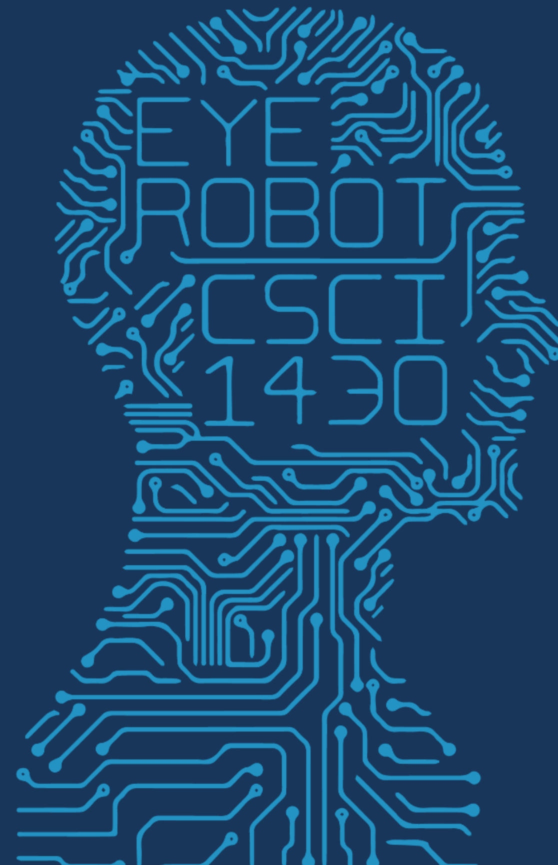
1950

Future Vision



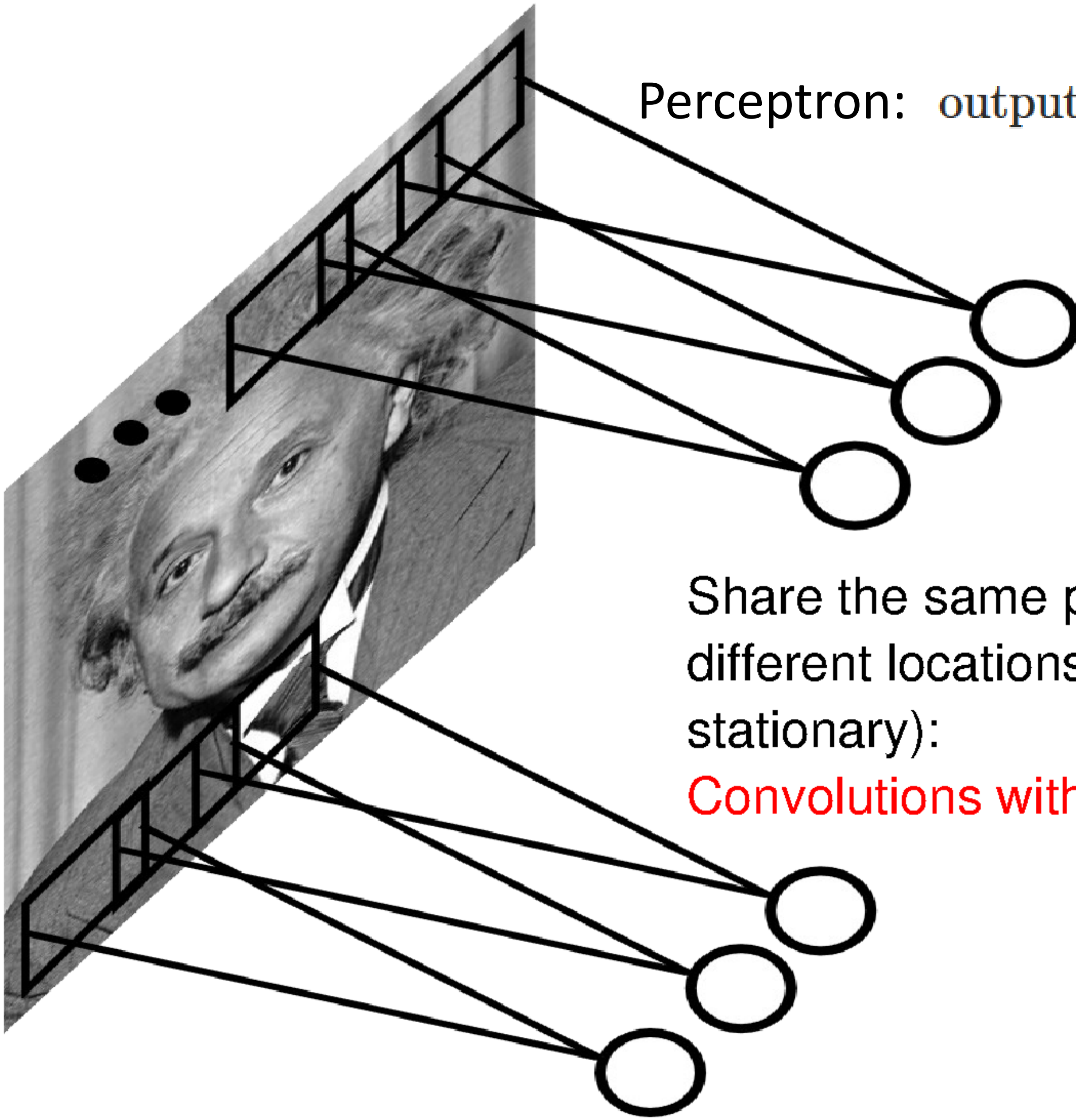
2017 MWF 1pm 368

Computer Vision

Russell, 2009

Simultaneous contrast

Convolutional Layer



Perceptron: $\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$

$w \cdot x \equiv \sum_j w_j x_j,$

This is convolution!

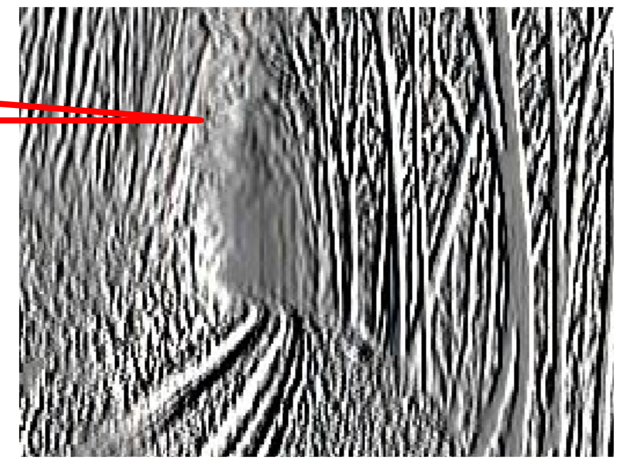
Share the same parameters across different locations (assuming input is stationary):
Convolutions with learned kernels

Ranzato

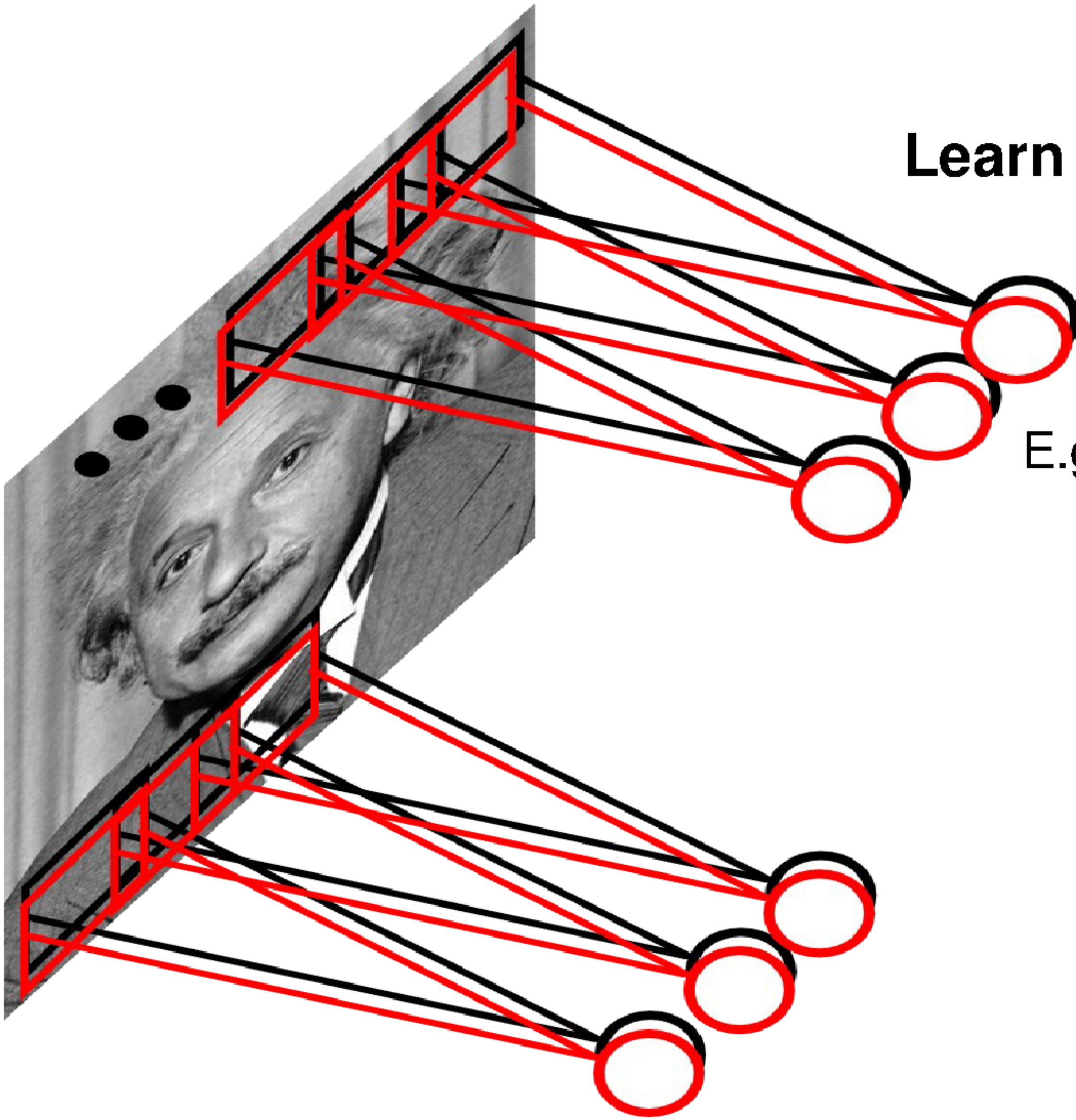
Convolutional Layer

$$* \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} =$$

Shared weights



Convolutional Layer

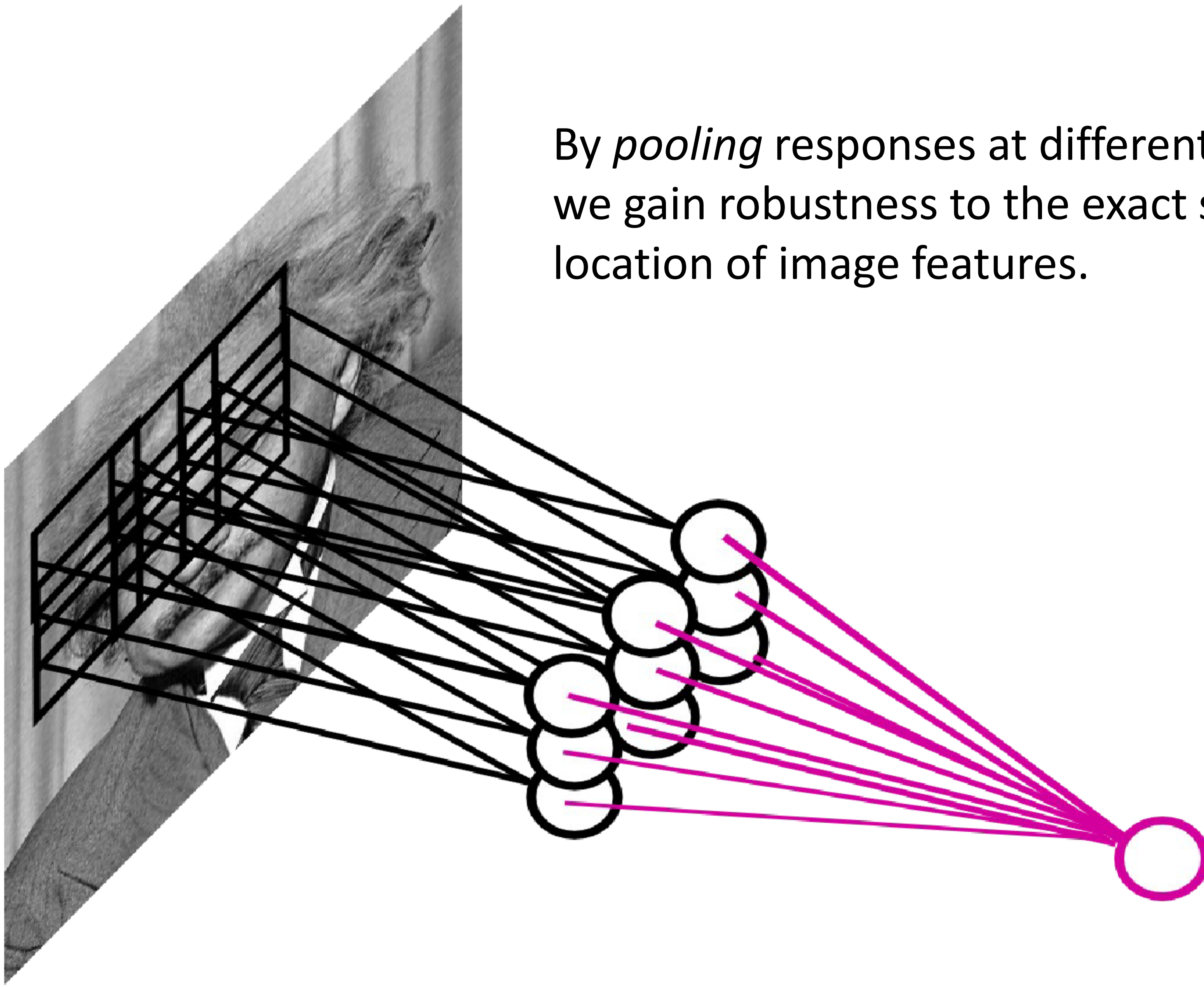


Learn multiple filters.

Filter = 'local' perceptron.
Also called *kernel.*

E.g.: 200x200 image
100 Filters
Filter size: 10x10
10K parameters

Pooling Layer

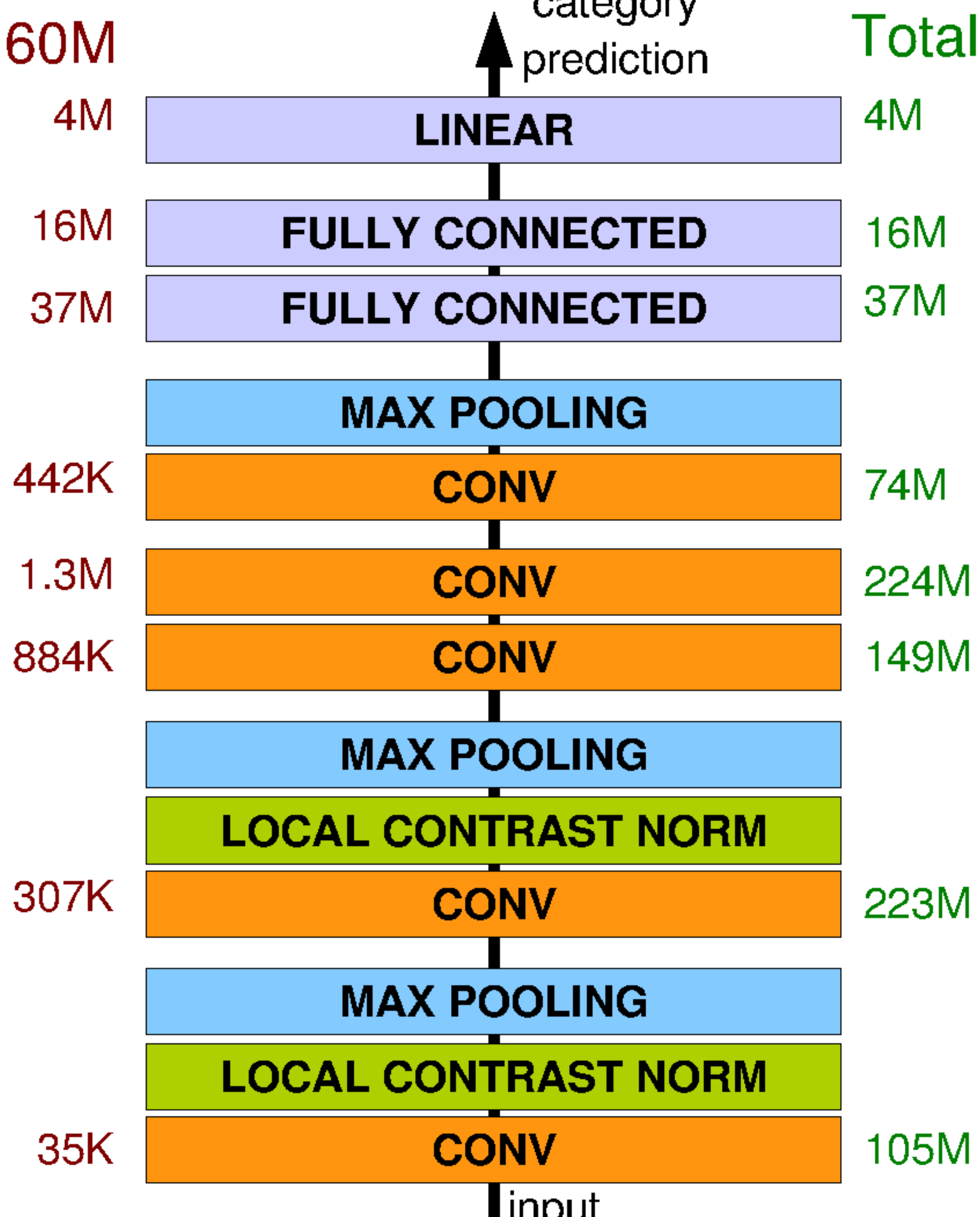


By *pooling* responses at different locations, we gain robustness to the exact spatial location of image features.

Architecture for Classification

Total nr. params: 60M

Total nr. flops: 832M



Krizhevsky et al. "ImageNet Classification with deep CNNs" NIPS 2012

Universality

- A single-layer network can learn any function:
  - So long as it is differentiable
  - To some approximation;
    More perceptrons = a better approximation

- Visual proof (Michael Nielson):

http://neuralnetworksanddeeplearning.com/chap4.html

If a single-layer network can learn any function…

- …given enough parameters…
- …then why do we go deeper?

Intuitively, composition is efficient because it allows reuse.

Empirically, deep networks do a better job than shallow networks at learning such hierarchies of knowledge.

Problem of fitting

- Too many parameters = overfitting
- Not enough parameters = underfitting

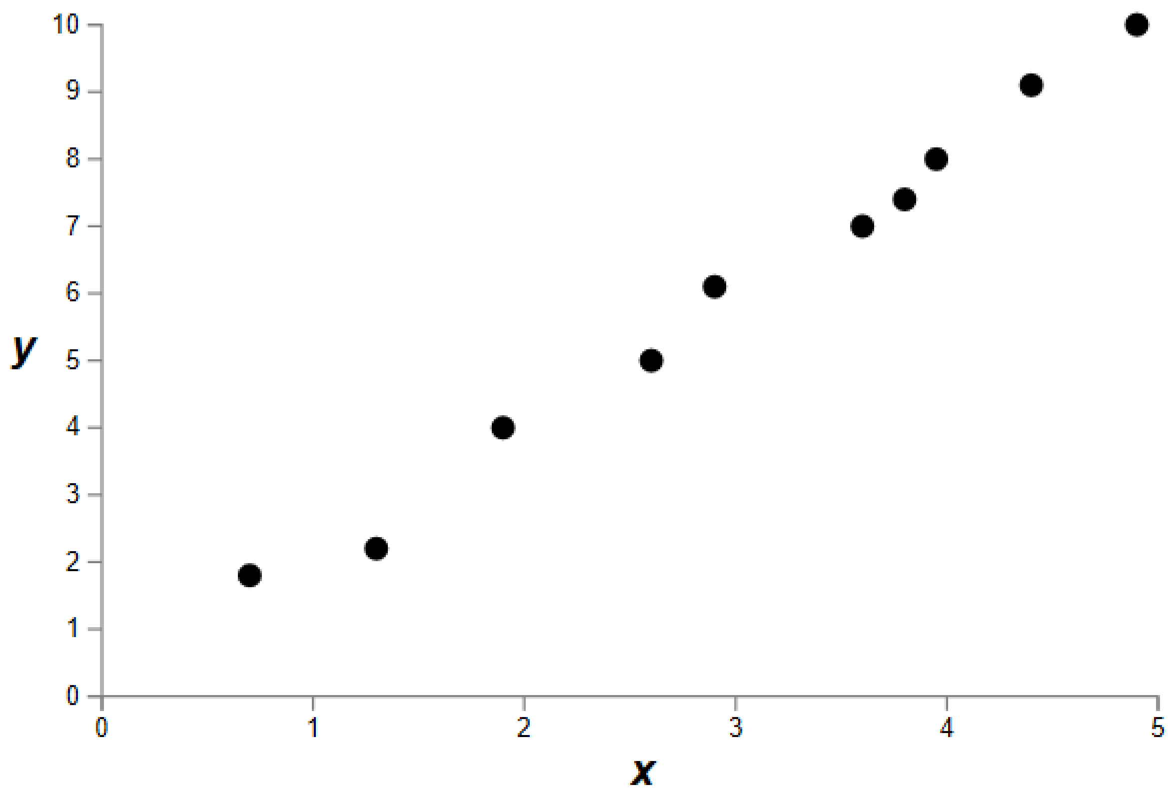
- More data = less chance to overfit

- How do we know what is required?

Regularization

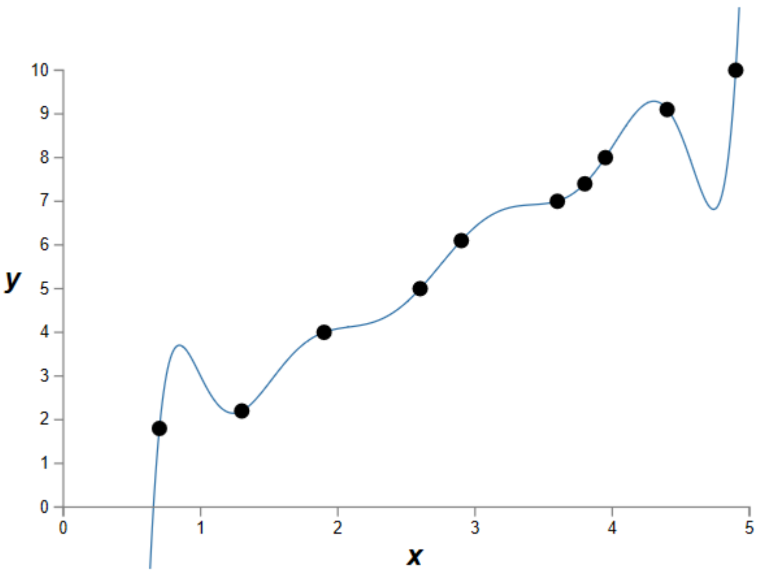
- Attempt to guide solution to *not overfit*
- But still give freedom with many parameters

Data fitting problem

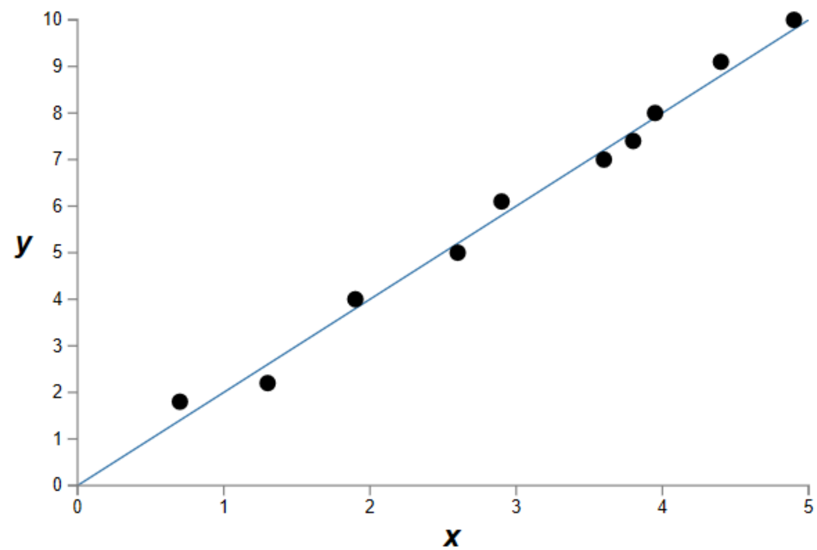


[Nielson]

Which is better?
Which is better *a priori*?



9th order polynomial



1st order polynomial

[Nielson]

Regularization

- Attempt to guide solution to *not overfit*
- But still give freedom with many parameters

- Idea: Penalize the use of parameters to prefer small weights.

Regularization:

- Idea: add a cost to having high weights
- λ = regularization parameter

$$C = C_0 + \frac{\lambda}{2n} \sum_w w^2,$$

[Nielson]

Both can describe the data…

- …but one is simpler.

- Occam's razor:

  "Among competing hypotheses, the one with the fewest assumptions should be selected"

For us:

Large weights cause large changes in behaviour in response to small changes in the input.

Simpler models (or smaller changes) are more robust to noise.

Regularization

- Idea: add a cost to having high weights
- λ = regularization parameter

$$C = C_0 + \frac{\lambda}{2n} \sum_w w^2,$$

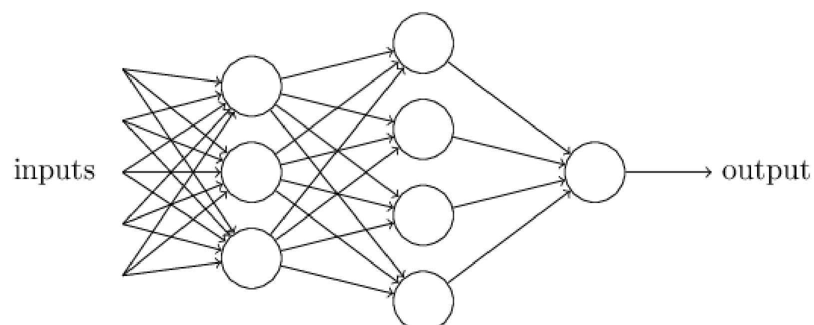
$$C = \underbrace{-\frac{1}{n} \sum_{xj} \left[y_j \ln a_j^L + (1 - y_j) \ln(1 - a_j^L) \right]}_{\text{Normal cross-entropy loss (binary classes)}} + \underbrace{\frac{\lambda}{2n} \sum_w w^2}_{\text{Regularization term}}.$$

[Nielson]

Regularization: Dropout

- Our networks typically start with random weights.
- Every time we train = slightly different outcome.

- Why random weights?
- If weights are all equal, response across filters will be equivalent.
  - Network doesn't train.



$$w \cdot x \equiv \sum_j w_j x_j,$$

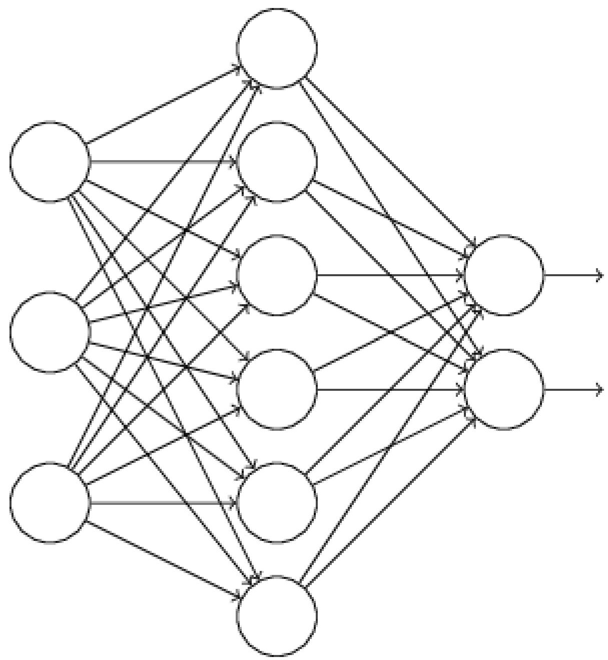
[Nielson]

Regularization

- Our networks typically start with random weights.
- Every time we train = slightly different outcome.

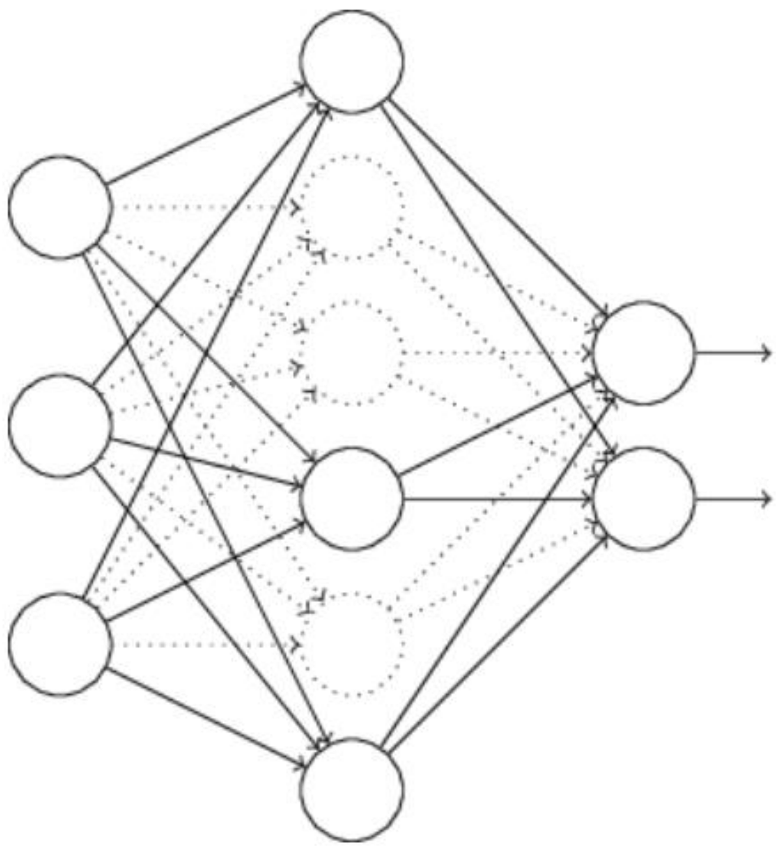
- Why not train 5 different networks with random starts and vote on their outcome?
  - Works fine!
  - Helps generalization because error is averaged.

Regularization: Dropout



[Nielson]

Regularization: Dropout



At each mini-batch:
- Randomly select a subset of neurons.
- Ignore them.

On test: half weights outgoing to compensate for training on half neurons.

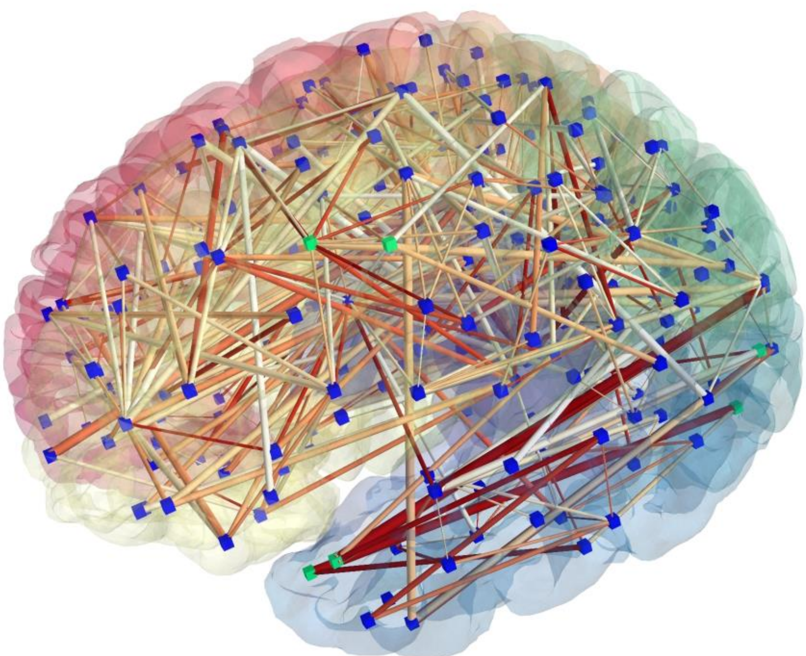
Effect:
- Neurons become less dependent on output of connected neurons.
- Forces network to learn more robust features that are useful to more subsets of neurons.
- Like averaging over many different trained networks with different random initializations.
- Except cheaper to train.

[Nielson]

More regularization

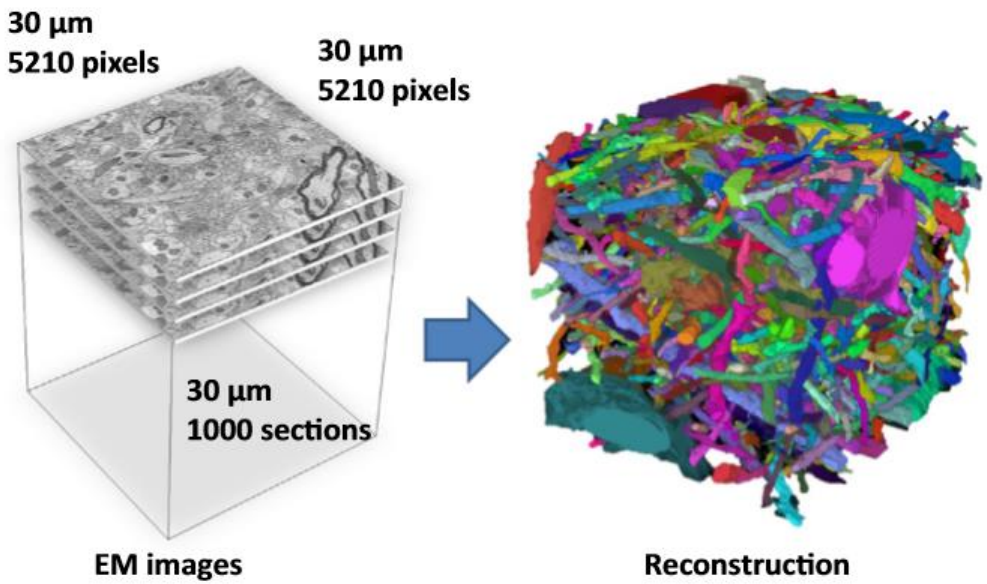
- Adding more data is a kind of regularization
- Pooling is a kind of regularization
- Data augmentation is a kind of regularization

CNNs for Medical Imaging - Connectomics



[Patric Hagmann]

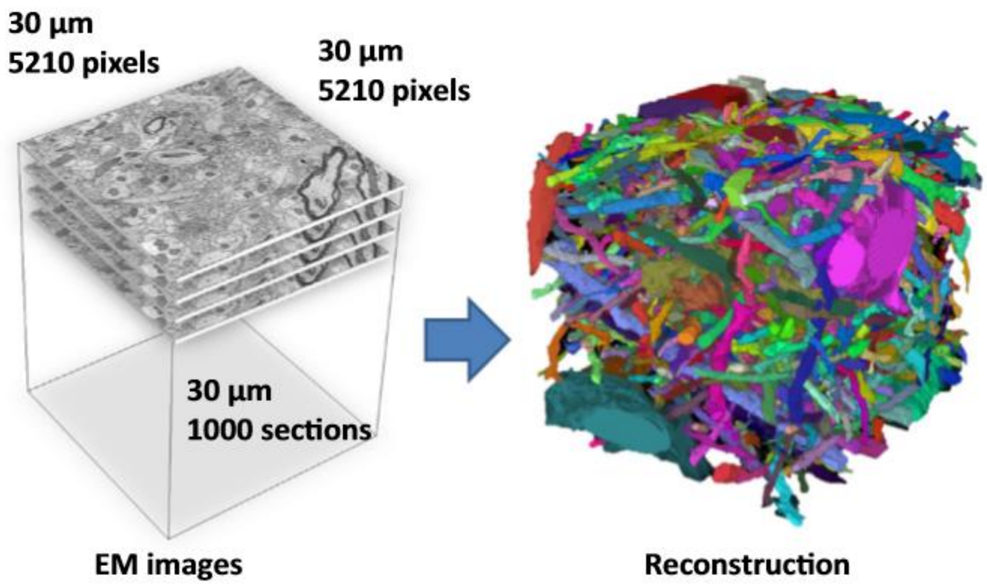
Vision for understanding the brain



- 1mm cubed of brain
- Image at 5-30 nanometers
- How much data?

[Kaynig-Fittkau et al.]

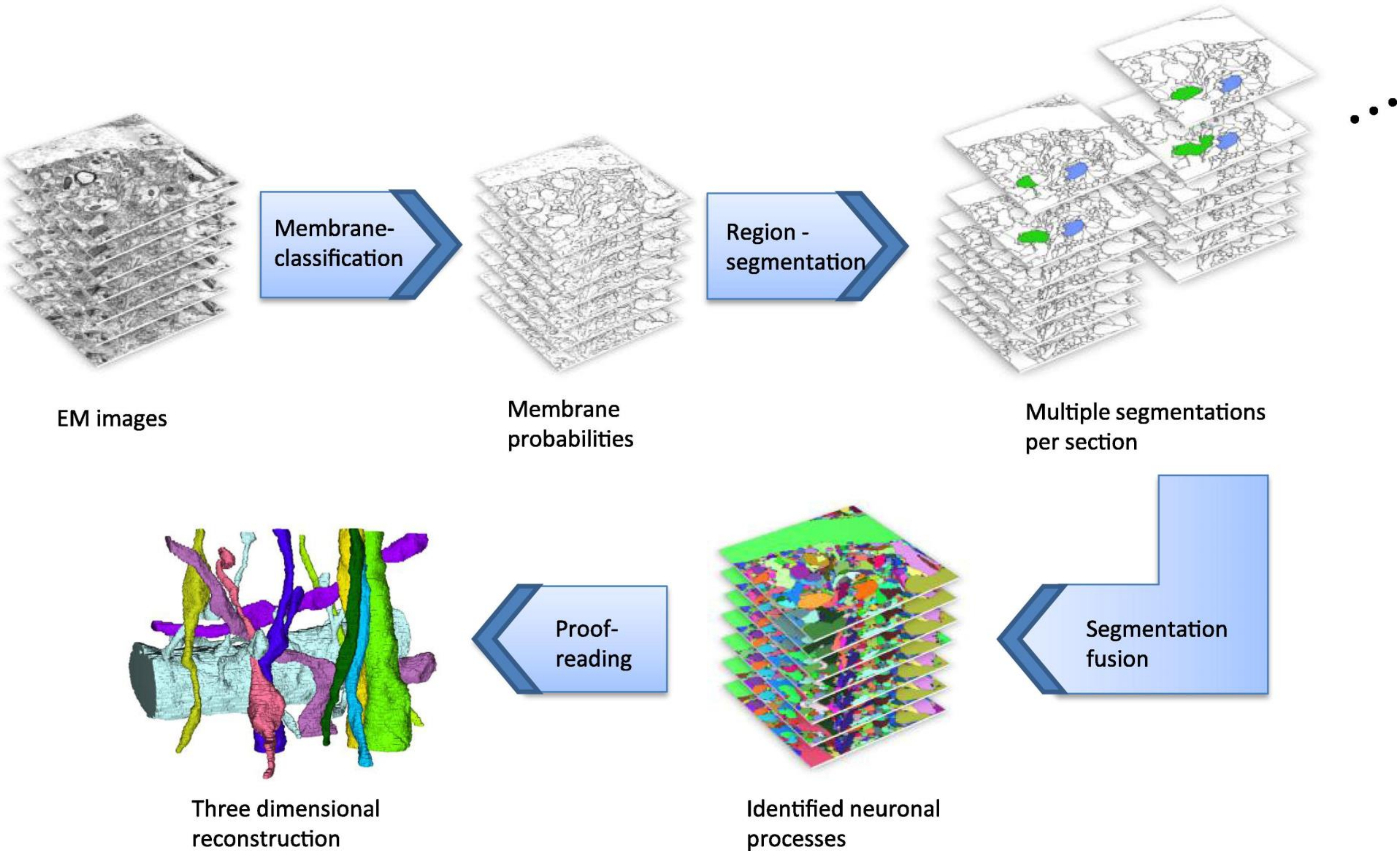
Vision for understanding the brain



- 1mm cubed of brain
- Image at 5-30 nanometers
- How much data?
- 1 Petabyte – 1,000,000,000,000,000

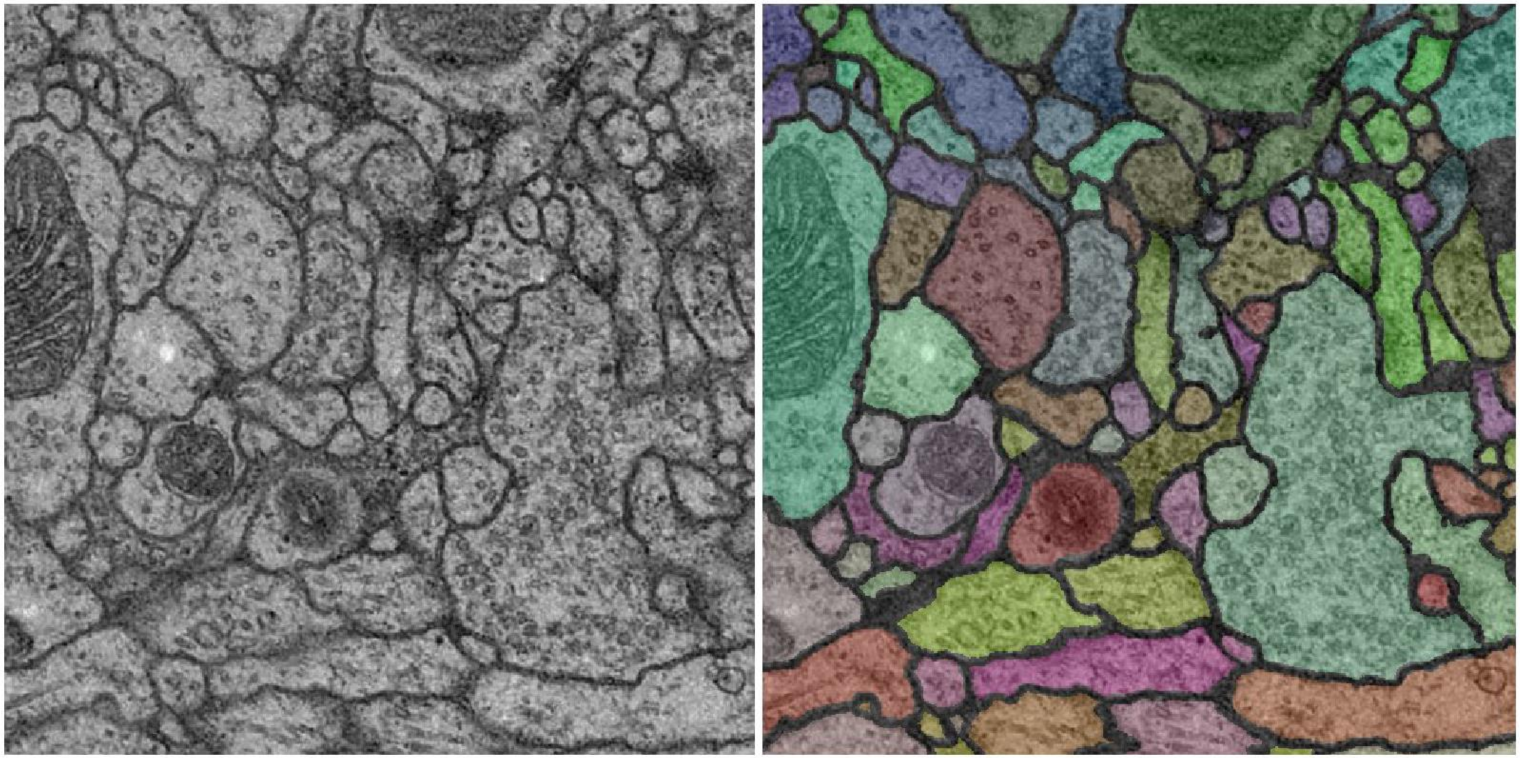
~ All photos uploaded to Facebook per day

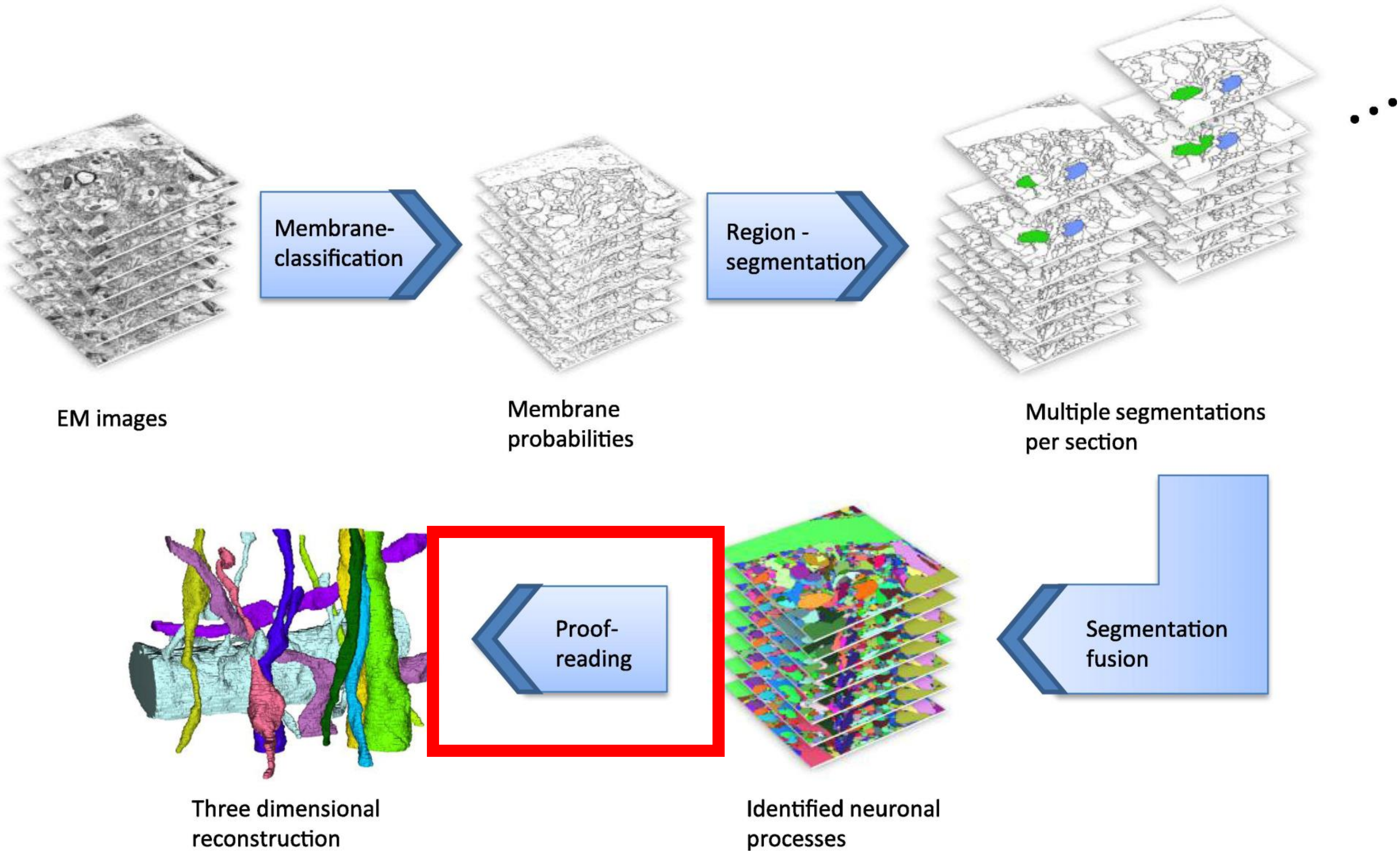
[Kaynig-Fittkau et al.]



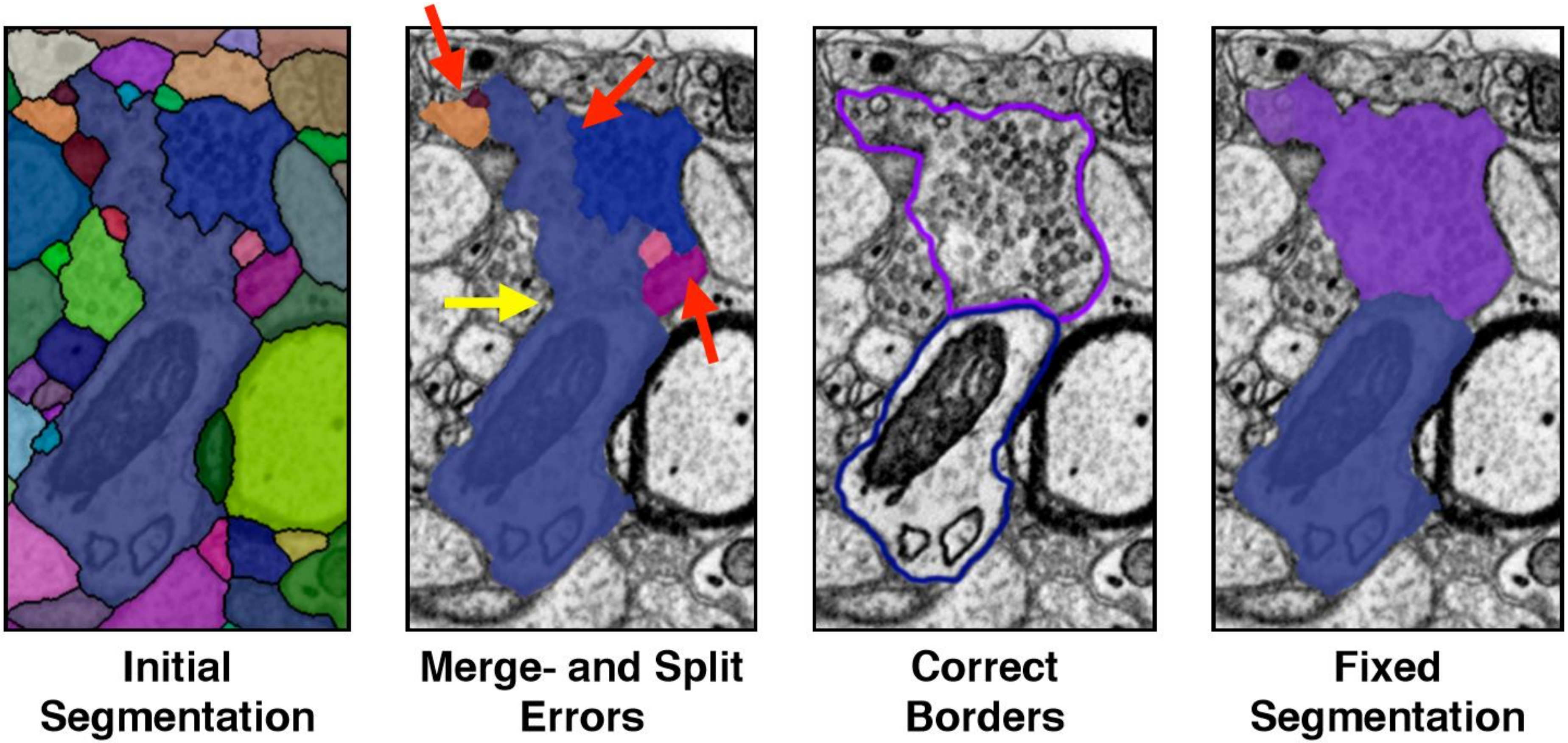
[Kaynig-Fittkau et al.]

Vision for understanding the brain



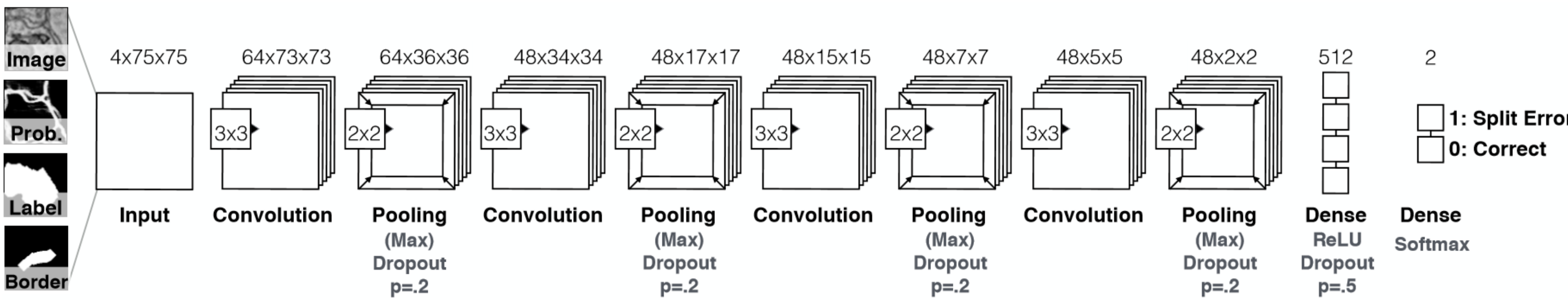


[Kaynig-Fittkau et al.]

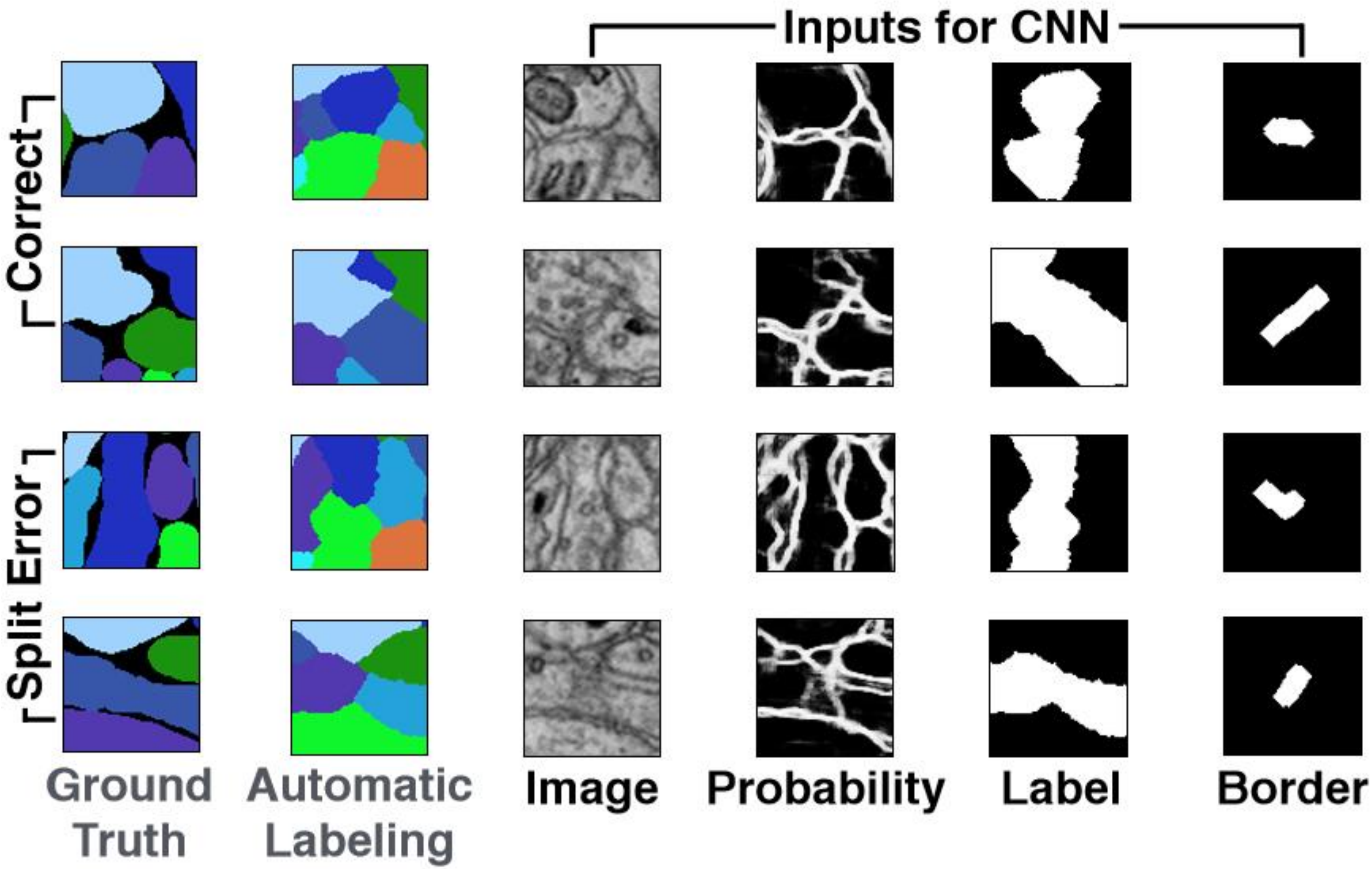

Initial Segmentation
Merge- and Split Errors
Correct Borders
Fixed Segmentation

[Haehn et al.]

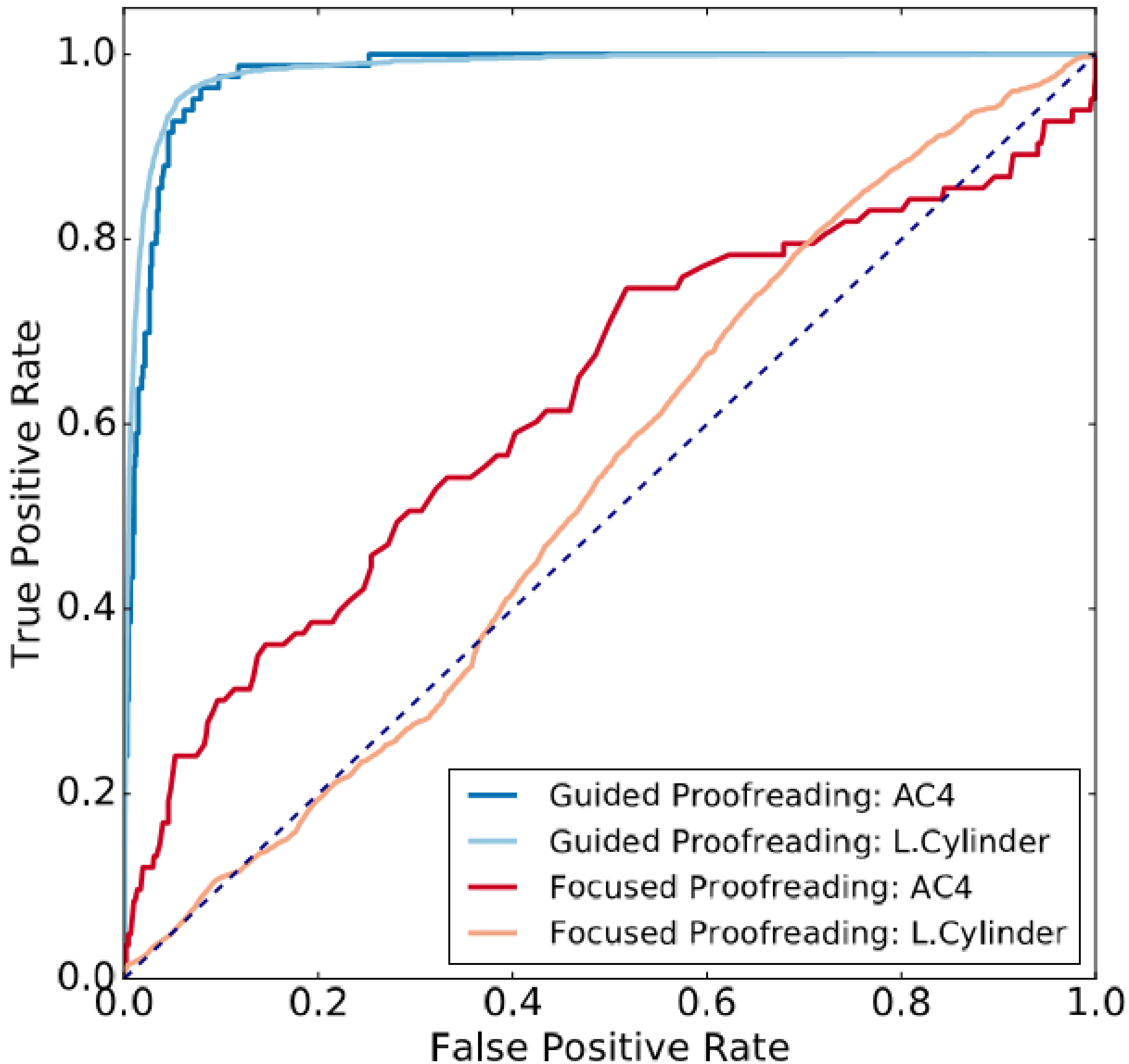
Network Architecture



[Haehn et al.]


Inputs for CNN
Correct
Split Error
Ground Truth
Automatic Labeling
Image
Probability
Label
Border

[Haehn et al.]


1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
Guided Proofreading: AC4
Guided Proofreading: L.Cylinder
Focused Proofreading: AC4
Focused Proofreading: L.Cylinder

[Haehn et al.]