Rice Crop Disease Detection using YoLO

Rice is the main ingredient of food in india. Increasing grain production by controlling crop diseases in time should be effective. To construct a prediction model for plant diseases and to build a real-time application for crop diseases in the future, a deep learning-based image detection architecture with a custom backbone was proposed for detecting plant diseases.

In this project i have used ***yolov3*** model. YoLO (you only look once), an algorithm that is used for object detection in real time. It predicts classes and bounding boxes for the testing images in one run of the algorithm.

Dataset:

I have taken custom images from google for our project.
In this dataset i have taken 4 classes - Bacterial Blight, Leaf smut, White tip, and No diseases.
I have downloaded images of the above mentioned classes from the google.

To annotate images I have used a tool named as "***Vott***" - Visual Object Tagging Tool. It is an open source image annotation and labelling tool developed by microsoft.

I have downloaded nearly 200-300 images and annotated them using vott. The vott tool allows you to export these annotation coordinates into different files. Here I have exported them into csv file.

Yolov3:

YOLO-based Convolutional Neural Network family of models for object detection and the most recent variation called YOLOv3. It is the version 3 of YoLo algorithm, it identifies specific objects in videos, live feeds or images. The YoLo algorithm is deployed in DarkNet. Darknet is an open source neural network framework written in C and CUDA.

yolov3 has a deeper architecture of feature extractor called Darknet-53. A 53-layer trained on Imagenet and for the task of detection 53 more layers has been stacked onto it, which gives us a 106 layer fully convolutional underlying architecture for yolov3.

Training YOLO:

By using the generic code files of yolo model, start your training. First download the pre-trained yolov3 weights (these are pre-trained on Imagenet 1000 dataset) - works well for object detection tasks. After downloading, convert the pre-trained weights into keras format.
Now start training by running your yolo code. The training takes time based on how many images you have given, if you have more images then it takes more processing time. The trained weights will be saved into a .h5 file.

It took 8.5 hours for me to train my model.

Testing:

To test the model, create a test images folder and download some images into that folder. I have given 5 images for testing. After testing the output images will go into "Test_Image_Detection_Results".

Output:

The Output can be seen in the "Test_Image_Detection_Results" folder, the image consists of bounding boxes over that output and the class name will be on the bounded box. A csv file will also be created with the coordinate values.

Conclusion:

The accuracy I got after testing is 75%. As I have trained the model with less images the accuracy is less. By training with more than 500 images then we can get an accuracy above 90%.