Documentation du POC NPImageNearest

Cyril Vincent

Introduction

Utilisation des techniques d'IA pour rechercher les images d'un produit les plus proches des images d'autres produits dans NextPage.
Utilisation d'un modèle de Deep Learning CNN (Convolutional Neural Network) préentrainé de Google nommé FV (Feature Vector) et d'un modèle de Machine Learning nommé ImageHash inventé par Johannes Buchner et adapté par mes soins, le tout avec le framework Google TensorFlow 2.3.

NPImageNearest

Pour accéder à la démo, voici l'URL : https://www.cyrilvincent.com/np/html/iindex.html

Cette page liste l'intégralité des images du CHUV, avec l'id de l'image et la miniature de l'image.
Quand on clic sur l'image la page des plus proches images s'ouvre :

NP Image Nearest

Image Index

Product Index

Search Nearests images of 266846 07640160162588_PIECE_01.JPG 

Found 4 image(s)

Image: 266849 07640160162595_PIECE_01.JPG at 84%  {'dah': 0.812, 'ddh': 0.703, 'dfv': 0.827, 'dsize': 27956, 'dn': 0.913}

Image: 266875 07640178432178_PIECE_01.JPG at 84%  {'dah': 0.859, 'ddh': 0.609, 'dfv': 0.838, 'dsize': 7923, 'dn': 0.739}

NPImageNearest trouve les 10 plus proches images avec un score > 82%

Signification des scores :

- >99% : c'est la même image
- >95% : c'est la même image mais légèrement retouchée
- >90% : c'est un produit très similaire
- >82% : image assez similaire
- <82% : non significatif

Essayons avec la seringue 266723

NP Image Nearest

Image Index

Product Index

Search Nearests images of 266723 04046963435738_PIECE_01.JPG 

Found 10 image(s)

Image: 266708 04022495251114_PIECE_01.JPG at 90%  {'dah': 0.875, 'ddh': 0.781, 'dfv': 0.942, 'dsize': 47, 'dn': 0.652}

Image: 266710 04022495251862_PIECE_01.JPG at 88%  {'dah': 0.781, 'ddh': 0.797, 'dfv': 0.942, 'dsize': 1699, 'dn': 0.609}

Nous remarquons que l'image 266708 est très proche mais il ne s'agit pas du même produit

J’ai essayé avec des images dupliquées et il les trouve avec un score de 100%
J’ai essayé avec des images recompressées et il les trouve avec un score de 99%
J’ai essayé avec des images retouchées et il les trouve avec un score de 90% à 95%

Dans le jeu de tests CHUV il n’y aucune image strictement identique, le tout sur 307 images

Signification de : {'dah': 0.875, 'ddh': 0.781, 'dfv': 0.942, 'dsize': 47, 'dn': 0.652} :

- Dah est le score sur 1 du modèle AverageHash de ImageHash, il s’agit d’une hyper compression de l’image en 8x8 puis d’une mesure de distance entre les 2 images compressée. Ce modèle est très rapide mais donne des faux positifs
- Ddh est le score sur 1 du modèle DifferenceHash de ImageHash qui fonction comme AverageHash mais sur les gradients de luminosités. Ce modèle est rapide et détecte très bien les images retouchées
- Dfv est le score sur 1 du modèle Feature Vector qui est un réseau Deep Learning CNN préentrainée sur ImageNet qui va hacher l’image sur 1792 bits. Ce modèle va trouver une même forme sur des images pourtant différentes. Ce modèle est assez lent en apprentissage mais rapide en prédiction
- Dsize est la différence entre la signature de l’image, ce modèle détecte pour un coût dérisoire 2 images strictement identiques
- Dn n’est pas utilisé
- Le score est calculé par une moyenne pondérée des différents scores après un léger apprentissage.

Conclusion

L’outil est fiable à 100% pour retrouver des images strictement identiques
L’outil est fiable à 95% pour retrouvées des images recompressées ou retouchées
L’outil est fiable à 95% pour retrouver des produits identiques sur un ensemble de produits assez discriminants les uns aux autres
L’outil est fiable à 85% pour retrouver des produits identiques sur un ensemble de produits où les produits se ressemblent comme pour CHUV

Efficacité

J’ai utilisé une technique d’indexation NoSql pour améliorer les performances.
J’ai généré des fichiers fictifs avec 10000 et 100000 images pour tester les temps de traitement.
Voici les performances en prédictions des produits proches sur mon PC perso (i7 récent + GPU + 16Go de RAM) :

- Pour 305 images : <0.1s
- Pour 1000 images : 0.1s
- Pour 100000 images : 5s

Le temps d’apprentissage est le suivant :

- Pour 305 images : 9.7s
- Pour 1000 images : 32s
- Pour 100000 images : <1h

Voici les performances de l'apprentissage pour comparer N images parmi M images en désactivant le modèle FV pour accélérer l'apprentissage et en baissant un peu la qualité de prédiction :

- Pour 305x305 images : 7s
- Pour 1000x1000 images : 74s
- Pour 5000x5000 images : <1h

Rapprochement par produit

Le rapprochement par produit en fonction de leurs images a été implémenté, ainsi si un produit possède plusieurs images ou plusieurs références contenant des images, le rapprochement par produit s'effectue si au moins une image du produit match l'image d'un autre produit.
Dans le cas particulier du CHUV, chaque produit n'a jamais plus d'une image.

Détail d'utilisation du modèle CNN FV et ImageHash

- Clonage du modèle CNN FV entrainé depuis MobileNet
- Adaptation des poids par transfert learning
- Création de l'indexation des caractéristiques en JSON et Pickle
- Hachage des caractéristiques par prédiction du modèle avec création d'un vecteur de 172 flottants 64 bits et 2x64 flottants 64 bits pour AH et DH
- Stockage des hachages en index
- Recherche des plus proches voisins par application un calcul de distance avec similarité cosine
- Modification et normalisation des poids

Industrialisation et intégration dans NextPage

Comme le précédent POC, l'intégration dans le Backend NP nécessite 2 jours de ma part (hors Angular)

Opportunités

Il est possible de poursuivre la démarche du POC pour améliorer NextPage :

- Catégorisation : Trouver automatiquement la famille d'une image (2j)
- Prédiction : Trouver un produit similaire (1j)
- Barcode : Lire automatiquement le code barre ou le QRCode sur l'image (2j)

    - 

    - CODE128: 0107323190073177172205281019F011
- OCR : Lire automatiquement le texte sur l'image (3j)
    - 

    - Donne : Biogel PI Indicator Underglove …

Ces fonctionnalités peuvent se faire en POC en 1 à 3 jours chacune au lieu de 4.

Références

TensorFlow : https://www.tensorflow.org/?hl=fr
Feature Vector : https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4
ImageHash : https://pypi.org/project/ImageHash/