# MSBD 6000B Individual Project 2

KWAN Hok Ming 20386486

## Introduction

There are three set of photos with 5 classes for this project: train, val and test. The training data contains 2569 pictures, the validation set contains 550 pictures while the testing data contains 551 pictures.

## Task

Different Convolutional Neural Networks architectures are utilized for classifying 5 different types of flowers – daisy, dandelion, rose, sunflower and tulip, and the accuracies of models are tried to improve.

## Data Processing

Due to different images size for different architectures, VGG16, VGG19, and ResNet all accept 224×224 input images while Inception V3 and Xception require 299×299 pixel inputs. The training images, validation images and testing images are changed to an float32 array and normalized by dividing 255 before training the models. Image data generator is used for generating batches of tensor image data with real-time data augmentation. The 5 different classes are turned to dummy variables.

## Architectures

- **Self-created CNN**
  Below is the summary of a CNN which is created by me.
  Convolutional input layer, 32 feature maps with a size of 3×3 and a rectifier activation function.
  Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function.
  Max Pool layer with size 2×2.
  Dropout layer at 25%.
  Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function.
  Convolutional layer, 128 feature maps with a size of 3×3 and a rectifier activation function.
  Max Pool layer with size 2×2.
  Dropout layer at 25%.
  Convolutional layer, 128 feature maps with a size of 3×3 and a rectifier activation function.
  Convolutional layer,256 feature maps with a size of 3×3 and a rectifier activation function.
  Max Pool layer with size 2×2.
  Dropout layer at 25%.
  Flatten layer.
  Fully connected layer with 1000 units and a rectifier activation function.
  Dropout layer at 50%.

Fully connected output layer with 5 units and a softmax activation function. RMSprop optimizer is utilized for this architecture. The accuracy is about 89% and the architecture is shown as figure 1.

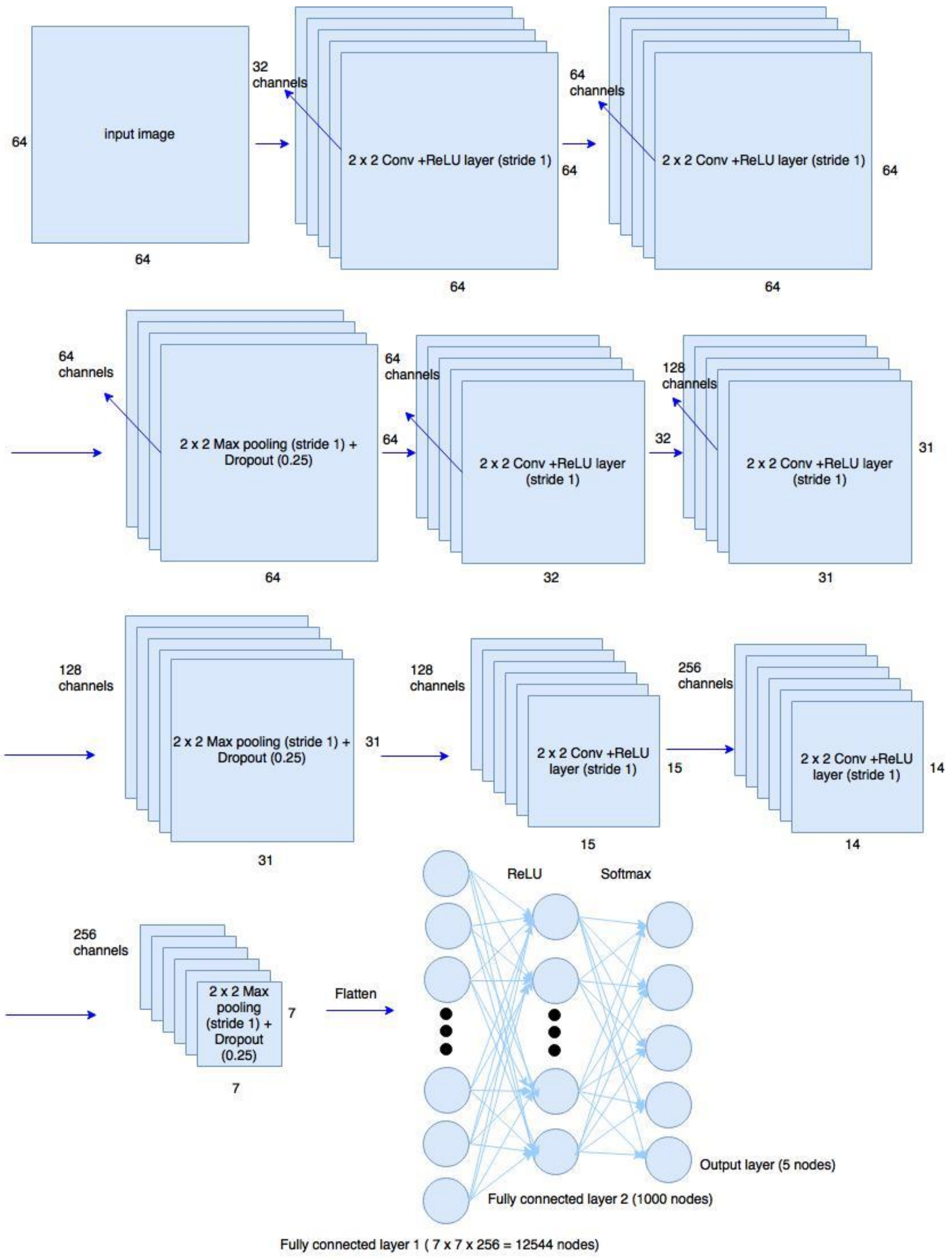
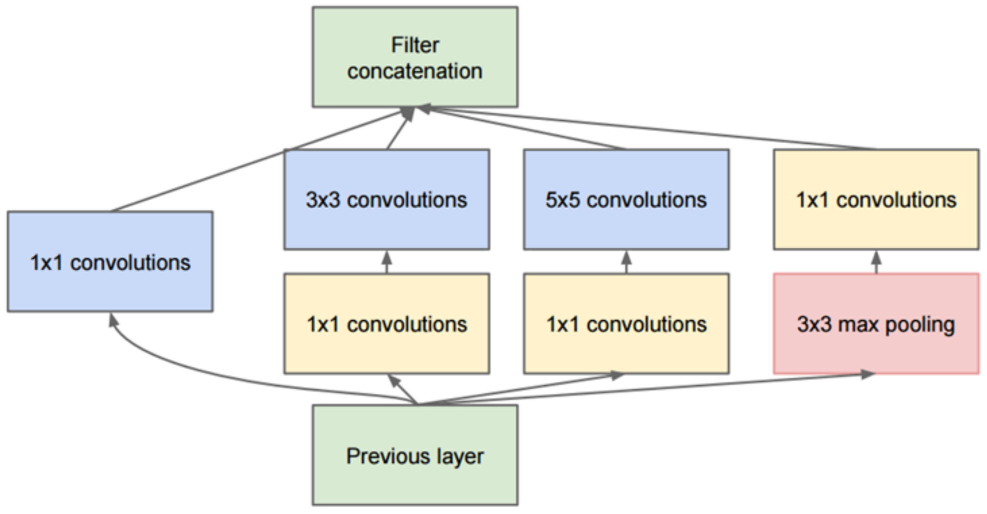

input image
64
64
32 channels
2 x 2 Conv +ReLU layer (stride 1)
64
64
64 channels
2 x 2 Conv +ReLU layer (stride 1)
64
64
64 channels
2 x 2 Max pooling (stride 1) + Dropout (0.25)
64
64
64 channels
2 x 2 Conv +ReLU layer (stride 1)
32
32
128 channels
2 x 2 Conv +ReLU layer (stride 1)
31
31
128 channels
2 x 2 Max pooling (stride 1) + Dropout (0.25)
31
31
128 channels
2 x 2 Conv +ReLU layer (stride 1)
15
15
256 channels
2 x 2 Conv +ReLU layer (stride 1)
14
14
256 channels
2 x 2 Max pooling (stride 1) + Dropout (0.25)
7
7
Flatten
ReLU
Softmax
Output layer (5 nodes)
Fully connected layer 2 (1000 nodes)
Fully connected layer 1 ( 7 x 7 x 256 = 12544 nodes)

Figure 1: The self-made CNN structure.

- **Inception V3**

  The Inception V3 micro-architecture was first introduced in 2014 [1]. The inception module is tried to act as a multi-level feature extractor by computing 1×1, 3×3, and 5×5 convolutions within the same module of the network. The output of these filters are then stacked along the channel dimension and before being fed into the next layer in the network. The architecture is shown as figure 2.



Full Inception module

Figure 2: The full inception module structure.

- **Xception**

The Xception was first introduced in 2017 [2]. The architecture of Xception is an extension and improvement of the Inception which replaces the standard Inception modules with depth-wise separable convolutions and the Xception has smaller weight than Inception V3. The architecture has 36 convolutional stages (Figure 3).

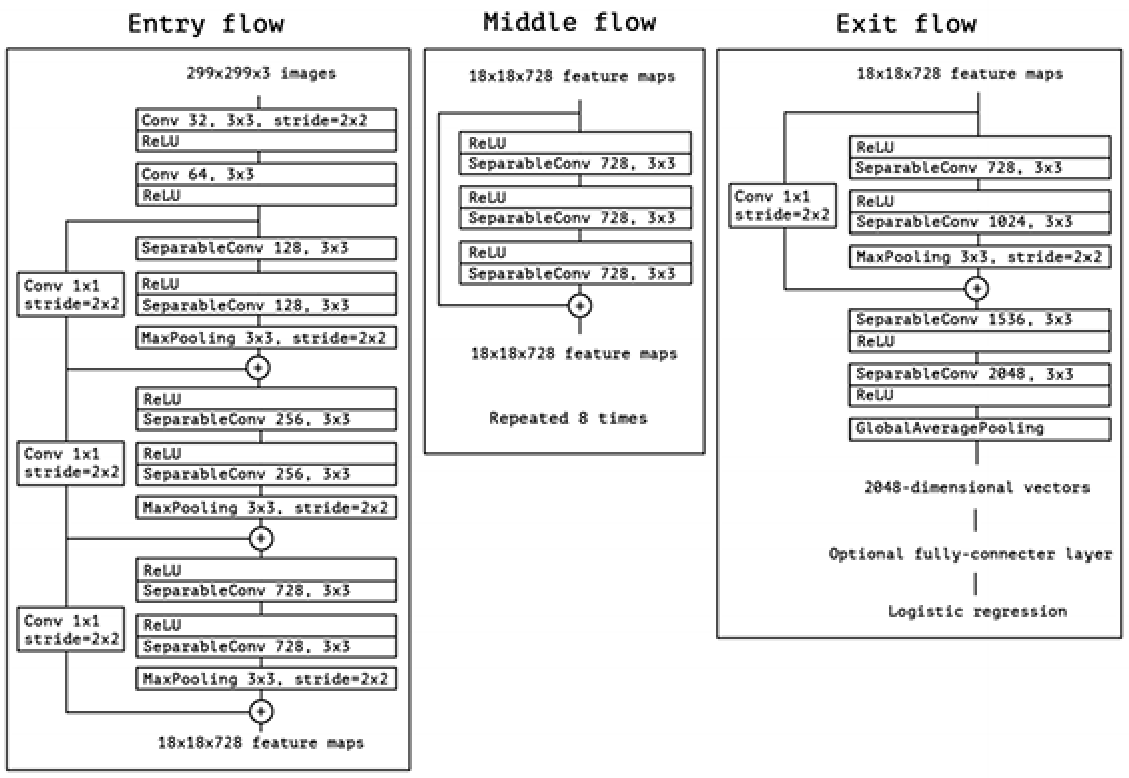


Figure 3: The architecture of Xception.

- **VGG 16**

The VGG network architecture was introduced by Simonyan and Zisserman in their 2014[3]. The macro-architecture of VGG16 can be seen in Figure 4. The network uses only 3×3 convolutional layers stacked on top of each other in increasing depth and utilize max pooling for reducing volume size. After that, two fully-connected layers of which each with 4,096 nodes are then followed by a softmax classifier.

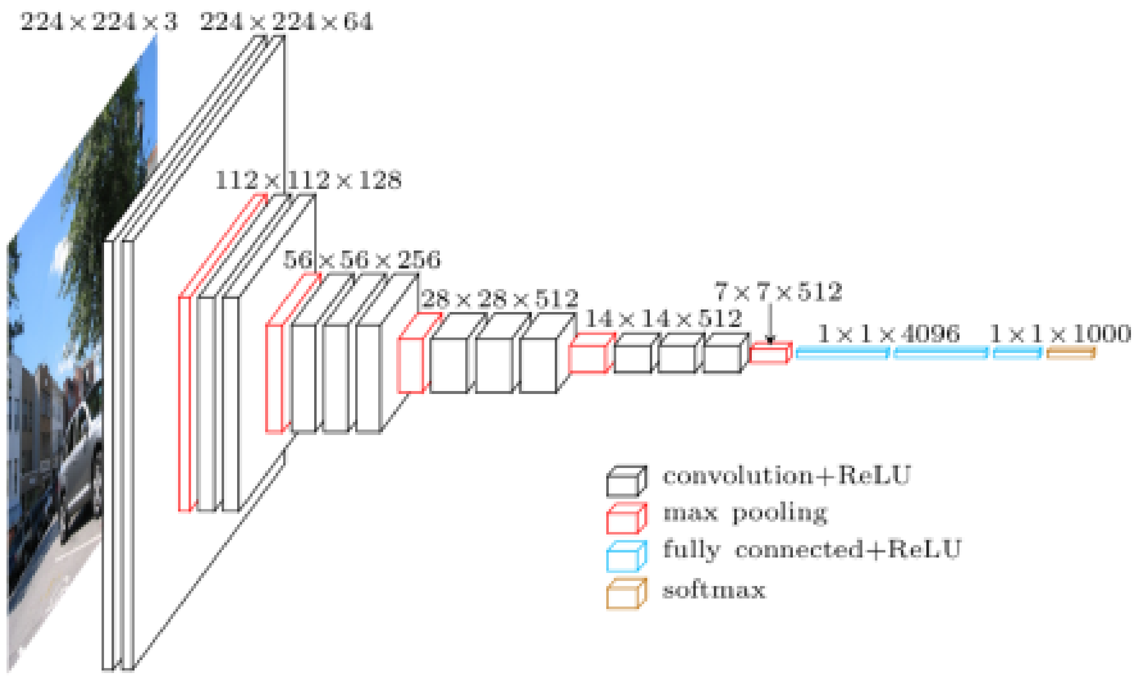
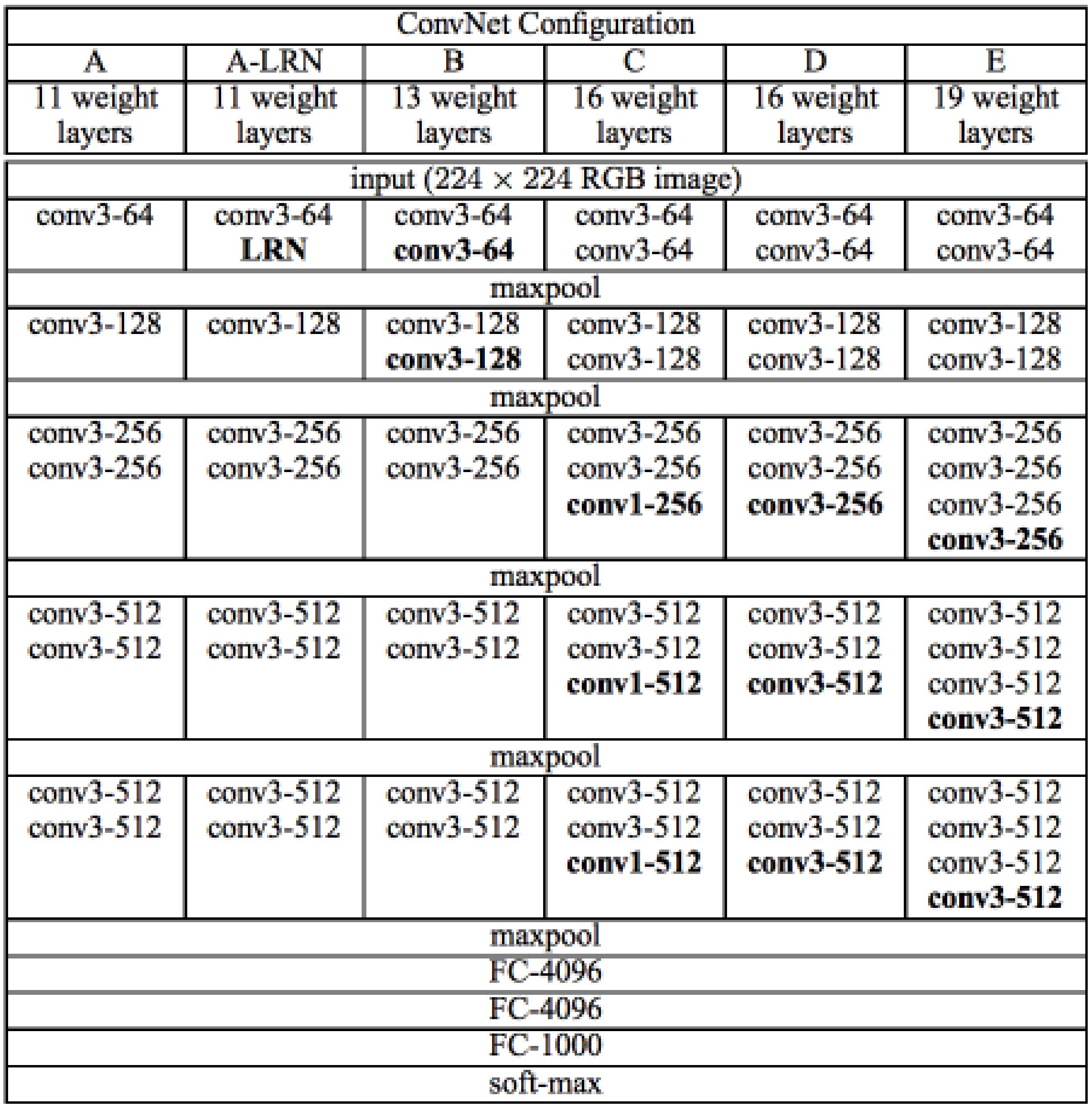


Figure 4: The architecture of VGG16.

- **VGG 19**

The difference between VGG16 and VGG19 is the number of weight layers in the network – columns D and E (Figure 5).



| ConvNet Configuration | | | | | |
|---|---|---|---|---|---|
| A | A-LRN | B | C | D | E |
| 11 weight layers | 11 weight layers | 13 weight layers | 16 weight layers | 16 weight layers | 19 weight layers |
| input (224 × 224 RGB image) | | | | | |
| conv3-64 | conv3-64 | conv3-64 | conv3-64 | conv3-64 | conv3-64 |
| | **LRN** | **conv3-64** | conv3-64 | conv3-64 | conv3-64 |
| maxpool | | | | | |
| conv3-128 | conv3-128 | conv3-128 | conv3-128 | conv3-128 | conv3-128 |
| | | **conv3-128** | conv3-128 | conv3-128 | conv3-128 |
| maxpool | | | | | |
| conv3-256 | conv3-256 | conv3-256 | conv3-256 | conv3-256 | conv3-256 |
| conv3-256 | conv3-256 | conv3-256 | conv3-256 | conv3-256 | conv3-256 |
| | | | **conv1-256** | **conv3-256** | conv3-256 |
| | | | | | **conv3-256** |
| maxpool | | | | | |
| conv3-512 | conv3-512 | conv3-512 | conv3-512 | conv3-512 | conv3-512 |
| conv3-512 | conv3-512 | conv3-512 | conv3-512 | conv3-512 | conv3-512 |
| | | | **conv1-512** | **conv3-512** | conv3-512 |
| | | | | | **conv3-512** |
| maxpool | | | | | |
| conv3-512 | conv3-512 | conv3-512 | conv3-512 | conv3-512 | conv3-512 |
| conv3-512 | conv3-512 | conv3-512 | conv3-512 | conv3-512 | conv3-512 |
| | | | **conv1-512** | **conv3-512** | conv3-512 |
| | | | | | **conv3-512** |
| maxpool | | | | | |
| FC-4096 | | | | | |
| FC-4096 | | | | | |
| FC-1000 | | | | | |
| soft-max | | | | | |

Figure 5: Table 1 of Very Deep Convolutional Networks for Large Scale Image Recognition, Simonyan and Zisserman (2014).

- **ResNet**

The ResNet network architecture was introduced by in 2015[4]. ResNet relies on micro-architecture modules which is also called "network-in-network architectures".
The micro-architecture is the set of "building blocks" used to construct the network. A collection of micro-architecture building blocks, for example CONV, POOL, leads to the macro-architecture (Figure 6).
ResNet is much deeper than VGG16 and VGG19 but the model size is significantly smaller because of the utilization of global average pooling rather than fully-connected layers.

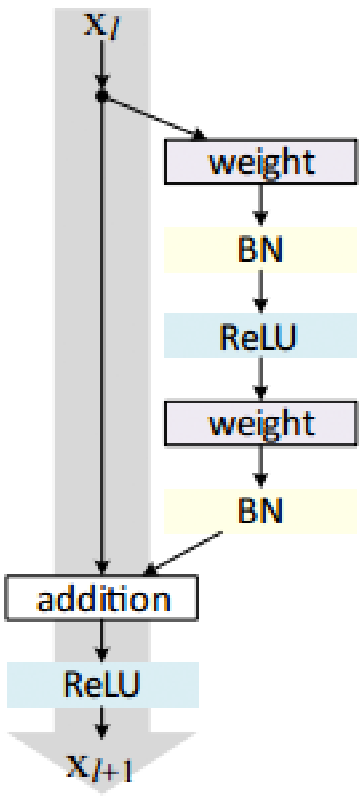


Figure 6: The residual module in ResNet.

## Validation

The validation set is used to evaluate the performance of different architectures. Below is the accuracy of validation set by each different architecture.

| Architecture | Validation Set Accuracy |
|---|---|
| Self-created CNN | 0.8927271355 |
| Inception V3 | 0.942181786 |
| Xception | 0.962909062 |
| VGG 16 | 0.94836364 |
| VGG 19 | 0.949454558 |
| ResNet | 0.956727246 |

## Prediction

After testing with different architectures, the Xeception architecture has the best validation accuracy, so it is used to predict the testing data. The result is saved as a csv file.

## Conclusion

Despite similar accuracy from different CNN architectures, Xception not only outperforms others but also needs much smaller time. The batch size and input image size can affect the accuracy, which is learnt from forming self-made CNN architecture. The hyper-parameters and architectures is an interesting direction to work toward.

## References:

[1] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich: Going Deeper with Convolutions, 2014.
[2] François Chollet, Xception: Deep Learning with Depthwise Separable Convolutions, 2016.
[3] Karen Simonyan, Andrew Zisserman: Very Deep Convolutional Networks for Large-Scale Image Recognition, 2014.
[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun: Deep Residual Learning for Image Recognition, 2015.