Plant Seedling Classification using Deep Convolution Neural Network

Tejas G. Mahale

April 30, 2018

Table of Content:

1. Abstract

Plants have an important role in the natural life circle. They are vital to almost every other form of life as plants are main source of food as well as almost all oxygen in the air that humans and other animals breathe is produced by plants.

Application of the benefits of modern computing technology to improve the efficiency of agricultural fields mainly constitutes of plant, is inevitable with growing concerns about increasing world population and limited food resources[2]. Hence Classifying plants helps at ensuring the protection and survival of all natural life. The process of plant classification can be performed using different ways, such as cellular and molecular biology as well as using the plants' leaves[1]. Implementing machine learning methods such as deep neural networks on agricultural data could be helpful for problems like classification of plant species based on their types. Classification of plants is one of important task because plant have important role in natural circle of life as they form largest part of living organism.

1.1 Goal :

- Aim of this project is to design plant seedling classifier using cascaded convolution neural network which can classify 12 different types of plant species.
- The proposed approach consist of three phases : reprocessing, feature extraction and classification.
- In this project, I am going to use Deep Convolutional Neural Network architecture to classify the type of plants from the plant seedling images collected from A Public Image Database for Benchmark of Plant Seedling Classification[3].

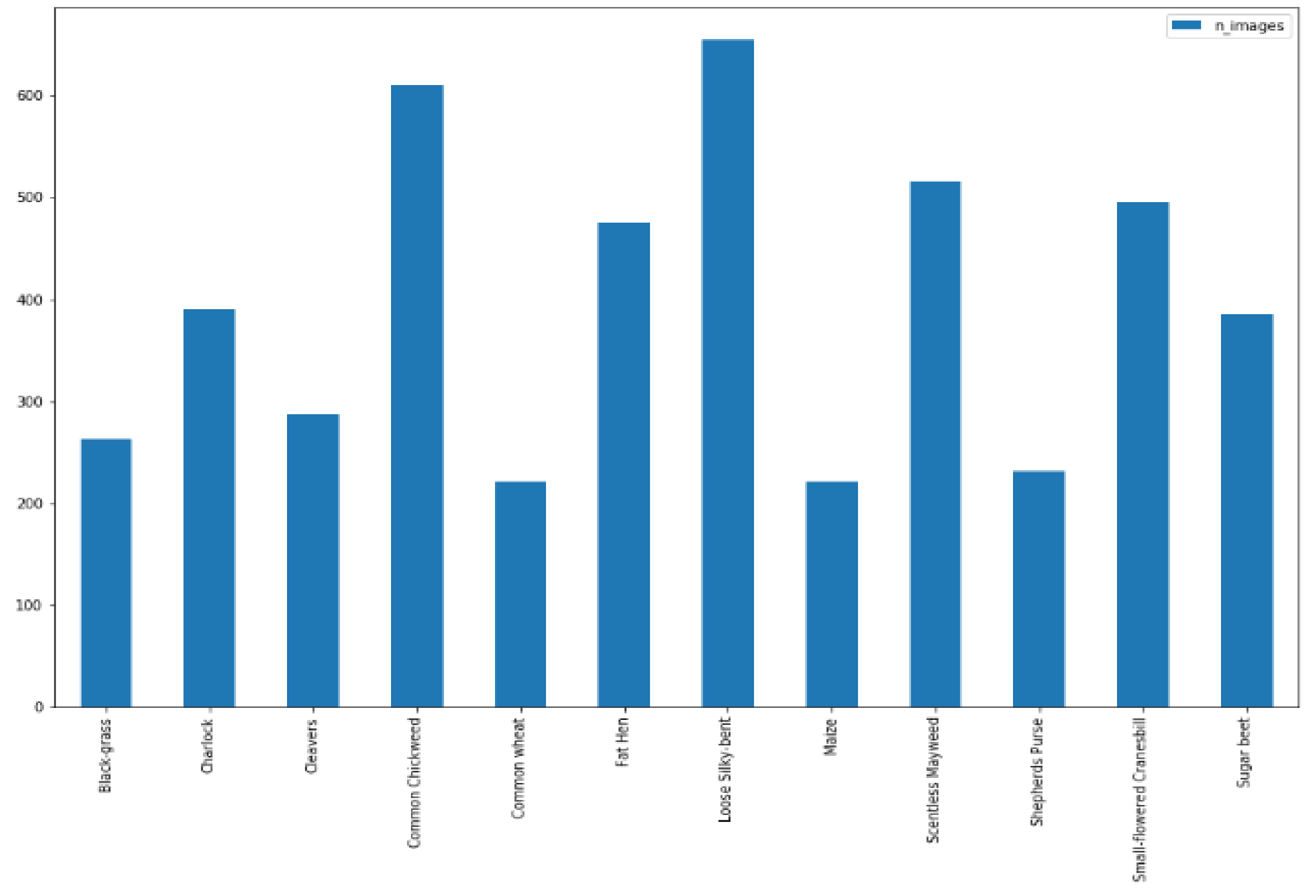
2. Dataset

- Plant seedling dataset given by Signal Processing group of Aarhus University[3] (available on Cornell website)
- Classes : Total Number of species (classes) = 12

Total Number of Images = 5554

Black-Grass	Char-lock	Cleavers	Common Chickweed	Common Wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed
310	452	335	713	253	538	766	257	607

Shepherd’s Purse	Sugar Beat
273	463



Class wise distribution Bar chart

2.1 Data Splitting on dataset:

I am going to split data randomly into train, validation and test sets according to following splitting:

Training data = 80 %

Validation data = 10%

Testing data =10%

Some of examples from data set:

Charlock

Black grass

Maize

Shepherd’s Purse

Data set link: https://vision.eng.au.dk/plant-seedlings-dataset/

choose Non-segmented dataset

Segmented datset by me :

https://drive.google.com/drive/folders/1uqv3yR9b0p6Ln8hNxapeRShAgxJEddc8?usp=sharing

choose file **image_Processed_1data.mat**

3 Related Work :

3.1 Related work on dataset area:

- There is no prior work on this data set as it was release on 16th November, 2017 ie 3 months ago. This dataset is relatively new and biggest dataset available for plant classification problem domain.
- Dataset was compiled by researchers from university of Aarhus University and University of Southern Denmark.
- Dataset can be downloaded from Cornell University Library with tag as " A Public Image Database for Benchmark of Plant Seedling Classification Algorithms"[3]

3.2 Related work on Plant Classification domain:

Despite numerous studies, plant classification based on digital images is still considered as a challenging problem.

- Caglayan *et al.* have exploited color and shape features of the leaf images to classify plants [6]
- Gaber *et al.* applied Linear Discriminant Analysis (LDA) and Principal Component Analysis (PCA) to extract visual features from plants [5]
- Salar *et al.* simple convolution neural network with 5 convolution layers automatically extract features from two-dimensional plant images [2]

All above methods include individual plant leaves. acquiring plant leaf is not always an easy task because of the occlusions and some plants may not even have recognizable leaf pieces. Classification is an active research area in pattern recognition which has been used in many applications that involves a decision mechanism that assigns a new observation to a set of classes, based on a training dataset[2].

In this project, I am going to use deep learning methods that can automatically extract features from plant image. I am proposing deep convolution neural network method for plant classification using pyramid network and I will comparing it to available deep convolution network given in salar.et.al paper on this pristine datset.

4 Methods

4.1 Pyramid unit:

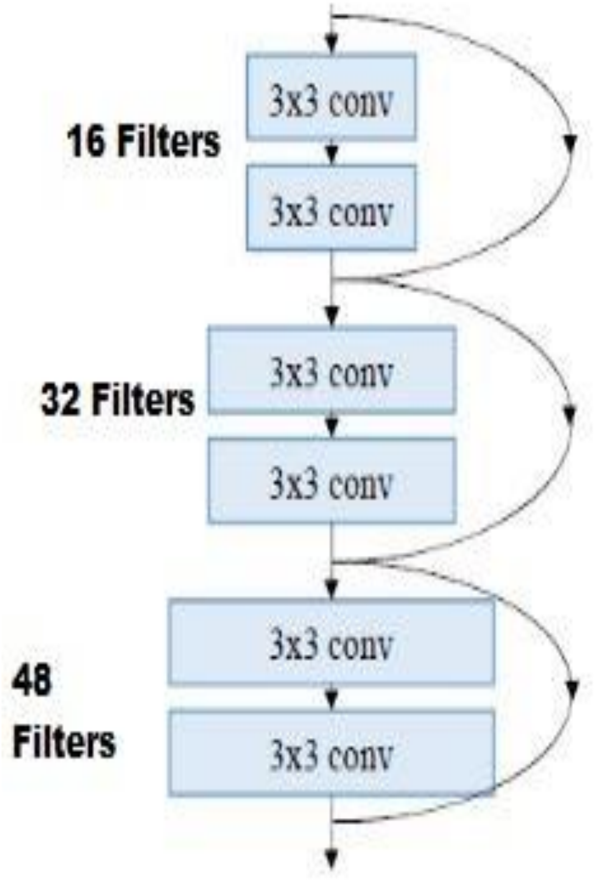


fig 4.1.1 Pyramid Unit

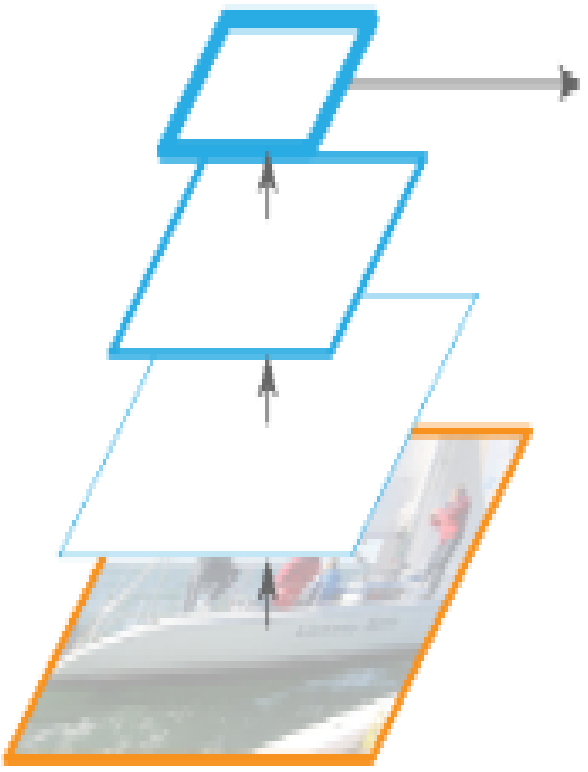
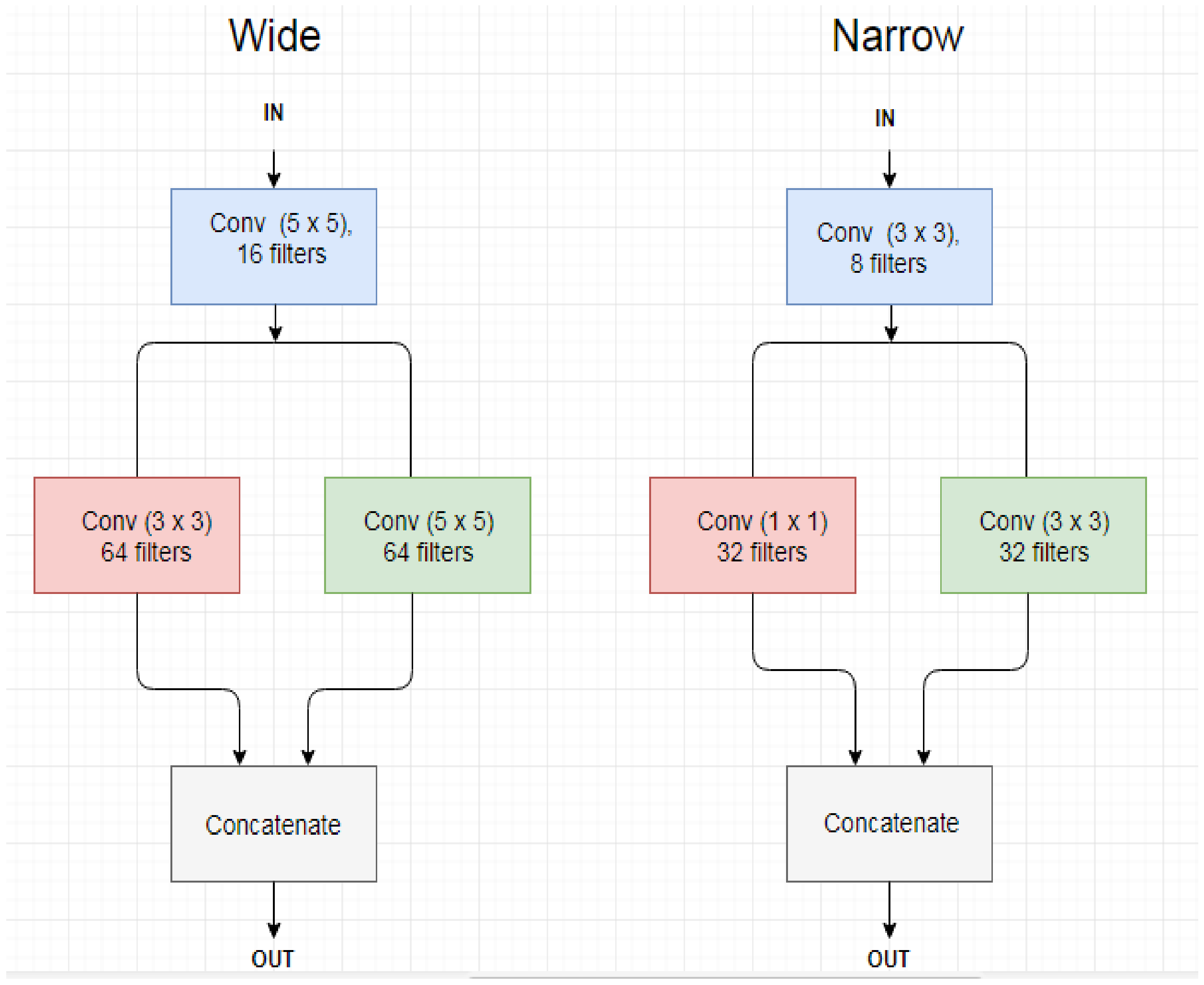


fig 4.1.2 Multiple Pyramid layers

Pyramid unit consist of 6 layers divided into 3 categories according to number of filters used in convolution layer. As shown in fig. 4.1.1, number of filters increases as category number increases. Hence it is called as Pyramid unit, though filter size remains same for every convolution layer. Response of output of each category is added to feed forward layer which feeded to next category of convolution layers.

In fig 4.1.2, multiple pyramid units connected in via masxpool layer which interns downsamples size.

4.2 Wide and Narrow Network



Wide and narrow networks can be called as shorthand version of inception network. Output of one convolution layer is passed through two different branches of convolution layers, one has lesses filter size of other, but number of filters used in every convolution layer are same. Output of both layers are concatenated to get better response from network.

4.3 Plant Classifier using Pyramid unit and Narrow-Wide Network

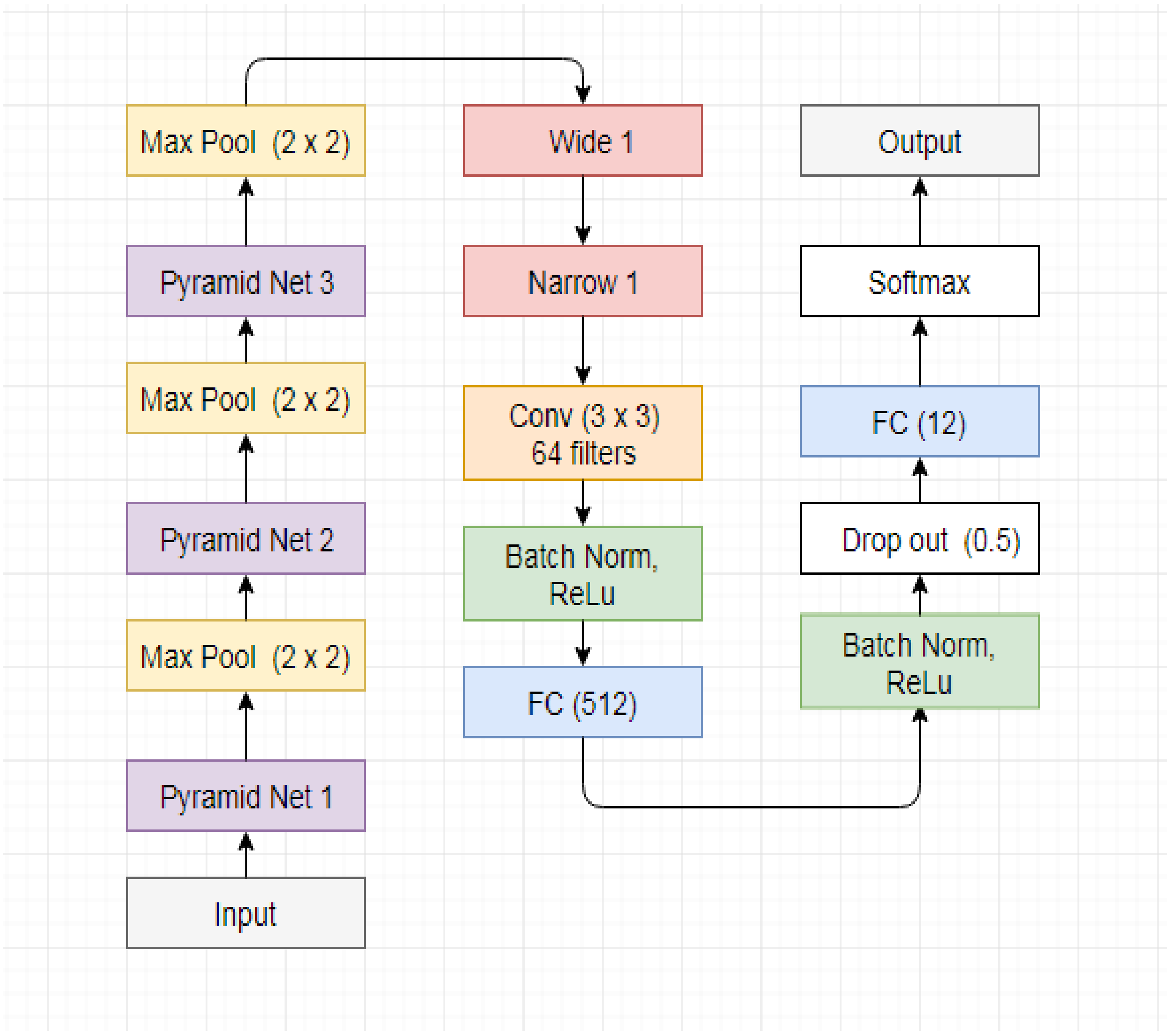
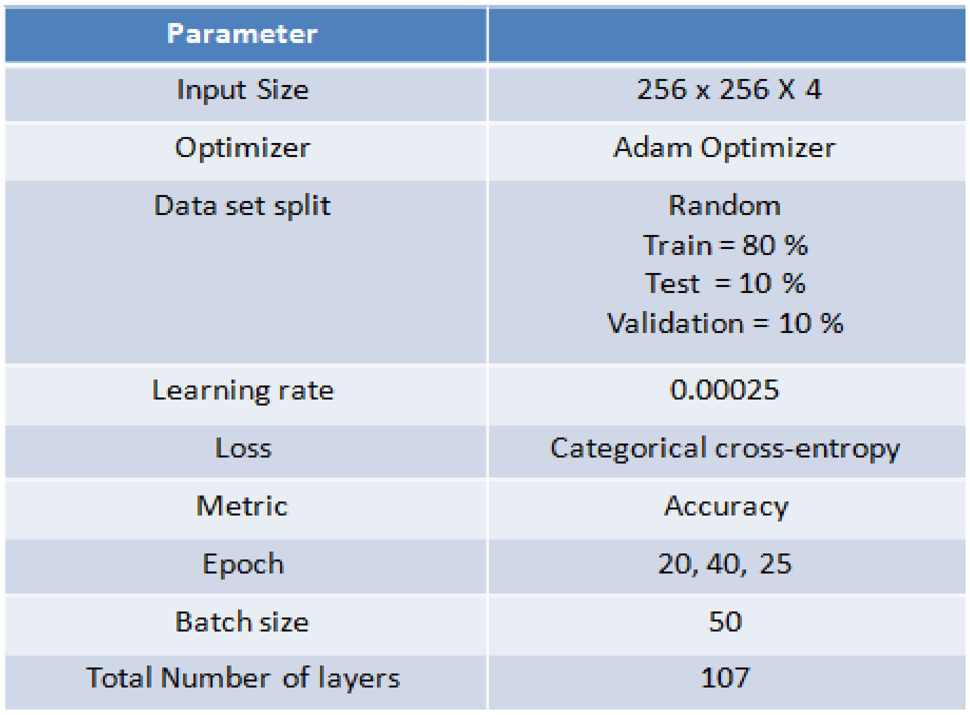


fig 4.3.1 Plant classifier

fig 4.3.1 is plant classification architecture which I am going to use in this project. It consist of 3- Pyramid networks, each followed by max pool, wide-narrow networks, fully connected layers along with relu and batch normalisation layers.

Output is class value which is in form of one hot encoding corrsponding to class number.

Model summary:

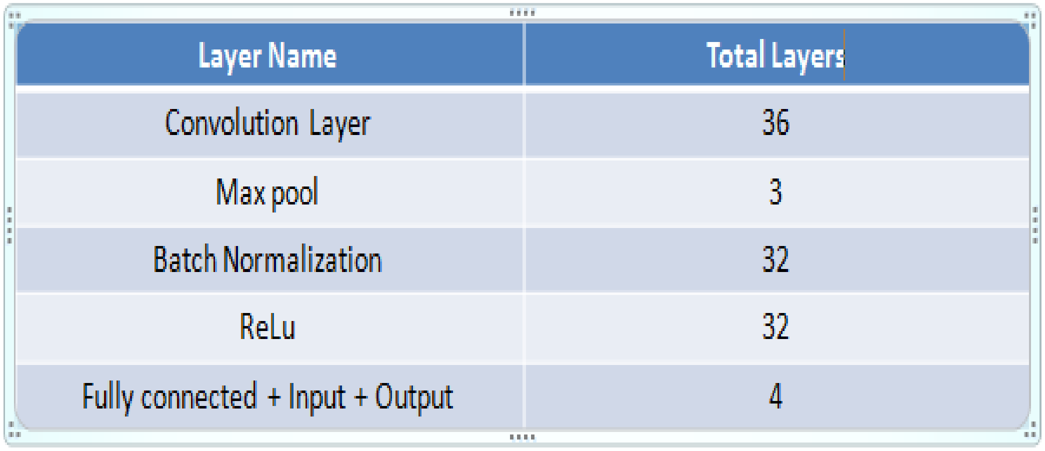


Parameter	
Input Size	256 x 256 X 4
Optimizer	Adam Optimizer

Data set split	Random
	Train = 80 %
	Test = 10 %
	Validation = 10 %

Learning rate	0.00025
Loss	Categorical cross-entropy
Metric	Accuracy
Epoch	20, 40, 25
Batch size	50
Total Number of layers	107

Table 4.3.1



Layer Name	Total Layers
Convolution Layer	36
Max pool	3
Batch Normalization	32
ReLu	32
Fully connected + Input + Output	4

Table 4.3.2

4.4 Feature Pyramid Network :

Feature Pyarmid network(FPN) is sililar to pyramid unit we have seen earlier sections. It uses pyramid unit along with upsamplers and feedforward concatenated network from convolutional block to upsamplers.

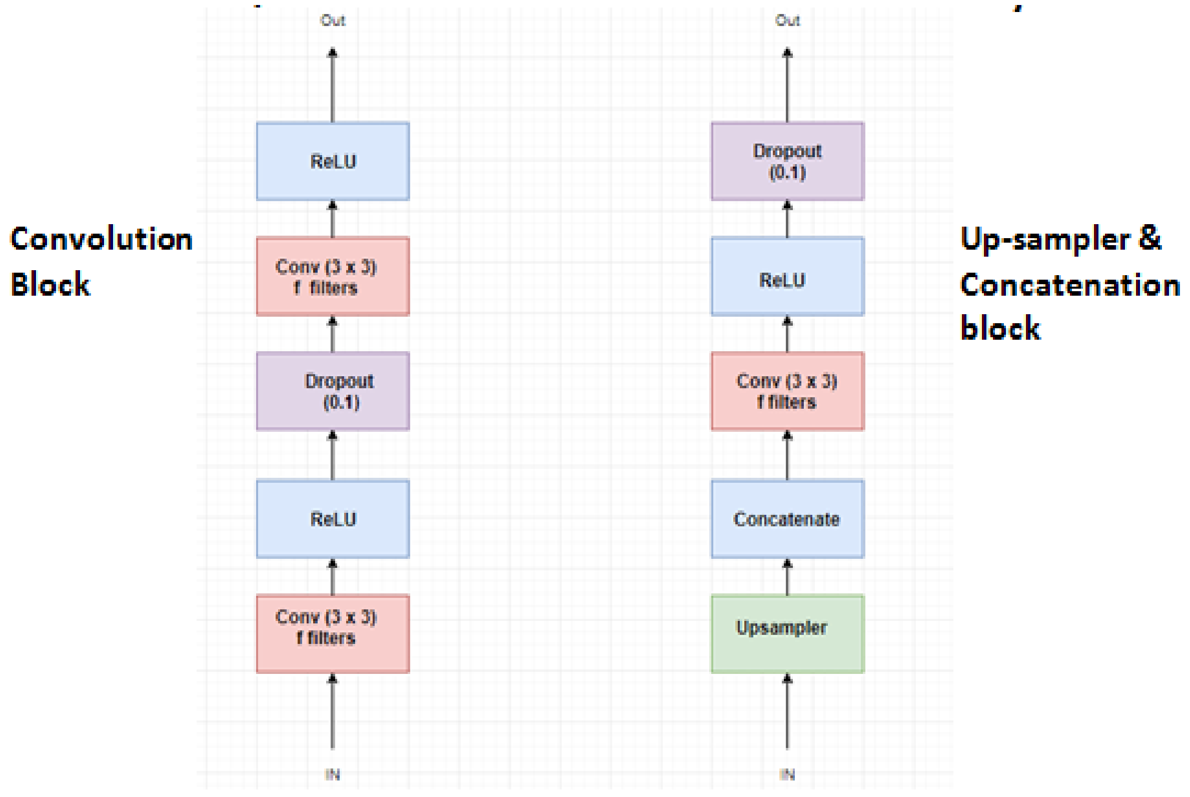


fig 4.4.1

In one convolution block have convolution layers having same size of filter and same number of filters. But different convolution block has different number of filters. In case of up-sampler and concatenation block, first input is upsampled by factor of 2 then that output and feedforward input from corresponding convolution block is concatenated. The putput is given to next layers of up-sampler.

Convolution nets feature hierarchy has semantics from low to high level, used to build feature pyramid with high level semantics throughout[8]. Aside from representing higher-level semantics, ConvNets are also more robust to variance in scale and thus facilitate recognition from features computed on a single input scale [8]. These are reasons, FPN are mostly used in object recognition task. But these feature pyramids can be useful in classification task as well.

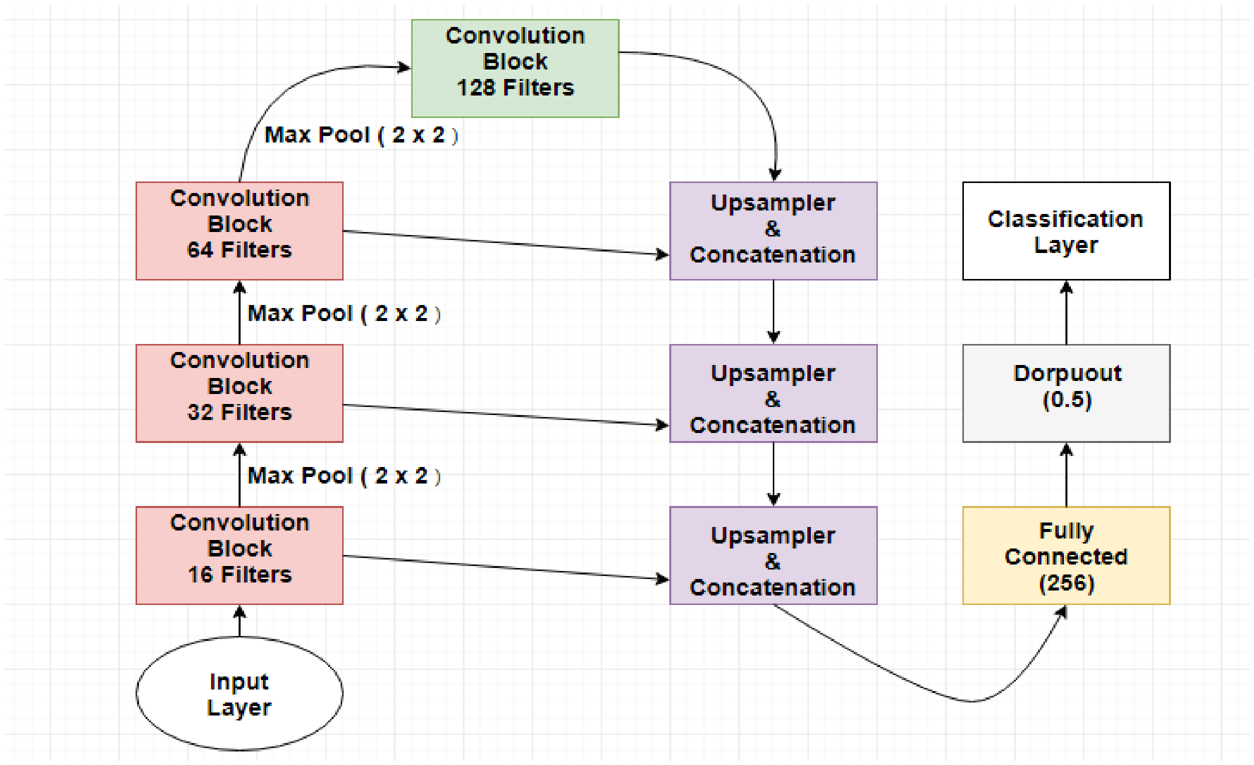

Convolution Block 128 Filters
Max Pool (2 x 2)
Convolution Block 64 Filters
Max Pool (2 x 2)
Convolution Block 32 Filters
Max Pool (2 x 2)
Convolution Block 16 Filters
Input Layer
Upsampler & Concatenation
Upsampler & Concatenation
Upsampler & Concatenation
Classification Layer
Dorpuout (0.5)
Fully Connected (256)

fig 4.4.2

Figure 4.4.2 represents final block diagram of feature Pyramid Networks. Input image is of 256 x 256 x 4 size passed through 3 Convolution blocks along with max pooling which reduces size to 32 x 32 x 64. Then it passed throgh upsampler which upsamples image by factor of 2 before adding to feed forward path from corresponding convolution block.

Model Summary of FPN:

Parameter	
Input Size	256 x 256 X 4
Optimizer	Adam Optimizer

Data set split	Random
	Train = 80 %
	Test = 10 %
	Validation = 10 %

Learning rate	0.001
Loss	Categorical cross-entropy
Metric	Accuracy
Epoch	7
Batch size	128
Total Number of layers	52

Table 4.4.1

5 Result :

5.1 Pre-processing:

Image segmentation:

Most of images of every class in dataset has black and white pattern at background which lead to mis classification of multiple examples. To get rid of this situation, I did image segmentation in which I am masking green part of image and stacking it's greyscale part on original image. Hence after image segmentation, input image has total 4 channels rather than 3 usual channels.

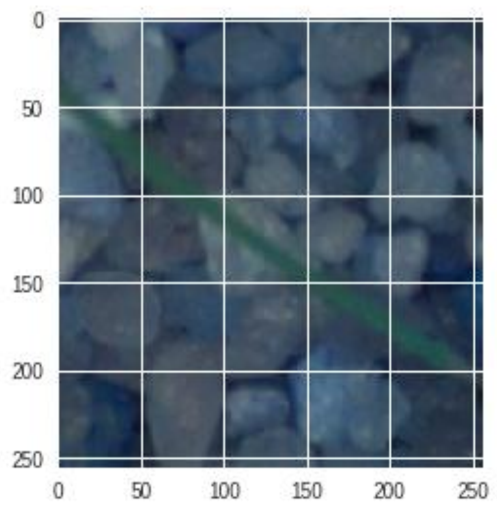


fig 5.2.1 Original Image

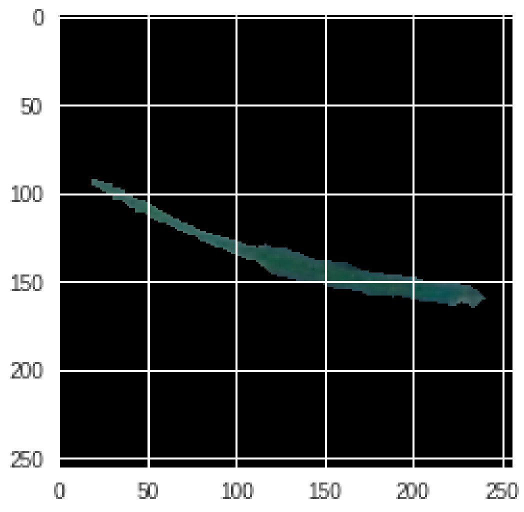


fig. 5.2.2 Segmented Image

Image Rescaling:

All my models are designed for input 256 x 256. But images in dataset are are in different sizes ranging from 100 x 100 to 250 x 250. So I am resizing this images to 256 x 256.

Converted .mat dataset :

To avoid pre-processingat every run, I have conveted all images into numpy arrays, appeded all these image array in .mat file. This file I am sharing along with code.

5.2 Salar et al Convolution Network Result:

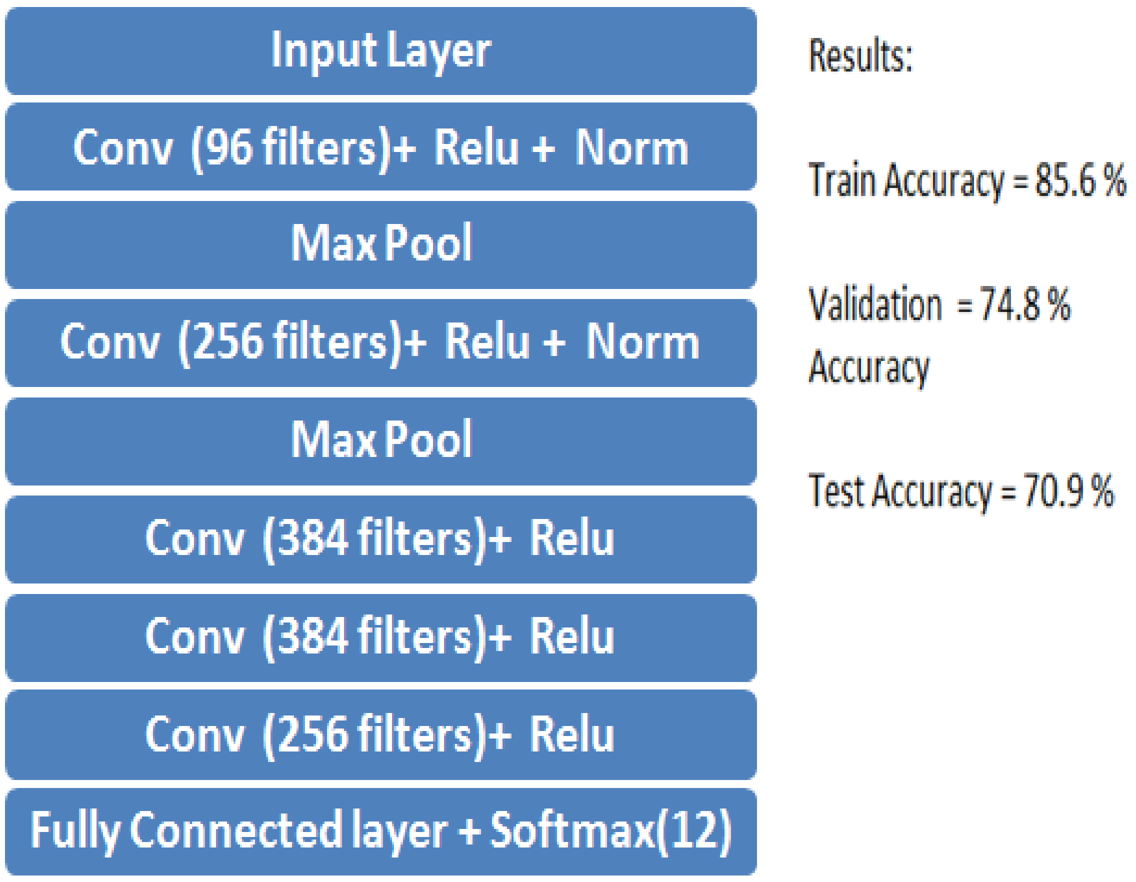


Table 5.1.1 Salar paper Network

Table 5.1.1 consist of convolutional neural network given in salar et al[2] paper for plant classification. This paper has used different dataset which has different number of classes, class types and examples. This is my base network ie I am going to compare results of this architecture with results of my architecture on dataset which I am using this project.

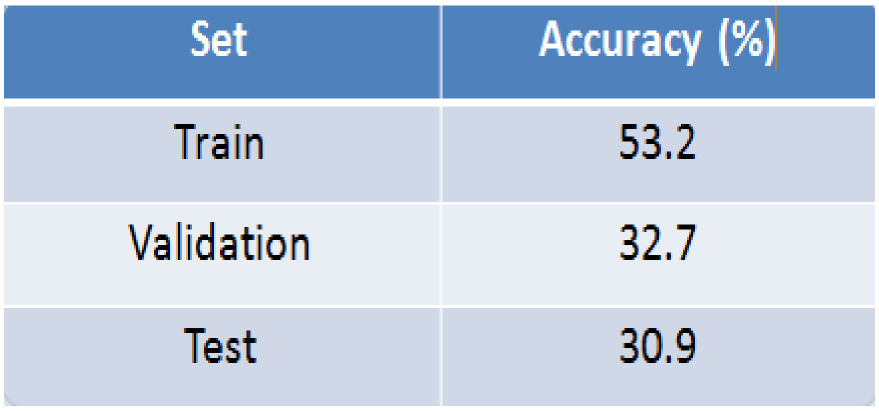
Right column of table 5.1.1 shows train and test results. From this result we can see that dataset has 5 times more images than that used in salar et al hence 5 convolution nets are not enoug for to train this dataset.

5.3 Pyramid unit and NarrowWide network Result:

Result 1:

Changes :

- Image resize (256 x 256 x 3),
- Number of Epoch : 20



Set	Accuracy (%)
Train	53.2
Validation	32.7
Test	30.9

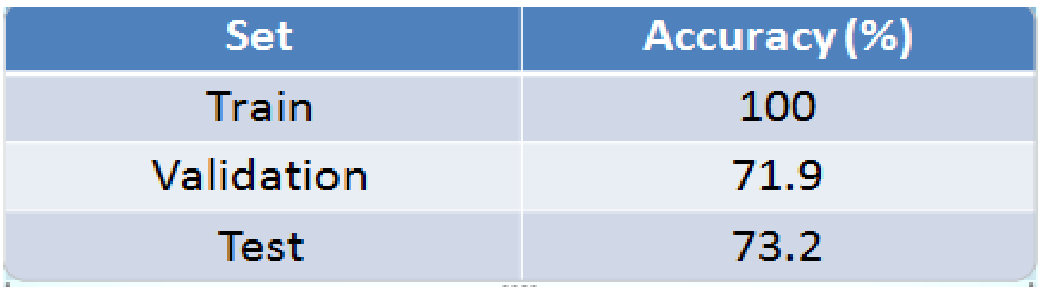
Table 5.3.1

Table 5.2.1. shows that given network has high bias and high variance. Given network should have trained for more number of epochs. That may increases training accuracy and try to fit model in better way.

Result 2:

Changes :

- Epoch = 40, drop out= 0.3
- Pre processing : Image Segmentation



Set	Accuracy (%)
Train	100
Validation	71.9
Test	73.2

table 5.3.2

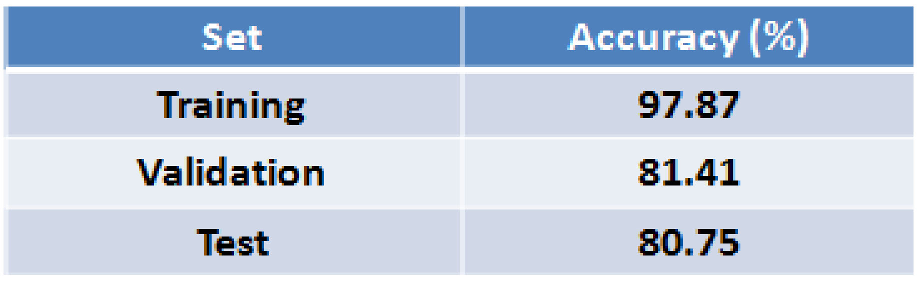
From table 5.2.2, we can see that this result has train accuracy of 100% but test and validation accuracies are around 70%. I trained this network for longer time, which lead to high variance (over- fitting).

Final Result :

Changes:

- Drop out 1 = 0.5 & Drop out 2 = 0.3
- Reduced learning rate from 0.00025 to 0.001

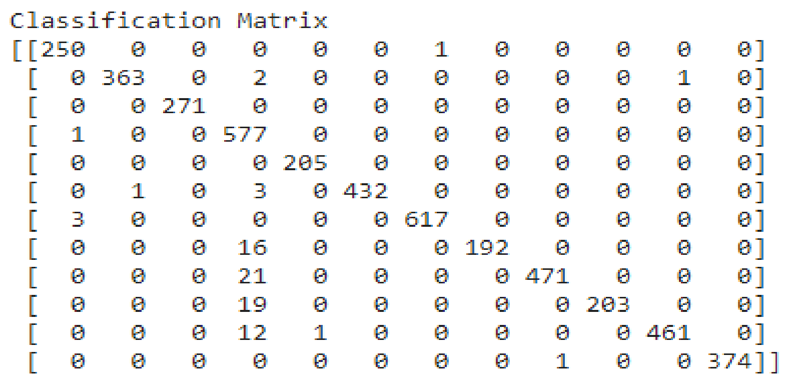
To reduce over-fitting results, I reduced number of epoch to 25 and added extra dropout layer with increases dropout percentage.
Test accuracy has increased by 10% which is impressive in this case.



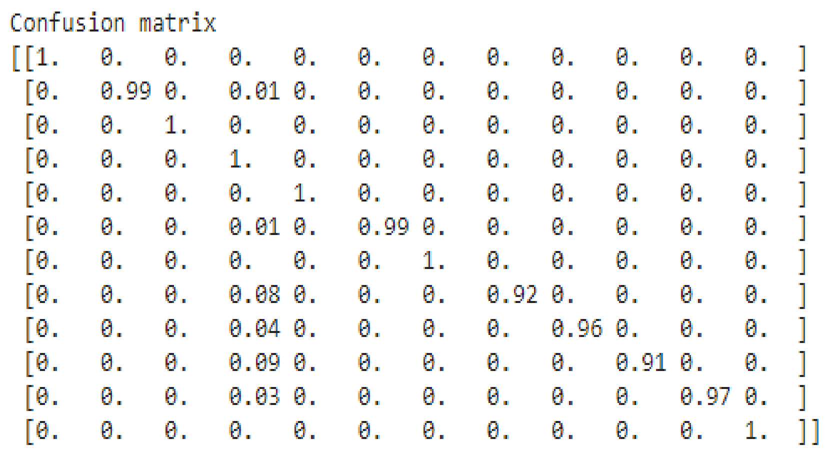
Set	Accuracy (%)
Training	97.87
Validation	81.41
Test	80.75

Table 5.3.3

Train Confusion and Classification Matrix:

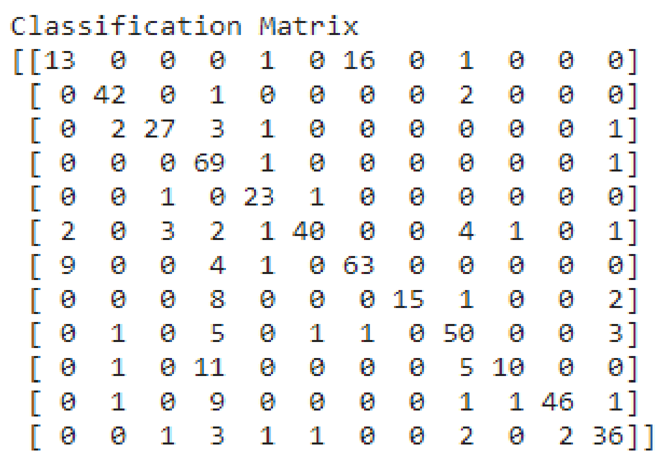


```
Classification Matrix
[[250   0   0   0   0   0   1   0   0   0   0   0]
 [  0 363   0   2   0   0   0   0   0   0   1   0]
 [  0   0 271   0   0   0   0   0   0   0   0   0]
 [  1   0   0 577   0   0   0   0   0   0   0   0]
 [  0   0   0   0 205   0   0   0   0   0   0   0]
 [  0   1   0   3   0 432   0   0   0   0   0   0]
 [  3   0   0   0   0   0 617   0   0   0   0   0]
 [  0   0   0  16   0   0   0 192   0   0   0   0]
 [  0   0   0  21   0   0   0   0 471   0   0   0]
 [  0   0   0  19   0   0   0   0   0 203   0   0]
 [  0   0   0  12   1   0   0   0   0   0 461   0]
 [  0   0   0   0   0   0   0   0   1   0   0 374]]
```

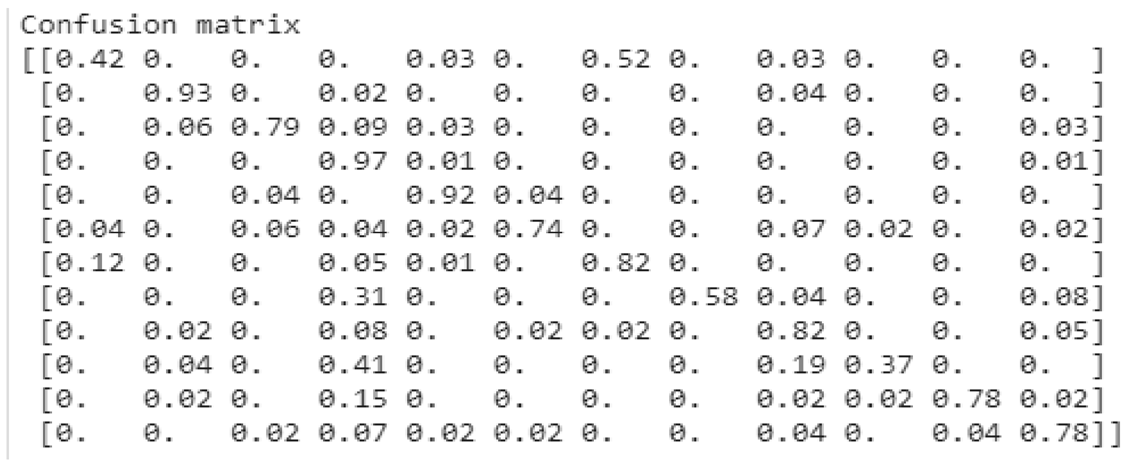


```
Confusion matrix
[[1.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.99 0.   0.01 0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   1.   0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   1.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   1.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.01 0.   0.99 0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   1.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.08 0.   0.   0.   0.92 0.   0.   0.   0.  ]
 [0.   0.   0.   0.04 0.   0.   0.   0.   0.96 0.   0.   0.  ]
 [0.   0.   0.   0.09 0.   0.   0.   0.   0.   0.91 0.   0.  ]
 [0.   0.   0.   0.03 0.   0.   0.   0.   0.   0.   0.97 0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   1.  ]]
```

Validation Classification and Confusion Matrix:

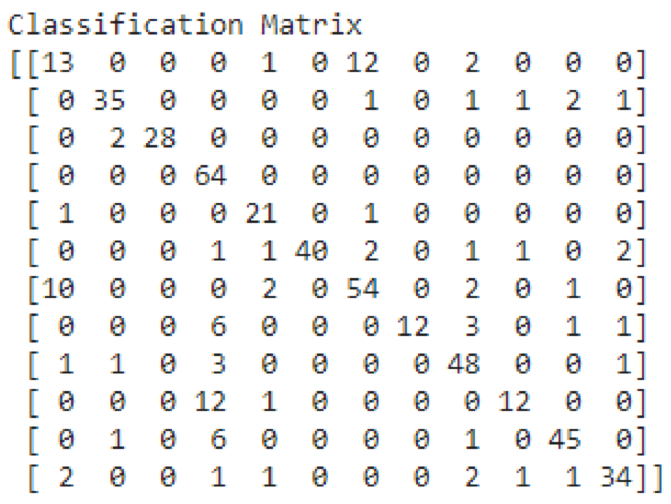


```
Classification Matrix
[[13  0  0  0  1  0 16  0  1  0  0  0]
 [ 0 42  0  1  0  0  0  0  2  0  0  0]
 [ 0  2 27  3  1  0  0  0  0  0  0  1]
 [ 0  0  0 69  1  0  0  0  0  0  0  1]
 [ 0  0  1  0 23  1  0  0  0  0  0  0]
 [ 2  0  3  2  1 40  0  0  4  1  0  1]
 [ 9  0  0  4  1  0 63  0  0  0  0  0]
 [ 0  0  0  8  0  0  0 15  1  0  0  2]
 [ 0  1  0  5  0  1  1  0 50  0  0  3]
 [ 0  1  0 11  0  0  0  0  5 10  0  0]
 [ 0  1  0  9  0  0  0  0  1  1 46  1]
 [ 0  0  1  3  1  1  0  0  2  0  2 36]]
```



```
Confusion matrix
[[0.42 0.   0.   0.   0.03 0.   0.52 0.   0.03 0.   0.   0.  ]
 [0.   0.93 0.   0.02 0.   0.   0.   0.   0.04 0.   0.   0.  ]
 [0.   0.06 0.79 0.09 0.03 0.   0.   0.   0.   0.   0.   0.03]
 [0.   0.   0.   0.97 0.01 0.   0.   0.   0.   0.   0.   0.01]
 [0.   0.   0.04 0.   0.92 0.04 0.   0.   0.   0.   0.   0.  ]
 [0.04 0.   0.06 0.04 0.02 0.74 0.   0.   0.07 0.02 0.   0.02]
 [0.12 0.   0.   0.05 0.01 0.   0.82 0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.31 0.   0.   0.   0.58 0.04 0.   0.   0.08]
 [0.   0.02 0.   0.08 0.   0.02 0.02 0.   0.82 0.   0.   0.05]
 [0.   0.04 0.   0.41 0.   0.   0.   0.   0.19 0.37 0.   0.  ]
 [0.   0.02 0.   0.15 0.   0.   0.   0.   0.02 0.02 0.78 0.02]
 [0.   0.   0.02 0.07 0.02 0.02 0.   0.   0.04 0.   0.04 0.78]]
```

Test Classification and Confusion Matrix:



```
Classification Matrix
[[13  0  0  0  1  0 12  0  2  0  0  0]
 [ 0 35  0  0  0  0  1  0  1  1  2  1]
 [ 0  2 28  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 64  0  0  0  0  0  0  0  0]
 [ 1  0  0  0 21  0  1  0  0  0  0  0]
 [ 0  0  0  1  1 40  2  0  1  1  0  2]
 [10  0  0  0  2  0 54  0  2  0  1  0]
 [ 0  0  0  6  0  0  0 12  3  0  1  1]
 [ 1  1  0  3  0  0  0  0 48  0  0  1]
 [ 0  0  0 12  1  0  0  0  0 12  0  0]
 [ 0  1  0  6  0  0  0  0  1  0 45  0]
 [ 2  0  0  1  1  0  0  0  2  1  1 34]]
```

```
Confusion matrix
[[0.46 0.   0.   0.   0.04 0.   0.43 0.   0.07 0.   0.   0.  ]
 [0.   0.85 0.   0.   0.   0.   0.02 0.   0.02 0.02 0.05 0.02]
 [0.   0.07 0.93 0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   1.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.04 0.   0.   0.   0.91 0.   0.04 0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.02 0.02 0.83 0.04 0.   0.02 0.02 0.   0.04]
 [0.14 0.   0.   0.   0.03 0.   0.78 0.   0.03 0.   0.01 0.  ]
 [0.   0.   0.   0.26 0.   0.   0.   0.52 0.13 0.   0.04 0.04]
 [0.02 0.02 0.   0.06 0.   0.   0.   0.   0.89 0.   0.   0.02]
 [0.   0.   0.   0.48 0.04 0.   0.   0.   0.   0.48 0.   0.  ]
 [0.   0.02 0.   0.11 0.   0.   0.   0.   0.02 0.   0.85 0.  ]
 [0.05 0.   0.   0.02 0.02 0.   0.   0.   0.05 0.02 0.02 0.81]]
```

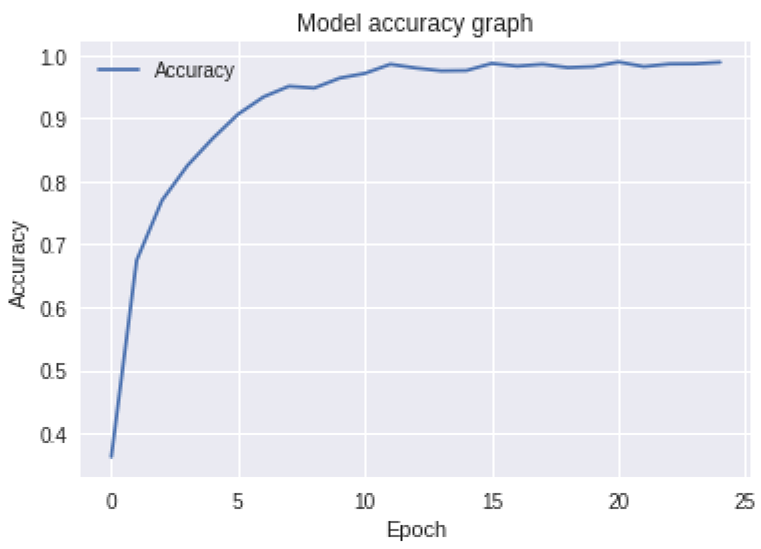


fig 5.3.1

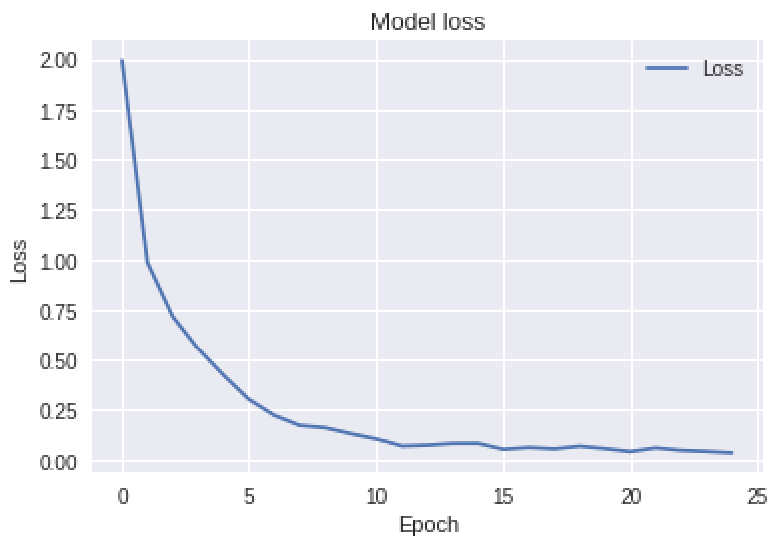


Fig 5.3.2

5.4 Feature Pyramid Network Results:

Result 1 :

(Filters: 8, 16, 32, 64)

Set	Accuracy (%)
Train	73.85
Validation	45.14
Test	41.6

Table 5.4.1

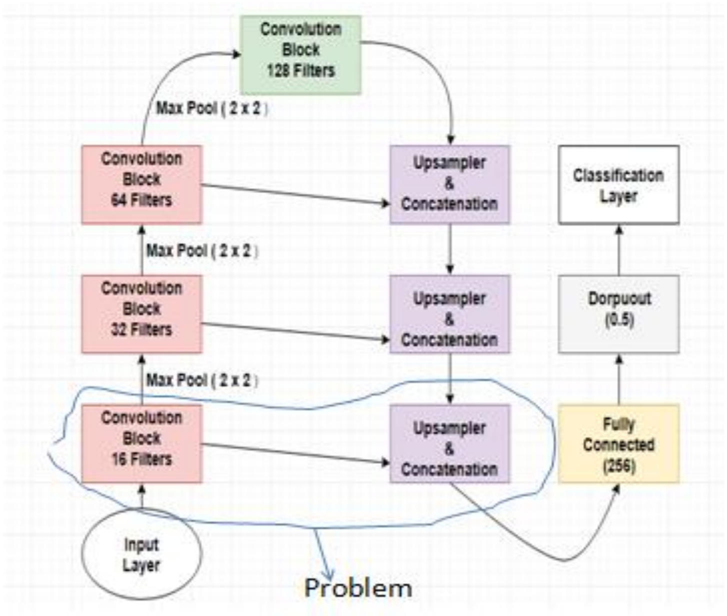
Result 2:

(Filters: 16, 32, 64, 128)

Set	Accuracy (%)
Train	90.68
Validation	62.58
Test	66.60

Table 5.4.2

Filters in both results show number of filters used in convolution blocks of feature pyramid network. Result 2 shows highest accuracy that can be achieved on this dataset by using feature pyramid network. Till 7th epoch training loss as well as validation loss was decreasing with number of epoch, but after 7th epoch, validation loss started increasing even though training loss was desreasing further.


Convolution Block 128 Filters
Max Pool (2 x 2)
Convolution Block 64 Filters
Upsampler & Concatenation
Classification Layer
Max Pool (2 x 2)
Convolution Block 32 Filters
Upsampler & Concatenation
Dorpuout (0.5)
Max Pool (2 x 2)
Convolution Block 16 Filters
Upsampler & Concatenation
Fully Connected (256)
Input Layer
Problem

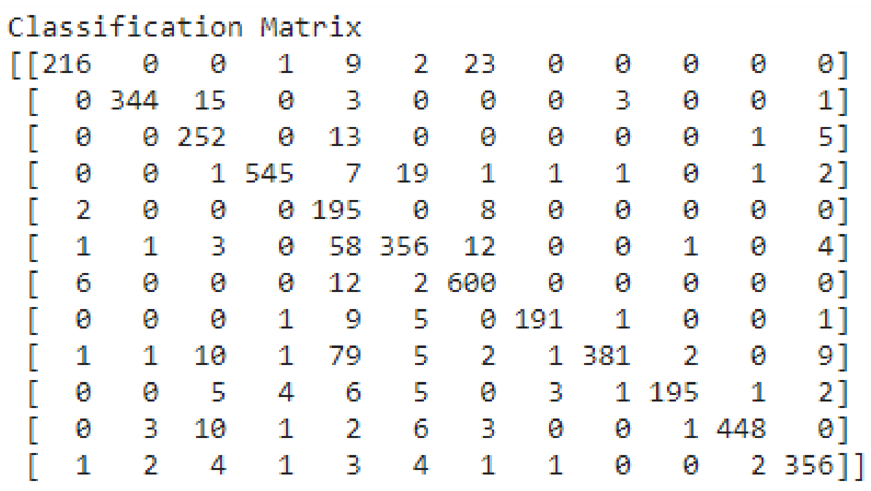
Black Grass

Maize

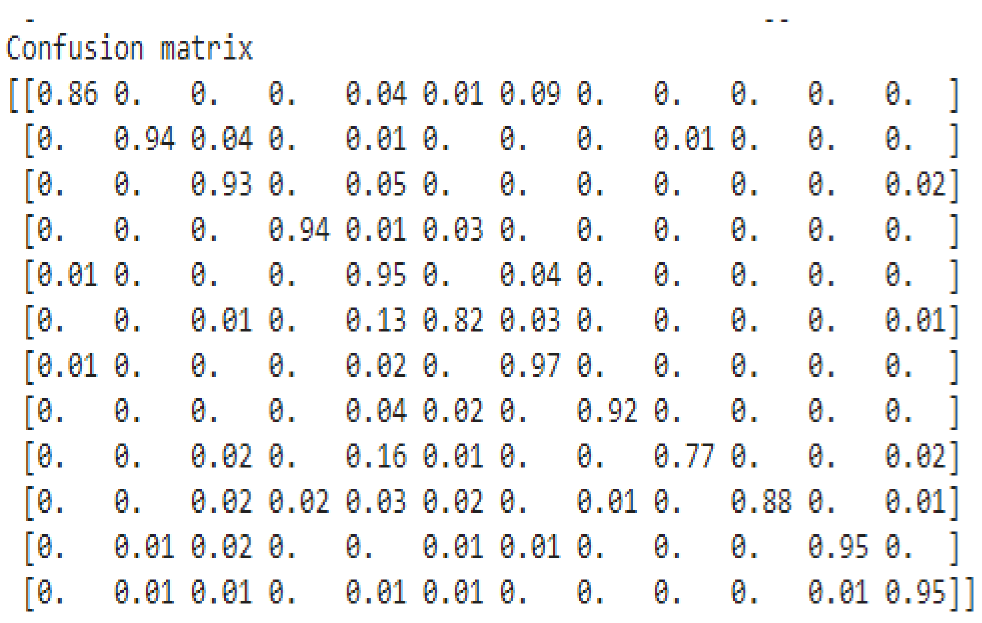
Fig 5.4.1

Figure 5.4.1 shows that issues in featur pyramid network for classification problem. After first convolution block, output is concatenated with last upsampling layer which is linked to classification layer. So in case of black grass and Maize shown in 5.4.1 figure, these features of black and white patterns in background are given given to final upsampling layer. These unnecessary patterns are causing additional mis classifications.

Train classification and confusion Matrix

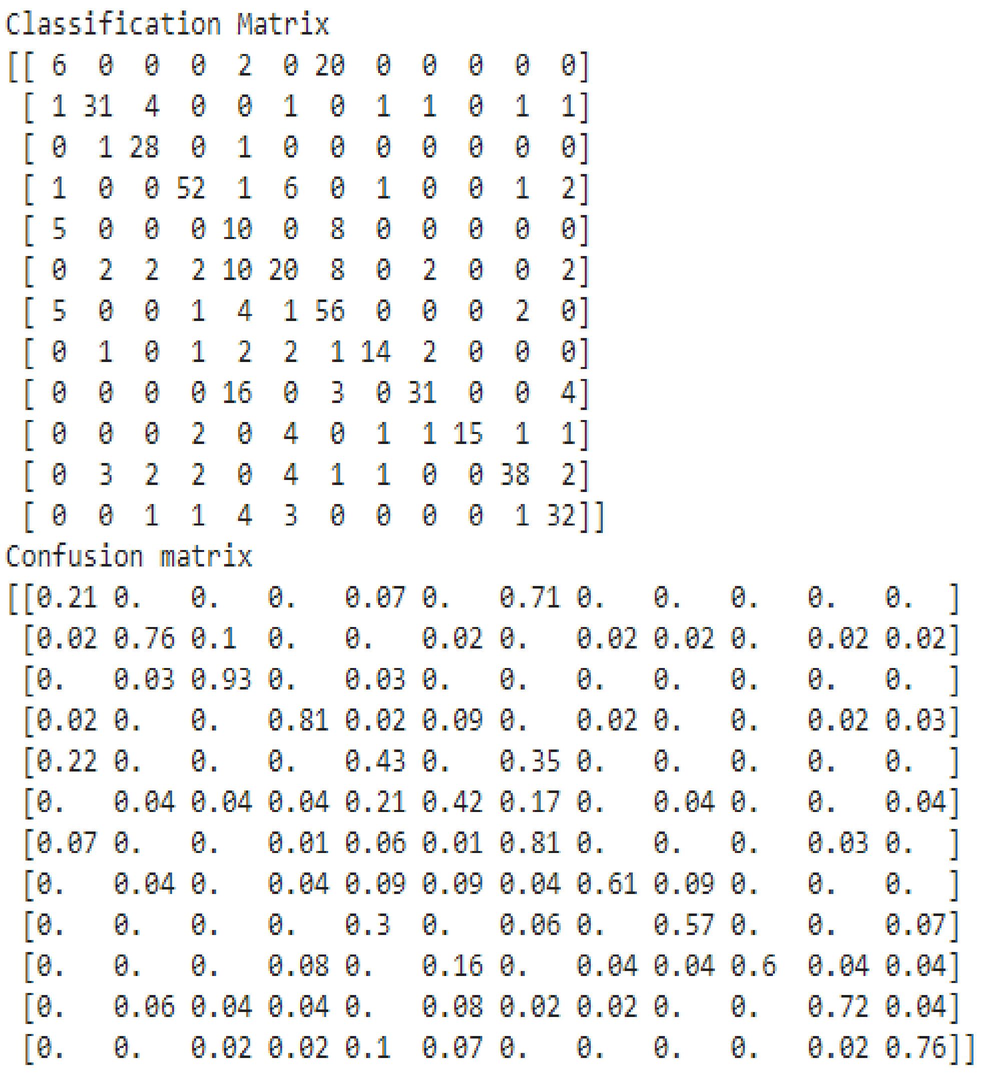


```
Classification Matrix
[[216   0   0   1   9   2  23   0   0   0   0   0]
 [  0 344  15   0   3   0   0   0   3   0   0   1]
 [  0   0 252   0  13   0   0   0   0   0   1   5]
 [  0   0   1 545   7  19   1   1   1   0   1   2]
 [  2   0   0   0 195   0   8   0   0   0   0   0]
 [  1   1   3   0  58 356  12   0   0   1   0   4]
 [  6   0   0   0  12   2 600   0   0   0   0   0]
 [  0   0   0   1   9   5   0 191   1   0   0   1]
 [  1   1  10   1  79   5   2   1 381   2   0   9]
 [  0   0   5   4   6   5   0   3   1 195   1   2]
 [  0   3  10   1   2   6   3   0   0   1 448   0]
 [  1   2   4   1   3   4   1   1   0   0   2 356]]
```



```
Confusion matrix
[[0.86 0.   0.   0.   0.04 0.01 0.09 0.   0.   0.   0.   0.  ]
 [0.   0.94 0.04 0.   0.01 0.   0.   0.   0.01 0.   0.   0.  ]
 [0.   0.   0.93 0.   0.05 0.   0.   0.   0.   0.   0.   0.02]
 [0.   0.   0.   0.94 0.01 0.03 0.   0.   0.   0.   0.   0.  ]
 [0.01 0.   0.   0.   0.95 0.   0.04 0.   0.   0.   0.   0.  ]
 [0.   0.   0.01 0.   0.13 0.82 0.03 0.   0.   0.   0.   0.01]
 [0.01 0.   0.   0.   0.02 0.   0.97 0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.04 0.02 0.   0.92 0.   0.   0.   0.  ]
 [0.   0.   0.02 0.   0.16 0.01 0.   0.   0.77 0.   0.   0.02]
 [0.   0.   0.02 0.02 0.03 0.02 0.   0.01 0.   0.88 0.   0.01]
 [0.   0.01 0.02 0.   0.   0.01 0.01 0.   0.   0.   0.95 0.  ]
 [0.   0.01 0.01 0.   0.01 0.01 0.   0.   0.   0.   0.01 0.95]]
```

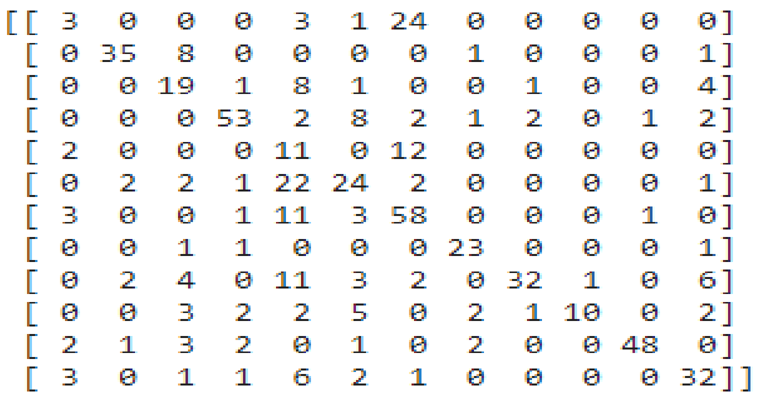
Validation classification and confusion matrix:



```
Classification Matrix
[[ 6  0  0  0  2  0 20  0  0  0  0  0]
 [ 1 31  4  0  0  1  0  1  1  0  1  1]
 [ 0  1 28  0  1  0  0  0  0  0  0  0]
 [ 1  0  0 52  1  6  0  1  0  0  1  2]
 [ 5  0  0  0 10  0  8  0  0  0  0  0]
 [ 0  2  2  2 10 20  8  0  2  0  0  2]
 [ 5  0  0  1  4  1 56  0  0  0  2  0]
 [ 0  1  0  1  2  2  1 14  2  0  0  0]
 [ 0  0  0  0 16  0  3  0 31  0  0  4]
 [ 0  0  0  2  0  4  0  1  1 15  1  1]
 [ 0  3  2  2  0  4  1  1  0  0 38  2]
 [ 0  0  1  1  4  3  0  0  0  0  1 32]]
Confusion matrix
[[0.21 0.   0.   0.   0.07 0.   0.71 0.   0.   0.   0.   0.  ]
 [0.02 0.76 0.1  0.   0.   0.02 0.   0.02 0.02 0.   0.02 0.02]
 [0.   0.03 0.93 0.   0.03 0.   0.   0.   0.   0.   0.   0.  ]
 [0.02 0.   0.   0.81 0.02 0.09 0.   0.02 0.   0.   0.02 0.03]
 [0.22 0.   0.   0.   0.43 0.   0.35 0.   0.   0.   0.   0.  ]
 [0.   0.04 0.04 0.04 0.21 0.42 0.17 0.   0.04 0.   0.   0.04]
 [0.07 0.   0.   0.01 0.06 0.01 0.81 0.   0.   0.   0.03 0.  ]
 [0.   0.04 0.   0.04 0.09 0.09 0.04 0.61 0.09 0.   0.   0.  ]
 [0.   0.   0.   0.   0.3  0.   0.06 0.   0.57 0.   0.   0.07]
 [0.   0.   0.   0.08 0.   0.16 0.   0.04 0.04 0.6  0.04 0.04]
 [0.   0.06 0.04 0.04 0.   0.08 0.02 0.02 0.   0.   0.72 0.04]
 [0.   0.   0.02 0.02 0.1  0.07 0.   0.   0.   0.   0.02 0.76]]
```

Test classification and confusion matrix:

Classification matrix:



```
[[ 3  0  0  0  3  1 24  0  0  0  0  0]
 [ 0 35  8  0  0  0  0  1  0  0  0  1]
 [ 0  0 19  1  8  1  0  0  1  0  0  4]
 [ 0  0  0 53  2  8  2  1  2  0  1  2]
 [ 2  0  0  0 11  0 12  0  0  0  0  0]
 [ 0  2  2  1 22 24  2  0  0  0  0  1]
 [ 3  0  0  1 11  3 58  0  0  0  1  0]
 [ 0  0  1  1  0  0  0 23  0  0  0  1]
 [ 0  2  4  0 11  3  2  0 32  1  0  6]
 [ 0  0  3  2  2  5  0  2  1 10  0  2]
 [ 2  1  3  2  0  1  0  2  0  0 48  0]
 [ 3  0  1  1  6  2  1  0  0  0  0 32]]
```

```
Confusion matrix
[[0.1  0.   0.   0.   0.1  0.03 0.77 0.   0.   0.   0.   0.  ]
 [0.   0.78 0.18 0.   0.   0.   0.   0.02 0.   0.   0.   0.02]
 [0.   0.   0.56 0.03 0.24 0.03 0.   0.   0.03 0.   0.   0.12]
 [0.   0.   0.   0.75 0.03 0.11 0.03 0.01 0.03 0.   0.01 0.03]
 [0.08 0.   0.   0.   0.44 0.   0.48 0.   0.   0.   0.   0.  ]
 [0.   0.04 0.04 0.02 0.41 0.44 0.04 0.   0.   0.   0.   0.02]
 [0.04 0.   0.   0.01 0.14 0.04 0.75 0.   0.   0.   0.01 0.  ]
 [0.   0.   0.04 0.04 0.   0.   0.   0.88 0.   0.   0.   0.04]
 [0.   0.03 0.07 0.   0.18 0.05 0.03 0.   0.52 0.02 0.   0.1 ]
 [0.   0.   0.11 0.07 0.07 0.19 0.   0.07 0.04 0.37 0.   0.07]
 [0.03 0.02 0.05 0.03 0.   0.02 0.   0.03 0.   0.   0.81 0.  ]
 [0.07 0.   0.02 0.02 0.13 0.04 0.02 0.   0.   0.   0.   0.7 ]]
```

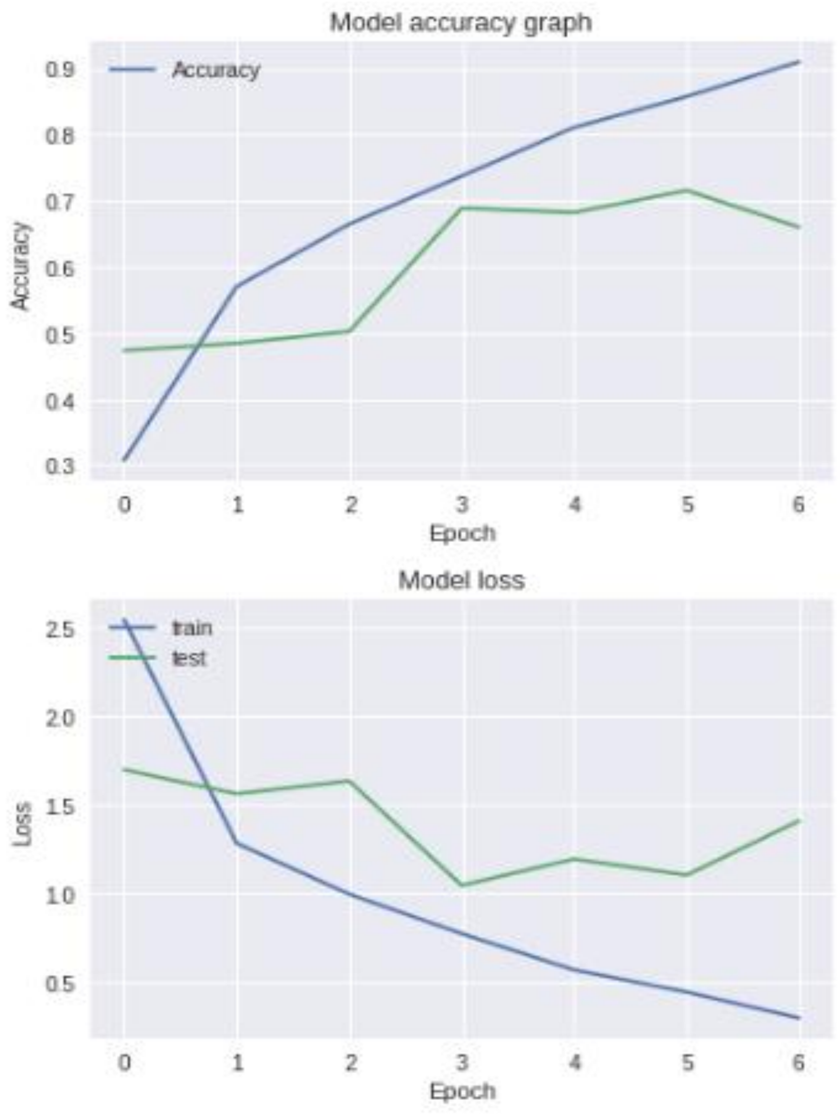
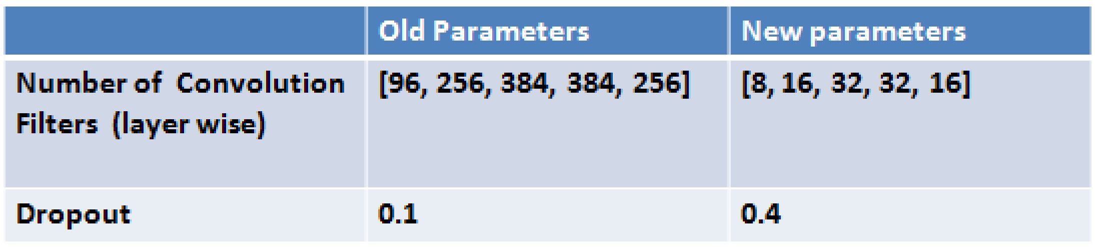


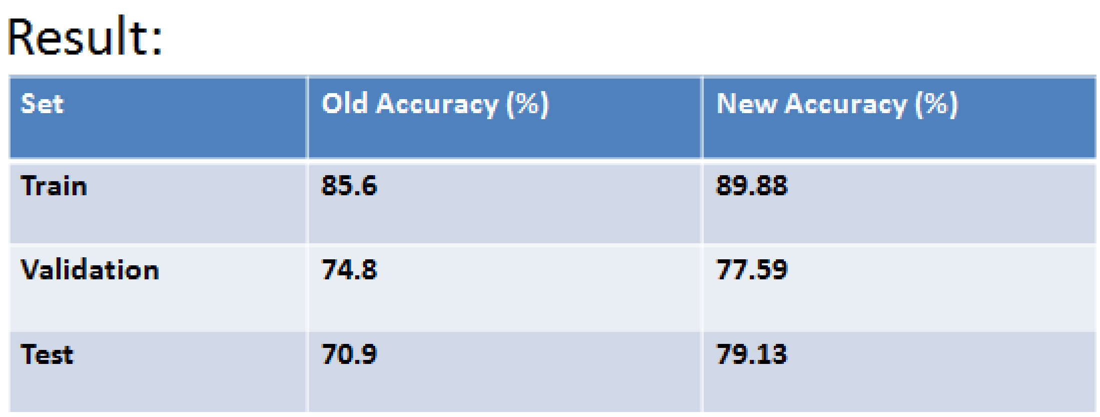
fig 5.4.2

5.5 Improvised Salar et al Convolution Network:

Changes:



	Old Parameters	New parameters
Number of Convolution Filters (layer wise)	[96, 256, 384, 384, 256]	[8, 16, 32, 32, 16]
Dropout	0.1	0.4



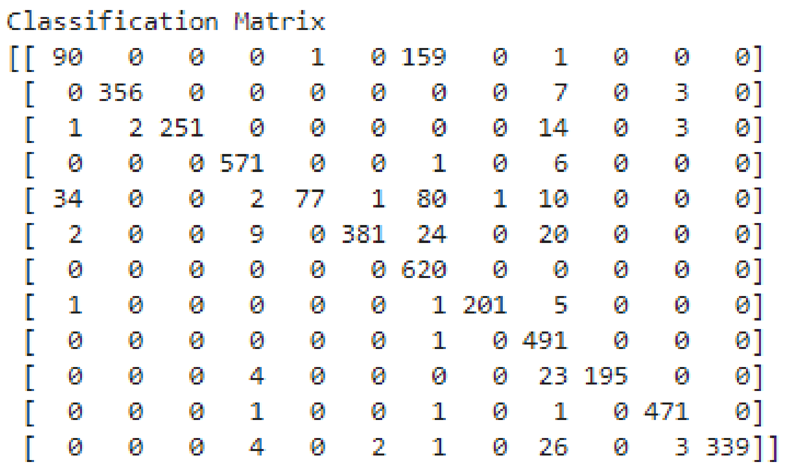
Result:

Set	Old Accuracy (%)	New Accuracy (%)
Train	85.6	89.88
Validation	74.8	77.59
Test	70.9	79.13

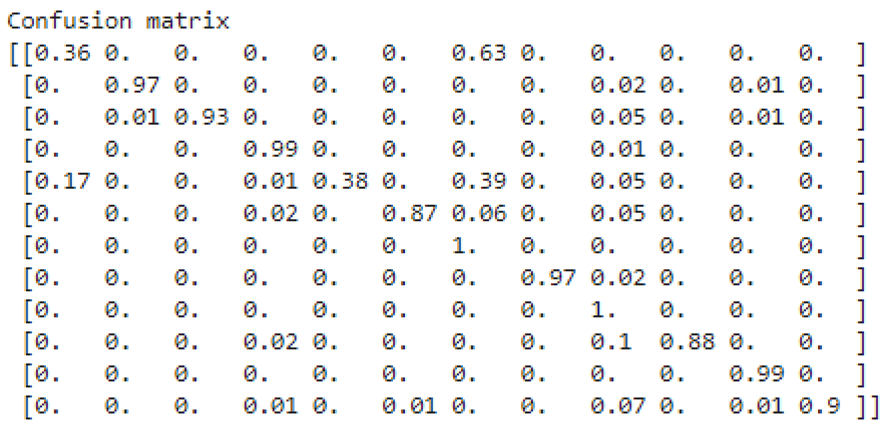
Table 5.5.1

Table 5.5.1 shows improvised results of salar et al model. Number of convolution filters in list are orderwise number of filters for convolution layers. I don't think by decreasing number of filters in each layer has improved accuracy but increasing dropout percentage could have increased it.

Train classification and confusion matrix:

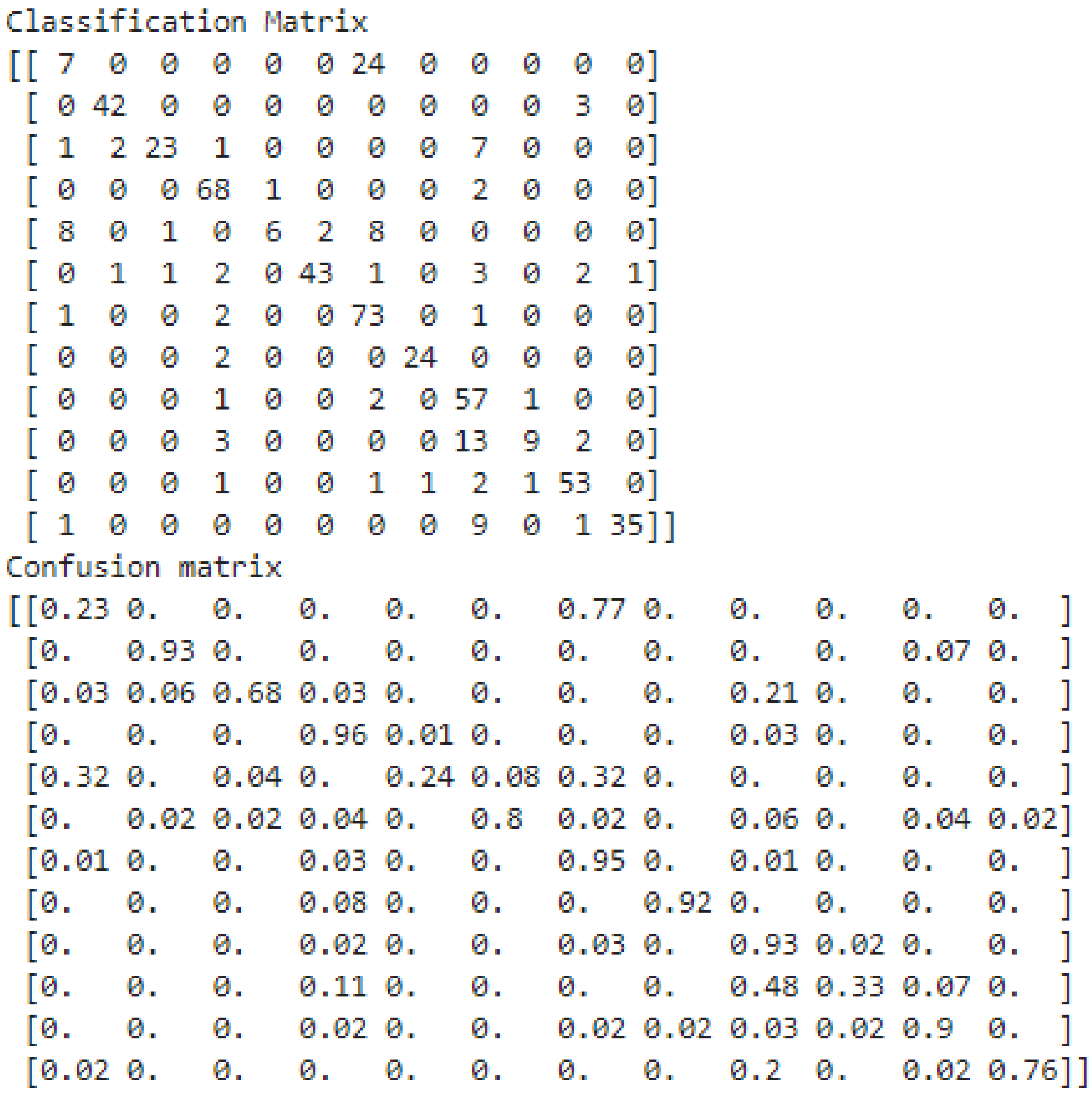


```
Classification Matrix
[[ 90   0   0   0   1   0 159   0   1   0   0   0]
 [  0 356   0   0   0   0   0   0   7   0   3   0]
 [  1   2 251   0   0   0   0   0  14   0   3   0]
 [  0   0   0 571   0   0   1   0   6   0   0   0]
 [ 34   0   0   2  77   1  80   1  10   0   0   0]
 [  2   0   0   9   0 381  24   0  20   0   0   0]
 [  0   0   0   0   0   0 620   0   0   0   0   0]
 [  1   0   0   0   0   0   1 201   5   0   0   0]
 [  0   0   0   0   0   0   1   0 491   0   0   0]
 [  0   0   0   4   0   0   0   0  23 195   0   0]
 [  0   0   0   1   0   0   1   0   1   0 471   0]
 [  0   0   0   4   0   2   1   0  26   0   3 339]]
```



```
Confusion matrix
[[0.36 0.   0.   0.   0.   0.   0.63 0.   0.   0.   0.   0.  ]
 [0.   0.97 0.   0.   0.   0.   0.   0.   0.02 0.   0.01 0.  ]
 [0.   0.01 0.93 0.   0.   0.   0.   0.   0.05 0.   0.01 0.  ]
 [0.   0.   0.   0.99 0.   0.   0.   0.   0.01 0.   0.   0.  ]
 [0.17 0.   0.   0.01 0.38 0.   0.39 0.   0.05 0.   0.   0.  ]
 [0.   0.   0.   0.02 0.   0.87 0.06 0.   0.05 0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   1.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.97 0.02 0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   1.   0.   0.   0.  ]
 [0.   0.   0.   0.02 0.   0.   0.   0.   0.1  0.88 0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.99 0.  ]
 [0.   0.   0.   0.01 0.   0.01 0.   0.   0.07 0.   0.01 0.9 ]]
```

Validation classification and confusion matrix:



```
Classification Matrix
[[ 7  0  0  0  0  0 24  0  0  0  0  0]
 [ 0 42  0  0  0  0  0  0  0  0  3  0]
 [ 1  2 23  1  0  0  0  0  7  0  0  0]
 [ 0  0  0 68  1  0  0  0  2  0  0  0]
 [ 8  0  1  0  6  2  8  0  0  0  0  0]
 [ 0  1  1  2  0 43  1  0  3  0  2  1]
 [ 1  0  0  2  0  0 73  0  1  0  0  0]
 [ 0  0  0  2  0  0  0 24  0  0  0  0]
 [ 0  0  0  1  0  0  2  0 57  1  0  0]
 [ 0  0  0  3  0  0  0  0 13  9  2  0]
 [ 0  0  0  1  0  0  1  1  2  1 53  0]
 [ 1  0  0  0  0  0  0  0  9  0  1 35]]
Confusion matrix
[[0.23 0.   0.   0.   0.   0.   0.77 0.   0.   0.   0.   0.  ]
 [0.   0.93 0.   0.   0.   0.   0.   0.   0.   0.   0.07 0.  ]
 [0.03 0.06 0.68 0.03 0.   0.   0.   0.   0.21 0.   0.   0.  ]
 [0.   0.   0.   0.96 0.01 0.   0.   0.   0.03 0.   0.   0.  ]
 [0.32 0.   0.04 0.   0.24 0.08 0.32 0.   0.   0.   0.   0.  ]
 [0.   0.02 0.02 0.04 0.   0.8  0.02 0.   0.06 0.   0.04 0.02]
 [0.01 0.   0.   0.03 0.   0.   0.95 0.   0.01 0.   0.   0.  ]
 [0.   0.   0.   0.08 0.   0.   0.   0.92 0.   0.   0.   0.  ]
 [0.   0.   0.   0.02 0.   0.   0.03 0.   0.93 0.02 0.   0.  ]
 [0.   0.   0.   0.11 0.   0.   0.   0.   0.48 0.33 0.07 0.  ]
 [0.   0.   0.   0.02 0.   0.   0.02 0.02 0.03 0.02 0.9  0.  ]
 [0.02 0.   0.   0.   0.   0.   0.   0.   0.2  0.   0.02 0.76]]
```

Test classification and confusion matrix:

```
Classification Matrix
[[ 7  0  0  0  0  0 21  0  0  0  0  0]
 [ 0 35  0  0  0  2  0  0  2  1  1  0]
 [ 0  2 25  0  0  0  1  0  2  0  0  0]
 [ 0  0  0 60  0  0  0  0  3  0  1  0]
 [ 6  0  1  0  2  1 11  0  1  0  0  1]
 [ 0  1  0  5  0 31  4  0  7  0  0  0]
 [ 0  0  0  0  0  0 66  0  2  0  1  0]
 [ 0  0  1  1  0  0  0 19  0  0  1  1]
 [ 1  0  0  0  0  0  1  1 51  0  0  0]
 [ 0  0  0  3  0  1  0  0  2 18  1  0]
 [ 0  2  0  0  0  0  0  0  1  0 50  0]
 [ 0  0  0  1  0  1  0  1 11  0  4 24]]
Confusion matrix
[[0.25 0.   0.   0.   0.   0.   0.75 0.   0.   0.   0.   0.  ]
 [0.   0.85 0.   0.   0.   0.05 0.   0.   0.05 0.02 0.02 0.  ]
 [0.   0.07 0.83 0.   0.   0.   0.03 0.   0.07 0.   0.   0.  ]
 [0.   0.   0.   0.94 0.   0.   0.   0.   0.05 0.   0.02 0.  ]
 [0.26 0.   0.04 0.   0.09 0.04 0.48 0.   0.04 0.   0.   0.04]
 [0.   0.02 0.   0.1  0.   0.65 0.08 0.   0.15 0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.96 0.   0.03 0.   0.01 0.  ]
 [0.   0.   0.04 0.04 0.   0.   0.   0.83 0.   0.   0.04 0.04]
 [0.02 0.   0.   0.   0.   0.   0.02 0.02 0.94 0.   0.   0.  ]
 [0.   0.   0.   0.12 0.   0.04 0.   0.   0.08 0.72 0.04 0.  ]
 [0.   0.04 0.   0.   0.   0.   0.   0.   0.02 0.   0.94 0.  ]
 [0.   0.   0.   0.02 0.   0.02 0.   0.02 0.26 0.   0.1  0.57]]
```

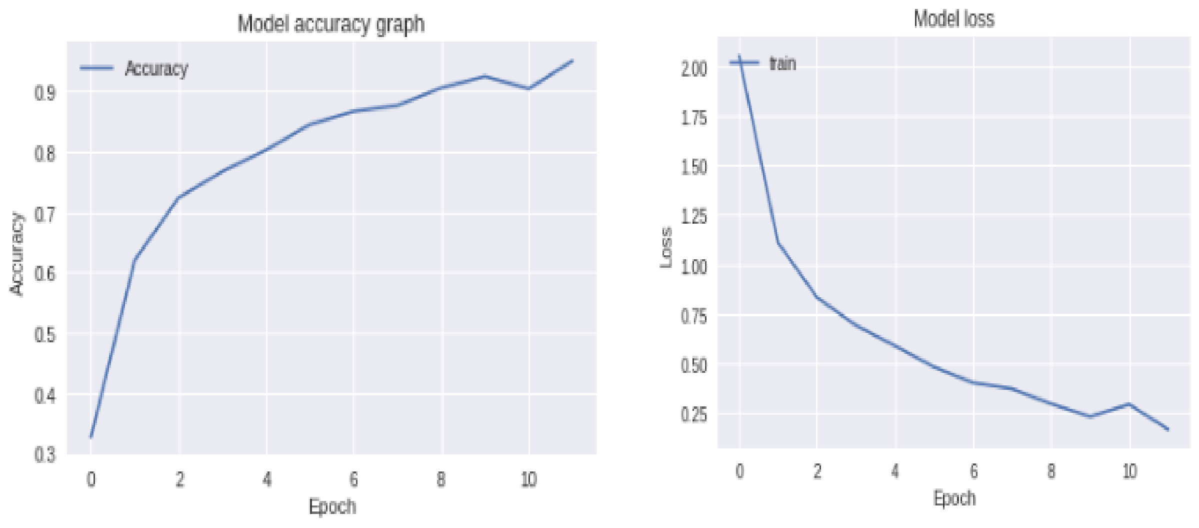
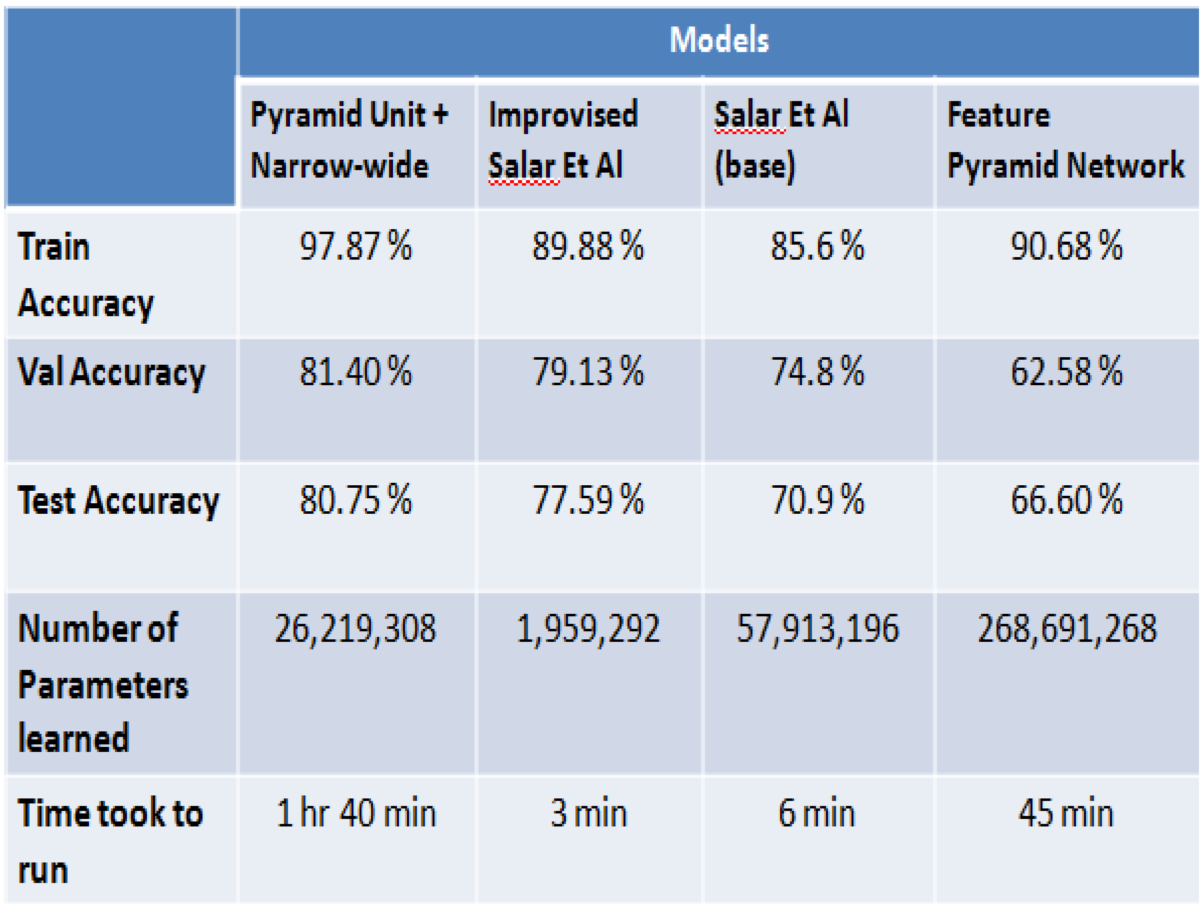


fig 5.5.2

6. Model Comaprisons



	Models			
	Pyramid Unit + Narrow-wide	Improvised Salar Et Al	Salar Et Al (base)	Feature Pyramid Network
Train Accuracy	97.87 %	89.88 %	85.6 %	90.68 %
Val Accuracy	81.40 %	79.13 %	74.8 %	62.58 %
Test Accuracy	80.75 %	77.59 %	70.9 %	66.60 %
Number of Parameters learned	26,219,308	1,959,292	57,913,196	268,691,268
Time took to run	1 hr 40 min	3 min	6 min	45 min

- In terms of accuracy, Pyramid unit with narrow wide model came out as winner. This model has surpassed base model which Salar et al model with highest validation and test accuracy in range of 80%.
- Feature pyramid network has highest parameters as compare to other models. It's accuracy lowest amongst other models.
- In terms of time to run the model and number of parameters, improvised version of Salar et al is most efficient model. It took only 3 min to run 12 epochs to give 79% accuracy on validation set. This accuracy is comparable to pyramid unit narrow wide model.
- So in case of accuracy-speed trade off and low computation power and applications where slightly low accuracy can be accepted than maximum possible accuracy, improvised Salar et al could be useful.

7. Conclusion

- Number of things came out of this project could be helpful such as structuring machine learning project in certain time limit. Now I am fully aware what should I know before approaching machine learning project.
- Pre-processing has important role in machine learning projects. If data is well pre-processed, it saves lot of computation cost and gives flexibility to train of multiple models.
- Hyper parameter selection is still well know research in machine learning domain. It is difficult to select hyper parameters especially when dealing with real world application like plant classification. I had to run each model at least on 10 different combination of hyper parameters like number of filters, filter size, learning rate etc.
- Computation power is must while dealing with deep learning projects. If you don't have computation power, you should do trade off between accuracy and speed. Some time choosing speed or time over accuracy may save unnecessary computation time.

8 References

1) Esraa Elhariri, Nashwa El-Bendary, Aboul Ella Hassanien, "Plant Classification System based on Leaf Features“, IEEE Xplore May 2014

2)Hulya Yalcin, Salar Razav, "Plant Classification using Convolutional Neural Networks"

3) A Public Image Database for Benchmark of Plant Seedling Classification Algorithms by Thomas Mosgaard Giselsson, Peter Kryger Jensen, Henrik Skov Midtiby

4) Danni ChengManhua Liu, "Classification of Alzheimer’s Disease by Cascaded Convolutional Neural Networks Using PET Images"

5)T. Gaber, A. Tharwat, V. Snasel, and A. E. Hassanien, "Plant Identification: Two Dimensional-Based Vs. One Dimensional-Based Feature Extraction Methods," in 10th International Conference on Soft Computing Models in Industrial and Environmental Applications, Á. Herrero, J. Sedano, B. Baruque, H. Quintián, and E. Corchado, Eds., ed Cham: Springer International Publishing, 2015, pp. 375-385.

6) A. Caglayan, O. Guclu, and A. B. Can, "A Plant Recognition Approach Using Shape and Color Features in Leaf Images," in Image Analysis and Processing – ICIAP 2013: 17th International Conference, Naples, Italy, September 9-13, 2013, Proceedings, Part II, A. Petrosino, Ed., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 161-170.

7) Karen Simonyan & Andrew Zisserman , "Very Deep Convolutional Networks for Large-Scale Image Recognition", arXiv:1409.1556v6 [cs.CV] 10 April, 2015

8) Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, " Feature Pyramid Networks for Object Detection"

** Credits:

1) Confusion matrix and Classification matrix python code: Python Machine learning book, second edition by Sebastian Raschka and Vahid Mirjalili

2) GPU : Google Colaboratory

3) Image Segmentation : CNN + SVM + XGBoost on kaggle by B. Thakur