Outline

ConvNets as Deep Neural Networks for Vision

1. Neural Nets
2. Deep Convolutional Neural Networks
3. Modern Deep Architectures
   - The big picture
   - AlexNet
   - GoogleNet
   - VGG
   - **ResNet**

ResNet
The deeper, the better

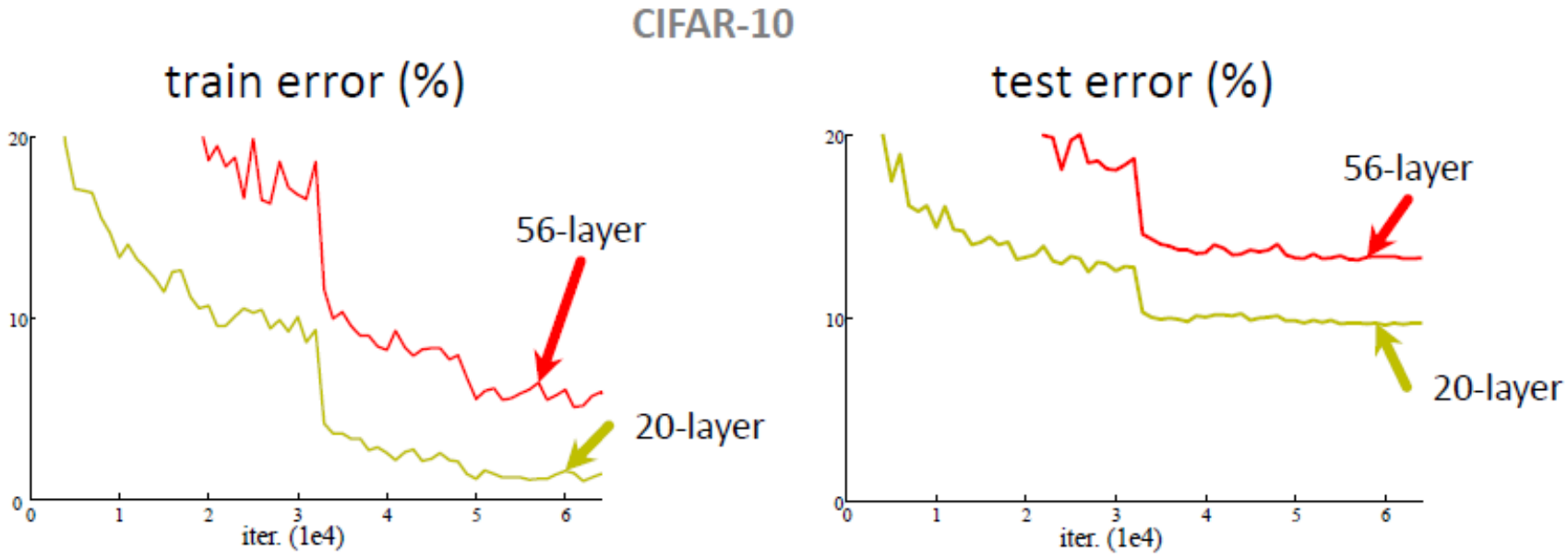
+ Deeper network covers more complex problems
  - Receptive field size ↑
  - Non-linearity ↑
- Training deeper network more difficult because of vanishing/exploding gradients problem

@ Kaiming He ILSVRC & COCO 2015

Deeper VGG: 56 Plain Network

Plain nets: stacking 3x3 conv layers

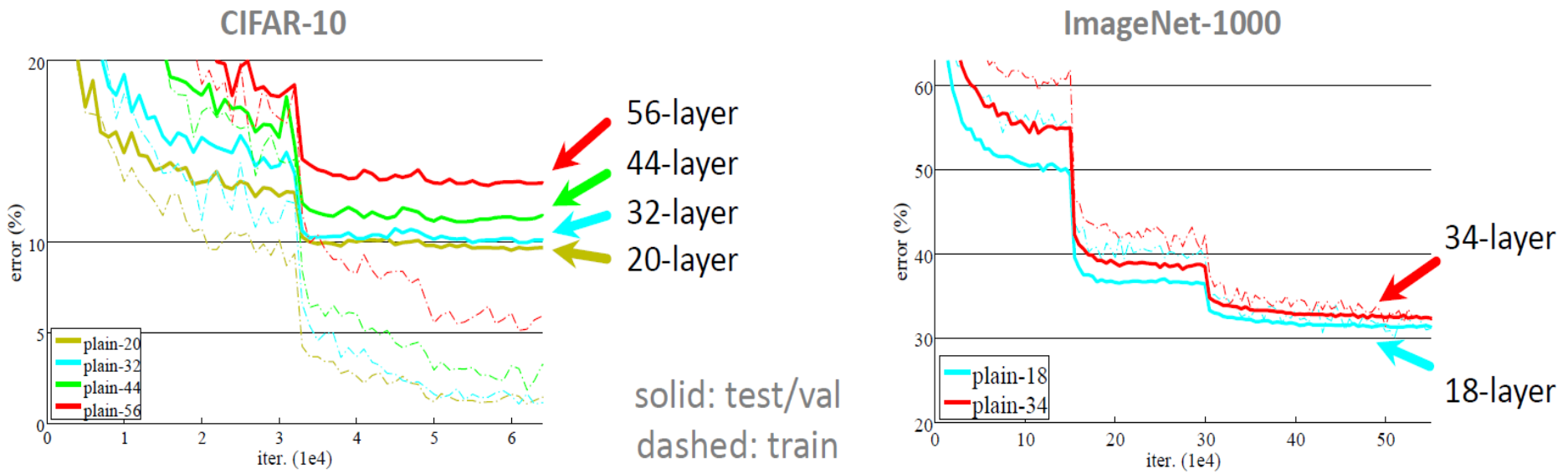
- 56-layer net has higher training error and test error than 20-layers net



Deeper VGG:

"Overly deep" plain nets have higher training error

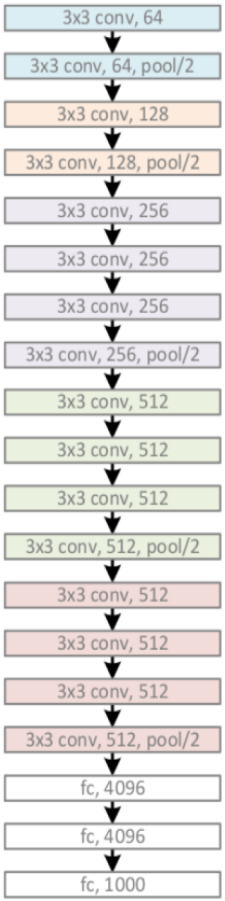
A general phenomenon, observed in many datasets



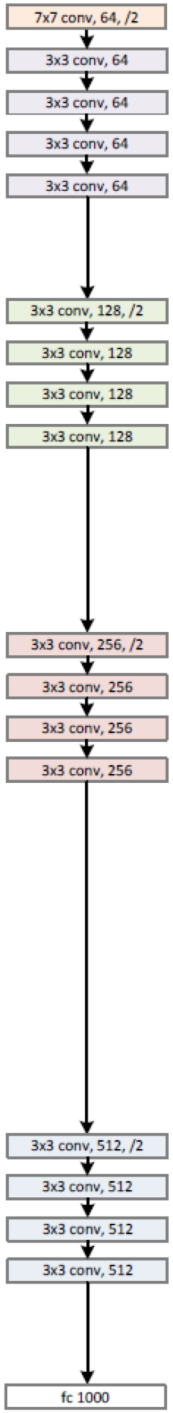
Residual Network

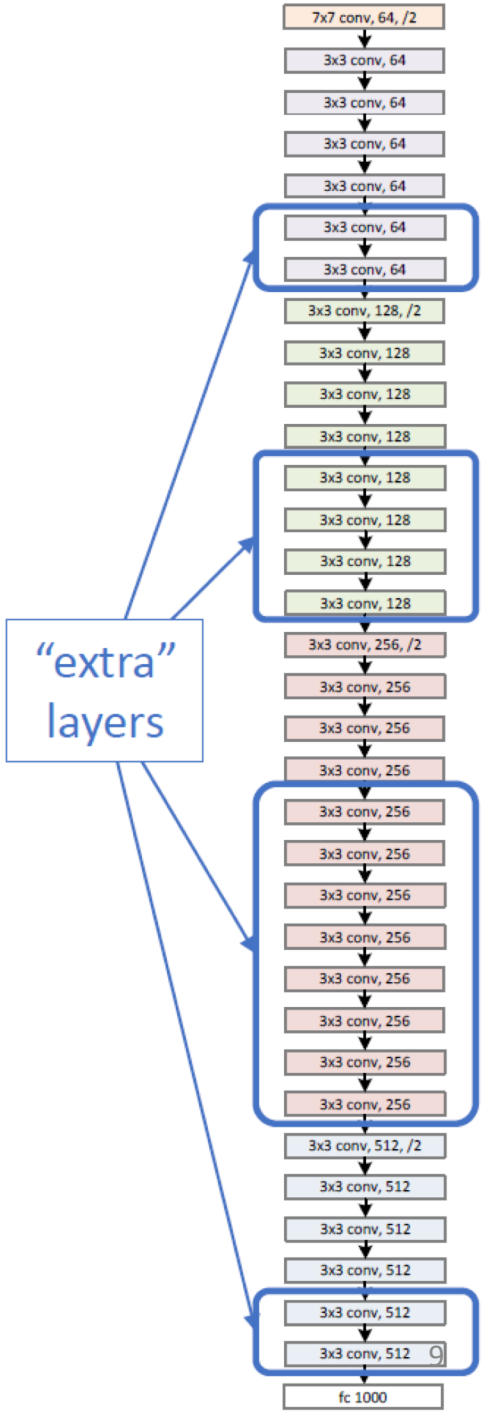
Naïve solution

If extra layers identity mapping, training error not increase



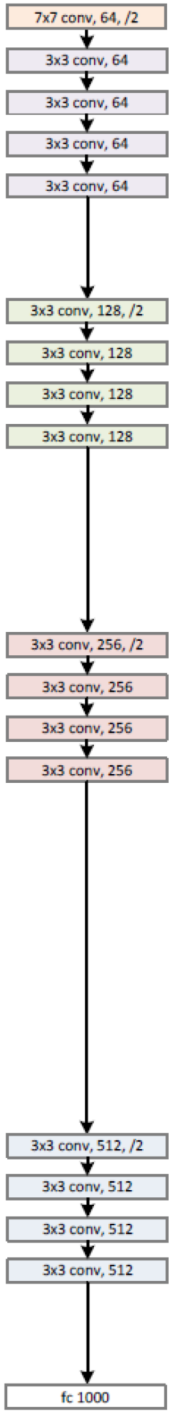
VGG, 16/19 layers, 2014

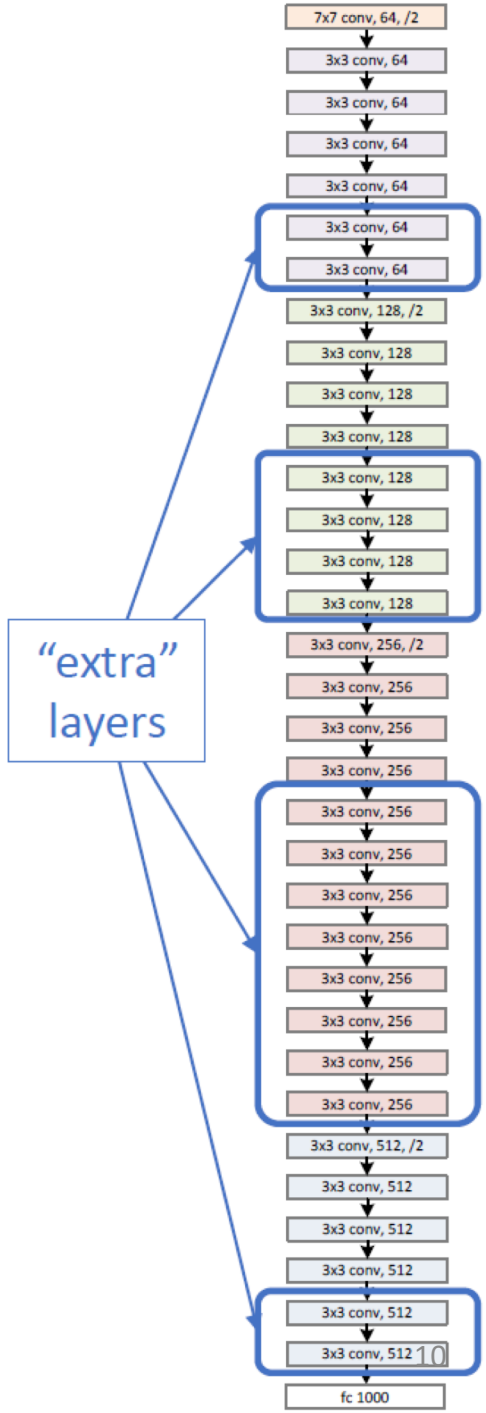




Residual Network

- Deeper networks maintain the tendency of results
  - Features in same level will be almost same
  - An amount of changes is fixed
  - Adding layers make smaller differences
  - Optimal mappings closer to an identity

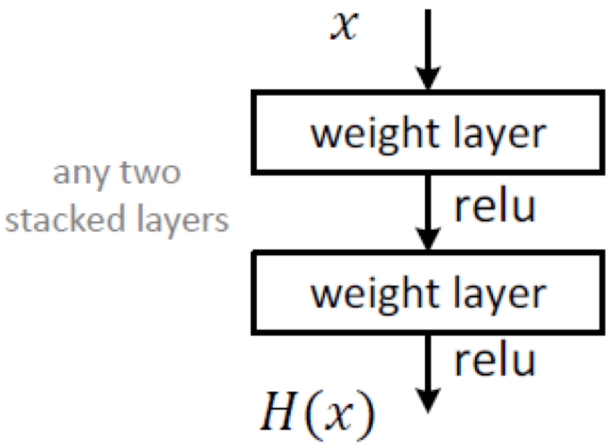




Residual Network

Plain block

- Difficult to make identity mapping because of multiple non-linear layers

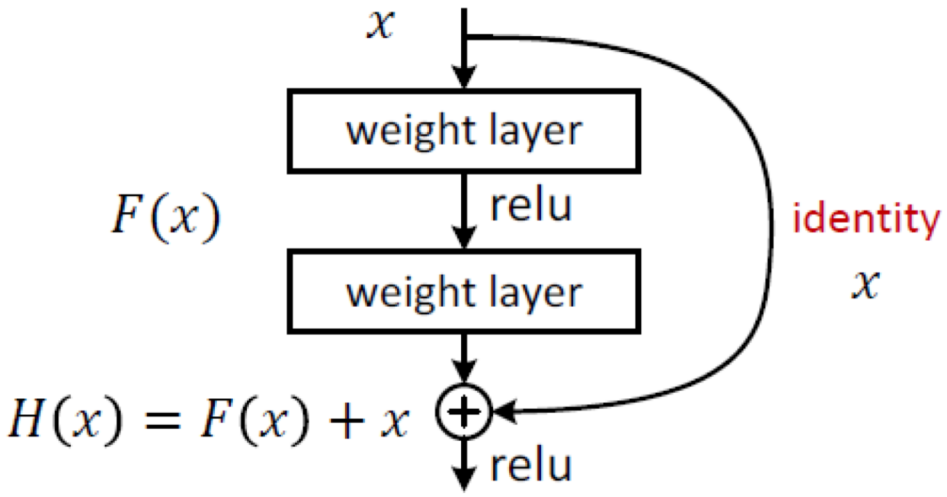


Residual Network

Residual block

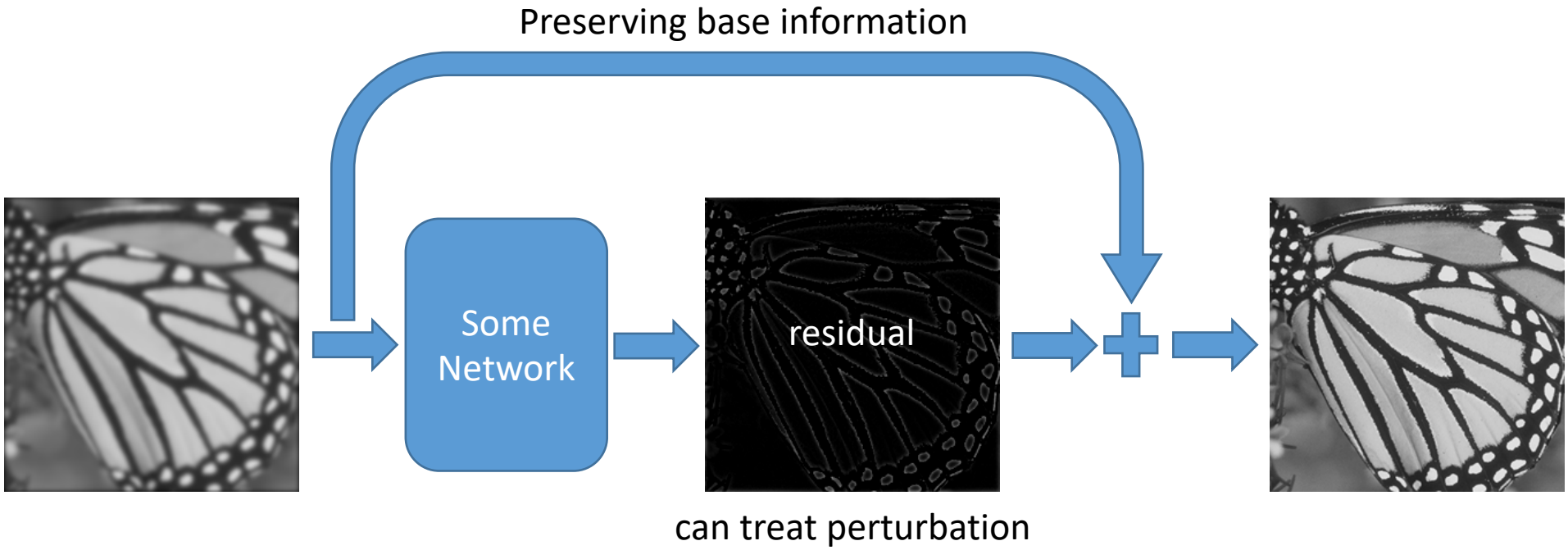
If identity were optimal,
easy to set weights as 0
If optimal mapping is closer
to identity, easier to find
small fluctuations

-> Appropriate for treating
perturbation as keeping a
base information



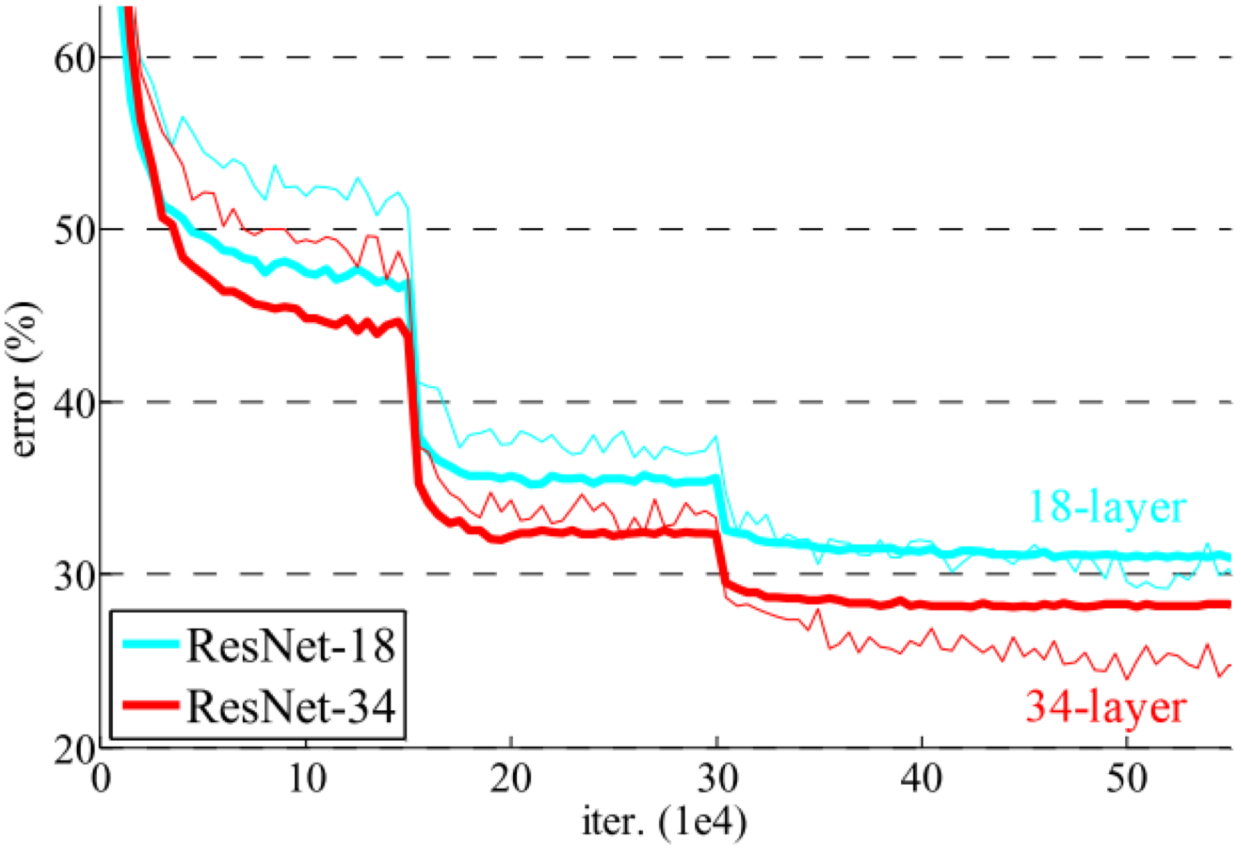
Residual Network

- Difference between an original image and a changed image



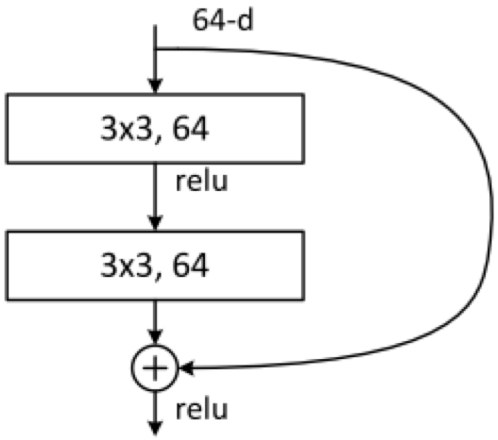
Residual Network

Deeper ResNets have lower training error

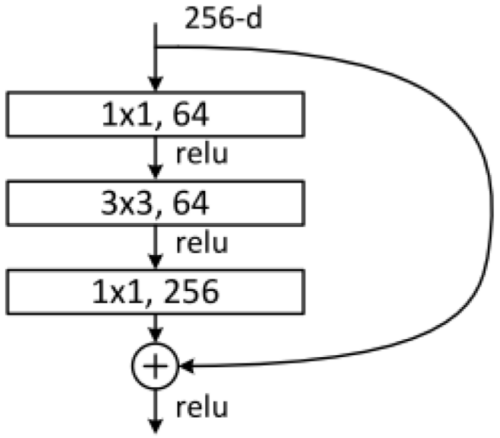


Residual Network

- Residual block
  - Very simple
  - Parameter-free



A naïve residual block



"bottleneck" residual block
(for ResNet-50/101/152)

Residual Network

- Shortcuts connections
  - Identity shortcuts

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}.$$

  - Projection shortcuts

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}.$$

Network Design

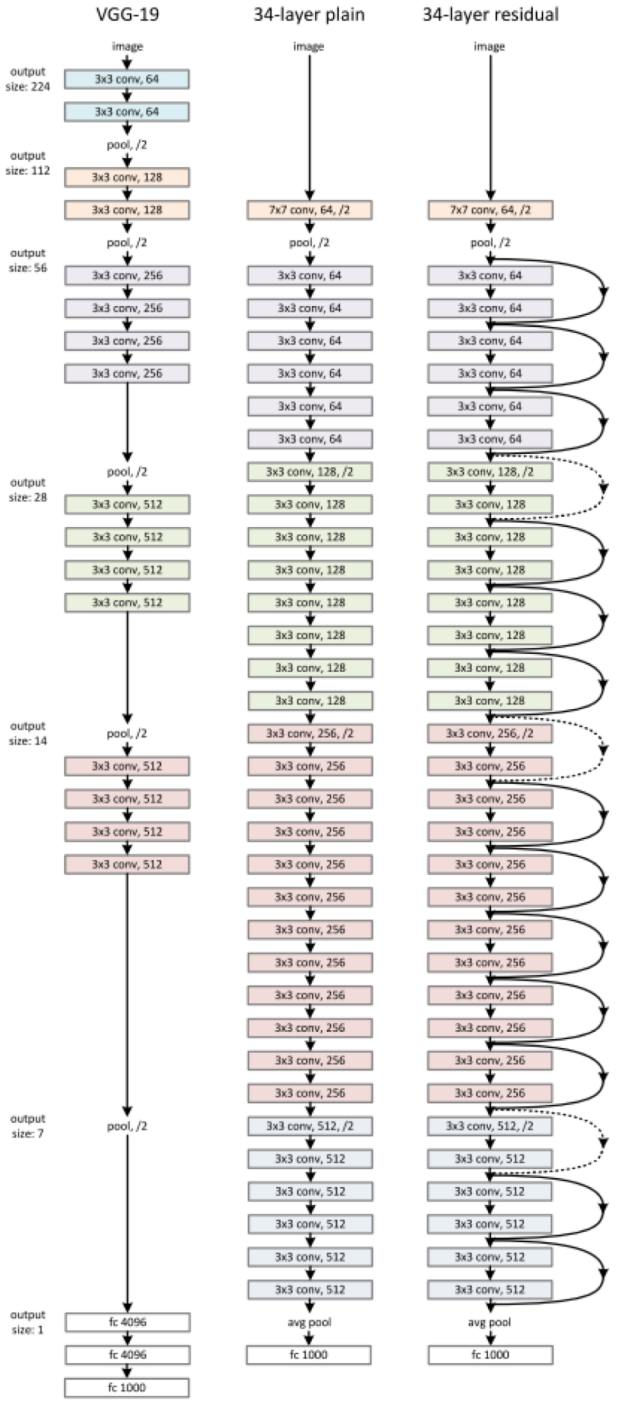
Basic design (VGG-style)

- All 3x3 conv (almost)
- Spatial size/2 => #filters x2
- Batch normalization
- Simple design, just deep

Other remarks

- No max pooling (almost)
- No hidden fc
- No dropout

ConvNet Configuration			
B	C	D	E
13 weight layers	16 weight layers	16 weight layers	19 weight layers
ut (224 × 224 RGB image)			
conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool			
conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool			
conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool			
conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool			
conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool			
FC-4096			
FC-4096			
FC-1000			
soft-max			

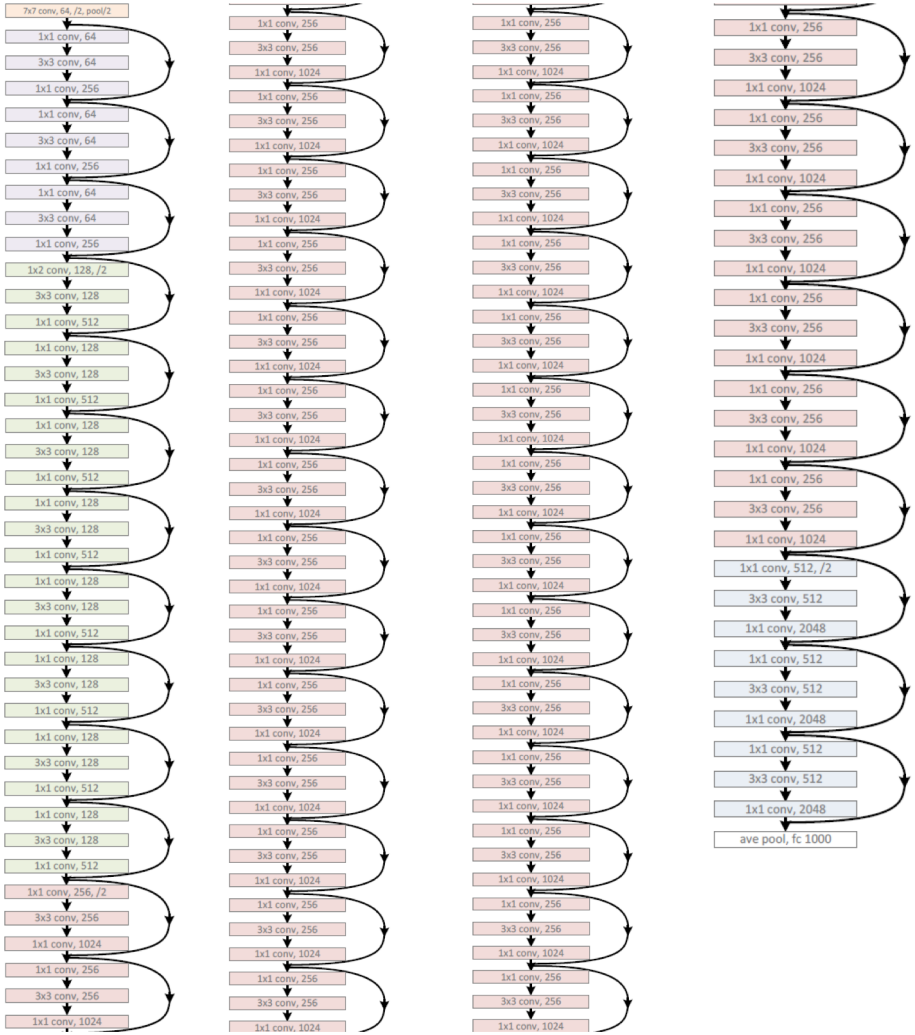


Network Design

ResNet-152

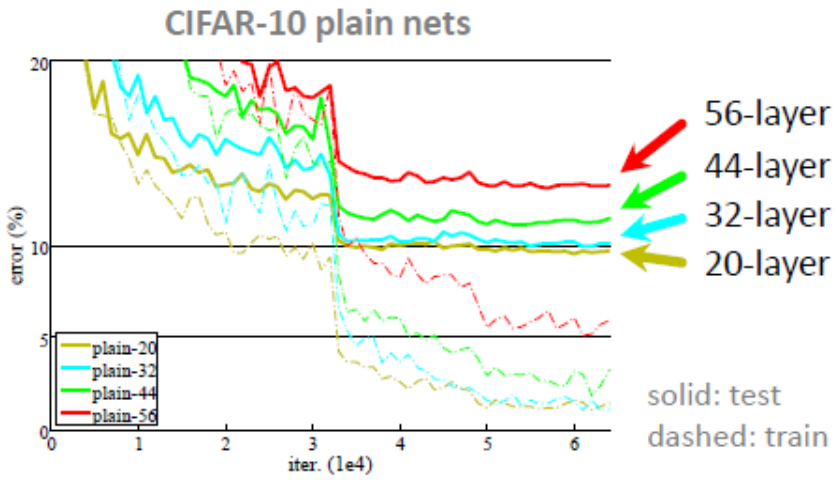
- Use bottlenecks
- ResNet-152 (11.3 billion FLOPs) lower complexity than VGG-16/19 nets (15.3/19.6 billion FLOPs)

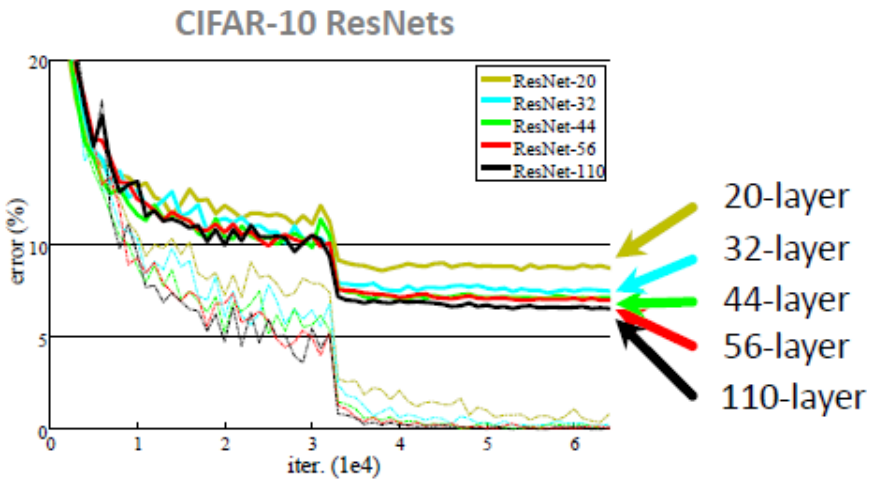
About 64M parameters



Results

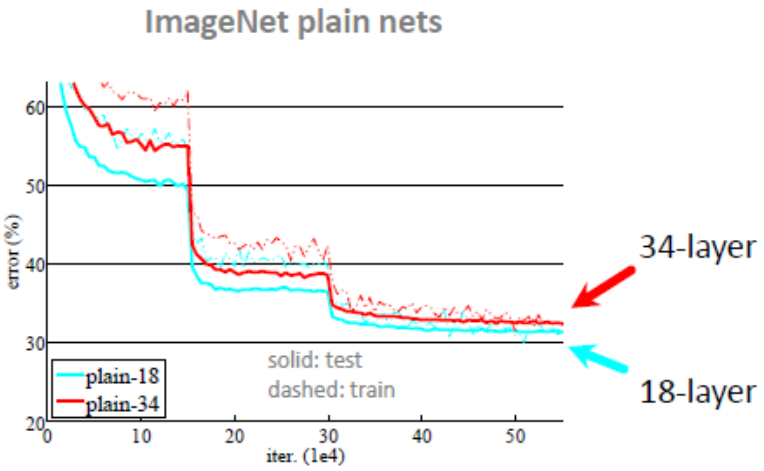
- Deep Resnets can be trained without difficulties
- Deeper ResNets have lower training error, and also lower test error

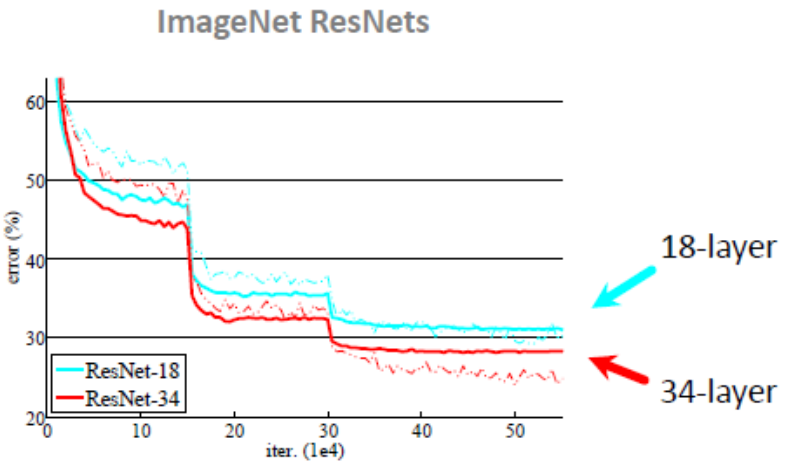




Results

- Deep Resnets can be trained “without difficulties”
- Deeper ResNets have lower training error, and also lower test error



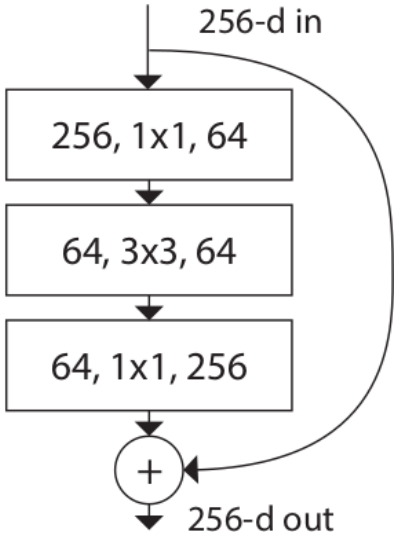


Results

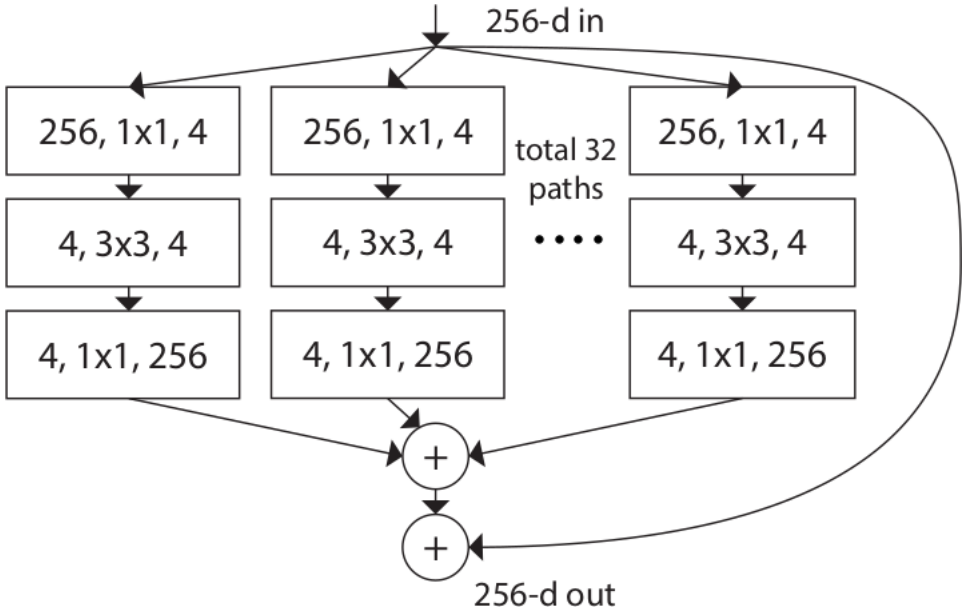
- 1st places in all five main tracks in “ILSVRC & COCO 2015 Competitions”
  - ImageNet Classification
  - ImageNet Detection
  - ImageNet Localization
  - COCO Detection
  - COCO Segmentation

Deep ConvNets for image classification

- **ResNeXt**
  - Multi-branch architecture



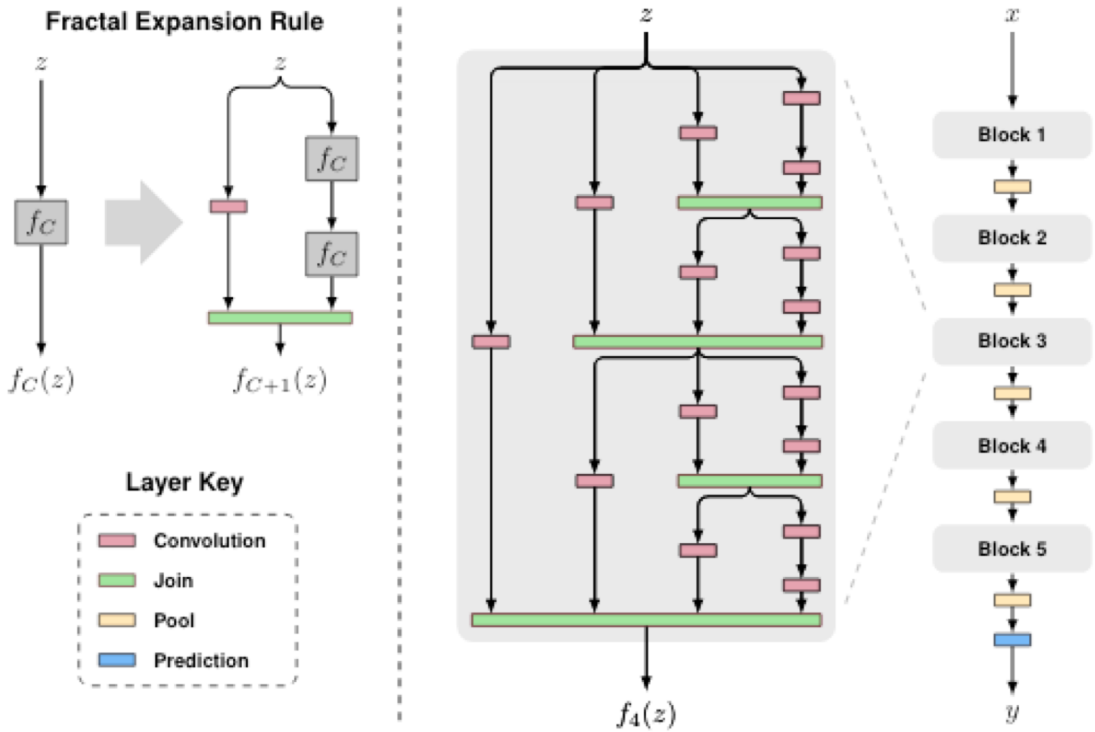
ResNet



ResNeXt

Saining Xie, Ross Girshick, Piotr Dollàr, Zhuowen Tu and Kaiming He
Aggregated Residual Transformations for Deep Neural Networks.
In *CVPR*, 2017.

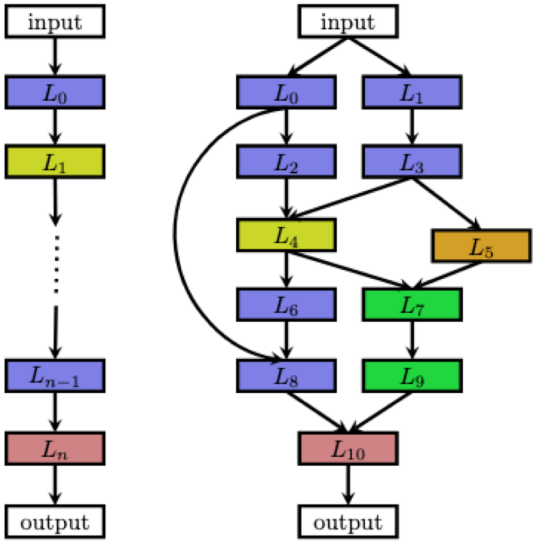
FractalNet: Ultra-Deep Neural N etworks without Residuals

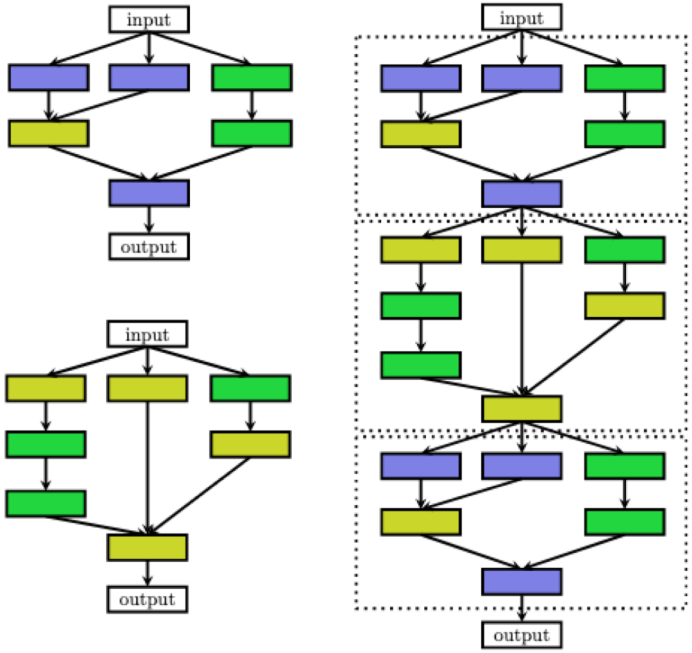


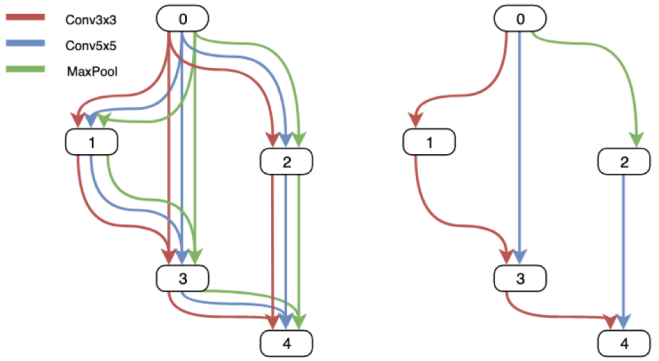
Comments from paper: *"Our experiments demonstrate that **residual representation is not fundamental to the success of extremely deep** convolutional neural networks. A fractal design achieves an error rate of 22.85% on CIFAR-100, matching the state-of-the-art held by residual networks."*

Exploring type of deep modules in Neural Nets

Neural Architecture Search







Conclusion

- ResNet: currently the best archi for large scale image classification
- Not yet consensus about the design of the Net (cf. FractalNet), Neural Architecture Search

- Fully Convolutional Net (FCN) very interesting option
  Beyond classification!

COMPLEMENTS

Batch normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

- Improves gradient flow through the network
- Allows higher learning rates
- Reduces the strong dependence on initialization
- Reduces need for dropout

Un-normalization!! Re-compute and apply the optimal scaling and bias for each neuron!
Learn γ and β (same dims as μ and σ^2).
It can (should?) learn the identity mapping!

Batch Normalization Gradients

[Ioffe and Szegedy, 2015]

$$\frac{\partial \ell}{\partial \widehat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$

$$\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2} (\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2}$$

$$\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} = \left(\sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \right) + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{\sum_{i=1}^{m} -2(x_i - \mu_{\mathcal{B}})}{m}$$

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$

$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \widehat{x}_i$$

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i}$$

Don't need these directly, they are subexpressions for the other gradients.

Think of this as backprop for the nodes $\hat{x}$, σ_B^2, μ_B, which are all internal to the minibatch update.

Batch Normalization Gradients

[Ioffe and Szegedy, 2015]

$$\frac{\partial \ell}{\partial \widehat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$

$$\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2} (\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2}$$

$$\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} = \left(\sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \right) + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{\sum_{i=1}^{m} -2(x_i - \mu_{\mathcal{B}})}{m}$$

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$

Gradient to propagate to the input layer

$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \widehat{x}_i$$

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i}$$

Batch Normalization Gradients

[Ioffe and Szegedy, 2015]

$$\frac{\partial \ell}{\partial \widehat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$

$$\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2} (\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2}$$

$$\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} = \left(\sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \right) + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{\sum_{i=1}^{m} -2(x_i - \mu_{\mathcal{B}})}{m}$$

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$

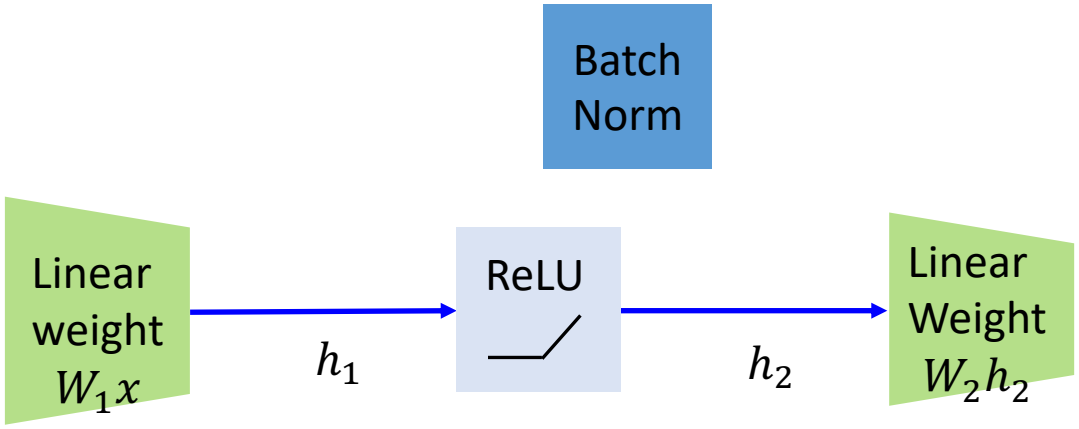
$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \widehat{x}_i$$

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i}$$

Gradients for the learnable parameters γ and β.

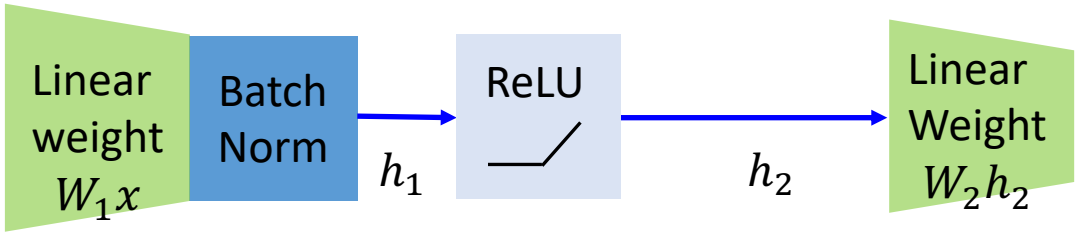
Where to do BatchNorm ?

BatchNorm is just a linear scale/bias layer in the limit of large batch sizes (μ, σ^2)



Where to do BatchNorm ?

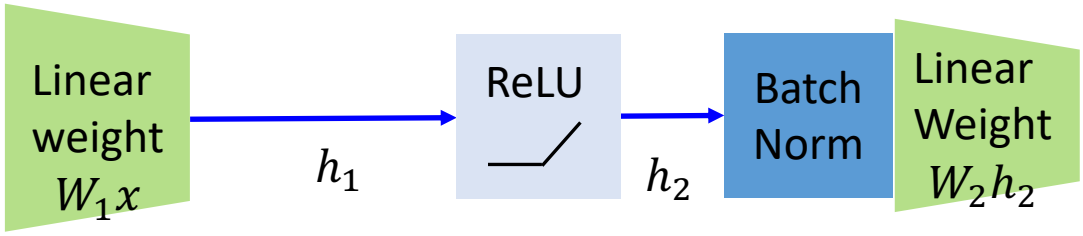
BatchNorm is just a linear scale/bias layer in the limit of large batch sizes (μ, σ^2).



?? Linear weight layer should already have optimal scale/bias in its output

Where to do BatchNorm ?

BatchNorm is just a linear scale/bias layer in the limit of large batch sizes (μ, σ^2).



?? Any bias/scaling by the batch norm layer should be over-ruled by the second linear weight layer.

Multitask learning / auxiliary loss in GoogLeNet