TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
ĐẠI HỌC QUỐC GIA TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN
-------***-------

# HW02: RECOGNITION BY NEURON NETWORK

**Môn học: Nhận Dạng**

**Giảng viên: Lê Hoàng Thái**
**Giảng viên: Lê Thanh Phong**
**Giảng viên: Nguyễn Ngọc Thảo**

*TP.HCM, ngày 03 tháng 04 năm 2022*

# Group Members

| **Student ID** | **Full Name** | **Assignment** |
|---|---|---|
| 19127135 | Phạm Bảo Hân | Inception-v3 |
| 19127078 | Nguyễn Đỗ Thanh Trúc | VGG16, DenseNet121 |
| 19127651 | Trần Anh Túc | AlexNet |

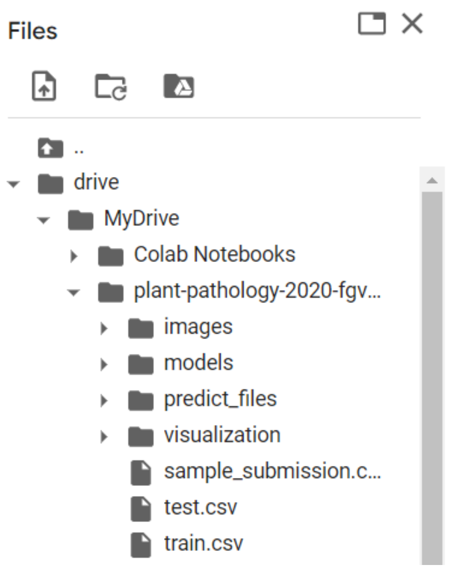
# Table of contents

## 1. Link:

- Drive (model, visualization, predict): [https://drive.google.com/drive/folders/1k5Y61GJeZJTlG2d78HTnWB2tcYZL9fPf?usp=sharing](https://drive.google.com/drive/folders/1k5Y61GJeZJTlG2d78HTnWB2tcYZL9fPf?usp=sharing)

## 2. Instruction for file code:

- Our code is used for Google Colab so all the file path direct to Google Drive folder which is the **plant-pathology-2020-fgvc7** dataset.
- The image below shows the example of folder structure for when running code in Google Colab:



- plant-pathology-2020-fgvc7 folder comprises:
  - images: folder contains images of training set and test set.
  - models: folder contains models of 4 CNN architectures.
  - predict_files: folder contains the prediction result of 4 models.
  - visualization: folder contains the images visualize the loss and accuracy through each epoch for 4 models.
  - sample_submission.csv: sample for submission.
  - test.csv: test set.
  - train.csv: training set.

## 3. Data analysis

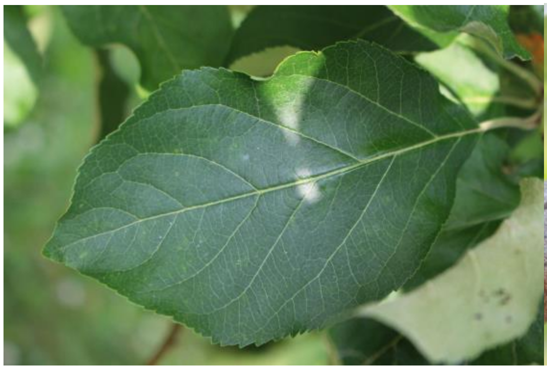
Given plant-pathology-2020-fgvc7 dataset includes:

- train.csv: information about classification of training images set
- test.csv: list file name of all images needing to be categorized
- sample_submission.csv: sample submission.
- images folder:
  - Test_id.jpg : 1821 images

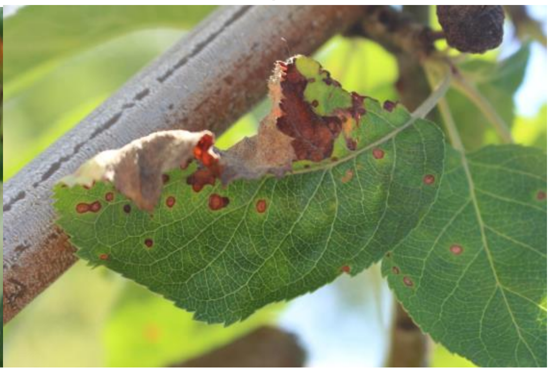
- Train_id.jpg : 1821 images

Pictures of 4 states of leaf: *healthy, multiple diseases, rust, scab.*

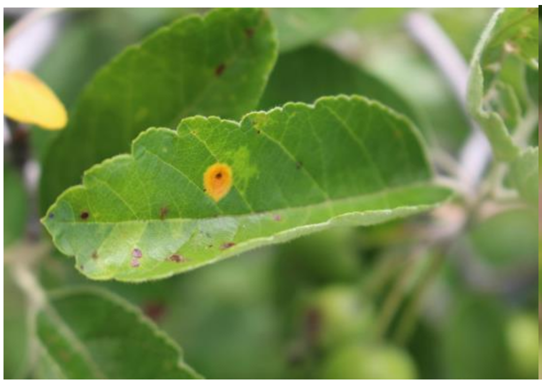
Healthy leaf


Train_704.jpg

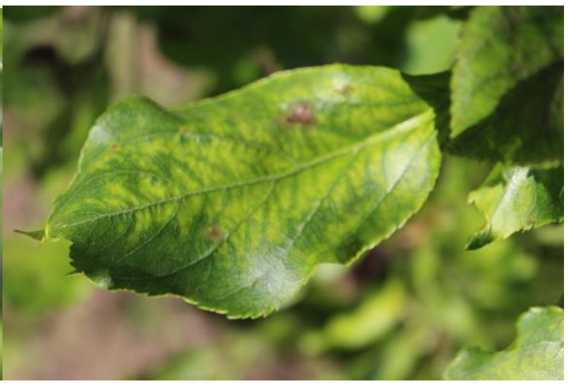
Multiple diseases


Train_02.jpg

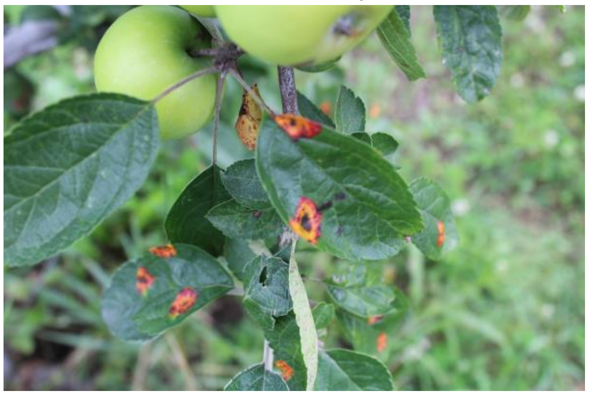
Rust


Train_508.jpg

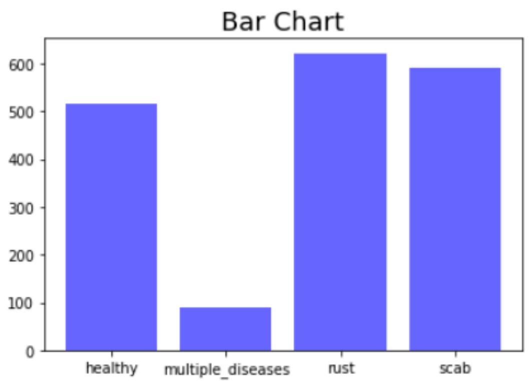
Scab

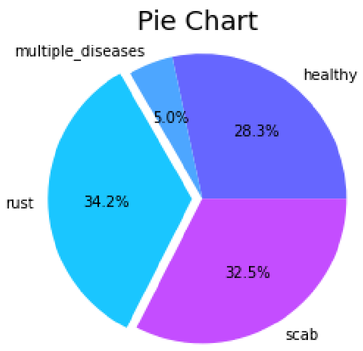

Train_0.jpg

- The area occupied by the leaf in images is different in each image.
- The light conditions are different in each image.
- Some images are confusing:
    - The area of the leaf is small.
    - Many leaves with different status in one picture.


Test_167.jpg

Training dataset has the proportion of training images as below:





- While rust, healthy and scab have the equivalent proportion in the training dataset, the multiple diseases account for a minor amount.

# 4. Model Architecture and Implementation

## 2.1 Model based on VGG16:

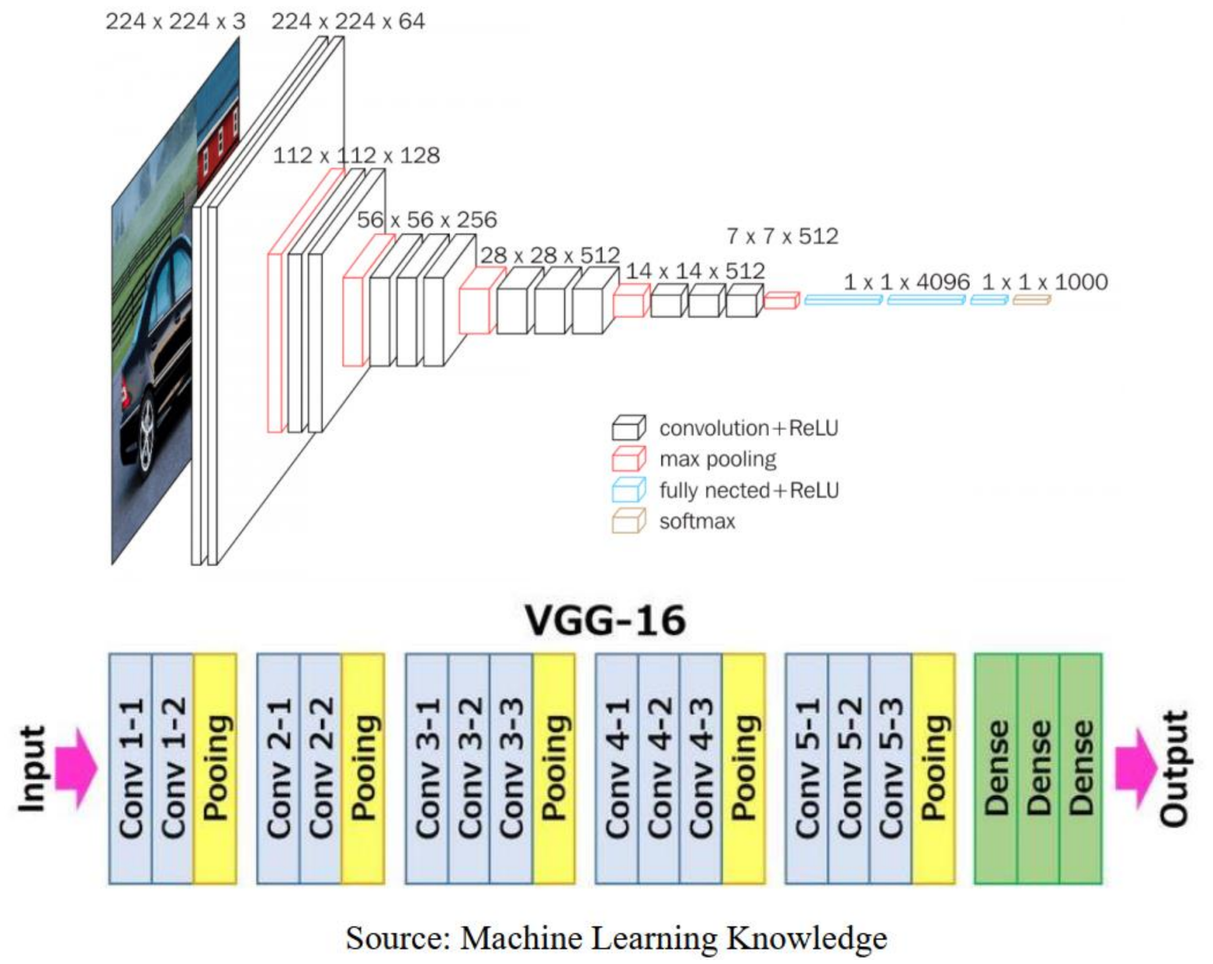
### Overview

- VGG16 Network is the simple convolutional neural network (CNN) architecture which is easy to understand and implement.
- K. Simonyan  et al. released VGG16 Network from paper“Very Deep Convolutional Networks for Large-Scale Image Recognition” **[1]** in 2014 by using small 3x3 convolutional filter to reduce significantly trainable parameters.

### Why is VGG16

- Easy to implement and custom.
- Extracting features by using small 3x3 convolutional filter.

## VGG16 Architecture



*VGG16 architecture*

- VGG16 contains 6 blocks with layers such as: Convolutional, Pooling, Fully connected layers with 16 weight layers in depth.
- The input image size is 224x224x3.
- All convolutional layers use 3x3 kernel, activate function ReLU, padding same to make the output same size as input
- Pooling layers use max-pooling with 2x2 filter and stride = 2. Therefore, after pooling layers, the image size is halved. The data matrix will be flatten (at the final pooling layer) to vector which is passed through Fully connected layers.
- The final block includes 3 fully connected layers. The first two layers contain 4096 units and ReLU activate function while the other one has 4 unit to produce probabilities of 4 classes.

## Model summary

```
Total params: 134,276,932
Trainable params: 134,276,932
Non-trainable params: 0
```

## 2.2 Model based on DenseNet121:

### Overview

- Dense Convolutional Network (DenseNet) is the architecture proposed by Gao Huang et al. released DenseNet from paper "Densely Connected Convolutional Networks" **[2]** in 2018.
- DenseNet can make the network deeper while preserving the compact and efficiency of model by connecting all layers with each other in a feed-forward fashion to leverage maximum information of all preceeding layers.

→ Take advantages of feature reuse throughout the network so it leads to a more compact model.

- DenseNet121 is the Dense Convolutional Network with has 121 trainable layers (exclude batch normalize layers).

### Why is DenseNet121

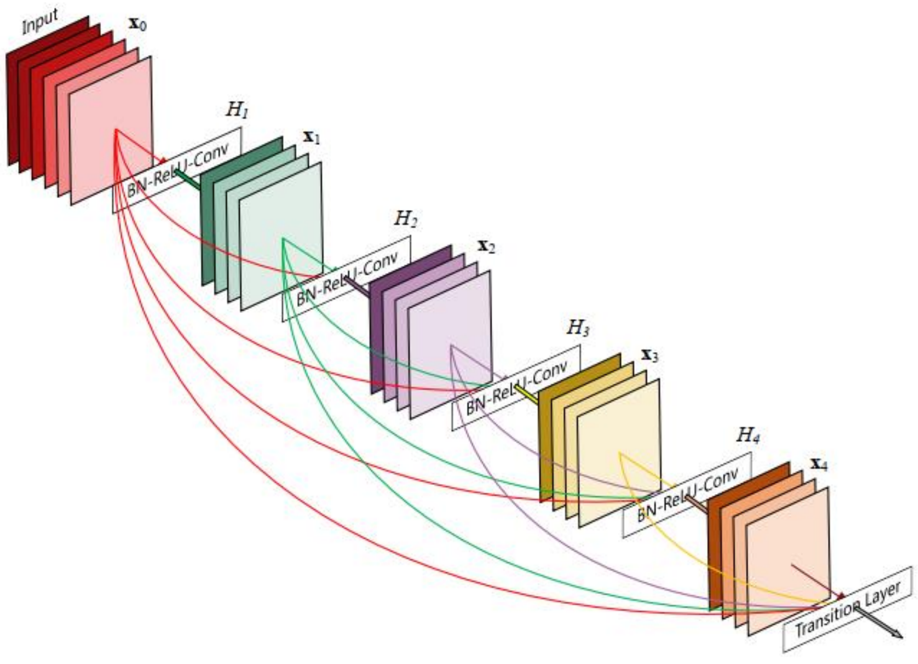
- Alleviate the vanishing-gradient problem
- Strengthen feature propagation and encourage feature reuse

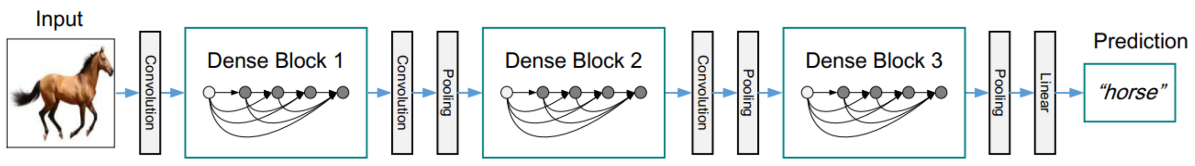
→ Reduce the number of parameters → More compact models → Easy to train.

- Dense connections have a regularizing effect

→ Reduces overfitting on problems with smaller training dataset.

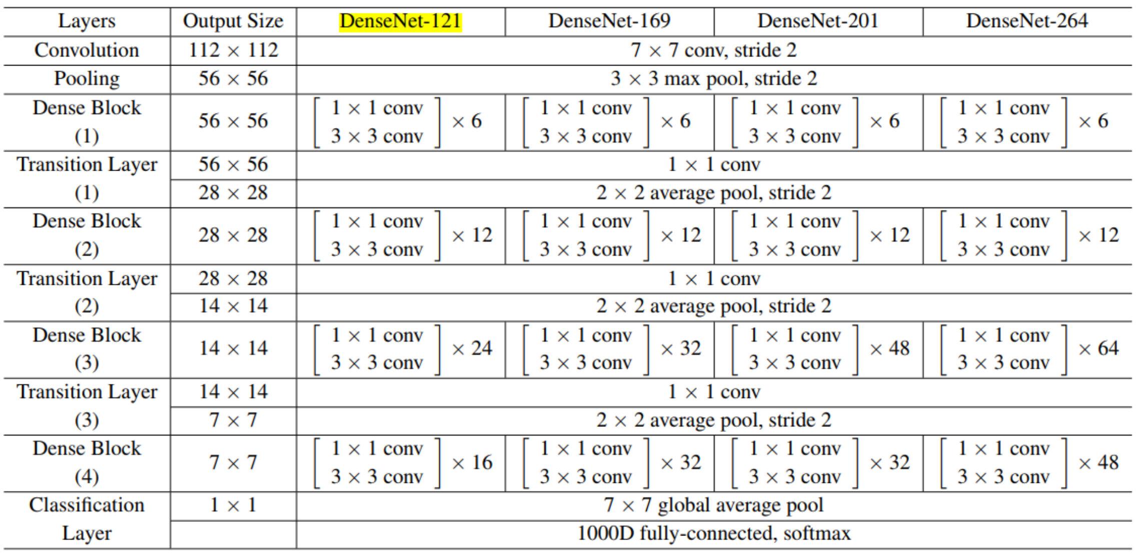
### DenseNet121 Architecture





*DenseNet121 architecture*

- Apart from the basic convolutional and pooling layers, DenseNet consists of two important blocks: Dense Blocks and the Transition layers.
- The input image size is 224x224x3.



| Layers | Output Size | DenseNet-121 | DenseNet-169 | DenseNet-201 | DenseNet-264 |
|---|---|---|---|---|---|
| Convolution | $112 \times 112$ | $7 \times 7$ conv, stride 2 | | | |
| Pooling | $56 \times 56$ | $3 \times 3$ max pool, stride 2 | | | |
| Dense Block (1) | $56 \times 56$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$ |
| Transition Layer (1) | $56 \times 56$ | $1 \times 1$ conv | | | |
| | $28 \times 28$ | $2 \times 2$ average pool, stride 2 | | | |
| Dense Block (2) | $28 \times 28$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$ |
| Transition Layer (2) | $28 \times 28$ | $1 \times 1$ conv | | | |
| | $14 \times 14$ | $2 \times 2$ average pool, stride 2 | | | |
| Dense Block (3) | $14 \times 14$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$ |
| Transition Layer (3) | $14 \times 14$ | $1 \times 1$ conv | | | |
| | $7 \times 7$ | $2 \times 2$ average pool, stride 2 | | | |
| Dense Block (4) | $7 \times 7$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$ |
| Classification Layer | $1 \times 1$ | $7 \times 7$ global average pool | | | |
| | | 1000D fully-connected, softmax | | | |

*Detail architecture of DenseNet121*

- Note that: Every dense blocks have two convolutions (1x1 and 3x3 sized kernels) which are repeated 6,12,24,16 times respectively.

### Model summary

```
Total params: 7,333,316
Trainable params: 7,249,668
Non-trainable params: 83,648
```

## 2.3 Model based on Inception:

### Overview

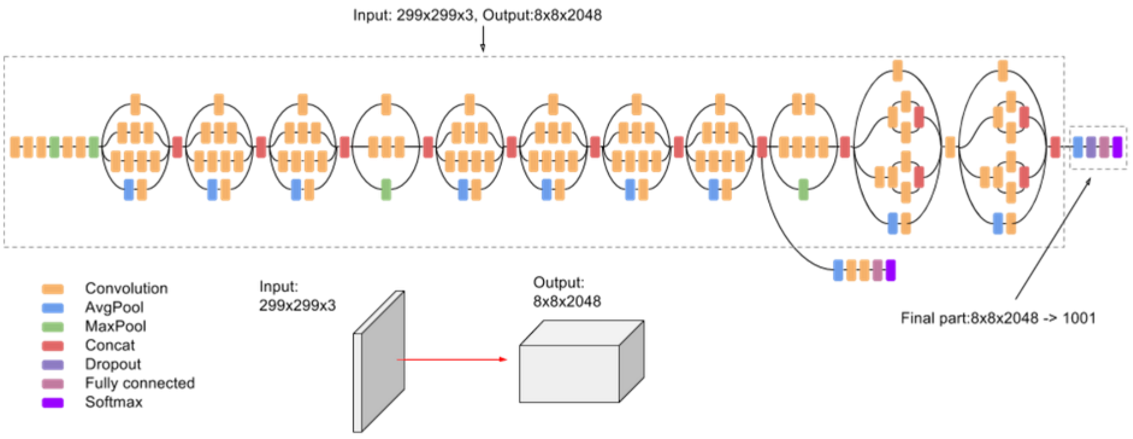
- Inception Network is the novel deep learning convolutional neural network (CNN) architecture.
- Min Lin et al. released paper "Network in Network" **[3]** in 2014, which looked at increasing the representational capability of neural networks by implementing embedded internal complex structures within networks.
- Inception Network was the largest and most efficient deep convolutional neural network at that time.

### Why is Inception

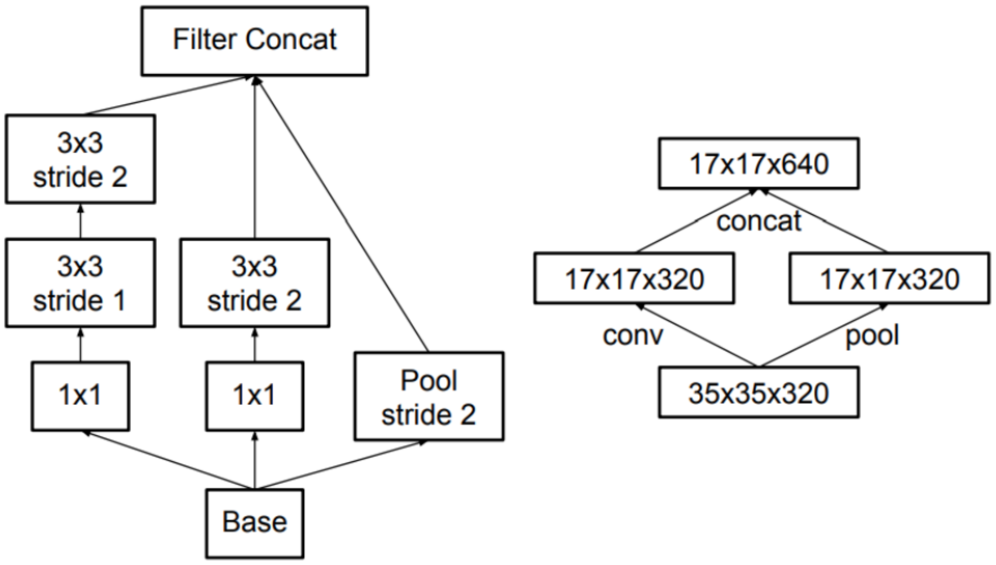
- Large deep convolutional neural network with high performance.
- Extracting features at varying scales.

- Low expense in computation.
- Low error rate.

## Inception-v3 Architecture



- Inception-v3 Network has 42 layers comprising convolution, pooling, dropout, fully connected, concatenation, softmax.
- The network receives a 299x299x3 input and outputs an 8x8x2048 result. To be suitable for the Plant Pathology problem, we add some dense layers at the end to convert the output to the softmax 1x4 output.
- Inception Network uses different convolutional filter sizes, helps to extract features from input data at various scales.
    - 1x1 conv reduces the dimensions of inputs within the network and Learns pattern across the channels of an image
    - 3x3 and 5x5 learns spatial patterns throughout the input's three dimensions (height, width, and depth).



*Inception modules*

- To increase the efficiency and reduce the cost, the Inception-v3 uses some techniques like the factorized convolution, asymmetric convolution, auxiliary classifier, grid size reduction and replacing the large conv with the smaller conv.

### Model summary

```
Total params: 22,073,572
Trainable params: 22,039,140
Non-trainable params: 34,432
```

## 2.4 Model based on AlexNet

### Overview

- AlexNet is the name of a convolutional neural network (CNN) architecture, designed by Alex Krizhevsky in collaboration with Ilya Sutskever and Geoffrey Hinton, finished in 2012
- The original paper's primary result was that the depth of the model was essential for its high performance, which was computationally expensive, but made feasible due to the utilization of graphics processing units (GPUs) during training.
- Working on Imagenet scale, AlexNet is considered easy to implement with only 8 learned layers in total with high performance and not long later, multi improvements over AlexNet results in models such as VGG, GoogleNet, and lately ResNet.

*(Wikipedia)*

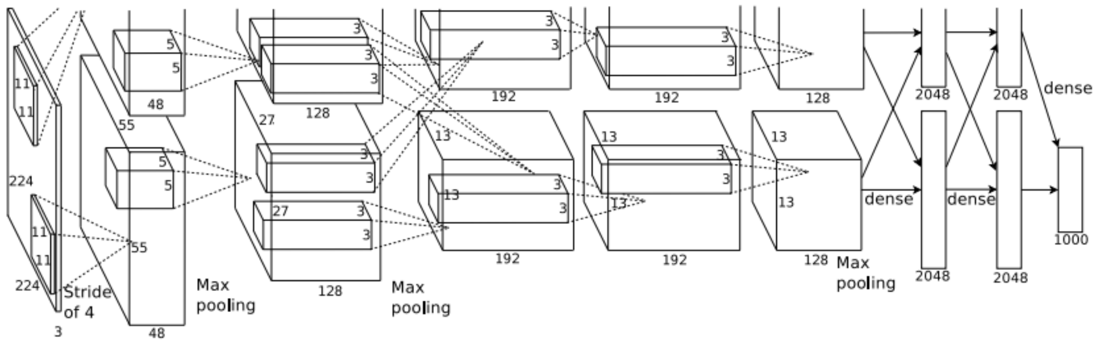
### Why is AlexNet

- AlexNet is considered one of the most influential papers published in computer vision, having spurred many more papers published employing CNNs and GPUs to accelerate deep learning.

*(Wikipedia)*

- AlexNet performs excellently in handling with high resolution input and very challenging dataset, also allows multi-GPUs training.
- To make training faster, the author used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers the author employed a recently-developed regularization method called “dropout” that proved to be very effective **[4]**.

## Model Architecture



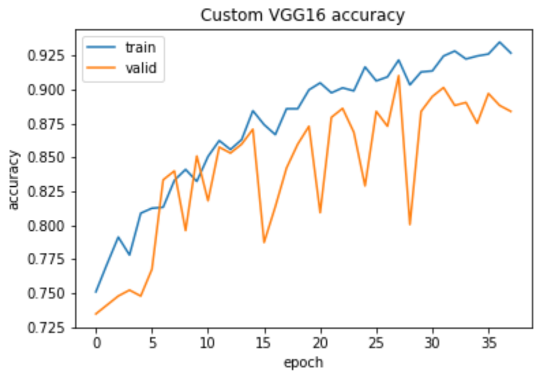
- AlexNet architecture consists of 5 convolutional layers, 3 max-pooling layers, 2 normalization layers, 2 fully connected layers, and 1 softmax layer.
- It contains eight learned layers: five convolutional and three fully-connected.
- The first convolutional layer filters the 224×224×3 input image with 96 kernels of size 11×11×3 with a stride of 4 pixels (this is the distance between the receptive field centers of neighboring neurons in a kernel map). The second convolutional layer takes as input the (response-normalized and pooled) output of the first convolutional layer and filters it with 256 kernels of size 5 × 5 × 48. The third, fourth, and fifth convolutional layers are connected to one another without any intervening pooling or normalization layers. The third convolutional layer has 384 kernels of size 3 × 3 ×256 connected to the (normalized, pooled) outputs of the second convolutional layer. The fourth convolutional layer has 384 kernels of size 3 × 3 × 192 , and the fifth convolutional layer has 256 kernels of size 3 × 3 × 192. The fully-connected layers have 4096 neurons each.
- In the experiment Input is images of 227x227x3

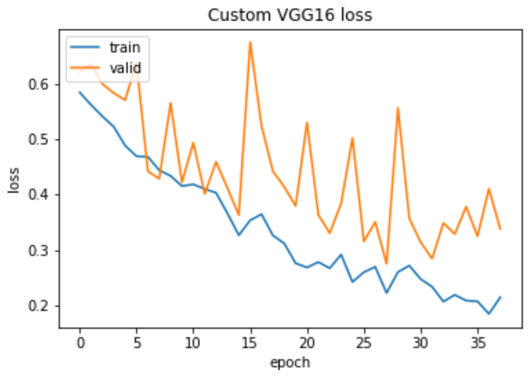
## Model summary

```
Total params: 58,303,236
Trainable params: 58,300,484
Non-trainable params: 2,752
```

# 5. Experiments and Results

## 3.1 Model based on VGG16

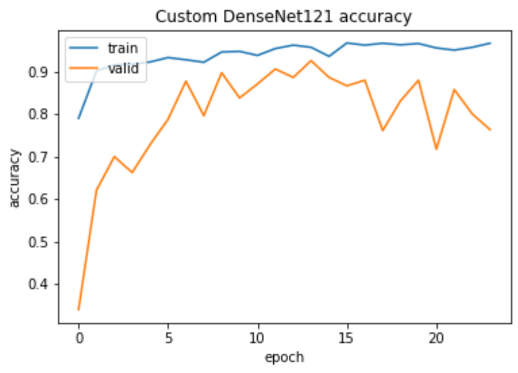


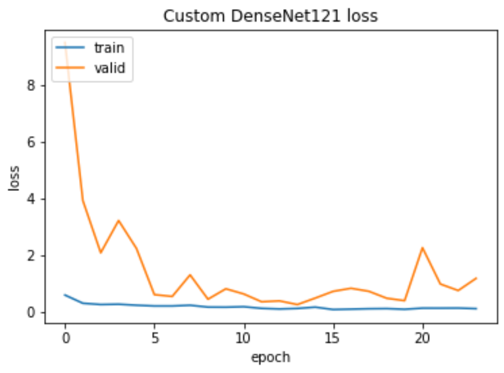


*The VGG16's accuracy and loss in training set and validation set*

- According to the accuracy and loss after training n epochs, we can see that the accuracy increases quite fast but it needs to be trained over multiple epochs.
  → Easily undergo the overfitting case due to the small dataset.
- Additionally, the weight of VGG16 is large (approximately 1 GB)
  → Take up more disk space and bandwidth that makes it inefficient.

## 3.2 Model based on DenseNet121

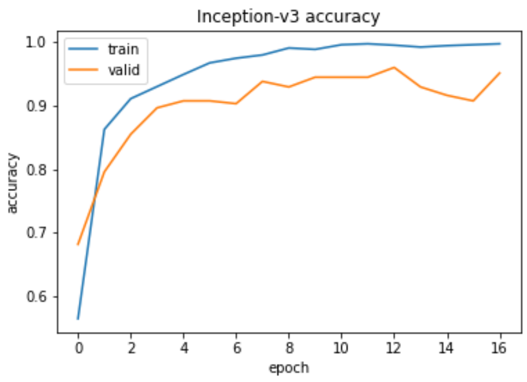


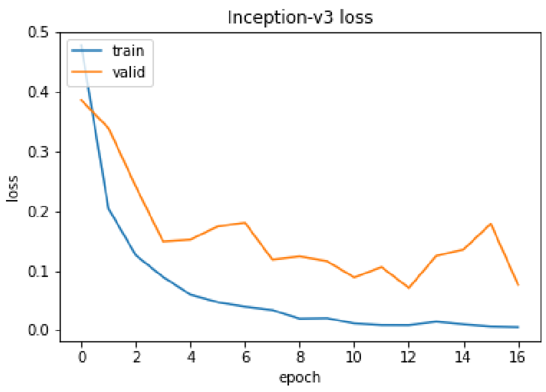


*The DenseNet121's accuracy and loss in training set and validation set*

- DenseNet121 needs fewer epochs for training than VGG16 but the accuracy and loss is efficient.
- Additionally, the weight of DenseNet121 is the smallest among all architectures.
  → Compact model, easy to train and custom further.

## 3.3 Model based on Inception-v3





*The Inception's accuracy and loss in training set and validation set*

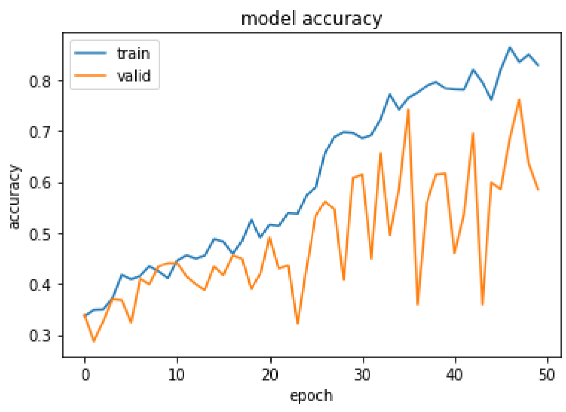
- Inception-v3 takes a small number of epochs to reach > 90% accuracy in both training set and valid set.
- The accuracy lines are  quite smooth.

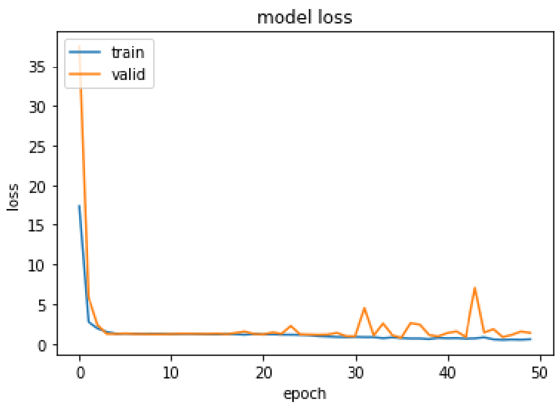
## 3.4 Model based on AlexNet

- 227x227 images data set is expected to be used in the experiments
- Training model over 50 epoches for 32 images per batch

```
Epoch 41/50
43/43 [==============================] - 3s 60ms/step - loss: 0.7171 - accuracy: 0.7810 - val_loss: 1.3707 - val_accuracy: 0.4605
Epoch 42/50
43/43 [==============================] - 3s 61ms/step - loss: 0.7369 - accuracy: 0.7802 - val_loss: 1.5769 - val_accuracy: 0.5351
Epoch 43/50
43/43 [==============================] - 3s 61ms/step - loss: 0.6685 - accuracy: 0.8198 - val_loss: 0.8602 - val_accuracy: 0.6952
Epoch 44/50
43/43 [==============================] - 3s 60ms/step - loss: 0.7035 - accuracy: 0.7949 - val_loss: 7.0677 - val_accuracy: 0.3596
Epoch 45/50
43/43 [==============================] - 3s 60ms/step - loss: 0.8364 - accuracy: 0.7604 - val_loss: 1.4354 - val_accuracy: 0.5987
Epoch 46/50
43/43 [==============================] - 3s 60ms/step - loss: 0.5734 - accuracy: 0.8198 - val_loss: 1.8503 - val_accuracy: 0.5855
Epoch 47/50
43/43 [==============================] - 3s 61ms/step - loss: 0.5154 - accuracy: 0.8630 - val_loss: 0.8583 - val_accuracy: 0.6864
Epoch 48/50
43/43 [==============================] - 3s 61ms/step - loss: 0.5598 - accuracy: 0.8344 - val_loss: 1.1349 - val_accuracy: 0.7610
Epoch 49/50
43/43 [==============================] - 3s 60ms/step - loss: 0.5333 - accuracy: 0.8491 - val_loss: 1.5591 - val_accuracy: 0.6360
Epoch 50/50
43/43 [==============================] - 3s 60ms/step - loss: 0.5949 - accuracy: 0.8286 - val_loss: 1.3903 - val_accuracy: 0.5855
```

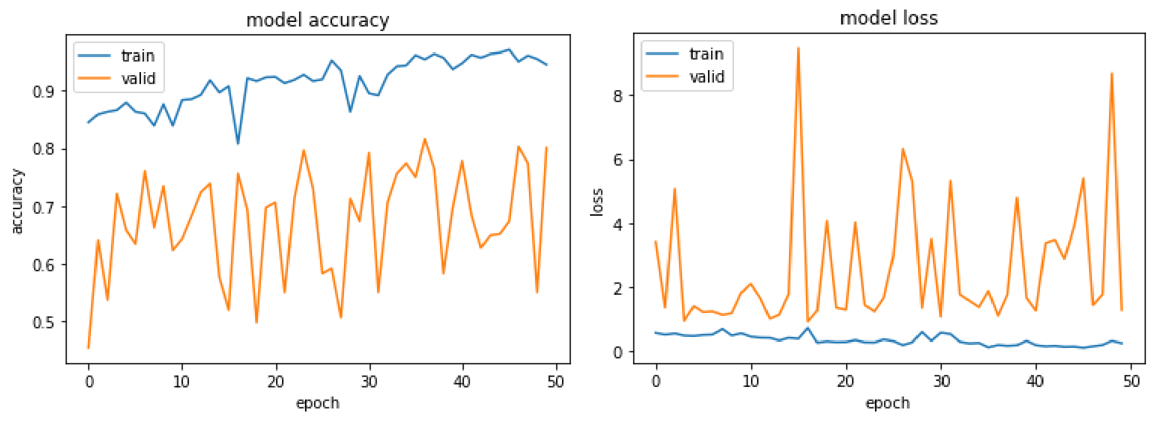
Model accuracy and loss





- According to the results, it's not too bad or good model training on a set of 1821 items, the highest accuracy on the training set is about 86% on epoch 47 in comparison with those on validation set ,76% on epoch 48.
- Even though there are points that are absurd, the accuracy on both datasets went upward.
- The lack of resources for training may be the reason for poor performance, because we witnessed that there is a growth of accuracy past time despite the fact that accuracy on validation set is starting to fall, which indicates the appearance of overfitting.
- In the poor condition of experiment, after shuffling the second time, result showed the effects of overfitting

Model accuracy and loss 2nd



## Reference

[1] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[2] Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, Densely Connected Convolutional Networks

[3] Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." arXiv preprint arXiv:1312.4400 (2013).

[4] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, 2012, ImageNet Classification with Deep Convolutional Neural Networks, ImageNet Classification with Deep Convolutional Neural Networks (neurips.cc)