d'entrée (dans notre cas soit asiatique ou non asiatique).

5.5.2 L'algorithme de système de classification de races à grande échelle sous Spark

Le système proposé est composé des étapes d'indexation, d'apprentissage et de reconnaissance. L'algorithme 3 résume ces trois étapes principales comme suit :

Comme un prétraitement, nous chargeons les images du système HDFS dans RDD (imgRDD) en utilisant sc.textFile (instruction 10). Ensuite, le mapper convertit les images couleur chargées en images en niveaux de gris, redimensionne les images à l'aide de **la fonction processImages** (instruction 11) et de l'instruction 12. Les images traitées dans le système HDFS.

(i) L'étape d'indexation :

Au cours de cette étape, le mapper extrait les vecteurs de caractéristiques LBP de chaque bloc d'images de visage traité à l'aide de la fonction **extractDescriptor** (instruction 15), qui est chargée à partir de HDFS (instruction 14). Ensuite, les vecteurs d'extraction LBP extraits sont transmis au reducer, qui les enregistre dans HDFS à l'aide de l'action save (instruction 16).

(ii) L'étape d'apprentisage :

Au cours de la phase d'apprentissage, le mapper charge les vecteurs de fonctionnalité

LBP d'apprentissage à partir du système de stockage HDFS. L'instruction 22 crée un point étiqueté avec une étiquette positive et un vecteur de caractéristiques denses. Les points marqués sont utilisés dans des algorithmes d'apprentissage supervisé (LR) et l'instruction 23 construit le modèle à partir des vecteurs d'entités LBP chargés à l'aide du classifieur de régression logistique (LR) de Spark MLlib.

(iii) L'étape de reconnaissance :

Dans cette dernière étape, le mapper charge le modèle de régression logistique, après avoir extrait le vecteur de caractéristiques LBP et crée les points étiquetés d'une image de requête de face en entrée (instruction 27). En conséquence, le réducer prédit l'étiquete de la classe d'entrée (dans notre cas, asiatique ou non asiatique) (instruction 28).

Enfin, les instructions (30 et 31) calculent la précision du modèle généré.

Algorithme 3 : Implementation of parallel face image-based race recognition on Spark using scala language.

```
1  begin
2  /*Création de SparkContext*/
3  val conf = new SparkConf().setAppName("Race Recognition")
4  .setMaster("spark://salihaInspiron3542:7077")
5  .set("spark.executor.memory", "800m")
6  .setJars(Seq("/usr/share/scala/faceProjets/FR/target/scala-
     2.10/raceRecognition2.10-1.0.jar"))
7  List("target/scala-2.10/raceRecognition2.10-1.0.jar")
8  val sc = new SparkContext(conf)
9  /*Pré-traitemrnt*/
10 var imgRDD=
     sc.textFile("hdfs://localhost:9000/Monsysteme/ColorferetCASimages")
11 val processedImages = imgRDD.map(f => processImages(f).mkString(", "))
12 processedImages.saveAsTextFile("hdfs://localhost:9000/Monsys-
     teme/processedImg")

13 /*L'étape d'indexation*/
14 var loadedImg =
     sc.textFile("hdfs://localhost:9000/Monsysteme/processedImg")
15 val lbpDescriptors = loadedImg.map(f => extractDescriptor(f).mkString(", "))
16 lbpDescriptors.saveAsTextFile("hdfs://localhost:9000/Monsysteme/LBPs")
17 var LBPsTraining =
     sc.textFile("hdfs://localhost:9000/Monsysteme/LBPsTRAIN")
18 /*Les étape d'apprentissage et de reconnaissance*/
19 /* Create a labeled point*/
20 val trainingData = LBPsTraining.map {line => val parts = line.split(",")
21 LabeledPoint(extractLabel(parts(0)), Vectors.dense(parts.slice(1,257)
22 .map(x => extractLabel(x)))}
23 val LRmodel = new
     LogisticRegressionWithLBFGS().setNumClasses(2000).run(LBPstraining)
24 var LBPsTesting = sc.textFile("hdfs://localhost:9000/Monsysteme/LBPsTEST")
25 val testData = LBPsTesting.map{line => val parts = line.split(",")
26 LabeledPoint(extractLabel(parts(0)), Vectors.dense(parts.slice(1,257)
27 .map(x => extractLabel(x)))) }
28 val labelAndPreds = testData.map { point => val prediction =
     LRmodel.predict(point.features) (point.label, prediction) }
29 /*Calculer l'accuracy du modèle*/
30 val metrics = new MulticlassMetrics(labelAndPreds)
31 val accuracy = metrics.accuracy println("the best accuracy is accuracy")
32  end;
```

5.6 Résultats expérimentaux

5.6.1 L'environnement expérimental

Pour évaluer l'efficacité de notre méthode proposée sur des bases d'images à grande échelle, nous combinons deux bases de visages différentes, qui sont CAS-PEAL [6], [12], et ColorFeret [7], [13],qui contiennent 34214 images pour former et tester notre système. Ensuite, les images ont été regroupées en groupes de races asiatiques et non-asiatiques.

Ici, le premier groupe de race CAS-PEAL est une base d'images à grande échelle, à partir de cette base de données, nous utilisons uniquement la partie pose, qui contient 21182 visages chinois sous différentes poses de 1042 classes, où chaque classe contient 21 faces, Cet ensemble de données est souvent traité comme un ensemble de données asiatique [94]. Ensuite, le deuxième groupe de race est Color FERET, qui contient 12332 visages non-Aassiens sous différentes poses de 1208 individus. Des exemples d'images de race provenant des bases de données sont présentés dans la figure 5.3, et 5.4. Tous les visages ont été redimensionnés en 24 * 24 pixels comme dans [113], [102] et convertis en images de niveaux de gris.

Tous les résultats rapportés dans cette section sont obtenus en utilisant une validation croisée 10 fois comme dans [114], [113].

Dans cet environnement, nous utilisons le framework Apache Spark pour configurer une grille. Ici, nous adoptons un cluster à un seul nœud, les spécifications de ce cluster sont les suivantes :

- Environnement logiciel : Ubuntu 14.04 LTS, Spark 2.0.0, MLlib de Spark, Oracle java 7.

-Language scala-2.10.4.deb, et sbt builder pour compiler nos projets de scala.

-Hardware environnement : Intel Core i5-42104 CPU vitesse 1,70 GHZ * 4, RAM 8 Go.

5.6.2 La précision de la classification de race (ethnicité)

Dans cette expérience, nous avons utilisé différents classifieurs MLlib de Spark, qui sont la forêt aléatoire (RF), Naive bayésienne (NB), l'arbre de décision (DT), Linear Support Vector Machine (SVM). Une fois que les vecteurs de caractéristiques sont extraits en utilisant LBP, ils sont introduits dans chacun des classifieurs choisis. Les premières images sont classées en deux groupes de races asiatiques et non-asiatiques. La table 5.1 montre la précision moyenne pour la classification de race à grande échelle.

6. CAS-PEAL Face Database : http ://www.jdl.ac.cn/peal/index.html
7. color FERET Database Share : https ://www.nist.gov/itl/iad/image-group/color-feret-database

FIGURE 5.3 – Des échantillons de la base de données. (a) CAS-PEAL dataset (POSE part)

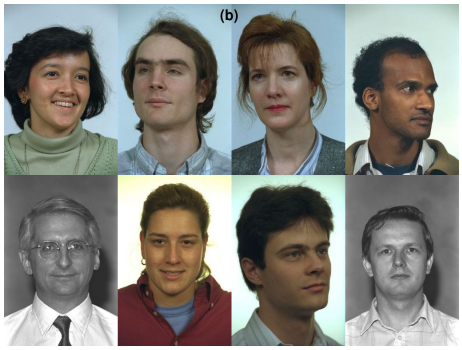

FIGURE 5.4 – Des échantillons de la base de données. (b) Color FERET dataset

Dans cette table avec le groupe asiatique, les classifieurs (SVM Linéaire, DT, RF, NB, LR) atteignent presque environ 99 % de précision. Cependant, le classifieur LR fonctionne mieux que les autres classifieurs. Par conséquent, avec le groupe non asiatique Linear SVM atteint 95,60% comme la plus faible précision moyenne, et les classifieurs RF, NB et DT atteignent respectivement (97,68%, 98,41%, 99,05%). Cependant, le classifieur LR atteint 99,78 % comme la plus grande précision.

5.6.3 L'effet de la résolution de l'image sur la reconnaissance de race

Afin de montrer l'effet de la résolution de l'image sur le processus de classification, les descripteurs LBP sont extraits sur quatre tailles d'image différentes (24×24, 42×42, 128×128, et 256×256 pixels). La table 5.2 présente la précision moyenne en utilisant le descripteur LBP, les descripteurs LBP extraits sont alimentés dans chaque classifieur (RF, NB, DT, SVM linéaire). Les résultats expérimentaux donnent une précision moyenne pour la classification de la race, impliquant deux catégories (asiatique, non-asiatique) à chacune des résolutions choisies. Nous pouvons observer que la précision de tous les classifieurs pour les non-asiatiques est inférieure à celle pour le groupe asiatique. Par conséquent, en augmentant la taille des images, nous avons une augmentation significative de la précision de certains classifieurs, c'est-à-dire (DT, NB, RF et LR). Cependant, la précision de Linear SVM diminue avec l'augmentation de la taille de l'image. Après réduction de la résolution, lorsque la taille de l'image du visage est de 21 * 21 pixels, l'efficacité de la reconnaissance diminue pour tous les classifieurs. En effet, notre méthode de classification de race est résistante aux au changement d'échelle de l'image.

5.6.4 Temps d'apprentissage et de test

Le graphe 1 de la figure 5.5 montre clairement les performances en termes de temps et de précision de classification. Il est clair que notre méthode proposée atteint 99,75% comme meilleure précision moyenne de classification. En outre, notre méthode proposée et NB fonctionnent plus rapidement que les autres méthodes.

5.6.5 Comparaison des performances de la méthode proposée avec certaines méthodes d'état de l'art (en termes de précision)

Le tableau 5.3 récapitule certaines méthodes existantes pour le classement des races et leurs performances. Nous pouvons remarquer une augmentation claire de l'efficacité au cours des dernières années. Les performances obtenues dépendent de nombreux paramètres, notamment des descripteurs et classificateurs utilisés, les bases de données existantes, leurs tailles, et de leurs groupes raciaux. On peut remarquer que

TABLE 5.1 – La précision de la classification de race pour chaque classifieur de Spark par rapport à la méthode proposée, toutes ces expériences réalisées en utilisant une validation croisée 10 fois. Les classes ethniques considérées ici sont asiatiques contre non-asiatiques.

La méthode	Accuracy (%)		accuracy moyenne (%)
	Asian	Non-Asian	
Linear SVM	99.59	95.60	97.59
DT	99.48	99.05	99.26
RF	99.62	97.68	98.65
NB	99.67	98.41	99.04
Notre méthode proposée	99.73	99.78	99.75

TABLE 5.2 – L'effet du changement d'échelle sur les performances de classification de race avec différents classifieurs en utilisant LBP sous Spark (validation croisée 10 fois) sur deux bases de données à grande échelle CAS-PEAL et ColorFeret. Il existe quatre résolutions différentes, 24 * 24, 42 * 42, 128 * 128, 256 * 256. Le contenu de chaque cellule représente le taux de précision moyen

Taille	**Classifieurs**	**Accuracy (%)**		
		Asian	**Non-Asian**	**Accuracy moyenne**
24*24	**RF**	99.62	97.68	98.65
	SVM linéaire	99.59	95.60	97.59
	DT	99.48	99.05	99.26
	NB	99.67	98.41	99.04
	Notre méthode	**99.73**	**99.78**	**99.75**
42*42	**RF**	99.80	99.60	99.70
	SVM linéaire	99.59	95.61	97.60
	DT	99.81	99.77	99.79
	NB	99.80	98.00	98.90
	Notre mthode	**99.85**	**99.81**	**99.83**
128*128	**RF**	99.89	99.88	99.88
	SVM linéaire	98.95	92.80	95.87
	DT	99.88	99.95	99.89
	NB	99.90	98.41	99.15
	Notre méthode	**99.95**	**99.93**	**99.91**
256*256	**RF**	99.99	99.89	99.94
	SVM linéaire	99.18	94.86	97.02
	DT	99.96	99.88	99.92
	NB	99.90	98.55	99.22
	Notre méthode	**100**	**99.99**	**99.99**

la meilleure précision obtenue, par d'autres méthodes, pour les races asiatiques/non asiatiques est de 99,6 %, tandis que notre méthode, qui utilise des bases de données à grand échelle, qui sont CAS-PEAL et Color FERET (FERET étendu), nous permet d'atteindre 99,99% comme précision moyenne la plus élevée.

Dans le tableau 5.4, nous comparons notre approche à d'autres méthodes traitant du classement des races asiatiques/non asiatiques. La plupart de ces méthodes sont évaluées à l'aide de bases de données de visage FERET et CAS-PEAL. Les résultats montrent que notre approche permet de classer deux groupes de race différents (asiatique et non asiatique) plus efficacement que les autres méthodes. En effet, la précision obtenue par notre méthode est très élevée pour les deux groupes de races, où nous atteignons respectivement 100% et 99,99% pour les races asiatiques et non asiatiques.

Le tableau 5.5 montre les comparaisons de temps d'apprentissage et de reconnaissance utilisant des approches récentes avec différentes tailles de bases de données d'apprentissage. Avec de petits ensembles de données d'apprentiage Shouman et Hamdy [115] et Masood et al. Les méthodes [95] ont un temps d'apprentissage très élevé par rapport aux autres méthodes. Néanmoins, notre méthode proposée, dans laquelle la base de données d'apprentissage est volumineuse, indique une accélération du temps d'apprentisage, passant de plusieurs heures avec la méthode Shouman et Hamdy [115] en utilisant une petite base de données d'apprentissage de 1 000 images de visage à 486s avec notre approche en utilisant 21.190 images de visage. Ce gain du temps d'apprentissage est dû à l'utilisation du classificateur LR sous Spark effectuant la phase d'apprentissage de manière parallèle. En outre, le temps de reconnaissance de notre méthode proposée sous la plateforme Spark est plus rapide lorsque la taille de la base de données d'apprentissage est augmentée; il atteint 0,06 seconde en utilisant 21.190 images d'apprentissage.

TABLE 5.3 – Comparaison avec certains systèmes de classification de race existants dans la littérature.

Auteurs	Année	Approches	Accuracy (%)	Base de donnée	groupe de Race
Lu et al. [102]	2004	Gausian+LDA	97.7	Asian PF01, NLPR, AR et Yale (2630 faces)	Asian vs. Non-Asian
Ou et al. [103]	2005	PCA+ICA +SVM	82.5	FERET	Asian vs. Non-Asian
Manesh et al. [104]	2010	Facial part templates Gabor+SVM	98	FERET + CAS-PEAL (1691faces)	Asian vs. Non-Asian
Roomi et al. [105]	2011	Skin color+ Adaboost	91.6	Yale+Feret (250 objets)	Caucasian, Asian, Affrican, American
Xie et al. [109]	2012	KCFA et color features	97 (caucasian) 97 (African) 95 (Asian)	MBGCDB (20000 Caucasian, 10000 Afriacan, 10000 Asian	Caucasian, African et Asian
Chen and Ross [107]	2013	Local gradient Gabor Pattern	98.7	MORPH et CAS-PEAL (750 faces)	Asian, Caucasian et African
Salah et al. [106]	2013	LBP+Harr wavelet +PCA+KNN classifieur	98.7	EGA DB (746 faces)	Asian, Caucasian, African et American
Han et al. [97]	2014	Biologically inspired features (BIF)	99.1(MORPH2) 98.7(PCSO)	MORPH2(78207) PCSO(100012)	Black and White
Momin and Tapamo [108]	2016	Gabor features+ Kmeans, NB, MLP and SVM	99.6 (Asian) 99.23 (Non-Asian)	MORPH base de donnée, Indian face (484 faces)	Asian vs. Non-Asian
Anwar and Naeem [110]	2017	CNN to extract features +SVM classifier	98.28(Asian) 99.66(African) 99.05(Caucasian)	Ten different facial base de donnée (13394)	Asian, African, American, Caucasian
Masood et al. [95]	2018	les méthodes CNN et ANN	82.4(ANN) 98.6(CNN)	FERET base de donnée (447 face images)	Caucasian, Mongolian, et Negroid
Notre méthode	2018	LBP + Logistic regression sous spark	**99.99**	CAS-PEAL (Pose part) + ColorFERET base de donnée (34214 faces)	Asian vs. Non Asian

TABLE 5.4 – Comparaison entre notre approche proposée et les travaux connexes.

Auteurs	Méthode	Base de donnée	Accuracy	
			Asian (%)	Non-Asian (%)
Manesh et al. (2010) [104]	Facial template Gabor+SVM	FERET +CAS-PEAL	98	96
Muhammed et al. (2012) [**?**]	LLC	FERET	99.47	-
Chen et Ross (2013) [107]	Local gradient Gabor Pattern	CAS-PEAL and MORPH II	98.7	-
Momin et Tapamo (2016) [108]	Gabor Features+ MLP	MORPH II	99.6	99.23
Anwar et Naeem (2017) [110]	CNN+SVM	FERET	98.28	-
Notre approche	LBP+Logistic Regression	FERET + CAS-PEAL	**100**	**99.99**

TABLE 5.5 – Comparaison de temps d'apprentissage et de reconnaissance en utilisant différentes méthodes.

Auteurs	Année	Méthodes	Temps d'apprentissage (s)	Temps de reconnaisssance (s)	Nombre d'images d'apprentissage
Muhammad et al. [96]	2012	LBP+minimum distance classifieur	-	0.1394s	2368
Bagwe et al. [116]	2016	PCA+KNN sous Hadoop et Map-reduce	-	0.2300s on 9 slaves hadoop	3120
Shouman et Hamdy [115]	2017	PCA+MPI	7713.812s	0.609s sous 5 serveurs	1000
Masood et al. [95]	2018	CNN	824.03s	-	357
Notre méthode	**2018**	**LBP+Logistic Regression classifieur sous Spark**	**486s**	**0.06s**	**21190**

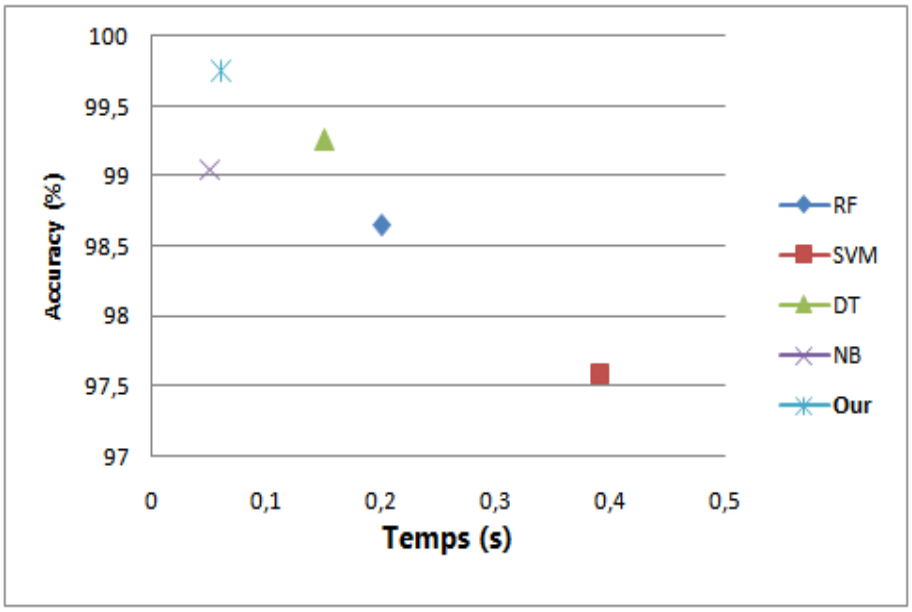

FIGURE 5.5 – Performances de la classification de race avec différents classifieurs utilisant le framework Spark sur deux bases de données à grande échelle CAS-PEAL et ColorFeret (temps de test vs. à la précision de la reconnaissance de race)

5.7 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle méthode de classification de race basée sur le visage sur des bases de données à grande échelle en utilisant le framework Spark. La méthode proposée utilise un classificateur de régression logistique (LR) pour classifier les descripteurs de caractéristiques extraits d'images de visage à l'aide de la méthode de Local Binary Pattern (LPB). Notre approche a été évaluée sur des classes raciales asiatiques et non asiatiques, elle est basée sur la la combinaison de deux bases de données connues, CAS-PEAL et Color FERET, utilisées pour la classification des races.

Les résultats expérimentaux montrent que la méthode proposée, qui utilise le classificateur de régression logistique, permet d'atteindre une précision moyenne de 99,99% par rapport à la méthode SVM linéaire, forêt aléatoire (RF), l'arbre de décision (DT) et au classificateur naïf bayésien (NB). De plus, notre méthode utilisant Spark sur des images de visage à grande échelle a montré une amélioration significative du temps de calcul par rapport aux autres classificateurs, où elle effectue les étapes d'apprentissage et de reconnaissance en seulement 486s et 0,06s, respectivement. Un autre avantage de ce travail est sa robustesse face aux changements d'échelle d'image.

L'approche de classification de race proposée peut être utile pour la reconnaissance de l'identité faciale sur des images à grande échelle et pourrait être étendue pour combiner davantage de catégories de race. Il convient également de tester la méthode proposée en termes de temps sur des clusters Spark et les GPU distribués.

Conclusion générale

La classification des images à grande échelle est une tâche très importante pour différents systèmes de reconnaissance d'images. Actuellement, aucune solution optimale au problème de la classification d'images à grande échelle n'est encore connue, Néanmoins, quelques progrès importants dans ce domaine aient permis le développement de quelques systèmes pratiques, afin d'atteindre des meilleurs résultats en termes de délais de traitement rapides et de précision pour faire face aux grandes quantités d'images produites (Big Data). Ces raisons rendent ce domaine de recherche constitué encore un terrain fertile pour de futurs travaux.

Cette thèse a été consacrée à l'étude et la réalisation d'un système de classification des images à grande échelle sur des machines massivement parallèles. Afin de réaliser ce type de système, nous avons abordé dans le Chapitre 2 le domaine de la vision par ordinateur et ces concepts de base. Dans le Chapitre 3, nous avons présenté d'une part l'état de l'art sur les technologies de Big Data spécialement les plateformes de stockage et de calcul, et d'autre part l'infrastructure distribuée nécessaire pour rendre ces plateformes faciles à utiliser. Le Chapitre 4 présente l'architecture d'un système de recherche d'images par le contenu (CBIR) pour les bases des images à grande échelle (big data) sous la plateforme Spark, dont nous avons utilisé le modèle distribué MapReduce sous la plateforme Spark pour traiter des grands volumes de données en parallèle en divisant le travail en un ensemble de tâches indépendantes. Afin d'accélérer le processus d'indexation nous avons utilisé également le système de stockage distribué, qui s'appelle Tachyon. Par ailleurs, pour accélérer le processus de recherche nous avons utilisé l'algorithme parallèle de K-plus poches voisins (k-NN) de classification basée sur le modèle MapReduce implémenté sous Apache Spark. En outre, nous avons exploité la méthode de cache de spark. Notre approche proposée permet d'accélérer les temps d'indexation, et de classification. Le système proposé est testé sur des bases d'images à grand échelle telles que ImageClef2008 [8] et ImageNet, et dans un cluster d'un seul nœud et multi-nœuds constitué de cinq nœuds sous grid5000 dans le cluster de Rennes [9]. Le Chapitre 5 présente un système de classification des faces basé sur la race, en utilisant des bases d'images de faces à grand échelle sous la plateforme Spark, dont, nous avons fait une comparaison entre notre système proposé qui utilise la régression logistique avec les différents classificateurs tels que SVM linéaires, naïfs bayé-

8. ImageCLEFphoto2008 : http ://www.imageclef.org/photodata
9. Grid5000 : Home :https ://www.grid5000.fr

siens (NB), forêts aléatoires (RF) et les arbres de décision (DT) sous Spark. L'évaluation de notre système proposée a été effectuée en combinant de deux grandes bases d'images de faces CAS-PEAL (partie de pose) [12] et Color FERET [13]. Par comparaison, notre méthode avec les classificateurs SVM linéaires, bayésiens naïfs (NB), forêts aléatoires (RF) et arbres de décision (DT) de Spark, nous avons obtenu une précision moyenne de 99,99%. Aussi, notre méthode proposée a été comparée à différentes méthodes de l'état de l'art de classification des races, les résultats montrent que notre méthode est très compétitive.

En tant que travaux futurs, le système parallèle de recherche d'images par le contenu basé sur les plateformes Spark et Tachyon (CBIR-S) pourraient être amélioré en utilisant des algorithmes plus efficaces en termes de précision sur des clusters et des bases de données plus grands. Et pour étendre la méthode proposée pour la classification de faces à grande échelle basée sur la race (ethnicité) sous la plateforme Spark, nous pourrions inclure davantage de catégories raciales, de problèmes multi-classes (classification selon le sexe et l'âge) et tester la méthode proposée en termes de temps sur le cluster Spark distribué et des données plus volumineuses.

Bibliographie

[1] Jeffrey Dean and Sanjay Ghemawat. Mapreduce : simplified data processing on large clusters. *Communications of the ACM*, 51(1) :107–113, 2008.

[2] Eric Sammer. *Hadoop operations*. " O'Reilly Media, Inc.", 2012.

[3] Aditya B Patel, Manashvi Birla, and Ushma Nair. Addressing big data problem using hadoop and map reduce. In *2012 Nirma University International Conference on Engineering (NUiCONE)*, pages 1–5. IEEE, 2012.

[4] Tom White. *Hadoop : The definitive guide*. " O'Reilly Media, Inc.", 2012.

[5] Nicu Sebe, Ira Cohen, Ashutosh Garg, and Thomas S Huang. *Machine learning in computer vision*, volume 29. Springer Science & Business Media, 2005.

[6] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. Face description with local binary patterns : Application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 28(12) :2037–2041, 2006.

[7] Jean Ponce, Tamara L Berg, Mark Everingham, David A Forsyth, Martial Hebert, Svetlana Lazebnik, Marcin Marszalek, Cordelia Schmid, Bryan C Russell, Antonio Torralba, et al. Dataset issues in object recognition. In *Toward category-level object recognition*, pages 29–48. Springer, 2006.

[8] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

[9] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2) :303–338, 2010.

[10] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3) :211–252, 2015.

[11] Gidudu Anthony, Hulley Greg, and Marwala Tshilidzi. Classification of images using support vector machines. *arXiv preprint arXiv :0709.3967*, 2007.

[12] Wen Gao, Bo Cao, Shiguang Shan, Xilin Chen, Delong Zhou, Xiaohua Zhang, and Debin Zhao. The cas-peal large-scale chinese face database and baseline evaluations. *IEEE Transactions on Systems, Man, and Cybernetics-Part A : Systems and Humans*, 38(1) :149–161, 2008.

[13] P Jonathon Phillips, Harry Wechsler, Jeffery Huang, and Patrick J Rauss. The feret database and evaluation procedure for face-recognition algorithms. *Image and vision computing*, 16(5) :295–306, 1998.

[14] Pedro M Domingos. A few useful things to know about machine learning. *Commun. acm*, 55(10) :78–87, 2012.

[15] T Huang. Computer vision : Evolution and promise. 1996.

[16] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image processing, analysis, and machine vision*. Cengage Learning, 2014.

[17] Reinhard Klette. *Concise computer vision*. Springer, 2014.

[18] S Nagabhushana. *Computer vision and image processing*. New Age International, 2005.

[19] Amir Arsalan Soltani, Haibin Huang, Jiajun Wu, Tejas D Kulkarni, and Joshua B Tenenbaum. Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks. In *The IEEE conference on computer vision and pattern recognition (CVPR)*, volume 3, page 4, 2017.

[20] Fred Turek. Machine vision fundamentals, how to make robots see. *NASA Tech Briefs*, 35(6) :60–62, 2011.

[21] Carsten Steger, Markus Ulrich, and Christian Wiedemann. *Machine vision algorithms and applications*. John Wiley & Sons, 2018.

[22] Richard Szeliski. *Computer vision : algorithms and applications*. Springer Science & Business Media, 2010.

[23] Ying Wu. An introduction to computer vision.

[24] Sang-Il Oh and Hang-Bong Kang. Object detection and classification by decision-level fusion for intelligent vehicle systems. *Sensors*, 17(1) :207, 2017.

[25] Xiaogang Wang et al. Deep learning in object recognition, detection, and segmentation. *Foundations and Trends® in Signal Processing*, 8(4) :217–382, 2016.

[26] Saliha Mezzoudj, Rachid Seghir, Yassmina Saadna, et al. A parallel content-based image retrieval system using spark and tachyon frameworks. *Journal of King Saud University-Computer and Information Sciences*, 2019.

[27] Xiangxin Zhu and Deva Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *2012 IEEE conference on computer vision and pattern recognition*, pages 2879–2886. IEEE, 2012.

[28] Nikos Papamarkos, I Spiliotis, and Anastasios Zoumadakis. Character recognition by signature approximation. *International journal of pattern recognition and artificial intelligence*, 8(05) :1171–1187, 1994.

[29] Matthew A Turk and Alex P Pentland. Face recognition using eigenfaces. In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 586–591. IEEE, 1991.

[30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3) :211–252, 2015.

[31] Khushboo Khurana and Reetu Awasthi. Techniques for object recognition in images and multi-object detection. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 2(4) :pp–1383, 2013.

[32] Bangalore S Manjunath, Philippe Salembier, and Thomas Sikora. *Introduction to MPEG-7 : multimedia content description interface*, volume 1. John Wiley & Sons, 2002.

[33] Majid Mirmehdi. *Handbook of texture analysis*. Imperial College Press, 2008.

[34] Francesco Bianconi and Antonio Fernández. Evaluation of the effects of gabor filter parameters on texture classification. *Pattern Recognition*, 40(12) :3325–3335, 2007.

[35] Robert M Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5) :786–804, 1979.

[36] Dong-Chen He and Li Wang. Texture unit, texture spectrum, and texture analysis. *IEEE transactions on Geoscience and Remote Sensing*, 28(4) :509–512, 1990.

[37] Farzin Mokhtarian, Sadegh Abbasi, and Josef Kittler. Efficient and robust retrieval by shape content through curvature scale space. In *Image Databases and Multi-Media Search*, pages 51–58. World Scientific, 1997.

[38] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2) :91–110, 2004.

[39] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf : Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[40] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape context : A new descriptor for shape matching and object recognition. In *Advances in neural information processing systems*, pages 831–837, 2001.

[41] Ismail Khalid Kazmi, Lihua You, and Jian Jun Zhang. A survey of 2d and 3d shape descriptors. In *Computer graphics, imaging and visualization (cgiv), 2013 10th international conference*, pages 1–10. IEEE, 2013.

[42] Maneela Jain and Pushpendra Singh Tomar. Review of image classification methods and techniques. *International journal of engineering research and technology*, 2(8), 2013.

[43] W Chao. Machine learning tutorial. *National Taiwan University*, 2011.

[44] S Neelamegam and E Ramaraj. Classification algorithm in data mining : An overview. *International Journal of P2P Network Trends and Technology (IJPTT)*, 4(8) :369–374, 2013.

[45] M Anthony Wong and Tom Lane. A kth nearest neighbour clustering procedure. In *Computer Science and Statistics : Proceedings of the 13th Symposium on the Interface*, pages 308–311. Springer, 1981.

[46] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013.

[47] Scott Menard. *Applied logistic regression analysis*, volume 106. Sage, 2002.

[48] Thorsten Joachims. Text categorization with support vector machines : Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[49] Christina Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel : A string kernel for svm protein classification. In *Biocomputing 2002*, pages 564–575. World Scientific, 2001.

[50] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[51] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3) :660–674, 1991.

[52] Mahesh Pal. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1) :217–222, 2005.

[53] Ahmed Oussous, Fatima-Zahra Benjelloun, Ayoub Ait Lahcen, and Samir Belfkih. Big data technologies : A survey. *Journal of King Saud University-Computer and Information Sciences*, 30(4) :431–448, 2018.

[54] Raghavendra Kune, Pramod Kumar Konugurthi, Arun Agarwal, Raghavendra Rao Chillarige, and Rajkumar Buyya. The anatomy of big data computing. *Software : Practice and Experience*, 46(1) :79–105, 2016.

[55] Desamparados Blazquez and Josep Domenech. Big data sources and methods for social and economic analyses. *Technological Forecasting and Social Change*, 130 :99–113, 2018.

[56] Alberto Fernández, Sara del Río, Abdullah Bawakid, and Francisco Herrera. Fuzzy rule based classification systems for big data with mapreduce : granularity analysis. *Advances in Data Analysis and Classification*, 11(4) :711–730, 2017.

[57] Lu Lu, Xuanhua Shi, Hai Jin, Qiuyue Wang, Daxing Yuan, and Song Wu. Morpho : a decoupled mapreduce framework for elastic cloud computing. *Future Generation Computer Systems*, 36 :80–90, 2014.

[58] Seyed Nima Khezr and Nima Jafari Navimipour. Mapreduce and its applications, challenges, and architecture : a comprehensive review and directions for future research. *Journal of Grid Computing*, 15(3) :295–321, 2017.

[59] Vignesh Prajapati. *Big data analytics with R and Hadoop*. Packt Publishing Ltd, 2013.

[60] Holden Karau, Andy Konwinski, Patrick Wendell, and Matei Zaharia. *Learning spark : lightning-fast big data analysis*. " O'Reilly Media, Inc.", 2015.

[61] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark : Cluster computing with working sets. *HotCloud*, 10(10-10) :95, 2010.

[62] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, pages 1–10. Ieee, 2010.

[63] Haoyuan Li, Ali Ghodsi, Matei Zaharia, Scott Shenker, and Ion Stoica. Tachyon : Reliable, memory speed storage for cluster computing frameworks. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 1–15. ACM, 2014.

[64] Sylvain Gault. *Improving MapReduce Performance on Clusters*. PhD thesis, Ecole normale supérieure de lyon-ENS LYON, 2015.

[65] Thierry Monteil. *Du cluster à la grille sous l'angle de la performance*. PhD thesis, Institut National Polytechnique de Toulouse-INPT, 2010.

[66] Ian Foster and Carl Kesselman. The grid : Blueprint for a future computing inf, 1999.

[67] Ian Foster. What is the grid ?-a three point checklist. *GRIDtoday*, 1(6), 2002.

[68] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.

[69] Said Jai-Andaloussi, Mathieu Lamard, Guy Cazuguel, Hamid Tairi, Mohamed Meknassi, Beatrice Cochener, and Christian Roux. Content based medical image retrieval based on bemd : Optimization of a similarity metric. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 3069–3072. IEEE, 2010.

[70] Ricardo da Silva Torres and Alexandre X Falcao. Content-based image retrieval : theory and applications. *RITA*, 13(2) :161–185, 2006.

[71] Chunhao Gu and Yang Gao. A content-based image retrieval system based on hadoop and lucene. In *2012 Second International Conference on Cloud and Green Computing (CGC)*, pages 684–687. IEEE, 2012.

[72] Jeffrey Shafer, Scott Rixner, and Alan L Cox. The hadoop distributed filesystem : Balancing portability and performance. In *Performance Analysis of Systems & Software (ISPASS), 2010 IEEE International Symposium on*, pages 122–133. IEEE, 2010.

[73] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1) :21–27, 1967.

[74] Jesús Maillo, Isaac Triguero, and Francisco Herrera. A mapreduce-based k-nearest neighbor approach for big data classification. In *Trustcom/BigDataSE/ISPA, 2015 IEEE*, volume 2, pages 167–172. IEEE, 2015.

[75] Ryszard S Choraś. Content-based image retrieval -a survey. In *Biometrics, Computer Security Systems and Artificial Intelligence Applications*, pages 31–44. Springer, 2006.

[76] Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, et al. Query by image and video content : The qbic system. *computer*, 28(9) :23–32, 1995.

[77] Jeffrey R Bach, Charles Fuller, Amarnath Gupta, Arun Hampapur, Bradley Horowitz, Rich Humphrey, Ramesh C Jain, and Chiao-Fe Shu. Virage image search engine : an open framework for image management. In *Storage and retrieval for still image and video databases IV*, volume 2670, pages 76–88. International Society for Optics and Photonics, 1996.

[78] Yong Rui, Thomas S Huang, and Sharad Mehrotra. Content-based image retrieval with relevance feedback in mars. In *Image Processing, 1997. Proceedings., International Conference on*, volume 2, pages 815–818. IEEE, 1997.

[79] Jing Zhang, Xianglong Liu, Junwu Luo, and Bo Lang. Dirs : Distributed image retrieval system based on mapreduce. In *5th International Conference on Pervasive Computing and Applications*, pages 93–98. IEEE, 2010.

[80] USN Raju, Shibin George, V Sairam Praneeth, Ranjeet Deo, and Priyanka Jain. Content based image retrieval on hadoop framework. In *2015 IEEE International Congress on Big Data*, pages 661–664. IEEE, 2015.

[81] Subrahmanyam Murala, RP Maheshwari, and R Balasubramanian. Local tetra patterns : a new feature descriptor for content-based image retrieval. *IEEE Transactions on Image Processing*, 21(5) :2874–2886, 2012.

[82] Luca Costantini and Raffaele Nicolussi. Performances evaluation of a novel hadoop and spark based system of image retrieval for huge collections. *Advances in Multimedia*, 2015 :11, 2015.

[83] Noha A Sakr, Ali I ELdesouky, and Hesham Arafat. An efficient fast-response content-based image retrieval framework for big data. *Computers & Electrical Engineering*, 54 :522–538, 2016.

[84] Yixiao Duan, Linhua Jiang, Xiao Lin, and Xiaodong Chen. An improved content based image retrieval system on apache spark. In *2016 International Conference on Mechatronics, Control and Automation Engineering (MCAE 2016)*, pages 107–110, 2016.

[85] Michal Lagiewka, Marcin Korytkowski, and Rafal Scherer. Distributed image retrieval with color and keypoint features. In *INnovations in Intelligent SysTems and Applications (INISTA), 2017 IEEE International Conference on*, pages 45–50. IEEE, 2017.

[86] Wei Dong, Charikar Moses, and Kai Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World wide web*, pages 577–586. ACM, 2011.

[87] Ge Song, Justine Rochas, Fabrice Huet, and Frédéric Magoules. Solutions for processing k nearest neighbor joins for massive data on mapreduce. In *Parallel, Distributed and Network-Based Processing (PDP), 2015 23rd Euromicro International Conference on*, pages 279–287. IEEE, 2015.

[88] Jesús Maillo, Isaac Triguero, and Francisco Herrera. A mapreduce-based k-nearest neighbor approach for big data classification. In *Trustcom/BigDataSE/ISPA, 2015 IEEE*, volume 2, pages 167–172. IEEE, 2015.

[89] Qin Ding and Robert Boykin. A framework for distributed nearest neighbor classification using hadoop. *Journal of Computational Methods in Sciences and Engineering*, 17(S1) :S11–S19, 2017.

[90] Jeffrey Dean and Sanjay Ghemawat. Mapreduce : simplified data processing on large clusters. *Communications of the ACM*, 51(1) :107–113, 2008.

[91] Holden Karau, Andy Konwinski, Patrick Wendell, and Matei Zaharia. *Learning spark : lightning-fast big data analysis*. " O'Reilly Media, Inc.", 2015.

[92] Marko Heikkilä, Matti Pietikäinen, and Cordelia Schmid. Description of interest regions with local binary patterns. *Pattern recognition*, 42(3) :425–436, 2009.

[93] Olli Lahdenoja, Janne Maunu, Mika Laiho, and Ari Paasio. A massively parallel algorithm for local binary pattern based face recognition. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pages 4–pp. IEEE, 2006.

[94] Siyao Fu, Haibo He, and Zeng-Guang Hou. Learning race from face : A survey. *IEEE transactions on pattern analysis and machine intelligence*, 36(12) :2483–2509, 2014.

[95] Sarfaraz Masood, Shubham Gupta, Abdul Wajid, Suhani Gupta, and Musheer Ahmed. Prediction of human ethnicity from facial images using neural networks. In *Data Engineering and Intelligent Computing*, pages 217–226. Springer, 2018.

[96] Ghulam Muhammad, Muhammad Hussain, Fatmah Alenezy, George Bebis, Anwar M Mirza, and Hatim Aboalsamh. Race classification from face images using local descriptors. *International Journal on Artificial Intelligence Tools*, 21(05) :1250019, 2012.

[97] Hu Han, Charles Otto, Xiaoming Liu, and Anil K Jain. Demographic estimation from face images : Human vs. machine performance. *IEEE transactions on pattern analysis and machine intelligence*, 37(6) :1148–1161, 2015.

[98] Holden Karau, Andy Konwinski, Patrick Wendell, and Matei Zaharia. *Learning spark : lightning-fast big data analysis*. " O'Reilly Media, Inc.", 2015.

[99] Srinivas Gutta, Harry Wechsler, and P Jonathon Phillips. Gender and ethnic classification of face images. In *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*, pages 194–199. IEEE, 1998.

[100] Srinvis Gutta and Harry Wechsler. Gender and ethnic classification of human faces using hybrid classifiers. In *Neural Networks, 1999. IJCNN'99. International Joint Conference on*, volume 6, pages 4084–4089. IEEE, 1999.

[101] Srinivas Gutta, Jeffrey RJ Huang, P Jonathon, and Harry Wechsler. Mixture of experts for classification of gender, ethnic origin, and pose of human faces. *IEEE Transactions on neural networks*, 11(4) :948–960, 2000.

[102] Xiaoguang Lu, Anil K Jain, et al. Ethnicity identification from face images. In *Proceedings of SPIE*, volume 5404, pages 114–123, 2004.

[103] Yongsheng Ou, Xinyu Wu, Huihuan Qian, and Yangsheng Xu. A real time race classification system. In *Information Acquisition, 2005 IEEE International Conference on*, pages 6–pp. IEEE, 2005.

[104] F Saei Manesh, Mohammad Ghahramani, and Yap-Peng Tan. Facial part displacement effect on template-based gender and ethnicity classification. In *Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on*, pages 1644–1649. IEEE, 2010.

[105] S Md Mansoor Roomi, SL Virasundarii, S Selvamegala, S Jeevanandham, and D Hariharasudhan. Race classification based on facial features. In *Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), 2011 Third National Conference on*, pages 54–57. IEEE, 2011.

[106] S Hma Salah, H Du, and N Al-Jawad. Fusing local binary patterns with wavelet features for ethnicity identification. In *Proceedings of World Academy of Science, Engineering and Technology*, number 79, page 471. World Academy of Science, Engineering and Technology (WASET), 2013.

[107] Cunjian Chen and Arun Ross. Local gradient gabor pattern (lggp) with applications in face recognition, cross-spectral matching, and soft biometrics. In *Biometric and Surveillance Technology for Human and Activity Identification X*, volume 8712, page 87120R. International Society for Optics and Photonics, 2013.

[108] Hajra Momin and Jules-Raymond Tapamo. A comparative study of a face components based model of ethnic classification using gabor filters. *Appl. Math*, 10(6) :2255–2265, 2016.

[109] Yiting Xie, Khoa Luu, and Marios Savvides. A robust approach to facial ethnicity classification on large scale face databases. In *Biometrics : Theory, Applications and Systems (BTAS), 2012 IEEE Fifth International Conference on*, pages 143–149. IEEE, 2012.

[110] Inzamam Anwar and Naeem Ul Islam. Learned features are better for ethnicity classification. *Cybernetics and Information Technologies*, 17(3) :152–164, 2017.

[111] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1) :51–59, 1996.

[112] Jeffrey Shafer, Scott Rixner, and Alan L Cox. The hadoop distributed filesystem : Balancing portability and performance. In *Performance Analysis of Systems & Software (ISPASS), 2010 IEEE International Symposium on*, pages 122–133. IEEE, 2010.

[113] Rodrigo Verschae, Javier Ruiz-del Solar, and Mauricio Correa. Gender classification of faces using adaboost. In *Iberoamerican Congress on Pattern Recognition*, pages 68–78. Springer, 2006.

[114] Gregory Shakhnarovich, Paul A Viola, and Baback Moghaddam. A unified learning framework for real time face detection and classification. In *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*, pages 16–23. IEEE, 2002.

[115] Dalia Shouman Ibrahim and Salma Hamdy. Parallel architecture for face recognition using mpi. *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, 8(1) :425–430, 2017.

[116] Jaafar Sadiq Qateef and Ammar Awad Kazm. Facial expression recognition via mapreduce assisted k-nearest neighbor algorithm. *International Journal of Computer Science and Information Security*, 14(2) :170, 2016.