July 28, 2020
Version 1.1

Team 1

PROJECT: Geospatial Assessment of Urban Farming Potential in Metropolitan Areas

Architecture Design Document (ADD)

List of Figures

INDEX

1. Introduction

This version of the document presents the initial concept of the architecture design of the map generation system which is being developed. A short description will provide a first overview together with the current status of the development of the map generation system, as well as a flow chart with the description of each processing step. Finally, the required input and the generated output will be presented and explained.

1.1 Mission objective

The outcome of the geospatial assessment of urban farming (UF) potential in metropolitan areas will be digital maps indicating potential rooftops and vacant lands in metropolitan areas. The generation system of the UF potential map shall be generalized to be applied to all metropolitan areas with adequate resolution geospatial data provided. For a detailed look into the system requirements, please refer to SRD.

1.2 Software Description

The code for image classification software should ideally be written parallelly in both Jupyter notebook and the Google Earth Engine (GEE). Jupyter notebook uses python as the programming language. So far we have written everything using Javascript in GEE.

The GEE platform enables researchers and developers to classify regions, detect changes and trends and monitor different features of our planet. It uses Javascript as the programming language. GEE can recall different satellite images that are available free of cost, as well as a variety of aerial images libraries. Hence, importing satellite data to GEE is a simple task once we identify a source with suitable resolution to detect rooftops.

On both platforms, the data will then be processed to select the area of interest (in this case metropolitan areas in the US) and extract useful information to define both the training and validation data for the machine learning algorithm (i.e., CNN for Jupyter notebooks and CART for GEE). Once the algorithm has been trained, it will be used to classify rooftops in the image, evaluate the result with respect to the validation set. These classified images will then be updated over time to examine additional or loss of rooftops. The final selection of the suitable parcels will be from the outcome of the classified images based on weighted cultivation criteria.

1.2.1 Location Selection

We selected New York City as our research location, abbreviated as NYC, which is located on the Atlantic coast of southeastern NYC. It is the largest city and port in the United States and its metropolitan area is one of the largest city circles in the world.

As one of the most developed cities in the world, NYC has a large potential of urban farming since it has dense high-rise buildings, providing a large area of vacant rooftops. Large populations and strained urban land also make urban agriculture an important part of the region's demand for a stable food supply. In the meantime, urban agriculture has been developed in New York for several years, and it has had the largest commercial rooftop farm in the world. It is of high commercial value to select this location for research, and the experimental results will be more obvious.

1.3 Current Status

Initially the image classification was done with Sentinel-2 data. The outcome was not satisfying due to the poor resolution (20 m). Images with a much better resolution (1m) were found with free access on the GEE platform from the National Agriculture Imagery Program (NAIP). The classification outcome was much better, yet currently we have two problems. One problem is the illumination of the Sun. Due to the diurnal and seasonal variations of sun light, the illumination conditions of trees vary greatly. Some trees in the images are classified as the rooftops under certain illumination conditions. Another problem is that some ground features are classified as rooftops since they share certain common features.

The solutions to these two problems will be our next processing steps. To the first problem, we did NDVI calculation to distinguish the vegetation from other objects. After this step, we were able to mask vegetation pixels in the input image pixels. To the second problem, we used DEM to mask the ground pixels in the input image pixels. After this step, the rooftop detection has been carried out. After the image classification, the accuracy of the final result was evaluated using a confusion matrix. Furthermore, since the software was developed in Python language yet GEE uses JavaScript language, the final step was to convert the code from Javascript on the GEE platform to Python in Jupyter notebooks.

2. Initial brainstorming and developing stages

2.1 Classifiers comparison

2.1.1 Cart classification (Google Earth Engine) - multispectral supervised classification

CART (Classification And Regression Tree) algorithm is a decision tree classification method. It uses a recursive binary segmentation technique, based on a minimum distance estimation function of Gini impurity. In this algorithm the current sample sets are divided into two subsets, so that each non-leaf node has two branches generated. Therefore, decision trees generated by the CART algorithm are simple binary tree structure. To better understand CART classifier, the following elements are identified before applying the classifier,

1. Set of rules to answer the question where to split to new branches
2. Decision rule for stopping at terminal branches
3. Prediction criteria for the variable

Identifying those three elements is the first step to build the classifier. Moreover, CATR is an example of supervised classification techniques. Hence, it is important to identify training data in advance to train the classifier. For example, identifying different pixels as rooftops, vacant lands, and waters means that the result will contain three classes: rooftops, vacant lands, and waters. The rest will be pixels that do not identify as any of the mentioned classes. Therefore, the resolution of training data is important because it is vital to the classification process.

For the purpose of this project, we used CART classifier that is built in Google Earth Engine by performing these three steps,

1. Import the classifier using the command ee.Classifier
2. Train the classifier by typing classifier.train()
3. Classify the image using classify()

In our system we aim to separate the objects in metropolitan images into the rooftop and non-rooftop parts. The target variable is a discrete variable, so classification trees are generated. The pseudocode is as belows:

```
Input: training data set D, stop condition e.
Output: classification tree T.
Method: CART_classification_tree(D,e)
(1) create node N.
```

(2) if it satisfies the condition (it can be that the quantity in D is less than a certain degree or the Gini impurity is less than e, etc.)

return N (N is the mode in D).

(3) traverse each feature j, scan the syncopation point s for each fixed syncopation variable j, obtain the optimal syncopation point s through formula calculation, and divide the data set D into Ds1 and Ds2.N values are the optimal syncopation variable j and syncopation point s.

(4) Recursive call CART_classification_tree (Ds1,e) and CART_classification_tree (Ds2,e).

(5) generate decision classification tree T.

/ / traversing the feature in step (3) should be noted that when the dataset is divided according to the discrete feature branch, the feature is no longer included in the subdataset. According to the continuous feature branch, the sub-data set under each branch must still contain the feature Because the continuous feature may still play a decisive role in the subsequent branching process of the tree.

2.1.2 Support Vector Machine

The main idea of support vector Machine (SVM) is to build a hyperplane as a decision surface, so that the isolation edge between positive and negative examples is maximized.
The pseudocode are as belows:

```
import elefant.kernels.vector
# linear kernelk = elefant.kernels.vector.CLinearKernel()
# Gaussian RBF kernelk = elefant.kernels.vector.CGaussKernel(rbf)
import elefant.estimation.svm.svmclass assvmclass
svm = svmclass.SVC(C, kernel=k)
alpha, b = svm.Train(x, y)
ytest = svm.Test(xtest)
```

2.2 Image segmentation techniques

2.2.1 Convolution Neural Network (CNN)

The Machine Learning technique is employed to do the rooftop detection for the google earth images. This requires prior training samples consisting of input images and output labels for the supervised learning. Among all the machine learning methods, the Convolution Neural Network (CNN) is chosen as the training model as it is popular for image classification.

2.2.2 Histogram-based threshold

In general, the pixel values of the rooftop area in the image are often distinctive, showing a uniform and bright color in a sheet shape. From the histogram, these pixels concentrate in the right part with a large amount. Therefore, the threshold-based method from the histogram can be used to separate the rooftop areas.

2.2.3 Edge detection

To segment an image, one can detect different edges in the image, which can give an idea of what separates two bodies in an image. Edges are defined as discontinuities among pixels or a sudden and large change in neighbor pixel values. To detect edges, one can use gradient operators, which detect the maximum change in pixel values and to do so, finding first and second derivatives are necessary. However, taking the derivative requires a continuous function but images are discrete grids. Therefore, first and second derivatives are approximated by convolution kernels. Examples of Convolution filters are Sobel operator or Laplace filter. Sobel operators can detect changes in horizontal or vertical directions, which can be used later to find angle or magnitude, while Laplace filter finds the sum of second partial derivatives with respect to vertical and horizontal directions. Each of those filters use a kernel of predefined size and values to be convolved over the entire grid of the image pixels.

2.2.4 K-means clustering

K-means clustering assigns different pixels to different clusters based on similarities (in color). It is an unsupervised technique, which groups pixels together without predefining them. This algorithm works by defining k clusters with k centroids, one for each cluster, then assigning pixels with the minimum distance from each centroid to this cluster. Afterwards, new centroids are calculated based on the clusters generated. This process is done iteratively until centroids are fixed in place and the clusters are clearly defined.
The algorithm steps are as the following:

1) Let X = {x1,x2,x3,……..,xn} be the set of data points and V = {v1,v2,…….,vc} be the set of centers.
2) Randomly select ‘c’ cluster centers.
3) Calculate the distance between each data point and cluster centers.
4) Assign the data point to the cluster center whose distance from the cluster center is the minimum of all the cluster centers.
5) Recalculate the new cluster center using:

$$v_i = (1/c_i) \sum_{j=1}^{c_i} x_i$$

where, ‘ci’ represents the number of data points in i^{th} cluster.

6) Recalculate the distance between each data point and new obtained cluster centers.
7) If no data point was reassigned then stop, otherwise repeat from step 3.

2.3 Our conclusion

After investigating theoretically all of the mentioned options above, we came to the conclusion to focus on using CART classification at this stage and work on enhancing the accuracy of the output. At the same time, we are leaving some space to investigate other classifiers that are supported by GEE if they could provide higher accuracy in the end.

3. High level architecture of the software

3.1 Short description of the software

The software should be able to fetch the NAIP imagery for the NYC and should allow the user to select the rooftop pixels and non-rooftop pixels for training data/validation data. The Cart classifier in Google earth engine is used for multispectral classification of rooftops and vacant parcels. Due to uneven illumination of sun, reflectance values of features like vegetation, road, pavements are sometimes considered to be rooftop pixels and hence after classification through cart classifier there are so many pixels of vegetation, road and other non-rooftop objects that are identified as rooftops. These incorrectly assigned pixels should be discarded by using NDVI calculation and DEM of that location.

3.2 Flow chart of the software hierarchical design

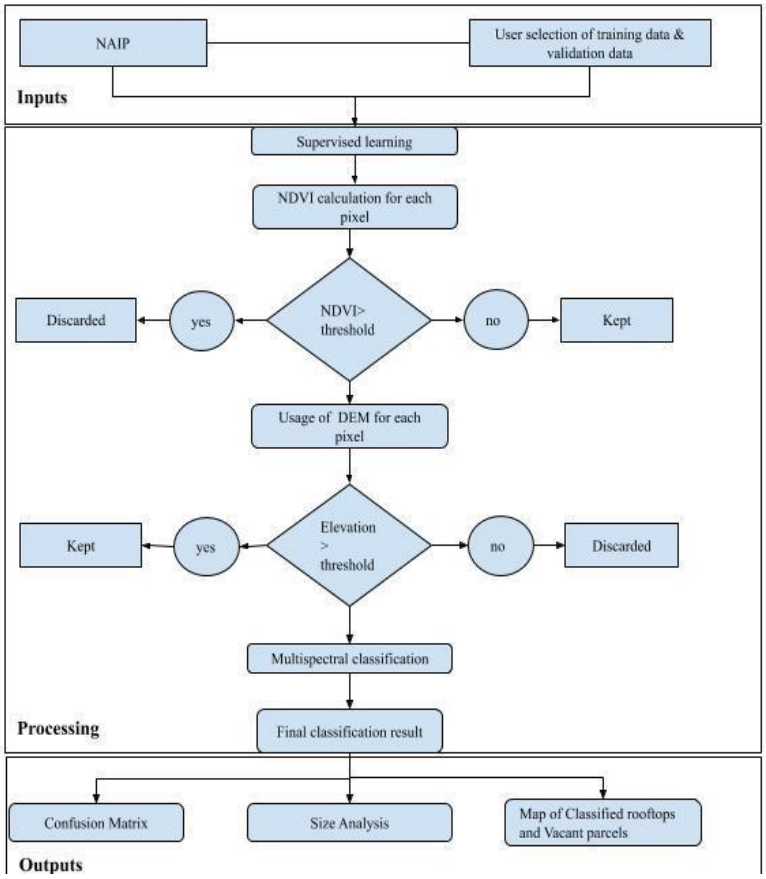

Fig 1. Higher architectural level of the software design

The above figure illustrates the software architectural design for detecting rooftops or vacant parcels and a flow diagram of the same. The software architecture provides a bigger picture of how the software looks and establishes clear connections between various steps in the software(from user inputs to output).

3.3 Required inputs

Following are the inputs required for classification of rooftops and vacant parcels.

- **NAIP imagery of the ROI**

Software requires the input of NAIP(National Agricultural Imagery Program) imagery for classification. NAIP imagery has a spatial resolution of 1m and it contains images in Red, Green, Blue and near infrared spectral regions. Imagery taken between 2017-01-01-2018-12-31 will be used. Google earth engine will fetch the required NAIP imagery which is pre-processed and ortho corrected.

- **User selection of rooftop and non-rooftop polygons**

The pixels for supervised learning should be selected by the user as training data and validation data for the classifier. A minimum of 20-30 polygons should be selected for each class (i.e rooftops and non-rooftops) on different locations in the rectangular tile that is overlaid on the GEE interactive map. The training data would have the same resolution as that provided by the input source, NAIP in this case.
For the training data polygons were used instead of individual pixels since it was noticeable that the overall accuracy of the classification was improved. Therefore, we decided to use polygon areas as the training data with 60 polygons for each training dataset. In the same way, validation data (different from training data) should be selected in the image for calculation of confusion matrix.

- **DEM of the ROI**

Digital elevation model of the desired location is required to differentiate between ground and non-ground features in the classification. The elevation model we found for the USA has a resolution of 0.33 arcseconds (~8m.) and it is called National Elevation Dataset (NED). It was integrated into GEE.

3.4 Overview of individual processing steps

3.4.1 Supervised Learning

The process of designing a classifier by using training samples from the classes of interest is referred to as supervised learning. The user selects manually a few areas that belong to the rooftop and pixels that are not rooftops and adds them to their respective classes. An interactive map in GEE helps the user to add these pixels to their respective training sets easily. The classifier is trained based on these training sets chosen by the user. The accuracy of the classification improves with more pixels in the training sets.

3.4.2 Discarding vegetation pixels in the classification

There are always some pixels that belong to vegetation misclassified as rooftops. A vegetation mask is introduced to avoid this misclassification. NDVI is calculated for every pixel in the image. If the NDVI value of the pixel is greater than the threshold, it is a vegetation pixel and it is discarded from the classification and included in non-rooftops.

NDVI value for each pixel is determined using the spectral values of pixel in near infrared and red and calculated as follows:

$$\text{NDVI} = (NIR - RED)/(NIR + RED)$$

The following code snippet explains how the NDVI calculation was integrated in our code.

```
// Compute the Normalized Difference Vegetation Index (NDVI).
var nir = image.select('N');
var red = image.select('R');
var ndvi = nir.subtract(red).divide(nir.add(red)).rename('NDVI');

// Display the result.
Map.centerObject(image, 9);
var ndviParams = {min: -1, max: 1, palette: ['blue', 'white', 'green']};
Map.addLayer(ndvi, ndviParams, 'NDVI image');
print(ndvi);

var NDVI = ndvi.lte(0.4)

var masked_image_ndvi = image.updateMask(NDVI);
Map.addLayer(masked_image_ndvi,{bands: ['R', 'G', 'B'],min:0, max: 255}, 'True colour masked image after NDVI');

print(masked_image_ndvi)
```

Fig 2. NDVI calculation in GEE

First, identify the red and near infrared pixel values to insert them in the NDVI equation. The upper and lower limits of the NDVI vary between 1 and -1 values (because that is the range for NDVI). After experimenting, the appropriate threshold was identified to be equal 0.4, which gave the most accurate results. Finally, vegetation pixels are identified in the image and then masked on to the input imagery. The results of before and after NDVI calculation are shown below. On the right image, vegetation is masked with light green polygons, and comparing the polygons to the true vegetation locations on the left image, the resulting polygons are accurate to a high degree. Moving forward from this step, the identification of rooftops can be safely done without interfering with the green areas.

Fig 3. True color masked image after NDVI

Fig 4. Before NDVI classification

3.4.3 Discarding ground object pixels in the classification.

The features like roads, pavements, and other grounded objects are misclassified because of their similar reflectance values as rooftops.This misclassification can be avoided by introducing DEM for the image location and calculating elevation for each pixel by interpolation. If the elevation of a pixel is less than a threshold, it is considered to be a ground pixel and can be discarded from the rooftop class. Implementation of DEM is shown below.

```
var dataset = ee.Image('USGS/NED');
var elevation = dataset.select('elevation');
var elevationVis = {
  min: 0.0,
 max: 4000.0,
  gamma: 3,
};
Map.setCenter(-73.9928817565918,40.72760974624499);
Map.addLayer(elevation, elevationVis, 'Elevation');

var ELEVATION = elevation.lte(10)
print(ELEVATION)

var masked_image_DEM = masked_image_ndvi.updateMask(ELEVATION);
Map.addLayer(masked_image_DEM,{bands: ['R', 'G', 'B'],min:0, max: 255}, 'True colour masked image after DEM');

print(masked_image_DEM)
```

Fig 5. NED model integration into GEE code

Unfortunately, the resolution of the model was not good enough to distinguish roads and ground pixels as it mainly highlights the higher elevation terrain such as mountains and so on. Therefore, the resolution was not suitable for our purpose and it did not help much in improving the accuracy of rooftops detection after excluding vegetation areas because many rooftop pixels were misclassified as ground pixels.

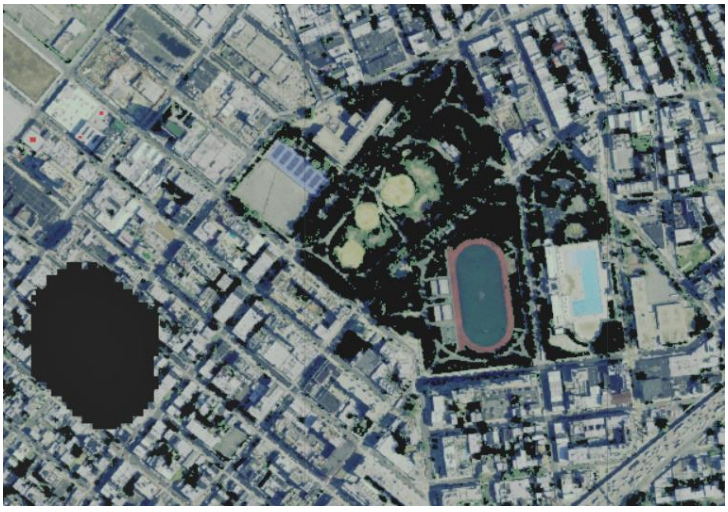

Fig 6. True color masked image after DEM

3.4.4 Multispectral classification

NAIP imagery contains four bands in Blue, Green, Red, and Near-infrared spectral regions. Each pixel in the training set contains the value of four spectral bands using this a 4-dimensional feature space is created for rooftops and non-rooftop classes. The cart classifier in GEE creates decisive boundaries between rooftop and non-rooftop classes. Every pixel belonging in the image is decided based on the boundary created by the classifier.

3.5 Outputs

The output of the classification is then subjected to the test of its accuracy by determining its confusion matrix to calculate the user's accuracy and producer's accuracy. The classification of rooftops and vacant parcels can be located easily on the interactive map provided by GEE. These features are indicated in different legends thereby identification and classification are noticeable and understandable.

3.5.1 Confusion matrix

A confusion matrix is a specific table layout that allows quantization of the performance of an algorithm, typically for supervised learning. User selected validation data were used to generate the confusion matrix (Table 1). With the confusion matrix, we can compute the overall accuracy, producer's accuracy, user's accuracy and kappa coefficient.

		Predicted class	
		P	*N*
Actual class	*P*	2 (True Positive)	0 (False Negative)
	N	2 (False Positive)	13 (True Negative)

Table 1. Confusion Matrix

In this confusion matrix, of the 2 actual rooftops, the system correctly predicted them as rooftops, and of the 15 non-rooftop features, it predicted that 2 were rooftops. The correct predictions are the true positive and true negative, which are 15 features. The overall accuracy of the system is the correct prediction out of all the classes, which is 88% (Equation 1).

$$Overall\ accuracy\ = \frac{TP\ +TN}{TP+TN+FP+FN} = \frac{2+13}{2+13+2+0} = \frac{15}{17} \simeq 0.88$$

Equation 1. Overall accuracy calculation from confusion matrix

The producer's accuracy is the correctly classified rooftops out of the total predicted number of rooftops, which is 50%. The user's accuracy is the correctly classified rooftops out of total actual rooftop class, which is 100%.

Kappa coefficient is also used to evaluate the accuracy of the classification. It evaluates how well the classification performed as compared to just random assigning values. It ranges from -1 to 1. A value of 0 indicates that the classification is no better than a random classification. A negative number indicates the classification is significantly worse than random. A value close to 1 indicates that the classification is significantly better than random. The kappa coefficient in our case is 0.6 (Equation 2-5), which is significantly better than random.

$$p_P = \frac{2+0}{2+0+2+13} \cdot \frac{2+2}{2+13+2+0} = 0.117 \times 0.235 \simeq 0.03$$
$$p_N = \frac{2+13}{2+0+2+13} \cdot \frac{0+13}{2+13+2+0} = 0.882 \times 0.765 \simeq 0.67$$
$$p_e = 0.03 + 0.67 = 0.70$$
$$\kappa = \frac{overall\ accuracy - p_e}{1 - p_e} = \frac{0.88 - 0.70}{1 - 0.70} = 0.6$$

Equation 2-5. The calculation of kappa coefficient

3.5.2 Map of classified rooftops and vacant parcels

After classification, a classification layer is placed on top of the true color image so that the user can visualize the features easily. Below is a classification layer shown:

Fig 7. True color image

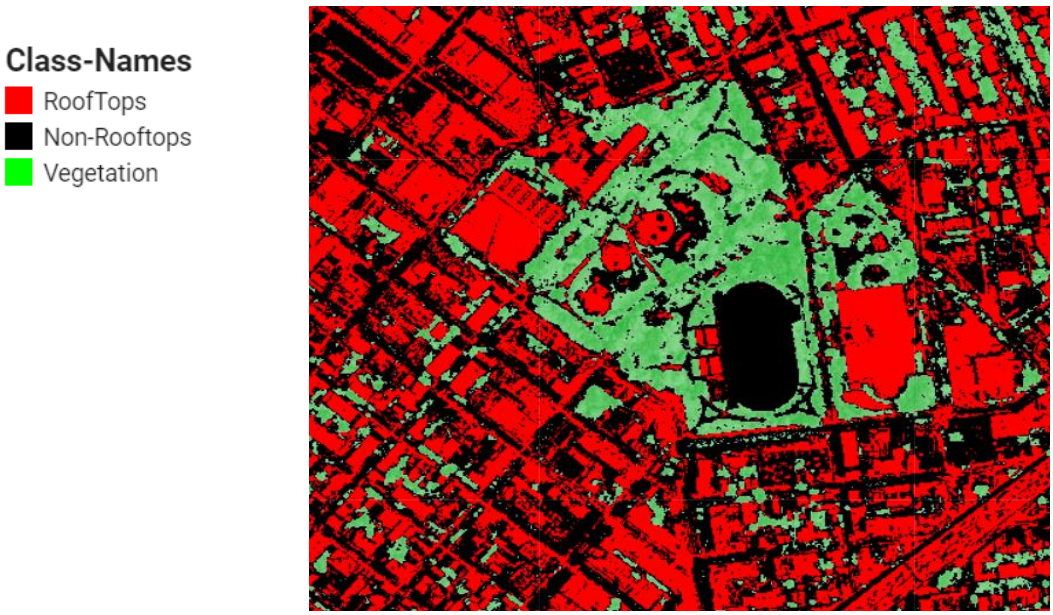

Fig 8. Classification results

4. Graphical user interface (GUI)

The GUI is based on the Google platform. We use Google Map API to realize our interactive map and display our result. The link of our App is:
https://susanshende.users.earthengine.app/view/urban-farm-app

The main components of GUI are as follows:

- ❖ Training Data Selecting Module
- ❖ Image Selecting Module
- ❖ Map control Module
- ❖ Classification Module
- ❖ Result Display Module

4.1 Training Data Selecting Module

The training data selection allows users to select training pixels (or areas in case of using other classifiers). When the user clicks the button “Select Training Data”, then the drawing tools will

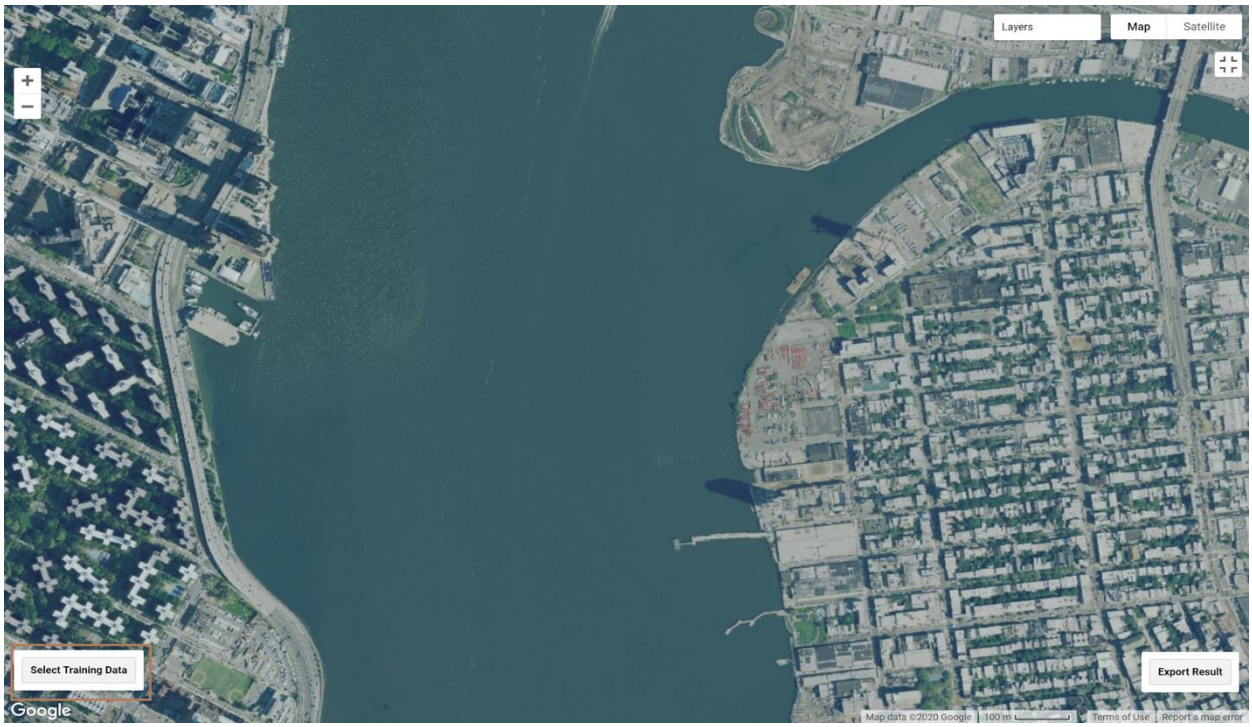

Fig 9. Training data selection visualization

Then users can select the drawing tool in the left bar and then for instance draw polygons or points on the map, which define the areas for training. Besides, user can defined the classes of training classes and validation data.

Fig 10. Tools available to the user while selecting training data

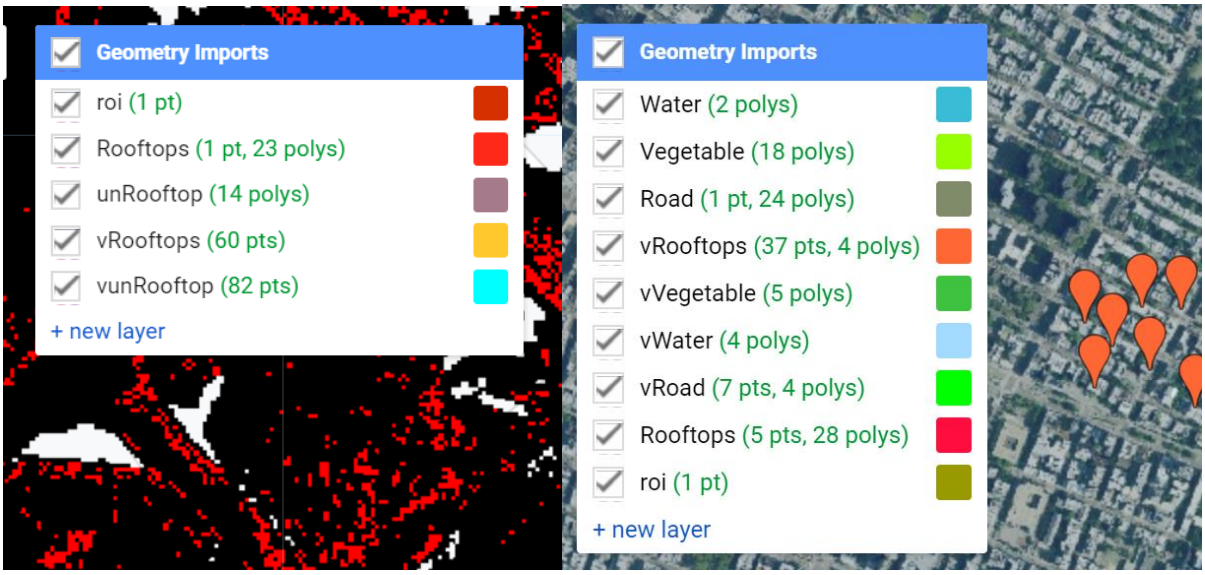

Fig 11. Different training classes and validation data

4.2 Image Selecting Module

Users can select the data source in the GUI, the control module located in the upper right. For example, here we select the NAIP data and the map layer has the resolution with 1 meter.

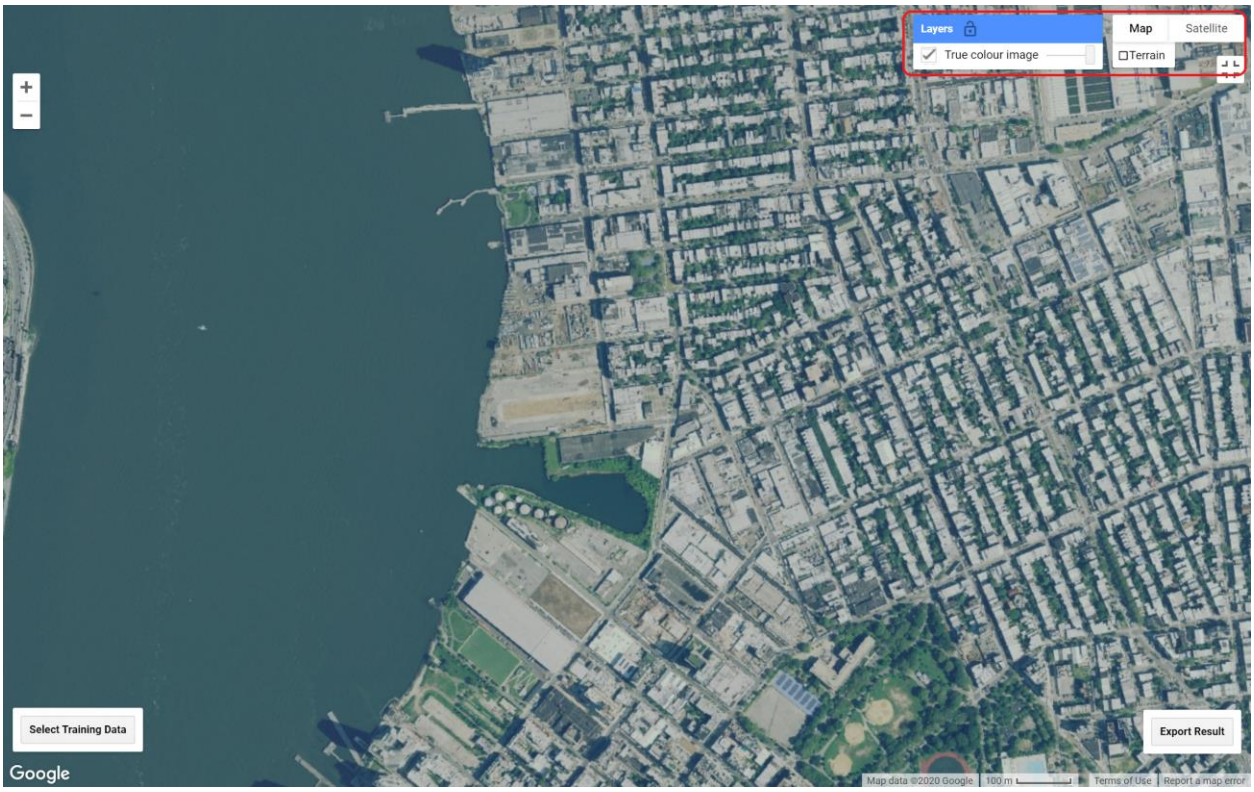

Fig 12. Layers available for the user (top right corner)

4.3 Map control Module

The map control module includes zoom in and zoom out, as well as full screen. The scale range can be selected from 1:200 to 1:10000000.

4.4 Classification Module

After the selection of the training area, users can click the “Start training” button. The selected area will be used to conduct the classification process with the CART algorithm. The progress bar of the classification time will be displayed on the screen as well.

4.5 Result Display Module

The result will be shown on the map. In addition, on the left side, we can get the confusion matrix of the result with the overall accuracy as well as the total rooftop area. Based on the area, we can estimate the total grain output.

Fig 13. Final result visualization of rooftop detection

4.6 Result Export

On the right-bottom, there is a button named “Export Result”. When we click it, You can export images from Earth Engine in GeoTIFF or TFRecord format to your Drive account.

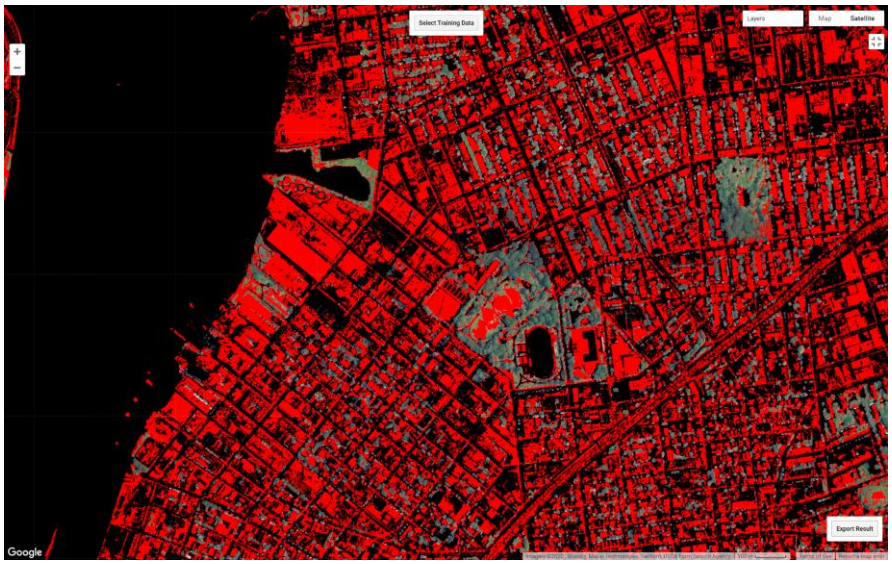

Fig 14. Export Result

When we finish the save, the system will pop up a dialog to remind you that your result is already saved.

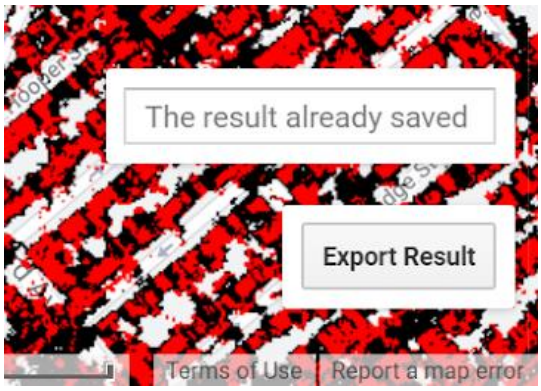

Fig 15. Dialog

4.7 Support Devices

Our GUI can be used in many kinds of devices, which also support browsers. For example, laptop, PAD, Smartphone and so on.

5. Milestones

5.1 Input resolution decision

We selected and compared the sentinel-2 and NAIP images as the input data, in which the resolution of sentinel-2 we got can get access is not enough (60 m). Thus, we decided on NAIP imagery as the input of our system, which has a spatial resolution of 1m and is taken between 2017-01-01-2018-12-31.

5.2 Classifier selection

We conducted a Cart classification and Convolution Neural Network (CNN). The result with Cart can be more precise and the edges of rooftops are better detected, and the Cart algorithm is encapsulated in google earth engine and easy to invoke. Thus, we currently chose the Cart method as the classifier of the system.

5.3 First test run

5.3.1 Results from Sentinel-2 data

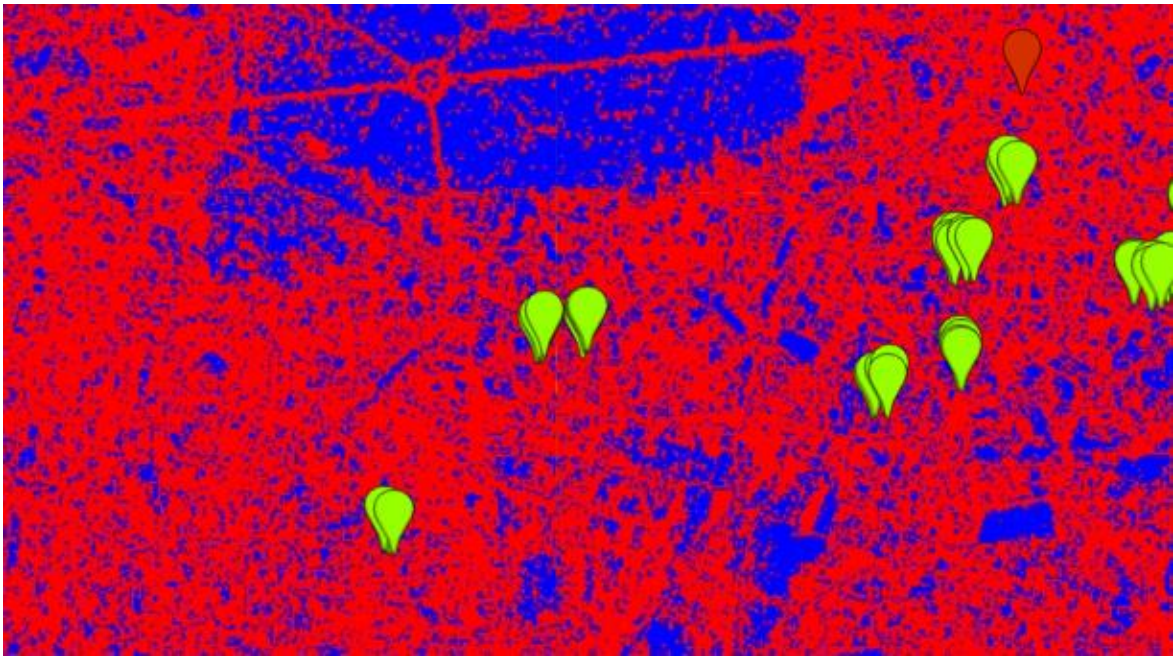

Fig 16. Result of rooftop detection after using Sentinel-2 data (red is rooftops)

Using Sentinel-2 image as our input resulted in a very poor resolution (10m - 60m) for our purpose and it is not possible to distinguish rooftops from non-rooftops as lots of other pixels were misclassified. The red regions represent rooftops and blue represents non rooftops. Therefore, we looked into aerial imagery as the input source.

5.3.2 Results from NAIP data

Fig 17. Resulted map after using NAIP as the input source (gray is rooftops)

In the figure above, grey regions represent rooftops and black is non rooftop. Compared to Sentinel-2 results, there is a noticeable improvement in the outcome because the input source resolution is significantly higher (1m) and by using almost the same amount of training pixels in both cases. However, some of the vegetation pixels are misclassified as rooftops and in the following steps, those pixels are masked using NDVI.

5.4 Improved test run

5.4.1 NDVI results

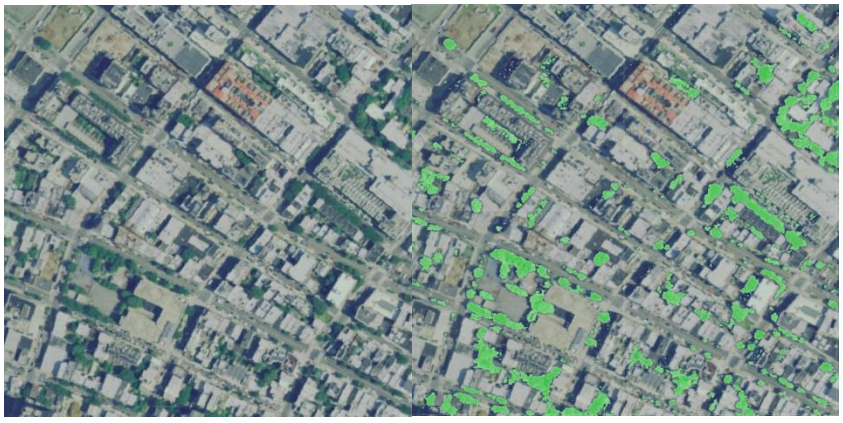

Fig 18. NDVI results

After calculating the NDVI for the entire city, vegetation pixels are masked accordingly. Proceeding from this step, rooftops can be detected.

5.4.2 Classification results

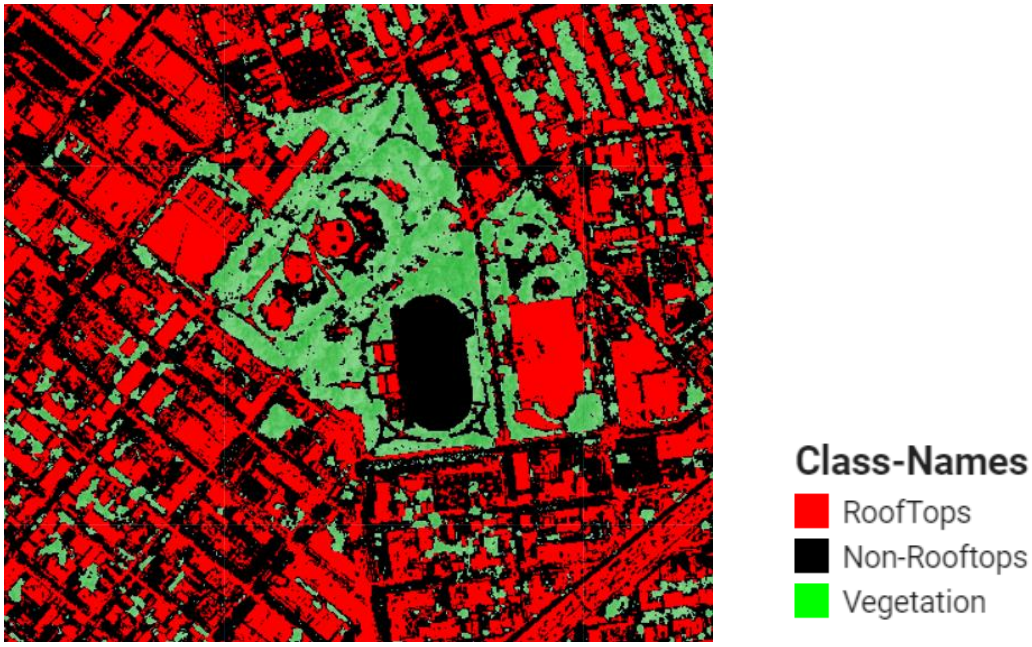

Fig 19. Classification results

Compared to the resulting classification from the first run, the above results are improved dramatically after masking vegetation pixels. Moreover, a legend specifying color-coded classes is included on the right.

5.4.3 Confusion matrix generation

The accuracy of the classification is determined by the confusion matrix. We have achieved an overall accuracy of 88%.

6. Future steps

6.1 Rooftop selection criteria for urban farming

After classification, the image is divided into rooftop area and non-rooftop area. However, not all rooftop areas are suitable for urban farming. In general, only those roofs with sufficient area, flat slope, low winds and adequate sunlight are suitable for growing crops. Based on the available data source, we plan to conduct size analysis, height analysis and slope analysis for detecting potential urban farming rooftops.

6.1.1 Morphological operations

The classified map is a binary image consisting of 'rooftop' pixels and 'non-rooftop' pixels. Due to the image resolution limitation and classification error, the detected rooftop pixels are distributed sparsely, and the corresponding aggregated rooftop areas are irregularly shaped. In order to conduct further analysis, morphological operators shall be applied to connect neighboring pixels and smooth regions.

Among the binary morphological operators, the operator 'Opening' is suitable for our case, which is useful for eliminating small objects, separating objects at slender points, and smoothing the boundaries of larger objects without significantly changing their area.

In mathematical morphology, opening is the dilation of the erosion of a set A by a structuring element B:

$$A \circ B = (A \ominus B) \oplus B$$

Where$\ominus$ and $\oplus$ denote erosion and dilation, respectively.

The size of the structuring element B can be set as 3 or 5, and the opening operators can be applied multiple times until there is no change for the image after the operation. Similarly, the operator 'Closing' can be also employed to fill the small holes for the rooftop areas, as a supplement for Opening.

6.1.2 Size analysis

With the morphological adjustment, the rooftop pixels are aggregated more regularly as regutanglars. The next step is to do size analysis to make sure each rooftop region has sufficient area to grow crops. The size calculation for each region can be realized by the tool OpenCV (Open Source Computer Vision Library) automatically. With a selected threshold (minimum roof surface area), we can further filter out the rooftops with suitable planting areas.

6.1.3 Height analysis

Typically, rooftop conditions above 100ft (30m) are assumed as being less hospitable to plants due to high winds, and there are logistical and safety concerns with access for people and supplies. Therefore we shall remove those rooftops with large height. This requires Digital Surface Model (DSM) data, which is generated by the Lidar point cloud. The height for each pixel can be derived by subtracting the corresponding DSM value with DEM value. As for a rooftop region, an average height of all the pixels in this region shall be less than 30 meters.

6.1.4 Slope analysis

Together with the maximum height limitation, the surface slope of rooftops for urban farming should be smaller than 5 degrees to prevent water and soil erosion. The calculation for Slope also

requires the DSM data and can be performed using the *Surface slope tool* of ArcMap spatial analyst. The DSM layer is used as an input and degree (pitch) is chosen as output slope layer unit. Building polygons with average slopes of <5° will be reclassified as flat roofs using the Reclassify tool.

6.2 DEM accuracy

The accuracy of our classifier can be further improved by including DEM to discard ground pixels. However, finding a high-resolution DEM is a challenging task and hence, finding DEM with suitable resolution is vital to improve the accuracy of the classifier. For example, this DEM of the Netherlands with an accuracy of 0.5m. could be ideal for our purpose.

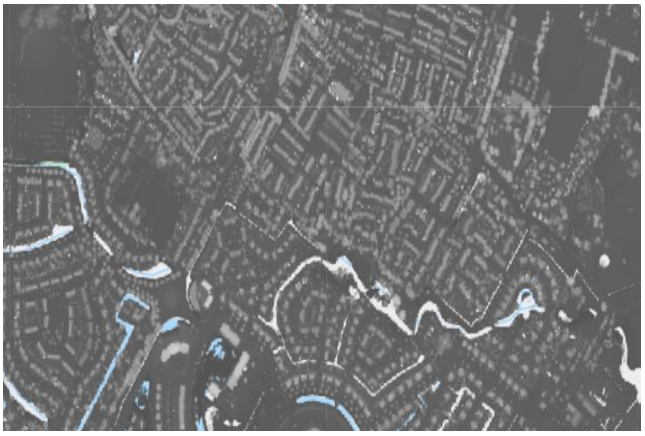

Fig 20. Netherlands DEM example

END OF THE DOCUMENT