PROJET 5 : CLASSEZ DES IMAGES À L'AIDE D'ALGORITHMES DE DEEP LEARNING

SOUTENANCE OPENCLASSROOMS, LE 28/07/2022

ERWAN CHESNEAU

PLAN :

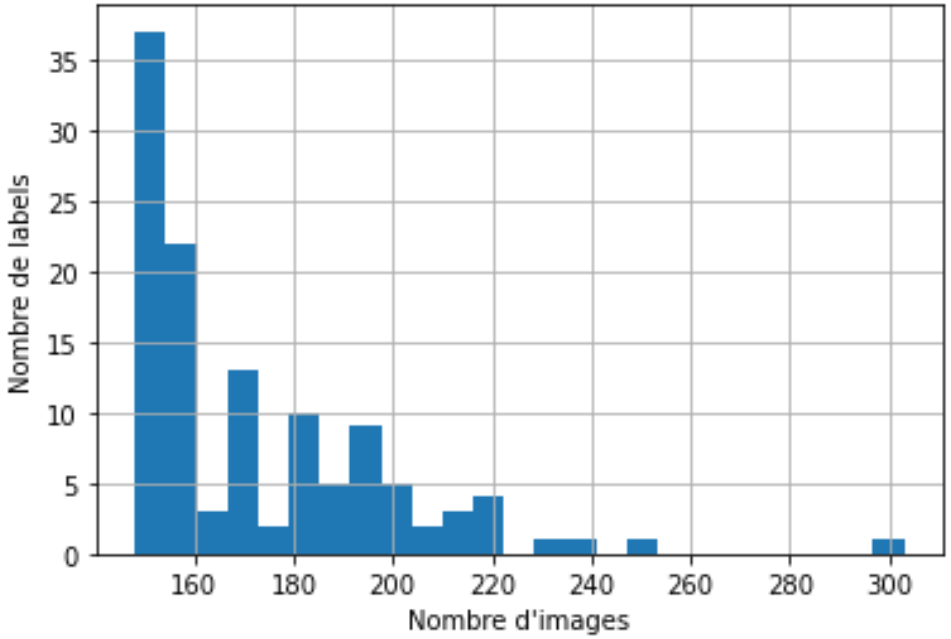
I. Contexte
II. Exploration des données
III. Traitements d'images
IV. Réseaux de neurones simples
V. Transfert learning
VI. Optimisation du modèle sélectionné
VII. déploiement
VIII. Conclusions
IX. Améliorations à envisager

I. CONTEXTE:

- Une association de protection des animaux souhaite référencer les photos des animaux automatiquement
- Développer un algorithme de classification d'image pour déterminer la race du chien sur l'image
- Utiliser la base de donnée Stanford Dogs Dataset
- Démarches :
  - Explorer les images de la base de données
  - Tester différents pré-traitement d'images
  - Développer mon propre NN
  - Tester différents modèles pré-entrainés
  - Déployer un code avec le meilleur modèle

II EXPLORATION DES DONNÉES : LABELS

- Nombre d'images par race :
  - Toutes les races ont au moins 148 images
  - Le choix des races pour le développement à peu d'importance
- Exemples d'images :
  - Tailles différentes
  - Qualités différentes



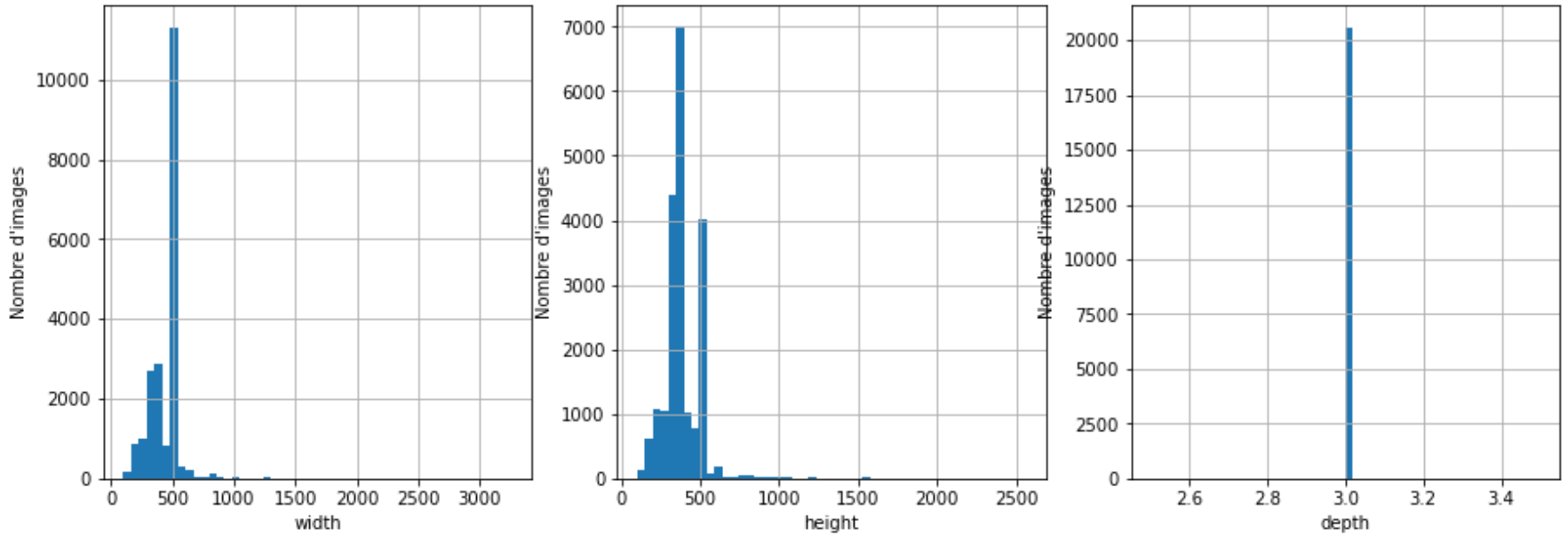
coated_retriever

Pomeranian

Irish_wolfhound

II. EXPLORATION DES DONNÉES : TAILLE

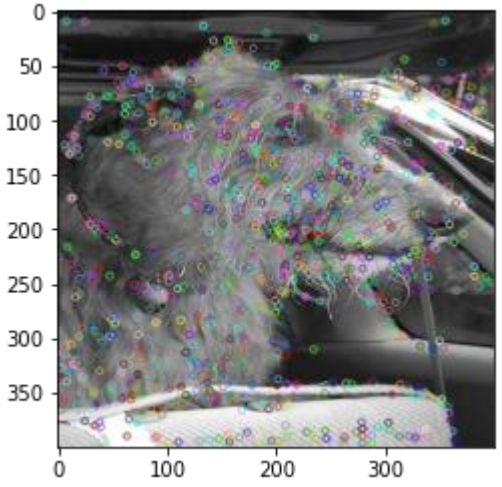
- Distribution de la taille des images :

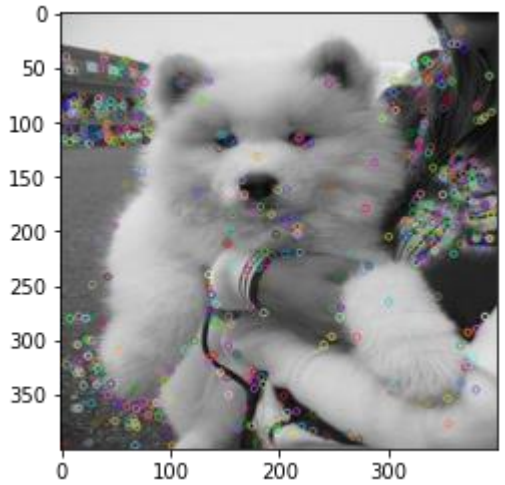


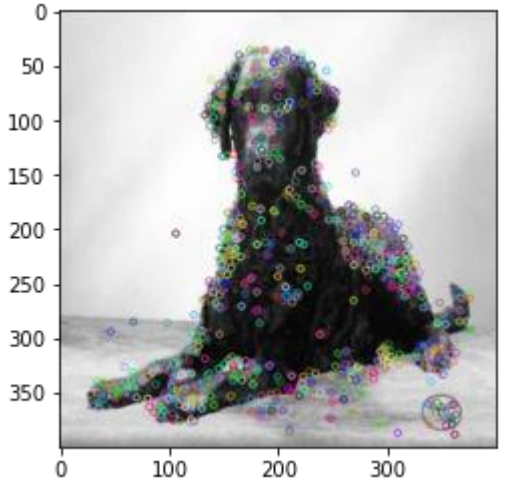
- La taille des images dépend de l'orientation (portrait / paysage)
  - Utilisation de la méthode resize pour changer la taille
  - Le choix de la taille est important :
    - Compromis entre mémoire et précision

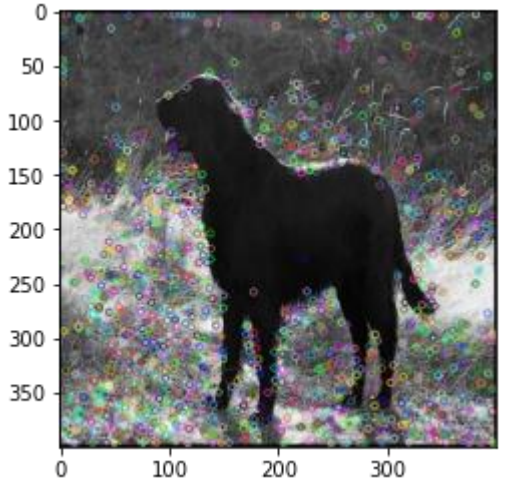
II. EXPLORATION DES DONNÉES : SIFT FEATURES

- Extraction des zones d'intérets de l'image par la méthode SIFT
  - Extraction des descripteurs
  - Extraction de la localisation des points d'intérets
  - Utilisation de la librairie openCV
- Les points d'intérêts ne sont pas forcément sur le chien
  - Difficile à utiliser directement dans un classifieur



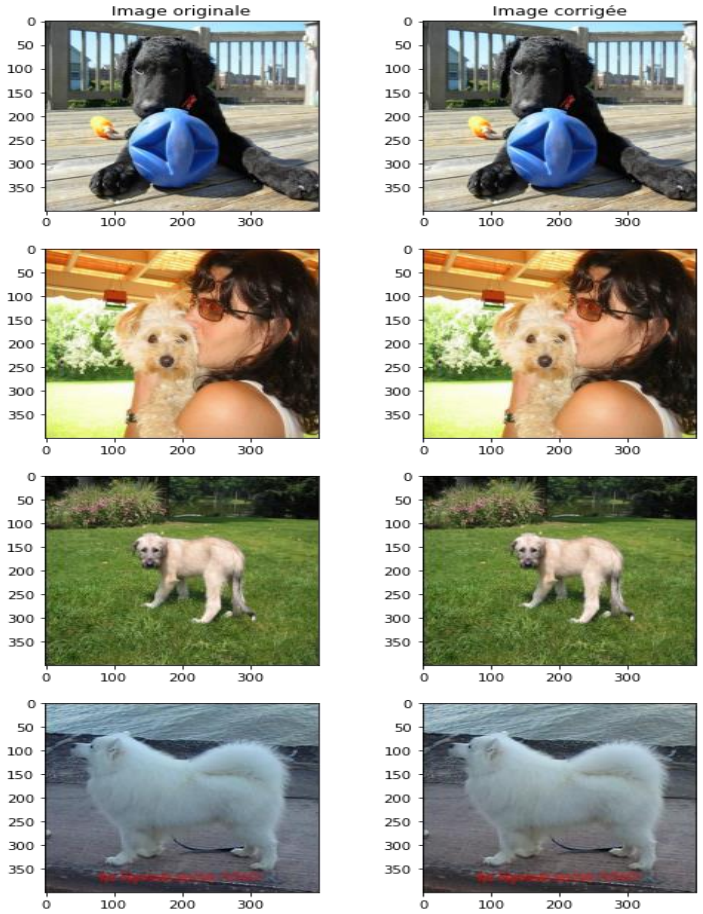






III. TRAITEMENT DES IMAGES : CORRECTION LUMINOSITÉ

- Principe : Etirement de l'histogramme des intensités par une règle de trois

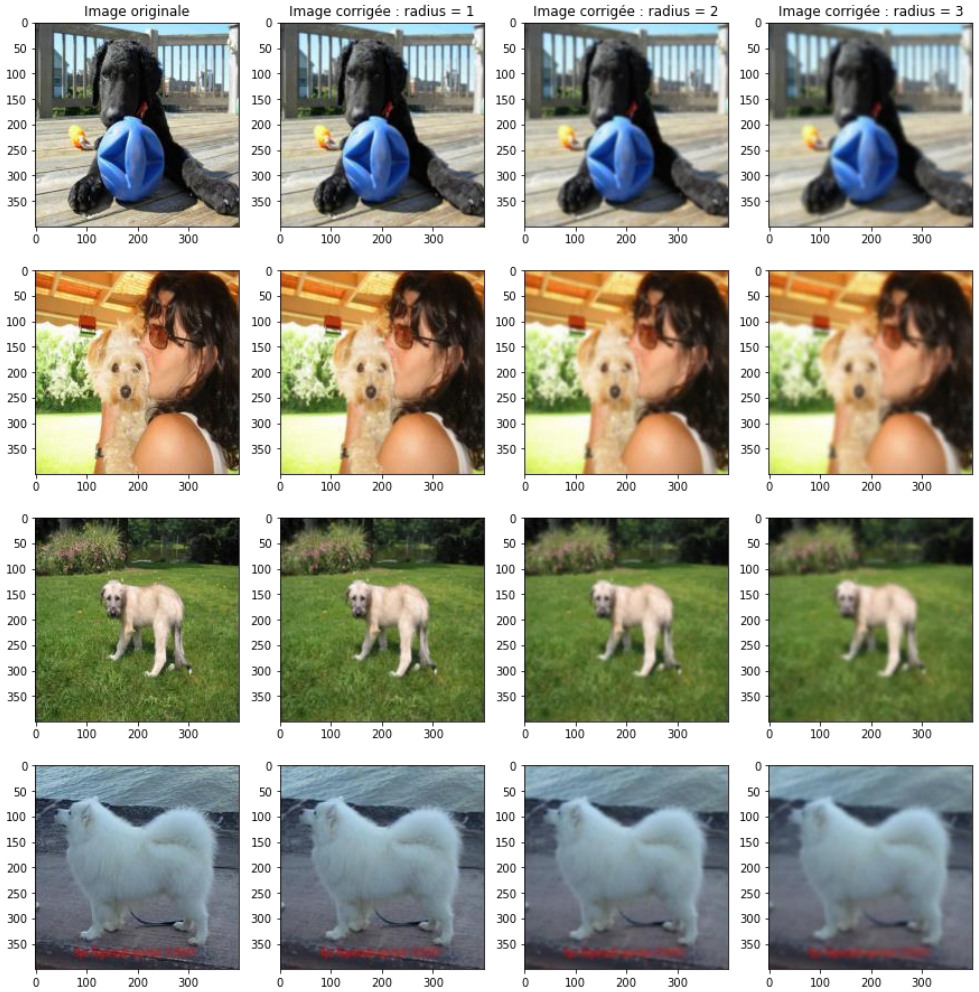


III. TRAITEMENT DES IMAGES : CORRECTION CONTRASTE

- Principe : Corriger les différences d'intensités entre le point le plus sombre et le point le plus clair
  - Normaliser la distribution des intensités pour que chaque niveau de couleur contienne le même nombre de pixels

III. TRAITEMENT DES IMAGES : DÉBRUITAGE

- Principe : Appliquer une correction mathématique pour atténuer le bruit
  - La méthode dépend du type de bruit
  - Pour le bruit aléatoire : application d'un élargissement gaussien sur les niveaux de couleurs



IV. RÉSEAUX DE NEURONES : PRÉPARATION DES DONNÉES

- Chargement des données :
  - Fonction tf. keras.utils.image_dataset_from_directory pour créer des batchs
    - Spécifier le type d'image
    - Spécifier la taille désirée : resizing automatique
    - Création du jeu de validation
  - Création de jeu de test en sortant un nombre de batch du jeu de validation
- Passage des batch en mémoire vive pour améliorer les performances du temps de calculs
- Normalisation des données:
  - NN plus efficaces avec des données dont les valeurs sont comprises entre 0 et 1
  - Couches Rescaling avec 1/255 en argument (256 niveau de couleurs)
  - La couche peut être ajoutée en entrée du NN

IV. RÉSEAUX DE NEURONES SIMPLES : TYPE DE COUCHES

- Couche d'input :
  - Récupère les données en entrées et les transferts vers la couche suivante
- Couche de convolution
  - Extraction des zones d'intérets par l'application de filtre de taille (h,w)
  - Fonction d'activation RelU
- Couche Flatten
  - Conversion des matrices de sortie des couches de convolution en vecteurs
  - Les couches fully connected prennent en entrée des vecteurs
- Couche fully-connected (Dense)
  - Fonction activation RelU
- Couche de prédiction
  - Fonction d'activation softmax pour obtenir la probabilité de chaque classe

IV. RÉSEAUX DE NEURONES SIMPLES : ARCHITECTURE

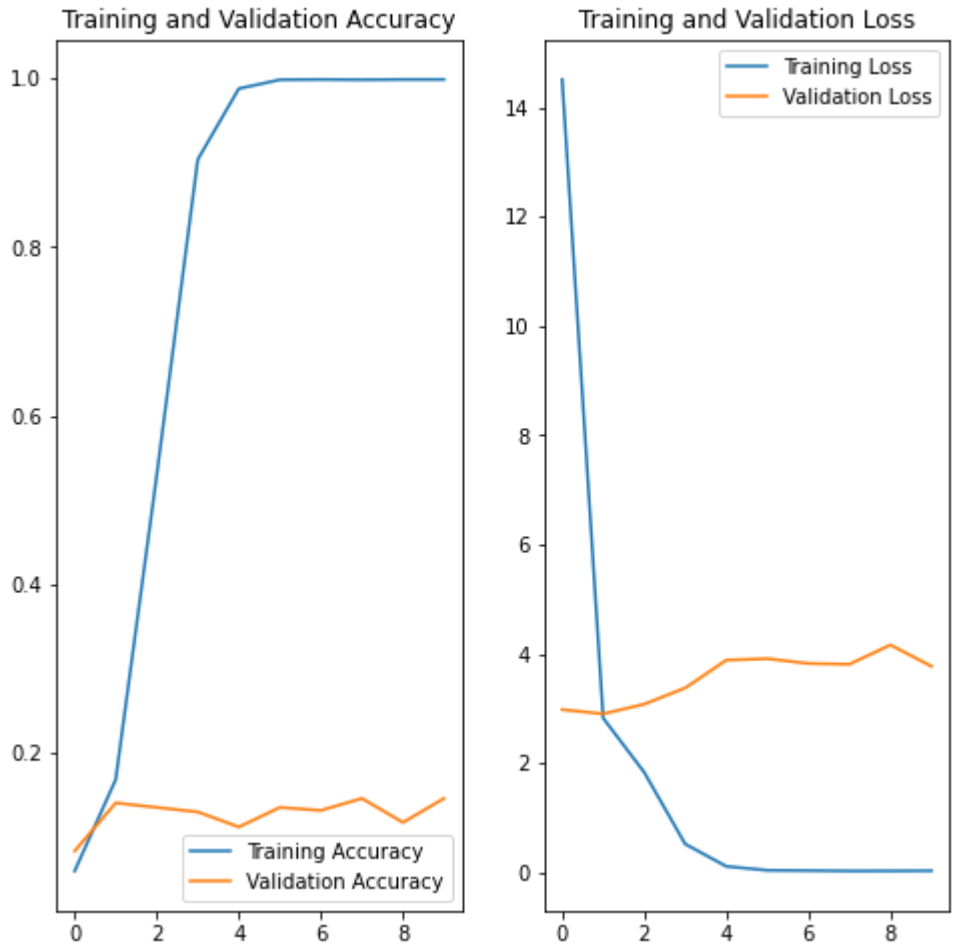
- Une couche de convolution
  - 32 filtres de taille (3,3)
  - Input shape (200,200,3)
  - Output shape (200,200,32)
  - 896 paramètres
- Couche Flatten
- Couche fully-connected
  - 128 neurones
  - 163840128 paramètres
- Couche de prédiction :
  - Taille n_class
  - 2580 paramètres

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 200, 200, 32)      896
flatten_1 (Flatten)          (None, 1280000)           0
dense_2 (Dense)              (None, 128)               163840128
dense_3 (Dense)              (None, 20)                2580
=================================================================
 Total params: 163,843,604
Trainable params: 163,843,604
Non-trainable params: 0
_________________________________________________________________
```

IV. RÉSEAUX DE NEURONES SIMPLES : APPRENTISSAGE

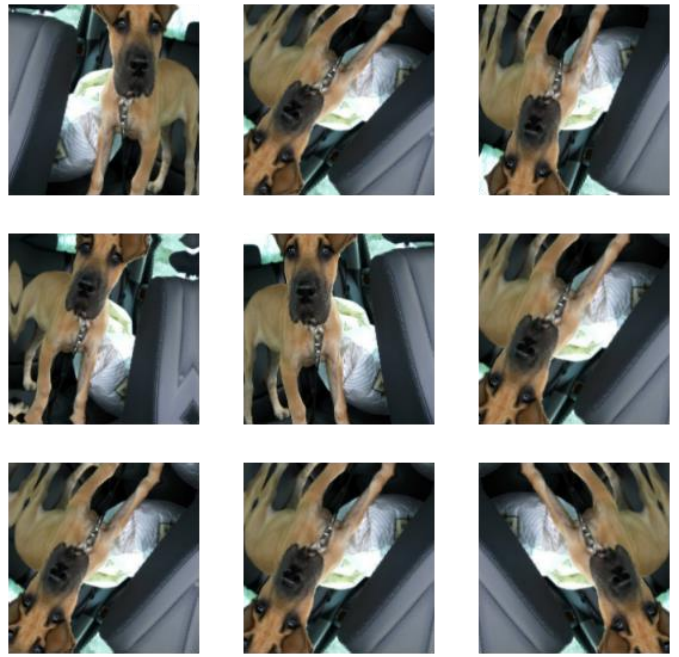
- Jeu d'apprentissage :
  - Fonction de perte diminue jusqu'à une valeur proche de 0
  - La précision augmente jusqu'à 1 environ
- Jeu de validation :
  - Fonction de perte et la précision presque constante…
  - La précision n'est pas beaucoup supérieur à un modèle aléatoire

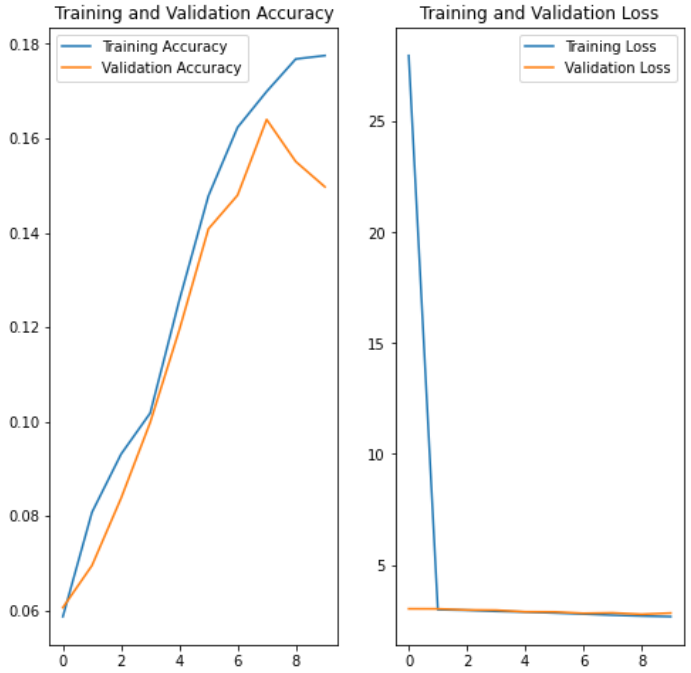
➔ Sur apprentissage



IV. RÉSEAUX DE NEURONES SIMPLES : SUR-APPRENTISSAGE

- Augmentation de données
  - Transformer de manière aléatoire les images en entrée
  - Opération de flip, de rotation et de zoom
  - Permet d'éviter que le modèle apprenne du bruit (features non caractéristiques)
- Apprentissage :
  - Faible différence entre le jeu d'apprentissage et le jeu de validation
  - Niveau de précision plus faible
  - Le modèle est trop simple





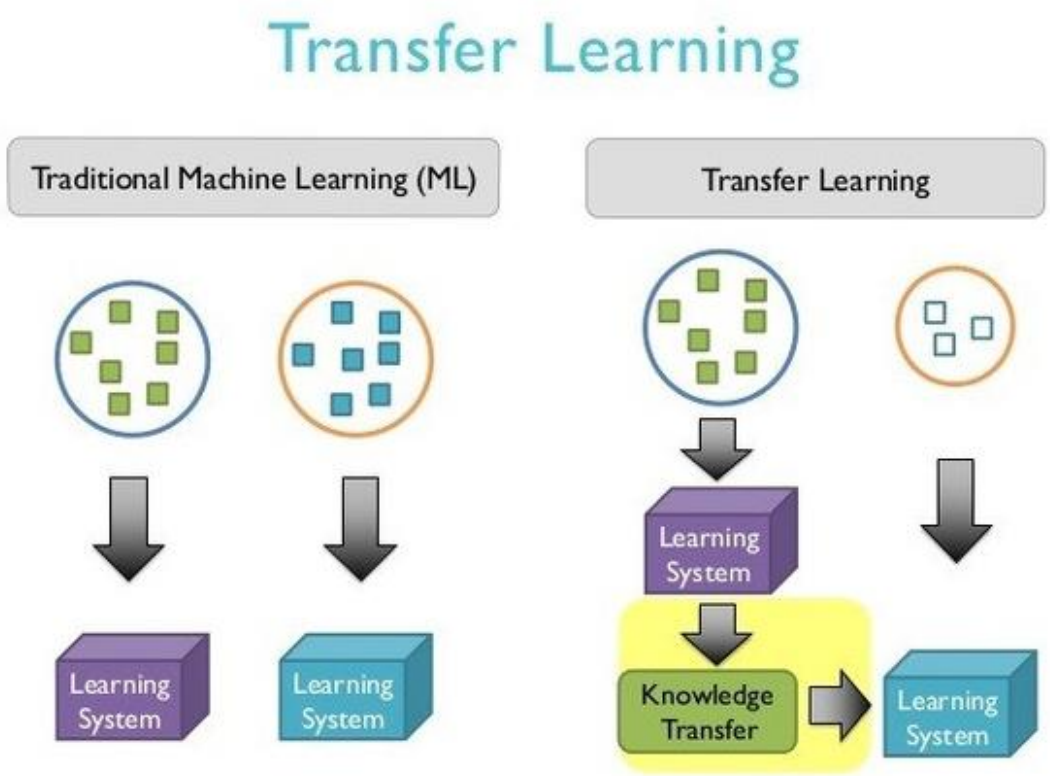
IV. RÉSEAUX DE NEURONES PLUS COMPLEXES

- Plus de couches de convolutions
- Couches de Pooling
- 3 couches Denses


```
_________________________________________________________________
Layer (type) Output Shape Param #
=================================================================
sequential_3 (Sequential) (None, 200, 200, 3) 0
conv2d_32 (Conv2D) (None, 200, 200, 32) 896
max_pooling2d_18 (MaxPoolin (None, 100, 100, 32) 0 g2D)
conv2d_33 (Conv2D) (None, 100, 100, 32) 9248
max_pooling2d_19 (MaxPoolin (None, 50, 50, 32) 0 g2D)
conv2d_34 (Conv2D) (None, 50, 50, 32) 9248
max_pooling2d_20 (MaxPoolin (None, 25, 25, 32) 0 g2D)
flatten_14 (Flatten) (None, 20000) 0
dense_35 (Dense) (None, 128) 2560128
dense_36 (Dense) (None, 128) 16512
dense_37 (Dense) (None, 128) 16512
dense_38 (Dense) (None, 20) 2580
=================================================================
Total params: 2,615,124
Trainable params: 2,615,124
Non-trainable params: 0
```

V. TRANSFERT LEARNING : PRINCIPE

- Utiliser des modèles pré-entrainés
  - Modèles complexes
  - Pré-entrainés avec un gros dataset
- Méthode :
  - Récupérer l'architecture du modèle
  - Charger les poids
  - Conserver que les couches basses
  - Ajout de la couche de prédiction
  - Apprentissage

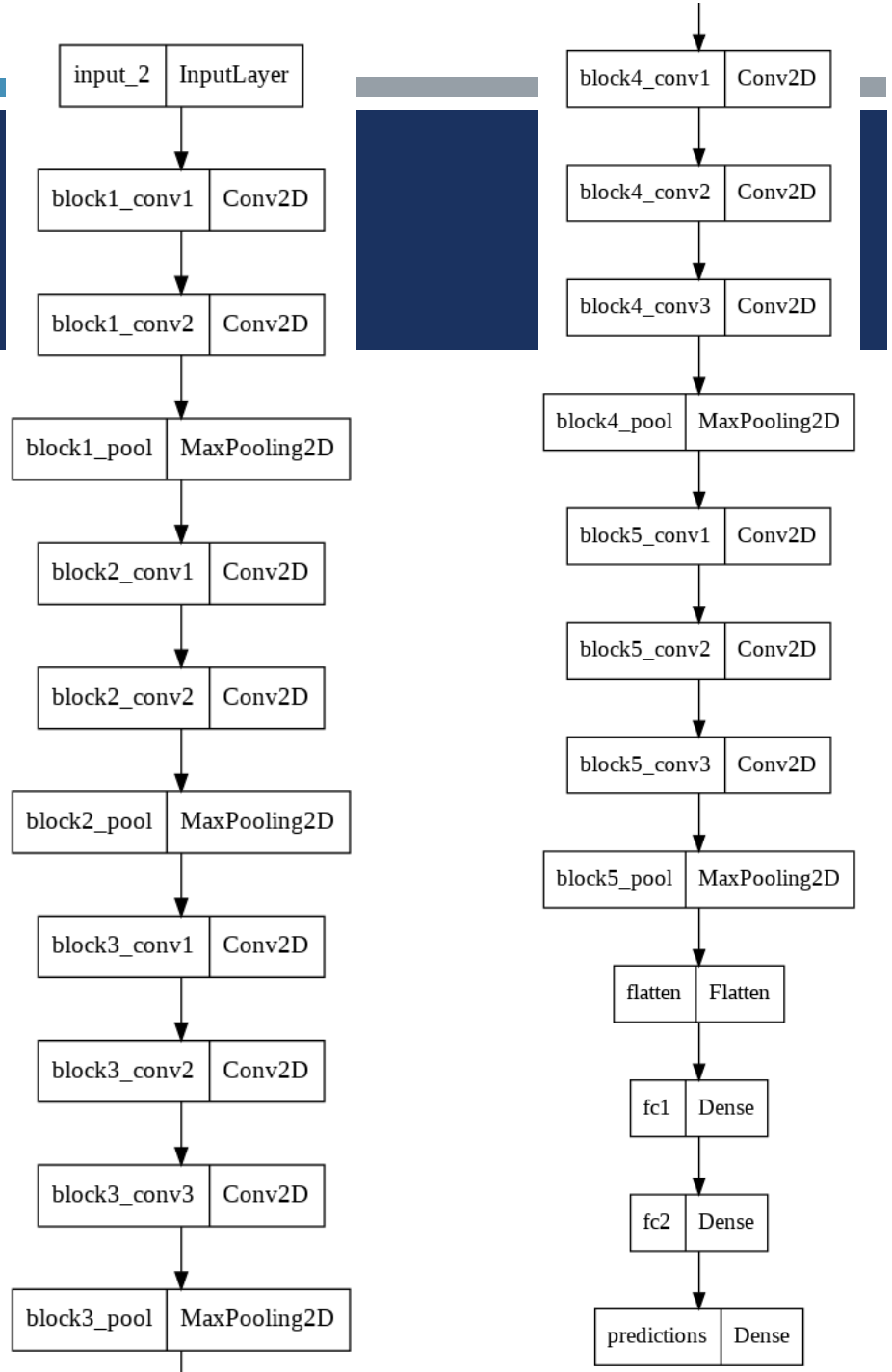


- Test des modèles VGG16 (Université d'Oxford), MobileNet (Google), Xception (Google) et ResNet50
- Sélection de 20 races de chiens

V. TRANSFERT LEARNING :VGG16

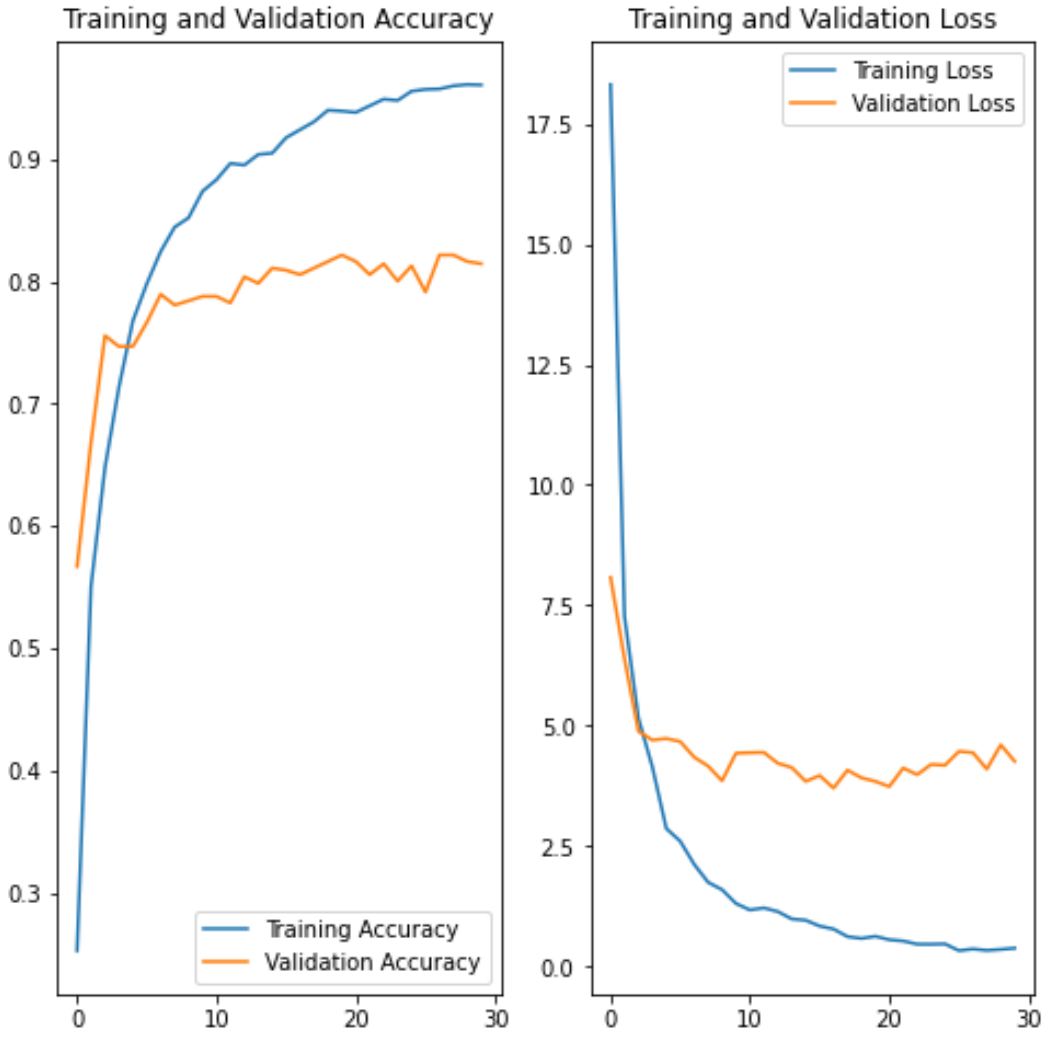
- Présentation du modèle :
  - 5 blocks constitués de :
    - 2 couches de convolutions
    - 1 couche de MaxPooling
- 1 couche Flatten
- 2 couches fully-connected
- 1 couche de prédiction

- On conserve que les couches basses
  - 14,714,688 de paramètres
- Ajout de la couche de prédictions



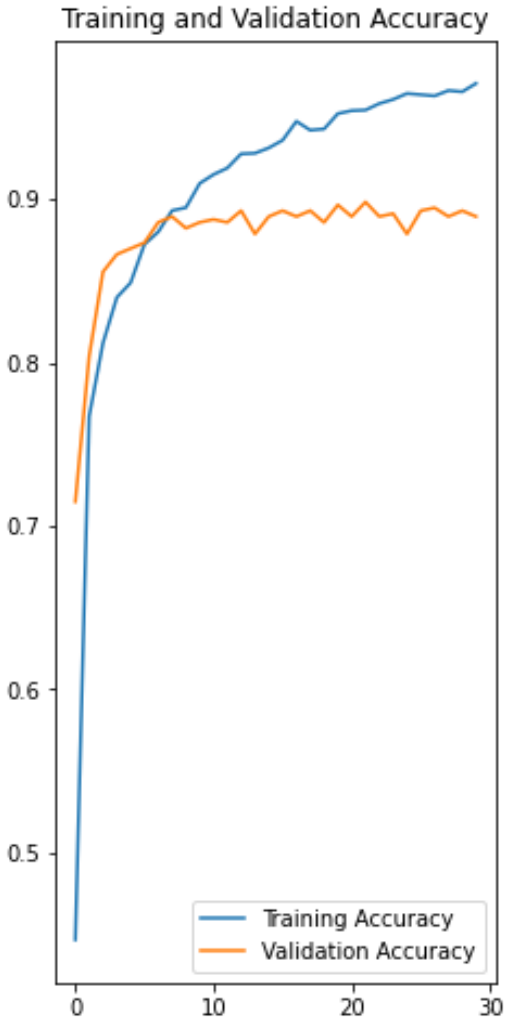
V. TRANSFERT LEARNING :VGG16

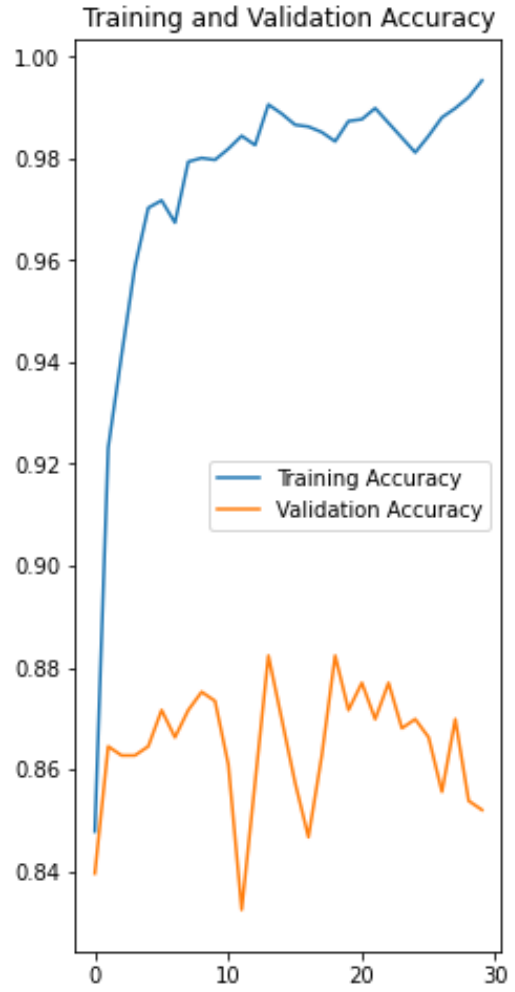
- Apprentissage de la couche de prédiction
  - 501,780 paramètres
  - Début de surapprentissage



V. TRANSFERT LEARNING :VGG16

- Ajout des couches fully-connected
  - 501,780 paramètres
  - Début de surapprentissage
  - Amélioration de la précision
- Apprentissage de toutes les couches fully-connected
  - Sur-apprentissage



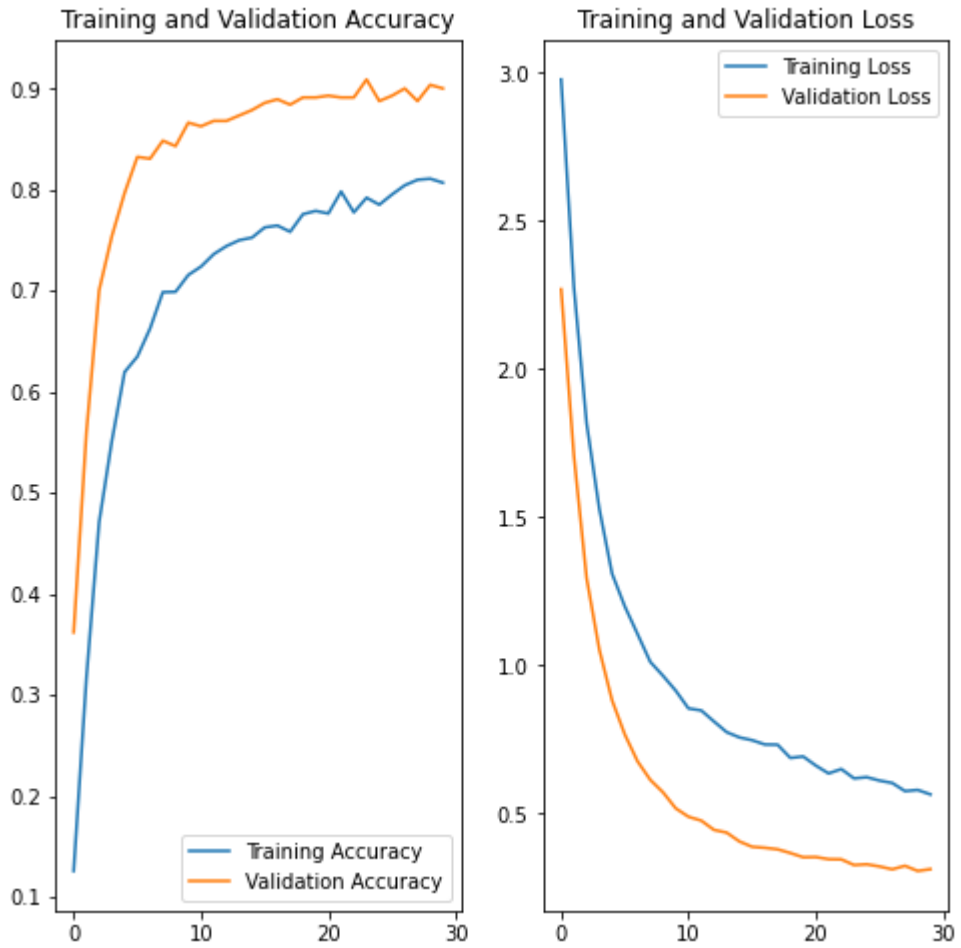


V. TRANSFERT LEARNING : MOBILENET

- Présentation du modèle :
  - Modèle beaucoup plus complexe
  - 16 blocks composés de :
    - Couche de convolution
    - Couche de BatchNormalization : renormalisation des outputs
    - Couche ReLU
    - Couche DepthWiseConv : convolution de chaque channel avec un kernel différent
  - Ne comporte pas de couche fully-connected
  - 3,538,984 paramètres
- Remplacement uniquement de la couche de prédictions

V. TRANSFERT LEARNING : MOBILENET

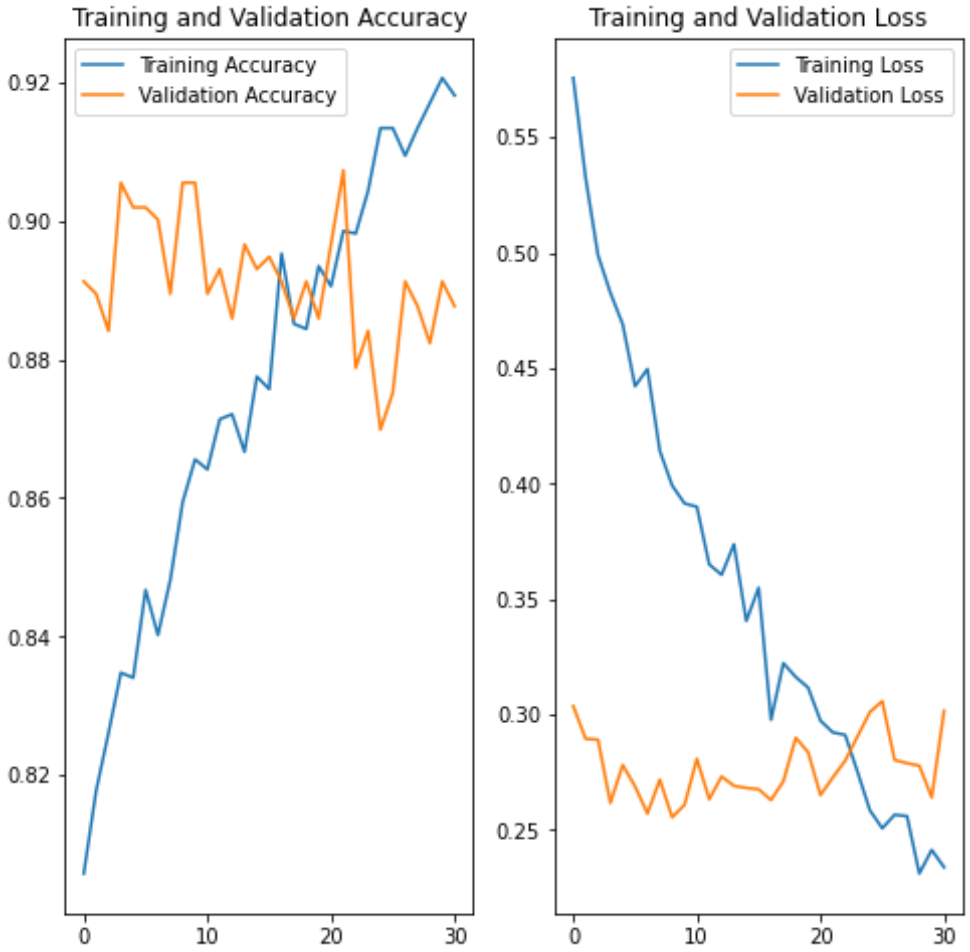
- Apprentissage de la couche de prédiction :
  - 25,620 paramètres
  - Très bonne précision
  - Peu sensible au sur apprentissage



V. TRANSFERT LEARNING : MOBILENET

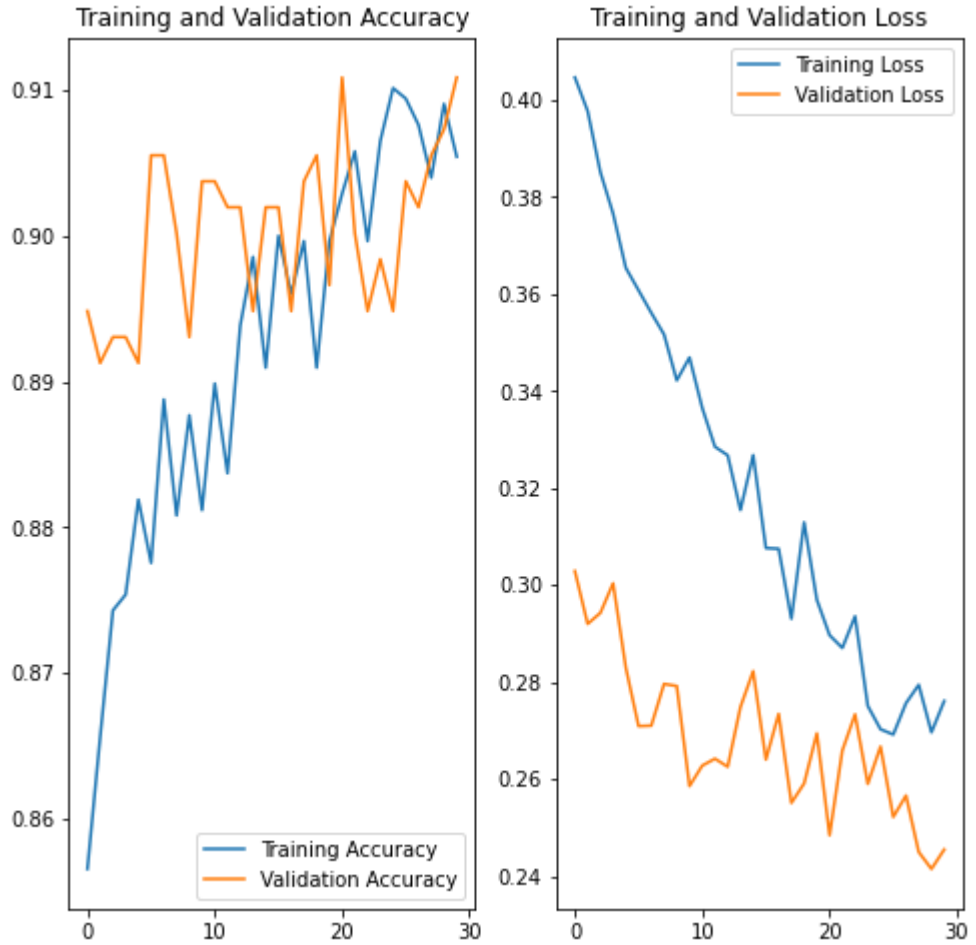
- Fine tunning :
  - Apprentissage des 54 couches les plus hautes (proche de la sortie)
  - 1,887,060 paramètres
  - Amélioration de la précision sur le jeu d'apprentissage
  - Aucun effet sur le jeu de validation

  ➔ Sur apprentissage



VI. OPTIMISATION DU MODÈLE SÉLECTIONNÉ : MOBILENET_V2

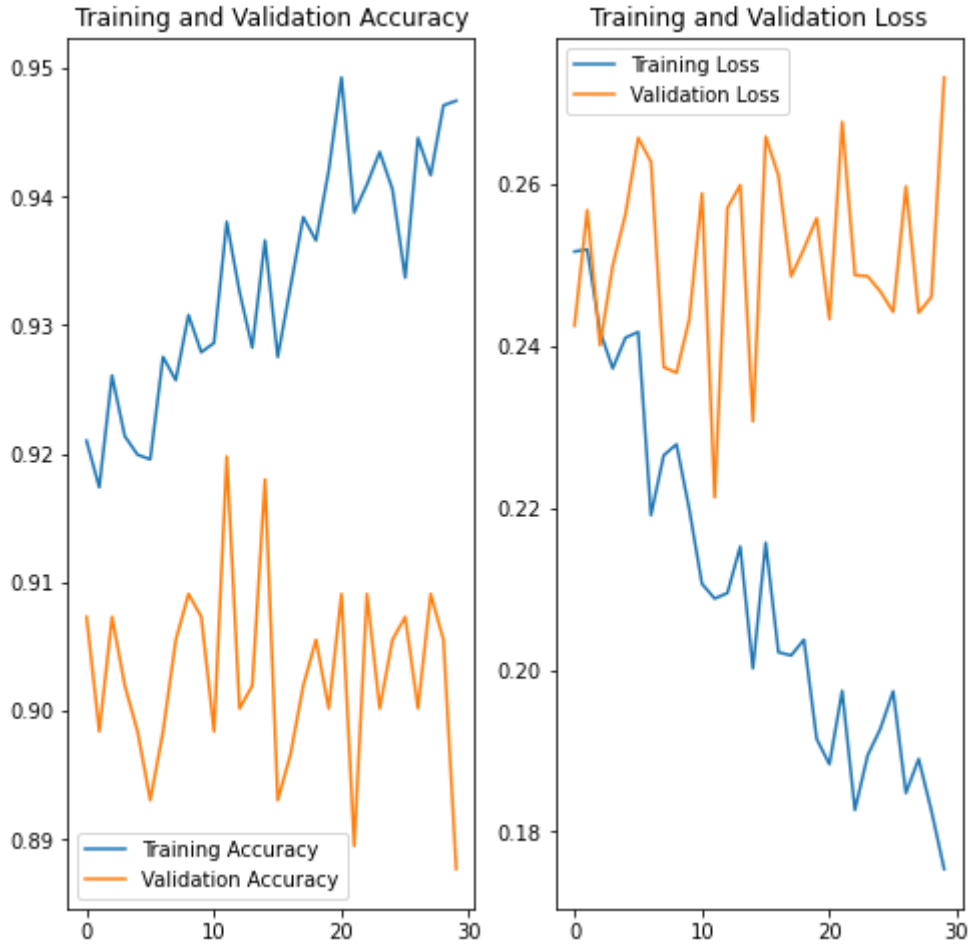
- Diminution de l'augmentation de données et apprentissage de la couche de prédiction uniquement :
  - Permet d'améliorer légèrement la précision
  - Tendance à réintroduire du sur apprentissage
  - Gain d'environ de 2% de précision sur le jeu de validation



VI. OPTIMISATION DU MODÈLE SÉLECTIONNÉ : MOBILENET_V2

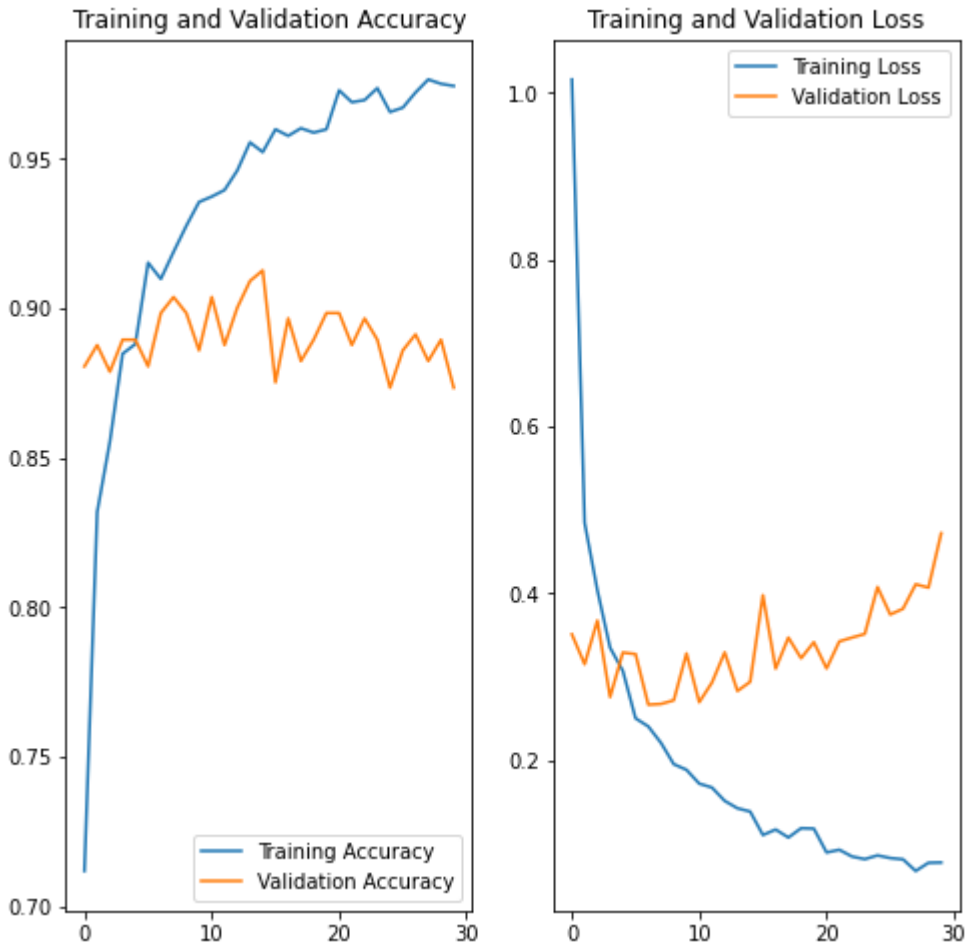
- Diminution du dropout :
  - Permet d'améliorer la précision sur le jeu d'apprentissage
  - Aucun effet sur le jeu de validation

  ➔ Sur apprentissage



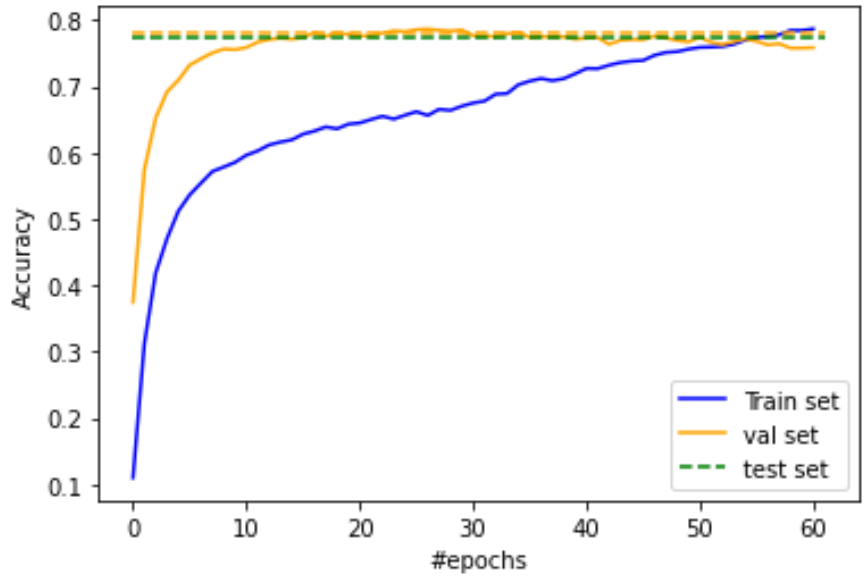
VI. OPTIMISATION DU MODÈLE SÉLECTIONNÉ : MOBILENET_V2

- Ajout de couches fully-connected :
  - 3,304,980 paramètres (forte augmentation)
  - Apprentissage beaucoup plus rapide
  - Précision sur le jeu d'entrainement presque parfait
  - Faible amélioration de la précision du jeu de validation
  - Trop peu de données pour entrainer autant de paramètres

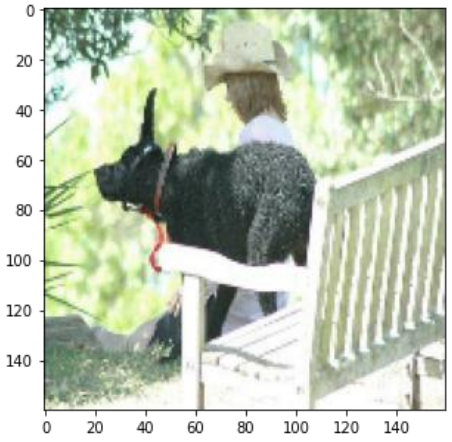


VII. DÉPLOIEMENT

- Modèle MobileNet_V2 choisi pour la bonne précision et la rapidité d'apprentissage
- Apprentissage sur la totalité des données :
  - Précision d'environ 80%
  - Pas de sur-apprentissage
  - Performance sur le jeu de tes similaire également
- Identification des erreurs :

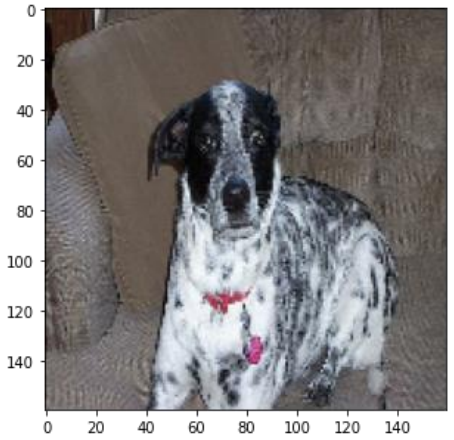


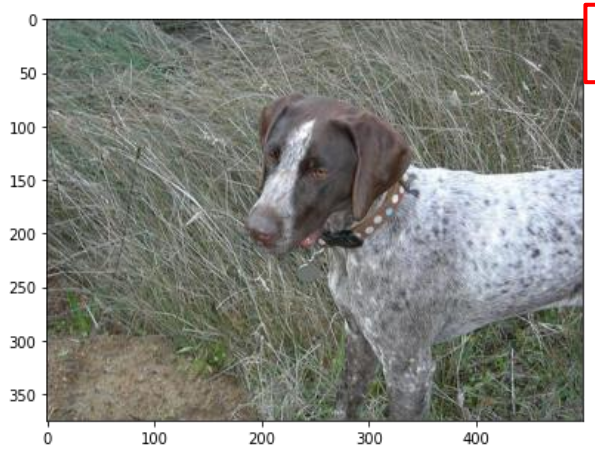
y



Y_pred

y

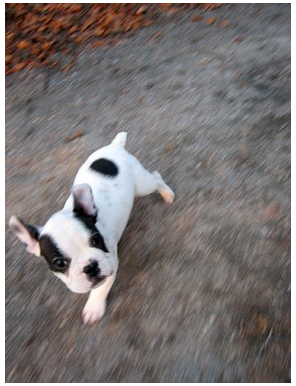




Y_pred

VII. DÉPLOIEMENT

- Création d'un code permettant de :
  - Lire une image en input
  - Charger le modèle entrainé
  - Redimensionner l'image pour respecter l'input shape
  - Créer un batch (ajout d'une dimension)
  - Utiliser le modèle pour obtenir la probabilité de chaque race de chiens
  - Retourner la race de chien avec la plus forte probabilité :
- Démonstration :



VIII. CONCLUSIONS

- La base de données comporte des photos de 120 races de chiens contenant environ 150 images chacunes
- Des essais de traitements d'images ont été réalisés pour corriger et débruiter les images
- La création d'un réseau de neurones from scratch ne permet pas d'obtenir des performances satisfaisantes
- Le transfer learning permet d'améliorer fortement les performances
- Tous les modèles donnent des performances similaires, mais on des couts d'utilisation différents
- Des modèles sont plus simples à ajuster avec un faible nombre de paramètres
- Le modèle MobileNetV2 a été optimisé et déployé
- Un code a été créé pour effectuer la prédiction à partir d'une image donnée en entrée

IX. AMÉLIORATIONS

- Augmenter la taille de la base de données pour lutter contre le sur apprentissage
- Optimiser le modèle sur la totalité des données
  - Utilisation d'un compte pro Google Colab ou AWS avec un GPU
- Essayer d'introduire des données transformées en entrée du modèle
  - Correction du contraste et de la luminosité
  - Images débruitées

MERCI POUR VOTRE ATTENTION !