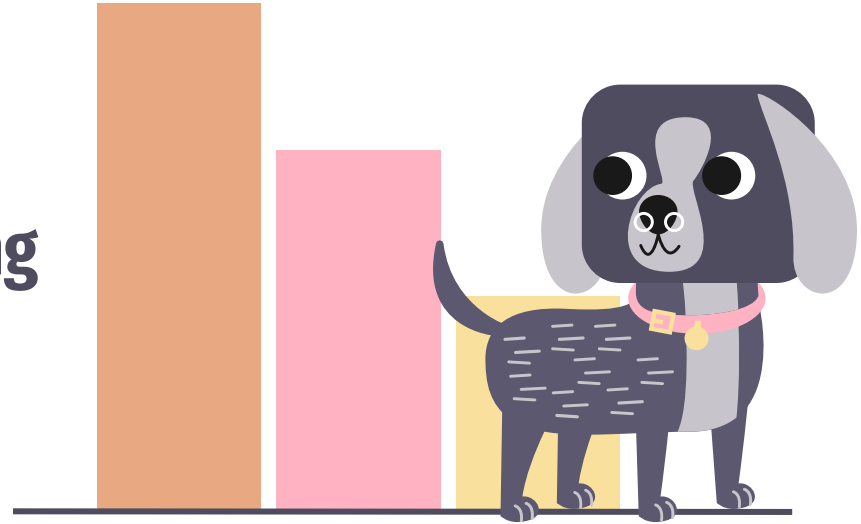
Projet 6 :
Classez des images à l'aide d'algorithmes de Deep Learning

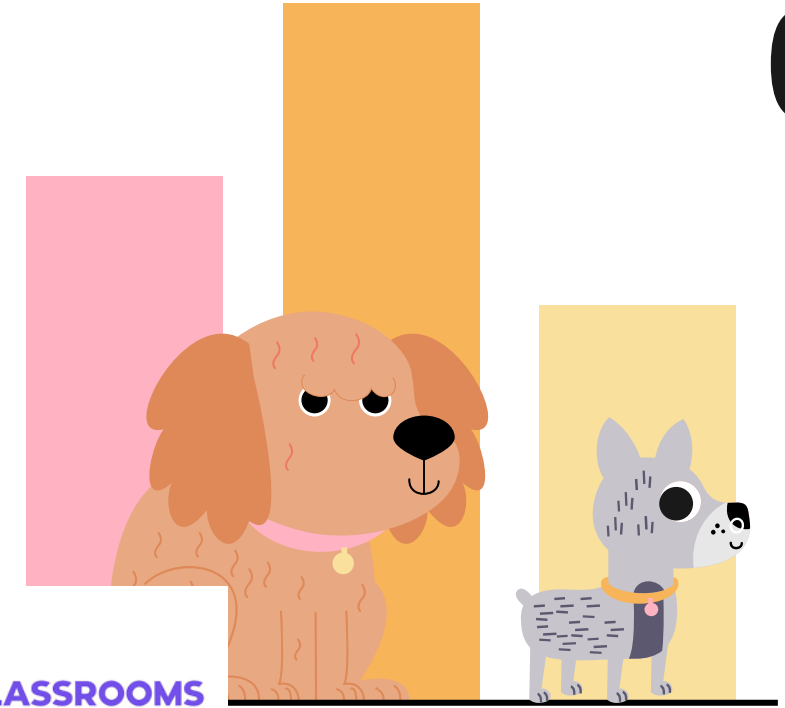


OPENCLASSROOMS

Cédric Randrianarivélo

Table des matières

Comment se déroule la présentation ?



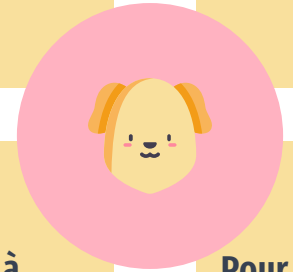
01. Contexte

Cédric Randrianarivélo

01. Contexte

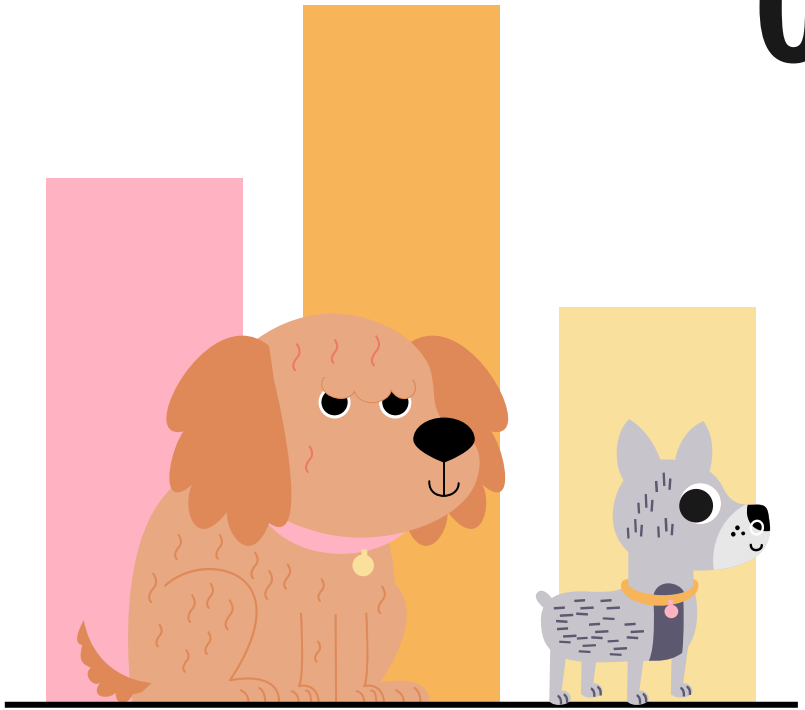
L'association a accumulé de nombreuses images d'animaux, mais n'a pas le temps de les trier. Nous voulons aider à identifier la race des chiens sur les photos.

Les bénévoles de l'association nous laissent à disposition des données sur le site de « Stanford Dogs Dataset » pour pouvoir entrainer notre algorithme.



L'objectif de ce projet est d'aider l'association à développer un algorithme de détection de race des chiens grâce à des photos.

Pour y arriver, nous allons pré-processer des images avec des techniques spécifiques et mettre en œuvre une approche de de Deep Learning pour parvenir à nos fins.



02. Exploration des données

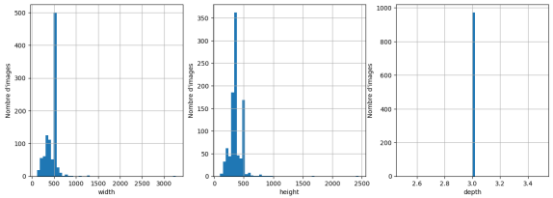
02. Exploration des données

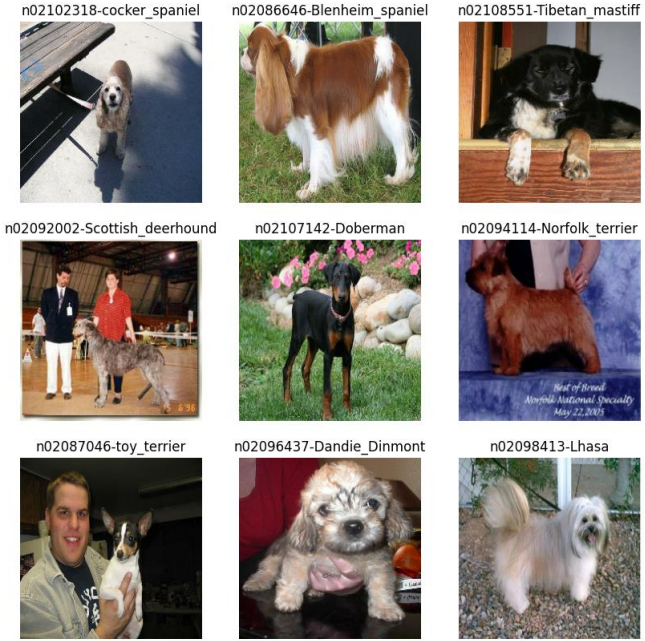
Exploration

Des données provenant du site « Stanford Dogs Dataset »

Environs 20 000 images pour notre modèle.

120 classes de chiens identifiées





02. Exploration des données

Traitement des données

- Analyse d'une image
- Transformation des images sous format pixels

01

Batch size : 64 et Epoch : 25
Pour augmenter la rapidité de traitement et avoir un modèle stable

02

Redimensionnement :
Pour avoir la même taille d'image.
Pour chaque photo.

03

Interpolation :
Bilinear

04

Train - Validation :
80%- 20%

02. Exploration des données

Image size

Image taille : 256 x 256 = > 224 X 224

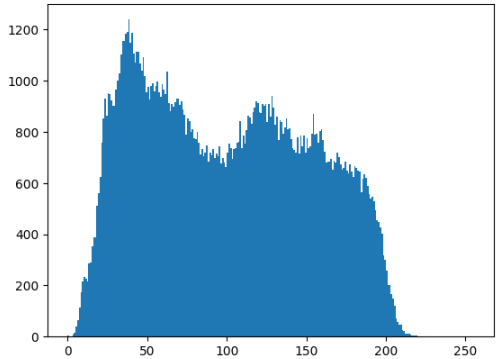
```
array([[[  9,   9,   7],
        [  9,   9,   7],
        [  9,   9,   7],
        ...,
        [ 83, 101, 123],
        [ 84, 102, 124],
        [ 85, 103, 123]],
```

Transformation

- Transformation, des images sous format pixels

02. Exploration des données

- Filtre bilinéaire
- Redimensionnement
- Batch Size 64



03. Modélisation

Cédric Randrianarivélo

03. Modélisation

3 modèles

Modèles A-Z

Modèles crées de A-Z. Avec inspiration sur les modèles en ligne.

Modèles pré-entrainés

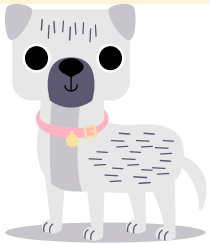
2 modèles

Modèles récupérés dans les bibliotéques. Entrainement des dernières couches avec nos images.

03. Modélisation

Couche Input	Couche qui spécifie la forme d'entrée.
Couche Conv2D	Elle effectue des opérations de convolution pour extraire des caractéristiques spatiales des images. Elles détectent des motifs et des structures dans les données d'entrée, ce qui permet au modèle d'apprendre des représentations de plus haut niveau à partir des pixels bruts.
Couche Flatten	Cette couche transforme les caractéristiques extraites en un vecteur unidimensionnel. Elle réduit la dimension pour la prédiction. Cela permet de passer de la représentation spatiale des caractéristiques à une représentation linéaire qui peut être traitée par les couches Dense.
Couche Dropout	La couche Dropout désactive aléatoirement un pourcentage de neurones à chaque itération, ce qui aide à prévenir le surajustement (overfitting) en encourageant le modèle à généraliser.

Couche Rescaling	Couche qui met à l'échelle les valeurs des pixels des images entre 0 et 1. Cela permet de normaliser les données d'entrée et de faciliter l'apprentissage du modèle en réduisant les variations d'échelle entre les pixels.
Couche MaxPooling2D	Couche qui effectue une opération de pooling pour réduire la dimension spatiale des caractéristiques extraites. Cela permet de réduire le nombre de paramètres et de calculs, tout en préservant les informations les plus importantes
Couche Dense	Couche qui contient le nombre de classes (120) neurones avec une fonction d'activation softmax pour la classification multi classe.
Couche Dense	Ces couches appliquent des opérations linéaires et non linéaires aux caractéristiques extraites. Elles permettent d'apprendre des combinaisons complexes des caractéristiques extraites et de produire des prédictions finales.



03. Modélisation

Modéle

CNN 1 – CNN from scratch

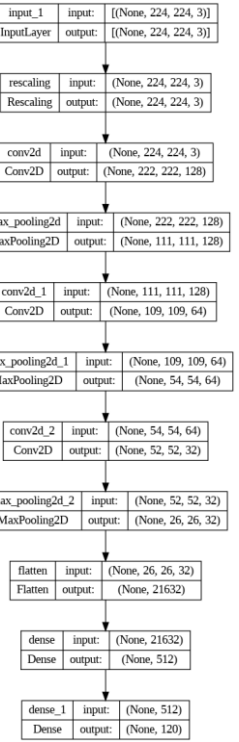
Layers

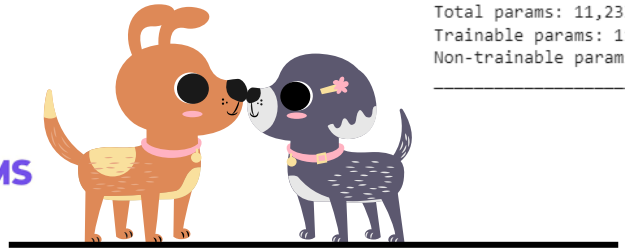
10 couches

Structure

- 1 couches “rescalling”
- 3 couches “Conv2D”
- 3 couches “MaxPooling2D
- 1 couche “Flatten”
- 2 couches “Dense”

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling (Rescaling)       (None, 224, 224, 3)       0

 conv2d (Conv2D)             (None, 222, 222, 128)     3584

 max_pooling2d (MaxPooling2D  (None, 111, 111, 128)    0
 )

 conv2d_1 (Conv2D)           (None, 109, 109, 64)      73792

 max_pooling2d_1 (MaxPooling  (None, 54, 54, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 52, 52, 32)        18464

 max_pooling2d_2 (MaxPooling  (None, 26, 26, 32)       0
 2D)

 flatten (Flatten)           (None, 21632)             0

 dense (Dense)               (None, 512)               11076096

 dense_1 (Dense)             (None, 120)               61560

=================================================================
Total params: 11,233,496
Trainable params: 11,233,496
Non-trainable params: 0
_________________________________________________________________
```





OPENCLASSROOMS

03. Modélisation

Modéle

CNN 2 – CNN from scratch

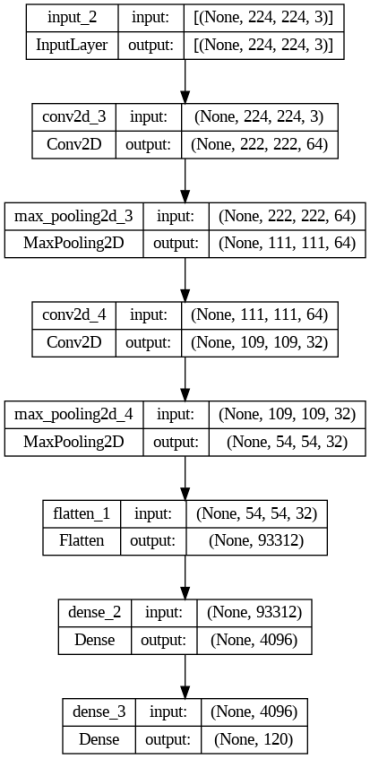
Layers

7 Couches

Structure

- 2 couches “Conv2D”
- 2 couches “MaxPooling2D”
- 1 couche “Flatten”
- 2 couches Dense

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 222, 222, 64)      1792

 max_pooling2d_3 (MaxPooling  (None, 111, 111, 64)     0
 2D)

 conv2d_4 (Conv2D)           (None, 109, 109, 32)      18464

 max_pooling2d_4 (MaxPooling  (None, 54, 54, 32)       0
 2D)

 flatten_1 (Flatten)         (None, 93312)             0

 dense_2 (Dense)             (None, 4096)              382210048

 dense_3 (Dense)             (None, 120)               491640

=================================================================
Total params: 382,721,944
Trainable params: 382,721,944
Non-trainable params: 0
_________________________________________________________________
```



03. Modélisation

Modéle

CNN 3 – Réseau from scratch

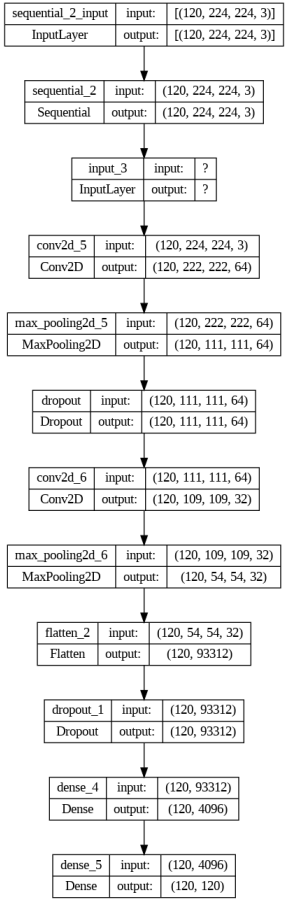
Layers

12 layers

Structure

- 1 couche data augmentation
- 2 couches “Conv2D”
- 2 couches “dropout”
- 2 couches “MaxPooling2D”
- 1 couche “Flatten”
- 2 couches Dense

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_2 (Sequential)   (120, 224, 224, 3)        0

 input_3 (InputLayer)        multiple                  0

 conv2d_5 (Conv2D)           (120, 222, 222, 64)       1792

 max_pooling2d_5 (MaxPooling  (120, 111, 111, 64)      0
 2D)

 dropout (Dropout)           (120, 111, 111, 64)       0

 conv2d_6 (Conv2D)           (120, 109, 109, 32)       18464

 max_pooling2d_6 (MaxPooling  (120, 54, 54, 32)        0
 2D)

 flatten_2 (Flatten)         (120, 93312)              0

 dropout_1 (Dropout)         (120, 93312)              0

 dense_4 (Dense)             (120, 4096)               382210048

 dense_5 (Dense)             (120, 120)                491640

=================================================================
Total params: 382,721,944
Trainable params: 382,721,944
Non-trainable params: 0
```



03. Modélisation

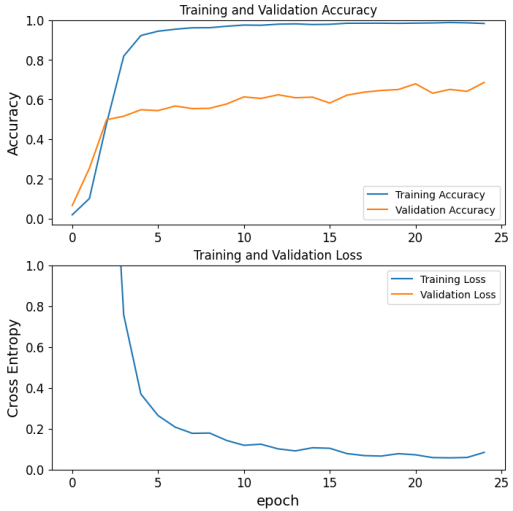
OPENCLASSROOMS

Modèle 1

CNN 1 : 10 Layers

Résultat améliorable

Train	95%
Validation	68%

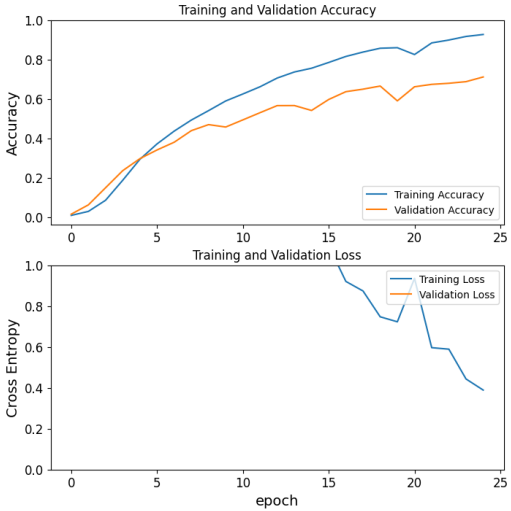


Modèle 2

CNN 2 : 7 Layers

Bon résultat

Train	92%
Validation	72%

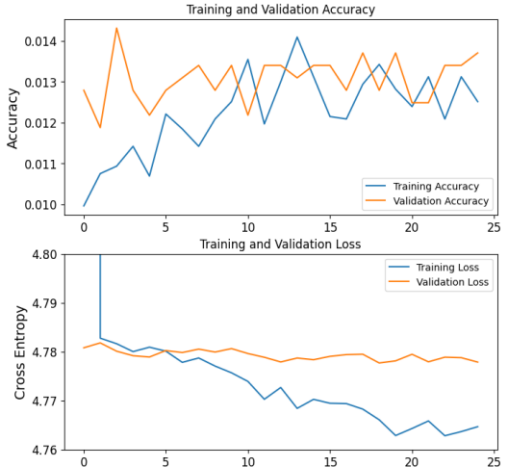


Modèle 3

CNN 3 : 10 Layers

Mauvais résultats

Train	
Validation	



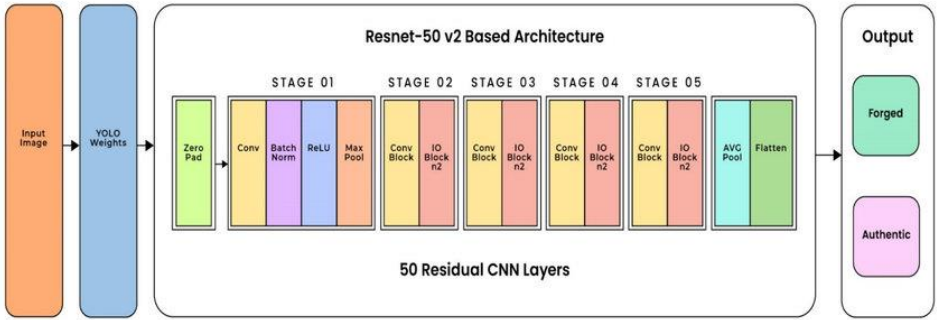
03. Modélisation

ResNet50V2

Modèle pré-entrainé

3 couches entrainées

```python
# Creer un modele ResNet50V2 pre-entraine
base_model = ResNet50V2(include_top=False, weights='imagenet', input_shape=(224, 224, 3))

for layer in base_model.layers:
    layer.trainable = False

# Ajouter des couches de classification personnalisées
x = keras.layers.Flatten()(base_model.output)
x = keras.layers.Dense(4096, activation='relu')(x)
x = keras.layers.BatchNormalization()(x)
predictions = keras.layers.Dense(num_classes, activation = 'softmax')(x)
```



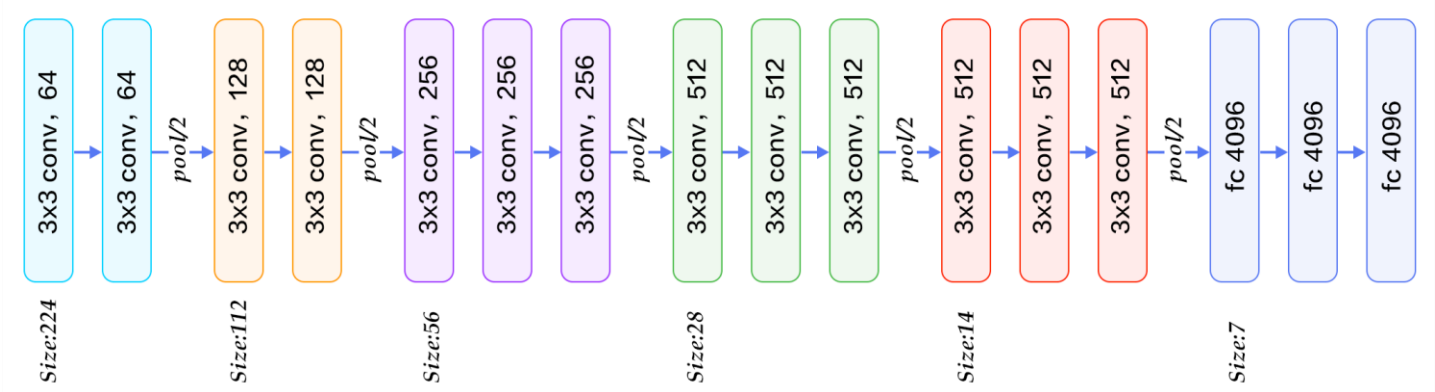
03. Modélisation

VGG16

Modèle pré-entrainé

9 dernière couches entrainées

```
base_model = VGG16(weights="imagenet", include_top=False, input_shape=(224,224,3))

for layer in base_model.layers:
    layer.trainable = False

x = keras.layers.Flatten()(base_model.output)
x = keras.layers.Dropout(0.5)(x)
x = keras.layers.Dense(4096, activation='relu')(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Dropout(0.5)(x)
x = keras.layers.Dense(4096, activation='relu')(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Dropout(0.5)(x)
predictions = keras.layers.Dense(num_classes, activation = 'softmax')(x)

model = keras.Model(inputs=base_model.input, outputs=predictions)
```



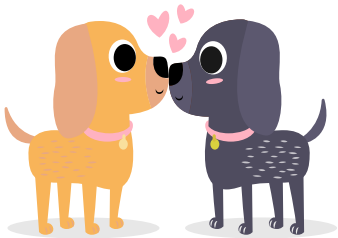
03. Modélisation

Modèle 1 Transfert learning

CNN transfert learning – ResNet50V2

Bon résultat

Train	**93%**
Validation	**48%**

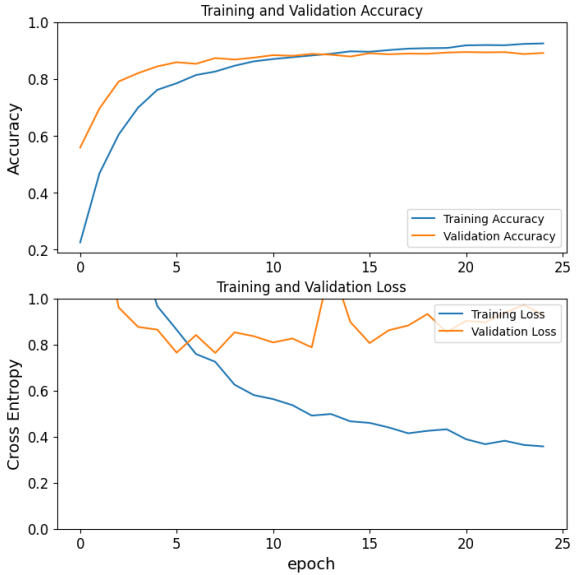


Modèle 1 Transfert learning

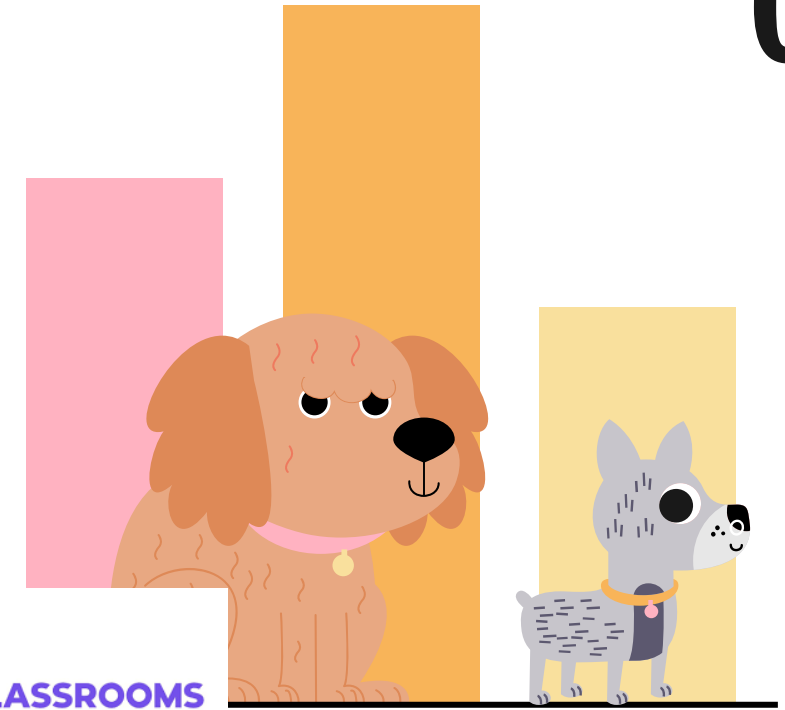
CNN transfert learning - VGG16

Très bon résultat – Modèle retenu

Train	**92%**
Validation	**89%**



04. Conclusion



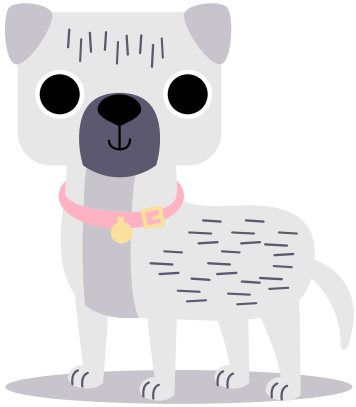
04. Conclusion

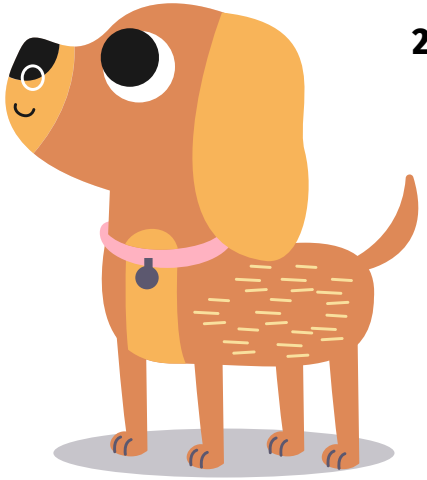
- Manque de temps. Plusieurs pistes intéressantes auraient pu être explorées ave plus de temps.
- La création d'un réseau de neurones à partir de zéro n'a pas donné des performances satisfaisantes.
- Meilleure performance sur le modèle transfert learning.
- Certains modèles étaient plus faciles à ajuster avec un nombre de paramètres réduit.
- Choix sur VGG16 - Modèle de transfert Learning plus pertinents correction des dernières lignes

04. Conclusion

Les axes d'améliorations

Modèle	Un nombre d' Epoch plus grand pour un meilleur entrainement des modèles. Contrainte de performance - le google colab se déconnecté avec trop d'epoch.
Chargement image	Le nombre d'échantillons d'entrainement utilisés auraient pu être augmenté. Cela aurait augmenté l'efficacité des calculs et la stabilité de l'apprentissage. Mais augmente l'utilisation de la mémoire.
Performance	Le projet était bridé par la performance de google colab en termes de GPU. Des solutions cloud ou même une version premium de google collab peuvent être envisagées pour augmenter la performance.
Traitement des images	Autres traitements à explorer. (ex : Autres filtres que bilinéaire, échelle de gris)
Transformation des variables	Les pistes de correction du contraste, équalization et débruitage aurait pu être mise en œuvre pour améliorer la performance.





Merci pour votre attention !

Cédric Randrianarivélo