[Fall, 2021. ICE3050]

Deep Learning

- Convolutional Neural Networks 2-

Eunbyung Park

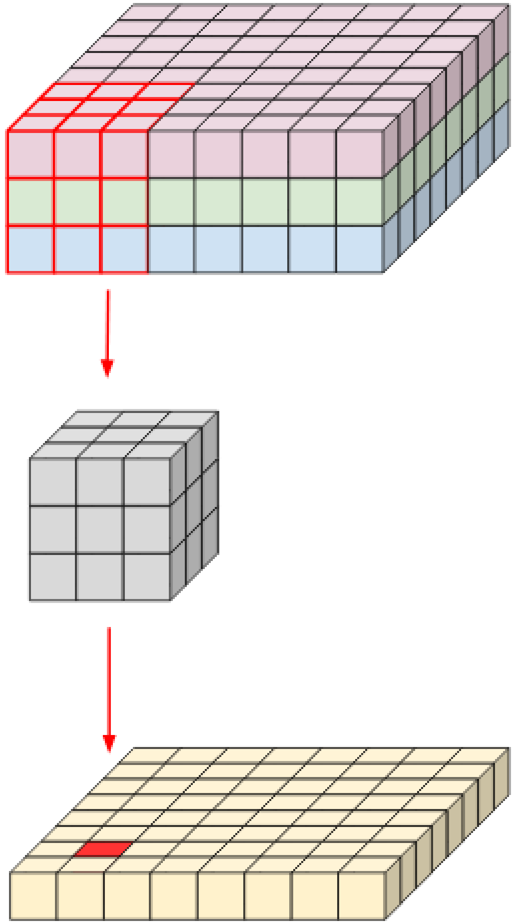
Assistant Professor

School of Electronic and Electrical Engineering

Eunbyung Park (silverbottlep.github.io)

Various Types of Convolutions

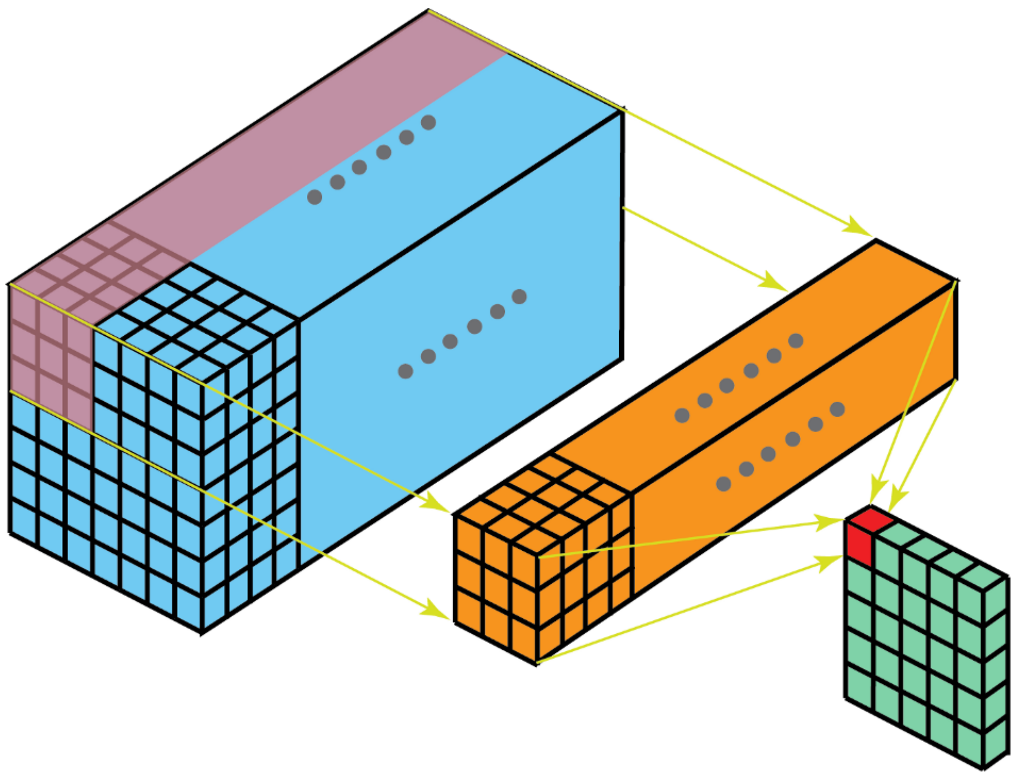
Normal Convolutional Layer



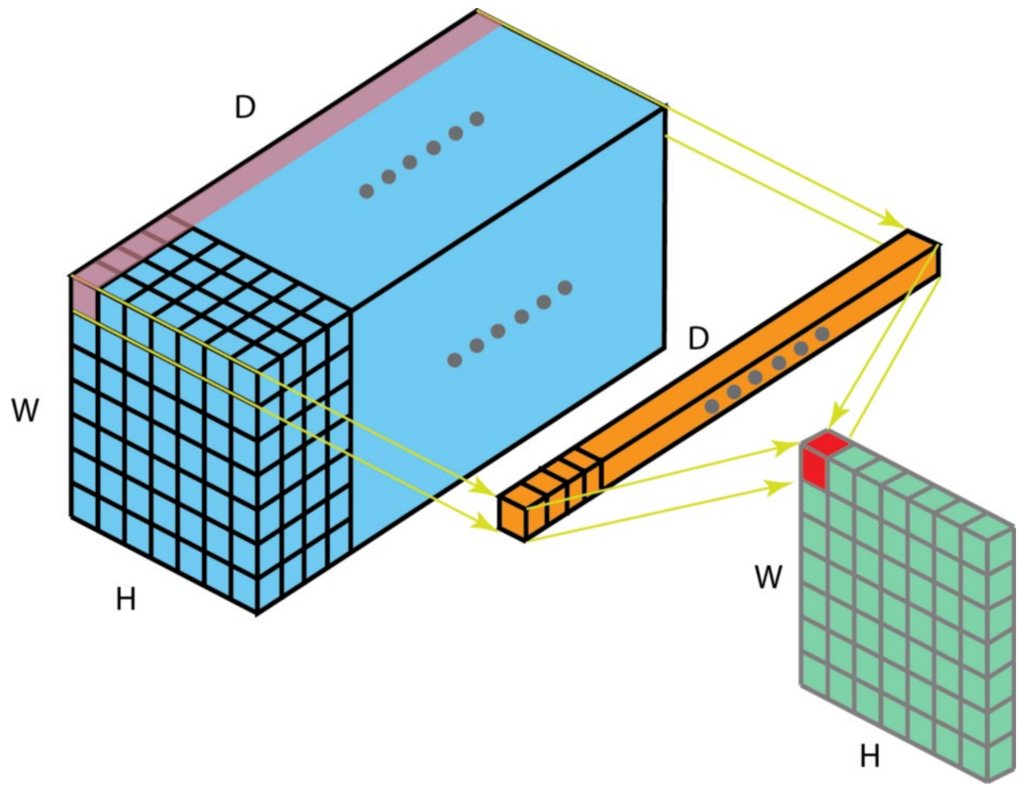
[Depthwise separable convolutions for machine learning - Eli Bendersky's website (thegreenplace.net)]

1x1 Convolution (a.k.a Pointwise Convolution)

3x3 Convolution



1x1 Convolution



[A Comprehensive Introduction to Different Types of Convolutions in Deep Learning | by Kunlun Bai | Towards Data Science]

Separable Convolutions

- Spatial separable convolution

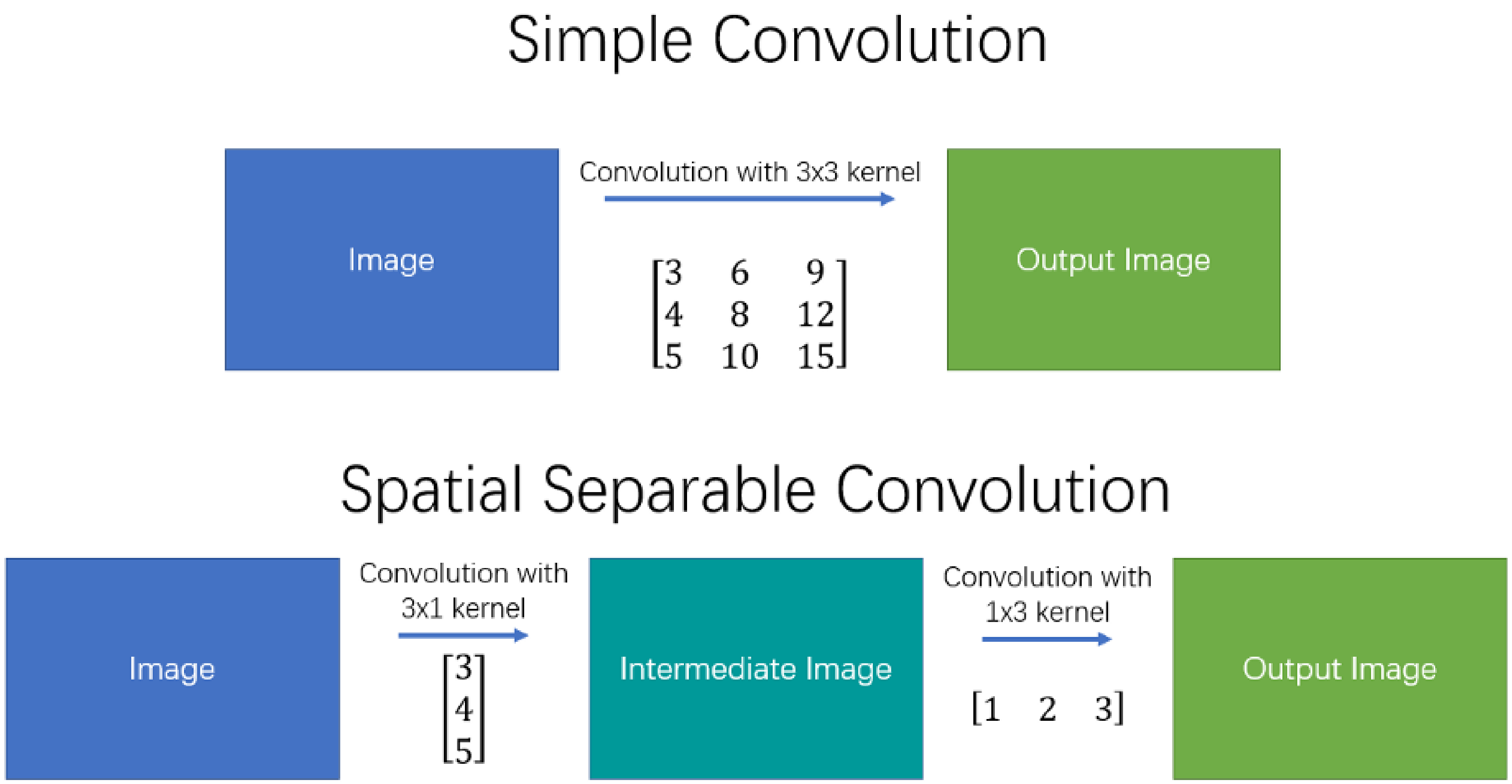
$$\begin{bmatrix} 3 & 6 & 9 \\ 4 & 8 & 12 \\ 5 & 10 & 15 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

A Basic Introduction to Separable Convolutions | by Chi-Feng Wang | Towards Data Science

Separable Convolutions

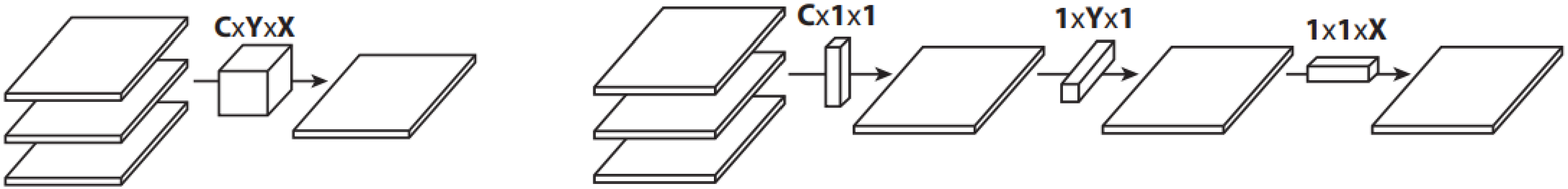
- Spatial separable convolution



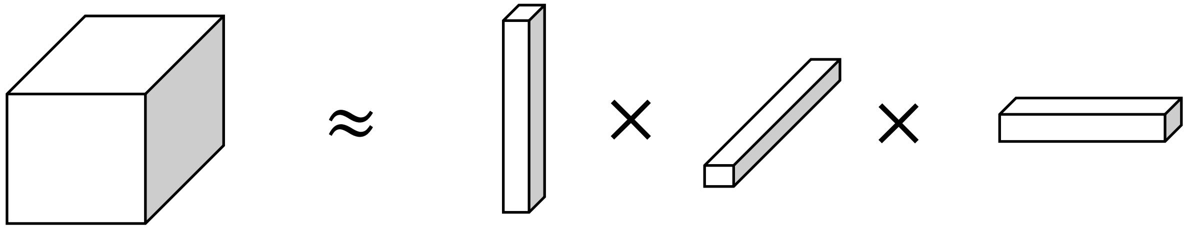
[A Basic Introduction to Separable Convolutions | by Chi-Feng Wang | Towards Data Science]

Separable Convolutions

- Flattened convolution

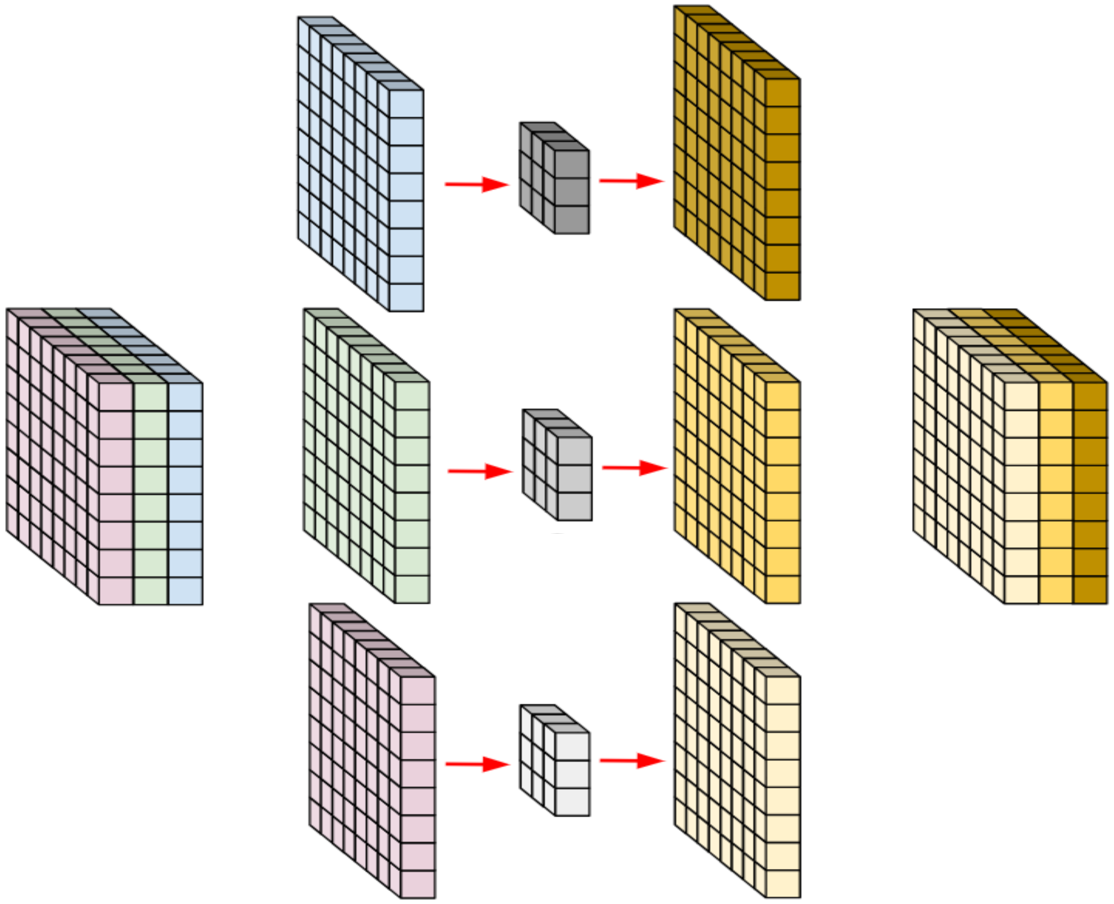


(a) 3D convolution (b) 1D convolutions over different directions



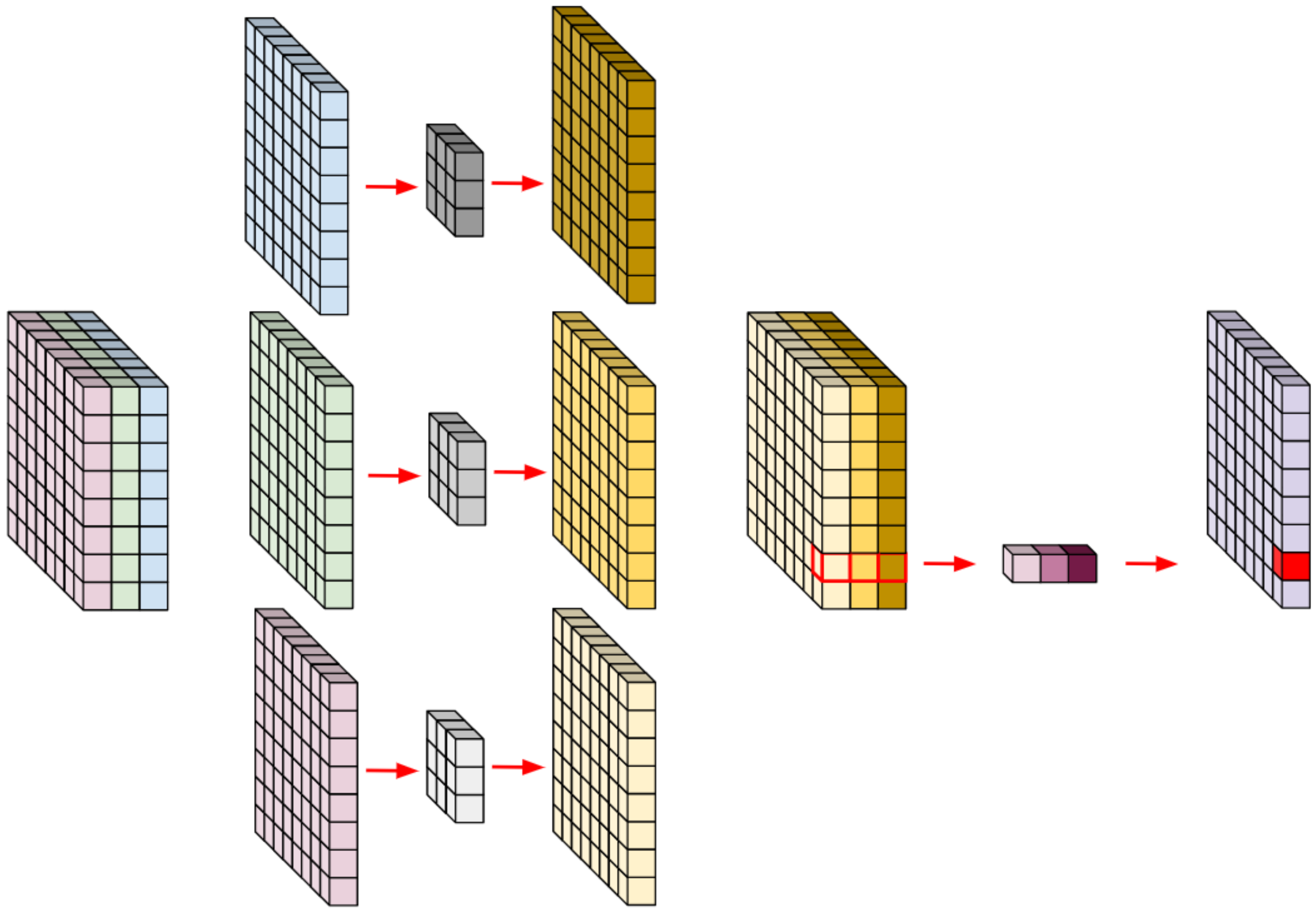
Flattened convolutional neural networks for feedforward acceleration, Jin et al, ICLR Workshop 2015

Depthwise Convolution



[Depthwise separable convolutions for machine learning - Eli Bendersky's website (thegreenplace.net)]

Depthwise Separable Convolution



[Depthwise separable convolutions for machine learning - Eli Bendersky's website (thegreenplace.net)]

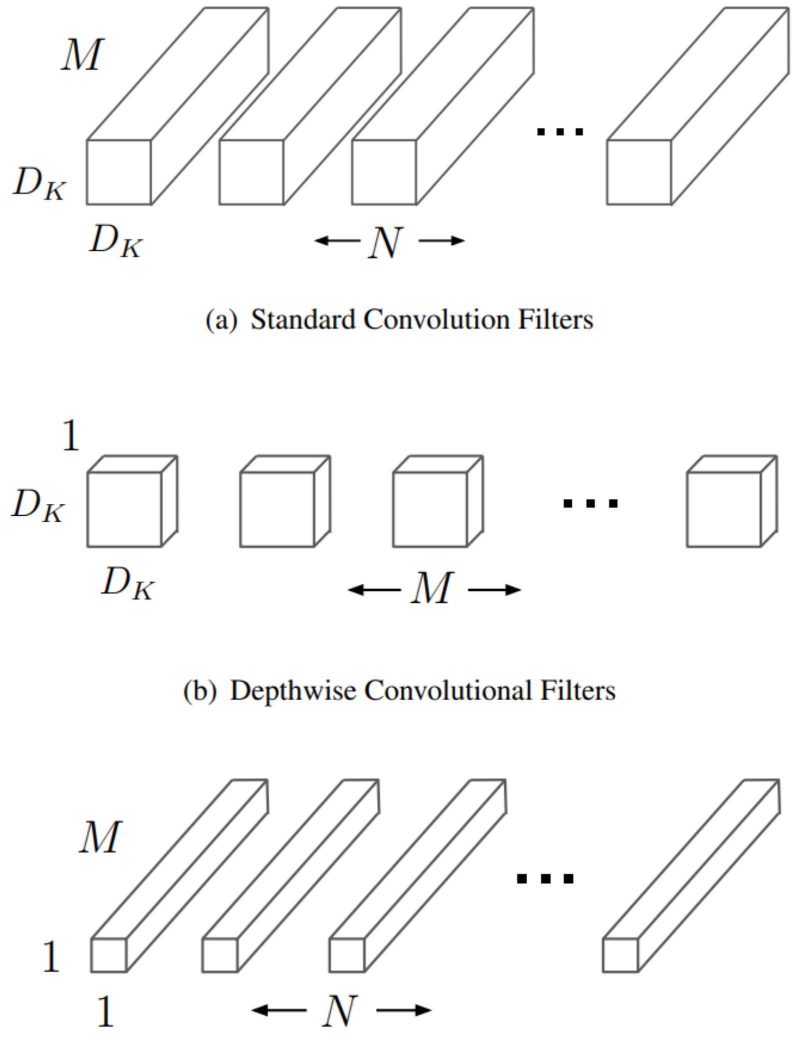
Depthwise Separable Convolutional Layer

- # parameters

M: 128 The number of input channels
N: 128 The number of output channels
D_K: 3 The size of a filter
D_F: 64 The size of a feature map

Standard Convolution

Depthwise Separable Convolution



MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, Howard et al, CVPR 2017

Depthwise Separable Convolutional Layer

- # parameters

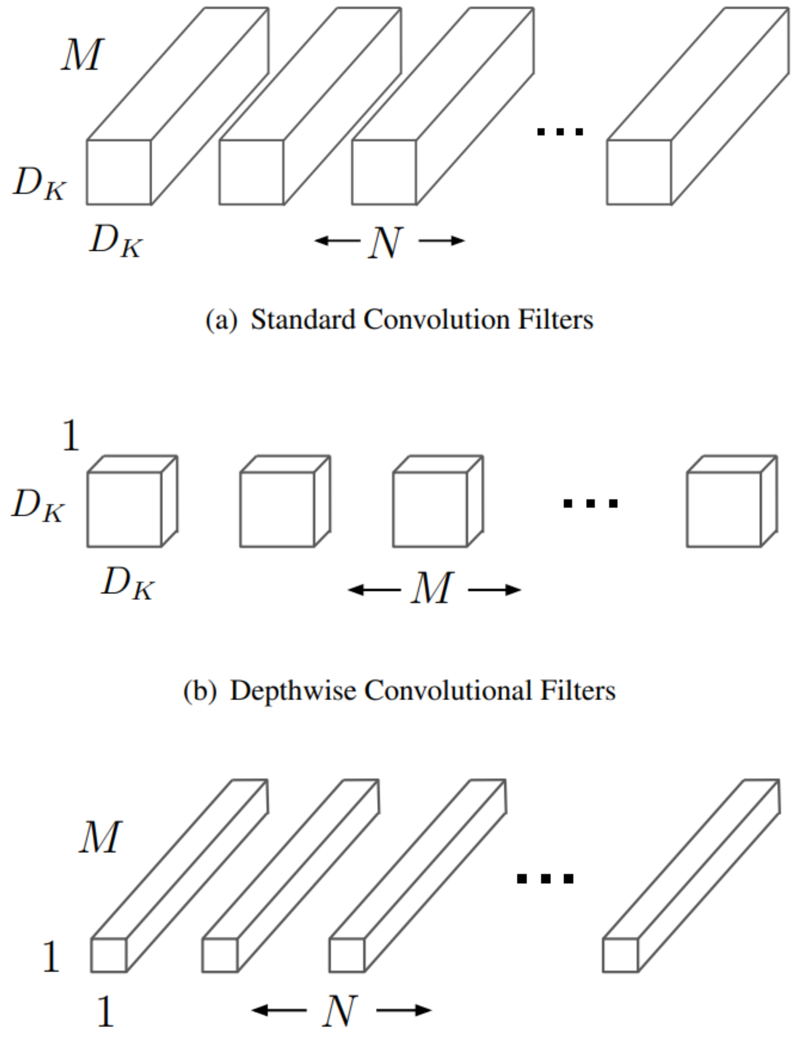
M: 128	The number of input channels
N: 128	The number of output channels
D_K: 3	The size of a filter
D_F: 64	The size of a feature map

Standard Convolution

$$D_K D_K M N = 147{,}456$$

Depthwise Separable Convolution

$$D_K D_K M + M N = 17{,}536$$



MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, Howard et al, CVPR 2017

Depthwise Separable Convolutional Layer

- Computational Cost (FLOPS)

M: 128	The number of input channels
N: 128	The number of output channels
D_K: 3	The size of a filter
D_F: 64	The size of a feature map

Standard Convolution

Depthwise Separable Convolution

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, Howard et al, CVPR 2017

Depthwise Separable Convolutional Layer

- Computational Cost (FLOPS)

M: 128	The number of input channels
N: 128	The number of output channels
D_K: 3	The size of a filter
D_F: 64	The size of a feature map

Standard Convolution

$$D_K D_K M N D_F D_F$$
$$= 603{,}979{,}776$$

Depthwise Separable Convolution

$$D_K D_K M D_F D_F + M N D_F D_F$$
$$= 71{,}827{,}456$$

$$\frac{D_K D_K M D_F D_F + M N D_F D_F}{D_K D_K M N D_F D_F} = \frac{1}{N} + \frac{1}{D_K^2}$$

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, Howard et al, CVPR 2017

MobileNets

- ConvNet w/ depthwise separable convolution

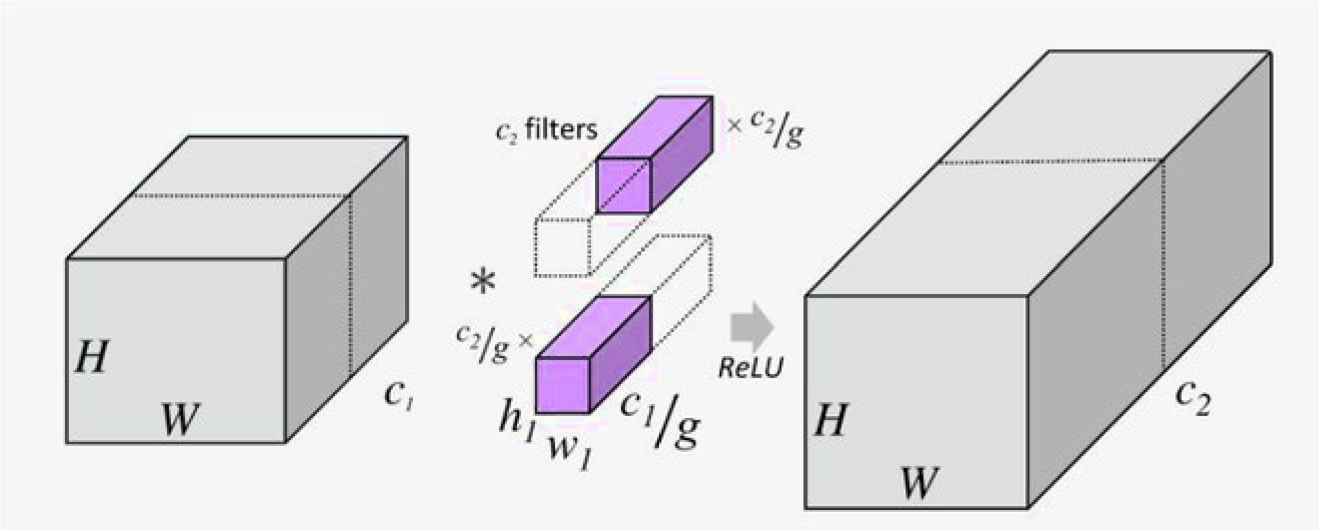
Table 4. Depthwise Separable vs Full Convolution MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, Howard et al, CVPR 2017

Grouped Convolution

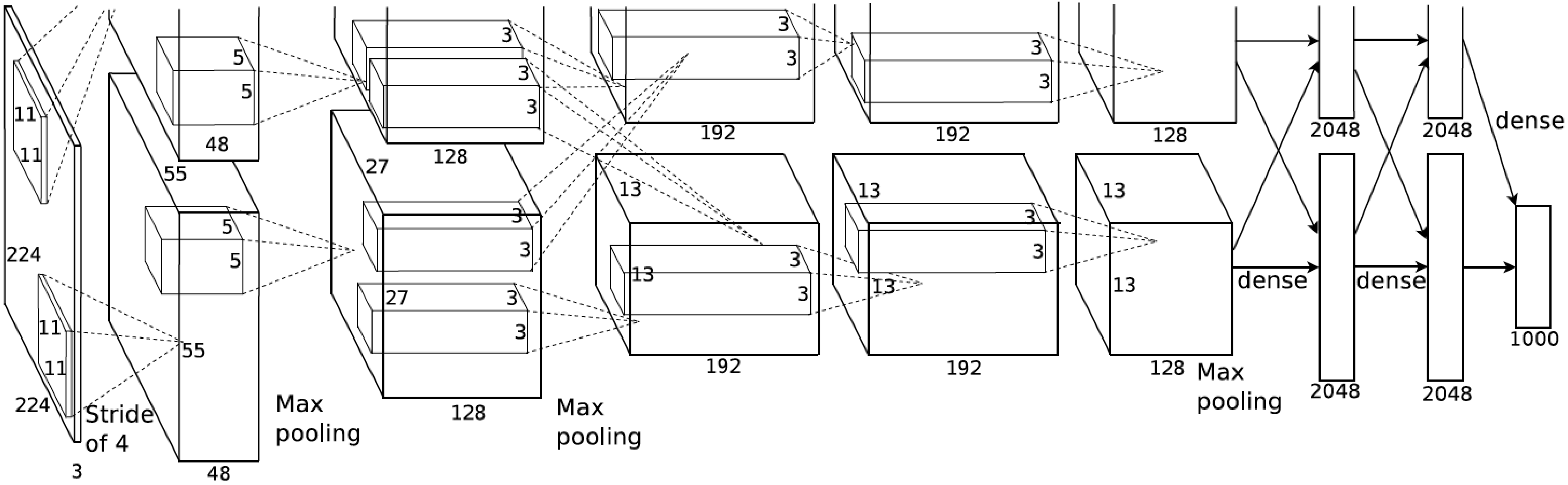
- Convolutions in parallel
  - It was first Introduced in AlexNet to utilize the 2 GPUs distributed training
  - 1.5GB x 2 = 3GB



[Grouped Convolution - GeeksforGeeks]

Grouped Convolution

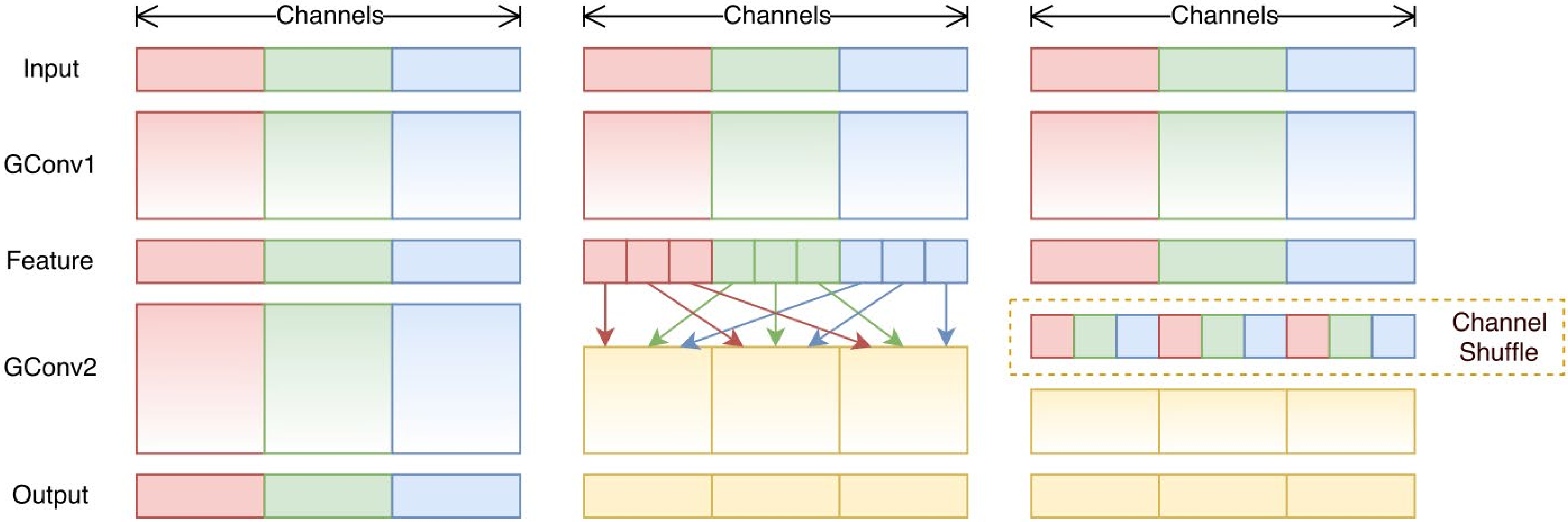
- Convolutions in parallel
  - It was first Introduced in AlexNet to utilize the 2 GPUs distributed training
  - 1.5GB x 2 = 3GB



ImageNet Classification with Deep Convolutional Neural Networks, Krizhevsky et al, NeurIPS 2012

Shuffled Convolution

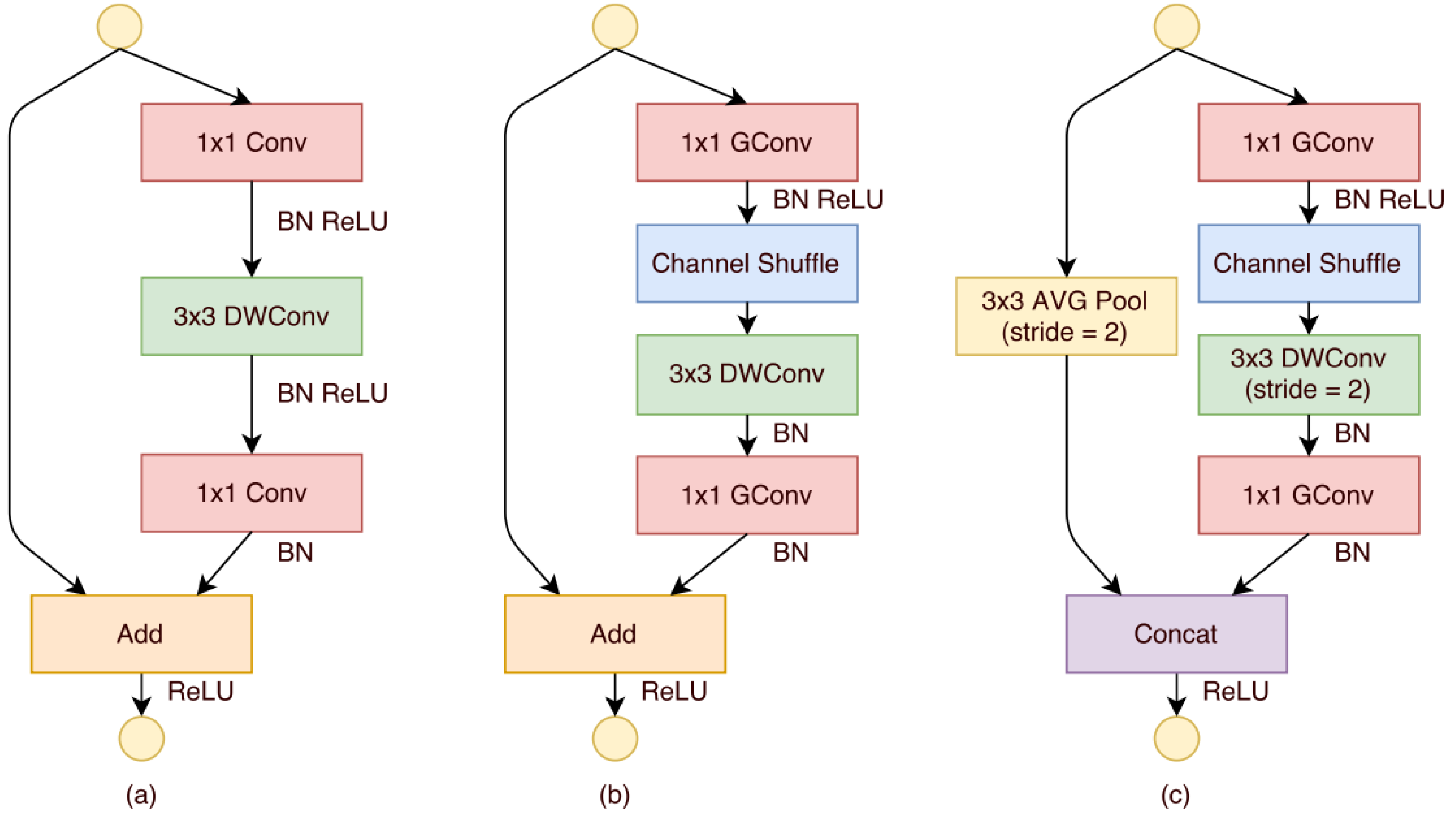
- To eliminate a side effect of the grouped convolutions
  - The outputs from a certain channel are only derived from a small fraction of input channels



ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices, Zhange et al, CVPR 2018

Shuffled Convolution

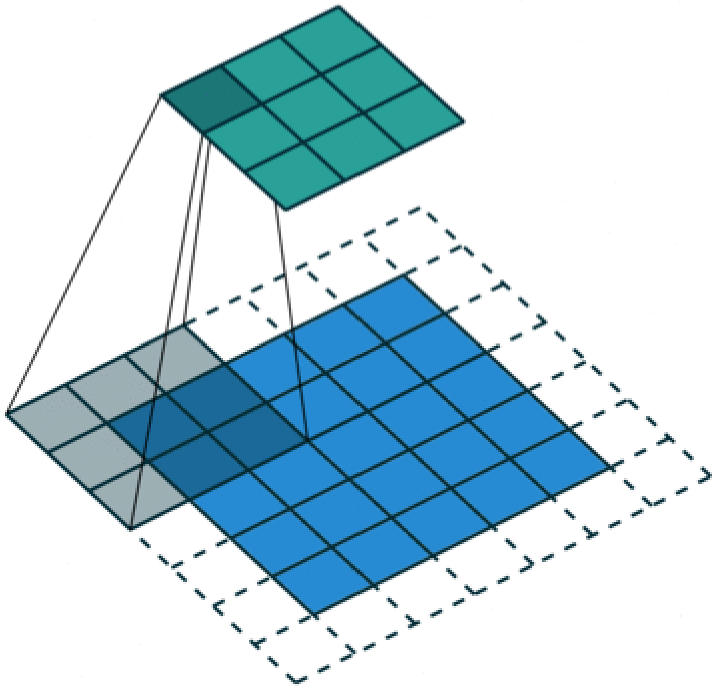
- ShuffleNet



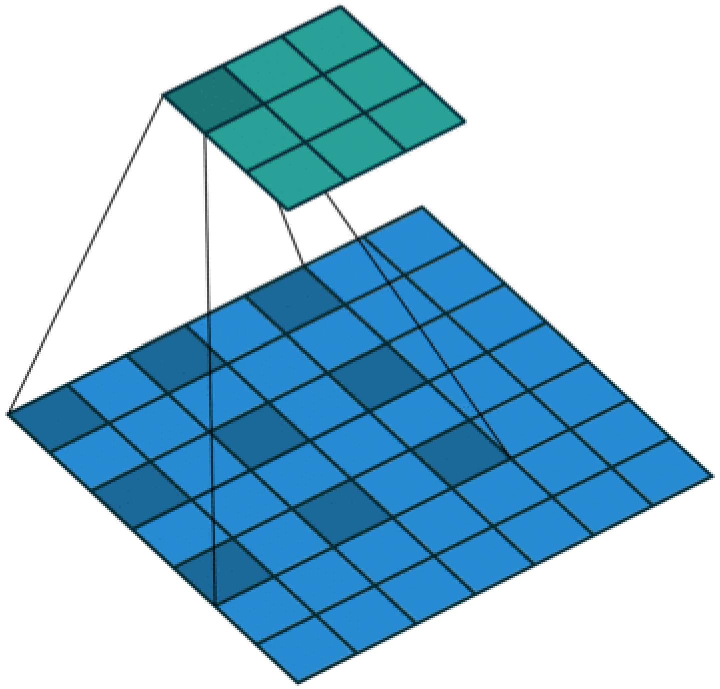
ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices, Zhange et al, CVPR 2018

Dilated Convolution

- To aggregate multi-scale contextual information without losing resolution



Standard Convolution (l=1)



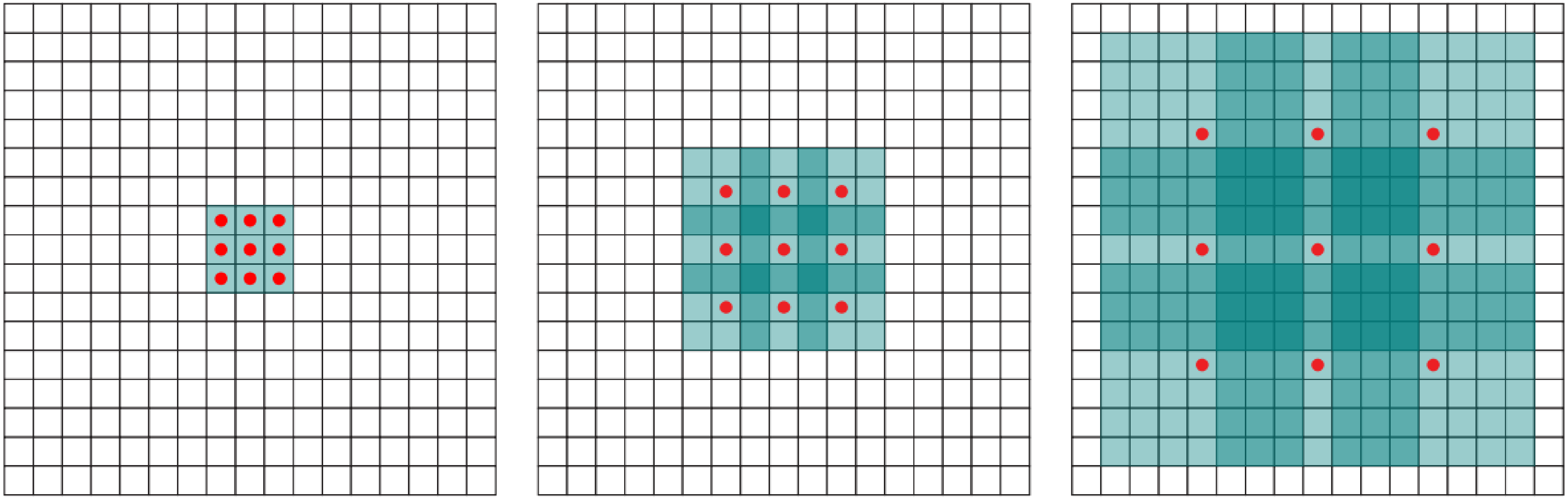
Dilated Convolution (l=2)

[Review: DilatedNet — Dilated Convolution (Semantic Segmentation) | by Sik-Ho Tsang | Towards Data Science]

Dilated Convolution

- To aggregate multi-scale contextual information without losing resolution

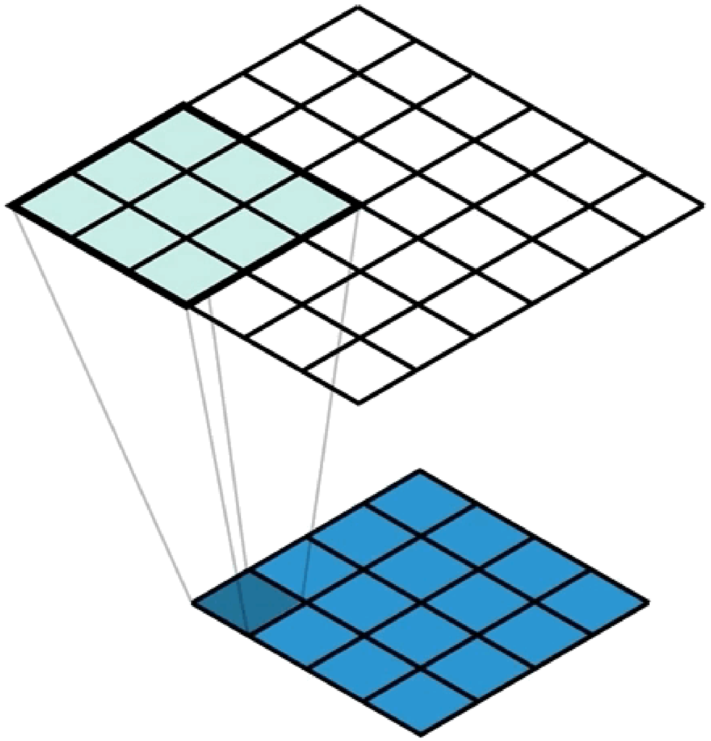
Receptive Fields



$F_1 = \mathrm{DilatedConv}(F_0, l = 1)$ $F_2 = \mathrm{DilatedConv}(F_1, l = 2)$ $F_3 = \mathrm{DilatedConv}(F_2, l = 4)$

Multi-Scale Context Aggregation by Dilated Convolutions, Yu et al, ICLR 2016

Transposed Convolution



[Transposed Convolutions explained with… MS Excel! | by Thom Lane | Apache MXNet | Medium]

ImageNet Large Scale Visual Recognition Challenge

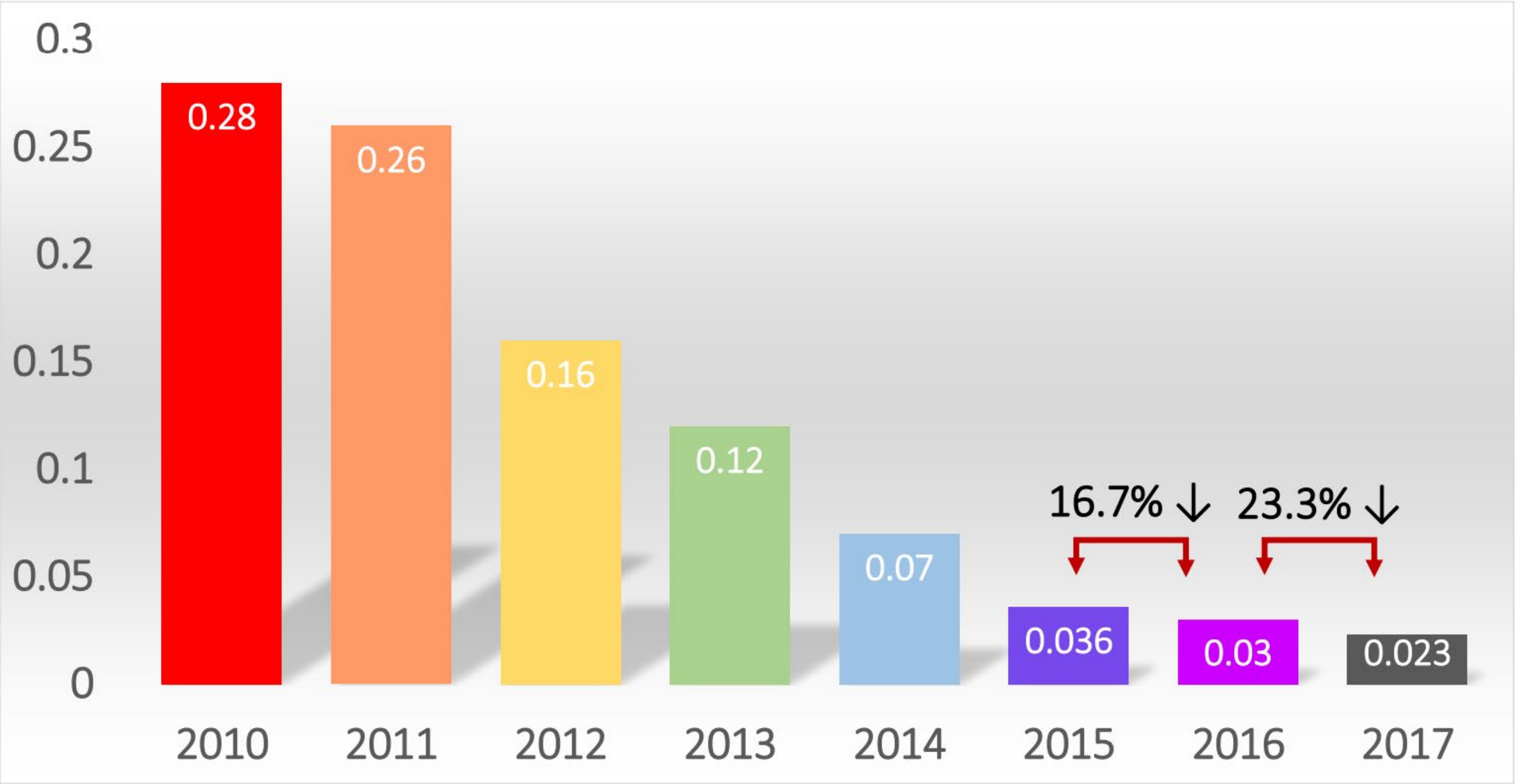
(ILSVRC)

ILSVRC

- ImageNet is an image database organized according to the WordNet hierarchy (nouns)
  - 1000 object classes
  - About 1.2M training images, 50K validation images, 100K test images

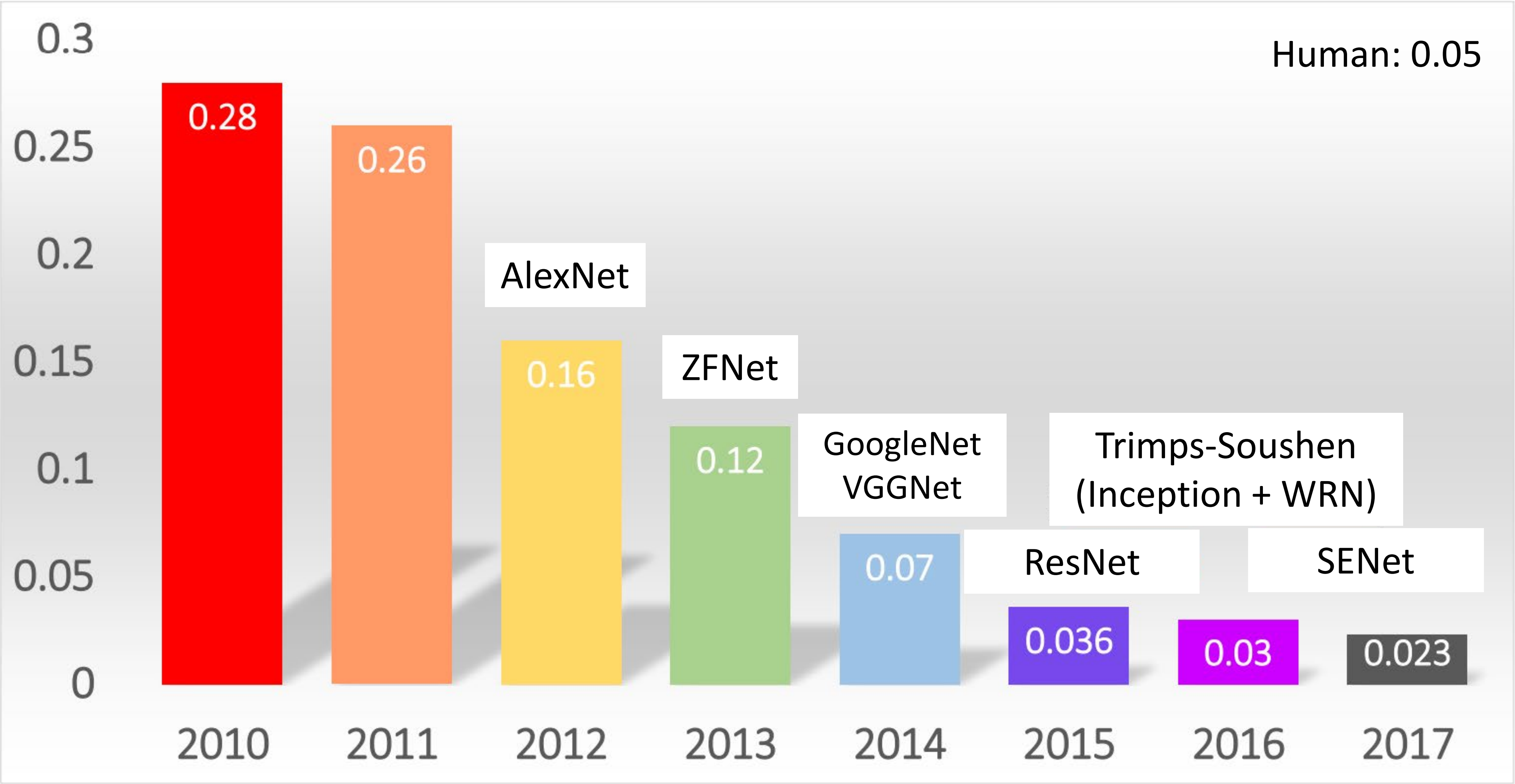
- The ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
  - 8 years history (2010 – 2017)
  - It was the most powerful driving force to facilitate deep learning research

Classification Results



Beyond ILSVRC workshop 2017 (image-net.org)

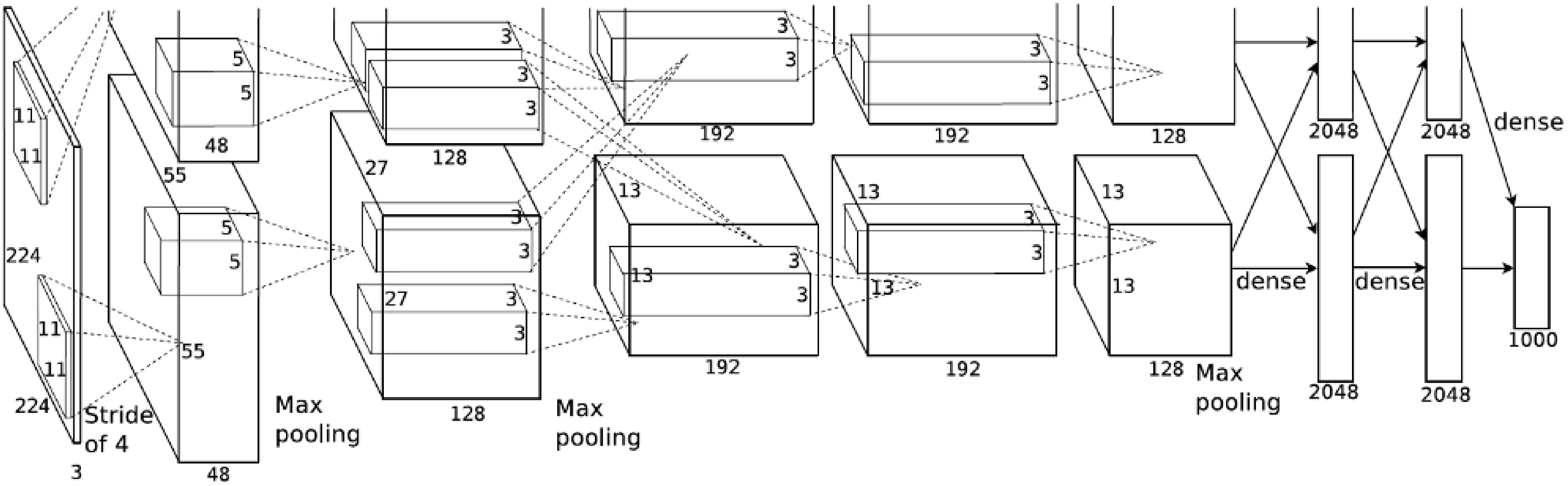
Classification Results



[Beyond ILSVRC workshop 2017 (image-net.org)]

AlexNet

- The winner of ILSVRC 2012
- It changed the entire computer vision research

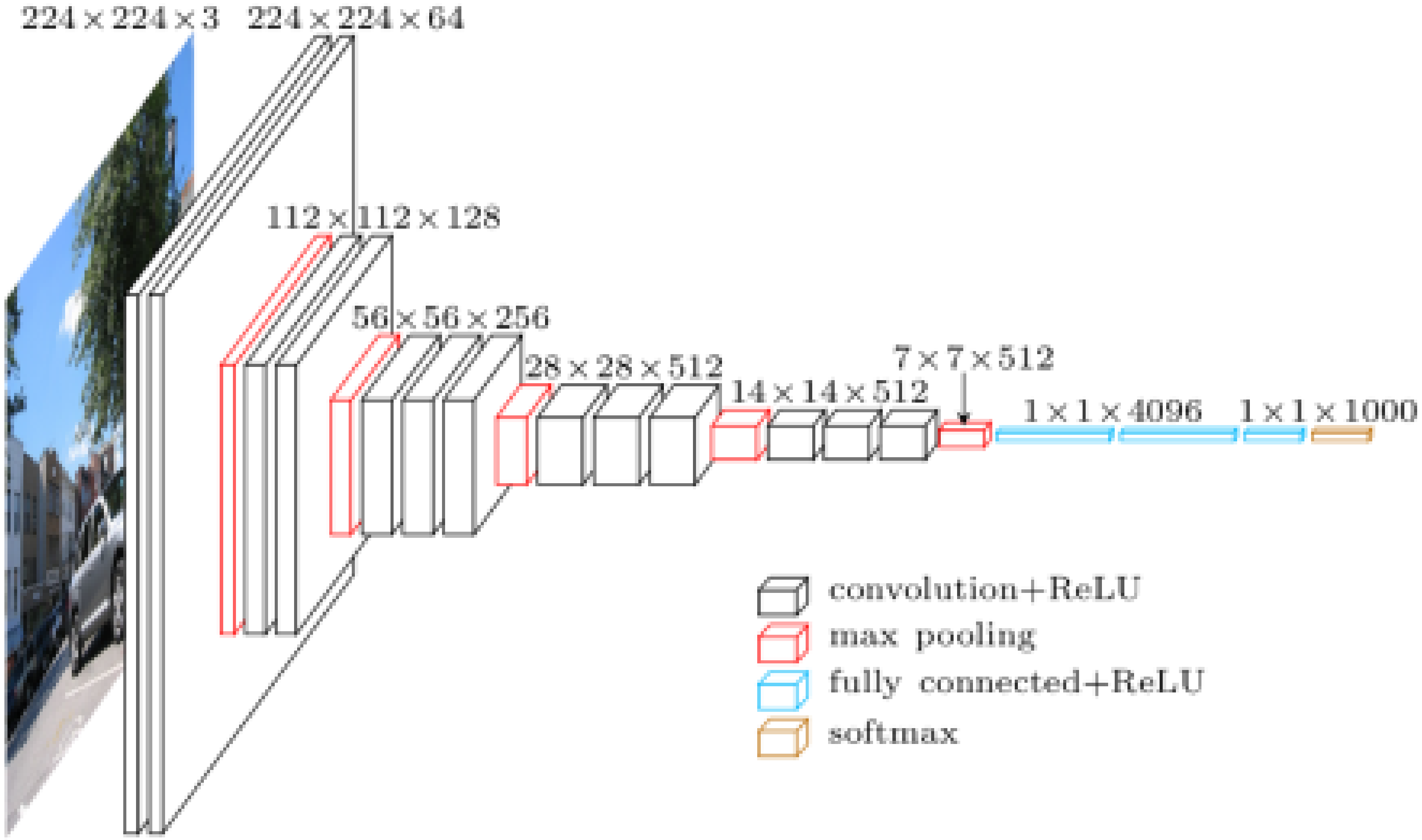


ZFNet

- The winner of ILSVRC 2013
- The network architectures were developed by using the visualization techniques
  - Visualizing and Understanding Convolutional Networks, Zeiler et al, ECCV 2014

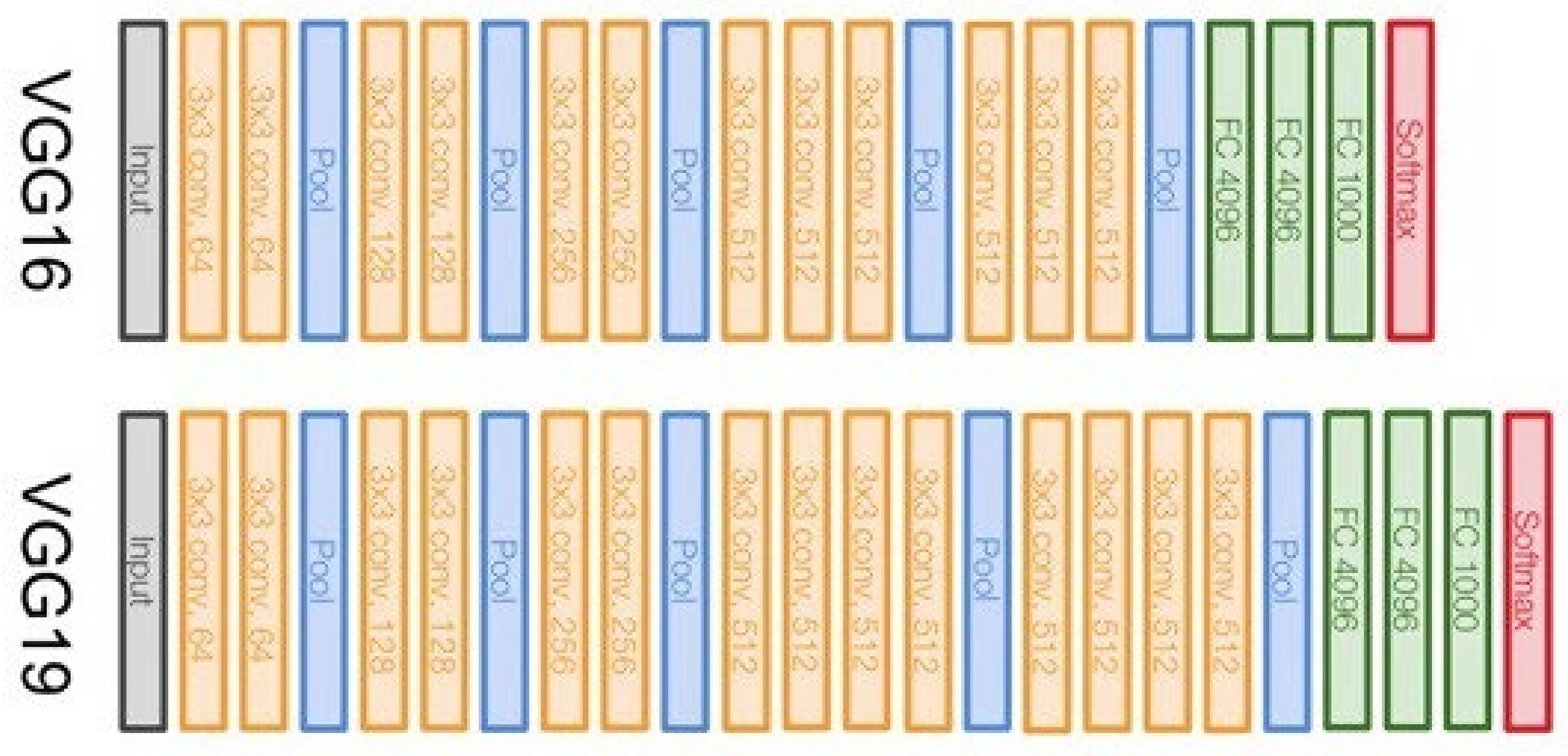
- Reduced the 1^{st} layer filter size from 11x11 to 7x7
- 1^{st} layer stride from 4 -> 2

VGGNet



[Architecture comparison of AlexNet, VGGNet, ResNet, Inception, DenseNet | by Khush Patel | Towards Data Science]

VGGNet

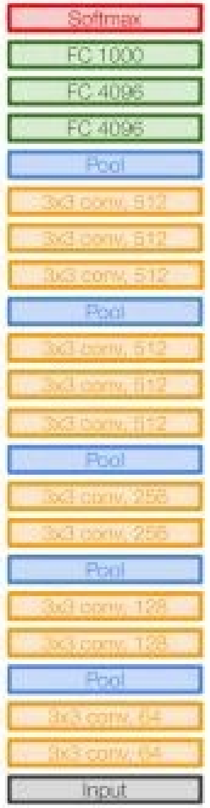


[Architecture comparison of AlexNet, VGGNet, ResNet, Inception, DenseNet | by Khush Patel | Towards Data Science]

VGGNet

INPUT: [224x224x3] memory: 224*224*3=150K params: 0 (not counting biases)
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= 96MB / image (for a forward pass)
TOTAL params: 138M parameters



VGG16

[Architecture comparison of AlexNet, VGGNet, ResNet, Inception, DenseNet | by Khush Patel | Towards Data Science]

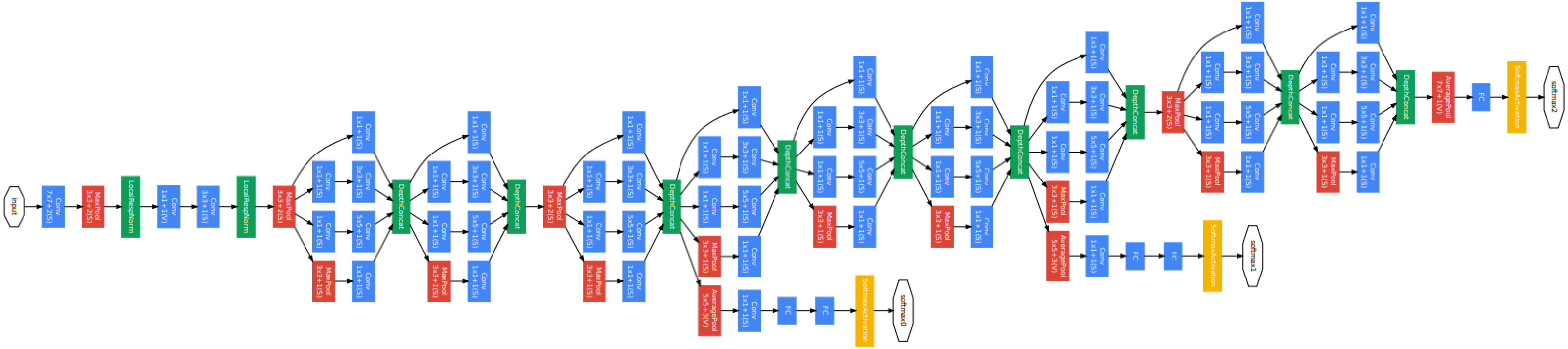
GoogLeNet

- Winner of ISLVRC 2014
- Also called 'Inception'

Max pooling

Concatenation

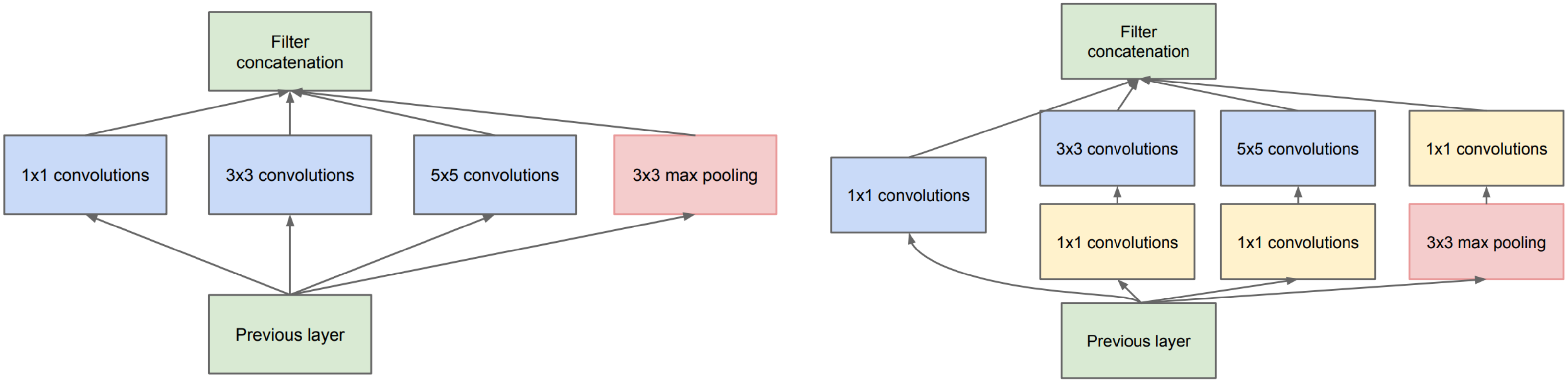
Convolution



Going deeper with convolutions, Szegedy et al, CVPR 2015

GoogLeNet

- Inception module



(a) Inception module, naïve version

(b) Inception module with dimensionality reduction

Going deeper with convolutions, Szegedy et al, CVPR 2015

GoogLeNet

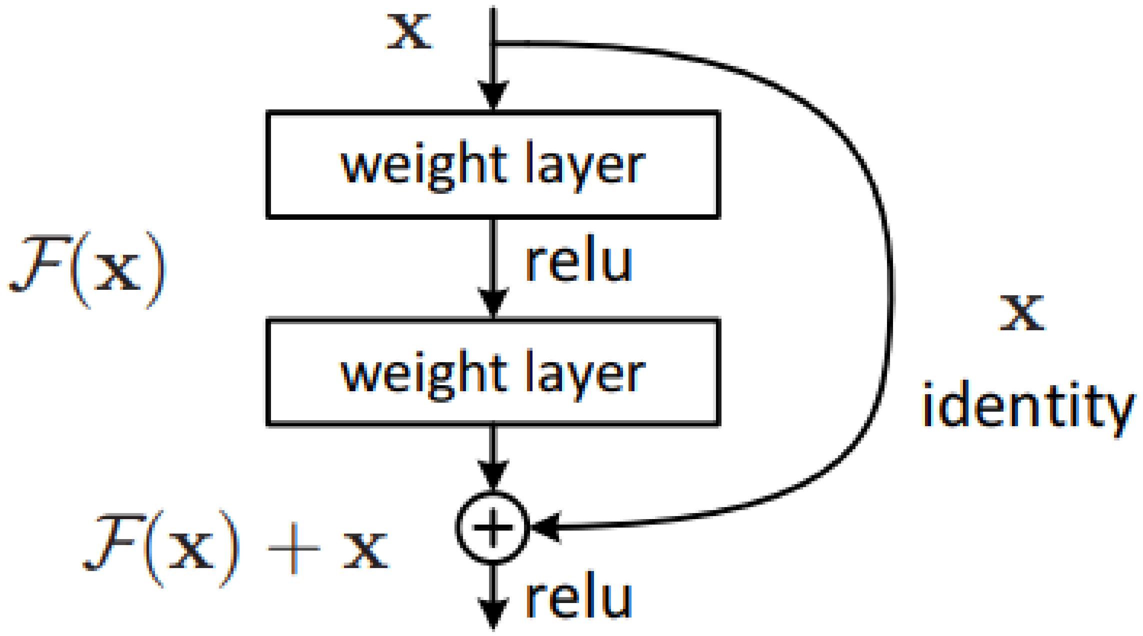
- ILSVRC 2014 classification results

Team	Year	Place	Error (top-5)	Uses external data
SuperVision	2012	1st	16.4%	no
SuperVision	2012	1st	15.3%	Imagenet 22k
Clarifai	2013	1st	11.7%	no
Clarifai	2013	1st	11.2%	Imagenet 22k
MSRA	2014	3rd	7.35%	no
VGG	2014	2nd	7.32%	no
GoogLeNet	2014	1st	6.67%	no

Going deeper with convolutions, Szegedy et al, CVPR 2015

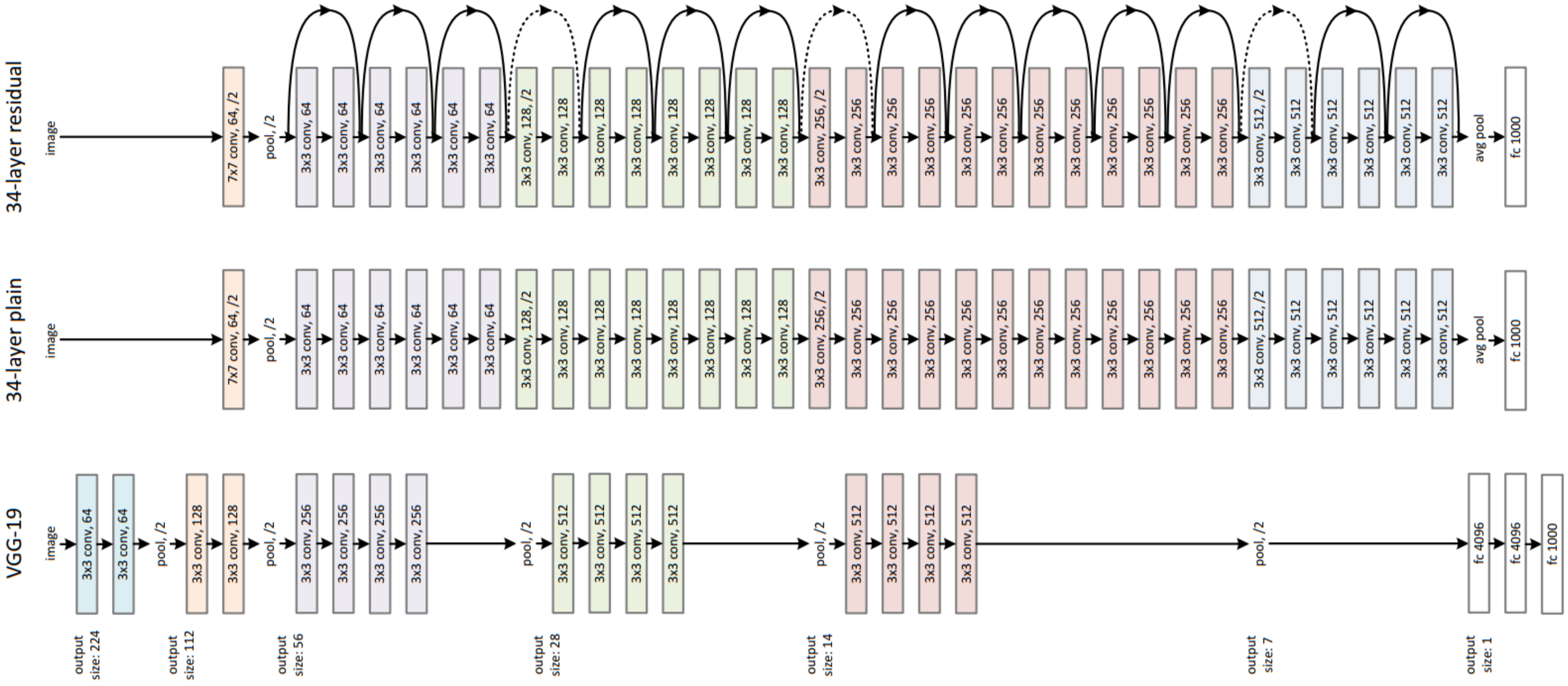
ResNet

- The winner of ILSVRC 2015
- Residual building block



Deep residual learning for image recognition, He et al, CVPR 2016

ResNet



Deep residual learning for image recognition, He et al, CVPR 2016

ResNet

- Training on ImageNet

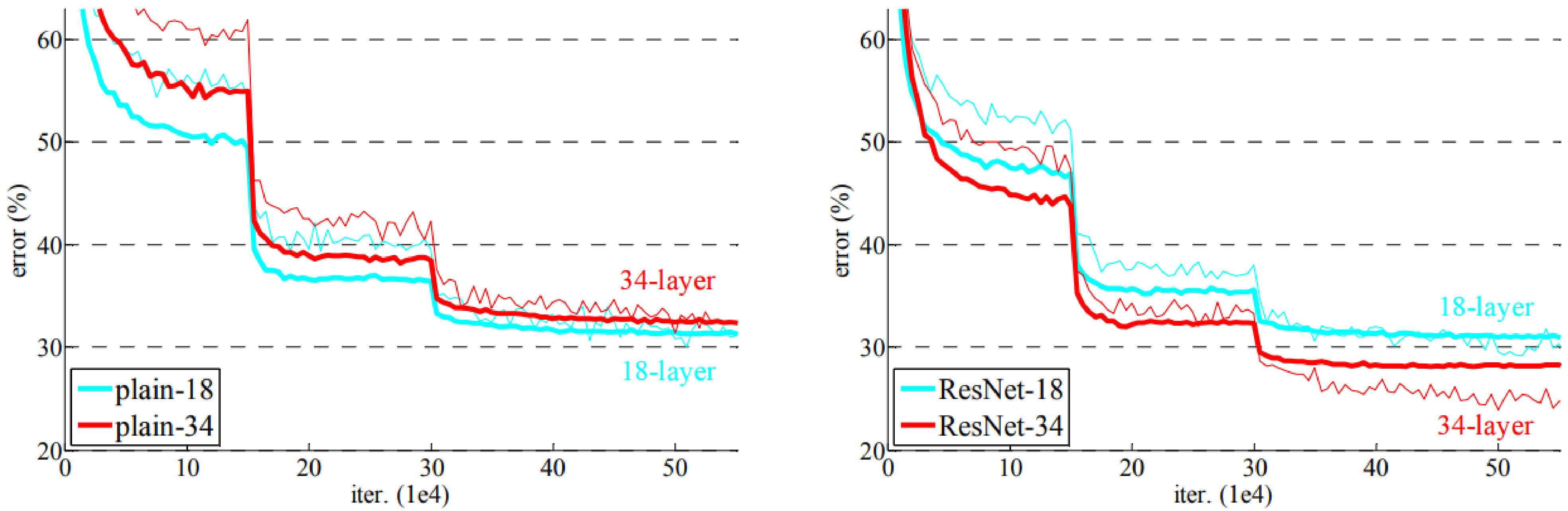


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

Deep residual learning for image recognition, He et al, CVPR 2016

ResNet

- ILSVRC 2015 classification results

method	top-5 err. (**test**)
VGG [40] (ILSVRC'14)	7.32
GoogLeNet [43] (ILSVRC'14)	6.66
VGG [40] (v5)	6.8
PReLU-net [12]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

Table 5. Error rates (%) of **ensembles**. The top-5 error is on the test set of ImageNet and reported by the test server.

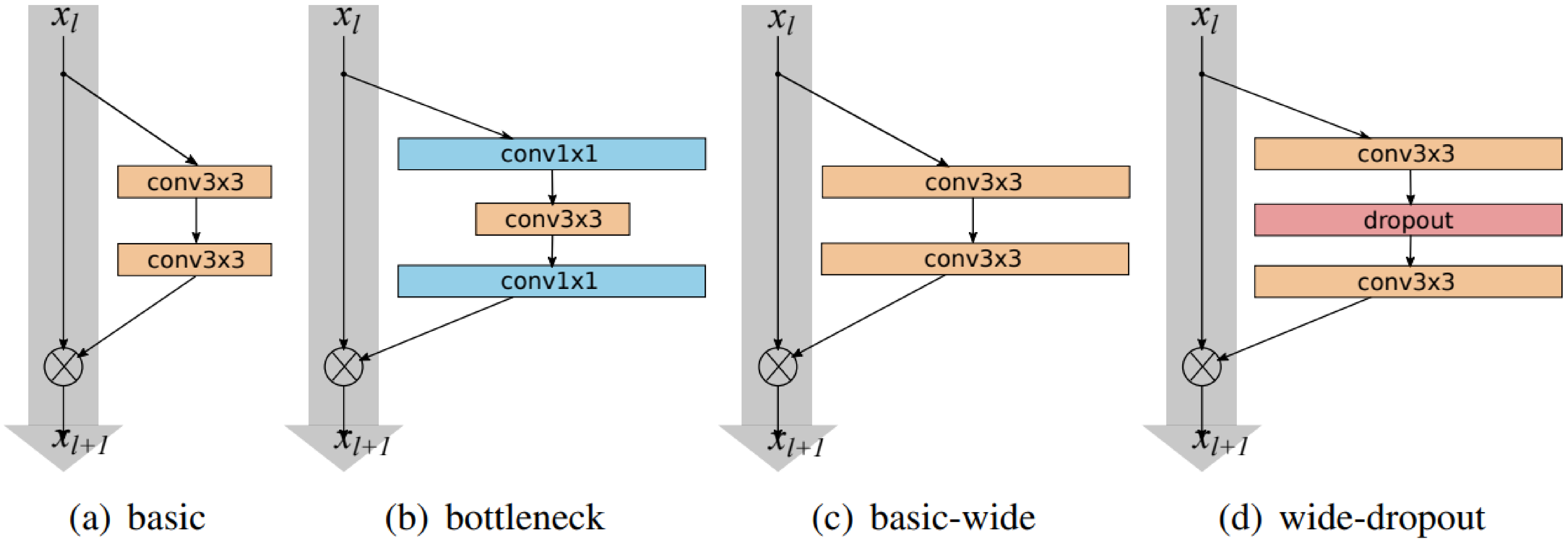
Deep residual learning for image recognition, He et al, CVPR 2016

Wide Residual Networks (WRNs)

- Diminishing feature reuse
  - The circuit complexity theory says that shallow circuits can require exponentially more parameters than deeper ones.
  - However, in the residual block w/ identity mapping, there is nothing to force the gradients to go through residual block weights
  - It is possible that there is there is either only a few blocks that learn useful representations or many blocks share very little information with small contribution
- ‘Widening’ of ResNet provides a much more effective way of improving performance
  - 50 times less layers and being more than 2 times faster

Wide residual networks, Zagoruyko et al, BMVC 2017

Wide Residual Networks (WRNs)



Wide residual networks, Zagoruyko et al, BMVC 2017

Wide Residual Networks (WRNs)

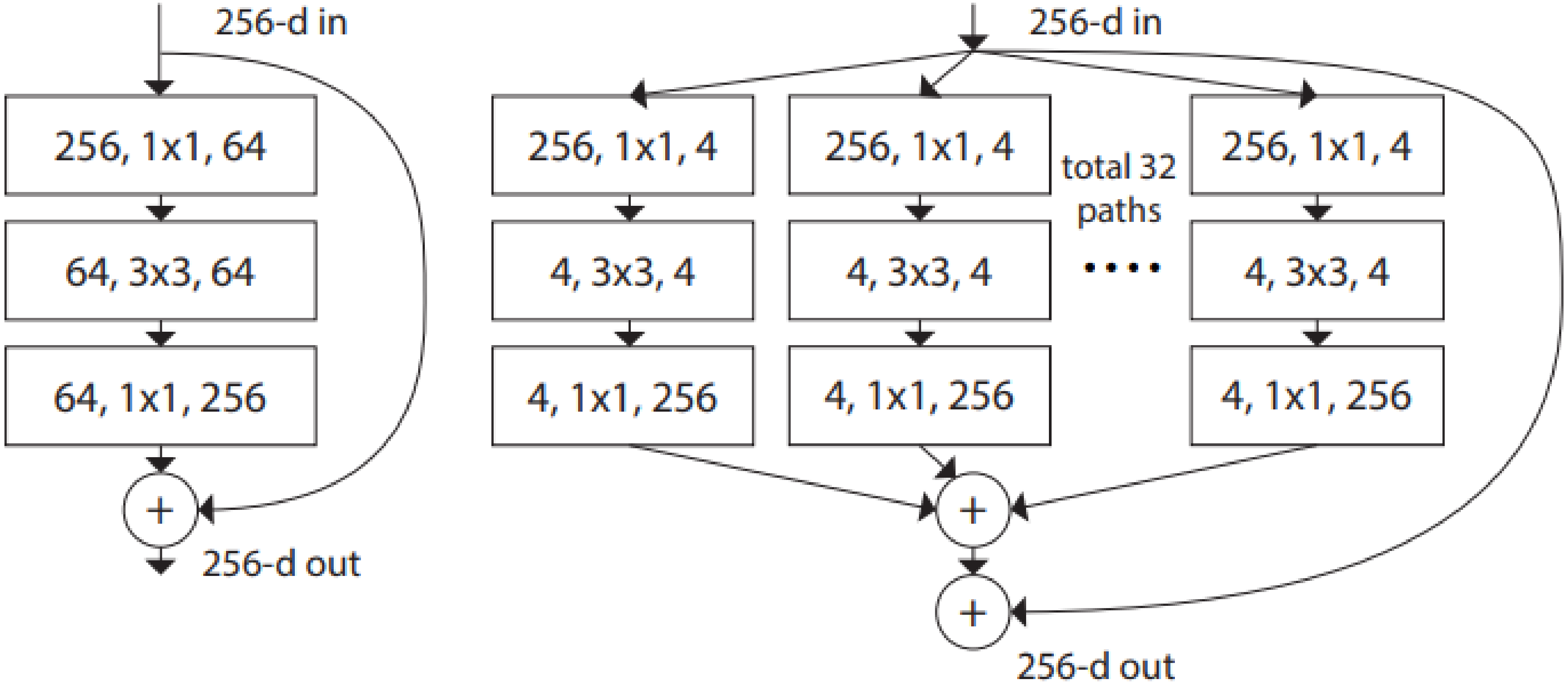
- ILSVRC classification results (single crop)

Model	top-1 err, %	top-5 err, %	#params	time/batch 16
ResNet-50	24.01	7.02	25.6M	49
ResNet-101	22.44	6.21	44.5M	82
ResNet-152	22.16	6.16	60.2M	115
WRN-50-2-bottleneck	21.9	6.03	68.9M	93
pre-ResNet-200	21.66	5.79	64.7M	154

Wide residual networks, Zagoruyko et al, BMVC 2017

ResNeXt

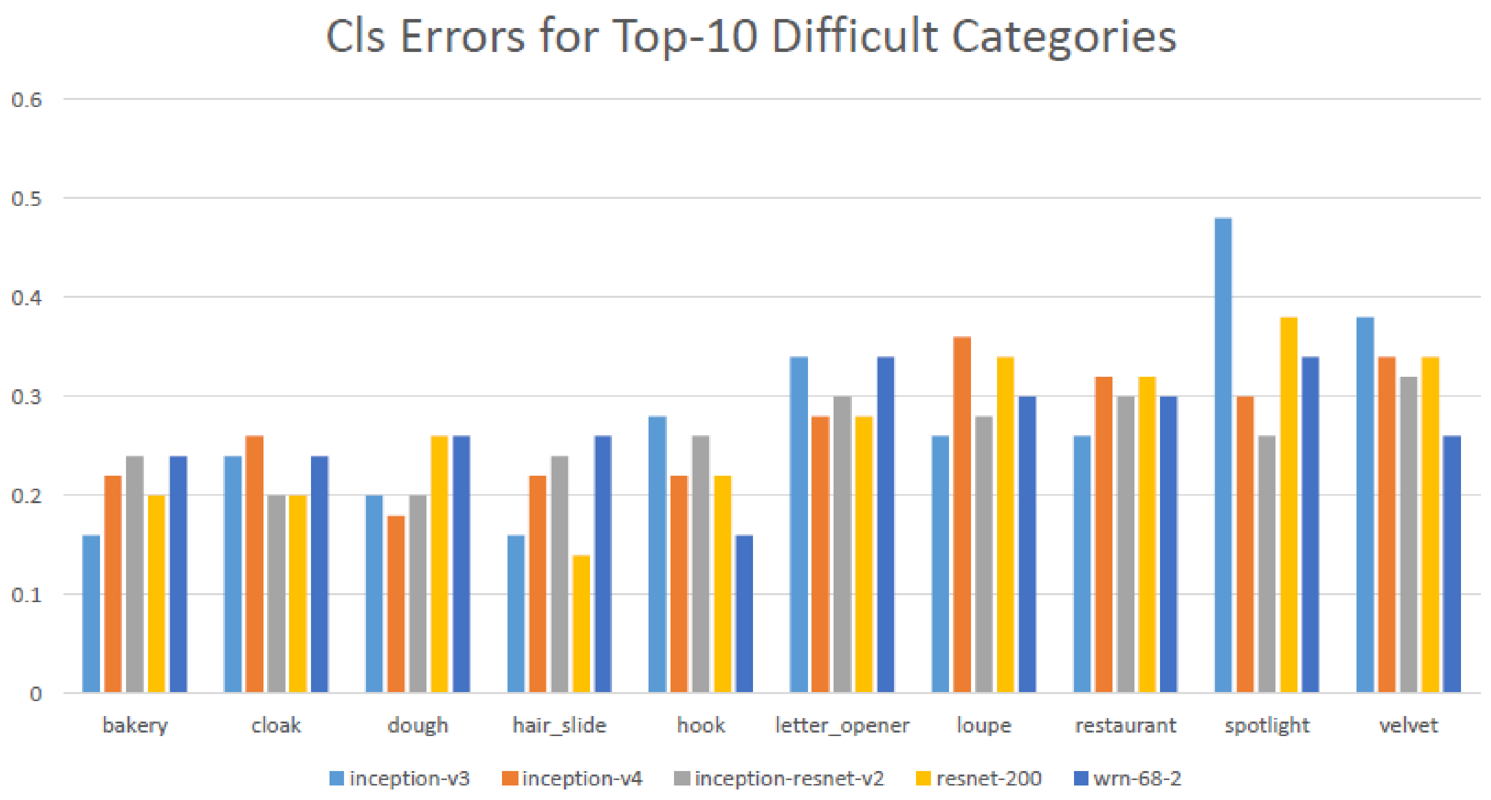
- The second ranked in ILSVRC 2016
  - ‘Cardinality’ matters (the number of transformations in the layers)
  - next dimension -> cardinality



Aggregated residual transformations for deep neural networks, Sie et al, BMVC 2017

Ensemble of Diverse Architecture

- The winner of ILSVRC 2016
  - Trimps-Soushen Team



[Review: Trimps-Soushen — Winner in ILSVRC 2016 (Image Classification) | by Sik-Ho Tsang | Towards Data Science]

Ensemble of Diverse Architecture

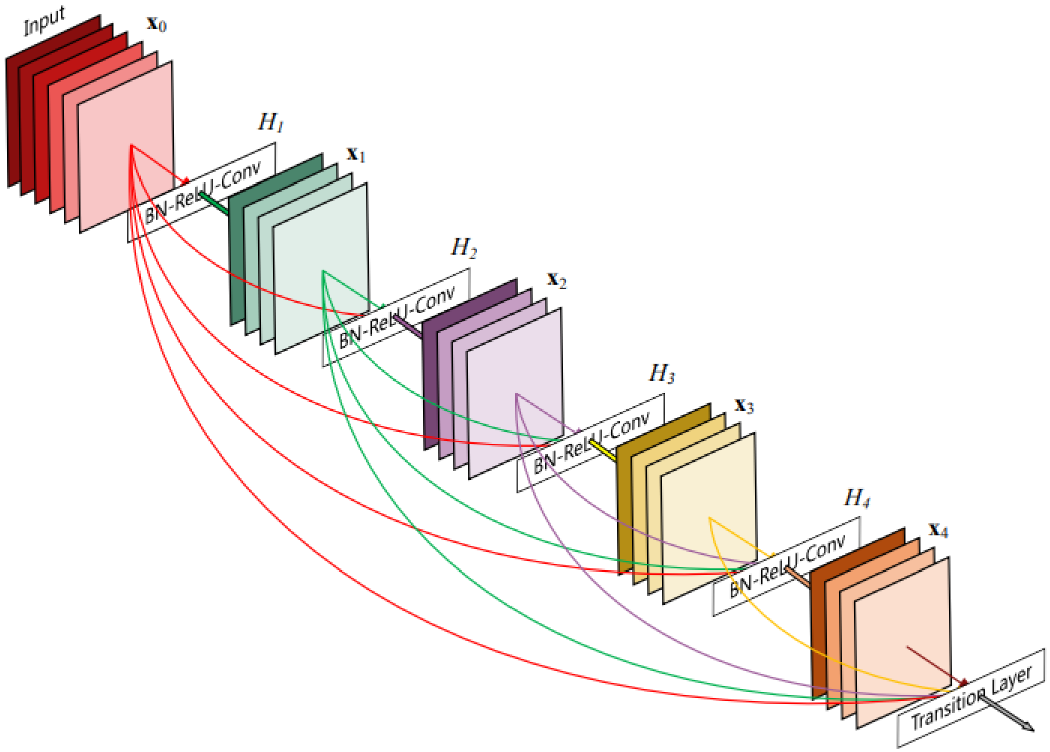
- ILSVRC classification results

	Inception-v3	Inception-v4	Inception-Resnet-v2	Resnet-200	Wrn-68-3	Fusion（Val.）	Fusion（Test）
Err. (%)	4.20	4.01	3.52	4.26	4.65	2.92 (-0.6)	2.99

[Review: Trimps-Soushen — Winner in ILSVRC 2016 (Image Classification) | by Sik-Ho Tsang | Towards Data Science]

DenseNet

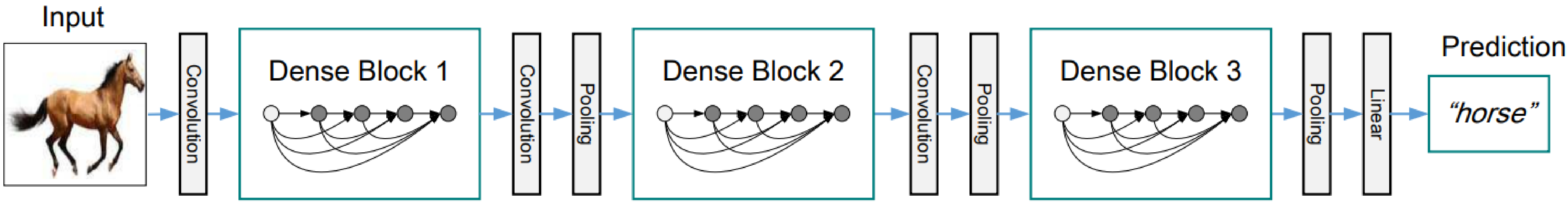
- The feature-maps of all preceding layers are used as inputs
  - For L layers block, $\frac{L(L+1)}{2}$ direct connections



Densely connected convolutional networks, Huang et al, CVPR 2017

DenseNet

- The feature-maps of all preceding layers are used as inputs
  - For L layers block, $\frac{L(L+1)}{2}$ direct connections



ResNet

$$\mathbf{x}_\ell = H_\ell(\mathbf{x}_{\ell-1}) + \mathbf{x}_{\ell-1}$$

DenseNet

$$\mathbf{x}_\ell = H_\ell([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}])$$

Densely connected convolutional networks, Huang et al, CVPR 2017

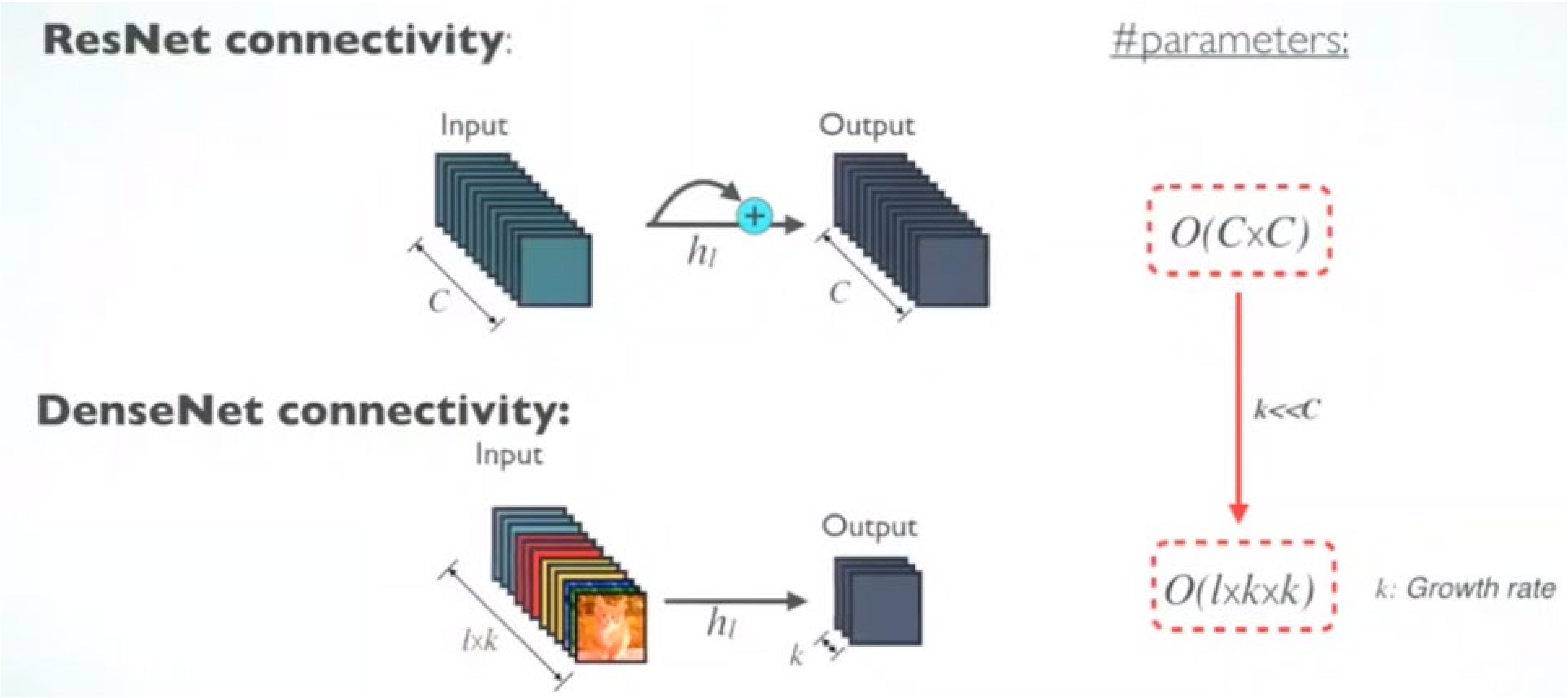
DenseNet

- Parameter efficient

[Review: DenseNet — Dense Convolutional Network (Image Classification) | by Sik-Ho Tsang | Towards Data Science]

DenseNet

- Parameter efficient



[Review: DenseNet — Dense Convolutional Network (Image Classification) | by Sik-Ho Tsang | Towards Data Science]

DenseNet

- Parameter efficient

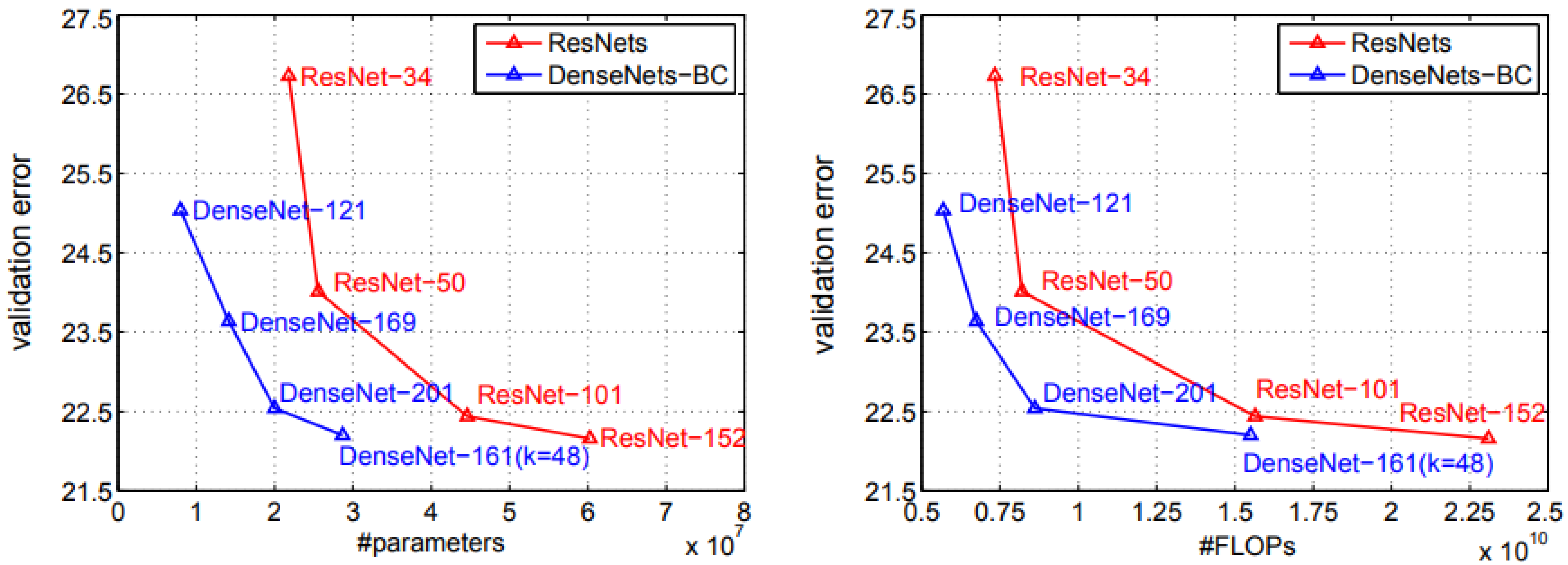
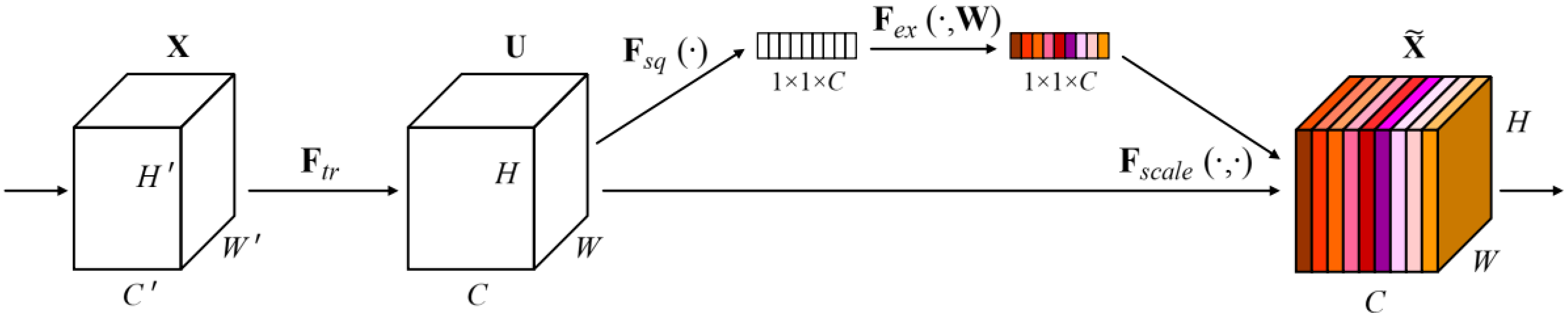


Figure 3: Comparison of the DenseNets and ResNets top-1 error rates (single-crop testing) on the ImageNet validation dataset as a function of learned parameters (*left*) and FLOPs during test-time (*right*).

Densely connected convolutional networks, Huang et al, CVPR 2017

Squeeze-and-Excitation Networks

- The winner of ILSVRC 2017
- SE Block
  - Easily integrated into popular convolutional modules, e.g. ResNet, Inception, etc.
  - A ‘light weight’ gating mechanism to model channel-wise relationships

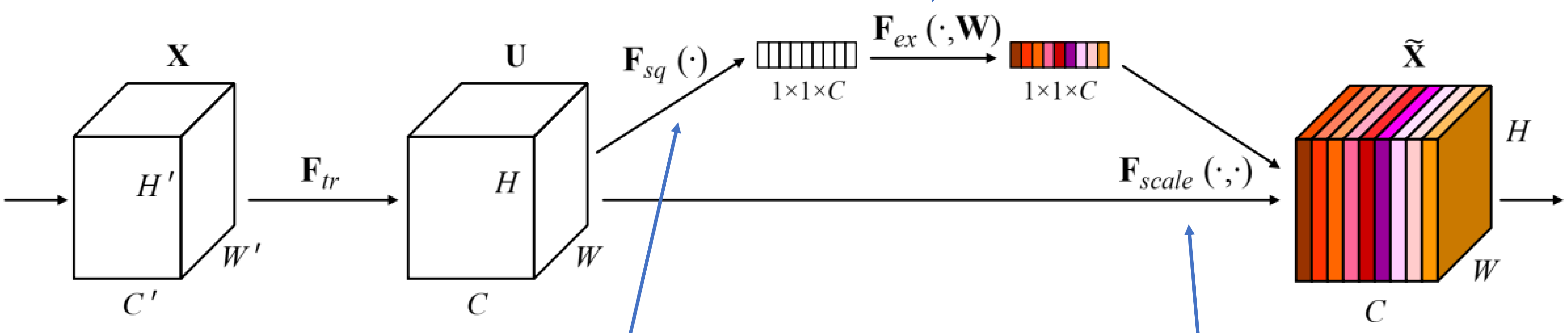


Squeeze-and-excitation networks, Hu et al, CVPR 2018

Squeeze-and-Excitation Networks

- SE Block

$$\mathbf{s} = \mathbf{F}_{ex}(\mathbf{z}, \mathbf{W}) = \sigma(g(\mathbf{z}, \mathbf{W})) = \sigma(\mathbf{W}_2\delta(\mathbf{W}_1\mathbf{z}))$$

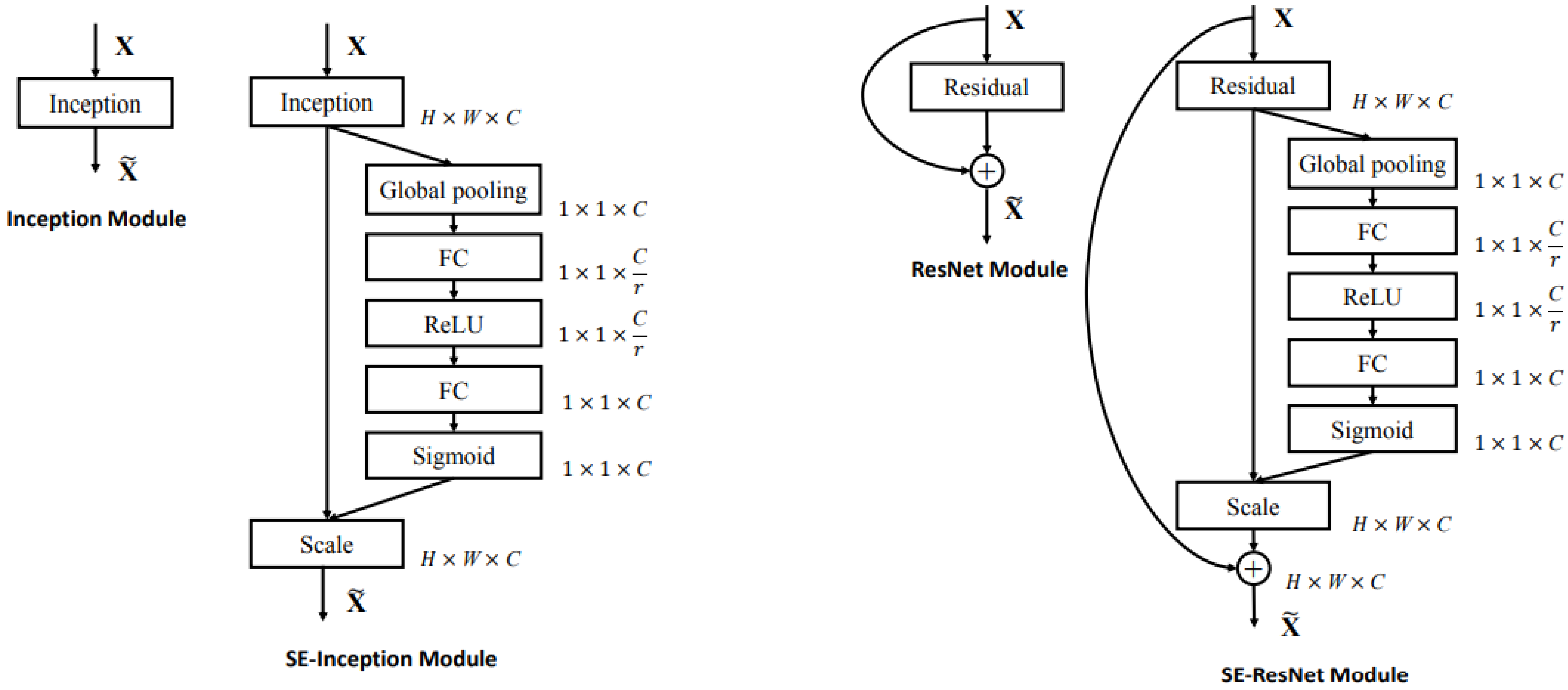


$$z_c = \mathbf{F}_{sq}(\mathbf{u}_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i, j).$$

$$\widetilde{\mathbf{x}}_c = \mathbf{F}_{scale}(\mathbf{u}_c, s_c) = s_c \cdot \mathbf{u}_c$$

Squeeze-and-excitation networks, Hu et al, CVPR 2018

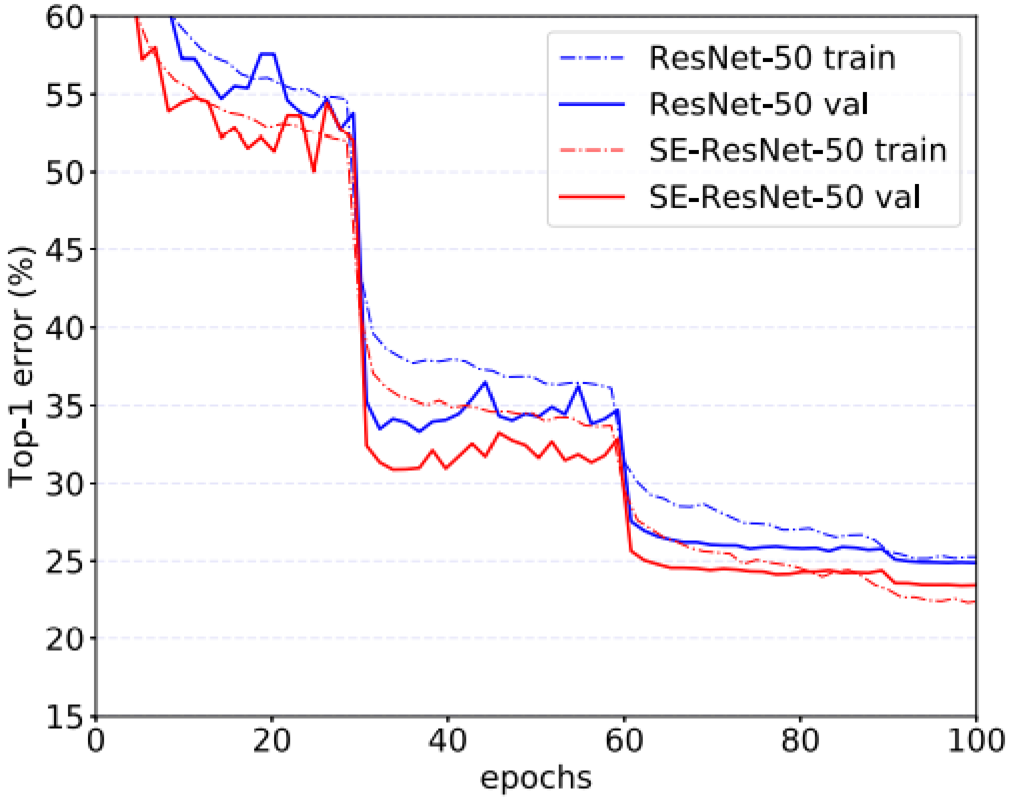
Squeeze-and-Excitation Networks



Squeeze-and-excitation networks, Hu et al, CVPR 2018

Squeeze-and-Excitation Networks

- ILSVRC classification results



	224 × 224		320 × 320 / 299 × 299	
	top-1 err.	top-5 err.	top-1 err.	top-5 err.
ResNet-152 [10]	23.0	6.7	21.3	5.5
ResNet-200 [11]	21.7	5.8	20.1	4.8
Inception-v3 [44]	-	-	21.2	5.6
Inception-v4 [42]	-	-	20.0	5.0
Inception-ResNet-v2 [42]	-	-	19.9	4.9
ResNeXt-101 (64 × 4d) [47]	20.4	5.3	19.1	4.4
DenseNet-264 [14]	22.15	6.12	-	-
Attention-92 [46]	-	-	19.5	4.8
Very Deep PolyNet [51] †	-	-	18.71	4.25
PyramidNet-200 [8]	20.1	5.4	19.2	4.7
DPN-131 [5]	19.93	5.12	18.55	4.16
SENet-154	**18.68**	**4.47**	**17.28**	**3.79**
NASNet-A (6@4032) [55] †	-	-	17.3‡	3.8‡
SENet-154 (post-challenge)	-	-	**16.88**‡	**3.58**‡

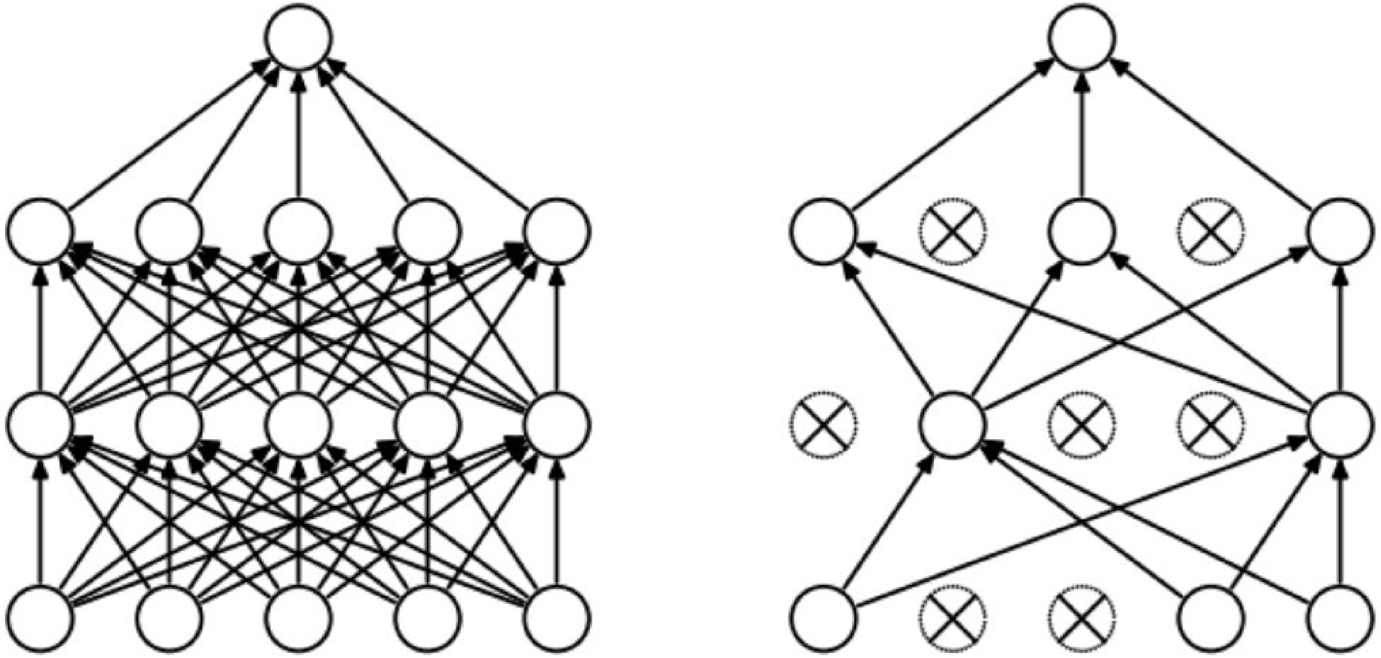
Squeeze-and-excitation networks, Hu et al, CVPR 2018

The Key Ingredients of Training CNNs

Drop-out/Drop-path

Dropout

- Turning off neurons w/ given probability (e.g. 0.5)
- Every iterations, new network architectures emerge

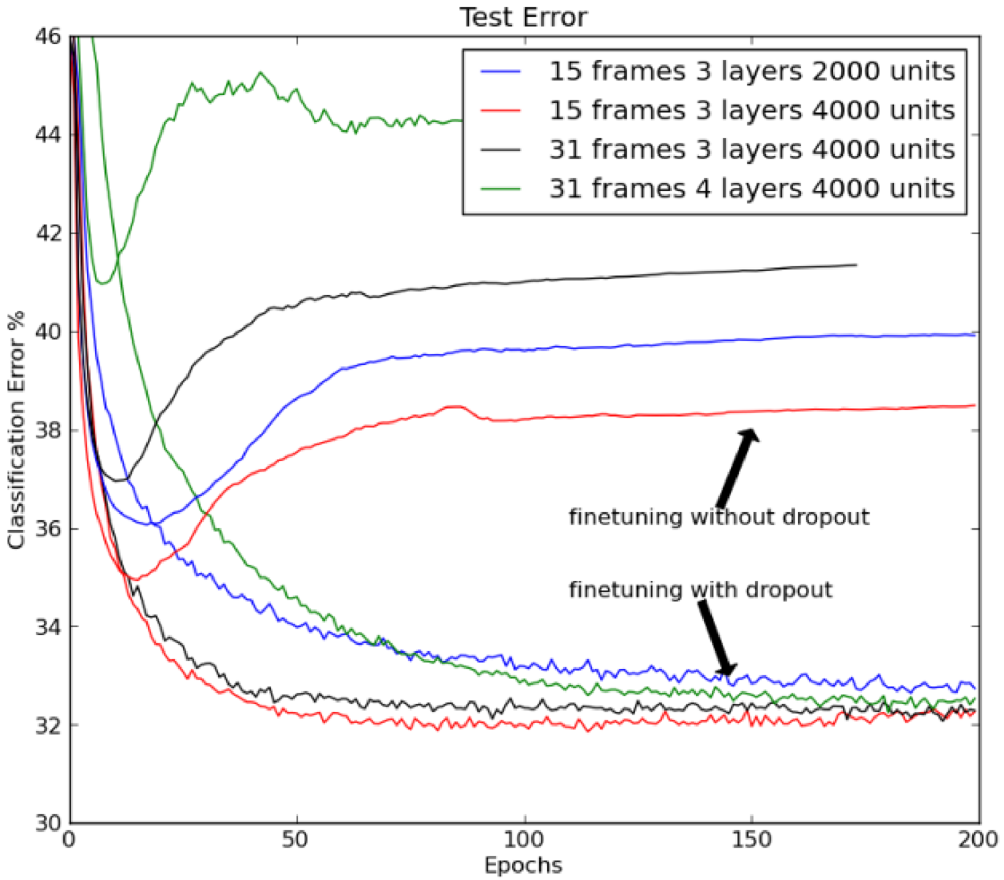


(a) Standard Neural Net (b) After applying dropout.

Improving neural networks by preventing co-adaptation of feature detectors, hinton et al, arXiv 2012

Dropout

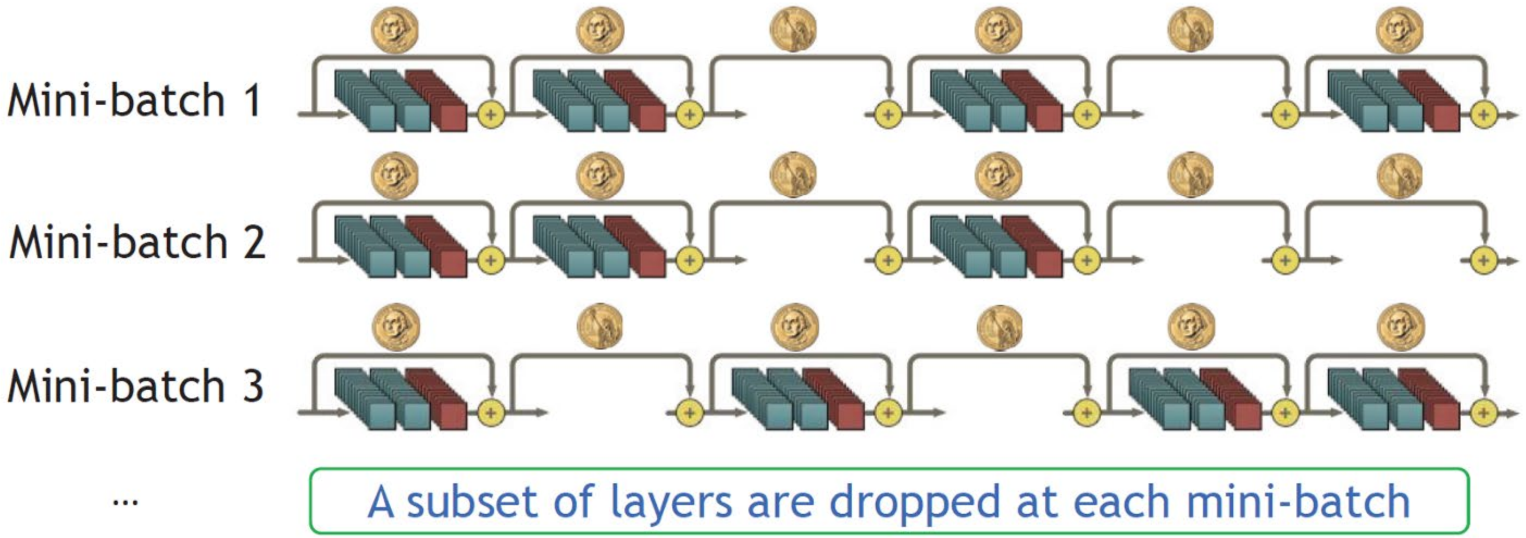
- A simple way to train deep neural networks for improving generalization performance
- Avoiding co-adaptations: a hidden unit cannot rely on other hidden units being present
- Model averaging



Improving neural networks by preventing co-adaptation of feature detectors, hinton et al, arXiv 2012

Stochastic Depth (a.k.a DropPath)

- Training short networks and use deep networks at test time
- During training, randomly drop a subset of layers and bypass them with identity function



$$H_\ell = \mathrm{ReLU}(b_\ell f_\ell(H_{\ell-1}) + \mathrm{id}(H_{\ell-1}))$$

Bernoulli random variable

Deep Networks with Stochastic Depth, Huang et al, ECCV 2016

Stochastic Depth (a.k.a DropPath)

- Linearly decaying ‘drop probability’
  - Later layers will be dropped more frequently

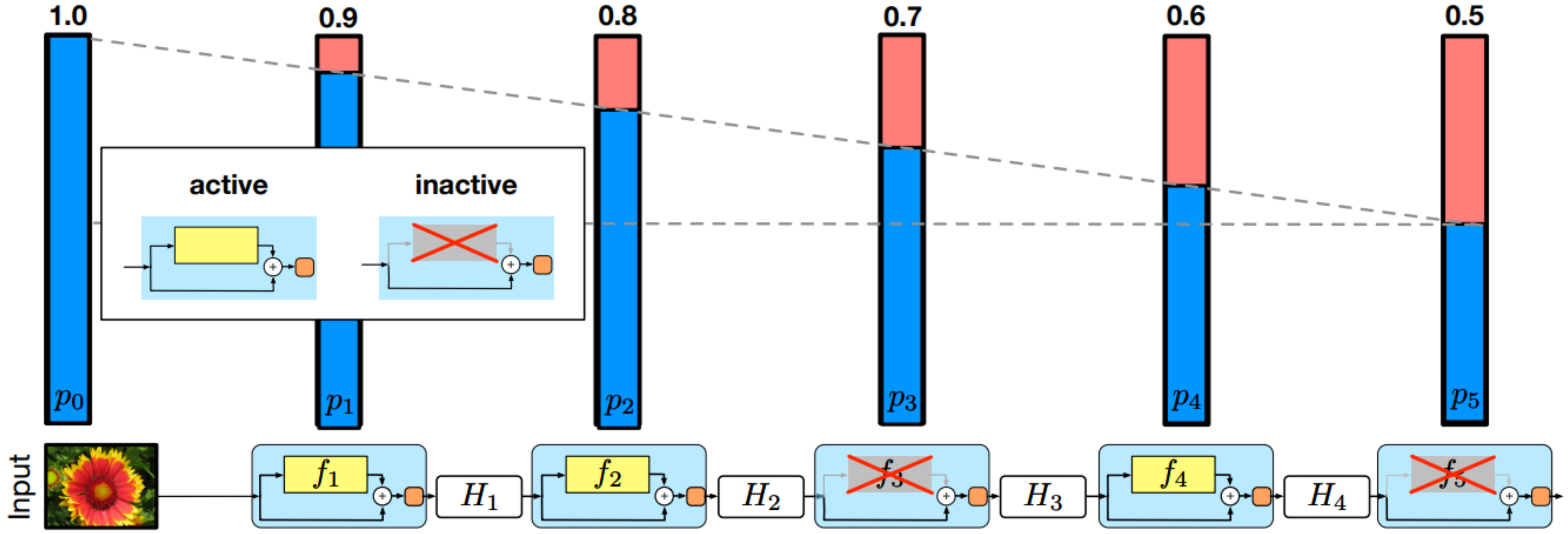
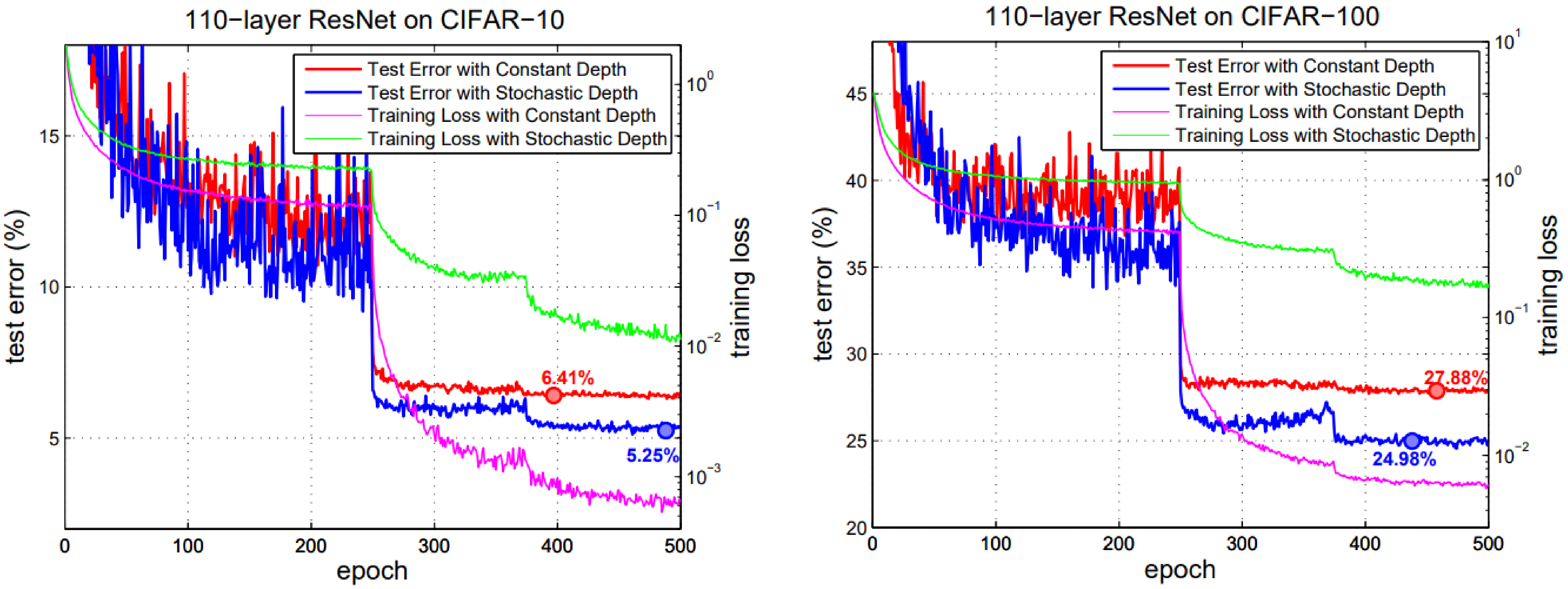


Fig. 2. The linear decay of p_ℓ illustrated on a ResNet with stochastic depth for $p_0 = 1$ and $p_L = 0.5$. Conceptually, we treat the input to the first ResBlock as H_0, which is always active.

Deep Networks with Stochastic Depth, Huang et al, ECCV 2016

Stochastic Depth (a.k.a DropPath)

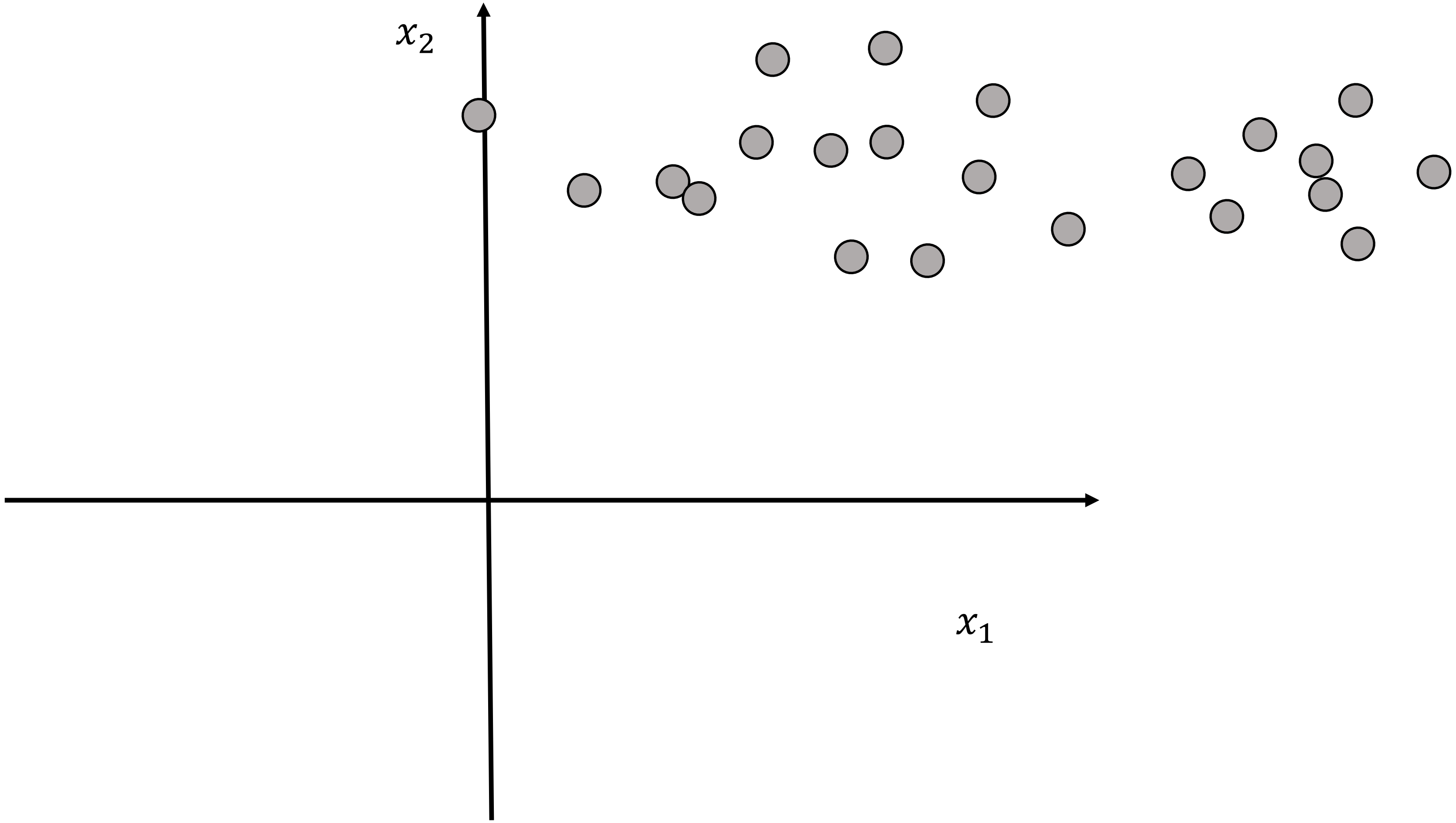


Deep Networks with Stochastic Depth, Huang et al, ECCV 2016

Normalization Methods

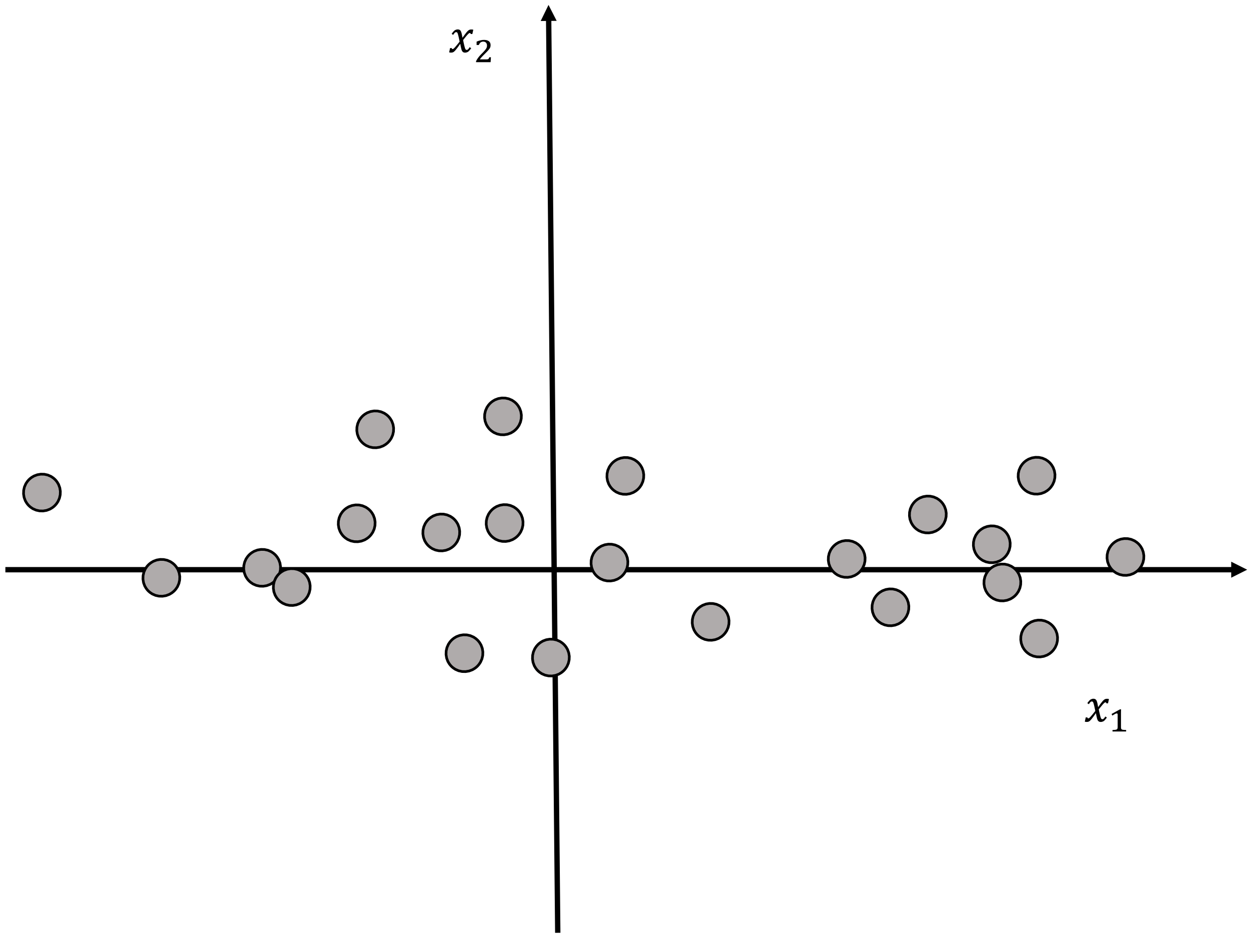
Batch Normalization

- Normalizing training sets



Batch Normalization

- Subtracting the mean

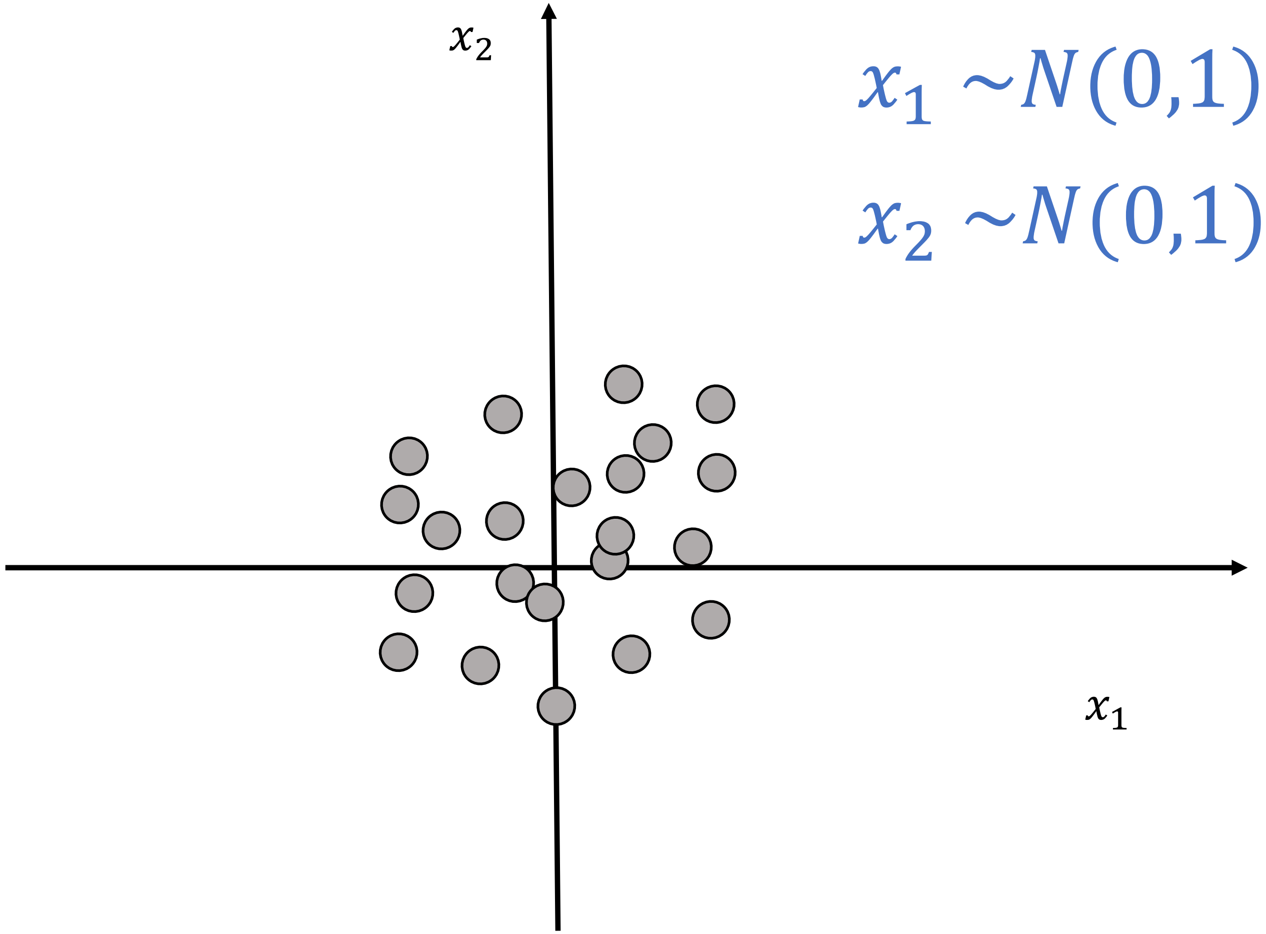


$$\mu_1 = \frac{1}{N}\sum_{i=1}^{N} x_1^{(i)}$$

$$x_1^{(i)} \coloneqq x_1^{(i)} - \mu_1$$

Batch Normalization

- Divide by standard deviation



$x_1 \sim N(0,1)$

$x_2 \sim N(0,1)$

$$\sigma_1^2 = \frac{1}{N}\sum_{i=1}^{N} {x_1^{(i)}}^2$$

$$x_1^{(i)} := x_1^{(i)} / \sigma_1$$

Batch Normalization

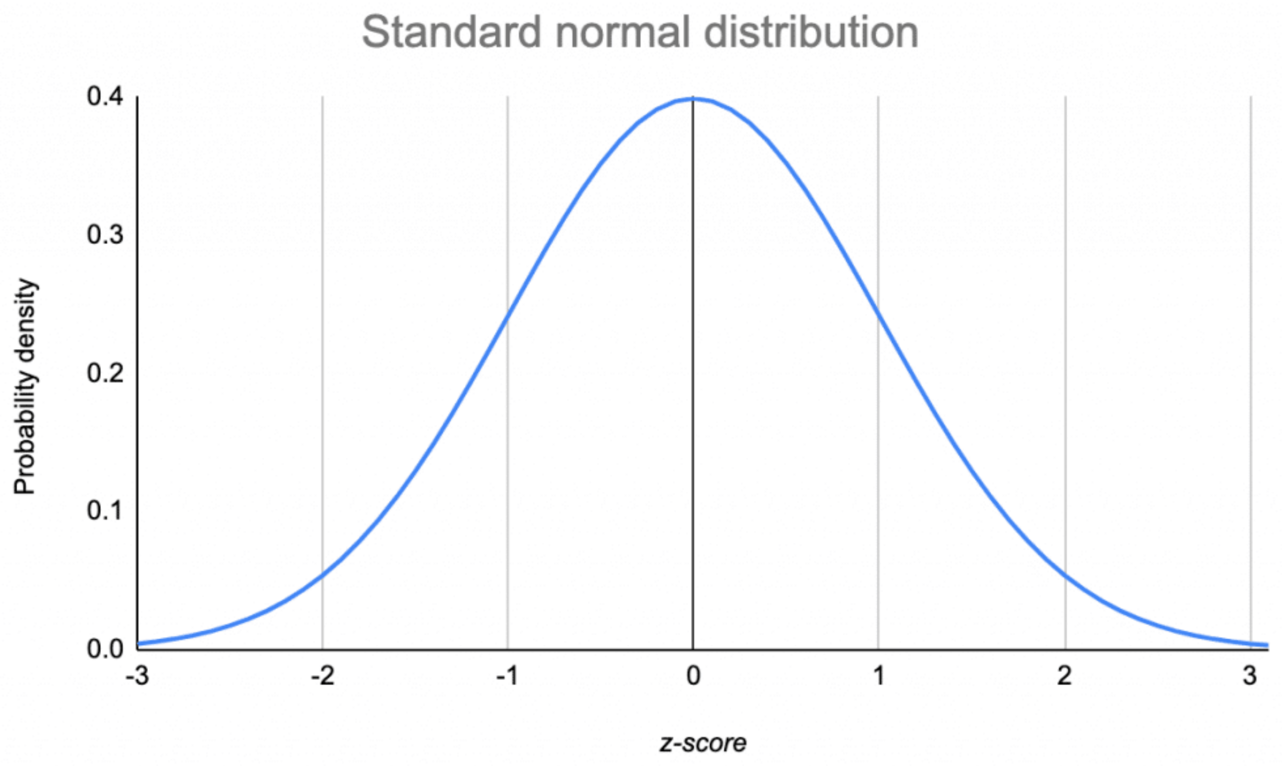
- Standardization

mean

$$z = \frac{x - \mu}{\sigma}$$

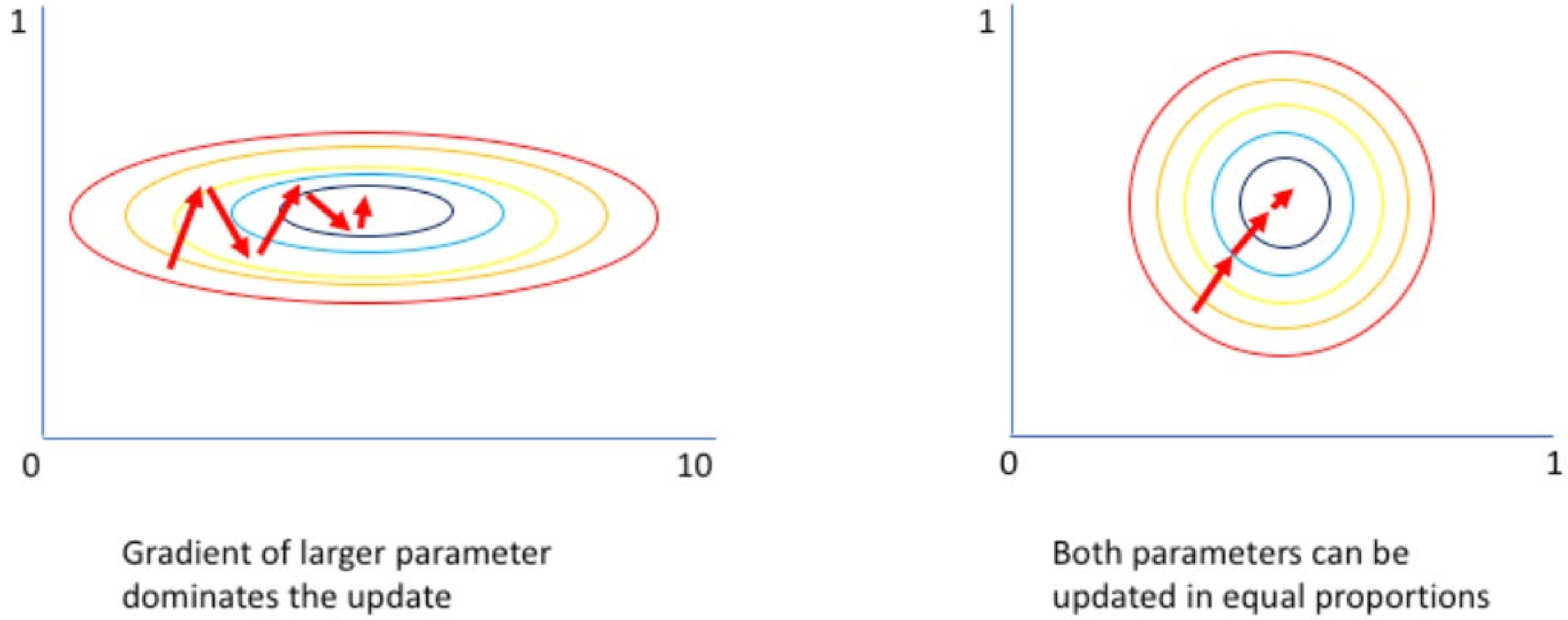
Standard deviation

$$z \sim N(0,1)$$



Batch Normalization

- When un-normalized, the loss surface is more skewed (elongated)
  - Input feature scales are very different each other



Batch Normalization

- Normalizing inputs (also hidden units) based on mini-batch statistics
- Computing mean and variance given the current batch
- During testing, we may not have enough batch size for this (e.g. 1 batch), using mean and variance from the training phase

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

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

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

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

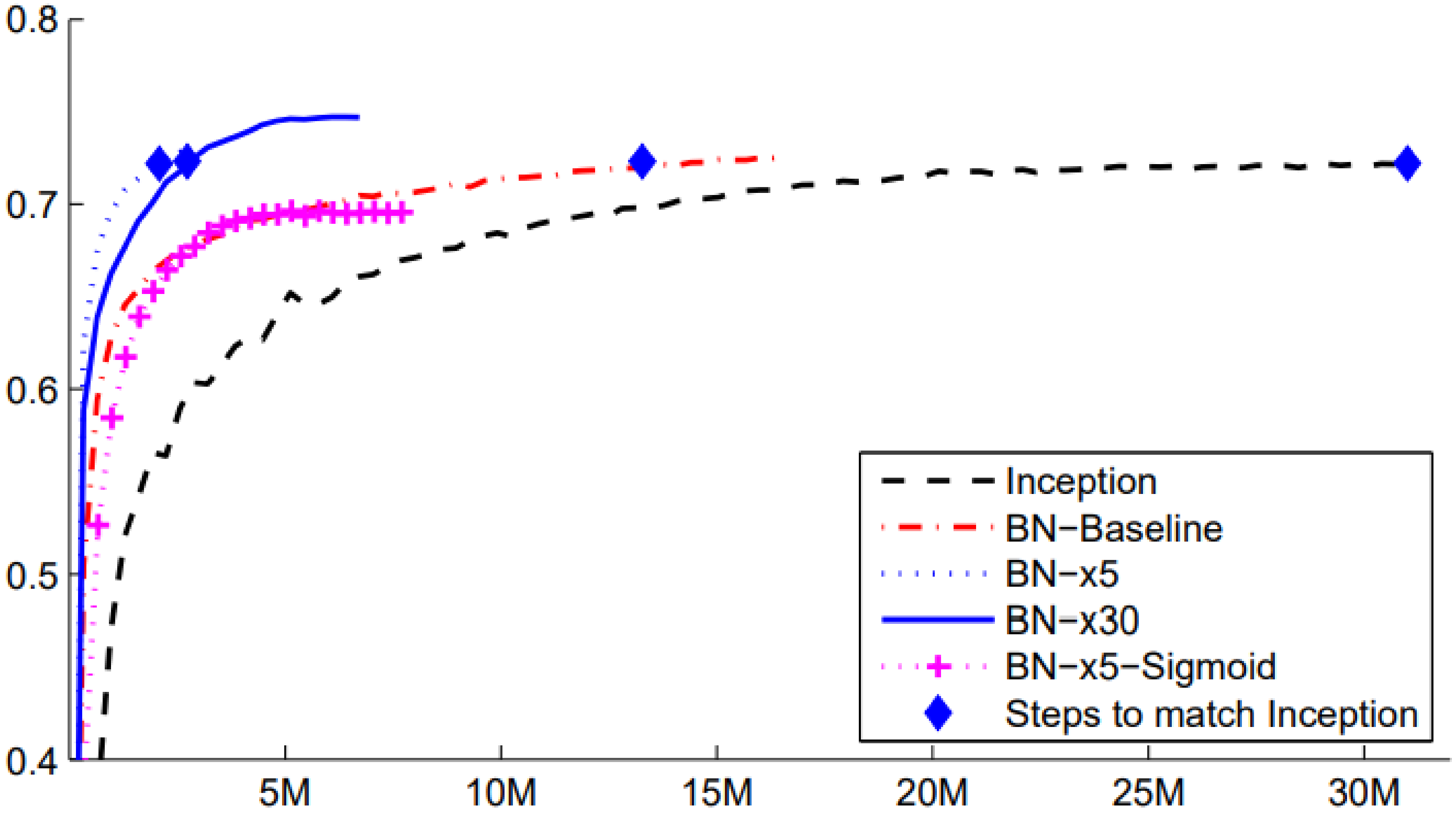
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

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

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

Batch normalization: accelerating deep network training by reducing internal covariate shift, Ioffe et al, ICML 2015

Batch Normalization



Batch normalization: accelerating deep network training by reducing internal covariate shift, Ioffe et al, ICML 2015

Batch Normalization in CNN

M: 128	The number of input channels
D_F: 64	The size of a feature map
$\|B\|$: 32	The mini-batch size

$$\mu \in \mathbb{R}^{?}, \sigma^2 \in \mathbb{R}^{?}$$

Batch Normalization in CNN

M: 128	The number of input channels
D_F: 64	The size of a feature map
$\|B\|$: 32	The mini-batch size

$$\mu \in \mathbb{R}^{128}, \sigma^2 \in \mathbb{R}^{128}$$

Why Batch Normalization Works?

1. Normalization usually makes loss surface less ‘skewed’
2. BN may reduce the internal covariance shift
   - [1502.03167] Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift (arxiv.org)
3. BN makes loss surface smoother
   - [1805.11604] How Does Batch Normalization Help Optimization? (arxiv.org)

Layer Normalization

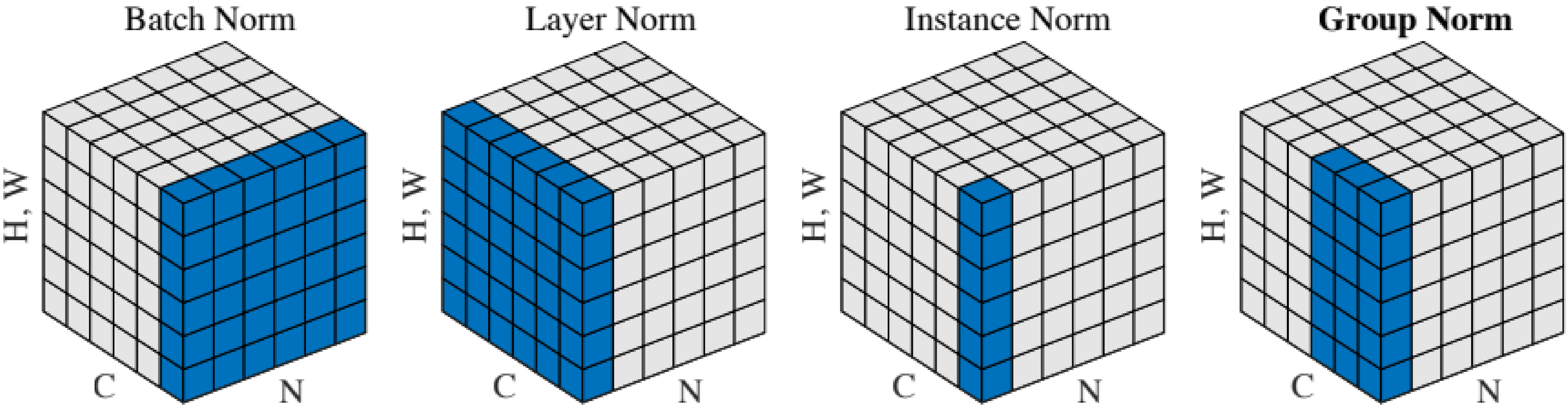
- Batch normalization is dependent on the mini-batch size
  - What about the network size is too big, so only few mini-batch sizes are allowed?
- It is not obvious how to apply batch normalization to RNNs

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \qquad \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^l - \mu^l\right)^2}$$

H: the number of hidden units in a layer

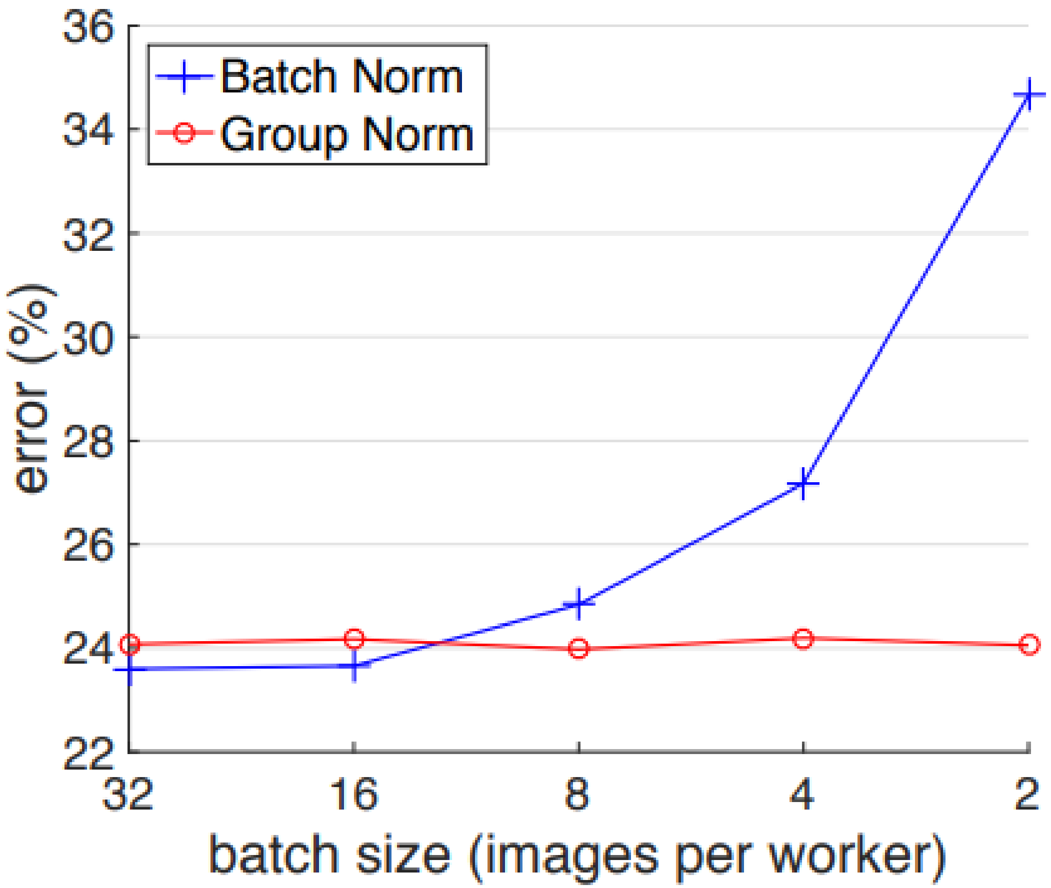
Layer normalization, Ba et al, arXiv 2016

Other Normalizaion Methods



Group normalization, Wu et al, ECCV 2018

Other Normalizaion Methods



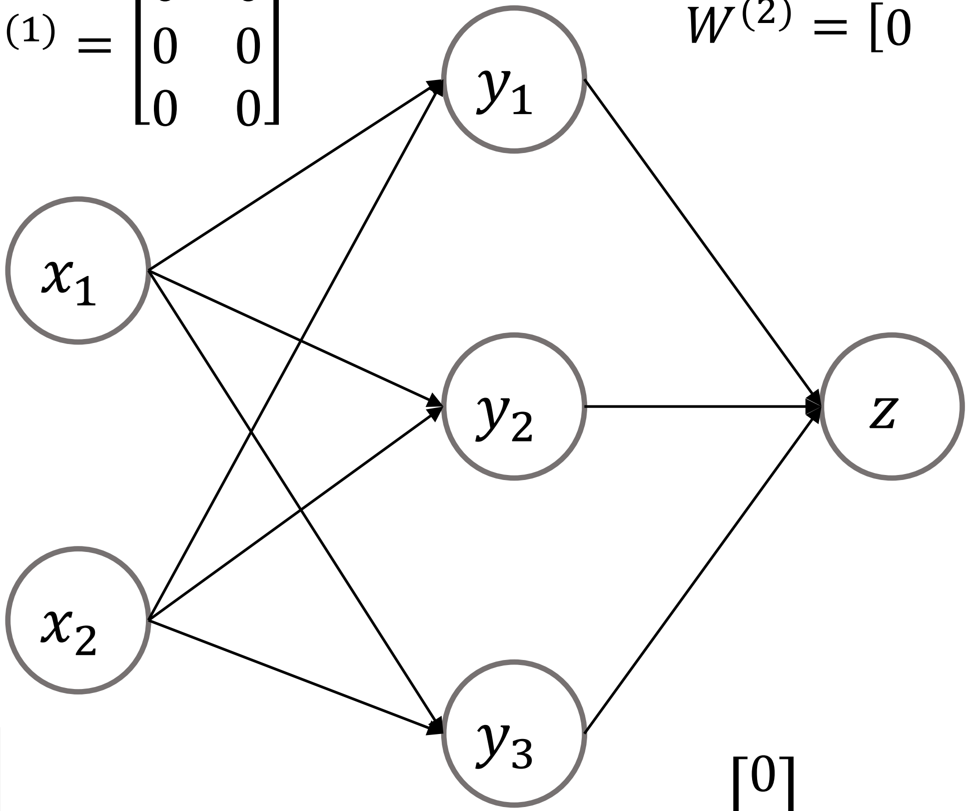
Group normalization, Wu et al, ECCV 2018

Initialization

Weight Initialization

- Zero initialization

$$W^{(1)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad W^{(2)} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$



$$y = W^{(1)}x$$

$$z = W^{(2)}y$$

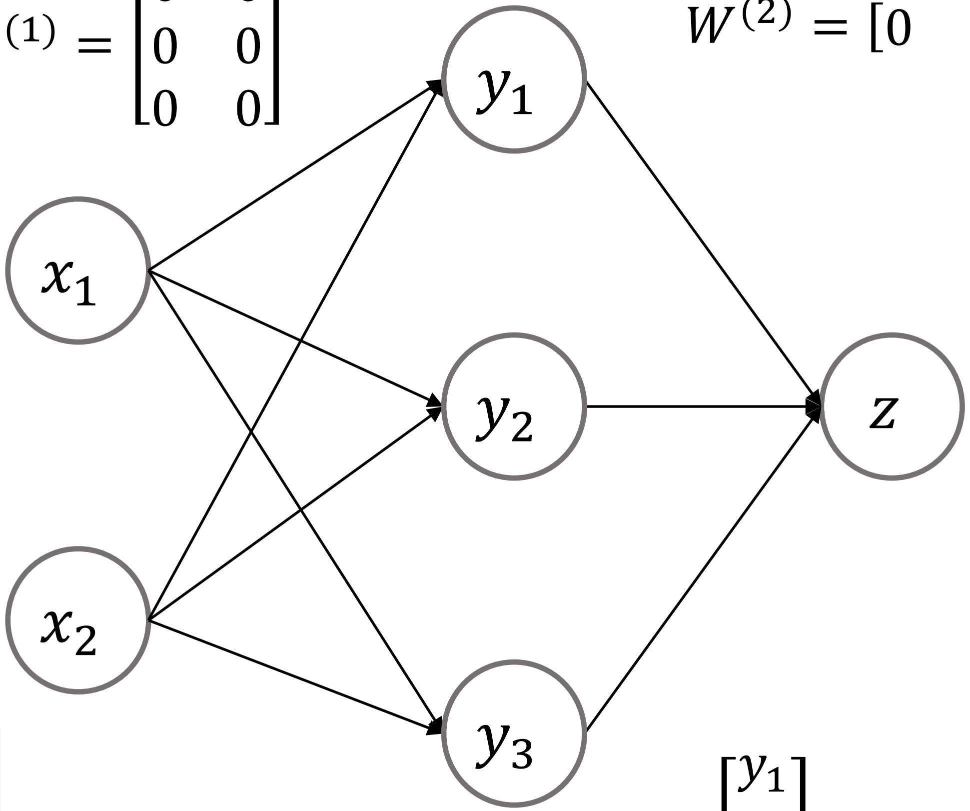
$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad y = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Weight Initialization

- Zero initialization

$$W^{(1)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$



$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$\frac{\partial L}{\partial z} = k$$

$$\frac{\partial L}{\partial W^{(2)}} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial W^{(2)}} = k y^{\top} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

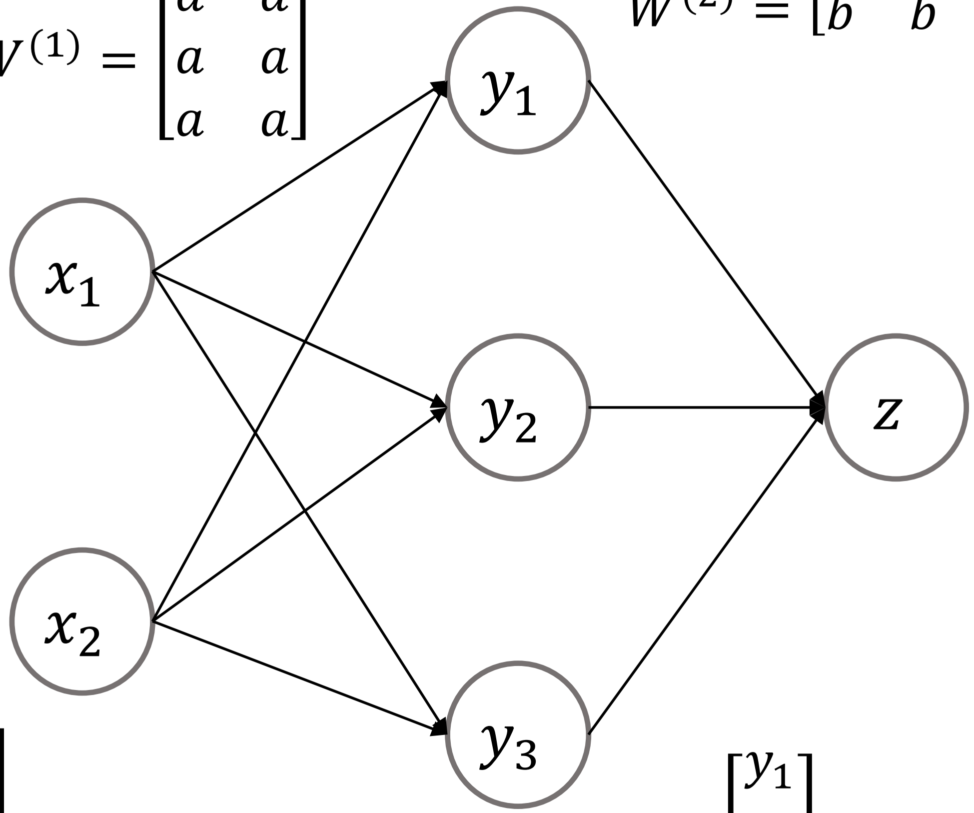
$$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y} = k {W^{(2)}}^{\top} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\frac{\partial L}{\partial W^{(1)}} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial W^{(1)}} = \frac{\partial L}{\partial y} x^{\top} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Weight Initialization

- Same value initialization

$$W^{(1)} = \begin{bmatrix} a & a \\ a & a \\ a & a \end{bmatrix} \qquad W^{(2)} = \begin{bmatrix} b & b & b \end{bmatrix}$$



$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$\frac{\partial L}{\partial z} = k$$

$$\frac{\partial L}{\partial W^{(2)}} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial W^{(2)}} = k y^{\top} = \begin{bmatrix} k y_1 & k y_2 & k y_3 \end{bmatrix}$$

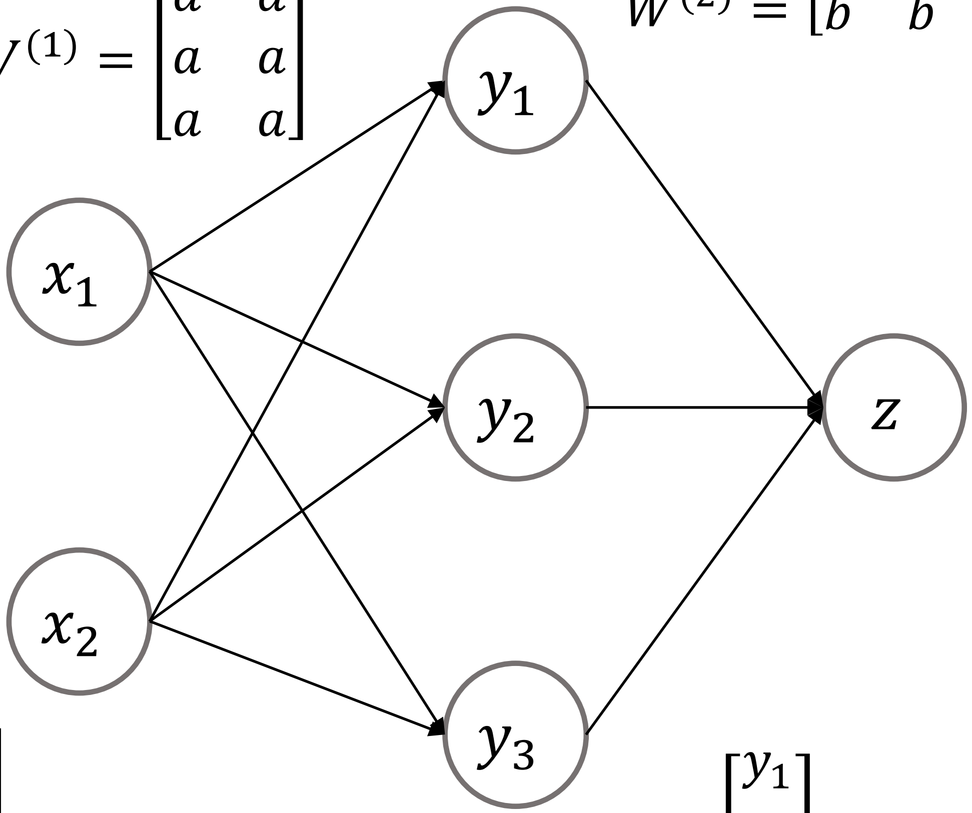
$$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y} = k W^{(2)^{\top}} = \begin{bmatrix} kb \\ kb \\ kb \end{bmatrix}$$

$$\frac{\partial L}{\partial W^{(1)}} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial W^{(1)}} = \frac{\partial L}{\partial y} x^{\top} = \begin{bmatrix} kbx_1 & kbx_2 \\ kbx_1 & kbx_2 \\ kbx_1 & kbx_2 \end{bmatrix}$$

Weight Initialization

- Same value initialization

$$W^{(1)} = \begin{bmatrix} a & a \\ a & a \\ a & a \end{bmatrix} \qquad W^{(2)} = \begin{bmatrix} b & b & b \end{bmatrix}$$



$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$\frac{\partial L}{\partial W^{(1)}} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial W^{(1)}} = \frac{\partial L}{\partial y} x^{\top} = \begin{bmatrix} kbx_1 & kbx_2 \\ kbx_1 & kbx_2 \\ kbx_1 & kbx_2 \end{bmatrix}$$

$$y_1 = y_2 = y_3$$

Need to break ‘symmetry’

Weight Initialization

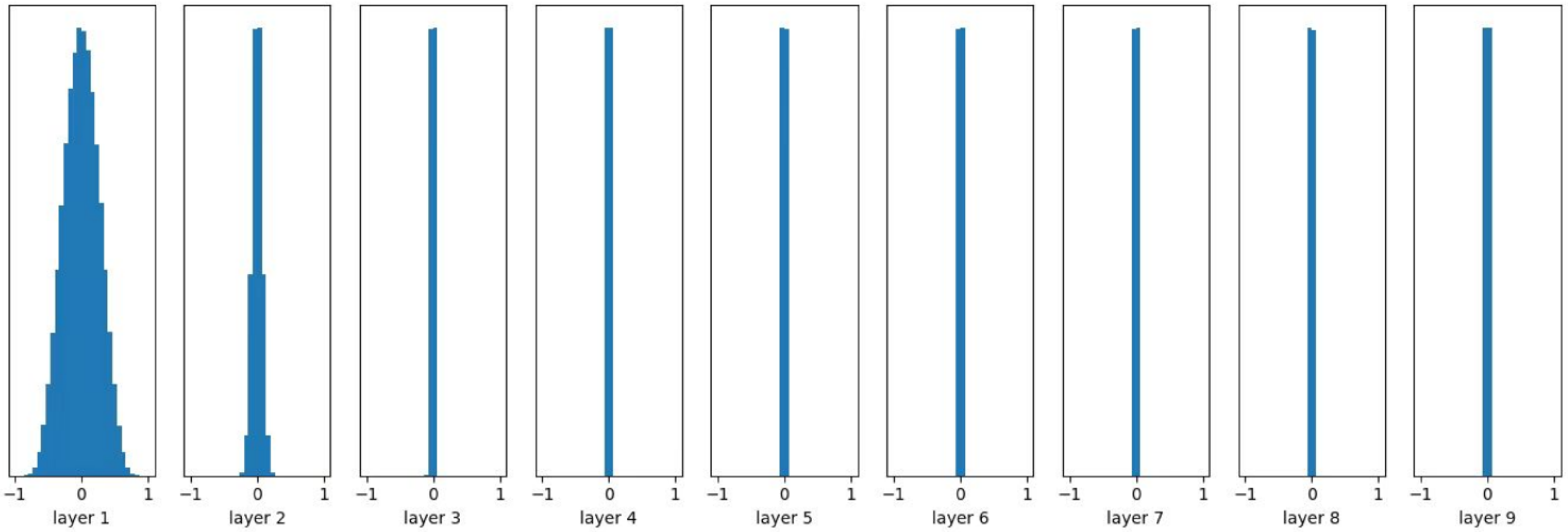
- Random initialization

$$W \sim N(0, \sigma^2)$$

Weight Initialization

- Random initialization
  - Gaussian with zero mean and 'small standard deviation'
  - Gaussian distributed input data

$$W \sim N(0, 0.01)$$ Tanh



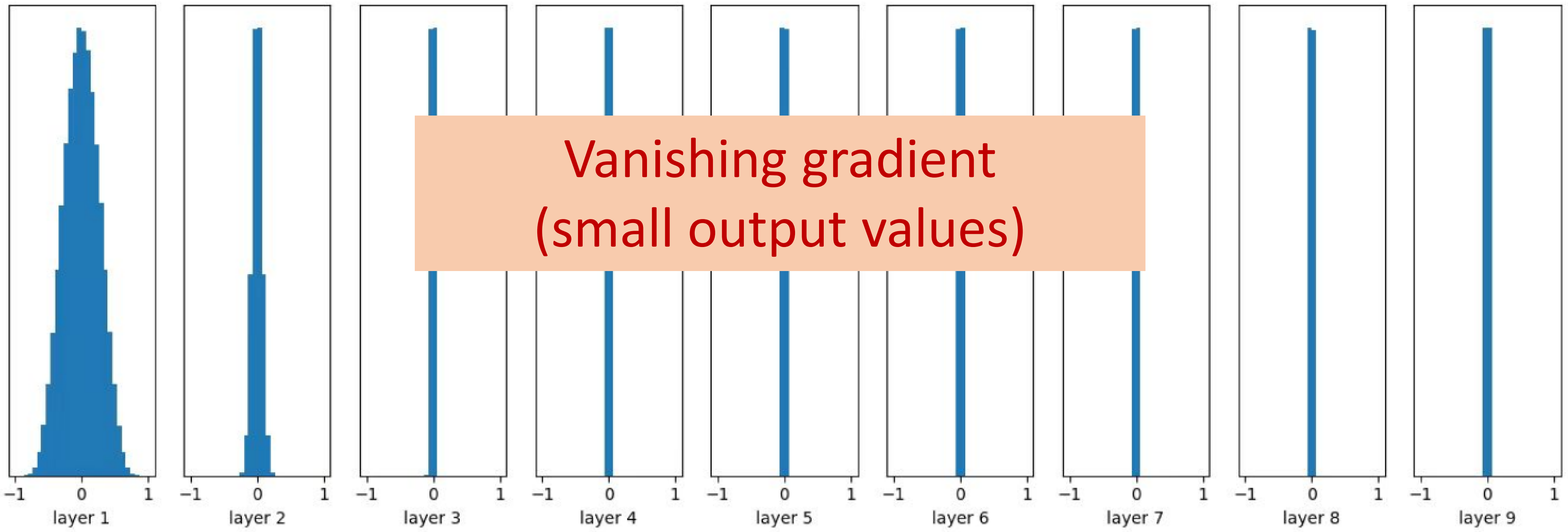
[I2DL (niessner.github.io)](niessner.github.io)

Weight Initialization

- Random initialization
  - Gaussian with zero mean and 'small standard deviation'
  - Gaussian distributed input data

$$W \sim N(0, 0.01)$$

Tanh

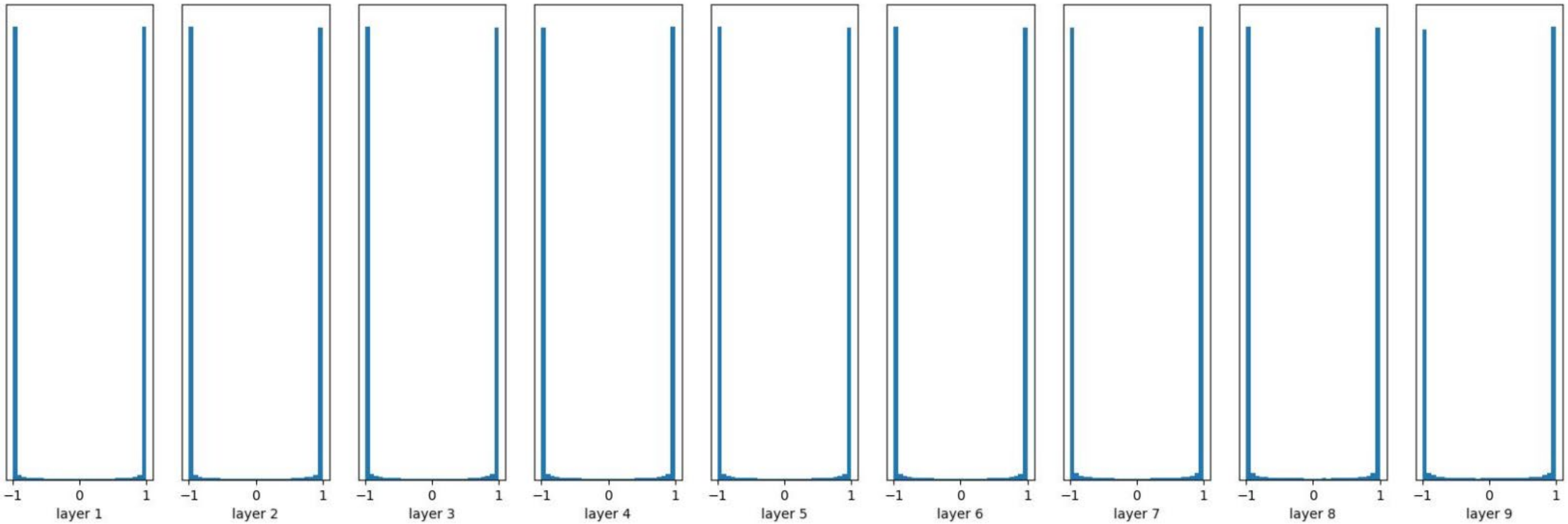


Weight Initialization

- Random initialization
  - Gaussian with zero mean and 'large standard deviation'
  - Gaussian distributed input data

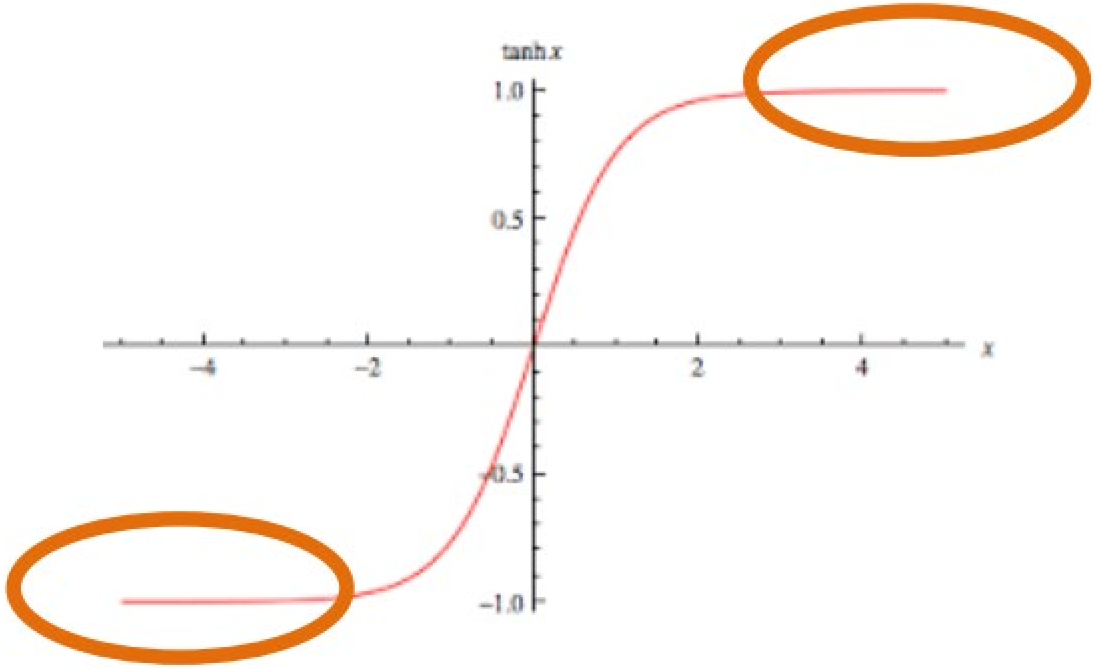
$$W \sim N(0, 1)$$

Tanh



Weight Initialization

$$y_1 = w_1 x_1 + w_2 x_2 + w_3 x_3 \cdots w_n x_n$$



Xavier Initialization

- The more hidden units, less weight initial values
- Trying to 'match' the variance of each layers

$$\mathrm{Var}\left[\sum_{i=1}^{n} w_i x_i\right] = \sum_{i=1}^{n} \mathrm{Var}[w_i x_i]$$

$$= \sum_{i=1}^{n} \mathbb{E}[w_i^2 x_i^2] - \mathbb{E}[w_i x_i]^2$$

(Independent)

$$= \sum_{i=1}^{n} \mathbb{E}[w_i]^2 \mathrm{Var}[x_i] + \mathbb{E}[x_i]^2 \mathrm{Var}[w_i] + \mathrm{Var}[x_i]\mathrm{Var}[w_i]$$

(Independent)

$$= \sum_{i=1}^{n} \mathrm{Var}[x_i]\mathrm{Var}[w_i] = n\mathrm{Var}[x_i]\mathrm{Var}[w_i]$$

(i.i.d)

Xavier Initialization

- The more hidden units, less weight initial values
- Trying to ‘match’ the variance of each layers

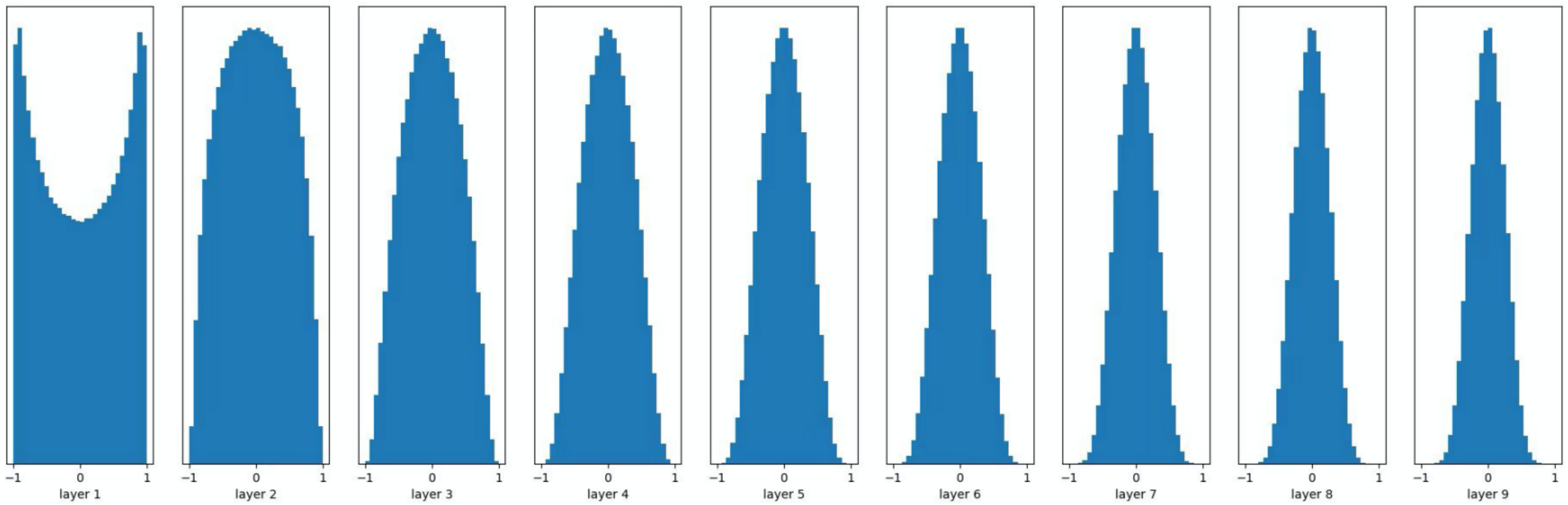
$$\mathrm{Var}\left[\sum_{i=1}^{n} w_i x_i\right] = n\mathrm{Var}[x_i]\mathrm{Var}[w_i] = \mathrm{Var}[x_i] \qquad (\mathrm{Var}[w_i] = \frac{1}{n})$$

Xavier Initialization

- The more hidden units, less weight initial values
- Trying to 'match' the variance of each layers

$$W \sim N(0, \frac{1}{n})$$

Tanh

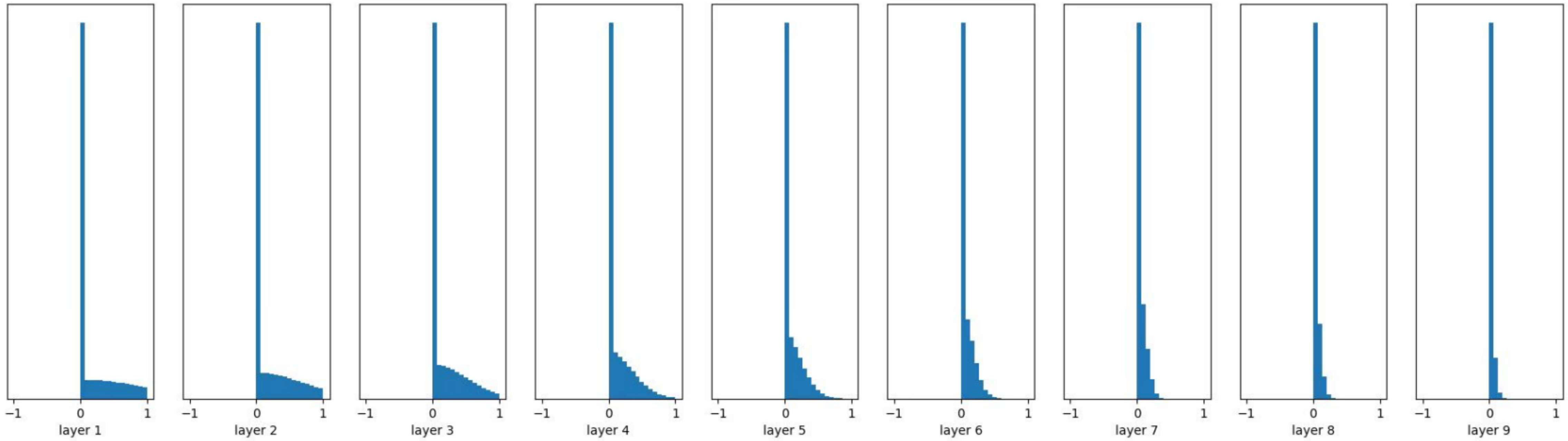


Xavier Initialization

- The more hidden units, less weight initial values
- Trying to 'match' the variance of each layers

$$W \sim N(0, \frac{1}{n})$$

ReLU

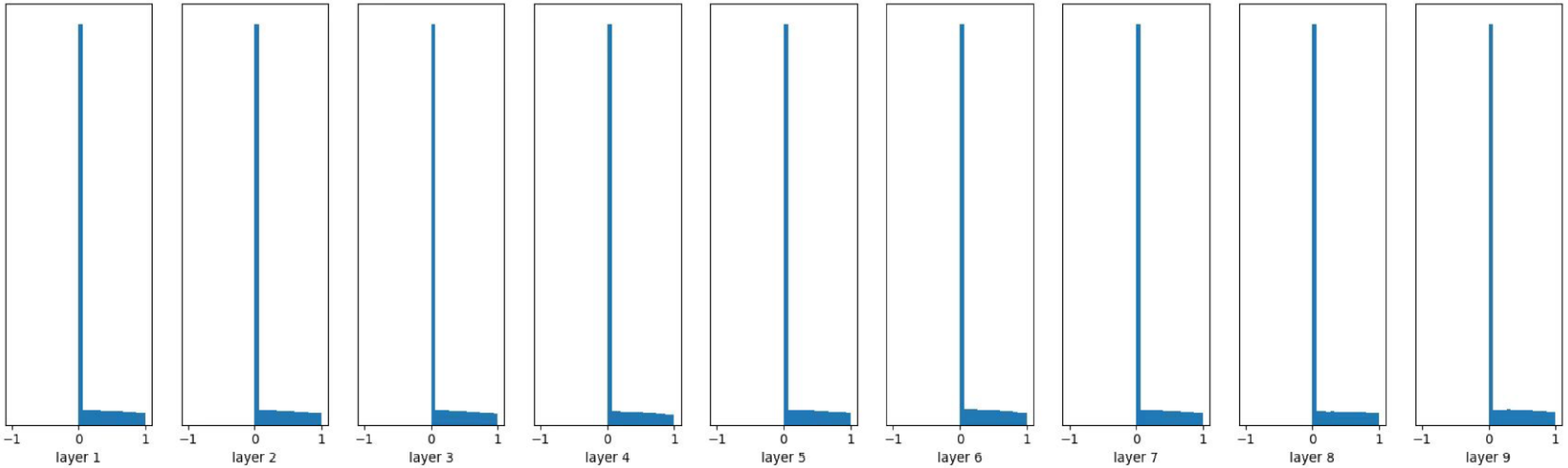


Xavier Initialization

- ReLU zeros out the half of activations

$$W \sim N(0, \frac{2}{n})$$

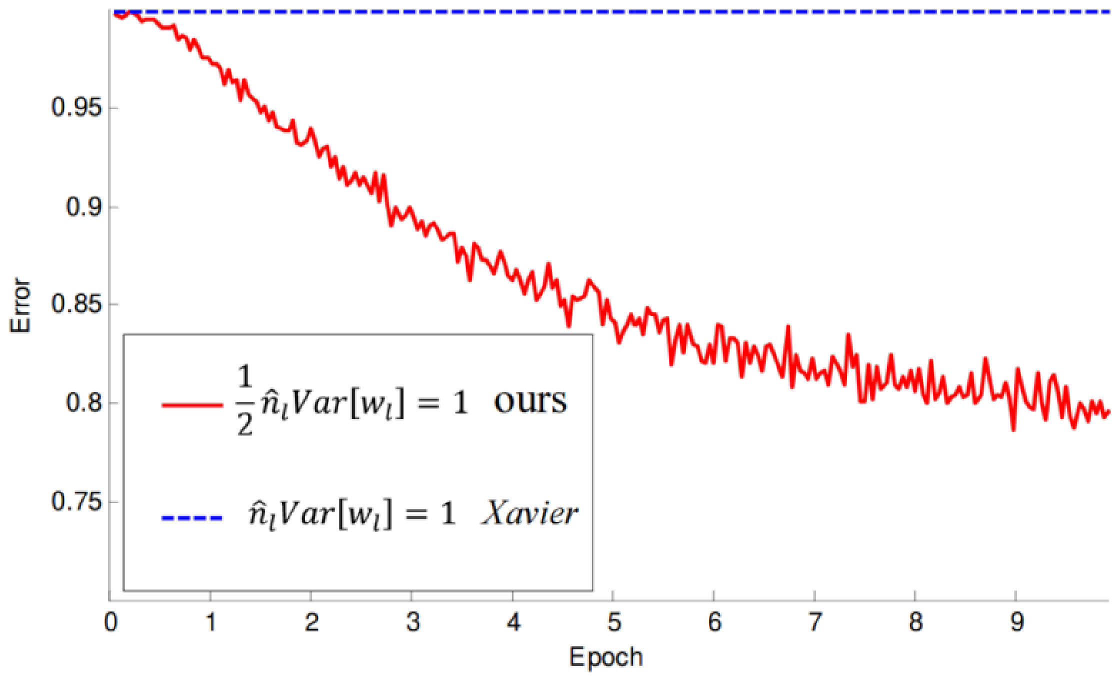
ReLU



Xavier Initialization

- ReLU zeros out the half of activations

$$W \sim N(0, \frac{2}{n})$$ ReLU



Xavier Initialization

```
[docs]def xavier_normal_(tensor: Tensor, gain: float = 1.) -> Tensor:
    r"""Fills the input `Tensor` with values according to the method
    described in `Understanding the difficulty of training deep feedforward
    neural networks` - Glorot, X. & Bengio, Y. (2010), using a normal
    distribution. The resulting tensor will have values sampled from
    :math:`\mathcal{N}(0, \text{std}^2)` where

    .. math::
        \text{std} = \text{gain} \times \sqrt{\frac{2}{\text{fan\_in} +
\text{fan\_out}}}

    Also known as Glorot initialization.

    Args:
        tensor: an n-dimensional `torch.Tensor`
        gain: an optional scaling factor

    Examples:
        >>> w = torch.empty(3, 5)
        >>> nn.init.xavier_normal_(w)
    """
    fan_in, fan_out = _calculate_fan_in_and_fan_out(tensor)
    std = gain * math.sqrt(2.0 / float(fan_in + fan_out))

    return _no_grad_normal_(tensor, 0., std)
```

$$W \sim N(0, \frac{2}{n})$$

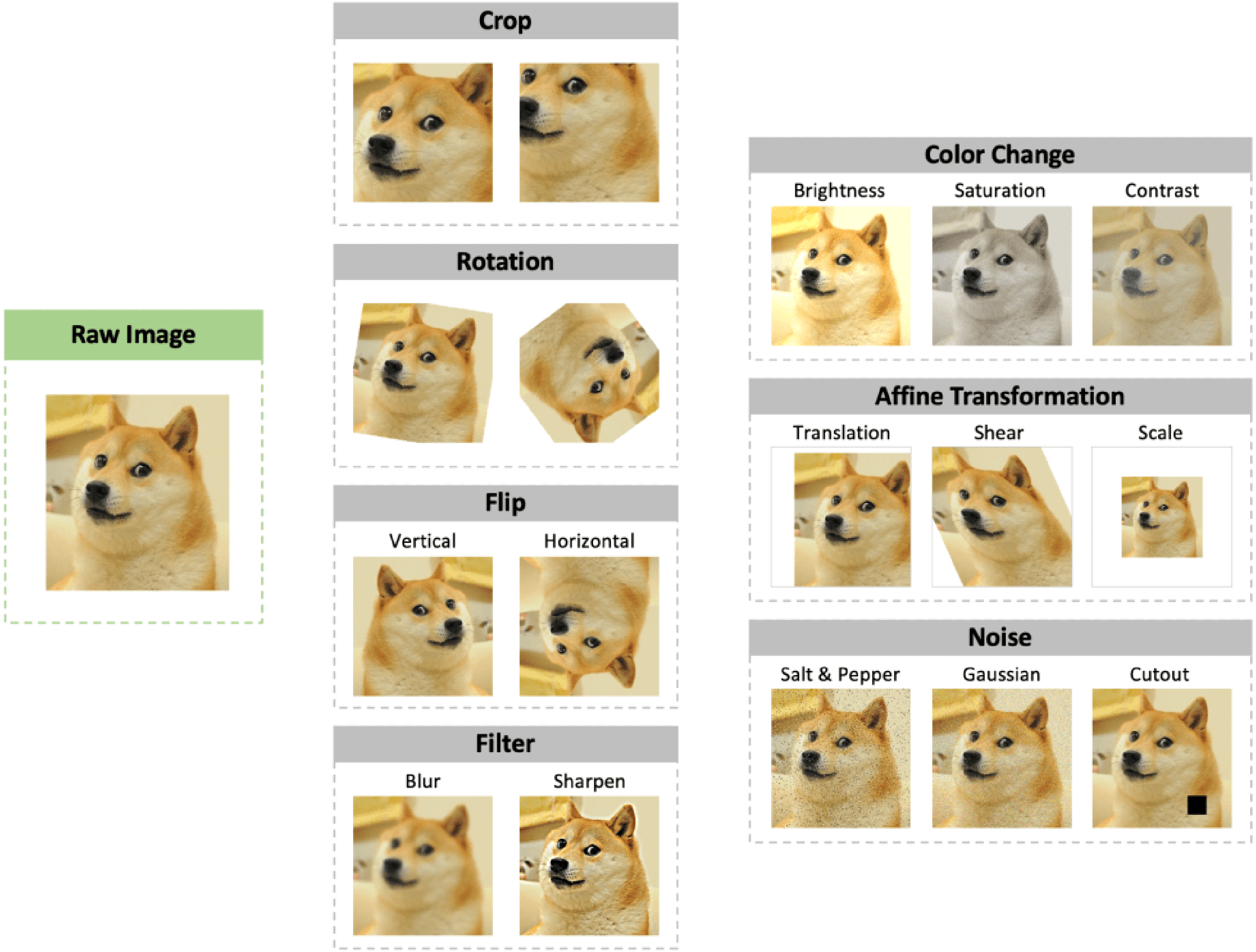
Data Augmentation

Data Augmentation

- Deep learning is data hungry and can be easily overfitted
- Let’s augment our datasets!

[Data Augmentation by fastai v1. This article presents the techniques of… | by Pierre Guillou | Medium]

Data Augmentation



[Complete Guide to Data Augmentation for Computer Vision (notrocketscience.blog)]

Data Augmentation

- Cutout

Improved Regularization of Convolutional Neural Networks with Cutout, DeVries et al, arXiv 2017

Data Augmentation

- Mixup

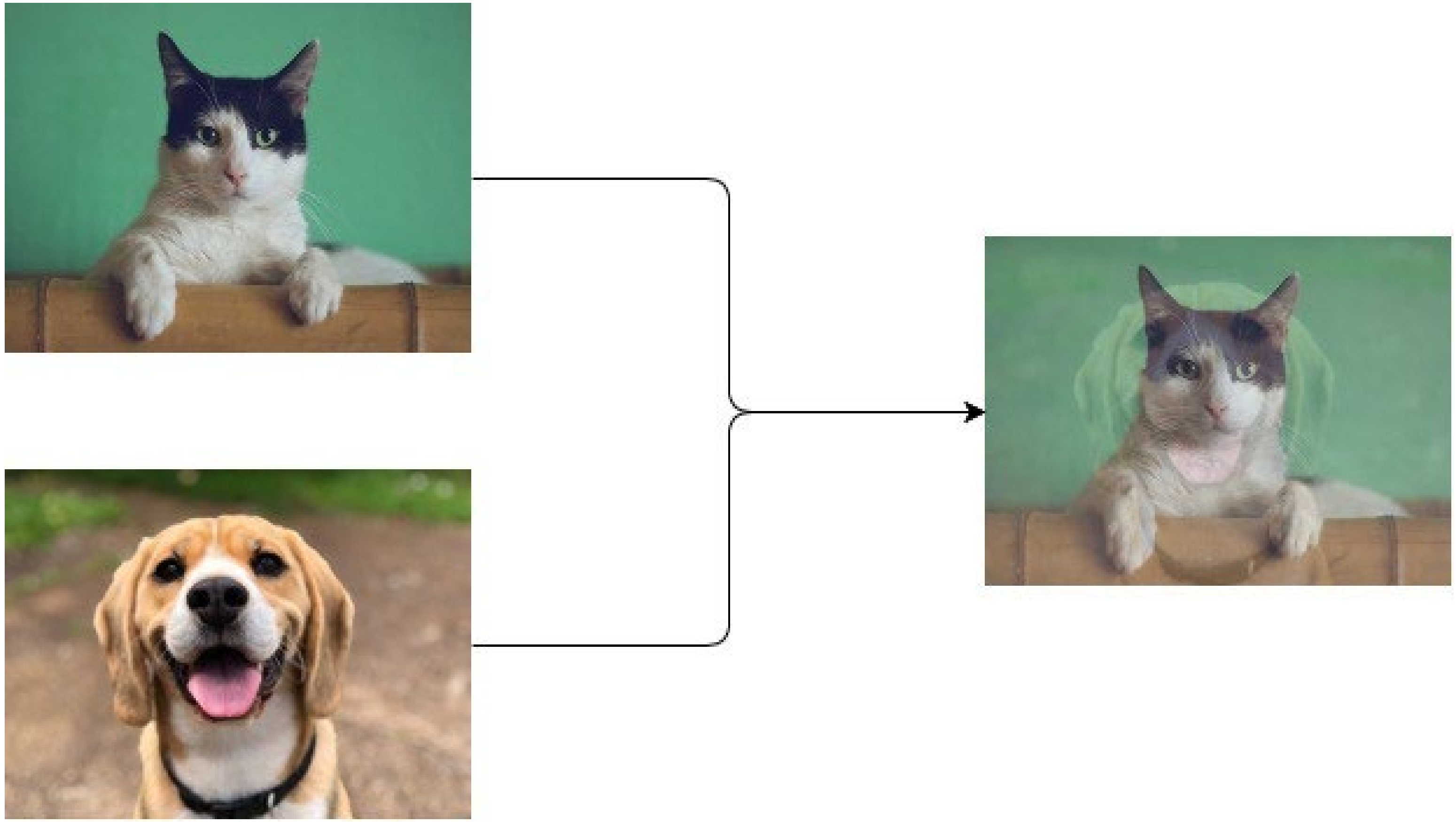
Data (image) $$\hat{x} = \lambda x_i + (1-\lambda) x_j$$

Label (one-hot) $$\hat{y} = \lambda y_i + (1-\lambda) y_j$$

mixup: Beyond Empirical Risk Minimization, Zhang et al, ICLR 2018

Data Augmentation

- Mixup



mixup: Beyond Empirical Risk Minimization, Zhang et al, ICLR 2018

Learning Rates

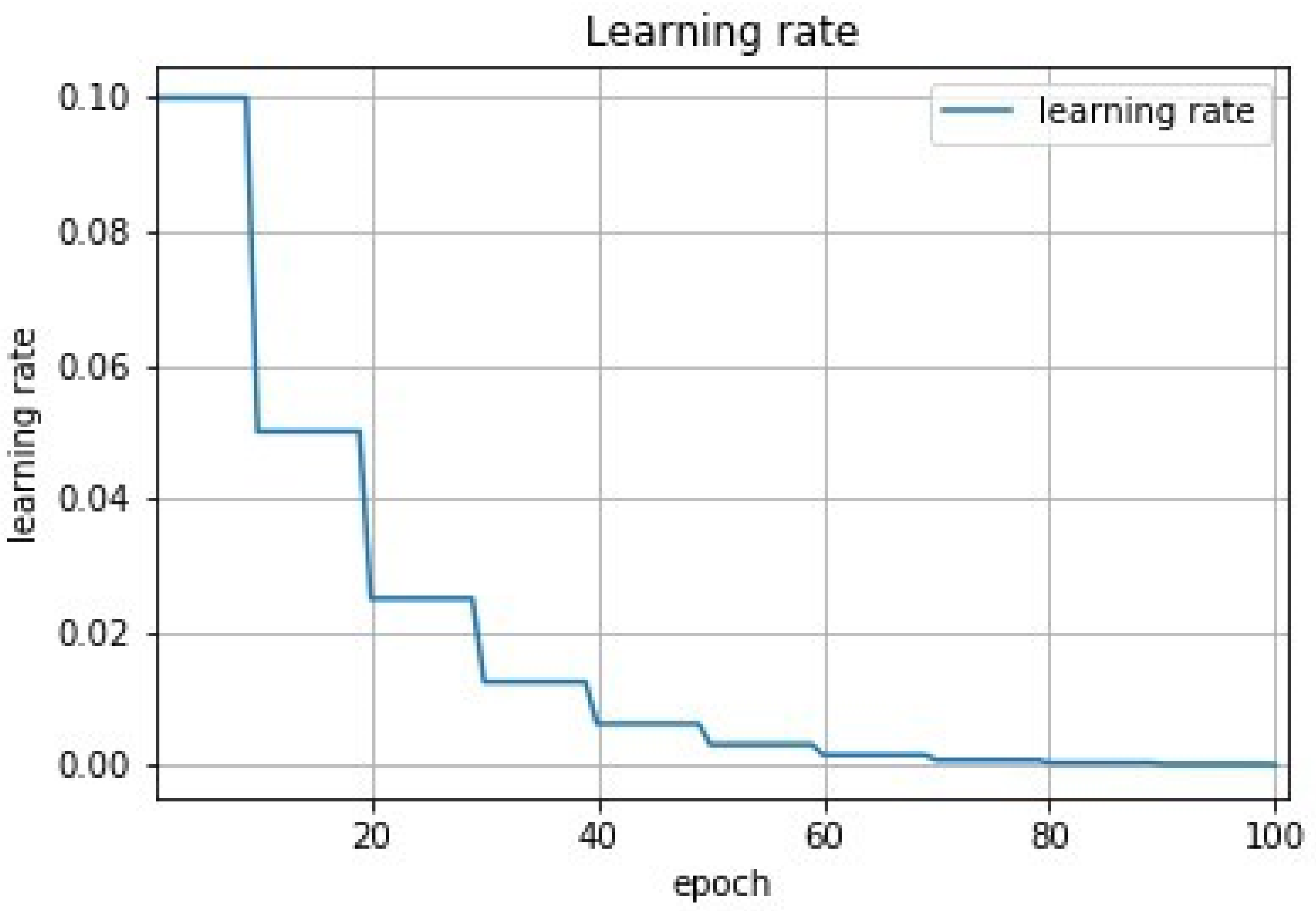
Learning Rates

- The most important hyperparameter in training deep neural networks
- Even with the adaptive optimizers, e.g. ADAM, learning rate schedule is very important

1. The magnitude of the learning rate
   - 0.1, 0.01, 0.001, 0.0001 would be a good candidate;;
2. The rate of decay
3. Initialization

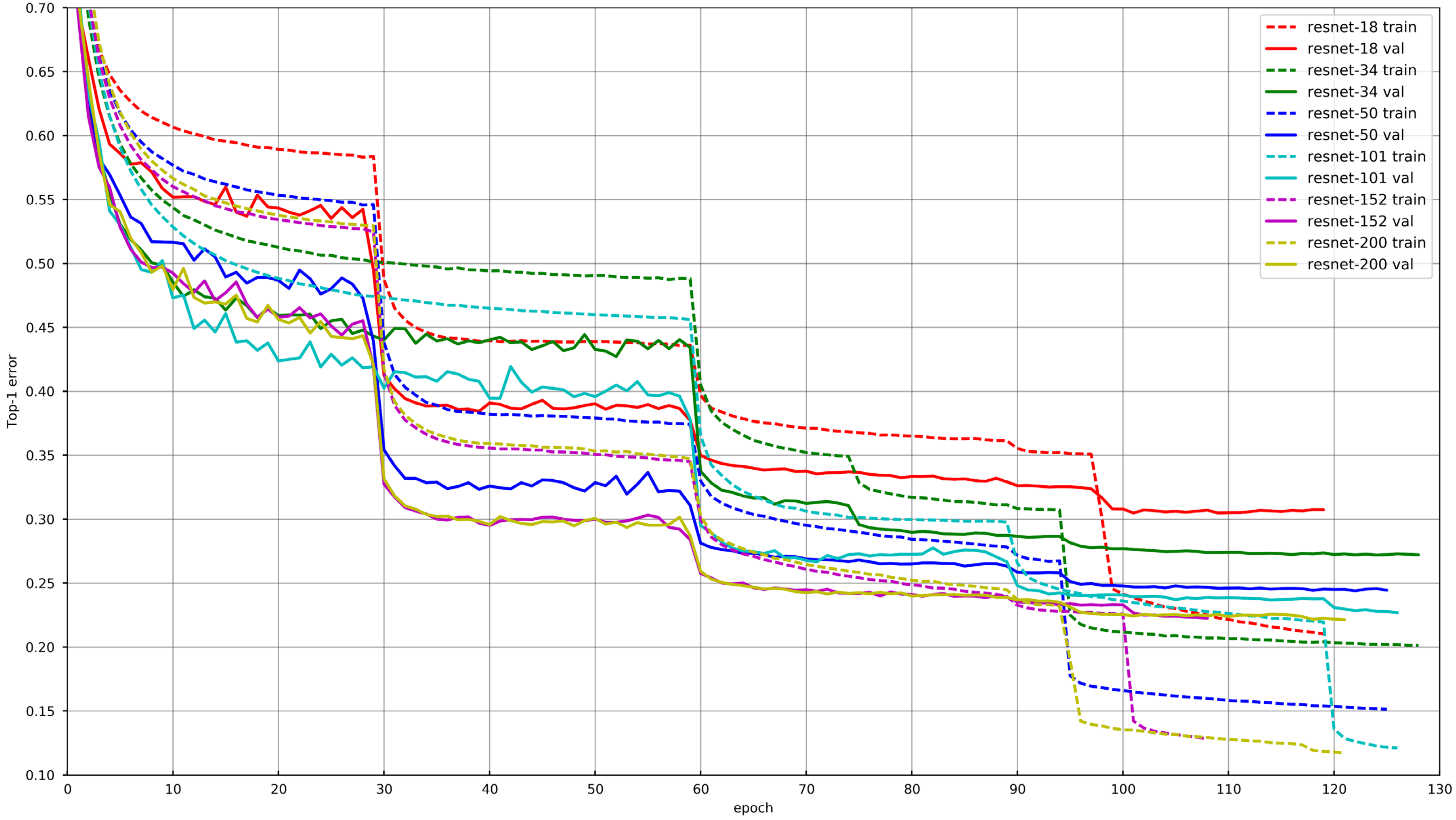
[11.11. Learning Rate Scheduling — Dive into Deep Learning 0.17.0 documentation (d2l.ai)]

Step Decay



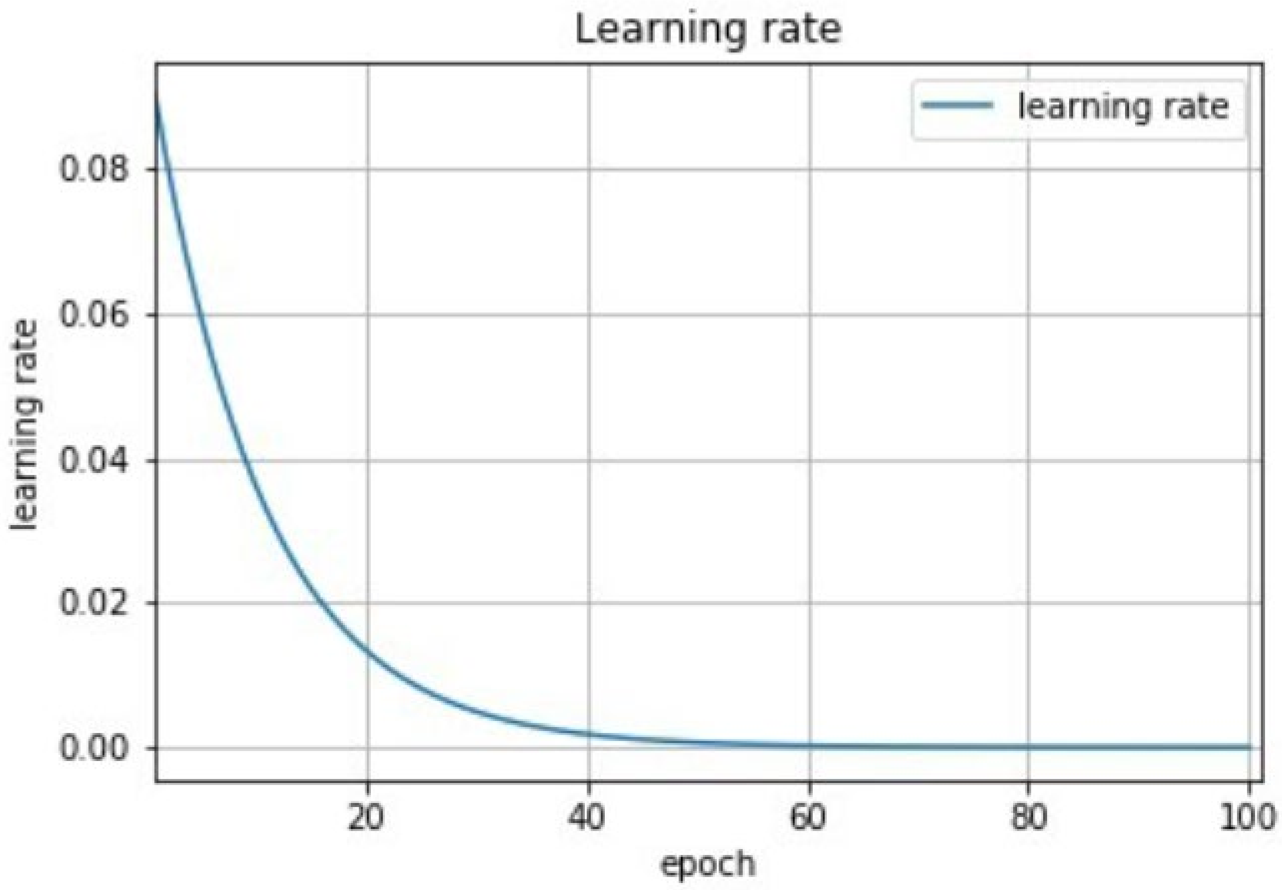
[Learning Rate Schedules and Adaptive Learning Rate Methods for Deep Learning | by Suki Lau | Towards Data Science]

Step Decay



[Model Zoo - ResNet MXNet Model]

Exponential Decay

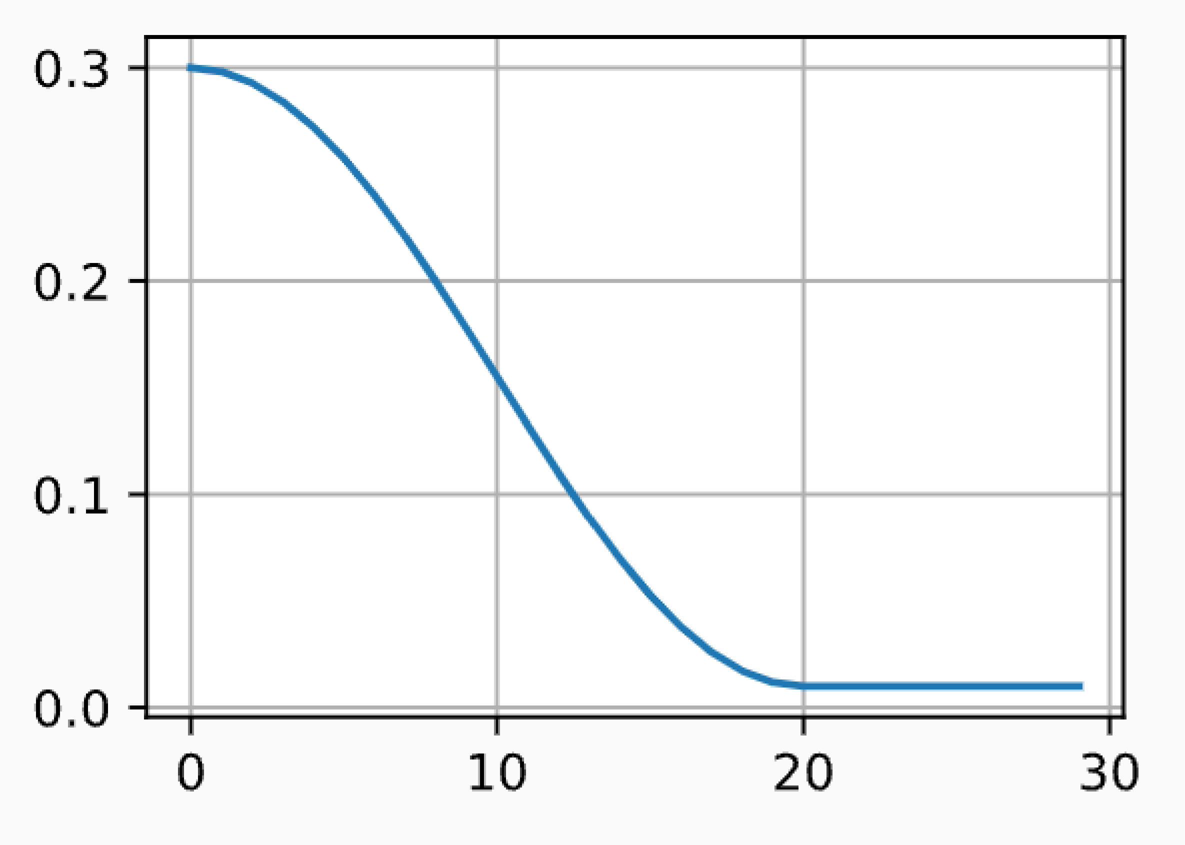


Learning Rate Schedules and Adaptive Learning Rate Methods for Deep Learning | by Suki Lau | Towards Data Science

Cosine Scheduler

- We might not want to decrease the learning rate too drastically in the beginning
- We might want to refine the solution at the end

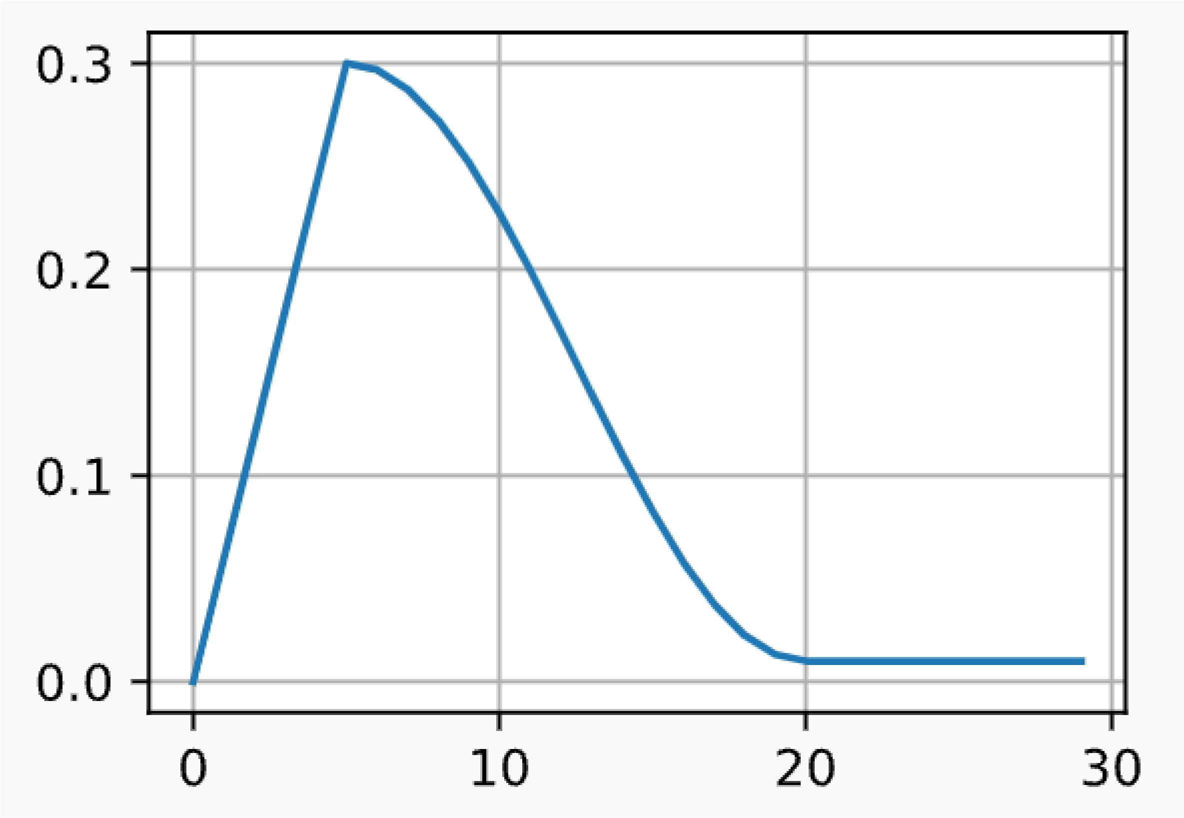
$$\eta_t = \eta_T + \frac{\eta_0 - \eta_T}{2}(1 + \cos(\pi t/T))$$



11.11. Learning Rate Scheduling — Dive into Deep Learning 0.17.0 documentation (d2l.ai)

Warmup

- Initialization is important
- By choosing sufficiently small learning rate to prevent divergence in the beginning



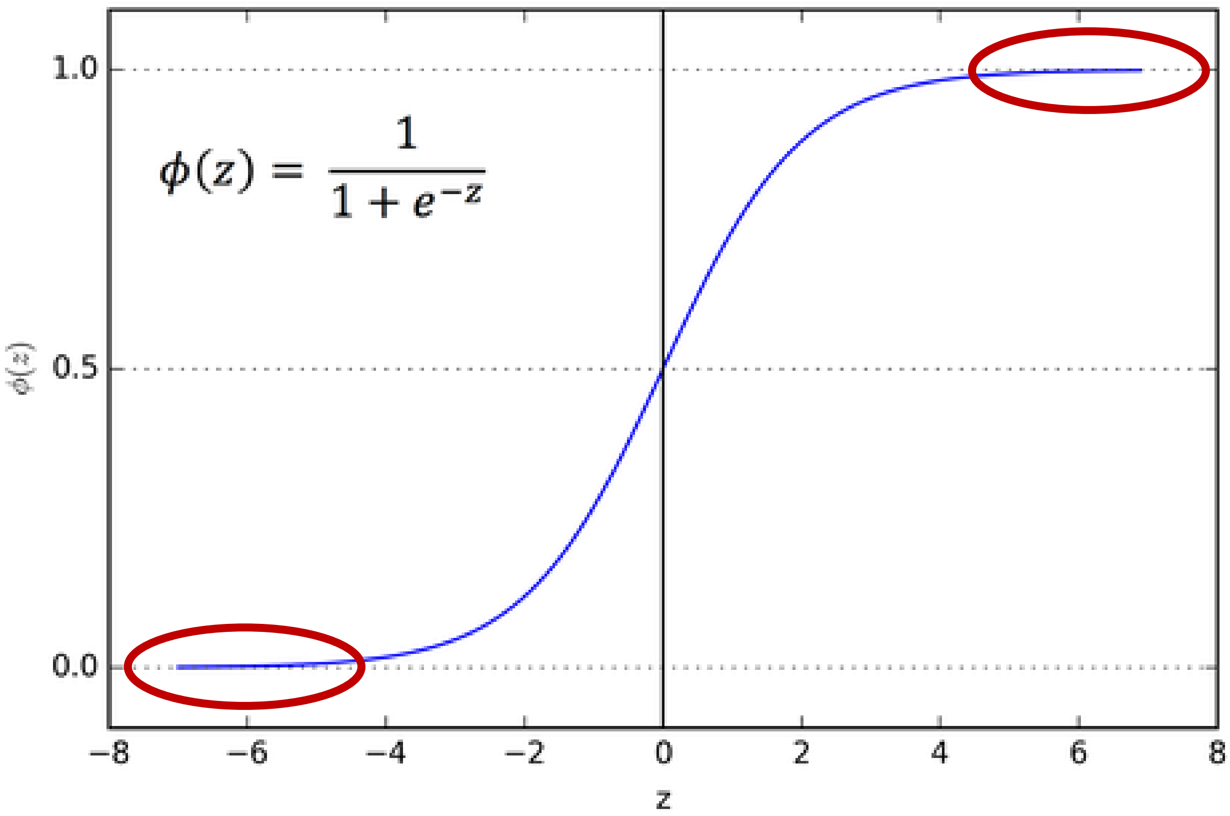
[11.11. Learning Rate Scheduling — Dive into Deep Learning 0.17.0 documentation (d2l.ai)]

Activation Functions

Sigmoid (a.k.a logistic)

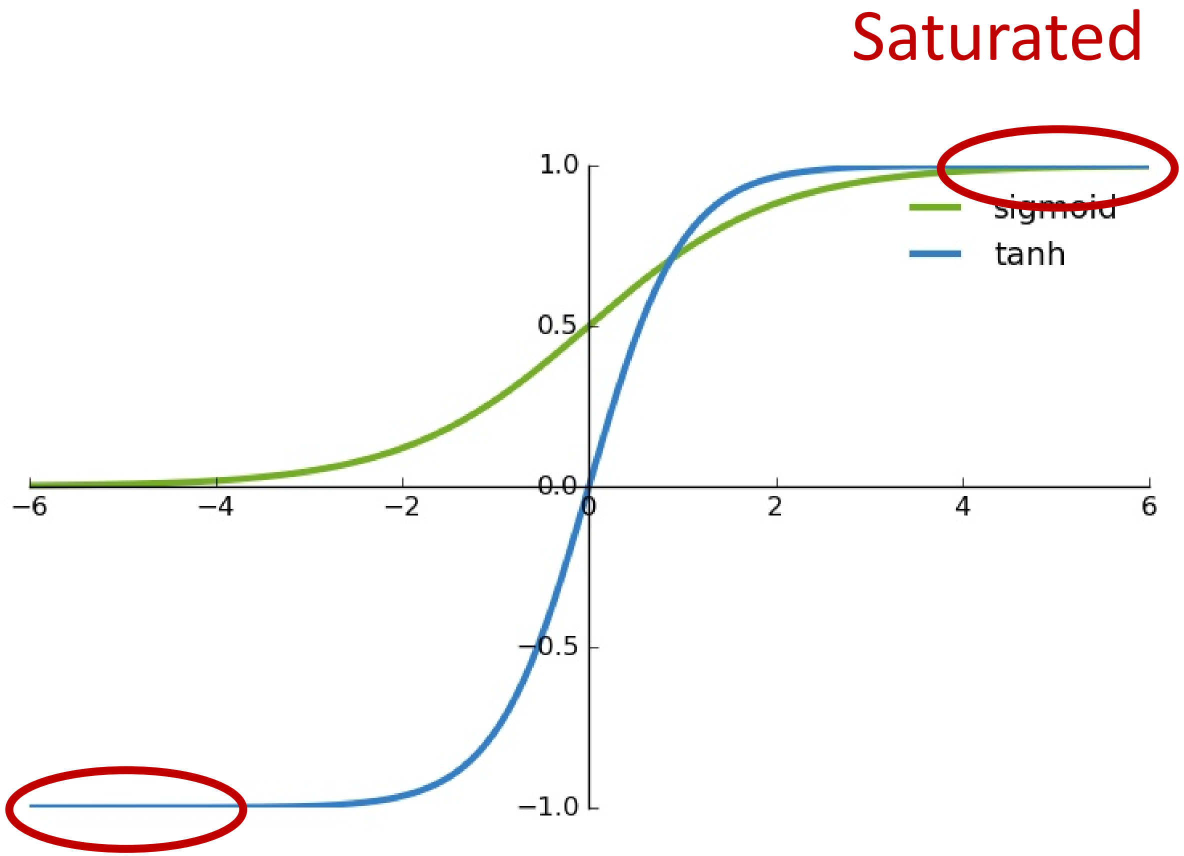
$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

Saturated



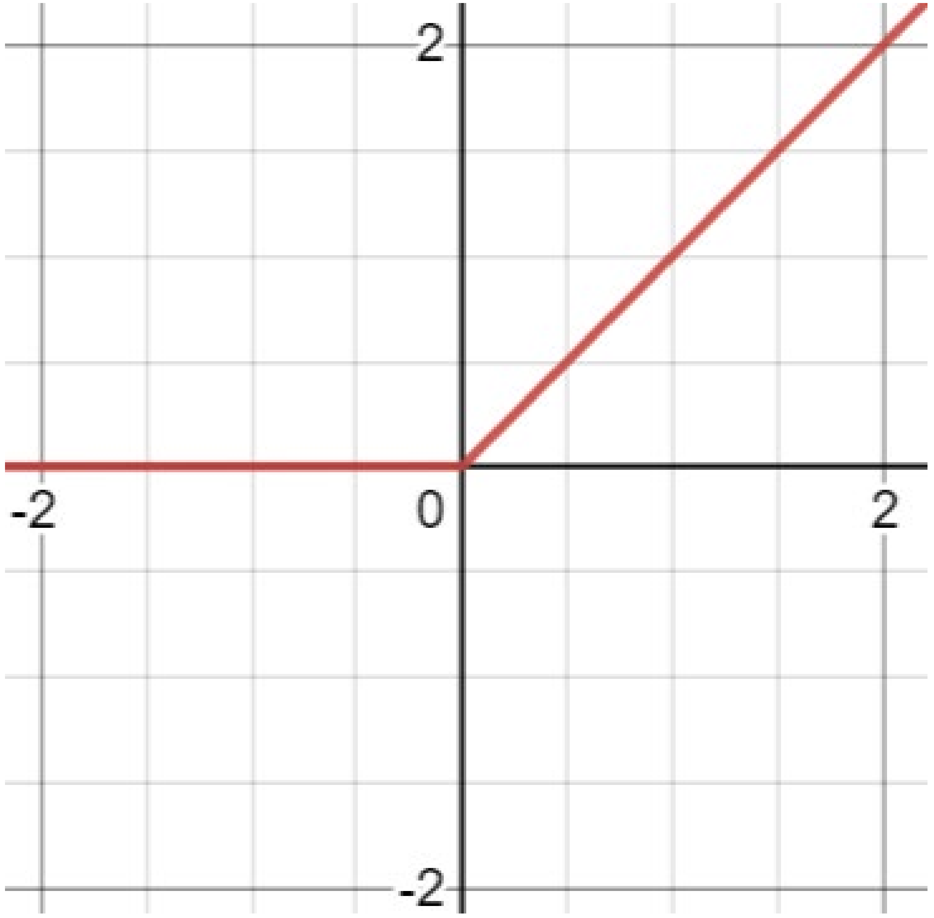
Tanh

$$\sigma(\mathrm{z}) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}$$



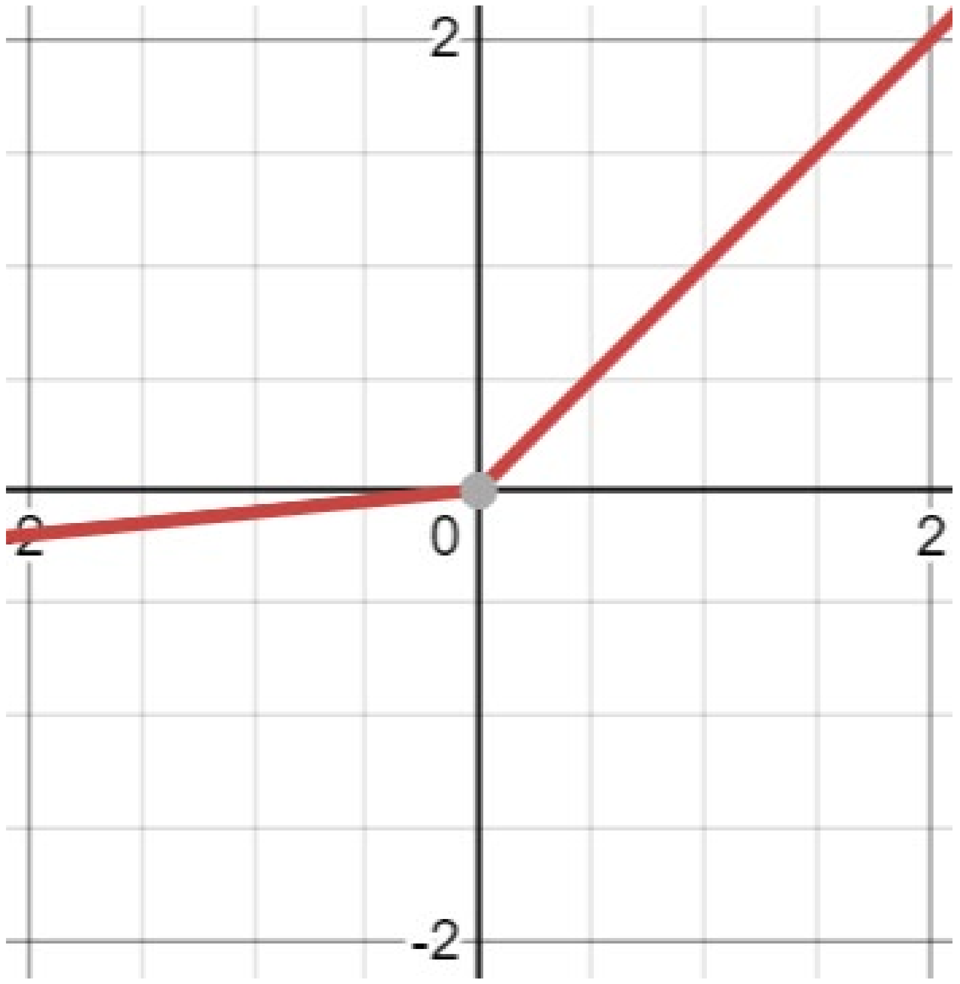
ReLU

$$\sigma(z) = \max(z, 0)$$



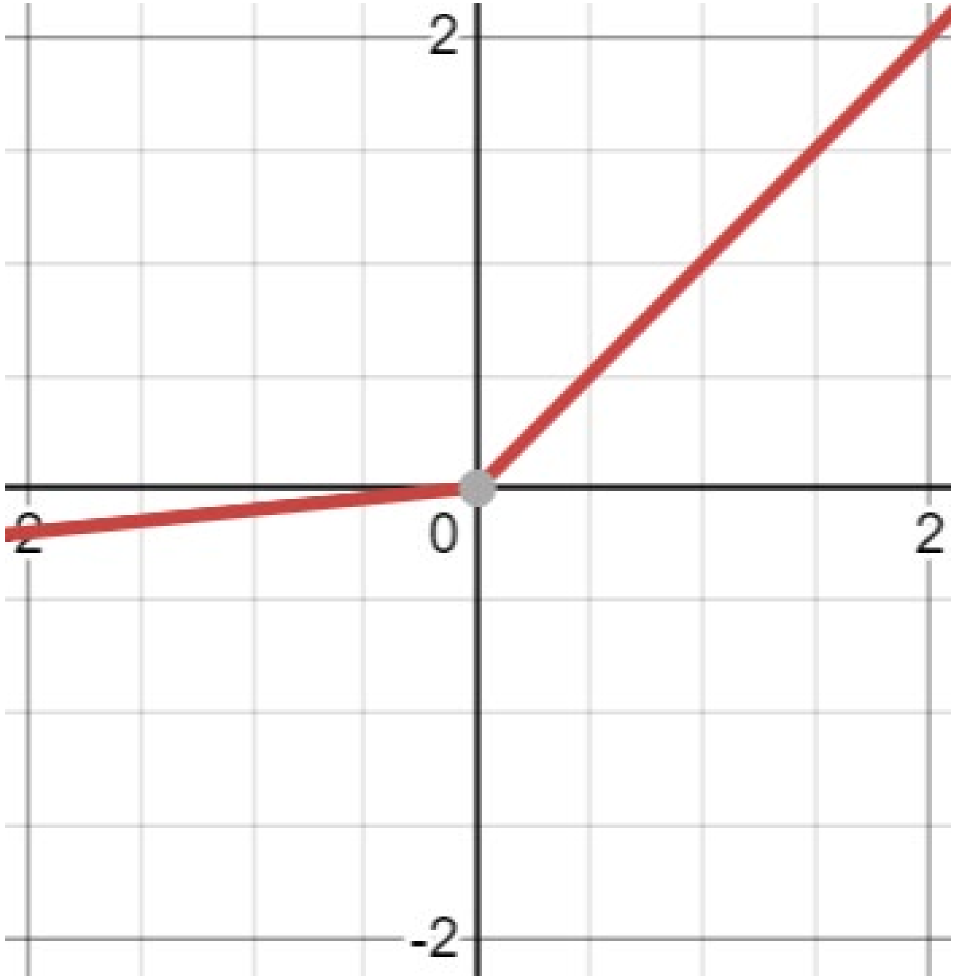
LeakyReLU

$$\sigma(z) = \max(z, 0.01z)$$



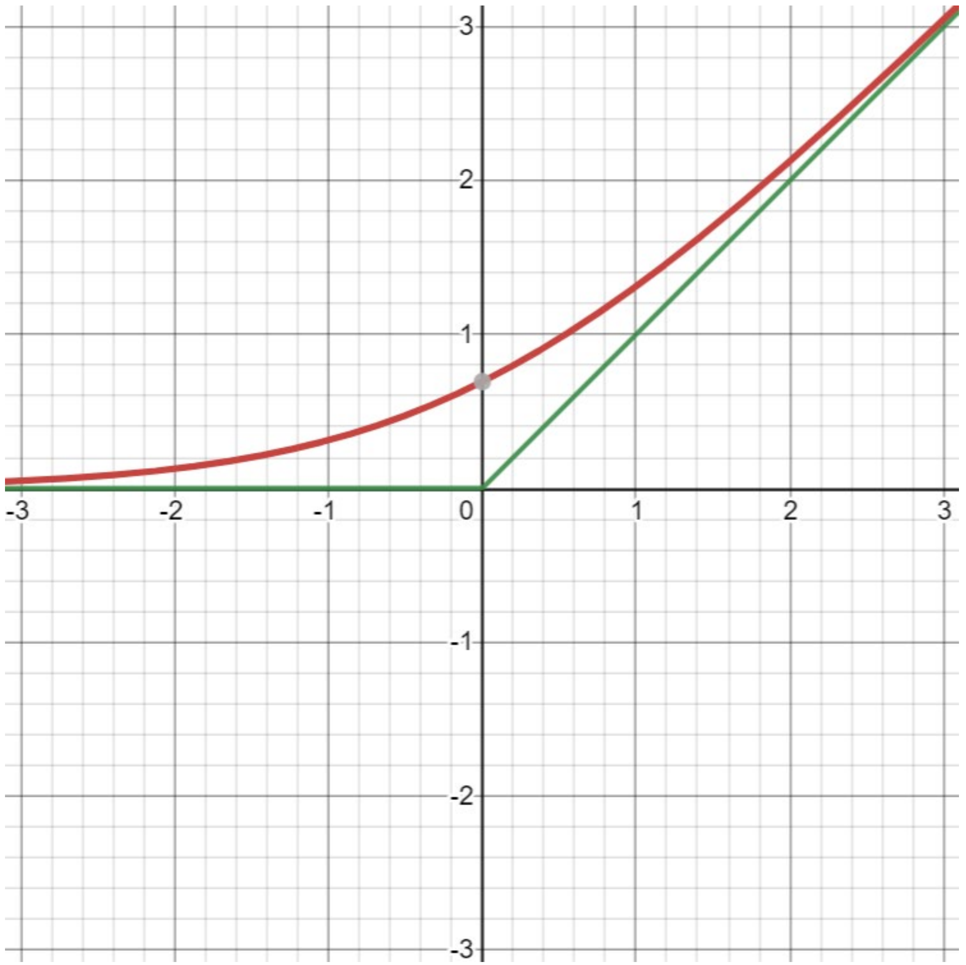
PReLU

$$\sigma(z) = \begin{Bmatrix} z, & z \geq 0 \\ az, & z < 0 \end{Bmatrix}$$



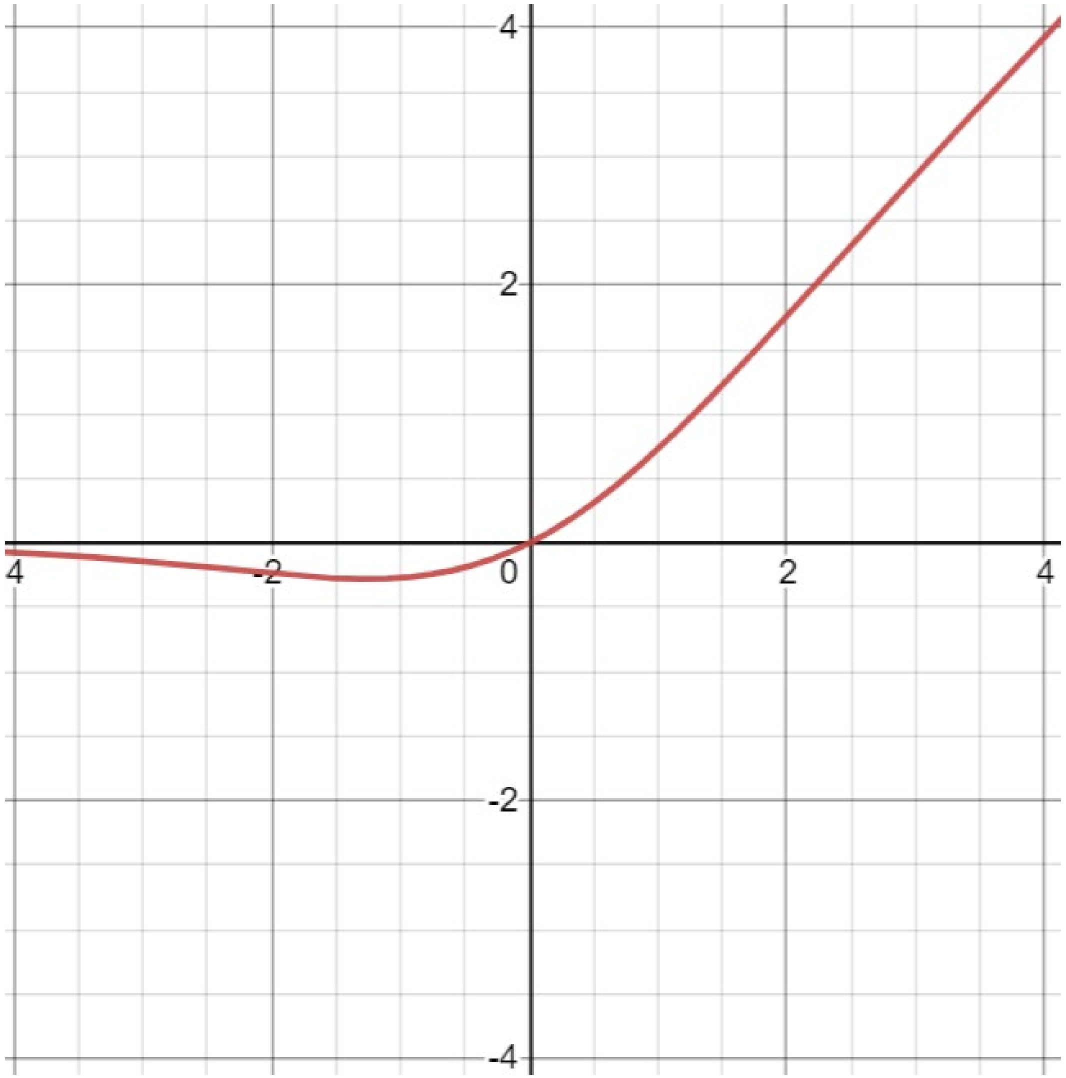
SoftPlus

$$\sigma(\mathrm{z}) = \ln(1 + \exp(x))$$



Swish

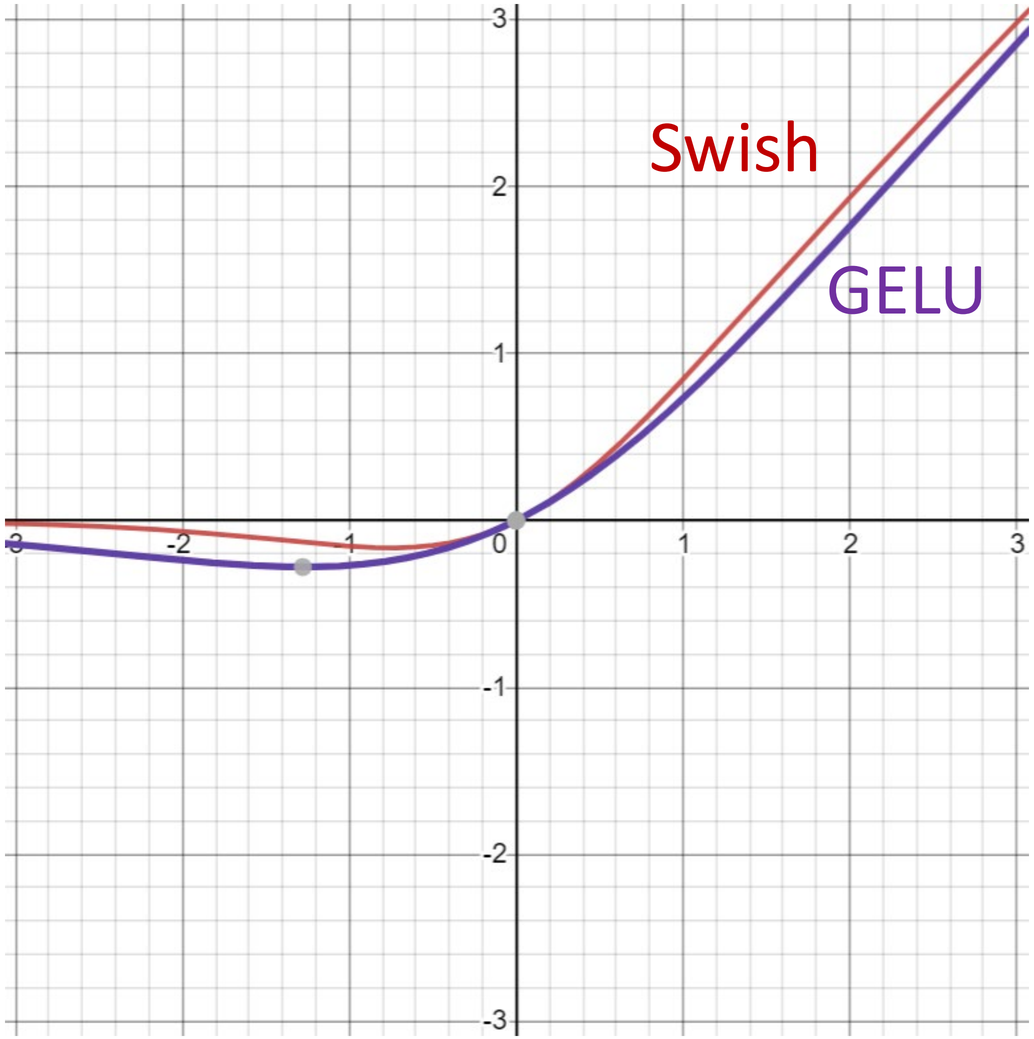
$$\sigma(\mathrm{z}) = \frac{x}{1 + \exp(-x)}$$



GELU

- Used in GPT3
- Recent MLP architectures

$$\sigma(z) = \frac{x}{1 + \exp(-1.702x)}$$



Sine

- Used in implicit representation

$$\sigma(\mathrm{z}) = \sin(x)$$