**CODE FOR GOOD - NTT**:

**OPEN INNOVATION:**

Problem Statement:

**PLANT DISEASE DETECTION:**

In countries like India, whose primary occupation is agriculture faces a huge loss when the crops get affected by any disease. These diseases attack the crops in various stages and destroy the entire production. Since most of the diseases are transmitted from one crop to other there is much need to detect the exact type of disease the crop has been affected so that farmers can take the required steps to the “save the crops” and production. But detecting the kind of disease that a crop has been affected is very difficult to farmers since there are various kinds of diseases.

**Solution:**

Detecting the type of plant disease can be done accurately when all the diseases related to a plant or a crop is known and to carry this type of data is really a difficult task for a farmer. This problem can be solved using Machine Learning/Deep Learning. Our model uses the photo of plant leaf and detects whether the plant is healthy or not. If the plant is unhealthy then this model also shows what kind of disease that attacked the plant.

Here we trained the model to detect the two kinds of diseases in potato plant. The two types of

diseases are:

1. EARLY\_BLIGHT

2. LATE\_BLIGHT

**WORKING OF MODEL:**

One need to upload the photo of potato’s plant leaf then the model shows whether the leaf is healthy or unhealthy. In case of unhealthy the model detects type of disease that potato plant got affected.

**Data Description:**

The dataset is taken from: “ <https://www.kaggle.com/emmarex/plantdisease> “.

From the above link we took data related to potato plants disease only.

**Implementation**:

Here we did the *custom training* to the model “*faster\_rcnn\_resnet101\_coco\_11\_06\_2017*"[Custom object detection using tensor flow API].

Note: Model training and testing are performed in Google Colaboratory.

STEP-1:

1. Download the given dataset from the above link

2. To begin, install the Tensor flow and all of required dependencies.

-pip install tensorflow

3. Now clone the tensorflow models through command:

-git clone https://github.com/tensorflow/models

4. Next head to protoc releases page and download the "protoc-3.4.0-win32.zip", extract it and you will find protoc.exe in the bin directory

5. Move the "protoc.exe" file to models->research (i.e. into research directory)

6. Now execute the command:

- "protoc object\_detection/protos/\*.proto --python\_out=."

STEP-2:

1. Collect a few hundred images that contain your object - The bare minimum would be about 100, ideally more like 500+, but, the more images you have, the more tedious step 2 is...

2. Annotate/label the images using LabelImg.

3. The above step is basically drawing boxes around your object(s) in an image.

4. This labelImg automatically will create an XML file that describes the object(s) in the pictures.

3. Split this image data into train/test samples

4. Create train.csv and test.csv file for above samples.

-this can be done through xml\_to\_csv.py file which is available from link:" https://github.com/datatrain/raccoon\_dataset/blob/master/xml\_to\_csv.py"

5. Create the new folder named "data" and now place the generated ".csv" files in data folder.

STEP-3:

1. Firstly change the directory to:"models/research"

1. Finally, install the object\_detection library formally by doing the following from within the models->research directory:

-python setup.py install

2. Now run the generate\_tfrecord.py script. We will run it twice, once for the train TFRecord and once for the test TFRecord.

-python generate\_tfrecord.py --csv\_input=data/train\_labels.csv --output\_path=data/train.record

-python generate\_tfrecord.py --csv\_input=data/test\_labels.csv --output\_path=data/test.record

3. Now generated files are available in data folder.

Note: Don’t change the directory path

STEP-4:

1. TensorFlow has quite a few pre-trained models with checkpoint files available, along with configuration files.

2. Here we are using "faster\_rcnn\_resnet101\_coco\_11\_06\_2017".

3. Using the following checkpoint and configuration file

config\_file :-wget https://raw.githubusercontent.com/tensorflow/models/master/object\_detection/samples/configs/faster\_rcnn\_resnet101\_coco\_11\_06\_2017.config

checkpoint :-wget http://download.tensorflow.org/models/object\_detection/faster\_rcnn\_resnet101\_coco\_11\_06\_2017

4. Create new directory "training"

5. Put the config in the training directory.

6. Extract the faster\_rcnn\_resnet101\_coco\_11\_06\_2017 in the research directory.

8. Now in this file search for all of the PATH\_TO\_BE\_CONFIGURED points and change them:

--->fine\_tune\_checkpoint: "faster\_rcnn\_resnet101\_coco\_11\_06\_2017/model.ckpt"

--->train\_input\_reader: {

tf\_record\_input\_reader {

` input\_path: "data/train.record"

}

label\_map\_path: "data/object-detection.pbtxt"

}

---->eval\_input\_reader: {

tf\_record\_input\_reader {

input\_path: "data/test.record"

}

label\_map\_path: "training/object-detection.pbtxt"

9. Create object-detection.pbtxt file and save it in training folder:

-the format of object-detection is as follows: item {

id: 1

name: 'healthy'

}

STEP-5:

1. In this step delete the file train.py in research directory.

2. Copy the train.py file that is available in "research/object\_detection/legacy"

3. Now from slim directory which is in research directory, copy all the files in it and paste it in research directory

4. Now we start the training through command:

-python3 train.py --logtostderr --train\_dir=training/ --pipeline\_config\_path=faster\_rcnn\_resnet101\_coco\_11\_06\_2017

5. This takes few time (based on resources available)

6. This generates data in training directory which consist of checkpoints that can be viewed via Tensor Board.

STEP-6:

1. Now finally we test our model running through few commands.

2. Firstly delete the file "export\_inference\_graph.py" available in research directory.

3. Now from research/object\_detection copy the "export\_inference\_graph.py" and paste in research directory.

4. And run the command:

###->python3 export\_inference\_graph.py \

--input\_type image\_tensor \

--pipeline\_config\_path training/faster\_rcnn\_resnet101\_coco\_11\_06\_2017.config \

--trained\_checkpoint\_prefix training/model.ckpt-10856 \ #here model.ckpt-10856 is one of checkpoint available in training directory#

--output\_directory mydata

5. This command generates the directory "mymodel" (model name)

6. Upload the test.csv file in research directory

6. Clean the test\_images directory data in research directory.

6. Place the image need to be tested in test\_images

6. Now load the object-detection.ipynb file from research/object\_detection

7. Make the following changes in .ipynb file:

MODEL\_NAME = 'mymodel'

PATH\_TO\_CKPT = MODEL\_NAME + '/frozen\_inference\_graph.pb'

PATH\_TO\_LABELS = os.path.join('training', 'object-detection.pbtxt')

NUM\_CLASSES = 3 #####['no of classes in data , here it is 3']#####

\*\*delete next following cell

(as required)TEST\_IMAGE\_PATHS = [os.path.join(PATH\_TO\_TEST\_IMAGES\_DIR, '{}.jpg'.format(i)) for i in range (3, 8) ]

8. Now run all cells.

9. For bounding box data, iterate over box variable at last cell

10. At last cell we see visualized data i.e. testing image with bounding box

over object.

Reference:

Video link: "https://pythonprogramming.net/introduction-use-tensorflow-object-detection-api-tutorial/"