Classez des images à l'aide d'algorithmes de Deep Learning

Contenu de la présentation

01 Problématique

02 Modèle Personnel

03 Modèle pré-entraînés

04 Simulation

05 Conclusion

01

Problématique

Mission

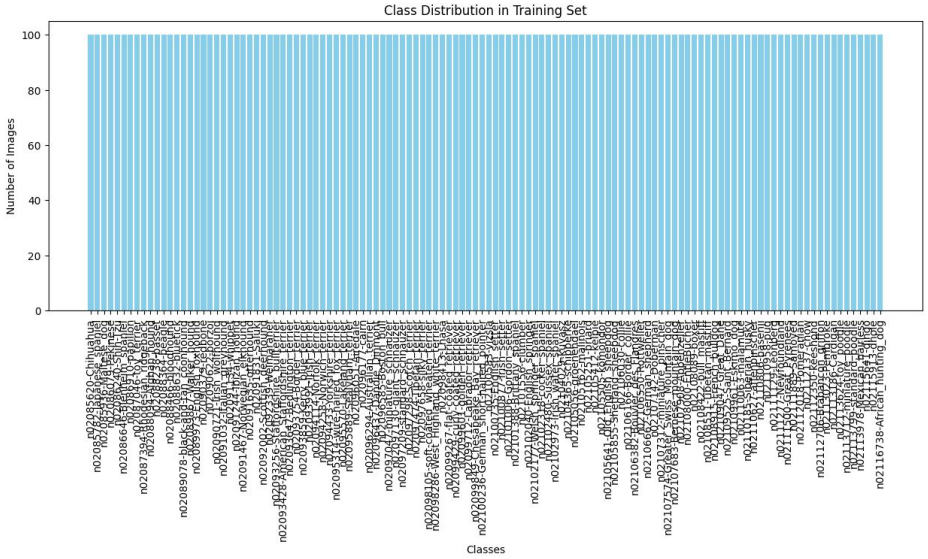
Pour l'association “Le refuge”, créé un algorithme capable de classer les images en fonction de la race du chien présent sur l'image

- Une approche avec notre propre réseau CNN
- Une approche utilisant le transfer learning
- Une démonstration du modèle via un script python, adaptable par la suite sur une API ou autre

La Donnees

- La donnes d'entraînement provient de “Stanford Dogs Dataset”
- Elle contient 120 races de chien pour 20 580 Images
- Elle possèdes des tailles et des qualités diversifier

Répartition des individus dans le train :



02

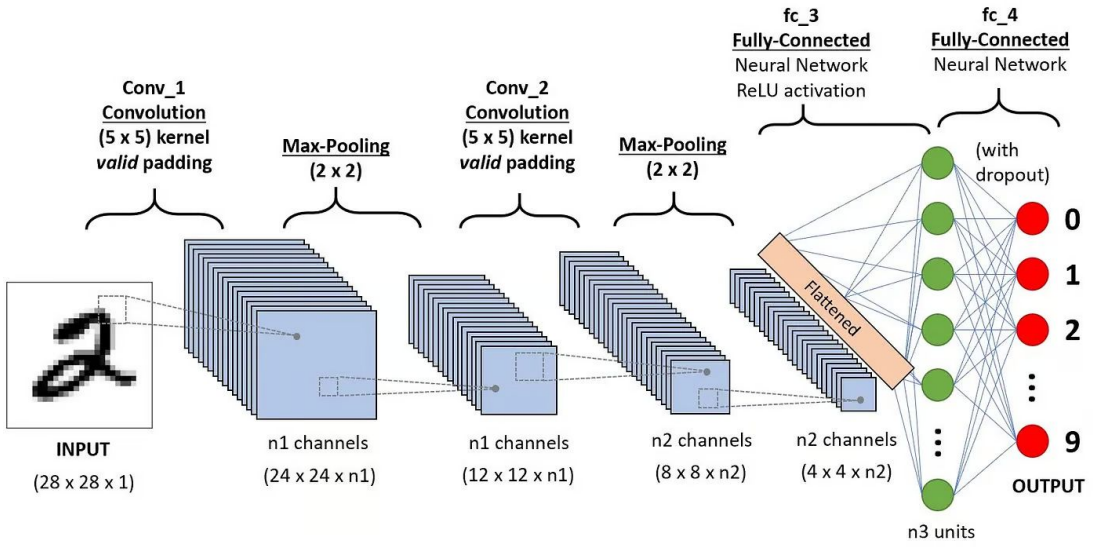
Modèle Personnel

Architecture

Sur la base de l'architecture de MobileNet:

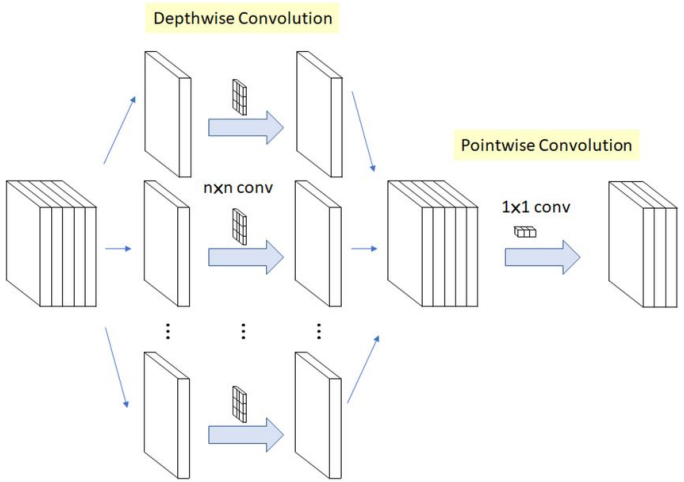
- modèle avec peux de paramètre, idéal pour faire des essais et qui a prouvé son efficacité
- Réseau conçu pour être efficace en termes de calcul et de mémoire, en utilisant des convolutions séparables en profondeur (Depth Wise Séparable Convolution)

CNN



Dans un réseau de neurones convolutif standard (CNN), chaque convolution applique un filtre k x k sur tous les canaux d'entrée en même temps.

Depthwise Separable Convolution

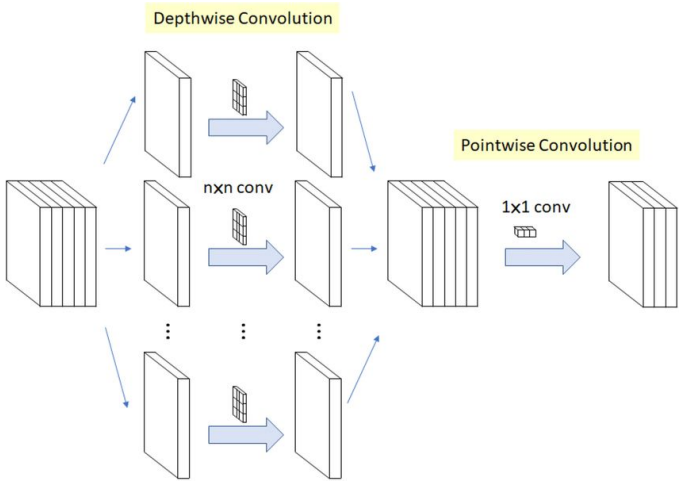


Au lieu d'utiliser une seule convolution complète, on la décompose en deux étapes :

Convolution Depthwise :
Un filtre kxk est appliqué séparément sur chaque canal d'entrée.
Chaque canal d'entrée garde son propre filtre (contrairement à une convolution standard où tous les canaux sont combinés).

Convolution Pointwise :
Une convolution 1x1 est appliquée pour combiner les résultats des convolutions depthwise et modifier le nombre de canaux de sortie.

Depthwise Separable Convolution



Au lieu d'utiliser une seule convolution complète, on la décompose en deux étapes :

Convolution Depthwise :
Un filtre kxk est appliqué séparément sur chaque canal d'entrée.
Chaque canal d'entrée garde son propre filtre (contrairement à une convolution standard où tous les canaux sont combinés).

Convolution Pointwise :
Une convolution 1x1 est appliquée pour combiner les résultats des convolutions depthwise et modifier le nombre de canaux de sortie.

Batch-Size/ epoch

Au lieu de présenter à notre modèle une seul image a la fois, afin de réduire le temps d'entraînement nous pouvons assembler les images en batch, cependant nous devons maîtriser la taille de celui-ci :

- un batch trop petit prendra trop de temps et notre modèle sera mal optimiser
- un batch trop grand saturera notre RAM et VRAM, ce qui mènera au crash durant un epoch

Pour le nombre d’epoch, j’utilise une méthode d'early stopping ou je module la tolérance en fonction de mes besoins

Loss/ Accuracy

Afin de calculer l’erreur j’utilise une cross entropy loss, car performant pour un modèle de classification

Son résultat est alors utiliser pour réaliser la backpropagation et permet de changer les valeurs des poids.

La loss permet de savoir comment le modèle est performant avec son train set cependant afin de vraiment voir la performance d’un modèle, il faut le tester avec de la donne inconnu, pour cela je calcule un score d'accuracy qui permet de déterminer les erreurs de mon modèle

Data Augmentation

Comme notre dataset contient peux d'images et afin de varier à chaque entraînement de notre modèle nous appliquons une data augmentation.

C'est à dire des modification variable a chaque epoch afin de ne pas sur-entraîner notre modèle et qu'il soit performant dans beaucoup plus de setup d'images (example position du chiens, et place sur l'image)

Ici nous avons réalisé et data augmentation :

- Random flip avec une probabilité de 50%
- Un Crop que zoom dans l'image compris dans une range de 1 à 0.8

Resultat

Premier entraînement avec data augmentation :

- Epoch 23/1000, Loss: 4.2104
  Epoch 23/1000, Validation Accuracy: 0.90%

Affinage avec la donnée sans data augmentation :

- Epoch 72/1000, Loss: 3.2619
  Epoch 72/1000, Validation Accuracy: 14.45%

03

Modèle pré-entraînés

Les différents modèles

Restnet : modèle de Microsoft Research conçu autour des "Residual Blocks", qui permet de d'introduire la notion de "skip connections" qui permet de transférer directement l'information d'une couche a une autres ce qui permet de palier au problème des réseau profond qui ont du mal a propager les gradients pendant l'entrainement

Efficientnet : modèle de Google Brain, qui reprend les bases de resnet, mais qui ajoute une notion de "Compound Scaling" qui permet d'ajuster automatiquement la largeur et la profondeure du réseau

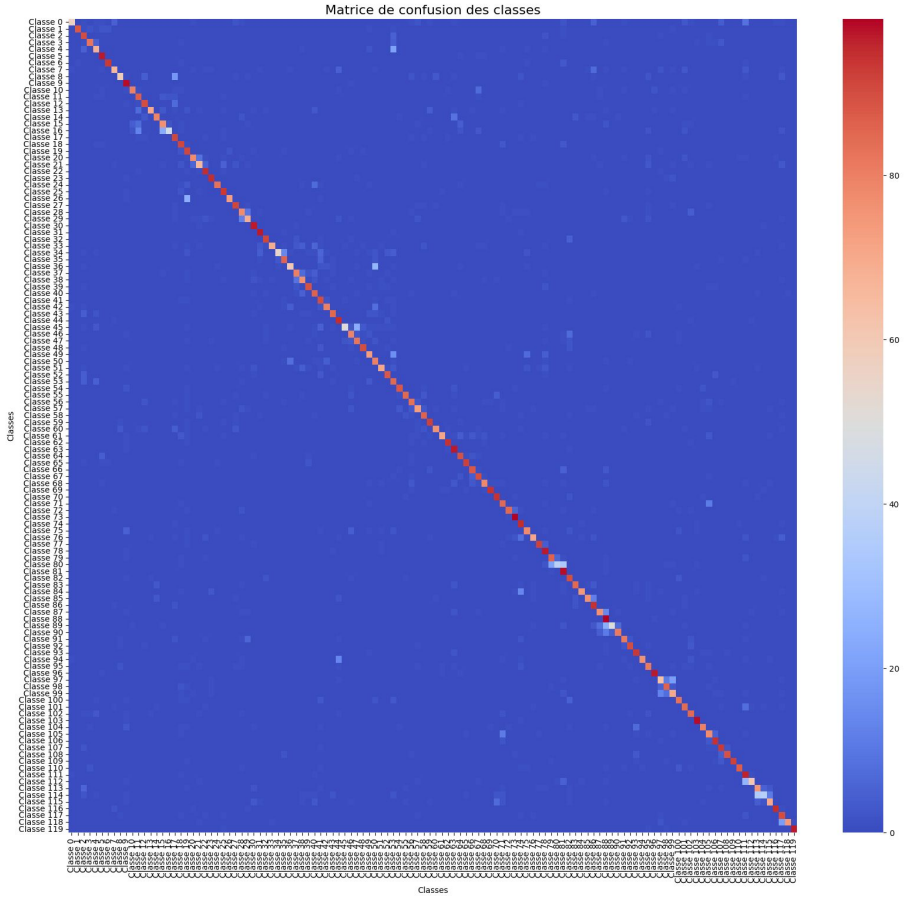
Resultat

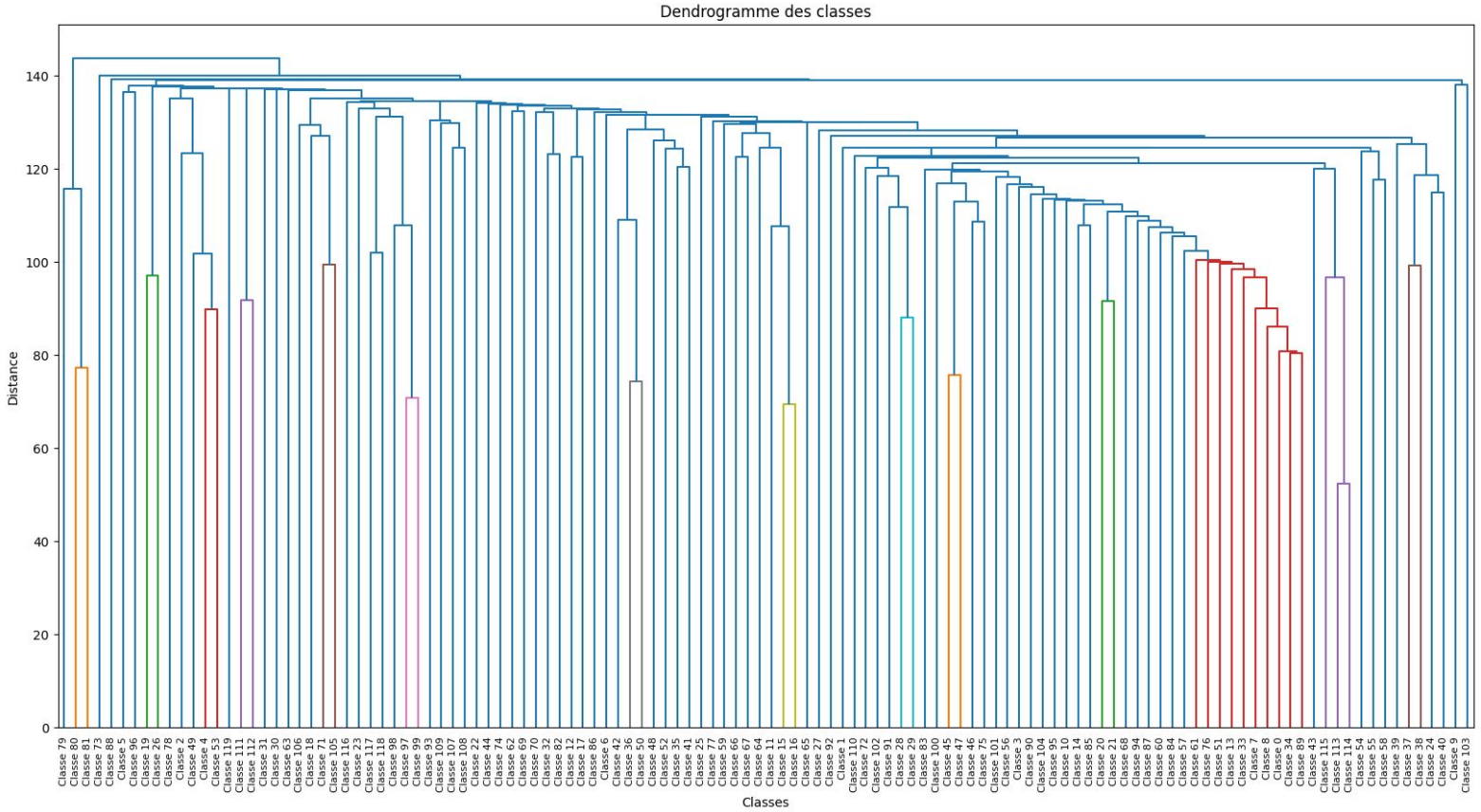
Resnet avec data augmentation :
Loss: 0.8781
Validation Accuracy: 82.20%

Resnet sans data augmentation :
Loss: 0.2969
Validation Accuracy: 70.93%

Efficientnet avec data augmentation :
Loss: 0.7034
Validation Accuracy: 79.02%

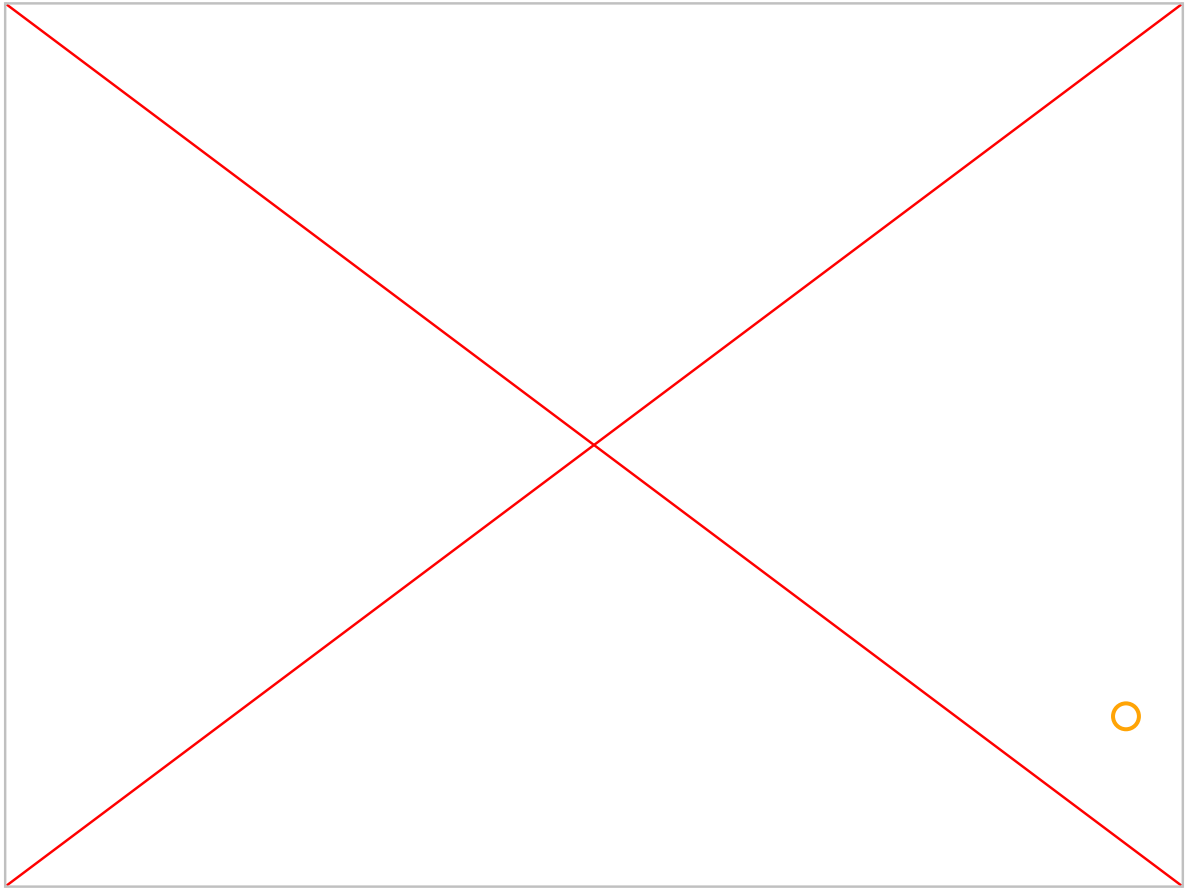
Resnet
choix final pour notre simulation




Dendrogramme des classes
Distance
0
20
40
60
80
100
120
140
Classes
Classe 79
Classe 80
Classe 81
Classe 73
Classe 88
Classe 5
Classe 96
Classe 19
Classe 26
Classe 78
Classe 2
Classe 49
Classe 4
Classe 53
Classe 119
Classe 111
Classe 112
Classe 31
Classe 30
Classe 63
Classe 106
Classe 18
Classe 71
Classe 105
Classe 116
Classe 23
Classe 117
Classe 118
Classe 98
Classe 97
Classe 99
Classe 93
Classe 109
Classe 107
Classe 108
Classe 22
Classe 44
Classe 74
Classe 62
Classe 69
Classe 70
Classe 32
Classe 82
Classe 12
Classe 17
Classe 86
Classe 6
Classe 42
Classe 36
Classe 50
Classe 48
Classe 52
Classe 35
Classe 41
Classe 25
Classe 77
Classe 59
Classe 66
Classe 67
Classe 64
Classe 11
Classe 15
Classe 16
Classe 65
Classe 27
Classe 92
Classe 1
Classe 110
Classe 72
Classe 102
Classe 91
Classe 28
Classe 83
Classe 100
Classe 45
Classe 47
Classe 46
Classe 75
Classe 101
Classe 56
Classe 3
Classe 90
Classe 104
Classe 95
Classe 10
Classe 14
Classe 85
Classe 20
Classe 21
Classe 68
Classe 94
Classe 87
Classe 60
Classe 84
Classe 57
Classe 61
Classe 76
Classe 51
Classe 13
Classe 33
Classe 7
Classe 8
Classe 0
Classe 24
Classe 89
Classe 43
Classe 115
Classe 113
Classe 114
Classe 54
Classe 55
Classe 58
Classe 39
Classe 37
Classe 38
Classe 24
Classe 40
Classe 9
Classe 103

04

Simulation



05

Conclusion

Conclusion

Le modèle “from scratch” pourrait fonctionner si on prend le temps d'entraîner avec plusieurs cycle et en faisant apprendre a nos filtre de convolution à être plus efficace, mais cela serait extrêmement consommateur de ressources, alors que le transfert learning en entraînant que la dernière couche permet d’obtenir de très bon résultat, pour une consommation de ressources et un temps d’investissement beaucoup moins élevé

Un point a été un peu mis de côté lors de ce projet, nous avons sélectionné resnet pour son efficacité, cependant si nous avions mesurer le temps d'entraînement et de prédiction est-ce que nous aurions fait le même choix