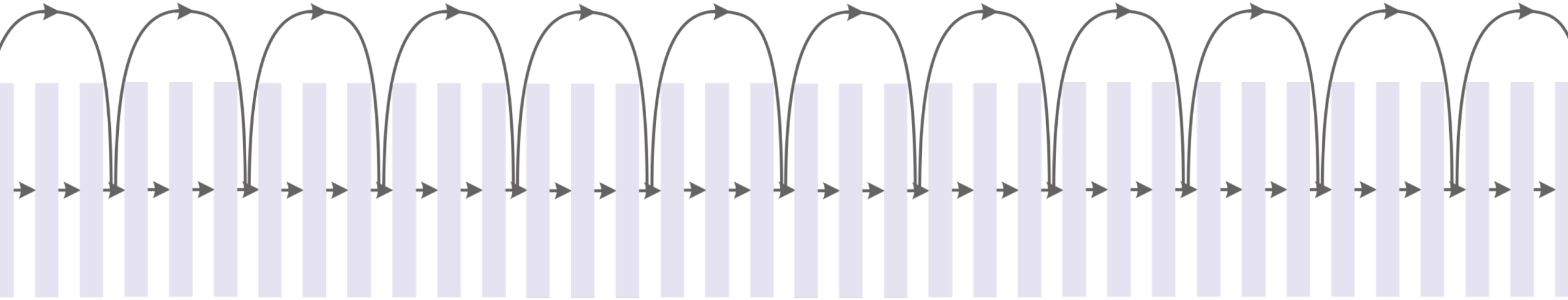
Lecture 10

Going Deep with Neural Networks

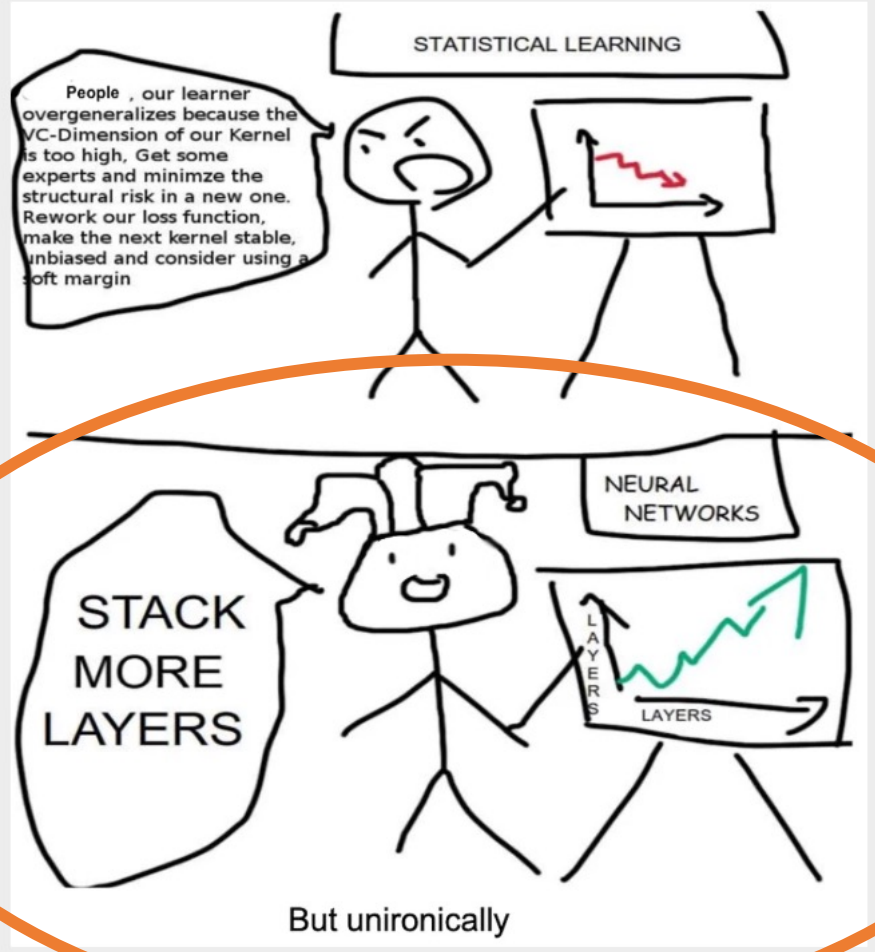


Sara Beery, Kaiming He, Mina Konaković Luković, Vincent Sitzmann

Overview

- Why going deep is difficult?
- Weight initialization
- Normalization modules
- Deep Residual Learning

User Can you explain why this is funny. Think about it step-by-step.

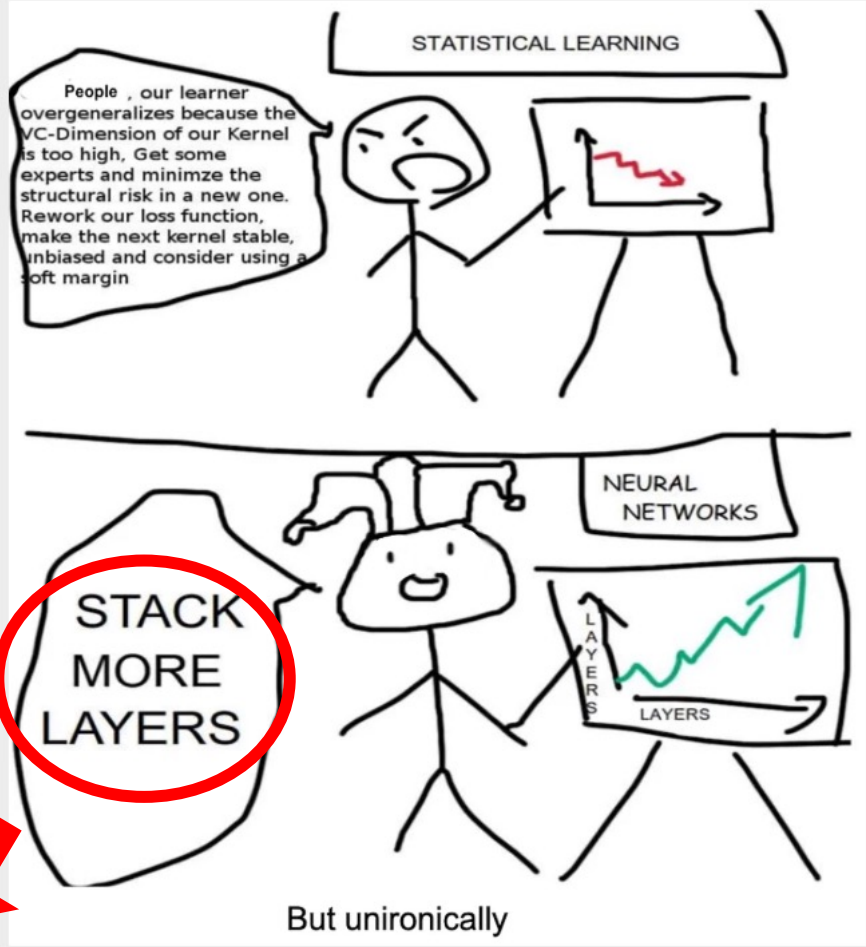


this lecture

GPT-4 The comic is satirizing the difference in approaches to improving model performance between statistical learning and neural networks.

source: https://openai.com/research/gpt-4

User Can you explain why this is funny. Think about it step-by-step.



GPT-4 The comic is satirizing the difference in approaches to improving model performance between statistical learning and neural networks.

source: https://openai.com/research/gpt-4

Why going deep is difficult?

- Troubles accumulate w/ more layers
- Signals get distorted when propagated
- in forward and backward passes

Why going deep is difficult?

All about signal propagation

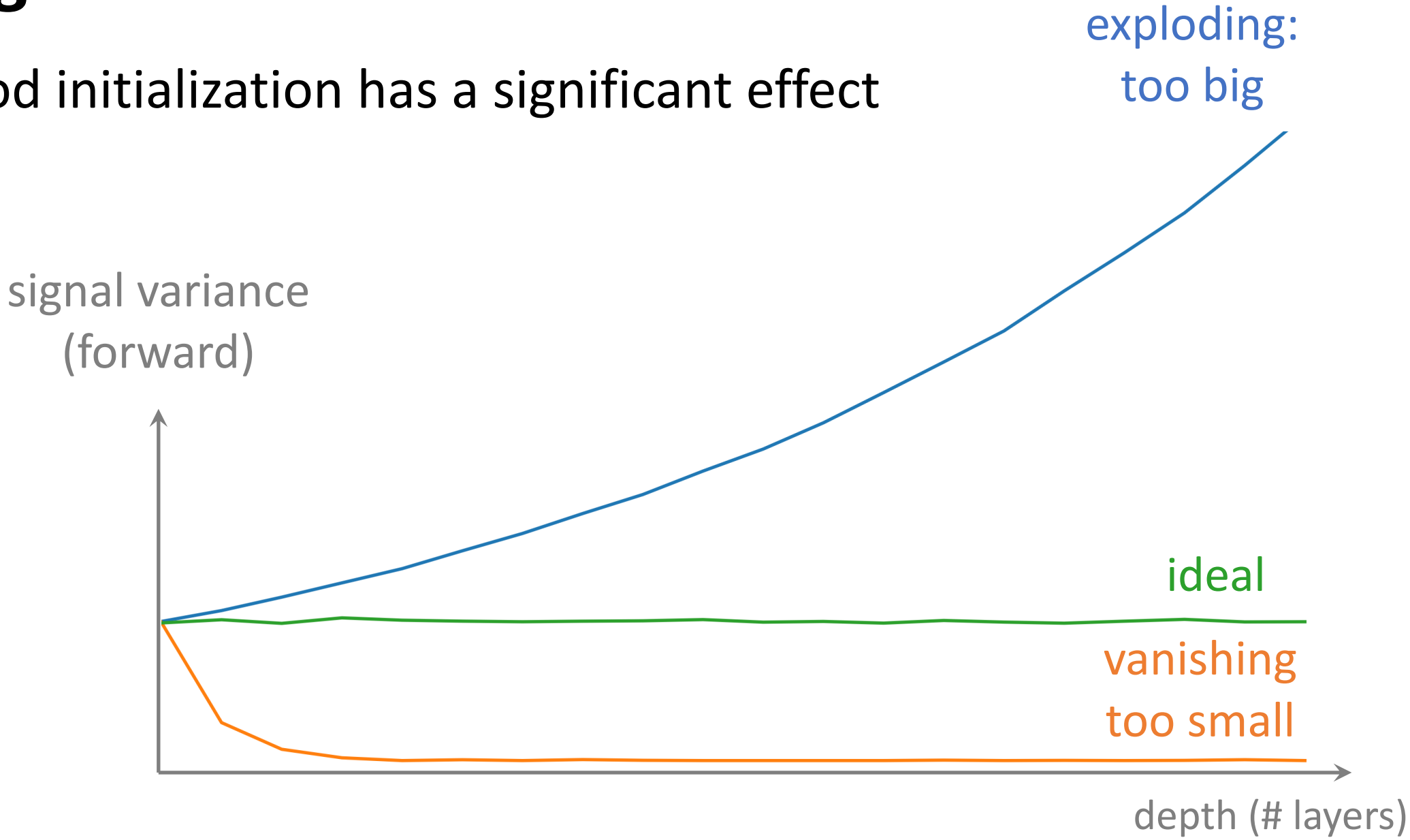
- ReLU activation
- Weight initialization
- Normalization modules
- Deep Residual Learning

} this lecture

Weight Initialization

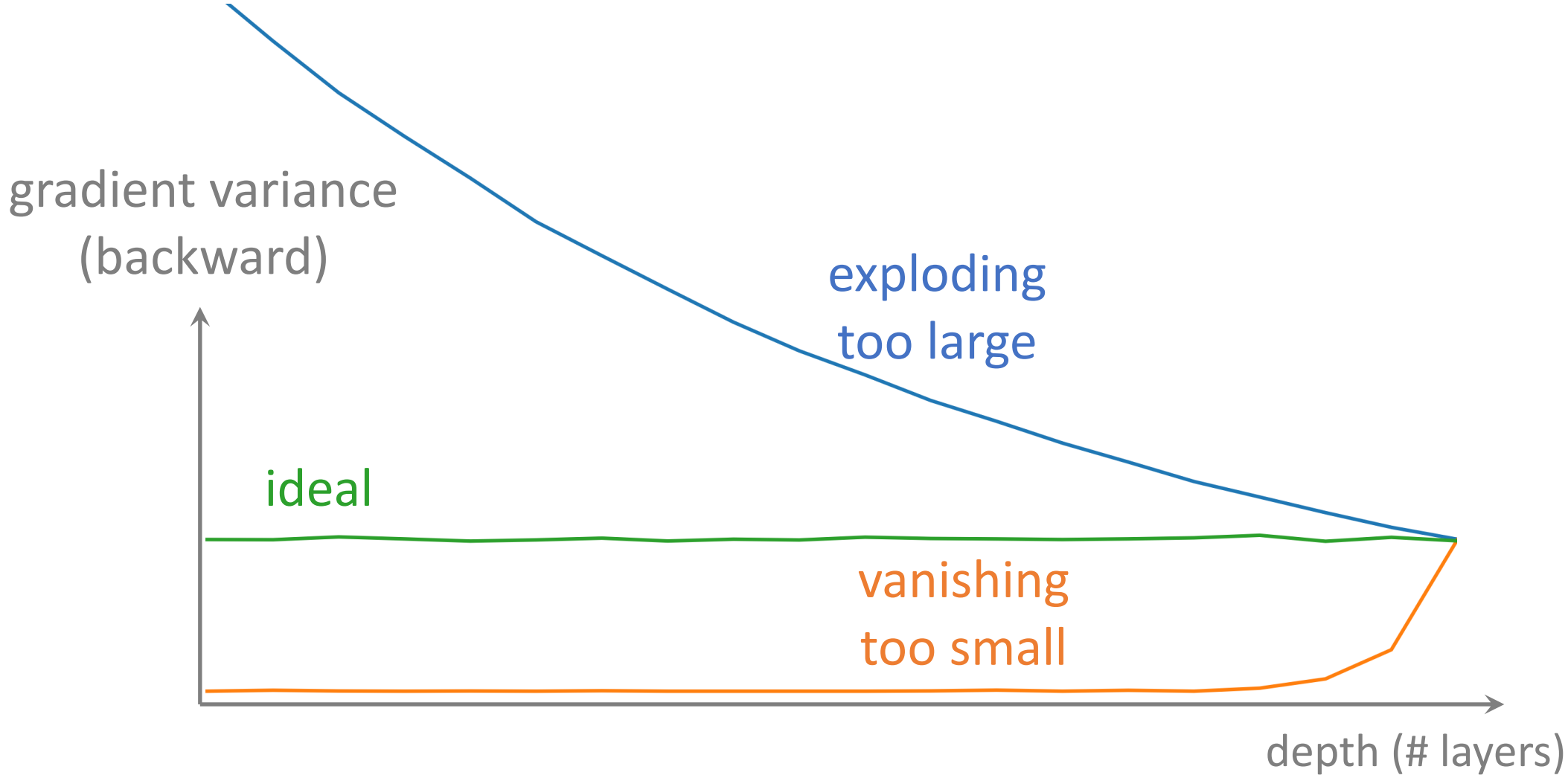
Weight initialization

- Good initialization has a significant effect



Weight initialization

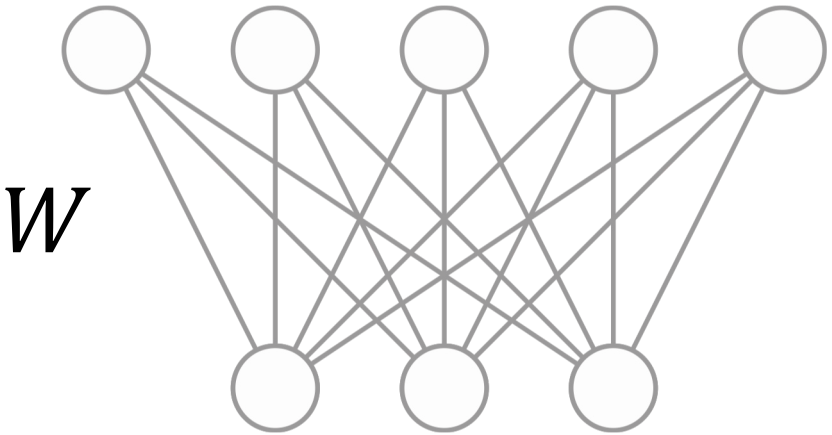
- Good initialization has a significant effect



Weight initialization

$\boldsymbol{x}$

(n neurons)



$\boldsymbol{y} = W\boldsymbol{x}$

(n' neurons)

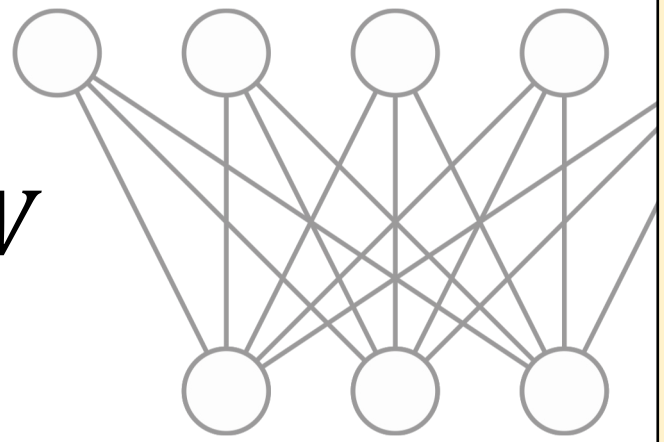
- if linear activation
- one layer: variance scaled by

$$\mathrm{Var}[y] = n\mathrm{Var}[w]\mathrm{Var}[x]$$

Weight initiali

$\boldsymbol{x}$

(n neurons)



W

$\boldsymbol{y} = W\boldsymbol{x}$

(n' neurons)

derivation

$$y_i = \sum_j W_{ij} x_j$$

- definition

$$\mathrm{Var}[y_i] = \mathrm{Var}[\sum_j W_{ij} x_j]$$

$$\mathrm{Var}[y_i] = \sum_j \mathrm{Var}[W_{ij} x_j]$$

- independence

$$\mathrm{Var}[y_i] = \sum_j \mathrm{Var}[W_{ij}]\,\mathrm{Var}[x_j]$$

- independence & zero-mean

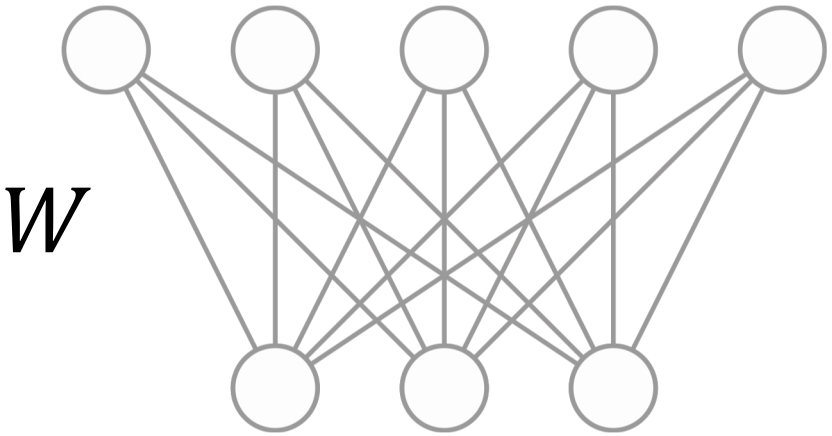
$$\mathrm{Var}[y] = n\mathrm{Var}[w]\mathrm{Var}[x]$$

- identical distributions

Weight initialization

$\boldsymbol{x}$

(n neurons)



$\boldsymbol{y} = W\boldsymbol{x}$

(n' neurons)

- if linear activation
- one layer: variance scaled by

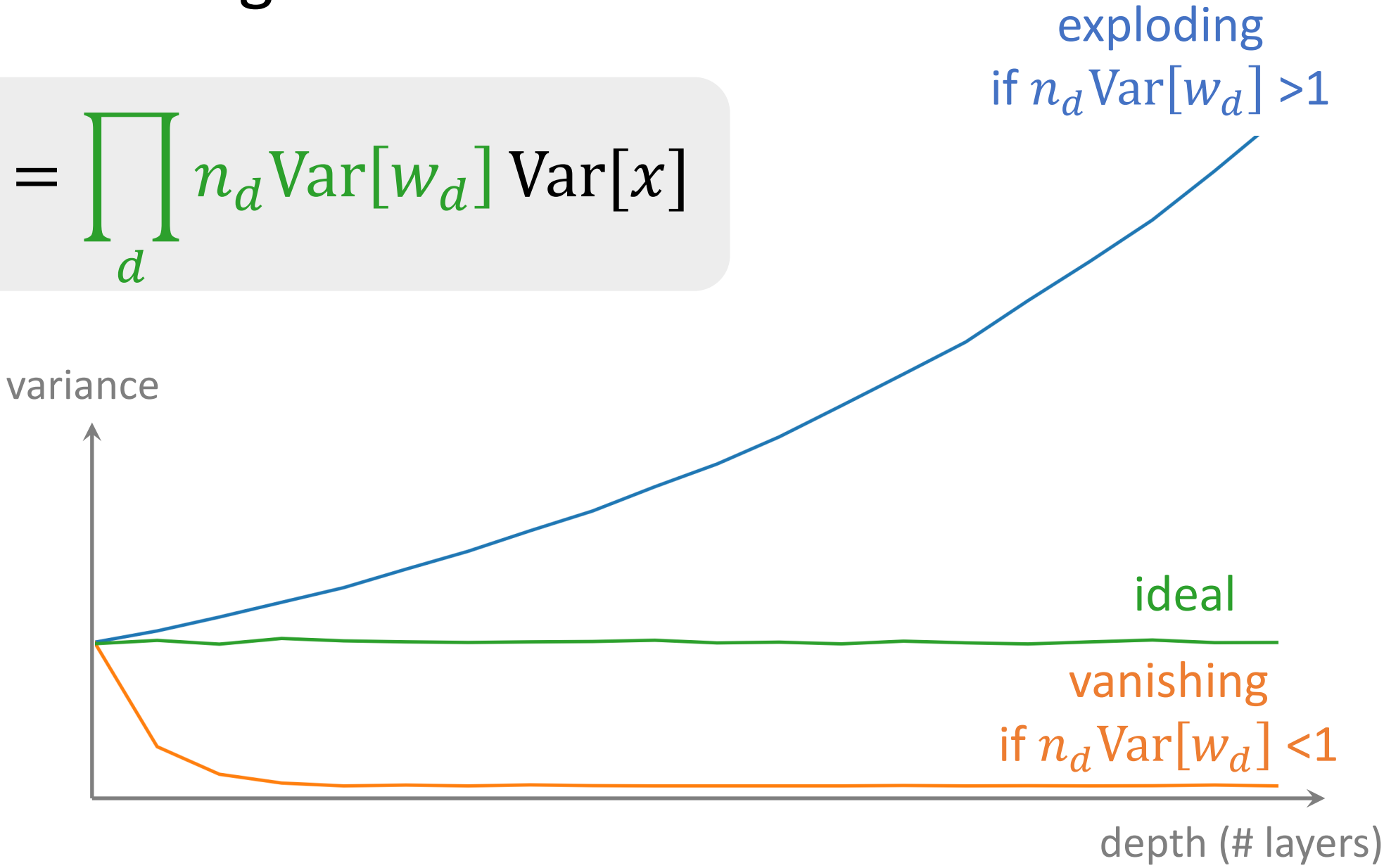
$$\mathrm{Var}[y] = n\mathrm{Var}[w]\mathrm{Var}[x]$$

- many layers: variance scaled by

$$\mathrm{Var}[y] = \prod_d n_d \mathrm{Var}[w_d]\,\mathrm{Var}[x]$$

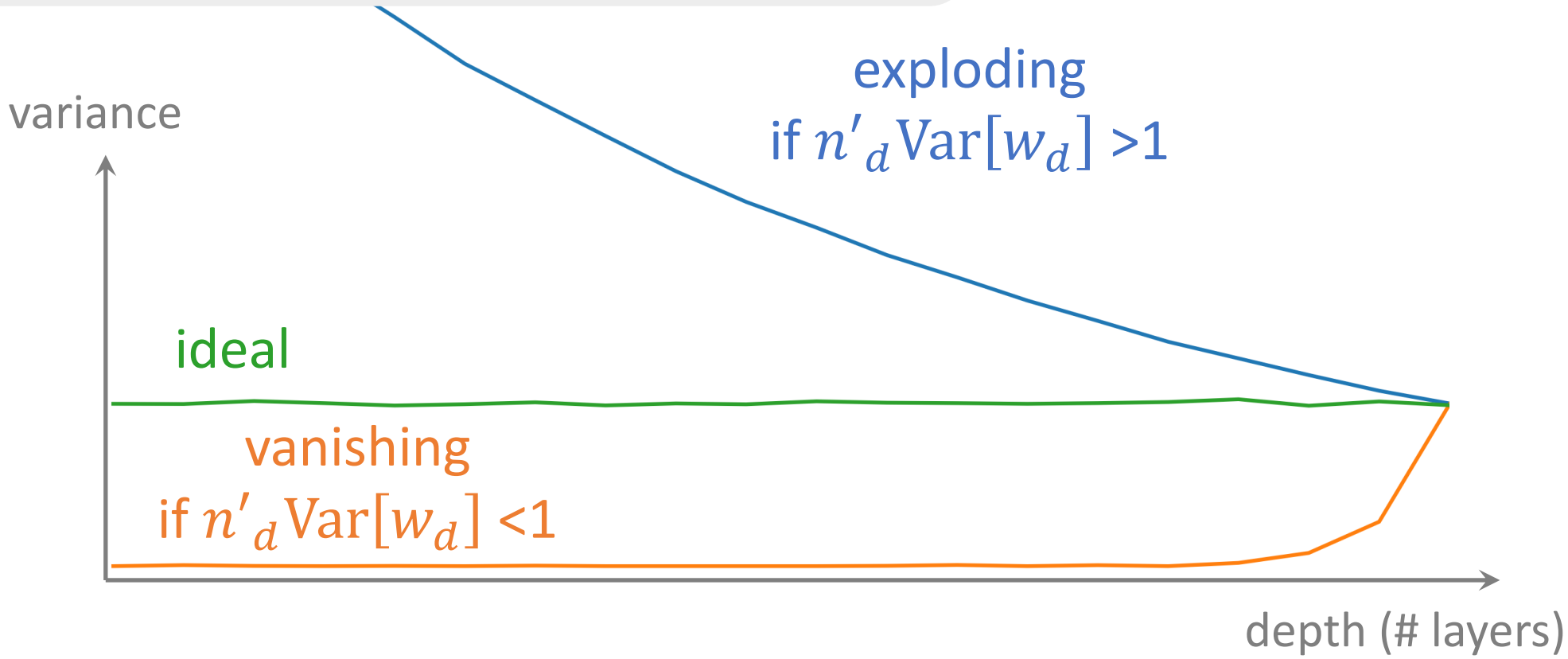
Variance Scaling: forward

$$\mathrm{Var}[y] = \prod_d n_d \mathrm{Var}[w_d]\, \mathrm{Var}[x]$$



Variance Scaling: backward

$$\mathrm{Var}\left[\frac{\partial \mathcal{E}}{\partial x}\right] = \prod_d n'_d \mathrm{Var}[w_d]\ \mathrm{Var}\left[\frac{\partial \mathcal{E}}{\partial y}\right]$$



Xavier initialization: torch.nn.init.xavier_normal_

forward:

$$n\mathrm{Var}[w] = 1$$

backward:

$$n'\mathrm{Var}[w] = 1$$

- Gaussian distribution:

$$w \sim \mathcal{N}(\mu = 0, \sigma = \sqrt{1/n})$$

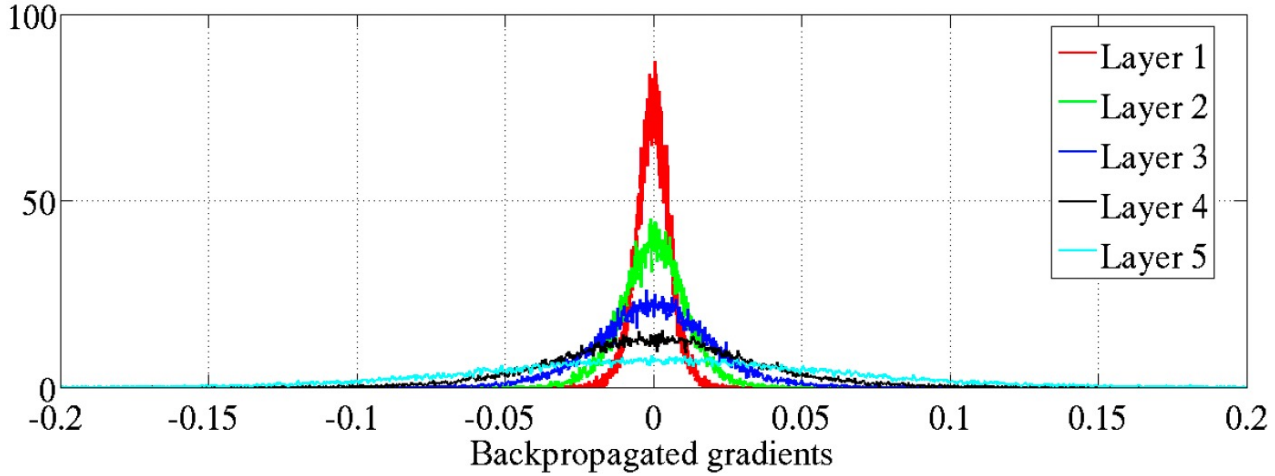
- Uniform distribution:

$$w \sim \mathcal{U}(-a, +a), a = \sqrt{3/n}$$

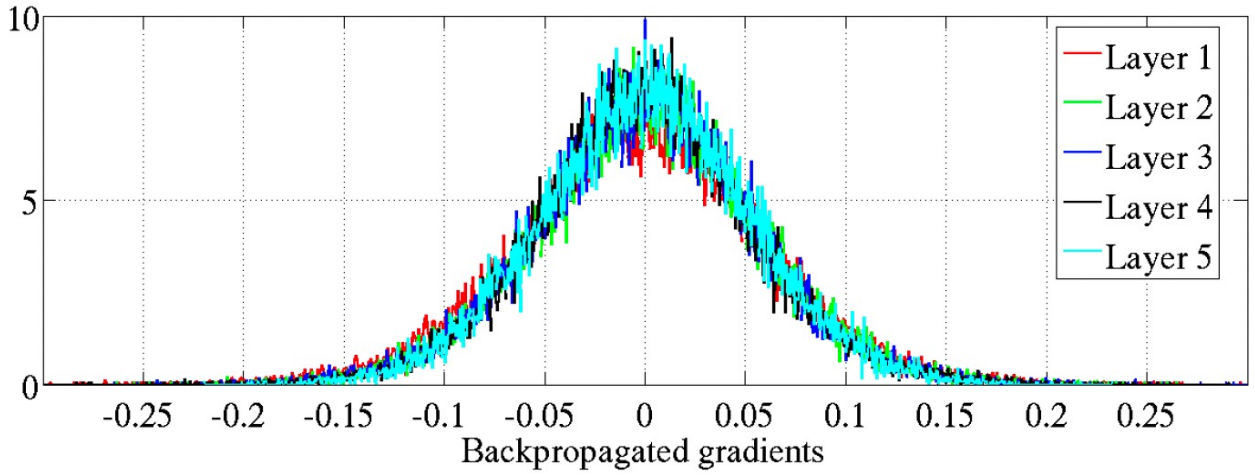
- Consider forward and backward:
  replace n with $(n + n')/2$

LeCun et al 1998 “Efficient Backprop”
Glorot & Bengio 2010 “Understanding the difficulty of training deep feedforward neural networks”

Xavier initialization: torch.nn.init.xavier_normal_



poor initialization:
earlier layer has smaller gradients



Xavier initialization:
all layers have similar gradient scale

Glorot & Bengio 2010 “Understanding the difficulty of training deep feedforward neural networks”

What are n and n' in ConvNet?

The number of connections feeding into/out of a node:

n ("fan_in"): $C_{in} \times K_h \times K_w$

n' ("fan_out"): $C_{out} \times K_h \times K_w$

$K_h \times K_w$: kernel size

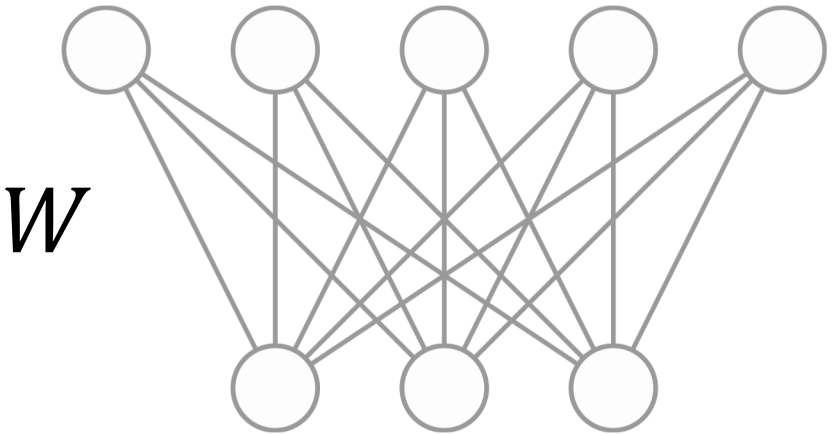
fully-connected: viewed as H×W conv

LeCun et al. "Efficient BackProp". 1998.

Weight initialization: ReLU

$\boldsymbol{x}' = \mathrm{ReLU}(\boldsymbol{x})$

(n neurons)



$\boldsymbol{y} = W\boldsymbol{x}'$

(n' neurons)

- if ReLU activation
- one layer: variance scaled by

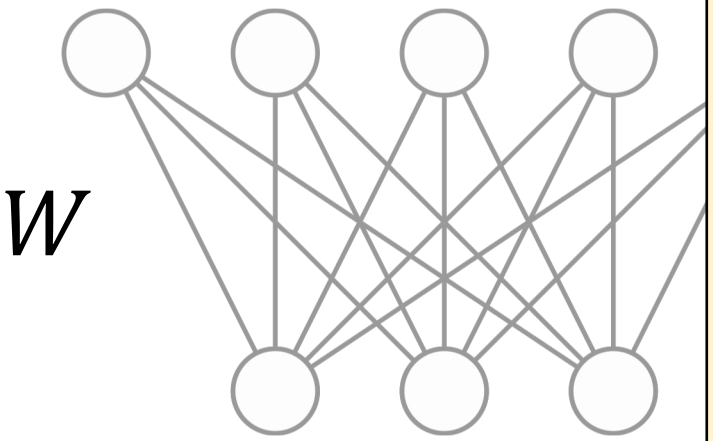
$$\mathrm{Var}[y] = \frac{1}{2} n \mathrm{Var}[w] \mathrm{Var}[x]$$

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. ICCV 2015.

Weight initiali

$\boldsymbol{x'} = \mathrm{ReLU}(\boldsymbol{x})$

(n neurons)



W

$\boldsymbol{y} = W\boldsymbol{x'}$

(n' neurons)

derivation

$$y_i = \sum_j W_{ij} x'_j$$

- $x' = \mathrm{ReLU}(x)$

$$\mathrm{Var}[y_i] = \mathrm{Var}[\sum_j W_{ij} x'_j]$$

$$\mathrm{Var}[y_i] = \sum_j \mathrm{Var}[W_{ij} x'_j]$$

$$\mathrm{Var}[y_i] = \sum_j \mathrm{Var}[W_{ij}]\, \mathrm{E}[x'^2_j]$$

$\mathrm{Var}[wx']$
$= E[(wx')^2] - (E[wx'])^2$
$= E[(wx')^2] - (E[w]E[x])^2$
$= E[w^2]E[x'^2] = \mathrm{V}ar[w]E[x'^2]$
if w is zero-mean but x' is not.

$$\mathrm{Var}[y] = n\mathrm{Var}[w]\frac{1}{2}\mathrm{Var}[x]$$

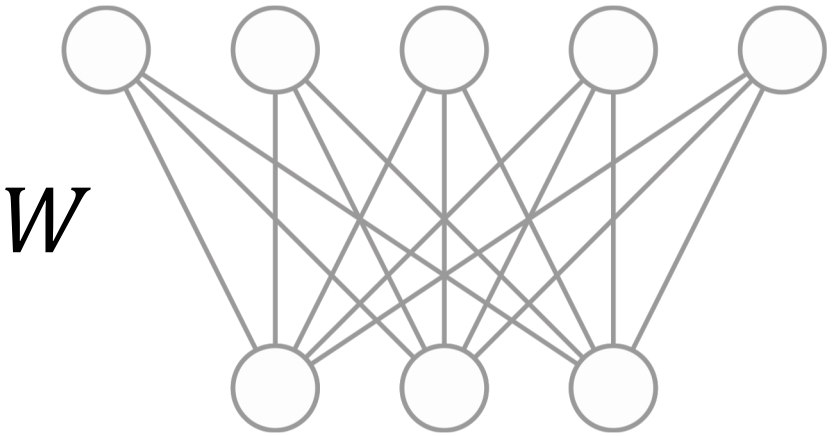
$\mathrm{E}[x'^2] = \mathrm{Var}[x]/2$
if x is zero-mean and symmetric

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. ICCV 2015.

Weight initialization: ReLU

$\boldsymbol{x}' = \mathrm{ReLU}(\boldsymbol{x})$

(n neurons)



$\boldsymbol{y} = W\boldsymbol{x}'$

(n' neurons)

- if ReLU activation
- one layer: variance scaled by

$$\mathrm{Var}[y] = \frac{1}{2} n \mathrm{Var}[w] \mathrm{Var}[x]$$

- many layers: variance scaled by

$$\mathrm{Var}[y] = \prod_d \frac{1}{2} n_d \mathrm{Var}[w_d] \, \mathrm{Var}[x]$$

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. ICCV 2015.

Kaiming initialization: torch.nn.init.kaiming_normal_

forward:

$$\frac{1}{2} n \mathrm{Var}[w] = 1$$

backward:

$$\frac{1}{2} n' \mathrm{Var}[w] = 1$$

- Gaussian distribution:

$$w \sim \mathcal{N}(\mu = 0, \sigma = \sqrt{2/n})$$

- Uniform distribution:

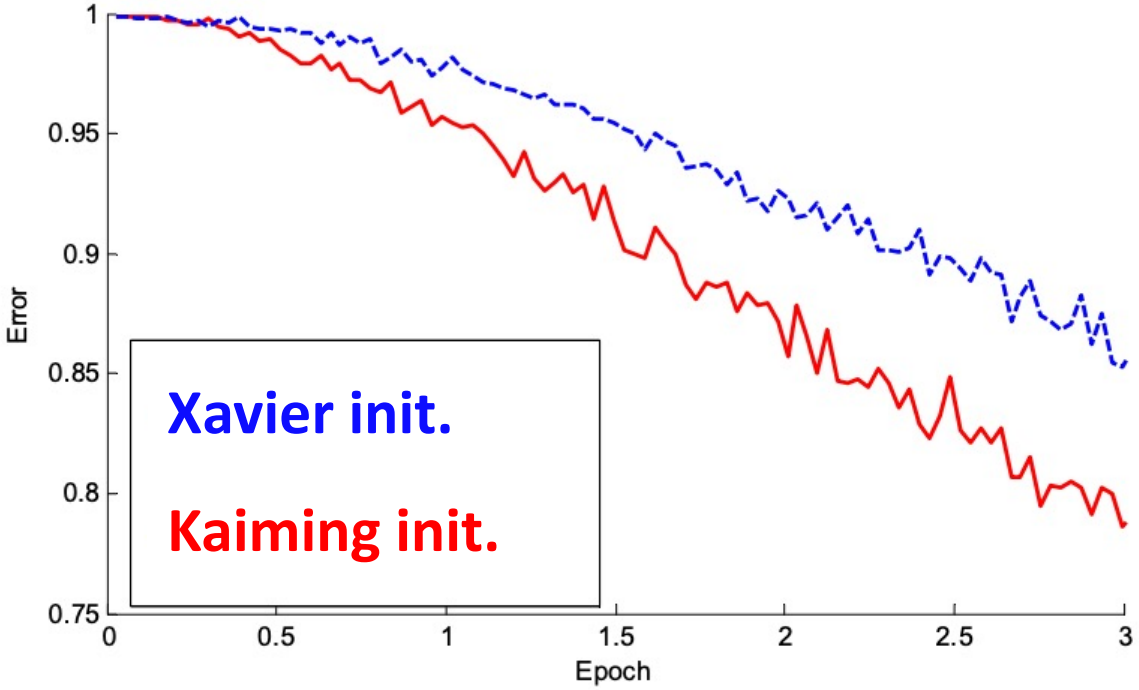
$$w \sim \mathcal{U}(-a, +a), a = \sqrt{6/n}$$

- sufficient to use n or n'

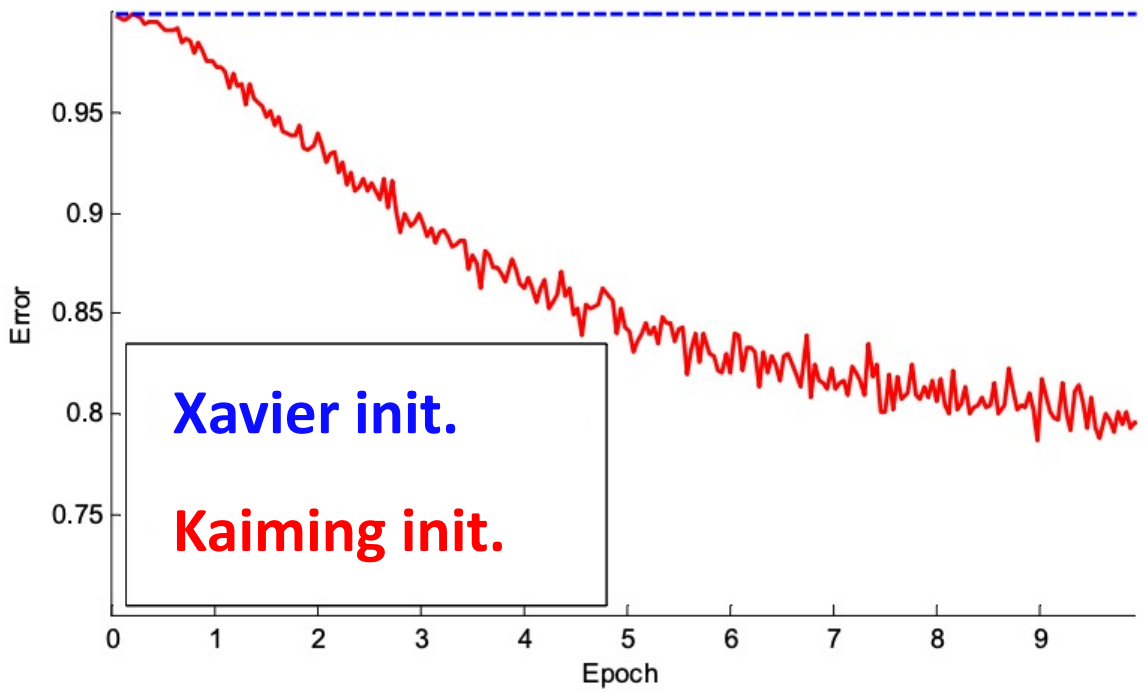
LeCun et al 1998 “Efficient Backprop”

Glorot & Bengio 2010 “Understanding the difficulty of training deep feedforward neural networks”

Kaiming initialization: torch.nn.init.kaiming_normal_



22-layer VGG w/ ReLU :
better init converges faster



30-layer VGG w/ ReLU:
better init enables training

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. ICCV 2015.

Weight initialization: Takeaways

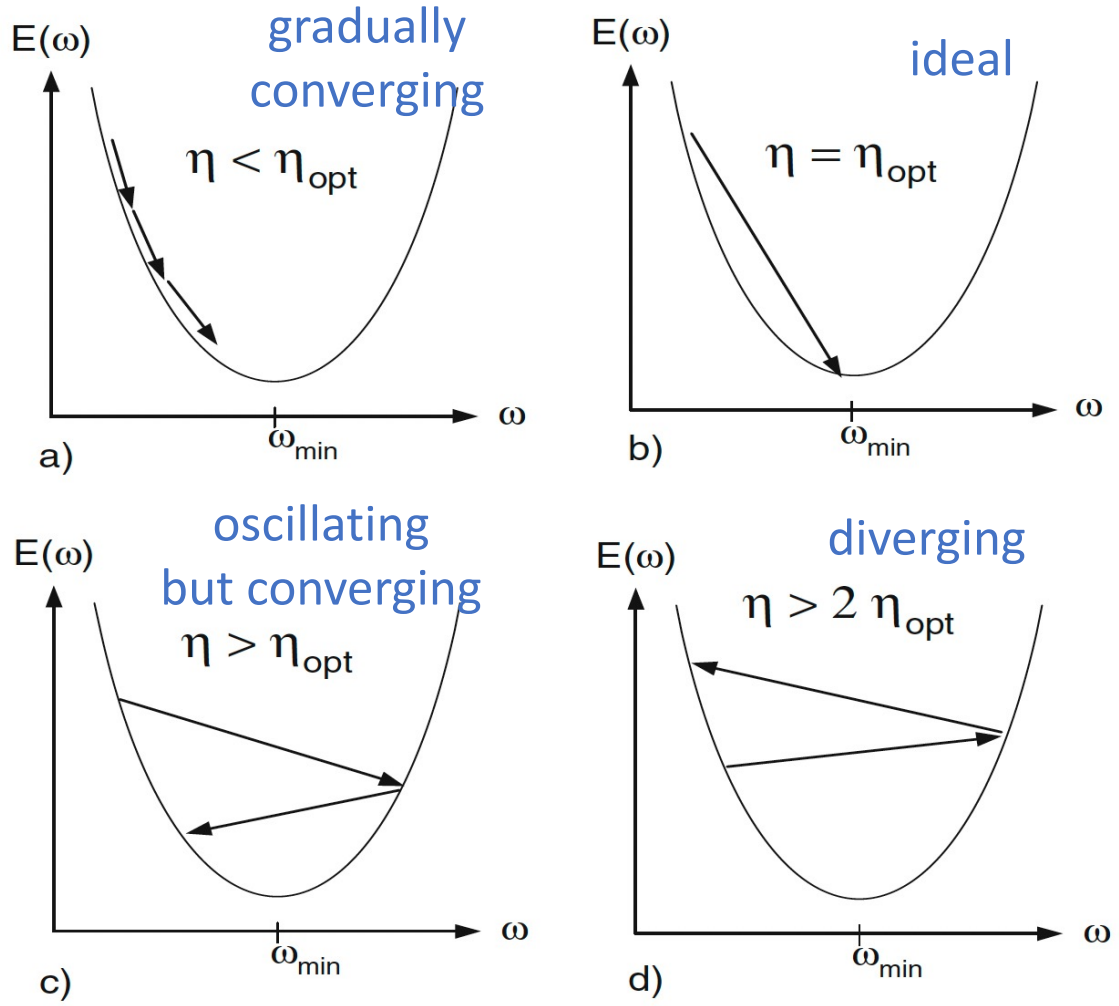
- Signals can be exploding/vanishing
- Product of scaling factors
- Initialization is driven by normalization
- Initialization is based on strong assumptions
- Initialization is necessary, but often not sufficient

Normalization Modules

Learning Rate in Gradient Descent

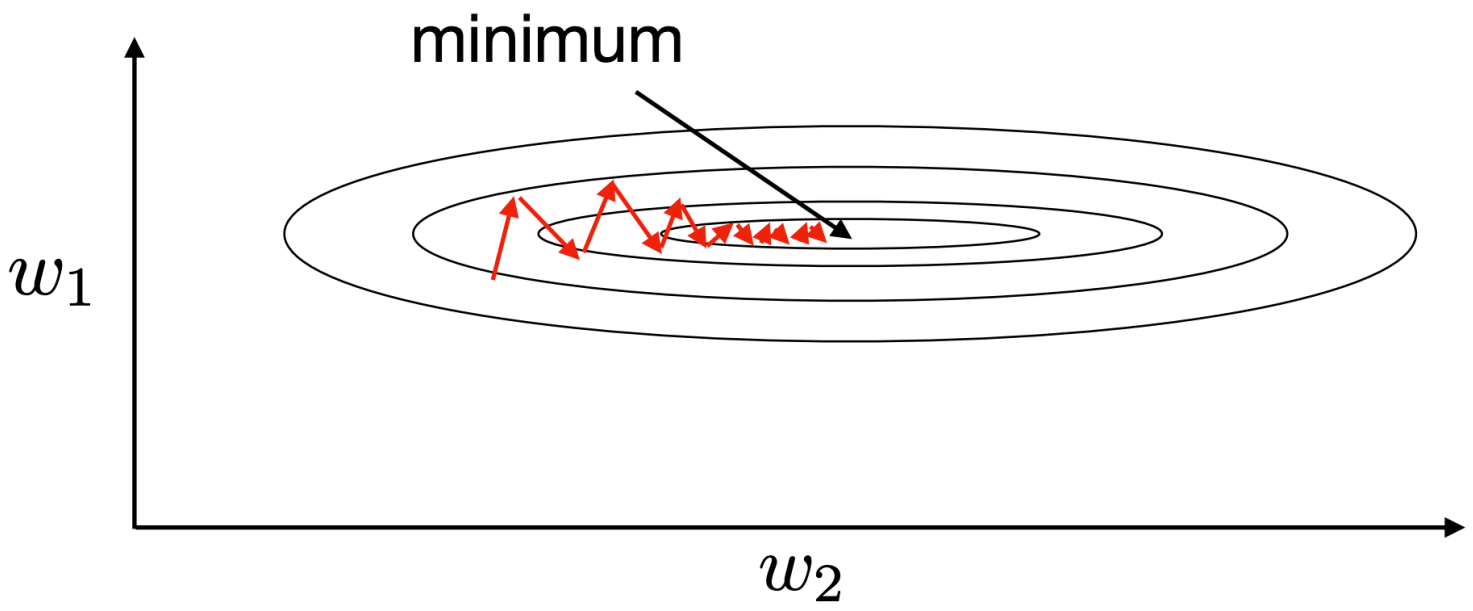
$$W := W - \eta \frac{\partial \mathcal{E}}{\partial W}$$

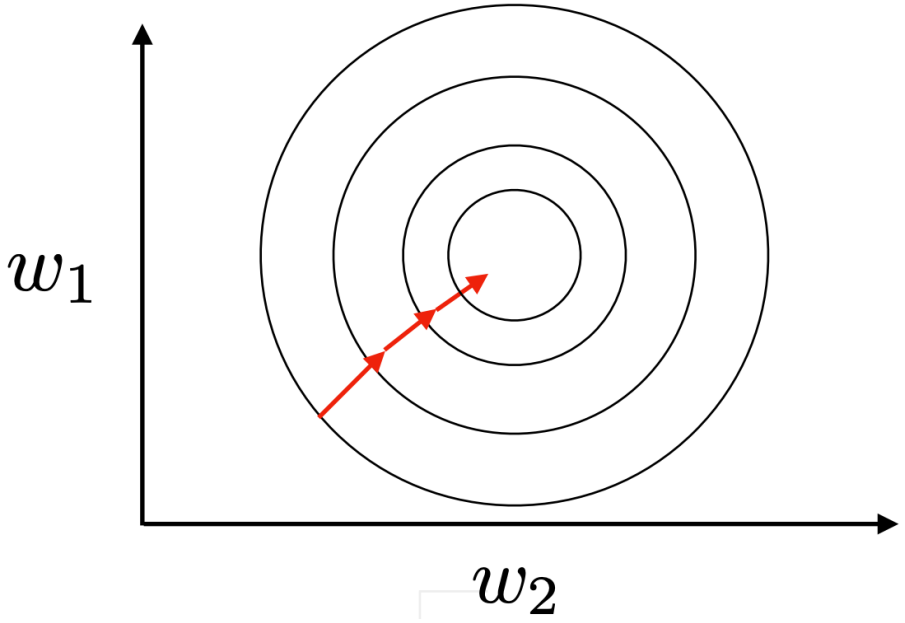
η: learning rate



LeCun et al 1998 “Efficient Backprop”

Normalization





- Same learning rate applied to all weights
- Large weights dominate updates
- Small weights oscillate (or diverge)

- Similar pace for all weights

Figures adapted from Sebastian Raschka: L11.1 Input Normalization

Input Normalization

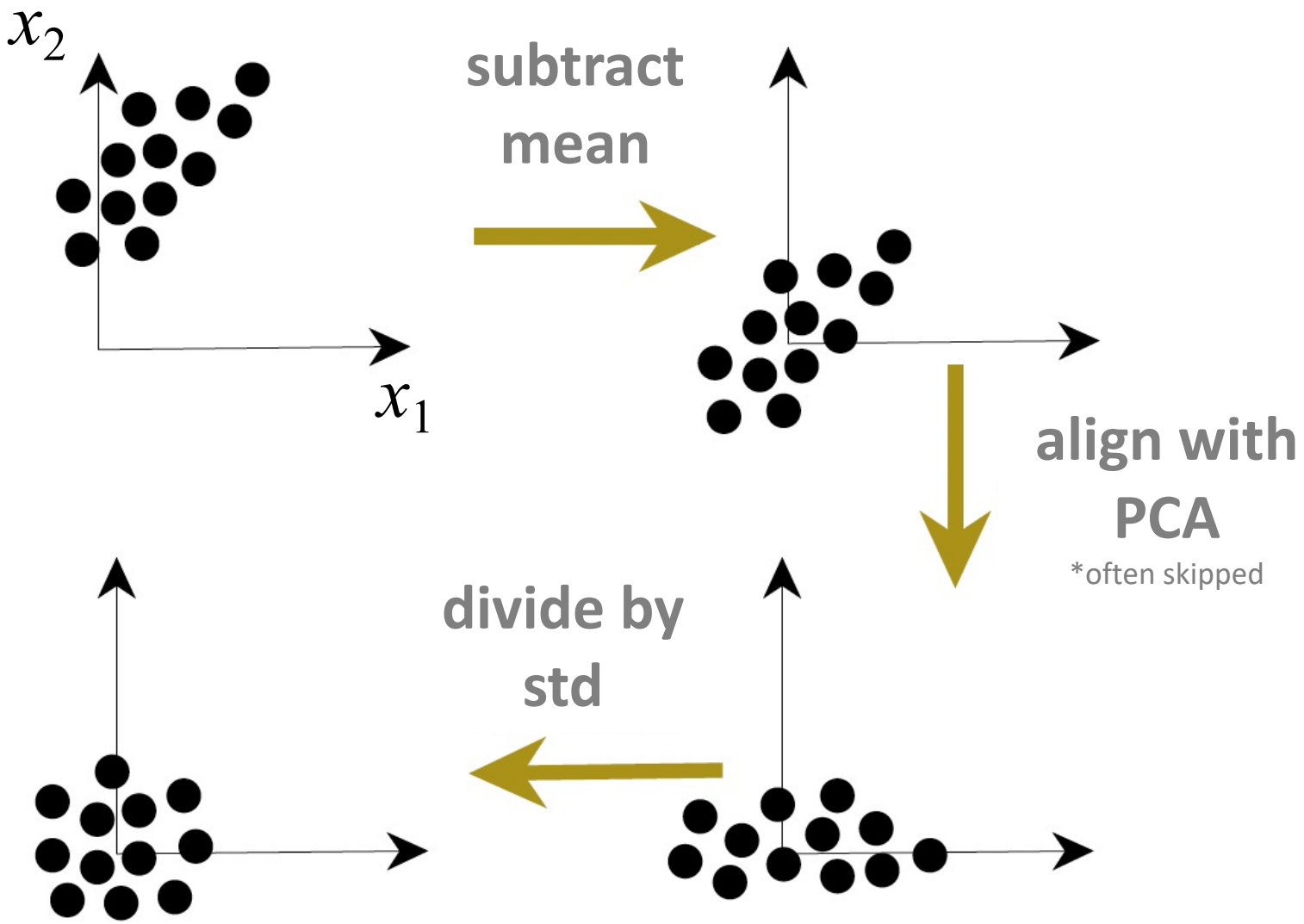
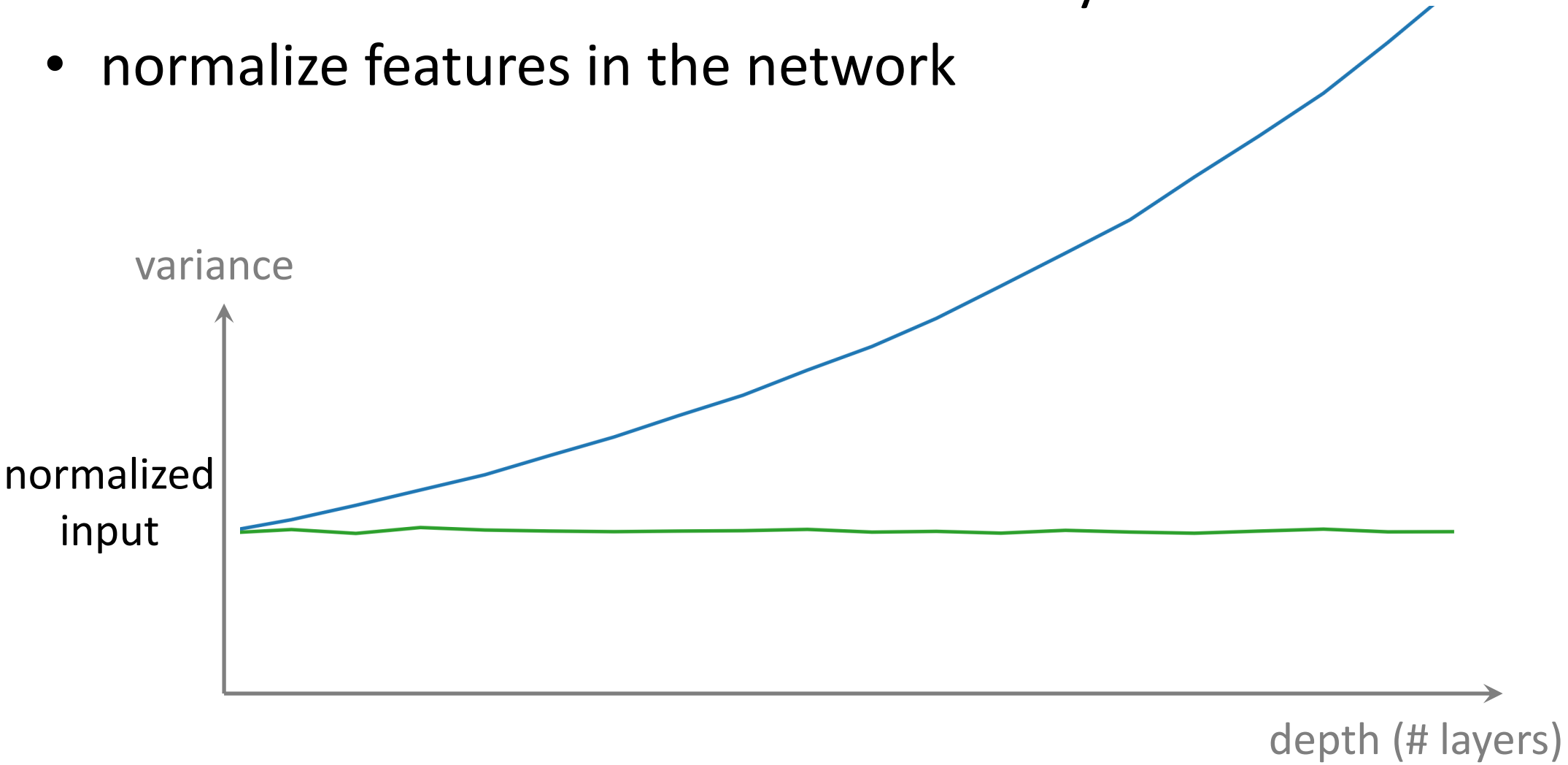


Figure adapted from: LeCun et al. “Efficient BackProp”. 1998.

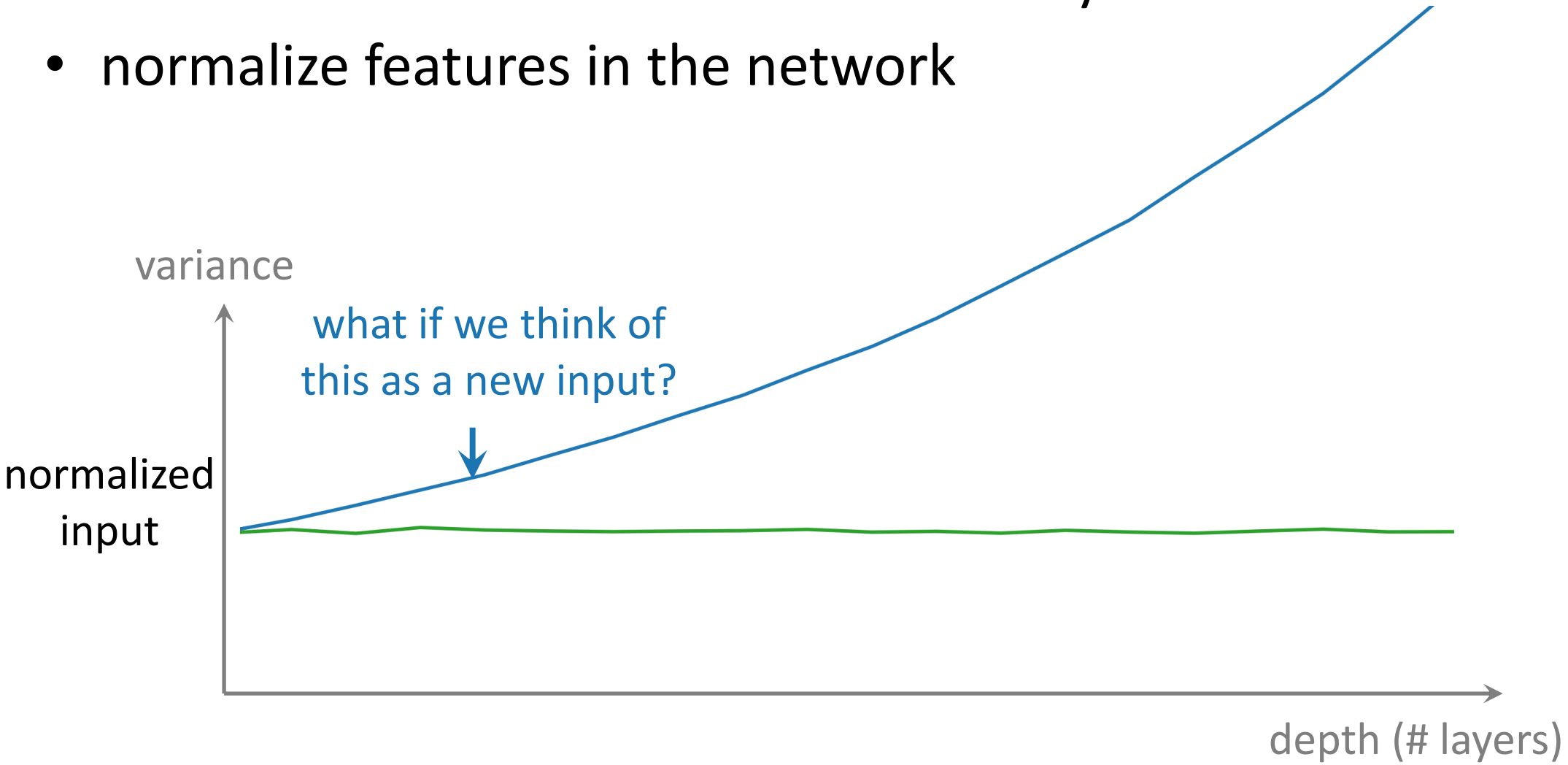
Normalization Modules

- We want to maintain variance for all layers
- normalize features in the network



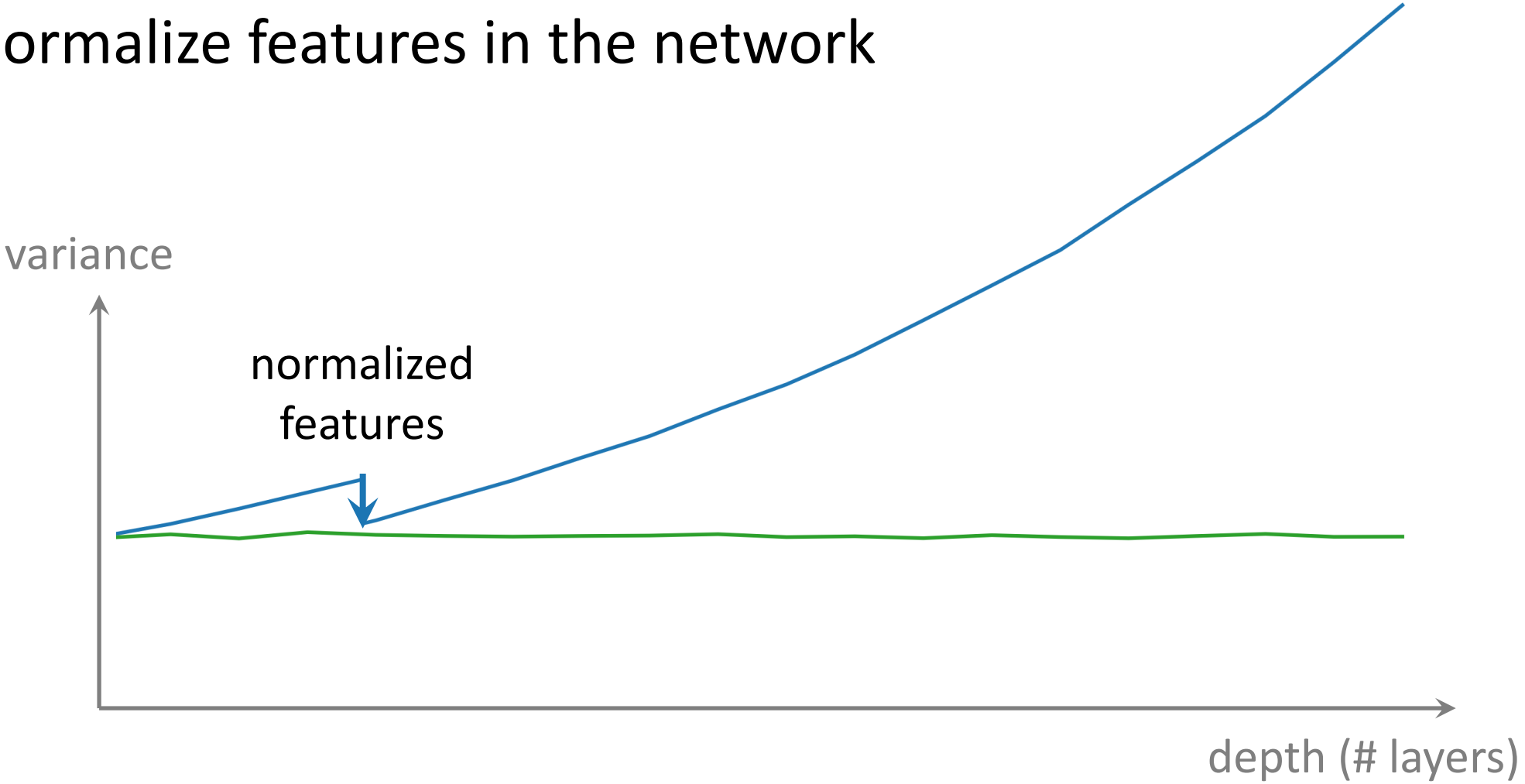
Normalization Modules

- We want to maintain variance for all layers
- normalize features in the network



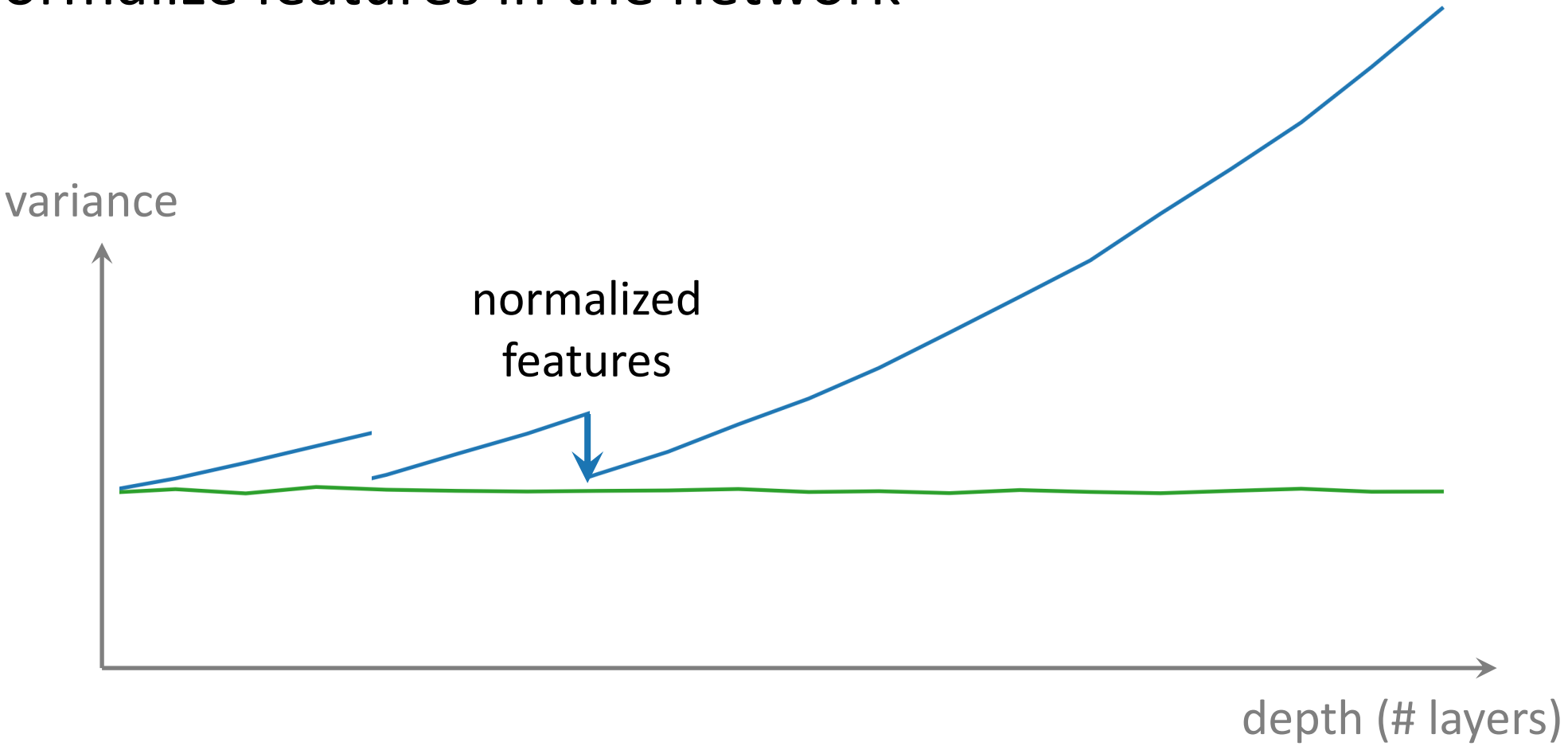
Normalization Modules

- We want to maintain variance for all layers
- normalize features in the network



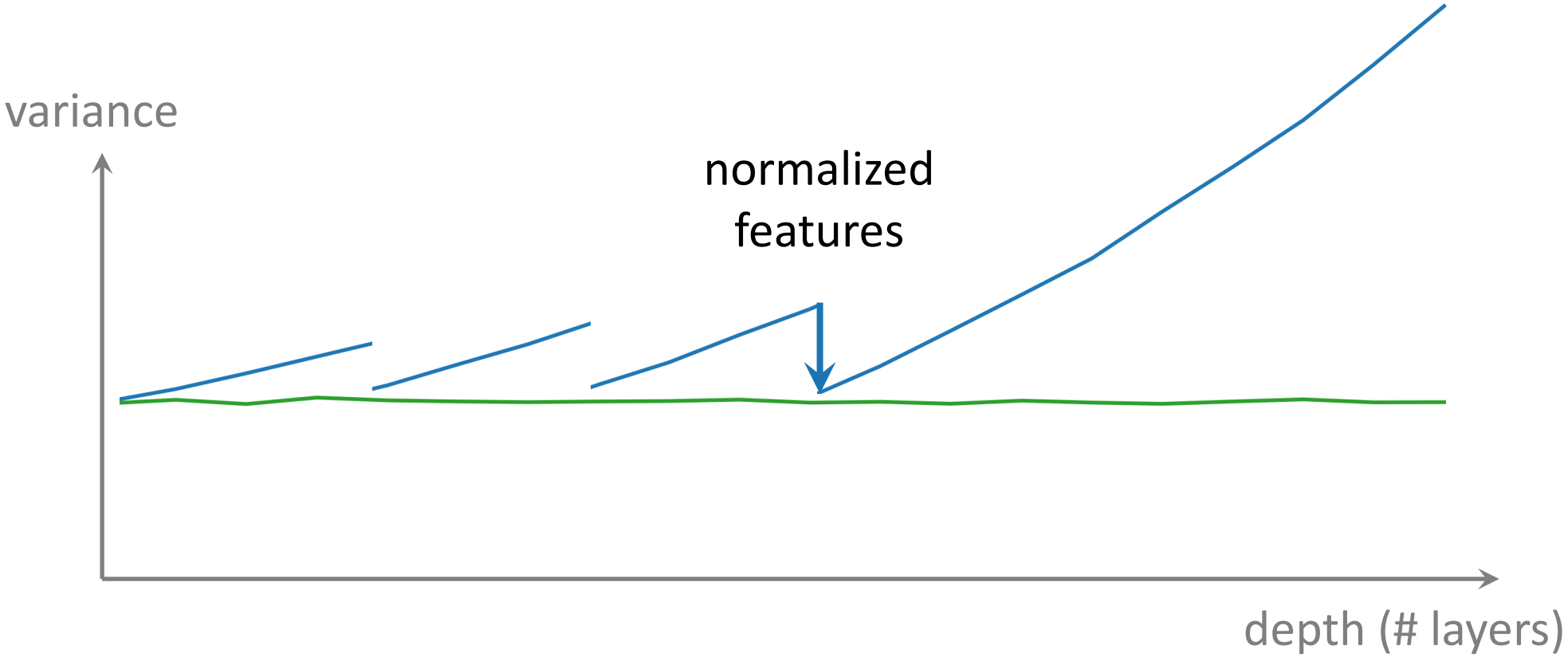
Normalization Modules

- We want to maintain variance for all layers
- normalize features in the network



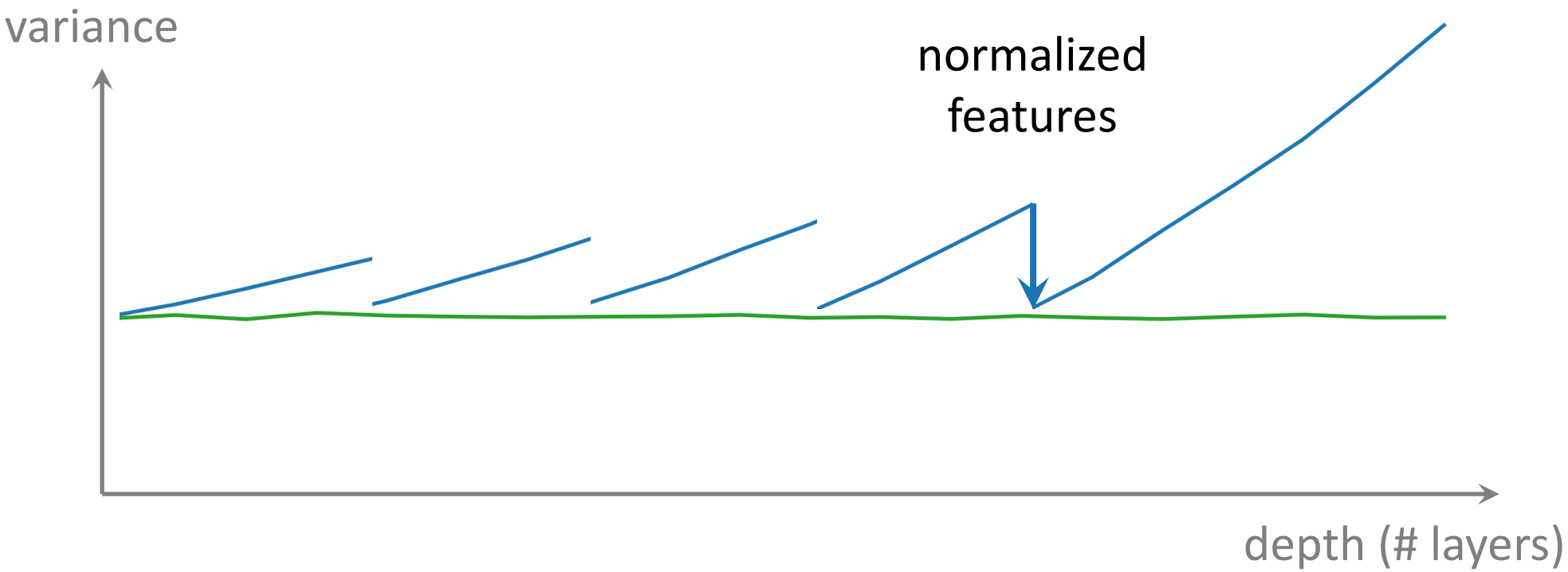
Normalization Modules

- We want to maintain variance for all layers
- normalize features in the network



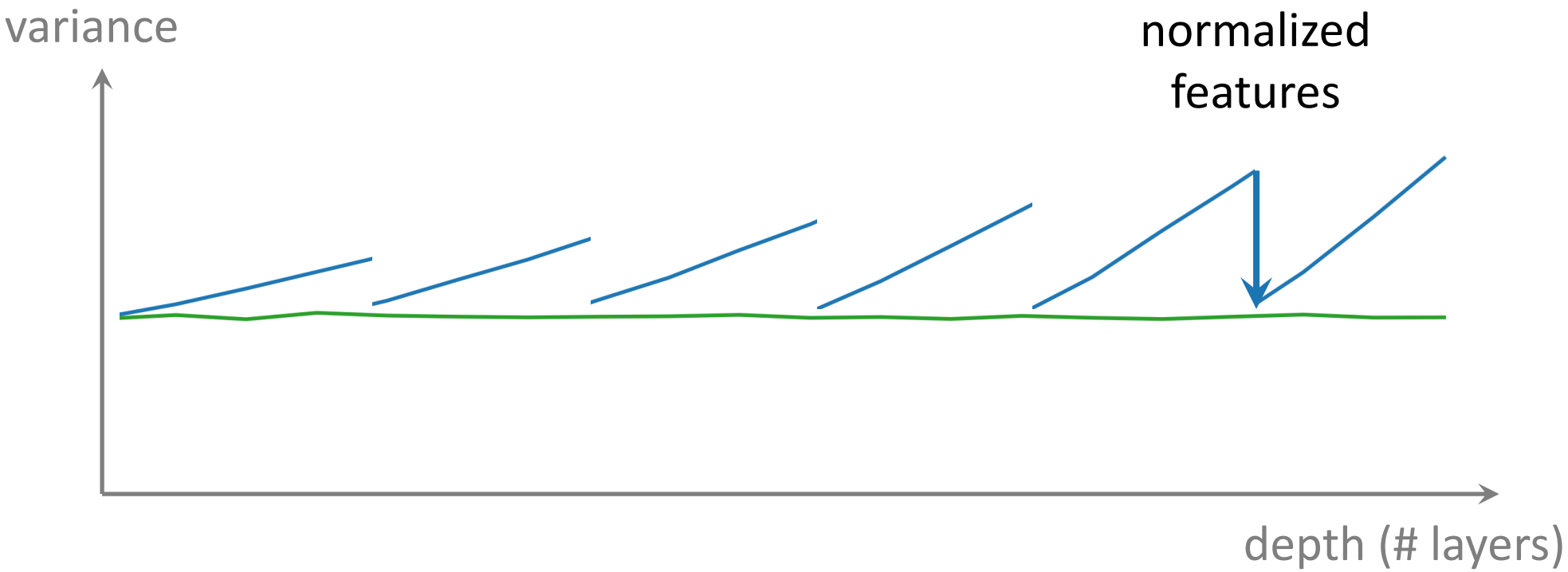
Normalization Modules

- We want to maintain variance for all layers
- normalize features in the network



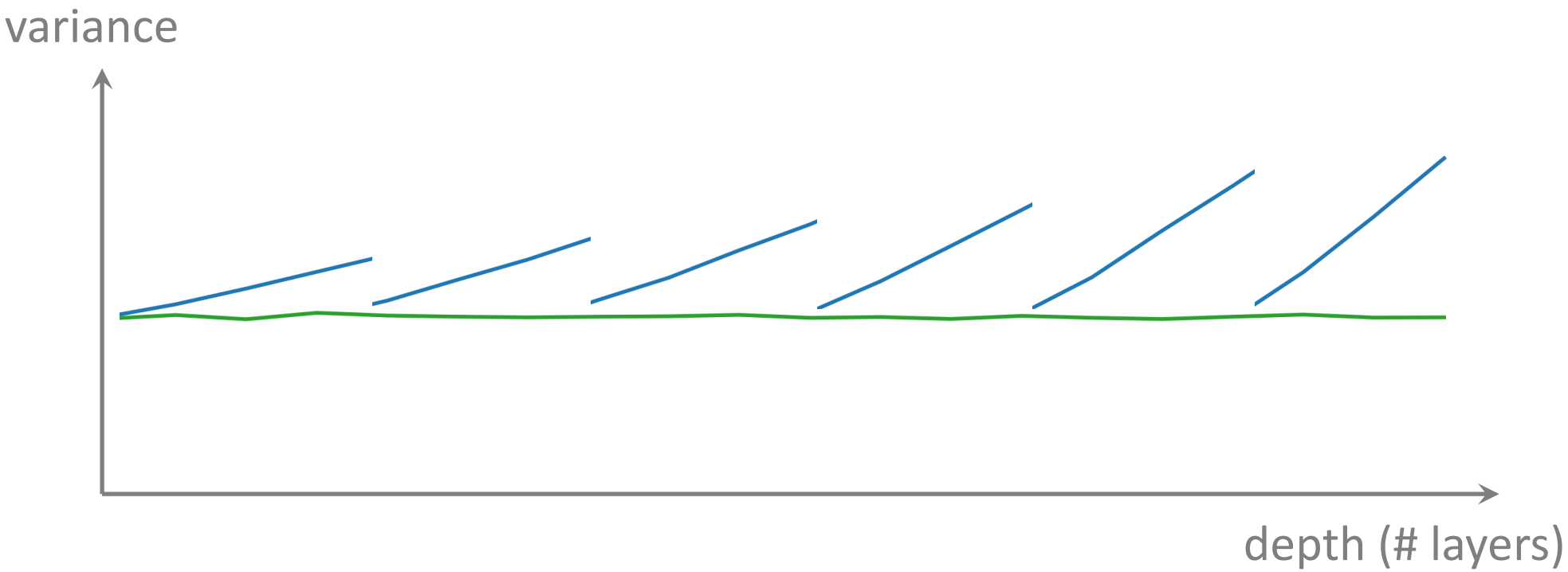
Normalization Modules

- We want to maintain variance for all layers
- normalize features in the network



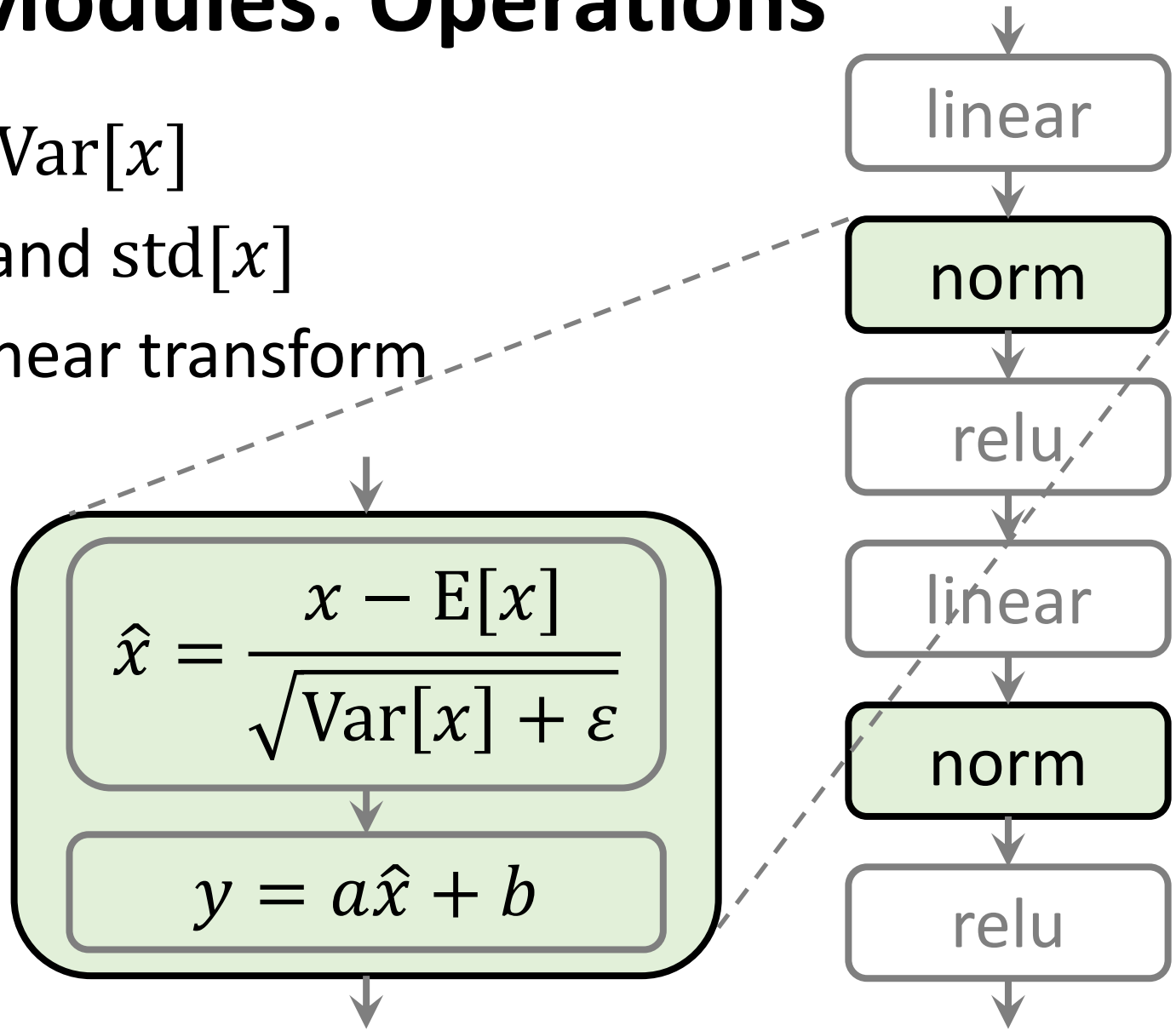
Normalization Modules

- We want to maintain variance for all layers
- normalize features in the network
- train end-to-end by BackProp



Normalization Modules: Operations

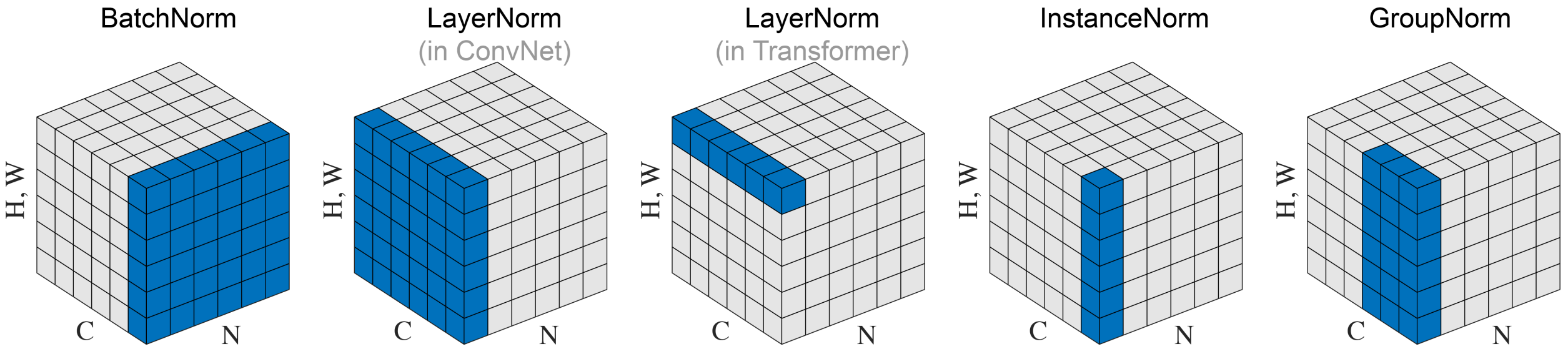
1. compute $\mathrm{E}[x]$ and $\mathrm{Var}[x]$
2. normalize by $\mathrm{E}[x]$ and $\mathrm{std}[x]$
3. compensate by a linear transform



Ioffe & Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. ICML 2015

Normalization Modules: Variants

differ in support sets of $\mathrm{E}[x], \mathrm{Var}[x]$

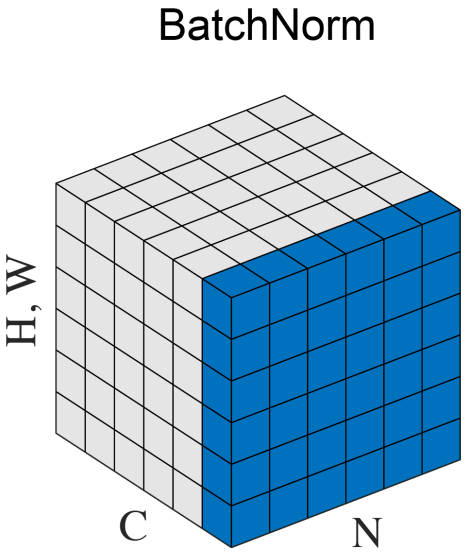


Ioffe & Szegedy, 2015; Ba, Kiros, Hinton, 2016; Ulyanov, Vedaldi, Lempitsky, 2016; Wu & He, 2018

Case Study: Batch Normalization (BN)

$\mathrm{E}[x]$ over N, H, W axes

Einstein sum: $\sum_{\mathrm{n}} \sum_{\mathrm{h}} \sum_{\mathrm{w}} x_{\mathrm{nchw}}$




```
mean = torch.einsum('nchw->c', x) / (N * H * W)
```

C-dim vector

BN at inference time

- batch (N axis) is not legitimate
- maintain μ and σ^2 in train set (moving average)
- use μ and σ^2 at inference time

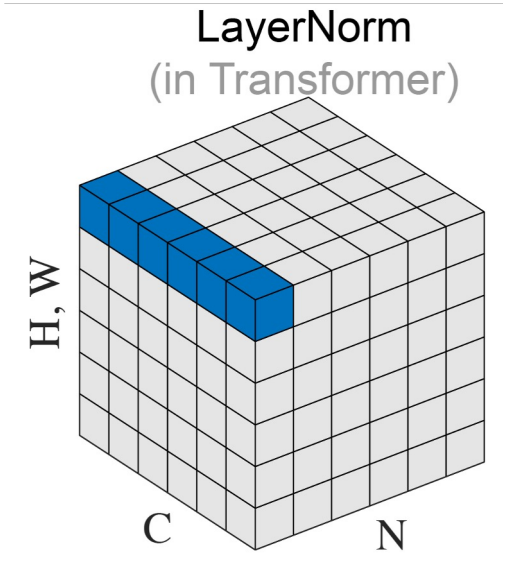
- tl;dr: set model.eval() in torch

Ioffe & Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". ICML 2015

Case Study: Layer Normalization (LN)

$\mathrm{E}[x]$ over C axis

Einstein sum: $\sum_{\mathrm{c}} x_{\mathrm{nchw}}$




```
mean = torch.einsum('nchw->nhw', x) / C
```

N×H×W tensor

- same operation at train and inference
- popular in Transformers and sequences

*More fun with Einstein sum:
https://rockt.github.io/2018/04/30/einsum

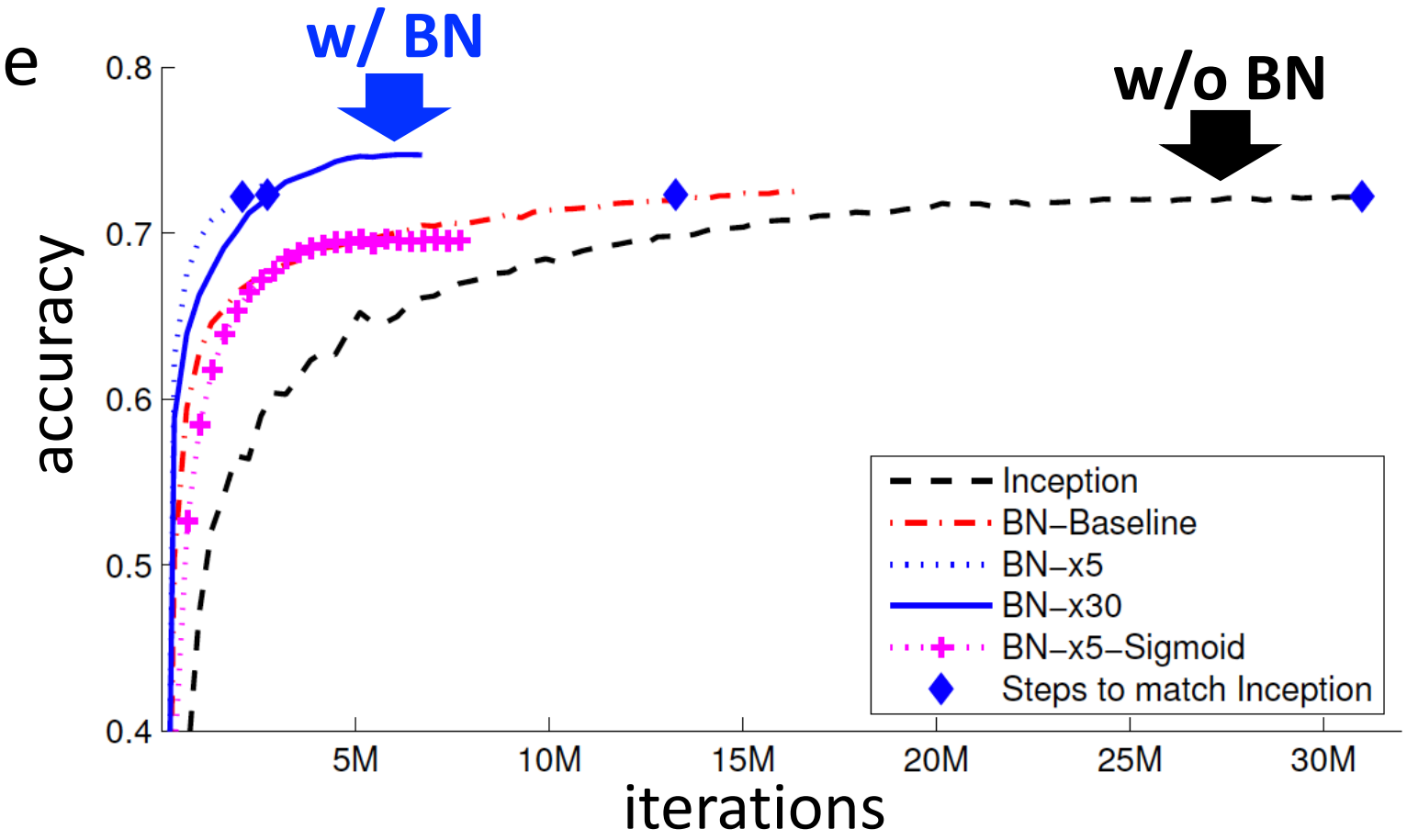
Ba, Kiros, Hinton. “Layer Normalization”. 2016

Normalization Modules: Effects

- **Enable training models that are otherwise not trainable**

- Speed up convergence

- Improve accuracy



Ioffe & Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". ICML 2015

Deep Residual Learning

Deep Residual Learning

- Deep Learning gets way deeper
- simple component: identity shortcut
- enable networks w/ hundreds of layers

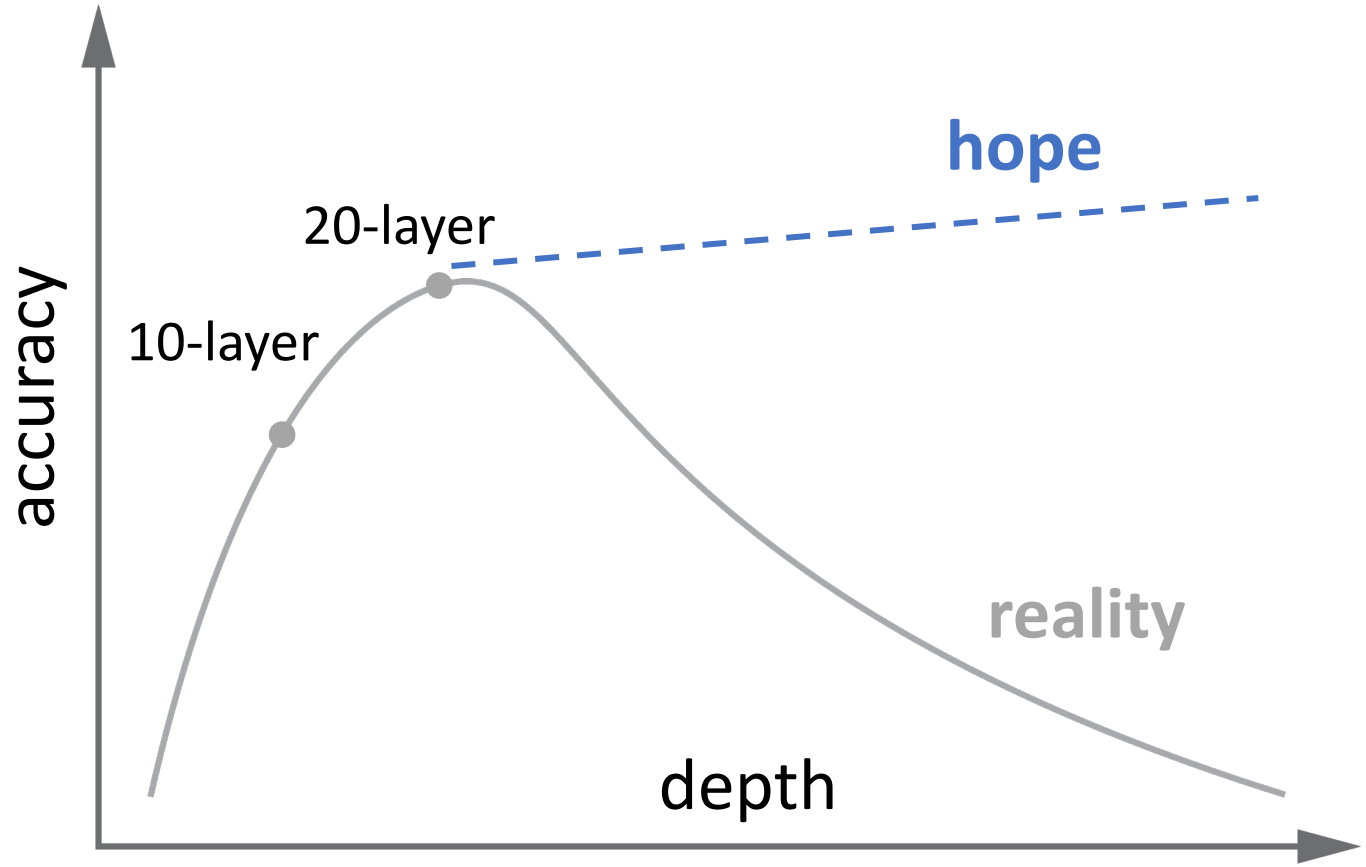
Compose simple modules into complex functions

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015, CVPR 2016.

The degradation problem

- Good init + norm enable training deeper models
- Simply stacking more layers?

- Degrade after ~20 layers
- Not overfitting
- Difficult to train

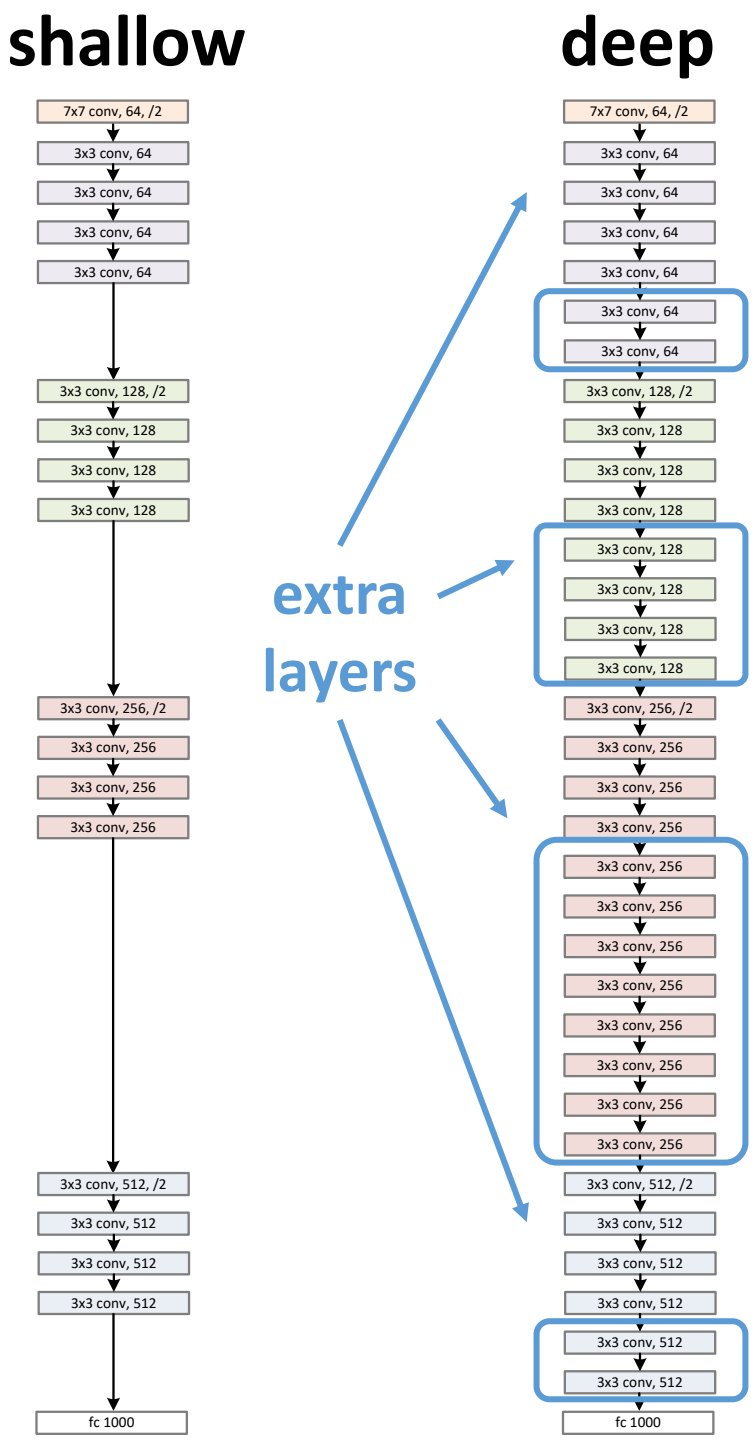


A thought experiment

- Richer solution space
- Deeper nets should not degrade in training

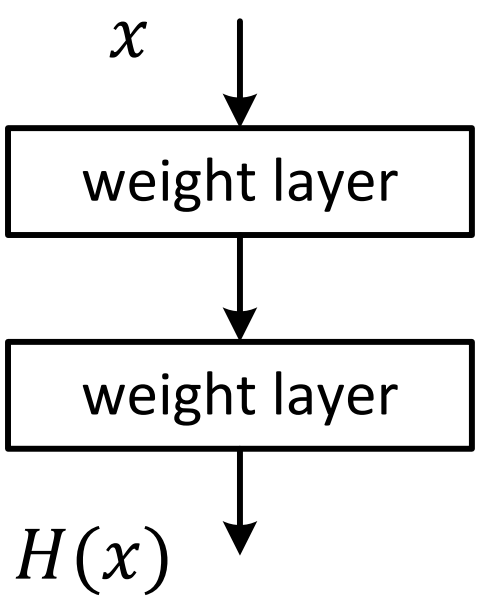
- Solution by construction
  - take a good shallow net
  - copy trained layers
  - let extra layers be **identity mapping**
  - no worse training accuracy

- Optimizer cannot find the solution



Deep Residual Learning

a subnet in a deep net



classical network

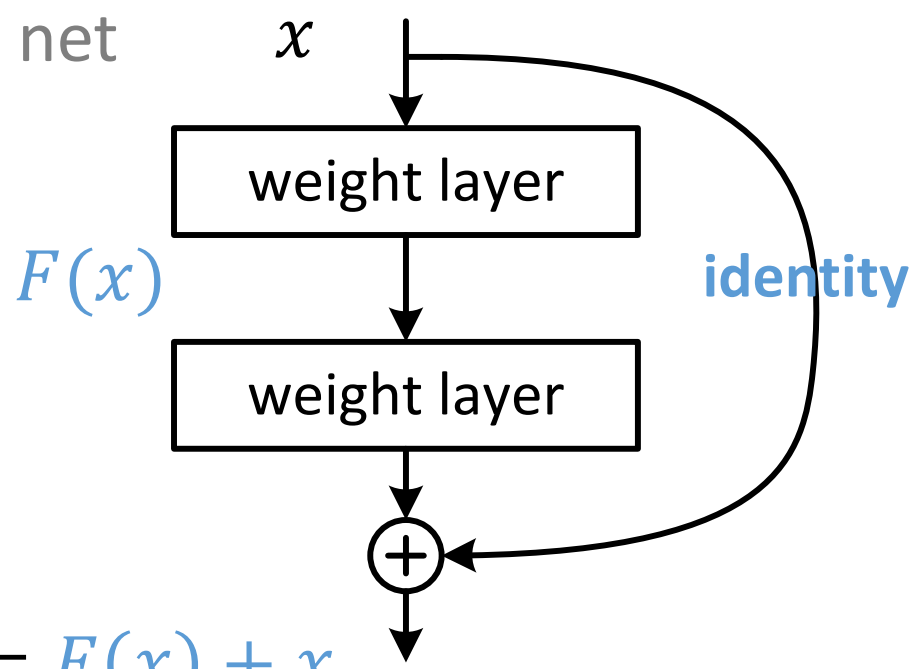
- $H(x)$: desired function to be fit by a subnet

- let weight layers fit $H(x)$

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015, CVPR 2016.

Deep Residual Learning

a subnet in a deep net



$H(x) = F(x) + x$

residual block

- $H(x)$: desired function to be fit by a subnet

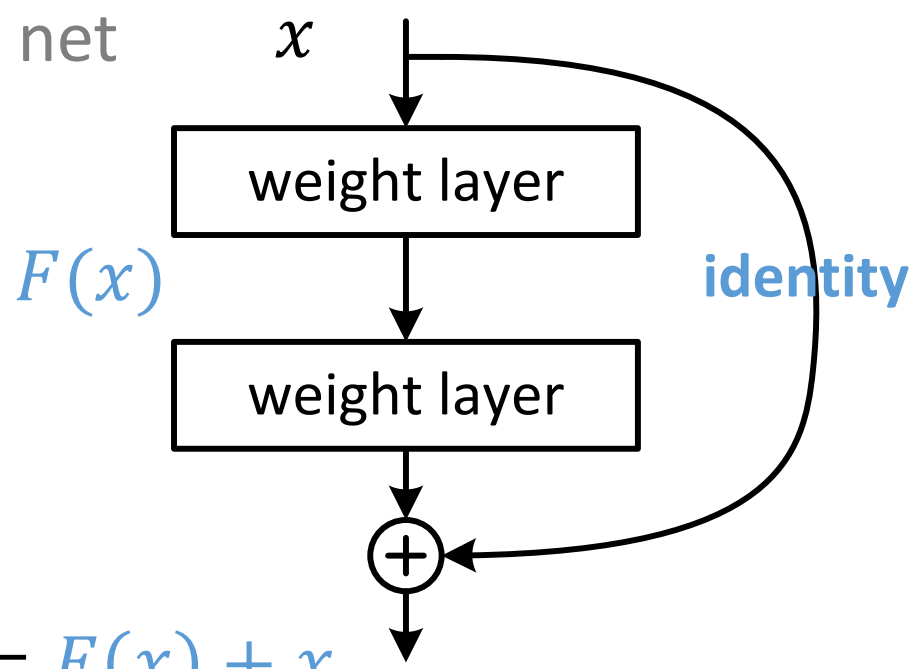
- ~~let weight layers fit $H(x)$~~
- let weight layers fit $F(x)$

- set $H(x) = F(x) + x$

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015, CVPR 2016.

Deep Residual Learning

a subnet in
a deep net



$$H(x) = F(x) + x$$

residual block

- $F(x)$: residual function
- if $H(x)$ = identity is near-optimal
  - push weights to small
  - encourage small changes
- initialization
  - small or zero weights

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015, CVPR 2016.

Residual Networks (ResNet)

Building very deep nets:

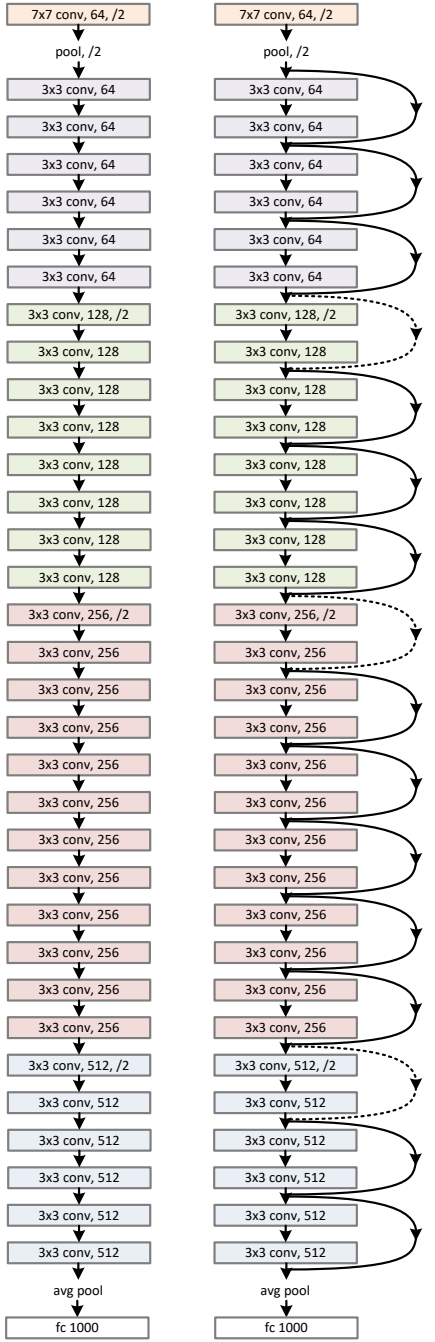
- add **identity connections** to vanilla nets
  (a.k.a. skip/shortcut/residual connections)

or:

- stack many **residual blocks**

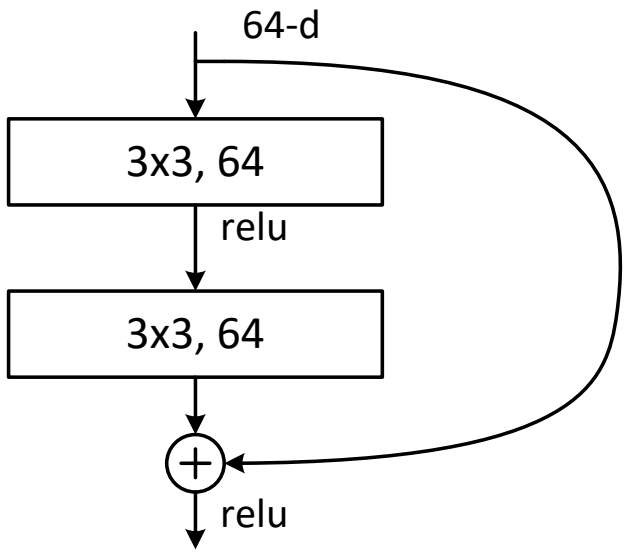
Residual Blocks:

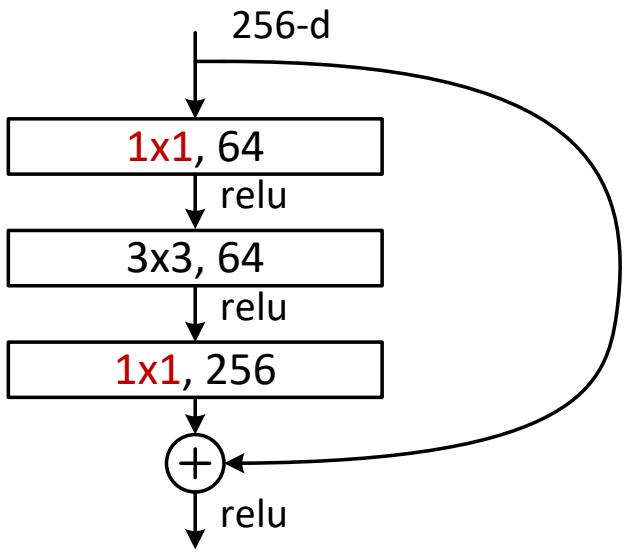
- new generic modules for neural nets
- design blocks and compose them



Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015, CVPR 2016.

Residual Block





Basic Block

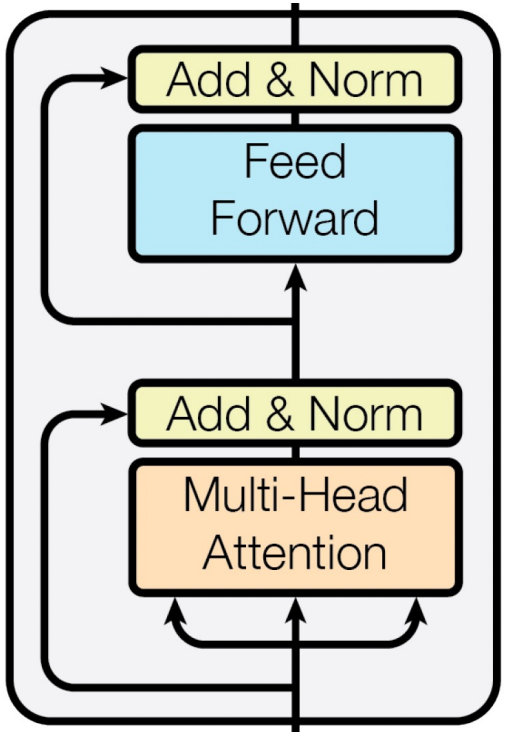
- 2-layer per block
- uniform width
- VGG style: all 3x3 conv

Bottleneck Block

- 3-layer per block
- bottleneck: reduce dim
- w/ 1x1 conv for dim control

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015, CVPR 2016.

Residual Block: Transformer

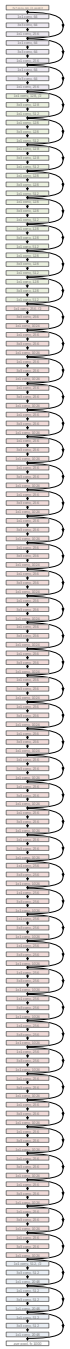


A Transformer Block has two Residual Blocks.

"Attention is all you need", Vaswani, et al., 2017

Residual Networks (ResNet)

ResNet-152



- For legacy reasons, “ResNet” often refers to the original instantiations
  - ResNet-18, 34: basic blocks
  - ResNet-50, 101, 152: bottleneck blocks

- In general, a ResNet is a network with the “$F(x) + x$” blocks

- The effect of residual learning is not limited to special instantiations

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015, CVPR 2016.

Deep Residual Networks (ResNet)

ResNet-152

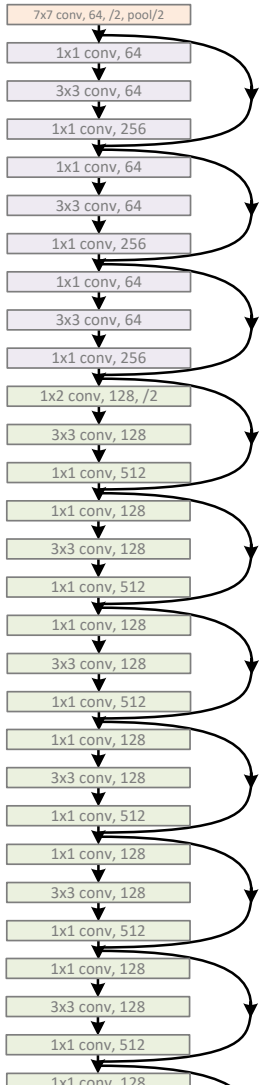
VGG-19

AlexNet

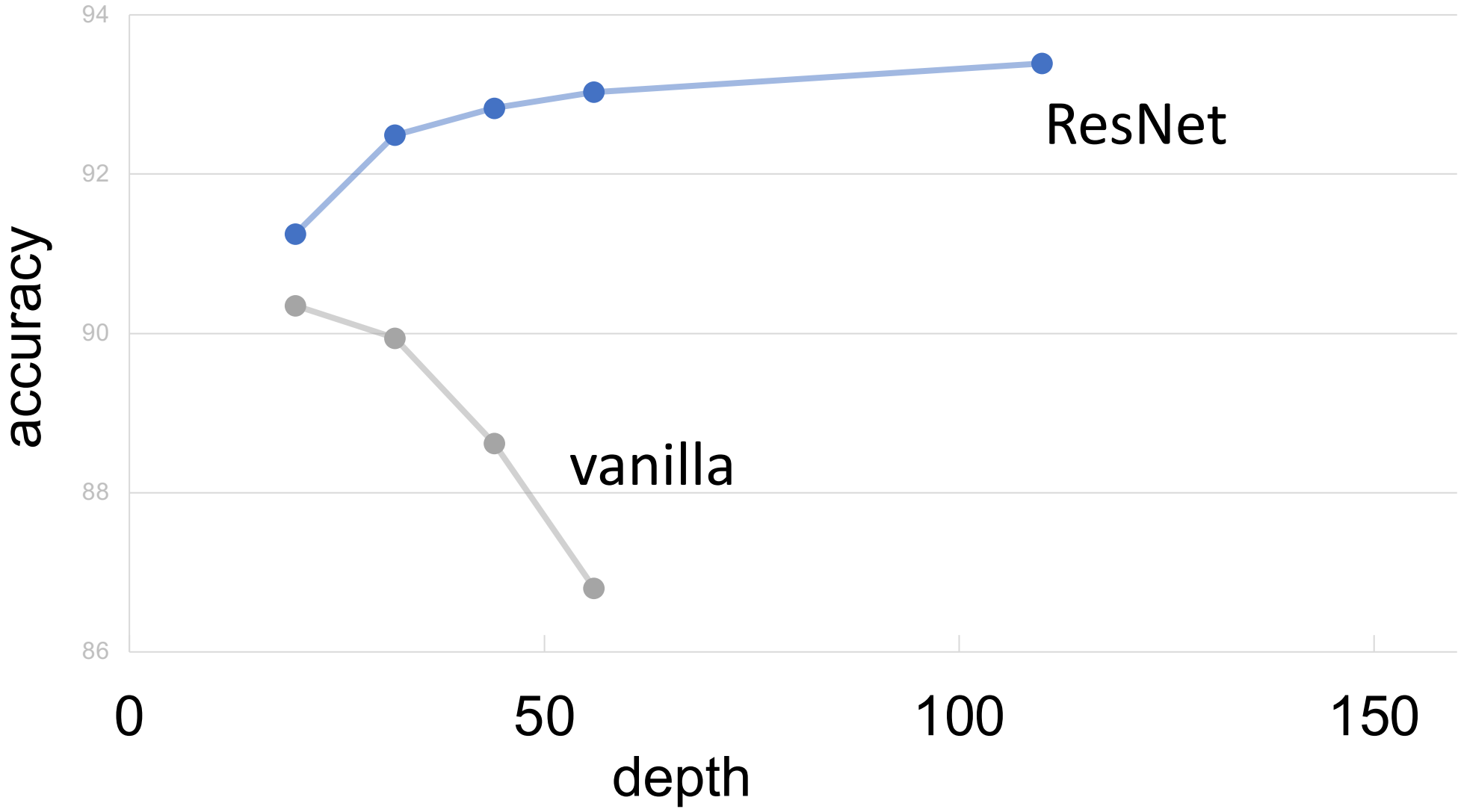
Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015, CVPR 2016.

Deep Residual Networks (ResNet)

ResNet-152

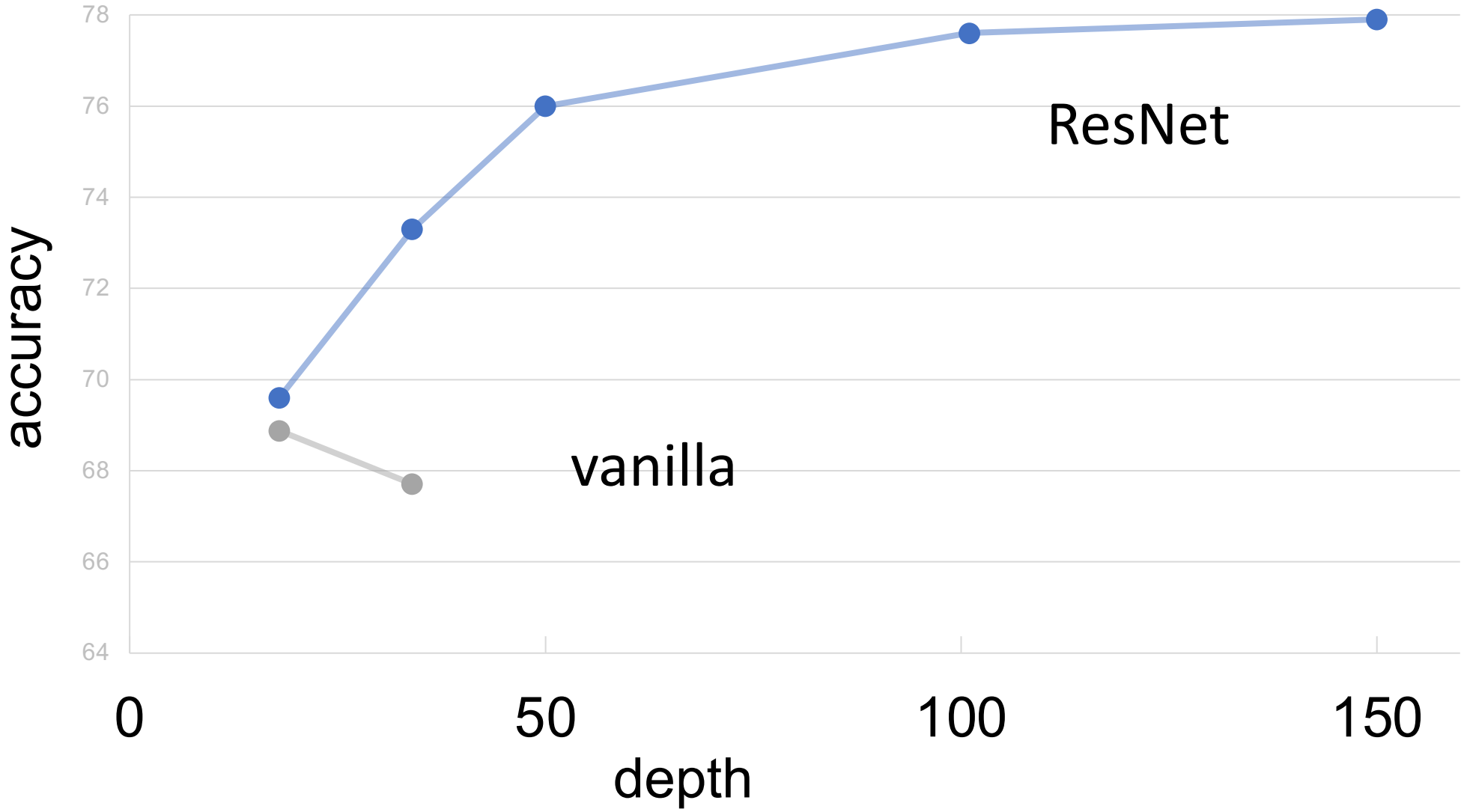


Experiment on CIFAR-10 classification



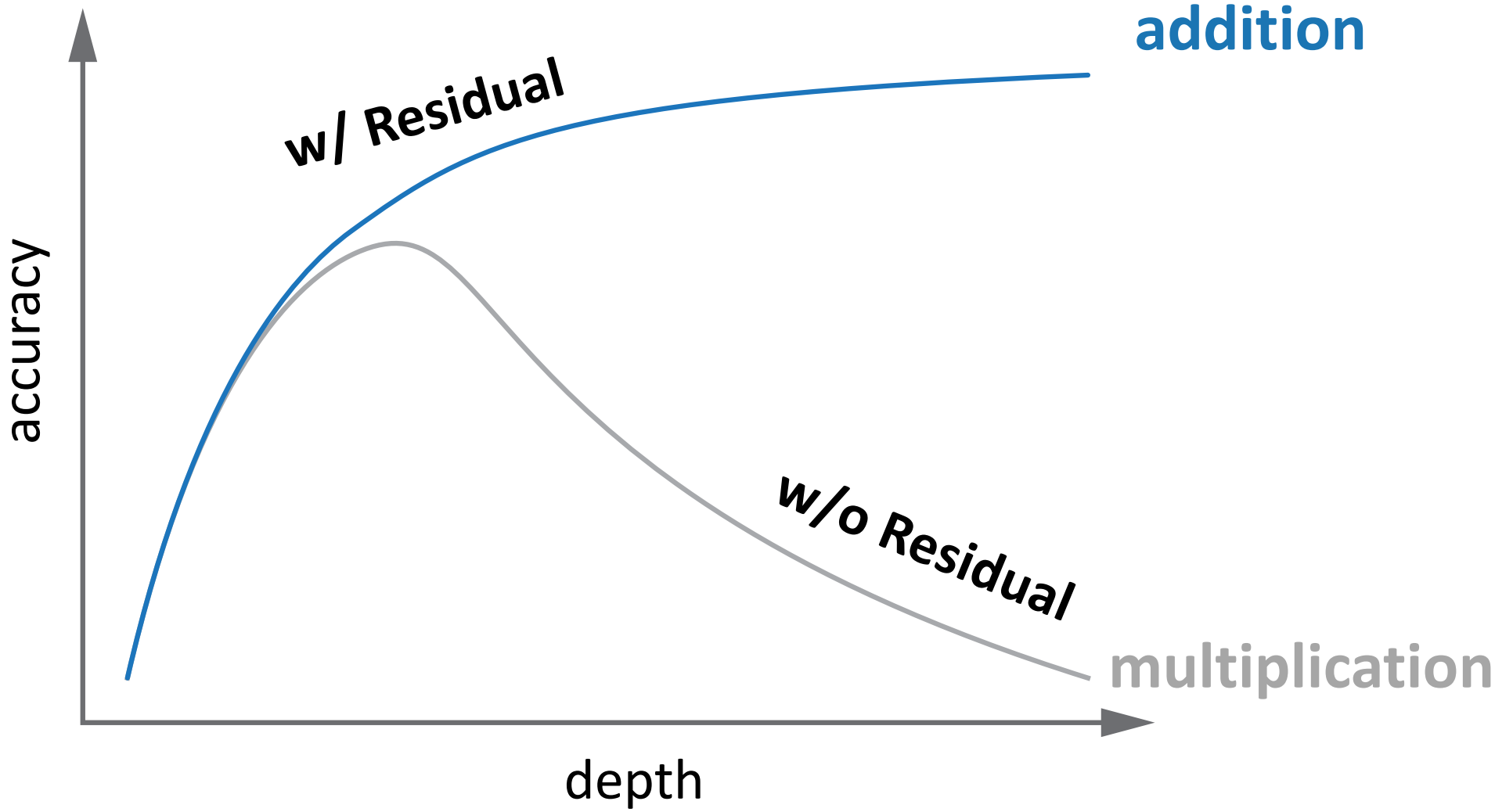
Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015, CVPR 2016.

Experiment on ImageNet classification



Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015, CVPR 2016.

Understanding Residual Learning



Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015, CVPR 2016.

Understanding Residual Learning

Vanilla network

$$x_{l+1} = W_l x_l$$

recursively

$$x_L = \left(\prod W_i\right) x_l$$

gradients

$$\frac{\partial \mathcal{E}}{\partial x_l} = \frac{\partial \mathcal{E}}{\partial x_L}\left(\prod W_i\right)$$

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Identity Mappings in Deep Residual Networks”, ECCV 2016

Understanding Residual Learning

Vanilla network

$$x_{l+1} = W_l x_l$$

recursively

$$x_L = \left(\prod W_i\right) x_l$$

gradients

$$\frac{\partial \mathcal{E}}{\partial x_l} = \frac{\partial \mathcal{E}}{\partial x_L}\left(\prod W_i\right)$$

- **vanishing signal**

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Identity Mappings in Deep Residual Networks”, ECCV 2016

Understanding Residual Learning

Residual network

$$x_{l+1} = x_l + F_l(x_l)$$

recursively

$$x_L = x_l + \sum F_i(x_i)$$

gradients

$$\frac{\partial \mathcal{E}}{\partial x_l} = \frac{\partial \mathcal{E}}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum F_i(x_i)\right)$$

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Identity Mappings in Deep Residual Networks”, ECCV 2016

Understanding Residual Learning

Residual network

$$x_{l+1} = x_l + F_l(x_l)$$

recursively

$$x_L = x_l + \sum F_i(x_i)$$

gradients

$$\frac{\partial \mathcal{E}}{\partial x_l} = \frac{\partial \mathcal{E}}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum F_i(x_i)\right)$$

- **not vanishing**

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Identity Mappings in Deep Residual Networks”, ECCV 2016

Understanding Residual Learning

Residual network

$$x_{l+1} = x_l + F_l(x_l)$$

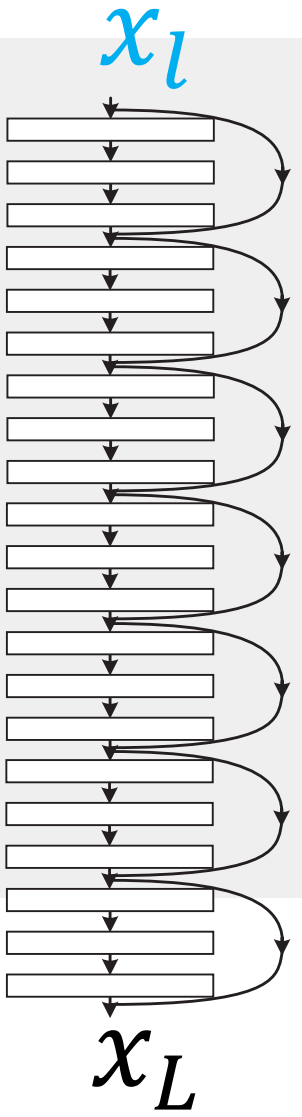
recursively

$$x_L = x_l + \sum F_i(x_i)$$

gradients

$$\frac{\partial \mathcal{E}}{\partial x_l} = \frac{\partial \mathcal{E}}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum F_i(x_i)\right)$$

- **not vanishing**



Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. "Identity Mappings in Deep Residual Networks", ECCV 2016

Understanding Residual Learning

Residual network

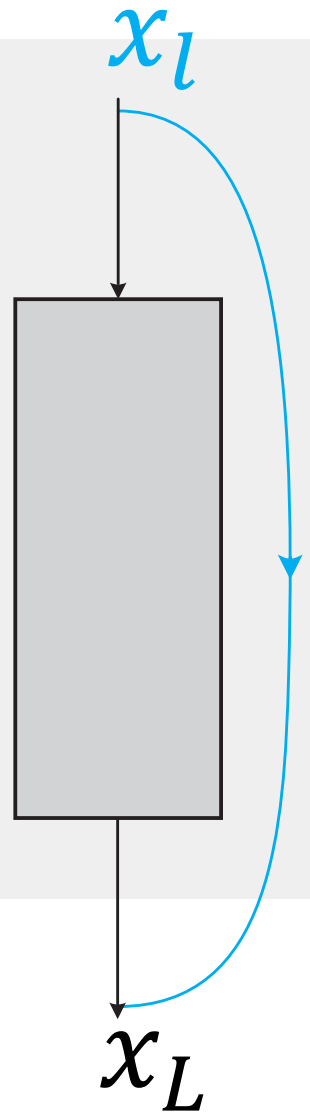
$$x_{l+1} = x_l + F_l(x_l)$$

recursively

$$x_L = x_l + \sum F_i(x_i)$$

gradients

$$\frac{\partial \mathcal{E}}{\partial x_l} = \frac{\partial \mathcal{E}}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum F_i(x_i)\right)$$



- **not vanishing**

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Identity Mappings in Deep Residual Networks”, ECCV 2016

Understanding Residual Learning

Residual network

$$x_{l+1} = x_l + F_l(x_l)$$

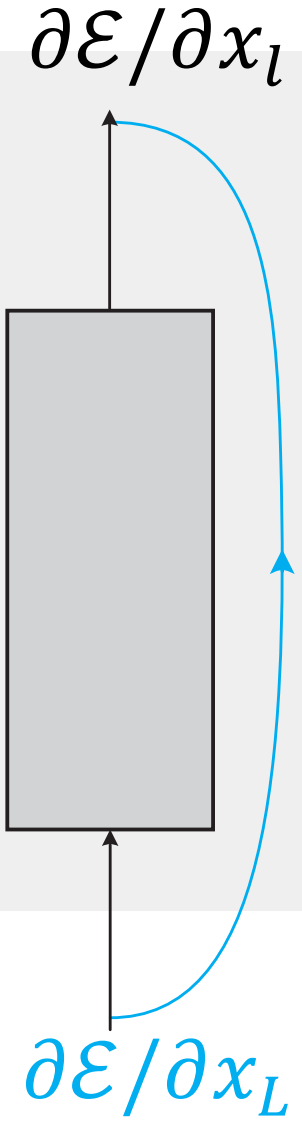
recursively

$$x_L = x_l + \sum F_i(x_i)$$

gradients

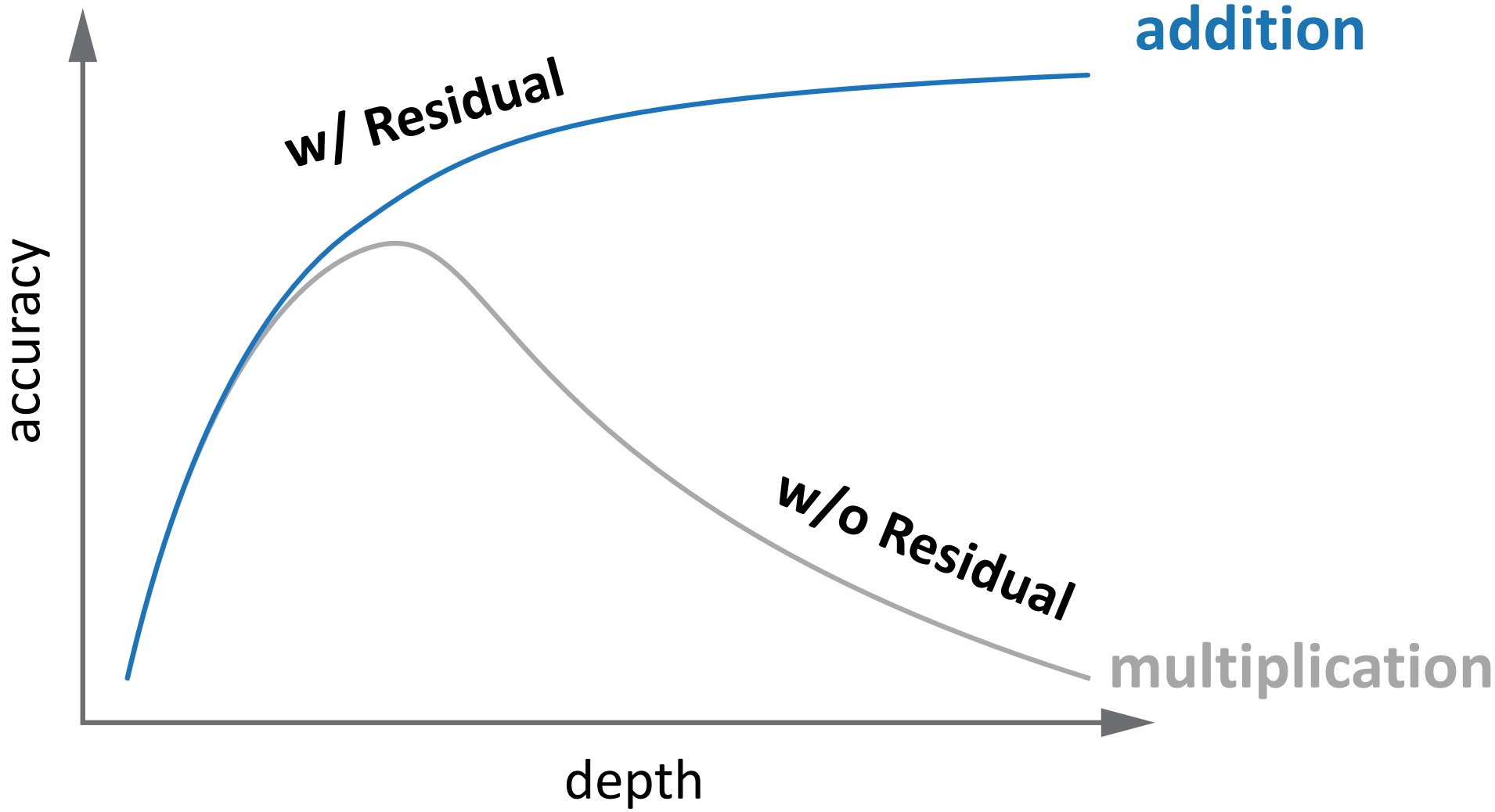
$$\frac{\partial \mathcal{E}}{\partial x_l} = \frac{\partial \mathcal{E}}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum F_i(x_i)\right)$$

- **not vanishing**



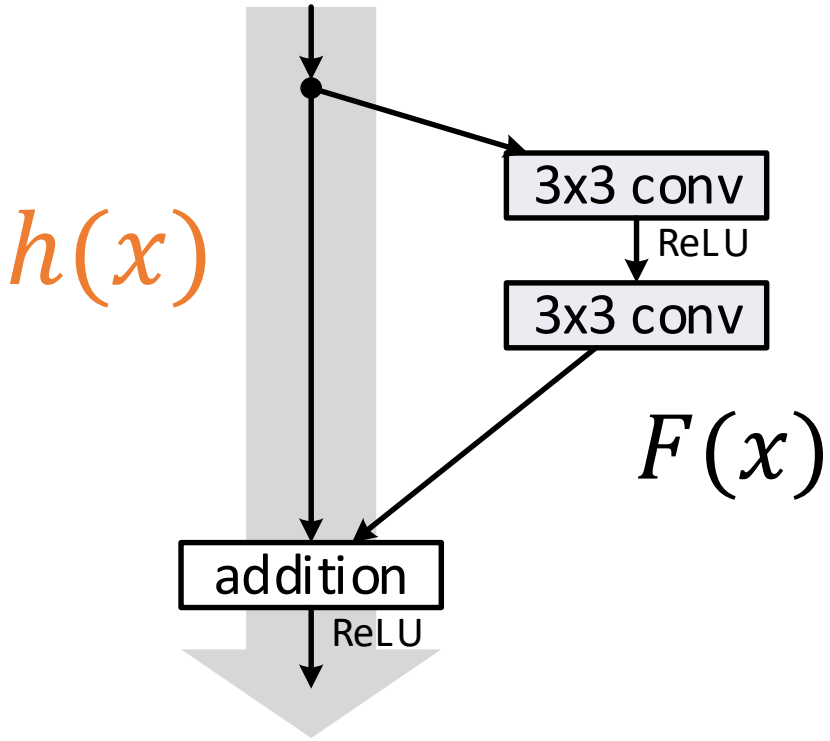
Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. "Identity Mappings in Deep Residual Networks", ECCV 2016

Understanding Residual Learning



Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015, CVPR 2016.

Identity Mappings in Residual Learning



- Residual Learning: h = identity mapping

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Identity Mappings in Deep Residual Networks”, ECCV 2016

Question 1:
What if shortcut mapping h ≠ identity?

if h is multiplicative …

consider $h(x) = \lambda x$:

$$x_{l+1} = \lambda x_l + F_l(x_l)$$

recursively

$$x_L = \lambda^{L-l} x_l + \sum \hat{F}_i(x_i)$$

gradients

$$\frac{\partial \mathcal{E}}{\partial x_l} = \frac{\partial \mathcal{E}}{\partial x_L}\left(\lambda^{L-l} + \frac{\partial}{\partial x_l}\sum \hat{F}_i(x_i)\right)$$

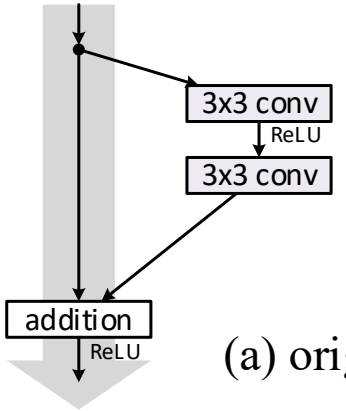
- **vanishing signals, if $\lambda < 1$**

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Identity Mappings in Deep Residual Networks”, ECCV 2016

ResNet-110 on CIFAR-10

$h(x) = x$

error: **6.6%**

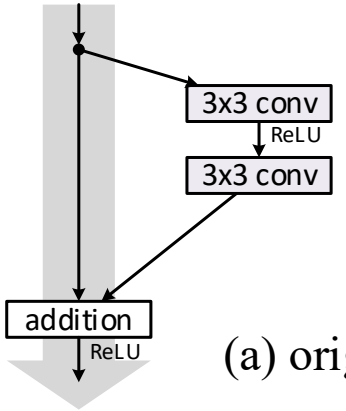


(a) original

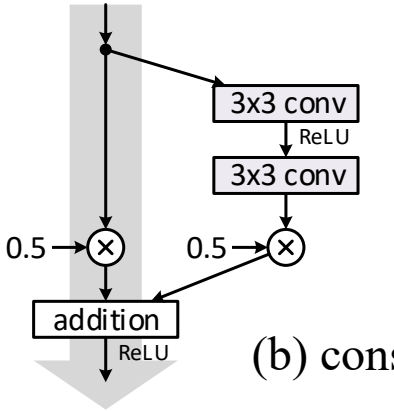
ResNet-110 on CIFAR-10

$h(x) = x$

error: **6.6%**



(a) original



(b) constant scaling

$h(x) = 0.5x$

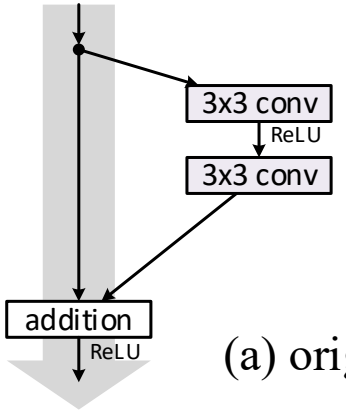
error: **12.4%**

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. "Identity Mappings in Deep Residual Networks", ECCV 2016

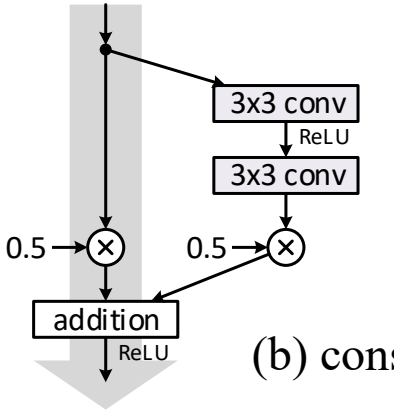
ResNet-110 on CIFAR-10

$h(x) = x$

error: **6.6%**



(a) original



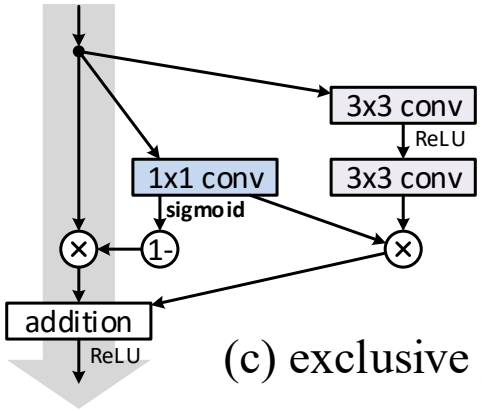
(b) constant scaling

$h(x) = 0.5x$

error: **12.4%**

$h(x) = \text{gate} \cdot x$

error: **8.7%**



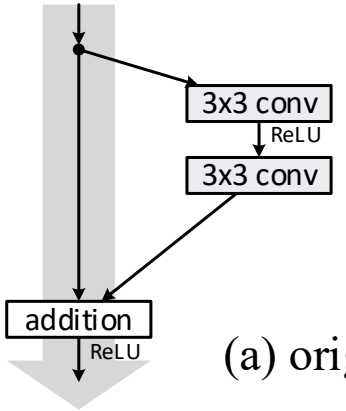
(c) exclusive gating

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. "Identity Mappings in Deep Residual Networks", ECCV 2016

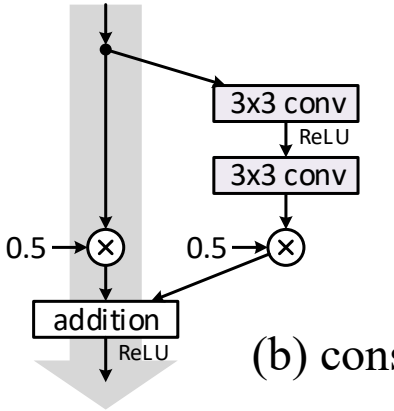
ResNet-110 on CIFAR-10

$h(x) = x$

error: **6.6%**



(a) original



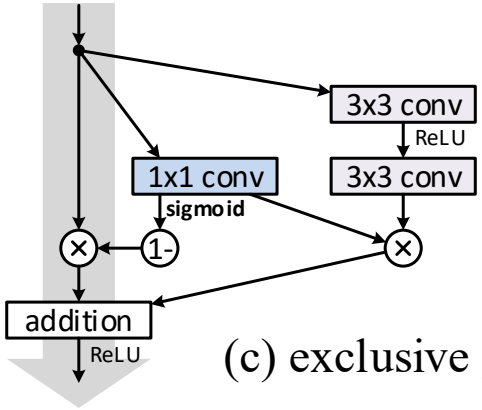
(b) constant scaling

$h(x) = 0.5x$

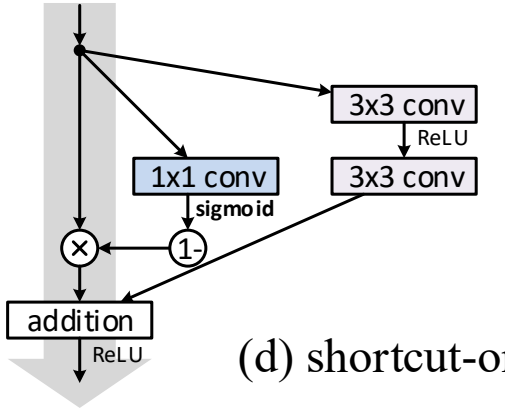
error: **12.4%**

$h(x) = \text{gate} \cdot x$

error: **8.7%**



(c) exclusive gating



(d) shortcut-only gating

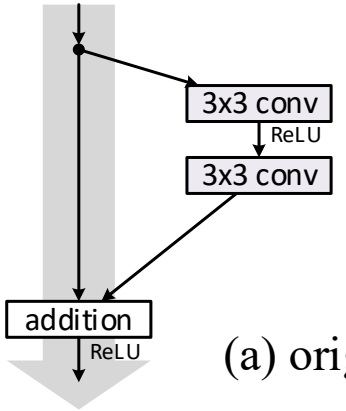
$h(x) = \text{gate} \cdot x$

error: **12.9%**

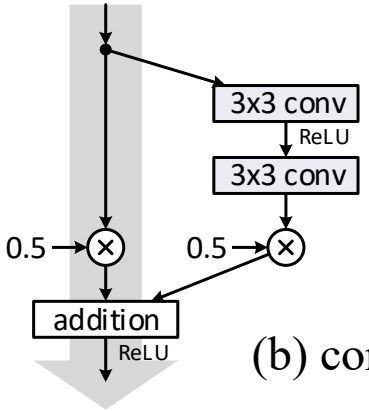
Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. "Identity Mappings in Deep Residual Networks", ECCV 2016

$h(x) = x$

error: **6.6%**



(a) original



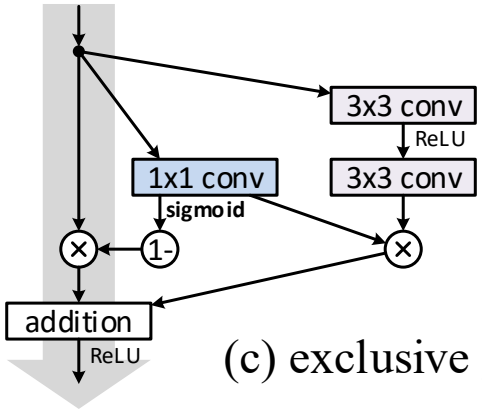
(b) constant scaling

$h(x) = 0.5x$

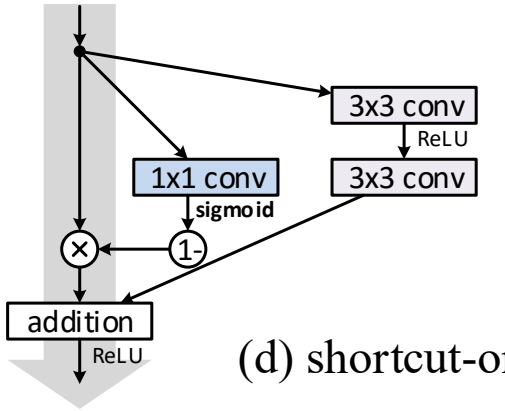
error: **12.4%**

$h(x) = \text{gate} \cdot x$

error: **8.7%**



(c) exclusive gating



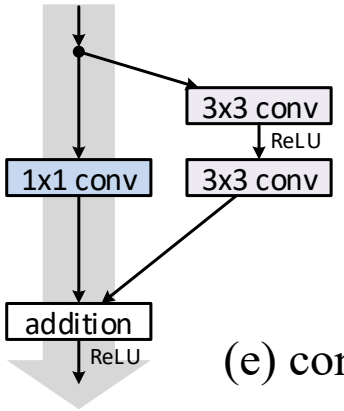
(d) shortcut-only gating

$h(x) = \text{gate} \cdot x$

error: **12.9%**

$h(x) = \text{conv}(x)$

error: **12.2%**

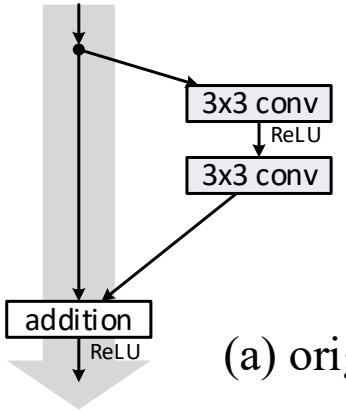


(e) conv shortcut

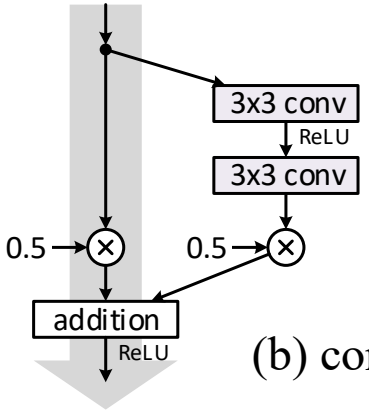
ResNet-110 on CIFAR-10

$h(x) = x$

error: **6.6%**



(a) original



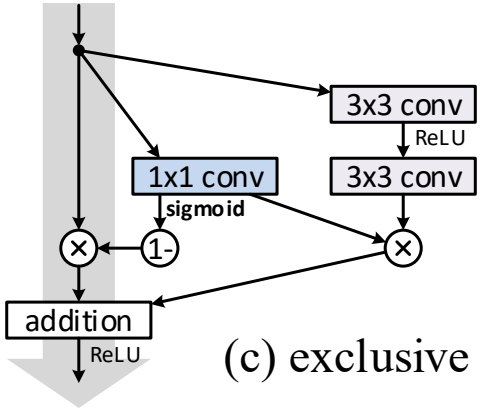
(b) constant scaling

$h(x) = 0.5x$

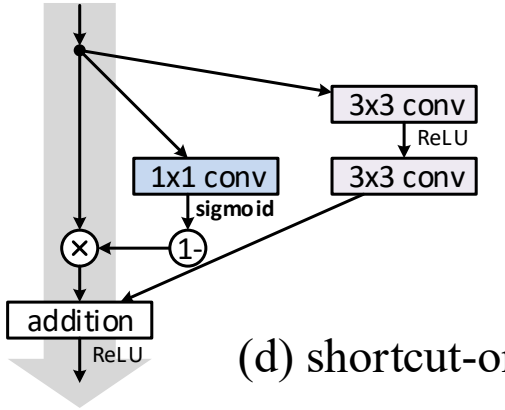
error: **12.4%**

$h(x) = \text{gate} \cdot x$

error: **8.7%**



(c) exclusive gating



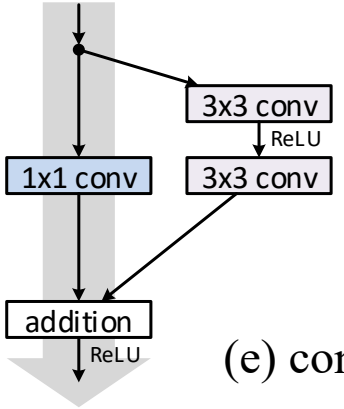
(d) shortcut-only gating

$h(x) = \text{gate} \cdot x$

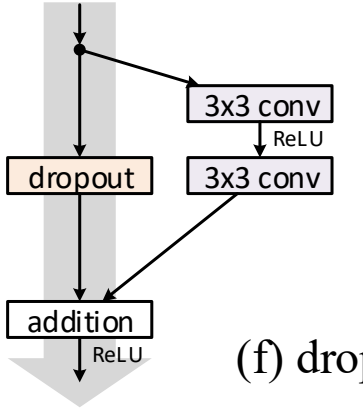
error: **12.9%**

$h(x) = \text{conv}(x)$

error: **12.2%**



(e) conv shortcut



(f) dropout shortcut

$h(x) = \text{dropout}(x)$

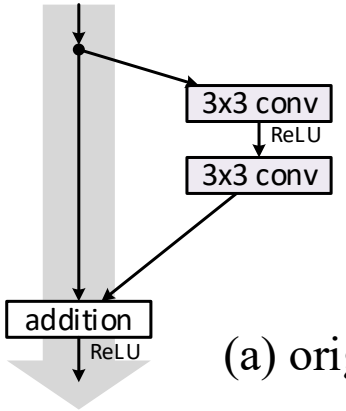
error: **> 20%**

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. "Identity Mappings in Deep Residual Networks", ECCV 2016

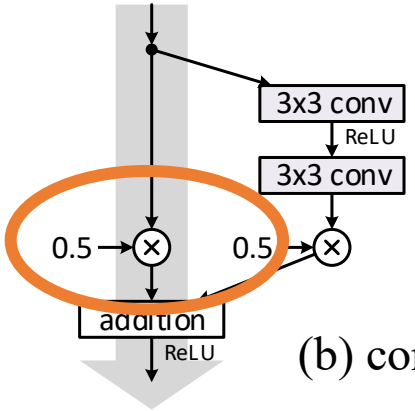
ResNet-110 on CIFAR-10

$h(x) = x$

error: **6.6%**



(a) original



(b) constant scaling

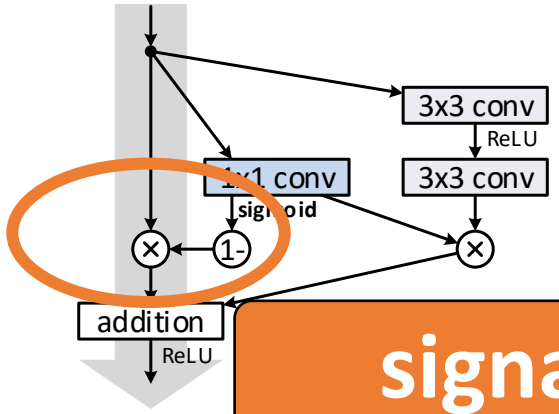
$h(x) = 0.5x$

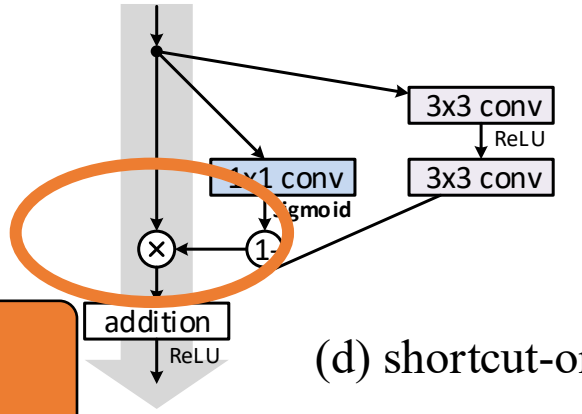
error: **12.4%**

$h(x) = \text{gate} \cdot x$

error: **8.7%**

*like "Highway Net"





(d) shortcut-only gating

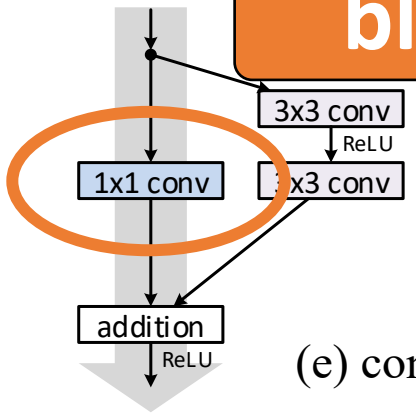
$h(x) = \text{gate} \cdot x$

error: **12.9%**

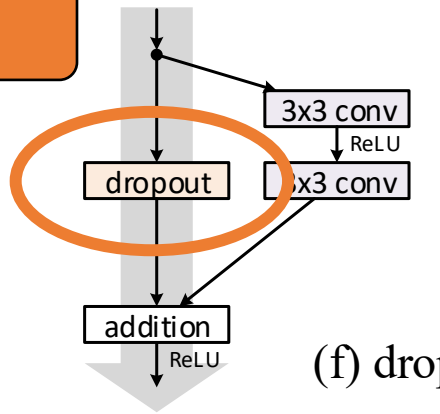
signals blocked!

$h(x) = \text{conv}(x)$

error: **12.2%**



(e) conv shortcut



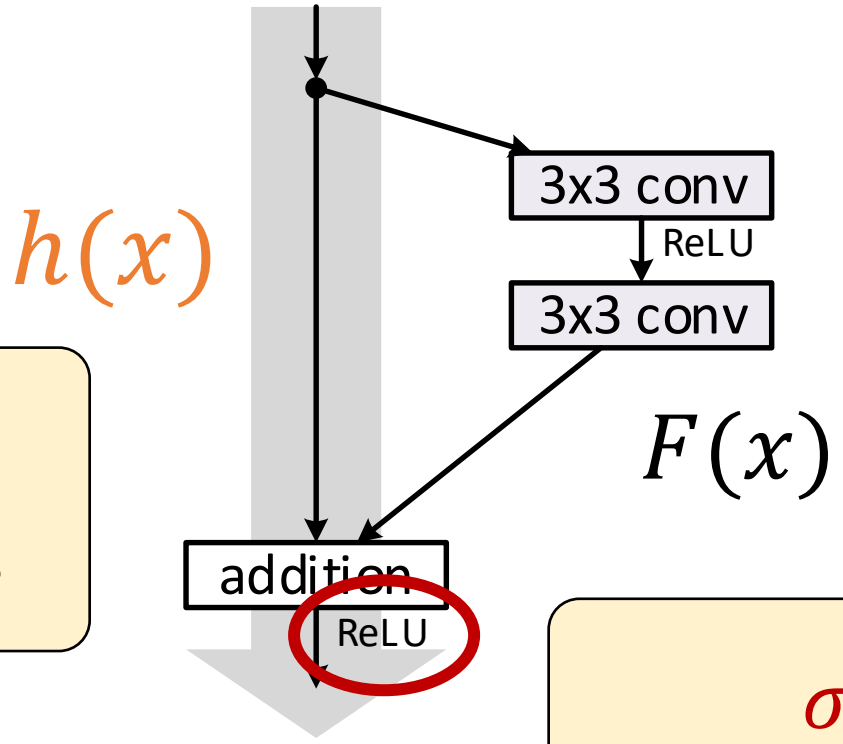
(f) dropout shortcut

$h(x) = \text{dropout}(x)$

error: **> 20%**

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. "Identity Mappings in Deep Residual Networks", ECCV 2016

Identity Mappings in Residual Learning



h = identity:
Don't block signals!

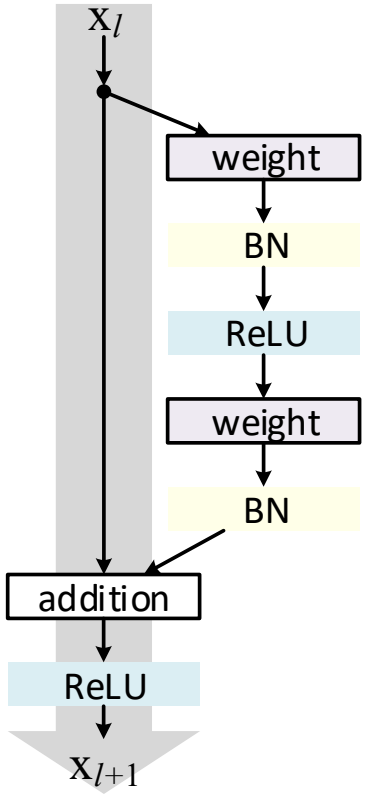
σ = ReLU:
Does it block signals?

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. "Identity Mappings in Deep Residual Networks", ECCV 2016

Question 2:
What if post activation σ = identity?

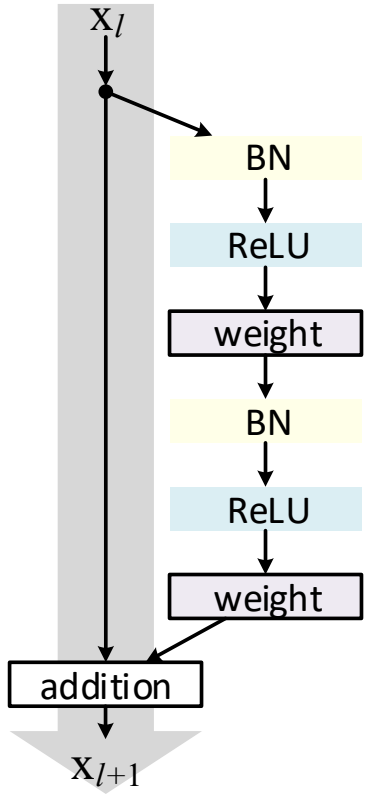
Pre-activation (Pre-normalization)

σ = ReLU



post-activation

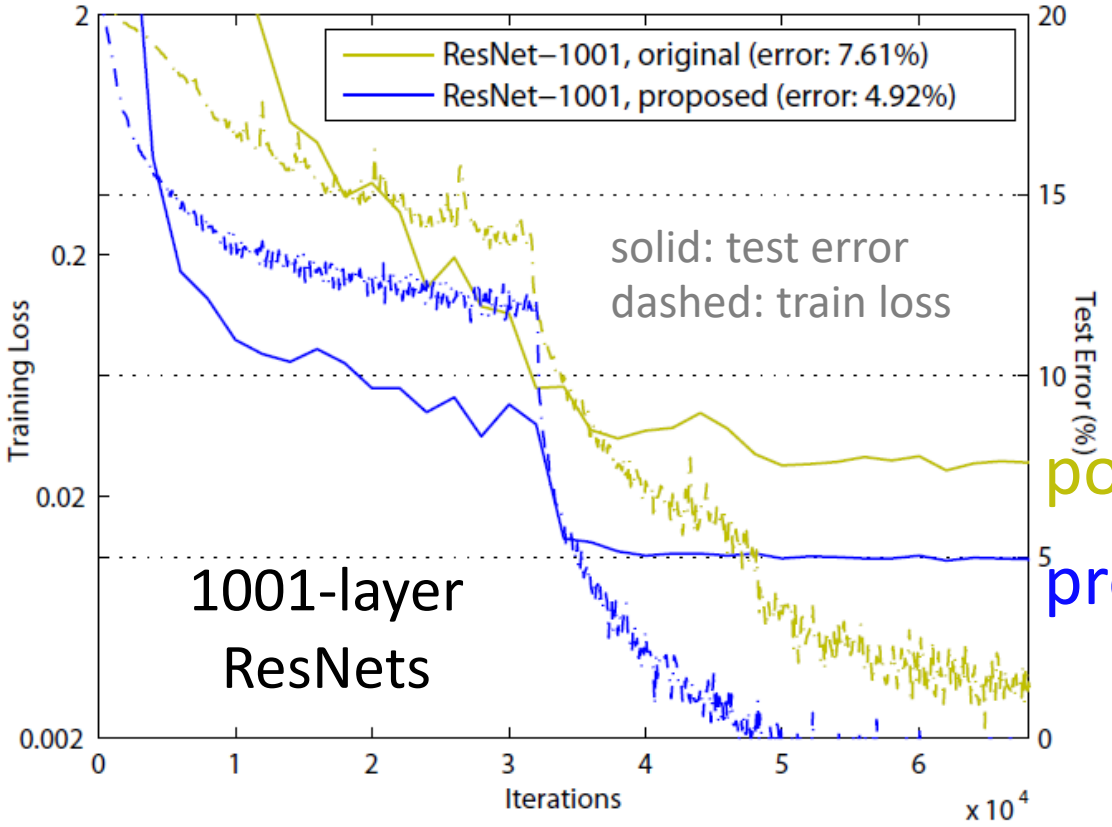
σ = identity



pre-activation

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Identity Mappings in Deep Residual Networks”, ECCV 2016

Pre-activation (Pre-normalization)



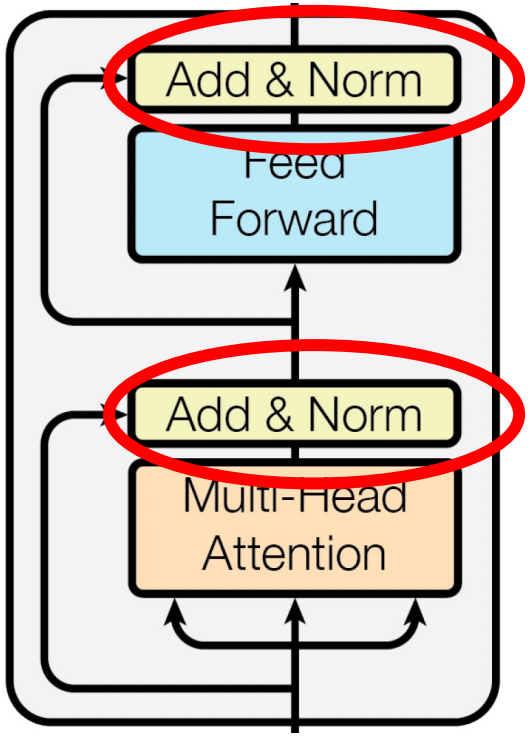
post-activation (σ = ReLU)

pre-activation (σ = identity)

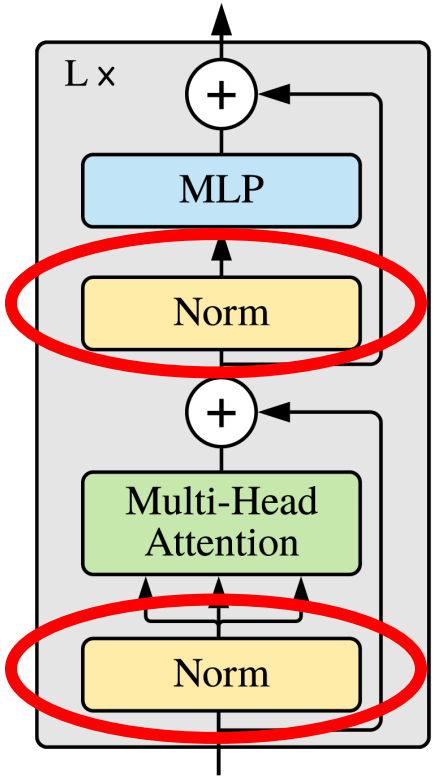
- ReLU can also block signals when there are many layers
- Pre-activation eases optimization

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Identity Mappings in Deep Residual Networks”, ECCV 2016

Pre-activation (Pre-normalization)



post-norm
(original Transformer)



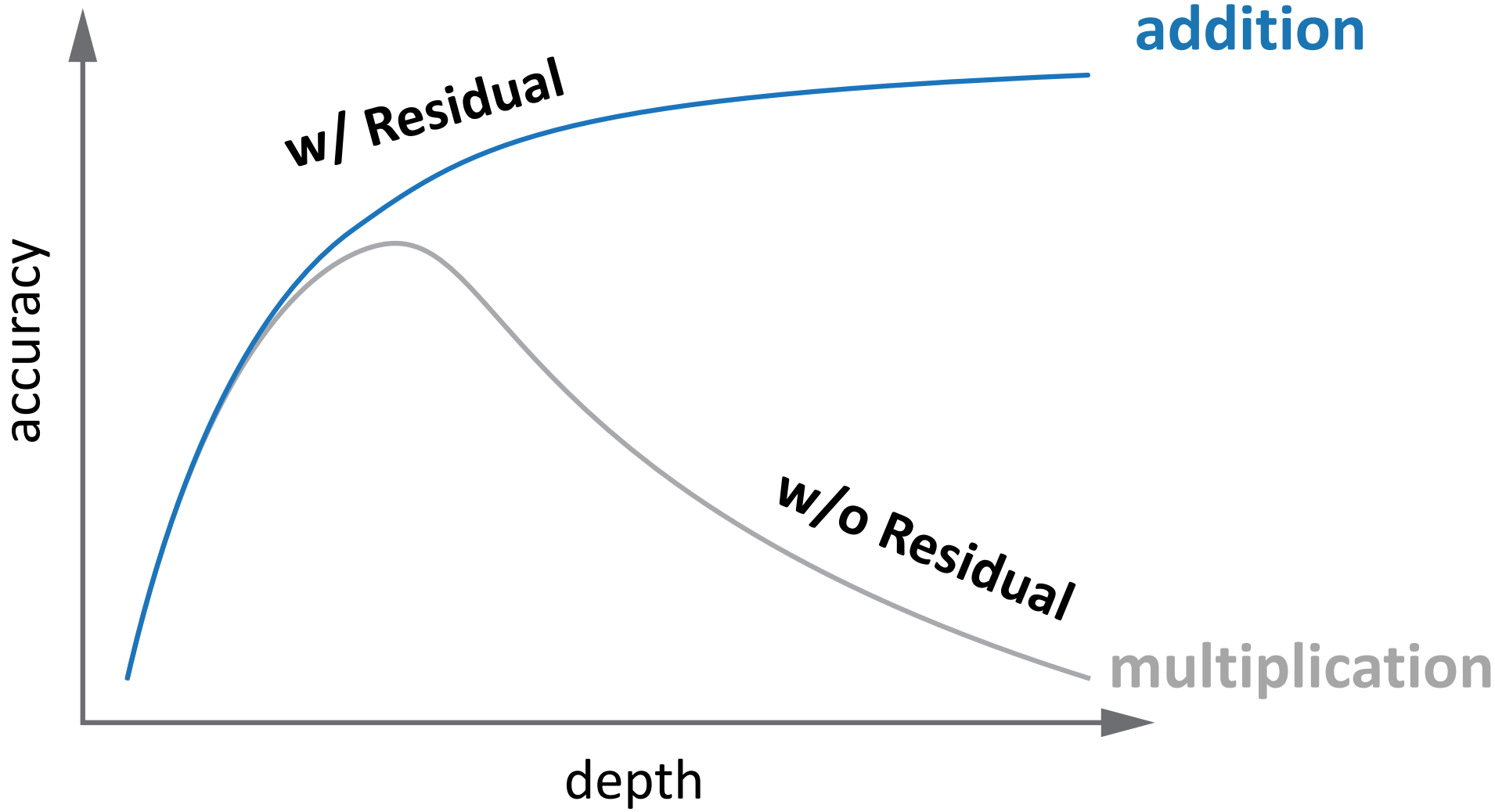
pre-norm
(after GPT-2)

few modifications. Layer normalization (Ba et al., 2016) was moved to the input of each sub-block, similar to a pre-activation residual network (He et al., 2016) and an additional layer normalization was added after the final self-attention block. A modified initialization which accounts

from the GPT-2 paper
[Radford, et al., 2019]

"Attention is all you need", Vaswani, et al., 2017
"Language Models are Unsupervised Multitask Learners" (GPT-2), Radford, et al., 2019

Understanding Residual Learning



Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015, CVPR 2016.

Understanding Residual Learning

The Shattered Gradients Problem:
If resnets are the answer, then what is the question?

David Balduzzi [1] Marcus Frean [1] Lennox Leary [1] JP Lewis [1 2] Kurt Wan-Duo Ma [1] Brian McWilliams [3]

Residual Networks Behave Like Ensembles of Relatively Shallow Networks

Andreas Veit **Michael Wilber** **Serge Belongie**
Department of Computer Science & Cornell Tech
Cornell University
{av443, mjw285, sjb344}@cornell.edu

Visualizing the Loss Landscape of Neural Nets

Hao Li[1], **Zheng Xu**[1], **Gavin Taylor**[2], **Christoph Studer**[3], **Tom Goldstein**[1]
[1]University of Maryland, College Park [2]United States Naval Academy [3]Cornell University
{haoli,xuzh,tomg}@cs.umd.edu, taylor@usna.edu, studer@cornell.edu

Identity Matters in Deep Learning

Moritz Hardt * Tengyu Ma†

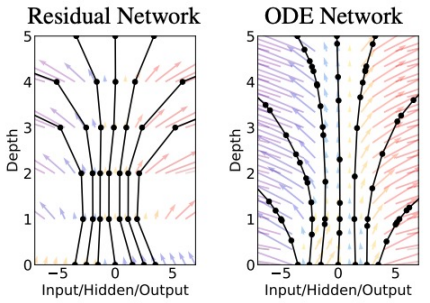
July 23, 2018

SKIP CONNECTIONS ELIMINATE SINGULARITIES

A. Emin Orhan **Xaq Pitkow**
aeminorhan@gmail.com xaq@rice.edu
Baylor College of Medicine & Rice University

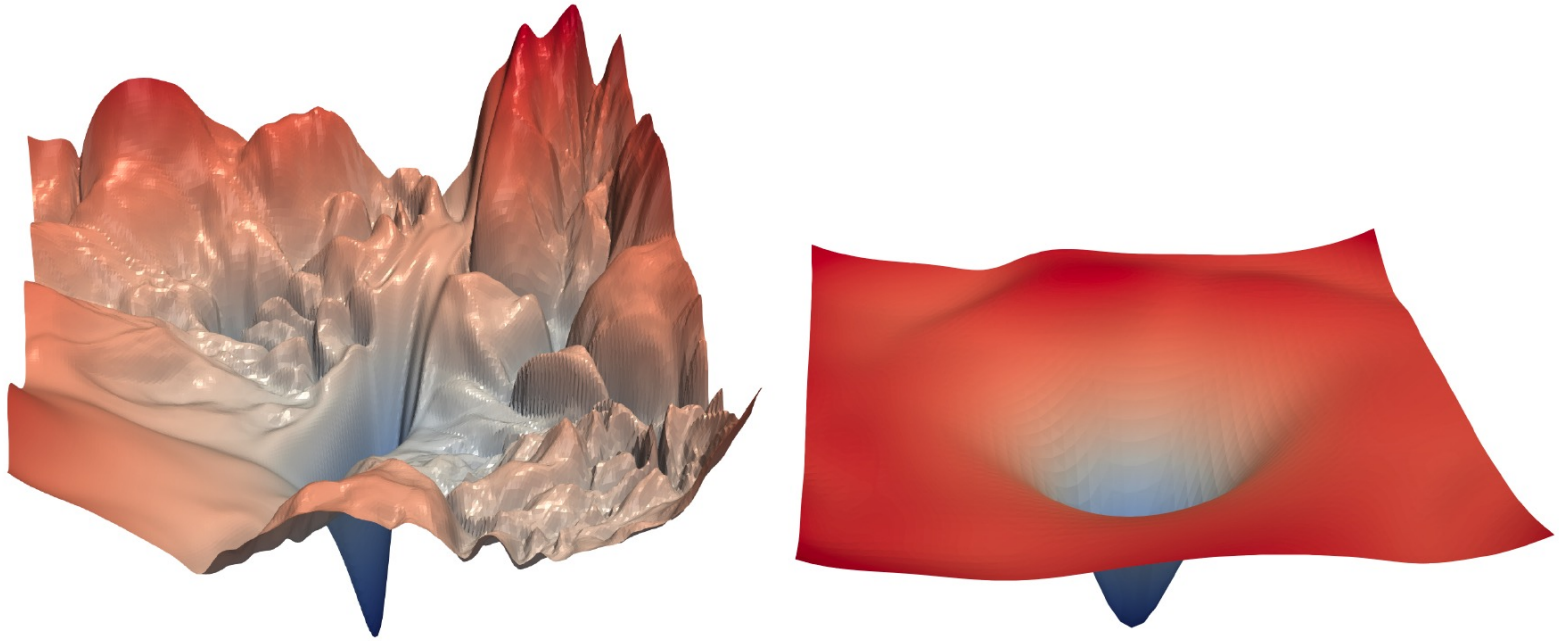
Neural Ordinary Differential Equations

Ricky T. Q. Chen*, **Yulia Rubanova***, **Jesse Bettencourt***, **David Duvenaud**
University of Toronto, Vector Institute
{rtqichen, rubanova, jessebett, duvenaud}@cs.toronto.edu



Understanding Residual Learning

- Residual connections produce smoother loss landscapes



(a) without skip connections

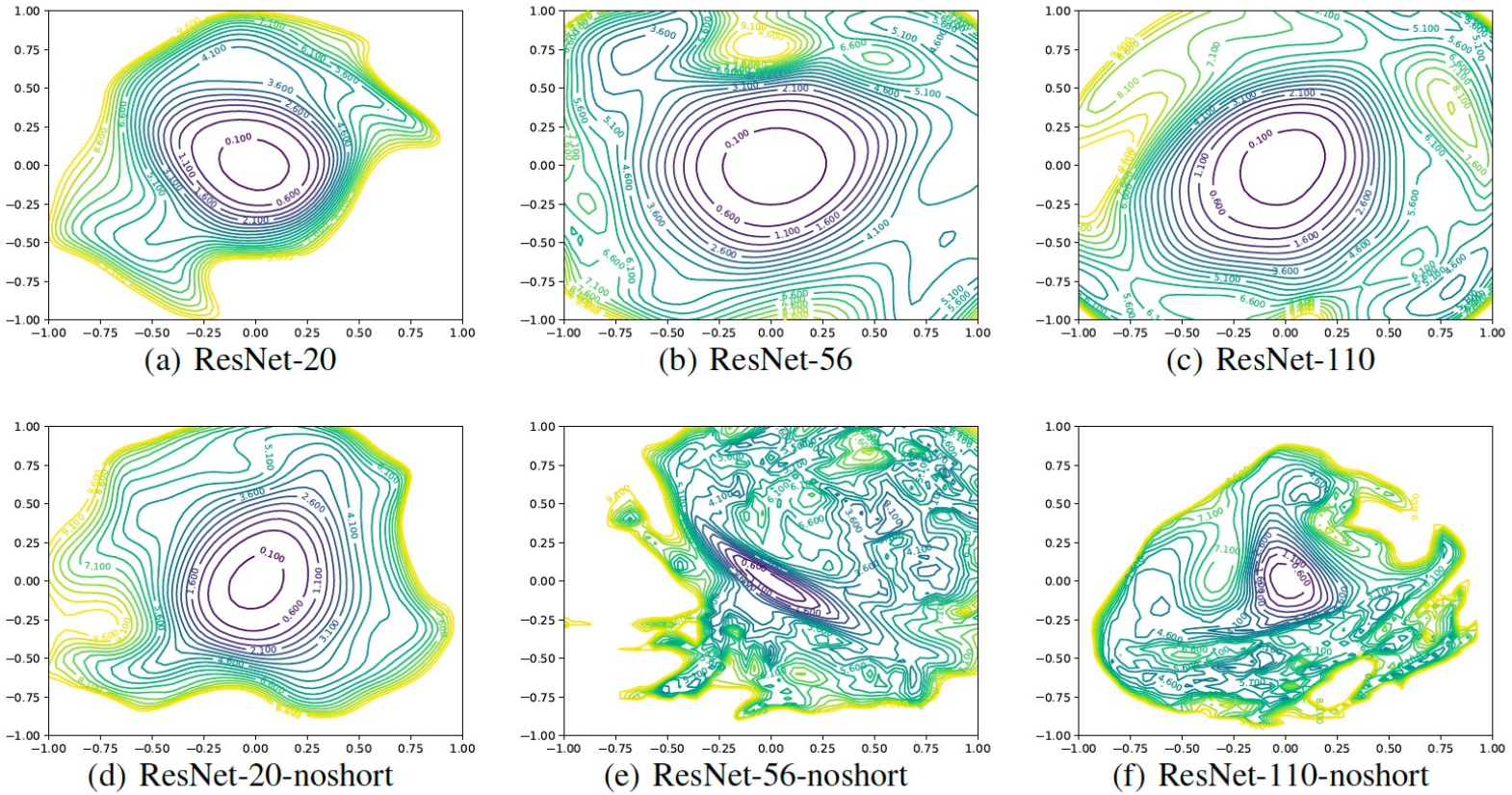
(b) with skip connections

Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The vertical axis is logarithmic to show dynamic range. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

Li, et al. “Visualizing the Loss Landscape of Neural Nets”, NeurIPS 2018

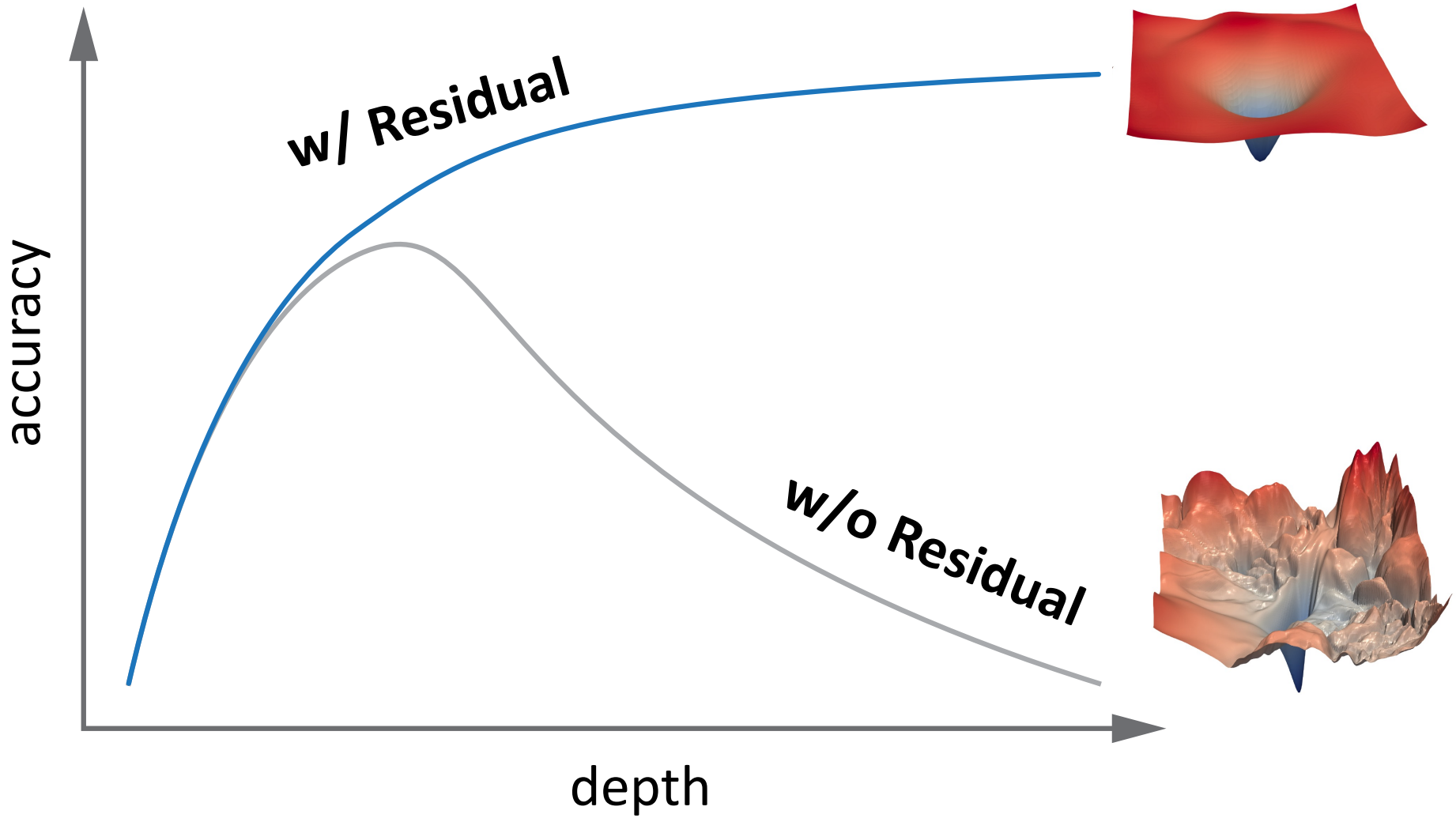
Understanding Residual Learning

- Without residual connections, deeper nets have more chaotic landscapes



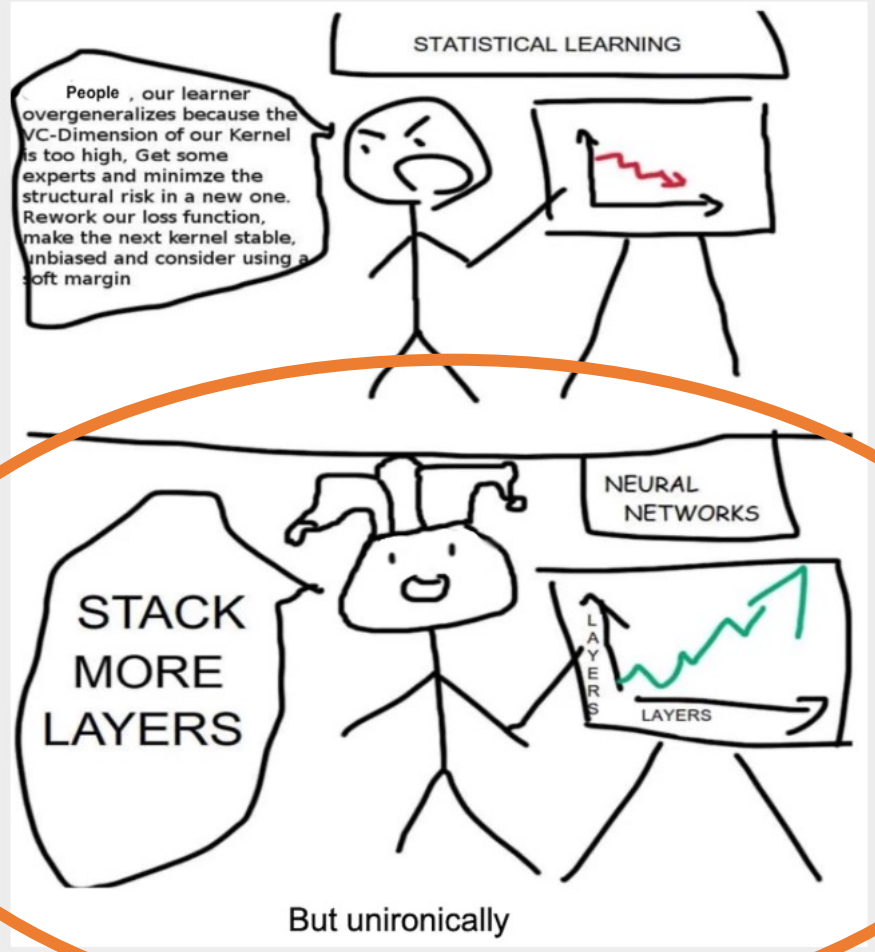
Li, et al. “Visualizing the Loss Landscape of Neural Nets”, NeurIPS 2018

Understanding Residual Learning



Li, et al. “Visualizing the Loss Landscape of Neural Nets”, NeurIPS 2018

User Can you explain why this is funny. Think about it step-by-step.



this lecture

GPT-4 The comic is satirizing the difference in approaches to improving model performance between statistical learning and neural networks.

source: https://openai.com/research/gpt-4

Lecture 10: Going deep with neural networks

- Why going deep is difficult?
- Weight initialization
- Normalization modules
- Deep Residual Learning