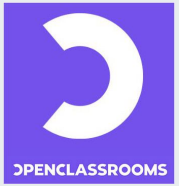
Classez des images à l'aide d'algorithmes de Deep Learning

Openclassroom

29 août 2022



Présentation du projet :

Objectif :

Développer un algorithme capable de classer les images en fonction de la race du chien présent sur l'image.

Cet objectif peut se rapprocher d'une tâche de classification multi class supervisée.

Données :

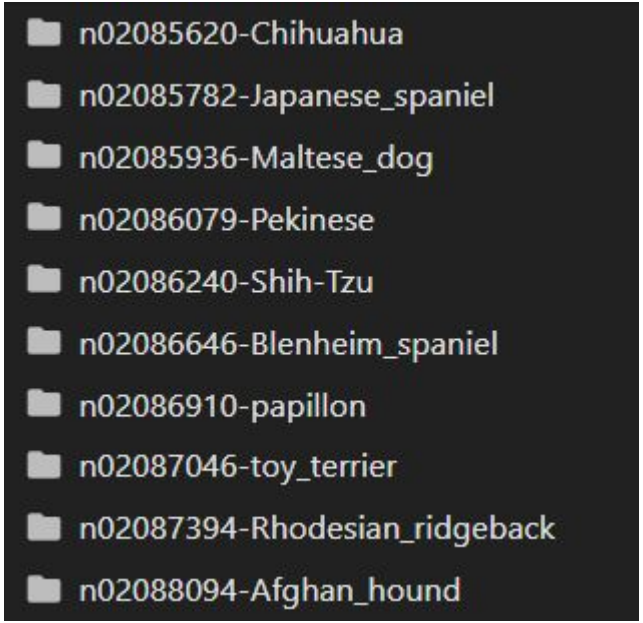
Utilisation de données présentes sur Vision Stanford.

Comprenant les images de chien classé par race dans différents dossiers.

Données

Il y a 120 races représentées avec un total de 20 430 d'images.

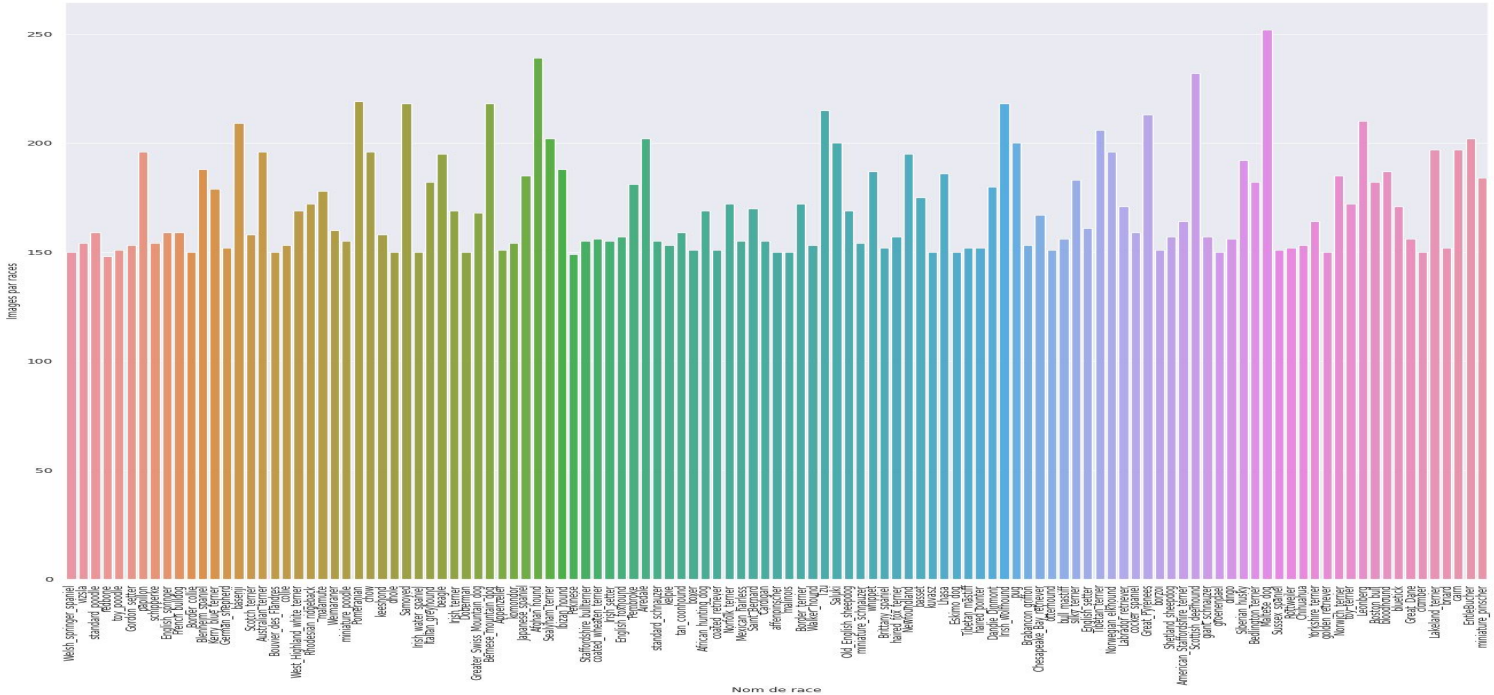
Chaque dossier correspond à une race de chien particulière avec un ID unique.



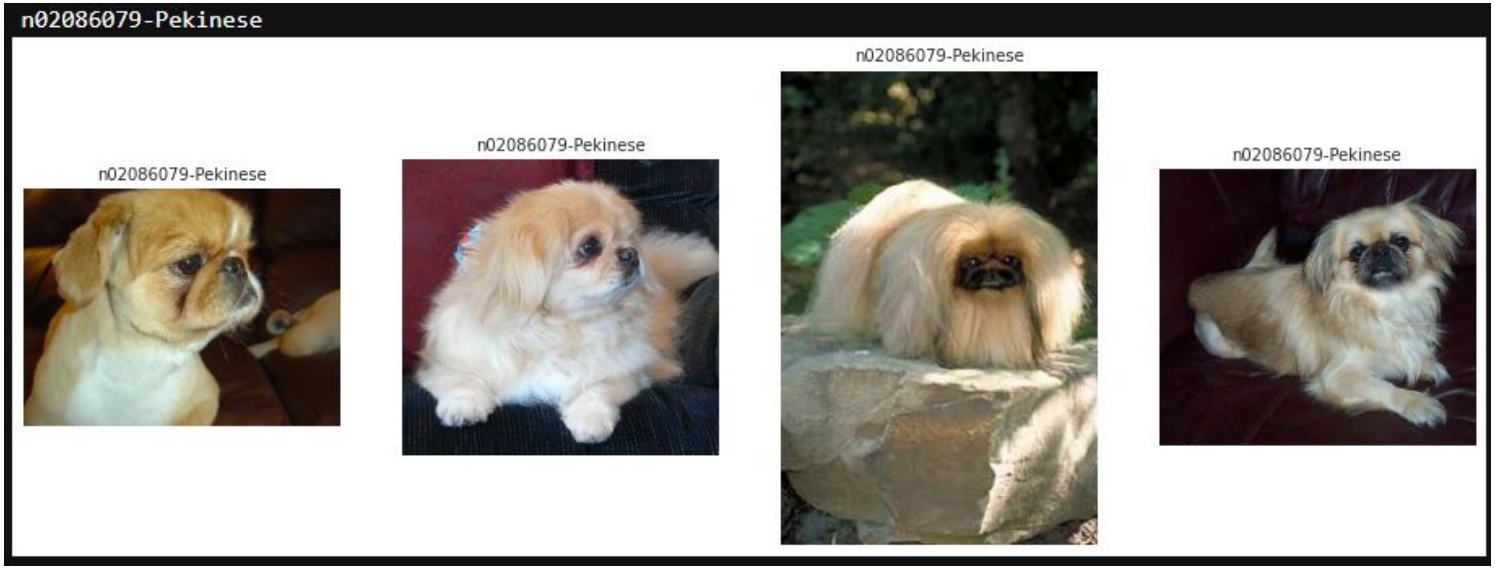
Analyse exploratoire

Visualisation univariée

Nombre d'image par race de chien



Exemple d’image



Preprocessing

normalisation, sélection de donnée et encodage

cv2, encoder

Normalisation, selection and Encoding

```
le = LabelEncoder()

def get_training(nb_class):
    X = list()
    Z = list()

    def make_train_data(breed_name,DIR):
        for img in tqdm(os.listdir(DIR)):
            path = os.path.join(DIR,img)
            img = cv2.imread(path, cv2.IMREAD_COLOR)
            img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
            X.append(np.array(img))
            Z.append(str(breed_name))

    for breed in breed_list[:nb_class]:
        name = breed.split('-')[-1]
        img_dir = f"{IMAGES_DIR}/{breed}/"
        make_train_data(name, img_dir)

    # converting labels to categorical data
    Y = le.fit_transform(Z)
    Y = to_categorical(Y, nb_class)

    # Normalising X
    X = np.array(X) #.reshape(-1, IMG_SIZE, IMG_SIZE, 3)
    X = X.astype('float32')
    X = X/255

    return X, Y
```

Data augmentation

Rotation, Equalization, whitening

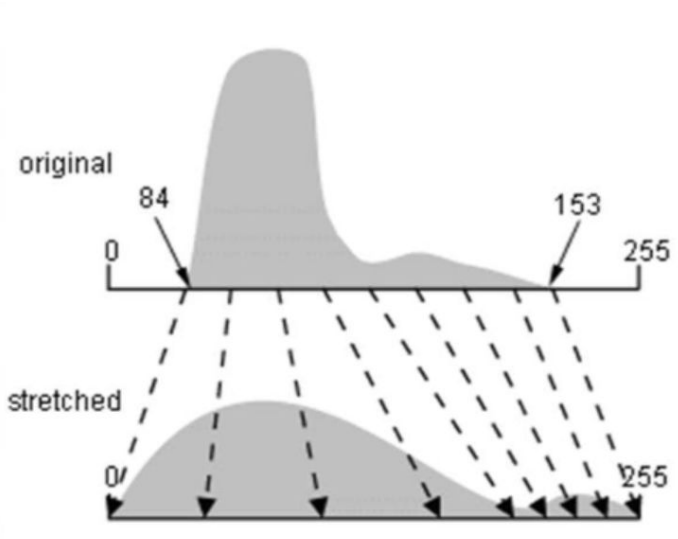
ImageDataGenerator, exposure

ImageDataGenerator

```
datagen = ImageDataGenerator(
        featurewise_center=False,  # set input mean to 0 over the dataset
        samplewise_center=False,  # set each sample mean to 0
        featurewise_std_normalization=False,  # divide inputs by std of the dataset
        samplewise_std_normalization=False,  # divide each input by its std
        zca_whitening=False,  # apply ZCA whitening
        rotation_range=10,  # randomly rotate images in the range (degrees, 0 to 180)
        zoom_range = 0.1, # Randomly zoom image
        width_shift_range=0.2,  # randomly shift images horizontally (fraction of total width)
        height_shift_range=0.2,  # randomly shift images vertically (fraction of total height)
        horizontal_flip=True,  # randomly flip images
        vertical_flip=False)  # randomly flip images

datagen.fit(x_train)
```

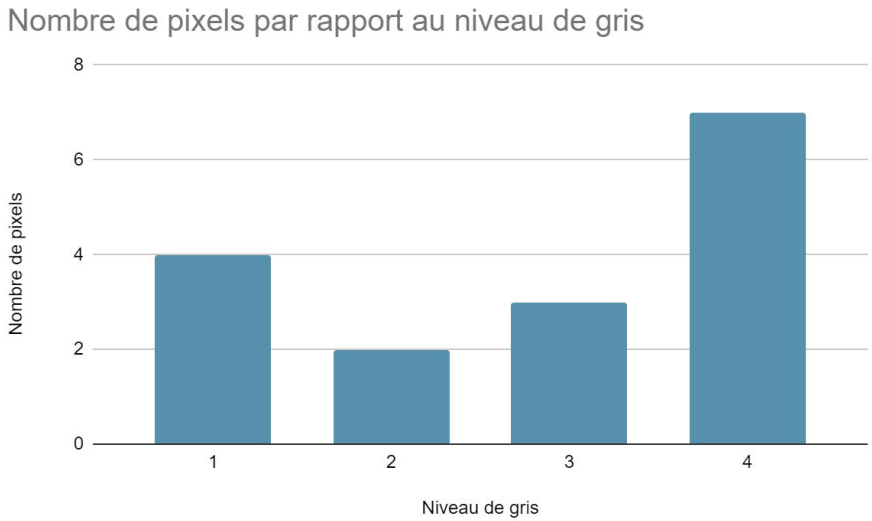
Expansion de contraste (contrast stretching)

L'expansion de contraste est une méthode data augmentation qui permet d'améliorer le contraste d'une image en étendant l'intervalle d'intensité des valeurs qu'il contient.



Création histogramme

Histogramme par niveau de gris d'une image



Valeur niveau de gris image 4x4

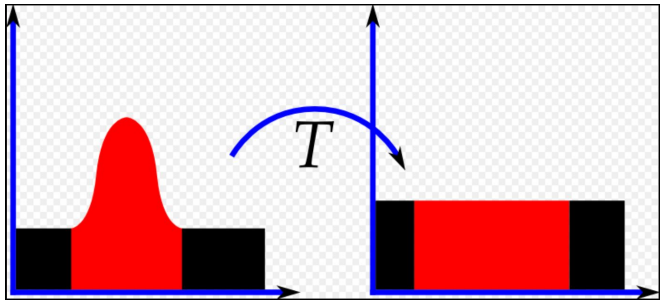
1	2	3	4
1	2	4	4
1	3	4	4
1	3	4	4

Niveau de gris	1	2	3	4
Nombre de pixels	4	2	3	7

Égalisation d'histogramme

Proportion de niveau de gris par rapport à l'histogramme cumulé

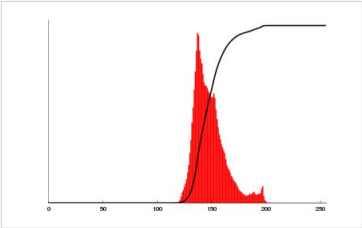
$$\forall v \in [0..n], eg(v) = \frac{V_{max} - V_{min}}{N} C_f(v) + V_{min}$$



La transformation T permet d'égaliser l'histogramme et d'améliorer le contraste.

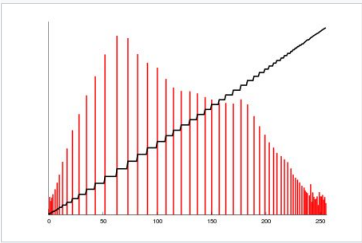
Source : Wikipédia

image originale



Son histogramme (rouge) et l'histogramme cumulé (noir)

L'image après égalisation d'histogramme

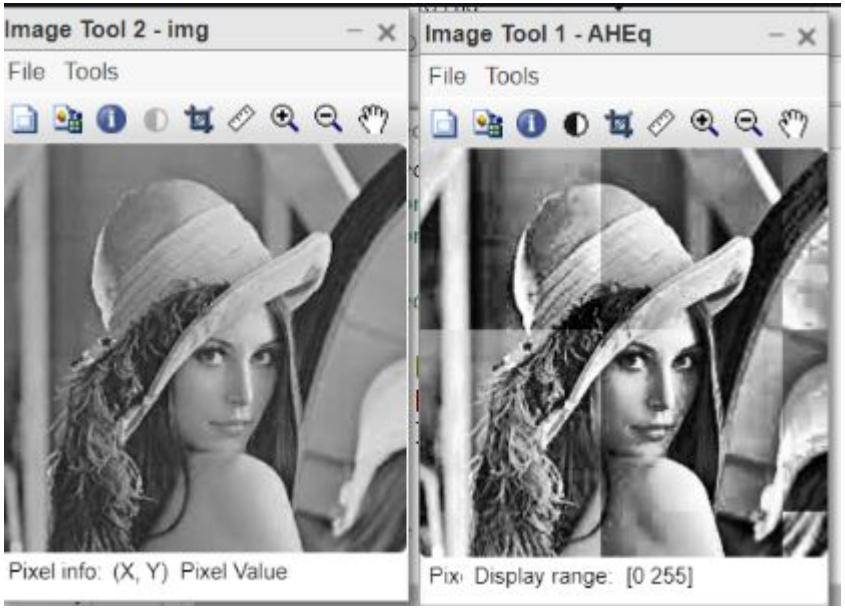


L'histogramme (rouge) et l'histogramme cumulé (noir) après égalisation

Source : Wikipédia

Égalisation adaptative d'histogramme

Le principe de cette méthode est d'appliquer une égalisation d'histogramme, sur des régions, ce qui donne du contexte à l'application.



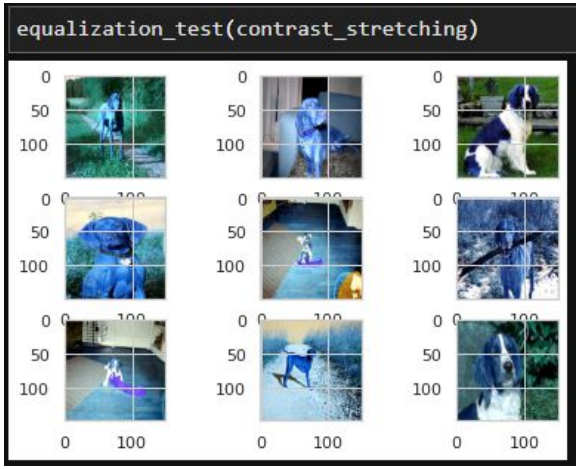
Égalisation avec ImageDataGenerator

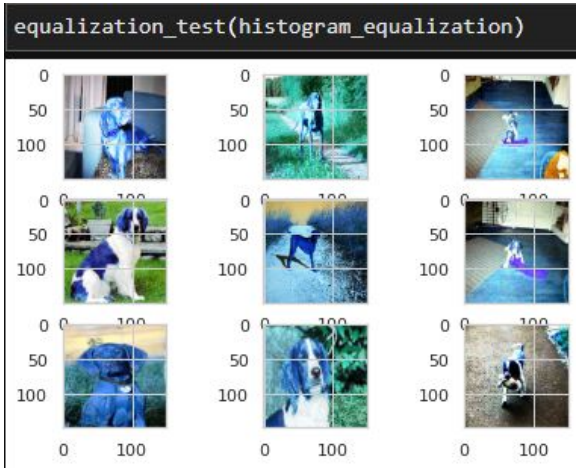
Création de 3 méthodes personnalisées pour ImageDataGenerator

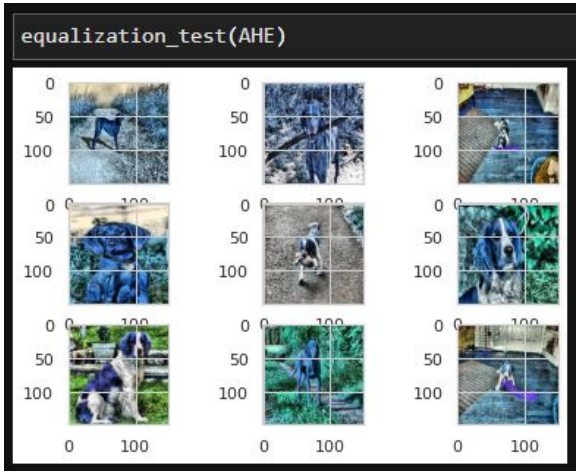
```python
def AHE(img):
    img_adapteq = exposure.equalize_adapthist(img, clip_limit=0.03)
    return img_adapteq

def histogram_equalization(img):
    img_adapteq = exposure.equalize_hist(img) #####
    return img_adapteq

def contrast_stretching(img):
    p2, p98 = np.percentile(img, (2, 98)) #####
    img_adapteq = exposure.rescale_intensity(img, in_range=(p2, p98)) #####
    return img_adapteq
```

```python
def equalization_test(fct):
    datagen_equalize = ImageDataGenerator(preprocessing_function=fct)
    x_ex, y_ex = x_train[:10], y_train[:10]
    # fit parameters from data
    datagen_equalize.fit(x_ex)
    img_rows, img_cols = 150,150
    # Configure batch size and retrieve one batch of images
    for X_batch, y_batch in datagen_equalize.flow(x_ex, y_ex, batch_size=9):
        # Show 9 images
        for i in range(0, 9):
            plt.subplot(330 + 1 + i)
            plt.imshow(X_batch[i].reshape(img_rows, img_cols, 3))
        # show the plot
        plt.show()
        break
```

Exemple d'application :







Modèle CNN

Utilisation de méthode de Convolutional Neural Network

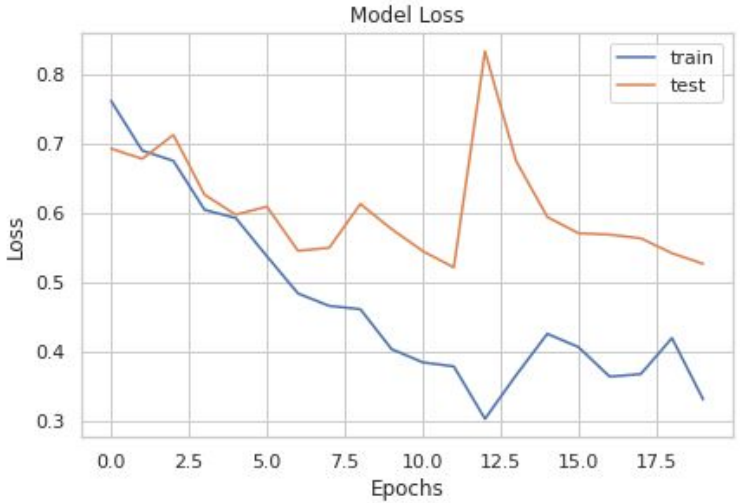
Implémentation faite avec Keras

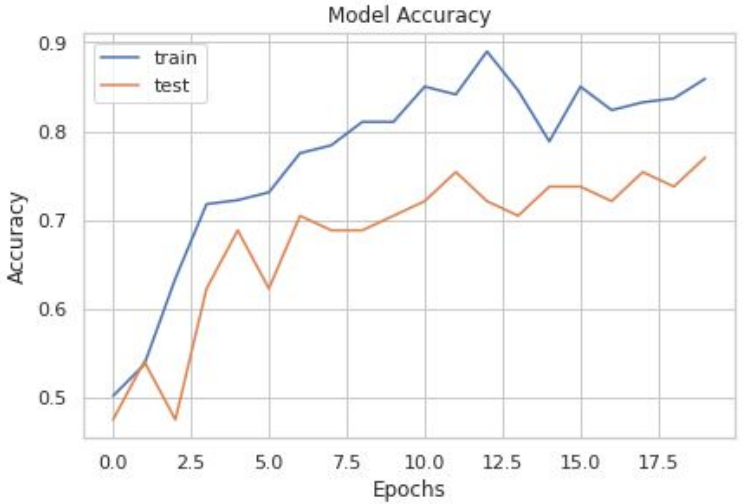
Présentation du modèle

```
model = Sequential()
model.add(Conv2D(filters = 32, kernel_size = (5,5), padding = 'Same', activation ='relu', input_shape = (IMG_SIZE, IMG_SIZE,3)))
model.add(MaxPooling2D(pool_size=(2,2)))


model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))


model.add(Conv2D(filters =96, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Conv2D(filters = 96, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dense(nb_class, activation = "softmax"))
```

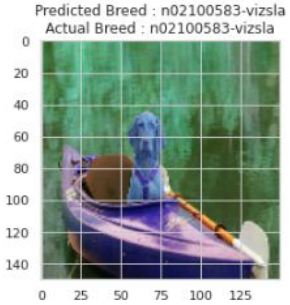
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 150, 150, 32)      2432

 max_pooling2d (MaxPooling2D  (None, 75, 75, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 75, 75, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 37, 37, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 37, 37, 96)        55392

 max_pooling2d_2 (MaxPooling  (None, 18, 18, 96)       0
 2D)

 conv2d_3 (Conv2D)           (None, 18, 18, 96)        83040

 max_pooling2d_3 (MaxPooling  (None, 9, 9, 96)         0
 2D)

 flatten (Flatten)           (None, 7776)              0

 dense (Dense)               (None, 512)               3981824

 activation (Activation)     (None, 512)               0

 dense_1 (Dense)             (None, 2)                 1026

=================================================================
Total params: 4,142,210
Trainable params: 4,142,210
Non-trainable params: 0
_________________________________________________________________
```

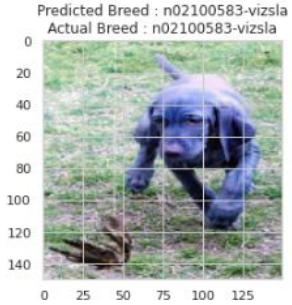
Evaluation modèle

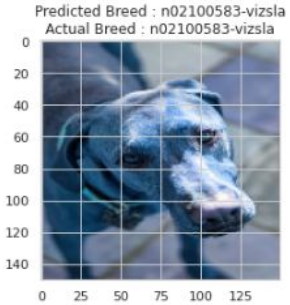


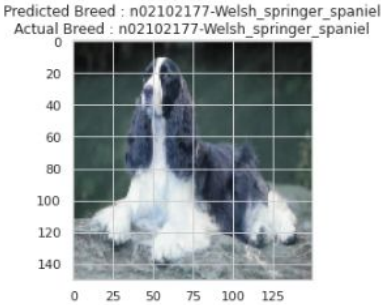


Test du modèle









Transfer Learning

Gagner en ressource et efficacité

Modèle DenseNet

Utilisation de modèle pré entraîné de ImageNet

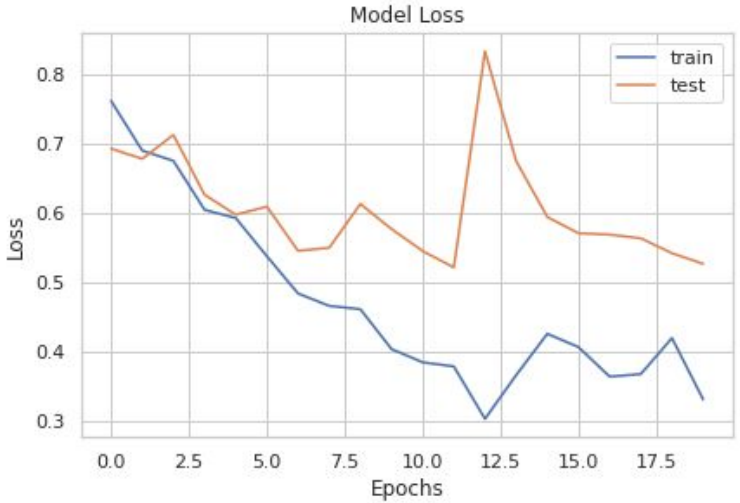
Sequential

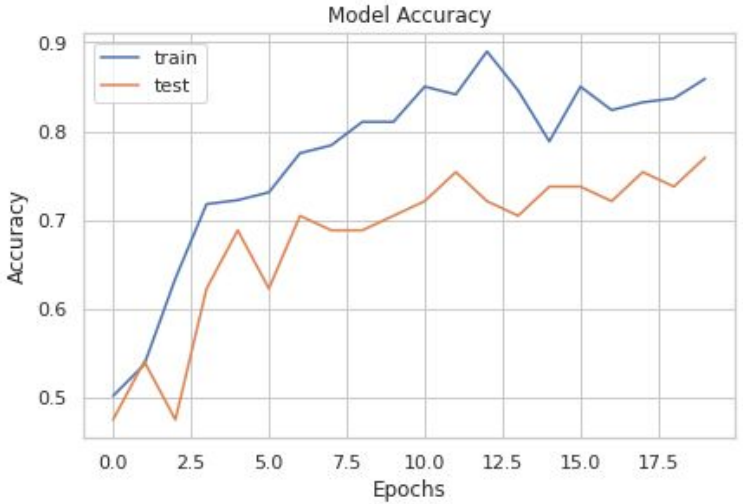
Présentation du modèle

Explication rapide de DenseNEt

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 150, 150, 32)      2432

 max_pooling2d (MaxPooling2D  (None, 75, 75, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 75, 75, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 37, 37, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 37, 37, 96)        55392

 max_pooling2d_2 (MaxPooling  (None, 18, 18, 96)       0
 2D)

 conv2d_3 (Conv2D)           (None, 18, 18, 96)        83040

 max_pooling2d_3 (MaxPooling  (None, 9, 9, 96)         0
 2D)

 flatten (Flatten)           (None, 7776)              0

 dense (Dense)               (None, 512)               3981824

 activation (Activation)     (None, 512)               0

 dense_1 (Dense)             (None, 2)                 1026

=================================================================
Total params: 4,142,210
Trainable params: 4,142,210
Non-trainable params: 0
_________________________________________________________________
```

Evaluation modèle





Présentation application Flask