Counting Plants With Deep Learning

Javier Ribera[1], Yuhao Chen[1],
Christopher Boomsma[2], Edward J. Delp[1]

[1]Video and Image Processing Laboratory (*VIPER*)
School of Electrical and Computer Engineering
Purdue University , West Lafayette, Indiana USA

[2]Department of Agronomy
Purdue University , West Lafayette, Indiana USA

Introduction

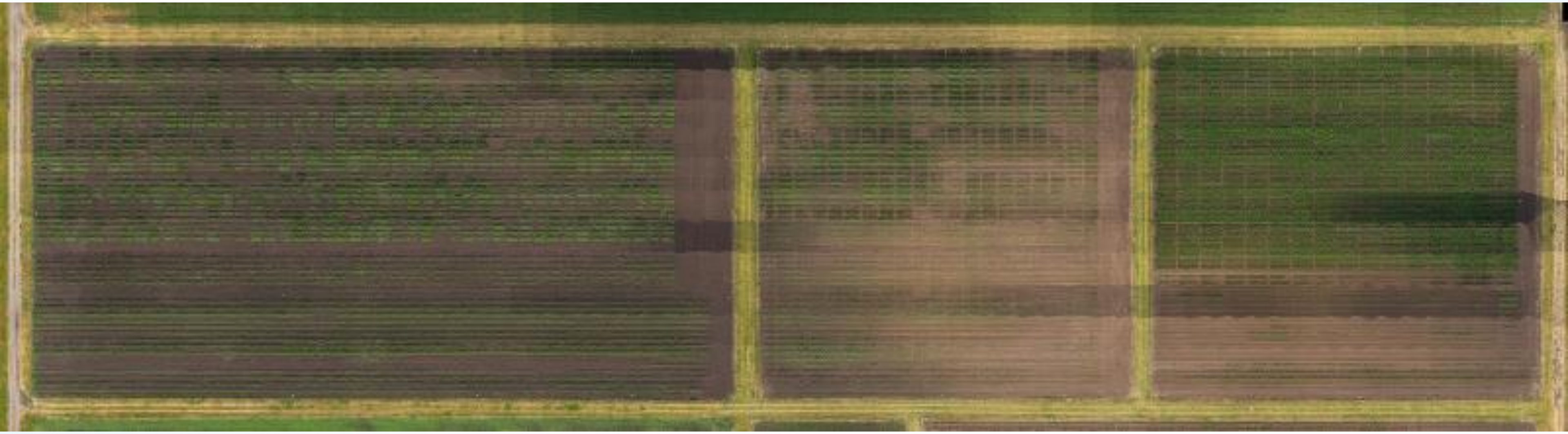
- Agronomists and farmers need to know the number of plants in their crops to predict future yield

- Can we count without locating?
- We count plants in a crop field without knowing where they are

- We build our plant dataset from a single image of the entire crop field
- We describe a method to extract images sections or "plots" from an orthorectified image

1. Plot Extraction

Dataset



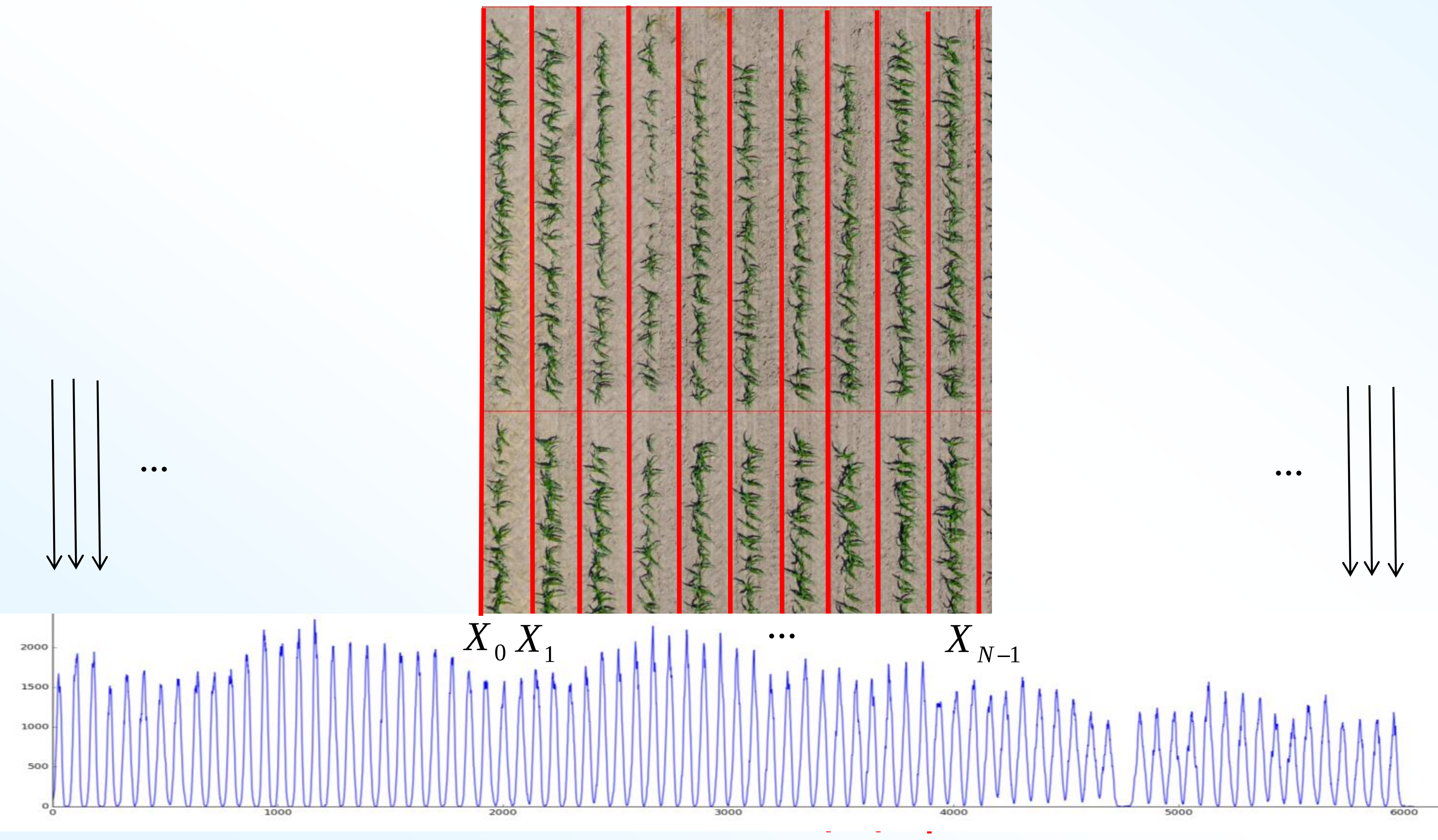
420 meters ~ 4 football fields

Dataset

1,240 images were extracted

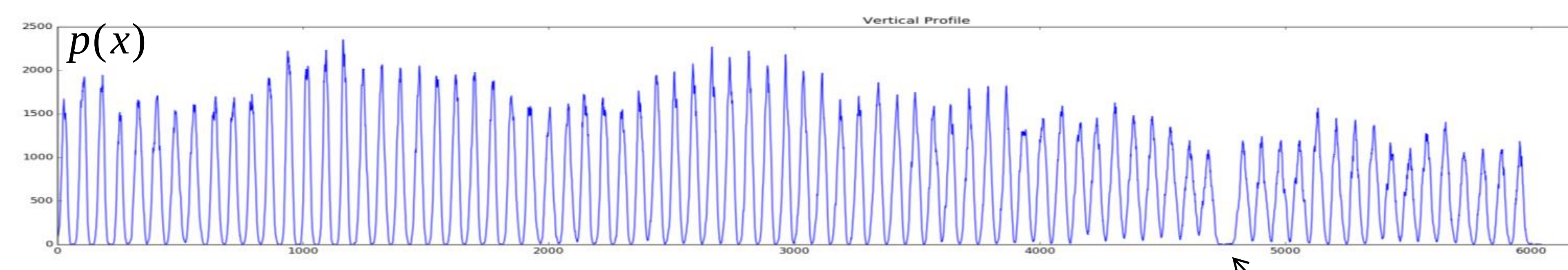
420 meters ~ 4 football fields

Vertical Projection



Profile Function

- **Vertical profile over the entire crop field:**



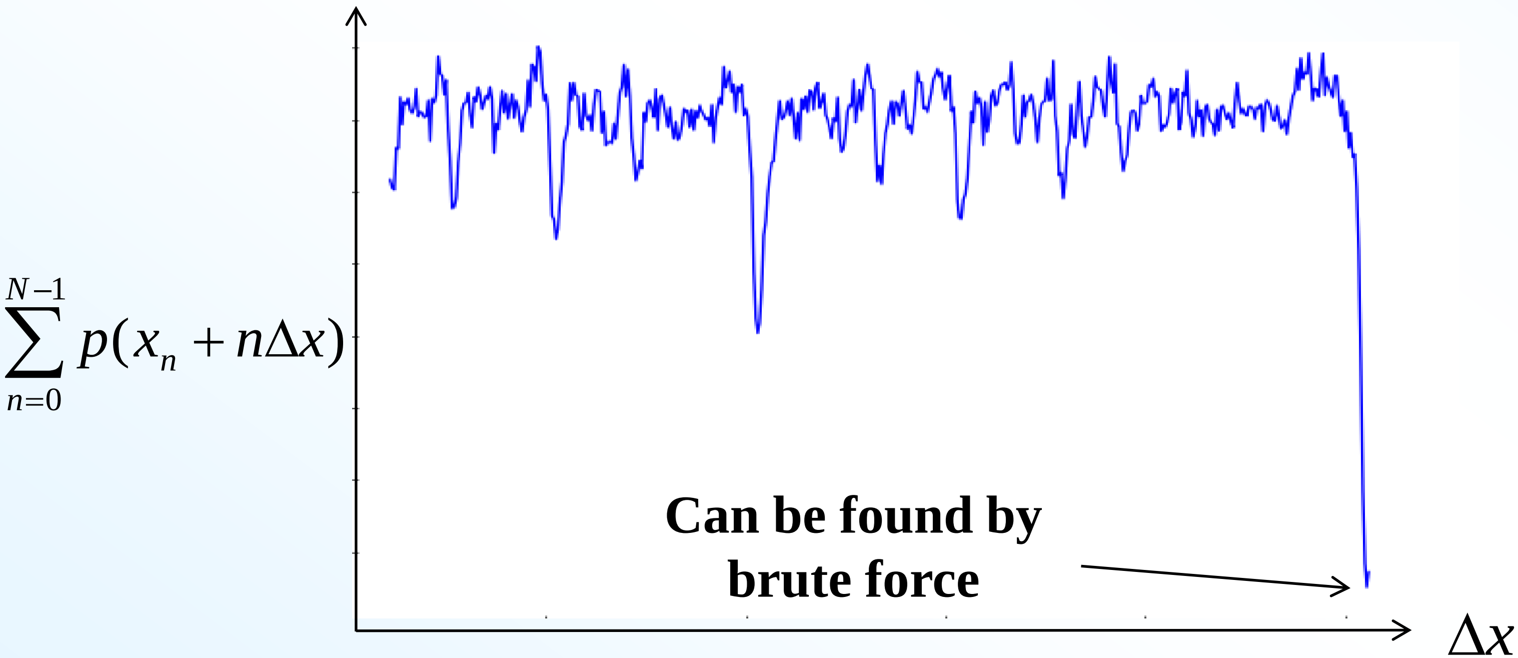
$$\widehat{X}_0, \Delta\widehat{X} = \underset{x_0,\, \Delta x}{\arg\min} \sum_{n=0}^{N-1} p(x_0 + n\Delta x)$$

Row harvested?

- N **: # lines**
- $X_0, \ldots, X_{N-1}$: **Coordinates of each plot-separating line**
- $X_n = X_0 + n\Delta X \qquad n = 0,1,\ldots, N-1$

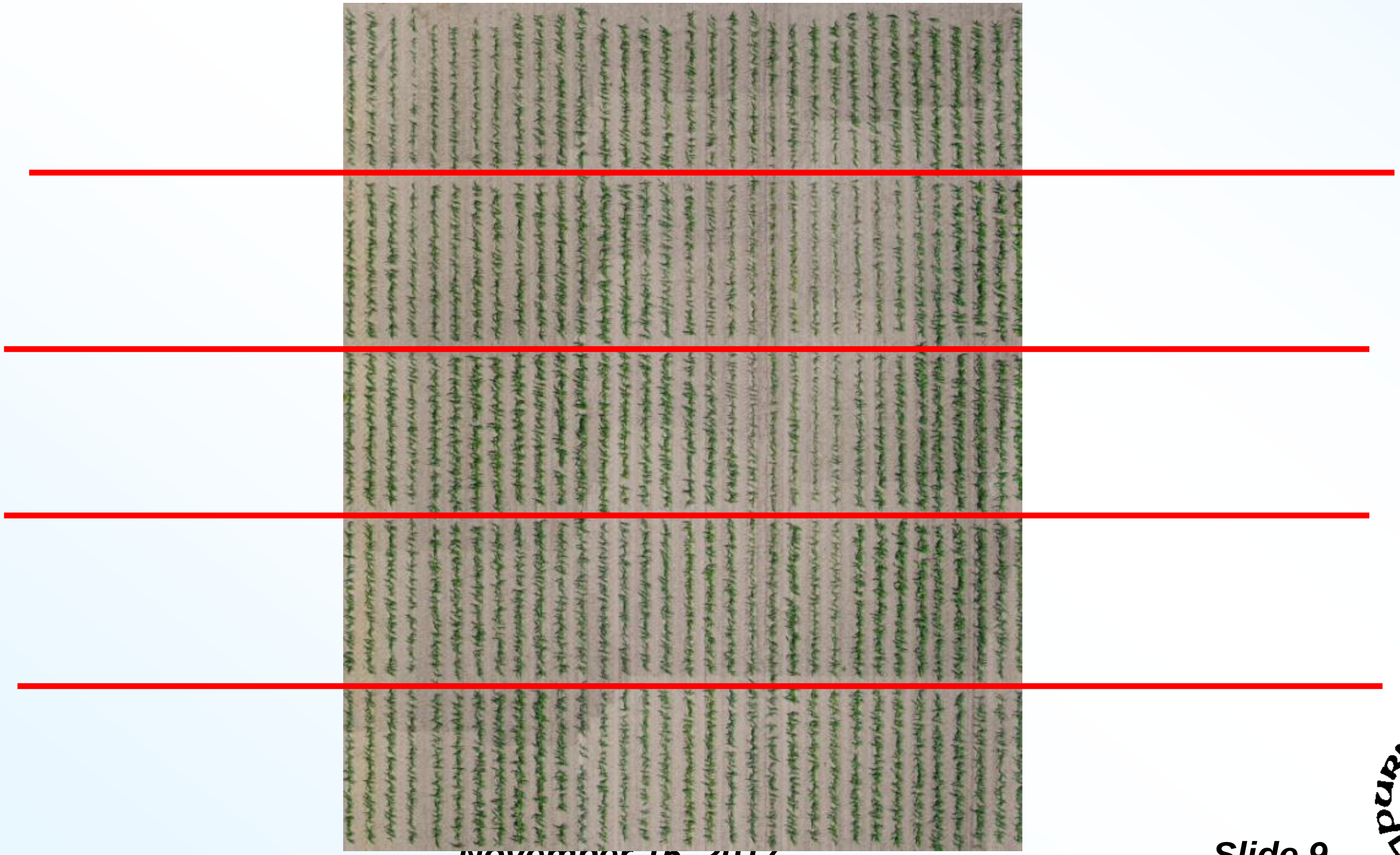
The Cost Function

- **It does not seem appropriate for gradient descent:**



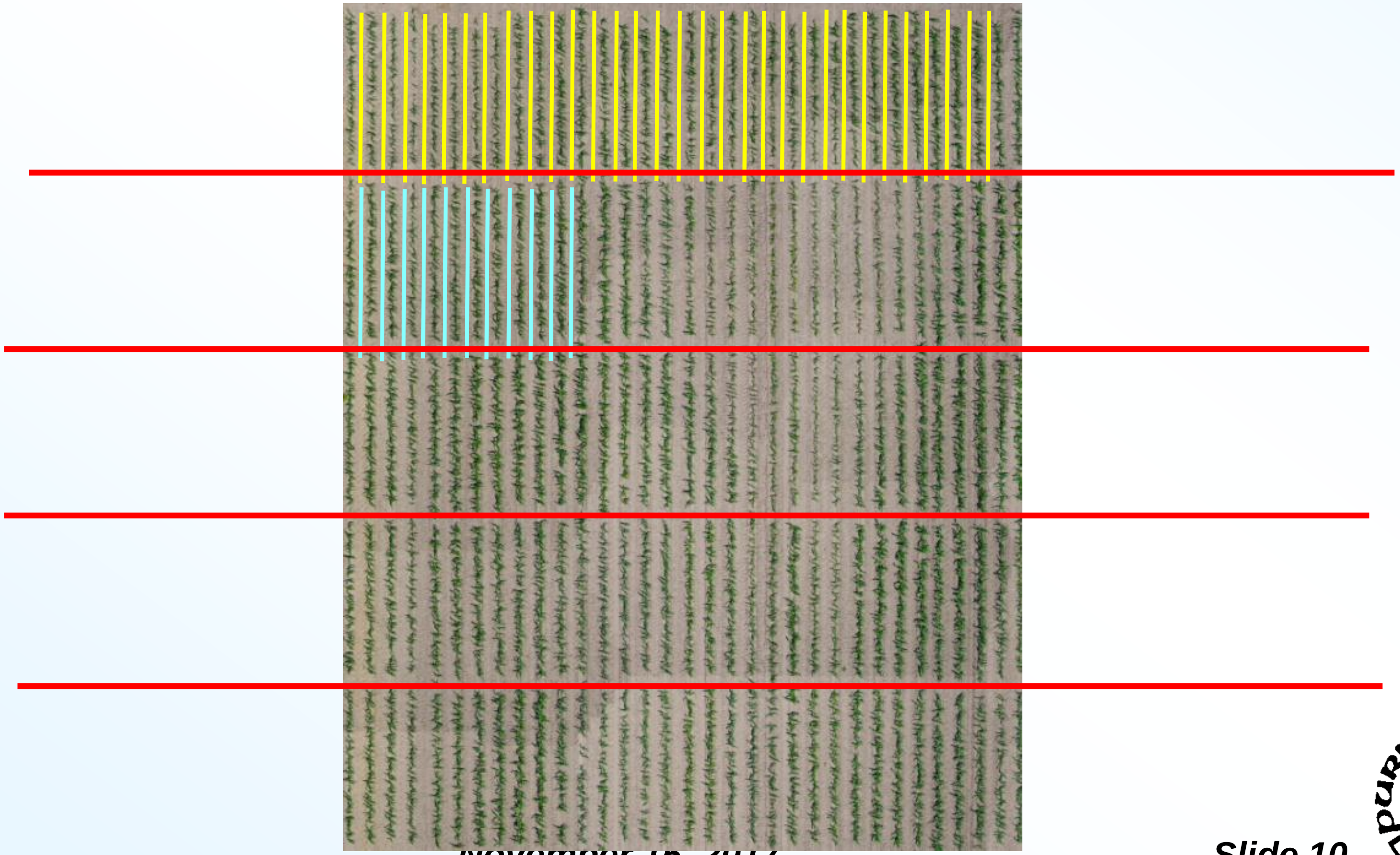
Method

1. User provides: (a) number of rows and (b) number of ranges
2. Find range-separating lines:



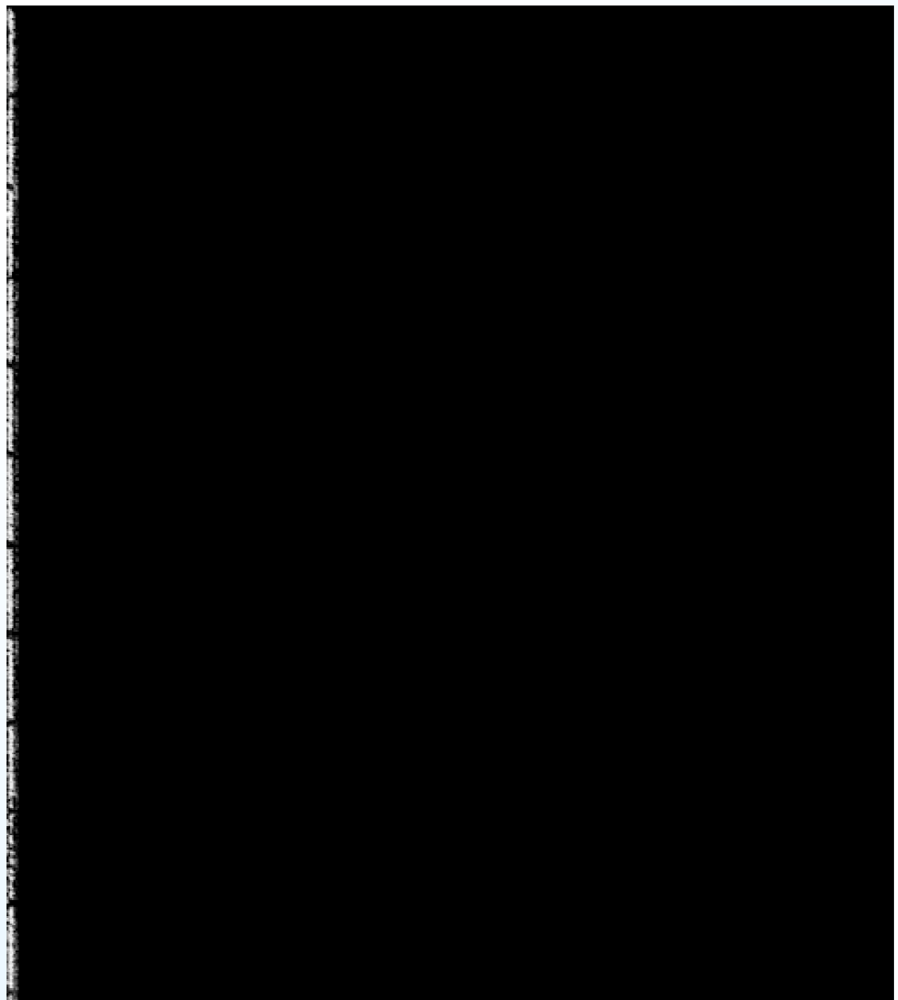
Method

3. For each range, find row-separating lines:



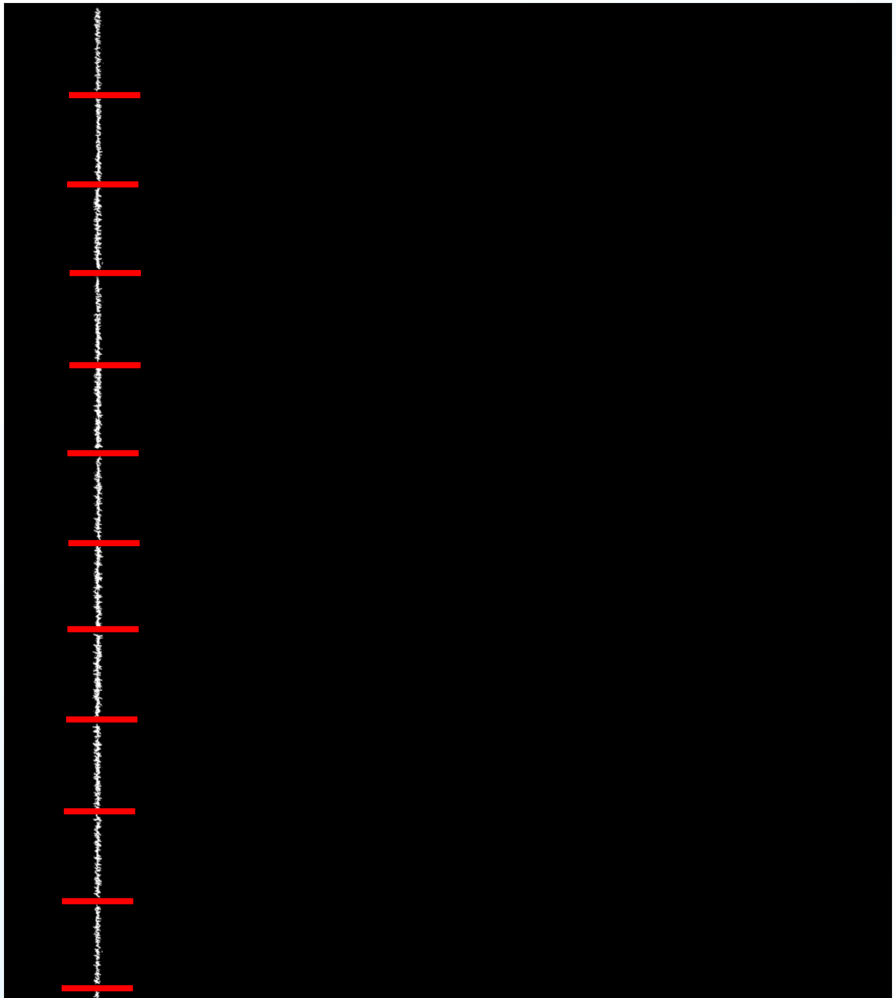
Method

4. Select the *n*-th row of each range



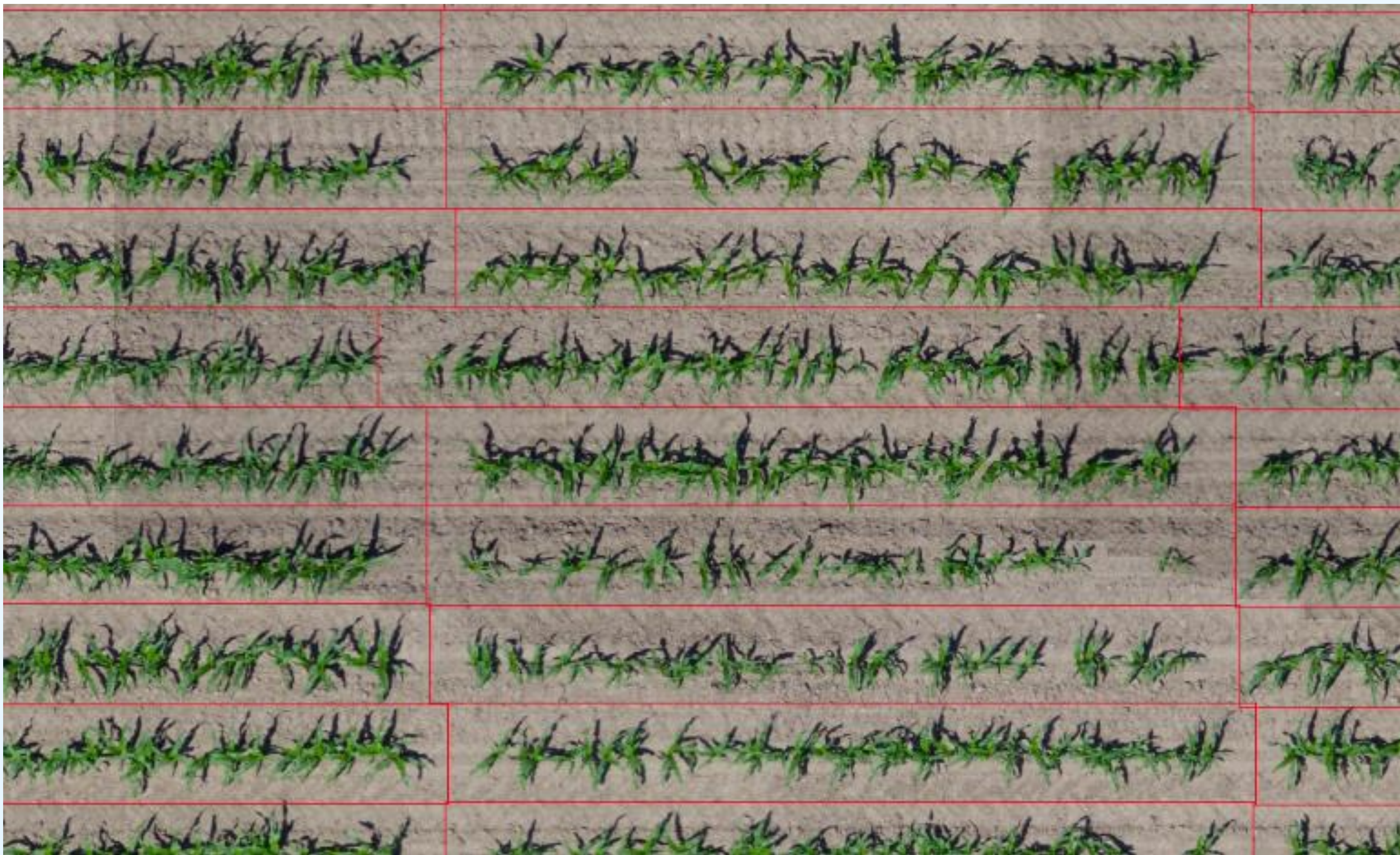
Method

4. **Find range-separating lines for the *n*-th row**



Resulting Cropped Images

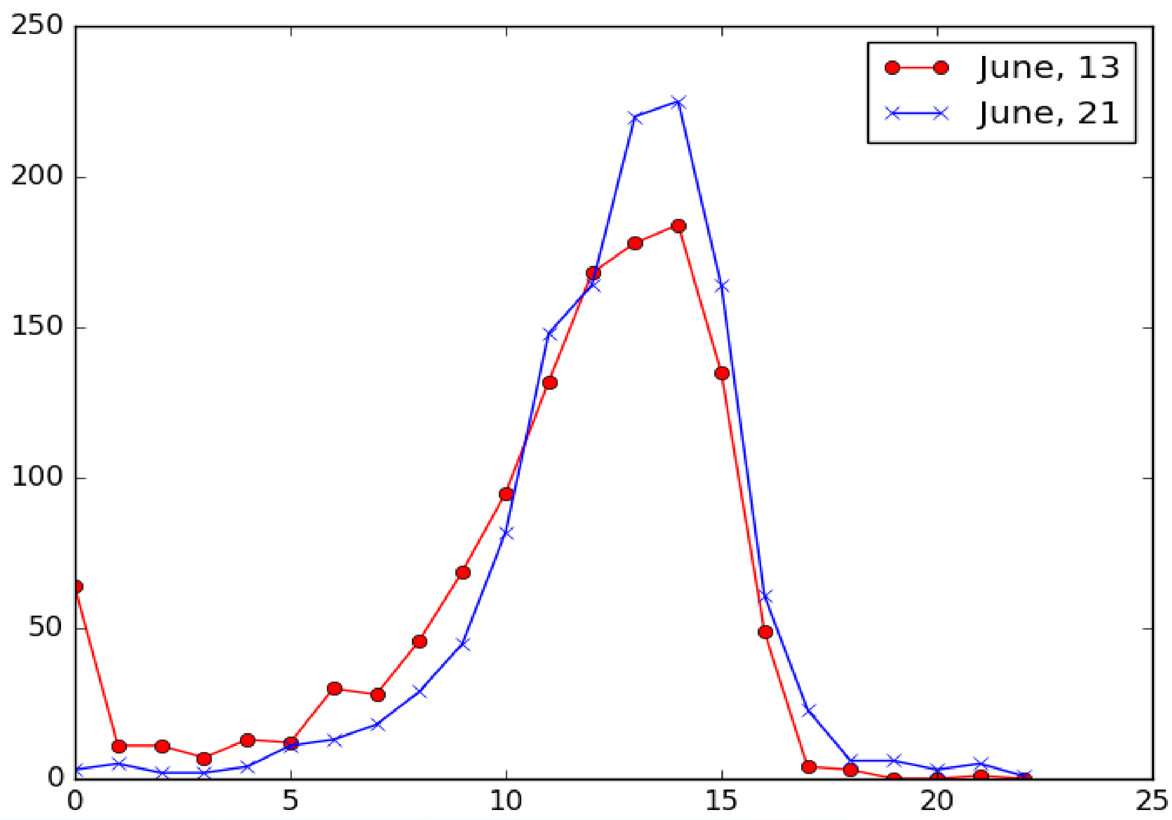
Resulting Cropped Images



Dataset

- **We groundtruthed 2,480 labeled images**
  - **80% for training, 10% for validation, 10% for testing**

 # images

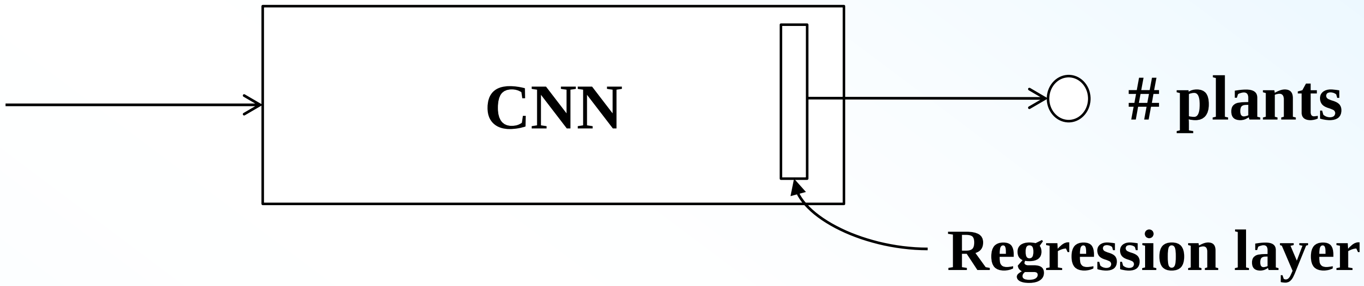


Plant count

Example image
Plant count: 15

2. Plant Counting

Counting Plants With Deep Learning



Compared CNNs:
- **AlexNet-v2**
- **Inception-v2**
- **Inception-v3**
- **Inception-v4**

With minimal modification to adapt to image size

Cost Function

- **Most research uses cross entropy as cost function, which reduces to**

  $$H(p,q) = -\log q(C)$$

  where $q(x)$ are the activations of the last layer, and C is the true number of plants
- **This cost function is not appropiate when the classes are not independent, and there is label noise**
- **We want to count, not classify**
- **We propose to use the L_p norm**

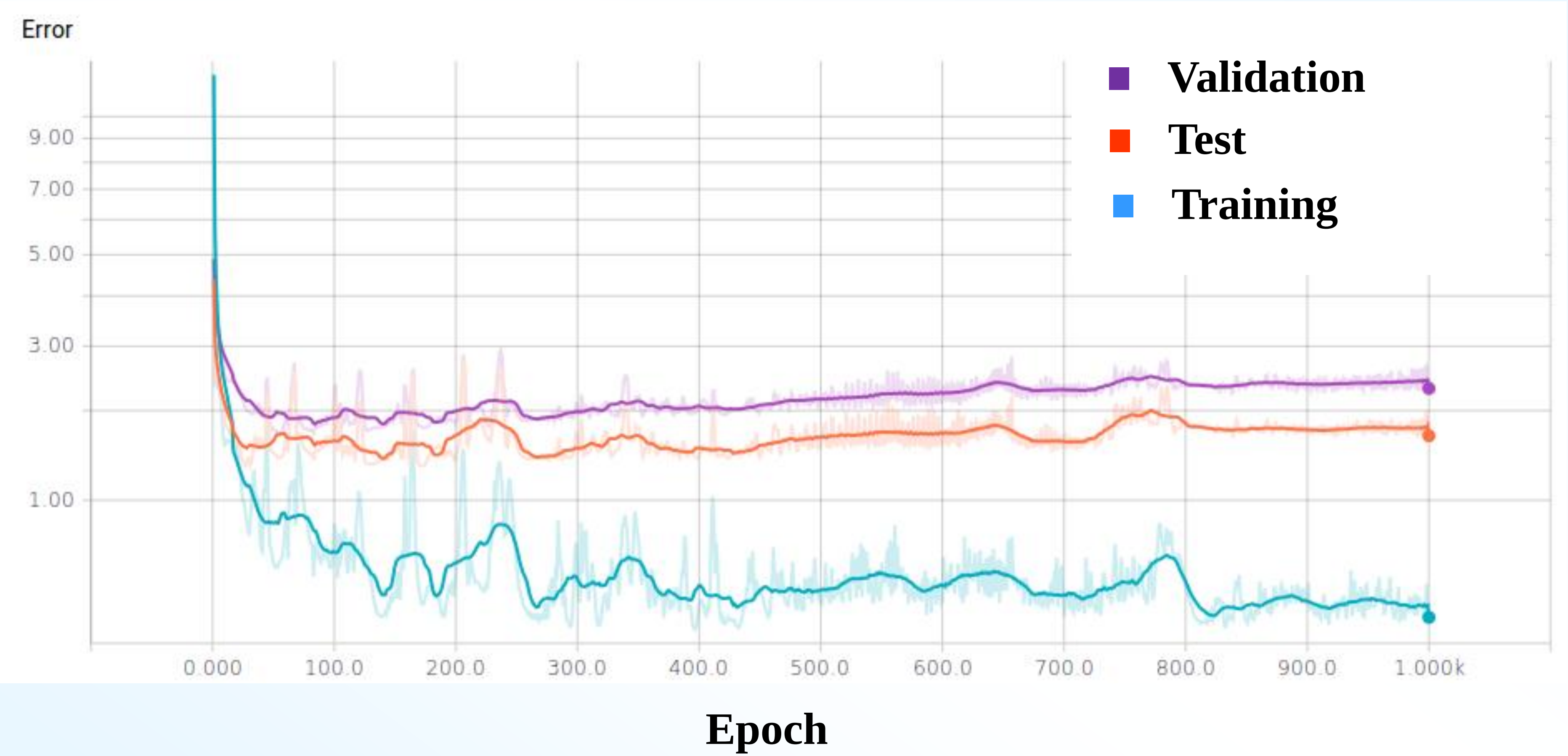
  $$L_p(x,\hat{x}) = \left| x - \hat{x} \right|^p$$

  and test which value of p provides the lowest error

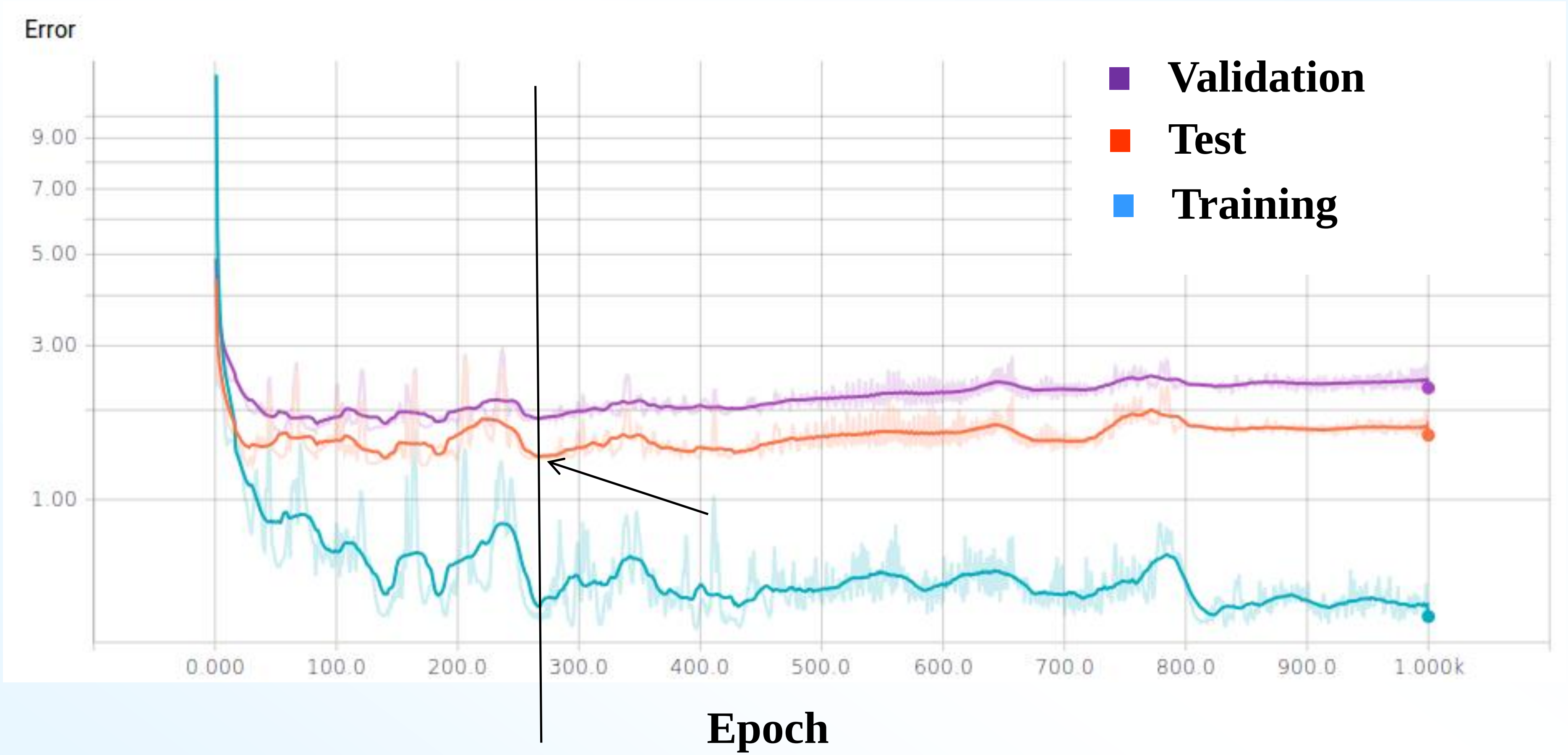
Network Architectures

- We examined several CNN architectures:
  - Alexnet
  - Inception-v2
  - Inception-v3
  - Inception-v4
- We modify the last layers to be able to process non-rectangular images (of size 546×103)

Stopping Criteria



Stopping Criteria



Value For *p*

- **Our metric for testing is Mean Average Percentage Error**

$$MAPE = 100\overline{\frac{|\hat{x} - C|}{C}}$$

- **Effect of *p* on the error, evaluated using AlexNet**

p	MAPE
2	8.2%
1.8	8.4%
1.5	8.5 %
1	7.9 % ←

Results

- **Performance of different architectures was evaluated**

Network	MAPE (w/o data augm)	MAPE (w/ data augm)
AlexNet	8.3%	7.9%
Inception-v2	8.2%	6.7%
Inception-v3	7.1%	6.7%
Inception-v4	12.4%	11.4%

←

- **(using *p*=1)**

Conclusions

- We presented a CNN-based method to count plants without locating them
- We presented a method to segment (or extract) image sections, or plots, from an orthorectified image

- Future work will include investigating loss functions more stable than the L1, such as the smooth L1, and training with larger datasets