Corn Plant Segmentation

A DEEP LEARNING PROJECT FOR AGRICULTURAL PURPOSES

By **Iliya Rubinchik** and **Snir Homsky – 29/03/20**

Corn segmentation project

- We will be using Mask R-CNN, a segmentation neural network, which outperforms the COCO 2016 challenge winners.
- The challenge in this project, is to take a small dataset, of just 45 annotated images, and by using Data Augmentation, create a large dataset, which the neural network can train on.
- Another challenge is, that each image contains plants which are annotated, and plants which are not.

Corn segmentation project

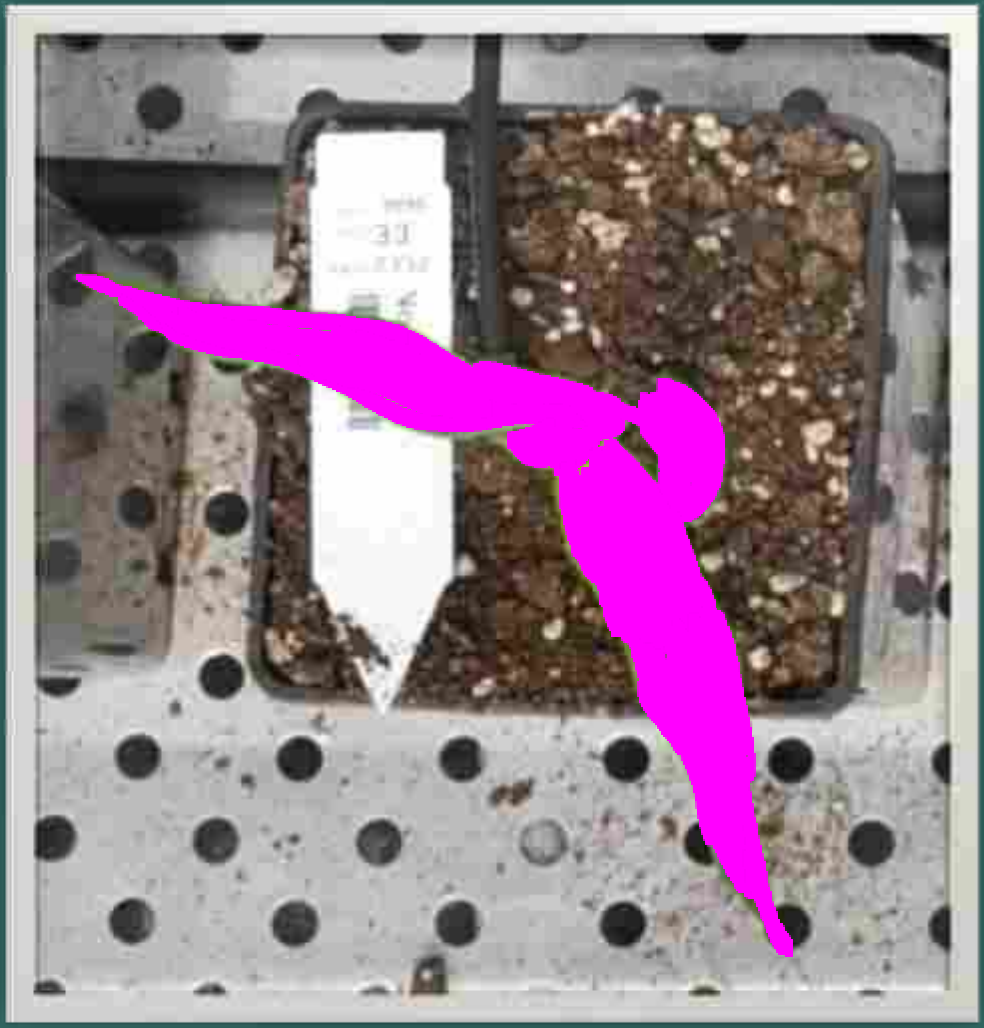
- We will be using Mask R-CNN, a segmentation neural network, which outperforms the COCO 2016 challenge winners.
- The challenge in this project, is to take a small dataset, of just 45 annotated images, and by using Data Augmentation, create a large dataset, which the neural network can train on.
- Another challenge is, that each image contains plants which are annotated, and plants which are not.

Project Goals

Project Goals

- Segmentation of corn leaves
  - Using the Mask R-CNN neural network

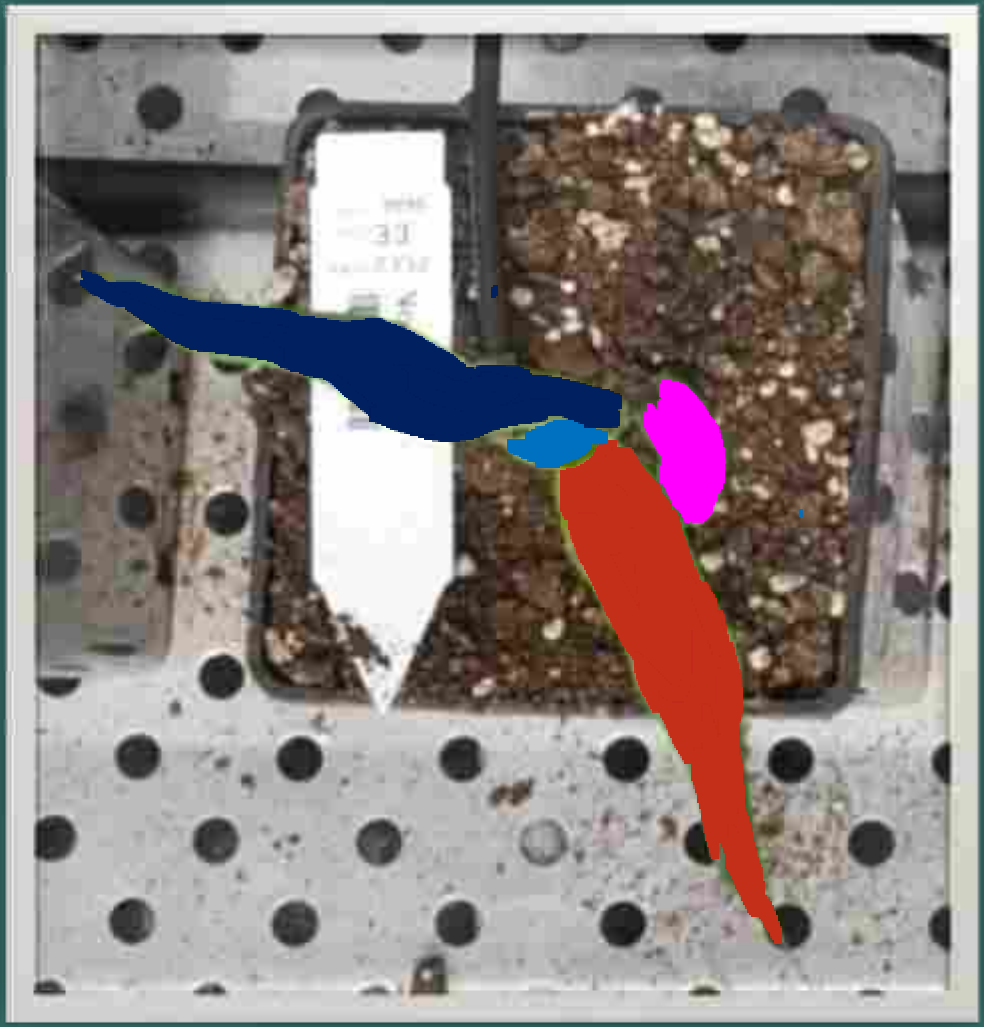
Example only:



Project Goals

- Segmentation of corn leaves
- Separate between different leaves

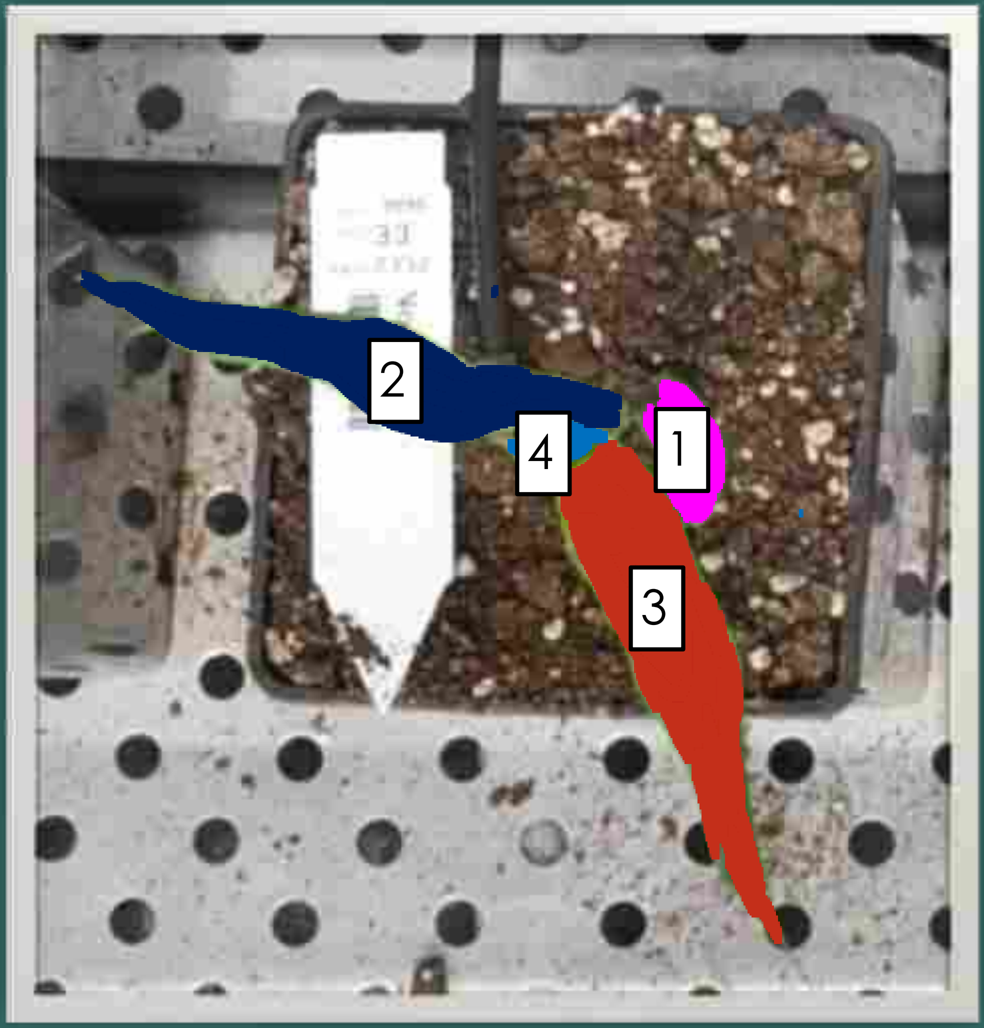
Example only:



Project Goals

- Segmentation of corn leaves
- Separate between different leaves
- Number the leaves from youngest to oldest

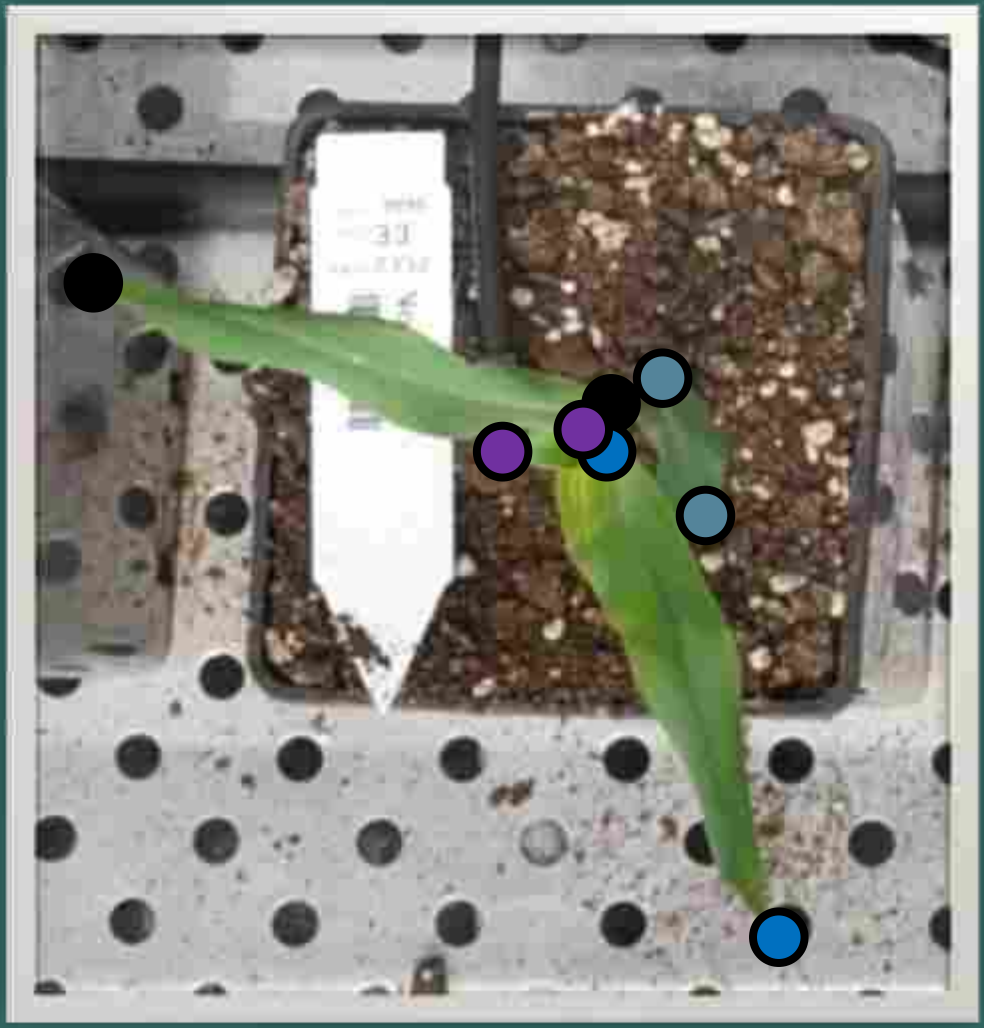
Example only:



Project Goals

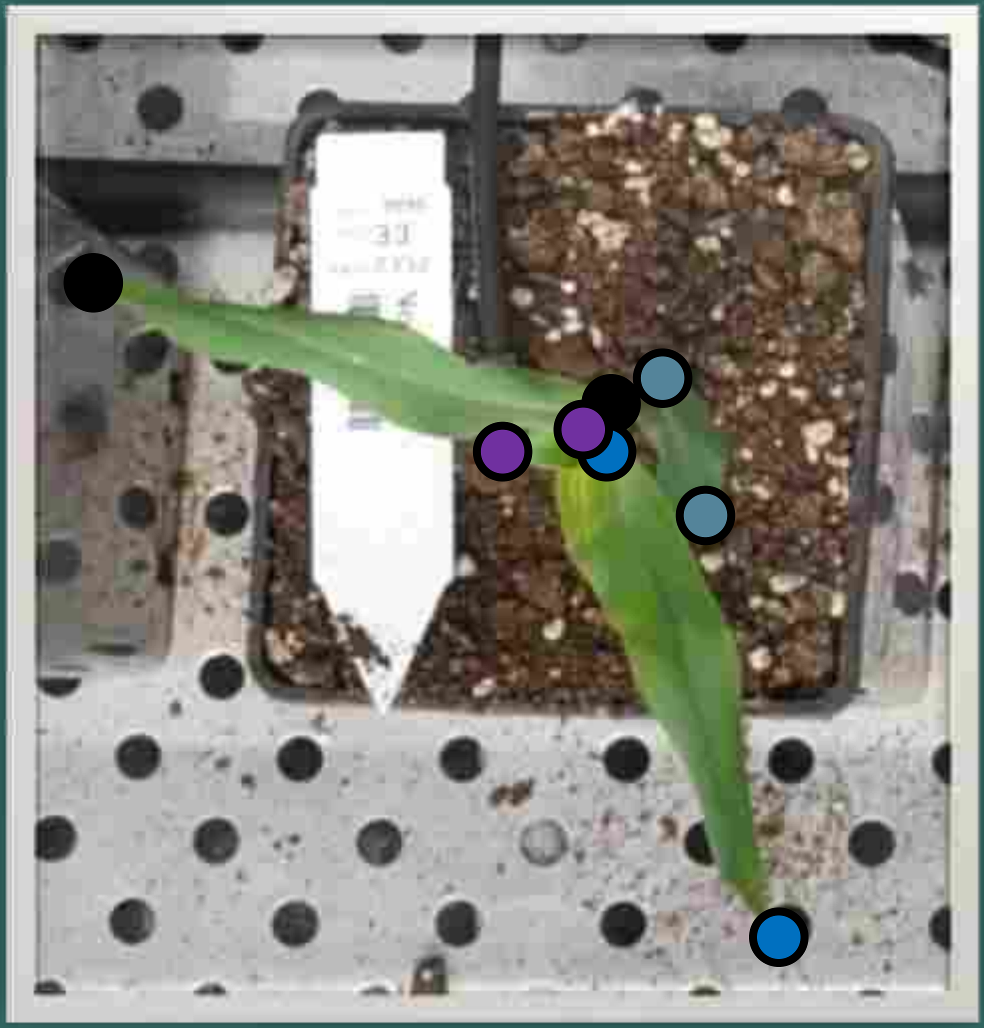
- Segmentation of corn leaves
- Separate between different leaves
- Number the leaves from youngest to oldest
- Find start and end points of each leaf

Example only:



Project Goals

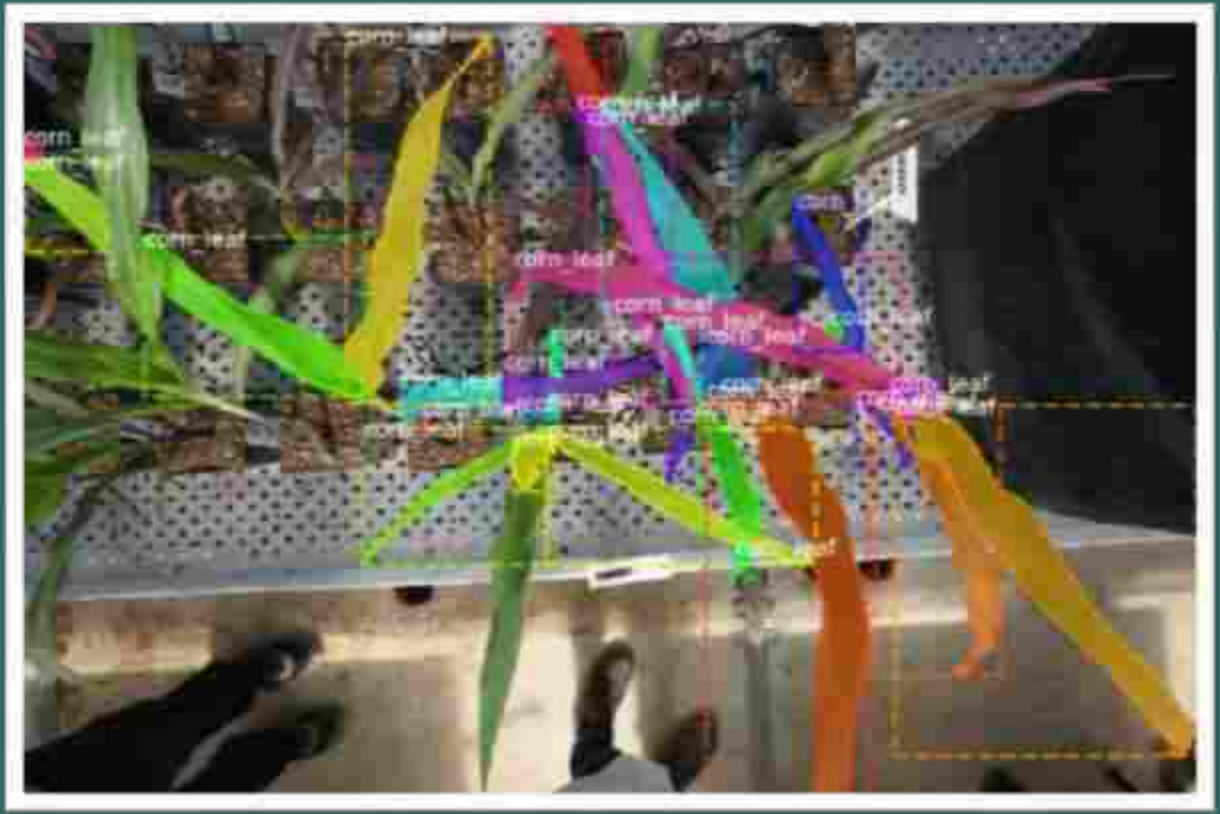
Example only:

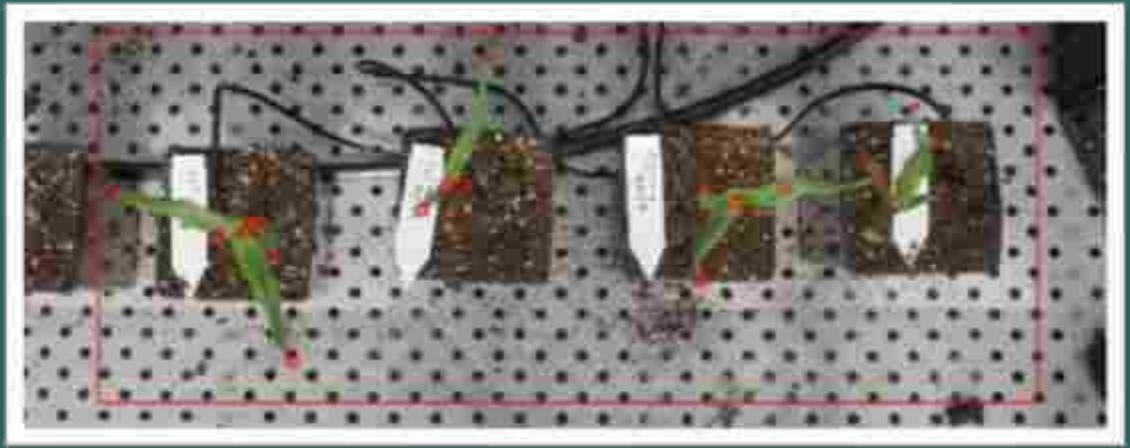


- Segmentation of corn leaves
- Separate between different leaves
- Number the leaves from youngest to oldest
- Find start and end points of each leaf
- NOTE: this was not possible, since the notated data we were given, was wrong in regard the numbering of the leaves

Visualizing the annotations

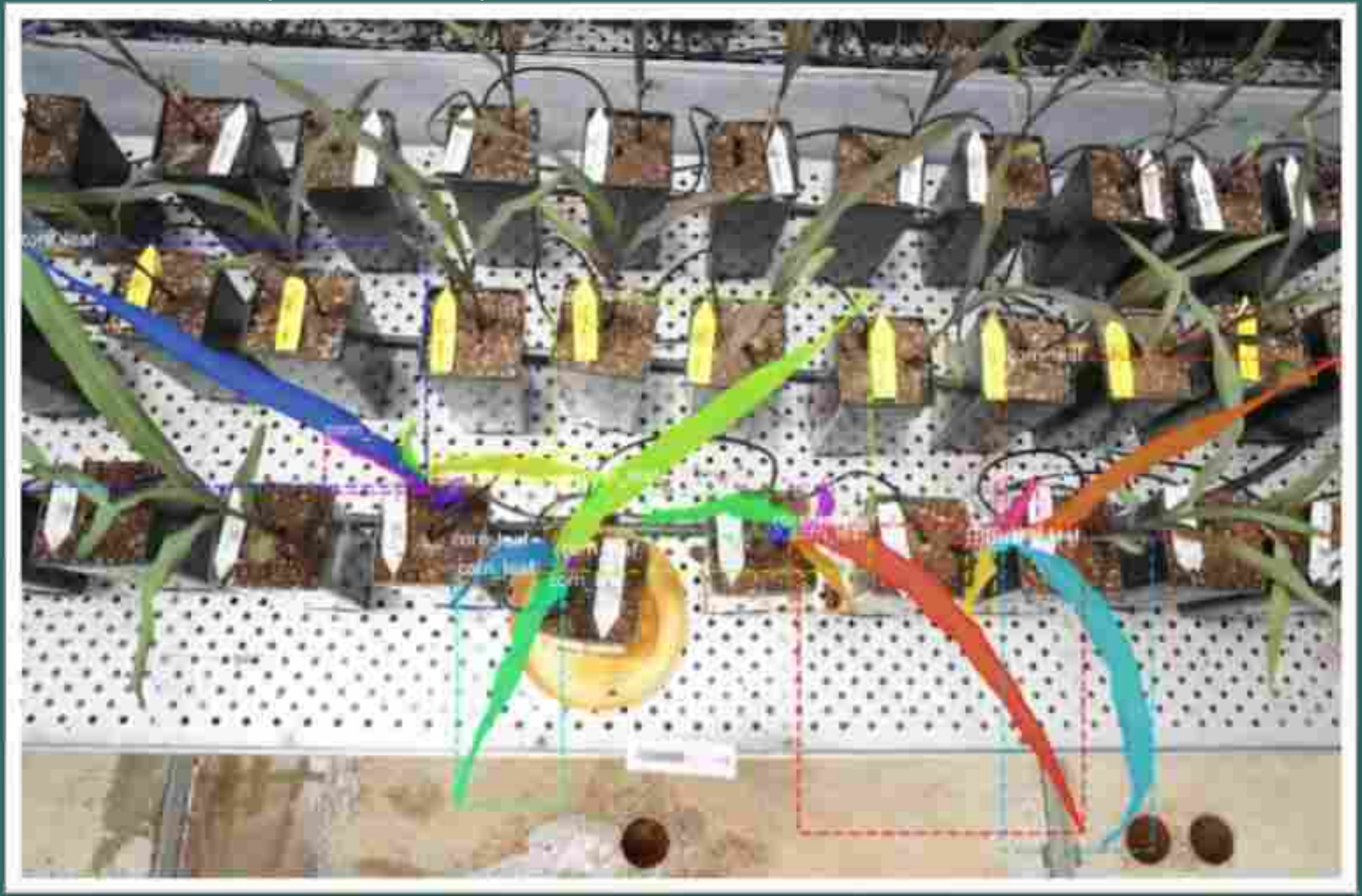
- As you can see, not all plants are annotated.





Another Example

- Again, not all plants are annotated, and cropping out **only** the annotated plants is a problem.



Dataset preparations - problem

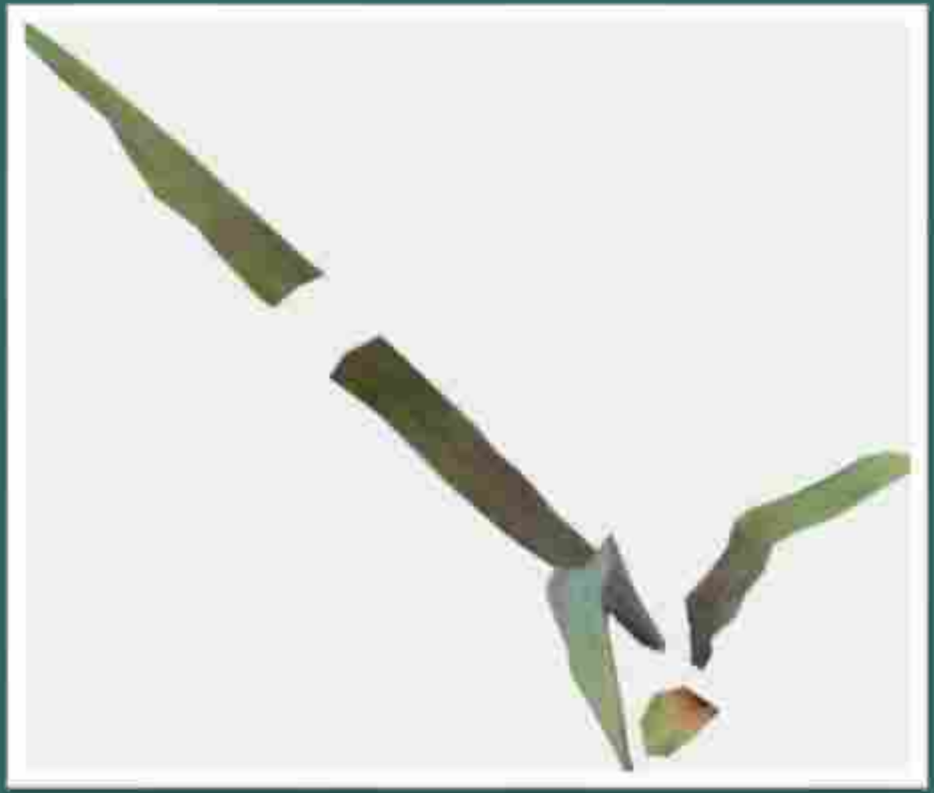
- Before we can start using data augmentation techniques, we realized, we have a problem, not all plants are annotated.

Not all plants in this image are annotated.
If we use images like this, Mask R-CNN will learn to differentiate between plants that were annotated, and plants that weren't.

Dataset preparations - solution

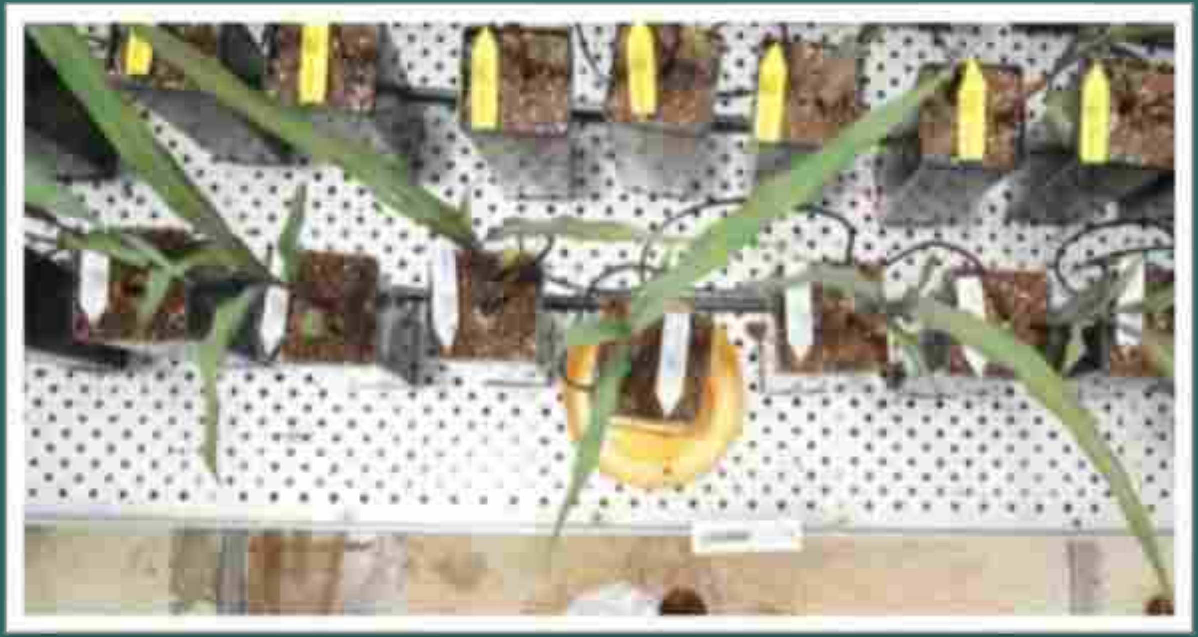
1. Cut the images to individual corn plants.
   Include also images with multiple plants (ROIs).
2. Remove backgrounds. This is done to remove any untagged plant.

- For example:



Dataset preparations -solution

- Another example:



Dataset preparations - Background

- We’ve decided to create the background out of the original images, in hopes that it will help Mask RCNN differentiate between the background and the plants.

Dataset preparations –solution

3. Creating the background

3.1 we take one of the original images.

3.2 we crop out the annotated plants and leave only the ROI.

- Notice there are still leaves after deletion
- The white border represents the ROI

Dataset preparations -solution

- However, since the annotations aren't perfect, we are left with parts of the leaves.

Dataset preparations -solution

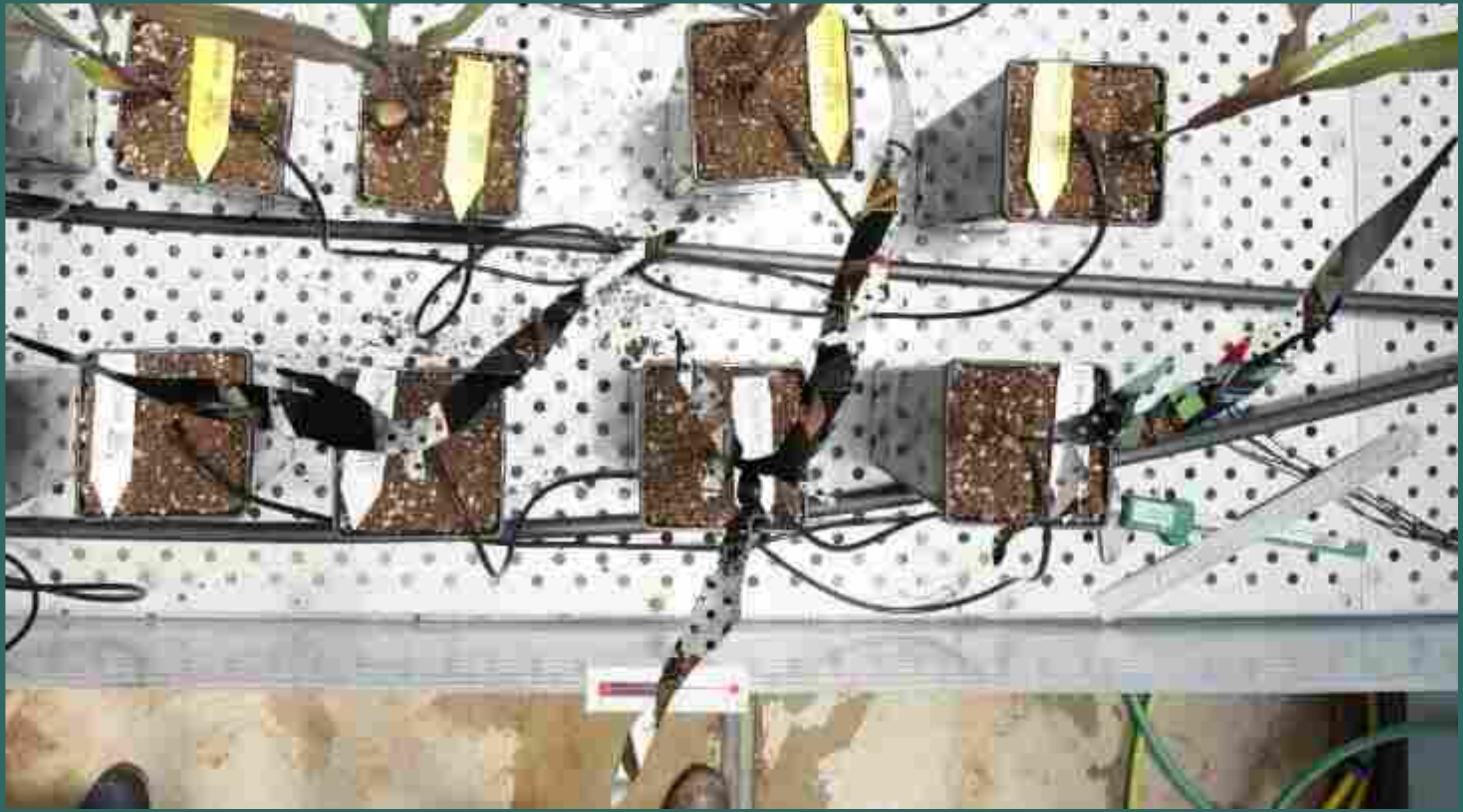
4. So we used: erosion and smoothing.

Dataset preparations -solution: Filling the Holes left by the cropped plants.

5. We stacked the images one on top of the other, until all holes were filled

Dataset preparations -solution: Filling the Holes left by the cropped plants.

- But there were still problems – as you can see, there are still plants in the background



Elaborating on the background creation process

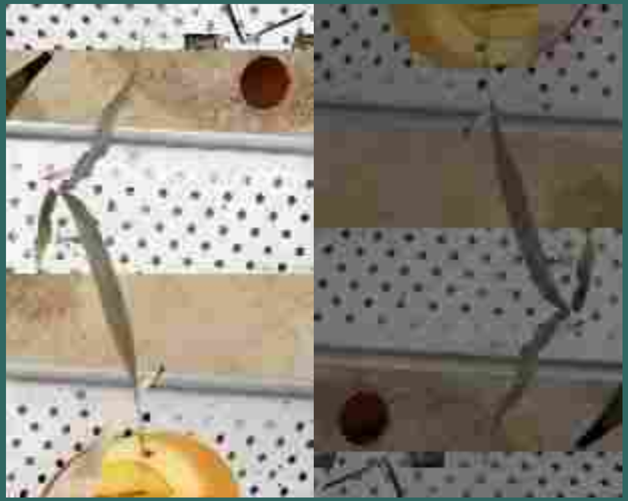
- Create histogram of leaf colors from the cropped images.
  - Only most dominant colors are chosen.
    - Compute a color histogram for each plant
    - Take 20 most common colors from each plant (~3000 colors overall)
    - Dilute the list by only choosing colors which are at least 10 units apart (Cartesian Length)
    - We end up with ~ 150.
- Create tiles of backgrounds without leaves in them (compare all pixels to the leaf colors).
- Trim tiles to standard sizes (e.g. 113X503 ->100X500)
- Construct bigger backgrounds by combining tiles.
- Trim the edges

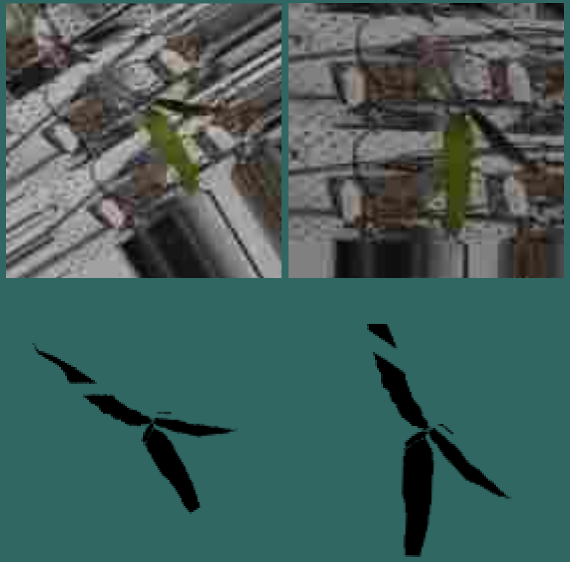
Result of dataset preparations

Result of dataset preparations

Data Augmentation - finally

- First we tried doing it ourselves
  - We've implemented:
    - Horizontal flip
    - Vertical flip
    - Rotate
    - Stretch
    - Brightness adjustment
  - This required a lot of effort, since editing images takes a lot of time, we had to use multiprocessing to cut down times.





Training time

- As we implemented more data augmentation techniques, the datasets grew, and the training time grew as well. It took us up to 30 hours to train a network.
- So at first, we tried to switch to data augmentation while running the trainings, since we thought maybe the reading time from the disk took too long.
  - However, the results were even slower.
  - On a positive note, implementing the data augmentation this way made it easier to find good data augmentations, since there is no need to create a new dataset each time.
- Next, we tried rescaling the image sizes to 256x256 - and it worked, we cut down training time by 800%.

Experiment #1 – affine data augmentation

The affine augmentation is composed of:

1. Resizing the image
2. Translate – moving the image
3. Rotate
4. Shear – stretch the image

Dataset size: ~3200 images

Experiment #2 and #3 – data augmentation

The augmentation is composed of:

1. Affine
2. Invert – inverts the pixel value per channel
3. Hue and saturation
4. Contrast normalization
5. Flip – both vertical and horizontal

Dataset size #2: ~3200 images
Dataset size #3: ~6400 images

Experiment #4 – affine and emboss

The augmentation is composed of:

1. Affine
2. Emboss

Dataset size: ~3200 images

Experiment #5 – affine and flip

The augmentation is composed of:

1. affine
2. flip

Dataset size: ~3200 images

Experiment #6 – data augmentation – affine and gauss

The affine augmentation is composed of :

1. affine
2. gauss

Dataset size: ~3200 images

Experiment #7 – data augmentation – affine and hue

The affine augmentation is composed out of affine and :
1. hue

Dataset size: ~3200 images

Affine – Flip (best result)

Mask Results

We couldn't see this, until we colored the masks.

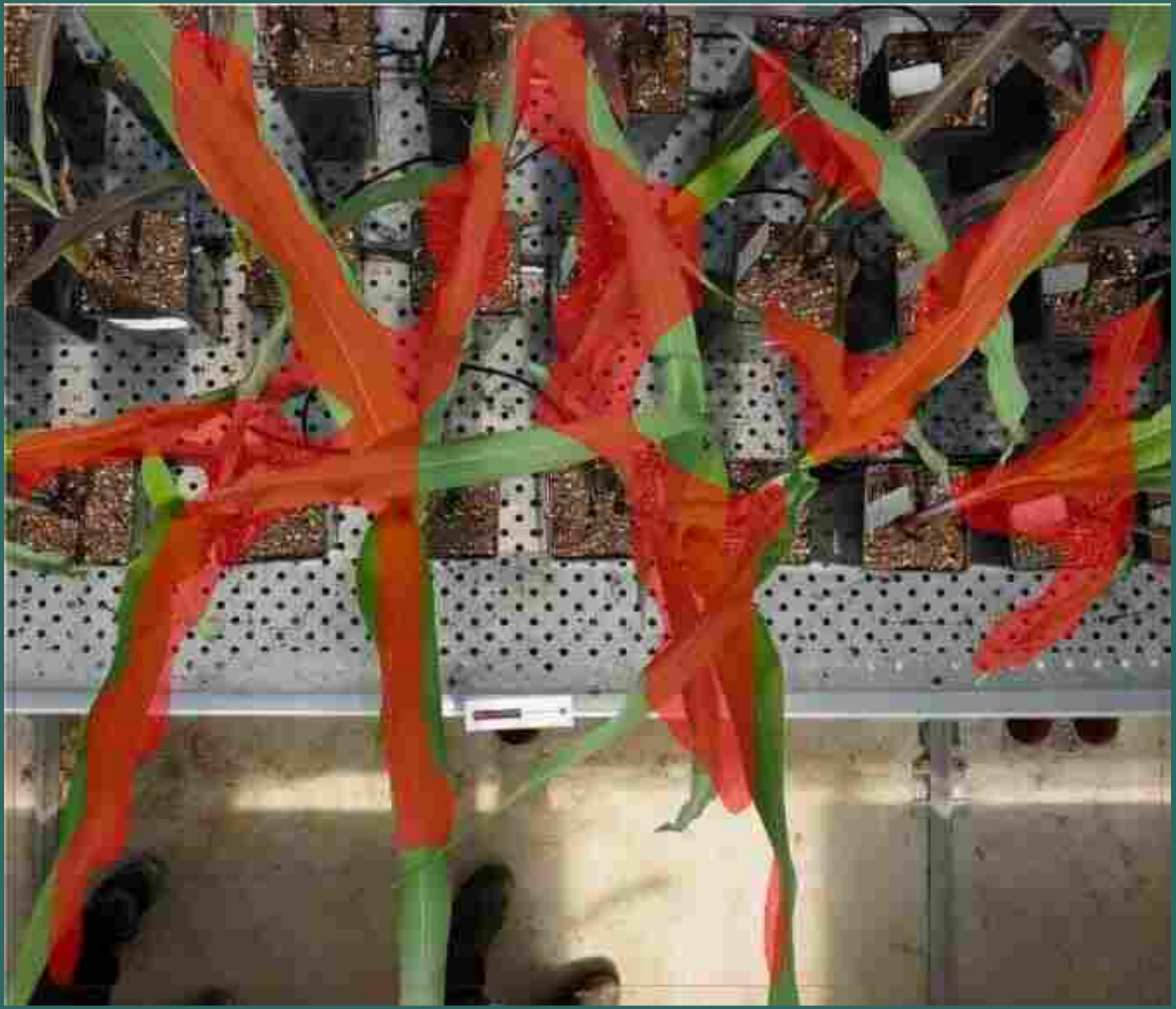
Problem, since we gave Mask RCNN only images with one plant in them, the resulted NN couldn't handle multiple plants in one image.

Affine – Flip (best result)

Ground Truth

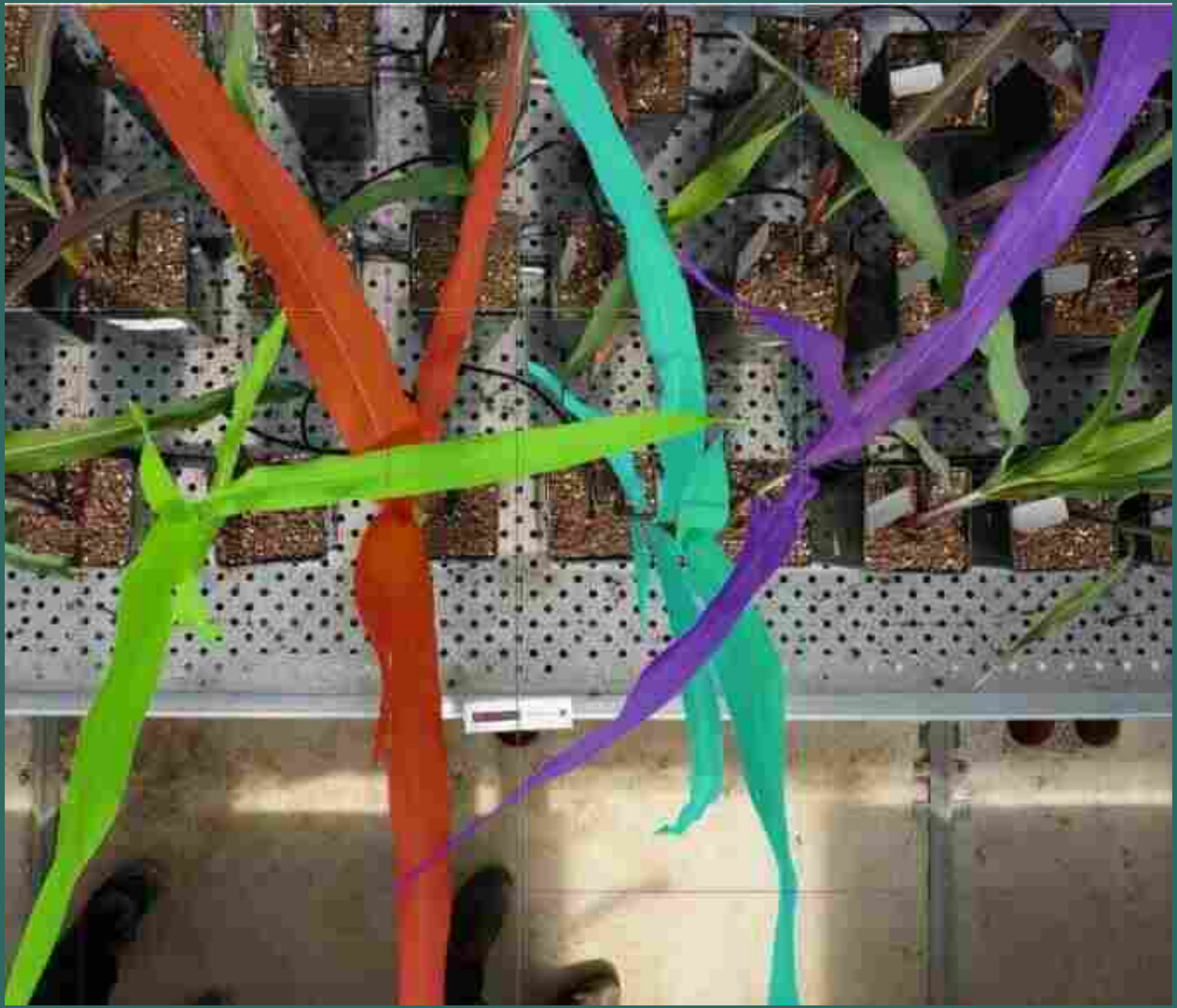
Affine – Flip (best result)

Mask Results



Affine – Flip (best result)

Ground Truth



All Results

Type of images		affine	affine						All
			hue, invert, flip, contrast		emboss	flip	gauss	hue	
			3200 steps	6400 steps					
IOU on original images from test set (ROIs)	Original size~3000x4000	9%	9%	11%	10%	13%	10%	8%	9%

As you can see, the results weren't good as we expected, so we thought that maybe if we change the image size of the original test image, we would get better results.

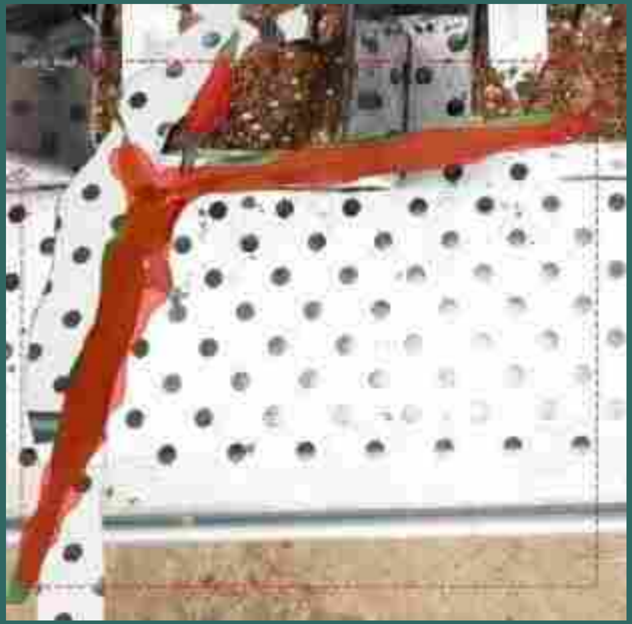
All Results

Type of images		affine	affine						All
			hue, invert, flip, contrast		emboss	flip	gauss	hue	
			3200 steps	6400 steps					
IOU on original images from test set (ROIs)	Original size~3000x4000	9%	9%	11%	10%	13%	10%	8%	9%
	Min(width,height)= 1024	11%	11%	8%	8%	12%	10%	9%	9%
	Max(width,height)= 1024	12%	11%	8%	11%	14%	10%	9%	7%
	50% of the original size	9%	11%	9%	10%	14%	10%	9%	8%

We were wrong, changing the original image size didn't help.
So we ran the NN on the augmented images from the test set.

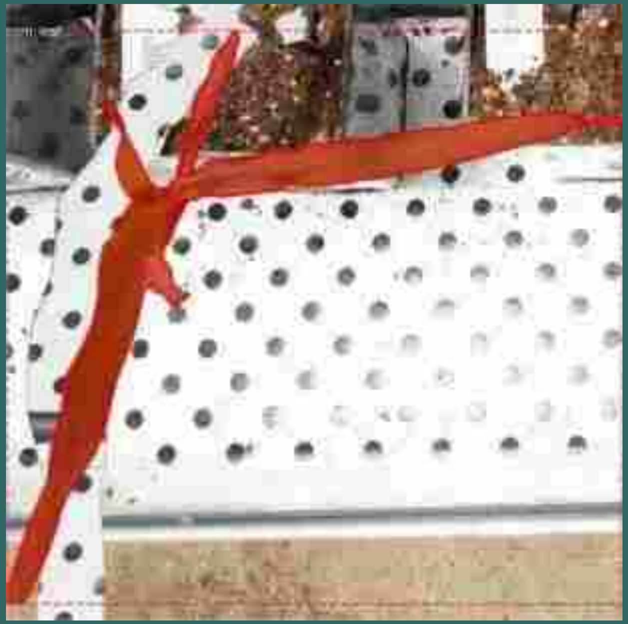
Affine – Flip
(best result on augmented plants)

Mask Results



Affine – Flip
(best result on augmented plants)

Ground Truth



All Results

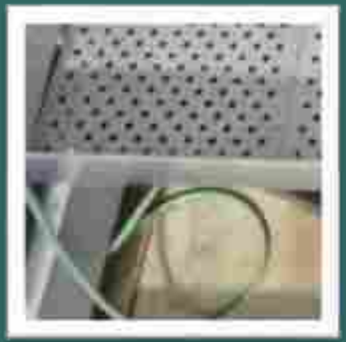
Type of images		affine	affine						All
			hue, invert, flip, contrast		emboss	flip	gauss	hue	
			3200 steps	6400 steps					
IOU on original images from test set (ROIs)	Original size~3000x4000	9%	9%	11%	10%	13%	10%	8%	9%
	Min(width,height)= 1024	11%	11%	8%	8%	12%	10%	9%	9%
	Max(width,height)= 1024	12%	11%	8%	11%	14%	10%	9%	7%
	50% of the original size	9%	11%	9%	10%	14%	10%	9%	8%
IOU on cropped out single plants from test set		49%	57%	49%	57%	54%	44%	53%	54%

Conclusions

- Possible reasons for the low IOU results:
  - The custom backgrounds:
    - still have cut-outs in the shape of leaves that may confuse the network.
    - might have small leaves that we and the script missed.
  - One plant per image.

Improvement Ideas

- Use random backgrounds from the internet.
  - Add hand picked backgrounds cropped out of the original images, to eliminate detection of objects in the background as leaves.
- Place a few plants in each image
  - Random location and rotation






2-5 plants, mask per plant

Mask Results

2-5 plants, mask per plant

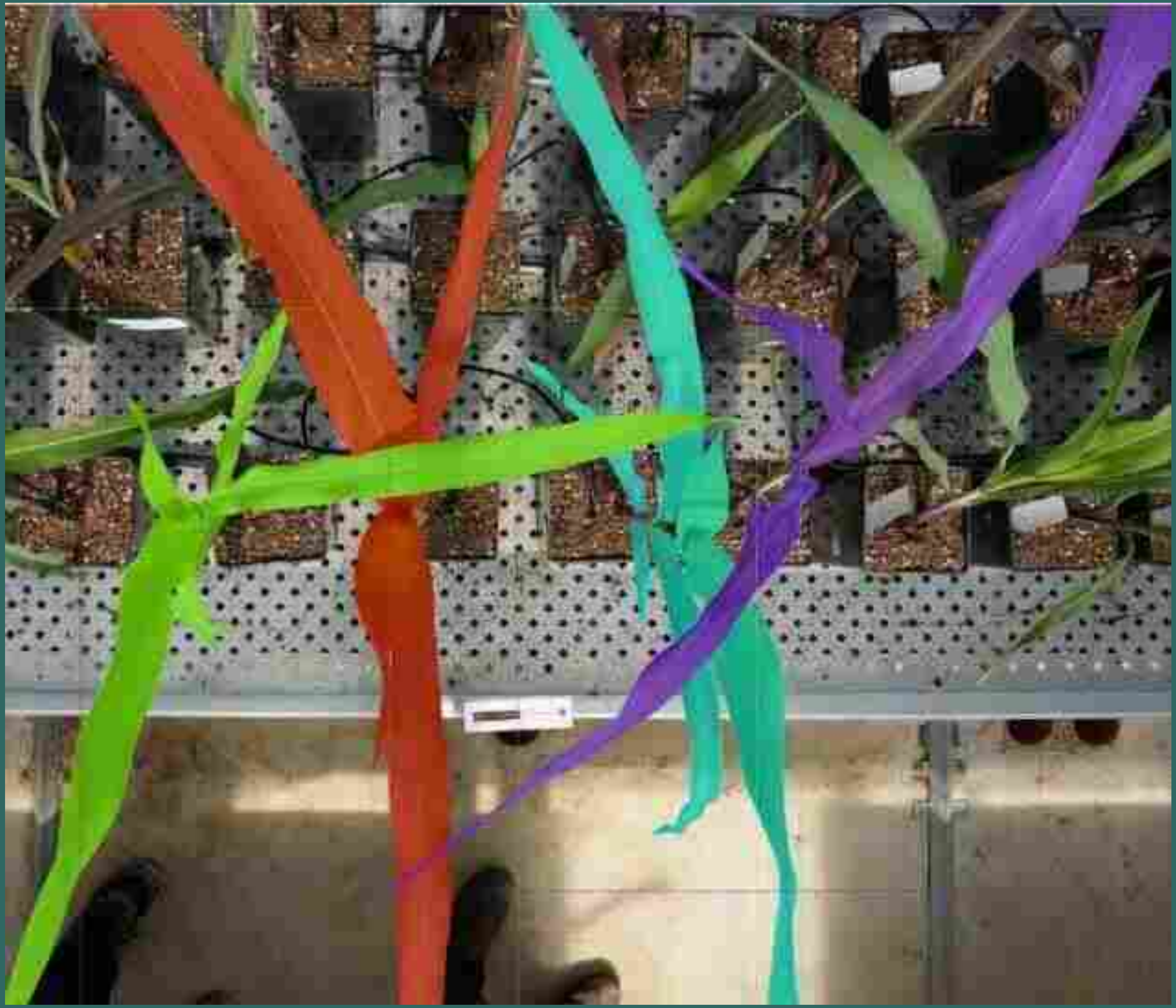
Ground Truth

2-5 plants, mask per plant

Mask Results

2-5 plants, mask per plant

Ground Truth



Rethinking Our Strategy

- Using one mask for each plant we didn't get good enough results, so we tried training using one mask for each leaf.

2-5 plants, mask per leaf

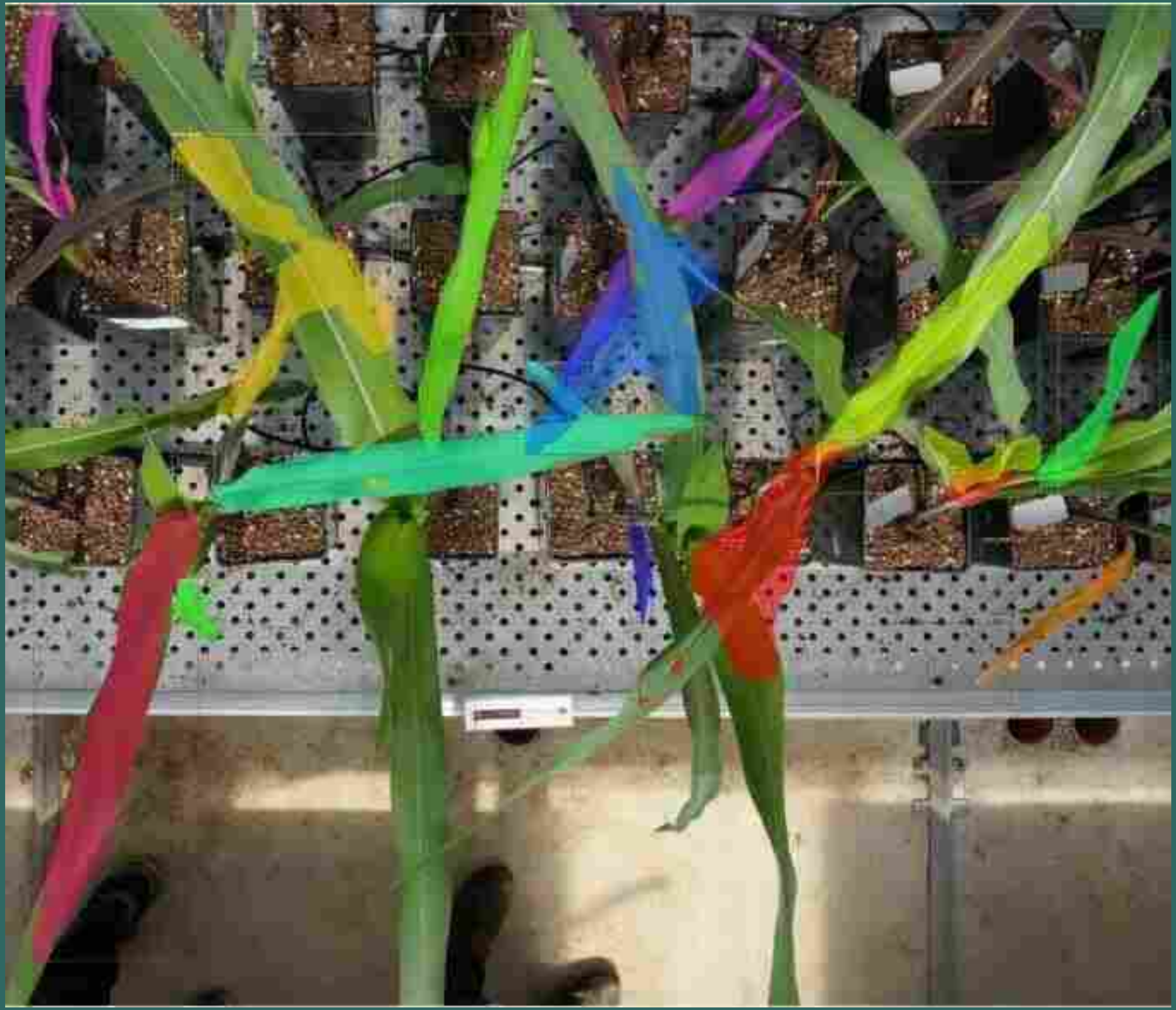
Mask Results

2-5 plants, mask per leaf

Ground Truth

2-5 plants, mask per leaf

Mask Results



2-5 plants, mask per leaf

Ground Truth

4-10 plants, mask per leaf

Mask Results

4-10 plants, mask per leaf

Ground Truth

4-10 plants, mask per leaf

Mask Results

4-10 plants, mask per leaf

Ground Truth

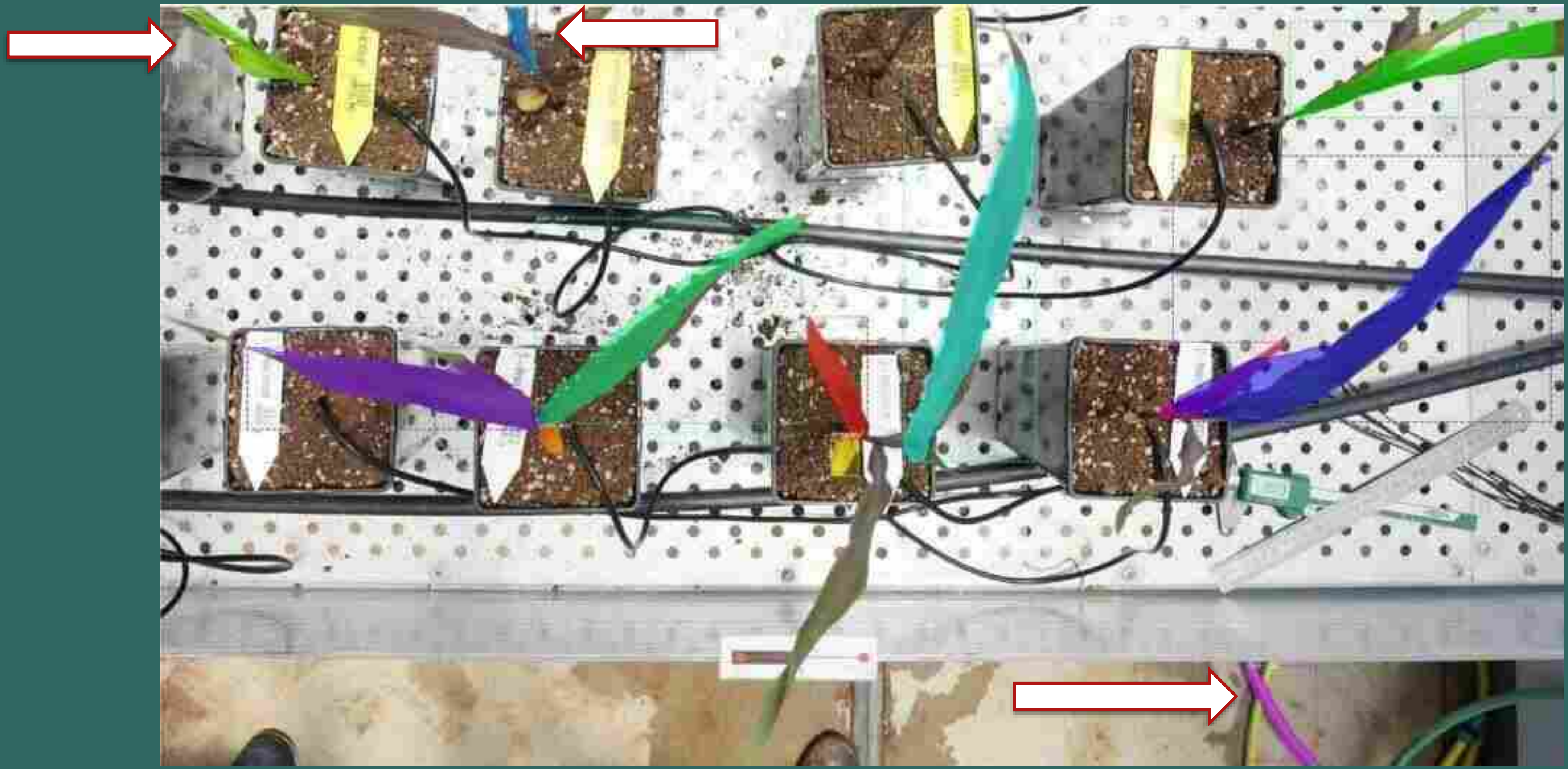
Results

Type of images	2-5 plants per image		4-10 plants per image	
	mask per plant	mask per leaf	mask per plant	mask per leaf
IOU on original images from the test set (ROIs)	32%	55%	-	61%
IOU on cropped out single plants from test set	53%	59%	-	59%

New IOU	IOU on original images from test set (ROIs)	IOU average	32%	60%	-	64%
		clashes	18%	2%	-	3%
		Incorrect	0%	23%	-	20%
		Not found	43%	40%	-	39%

New IOU - match result segments to ground truth segments
Clash – two or more result segments match the same ground truth segment (percent of all result segments)
Incorrect – a result segment which does not match any ground truth segment (percent of all result segments)
Not Found – a ground truth segment which was not matched to any result segment (percent of all ground truth segments)

Incorrect Mask Example



Conclusions

- More plants per image perform better, since we then get more overlapping between plants, as we have in the original images.
- We still don't get good enough results.
  - Probably because there is not enough variation – we use the same 160 plants and augment them.

Solution

- Create new plants of the individual leaves to increase variation.

Creating New Plants

- Problems
  - How to Align the leaves so that the bottom of the leaf is at the bottom of the image? (the provided annotations were inconsistent)
  - How to assemble a plant that looks real?
  - We’ve already reached the maximum RAM (90GB) on the server. How do we lower the size of the masks?

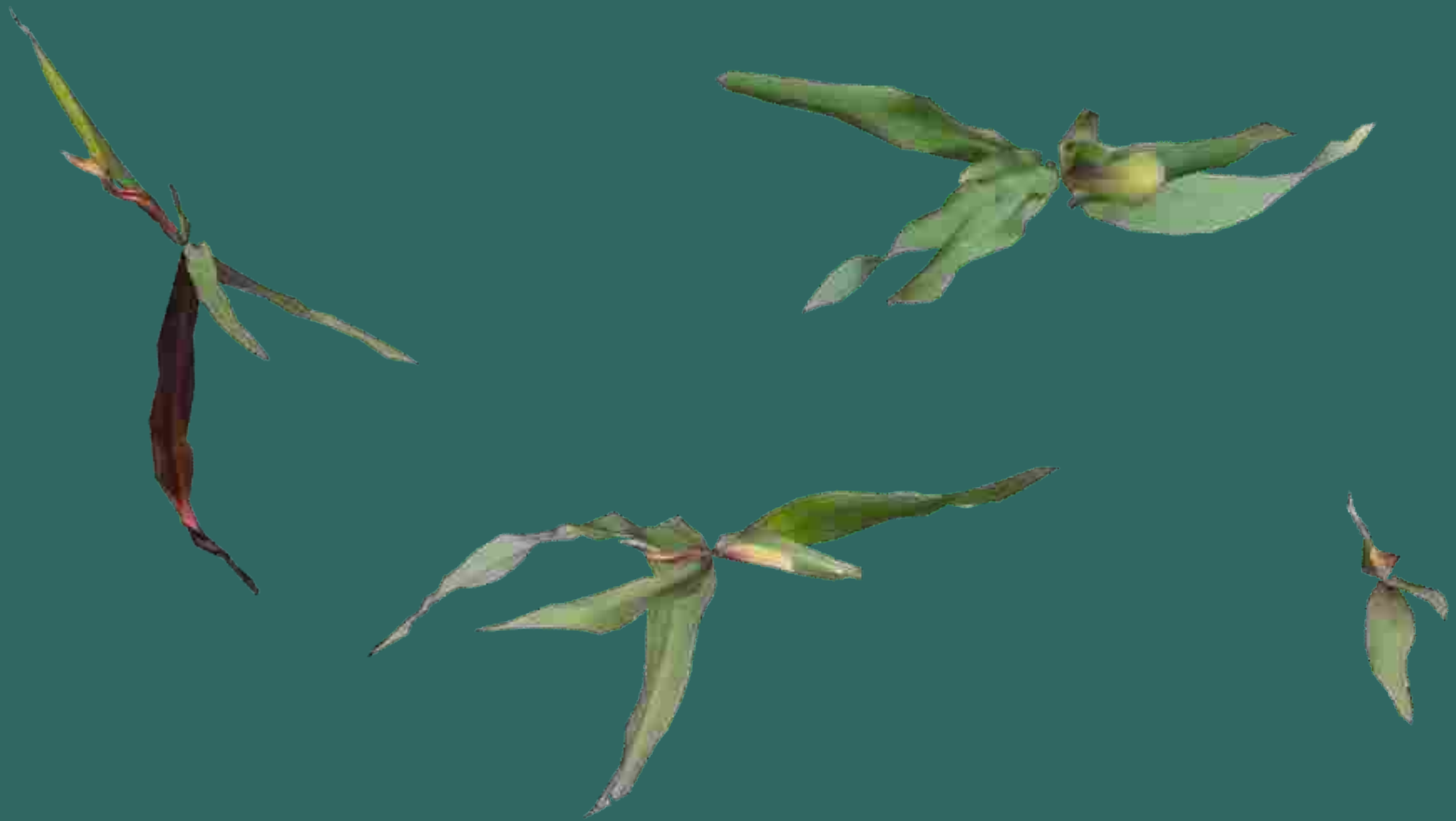
Heuristic for Aligning Leaves

- Per plant, we created a polygon for each leaf.
  - For each leaf we got the two most distant points in the polygon
- For all the points we gathered(2n points, since we have n leaves), we searched for the closest n points
- We took the n points and averaged them, the result is the center point of the plant
- Now that we know the center of each plant, we can infer the bottom (and top) of each leaf.
- Results:

Assembling a plant

- We choose a random set of 3 to 7 leaves
- we sort the leaves by size from small to big
- for each leaf in the random set
  - Each odd leaf we place at 0 degrees + a random degree between -30 and 30
  - Each even leaf we place at 180 degrees + a random degree between -30 and 30

Samples of Assembled plants



4-10 assembled plants, mask per leaf

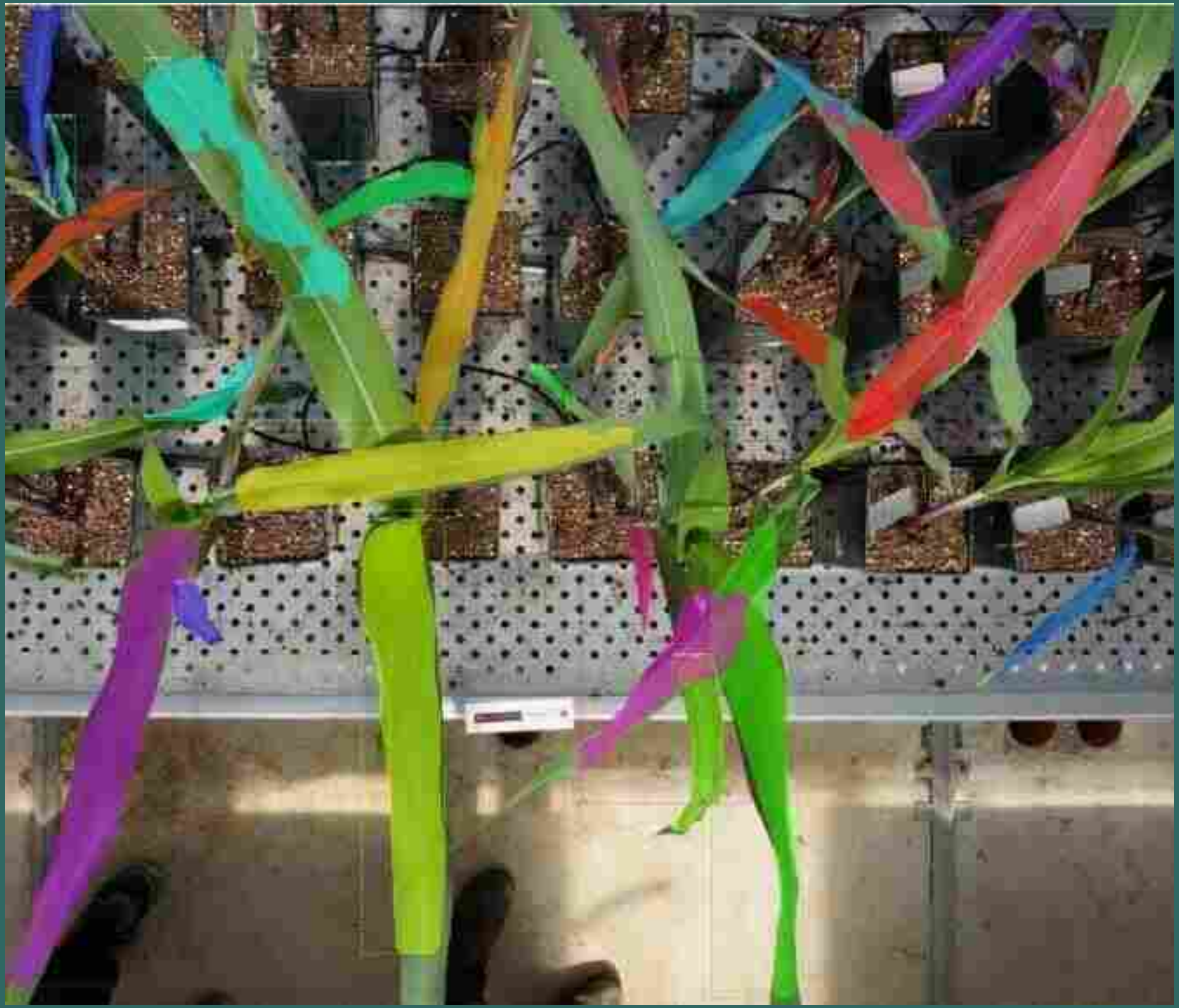
Mask Results

4-10 assembled plants, mask per leaf

Ground Truth

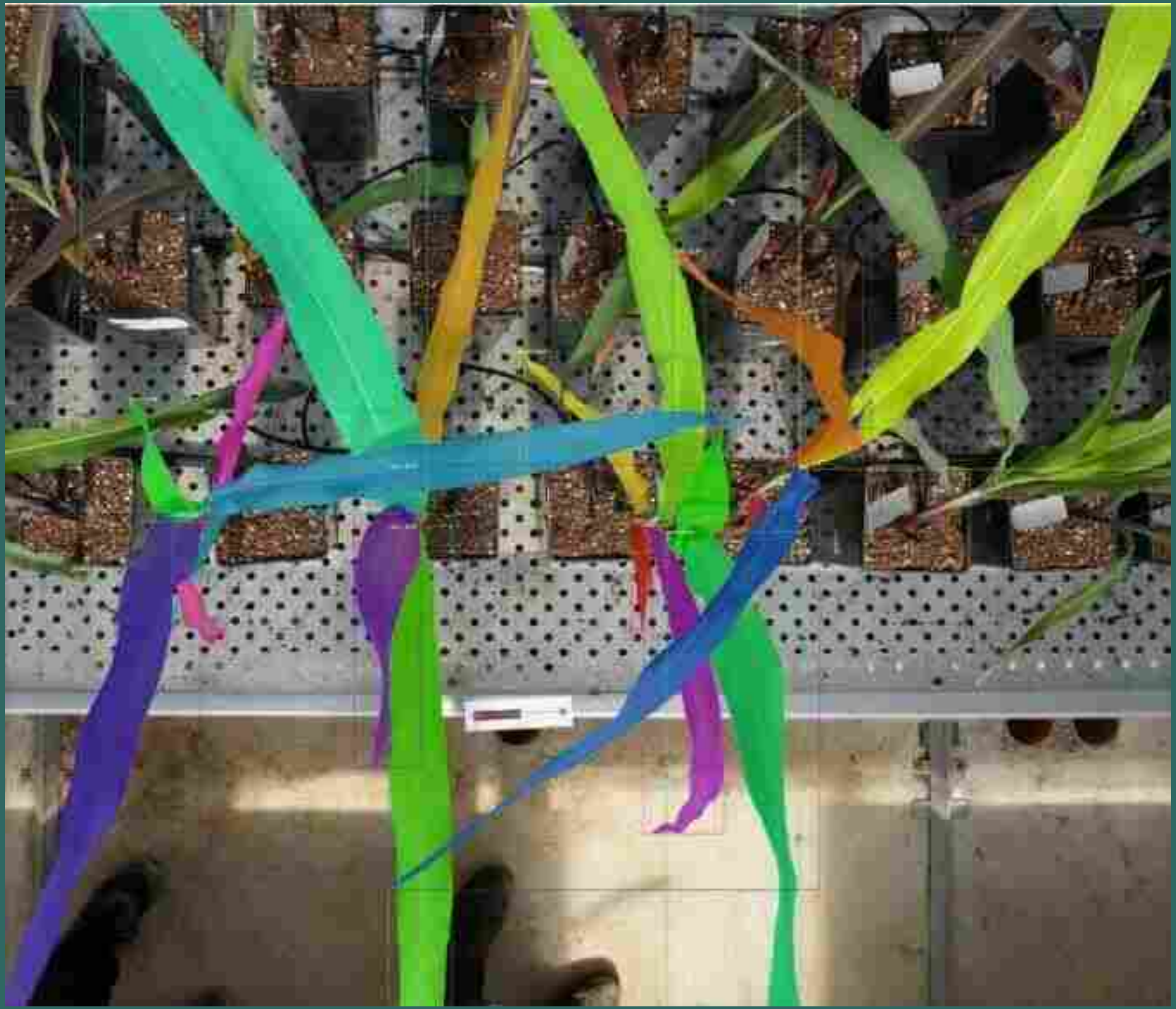
4-10 assembled plants, mask per leaf

Mask Results

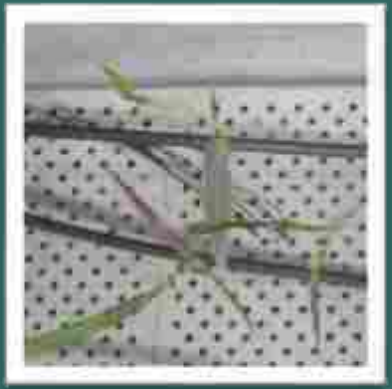


4-10 assembled plants, mask per leaf

Ground Truth



Using The Assembled Plants

<table>
<tr><th colspan="2"></th><th>4-10 plants
mask per leaf</th></tr>
<tr><td colspan="2">IOU on images from test
(original ROIs)</td><td>56%</td></tr>
<tr><td colspan="2">IOU on cropped out single plants from test:</td><td>60%</td></tr>
<tr><td rowspan="4">New IOU
on images from test
(original ROIs)</td><td>IOU average</td><td>60%</td></tr>
<tr><td>clashes</td><td>5%</td></tr>
<tr><td>Incorrect</td><td>16%</td></tr>
<tr><td>Not found</td><td>39%</td></tr>
</table>

No significant improvement

More Improvements

We thought that maybe the plants should be better separated to look more like the original images, and to limit them from overlapping too much.

We created new images, with multiple plants in each image.

1. For each image, we randomly selected if it was going to have 1, 2, or 3 rows and 1, 2, or 3 columns. So each image had 1 to 9 plants in it.
2. Each plant was placed randomly in one of the cells in order to mimic the original images (e.g. you can't have the base of two plants in the same coordinate).

Unfortunately, this did not improve the results.

Training images - better separated plants

Conclusions

1. Using affine augmentation and color augmentation didn't really help, the only thing that did was flipping horizontally/vertically.
2. Categorizing all plants as one class when they look so different from one another was a mistake. A better approach would have been to create 2 classes - one for old plants, and a second one for young plants.
3. Creating backgrounds out of the given dataset was a waste of time, we saw no difference when using generic backgrounds.
4. Working with images above 1024x1024 was a waste of time for epochs larger than 1800 images and training session with 30 epochs. Each training session took a minimum of 20 hours.
5. Training MASK R-CNN on images with single plants resulted in bad IOU since all the grown plants intersect consistently in images from the original dataset.

In conclusion, if we had split the dataset to a dataset of young plants and grown plants, we would have gotten much better results. In addition, images with all the plants annotated in them would have helped too, since then we could have used data augmentation on them directly.

Further Work and Suggestions

1. Build a better data augmentation - synthesize a plant that resembles a grown plant from the ground truth. Perhaps only leaves from grown plants should be used for that purpose.
2. Create an image with multiple grown plants that overlap and then minimize it to 1024x1024.