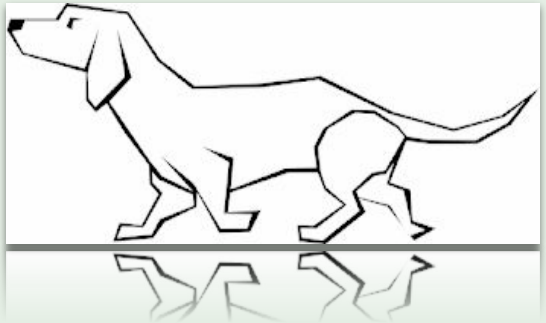
Projet n° 6 : Classez des images à l'aide d'algorithmes de deep learning



01

Introduction

Présentation du problème, présentation et analyses des données.

02

CNN from scratch

Création de CNN pour la classification, optimisation des hyperparamètres

03

Transfer learning

Utilisation de réseaux de neurones pré-entraînés

04

Conclusion

Choix du modèle

01

Introduction

Présentation de la problématique

- Pour aider une association de protection des animaux on souhaite développer un modèle qui permet de classifier des images de chien en fonction de leur race.
- Les données sont récupérées du Stanford Dogs dataset : Il s'agit d'une base de données de photos de chiens de 120 races, construite à partir d'ImageNet.

Quelques Analyses

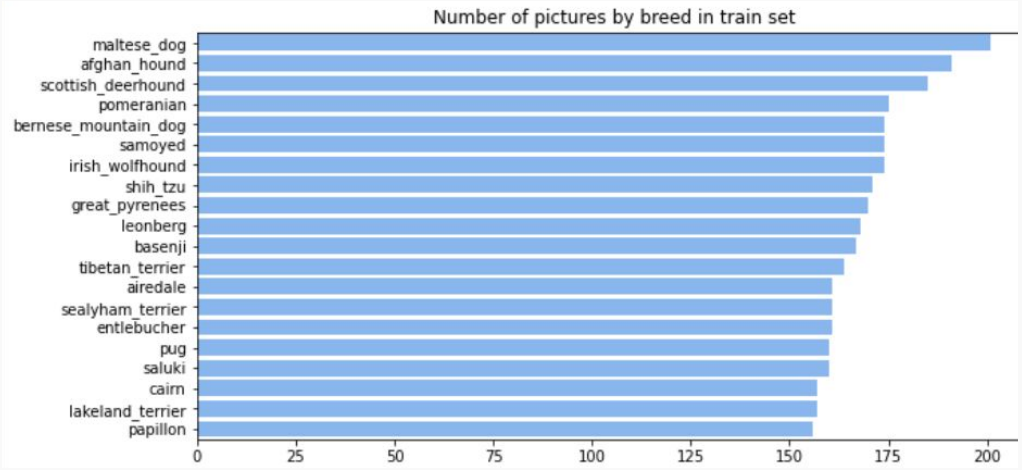
- En moyenne 171 images par races
- Pour des raisons de capacité de mémoire, on se limite aux 20 races les plus présentes.

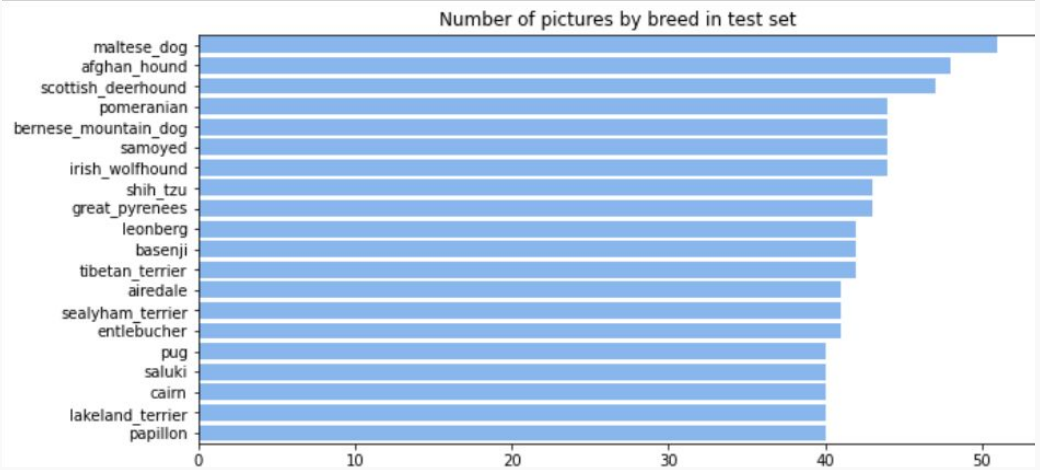
Train-Test split

On crée :

- Un dossier Train qui contient 3 387 images (80% du jeu de données)

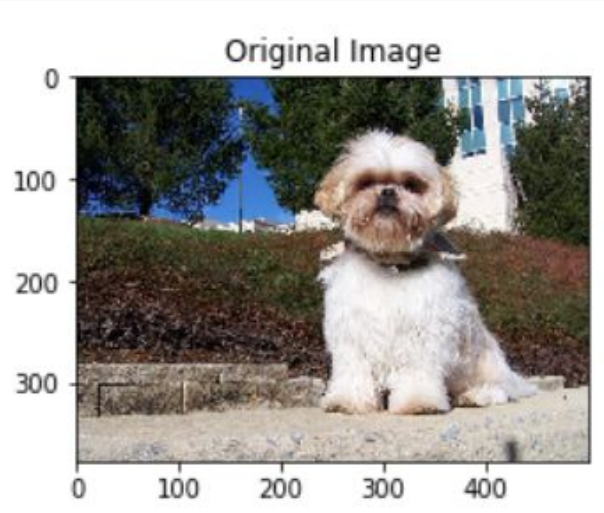
- Un dossier Test qui contient 857 images (20% du jeu de données)



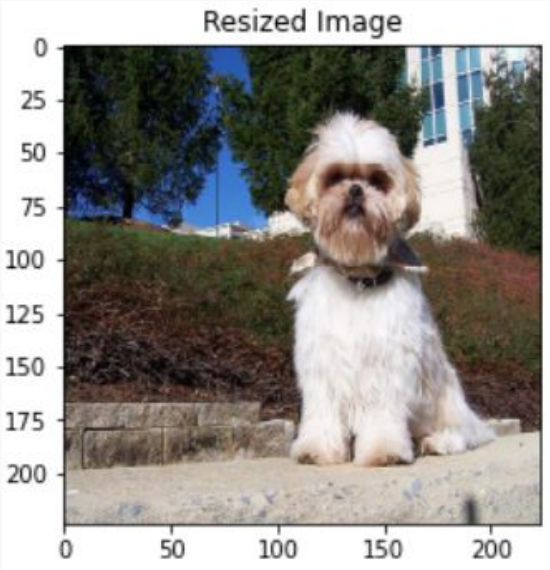


Redimensionnement des images

Premier pre-processing : redimensionnement des images



Dimensions format RGB : `(375, 500, 3)`

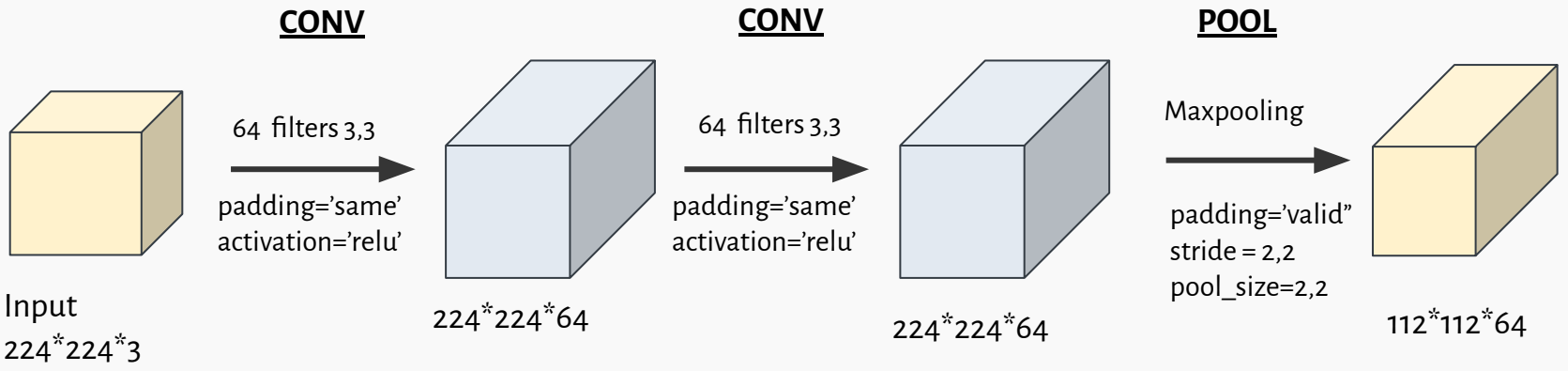


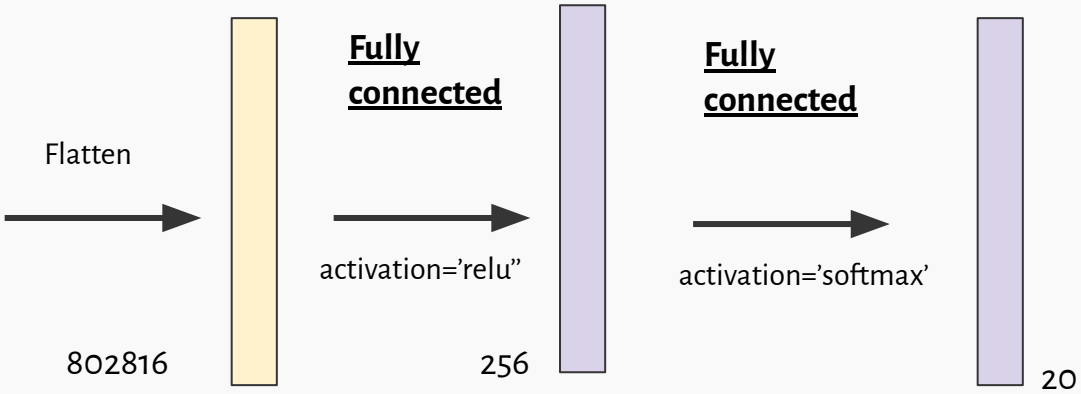
Dimensions format RGB : `(224, 224, 3)`

02

CNN from scratch

Premier CNN


CONV
CONV
POOL
64 filters 3,3
padding='same'
activation='relu'
64 filters 3,3
padding='same'
activation='relu'
Maxpooling
padding='valid"
stride = 2,2
pool_size=2,2
Input
224*224*3
224*224*64
224*224*64
112*112*64


Flatten
Fully connected
Fully connected
activation='relu"
activation='softmax'
802816
256
20

Résultats du premier CNN:

Accuracy

99 % sur le jeu d'entraînement

11 % sur le jeu de test

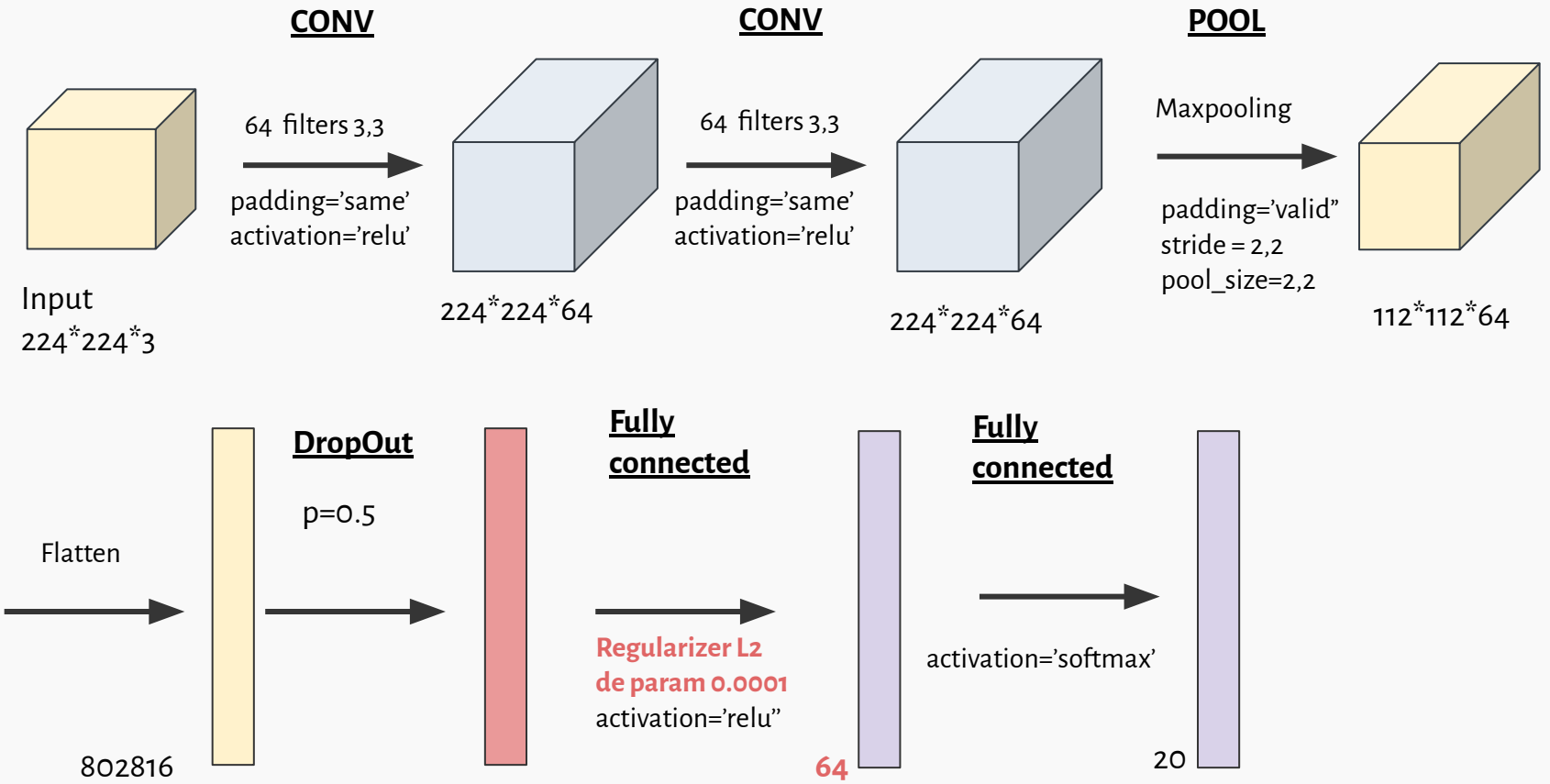
Overfitting

```
Epoch 1/10
106/106 [==============================] - 20s 98ms/step - loss: 5.2618 - accuracy: 0.0655
Epoch 2/10
106/106 [==============================] - 10s 94ms/step - loss: 2.8239 - accuracy: 0.1281
Epoch 3/10
106/106 [==============================] - 10s 98ms/step - loss: 2.2238 - accuracy: 0.3407
Epoch 4/10
106/106 [==============================] - 10s 91ms/step - loss: 0.6411 - accuracy: 0.8326
Epoch 5/10
106/106 [==============================] - 10s 92ms/step - loss: 0.1265 - accuracy: 0.9844
Epoch 6/10
106/106 [==============================] - 10s 91ms/step - loss: 0.0340 - accuracy: 0.9973
Epoch 7/10
106/106 [==============================] - 10s 91ms/step - loss: 0.0232 - accuracy: 0.9991
Epoch 8/10
106/106 [==============================] - 10s 91ms/step - loss: 0.0224 - accuracy: 0.9991
Epoch 9/10
106/106 [==============================] - 10s 93ms/step - loss: 0.0158 - accuracy: 0.9994
Epoch 10/10
106/106 [==============================] - 10s 91ms/step - loss: 0.0412 - accuracy: 0.9968
```

```
[ ]  my_VGG16.evaluate(X_test, to_categorical(y_test))

     27/27 [==============================] - 2s 47ms/step - loss: 6.4583 - accuracy: 0.1167
     [6.458297252655029, 0.11668611317873001]
```

Limitation de l'overfitting



Résultats avec DropOut + regularizers + diminution unités de sortie FC1

```
Epoch 1/10
106/106 [==============================] - 20s 91ms/step - loss: 4.0735 - accuracy: 0.0644
Epoch 2/10
106/106 [==============================] - 9s 83ms/step - loss: 3.0041 - accuracy: 0.1110
Epoch 3/10
106/106 [==============================] - 9s 85ms/step - loss: 2.4909 - accuracy: 0.3153
Epoch 4/10
106/106 [==============================] - 9s 85ms/step - loss: 1.3110 - accuracy: 0.6823
Epoch 5/10
106/106 [==============================] - 9s 83ms/step - loss: 0.6210 - accuracy: 0.9153
Epoch 6/10
106/106 [==============================] - 9s 83ms/step - loss: 0.4257 - accuracy: 0.9720
Epoch 7/10
106/106 [==============================] - 9s 86ms/step - loss: 0.3926 - accuracy: 0.9826
Epoch 8/10
106/106 [==============================] - 9s 83ms/step - loss: 0.3526 - accuracy: 0.9920
Epoch 9/10
106/106 [==============================] - 9s 83ms/step - loss: 0.3595 - accuracy: 0.9900
Epoch 10/10
106/106 [==============================] - 9s 84ms/step - loss: 0.4289 - accuracy: 0.9728
```

```
[ ] print('Time : {} s'.format(t2-t1))

    Time : 105.04740619659424 s

[ ] my_VGG16.evaluate(X_test, to_categorical(y_test))

    27/27 [==============================] - 2s 48ms/step - loss: 7.7096 - accuracy: 0.1459
    [7.7096171379089355, 0.14585764706134796]
```

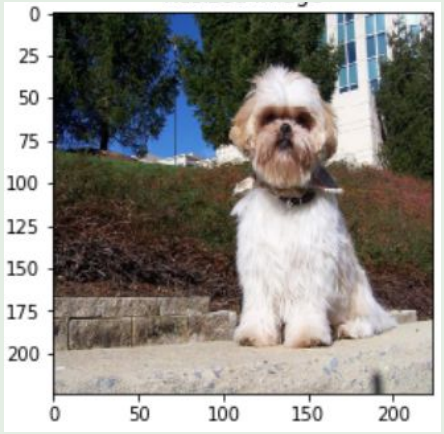
Data Augmentation

Principe : Générer davantage de données d'entraînement en transformant nos images afin d'améliorer notre modèle.

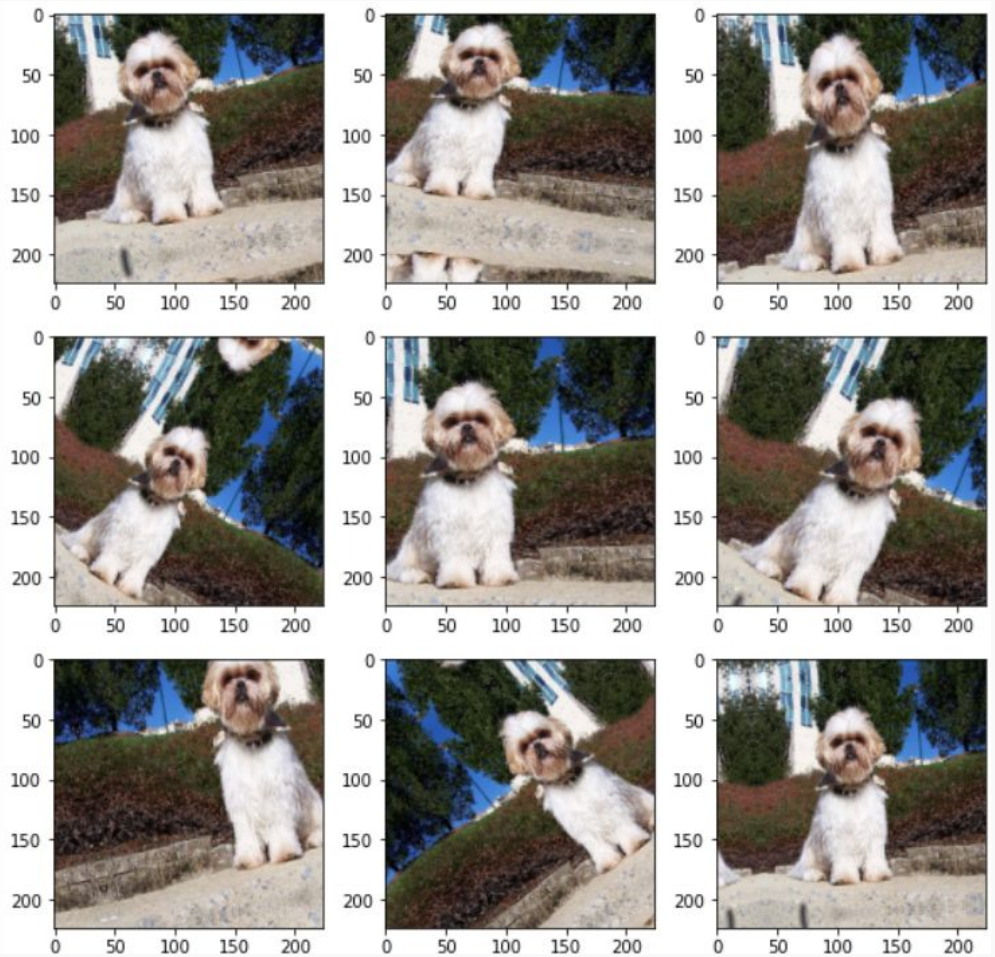
Transformations courantes:

- le shifting : décaler l'image en hauteur ou largeur
- le flipping : créer une image symétrique
- la rotation
- la modification de luminosité
- le zoom

Data Augmentation



```
ImageDataGenerator(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='reflect')
```



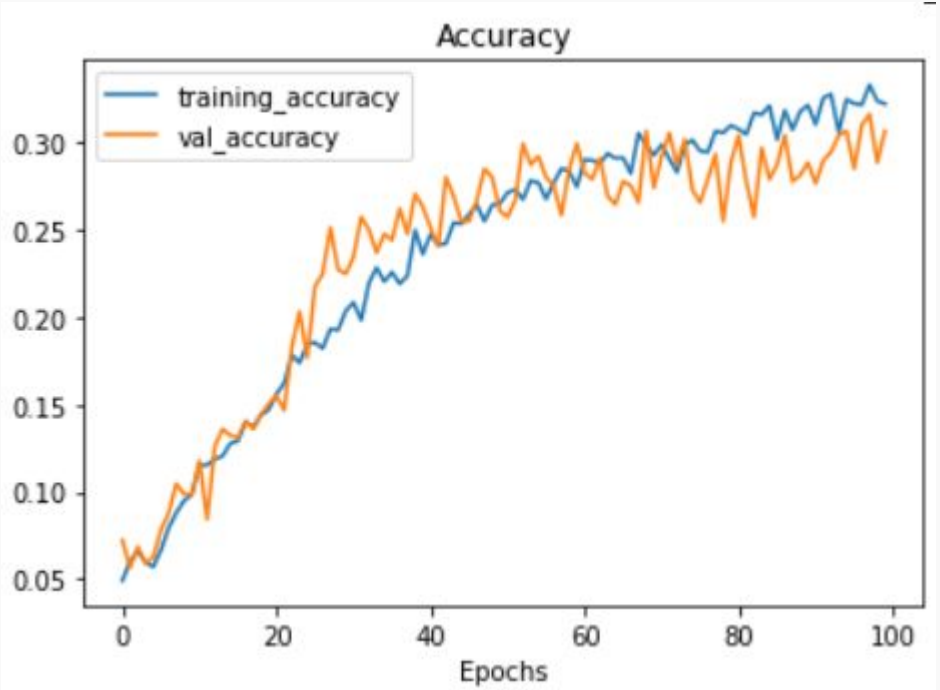
D’avantage de pre-preprocessing

- Rescale : recentrer les valeurs des pixels entre -1 et 1 (division des valeurs d’entrée par 255)

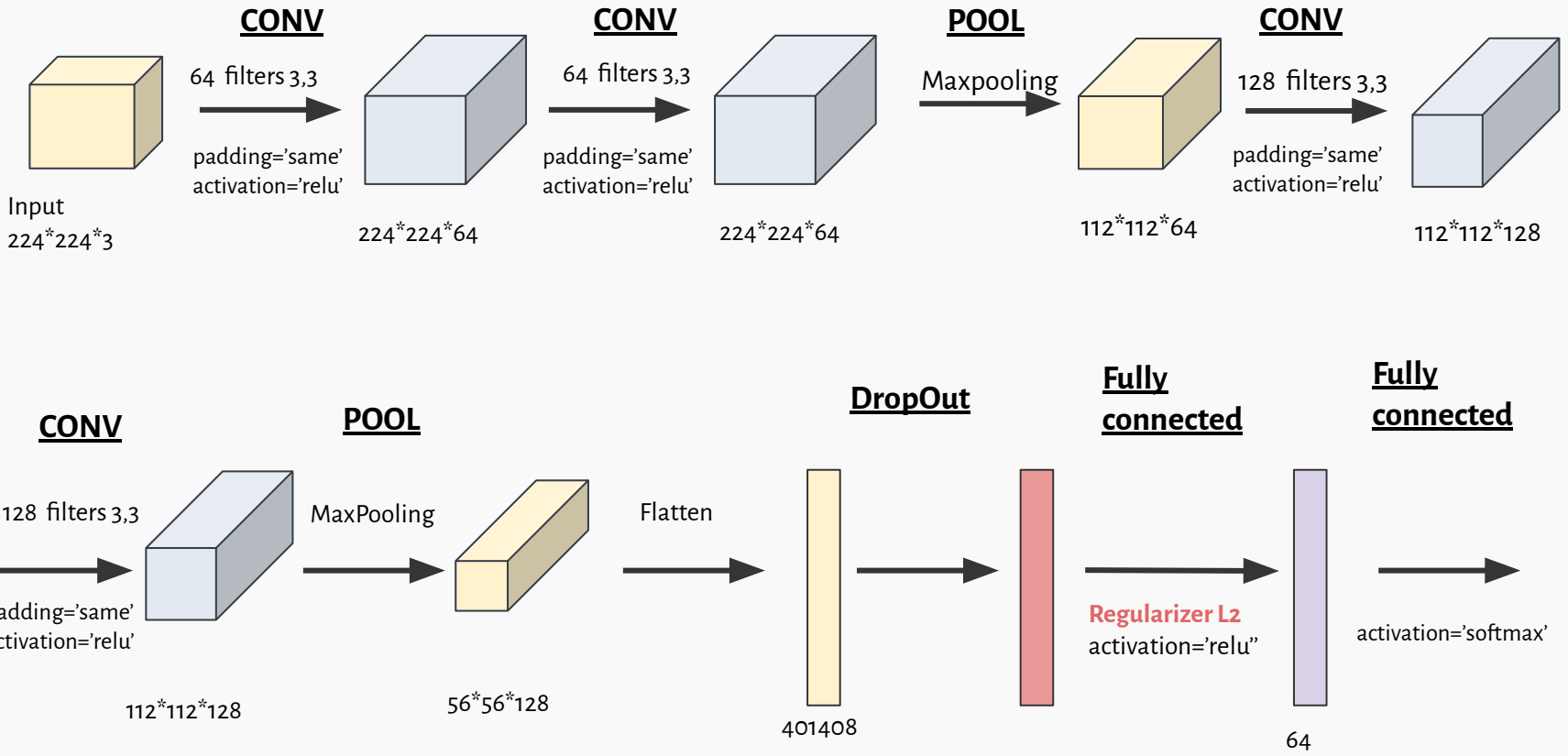
- zca_whitening : Technique de réduction de dimension pour les images. Elle se base sur le fait que des pixels adjacents ont des valeurs redondantes.

Résultats 1er CNN avec

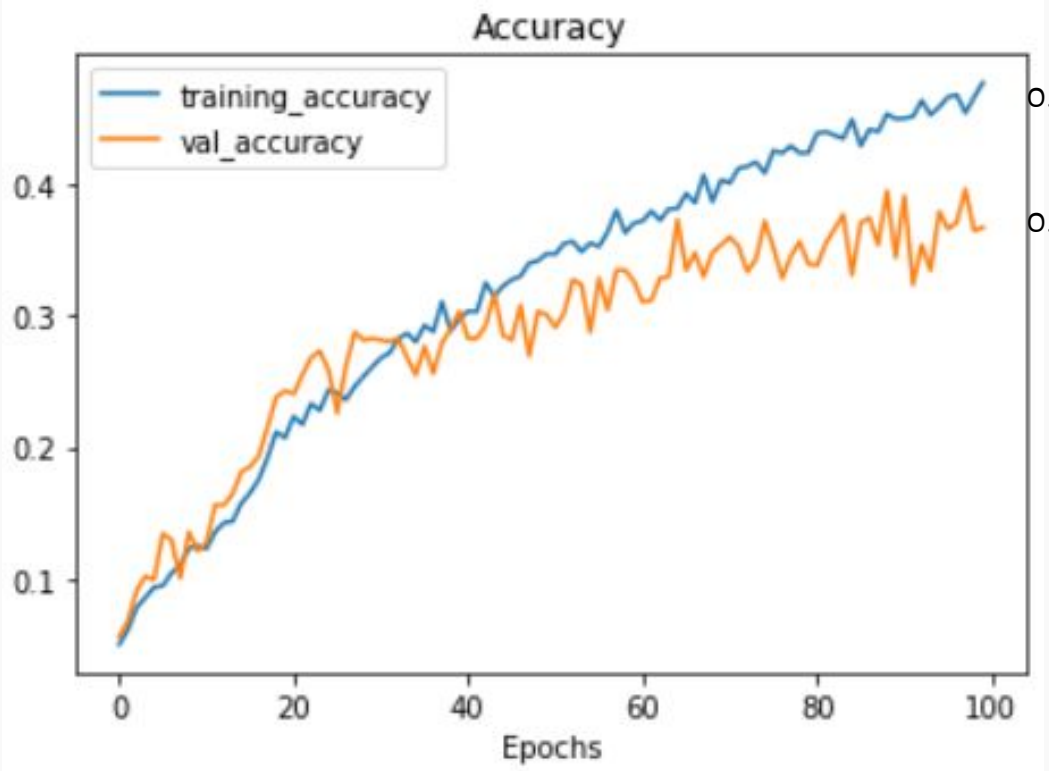
-Data Augmentation
-Rescale
-ZCA whitening



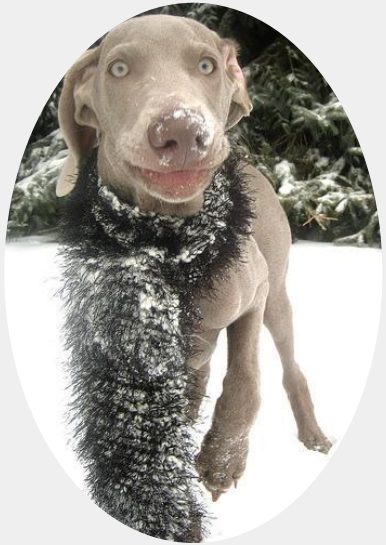
Ajout de deux couches de convolution (2e CNN)



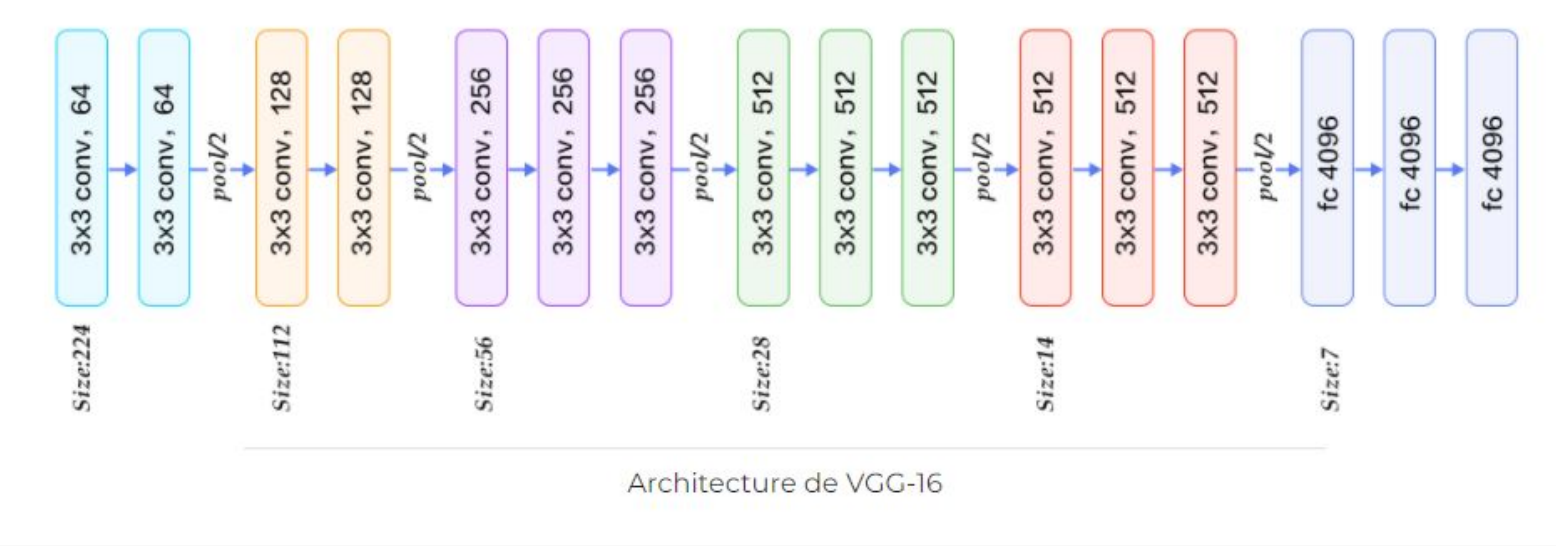
Performances 2e CNN



03 Transfer learning



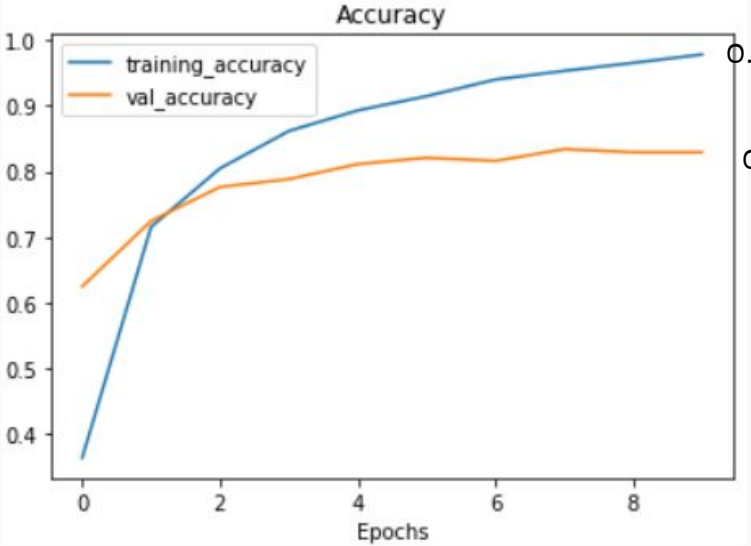
VGG-16



On utilise le réseau de neurones VGG-16 pré-entraîné sur Image Net.

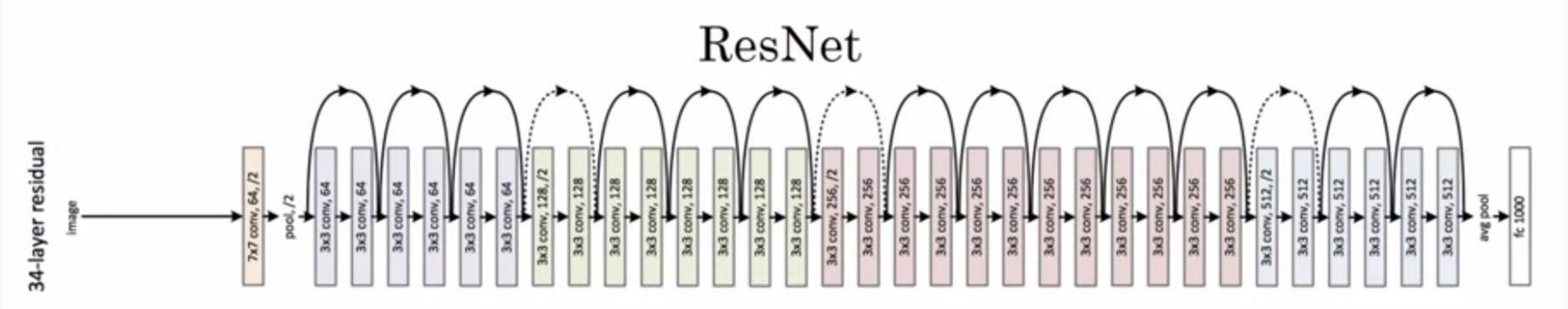
Utilisation du réseau VGG-16 pour notre classification

On ré-entraine uniquement la dernière couche, conservation des poids pour les précédentes



10 epochs, batch-size =32, optimizer= Adam, learning_rate=0.001

Residual Network



Resnet50

On utilise le réseau de neurones RESNET 5 0 pré-entraîné sur Image Net. optimizer=Adam

Data augmentation	Ré-entrainement	Batch size	Nombre d'epoch	learning rate	Accuracy train	Accuracy test
non	uniquement la dernière couche (extraction des features)	**32**	**10**	**0.001**	**0.999**	**0.92**
		256	20	0.001	0.999	0.92
		64	20	0.0001	0.96	0.90
non	dernières couches	128	5	0.0001	0.999	0.91
oui	uniquement la dernière couche	32	15	0.001	0.97	0.91

04

Conclusion