**IMPLEMENTATION**

**MODULES:**

* Dataset
* Importing the necessary libraries
* Retrieving the images
* Splitting the dataset
* Building the model
* Apply the model and plot the graphs for accuracy and loss
* Accuracy on test set
* Saving the Trained Model

**MODULES DESCSRIPTION:**

**Dataset:**

In the first module, we developed the system to get the input dataset for the training and testing purpose. Dataset is given in the model folder. The dataset consists of 70,295 Plant images of Apple, Blueberry, Cherry, Corn (maize), Grape, Orange, Peach, Pepper bell, Potato, Raspberry, Soybean, Strawberry and Tomato. The dataset is referred from the popular public repository kaggle. The following is the link of the dataset.

Kaggle dataset Link:

<https://www.kaggle.com/datasets/jayaprakashpondy/plant-disease-dataset>

**Importing the necessary libraries:**

In the next module, we will be importing the necessary libraries for our plant disease detection system. We will be using Python language for this. First we will import the necessary libraries such as keras for building the main model, sklearn for splitting the training and test data, PIL for converting the images into array of numbers and other libraries such as pandas, numpy, matplotlib and tensorflow.

**Retrieving the images:**

We will retrieve the images and their labels. Then resize the images to (224,224) as all images should have same size for recognition. Then convert the images into numpy array.

**Splitting the dataset:**

Split the dataset into train and test. 80% train data and 20% test data.

**Building the model:**

In this module, we will be building the model.

**InceptionV3| CNN model**

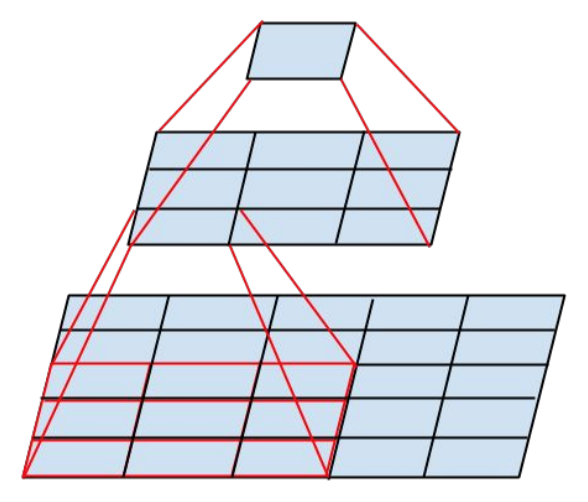
**Inception v3:** mainly focuses on burning less computational power by modifying the previous Inception architectures. In comparison to VGGNet, Inception Networks (GoogLeNet/Inception v1) have proved to be more computationally efficient, both in terms of the number of parameters generated by the network and the economical cost incurred (memory and other resources). If any changes are to be made to an Inception Network, care needs to be taken to make sure that the computational advantages aren’t lost. Thus, the adaptation of an Inception network for different use cases turns out to be a problem due to the uncertainty of the new network’s efficiency. In an Inception v3 model, several techniques for optimizing the network have been put suggested to loosen the constraints for easier model adaptation. The techniques include factorized convolutions, regularization, dimension reduction, and parallelized computations.

### Inception v3 Architecture:

The architecture of an Inception v3 network is progressively built, step-by-step, as explained below:

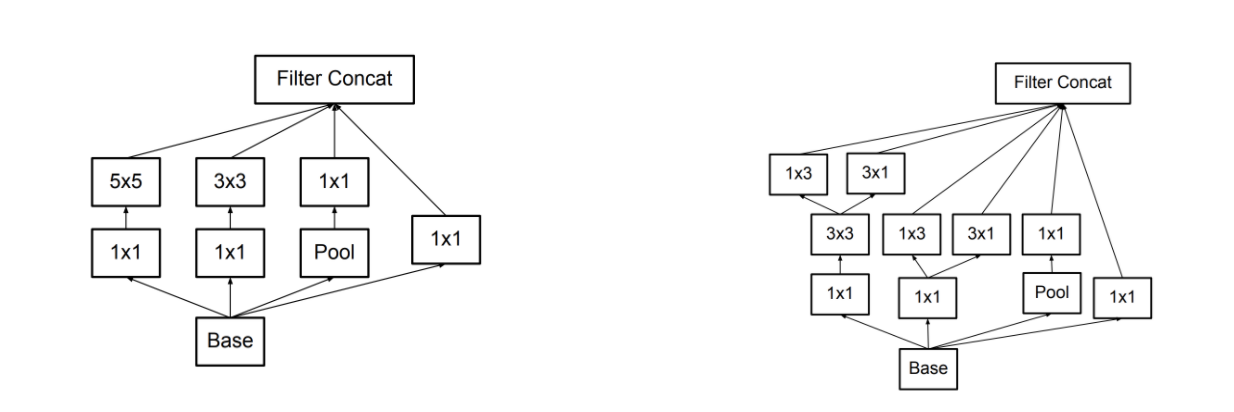
1. **Factorized Convolutions:** this helps to reduce the computational efficiency as it reduces the number of parameters involved in a network. It also keeps a check on the network efficiency.

**2. Smaller convolutions:** replacing bigger convolutions with smaller convolutions definitely leads to faster training. Say a 5 × 5 filter has 25 parameters; two 3 × 3 filters replacing a 5 × 5 convolution has only 18 (3\*3 + 3\*3) parameters instead.

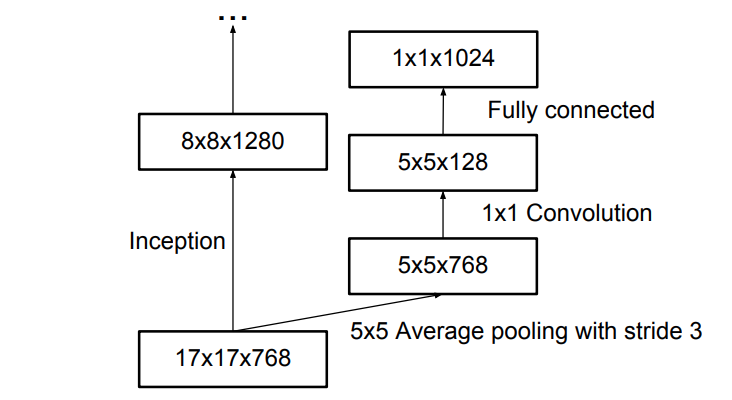


In the middle we see a 3x3 convolution, and below a fully-connected layer. Since both 3x3 convolutions can share weights among themselves, the number of computations can be reduced.

**3. Asymmetric convolutions:** A 3 × 3 convolution could be replaced by a 1 × 3 convolution followed by a 3 × 1 convolution. If a 3 × 3 convolution is replaced by a 2 × 2 convolution, the number of parameters would be slightly higher than the asymmetric convolution proposed.



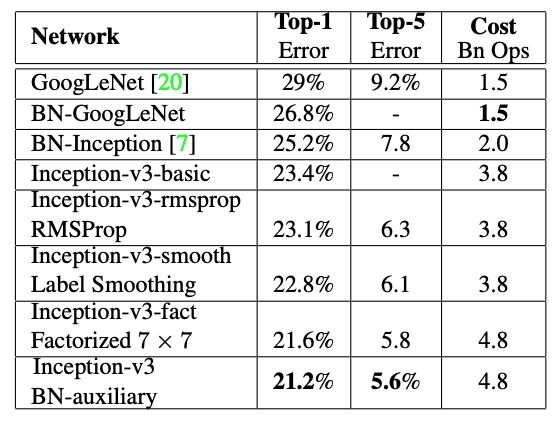
**4. Auxiliary classifier:** an auxiliary classifier is a small CNN inserted between layers during training, and the loss incurred is added to the main network loss. In GoogLeNet auxiliary classifiers were used for a deeper network, whereas in Inception v3 an auxiliary classifier acts as a regularizer.



**5. Grid size reduction:** Grid size reduction is usually done by pooling operations. However, to combat the bottlenecks of computational cost, a more efficient technique is proposed:

### Inception v3 Training and Results

Inception v3 was trained on ImageNet and compared with other contemporary models, as shown below.



As shown in the table, when augmented with an auxiliary classifier, factorization of convolutions, RMSProp, and Label Smoothing, Inception v3 can achieve the lowest error rates compared to its contemporaries.

**Apply the model and plot the graphs for accuracy and loss:**

We will compile the model and apply it using fit function. The batch size will be 10. Then we will plot the graphs for accuracy and loss. We got average validation accuracy of 91.00% .

**Accuracy on test set:**

We got an accuracy of 89.00% on test set.

**Saving the Trained Model:**

Once you’re confident enough to take your trained and tested model into the production-ready environment, the first step is to save it into a .h5 or .pkl file using a library like pickle.

Make sure you have pickle installed in your environment.

Next, let’s import the module and dump the model into .h5 file.