Supervised learning :
Classification de feuilles de pommier

UE 1 - Mathématiques et informatique, DATA CHALLENGE

Valentin GOUPILLE
Dorcas LIN

M2BV PHP
Supervised by : Felix MERCIER

Introduction

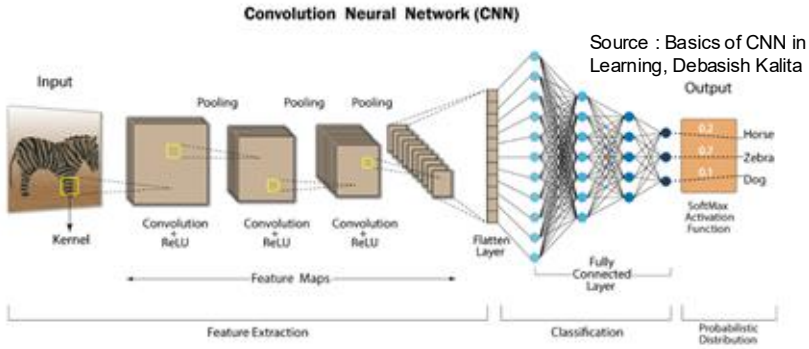
Sujet : **classification des feuilles de pommier**

- Dataset :

  - 4 catégories : **healthy, apple scab, black rot, cedar apple rot**

- Objectifs :

  - **classification** automatique avec **CNN**
  - **optimiser des modèles**
    - varier les **hyperparamètres, paramètres de l'architecture**
    - avec entraînement, test et validation sur banque annotée



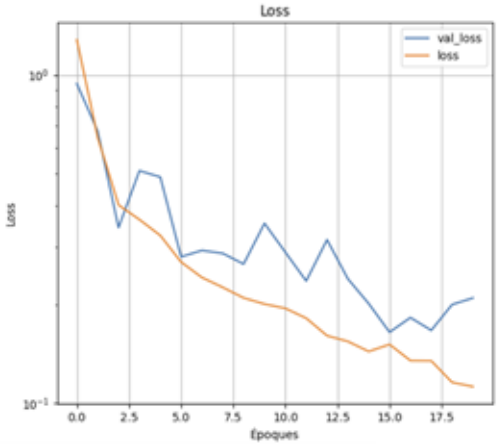
Source : Basics of CNN in Deep Learning, Debasish Kalita

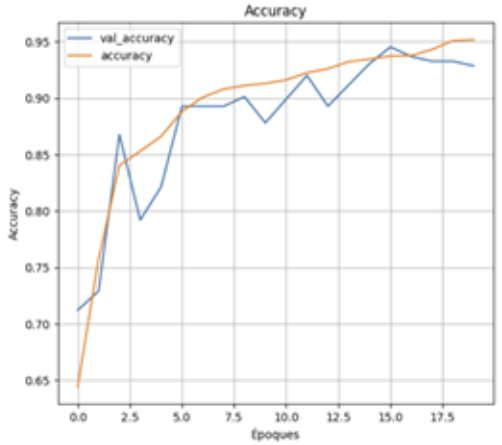
Google Colaboratory

Optimisation de l'entraînement :

Batch size : 32
Epoch : 20
Learning rate : 0.0001

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 first_conv_layer (Conv2D)   (None, 126, 126, 32)      896

 max_pooling2d (MaxPooling2  (None, 63, 63, 32)        0
 D)

 conv2d (Conv2D)             (None, 61, 61, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 30, 30, 64)        0
 g2D)

 conv2d_1 (Conv2D)           (None, 28, 28, 128)       73856

 max_pooling2d_2 (MaxPoolin  (None, 14, 14, 128)       0
 g2D)

 last_conv_layer (Conv2D)    (None, 12, 12, 128)       147584

 max_pooling2d_3 (MaxPoolin  (None, 6, 6, 128)         0
 g2D)

 flatten (Flatten)           (None, 4608)              0

 dense (Dense)               (None, 128)               589952

 dense_1 (Dense)             (None, 4)                 516

=================================================================
Total params: 831300 (3.17 MB)
Trainable params: 831300 (3.17 MB)
Non-trainable params: 0 (0.00 Byte)
```



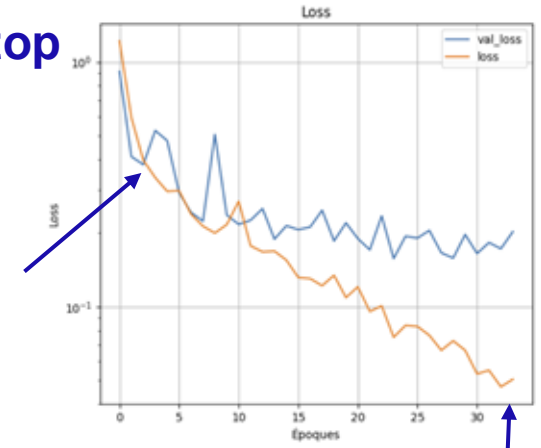


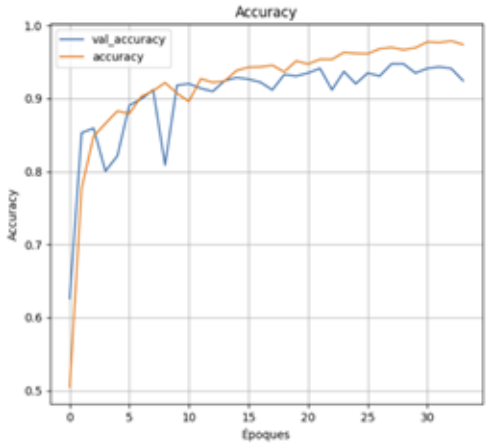
Underfitting

Test score: 0.209
Test accuracy: 0.928

Optimisation de l'entraînement : hyperparamètre

Batch size : 32
Epoch : 20→**200 + early stop**
Learning rate : 0.0001





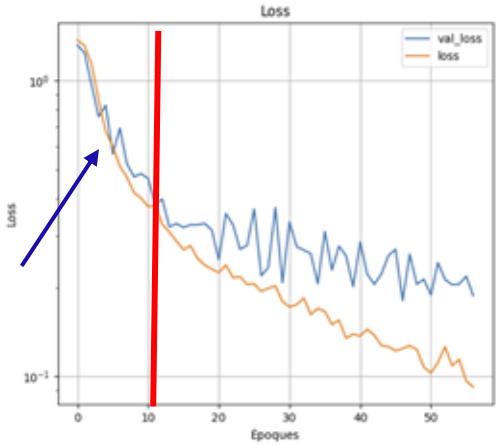
Overfitting

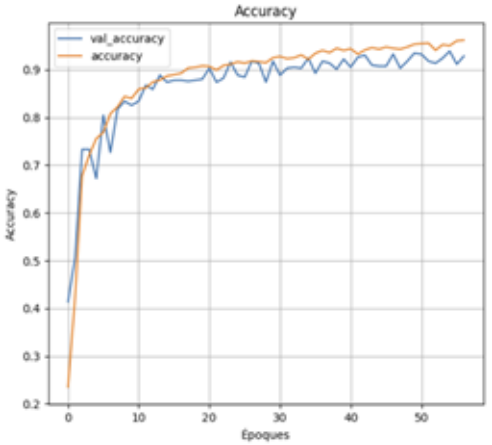
33

Test score: 0.202
Test accuracy: 0.924

Optimisation de l'entraînement : hyperparamètre

Batch size : 32→**128**
Epoch : 200 + early stop
Learning rate : 0.0001





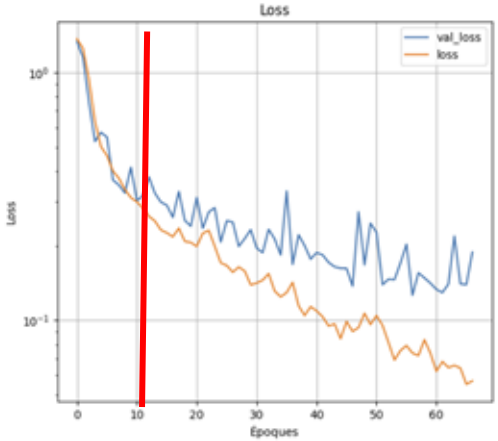
Overfitting

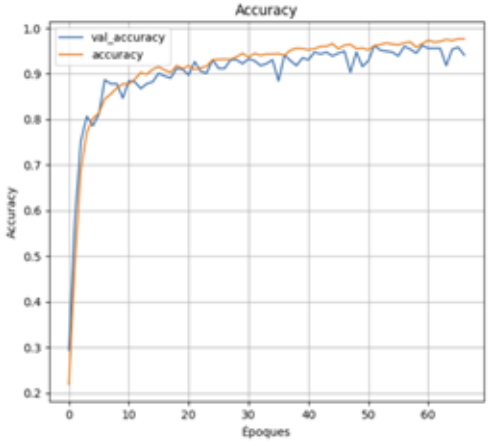
Test score: 0.188
Test accuracy: 0.929

Optimisation de l'entraînement : architecture

Batch size : 128
Epoch : 200 + early stop
Learning rate : 0.0001
Dropout : n°1 : 0.3

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 first_conv_layer (Conv2D)   (None, 126, 126, 32)      896

 max_pooling2d_12 (MaxPooli  (None, 63, 63, 32)        0
 ng2D)

 conv2d_6 (Conv2D)           (None, 61, 61, 64)        18496

 max_pooling2d_13 (MaxPooli  (None, 30, 30, 64)        0
 ng2D)

 conv2d_7 (Conv2D)           (None, 28, 28, 128)       73856

 max_pooling2d_14 (MaxPooli  (None, 14, 14, 128)       0
 ng2D)

 last_conv_layer (Conv2D)    (None, 12, 12, 128)       147584

 max_pooling2d_15 (MaxPooli  (None, 6, 6, 128)         0
 ng2D)

 flatten_3 (Flatten)         (None, 4608)              0

 dropout (Dropout)           (None, 4608)              0

 dense_6 (Dense)             (None, 128)               589952

 dense_7 (Dense)             (None, 4)                 516

=================================================================
Total params: 831300 (3.17 MB)
Trainable params: 831300 (3.17 MB)
Non-trainable params: 0 (0.00 Byte)
```





Overfitting

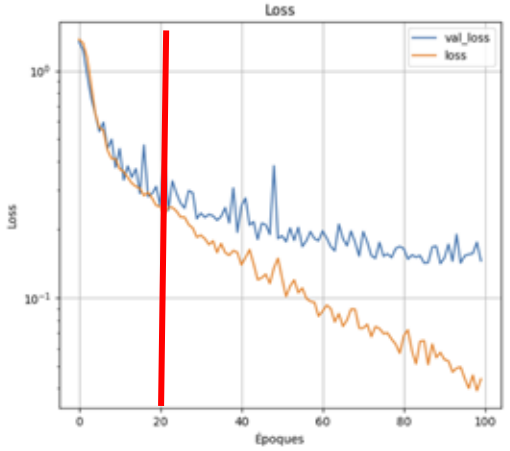
Test score: 0.188
Test accuracy: 0.941

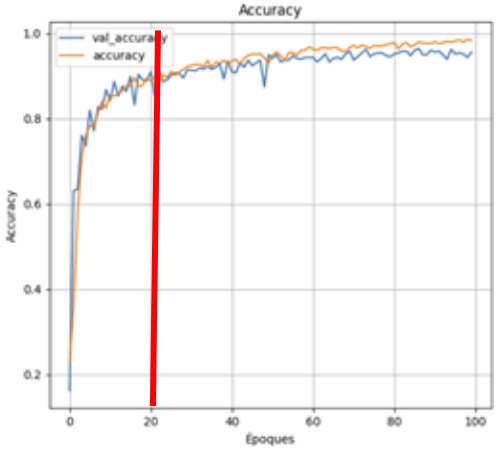
Optimisation de l'entraînement : architecture

Batch size : 128
Epoch : 200 + early stop
Learning rate : 0.0001
Dropout : n°1 : 0.3

n°2 : 0.3

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 first_conv_layer (Conv2D)   (None, 126, 126, 32)      896

 max_pooling2d_16 (MaxPooli  (None, 63, 63, 32)        0
 ng2D)

 conv2d_8 (Conv2D)           (None, 61, 61, 64)        18496

 max_pooling2d_17 (MaxPooli  (None, 30, 30, 64)        0
 ng2D)

 conv2d_9 (Conv2D)           (None, 28, 28, 128)       73856

 max_pooling2d_18 (MaxPooli  (None, 14, 14, 128)       0
 ng2D)

 last_conv_layer (Conv2D)    (None, 12, 12, 128)       147584

 max_pooling2d_19 (MaxPooli  (None, 6, 6, 128)         0
 ng2D)

 flatten_4 (Flatten)         (None, 4608)              0

 dropout_1 (Dropout)         (None, 4608)              0

 dense_8 (Dense)             (None, 128)               589952

 dropout_2 (Dropout)         (None, 128)               0

 dense_9 (Dense)             (None, 4)                 516

=================================================================
Total params: 831300 (3.17 MB)
Trainable params: 831300 (3.17 MB)
Non-trainable params: 0 (0.00 Byte)
```





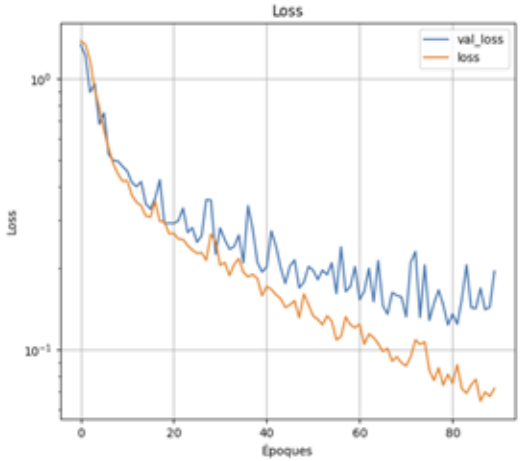
10→**20**

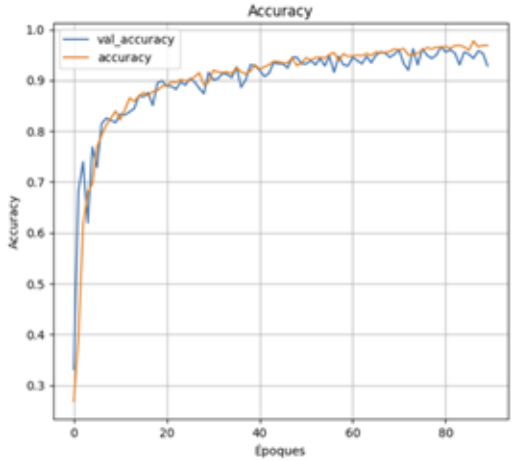
Less overfitting

Test score: 0.145
Test accuracy: 0.955

Optimisation de l'entraînement : architecture

Batch size : 128
Epoch : 200 + early stop
Learning rate : 0.0001
Dropout : n°1 : 0.3 → 0.5
n°2 : 0.3



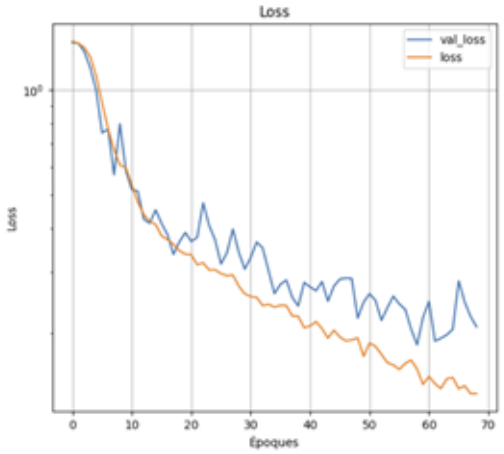


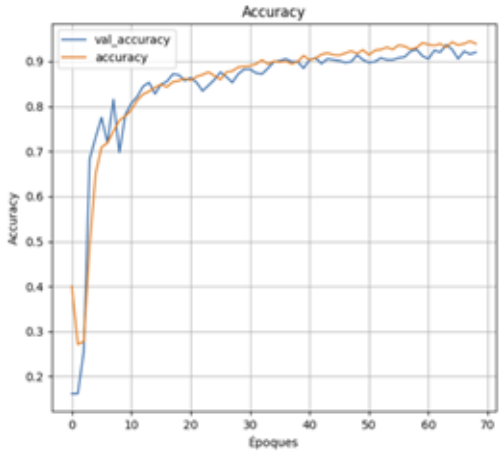
Less overfitting

Test score: 0.194
Test accuracy: 0.928

Optimisation de l'entraînement : hyperparamètre

Batch size : 128→**256**
Epoch : 200 + early stop
Learning rate : 0.0001
Dropout :1 x 0.5
1 x 0.3





Less noise

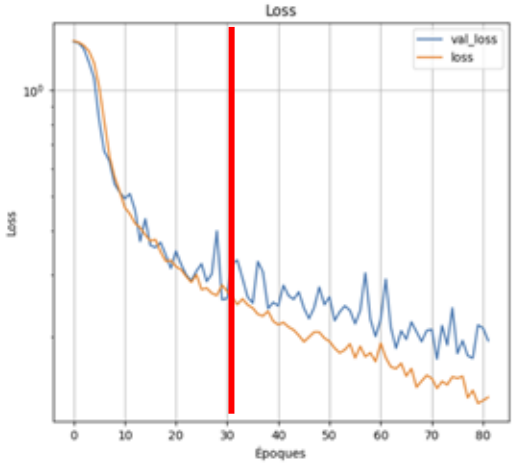
Test score: 0.208
Test accuracy: 0.920

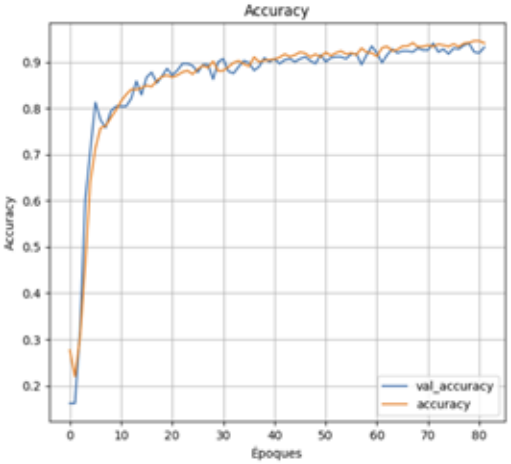
Modification de l'architecture

Réduction du nombre de noyaux :

831 300 → 687 908 parameters

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 first_conv_layer (Conv2D)   (None, 126, 126, 16)      448

 max_pooling2d_28 (MaxPooli  (None, 63, 63, 16)        0
 ng2D)

 conv2d_14 (Conv2D)          (None, 61, 61, 32)        4640

 max_pooling2d_29 (MaxPooli  (None, 30, 30, 32)        0
 ng2D)

 conv2d_15 (Conv2D)          (None, 28, 28, 64)        18496

 max_pooling2d_30 (MaxPooli  (None, 14, 14, 64)        0
 ng2D)

 last_conv_layer (Conv2D)    (None, 12, 12, 128)       73856

 max_pooling2d_31 (MaxPooli  (None, 6, 6, 128)         0
 ng2D)

 flatten_7 (Flatten)         (None, 4608)              0

 dropout_7 (Dropout)         (None, 4608)              0

 dense_14 (Dense)            (None, 128)               589952

 dropout_8 (Dropout)         (None, 128)               0

 dense_15 (Dense)            (None, 4)                 516

=================================================================
Total params: 687908 (2.62 MB)
Trainable params: 687908 (2.62 MB)
Non-trainable params: 0 (0.00 Byte)
```





15→**30**

Less overfitting

Test score: 0.195
Test accuracy: 0.932

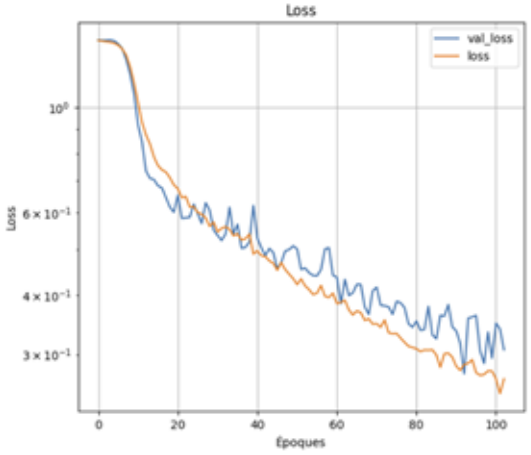
Modification de l'architecture

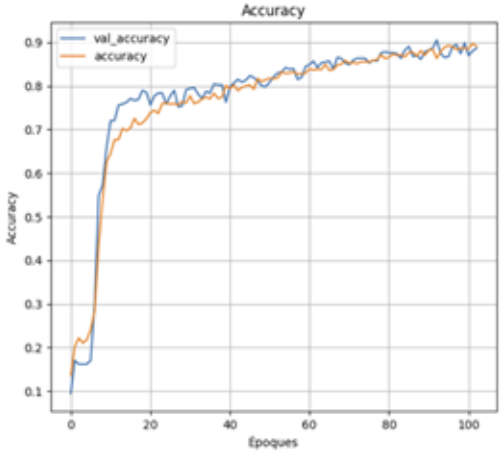
Ajout d'une couche de convolution :

687 908 parameters →165 940

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 first_conv_layer (Conv2D)   (None, 126, 126, 16)      448

 max_pooling2d_37 (MaxPooli  (None, 63, 63, 16)        0
 ng2D)

 conv2d_18 (Conv2D)          (None, 61, 61, 16)        2320

 max_pooling2d_38 (MaxPooli  (None, 30, 30, 16)        0
 ng2D)

 conv2d_19 (Conv2D)          (None, 28, 28, 32)        4640

 max_pooling2d_39 (MaxPooli  (None, 14, 14, 32)        0
 ng2D)

 conv2d_20 (Conv2D)          (None, 12, 12, 64)        18496

 max_pooling2d_40 (MaxPooli  (None, 6, 6, 64)          0
 ng2D)

 last_conv_layer (Conv2D)    (None, 4, 4, 128)         73856

 max_pooling2d_41 (MaxPooli  (None, 2, 2, 128)         0
 ng2D)

 flatten_9 (Flatten)         (None, 512)               0

 dropout_11 (Dropout)        (None, 512)               0

 dense_18 (Dense)            (None, 128)               65664

 dropout_12 (Dropout)        (None, 128)               0

 dense_19 (Dense)            (None, 4)                 516

=================================================================
Total params: 165940 (648.20 KB)
Trainable params: 165940 (648.20 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

589 952 parameters → 65 664

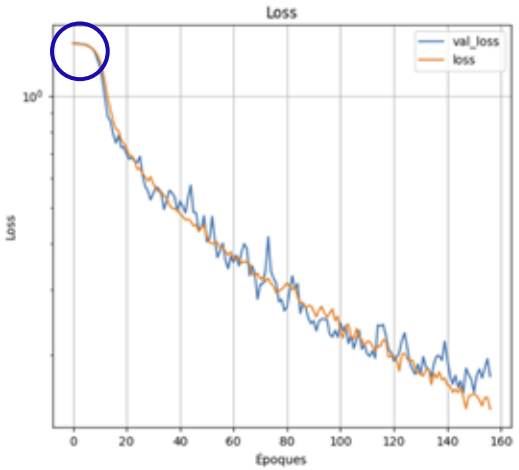


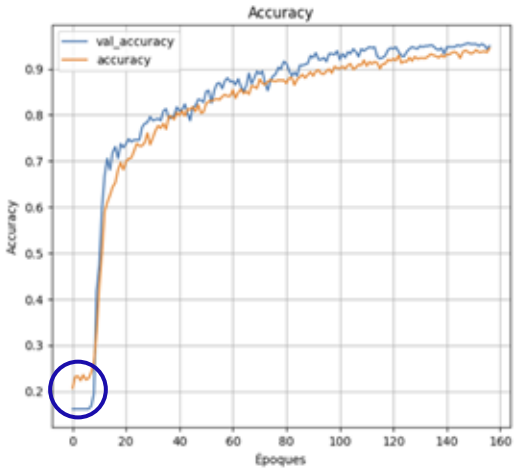


Test score: 0.306
Test accuracy: 0.887

Optimisation de l'entraînement

Batch size : 128→256
Epoch : 200 + early stop
Learning rate : 0.0001
Dropout : n°1 : 0.5
n°2 : 0.3 → 0.5





Less overfitting

Test score: 0.176
Test accuracy: 0.950

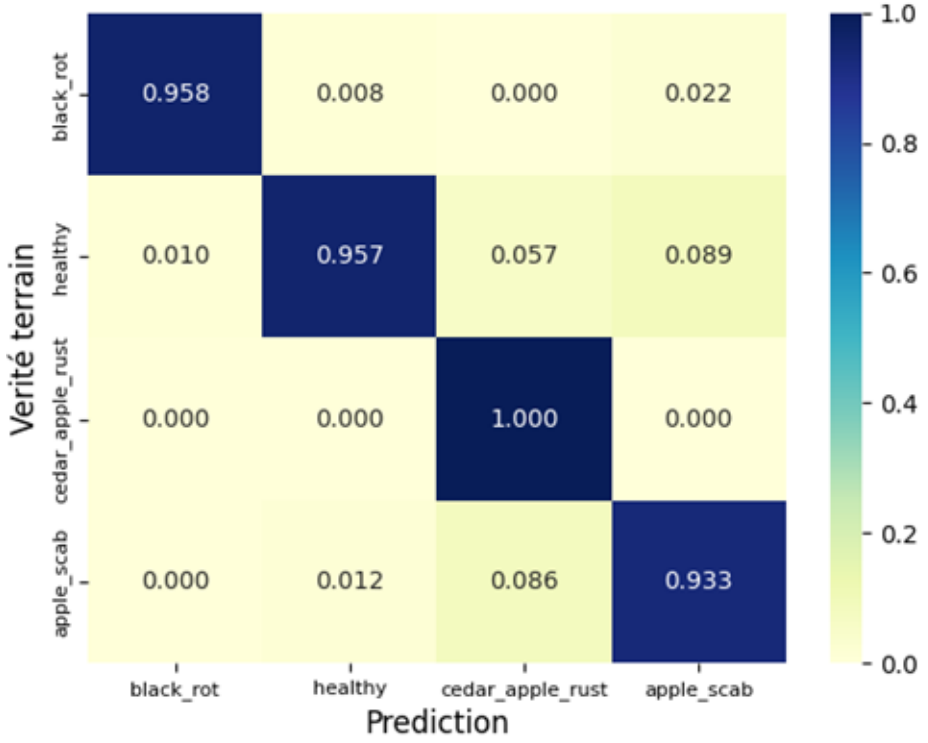
Test du modèle sur plusieurs jeux de données

Répétition	Test score	Test accuracy	Erreurs
1	0,2275	0,9180	39
2	0,1552	0,9516	23
3	0,2967	0,8991	48
4	**0,3513**	**0,8739**	**60**
5	0,1369	0,9474	25
6	**0,1243**	**0,9558**	**21**
7	0,1942	0,9327	32
8	0,2611	0,9096	43
9	0,2153	0,9327	32
10	0,2072	0,9432	27
Moyenne	**0,2170**	**0,9264**	**35**
Ecart-type	0,0714	0,0261	12

Evaluation des répétitions

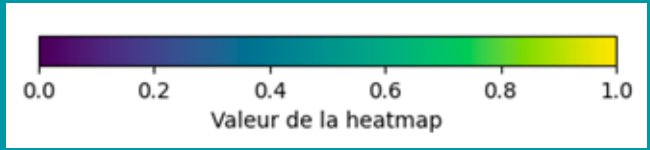
Interprétation : Matrice de confusion

- **100% de prédiction pour cedar apple rust**
- **Prédiction moins précise pour apple scab**
  - Faux positif avec cedar apple rust
  - Faux négatif avec healthy

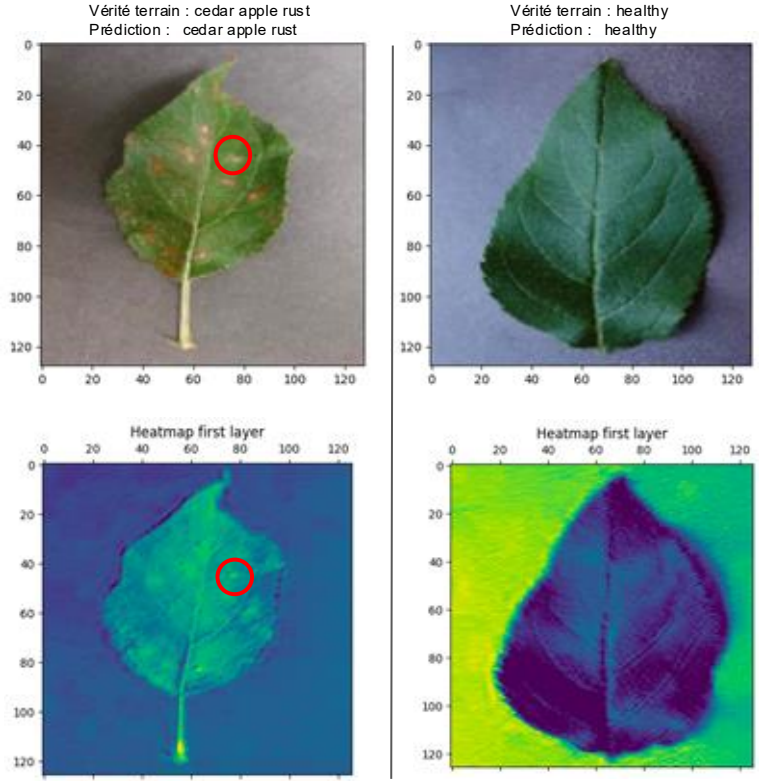


Matrice de confusion de la répétition n°6

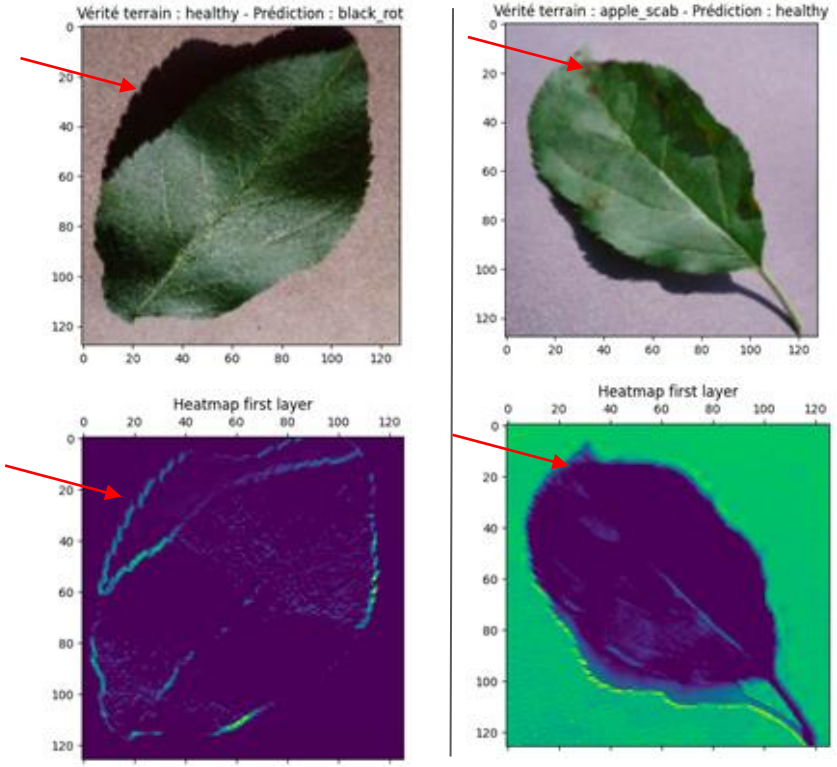
Interprétation : Gradcam



Bonne classification :



Mauvaise classification :



Pour finir…

Conclusion

- Méthode → modèle +/- précis
- Variation du modèle selon le jeu de données
- Plus grande variation de prédiction : apple scab

Perspectives

- Utiliser une plus grande banque d'images annotées
- Tester d'autres architectures :

  augmenter le nombre de noyaux, jouer avec plus de dropouts, …
- Automatiser pour trouver les paramètres adaptés : random/boucles

Pour aller plus loin…

- recherche automatique des paramètres

```
###Création du modèle aléatoire avec ses différentes caractéristiques possibles: ex nb de couches, de noyaux...
def build_model(hp):
    model = keras.Sequential([
        Conv2D(
            filters=hp.Int('conv_1_filter', min_value=32, max_value=64, step=16),###ici on peut modifier
            kernel_size=hp.Choice('conv_1_kernel', values=[3, 5, 7,9]),###ici on peut modifier [3,5,7,9]
            activation='relu',
            input_shape=X_train[0].shape
        ),
        MaxPooling2D(pool_size=(2, 2)),

        Conv2D(
            filters=hp.Int('conv_2_filter', min_value=64, max_value=256, step=16),###ici on peut modifier max_value=128
            kernel_size=hp.Choice('conv_2_kernel', values=[3, 5]),
            activation='relu'
        ),
        MaxPooling2D(pool_size=(2, 2)),

        Conv2D(
            filters=hp.Int('conv_3_filter', min_value=64, max_value=512, step=16),###ici on peut modifier.  max_value=128
            kernel_size=hp.Choice('conv_3_kernel', values=[3, 5]),
            activation='relu'
        ),
        MaxPooling2D(pool_size=(2, 2)),

        Conv2D(
            filters=hp.Int('conv_3_filter', min_value=64, max_value=512, step=16),###ici on peut modifier.  max_value=128
            kernel_size=hp.Choice('conv_3_kernel', values=[3, 5]),
            activation='relu'
        ),
        MaxPooling2D(pool_size=(2, 2)),

        Flatten(),
        #Dropout(0.5),
        Dense(
            units=hp.Int('dense_1_units', min_value=32, max_value=128, step=16),
            activation='relu'
        ),
        #Dropout(0.5),
        keras.layers.Dense(4, activation='softmax')  # Change 4 to the number of output classes ###ici on met le nombre de classe (la sortie)
    ])
```

```
nom_dossier_autofind= "Autofind_1" ####choisir le nom de la recherche auto
#importing random search
import keras_tuner as kt
#from keras-tuner import RandomSearch
#creating randomsearch object
tuner = kt.RandomSearch(build_model,
    objective= 'val_accuracy', #### on peut changer val_accuracy
    max_trials = 20,
    directory= sous_dossier_modele,
    project_name= nom_dossier_autofind                    ) ###on peut mettre 5, 15, 20
#search best parameter
tuner.search (X_train, y_train, validation_data=(X_test,y_test),class_weight = class_weights, #ajuste les poids
    epochs=100, batch_size=64,
    verbose=1,
    callbacks=[early_stop]
              ) ###ici on peut modifier le nombre d'époque de base c'était 3 , 60

#callbacks=[Model_check, early_stop ,reduce_lr]
tuner.results_summary()
model = tuner.get_best_models (num_models=1)[0]


#summary of best model
model.summary() ## affiche les caractéristiques du modèle sauf le taux de dropout , ici il y en a pas
```

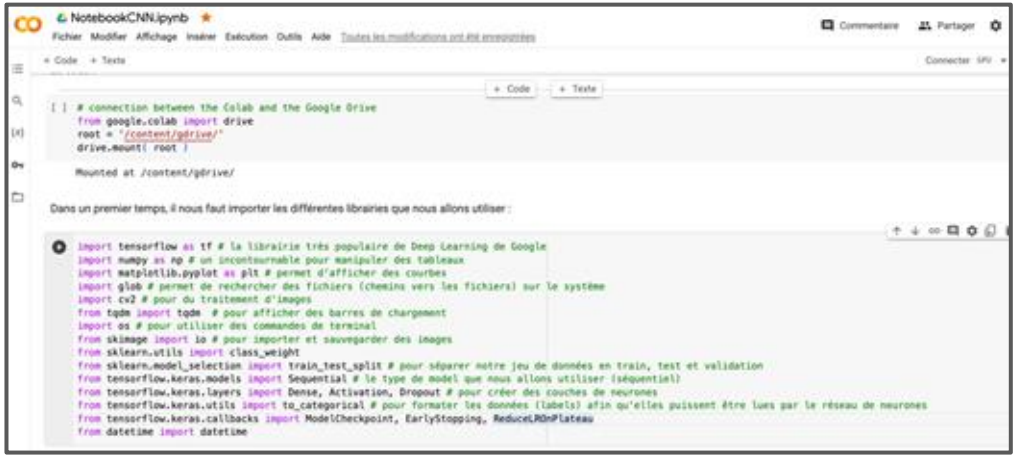
```
Trial 20 Complete [00h 03m 37s]
val_accuracy: 0.9327731132507324

Best val_accuracy So Far: 0.9621848464012146
Total elapsed time: 00h 44m 45s
Results summary
Results in /content/gdrive/MyDrive/resultats_test_09_11/modele_0/Autofind_1
Showing 10 best trials
Objective(name="val_accuracy", direction="max")
```

```
Trial 00 summary
Hyperparameters:
conv_1_filter: 48
conv_1_kernel: 3
conv_2_filter: 128
conv_2_kernel: 5
conv_3_filter: 240
conv_3_kernel: 5
dense_1_units: 48
learning_rate: 0.0001
Score: 0.9621848464012146
```

RandomSearch,
Hyperband,
BayesianOptimization,
Sklearn

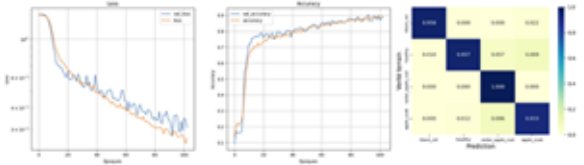
- **Notebook**

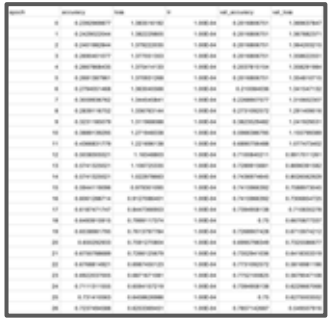
Google Colaboratory



Google Drive

- **Cahier de manipulation**
- **Sauvegarde automatique**



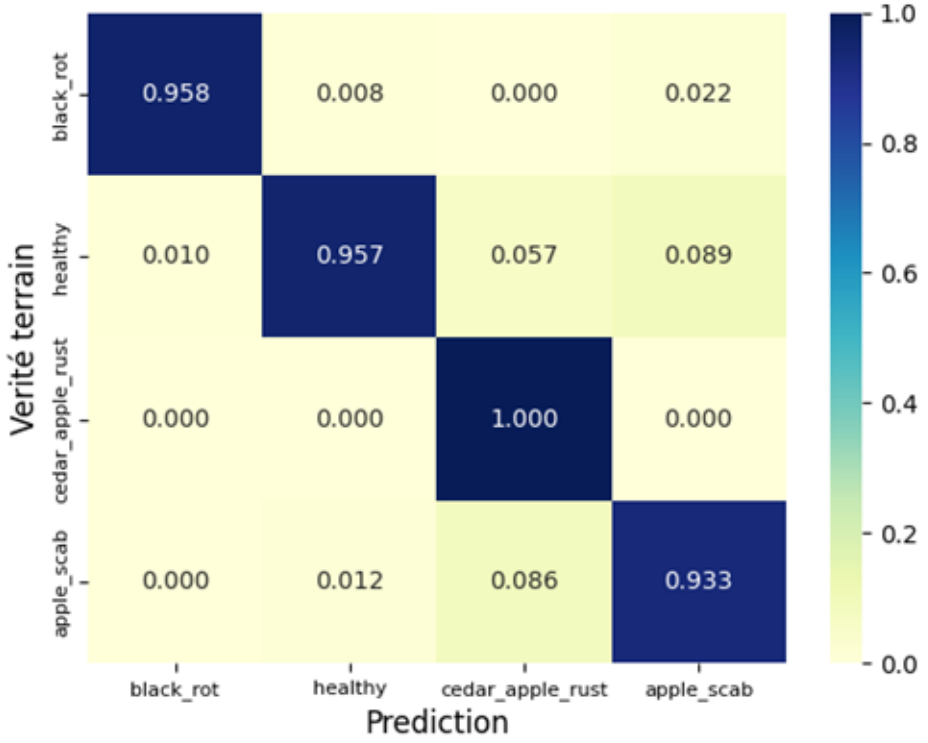


Interprétation : Matrice de confusion

- **100% de prédiction pour cedar apple rust**
- **Prédiction moins précise pour apple scab**
  - Faux positif avec cedar apple rust
  - Faux négatif avec healthy

Moyenne et écart-type de prédictions des 10 modèles :

Modèle	black_rot	healthy	cedar_apple_rust	apple _scab
mean	95,62	91,85	98,83	89,03
sd	2,21	3,40	1,25	6,25



Matrice de confusion de la répétition n°6