

19UK1A05J9

RICE CROP DISEASE DETECTION USING YOLO

Project UG Phase 1

Project Description:

Increasing grain production is essential to those areas where food is scarce. 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, deep learning-based image detection architecture with a custom backbone was proposed for detecting plant diseases.

In order to get a good amount of crop, we need to detect the disease at the earliest.

Basically, crop disease diagnosis depends on different characteristics like color, shape, texture, etc. Here the person can capture the images of the crop and then the image will be sent to the trained YOLO model. The model analyzes the image and detects crop diseases like Bacterial Blight, Leaf Smut, and White Tip.

Architecture:

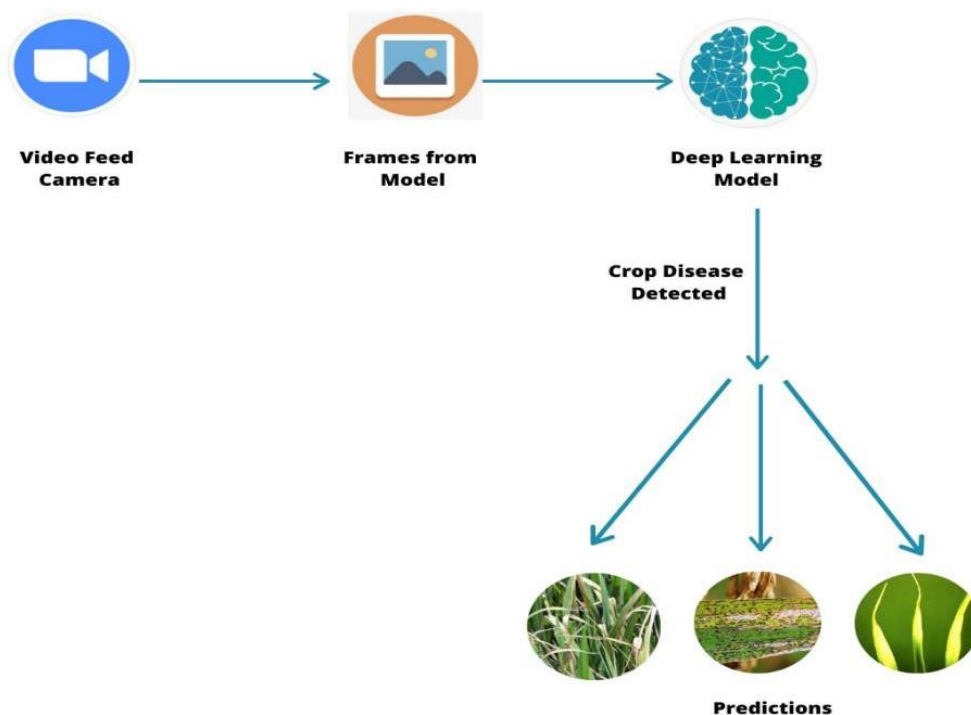


Fig. Project Architectural Diagram

Project Objectives:

By the end of this project you'll understand:

- YOLO-based Convolution Neural Network family of models for object detection and the most recent variation called YOLOv3.
- How to train a YOLO model in a windows environment.
- How to annotate images using Microsoft's Visual Object Tagging Tool (VoTT).

Pre-Requisites:

To complete this project, you must require the following software's, concepts, and packages

- Python IDE (IDLE / Spyder / PyCharm)(Python 3.7)
- Microsoft's Visual Object Tagging Tool (VoTT)
- Python Packages need to be installed

Task 1: Anaconda Installation

To complete this project you should have the following software and packages

Anaconda Navigator:

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS. Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook,

QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupyter notebook and spyder

To install Anaconda navigator and to know how to use Jupyter Notebook a Spyder using Anaconda watch the video:

Link: <https://youtu.be/5mDYijMfSzs>

To build Deep learning models you must require the following packages:

Tensor flow: Tensor Flow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers can easily build and deploy ML-powered applications.

Keras: Keras leverages various optimization techniques to make high-level neural network API easier and more performant. It supports the following features:

- Consistent, simple, and extensible API.
- Minimal structure - easy to achieve the result without any frills.
- It supports multiple platforms and backends.
- It is a user-friendly framework that runs on both CPU and GPU.
- Highly scalability of computation.

To implement this project we will be using Python 3.7 strictly.

Link: https://repo.anaconda.com/archive/Anaconda3-2019.07-Windows-x86_64.exe

Task 2: Install python packages

Follow the below steps to install packages :

- Open anaconda prompt as administrator.
- Type "pip install tensorflow==1.15.1" and click enter.
- Type "pip install keras=2.2.4" and click enter.
- Type "pip install opencv-python==4.1.0.25" and click enter.
- Type "pip install Pillow==6.2.2" and click enter.

the above steps allow you to install the packages in the anaconda environment

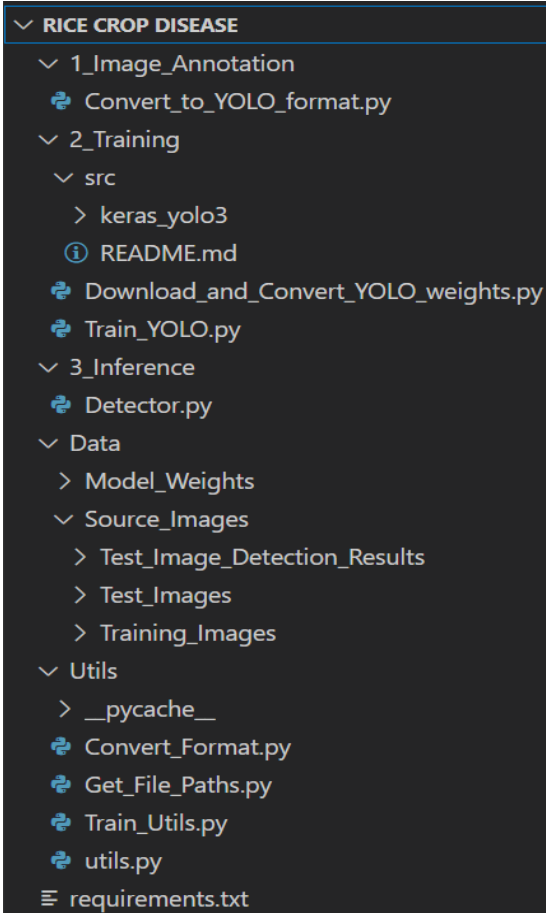
Task 3: Install Microsoft's Visual Object Tagging Tool (VoTT)

Head to VoTT [releases](https://github.com/Microsoft/VoTT/releases) and download and install the version for your operating system. Under `Assets` select the package for your operating system:

- 'vott-2.x.x-darwin.dmg' for Mac users,
- 'vott-2.x.x-win32.exe' for Windows users and
- 'vott-2.x.x-linux.snap' for Linux users.

Link: <https://www.youtube.com/watch?v=h8NUIiOt1Hk>

Project Structure:



```

▼ RICE CROP DISEASE
  ▼ 1_Image_Annotation
    📄 Convert_to_YOLO_format.py
  ▼ 2_Training
    ▼ src
      > keras_yolo3
    ⓘ README.md
    📄 Download_and_Convert_YOLO_weights.py
    📄 Train_YOLO.py
  ▼ 3_Inference
    📄 Detector.py
  ▼ Data
    > Model_Weights
    ▼ Source_Images
      > Test_Image_Detection_Results
      > Test_Images
      > Training_Images
  ▼ Utils
    > __pycache__
    📄 Convert_Format.py
    📄 Get_File_Paths.py
    📄 Train_Utils.py
    📄 utils.py
  📄 requirements.txt

```

Note: These are generic YoLo code files that are not confined to particular project

Create Dataset:

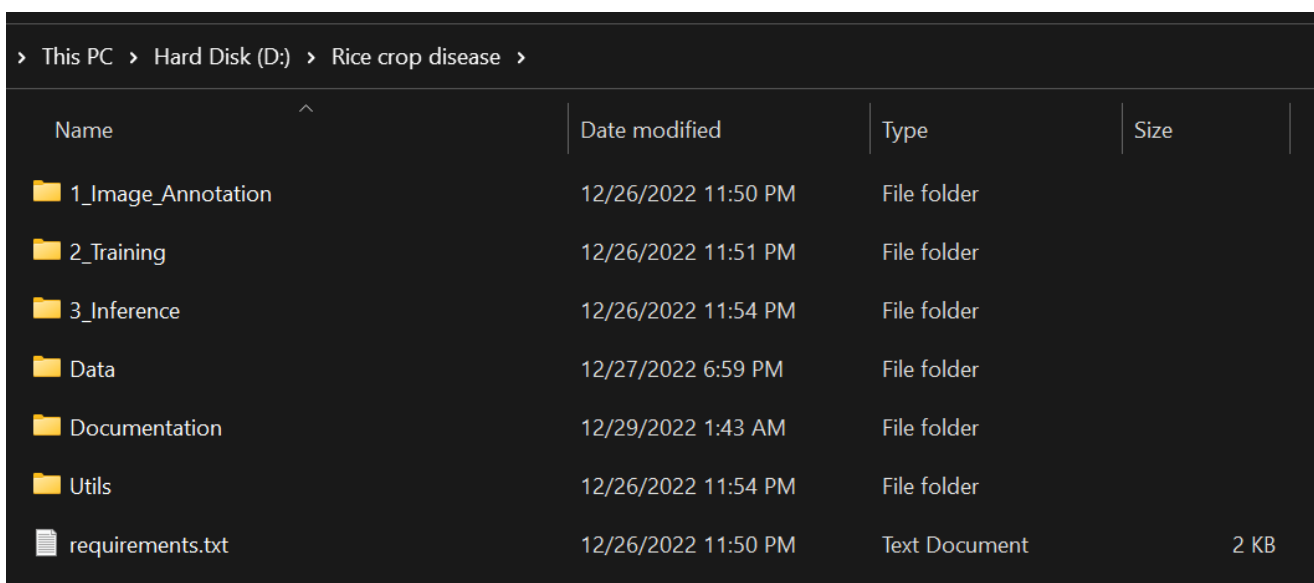
Now collect the images of Diseased Plants from Google or [click here](#) to download the dataset

Link: <https://www.kaggle.com/minhhuy2810/rice-diseases-image-dataset>

Task 1: Create Dataset from Scratch

To train the YOLO object detector on your own dataset, copy your training images to [Rice crop disease/Data/Source_Images/Training_Images`]

Collect at least 50 images of Bacterial Blight, 50 images of Leaf Smut, 50 images of White Tip disease images, and place them in this folder.



> This PC > Hard Disk (D:) > Rice crop disease >				
Name	Date modified	Type	Size	
1_Image_Annotation	12/26/2022 11:50 PM	File folder		
2_Training	12/26/2022 11:51 PM	File folder		
3_Inference	12/26/2022 11:54 PM	File folder		
Data	12/27/2022 6:59 PM	File folder		
Documentation	12/29/2022 1:43 AM	File folder		
Utils	12/26/2022 11:54 PM	File folder		
requirements.txt	12/26/2022 11:50 PM	Text Document	2 KB	

Tip: If you do not already have an image dataset, consider using a Chrome extension such as [FatkunBatchDownloader]

(<https://chrome.google.com/webstore/detail/fatkun-batch-download-ima/nnjjahlikiabnchcpehpcpkdeckfgnohf?hl=en>) which lets you search and download images from Google Images.

Annotate Images:

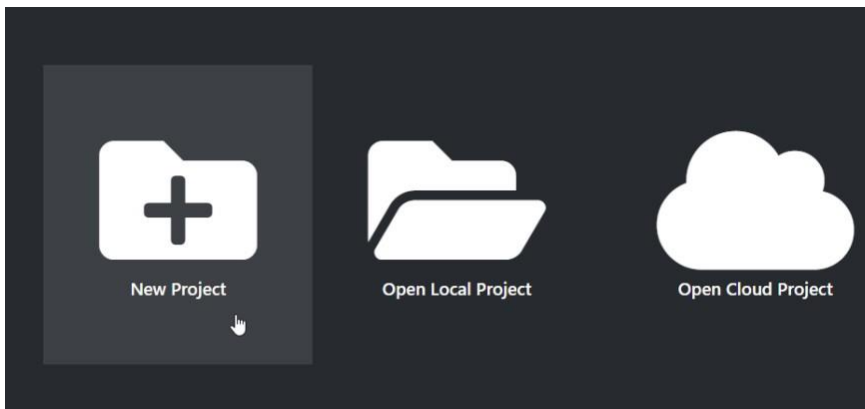
To make our detector learn, we first need to feed it some good training examples. We use Microsoft's Visual Object Tagging Tool (VoTT) to manually label images in our training folder [Rice crop disease/Data/Source_Images/Training_Images].

To achieve decent results annotate at least 100 images for each category. For good results label at least 300 images and for great results label 1000+ images for each category.

Tip: If you increase no of images processing time increases

Annotating Images using VOTT:

Task 1: Create a New Project



Step1:

Create a '**New Project**' and call it '**Annotations**'. It is highly recommended to use '**Annotations**' as your project name. If you like to use a different name for your project, you will have to modify the command line arguments of subsequent scripts accordingly.

Step2:

Under '**Source Connection**' choose '**Add Connection**' and put "**Images**" as '**Display Name**'. Under '**Provider**' choose '**Local File System**' and select [Rice crop disease/Data/Source_Images/Training_Images]) And then '**Save Connection**'. For '**Target Connection**' choose the same folder as for '**Source Connection**'. And Tags of diseases. Hit '**Save Project**' to finish project creation.

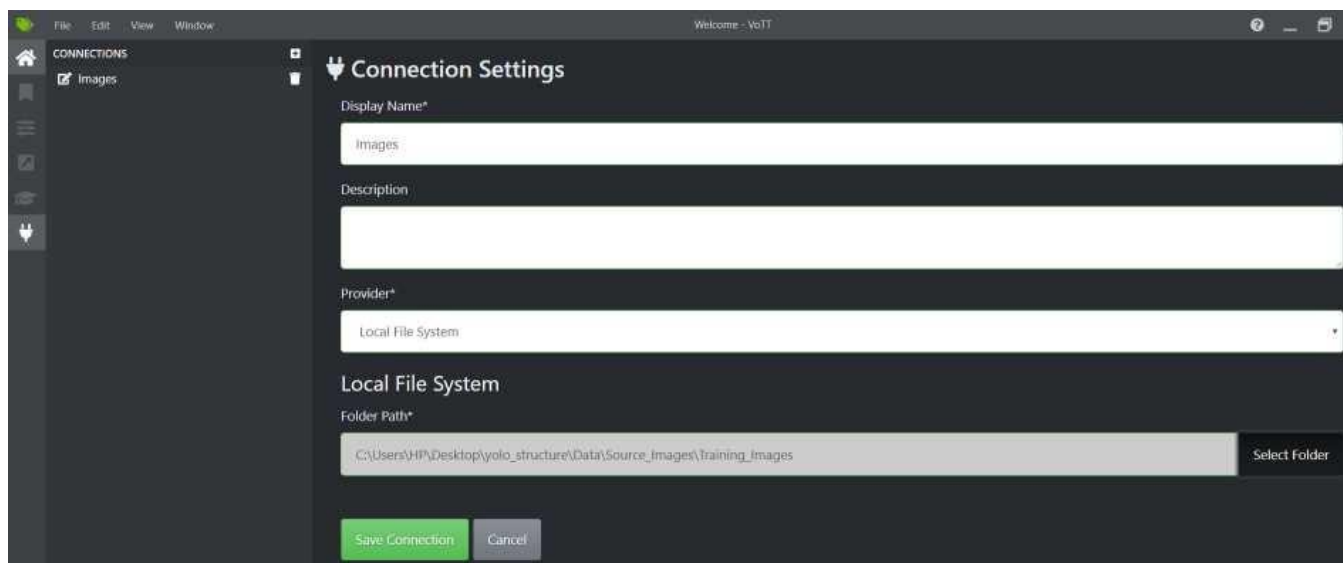


Fig. Connection Settings in VOTT

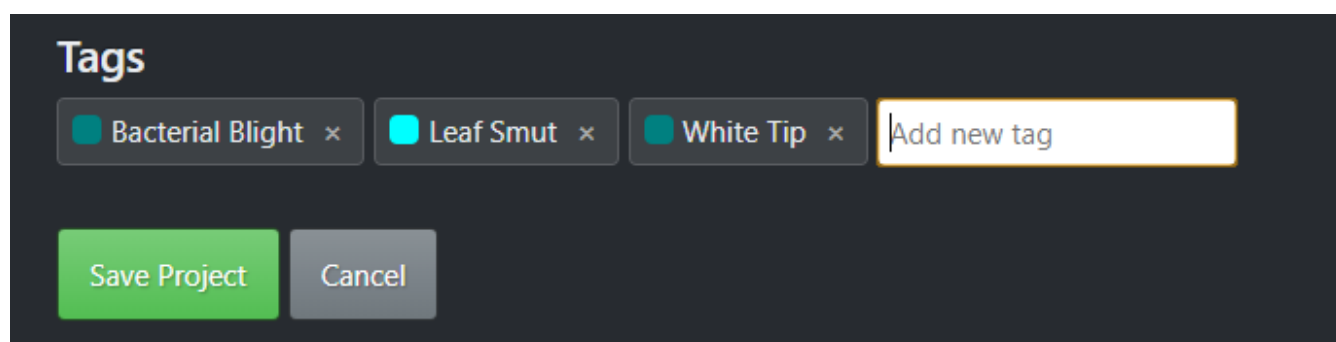


Fig. Add Tags under in VOTT

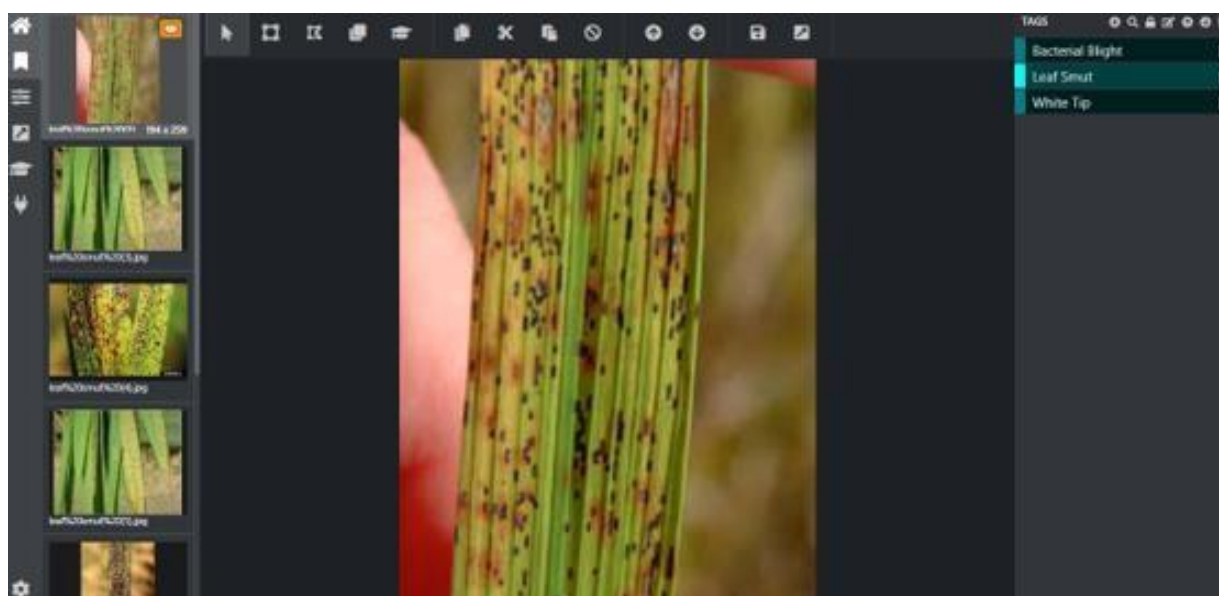


Fig. Annotating Images

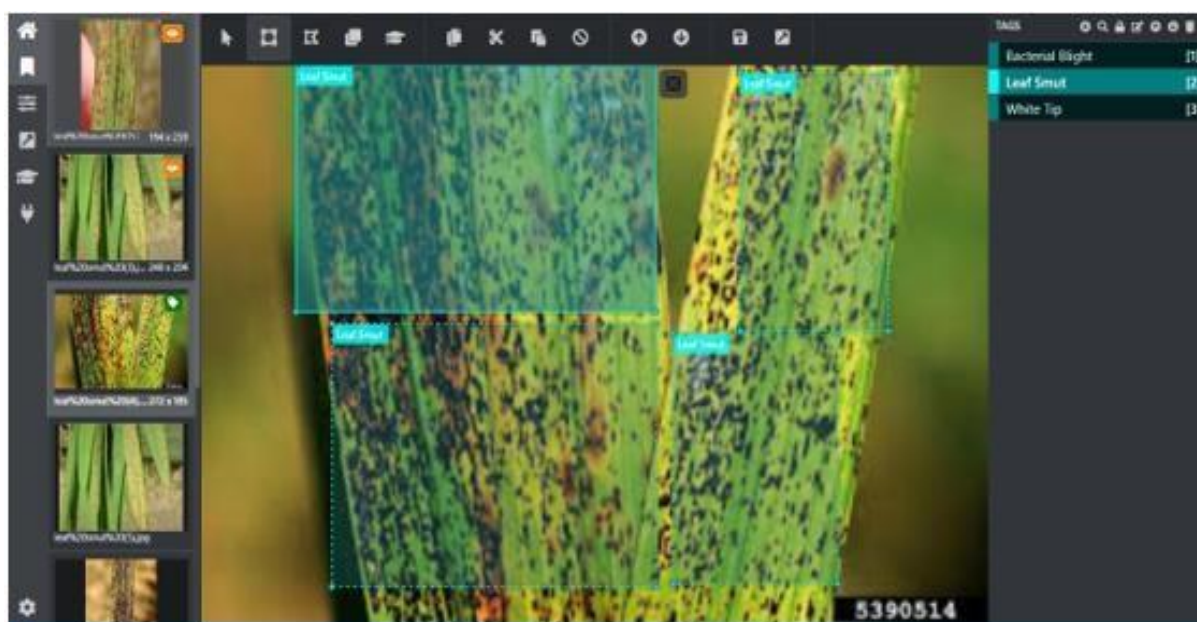
Step3:

Navigate to 'Export Settings' in the sidebar and then change the 'Provider' to 'CommaSeparated Values (CSV)', then hit 'Save Export Settings'.



Step4:

First, create a new tag on the right and give it a relevant tag name. In our example, we choose 'BacterialBlight, Leaf Smut, White Tip'. Then draw bounding boxes around your objects for respective diseases.



Step5:

Once you have labeled enough images press ‘**CRTL+E**’ to export the project. You should now see a folder called [vott-csv-export](/Data/Source_Images/Training_Images/vott-csv-export) directory. Within that folder, you should see a .csv` file called [Annotations-export.csv](/Data/Source_Images/Training_Images/vott-csv-export/Annotations export.csv) which contains file names and bounding box coordinates.



Step 6:

As a final step, convert the VoTT CSV format to the YOLOv3 format. To do so, run the conversion script from within the [Rice crop disease/1_Image_Annotation] folder.

Here ‘**Rice1**’ is the Environment.

To run the file open the anaconda prompt navigate to yolostructure/1_Image_Annotation and run

Convert_to_YOLO_format.py

```
Anaconda Prompt (Anaconda) × + ∨  
  
(base) C:\Users\habee>cd..  
(base) C:\Users>cd..  
(base) C:\>D:  
(base) D:\>conda activate rice1  
(rice1) D:\>cd D:\Rice crop disease\1_Image_Annotation  
(rice1) D:\Rice crop disease\1_Image_Annotation>python Convert_to_YOLO_format.py
```

The script generates two output files: [data_train.txt](/Data/Source_Images/Training_Images/vott-csv-export/data_train.txt) located in the

[Rice crop disease \Data\Source_Images\Training_Images\vott-csv-export]
(/Data/Source_Images/Training_Images/vott-csv-export) folder and[data_classes.txt]
(\Data\Model_Weights\data_classes.txt) located in the
[Rice crop disease\Data\Model_Weights](\Data\Model_Weights\) folder.

To list available command line options run `python Convert_to_YOLO_format.py -h`.

Link : <https://youtu.be/uDWgWJ5Gpwc>

Training Yolo: In this milestone we will train our model using YOLO weights.

Task 1: Download and Convert Pre-Trained Weights

Step1:

Using the training images located in[Rice crop disease \Data\Source_Images\Training_Images]
(\Data\Source_Images\Training_Images)and the annotation file
[data_train.txt](\Data\Source_Images\Training_Images\vott-csv-export) which we have created in
the [previous step](1_Image_Annotation/) we are now ready to train our YOLOv3detector.

Step2:

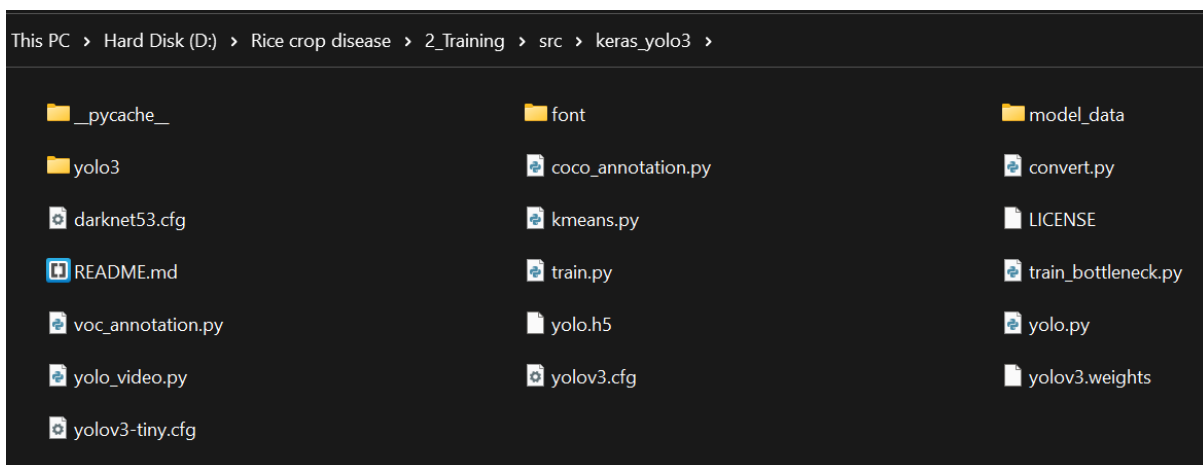
Before getting started download the pre-trained YOLOv3 weights and convert them to the Keras format.To run both steps run the download and conversion script from within the

[Rice crop disease \2_Training](\2_Training/) directory

Run python Download_and_Convert_YOLO_weights.py

```
(rice1) D:\Rice crop disease>cd 2_Training  
(rice1) D:\Rice crop disease\2_Training>python Download_and_Convert_YOLO_weights.py|
```

- This will download yolov3 weights and convert them into keras format. These files will be downloaded in [Rice crop disease \2_Training\src\keras_yolo3].
- Tip: This step would take some time as it needs to download the weights and convert into keras format.



- The weights are pre-trained on the [ImageNet 1000 dataset](<http://image-net.org/challenges/LSVRC/2015/index>) and thus work well for object detection tasks that are very similar to the types of images and objects in the ImageNet 1000 dataset.

Task 2:

To start the training, run the training script from within the [Rice crop disease \2_Training\src\keras_yolo3] directory

Run python Train_YOLO.py

```
(rice1) D:\Rice crop disease>cd 2_Training  
(rice1) D:\Rice crop disease\2_Training>python Train_YOLO.py|
```

Depending on your set-up, this process can take a few minutes to a few hours. The final weights are saved in [`Rice crop disease/Data/Model_weights``](`/Data/Model_weights`). To list available command line options run `python Train_YOLO.py -h``.

Testing the model:

In this step, we test our detector on infected crop images located in [`yolostructure/Data/Source_Images/Test_Images``](`/Data/Source_Images/Test_Images`). If you like to test the detector on your own images or videos, place them in the [`Test_Images``](`/Data/Source_Images/Test_Images`) folder.

Run python Detector.py

```
(rice1) D:\Rice crop disease>cd 3_Inference  
(rice1) D:\Rice crop disease\3_Inference>python Detector.py|
```

Output:

The result of test images is stored in the test images folder.

D:\Rice crop disease\Data\Source_Images\Test_Image_Detection_Result

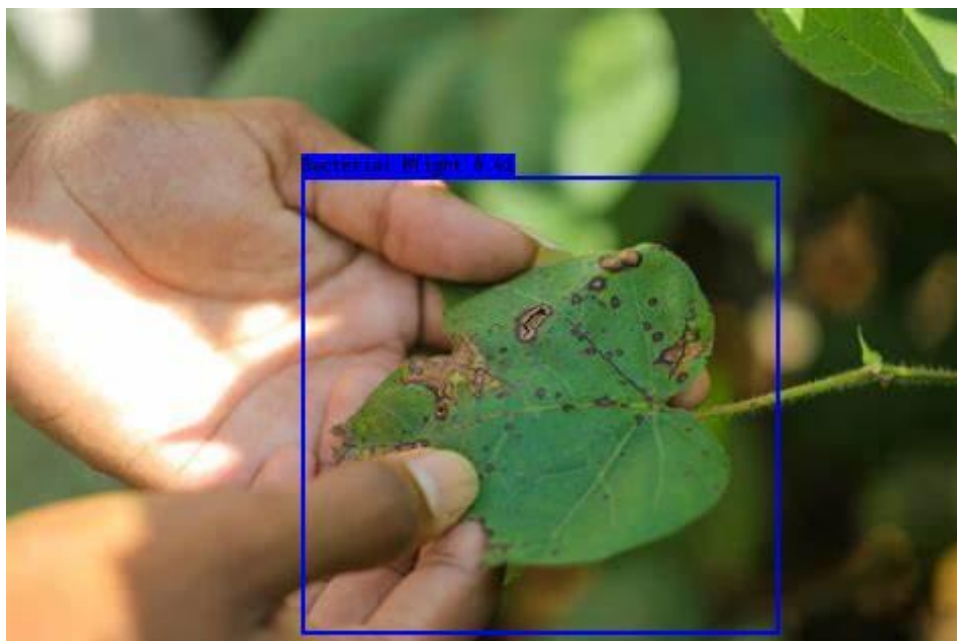


Fig. Bacterial Blight

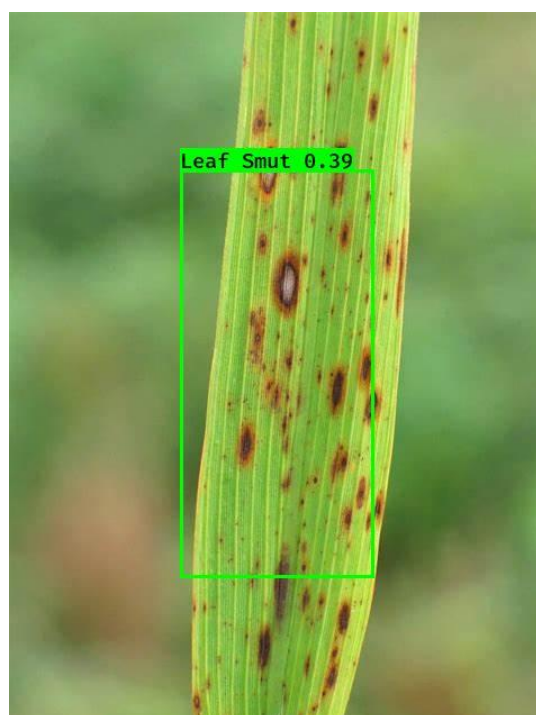


Fig. Leaf Smut



Fig. White Tip