Rapport Projet Deep Learning

Université Paris Dauphine

Olivier Berthier

Table of contents

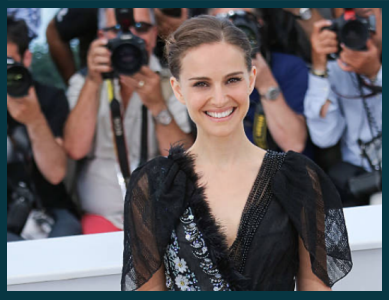
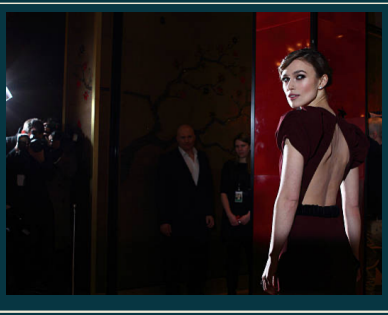
1 Who is she ?

Ce projet consiste à reconnaître les visages des actrices Nathalie Portman et Keira Knightley parmi des photos téléchargées sur internet.

Les données consistent en :

- 1 jeu de photos d'entraînement ("*train*") selon 3 catégories : *natahlie*, *kaira*, et *others* (143 photos dans chaque catégorie)
- 1 jeu de photos de validation ("*val*") selon les mêmes catégories (56 photos dans chaque catégorie)
- 1 jeu de photos de test ("*test*") selon les mêmes catégories (56 photos dans chaque catégorie). Notons que les labels sont présents pour le jeu de test.

  Exemples de photos:

Nathalie	Keira	Others
		

Toutes les photos sont au format PNG, définition: 530 x 400.

Le cadrage, la position de l'actrice, la distance au sujet et la qualité de la photo originale sont variés.

D'autres visages peuvent également être présents sur la photo.

2 Stratégie employée

Nous avons utilisé Kaggle pour profiter des ressources GPU nécessaires aux modèles développés.

Nous détaillerons plusieurs modèles dans ce rapport (d'autres modèles ont été essayés mais ne sont pas présentés ici pour éviter la redondance):

- Modèle fait à la main
- Modèle ConvNext
- Modèle ConvNext sur visages préalablement isolés

Tous nos modèles travaillent en couleurs (RGB), des essais sur un seul canal (dégradé de gris) n'ayant pas amené de meilleures performances. N'ayant pas insisté dans cette direction, il est difficile d'exclure à priori de travailler sur un seul canal -au moins pour le gain de temps computationnel.

Nous utilisons AdamW comme optimiseur pour tous les modèles (avec ou sans *scheduler* selon les essais).

La précision est ajustée à *16-mixed* afin de gagner du temps de calcul (pas de baisse de performance observée).

3 Modèle "fait à la main" *CNN_base_5_blocs*

Architecture type VGG-like simple en cinq blocs selon:

[Input (3, 530, 400)

↓

[Bloc1: Conv2d*2 + BN + ReLU + MaxPool] → (32, 265, 200)

↓

[Bloc2: Conv2d*2 + BN + ReLU + MaxPool] → (64, 132, 100)

↓

[Bloc3: Conv2d*2 + BN + ReLU + MaxPool] → (128, 66, 50)

↓

[Bloc4: Conv2d*2 + BN + ReLU + MaxPool] → (256, 33, 25)

↓

[Bloc5: Conv2d*2 + BN + ReLU + MaxPool] → (512, 16, 12)

↓

[Flatten] 98 304

↓

[FC] 98 304 → 1 024 → 512 → 3

Les images sont augmentées dans le *datamodule* pour ajouter de la diversité (démarche particulièrement utile ici en raison du relatif petit nombre de données).

Les dimensions originales ont été conservées (ce qui limite la taille du batch pour ne pas surcharger la mémoire des GPU); des essais en les réduisant en 224 x 224 ont néanmoins montré des résultats comparables.

Nous avons utilisé la fonction *Tuner* pour aider à définir le *learning rate*.

Nous obtenons de bons résultats en entraînement: facilement entre 95 et 100% selon le learning rate et les augmentations d'images retenues (en une trentaine d'époques sans *weigth decay*). Par contre, le modèle ne parvient pas à obtenir de bons résultats en val et test (difficilement au dessus de 50%).

Nos meilleures performances sont obtenues avec de faibles *learning rate*, en augmentant le *weigth decay* (régularisation L2) et en ajoutant du *drop out* sur la couche fc et sur le dernier bloc convolutionnel. La convergence est largement ralentie mais nous parvenons ainsi à diminuer *l'overfitting*.

Performances: Notre meilleur score est de 66% sur l'ensemble test après 200 époques.

4 Modèle ConvNext

Nous avons retenu le modèle ***convnext_tiny.in12k_ft_in1k_384*** pré-entraîné pour un bon compromis efficacité / coût computationnel (nous avons essayé la version large du ConvNext sans augmentation notable des performances mais avec un temps de calcul nettement supérieur).

Nous conservons les dimensions originales des images (des essais en 224 x 224 donnaient des résultats comparables mais notre meilleur score a bien été obtenu avec les dimensions originales).

Performances: Notre meilleur score en 20 époques est de 85% sur l'ensemble test.

5 Modèle ConvNext sur visages préalablement isolés

Notre objectif est ici de fournir au modèle les visages en gros plan uniquement. Nous éliminons l'arrière plan afin que le modèle devienne un meilleur spécialiste des visages.

La détection des visages dans chaque image s'appuie sur le modèle pré-entraîné **YOLOv8-Face,** disponible sur HuggingFace.
La fonction *detect_best_face_and_crop* permet d'extraire et sauvegarder pour chaque image le visage jugé le plus pertinent, selon un score combinant plusieurs critères.

Étapes de la fonction de détection des visages :

- **Détection des Visages :** Chaque image du dossier d'entrée est analysée par le modèle YOLOv8-Face, qui détecte un ou plusieurs visages et retourne leurs *bounding boxes* et scores de confiance.
- **Sélection du Meilleur Visage:** Lorsque plusieurs visages sont présents sur l'image, le script calcule pour chaque détection un **score global**, basé sur trois critères :
  - **Probabilité de détection** (score de confiance du modèle)
  - **Surface** occupée (plus la boîte est grande, plus le score est élevé)
  - **Proximité du centre de l'image** (les visages proches du centre sont favorisés)

  Les trois critères sont ajustés et combinés pour sélectionner automatiquement le visage le plus pertinent.

La fonction de détection des visages obtient de bons résultats. Seul le jeu *train* keira nécessite de rectifier deux erreurs (où des visages autres que celui de Keira Knightley ont été sélectionnés). Pour cela, nous utilisons la fonction de correction ad-hoc *detect_all_faces_and_select.*

Aucune erreur n'est constatée sur les jeux de validation et de test (le cahier des charges de l'exercice nous interdirait de corriger les erreurs sur l'ensemble de test).

Les images sont ensuite redimensionnées en 224 x 224 avec conservation du ratio; du *padding* est ajouté en périphérie pour ajuster aux nouvelles dimensions si nécessaire (fonction *resize_with_padding*).

Exemple de photo après *détection et redimensionnement:*

Sur ces images, nous appliquons le même modèle ConvNext que précédemment avec ses propres réglages d'optimiseur.

Notre meilleur score obtenu est de **96.4%** sur l'ensemble test.