



Department of Computer Science
Winter 2020
CS 890ES Data Science Fundamentals

Produce Item Recognition Application For Retail Stores Based on Machine Learning

User Guide for Engineers, Analysts and Executives

Prepared By:
Adarsh Koppa Manjunath
Stephen Oluwatobi Bello
Iveatu Jane Obioha

Table of Content

- 1) Introduction**
- 2) Data Analytic Life Cycle**
- 3) Source Code User Guide for Engineers and Analysts**
- 4) Operationalize User Guide for Executives, Engineers, and Analysts**
- 5) Maintenance User Guide for Engineers and Analysts**

1) Introduction

Produce is the generalized term for farm raised crops which includes fruits and vegetables. This same term is well adopted by most retail stores in Canada to describe the section where fruits and vegetables are kept in stores. Produce section in retail stores have items which are not UPC tagged thus making the self-checkout process difficult for both the customer and employees. We approached this problem with the help of image recognition with the help of data analytic life cycle. In this file, we will be explaining in detail how to use this project by considering different users like engineer, analysts and executives.

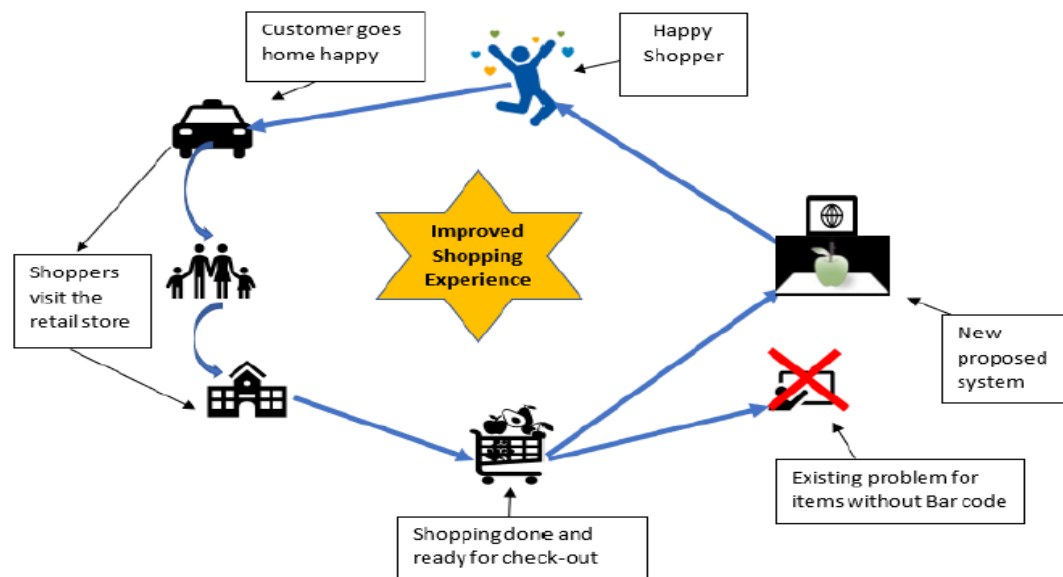


Fig 1: Solution Overview

2) Data Analytic Life Cycle

For this project we have followed the data analytic life cycle. Which consist of six phases. Understanding these phases is necessary for engineer and analysts before moving to user guide.

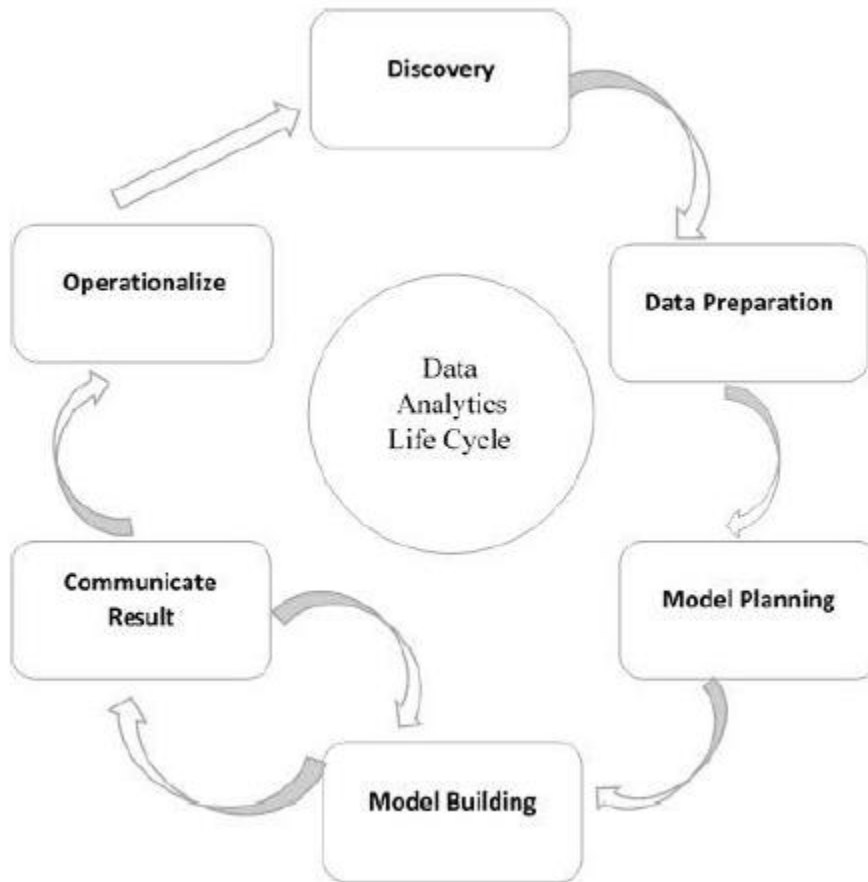


Fig 2- Data Analytic Life Cycle

- Discovery- is the phase where we met retail store managers and asked for their current problems in the store. After going through a couple of meeting and interview we finally decided to address non-UPC tagged produce item issues
- Data Preparation- in this phase, we captured videos of fruits and vegetables, and converted them to images, also multiplied a few internet images using augmentation. Dataset is prepared with 24 classes, 800 images for each class in train, and 320 images for each class in valid and test.

S/N	PROPERTIES	DESCRIPTION
1	Total number fruits	8
2	Total number of Vegetables	16
3	Train Dataset	19181
4	Validation Dataset	7680
5	Test Dataset	7680
6	Total number of images in Dataset	34541
7	Image Size	3GB (80:20)
8	Image filename	Parent foldername_***.jpg
9	Image Dimension	100 x 100 Pixels

○ Table 1: Data Set Summary

- Model Planning- since it was a classification and image recognition task, CNN is been used. We decided to use pretrained models like VGG16, Resnet50 and CNN from scratch
- Model Building: Involves Data Reading, Image Resizing and Rescaling, Pretrained Model Loading, Model Training and Testing, and Testing Metrics.
- Communication of Results: Results of three models been compared here.
- Operationalize: Kivy application is developed here for prediction.

3) Source Code User Guide for Engineers and Analysts

3.1) Prerequisites:

Required Hardware Configuration - NVIDIA® GPU card with CUDA® Compute Capability 3.5 or higher

Required Software Configuration - NVIDIA® GPU drivers —CUDA 10.1 requires 418.x or higher. CUDA® Toolkit —TensorFlow supports CUDA 10.1 (TensorFlow >= 2.1.0) CUPTI ships with the CUDA Toolkit. cuDNN SDK (>= 7.6)

After making sure of hardware and software environment following steps must be performed-

step a: Install Anaconda from <https://www.anaconda.com/>

step b: Open Anacond Navigator and click on console option

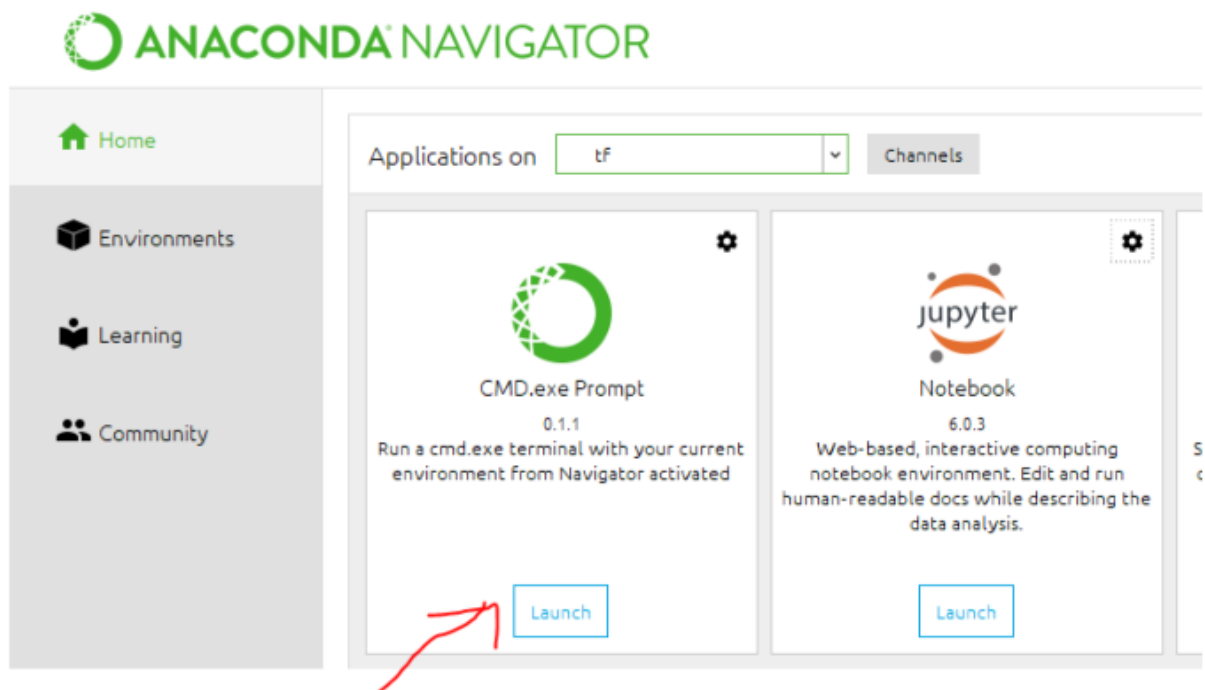


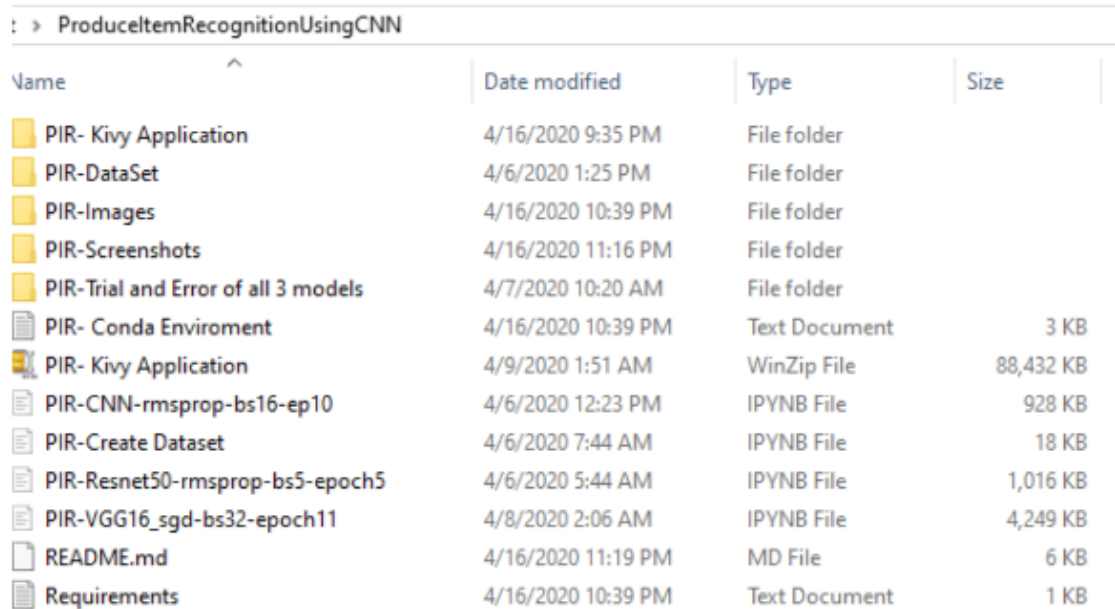
Fig 3- Anaconda Console

step c: Install tensorflow in anaconda with the following commands
conda create -n tf tensorflow
conda activate tf

step d: Clone github repository from the command line

git

clone <https://github.com/AdarshKoppManjunath/ProduceItemRecognitionUsingCNN.git>



Name	Date modified	Type	Size
PIR- Kivy Application	4/16/2020 9:35 PM	File folder	
PIR-DataSet	4/6/2020 1:25 PM	File folder	
PIR-Images	4/16/2020 10:39 PM	File folder	
PIR-Screenshots	4/16/2020 11:16 PM	File folder	
PIR-Trial and Error of all 3 models	4/7/2020 10:20 AM	File folder	
PIR- Conda Enviroment	4/16/2020 10:39 PM	Text Document	3 KB
PIR- Kivy Application	4/9/2020 1:51 AM	WinZip File	88,432 KB
PIR-CNN-rmsprop-bs16-ep10	4/6/2020 12:23 PM	IPYNB File	928 KB
PIR-Create Dataset	4/6/2020 7:44 AM	IPYNB File	18 KB
PIR-Resnet50-rmsprop-bs5-epoch5	4/6/2020 5:44 AM	IPYNB File	1,016 KB
PIR-VGG16_sgd-bs32-epoch11	4/8/2020 2:06 AM	IPYNB File	4,249 KB
README.md	4/16/2020 11:19 PM	MD File	6 KB
Requirements	4/16/2020 10:39 PM	Text Document	1 KB

Fig 5- ProuceItemRecognitionUsingCNN Repository

step e: execute `cd ProduceItemRecognitionUsingCNN` to navigate to repository and install required libraries with the below command

`pip install -r requiremnets.txt`

step f: Open Jupiter notebook with the tf environment

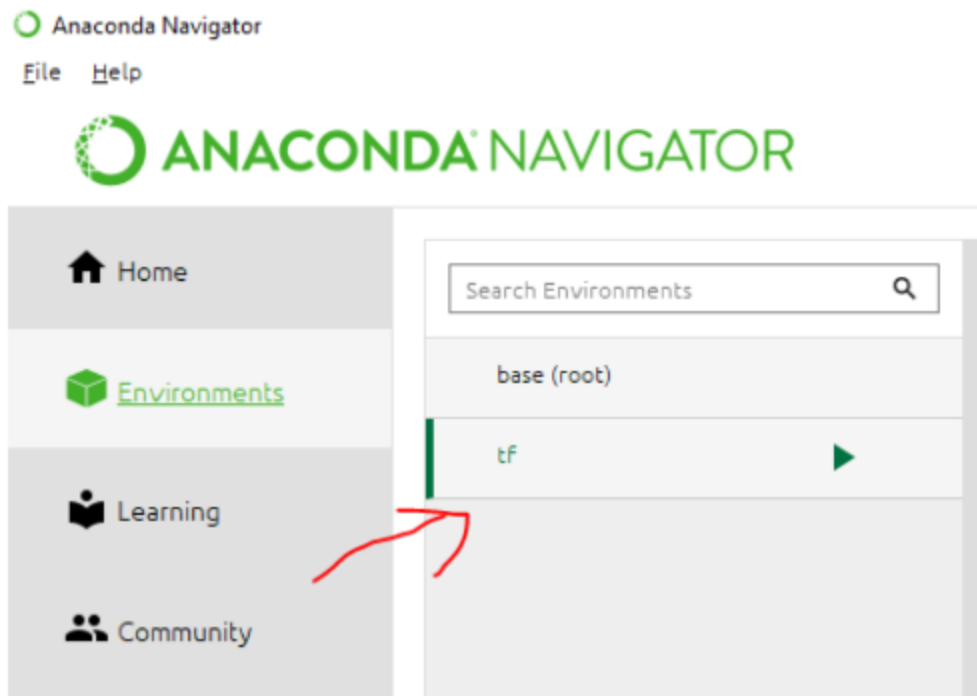


Fig 4- tf environment

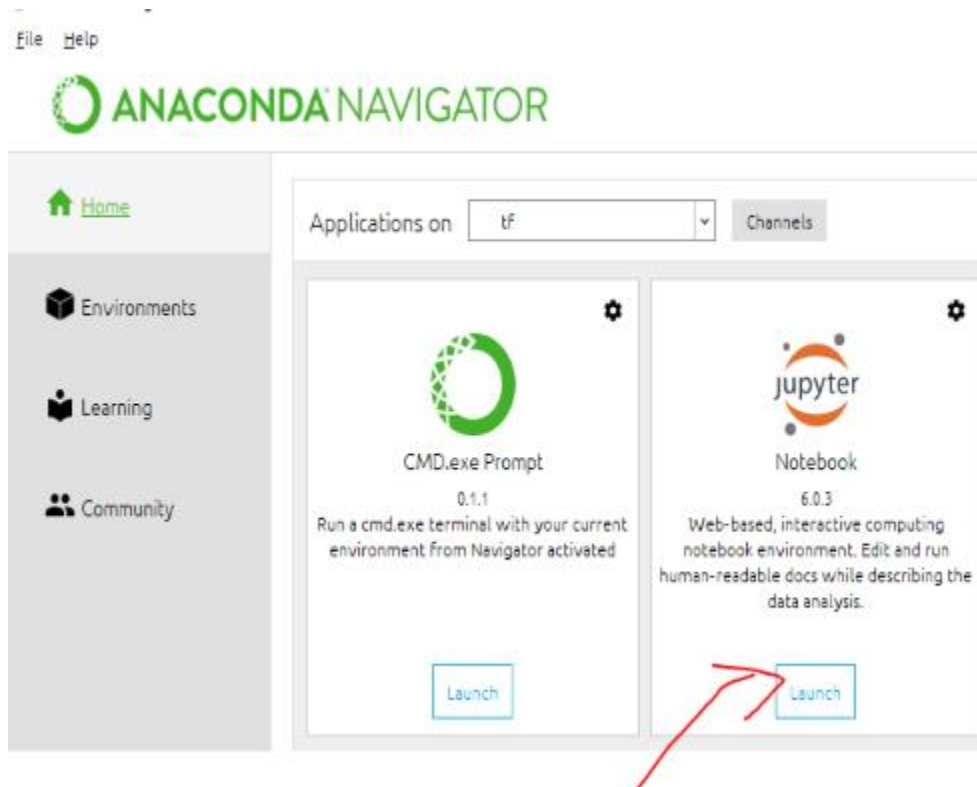


Fig 5 – Jupyter notebook

3.2) Data Set Creation: As said in data analytic life cycle, video captured for this step and few internet images were chosen. Videos were converted to images using **python opencv library** and internet images are multiplied using keras augmentation technique.

To perform this, You have to have new video and image saved and use the below code file .

<https://github.com/AdarshKoppManjunath/ProduceItemRecognitionUsingCNN/blob/master/PIR-Create%20Dataset.ipynb>

3.2.1) This code will help you to create directory for new class in all three train, valid and test set. With little modification.

```
import os

train_path = 'C:/Users/Owner/Desktop/Produce Item Recognition/DataSet/TrainSet/'
valid_path = 'C:/Users/Owner/Desktop/Produce Item Recognition/DataSet/ValidationSet/'
test_path = 'C:/Users/Owner/Desktop/Produce Item Recognition/DataSet/TestSet/'

folders=['Blueberry','Garlic','Yellow Potato','Spinach','Yellow Onion','Apple', 'Avocado', 'Banana', 'bell pepper','Cabbage','Brocolli', 'Canada Pear', 'Carrot', 'Green Peas', 'Green Pepper', 'Lettuce', 'Mangoes', 'Okra', 'Orange', 'Pineapple', 'Red Chilli', 'Red Onions', 'Spring Onion', 'Tomato']

path=[train_path,valid_path,test_path]

for dataset_path in path:
    create_folder(dataset_path,folders)
```

Code Snippet 1- To create new class folders

Set train, valid and test path according to your local set up and mention the new class in folders and remove the old one.

3.2.2) After having directory ready. Video to image convert function needs to be called. Make sure frame rate, number of images, source and destination path is set according to your requirements,

```
source_path="C:/Users/Owner/Desktop/Produce Item Recognition/Fruits and Vegetables Videos/"
destination_path=valid_path
frame_rate=0.12
number_of_images=330
image_name="valid_00_1"
folders=['Spinach','Apple','Pineapple','Carrot','Tomato','Orange','Brocolli']
video_to_image_convert(destination_path,source_path,folders,frame_rate,number_of_images,image_name)
```

Code Snippet 2- To call video to image function

3.2.3) Image Generator function multiplies the internet image. This step is required only if you want your dataset to have diverse image. Otherwise this step can be ignored.

3.2.4) Finally we are limiting the class in train to have only 800 images and 320 in valid and test classes by removing random samples.

After having all the above said modification in the code, you can execute all the cells.

3.3) Data Preparation, Model Building and Communication of Results

All these steps are done in the same notebook, there are totally 3 notebooks for each model which are below.

VGG16

https://github.com/AdarshKoppManjunath/ProduceItemRecognitionUsingCNN/blob/master/PIR-VGG16_sgd-bs32-epoch11.ipynb

Resnet50

<https://github.com/AdarshKoppManjunath/ProduceItemRecognitionUsingCNN/blob/master/PIR-Resnet50-rmsprop-bs5-epoch5.ipynb>

CNN-Scratch

<https://github.com/AdarshKoppManjunath/ProduceItemRecognitionUsingCNN/blob/master/PIR-CNN-rmsprop-bs16-ep10.ipynb>

Before executing these files make sure paths are set according to your local environment. Below code snippets shows the place where code needs to be changed.

```
train_path = 'C:/Users/Owner/Desktop/Produce Item Recognition/DataSet/TrainSet/'
valid_path = 'C:/Users/Owner/Desktop/Produce Item Recognition/DataSet/ValidationSet/'
test_path = 'C:/Users/Owner/Desktop/Produce Item Recognition/DataSet/TestSet/'
```

```
from keras.callbacks import ModelCheckpoint, EarlyStopping
checkpoint = ModelCheckpoint("C:/Users/Owner/Desktop/Produce Item Recognition/vgg16.h5", monitor='val_accu
```

Code Snippet 3: Set data set paths

After changing these variables, you can execute all cells. Parameters like batch size, optimizer, image size, augmentation parameter, epochs etc., can be changed according the requirements. At the end, you will also get model in .h5 format for all 3 notebooks. These models will be saved with the algorithm and epochs. For example, VGG16 model will be saved as VGG16_11.h5 where 11 is number of epochs. We will be using the best model for the operationalize.

4) Operationalize User Guide for Executives, Engineers and Analysts.

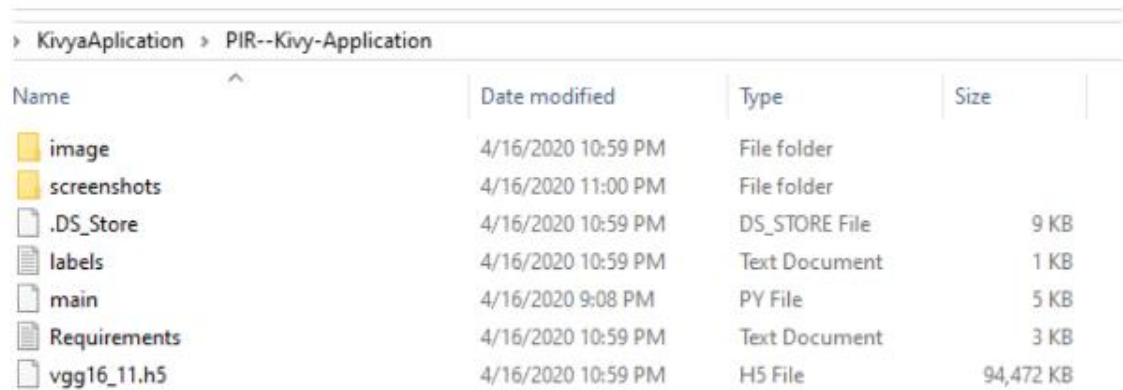
Kivy Application

If you are executing only application and don't bother to run model, you might have to follow only below few steps and can ignore all above steps. If you have already installed all libraries mentioned in above steps, no need to follow below steps.

step a: Follow step a to c from the section 3.1

step b: Download kivy application code from the below command

git clone <https://github.com/AdarshKoppManjunath/PIR--Kivy-Application.git>



KivyaApplication > PIR--Kivy-Application				
Name	Date modified	Type	Size	
image	4/16/2020 10:59 PM	File folder		
screenshots	4/16/2020 11:00 PM	File folder		
.DS_Store	4/16/2020 10:59 PM	DS_STORE File	9 KB	
labels	4/16/2020 10:59 PM	Text Document	1 KB	
main	4/16/2020 9:08 PM	PY File	5 KB	
Requirements	4/16/2020 10:59 PM	Text Document	3 KB	
vgg16_11.h5	4/16/2020 10:59 PM	H5 File	94,472 KB	

Fig 6 – Kivy Application Repository

Repository comes with the model, labels and code. Every time when there is add or removal of produce item, label needs to be updated and models should be latest.

step c: Execute any of below command to install kivy

conda install -c conda-forge kivy

conda install -c conda-forge/label/cf201901 kivy

conda install -c conda-forge/label/cf202003 kivy

Navigate to ***cd PIR--Kivy-Application***

Execute ***python main.py***

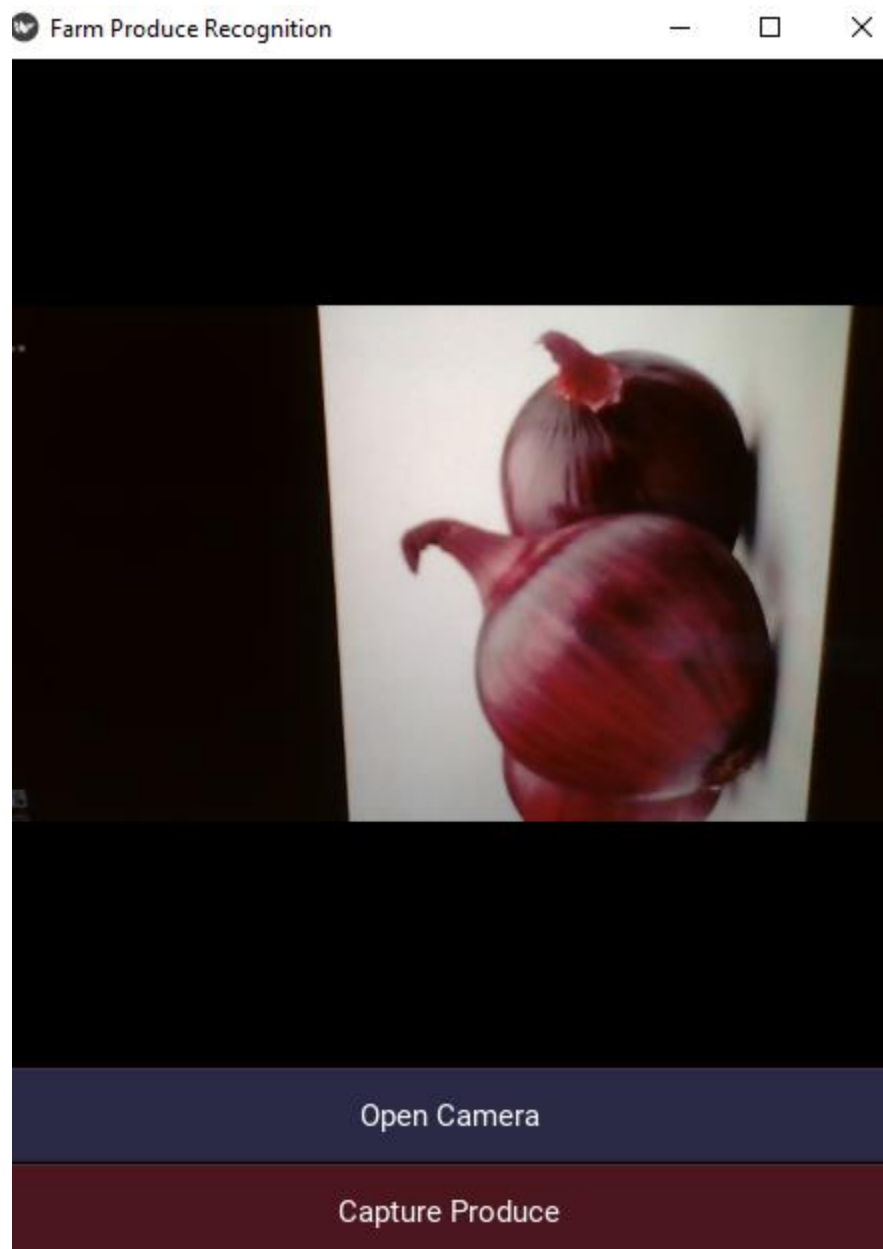
Kivy Screenshots

Open Camera



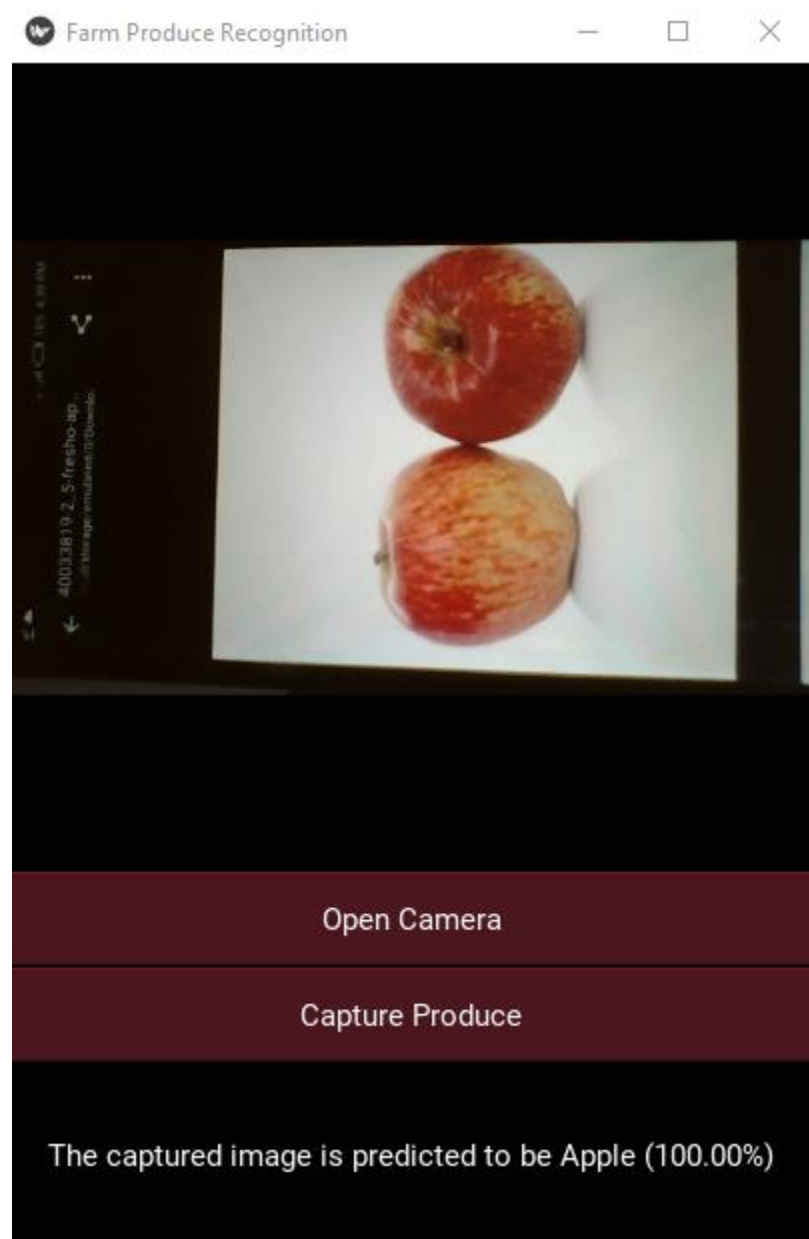
Screenshot 1- Open camera

Capture Image

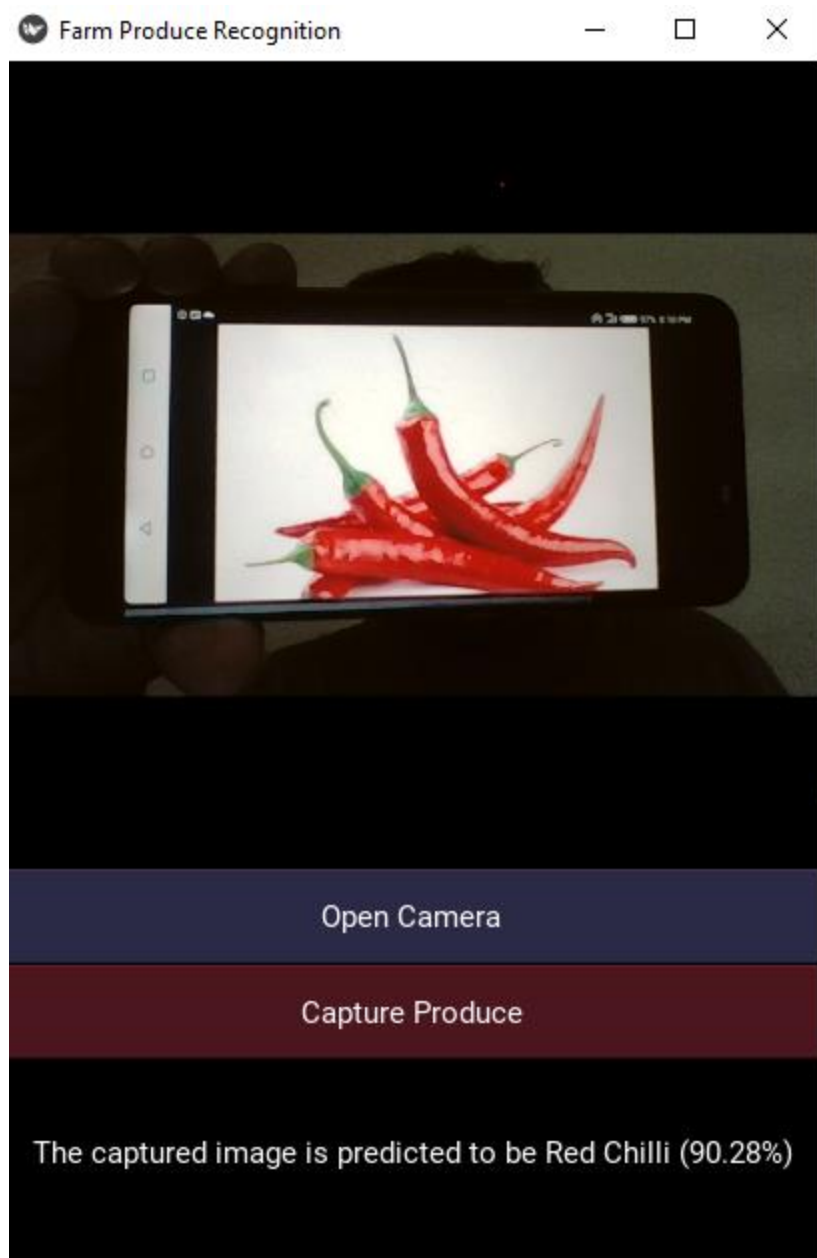


Screenshot 2- Capture Produce

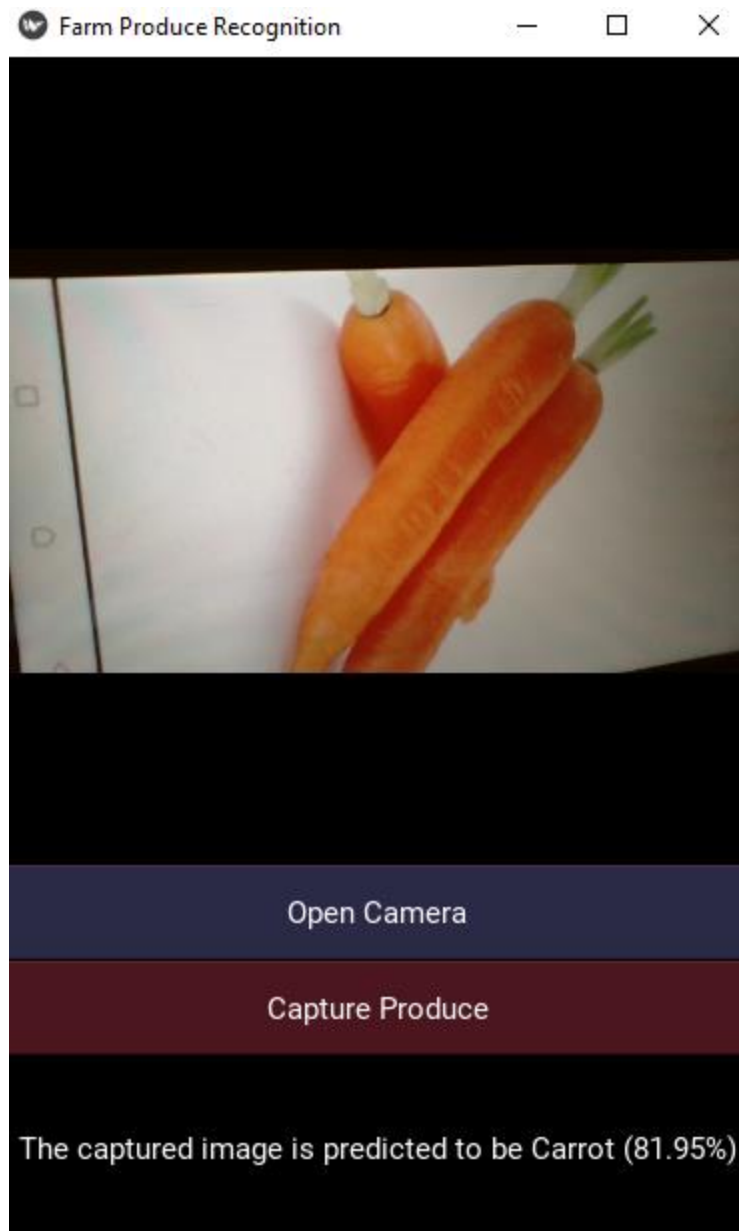
Results of different produce items



Screenshot 3- Predicted Item Apple



Screenshot 3- Predicted Item Red Chilli



Screenshot 3- Predicted Item Carrot

5) Maintenance User Guide for Engineers and Analysts:

There is always scope for addition of new fruits and vegetable, in future if there is addition of fruits or Vegetable, below steps have to performed to have code up and running.

- As mentioned in data preparation step, Create_DataSet.ipynb file will help in adding more fruits and vegetables. Have videos recorded for fruits and vegetables. Create folders in all three train, valid and test data set using create_folder function. Convert video to images using video to image convert function as shown in the data preparation step. Internet image is optional, and finally limit the train to 800, valid and test to 320 images.
- For model building we have three code files as mentioned in the model building section. Consider these three files, modify the Train, Valid and Test paths as shown in the model preparation step. Also, there is a scope to enhance model by customizing more. After executing all cells in all three notebooks, you will get the three model. Choose the best one for operationalize.
- As shown in the kivy repository, repository comes with the label and model. Upload your latest model to the directory, put the same path in main.py and update label.txt with the new fruits or vegetable that you have added.