Projet Dog Breed Prediction

Présenté par : Yasmine Nahal

Dans le cadre de : Majeure Deep Learning - YNOV Informatique

Objectif du projet

Elaborer un algorithme de **prédiction de la race d'un chien**

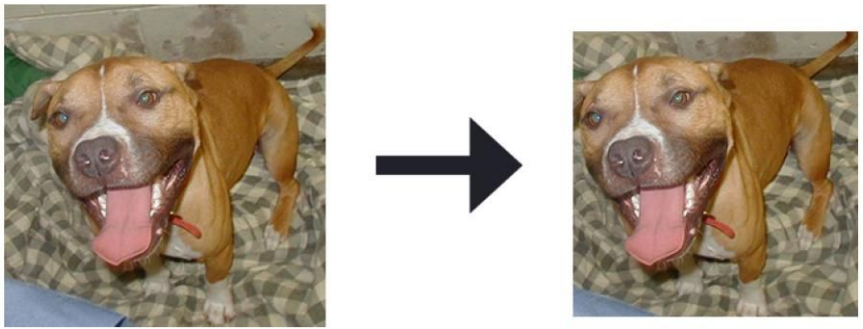
Ressources :

- Stanford Dogs Dataset (20850 images de chiens de 120 races différentes préalablement annotées)
- Tensorflow - Keras (APIs de Deep Learning)
- Calculs effectués sur Google Colaboratory (12.5 GB de RAM + accélérateur TPU)

http://vision.stanford.edu/aditya86/ImageNetDogs/main.html

Préparation des données : *pre-processing*

- Les dimensions de l'image, l'espace qu'occupe le chien dans l'image, la résolution de l'image… ce sont des paramètres à prendre en compte !
- **Méthodologie :**
  - **Cropping** des images selon les **annotations** associées à chaque image -> **focus sur le chien**
  - **Resizing** (224x224 pixels)



Préparation des jeux d'entraînement, validation et test

- Stanford a mis à disposition les images d'entraînement et de test dans les 2 fichiers suivant : ***train_list.mat*** et ***test_list.mat***
- **Méthodologie :**
  - Depuis le dossier des images rognées, récupération des images listées dans t***rain_list.mat*** et ***test_list.mat*** et les placer dans 2 nouveaux dossiers : **Train** et **Test** respectivement
  - Créer un jeu de validation depuis le jeu d'entraînement (10% validation)
  - **Résultat** :
    - 10800 images pour le jeu train
    - 8050 images pour le jeu test
    - 1200 pour le jeu de validation
    - (120 races dans chaque jeu)

Features engineering

- Convertir les images en pixels et encodage des classes à prédire (les races) pour permettre à la machine de comprendre les données et faciliter la modélisation

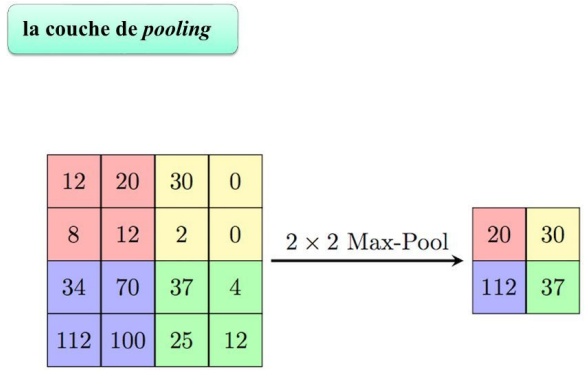
- **Méthodologie :**
  - Convertir toutes les images en tableaux de pixels de dimensions: 224x224x3
  - Encodage des 120 classes : associer à chaque classe un vecteur de 120 bits (où une seule valeur est égale à 1 et dont la position dans le vecteur diffère selon la classe)

Premiers entraînements (from scratch)

Méthodologie :

- **Modèle simple et intuitif**
  - constitué de 2 couches de convolution 2D séparés par une couche de Max Pooling 2D + dropout de 25%

- **Modèle un peu plus avancé**
  - 3 alternances de couches conv 2D et Max Pooling 2D + dropout augmenté à 40% et un Batch Normalization

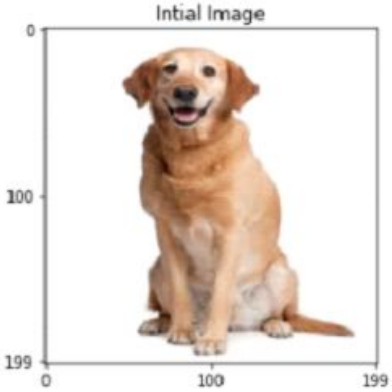


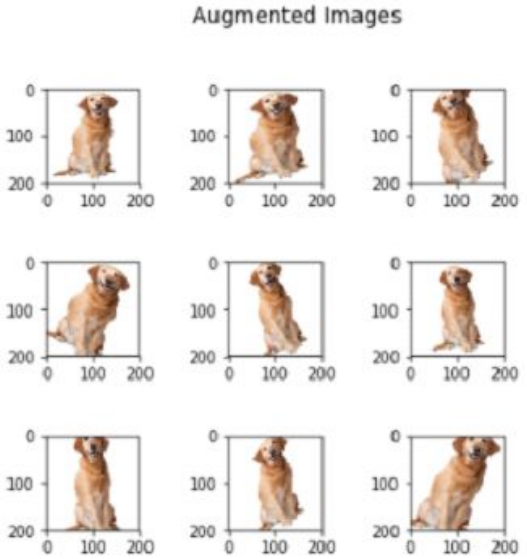
Une bonne accuracy sur les données d'entraînement, mais très faible sur les données de validation
⇒ sur-apprentissage (*overfitting*)

Data augmentation

- Une pratique destinée à **éviter l'overfitting** pendant la phase d'apprentissage

- **Méthodologie :**
  - Duplication des données par la technique du ***mirroring***, rotation de 45°, des shift à gauche et droite et des zooms





Transfer Learning

- **Pourquoi faire du Transfer Learning** :
  - On démarre rarement d'un CNN à partir de zéro.
  - Utiliser des poids d'un modèle pré-entraîné comme initialisations aide à résoudre plus rapidement un problème.
  - Les CNNs sont coûteux en ressources de calcul.

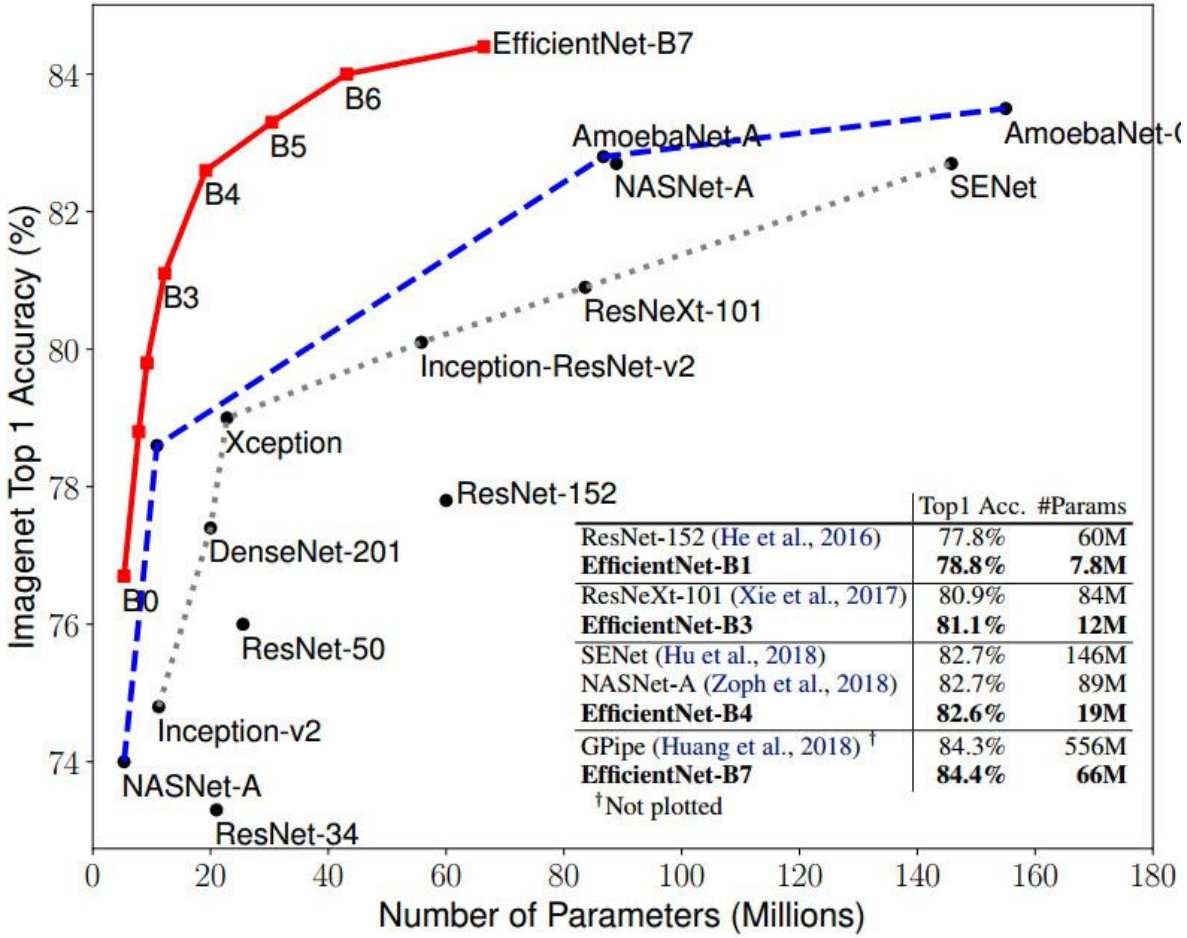
=> Le transfer learning permet d'accélérer, de guider la méthode d'entraînement et choisir les hyper paramètres optimaux.

Transfer learning

- **Méthodologie :**

D'après le graphique ci-contre, j'ai choisi d'initialiser les poids de mon modèle avec ceux de l'EfficientNet-B7 et re-tenter un entraînement

/!\ Problèmes de Out of Memory rencontrés au cours de l'entraînement mais le modèle a réussi à achever une accuracy > 77% en validation



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**78.8%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.1%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.6%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted

Transfer learning et inférence

- **Méthodologie :**
  - Import de ResNet50 depuis keras et l'utiliser pour inférence sur de nouvelles images

Résultat obtenu : le modèle ResNet50 a réussi à classer correctement 6 images différentes de mon chien avec une probabilité de plus de 90%

Next steps

- Réussir à entraîner EfficientNetB7 + les dernières couches de mon modèle (résoudre le problème Out of Memory)
- Choisir entre ce modèle et le ResNet50 pour intégrer le modèle le plus performant dans une API déployée sur le web (Flask, Django)

Livrables

Lien repo GitHub :

- Notebook
- Script python qui prend en argument une image et les poids d'un CNN prenant en entrée un tableau de pixels de dimensions (224, 224, 3) et affiche le résultat de prédiction
- Slides explicatives en guise de support