**Date: 02-03-2020**

**Author: MD Fakrul Islam**

**fakrul.islam@tsi.com.bd**

**Project Brief:**

Detecting specific diseases (listed below) of Tomato, Potato and Maize through the power of AI. Develop the AI model using TensorFlow, optimize and inference the model through Intel distributed OpenVino toolkit, execute optimization and inferencing in local workstation as well as in the Intel® DevCloud, and finally in edge computing nodes for selective target/platforms.

List of diseases captured or worked in this project,

Tomato:

1. Early Blight
2. Late Blight
3. Leaf Curl
4. Leaf Mold
5. Healthy Leaf

Potato:

1. Early Blight
2. Late Blight
3. Healthy Leaf

Maize:

1. Common Rust
2. Gray Leaf Spot
3. Northern Leaf Blight
4. Healthy Leaf

**Source of Master Dataset:**

The master dataset has been collected from:

<https://github.com/spMohanty/PlantVillage-Dataset>

**How to minimize the bias for a particular class:**

The distribution of the images were not uniform in the the master dataset. To avoid bias for a particular class or diseases, we applied below distribution through image augmentation so that each class has a similar number of images. You will get this distribution file in the git in the name of, “aug\_Classification\_Images.xlsx”.

|  |  |  |
| --- | --- | --- |
| Classification | No of images in Kaggle Dataset | Total Augmentation |
| Potato\_\_\_Early\_blight | 1000 | 0 |
| Potato\_\_\_Late\_blight | 1000 | 0 |
| Potato\_\_\_healthy | 152 | 848 |
| **MAX** | **1000** |  |
|  |  |  |
| Tomato\_\_\_Early\_blight | 1000 | 4357 |
| Tomato\_\_\_Late\_blight | 1909 | 3448 |
| Tomato\_\_\_Leaf\_Curl\_Virus | 5357 | 0 |
| Tomato\_\_\_Leaf\_Mold | 952 | 4405 |
| Tomato\_\_\_healthy | 1591 | 3766 |
| **MAX** | **5357** |  |
|  |  |  |
| Corn\_(maize)\_\_\_Common\_rust\_ | 1192 | 0 |
| Corn\_(maize)\_\_\_Gray\_leaf\_spot | 513 | 679 |
| Corn\_(maize)\_\_\_Northern\_Leaf\_Blight | 985 | 207 |
| Corn\_(maize)\_\_\_healthy | 1162 | 30 |
| **MAX** | **1192** |  |

The augmentation script name is, “**aug.py**” which is also added into git. To run this script, you need to provide the location of images, the location of augmented images as well as the number of augmented images. For example:

folder\_path = '/.............../Healthy' # Source of images

augmented\_path = '/……./Healthy' #Destination of augmented images

num\_files\_desired = 3765 #No of augmented files.

The script “**aug.py**” applied 3 techniques randomly for image augmentation

1. Random rotation
2. Random noise and
3. Random horizontal flip

**How to split the master dataset into training and validation:**

To split the dataset into training and validation, Python library was used:

<https://pypi.org/project/split-folders/> under the MIT open license. There is a script added in the git as a name of “**train\_valid.py**” to split the master dataset.

For example, ratio = (.80, .20) will split the master dataset into 80% for training and 20% for validation.

**How to collect the augmented and split dataset for this project:**

After augmentation and splitting (training and validation) of the master dataset, you can download the working dataset for this project from

<https://drive.google.com/open?id=1n5b9eZ1pHJAc4estMoKYGYdfc_u4YPhM>

**General information on Mean Validation accuracy:**

Model developed by images with resolution of 32\*32 pixels will have the lowest mean validation prediction accuracy, around, ~90%

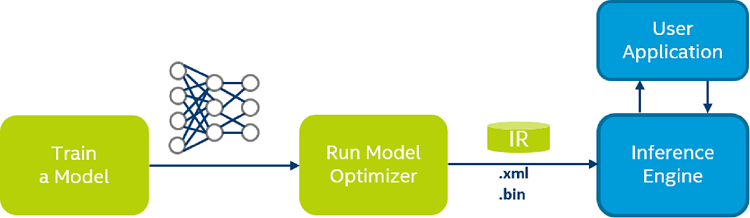
comparing with

Model developed by images with resolution of 224\*224 pixels will have the highest mean validation prediction accuracy around, ~ 98%.

The model was developed through the strategy of transfer learning. We used the vgg16 pre-trained model to develop the model.

**General information of optimization and inferencing:**

The general idea of optimization and inferencing of a deep neural network or model can be depicted as below:



Source:

<https://docs.openvinotoolkit.org/latest/_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html>

1. Model optimizer can optimize models developed by ONNX, Caffe, TensorFlow, MXNet and Kaldi.
2. Optimization abstracts the model framework whether it is developed by TensorFlow or ONNX, Caffe, MXNet and Kaldi to a set of neutral files, model.bin, model.xml and model.mapping.
3. Inferencing engine delivers the compute power of Intel processors (CPU or GPU) when we want to inference on inputs for predictions or detections..
4. Optimization and inferencing process never improves the model accuracy or loss, it only improves the utilization of the compute power.

For example, a model developed by TensorFlow having a mean validation accuracy of 90% will never be changed (increased or decreased) after optimization and inferencing.

The computing time for inferencing will be varied when tested/implemented on different Intel platforms like:

|  |  |  |
| --- | --- | --- |
| **Group ID** | **RAM** | **Compute Devices** |
| idc001skl | 8 GB | Intel Core i5 6500TE Intel HD Graphics 530 |
| idc002mx8 | 8 GB | Intel Core i5 6500TE Intel Vision Accelerator Design (HDDL-R) Intel HD Graphics 530 |
| idc003a10 | 8 GB | Intel Core i5 6500TE Intel Vision Accelerator Design (HDDL-F) Intel HD Graphics 530 |
| idc004nc2 | 8 GB | Intel Core i5 6500TE Intel Neural Compute Stick 2 Intel HD Graphics 530 |
| idc006kbl | 8 GB | Intel Core i5 7500T Intel HD Graphics 630 |
| idc007xv5 | 32 GB | Intel Xeon E3 1268L v5 Intel HD Graphics 505 |
| idc008u2g | 4 GB | Intel Atom x7-E3950 Intel Neural Compute Stick 2 Intel HD Graphics 505 |
| idc009jkl | 8 GB | Intel Core i5 7500 Intel HD Graphics 630 |
| idc010jal | 4 GB | Intel Atom x7-E3950 Intel HD Graphics 505 |
| idc011ark2250s | 8 GB | Intel Core i5 6442EQ Intel HD Graphics 530 |
| idc013ds580 | 2 GB | Intel Atom x7-E3950 Intel HD Graphics 505 |
| idc014upxa10fx1 | 16 GB | Intel Core i7 8665UE Intel UHD Graphics 620 |
| idc015ai5 | 8 GB | Intel Core i5 8365UE Intel UHD Graphics 620 Intel Vision Accelerator Design (HDDL-R) |
| idc016ai7 | 16 GB | Intel Core i7 8665UE Intel UHD Graphics 620 Intel Vision Accelerator Design (HDDL-R) |

Source,

<https://devcloud.intel.com/edge/resource_docs/selecting_targets/edge-nodes>

Please follow the guideline sequentially,

1. Readme\_1\_Model\_Development\_LocalWorkStation\_or\_Colab
2. How to prepare your workstation to utilize OpenVino toolkit
3. How to develop a TensorFlow model in your local work-station or colab.

2. Readme\_2\_Model\_Optimization\_Inferencing\_LocalworkStation (OpenVino, 2019 R3.1, Build date:23 Oct 2019)

1. How to optimize a TensorFlow model (Tomato, Potato and Maize) in your local workstation.
2. How to inference an optimized TensorFlow model in your work-station.

3. Readme\_3\_Model\_Optimization\_Inferencing\_intel\_dev\_cloud

1. How to optimize a TensorFlow model (Tomato, Potato and Maize) in dev cloud.
2. How to inference an optimized TensorFlow model (Tomato, Potato and Maize) in dev cloud.

4. Readme\_4\_Model\_Inferencing\_intel\_dev\_cloud\_edge\_computing

1. How to inference an optimized TensorFlow model (Tomato, Potato and Maize) in edge computing node for selective platforms.
2. Verify the computation in selective platforms to ensure reference implementation.

**Reference Implementation in:**

<https://www.ieiworld.com/en/product/model.php?II=525>

TANK-870-Q170

and

<https://www.aaeon.com/en/p/iot-gateway-maker-boards-up-core-plus>

Atom™ x7-E3950 Processor SoC