TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHÊN TPHCM

KHOA CÔNG NGHỆ THÔNG TIN

Pattern Recognition - 19KHMT1

HW02: Classification with Neural Network

Instructor

- Lê Hoàng Thái
- Nguyễn Ngọc Thảo
- Lê Thanh Phong

Group 8 – 19KHMT1

Member – Student ID:

- Nguyễn Việt Thanh Duy - 19127378
- Hình Ích Trình - 19127303
- Trần Phạm Minh Nhựt - 19127501

1. Tasks assignment:

Member	Task	Progress
Nguyễn Việt Thanh Duy	Preprocessing data Problem Analysis and designing model architecture Optimize the model (increasing accuracy and reducing running time)	100%
Hình Ích Trình	Preprocessing data Problem Analysis and designing model architecture Writing report	100%
Trần Phạm Minh Nhựt	Preprocessing data Problem Analysis and designing model architecture Writing report	100%

2. Problem and data analysis:

2.1. Problem analysis:

- Problem: given a dataset which contains 1821 training images and 1821 testing images, decide a neural network model to predict the plant disease each testing images.
- Input: a plant image.
- Output: 1 of 4 diseases: multiple_diseases, healthy, rust or scab.
- There are a lot of models for image classification problem, in this specific problem, we will use the most popular one: convolutional neural network (CNN).

2.2. Data analysis:

- The images folder contains: 1821 training images and 1821 testing images, each image has the dimension of 2048 × 1365.
- Each class sample:

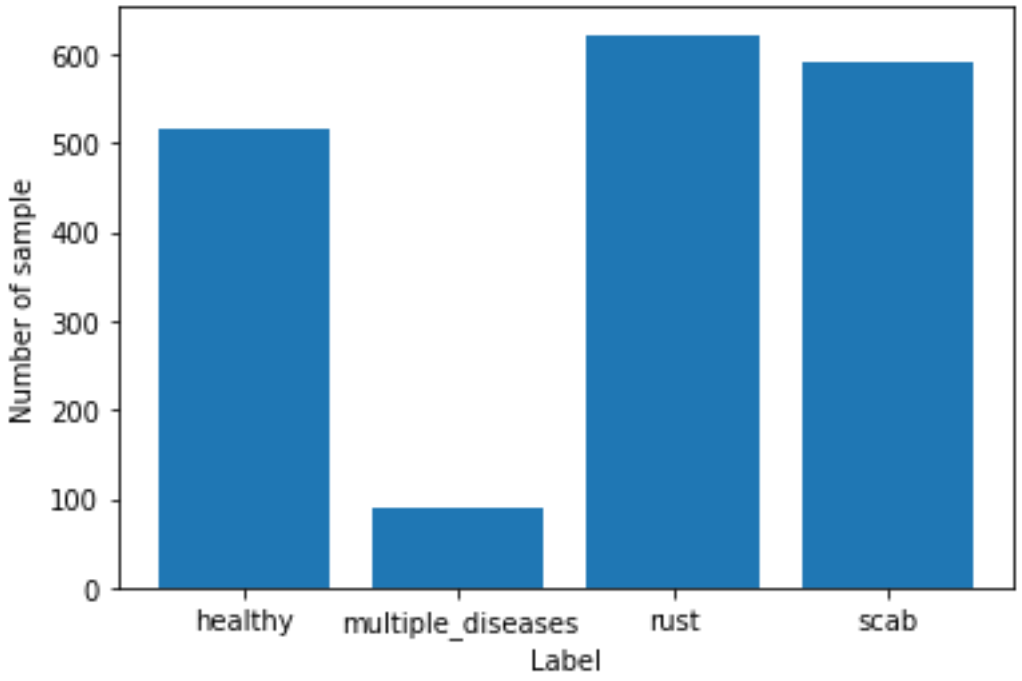
Scab

Healthy

Multiple diseases

Rust

- Number of samples for each disease:



- We can see the number of samples of multiple_disease class is way fewer than remaining, so this training set is imbalance.
  => The model is trained based on this training set so it may predict the multiple_disease class less effective than the others (misclassification).
- For this problem, we need a training set contains:
  - x_train: an array of training images, each image is an 3-dimensional array contains 3 channel Red, Green, Blue. Each channel is a matrix.
  - y_train: an array contains the disease of each respective x_train image (0: multiple_diseases, 1: healthy, 2: rust, 3: scab).
- First, we will use **opencv** module to read the images.
- For each training image, we will resize it to 180 × 180 and append it to the x_train array.
- As the pixel values range from 0 to 256, apart from 0 the range is 255. We can divide all the values of x_train by 255 to convert it to range from 0 to 1. In this way, the numbers will be small, and the computation becomes easier and faster.

- We construct y_train by first reading the train.csv file with **pandas** module. Then we use the argmax function on 4 columns to get the label.
- We use **pickle** module to save x_train and y_train to a .pkl file. So that we don't need to read the dataset multiple times.
- We divide the training set into 2 parts: training set and validation set with the ratio of 7:3. This is for model training process.
- Before diving into training phase, there are just 1821 training images, so we use the concept of data augmentation to generate more training data. Specifically, we use the ImageDataGenerator from **keras** image preprocessing module with some parameters' adjustment.

3. Relative concepts:

3.1. Data Augmentation:

- **Data augmentation** in data analysis are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It acts as a regularizer and helps reduce overfitting when training a machine learning model.[1] It is closely related to oversampling in data analysis.

3.2. Batch Normalization:

- To facilitate learning, we typically normalize the initial values of our parameters by initializing them with zero mean and unit variance. As training progresses and we update parameters to different extents, we lose this normalization, which slows down training and amplifies changes as the network becomes deeper.
- Batch normalization [30] reestablishes these normalizations for every mini-batch and changes are back-propagated through the operation as well. By making normalization part of the model architecture, we are able to use higher learning rates and pay less attention to the initialization parameters. Batch normalization additionally acts as a regularizer, reducing (and sometimes even eliminating) the need for Dropout.

Input: $x : N \times D$

Learnable params: $\gamma, \beta : D$

Intermediates: $\mu, \sigma : D$, $\hat{x} : N \times D$

Output: $y : N \times D$

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \mu_j)^2$$

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

3.3. Dropout Layer:

- Dropout is a technique meant to prevent overfitting the training data by dropping out units in a neural network. In practice, neurons are either dropped with probability *p* or kept with probability p-1

3.4. Sparse categorical cross entropy (loss function):

- The **Cross-Entropy Loss** is actually the only loss we are discussing here. The other losses names written in the title are other names or variations of it. The CE Loss is defined as:

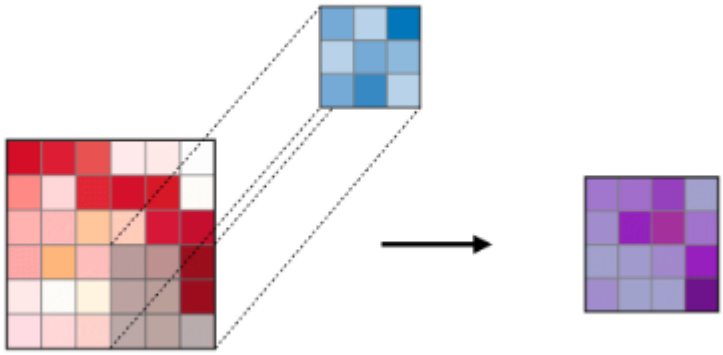
$$CE = -\sum_{i}^{C} t_i log(s_i)$$

Where t_i and S_i are the groundtruth and the CNN score for each class i in C

3.5. Convolutional Neural Network (CNN):

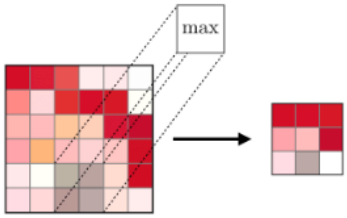
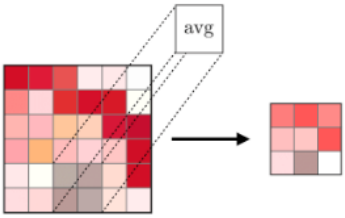
3.5.1. Convolutional Layer:

- The convolution layer (CONV) uses filters that perform convolution operations as it is scanning the input *I* with respect to its dimensions. Its hyperparameters include the filter size F and stride S. The resulting output O is called *feature map* or *activation map*.



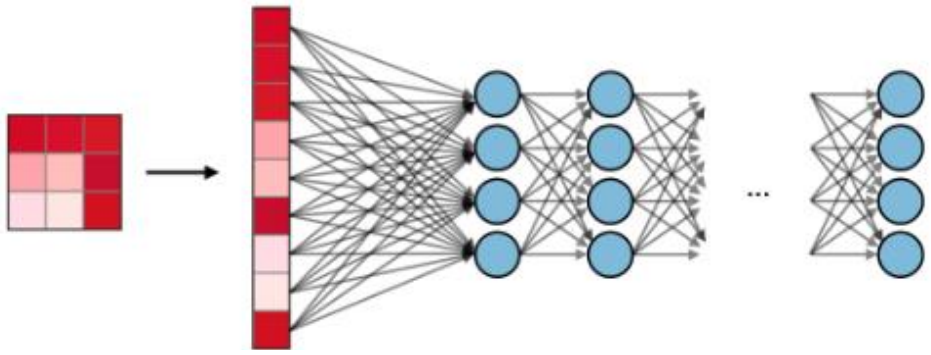
3.5.2. Pooling Layer:

- The pooling layer (POOL) is a downsampling operation, typically applied after a convolution layer, which does some spatial invariance. In particular, max and average pooling are special kinds of pooling where the maximum and average value is taken, respectively.

Type	Max pooling	Average pooling
Purpose	Each pooling operation selects the maximum value of the current view	Each pooling operation averages the values of the current view
Illustration		
Comments	• Preserves detected features • Most commonly used	• Downsamples feature map • Used in LeNet

3.5.3. Fully Connected Layer:

- The fully connected layer (FC) operates on a flattened input where each input is connected to all neurons. If present, FC layers are usually found towards the end of CNN architectures and can be used to optimize objectives such as class scores.

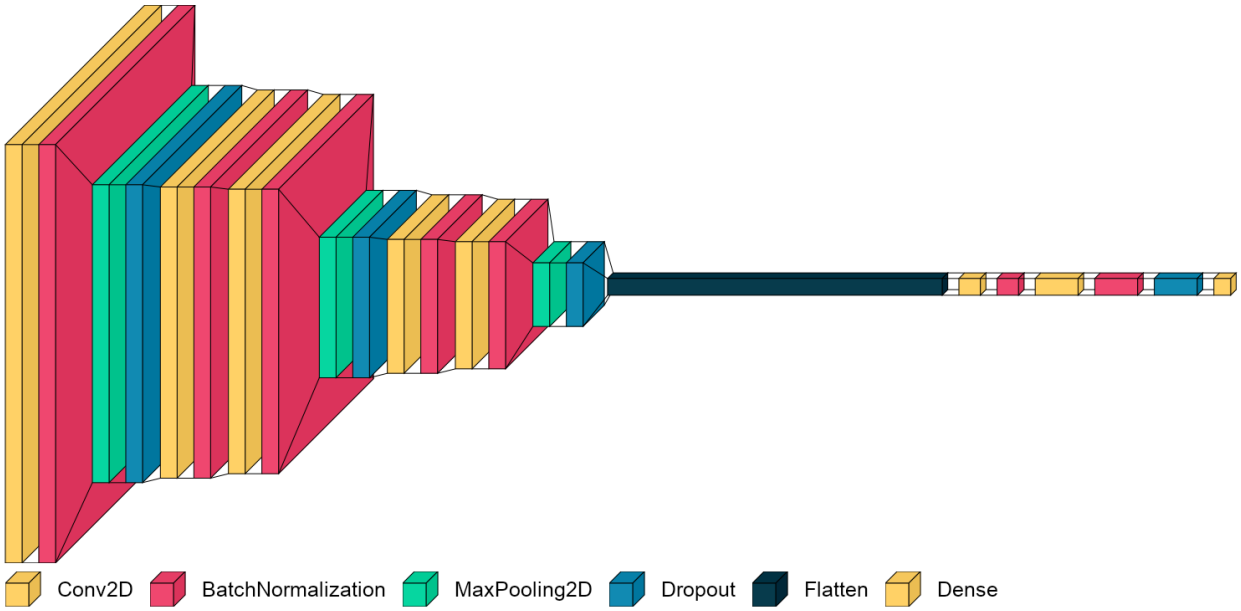


4. Designing the model architecture:

- Layers of the network along with the functions of each of these layers:
  - The first layer of the network is an Input Layer, which contains the images to be classified. The size of the image is 180×180x3, then each image has 97200 elements.
  - The next layer of the network is a Convolutional Layer called Conv2D. This layer takes as input images from the Input layer. In the Convolutional Layer, our group uses many filters of equal size to scan the input image (from the Input layer) and create feature mappings for the image. After the Convolutional Layer, our group also uses the ReLU Activate function and Batch Normalization layer.
  - Next, our team defines a Pooling Layer with a maximum value called MaxPooling2D. This layer receives the input as a result of the Convolutional Layer above and it performs information filtering, removing noise information before transmitting to the next layer of the network. In the experimental part, our group uses a window with a pool size of 2 × 2 to get the largest value of the 4 values that this window scans on the output matrix of the Convolutional Layer.
  - After the Pooling Layer, our group uses a Dropout Layer with the Dropout value set to 0.25. It is configured to randomly exclude 25% of the total number of neurons in the layer to reduce the problem of overfitting.
  - To extract a lot of useful information from the image, our team built a deeper CNN network by adding some layers of the network, they include the following layers: Convolutional, Batch Normalization, Pooling, Dropout. In which, Convolutional Layer

will use many filters with equal size, Pooling Layer uses window with pool size of 2×2, Dropout Layer with Dropout value of 0.25.

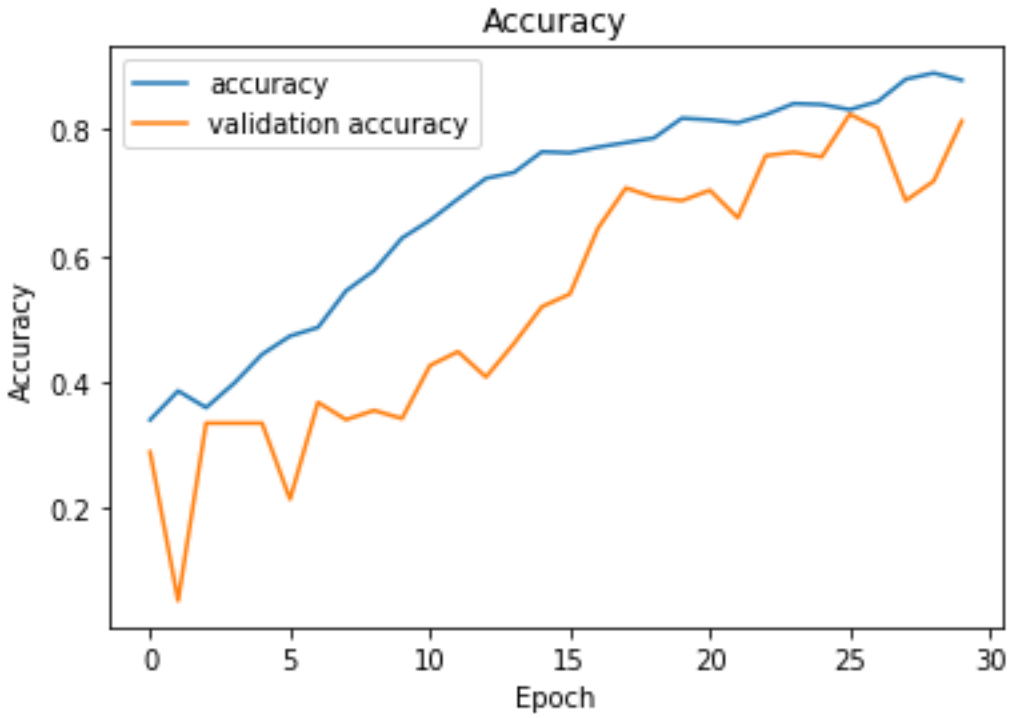
- Next, our team uses a layer that converts the 2D matrix data into a vector called Flatten. As a result, our group obtains a vector of feature values of the image, which is suitable for the input format of an MLP.
- After obtaining feature vectors of images through layers of CNN, our team uses MLP as a classifier to classify images. Our team's MLP uses many hidden layers with the number of neurons configured during the experiment. Since this is a multi-class (4-class) classification task, the output of our group's MLP uses 4 neurons and a softmax activation function to make predictions that are probabilistic values for each class.



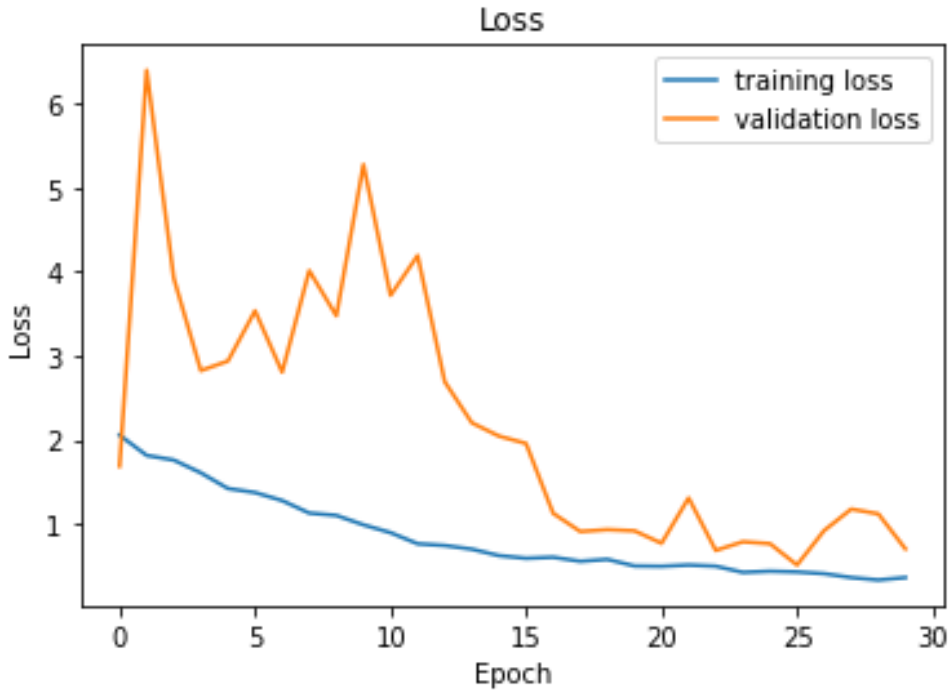
Our CNN architecture

5. Evaluation, commenting on the model's result:

5.1. Evaluation:

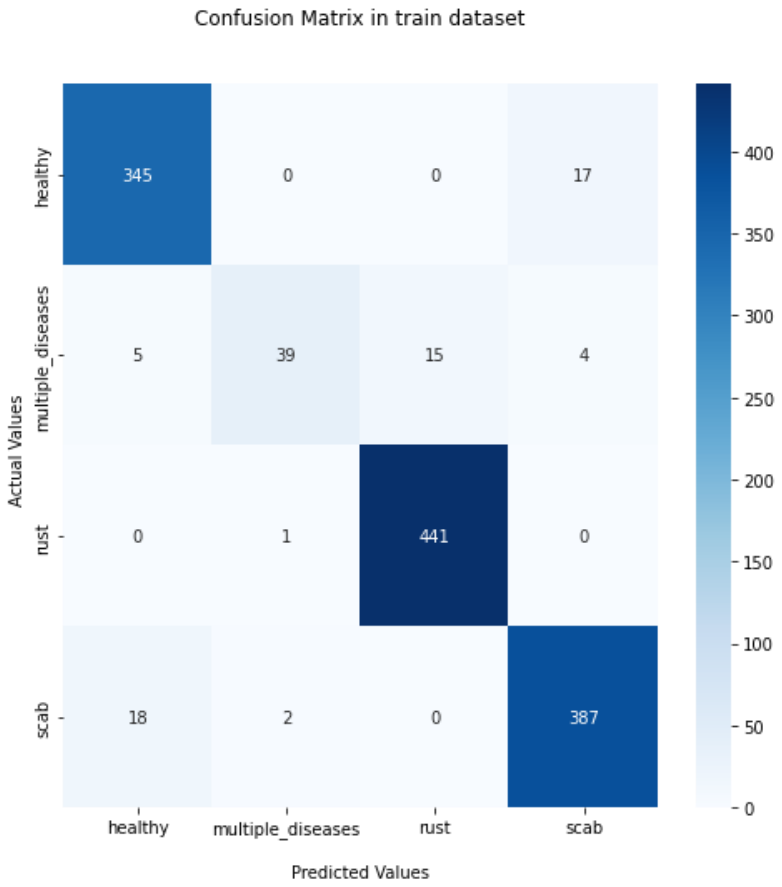


History of accuracy in training process of the last trained model



History of loss in training process of the last trained model

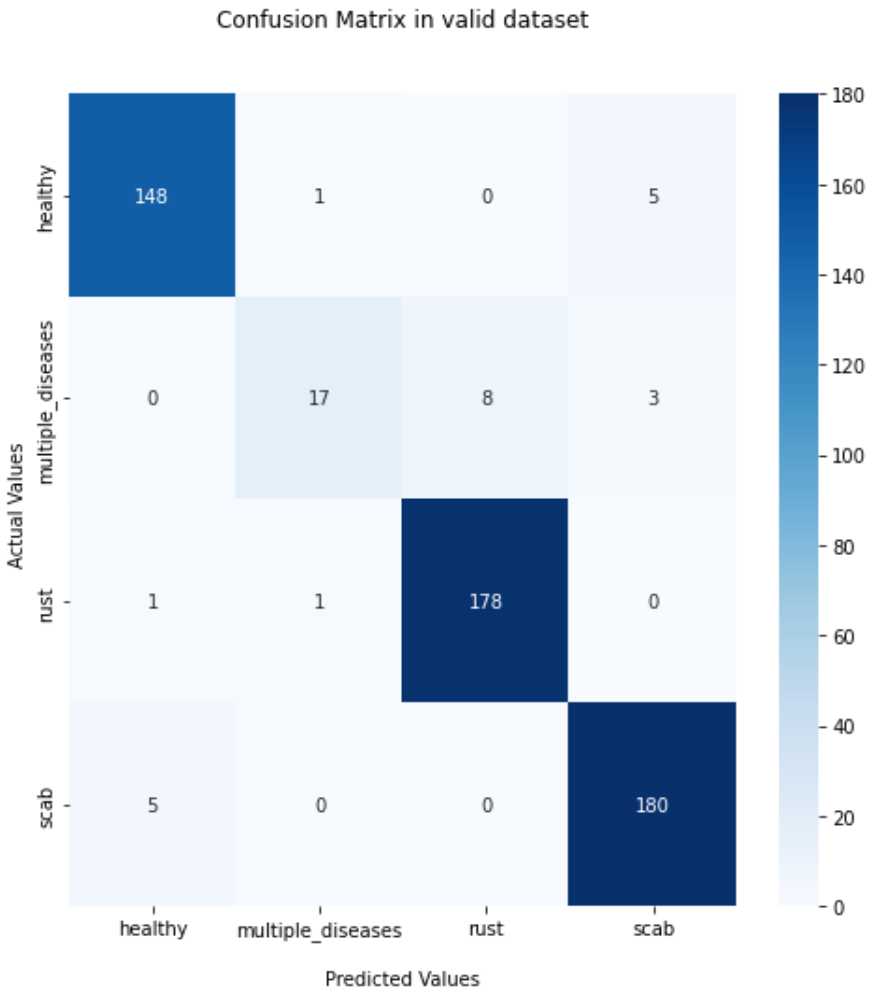
- After running throw multiple models (adjust layers, activation function, learning rate, ...), the final result of our best model:
    - On training set:



	precision	recall	f1-score	support
healthy	0.94	0.95	0.95	362
multiple_diseases	0.93	0.62	0.74	63
rust	0.97	1.00	0.98	442
scab	0.95	0.95	0.95	407
accuracy			0.95	1274
macro avg	0.95	0.88	0.90	1274
weighted avg	0.95	0.95	0.95	1274

Classification report on training set

- On validation set:



	precision	recall	f1-score	support
healthy	0.96	0.96	0.96	154
multiple_diseases	0.89	0.61	0.72	28
rust	0.96	0.99	0.97	180
scab	0.96	0.97	0.97	185
accuracy			0.96	547
macro avg	0.94	0.88	0.91	547
weighted avg	0.96	0.96	0.95	547

Classification report on validation set

5.2. Commenting on the result:

- Our best model reached a very high accuracy on both training set and validation set, so this model is not overfitting.
- Because the imbalance of dataset mentioned above, the recall of multiple_diseases class is quite low compared to others. This means if the input plant is in fact has multiple diseases, the model may have a high chance to predict it to be just one disease (rust, scab) or even healthy.

6. References:

Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names. (2018, May 23). Github.Com. https://gombru.github.io/2018/05/23/cross_entropy_loss/

CS 229 - Deep Learning Cheatsheet. (n.d.). Https://Stanford.Edu/. https://stanford.edu/%7Eshervine/teaching/cs-229/cheatsheet-deep-learning#nn

CS 230 - Convolutional Neural Networks Cheatsheet. (n.d.). Stanford.Edu. https://stanford.edu/%7Eshervine/teaching/cs-230/cheatsheet-convolutional-neural-networks