Deep Learning for Data Science
DS 542

https://dl4ds.github.io/fa2025/

Residual Networks

Original slides by Tom Gardos,
other content from Understanding Deep Learning unless otherwise cited

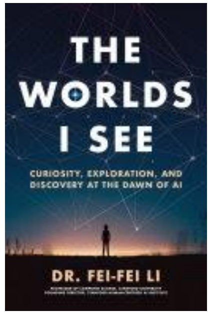
Plan for Today

- Finish up CNN examples (from last time)
- Residual networks

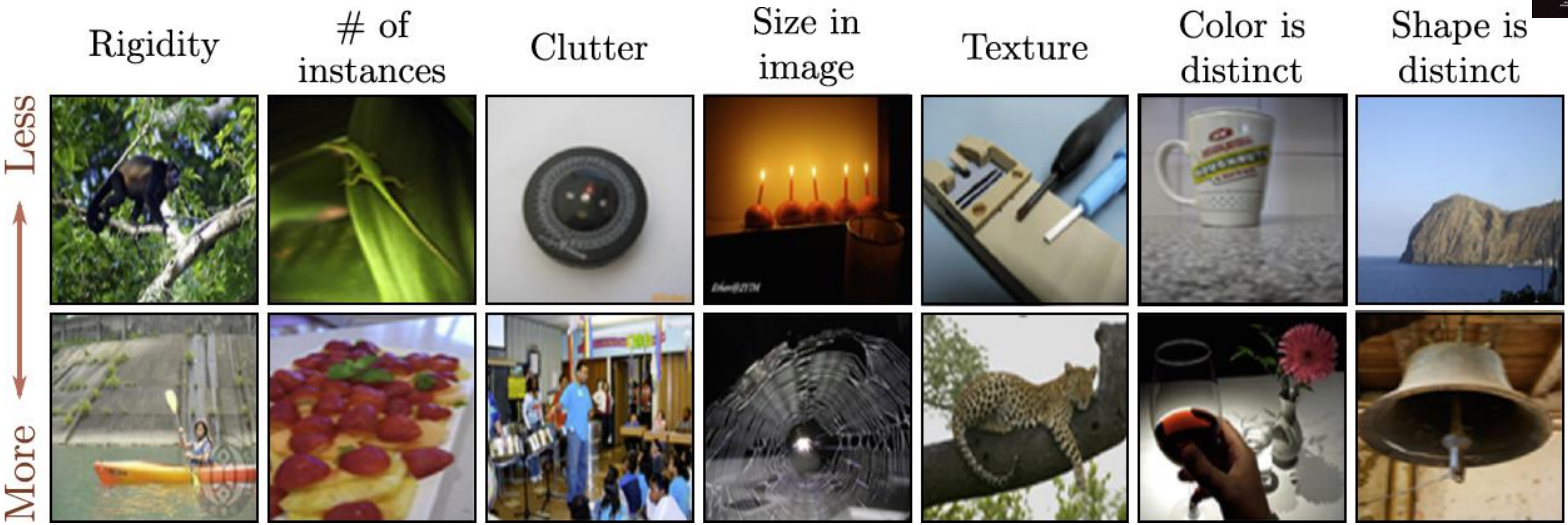
Project 1:

- Image classification
- Object detection
- Semantic segmentation

Fei-Fei Li

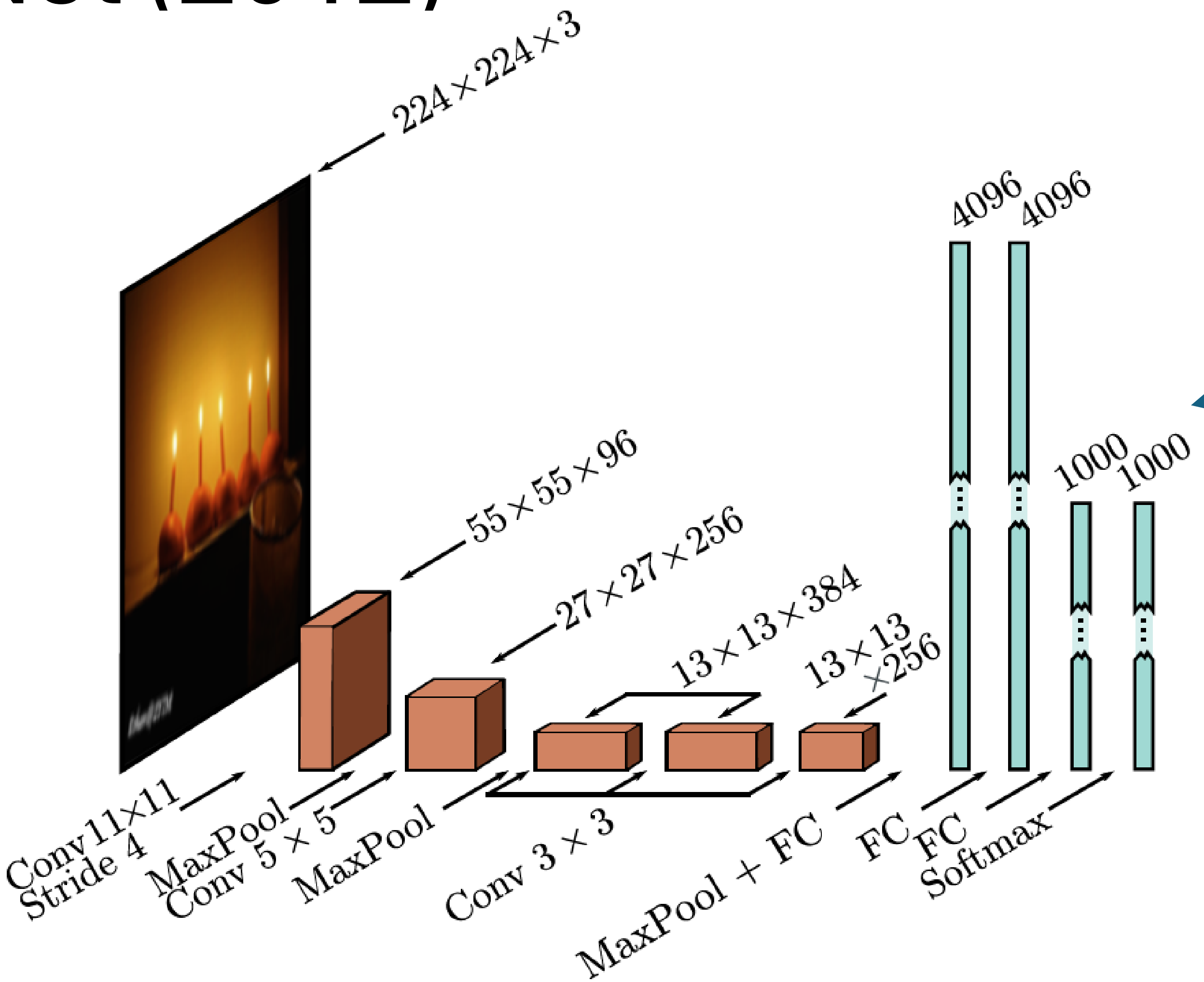


ImageNet 1K database



- 224 x 224 images
- 1,281,167 training images, 50,000 validation images, and 100,000 test images
- 1000 classes

AlexNet (2012)



Almost all the 60 million parameters parameters are in fully connected layers

A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2012, doi: 10.1145/3065386.

AlexNet (2012)

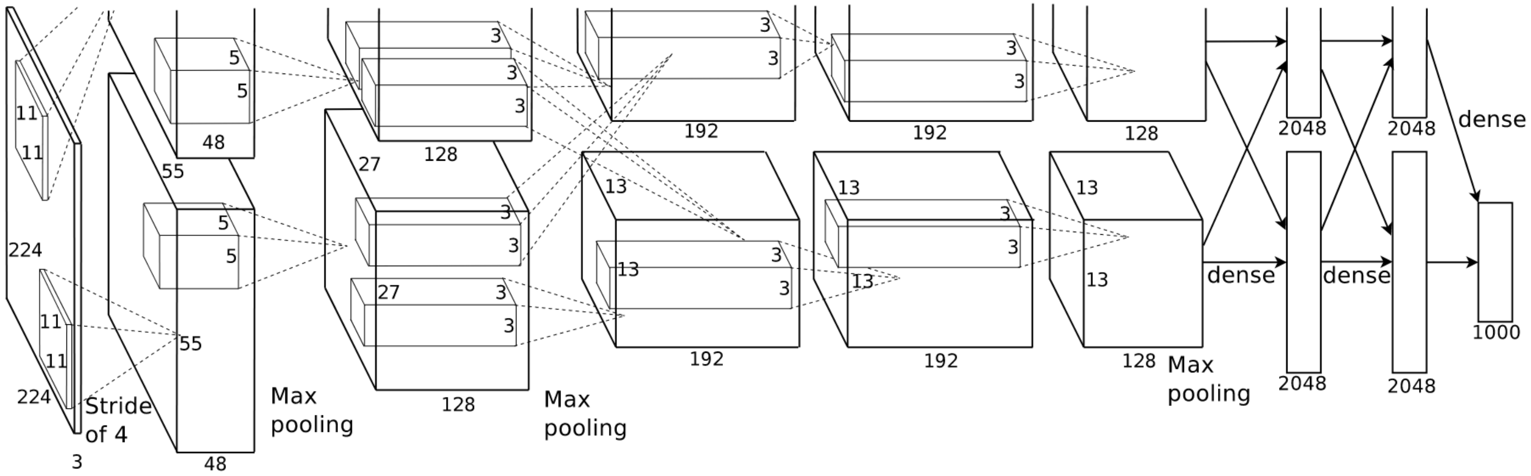


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

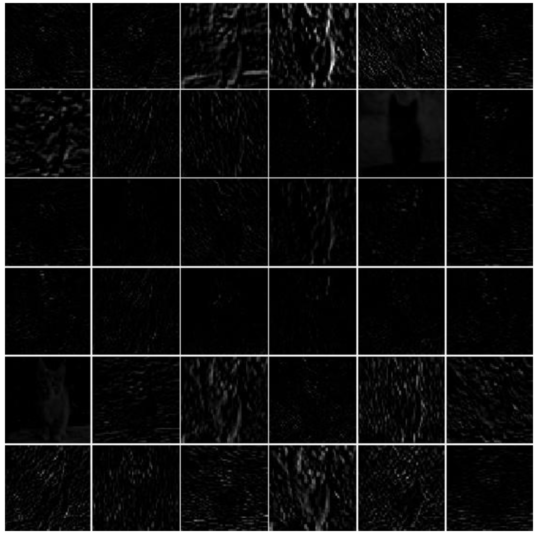
Won the 2012 Large-Scale Vision Recognition Challenge (ILSVRC) by a big margin.

Took between five and six days to train on two GTX 580 3GB GPUs with manually optimized compute kernels.

A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2012, doi: 10.1145/3065386.

AlexNet

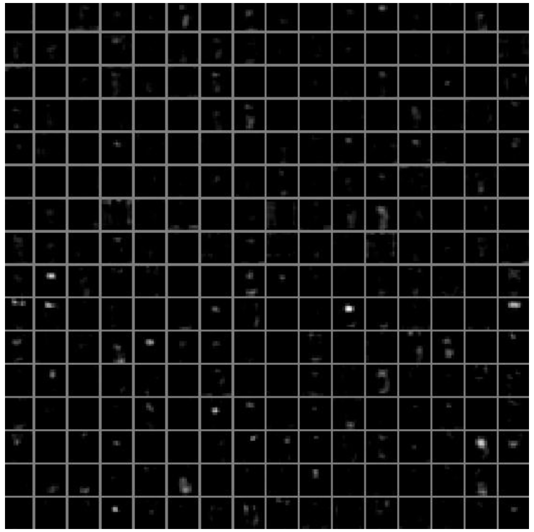
Activations (feature maps)



1st Layer

• • •

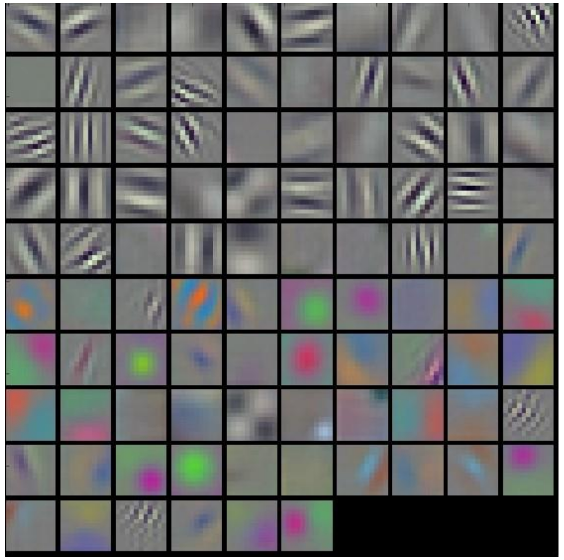
2nd Layer

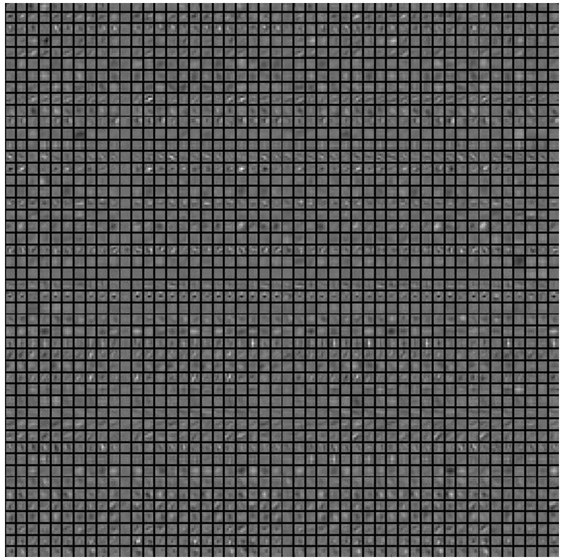


5th Layer

Cat image input
(not actual image)

Filter Kernels





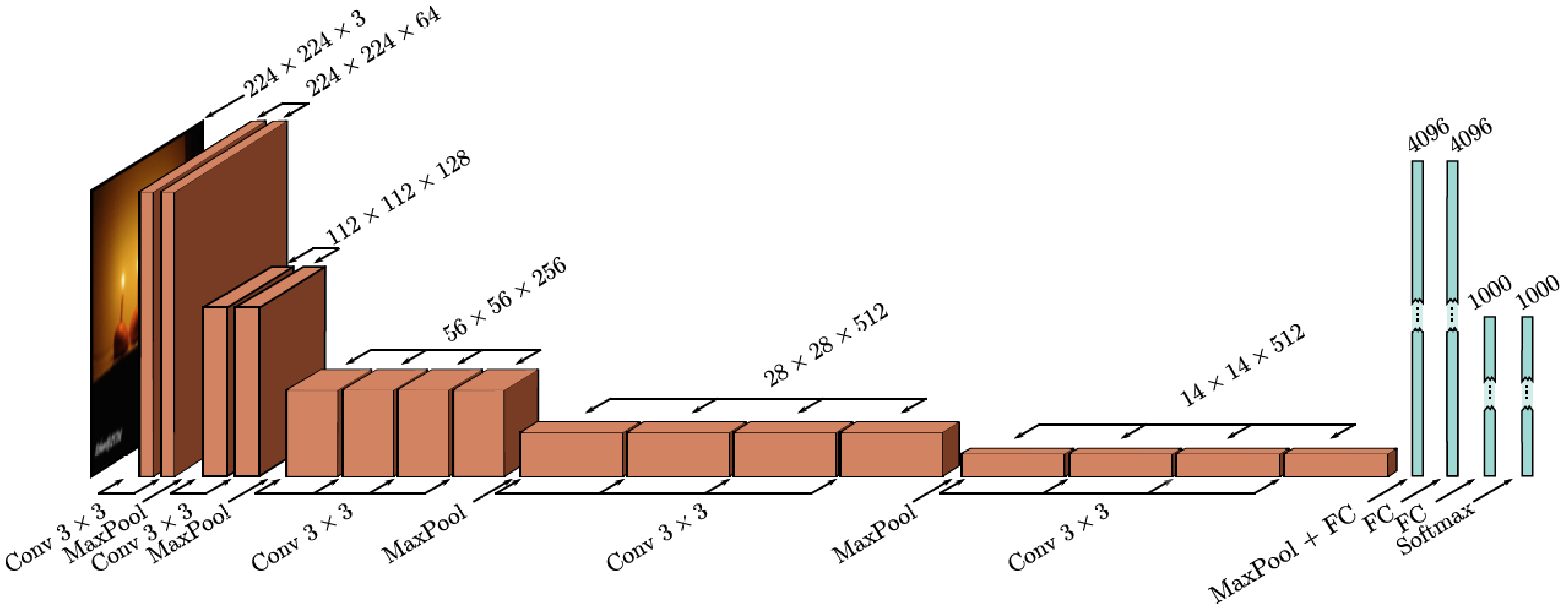
• • •

https://cs231n.github.io/understanding-cnn/

Details

- At test time average results from five different cropped and mirrored versions of the image
- SGD with a momentum coefficient of 0.9 and batch size of 128.
- L2 (weight decay) regularizer used.
- This system achieved a 16.4% top-5 error rate and a 38.1% top-1 error rate.

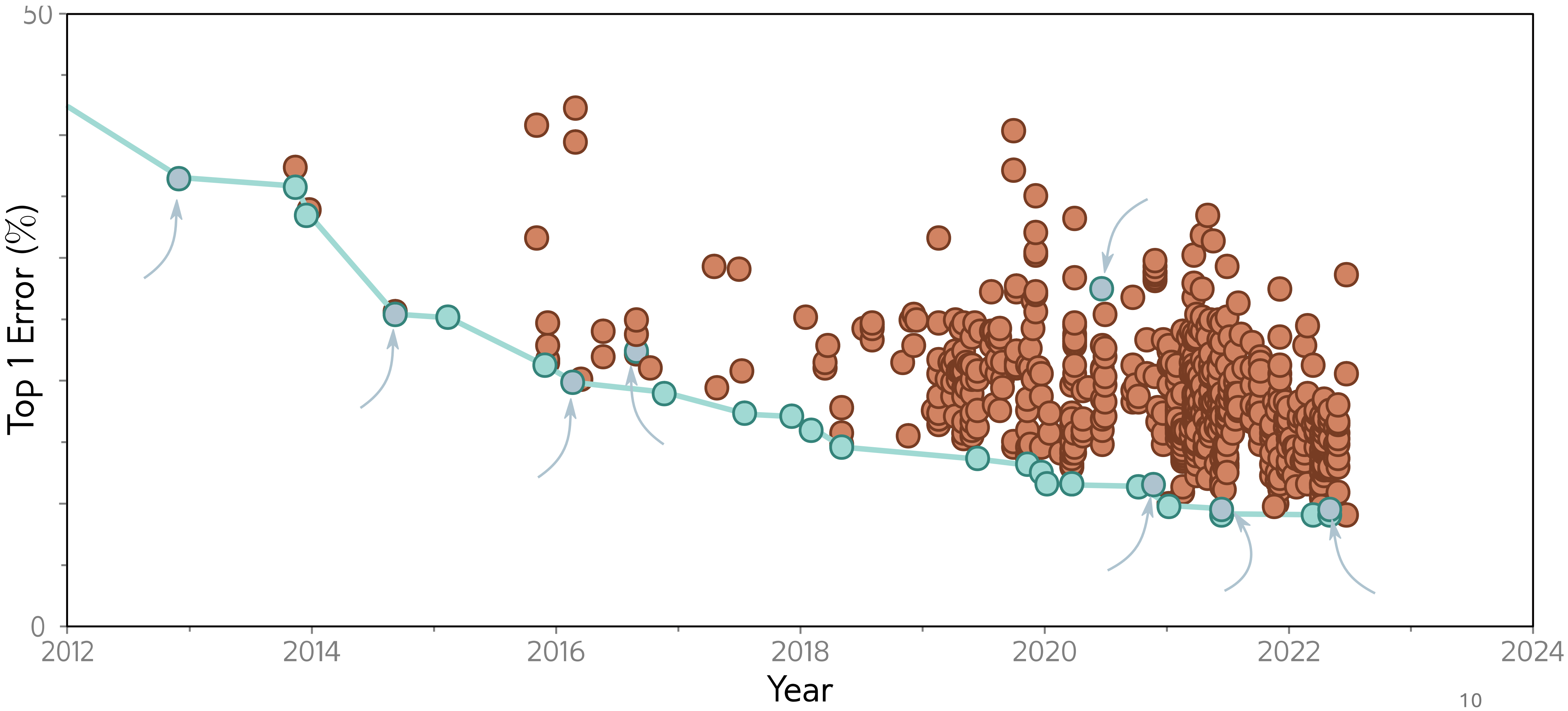
VGG (2015)


224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
4096
4096
1000
1000
Conv 3 × 3
MaxPool
Conv 3 × 3
MaxPool
Conv 3 × 3
MaxPool
Conv 3 × 3
MaxPool
Conv 3 × 3
MaxPool + FC
FC
FC
Softmax

Details

- 19 hidden layers
- 144 million parameters
- 6.8% top-5 error rate, 23.7% top-1 error rate

ImageNet History



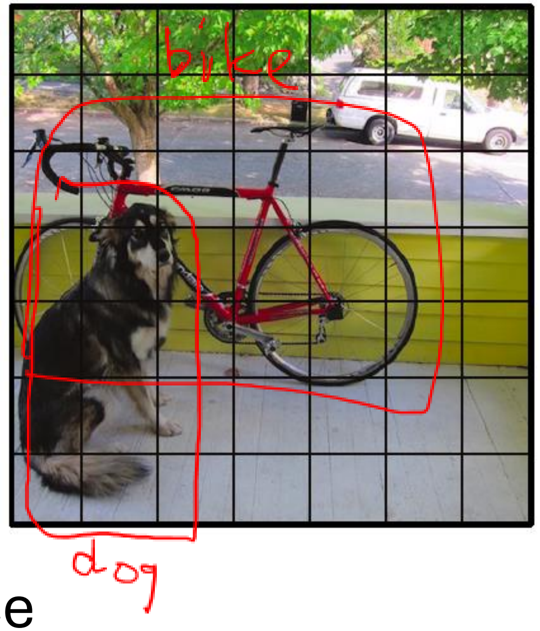
Any Questions?

???

Moving on

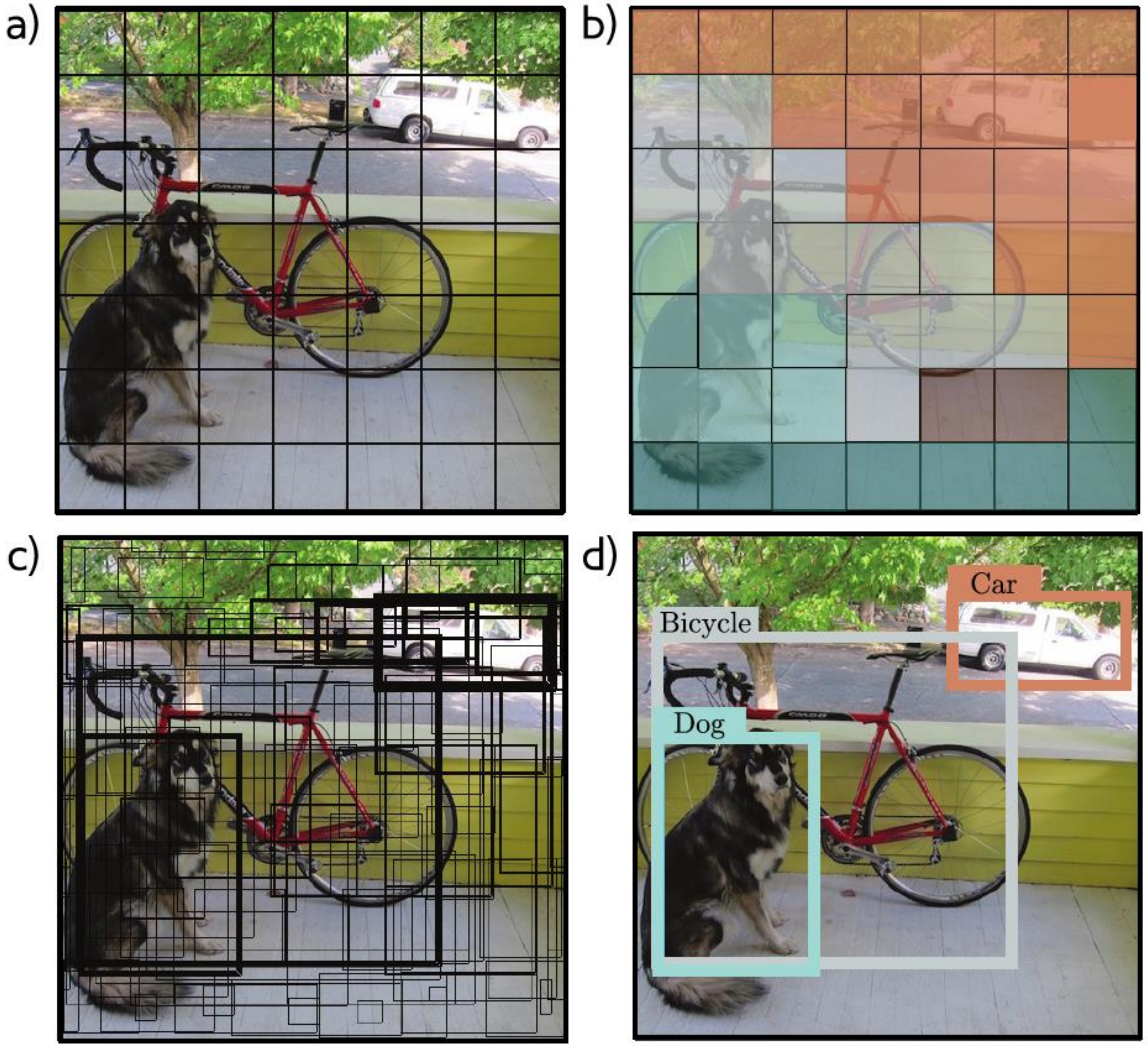
- Image classification
- Object detection
- Semantic segmentation

You Only Look Once (YOLO)

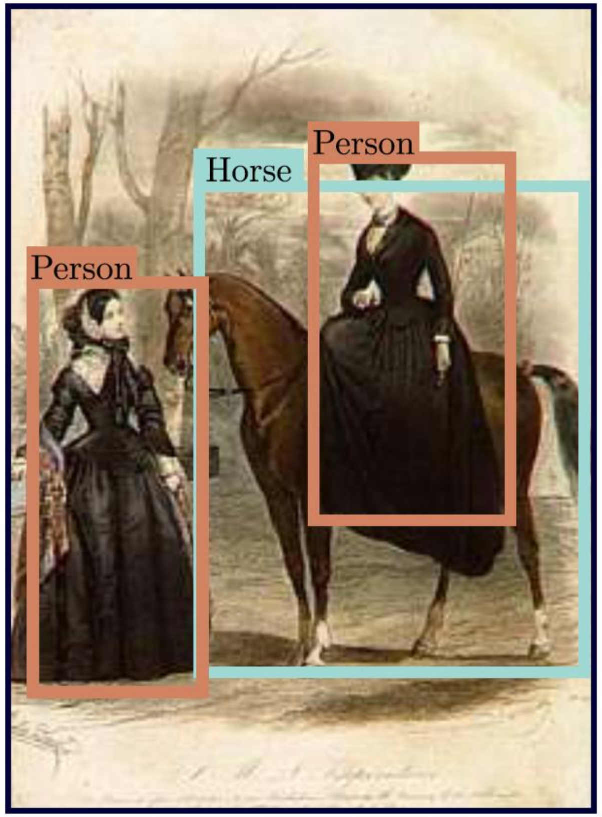


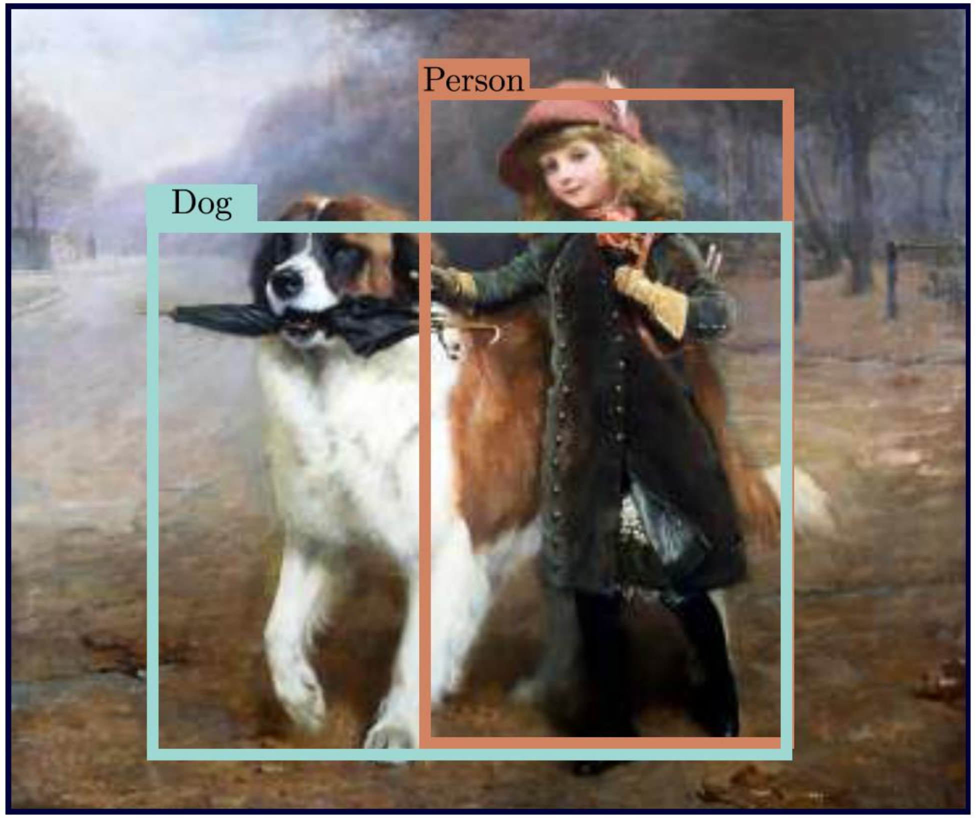
- Network similar to VGG (448x448 input)
- 7×7 grid of locations
- Predict class at each location
- Predict 2 bounding boxes at each location
  - Five parameters –x,y, height, width, and confidence
- Momentum, weight decay, dropout, and data augmentation
- Heuristic at the end to threshold and decide final boxes – (non maximum suppression)

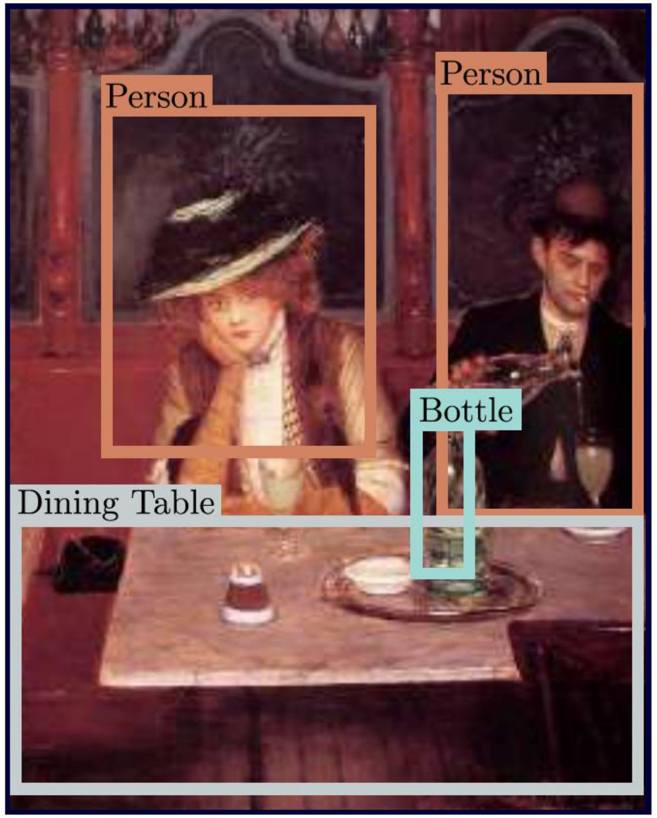
Object detection (YOLO)



Results


Person
Horse
Person


Person
Dog


Person
Person
Bottle
Dining Table

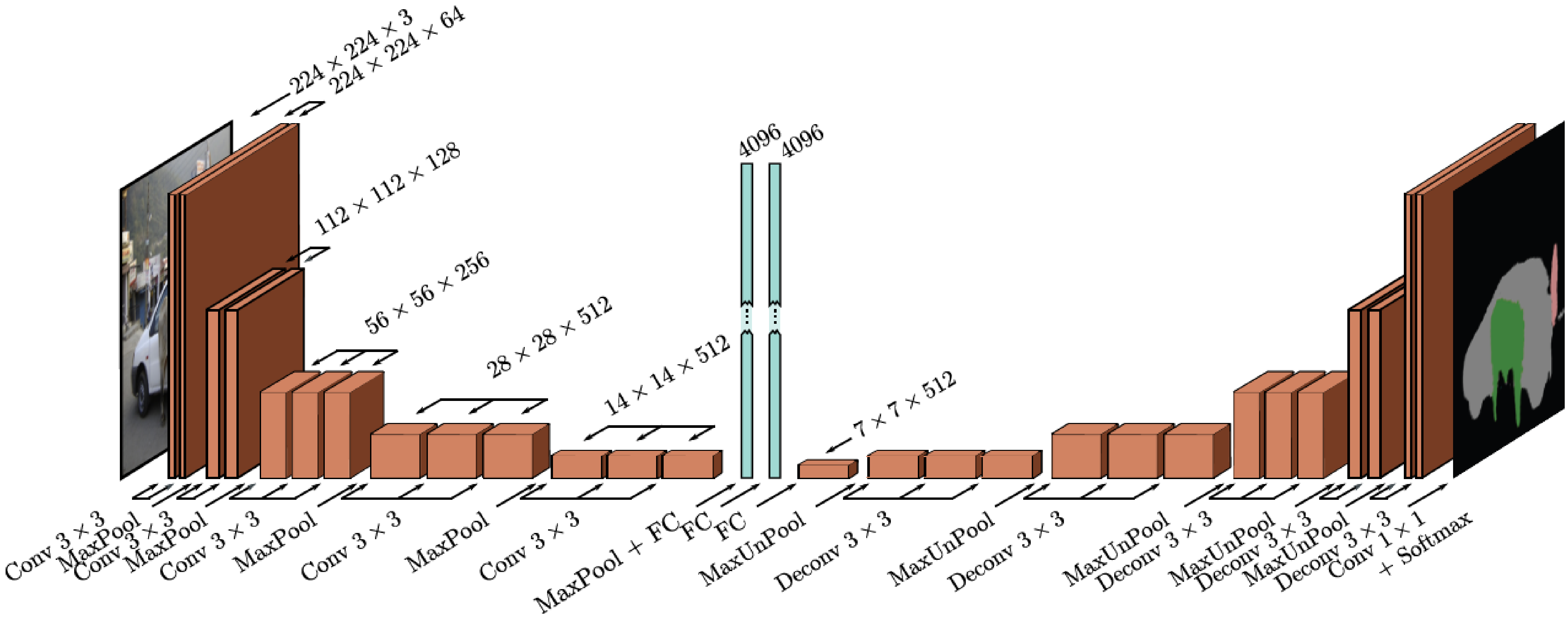
Any Questions?

???

Moving on

- Image classification
- Object detection
- Semantic segmentation

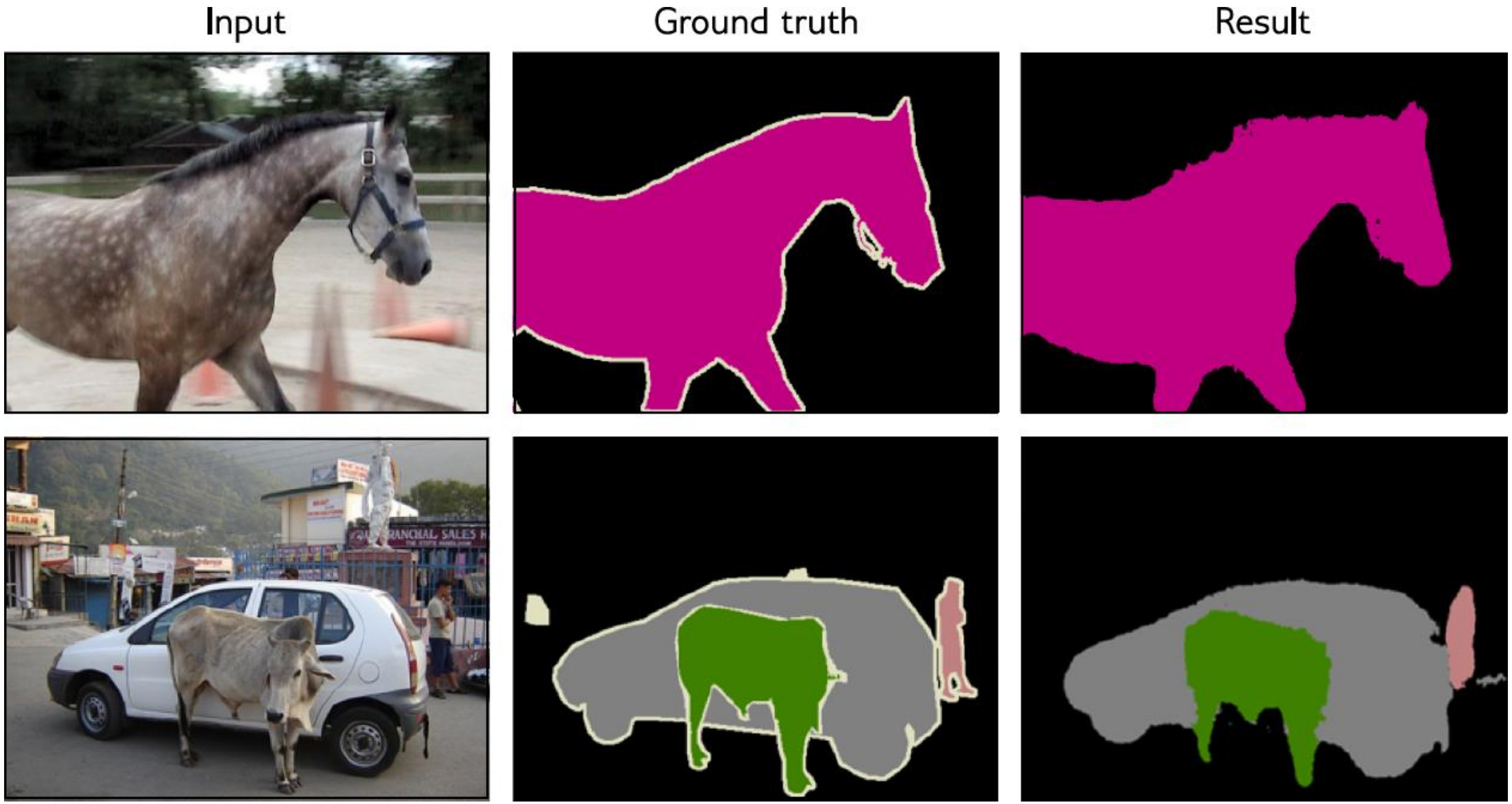
Semantic Segmentation (2015)



Encoder

Decoder

Semantic segmentation results

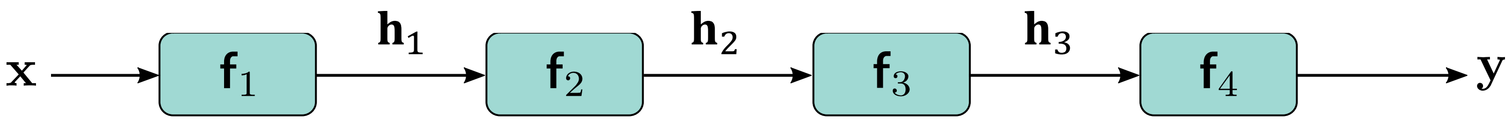


Any Questions?

- Finish up CNN examples (from last time)
- Residual networks
- Challenges of deep networks
- Residual connections and residual blocks
- Exploding gradients in residual networks
- Batch normalization
- Common residual architectures

Previously we saw a sequential network:

$$
\begin{aligned}
\mathbf{h}_1 &= \mathbf{f}_1[\mathbf{x}, \boldsymbol{\phi}_1] \\
\mathbf{h}_2 &= \mathbf{f}_2[\mathbf{h}_1, \boldsymbol{\phi}_2] \\
\mathbf{h}_3 &= \mathbf{f}_3[\mathbf{h}_2, \boldsymbol{\phi}_3] \\
\mathbf{y} &= \mathbf{f}_4[\mathbf{h}_3, \boldsymbol{\phi}_4]
\end{aligned}
$$



Fully connected network:

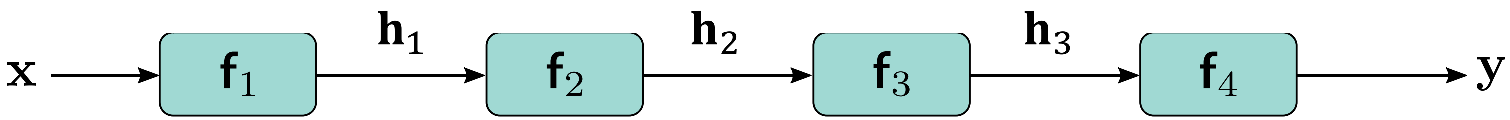
$$h_i = \mathrm{a}\left[\beta_i + \sum_{j=1}^{D} \omega_{ij} x_j\right]$$

Convolutional network (e.g. 1 channel → 1 channel):

$$
\begin{aligned}
h_i &= \mathrm{a}\left[\beta + \omega_1 x_{i-1} + \omega_2 x_i + \omega_3 x_{i+1}\right] \\
&= \mathrm{a}\left[\beta + \sum_{j=1}^{3} \omega_j x_{i+j-2}\right]
\end{aligned}
$$

Previously we saw a sequential network:

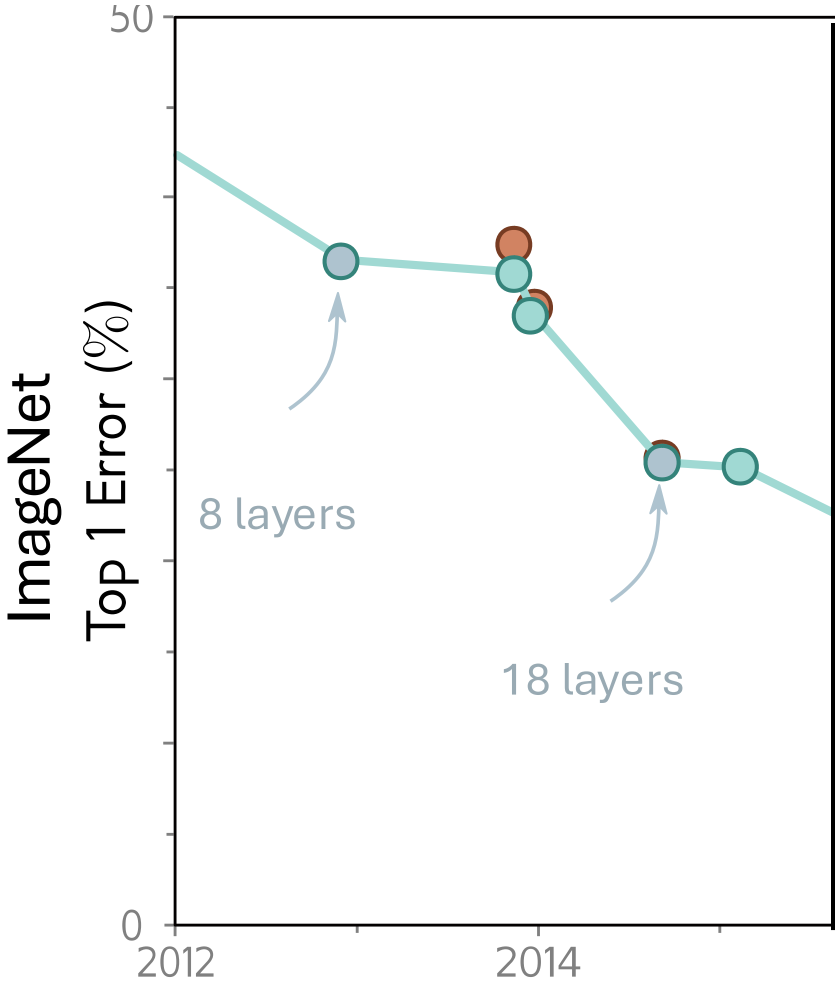
$$\begin{aligned}\mathbf{h}_1 &= \mathbf{f}_1[\mathbf{x}, \phi_1]\\ \mathbf{h}_2 &= \mathbf{f}_2[\mathbf{h}_1, \phi_2]\\ \mathbf{h}_3 &= \mathbf{f}_3[\mathbf{h}_2, \phi_3]\\ \mathbf{y} &= \mathbf{f}_4[\mathbf{h}_3, \phi_4]\end{aligned}$$



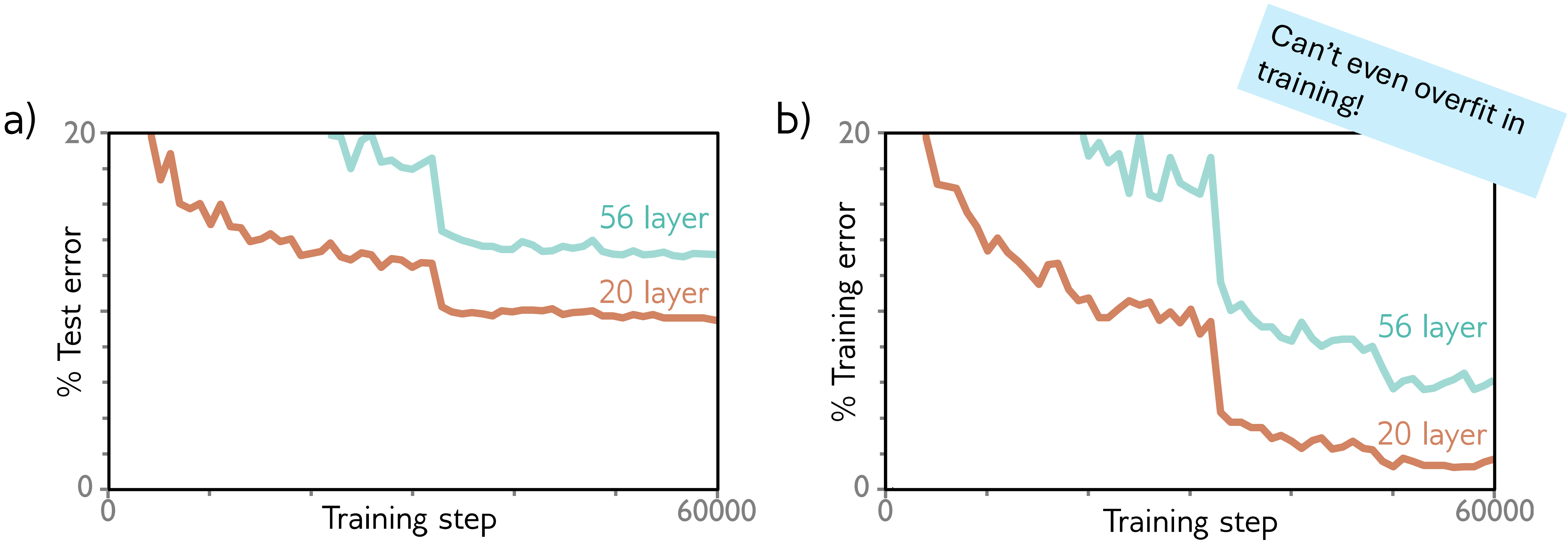
Can think of as a sequence of nested functions:

$$\mathbf{y} = \mathbf{f}_4\left[\mathbf{f}_3\left[\mathbf{f}_2\left[\mathbf{f}_1[\mathbf{x}, \phi_1], \phi_2\right], \phi_3\right], \phi_4\right]$$

More layers are better...



More layers are better… up to a point



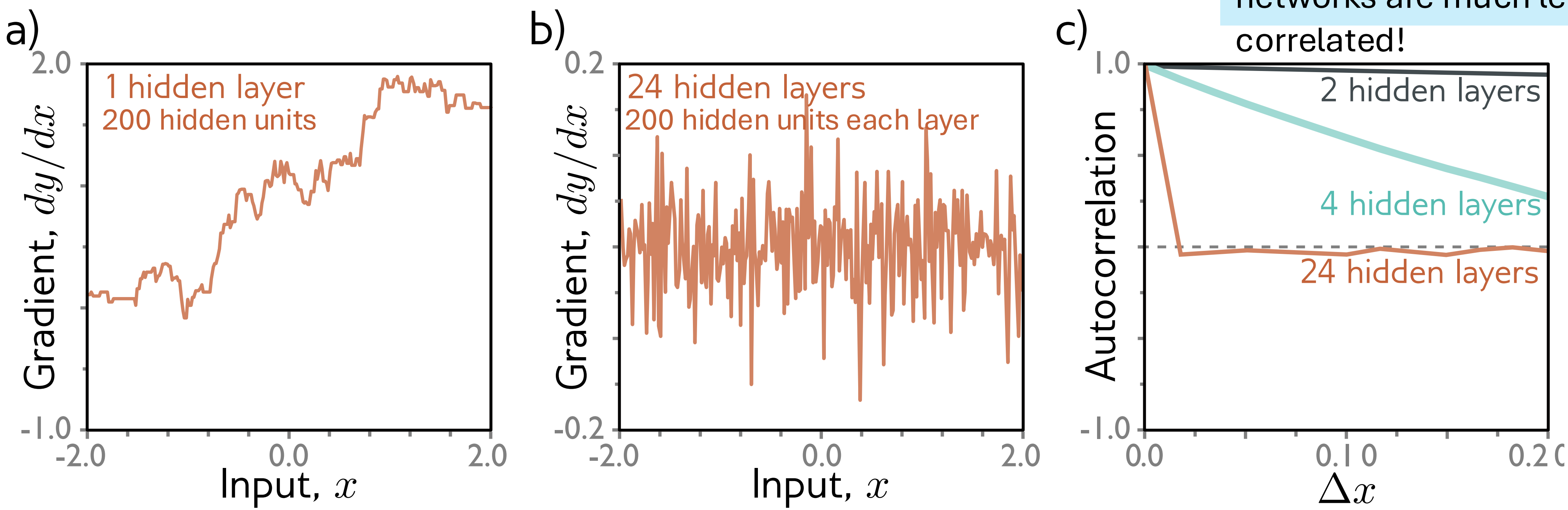
Convolutional Network on CIFAR10

What's going on?

Not completely understood, but...

Take a look at $\partial y/\partial x$ for shallow and deep networks.

Gradients of deeper networks are much less correlated!



Autocorrelation

A small step in gradient descent may jump to wildly different valued gradient!

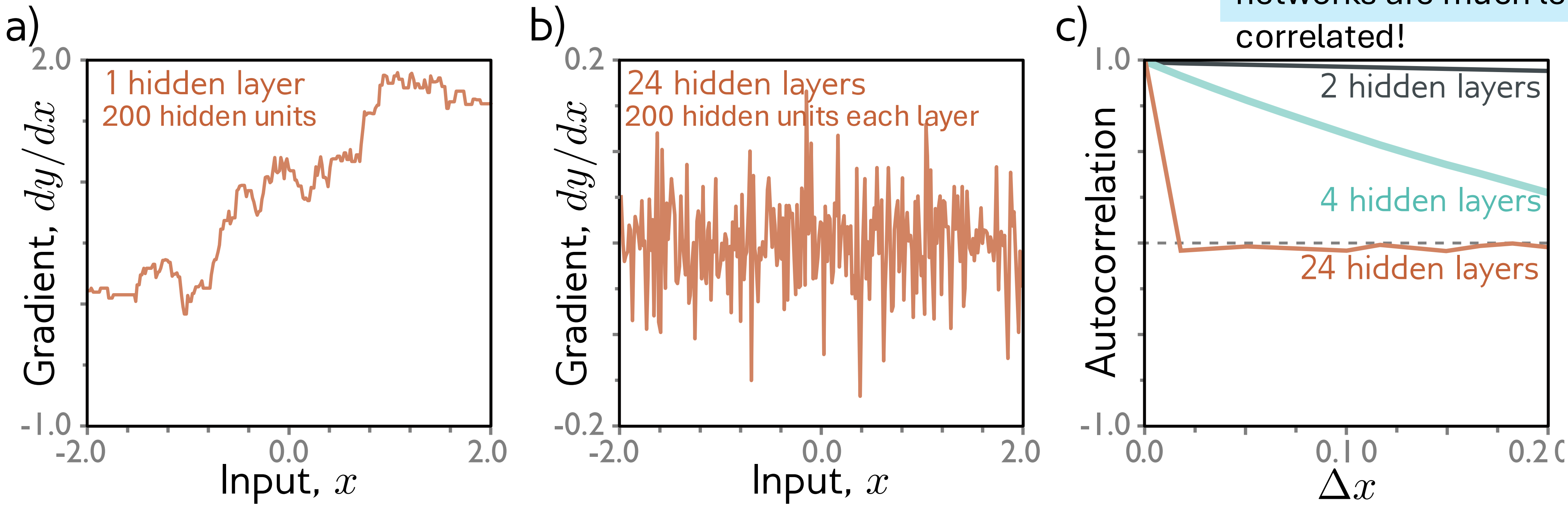
What's going on?

The Shattered Gradient Phenomenon

Not completely understood, but...

Take a look at $\partial y/\partial x$ for shallow and deep networks.

Gradients of deeper networks are much less correlated!



Autocorrelation

A small step in gradient descent may jump to wildly different valued gradient!

What's going on?

The Shattered Gradient Phenomenon

$$\mathbf{x} \longrightarrow \boxed{\mathbf{f}_1} \longrightarrow \boxed{\mathbf{f}_2} \longrightarrow \boxed{\mathbf{f}_3} \longrightarrow \boxed{\mathbf{f}_4} \longrightarrow \mathbf{y}$$

$$\mathbf{y} = \mathbf{f}_4\left[\mathbf{f}_3\left[\mathbf{f}_2\left[\mathbf{f}_1[\mathbf{x}, \phi_1], \phi_2\right], \phi_3\right], \phi_4\right]$$

The derivative of the output y w.r.t. the first layer f_1 is, by the chain rule:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{f}_1} = \frac{\partial \mathbf{f}_4}{\partial \mathbf{f}_3}\frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_2}\frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1}$$

f_1 impacts f_2 impacts f_3, etc...

Any Questions?

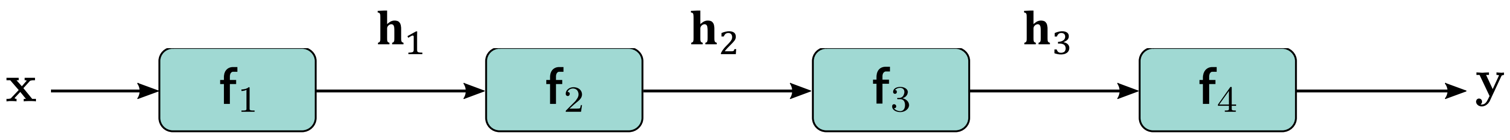
- Finish up CNN examples (from last time)
- Residual networks

- Challenges of deep networks
- Residual connections and residual blocks
- Exploding gradients in residual networks
- Batch normalization
- Common residual architectures

Solution: Residual connections

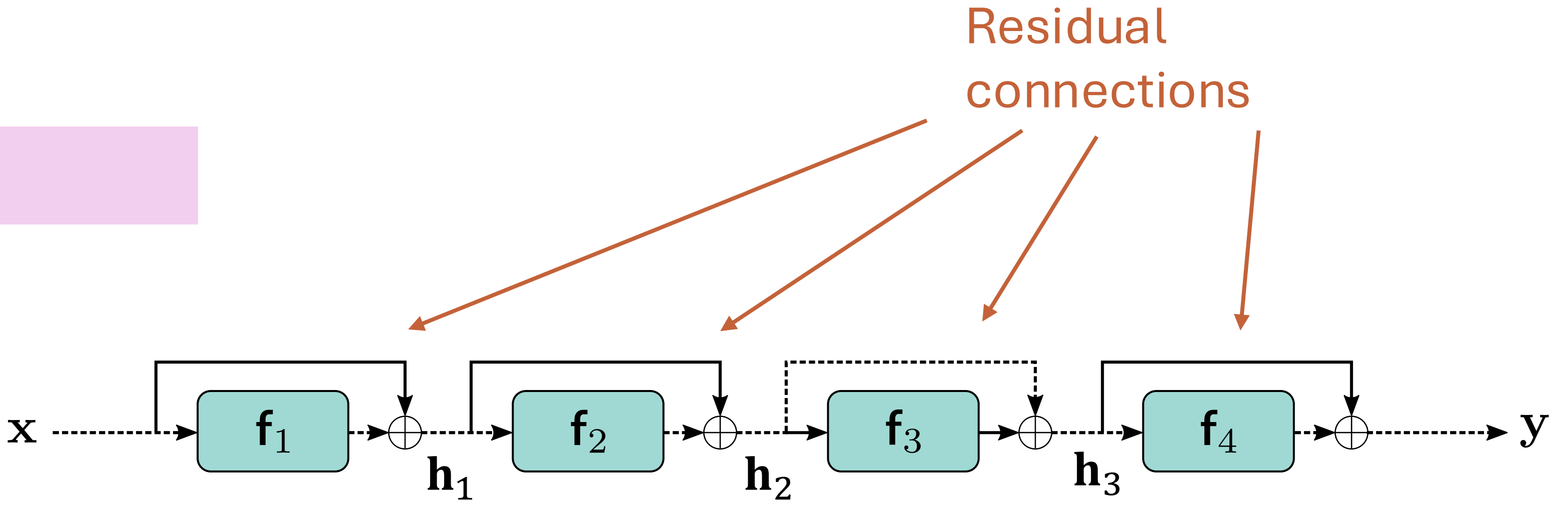
Regular sequential network:

$$
\begin{aligned}
\mathbf{h}_1 &= \mathbf{f}_1[\mathbf{x}, \phi_1] \\
\mathbf{h}_2 &= \mathbf{f}_2[\mathbf{h}_1, \phi_2] \\
\mathbf{h}_3 &= \mathbf{f}_3[\mathbf{h}_2, \phi_3] \\
\mathbf{y} &= \mathbf{f}_4[\mathbf{h}_3, \phi_4]
\end{aligned}
$$



Residual network:

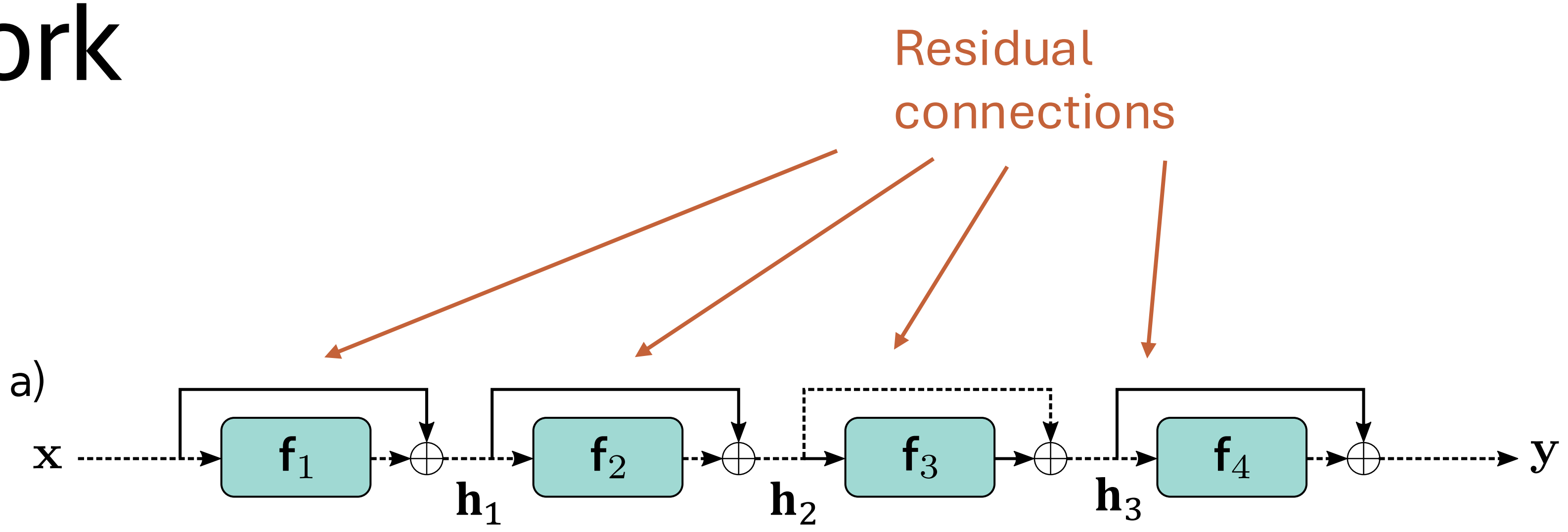
$$
\begin{aligned}
\mathbf{h}_1 &= \mathbf{x} + \mathbf{f}_1[\mathbf{x}, \phi_1] \\
\mathbf{h}_2 &= \mathbf{h}_1 + \mathbf{f}_2[\mathbf{h}_1, \phi_2] \\
\mathbf{h}_3 &= \mathbf{h}_2 + \mathbf{f}_3[\mathbf{h}_2, \phi_3] \\
\mathbf{y} &= \mathbf{h}_3 + \mathbf{f}_4[\mathbf{h}_3, \phi_4]
\end{aligned}
$$



K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” *arXiv:1512.03385 [cs]*, Dec. 2015, http://arxiv.org/abs/1512.03385

Residual Network

$$\mathbf{h}_1 = \mathbf{x} + \mathbf{f}_1[\mathbf{x}, \phi_1]$$
$$\mathbf{h}_2 = \mathbf{h}_1 + \mathbf{f}_2[\mathbf{h}_1, \phi_2]$$
$$\mathbf{h}_3 = \mathbf{h}_2 + \mathbf{f}_3[\mathbf{h}_2, \phi_3]$$
$$\mathbf{y} = \mathbf{h}_3 + \mathbf{f}_4[\mathbf{h}_3, \phi_4]$$

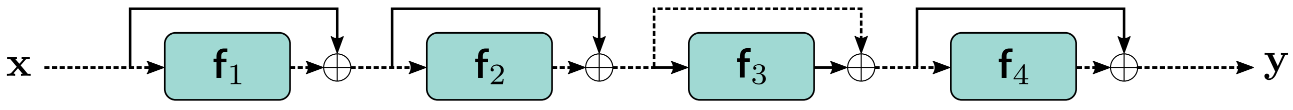


Think of optimizing f_1, then f_2, ...

Substituting in:

$$\begin{aligned}\mathbf{y} = \mathbf{x} &+ \mathbf{f}_1[\mathbf{x}] \\ &+ \mathbf{f}_2\big[\mathbf{x} + \mathbf{f}_1[\mathbf{x}]\big] \\ &+ \mathbf{f}_3\Big[\mathbf{x} + \mathbf{f}_1[\mathbf{x}] + \mathbf{f}_2\big[\mathbf{x} + \mathbf{f}_1[\mathbf{x}]\big]\Big] \\ &+ \mathbf{f}_4\Big[\mathbf{x} + \mathbf{f}_1[\mathbf{x}] + \mathbf{f}_2\big[\mathbf{x} + \mathbf{f}_1[\mathbf{x}]\big] + \mathbf{f}_3\big[\mathbf{x} + \mathbf{f}_1[\mathbf{x}] + \mathbf{f}_2[\mathbf{x} + \mathbf{f}_1[\mathbf{x}]]\big]\Big]\end{aligned}$$

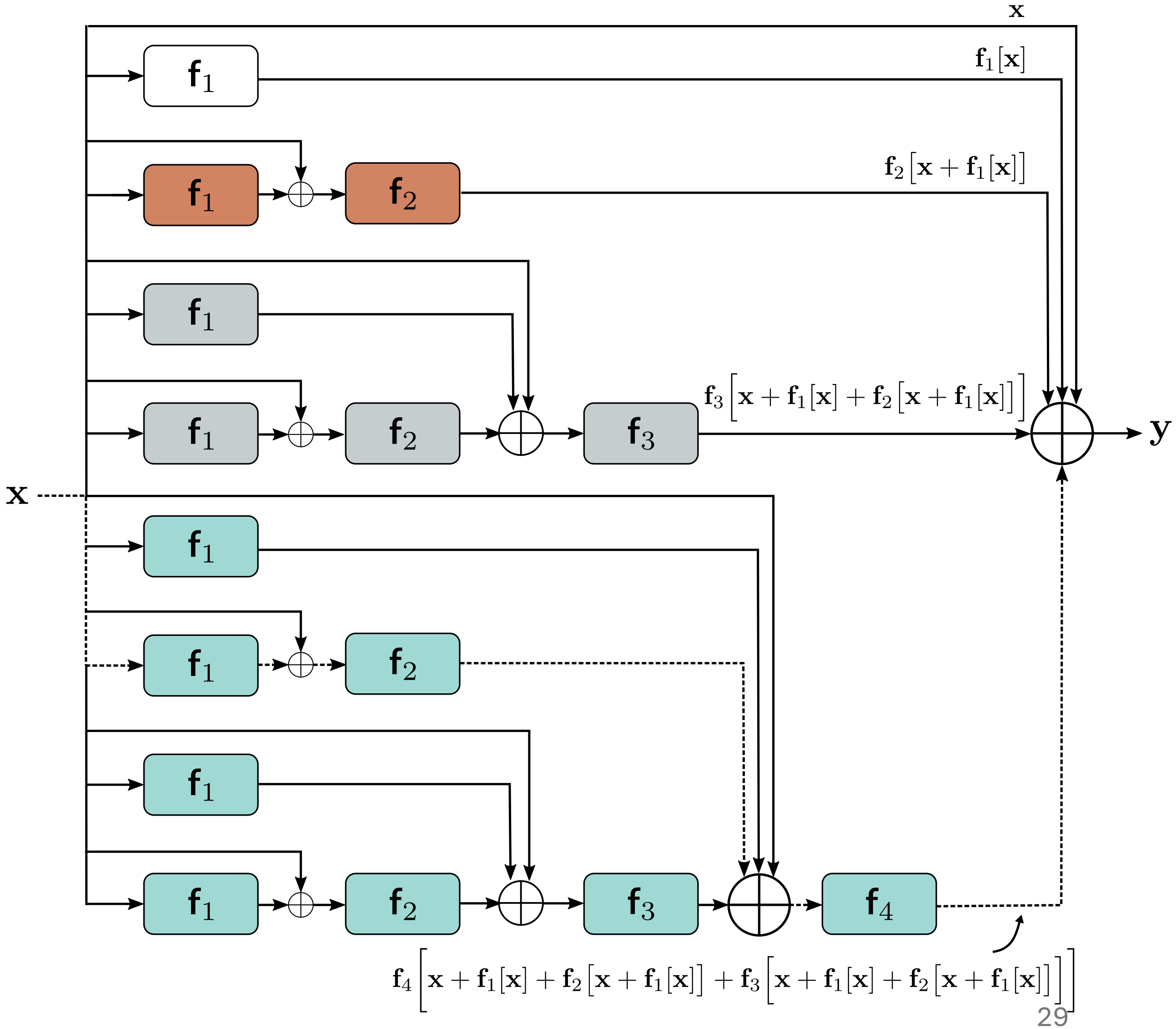
Residual Network



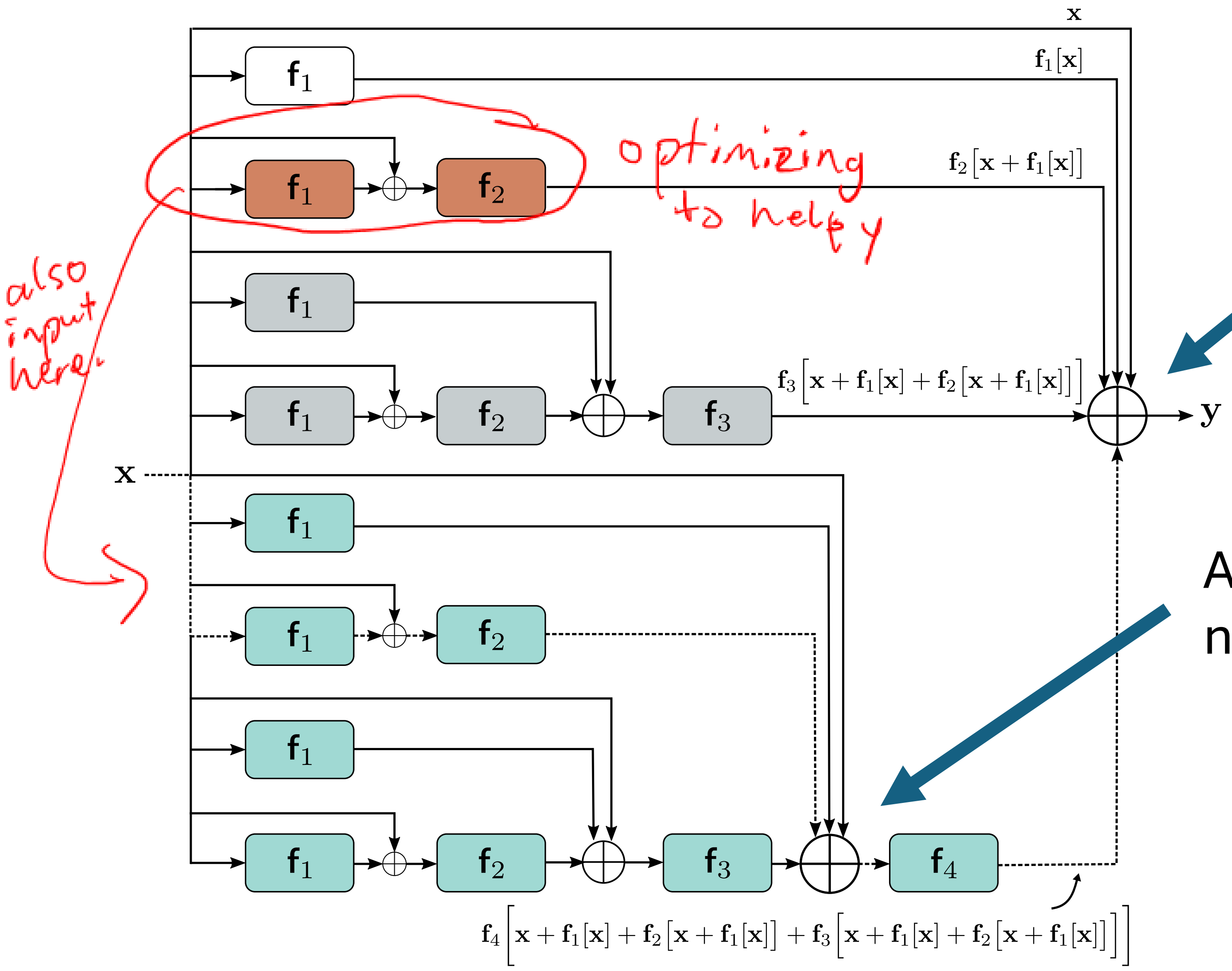
We can unravel all the possible paths

The output is the sum of the input plus 4 partial networks.

$$
\begin{aligned}
\mathbf{y} = \mathbf{x} \; + \; & \mathbf{f}_1[\mathbf{x}] \\
+ \; & \mathbf{f}_2\big[\mathbf{x} + \mathbf{f}_1[\mathbf{x}]\big] \\
+ \; & \mathbf{f}_3\Big[\mathbf{x} + \mathbf{f}_1[\mathbf{x}] + \mathbf{f}_2\big[\mathbf{x} + \mathbf{f}_1[\mathbf{x}]\big]\Big] \\
+ \; & \mathbf{f}_4\bigg[\mathbf{x} + \mathbf{f}_1[\mathbf{x}] + \mathbf{f}_2\big[\mathbf{x} + \mathbf{f}_1[\mathbf{x}]\big] + \mathbf{f}_3\Big[\mathbf{x} + \mathbf{f}_1[\mathbf{x}] + \mathbf{f}_2\big[\mathbf{x} + \mathbf{f}_1[\mathbf{x}]\big]\Big]\bigg]
\end{aligned}
$$



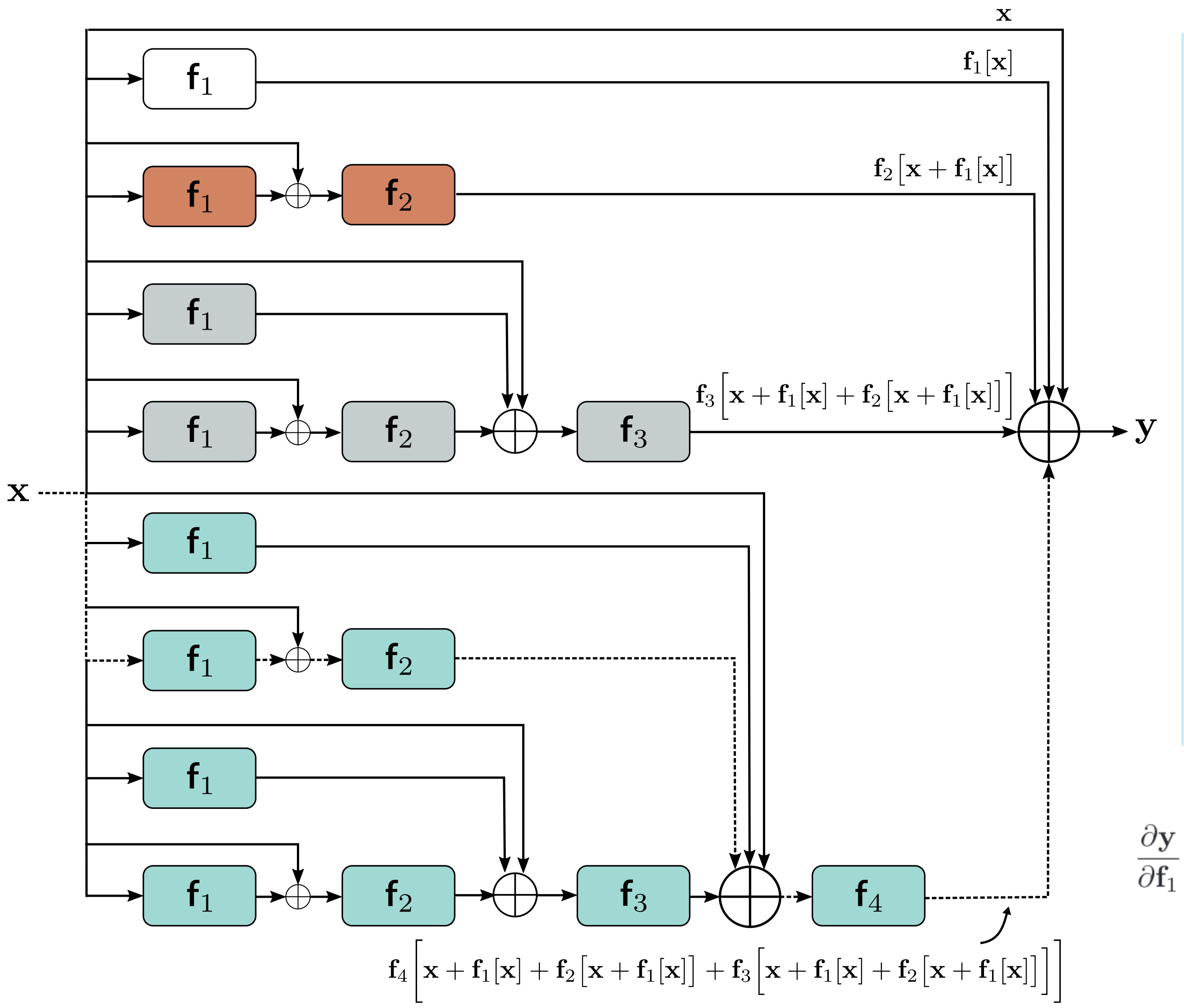
Residual Network as Ensemble of Networks



Ensemble of four networks

Another ensemble of four networks

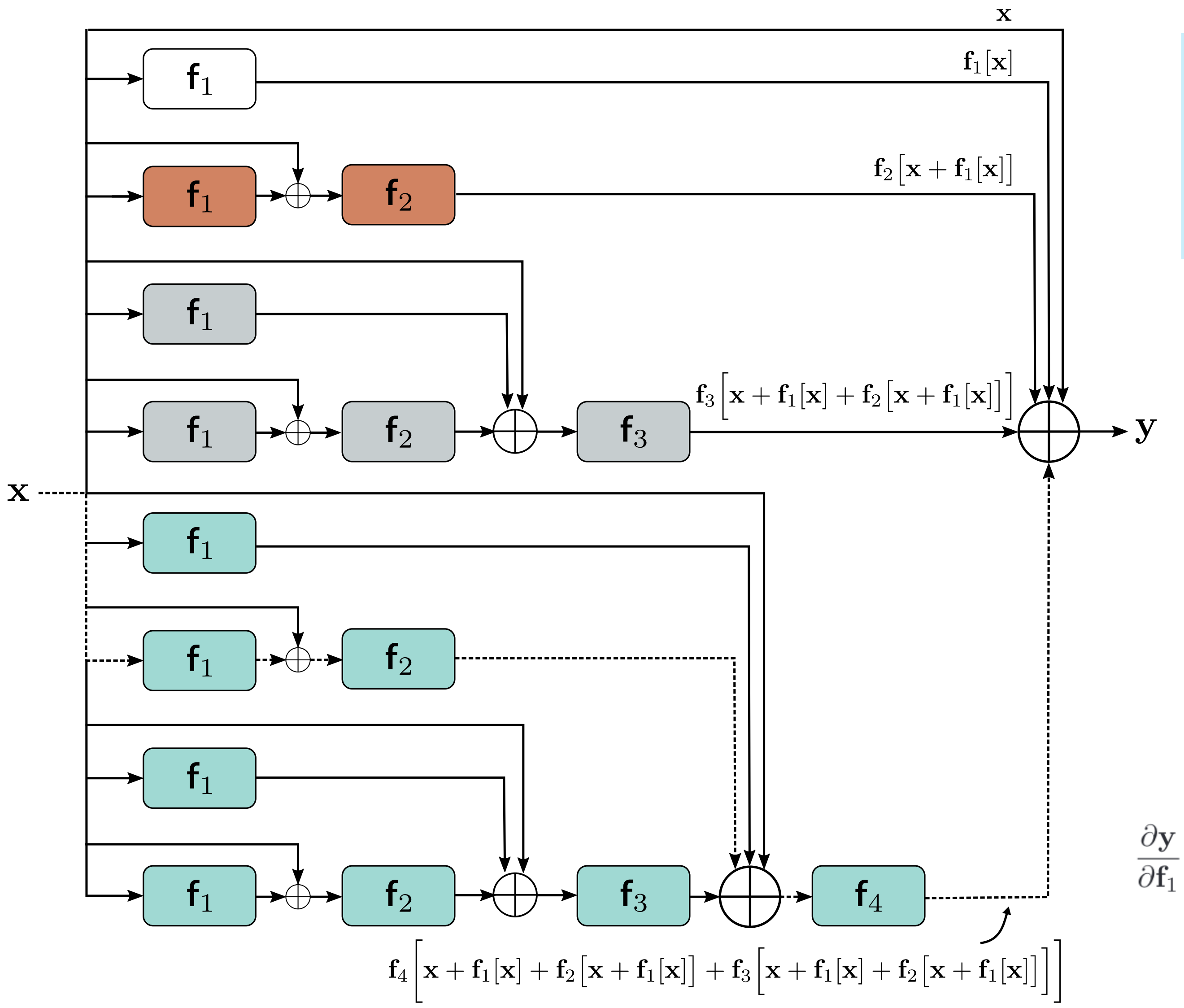
Residual Network as Ensemble of Networks



- 16 possible paths through the network!
- 8 paths include f_1
- The influence of f_1 on $\partial y/\partial f_1$ takes 8 different forms
- Gradients on shorter paths generally better behaved.

$$\frac{\partial \mathbf{y}}{\partial \mathbf{f}_1} = \mathbf{I} + \frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1} + \left(\frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_1} + \frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_2}\frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1}\right) + \left(\frac{\partial \mathbf{f}_4}{\partial \mathbf{f}_1} + \frac{\partial \mathbf{f}_4}{\partial \mathbf{f}_2}\frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1} + \frac{\partial \mathbf{f}_4}{\partial \mathbf{f}_3}\frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_1} + \frac{\partial \mathbf{f}_4}{\partial \mathbf{f}_3}\frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_2}\frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1}\right)$$

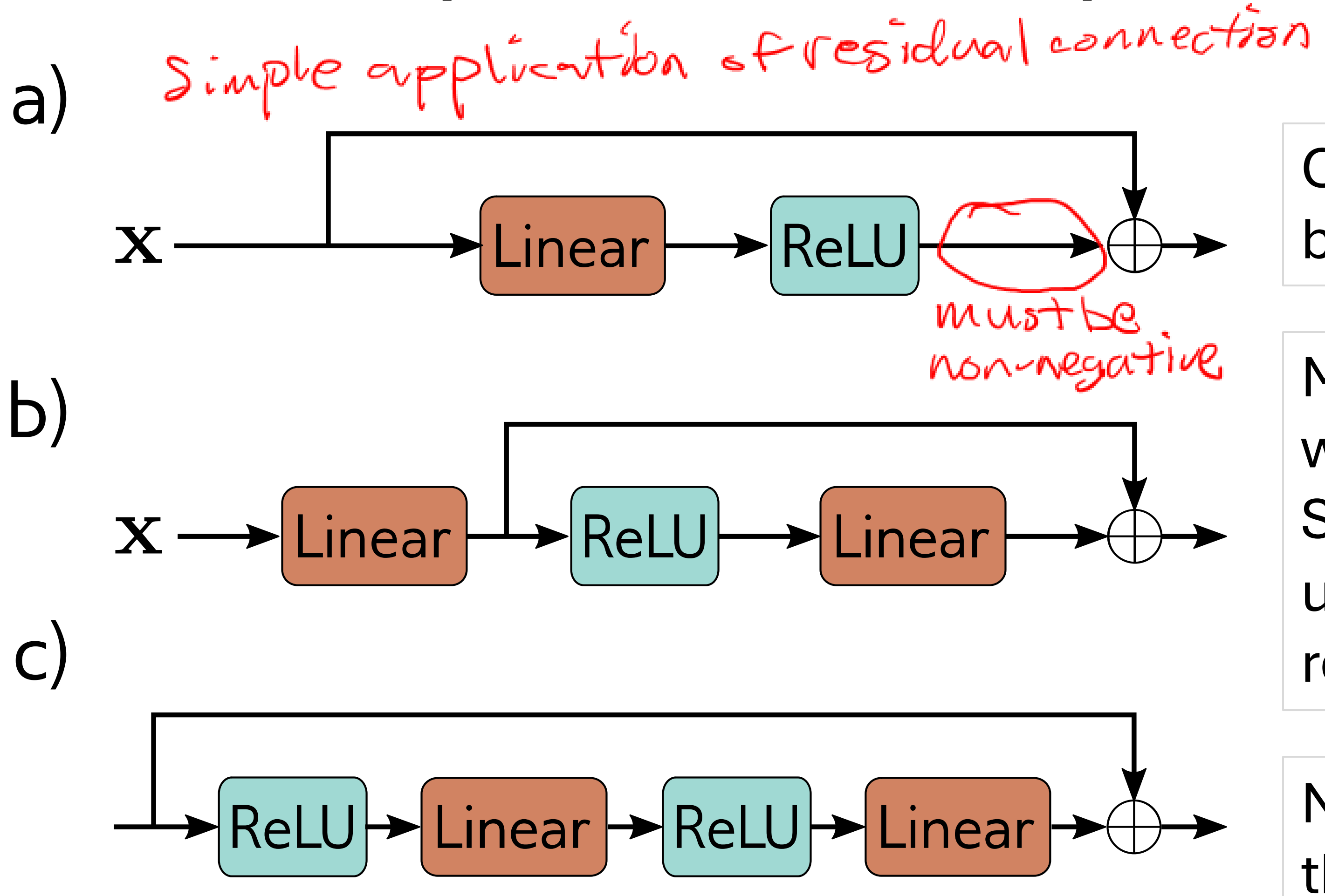
Residual Network as Ensemble of Networks



During training, the model can amplify or attenuate the different paths to achieve the best results

$$\frac{\partial \mathbf{y}}{\partial \mathbf{f}_1} = \mathbf{I} + \frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1} + \left(\frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_1} + \frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_2}\frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1}\right) + \left(\frac{\partial \mathbf{f}_4}{\partial \mathbf{f}_1} + \frac{\partial \mathbf{f}_4}{\partial \mathbf{f}_2}\frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1} + \frac{\partial \mathbf{f}_4}{\partial \mathbf{f}_3}\frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_1} + \frac{\partial \mathbf{f}_4}{\partial \mathbf{f}_3}\frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_2}\frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1}\right)$$

Order of operations is important



can only increase from x.

Can only add to the residual because of the ReLU

More flexible approach to end with linear block.
Starting with linear block gives us some flexibility on spatial resolution.

Note: if we start with a ReLU, then will clamp negative values and so do nothing

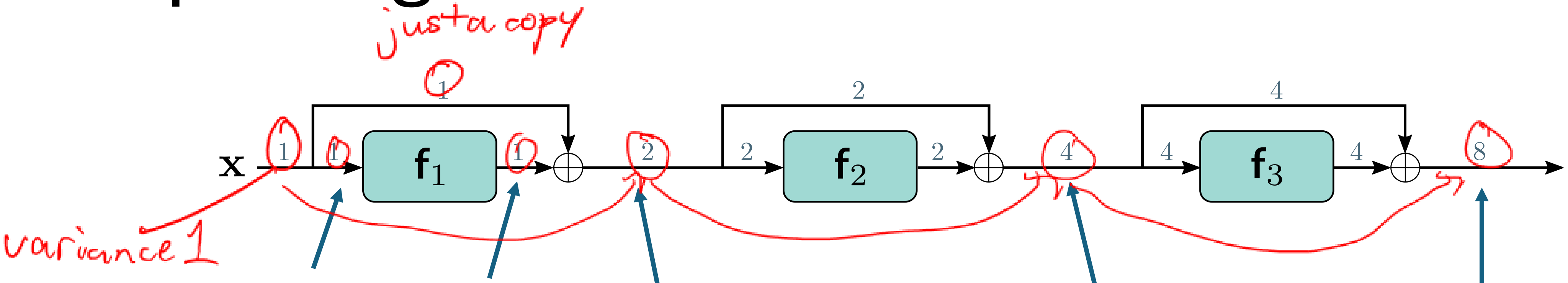
This helps increase depth up to a point…

Any Questions?

???

- Challenges of deep networks
- Residual connections and residual blocks
- Exploding gradients in residual networks
- Batch normalization
- Common residual architectures

Exploding Gradients in Residual Networks

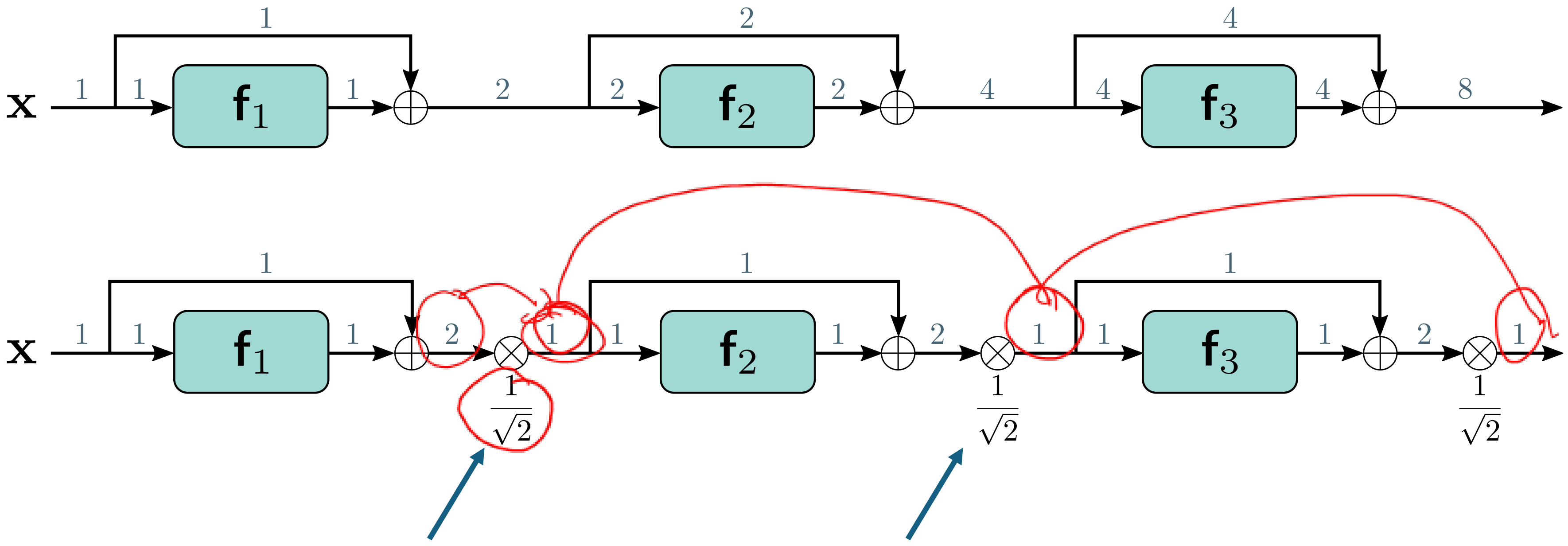


With He initialization we can control the variance inside the block

But variance doubles when we add the residual back in.

And then grows exponentially.

Exploding Gradients in Residual Networks



Could stabilize by renormalizing after
adding each residual.

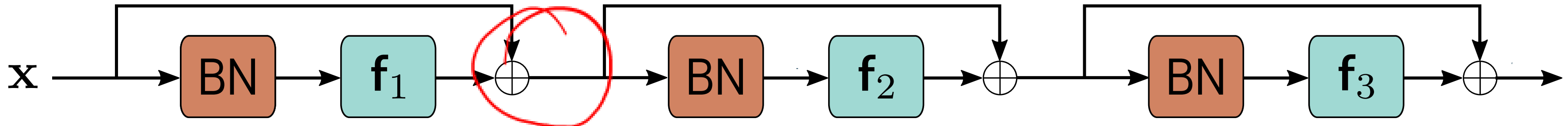
More common to apply *batch normalization*.

Plan for Today

- Finish up CNN examples (from last time)
- Residual networks

- Challenges of deep networks
- Residual connections and residual blocks
- Exploding gradients in residual networks
- Batch normalization
- Common residual architectures

Batch Normalization (a.k.a. *BatchNorm*)



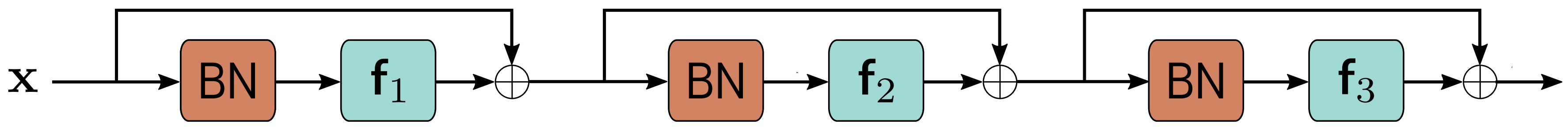
- Shifts and rescales each activation so that its mean and variance across the batch become values that are learned during training

fix the scaling w/o needing to assume correlation structure.

Collect stats from whole batch.
Use those to normalize batch to mean 0, std 1.

S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv:1502.03167 [cs]*, Mar. 2015, http://arxiv.org/abs/1502.03167

Batch Normalization (a.k.a. *BatchNorm*)

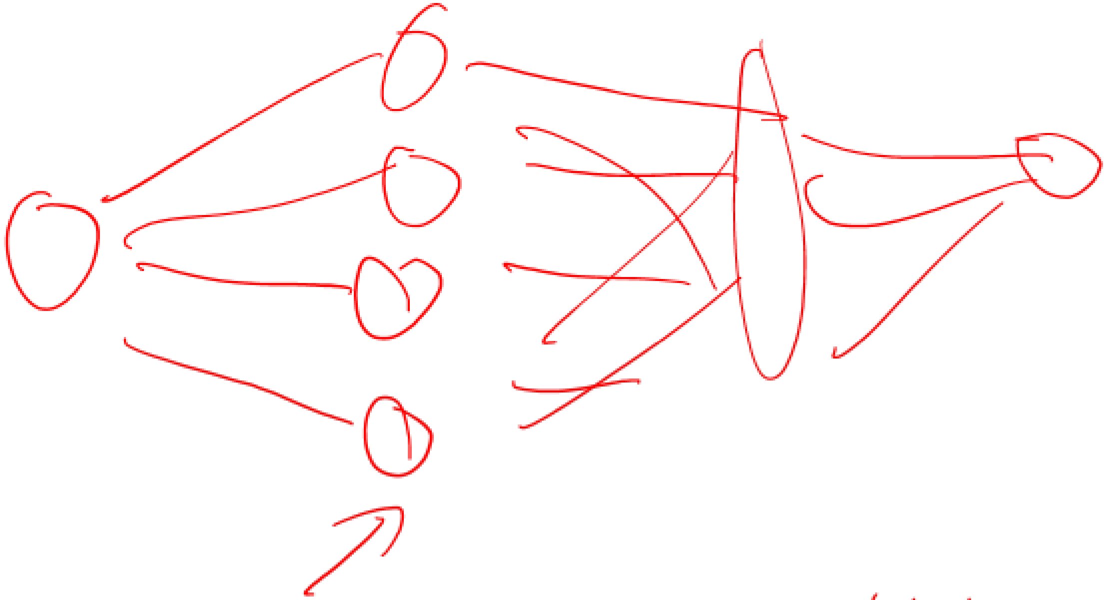


- Shifts and rescales each activation so that its mean and variance across the batch become values that are learned during training

Calculate the sample *mean* and *standard deviation* for each hidden unit across samples of the batch.

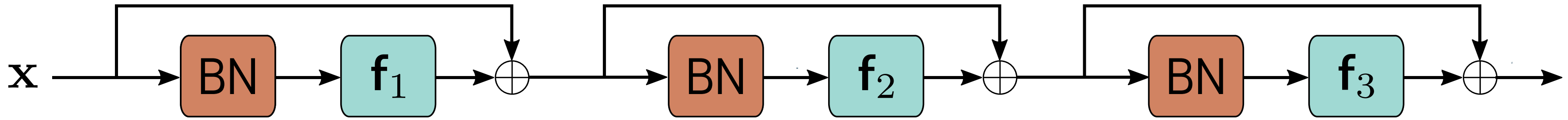
$$m_h = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} h_i$$

$$s_h = \sqrt{\frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}(h_i - m_h)^2},$$



Each has its own BatchNorm

Batch Normalization (a.k.a. *BatchNorm*)



- Shifts and rescales each activation so that its mean and variance across the batch become values that are learned during training

Calculate the sample *mean* and *standard deviation* for each hidden unit across samples of the batch.

$$m_h = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} h_i$$

$$s_h = \sqrt{\frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}(h_i - m_h)^2},$$

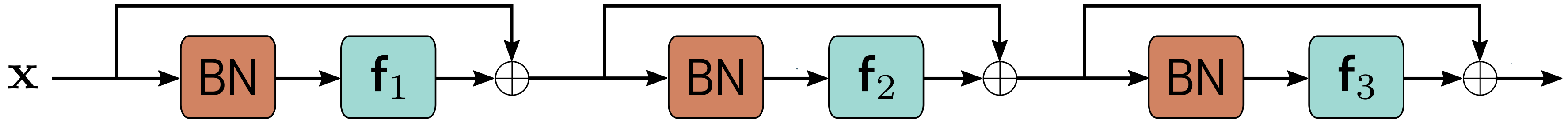
Standardize (*normalize*) to zero-mean and unit standard deviation.

$$\hat{h}_i \leftarrow \frac{h_i - m_h}{s_h + \epsilon} \qquad \forall i \in \mathcal{B},$$

Scale by γ and shift by δ, which are *learned* parameters.

$$h_i \leftarrow \gamma \hat{h}_i + \delta \qquad \forall i \in \mathcal{B}.$$

Batch Normalization (a.k.a. *BatchNorm*)



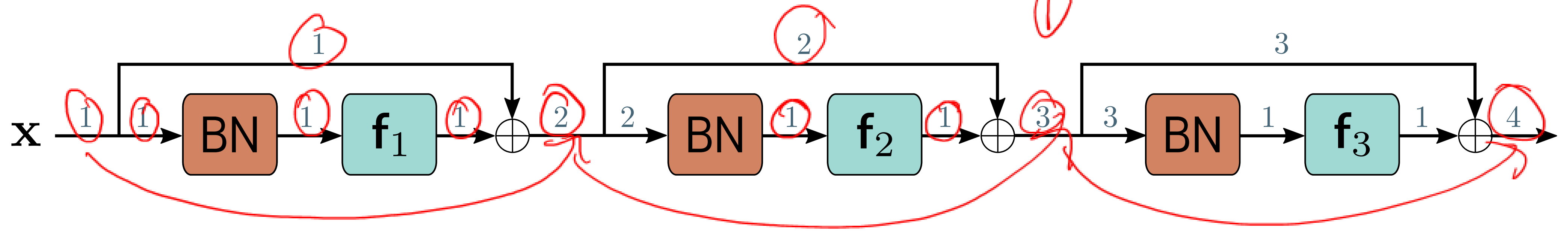
- Applied independently to each hidden unit

- Standard FC Network with K layers, each with D hidden units:
  KD learned scales, γ , and KD learned offset, δ

- Convolutional Network with K layers, each with C channels:
  KC learned scales, γ , and KC learned offset, δ

Benefits of BatchNorm

BN here sometimes used to force 1 variance



Variance only increases linearly

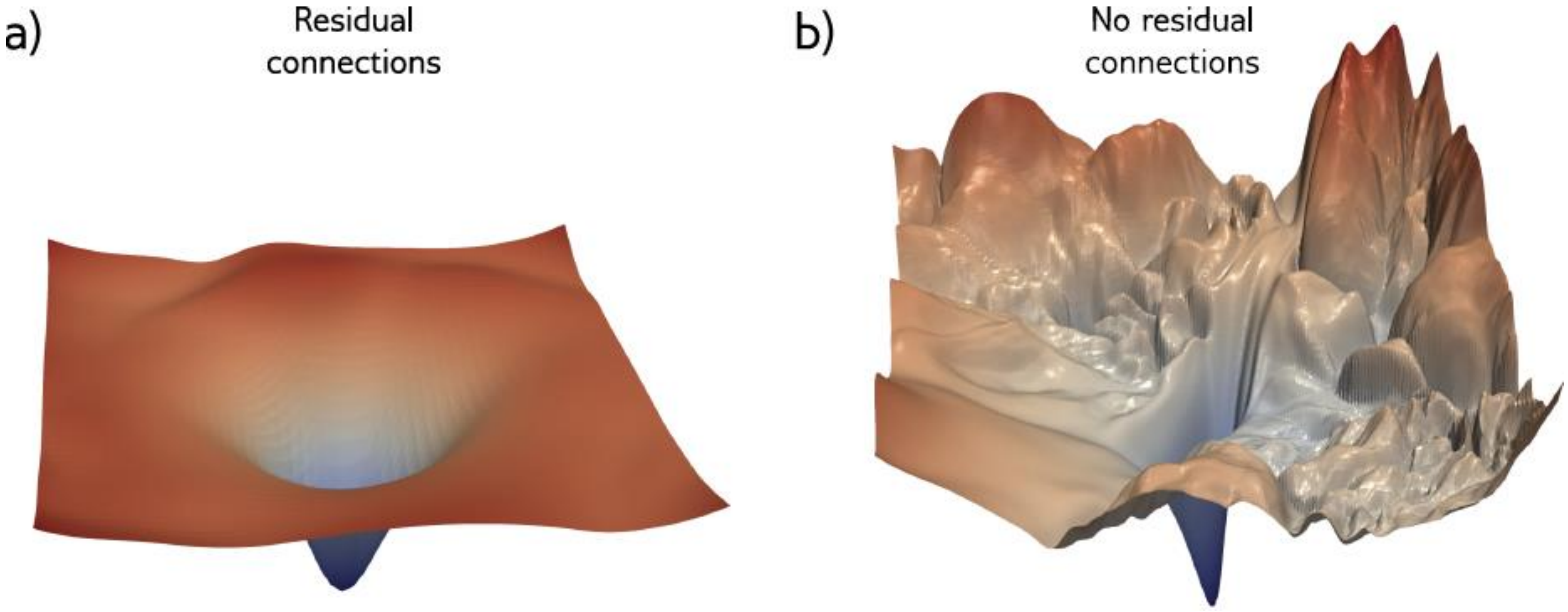
Stable forward propagation

- Initialize offsets δ to zero and scales γ to 1
- Variance now increases linearly
- k^{th} block adds *one unit of variance* to variance of k
- At initialization, later layers make smaller relative change to overall variation
- During training, the scales can increase in later layers if helpful
  →control the effective depth

Benefits of BatchNorm

Supports higher learning rates

Makes the loss surface smoother (reduces shattered gradients)



H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, “Visualizing the Loss Landscape of Neural Nets,” arXiv.org, https://arxiv.org/abs/1712.09913v3

Benefits of BatchNorm

Regularization via added noise

BatchNorm injects noise since BN scale and shift depend on batch statistics.

Disadvantages of Batch Normalization

- Batch normalization makes results dependent on what else is in the same batch.
    - Much more likely if you group on target value.
- Layer normalization instead?
    - Similar spirit.
    - Normalize hidden layer activations of same input.
- Both actively used nowadays.

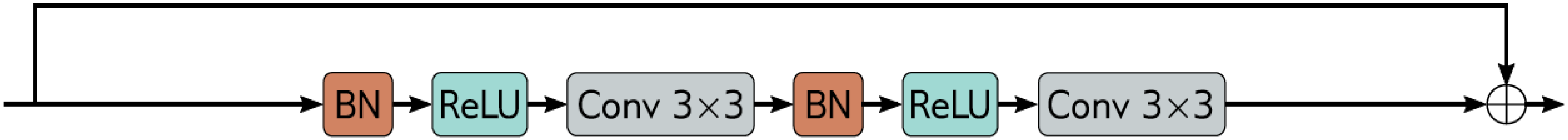
Any questions?

???

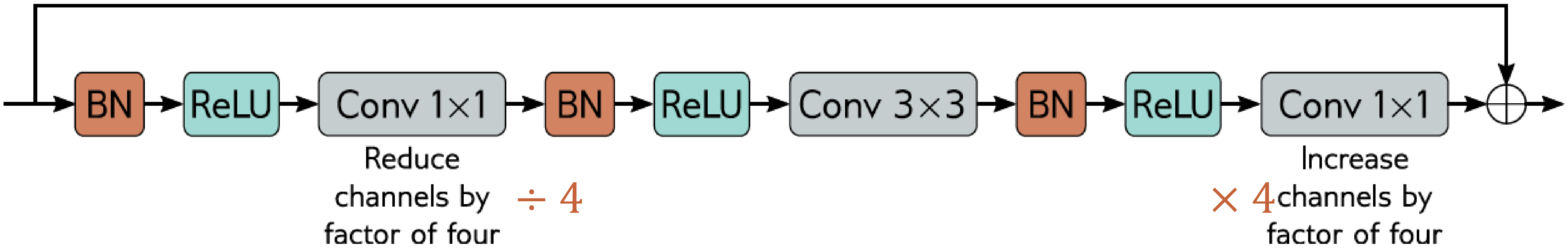
- Challenges of deep networks
- Residual connections and residual blocks
- Exploding gradients in residual networks
- Batch normalization
- Common residual architectures

ResNet (2015)

ResNet Block

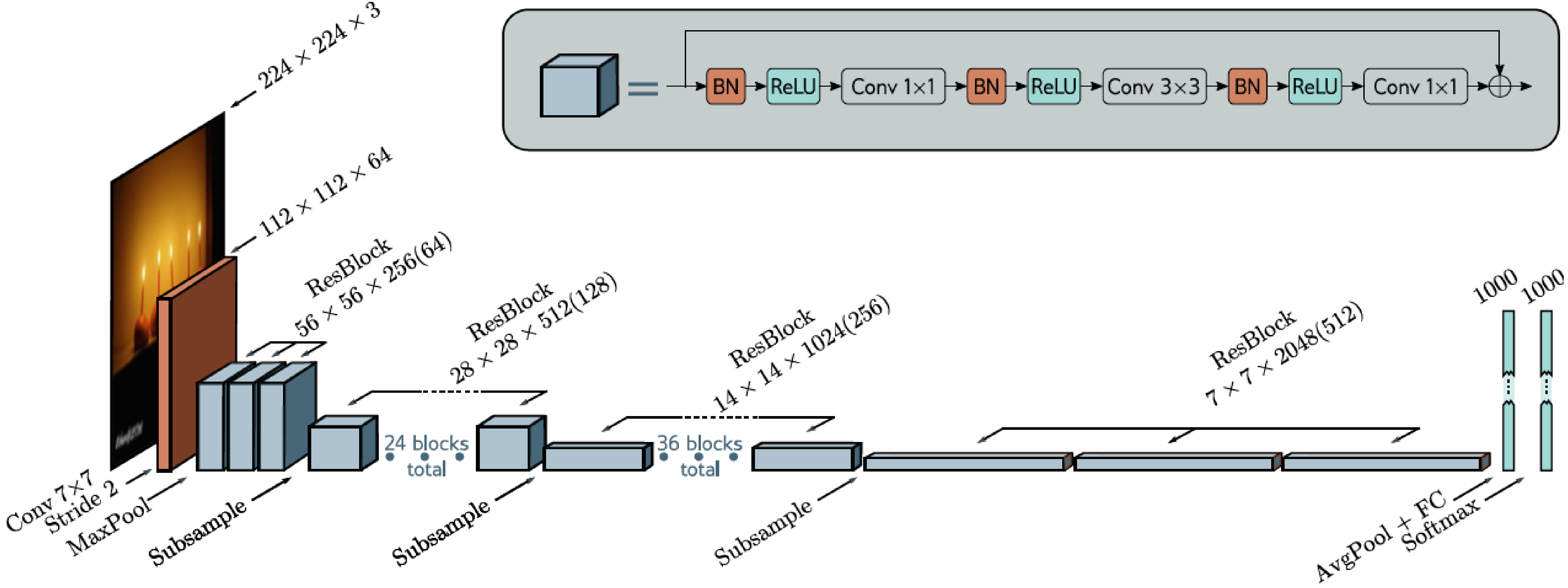


Bottleneck Residual



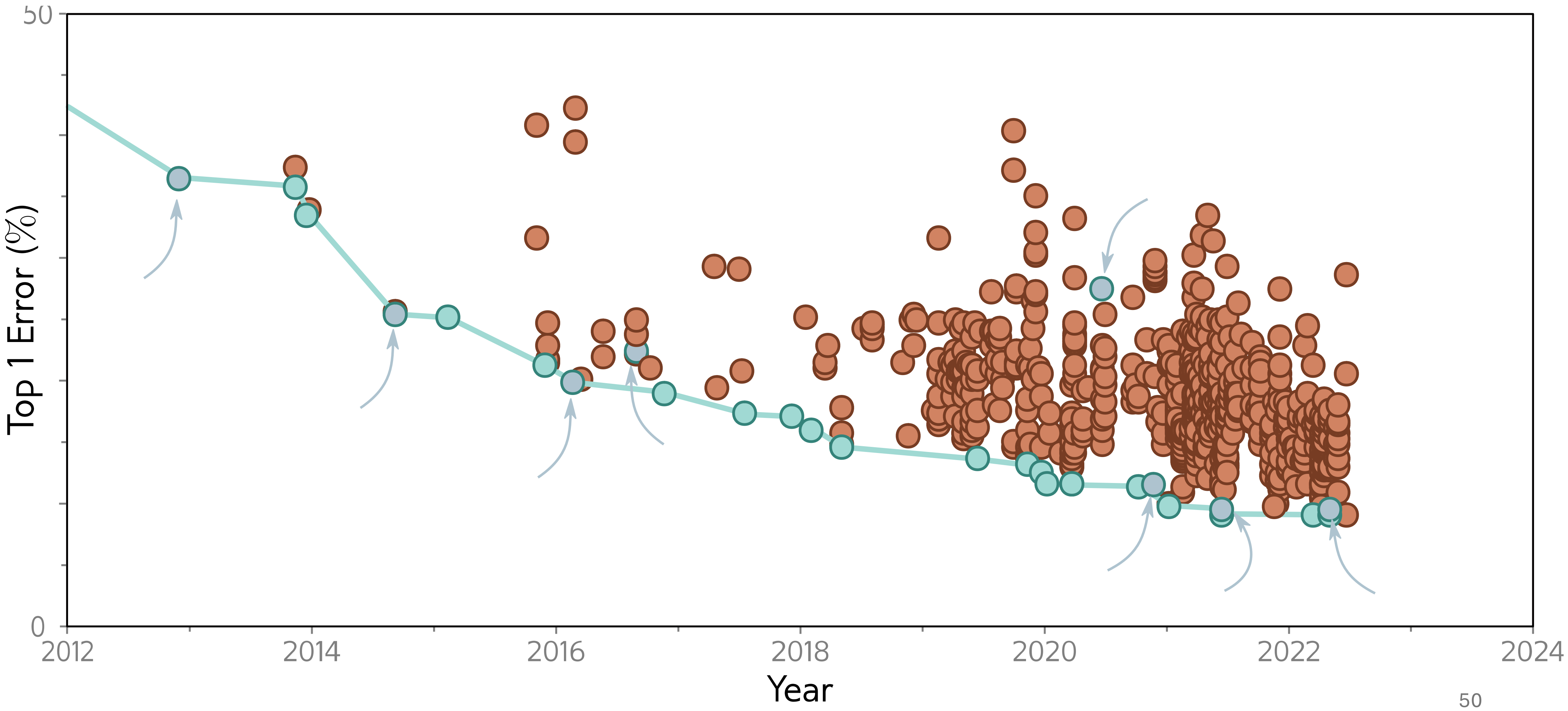
K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385 [cs]*, Dec. 2015, http://arxiv.org/abs/1512.03385

Resnet 200 (2016) for ImageNet Classification



K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385 [cs]*, Dec. 2015, http://arxiv.org/abs/1512.03385

ImageNet History



DenseNet

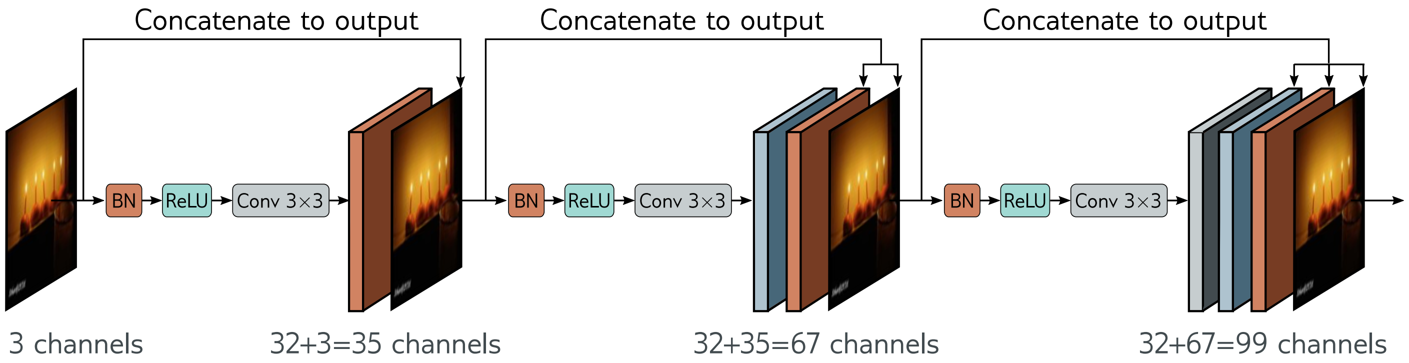


Figure from UDL

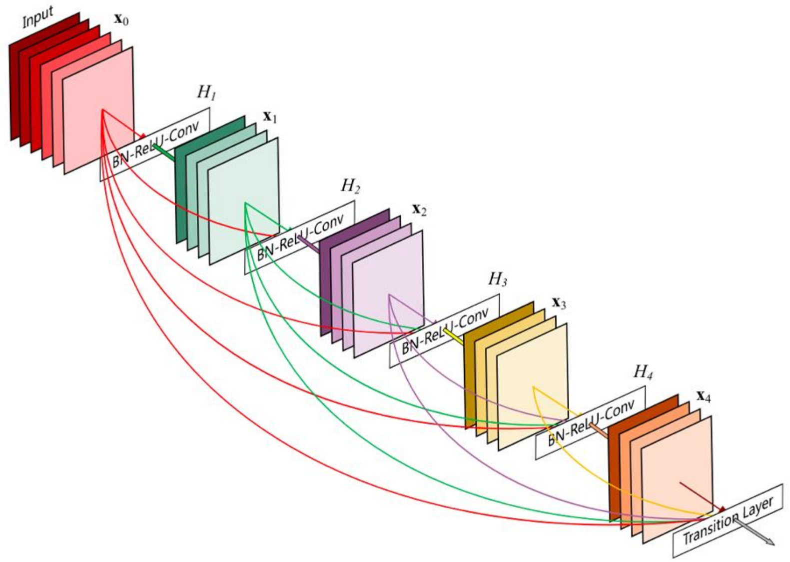
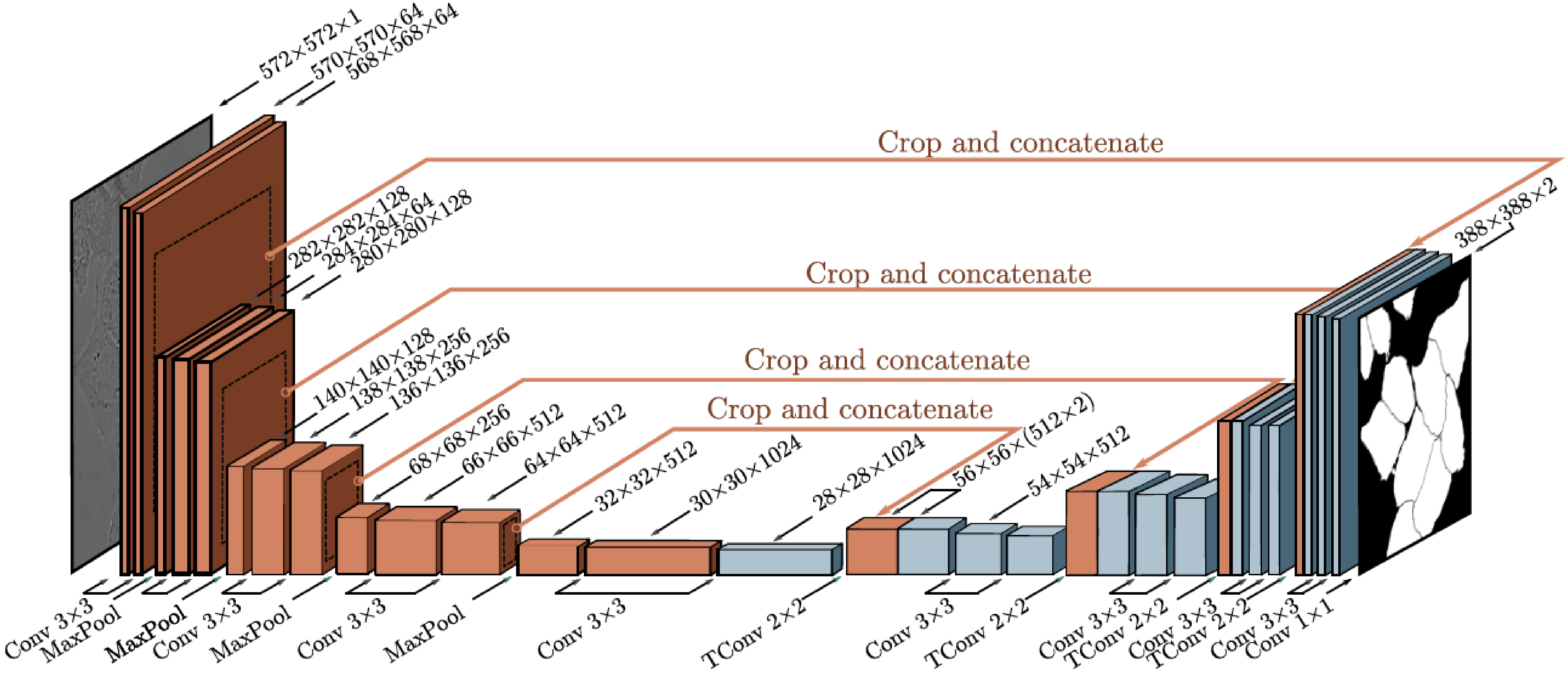


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

Figure from paper

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017b). Densely connected convolutional networks. IEEE/CVF Computer Vision & Pattern Recognition, 4700–4708.

U-Net (2016)



Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. International Conference on Medical Image Computing and ComputerAssisted Intervention, 234–241.

U-Net Results

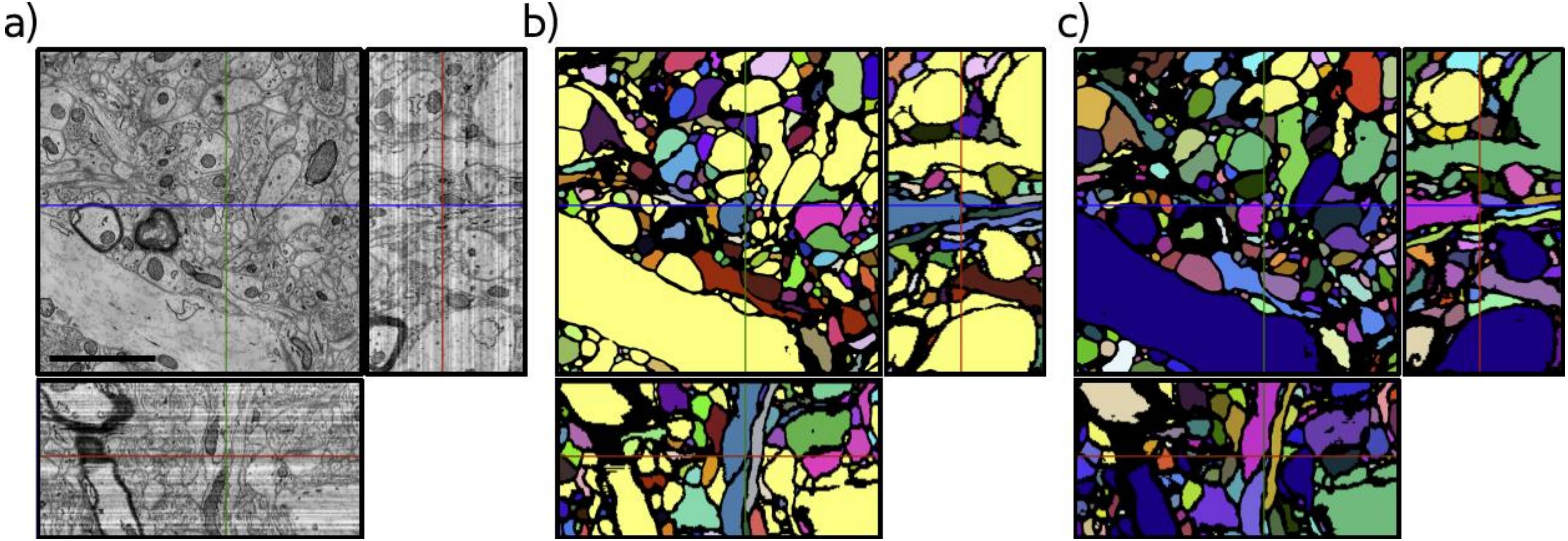
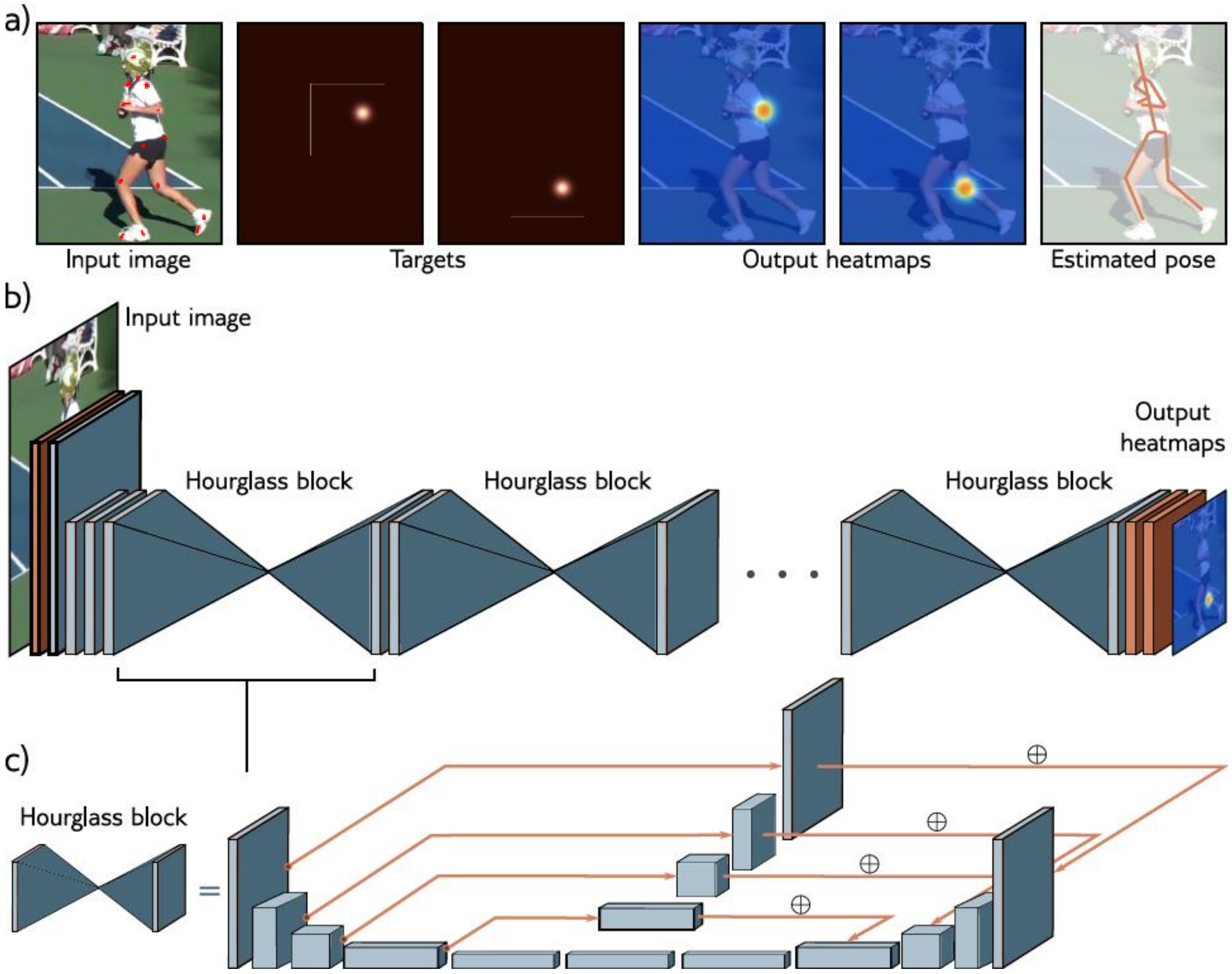


Figure 11.11 Segmentation using U-Net in 3D. a) Three slices through a 3D volume of mouse cortex taken by scanning electron microscope. b) A single U-Net is used to classify voxels as being inside or outside neurites. Connected regions are identified with different colors. c) For a better result, an ensemble of five U-Nets is trained, and a voxel is only classified as belonging to the cell if all five networks agree. Adapted from Falk et al. (2019).

Stacked hourglass networks for Pose Estimation



Newell, A., Yang, K., & Deng, J. (2016). Stacked hourglass networks for human pose estimation. European Conference on Compute r Vision, 483–499.

Any Questions?

???

- Challenges of deep networks
- Residual connections and residual blocks
- Exploding gradients in residual networks
- Batch normalization
- Common residual architectures