Continuous memories for representing sets of vectors and image collections

Ahmet Iscen

ANNÉE 2017

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Bretagne Loire

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

École doctorale 601 « Mathématiques et Sciences et Technologies de l'Information et de la Communication (MATHSTIC) »

présentée par

Ahmet ISCEN

préparée à l'unité de recherche Inria Rennes - Bretagne Atlantique
Institut national de recherche en informatique et en automatique
Université de Rennes 1

Continuous memories for representing sets of vectors and image collections

Thèse soutenu à Rennes le 25/09/2017

Acknowledgements

I would like give my sincere thanks to my advisor Teddy Furon for his outstanding guidance and advice over three years.

I would also like to thank the exceptional scientists that I was fortunate to work with during my PhD, including Hervé Jégou, Vincent Gripon, Michael Rabbat, Ondřej Chum, Giorgos Tolias and Yannis Avrithis.

A special thanks to my reviewers and jury members, Josef Sivic, Patrick Pérez, Tinne Tuytelaars, Ondřej Chum, Hervé Jégou and Patrick Bouthemy, for evaluating my work and giving valuable feedback.

I am also very thankful that I had a chance to meet many wonderful colleagues and friends during my PhD. Past three years have been fantastic thanks to them.

Last but not least, I would like to thank my parents and sisters, who have always provided me with their support and unconditional love throughout my life.

Contents

Résumé en français

Nous vivons à l'ère du Big Data. Internet et les réseaux sociaux sont omniprésents dans nos vies numériques et rendent accessible une quantité énorme de données. L'accès à l'image était souvent limité aux albums photo de famille il y a encore quelques décennies. Maintenant, nous avons accès à des millions de photos en quelques clics. Ceci rend la vision et la compréhension de l'image par l'ordinateur d'autant plus important pour gérer et analyser ces grandes collections d'images.

La vision par ordinateur se définit comme l'art d'enseigner aux ordinateurs comment voir. Les images sont traduites en des formes mathématiques sur lesquelles des opérations numériques sont appliquées. La vision par ordinateur utilise les avancées de beaucoup de domaines comme l'apprentissage automatique, le traitement du signal, les mathématiques appliquées et la géométrie. Les problèmes types en vision par ordinateur sont par exemple la classification d'images/vidéos, la recherche d'images, la génération d'images, la détection d'objet, la reconstruction 3D et plein d'autres encore. Plus récemment, la vision par ordinateur est capitale dans les voitures autonomes.

Cette thèse traite principalement du problème de la recherche d'image. Dans un scénario type, on nous donne une image requête qui contient un objet important pour l'utilisateur. Cet objet peut-être n'importe quoi, un bâtiment ou lieu connu, un objet du quotidien... Le but est de retrouver des images contenant la même instance de l'objet d'intérêt dans une grande collection contenant des millions d'images. Ce problème n'est pas de la classification d'image ou de la détection d'objet car on recherche la même instance. En classification ou en détection d'objet, le but est de décider si les objets présents dans l'image appartiennent à des catégories génériques telles que 'voiture', 'bâtiments', 'chats' etc. Notre problème est plus spécifique. Si un utilisateur soumet une photo d'un lieu particulier, nous devons retrouver ce même lieu, et non pas n'importe quel bâtiment. Par conséquent, il n'est pas possible d'entrainer un classifieur pour cet objet requête particulier, le nombre d'objets d'intérêt étant virtuellement infini.

Les Défis de la Recherche d'Images

Un des problèmes clés en vision par ordinateur est la représentation du contenu visuel. Un grand travail de recherche a été fait dans ce contexte, allant du design 'maison' des descripteurs SIFT [Low04] aux récents descripteurs basés CNN (réseau de neurones convolutionnels) [KSH12, SZ14]. La représentation visuelle reste un problème aussi en recherche d'image, car l'objet d'intérêt est présent dans les images avec beaucoup de variations concernant sa taille, son illumination, les occlusions, etc. De plus, la recherche d'image doit être efficace. Cette contrainte est cruciale en pratique. L'utilisateur doit retrouver des contenus pertinents sans trop attendre.

Efficacité

Grâce à la représentation du contenu visuel, la recherche d'images se réduit à la recherche des plus proches voisins dans un espace affine à grande dimension. On mesure la similarité par une métrique comme la similarité cosinus ou la distance Euclidienne entre le vecteur requête et les vecteurs de la collection. Les ordinateurs actuels sont très rapides pour effectuer ce genre de calcul, mais les descripteurs ont une grande dimension et le nombre de similarité à calculer est aussi grand que la taille de la collection. Ce problème est donc délicat quand on manipule de très grande collection d'images. En plus de la complexité de la recherche, il ne faut pas oublier l'espace mémoire pour stocker tous les vecteurs de la collection. L'indéxation est une solution connue qui évite la comparaison exhaustive entre le vecteur requête et tous les vecteurs de la base. Par exemple, on peut partitionner l'espace en régions et ne chercher que dans les regions proches de la requête. Cependant, ces approaches perdent de leur efficacité en grande dimension.

Représentation Visuelle

Beaucoup de système traditionnels décrivent une image par un ensemble de descripteurs locaux, tels que les SIFT [Low04]. Cependant, cela ne constitue pas une solution efficace car la complexité et la mémoire croit de plusieurs facteurs avec le nombre d'images. Des travaux ont proposé des stratégies de regroupement pour agréger un ensemble de descripteurs locaux en un seul descripteur global [SZ03, SPMV13, JPD$^+$12]. Plus récemment, les descripteurs basés CNN ont atteint des performances état de l'art dans de nombreux benchmarks en recherche d'images [RTC16, GARL16b]. Cependant, les instances d'objet sous de sévères variations (objet de petite taille dans l'image, objet occulté, objet noyé dans l'arrière plan, objet pris par un angle de vue différent) demeurent un problème. Une approche est l'expansion de requête ("query expansion"). L'idée est de faire une première recherche avec l'image requête, pour obtenir les plus proches voisins ; puis, de rechercher en prenant ces plus proches voisins comme requête. Cela aide à retrouver des images contenant l'objet d'intérêt mais sous des variations plus sévères.

Contenu de la Thèse

Cette thèse apporte des solutions aux défis présentés dans la Section . Ces solutions s'appuient sur une représentation continue d'un ensemble de vecteurs.

Vecteur Mémoire pour l'Indéxation

Ce chapitre se rapporte à la recherche de vecteurs similaires à un vecteur requête dans une grande collection. De nombreux travaux étudient comment le fléau de la dimension (la taille des vecteurs) rend les techniques d'indéxation inefficaces [WSB98, ML14]. L'article [ML14] par exemple analyse expérimentalement la méthode FLANN et observe que cette méthode de l'état de l'art donne des performances pauvres sur des vecteurs synthétiques en grande dimension. Les auteurs concluent que "les collections de vecteurs aléatoires constituent un des problèmes les plus difficiles en recherche de plus proches voisins".

La contribution de ce chapitre est un algorithme de recherche par similarité spécifique aux vecteurs de grande dimension, comme ceux récemment utilisés en vision par ordinateur pour représenter des images [PD07, JDSP10]. La technique d'indéxation consiste à créer des unités de mémoire, chacune étant associée à plusieurs vecteurs de la collection. Un représentant, appelé vecteur mémoire, est produit pour chaque unité de mémoire. Il est construit de telle manière qu'il est possible de prédire si le vecteur requête est similaire à au moins un vecteur de l'unité mémoire associée.

Soit une unité mémoire composée des vecteurs $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^d$ où $\mathbf{x}_i$ est le descripteur de l'image i de la collection. Son représentant vecteur mémoire est tel que, pour un vecteur requête $\mathbf{q}$ donné, nous pouvons construire un test statistique de similarité pour répondre à la question: est-ce que $\mathbf{q}$ est une quasi-copie d'au moins un des vecteurs de l'unité mémoire ?

Nous proposons deux constructions de vecteur mémoire. La première est simplement la somme des vecteurs de l'unité mémoire :

$$\mathbf{m}(\mathcal{X}) = \sum_{\mathbf{x} \in \mathcal{X}} \mathbf{x}, \tag{1}$$

où nous supposons que $\mathcal{X}$ est composée de n vecteurs différents. Bien que très simple, cette construction n'est pas dénuée d'intérêt en grande dimension.

L'autre construction de vecteur mémoire est optimisée suivant la méthode suivante. Soit $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ la matrice $d \times n$ représentant l'unité mémoire. Nous imposons que, pour tout $i \in \{1, \ldots, n\}$, ${\mathbf{x}_i}^\top \mathbf{m}(\mathcal{X}) = 1$ exactement. Autrement dit, nous recherchons $\mathbf{m}(\mathcal{X})$ parmi les vecteurs $\mathbf{m}$ satisfaisant $\mathbf{X}^\top \mathbf{m} = \mathbf{1}_n$ où $\mathbf{1}_n$ est le vecteur dont les n composantes sont égales à 1. Si un tel vecteur existe, alors pour $\mathbf{q} = \mathbf{x}_i$, $\mathbf{q}^\top \mathbf{m} = 1$. Nous éliminons ainsi les interférences avec les autres vecteurs de l'unité mémoire, qui étaient la source principale de bruit dans la première construction.

Sous l'hypothèse $\mathcal{H}_0$ où le vecteur requête est aléatoire, la variance du score est en $\|\mathbf{m}\|^2/d$. Nous cherchons donc le représentant $\mathbf{m}$ de module $\|\mathbf{m}\|$ minimal sous la contrainte $\mathbf{X}^\top \mathbf{m} = \mathbf{1}_n$. La solution est donnée grâce à la pseudo-inverse de Moore-Penrose [RM72] :

$$\mathbf{m}^\star = (\mathbf{X}^+)^\top \mathbf{1}_n. \tag{2}$$

Nous analysons les performances de ces deux constructions de vecteur mémoire théoriquement et expérimentalement. Pour une assignation aléatoire des vecteurs de la collection dans les unités de mémoire, la construction pinv donne de meilleurs résultats que la simple somme. Cependant, lorsque les vecteurs de la collection regroupés dans une unité mémoire partagent une certaine corrélation, la construction sommme donne un test d'hypothèse aussi fiable que la construction pinv. Mais, la construction pinv utilisée dans l'assignation supervisée de vecteurs corrélés produit des unités mémoires plus homogènes en taille. Cela donne des temps de réponse plus uniformes d'une requête à une autre. L'assignation supervisée en revanche est plus complexe, mais comme elle est réalisée en amont de la recherche, ce surplus de complexité est souvent négligé dans la littérature. Nous avons eu à coeur de proposer une assignation tout de même efficace pour manipuler des grandes collections de vecteurs. Elle est basée sur une assignation faiblement supervisée fonctionnant par lots. Ceci ne diminue pas les performances de la recherche in fine.

Applications des Vecteurs Mémoires

Ce chapitre donne une implémentation pratique d'une recherche par similarité dans une grande collection d'images à l'aide de vecteurs mémoires. De plus, il étudie une application du concept de vecteur mémoire à la recherche d'information avec des contraintes de sécurité et confidentialité.

La section 4.1 répond au cahier des charges d'une recherche par similarité dans une grande collection d'images. Nous montrons que les vecteurs mémoires sont facilement 'indéxables' par des techniques classiques de partitionnement d'espace comme FLANN, contrairement aux descripteurs d'image bruts. Cette indéxation rend la recherche encore plus efficace en termes de temps de réponse et d'espace mémoire pour des collections de millions d'images.

La section 4.2 étudie l'usage des vecteurs mémoires pour le respect de la confidentialité. Le but est de construire un système de recherche d'information multimédia efficace protégeant les données. Le scénario réunit les trois acteurs suivants : l'utilisateur, le serveur, et le propriétaire de la collection de données. Nous montrons l'utilité des vecteurs mémoires en terme d'efficacité de la recherche et de respect de la confidentialité des données de chaque acteur. Les vecteurs mémoires produisent une seconde ligne de défense rendant le vol des vecteurs de la collection par un utilisateur "curieux" plus difficile.

Optimisation du Test par Groupe par Factorisation de Matrice

Ce chapitre poursuit l'idée du test par groupe, c'est à dire de déduire quels vecteurs parmi les N vecteurs de la collection sont similaires à la requête à partir de seulement

M mesures. Plutôt qu'une assignation des vecteurs de la collection dans des unités mémoires suivie d'une construction fixe des vecteurs mémoires, nous formulons le problème sous la forme d'une optimization jointe de l'assignation et de la construction. En levant la contrainte d'une assignation binaire (tel vecteur appartient ou pas à telle unité de mémoire), la version continue de ce problème d'optimisation se révèle être équivalent à un problème d'apprentissage de dictionnaire en représentation parcimonieuse. Nous appelons cette opération l'encodage, et nous nous restreignons à la famille des encodages linéaires :

$$\mathbf{Y} = \mathsf{enc}(\mathbf{X}) = \mathbf{X}\mathbf{G}^\top. \tag{3}$$

Pour une requête $\mathbf{q}$ donnée, nous calculons les scores des unités mémoires ainsi :

$$\mathbf{s} = \mathbf{q}^\top \mathbf{Y}. \tag{4}$$

Puis, nous estimons les similarités requête - vecteurs de la collection $\mathbf{c} = \mathbf{q}^\top \mathbf{X}$ à partir des mesures $\mathbf{s}$. Encore une fois, nous nous restreignons à un estimateur linéraire :

$$\hat{\mathbf{c}} = \mathsf{dec}(\mathbf{s}) = \mathbf{s}\mathbf{H}. \tag{5}$$

Nous proposons deux solutions. La première trouve les matrices $\mathbf{G} \in \mathbb{R}^{M\times N}$ et $\mathbf{H} \in \mathbb{R}^{M\times N}$ telles que les scores estimés $\hat{\mathbf{c}}$ et les scores exacts $\mathbf{c}$ soient les plus proches possibles. Cela revient à résoudre le problème suivant :

$$\min_{\mathbf{G},\mathbf{H}} \sum_{\mathbf{q}\in\mathcal{Q}} \|\mathbf{c} - \hat{\mathbf{c}}\|_2^2 = \min_{\mathbf{G},\mathbf{H}} \sum_{\mathbf{q}\in\mathcal{Q}} \|\mathbf{q}^T\mathbf{X} - \mathbf{q}^T\mathbf{X}\mathbf{G}^\top\mathbf{H}\|_2^2,$$

où $\mathcal{Q}$ est un ensemble de vecteurs requêtes typiques. En fait, nous utilisons la collection de vecteurs elle-même comme représentation de cet ensemble de requêtes typiques, remplacant ainsi $\mathbf{q}$ par $\mathbf{X}$ et la norme Euclidienne sur les vecteurs par la norme de Frobenius sur les matrice :

$$\min_{\mathbf{G},\mathbf{H}} \left\|\mathbf{X}^\top\mathbf{X} - \mathbf{X}^\top\mathbf{X}\mathbf{G}^\top\mathbf{H}\right\|_F^2. \tag{6}$$

Ce problème se résout en général par une décomposition en vecteurs propres. Soit $\mathbf{A} = \mathbf{X}^\top\mathbf{X}$ la matrice de Gram associée à la matrice $\mathbf{X}$. Etant symétrique et réelle, $\mathbf{A}$ est diagonalisable : $\mathbf{A} = \mathbf{U}\Lambda\mathbf{U}^\top$, avec $\mathbf{U}$ matrice unitaire ($\mathbf{U}^\top\mathbf{U} = \mathbf{U}\mathbf{U}^\top = \mathbf{I}_N$). Ainsi, on choisit $\mathbf{G}^\top = \mathbf{U}_M$ et $\mathbf{H} = \mathbf{U}_M^\top$, où $\mathbf{U}_M$ est la restriction de $\mathbf{U}$ aux valeurs propres les plus grandes en amplitude.

La deuxième solution applique des méthodes d'apprentissage de dictionnaire pour trouver des représentations parcimonieuses de la collection de vecteurs. La parcimonie améliore ici l'efficacité de la recherche. Nous voulons une approximation de $\mathbf{X}$ par $\mathbf{Y}\mathbf{H}$ où $\mathbf{H} \in \mathbb{R}^{M\times N}$ stocke les représentations parcimonieuses des vecteurs selon les colonnes du dictionnaire $\mathbf{Y} \in \mathbb{R}^{d\times M}$ (aussi appelées atomes). Cela conduit à ce problème d'optimisation :

$$\begin{array}{ll} \min\limits_{\mathbf{Y},\mathbf{H}} & \frac{1}{2}\|\mathbf{X} - \mathbf{Y}\mathbf{H}\|_F^2 + \lambda\|\mathbf{H}\|_1 \\ \text{sous contrainte} & \|\mathbf{y}_k\|_2 \leq 1 \text{ for all } 0 \leq k < M. \end{array}$$

Nos expériences montrent que la décomposition en vecteurs propres n'est opérante que pour de petites collections, alors que l'apprentissage du dictionnaire peut s'attaquer à des très grandes collections. Testé sur des collections de plus de 10^8 images, cette solution atteint des qualités de recherche comparables (et parfois meilleures) que la méthode exhaustive mais avec une complexité bien moindre.

Recherche sur des variétés géométriques

Les chapitres 6 and 7 porte sur l'expansion de la requête. Nous utilisons une représentation d'une collection d'images basée sur un graphe pondéré donnant les relations entre les vecteurs de la collection. Ce graphe sert pour retourner les images qui sont sur la même variété géométrique que le vecteur requête. Alors que le problème principal des chapitres précédents est d'améliorer l'efficacité de la recherche quitte à sacrifier la qualité des résultats retournés, ce chapitre explore ce compromis dans l'autre sens. On s'autorise une plus grande complexité dans le but d'améliorer la qualité de la recherche.

Le chapitre 6 introduit le mécanisme de diffusion régionale, qui prend en compte plusieurs vecteurs requêtes pour un même coût en complexité. L'image est découpée en région et un descripteur est extrait par région. Cette approche améliore nettement la recherche d'objets d'intérêt petits ou noyés dans l'arrière plan. Dans les mécanismes de diffusion sur graphe [PBMW99, ZWG+03, DB13], les vecteurs requêtes sont habituellement intégrés dans le graphe ; ils sont donc connus lors de l'indéxation. Notre nouvelle approche se libère de cette hypothèse. Elle gère des requêtes jamais vues sans surplus de complexité. La formule de la solution optimale est connue, mais tous les systèmes actuels y dérogent [DB13] sous prétexte de la trop grande complexité de son calcul. En fait, nous montrons que la solution alternative utilisée n'est que la résolution itérative d'un système linéaire. Comme la matrice associée est creuse et définie positive, la méthode du gradient conjugué est plus efficace donnant la solution en moins d'une seconde. Nous nous sommes servis du jeux de données INSTRE [WJ15], qui n'a pas recu beaucoup d'attention jusqu'à présent, afin d'étudier la corrélation entre la qualité des résultats retournés et la taille de l'objet d'intérêt dans les images. Nous proposons aussi un nouveau protocole d'expérimentation plus adapté à de futures comparaisons.

Nos expériences montrent que notre méthode améliore nettement la recherche de petits objets, et ce uniquement avec 5 à 10 descripteurs régionaux par image (ce sont des descripteurs issus de CNN). Une telle qualité de recherche nécessitait auparavant des milliers de descripteurs locaux. Le graphe des plus proches voisins n'est pas si grand. Nos performances sont égales à celles de l'état de l'art sur deux benchmarks bien connus.

Le chapitre 7 reproduit la même qualité de recherche que le chapitre 6, mais cette fois plus rapidement. La complexité est plus grande pendant la phase d'indéxation. Elle est linéaire avec la taille de la collection mais facilement parallèlisable. Elle prend quelques heures pour une grande collection d'images. La complexité de la recherche est quant à elle plus petite. L'expansion de la requête n'est plus qu'un simple post-traitement. La qualité de la recherche est presque parfaite par rapport à la vérité terrain.

Application à la Reconnaissance de Lieux

La reconnaissance de lieux s'explique ainsi : Nous avons une collection d'images avec leurs coordonnées GPS. Plusieurs images ont été prises à la même localisation, pouvant former un panorama. Plusieurs solutions traitent de ce problème comme une application de la recherche d'instances. L'utilisateur soumet une seule image d'où il est, et cette image joue le rôle de la requête. Ce chapitre propose d'améliorer la géolocalisation en calculant une similarité panorama à panorama, et non image à image.

Nous construisons une représentation d'un panorama en utilisant les vecteurs mémoires du chapitre 3. La grande différence étant que les unités mémoires sont maintenant constituées naturellement : l'assignation n'est plus aléatoire ou supervisée selon la proximité des vecteurs, mais elle groupe ensemble les images d'une même localisation GPS. L'espace mémoire est conservé car un vecteur décrit un panorama. La similarité panorama - panorama améliore grandement la géolocalisation par rapport à l'état de l'art. Nous montrons aussi dans la Section 8.3 qu'il surpasse même la recherche sur des variétés géométriques du chapitre 6.

Chapter 1

Introduction

We live in the age of big data. The rapid growth of social media and internet have taken over a large part of our lives, but also made available enormous amount of data. While our access to personal photography was mostly limited to family albums about a decade ago, we now have access to millions of photos with a single click. Thanks to these developments, computer vision and image understanding has become more important than ever.

Computer vision can be briefly defined as "teaching computers how to see". More specifically, images are translated into mathematical representations, and are applied various numeral operations. It makes use of the advancements of many other fields, such as machine learning, signal processing, applied math and geometry. Some of the computer vision problems include image/video classification, image/video retrieval, image generation, object detection, 3D reconstruction and many others. More recently, computer vision is being used in self-driving cars.

This thesis is mainly concerned about the problem of instance-based image retrieval. In a typical scenario, we are given a query image which contains an object of interest. This object can be anything from landmarks to daily objects. The goal is to return images containing the same instance of the queried object from a large collection, possibly containing millions of images. This problems differs from image classification or object detection due to its instance-based nature. In image classification or object detection, the goal is to classify objects into generic categories, such as cars, buildings, cats etc. Our problem is much more specific. If a user submits an image of a particular landmark, we must return the same landmark, not any other building. The same applies for any other object of interest. Hence, it is not possible to train classifiers (since the possibility of real world instances are infinite) and we must search the entire image collection with the given query. Visual example of a typical image retrieval framework is shown in Figure 1.1.

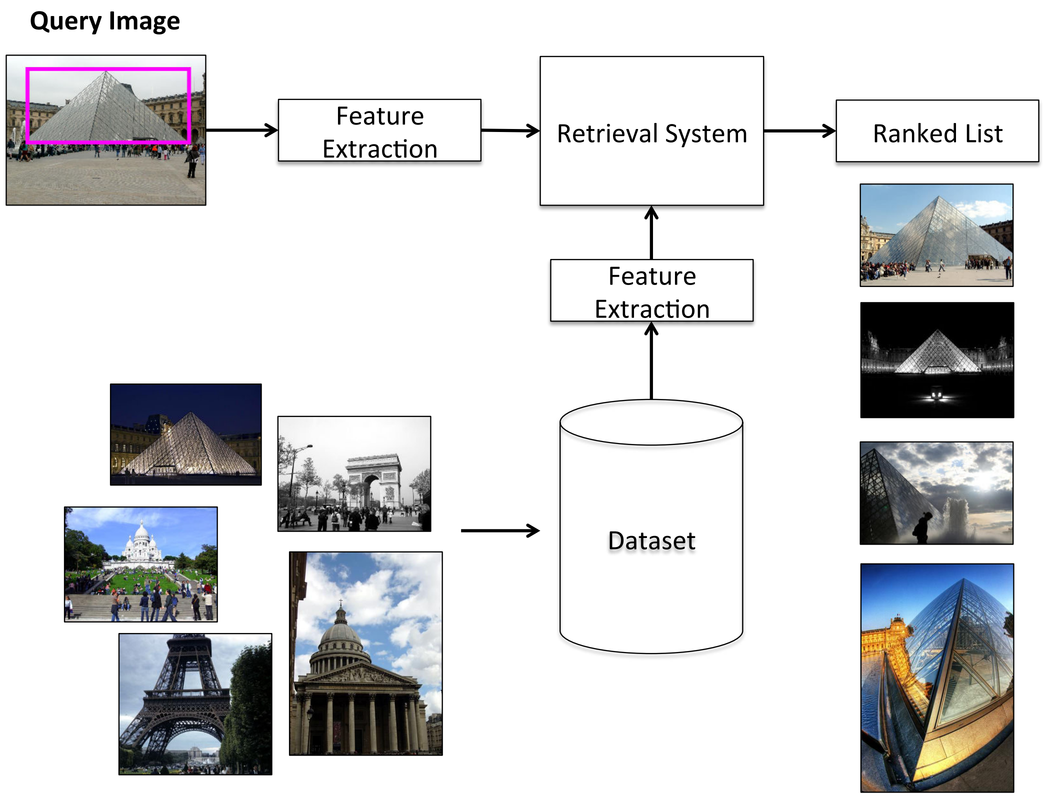


Figure 1.1: Example of a basic image retrieval framework. Dataset images are collected and feature extraction is applied. When a new query image is submitted, its feature representation is compared against all the dataset vectors, and a ranked list of images is returned to the user.

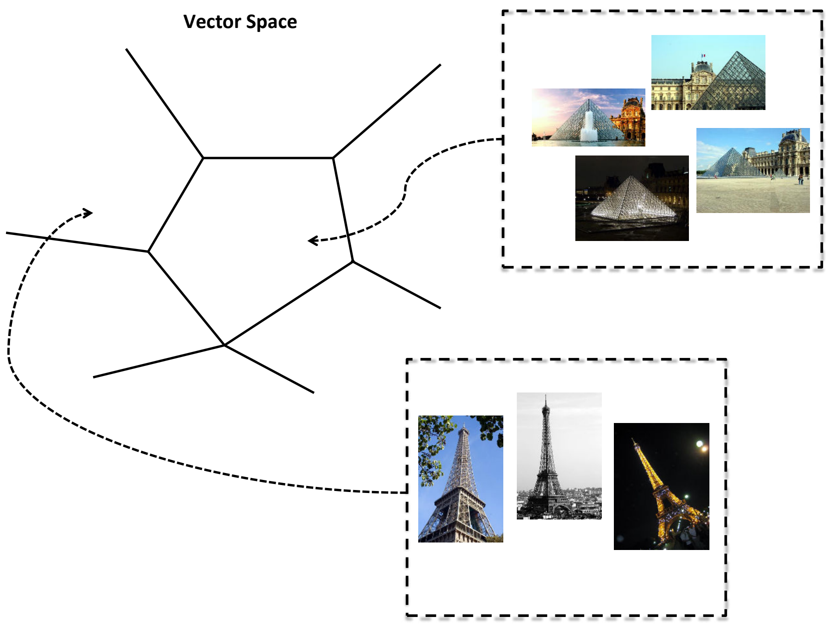


Figure 1.2: Visualization of space partitioning in $\mathbb{R}^2$. Similar images produce similar visual representations, and are assigned to the same bin.

1.1 Challenges

One of the main problems in computer vision, in general, is the representation of visual content. Significant amount of research has been done in this context, from traditional hand-designed SIFT features [Low04], to more modern CNN-based descriptors [KSH12, SZ14]. Visual representation also remains a challenge for image retrieval, thanks to different variations of the object of interest, such as scale and illumination changes, occlusion etc. In addition to visual representation, image retrieval also deals with the problem of efficiency. This is an absolutely crucial constraint for any proposed system. In practice, we must make sure that the user is able to retrieve the relevant content without waiting too long.

1.1.1 Efficiency

Given a visual representation of images, basic image search boils down to a nearest neighbors search in the vector space. This involves computing a similarity metric, such as cosine similarity or Euclidean distance, between the query vector and dataset vectors. While modern computers are relatively fast at making this kind of computations, image descriptors have high dimensionalities, and the complexity of such an operation grows linearly with the number of comparisons. Thus, this problem still remains important when considering a dataset with millions of images. In addition to search complexity, memory footprint of dataset vectors is another aspect of efficiency which must be considered. Typical solutions to this problem involve indexing, which avoids making

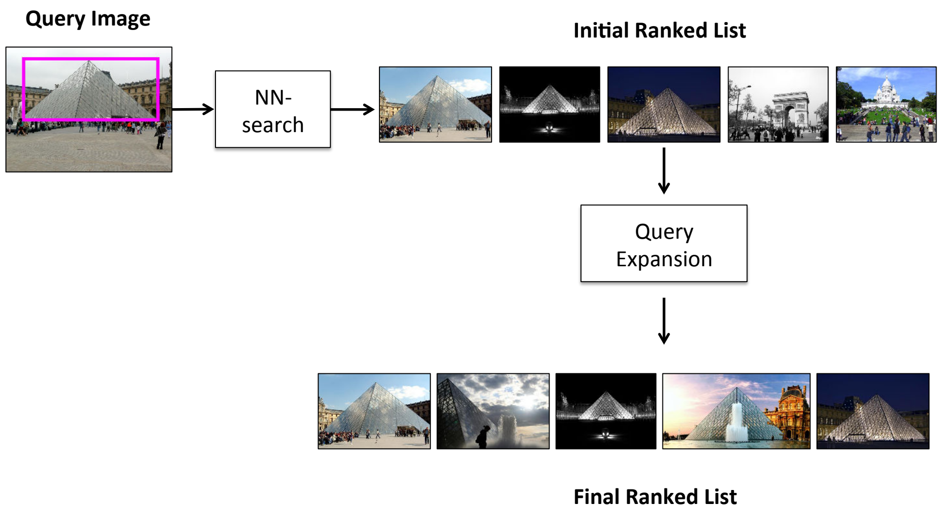


Figure 1.3: Query expansion techniques are usually applied to the initial ranked list returned by NN-search. It is a useful technique to retrieve positive images under severe visual variations.

exhaustive comparison between the query and dataset vectors. An example solution involves grouping geometrically close dataset vectors into the same bin, and avoid dissimilar bins for a given query. However, these approaches are not as effective in higher dimensional spaces. An example of space partitioning in $\mathbb{R}^2$ is shown in Figure 1.2.

1.1.2 Visual Representation

Many of the traditional systems described an image with a set of local features, such as SIFT [Low04]. Although effective, this was not an efficient solution since the search and memory complexity grew exponentially with the number of images. Other works designed pooling strategies to aggregate a set of local features into a single global descriptor [SZ03, SPMV13, JPD$^+$12]. More recently, CNN-based descriptors achieved state-of-the-art performance in many image retrieval benchmarks [RTC16, GARL16b]. However, objects with severe visual variations, such as small and occluded objects, objects with background clutter, and objects from different viewpoints still remain a significant problem. One approach is called "query expansion". The main idea is to conduct a basic nearest neighbor search with the query image and retrieve the nearest neighbors from dataset. Then, the second part of the process involves searching the dataset using the retrieved nearest neighbors as queries. This helps retrieving images that contain the same object as query, but under severe visual variations. This process is shown in Figure 1.3.

1.2 Solutions and Content

In this thesis, we attempt to solve the challenges mentioned in Section 1.1. Our solutions for both problems involve continuous representations of a set of vectors. Initial chapters investigate the problem of indexing vectors in high-dimensional spaces and propose efficient solutions. This is done by assigning images to groups and creating a single continuous representation for each group. We then switch gears and propose a query expansion framework which successfully retrieves images with different visual variations. We use a weighted graph to represent the relationship between dataset images for this problem.

This transcript is organized as follows:

- Chapter 2 gives brief literature background on the aforementioned challenges. More specifically, we explain existing works in feature description, indexing, and query expansion in image retrieval.
- Chapter 3 introduces an indexing scheme. We present a theoretical analysis of how to represent a set of vectors in high-dimensional spaces. We call our group representations memory vectors and show their effectiveness in various image retrieval benchmarks.
- Chapter 4 presents practical applications and extensions of memory vectors. We first show the effectiveness of using memory vectors for multimedia retrieval domain in terms of efficiency and privacy. We then investigate practical extensions of memory vectors, such as their combination with existing space-partitioning techniques.
- Chapter 5 presents a data-driven grouping strategy for indexing. Unlike Chapter 3, group representation and assignment we propose in this chapter is completely data-driven.
- Chapter 6 changes our focus from efficiency to accuracy, as we explore a query expansion method using the relationship between sets of images in the dataset. We propose to apply an efficient manifold search technique to retrieve positive images that were originally ranked in the bottom of the list.
- Chapter 7 extends our query expansion technique by making it more efficient during the query time. While this increases the offline training time, our method proposed in this chapter is up to 300 faster in query time than the one we proposed in Chapter 6.
- Chapter 8 investigates the grouping of images for location recognition task. We examine multiple aggregation techniques based on the location of images, including memory vectors proposed in Chapter 3. We also compare location aggregation against manifold search proposed in Chapter 6.

Chapter 2

Background

This chapter is designed to give a brief background about instance-based image retrieval. In Section 2.1, we explain the basics of image representation for image retrieval, including feature detection, representation and pooling for traditional hand-crafted descriptors, and also more recent CNN-based learned feature representations. Then, we discuss some of the existing image indexing techniques for image retrieval in Section 2.2, such as space-partitioning and similarity estimation techniques. In Section 2.3, we give a brief introduction to online query expansion.

2.1 Feature Representation and Pooling

This section talks about the existing work involving local feature detection, representation, and pooling. We first investigate the traditional part of the literature involving hand-crafted local features, then move on the more recent research involving CNN-based features. We use the terms pooling and aggregation interchangeably throughout this section. Some parts of this section have been published in [ITGJ15].

2.1.1 Hand-crafted Features

Hand-crafted features refer to those that are detected, represented and pooled based on certain fixed mathematical operations. Extracting such features from an image consists of three steps. The detection step selects regions of interest, which are normalized into fixed-size patches. The description step produces a vector representation for each of the detected patches. While not mandatory, pooling step aggregates the set of vectors extracted from an image into a single vector. We now describe the existing work for each of the three steps.

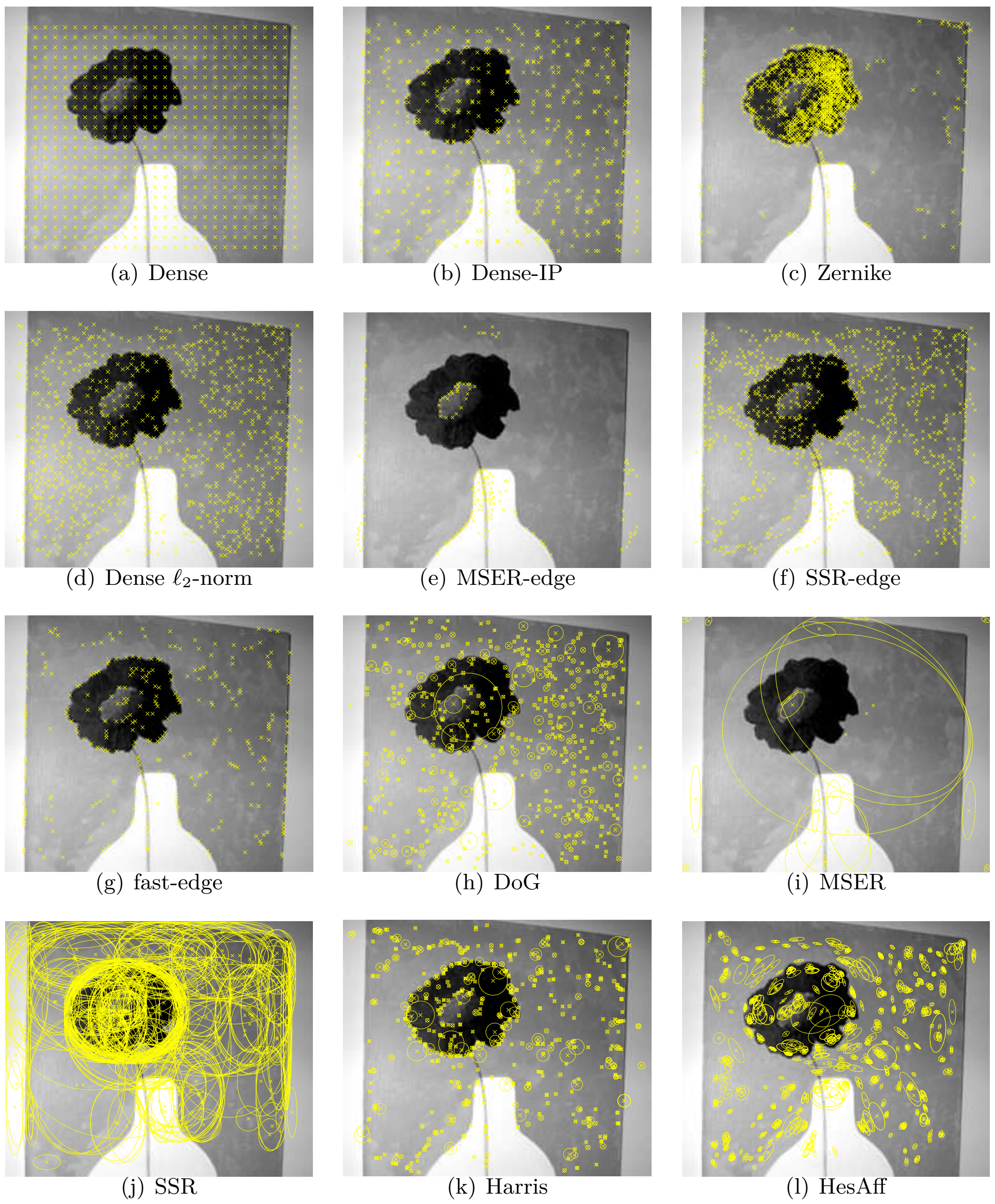


Figure 2.1: Visualization of some of the local feature detection methods as shown in Iscen *et al.* [ITGJ15]. For methods with predefined scales, we visualize the center of points detected on the first scale (a)-(g), while for the ones that perform scale selection we draw the corresponding ellipses or circles (h)-(l).

2.1.1.1 Local Feature Detection

Typically, local interest points are tuned to detect image structures, such as corners and blobs, highly distinctive and repeatable. They are rather designed to be appropriate for image matching applications. The standard evaluation metrics for such detectors [MTS$^+$05], e.g. repeatability, are designed to reflect sufficiency for matching applications. One of the earlier exceptions is the work of Mikolajczyk et al. [MZS03] where they focus on edges to represent objects. However, we are not necessarily interested in the repeatability of patches, but we discuss local features from the perspective of the final application, image retrieval.

A related survey is the work of Nowak et al. [NJT06]. They are not interested in the repeatability of patches, but directly measure classification accuracy. According to them, detectors designed to obtain high repeatability perform the same as randomly selected patches. They observe that performance is, up to some extent, an increasing function of the number of points per image. Similarly, Avrithis and Rapantzikos [AR11] do not restrict comparison to repeatability, but further compared image retrieval performance, while considering the average number of points as a crucial parameter. Finally, Iscen et al. [ITGJ15] compare sparse and dense interest detectors for image retrieval, and propose a hybrid detection method.

We briefly describe existing methods for interest point detection or dense patch sampling. In all those methods, a local descriptor is extracted from each region of interest and an image is represented by a set of such descriptors. Figure 2.1 visualizes some of the feature detectors used in computer vision.

Harris-Laplace detector. Harris detector localizes corners, based on the fact that gradient values will change in multiple directions around a corner. It uses a scale adapted version of the second moment matrix, known as Harris matrix [HS88]:

$$M = \sigma_D^2 \cdot g(\sigma_I) * \begin{bmatrix} L_x^2(x,\sigma_D) & L_xL_y(x,\sigma_D) \\ L_xL_y(x,\sigma_D) & L_y^2(x,\sigma_D) \end{bmatrix}, \tag{2.1}$$

where σ_D is the differentiation scale, σ_I is the integration scale and L_z is the derivative computed in z direction. Differentiation scale σ_D is used to compute the local derivatives with Gaussian kernels, and a Gaussian window with a size σ_I is used to smooth and average the neighborhood around the point. The eigenvalues of this matrix represent the gradient changes in two directions. Consequently, if one eigenvalue is large while the other is small, there exists an edge, whereas if both eigenvalues are large, there exists a corner. The interestingness of a point is captured by the cornerness function, defined as $\textit{cornerness} = \det(A) - \alpha \text{trace}^2(A)$, where α is usually set to 0.05.

Extending Harris detector to be scale invariant, Mikolajczyk et al. [MS04] use Laplacian-of-Gaussian (LoG) response and detect local extrema over multiple scales to perform scale selection [Lin98]. This is the well known Harris-Laplace detector. Only points with cornerness value higher than threshold τ are retained and final point locations are chosen by a local maxima search procedure.

Hessian-Affine detector [MTS+05], similar to Harris detector, detects image locations that have large derivatives in both directions. Point locations are selected as local maxima of the Hessian matrix determinant, which now constitutes the interestingness measure. The Hessian matrix is defined as

$$H = \begin{bmatrix} L_{xx}(x, \sigma_D) & L_{xy}(x, \sigma_D) \\ L_{xy}(x, \sigma_D) & L_{yy}(x, \sigma_D) \end{bmatrix}, \tag{2.2}$$

where L_{zz} is the second order partial derivative. All points with interestingness below threshold τ are discarded. Although Hessian and Harris detectors are quite similar, the detected points may be slightly different. In particular, unlike Harris, Hessian detector tends to select locations with texture variations, in addition to corners.

The affine shape of the point is estimated by the eigenvalues of the second order moment matrix M. An iterative procedure modifies the point's location, scale and shape until the estimated affine transform is able to map the detected region into one that has equal eigenvalues of its second order moment matrix.

MSER Maximally Stable Extremal Regions were introduced by Matas et al. [MCMP02] as a feature detector that tends to localize blobs. An extremal region has all its intensity values greater (or less) than the outer region boundary pixels. Then, a sequence of nested extremal regions is considered. Along this sequence scale change between neighboring regions is estimated. Maximally stable are the local minima of this quantity. In this fashion, nested regions are also likely to appear. The detected regions can have very irregular shapes, therefore an ellipse is fitted to detected regions in order to extract local descriptors. A parameter Δ controls the locality of the scale change computation.

Difference of Gaussians (DoG) was originally used for local feature detection by Lowe [Low04]. DoG is an efficient way to approximate the Laplacian of Gaussian and to detect edges at various image scales. A Gaussian kernel is used to create multiple blurred versions of the image per octave. Simple subtraction of two consecutive blurred images produces the DoG response. Interest points are located in the spatial-space as local-maxima in a 3D search area of size 3.

Regular grid dense sampling. In contrast to interest points, dense sampling methods give less importance to high repeatability and try to provide a dense coverage of the depicted objects. The most popular method is to sample points on a regular grid, every δ_{xy} pixels. Depending on the application δ_{xy} can be really small, such as 3, or quite larger, such as 16. In order to provide some scale tolerance, different scales are considered by following the same procedure at n_σ multiple scales of the image. All the patches from different scales are pooled together in the end. A typical value for n_σ is 5, with 2 scales per octave. A significant improvement is achieved by filtering out descriptors with low ℓ_2-norm [GMJP14].

Dense interest points were introduced by Tuytelaars [Tuy10] as a hybrid solution to trade-off between sparse interest point detection and dense sampling. Instead of selecting the center pixel of grid cell, as in regular dense sampling, they conduct local search inside each cell over spatial and scale space. The point with maximum response is kept per cell. Selected points are not necessarily local maxima. Since a single interest point is selected from each cell, patches are very likely to be localized on smooth regions. Moreover, quite a few patches are localized on the cell borders.

Zernike polynomials was introduced by Iscen et al. [ITGJ15] as another hybrid solution between sparse and dense interest point detection. The authors propose to use a bank of Zernike polynomials [Zer34] as detectors. Each pseudo-Zernike function is used as a filter and this response constitutes the interestingness measure for detection of point locations. Local maxima and local minima are detected for each filter independently. Filter responses are ranked per filter, and point locations are selected until each filter capacity is filled or until there are no more local extrema left. It is shown that this method provides dense, yet localized image patches, and outperforms other detection methods in image retrieval task.

2.1.1.2 Local Feature Representation

The SIFT descriptor [Low04] and its RootSIFT extension [AZ12b] have been shown to perform very well for most applications. Many descriptors have been introduced in the last years to improve the description speed or descriptor compactness, such as SURF [BETG08], CHOG [CTC$^+$09] or BRIEF [CLSF10]. The matching accuracy is improved by learning the descriptor design [WB07, WHB09, SVZ14]. We will now briefly describe SIFT and its extension RootSIFT.

SIFT, or Scale Invariant Feature Transform reduces each detected image patch into a vector representation. The original SIFT detection algorithm [Low04] employs DoG for interest point detection, and further applies additional steps to filter out points belonging to edges or low-contrast regions. Nevertheless, SIFT can be used with any of the feature detection techniques described in Section 2.1.1.1.

Given a detected image patch, SIFT first assigns an orientation to it. The goal of this operation is to represent the detected image in a rotation-invariant manner. We first calculate the gradient orientation of evenly sampled points from the detected region. These values are quantized into a 36-bin histogram, evenly dividing the range of 360°orientation angles. Contribution of each sampled point to the orientation histogram is weighted by its gradient magnitude. The orientation corresponding to the highest peak of the orientation histogram is considered as the dominant orientation and is assigned to the detected keypoint. In the case of multiple dominant orientations, local peaks among the 80% of the highest peaks of the orientation histogram are assigned. Multiple keypoints, one for each orientation, are created in that case.

The next step of the algorithm is to create a discriminative but illumination-invariant feature descriptor for each detected region. A square region of 16×16 pixels is taken around the detected keypoint. This is further divided into 16 subregions of

4×4 pixels. Each sub-region contains an orientation histogram of 8 bins. After computing 8-bin orientation histograms of each sub-region, the local feature representation is obtained by concatenating them together, resulting in a $16 \times 8 = 128$ dimensional feature descriptor. In practice, this vector is usually ℓ_2-normalized for better matching accuracy.

Traditionally, Euclidean distance was used to compare the similarity of two SIFT features. However, since they are histograms, it is also possible to use other appropriate distances or metrics. Arandjelović and Zisserman [AZ12b] show that with just a couple of basic operations, it is possible to compute the Hellinger distance between two sets of SIFT vectors. Given a SIFT vector, they (i) ℓ_1-normalize it (ii) take the square root of each element. They call the resulting descriptor RootSIFT. It is easily shown that the Euclidean distance between two RootSIFT features is equivalent to the Hellinger distance between two SIFT features. Assume that $\mathbf{x}$ and $\mathbf{y}$ are two different sets of ℓ_2-normalized SIFT descriptors. Their Euclidean distance is computed as

$$d(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||^2 = ||\mathbf{x}||_2^2 + ||\mathbf{y}||_2^2 - 2\mathbf{x}^\top \mathbf{y} = 2 - 2\mathbf{x}^\top \mathbf{y}. \tag{2.3}$$

The Hellinger kernel for two ℓ_1-normalized histograms $\mathbf{x}$ and $\mathbf{y}$ is defined as

$$H(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \sqrt{x_i y_i}, \tag{2.4}$$

where n is the length of each histogram.

It then follows that the similarity between two RootSIFT features is $\sqrt{\mathbf{x}}^\top \sqrt{\mathbf{y}} = H(\mathbf{x}, \mathbf{y})$. Hence, the Euclidean distance between two RootSIFT features is equivalent to the Helinger distance between two SIFT features:

$$d(\sqrt{\mathbf{x}}, \sqrt{\mathbf{y}}) = ||\sqrt{\mathbf{x}} - \sqrt{\mathbf{y}}||^2 = ||\mathbf{x}||_2^2 + ||\mathbf{y}||_2^2 - 2\sqrt{\mathbf{x}}^\top \sqrt{\mathbf{y}} = 2 - 2H(\mathbf{x}, \mathbf{y}). \tag{2.5}$$

2.1.1.3 Local Feature Pooling

Depending on the feature detection technique, each image may contain a set of hundreds to thousands of SIFT vectors. When comparing the similarity of two images, one possibility is to cross-match all SIFT vectors from both images. This involves computing the cross-distance between thousands of vectors. The complexity grows as the square of the number of vectors, making it inefficient for a practical image retrieval scenario. One solution is to approximate cross-matching by aggregating a set of SIFT vectors into a single global descriptor of higher dimensionality. In this section, we describe some of the popular aggregation techniques in the literature.

Bag-of-Words models were originally developed for text understanding [SM83]. Given a document, the idea is to create an orderless representation in order to compare it against other documents. This is done with the help of a dictionary, which is assumed to contain all the possible words that may appear in a document. For each document, we keep a histogram whose length is the size of the dictionary. We then simply count the frequencies of each word in the document and assign it its corresponding bin in the histogram. As a result, the similarity comparison between two documents is reduced to a histogram comparison.

A similar model is applied in computer vision and image retrieval [SZ03, CDF$^+$04]. In that context, images are considered as documents, and local SIFT features are the so-called "visual words". One of the main differences is that, even though it is possible to store a fixed dictionary which may correspond to all the words in a language for a text application, this is not possible for computer vision. One cannot simply keep a list of all possible SIFT vectors. In order to overcome this issue, the visual dictionary is created with k-means clustering. This is part of the training process, where all SIFT features from all training images are extracted and gathered together. Then, we cluster these features into k clusters using the k-means algorithm [Llo82]. Resulting cluster centers are considered as the entries of the visual dictionary, and are called visual words. Given an image, each SIFT feature is assigned to the closest cluster center, and the frequency of each visual word is recorded in a k-dimensional histogram, which is the BoW representation of the image. Additionally, ℓ_2-normalization and idf computation are applied for better performance [SZ03].

Fisher vectors proposed by Perronnin et al. [PD07], is a more sophisticated aggregation approach. The goal is similar to BoW: aggregation of many local SIFT features into a single global descriptor. However, Fisher vectors usually require fewer visual words (k) due to its complexity.

Fisher vectors is an approximation of the more general Fisher Kernel used for classification [JH98]. Contrary to the BoW-models, Fisher vectors models the visual word space using a Gaussian mixture model (GMM), and not the k-means algorithm. GMM parametrizes the SIFT feature space using the mean μ_i, covariance Σ_i and priors w_i for each visual word $i = 1, ..., k$. After the training of GMM, each image is encoded into a Fisher vector using the following procedure.

Assume that $\mathbf{x}$ is the set of SIFT descriptors for an image. We first compute the posterior probability between ith visual word and the jth SIFT feature

$$\gamma_{ij} = \frac{\exp\{-\frac{1}{2}(\mathbf{x}_j - \boldsymbol{\mu}_i)^\top \Sigma_i^{-1}(\mathbf{x}_j - \boldsymbol{\mu}_i)\}}{\sum\limits_{t=1}^{k} \exp\{-\frac{1}{2}(\mathbf{x}_j - \boldsymbol{\mu}_t)^\top \Sigma_i^{-1}(\mathbf{x}_j - \boldsymbol{\mu}_t)\}}. \tag{2.6}$$

Compared to the basic BoW model which follows hard-assignment to cluster centers, (2.6) acts as a soft assignment of $\mathbf{x}_j$ to each visual word. Then, we compute the mean and covariance deviation vectors with the following:

$$\mathbf{u}_i = \frac{1}{n\sqrt{w_i}} \sum_{j=1}^{n} \gamma_{ij}(\mathbf{x}_j - \boldsymbol{\mu}_i) \odot \frac{1}{\boldsymbol{\sigma}_i}, \tag{2.7}$$

$$\mathbf{v}_i = \frac{1}{n\sqrt{2w_i}} \sum_{j=1}^{n} \gamma_{ij} \Big\{ \Big((\mathbf{x}_j - \boldsymbol{\mu}_i) \odot \frac{1}{\boldsymbol{\sigma}_i} \Big)^{\circ 2} - 1 \Big\}, \tag{2.8}$$

where n is the number local SIFT features in the image. Note that, since we are considering local SIFT features, all $\mathbf{x}_j$, $\mathbf{u}_i$, and $\mathbf{v}_i$ have $d = 128$ dimensions. The final representation for Fisher vectors is produced by concatenating (2.7) and (2.8) for each component. In other words:

$$\mathbf{f} = [\ldots \mathbf{u}_i \ldots \mathbf{v}_i \ldots]^\top, \tag{2.9}$$

for $i = 1, ..., k$ components. Therefore, the final representational dimensionality of Fisher Vectors is $2dk$. Even though, the final dimensionality of $\mathbf{f}$ is much higher than BoW for a given k, the complexity behind this method makes it possible to use it with small k, giving relatively less overall dimensionality. In practice, ℓ_2-normalization is also applied to $\mathbf{f}$.

VLAD or Vector of Locally Aggregated Descriptors [JDSP10], is another aggregation method of local SIFT features. VLAD is based on the same idea as Fisher Vectors, as it keeps statistics about relationship between visual words and local SIFT features in addition to computing frequencies of each visual word. Contrary to the Fisher Vectors, VLAD follows a simpler aggregation technique which does not require storing second-order statistics. It can be considered as a more efficient variant of Fisher Vectors.

We first learn a dictionary of visual words using the k-means algorithm. Dictionary entries (visual words) are cluster centers. For each image, we first find the closest visual word for local SIFT features. Instead of just storing the frequency, we accumulate residual between each visual word and the local SIFT features assigned to it:

$$\mathbf{r}_i = \sum_{j \text{ s.t. } \mathrm{NN}(\mathbf{x}_j)=\mathbf{c}_i} \mathbf{x}_j - \mathbf{c}_i, \tag{2.10}$$

where $\mathbf{c}_i$ is the ith visual word. The final global representation is created by concatenating $\mathbf{r}_i$

$$\mathbf{f} = [\mathbf{r}_1^\top \ldots \mathbf{r}_i^\top \ldots \mathbf{r}_k^\top]^\top. \tag{2.11}$$

Similarly to BoW and Fisher Vectors, ℓ_2-normalization is applied in the end. The overall dimensionality of $\mathbf{f}$ is dk, which is twice as less compared to Fisher Vectors.

GMP , or General Max Pooling [MP14], is a pooling technique designed to solve the burstiness problem. In some cases, images have frequently-occurring visual patterns that are not discriminative. These may be patches extracted from background, grass, sky or other uninformative regions. If we think of aforementioned pooling techniques (BoW, Fisher Vectors, VLAD) as some kind of sum-pooling, local SIFT features corresponding to such frequently-occurring patterns dominate the final representation due to their frequency. Max-pooling can simply be applied instead of sum-pooling [BPL10], but this not compatible with higher-order statistical representations such as Fisher vectors, because it treats each dimension independently. Several other heuristic approaches attempted to solve the burstiness problem through different normalization schemes [JDS09, TSPO13a, AZ13, DGJP13], such as power normalization [PSL10, PJM10, JPD+12]. Jégou and Zisserman [JZ14] also propose a similar aggregation method to GMP, where the weights are solved in a different manner.

Murray and Perronnin propose GMP [MP14], which attempts to mimic max-pooling while being compatible with global representations such as Fisher Vectors. Their goal is to create a pooling operator, which preserves the similarity between rarely-occuring features. Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ be a $d \times n$ matrix containing n local features. GMP wants to preserve the matching similarity between the final representation $\mathbf{f}$ and $\mathbf{x}_j$, s.t. $\mathbf{x}_j^\top \mathbf{f} = 1$ for all j. In matrix form, this is written as:

$$\mathbf{X}^\top \mathbf{f} = \mathbf{1}_n. \tag{2.12}$$

The optimal solution with the minimal ℓ_2-norm is found after solving the least-squares regression problem

$$\mathbf{f}^\star = \arg\min_{\mathbf{f}} ||\mathbf{X}^\top \mathbf{f} - \mathbf{1}_n||^2. \tag{2.13}$$

There also exists a closed-form solution, which is solved with Moore-Penrose pseudo-inverse [RM72] denoted by $^+$:

$$\mathbf{f}^\star = (\mathbf{X}^\top)^+ \mathbf{1}_n = (\mathbf{X}\mathbf{X}^\top)^+ \mathbf{X}\mathbf{1}_n. \tag{2.14}$$

The authors also propose a regularized version of this solution with parameter λ . This is also known as ridge regression or Tikhonov regularization in the literature:

$$\mathbf{f}^\star_\lambda = (\mathbf{X}\mathbf{X}^\top + \lambda I)^+ \mathbf{X}\mathbf{1}_n, \tag{2.15}$$

where I is the identity matrix of size n. Note that we have max-pooling when $\lambda = 0$ and sum pooling as $\lambda \to \infty$.

2.1.2 CNN-based Features

CNN-based global descriptors are becoming popular in image retrieval, especially for instance-level search. Existing works [BL15, RSCM16, KMO15, TSJ16] employ "off-the-shelf" networks, originally trained on ImageNet, to extract descriptors via various pooling strategies. The invariance is partially designed by global max [ARS+14, TSJ16] or sum [KMO15, BL15] pooling layers or multi-scale querying [GARL16b], and partially learned by the choice of the training data. Robustness to background clutter is improved by computing descriptors over object proposals [MB15, GARL16a, XHZT15] or over a fixed grid of regions [TSJ16]. Better performance is observed at a cost of increased memory footprint [RSCM16].

Other approaches [BSCL14, RTC16, GARL16a] fine-tune such networks to obtain descriptor representations specifically adapted for instance search. NetVLAD [AGT+16] trains a VLAD layer on top of convolutional layers in an end-to-end manner. It is tuned for the location recognition task. The training images are obtained from panoramas, fed to a triplet loss to make it more compatible

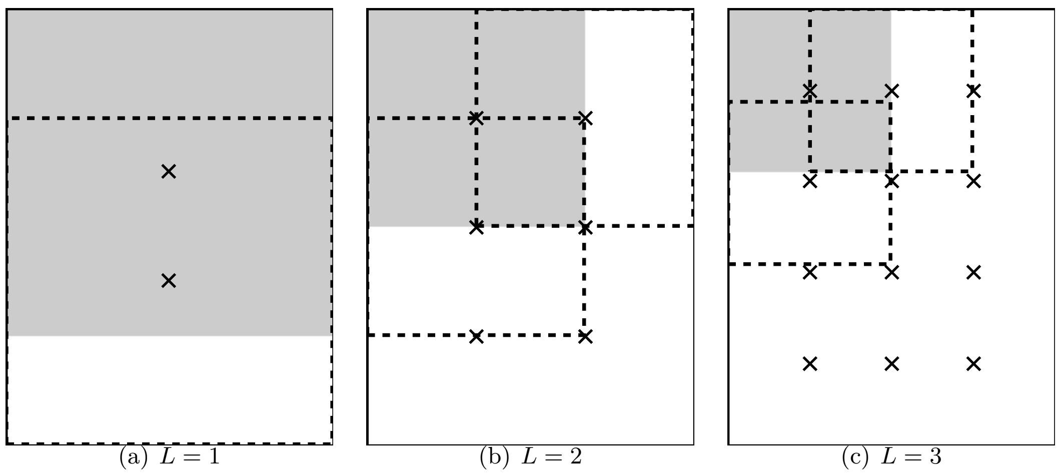


Figure 2.2: Visualization of sample square regions for R-MAC in the first 3 scales. The top-left region of each scale is shown in grey, and its neighbors are shown with dashed lines. Figure is taken from [TSJ16] with the permission of the corresponding author.

with image retrieval. As a result, their representation outperforms existing works in standard location recognition benchmarks.

In this section, we describe the MAC and R-MAC pooling methods [TSJ16] which are also used in the state-of-the-art image retrieval fine-tuned networks [RTC16, GARL16a]. MAC (Maximum activation of convolutions) acts on the output of an existing fully-convolutional network, such as VGG [SZ14] or ResNet [XHZT15], to create a representation for the entire image. We discard the fully connected layers and rather operate on the 3D output tensor of the convolutional layers. Assume that the output tensor $\mathcal{X}$ has k feature maps. Each feature map $\mathcal{X}_i$, s.t. $i = 1, \dots, k$, contains a set of $W \times H$ activations obtained from the network. MAC image representation is then created by max-pooling over all dimensions per feature map:

$$\mathbf{f} = [\mathrm{f}_1 \dots \mathrm{f}_i \dots \mathrm{f}_k]^\top, \text{s.t. } \mathrm{f}_i = \max_{\mathbf{x} \in \mathcal{X}_i} x \cdot \mathbb{1}(x > 0), \tag{2.16}$$

where $\mathbb{1}$ is the indicator function which only fires for non-negative values. The dimensionality of the representation is equal to the number of feature maps.

R-MAC (Regional Maximum Activation of Convolutions) is a variant of MAC, which also considers the location of the activations in the representation. Given $W \times H \times k$ dimensional activation map $\mathcal{X}$, idea is to uniformly sample square regions from the $W \times H$ grid. The sampling is done on L different scales. At the first scale $l = 1$, the size of the square is as large as possible, s.t. the height and the weight is $\min(W, H)$. Regions are then sampled such that the overlap between them is 40%. At each scale l, regions of width $2\min(W, H)/(l + 1)$ are sampled. Finally, all region vectors are sum-aggregated , ℓ_2-normalized, applied PCA-whitening [JC12], and ℓ_2-normalized again. This gives a $d = k$ dimensional final global representation of the image.

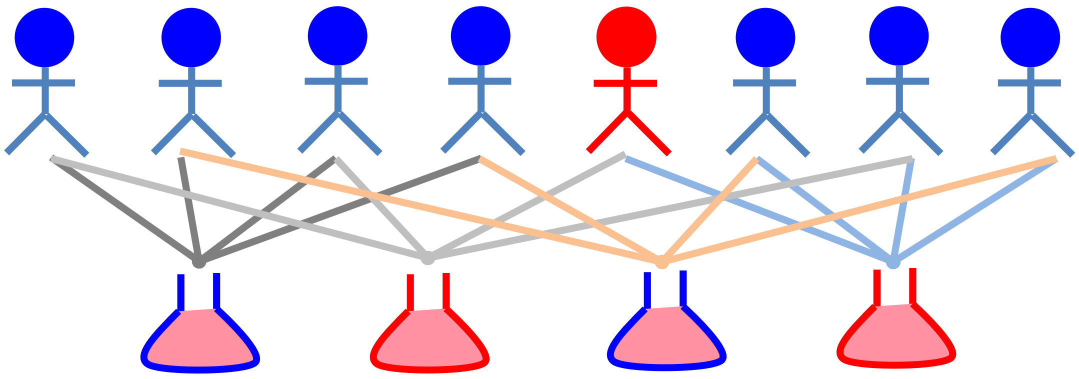


Figure 2.3: Group testing was first introduced in World War II by the American army. Due to many injuries American army required all of its soldiers to make blood donations. However, a small fraction of soldiers were infected with syphilis, and it was too expensive to exhaustively search everyone one by one. So they decided to pool blood samples into mixtures, and if a mixture was found positive, then the blood samples were individually tested. Figure taken from Shi *et al.* [SFJ14].

2.2 Image Indexing and Similarity Estimation

Image retrieval aims to find the images in a large scale dataset that are most similar to a given query image. Recent approaches create a global descriptor vector for each image. Visual similarity is then quantified by measuring the similarity of these vectors (e.g., cosine similarity). If the dataset has N images each represented by a d-dimensional feature vector, then an exhaustive search for each query requires dN operations.

A common approach to accelerate image search is space-partitioning indexing, which operates in sub-linear time [ML14]. Indexing partitions the feature space $\mathbb{R}^d$ into clusters and computes similarities between the query and dataset vectors that fall in the same or neighboring clusters. Yet, as the dimension d grows, the chance that similar images are assigned to different clusters increases, and the efficiency of these methods collapses due to curse of dimensionality [WSB98, ML14]. This is problematic in computer vision since most state-of-the-art image descriptors have high intrinsic dimensionality.

Another popular approach for efficient image search performs a linear scan over the dataset, computing approximate similarities using compact codes [Cha02, AZ14a, WTF09, DCL08, JDS11, BL12]. These techniques have a complexity of $d'N$ where $d' < d$ is the reduced dimensionality of the compact code. The similarity between vectors in $\mathbb{R}^d$ is approximated by the distance between their compact codes. State-of-the-art large scale search algorithms combine indexing strategies with approximated similarities [JDS11]. In other words, these works trade complexity for memory.

In this section, we briefly describe some of the existing indexing methods in image retrieval.

2.2.1 Space Partitioning

Similarity search for image retrieval often boils down to running a k-nearest neighbor process. Assume that the dataset $\mathbf{X}$ is composed of N d-dimensional vectors $\{\mathbf{x}_i\}_{i=1}^N$ such that $\|\mathbf{x}_i\| = 1, \forall 1 \leq i \leq N$. The similarity between the query $\mathbf{q}$ and the vector $\mathbf{x}_i$ can be measured with the scalar product $\mathbf{q}^\top \mathbf{x}_i$ when all vectors are unit normalized.

FLANN (Fast Library for Approximate Nearest Neighbors) is a well-known approximate similarity search framework and an example of space partitioning. It quantizes the feature space based on the geometrical positions of the data vectors. The authors propose two methods for quantization: the randomized k-d tree algorithm and the priority search k-means tree algorithm.

In the classic k-d tree algorithm [FBF77], the split dimension is chosen as the dimension with the highest variance. FLANN's randomized k-d tree algorithm creates the tree similarly, with the split dimension chosen randomly from the top D dimensions with the highest variance. This allows for the creation of multiple k-d trees, which helps the performance.

The second alternative of the framework, the priority search k-means tree algorithm, quantizes the data using the distance across all dimensions instead of single dimension at a time. This is done by clustering the data points in K clusters in each level hierarchically, until the number of points in a cluster is less than K.

Even though it is a state-of-art library, FLANN does not perform well for truly high-dimensional data, as acknowledged by its authors [ML14]. Effectiveness of FLANN decreases when indexing such high-dimensional descriptors. It is only possible to return high-quality results at the expense of a dramatic increase in the response time. This is caused by FLANN performing a very large number of checks to eventually gather enough good neighbors. This problem is not specific to FLANN but caused by the curse of dimensionality whose effects are mitigated with extensive multiprobing.

2.2.2 Distance Approximation

LSH [Cha02], or Locality-sensitive hashing, is a well-known distance approximation technique. It is used to create compact binary embedding from continuous feature descriptors. Fast search often uses binary embedding for two reasons. It offers a compact representation of the vectors and SSE instructions computes Hamming distances much faster then Euclidean distance or cosine similarity.

An embedding casts a vector $\mathbf{x} \in \mathbb{R}^d$ into a vector of D discrete components.

$$\begin{aligned} e : \mathbb{R}^d &\rightarrow \mathcal{A}^D \\ \mathbf{x} &\rightarrow e(\mathbf{x}) := (e(\mathbf{x})_1, \cdots, e(\mathbf{x})_D) \end{aligned} \tag{2.17}$$

This function is designed to provide i) a compact representation of vectors, and ii) an estimation of the similarity $s(\mathbf{q}, \mathbf{x})$ between two vectors from their embeddings: $s_e(e(\mathbf{q}), e(\mathbf{x}))$. Function $s_e(\cdot, \cdot)$ is faster to compute than $s(\cdot, \cdot)$.

LSH [Cha02] uses binary embeddings where $\mathcal{A} = \{0, 1\}$. In a nutshell, once D directions $\{\mathbf{u}_k\}_{k=1}^D$ have been randomly drawn i.i.d. uniformly in hypersphere $\mathbb{S}^d$, the

k-th symbol of the embedding is computed as follows:

$$e(\mathbf{x})_k := \mathsf{sign}(\mathbf{x}^\top \mathbf{u}_k), \forall k \in [D], \tag{2.18}$$

with $\mathsf{sign}(x) := 1$ if $x \geq 0$, and 0 otherwise. LSH has the following well-known property:

$$\frac{\mathbf{x}^\top \mathbf{q}}{\|\mathbf{x}\| \|\mathbf{q}\|} \approx \cos\left(\frac{\pi}{D} d_H(e(\mathbf{q}), e(\mathbf{x}))\right). \tag{2.19}$$

The cosine is estimated via the Hamming distance $d_H(\cdot, \cdot)$.

Many extensions of LSH exist, including spectral-hashing [WTF09] or shift-invariant kernels [RL10]. More recent approach, Product Quantization [JDS11], uses a non-binary finite alphabet to quantize each vector using its subspace indices.

2.2.3 Group Testing

Group testing has been used in other computer science fields in the past to solve typical 'needles in a haystack' problems. It did not receive attention in the image retrieval community until the pioneering work of Shi et al. [SFJ14]. Here the goal is to reduce the number of vectors against which the query is compared. The full dataset of N vectors is first summarized by $M \ll N$ group vectors, where each group vector is also d-dimensional. As the name suggests, each group vector represents a small subset of images in the original dataset. These groups are composed randomly with or without overlapping. Computation of the group vectors is performed offline under a specific construction such that a comparison group vector *vs* query vector measures how likely the group contains query matching vectors. Then, when presented with a query, the system compares the query with the group vectors instead of individual image vectors. This reduces the complexity from dN to dM.

Initial attempts [SFJ14] considered an adaptive group testing approach. M groups are composed from the dataset, and querying proceeds in two stages. In the first stage, the scores between group vectors and the query are computed. They measure how likely their group contains some matching images. Then, in the second stage, the query is compared with individual image vectors for only the most likely positive groups. If the groups are roughly balanced in size and the query only matches a small number of group vectors, then the complexity is reduced from dN to $d(M + N/M)$. Although this results in efficient image retrieval, it has one major drawback: memory usage is increased since the group vectors and mapping from images to groups are stored in addition to the dataset feature vectors.

Shi et al. [SFJ14] randomly assigns each feature to m multiple groups, and each group consists of n vectors. This gives M groups in total, such that

$$M = \frac{mN}{n}. \tag{2.20}$$

The representative vector for each group is simply the sum of features belonging to that group. More precisely, let $[\mathbf{x}_i]_{\mathcal{M}_j}$ be the $d \times n$ matrix storing the features of jth group, then the representative vector $\{\mathbf{y}_j\}_{j=1}^{M}$ is computed as:

$$\mathbf{y}_j = \sum_{\mathbf{x}_i \in \mathcal{M}_j} \mathbf{x}_i. \tag{2.21}$$

At search time, the query is compared to all the representative vectors. Assume that $\mathbf{Y} \in \mathbb{R}^{d \times M}$ is the matrix containing the representative vectors in each column. Then, M group similarities for a given query $\mathbf{q}$ are calculated as:

$$\mathbf{v} = [v_1, ..., v_M]^\top = \mathbf{Y}^\top \mathbf{q}. \tag{2.22}$$

The group similarity v_j is seen as a test output roughly indicating whether or not the query is matching one of the features of the j-th group. This step has a complexity in $O(Md)$.

The score u_i reveals how likely feature i matches the query. It is simply the sum of the group similarities it belongs to:

$$u_i = \sum_{j \in \mathcal{L}_i} v_j, \tag{2.23}$$

where $\mathcal{L}_i$ is the set of groups where the i-th feature is stored. This step has a complexity in $O(NM)$.

Finally, a shortlist of size $R << N$ is created based on the top scores of $\mathbf{u}$, and a verification step reranks these candidate features based on their true similarities with the query. This final step has a complexity in $O(Rd)$.

2.3 Query Expansion

Online query expansion techniques are used to re-rank a ranked list of images after the initial search. A variety of methods [CPS+07, CMPM11, TJ14] employ local features and are well adapted to the Bag-of-Words model [SZ03]. Others are generic and applicable on any global image representation [JHS07, QGB+11, AZ12b, SLBW14, DJAH14]. In both cases, ranking is performed on the image level.

2.3.1 Average Query Expansion

In average query expansion [CPS+07], Chum et al. construct a new query from the top ranked images returned from the original query. Assume that $\mathbf{q}$ is the original query vector. We search the dataset of image using $\mathbf{q}$, i.e. similarity search, and retrieve top m ranked images. Then a new query is formed based on the average:

$$\mathbf{q}_{\text{avg}} = \frac{1}{m+1}\left(\mathbf{q} + \sum_{i=1}^{m} \mathbf{x}_i\right), \tag{2.24}$$

where $\mathbf{x}_i$ is the ith returned dataset image. This simple process is shown to improve the accuracy of the image search for recent CNN-based representations as well [RTC16, GARL16a].

A similar approach called database-side feature augmentation [TL09, AZ12b], applies a similar process offline to the dataset image representations. Nearest m neighbors for each dataset image is found, and each image represented by the sum-aggregation of its own representation and its neighbors' representations. Furthermore, weighting can be applied during the sum-aggregation, where $\text{weight}(r) = \frac{m-r}{m}$ is the weight for the rth ranked neighbor.

2.3.2 Manifold Search

In this section, we focus on a different query expansion method based on manifold search, which propagates similarities through a pairwise affinity matrix [DB13, PBMW99]. Manifold-based techniques are applied to many computer vision problems, such as semi-supervised classification [ZBL+03], seeded image segmentation [Gra06], saliency detection [LMV14, CZHZ16], clustering [Don13] and image retrieval [JB08, EKG10, YKTL09, DB13, XTZZ14]. It is also used for the retrieval of general scenes or shapes of particular objects [JB08, EKG10, YKTL09, DB13]. Additionally, it can also fuse multiple feature modalities [ZYC+12, YMD15] by jointly modeling them on the same graph.

The power of such methods lies in capturing the intrinsic manifold structure of the data [ZBL+03]. The popular PageRank algorithm [PBMW99] was originally used to estimate the importance of web pages by exploiting their links in a graph structure. Our retrieval scenario comes closer to its so called personalized [PBMW99] or query dependent versions [RD01], where the final ranking both respects the data manifold and the similarity to a number of query vectors. Donoser and Bischof [DB13] review a number of diffusion mechanisms for retrieval. Note that the term diffusion which they use is only weakly related to continuous time diffusion process or random walks on graph. We mainly follow Zhou et al. [ZWG+03] below.

2.3.2.1 Affinity matrix

Given a dataset $\mathcal{X} := \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^d$, we define the affinity matrix $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ having as elements the pairwise similarities between points of $\mathcal{X}$:

$$a_{ij} := s(\mathbf{x}_i, \mathbf{x}_j), \quad \forall (i,j) \in [n]^2, \tag{2.25}$$

where $[n] := \{1, \dots, n\}$ and $s : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a similarity measure assumed to be symmetric ($A = A^\top$), positive ($A > 0$), and with zero self-similarities ($\text{diag}(A) = \mathbf{0}$).

Matrix A is the adjacency matrix of a weighted undirected graph G with vertices $\mathcal{X}$. The degree matrix of the graph is $D := \text{diag}(A\mathbf{1}_n)$, i.e. a diagonal matrix with the row-wise sum of A. The Laplacian of the graph is defined as $L := D - A$. It is usual to symmetrically normalize these matrices, for instance,

$$S := D^{-1/2} A D^{-1/2}, \tag{2.26}$$

for the affinity matrix and $\mathcal{L} := I_n - S$ for the Laplacian, where I_n denotes the identity matrix of size n. Matrices $L, \mathcal{L}$ are positive-semidefinite [Chu97].

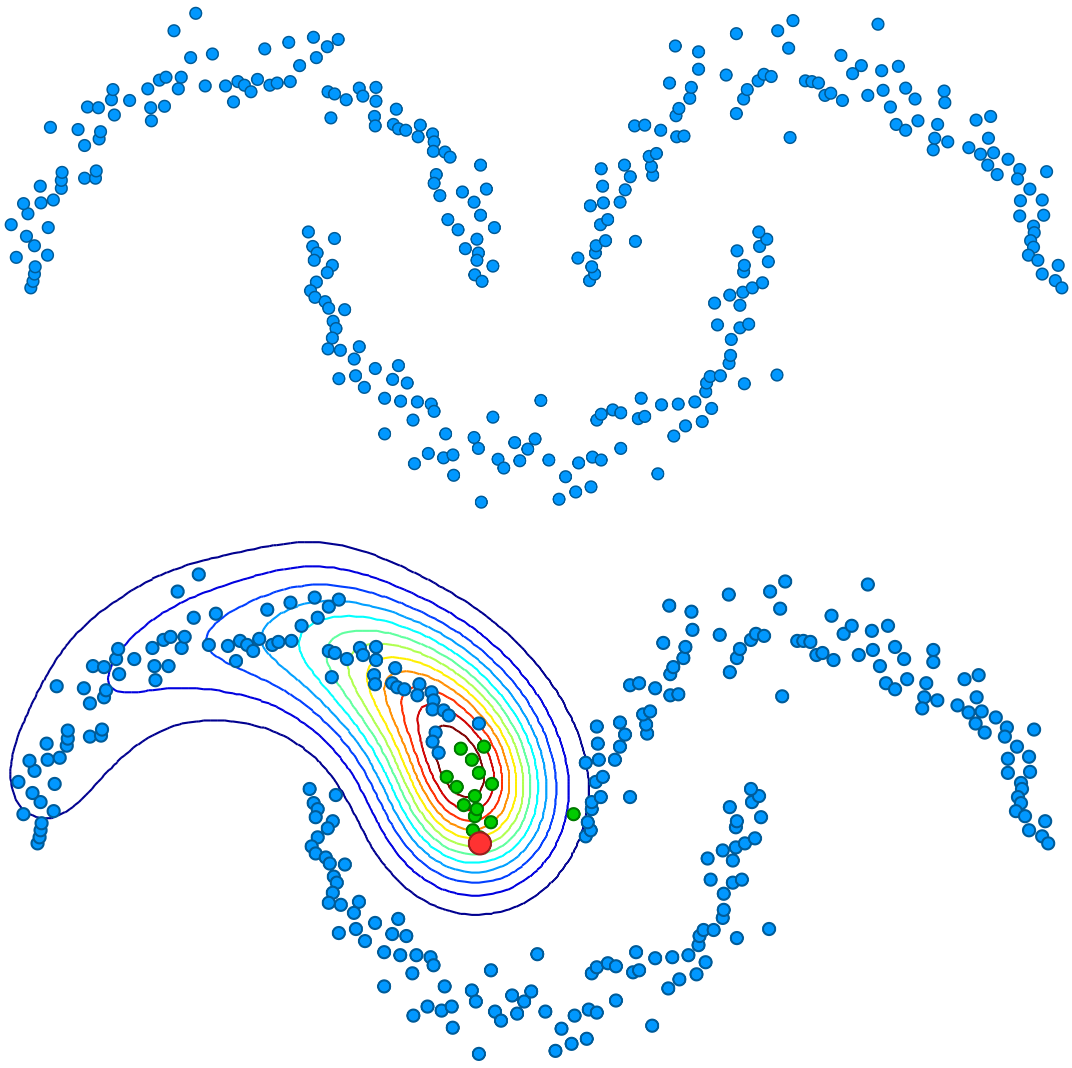


Figure 2.4: An example visualization of searching in manifolds. The dataset is searched with the original query (red), and its nearest neighbors (green) are returned. Manifold search then expands the ranked list so that the other dataset instances that lie on the same manifold are also returned.

2.3.2.2 Searching on manifolds

In the work of Zhou et al. [ZWG+03], a vector $\mathbf{y} = (y_i) \in \mathbb{R}^n$ specifies a set of query points in $\mathcal{X}$, with $y_i = 1$ if $\mathbf{x}_i$ is a query and $y_i = 0$ otherwise. The objective is to obtain a ranking score u_i for each point $\mathbf{x}_i \in \mathcal{X}$, represented as vector $\mathbf{u} = (u_i) \in \mathbb{R}^n$.

We focus on a particular diffusion mechanism that, given an initial vector $\mathbf{u}^0$, iterates according to

$$\mathbf{u}^t = \alpha S \mathbf{u}^{t-1} + (1-\alpha)\mathbf{y}. \tag{2.27}$$

If S is a transition matrix and $\mathbf{y}$ an ℓ^1 unit vector, this defines the following 'random walk' on the graph: with probability α one jumps to an adjacent vertex according to distribution stated in S, and with $1-\alpha$ to a query point uniformly at random. In this fashion, points spread their ranking score to their neighbors in the graph. The benefit is the ability to capture the intrinsic manifold structure represented by the affinity matrix and to combine multiple query points.

Assuming $0 < \alpha < 1$, Zhou et al. [ZBL+03, ZWG+03] show that sequence $\{\mathbf{u}^t\}$ defined by (2.27) converges to

$$\mathbf{u}^\star = (1-\alpha)\mathcal{L}_\alpha^{-1}\mathbf{y} \tag{2.28}$$

where $\mathcal{L}_\alpha := I_n - \alpha S$ is positive-definite, since $\mathcal{L}_\alpha = \alpha\mathcal{L} + (1-\alpha)I_n \succ \alpha\mathcal{L} \succeq 0$.

2.3.2.3 Relation to other domains

A diffusion mechanism also appears in seeded image segmentation [Gra06], where query points correspond to labeled pixels (seeds) and database points to the remaining unlabeled pixels. This problem is equivalent to semi-supervised classification [ZBL+03]. In our context, the approach of Grady [Gra06] decomposes $\mathbf{u} = (\mathbf{u}_d^\top, \mathbf{u}_q^\top)^\top$ for the scores of the query (fixed $\mathbf{u}_q$) and database (unknown $\mathbf{u}_d$) points. Diffusion interpolates $\mathbf{u}_d$ from $\mathbf{u}_q$ by minimizing, w.r.t. $\mathbf{u}_d$, the quadratic cost $\sum_{i,j} a_{ij}(u_i - u_j)^2 = \mathbf{u}^\top L \mathbf{u}$ to enforce that neighboring points should have similar scores. By decomposing $L = [L_d,\ -S_{qd};\ -S_{qd}^\top,\ L_q]$, it is shown [Gra06] that the solution fulfills $L_d\mathbf{u}_d = \mathbf{y}$ with $\mathbf{y} = S_{qd}^\top \mathbf{u}_q$. In our setup, L_d would be singular, preventing us to single out a solution $\mathbf{u}_d^\star$. Yet, it is easy to show that the minimizer of the cost $\alpha\mathbf{u}^\top L\mathbf{u} + (1-\alpha)\|\mathbf{u}\|^2$ has a similar expression to (2.28). The regularization term singles out a solution by forcing $\mathbf{u}$ to be zero in subgraphs not connected to any query point.

2.3.2.4 Local constraints

To handle noise and outliers, a locally constrained random walk [KDB09] is adopted, where only pairs of points that are reciprocal (mutual) nearest neighbors are kept as edges in the graph. In particular, given $\mathbf{z} \in \mathbb{R}^d$, let

$$s_k(\mathbf{x}|\mathbf{z}) = \begin{cases} s(\mathbf{x},\mathbf{z}), & \text{if } \mathbf{x} \in \mathrm{NN}_k(\mathbf{z}) \\ 0, & \text{otherwise} \end{cases} \tag{2.29}$$

be the similarity of $\mathbf{x} \in \mathcal{X}$ given $\mathbf{z}$, that is, restricted to the k nearest neighbors $\mathrm{NN}_k(\mathbf{z})$ of $\mathbf{z}$ in $\mathcal{X}$. Then,

$$s_k(\mathbf{x}, \mathbf{z}) = \min\{s_k(\mathbf{x}|\mathbf{z}), s_k(\mathbf{z}|\mathbf{x})\} \tag{2.30}$$

equals $s(\mathbf{x}, \mathbf{z})$ if $\mathbf{x}, \mathbf{z}$ are the k-nearest neighbors of each other in $\mathcal{X}$, and zero otherwise. Such similarity function is used to construct affinity matrix A like in (2.25).

2.4 Conclusion

This chapter provides an overview on certain aspects of image retrieval which will be discussed in this manuscript. In Section 2.1, we gave a brief overview of image description for instance-based image retrieval. The goal of this section is to provide a basic summary of some of the feature descriptors that will be used in the remainder of this manuscript. The first half of this manuscript is concerned about the indexing problem (as explained in Sec. 2.2), where we will build on existing group testing approaches, in terms of representation and assignment. We will then turn our attention to query expansion (as explained in Sec. 2.3) in Chapters 6 and 7, and propose efficient extensions to existing manifold search techniques, such as querying multiple vectors at once and using faster solutions in the query time instead of random walks.

Chapter 3

Memory Vectors for Indexing

In this chapter, we consider the problem of searching for vectors similar to a query vector in a large database. The typical applications are the search and exploration in Big Media Data where documents are represented by feature vectors [WQSA15]. In this context, many papers report how the curse of dimensionality (due to the size of the vectors) makes indexing techniques ineffective [WSB98, ML14]. The recent paper [ML14] describing and analyzing the popular FLANN method experimentally observes that even this state-of-the-art method performs poorly on synthetic high-dimensional vectors, and the authors conclude that “random datasets are one of the most difficult problems for nearest neighbor search”.

Some strategies have been proposed to (partly) overcome this problem. For instance, the vector approximation file [WSB98] first relies on exhaustive search with approximate measurements and then computes the exact similarities only for a subset of vectors deemed of interest. The cosine sketch [Cha02] approximates cosine similarity with faster Hamming distance. Other works like spectral hashing [WTF09], Euclidean sketches [DCL08], product quantization [JDS11] and inverted multi-index [BL12] also rely on compact codes to speed up neighbor search while compressing the data. An interesting strategy is the Set Compression Tree [AZ14a], which uses a structure similar to a k-d tree to compress a set of vectors into an extremely compact representation. Again, this method is dedicated to small dimensional vectors (its authors recommend the dimension be smaller than $\log_2(N)$ where N is the size of the database) so that it is used in conjunction with a drastic dimension reduction via PCA to work with classical computer vision descriptors.

The main contribution of this chapter is a similarity search approach specifically adapted to high-dimensional vectors such as those recently introduced in computer vision to represent images [PD07, JDSP10]. The proposed indexing architecture consists of memory units, each of which is associated with several database vectors. A representative, called a memory vector, is produced for each memory unit and defined such that one can quickly and reliably infer whether or not at least one similar vector is stored in this unit by computing a single inner product with a new query.

Our approach is similar to the descriptor pooling problem in computer vision, but at a higher level. Many successful descriptors, such as BOV [CDF+04, SZ03], VLAD [JDSP10], FV [PD07], and EMK [BS09], encode and aggregate a set of local features into global representations. Yet, the new representation has a larger dimension than the local features. We use a similar approach at a higher-level: instead of aggregating local features into one global image representation, we aggregate global representations into group representations, so called memory vectors. These have no semantical meaning. Their purpose is to allow efficient search. Another difference is that we keep the same ambient dimension.

A second contribution of this chapter is to study similarity search from the perspective of statistical signal processing and decision theory. Our analysis provides insight into when and why the proposed approach provides low complexity (fast response time) without sacrificing accuracy. A third contribution is the experimental work using computer vision Big Media datasets as large as 100 million images. Our results suggest that the approach we propose can achieve accuracy comparable to state-of-the-art while providing results 5–10$\times$ faster.

This chapter is organized is as follows. Section 3.1 gives a formal problem statement. Section 3.2 focuses on the design of a single memory vector. We formalize the similarity of a query with the vectors of one memory unit as a hypothesis test. We derive the optimal representative vector under some design constraints and show how to compute it in an online manner. Section 3.3 proposes and analyzes two different ways to assign database vectors to memory units: random assignment and weakly supervised assignment which packs similar vectors into memory units. We provide a theoretical and experimental analysis of the different design and assignment strategies. Section 3.4 presents the results of experiments that evaluate our approach on standard benchmarks for image search. We use descriptors (vectors describing images) extracted with the most recent state-of-the-art algorithm in computer vision [JZ14]. Our results show the potential of our approach for this application. The contents of this chapter have been published in [Iscen et al., 2017b].

3.1 Problem Formulation

Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ denote a collection of d-dimensional vectors. We assume that all vectors are normalized so that $\|\mathbf{x}_i\|_2^2 = 1$ for all $i = 1, \ldots, n$.

Given a query vector $\mathbf{y} \in \mathbb{R}^d$ with $\|\mathbf{y}\|_2^2 = 1$, our objective is to determine which vectors $\mathbf{x}_i \in \mathcal{X}$ are most similar to $\mathbf{y}$. Determining which images would be "most similar" to a human is subjective and not easily quantified. The real image datasets used in the experiments we report in Section 3.4 include a set of queries and the corresponding human-determined response sets, which are treated as ground truth for those experiments. While humans can provide result sets for individual queries, generalizing to produce result sets for never-seen-before queries is still an open problem.

More generally, when such a ground truth is not given to measure performance, since all vectors lie on the unit sphere, we measure similarity using the inner product ${\mathbf{x}_i}^\top \mathbf{y}$. Then the problem of determining the most similar vectors to the query can be stated more precisely as: given $\alpha_0 > 0$, find all vectors $\mathbf{x}_i \in \mathcal{X}$ such that ${\mathbf{x}_i}^\top \mathbf{y} \geq \alpha_0$. The accuracy of a technique can be measured in terms of its precision and recall.

Clearly a naive baseline approach to this problem would compute all n inner-products ${\mathbf{x}_i}^\top \mathbf{y}$. Although it may provide perfect accuracy, this approach would have computational complexity (strongly related to run-time) of $\mathcal{O}(nd)$ operations. This is generally unacceptable when n and/or d are large, and we aim to obtain high accuracy while reducing the complexity.

3.2 Memory Vectors

Given a memory unit $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^d$, our objective in this section is to produce a representative, a so-called memory vector, such that, given a query vector $\mathbf{y}$ regarded as a random variable, we can efficiently perform a "similarity" test answering: is $\mathbf{y}$ a quasi-copy of, or similar to, at least one of the vectors of the memory unit?

For the sake of analysis, this section assumes that the vectors $\mathbf{x}_i$ are independent and identically distributed samples from a uniform distribution on the d-dimensional unit hypersphere. We model the query as a random vector $\mathbf{y}$ distributed according to one of the two laws:

- Hypothesis $\mathcal{H}_0$: $\mathbf{y}$ is not related to any vector in $\mathcal{X}$. $\mathbf{y}$ is then uniformly distributed on the unit hypersphere.
- Hypothesis $\mathcal{H}_1$: $\mathbf{y}$ is related to one vector in $\mathcal{X}$, say $\mathbf{x}_1$ without loss of generality. We write this relationship as $\mathbf{y} = \alpha\mathbf{x}_1 + \beta\mathbf{Z}$, where $\mathbf{Z}$ is a random vector orthogonal to $\mathbf{x}_1$ and $\|\mathbf{Z}\| = 1$. This means that $\mathbf{y}$ is more similar to $\mathbf{x}_1$ as α gets closer to 1. We have $\alpha^2 + \beta^2 = 1$ because $\|\mathbf{y}\| = 1$.

We look for a representation scheme satisfying the following design constraints. First, the set of vectors $\mathcal{X}$ is summarized by a single vector of the same dimension, $\mathbf{m}(\mathcal{X}) \in \mathbb{R}^d$, called the memory vector and denoted by $\mathbf{m}$ when not ambiguous. Second, the potential membership of query $\mathbf{y}$ to $\mathcal{X}$ is tested by thresholding the inner product $\mathbf{m}^\top \mathbf{y}$.

3.2.1 Sum-memory Vector: Analysis

A very simple way to define the memory vector is

$$\mathbf{m}(\mathcal{X}) = \sum_{\mathbf{x} \in \mathcal{X}} \mathbf{x}, \tag{3.1}$$

where we assume that $\mathcal{X}$ is composed of n different vectors. Albeit naive, this strategy offers some insights when considering high-dimensional spaces.

Iscen et al. [IFG+17] derive the pdf of the score $\mathbf{m}^\top \mathbf{y}$ under $\mathcal{H}_0$ when $\mathbf{m}$ is a known vector. This score has expectation 0 and variance $\|\mathbf{m}\|^2/d$, and it is asymptotically distributed as $\mathcal{N}(0, \|\mathbf{m}\|^2/d)$ as $d \to \infty$. This gives an approximate pdf of $\mathbf{m}^\top \mathbf{y}$ under $\mathcal{H}_0$. In contrast, under $\mathcal{H}_1$, the inner product equals

$$\mathbf{m}^\top \mathbf{y} = \alpha + \alpha \mathbf{m}(\mathcal{X}')^\top \mathbf{x}_1 + \beta \mathbf{m}(\mathcal{X}')^\top \mathbf{Z}, \tag{3.2}$$

with $\mathcal{X}' = \mathcal{X} - \{\mathbf{x}_1\}$. This shows two sources of randomness: the interference of $\mathbf{x}_1$ with the other vectors in $\mathcal{X}$ and with the noise vector $\mathbf{Z}$. Assuming that $\mathbf{y}$ is statistically independent of the vectors in $\mathcal{X}'$ (this implies that the vectors of $\mathcal{X}$ are mutually independent), we have

$$\begin{aligned} \mathbb{E}_{\mathbf{y}}[\mathbf{m}^\top \mathbf{y} | \mathcal{H}_1] &= \alpha, \\ \mathbb{V}[\mathbf{m}^\top \mathbf{y} | \mathcal{H}_1] &= \|\mathbf{m}(\mathcal{X}')\|^2/d. \end{aligned} \tag{3.3}$$

Assuming that $\mathcal{X}$ is composed of $n < d$ statistically independent vectors on the unit hypersphere also gives $\mathbb{E}_{\mathcal{X}}[\|\mathbf{m}(\mathcal{X})\|^2] = n$ and $\mathbb{E}_{\mathcal{X}'}[\|\mathbf{m}(\mathcal{X}')\|^2] = n - 1$. To summarize, for large d, we expect the following distributions:

$$\mathcal{H}_0 : \quad \mathbf{m}^\top \mathbf{y} \sim \mathcal{N}(0, n/d), \tag{3.4}$$

$$\mathcal{H}_1 : \quad \mathbf{m}^\top \mathbf{y} \sim \mathcal{N}(\alpha, (n-1)/d). \tag{3.5}$$

Making a hard decision by comparing the inner product to a threshold τ, the error probabilities (false positive and false negative rates) are given by

$$\mathbb{P}_{\text{fp}} \approx 1 - \Phi\left(\tau \sqrt{d/n}\right) \tag{3.6}$$

$$\mathbb{P}_{\text{fn}} \approx \Phi\left((\tau - \alpha)\sqrt{d/(n-1)}\right), \tag{3.7}$$

where $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt$.

The number of elements one can store in a sum-memory vector is linear with the dimension of the space when vectors are drawn uniformly on the unit hypersphere. This construction is therefore useful for high-dimensional vectors only, as opposed to traditional indexing techniques that work best in low-dimensional spaces.

If the vectors were pair-wise orthogonal, the dominant source of randomness (the interference between $\mathbf{x}_1$ and vectors of $\mathcal{X}'$) is cancelled in (3.2). The variance under $\mathcal{H}_1$ reduces to $\beta^2(n-1)/d$. This prevents any false negative if $\beta \to 0$. We further exploit this intuition that orthogonality helps in the next section.

3.2.2 Optimization of the Hypothesis Test per Unit

We next consider optimizing the construction of the memory vector of a given set $\mathcal{X}$. Denote the $d \times n$ matrix $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]$. We impose that, for all i, $\mathbf{x}_i{}^\top \mathbf{m}(\mathcal{X}) = 1$ exactly and not only in expectation, as assumed above. In other words, $\mathbf{X}^\top \mathbf{m} = \mathbf{1}_n$ where $\mathbf{1}_n$ is the length-n vector with all entries equal to 1. Achieving this, when $\mathbf{y} = \mathbf{x}_1$, we eliminate the interference with the remaining vectors in $\mathcal{X}'$ which was previously the dominant source of noise. In other words, under $\mathcal{H}_1$, Eq. (3.2) becomes

$$\mathbf{m}^\top \mathbf{y} = \alpha + \beta \mathbf{m}^\top \mathbf{Z}. \tag{3.8}$$

Under $\mathcal{H}_0$, the variance of the score remains $\|\mathbf{m}\|^2/d$. Therefore, the norm of the memory vector is the key quantity determining the false positive probability.

We thus seek the representation $\mathbf{m}$ minimizing the energy $\|\mathbf{m}\|^2$ subject to the constraint that $\mathbf{X}^\top \mathbf{m} = \mathbf{1}_n$. If multiple solutions exist, the minimal norm solution is given by the Moore-Penrose pseudo-inverse [RM72]:

$$\mathbf{m}^\star = (\mathbf{X}^+)^\top \mathbf{1}_n. \tag{3.9}$$

Since $n < d$, $\mathbf{m}^\star = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{1}_n$. If no solution exists, $\mathbf{m}^\star$ is a minimizer of $\|\mathbf{X}^\top \mathbf{m} - \mathbf{1}_n\|$. This formulation amounts to treating the representation of a memory unit as a linear regression problem [B+06] with the objective of minimizing over $\mathbf{m}$ the quantity $\|\mathbf{X}^\top \mathbf{m} - \mathbf{1}_n\|^2$. Taking the gradient, setting it equal to zero, and solving for $\mathbf{m}$ gives back $\mathbf{m}^\star$. When possible, and for large d, this new construction leads to the distributions:

$$\mathcal{H}_0 : \quad \mathbf{m}^{\star\top} \mathbf{y} \sim \mathcal{N}(0, \|\mathbf{m}^\star\|^2/d) \tag{3.10}$$

$$\mathcal{H}_1 : \quad \mathbf{m}^{\star\top} \mathbf{y} \sim \mathcal{N}(\alpha, \beta^2\|\mathbf{m}^\star\|^2/d). \tag{3.11}$$

The major improvement comes from the reduction of the variance under $\mathcal{H}_1$ for small values of β^2, i.e., $\alpha \lesssim 1$. Iscen et al. [IFG+17] show that if the vectors of $\mathcal{X}$ are uniformly distributed then $\|\mathbf{m}^\star\|^2$ is larger in expectation than the square norm of the naive sum representation from Section 3.2.1. The reduction of the variance under $\mathcal{H}_1$ comes at the price of an increase of the variance under $\mathcal{H}_0$. However, this increase is small if n/d remains small. For large d, we have

$$\mathbb{P}_{\text{fp}} \approx 1 - \Phi\left(\tau\sqrt{\frac{d}{n} - 1}\right), \tag{3.12}$$

$$\mathbb{P}_{\text{fn}} \approx \Phi\left(\frac{\tau - \alpha}{\beta}\sqrt{\frac{d}{n} - 1}\right). \tag{3.13}$$

Note that $\beta = \sqrt{1 - \alpha^2}$ is a decreasing function of α. Therefore, if $\tau < \alpha$, $\mathbb{P}_{\text{fn}}$ is a decreasing function of α. In particular $\mathbb{P}_{\text{fn}} \to 0$ when $\alpha \to 1$ as claimed above. In contrast to the naive sum approach from Section 3.2.1, there is no longer false negative when the query $\mathbf{y}$ is exactly one of the vectors in $\mathcal{X}$. This holds for any value of $\tau < 1$ when $\alpha = 1$, so that the false positive rate can be as low as $1 - \Phi(\sqrt{d/n - 1})$.

Remark. This solution is identical (up to a regularization) to the "generalized max-pooling" method introduced to aggregate local image descriptors [MP14]. However in our case the aggregation is performed on the database side only. Our solution is moreover theoretically grounded by a hypothesis test interpretation.

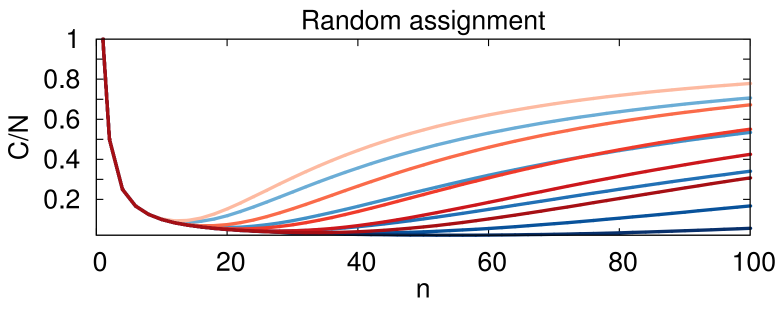


Figure 3.1: Ratio of the global cost by the cost of the exhaustive search $C_{\mathcal{H}_0}/N$ as a function of n using random assignment on synthetic data. Setup: $d = 1000$, $\epsilon = 10^{-2}$. The different curves correspond to values of $\alpha_0 \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$. Red and blue lines correspond to *sum* and *pinv* constructions respectively. Darker shades correspond to higher α_0.

3.2.3 Weakly Supervised Assignment

We now analyze a scenario where the vectors packed in the same memory unit are random but no longer uniformly distributed over the hypersphere: There is some correlation among them. The vectors in a memory unit are now uniformly distributed over a spherical cap [IFG+17]. This models the effect of a pre-processing which analyses the database vectors in order to assign similar vectors to memory units. For instance, Section 3.3.1 uses the k-means algorithm to process batches of database vectors.

We derive the same analysis as in the previous subsection with expectations and variances which now depend on the angle of the spherical cap. These expressions are complex and their derivation is detailed in [IFG+17]. In summary, the Kullback-Leibler distance between the distributions of $\mathbf{m}^\top \mathbf{y}$ under both hypotheses increases as the spherical cap gets narrow. In other words, identifying the positive memory units becomes easier when we assign correlated vectors to the same memory unit. Interestingly, this mechanism helps the sum construction more than pinv, so that when the vectors are very correlated, both constructions indeed perform equivalently.

3.3 Experimental Investigation

We next consider application scenarios where we need to store a large number N of vectors and perform similarity search. One memory vector is not sufficient to achieve a reliable test. We therefore consider an architecture that consists of M memory vectors. The search strategy is as follows. A given query vector is compared with all the memory vectors. Then we compare the query with the vectors stored in the memory units associated with the high responses, i.e., those likely to contain a similar vector.

3.3.1 Experimental Setup

Our experimental investigations are carried out using synthetic data (vectors uniformly distributed over the hypersphere) as well as real data, which are described in this section.

Datasets. We use the Inria Holidays [JDS08], Oxford5k [PCI$^+$07], and UKB [NS06] image datasets in our experiments. Additionally, we conduct large scale experiments in Holidays+Flickr1M, which is created by adding images from the Flickr1M [PCI$^+$07] dataset to the Holidays dataset. Similarly we use the recently introduced Yahoo100M dataset [TSF$^+$16] to increase the dataset size.

Descriptors. We use the triangular embedding descriptor [JZ14], denoted by ϕ_Δ. The only difference is that we do not apply the "powerlaw normalization" to better illustrate the benefit of the pinv technique for the memory vector construction compared to the sum (when applying the powerlaw normalization, both designs perform equally well since the vectors are nearly orthogonal). Ultimately, we have $d = 8064$ (or $d = 1920$) dimensional feature vectors for each image, obtained by using a vocabulary of size 64 (resp. 16). For large experiments in Yahoo100M, we use $d = 1024$ VLAD descriptors as extracted by [SXPK$^+$14].

We also experiment using deep learning features ($d = 4096$) provided by Babenko et al. [BSCL14]. As explained in their paper, the performance for the UKB dataset drops with adapted features trained on the Landmarks dataset. Therefore, we use the original neural codes trained on ILSVRC for the UKB dataset, and the adapted features for Holidays and Oxford5k.

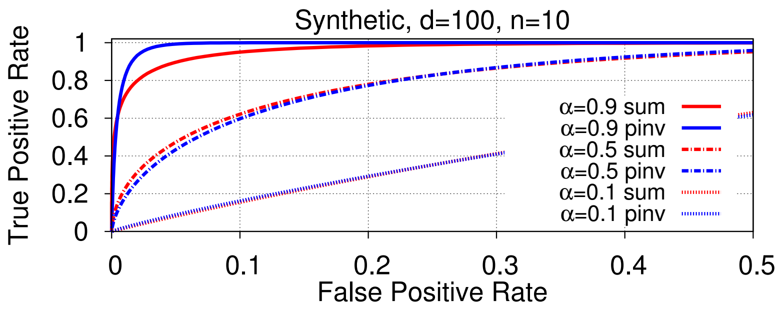
3.3.2 Random Assignment

We suppose that the N vectors in the database are randomly grouped into M units of n vectors: $N = nM$. We aim at finding the best value for n. When the query is related to the database (i.e., under $\mathcal{H}_1$), we make the following assumption: $\alpha_0 < \alpha < 1$, and we fix the following requirement: $\mathbb{P}_{\text{fn}} < \epsilon < 1/2$. Since $\mathbb{P}_{\text{fn}}$ is a decreasing function of α, we need to ensure that $\mathbb{P}_{\text{fn}}(\alpha_0) = \epsilon$. This gives us the threshold τ:

$$\tau = \mu_{\mathcal{H}_1} + \sigma_{\mathcal{H}_1}\Phi^{-1}(\epsilon), \tag{3.14}$$

with $\mu_{\mathcal{H}_1}$ and $\sigma_{\mathcal{H}_1}$ being the expectation and the standard deviation of $\mathbf{m}_j{}^\top\mathbf{y}$ under $\mathcal{H}_1$. Note that $\Phi^{-1}(\epsilon) < 0$ because $\epsilon < 1/2$ so that $\tau < \alpha$. The probability of false positive equals $1 - \Phi(\tau/\sigma_{\mathcal{H}_0})$ which depends on n, denoted by $\mathbb{P}_{\text{fp}}(n)$. This is indeed an increasing function for both memory vector constructions. Now, we decide to minimize the expectation of the total computational cost $C_{\mathcal{H}_0}$ when the query is not related. We need to compute one inner product $\mathbf{m}_j{}^\top\mathbf{y}$ per unit, and then to compute n inner products $\mathbf{x}_i{}^\top\mathbf{y}$ for the units giving a positive detection. In expectation, there are $M \cdot \mathbb{P}_{\text{fp}}(n)$ such units, and so

$$C_{\mathcal{H}_0} = M + M \cdot \mathbb{P}_{\text{fp}}(n) \cdot n = N(n^{-1} + \mathbb{P}_{\text{fp}}(n)). \tag{3.15}$$



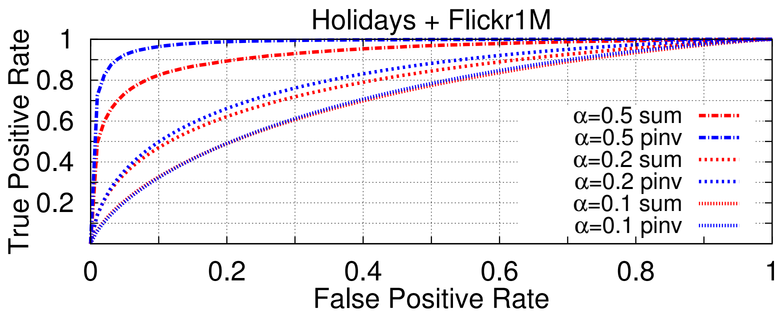


Figure 3.2: ROC curves ($1-\mathbb{P}_{\mathrm{fn}}$ as a function of $\mathbb{P}_{\mathrm{fp}}$) for *sum* and *pinv* constructions evaluated using (*top*) synthetic data with $d = 100$ and $n = 10$, (*bottom*) Holidays+Flickr1M real dataset and ϕ_Δ features with $d = 1920$ and $n = 10$.

The total cost is the sum of a decreasing function (n^{-1}) and an increasing function ($\mathbb{P}_{\text{fp}}(n)$).

For the random assignment strategy, there is a tradeoff between having a few big units (n large) and many small units (n small). Fig. 3.1 illustrates this tradeoff for different values of α_0 with synthetic data. It is not possible to find a closed form expression for the cost minimizer $n^\star$. When α_0 is close to 1, the threshold is set to a high value, producing reliable tests, and we can pack many vectors into each unit: $n^\star$ is large allowing a huge reduction in complexity. Even when α_0 is as small as 0.5, $n^\star$ is small but the improvement remains significant. In the setup of Fig. 3.1, the proposed approach has a complexity that is less than one tenth of that of searching through all database vectors (equivalent to $n = 1$). However, in order to increase the efficiency, we introduce an additional $\mathcal{O}(Md) = \mathcal{O}(dN/n)$ memory overhead for storing memory vectors.

Figure 3.2 depicts the theoretical and empirical Receiver Operating Characteristic (ROC) curves for different values of α. For the synthetic data, $\mathbb{P}_{\text{fn}}$ and $\mathbb{P}_{\text{fp}}$ are evaluated using Eq. (3.6) and (3.12). As expected the test performs better when α is closer to 1 and when $n \ll d$. For the real data, we use cosine similarity-based ground truth, since it is directly related with the model we considered theoretically. For each query vector, we deem a database vector as relevant if their cosine similarity is greater than α_0. To have enough ground-truth vectors, we look for these matching vectors on the Holidays+Flickr1M dataset using various α_0 values. This experiment using real data confirms the findings of the theoretical analysis. The pinv construction performs better than the sum as long as α_0 is big as explained in Sect. 3.2.2. However, the theoretical analysis is unable to predict performance levels on real dataset. It seems that the vectors of this real dataset have a much lower intrinsic dimensionality than their representational dimension $d = 1920$.

3.3.3 Weakly Supervised Assignment

A well-known technique in the approximate search literature is to partition the space $\mathbb{R}^d$ by clustering the database vectors. This assigns similar database vectors to the same cell [ML14]. In Section 3.2.3, we explain the advantage of a weakly supervised assignment by showing that the distance between the distributions of positive and negative memory units similarities increases. To show this point experimentally, we modify the spherical k-means clustering [DM01], so that the clusters are represented using pinv (or sum) in the update stage [1].

Better hypothesis test. Figure 3.4 shows that highly ranked memory units are very likely true positives containing at least one matching vector. On the contrary, with the random assignment, a positive memory unit may have a low rank. This means that we

[1]Note that, when we use such assignment techniques in spherical k-means, the number of vectors per clusters is not evenly distributed. The dot product may be dominated by long cluster representation vectors. Hence, we also propose a normalized version of the assignment, where the cluster representation vectors obtained are normalized to the unit norm.

now can analyse the database vectors of a shorter list of memory units to find most of the matching vectors.

More than one match. Another byproduct of weakly supervised assignment is that positive memory units are likely to contain more than one match, since matching vectors usually have high cosine similarity with each other. This helps the search efficiency by returning most of the matching vectors by only scanning a few positive memory units. This is also experimentally shown in Figure 3.4. With the random assignment, we have almost surely at most one matching vector in each positive memory unit.

Imbalance factor. We now analyze the cost of search with weakly supervised assignment. In Eq. (3.15), we assume that each unit contains n vectors. Up to this approximation, Fig. 3.3 shows that both constructions sum and pinv perform better as the inner correlation increases, but more surprisingly, they perform equivalently. It is also shown that it is possible to pack many vectors into the same memory unit with weakly supervised assignment, and still obtain a low search cost.

In practical applications, assuming that each unit contains a constant number of vectors is no longer true with weakly supervised assignment. This makes the analysis of the complexity more involved than (3.15). Moreover, this is potentially problematic in some applications: the complexity and thus the runtime can change dramatically from one query to another.

Imbalance factor is a metric to measure the impact of unbalanced clusters [JDS10]. It is defined as

$$\delta = M \sum_{i=1}^{M} {p_i}^2, \tag{3.16}$$

where M is the number of clusters, and p_i is the empirical probability that a database vector belongs to the i-th cluster. This is measured as frequency $p_i = n_i/N$, where n_i is the cardinality of the i-th cluster. Simple derivations give the following expectation and variance: $\mathbb{E}(n_i) = N/M$ and $\mathbb{V}(n_i) = (\delta - 1)N^2/M^2$. This shows that higher imbalance factor corresponds to clusters with varying sizes. This gives birth to a wide variability of the complexity from one query to another.

Table 3.1 shows the imbalance factor for different weakly supervised assignments. It is shown that pinv variants have more balanced cells compared to traditional sum, making the search process more effective. The negative effect of high imbalance factor in practice is better observed in Figure 3.5. In this figure, the algorithm visits a fixed number of positive memory units: 7 (Holidays), 30 (Oxford5k), or 60 (UKB). This roughly gives us a complexity ratio of $C_{\mathcal{H}_0}(\tau) \approx 0.2$ on average. We then show the complexity ratio per query in a histogram. It is clearly seen that the distribution for pinv has smaller standard deviation compared to sum, even though their means are almost the same. This makes pinv variant of spherical k-means a better alternative for weakly supervised assignment.

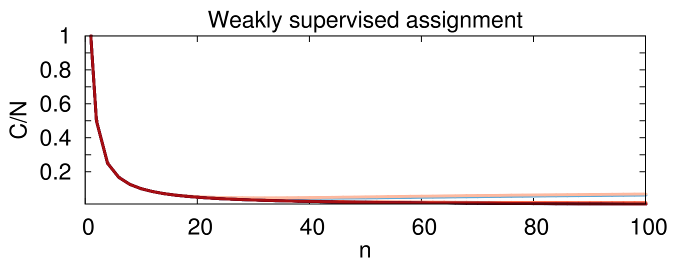


Figure 3.3: Ratio of the global cost by the cost of the exhaustive search $C_{\mathcal{H}_0}/N$ as a function of n using weakly supervised assignment and synthetic data. Setup: $d = 1,000$, $\epsilon = 10^{-2}$. The different curves correspond to values of $\alpha_0 \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$. Red and blue lines correspond to *sum* and *pinv* constructions respectively. Darker shades correspond to higher α_0.

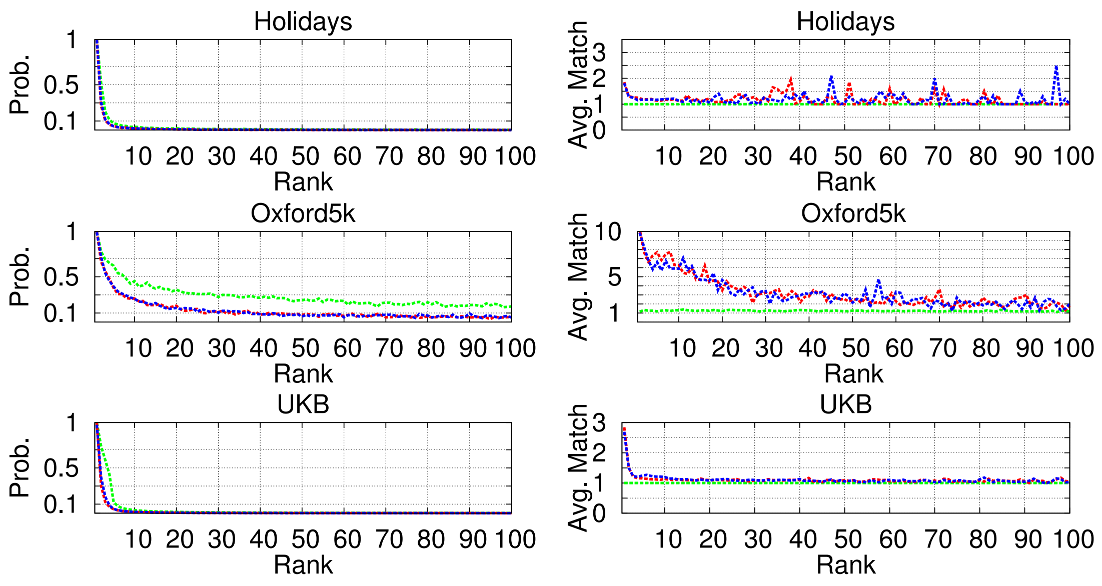


Figure 3.4: **Left:** Probability that a memory unit contains at least one match with respect to their rank. Green, red and blue lines correspond to random, *sum* spherical k-means and *pinv* spherical k-means respectively. There is a high probability of retrieving a match in highly ranked memory units, but it decreases faster with a weakly supervised assignment. **Right:** The average number of matches given that the memory unit is positive. Green, red and blue lines correspond to random, *sum* spherical k-means and *pinv* spherical k-means respectively. Using random assignment, we have about 1 matching vector per memory unit. The weakly supervised assignment improves this especially for higher-ranked units.

	sum	sum + norm.	pinv	pinv + norm.
Holidays	2.08	2.17	**1.90**	2.03
Oxford5k	2.76	2.69	2.27	**2.23**
UKB	2.58	2.56	**2.06**	2.09

Table 3.1: Imbalance factor for different datasets using *sum*, *pinv* and their normalized variants of k-means. Each dataset is clustered into $M = N/10$.

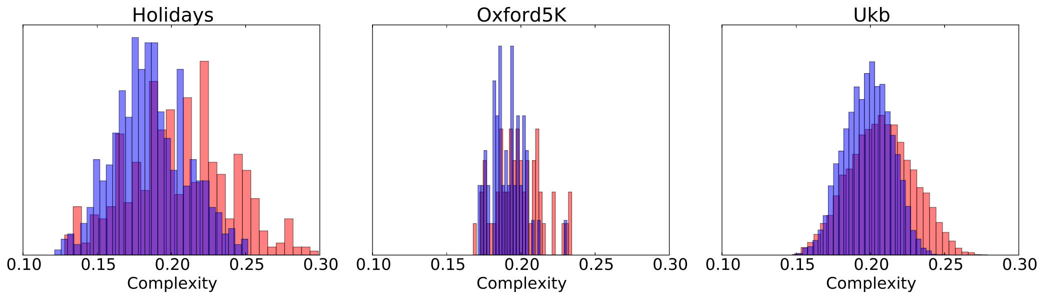


Figure 3.5: Complexity per query for *pinv* k-means (blue) and *sum* k-means (red). Although the averaged complexity ratios over all queries are similar, *pinv* has less variance thanks to lower imbalance factor.

3.4 Application to Image Search

This section shows that memory vectors perform extremely well on typical computer vision benchmarks. We assume two scenarios: closed-datasets in Section 3.4.1, and large-scale and streaming data in Section 3.4.2.

Whereas the datasets were already introduced in Sect. 3.3.1, let us describe the measure of performances. We follow the standard image retrieval protocol where each image is represented by a feature vector and the ground truth is now based on the visual similarity. The goal is to return visually similar images for a given query image. The similarity of two images is measured by the cosine of their descriptor vectors, and the images are ordered accordingly. We adopt the performance measure defined for each benchmark: mAP (mean average precision) which measures the area under the precision-recall curve [PCI$^+$07] or 4-recall@4, which is the average number of correct images in the top-4 positions.

As for the complexity, we first measure the similarities between the query and M memory vectors. We compare these similarities with a given threshold τ. Then, we re-rank all the vectors in positive memory units according to their similarities with the query vector. We characterize the complexity of the search per database vector by:

$$C_{\mathcal{H}_1}(\tau) = M + \sum_{i:\mathbf{y}^\top \mathbf{m}_i > \tau} n_i, \tag{3.17}$$

where n_i is the number of database vectors in the i-th memory unit. We measure the complexity ratio $C_{\mathcal{H}_1}(\tau)/N$ and the retrieval performance for different values of the

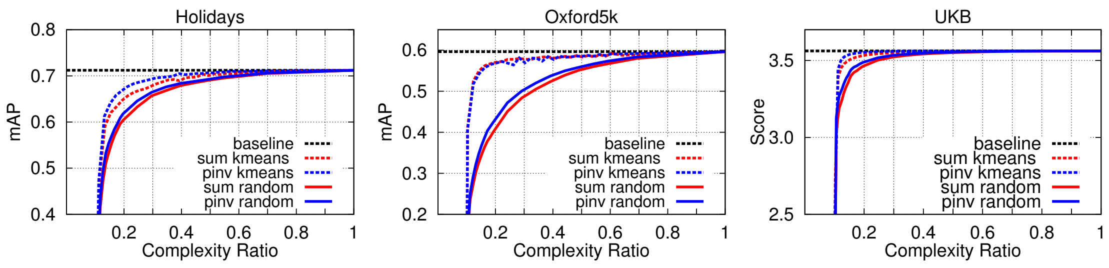


Figure 3.6: Image retrieval performance using visual similarity ground truth. K-means variants bring significant improvement.

threshold τ. For large τ, no memory unit is positive, resulting in $C_{\mathcal{H}_1}(\tau)/N = M$ and no candidate is returned. As τ decreases, more memory units trigger reranking.

3.4.1 Closed Dataset

Recall from Section 3.3 that weakly supervised assignment provides better approximate search than a random assignment. This is confirmed for the image search benchmark in Figure 3.6. Additionally, we show that it is possible to pack more vectors in a memory unit using weakly supervised assignment in Figure 3.7. We use this approach (spherical k-means with pinv) for the rest of our experiments.

The dimensionality of the descriptor linearly impacts the efficiency of any system. Dimensionality reduction with PCA is one way to improve this point. Our method is compatible with dimensionality reduction as shown in Fig. 3.8, where we reduce the vectors to $d' = 1024$ components. The search performance is comparable to the baseline with less computational complexity. We also apply our method to features learned with deep learning ($d = 4096$). Fig. 3.8 shows that the reduction in complexity also applies when using high performance deep learning features.

Compact codes are another way to increase efficiency. We reduce the dimensionality of the vectors to d'=1024 and binarize them by taking the sign of each component, in the spirit of cosine sketches [Cha02]. In the asymmetric case [GP11, JJG11], only memory vectors and dataset vectors are binarized, whereas in the symmetric case query vectors are also binarized during the query time.

Figures 3.9 and 3.9 show the performance when using compact binary codes. For the symmetric case, the sum method seems to perform better than pinv on the Holidays and Oxford5k datasets. In the asymmetric case, both methods perform similarly. In all cases, we achieve convergence to the baseline with a complexity ratio well below 1. Implementation efficiency is further improved in the symmetric case by using the Hamming distance calculation instead of dot product.

Comparison with FLANN [ML14]. Running the FLANN algorithm on the Holidays+Flickr1M dataset reveals that the convergence to the baseline is achieved with a

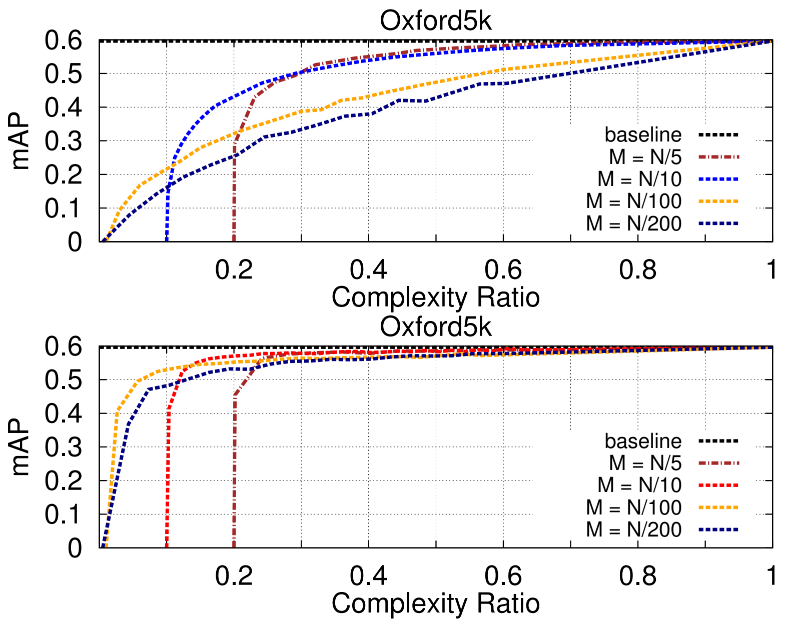


Figure 3.7: Search performance using random assignment (left) and weakly supervised assignment (right). This last option uses fewer memory units and still obtains a good search performance, which is not possible with random assignment.

speedup of 1.25, which translates to a complexity ratio of around 0.8. We can achieve similar performance with a complexity ratio of only 0.3. This confirms that FLANN is not effective for high-dimensional vectors. In this experiment, we use the autotune option of the FLANN library, and set target_precision = 0.95, build_weight = 0.01, and memory_weight = 0.

Execution time. We have shown that we get close to baseline performance while executing significantly fewer operations. We now measure the difference in execution time under a simple setup: $d = 1024$ and $N = 10^6$ dataset vectors. An average dot product calculation between the query and all dataset vectors is $0.2728s$. With $N/10$ memory vectors and $\approx 100k$ vectors in positive memory vectors, the execution time decreases to $0.0544s$. We improve the efficiency even further with symmetric compact codes and Hamming Distance computation: the execution time becomes $0.0026s$. Our method is parallelized for further improvement.

3.4.2 Large Scale and Streaming Data

We conduct large scale experiments on Holidays+Flickr1M and Yahoo100M datasets. The main advantage of our approach is its compatibility with large scale and streaming data, where pre-clustering the data may not be possible. More specifically, we assume that we have streaming images which we would like to index. As the size of the data keeps growing continuously, it is not possible to apply traditional k-means in such a

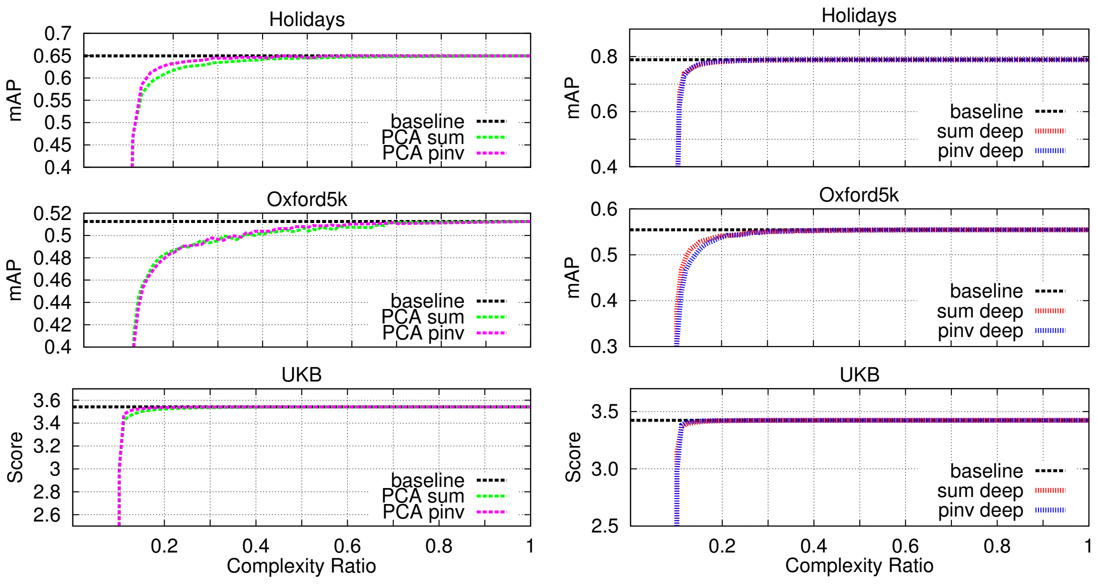


Figure 3.8: **Left**: The performance of memory vectors after PCA dimensionality reduction with $d' = 1024$. **Right**: Image retrieval performance with deep learning features ($d = 4096$), as trained by Babenko *et al.* [BSCL14]. Features for Holidays and Oxford5K are retrained on the Landmarks dataset, whereas the ones for UKB are trained on ILSVRC.

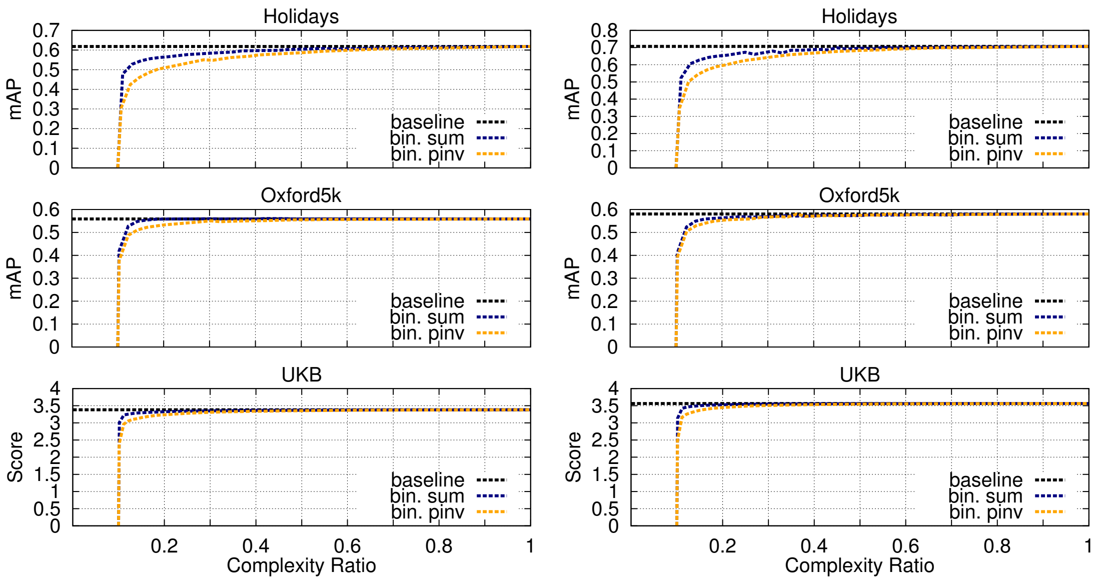


Figure 3.9: **Left:** Experiments with binary codes after PCA reduction to $d' = 1024$. The quantization is symmetric: real query vectors are binarized and then compared to binary memory vectors. **Right:** Experiments with binary codes after PCA reduction to $d' = 1024$. The quantization is asymmetric: real query vectors are compared to binary memory vectors.

scenario. We investigate two different approaches: random assignment and weakly supervised assignment over mini-batches.

Online indexing assumes that we would like to index items in streaming data as they become available. In such case, the random assignment is applicable provided that the successive vectors in the stream are independent.

Figure 3.10 shows the image retrieval performance based on random assignment with different group sizes n. With $n = 10$, the performance is close to the baseline while performing roughly three times fewer vector operations than exhaustive search. On the other hand, larger groups make it possible to have smaller complexity ratio with a degrading effect on the quality of search, since the scores obtained from memory vectors are noisier (see (3.6) and (3.12)). The pinv construction performs better than sum in all cases except for a very large memory units of size $n = 500$, where the quality of search is low in general.

Batch assignment. The alternative approach runs weakly supervised assignment on small batches. A batch spherical k-means is the same as the regular spherical k-means discussed in previous sections. We do not cluster the whole dataset at once because this would not be tractable with large collections. Instead, we randomly divide the dataset into batches of the same size and run a weakly supervised assignment separately for each batch. Fig. 3.11 shows that this strategy improves the performance while keeping the complexity of the clustering algorithm manageable.

We compare our approach to a well-known mini-batch k-means algorithm [Scu10] (referred to as mbk) as implemented by [PVG$^+$11]. We compare both strategies using a batch size of 10k, and show the image retrieval performance for different complexity ratio in Figure 3.12.

The first observation is that the plot of mbk is not smooth. This is due to the clusters being unbalanced when using such mini-batch approaches. In fact, when we measure the imbalance factor (3.16), we obtain $\delta = 183 \pm 8$. As a result, few clusters contain a large number of dataset vectors, and when a cluster is accessed, the cost of the verification step becomes expensive. On the contrary, the imbalance factor observed using our batch spherical k-means is only 2.47 ± 0.01, resulting in a more efficient verification step for positive memory units.

The complexity ratio per query for the two methods can be seen in Figure 3.13. When we set the number of positive memory units to 3500, the mean complexity ratio for our batch spherical k-means approach is 0.17, with a standard deviation of 0.01. On the other hand, when mbk is used, the mean increases to 0.26 with a standard deviation of 0.51.

Finally, we apply our batch assignment strategy to the Yahoo100M, which consists of 97.6 million vectors. We divide the dataset into batches of 100k, and run three different indexing strategies for each batch: random assignment with pinv, pinv-based spherical k-means and sum-based spherical k-means. This dataset does not have manually annotated ground truth or designated queries, therefore we use an existing

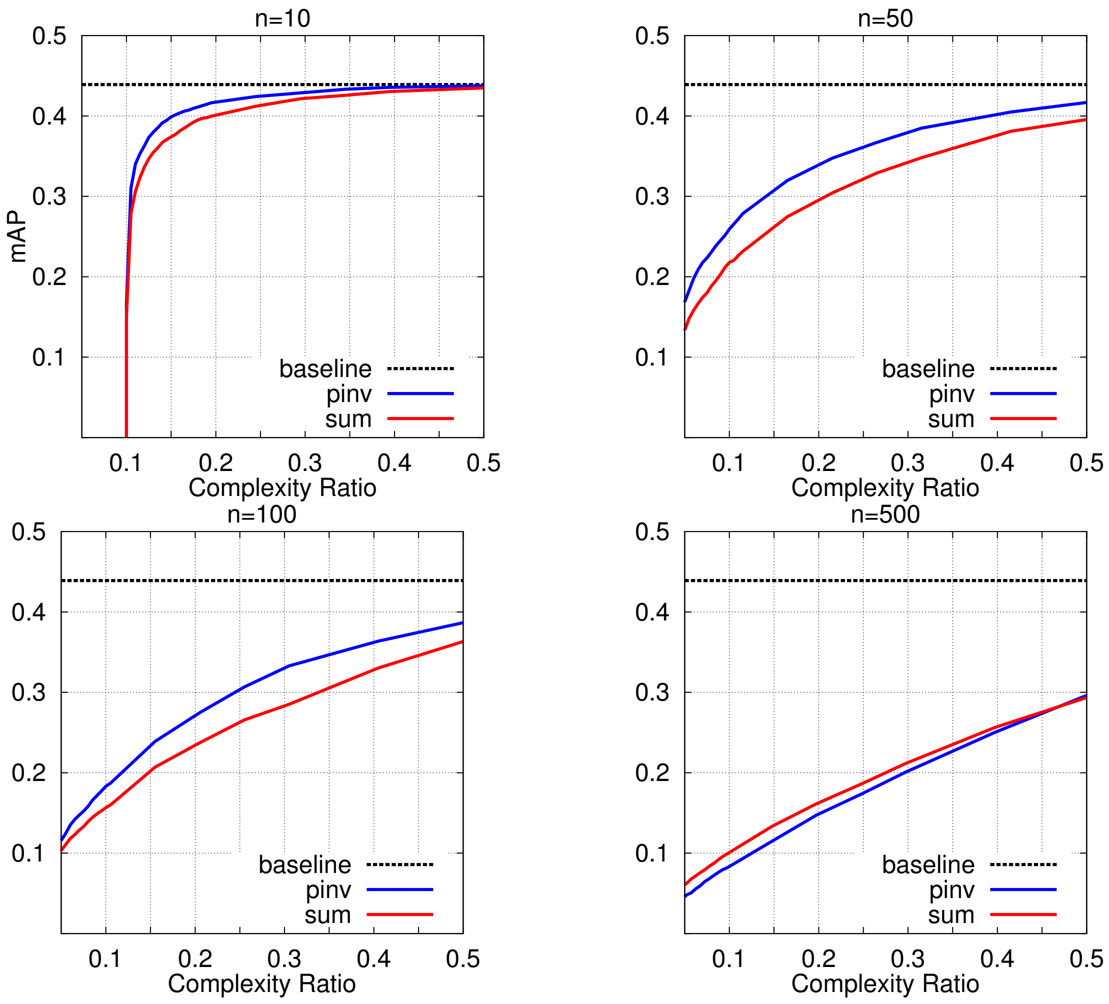


Figure 3.10: Image retrieval performance in Holidays+Flickr1M with different memory unit size n. The data is streamed in this scenario, and the memory untis are created randomly.

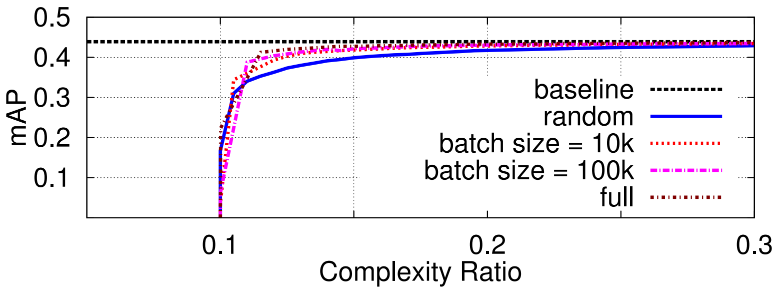


Figure 3.11: Comparison of random and weakly supervised assignments over batches in a large-scale setup. Option *full* corresponds to a unique batch of size 10^6. We run each experiment multiple times.

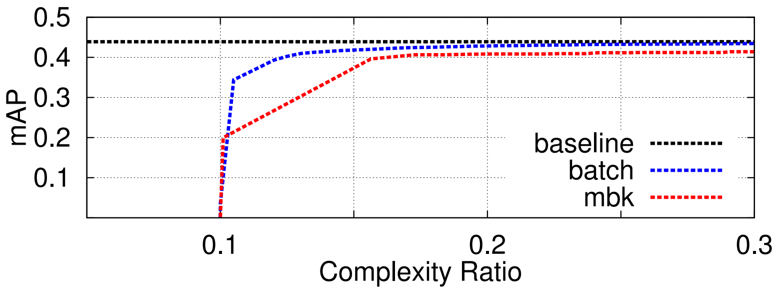


Figure 3.12: Comparison of our *batch spherical k-means* approach with the mini-batch k-means algorithm (mbk) [Scu10]. The batch size equals 10k.

evaluation protocol [IRF16]. Dataset vectors are considered a match if they have a similarity of 0.5 with the query vector. 1000 query vectors are randomly chosen and those that have 0 or more than 1000 matches are filtered out. In the end, we have 112 queries, with each query having 11.4 matches on average.

We present our results in Figure 3.14. Since we have $N/10$ memory vectors for each batch, a lower bound for the complexity ratio is 0.1. We see that pinv-based methods converge to the baseline much faster than sum. Using our pinv-based k-means variant, we achieve the same performance as baseline with 0.12 complexity ratio. This saves us almost 90 million vector operations at query time.

3.5 Conclusion

This chapter takes a statistical signal processing point of view for image indexing, instead of traditional geometrical approaches in the literature. This shift of paradigm allows us to bring theoretical justifications for representing a set of vectors. We have presented and analyzed two strategies for designing memory vectors, enabling efficient membership tests for real-valued vectors. We have also showed two possible assignment strategies and analyzed their performance theoretically and experimentally. For random assignment, the optimized pinv construction gives better results than the simple sum

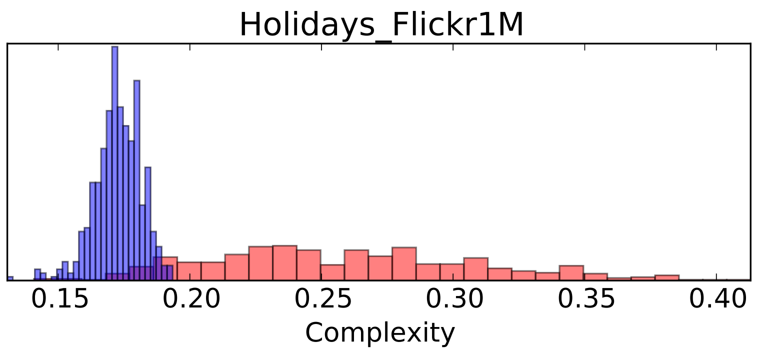


Figure 3.13: Complexity of each query for Holidays+Flickr1M using our batch spherical k-means (blue) and mbk (red). Our scheme has less complexity on average and less variance.

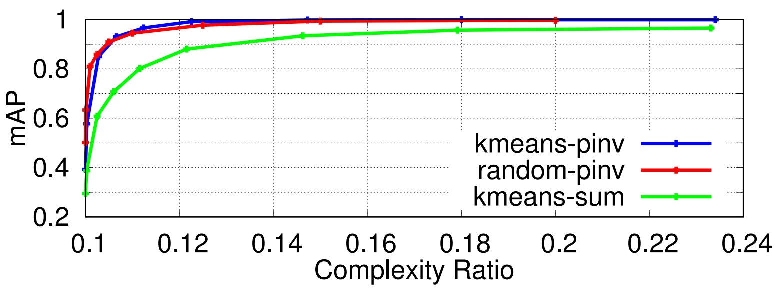


Figure 3.14: Comparison of *pinv* and *sum* based methods in Yahoo100M. Our *pinv*-based k-means variant achieves baseline performance with only 0.12 complexity ratio.

aggregation. On the other hand, when the vectors in the same memory unit share some correlation, sum is on par with the pinv construction as for the quality of the hypothesis test. Yet, the pinv construction when used in the weakly supervised assignment offers a lower imbalance factor. This yields less variability of the search runtime from one query to another. This procedure is done offline and its complexity is often ignored in the image search literature. On the contrary, we did pay attention to this bottleneck: we proposed to run the weakly supervised assignment by batch and showed that it does not spoil the overall performance of the image search.

Chapter 4

Applications and Extensions for Memory Vectors

In this chapter, we study practical extensions of memory vectors in a large-scale image search scenario, and their application in privacy-based multimedia domain. In Section 4.1, we examine the further required considerations to apply memory vectors in a large-scale instance search scenario. We show that, unlike raw high-dimensional image image descriptors, our group representations are easily indexable using existing space partitioning methods, such as FLANN. This makes the proposed system even more efficient in a large-scale scenario with millions of vectors.

Section 4.2 investigates the use of memory vectors in the privacy domain. The goal is to design an efficient multimedia retrieval system while keeping the data safe. We consider a use case of with three entities: the user, the server, and the owner of the data. We then show the effectiveness of memory vectors in such domain in terms of search efficiency and privacy. It is shown that our proposed system adds a second line of defense which makes it very hard for a "curious" user to reconstruct the data.

4.1 Scaling Group Testing with Memory Vectors

Most of the traditional high-dimensional indexing techniques fail due to the curse of dimensionality [WSB98]. The partitioning strategies they all use eventually fail at preserving neighbors in the same partitions, forcing the search procedure to scan many partitions in order to determine good candidates for a query [LJW$^+$07]. This extensive scanning is very inefficient and dramatically increases resource consumption and response time.

Similarity search techniques inspired by the group testing framework have recently been proposed in an attempt to specifically defeat the curse of dimensionality. Group testing has found applications in pharmacology, genome screening and has first been adapted to image retrieval by Shi et al. [SFJ14] and then extended by Iscen et al. [IFG$^+$17]. In this context, group testing specifically exploits the properties underpinning truly high-dimensional spaces: the correlation between two random unit vectors concentrates around zero as their dimension increases, facilitating by contrast the identification of correlated (hence similar) vectors.

Like some traditional high-dimensional indexing schemes, group testing first forms overlapping groups of features and computes a representative vector for each group. Then, at query time, a test is performed on each group by checking the query against the representative vector. The output of the test reveals how likely the query matches one of the features in that group. The decoding step identifies some candidate features by analyzing all the test outputs. A final verification step computes the true similarities of the candidate features with the query, and gives back a ranking from which it is easy to return the final result to the user.

Compared to more traditional high-dimensional indexing methods, the key differences with group testing are in the group formation strategies, the procedure to compute representatives and the decoding process that identifies candidates features from the test outputs. Works [SFJ14] and [IFG$^+$17] prove group testing is particularly robust at higher dimensionalities and allows to return high-quality results faster than the sequential scan, which is often the de facto preferred method to run at high-dimension.

The group testing paradigm as it is applied in [SFJ14] and [IFG$^+$17] can hardly be scaled to searching very large image collections. First, too many tests are performed at query time when checking the query against the group representatives — the larger the collection, the more such representatives, which limits scalability. Second, unnecessary computations are done at decoding time because most of the tests indeed yield negative output.

This section tackles these difficulties and proposes extensions to the group testing in order to handle larger collections of feature vectors. Specifically, it shows that it is surprisingly worth indexing the representatives, allowing to very quickly identify the few positive tests. It also shows that these few positive tests not only deliver enough information to identify the good candidates, but also accelerate the decoding step. In terms of quality, it shows that the group testing technique proposed here performs almost as well as other traditional indexing strategies (FLANN [ML14], LSH [IM98, GIM99, DIIM04]) at a much lower cost, however.

This work has been published in [Iscen et al., 2016a]. The reader is encouraged to read Section 2.2.3 for a background in group testing.

4.1.1 Discussion

We now discuss the cons (first) and then the pros of using group testing versus using traditional similarity search techniques when indexing large collections of high-dimensional features like Fisher vectors [PD07], VLAD [AZ13], T-embedding [JZ14] or other CNN-based [BSCL14, GWGL14] descriptors. This discussion ends with a list of requirements that group testing approaches must meet in order to cope with scale, i.e., indexing very large collections of high-dimensional vectors.

Even though it is a state-of-art library, FLANN does not perform well for truly high-dimensional data, as acknowledged by its authors [ML14]. The experiments described later confirm that the effectiveness of FLANN decreases when indexing such high-dimensional descriptors. It is only possible to return high-quality results, however, at the expense of a dramatic increase in the response time. This is caused by FLANN performing a very large number of checks to eventually gather enough good neighbors. This problem is not specific to FLANN but caused by the curse of dimensionality whose effects are mitigated with extensive multiprobing.

In contrast, group testing is less sensitive to the curse of dimensionality and better handles high-dimensional features. Yet, its recent application to image search [SFJ14, IFG+17] experiences scalability problems. First, group testing has to compare the query to all the representatives group vectors and that comparison is currently done using a linear scan. This is clearly not scalable when the number of representative group vectors grows, which is the case when indexing larger collections of images. Indexing larger collections (N grows) means for [SFJ14, IFG+17] that M as to grow too. At scale, this comparison step becomes excessively costly.

By construction, only a handful of group representative vectors are meaningful for a given query, hence most comparisons between the query and the group representative vectors are done in a pure waste. Quickly identifying the best representatives is therefore a key requirement for designing scalable group testing approaches.

More importantly, the effectiveness of the group testing proposed in [SFJ14, IFG+17] vanishes as N grows. Compared to a linear scan of the features in $O(Nd)$, the gain in complexity (as detailed in Sect. 2.2.3) is:

$$\gamma = \frac{M}{N} + \frac{M}{d} + \frac{R}{N}, \tag{4.1}$$

where M is the number of groups, and R is the size of the shortlist.

The second term shows that having $M << N$ is not sufficient: At scale, M will become larger than d, spoiling the gain in complexity. This is solely due to the decoding procedure of [SFJ14] and [IFG+17].

Furthermore, equation (2.23) is likely to sum extremely small values for the v_j associated to the representatives that turn out not to be relevant for a query. Therefore, identifying the best representatives and ignoring the others, i.e., setting their v_j to zero, diminishes the cost of decoding as many factors can now be excluded from the sum in equation (2.23). It is not needed to take into account similarities to all representatives as [SFJ14] and [IFG+17] do.

Finally, the contributions in [SFJ14] and [IFG+17] compute the true similarities between the query and $R \ll N$ features. According to [SFJ14] and [IFG+17], R is typically equal to $N/10$, which, at scale, becomes prohibitively expensive as well. Therefore, the value of R has to be significantly reduced to allow scaling. Works [SFJ14] and [IFG+17] can not easily diminish R without severely endangering quality. They essentially keep M small to control the cost of evaluating equation (2.23) (which corresponds also to the second term of (4.1)). This is done by choosing m small, the number of groups each image is assigned to(see eq. (2.20)). This implies that [SFJ14] and [IFG+17] have a rather small number m of evidences (i.e., test outputs v_j) to evaluate the likelihood u_i of being a match. Very efficiently identifying the best representatives allows to use a larger value for m which strengthens the redundancy between the groups.

Consequently, the likelihood scores of the database features identified become more reliable because these scores are computed from more test outputs. For the non matching features, their likelihoods become less noisy, i.e., shrink to zero, because $v_j = 0$ for irrelevant representative group vectors. This improves the reliability of the decoding and makes possible the use of a lighter verification step, i.e., using a small value for R.

On the pro side, the lessons learnt from the two group testing papers are the following ones. The most salient property of the group testing framework is its capacity to cope with truly high-dimensional datasets. This has been demonstrated in [SFJ14] where the performance improves as the dimensionality of the features increases. Iscen et al. [IFG$^+$17] show that the Moore-Penrose pseudo-inverse construction (see eq. (3.9)) produces more informative representatives compared to the simple sum that is used in [SFJ14]. [IFG$^+$17] also demonstrates that representatives are more informative when some form of similarity between features is used to weakly guide the creation of the groups instead of relying on a pure random strategy. They derived groups from running a k-means which clusters similar features in groups of very different cardinalities. However, reducing the unbalanced cardinalities between these groups is key to avoid similarities being dominated by overcrowded groups.

Taking the lessons learnt from [SFJ14, IFG$^+$17] for granted, the above discussion allows to isolate two main requirements that must be addressed in order to design scalable group testing techniques:

- Requirement R1: Quick identification of the best representatives for a query and removal of the bottleneck from eq. (2.23) by ignoring irrelevant v_j values.
- Requirement R2: Increasing the groups redundancy to improve quality (m) allowing to decrease the number of true similarities calculations (R).

4.1.2 Scaling Group Testing

This section describes the extensions to the group testing framework in order to facilitate scaling. The resulting extended group testing algorithm is called sGT which stands for scalable group testing. We detail how the learnt lessons and the requirements detailed above can be addressed.

4.1.2.1 Even-Size Groups of Similar Vectors

We first address the lessons learnt from [SFJ14, IFG$^+$17]. In order to create even-size groups of similar features, we propose to use the random k-d tree algorithm from FLANN. That algorithm gradually cuts in two equal subsets the collection of vectors to be indexed, until the bottom leaf of the k-d tree contains one unique feature. It is easy to navigate down a fully constructed k-d tree and identify the node where the two child sub-trees each consist of a total of n vectors. This is where we take control of the algorithm in the FLANN library: All the vectors below that particular node now go into two distinct groups. In order to have each feature assigned to m groups, we simply launch the creation of m random k-d trees.

Once the groups have been formed this way, we compute the pseudo-inverse for each group in order to determine its representative (see eq. (3.9)). It is important to note that we take control of the construction process of the k-d trees in FLANN. Once the representatives are stored, the k-d trees are no longer needed and they are deallocated.

Please note that we have taken control of the k-d tree construction per se and there is nothing really specific to FLANN here. Using FLANN is handy because it is a state-of-the-art library anyone can download, facilitating the reproduction of our methodology, and because it better copes with high-dimensionality (compared to the regular k-d tree) thanks to his enhanced dimension splitting criterion.

4.1.2.2 Indexing Representative Vectors

We then address requirements R1 and R2. It is rather straightforward to insert the group representative vectors into an index in order to circumvent the linear scanning (see eq. (2.22)) limiting the scalability of [SFJ14] and [IFG$^+$17]. Instead of determining the similarities to all M representatives, it is worth probing the index and get back the k representatives with the highest similarities to the query.

First, this is faster especially because most modern high-dimensional techniques are approximate and trade response-time for efficiency. Second, it is safe to force to zero the similarities v_j between the query and the $(M-k)$ representatives that remain. Since k is typically orders of magnitude smaller than M, $(M-k)$ is a large value which saves a lot of resource consumption when running the decoding step: many $v_j = 0$, making the computation of equation (2.23) very sparse. Requirement R1 is hereby addressed.

Two comments are in order, however. First, most high-dimensional indexing schemes work for Euclidean distances. This is the case for some of the algorithms implemented within the FLANN library. It is not very complicated to translate a ℓ_2 distance into a dot product as it is needed to evaluate equation (2.22). We ℓ_2-normalize the representative vectors to be compatible with the query whose norm equals one. We ask FLANN to index the database of these normalized representative vectors $\bar{\mathbf{y}}_j = \mathbf{y}_j/\|\mathbf{y}_j\|$. For the k nearest neighbors output by FLANN, we compute the similarities:

$$v_j = \frac{2 - \|\bar{\mathbf{y}}_j - \mathbf{q}\|^2}{2}\|\mathbf{y}_j\| = (\bar{\mathbf{y}}_j^\top \mathbf{q})\|\mathbf{y}_j\| = \mathbf{y}_j^\top \mathbf{q}. \tag{4.2}$$

This conversion needs the extra storage of one scalar, $\|\mathbf{y}_j\|$, per representative vector. After obtaining these similarities, we compute (2.23) and re-rank R top scoring features based on their true similarities with the query.

Second, indexing the M representatives facilitates increasing m, which is the number of groups each feature is assigned to. This addresses the requirement R2 and also positively impacts the quality of the results returned by group testing indexing algorithms. The time it takes to retrieve k representatives indeed does not change that much when M increases. It is anyway much faster than the linear scan. Works [SFJ14] and [IFG$^+$17] had to set m to a small value in order to reduce the complexity of their implementations when analyzing the M representatives. They had a complexity objective. In contrast, indexing these representatives to quickly retrieve k of them allows us to set m such that it is possible to meet a quality objective. At last, this better quality yields a more reliable ranking of the candidates. The true matching features are ranked higher. This allows us to reduce the size R of the shortlist and to decrease the complexity of the final verification step.

4.1.2.3 Summary

We argue that it is possible to scale group testing high-dimensional approaches provided that these lessons learnt from [SFJ14, IFG$^+$17] are enforced and the two requirements presented above are addressed. We therefore implemented within the sGT algorithm a group construction strategy based on k-d tree to marry fixed size groups and some similarity between the vectors of each group. sGT also includes the implementation of the indexing of the representative vectors to quickly identify the k best representatives. This, in turn, allows to increment the redundancy of vectors in groups (m), improving quality. The candidate vectors being more accurate, it allows to decrease the length of the candidate list R, which saves response time.

The next section demonstrates that these extensions indeed make group testing approaches easier to scale to larger collections of very high-dimensional vectors.

4.1.3 Experiments

This section basically compares the behavior of three high-dimensional indexing schemes. Two of them are state-of-the-art: the first technique is the traditional k-d tree based index from the FLANN library [ML14]. The second technique is our own implementation of the original group testing scheme proposed by Shi et al. [SFJ14]. The third technique is sGT which includes the various extensions discussed in the previous section aiming at better handling large scale collections.

We first describe our experimental setup before moving to the experiments themselves where the comparison of the behavior of the three techniques is made using the mean average precision and running time indicators.

4.1.3.1 Experimental Setup

We use a public large-scale image retrieval benchmark dataset known as Holidays+Flickr1M for our experiments. This dataset is the combination of the original Holidays image dataset [JDS08] with 1 million distractor images from Flickr1M [PCI$^+$07]. The dataset contains N=1,001,491 images in total. The image retrieval quality is evaluated with the provided ground-truth for 500 queries and measured by mean average precision (mAP).

To facilitate the direct comparison with existing methods [SFJ14, IFG$^+$17], we use the same T-embedding features [JZ14] with $d = 1{,}920$ which corresponds to a codebook vocabulary of size 16. These features are made available online.[1] Following the pipeline presented in the original paper [JZ14], we post-process these features with power-law normalization, setting $\alpha = 0.5$.

4.1.3.2 Image Features are Hardly Indexable

We first show that the T-embedding features extracted from the images can hardly be indexed due to their dimensionality. This motivates the need for alternative high-dimensional indexing schemes such as group testing based approaches better coping with the curse of dimensionality.

We started by using the linear search option of FLANN in order to determine the quality and the time it takes when using the 500 queries. It takes 901 seconds to run the entire query set, and the resulting mAP is equal to 51.9. These values form the baseline we will refer to in the remainder of this section.

We then ask FLANN to determine what would be the best high-dimensional strategy in order to index the image features. We therefore set the `FLANNParameters` internal data structure to `AUTOTUNED` with a target precision set to 0.95. The resulting index determined by FLANN is a k-d tree. It takes 677 seconds to probe the index with the 500 queries. Using the index instead of relying on the linear scan is only 1.29 times faster. This little speedup is explained by the very large number of checks FLANN has to perform in order to meet the required precision. Observing the logs produced by FLANN shows that close to 450,000 checks are done for each query – so half of the features are indeed scanned, making the indexing useless as it can not confine the search to a small portion of the database. This is a direct consequence of the high-dimensionality of the indexed features.

The T-embedding features are indeed of a very large dimensionality. Their representational dimension is 1,920 while their local intrinsic dimensionality (see [ACF+15]) is estimated to be 150.3 ± 41.3. This is a very high value, which emphasizes the negative consequences of the curse of dimensionality. Please note that FLANN proved to work with descriptors having a much smaller intrinsic dimensionality. For example, SIFT has a representational dimension equal to 128 while its estimated local intrinsic dimensionality is 12.3 ± 3.0 (again, see [ACF+15], Section 6.3). Their indexing is thus much easier.

One obvious approach to circumvent this dimensionality problem is to apply a dimension reduction technique to the features with the hope not to degrade too severely the resulting mAP. We therefore took our collection of image features, applied a PCA and kept a varying number of the most significant resulting dimensions. We then ran the (PCA-ed) queries against that transformed collection and computed the resulting mAP when using a linear scan. Figure 4.1 plots the outcome of this experiment. The figure shows that the mAP rapidly decreases when reducing the number of dimensions for the transformed features. While such transformed features might be more indexable, the resulting quality would be too low to have any practical interest. Using a linear scan is not an option either as it is too slow.

[1] `http://people.rennes.inria.fr/Ahmet.Iscen/memoryvectors.html`

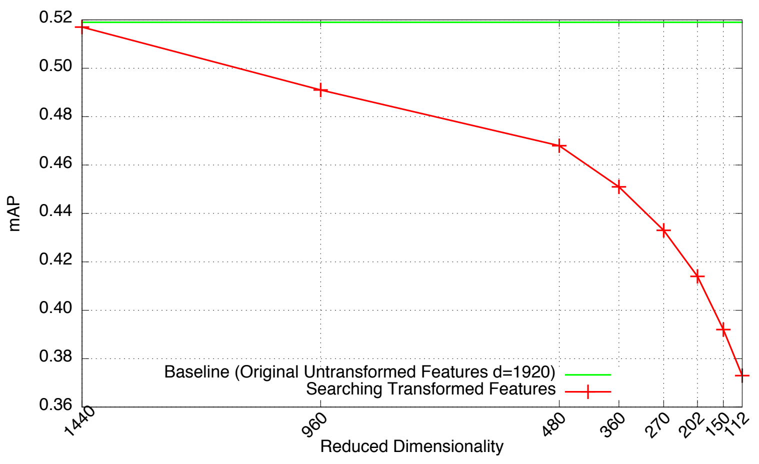


Figure 4.1: Observed mAP after PCA applied to the T-embeddings image features. Varying number of retained dimensions.

Most modern image features (VLAD, CNN-based features, etc.) share with the T-embeddings this property of lying in truly high-dimensional spaces. They can therefore hardly be indexed using traditional approaches. The following experiments demonstrate that group testing strategies better handle dimensionality.

4.1.3.3 Even-Size Groups of Similar Vectors

The lessons learnt from studying the literature suggest that creating even-size groups of similar vectors is best for group testing approaches. We proposed in Section 4.1.2.1 to take control of the random k-d tree creation process to this aim. The experiment described now shows this indeed works as the resulting performances are better. The competitive group testing method we compare against is [SFJ14]. Its group construction uses a fully random assignment procedure to create even-size groups. This experiment uses a similar setting as the one defined in [SFJ14], i.e., $n = 15$, $m = 2$, and R=M=133,532.

The mAP when using the pure random group creation technique is 50.8. In contrast, the mAP we observe when taking control of the k-d tree reaches 51.6. Forming groups according to some similarity (even quite weak as we utilize higher levels of the k-d tree) improves quality, which is however slightly below the baseline (mAP=51.9).

4.1.3.4 Redundancy for Better Candidate Vectors

When discussing the design options to scale group testing, we highlighted the necessity to decrease the number of true similarity computations between the query and the candidate features. This is requirement R2. We also claimed this goal could be achieved by increasing the redundancy of features in groups, i.e., increasing the value for m. The experiments discussed now are evaluating the mAP for sGT, varying the values of m

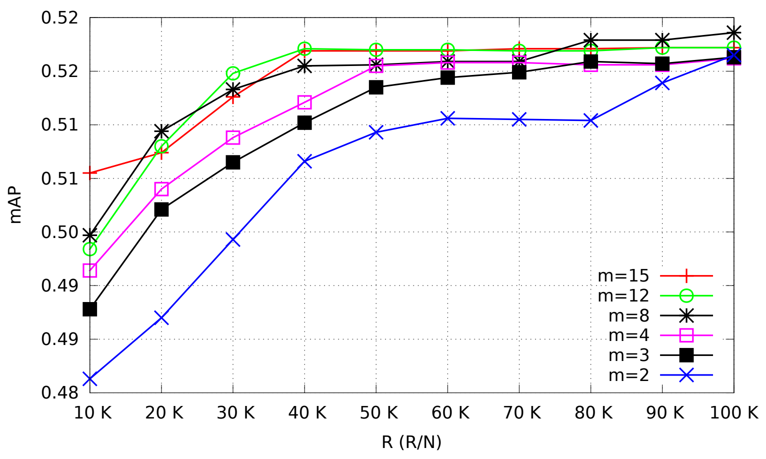


Figure 4.2: Search quality with different m and R. Percentages in parenthesis give R/N.

and R.

The resulting experiments, illustrated by the Figure 4.2, show the mAP achieved by sGT when m goes from 2 (low redundancy) to 15 (high redundancy) as well as when R goes from 10,000 (small value, 1% of the database) to 100,000 (large value, 10% of the database, same settings as in [SFJ14]). It can be observed that for a fixed value for R, a better mAP is reached if m is larger. In other words, it is worth increasing m which reduces R for a fixed mAP.

However, increasing m also increases M (see eq. (2.20)), the total number of groups. This has a direct impact on the costs of equations (2.22) and (2.23). It is possible to keep these costs under control thanks to the indexing of the group representatives, as demonstrated by the next experiment.

4.1.3.5 Indexing Group Representatives

Indexing the vectors that represent the groups is a way to meet the requirement R1 defined above. It is indeed quite straightforward. We used the same settings as above, with $m = 3$. We then give the collection of vector representatives to FLANN instructed to automatically determine the best index thanks to its `AUTOTUNE` setting. When $m = 3$, there are 200,298 representatives corresponding to as many groups created with the taking control of the k-d tree process. FLANN finds that it is best to index that collection using a k-d tree which will provide a speed up of 39.26 compared to the linear scanning of all the representatives. Higher settings of m confirm significant speed up gains: FLANN estimates the speed up to be equal to 82.96 when $m = 4$ and 74.67 when $m = 8$. The scalability objective linked to R1 is therefore achieved.

It is interesting to try to index group representatives when they result from the random construction process. Asking FLANN to determine the best index when fed with such 200,298 representatives fails: the library estimates that it is the linear scan

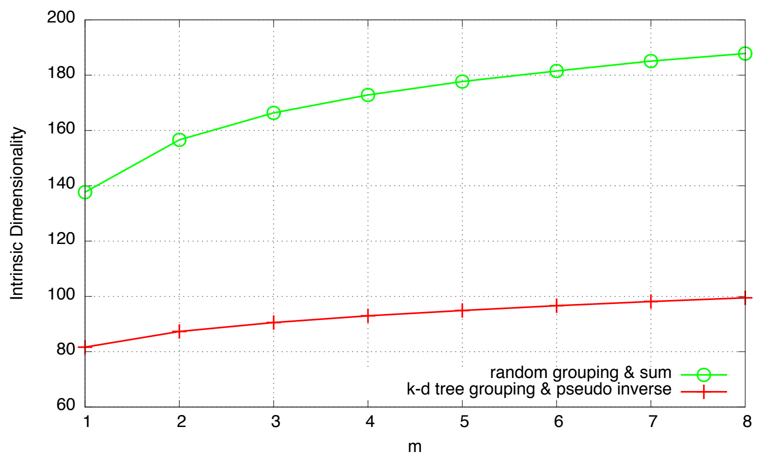


Figure 4.3: Intrinsic dimensionality varying m. Random and sum vs. k-d tree and pseudo-inverse group creation methods.

	m=3		m=4		m=8	
	Time (s)	mAP	Time (s)	mAP	Time (s)	mAP
Random G.T. [SFJ14]	187	0.507	252	0.512	483	0.518
sGT [this method]	180	0.505	164	0.504	172	0.500

Table 4.1: Comparing the search time of different group construction methods for group testing.

that is the most competitive method when trying to identify the best representatives. Digging into the logs of FLANN shows that it is again the dimensionality of the group representatives which can explain this remarkable difference in behavior. While both k-d tree based representatives and random representatives have the same representational dimensionality (i.e. d =1,920), they have however a quite different intrinsic dimensionality.

With $m = 3$, k-d tree based group representatives have an estimated local intrinsic dimensionality of 90.6$\pm$29.5 while random based group representatives have an intrinsic dimensionality of 166.4$\pm$13.5. It is therefore much harder to index random based group vectors, and resorting on the linear scan is possibly the best strategy.

Indeed, blending randomly the features increases the entropy, explaining why the resulting intrinsic dimensionality is even higher than the one of the raw features. Such random group vectors are distributed all over the high-dimensional space, making it quite hard to distinguish the near neighbors from the most distant ones, as clearly stated in the seminal paper about neighbor meaningfulness [BGRS99]. Similar observations can be made for other values of m, as it is reported in Figure 4.3.

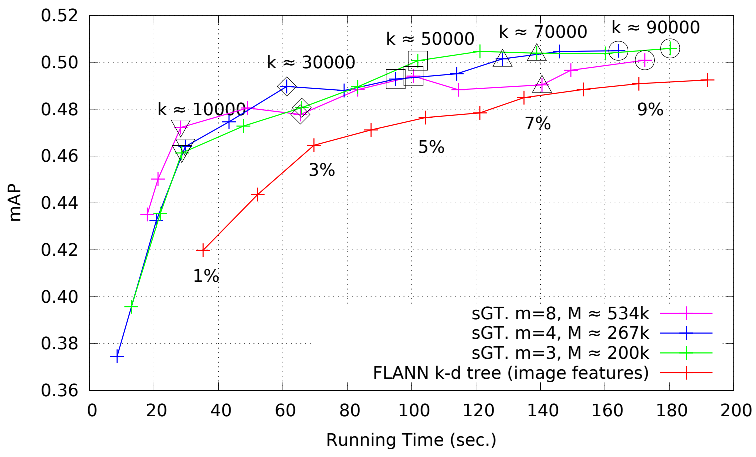


Figure 4.4: Response times vs. mAP. FLANN k-d tree, several configuration for sGT.

4.1.3.6 Overall Performance: Comparing Quality and Run Time

The last experiment compares the overall performance of sGT with (i) the performance of the k-d tree created by FLANN over all of the image features, as described in Section 4.1.3.2 and (ii) with the state-of-the-art group testing scheme by Shi et al. [SFJ14].

To measure the performance of the k-d tree of FLANN, we ask the library to return the top 1%, 2%, . . . , 10% image features that are the most similar to the queries. This obviously impacts the response time as well as the resulting mAP. This is depicted by Figure 4.4 where a line that gives the performance of the FLANN index is plotted. This line shows that when an increasing number of features are considered, then the response time increases as does the mAP, as expected. It is possible to know the percentages of the most similar features that are considered at query time as it is specified along the line. For clarity, not all such percentages are indicated; the missing ones can easily be deduced.

The other series of lines display the performance of sGT. The parameter of utmost importance in this strategy is the redundancy, m, which we vary here. We therefore created 3 configurations of the groups for three specific values of that m parameter that are $m = 3$, $m = 4$ and $m = 8$. The Figure 4.4 therefore contains three other lines, each depicting the behavior of one group testing configuration for sGT.

To better understand the behavior of sGT, we have changed for each configuration the number of representatives that are identified to subsequently evaluate equations (2.22) and (2.23). One of the claims we make in this section is that indexing the representatives is beneficial to performance because returning a particular number of them (k) among a total of M existing representatives is mostly independent of that M value while it boosts performance. We therefore instructed sGT to return the best k group representatives (with k ranging from 2,560 to 90,000) before identifying the R =50,000 best image features. We visualise along the sGT lines in Figure 4.4 some of these values for k. We explicitly give few such values to preserve the clarity of this figure. For example, the response times and the corresponding mAP obtained with the three configurations of sGT when $k \approx$10,000 are all three represented on the lines using a triangle pointing downward symbol. Similarly, the operating points are represented with the square symbol for $k \approx$50,000.

Using these symbols facilitates observing that indeed, the response times of sGT is mostly independent of M (which increases as m grows). When $m = 3$, then there are overall M =200,298 group vector representatives, while there are M =534,128 when $m = 8$. Despite these different values for M, retrieving the same number k of representatives takes roughly the same amount of time, between 160 and 180 seconds when $k \approx$90,000, see the circles on the figure. Furthermore, the order of these three circles on the time-line is not ruled by m, materializing the independence between M and the response time.

This figure also shows that the mAP improves as m increases, while the response time stays quasi identical thanks to having indexed the group representatives.

Finally, it is interesting to compare the response times and the mAP that Shi et al. [SFJ14] could achieve using the same parameters. Table 4.1 compares three configurations of the original group testing technique (based on random grouping and the use of the sum (2.21) to compute the representatives) to the same three configurations for sGT that we have described. Please note, however, that [SFJ14] uses all M representatives when evaluating equations (2.22) and (2.23) while we use only the $k \approx$90,000 best ones. The table shows that sGT is much faster than the original group testing proposal, while the quality does not differ significantly.

4.2 Memory Vectors for Identification with Privacy

This section considers a typical scenario involving the following three entities. The owner $\mathcal{O}$ has a collection of N objects. The user $\mathbf{U}$ would like to know whether there is an object in this collection similar enough to query object, and in that case, which object it is. The owner is not willing to operate the identification and outsources this task to a server $\mathbf{S}$. The targeted applications are typically related to multimedia retrieval, medical diagnosis, biometrics. In the later case, the owner $\mathcal{O}$ is a personification of the enrollment phase. A feature vector is extracted from the objects (iris, face captures) and used as a proxy: similar objects share similar features. Overall, this boils down to managing a database of vectors probed with query vectors.

We add the following privacy / security requirements:

- $\mathbf{U}$ doesn't want to reveal his query,
- $\mathcal{O}$ is reluctant in disclosing his database.

In other words, $\mathbf{S}$, the outsourced server is not trusted. This actor is deemed as semi-honest (or honest but curious): it operates the search task, however it tries to grab information about the query or the database. This would allow $\mathbf{S}$ to profile $\mathbf{U}$, i.e. to disclose what $\mathbf{U}$ is interested in, or to perform the search with unauthorized users, i.e. without the agreement of $\mathcal{O}$. In biometrics application, disclosing database vectors could lead to spoofing [HEMF15]. Some make the distinction between the privacy, which protects user **U**, and the security, which protect the data of the owner $\mathcal{O}$.

Section 4.2.2 describes a classical solution based on embedding and secure multi-party computation. We selected the LSH embedding which maps real vectors to binary hashes. Evaluating the cosine similarity between vectors amounts to computing the Hamming distance between their hashes. We use a very efficient cryptographic protocol called SHADE for securing Hamming distances computations[2]: while **S** learns nothing from the query, **U** learns nothing from the database vector except the index of the most similar vector. This system enables privacy under the semi-honest model, but the security of owner's data is weak if **U** and **S** collude.

This weakness is due to **S** having binary hashes in the clear. The main countermeasure in literature is fully homomorphic encryption. It allows computing distances between two ciphers s.t. **S** now manipulates data previously encrypted by $\mathcal{O}$. However, computation in the encrypted domain is very low preventing the scalability of the system.

This method trades privacy of the user and security of the data for scale of the database, speed of similarity search and quality of the identification. This is achieved by resorting to advanced signal processing rather than more cryptography. Section 4.2.3 describes a second version of the system adding on top of LSH and SHADE a group testing scheme for dealing with efficiency of the search while leaking only very little useful information.

In order to understand the basic concepts about LSH, we encourage the reader to read Section 2.2.2 before reading the following. This work has been published [Iscen and Furon, 2016].

4.2.1 Setup

The owner $\mathcal{O}$ has a database of vectors $\mathbf{X} := \{\mathbf{x}_i\}_1^N$ s.t. $\mathbf{x}_i \in \mathbb{R}^d$ with Euclidean norm $\|\mathbf{x}_i\| = 1, \forall i \in [N]$, where $[N] := \{1, \cdots, N\}$. We denote the query of the user by $\mathbf{q} \in \mathbb{R}^d$ with $\|\mathbf{q}\| = 1$. The similarity between the query and a database vector $\mathbf{x}$ is defined by $s(\mathbf{q}, \mathbf{x}) := \mathbf{q}^\top \mathbf{x}$.

In this section, we analyse the proposed system having in mind biometrics identification. Database vectors are biometrics recorded at the enrollment phase. **U** is interested in knowing the index of the most similar vector in the database if similar enough:

$$\hat{\imath} := \begin{cases} \arg\max_{i\in[N]} s(\mathbf{q}, \mathbf{x}_i) & \text{if } s(\mathbf{q}, \mathbf{x}_{\hat{\imath}}) \geq \rho \\ \emptyset & \text{otherwise} \end{cases} \tag{4.3}$$

There are two cases: The query is a noisy version of an enrolled vector $\mathbf{x}_{i^\star}$; or the query is random and we denote $i^\star = \emptyset$. For a given threshold ρ, three errors are possible:

- False negative: $\hat{\imath} = \emptyset$ while $i^\star \neq \emptyset$

[2]Another option is partial homomorphic encryption

- False identification: $\hat{\imath} \neq i^\star$ while $i^\star \neq \emptyset$
- False positive: $\hat{\imath} \neq \emptyset$ while $i^\star = \emptyset$

We denote by $\mathbb{P}_{\text{fn}}$, $\mathbb{P}_{\text{fid}}$, and $\mathbb{P}_{\text{fp}}$ respectively the probabilities of these events.

The 'real' search defined in (4.3) produces non zero error probabilities depending on how strongly the query is correlated with $\mathbf{x}_{i^\star}$ (when $i^\star \neq \emptyset$) and on the size N of the database. Due to privacy, $\mathbf{S}$ can not perform the real search as defined above. In agreement with $\mathcal{O}$, $\mathbf{S}$ runs a secure and approximate search with lower performances.

4.2.1.1 SHADE Protocol

Suppose now that $\mathbf{U}$ and $\mathbf{S}$ have respectively the binary words $\mathbf{w}^{\mathbf{U}}$ and $\mathbf{w}^{\mathbf{S}}$, and that $d_H(\mathbf{w}^{\mathbf{U}}, \mathbf{w}^{\mathbf{S}})$ is needed. SHADE is an efficient protocol [BCP13] allowing $\mathbf{U}$ to learn nothing about $\mathbf{w}^{\mathbf{S}}$ except $d_H(\mathbf{w}^{\mathbf{U}}, \mathbf{w}^{\mathbf{S}})$, while $\mathbf{S}$ gains no information about $\mathbf{w}^{\mathbf{U}}$. In short, at the k-th round, $\mathbf{S}$ creates two messages: $m_0^{(k)} = \mathbf{w}_k^{\mathbf{S}} + r_k$ and $m_1^{(k)} = (\mathbf{w}_k^{\mathbf{S}} \bigoplus 1) + r_k$, where r_k is an alea uniformly distributed over $\mathbb{Z}_{D+1}$. Thanks to an oblivious transfer, $\mathbf{U}$ receives $v_k := m^{(k)}_{\mathbf{w}_k^{\mathbf{U}}}$. After D rounds, $\mathbf{S}$ sends $R := \sum_{k=1}^{D} r_k$ to $\mathbf{U}$ who computes $V := \sum_{k=1}^{D} v_k$ and $V - R = d_H(\mathbf{w}^{\mathbf{U}}, \mathbf{w}^{\mathbf{S}})$. SHADE is secure in the static semi-honest model: $\mathbf{U}$ and $\mathbf{S}$ are honest but curious. There is an efficient version of SHADE for computing a batch of Hamming distances between one query and N vectors [BCF$^+$14].

4.2.1.2 Adding Comparison and Minimum

Letting $\mathbf{U}$ knowing the Hamming distance $d_H(\mathbf{w}^{\mathbf{U}}, \mathbf{w}^{\mathbf{S}})$ is dangerous in the dynamic model where $\mathbf{U}$ is allowed to perform several distance computations: With D well chosen queries, $\mathbf{U}$ can disclose $\mathbf{w}^{\mathbf{S}}$. The authors of SHADE recommend the use of the GMW protocol [SZ13] to first securely compare $V - R$ to a threshold τ. $\mathbf{U}$ only gets $\mathsf{sign}(d_H(\mathbf{w}^{\mathbf{U}}, \mathbf{w}^{\mathbf{S}}) - \tau)$. For a batch of distances, their minimum is securely computed and $\mathbf{U}$ only learns $\hat{\imath} = \arg\min d_H(\mathbf{w}^{\mathbf{U}}, \mathbf{w}^{\mathbf{S}})$.

4.2.2 First Version of the System

A system is a list of procedures followed by the three actors $\mathcal{O}$, $\mathbf{S}$ and $\mathbf{U}$. The owner $\mathcal{O}$ draws D random directions stacked in matrix $\mathbf{A} := [\mathbf{u}_1, \cdots, \mathbf{u}_D]$, computes the embeddings of its vectors, and sends $\mathbf{S}$ the database $\mathbb{E} := \{e(\mathbf{x}_i)\}$. $\mathcal{O}$ grants $\mathbf{U}$ the access of the identification by sending $\mathbf{A}$ so that $\mathbf{U}$ can compute the embedding of its query $\mathbf{q}$.

At query time, entities $\mathbf{U}$ and $\mathbf{S}$ perform the SHADE protocol, where $\mathbf{w}^{\mathbf{U}} = e(\mathbf{q})$ and $\mathbf{w}^{\mathbf{S}} = e(\mathbf{x}_i)$ with the batch option. $\mathbf{U}$ maps the Hamming distances $d_H(e(\mathbf{x}_i), e(\mathbf{q}))$ to similarity estimates $\hat{s}(\mathbf{q}, \mathbf{x}_i)$ thanks to (2.19). $\mathbf{U}$ finds the index with the biggest similarity and takes it as the output if it is above the threshold ρ. The search is approximate because it is based on similarity estimates.

An option is to add GMV protocol to let $\mathbf{U}$ learn either indices of vectors whose approximated similarity with the query is above a threshold (secure comparison), or the index of the vector whose approximated similarity is maximum (secure maximum). Note that this is done on the Hamming distances because (2.19) is a monotonic mapping. A combination of the two implement the search as defined in (4.3).

4.2.2.1 Analysis

Complexity: The quality of the approximate search depends on D, the embedding length. For any two vectors $\mathbf{x}$ and $\mathbf{q}$ s.t. $\mathbf{q}^\top\mathbf{x} = \cos(\theta)$, and $\mathbf{A}$ randomly generated as early described, $\frac{\pi}{D} d_H(e(\mathbf{q}), e(\mathbf{x}))$ in (2.19) is indeed an estimation of θ with no bias and variance $\theta(\pi - \theta)/D$. However, the complexity of the SHADE protocol deeply depends on D. The secure computation of a batch of size N has a complexity [BCF$^+$14]:

$$\mathcal{C} \propto [(2ND\log_2(D))/o + 3D]\mathcal{C}_{sym}, \tag{4.4}$$

where o is a constant setting the security of the protocol (at least 128), and $\mathcal{C}_{sym}$ is the complexity of one symmetric cryptography operation (i.e. AES encryption / decryption).

Even though SHADE has been a breakthrough considerably lowering the complexity of the secure computation of Hamming distances, it is the bottleneck of our system. It takes 0.2s to securely compute and run the GMV protocol over $N = 200$ embeddings of length $D = 900$ (see [BCF$^+$14]).

Security: SHADE enables the privacy of $\mathbf{U}$, but the owner $\mathcal{O}$ has some concerns. The parameter of the embedding (here matrix $\mathbf{A}$) is generated by the owner $\mathcal{O}$ and only shared with $\mathbf{U}$. This parameter is the only secret that prevents a curious server $\mathbf{S}$ from performing illegal identification or estimating the database vectors by inverting the embedding. By colluding with one user, $\mathbf{S}$ learns this secret.

Since (2.17) is a surjection, the embedding is not reversible and perfect reconstruction of $\mathbf{X}$ is impossible. However, one can see the embedding as a quantizer producing quantification noise. Sect. 4.2.2.2 measures the accuracy of a reconstruction of $\mathbf{x}$ from $e(\mathbf{x})$ provided matrix $\mathbf{A}$ is known.

4.2.2.2 Inverting LSH

A simple reconstruction of a unit norm vector $\mathbf{x}$ from its embedding is:

$$\hat{\mathbf{x}} = \kappa(\mathbf{A}(2e(\mathbf{x}) - \mathbf{1}_D)), \tag{4.5}$$

where κ is s.t. $\mathbb{E}_{\mathbf{A}}[\|\hat{\mathbf{x}}\|^2] = 1$. To quantify its accuracy, we introduce $a(\mathbf{x}) := \mathbf{x}^\top\hat{\mathbf{x}}$. The closer to 1, the better the reconstruction. We have:

$$a(\mathbf{x}) = \kappa \sum_{k=1}^{D} |\mathbf{u}_k^\top \mathbf{x}|. \tag{4.6}$$

LSH was originally proposed with $\{\mathbf{u}_k\}$ being independent random directions in $\mathbb{R}^d$. In this case, Section 4.2.6 shows:

$$\mathbb{E}_{\mathbf{A}}[a(\mathbf{x})] \approx \sqrt{\frac{g}{1+g}} \text{ with } g := \frac{2D}{\pi d}. \tag{4.7}$$

However, if $D \leq d$, it is known that $\mathbf{A}$ being a random basis of rank D yields a better search [WTF09]. In this case, $\|\mathbf{A}(2e(\mathbf{x}) - \mathbf{1}_D)\|^2 = D$ for any $\mathbf{x}$. Section 4.2.6 shows:

$$\mathbb{E}_{\mathbf{A}}[a(\mathbf{x})] \approx \sqrt{g} \text{ with } g \leq 1. \tag{4.8}$$

If $D > d$, a good choice is $\mathbf{A}$ as a random tight frame [SVZ13] s.t. $\|\mathbf{A}(2e(\mathbf{x}) - \mathbf{1}_D)\|^2 = d/D\|2e(\mathbf{x}) - \mathbf{1}_D)\|^2 = d$. Yet, the r.v. $\mathbf{u}_k^\top \mathbf{x}$ are no longer independent and it is hard to say something more than:

$$\sqrt{\frac{2}{\pi}} \approx 0.80 \leq \mathbb{E}_{\mathbf{A}}[a(\mathbf{x})] \leq 1. \tag{4.9}$$

Approximate search based on LSH embedding usually sets $D > d$. On expectation, $\hat{\mathbf{x}}$ is then a good approximation of $\mathbf{x}$ since they correlate more than 0.8.

Fig. 4.8 shows the reconstruction accuracy w.r.t. D, when $\mathbf{A}$ is a random tight frame or random Gaussian matrix.

4.2.2.3 Lessons Learnt from the First System

The parameter of utmost importance is the length of the embedding D. It sets a trade-off between the quality of the identification, the complexity of the protocol, and the security when $\mathbf{A}$ has been compromised. Yet, this trade-off is poor as the reconstruction provides a good accuracy when $D > d$.

4.2.3 The Proposed System

Our aim is to provide a second line of defense. Group testing was recently introduced in approximate search. The idea is to pack database vectors into groups and to compute a representation per group, so-called memory vector, which allows to perform a group test. This test reveals whether the query is similar to at least one of the vectors in the group. For large dimension d, more vectors can be packed into one group because the correlation between two independent vectors with unit norm concentrates around 0.

Each database vector belongs to several groups. The decoding aims at identifying the matching vector(s) from the results of the group tests. A matching vector is defined as having a similarity with the query high enough: $s(\mathbf{q}, \mathbf{x}) \geq \alpha$. Naively, matching vectors are lying at the intersection of the groups yielding a positive group test. Things are not that simple because these tests suffer from false positives and false negatives. Yet, if the number of groups is large enough, the decoding succeeds in identifying matching vectors.

This approach brings two advantages to our system:

- The number of groups M is smaller than the number of vectors N. This lowers both the storage space and the complexity of the protocol by a gain $\gamma := M/N < 1$.
- $\mathcal{O}$ will not give $\mathbf{S}$ embeddings of database vectors, but of memory vectors. This makes harder the reconstruction of database vectors.

4.2.3.1 Encoding

The owner $\mathcal{O}$ randomly packs the N database vectors into M groups $\{\mathcal{G}_j\}_{j=1}^M$ s.t. i) any group comprises n vectors, ii) any vector belongs to m groups. This rules enforces that $M = mN/n$. A possible construction is explained in [IFG$^+$17]. We call the map the $M \times N$ binary matrix $\mathbf{B}$ indicating which vector belongs to which group: $b_{i,j} = 1$ if $\mathbf{x}_i$ belongs to group $\mathcal{G}_j$, 0 otherwise.

Then, $\mathcal{O}$ computes the memory vectors, i.e. the representatives of each group. We adopt here the most simple constructions investigated in [IFG$^+$17]:

$$\mathbf{m}_j = \sum_{i:b_{i,j}=1} \mathbf{x}_i. \tag{4.10}$$

Finally, $\mathcal{O}$ computes the embeddings of the memory vectors and sends $\mathbf{S}$ the following compact description $\mathbb{E}' = \{e(\mathbf{m}_j)\}_{j=1}^M$. $\mathcal{O}$ sends $\mathbf{U}$ parameters $(\mathbf{A}, \mathbf{B}, \{\|\mathbf{m}_j\|\}_{j=1}^M)$: $\mathbf{A}$ is needed to compute query embedding, $\mathbf{B}$ and $\{\|\mathbf{m}_j\|\}_{j=1}^M$ to decode the tests. Note that the database $\mathbb{E}'$ is smaller than $\mathbb{E}$ since it has $M < N$ entries.

4.2.3.2 Querying

$\mathbf{U}$ computes $e(\mathbf{q})$ thanks to $\mathbf{A}$ and runs the SHADE protocol with $\mathbf{S}$, who learns nothing about the query. $\mathbf{U}$ obtains estimations of the cosine between $\mathbf{q}$ and $\mathbf{m}_j$. Multiplying by the norm $\|\mathbf{m}_j\|$, this gives estimated similarities $\hat{s}_j \approx \mathbf{q}^\top \mathbf{m}_j$.

The big benefit is the decrease of SHADE complexity, which was the bottleneck of the previous system. Instead of securely computing N Hamming distances $d_H(e(\mathbf{q}), e(\mathbf{x}_i))$, we compute $M < N$ distances $d_H(e(\mathbf{q}), e(\mathbf{m}_j))$.

Soft decoding: From $\{\hat{s}_j\}_{j=1}^M$ and map $\mathbf{B}$, $\mathbf{U}$ runs the decoding to identify $i^\star$, index of the matching vector.

In summary, the decoding computes N scores. c_i is the likelihood ratio w.r.t. two hypothesis: $\mathbf{x}_i$ is the matching vector ($\mathcal{H}_i$) or not ($\bar{\mathcal{H}}_i$).

$$c_i = \sum_{j=1}^{M} \log \frac{f_{\mathcal{H}_i}(\hat{s}_j, b_{i,j})}{f_{\bar{\mathcal{H}}_i}(\hat{s}_j, b_{i,j})}, \tag{4.11}$$

where the pdfs are modeled as mixtures of two Gaussian distributions, $\eta\mathcal{N}(\alpha; (n-1)/d) + (1-\eta)\mathcal{N}(0; n/d)$, with parameter η given in Table 4.2. We refer the reader to [IFG$^+$17] for justifications of this statistical model. Then, $\mathbf{U}$ computes the maximum of these scores and compares with the threshold.

Table 4.2: Definition of the weights

	$b_{i,j} = 1$	$b_{i,j} = 0$
$\mathcal{H}_i$	$\eta = 1$	$\eta = 0$
$\bar{\mathcal{H}}_i$	$\eta = (n-1)/N$	$\eta = n/N$

Hard decoding: To prevent oracle attacks from $\mathbf{U}$, we use the GMW protocol to apply a secure comparison: $\mathbf{U}$ learns nothing more than $d_j = \mathsf{sign}(\hat{s}_j - \tau)$, where threshold τ has been carefully selected by $\mathcal{O}$. Then, the decoding computes the following scores:

$$c_i = \sum_{j=1}^{M} d_j \log \frac{p_{\mathcal{H}_i}(b_{i,j})}{p_{\bar{\mathcal{H}}_i}(b_{i,j})} + (1 - d_j) \log \frac{1 - p_{\mathcal{H}_i}(b_{i,j})}{1 - p_{\bar{\mathcal{H}}_i}(b_{i,j})}, \tag{4.12}$$

with $p_{\mathcal{H}_i}(b_{i,j}) = \mathbb{E}_{\hat{s}_j \sim f_{\mathcal{H}_i}(\hat{s}_j, b_{i,j})}[\hat{s}_j > \tau]$:

$$p_{\mathcal{H}_i}(b_{i,j}) = \eta\Phi\left((\tau - \alpha)\sqrt{\frac{d}{n-1}}\right) + (1-\eta)\Phi\left(\tau\sqrt{\frac{d}{n}}\right)$$

and parameter η given in Table 4.2.

4.2.4 Security

Unauthorized identification is possible whenever an untrusted actor, be it $\mathbf{U}$ and/or $\mathbf{S}$, has in his hands both $(\mathbf{A}, \mathbf{B}, \{\|\mathbf{m}_j\|\}_{j=1}^{M})$ and $\mathbb{E}'$. This happens only when $\mathbf{S}$ and $\mathbf{U}$ colluded, or if $\mathbf{U}$ succeeds to steal $\mathbb{E}'$.

The group testing approach brings a second line of defense by mixing the database vectors into memory vectors. We focus here on the reconstruction of the database vectors. To do so, the untrusted actor must i) reconstruct the memory vectors by 'inverting' their embeddings, and ii) estimate database vectors from the reconstructed memory vectors of groups they belong to. Reconstruction i) needs $\mathbb{E}'$, $\mathbf{A}$ and $\{\|\mathbf{m}_j\|\}$. The quality of the reconstruction is quite high as shown in Sec. 4.2.2.2. Estimation ii) needs $\mathbf{B}$. The quality of the reconstruction is investigated in the next section.

4.2.4.1 Inverting Memory Units

Equation (4.10) can be rephrased as $\mathbf{M} = \mathbf{X}\mathbf{B}^\top$ where $\mathbf{M} := [\mathbf{m}_1, \cdots, \mathbf{m}_M]$ is a $d \times M$ matrix storing the memory units while $\mathbf{X} := [\mathbf{x}_1, \cdots, \mathbf{x}_N]$ stores the database vectors and $\mathbf{B}$ is the map (See Sect. 4.2.3.1). Estimating back $\mathbf{X}$ from $\mathbf{M}$ is possible using a ridge regression or the pseudo-inverse of $\mathbf{B}^\top$: $\hat{\mathbf{X}} \propto \mathbf{M}(\mathbf{B}^\top)^\dagger$. We can show that reconstructing $\mathbf{x}$ from the exact memory units, produces an estimation $\hat{\mathbf{x}}$ s.t.

$$\mathbb{E}[a(\mathbf{x})] = \min(1, \sqrt{\gamma}). \tag{4.13}$$

Yet, this requires the inverse of large $M \times M$ matrix $\mathbf{B}\mathbf{B}^\top$. The average of some memory vectors albeit not optimal is faster: $\hat{\mathbf{X}} \propto \mathbf{M}\mathbf{B}$ achieving ($\nu := n/N$):

$$\mathbb{E}[a(\mathbf{x})] \approx \sqrt{\frac{\gamma}{1 + \nu^2(M-1) + \gamma(1-\nu)^2}}. \tag{4.14}$$

4.2.4.2 Full Reconstruction

The attacker first reconstructs the group vectors $\{\hat{\mathbf{m}}_k\}$ from their embeddings $\{e(\mathbf{m}_k)\}$, and then reconstructs the database vectors $\{\hat{\mathbf{x}}_i\}$. It is easy to show that if the first step produces a reconstruction accuracy measured by $\mathbb{E}[a(\mathbf{m})] = a_1$ while the second step achieves $\mathbb{E}[a(\mathbf{x})] = a_2$ starting from true memory vectors, then the total reconstruction yields $\mathbb{E}[a(\mathbf{x})] = a_1 a_2$ thanks to the linearity of the second step. This is evidenced in the experimental work.

4.2.5 Experiments

4.2.5.1 Experimental Setup

We test our system using both synthetic and real data. For both cases, we keep the ratio $\gamma = M/N = m/n = 0.1$. This means that the retrieval system needs only 0.1 of memory storage and vector comparisons compared to exhaustive search. The embedding is parametrized s.t. $D = 2d$.

Synthetic Data. We create a synthetic dataset of N vectors distributed randomly on the unit hypersphere of $\mathbb{R}^d$. We then create N_q random query vectors, such that each query has exactly one match in the dataset, $\mathbf{x}_{i^\star}$, and the similarity between the query and the matching vector is $\mathbf{q}^\top \mathbf{x}_{i^\star} = \alpha$. We set $d = 1,920$, $N = 10,000$, $N_q = 100$, and $\alpha = 0.5$.

Real Data. We also use Labeled Faces in the Wild (LFW) dataset [HRBLM07], which has 13,233 images of 5,749 people. For the query set, we choose a random image from people who have i) at least two images , ii) have at least one match with a similarity greater than or equal to α. We use the full ($d = 67,584$) Fisher face descriptors [SPVZ13] to calculate the similarities to generate the query set. Then, we use the same query set for all experiments, regardless of different descriptors or dimensionality. Setting $\alpha = 0.5$ gives us $N_q = 104$ queries, each belonging to a different person. We also choose $1,000$ random queries from people who have no other matching vectors. All queries and random queries are removed from the dataset.

For our experiments, we reduce the dimension of Fisher face descriptors from $67,584$ to $1,920$ for fair comparison with synthetic data. We also use CNN-based descriptors [BSJ16], whose dimension is reduced from $4,096$ to $1,920$.

Parameters setting. As stated earlier, we can set the ratio of $M/N = 0.1$ with different m and n combinations. We choose the optimal setting empirically by maximizing the Kullback-Leibler distance between negative and positive distributions. This gives $n = 200$, $m = 20$, and $\tau = 0.8 \times \alpha$.

Evaluation. The three metrics for evaluating the performance are the probabilities of the three types of error, $\mathbb{P}_{\text{fp}}$, $\mathbb{P}_{\text{fn}}$ and $\mathbb{P}_{\text{fid}}$ as functions of threshold ρ (see Sect. 4.2.1). Security is gauged by the quality of the reconstruction of the database vectors measured by $\mathbb{E}[a(\mathbf{x})]$.

4.2.5.2 Approximate Search

Fig. 4.5 shows the performance of the identification of the baseline, i.e. the exhaustive search on real vectors, without any privacy and security issues, as defined in (4.3). Fig. 4.6 shows the same evaluation for the first system described in Sect. 4.2.2. Fig. 4.7 shows the system proposed in Sect. 4.2.3 with the hard decoding variant (4.12). We are smoothly degrading the performance of the approximate search over synthetic data, however it still performs well on the real dataset.

4.2.5.3 Security

Fig. 4.8 shows the reconstruction performance while inverting LSH with the synthetic dataset. It encompasses two matrix $\mathbf{A}$ generation procedures: Gaussian i.i.d. entries and uniform tight frame. The latter option is known to produce better approximate search. This illustrates the security of the first system described in Sect. 4.2.2. In our setup where $D = 2d$, we end up with $\mathbb{E}[a(\mathbf{x})] \approx 0.88$.

Fig. 4.8 shows the reconstruction performance while inverting the memory units with $m \in [2, 20]$ and $n = 200$ on the synthetic dataset. It encompasses two reconstruction methods: the pseudo inverse of $\mathbf{B}$ and the average of memory vectors (See Sect. 4.2.4.1). This illustrates the security improvement thanks to the second line of defense provided by group testing. In our setup where $n = 200$ and $m = 20$, we end up with $\mathbb{E}[a(\mathbf{x})] \approx 0.33$.

Overall, the reconstruction of vectors $\{\mathbf{x}_i\}$ from $\{e(\mathbf{m}_k)\}$ outputs estimates correlated on expactation $\mathbb{E}[a(\mathbf{x})] \approx 0.88 \times 0.33 = 0.29$. We run the attack on the real datasets and get $\mathbb{E}[a(\mathbf{x})] \approx 0.88 \times 0.31 = 0.27$.

4.2.6 Reconstruction from LSH

Assuming that $\mathbf{A}$ has an isotropic distribution (either independent Gaussian entries or a random frame), w.l.o.g. we set $\mathbf{x} = (1, 0, \ldots, 0)$ and

$$\|\hat{\mathbf{x}}\|^2 = \kappa^2 \left(\sum_{k=1}^{D} \|\mathbf{u}_k\|^2 + \sum_{k \neq \ell} |a_k(1)||a_\ell(1)| \right) \tag{4.15}$$

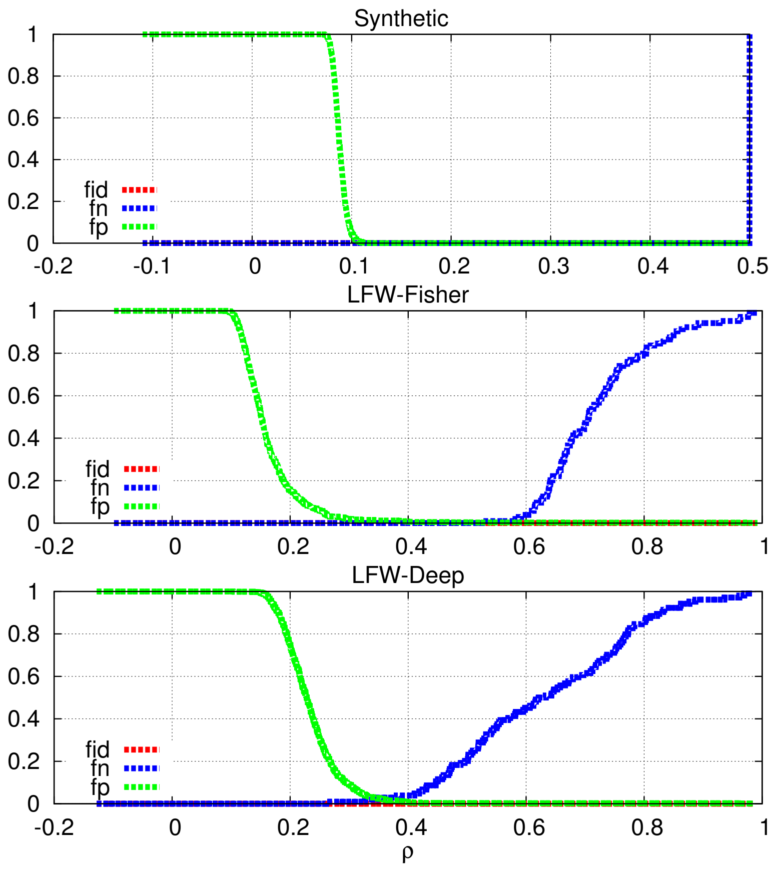


Figure 4.5: Baseline performance, synthetic and LFW (Fisher / deep learning features) datasets.

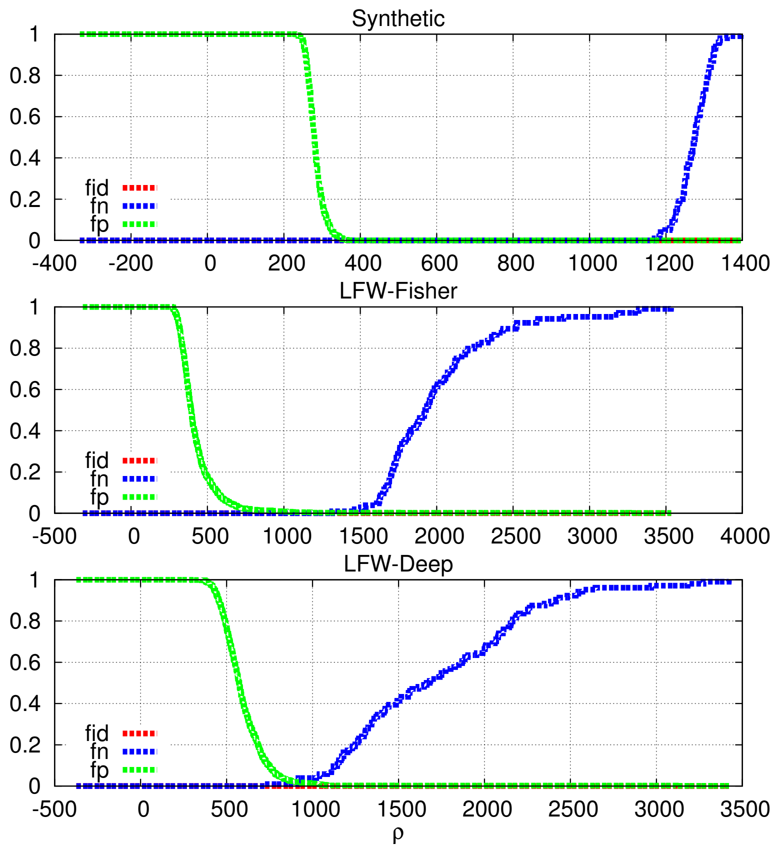


Figure 4.6: 1st system performance, synthetic and LFW (Fisher / deep learning features) datasets.

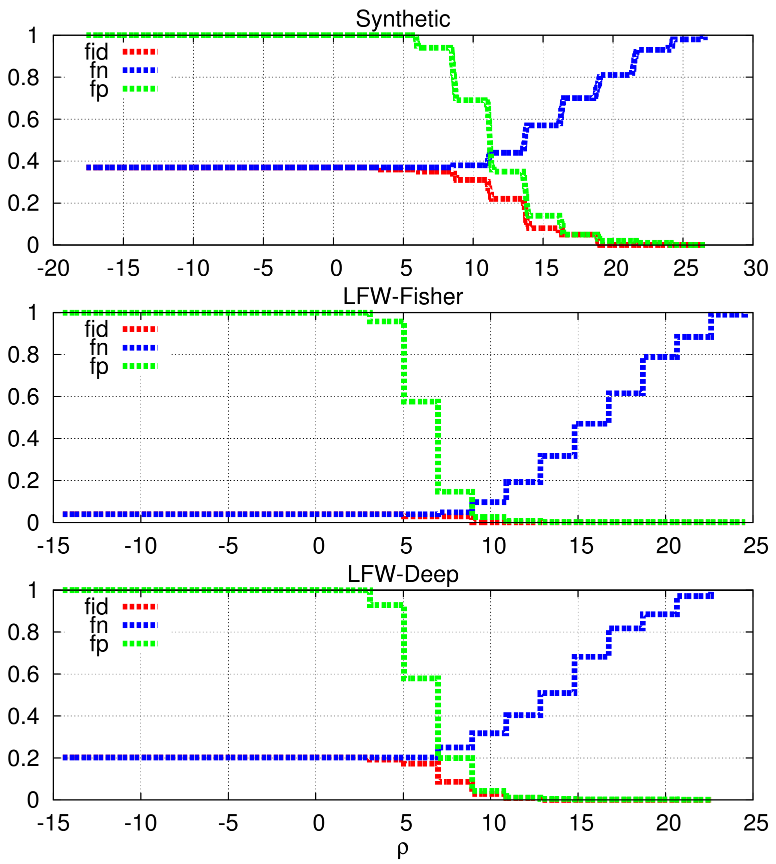


Figure 4.7: Proposed system performance, synthetic and LFW (Fisher / deep learning features) datasets.

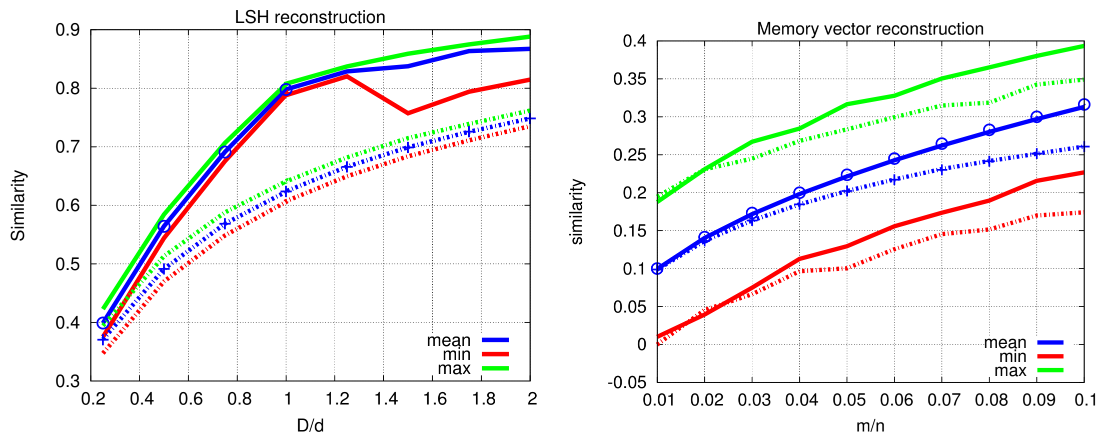


Figure 4.8: **Left:**Reconstruction from LSH: $a(\mathbf{x})$ as a function of D/d. Empirical mean, min. and max. over 10,000 reconstructions. Unif. Tight Frame (plain), Gaussian i.i.d. (dashed), Eq. (4.7) (+) and Eq. (4.8) (o). **Right:** Reconstruction from memory vectors: $a(\mathbf{x})$ as a function of m/n. Empirical mean, min., and max. over 40,000 reconstructions. Pseudo-inverse (plain), average (dashed), Eq. (4.13) (o) and (4.14) (+).

If $a_k(1) \sim \mathcal{N}(0,1)$, $\mathbb{E}_{\mathbf{A}}[\|\hat{\mathbf{x}}\|^2] = \kappa^2 D(d + 2(D-1)/\pi)$ and

$$\kappa \approx \sqrt{\frac{1}{Dd(1+g)}} \text{ with } g := \frac{2D}{\pi d}. \tag{4.16}$$

For this $\mathbf{x}$, $a(\mathbf{x}) = \kappa \sum_{k=1}^{D} |a_k(1)|$ s.t.

$$\mathbb{E}_{\mathbf{A}}[a(\mathbf{x})] = \kappa D \sqrt{2/\pi} = \sqrt{\frac{g}{1+g}}. \tag{4.17}$$

If $\mathbf{u}_k$ is uniformly distributed on the unit sphere (random uniform frame or basis), the marginal distribution of $a_k(1)$:

$$f(s) = \frac{(1-s^2)^{\frac{d-3}{2}}}{B(1/2,(d-1)/2)}, \forall s, -1 \leq s \leq 1, \tag{4.18}$$

s.t. $\mathbb{E}[|a_k(1)|] = \sqrt{2/d\pi}(d-2)!!/(d-1)!! \approx \sqrt{2/d\pi}$. For $D \leq d$, the random uniform tight frame is indeed a basis: $\mathbb{E}_{\mathbf{A}}[\|\hat{\mathbf{x}}\|^2] = D$ and $\mathbb{E}_{\mathbf{A}}[a(\mathbf{x})] = \sqrt{g}$.

4.2.7 Reconstruction from Memory

We assume the following model for matrix $\mathbf{B}$. For row r, $1 \leq r \leq M$, we randomly select n indices in $[N]$ and set these coefficients to 1. With a high probability $\mathsf{rank}(\mathbf{B}) = M$ and $\mathrm{Tr}(\mathbf{B}\mathbf{B}^\top) = nM$. As for the vector, $\mathbb{E}_{\mathbf{X}}[\mathbf{X}^\top \mathbf{X}] = \mathbf{I}_N$.

Pseudo-inverse reconstruction: $\hat{\mathbf{X}} = \eta \mathbf{M}(\mathbf{B}^\top)^\dagger$, with $(\mathbf{B}^\top)^\dagger = (\mathbf{B}\mathbf{B}^\top)^{-1}\mathbf{B}$. Constant η is s.t. the expectation (over $\mathbf{X}$ and $\mathbf{B}$) of the average squared norm of $\hat{\mathbf{x}}_i$ equals 1.

$$\begin{aligned}\frac{\mathbb{E}[\sum_{i=1}^{N} \|\hat{\mathbf{x}}_i\|^2]}{N} &= \mathbb{E}[\mathrm{Tr}(\hat{\mathbf{X}}^\top \hat{\mathbf{X}})]/N \qquad (4.19)\\ &= \eta^2 \mathbb{E}[\mathrm{Tr}(\mathbf{X}^\top \mathbf{X}\mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top)^{-1}\mathbf{B})]/N\\ &= \eta^2 \mathbb{E}[\mathrm{Tr}((\mathbf{B}\mathbf{B}^\top)^{-1}\mathbf{B}\mathbf{B}^\top)]/N = \eta^2 M/N\end{aligned}$$

The expectation of the average correlation is given by:

$$\begin{aligned}\frac{\mathbb{E}[\sum_{i=1}^{j} \mathbf{x}_i^\top \hat{\mathbf{x}}_i]}{N} &= \mathbb{E}[\mathrm{Tr}(\mathbf{X}^\top \hat{\mathbf{X}})]/N \qquad (4.20)\\ &= \eta \mathbb{E}[\mathrm{Tr}(\mathbf{B}^\top(\mathbf{B}\mathbf{B}^\top)^{-1}\mathbf{B})]/N = \sqrt{M/N}.\end{aligned}$$

Sum reconstruction: $\hat{X} = \eta \mathbf{M}\mathbf{B}$. Constant η is s.t. the expectation of the average squared norm of $\|\hat{\mathbf{x}}_i\|^2$ equals 1.

$$\begin{aligned}\frac{\mathbb{E}[\sum_{i=1}^{N} \|\hat{\mathbf{x}}_i\|^2]}{N} &= \eta^2 \mathbb{E}[\mathrm{Tr}(\hat{\mathbf{X}}^\top \hat{\mathbf{X}})]/N\\ &= \eta^2 \mathbb{E}[\mathrm{Tr}((\mathbf{B}\mathbf{B}^\top)^2)]/N. \qquad (4.21)\end{aligned}$$

We define $\tilde{\mathbf{B}} := \frac{(\mathbf{B}-\nu \mathbf{1}_{M:N})}{\nu(1-\nu)}$ with $\nu := n/N$. Its columns are centered random vectors whose covariance matrix is the identity. The eigenvalues of their empirical covariance matrix $\tilde{\mathbf{B}}\tilde{\mathbf{B}}^\top/N$ follows the Marchenko-Pastur distribution. For large N and $\gamma = M/N$, we then have:

$$\mathbb{E}[\mathrm{Tr}((\tilde{\mathbf{B}}\tilde{\mathbf{B}}^\top)^2)]/MN^2 = 1 + \gamma. \qquad (4.22)$$

Expressing $(\mathbf{B}\mathbf{B}^\top)^2$ as a function of $\tilde{\mathbf{B}}\tilde{\mathbf{B}}^\top$ leads to:

$$\mathbb{E}[\mathrm{Tr}((\mathbf{B}\mathbf{B}^\top)^2)] = \nu^2 MN^2(1-\nu^2+\gamma(1-\nu)^2+\nu^2 M), \qquad (4.23)$$

which gives the value of η thanks to (4.21). On the other hand, the expectation of the average correlation is given by:

$$\begin{aligned}\frac{\mathbb{E}[\sum_{i=1}^{j} \mathbf{x}_i^\top \hat{\mathbf{x}}_i]}{N} &= \eta \mathbb{E}[\mathrm{Tr}(\mathbf{X}^\top \mathbf{X}\mathbf{B}^\top(\mathbf{B}^\top\mathbf{B})^{-1}\mathbf{B}]/N\\ &= \eta \mathbb{E}[\mathrm{Tr}(\mathbf{B}^\top(\mathbf{B}^\top\mathbf{B})^{-1}\mathbf{B}]/N = \eta M/N\\ &= \sqrt{\frac{\gamma}{1-\nu^2+\gamma(1-\nu)^2+\nu^2 M}} \qquad (4.24)\end{aligned}$$

4.3 Conclusion

This chapter proposes further applications and extensions of memory vectors. First, drawing the lessons from the pioneer applications of group testing to similarity search [SFJ14, IFG$^+$17], we propose a grouping strategy based on kd-trees. This ensures groups of even sizes comprising weakly similar features. An important byproduct of this approach is that the resulting representative vectors have small intrinsic dimensionality compared to the raw dataset features. This allows us to integrate an ANN search algorithm, such as FLANN, into the group testing framework. This is the keystone for freeing the total number of groups. We seize this opportunity to increase this number to compute more reliable scores. This in turn gives birth to a shorter list of candidates for the verification step. The overall construction improves the scalability of group testing similarity search while maintaining high search quality.

Secondly, we presented a privacy and security enhancing scheme for approximate search. It is built upon a first version which enables users privacy but not security of the owner's data especially when user and server collude. Contrary to the "Signal Processing in the Encrypted Domain" trend which uses even more advanced cryptographic primitives like full homomorphic encryption, we propose an alternative only based on signal processing. It is much faster and more scalable (even more that the first version). Yet, the attack is not absolutely blocked, but relatively in the sense that vector reconstruction is so bad that it can't be exploited.

Chapter 5

Optimizing Group Testing Using Matrix Factorization

Previous chapters demonstrated the use of memory vectors and group testing in image retrieval. Even though the proposed methods showed significant improvements in terms of efficiency, there still are certain drawbacks. First of all, we only consider engineered group representations (sum or pinv), and group assignment is independent from the representation. Secondly, our initial attempts considered an adaptive group testing approach. M groups are composed from the dataset, and querying proceeds in two stages. In the first stage, the scores between group vectors and the query are computed. They measure how likely their group contains some matching images. Then, in the second stage, the query is compared with individual image vectors for only the mostly likely positive groups. If the groups are roughly balanced in size and the query only matches a small number of group vectors, then the complexity is reduced from dN to $d(M + N/M)$. Although this results in efficient image retrieval, it has one major drawback: memory usage is increased since the group vectors and mapping from images to groups are stored in addition to the dataset feature vectors. In other words, these works trade complexity for memory.

In this chapter, we pursue the idea of deducing which vectors are matching in a database of size N from only $M < N$ measurements. We re-examine the group testing formulation. Rather than a random partition of the dataset into groups followed by a specific construction of the group vectors, we formulate the problem of finding an optimal group testing design for a given image dataset. Removing the restriction to binary designs, the continuous version of this optimization problem turns out to be equivalent to dictionary learning. For small and medium sized datasets, with $N < d$, one can remove the requirement of a sparse design matrix, and then the problem simplifies further to that of a matrix factorization whose solution is given by the SVD.

The chapter is organized as follows. Section 5.1 introduces the problem formulation and notation. Section 5.2 proposes different techniques to solve the problem depending on the parameters N and d. Section 5.3 shows the compatibility of our approach with an existing coding method in the literature. Section 5.4 presents the evaluation of proposed method using real image datasets. This work has been published in [Iscen et al., 2016b].

5.1 Problem Statement

The dataset is composed of N d-dimensional vectors $\{\mathbf{x}_i\}_{i=1}^N$ such that $\|\mathbf{x}_i\| = 1$, for all i, and each $\mathbf{x}_i$ is the global feature vector of one image in the dataset. The similarity between two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ is the scalar product $\mathbf{x}_i^\top \mathbf{x}_j$. Denote by $\mathbf{X}$ the $d \times N$ matrix $[\mathbf{x}_1, \dots, \mathbf{x}_N]$.

We aim to find M group vectors of dimension d, $\{\mathbf{y}_i\}_{i=1}^M$, stored in $d \times M$ matrix $\mathbf{Y}$. Unlike the previous group testing approaches, we do not randomly assign dataset vectors to groups and we do not compute the group vectors according to a specific construction. Our goal is to directly find the best M group vectors globally summarizing the dataset. We call this process the encoding, and we restrict our scope to a linear encoding:

$$\mathbf{Y} = \mathsf{enc}(\mathbf{X}) = \mathbf{X}\mathbf{G}^\top. \tag{5.1}$$

Given a query image, represented by its global descriptor vector $\mathbf{q}$, we compute the group scores,

$$\mathbf{s} = \mathbf{q}^\top \mathbf{Y}. \tag{5.2}$$

Finally, we estimate the similarities between query and database vectors $\mathbf{c} = \mathbf{q}^\top \mathbf{X}$ from the measurements $\mathbf{s}$. Again, we assume a linear estimator:

$$\hat{\mathbf{c}} = \mathsf{dec}(\mathbf{s}) = \mathbf{s}\mathbf{H}. \tag{5.3}$$

Our aim is to design $\mathbf{G} \in \mathbb{R}^{M \times N}$ and $\mathbf{H} \in \mathbb{R}^{M \times N}$ to allow for a fast and accurate search. Note that this setup is similar to the pioneering work of Shi et al. [SFJ14]: in their paper, $\mathbf{G}$ is indeed a randomly generated binary matrix where $G(i,j) = 1$ if $\mathbf{x}_j$ belongs to the i-th group and $G(i,j) = 0$ otherwise. Hence, in the previous group testing approach, $\mathbf{G}$ captures both how groups are made and how the group vectors are computed (a simple sum in [SFJ14]). On the contrary, we look for the best matrix representing the dataset, which will heavily depend on $\mathbf{X}$.

Complexity. Exhaustive search involves computing $\mathbf{q}^\top \mathbf{X}$, which has a complexity of dN. Computing the group measurements (5.2) takes dM operations, and the decoding (5.3) takes MN. This gives a complexity of $dM + NM$ for group-testing search, compared to dN operations for exhaustive search. The complexity ratio is thus $\rho = M/N + M/d$, implying that M must be smaller than both N and d to yield efficient queries.

Previous work based on group testing [SFJ14, IFG+17] designs groups so that every column of $\mathbf{G}$ has exactly $m \ll M$ ones; i.e., each dataset vector belongs to m groups.

This produces a sparse decoding matrix $\mathbf{H}$ which, in turn, yields the better complexity ratio $\rho = M/N + m/d$. However, none of the approaches [SFJ14, IFG$^+$17] attempt to optimize $\mathbf{G}$ and $\mathbf{H}$. They either create $\mathbf{G}$ randomly or use a clustering algorithm to coarsely group similar dataset vectors [IFG$^+$17]. In the following sections, we discuss two techniques that optimize the matrices $\mathbf{G}$ and $\mathbf{H}$ for a particular dataset $\mathbf{X}$.

We focus on the complexity of performing a query. Determining the optimal encoding and decoding matrices $\mathbf{G}$ and $\mathbf{H}$ requires additional computation applied offline or periodically. We assume that the corresponding complexity is not as critical as in the query stage. Our only requirement is that the complexity of this offline computation be polynomial in N and d to ensure that it is tractable.

5.2 Proposed Solutions

We now provide two alternative solutions for the setup described in Section 5.1. As we will show in the experimental section, both solutions have advantages and drawbacks, and can be chosen depending on the feature vectors and the number of items in the dataset.

5.2.1 First Solution: Eigendecomposition

In the first approach, we consider finding matrices $\mathbf{G} \in \mathbb{R}^{M\times N}$ and $\mathbf{H} \in \mathbb{R}^{M\times N}$ so that the approximate scores $\hat{\mathbf{c}}$ and exact scores $\mathbf{c}$ are as close as possible. Based on (5.1), (5.2) and (5.3), this amounts to:

$$\begin{aligned} \underset{\mathbf{G},\mathbf{H}}{\text{minimize}} \quad & \sum_{\mathbf{q}\in\mathcal{Q}} \|\mathbf{c} - \hat{\mathbf{c}}\|_2^2 \quad = \\ \underset{\mathbf{G},\mathbf{H}}{\text{minimize}} \quad & \sum_{\mathbf{q}\in\mathcal{Q}} \|\mathbf{q}^T\mathbf{X} - \mathbf{q}^T\mathbf{X}\mathbf{G}^\top\mathbf{H}\|_2^2, \end{aligned}$$

where $\mathcal{Q}$ is assumed to be representative of typical queries. Of course, this distance cannot be zero for all $\mathbf{q} \in \mathbb{R}^d$ since the $N \times N$ matrix $\mathbf{G}^\top\mathbf{H}$ has rank at most $M < N$. We focus on providing accurate scores for typical queries. We use the dataset of vectors itself as a proxy of the typical ensemble of queries. This amounts to replacing $\mathbf{q}$ by $\mathbf{X}$ and to consider the Frobenius matrix norm:

$$\underset{\mathbf{G},\mathbf{H}}{\text{minimize}} \left\|\mathbf{X}^\top\mathbf{X} - \mathbf{X}^\top\mathbf{X}\mathbf{G}^\top\mathbf{H}\right\|_F^2. \tag{5.4}$$

This problem is commonly solved by eigendecomposition. Let $\mathbf{A} = \mathbf{X}^\top\mathbf{X}$ be the Gramian symmetric matrix associated to $\mathbf{X}$. As a real symmetric matrix, $\mathbf{A}$ is diagonalizable: $\mathbf{A} = \mathbf{U}\Lambda\mathbf{U}^\top$, where $\mathbf{U}$ is an orthogonal matrix ($\mathbf{U}^\top\mathbf{U} = \mathbf{U}\mathbf{U}^\top = \mathbf{I}_N$). This means that we can simply assign $\mathbf{G}^\top = \mathbf{U}_M$ and $\mathbf{H} = \mathbf{U}_M^\top$, where $\mathbf{U}_M$ are the eigenvectors associated with the M largest eigenvalues.

In practice, we do not need to compute the Gram matrix $\mathbf{A} = \mathbf{X}^\top\mathbf{X}$. The singular value decomposition (SVD) of $\mathbf{X}$ is defined as $\mathbf{X} = \mathbf{S}\Sigma\mathbf{U}^\top$, where $\mathbf{S}$ are the eigenvectors of $\mathbf{X}\mathbf{X}^\top$, and $\mathbf{U}$ are the eigenvectors of $\mathbf{X}^\top\mathbf{X}$. Hence, this SVD gives us the desired output without having to calculate $\mathbf{A}$. It is worth noting that this solution resembles a well known dimension reduction method: Principal Component Analysis (PCA). However, while PCA is usually employed to reduce the dimensionality of the vectors from d to d' components, in our approach we use it to reduce the number of vectors from N to M. Alternatively, more efficient dimensionality reduction methods, such as sparse projectors [NPG14], can be used to construct $\mathbf{H}$.

The major drawback of this approach is that $\mathbf{H}$ is not sparse. Therefore, the complexity of the decoding (5.3) is in $\mathcal{O}(MN)$. Hence, this solution is efficient for scenarios where d is larger than N.

5.2.2 Second Solution: Dictionary Learning

Dictionary learning has been widely applied in imaging problems, e.g., to obtain efficient representations and discover structure using local patches; see [MBP14] for a survey. Our second solution applies dictionary learning to find a sparse description of the dataset enabling efficient image search. For any query $\mathbf{q}$, we expect the score vector $\mathbf{c}$ to be sparse; the few high-amplitude coefficients correspond to the matching images, and remaining low-amplitude coefficients correspond to non-matching images. Moreover, we do not need the estimate $\hat{\mathbf{c}}$ to be very close to $\mathbf{c}$, per se, as long as the matching images receive a substantially higher score than the non-matching ones.

Because the three steps (5.1), (5.2) and (5.3) of our method are linear, this reconstruction of the similarities through a sparse matrix $\mathbf{H}$ implies a sparse representation of the dataset vectors, which leads to the connection with dictionary learning. Specifically, we aim to approximate $\mathbf{X}$ by $\mathbf{YH}$ where $\mathbf{H} \in \mathbb{R}^{M\times N}$ stores the sparse representations of the dataset vectors in terms of columns (so-called atoms) of the dictionary $\mathbf{Y} \in \mathbb{R}^{d\times M}$. This leads to the following optimization problem:

$$\begin{aligned} \underset{\mathbf{Y},\mathbf{H}}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{X}-\mathbf{YH}\|_F^2 + \lambda\|\mathbf{H}\|_1 \\ \text{subject to} \quad & \|\mathbf{y}_k\|_2 \le 1 \text{ for all } 0 \le k < M. \end{aligned}$$

The ℓ_1-norm penalty on $\mathbf{H}$ (sum of the magnitude of its elements) encourages a solution where each column of $\mathbf{X}$ can be represented as a sparse combination of columns of the dictionary $\mathbf{Y}$. The level of sparsity depends on λ. Unlike the previous solution of Section 5.2.1, this scheme is competitive when N is larger than d since we benefit from the reduced complexity of sparse matrix multiplication. An algorithm such as Orthogonal Matching Pursuit (OMP) [PRK93, DMZ94] allows us to strictly control the sparsity of $\mathbf{H}$. For a given dictionary $\mathbf{Y}$, OMP finds $\mathbf{H} = [\mathbf{h}_1, \cdots, \mathbf{h}_N]$ by sequentially solving

$$\begin{aligned} \underset{\mathbf{h}_i}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{x}_i-\mathbf{Y}\mathbf{h}_i\|_2^2 \\ \text{subject to} \quad & \|\mathbf{h}_i\|_0 \le m. \end{aligned}$$

Adopting this algorithm, we control the sparsity of the matrix $\mathbf{H}$ by setting m to a desired value. Note that this solution is directly related with the problem statement in

Section 5.1, even if $\mathbf{G}$ is not directly a part of the solution. The reconstruction of the vectors $\mathbf{X}$ is linear up to an approximation, $\mathbf{X} \approx \mathbf{YH}$. Since this is a linear process , we have $\mathbf{Y} = \mathbf{XG}^\top$ (1) where $\mathbf{G}^\top = \mathbf{H}^+$ (pseudo-inverse). Therefore, the connection is obvious. Furthermore, $\mathbf{G}$ is not needed during the search; what matters is $\mathbf{Y}$ and $\mathbf{H}$.

This solution is similar to the recently proposed indexing strategy based on sparse approximation [BMM15], which also involves training a dictionary $\mathbf{Y}$ and a sparse matrix $\mathbf{H}$. However, the way these matrices are used in [BMM15] is completely different from the approach proposed here. Their framework adheres to a space partitioning approach; it indexes each descriptor in buckets using an inverted file based on the non-zero entries of $\mathbf{H}$. For a given query, their system runs orthogonal matching pursuit (OMP) to find a sparse approximation, and then it calculates distances between the query and the dataset vectors that share the same buckets. In contrast, the method proposed here involves no indexing and makes no direct distance calculations between the query and the dataset vectors. Indeed, this allows us to completely avoid touching dataset vectors at query time.

Similarly, clustering can be used to make groups, as in traditional indexing approaches [ML14], but the decoding does not perform well for the following reason. The decoding matrix is too sparse: a single non-zero component in each column (this vector belongs to that cluster). This requires an additional verification step after the decoding step for the vectors in the leading cluster. This is not needed in our method, hence we obtain huge savings in complexity and memory. Our approach can be seen as performing a sort-of soft clustering, where each vector belongs to multiple clusters with different weights.

5.2.3 Large-scale Dictionary Learning

When designing an image search system, one must consider large-scale problems consisting of millions to billions of images. Our primary goal is an efficient image search system whose query time complexity (computational, and memory) is reduced. Although we have been ignoring the complexity of the encoding phase, by assuming that the complexity of this stage is less critical application-wise, it should remain tractable.

One of the most widely-known dictionary learning algorithms is that proposed by Mairal et al. [MBPS10]. This algorithm provides a fast implementation and allows other possibilities such as mini-batch learning and online dictionary updates. These features make it an attractive algorithm for large-scale problems. However, the training time increases dramatically with M for large-N datasets, as reported in Section 5.4. Even though this calculation needs to be done only once in the offline stage, we still need a scalable training approach to index all dataset vectors easily.

One solution is to use a subset of dataset vectors as a surrogate for the entire dataset. Once the dictionary $\mathbf{Y}$ is trained on the subset, a less expensive sparse decoding algorithm, such as OMP, can be used to compute the matrix $\mathbf{H}$ for the entire dataset.

Elhamifar et al. [ESV12] propose a solution similar to dictionary learning, with the sole aim of finding representatives from the data. A related approach is to use coresets [AHPV04]. A coreset $\mathbf{C}$ is a problem-dependent approximation of a dataset

$\mathbf{X}$. Feldman et al. [FFS13] show that for every $\mathbf{X}$ and $\epsilon > 0$ there exists a coreset $\mathbf{C} \in \mathbb{R}^{d \times N'}$, $N' < N$, for which the following inequality holds:

$$\begin{aligned}(1-\epsilon)\cdot \min_{\mathbf{H}\in\mathbb{R}^{M\times N}} \|\mathbf{X}-\mathbf{Y}\mathbf{H}\|_F^2 &\le \min_{\widetilde{\mathbf{H}}\in\mathbb{R}^{M\times N'}} \left\|\mathbf{C}-\mathbf{Y}\widetilde{\mathbf{H}}\right\|_F^2 \\ &\le (1+\epsilon)\cdot \min_{\mathbf{H}\in\mathbb{R}^{M\times N}} \|\mathbf{X}-\mathbf{Y}\mathbf{H}\|_F^2 .\end{aligned}$$

Typically, $\mathbf{C}$ has many fewer columns than $\mathbf{X}$, thereby summarizing the whole dataset with just a few representatives. The main advantage of this approach is its speed. Finding a coreset for a large-scale dataset takes a short time, only a few seconds in our experiments. Then, running dictionary learning on the coreset is significantly faster than on the original dataset. We empirically evaluate the speedup and the effect on accuracy in the experimental section.

5.3 Compressed Dictionaries

Instead of dealing with a database of N image vectors of length d, our novel approach now manages a database of M group vectors of the same dimension. Compared to a linear scan, we reduce the number of comparisons from N to M, and yet, rank N items based on their estimated score.

Nevertheless, our scheme remains compatible with the traditional coding methods. Instead of a linear scan browsing group vectors, we can add on top of our method an approximate search. This can take the form of either an embedding producing compact representations of the group vectors, or an indexing structure finding the closest group vectors w.r.t. a query. This improves even further the overall efficiency.

Case study: Combination with PQ-codes. An embedding offers a compact representation of group vectors allowing a fast approximation of their dot products with the query. PQ-codes [JDS11], for instance, are a priori not compliant since they operate on Euclidean distances. We convert Euclidean distance to cosine similarity in the following way. Each group vector $\mathbf{y}$ is split into ℓ subvectors $\tilde{\mathbf{y}}_u$, where $1 \le u \le \ell$. Each subvector $\tilde{\mathbf{y}}_u$ is quantized using the codebook $\mathcal{C}_u = \{\mathbf{c}_{i,u}\}_{i=1}^Q$: $v_u = \arg\min_{1\le i\le Q} \|\tilde{\mathbf{y}}_u - \mathbf{c}_{i,u}\|$. The compact representation of $\mathbf{y}$ is the list of codeword indices $(v_1, \dots, v_\ell) \in \{1, \dots, Q\}^\ell$. This is exactly the same encoding stage as the original PQ-codes [JDS11].

The dot product query *vs* group vector is approximated by the dot product query *vs* quantized group vector:

$$\mathbf{q}^\top \mathbf{y} = \sum_{u=1}^{\ell} \tilde{\mathbf{q}}_u^\top \tilde{\mathbf{y}}_u \approx \sum_{u=1}^{\ell} \tilde{\mathbf{q}}_u^\top \mathbf{c}_{v_u,u}, \tag{5.5}$$

where $\tilde{\mathbf{q}}_u$ is the u-th subvector of the query. As in the original application of PQ-codes, the quantities $\{\tilde{\mathbf{q}}_u^\top \mathbf{c}_{i,u}\}$ are computed at query time and stored in a lookup table for evaluating (5.5) efficiently over a large number of group vectors. Using approximate dot products is an additional source of error, but experiments in the next section show that the decoding schemes described above gracefully handle this.

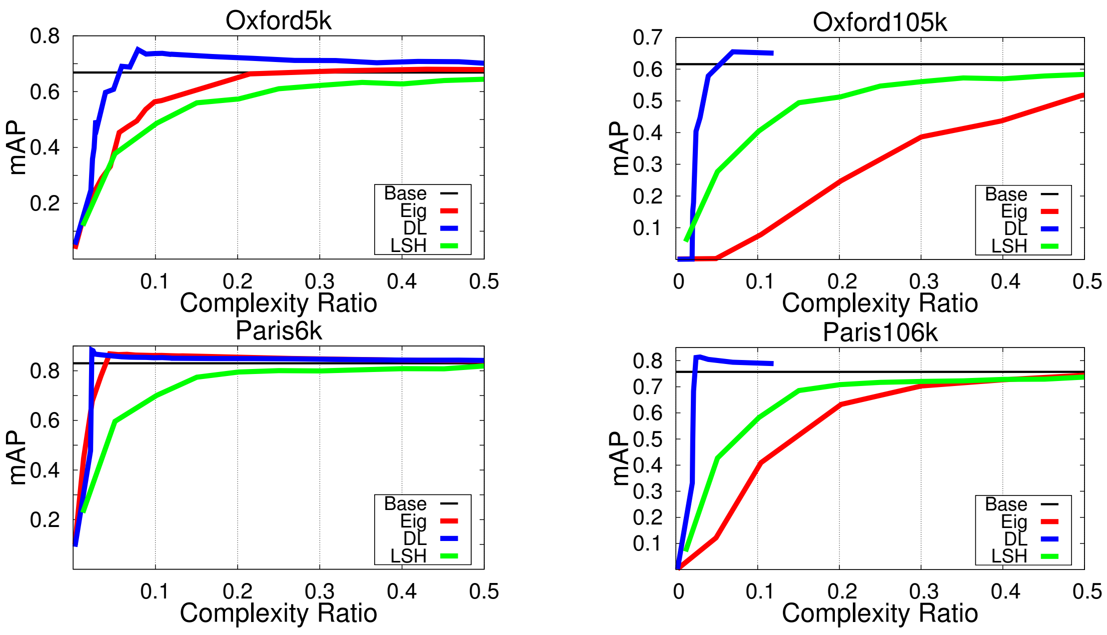


Figure 5.1: Comparison of eigendecomposition, dictionary learning (DL), and LSH [Cha02]. DL gives better performance, all the more so as the dataset is large. We only evaluate DL up to $M/N = 1/10$ for Oxford105k and Paris106k. Performance eventually converges to the baseline after this point.

5.4 Experiments

After detailing the experimental protocol, we report retrieval performance results together with a comparison with other image retrieval approaches.

5.4.1 Experimental Setup

Datasets. We evaluate our retrieval system using the Oxford5k [PCI+07] and Paris6k [PCI+08] datasets, which contain 5,063 and 6,412 images respectively. For large-scale experiments we add 100,000 Flickr distractor images [PCI+07], resulting in datasets referred as to Oxford105k and Paris106k. Additionally, we use the Yahoo Flickr Creative Commons 100M dataset [TSF+16] (referred as to Yahoo100M), which comprises about 100 million image vectors. For comparison with other works, we also run experiments on the Holidays [JDS08] and UKB [NS06] datasets.

For each dataset, we follow its standard procedure to evaluate performances. The mean Average Precision (mAP) measures the retrieval quality in all datasets except for UKB, where the performance is gauged by 4×recall@4.

Features. For most of our experiments, we use the state-of-the-art R-MAC features [TSJ16]. Depending on the network used, these features have dimensionality of either $d = 512$ or $d = 256$.[1] In section 5.4.3, we use T-embedding features [JZ14] with $d = 8,064$ to allow a more direct comparison with the most similar concurrent methods. For Yahoo100M, we use VLAD [JDSP10] with $d = 1,024$ as in [SXPK$^+$14].

Complexity analysis. We report the complexity ratio, $\rho = (Md+s)/dN$, where $s = \text{nnz}(\mathbf{H})$ is the number of non-zero elements of matrix $\mathbf{H}$. For the eigendecomposition, we set $s = MN$, whereas for dictionary learning (Section 5.2.2), m controls the sparsity of $\mathbf{H}$ making the complexity ratio $\rho = M/N + m/d$. Unless otherwise specified, we set $m = 10$ for R-MAC features; when $d = 512$ then decoding contributes only 0.02 to ρ (i.e., 2% of the complexity of exhaustive search). The memory ratio, the ratio of the memory required compared to that of exhaustive search, is equal to ρ for non-sparse $\mathbf{H}$. When $\mathbf{H}$ is sparse, we need to store mN scalars and their indices, making the memory ratio $M/N + m/d + m\log_2(M)/d \approx \rho$.

5.4.2 Retrieval Performance

We first evaluate our system for different M using either eigendecomposition or dictionary learning solutions. We also include the popular sketching technique LSH [Cha02], which approximates similarity by comparing binary compact codes of length $d' = \rho d$. We measure the retrieval performance in terms of mAP and complexity ratio as mentioned in Section 5.4.1.

Figure 5.1 shows the retrieval performance for different complexity ratios. It is clearly seen that eigendecomposition suffers at low complexity ratio in large-scale datasets. This is expected because we must set M to a very small value to obtain a low complexity ratio since the decoding matrix $\mathbf{H}$ is not sparse in this solution. On the other hand, we can set M to a much higher value for a given complexity ratio using dictionary learning since $\mathbf{H}$ is sparse.

Our variant based on dictionary learning performs better than the baseline on all datasets. One would expect the performance to be worse than baseline for $M \ll N$ due to loss of information, but this is surprisingly not the case. A possible explanation is that dictionary learning "denoises" similarities between vectors by looking at the distribution of images in the dataset.

To explore this phenomenon further, we plot the distribution of matching and non-matching vector similarities from Oxford5k using the original global descriptors. We repeat the same process using the reconstructed similarities from dictionary learning. As we see in Figure 5.2, both reconstructed similarity distributions have a lower variance than the original distributions. This is especially true for the non-matching distribution. This variance reduction increases the separation between the distributions, which translates to the better performance of our dictionary learning method.

Sparsity of H is controlled by parameter m in dictionary learning (see Section 5.2.2). This is an important factor in the complexity ratio ρ. The ratio between m and d

[1]Features available online: `ftp://ftp.irisa.fr/local/texmex/corpus/memvec/cvpr16/rmac/`

	Original	DL
Pos. μ	0.29	0.28
Pos. σ	0.16	0.15
Neg. μ	0.02	0.02
Neg. σ	0.06	0.04

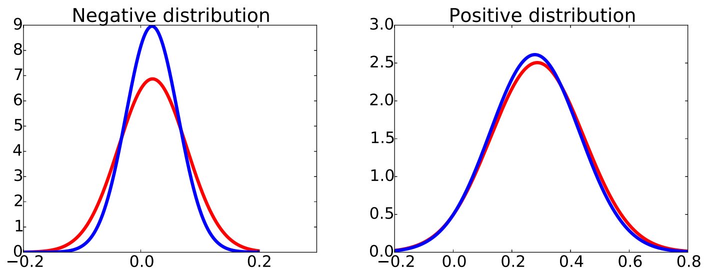


Figure 5.2: Distributions of matching and non-matching vector similarities from Oxford5k dataset. Red (blue) curves represent distributions of true (resp. reconstructed) similarities. The main improvement comes from the reduction of variance under the negative distribution.

contributes to ρ independently from M. It is possible to set this ratio to a small value to eliminate its influence.

We compute a dictionary of M atoms and we calculate several matrices $\mathbf{H}$ by applying OMP with different m. We plot the retrieval performance for different m and M in Figure 5.3. In most cases, the performance does not vary much w.r.t. m. The biggest difference is observed for Oxford105k where larger m leads to better performance for small M.

The dimensionality of the vectors is an important factor affecting the overall complexity. Lower dimensionality implies lower complexity and less memory usage. Although our experiments up to now are done in what can be considered as a low-dimensional feature space ($d = 512$), we evaluate our system with even smaller features, $d = 256$, in Figure 5.4. The results are similar to those for $d = 512$, although the accuracy of eigendecomposition increases at a slower rate for large N.

The training stage computes $\mathbf{Y}$ and $\mathbf{H}$ and is performed only once and offline. However, it is important that this stage is scalable for updating the dictionary if needed. Experimentally, a small number of iterations (≈ 100) is sufficient for dictionary learning. This does not require much training time. Using Mairal et al. 's algorithm [MBPS10], we report the duration of the offline training on Figure 5.5. All experiments are done on a server with Intel® Xeon® E5-2650 2.00GHz CPU and 32 cores. The training time is reasonable for all datasets, but it increases dramatically with M in large datasets. Other training procedures would be necessary for handling large M and N.

Coresets , as explained in Section 5.2.3, reduce the training time even further for large datasets. Instead of using the entire dataset, we find a coreset $\mathbf{C}$ which represents the data with a few representatives vectors to train the dictionaries. We report results for coresets of different sizes in Table 5.1. Empirically, we achieve a similar performance by training on coresets of vectors. This allows us to train the dictionary for larger M in

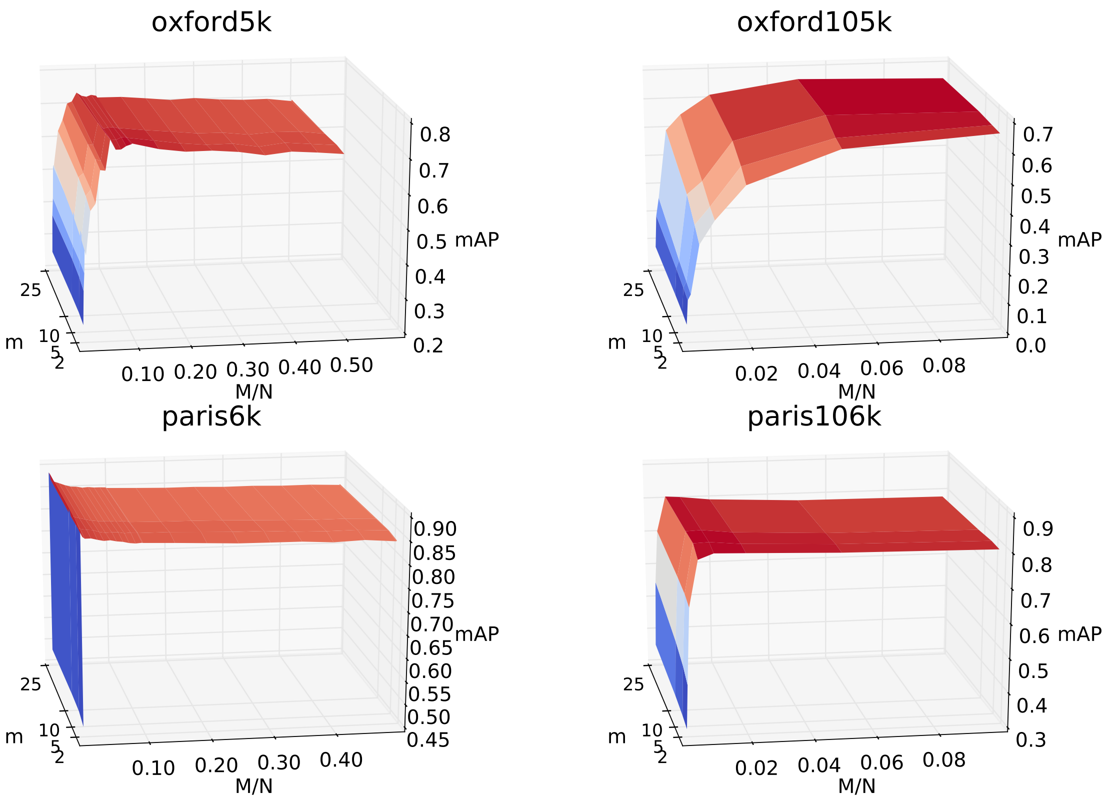


Figure 5.3: Retrieval performance with different M and m. Varying m does not affect the performance in most cases, except for Oxford105k, where increasing m improves performance for small M.

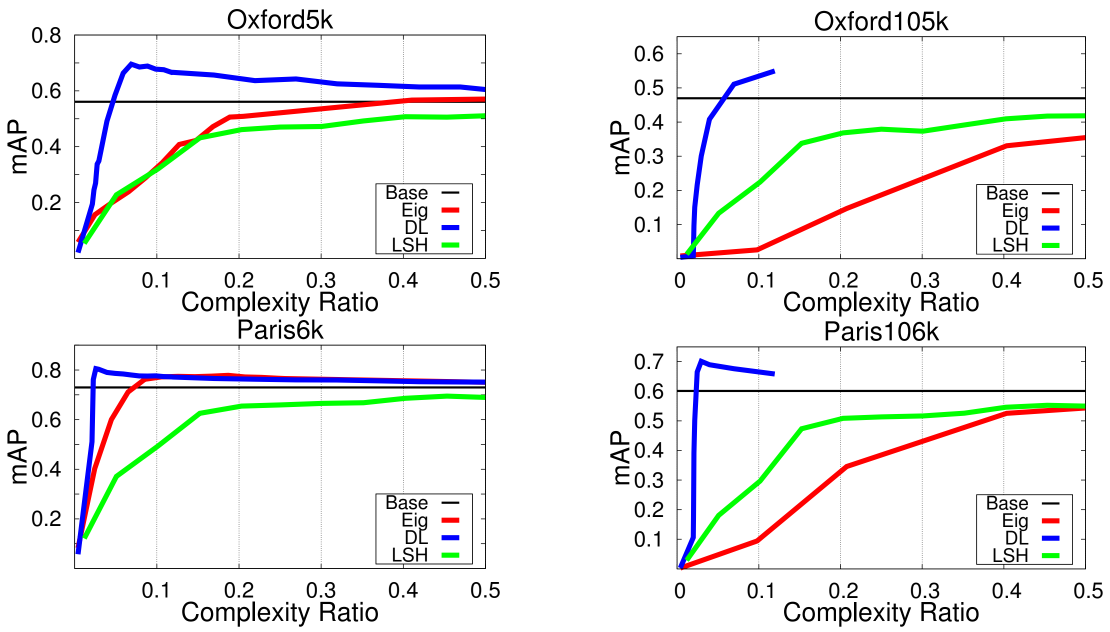


Figure 5.4: Retrieval performance using smaller features: $d = 256$.

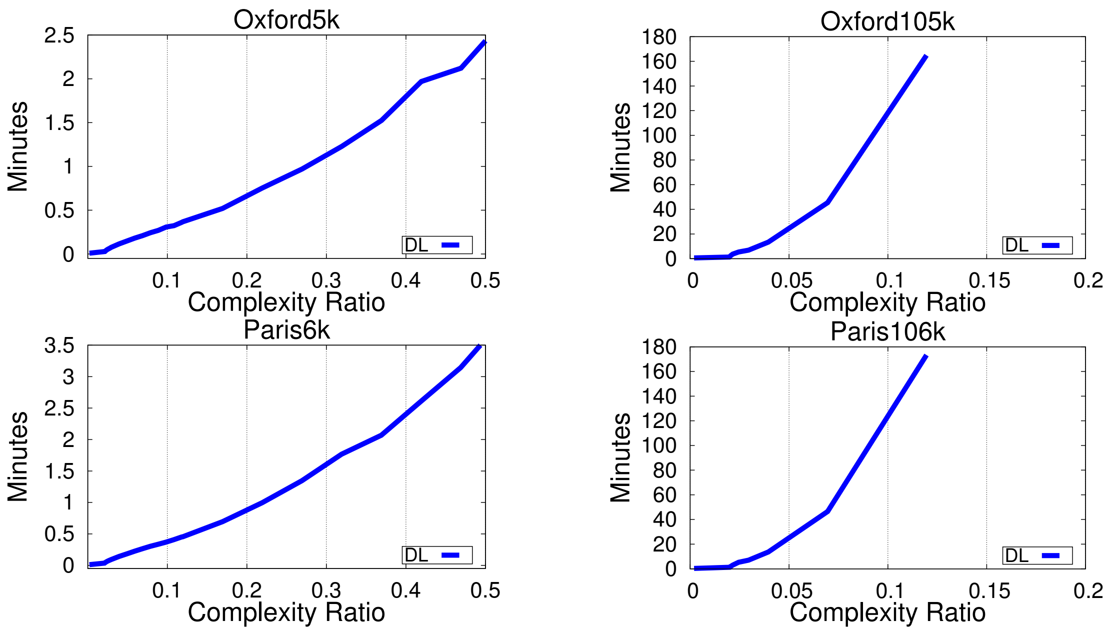


Figure 5.5: Offline training time needed for dictionary learning with 100 iterations.

	Oxford105k		Paris106k	
	mAP	**Time**	**mAP**	**Time**
$\lvert\mathbf{C}\rvert = N/10$	60.1±1.1	14.6	78.3±1.0	1.8
$\lvert\mathbf{C}\rvert = N/5$	62.1±1.2	16.9	79.2±0.8	2.3
$\lvert\mathbf{C}\rvert = N/2$	62.7±0.4	23.9	79.5±0.4	3.3
$\mathbf{X}$	65.5	45.5	81.2	5.3

Table 5.1: Performance and training time (in minutes) using coresets to train the dictionary. M is set to $5,257$ and 532 for Oxford105k and Paris105k respectively, and $m = 50$. Each experiment is run 5 times, and we report the mean and the standard deviation.

	Mem. Ratio	Holidays	Oxford5k	UKB
Exhaustive	1.0	77.1	67.4	3.63
Iscen *et al.* [IFG$^+$17]-Kmeans	1.4	**76.9**	67.3	**3.63**
Iscen *et al.* [IFG$^+$17]-Rand	1.4	75.8	62.0	**3.63**
Shi *et al.* [SFJ14] w/ bp.	1.4	75.5	64.4	**3.63**
Borges *et al.* [BMM15]	1.0	59.2	59.9	3.43
LSH [Cha02]	0.4	73.9	65.8	3.61
PCA	0.4	75.4	64.3	3.61
Shi *et al.* [SFJ14] w/o bp.	0.4	8.7	24.1	1.33
Ours - Eigen.	0.4	**76.9**	67.7	**3.63**
Ours - Dict. Learn.	0.4	55.2	**68.8**	3.59

Table 5.2: Comparison in image retrieval for a given complexity ratio of 0.4. This experiment uses long T-embedding features ($d = 8,096$). Eigendecomposition and dictionary learning generally perform better at lower memory ratio.

just a few minutes. Note that Paris106k has fast training time even without coresets. This is because the best performance for this dataset is obtained with $M = 532$, a rather small value. The drawback to using coresets is that $\mathbf{H}$ is less sparse: $m = 50$. This results in the same performance but slightly higher complexity.

The search time is the average number of seconds to respond to a query. Although comparing vector operations is reliable in general, we also include the actual timings. Exhaustive search takes $0.029s$ on Oxford105k and $0.03s$ on Paris106k (average per query). Our method takes $0.003s$ on Oxford105k ($M = 5,257$), and $0.001s$ on Paris106k ($M = 532$), with higher mAP than exhaustive search.

5.4.3 Comparison with Other Methods

We compare our system with other image retrieval approaches. First we compare with the popular FLANN toolbox [ML14] using Oxford105k and R-MAC features. We set the target precision to 0.95 and use the “autotuned” setting of FLANN, which optimizes the indexing structure based on the data. We repeat this experiment 5 times. The average speed-up ratio provided by the algorithm is 1.05, which corresponds to a complexity

	Mem. Ratio	Oxf5k	Oxf105k	Paris6k	Paris106k
Exhaustive	1.0	66.9	61.6	83.0	75.7
Iscen *et al.* [IFG$^+$17]-Kmeans	1.1	65.6	61.2	79.7	75.7
Iscen *et al.* [IFG$^+$17]-Rand	1.1	25.1	43.7	21.2	44.4
Shi *et al.* [SFJ14] w/ bp.	1.1	15.4	28.1	18.7	37.7
Borges *et al.* [BMM15]	1.0	8.5	22.7	8.2	18.9
LSH [Cha02]	0.1	48.6	40.5	70.1	58.2
PCA	0.1	58.1	8.0	86.1	38.9
Ours-Eigen.	0.1	56.8	8.0	**86.3**	40.9
Ours-D.L.	0.1	**73.7**	**65.5**	85.3	**78.9**

Table 5.3: Comparison with R-MAC features ($d = 512$) and 0.1 complexity ratio.

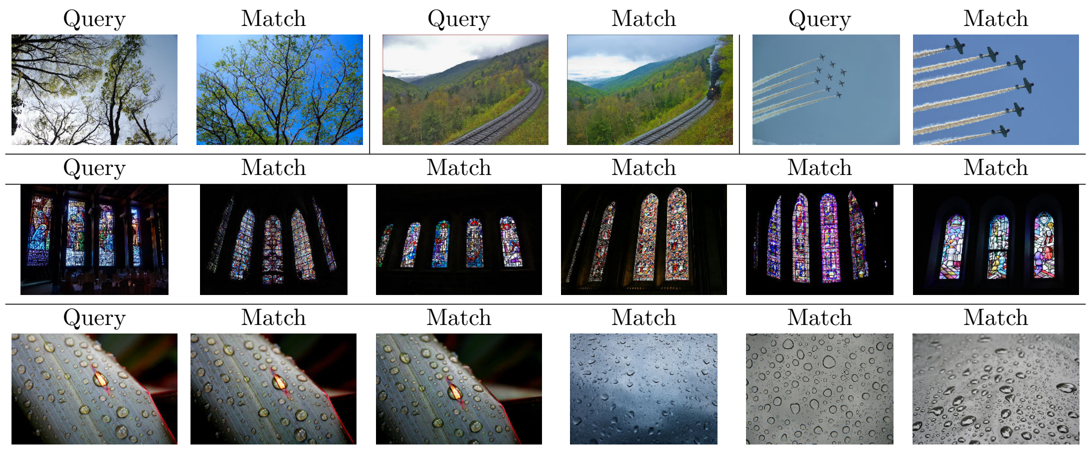


Figure 5.6: Some examples of match and query in Yahoo100M dataset. Two vectors are considered a match if their similarity is above 0.5.

ratio of 0.95. In other words, FLANN is ineffective for these R-MAC descriptors, most likely due to their high intrinsic dimensionality ($d = 512$): as discussed by its authors [ML14], FLANN is not better than exhaustive search when applied to truly high-dimensional vectors. In contrast, our approach does not partition the feature space and does not suffer as much the curse of dimensionality. Our descriptors are whitened for better performance [JC12], which tends to reduce the effectiveness of partitioning-based approaches.

Next we compare our method with other group testing and indexing methods in the image retrieval literature. To have a fair comparison, we report the performance using the same high-dimensional features ($d = 8,064$), same datasets, and the same complexity ratio as the group testing methods. Additionally, we also compare our scores to a dictionary learning-based hashing method [BMM15], LSH [Cha02] and PCA, where dimensionality of vectors is reduced such that $d' = 0.4d$.

Table 5.2 shows the comparison for a fixed complexity ratio. We outline two observations. First, eigendecomposition works well in these experiments. This is especially true for the Holidays dataset where $N = 1,491$ and $d = 8,064$; large M can be used while keeping the complexity ratio low since $N < d$. This is clearly a scenario where it is plausible to use the eigendecomposition approach. Second, dictionary learning performs poorly for Holidays. This dataset contains only $1,491$ images, which constrains the size of the dictionary M to be small and prevents sparsity: the best parameters (via cross-validation) are found to be $M = 519$ and $m = 409$, giving $\rho = 0.4$. Note that this experiment uses long t-embedding descriptors ($d = 8,096$) in small and mid-scale datasets. Most likely, these features have low intrinsic dimensionality, and PCA and LSH are thus favored. Table 5.3 uses shorter R-MAC features ($d = 512$) for comparison. The increase in performance is more significant, especially for large datasets.

Yahoo100M is a recently released large-scale dataset consisting of approximately 100M images. Since there is no manually annotated ground-truth, we use the following evaluation protocol: a dataset vector is considered to match the query if its cosine similarity is at least 0.5. There are 112 queries randomly selected from the dataset. Each query has between 2 and 96 matches, and 11.4 matches on average. Table 5.4.2 shows visual examples of queries and matches.

This dataset is split into chunks of $N' = 100$k images. We run dictionary learning and OMP independently to learn matrices $\mathbf{Y}$ and $\mathbf{H}$ for each chunk, setting $M' = N'/100$ and $m = 100$. Overall, it results in $M = N/100$. We can perform this offline stage in parallel. At query time, we pool scores from each chunk together and sort them to determine a final ranking. When we evaluate the retrieval performance, we obtain a mAP of 89.4 with $\rho \approx 1/10$. This is a significant increase compared to running the same setup with LSH, which results in a mAP of 70.9. Furthermore, it is still possible for the dictionary learning approach to obtain very good performance with $\rho < 1/10$ by setting M and m to smaller values as shown in Table 5.4.3.

Similar to other datasets, we apply coresets for the Yahoo100M dataset. We learn a coreset for each chunk separately, which makes its calculation feasible. We set $|\mathbf{C}| = N/2$, $m = 100$ and $M = N/100$, and obtain a mAP of 87.9 compared to a mAP of 89.4

	$M = N/200$		$M = N/100$		$M = N/50$	
	mAP	ρ	**mAP**	ρ	**mAP**	ρ
$m = 100$	85.7	0.105	89.4	0.11	92.8	0.12
$m = 50$	81.0	0.055	84.7	0.06	87.4	0.07
$m = 20$	61.8	0.025	71.4	0.03	78.2	0.04

Table 5.4: Performance (mAP) and complexity ratio (ρ) in Yahoo100M for different M and m.

	Baseline	Our Method	$b = 8$	$b = 64$
Oxford5k	66.9	73.4	73.1	72.9
Paris6k	83.0	88.1	87.7	85.6
Oxford105k	61.6	65.5	63.1	30.4
Paris106k	75.7	81.2	80.9	76.8

Table 5.5: Combination of our method with PQ-codes. We use $M = 350$ for Oxford5k, $M = 30$ for Paris6k, $M = 5257$ for Oxford105k, and $M = 532$ for Paris106k.

using the entire chunks.

Compatibility with coding methods. One of the main strengths of our method is its complementarity with other popular coding strategies in computer vision. We combine our method with PQ-codes [JDS11] as explained in Section 5.3. We use $\ell = d/b$ subvectors for different values of b and $Q = 256$ codewords per each subquantizer (except for Paris6k where $Q = 16$ due to small M). This reduces the term $\mathcal{O}(M \times d)$ by a factor of b if we neglect the fixed cost of complexity of building the lookup table.

Table 5.5 shows the difference of performance with and without PQ-codes. Observe that the performance remains almost the same for $b = 8$. The compression factor by PQ-code is significant (8 floats replaced by 1 byte).

5.5 Conclusion

This chapter lowers the complexity of image search by reducing the number of vector comparisons. We formulate the image search problem as a matrix factorization problem, which can be solved using eigendecomposition or dictionary learning. We show that the former is a plausible option for small datasets, whereas the latter can be applied for large-scale problems in general. When applied to real datasets comprising up to 10^8 images, our framework achieves a comparable, and sometimes better performance, than exhaustive search within a fraction of complexity. It is worth noting that this approach is complementary to other indexing/approximated similarity approaches such that it can be combined to further increase efficiency.

Chapter 6

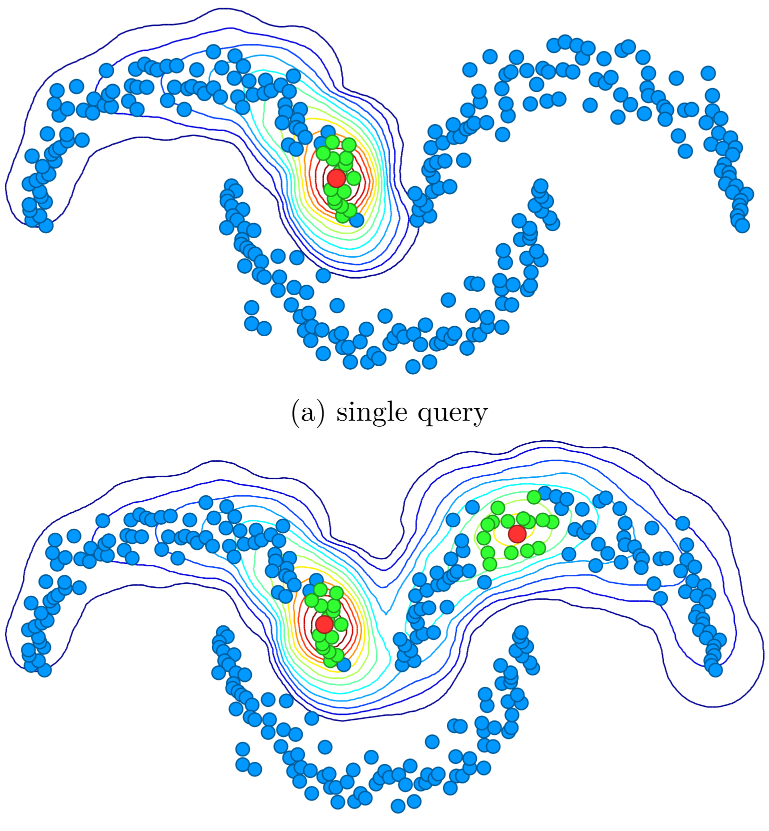
Manifold Search: Efficient Diffusion on Region Manifolds

Up until this point, the main focus of the problems that we studied in this thesis manuscript has been indexing. We derived group representations of sets of images to improve the efficiency of image retrieval. We switch gears in this chapter and focus on a different area of image retrieval, namely query expansion. We resort to a different kind of representation of sets of images; we use a weighted graph to represent the relationship between vectors in the dataset. This graph is used in the query time to retrieve images that lie on the same manifold as the query. While the main concern of previous chapters was to improve the efficiency of the search while sacrificing the accuracy, this chapter explores the other side of this trade-off. Our proposed method adds extra complexity during the query time, but the accuracy of the search is significantly improved compared to the baseline. The reader is encouraged to read Section 2.3 for a basic background knowledge of the topics that will be discussed in this chapter. The contents of this chapter has been published in [Iscen et al., 2017c].

6.1 Introduction

Object search is a key tool behind a number of applications like content based image collection browsing [WL11, MCM13], visual localization [SWLK12, AGT$^+$16], and 3D reconstruction [HSDF15, SRCF15]. Many applications benefit from retrieving images taken from various viewing angles and under different illumination, e.g. more information for the user while browsing, localization in day and night, and complete 3D models. Each image is represented by one or more descriptors designed or learned to exhibit a certain degree of invariance to imaging conditions. Retrieval is formulated as a nearest neighbor search in the descriptor space, performed by approximate methods [ML14, JDS11, KA14, BL16].

While collections of local descriptors provide good invariance, global descriptors



(a) single query

(b) multiple queries

Figure 6.1: Diffusion on a synthetic dataset in $\mathbb{R}^2$. Dataset points, query points and their k-nearest neighbors are shown in blue, red, and green respectively. Contour lines correspond to ranking scores after diffusion. In this work, points are region descriptors.

like VLAD [JDSP10] have smaller memory footprint, but are more prone to locking onto the clutter. This mainly holds when the queried object covers only a small part of the image. In case of global CNN descriptors, the invariance is partially designed by global max [ARS+14, TSJ16] or sum [KMO15, BL15] pooling layers or multi-scale querying [GARL16b], and partially learned by the choice of the training data. Robustness to background clutter is improved by computing descriptors over object proposals [MB15, GARL16a, XHZT15] or over a fixed grid of regions [TSJ16]. Better performance is observed at a cost of increased memory footprint [RSCM16].

In image collections, objects are depicted in various conditions. As a consequence, query and relevant images are often connected by a sequence of images, where consecutive images are similar. The descriptors of these images form a manifold in the descriptor space. Even though the images of the sequence contain the same object, the descriptors may be completely unrelated after a certain point.

This idea has been first exploited by Chum et al. [CPS+07] who introduce query expansion. The average query expansion (AQE) is now used as a standard tool in image retrieval, due to its efficiency and significant performance boost. However, AQE only explores the neighborhood of very similar images. Recursive and scale-band recursive methods [CPS$^+$07] further improve the results by explicitly crawling the image manifold. This is at a cost of increased query time.

Query expansion exploits the manifold of images at query time—starting from nearest neighbors of the query and using these neighbors to issue new queries. On the other hand, diffusion [PBMW99, ZWG$^+$03, DB13] is based on a neighborhood graph of the dataset that is constructed off-line and efficiently uses this information at query time to search on the manifold in a principled way.

We make the following contributions:

- We introduce a regional diffusion mechanism, which handles one or more query vectors at the same cost. There is one vector per region and a few regions per image so that constructing and storing the graph is tractable. This approach significantly improves retrieval of small objects and cluttered scenes.
- In diffusion mechanisms [PBMW99, ZWG$^+$03, DB13], query vectors are usually part of the dataset and available at the indexing stage. A novel approach to unseen queries with no computational overhead is proposed.
- Though a closed form solution to diffusion is known to exist, it has been explicitly avoided so far [DB13]. We show that the commonly used alternative is in fact a well known iterative linear system solver. Since the relevant matrix is sparse and positive definite, the conjugate gradient method is more efficient resulting in practical query times well below one second.
- To study the dependence of performance on relative object size, we experiment on INSTRE dataset [WJ15], which has not received much attention so far. We propose a new evaluation protocol that is in line with other well known datasets and provide a rich set of baselines to facilitate future comparisons.

6.2 Method

This section describes our contributions on image retrieval: handling new query points not in the dataset, searching for multiple regions with a single diffusion mechanism, and efficiently computing the solution.

6.2.1 Handling New Queries

In prior work on diffusion, a query point $\mathbf{q}$ is considered to be contained in the dataset $\mathcal{X}$ [ZBL$^+$03, DB13]. This does not hold in a retrieval scenario, but a query can be included in the dataset graph at query time [ZYC$^+$12] as follows. The k nearest neighbors $\mathrm{NN}_k(\mathbf{q})$ of $\mathbf{q}$ in $\mathcal{X}$ are found and reciprocity is checked. The rows and columns of the affinity matrix A corresponding to $\mathrm{NN}_k(\mathbf{q})$ are updated to maintain (2.30) in the presence of $\mathbf{q}$, and A is augmented by appending an extra row and column for $\mathbf{q}$. Matrix S is computed by normalizing A (2.26). Finally, vector $\mathbf{y}$ indicates that $\mathbf{q}$ is a query. Generalizing to multiple query points is straightforward.

Even if we ignore the time needed for the above computation, we argue that locking, modifying and augmenting the entire affinity matrix for each query is not acceptable in terms of space requirements[1]. We introduce here an alternative method which defines vector $\mathbf{y}$ in a new way rather than modifying A. Qualitatively, instead of searching for $\mathbf{q}$, we are searching for its neighbors $\mathrm{NN}_k(\mathbf{q})$, appropriately weighted. In particular, we define $\mathbf{y}$ as

$$y_i = s_k(\mathbf{x}_i|\mathbf{q}), \quad \forall i \in [n]. \tag{6.1}$$

Our motivation for this choice is detailed in Section 6.2.2 including the more general case of multiple query points. Figure 6.1 shows a toy 2-dimensional example of diffusion, where the k-nearest neighbors to each query point taken into account in (6.1) are depicted. It is evident that multiple manifolds are captured when multiple queries are issued. Section 6.3 experimentally shows improved performance compared to the conventional approach.

6.2.2 Regional Diffusion

The diffusion mechanism described so far is applicable to image retrieval when database and query images are globally represented by single vectors. We call this global diffusion in the rest of the chapter. Unlike the traditional representation with local descriptors [SZ03, PCI+07], global diffusion fits perfectly with the early CNN-based global features [BL15, KMO15, RTC16].

Global features still fail under severe occlusion or when the object of interest is small. Local CNN features from multiple image regions have been investigated for this purpose, either aggregated [GWGL14, TSJ16] or represented as a set [RSCM16]. Given a query image, the latter means that one searches for each query feature individually.

Fortunately, diffusion as defined in section 2.3.2 can already handle multiple queries. In the following, an image is represented by a set $X_i \subset \mathbb{R}^d$ of m points, one for each region. Dataset $\mathcal{X}$ is the union of such sets over all images; n still denotes its size. The query image is also represented by a set Q of m points. Each region feature is a point possibly lying on a different manifold. We discuss below the new definition of vector $\mathbf{y}$ and the combination of individual region ranking scores into a single score per image. We call this mechanism regional diffusion.

Specifying queries. In the conventional approach where query points are in the dataset, one directly applies (2.27) with $\mathbf{y} \in \{0,1\}^{n+m}$ with m non-zero elements indicating the query points. This situation resembles the personalized PageRank [PBMW99]. However, it is simpler to keep A as an $n \times n$ affinity matrix and to set $\mathbf{y} \in \mathbb{R}^n$ as

$$y_i := \sum_{\mathbf{q} \in Q} s_k(\mathbf{x}_i|\mathbf{q}), \quad \forall i \in [n]. \tag{6.2}$$

[1]Imagine the case of multiple users querying at the same time; a different matrix per query is required. Also, updating mutual neighbors requires k-NN lists which are not available any longer.

Each dataset point $\mathbf{x}_i$ is assigned a scalar that is the sum of similarities over all query points $\mathbf{q}$ for which $\mathbf{x}_i$ appears in the corresponding k-nearest neighbor set $\mathrm{NN}_k(\mathbf{q})$, and zero if it appears in no such set.

Derivation. Our work is inspired by the analysis in the work of Grady [Gra06] that we apply to the diffusion mechanism of Zhou et al. [ZWG$^+$03], where query points Q are in the dataset. We decompose the quantities in (2.27) as $\mathbf{f} = (\mathbf{f}_d^\top, \mathbf{f}_q^\top)^\top$, with $\mathbf{f}_d \in \mathbb{R}^n$ and $\mathbf{f}_q \in \mathbb{R}^m$,

$$S = \begin{pmatrix} S_d & B_{dq} \\ B_{qd} & S_q \end{pmatrix}, \tag{6.3}$$

and $\mathbf{y} = (\mathbf{0}_n^\top, \mathbf{1}_m^\top)^\top$. Subscripts d, q denote data and query respectively. Then, (2.27) is written as

$$\mathbf{f}_d^t = \alpha S_d \mathbf{f}_d^{t-1} + \alpha B_{dq} \mathbf{f}_q^{t-1} \tag{6.4}$$

$$\mathbf{f}_q^t = \alpha B_{qd} \mathbf{f}_d^{t-1} + \alpha S_q \mathbf{f}_q^{t-1} + (1-\alpha)\mathbf{1}_m. \tag{6.5}$$

Provided this system converges, the data part satisfies

$$\mathbf{f}_d^\star \propto \mathcal{L}_\alpha^{-1} B_{dq} \mathbf{1}_m \tag{6.6}$$

if $\mathbf{f}_q^\star \propto \mathbf{1}_m$, $S_q = \mathbf{0}_{m\times m}$ and $B_{qd} = \mathbf{0}_{m\times n}$. In words, the query points are perfectly retrieved, they are dissimilar to each other, and the graph is indeed directed with query regions pointing to dataset regions, but the reverse is not allowed. Comparing (6.6) with (2.28), it follows that $B_{dq}\mathbf{1}_m$ is a good choice for $\mathbf{y}$. Since B_{dq} stores the similarities between the dataset and the query points, this analysis justifies the single query (6.1) and the multiple queries (6.2) cases.

Diffusion. Given this definition of $\mathbf{y}$, diffusion is now performed on dataset $\mathcal{X}$, jointly for all query points in Q. Affinities of multiple query points are propagated in the graph in a single process at no additional cost compared to the case of a single query point. Here we are excluding the additional cost of computing $\mathbf{y}$ itself in (6.2) compared to (6.1). This search takes place in all related work. We also do not discuss how to make this search more efficient in space and time [BL16], which is beyond the scope of this work.

Figure 6.1 illustrates the diffusion on single and multiple query points. The contour lines show the ranking score any point on the plane would be assigned given the query point(s). It is evident that multiple manifolds are captured when multiple queries are issued.

Pooling. After diffusion, each image is associated with several elements of the ranking score vector $\mathbf{f}^\star$, one for each point $\mathbf{x}$ in $X \subset \mathcal{X}$. A simple way to combine these scores is to define the score of image X as

$$f(X) = \sum_{j\in[m]} w_j f^\star_{i_X(j)}, \tag{6.7}$$

where $i_X(j)$ is the index of the j-th point of X in the dataset $\mathcal{X}$ and $\mathbf{w} = (w_j)$ a weighting vector. The latter is defined as $\mathbf{w} = \mathbf{1}_m$ for sum pooling and, assuming $m < d$,

$$\mathbf{w} = (\Phi\Phi^\top + \lambda I_m)^{-1}\mathbf{1}_m \tag{6.8}$$

for generalized max pooling (GMP) [MP14, IFG+17], where $\Phi = (\mathbf{x}_{i_X(1)}^\top, \dots, \mathbf{x}_{i_X(m)}^\top)^\top$ and $\lambda \in \mathbb{R}^+$ is a regularization parameter. Our experiments show that GMP always outperforms sum pooling.

6.2.3 Efficient Solution

Iteration (2.27) works well in practice but is slow at large scale. Taking the closed-form solution (2.28) literally, one may be tempted to compute the inverse $\mathcal{L}_\alpha^{-1}$ offline, but this matrix is not sparse like $\mathcal{L}_\alpha$. We propose a more efficient solution by making the connection to linear system solvers.

Diffusion is an iterative solver. Eq. (2.27) can be seen as an iteration of the Jacobi solver [Hac94]. Given a linear system $A\mathbf{x} = \mathbf{b}$[2], Jacobi decomposes A as $A = \Delta + R$ where $\Delta = \operatorname{diag}(A)$. It then iterates according to

$$\mathbf{x}^t = \Delta^{-1}(\mathbf{b} - R\mathbf{x}^{t-1}). \tag{6.9}$$

In our case, $\mathbf{x} = \mathbf{f}$, $\mathbf{b} = (1-\alpha)\mathbf{y}$, and $A = \mathcal{L}_\alpha = I - \alpha S$. It follows that $\Delta = I_n$ and $R = -\alpha S$, so that

$$\mathbf{f}^t = \alpha S \mathbf{f}^{t-1} + (1-\alpha)\mathbf{y}. \tag{6.10}$$

We have just re-derived (2.27). Note that a sufficient condition for Jacobi's convergence is that matrix A is strictly diagonally dominant, i.e. $|a_{ii}| > \sum_{j \neq i} a_{ij}$ for $i \in [n]$. It is easily checked that $\mathcal{L}_\alpha$ does satisfy this condition by construction, given that $0 < \alpha < 1$. This provides an alternative proof of the main result of Zhou et al. [ZBL+03].

Conjugate gradient (CG) [NW06] is the method of choice for solving linear systems like ours

$$\mathcal{L}_\alpha \mathbf{f} = (1-\alpha)\mathbf{y}, \tag{6.11}$$

where $\mathcal{L}_\alpha$ is positive-definite, and in particular for graph-related problems [Vis12]. It has been used for random walk problems [Gra06], but not diffusion-based retrieval according to our knowledge. In fact, the linear system formulation has been explicitly avoided in this context [DB13].

Here we argue, as in [LM04], that it is the solution of (6.11) that we seek, rather than the path followed by iteration (2.27). However, we use CG to approximate this solution, since matrix $\mathcal{L}_\alpha$ is indeed positive-definite. At each iteration, CG minimizes the quadratic function $\phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top A\mathbf{x} - \mathbf{x}^\top \mathbf{b}$ in a particular direction by analytically computing the optimal step length. More importantly, the direction chosen at each

[2]We adopt the standard linear system notation in this section; matrix A is not to be confused with our affinity matrix defined in (2.25).

iteration is conjugate to previous ones. Thus, any update of $\mathbf{x}$ along this direction does not destroy the optimality reached in the entire subspace considered thus far.

Contrary to other iterative methods including (6.10), CG is guaranteed to terminate in n steps. Remarkably, it provides good approximations in very few steps.

Normalization is preconditioning. Finally, a standard improvement is preconditioning, i.e. , solving a related system with A replaced by $C^{-1}AC^{-\top}$, a matrix satisfying a weak condition like its eigenvalues being clustered. Unfortunately, finding an appropriate matrix C can be quite complex [Vis12]. We observe that normalization (2.26) is preconditioning. Indeed, we could equally consider matrix $L_\alpha = D - \alpha A = \alpha L + (1-\alpha)I \succ 0$ and solve the linear system

$$L_\alpha(D^{-1/2}\mathbf{f}) = (1-\alpha)(D^{1/2}\mathbf{y}) \tag{6.12}$$

instead, which is equivalent to (6.11). By normalizing L_α into $\mathcal{L}_\alpha$, we are actually performing preconditioning with $C = \operatorname{diag}(L_\alpha)^{1/2}$. This is a simple form of symmetric preconditioning, known as diagonal scaling or Jacobi [TBI97]. It improves convergence, be it for CG or diffusion (2.27).

6.2.4 Scaling Up

Despite the efficient solution described in the previous section, there are still issues concerning space and offline pre-processing at large scale. We address these issues here.

Compact representation. At large scale, the number of region features per database image should be kept as low as possible. For this reason, we learn a Gaussian Mixture Model (GMM) on the original features of each database image and represent the image by the unit normalized means. This is an even more natural choice when dealing with overlapping regions (see Section 6.3). As a result, it decreases the number of region features and their redundancy.

The off-line construction of the affinity matrix is quadratic in the number of vectors in the database and might not be tractable at large scale. We employ the efficient and approximate k-NN graph construction method by Dong et al. [DCL11]. Section 6.3 shows that it is orders of magnitude faster than exhaustive search and has almost no effect on performance.

Truncating the affinity matrix. Instead of ranking the full dataset, diffusion re-ranks an initial search. This baseline in our experiments is done with global descriptors and kNN search. Then we apply diffusion only on the top ranked images. We truncate the affinity matrix keeping only the rows and columns related to the regions of the top ranked images and re-normalize it according to (2.26). The cost of this step is not significant compared to the actual diffusion.

Pooling	INSTRE	Oxf5k	Oxf105k	Par6k	Par106k
sum	79.1	92.2	90.6	96.1	94.4
GMP	80.0	93.2	91.6	96.5	94.6

Table 6.1: Retrieval performance (mAP) of regional diffusion with sum and generalized max pooling (GMP), with $\lambda = 1$ in (6.8).

6.3 Experiments

This section presents the experimental setup and investigates the accuracy of our methods for image retrieval compared with the state-of-the-art approaches.

6.3.1 Experimental Setup

Datasets. We use three datasets. Two are well-known image retrieval benchmarks: Oxford Buildings [PCI+07] and Paris [PCI+08]. We refer to them as Oxford5k and Paris6k. We experiment at large-scale by adding 100k distractor images from Flickr [PCI+07], forming Oxford105k and Paris106k datasets. The third corpus is the recently introduced instance search dataset called INSTRE [WJ15]. It contains various everyday 3D or planar objects from buildings to logos with many variations such as different scales, rotations, and occlusions. Some objects cover a small part of the image, making it a challenging dataset. It consists of 28,543 images from 250 different object classes. In particular, 100 classes with images retrieved from on-line sources, 100 classes with images taken by the dataset creators, and 50 classes consisting of pairs from the second category. We differentiate from the original protocol [WJ15], which uses all database images as queries. We randomly split the dataset into 1250 queries, 5 per class, and 27293 database images, while a bounding box defines the query region[3]. The query and the database sets have no overlap. We use mean average precision (mAP) as a performance measure in all datasets.

Representation. We employ a CNN that is fine-tuned for image retrieval [RTC16] to extract global and regional representation. In particular, this fine-tuned VGG produces 512 dimensional descriptors. We extract regions at 3 different scales as in R-MAC [TSJ16], while we additionally include the full image as a region. In this fashion, each image has on average 21 regions. The regional descriptors are aggregated and re-normalized to unit norm in order to construct the global descriptors, which is exactly as in R-MAC. We apply supervised whitening [RTC16] to both global and regional descriptors. We use this network to perform all our initial experiments. In Section 6.3.4, we also report scores with higher dimensional descriptors derived from the fine-tuned ResNet101 [GARL16b] using the same fixed grid.

[3]`http://people.rennes.inria.fr/Ahmet.Iscen/diffusion.html`

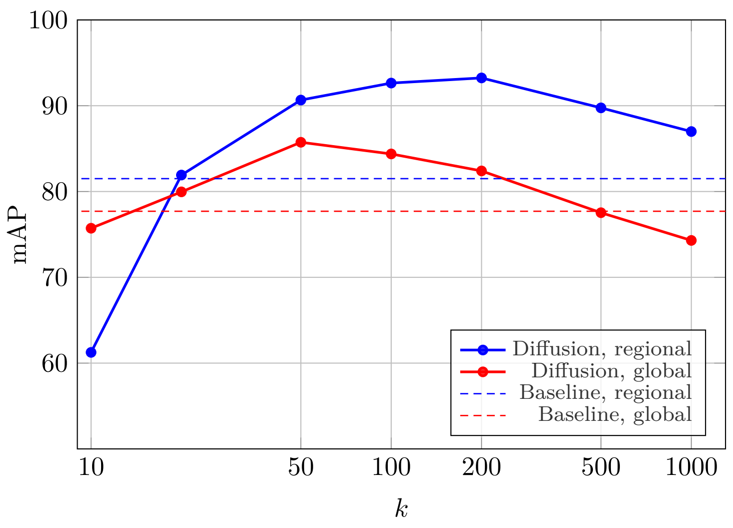


Figure 6.2: Impact of the number of nearest neighbors k in the affinity matrix. mAP performance for global and regional diffusion on Oxford5k; baselines are R-MAC and R-match respectively.

Implementation details. We define the affinity function using a monomial kernel [TAJ13] as $s(\mathbf{x}, \mathbf{z}) = \max(\mathbf{x}^\top \mathbf{z}, 0)^3$. The diffusion parameter α is always 0.99, as in the work of Zhou et al. [ZBL+03]. The k-NN search required by (6.2) is assumed to access all database vectors exhaustively. Our work does not investigate how approximate search methods [ML14, JDS11, KA14, BL16, IRF16] could improve time and space consumed by this process. After computing (6.2), we only keep the largest k values of $\mathbf{y}$ and set the rest to zero.

6.3.2 Impact of Different Components

Neighbors. We vary the number of nearest neighbors k for constructing the affinity matrix and evaluate performance for both global and regional diffusion. The global baseline method is k-NN search with R-MAC, while the regional one is the method by Razavian et al. [RSCM16], where image regions are indexed and cross-matched. We refer to the latter as R-match in the rest of our experiments.

Results for Oxford5k are presented in Figure 6.2, and are consistent in other datasets. The performance stays stable over a wide range of k. The drop for low k is due to very few neighbors being retrieved (where regional diffusion is more sensitive), whereas for high k, it is due to capturing more than the local manifold structure (where regional diffusion is superior). This behavior is consistent with the fact that small patterns appear more frequently than entire images.

We set $k = 200$ for regional diffusion, and $k = 50$ for global diffusion for the rest of the chapter. Since only mutual neighbors are linked, the actual number of edges per element is less: The average number of edges per image (resp. region) is 25 (resp.

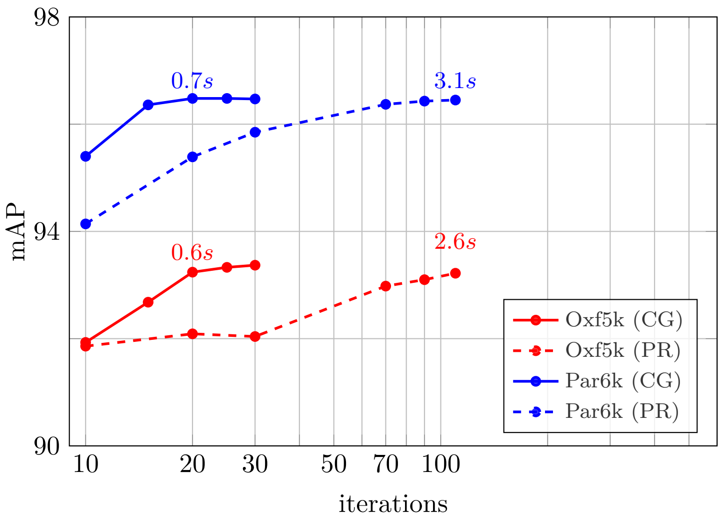


Figure 6.3: mAP performance of regional diffusion *vs.* number of iterations for conjugate gradient (CG) and iterative diffusion (2.27). Labels denote diffusion time.

75) for global (resp. regional) diffusion, measured on INSTRE. We set $k = 200$ for the query as well in the case of the regional diffusion, while for the global one $k = 10$ is needed to achieve good performance.

Pooling. We evaluate the two pooling strategies after regional diffusion in Table 6.1. Generalized max pooling has a small but consistent benefit in all datasets. We use this strategy for the rest of our experiments. Weights (6.8) are computed off-line and only one scalar per region is stored.

Efficient diffusion with conjugate gradient. We compare the iterative diffusion (2.27) to our conjugate gradient solution. We iterate each method until convergence. Performance is presented in Figure 6.3 with timings measured on a machine with a 4-core Intel Xeon 2.00GHz CPU. CG converges in as few as 20 iterations, which are also faster, while (2.27) reaches the same performance as CG only after 110 iterations.

The average query time on Oxford5k including all stages for global baseline, regional baseline, global diffusion and regional diffusion without truncation is 0.001s, 0.321s, 0.02s, and 0.664s, respectively.

Handling new queries. We compare our new way of handling new queries to the conventional approach that assumes queries to be part of the dataset. Our method achieves 80.0 mAP on INSTRE compared to 77.7 achieved by the conventional approach. We therefore not only offer space improvements but also better performance,mainly in the case of regional diffusion. The main difference is that k nonzero elements are kept both per query region (6.2) and for the entire vector $\mathbf{y}$. This, due to the overlapping nature of the CNN regions, may filter out incorrect neighbors.

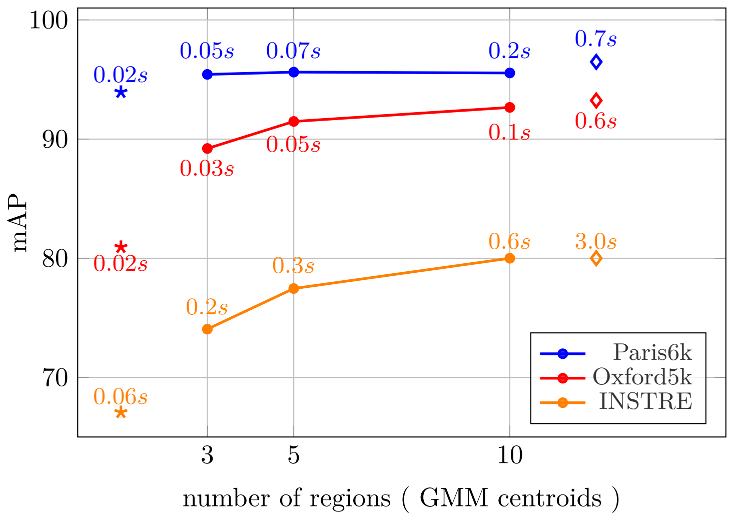


Figure 6.4: mAP performance for varying number of regional descriptors after learning a GMM per image. Symbol $\star$ denotes global diffusion, and $\diamond$ to the default number of regions (21) per image. Average diffusion time in seconds is shown in text labels.

6.3.3 Large-scale Diffusion

We now focus on the large scale solutions of Section 6.2.4.

Reduced number of regions. Figure 6.4 shows the impact of reducing the number of regions with Gaussian mixture models. Having as few as 5 descriptors per image already achieves competitive performance, while reducing the online search complexity. We decrease the number of neighbors k to 50 when GMM reduction is used, as there are now fewer positive neighbors.

Affinity matrix with Dong's algorithm [DCL11]. We compare the exhaustive construction of matrix A to Dong's efficient k-NN graph algorithm [DCL11]. Exhaustive search for Oxford105k composed of 2.2M regions takes 96 hours on a machine with a 12-core Intel Xeon 2.30GHz CPU. The approximate graph only takes 45 minutes and affects the final retrieval performance only slightly. It achieves 91.6 mAP on Oxford105k and 94.6 on Paris106k, while the exhaustive construction yields 92.5 and 95.2 respectively.

Truncation is a means to handle large scale datasets, i.e. more than 100k images. Regional diffusion on the full dataset takes 13.9s for Oxford105k, which is not practical. We therefore rank images according to the aggregated regional descriptors, which is equivalent to the R-MAC representation [TSJ16], and then perform diffusion on a short-list. Figure 6.5 reports results with truncation. The performance of the full database diffusion is nearly attained by re-ranking less than 10% of the database. The entire truncation and diffusion process on Oxford105k takes 1s, with truncation and

Method	$m \times d$	INSTRE	Oxf5k	Oxf105k	Par6k	Par106k
Global descriptors - nearest neighbor search						
CroW [KMO15]†	512	-	68.2	63.2	79.8	71.0
R-MAC [RTC16]	512	47.7	77.7	70.1	84.1	76.8
R-MAC [GARL16b]	2,048	62.6	83.9	80.8	93.8	89.9
NetVLAD [AGT$^+$16]†	4,096	-	71.6	-	79.7	-
Global descriptors - query expansion						
R-MAC [RTC16]+AQE [CPS$^+$07]	512	57.3	85.4	79.7	88.4	83.5
R-MAC [RTC16]+SCSM [SLBW14]	512	60.1	85.3	80.5	89.4	84.5
R-MAC [RTC16]+HN [QGB$^+$11]	512	64.7	79.9	-	92.0	-
Global diffusion	512	70.3	85.7	82.7	94.1	92.5
R-MAC [GARL16b]+AQE [CPS$^+$07]	2,048	70.5	89.6	88.3	95.3	92.7
R-MAC [GARL16b]+SCSM [SLBW14]	2,048	71.4	89.1	87.3	95.4	92.5
Global diffusion	2,048	80.5	87.1	87.4	96.5	95.4
Regional descriptors - nearest neighbor search						
R-match [RSCM16]	21×512	55.5	81.5	76.5	86.1	79.9
R-match [RSCM16]	21×2,048	71.0	88.1	85.7	94.9	91.3
Regional descriptors - query expansion						
HQE [TJ14]	2.4k×128	74.7	89.4†	84.0†	82.8†	-
R-match [RSCM16]+AQE [CPS$^+$07]	21×512	60.4	83.6	78.6	87.0	81.0
Regional diffusion*	5×512	77.5	91.5	84.7	95.6	93.0
Regional diffusion*	21×512	80.0	93.2	90.3	96.5	92.6
R-match [RSCM16]+AQE [CPS$^+$07]	21×2,048	77.1	91.0	89.6	95.5	92.5
Regional diffusion*	5×2,048	88.4	95.0	90.0	96.4	**95.8**
Regional diffusion*	21×2,048	**89.6**	**95.8**	**94.2**	**96.9**	95.3

Table 6.2: Performance comparison to the state of the art. Results from original publications are marked with †, otherwise they are based on our implementation. Our methods are marked with *. Points at 512D are extracted with VGG [RTC16] and at 2048D with ResNet101 [GARL16b]. Regional diffusion with 5 regions uses GMM.

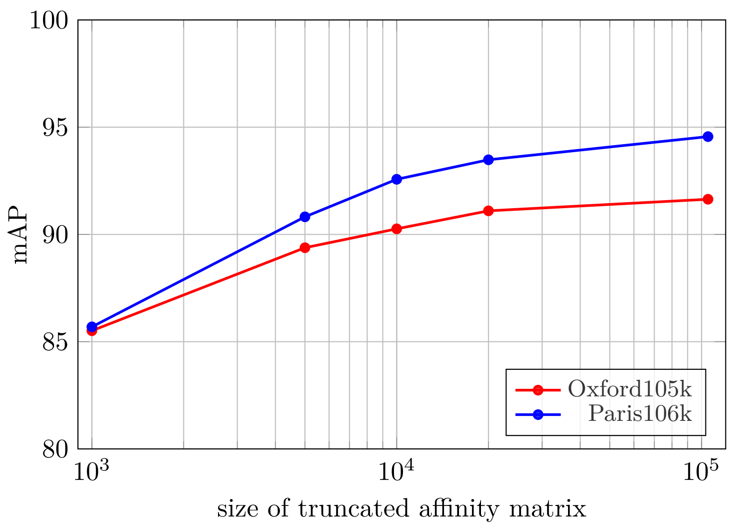


Figure 6.5: Retrieval performance (mAP) versus the shortlist size used for affinity matrix truncation.

re-normalization taking only a small part of it. In the following, search on Oxford105k and Paris105k is performed by truncating the top 10k images. This choice results in an affinity matrix A of around 200k regions. When GMM reduction is used, our short-list size is chosen so that A has 2M regions too, keeping re-ranking complexity fixed.

Our approach is scalable thanks to truncation: the shortlist length is fixed and so is the re-ranking time, regardless of the database size and the dimensionality of the descriptors. Although this shortlist contains a small fraction of the database, its

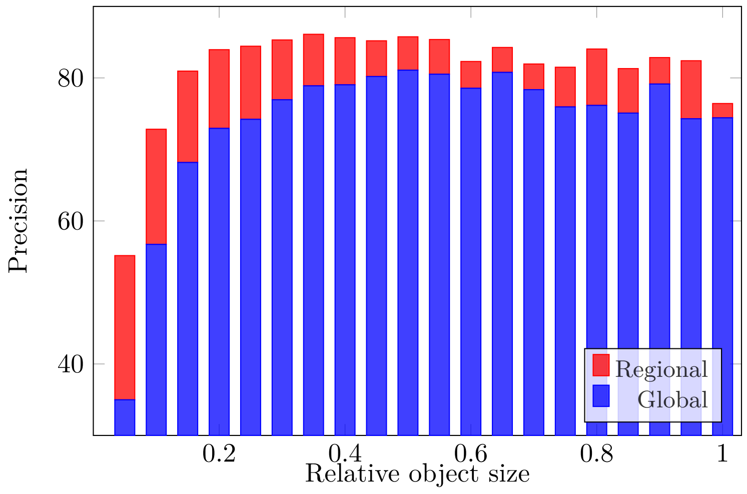


Figure 6.6: Precision of each positive image measured at the position where it was retrieved, averaged over positive images according relative object size. Statistics computed on INSTRE over all queries for global and regional diffusion.

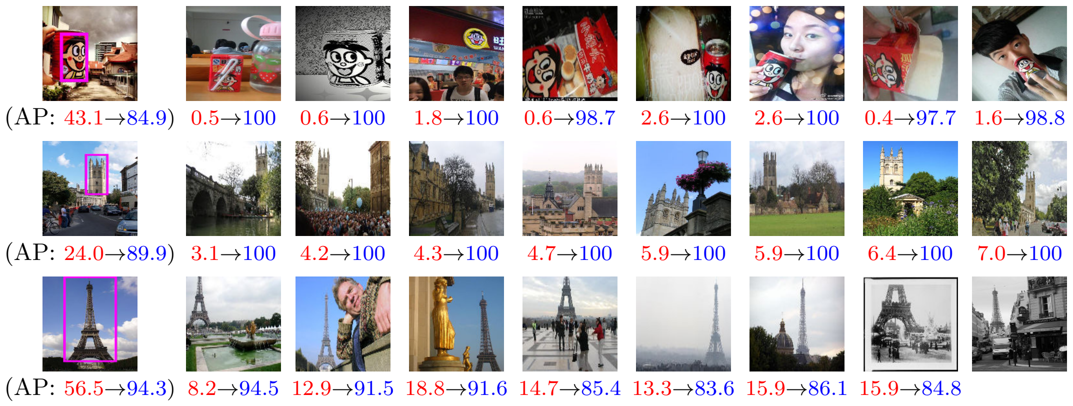

(AP: 43.1→84.9) 0.5→100 0.6→100 1.8→100 0.6→98.7 2.6→100 2.6→100 0.4→97.7 1.6→98.8
(AP: 24.0→89.9) 3.1→100 4.2→100 4.3→100 4.7→100 5.9→100 5.9→100 6.4→100 7.0→100
(AP: 56.5→94.3) 8.2→94.5 12.9→91.5 18.8→91.6 14.7→85.4 13.3→83.6 15.9→86.1 15.9→84.8

Figure 6.7: Query examples from INSTRE, Oxford, and Paris datasets and retrieved images ranked by decreasing order of ranking difference between global and regional diffusion. We measure precision at the position where each image is retrieved and report this under each image for global and regional diffusion. Average Precision (AP) is reported per query for the two methods.

significantly outperforms the baseline.

Small objects. We present quantitative and qualitative results revealing that images benefit from our method mainly when the depicted object is small and the scene is cluttered. Figure 6.7 shows that the retrieved images with the highest increase of precision of regional compared to global diffusion contain small objects that the latter cannot see. Since the bounding boxes are available for all images of INSTRE, we quantitatively measure precision for all positive images: Figure 6.6 shows that the highest improvement indeed comes for objects with small relative size.

6.3.4 Comparison to Other Methods

We compare with the state-of-the-art approaches with global or regional representation, with or without query expansion. Table 6.2 summarizes the results. We implement three methods typically combined with BoW, namely Average Query Expansion (AQE) [CPS+07], Spatially Constrained Similarity Measure (SCSM) [SLBW14] and Hello Neighbor (HN) [QGB+11]. AQE is also effective with CNN global representation [TSJ16, KMO15, GARL16a]. A baseline for the regional scenario is R-match [RSCM16]. We additionally extend AQE to regional representation[4] combined with the similarity used in R-match. Hamming Query Expansion[5] (HQE) [TJ14] is the only method not using CNNs, but local descriptors.

[4]AQE has not been proposed in a regional scenario. We extend it as competitive baseline derived from prior work.

[5]We evaluated HQE on INSTRE for the purposes of this work.

Regional diffusion significantly outperforms all other methods in all datasets. Global diffusion performs well on Paris because query objects almost fully cover the image in most of the database entries. This does not hold on INSTRE, which contains a lot of small objects. The improvements of regional diffusion are in this case much larger.

6.4 Conclusion

This chapter proposes a retrieval approach capturing distinct manifolds in the description space at no additional cost compared to a single query. We experimentally show that it significantly improves retrieval of small objects and cluttered scenes. The conclusion is that as few as 5-10 regional CNN descriptors can convey important information on small objects while thousands of conventional local descriptors are typically needed. Thus, a regional affinity matrix becomes possible. Regional diffusion was not possible before. In contrast to prior work, we use the closed form solution of the diffusion iteration, obtained by the conjugate gradient method. Combined with our contributions on space efficiency, this achieves large scale search at reasonable query times. Using recent CNN architectures, we achieve state-of-the-art and near optimal performance on two popular benchmarks and a recent more challenging dataset.

Chapter 7

Manifold Search: Fast Spectral Ranking

In Chapter 6, we make diffusion process efficient by solving a linear system online with conjugate gradients (CG), also handling multiple regions at the same cost. However, query times are still in the order of one second at large scale. In this chapter, our work does more work offline than graph construction and shows that query expansion is more than just "post-processing".

Given a dataset of size n and a query vector, a sparse observation vector $\mathbf{y} \in \mathbb{R}^n$ is first constructed based on a similarity search. The final ranking vector $\mathbf{x}^* \in \mathbb{R}^n$ results from a linear operator $T : \mathbb{R}^n \to \mathbb{R}^n$ applied on $\mathbf{y}$ [ITA+17]. Our work computes and stores a sparse rank-r approximation of T represented by $n \times r$ matrix Φ offline, such that $\mathbf{x} = \Phi\Phi^\top \mathbf{y} \approx \mathbf{x}^*$. This is obtained online by projecting $\mathbf{y}$ onto a r-dimensional embedding space followed by a search of the NN of $\Phi^\top \mathbf{y}$ among the n rows of Φ in $\mathbb{R}^r$. This is achieved without ever forming a dense representation for T. Figure 7.1 illustrates a 1d signal processing miniature of this idea. We make the following contributions:

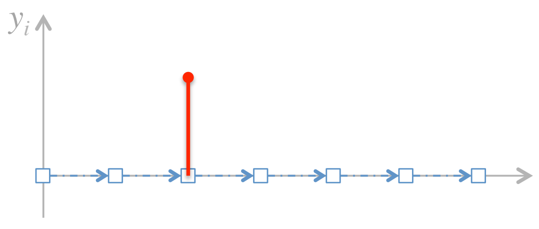
1. Image retrieval is cast as linear filtering over a graph, efficiently performed in the frequency domain.

2. A truly scalable solution computes an approximate Fourier basis of the graph offline, accompanied by performance bounds.

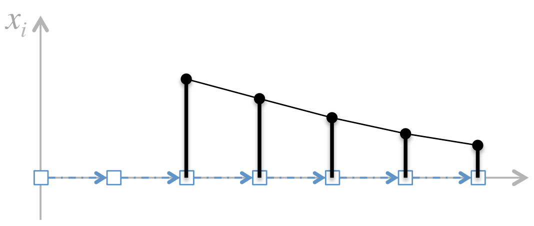
3. Manifold search is reduced to a two-stage similarity search thanks to an explicit embedding. This is useful in many other applications than retrieval.

4. A rich set of interpretations connects to different fields.

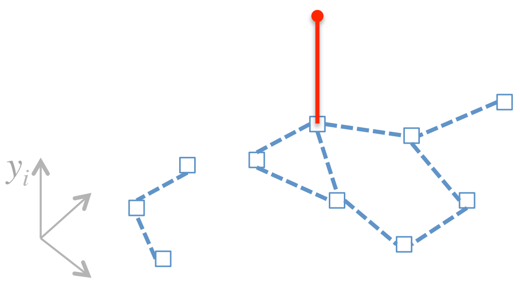
This work has been published in [Iscen et al., 2017a].



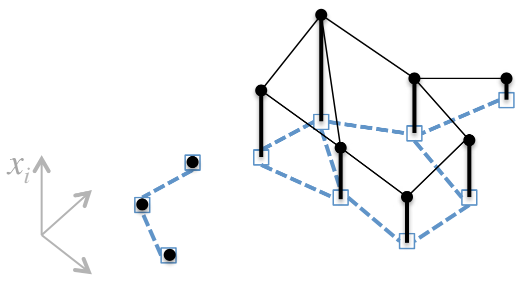
(a) Input signal $\mathbf{y}$



(b) Output signal $\mathbf{x}$



(c) Input signal $\mathbf{y}$



(d) Output signal $\mathbf{x}$

Figure 7.1: (a), (b) Output signal is $x_i := \beta \sum_{t=0}^{\infty} \alpha^t y_{i-t}$ with $\alpha \in [0, 1)$ and $\beta := 1 - \alpha$. This is a low-pass filter determined by $x_i = \alpha x_{i-1} + (1 - \alpha) y_i$, with impulse response $h_t = \beta \alpha^t u_t$ and transfer function $H(z) := \beta \sum_{t=0}^{\infty} (az^{-1})^t = \beta/(1 - \alpha z^{-1})$. This assumes a directed graph G with vertices $V = \mathbb{Z}$ and edges $E = \{(i, i + 1) : i \in \mathbb{Z}\}$, shown in blue. (c), (d) Using a weighted undirected graph G instead. Information "flows" in all directions, controlled by edge weights. In retrieval, the sample in red is the query, and the output x is its similarity to all samples.

7.1 Method

This section presents the problem and the proposed algorithm in abstract form, requiring as input a graph and a matrix function. Concrete choices relevant to image retrieval are discussed in the end.

7.1.1 Problem

We are given a weighted undirected graph $G = (V, E, w)$ with n vertices $V = \{v_1, \dots, v_n\}$, ℓ edges $E \subset V^2$, and a positive weight (similarity) function w. The graph is determined by its $n \times n$ symmetric nonnegative adjacency matrix W with elements $w_{ij} = w(v_i, v_j)$ if $(v_i, v_j) \in E$ and zero otherwise. Graph G contains no self-loops, i.e. W has zero diagonal. We assume W is sparse with $2\ell \leq kn$ nonzero elements for given $k \ll n$.

We define the $n \times n$ degree matrix $D := \operatorname{diag}(W\mathbf{1})$ where $\mathbf{1}$ is the all-ones vector, and the symmetrically normalized adjacency matrix $\mathcal{W} := D^{-1/2} W D^{-1/2}$ with the convention $0/0 = 0$. We also define the Laplacian and normalized Laplacian of G as $L := D - W$ and $\mathcal{L} := D^{-1/2} L D^{-1/2} = I - \mathcal{W}$, respectively. Both are singular and positive-semidefinite; the eigenvalues of $\mathcal{L}$ are in the interval $[0, 2]$ [Chu97]. Hence, if $\lambda_1, \dots, \lambda_n$ are the eigenvalues of $\mathcal{W}$, its spectral radius $\varrho(\mathcal{W}) := \max_i |\lambda_i|$ is 1. Each eigenvector $\mathbf{u}$ of L associated to eigenvalue 0 is constant within connected components (e.g. , $L\mathbf{1} = D\mathbf{1} - W\mathbf{1} = \mathbf{0}$), while the corresponding eigenvector of $\mathcal{L}$ is $D^{1/2}\mathbf{u}$.

We are also given a transfer function $h : \mathcal{S} \to \mathcal{S}$, where $\mathcal{S}$ is the set of real symmetric square matrices including scalars, $\mathbb{R}$. Our results hold for any function h under conditions given in section 7.2, but our standard choice is

$$h_\alpha(A) := (1 - \alpha)(I - \alpha A)^{-1} \tag{7.1}$$

parametrized by $\alpha \in [0, 1)$ and provided that $I - \alpha A$ is nonsingular. Accordingly, we define the $n \times n$ matrices $L_\alpha := \beta^{-1}(D - \alpha W)$ and $\mathcal{L}_\alpha := D^{-1/2} L_\alpha D^{-1/2} = \beta^{-1}(I - \alpha\mathcal{W})$, where $\beta := 1 - \alpha$. Both are positive-definite [ITA+17] and indeed $\mathcal{L}_\alpha^{-1} = h_\alpha(\mathcal{W})$.

Now, given a $n \times 1$ observation vector $\mathbf{y}$ on the graph vertices, the problem is to compute $n \times 1$ ranking vector

$$\mathbf{x}^* := h(\mathcal{W})\mathbf{y} \tag{7.2}$$

efficiently, in the sense that $h(\mathcal{W})$ is not explicitly computed or stored. For h_α in particular, we look for a more efficient solution than solving linear system

$$\mathcal{L}_\alpha \mathbf{x} = \mathbf{y} \tag{7.3}$$

as in [ITA+17]. The idea is that $\mathcal{W}$ and h are given in advance and we are allowed to pre-process them offline, while $\mathbf{y}$ ($\mathbf{x}$) is given (resp. computed) online. Moreover, $\mathbf{y}$ is sparse in practice as discussed in section 7.1.3.

7.1.2 Algorithm

We describe our algorithm given an arbitrary $n \times n$ matrix $A \in \mathcal{S}$ (instead of $\mathcal{W}$) and a transfer function h. Our solution is based on a sparse low-rank approximation of A computed offline such that online, $\mathbf{x} \approx h(A)\mathbf{y}$ is reduced to a sequence of sparse matrix-vector multiplications. The approximation is based on a randomized algorithm [RST09] that is similar to Nyström sampling [DM05] but comes with performance guarantees [HMT11, WC13]. In the following, $r \ll n$, $p < r$, q and τ are given parameters, and $\hat{r} = r + p$.

1. (Offline) Using simultaneous iteration [TBI97], compute an $n \times \hat{r}$ matrix Q with orthonormal columns that represents an approximate basis for the range of A, i.e. $QQ^\top A \approx A$. In particular, this is done as follows [HMT11]: randomly draw an $n \times \hat{r}$ standard Gaussian matrix $B^{(0)}$ and repeat for $t = 0, \dots, q-1$:

   (a) Compute QR factorization $Q^{(t)}R^{(t)} = B^{(t)}$.

   (b) Define the $n \times \hat{r}$ matrix $B^{(t+1)} := AQ^{(t)}$.

   Finally, set $Q := Q^{(q-1)}$, $B := B^{(q)} = AQ$.

2. (Offline) Compute an approximate rank-r eigenvalue decomposition $U\Lambda U^\top \approx A$, where $n \times r$ matrix U has orthonormal columns and $r \times r$ matrix Λ is diagonal. In particular, roughly following [HMT11]:

   (a) Form the $\hat{r} \times \hat{r}$ matrix $C := Q^\top B = Q^\top AQ$.

   (b) Compute its eigendecomposition $\hat{V}\hat{\Lambda}\hat{V}^\top = C$.

   (c) Form (V, Λ) by keeping from $(\hat{V}, \hat{\Lambda})$ the slices (rows/columns) corresponding to the r largest eigenvalues.

   (d) Define the matrix $U := QV$.

3. (Offline) Make U sparse by keeping its τ largest entries and dropping the rest.

4. (Online) Given $\mathbf{y}$, compute
$$\mathbf{x} := Uh(\Lambda)U^\top \mathbf{y}. \tag{7.4}$$

Observe that $U^\top$ projects $\mathbf{y}$ onto $\mathbb{R}^r$. With Λ being diagonal, $h(\Lambda)$ is computed elementwise, so h can be given online. Finally, multiplying by U and ranking $\mathbf{x}$ amounts to dot product similarity search in $\mathbb{R}^r$.

7.1.3 Retrieval

The above algorithm solves any problem of the form (7.2) satisfying our assumptions. We now apply it to image retrieval, following [ITA+17]. We are given a dataset of N images, each represented by m region descriptors in $\mathbb{R}^d$. Global image description is just the special case $m = 1$. The entire dataset is represented by a total of $n = Nm$ descriptors $\mathcal{V} = \{\mathbf{v}_1, \dots, \mathbf{v}_n\}$, with each $\mathbf{v}_i$ associated to vertex v_i of G. Descriptors are obtained by sampling a set of rectangular regions on CNN activations of a network, followed by pooling, normalization, PCA, whitening, and optionally reduced by a Gaussian mixture model [ITA$^+$17].

Given descriptors $\mathbf{v}, \mathbf{z} \in \mathbb{R}^d$, we measure their similarity by $s(\mathbf{v}, \mathbf{z}) = (\mathbf{v}^\top \mathbf{z})^\gamma$, where exponent $\gamma > 0$ is a parameter. We denote by $s(\mathbf{v}_i|\mathbf{z})$ the similarity $s(\mathbf{v}_i, \mathbf{z})$ if $\mathbf{v}_i$ is a k-NN of $\mathbf{z}$ in $\mathcal{V}$ and zero otherwise. The sparse $n \times n$ similarity matrix S is defined element-wise as $s_{ij} := s(\mathbf{v}_i|\mathbf{v}_j)$, and the symmetric adjacency matrix W as $w_{ij} := \min(s_{ij}, s_{ji})$, representing mutual neighborhoods.

Given a query image represented by descriptors $\{\mathbf{q}_1, \dots, \mathbf{q}_m\} \subset \mathbb{R}^d$, we form the observation vector $\mathbf{y} \in \mathbb{R}^n$ with elements $y_i := \sum_{j=1}^m s(\mathbf{v}_i|\mathbf{q}_j)$ by pooling over query regions. We make $\mathbf{y}$ sparse by keeping the k largest entries and dropping the rest, and compute the ranking vector $\mathbf{x} \in \mathbb{R}^n$ by (7.4), containing the ranking score x_i of each dataset region $\mathbf{v}_i$. To obtain a score per image, we perform a linear pooling operation [ITA$^+$17] represented as $\overline{\mathbf{x}} := \Sigma\mathbf{x}$ where Σ is a sparse $N \times n$ pooling matrix. The $N \times r$ matrix $\overline{U} := \Sigma U$ is indeed computed offline so that we directly compute $\overline{\mathbf{x}} = \overline{U} h(\Lambda) U^\top \mathbf{y}$ online.

Computing $\mathbf{y}$ involves Euclidean search in $\mathbb{R}^d$, which happens to be dot product because vectors are ℓ^2-normalized. Applying $\overline{U}$ and ranking $\mathbf{x}$ amounts to a dot product similarity search in $\mathbb{R}^r$. We thus reduce manifold search to Euclidean followed by dot product search. The number of nonzero elements of $\mathbf{y}$ and rows of $\overline{U}$, whence the cost, are the same for global or regional search.

7.2 Analysis

We refer to our algorithm as fast spectral ranking (FSR) with the following variants:

- FSR.sparse: This is the complete algorithm.
- FSR.approx: Drop sparsification stage 3.
- FSR.rank-r: Drop approximation stage 1 and sparsification stage 3. Set $\hat{r} = n$, $Q = I$, $B = A$ in stage 2.
- FSR.exact: same as FSR.rank-r for $r = n$.

Given observation vector $\mathbf{y} \in \mathbb{R}^n$, $n \times n$ matrix $A \in \mathcal{S}$, and function h, we denote by $\mathbf{x} = \text{FSR}.\mathsf{v}(\mathbf{y}|A, h) \in \mathbb{R}^n$ the vector obtained by variant v of the algorithm. We consider here functions h for which there exists a series expansion

$$h(A) = \sum_{t=0}^{\infty} c_t A^t \tag{7.5}$$

for $A \in \mathcal{S}$. We denote by $\mathcal{H}$ the family of such functions.

7.2.1 Correctness

Algorithm FSR.exact is not useful: U is a dense $n \times n$ matrix in this case and even if space is not an issue, one is still better off solving linear system (7.3) as in [ITA$^+$17] rather than using (7.4). However, showing its correctness is instructive in light of its interpretation in section 7.3.

Theorem 1 If $h \in \mathcal{H}$ and series (7.5) converges, then FSR.exact$(\mathbf{y}|A, h) = h(A)\mathbf{y}$. In particular, FSR.exact$(\mathbf{y}|\mathcal{W}, h_\alpha) = \mathbf{x}^*$.

Proof: Matrix A is real symmetric hence diagonalizable, and FSR.exact computes its exact eigenvalue decomposition $U\Lambda U^\top = A$, where U is orthogonal. Because the series converges, $h(A) = Uh(\Lambda)U^\top = U \operatorname{diag}(h(\lambda_1), \dots, h(\lambda_n))U^\top$ [AM05]. In particular, $h_\alpha \in \mathcal{H}$ and has the geometric progression expansion

$$h_\alpha(A) := \beta(I - \alpha A)^{-1} = \beta \sum_{t=0}^{\infty} (\alpha A)^t, \tag{7.6}$$

which converges absolutely if $\varrho(\alpha A) < 1$ [AM05]. This holds for $A = \mathcal{W}$ because $\alpha < 1$ and $\varrho(\mathcal{W}) = 1$.

7.2.2 Complexity

The offline complexity is mainly determined by the number of columns $\hat{r}$ of matrix Q: Stage 1 reduces the size of the problem from n^2 down to $n\hat{r}$. The online complexity is determined by the number of nonzero entries in matrix U. A straightforward analysis leads to the following:

- FSR.approx: The offline complexity is $O(qn(k + \hat{r})\hat{r})$ time and $O(n\hat{r})$ space; its online (time and space) complexity is $O(nr)$.
- FSR.sparse: The offline complexity is $O(qn(k + \hat{r})\hat{r} + \tau \log \tau)$ time and $O(n\hat{r})$ space; its online complexity is $O(\tau)$.

Stage 1 is "embarrassingly parallelizable" meaning that it is dramatically accelerated on parallel and distributed platforms. Since the online stage 4 amounts to NN search, any approximate method applies, making it sublinear in n.

7.2.3 Error Bound

We present main ideas for bounding the approximation error of FSR.rank-r and FSR.approx coming from literature. The approximation $QQ^\top A \approx A$ of stage 1 is studied in [HMT11]: an average-case bound on $\left\| A - QQ^\top A \right\|$ decays exponentially fast in the number of iterations q to $|\lambda_{r+1}|$. Stage 2 yields an approximate eigenvalue decomposition of A: Since A is symmetric, $A \approx QQ^\top AQQ^\top = QCQ^\top \approx QV\Lambda V^\top Q^\top = U\Lambda U^\top$. The latter approximation $C \approx V\Lambda V^\top$ is essentially a best rank-r approximation of

$C = Q^\top AQ$. This is also studied in [HMT11] for the truncated SVD case of a non-symmetric matrix. It involves an additional term of $|\lambda_{r+1}|$ in the error.

We are actually approximating $h(A)$ by $Uh(\Lambda)U^\top$, so that $|h(\lambda_{r+1})|$ governs the error instead of $|\lambda_{r+1}|$. A similar situation appears in [TFP06]. Therefore, our method makes sense only when the restriction of h to scalars is nondecreasing. This is the case for h_α.

7.3 Interpretation

Our work is connected to studies in different fields with a long history. Here we give here a number of interpretations both in general and to the particular case $h = h_\alpha$.

7.3.1 Random Walks

Consider the iterating process: for $t = 1, 2, \dots$

$$\mathbf{x}^{(t)} := \alpha A\mathbf{x}^{(t-1)} + (1-\alpha)\mathbf{y}. \tag{7.7}$$

If A is a stochastic transition matrix and $\mathbf{x}^{(0)}, \mathbf{y}$ are distributions over vertices, this specifies a random walk on a (directed) graph: at each iteration a particle moves to a neighboring vertex with probability α or jumps to a vertex according to distribution $\mathbf{y}$ with probability $1-\alpha$. This is called a Markov chain with restart [BLSV06] or random walk with restart [PYFD04]. State $\mathbf{x}^{(t)}$ converges to $\mathbf{x}^* = h_\alpha(A)\mathbf{y}$ as $t \to \infty$ provided $\varrho(\alpha A) < 1$ [ZBL$^+$03]. In fact, (7.7) is equivalent to Jacobi solver [Hac94] on linear system (7.3) [ITA$^+$17].

If $\mathbf{y} = \mathbf{e}_i$, the i-th canonical vector, then $\mathbf{x}^*$ is used to rank the vertices of G, expressing a measure of "similarity" to v_i [ZWG$^+$03]. Parameter α controls how much $\mathbf{x}^*$ is affected by boundary condition $\mathbf{y}$ [Vig09]: $\mathbf{x}^*$ equals $\mathbf{y}$ for $\alpha = 0$, while in the limit $\alpha \to 1$, $\mathbf{x}^*$ tends to a dominant eigenvector of A. Indeed, for $\alpha = 1$, (7.7) becomes a power iteration.

7.3.2 Random Fields

Given a positive-definite $n \times n$ precision matrix $A \in \mathcal{S}$ and a mean vector $\boldsymbol{\mu} \in \mathbb{R}^n$, a Gaussian Markov random field (GMRF) [RH05] with respect to an undirected graph G is a random vector $\mathbf{x} \in \mathbb{R}^n$ with normal density $p(\mathbf{x}) := \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, A^{-1})$ iff A has the same nonzero off-diagonal entries as the adjacency matrix of G. Its canonical parametrization $p(\mathbf{x}) \propto e^{-E(\mathbf{x}|\mathbf{b},A)}$ where $E(\mathbf{x}|\mathbf{b}, A) := \frac{1}{2}\mathbf{x}^\top A\mathbf{x} - \mathbf{b}^\top\mathbf{x}$ is a quadratic energy. Its expectation $\boldsymbol{\mu} = A^{-1}\mathbf{b}$ is the minimizer of this energy. Now, $\mathbf{x}^* = \mathcal{L}_\alpha^{-1}\mathbf{y}$ (7.3) is the expectation of a GMRF with energy

$$f_\alpha(\mathbf{x}) := E(\mathbf{x}|\mathbf{y}, \mathcal{L}_\alpha) = \frac{1}{2}\mathbf{x}^\top\mathcal{L}_\alpha\mathbf{x} - \mathbf{y}^\top\mathbf{x}. \tag{7.8}$$

A mean field method on this GMRF is equivalent to Jacobi or Gauss-Seidel solvers on (7.3) [WJ08]. Yet, conjugate gradients (CG) [NW06] is minimizing $f_\alpha(\mathbf{x})$ more efficiently [ITA$^+$17, CK16].

If we expand $f_\alpha(\mathbf{x})$ using $\beta \mathcal{L}_\alpha = \alpha \mathcal{L} + (1-\alpha) I$, we find that it has the same minimizer as

$$\alpha \sum_{i,j} w_{ij} \left\| \hat{x}_i - \hat{x}_j \right\|^2 + (1-\alpha) \left\| \mathbf{x} - \mathbf{y} \right\|^2 , \tag{7.9}$$

where $\hat{\mathbf{x}} := D^{-1/2}\mathbf{x}$. The pairwise smoothness term encourages $\mathbf{x}$ to vary little across edges with large weight whereas the unary fitness term to stay close to observation $\mathbf{y}$ [ZBL+03]. Again, α controls the trade-off: $\mathbf{x}^*$ equals $\mathbf{y}$ for $\alpha = 0$, while for $\alpha \to 1$, $\mathbf{x}^*$ tends to be constant over connected components like dominant eigenvectors of $\mathcal{W}$.

7.3.3 Regularization and Kernels

The first term of (7.8) is interpreted as a regularization operator related to a kernel $K = \mathcal{L}_\alpha^{-1}$ [SSM98, SK03, KV04]. In a finite graph, a kernel can be seen either as an $n \times n$ matrix K or a function $\kappa : V^2 \to \mathbb{R}$ operating on pairs of vertices. More generally, if $h(x) > 0$ for $x \in \mathbb{R}$, which holds for h_α, then $K := h(\mathcal{W})$ is positive-definite and there is an $n \times n$ matrix Φ such that $K = \Phi^\top \Phi$, or $\kappa(v_i, v_j) = \phi(v_i)^\top \phi(v_j)$ where feature map $\phi : V \to \mathbb{R}^n$ is given by $\phi(v_i) := \Phi \mathbf{e}_i$. A particular choice for Φ is

$$\Phi := h(\Lambda)^{1/2} U^\top \tag{7.10}$$

where $U \Lambda U^\top$ is the eigenvalue decomposition of $\mathcal{W}$. If we choose a rank-r approximation instead, then Φ is an $r \times n$ matrix and ϕ is a low-dimensional embedding onto $\mathbb{R}^r$.

The goal of out-of-sample extension is to compute a "similarity" $\hat{\kappa}(\mathbf{z}_1, \mathbf{z}_2)$ between two unseen vectors $\mathbf{z}_1, \mathbf{z}_2 \in \mathbb{R}^d$ not pertaining to the graph. Here we define

$$\hat{\kappa}(\mathbf{z}_1, \mathbf{z}_2) := \psi(\mathbf{z}_1)^\top \Phi^\top \Phi \psi(\mathbf{z}_2) \tag{7.11}$$

given any mapping $\psi : \mathbb{R}^d \to \mathbb{R}^n$, e.g. $\psi(\mathbf{z})_i := s(\mathbf{v}_i | \mathbf{z})$ discussed in section 7.1.3. This extended kernel is also positive-definite and its embedding $\hat{\phi}(\mathbf{z}) = \Phi \psi(\mathbf{z})$ is a linear combination of the dataset embeddings. For $r \ll n$, our method allows rapid computation of κ or $\hat{\kappa}$ for any given function h, without any dense $n \times n$ matrix involved.

7.3.4 Paths on Graphs

Many nonlinear dimension reduction methods replace Euclidean distance with an approximate geodesic distance, assuming the data lie on a manifold [LV07]. This involves the all-pairs shortest path (APSP) problem and Dijkstra's algorithm is a common choice. Yet, it is instructive to consider a naïve algorithm [CLRS09]. We are given a distance matrix where missing edges are represented by ∞ and define similarity weight $w_{ij} = e^{-d_{ij}}$. A path weight is a now a product of similarities and "shortest" means "of maximum weight". Defining matrix power $A^{\otimes t}$ as A^t with $+$ replaced by max, the algorithm is reduced to computing $\max_t W^{\otimes t}$ (element-wise). Element i, j of $W^{\otimes t}$ is the weight of the shortest path of length t between v_i, v_j.

Besides their complexity, shortest paths are sensitive to changes in the graph. An alternative is the sum[1] of weights over paths of length t, recovering the ordinary matrix

[1] In fact, similar to *softmax* due to the exponential and normalization.

power $\mathcal{W}^t$, and the weighted sum over all lengths $\sum_{t=0}^{\infty} c_t \mathcal{W}^t$, where coefficients $(c_t)_{t\in\mathbb{N}}$ allow for convergence [Vig09], [STC04]. This justifies (7.5) and reveals that coefficients control the contribution of paths depending on length. A common choice is $c_t = \beta\alpha^t$ with $\beta = 1 - \alpha$ and $\alpha \in [0, 1)$ being a damping factor [Vig09], which justifies function h_α (7.6).

7.3.5 Graph Signal Processing

In signal processing [OS10], a discrete-time signal of period n is a vector $\mathbf{s} \in \mathbb{R}^n$ where indices are represented by integers modulo n, that is, $s_{\bar{i}} := s_{(i \bmod n)+1}$ for $i \in \mathbb{Z}$. A shift (or translation, or delay) of $\mathbf{s}$ by one sample is the mapping $s_{\bar{i}} \mapsto s_{\overline{i-1}}$. If we define the $n \times n$ circulant matrix $C_n := (\mathbf{e}_2\ \mathbf{e}_3 \ldots \mathbf{e}_n\ \mathbf{e}_1)^2$, a shift can be represented by $\mathbf{s} \mapsto C_n\mathbf{s}$ [SM13]. A linear, time (or shift) invariant filter is the mapping $\mathbf{s} \mapsto H\mathbf{s}$ where H is an $n \times n$ matrix with a series representation $H := h(C_n) = \sum_{t=0}^{\infty} h_t C_n^t$. Matrix C_n has the eigenvalue decomposition $U\Lambda U^\top$ where $U^\top$ is the $n \times n$ discrete Fourier transform matrix $\mathcal{F}$. If the series $h(C_n)$ converges, filtering $\mathbf{s} \mapsto H\mathbf{s}$ is written as

$$\mathbf{s} \mapsto \mathcal{F}^{-1} h(\Lambda) \mathcal{F}\mathbf{s}. \tag{7.12}$$

That is, $\mathbf{s}$ is mapped to the frequency domain, scaled element-wise, and mapped back to the time domain.

Graph signal processing [SM13, SNF$^+$13] generalizes the above concepts to graphs by replacing C_n by $\mathcal{W}$, an appropriately normalized adjacency matrix of an arbitrary graph. If $U\Lambda U^\top$ is the eigenvalue decomposition of $\mathcal{W}$, we realize that (7.4) treats $\mathbf{y}$ as a (sparse) signal and filters it in the frequency domain via transfer function h to obtain $\mathbf{x}$. Function h_α in particular is a low-pass filter that reconstructs $\mathbf{x}$ even from a single nonzero sample $\mathbf{y}$ over the graph. By varying α from 0 to 1, the frequency response varies from all-pass to sharp low-pass, allowing only the DC component.

7.4 Related Work

The history of the particular case $h = h_\alpha$ is the subject of the excellent study of spectral ranking [Vig09]. The fundamental contributions originate in the social sciences and include the eigenvector formulation by Seeley [See49], damping by α (7.6) by Katz [Kat53] and the boundary condition $\mathbf{y}$ (7.3) by Hubbell [Hub65]. The most well-known follower is PageRank [PBMW99]. In machine learning, h_α has been referred to as the von Neumann [KSTC02, STC04] or regularized Laplacian kernel [SK03]. Along with the diffusion kernel [KL02, KV04], it has been studied in connection to regularization [SSM98, SK03].

Random fields are routinely used for low-level vision tasks where one is promoting smoothness while respecting a noisy observation, like in denoising or segmentation, where both the graph and the observation originate from a single image [TLAF07,

[2]Observe that C_n is the adjacency matrix of the directed graph of Figure 7.1 after adding an edge from the rightmost to the leftmost vertex.

CK16]. A similar mechanism appears in semi-supervised learning [ZBL$^+$03, ZLG03, ZGL03, CWS03] or interactive segmentation [Gra06, KLL08] where the observation is composed of labels over a number of samples or pixels. In our retrieval scenario, the observation is formed by the neighbors in the graph of an external query image (or its regions).

The random walk or random walk with restart (RWR) formulation [ZWG$^+$03, ZBL$^+$03, PYFD04] is an alternative interpretation to retrieval [DB13]. Yet, directly solving a linear system is superior [ITA$^+$17]. Offline matrix decomposition has been studied for RWR [TFP06, FNOK12, JSSK16]. All three methods are limited to h_α while sparse LU decomposition [FNOK12, JSSK16] assumes an uneven distribution of vertex degrees [KF11], which is not the case for k-NN graphs. In addition, we reduce manifold search to two-stage Euclidean search via an explicit embedding, which is data dependent throught the kernel $K = \mathcal{L}_\alpha^{-1}$.

In the general case, the spectral formulation (7.4) has been known in machine learning [CWS03, STC04, NLCK05, ZKLG06, VSKB10] and in graph signal processing [SM13, SNF$^+$13, HVG11]. The latter is becoming popular in the form of graph-based convolution in deep learning [BZSL13, HBL15, DBV16, BBL$^+$16, MBM$^+$16, PKP17]. However, with few exceptions [BZSL13, HBL15], which rely on an expensive decomposition, there is nothing spectral when it comes to actual computation. It is rather preferred to work with finite polynomial approximations of the graph filter [DBV16, BBL$^+$16] using Chebyshev polynomials [HVG11, SVF11] or translation-invariant neighborhood templates in the spatial domain [MBM$^+$16, PKP17].

We cast retrieval as graph filtering by constructing an appropriate observation vector. We actually perform the computation in the frequency domain via a scalable solution. Comparing to other applications, retrieval conveniently allows offline computation of the graph Fourier basis and online reuse to embed query vectors. An alternative is to use random projections [TB14, RTB$^+$15]. This roughly corresponds to a single iteration of our step 1. Our solution is thus more accurate, while h can be specified online.

7.5 Practical Considerations

Block diagonal case. Each connected component of G has a maximal eigenvalue 1. These maxima of small components dominate the eigenvalues of the few (or one) "giant" component that contain the vast majority of data [KF11]. For this reason we find the connected components with the union-find algorithm [CLRS09] and reorder vertices such that A is block diagonal: $A = \operatorname{diag}(A_1, \dots, A_c)$. For each $n_l \times n_l$ matrix A_l, we apply offline stages 1-3 to obtain an approximate rank-r_l eigenvalue decomposition $\hat{U}_l \hat{\Lambda}_l \hat{U}_l^\top \approx A_l$ with $r_l = \max(\rho, \lceil r n_l / n \rceil)$ if $n_l > \rho$, otherwise we compute an exact decomposition. Integer ρ is a given parameter. We form (U_l, Λ_l) by keeping up to ρ slices from each pair $(\hat{U}_l, \hat{\Lambda}_l)$ and complete with up to r slices in total, associated to the largest eigenvalues of the entire set $\operatorname{diag}(\hat{\Lambda}_1, \dots, \hat{\Lambda}_c)$. Online, we partition $(\mathbf{y}_1; \dots; \mathbf{y}_c) = \mathbf{y}$, compute each $\mathbf{x}_l$ from $\mathbf{y}_l$ by (7.4) and form back vector $\mathbf{x} = (\mathbf{x}_1; \dots; \mathbf{x}_c)$.

Sparse neighborhoods. Denote by η_i the ℓ_2-norm of the i-th row of U. FSR.exact yields $\boldsymbol{\eta} = \mathbf{1}$ but this is not the case for FSR.approx. Larger (smaller) values appear to correspond to densely (sparsely) populated parts of the graph. For small rank r, norms η_i are more severely affected for uncommon than common vectors in the dataset. We propose replacing each element x_i of (7.4) by

$$x'_i = x_i + (1 - \eta_i)\mathbf{v}_i^\top \mathbf{q}, \tag{7.13}$$

for global descriptors, with a straightforward extension for regional ones. This is referred to as FSRw.approx and is a weighted combination of manifold search and Euclidean search. It approaches the former for common elements and to the latter for uncommon ones. Our experiments show that this is essential at large scale.

7.6 Experiments

This section introduces our experimental setup, investigates the performance and behavior of the proposed method and its application to large-scale image retrieval.

7.6.1 Experimental Setup

Datasets. We use three image retrieval benchmarks: Oxford Buildings (Oxford5k) [PCI+07], Paris (Paris6k) [PCI+08] and Instre [WJ15], with the evaluation protocol introduced in [ITA+17] for the latter. We conduct large-scale experiments by following a standard protocol of adding 100k distractor images from Flickr [PCI+07] to Oxford5k and Paris6k, forming the so called Oxford105k and Paris106k. Mean average precision (mAP) evaluates the retrieval performance in all datasets.

Image Descriptors. We apply our method on the same global and regional image descriptors as in [ITA+17]. Global description is R-MAC with 3 different scales [TSJ16], including the full image as a separate region. Regional descriptors consist of the same regions as those involved in R-MAC but without sum pooling, resulting in 21 vectors per image on average. Global and regional descriptors are processed by supervised whitening [RTC16]. In particular we work with d-dimensional vectors extracted from VGG [SZ14] ($d = 512$) and ResNet101 [HZRS16] ($d = 2,048$) networks fine-tuned specifically for image retrieval [RTC16, GARL16b].

Implementation . We adopt the same parameters for graph construction and search as in [ITA+17]. A monomial kernel is used for pairwise descriptor similarity, i.e. $s(\mathbf{v}, \mathbf{z}) = (\mathbf{v}^\top \mathbf{z})^3$. We use $\alpha = 0.99$, and keep the top $k = 50$ and $k = 200$ mutual neighbors when constructing the graph for global and regional vectors, respectively. These choices make our experiments directly comparable to prior results on manifold search for image retrieval with CNN-based descriptors [ITA+17]. In all our FSR.approx experiments, we limit the algorithm within the largest connected component only, while each element x_i for vertex v_i in some other component is just copied from y_i.

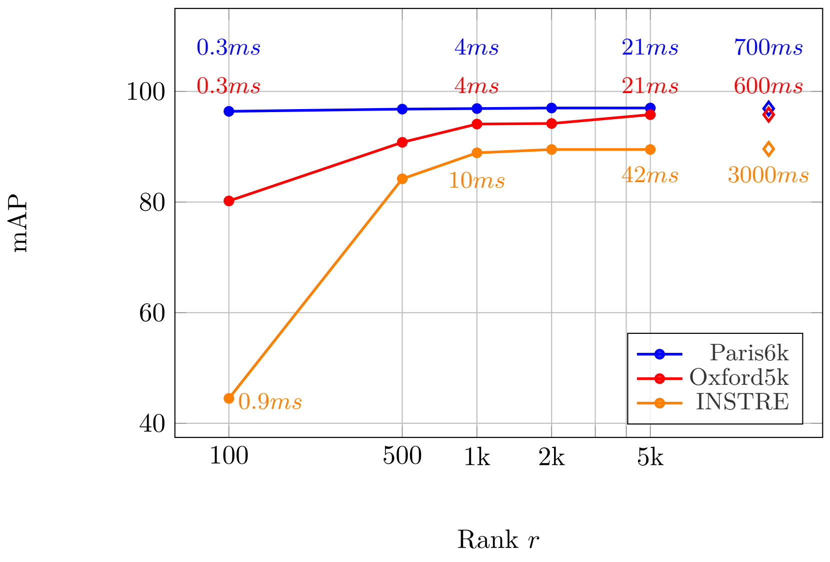


Figure 7.2: Performance of regional search with FSR.RANK-r. Runtimes are reported in text labels. $\diamond$ refers to FSR.EXACT performed with conjugate gradients as in [ITA+17]

This choice works well because the largest component holds nearly all data in practice. Time measurements are reported with a 4-core Intel Xeon 2.00GHz CPU. Runtimes refer to the search time excluding the construction of the observation vector, since this task is common to all baseline and our methods.

7.6.2 Retrieval Performance

Rank-r. We analyze the performance of FSR.rank-r for varying values of r. Parameter r affects the quality of the approximation and defines the dimensionality of the embedding space. The initial vectors in $\mathbb{R}^d$ are projected to an r-dimensional space where manifold search is enabled.

Figure 7.2 reports performance for regional search. The optimal value of r depends on the structure of the dataset. In all cases the optimal performance appears already before r = 1k. In particular for Paris6k, performance reaches its peak with as small as $r = 100$. Compared to FSR.exact as implemented in [ITA+17], it achieves the same mAP but 150 times faster on Oxford5k and Paris6k, while 300 times faster on Instre. Global search demonstrates a similar behavior but is skipped due to lack of space.

We achieve 97.0 mAP on Paris6k which is near-perfect performance. Figure 7.3 presents the two queries with the lowest average precision and their top-ranked negative images based on the ground-truth. It appears that in most cases the ground-truth is incorrect, as these images have visual overlap with the query bounding box. In both cases, we retrieve the object of interest with variations thanks to manifold search. The first true (visually) negative image for "La Défense" query appears at rank 126. This image does not contain "La Grande Arche" but it depicts buildings from the surroundings due to "topic drift" of manifold search. The same outcome is seen for the

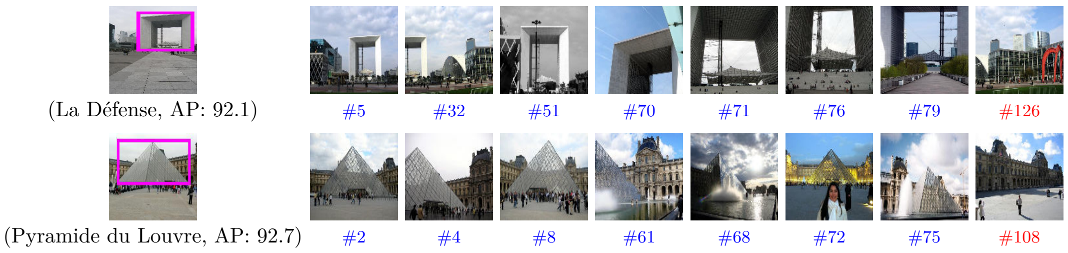


Figure 7.3: Two queries with the *lowest* AP from Paris6k and the corresponding top-ranked negative images based on the ground-truth. The left-most image contains the query object, and its AP is reported underneath it. Images considered to be negative based on the ground-truth are shown with their rank underneath. Ranks are marked in blue for incorrectly labeled images that are visually relevant (overlapping), and red otherwise.

other query, where the first visually negative image containing "Palais du Louvre" is at rank 108.

Regional search performs better than the global one [ITA+17] at the cost of more memory and slower query time. Our approach unlocks this bottleneck thanks to the offline pooling $\overline{U} = \Sigma U$. Indeed, global and regional search on Instre take 0.040s and 0.042s, respectively, with our method, while the corresponding times for FSR.exact are 0.055s and 3s.

Approximate eigendecomposition keeps the off-line stage tractable at large scale. The mAP equals 89.5 with FSR.rank-5000 and regional search on Instre. The drop is small with FSR.approx (89.2) while the offline computation is 20 times faster: it takes 3 hours instead of 60 hours with 570k Instre regional descriptors, using 16-core Intel Xeon 2.00GHz CPU. This is important at large scale because the off-line complexity of FSR.rank-r increases polynomially.

7.6.3 Large-scale Experiments

We now apply our approach to a larger scale by using only 5 descriptors per image thanks to a GMM reduction [ITA+17]. This choice improves the scalability while minimizing the accuracy losses.

FSRw**.approx** becomes crucial, especially at large scale, because vectors of sparsely populated parts of the graph are not well represented. Figure 7.4 shows the comparison between FSRw.approx and FSR.approx. We achieve 94.2 and 90.2 with FSRw.approx and FSR.approx respectively with r = 10k and Resnet101 descriptors.

We further report the performance separately for each of the 11 queries of Oxford105k dataset. Results are shown in Figure 7.5. Low values of r penalize sparsely populated parts of the graph, i.e. landmarks with less similar instances in the dataset. FSRw.approx partly solves this issue.

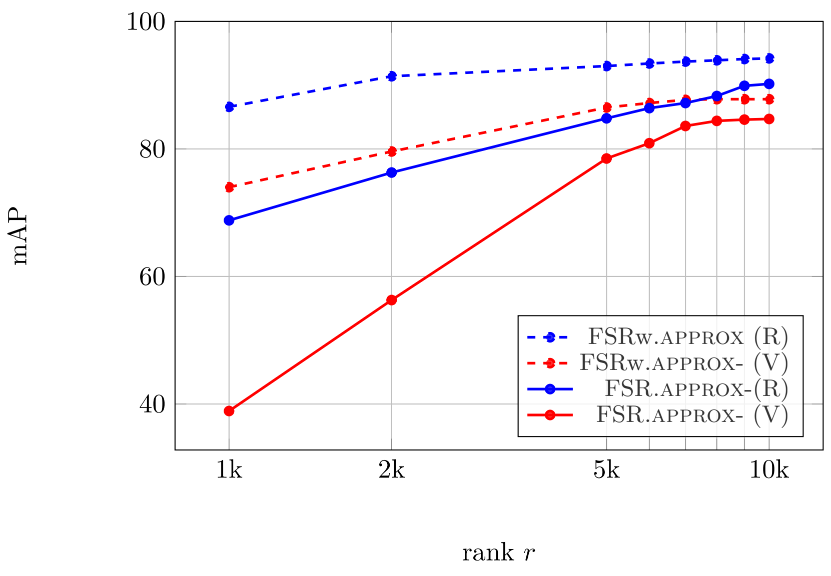


Figure 7.4: Oxford105k with FSR.APPROX and FSRw.APPROX using Resnet101(R) and VGG(V).

The search time is $0.14s$. and $0.3s$. per query for $r = 5k$ and $r = 10k$ respectively in Oxford105k. It is two orders of magnitude faster than FSR.exact: The implementation of [ITA$^+$17] requires about 14 seconds per query. Iscen et al. [ITA$^+$17] reduce this timing thanks to dataset truncation: manifold search is a re-ranking only applied to top-ranked images. We do not use any truncation. This improves the mAP by 4% and our method is still 1 second faster.

Sparse embeddings. We set r to a large enough value, such as 10k, to avoid compromising the search accuracy. This results in r-dimensional embeddings. Most descriptors belong only to few manifolds and each embedding vector has high energy in the components of the corresponding manifolds. Figure 7.6 presents the results of FSRw.approx. Remarkably, the drop equals %2 mAP for embeddings that are $\approx 90\%$ sparse. Making the embeddings sparser yields memory savings.

Quantized descriptors. Construction of the observation vector requires storage of the initial descriptors. We further use product quantization (PQ) [JDS11] to compress those and perform approximate search method in Euclidean space. When using PQ with 64 and 256 bytes, we achieve mAP equal to 91.1 and 94.2 in Oxford105k, respectively, while without PQ 94.4 is achieved. This comparison is performed with FSRw.approx.

7.6.4 Comparison to Other Methods

Table 7.1 compares our method with the state-of-the-art. We report results for $r = 5k$, FSR.rank-r for global description, FSR.approx for regional description, and

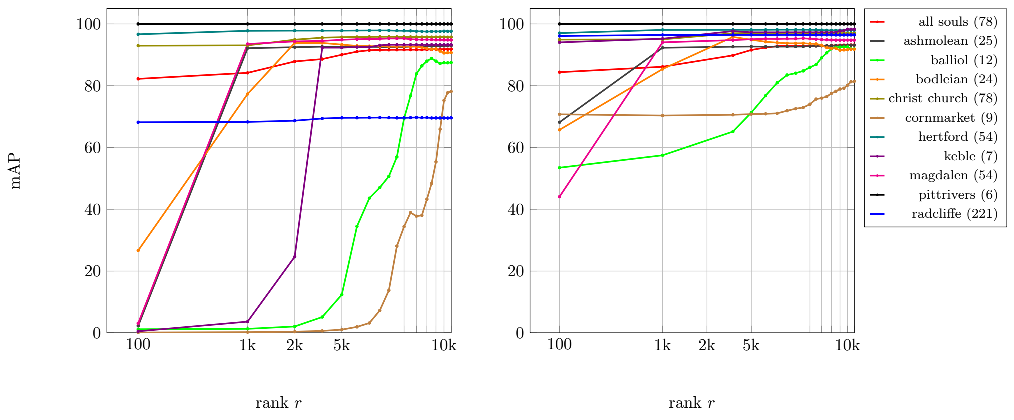


Figure 7.5: mAP reported separately for each landmark in Oxford105k with FSR.APPROX (left) and FSRw.APPROX (right). Number of positive images per landmark queries is shown in the labels.

Method	$m \times d$	INSTRE	Oxf5k	Oxf105k	Par6k	Par106k
Global descriptors - Euclidean search						
R-MAC [RTC16]	512	47.7	77.7	70.1	84.1	76.8
R-MAC [GARL16b]	2,048	62.6	83.9	80.8	93.8	89.9
Global descriptors - Manifold search						
Diffusion [ITA+17]	512	70.3	85.7	82.7	94.1	92.5
FSR.RANK-r	512	70.3	85.8	85.0	93.8	92.4
Diffusion [ITA+17]	2,048	80.5	87.1	87.4	96.5	95.4
FSR.RANK-r	2,048	80.5	87.5	87.9	96.4	95.3
Regional descriptors - Euclidean search						
R-match [RSCM16]	21×512	55.5	81.5	76.5	86.1	79.9
R-match [RSCM16]	$21\times2{,}048$	71.0	88.1	85.7	94.9	91.3
Regional descriptors - Manifold search						
Diffusion [ITA+17]	5×512	77.5	91.5	84.7	95.6	93.0
FSR.APPROX	5×512	78.4	91.6	86.5	95.6	92.4
Diffusion [ITA+17]	21×512	80.0	93.2	90.3	96.5	92.6
FSR.APPROX	21×512	80.4	93.0	-	96.5	-
Diffusion [ITA+17]	$5\times2{,}048$	88.4	95.0	90.0	96.4	95.8
FSR.APPROX	$5\times2{,}048$	88.5	95.1	93.0	96.5	95.2
Diffusion [ITA+17]	$21\times2{,}048$	89.6	95.8	94.2	96.9	95.3
FSR.APPROX	$21\times2{,}048$	89.2	95.8	-	97.0	-

Table 7.1: Performance comparison to the baseline methods and to the state of the art on manifold search [ITA+17]. Points at 512D are extracted with VGG [RTC16] and at 2048D with ResNet101 [GARL16b]. Regional representation with $m = 5$ descriptors per image uses GMM. Large-scale regional experiments use the FSRw.APPROX variant. Dataset truncation is used in [ITA+17] at large scale.

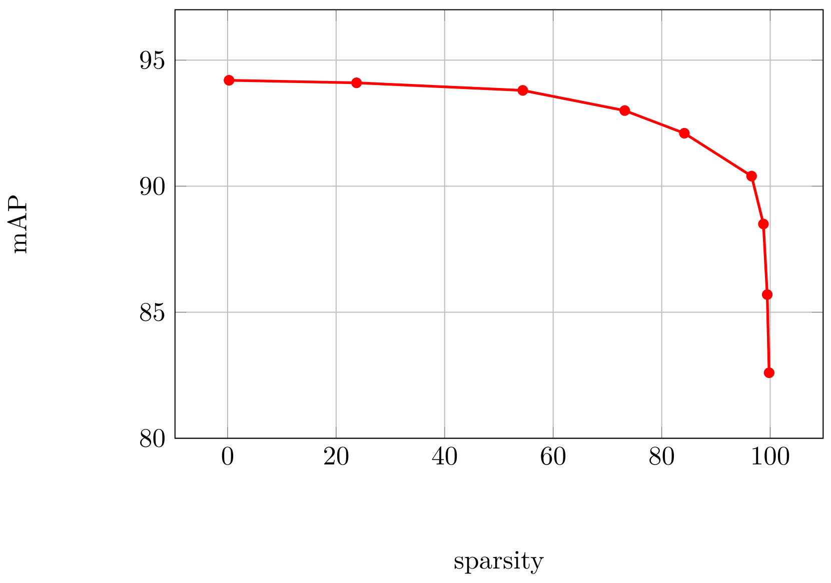


Figure 7.6: Effect of sparsification of U in Oxford105k with FSRw.APPROX and Resnet101 descriptors. The τ largest values of U are kept to achieve different levels of sparsity.

FSRw.approx in large-scale (with 100k distractors) and regional experiments. GMM reduces the number of regions per image from 21 to 5 [ITA$^+$17]. We do not experiment large-scale without GMM since there is not much improvement any more and it is less scalable. Our method reaches performance similar to that of FSR.exact as evaluated with CG [ITA$^+$17]. Our benefit comes from the dramatic speed-up. For the first time, manifold search runs almost as fast as Euclidean search. Consequently, dataset truncation is no longer needed and this improves the mAP.

7.7 Conclusion

This chapter reproduces the excellent results of online linear system solution [ITA$^+$17] at fraction of query time. We even improve performance by avoiding to truncate the graph online. The offline stage is linear in the dataset size, embarrassingly parallelizable and takes a few hours in practice for the large scale datasets of our experiments. The approximation quality is arbitrarily close to the optimal one at a given embedding dimensionality. The required dimensionality for good performance is large but in practice the embedded vectors are very sparse. This resembles an encoding based on a large vocabulary, searched via an inverted index. Although it is not the focus of this work, both Euclidean and dot product search stages can be handled by any efficient search method [JDS11]. Our method is generic and may be used for problems other than search, including clustering or unsupervised learning.

Chapter 8

Location Recognition with Set Representation

This chapter studies a location recognition problem. Our problem is a follows. We have a dataset corresponding to images with GPS locations. We assume that there are multiple images taken from a single location, corresponding to a panorama. Some of the traditional location recognition techniques treated this problem as a visual instance retrieval task. The user queries a single image from their location, and this image is matched against every image of the dataset. In this chapter, we propose to match panoramas to panoramas, instead of images to images.

We propose to construct panorama representatives by applying memory vectors introduced in Chapter 3. The main difference to the previous task is that now the groups are "natural", meaning that instead of random or weakly-supervised assignment, we assign images to the groups based on their GPS location. Surprisingly, not only panorama matching gives a significant improvement compared to the baseline in terms of accuracy. We also show in Section 8.3 that it even surpasses the manifold search technique we introduced in Chapter 6.

This work is published in [Iscen et al., 2017d].

8.1 Introduction

Location recognition has been treated as a visual instance retrieval task for many years [ZK06, SBS07, AZ14b, AGT^{+}16, HE08, SHSP16]. Additional, task-specific approaches include ground truth locations to find informative features [SBS07], regression for a more precise localization [TSP11, KGC15], or representation of the dataset as a graph [CS13]. A dense collection of multiple views allows 3D representations are possible, e.g. structured from motion [IZFB09], searching 2D features in 3D models [SLK12, LSHF12], or simultaneous visual localization and mapping [CN10]. How-

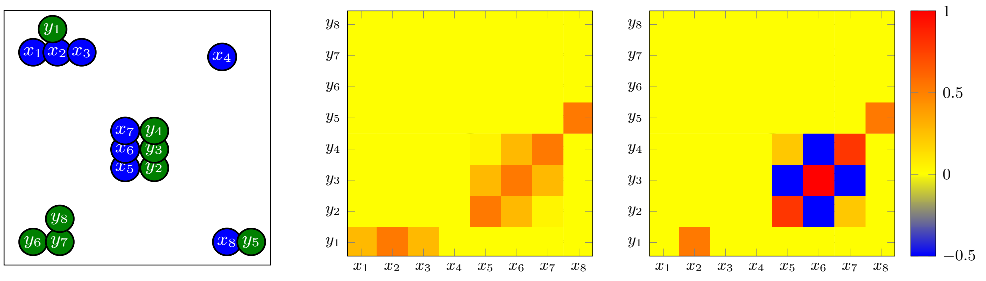


Figure 8.1: Left: Toy example of two vector sets $\mathbf{X}$, $\mathbf{Y}$ on the 2D plane are shown on the left. Middle: Pairwise similarity between all vectors, cross-matching with sum-vectors, *i.e.* $\mathbf{X}^\top\mathbf{Y}$ (8.1). Only for visualization purposes, and since we are dealing with unnormalized 2D vectors, the similarity between vectors $\mathbf{x}, \mathbf{y}$ is defined as $e^{\|\mathbf{x}-\mathbf{y}\|^2}$. Right: weighted pairwise-similarity between all vectors, cross-matching with pinv-vectors, *i.e.* $\mathbf{G}_\mathbf{X}^{-1}\mathbf{X}^\top\mathbf{Y}\mathbf{G}_\mathbf{Y}^{-1}$ (8.2).

ever, this does not apply to sparse “street-view” imagery [TSPO13b, TAS+15], where dataset items and queries are groups of images taken at a single location, in a panorama-like layout.

Several approaches on visual instance retrieval propose to jointly represent a set of images. These sets of images can appear at the query or at the database side. In the former case, these images are different views of the same object or scene [AZ12a, SJ15] and finally performance is improved. This joint representation, which commonly is an average query vector constructed via aggregation, is presumably more robust than each individual query vector. On the other hand, when aggregating images on the database side it is better to group them together by similarity [IFG+17]; images are assigned to sets, and a joint representation is created per set.

This work revisits location recognition by aggregating images both on query and database sides. Our method resembles implicit construction of a panorama, i.e. images are combined in the feature space and not in the image space, but we also experiment with an explicit construction. Contrary to the general case of visual instance retrieval, it is easy to obtain multiple query images, e.g. capturing them with a smartphone or with multiple cameras in the case of autonomous driving. On the database side, location provides a natural way of grouping images together. Thus, contrary to generic retrieval, the images to be aggregated on the query and database sides, may not be similar to each other; they rather depict whatever is visible around a particular location.

We significantly outperform the state of the art without any form of supervision other than the natural, location-based grouping of images, and without any costly offline process like 3D reconstruction. Indeed we are reaching near perfect location recognition on the Pittsburgh dataset [TSPO13b] even when we use as few as four views on the query side.

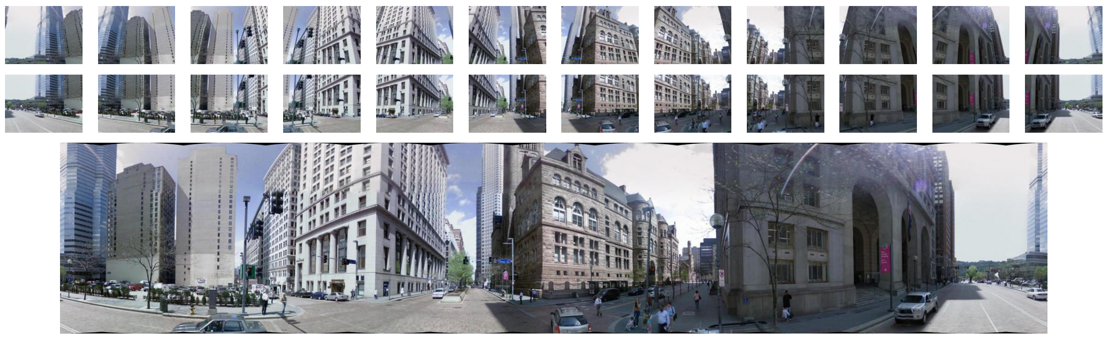


Figure 8.2: Example of all images assigned to a single location (first two rows) and the corresponding panorama (last row) covering a full 360 degree view, constructed by automatic stitching.

8.2 Panorama to Panorama Matching

This section describes our contribution for location recognition. We assume that for each possible location we are given a set of images covering a full 360 degree view while consecutive images have an overlap (see Figure 8.2). We propose two ways to construct a panoramic representation of each location: an implicit way by vector aggregation and an explicit way by image stitching into a panorama and extraction of a single descriptor.

8.2.1 Implicit Panorama Construction

We form a panoramic representation by aggregating the descriptors of images from the same location. In this way, we implicitly construct a panorama in the descriptor space. In order to achieve this, we employ two approaches for creating memory vectors, i.e. sum-vector (3.1) and pinv-vector (3.9).

In contrast to previous works that aggregate the image vectors only on the dataset side [IFG$^+$17] or only on the query side [SJ15], we rather do it for both. This requires that the query is also defined by a set of images which offer a 360 degree view. A realistic scenario of this context is autonomous driving and auto-localization where the query is defined by such a set of images.

Assume that n images in a dataset location are represented by $d \times n$ matrix $\mathbf{X}$ and that k images in the query location by $d \times k$ matrix $\mathbf{Y}$. Analyzing the similarity between the two sum-vectors is straightforward. Panorama similarity is given by the inner product

$$s(\mathbf{X}, \mathbf{Y}) = m(\mathbf{X})^\top m(\mathbf{Y}) = \mathbf{1}_n^\top \mathbf{X}^\top \mathbf{Y} \mathbf{1}_k. \tag{8.1}$$

Similarly, panorama similarity for pinv-vectors is given by

$$s^+(\mathbf{X}, \mathbf{Y}) = m^+(\mathbf{X})^\top m^+(\mathbf{Y}) = \mathbf{1}_n^\top \mathbf{G}_\mathbf{X}^{-1} \mathbf{X}^\top \mathbf{Y} \mathbf{G}_\mathbf{Y}^{-1} \mathbf{1}_k, \tag{8.2}$$

where $\mathbf{G}_\mathbf{X} = \mathbf{X}^\top \mathbf{X}$ is the Gram matrix for $\mathbf{X}$. Compared to (8.1), the sum after cross-matching is weighted now, and the weights are given by $\mathbf{G}_\mathbf{X}^{-1}$ and $\mathbf{G}_\mathbf{Y}^{-1}$. This is

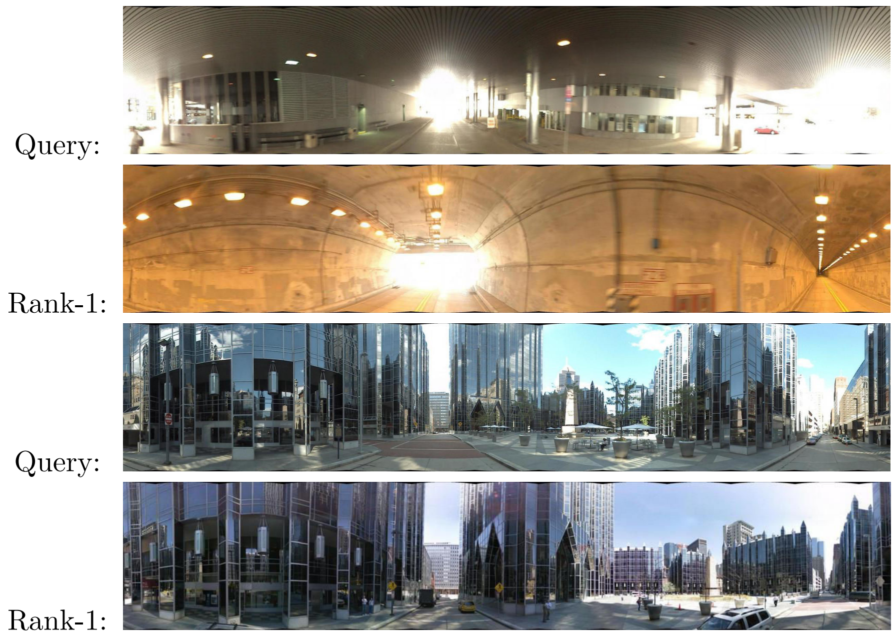


Figure 8.3: Two examples of failures with *pan2pan/net*. We show the query and the top ranked image from the dataset.

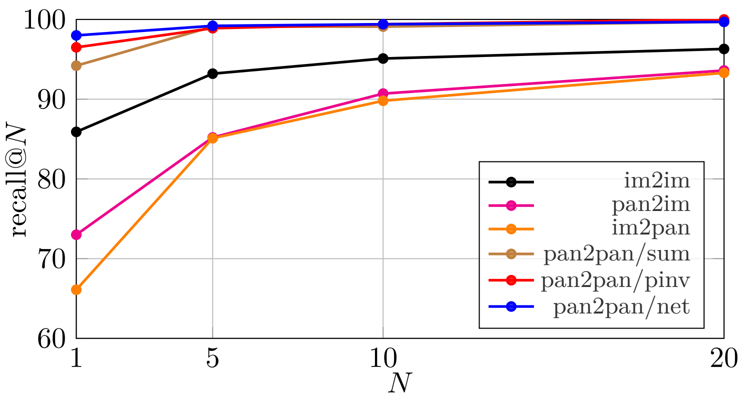
interpreted as "democratizing" the result of cross-matching; the contribution of vectors that are similar within the same set are down-weighted, just as in handling the burstiness phenomenon for local descriptors [JZ14]. We visualize this with a toy example in Figure 8.1. Unweighted cross-matching is dominated by "bursty" vectors in the same cluster. Democratization down-weights these contributions.

8.2.2 Explicit Panorama Construction

Our second approach explicitly creates a panoramic image. The descriptors are then extracted from the panorama. Given that images of a location are overlapping, we construct a panoramic image using an existing stitching method. In particular, we use the work of Brown and Lowe [BL07], which aligns, stitches, and blends images automatically based on their local SIFT descriptors and inlier correspondences. Figure 8.2 shows a stitched panoramic image. Once stitching is complete, we extract a single global descriptor from the panorama image, capturing the entire scene.

8.3 Experiments

In this section, we describe our experimental setup, and compare our method to a number of baselines using the state-of-the-art NetVLAD network in a popular location recognition benchmark.



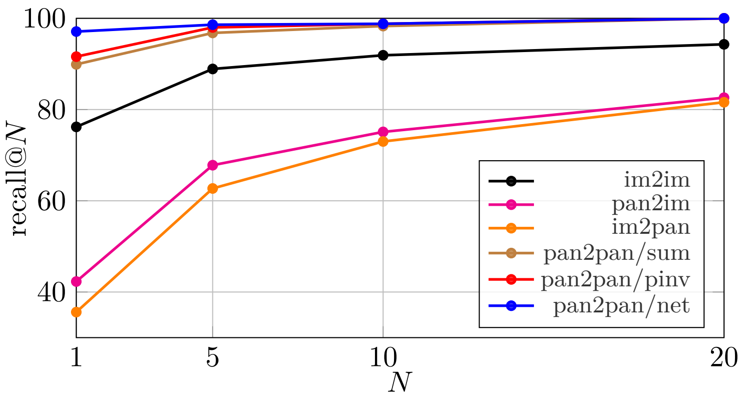


Figure 8.4: Comparison of existing approaches (im2im [AGT$^+$16], im2pan [IFG$^+$17], pan2im [SJ15]) with our methods (pan2pan/sum, pan2pan/pinv and pan2pan/net) for the full 4096D (left) and for reduced dimensionality to 256D (right).

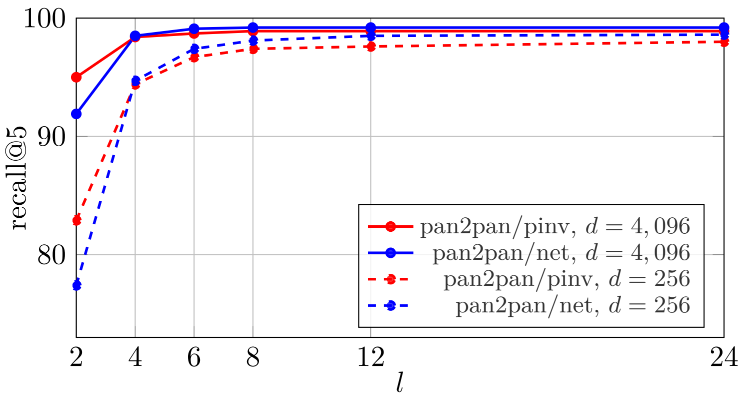


Figure 8.5: Recall@5 on Pitt250k, sampling l images from each query panorama and using NetVLAD descriptors of two different dimensionalities d. We report average measurements over 10 random experiments and compare our methods pan2pan/pinv and pan2pan/net.

8.3.1 Experimental Setup

The methods are evaluated on the Pittsburgh dataset [TSPO13b] referred to as Pitt250k. It contains 250k database and 24k query images from Google Street View. It is split into training, validation, and test sets [AGT$^+$16]. We evaluate our approach on the test set, which consists of 83,952 dataset images and 8,280 query images. Each image is associated with a GPS location and 24 images are associated with the same GPS location. Therefore, each panoramic representation aggregates 24 images. There is a total of 345 query locations and 3,498 dataset locations. We use NetVLAD for our descriptor representation in all experiments. While the original representation is $d = 4{,}096$ dimensional, we also experiment with reducing dimensionality to $d = 256$ by PCA.

The standard evaluation metric is Recall@N. It is defined to equal 1 if at least one of the top N retrieved dataset images is within 25 meters from the spatial location of the query. Average is reported over all queries. We follow this protocol for the baseline and other cases where the query images are used individually.

Aggregating on the query side implies that there is a single query per location: the number of queries decreases from 8,280 to 345. We report the average recall@N from these 345 panorama queries. Section 8.3.3 also experiments with a larger number of random queries, each capturing only a fraction of the panoramic view. In this case, recall@N is averaged over those random queries. Aggregating on the dataset side does not affect the standard evaluation.

8.3.2 Panorama Matching

We refer to our proposed method as panorama to panorama or pan2pan matching, in particular pan2pan/sum and pan2pan/pinv when aggregating descriptors with sum-vector and pinv-vector respectively; and as pan2pan/net when using a NetVLAD descriptor from an explicit panorama. We compare against the following baselines: image to image matching (im2im) as in the work by Arandjelovic et al. [AGT$^+$16], image to panorama matching (im2pan) corresponding to dataset-side aggregation as in the work by Iscen et al. [IFG$^+$17], and panorama to image matching (pan2im) corresponding to query-side aggregation as in the work by Sicre and Jégou [SJ15].

Figure 8.4 compares all methods for different descriptor dimensions. Clearly, panorama to panorama matching outperforms all other methods. The improvement is consistent for all N and significant for low N: pan2pan/net obtains 98% recall@1! There are only 7 failure queries. Two of them are shown in Figure 8.3. One is a challenging query depicting an indoor parking lot and the other actually retrieves the same building, which is incorrectly marked in the dataset's ground truth.

The recall is not only improved, but the search is also more efficient both speed-wise ($24^2\times$ faster) and memory-wise ($24\times$ less memory). Instead of comparing a given query image against 83k vectors, we only make 3.5k comparisons. Additional operations are introduced when aggregating the set of query images, but this cost is fixed and small compared to the savings from the dataset side.

Comparing to results in prior work, im2pan behaves as in the work of Iscen et al. [IFG+17] when compared to the baseline im2im. That is, memory compression and speed up at the cost of reduced performance. However, pan2im does not appear to be effective in our case, in contrast to the work of Sicre and Jégou [SJ15]. On the contrary, pan2pan significantly improves the performance while enjoying both memory compression and search efficiency.

8.3.3 Sparse Panorama Matching

Aggregating on the dataset side is performed off-line. However, the user is required to capture images and to construct a full panorama (24 images in our case) at query time. Even though this is not a daunting task given the advances of smartphones and tablets, we additionally investigate a scenario where the user only captures a partial panoramic view.

In particular, we randomly sample a subset of l images from the query location and consider them as the query image set. Explicit panorama construction is no longer possible because the sampled images may not overlap and so we cannot stitch them. In this case, we feed sampled images through the convolutional layers only, and stack together all activations before pooling them through the NetVLAD layer (pan2pan/net for sparse panoramas).

Figure 8.5 shows the results. Our methods have near-perfect performance even for a small number of sampled images. When the user only takes four random photos, we are able to locate them up to 99% recall@5. Another interesting observation is that pan2pan/pinv outperforms pan2pan/net for $l = 2$, which is expected due to the nature of pinv-vec construction. It is theoretically shown to perform well even if all the vectors in the set are random, as shown in the original paper [IFG+17].

8.3.4 Comparison to Diffusion-based Retrieval

This work casts location recognition as a retrieval task. Query expansion techniques significantly improve retrieval performance. We compare to the state-of-the-art retrieval method by Iscen et al. [ITA+17], a kind of query expansion based on graph diffusion. In this method, an image is represented by individual region descriptors and at query time all query regions are processed. We compare to this method by considering that regions and images in [ITA+17] correspond to images and panoramas respectively in our scenario.

Our pan2pan/pinv and pan2pan/net approaches achieve 96.5% and 98% recall@1 respectively, while the approach [ITA+17] gives 91.9%. Even though query expansion improves the baseline, it does not help as much as our methods. This can be expected because [ITA+17] is based on many instances of the same object, which is not the case for location recognition on street view imagery.

8.4 Conclusion

This chapter proposes an unsupervised and conceptually very simple method, yet very effective. Besides the performance gain, we make significant savings in space by aggregating descriptors of individual images over each group. The need for multiple query views is not very demanding because only four views are enough—an entire query panorama is definitely not needed. Although our aggregation methods have been used for instance retrieval in the past, we are the first to successfully aggregate on both dataset and query-side for location recognition (which in fact has failed for instance retrieval [SAJ15]).

Conclusion

This manuscript addresses the issues of efficiency and accuracy in image retrieval by designing continuous representations using a set of vectors. Our first contribution considers the indexing problem. We take a statistical signal processing point of view to represent a group of vectors in a continuous manner. We start with basic sum-pooling (3.1), and investigate certain properties of such representation in a membership test scenario. We then look for the optimal solution based on our hypothesis, which happens to be the pinv-pooling (3.9). This solution is shown to be better, theoretically and empirically, when the dimensionality of the descriptors is high. This is especially useful for the high-dimensional state-of-the-art descriptors.

Pooling strategies, such as sum (3.1) and pinv (3.9) assume that a set of vectors are already assigned to groups. We also investigate different group assignment strategies in Chapter 3. Random assignment is the most straightforward assignment strategy, and requires no additional offline computation (Sec. 3.3.2). In that case, vectors are randomly assigned to groups of same size. This is relevant for cases such as streaming data. We show that pinv construction is significantly better than sum construction with random assignment. On the other hand, it is also possible to assign similar vectors to the same group. We call this strategy weakly-supervised assignment (Sec. 3.3.3). It requires an additional offline data-driven process, such as clustering, to assign similar vectors together. Group sizes are not necessarily the same anymore, and depend on the distribution of the data. An interesting finding is that the the difference between the pinv and sum constructions diminishes as the similarity between intra-group vectors increases.

The weakly-supervised strategy proposed in Chapter 3 is not fully data-driven. Assignment strategy is data-driven, but the group representation is a generic design, such as pinv or sum-pooling. We look for purely data-driven solutions in Chapter 5. The idea is to adapt both assignment and representation based on data, and optimize them jointly. Our first solution is given by eigendecomposition (5.2.1). Its alternative based on dictionary learning (5.2.2) gives a sparser, hence more efficient solution. Our method is shown to be effective in well-known benchmarks, including a large-scale experiment with almost 100 million vectors. We achieve the same accuracy as baseline with only 10% of original complexity in almost all cases.

The second half of this transcript investigates manifold-based query expansion techniques. Contrary to indexing, such techniques trade efficiency for accuracy by introducing additional processing in the query time to improve the accuracy of the search. The relationship between the vectors is captured with a weighted graph in this scenario. In Chapter 6, we analyze the existing manifold search techniques, particularly diffusion processes. We make additional contributions related to image search, such as a way to handle multiple queries at the same time for regional vectors and a more efficient solution in the query time using conjugate gradients. We also propose large-scale extensions, such reducing the number of regional vectors per image with GMM, or truncating the affinity matrix based on the initial ranked-list. We achieve close to perfect performance in multiple benchmarks, while keeping the query time below one second even in large-scale datasets consisting of 2 million vectors.

Chapter 7 proposes to make manifold search even faster in the query time by introducing additional computations offline. The closed-form solution of diffusion involves taking the inverse of a large sparse matrix, and is not scalable for large-scale. We compute its low-rank approximation without computing the inverse directly. This is based on the fact that the solution can be written as geometric series. This means that only the eigenvalues change after each iteration and eigenvectors remain the same. Our experiments reproduce our state-of-the-art scores from Chapter 6, but they are up to 300 times faster in the query time.

Manifold Search vs Memory Vectors. Throughout this transcript, we propose two separate methods for hashing and query expansion. In Chapters 3 and 4, memory vectors do not bring any improvement in terms of search accuracy, and are solely used for indexing to benefit complexity. However, this is not the case in Chapter 8, where memory vectors improve the search accuracy dramatically, when used to group panoramas on the dataset and query side. It is also shown that manifold search (Chapter 6), which has improved the search accuracy in instance-based image retrieval, is not as effective as memory vectors in this scenario.

Another similar setup for a comparison between the two approaches would be to aggregate the sub-regions of an image to a single vector using pinv construction, as opposed to regional diffusion in Chapters 6 and 7. In that case, the search accuracy even drops for Oxford5k, Paris6k, and Instre using ResNet descriptors (see Sec. 6.3). We achieve 74.2, 90.0, and 45.7 respectively with pinv construction in these datasets, compared to 95.8, 95.3, and 89.6 with regional diffusion.

This observation can be explained by the definition of the problem. Images in the same group for location recognition are composed of a panorama, meaning that there are multiple views and perspectives. Furthermore, it is likely that multiple sub-images from a panorama will be relevant, since there are may be multiple buildings and landmarks that discriminate a location. On the other hand, the situation is different when sampling regions from an image. There may be a single object of interest corresponding to a few regions, and the rest may be clutter or background. Pseudo-inverse pooling is especially appropriate for the former problem, since it creates a representation which equally weighs the contributions of all the viewpoints, not just the dominant ones, hence the improvement in search accuracy.

8.4.1 Perspectives

We conclude this manuscript by discussing possible future directions related to the methods discussed in this manuscript.

New memory vector designs

Our solutions to design memory vectors, i.e. sum (3.1) and pinv (3.9), require the dimensionality of group vectors to be the same as image vectors. One possible direction is to remove this constraint and investigate a scenario where the dimensionality of group representation can vary. This introduces another degree of freedom, and yet another parameter that may effect the trade-off between accuracy and efficiency. Going to higher-dimensional spaces is likely to improve the accuracy, but hurt efficiency due to bigger vectors. A thorough analysis about the theoretical and empirical performance of such approach may benefit group testing in image retrieval even further.

Random walks

We only investigate a certain random walk strategy (2.27) in Chapter 6. Many other similar techniques exist in the literature [DB13], and their application in our task may give us a better understanding of manifold search in image retrieval. An important constraint in that direction is that the efficiency of the search system must be preserved. Thus, whatever process is used for manifold search, it must converge to a good solution quickly in practice. Another alternative is to derive a closed-form solution and apply it to our scheme in Chapter 7. Assuming that the closed-form solution is based on element-wise operations on a diagonal matrix, they search system would remain efficient.

Low-shot learning

This manuscript shows the effectiveness of manifold search in image retrieval. However, it can be extended to different computer vision tasks as well. An example would be to use manifold search for learning without fully-labeled data, such as low-shot learning. In that scenario, only a few labeled instances of a class exist and the rest of the data is unlabeled. It's important to retrieve the unlabeled data of the same class, so that it can be used to learn about different visual variations of the same object. Manifold search can be utilized in such case, in order to retrieve the unlabeled data of a given class and use it during the learning process.

Multimodal embeddings

Our embedding technique proposed in Chapter 7 proposes embeddings in diffusion space. Even though we only use it with images, an interesting direction would be to investigate its capability of creating common spaces for multimodal data. A straightforward approach would be to build separate subgraphs for different modalities, and create edges between them based on some ground-truth. Resulting would be a single connected graph, whose decomposition would give us embeddings in a common space.

Full list of publications

[Iscen et al., 2016a] Iscen, A., Amsaleg, L., and Furon, T. (2016a). Scaling group testing similarity search. In ACM International Conference on Multimedia Retrieval.

[Iscen et al., 2017a] Iscen, A., Avrithis, Y., Tolias, G., Furon, T., and Chum, O. (2017a). Fast spectral ranking for similarity search. arXiv preprint.

[Iscen and Furon, 2016] Iscen, A. and Furon, T. (2016). Group testing for identification with privacy. In ACM Workshop on Information Hiding and Multimedia Security.

[Iscen et al., 2017b] Iscen, A., Furon, T., Gripon, V., Rabbat, M., and Jégou, H. (2017b). Memory vectors for similarity search in high-dimensional spaces. IEEE Transactions on Big Data.

[Iscen et al., 2016b] Iscen, A., Rabbat, M., and Furon, T. (2016b). Efficient large-scale similarity search using matrix factorization. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[Iscen et al., 2017c] Iscen, A., Tolias, G., Avrithis, Y., Furon, T., and Chum, O. (2017c). Efficient diffusion on region manifolds: Recovering small objects with compact cnn representations. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[Iscen et al., 2017d] Iscen, A., Tolias, G., Avrithis, Y., Furon, T., and Chum, O. (2017d). Panorama to panorama matching for location recognition. In ACM International Conference on Multimedia Retrieval.

Bibliography

[ACF$^+$15] Laurent Amsaleg, Oussama Chelly, Teddy Furon, Stéphane Girard, Michael E Houle, Ken-ichi Kawarabayashi, and Michael Nett. Estimating local intrinsic dimensionality. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 29–38. ACM, 2015.

[AGT$^+$16] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In CVPR, 2016.

[AHPV04] Pankaj K Agarwal, Sariel Har-Peled, and Kasturi R Varadarajan. Approximating extent measures of points. Journal of the ACM, 51(4):606–635, 2004.

[AM05] Karim M Abadir and Jan R Magnus. Matrix algebra. Cambridge University Press, 2005.

[AR11] Y. Avrithis and K. Rapantzikos. The medial feature detector: Stable regions from image boundaries. In ICCV, Nov. 2011.

[ARS$^+$14] Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. From generic to specific deep representations for visual recognition. arXiv preprint arXiv:1406.5774, 2014.

[AZ12a] Relja Arandjelovic and Andrew Zisserman. Multiple queries for large scale specific object retrieval. In BMVC, 2012.

[AZ12b] Relja Arandjelovic and Andrew Zisserman. Three things everyone should know to improve object retrieval. In CVPR, Jun. 2012.

[AZ13] R. Arandjelovic and A. Zisserman. All about VLAD. In CVPR, Jun. 2013.

[AZ14a] R. Arandjelović and A. Zisserman. Extremely low bit-rate nearest neighbor search using a set compression tree. Trans. PAMI, 2014.

[AZ14b] Relja Arandjelovic and Andrew Zisserman. Dislocation: Scalable descriptor distinctiveness for location recognition. In ACCV, 2014.

[B$^+$06] Christopher M Bishop et al. Pattern recognition and machine learning, volume 1. springer New York, 2006.

[BBL$^+$16] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. arXiv preprint arXiv:1611.08097, 2016.

[BCF$^+$14] Julien Bringer, Herve Chabanne, Melanie Favre, Alain Patey, Thomas Schneider, and Michael Zohner. GSHADE: faster privacy-preserving distance computation and biometric identification. In Proceedings of the 2Nd ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec '14, pages 187–198, New York, NY, USA, 2014. ACM.

[BCP13] Julien Bringer, Hervé Chabanne, and Alain Patey. SHADE: secure hamming distance computation from oblivious transfer. In Financial Cryptography and Data Security, volume 7862 of Lecture Notes in Computer Science, pages 164–176, 2013.

[BETG08] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. Computer Vision and Image Understanding, 110(3):346–359, May 2008.

[BGRS99] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In Proceedings of the International Conference on Database Theory, pages 217–235, Aug. 1999.

[BL07] Matthew Brown and David G Lowe. Automatic panoramic image stitching using invariant features. IJCV, 74:59–73, 2007.

[BL12] Artem Babenko and Victor Lempitsky. The inverted multi-index. In CVPR, Jun. 2012.

[BL15] Artem Babenko and Victor Lempitsky. Aggregating deep convolutional features for image retrieval. In ICCV, 2015.

[BL16] Artem Babenko and Victor Lempitsky. Efficient indexing of billion-scale datasets of deep descriptors. In CVPR, 2016.

[BLSV06] Paolo Boldi, Violetta Lonati, Massimo Santini, and Sebastiano Vigna. Graph fibrations, graph isomorphism, and PageRank. RAIRO-Theoretical Informatics and Applications, 40(2):227–253, 2006.

[BMM15] Pedro Borges, André Mourão, and João Magalhães. High-dimensional indexing by sparse approximation. In Proceedings of the 5th ACM on International Conference on Multimedia Retrieval, 2015.

[BPL10] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In ICML, 2010.

[BS09] Liefeng Bo and Cristian Sminchisescu. Efficient match kernels between sets of features for visual recognition. In NIPS, 2009.

[BSCL14] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural codes for image retrieval. In ECCV, 2014.

[BSJ16] Binod Bhattarai, Gaurav Sharma, and Frederic Jurie. Cp-mtml: Coupled projection multi-task metric learning for large scale face retrieval. In CVPR, 2016.

[BZSL13] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203, 2013.

[CDF$^+$04] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In ECCV Workshop Statistical Learning in Computer Vision, 2004.

[Cha02] Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In STOC, pages 380–388, May 2002.

[Chu97] Fan RK Chung. Spectral graph theory, volume 92. American Mathematical Soc., 1997.

[CK16] Siddhartha Chandra and Iasonas Kokkinos. Fast, exact and multi-scale inference for semantic image segmentation with deep Gaussian CRFs. In ECCV, pages 402–418, 2016.

[CLRS09] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. Introduction to algorithms. Massachusetts Institute of Technology, 2009.

[CLSF10] M. Calonder, Vincent Lepetit, C. Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In ECCV, Oct. 2010.

[CMPM11] O. Chum, A. Mikulik, M. Perdoch, and J. Matas. Total recall II: Query expansion revisited. In CVPR, Jun. 2011.

[CN10] M. Cummins and P. Newman. Fab-map: Appearance-based place recognition and mapping using a learned visual vocabulary model. In ICML, 2010.

[CPS$^+$07] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In ICCV, Oct. 2007.

[CS13] Song Cao and Noah Snavely. Graph-based discriminative learning for location recognition. In CVPR, 2013.

[CTC$^+$09] V. Chandrasekhar, G. Takacs, D. Chen, S. Tsai, R. Grzeszczuk, and B. Girod. CHoG: compressed histogram of gradients. In CVPR, Jun. 2009.

[CWS03] Olivier Chapelle, Jason Weston, and Bernhard Scholkopf. Cluster kernels for semi-supervised learning. NIPS, pages 601–608, 2003.

[CZHZ16] Shuhan Chen, Ling Zheng, Xuelong Hu, and Ping Zhou. Discriminative saliency propagation with sink points. Pattern recognition, 60:2–12, 2016.

[DB13] Michael Donoser and Horst Bischof. Diffusion processes for retrieval revisited. In CVPR, 2013.

[DBV16] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In NIPS, pages 3837–3845, 2016.

[DCL08] Wei Dong, Moses Charikar, and Kai Li. Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces. In SIGIR, pages 123–130, Jul. 2008.

[DCL11] Wei Dong, M. Charikar, and K. Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In WWW, Mar. 2011.

[DGJP13] Jonathan Delhumeau, Philippe-Henri Gosselin, Hervé Jégou, and Patrick Pérez. Revisiting the VLAD image representation. In ACM Multimedia, Oct. 2013.

[DIIM04] M. Datar, N. Immorlica, P. Indyk, and V.S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In Proceedings of the Symposium on Computational Geometry, 2004.

[DJAH14] Agni Delvinioti, Hervé Jégou, Laurent Amsaleg, and Michael Houle. Image retrieval with reciprocal and shared nearest neighbors. In VISAPP, Jan. 2014.

[DM01] Inderjit S Dhillon and Dharmendra S Modha. Concept decompositions for large sparse text data using clustering. Machine learning, 42(1-2):143–175, 2001.

[DM05] Petros Drineas and Michael W Mahoney. On the Nyström method for approximating a gram matrix for improved kernel-based learning. Journal of Machine Learning Research, 6(Dec):2153–2175, 2005.

[DMZ94] Geoffrey M Davis, Stephane G Mallat, and Zhifeng Zhang. Adaptive time-frequency decompositions with matching pursuit. In SPIE's International Symposium on Optical Engineering and Photonics in Aerospace Sensing, pages 402–413, 1994.

[Don13] Michael Donoser. Replicator graph clustering. In BMVC, 2013.

[EKG10] Amir Egozi, Yosi Keller, and Hugo Guterman. Improving shape retrieval by spectral matching and meta similarity. IEEE Transactions on Image Processing, 19(5):1319–1327, 2010.

[ESV12] Ehsan Elhamifar, Guillermo Sapiro, and Rene Vidal. See all by looking at a few: Sparse modeling for finding representative objects. In CVPR, 2012.

[FBF77] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. ACM Transaction on Mathematical Software, 3(3):209–226, 1977.

[FFS13] Dan Feldman, Micha Feigin, and Nir Sochen. Learning big (image) data via coresets for dictionaries. Journal of Mathematical Imaging and Vision, 46(3):276–291, 2013.

[FNOK12] Yasuhiro Fujiwara, Makoto Nakatsuji, Makoto Onizuka, and Masaru Kitsuregawa. Fast and exact top-k search for random walk with restart. Proceedings of the VLDB Endowment, 5(5):442–453, 2012.

[GARL16a] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. Deep image retrieval: Learning global representations for image search. ECCV, 2016.

[GARL16b] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. End-to-end learning of deep visual representations for image retrieval. arXiv preprint arXiv:1610.07940, 2016.

[GIM99] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimension via hashing. In Proceedings of the International Conference on Very Large DataBases, pages 518–529, 1999.

[GMJP14] Philippe-Henri Gosselin, Naila Murray, Hervé Jégou, and Florent Perronnin. Revisiting the fisher vector for fine-grained classification. Pattern recognition letters, 2014. to appear.

[GP11] Albert Gordo and Florent Perronnin. Asymmetric distances for binary embeddings. In CVPR, 2011.

[Gra06] Leo Grady. Random walks for image segmentation. Trans. PAMI, 28(11):1768–1783, 2006.

[GWGL14] Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In ECCV, 2014.

[Hac94] Wolfgang Hackbusch. Iterative solution of large sparse systems of equations. Springer Verlag, 1994.

[HBL15] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. arXiv preprint arXiv:1506.05163, 2015.

[HE08] James Hays and Alexei A Efros. Im2gps: estimating geographic information from a single image. In CVPR, 2008.

[HEMF15] A. Hadid, N. Evans, S. Marcel, and J. Fierrez. Biometrics systems under spoofing attack. IEEE Signal Processing Magazine, 32(5):20–, September 2015.

[HMT11] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM Review, 53(2):217–288, 2011.

[HRBLM07] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Oct. 2007.

[HS88] Chris Harris and Mike Stephens. A combined corner and edge detector. In Alvey vision conference, 1988.

[HSDF15] Jared Heinly, Johannes L Schonberger, Enrique Dunn, and Jan-Michael Frahm. Reconstructing the world* in six days*(as captured by the yahoo 100 million image dataset). In CVPR, 2015.

[Hub65] Charles H Hubbell. An input-output approach to clique identification. Sociometry, 1965.

[HVG11] David K Hammond, Pierre Vandergheynst, and RÉmi Gribonval. Wavelets on graphs via spectral graph theory. Applied and Computational Harmonic Analysis, 30(2):129–150, 2011.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016.

[IFG^{+}17] Ahmet Iscen, Teddy Furon, Vincent Gripon, Michael Rabbat, and Hervé Jégou. Memory vectors for similarity search in high-dimensional spaces. IEEE Trans. Big Data, 2017. to appear.

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In STOC, pages 604–613, 1998.

[IRF16] Ahmet Iscen, Michael Rabbat, and Teddy Furon. Efficient large-scale similarity search using matrix factorization. In CVPR, 2016.

[ITA$^+$17] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, Teddy Furon, and Ondrej Chum. Efficient diffusion on region manifolds: Recovering small objects with compact cnn representations. In CVPR, 2017.

[ITGJ15] Ahmet Iscen, Giorgos Tolias, Philippe-Henri Gosselin, and Hervé Jégou. A comparison of dense region detectors for image search and fine-grained classification. IEEE Transactions on Image Processing, 24(8), 2015.

[IZFB09] Arnold Irschara, Christopher Zach, J-M Frahm, and Horst Bischof. From structure-from-motion point clouds to fast location recognition. In CVPR, 2009.

[JB08] Yushi Jing and Shumeet Baluja. Visualrank: Applying PageRank to large-scale image search. Trans. PAMI, 30(11):1877–1890, 2008.

[JC12] Hervé Jégou and Ondrej Chum. Negative evidences and co-occurrences in image retrieval: The benefit of PCA and whitening. In ECCV, Oct. 2012.

[JDS08] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In ECCV, Oct. 2008.

[JDS09] Hervé Jégou, Matthihs Douze, and Cordelia Schmid. On the burstiness of visual elements. In CVPR, Jun. 2009.

[JDS10] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Improving bag-of-features for large scale image search. IJCV, 87(3):316–336, Feb. 2010.

[JDS11] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. Trans. PAMI, 33(1):117–128, Jan. 2011.

[JDSP10] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In CVPR, Jun. 2010.

[JH98] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In NIPS, 1998.

[JHS07] Hervé Jégou, Hedi Harzallah, and Cordelia Schmid. A contextual dissimilarity measure for accurate and efficient image search. In CVPR, Jun. 2007.

[JJG11] Mihir Jain, Hervé Jégou, and Patrick Gros. Asymmetric hamming embedding. In ACM Multimedia, Oct. 2011.

[JPD$^+$12] Herve Jegou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Perez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. Trans. PAMI, 34:1704–1716, 2012.

[JSSK16] Jinhong Jung, Kijung Shin, Lee Sael, and U Kang. Random walk with restart on large graphs using block elimination. ACM Transactions on Database Systems, 41(2):12, 2016.

[JZ14] Hervé Jégou and Andrew Zisserman. Triangulation embedding and democratic kernels for image search. In CVPR, Jun. 2014.

[KA14] Yannis Kalantidis and Yannis Avrithis. Locally optimized product quantization for approximate nearest neighbor search. In CVPR, 2014.

[Kat53] Leo Katz. A new status index derived from sociometric analysis. Psychometrika, 18(1):39–43, 1953.

[KDB09] Peter Kontschieder, Michael Donoser, and Horst Bischof. Beyond pairwise shape similarity analysis. In ACCV, 2009.

[KF11] U Kang and Christos Faloutsos. Beyond 'caveman communities': Hubs and spokes for graph compression and mining. In Proceedings of the IEEE International Conference on Data Mining, pages 300–309. IEEE, 2011.

[KGC15] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In ICCV, 2015.

[KL02] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In ICML, 2002.

[KLL08] Tae Hoon Kim, Kyoung Mu Lee, and Sang Uk Lee. Generative image segmentation using random walks with restart. In ECCV, pages 264–275. Springer, 2008.

[KMO15] Yannis Kalantidis, Clayton Mellina, and Simon Osindero. Cross-dimensional weighting for aggregated deep convolutional features. arXiv preprint arXiv:1512.04065, 2015.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

[KSTC02] Jaz Kandola, John Shawe-Taylor, and Nello Cristianini. Learning semantic similarity. In NIPS, 2002.

[KV04] Risi Kondor and Jean-Philippe Vert. Diffusion kernels. Kernel Methods in Computational Biology, pages 171–192, 2004.

[Lin98] T. Lindeberg. Feature detection with automatic scale selection. IJCV, 30(2):77–116, 1998.

[LJW$^+$07] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe LSH: Efficient indexing for high-dimensional similarity search. In Proceedings of the International Conference on Very Large DataBases, pages 950–961, 2007.

[Llo82] Stuart Lloyd. Least squares quantization in pcm. IEEE transactions on information theory, 28:129–137, 1982.

[LM04] Amy N Langville and Carl D Meyer. Deeper inside PageRank. Internet Mathematics, 1(3):335–380, 2004.

[LMV14] Song Lu, Vijay Mahadevan, and Nuno Vasconcelos. Learning optimal seeds for diffusion-based salient object detection. In CVPR, 2014.

[Low04] D. G. Lowe. Distinctive image features from scale-invariant keypoints. IJCV, 60(2):91–110, 2004.

[LSHF12] Yunpeng Li, Noah Snavely, Dan Huttenlocher, and Pascal Fua. Worldwide pose estimation using 3d point clouds. In ECCV, 2012.

[LV07] John A Lee and Michel Verleysen. Nonlinear dimensionality reduction. Springer Science & Business Media, 2007.

[MB15] Konda Reddy Mopuri and R. Venkatesh Babu. Object level deep feature pooling for compact image representation. CVPRW, 2015.

[MBM$^+$16] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. arXiv preprint arXiv:1611.08402, 2016.

[MBP14] Julien Mairal, Francis Bach, and Jean Ponce. Sparse modeling for image and vision processing. arXiv preprint arXiv:1411.3230, 2014.

[MBPS10] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. The Journal of Machine Learning Research, 11:19–60, 2010.

[MCM13] Andrej Mikulik, Ondřej Chum, and Jiří Matas. Image retrieval for online browsing in large image collections. In International Conference on Similarity Search and Applications, 2013.

[MCMP02] Jiri Matas, Ondrej Chum, U. Martin, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In BMVC, pages 384–393, Sep. 2002.

[ML14] Marius Muja and David G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. Trans. PAMI, 36, 2014.

[MP14] Naila Murray and Florent Perronnin. Generalized max-pooling. In CVPR, Jun. 2014.

[MS04] Krystian Mikolajczyk and Cordelia Schmid. Scale and affine invariant interest point detectors. IJCV, 60(1):63–86, Oct. 2004.

[MTS$^+$05] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, F. Schaffalitzky, T. Kadir, and Luc Van Gool. A comparison of affine region detectors. IJCV, 65(1/2):43–72, 2005.

[MZS03] K. Mikolajczyk, A. Zisserman, and C. Schmid. Shape recognition with edge-based features. In BMVC, 2003.

[NJT06] Eric Nowak, Frédéric Jurie, and Bill Triggs. Sampling strategies for bag-of-features image classification. In ECCV, pages 490–503, 2006.

[NLCK05] Boaz Nadler, Stephane Lafon, Ronald R Coifman, and Ioannis G Kevrekidis. Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. NIPS, 2005.

[NPG14] Romain Negrel, David Picard, and Philippe-Henri Gosselin. Dimensionality reduction of visual features using sparse projectors for content-based image retrieval. In ICIP 2014, pages 2192–2196, 2014.

[NS06] David Nistér and Henrik Stewénius. Scalable recognition with a vocabulary tree. In CVPR, pages 2161–2168, Jun. 2006.

[NW06] Jorge Nocedal and Stephen Wright. Numerical optimization. Springer, 2006.

[OS10] Alan V Oppenheim and Ronald W Schafer. Discrete-Time Signal Processing: Pearson New International Edition. Pearson Higher Ed, 2010.

[PBMW99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: bringing order to the web. 1999.

[PCI$^+$07] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In CVPR, Jun. 2007.

[PCI$^+$08] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In CVPR, Jun. 2008.

[PD07] Florent Perronnin and Christopher R. Dance. Fisher kernels on visual vocabularies for image categorization. In CVPR, Jun. 2007.

[PJM10] F. Perronnin, J.Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In ECCV, Sep. 2010.

[PKP17] Gilles Puy, Srdan Kitic, and Patrick Pérez. Unifying local and non-local signal processing with graph cnns. arXiv preprint arXiv:1702.07759, 2017.

[PRK93] Yagyensh Chandra Pati, Ramin Rezaiifar, and PS Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In ASILOMAR, pages 40–44, 1993.

[PSL10] Florent Perronnin, J. Sánchez, and Y. Liu. Large-scale image categorization with explicit data embedding. In CVPR, 2010.

[PVG+11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. The Journal of Machine Learning Research, 12:2825–2830, 2011.

[PYFD04] Jia-Yu Pan, Hyung-Jeong Yang, Christos Faloutsos, and Pinar Duygulu. Automatic multimedia cross-modal correlation discovery. In International Conference on Knowledge Discovery and Data Mining. ACM, 2004.

[QGB+11] Danfeng Qin, Stephan Gammeter, Lukas Bossard, Till Quack, and Luc Van Gool. Hello neighbor: Accurate object retrieval with k-reciprocal nearest neighbors. In CVPR, 2011.

[RD01] Matthew Richardson and Pedro M Domingos. The intelligent surfer: Probabilistic combination of link and content information in PageRank. In NIPS, 2001.

[RH05] Havard Rue and Leonhard Held. Gaussian Markov random fields: theory and applications. CRC Press, 2005.

[RL10] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In NIPS, 2010.

[RM72] C. Radhakrishna Rao and Sujit Kumar Mitra. Generalized inverse of a matrix and its applications. In Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Theory of Statistics, 1972.

[RSCM16] Ali S Razavian, Josephine Sullivan, Stefan Carlsson, and Atsuto Maki. Visual instance retrieval with deep convolutional networks. ITE Transactions on Media Technology and Applications, 4:251–258, 2016.

[RST09] Vladimir Rokhlin, Arthur Szlam, and Mark Tygert. A randomized algorithm for principal component analysis. SIAM Journal on Matrix Analysis and Applications, 31(3):1100–1124, 2009.

[RTB$^+$15] SG Roux, N Tremblay, P Borgnat, P Abry, H Wendt, and P Messier. Multiscale anisotropic texture unsupervised clustering for photographic paper. In IEEE International Workshop on Information Forensics and Security, pages 1–6, 2015.

[RTC16] Filip Radenović, Giorgos Tolias, and Ondřej Chum. CNN image retrieval learns from bow: Unsupervised fine-tuning with hard examples. ECCV, 2016.

[SAJ15] Miaojing Shi, Yannis Avrithis, and Herve Jégou. Early burst detection for memory-efficient image retrieval. In CVPR, 2015.

[SBS07] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In CVPR, 2007.

[Scu10] David Sculley. Web-scale k-means clustering. In Proceedings of the 19th international conference on World wide web. ACM, 2010.

[See49] John R Seeley. The net of reciprocal influence. a problem in treating sociometric data. Canadian Journal of Experimental Psychology, 3:234, 1949.

[SFJ14] Miaojing Shi, Teddy Furon, and Hervé Jégou. A group testing framework for similarity search in high-dimensional spaces. In ACM Multimedia, Nov. 2014.

[SHSP16] Torsten Sattler, Michal Havlena, Konrad Schindler, and Marc Pollefeys. Large-scale location recognition and the geometric burstiness problem. In CVPR, 2016.

[SJ15] Ronan Sicre and Hervé Jégou. Memory vectors for particular object retrieval with multiple queries. In ICMR, 2015.

[SK03] Alexander J Smola and Risi Kondor. Kernels and regularization on graphs. In Learning Theory and Kernel Machines, pages 144–158. Springer, 2003.

[SLBW14] Xiaohui Shen, Zhe Lin, Jonathan Brandt, and Ying Wu. Spatially-constrained similarity measure for large-scale object retrieval. Trans. PAMI, 36(6):1229–1241, 2014.

[SLK12] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Improving image-based localization by active correspondence search. In ECCV, 2012.

[SM83] Gerard Salton and Michael J. McGill. Introduction to modern information retrieval. McGraw-Hill, 1983.

[SM13] Aliaksei Sandryhaila and Jose MF Moura. Discrete signal processing on graphs. IEEE Transactions on Signal Processing, 61(7):1644–1656, 2013.

[SNF$^+$13] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. IEEE Signal Processing Magazine, 30(3):83–98, 2013.

[SPMV13] Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. IJCV, 105:222–245, 2013.

[SPVZ13] K. Simonyan, O. M. Parkhi, A. Vedaldi, and A. Zisserman. Fisher Vector Faces in the Wild. In British Machine Vision Conference, 2013.

[SRCF15] Johannes L Schonberger, Filip Radenovic, Ondrej Chum, and Jan-Michael Frahm. From single image query to detailed 3d reconstruction. In CVPR, 2015.

[SSM98] Alex J Smola, Bernhard Scholkopf, and Klaus-Robert Muller. The connection between regularization operators and support vector kernels. Neural Networks, 11(4):637–649, 1998.

[STC04] John Shawe-Taylor and Nello Cristianini. Kernel methods for pattern analysis. Cambridge university press, 2004.

[SVF11] David I Shuman, Pierre Vandergheynst, and Pascal Frossard. Chebyshev polynomial approximation for distributed signal processing. In International Conference on Distributed Computing in Sensor Systems and Workshops, pages 1–8. IEEE, 2011.

[SVZ13] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. Technical report, Department of Engineering Science, University of Oxford, 2013.

[SVZ14] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. Trans. PAMI, 2014.

[SWLK12] Torsten Sattler, Tobias Weyand, Bastian Leibe, and Leif Kobbelt. Image retrieval for image-based localization revisited. In BMVC, 2012.

[SXPK$^+$14] Eleftherios Spyromitros-Xioufis, Symeon Papadopoulos, Ioannis Kompatsiaris, Grigorios Tsoumakas, and Ioannis Vlahavas. A comprehensive study over VLAD and product quantization in large-scale image retrieval. IEEE Trans. on Multimedia, 2014.

[SZ03] Josef Sivic and Andrew Zisserman. Video Google: A text retrieval approach to object matching in videos. In ICCV, 2003.

[SZ13] Thomas Schneider and Michael Zohner. GMW vs. Yao? efficient secure two-party computation with low depth circuits. In Springer Berlin Heidelberg, editor, Financial Cryptography and Data Security, 2013.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. ICLR, 2014.

[TAJ13] Giorgios Tolias, Yannis Avrithis, and Hervé Jégou. To aggregate or not to aggregate: Selective match kernels for image search. In ICCV, Dec. 2013.

[TAS$^+$15] Akihiko Torii, Relja Arandjelovic, Josef Sivic, Masatoshi Okutomi, and Tomas Pajdla. 24/7 place recognition by view synthesis. In CVPR, 2015.

[TB14] Nicolas Tremblay and Pierre Borgnat. Graph wavelets for multiscale community mining. IEEE Transactions on Signal Processing, 62(20):5227–5239, 2014.

[TBI97] Lloyd N Trefethen and David Bau III. Numerical linear algebra. SIAM, 1997.

[TFP06] Hanghang Tong, Christos Faloutsos, and Jia Y Pan. Fast random walk with restart and its applications. In Proceedings of the IEEE International Conference on Data Mining, pages 613–622, 2006.

[TJ14] Giorgos Tolias and Hervé Jégou. Visual query expansion with or without geometry: refining local descriptors by feature aggregation. Pattern recognition, 47(10):3466–3476, 2014.

[TL09] P. Turcot and D. G. Lowe. Better matching with fewer features: The selection of useful features in large database recognition problems. In ICCV Workshop on Emergent Issues in Large Amounts of Visual Data, Oct. 2009.

[TLAF07] Marshall F Tappen, Ce Liu, Edward H Adelson, and William T Freeman. Learning Gaussian conditional random fields for low-level vision. In CVPR, pages 1–8. IEEE, 2007.

[TSF$^+$16] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. Commun. ACM, 59(2), 2016.

[TSJ16] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of cnn activations. ICLR, 2016.

[TSP11] Akihiko Torii, Josef Sivic, and Tomas Pajdla. Visual localization by linear combination of image descriptors. In ICCVW, 2011.

[TSPO13a] Akihiko Torii, Josef Sivic, Tomas Pajdla, and Masatoshi Okutomi. Visual place recognition with repetitive structures. In CVPR, Jun. 2013.

[TSPO13b] Akihiko Torii, Josef Sivic, Tomas Pajdla, and Masatoshi Okutomi. Visual place recognition with repetitive structures. In CVPR, 2013.

[Tuy10] Tinne Tuytelaars. Dense interest points. In CVPR, 2010.

[Vig09] Sebastiano Vigna. Spectral ranking. arXiv preprint arXiv:0912.0238, 2009.

[Vis12] Nisheeth K Vishnoi. Laplacian solvers and their algorithmic applications. Theoretical Computer Science, 8(1-2):1–141, 2012.

[VSKB10] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. Journal of Machine Learning Research, 11(Apr):1201–1242, 2010.

[WB07] Simon Winder and Matthew Brown. Learning local image descriptors. In CVPR, Jun. 2007.

[WC13] Rafi Witten and Emmanuel Candes. Randomized algorithms for low-rank matrix factorizations: Sharp performance bounds. arXiv preprint arXiv:1308.5697, 2013.

[WHB09] Simon Winder, Gang Hua, and Matthew Brown. Picking the best Daisy. In CVPR, Jun. 2009.

[WJ08] M.J. Wainwright and M.I Jordan. Graphical models, exponential families, and variational inference. Foundations and Trends in Machine Learning, 649, 2008.

[WJ15] Shuang Wang and Shuqiang Jiang. Instre: a new benchmark for instance-level object retrieval and recognition. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 11:37, 2015.

[WL11] Tobias Weyand and Bastian Leibe. Discovering favorite views of popular places with iconoid shift. In ICCV, 2011.

[WQSA15] J. Wang, G.-J. Qi, N. Sebe, and C. Aggarwal. Special issue on big media data: Understanding, search, and mining. IEEE Trans. on Big Data, 1(3), September 2015.

[WSB98] Roger Weber, Hans-J. Schek, and Stephan Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In Proceedings of the International Conference on Very Large DataBases, pages 194–205, 1998.

[WTF09] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In NIPS, Dec. 2009.

[XHZT15] Lingxi Xie, Richang Hong, Bo Zhang, and Qi Tian. Image classification and retrieval are one. In ICMR, 2015.

[XTZZ14] Lingxi Xie, Qi Tian, Wengang Zhou, and Bo Zhang. Fast and accurate near-duplicate image search with affinity propagation on the imageweb. Computer Vision and Image Understanding, 124, 2014.

[YKTL09] Xingwei Yang, Suzan Koknar-Tezel, and Longin Jan Latecki. Locally constrained diffusion process on locally densified distance spaces with applications to shape retrieval. In CVPR, 2009.

[YMD15] Fan Yang, Bogdan Matei, and Larry S Davis. Re-ranking by multi-feature fusion with diffusion for image retrieval. In WACV, 2015.

[ZBL$^+$03] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In NIPS, 2003.

[Zer34] von F Zernike. Beugungstheorie des schneidenver-fahrens und seiner verbesserten form, der phasenkontrastmethode. Physica, 1(7):689–704, 1934.

[ZGL03] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In ICML, 2003.

[ZK06] W. Zhang and J. Kosecka. Image based localization in urban environments. In 3DPVT, 2006.

[ZKLG06] Xiaojin Zhu, Jaz Kandola, John Lafferty, and Zoubin Ghahramani. Graph kernels by spectral transforms. Semi-Supervised Learning, pages 277–291, 2006.

[ZLG03] Xiaojin Zhu, John D Lafferty, and Zoubin Ghahramani. Semi-supervised learning: From Gaussian fields to Gaussian processes. Technical report, 2003.

[ZWG$^+$03] Dengyong Zhou, Jason Weston, Arthur Gretton, Olivier Bousquet, and Bernhard Schölkopf. Ranking on data manifolds. In NIPS, 2003.

[ZYC$^+$12] Shaoting Zhang, Ming Yang, Timothee Cour, Kai Yu, and Dimitris N Metaxas. Query specific fusion for image retrieval. In ECCV, 2012.

List of Tables

List of Figures

Résumé

Cette thèse étudie l'indexation et le mécanisme d'expansion de requête en recherche d'image. L'indexation sacrifie la qualité de la recherche pour une plus grande efficacité; l'expansion de requête prend ce compromis dans l'autre sens : il améliore la qualité de la recherche avec un coût en complexité additionnel. Nous proposons des solutions pour les deux approches qui utilisent une représentation continue d'un ensemble de vecteurs. Pour l'indexation, notre solution est basée sur le test par groupe. Chaque vecteur image est assigné à un groupe, et chaque groupe est représenté par un seul vecteur. C'est la représentation continue de l'ensemble des vecteur du groupe. L'optimisation de cette représentation pour produire un bon test d'appartenance donne une solution basée sur la pseudo-inverse de Moore-Penrose. Elle montre des performances supérieures à celles d'une somme basique des vecteurs du groupe. Nous proposons aussi une alternative suivant au plus près les vecteurs-images de la base. Elle optimise conjointement l'assignation des vecteurs images à des groupes ainsi que la représentation vectorielle de ces groupes. La deuxième partie de la thèse étudie le mécanisme d'expansion de requête au moyen d'un graphe pondéré représentant les vecteurs images. Cela permet de retrouver des images similaires le long d'une même variété géométrique, mais éloignées en distance Euclidienne. Nous donnons une implémentation ultra-rapide de ce mécanisme en créant des représentations vectorielles incorporant la diffusion. Ainsi, le mécanisme d'expansion se réduit à un simple produit scalaire entre les représentations vectorielles lors de la requête. Les deux parties de la thèse fournissent une analyse théorique et un travail expérimental approfondi utilisant les protocoles et les jeux de données standards en recherche d'images. Les méthodes proposées ont des performances supérieures à l'état de l'art.

Abstract

In this thesis, we study the indexing and query expansion problems in image retrieval. The former sacrifices the accuracy for efficiency, whereas the latter takes the opposite perspective and improves accuracy with additional cost. Our proposed solutions to both problems consist of utilizing continuous representations of a set of vectors. We turn our attention to indexing first, and follow the group testing scheme. We assign each dataset vector to a group, and represent each group with a single vector representation. We propose memory vectors, whose solution is optimized under the membership test hypothesis. The optimal solution for this problem is based on Moore-Penrose pseudo-inverse, and shows superior performance compared to basic sum pooling. We also provide a data-driven approach optimizing the assignment and representation jointly. The second half of the transcript focuses on the query expansion problem, representing a set of vectors with weighted graphs. This allows us to retrieve objects that lie on the same manifold, but further away in Euclidean space. We improve the efficiency of our technique even further, creating high-dimensional diffusion embeddings offline, so that they can be compared with a simple dot product in the query time. For both problems, we provide thorough experiments and analysis in well-known image retrieval benchmarks and show the improvements achieved by proposed methods.