ÉCOLE NATIONALE SUPÉRIEURE D'INGÉNIEURS DE CAEN

Analyse Forensique de fichiers
Géolocalisation et description en langage naturel

Auteur :
Pierre HUSSON
Oscar MATHEY
Antoine JOURDAN
David PLESSIS

Encadrants :
Christophe ROSENBERGER
Emmanuel GIGUET
Référent :
Loick LHOTE

4 Avril 2022

Résumé

L'objectif principale de ce projet est d'évaluer les capacités des projets open source capables de géolocaliser les lieux de capture de photographies à partir du simple traitement du contenu de ces dernières, c'est-à-dire sans utiliser les métadonnées, et de ceux capables de description en langage naturel de scènes photographiées.

Il s'agit donc d'effectuer un état de l'art des solutions existantes pour ces deux types de projet, et d'évaluer de manière quantitative les performances des outils les plus prometteurs en vue d'une possible utilisation dans le cadre de l'analyse forensique de fichiers. Ainsi, après des recherches et la lecture de plusieurs papiers d'équipes compétentes dans ces deux domaines, nous avons sélectionné le modèle LocationNet [2] pour la géolocalisation des images. Pour ce qui est de la description en langage naturel, nous détaillerons dans la suite de ce rapport nos mesures réalisées sur le projet de HughKu [1] qui nous permet également de détecter la présence d'humain sur des photographies.

Table des matières

Chapitre 1

Géolocalisation

Pour évaluer le modèle d'estimation de géolocalisation, nous avons eu l'approche suivante. Il est peu probable qu'il n'y ait qu'une image à géolocaliser. Si tel était le cas, une évaluation des performances avec une approche quantitative image par image est disponible dans le papier original. Dans un contexte de forensique, nous avons choisi d'évaluer les performances moyennées sur plusieurs dizaines d'images, afin d'en extrapoler une localisation plus précise. Pour ce faire ces images doivent être représentatives de ce que l'on pourrait trouver en recherchant des photos de vacances ou de festivités sur les disques à fouiller. Il faut donc créer des jeux de données constitués de photographies prises par des particuliers lors de voyages ou d'évènements festifs. On quantifira par la suite les capacités de ce modèle à trouver une ville à travers le Monde, l'Europe et enfin la France.

1.1 Création des datasets

Pour la création des datasets sur lesquels nous allons évaluer le modèle LocationNet, nous nous sommes tournés vers Flickr. En effet, ce réseau social orienté photographie nous permet via son API de facilement faire des requêtes par dizaines. Ainsi, il nous est possible de demander à cette API de nous envoyer toutes les images contenant des tags comme le nom de la ville et celui du pays que l'on veut. Ces images ont l'avantages d'avoir été précisément prisent dans la ville du pays que l'on a choisi, mais le requête est assez vague pour que l'on n'ait pas uniquement des photos de monuments ou de lieu facilement reconnaissable qui seraient trop facile à traiter pour notre modèle. On obtient ainsi aussi bien des photos de jour que de nuit, d'endroits reconnaissables que de petites ruelles, avec beaucoup de gens dessus ou de petites échoppes avec 1 commerçant. Le premier dataset est donc constitué avec les 30 plus grandes villes du monde de la manière suivante :

```
    ⋮

cities = ['Shanghai', 'Delhi', 'Tokyo', 'Sao Paulo', 'Mexico', 'Dhaka', 'Cairo',
'Beijing', 'Bombay','Osaka','New York', 'Karachi', 'Chongqing', 'Istanbul',
'Buenos Aires', 'Kolkata', 'Kinshasa', 'Lagos', 'Manila','Tianjin', 'Canton',
'Rio de Janeiro', 'Lahore', 'Bangalore', 'Moscow', 'Shenzen', 'Bogota',
'Jakarta','Lima','Paris']

countries = [...]

for i, city in enumerate(cities):
    os.mkdir('./resources/images/citiesSet/' + city)
    photos = flickr.walk(extras='url_c',
                         per_page=100,
                         content_type=1, #Type : Photograhpie
                         geo_context='2', #Photo en extérieur
                         min_date_upload=1070030468.0,
                         tag_mode='all',
                         tags=city + ', city, ' + countries[i],
                         text=city + ', city, ' + countries[i])

    ⋮
```

On dispose donc d'un dataset de 3000 images avec 100 images par ville. Pour la suite on recommencera ce même procédé avec les 15 plus grandes villes européennes ainsi que les 10 plus grandes villes françaises. Il suffit de changer les tableaux *cities* et *countries*.

1.2 Méthodes d'estimation

Pour tous les datasets, nous avons itéré à travers toutes les images pour produire, grâce au modèle, des fichiers de donnéees contenant pour chaque image les coordonnées GPS estimées (latitude et longitude format EPSG :4326) ainsi que l'indice de confiance dans la prédiction du modèle. On définit en fonction du nombre d'image une confiance cumulée par coordonnées GPS : chaque fois que une image est estimée au même endroit qu'une autre, on rajoute l'indice de confiance à la valeur du point représentant ces coordonnées. On retient ainsi que le point de confiance cumulée la plus élevée. Ce sont les coordonnées de ce point que l'on considère comme les coordonnées les plus probables du lieu de capture des photographies traitées.

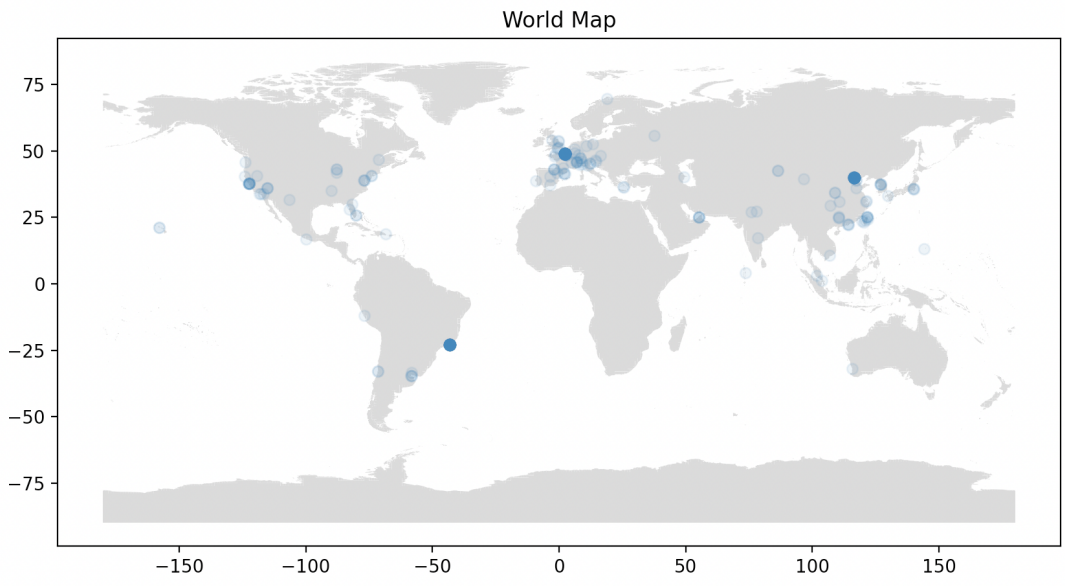


FIGURE 1.1 – Visualisation des estimations du modèle pour les datasets Paris, Beijing et Rio de Janeiro

Sur la figure ci-dessus on constate qu'il y a beaucoup de bruit, par le calcul on peut obtenir les coordonnées des points les plus marqués. On rajoute en rouge le centre de la ville de Paris et on zoom dessus. On obtient la figure 1.2, on remarque la précision de l'estimation.

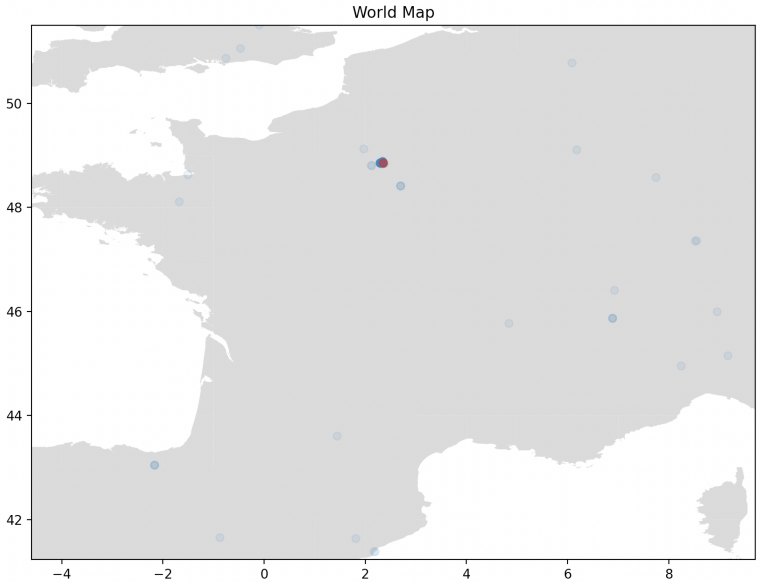


FIGURE 1.2 – Zoom sur la France

1.3 Résultats et interprétations

On réalise un calcul de distance entre les coordonnées effectives d'une ville et l'estimation qu'en fait notre algorithme. On obtient la figure 1.3 pour le dataset concernant les 30 plus grandes villes du monde.

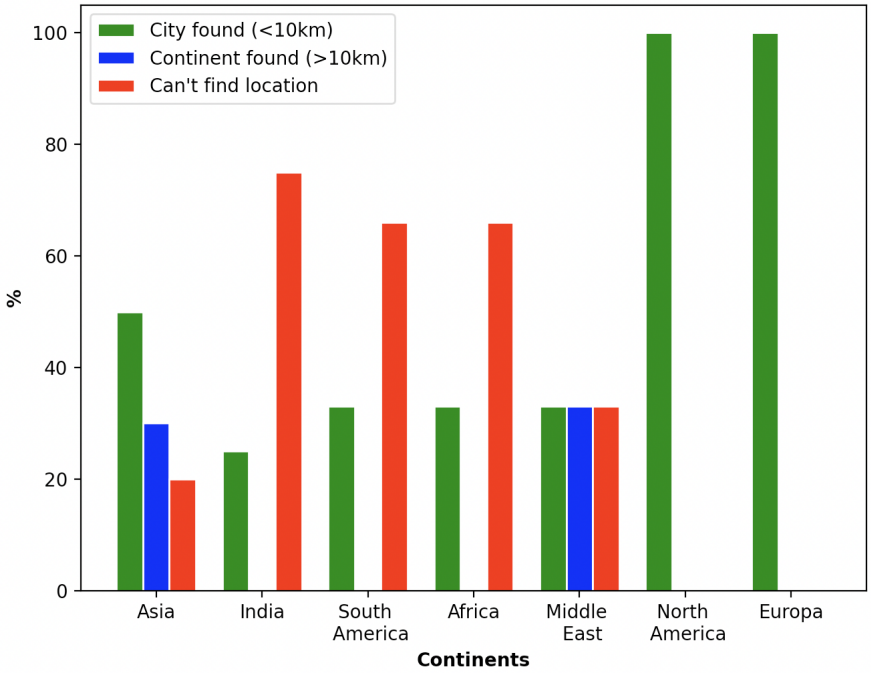


FIGURE 1.3 – Résultats obtenues pour les 30 plus grandes villes du monde en fonction de leur continent

On remarque plusieurs choses. Déjà les villes d'Asie ont tendance à être confondues par le modèle, cela se reflète notamment dans le pourcentage de ville dont la position est estimée à plus de 10km près. Par exemple, Shanghai est reconnu en tant que Beijing et Tokyo. Ce n'est certes pas loin mais incorrect et une telle imprécision avec autant de données traités est révélatrice d'une imprécision du modèle au niveau des villes asiatiques. Pour ce qui est des autres continents, le modèle estime très mal les coordonnées des villes d'Inde, d'Amérique du sud ou encore d'Afrique. On remarque que les coordonnées GPS des villes d'Europe et d'Amérique du Nord sont estimées avec une réussite de 100% : c'est normal celà ne concerne que 3 villes : New-York, Paris et Moscou. Il faut donc venir completer ces résultats avec des résultats plus poussés. C'est pourquoi nous avons choisi de nous intéresser plus en détail au continent Européen, celui le plus susceptible de nous intéresser en cas d'utilisation de ce modèle dans le cadre d'enquête forensique. En relançant tous nos scripts sur les 15 plus grandes villes européennes, on obtient ces résultats :

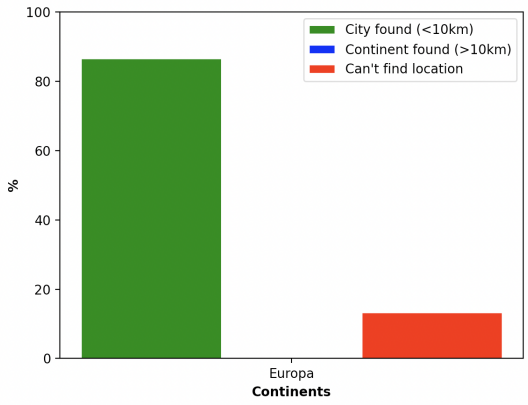


FIGURE 1.4 – Résultats obtenues pour les 15 plus grandes villes d'Europe

Le modèle retrouve la plupart des grandes villes européennes à quelques kilomètres près. De plus, il trouve parfois les coordonnées exactes du centre de la ville (précision à environ 1 km lors des mesures GPS).

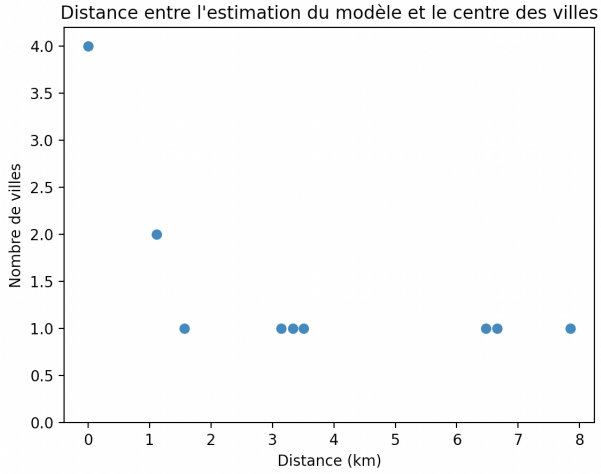


FIGURE 1.5 – Nombre de villes européennes dont les coordonnées sont estimées à moins de 10km

On répète ce procédé une dernière fois pour les 10 plus grandes villes de France.

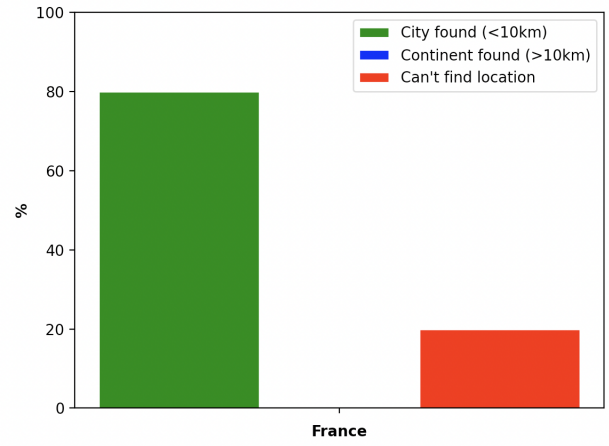


FIGURE 1.6 – Résultats obtenues pour les 10 plus grandes villes de France

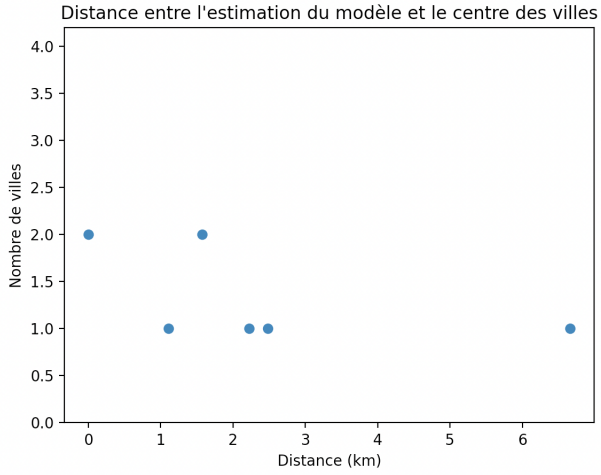


FIGURE 1.7 – Nombre de villes françaises dont les coordonnées sont estimées à moins de 10km

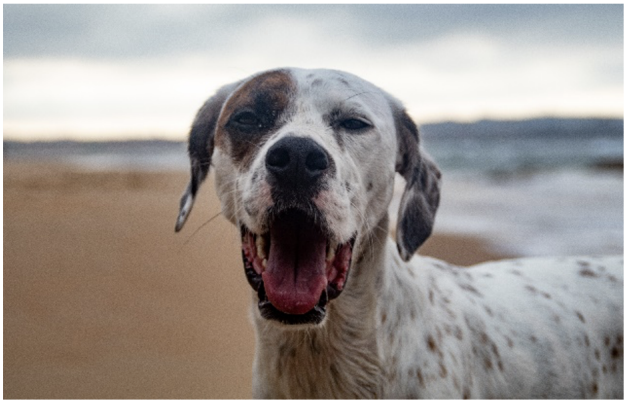
On remarque ainsi que la meilleur performance sur les villes européennes que l'on avait subodorée à l'échelle

du monde se confirme bien lorsque l'on traite des photographies exclusivement prises en Europe. De plus, on remarque que ces performances sont semblables en France avec 80% de réussite. En diminuant le nombre d'images par ville de moitié, on perd environ 15% de réussite sur l'estimation des géolocalisations des villes françaises. En réduisant encore de moitié, on arrive à un taux de réussite de l'ordre des 55-60% avec 25 images par villes. Dans l'ensemble l'outil est assez performant mais ses estimations dépendent grandement du type de décor présent sur les photographies à traiter ainsi que, comme on vient de le voir, de la zone géographique recherchée.

Chapitre 2

Description en langage naturel

Pour la description en langage naturel nous avons étudié le projet de HughKu basé sur le projet Im2txt. Il s'agit d'une IA prenant en entrée une image et renvoyant 3 descriptions possible de l'image chacune accompagnée de l'indice de confiance. Pour l'intégralité des tests, l'IA se base sur le modèle TensorFlow model.ckpt-2000000 issue d'un entraînement de deux millions d'itérations, ainsi que sur un vocabulaire de 11519 mots. Voici quelques exemples de ce que peut faire cette IA du meilleur au moins bien :



0) a brown and white dog is looking at the camera. (p=0.000155)

1) a brown and white dog sitting on top of a beach. (p=0.000123)

2) a brown and white dog is looking at the camera (p=0.000109)

0) a tall clock tower with a sky background (p=0.002517)

1) a tall clock tower with a sky in the background (p=0.001261)

2) a tall clock tower with trees in the background (p=0.000807)

0) a woman sitting on a bench looking at the ocean. (p=0.000104)

1) a woman sitting on a bench looking at a cell phone. (p=0.000086)

2) a woman sitting on a bench looking at the water. (p=0.000061)

0) a motorcycle parked on the side of a road . (p=0.002468)

1) a motorcycle parked on the side of the road . (p=0.001442)

2) a motorcycle parked on the side of the road (p=0.000504)

FIGURE 2.1 – Exemples d'output du modèle

2.1 Création des datasets

Pour mesurer les performances du modèle il a d'abord fallu créer nos datasets. Pour commencer nous avons créé un dataset de 100 images choisies aléatoirement (general dataset) afin de mesurer les performances globales du modèle. Au vu du contexte futur de l'utilisation de cet outil nous avons trouvé important de mesurer la capacité du modèle à détecter les humains sur les photos de ce fait nous avons créé 2 autres datasets. Le premier (human dataset), de 100 images également, contenant uniquement des humains facilement reconnaissables, avec au moins toute la partie haute du corps sur la photo. Le second dataset (complex human dataset) contenait 40 images d'humain plus difficilement reconnaissable, seul un membre du corps était visible sur la photo ou des photos floues. Cependant toutes ces images étaient de très haute résolution (au-delà de 6000*4000) ce qui certes facilitait le travail de l'IA mais n'était pas représentatif des images qui seront utilisées dans le futur. Ainsi nous avons créé 4 nouveaux datasets 2 dérivés du genral dataset (medium quality genral dataset et low quality genral dataset) et 2 du human dataset (medium quality human dataset et low quality human dataset) comprenant les mêmes images mais redimensionnées afin d'en réduire la qualité (20% et 5% de la taille originale). Une fois les datasets crées il a fallu tagger toutes les images avec une description de la scène afin d'avoir une vérité terrain à comparer avec les résultats de l'IA.

2.2 Métriques

Nous avons ensuite sélectionné des métriques afin de mesurer et évaluer les propositions de l'IA. Les 3 métriques retenues sont la distance de Levenshtein, la distance de Jaro-Winkler et la méthode cosine similarity. La distance de Levenshtein retourne le nombre de permutation, modification et d'ajout ou suppression de caractère qu'il faut pour passer d'une chaine de caractère à l'autre. La distance de Jaro-Winkler mesure la similarité entre 2 chaines de caractère en retournant une valeur entre 0 et 1 (0 étant l'absence de similarité et 1 l'égalité des 2 chaines). La distance de Jaro est définie par :

$$d_j = \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) \tag{2.1}$$

où :
$|s_i|$ est la longueur de la chaîne de caractères s_i ;
m est le nombre de "caractères correspondants"
t est le nombre de "transpositions'

Deux caractères identiques de S1 et S2 sont considérés comme correspondants si leur éloignement (la différence entre leurs positions dans leurs chaînes respectives) ne dépasse pas :

$$\left\lfloor \frac{\max(|s_1|,|s_2|)}{2} \right\rfloor - 1$$

Winkler va introduire ensuite un coefficient p (il propose comme valeur 0.1) favorisant les chaînes commencent par un préfixe de taille l<5 et proposera comme formule pour la distance de Jaro-Winkler :

$$d_w = d_j + (\ell p(1 - d_j)) \tag{2.2}$$

Enfin la méthode cosine similarity mesure la similarité entre 2 phrases. Il s'agira par la suite de la métrique la plus intéressante car elle ne se cantonne pas à la comparaison caractère par caractère mais bien à comparer les mots entre 2 phrases. Pour cela on créé une matrice composée de 0 et de 1 de taille k*n avec k le nombre de phrases et n le nombre de mots uniques total. Une fois fait on peut facilement déterminer la similitude entre 2 phrases en comparant le nombre de mot en commun. Nous avons également utilisé une 4e implémentée manuellement (les autres étant déjà implémentées dans des librairies) permettant de vérifier la fiabilité du modèle à détecter les humains. Pour cela on vérifie simplement que quand la vérité terrain indique la présence d'un humain l'IA le voit également indépendamment du contexte donné, et à l'inverse que si le modèle détecte une personne on vérifie que la vérité terrain en indiquait la présence. Pour cela, on utilise une fonction qui vérifie parmi une liste de mots correspondant à un humain (person, man, woman, boy, girl, ...) que la phrase passée en argument contient bien un humain.

2.3 Mesures et interprétations

Une fois nos datasets et métriques déterminés nous pouvons commencer les phases de tests. Pour commencer on s'intéresse aux 3 premières métriques sur tous les datasets sauf le complex human dataset, ce dernier servant à vérifier la capacité de l'IA à repérer des humains et non sa capacité à décrire précisément une scène.

Métriques / Datasets	Distance de Levenshtein	Distance de Jaro-Winkler	Cosine Similarity
General	18.650000	0.822400	0.685931
Medium Quality General	22.010000	0.803400	0.667122
Low Quality General	25.390000	0.766600	0.619923
Human	22.810000	0.799500	0.630177
Medium Quality Human	22.810000	0.795500	0.661135
Low Quality Human	24.310000	0.775200	0.574442

FIGURE 2.2 – Moyennes des métriques sur chaque dataset

On constate tout d'abord que pour les deux datasets que la précision des prédictions de l'IA diminue avec la qualité, et cela pour les 3 métriques. En regardant de plus près les deux mesures de similarité, on constate que Jaro-Winkler annonce de meilleur résultat que cosine similarity (CSim) cela s'explique par le fait que Jaro-Winkler compare les caractères alors que CSim va faire une comparaison par rapport aux mots. Ainsi des mots différents mais avec des lettresen commun seront considérés comme similaire par Jaro-Winkler mais différents par CSim. En prenant ça en compte, et le fait que Levenshtein se comporte de façon similaire à Jaro-Winkler, la mesure la plus intéressante pour qualifier la précision du modèle est CSim. On a donc, une précision comprise entre 57% dans le pire des cas et 68% dans le meilleur. On constate également que le modèle est plus précis en moyenne pour décrire des photos générales que des photos d'humains probablement lié au fait que le modèle utilisé ait été entrainé sur des datasets contenant une grande variété d'image et non un dataset composé exclusivement de photo d'humain. Comme dit précédemment il peut être intéressant de connaitre la capacité du modèle à détecter la présence d'humain, indépendamment du contexte, sur une photo. Pour cela on se sert des dataset human dataset (et ses dérivés) et complex human dataset.

Datasets / Métrique	Human	Medium Quality Human	Low Quality Human	Complex Human
Human detection	96.666667 %	96.333333 %	95.666667 %	89.166667 %

FIGURE 2.3 – Human detection

Encore une fois, on constate que la diminution de la qualité d'image apporte une diminution de la précision, cependant la capacité de détection d'humain reste excellente avec une précision de plus de 95% pour les human dataset. Pour le complex human, on a une précision inférieure aux 3 autres avec 89% mais cela reste plutôt précis étant donné que l'évaluation se fait sur des images on l'on ne voit parfois qu'une main ou même des photos entièrement floues. Il faut cependant prendre en compte qu'il s'agit ici de mesurer la capacité à détecter un humain en général et non pas de quel genre d'humain il s'agit où du nombre. En effet dans le cas où, par exemple, l'IA détecte un homme au lieu d'une femme ou une seule personne au lieu de plusieurs, le résultat est quand même compté comme positif.

Enfin nous avons déterminer les écarts types de chaque mesure sur tous les datasets à l'exception du complex human.

Ecart type / Datasets	Distance de Levenshtein	Distance de Jaro-Winkler	Cosine Similarity
General	9.873575846672773	0.10761152354650501	0.25936096677940496
Medium Quality General	9.592879303594582	0.10284848918868507	0.2514343560423392
Low Quality General	9.718976906584698	0.10432154760219786	0.25573271949104437
Human	9.688136179827822	0.10403984780148204	0.2558242781755911
Medium Quality Human	9.35055027938696	0.10155276876456873	0.24896431471722544
Low Quality Human	9.718976906584698	0.10432154760219786	0.25573271949104437

FIGURE 2.4 – Ecart types sur chaque dataset

On constate ici des écarts type plutôt importants quel que soient le dataset et la métrique. En effet l'écart type de Levenshtein est d'environ 10 pour une moyenne des valeurs de 20 et CSim a un écart type de 0,25 pour des moyennes entre 0,6 et 0,7 dans les deux l'écart type représente près de la moitié de la moyenne indiquant une très grande disparité des valeurs. Concrètement dans notre cas cela signifie que si même si le modèle a une précision moyenne autour de 65% il arrivera régulièrement que les prédictions soient bien plus précises, ce qui ne dérange pas, mais également qu'elles peuvent être bien moins précise il s'agit donc d'être vigilent avec les résultats fournis par l'IA. Finalement ce modèle permet en moyenne une bonne description d'images et parvient à détecter de manière très fiable la présence d'humain sur une photo même si celle-ci est de faible qualité ou que l'humain est difficile à détecter. Il faut cependant prendre du recul sur les résultats quand on sait que l'écart type des 3 métriques utilisées est très important indiquant une grande disparité des résultats possibles.

Conclusion

Les multiples mesures réalisées sur les performances des différents projets que l'on vient d'évalués ont permis de confirmer les intuitions que nous avions lors de la phase de recherche. Ces projets sont plutôt performants et adaptés à une utilisation en forensique. Ils font donc l'objet d'une implémentation au format GDIP. Ils pouront par la suite faire partie des filtres que cette dernière propose.

Bibliographie

[1] HughKu. Im2txt. https://github.com/HughKu/Im2txt, 2018. [Online ; accessed 3-april-2022].

[2] Kevin Li Jaeyoung Choi. Estimating the location of images using apache mxnet and multimedia commons dataset on aws ec2. https://aws.amazon.com/fr/blogs/machine-learning/estimating-the-location-of-images-using-mxnet-and-multimedia-commons-dataset-on-aws-ec2, 2017. [Online ; accessed 3-april-2022].