R. JALILI
L. PLANTIVE
MMI3 TP3

Projet : Authenticité des images

Introduction

Ce projet a pour but de réaliser une application web proposant 3 fonctionnalités :

- Déterminer le niveau de confiance en l'authenticité d'une photo
- Donner la date et l'heure de prise de la photo
- La localisation estimée et sa précision (par reconnaissance via des du machine-learning)

Ces fonctionnalités seront programmées en python puis implémentées dans une application web via node.js.

Ce rapport détaille comment ces 3 fonctionnalités ont été conçues.

Niveau de confiance

Le script **score_authenticite.py** a pour objectif de fournir un score d'authenticité pour une image donnée, exprimé en pourcentage. Ce score est calculé sur la base de trois analyses principales :

- **Vérification des métadonnées EXIF**
  Le script examine les métadonnées EXIF de l'image pour détecter des signes de modification ou de falsification. Il identifie notamment si un logiciel comme Photoshop a été utilisé et analyse la cohérence de la date de prise de vue. Une anomalie dans les métadonnées, comme une date future ou l'absence de ces données, peut diminuer la confiance dans l'authenticité de l'image.
- **Analyse des artefacts de compression JPEG**
  Cette analyse repose sur l'identification d'artefacts de compression, souvent présents dans des images modifiées ou reconstruites. Le script applique un flou gaussien pour détecter les bords artificiels causés par une compression excessive. Si des pics significatifs apparaissent dans l'histogramme des différences, cela peut indiquer une altération.

- **Examen de l'histogramme des couleurs**
  L'histogramme des couleurs est analysé pour détecter des anomalies dans la répartition des canaux rouge, vert et bleu. Une faible variation des couleurs peut suggérer une manipulation numérique, comme une superposition ou un collage.

Calcul du score

Le score d'authenticité est une moyenne pondérée des résultats obtenus pour chaque critère. Chaque analyse contribue équitablement au calcul final, avec un score maximal de 100 % indiquant une image vraisemblablement authentique.

Points forts et limites

- **Points forts** : Le script s'appuie sur des techniques variées, combinant la vérification des métadonnées et des analyses visuelles, pour fournir une évaluation globale de l'authenticité.
- **Limites** : Les résultats peuvent être influencés par des cas particuliers, comme des images compressées de manière excessive ou des absences de métadonnées légitimes.

En conclusion, cette méthode fournit une estimation fiable du niveau de confiance en l'authenticité d'une image, tout en restant perfectible, notamment dans le traitement des images dépourvues de métadonnées ou présentant des caractéristiques atypiques.

Date et Heure

Cette partie est la plus simple, pour récupérer ces informations il suffit d'extraire les données EXIF de l'image. Dans le cas où l'image ne possède pas de données EXIF, la date et heure ne peuvent pas être extraites.
Pour extraire les données EXIF on utilise deux bibliothèques python :

- exifread : pour extraire les données EXIF
- PIL : pour ouvrir et traiter l'image

Si l'image possède des données EXIF, le script récupère les données des tags EXIF qui nous intéressent, à savoir EXIF DateTimeOriginal et GPS Latitude / GPS Longitude, et les renvoient en sortie. L'analyse de la Date et Heure est inclue dans le script d'authenticité.

Localisation estimée

La majeure partie de cette application est la fonction de localisation via reconnaissance par Machine-Learning.

Cette fonctionnalité comporte 2 parties principales :

- La création d'une base de données d'extraction de caractéristiques d'images
- La comparaison entre les caractéristiques d'une image avec la base de données

Extraction des features

Le script concerné est **preprocess_flickr.py.**
Le but de cette partie est d'extraire des features (caractéristiques) d'un grand nombre d'images en associant ces features aux données de géolocalisation de chaque image.
Pour cela nous avons besoin en premier lieu d'une grande quantité d'images géolocalisées.
Nous avons choisi d'utiliser un dataset d'images géolocalisées extraites de Flickr :
https://www.kaggle.com/datasets/habedi/large-dataset-of-geotagged-images
Chaque image de ce dataset est sauvegardé dans un fichier MessagePack au format suivant :

```
{'image': b'\xff\xd8\xff\xe0...
\x05\x87\xef\x1e\x94o\xf6\xa6QG\xb4\x90Xv\xfa7\xd3h\',
'id': '13/20/8010869266.jpg', 'latitude': 29.426458,
'longitude': -98.490723}
```

L'étape suivante est d'extraire les features de ces images pour les sauvegarder dans notre BDD. Pour cela nous utilisons deux modèles de vision par ordinateur utilisant des réseaux de neurones :

- DELF (Deep Local Features) : permet d'extraire les caractéristiques locales des images
- ResNet50 (ResidualNetworks) : permet d'extraire les caractéristiques globales des images

L'utilisations conjointes des deux modèles permet une meilleure précision lors de la comparaison d'une image à la BDD.
Pour utiliser DELF en python on utilise la bibliothèque TensorFlow, pour ResNet on utilise PyTorch.

Le script commence par décoder les images depuis le fichier MessagePack et convertit et normalise chaque image pour répondre aux exigences de chacun des modèles.
Les modèles extraient ensuite les features de chaque image.
Enfin les features et les metadata (latitude, longitude, id) sont sauvegardées dans la BDD dans un fichier PKL.

Pour avoir une reconnaissance efficace il faut pouvoir extraire les données d'un grand nombre d'images. Nous avons choisi d'extraire les features d'un seul "fragment" de notre dataset comprenant 30 000 images. Le traitement aura pris environ 4h.

Comparaison des features

Maintenant que nous avons une BDD de features, il nous faut un deuxième script pour comparer une image en entrée aux features de la BDD. Le script **image_process_node.py** commence par extraire les features de l'image en utilisant les mêmes modèles et les mêmes paramètres que lors de la création de la BDD.
Il calcule ensuite la distance entre les vecteurs de features de l'image aux vecteurs de la BDD et on récupère les metadatas de l'image ayant la distance la plus proche.

Commentaires

Cette méthode marche bien pour les images suffisamment "unique". Si une image est "banale" c'est-à-dire que ses caractéristiques peuvent être retrouvées dans plein de paysage (exemple : une plaine) elle pourra être identifiée à de nombreuses images de la BDD et donc la position extraite ne sera pas forcément la bonne. Pour les images uniques (exemple : un monument reconnaissable comme la tour eiffel) l'identification sera facile et la position extraite sera donc précise.
Pour contrer cela, la solution la plus accessible est d'augmenter la taille de la BDD mais cela impliquerait un temps de création plus élevé (dans notre cas 4h pour 30 000 images). Pour réduire ce temps, il est possible d'utiliser une carte graphique compatible avec CUDA et cuDNN.

Pour la comparaison entre l'image en entrée et la BDD nous avons essayé de faire une moyenne pondérée des images similaires mais les coordonnées issues de ce traitement donnaient trop souvent des points très éloignés des coordonnées de l'image en entrée. Cela est dû au problème mentionné précédemment d'images trop "banales" pouvant être retrouvées à beaucoup d'endroits autour du globe. Nous avons décidé de ne pas faire de moyenne pour avoir des estimations plus correctes pour le plus d'images.