TELECOM SudParis

Projet Informatique 1ère année
PRO 3600

Détection des styles architecturaux par une Intelligence Artificielle

Membres du projet : Elliot Cole, Brun Baptiste, Naama Mohamed, Abri Saad, Mateo Zoughebi

Mohamed Sellami CSC 3502 - Département INF

Version 18/02/2022

Table des matières

Introduction

Ce document décrit le développement d'une application permettant l'identification de styles architecturaux sur une photo, dans le cadre du module CSC-3502. Le développement de l'application implique deux domaines d'étude: L'Intelligence Artificielle (IA) et la création d'une application mobile. L'intelligence artificielle développée permet d'ajouter une nouvelle dimension aux applications d'identification des monuments par géolocalisation. Dans le cadre de ce module nous nous restreindrons à l'identification de bâtiments religieux pour la création de l'IA.

2 Cahier des charges

2.1 Partie application:

- *Fonctionnalités :*

L'application aura pour but principal de scanner un bâtiment de sorte à obtenir son style architectural.
Cette application devra :

- prendre en photo un édifice (scanner) à partir de l'appareil photo du téléphone
- obtenir des détails depuis cette photo (localisation, date de construction, histoire du bâtiment, autres photos…)
- garder en mémoire tous les scans effectués pour constituer des "memories"
- pouvoir gérer l'historique des scans (supprimer, réorganiser en dossier…)
- partager sur d'autres applications les scans effectués
- importer des photos à partir de la galerie sur lesquels on pourra aussi faire des scans
- indiquer la localisation de la personne en temps réel et être redirigé vers la Map

- *Contrainte:*

    - Apprentissage du langage de programmation Swift
    - Gestion de l'Intelligence artificielle depuis l'application

- *Livrable* **:**

    - Application mobile développée en Swift sur Apple (maquette en annexe 1).

2.2 Partie IA

- *Contraintes :*

    - Apprendre de façon complète un langage de programmation: Python avec les bibliothèques suivantes : torch, torchvision numpy, json,time, os, random, matplotlib, seaborn, PIL, argparse.
    - Constitution d'une base de donnée assez large pour pouvoir entraîner notre IA (database)

- *Fonctionnalités :*

- Pouvoir reconnaître le style architectural d'un bâtiment (dans un premier temps un édifice religieux) à partir d'une photo prise dans l'application mobile.

- Livrable :

    - Réseau de neurones convolutifs fonctionnel sous Python à l'aide de Pytorh.

3) Développement

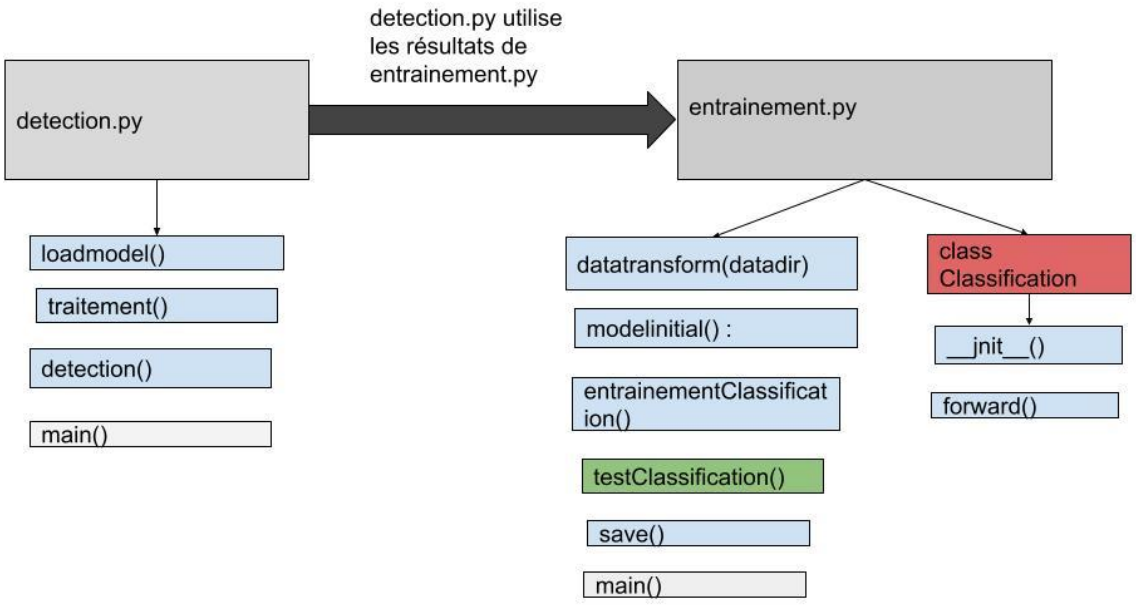
Conception préliminaire

A haut niveau, le logiciel se décompose en deux parties, une partie "front-end" : c'est la partie IHM de l'application, elle sera à terme une interface logiciel sur mobile écrite en Swift, mais dans un premier temps, on utilisera directement le terminal de l'IDE Python pour notre prototype.

La partie "back-end" est notre réseau de neurones convolutifs qui permettra de prédire l'architecture d'une photo que l'on enverra au programme.

Les structures de données utilisées seront ici des images d'architectures que l'on trouvera sur https://www.kaggle.com/brsdincer/architecture-art-cyclegan-process/data . Elles seront ensuite adaptées au format 224x224 pixels par souci d'homogénéité.

Ici un descriptif des modules de notre logiciel (partie "back-end") :



Fichiers (en gris foncé) : Le fichier *detection.py* sera celui qui prendra en entrée l'image de l'utilisation. Il utilisera pour cela le réseau de neurones convolutifs déjà entraîné, programmé dans le fichier *_entrainement.py*

Méthodes (bleu clair) :

main() : Méthode principale qui exécute le réseaux de neurones et responsable de l'affichage sur le terminal des prédictions.

datatransform(datadir) : Cette première méthode permettra de normaliser, et de traiter chacune des images du datadir (dossier d'images), afin de renvoyer 3 jeux de données en sortie : les jeux d'apprentissage, de validation et de test.

modelinitial() : La bibliothèque Pytorch fournit des modèles de réseaux de neurones pré-entraînés adaptés à notre problème afin de servir de base à notre réseau de neurones final. Elle renvoie un objet *model* de la classe torch

entrainementClassification() : C'est la méthode qui va nous permettre d'entraîner le modèle pour classifier les images. Elle implémente une méthode des gradients ainsi qu'une fonction de coût (en EQM) dans le but de calculer l'erreur sur le jeu de données d'entraînement/ de validation/de test, ainsi que la précision par rapport à la cible. Elle renvoie l'erreur sur les 3 jeux de données ainsi que la précision.

save() : Cette méthode sauvage le modèle ainsi entraîné, afin d'être utilisé dans *prediction.py.* Elle renvoie en sortie le chemin vers le fichier du modèle

loadmodel() : Elle prend un chemin de fichier du modèle de entrainement.py et renvoie un model (objet de la classe Pytorch).

traitement() : Elle prend l'image de l'utilisateur en entrée, et la traite (redimensionnement, normalisation) afin de pouvoir être utilisée par le programme. Elle renvoie une image.

detection() : Elle prend l'image en argument et renvoie une probabilité associée à une prédiction, en utilisant le modèle de réseaux de neurones.

Classes et méthodes d'instances (en rouge) :

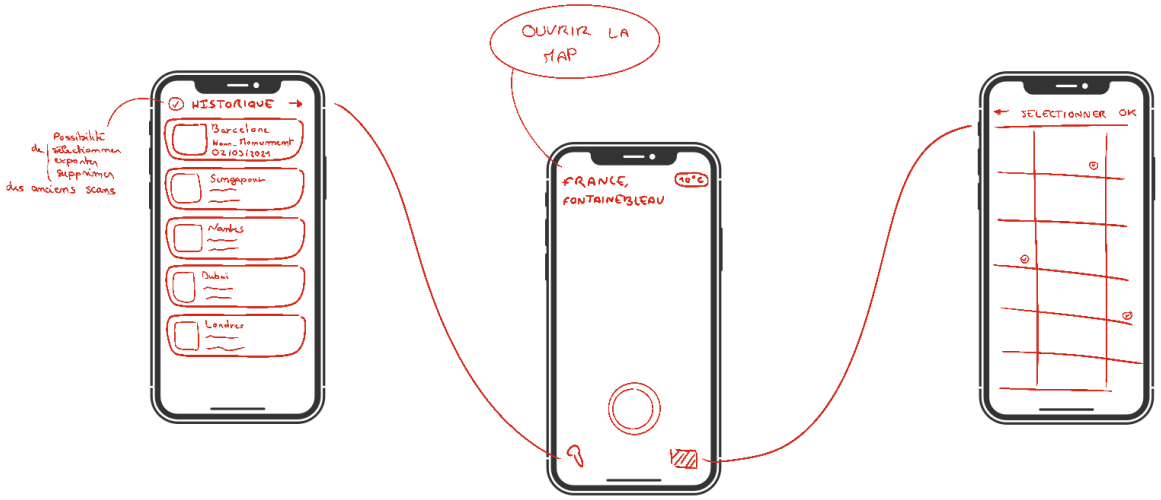
Classe *Classification* : Classe qui possède entre autres comme attributs la taille des entrées, des sorties et le nombre de couches du réseau de neurones associé.

forward() : C'est la méthode d'instance (de la classe *Classification*) qui prend en paramètre un input x (ici une image) et renvoie en sortie l'image ayant traversé toutes les couches du réseau de neurones.

Tests (en vert) :

testClassification() : Elle prend le jeu de données de test ainsi que le modèle en argument, et après le passage par le réseaux de neurones, elle renvoie la perte (résultat de la fonction de coût) ainsi que la précision

Annexe 1


OUVRIR LA MAP
HISTORIQUE
Barcelone
Nom_Monument
02/03/2024
Singapour
Nantes
Dubai
Londres
Possibilité de sélectionner/ supprimer des anciens scans
FRANCE,
FONTAINEBLEAU
10°C
SELECTIONNER OK