Deep Learning for Data Science
DS 542

Lecture 11
Residual Networks

Slides originally by Thomas Gardos.
Images from [Understanding Deep Learning] unless otherwise cited.

Where we are

=== Foundational Concepts ===

- ✔ 02 -- Supervised learning refresher
- ✔ 03 -- Shallow networks and their representation capacity
- ✔ 04 -- Deep networks and depth efficiency
- ✔ 05 -- Loss function in terms of maximizing likelihoods
- ✔ 06 – Fitting models with different optimizers
- ✔ 07a – Gradients on deep models and backpropagation
- ✔ 07b – Initialization to avoid vanishing and exploding weights & gradients
- ✔ 08 – Measuring performance, test sets, overfitting and double descent
- ✔ 09 – Regularization to improve fitting on test sets and unseen data

=== Network Architectures and Applications ===

- ✔ 10 – Convolutional Networks
- 11 – Residual Networks and Recurrent Neural Networks
- 12 – Transformers
- Large Language and other Foundational Models
- Generative Models
- Graph Neural Networks
- …

Topics

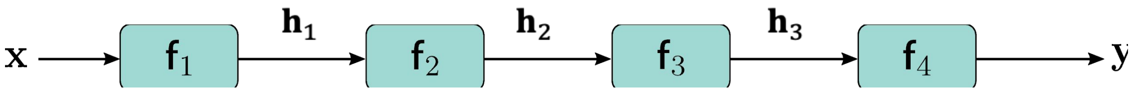
- Residual connections and residual blocks
- Exploding gradients in residual networks
- Batch normalization
- Common residual architectures

Topics

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

Previously we saw a sequential network:

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



Fully connected network:

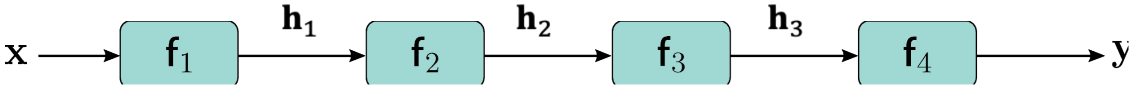
$$h_i = \mathrm{a}\left[\beta_i + \sum_{j=1}^{D} \omega_{ij} x_j\right]$$

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

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

Previously we saw a sequential network:

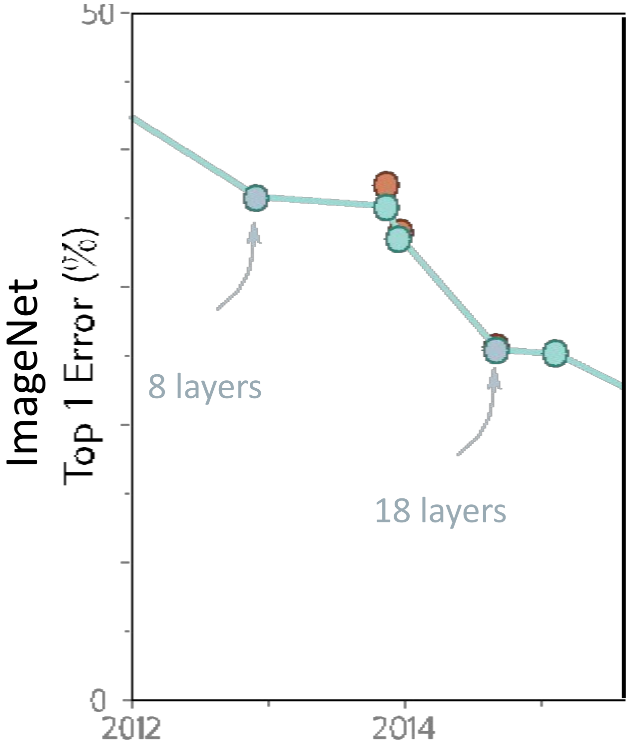
$$
\begin{aligned}
\mathbf{h}_1 &= \mathbf{f}_1[\mathbf{x}, \phi_1] \\
\mathbf{h}_2 &= \mathbf{f}_2[\mathbf{h}_1, \phi_2] \\
\mathbf{h}_3 &= \mathbf{f}_3[\mathbf{h}_2, \phi_3] \\
\mathbf{y} &= \mathbf{f}_4[\mathbf{h}_3, \phi_4]
\end{aligned}
$$



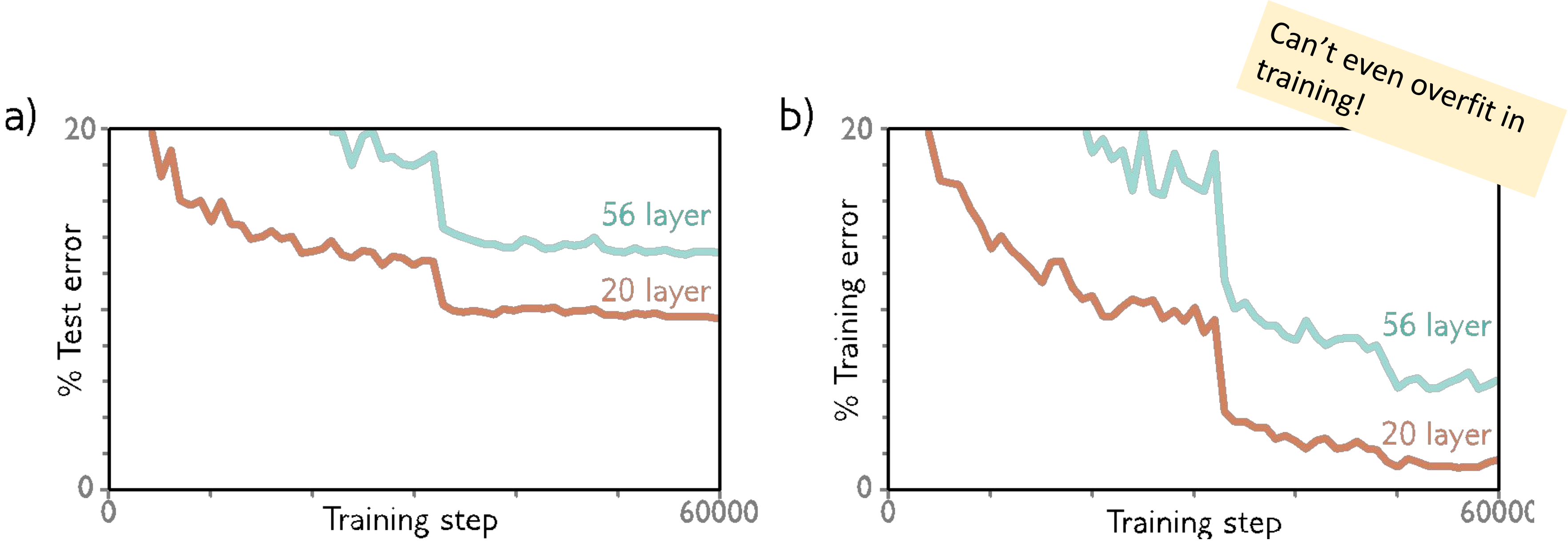
Can think of as a sequence of nested functions:

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

More layers are better…



More layers are better… up to a point



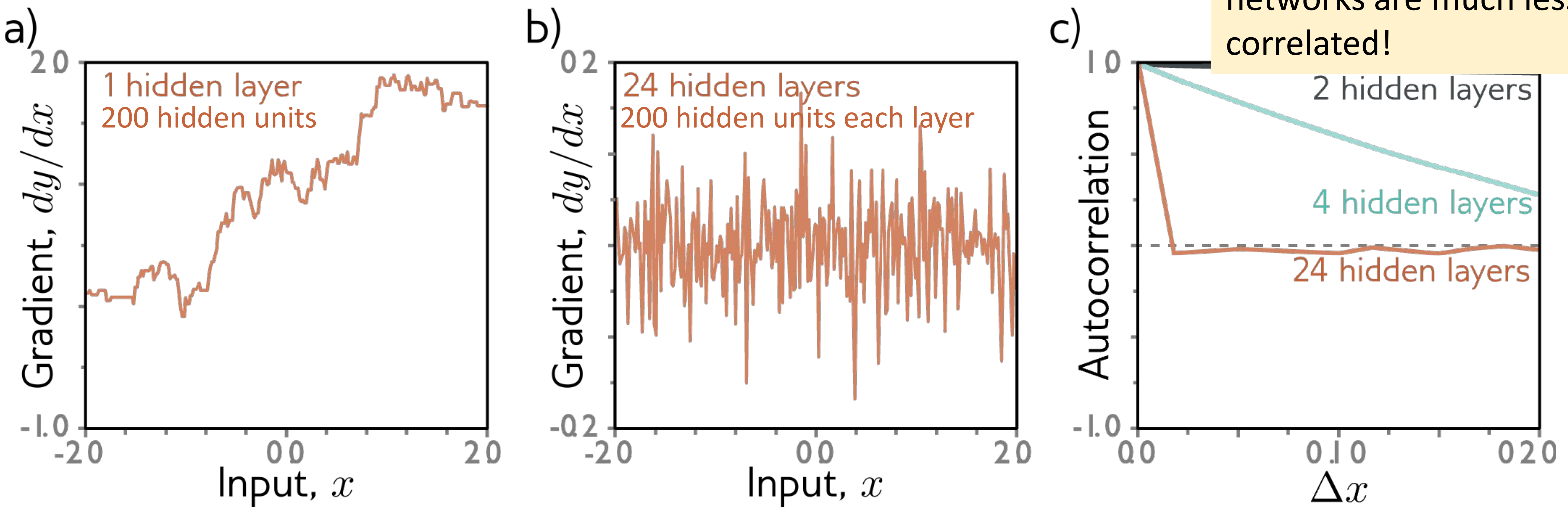
Convolutional Network on CIFAR10

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!



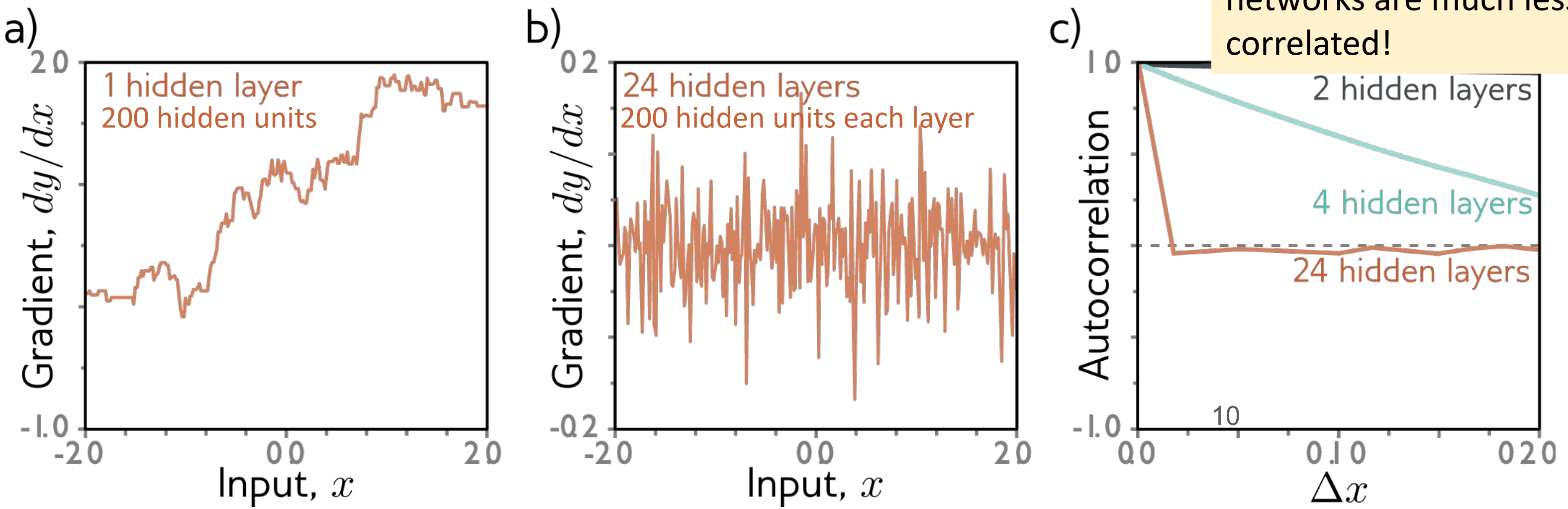
A small step in gradient descent may jump to wildly different valued gradient!

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!

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

What's going on?

The Shattered Gradient Phenomenon

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

$$\mathbf{y} = \mathbf{f}_4\left[\mathbf{f}_3\left[\mathbf{f}_2\left[\mathbf{f}_1[\mathbf{x}, \boldsymbol{\phi}_1], \boldsymbol{\phi}_2\right], \boldsymbol{\phi}_3\right], \boldsymbol{\phi}_4\right]$$

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

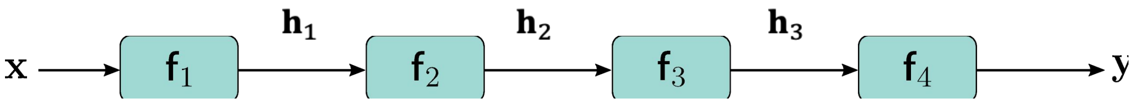
$$\frac{\partial \mathbf{y}}{\partial \mathbf{f}_1} = \frac{\partial \mathbf{f}_4}{\partial \mathbf{f}_3}\frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_2}\frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1}$$

f_1 impacts f_2 impacts f_3, etc…

Solution: Residual connections

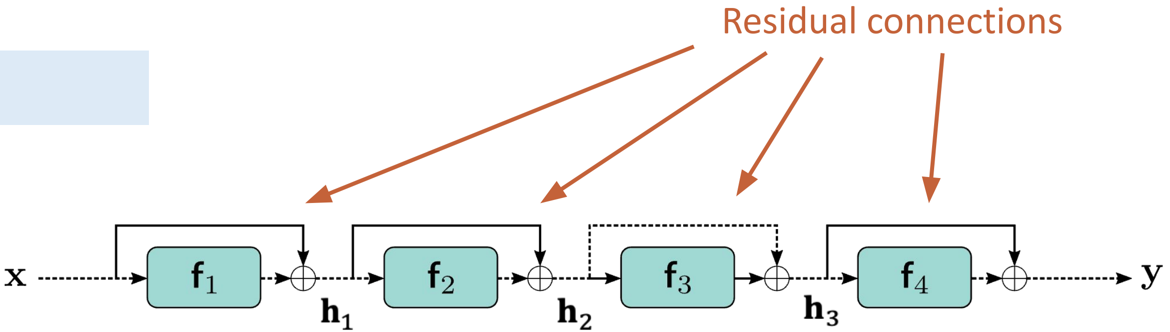
Regular sequential network:

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



Residual network:

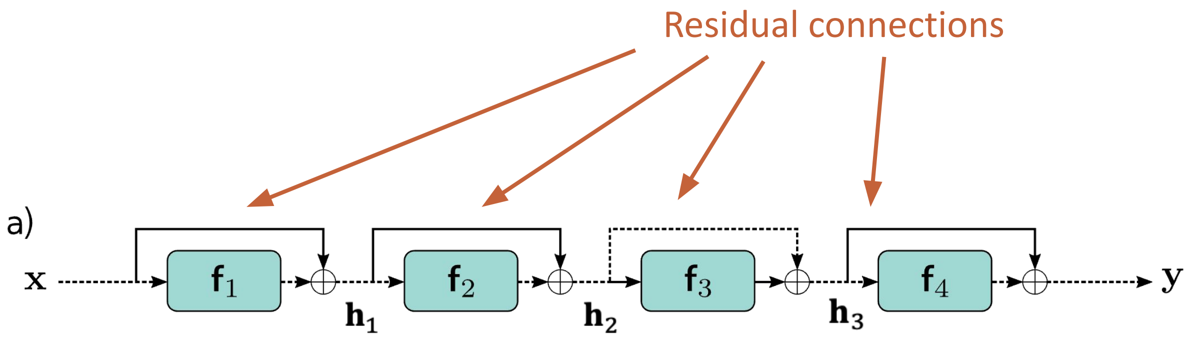
$$\begin{aligned}
\mathbf{h}_1 &= \mathbf{x} + \mathbf{f}_1[\mathbf{x}, \phi_1] \\
\mathbf{h}_2 &= \mathbf{h}_1 + \mathbf{f}_2[\mathbf{h}_1, \phi_2] \\
\mathbf{h}_3 &= \mathbf{h}_2 + \mathbf{f}_3[\mathbf{h}_2, \phi_3] \\
\mathbf{y} &= \mathbf{h}_3 + \mathbf{f}_4[\mathbf{h}_3, \phi_4]
\end{aligned}$$



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

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



Substituting in:

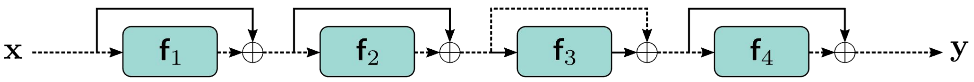
$$
\begin{aligned}
\mathbf{y} = \mathbf{x} \;+\;& \mathbf{f}_1[\mathbf{x}] \\
+\;& \mathbf{f}_2\big[\mathbf{x} + \mathbf{f}_1[\mathbf{x}]\big] \\
+\;& \mathbf{f}_3\Big[\mathbf{x} + \mathbf{f}_1[\mathbf{x}] + \mathbf{f}_2\big[\mathbf{x} + \mathbf{f}_1[\mathbf{x}]\big]\Big] \\
+\;& \mathbf{f}_4\Bigg[\mathbf{x}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

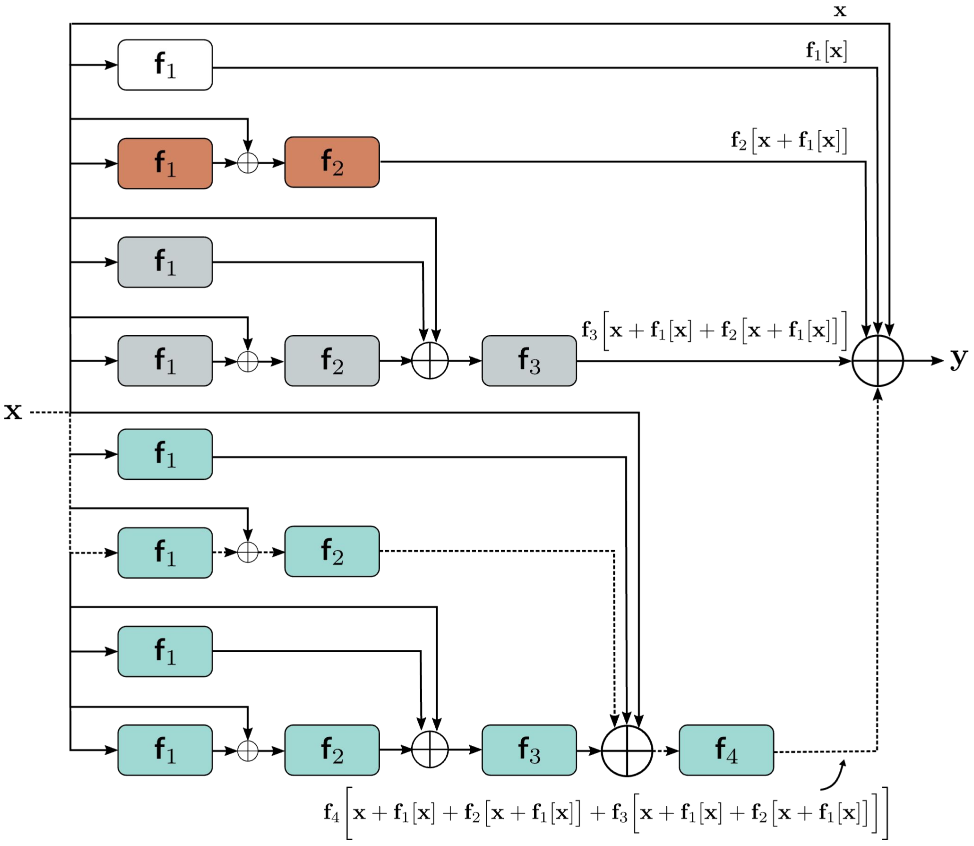
Residual Network

We can unravel all the possible paths

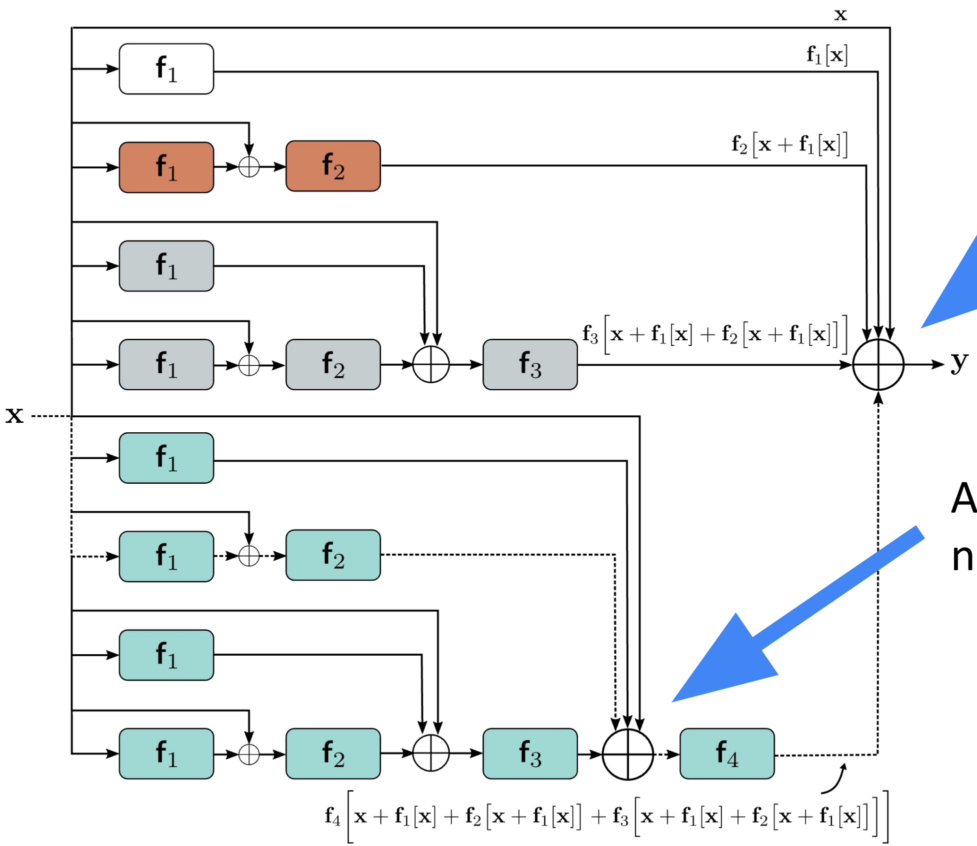
The output is the sum of the input plus 4 partial networks.

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





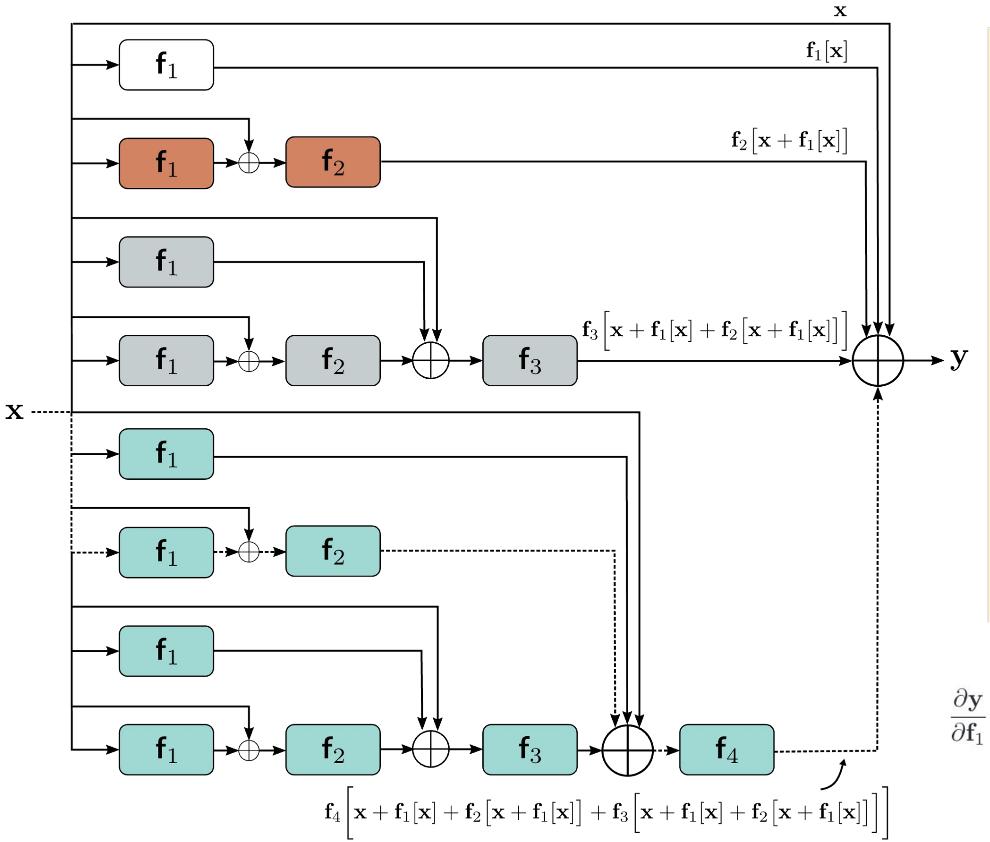
Residual Network as Ensemble of Networks



Ensemble of four networks

Another ensemble of four networks

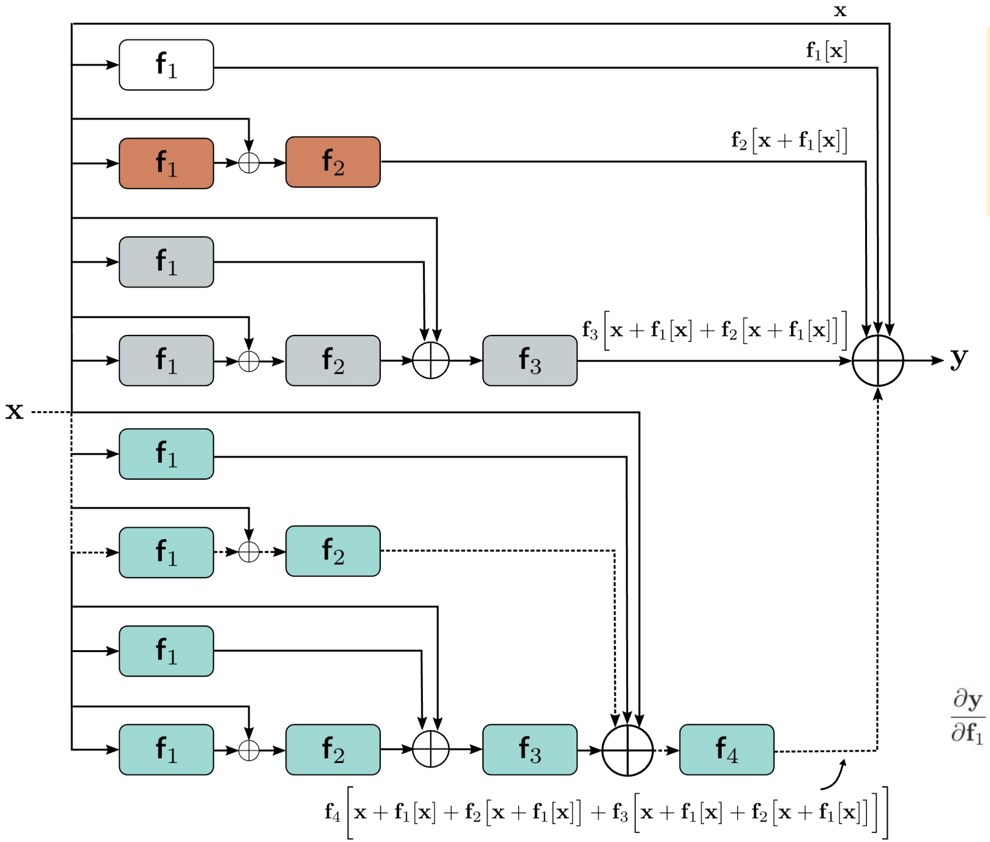
Residual Network as Ensemble of Networks



- 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.

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

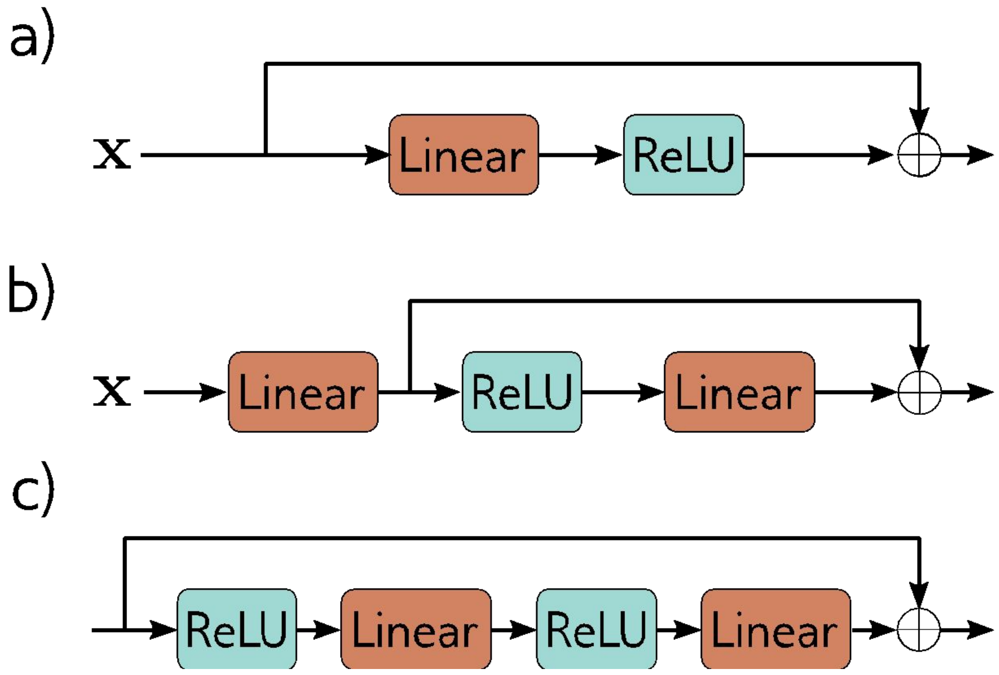
Residual Network as Ensemble of Networks



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 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.

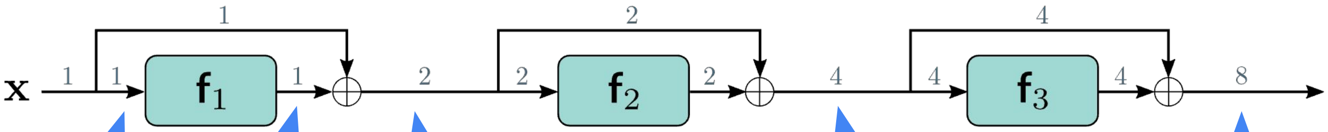
Note: if we start with a ReLU, then will clamp negative values and so do nothing

This helps increase depth up to a point…

Topics

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

Exploding Gradients in Residual Networks

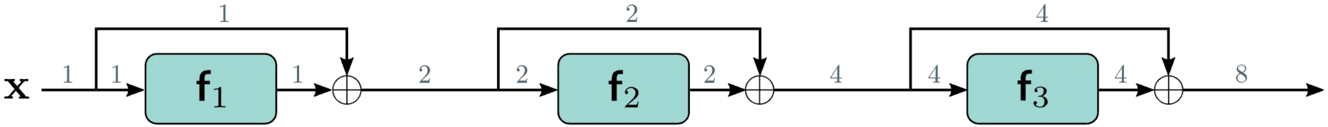


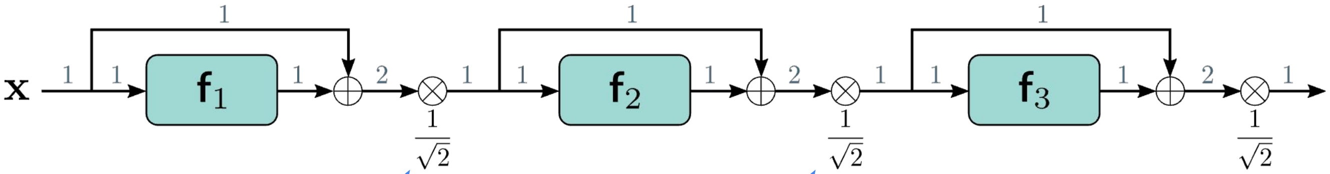
With He initialization we can control the variance inside the block

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*.

Topics

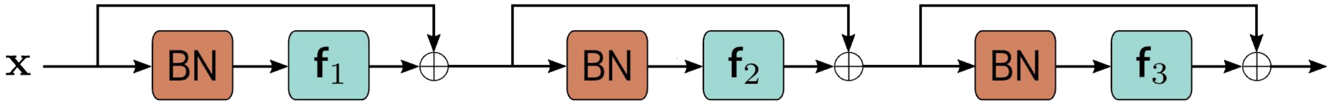
- Residual connections and residual blocks
- Exploding gradients in residual networks
- Batch normalization
- Common residual architectures

Batch Normalization

We already talked about batch normalization in the context of regularization…

- Layer normalization is generally considered better now, but it wasn't invented yet when the following work was done.

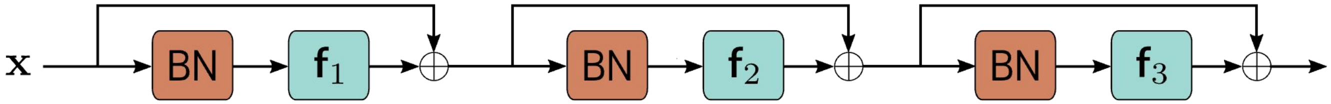
Batch Normalization (a.k.a. *BatchNorm*)



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

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*)



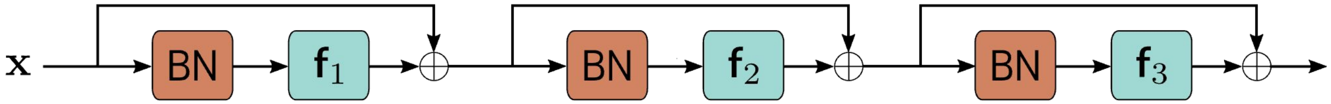
- Shifts and rescales each activation so that its mean and variance across the batch become values that are learned during training

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

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

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

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.

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

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

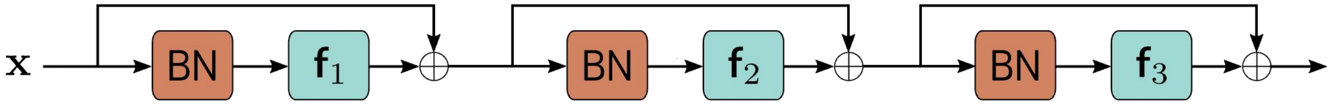
Standardize (*normalize*) to zero-mean and unit standard deviation.

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

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*)

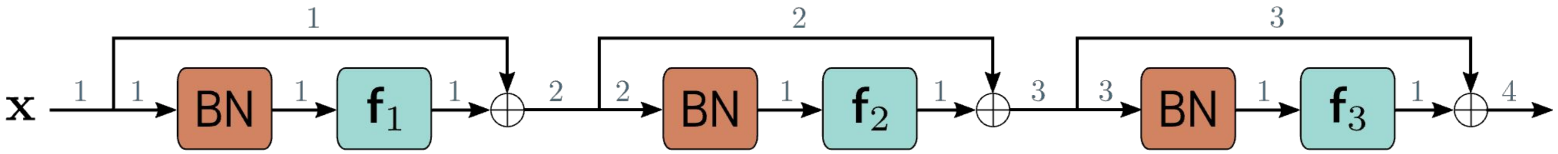


- Applied independently to each hidden unit

- 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



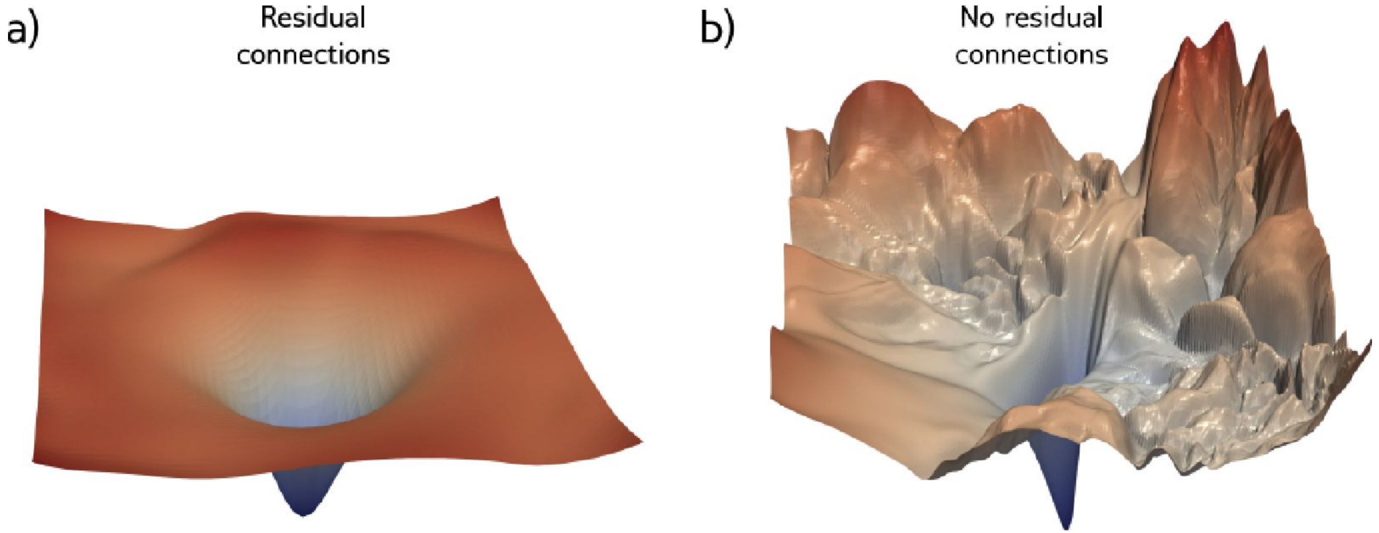
Stable forward propagation

- 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

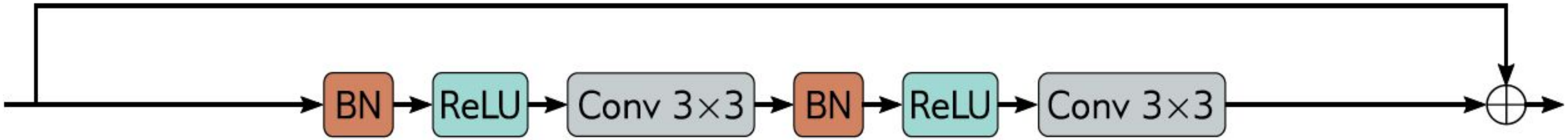
BatchNorm injects noise since BN scale and shift depend on batch statistics

Topics

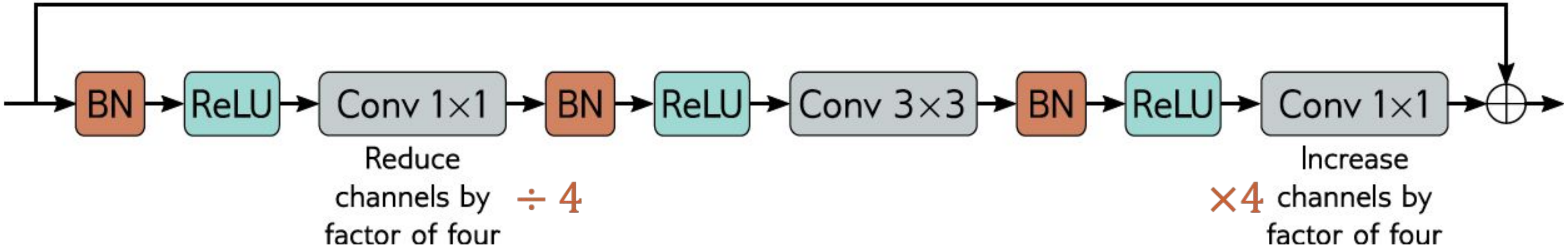
- Residual connections and residual blocks
- Exploding gradients in residual networks
- Batch normalization
- Common residual architectures

ResNet (2015)

ResNet Block

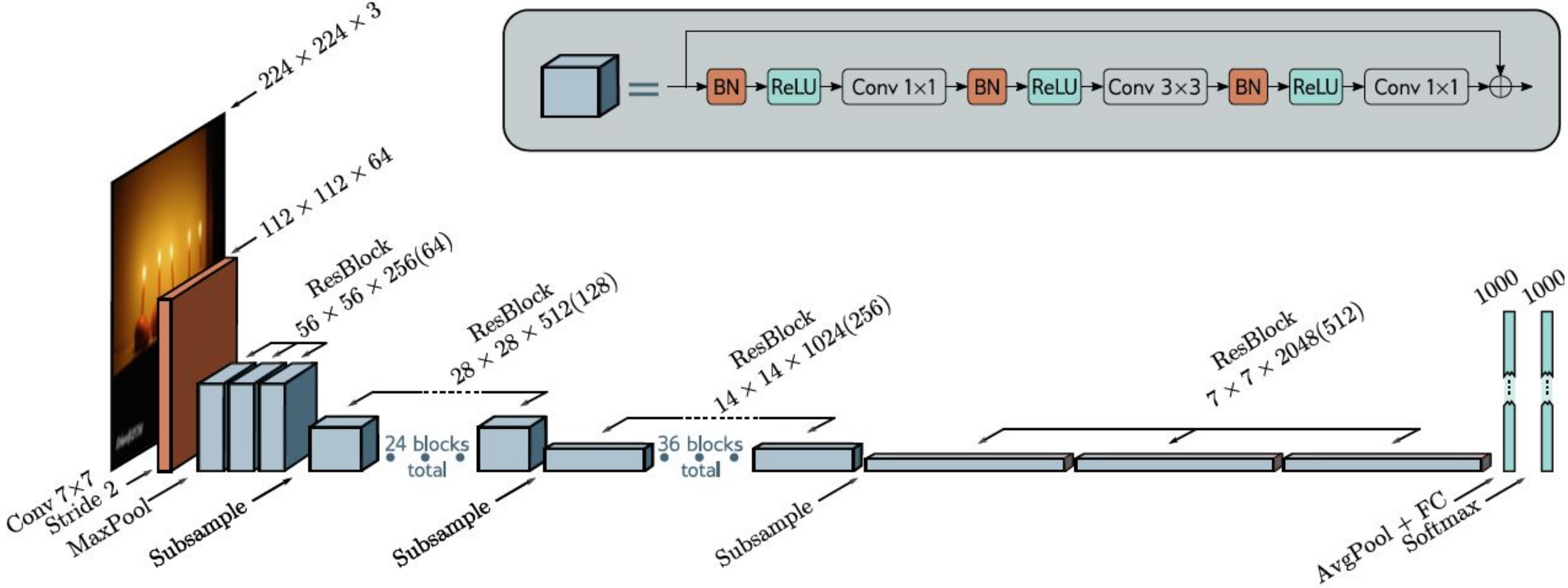


Bottleneck Residual



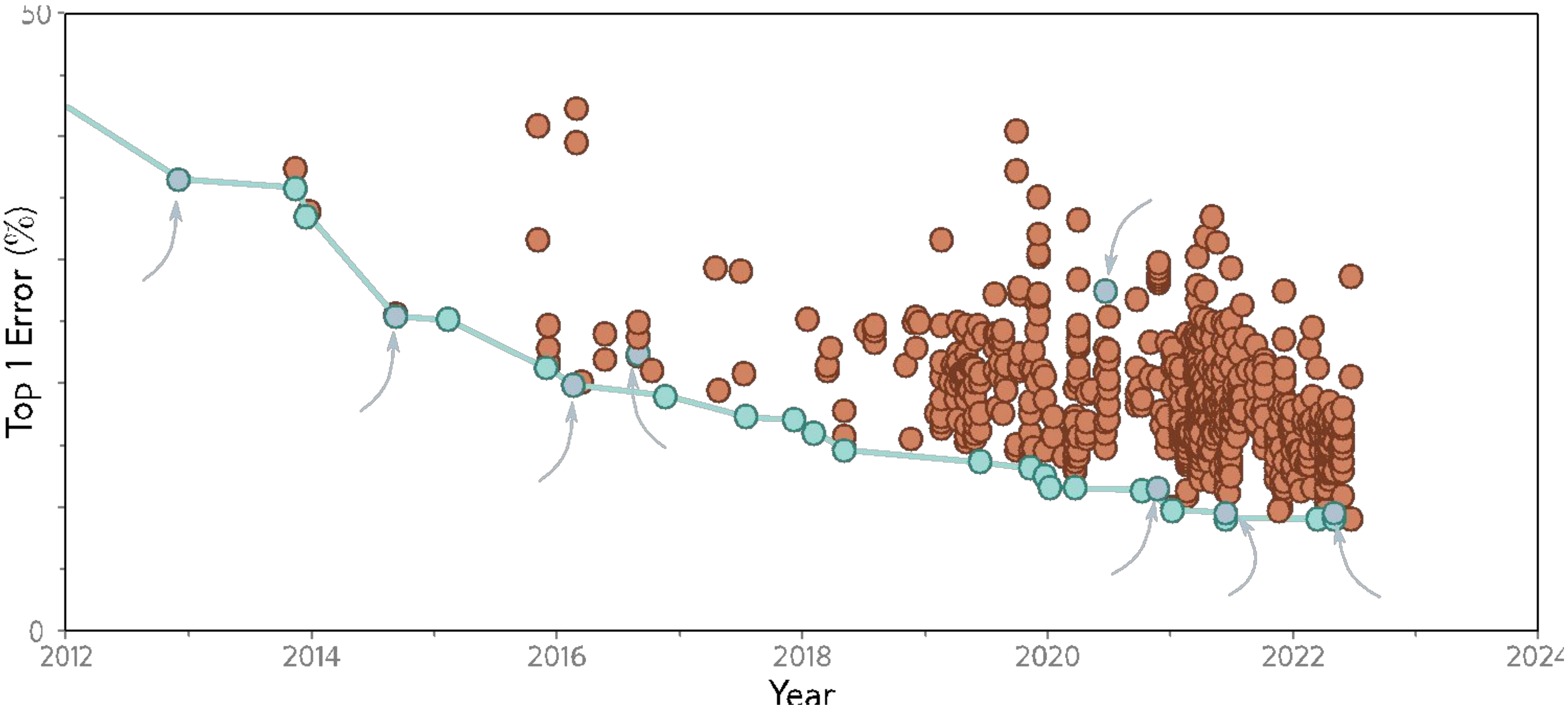
K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” *arXiv:1512.03385 [cs]*, Dec. 2015, http://arxiv.org/abs/1512.03385

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

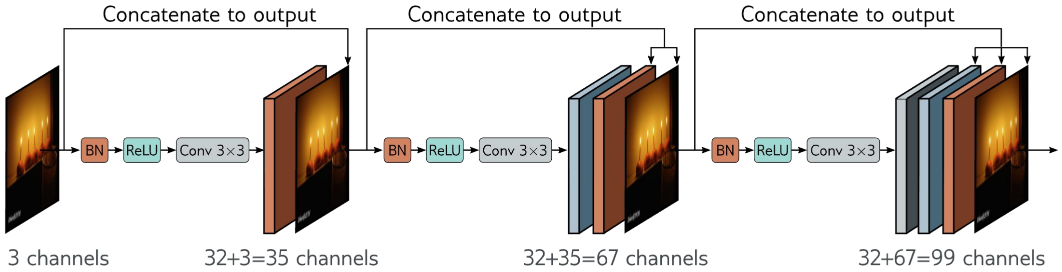


Figure from
UDL

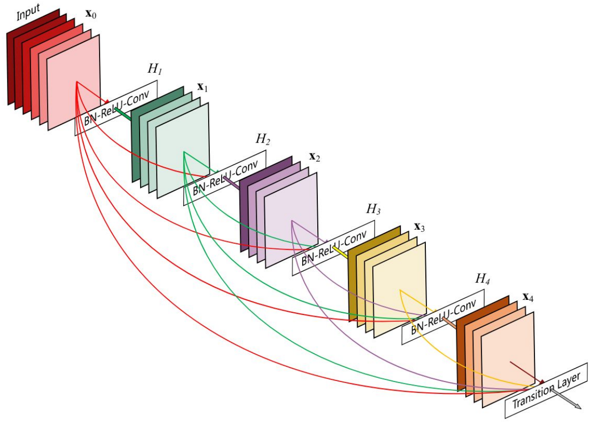
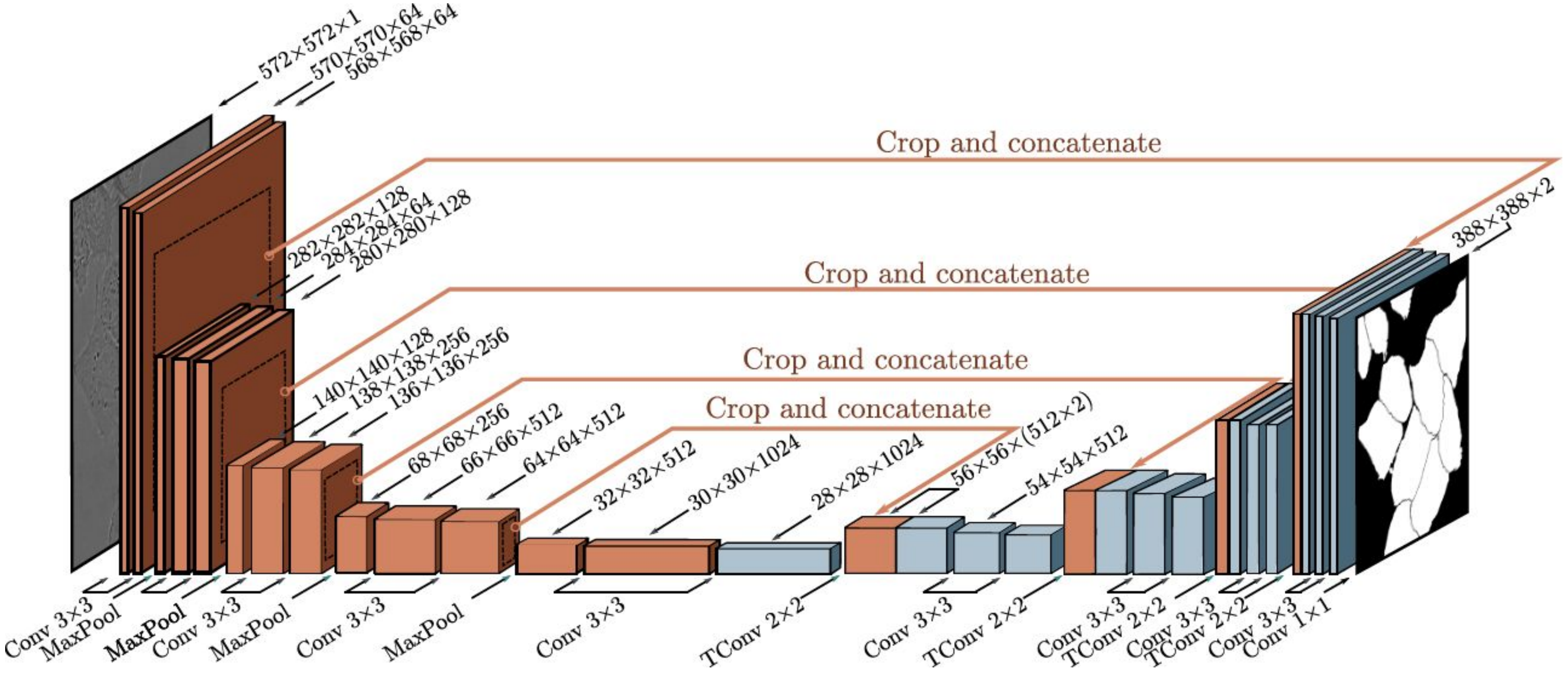


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

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

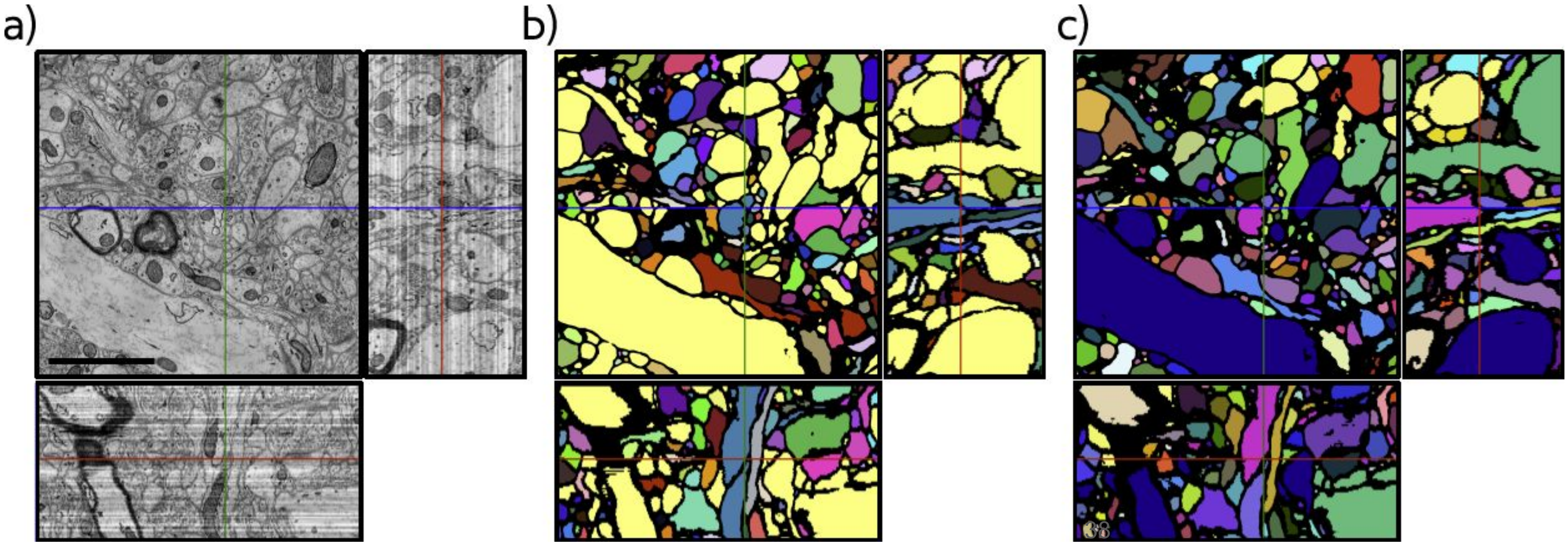
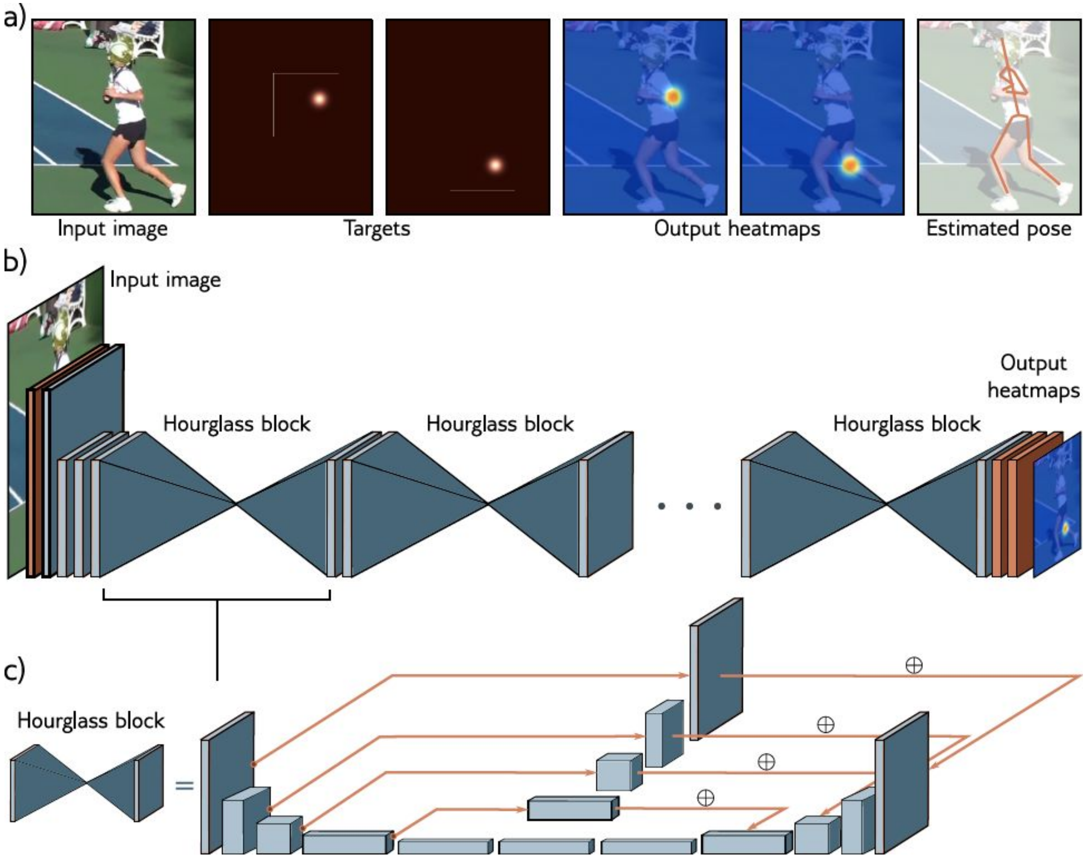


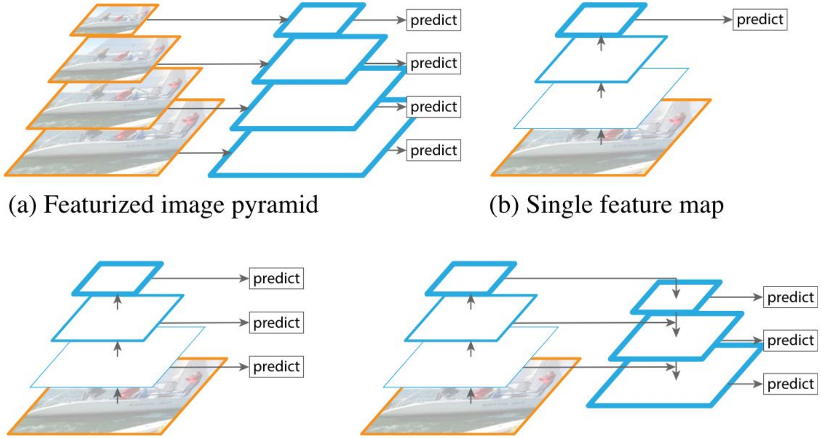
Figure 11.11 Segmentation using U-Net in 3D. a) Three slices through a 3D volume of mouse cortex taken by scanning electron microscope. b) A single U-Net is used to classify voxels as being inside or outside neurites. Connected regions are identified with different colors. c) For a better result, an 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 Computer Vision, 483–499.

Feature Pyramid Networks



(a) Featurized image pyramid

(b) Single feature map

(c) Pyramidal feature hierarchy

(d) Feature Pyramid Network

Figure 1. (a) Using an image pyramid to build a feature pyramid. Features are computed on each of the image scales independently, which is slow. (b) Recent detection systems have opted to use only single scale features for faster detection. (c) An alternative is to reuse the pyramidal feature hierarchy computed by a ConvNet as if it were a featurized image pyramid. (d) Our proposed Feature Pyramid Network (FPN) is fast like (b) and (c), but more accurate. In this figure, feature maps are indicate by blue outlines and thicker outlines denote semantically stronger features.

T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature Pyramid Networks for Object Detection,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI: IEEE, Jul. 2017, pp. 936–944. doi: 10.1109/CVPR.2017.106.

Feature Pyramid Networks

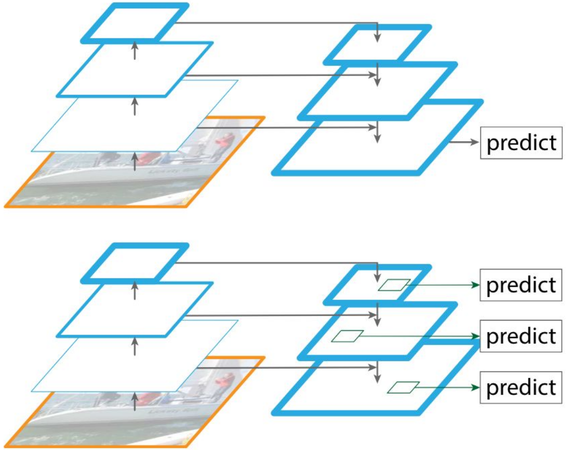


Figure 2. Top: a top-down architecture with skip connections, where predictions are made on the finest level (*e.g.*, [28]). Bottom: our model that has a similar structure but leverages it as a *feature pyramid*, with predictions made independently at all levels.

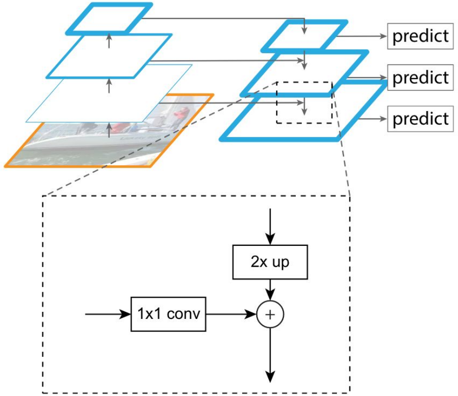


Figure 3. A building block illustrating the lateral connection and the top-down pathway, merged by addition.

T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature Pyramid Networks for Object Detection,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI: IEEE, Jul. 2017, pp. 936–944. doi: 10.1109/CVPR.2017.106.

Midterm Reminder

Please bring your laptops to both classes next week.

- Tuesday: Review / coding practice
- Wednesday: Midterm starts!
- Friday: Midterm due

Feedback?