

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

Mestapha Oumouni

P5 OC parcours IML
Mentor: Samir Tanfous
22/09/2022



Sommaire

Problématique et contexte du projet

Analyse exploratoire des données

Pré-processing des images

Classification avec CNN from scratch

Transfert learning et modèle élu

Conclusion

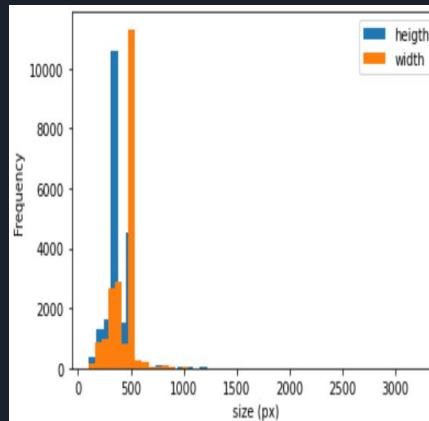


Problématique et contexte

- Association de protection des animaux veut réaliser une indexation suivant les races des chiens
- Classification des images suivant la race des chiens afin d'accélérer le processus d'indexation
- Données: Stanford Dogs dataset
- Beaucoup de races et peu d'images/race
 - CNN from Scratch
 - Transfert learning

120 races

Images RGB (une avec couche alpha corrigée)





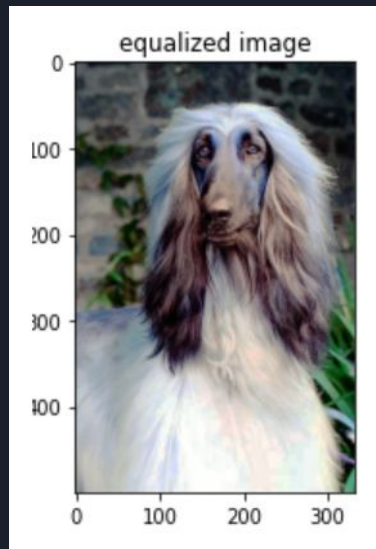
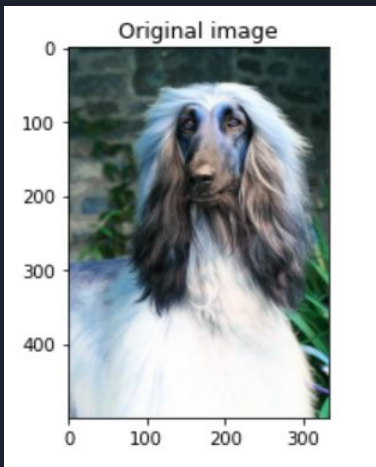
Pré-processing des images



Égalisation d'histogramme

Égalisation d'histogramme est une méthode d'ajustement du contraste
répartir les intensités sur l'ensemble du spectre de l'image

```
image = cv2.imread(uri)
# Equalization
r_image, g_image, b_image = cv2.split(image)
r_image_eq = cv2.equalizeHist(r_image)
g_image_eq = cv2.equalizeHist(g_image)
b_image_eq = cv2.equalizeHist(b_image)
image_eq = cv2.merge((r_image_eq, g_image_eq, b_image_eq))
cmap_val = None
```

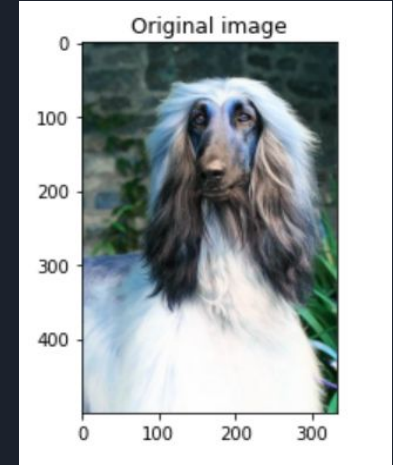


Débruitage

Réduction du bruit: Mauvaise résolution, caractéristique de la caméra ...

Application d'un filtre non linéaire "Non-local means"

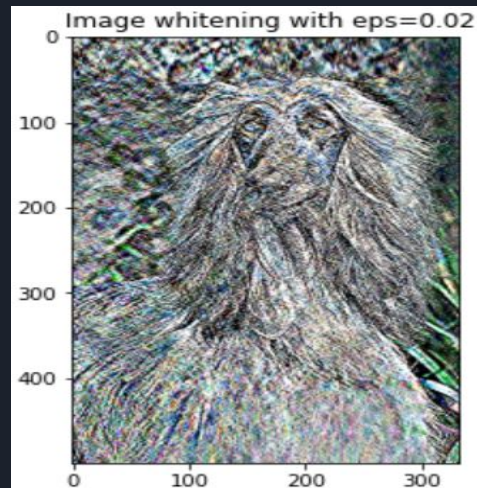
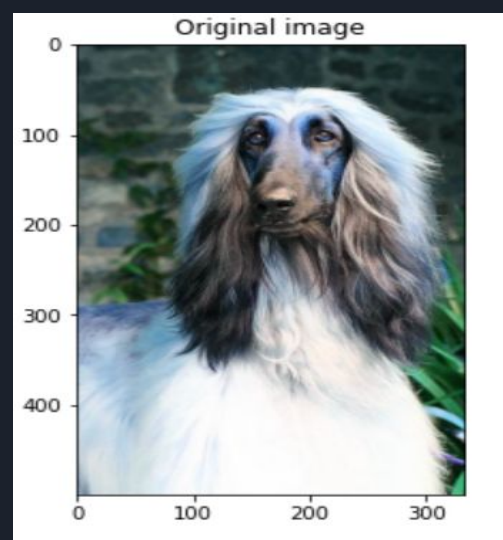
le filtre prend une moyenne de tous les pixels de l'image pondérée par la similarité de chaque cible par les autres pixels



Whitening

Le whitening “blanchiment” d’une image consiste à décorrélérer l’image

En supprimant la corrélation spatiale entre les pixels, le réseau peut converger plus rapidement que sans blanchiment.



Data augmentation

Création d'images par à partir d'une seule par des transformations géométriques:

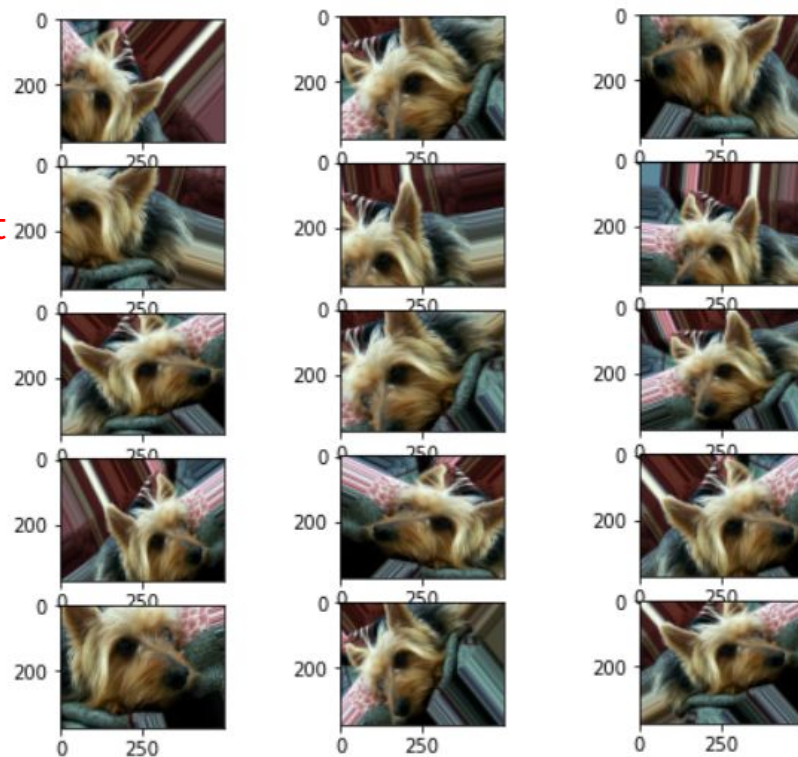
Rotation, décalage, zoom, cisaillement, retournement

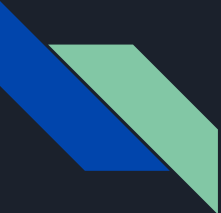
```
# Data generator on train set with Data Augmentation
```

```
train_datagen = ImageDataGenerator(  
    rescale=1./255,  
    rotation_range=40,  
    width_shift_range=0.2,  
    height_shift_range=0.2,  
    shear_range=0.2,  
    zoom_range=0.2,  
    horizontal_flip=True,  
    fill_mode='nearest',  
    validation_split=0.2)
```

```
#Rescale test set
```

```
test_datagen = ImageDataGenerator(rescale=1./255)
```





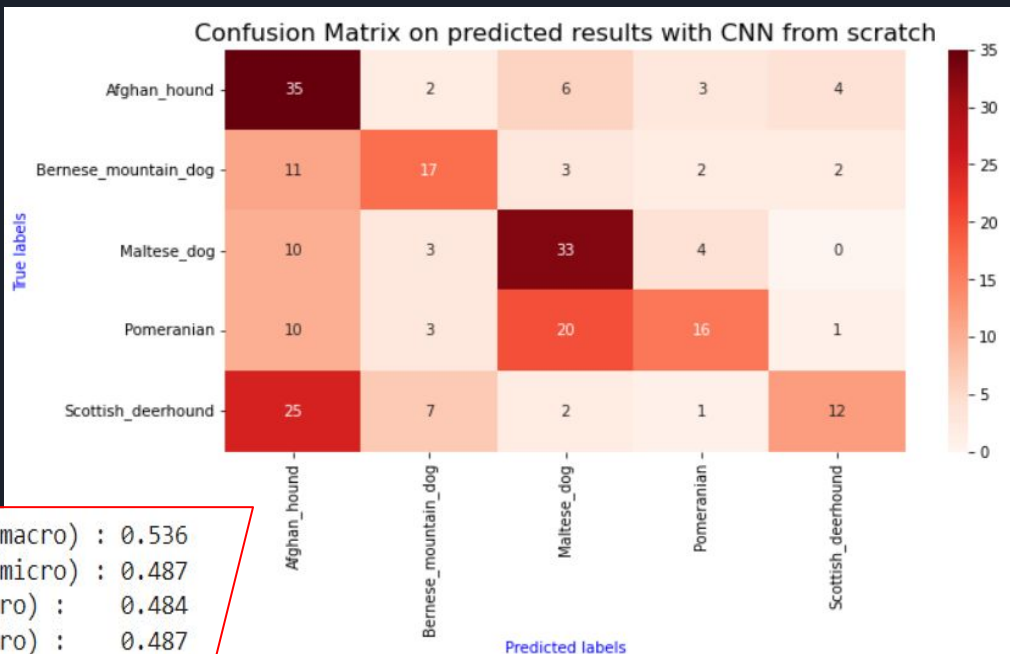
CNN from scratch



CNN from Scratch (nb_breeds =5)

Choix des hyperparamètres 1/1 suivant (loss<&accuracy>):

func d'activation, algo d'optimisation, nombre et taille de filtres, taille et pas des cellules, nb d'epochs.



Precision Score (macro) : 0.536
 Precision Score (micro) : 0.487
 Recall Score (macro) : 0.484
 Recall Score (micro) : 0.487

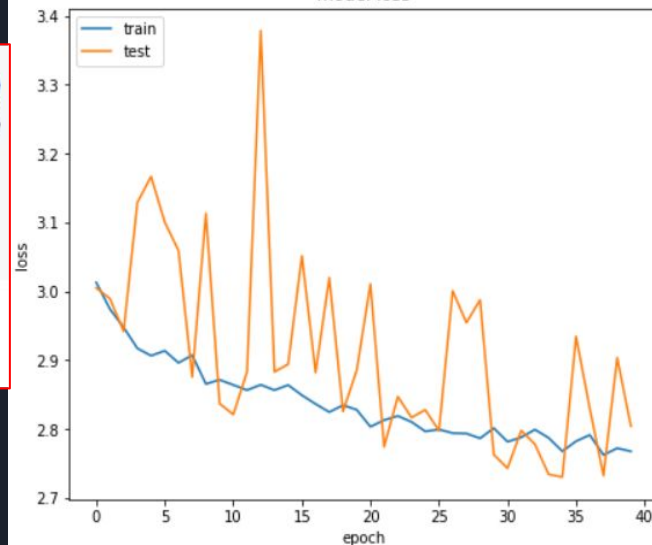
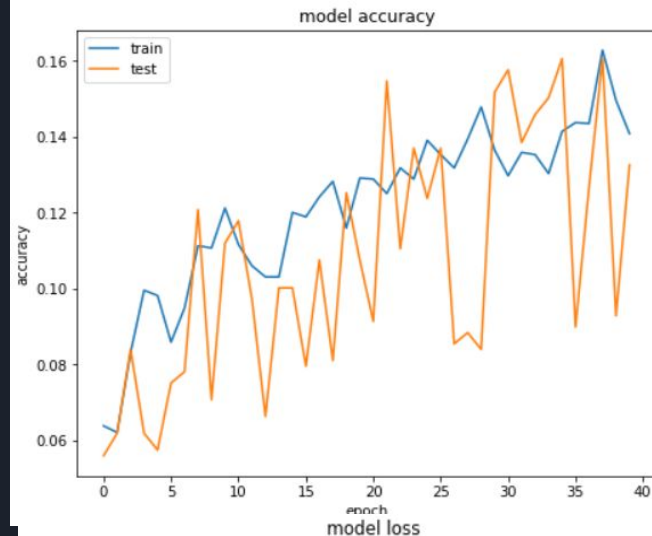
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 300, 300, 4)	196
batch_normalization (Batch Normalization)	(None, 300, 300, 4)	16
activation (Activation)	(None, 300, 300, 4)	0
max_pooling2d (MaxPooling2D)	(None, 150, 150, 4)	0
conv2d_1 (Conv2D)	(None, 150, 150, 8)	520
batch_normalization_1 (Batch Normalization)	(None, 150, 150, 8)	32
activation_1 (Activation)	(None, 150, 150, 8)	0
max_pooling2d_1 (MaxPooling2D)	(None, 75, 75, 8)	0
conv2d_2 (Conv2D)	(None, 75, 75, 16)	2064
batch_normalization_2 (Batch Normalization)	(None, 75, 75, 16)	64
activation_2 (Activation)	(None, 75, 75, 16)	0
max_pooling2d_2 (MaxPooling2D)	(None, 38, 38, 16)	0
conv2d_3 (Conv2D)	(None, 38, 38, 32)	8224
batch_normalization_3 (Batch Normalization)	(None, 38, 38, 32)	128
activation_3 (Activation)	(None, 38, 38, 32)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 32)	0
dropout (Dropout)	(None, 32)	0
dense (Dense)	(None, 64)	2112
dense_1 (Dense)	(None, 5)	325
Total params: 13,681		
Trainable params: 13,561		
Non-trainable params: 120		

CNN from Scratch (nb_breeds = 20)

Total params: 14,656
Trainable params: 14,536
Non-trainable params: 120

```
print("Precision Score (macro) : {:.3f}".format(precision_score(y_test, y_pred, average='macro', zero_division=1)))  
print("Precision Score (micro) : {:.3f}".format(precision_score(y_test, y_pred, average='micro', zero_division=1)))  
  
print("Recall Score (macro) : {:.3f}".format(recall_score(y_test, y_pred, average='macro')))  
print("Recall Score (micro) : {:.3f}".format(recall_score(y_test, y_pred, average='micro')))
```

Precision Score (macro) : 0.396
Precision Score (micro) : 0.135
Recall Score (macro) : 0.147
Recall Score (micro) : 0.135





Transfert learning modèle élu

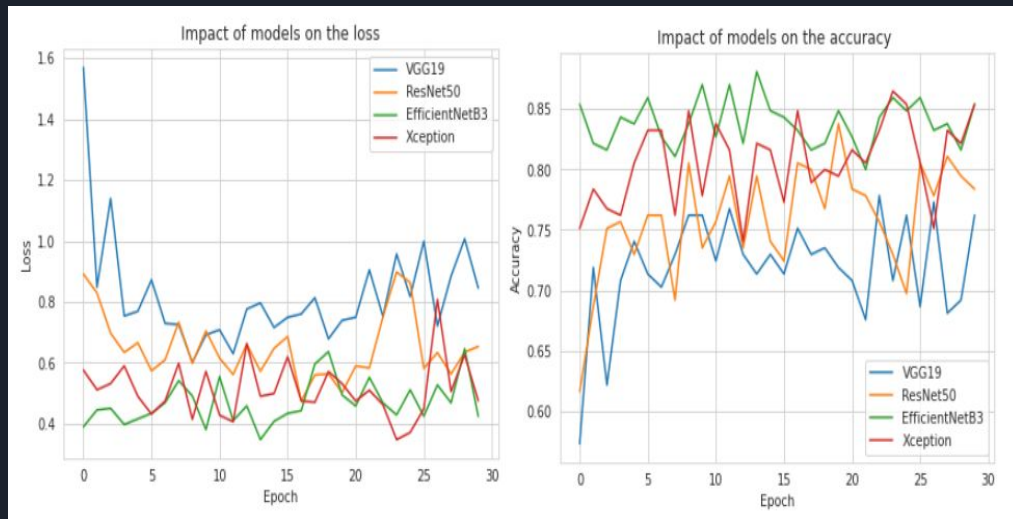


Transfert learning (extraction des features)

- Transfer learning:

utilisation d'un modèle pré-entraîné sans la couche dense et avec une congélation totale ou partielle des certaines couches

Comparaison de 4 modèles avec 20 races, VGG19, ResNet50, EffecientNetB3 et Xception



	time_of_fit (s)	accuracy_train	accuracy_val	accuracy_test	recall_test
VGG19	528.0	0.935345	0.762162	0.844828	0.84265
ResNet50	527.9	0.928879	0.783784	0.853448	0.846967
EfficientNetB3	507.9	0.971429	0.854054	0.922414	0.918334
Xception	479.5	0.914286	0.854054	0.900862	0.892243

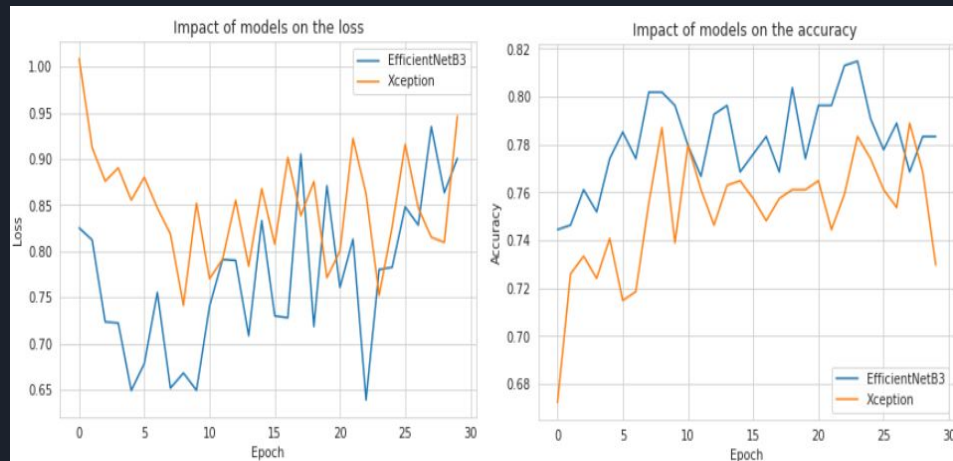
Transfert learning (20 races)

```
# Classification report
from sklearn.metrics import classification_report
# Inverse transform
test_classes = encoder.inverse_transform(y_test)
print(classification_report(y_test, predict["EfficientNetB3"],
                           target_names=sorted(set(test_classes))))
```

	precision	recall	f1-score	s	precision	recall	f1-score
African_hunting_dog	1.00	1.00	1.00		0.97	0.97	0.97
Appenzeller	0.88	0.96	0.92		0.75	1.00	0.86
Cardigan	0.93	0.87	0.90		0.90	0.90	0.90
Chesapeake_Bay_retriever	0.78	0.83	0.81		0.81	0.73	0.77
English_setter	0.85	0.85	0.85		0.80	0.80	0.80
German_short	0.80	0.70	0.74		0.73	0.70	0.71
Gordon_setter	0.83	0.88	0.85		0.82	0.85	0.84
Irish_terrier	0.81	0.83	0.82		0.72	0.66	0.69
Irish_wolfhound	0.80	0.77	0.78		0.86	0.71	0.78
Norfolk_terrier	0.91	0.89	0.90		0.78	0.81	0.79
Saluki	0.88	0.93	0.90		0.85	0.93	0.89
Samoyed	0.93	0.95	0.94		0.92	0.92	0.92
Shetland_sheepdog	0.81	0.96	0.88		0.89	0.93	0.91
bluetick	0.73	0.86	0.79		0.73	0.79	0.76
borzoi	0.84	0.76	0.80		0.70	0.76	0.73
flat	0.93	0.82	0.87		0.90	0.79	0.84
giant_schnauzer	0.97	1.00	0.98		1.00	0.90	0.95
groenendael	0.97	1.00	0.99		0.91	0.94	0.93
papillon	0.97	0.90	0.94		0.97	0.90	0.94
wire	0.96	0.81	0.88		0.74	0.91	0.82
accuracy			0.88				0.84
macro avg	0.88	0.88	0.88		0.84	0.84	0.84
weighted avg	0.88	0.88	0.88		0.85	0.84	0.84

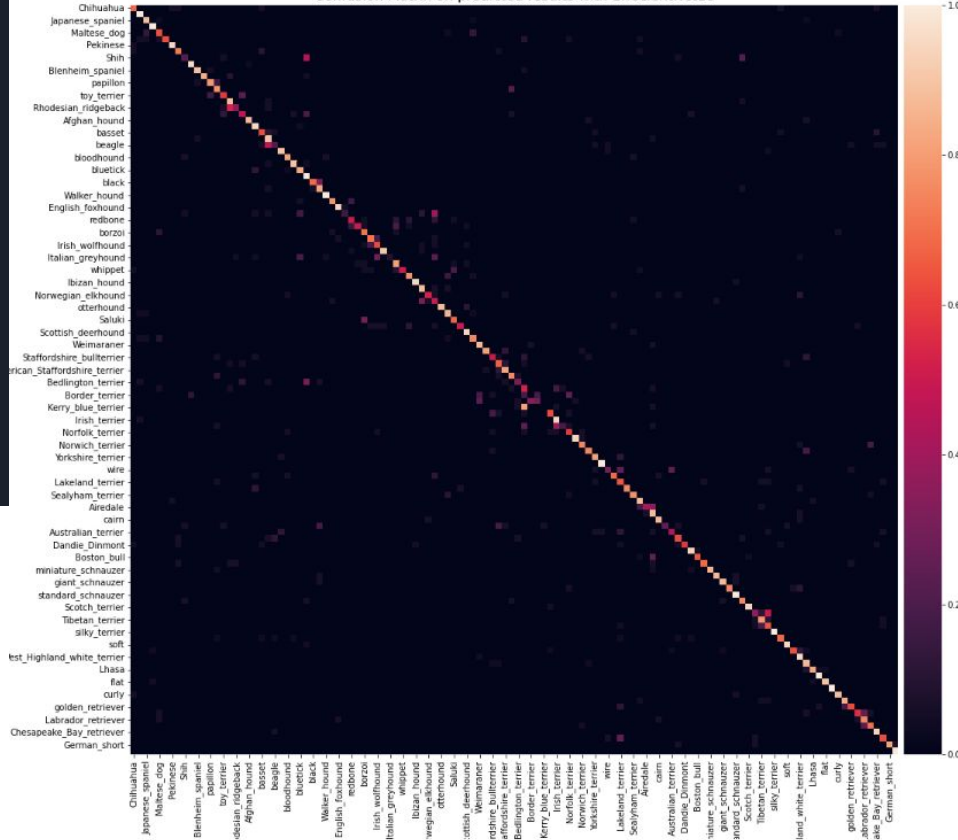
Xception scores

	time_of_fit (s)	accuracy_train	accuracy_val	accuracy_test	recall_test
EfficientNetB3	1324.8	0.949265	0.783333	0.881657	0.878953
Xception	1184.1	0.881618	0.72963	0.844675	0.844691



Modèle final

Confusion Matrix on predicted results with EffiecentNetB3



EffiecentNetB3 avec extraction des features

Train

15394 (75%)

Validation

3025 (15%)

Test

2161 (10%)

Preprocessing:

redimension (300,300,3) + normalisation (0,1)

Modèle final



Precision Score (macro) : 0.745
Precision Score (micro) : 0.725
Recall Score (macro) : 0.719
Recall Score (micro) : 0.725

Model: "sequential"

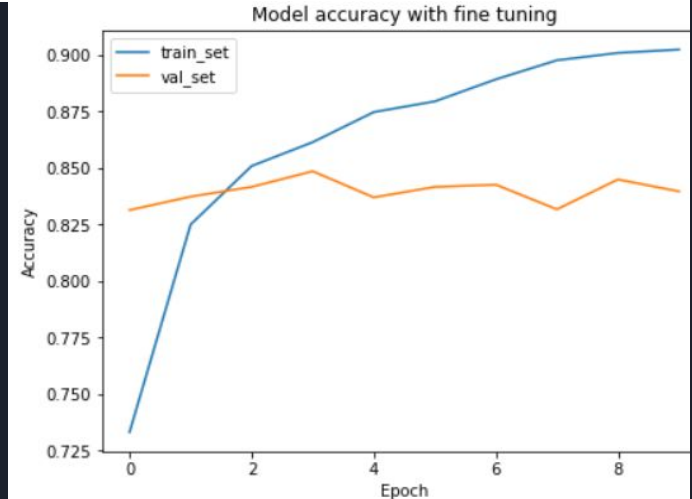
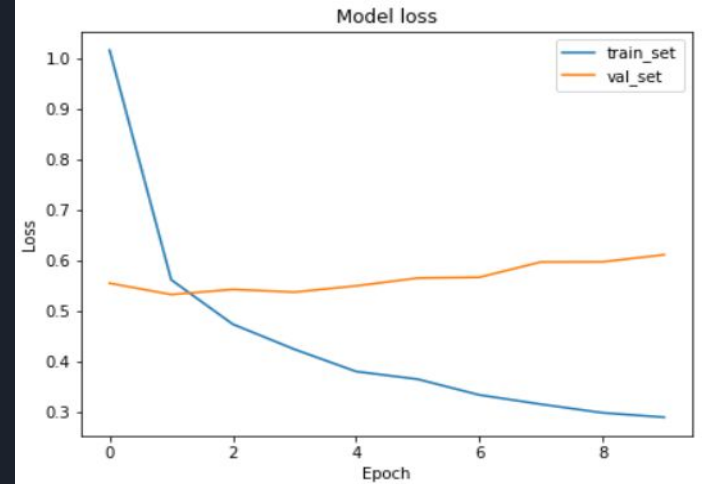
Layer (type)	Output Shape	Param #
efficientnetb3 (Functional)	(None, 10, 10, 1536)	10783535
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1536)	0
dense (Dense)	(None, 512)	786944
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 120)	61560

=====
Total params: 11,632,039
Trainable params: 848,504
Non-trainable params: 10,783,535
=====

```
# Save Tf_efficientnetb3 model
Tf_efficientnetb3.save('/content/drive/MyDrive/Saved_model/fine_tune_efficientnetb3.h5')

print("fine_tune_efficientnetb3 model")

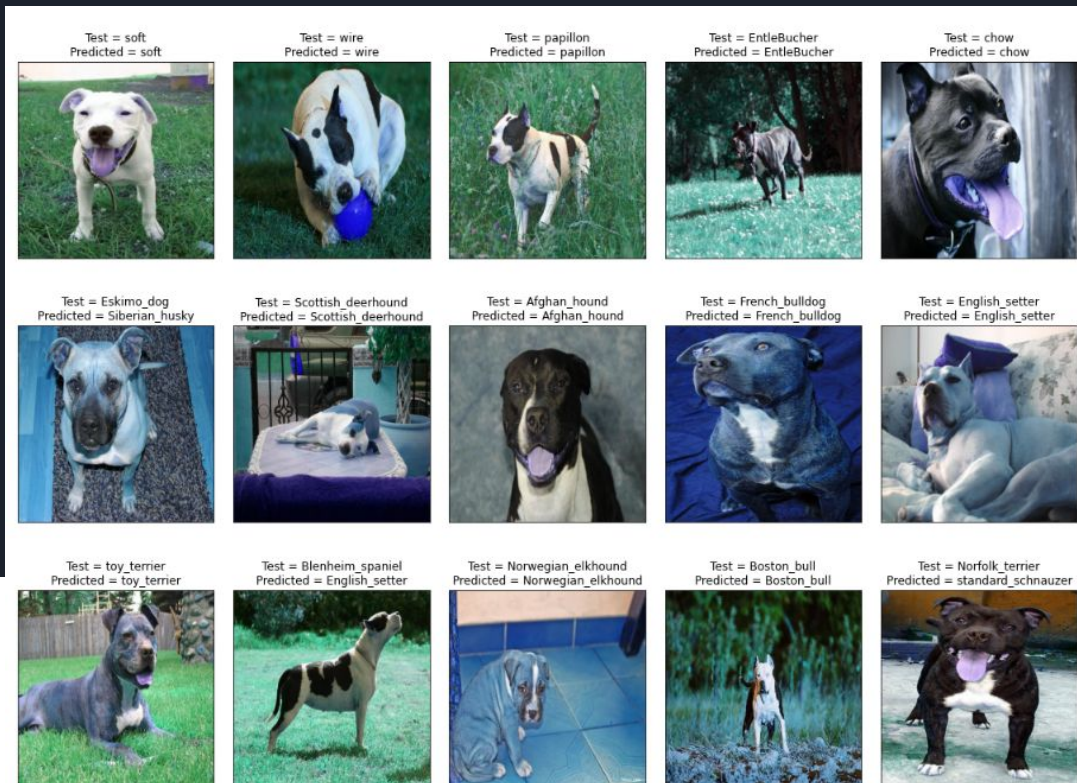
fine_tune_efficientnetb3 model
```




Merci de votre attention

La matrice de confusion présente bien les couples predic/true majoritaires sur la diagonale

une ou deux classes confondues en dehors de la diagonale, ça concerne des races avec des caractéristiques proches et semblables



Api avec gradio



Nettoyer

Soumettre


output

Appenzeller

Appenzeller	99%
EntleBucher	1%
basenji	0%
Rottweiler	0%
Greater_Swiss_Mountain_dog	0%

Signaler

Interpréter



Nettoyer

Soumettre


output

Sussex_spaniel

Sussex_spaniel	86%
cocker_spaniel	14%
Blenheim_spaniel	0%
clumber	0%
otterhound	0%

Signaler

Interpréter



Nettoyer

Soumettre

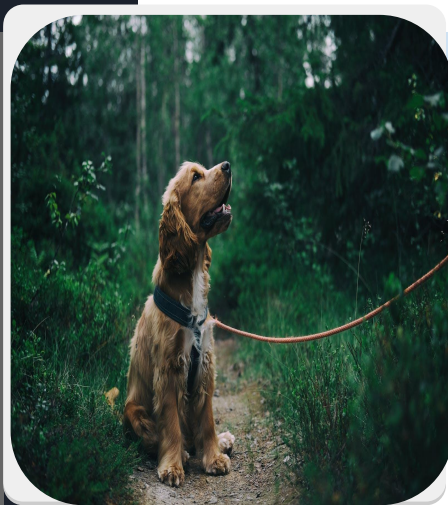
output

Great_Pyrenees

Great_Pyrenees	77%
kuvasz	23%
Tibetan_mastiff	1%
Newfoundland	0%
komondor	0%

Conclusion

- Compétences acquises (traitement d'image, deep learning)
- Difficultés techniques (GPU)
- CNN initial optimisé sur 5 races
- Transfert learning et comparaison avec des modèle pré-entraînés
- Sur-apprentissage léger
- erreur de prédiction pour des races assez ressemblantes



Perspectives:

- ❑ CNN + modèle de classification de ML (RF)
- ❑ Transfert learning avec des images aux bounding boxes



Merci de votre attention

