Ecole Publique d'Ingénieurs en 3 ans

Rapport

LOST IN CAMPUS 2

le 30 mars 2023,
version 1.1

DUCASTEL Matéo, COMBET Marceau, SENG Thomas, STEIMETZ Tangui,
mateo.ducastel@ecole.ensicaen.fr,
marceau.combet@ecole.ensicaen.fr,
thomas.seng@ecole.ensicaen.fr,
tangui.steimetz@ecole.ensicaen.fr

Tuteurs :
ROSENBERGER Christophe,
GIGUET Emmanuel,

ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

www.ensicaen.fr

TABLE DES MATIERES

TABLE DES FIGURES

PROJET

1. Présentation générale du projet et du contexte.

La ville de Caen a fait appel à un projet innovant pour fêter ses 1000 ans, en 2025, afin de se mettre en avant. Le Laboratoire GREYC veut y répondre avec le projet « *Lost in Caen* ».

Les chercheurs M. Rosenberger et M. Giguet nous ont chargé d'effectuer une première partie de ce projet : *Lost in Campus 2*. Il s'agit d'un projet de localisation par photographie, i.e. connaître sa position géographique dans le Campus 2, dans un environnement extérieur, à partir d'une image prise par un utilisateur (cf. figure 1) sans avoir recours aux métadonnées.

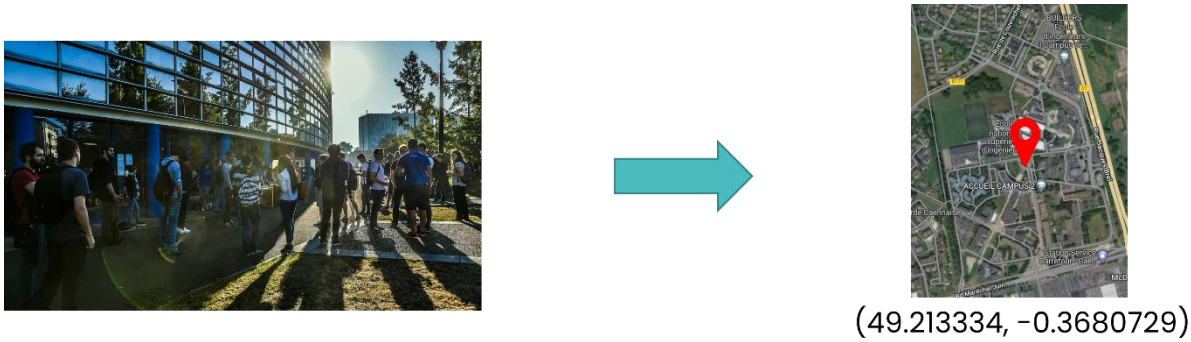

Figure 1 - Illustration du résultat attendu.

2. Réponse apportée.

Le projet utilise un réseau de neurones afin de pouvoir apprendre des informations depuis des images d'un jeu de données, dans le but d'estimer des coordonnées GPS depuis une nouvelle image. Initialement, nos tuteurs nous ont proposé le projet *GeoEstimation* comme base de travail, mais suite à plusieurs problèmes, nous nous sommes dirigés vers un autre projet, nommé *CosPlace* (cf. *GitHub* : https://github.com/gmberton/CosPlace). Ce dernier permet de prédire des coordonnées GPS depuis une image suite à un entraînement supervisé, c'est-à-dire que l'image et les coordonnées GPS associées sont connues.

3. Objectifs du projet.

Afin de commencer le développement des différentes parties du projet, nous devons préparer un protocole pour la prise de photographie. Celui-ci permet de s'assurer que le modèle a suffisamment de données pour pouvoir faire des prédictions correctes.

Une fois ce protocole établi, nous devons récupérer des images pour créer un premier jeu de données, dans l'optique de pouvoir faire des tests.

Ensuite, nous développons un script de traitement d'images qui transforme des images prises avec un appareil classique (smartphone ou autre) en images utilisables par le modèle. Nous commençons également à nous intéresser au modèle qui est utilisé par la suite dans le but de comprendre son fonctionnement, et comment l'entraîner correctement.

Pour vérifier que le jeu de données est complet, nous développons une *HeatMap* qui permet de détecter les zones n'étant pas suffisamment couverte. Nous introduisons également des outils qui permettent de tester le modèle, en affichant les résultats des estimations sur chaque section.

Finalement, des classes permettant d'effectuer des prédictions sont implémentées pour faciliter l'utilisation du modèle, ainsi qu'un exemple d'application web qui sert de base pour la suite du projet.

Le cahier des charges (cf. figure 2) présente les différentes contraintes que nous nous sommes fixées pour les objectifs cités précédemment, en lien avec le diagramme pieuvre (cf. figure 14, annexe).

Repère	FONCTION	CRITÈRE	NIVEAU
FP1	Doit permettre de trouver les coordonnées GPS d'une photo	Précision du programme Rapidité d'exécution	Précision de 10m Résultat en moins 5 secondes
FC1	Doit être accessible	Appareils compatibles Utilisation	CLI et éventuellement application web
FC2	Doit être open source	Licence	MIT
FC3	Doit respecter les normes Francaises et Européennes	Normes en vigueur RPGD	Normes Euro / FR Respect Norme RGPD Euro
FC4	Doit être maintenable et facilement extensible	Code propre Extensibilité	Appliquer les règles du code propre Possibilité d'élargir le maillage

Figure 2 - Cahier des charges fixé au début du projet.

4. Méthodologie et outils utilisés.

Nous prenons plusieurs précautions afin de nous assurer du bon déroulement du projet. Tout d'abord, un contact fréquent avec nos clients est nécessaire pour vérifier les préparatifs, et garantir le respect de leurs attentes. Ce contact se fait principalement via *Discord*, sur un serveur regroupant tous les projets encadrés par nos tuteurs, mais nous prenons régulièrement rendez-vous pour les informer de nos avancées. De plus, l'organisation doit être adaptée pour accepter de nouvelles données et des modifications. Enfin, nous veillons à avoir une communication fréquente entre les membres du groupe sur un autre serveur *Discord* pour se

répartir les tâches, demander de l'aide si l'un d'entre nous rencontre un problème, et partager nos idées sur le projet.

5. Répartition des tâches.

Lors de ce projet, nous nous répartissons les tâches comme suit.

Durant le premier semestre, nous nous sommes tous concentrés sur la recherche et la compréhension d'un modèle, la préparation d'une méthode d'acquisition des données, la mise en place du modèle, ainsi que la construction de l'ensemble de données. Deux d'entre nous se sont penchés sur l'analyse et le pré-traitement des données acquises.

Pendant le deuxième semestre, une personne s'est occupée de l'organisation et du partitionnement des données pour le modèle. Celui-ci a également développé un outil permettant d'évaluer une image prise par l'utilisateur selon le modèle, ainsi que l'entraînement d'un modèle. Un autre a créé un outil permettant de visualiser la couverture et la densité des données. Un troisième membre a développé un outil permettant de visualiser les performances du modèle par section, et le dernier a initié le travail sur *Google Colab*, et créé une application web pour pouvoir utiliser le modèle.

6. Travail réalisé.

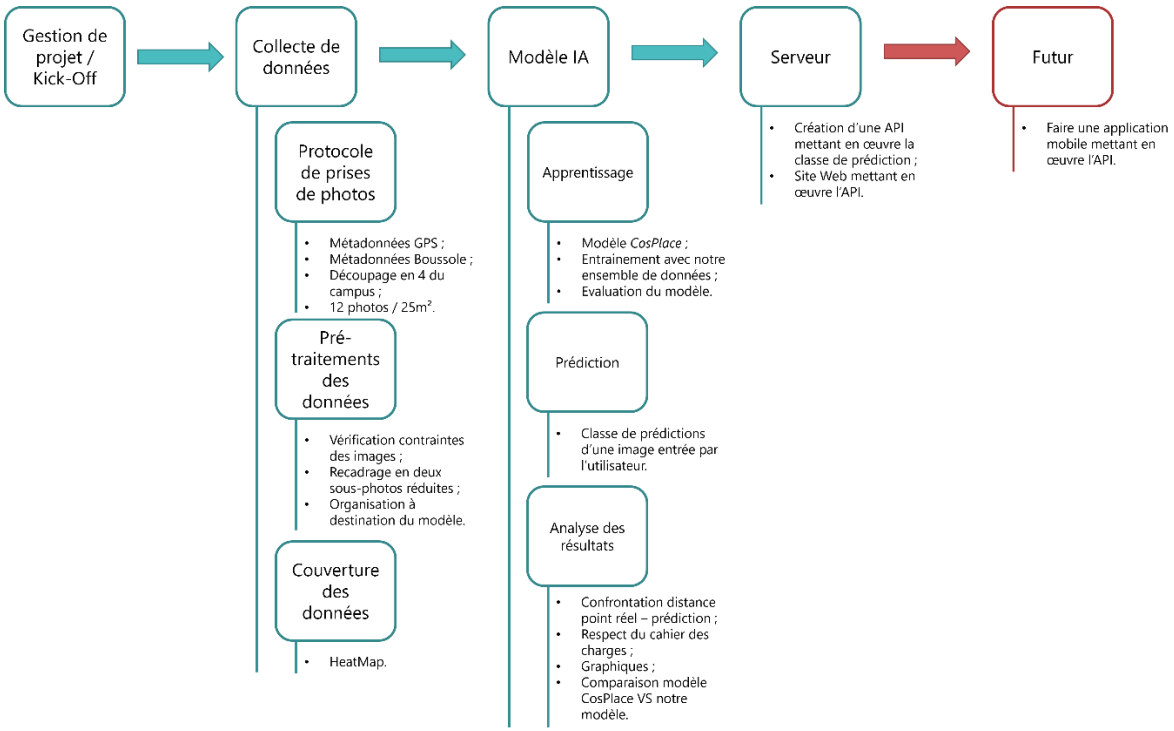

Figure 3 - Schéma global du travail réalisé.

Lien du *GitLab* associé aux travaux généraux : https://gitlab.ecole.ensicaen.fr/ducastel/lost-in-campus-2.

Lien du *GitLab* associé aux serveurs : https://gitlab.ecole.ensicaen.fr/ducastel/lic2-backend-django

6.1. Choix du modèle à utiliser pour la prédiction de la position GPS.

Nous faisons le choix d'utiliser le projet *CosPlace* (cf. *GitHub* : https://github.com/gmberton/CosPlace), libre de droit, que nous trouvons sur *GitHub*. Ce projet répond parfaitement à nos attentes et celles de nos tuteurs. De plus, celui-ci est récent et le créateur est très réactif, notamment pour répondre à nos interrogations.

Ce projet va nous permettre d'effectuer un entraînement via du *Deep Learning*, et de prédire des coordonnées GPS à partir d'une requête et d'un modèle entraîné.

Ce projet est assez particulier puisqu'il n'utilise pas une méthode conventionnelle pour prédire une coordonnée. Le projet *GeoEstimation* utilise la méthode *netvlad*, tandis que *CosPlace* a créé sa propre méthode, nommée d'après le nom de celui-ci. Les avantages proposés par cette méthode nous sont très favorables puisqu'elle permet un entraînement plus rapide, moins coûteux et, selon les dires de l'auteur, plus efficace, répondant à notre problème de matériels peu performants pour effectuer du *Deep Learning*.

Il nous faut cependant confectionner un ensemble de données plus important qu'avec *netvlad*, et qui soit surtout compatible avec les contraintes imposées par les modèles, à savoir que les images destinées à l'apprentissage doivent obligatoirement pointer vers le **Nord**. Il nous faut également adapter le code proposé pour répondre à nos besoins.

6.2. Confection de l'ensemble de données.

Afin de réaliser l'évaluation, il est nécessaire de confectionner une base de données composée d'images pour entraîner et/ou utiliser le modèle de reconnaissance d'images.

Le modèle sélectionné, *CosPlace*, utilise un ensemble d'images pour chercher des correspondances avec celle à évaluer.

Étant donné qu'il n'existe pas de base de données de la ville de Caen, encore moins du Campus 2, et que nous ne pouvons utiliser des API pour récupérer des images avec des métadonnées d'un lieu spécifique, car celles-ci sont payantes, nous devons prendre et traiter nos propres photographies du campus. Nous utilisons une application nous permettant de prendre des photographies tout en enregistrant les métadonnées, surtout pour les données de la boussole, qui ne sont pas enregistré par l'application *Appareil photo* intégré nativement aux smartphones *Android*. De plus, nous utilisons une autre application pour visualiser la boussole et prendre des photographies pointant vers le Nord.

6.3. Analyse du jeu de données.

Pré-traitement des images brutes :

Chaque script Python de pré-traitement créé est flexible, réutilisable et adaptable. Ils sont multi-threadés et affichent la progression des différents traitements.

Pour commencer, le traitement des images prises par un smartphone nécessite de vérifier certaines conditions pour que les images soient utilisables et classables :

- Les images corrompues ou ne possédant aucune métadonnée sont classées comme intraitables ;
- De même pour les images ne possédant pas de géolocalisation dans leurs métadonnées ;
- Il faut également renseigner les images en possession des données de la boussole puisque le modèle s'entraîne uniquement sur les images pointant vers le Nord (cf. article de recherche du projet *CosPlace*).

Cette étape est assurée par la classe *Verifier*. Elle possède les chemins des images correctement classées pour que la classe de traitements puisse être informée et guidée.

Dans un second temps, une fois que chaque image est classée et vérifiée, nous pouvons procéder aux traitements :

- Les images classées comme intraitables sont supprimées ou renommées suivant le choix spécifié lors de l'exécution du programme ;
- Si l'option de remise à zéro n'est pas spécifiée, les images déjà traitées ne sont pas traitées à nouveau ;
- L'étape de recadrage de l'image en deux sous-images :
    - Recadrage de l'image à la taille spécifiée (par défaut 512 pixels par 512, qui est la taille recommandée par le modèle *CosPlace*) ;
    - Les images sont recadrées en deux sous-images suivant le format Paysage ou Portrait de la photographie originale, ce qui permet d'augmenter le jeu de données et de conserver toute l'information après recadrage ;
- Enfin, il faut sauvegarder les images avec un nom particulier, puisque celles traitées sont dépourvues de métadonnées. Il faut alors renseigner toutes les informations dans le nom pour que le modèle y ait accès (en utilisant la convention renseignée par le projet *CosPlace*, cf. figure 4).

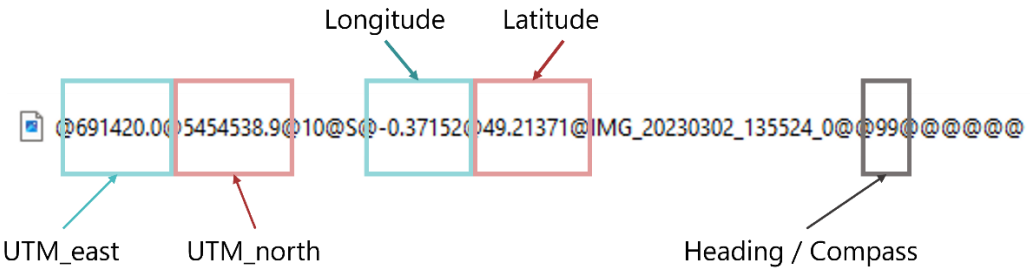

Figure 4 - Contraintes liées aux noms des images traitées, nécessaire pour le projet CosPlace.

Cette étape est effectuée par la classe *Preprocessor*. Elle nécessite l'utilisation du *Verifier* (voir étape précédente) pour fonctionner. Cette classe gère l'exécution du recadrage. Nous pouvons ainsi générer un total de 24 164 images.

Enfin, pour la dernière étape du pré-traitement, il faut organiser les images en une arborescence particulière, pour spécifier au modèle si les images sont à destination de l'entraînement, de l'évaluation du modèle en cours d'apprentissage, ou du test du modèle final. Chaque image ne peut être présente que dans un unique sous-dossier, pour éviter de biaiser les résultats.

Dans cette perspective, la classe *Organizer* est confectionnée pour répondre à ces contraintes et organiser l'ensemble des données (cf. figure 5), afin d'utiliser le projet *CosPlace*. Cette classe organise les images de manière aléatoire suivant ces contraintes :

- Si le jeu de données à organiser est à destination d'un entraînement ou non. C'est-à-dire s'il faut créer un sous-dossier d'entraînement et d'évaluation, et y ranger des images. Si nous désirons uniquement tester un modèle, nous pouvons simplement les organiser avec un unique sous-dossier.

- Il faut également renseigner des pourcentages pour le remplissage des sous-dossiers :

    - Un pourcentage pour spécifier la proportion d'images pointant vers le Nord, à utiliser pour l'apprentissage du modèle ;

    - Un pourcentage pour spécifier la proportion d'images à utiliser pour l'évaluation. Le restant est utilisé pour le test ;

    - Un pourcentage pour spécifier la proportion d'images à utiliser pour la base de données, parmi celles sélectionnées pour tester ou évaluer. Le restant est utilisé pour les requêtes de prédiction.

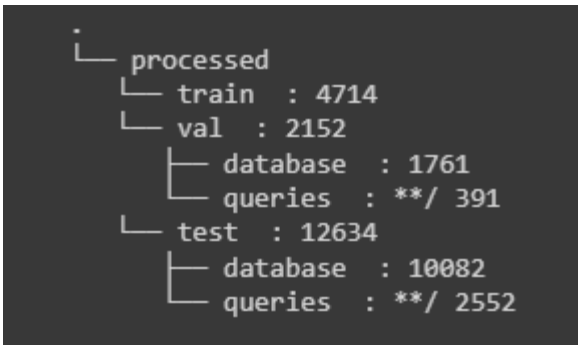

Figure 5 - Exemple d'arborescence créé par l'Organizer à destination d'un entrainement avec le nombre d'images associées à chaque sous-dossier.

6.4. Couverture par l'ensemble de données

Il est nécessaire de vérifier que notre jeu de données réponde à différents critères.

Il faut d'abord vérifier qu'il couvre toutes les zones piétonnes du Campus 2, et qu'il ait un nombre minimal de photographies par section. Ceci est fait dans le but d'anticiper les problèmes qui pourraient survenir lors de l'apprentissage, comme une section insuffisamment couverte.

Pour cela, une *HeatMap* est réalisée. Cette carte permet de visualiser la densité des photographies sur le campus (Cf. figures 6 et 7). Nous pouvons ajuster différents paramètres, comme la taille des sections (par défaut, surface de 5 m par 5 m), ainsi que le nombre minimal de photographies par section (12 images par défaut). Il est aussi possible de choisir d'afficher la carte en fond.

Une autre version avec un affichage qui ne tient pas compte de la répartition en sections précédente est disponible (cf. figure 7).

Il est alors possible d'analyser la couverture des catégories de données utilisées par le modèle, à savoir l'entraînement, l'évaluation et le test, simultanément ou non.

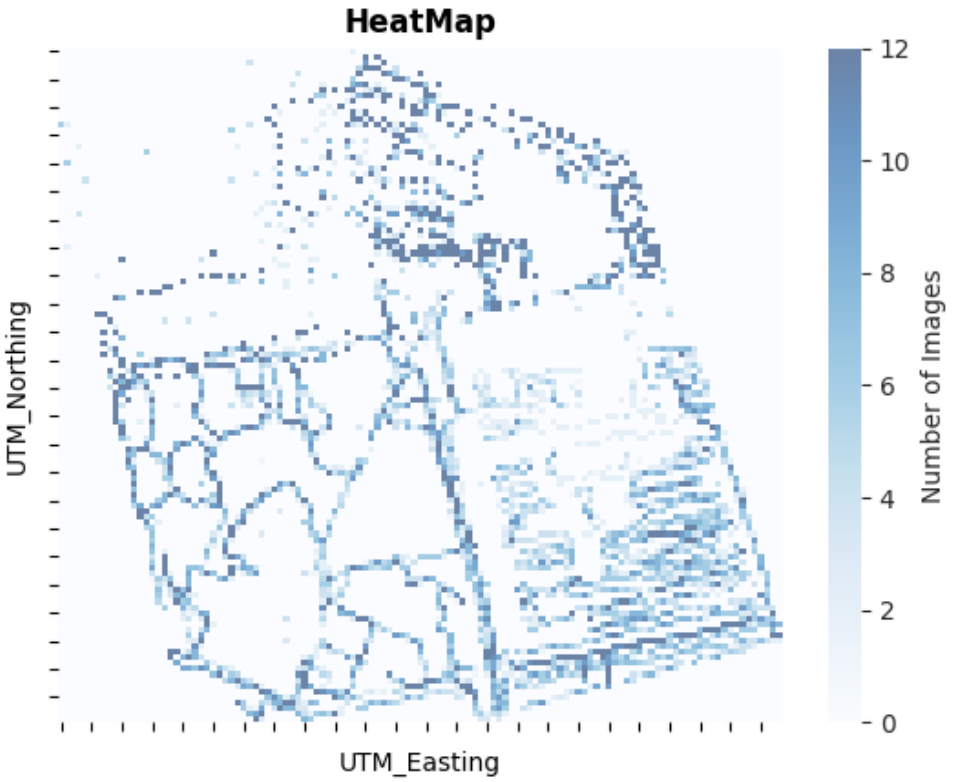

Figure 6 - HeatMap de notre ensemble de données (premier affichage)

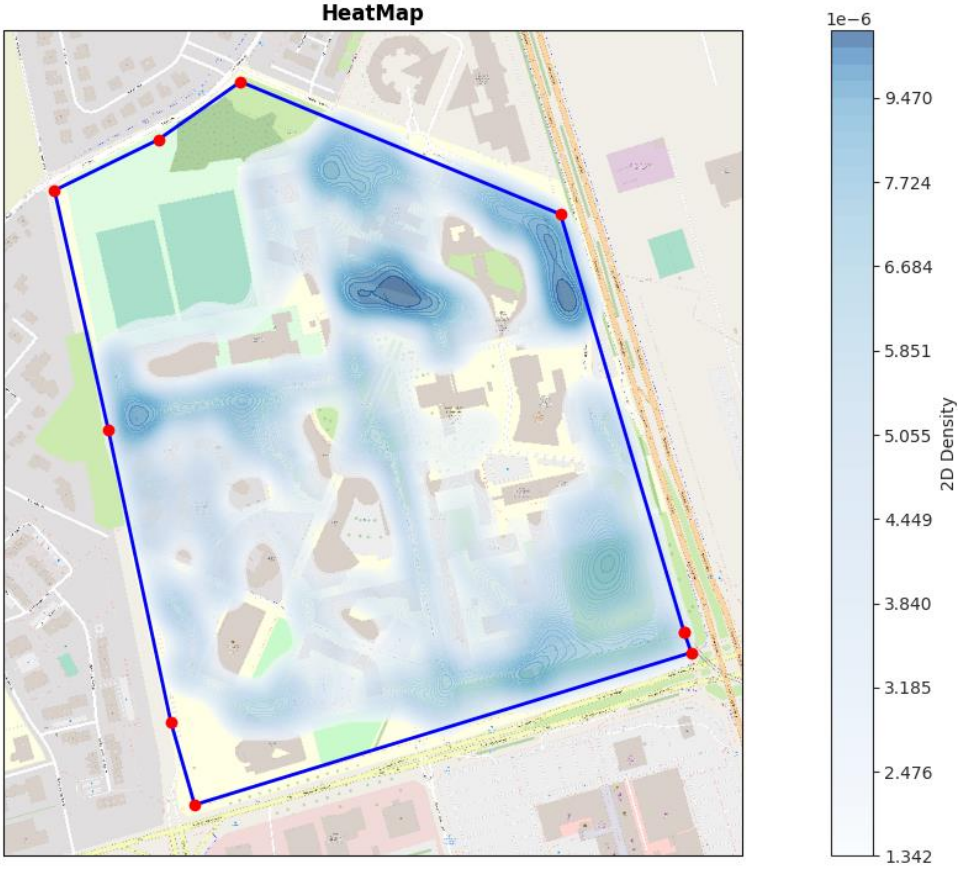

Figure 7 - HeatMap de notre ensemble de données (second affichage)

6.5. Entraînement du modèle

L'entraînement du modèle fonctionne en 3 phases : Apprentissage, Validation et Test.

Pour chacune des phases, nous retrouvons un dossier qui contient l'ensemble des images sur lesquelles le modèle va extraire des descripteurs, et un autre dossier contenant celles que nous essayons d'estimer. Un descripteur est un vecteur qui contient des informations que le modèle apprend à créer à partir d'une image. Ils sont ensuite utilisés pour évaluer les nouvelles images.

Dans un premier temps, la phase d'Apprentissage ajuste les paramètres du réseau de neurones, en comparant son estimation aux coordonnées réelles. Cette phase est aussi nommée *epoch*.

Ensuite, la phase de Validation teste différentes configurations du réseau de neurones précédemment généré durant les *epochs*. Celui avec le meilleur résultat est conservé. Les paramètres modifiés lors de cette phase s'appellent des hyperparamètres, et correspondent à différents moyens de créer les groupes de photographies.

Finalement, la phase de Test permet d'obtenir des résultats de performances du réseau de neurones sur des images jamais vues auparavant, afin d'éviter les biais d'apprentissage.

Suite à cela, nous obtenons un fichier qui contient les paramètres de notre modèle. Nous pouvons ensuite l'importer dans nos applications afin de pouvoir effectuer des estimations à l'aide des classes d'évaluation décrites ci-après.

6.6. Classe d'Évaluation de l'application finale.

Le but de cette classe est de remplir la fonction principale de l'application, à savoir fournir une prédiction de géolocalisation suivant l'image passée en entrée. Cette classe est chargée côté serveur.

Celle-ci est basée sur l'évaluation utilisée par le modèle. Il est nécessaire, dans un premier temps, de charger les descripteurs des images de la base de données en utilisant un modèle entrainé. Cette étape est extrêmement coûteuse en temps : en utilisant uniquement le processeur sans accélérateur graphique, cette étape peut prendre plusieurs jours.

Une fois chargés, nous pouvons ainsi comparer l'image à prédire avec les descripteurs de la base de données, trouver l'image la plus proche, et retourner la géolocalisation présente dans le nom de celle-ci en un temps limité (cf. figure 8).

```
Image to evaluate : ../../dataset/processed/test/queries/section_id_58/@691477.11@5454259.33@10@S@-0.37087@49.21118@IMG_LOST_20230216_160515_0@@14@@@@@@.jpg
Using base64 to evaluate, simulation of a server request. True
Creating Dataset from database folder : ../../dataset/processed/test/database
Creating Evaluator from previous Dataset
Extracting database descriptors: 100%|████████████████████| 101/101 [03:26<00:00,  2.04s/descriptor]
 -- Evaluation --
UTM_prediction = POINT (691476.38 5454259.31)
UTM_truth = 691477.11, 5454259.33
 - Diff UTM_east : 0.7299999999813735 meters
 - Diff UTM_north : 0.02000000048428774 meters
```

Figure 8 - Exemple d'évaluation, en simulant une requête au serveur en choisissant une image aléatoirement.

6.7. Classe d'Évaluation du modèle par section

Nous implémentons une classe afin de tester notre modèle par section de la zone de travail, formant une grille de taille spécifiée lors de l'organisation des données. Cette évaluation se distingue de celle de la classe précédente par le nombre d'images testées. Le but est d'évaluer la performance du modèle entrainé sur notre ensemble de données à estimer.

La classe récupère les données nécessaires à son fonctionnement, et initialise les attributs indispensables aux différentes étapes des traitements. Ainsi, les paramètres et chemins vers les différentes sections que l'utilisateur veut évaluer sont stockés. L'évaluation par section peut se faire sur une liste de sections donnée en argument, une liste de sections construite par tirage aléatoire d'un certain nombre de sections (définissable à l'aide d'un argument, 5 sections par défaut), ou sur la totalité des sections du dossier testé.

Lors de l'exécution, chaque section est parcourue afin de récupérer les images et effectuer leur évaluation. Les distances entre les positions réelles et prédites sont calculées puis sauvegardées, les temps de prédiction également, et les différentes fonctions permettant d'obtenir l'affichage d'une carte du Campus 2 sont appelées. Cette carte est marquée par les positions réelle et prédite de chaque image, accompagnées par une ligne de distance, ainsi qu'un message affichant la distance entre les deux coordonnées (cf. figure 9).

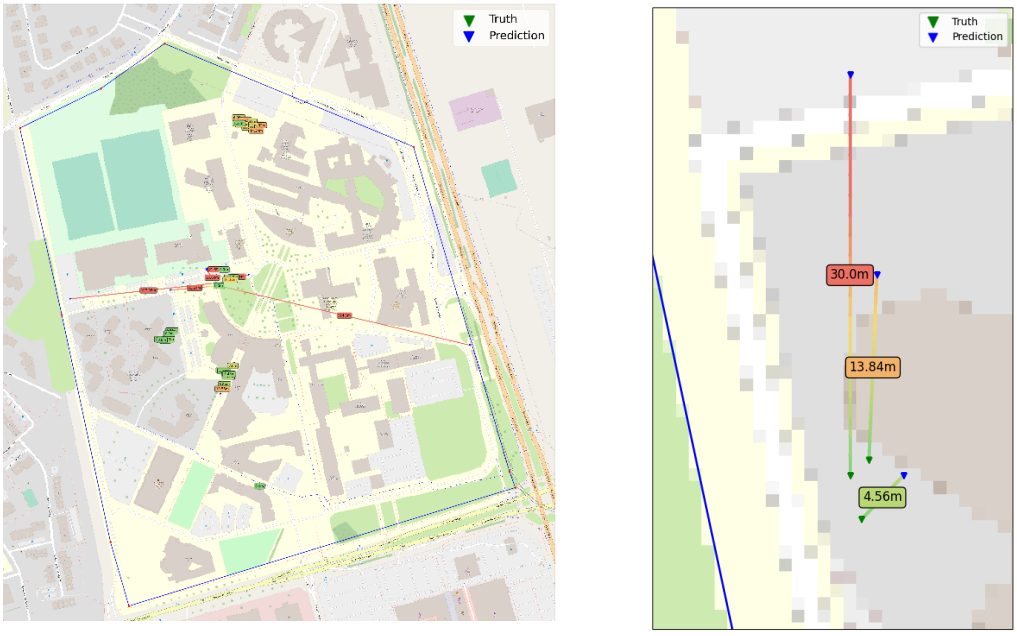

Figure 9 – A gauche, exemple d'évaluation graphique d'un modèle ResNet50 avec 2048 descripteurs entrainé par CosPlace sur 5 de nos sections choisies aléatoirement. A droite, zoom sur certaines lignes de distance.

Il est également possible de remplacer les marqueurs et lignes de distance par des messages, indiquant la moyenne des temps de prédiction d'une image, et des diagrammes circulaires, représentant la proportion de prédictions vérifiant la condition « *Inférieur à 10 m* » pour chaque section. De plus, cette fonctionnalité est également compatible avec la possibilité de traiter toutes les sections d'un dossier (cf. figure 10).

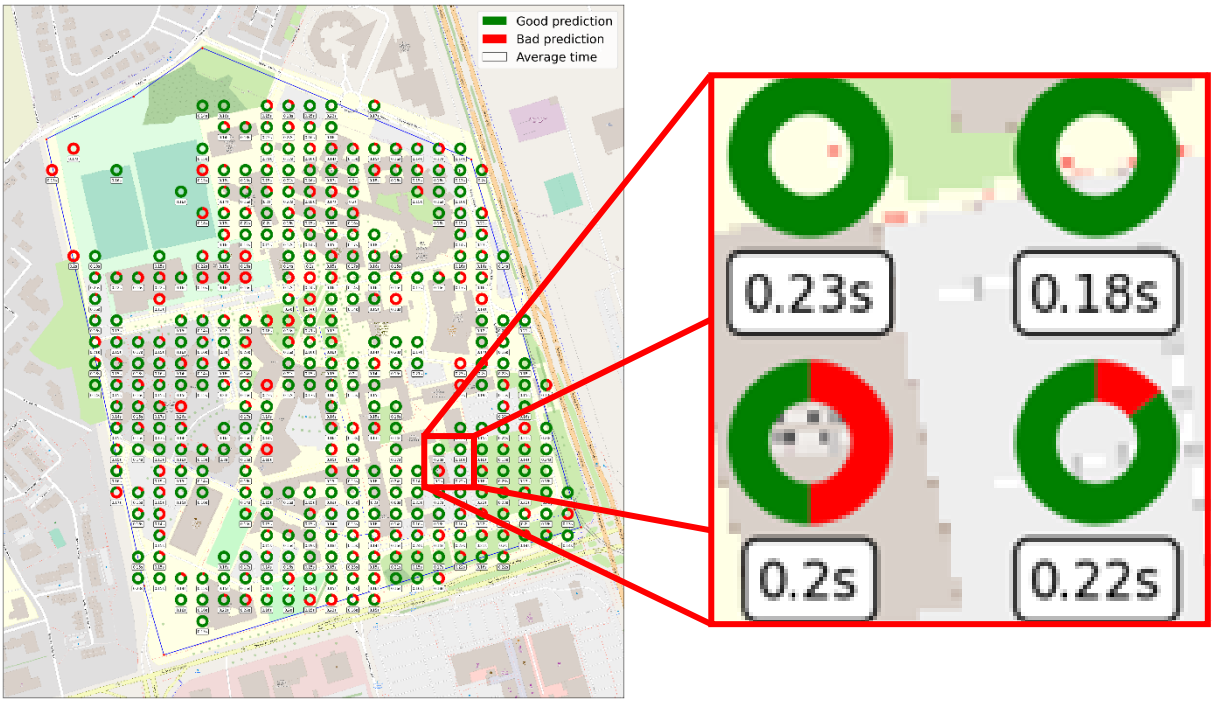

Figure 10 - Exemple d'évaluation graphique d'un modèle ResNet50 avec 2048 descripteurs, entrainé par CosPlace sur la totalité de nos sections, affichage des résultats par section seulement.

Par ailleurs, le nombre d'images à traiter pouvant être important selon la quantité de sections évaluées, les temps d'exécutions peuvent s'avérer relativement longs. Ainsi, à l'instar des scripts de pré-traitement, l'évaluation repose sur du multithread et la progression est également afficher (cf. figure 11).

```
Using local model : /content/gdrive/MyDrive/LIC/CosPlace/trained_models/ResNet50_2048_lic2.pth -- ResNet50 -- 2048
Model is loaded
Creating Dataset from database folder : /content/gdrive/MyDrive/dataset/processed/test/database
Creating Evaluator from previous Dataset
Extracting database descriptors: 100%|████████████████████| 101/101 [03:17<00:00,  1.96s/descriptor]
Evaluated sections: 100%|██████████| 338/338 [01:30<00:00,  3.75sections/s]
```

Figure 11 - Exemple d'affichage de l'exécution du programme d'évaluation par section

6.8. Résultats obtenus après Évaluations par section

Après une évaluation sur l'ensemble des sections avec les deux modèles, pré-entraîné par *CosPlace* et entraîné par notre équipe, nous obtenons deux résultats importants à analyser (cf. figure 12).

Figure 12 - Résultats obtenus après évaluations des deux modèles. A gauche, les résultats de notre modèle. A droite, les résultats du modèle pré-entraîné, fourni par CosPlace.

Le premier résultat correspond à la moyenne de toutes les distances calculées sur l'ensemble des sections sélectionnées (ici, toutes les sections dont nous disposons). Nous remarquons que le modèle pré-entraîné atteint une moyenne qui satisfait notre critère « Inférieur à 10 mètres », consigné dans le Cahier des Charges. Notre modèle, quant à lui, ne remplit pas cette condition. Cependant, il atteint une moyenne relativement proche de notre critère, ce qui est encourageant compte tenu du temps d'entraînement assez court qui lui a été consacré. Il est à noter que ces valeurs peuvent varier, mais, selon nos tests, celles-ci oscillent entre 13 m et 15 m pour notre modèle, et entre 9.5 m et 10 m pour le modèle pré-entraîné.

Le second résultat correspond à la moyenne des temps de prédiction de chaque image calculée sur l'ensemble des sections sélectionnées (une fois encore, toutes les sections dont nous disposons dans notre cas). Les deux modèles ont des temps similaires, de l'ordre du dixième de seconde. Notre condition « Inférieur à 5 secondes » est donc largement validée.

6.9. Back End avec API et sa page web associée.

À la suite de la création des classes utiles pour l'évaluation, nous créons un back end en utilisant *Django* (Framework back end pour Python). Nous faisons ce choix pour plusieurs raisons. Tout d'abord, il utilise Python, nous pouvons donc simplement récupérer les classes précédentes. Ensuite, il met en place une API REST avec *Django Rest Framework*, ce qui permet à l'utilisateur de récupérer une estimation de coordonnées GPS en format JSON.

Le serveur permet, dans un premier temps, de charger la classe d'évaluation avec tous les descripteurs de la base de données, avant de pouvoir commencer l'évaluation. Le but étant d'instancier cette classe une unique fois, pour conserver notre contrainte de temps de réponse ou prédiction.

Finalement, ayant déjà de l'expérience avec *Django*, la mise en place du backend s'en trouve faciliter.

Deux URLs sont valides pour notre back end :

- **/** : renvoie une page web afin que nous poussions envoyer l'image de notre choix, et le résultat est affiché (cf. figure 13).

- **/api/?url=XXXX** : (GET) prend en paramètre un lien vers une image, et renvoie un JSON contenant les prédictions du modèle (cf. figure 14).

- **/api/** : (POST) prend les données binaires d'une image, et renvoie un JSON contenant les prédictions du modèle. (Un script pour tester est présent dans le git).

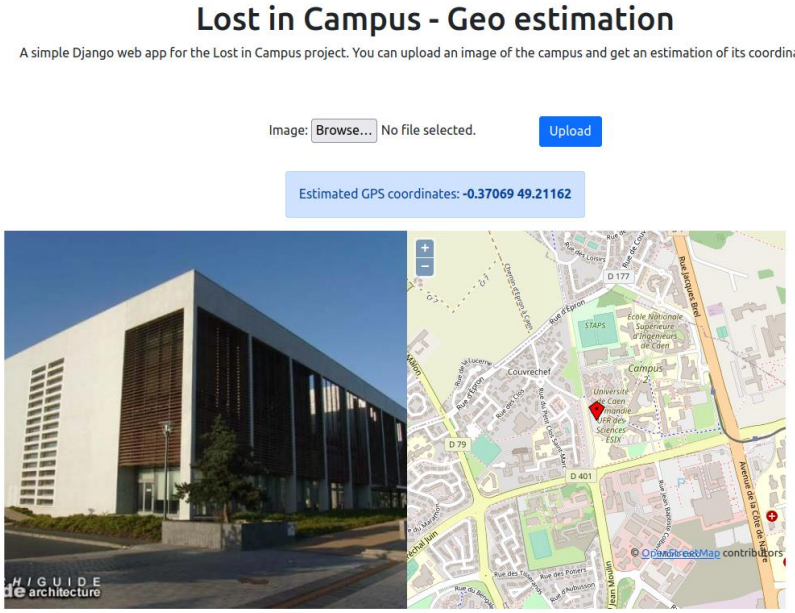

Figure 13 - Exemple de résultat de la page web.

Figure 14- Exemple de résultat associé à l'API créée.

Le back end n'est pas déployé sur un serveur, il fonctionne donc en local. Il est cependant une bonne ressource pour une future extension du projet à la ville de Caen, car nous pouvons aisément rajouter des fonctionnalités afin de respecter l'application finale sur l'ensemble de la ville de Caen, comme voulu.

BILAN

7. Objectifs non atteints, motifs et conséquences.

La construction du jeu de données n'a pas pu se faire avec l'aide des étudiants de l'ENSICAEN nous déposant leurs images via un formulaire *Google Form*, étant donné que les métadonnées sont perdues lors du dépôt.

Nous n'avons également pas eu le temps d'implémenter des filtres lors du pré-traitement des images. Ceux-ci auraient permis d'avoir plus d'images pour l'entraînement, sans avoir à prendre plus de photographies sur le terrain.

8. Suite du projet.

Ce projet est prévu pour être restreint au Campus 2, mais dans l'objectif de pouvoir l'étendre à toute la superficie de Caen, dans le cadre du projet initié par le laboratoire GREYC à l'occasion des 1000 ans de la ville. La forme exacte, pour l'utilisation de cette application, n'étant pas encore déterminée, nous avons donc fait le choix de développer plusieurs outils pour contrôler l'entraînement grâce à la *HeatMap*, analyser le modèle via les estimations, et donner un aperçu d'utilisation du modèle avec le serveur web *Django*.

Ces différents sous-projets ont été développés dans l'objectif d'assurer une réutilisabilité optimale, ce qui permet aux personnes qui continueront le projet de partir sur des bases solides pour développer une application efficacement.

ANNEXE

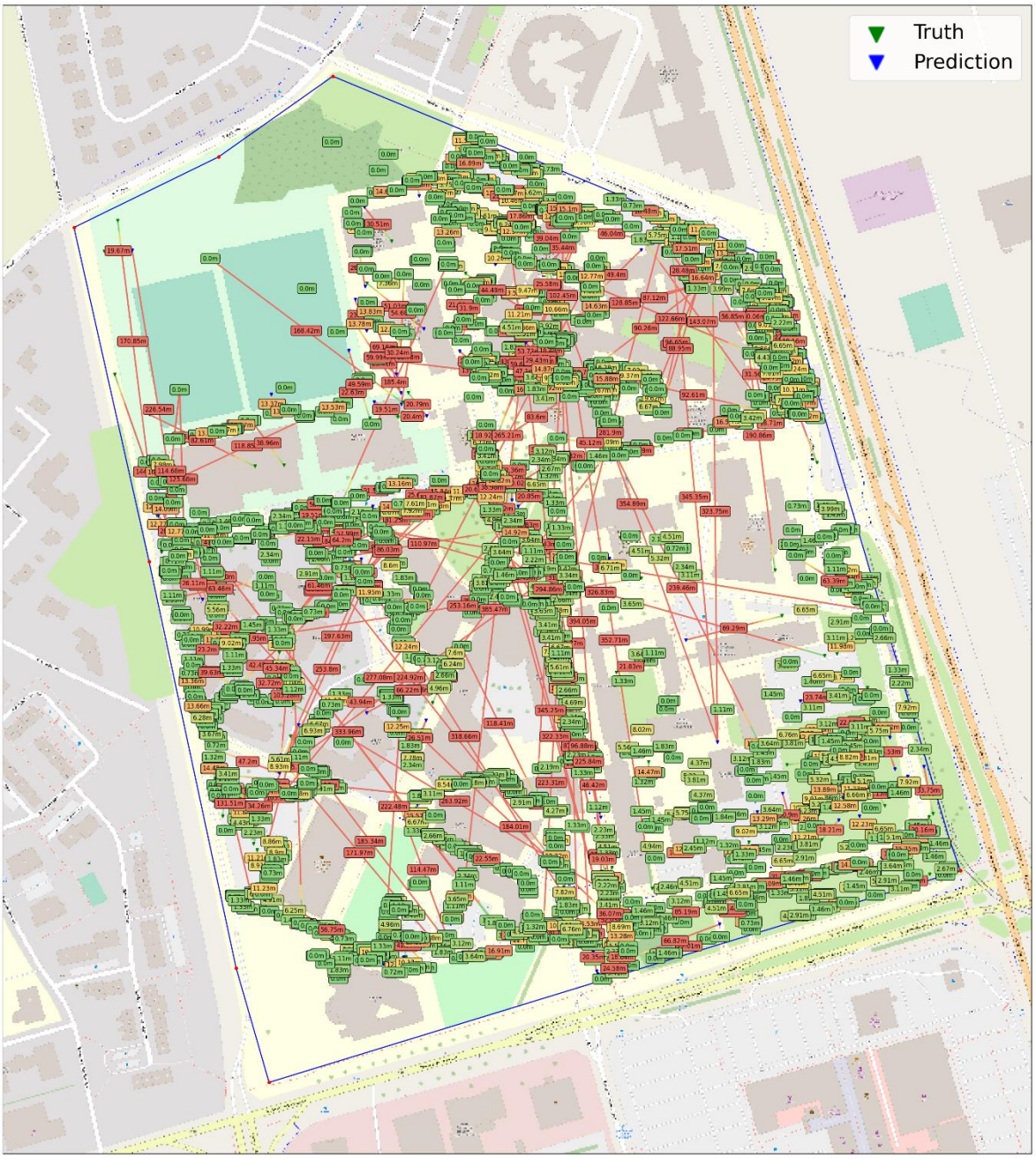

Figure 15 - Évaluation graphique d'un modèle ResNet50 avec 2048 descripteurs entrainé par CosPlace sur la totalité des sections

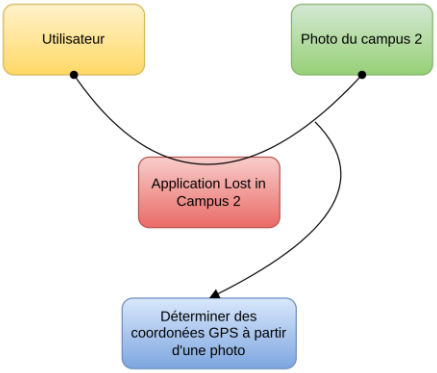

Figure 16 - Bête à corne.

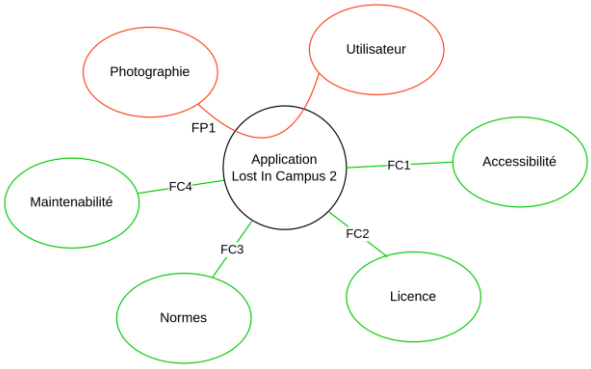

Figure 17 -Diagramme pieuvre.

```
@InProceedings {
        Berton_CVPR_2022_CosPlace,
        author = {Berton, Gabriele and Masone, Carlo and Caputo, Barbara},
        title = {Rethinking Visual Geo-Localization for Large-Scale Applications},
        booktitle = {Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern
Recognition (CVPR)},
        month = {June},
        year = {2022},
        pages = {4878-4888}
}
```

ENSI CAEN

ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Ecole Publique d'Ingénieurs en 3 ans

6 boulevard Maréchal Juin, CS 45053
14050 CAEN cedex 04

ministère éducation nationale enseignement supérieur recherche

Normandie Université

Cti Commission des titres d'ingénieur

cdefi Conférence des Directeurs des Écoles Françaises d'Ingénieurs

European Accreditation of Engineering Programmes EUR-ACE®

CONFÉRENCE DES GRANDES ÉCOLES