Content Based Image Retrieval

Taieb Hadjkacem & Mohamed Chaaben

1 Introduction

La recherche d'images par le contenu, ou Content Based Image Retrieval (CBIR), est une technique qui permet de chercher des images en se basant sur leurs caractéristiques visuelles de bas-niveau. Cette technique est utilisée pour comparer une image à d'autres afin de trouver des résultats similaires.

La technique de recherche d'images par le contenu s'oppose à la recherche d'images par mots-clés ou tags, qui fut historiquement proposée par les anciens moteurs de recherche grâce à des banques d'images où les images sont retrouvées en utilisant le texte qui les accompagne plutôt que le contenu de l'image elle-même. En d'autres termes, c'est **le contenu** des images (couleur, forme, texture...) qui est l'objet de comparaison et non plus les métadonnées accompagnant celles-ci.

Cette technique permet de contourner les aléas dus à un enregistrement manuel des métadonnées : fautes de frappe, subjectivité, erreurs d'indexation, etc... et de rendre les recherches plus efficaces. De plus, elle permet également d'explorer des images non-indexées, comme celles d'une caméra de vidéo-surveillance.

Maintes sont les applications des systèmes CBIR.
Dans le domaine juridique, les services de police possèdent de gigantesques collections d'indices visuels (visages, empreintes) exploitables par des systèmes de recherche d'images. Les applications militaires, bien que peu connues du grand public, sont sans doute les plus développées : reconnaissance d'engins ennemis via images radars, systèmes de guidage, identification de cibles via images satellites, etc. Dans le domaine du journalisme et publicité, les agences maintiennent, en effet, des bases d'images afin d'illustrer les articles et les supports publicitaires. Les systèmes CBIR permettent également de protéger la propriété intellectuelle en détectant les contrefaçons dans le monde de l'art ou du commerce. Ils interviennent aussi pour le filtrage d'images pornographiques ou pédophiles, ou d'images non-appropriées. D'autres applications incluent le diagnostic médical, les systèmes d'information géographique, la gestion d'œuvres d'art, les moteurs de recherche d'images sur Internet et la gestion de photos personnelles.

L'objectif de ce projet est de concevoir un moteur de recherche web de type CBIR qui prend en entrée une image et donne les images les plus similaires de la base de données dans un temps raisonnable.

2 Architecture et implémentation

2.1 Architecture générale

Le système se compose de trois parties:

- **L'indexation des images** de la base de données en mode *offline* pour éviter, lors d'une recherche sur la base par un utilisateur, la charge élevée de calcul et de temps lors de l'analyse et l'extraction des descripteurs visuels.
- **Le traitement de la requête** en mode *online* qui correspond à l'extraction des descripteurs de l'image requête, le calcul de similarité par rapport à la base des index et l'affichage des résultats triés par ordre de pertinence.
- **L'interface utilisateur** qui permet à l'utilisateur de rentrer son image requête puis affiche le résultat de la comparaison avec les images indexées de la base de données. L'affichage du résultat se fait avec un ordre décroissant du taux de similarité calculée.

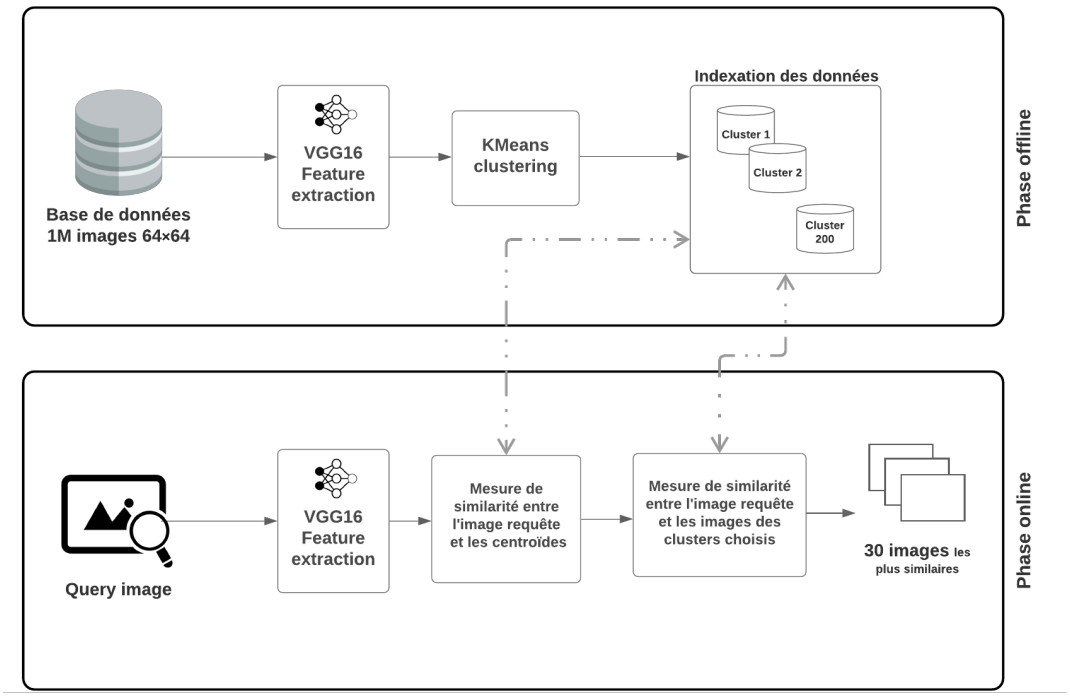

Figure 1: L'organigramme du système conçu

2.2 L'ensemble d'images

La base d'images est la donnée principale du système. Nous utilisons la collection MIRFlickr qui contient **1 million** d'images issues du site de partage de photos *Flickr*. Toutes les photos de cette collection ont été publiées par leurs utilisateurs sous une licence Creative Commons, leur permettant d'être librement utilisées à des travaux de recherche. La base de données est constituée d'images RGB variées, de taille 64×64 chacune, contenant divers objets et scènes. Le type d'images composant la base varie également : des portraits en noir et blanc, des peintures anciennes, images de tissus humains,etc. En fait, le type d'images influe fortement sur les performances globales du système, particulièrement sur les descripteurs de bas niveau calculés. D'une manière générale, plus la variabilité intra et inter images est importante, plus le système est apte à généraliser.

2.3 L'indexation

Contrairement à d'autres types de données, il serait naïf d'utiliser directement les images dans un CBIR étant donnée la taille importante de celles-ci. Au contraire, il faut les caractériser par des descripteurs.

Les méthodes d'extraction de descripteurs peuvent être pratiquement divisées en deux groupes: les descripteurs dits *handcrafted features* et ceux dits *deep features*.
Les descipteurs de type *handcrafted* sont extraits via des algorithmes soigneusement préconçus « à la main ». Par contre, les *deep features* sont obtenus par une procédure d'apprentissage profond.

Ces dernières années, dans le domaine de la vision par ordinateur particulièrement, les méthodes d'apprentissage profond se sont avérées plus efficaces que les techniques classiques dans plusieurs applications. Ceci dit, nous avons appliqué un réseau de neurones convolutif appelé VGG16 pour extraire des descripteurs profonds à chaque image de la base de données. Comme on a besoin de ce modèle uniquement comme extracteur de caractéristiques (et non pas un classifieur), nous avons supprimé la couche finale. Ainsi, à chaque image est associé un vecteur formé de 4096 éléments à partir d'un modèle VGG16 avec des poids pré-entraînés sur le datset ImageNet. Ce vecteur est ensuite enregistré dans un fichier NumPy.

- **VGG16**

C'est est une architecture de réseau neuronal convolutif (CNN) simple et largement utilisée pour ImageNet, un grand projet de base de données visuelle utilisé dans la recherche de logiciels de reconnaissance d'objets visuels. « VGG » est l'abréviation de *Visual Geometry Group*, qui est un groupe de chercheurs de l'Université d'Oxford et « 16 » implique que cette architecture comporte 16 couches.

Le modèle VGG16 atteint la précision du test 92,7% en ImageNet, qui est un ensemble de données de plus de 14 millions d'images appartenant à 1000 classes.VGG16 est utilisé dans de nombreuses techniques de classification d'images et est populaire en raison de sa facilité de mise en œuvre.

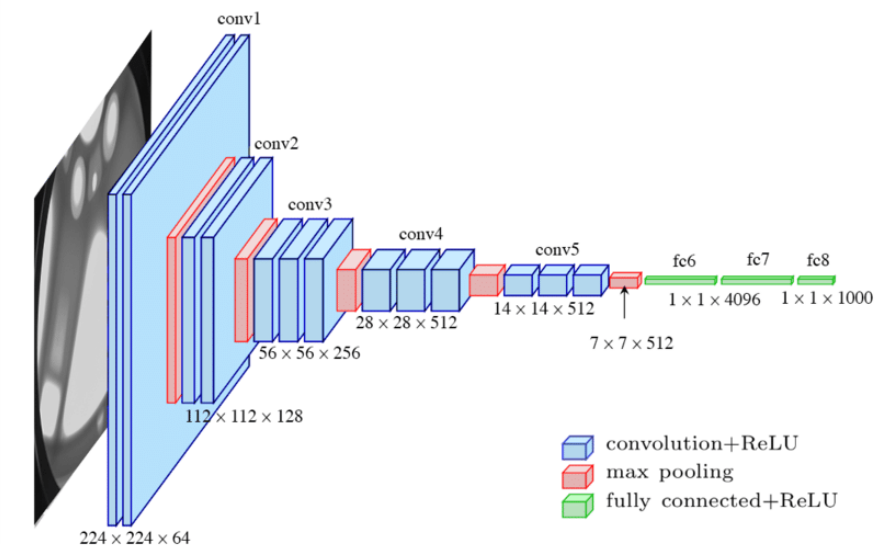

Figure 2: Architecture VGG16

Maintenant que nous avons formé un ensemble de caractéristiques pour nos images, nous sommes prêts à les regrouper en utilisant la méthode de clustering KMeans. KMeans est une méthode non supervisée de partitionnement de données en groupes homogènes ayant des caractéristiques communes: chaque cluster doit contenir des images visuellement similaires, et est référencé par un centroïde -calculé comme la moyenne des descripteurs d'un cluster- qui définit le centre du cluster.

Pour faire le compromis entre vitesse de recherche et précision des résultats, on a divisé l'ensemble de données en **200** clusters. À la fin de la tâche de clustering, nous avons à chaque cluster deux fichiers de type NumPy: un fichier contenant l'URL de chaque image dans ce cluster et un autre comportant les vecteurs caractéristiques correspondants.

- **Le type de fichier NumPy**

Il s'agit d'un format de fichier binaire standard permettant de conserver un seul tableau NumPy sur le disque. Le format stocke toutes les informations pour pouvoir reconstruire correctement le tableau, même sur une autre machine ayant une architecture différente. L'implémentation est destinée à être du Python pur et distribuée dans le cadre du package NumPy principal.

Le principal avantage de travailler avec le fichier NumPy est que le jeu de données conservera son format et nous n'avons pas besoin de spécifier et de réorganiser les valeurs du fichier texte externe. Les autres avantages de ce type de fichier qui nous intéressent le plus sont:

- Ils sont beaucoup plus rapides en lecture et écriture.
- Ils prennent moins d'espace mémoire que le fichier CSV correspondant.

À cet égard, [1], [2], [3] et autres benchmarks ont confirmé les bonnes performances obtenues en travaillant avec le format de fichier NumPy.

2.4 La recherche

Une fois l'utilisateur a entré une image, la phase de recherche commence. Celle-ci comporte deux phases: la phase de recherche des centroïdes similaires pour identifier les clusters cibles puis la phase de recherche d'images similaires dans ces clusteurs. D'abord, on calcule le descripteur VGG16 de l'image *input* puis on le compare aux 200 centroïdes pour mesurer la similarité, on choisit les 2 clusters les plus proches. Étant donnée que le centroïde ne donne pas une information exacte mais plutôt globale du cluster, la recherche dans un autre cluster donne des résultats meilleurs. Pour le calcul de similarité, la distance euclidienne est utilisée.

$$\|x\|_2 = \sqrt{|x_1| + ... + |x_n|} \tag{1}$$

2.5 Interface utilisateur

Il s'agit d'une interface web qui permet à l'utilisateur d'introduire son image requête et de visualiser les résultats de la recherche. La partie serveur est élaborée par Flask, c'est un *framework* léger de Python qui fournit des outils et des fonctionnalités utiles à la création d'applications web. L'interface renvoie à l'utilisateur les 30 images les plus pertinentes, et la distance euclidienne entre l'image *input* et l'image *output*.

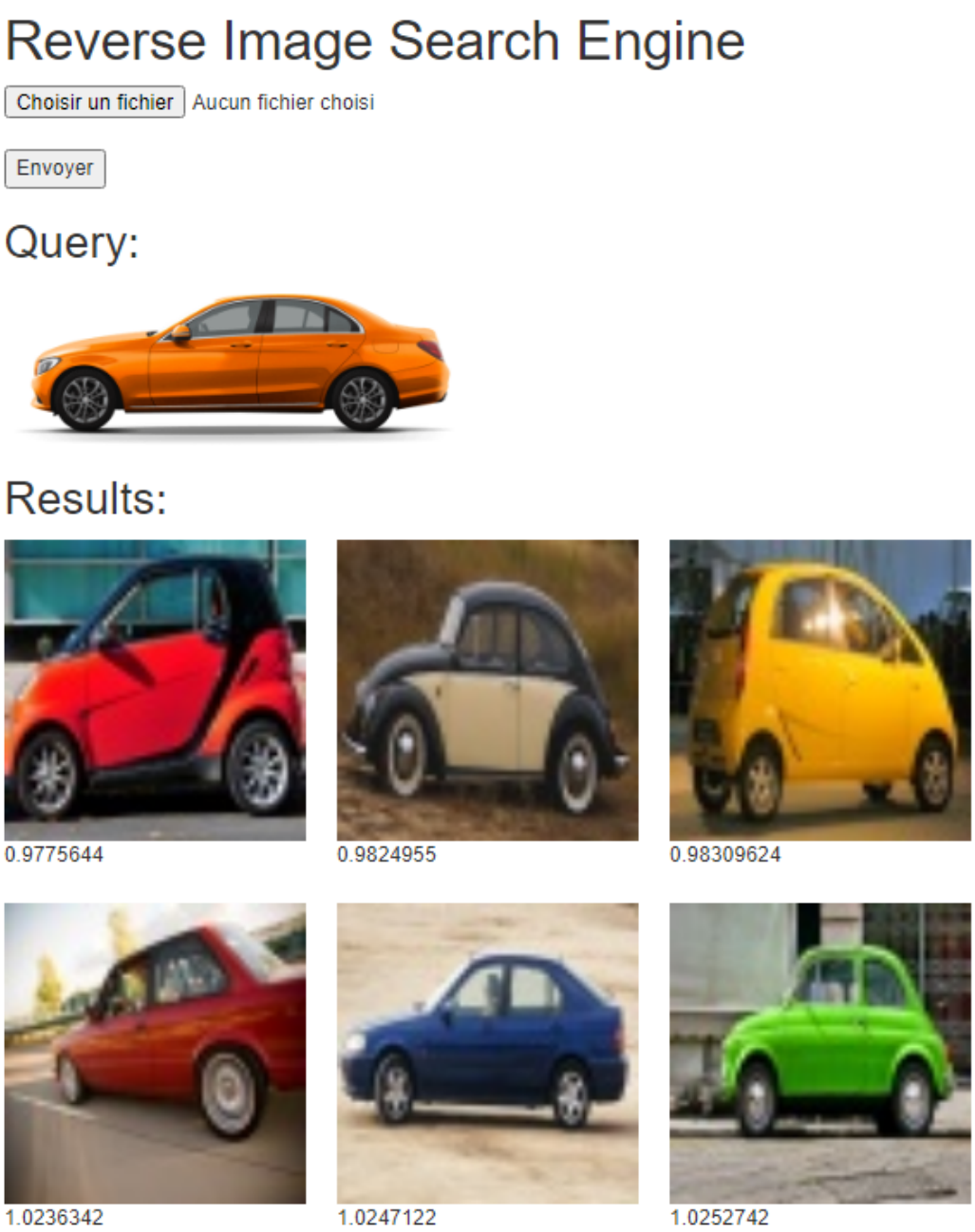

Figure 3: Interface web

3 Conclusion et perspectives

Tout au long de ce projet, on a fait la conception d'un système CBIR minimaliste en tirant profit principalement de l'architecture VGG16 et de la méthode de clustering KMeans.

Ce projet est, bien entendu, sujet à divers améliorations qui pourraient augmenter les performances de la chaîne, à savoir l'élargissement de la base de données pour avoir un moteur de recherche plus généraliste. En ce qui concerne le coté *user experience*, on peut ajouter une fonctionnalité permettant de choisir des caractéristiques telles que la couleur ou le type de l'image (image en trait, clipart, etc.) et ajouter l'option de recherche textuelle. En outre, une méthode d'évaluation de performances pourrait être envisagée telle que les métriques *Precison & Recall* en plus du calcul du temps de réponse.

Références

[1] **Nistrup**, Peter (7 mai 2019). « What is .npy files and why you should use them... », sur le site www.towardsdatascience.com. Consulté le 22 déc. 2021.
www.towardsdatascience.com/what-is-npy-files-and-why-you-should-use-them-603373c78883

[2] **KAMBARAKUN** (2018). « Dataset Load Speed - csv vs npy », sur le site Kaggle. Consulté le 22 déc.2021.
www.kaggle.com/kambarakun/dataset-load-speed-csv-vs-npy

[3] **Anindio**, Daneswara (2019). « Using NPY in NumPy instead of CSV », sur le site Medium. Consulté le 20 déc.2021.
www.medium.com/analyticbox/using-npy-in-numpy-instead-of-csv-f53865320687