Projet Image Mobile

Ayoub El Yagoubi
Benjamin Guilbert

Description

Pour le module « Project Informatique Image Mobile », on a eu l'opportunité de travailler sur une application de reconnaissance d'image. Le but était de développer une application Android qui permet d'identifier une marque selon une capturée ou sélectionnée via la bibliothèque. Puis les informations de cette marque sont communiquées à l'utilisateur grâce à la page internet de cette marque. (l'image de la marque, son nom , date de création ….).

Pour une première version, on devait appliquer un algorithme simplifié de reconnaissance d'image afin de s'initier dans la bibliothèque utilisée « JavaCV » et d'assurer l'intégration complète et opérationnelle de celle-ci dans notre application. Le but était de comparer l'image choisie (capturée ou sélectionnée depuis la galerie) avec une multitude d'images de références stockées sur l'appareil (le smartphone dans laquelle l'application est installée).

La deuxième version consistait à ne plus héberger les images de références sur l'appareil et de centraliser une base d'images dans un serveur distant. Cette deuxième version répondait au besoin de mise à jour d'images afin d'ajouter un nombre énorme d'images références, vu que dans la première version, c'était impossible suite à l'encombrement mémoire que cela peut impliquer sur l'appareil. L'algorithme de reconnaissance d'image était à changer aussi afin d'optimiser le temps de comparaison via une nouvelle méthode de classification.

Environnement de développement

- IDE : Android Studio
- Bibliothèques : JavaCV, OpenCV,Volley
- Gestion de versions de code : GitHub

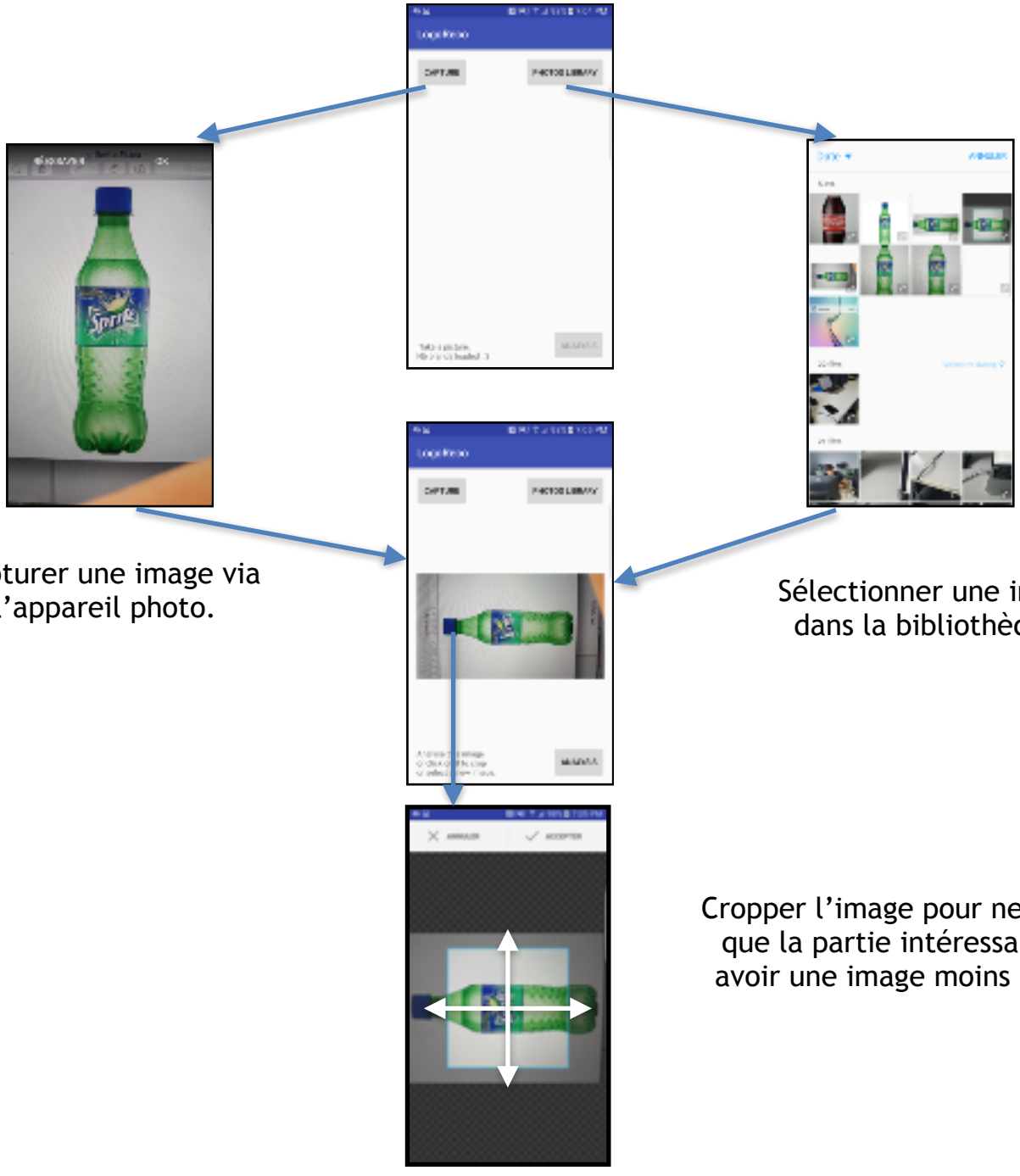
N'ayant jamais réalisé une application de reconnaissance d'image, les bibliothèques JavaCV et openCV ont été suggéré par les enseignants. Quant à GitHub, on a déjà eu la chance de l'utiliser aussi bien lors de notre formation académique que professionnelle.

Utilisation de l'application

Pour La bonne utilisation de notre application, voici les étapes qu'il faut respecter :

- ✓ Démarrer l'application.
- ✓ Capturer une image que l'on veut reconnaitre en cliquant sur le bouton « capture » ou « Photos Library ».
- ✓ Cropper l'image pour ne garder que l'essentiel
- ✓ Une fois satisfait de la qualité de la photo prise, Il faut cliquer sur le bouton « Analyser » afin de commencer la phase de reconnaissance d'image.
- ✓ Une fois finie, une image de la marque sera affichée et l'utilisateur sera redirigé vers un site web lorsqu'il cliquera sur l'image résultante contenant les informations sur le logo pris en photo.

Voici le schéma d'utilisation



Capturer une image via l'appareil photo.

Sélectionner une image dans la bibliothèque.

Cropper l'image pour ne garder que la partie intéressante et avoir une image moins lourde

Lancer l’analyse en cliquant sur « Analyse ».

Accéder au site web de la marque en cliquant sur l’image.

Ouverture du navigateur internet avec la page de la marque.

Architecture Logicielle

La classe Brand :

Elle contient toutes les informations de la marque. Toutes les informations sont récupérées sur le serveur web grâce au fichier index.json et sauvegardé dans les champs respectifs.

Elle dispose d'une méthode particulière « getClassifierFile », qui permet de télécharger et de sauvegarder le fichier Classifier de la marque en question. Elle prend en paramètre le contexte, l'url et la queue contenant toutes les requêtes vers le serveur.

Elle surenchérit la méthode « Equals » afin de comparer précisément le nom de chaque Marque et non l'objet en lui-même.

La classe ListOfBrands :

Elle contient tous les objets de type Brand. Elle étend la classe ArrayList<Brand> afin de re-coder la méthode contains aux spécificités de la classe Brand. Elle recherche si elle possède dèjà un objet Brand disposant du même nom.

La classe MainActivity :

Elle contient toutes les fonctions nécessaires à l'application pour télécharger les fichiers du serveurs Web, initialiser le vocabulaire, créer les objets Brands, cropper les images, analyser l'image sélectionnée, télécharger un fichier de référence pour afficher le résultat et ouvrir la page web.

Tout d'abord, l'application télécharge les fichiers index.json et vocabulaire.yml. Elle les enregistre toutes les marques dans des objets Brand dans une liste de type ListOfBrands. Un message, sur l'interface, affiche le nombre de brand téléchargées ou à défaut, un message d'erreur.

Puis, selon l'action effectué par l'utilisateur, elle fait appel à différentes méthodes ou fonctions.

La fonction analyse, fait appel à la méthode **setVocabulary** pour initialiser le dictionnaire des decriptors, puis traite l'image sélectionnée pour enfin faire appel à la méthode **downloadImage** pour télécharger une image représentant la marque résultante.

Traitement de l’image

Tout d’abord, on charge le vocabulaire dans un Mat à l’ouverture de l’application.

Puis, lorsqu’une action d’analyse est demandée:

- on initialise le SIFT
- on crée un matcher
- on initialise le dictionnaire grâce au vocabulaire
- on analyse l’image sélectionnée ou capturée en créant un Mat, les keypoints et l’histogramme
- on compare l’image avec le classifier de chaque marque chargée à partir du serveur
- on affiche le résultat en téléchargeant une image du serveurs web représentant la marque résultante.

Difficultés Rencontrées Vs leçons apprises.

Lors de la réalisation de cette application, On a eu plus de problèmes d'intégration que de développement. Au tout début, l'intégration de la bibliothèque JavaCV était problématique. On a eu du mal à bien entamer la phase de développement, surtout à distance. Tantôt c'était des erreurs dans le fichier graddle, tantôt des erreurs d'importations d'objets javaCV/OpenCv. Malgré cela, on a eu le réflexe de commencer l'intégration des fonctions qui permettent la reconnaissance d'images et le développement de la première partie de notre application en suivant le cahier de charge et les spécifications techniques et fonctionnelles données par les profs.

Malheureusement, suite à ces problèmes d'intégration, on n'a pas été capable d'envoyer une première version dans les délais. Pourtant, on était sûr d'avoir le code qu'il faut et que c'est à cause d'une erreur bête que ça ne fonctionnait pas. Suite au retour à l'école, M. Wanouss nous a proposé d'ouvrir un nouveau projet Android Studio et de copier/coller le code qu'on avait développé au fur et à mesure. Chose que nous avons faite et, magiquement, a marché. Ceci nous appris à toujours prendre du recul, ne pas hésiter à reprendre notre projet de zéro (From Scratch) étape par étape vu que c'est ce type d'erreur qu'on aura tendance a beaucoup rencontrer dans notre activité professionnel.

Ayant pris le réflexe de toujours bien vérifier les importations des librairies qu'on effectue ainsi que le log, la deuxième partie nous a été plus facile que la première. On avait bien compris comment récupérer les informations du serveur distant et pouvoir comparer les « **classifiers** » pour identifier la marque en question. On n'a pas eu de difficultés à intégrer la bibliothèque « Volley », qui est un Framework pour effectuer des requêtes http asynchrones, et c'était même agréable de pouvoir manipuler les classes qu'elle propose pour la récupération des fichiers ou images depuis le serveur distant. Cette partie nous aidé à manipuler du JSON, qui est un format de plus en plus utilisé, notamment en programmation web, afin de collecter de informations des systèmes tiers.

A noter aussi des petites difficultés techniques comme la taille des images capturées et la récupération du chemin de l'image enregistrée. La fonctionnalité de crop était d'une grande utilité pour contourner la première erreur. Quant à la première, l'implémentation de la fonction ToCache nous a débloquée.

Bilan

Suite à la réalisation de cette application, Mis à part un bagage technique important que ça soit en matière de programmation Android ou compréhension des algos de la reconnaissance d'image, on a appris à s'adapter aux différents problèmes rencontrées à cause de la distance et pouvoir rattraper du temps perdu en phase de débugeage et avancer dans le projet. La notion de Branches que GIT propose nous a été vraiment utile. Ça nous permettait de se répartir les tâches et même des fois travailler sur les même fonctionnalités avant d'optimiser le code et avoir une version master mature' était un plaisir de travailler sur cette application, malgré les différentes contraintes rencontrés, parce que le monde de maching Learning et data Science nous intéresse beaucoup et la reconnaissance d'image en fait partie.