

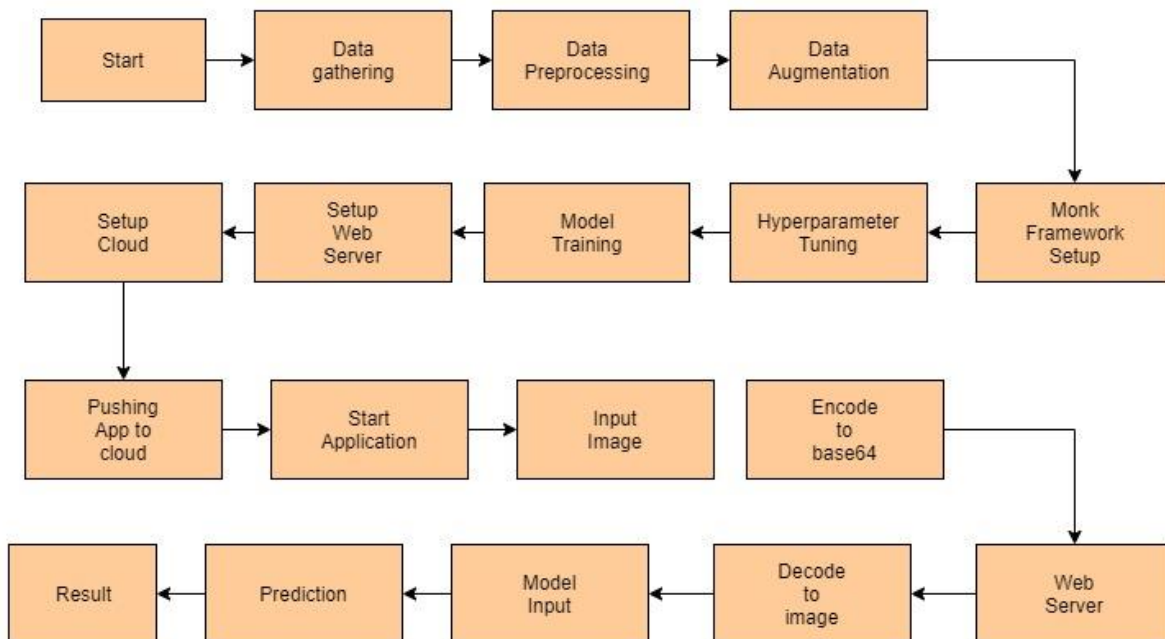
Plant Disease Classification

1. Description

In agriculture, leaf diseases cause a significant decrease in both quality and quantity of yields of crops. Automating the process of plant disease detection using Computer Vision could play a vital role in early detection and prevention of diseases.

Crop diseases are a significant threat to food security, but their rapid identification remains difficult in many parts of the world due to the lack of the necessary infrastructure. The combination of increasing edge and mobile devices penetration and recent advances in computer vision made possible by deep learning has paved the way for smartphone-assisted disease diagnosis. Using a public dataset of 54,306 images of diseased and healthy plant leaves collected under controlled conditions, we train a deep convolutional neural network to identify 14 crop species and 26 diseases.

2. Architecture



3. Data

We will be using the dataset gathered by PlantVillage (<https://plantvillage.psu.edu/>).

Dataset Link:-

<https://www.dropbox.com/s/hgt9uystjlinzlp/plantVillage.zip>

4. Data Description

We analyze 54,306 images of plant leaves, which have a spread of 38 class labels assigned to them. Each class label is a crop-disease pair, and we make an attempt to predict the crop-disease couple given just the image of the plant leaf. Figure 1 shows one example each from every crop-disease pair from the PlantVillage dataset. In all the approaches described in this paper, we resize the images to 256×256 pixels, and we perform both the model optimization and predictions on these downscaled images.

Dataset Size :-

- * Number of train images: 44014
- * Number of validation images: 11004
- * Number of classes: 39

Name of different categories: -

id	labels
0	Strawberry___healthy
1	background
2	Grape___Black_rot
3	Potato___Early_blight
4	Blueberry___healthy
5	Corn_(maize)___healthy
6	Tomato___Target_Spot
7	Peach___healthy
8	Potato___Late_blight
9	Tomato___Late_blight
10	Tomato___Tomato_mosaic_virus
11	Pepper_bell___healthy

12	Orange___Haunglongbing_(Citrus_greening)
13	Tomato___Leaf_Mold
14	Grape___Leaf_blight_(Isariopsis_Leaf_Spot)
15	Cherry_(including_sour)___Powdery_mildew
16	Apple___Cedar_apple_rust
17	Tomato___Bacterial_spot
18	Grape___healthy
19	Tomato___Early_blight
20	Corn_(maize)___Common_rust_
21	Grape___Esca_(Black_Measles)
22	Raspberry___healthy
23	Tomato___healthy
24	Cherry_(including_sour)___healthy
25	Tomato___Tomato_Yellow_Leaf_Curl_Virus
26	Apple___Apple_scab
27	Corn_(maize)___Northern_Leaf_Blight
28	Tomato___Spider_mites Two-spotted_spider_mite
29	Peach___Bacterial_spot
30	Pepper_bell___Bacterial_spot
31	Tomato___Septoria_leaf_spot
32	Squash___Powdery_mildew
	Corn_(maize)___Cercospora_leaf_spot
33	Gray_leaf_spot
34	Apple___Black_rot
35	Apple___healthy
36	Strawberry___Leaf_scorch
37	Potato___healthy
38	Soybean___healthy

Possible use cases in different domain

5. Data Augment

We can implement data augmentation by using the Augmentor Library.

(<https://github.com/mdbloice/Augmentor>)

But we will not be implementing this right now.

Install the library by :- `pip install Augmentor`

There are multiple functions that you can implement some of them are applied here. For more check <https://augmentor.readthedocs.io/en/master/>

```
import Augmentor

# Initiating the Augmentor Pipeline
p = Augmentor.Pipeline("C:\\Users\\soura\\Desktop\\Images")

# Applying various type of Transformations and augmentation strategies
p.rotate(probability=0.7, max_left_rotation=10, max_right_rotation=10)
p.rotate90(probability=0.5)
p.rotate270(probability=0.5)
p.flip_left_right(probability=0.75)
p.flip_top_bottom(probability=0.5)
p.crop_random(probability=1, percentage_area=0.5)
p.resize(probability=1.0, width=80, height=80)
p.random_brightness(probability = 0.5, min_factor=0.4, max_factor=0.9)
p.random_color(probability=0.5, min_factor=0.4, max_factor=0.9)
p.random_contrast(probability=0.5, min_factor=0.9, max_factor=1.4)
p.random_distortion(probability=0.5, grid_width=7, grid_height=8,
magnitude=9)
p.random_erasing(probability=0.5, rectangle_area=0.4)
p.zoom(probability=0.7, min_factor=1.1, max_factor=1.5)

#change the samples size according to requirements
p.sample(100)
```

6. Train test validation

7. Training

7.1 Installation and Setup

Here we will be using the monk framework (<https://clever-noyce-f9d43f.netlify.com/#/>) which is built on top of Pytorch to train and test our model.

Installation

1. Clone library from github

```
git clone https://github.com/Tessellate-Imaging/monk_v1
```

2. Setup virtual environment

```
virtualenv -p python3 monk
```

```
workon monk
```

3. Setup Dependencies

Install dependencies for Linux CPU-only

```
cd monk_v1/installation
```

```
pip install -r requirements-cpu.txt
```

Install dependencies for MacOS CPU-only

```
cd monk_v1/installation
```

```
pip install -r requirements-cpu_macos.txt
```

Install dependencies for systems with GPU

For CUDA == 9.0

```
cd monk_v1/installation
```

```
pip install -r requirements-cu9.txt
```

For CUDA == 10.0 (Colab/Kaggle)

```
cd monk_v1/installation
```

```
pip install -r requirements-cu10.txt
```

Dataset Download

```
wget https://www.dropbox.com/s/hgt9uystjlinzlp/plantVillage.zip
```

```
unzip plantVillage.zip
```

Import Monk library

Place monk inside your project folder

```
import os
```

```
import sys
```

```
sys.path.append("./monk/")
```

```
from pytorch_prototype import prototype
```

```
#import system packages
import sys
import os
sys.path.append("./monk_v1/monk")
```

```
from pytorch_prototype import prototype
```

Create Experiment Setup

Step 1. - Create experiment

```
experiment = prototype(verbose=1)
experiment.Prototype("plant_disease", "experiment1")
experiment.Default(dataset_path=["./dataset/train", "./dataset/val"], model_name="resnet18",
freeze_base_network=True, num_epochs=5);
experiment.train()
```

```
# Create experiment setup initializing the "experiment" object
```

```
experiment = prototype(verbose=1);
experiment.Prototype("plant_disease", "experiment1");
```

```
experiment.Default(dataset_path=[r"/home/paperspace/plantVillage/dataset/train",
r"/home/paperspace/plantVillage/dataset/val"],
model_name="resnet18", freeze_base_network=True, num_epochs=5);
```

Step2. – Using the model finder to find the best parameters for model training.

```
model_fetch = "Model_Finder";
```

```
models = [{"resnet34", True, True}, {"resnet50", False, True}, {"densenet121", False, True}, {"densenet169", True, True}, {"densenet201", True, True}];
```

- First element in the list — Model Name
- Second element in the list — Boolean value to freeze base network or not
- Third element in the list — Boolean value to use pretrained model as the starting point or not

Number of epochs for each experiment to run

```
epochs=5;
```

Percentage of original dataset to take in for experimentation

```
percent_data=10;
```

"keep_all" - Keep all the sub experiments created

"keep_non" - Delete all sub experiments created

```
experiment.Analyse_Models(model_fetch , models, percent_data, num_epochs=epochs,  
state="keep_none");
```

```
# Project Name: Model_Finder  
model_fetch = "Model_Finder";  
  
# Analyzing Model  
# In the first element list- Model Names  
# In the Second element list - Boolean value to freeze base network or not  
# In the Third element list - Boolean value to use pretrained model as the starting point or not  
models = [{"resnet34", True, True}, {"resnet50", False, True},  
          {"densenet121", False, True}, {"densenet169", True, True}, {"densenet201", True, True}];  
  
# Number of epochs for each experiment to run  
epochs=5;  
  
# Percentage of original dataset to take in for experimentation  
percent_data=10;  
  
# "keep_all" - Keep all the sub experiments created  
# "keep_non" - Delete all sub experiments created  
experiment.Analyse_Models(model_fetch, models, percent_data, num_epochs=epochs, state="keep_none");
```

Comparing Experiments

Comparison ID: Comparison_Model_Finder

Generated statistics post all epochs

Experiment Name	Train Acc	Val Acc	Train Loss	Val Loss
Model_resnet34_freeze_base_pretrained	0.824629	0.8407	0.614053	0.468221
Model_resnet50_unfreeze_base_pretrained	0.960775	0.973297	0.133728	0.112808
Model_densenet121_unfreeze_base_pretrained	0.967617	0.979742	0.124	0.066871
Model_densenet169_freeze_base_pretrained	0.880502	0.896869	0.466162	0.339534
Model_densenet201_freeze_base_pretrained	0.873432	0.921731	0.486347	0.304517

Update the model

```
experiment.update_model_name("densenet121");
```

```
experiment.update_freeze_base_network(True);
```

```
experiment.update_use_pretrained(True);
```

```
experiment.Reload();
```

```
In [7]: ## Updating Model Architecture
experiment.update_model_name("densenet121");
experiment.update_freeze_base_network(True);
experiment.update_use_pretrained(True);
experiment.Reload();
```

Finding the best hyperparameters for the model

1. Batch Size

Finding the right batch size for training

Analysis Name :- batch_fetch

```
batch_fetch= "Batch_Size_Finder";
```

Best batch sizes to explore

```
batch_sizes = [4, 8, 16, 32];
```

Number of epochs for each experiment to run

```
epochs = 10;
```

Percentage of original dataset to take in for experimentation


```
percent_data = 10;
```

```
# "keep_all" - Keep all the sub experiments created
```

```
# "keep_non" - Delete all sub experiments created
```

```
experiment.Analyse_Batch_Sizes(batch_fetch, batch_sizes, percent_data, num_epochs=epochs,  
state="keep_none");
```

```
# Project Name: Batch_Size_Finder  
batch_fetch = "Batch_Size_Finder";  
  
# Batch sizes to explore  
batch_sizes = [4, 8, 16, 32];  
  
# Num epochs for each experiment to run  
epochs = 10;  
  
# Percentage of original dataset to take in for experimentation  
percent_data = 10;  
  
# "keep_all" - Keep all the sub experiments created  
# "keep_non" - Delete all sub experiments created  
experiment.Analyse_Batch_Sizes(batch_fetch, batch_sizes, percent_data, num_epochs=epochs, state="keep_none");
```

Comparing Experiments

Comparison ID: Comparison_Batch_Size_Finder

Generated statistics post all epochs

Experiment Name	Train Acc	Val Acc	Train Loss	Val Loss
Batch_Size_4	0.896921	0.929098	0.368656	0.236665
Batch_Size_8	0.934322	0.918048	0.274333	0.271655
Batch_Size_16	0.94732	0.900552	0.286326	0.380388
Batch_Size_32	0.936374	0.833333	0.390712	0.613298

Update batch size

```
## Update Batch Size
```

```
experiment.update_batch_size(8);
```

```
experiment.Reload();
```

```
## Updating Batch Size  
experiment.update_batch_size(8);  
experiment.Reload();
```

2. Find the correct input size dimension

Analysis Project Name

```
inputsize_fetch = "Input_Size_Finder";
```

Input sizes to explore

```
input_sizes = [224, 256, 512];
```

Num epochs for each experiment to run

```
epochs=5;
```

Percentage of original dataset to take in for experimentation

```
percent_data=10;
```

"keep_all" - Keep all the sub experiments created

"keep_non" - Delete all sub experiments created

```
experiment.Analyse_Input_Sizes(inputsize_fetch , input_sizes, percent_data, num_epochs=epochs, state="keep_none");
```

```
# Project Name: Input Size Finder
input_fetch = "Input_Size_Finder";

# Input sizes to explore
input_sizes = [224, 256, 512];

# Num epochs for each experiment to run
epochs=5;

# Percentage of original dataset to take in for experimentation
percent_data=10;

# "keep_all" - Keep all the sub experiments created
# "keep_non" - Delete all sub experiments created
experiment.Analyse_Input_Sizes(input_fetch, input_sizes, percent_data, num_epochs=epochs, state="keep_none");
```

Comparing Experiments

Comparison ID: Comparison_Input_Size_Finder

Generated statistics post all epochs

Experiment Name	Train Acc	Val Acc	Train Loss	Val Loss
Input_Size_224	0.91106	0.881215	0.41042	0.405622
Input_Size_256	0.904675	0.89779	0.429658	0.388663
Input_Size_512	0.907412	0.922652	0.415748	0.331154

Update input size

```
experiment.update_input_size(224);
```

```
experiment.Reload();
```

```
## Update Input Size  
experiment.update_input_size(224);  
experiment.Reload();
```

3. Finding the perfect Learning rate

Analysis Project Name

```
lr_fetch = "Learning_Rate_Finder"
```

Learning rates to explore

```
lrs = [0.01, 0.005, 0.001, 0.0001];
```

Num epochs for each experiment to run

```
epochs=5
```

Percentage of original dataset to take in for experimentation

```
percent_data=10
```

"keep_all" - Keep all the sub experiments created

"keep_non" - Delete all sub experiments created

```
experiment.Analyse_Learning_Rates(lr_fetch, lrs, percent_data, num_epochs=epochs, state="keep_none");
```

```
# Project Name: Learning_Rate_Finder
lr_fetch = "Learning_Rate_Finder"

# Learning rates to explore
lrs = [0.01, 0.005, 0.001, 0.0001];

# Num epochs for each experiment to run
epochs=5

# Percentage of original dataset to take in for experimentation
percent_data=10

# "keep_all" - Keep all the sub experiments created
# "keep_non" - Delete all sub experiments created
experiment.Analyse_Learning_Rates(lr_fetch, lrs, percent_data, num_epochs=epochs, state="keep_none");
```

Comparing Experiments

Comparison ID: Comparison_Learning_Rate_Finder

Generated statistics post all epochs

Experiment Name	Train Acc	Val Acc	Train Loss	Val Loss
Learning_Rate_0.01	0.904219	0.879374	0.42099	0.431651
Learning_Rate_0.005	0.878905	0.888582	0.620732	0.476572
Learning_Rate_0.001	0.691448	0.765193	1.59023	1.33622
Learning_Rate_0.0001	0.270924	0.276243	3.01024	2.93441

Update Learning Rate

Update Learning Rate

```
experiment.update_learning_rate(0.01);
```

```
experiment.Reload();
```

```
## Update Learning Rate
experiment.update_learning_rate(0.01);
experiment.Reload();
```

Finding the best Optimizer

Analysis Project Name

```
optimizer_fetch = "Optimiser_Finder";
```

Optimizers to explore

```
optimizers = ["sgd", "adam", "adamax", "rmsprop"];
```

Num epochs for each experiment to run

```
epochs = 5;
```

Percentage of original dataset to take in for experimentation

```
percent_data = 10;
```

"keep_all" - Keep all the sub experiments created

"keep_non" - Delete all sub experiments created

```
experiment.Analyse_Optimizers(optimizer_fetch, optimizers, percent_data,
num_epochs=epochs, state="keep_none");
```

```
# Project Name: Optimiser Finder
optimizer_fetch = "Optimiser_Finder";

# Optimizers to explore
optimizers = ["sgd", "adam", "adamax", "rmsprop"]; #Model name, learning rate

# Num epochs for each experiment to run=
epochs = 5;

# Percentage of original dataset to take in for experimentation
percent_data = 10;

# "keep_all" - Keep all the sub experiments created
# "keep_non" - Delete all sub experiments created
experiment.Analyse_Optimizers(optimizer_fetch, optimizers, percent_data, num_epochs=epochs, state="keep_none");
```

Comparing Experiments

Comparison ID: Comparison_Optimiser_Finder

Generated statistics post all epochs

Experiment Name	Train Acc	Val Acc	Train Loss	Val Loss
Optimizer_sgd	0.903307	0.868324	0.424622	0.43945
Optimizer_adam	0.947548	0.935543	0.202524	0.340577
Optimizer_adamax	0.931585	0.940147	0.229603	0.197292
Optimizer_rmsprop	0.86431	0.494475	1.10529	8.14648

Updating the optimizer

```
experiment.optimizer_adamax(0.001);
```

```
experiment.Reload();
```

```
## Update Optimiser  
  
experiment.optimizer_adamax(0.001);  
experiment.Reload();
```

Set intermediate state saving to True

```
experiment.update_save_intermediate_models(True);
```

```
experiment.update_save_intermediate_models(True);
```

```
#train model with 5 epochs  
experiment.Train();
```

Here we training for only for 5 epochs

```
# Again we're Creating experiment setup initializing with "experiment2" object  
experiment = prototype(verbose=1);  
experiment.Prototype("plant_disease", "experiment2", copy_from=["plant_disease", "experiment1"]);
```

Again we are creating a new experiment and copy it from the previous one.

Update the number of epochs to 100

```
experiment.update_num_epochs(100)
```

```
experiment.Reload()
```

```
# update number of epochs  
experiment.update_num_epochs(100)  
experiment.Reload()
```

Train

```
experiment.Train();
```

```
#train model with 100 epochs  
experiment.Train()
```


Now the model training will be finally completed after few hours based on the system you are training.

Now three models will be saved in the workspace folder of the following experiment.

Performance tuning

Possible tuning analytics

Comparison with benchmarks

	AlexNet		GoogleNet	
	Transfer learning	Training from scratch	Transfer learning	Training from scratch
TRAIN: 200%, TEST: 80%				
Color	0.9736 _[0.9742, 0.9737, 0.9738]	0.9118 _[0.9137, 0.9132, 0.9130]	0.9820 _[0.9824, 0.9821, 0.9821]	0.9430 _[0.9440, 0.9431, 0.9429]
Grayscale	0.9361 _[0.9368, 0.9369, 0.9371]	0.8524 _[0.8539, 0.8555, 0.8553]	0.9563 _[0.9570, 0.9564, 0.9564]	0.8828 _[0.8842, 0.8835, 0.8841]
Segmented	0.9724 _[0.9727, 0.9727, 0.9726]	0.8945 _[0.8956, 0.8963, 0.8969]	0.9808 _[0.9810, 0.9808, 0.9808]	0.9377 _[0.9388, 0.9380, 0.9380]
TRAIN: 400%, TEST: 60%				
Color	0.9860 _[0.9861, 0.9861, 0.9860]	0.9555 _[0.9557, 0.9558, 0.9558]	0.9914 _[0.9914, 0.9914, 0.9914]	0.9729 _[0.9731, 0.9729, 0.9729]
Grayscale	0.9584 _[0.9588, 0.9589, 0.9588]	0.9088 _[0.9090, 0.9101, 0.9100]	0.9714 _[0.9717, 0.9716, 0.9716]	0.9361 _[0.9364, 0.9363, 0.9364]
Segmented	0.9812 _[0.9814, 0.9813, 0.9813]	0.9404 _[0.9409, 0.9408, 0.9408]	0.9896 _[0.9896, 0.9896, 0.9896]	0.9643 _[0.9647, 0.9642, 0.9642]
TRAIN: 50%, TEST: 50%				
Color	0.9896 _[0.9897, 0.9896, 0.9897]	0.9644 _[0.9647, 0.9647, 0.9647]	0.9916 _[0.9916, 0.9916, 0.9916]	0.9772 _[0.9774, 0.9773, 0.9773]
Grayscale	0.9661 _[0.9663, 0.9663, 0.9663]	0.9312 _[0.9315, 0.9318, 0.9319]	0.9788 _[0.9789, 0.9788, 0.9788]	0.9507 _[0.9510, 0.9507, 0.9509]
Segmented	0.9867 _[0.9868, 0.9868, 0.9869]	0.9551 _[0.9552, 0.9555, 0.9556]	0.9909 _[0.9910, 0.9910, 0.9910]	0.9720 _[0.9721, 0.9721, 0.9722]
TRAIN: 600%, TEST: 40%				
Color	0.9907 _[0.9908, 0.9908, 0.9907]	0.9724 _[0.9725, 0.9725, 0.9725]	0.9924 _[0.9924, 0.9924, 0.9924]	0.9824 _[0.9825, 0.9824, 0.9824]
Grayscale	0.9686 _[0.9689, 0.9688, 0.9688]	0.9388 _[0.9396, 0.9395, 0.9391]	0.9785 _[0.9789, 0.9786, 0.9787]	0.9547 _[0.9554, 0.9548, 0.9551]
Segmented	0.9855 _[0.9856, 0.9856, 0.9856]	0.9595 _[0.9597, 0.9597, 0.9596]	0.9905 _[0.9906, 0.9906, 0.9906]	0.9740 _[0.9743, 0.9740, 0.9745]
TRAIN: 80%, TEST: 20%				
Color	0.9927 _[0.9928, 0.9927, 0.9928]	0.9782 _[0.9786, 0.9782, 0.9782]	0.9934 _[0.9935, 0.9935, 0.9935]	0.9836 _[0.9839, 0.9837, 0.9837]
Grayscale	0.9726 _[0.9728, 0.9727, 0.9725]	0.9449 _[0.9451, 0.9454, 0.9452]	0.9800 _[0.9804, 0.9801, 0.9798]	0.9621 _[0.9624, 0.9621, 0.9621]
Segmented	0.9891 _[0.9893, 0.9891, 0.9892]	0.9722 _[0.9725, 0.9724, 0.9723]	0.9925 _[0.9925, 0.9925, 0.9924]	0.9824 _[0.9827, 0.9824, 0.9822]

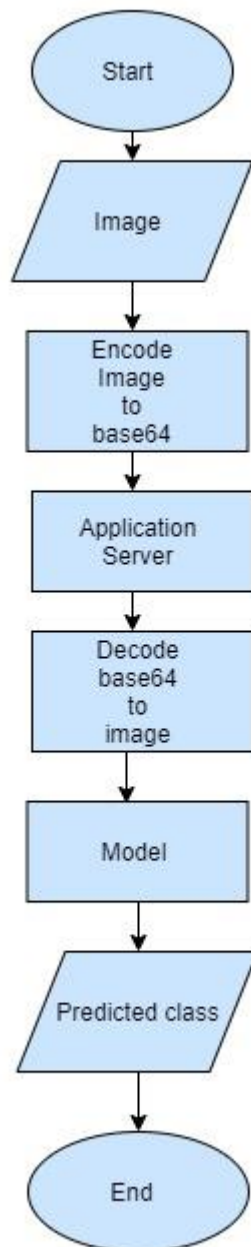
Each cell in the table represents the mean F_1 score (mean precision, mean recall, overall accuracy) for the corresponding experimental configuration. The bold values are the F_1 scores of the best performing models in the respective row/column.

These are the various benchmarks score when using different CNN architectures like AlexNet, GoogleNet with variety of transfer learning or scratch learning and different data size.

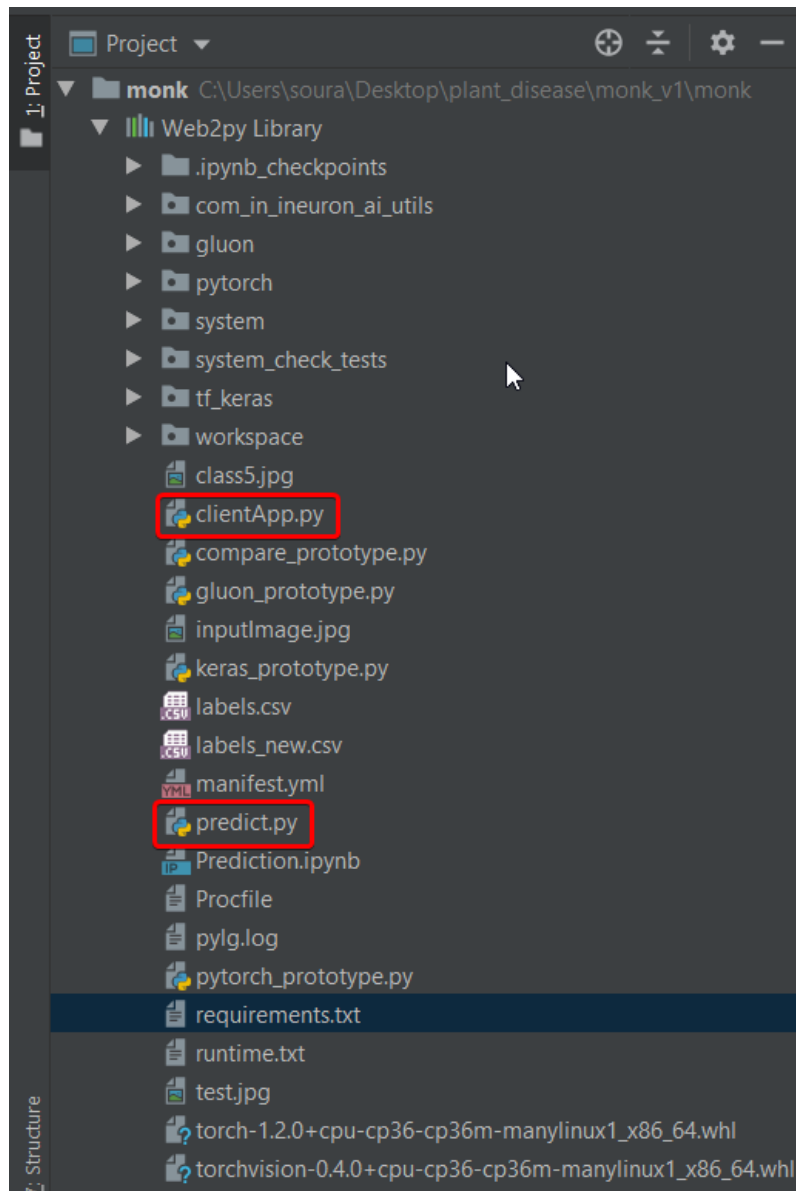
Deployment

We will be deploying the model to the Pivotal Cloud Foundry platform.

This is workflow diagram for prediction of diseases using the trained model.



Now let's see the Plant Disease Classification project folder structure.



requirements.txt file consists of all the packages that you need to deploy the app in the cloud.

```
clientApp.py × results.html × index.html × labels_new.csv × predict.py × manifest
8
9     app = Flask(__name__)
10    CORS(app)
11    class ClientApp:
12    def __init__(self):
13        self.filename = "inputImage.jpg"
14        self.objectDetection = plant_prediction(img_name=self.filename)
15
16    @app.route("/", methods=['GET'])
17    def home():
18        return render_template("index.html")
19
20    @app.route("/predict", methods=['POST'])
21    @cross_origin()
22    def predictRoute():
23        #image = request.json['image']
24        image = request.form['content']
25        decodeImage(image, clApp.filename)
26        result = clApp.objectDetection.predict_label()
27        #return jsonify(result)
28        return render_template("results.html", result=result)
29
30    port = int(os.getenv("PORT"))
31    if __name__ == "__main__":
32        clApp = ClientApp()
33        app.run(host='0.0.0.0', port=port)
```

ClientApp.py is the entry point of our application, where the flask server starts. Here we will be decoding a base64 to an image, and then we will be doing predictions.

```
clientApp.py × predict.py × requirements.txt × manifest.yml × Procfile × runtime.txt ×
1  import csv
2  from pytorch_prototype import prototype
3
4  class plant_prediction:
5      def __init__(self, img_name):
6          self.img_name = img_name
7
8      def read_labels(self):
9          mydict = {}
10         with open('labels.csv', mode='r') as infile:
11             reader = csv.reader(infile)
12             with open('labels_new.csv', mode='w') as outfile:
13                 writer = csv.writer(outfile)
14                 mydict = {rows[0]: rows[1] for rows in reader}
15         return mydict
16
17     def predict_label(self):
18         experiment = prototype(verbose=1)
19         experiment.Prototype("plant_disease", "exp3", eval_infer=True)
20         predictions = experiment.Infer(img_name=self.img_name, return_raw=True)
21         pred_class = predictions['predicted_class']
22         label_dict = self.read_labels()
23         out_label = label_dict[pred_class]
24         return out_label
25
```

This is the **predict.py** file where the predictions take place based on the image we are giving input to the model.

```

clientApp.py × predict.py × manifest.yml × Procfile × runtime.txt ×
1
2  ---
3  applications:
4  - name: agriculture_plant_disease_classification
5    memory: 1024MB
6    disk_quota: 2GB
7    random-route: true
8    parameters:
9      memory: 1024M
10     buildpack: 'python_buildpack'

```

manifest.yml:- This file contains the instance configuration, app name and buildpack language.

```

clientApp.py × predict.py × manifest.yml × Procfile × runtime.txt ×
1  web: python clientApp.py --master --processes 4 --threads 2

```

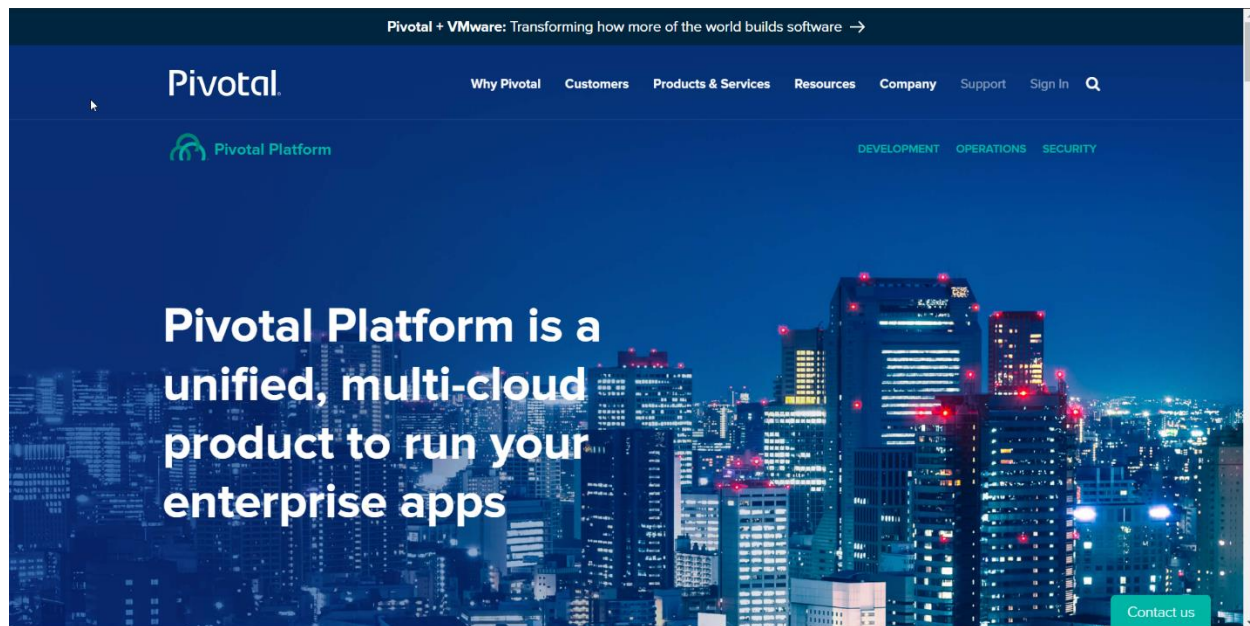
Procfile :- It contains the entry point of the app.

```

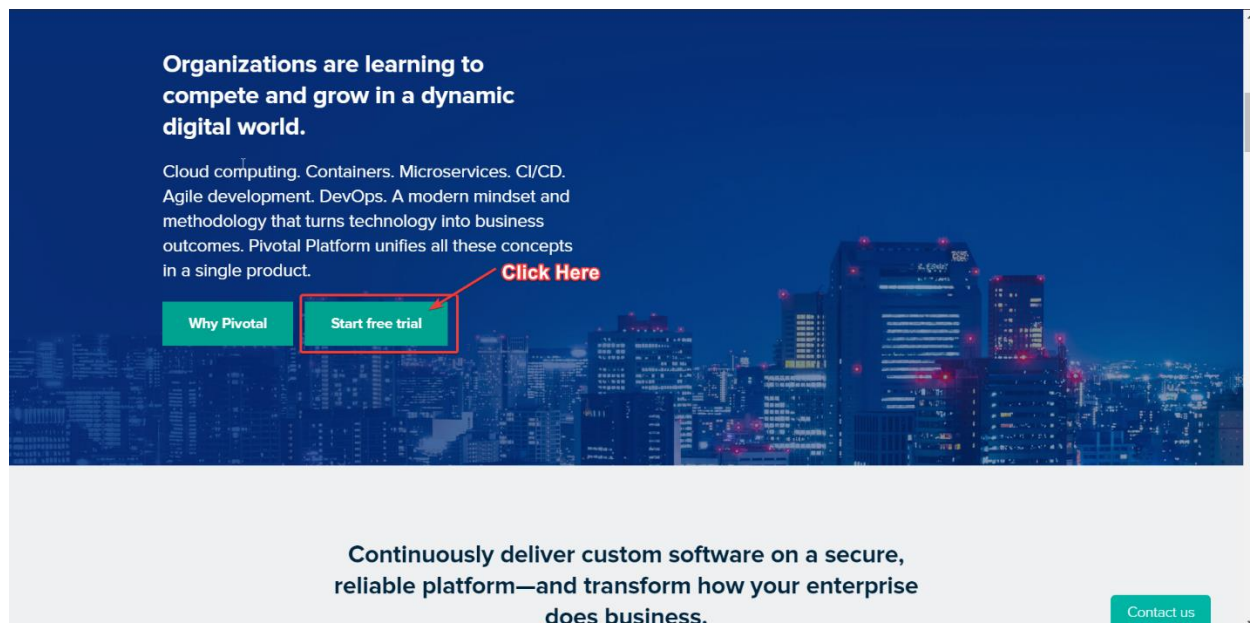
clientApp.py × predict.py × manifest.yml × Procfile × runtime.txt ×
1  python-3.6.8

```

runtime.txt:- It contains the Python version number.



Visit the official website <https://pivotal.io/platform>.



Scroll Down to see the **Start Trial Button**

Try Pivotal Platform on the Public Cloud

15-minute tutorial for learning Pivotal Platform app deployment concepts

Introduction

Install the CF CLI


Deploy the Sample App

View the Logs

Connect a Database

Scale the App

Next Steps

 Switch to Local

Introduction

Use this tutorial to get a simple Spring microservice up and running on Pivotal Platform. To save time, we will be using Pivotal Web Services, an instance of Pivotal Platform hosted by Pivotal.

Ensure you have:

- a free **Pivotal Web Services** account
- familiarity with command line interfaces

I'm ready to continue →

Click

Contact us

Click on the start trial button and next interface will open. Then I will click on **I'm ready to continue**

Introduction

Install the CF CLI


Deploy the Sample App

View the Logs

Connect a Database

Scale the App

Next Steps

 Switch to Local

Install the CF CLI

Download and install the Cloud Foundry Command Line Interface (cf CLI):

DOWNLOAD FOR WINDOWS 64 BIT

Try the following command to test that the cf CLI works:

```
> cf help
```

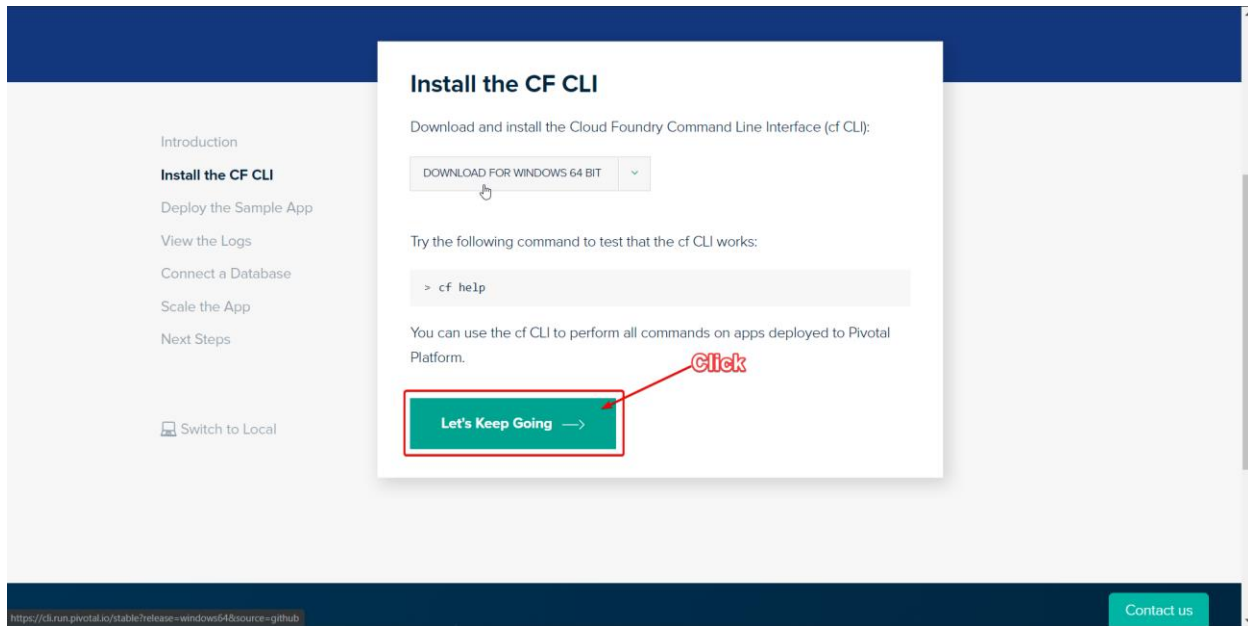
You can use the cf CLI to perform all commands on apps deployed to Pivotal Platform.

Let's Keep Going →

Click

Contact us

Click on Download for **Windows 64 bit** then zip file will be downloaded. Keep it for future uses.



Install the CF CLI

Download and install the Cloud Foundry Command Line Interface (cf CLI):

[DOWNLOAD FOR WINDOWS 64 BIT](#)

Try the following command to test that the cf CLI works:

```
> cf help
```

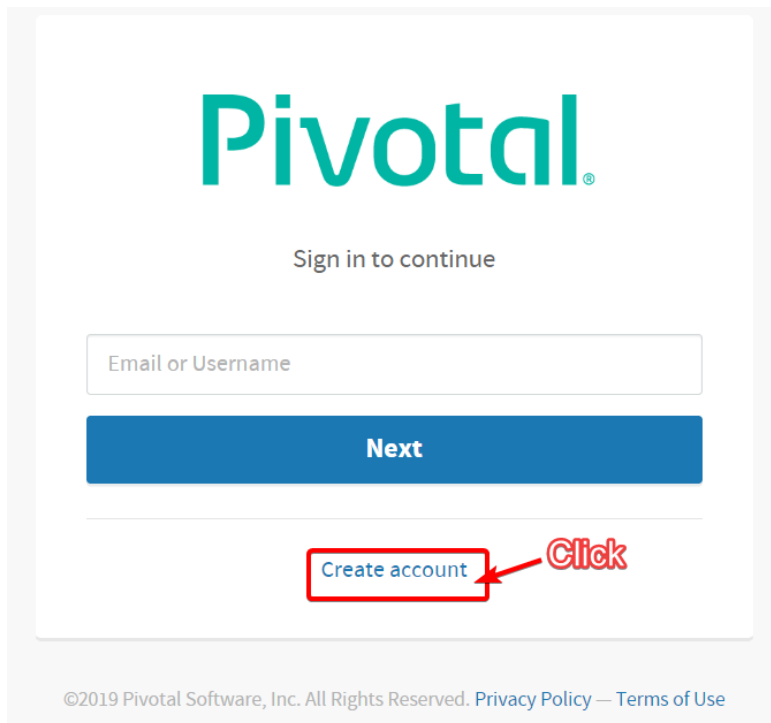
You can use the cf CLI to perform all commands on apps deployed to Pivotal Platform.

[Let's Keep Going](#) →

<https://cli.run.pivotal.io/stable/release-windows64?source=github>

[Contact us](#)

Now click on **Let's Keep Going**



Pivotal

Sign in to continue

Email or Username

[Next](#)

[Create account](#)

©2019 Pivotal Software, Inc. All Rights Reserved. [Privacy Policy](#) — [Terms of Use](#)

Click on **Create Your Account**

Pivotal®

Create your Pivotal Account

First name

Last name

Email address

Password

Password confirmation

☐ I agree to the terms of Pivotal's [Privacy Policy](#)

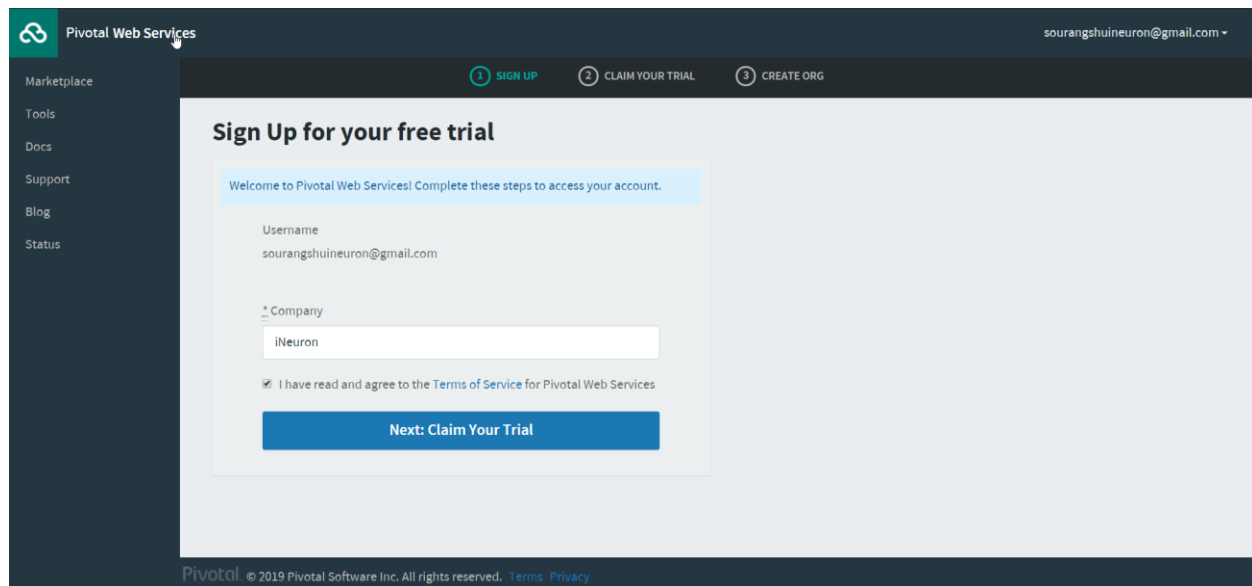
Sign Up

Already have an account? [Sign In](#)

Fill Up your Details For registration

Do the email verification

Then login in again



Pivotal Web Services

sourangshuineuron@gmail.com

1 SIGN UP 2 CLAIM YOUR TRIAL 3 CREATE ORG

Sign Up for your free trial

Welcome to Pivotal Web Services! Complete these steps to access your account.

Username
sourangshuineuron@gmail.com

* Company
iNeuron

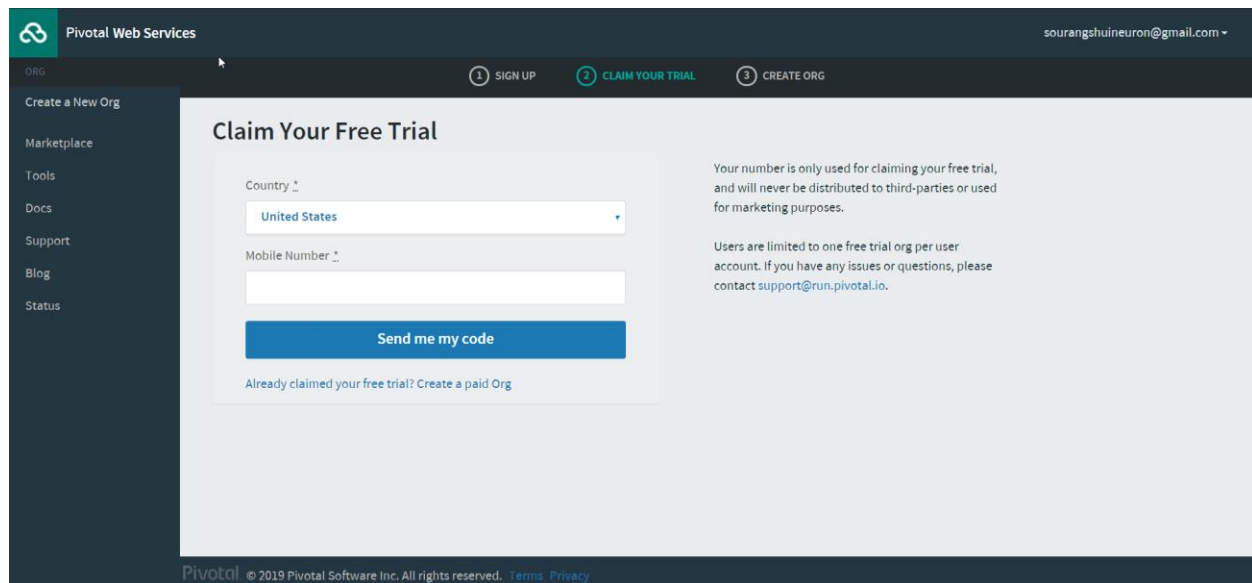
☒ I have read and agree to the Terms of Service for Pivotal Web Services

Next: Claim Your Trial

Pivotal © 2019 Pivotal Software Inc. All rights reserved. [Terms](#) [Privacy](#)

After logging you will see this screen below and start your free trial.

Write any Company or which one you prefer



Pivotal Web Services

sourangshuineuron@gmail.com

ORG

1 SIGN UP 2 CLAIM YOUR TRIAL 3 CREATE ORG

Claim Your Free Trial

Country
United States

Mobile Number

Send me my code

Already claimed your free trial? [Create a paid Org](#)

Your number is only used for claiming your free trial, and will never be distributed to third-parties or used for marketing purposes.

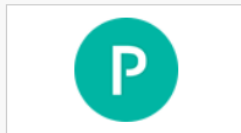
Users are limited to one free trial org per user account. If you have any issues or questions, please contact support@run.pivotal.io.

Pivotal © 2019 Pivotal Software Inc. All rights reserved. [Terms](#) [Privacy](#)

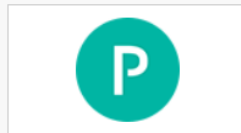
Enter your **Mobile Number** for Verification

Pivotal

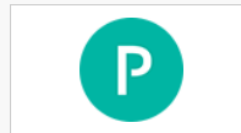
Where to?



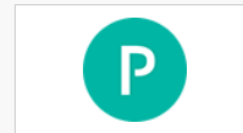
Pivotal Web Services



Pivotal Network



Partner Portal



Pivotal Support

Click on **Pivotal Web Services**

Pivotal Web Services

sourangshuineuron@gmail.com

1 SIGN UP 2 CLAIM YOUR TRIAL 3 CREATE ORG

Create a Trial Org

Org (or Project) Name *

App_Development

Have a promo code? Click here!

Start Free Trial

Organization (org) is a development account that encompasses computing resources, apps, and services. It can be owned and used by an individual or by multiple collaborators.

Set the org name to be the name of the project you'll be working on or the name of your team. Don't worry - you can change this name at any time!

Free Trial

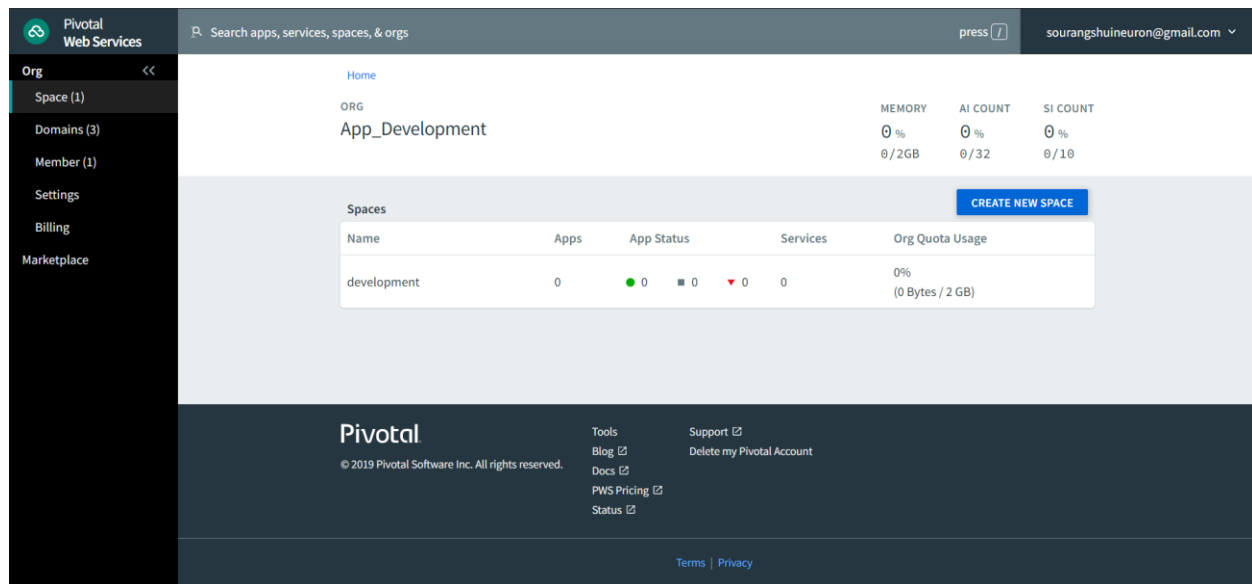
- \$87 credit for app usage to use for up to 1 year
- Up to 2GB of memory to share across app instances
- Choice of free marketplace services to try
- Unlimited collaborators

Upgrade at any time for

- Access to 25GB of memory at \$0.03/GB hr
- Premium service plans
- Pay for only what you use

Pivotal © 2019 Pivotal Software Inc. All rights reserved. [Terms](#) [Privacy](#)

Give any **Org** name



The screenshot shows the Pivotal Web Services interface. On the left is a dark sidebar with navigation links: Org, Space (1), Domains (3), Member (1), Settings, Billing, and Marketplace. The main content area has a top bar with a search bar and a user profile. Below this, the 'Org' section displays 'App_Development' with resource usage: MEMORY (0% / 2GB), AI COUNT (0% / 32), and SI COUNT (0% / 10). A 'Spaces' table lists the 'development' space with 0 apps and 0 services. A 'CREATE NEW SPACE' button is visible. The footer contains Pivotal branding, copyright information, and links to Tools, Support, and legal documents.

Name	Apps	App Status	Services	Org Quota Usage
development	0	0	0	0% (0 Bytes / 2 GB)

Now you are inside your Org and by default development space is created in your org. You can push your apps here.

The cloud signup process is done and setup is ready for us to push the app.

Previously you have downloaded the **CLI.zip** file. Unzip the file and install the .exe file with admin rights.

After successful installation, you can verify by opening your CMD and type **cf**.

Then you will get a screen which is shown below



```
Microsoft Windows [Version 10.0.18362.418]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\soura>cf
cf version 6.46.1+4934877ec.2019-08-23, Cloud Foundry command line tool
Usage: cf [global options] command [arguments...] [command options]

Before getting started:
  config      login,l      target,t
  help,h      logout,lo

Application lifecycle:
  apps,a      run-task,rt   events
  push,p      logs          set-env,se
  start,st    ssh          create-app-manifest
  stop,sp     app          delete,d
  restart,rs  env,e
  restage,rg  scale

Services integration:
  marketplace,m  create-user-provided-service,cups
  services,s     update-user-provided-service,uups
  create-service,cs  create-service-key,csk
  update-service    delete-service-key,dsk
  delete-service,ds  service-keys,sk
  service           service-key
  bind-service,bs   bind-route-service,brs
  unbind-service,us unbind-route-service,urs

Route and domain management:
  routes,r      delete-route   create-domain
  domains       map-route
  create-route  unmap-route

Space management:
  spaces      create-space   set-space-role
  space-users delete-space   unset-space-role

Org management:
  orgs,o      set-org-role
  org-users   unset-org-role

CLI plugin management:
  plugins      add-plugin-repo   repo-plugins
  install-plugin list-plugin-repos

Commands offered by installed plugins:

Global options:
```

If you see this screen in your CMD the installation is successful.

Now type the command to login via cf-cli

```
cf login -a https://api.run.pivotal.io
```

Next enter your email and password. Now you are ready to push your app.

Now let's go to the app which we have built.

```
Microsoft Windows [Version 10.0.18363.535]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\soura>cf login
API endpoint: https://api.run.pivotal.io

Email> sourangshuineuron@gmail.com

Password>
Authenticating...
OK

Targeted org App_Development

Targeted space development

API endpoint: https://api.run.pivotal.io (API version: 2.144.0)
User: sourangshuineuron@gmail.com
Org: App_Development
Space: development
```

Navigate to the project folder after downloading from the given below link :-

```
C:\Users\soura\Desktop\plant_disease\monk_v1\monk>cf push
```

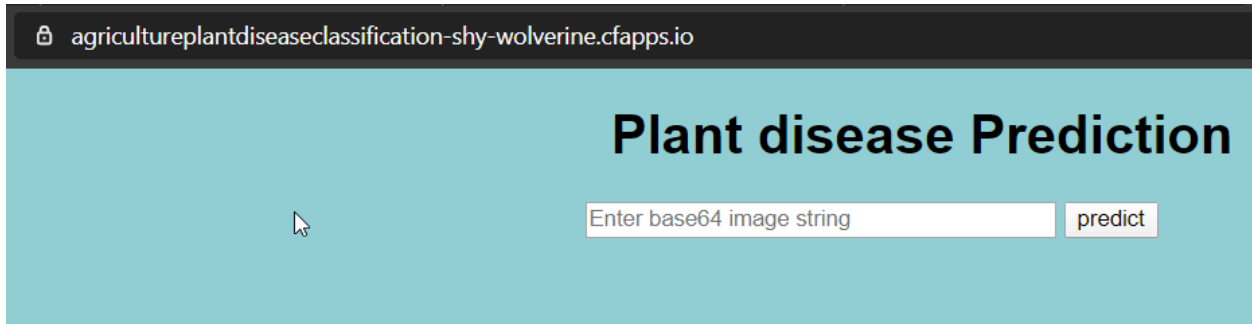
After the app is successfully deployed in cloud you will be seeing the below screen with the route.

```
name: agriculture_plant_disease_classification
requested state: started
routes: agricultureplantdiseaseclassification-shy-wolverine.cfapps.io
last uploaded: Sat 28 Dec 18:55:45 IST 2019
stack: cflinuxfs3
buildpacks: python

type: web
instances: 1/1
memory usage: 1024M
start command: python clientApp.py --master --processes 4 --threads 2

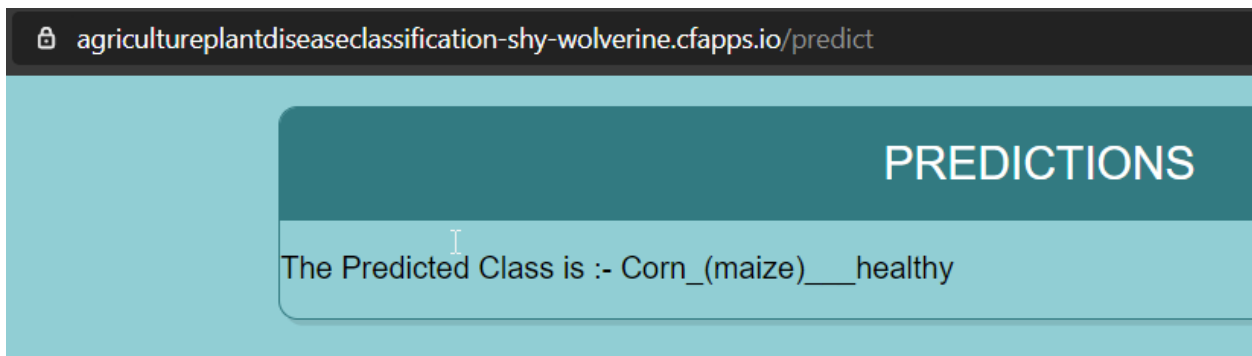
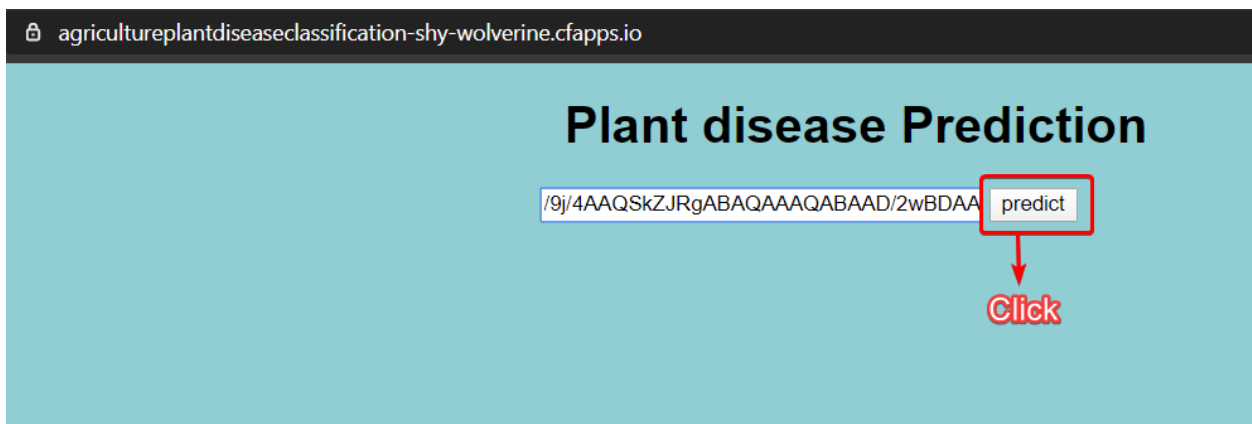
state since cpu memory disk details
#0 running 2019-12-28T13:27:26Z 99.6% 86.5M of 1G 1.3G of 2G
```

Lets visit the route in your browser



Visit this website to encode the predicted image to base64 encoding.

<https://base64.guru/converter/encode/image>



Finally we get the predicted class.



Conclusion

Hence we have successfully deployed the plant prediction model in cloud and we are able to do predictions.