# Crop Image Analysis Project 2020

Resources and References

---

**Important Literature:**

1. Photogrammetry-Wikipedia

2. A Photogrammetry Software as a Tool for Precision Agriculture: A Case Study

3. A Review on UAV-Based Applications for Precision Agriculture

4. Hyperspectral Classification of Plants: A Review of Waveband Selection Generalisability

5. Identifying Species and Monitoring Understorey from UAS-Derived Data: A Literature Review and Future Directions

6. Hyperspectral Image Classification

7. MULTISPECTRAL IMAGING, IMAGE-PROCESSING AND CLASSIFICATION FOR AGRICULTURE

## Useful Videos:

1. [Hyperspectral Image Classification - Philip Sellars]
2. [L2 Hyperspectral Image Classification]

## Libraries (Open Source Options):

1. [Micasense - GitHub]
2. [OpenDroneMap - GitHub]

## Images:

**Table 2.**
Comparison of results between the D-CNN and SVM using two data sets.

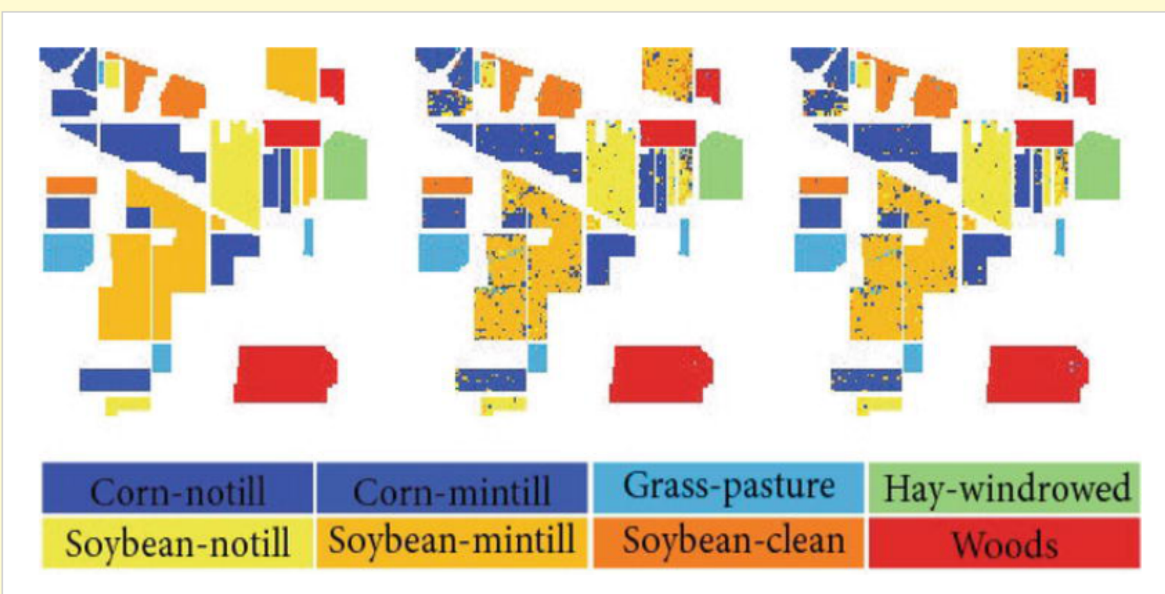
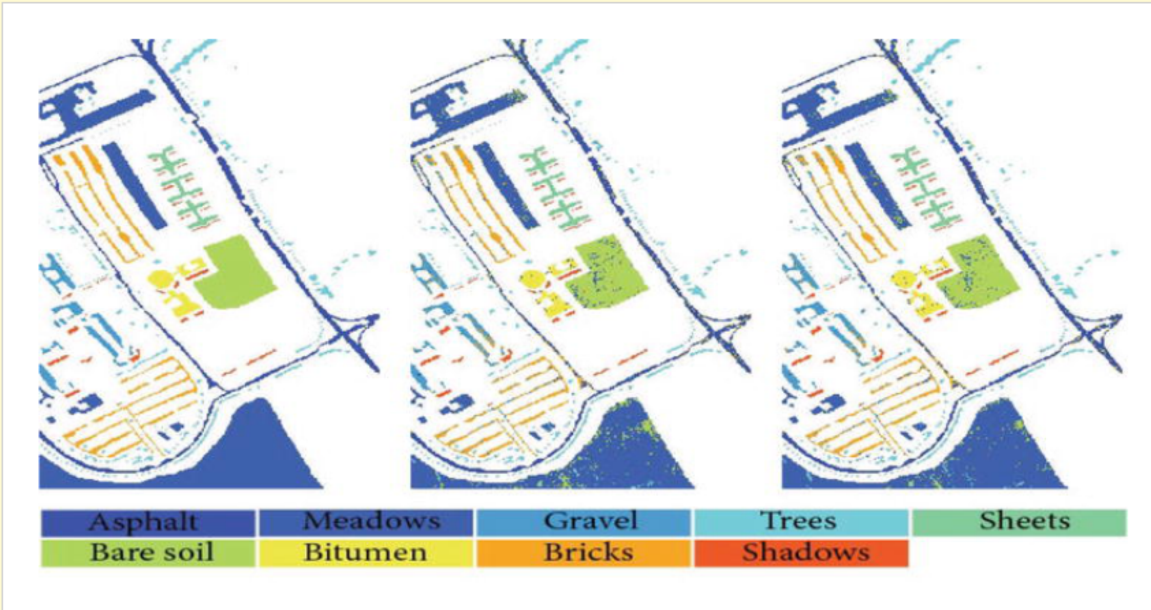


**Figure 8.**

RGB composition maps resulting from classification for the Indian Pines data set. From left to right: ground truth, SVM, and D-CNN.



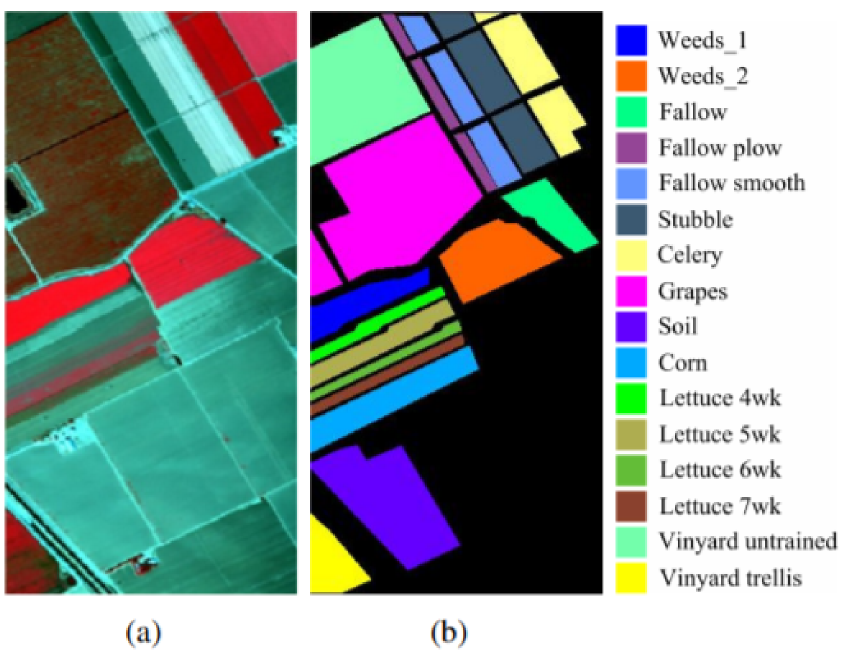
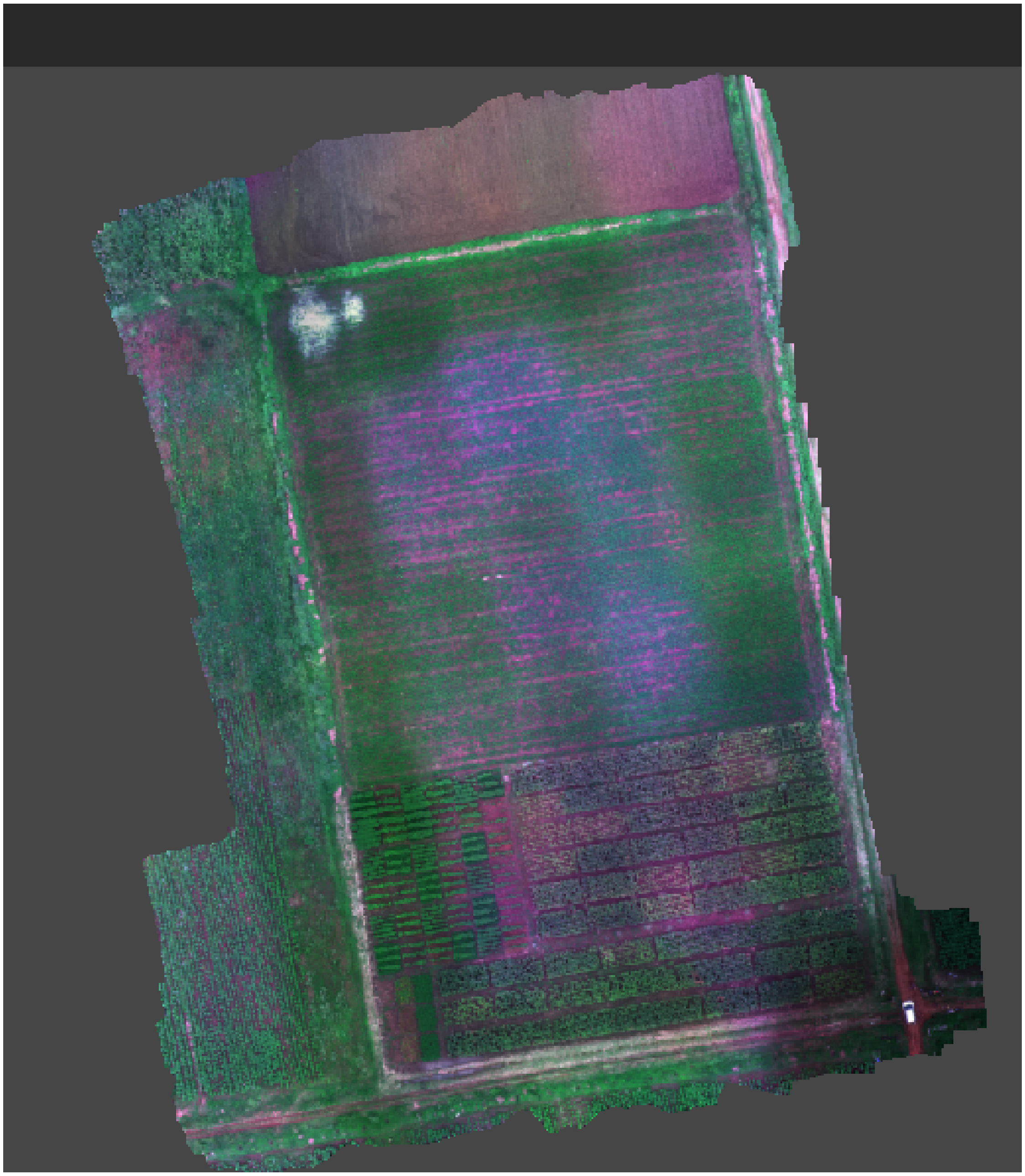




Fig. 15. The Salinas data set. (a) The three-band false color composite. (b) Ground reference data and color code.

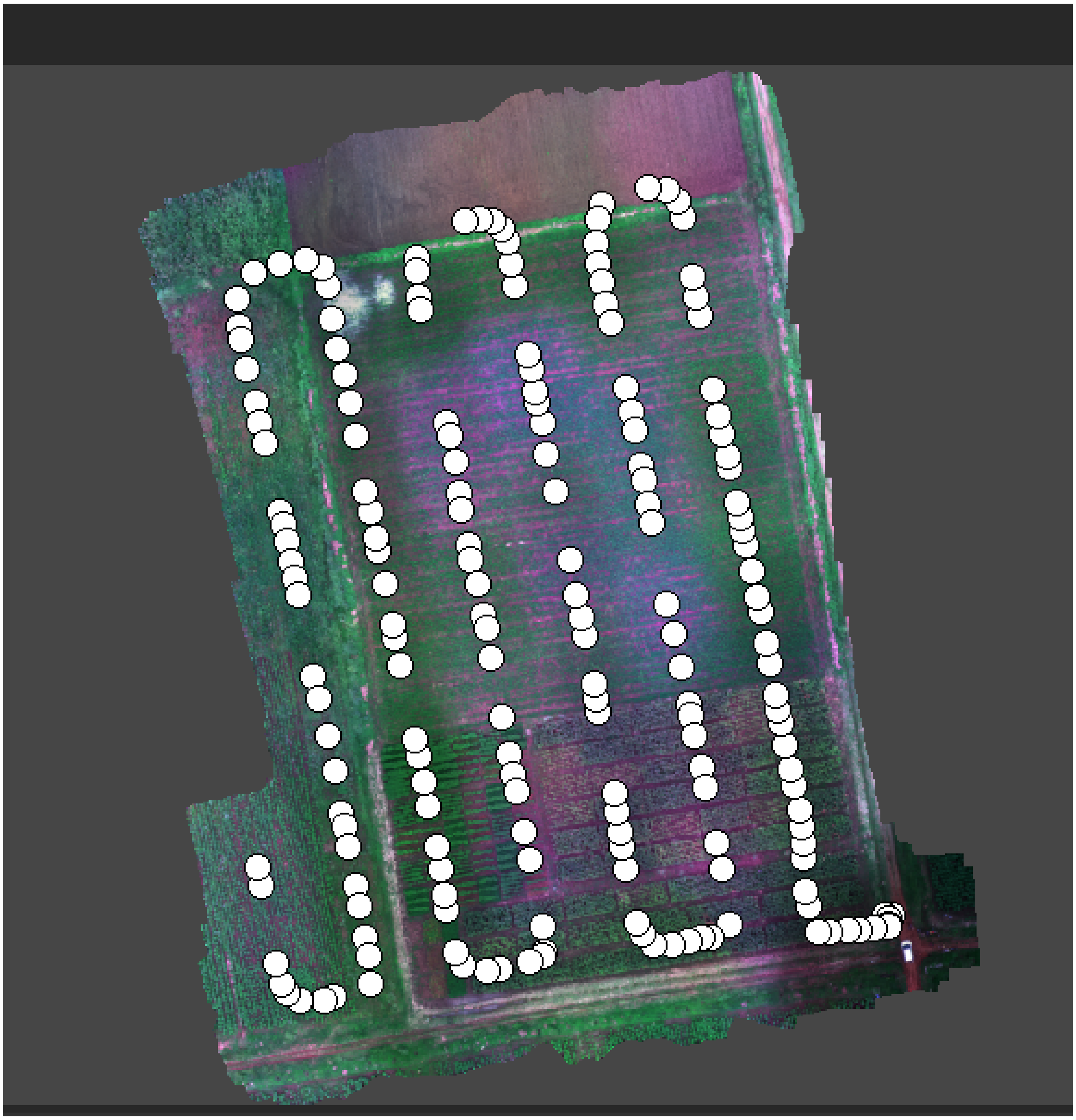
A 6-shot test orthomosaic obtained after ‘stitching’ 30 images from 6 different positions in Agisoft Metashape Professional. Note that 1 shot (1 camera) consists of 5 images corresponding to 5 different bands.

Screenshot of a 190-shot orthomosaic from the provided MicaSense RedEdge Dataset (Medium accuracy, medium quality dense cloud, high face count mesh, default texture settings, default ortho mapping settings) in Agisoft with RGB bands aligned.

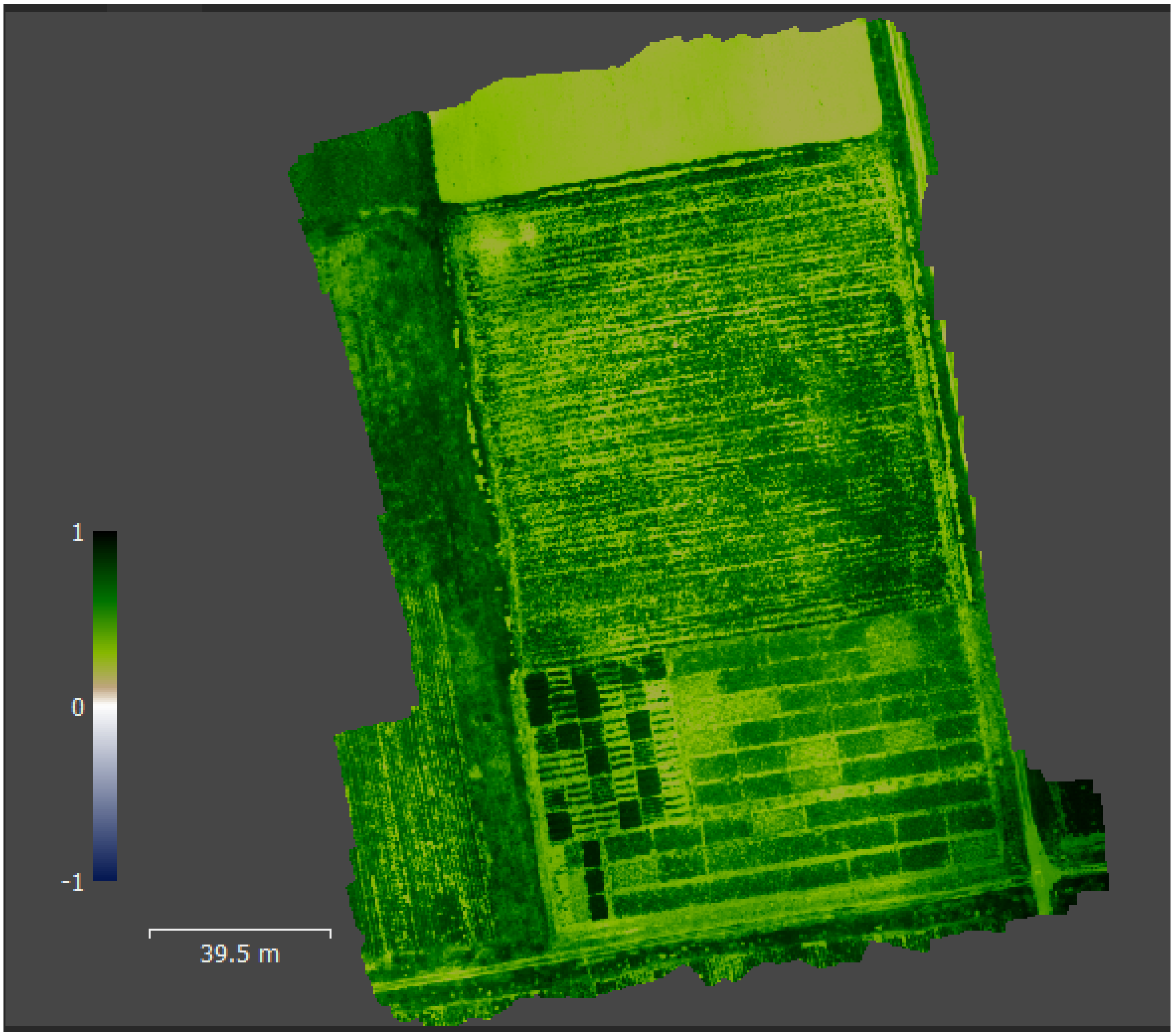


The quality of the above image is not representative of the original quality, as this is a screenshot. The original image is larger, with dimensions 10025 x 12333.

The capture points of the drone:



Calculating Normalized Difference Vegetation Index (NDVI) on the captured orthomosaic:




$$NDVI = (NIR - RED)/(NIR+RED)$$

where NIR and RED stand for the spectral reflectance values in the NIR (Near Infrared) and Red bands respectively. It lies in -1 to 1, 1 representing healthy and dense vegetation. The darker shades of green in the above image show thicker or denser vegetative cover. NDVI can also be normalized to have values between 0 and 1. It is useful in determining plant health, density of vegetation, forest cover, etc.

Some other [vegetation indices] can be used as well like VARI (Visible Atmospherically Resistant Index).

VARI = (GREEN - RED)/(GREEN+RED-BLUE)

---

**Experimenting with MicaSense:**

Each MicaSense camera model has some defined reflectance value for each band, which can be calibrated using the calibration panel provided along with the camera.

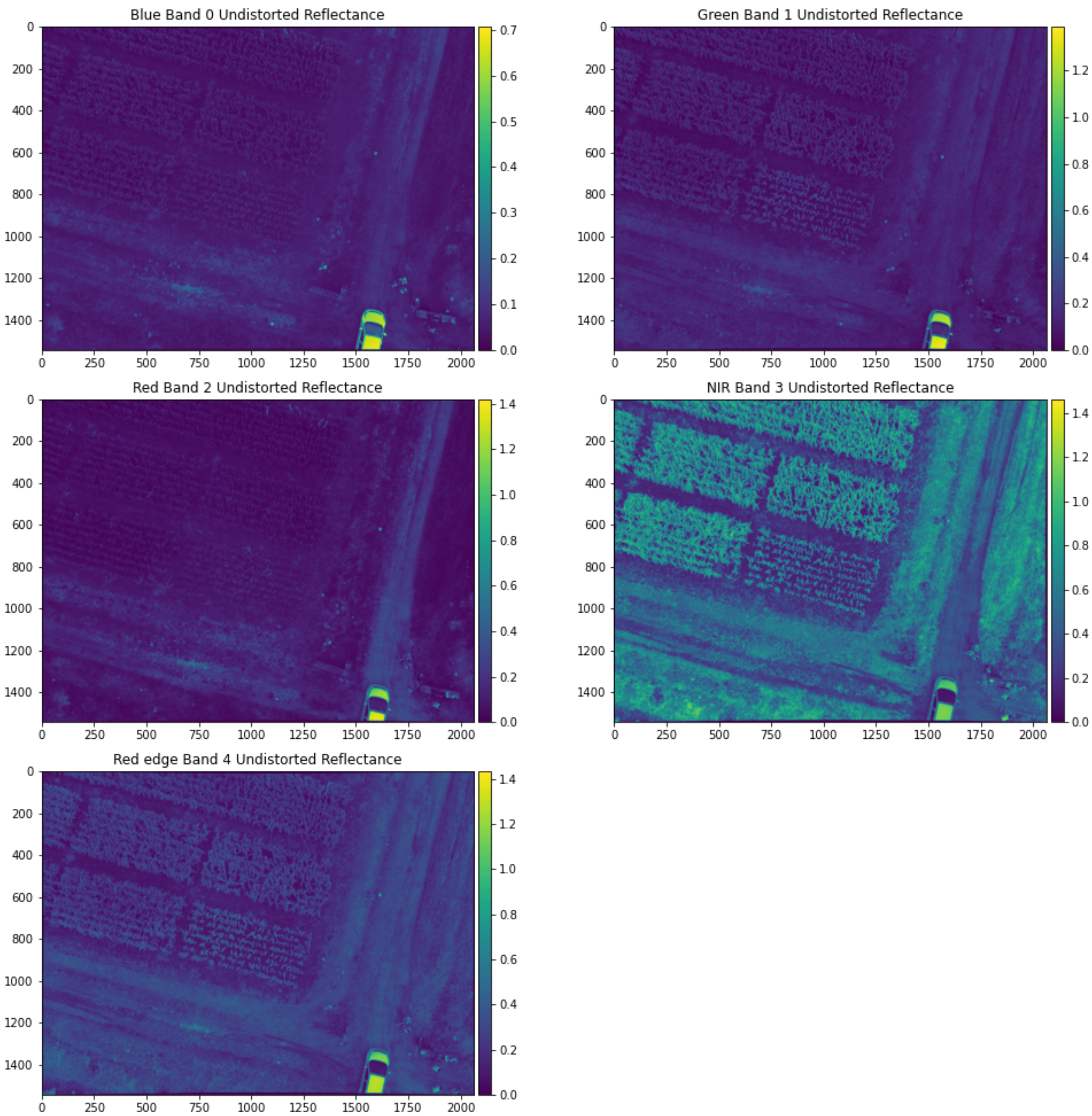

The shiny square panel made of some special reflective material is useful in calculating spectral reflectance values across various bands. The values are also printed on the panel. In this case the values are:

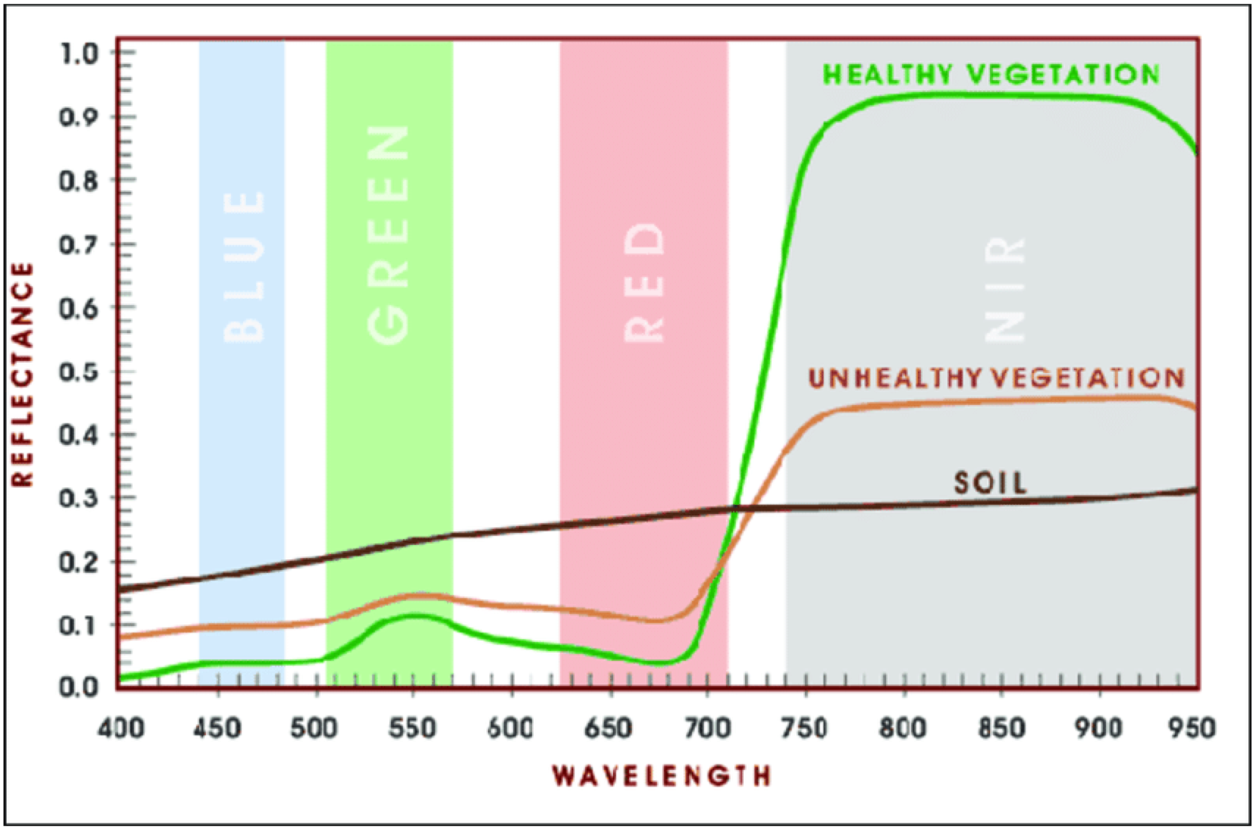
panel_calibration = [
"Blue": 0.513,
"Green": 0.512,

"Red": 0.51,
"Red edge": 0.51,
"NIR": 0.508
]

These values are helpful in radiometric calibration.

An example image displaying the reflectance across various bands:


Blue Band 0 Undistorted Reflectance
Green Band 1 Undistorted Reflectance
Red Band 2 Undistorted Reflectance
NIR Band 3 Undistorted Reflectance
Red edge Band 4 Undistorted Reflectance

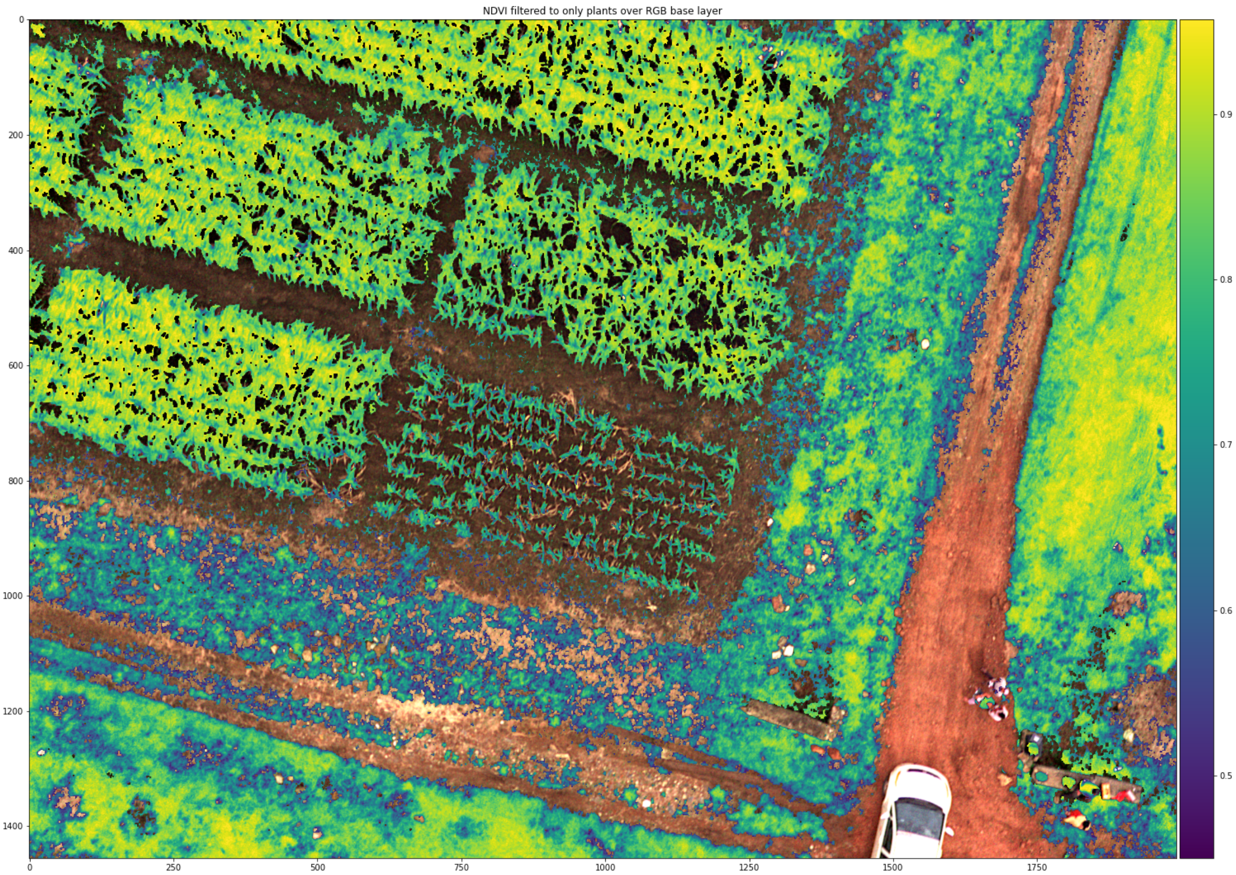



There is a sharp increase in reflectance as we move to NIR from red for healthy vegetation. Hence in the reflectance maps provided above, the vegetative region is bright in the NIR band image.

Aligning the RGB bands in the image:


Applying gamma correction and Gaussian blur to enhance:

Calculating NDVI and transforming raster:




The pixels having less than a threshold value (0.2) of reflectance (eg: soil, car, water, etc.) are excluded from NDVI transformation. The greener areas are regions having healthy vegetation.

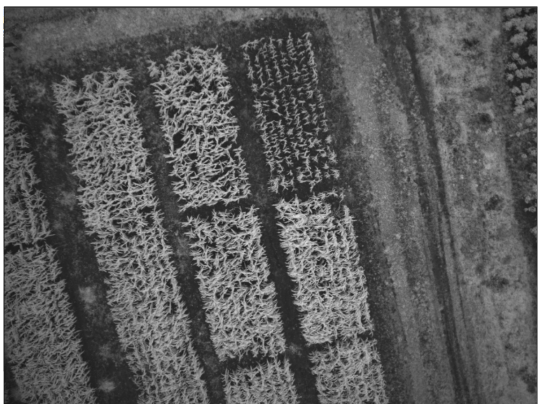
Note: The above transformations using MicaSense were applied only on one single capture (i.e 5 different bands of a single shot).

**Basic 3 Crop Classification using NIR Images:**

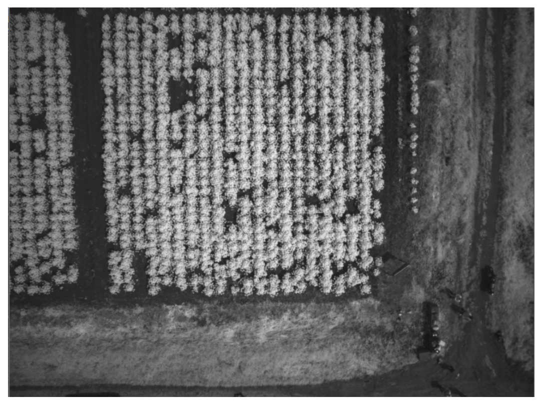
---

We decided to perform multiclass classification on 3 crops: Maize, Cotton and Wheat. We only picked NIR band drone images for each of these crops for experimentation. As these are drone images, multiple crops might be present in the same image. So, an image would be classified depending on the crop which appears in majority in that image.

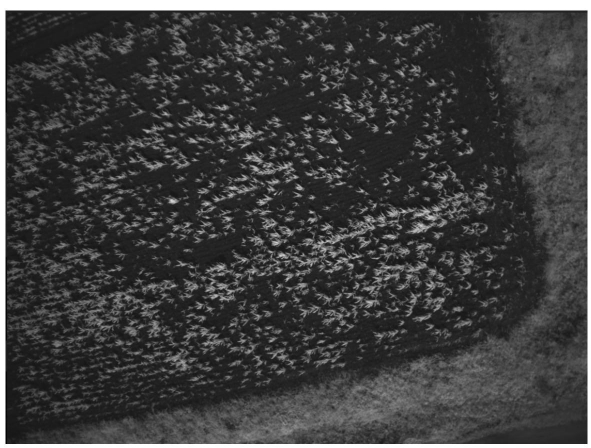
Sample images:



Maize(0)



Cotton(1)



Wheat(2)

We used 114 Maize images, 47 Cotton images, and 130 Wheat images overall.

Training: 85 Maize, 39 Cotton, 94 Wheat

Testing: 29 Maize, 8 Cotton, 36 Wheat

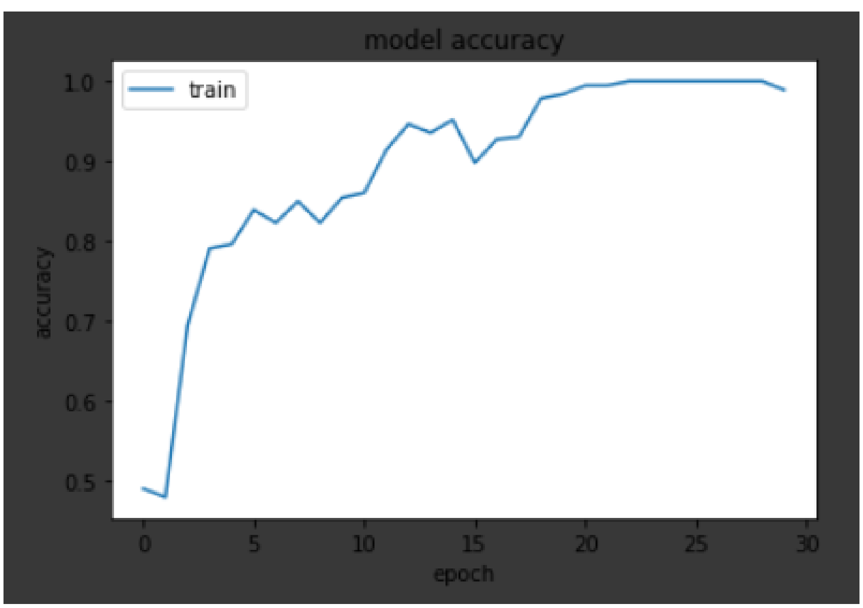
The image resolution is 2056 x 1544. For rescaling, each pixel's digital number value was divided by 50000, as the values of the pixels are different from RGB values.

A basic CNN was used for classification. It was done using Keras.

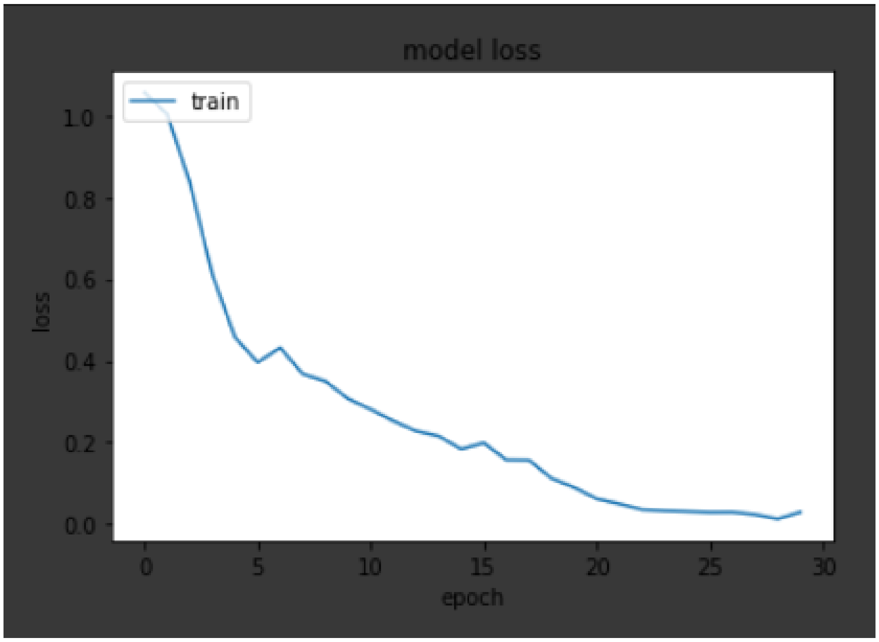
```
model = models.Sequential([
    layers.Conv2D(16, (3,3), strides=(2,2), padding='same', activation='relu', input_shape=(2064,1544,1)),
    layers.MaxPooling2D((3,3), padding='same' ),
    layers.Conv2D(32, (3,3),strides=(2,2),padding='same',  activation='relu'),
    layers.MaxPooling2D((3,3),padding='same'),
    layers.Conv2D(64, (3,3), padding='same',  strides=(2,2),activation='relu'),
    layers.Flatten(),
    layers.Dense(16, activation='relu'),
    layers.Dense(3, activation='softmax')
])
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 1032, 772, 16)     160
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 344, 258, 16)      0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 172, 129, 32)      4640
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 58, 43, 32)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 29, 22, 64)        18496
_________________________________________________________________
flatten_3 (Flatten)          (None, 40832)             0
_________________________________________________________________
dense_6 (Dense)              (None, 16)                653328
_________________________________________________________________
dense_7 (Dense)              (None, 3)                 51
=================================================================
Total params: 676,675
Trainable params: 676,675
Non-trainable params: 0
_________________________________________________________________
```

After 30 epochs, training accuracy was found to be 98.9%.

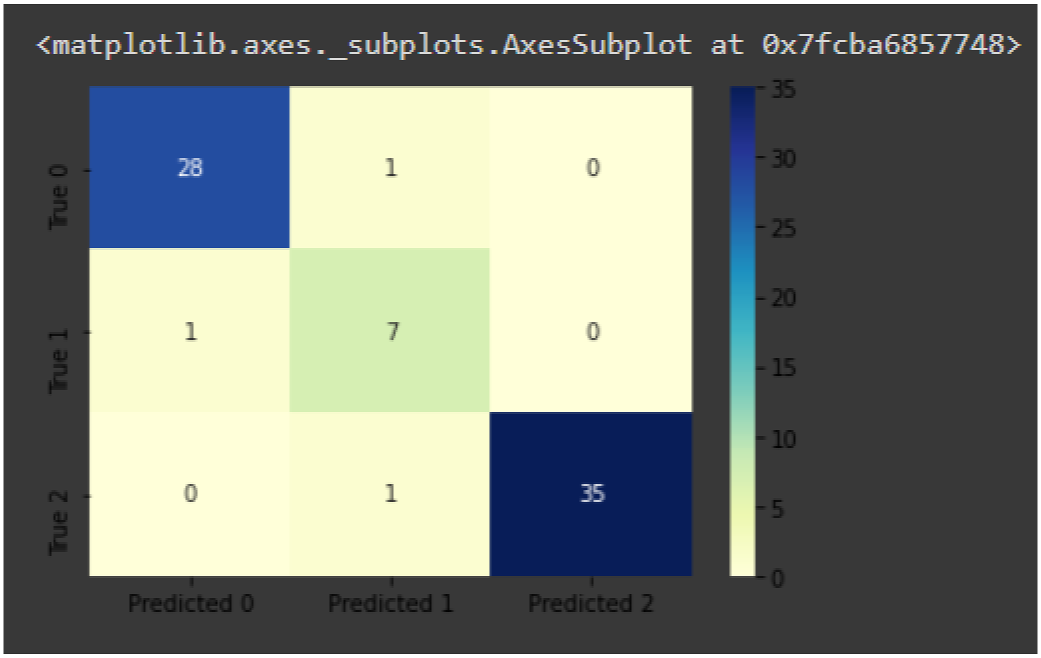


Model accuracy with epochs




Model loss with epochs

The test accuracy was 95.8%. The confusion matrix is given below:




Out of 73, 3 images were incorrectly classified.

Here is the colab notebook.

---

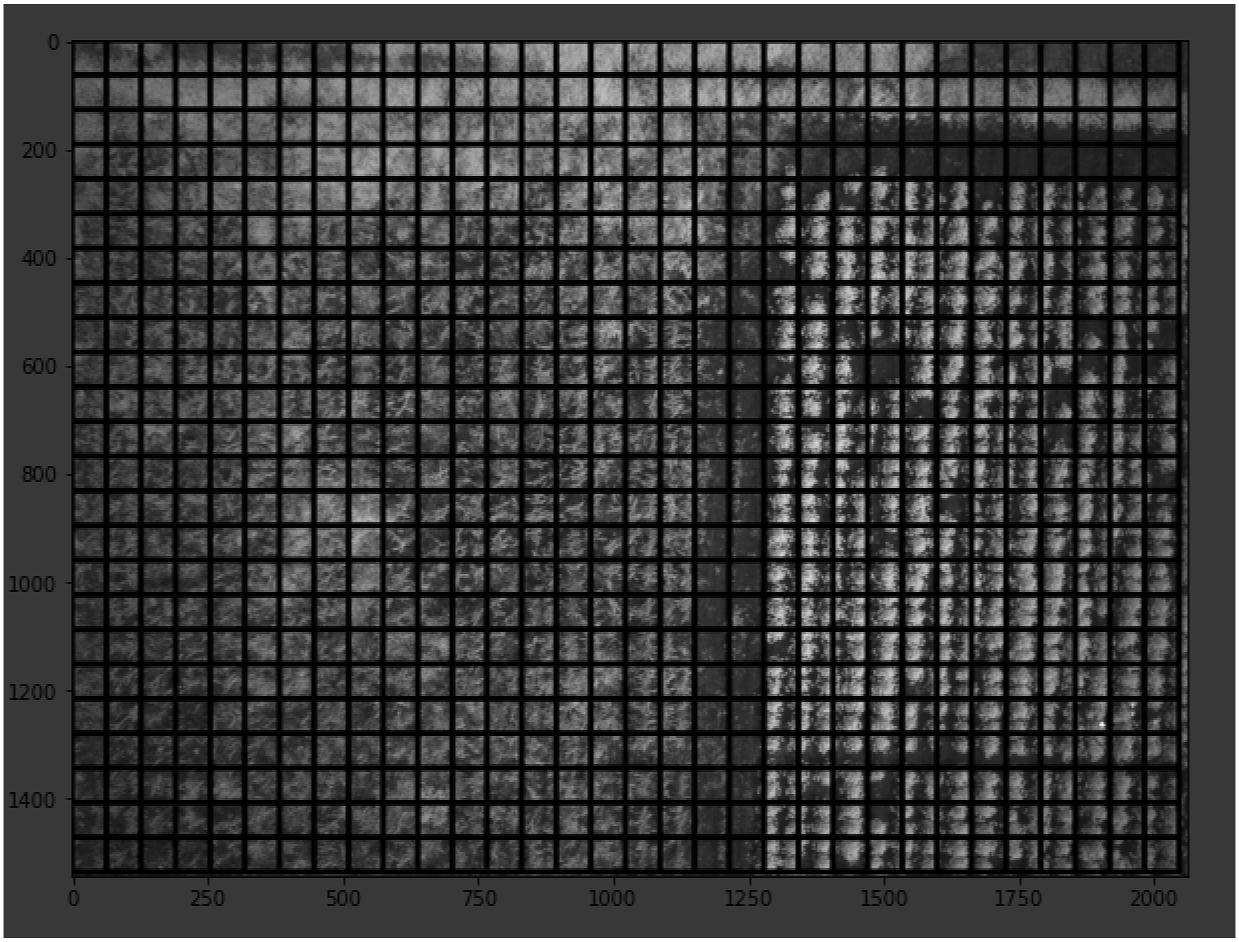
**Image Segmentation using NIR band:**

The previous classification was performed on original 1544x2064 raw NIR images based on the majority of crop cover. This might not be practically useful. So, we decided to experiment by implementing crop classification on a finer scale, i.e dividing the image into small blocks of 64 x 64 pixels, then classifying each such block/window. First we experimented only using NIR images.

Data:

Training: 872 Cotton, 939 Maize, 771 Wheat
Testing:  154 Cotton, 166 Maize, 137 Wheat

Sample image:

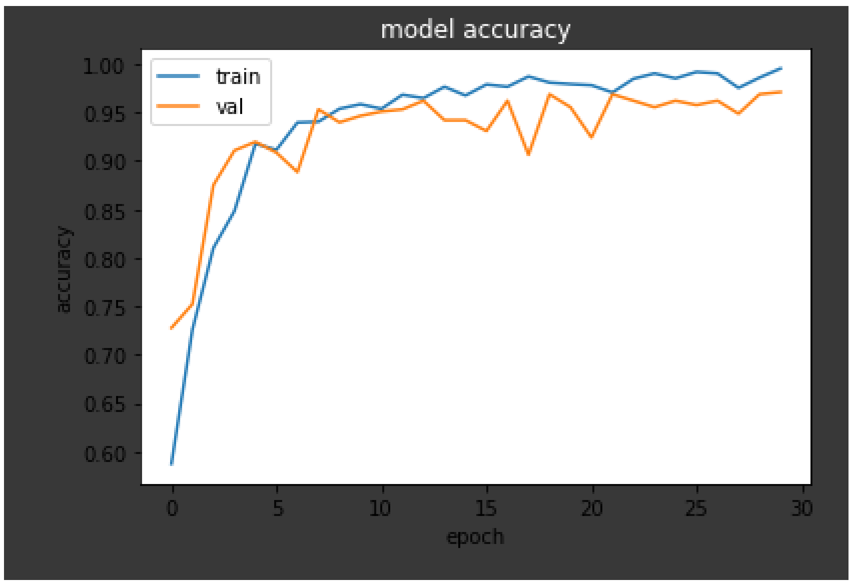


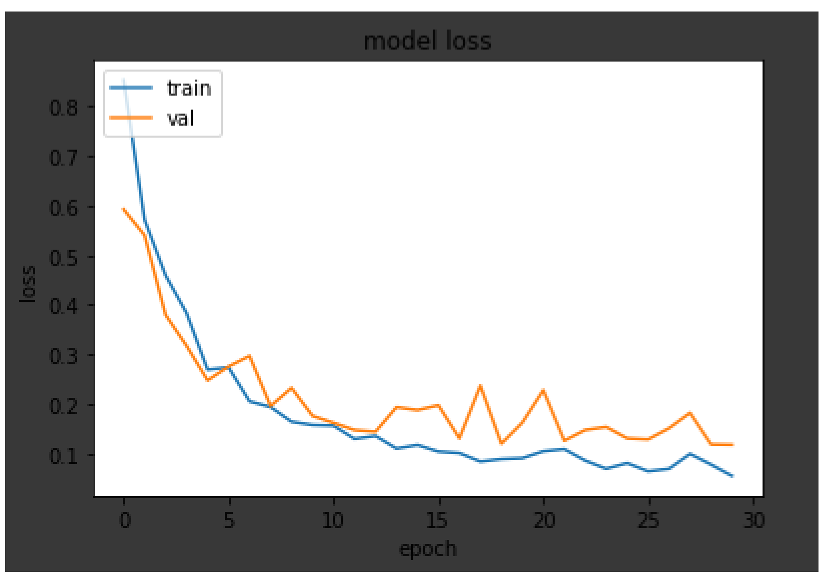

Model architecture and summary:

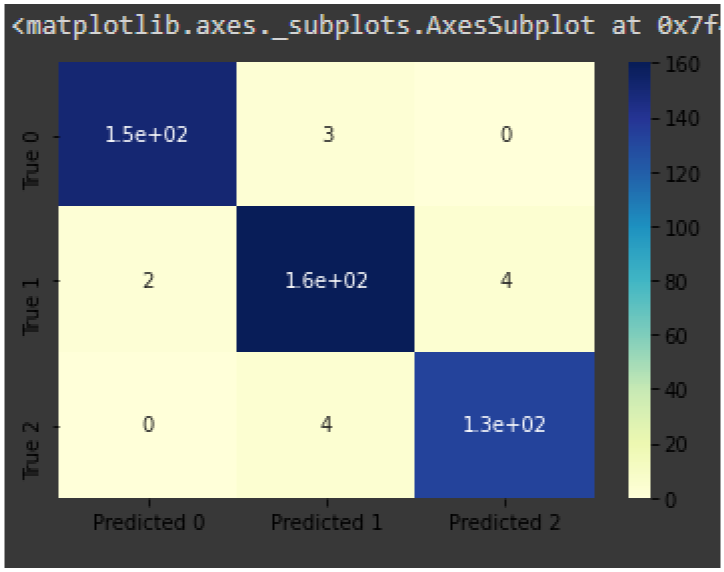
```
model = models.Sequential([
    layers.Conv2D(8, (3,3), strides=(1,1), padding='same', activation='relu',  kernel_regularizer=keras.regularizers.l1_l2(l1=1e-5, l2=1e-4), input_shape=(64,64,1)),
    layers.MaxPooling2D((3,3), padding='same'),
    layers.Conv2D(16, (3,3), strides=(1,1), padding='same', kernel_regularizer=keras.regularizers.l1_l2(l1=1e-5, l2=1e-4), activation='relu'),
    layers.MaxPooling2D((3,3), padding='same' ),
    layers.Conv2D(32, (3,3),strides=(1,1),padding='same', kernel_regularizer=keras.regularizers.l1_l2(l1=1e-5, l2=1e-4),  activation='relu'),
    layers.MaxPooling2D((3,3),padding='same'),
    layers.Flatten(),
    layers.Dense(128, kernel_regularizer=keras.regularizers.l1_l2(l1=1e-5, l2=1e-4), activation='relu'),
    layers.Dense(3, activation='softmax')
])
```

Adam optimization was used as optimizer (learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-7), with sparse categorical crossentropy as the loss.

The model was trained for 30 epochs, resulting in 99.53% training accuracy and 97.1 % testing accuracy.

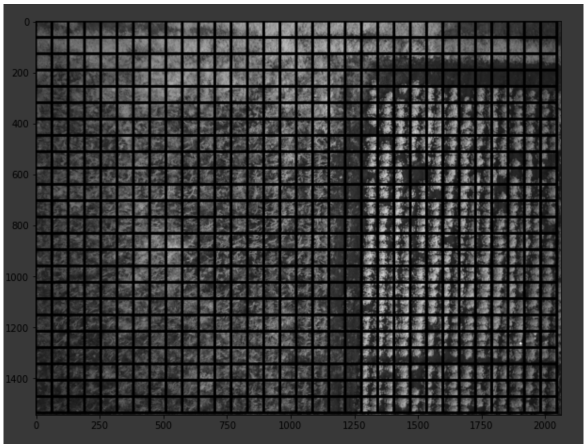


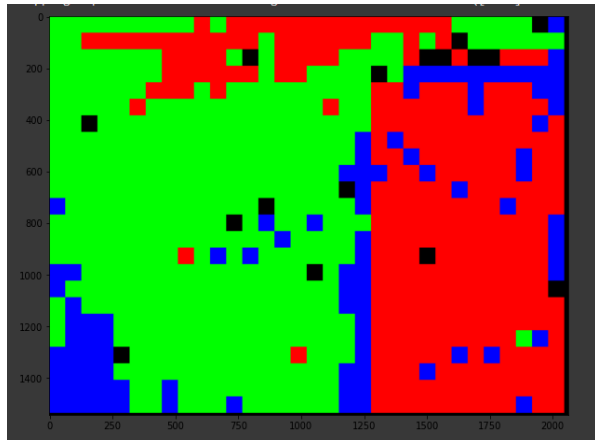



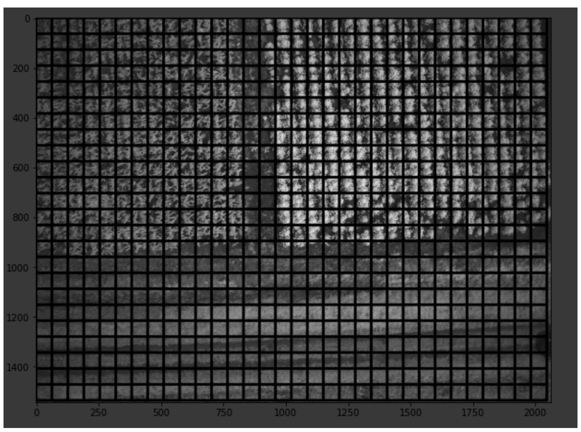


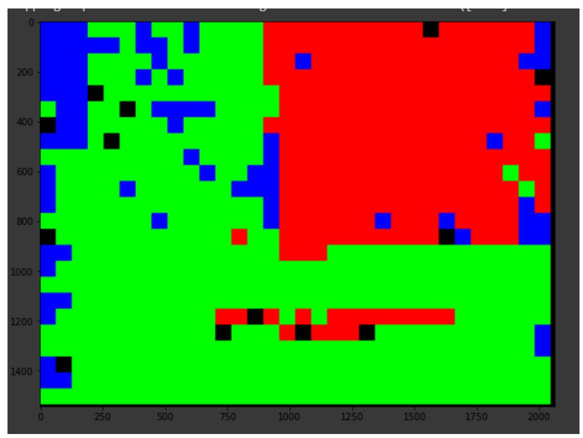

Segmentation results:

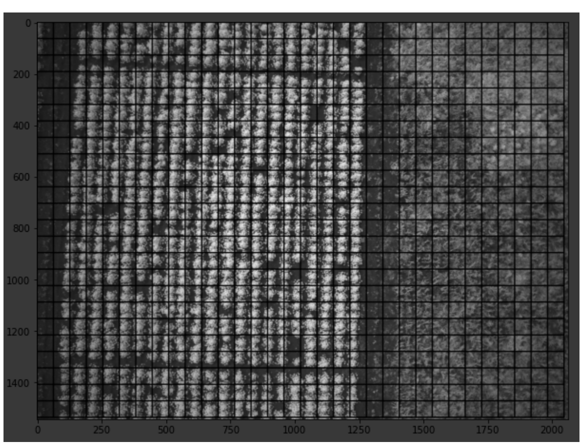
(Note: Green: Maize, Red: Cotton, Blue: Wheat, Black: not confident)

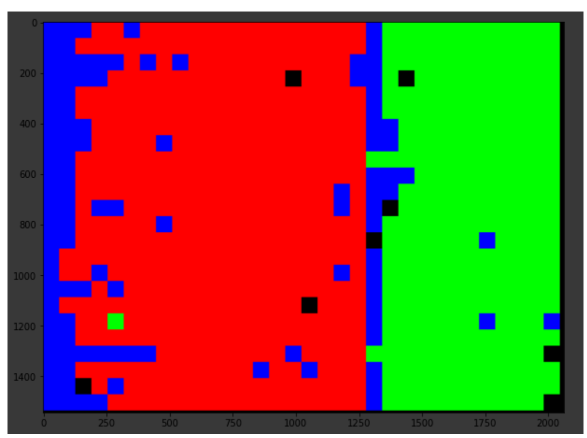


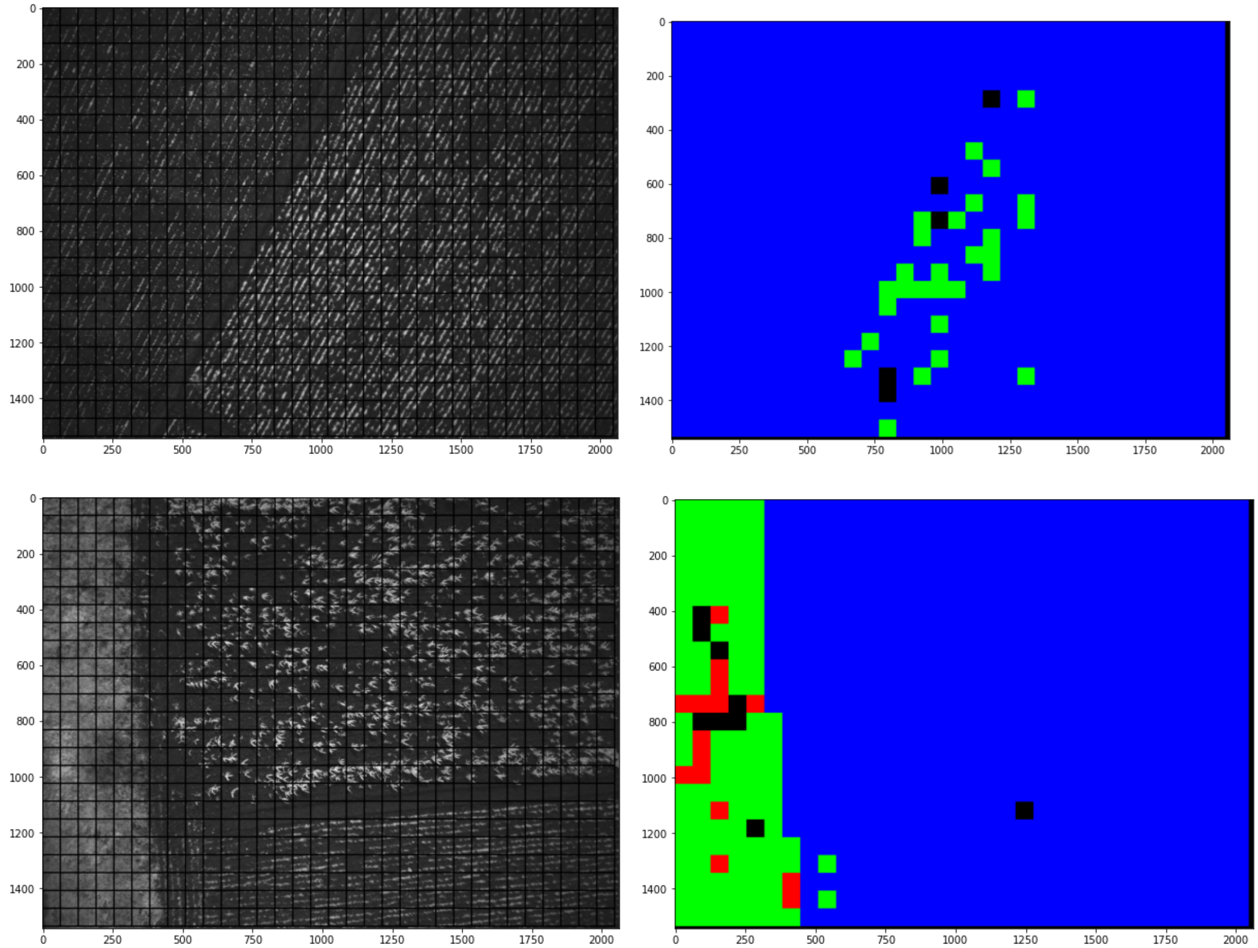










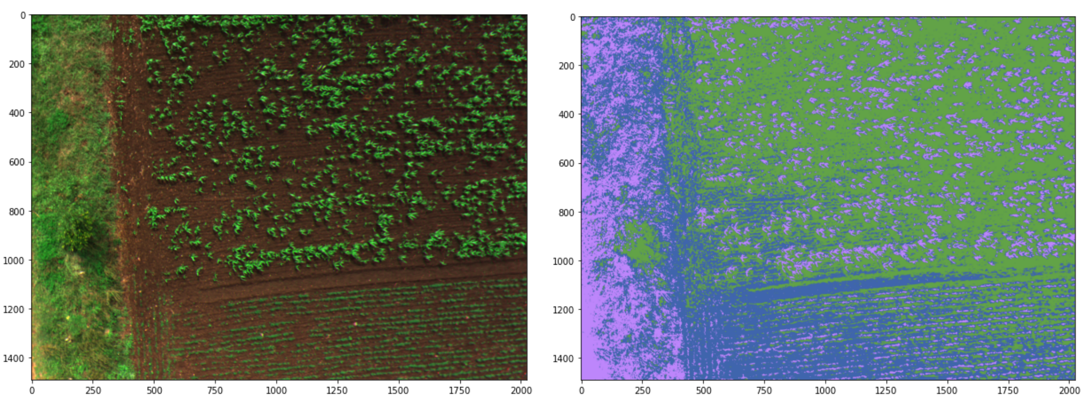


There were some issues concerning misclassification. We did not train the above model over stray grass or unrelated ground. The grass was understood mostly to be Maize.

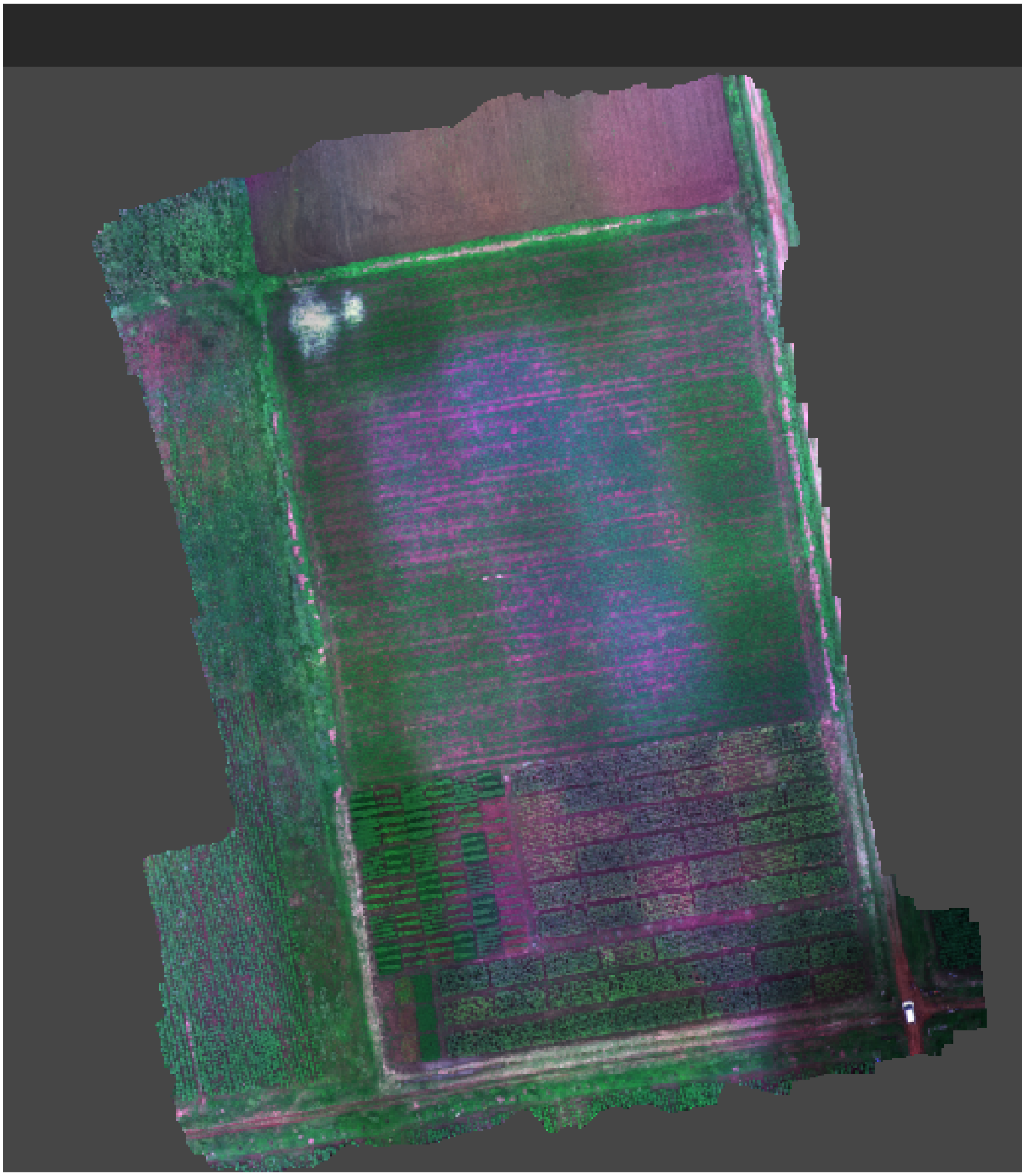
K-Means segmentation:

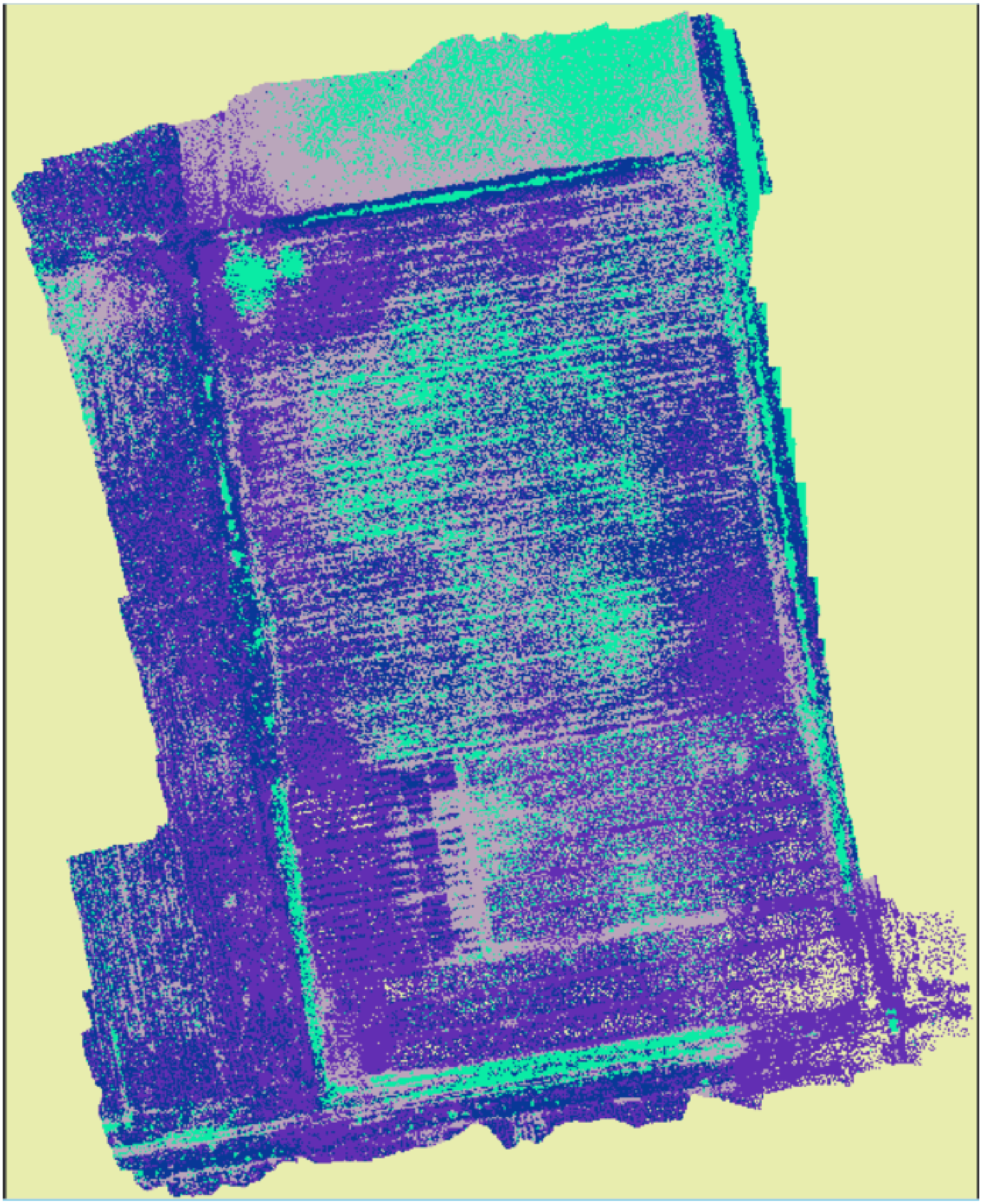
We can also use unsupervised methods like K-means clustering to group pixels together based on their DN values.

(5 bands, 5 color, OpenCV)



(5 bands, 3 color, OpenCV)





(3 bands, 5-color, VISAT BEAM Toolbox)

# Image segmentation and classification using all the 5 bands and 4 classes

## Terminology:

Mega Image: Refers to a complete 5 band aligned image
Segment : Refers to a 64x64 pixel cropped 5 band image

## Aim and brief explanation:

In the previous section and experiments, we had used only the 4th channel i.e only the NIR band. The aim here is to **combine all the 5 channels** and use **segments from these mega images to predict the class of the crop.**
Also note that in the previous section we had kept only 3 classes i.e **Wheat, Cotton and Maize**. Due to this, the network wasn't trained on segments containing barren land and grass covers. We have added another class i.e **Grass** which contains segments containing grassy areas.

## Data Preparation:

All the 5 bands of the raw images were aligned using the MicaSense library. The notebook used to do so can be found here (Or in Crop_Image_BTP/Code/Image-Alignment-with-Micasense). The Raw data and the aligned data can both be found in the Data folder. The aligned folder has a '-Align' appended to the end of the folder name. These images cannot be opened in the drive itself as they contain 5 bands. To view any such image, the corresponding '-Thumbs' folder can be seen. Note that the thumbnails are not that high in quality and are just used as a reference.

Now these aligned images or 'Mega' images can be segmented into 64x64 pixels and labelled. This is done using this notebook.
For the labelling of grass we labelled a higher no. of images as the grass cover or the barren land cover is a bit different near each type of crop.
We use the following no. of mega images to divide them into segments which can be used for training by the model:

Cotton: 2 mega images: 681 segments
Grass: 12 mega images: 2609 segments (600 used for training)
Maize : 3 mega images: 652 segments

Wheat: 3 mega images: 538 segments

## Training:

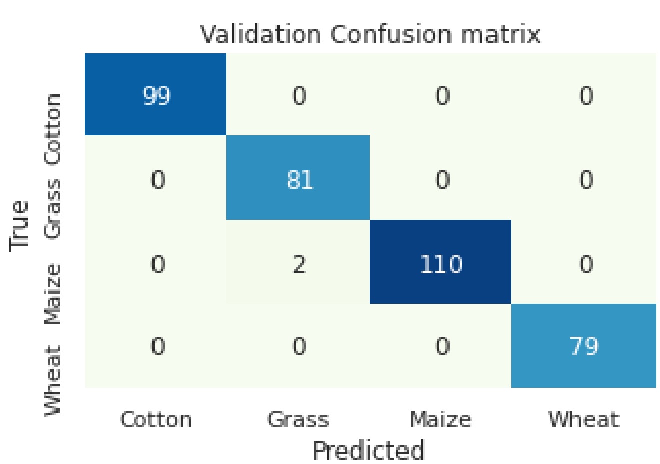
The segmented data was split into validation and test sets with 0.15 ratio.
The following architecture was used

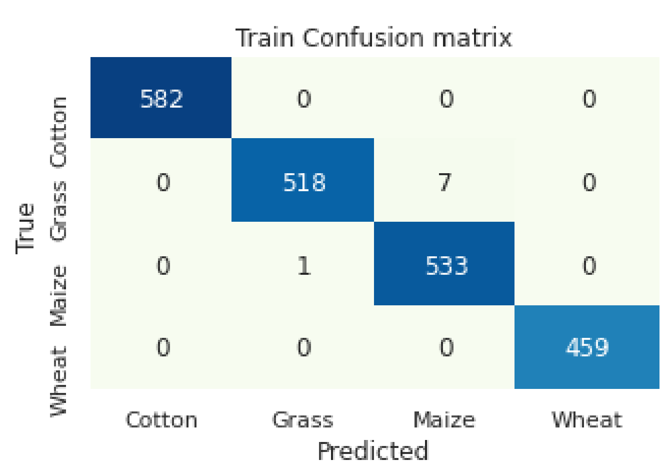
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 64, 64, 8)         368
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 22, 22, 8)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 22, 22, 16)        1168
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 16)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 32)          4640
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 288)               0
_________________________________________________________________
dense (Dense)                (None, 128)               36992
_________________________________________________________________
dense_1 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 516
=================================================================
Total params: 60,196
Trainable params: 60,196
Non-trainable params: 0
_________________________________________________________________
```

The model was trained for 30 epochs and took about 20 seconds. The low duration is mainly due to the small size of each 64x64 pixel segment (~40 KB)

We were able to reach train accuracy of 99.81% and a validation accuracy of 99.46%

**Validation and Train Confusion matrices** respectively:



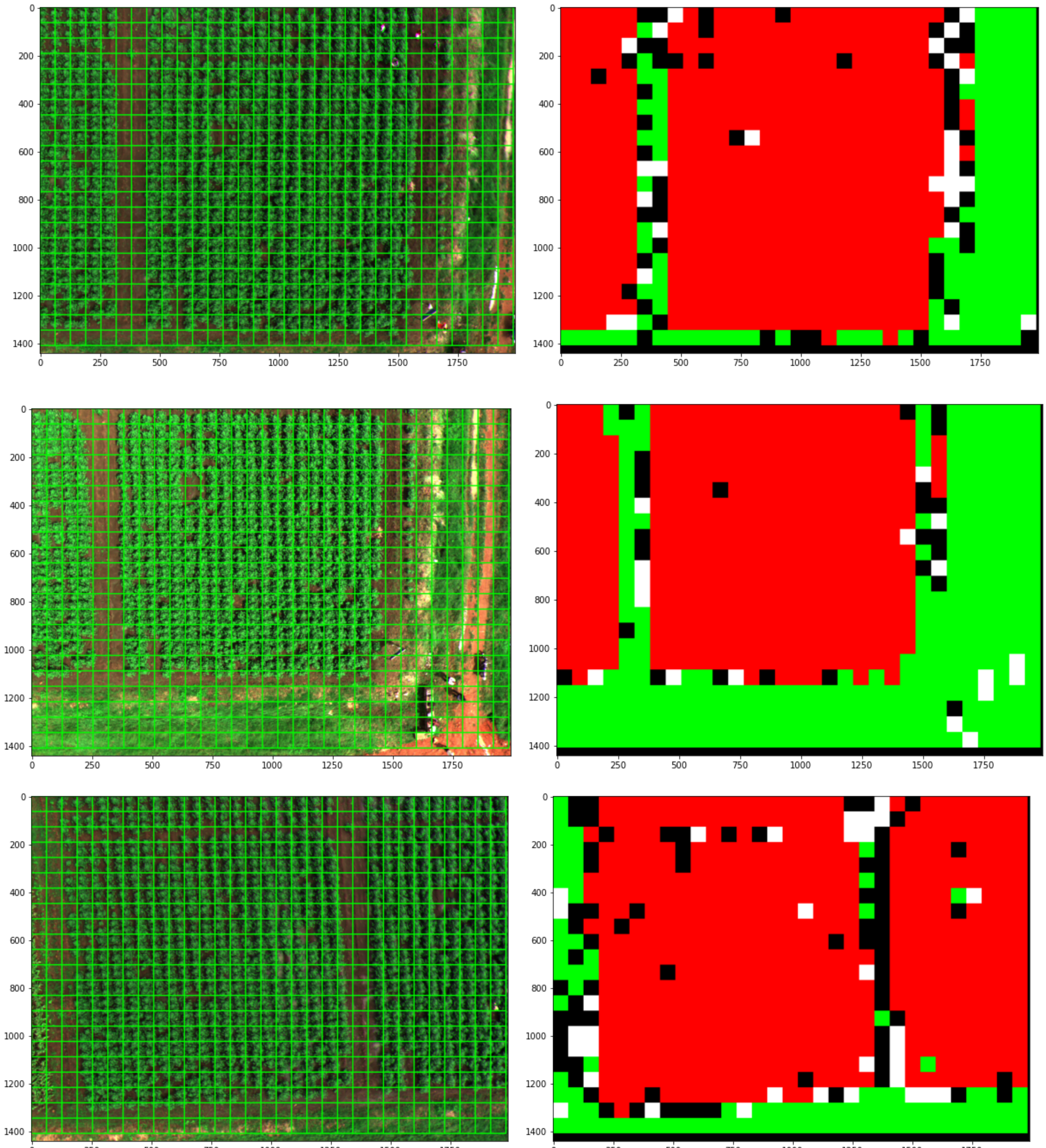



NN performance on Mega images

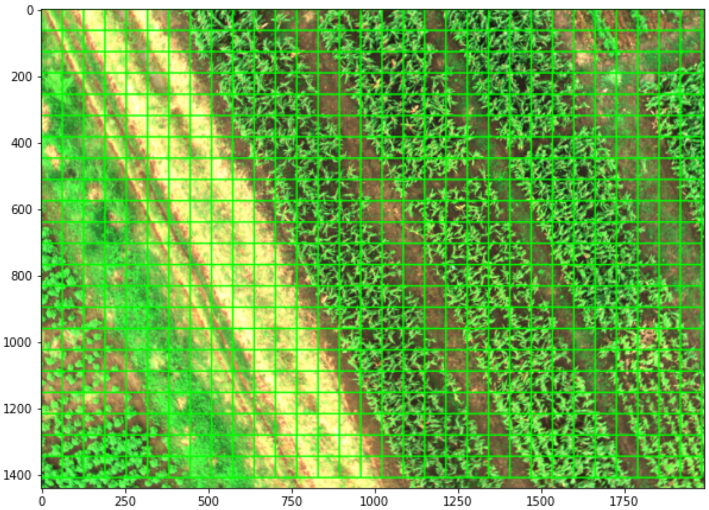
Color Scheme:

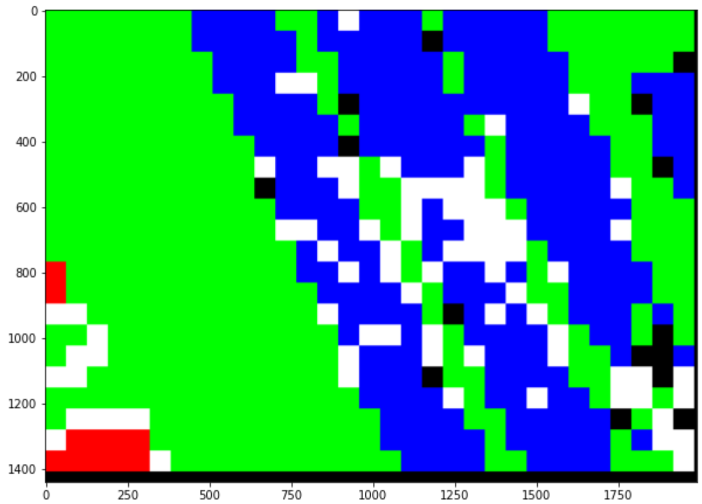
1. Cotton : Red
2. Grass : Green
3. Maize : Blue
4. Wheat : White
5. Undeterminable: No color/ Black

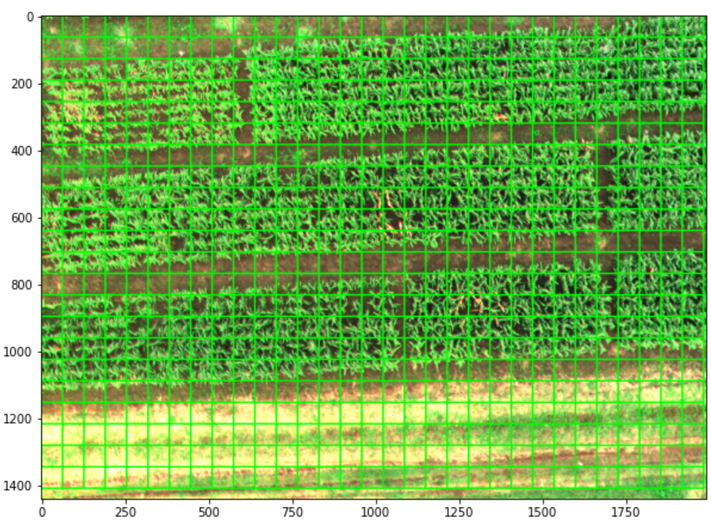
# Cotton Images examples

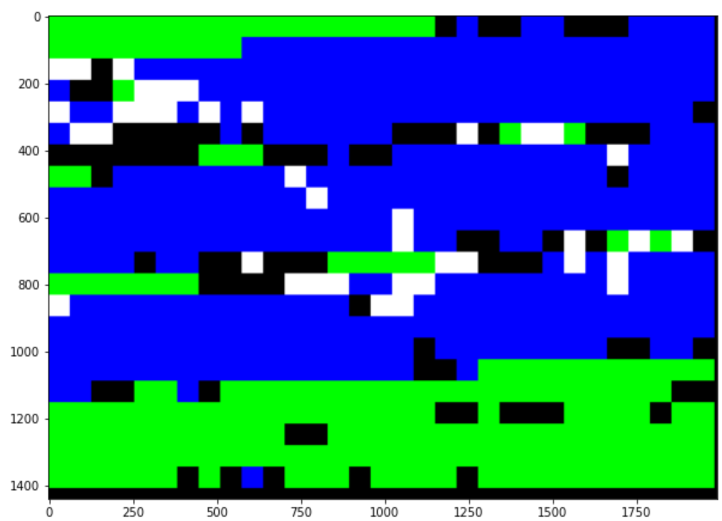


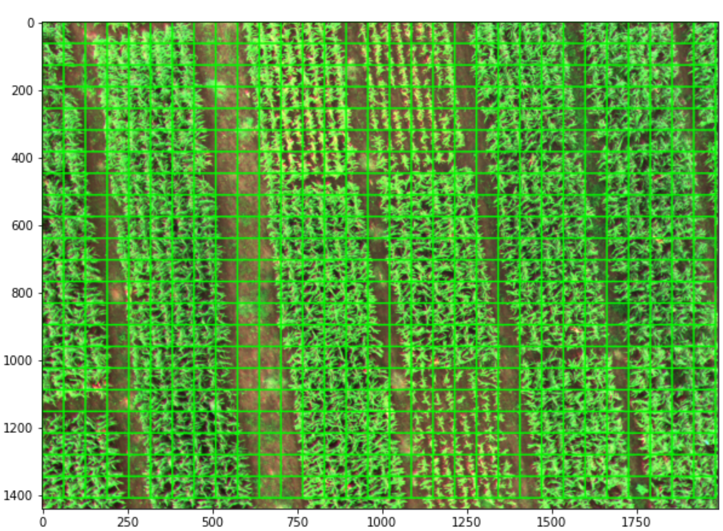
Maize Images examples:

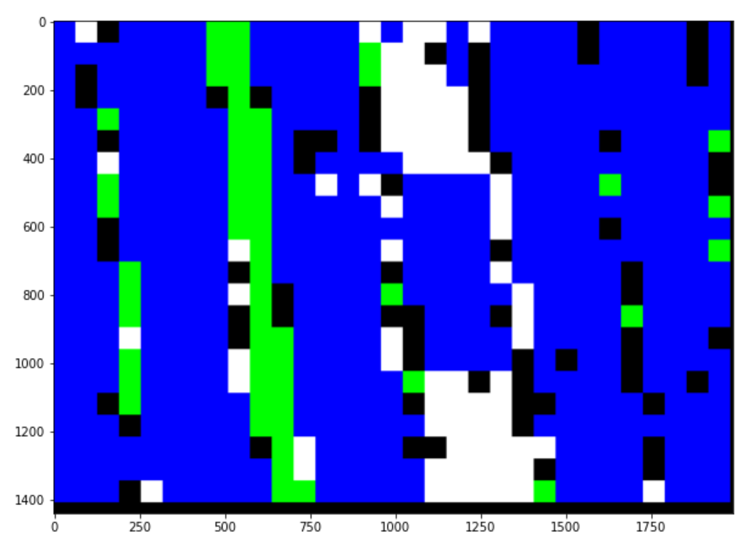


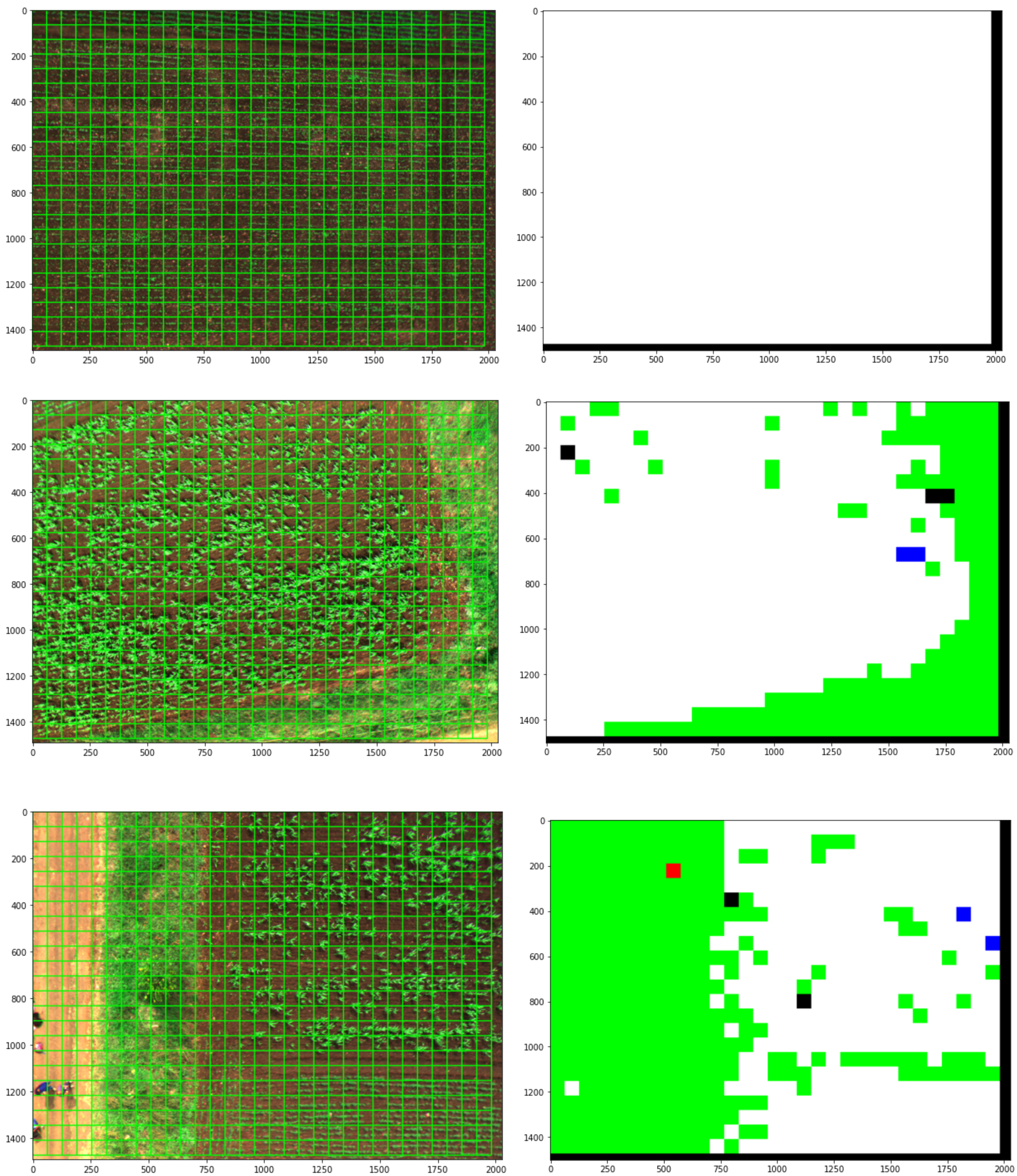



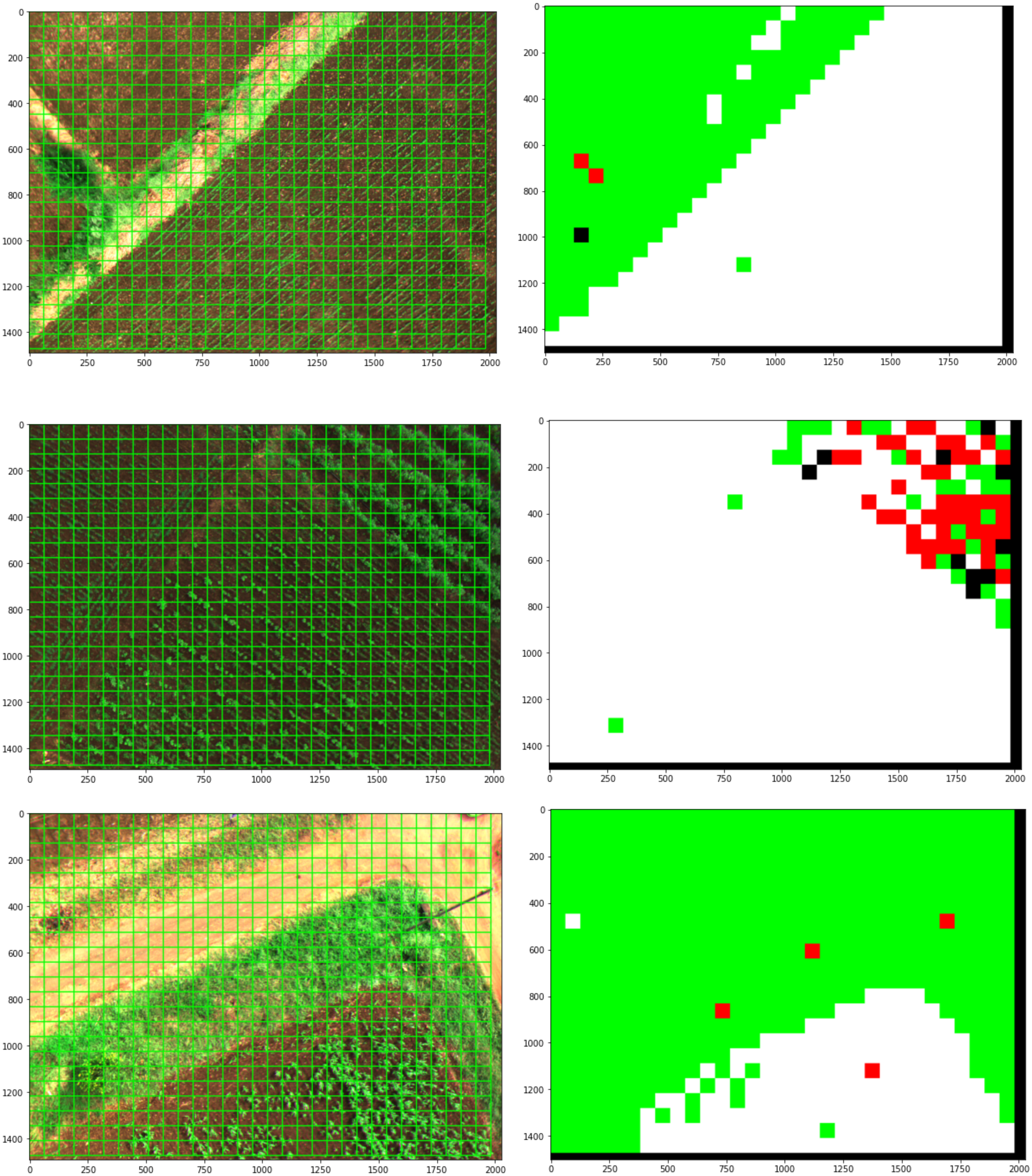



## Wheat Images example:





# Image Segmentation and Classification using 32x32 segments

## Aim/Reasoning behind this model

The last model worked on segmented images of size 64x64. We observed that a 64x64 segment generally doesn't contain a single crop. Also reducing segment size to 32x32 will increase the total no. of segments in an image 4 times. We label the smaller segments in a similar way as we did for 64x64 and use the same MEGA images that aligned in the previous section. The following are the total no. 32x32 segments which are used to train our model.

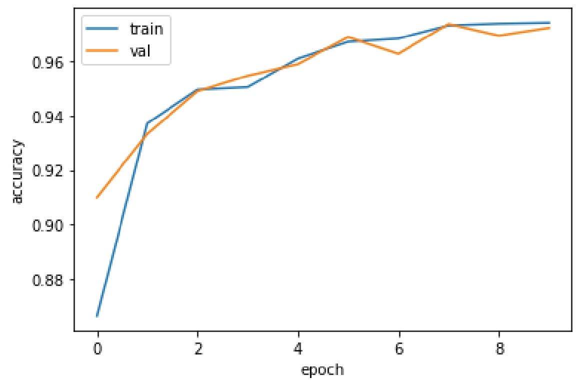
In the grass label, we have also added images containing only soil.

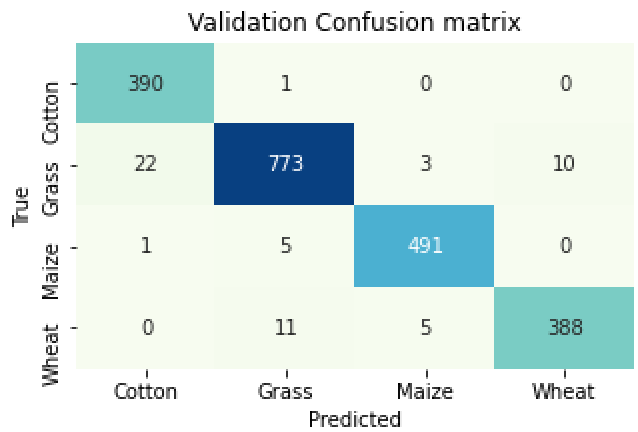
Cotton: 2716 segments
Grass: 5000 segments
Maize : 3299 segments
Wheat: 2982 segments

## Training:

The segmented data was split into validation and test sets with 0.15 ratio.
The following architecture was used

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 8)         368
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 11, 11, 8)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 16)        1168
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 4, 4, 16)          0
_________________________________________________________________
flatten (Flatten)            (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 32)                8224
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 132
=================================================================
Total params: 9,892
Trainable params: 9,892
Non-trainable params: 0
_________________________________________________________________
```



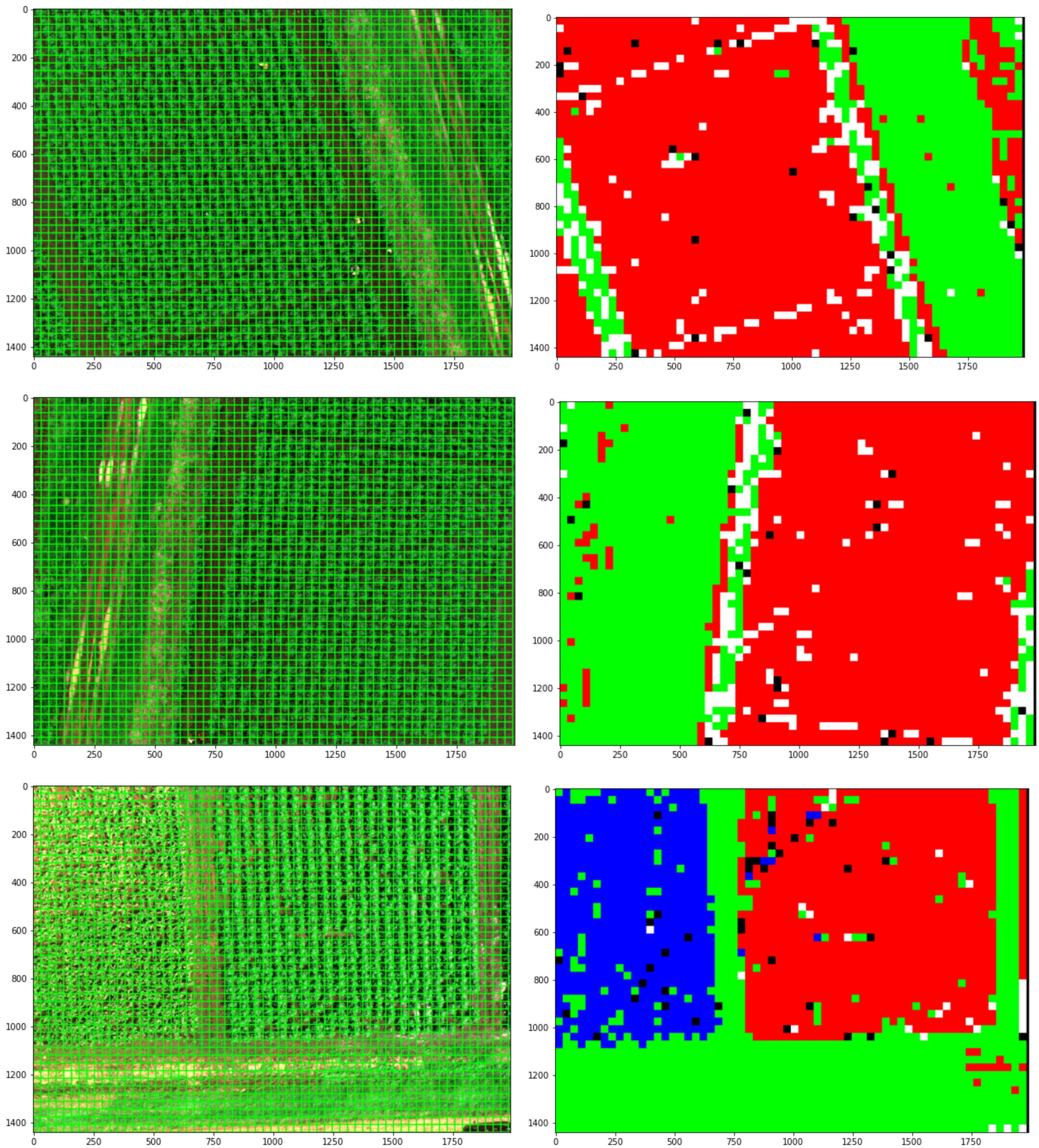

We were able to reach train accuracy of 97.43% and a validation accuracy of 97.24%

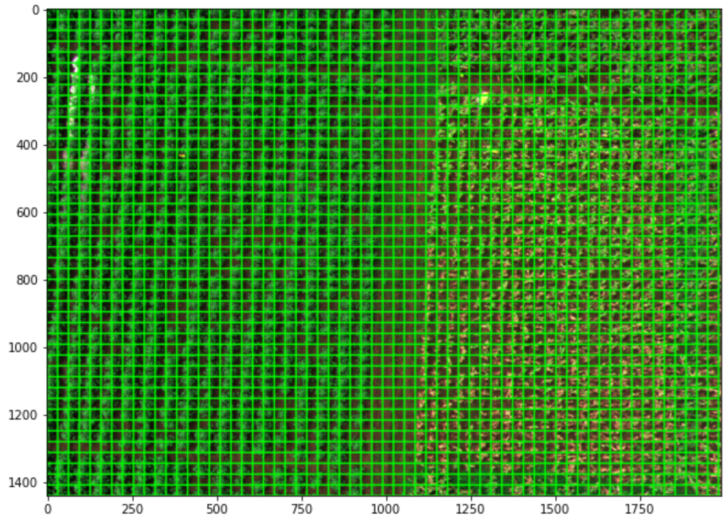
## NN performance on Mega images

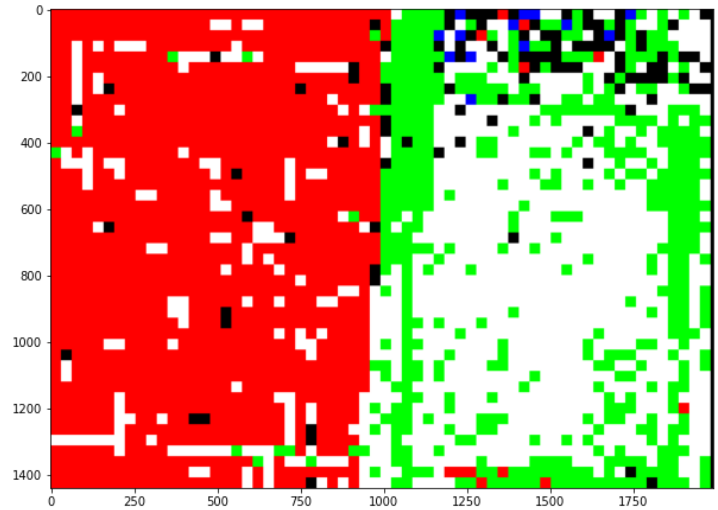
Color Scheme:

Cotton : Red
Grass : Green
Maize : Blue
Wheat : White
Undeterminable: No color/ Black

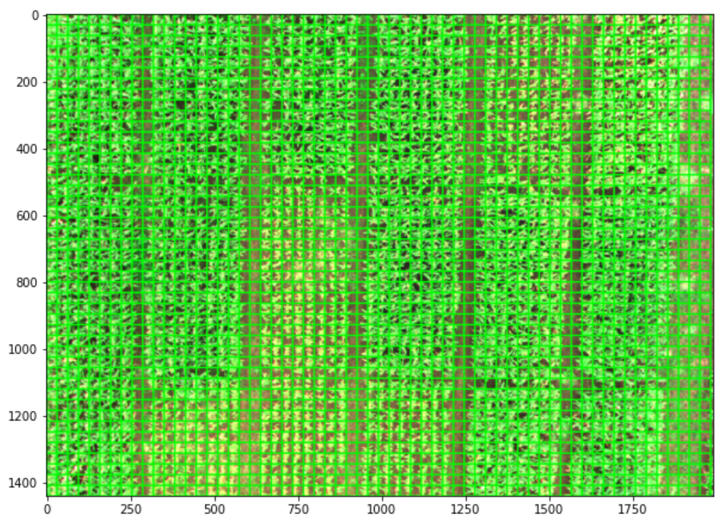
Cotton Images examples:

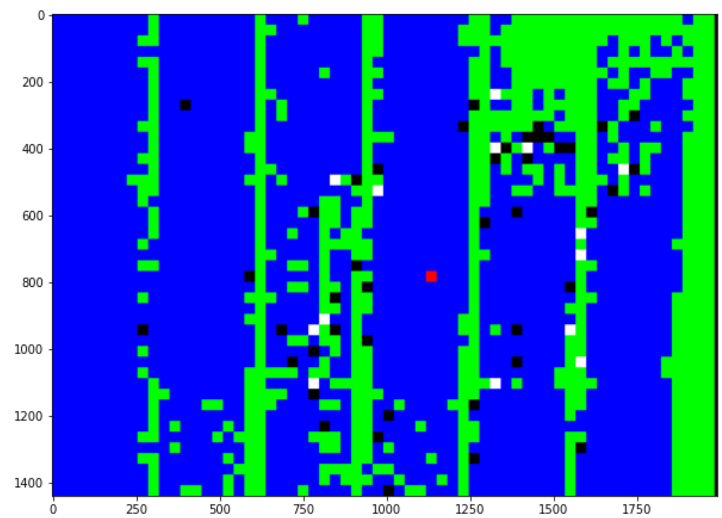


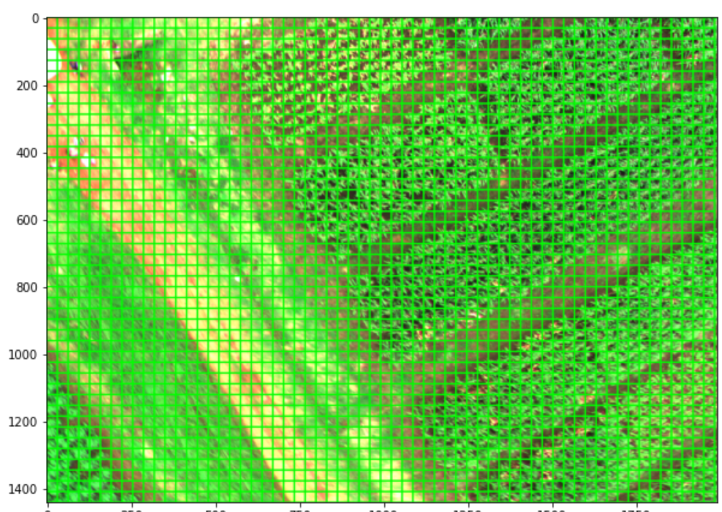


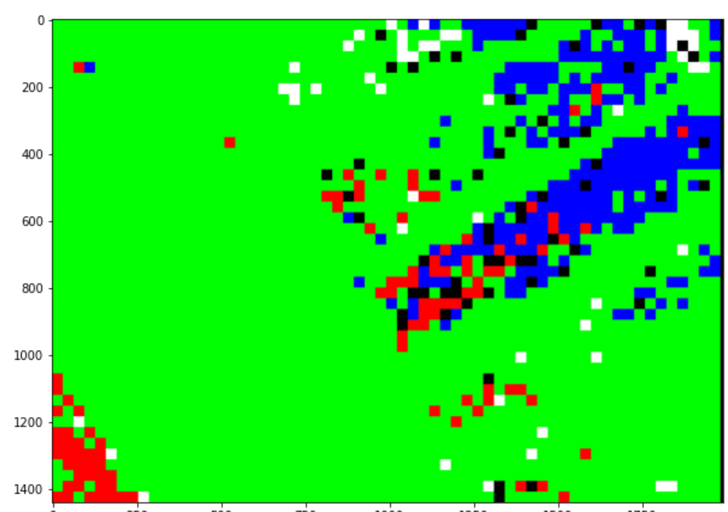


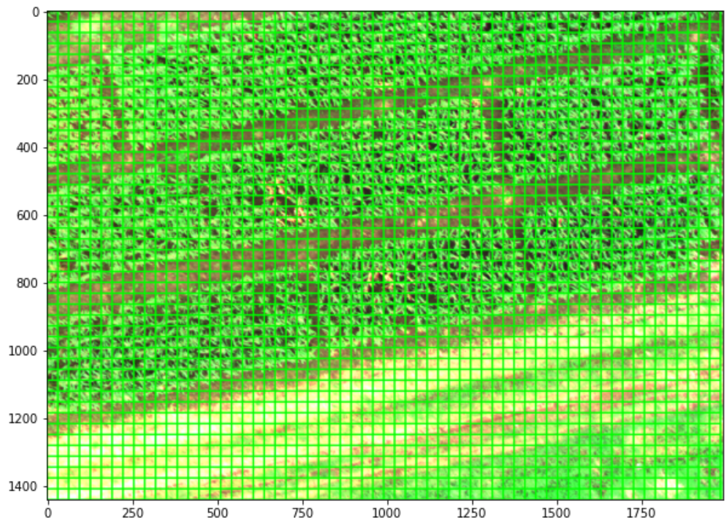

Maize Images examples:

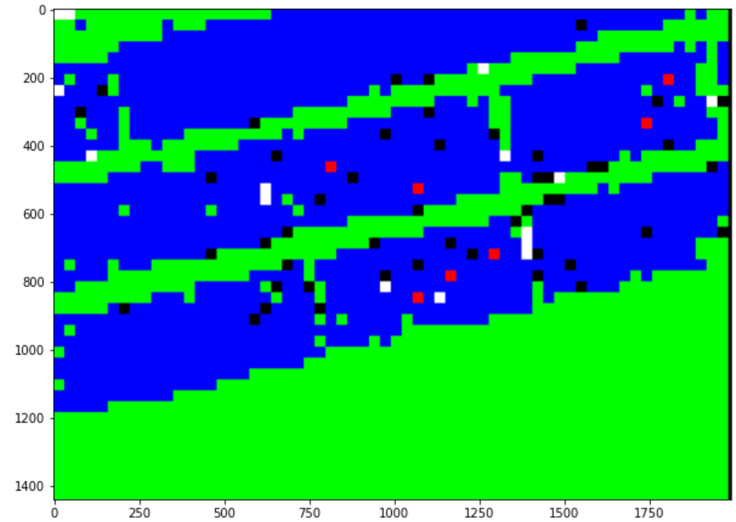




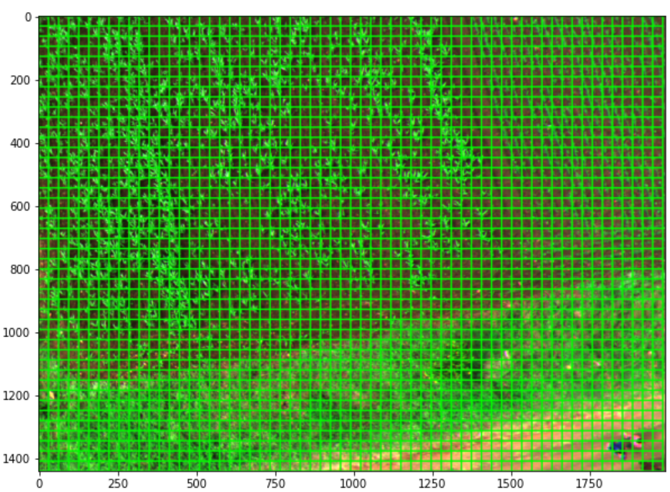


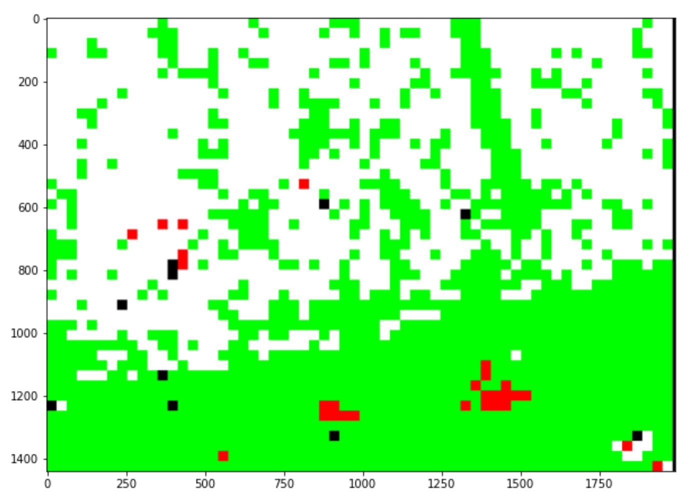


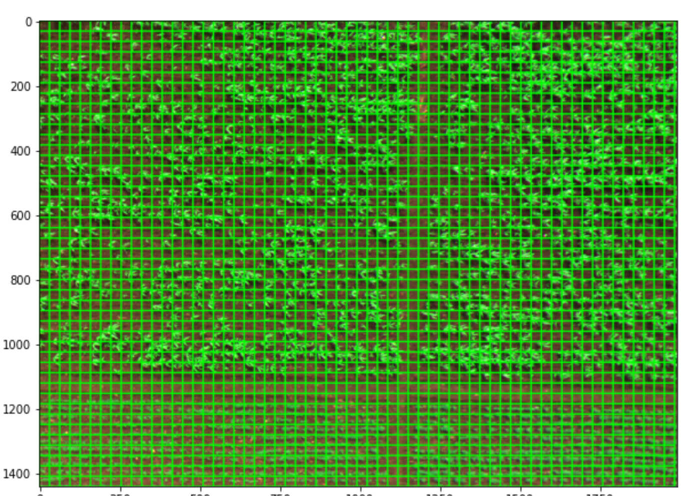


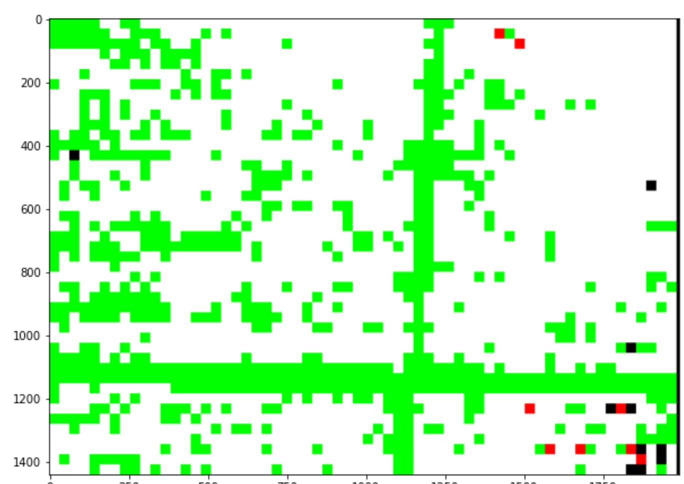


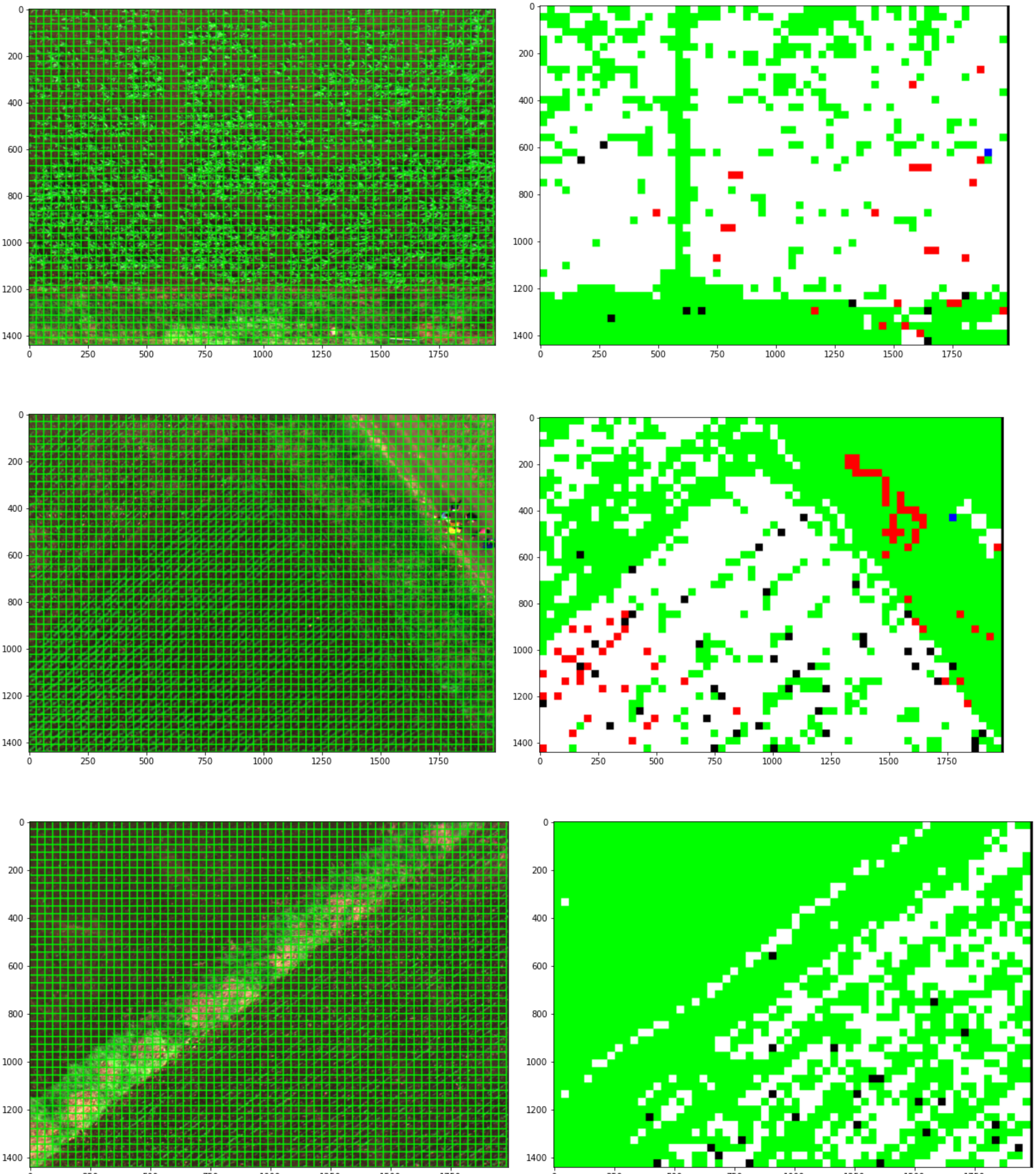

Wheat Images examples:

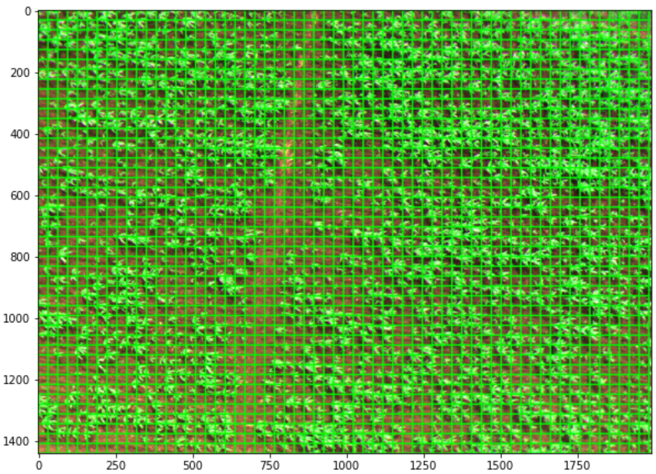


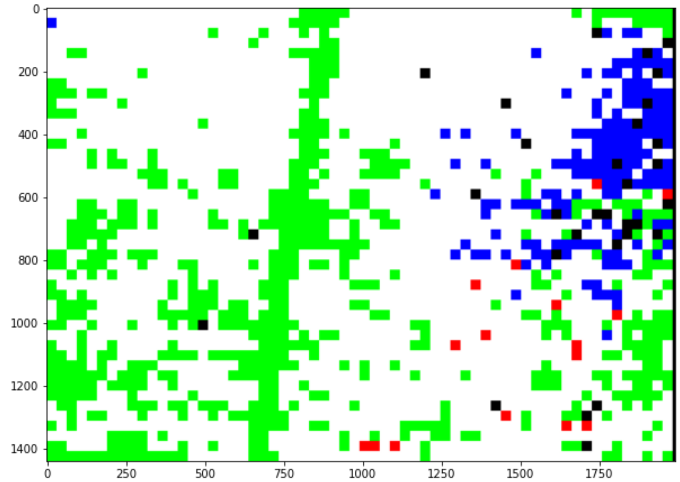










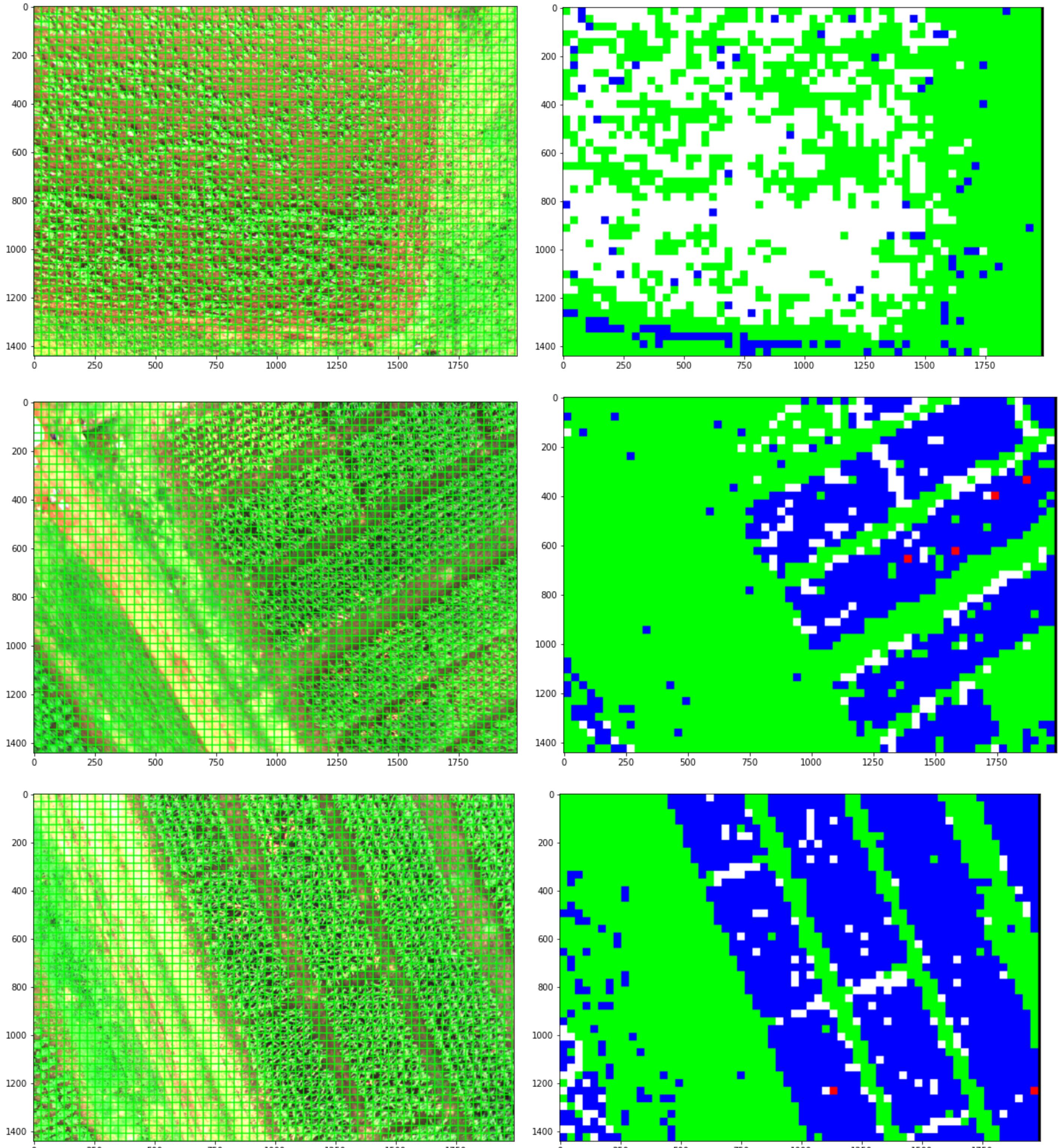


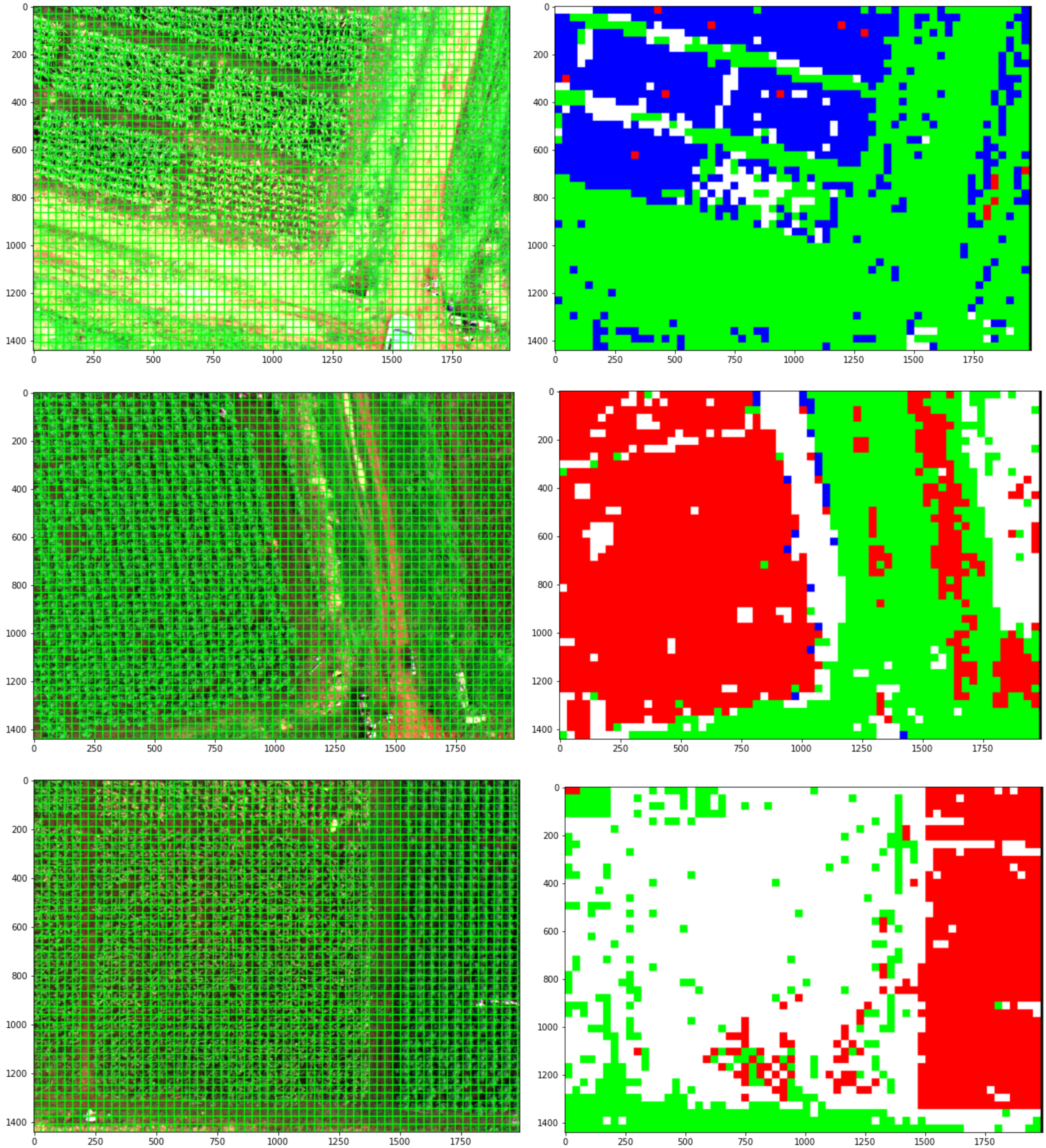
# Using Random Forests instead of Neural Networks

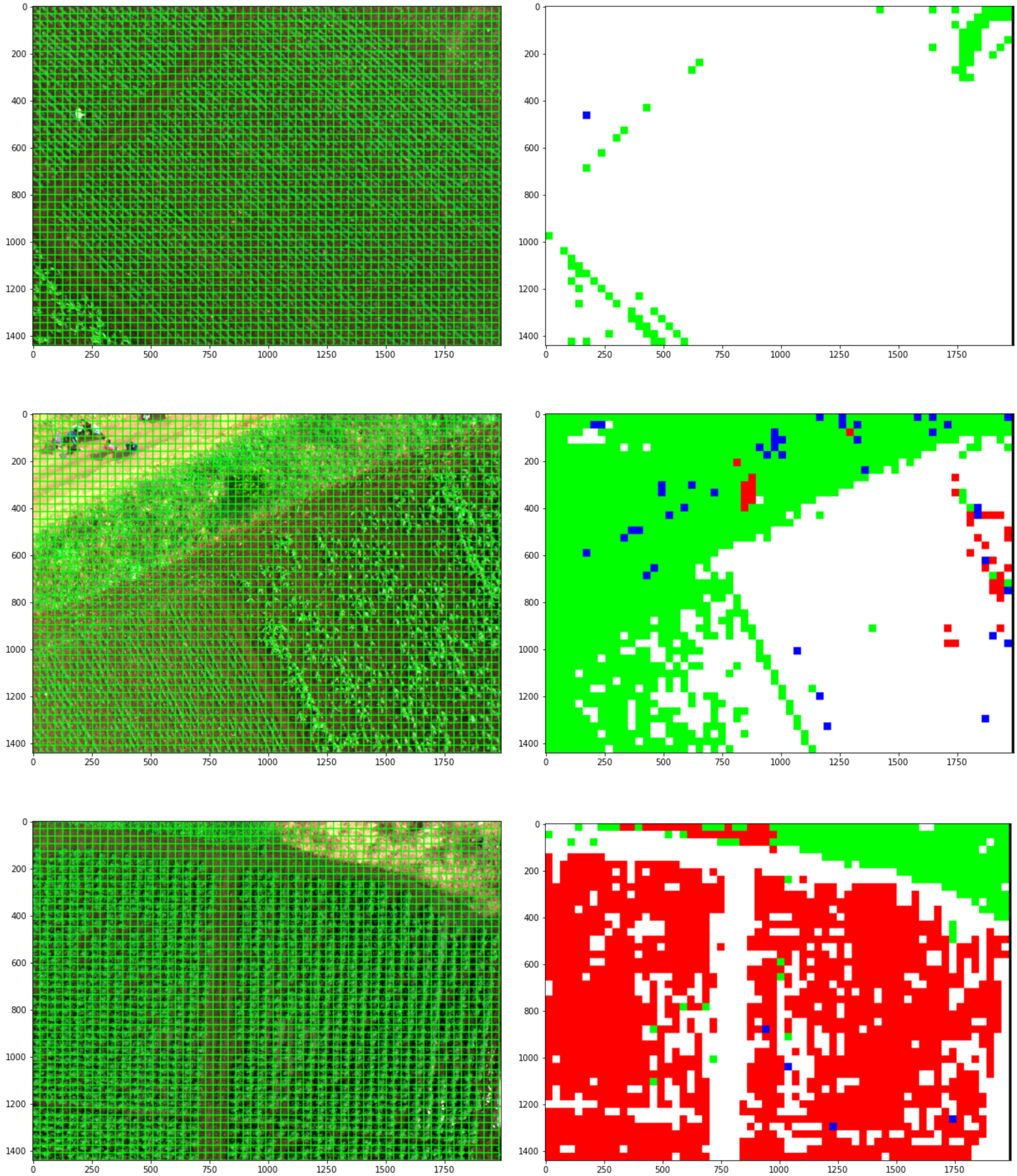
We tried using random forests instead of neural networks on the same 32x32 labelled dataset. Following are the results

Total Image segments used: 13997 (Same as NN)
No. of estimators used: 100
Test:Train ratio = 0.15
Test Accuracy achieved: 93.5 %

## Some Examples:


0
200
400
600
800
1000
1200
1400
0 250 500 750 1000 1250 1500 1750
0
200
400
600
800
1000
1200
1400
0 250 500 750 1000 1250 1500 1750
0
200
400
600
800
1000
1200
1400
0 250 500 750 1000 1250 1500 1750
0
200
400
600
800
1000
1200
1400
0 250 500 750 1000 1250 1500 1750
0
200
400
600
800
1000
1200
1400
0 250 500 750 1000 1250 1500 1750
0
200
400
600
800
1000
1200
1400
0 250 500 750 1000 1250 1500 1750


0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750



0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750
0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750
0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750
0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750
0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750
0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750

# Optimizer Experiments

## Window size = 32x32

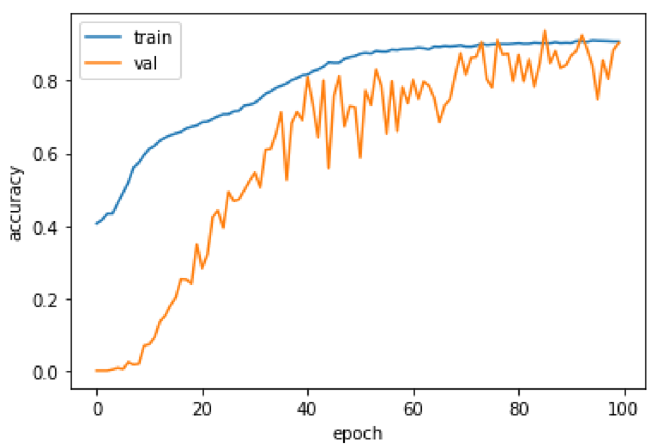
## **Only NIR band** (3 Classes)

**<u>To add Confusion Matrix</u>**

1.)

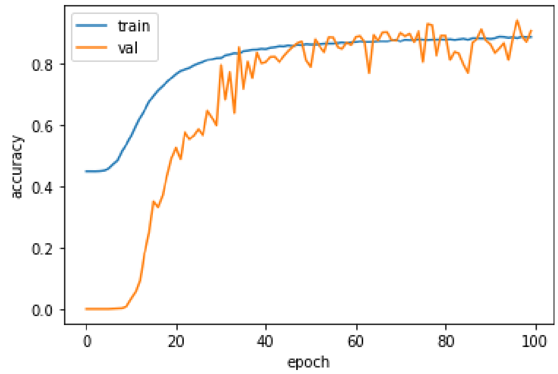
Dataset: C,M,W = train : 2000, 2000, 1853
Test : 403, 429, 432

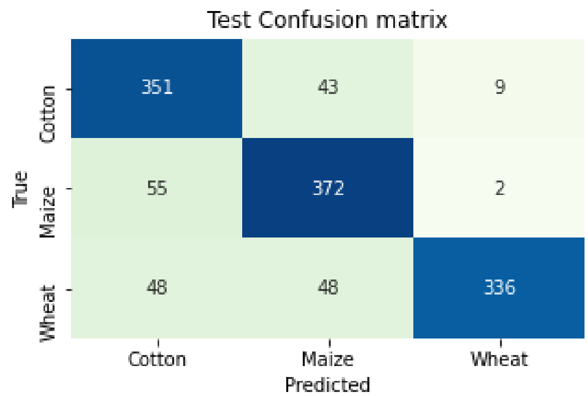
SGD, LR = 0.001, nesterov = True
Dropout = 0.5
Activation : relu and softmax(out layer)
100 epochs
Train accuracy : 90.83, Validation accuracy 90.43,
Test accuracy 80.46



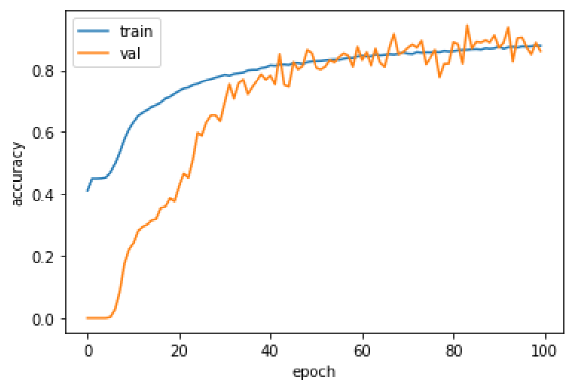

2.)

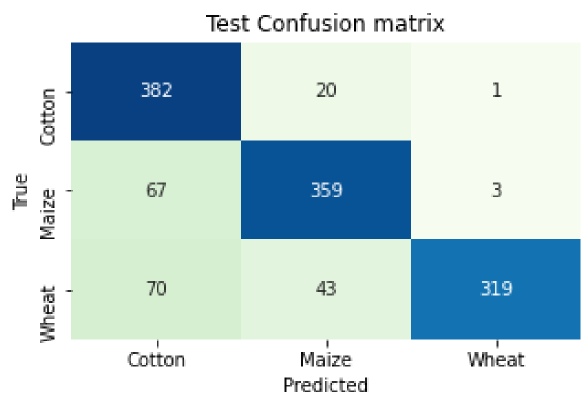
Data : 2395, 3223, 2664
SGD, LR = 0.001, nesterov = False
Dropout = nill
Activation : relu and softmax(out layer)
100 epochs
Train accuracy 88.75, vali, 90.7
Test 82



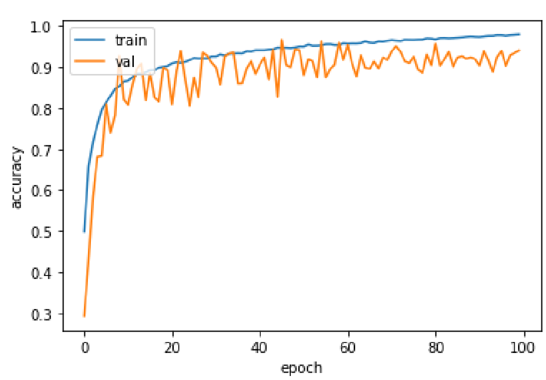



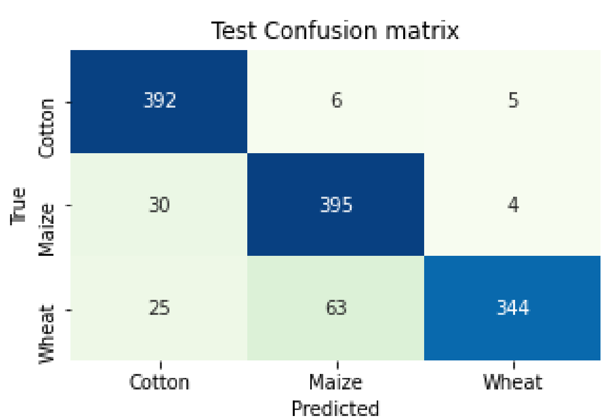

3.)
Activation = tanh
100 epochs
Train accuracy 87.88, vali, 86.16
Test 83.8



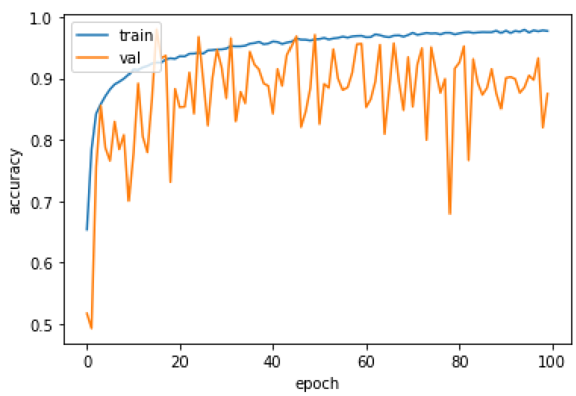



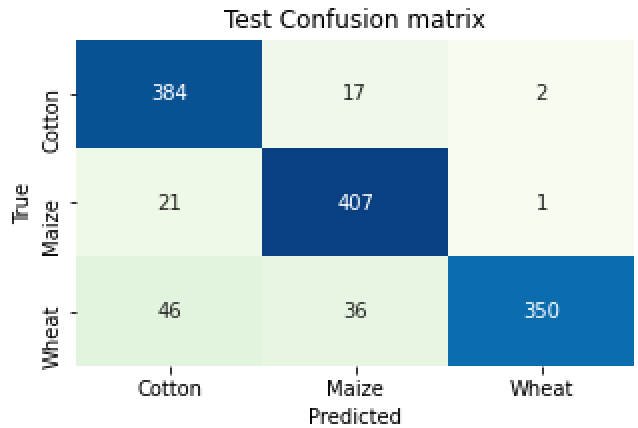

4.)Activation = tanh, LR = 0.01
100 epochs
Train accuracy 97.97, vali 94.06
Test 89.4



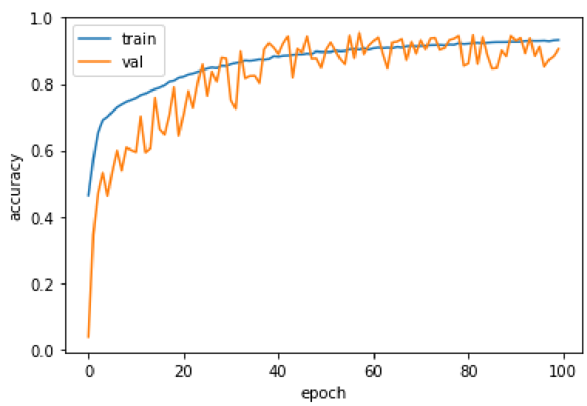



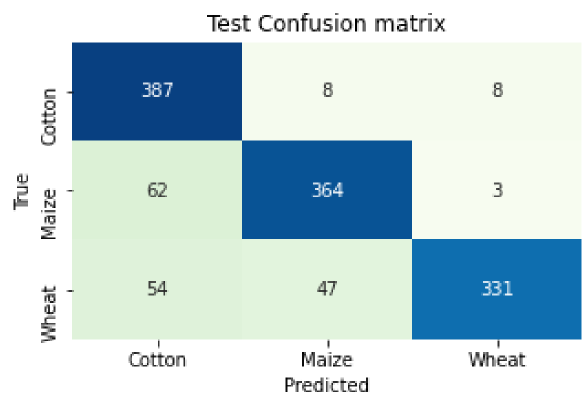
5.) Activation = tanh, LR = 0.001
RMSProp 100 epochs
Train 97.70, vali 87.46
Test 90.27



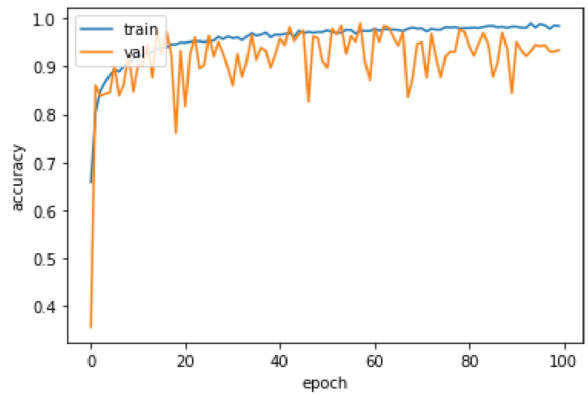



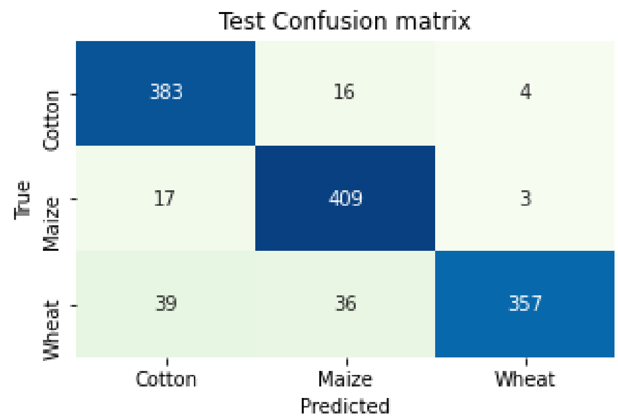

6.) Activation = tanh, LR = 0.0001
RMSProp 100 epochs
Train 93.24, vali 90.47
Test 85.6



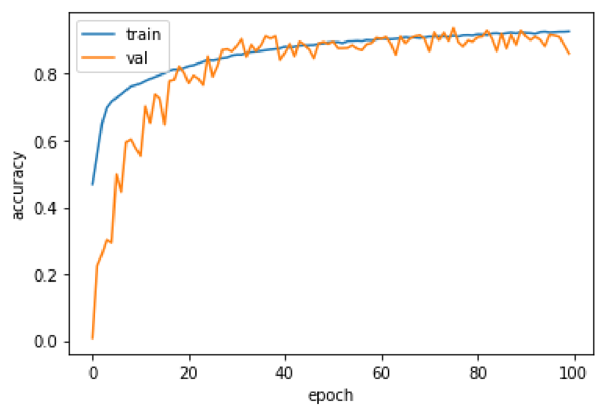



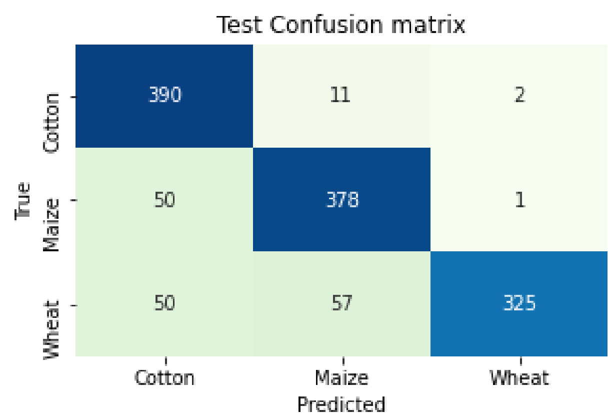

7.) Activation tanh, LR = 0.001
Adam 100 epochs
Train 98.45 Vali 93.4
Test 90.9




8.) activation tanh, LR = 0.0001
Adam 100 epochs
Train 92.7, vali 86.07
Test 86.47



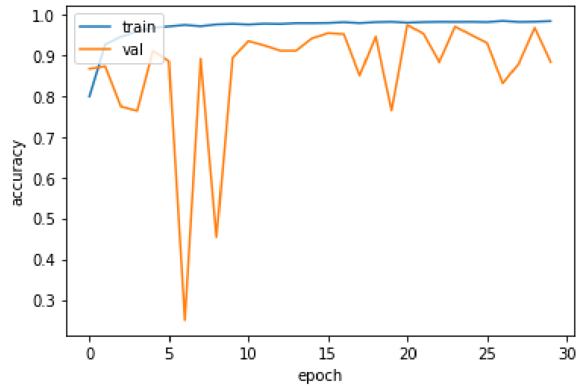



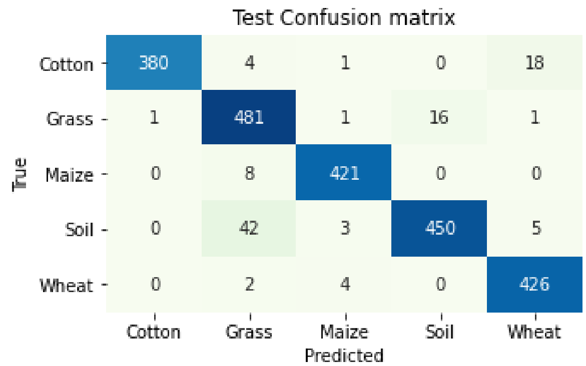

# RGB images

Data (C G M S W): Train :2395, 2500, 3123, 2500, 2664
Test : 403, 500, 429, 500, 432

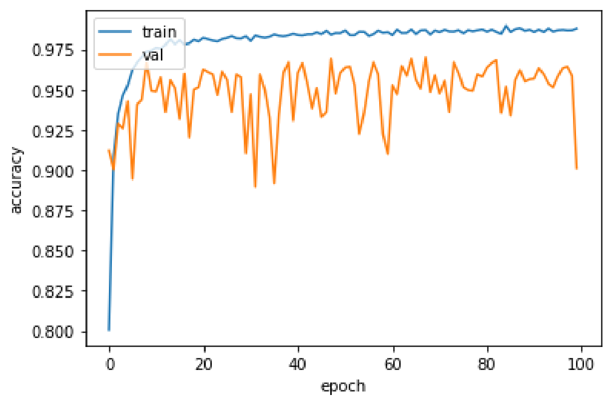
Adam, LR = 0.001, tanh 30 epochs

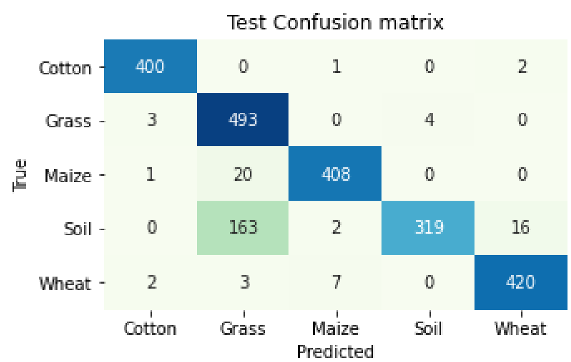
Accuracy
Train 98.38, val 88.37
Test 95.3



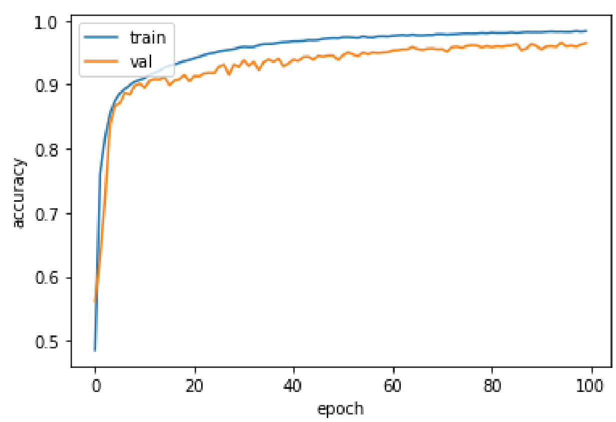



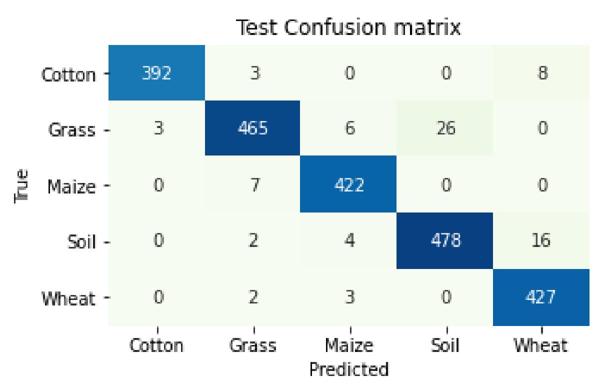

2.) adam 100 epoch , LR= 0.001
Train 98.70, valida 96.69
Test 94.9



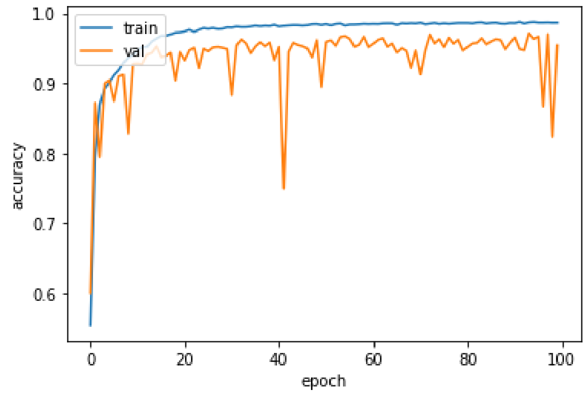

3.) Adam 100 epochs, LR = 0.0001
Train 98.35, validation 93.12
Test 96.22

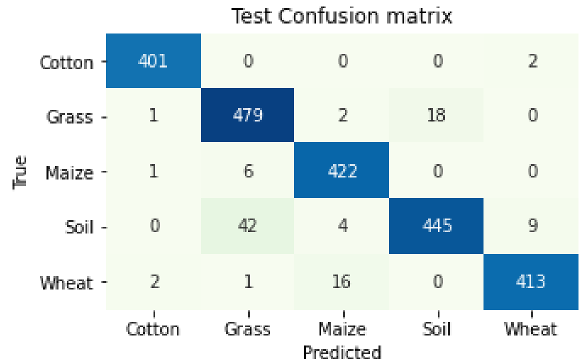


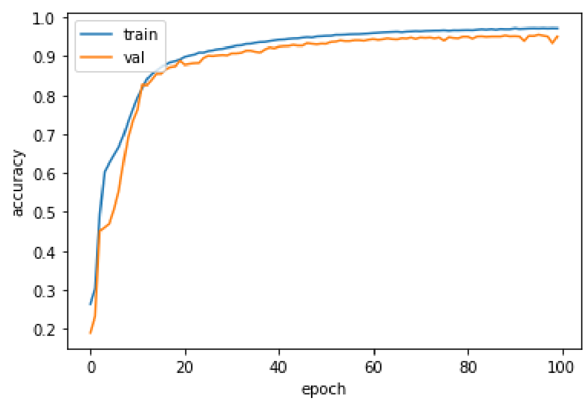

'/content/drive/My Drive/Crop_Image_BTP/Code/RGB_Model_5_classes_96.2'

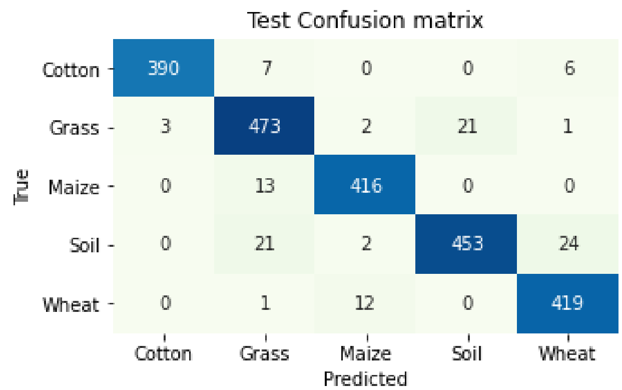
4.) SGD, 100 epochs, LR = 0.01
Train 98.64, val 94.96
Test 91.6




5.) SGD, 100 epochs, LR = 0.001
Train 97.15, val 93.10
Test 94.51



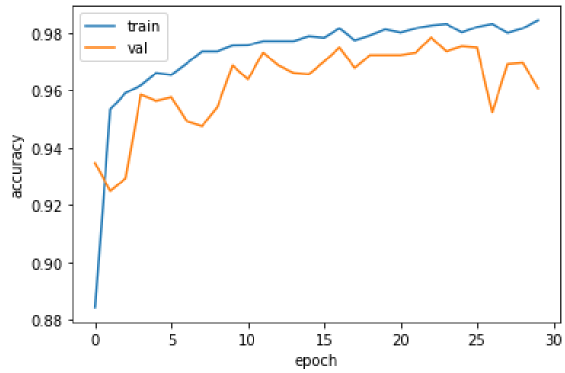



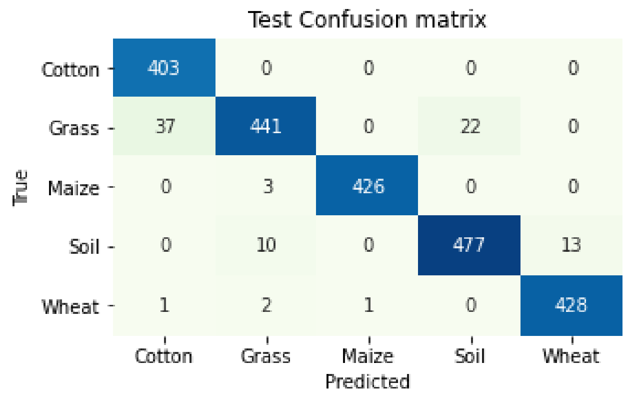

# 5-Band images

Data (C G M S W): Train :2395, 2500, 3123, 2500, 2664
Test : 403, 500, 429, 500, 432 (Same as RGB)

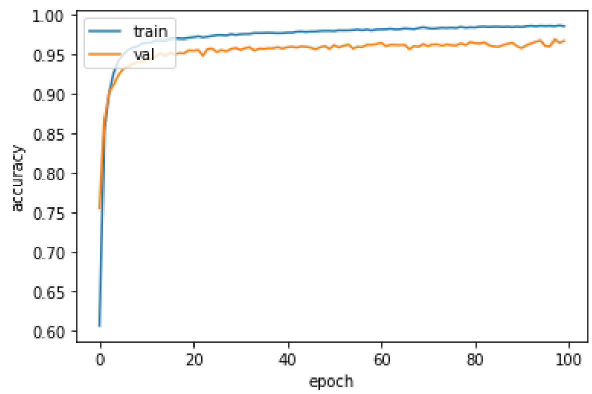
1. Adam 30 epochs LR = 0.001, tanh

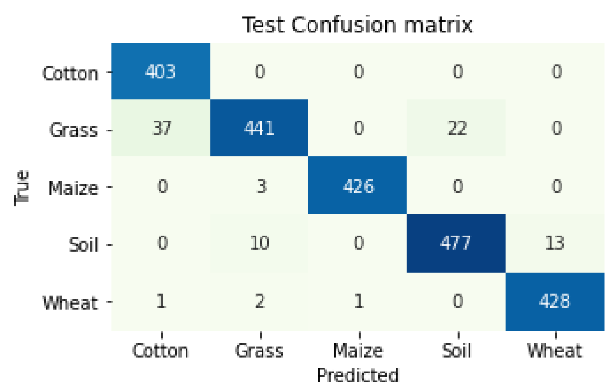
Train 98.4, validation 94.6
Test 96.07



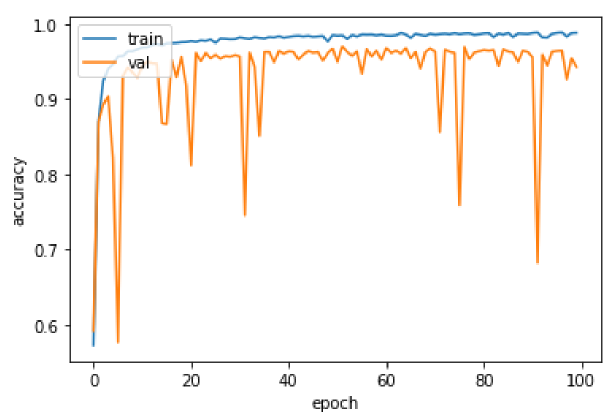



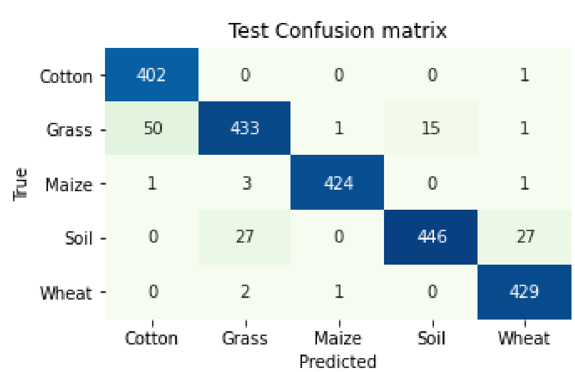

2 Adam 100 epochs LR = 0.0001 tanh
Train accuracy 98.5, validation 93.37
Test 96.6



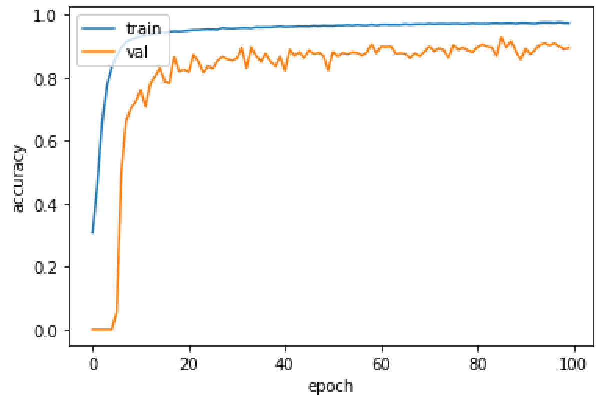



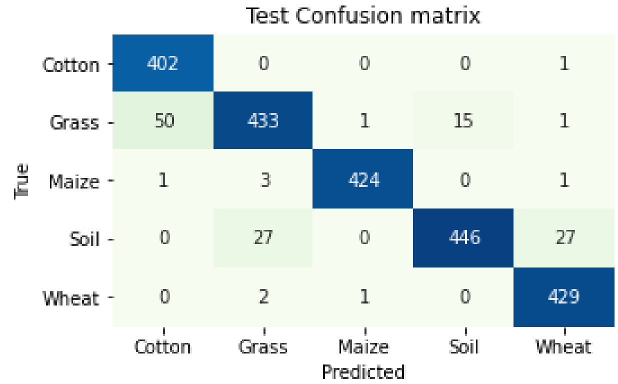

3. SGD 100 epoch, LR = 0.01 relu
Train 98.79 validation 88.9
Test 94.2



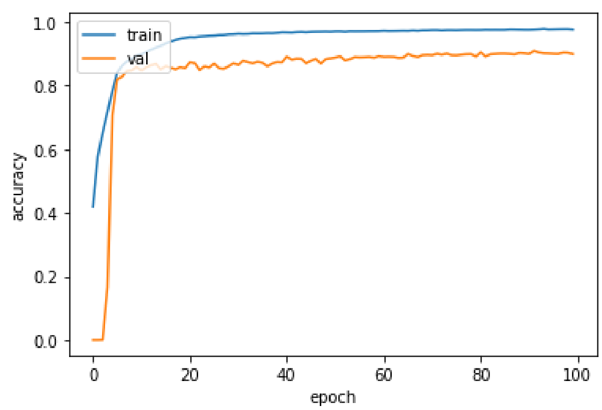



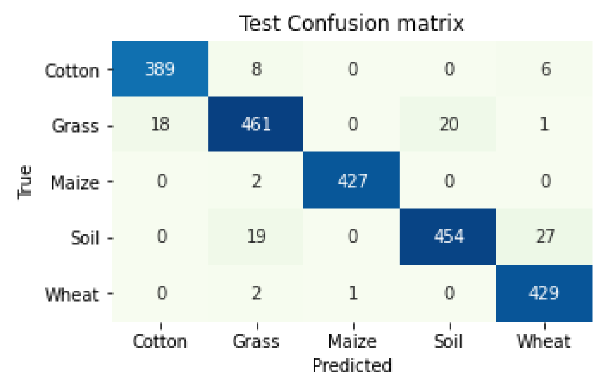

4. SGD 100 epoch, LR = 0.001 relu
Train 97.28 validation 89.3
Test 95.4



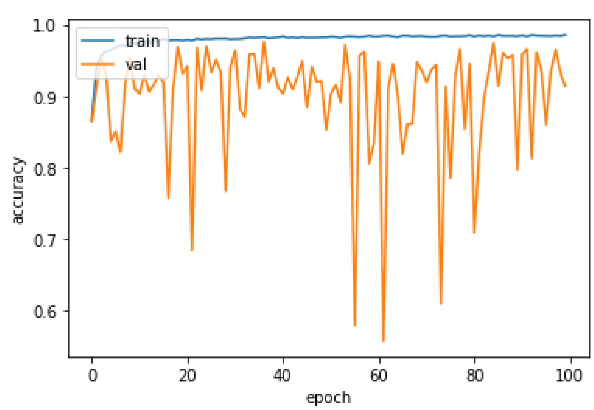



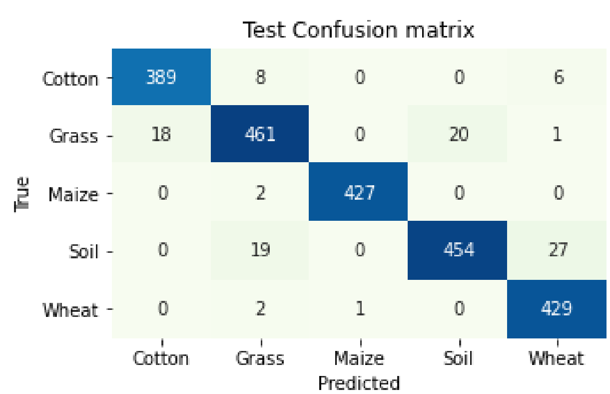

5. SGD 100 epoch, LR = 0.001 tanh
Train 97.6 validation 89.89
Test 96.33



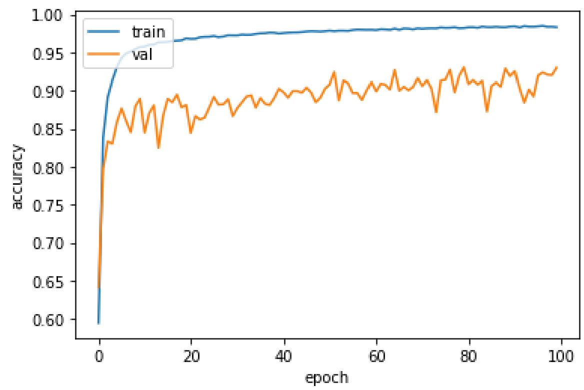



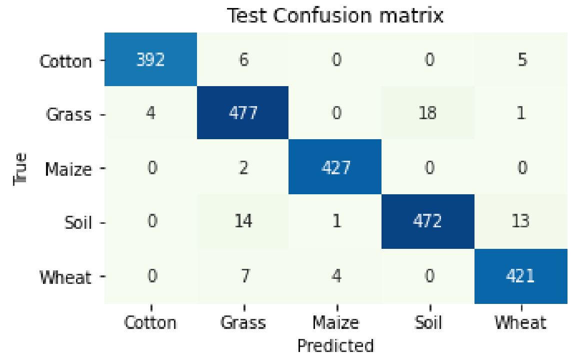

6. RMSProp 100 epoch LR = .001 tanh
Train 98.57, validation 91.41
Test 96.64



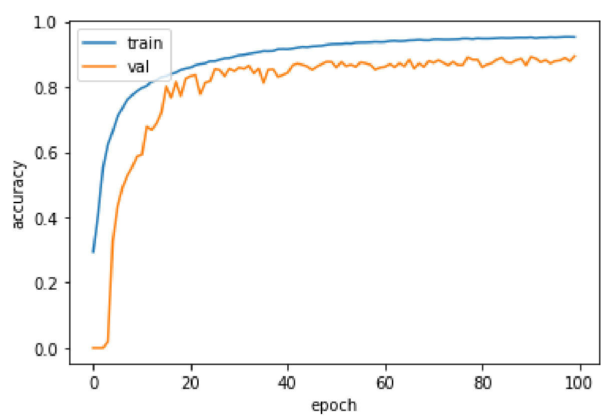
7. RMSProp 100 epoch LR = .0001 tanh
Train 98.3, validation 92.97
Test 96.68

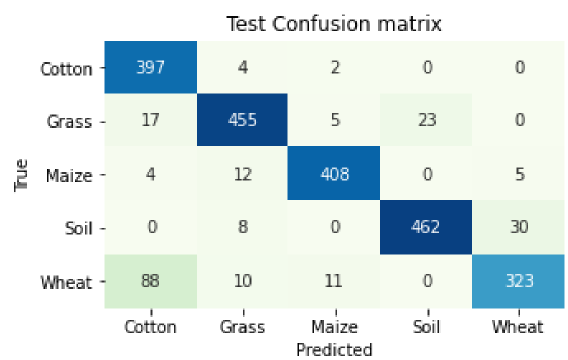



## NIR 5 Class

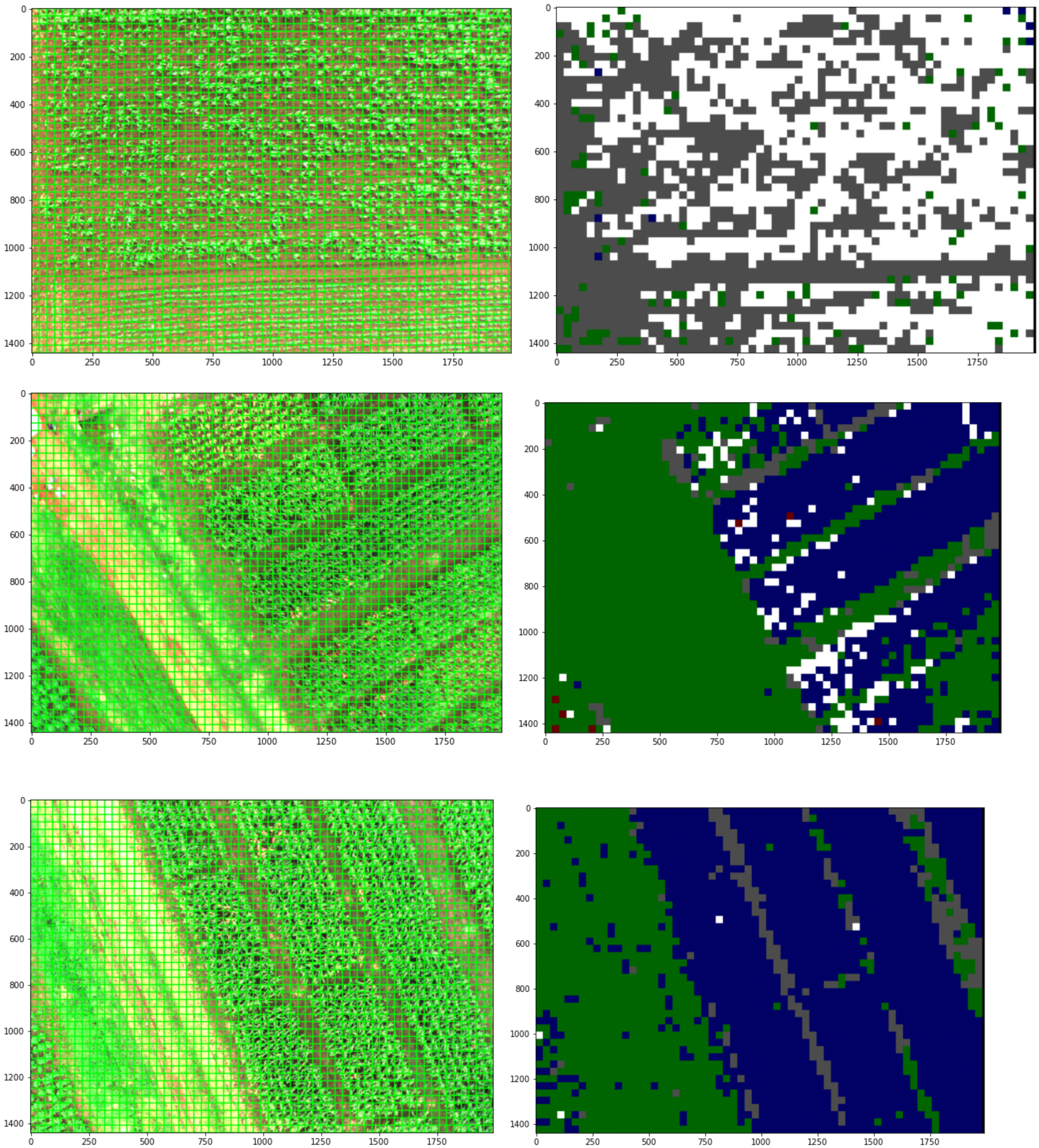
1. SGD 100 epochs, LR 0.001 tanh
Train 95.3 validation 89.4
Test 90.3

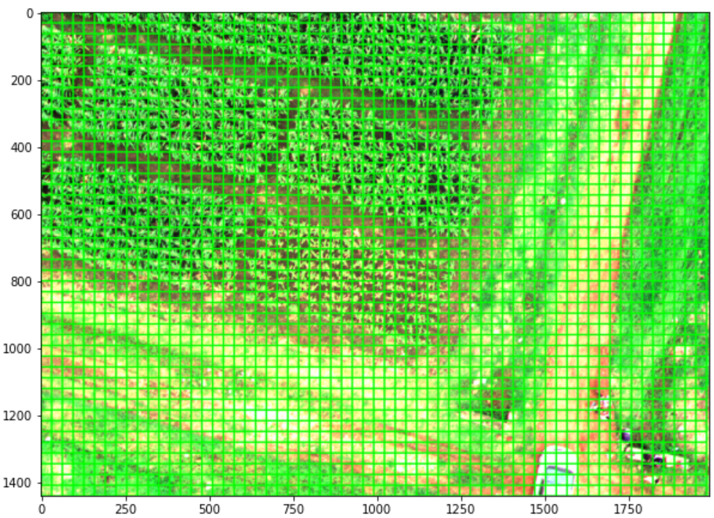


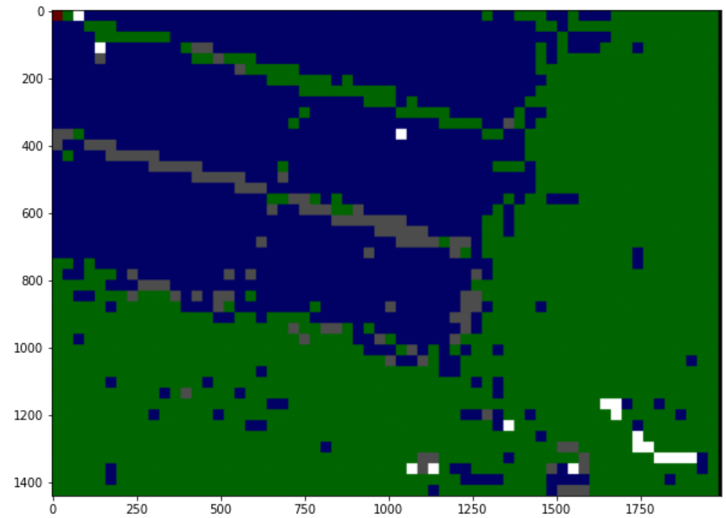



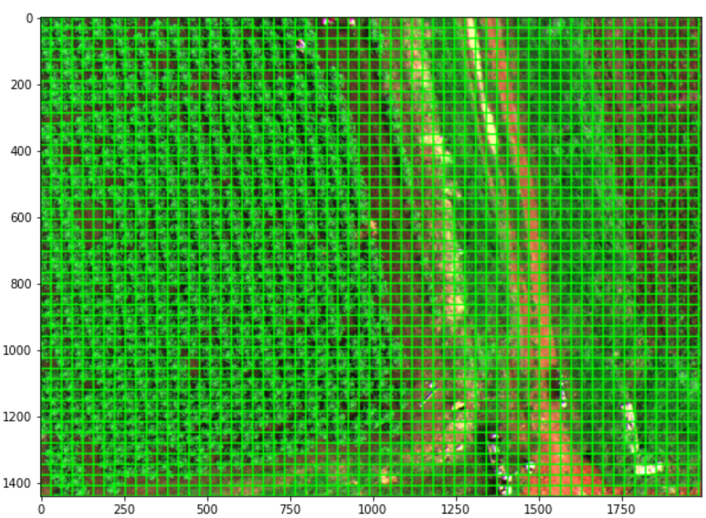

## Performance

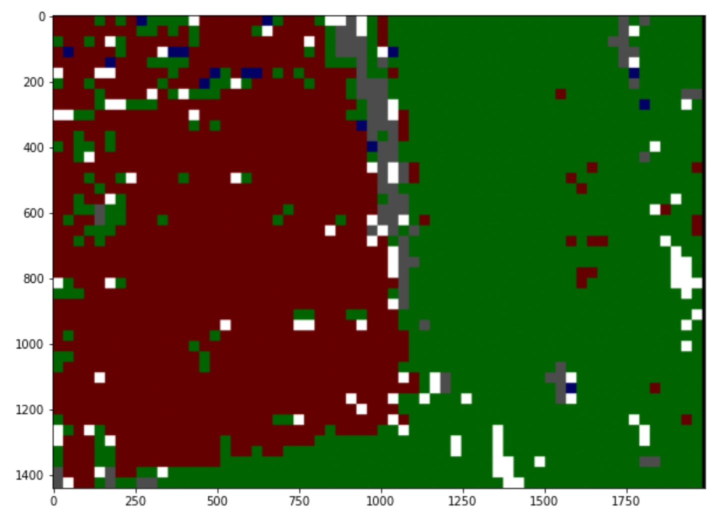
## 5 Band

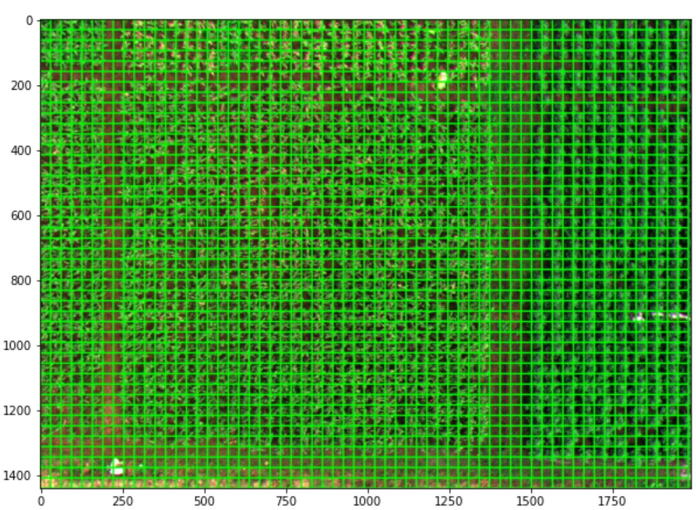

0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750

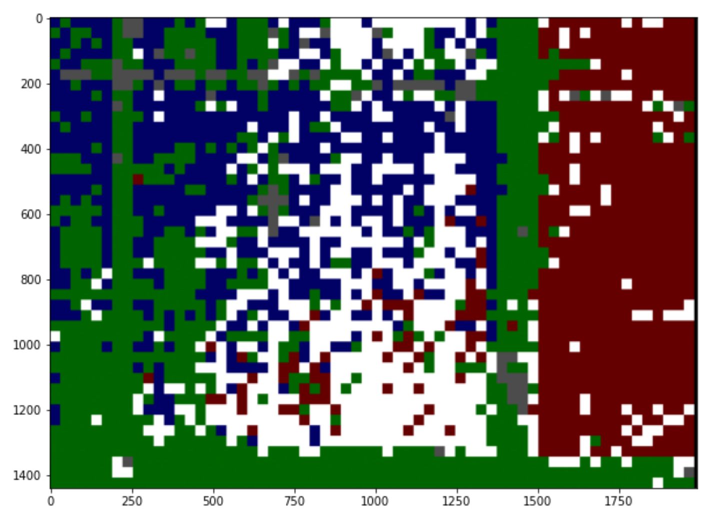

0
200
400
600
800
1000
1200
1400
0 250 500 750 1000 1250 1500 1750

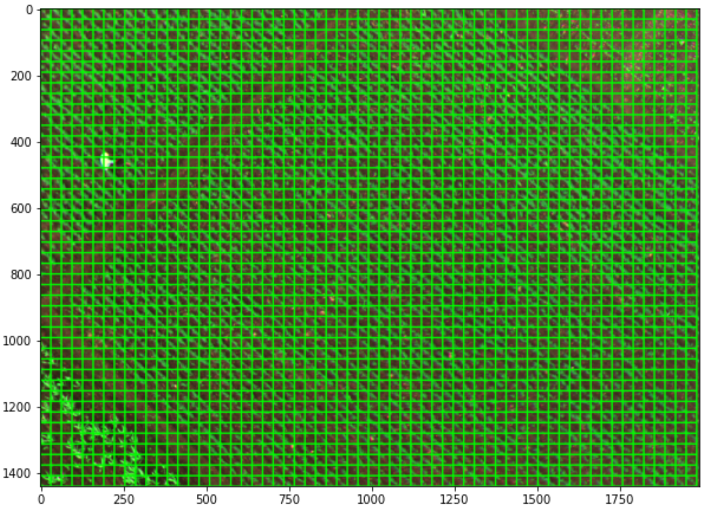

0
200
400
600
800
1000
1200
1400
0 250 500 750 1000 1250 1500 1750

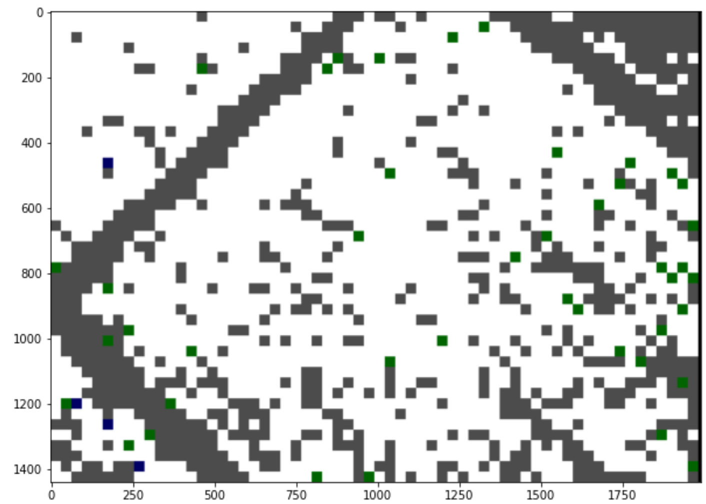

0
200
400
600
800
1000
1200
1400
0 250 500 750 1000 1250 1500 1750


0
200
400
600
800
1000
1200
1400
0 250 500 750 1000 1250 1500 1750

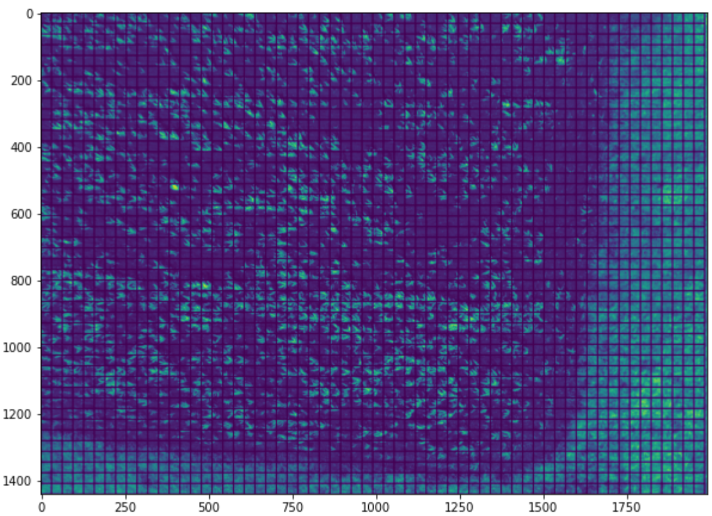

0
200
400
600
800
1000
1200
1400
0 250 500 750 1000 1250 1500 1750

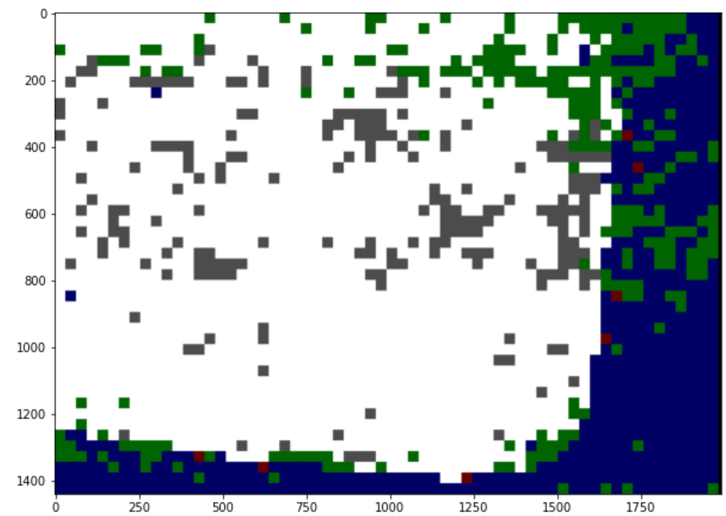

0
200
400
600
800
1000
1200
1400
0 250 500 750 1000 1250 1500 1750

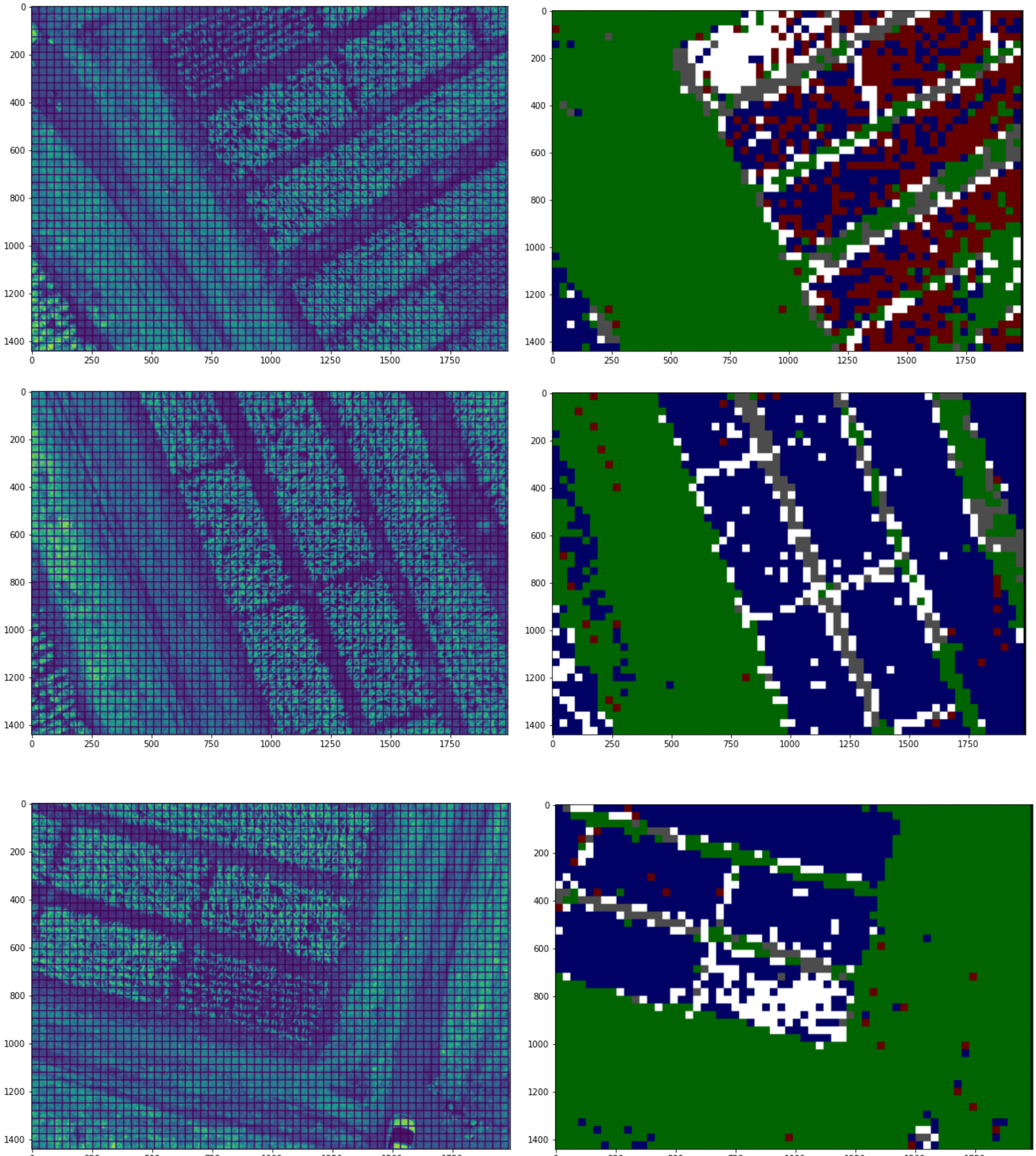


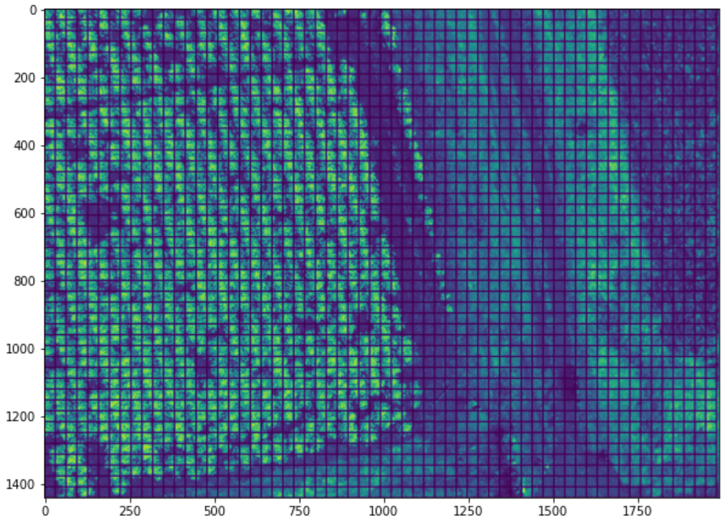
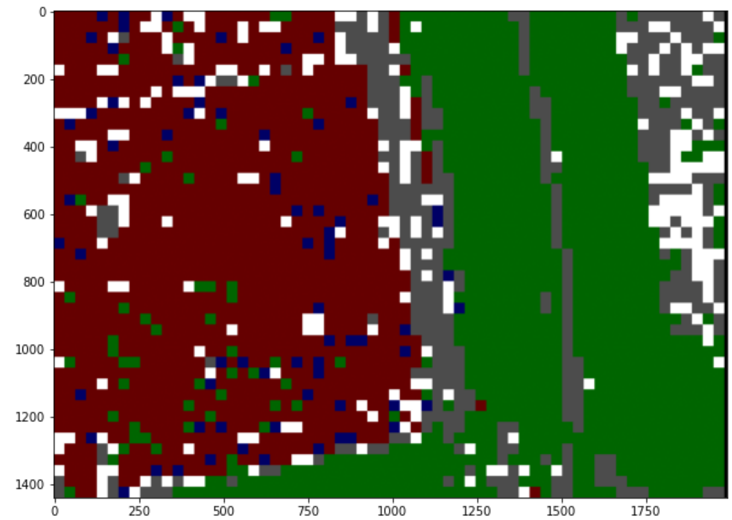
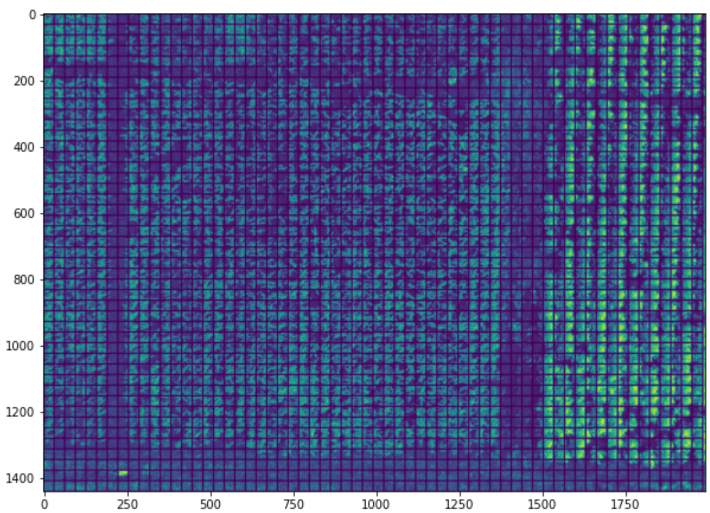
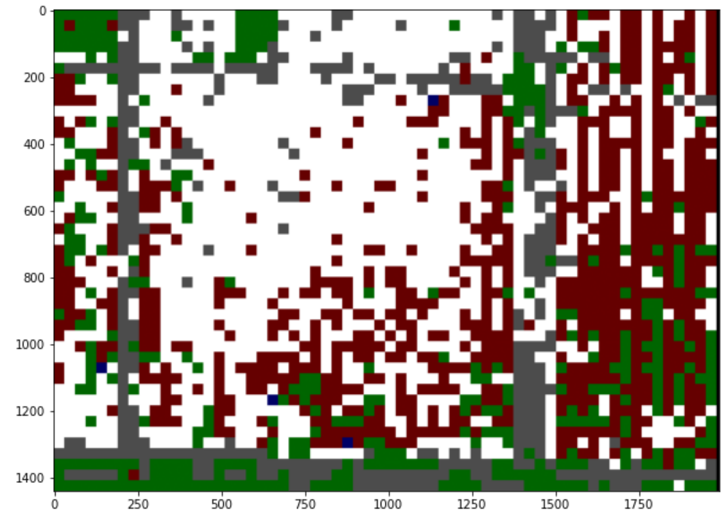
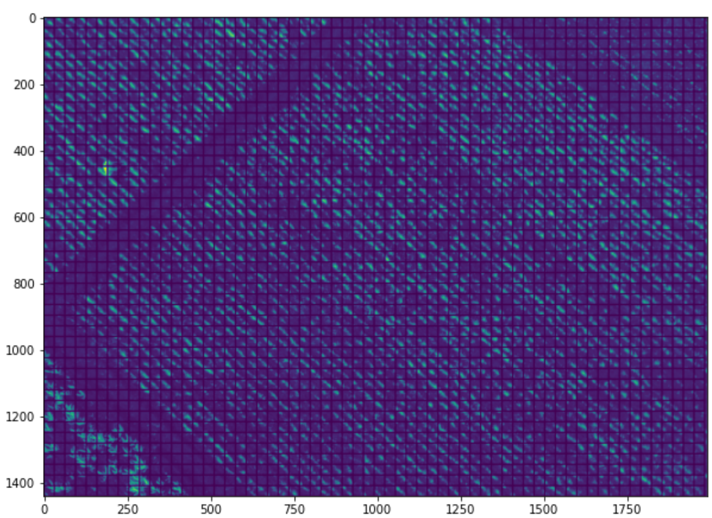
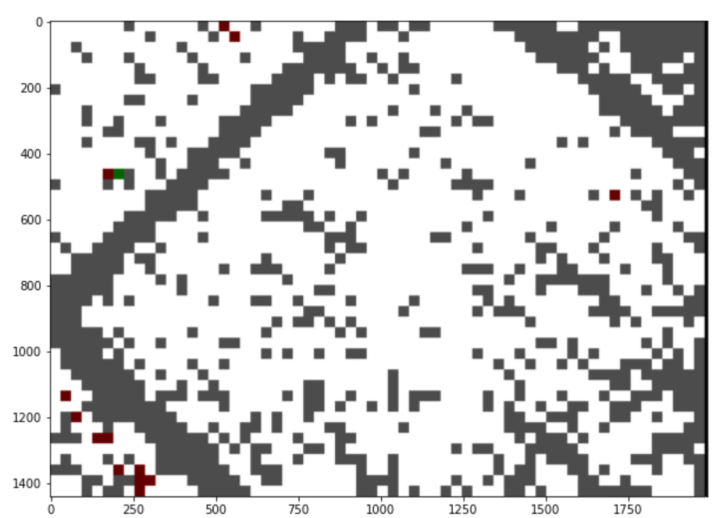




Only NIR Band

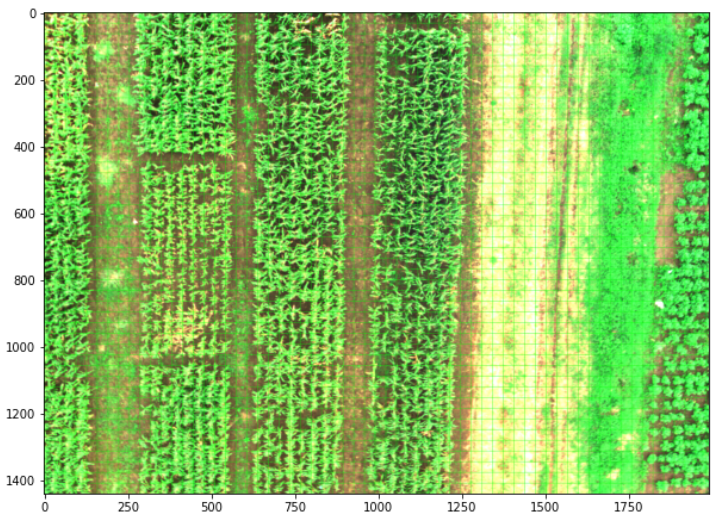


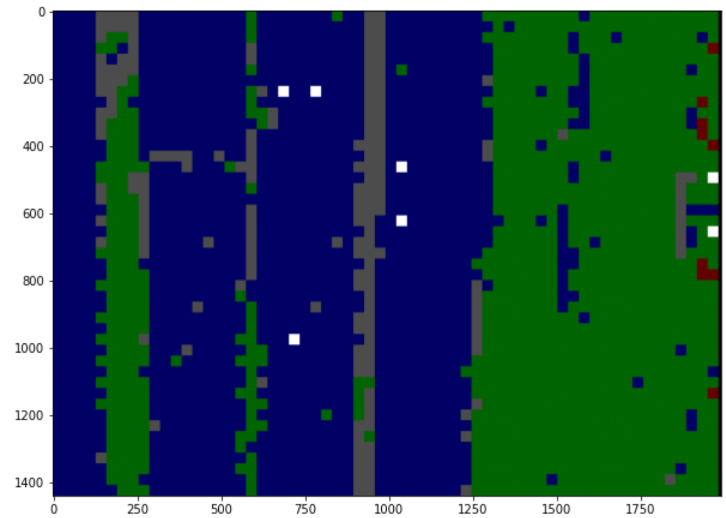



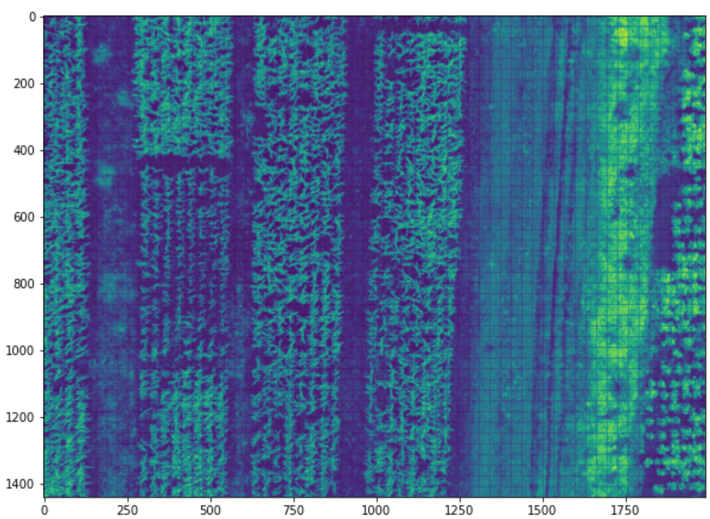

0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750

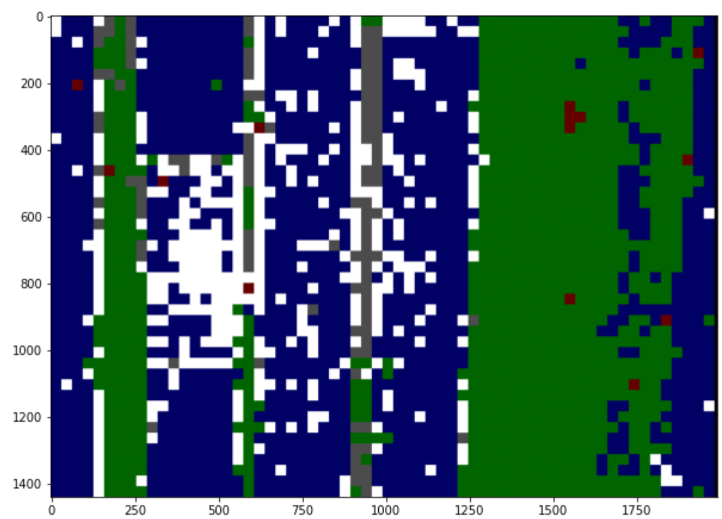

0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750

0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750

0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750

0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750

0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750

0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750

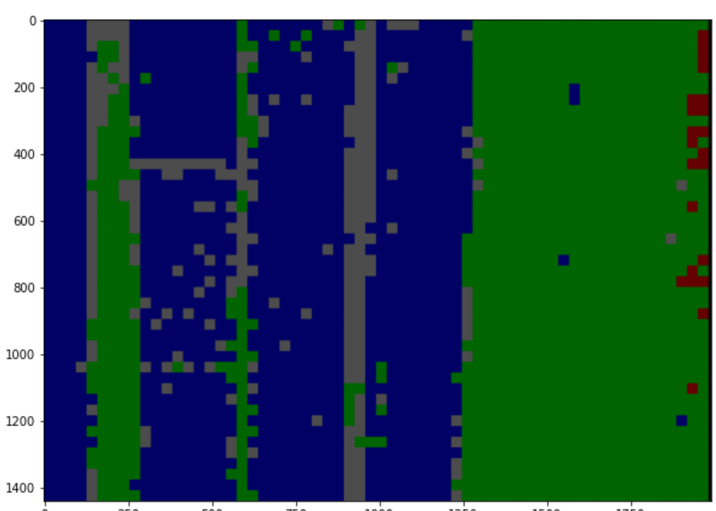
RMSprop model (96.67) vs NIR vs Model without Wheat data:

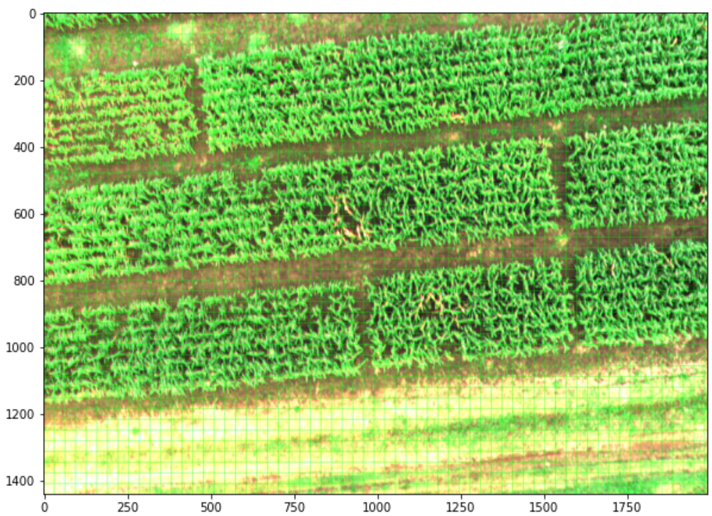


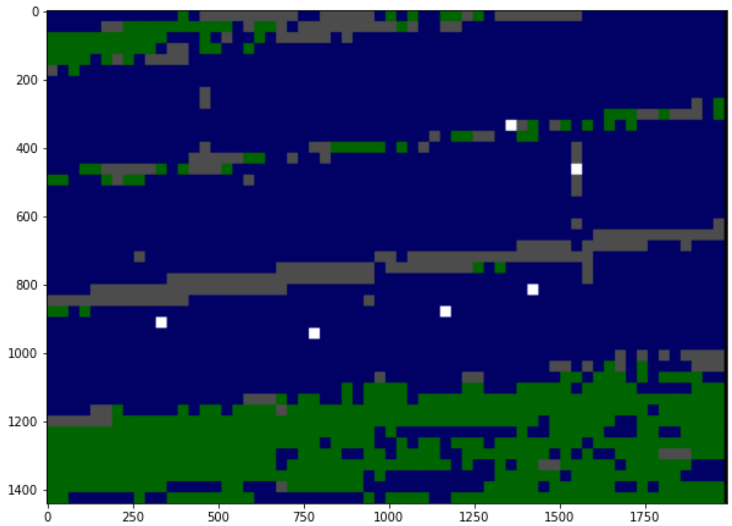



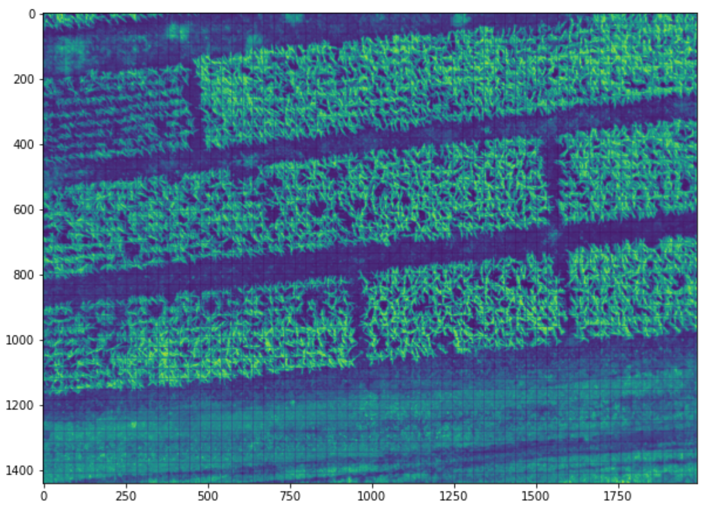



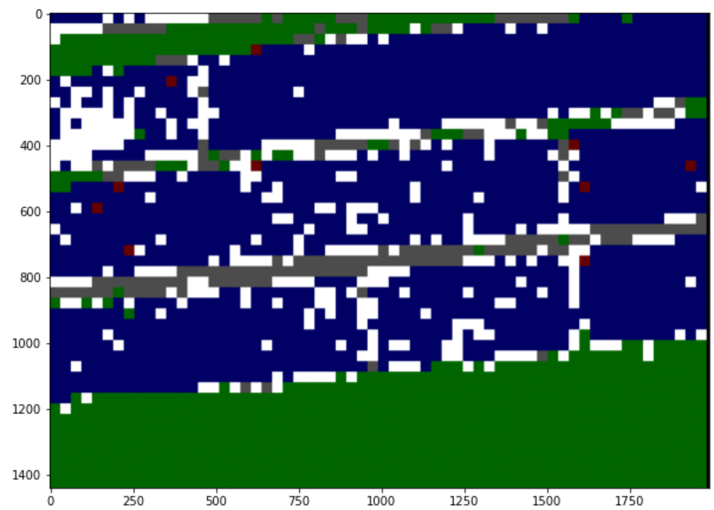



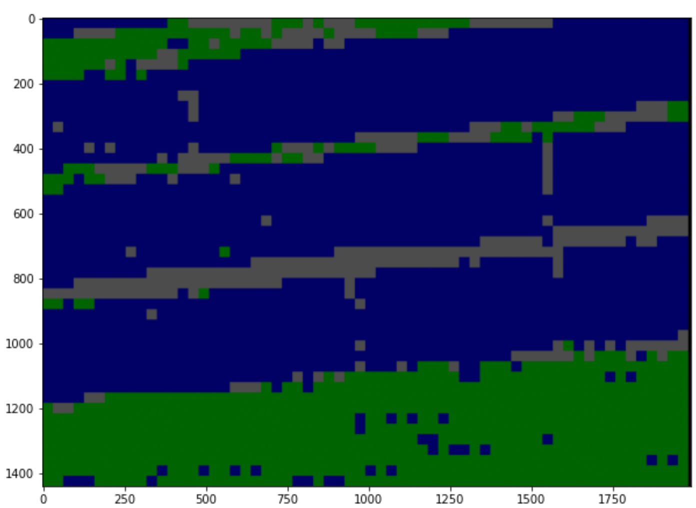



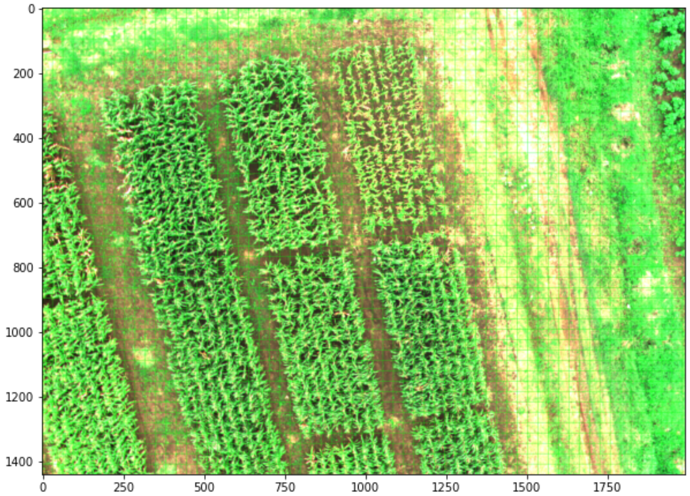

0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750

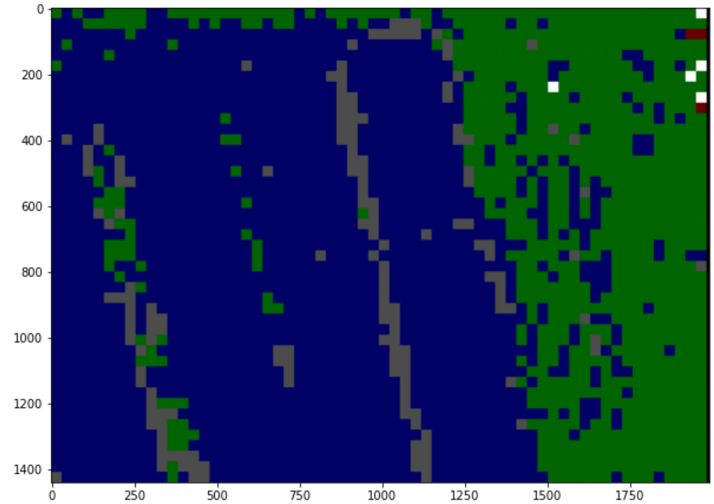

0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750

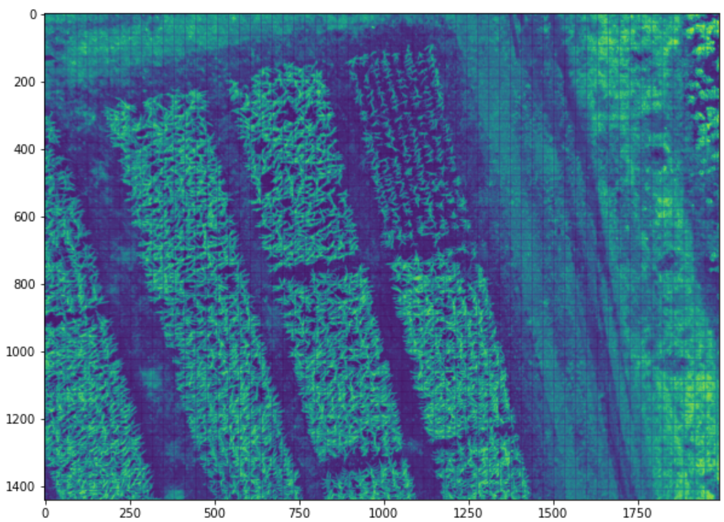

0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750

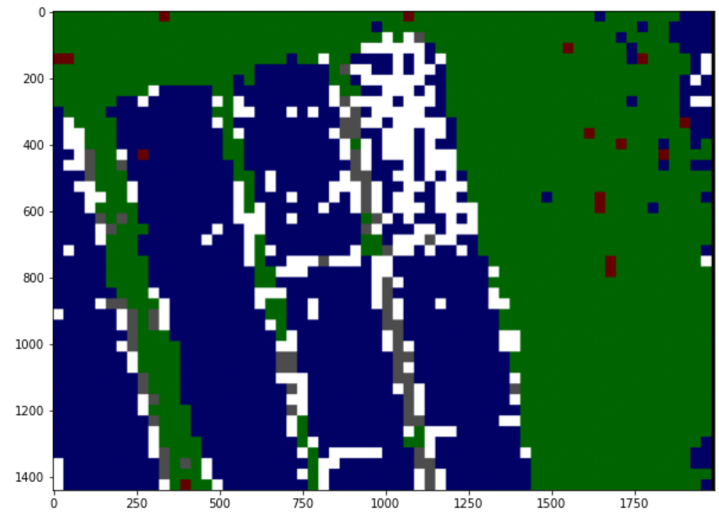

0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750

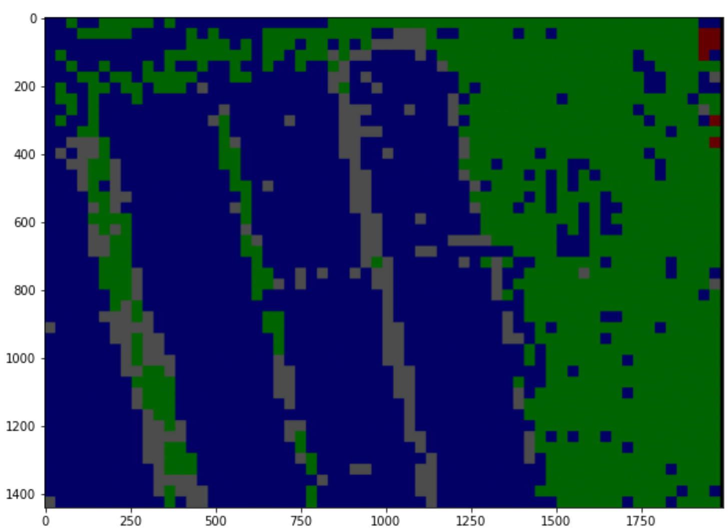

0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750


0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750

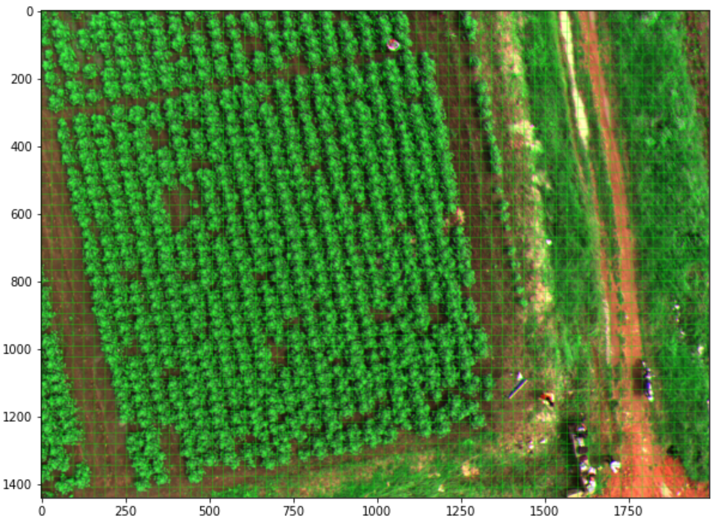

0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750

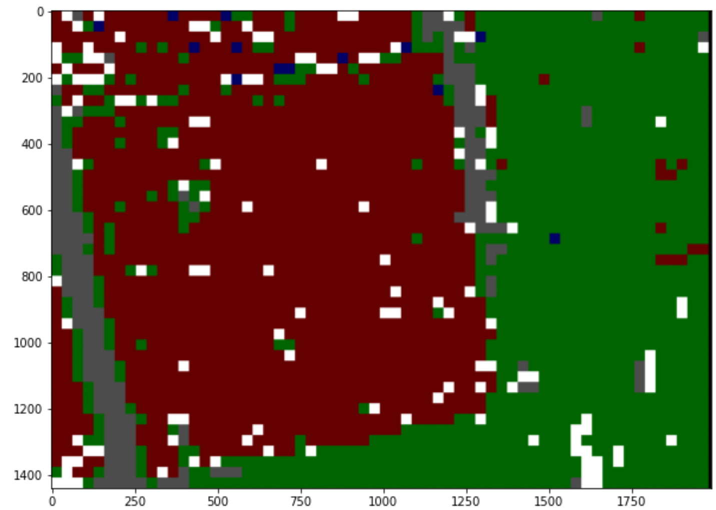

0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750

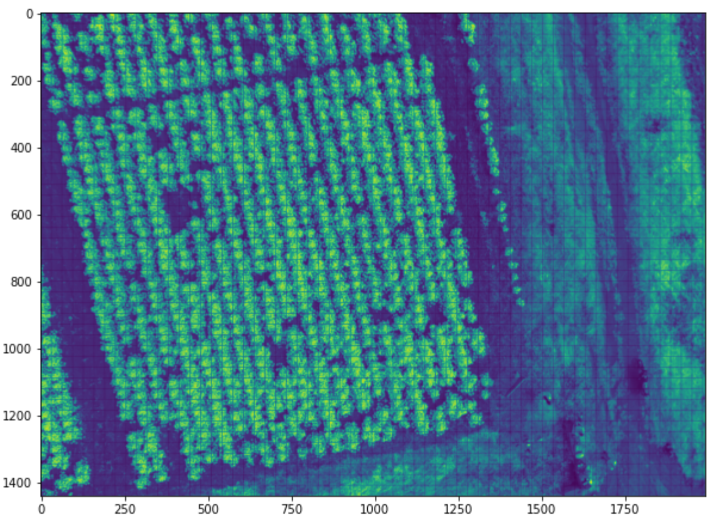

0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750

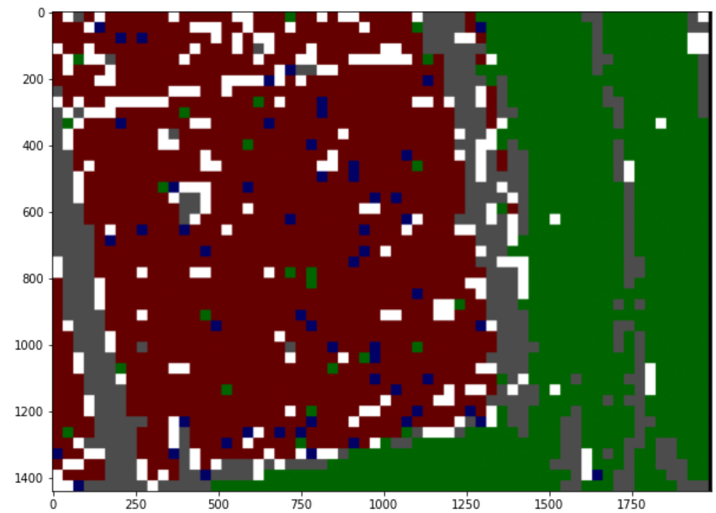

0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750

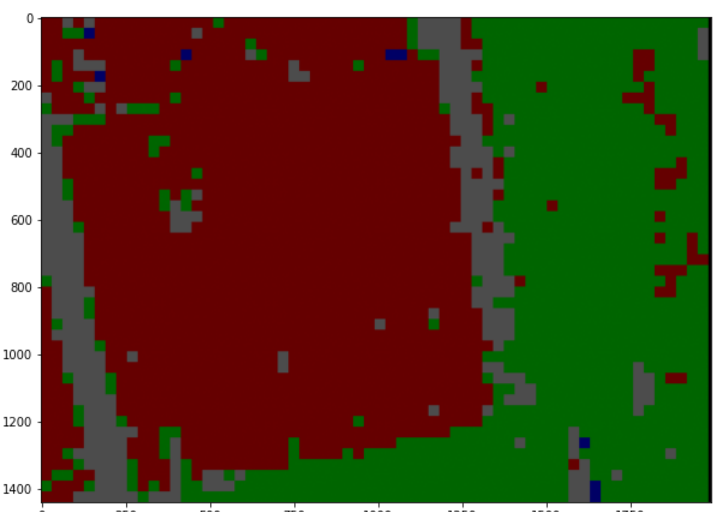
Cotton:

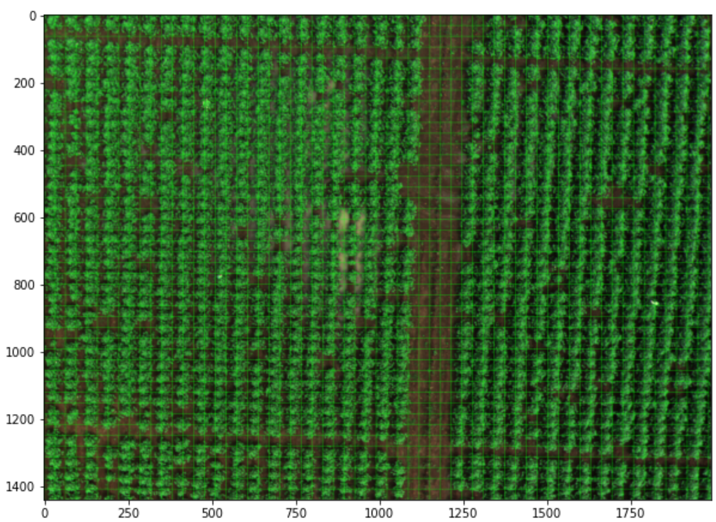
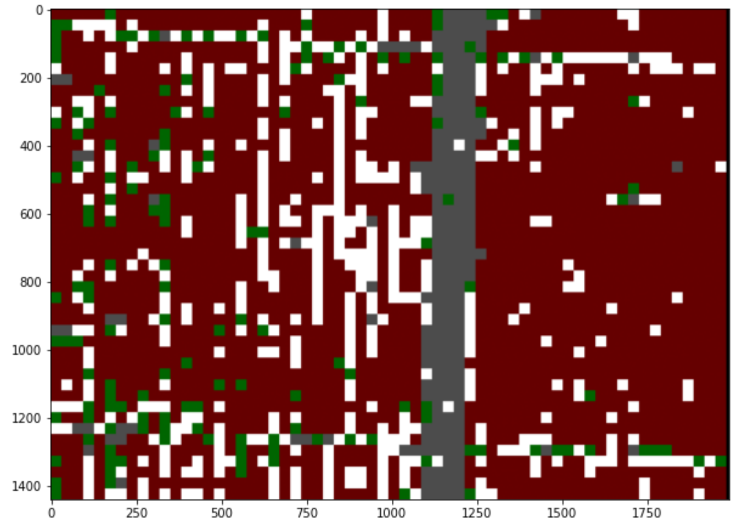
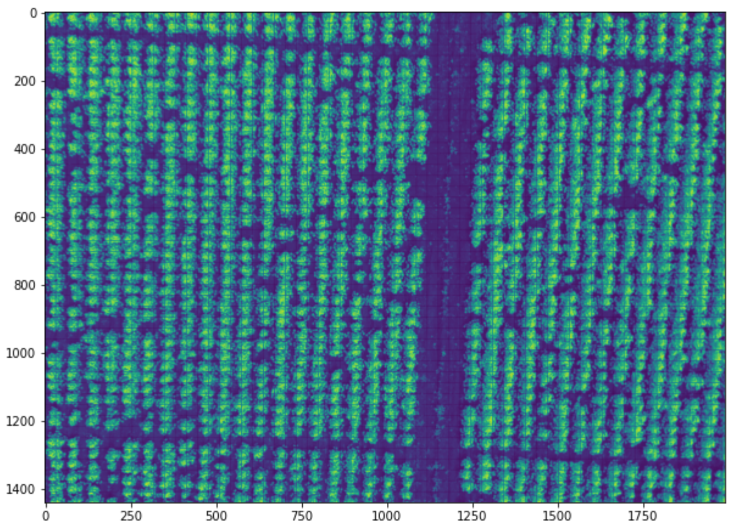
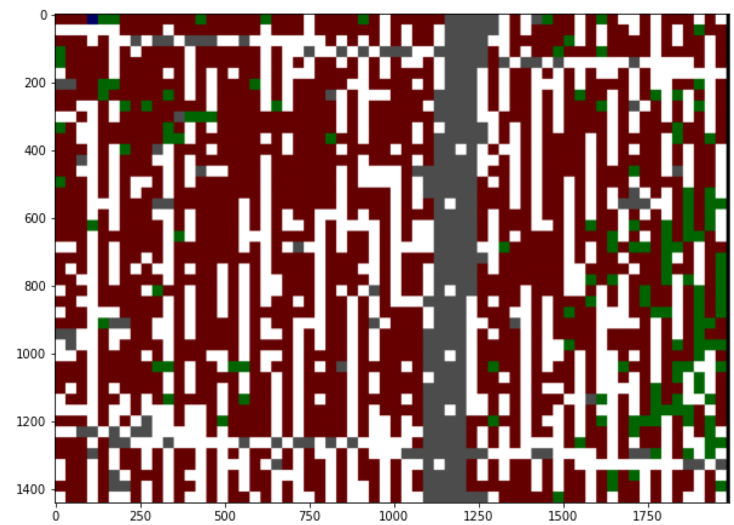
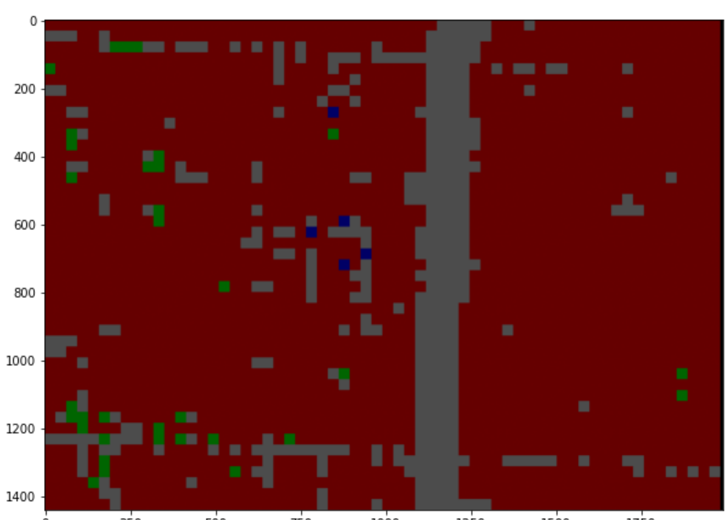

0
200
400
600
800
1000
1200
1400
0 250 500 750 1000 1250 1500 1750

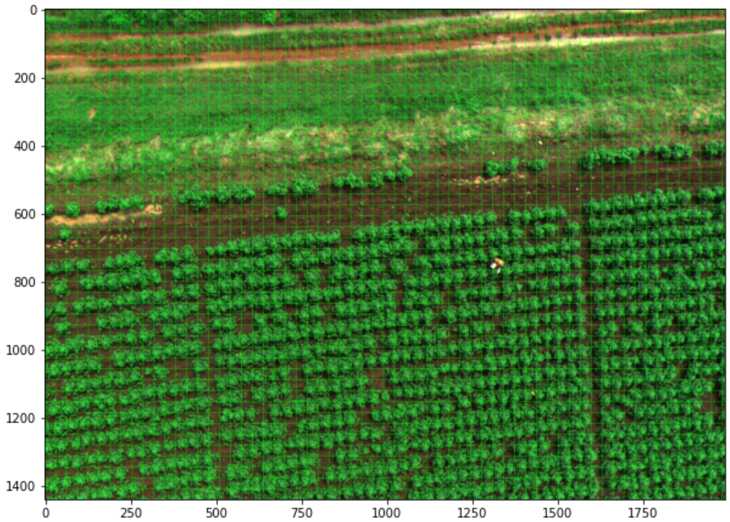
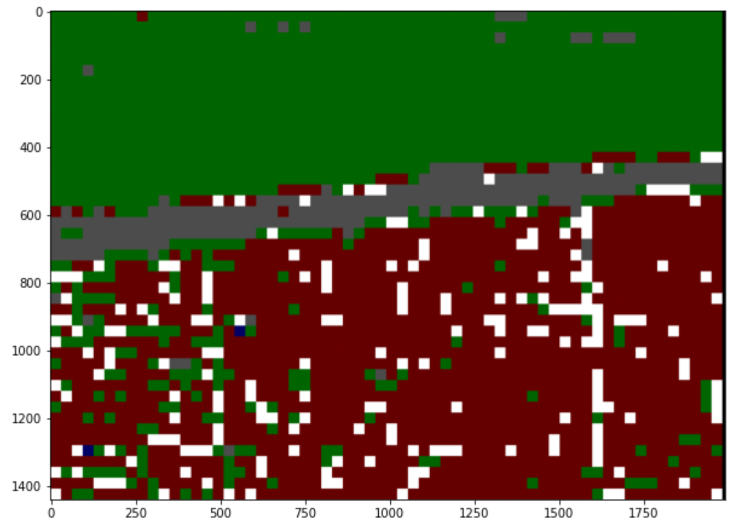
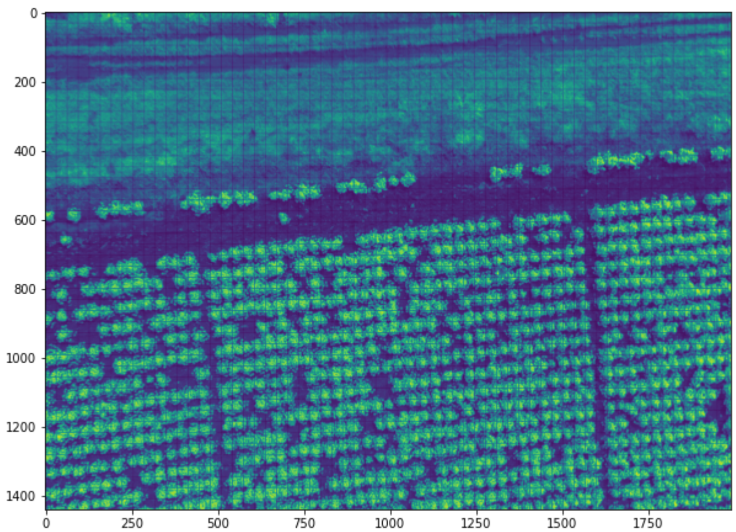
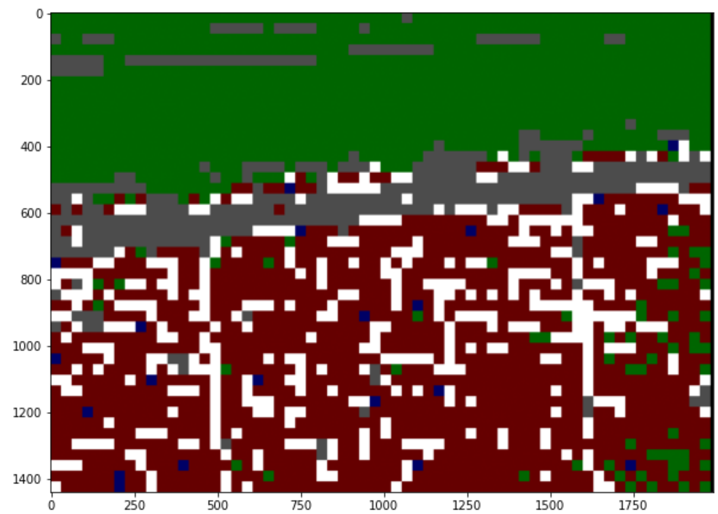
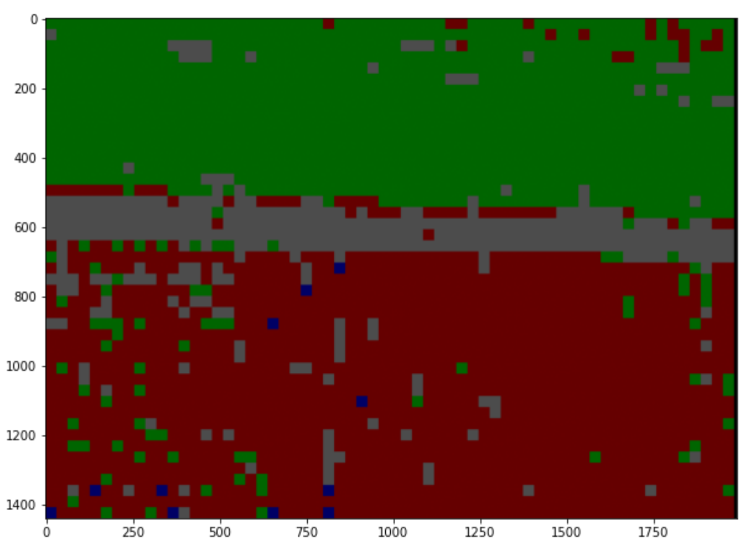

0
200
400
600
800
1000
1200
1400
0 250 500 750 1000 1250 1500 1750


0
200
400
600
800
1000
1200
1400
0 250 500 750 1000 1250 1500 1750

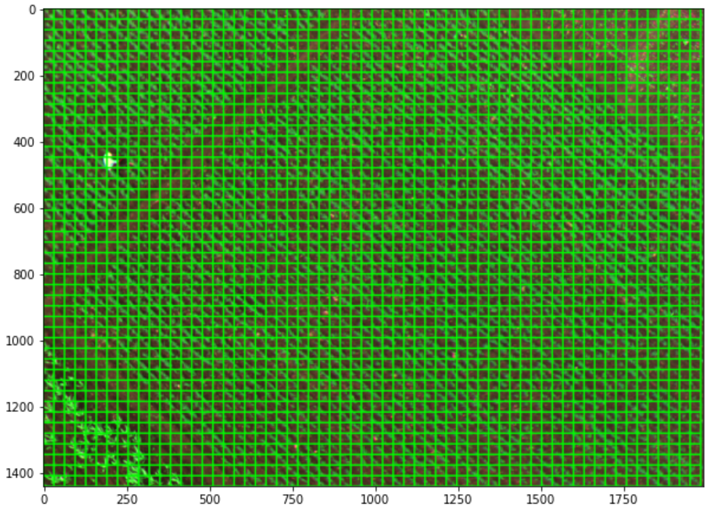

0
200
400
600
800
1000
1200
1400
0 250 500 750 1000 1250 1500 1750

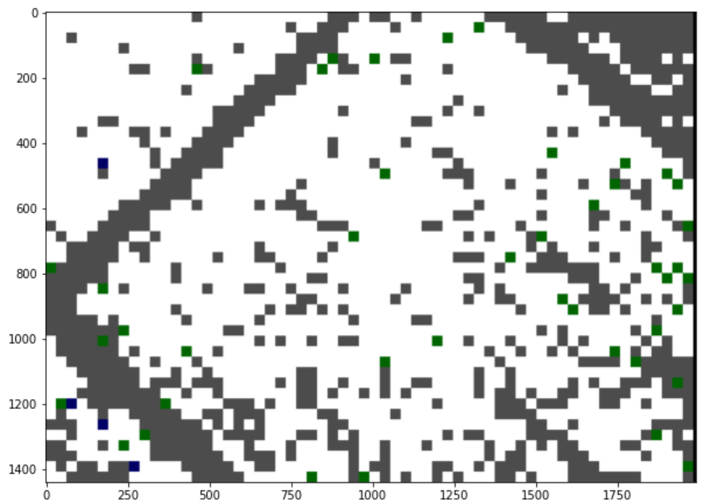

0
200
400
600
800
1000
1200
1400
0 250 500 750 1000 1250 1500 1750

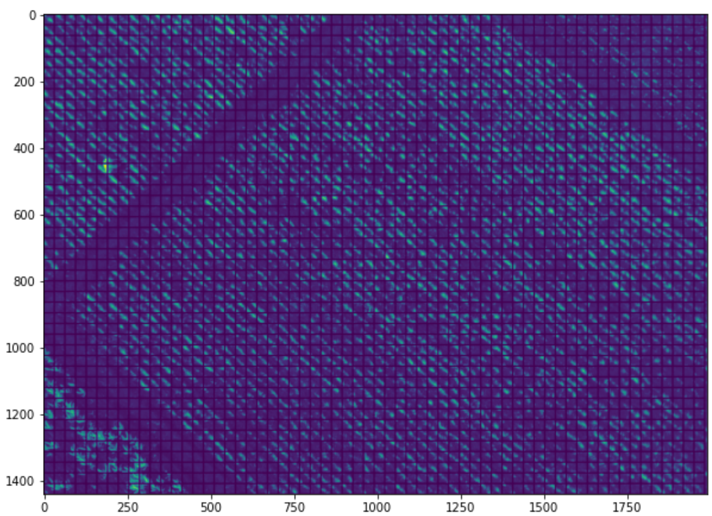

0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750

0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750

0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750

0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750

0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750

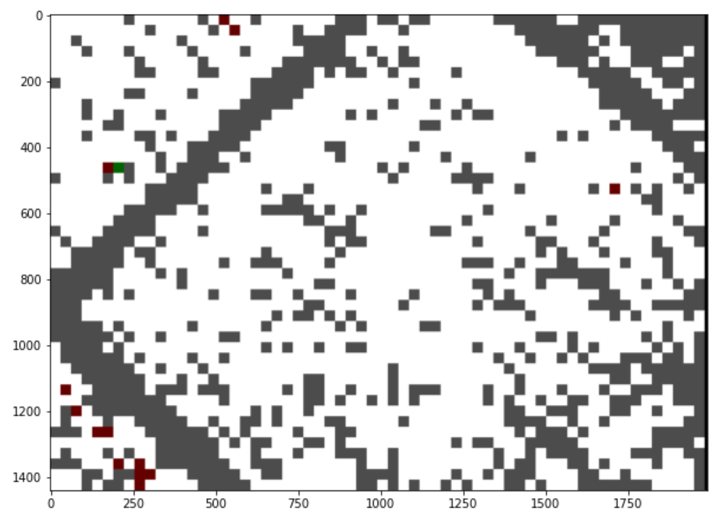

0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750

0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750

0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750

0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750

0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750

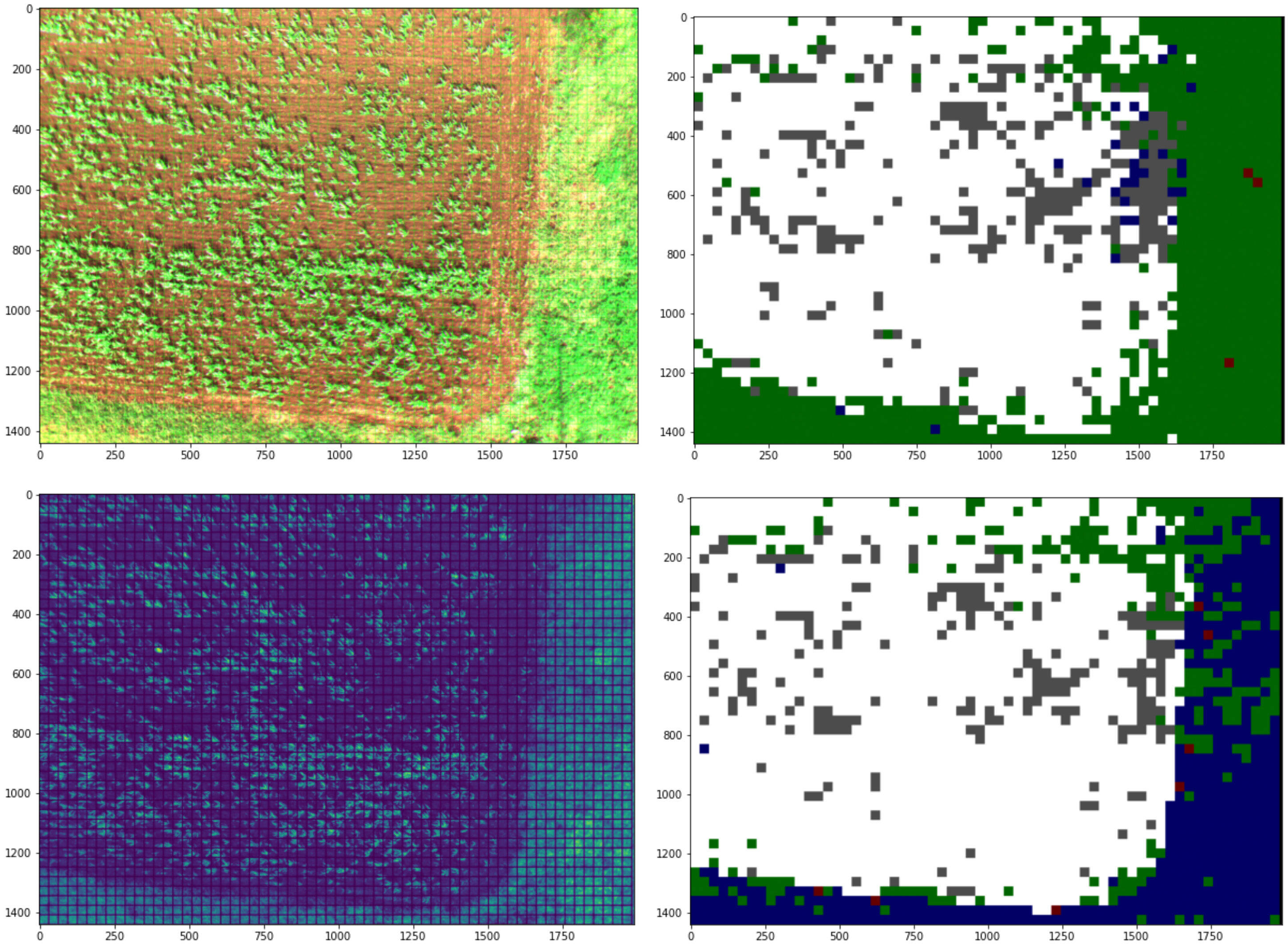
# Wheat

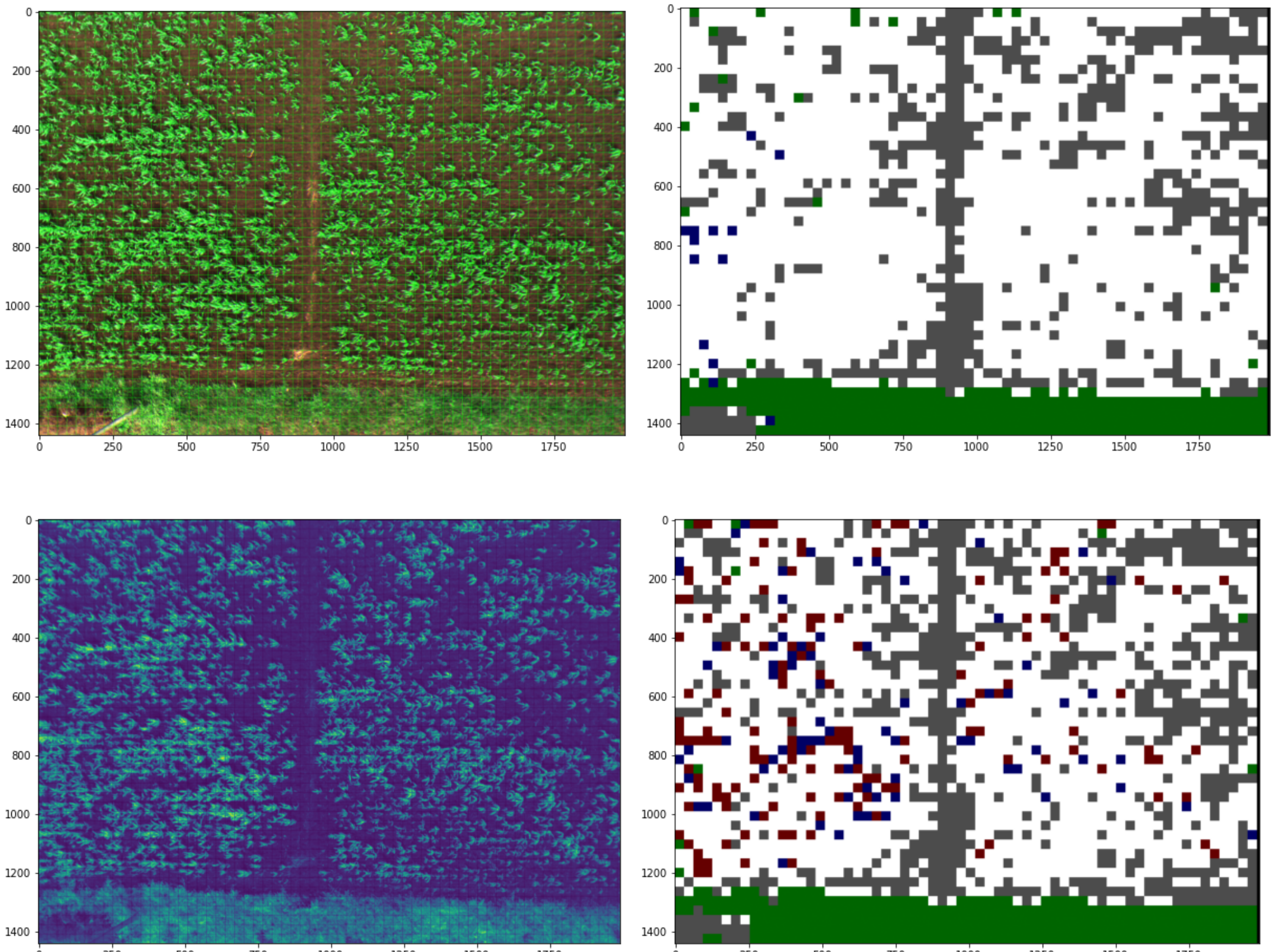

0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750

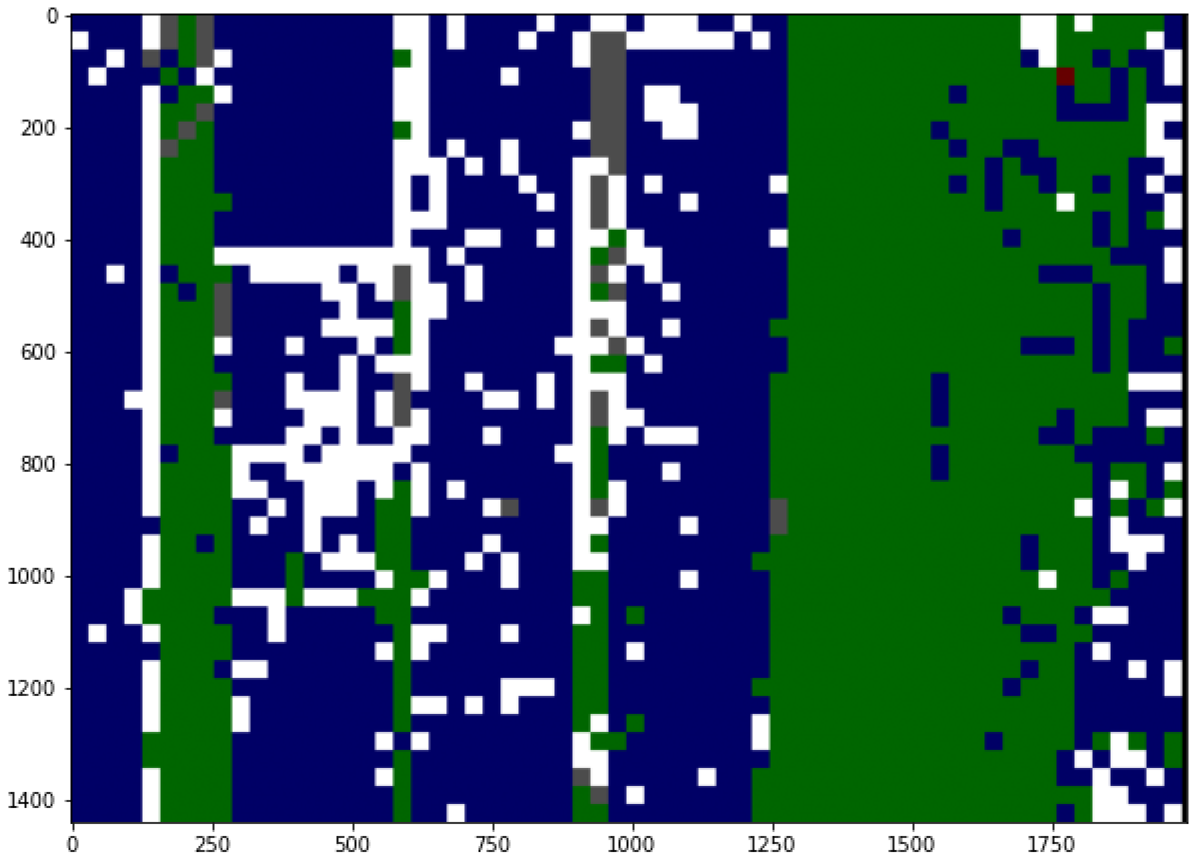

0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750


0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750

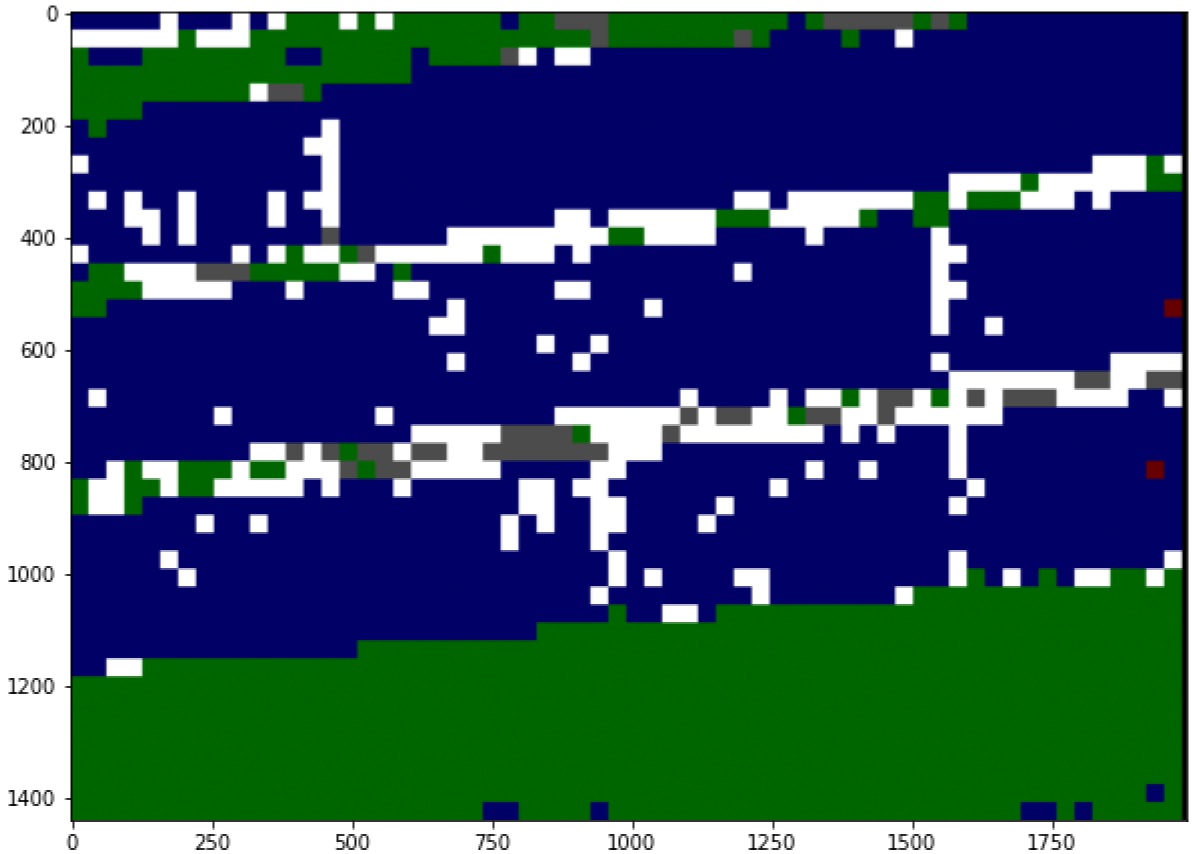

0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750

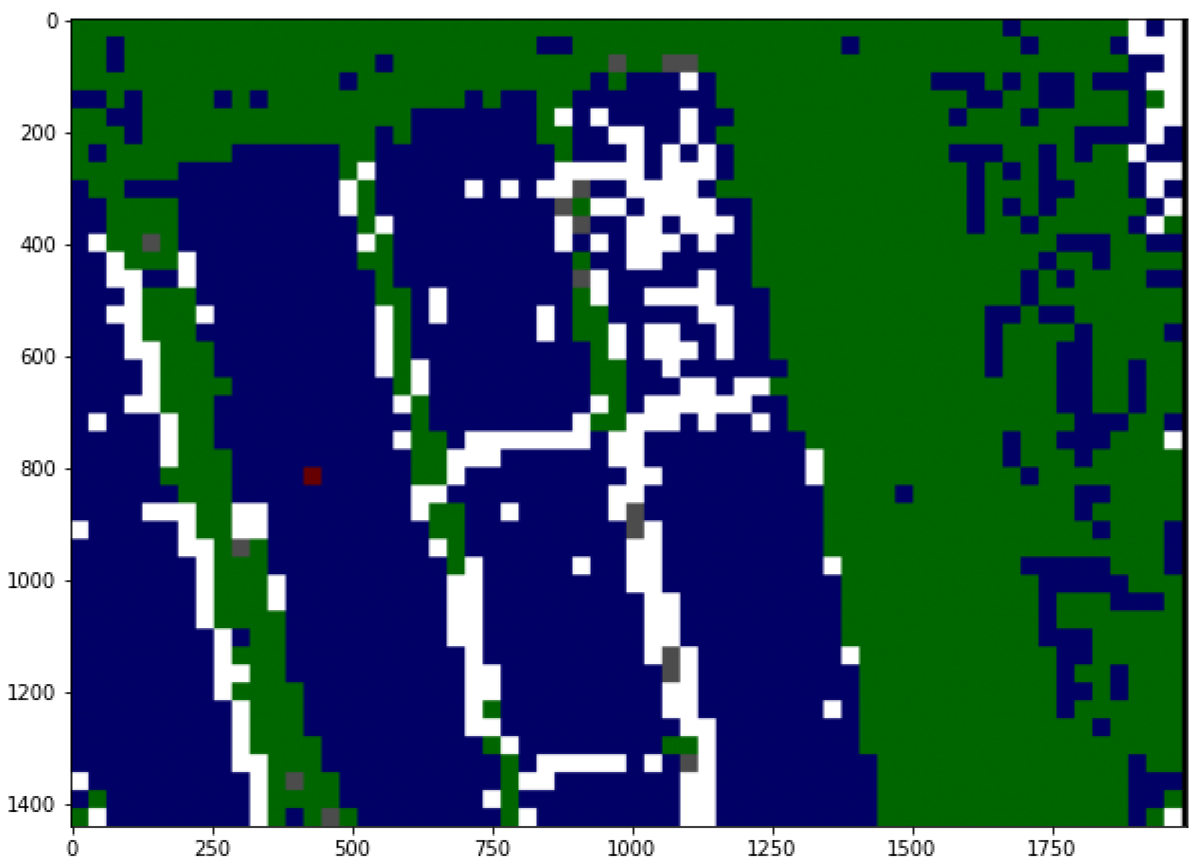

0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750
0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750
0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750
0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750

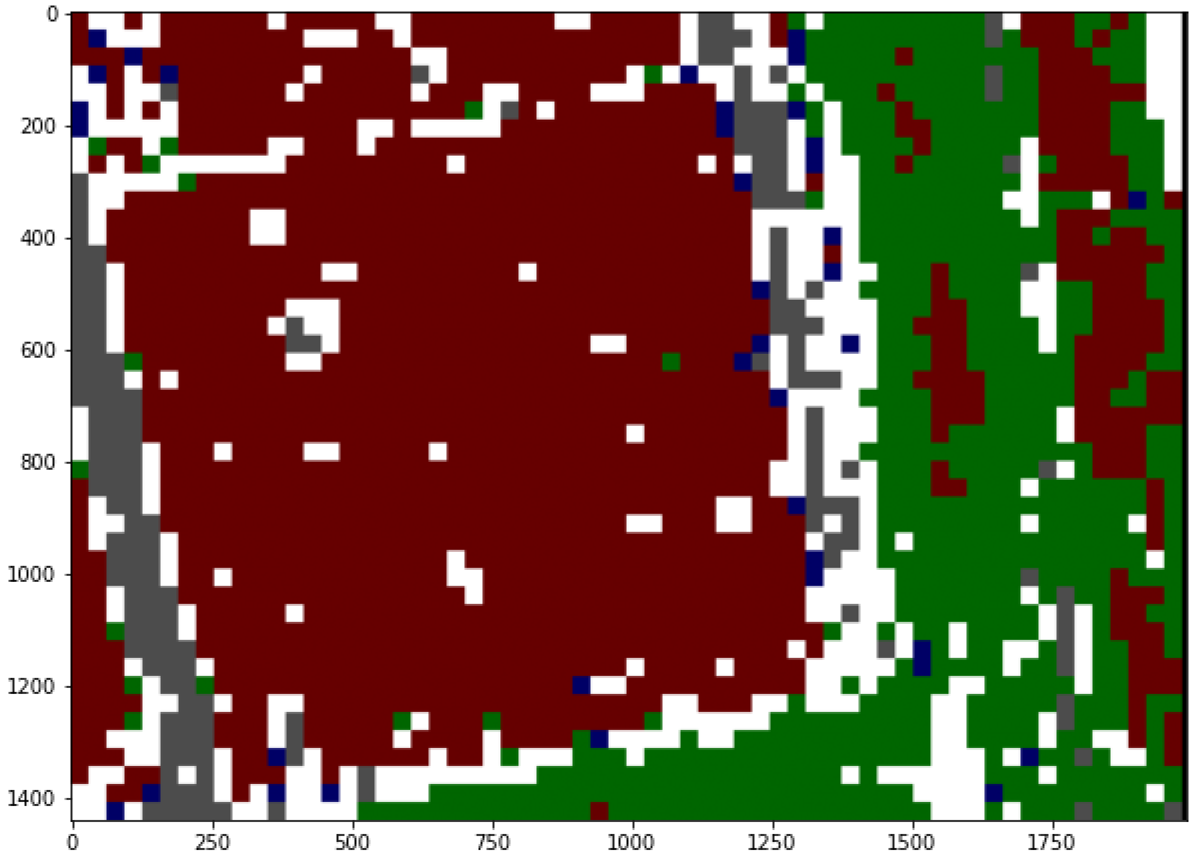

0
200
400
600
800
1000
1200
1400
0 250 500 750 1000 1250 1500 1750



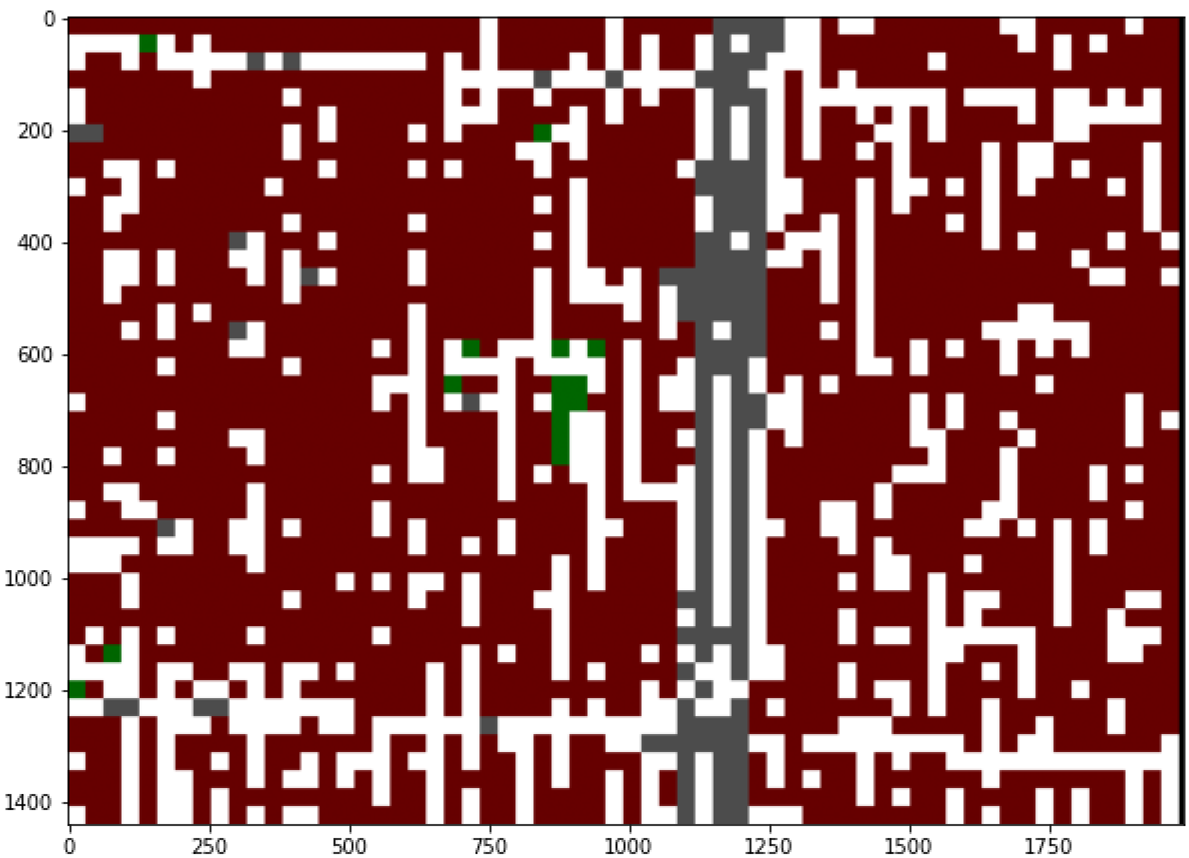
0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750

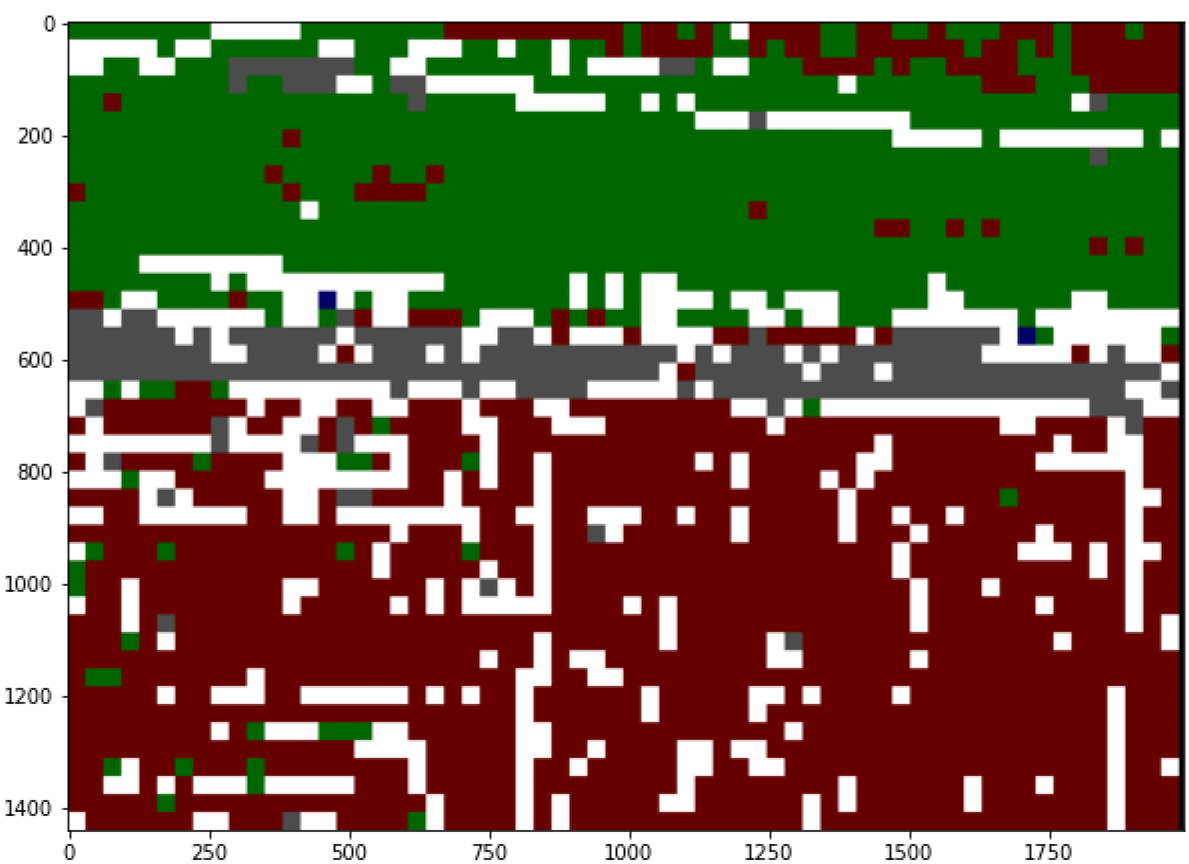

0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750

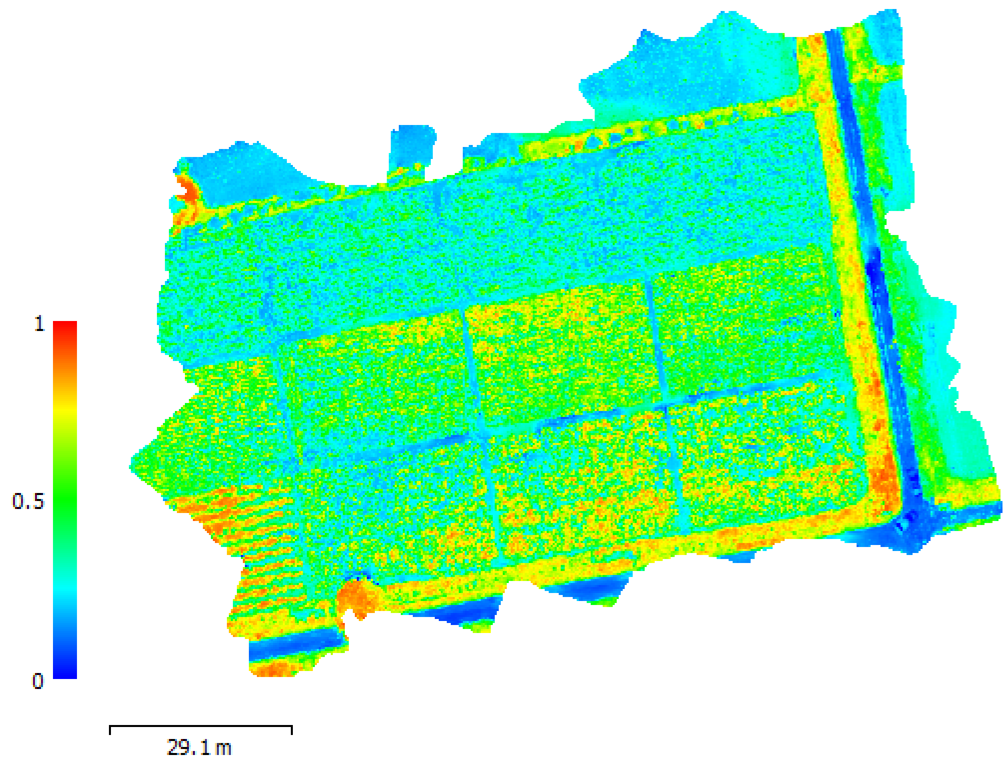

0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750

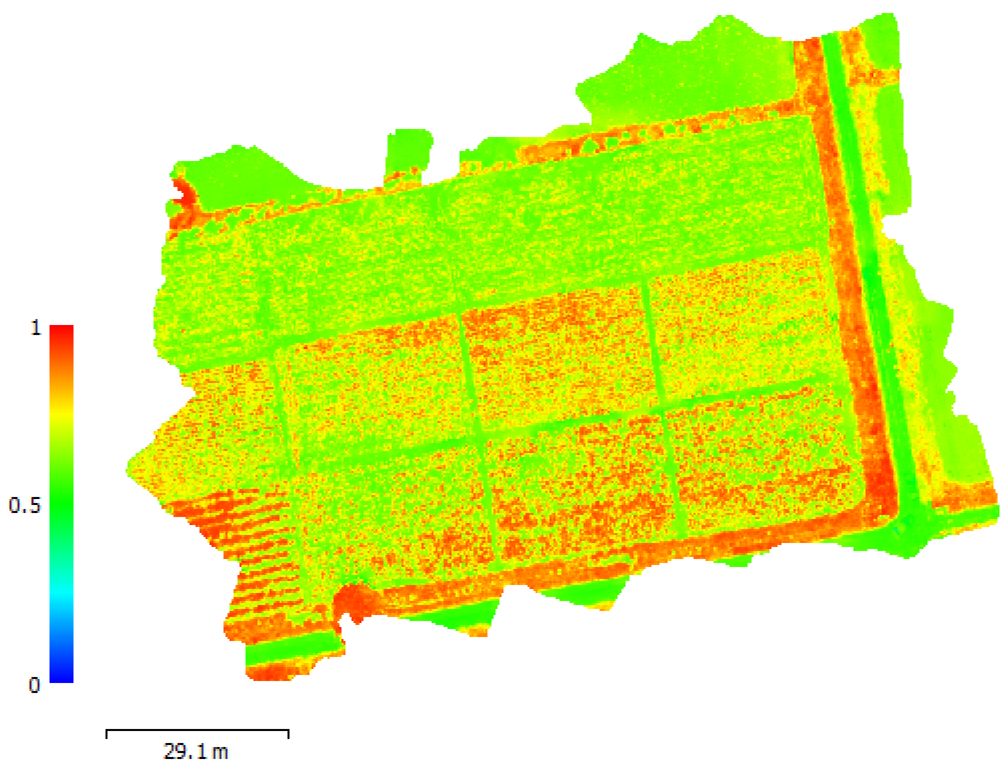


0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750


0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750


0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750


1
0.5
0
29.1 m


1
0.5
0
29.1 m