



# Farm Decision Support System (DSS)

---

## Data X I Project Report

**AI and Machine Learning for Agriculture**

Plaksha TLF Founding Batch of 2019-20



# Table of Contents

Team Members and Roles.....	2
Project Introduction.....	3
Project Infrastructure .....	3
Datasets .....	4
Kisan Call Centre (KCC).....	4
Soil Health Card Data .....	4
Mandi Price Data.....	4
Disease Details Data .....	5
Plant Village Data.....	5
Maize Field Images .....	5
Data Analysis and EDA .....	6
Kisan Call Center Data .....	6
Focus Crop 1: Cotton Crop .....	6
Focus Crop 2: Wheat Crop.....	7
Pricing Data .....	7
Soil Data .....	9
Maize Field Image Data .....	10
Project Component Details .....	11
Component 1A – Chat bot Using Rasa NLU.....	11
Component 1A - Chatbot Using Google Dialogflow .....	13
Component 1B – Know Your Price – Nearest Mandi Prices using KNN.....	15
Component 2A: Disease Detection – Plant village using ResNet50.....	17
Component 2B: Disease Detection – Maize Images using Faster RCNN.....	19
Acknowledgements.....	25
References .....	25

## Team Members and Roles

### The People behind the Product

#### **Manan**

ML Engineer and Data Engineer

#### **Jaya**

Data Analyst

#### **Bharat**

DevOps Engineer

#### **Anmol**

Hustler and Tool Optimizer

#### **Pranati**

Software Development and Data Science

#### **Saif**

Product Development

### Team Members

...



## Project Introduction

Agriculture sector is the largest employer in India, with more than half the total workforce of the country. However, the sector has been in a state of perennial distress (seeing negative growth in recent years) which has impacted farmers the most, the average annual household income of an Indian farmer being less than \$1500.

Some of the Major Reasons for this:

- Information asymmetry between the farmers and the data sources.
- Unaware of best practices
- Lack of market awareness
- Increasing climate change uncertainty

We propose a decision support system which consists of a suite of software to support farmers whenever they require more information about farming related practices. It consists of 2 solutions currently:

- Farm Assist:
  - Text based Chatbot - Advice on Plant Protection (For Cotton & Wheat)
  - Know Your Price – Nearest Mandi Commodity Price

*This Solution is Deployed on this Web Application.*

- Image Based Crop Disease Detection:
  - Plant Village Based Multi Crop Multi Disease Detection
  - Maize Field Images Disease Detection

*Currently this is running on our local system. Will be deployed soon.*

## Project Infrastructure

- GitHub Account: <https://github.com/pranatibalodia/DataX-AgriAI>
- Web App Link: <https://farm-assist.herokuapp.com/>

## Interesting Figures

...

- Agricultural sector employs nearly half of the total workforce, but it contributes to 17.5% of the GDP
- Industry growth has been volatile over the past decade, ranging from 5.8% in 2005-06 to 0.4% in 2009-10 and -0.2% in 2014-15
- India has the 10th largest arable land resources in the world
- The average income of a farmer household at current prices is Rs 96,703 (US\$ 1,505.27)
- Total agricultural exports from India grew at a CAGR of 16.45 per cent over FY10-18 to reach US\$ 38.21 billion in FY18

## Datasets

### Kisan Call Centre (KCC)

- Website Name: Open Government Data Platform
- Website Link: <https://data.gov.in/>
- About Data: KCC data is a monthly state and region-specific documentation of the queries asked by the farmer and solution provided by government employees via phone calls.
- Reason for Choosing the Dataset: Most comprehensive source of publicly available data when it comes to common problems arising faced by farmers in India dating back to 2008, available for all Indian states.
- Tool Used to Obtain Data: Selenium (Python) & Chrome Driver
- Data Type: Text Data
- Data Columns: Sector, Category, Crop, Query Type, Query Text, KCC Answer, State Name, District Name, Block Name and Created On
- Data Period: 2 Years – 2015 and 2016
- Data Geography: 8 largest cotton and wheat producing states in India
- Data Size: 5.2 Million records

### Soil Health Card Data

- Website Name: Department of Agriculture, Cooperation & Farmers Welfare, Govt of India
- Website Link: <https://soilhealth.dac.gov.in/>
- About Data: The data is about soil composition and field wise variation for different parameters of soil and sample wise fertilizer recommendations.
- Tool Used to Obtain Data: Selenium (Python) & Chrome Driver
- Data Type: Numerical and Text
- Data Columns: Sample No, Sample Collection Date, Land Area, Irrigation, Longitude, Latitude, pH, EC, Organic Carbon (OC), Available Nitrogen (N), Available Phosphorus (P), Available Potassium (K), Available Sulphur (S), Available Zinc (Zn), Available Boron (B), Available Iron (Fe), Available Manganese (Mn), Available Copper (Cu), Fertilizer Recommendation
- Data Period: Latest Available
- Data Geography: Region - Ajnala, Amritsar District, Punjab
- Data Size: 1400 Records

### Mandi Price Data

- Website Name: Open Government data platform
- Website Link: <https://data.gov.in/>
- About Data: Mandi is a wholesale marketplace for food and agri-commodities. The dataset has state and month-wise mandi pricing of various crops for last 4 years.

## Datasets

...





- Tool Used to Obtain Data: Selenium (Python) & Chrome Driver
- Data Type: Numerical Data
- Data Columns: State, District, Market, Commodity, Variety, Arrival Date, Minimum Price, Maximum Price and Modal Price
- Data Period: 2015 onwards (till 2019 August)
- Data Geography: All over India
- Data Size: 5.8 Million Records

## Disease Details Data

- Website Name: Plantix
- Website Link: <https://plantix.net/en/>
- About Data: Data is about the common diseases in the crop, their symptoms and pesticide suggested for treatment.
- Tool Used to Obtain Data: Selenium (Python) & Chrome Driver
- Data Type: Text
- Data Columns: Name, Symptoms, Trigger, Biological Control, Chemical Control and Preventive Measure
- Data Size: 556 Records

## Plant Village Data

- Data Source: Plant Village (A moderated Q & A forum for farmers)
- Dataset Link: <https://github.com/spMohanty/PlantVillage-Dataset>
- About Data: Dataset of diseased plant leaf images and corresponding labels. This was a part of Kaggle and crowdAI plant disease classification challenge too.
- Data Type: Image
- Data Size: 55,261 Images
- Data Details: Containing 38 classes of 14 crop species (like corn, apple, pepper, tomato etc.) and 26 diseases (common rust, leaf mold, Blight, etc.) or absence thereof.
- More Details: <https://arxiv.org/abs/1511.08060>

## Maize Field Images

- Data Source: Open Source Foundation
- Dataset Link: <https://osf.io/p67rz/>
- About Data: This repository contains images of maize (corn) leaves that have been annotated to mark lesions caused by Northern Leaf Blight (NLB), a common and devastating disease of maize.
- Data Type: Image
- Data Size: 18,222 Images
- Data Details: 105,735 Bounding Boxes
- More Details: <https://osf.io/p67rz/>

## Datasets



Plant Village



# Data Analysis and EDA

## Kisan Call Center Data

Total Number of Query Records: 5219780

Number of Queries State Wise: (Top 8 Cotton and Wheat Producing States in India)

- Maharashtra:1376489
- Rajasthan: 1042447
- Madhya Pradesh: 809774
- Haryana: 545790
- Punjab: 497354
- Karnataka: 349538
- Gujarat: 331982
- Andhra Pradesh:266406

### Focus Crop 1: Cotton Crop

Number of Records for Cotton: 266535

Top 5 Categories of Queries within Cotton:

- Plant Protection: 45%
- Weather: 10%
- Fertilizer Use and Availability: 8.6%
- Market Information: 8.5%
- Varieties: 8%

Number of Queries in Plant Protection Category for Cotton: 120962

Top 5 Most Asked Questions within Plant Protection Category for Cotton:

- Ask about sucking pests problem in crop?: 2526
- Attack of White Fly?: 2348
- Attack of Thrips and Jassids? : 2261
- Information regarding how to control fungal disease in cotton? : 2151
- how to control leaf curl in cotton crops?: 1995

Number of Unique Queries in Plant Protection for Cotton: 67286  
(unique considering geographical columns)

Number of Unique Queries in Plant Protection for Cotton: 10603  
(after removing geographical details)

Number of Unique Queries in Plant Protection for Cotton: 3014  
(after cleaning- stop words, defined library, punctuations, stemming)

## Focus Crop 2: Wheat Crop

Number of Records for Wheat: 355263

Top 5 Categories of Queries within Wheat:

- Weather: 44%
- Plant Protection: 19%
- Varieties: 10%
- Fertilizer Use and Availability: 4%
- Weed Management: 4%

Number of Queries in Plant Protection Category for Wheat: 68249

Top 5 Most Asked Questions within Plant Protection Category for Wheat:

- Information regarding for the control of aphids in wheat crop?: 3554
- Information regarding control of aphid in wheat?: 2060
- Information regarding control of zinc deficiency in wheat ? : 1986
- Information regarding for the yellow rust disease in wheat crop?: 1783
- Information regarding how to control aphids/sundi in wheat crop?: 1289

Number of Unique Queries in Plant Protection for Wheat: 39355  
(unique considering geographical columns)

Number of Unique Queries in Plant Protection in Wheat: 8061  
(after removing geographical details)

Number of Unique Queries in Plant Protection for Cotton: 2851  
(after cleaning- stop words, defined library, punctuations, stemming)

## Pricing Data

Number of Records for the Year 2019: 1.5 Million

Number of Records for Years prior to 2019: 4.3 Million

*Note: Only data for 2019 has been used*

Number of Districts: 488

Number of Markets: 2329

Analysis for Wheat:

- Average Minimum Price: Rs. 1825
- Average Maximum Price: Rs. 1984
- Average Modal Price: Rs. 1915

*Note: This data is based on 1241 markets in 296 districts till 1<sup>st</sup> August 2019*



[Link to Google Data Studio Report on Agriculture Pricing in India](#)

Top 3 Commodities as per Modal Price in 2019:

	COMMODITY	MAX_PRICE	MIN_PRICE	MODAL_PRICE ▾
1.	Cardamoms	196,043.9	163,063.18	178,631.81
2.	Cinamon(Dalchini)	101,666.67	83,333.33	92,500
3.	Jasmine	93,176.47	85,823.53	89,647.06

Pricing Trends for Wheat (2019) – All Varieties of Wheat



Average Prices of Top 8 Varieties of Wheat for 2019:

	COMMODITY	VARIETY	MAX_PRICE...	MIN_PRICE	MODAL_PRICE
1.	Wheat	Super Fine	3,037.96	2,758.47	2,899.17
2.	Wheat	JawarI	2,970.04	2,194.78	2,520.3
3.	Wheat	Red	2,692.03	2,126.01	2,467.41
4.	Wheat	Sharbati	2,562.3	2,138.81	2,350.22
5.	Wheat	MP Sharbati	2,450.87	1,874.1	2,112.06
6.	Wheat	Bansi	2,365.45	2,068.04	2,234.04
7.	Wheat	Sechor No. 1	2,356.53	1,878.68	2,101.08
8.	Wheat	Kirithi	2,332	2,080	2,190

Top 10 Markets for Wheat based on Modal Price (2019)

	MARKET	DISTRICT	STATE	MAX_PRICE	MIN_PRICE	MODAL_PRICE...
1.	Ner Parasopant	Yavatmal	Maharashtra	5,560	5,400	5,550
2.	Pune	Pune	Maharashtra	4,210.94	3,792.19	4,023.44
3.	Mumbai	Mumbai	Maharashtra	3,832.54	2,300.59	3,062.72
4.	Bijapur	Bijapur	Karnataka	3,257.37	2,840.7	3,052.37
5.	Dondlchare	Balod	Chhattisgarh	3,000	3,000	3,000
6.	Shimoga	Shimoga	Karnataka	3,109.13	2,788.1	2,949.8
7.	Davangere	Davangere	Karnataka	2,968.10	2,825	2,896.59
8.	Jamkhandi	Bagalkot	Karnataka	2,971.64	2,744.02	2,864.26
9.	Pallahara	Angul	Odisha	3,000	2,750	2,850
10.	Mysore (Bandipaly...	Mysore	Karnataka	3,108.7	2,551.09	2,827.17

## Soil Data

(We haven't used soil data in our solution at present)

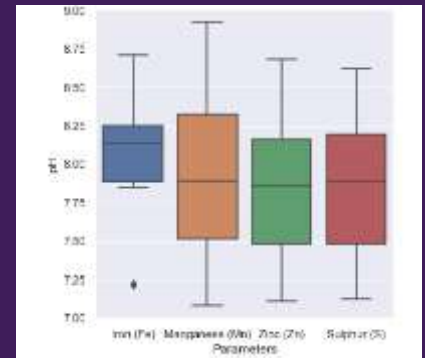
Details: Soil data consist of two csv files with soil composition sample wise and fertilizer recommended on sample number.

Insights observed are as followed:

- Most recommended fertilizer through soil = Manganese sulphate (10 -25 kg/ha)
- Most recommended fertilizer through spray = 1 % Manganese sulphate + 0.25% lime or 0.5% Manganese sulphate
- Average value of elements in soil composition sample
  - Avg pH = 7.938042
  - Avg EC (ds/m) = 0.299548
  - Avg Organic Carbon (% L) = 0.412855
  - Avg Phosphorus (kg/ha) = 15.539456
  - Avg Potassium (kh/ha) = 176.001230
  - Avg Sulphur (ppm) = 29.465081
  - Avg Zinc (ppm) = 1.149329
  - Avg Iron (ppm) = 11.752198
  - Avg Manganese (ppm) = 1.975138
  - Avg Copper(ppm) = 1.883986

## Plant Village Data

- No. of images: 55,261
- Dimension of Images: Width: 224 Height: 224 Depth: 3
- 38 classes of 14 crop species and 26 diseases
- Split between different plants:
  - Apple (3172)
  - Blueberry (1502)
  - Cherry (1906)
  - Corn (3852)
  - Grape (4063)
  - Orange (5507)
  - Peach (2657)
  - Bell Pepper (2475)
  - Potato (2152)
  - Raspberry (371)
  - Soybean (5090)
  - Squash (1835)
  - Strawberry (1565)
  - Tomato (18,162)
  - Fulvia-fulva-(952)
- Sample Images (Right)



Value of pH vs Other Soil Minerals



Plant Village Samples

## Maize Field Image Data

(Only Handheld Images part of the Data has been used)

- No. of images total = 1787,
- Train = 1426 (6257 Ground Truth Bounding boxes)
- Test = 361 (1442 Ground Truth Bounding boxes)
- No. of classes/labels: Label - 1 (Northern Light Blight), Label – 0 (Healthy)
- Images per class/label distribution (Bounding Boxes)
  - Train: Label 1: 5502, Label 0: 640
  - Test: Label 1: 1403, Label 0: 154
- Dimensions of the images: Varies
- Sample Images (Right Bare)

### Important:

The main difference between the two image-based datasets is that in Plant Village is one label on the image and in Maize Field Image data there are bounding boxes to mark lesions on each image. The Maize data is more realistic as it real field data.



Maize Field Image Samples

## Project Component Details

As mentioned in our introduction, our solution decision contains two broad software solutions at present. Here we detail each of them including some trial attempts we worked on while development but were later on rejected for better tools.

### Component 1A – Chat bot Using Rasa NLU

*Status: Developed, Not used due to other better options*

Tools and Algorithms Used: Rasa NLU, Chatito, Pusher, Word2Vec, Doc2Vec, K-Means, LDA

#### What is Rasa NLU?

Rasa NLU: Rasa NLU is an open-source natural language processing tool for intent classification, response retrieval and entity extraction in chatbots. Rasa provides a set of tools to build a complete Chatbot on your local machine. Being an open source framework, it is easy to customize. It's a library which does the classification of intent and extract the entity from the user input and helps the bot to understand what the user is trying to communicate.

#### Why Rasa NLU?

- The team could focus their energy on the Chatbot and not waste time on logistical tasks like deploying, creating servers, etc.
- The default set up of Rasa works well out of the box for intent extraction and dialogue management.
- Rasa stack is open source, which means we know exactly what is happening under the hood and that allows us to customize pieces to our taste.

#### Implementation Steps:

- Generating entity examples using Chatito  
We used Chatito to create a knowledge base for the Chatbot. This was used for training.
- Data preparation and format  
For training data in rasa, we preferred markdown format over json format. In markdown format we specified
- Intent  
We specified intent and list of questions attached to that.
- Entity  
Entity was specified inside the knowledge-based questions.



### Selecting Rasa Pipeline

Pipeline is a set of algorithms to be used to train your model. Two popular pipelines are supervised\_embeddings and pretrained\_embeddings\_spacy. Difference in between the pipelines are that pretrained\_embeddings\_spacy uses pipeline that uses pre trained word vectors. While supervised\_embeddings don't use any pre trained word vectors. This embedding can be fixed for our dataset prominently.

#### *Training model in Different Language:*

Supervised\_embedding pipeline is chosen to train model in Hindi language because supervised\_embedding uses your own training data to create custom word embeddings. Though we tried to train model using different language, but it was taking a lot of time for understanding the structure and architecture of Rasa NLU. Hence, we decided to switch to different platform.

```
pipeline:
- name: "tokenizer_whitespace"
- name: "ner_crf"
- name: "ner_synonyms"
- name: "intent_featurizer_count_vectors"
- name: "intent_classifier_tensorflow_embedding"
```

#### **Pipeline for Rasa**

### Limitations of Rasa

1. After uploading the knowledge base on Rasa NLU, entity recognition was very poor. For improving this we added more organized lookup tables and synced it with rasa, but the result was not enough convincing and time consuming.
2. No out of the box integration with other messaging platform and devices.

### Additional Trials:

Clustering and Cosine Similarity: The questions in the Kisan Call Center were converted into embedding's using Google Word2vec and doc2vec feature. These were then used to for the following:

1. Figuring out similarity percentage between questions using Cosine Similarity
2. Using Latent Dirichlet allocation probabilistic model to find out similar questions within the dataset
3. Clustering the questions using the embedding's and K-Means clustering algorithm

## Component 1A - Chatbot Using Google Dialogflow

***Status: Developed, Deployed In Production***

Tools and API's Used: Dialogflow, Google Speech to Text API, Flask, JQuery, Bootstrap Studio, Heroku

### What is Google Dialogflow?

Dialogflow is an end-to-end, build-once deploy-everywhere development suite for creating conversational interfaces for websites, mobile applications, popular messaging platforms, and IoT devices. You can use it to build interfaces (such as chatbots and conversational IVR) that enable natural and rich interactions.

### Why Dialogflow?

- Quick and easy to start building
- Built on Google infrastructure
- Easy to scale
- Strong natural language understanding (NLU) capabilities

### Implementation Steps:

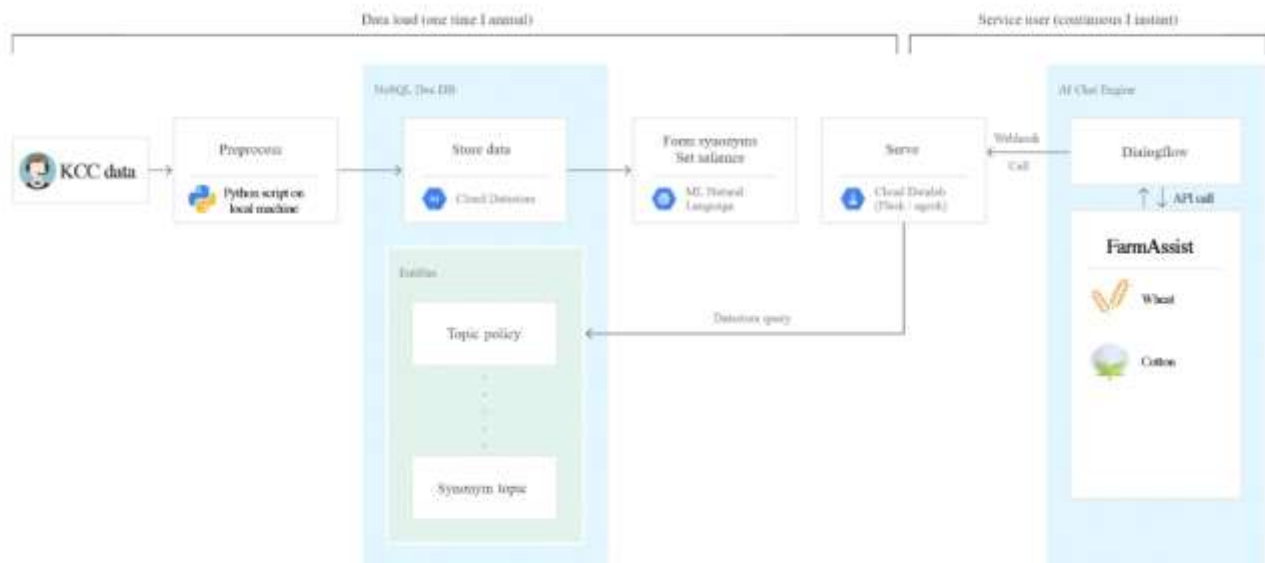
- We used the knowledge base feature of the tool which is in beta mode.
- 'Knowledge Connectors' lets you add your content in csv or plain text format as a Knowledge Base to your Agent.
- For a given user utterance, the Bot would then try to match it to the predefined intent, or a system generated intent based on its match to the text in the Knowledge Base.

This is a good approach to make the building of Chatbots more practical. There are challenges even with the use of 'Knowledge Connects' like the accuracy of the extracted answer from the Knowledge Base. Our initial evaluation did not give very good results. The extracted answers look more like a match based on keywords with some additional sophistication, but it does not appear to consider the context in which the question is asked.

### Why did we choose Dialogflow over Rasa NLU?

We finally went with Dialogflow as we did not have to manually create intent, training phrase for the question and add responses for all the Question and Answer. We just had to upload a CSV in the knowledgebase after some data manipulation.

## System Architecture



## Snippets of Working Bot:



Prototype Link: [https://farm-assist.herokuapp.com/display\\_chatbot](https://farm-assist.herokuapp.com/display_chatbot)

## Limitations

- If the auto-generated answer is incorrect, there is no way to identify and address that issue. There is no option to evaluate or seek feedback from the users which could be feedback to the system to improve the quality of the answers next time.
- Other limitation that we encountered was that knowledge Base needs to be in a structured format and does not let us create a hierarchical logic. Also, the knowledge base cannot be on websites from sites that require authentication of any sort or those which are behind a login page.

## Component 1B – Know Your Price – Nearest Mandi Prices using KNN

*Status: Developed, Deployed In Production*

Here based on the data we have, we wanted to develop a feature wherein when a user opens his application, his geolocation data is fetched and based on that the 5 nearest wholesale Mandi from the neighborhood are found out. Once this is done, the user can choose which Mandi he wants to see the prices from, the commodity he wants to check out the prices for and based on this the system will give out the last available prices of the given from the data.

Tools and Algorithms Used: Google Geocoding Services (Maps Platform) KNN, Bootstrap Studio, Flask, Heroku

### What is KNN?

In pattern recognition, the k-nearest neighbor's algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. K-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification.

### Why KNN?

Ideally we could have used a simple distance calculation function based on the latitude and longitude but since this is a large dataset we thought using a pre-trained KNN algorithm will be good to find out in which cluster the new Lat Long coordinates that have come in fall.

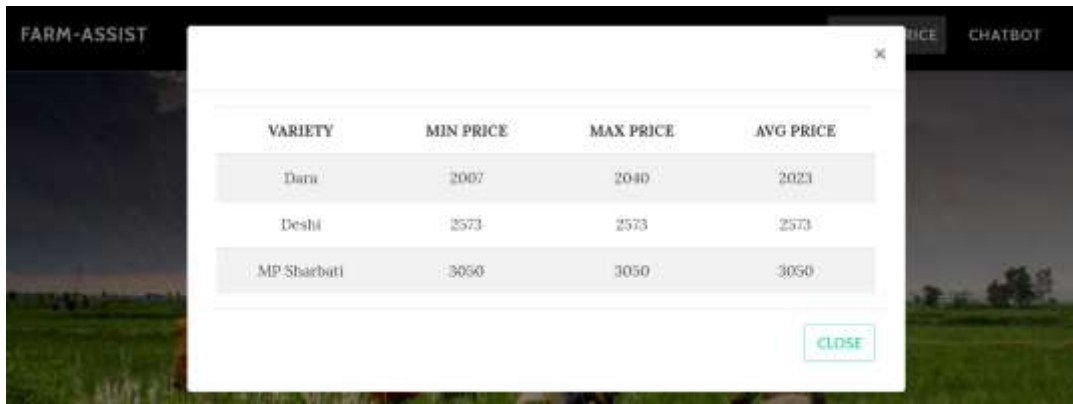
### Implementation Steps:

- Data from all the crop varieties and commodities, for all markets was combined for all of 2019
- We did preliminary analysis and EDA on Python as shared in the section above. Python and Google Data Studio were used for this
- Data was stored on Big Query in the Google Cloud Platform
- Lat Long was derived using the Google Geocoding API for all the market
- KNN was trained to form clusters using the Latitude and Longitude Data
- Lat Longitude is captured from the device and based on that the closest cluster is figure out.
- 5 closest markets names are displayed and you can choose crop. Latest Price is shown.

Prototype Link: <https://farm-assist.herokuapp.com/>



## Snippets of Working Solution:

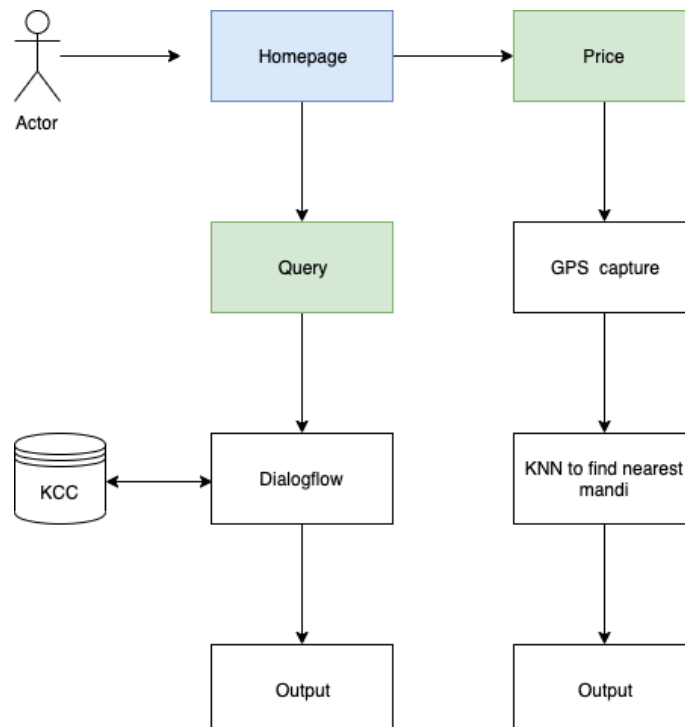


VARIETY	MIN PRICE	MAX PRICE	AVG PRICE
Dara	2007	2040	2023
Deshi	2573	2573	2573
MP Sharbati	3050	3050	3050

## Future Scope:

- Currently this is linked to a static data source, ideally it should be dynamically loading from the Government website on pricing so that it can fetch the most recent price available.
- At present we only show the last available price, what we can also include is a Pricing Trend curve which displays the trends over the duration chosen by the user and also predict pricing for some future date based on historical data from the last 5 years that we have.

## Complete Deployed Component 1A and 1B - User Architecture



## Component 2A - Disease Detection – Plant village using ResNet50

**Status: Developed, Not Deployed**

Here we took the widely available Plant Village dataset that is described in the above section to test out our first image based model using CNNs. The idea was to just do hand's with an easily available dataset so that we can use this knowledge to train a custom dataset.

Tools and Algorithm Used: Google Collab, ResNet50 Model, Keras

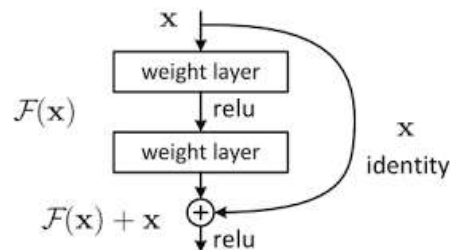
### What is ResNet50?

ResNet-50 is a convolutional neural network that is trained on more than a million images from the ImageNet database. The network is 50 layers deep and can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals.

### Why Res Net?

The main goal of the residual network is to build a deeper neural network. We can have two intuitions based on this:

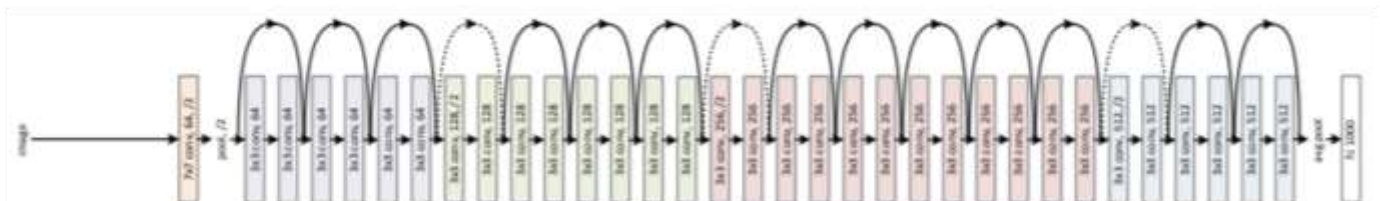
- As we keep going deeper into implementing large amount of layers, one should make sure not to degrade the accuracy and error rate. This can be handled by identity mapping.
- Keep learning the residuals to match the predicted with the actual



### Implementation Steps

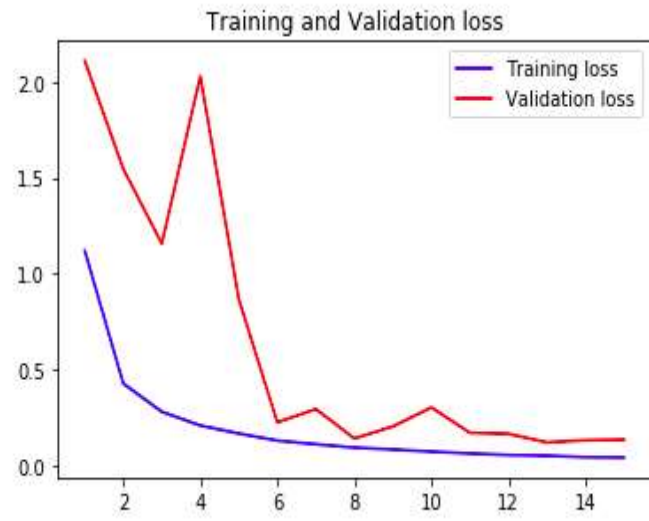
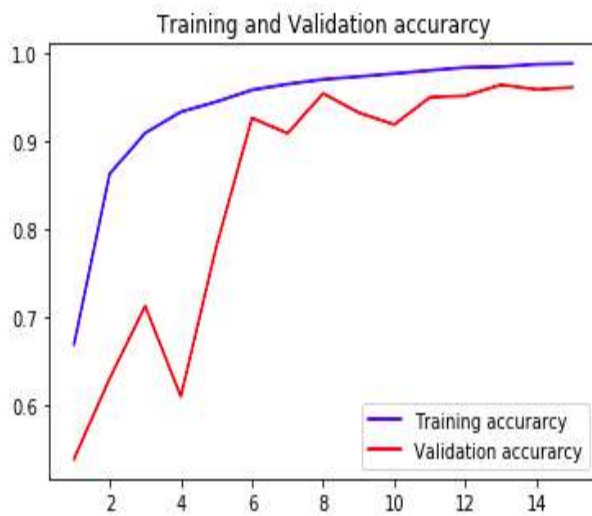
- The model was trained for 15 Epochs and with a Validation split of 20% of the data

### ResNet50 Architecture



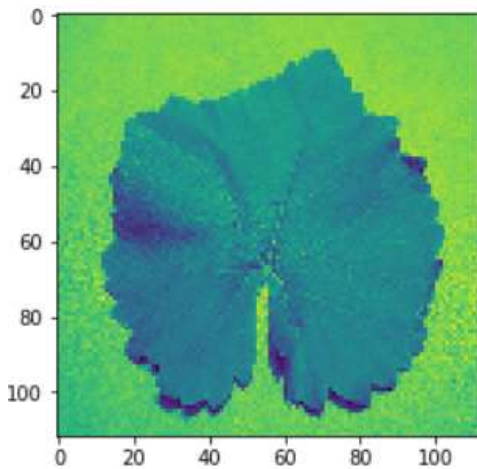


## Performance Evaluation



Training Accuracy: 97% | Validation Accuracy: 96%

## Snippets of Working Model:



```
y_test[target_index]: True  pred : [[ True]]
```

## Limitations

- Since this data is not in a real life setting and has been specially created for machine learning purposes, it has no real world usage. It's synthesized quite well as most of the leaf images are taken at a specific angle.

## Component 2B - Disease Detection – Maize Images using Faster RCNN

*Status: Developed, Not Deployed*

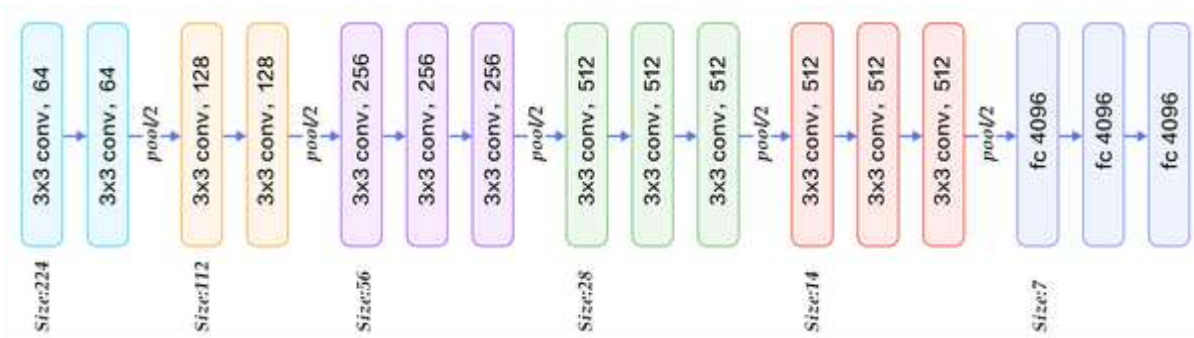
Tools and Algorithms Used: Google Cloud Platform, Faster RCNN

What is Faster RCNN?

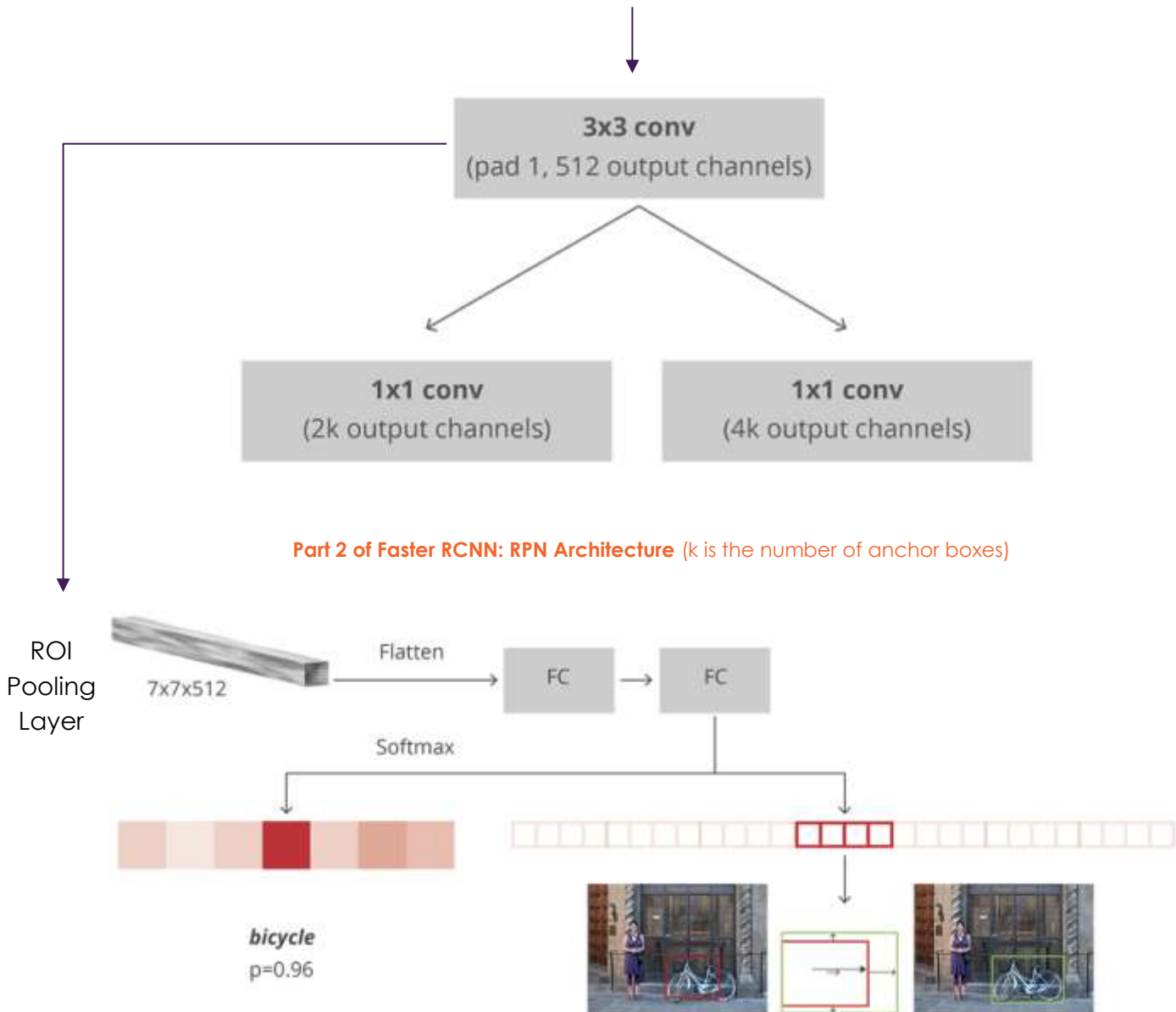
- **R-CNN** is the first step for Faster R-CNN. It uses search selective to find out the regions of interests and passes them to a ConvNet. The R-CNN paper uses 2,000 proposed areas (rectangular boxes) from search selective. Then, these 2,000 areas are passed to a pre-trained CNN model. Finally, the outputs (feature maps) are passed to a SVM for classification. The regression between predicted bounding boxes (bboxes) and ground-truth bboxes are computed.
- **Fast R-CNN** moves one step forward. Instead of applying 2,000 times CNN to proposed areas, it only passes the original image to a pre-trained CNN model once. Search selective algorithm is computed based on the output feature map of the previous step. Then, ROI pooling layer is used to ensure the standard and pre-defined output size. These valid outputs are passed to a fully connected layer as inputs. Finally, two output vectors are used to predict the observed object with a softmax classifier and adapt bounding box localizations with a linear regressor.
- **Faster R-CNN** (FRCNN for short) makes further progress than Fast R-CNN. Search selective process is replaced by Region Proposal Network (RPN). As the name revealed, RPN is a network to propose regions. Next, RPN is connected to a Conv layer with 3x3 filters, 1 padding, 512 output channels. The output is connected to two 1x1 convolutional layer for classification and box-regression

Model Architecture Detailed

- **VGG Layer:** Pre- trained VGG 16 Model is applied to the image develop feature map
- **RPN Layer:** Each point in feature map is considered as an anchor. We need to define specific ratios and sizes for each anchor
  - Then RPN is connected to Conv Layer with 3\*3 filter, 1 padding and 512 output channels.
  - Intermediate Output: Connected to 1\*1 Fully Connected Layer - box classification and box regression
- **RoI Layer:** ROI pooling is used for these proposed regions (ROIs). The output is 7x7x512.
  - Then, we flatten this layer with some fully connected layers.
- **Classifier Layer:** The final step is a softmax function for classification and linear regression to fix the boxes' location.



Part 1 of Faster RCNN: VGG16 Model on Images to Generate Feature Maps



Part 3 of Faster RCNN: ROI Pooling Layer, Fully Connected Layers and Classifier Layer

## Implementation Steps

### Setting up the Data

- We took only the handheld camera images from the maize data. No. of images total = 1787 (7699 Ground Truth Bounding boxes)
- This data was split into:
  - Train = 1426 (6257 Ground Truth Bounding boxes)
  - Test = 361 (1442 Ground Truth Bounding boxes)
- No. of classes/labels: Label - 1 (Northern Light Blight Disease), Label - 0 (Healthy). Here is the split between bounding boxes in the Train and Test data:
  - Train: Label 1: 5502, Label 0: 640
  - Test: Label 1: 1403, Label 0: 154
- Annotations CSV file with x1,x2, y1,y2 coordinates of the bounding boxes, along with label
- Each image is resized to 300\*450 for faster training
- First we take all the images in the train and plot the bounding boxes

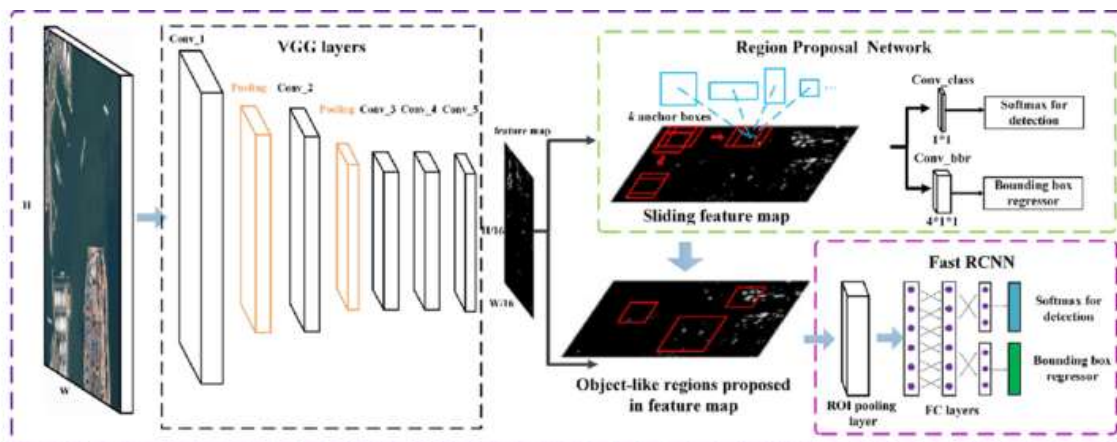
### Calculation Steps:

- Rebuild the structure of VGG-16 and load pre-trained model
- Calculate rpn for each image
- Calculate region of interest from RPN
- RoI Pooling layer and Classifier layer

### Parameters:

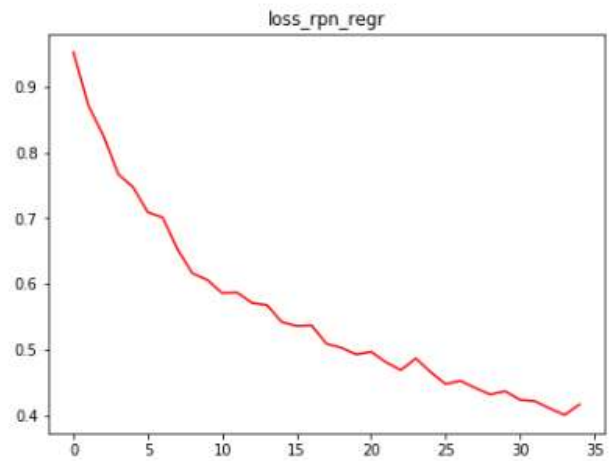
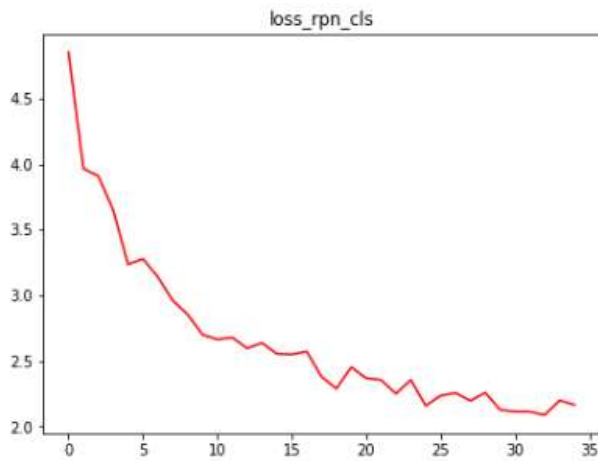
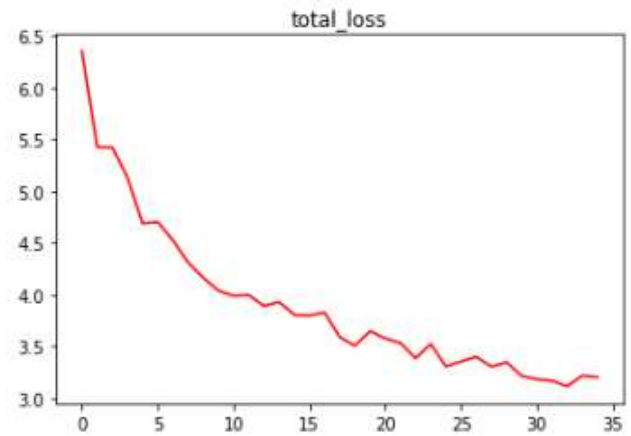
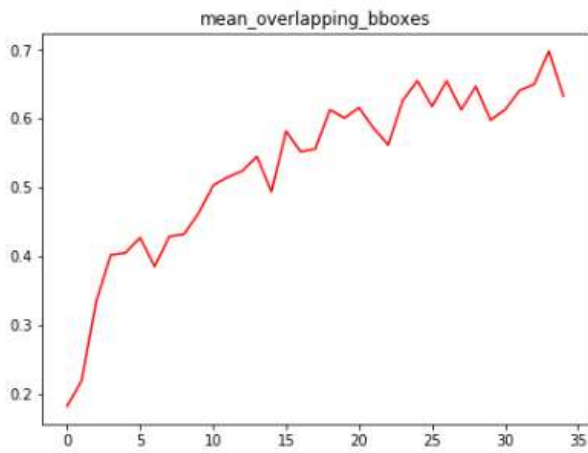
- Resized (im\_size) value is 300.
- The number of anchors is 9.
- Max number of non-max-suppression is 300.
- Number of RoI to process in the model is 4 (I haven't tried larger size which might speed up the calculation but more memory needed)
- Adam is used for optimization and the learning rate is 1e-5

### Full Model Architecture



## Performance Evaluation

Number of Epochs Trained: 35







Mean Overlapping Boxes	Classification Accuracy	RPN Class Loss	RPN Reg Loss	Classifier Class Loss	Classifier Reg Loss	Total Loss	Time Taken (m)
0.182	0.878	4.852	0.952	0.42	0.129	6.354	78.37
0.219	0.85	3.966	0.871	0.451	0.139	5.428	80.093
0.335	0.801	3.909	0.826	0.511	0.177	5.423	77.388
0.402	0.778	3.652	0.767	0.529	0.183	5.13	76.672
0.405	0.774	3.237	0.747	0.516	0.188	4.688	75.646
0.427	0.776	3.278	0.709	0.516	0.199	4.702	75.147
0.385	0.791	3.141	0.701	0.497	0.185	4.524	77.506
0.429	0.778	2.961	0.652	0.499	0.197	4.309	76.33
0.432	0.78	2.855	0.616	0.491	0.2	4.162	74.847
0.462	0.766	2.701	0.606	0.513	0.217	4.037	76.088
0.503	0.746	2.665	0.586	0.523	0.213	3.987	75.472
0.515	0.762	2.679	0.587	0.516	0.213	3.995	74.928
0.524	0.764	2.597	0.571	0.505	0.217	3.889	74.696
0.545	0.753	2.637	0.568	0.514	0.207	3.927	73.591
0.494	0.758	2.555	0.542	0.506	0.201	3.804	74.781
0.582	0.764	2.55	0.536	0.506	0.205	3.797	72.384
0.552	0.748	2.573	0.537	0.513	0.204	3.827	73.276
0.556	0.767	2.38	0.509	0.487	0.211	3.587	72.743
0.613	0.764	2.29	0.503	0.497	0.215	3.505	71.964
0.601	0.75	2.453	0.493	0.502	0.199	3.647	76.276
0.616	0.754	2.369	0.497	0.502	0.207	3.575	72.922
0.586	0.767	2.356	0.481	0.49	0.205	3.532	75.51
0.562	0.771	2.251	0.469	0.473	0.191	3.384	74.804
0.627	0.77	2.356	0.487	0.478	0.202	3.522	74.245
0.655	0.78	2.159	0.466	0.478	0.203	3.305	76.023
0.618	0.773	2.236	0.448	0.47	0.198	3.352	75.235
0.655	0.766	2.257	0.453	0.48	0.207	3.397	75.343
0.613	0.767	2.197	0.442	0.467	0.198	3.303	77.647
0.647	0.77	2.259	0.432	0.464	0.189	3.344	74.693
0.598	0.782	2.128	0.437	0.454	0.193	3.212	76.774
0.613	0.781	2.115	0.424	0.457	0.184	3.18	75.773
0.641	0.788	2.113	0.422	0.448	0.184	3.166	74.908
0.65	0.795	2.089	0.411	0.44	0.172	3.112	76.606
0.698	0.798	2.2	0.401	0.425	0.191	3.217	74.198

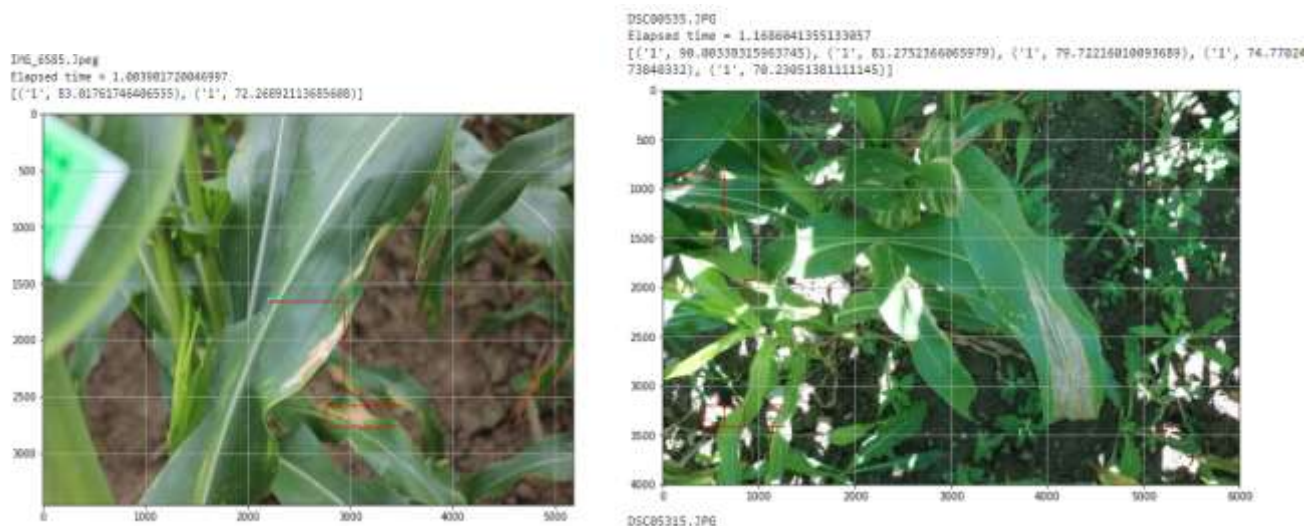


### Analysis of the Results:

- Mean Overlapping Boxes with the ground truth box increases substantially over the epochs. 69% - 35<sup>th</sup> Epoch.
- Classification Accuracy: 80%
- RPN Classification Loss has come down from 4.8 to 2.2
- RPN Regression Loss has come down from 0.9 to 0.4
- RoI Classifier Regression and Classification loss has been fluctuating but hasn't come down much. This needs to be studied.
- Total loss has reduced by half over the 5 epochs from 6.3 to 3.2

**Mean Average Precision on Test Data (361 Images, 1442 Bounding Boxes): 67.5%**

### Snippets of Working Solution:



### Future Scope:

- We can extend the usage of this model to the wider maize dataset which includes drone images as well as boom camera images
- Newer models like Masked RCNN and Yolo etc. can also be tried out
- The model can be deployed in a web app using which user can take pictures on the field

### Limitations

- The dataset is imbalanced with most of the images being diseased, very few healthy
- Currently, we only have data available for one disease on Maize. For a better model we would need more classes
- We would need data for other crops as well to train a more comprehensive model.

## Acknowledgements

We would like to express our sincere thanks to all the people who helped us in completing the project.

Technical Support from Co-Fellows:

1. Shikhar Seth – Faster RCNN
2. Anshumaan Dash – Object Detection
3. Praveen Sridhar – NLP and Raasa

Faculty Guidance on Ideas:

1. Prof. Dharmendra Saraswat – Purdue University
2. Prof. Alexander Fred Ojala – UC Berkeley SCET
3. Prof. Ravi Kothari – Former Chief Scientist at IBM Research

Plaksha TLF Program team for all their logistical support.

## References

This is not an exhaustive list of resources that we referred to for the project. As we referred to several research papers, technical documentations, articles and blogs as well as code implementations, we would be unable to list them all here. Only some of the most important ones have been mentioned here.

- Kisan Call Center: <https://data.gov.in/dataset-group-name/kisan-call-centre>
- Agricultural Marketing: <https://data.gov.in/dataset-group-name/agricultural-marketing>
- Farm Chat: Using Chatbots to Answer Farmer Queries in India  
<https://ubicomplab.cs.washington.edu/publications/farmchat/>  
<https://www.ictworks.org/wp-content/uploads/2019/01/farmer-chatbot.pdf>
- Crop Yield Prediction and Efficient Use of Fertilizer: <https://ieeexplore.ieee.org/document/8698087>
- Plant Disease Detection using Convolutional Neural Networks: <https://towardsdatascience.com/plant-ai-plant-disease-detection-using-convolutional-neural-network-9b58a96f2289>
- Deploying AI Models: <https://towardsdatascience.com/plant-ai-deploying-deep-learning-models-9dda5f6c1088>
- Crop Guidance Information: <https://www.indiaagronet.com/crop-cultivation/>
- Agri Bot: <https://github.com/dhishku/AgriBot>
- Automated Detection of NLB: <https://apsjournals.apsnet.org/doi/full/10.1094/PHYTO-11-16-0417-R#>
- Faster RCNN Implementation on Open Image Dataset v4: <https://towardsdatascience.com/faster-r-cnn-object-detection-implemented-by-keras-for-custom-data-from-googles-open-images-125f62b9141a>
- Dialogflow Documentation: <https://cloud.google.com/dialogflow/docs/basics>
- Rasa Documentation: <https://rasa.com/docs/>
- Google Cloud Documentation: <https://cloud.google.com/docs/>
- Keras/ Resnet50 Documentation: <https://keras.io/>