

# Computer Vision; Image Classification; Large Networks



[YouTube Playlist](#)

**Maziar Raissi**

**Assistant Professor**

Department of Applied Mathematics

University of Colorado Boulder

[maziar.raissi@colorado.edu](mailto:maziar.raissi@colorado.edu)



Boulder

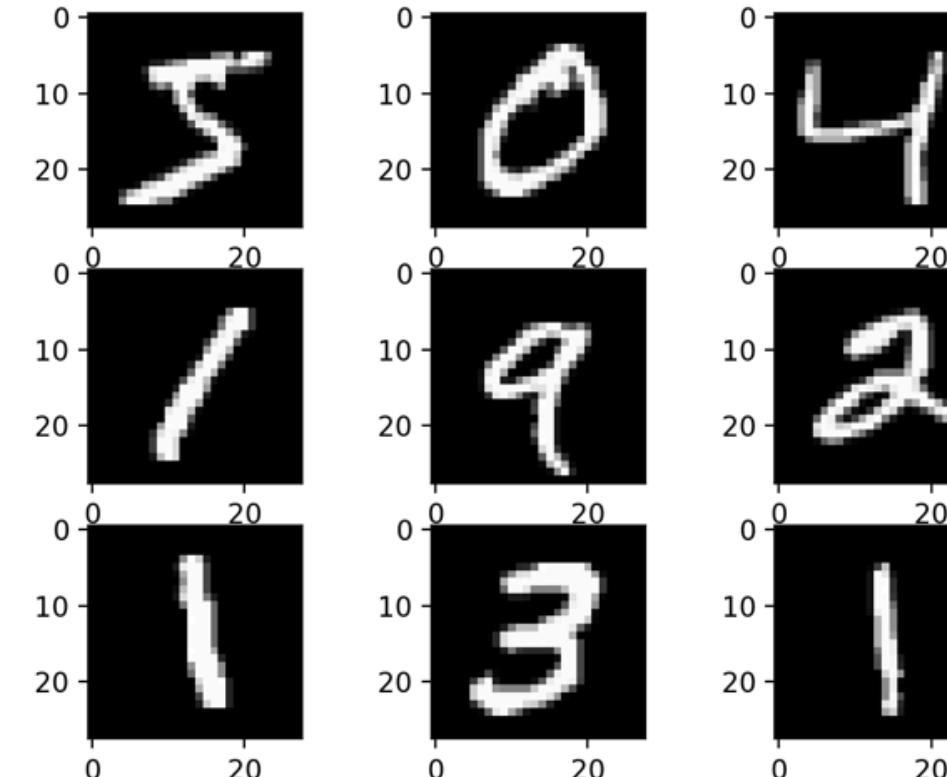
# Introduction to Convolutional Neural Networks



Question: What is an image?

Answer: An image is a tensor.

$$X \in \mathbb{R}^{H \times W \times C} \rightarrow \text{an image}$$



Question: What is a pixel?

Answer: Each pixel is a vector.

$$X(i, j) \in \mathbb{R}^C, i = 1, \dots, H, j = 1, \dots, W$$

$$X(i, j, c) \in \{0, 1, \dots, 255\} \rightarrow 256 = 2^8 \text{ numbers}$$

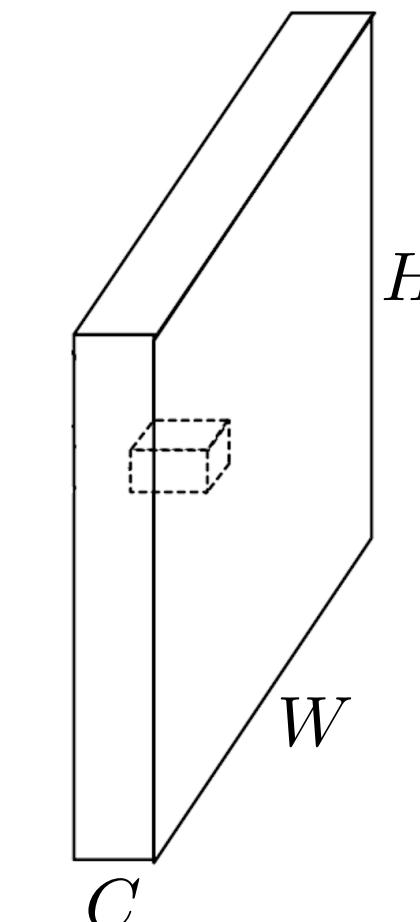
$X = X/255 \rightarrow$  normalize

**Training Data**

$$\{(X_n, y_n) : n = 1, \dots, N\}$$

$$X_n \in \mathbb{R}^{H \times W \times C} \rightarrow \text{input image}$$

$$y_n \in \{1, 2, \dots, K\} \rightarrow \text{labels}$$



$C = 3$  (RGB)

$C = 1$  (Black & White)

**Feature Learning**

$$g_\theta : X \in \mathbb{R}^{H \times W \times C} \mapsto x \in \mathbb{R}^d$$

$$\{\underbrace{(g_\theta(X_n), y_n)}_{x_n} : n = 1, \dots, N\} \rightarrow \text{tabular data (multinomial logistic regression)}$$

$g_\theta \rightarrow \text{CNN}$

The aim is to take an image as input and turn it into a vector!

**1 × 1 Convolution**

$$Y(i, j) = \underbrace{W}_{\in \mathbb{R}^{C' \times C}} \underbrace{X(i, j)}_{\in \mathbb{R}^C}, \forall (i, j) \rightarrow \text{parameter sharing}$$

**3 × 3 Convolution**

$$Y(i, j) = \sum_{i' \in \{-1, 0, 1\}} \sum_{j' \in \{-1, 0, 1\}} \underbrace{W(2 + i', 2 + j')}_{\in \mathbb{R}^{C' \times C}} \underbrace{X(si + i', sj + j')}_{\in \mathbb{R}^C}, \forall (i, j) \rightarrow \text{parameter sharing}$$

$s \rightarrow$  stride

The convolution operation takes as input an image and outputs another image (i.e., the feature maps).

**Pooling (e.g., max- and average-pooling)**

$$Z(i, j) = \underbrace{\max_{i' \in \{-1, 0, 1\}} \max_{j' \in \{-1, 0, 1\}}}_{\in \mathbb{R}^{C'}} \underbrace{Y(si + i', sj + j')}_{\in \mathbb{R}^{C'}}$$

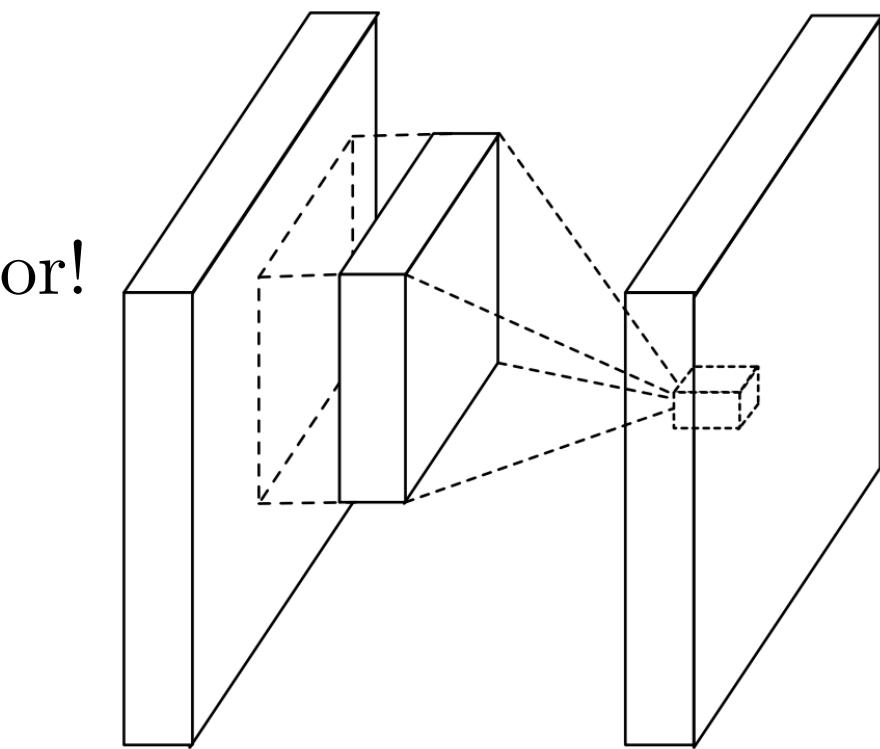
no additional parameters

**Vector Representation**

$x \rightarrow$  global average pooling

**Multi-Layer Perceptron (MLP)**

$$\text{softmax}(W \text{ReLU}(Vx + a) + b)$$

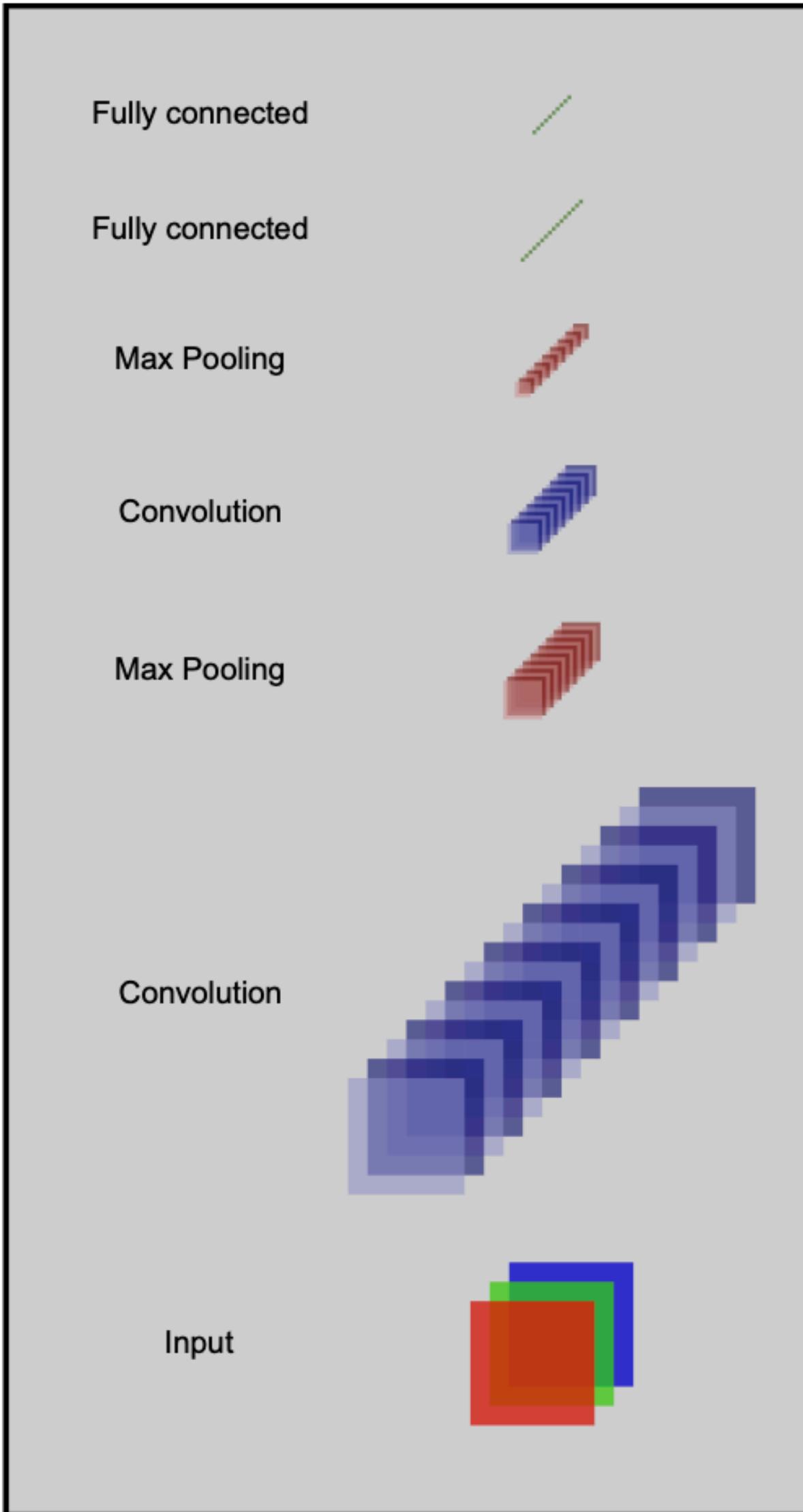


Play Dough

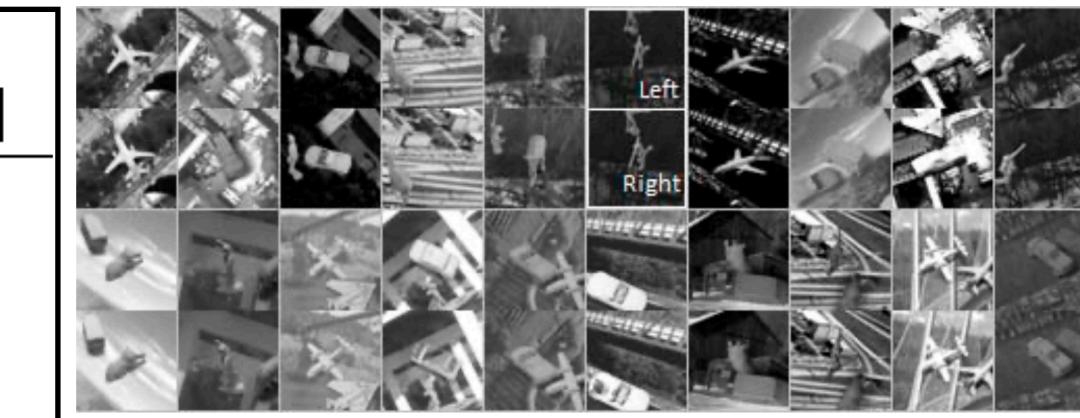


Boulder

# Multi-column Deep Neural Networks for Image Classification



Method	Results on MNIST dataset.	
	Paper	Error rate[%]
CNN	[32]	0.40
CNN	[26]	0.39
MLP	[5]	0.35
CNN committee	[6]	0.27
MCDNN	this	<b>0.23</b>



Twenty NORB stereo images  
(left image - up, right image - down)

Table 4. Average error rates of MCDNN for all experiments, plus results from the literature. \* case insensitive

Data (task)	MCDNN error [%]	Published results Error[%] and paper
all (62)	<b>11.63</b>	
digits (10)	<b>0.77</b>	3.71 [12] 1.88 [23]
letters (52)	<b>21.01</b>	30.91[16]
letters* (26)	<b>7.37</b>	13.00 [4] 13.66[16]
merged (37)	<b>7.99</b>	
uppercase (26)	<b>1.83</b>	10.00 [4] 6.44 [9]
lowercase (26)	<b>7.47</b>	16.00 [4] 13.27 [16]

This is the first time human-competitive results are reported on widely used computer vision benchmarks. On the MNIST handwriting benchmark, this method is the first to achieve near-human performance. On a traffic sign recognition benchmark it outperforms humans by a factor of two.

Dataset	Best result of others [%]	MCDNN [%]	Relative improv. [%]
MNIST	0.39	0.23	41
NIST SD 19	see Table 4	see Table 4	30-80
HWDB1.0 on.	7.61	5.61	26
HWDB1.0 off.	10.01	6.5	35
CIFAR10	18.50	11.21	39
traffic signs	1.69	0.54	72
NORB	5.00	2.70	46

→ Latin characters

→ Chinese characters

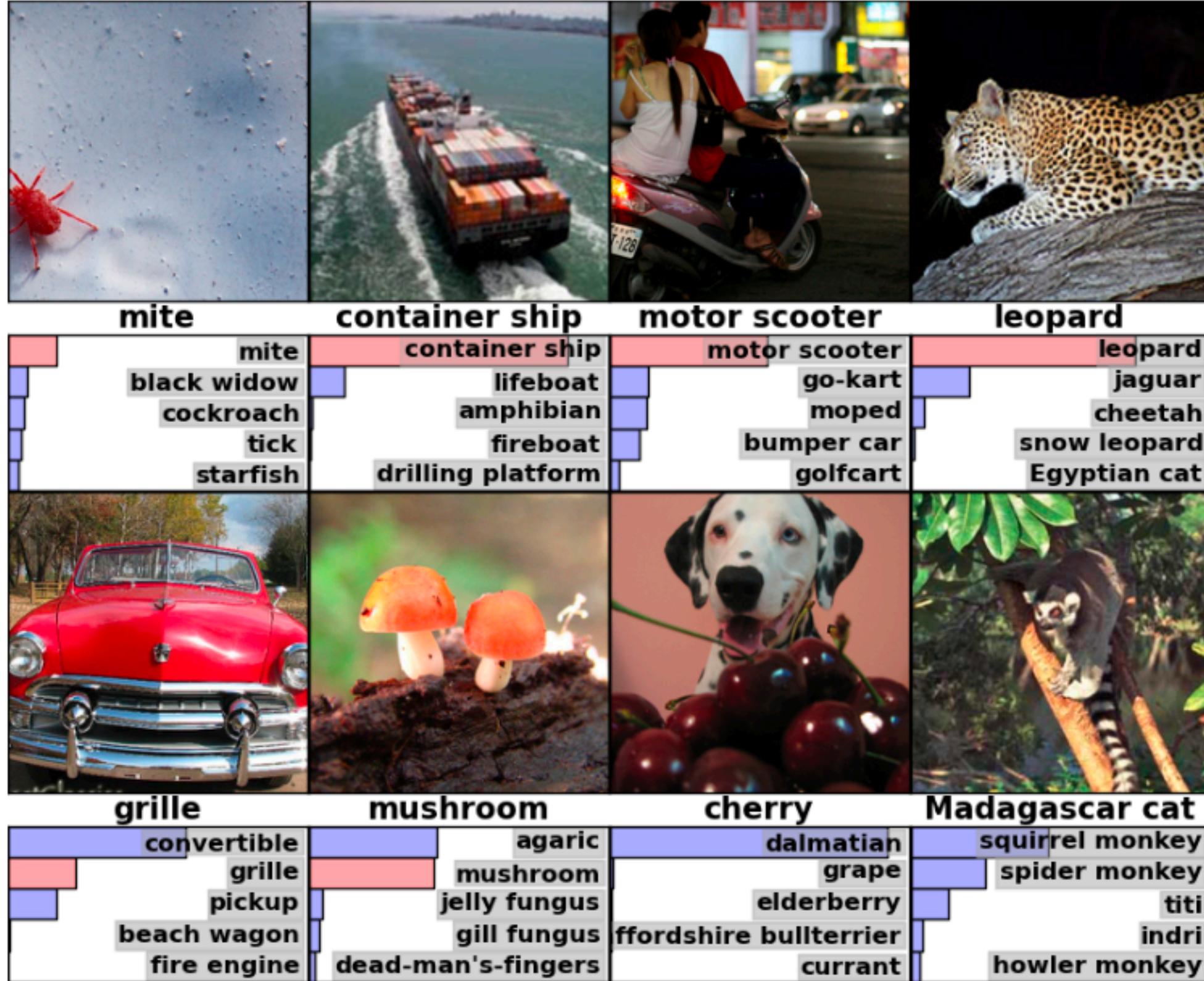


Boulder

# ImageNet Classification with Deep Convolutional Neural Networks

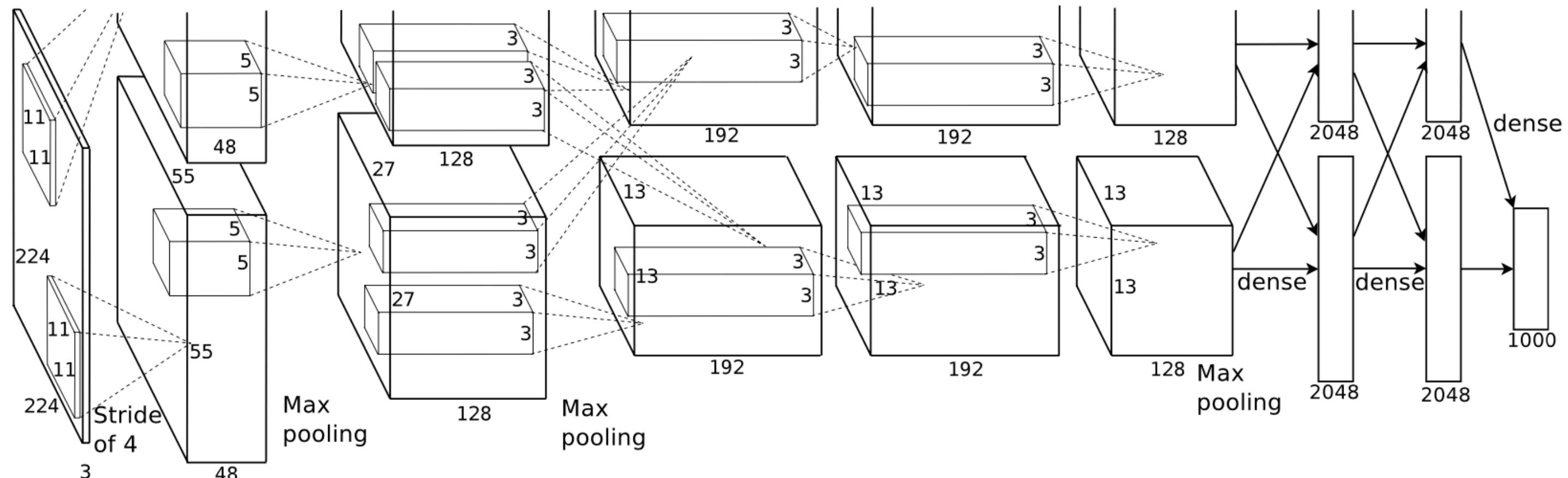


[YouTube Video](#)



"To learn about thousands of objects from millions of images, we need a model with a large learning capacity. However, the immense complexity of the object recognition task means that this problem cannot be specified even by a dataset as large as ImageNet, so our model should also have lots of prior knowledge to compensate for all the data we don't have."

"CNNs make strong and mostly correct assumptions about the nature of images (namely, **stationarity of statistics** and **locality of pixel dependencies**)."



**Architecture:**  $I \in \mathbb{R}^{224 \times 224 \times 3} \mapsto p \in \{p \in \mathbb{R}^{1000} : \sum_i p_i = 1, p_i \geq 0 \forall i\}$

**1x1 Conv:**  $X \in \mathbb{R}^{H \times W \times C} \mapsto Y \in \mathbb{R}^{H \times W \times F}$

$$Y(x, y) = WX(x, y), W \in \mathbb{R}^{F \times C}, X(x, y) \in \mathbb{R}^C, Y(x, y) \in \mathbb{R}^F$$

**3x3 Conv:**  $Y(x, y) = \sum_{i=1,2,3} \sum_{j=1,2,3} W(i, j)X(s(x-2)+i, s(y-2)+j), W \in \mathbb{R}^{3 \times 3 \times F \times C}$

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

$$\text{Loss: } \mathcal{L}(\theta) = - \sum_{n=1}^N \log(p_{i_n}(I_n))$$

Model	Top-1	Top-5
Sparse coding [2]	47.1%	28.2%
SIFT + FVs [24]	45.7%	25.7%
CNN	<b>37.5%</b>	<b>17.0%</b>

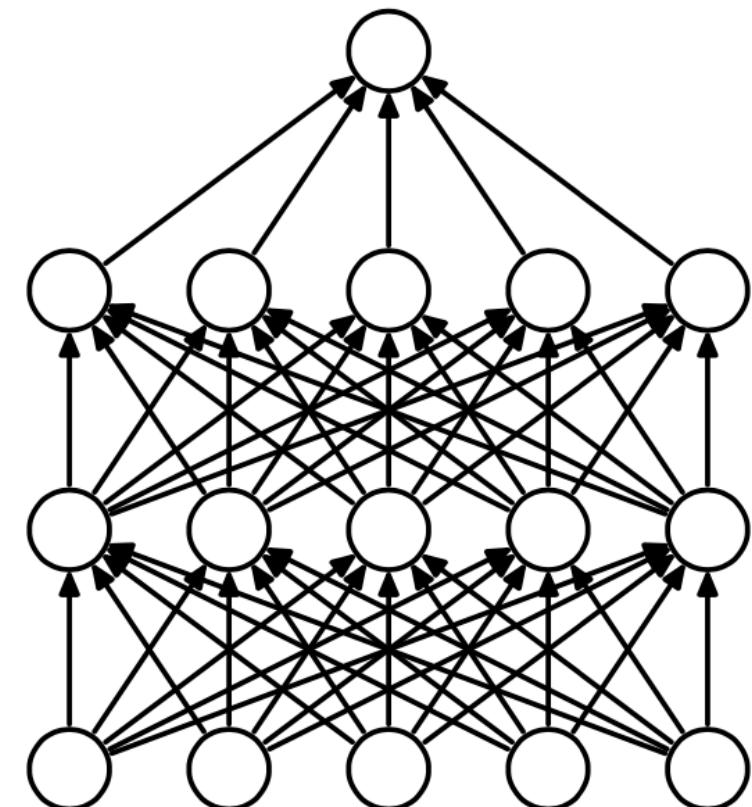


Boulder

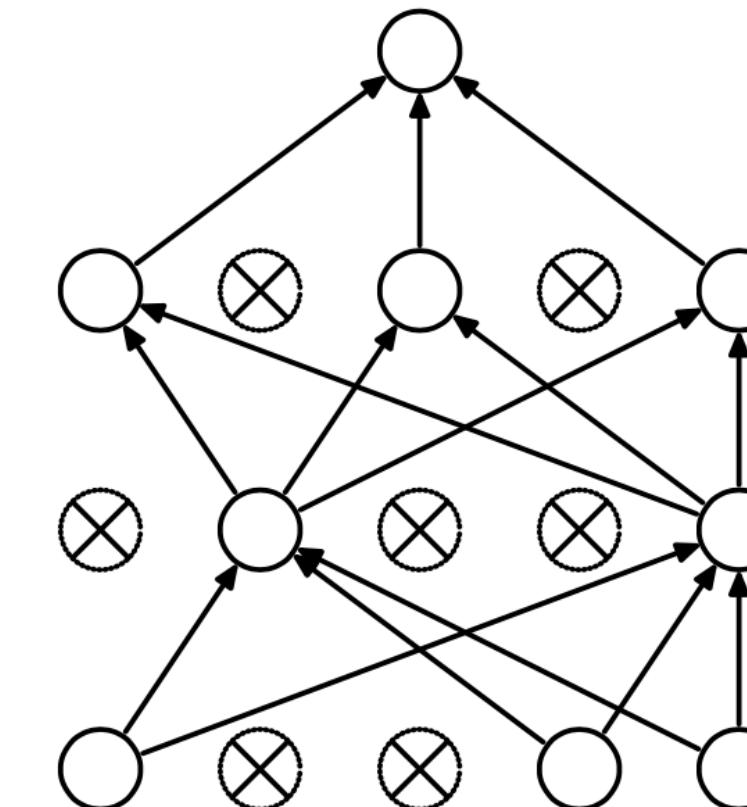
# Dropout: A Simple Way to Prevent Neural Networks from Overfitting



[YouTube Video](#)

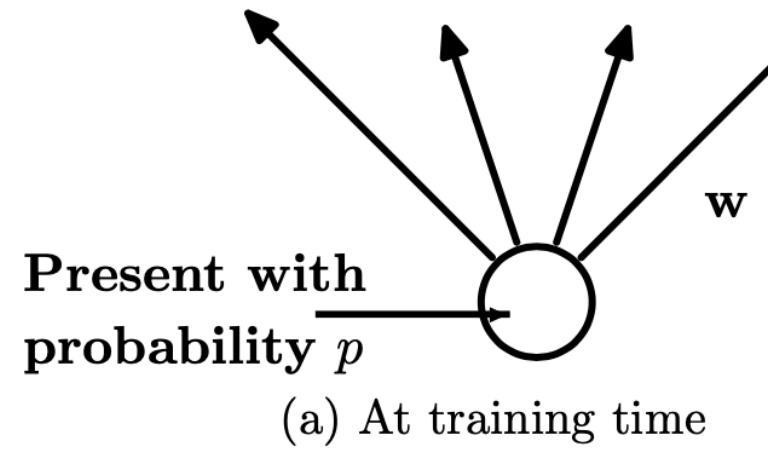


(a) Standard Neural Net

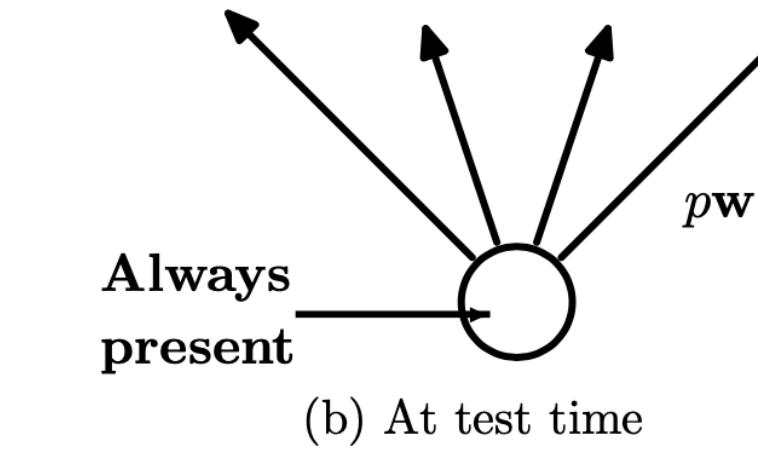


(b) After applying dropout.

1. Prevent overfitting (regularization technique)
2. Approximately combine exponentially many different neural network architectures efficiently



Present with probability  $p$

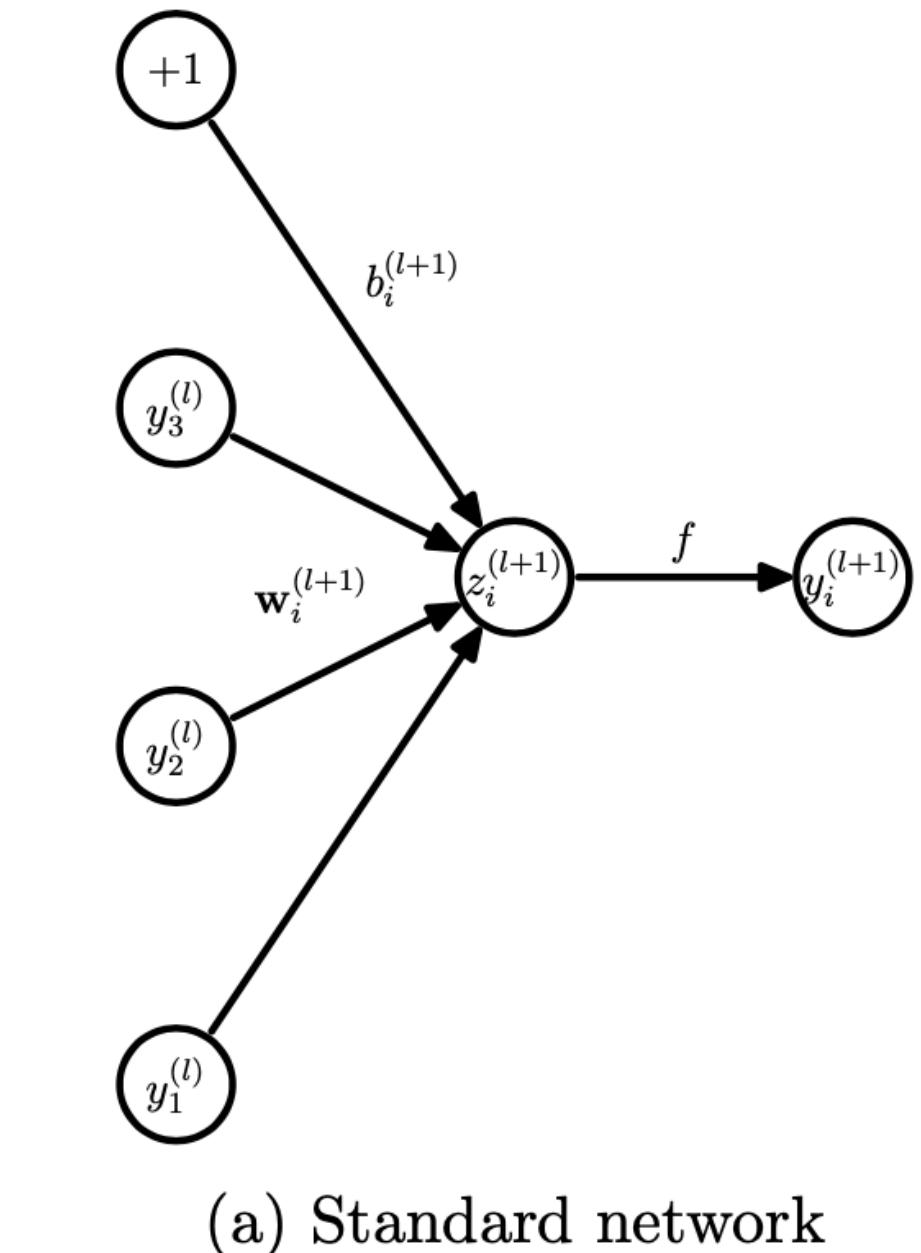


Always present

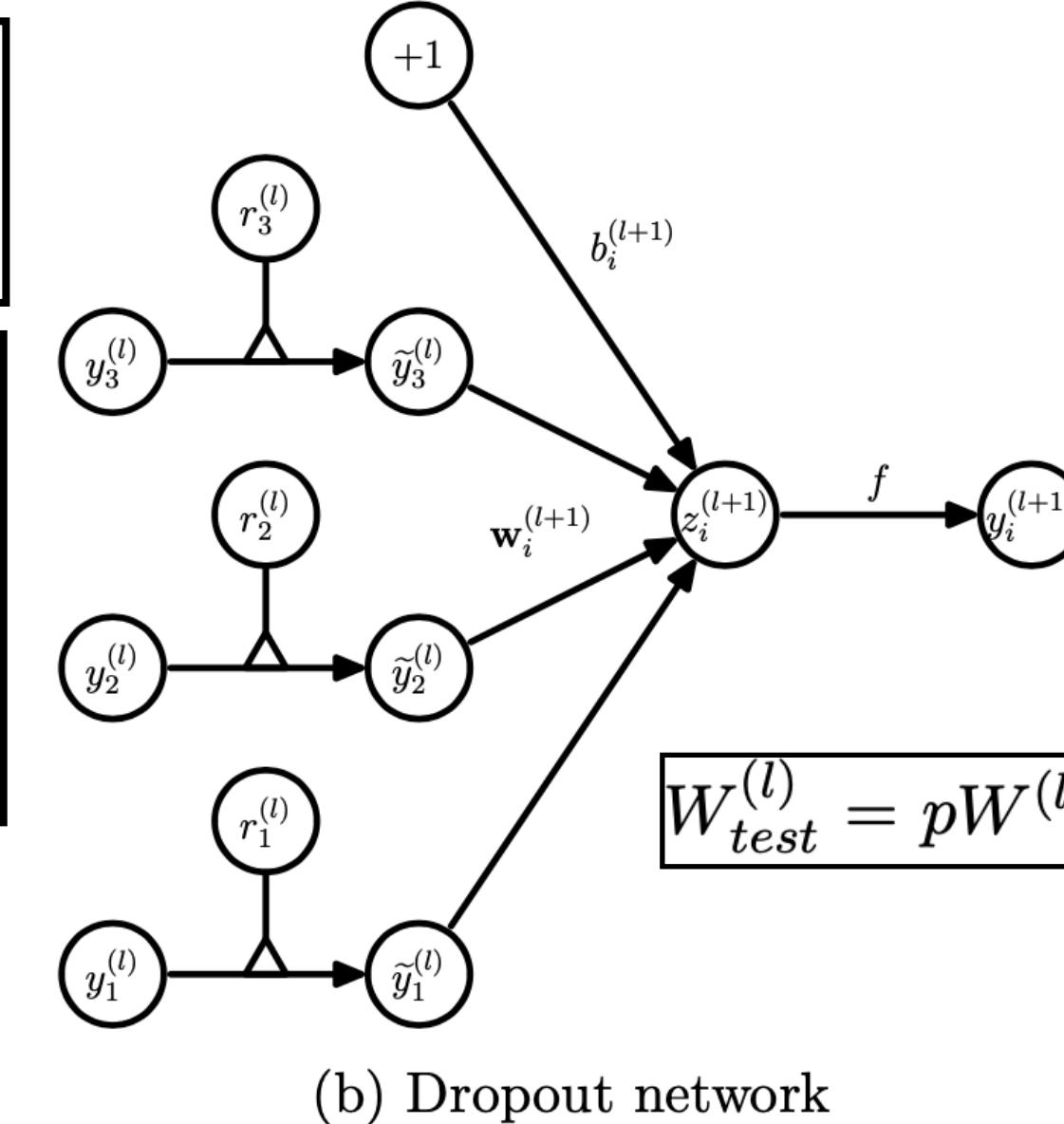
Each hidden unit in a neural network trained with dropout must learn to work with a randomly chosen sample of other units  
 $L \rightarrow$  number of hidden layers  
 $l \in \{1, \dots, L\}$

$$l \in \{1, \dots, L\}$$

$$\begin{aligned} z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \mathbf{y}^l + b_i^{(l+1)}, \\ y_i^{(l+1)} &= f(z_i^{(l+1)}), \\ r_j^{(l)} &\sim \text{Bernoulli}(p), \\ \tilde{\mathbf{y}}^{(l)} &= \mathbf{r}^{(l)} * \mathbf{y}^{(l)}, \\ z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \tilde{\mathbf{y}}^{(l)} + b_i^{(l+1)}, \\ y_i^{(l+1)} &= f(z_i^{(l+1)}). \end{aligned}$$



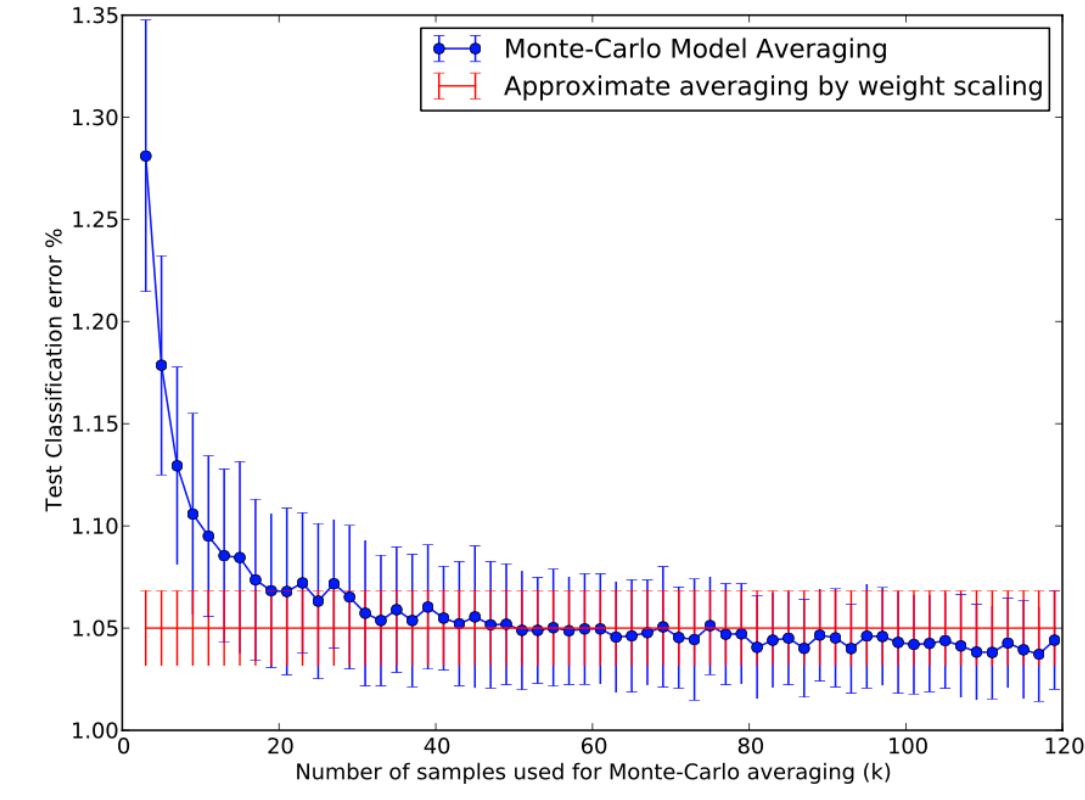
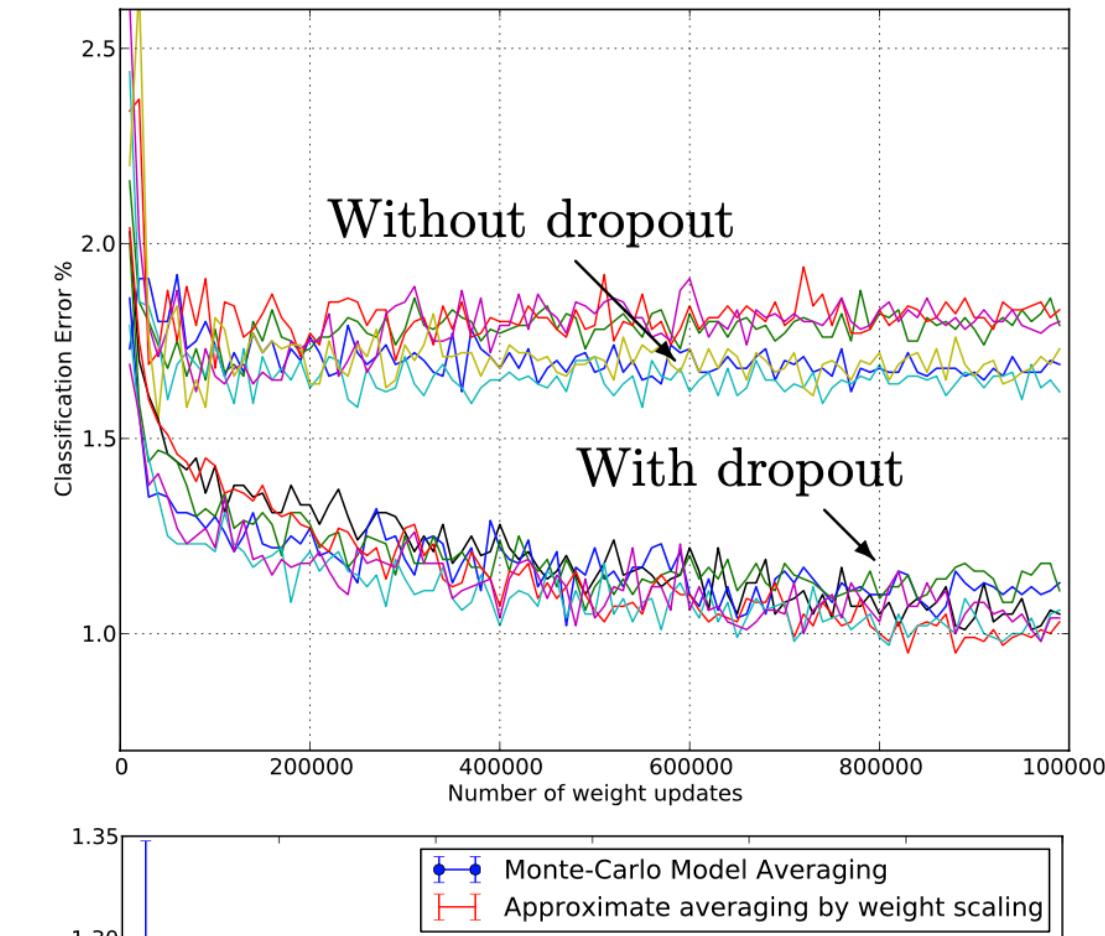
(a) Standard network



(b) Dropout network

$\|\mathbf{w}\|_2 \leq c$   
max-norm regularization

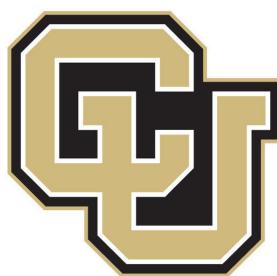
Dropout with linear regression is equivalent to ridge regression.



Data Set	Domain	Dimensionality	Training Set	Test Set
MNIST	Vision	784 ( $28 \times 28$ grayscale)	60K	10K
SVHN	Vision	3072 ( $32 \times 32$ color)	600K	26K
CIFAR-10/100	Vision	3072 ( $32 \times 32$ color)	60K	10K
ImageNet (ILSVRC-2012)	Vision	65536 ( $256 \times 256$ color)	1.2M	150K
TIMIT	Speech	2520 (120-dim, 21 frames)	1.1M frames	58K frames
Reuters-RCV1	Text	2000	200K	200K
Alternative Splicing	Genetics	1014	2932	733

Hinton, Geoffrey E., et al. "Improving neural networks by preventing co-adaptation of feature detectors." *arXiv preprint arXiv:1207.0580* (2012).

Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *The journal of machine learning research* 15.1 (2014): 1929-1958.



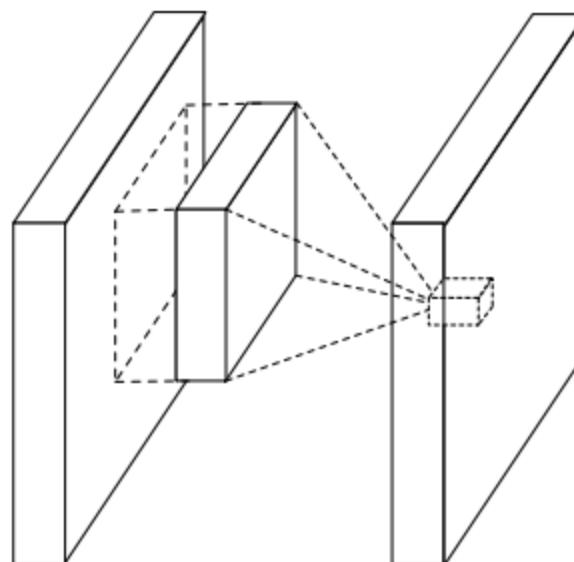
Boulder



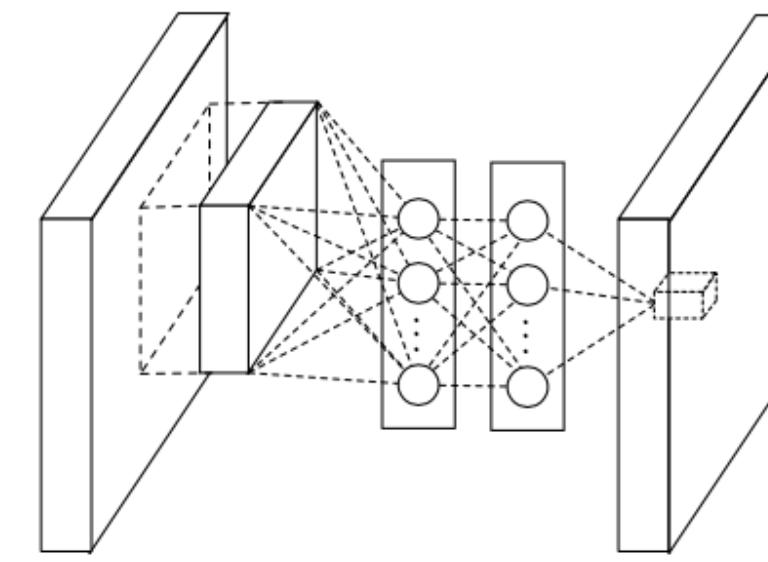
[YouTube Playlist](#)

# Network In Network

## 1) MLP Convolution Layers (1x1 Convolutions)



(a) Linear convolution layer



(b) Mlpconv layer

$$\begin{aligned}
 f_{i,j,k_1}^1 &= \max(w_{k_1}^1{}^T x_{i,j} + b_{k_1}, 0). && \text{Input patch centered at location } (i, j) \\
 &\vdots && \text{ReLU activation function} \\
 f_{i,j,k_n}^n &= \max(w_{k_n}^n{}^T f_{i,j}^{n-1} + b_{k_n}, 0).
 \end{aligned}$$

$k_\ell$  is used to index the channels of the feature map

## 2) Global Average Pooling

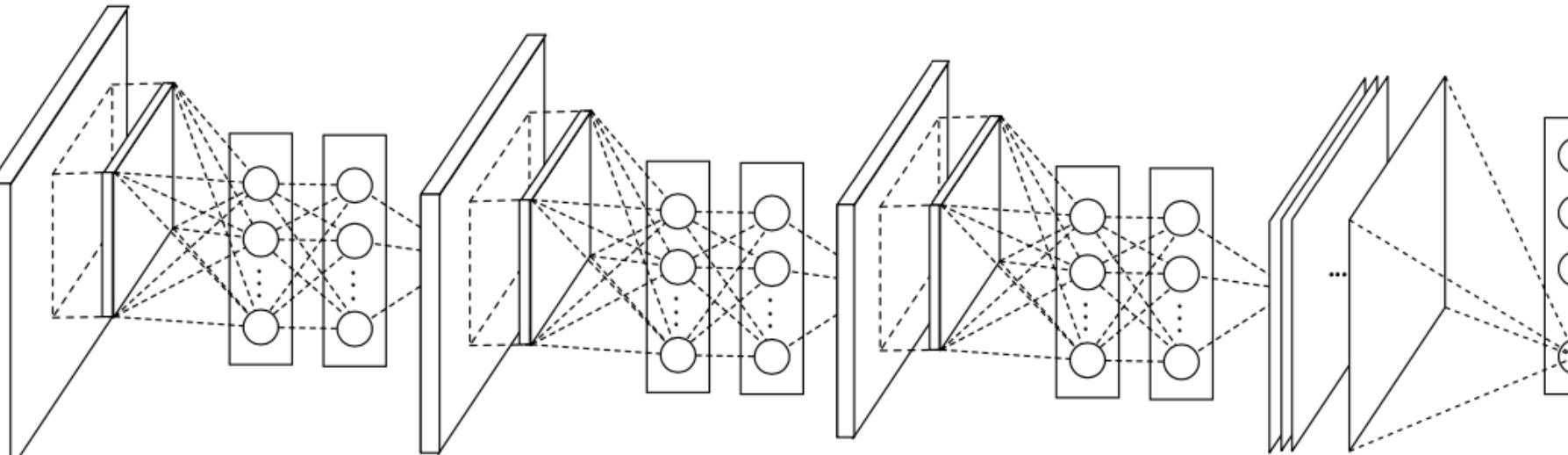


Table 3: Test set error rates for SVHN of various methods.

Method	Test Error
Stochastic Pooling [11]	2.80%
Rectifier + Dropout [18]	2.78%
Rectifier + Dropout + Synthetic Translation [18]	2.68%
Conv. maxout + Dropout [8]	2.47%
NIN + Dropout	2.35%
Multi-digit Number Recognition [19]	2.16%
<b>DropConnect [15]</b>	<b>1.94%</b>

Table 1: Test set error rates for CIFAR-10 of various methods.

Method	Test Error
Stochastic Pooling [11]	15.13%
CNN + Spearmint [14]	14.98%
Conv. maxout + Dropout [8]	11.68%
<b>NIN + Dropout</b>	<b>10.41%</b>
CNN + Spearmint + Data Augmentation [14]	9.50%
Conv. maxout + Dropout + Data Augmentation [8]	9.38%
DropConnect + 12 networks + Data Augmentation [15]	9.32%
<b>NIN + Dropout + Data Augmentation</b>	<b>8.81%</b>

Table 2: Test set error rates for CIFAR-100 of various methods.

Method	Test Error
Learned Pooling [16]	43.71%
Stochastic Pooling [11]	42.51%
Conv. maxout + Dropout [8]	38.57%
Tree based priors [17]	36.85%
<b>NIN + Dropout</b>	<b>35.68%</b>

Table 4: Test set error rates for MNIST of various methods.

Method	Test Error
2-Layer CNN + 2-Layer NN [11]	0.53%
Stochastic Pooling [11]	0.47%
NIN + Dropout	0.47%
<b>Conv. maxout + Dropout [8]</b>	<b>0.45%</b>

Table 5: Global average pooling compared to fully connected layer.

Method	Testing Error
mlpconv + Fully Connected	11.59%
mlpconv + Fully Connected + Dropout	10.88%
mlpconv + Global Average Pooling	10.41%



Boulder

# Very Deep Convolutional Networks for Large-Scale Image Recognition



YouTube Video

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input ( $224 \times 224$ RGB image)					
conv3-64	conv3-64 <b>LRN</b>	conv3-64 <b>conv3-64</b>	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 <b>conv3-128</b>	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 <b>conv1-256</b>	conv3-256 conv3-256 <b>conv3-256</b>	conv3-256 conv3-256 <b>conv3-256</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

## Local Response Normalization (LRN):

Create competition for big activities amongst neuron outputs computed using different kernels.

$$b_{x,y}^i = a_{x,y}^i / \left( k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a_{x,y}^j)^2 \right)^\beta$$

activity of a neuron computed by applying kernel  $i$   
at position  $(x, y)$  and then applying the ReLU nonlinearity

$$k = 2, n = 5, \alpha = 10^{-4}, \beta = 0.75$$

## Data Augmentation:

- 1) Image translations (random crops) and horizontal reflection
- 2) Altering the intensities of the RGB channels in training images (object identity is invariant to changes in the intensity and color of the illumination )

$$I_{xy} = [I_{xy}^R, I_{xy}^G, I_{xy}^B]^T + [p_1, p_2, p_3] [\underbrace{\alpha_1 \lambda_1, \alpha_2 \lambda_2, \alpha_3 \lambda_3}_\text{eigenvector and eigenvalue of the } 3 \times 3 \text{ covariance matrix of RGB pixel values}]^T$$

## 3) Scale jittering

Each training image is individually rescaled by randomly sampling  $S$  from a certain range  $[S_{\min}, S_{\max}]$   
 $S$  is the smallest side of an isotropically-rescaled training image.

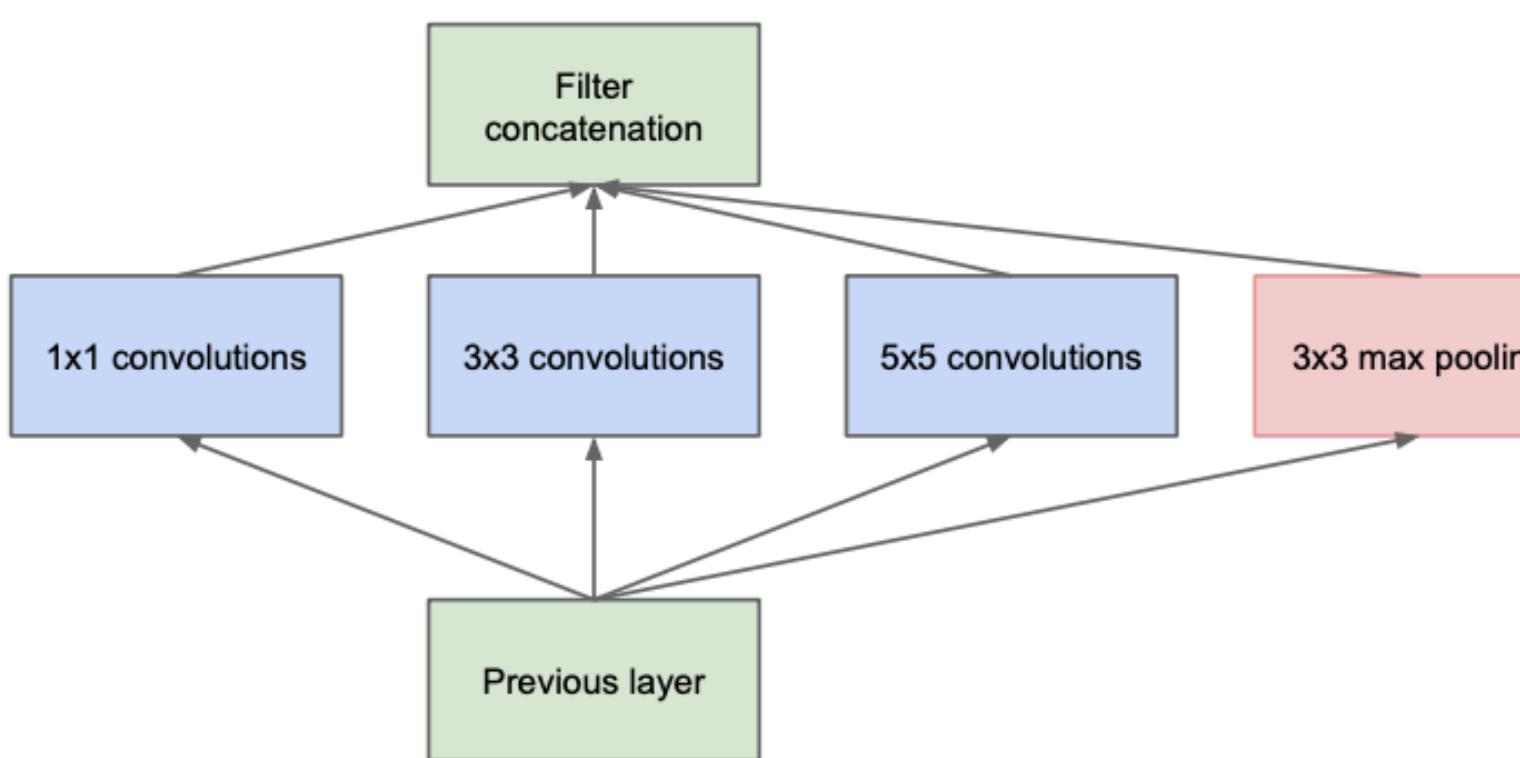


Boulder

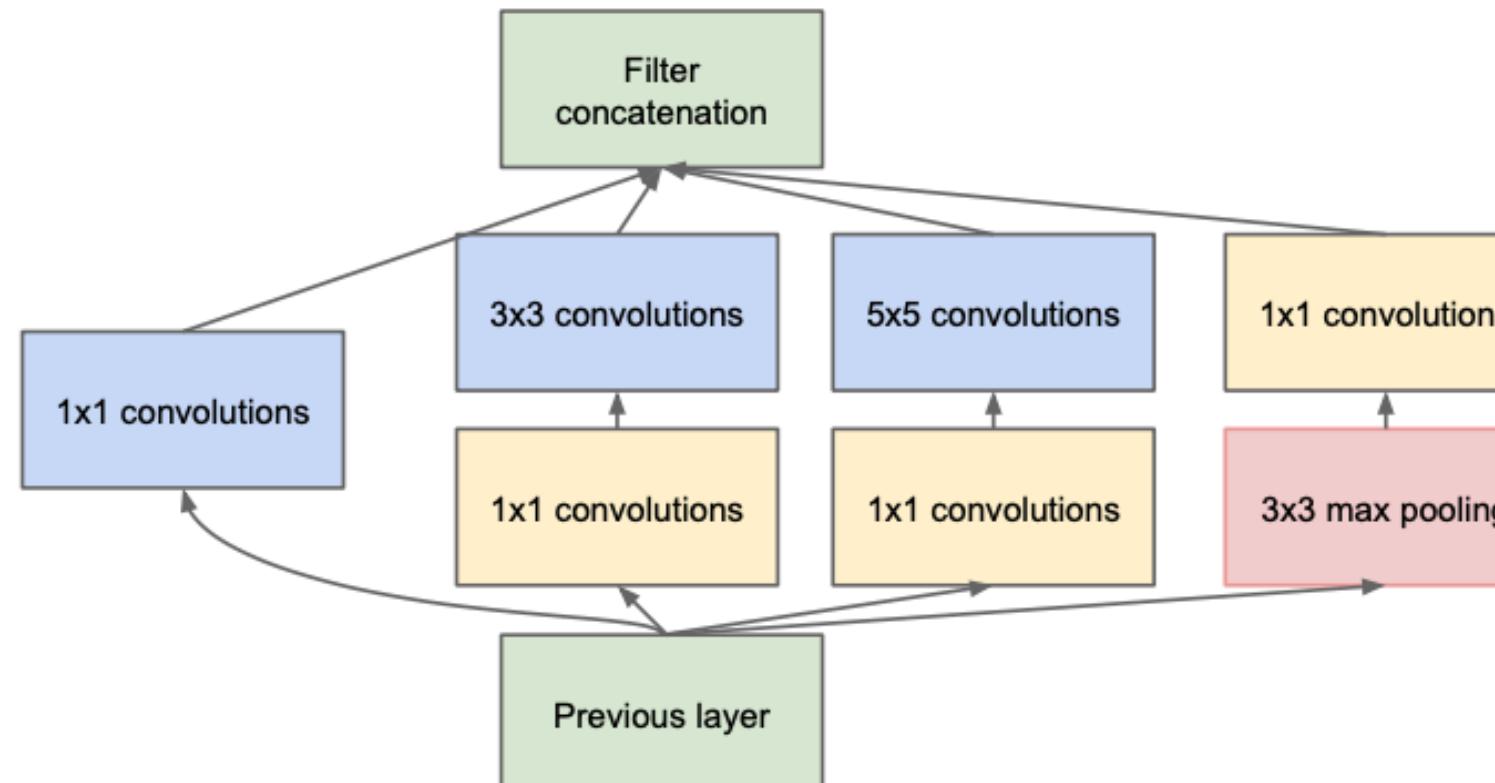


# Going Deeper with Convolutions

[YouTube Playlist](#)



(a) Inception module, naïve version



(b) Inception module with dimensionality reduction

type	patch size/stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Table 1: GoogLeNet incarnation of the Inception architecture.

# Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift


[YouTube Playlist](#)

statistics of layer's inputs change during training

$$z = g(Wu + b)$$

replace

**Training**

$$\text{BN}(x; \gamma, \beta, \underbrace{x_1, x_2, \dots, x_m}_{\text{mini-batch}}) = \gamma \frac{x - \overbrace{\mu(x_1, \dots, x_m)}^{\text{mean}}}{\sqrt{\overbrace{\sigma^2(x_1, \dots, x_m)}^{\text{variance}}} + \epsilon} + \beta$$

$$x \in \{x_1, x_2, \dots, x_m\}$$

$$x_i, \gamma, \beta, \mu, \sigma \in \mathbb{R}^d$$

All operations are element-wise (dimension-wise)

The gradients flow through  $\mu$  and  $\sigma^2$

**Inference**

$$\text{BN}(x; \gamma, \beta) = \gamma \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta$$

$$\mu = \mathbb{E}_{x_1, \dots, x_m} [\mu(x_1, \dots, x_m)]$$

$$\sigma^2 = \frac{m}{m-1} \mathbb{E}_{x_1, \dots, x_m} [\sigma^2(x_1, \dots, x_m)]$$

**Convolution**

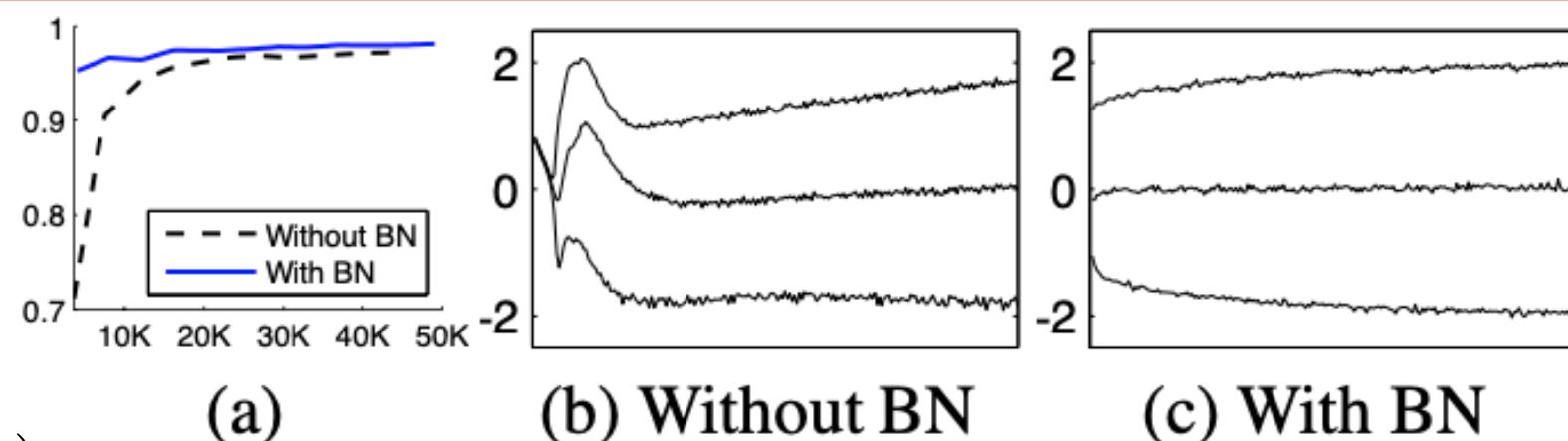
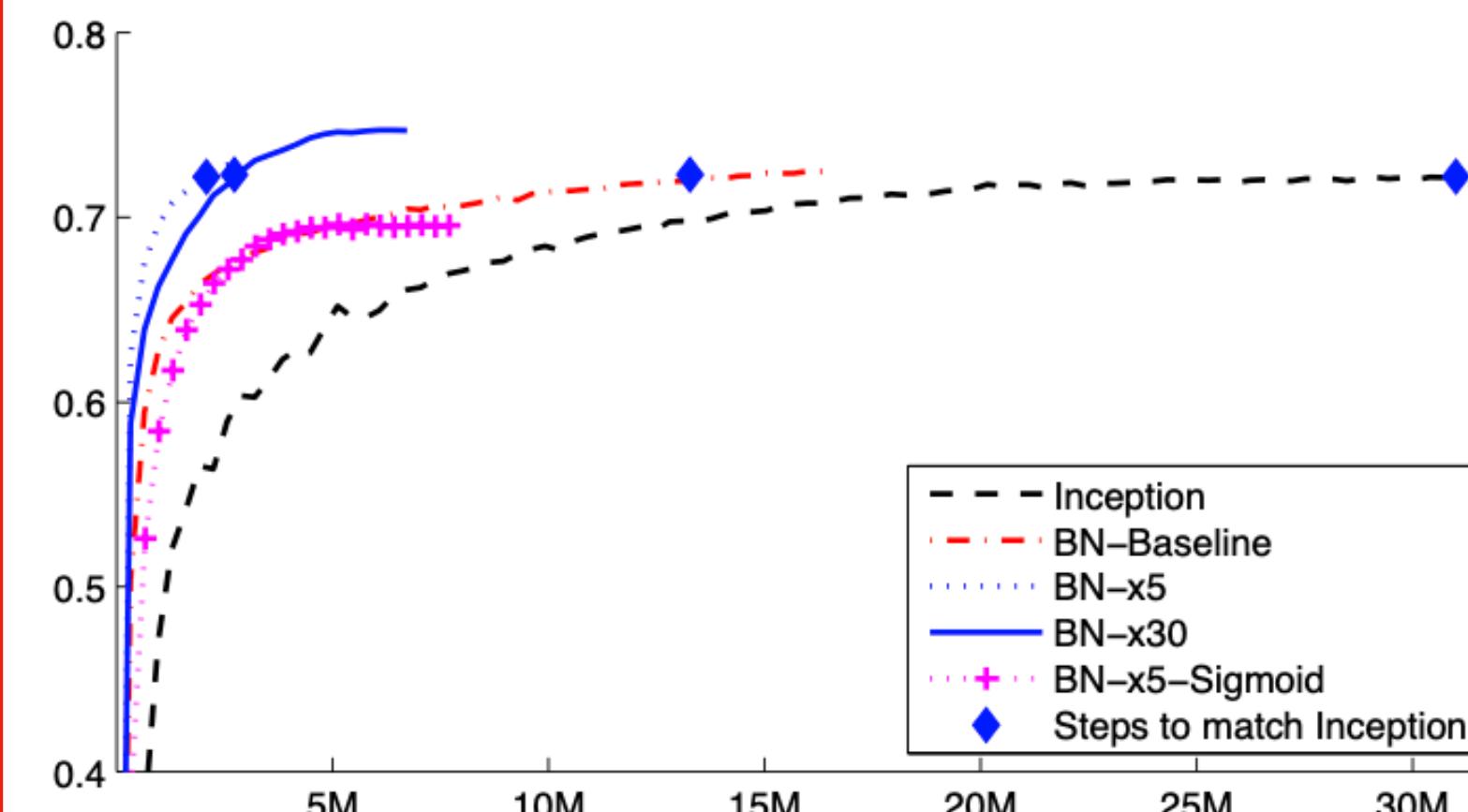
$$X \in \mathbb{R}^{p \times q \times d}$$

$$m' = mpq \rightarrow \text{effective mini-batch size}$$

Batch-norm is done per feature map

Benefits of batch normalization:

1. higher learning rate
2. less sensitive to initialization
3. less sensitive to activation functions
4. regularization effects
5. preserve gradient magnitudes (maybe)



Evolution of input distributions to a typical sigmoid, over the course of training, shown as {15, 50, 85}th percentiles.

Remove Dropout

Increase learning rate

Accelerate the learning rate decay

Reduce the L2 weight regularization

Remove Local Response Normalization

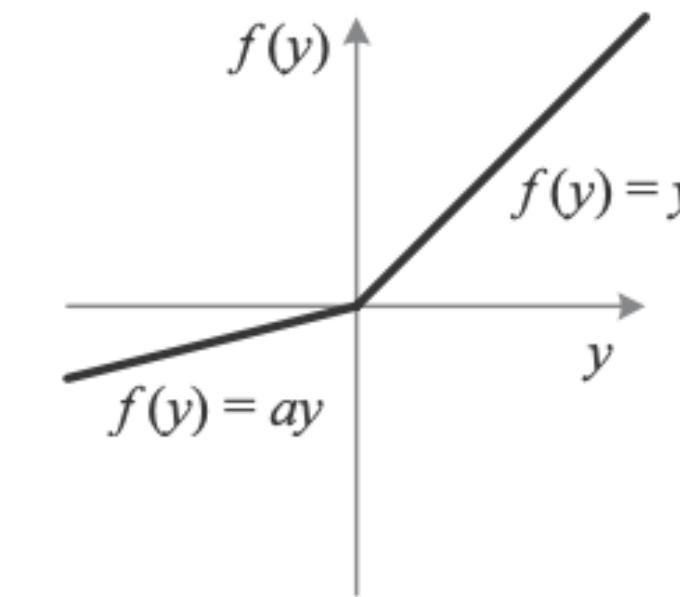
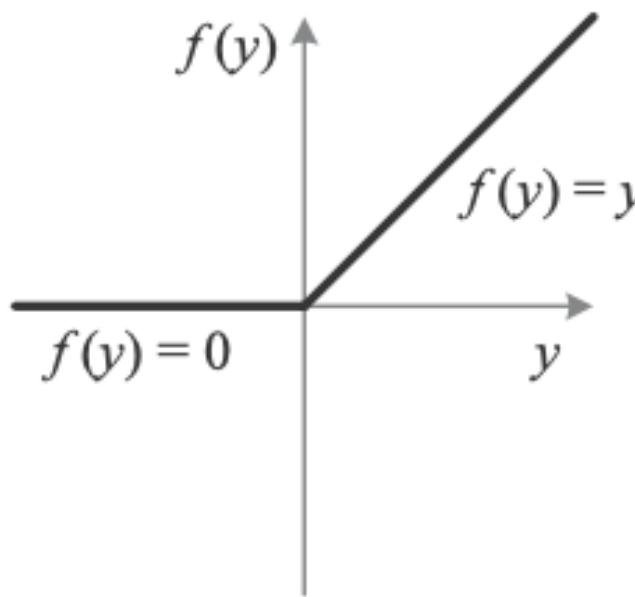
Shuffle training examples more thoroughly

Model	Resolution	Crops	Models	Top-1 error	Top-5 error
GoogLeNet ensemble	224	144	7	-	6.67%
Deep Image low-res	256	-	1	-	7.96%
Deep Image high-res	512	-	1	24.88	7.42%
Deep Image ensemble	variable	-	-	-	5.98%
BN-Inception single crop	224	1	1	25.2%	7.82%
BN-Inception multicrop	224	144	1	21.99%	5.82%
BN-Inception ensemble	224	144	6	20.1%	4.9%*

# Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification


[YouTube Playlist](#)

## Parametric Rectifiers



$f(y_i) = \max(0, y_i) + a_i \min(0, y_i) \rightarrow$  Parametric ReLU

$a_i = 0 \rightarrow$  ReLU

$a_i = 0.01 \rightarrow$  Leaky ReLU

$i \rightarrow$  indicates different channels

$$\Delta a_i \leftarrow \mu \Delta a_i + \epsilon \frac{\partial \mathcal{E}}{\partial a_i} \quad \text{momentum method}$$

	top-1	top-5
ReLU	33.82	13.34
LReLU ( $a = 0.25$ )	33.80	13.56
PReLU, channel-shared	32.71	12.87
PReLU, channel-wise	<b>32.64</b>	<b>12.75</b>

## Initialization of Filter Weights for Rectifiers

**Central idea:** investigate the variance of the responses in each layer

$$y_\ell = W_\ell x_\ell + b_\ell$$

$x_\ell \in \mathbb{R}^n \rightarrow$  co-located  $k \times k$  pixels in  $c$  input channels

$k \rightarrow$  spatial filter size of the layer  
 $n = k^2 c \rightarrow$  number of connections of a response  
 $d \rightarrow$  number of filters  
 $W_\ell \in \mathbb{R}^{d \times n} \rightarrow$  weights of  $d$  filters  
 $b_\ell \in \mathbb{R}^d \rightarrow$  bias  
 $y_\ell \rightarrow$  response at a pixel of the output map  
 $x_\ell = f(y_{\ell-1})$   
 $c_\ell = d_{\ell-1}$

$$\text{Var}[y_\ell] = n_\ell \text{Var}[w_\ell x_\ell] \underset{\mathbb{E}[w_\ell]=0}{=} n_\ell \underset{=\mathbb{E}[w_\ell^2]}{\text{Var}[w_\ell]} \underset{\neq \text{Var}[x_\ell]}{\mathbb{E}[x_\ell^2]}$$

$y_\ell \rightarrow$  random variable of each element in  $y_\ell$   
 $w_\ell \rightarrow$  random variable of each element in  $W_\ell$   
 $x_\ell \rightarrow$  random variable of each element in  $x_\ell$

$$\mathbb{E}[x_\ell^2] = \frac{1}{2} \text{Var}[y_{\ell-1}] \text{ for ReLU}$$

$$\text{Var}[y_\ell] = \frac{1}{2} n_\ell \text{Var}[y_{\ell-1}] \text{Var}[w_\ell]$$

$$\text{Var}[y_L] = \text{Var}[y_1] \left( \prod_{\ell=2}^L \frac{1}{2} n_\ell \text{Var}[w_\ell] \right)$$

$$\underbrace{\frac{1}{2} n_\ell \text{Var}[w_\ell]}_{=1} \implies w_\ell \sim \mathcal{N}(0, \sqrt{\frac{2}{n_\ell}})$$

## Backward Propagation

$$\Delta x_\ell = \hat{W}_\ell \Delta y_\ell$$

$$\Delta x_\ell = \frac{\partial \mathcal{E}}{\partial x_\ell} \in \mathbb{R}^c$$

gradient at a pixel of this layer

$$\Delta y_\ell = \frac{\partial \mathcal{E}}{\partial y_\ell} \in \mathbb{R}^{\hat{n}}$$

$\hat{k} \times k$  pixels in  $d$  channels

$$\hat{W}_\ell \in \mathbb{R}^{c \times \hat{n}} \rightarrow$$
 reshaped from  $W_\ell$

$$\Delta y_\ell = f'(y_\ell) \Delta x_{\ell+1}$$

$$\mathbb{E}[\Delta y_\ell] = \frac{1}{2} \mathbb{E}[\Delta x_{\ell+1}]$$

for ReLU  $f'(y_\ell) = 0$  or 1 with equal prob.

$$\mathbb{E}[\Delta y_\ell^2] = \text{Var}[\Delta y_\ell] = \frac{1}{2} \text{Var}[\Delta x_{\ell+1}]$$

$$\text{Var}[\Delta x_\ell] = \hat{n}_\ell \text{Var}[w_\ell] \text{Var}[\Delta y_\ell]$$

$$= \frac{1}{2} \hat{n}_\ell \text{Var}[w_\ell] \text{Var}[\Delta x_{\ell+1}]$$

$$\text{Var}[\Delta x_2] = \text{Var}[\Delta x_{L+1}] \left( \prod_{\ell=2}^L \frac{1}{2} \hat{n}_\ell \text{Var}[w_\ell] \right)$$

if  $\ell = 1, \forall \ell$

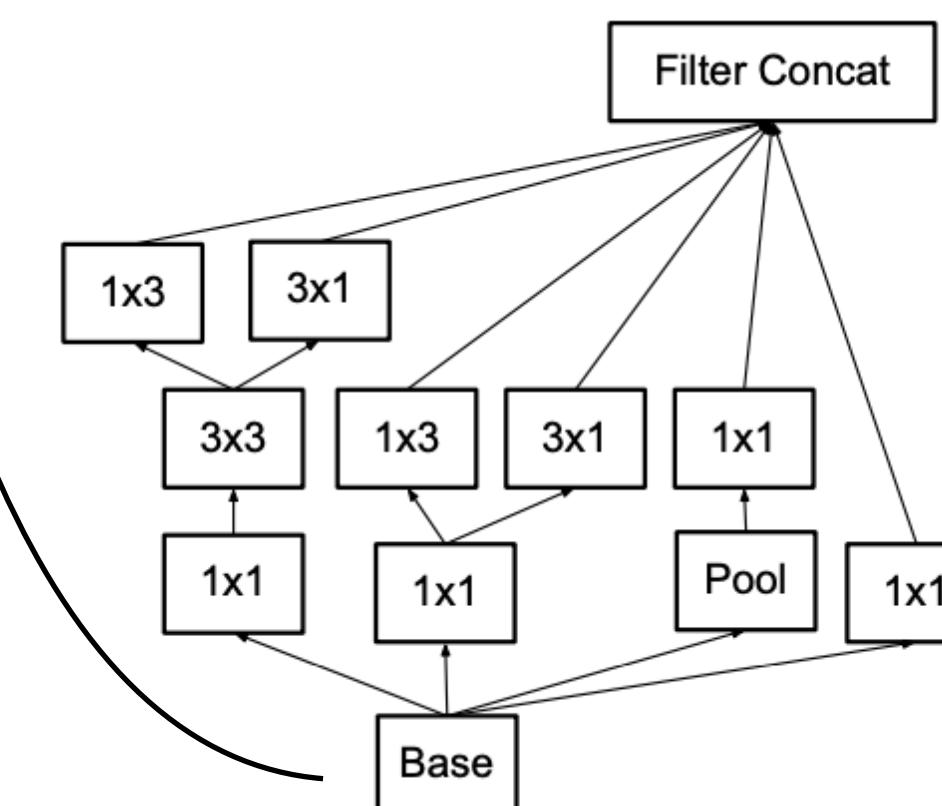


Boulder

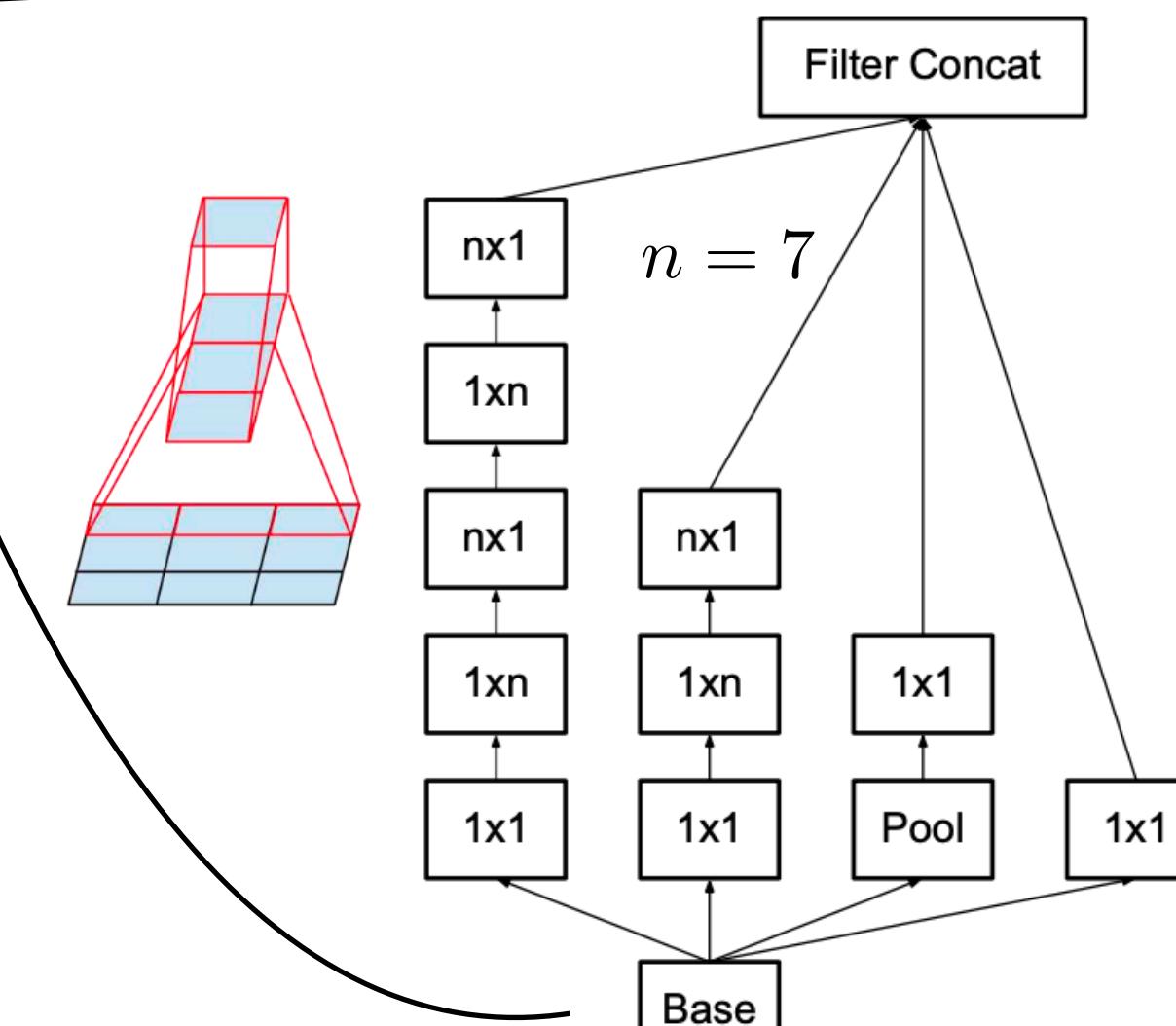
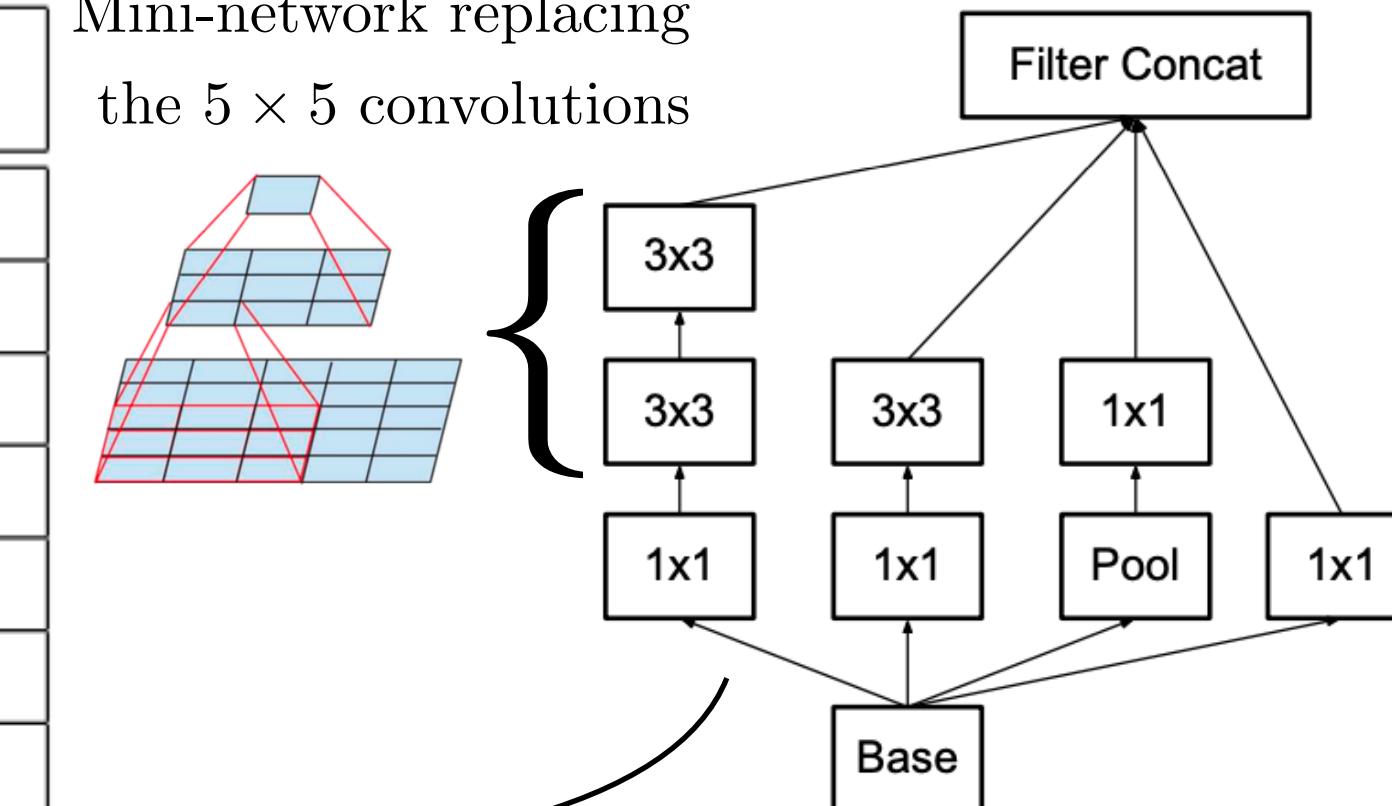
# Rethinking the Inception Architecture for Computer Vision

[YouTube Playlist](#)

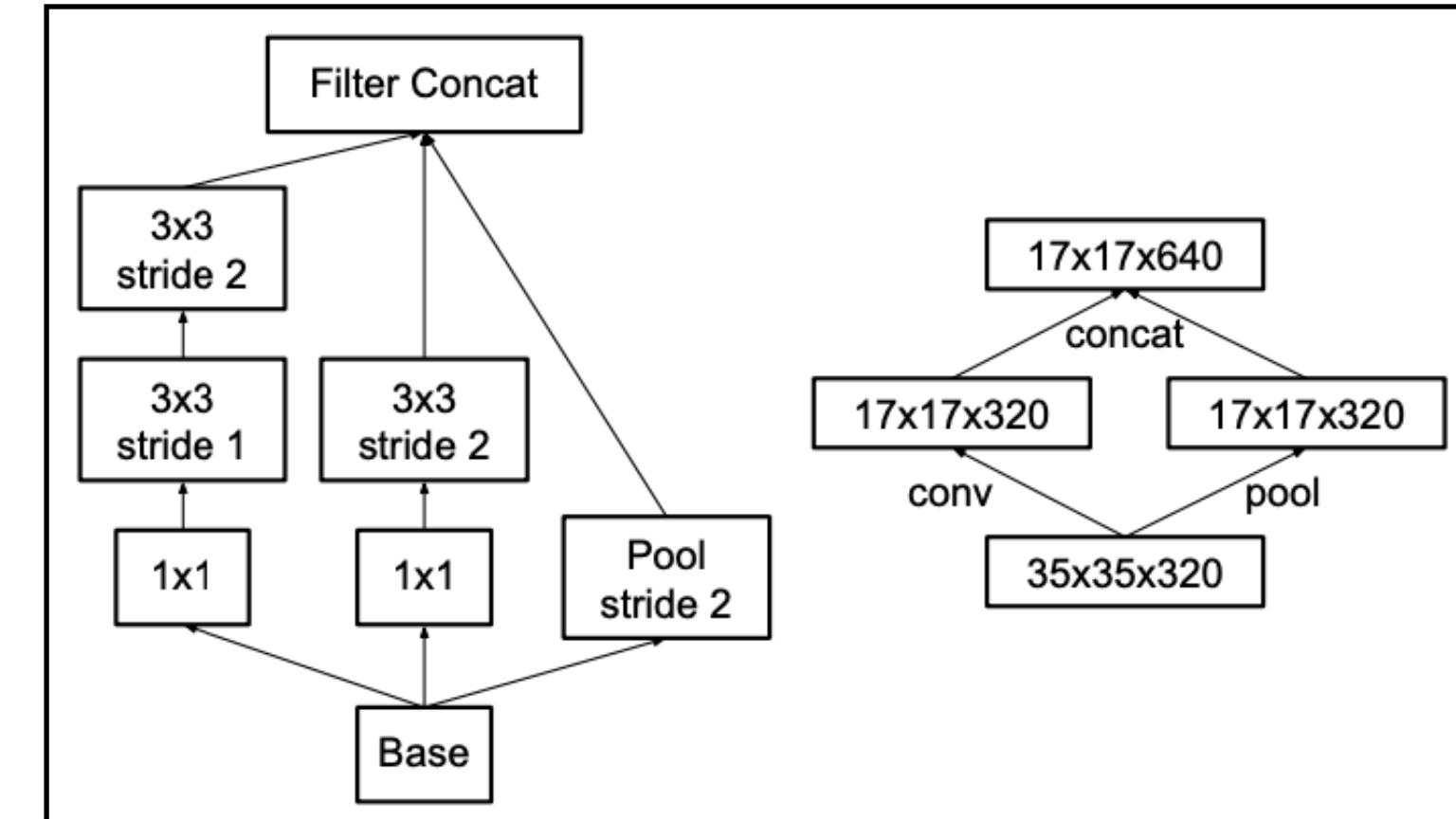
type	patch size/stride or remarks	input size
conv	$3 \times 3/2$	$299 \times 299 \times 3$
conv	$3 \times 3/1$	$149 \times 149 \times 32$
conv padded	$3 \times 3/1$	$147 \times 147 \times 32$
pool	$3 \times 3/2$	$147 \times 147 \times 64$
conv	$3 \times 3/1$	$73 \times 73 \times 64$
conv	$3 \times 3/2$	$71 \times 71 \times 80$
conv	$3 \times 3/1$	$35 \times 35 \times 192$
$3 \times$ Inception	As in figure 5	$35 \times 35 \times 288$
$5 \times$ Inception	As in figure 6	$17 \times 17 \times 768$
$2 \times$ Inception	As in figure 7	$8 \times 8 \times 1280$
pool	$8 \times 8$	$8 \times 8 \times 2048$
linear	logits	$1 \times 1 \times 2048$
softmax	classifier	$1 \times 1 \times 1000$



Mini-network replacing  
the  $5 \times 5$  convolutions



$$\text{cross-entropy loss} \leftarrow L = - \sum_{k=1}^K q(k|x) \log(p(k|x)) = -\log(p(k^*|x)) \rightarrow \text{negative log-likelihood}$$



Inception module that  
reduces the grid-size while  
expanding the filter banks

Replace  $q(k|x)$  with  $q'(k|x) = (1 - \epsilon)q(k|x) + \epsilon u(k)$

label-smoothing regularization

$$H(q', p) = (1 - \epsilon)H(q, p) + \epsilon H(u, p)$$

$$H(u, p) = KL(u||p) + H(u)$$

$$u = 1/K$$

$$p(k|x) = \exp(z_k) / \sum_{i=1}^K \exp(z_i)$$

$z_i \rightarrow$  logits (un-normalized log-probabilities)

$q(k|x) \rightarrow$  ground truth distribution over labels

$k^* \rightarrow$  ground truth label

$$q(k|x) = \delta_{k^*}(k) = \begin{cases} 0 & \text{if } k \neq k^*; \\ 1 & \text{if } k = k^*. \end{cases}$$



Boulder

# Training Very Deep Networks

## Highway Networks

$$y = H(x; W_H) \rightarrow \text{a single layer}$$

$H \rightarrow$  non-linear transformation parametrized by  $W_H$

(e.g., affine transformation followed by a non-linear activation function)

$$y = H(x; W_H) \cdot T(x; W_T) + x \cdot C(x; W_C)$$

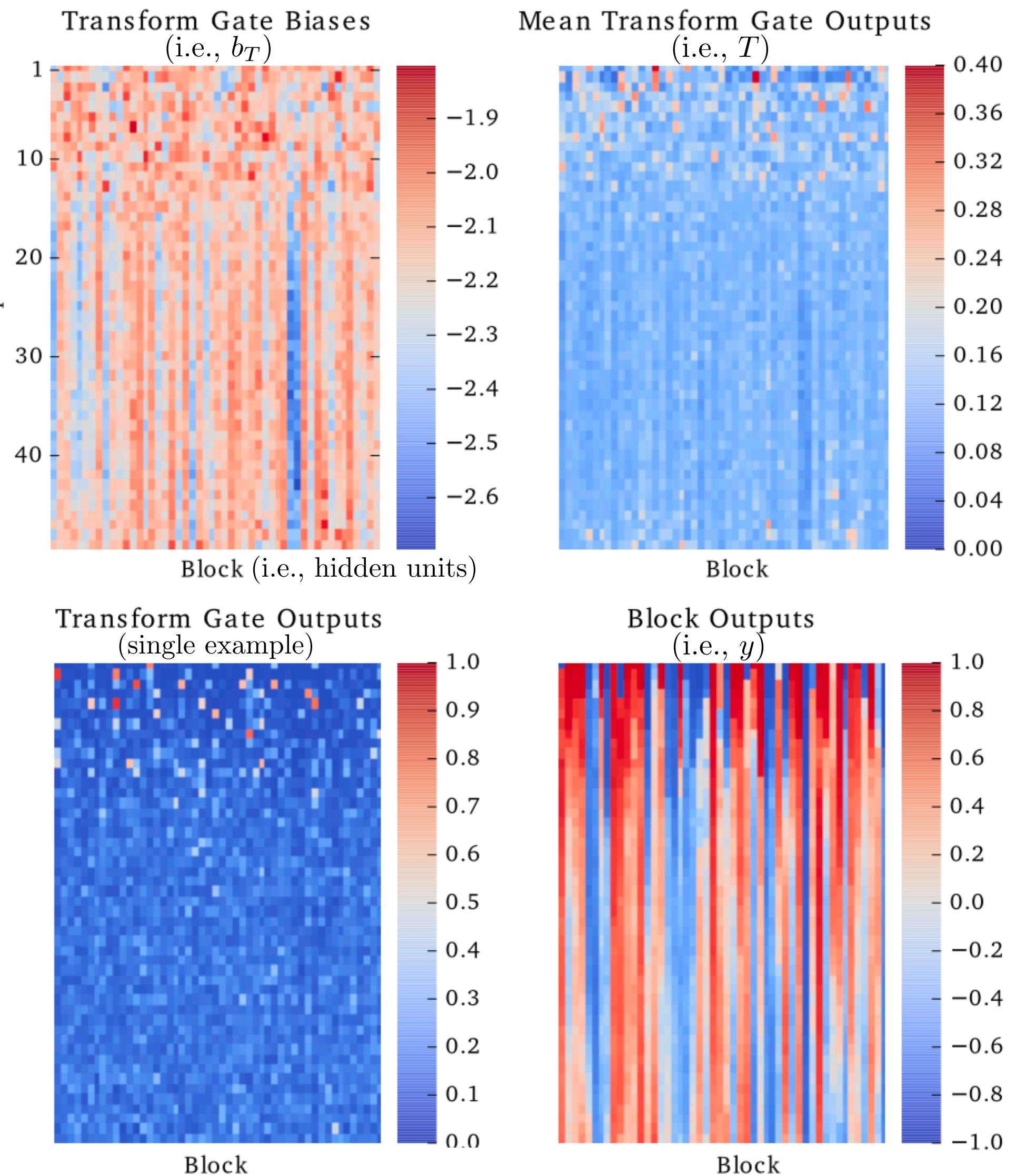
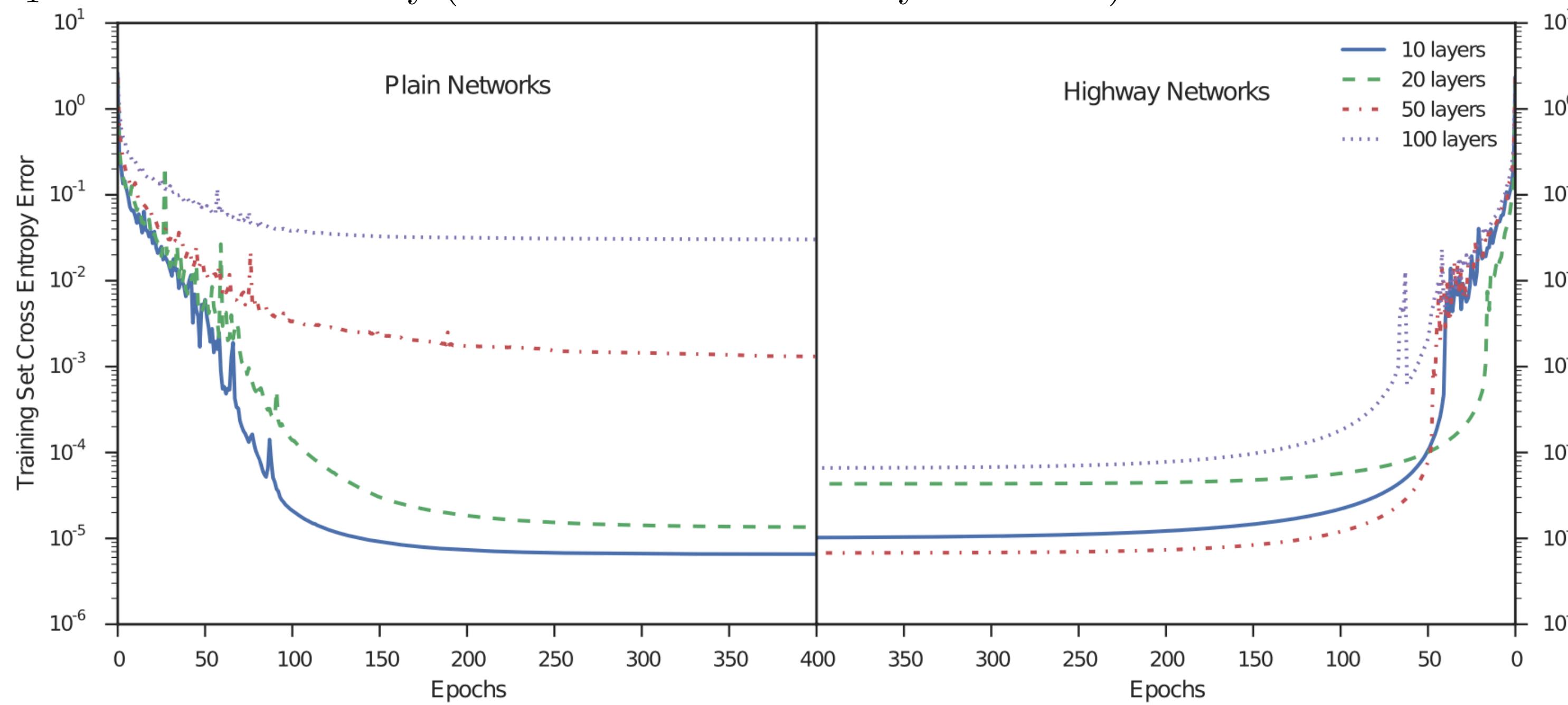
$T \rightarrow$  transform gate       $\underbrace{\cdot}_{\text{element-wise multiplication}}$

$C \rightarrow$  carry gate

$$C = 1 - T$$

$$T(x) = \sigma(W_T x + b_T)$$

$b_T = -1$  or  $-3$  initially (biased towards the “carry” behavior)



Srivastava, Rupesh Kumar, Klaus Greff, and Jürgen Schmidhuber. "Training very deep networks." *arXiv preprint arXiv:1507.06228* (2015).

Srivastava, Rupesh Kumar, Klaus Greff, and Jürgen Schmidhuber. "Highway networks." *arXiv preprint arXiv:1505.00387* (2015).

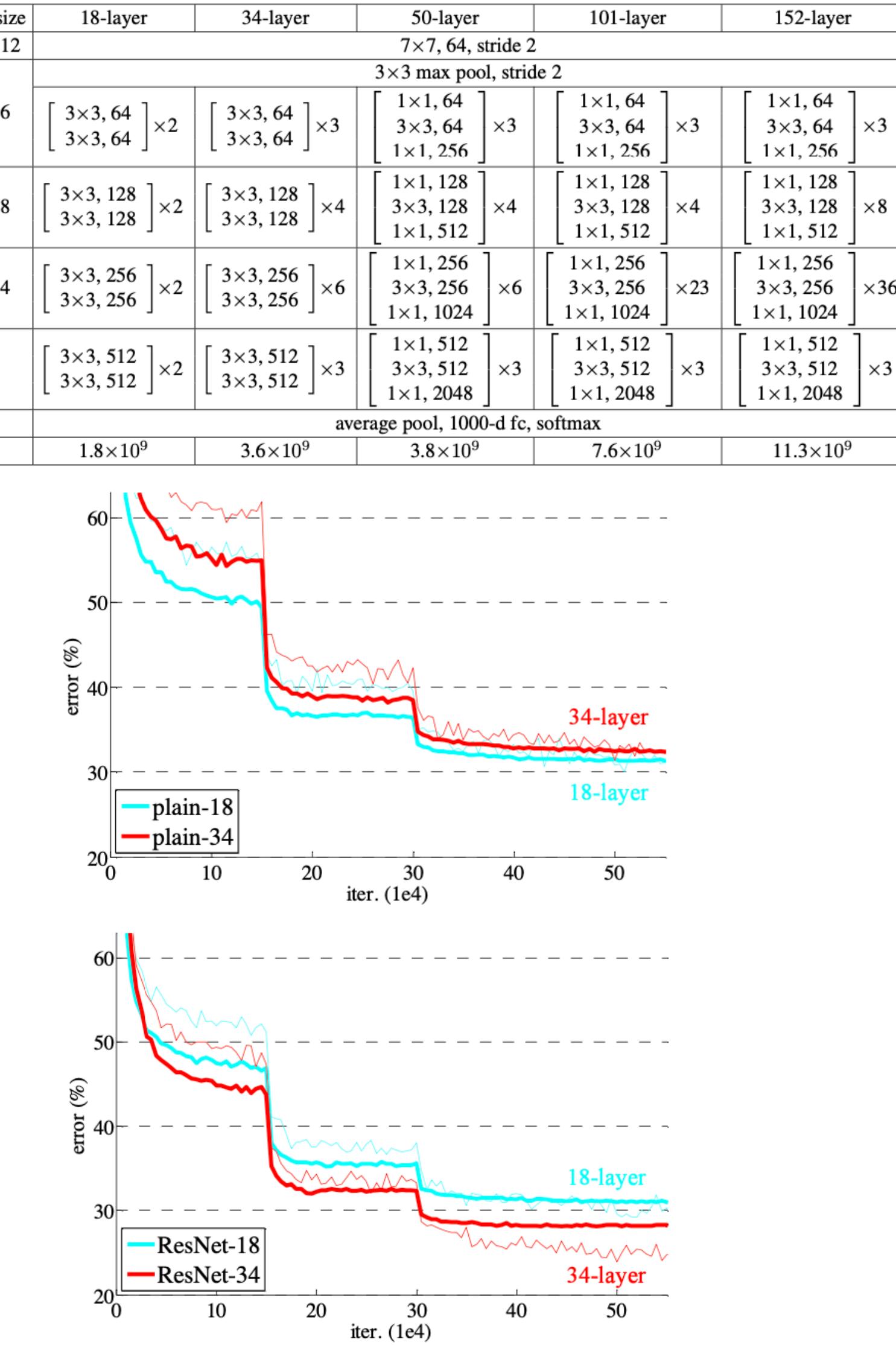
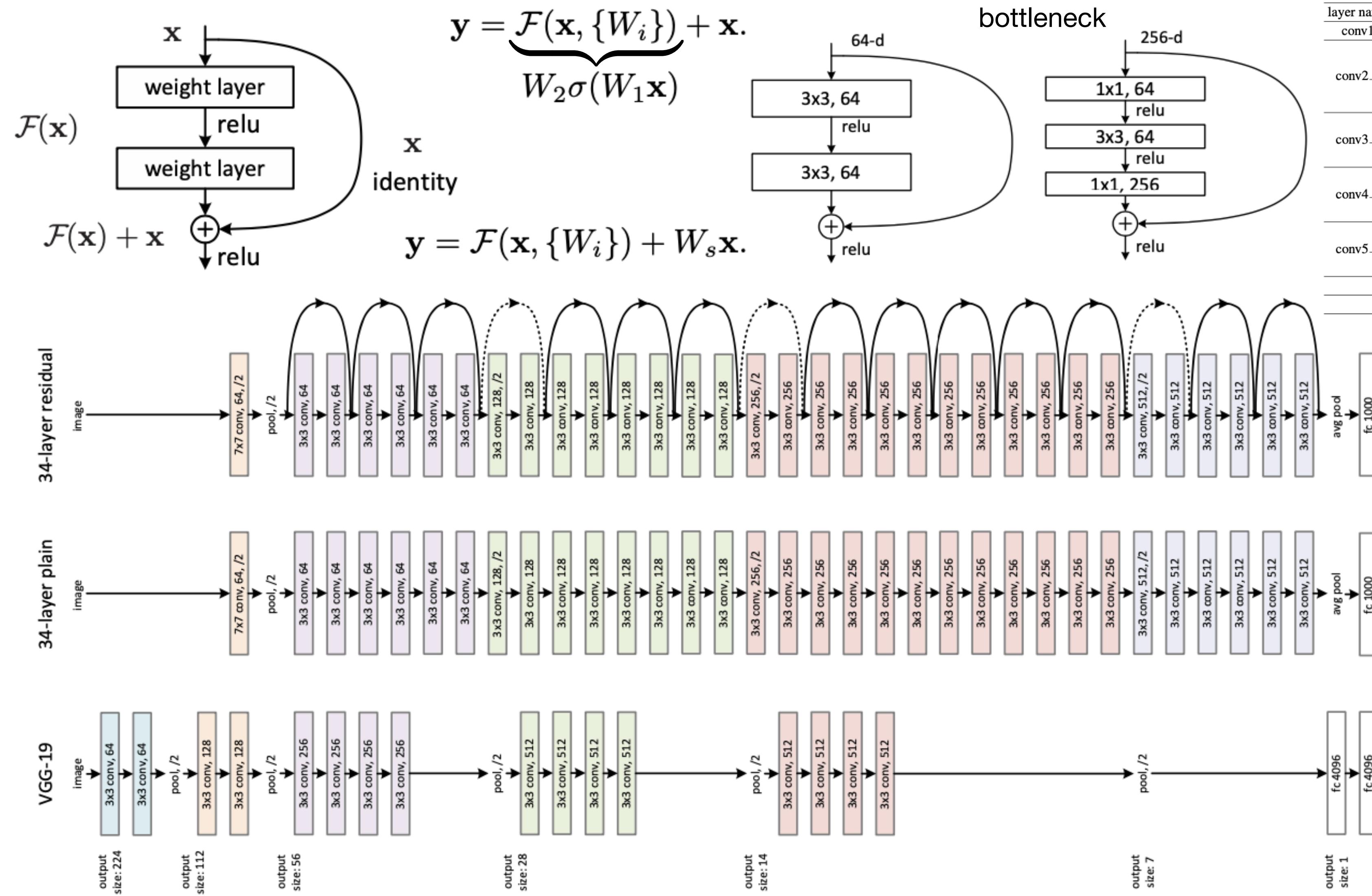


# Boulder

# Deep Residual Learning for Image Recognition



# YouTube Video



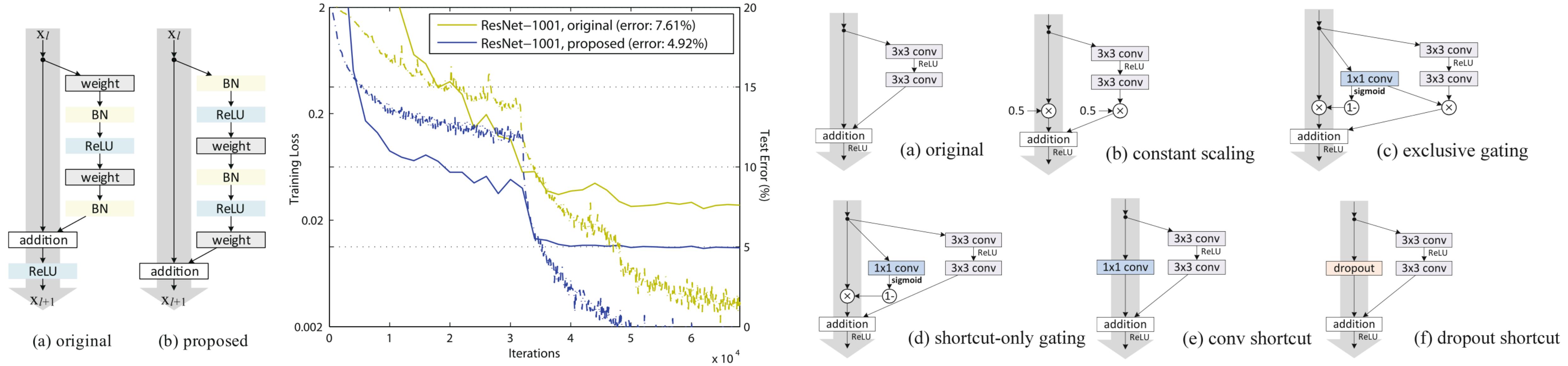


Boulder



[YouTube Video](#)

# Identity Mappings in Deep Residual Networks



$$\mathbf{y}_l = h(\mathbf{x}_l) + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l), \\ \mathbf{x}_{l+1} = f(\mathbf{y}_l).$$

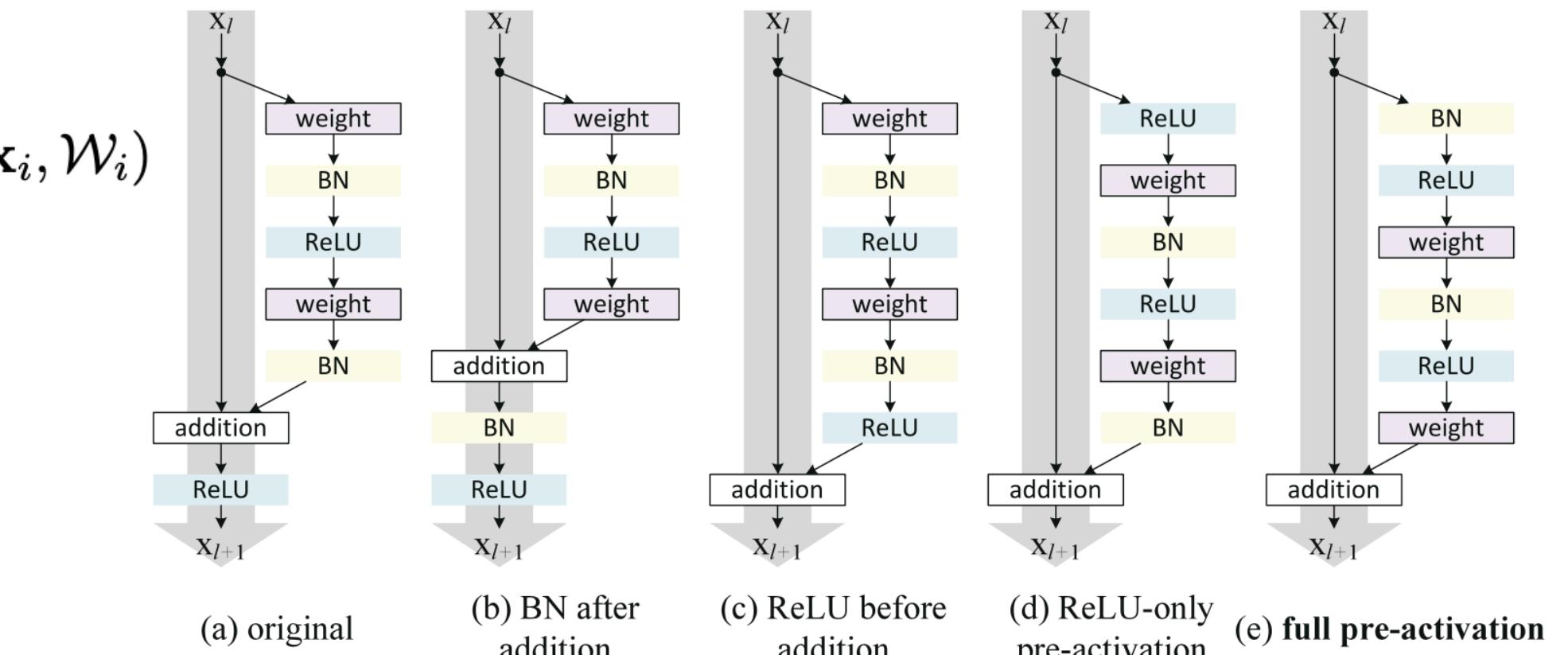
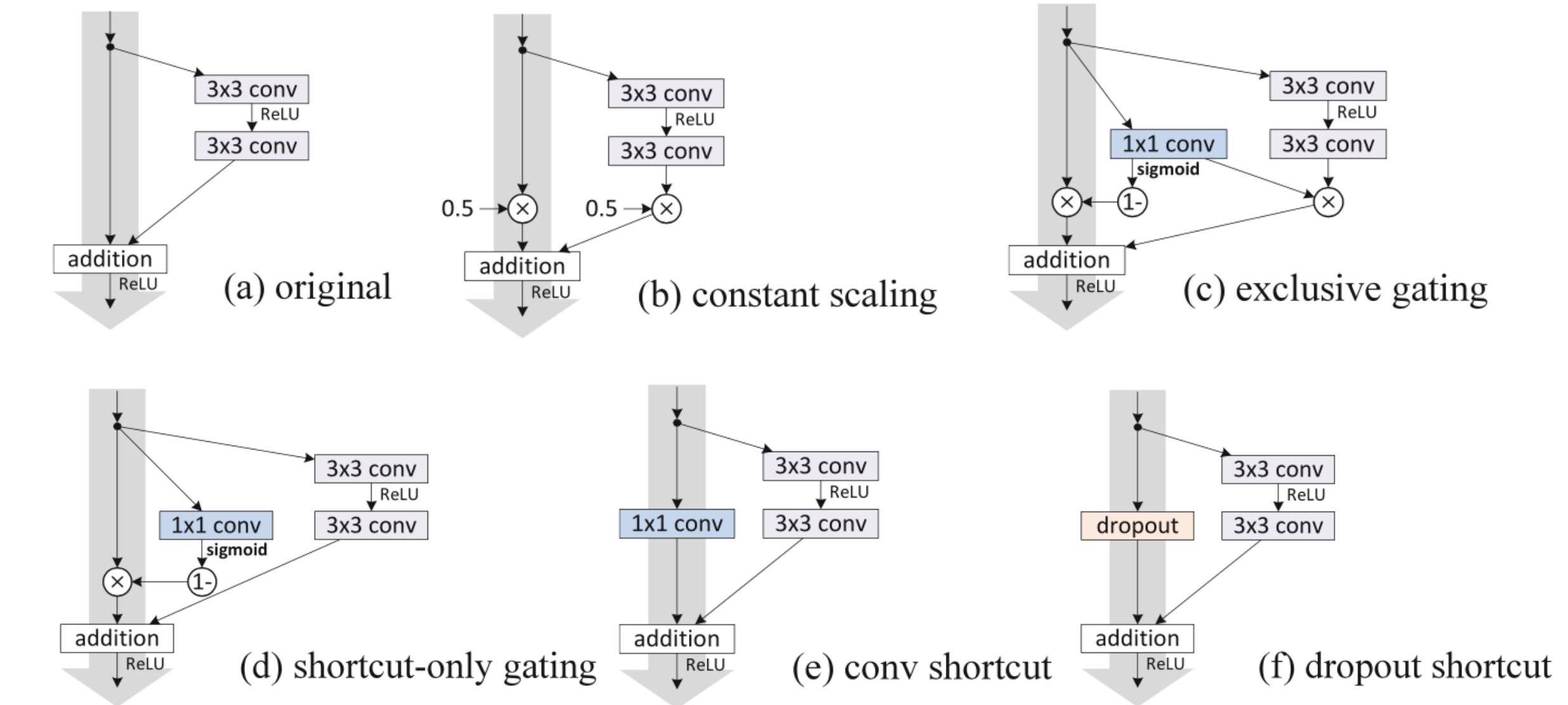
$$h(\mathbf{x}_l) = \mathbf{x}_l \\ f \text{ is ReLU} \quad \left. \right\} \text{Original}$$

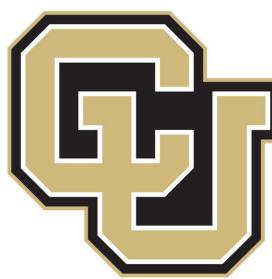
$$\mathbf{x}_{l+1} = \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l) \rightarrow \text{if } f \text{ and } h \text{ are both identity}$$

$$\mathbf{x}_L = \mathbf{x}_l + \sum_{i=l}^{L-1} \mathcal{F}(\mathbf{x}_i, \mathcal{W}_i),$$

$$\frac{\partial \mathcal{E}}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L} \frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L} \left( 1 + \frac{\partial}{\partial \mathbf{x}_l} \sum_{i=l}^{L-1} \mathcal{F}(\mathbf{x}_i, \mathcal{W}_i) \right)$$

$$h(\mathbf{x}_l) = \lambda_l \mathbf{x}_l \\ \mathbf{x}_{l+1} = \lambda_l \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l) \\ \mathbf{x}_L = (\prod_{i=l}^{L-1} \lambda_i) \mathbf{x}_l + \sum_{i=l}^{L-1} (\prod_{j=i+1}^{L-1} \lambda_j) \mathcal{F}(\mathbf{x}_i, \mathcal{W}_i) \\ \mathbf{x}_L = (\prod_{i=l}^{L-1} \lambda_i) \mathbf{x}_l + \sum_{i=l}^{L-1} \hat{\mathcal{F}}(\mathbf{x}_i, \mathcal{W}_i) \\ \frac{\partial \mathcal{E}}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L} \left( (\underbrace{\prod_{i=l}^{L-1} \lambda_i}_{\prod_{i=l}^{L-1} h'_i}) + \frac{\partial}{\partial \mathbf{x}_l} \sum_{i=l}^{L-1} \hat{\mathcal{F}}(\mathbf{x}_i, \mathcal{W}_i) \right)$$





Boulder

# Deep Networks with Stochastic Depth

For each mini-batch, randomly select sets of layers and remove their corresponding transformation functions, only keeping the identity skip connection.

**ResNet Architecture**

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

$f_\ell \rightarrow \text{Conv-BN-ReLU-Conv-BN}$

**Stochastic depth**

$b_\ell \rightarrow \text{Bernoulli random variable}$

$p_\ell = \Pr(b_\ell = 1) \rightarrow \text{"survival" probability}$

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

If  $b_\ell = 0$ ,  $H_\ell = \text{id}(H_{\ell-1})$

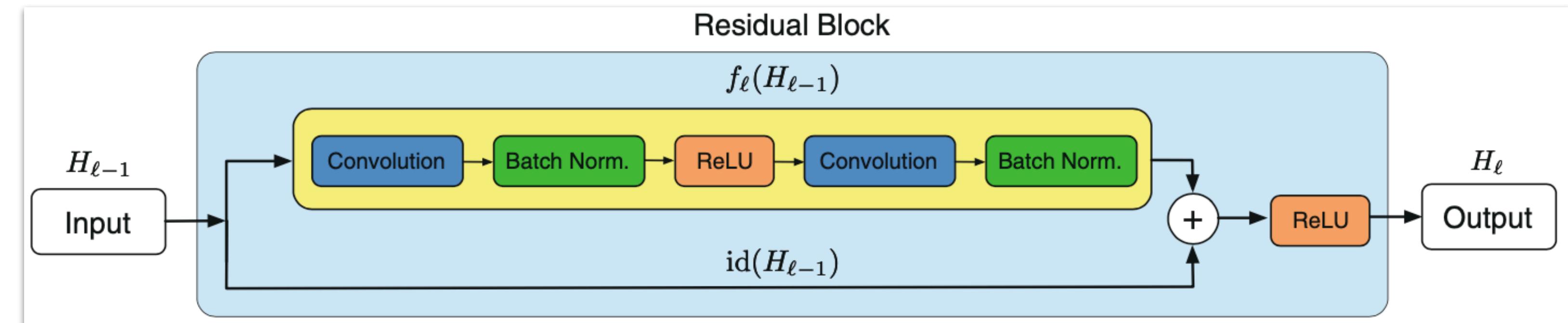
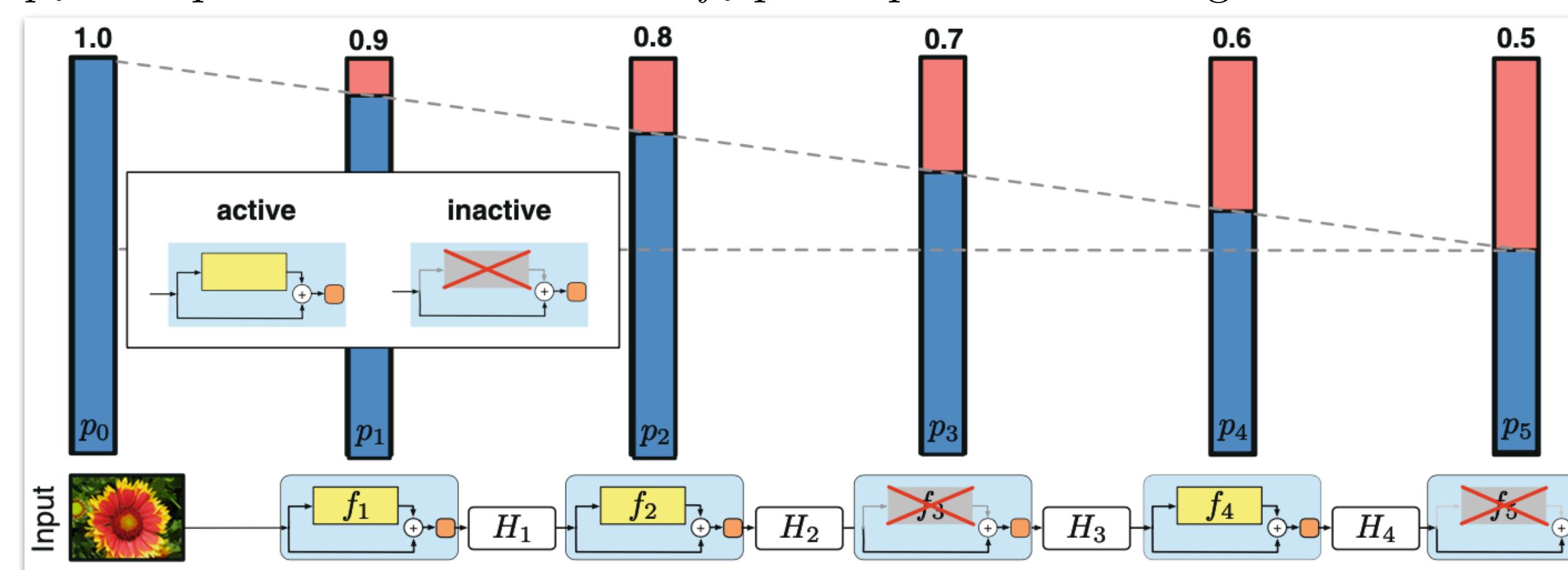
$H_{\ell-1}$  is always non-negative

$$p_\ell = 1 - \frac{\ell}{L}(1 - p_L)$$

**Stochastic depth during testing**

$$H_\ell^{\text{Test}} = \text{ReLU}(p_\ell f_\ell(H_{\ell-1}^{\text{Test}}; W_\ell) + H_{\ell-1}^{\text{Test}})$$

$p_\ell \rightarrow \text{expected number of times } f_\ell \text{ participates in training}$



Training time comparison on benchmark datasets.

	CIFAR10+	CIFAR100+	SVHN	ImageNet
Constant depth	20 h 42 m	20 h 51 m	33 h 43 m	-
Stochastic depth	15 h 7 m	15 h 20 m	25 h 33 m	-
Maxout [21]	9.38	-	2.47	-
DropConnect [20]	9.32	-	1.94	-
Net in Net [24]	8.81	-	2.35	-
Deeply Supervised [13]	7.97	-	1.92	33.70
Frac. Pool [25]	-	27.62	-	-
All-CNN [6]	7.25	-	-	41.20
Learning Activation [26]	7.51	30.83	-	-
R-CNN [27]	7.09	-	1.77	-
Scalable BO [28]	6.37	27.40	1.77	-
Highway Network [29]	7.60	32.24	-	-
Gen. Pool [30]	6.05	-	1.69	28.02
ResNet with constant depth	6.41	27.76	1.80	21.78
ResNet with stochastic depth	5.25	24.98	1.75	21.98



Boulder

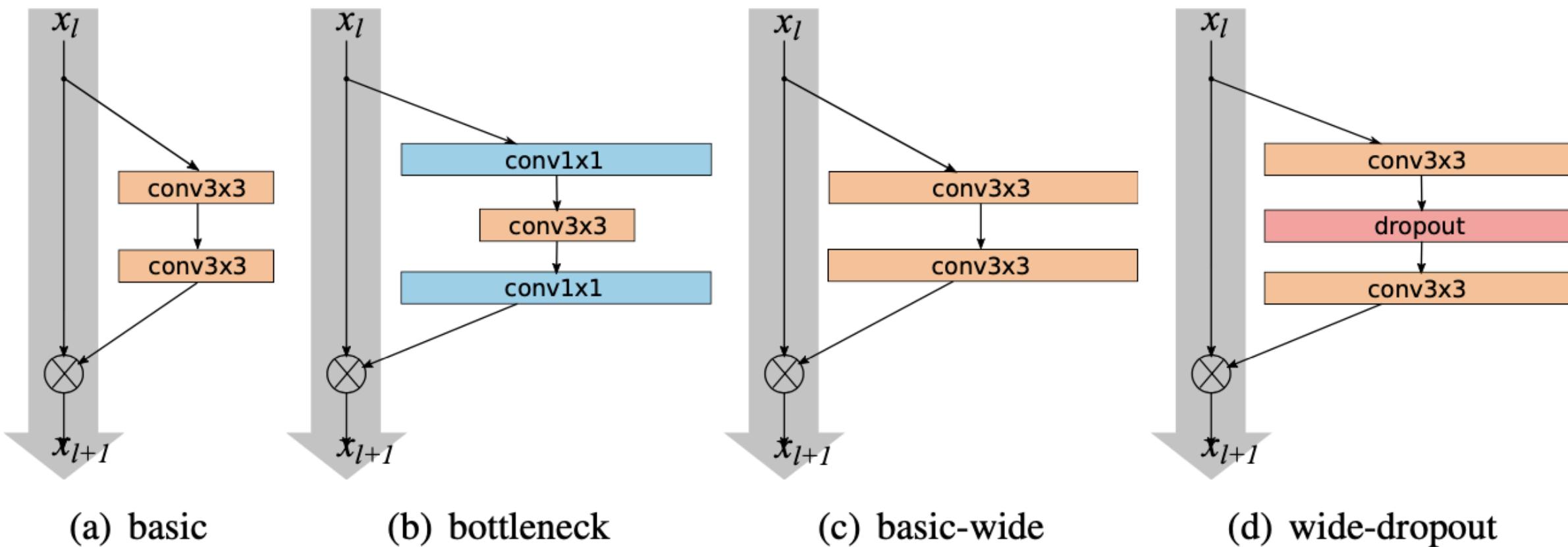


[YouTube Video](#)

# Wide Residual Networks

$$\mathbf{x}_{l+1} = \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l)$$

Conv-BN-ReLU → BN-ReLU-Conv  
ResNet                      Identity Mapping



$n \rightarrow$  total number of convolutional layers

$k \rightarrow$  widening factor (multiplies the number of features in conv layers)

WRN- $n-k$

1.  $B(3, 3)$  - original «basic» block
2.  $B(3, 1, 3)$  - with one extra  $1 \times 1$  layer
3.  $B(1, 3, 1)$  - with the same dimensionality of all convolutions, «straightened» bottleneck
4.  $B(1, 3)$  - the network has alternating  $1 \times 1$  -  $3 \times 3$  convolutions everywhere
5.  $B(3, 1)$  - similar idea to the previous block
6.  $B(3, 1, 1)$  - Network-in-Network style block

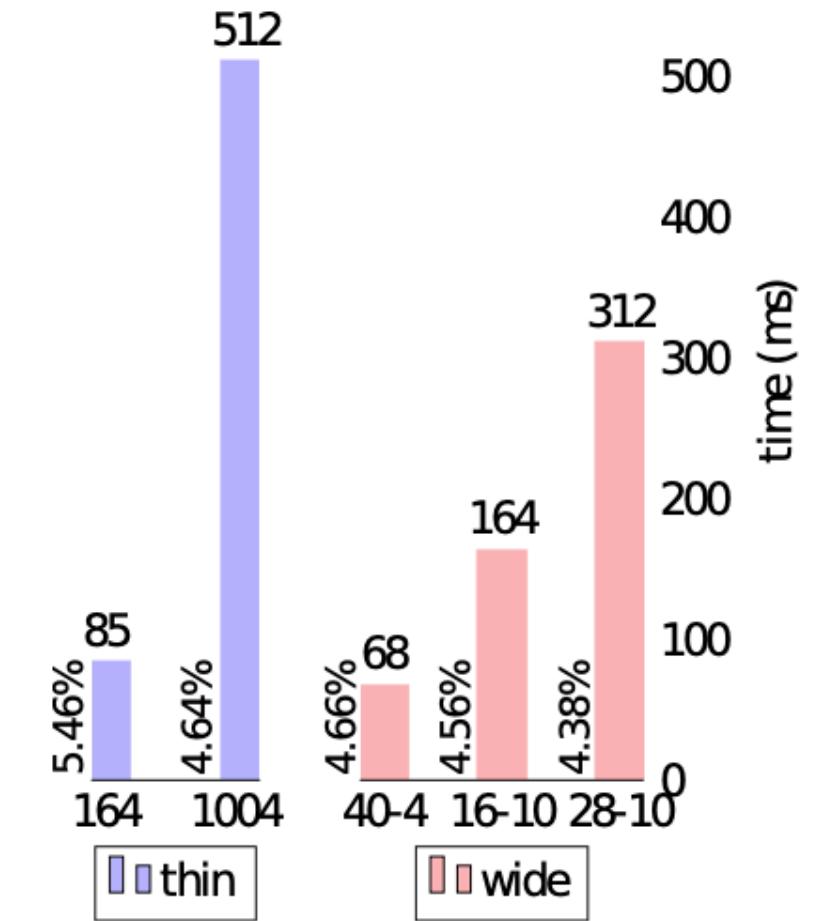
$\ell \rightarrow$  deepening factor (number of convs in a block)

$l$	CIFAR-10
1	6.69
2	5.43 ←
3	5.65
4	5.93

group name	output size	block type = $B(3, 3)$	depth	$k$	# params	CIFAR-10	CIFAR-100
conv1	$32 \times 32$	$[3 \times 3, 16]$	40	1	0.6M	6.85	30.89
conv2	$32 \times 32$	$[3 \times 3, 16 \times k]$ $[3 \times 3, 16 \times k] \times N$	40	2	2.2M	5.33	26.04
conv3	$16 \times 16$	$[3 \times 3, 32 \times k]$ $[3 \times 3, 32 \times k] \times N$	40	4	8.9M	4.97	22.89
conv4	$8 \times 8$	$[3 \times 3, 64 \times k]$ $[3 \times 3, 64 \times k] \times N$	40	8	35.7M	4.66	-
avg-pool	$1 \times 1$	$[8 \times 8]$	28	10	36.5M	<b>4.17</b>	20.50
			28	12	52.5M	4.33	<b>20.43</b>
			22	8	17.2M	4.38	21.22
			22	10	26.8M	4.44	20.75
			16	8	11.0M	4.81	22.07
			16	10	17.1M	4.56	21.59

block type	depth	# params	time,s	CIFAR-10	depth	$k$	dropout	CIFAR-10	CIFAR-100	SVHN
$B(1, 3, 1)$	40	1.4M	85.8	6.06	16	4		5.02	24.03	1.85
$B(3, 1)$	40	1.2M	67.5	5.78	16	4	✓	5.24	23.91	1.64
$B(1, 3)$	40	1.3M	72.2	6.42	28	10		4.00	19.25	-
$B(3, 1, 1)$	40	1.3M	82.2	5.86	28	10	✓	<b>3.89</b>	<b>18.85</b>	-
$B(3, 3)$	28	1.5M	67.5	5.73 ←	52	1		6.43	29.89	2.08
$B(3, 1, 3)$	22	1.1M	59.9	5.78	52	1	✓	6.28	29.78	1.70

	depth- $k$	# params	CIFAR-10	CIFAR-100
NIN [20]			8.81	35.67
DSN [19]			8.22	34.57
FitNet [24]			8.39	35.04
Highway [28]			7.72	32.39
ELU [5]			6.55	24.28
original-ResNet[11]	110	1.7M	6.43	25.16
	1202	10.2M	7.93	27.82
stoc-depth[14]	110	1.7M	5.23	24.58
	1202	10.2M	4.91	-
	110	1.7M	6.37	-
pre-act-ResNet[13]	164	1.7M	5.46	24.33
	1001	10.2M	4.92(4.64)	22.71
WRN	40-4	8.9M	4.53	21.18
	16-8	11.0M	4.27	20.43
	28-10	36.5M	<b>4.00</b>	<b>19.25</b>



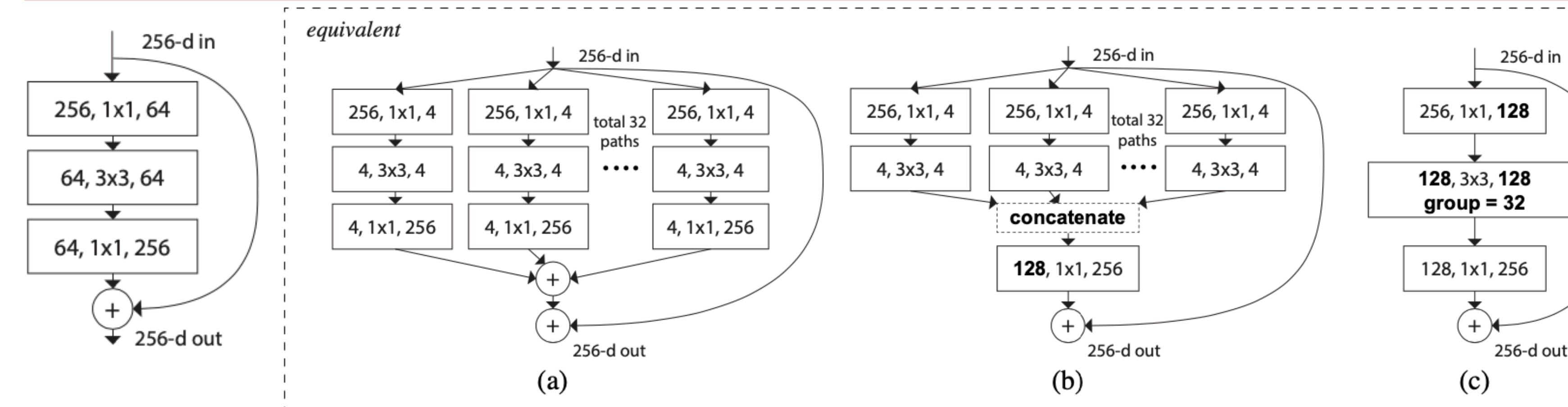


Boulder

# Aggregated Residual Transformations for Deep Neural Networks



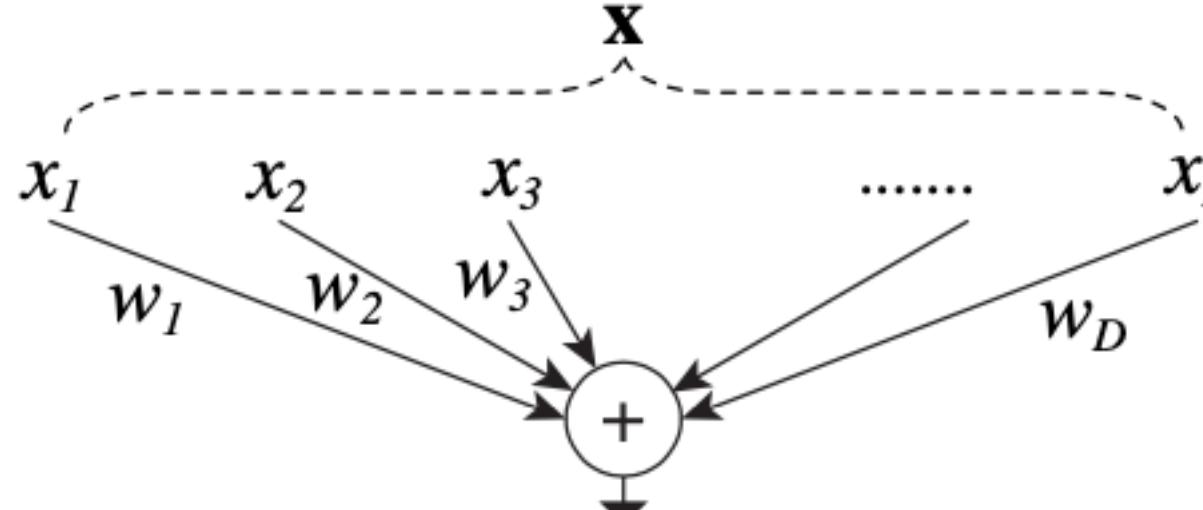
[YouTube Video](#)



$$\sum_{i=1}^{32} \underbrace{W_i}_{\in \mathbb{R}^{256 \times 4}} \underbrace{x_i}_{\in \mathbb{R}^4} = \underbrace{[W_1, \dots, W_{32}]}_{\in \mathbb{R}^{256 \times 128}} \underbrace{[x_1; \dots; x_{32}]}_{\in \mathbb{R}^{128}}$$

split — transform — merge strategy → inception  
 $1 \times 1$  conv     $3 \times 3, 5 \times 5$ , etc. conv    concatenation

cardinality: size of the set of transformations



A simple neuron that performs inner product.

$$\mathcal{F}(x) = \sum_{i=1}^C \mathcal{T}_i(x) \rightarrow \text{"Network in Neuron"}$$

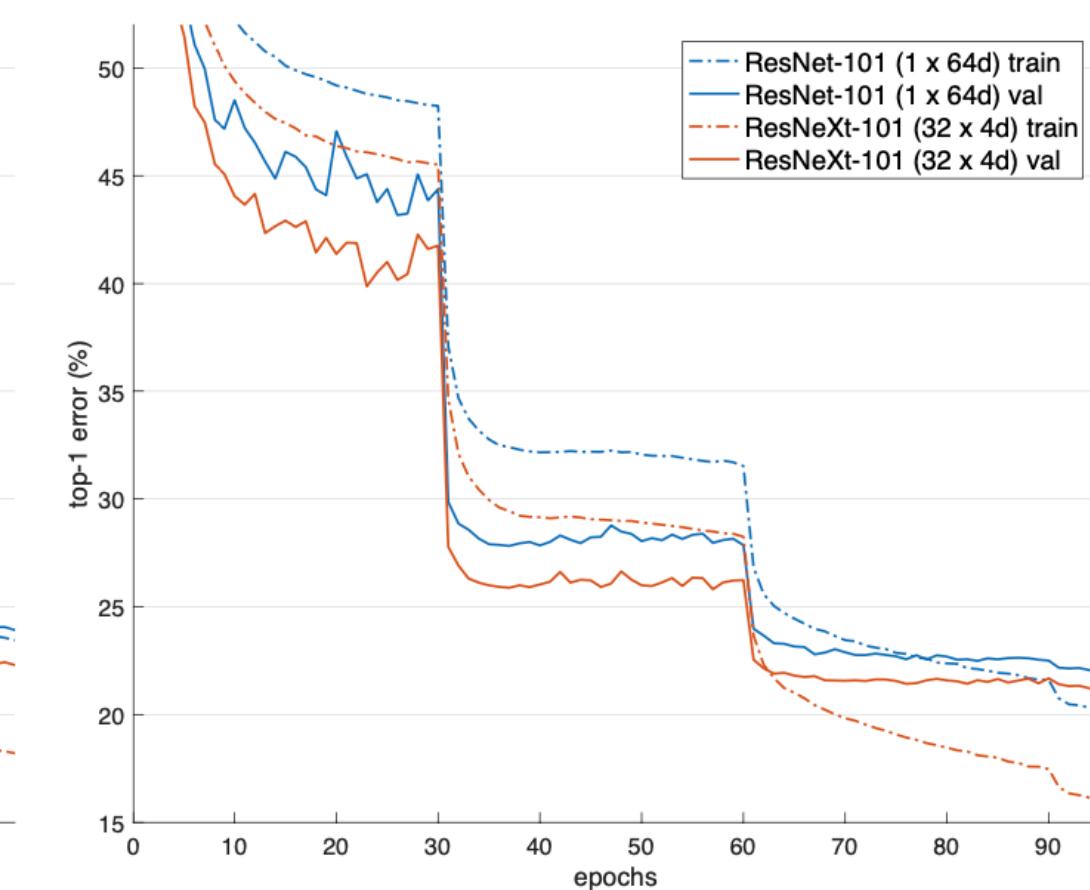
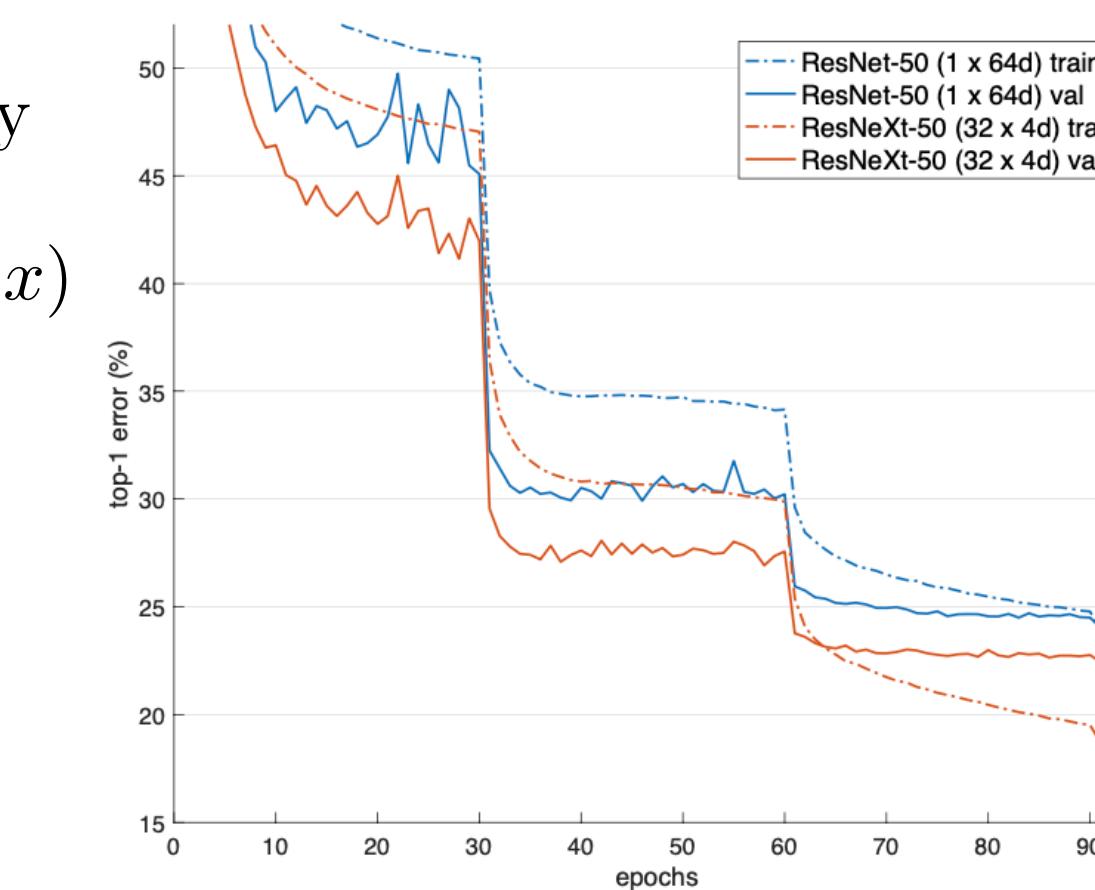
$C \rightarrow$  cardinality

$$y = x + \sum_{i=1}^C \mathcal{T}_i(x)$$

split:  $x \rightarrow x_i$

transform:  $x_i w_i$

$$\text{aggregate: } \sum_{i=1}^D w_i x_i$$



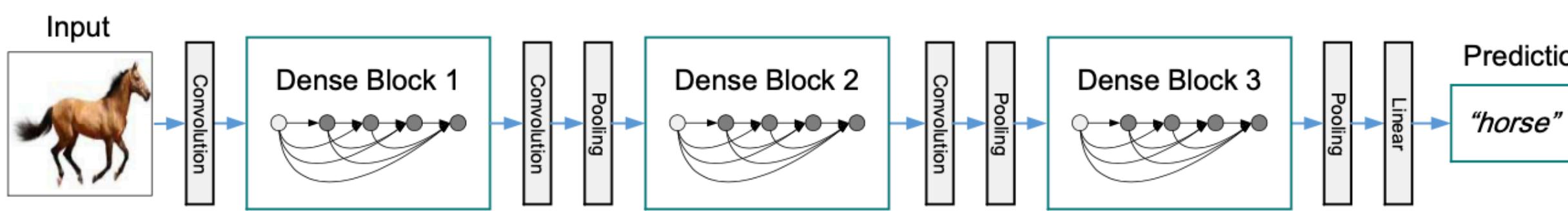
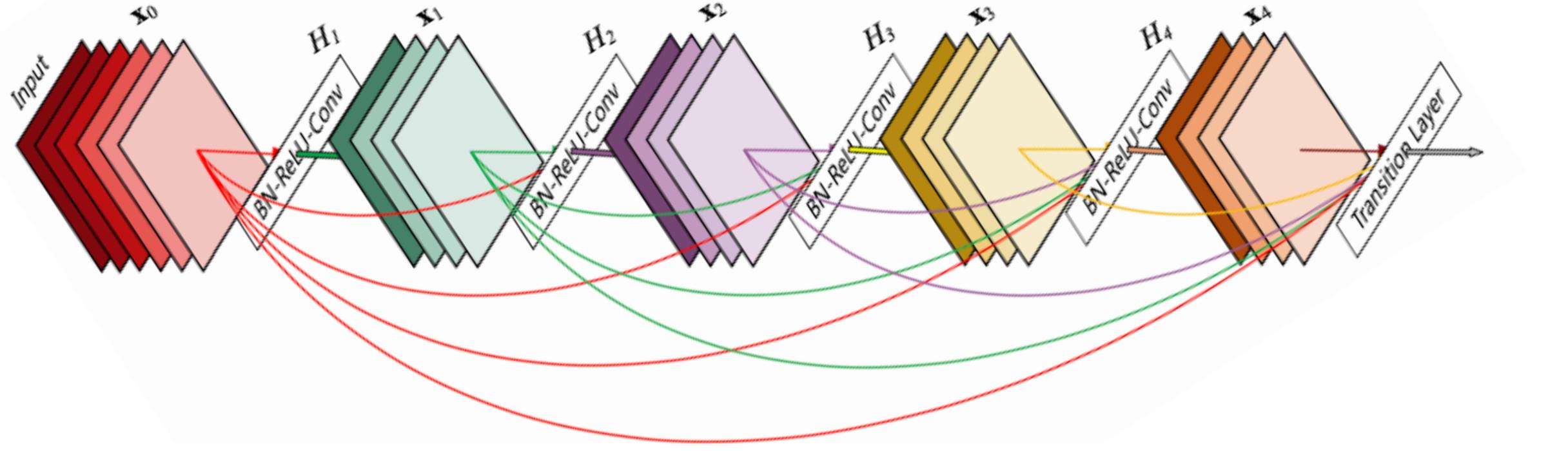


Boulder



[YouTube Video](#)

# Densely Connected Convolutional Networks



$x_0 \rightarrow$  single image

$L \rightarrow$  number of layers

$H_\ell(\cdot) \rightarrow$  nonlinear transformation

$\ell = 1, \dots, L \rightarrow$  layer index

$x_\ell \rightarrow$  output of the  $\ell$ -th layer

$x_\ell = H_\ell(x_{\ell-1})$

$x_\ell = H_\ell(x_{\ell-1}) + x_{\ell-1}$

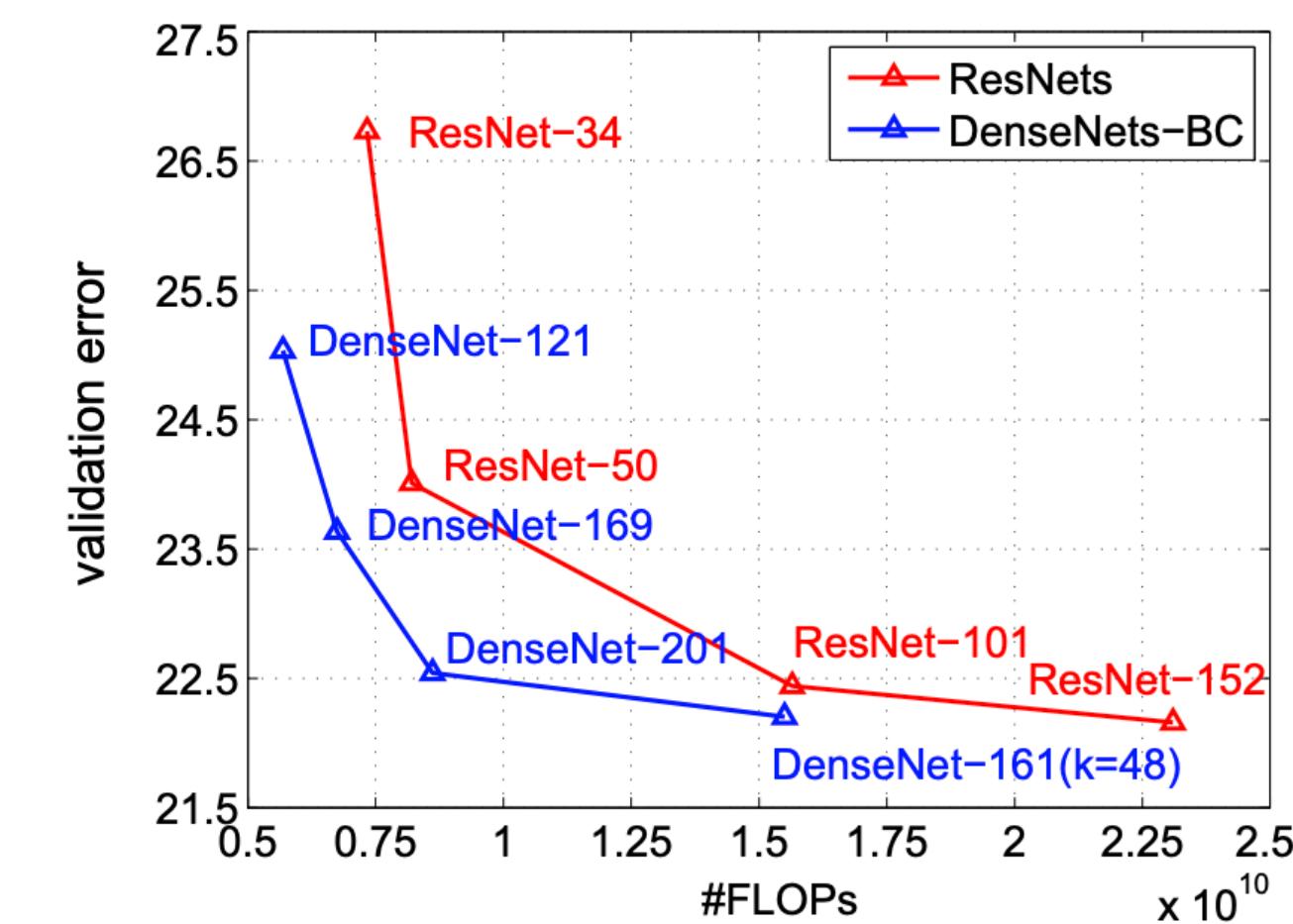
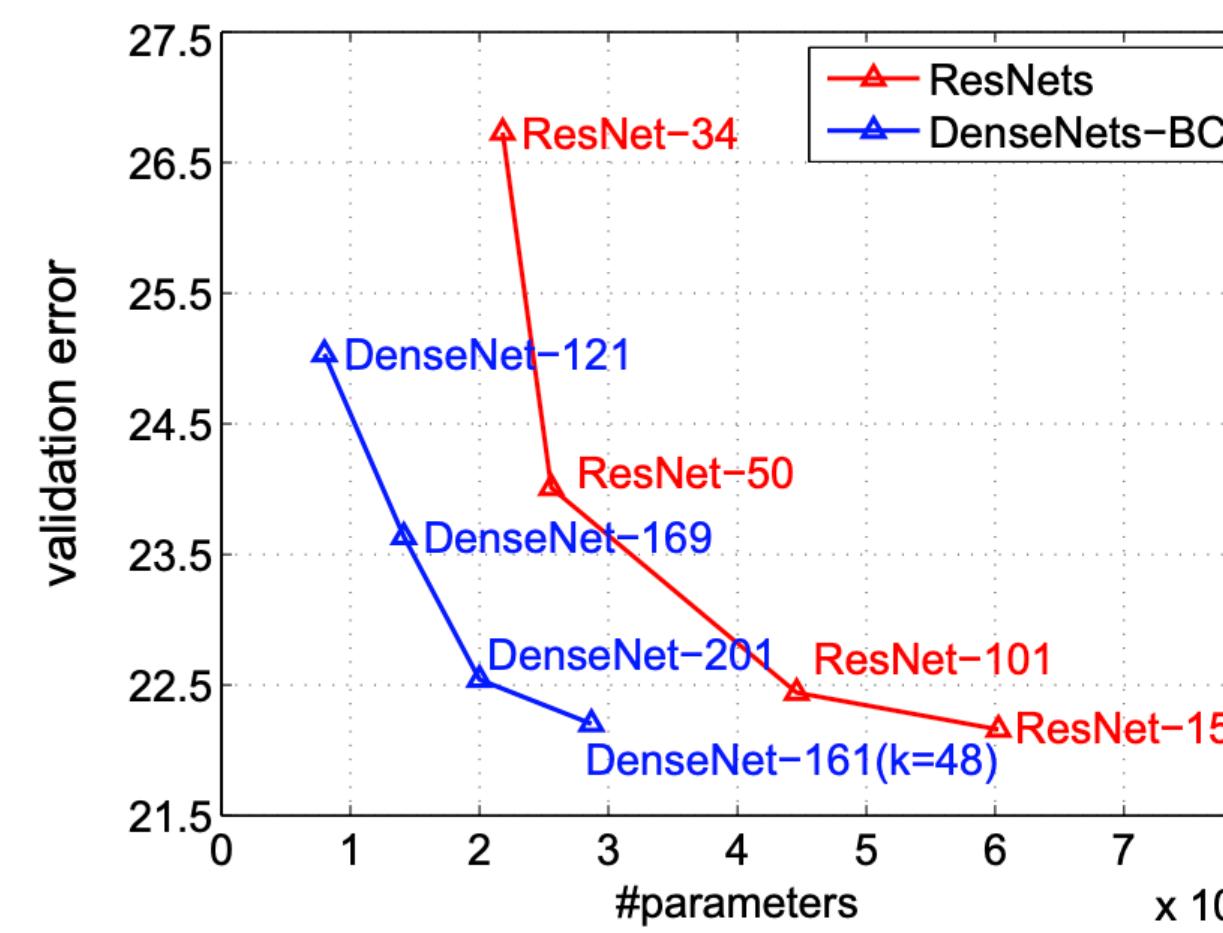
$x_\ell = H_\ell(\underbrace{[x_0, x_1, \dots, x_{\ell-1}]}_{\text{concatenate}})$

BN-ReLU-Conv(1×1)-BN-ReLU-Conv(3×3)  
↳ 4k feature maps

growth rate of the network  
 $k = 12$

If each function  $H_\ell$  produces  $k$  feature-maps,  
then the  $\ell$ -th layer has  $k_0 + k(\ell - 1)$  input feature maps

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-265
Convolution	112 × 112				
Pooling	56 × 56				
Dense Block (1)	56 × 56	[ 1 × 1 conv 3 × 3 conv ] × 6	[ 1 × 1 conv 3 × 3 conv ] × 6	[ 1 × 1 conv 3 × 3 conv ] × 6	[ 1 × 1 conv 3 × 3 conv ] × 6
	56 × 56				
Transition Layer (1)	56 × 56			1 × 1 conv	
	28 × 28			2 × 2 average pool, stride 2	
Dense Block (2)	28 × 28	[ 1 × 1 conv 3 × 3 conv ] × 12	[ 1 × 1 conv 3 × 3 conv ] × 12	[ 1 × 1 conv 3 × 3 conv ] × 12	[ 1 × 1 conv 3 × 3 conv ] × 12
	28 × 28				
Transition Layer (2)	28 × 28			1 × 1 conv	
	14 × 14			2 × 2 average pool, stride 2	
Dense Block (3)	14 × 14	[ 1 × 1 conv 3 × 3 conv ] × 24	[ 1 × 1 conv 3 × 3 conv ] × 32	[ 1 × 1 conv 3 × 3 conv ] × 48	[ 1 × 1 conv 3 × 3 conv ] × 64
	14 × 14				
Transition Layer (3)	14 × 14			1 × 1 conv	
	7 × 7			2 × 2 average pool, stride 2	
Dense Block (4)	7 × 7	[ 1 × 1 conv 3 × 3 conv ] × 16	[ 1 × 1 conv 3 × 3 conv ] × 32	[ 1 × 1 conv 3 × 3 conv ] × 32	[ 1 × 1 conv 3 × 3 conv ] × 48
	7 × 7				
Classification Layer	1 × 1			7 × 7 global average pool	
					1000D fully-connected, softmax



Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

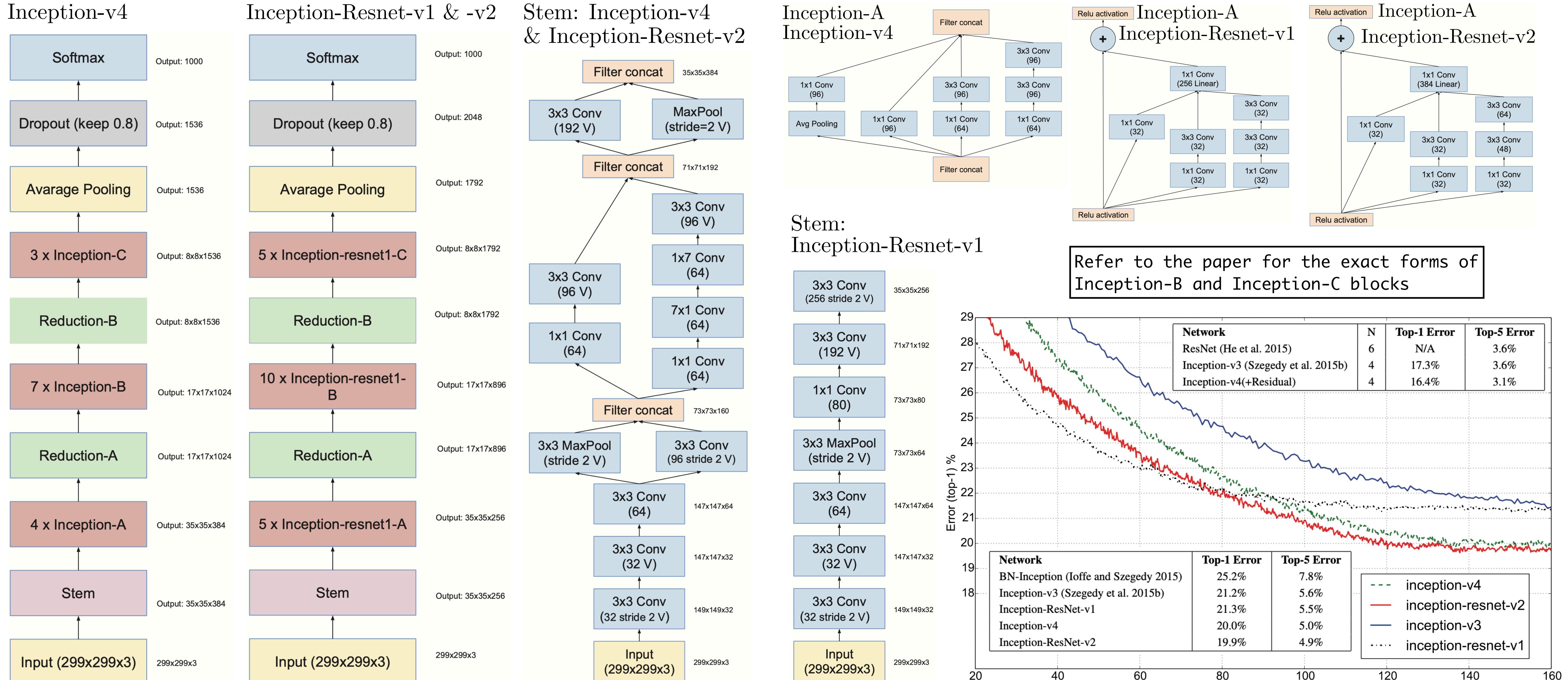
Huang, Gao, et al. "Convolutional networks with dense connectivity." IEEE transactions on pattern analysis and machine intelligence (2019).



# Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning



[YouTube Video](#)





Boulder

# mixup: Beyond Empirical Risk Minimization

data-agnostic data augmentation

**Empirical Risk Minimization (ERM)**

$f \in \mathcal{F} \rightarrow$  function

$X \rightarrow$  random feature vector

$Y \rightarrow$  random target vector

$P(X, Y) \rightarrow$  joint distribution

$\ell \rightarrow$  loss function (penalizes the difference btw predictions  $f(x)$  and actual targets  $y$ , for examples  $(x, y) \sim P$ )

$\mathcal{R}(f) = \int \ell(f(x), y) dP(x, y) \rightarrow$  expected risk

$P$  is unknown!

$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n \rightarrow$  training data

$(x_i, y_i) \sim P, i = 1, \dots, n$

$P_\delta(x, y) = \frac{1}{n} \sum_{i=1}^n \delta(x = x_i, y = y_i) \rightarrow$  empirical distribution

$\mathcal{R}_\delta(f) = \int \ell(f(x), y) dP_\delta(x, y) = \frac{1}{n} \sum_{i=1}^n \ell(f(x_i), y_i) \rightarrow$  empirical risk

**Vicinal Risk Minimization (VRM)**

$P_\nu(\tilde{x}, \tilde{y}) = \frac{1}{n} \sum_{i=1}^n \nu(\tilde{x}, \tilde{y} | x_i, y_i)$

$\nu \rightarrow$  vicinity distribution (measures the probability of finding the virtual feature-target  $(\tilde{x}, \tilde{y})$  in the vicinity of the training feature-target  $(x_i, y_i)$  pair)

$\nu(\tilde{x}, \tilde{y} | x_i, y_i) = \mathcal{N}(\tilde{x} - x_i, \sigma^2) \delta(\tilde{y} = y_i) \rightarrow$  Gaussian vicinity

(augmenting the training data with additive Gaussian noise)

$$\mathcal{R}_\nu(f) = \frac{1}{m} \sum_{i=1}^m \ell(f(\tilde{x}_i), \tilde{y}_i) \rightarrow \text{empirical vicinal risk}$$

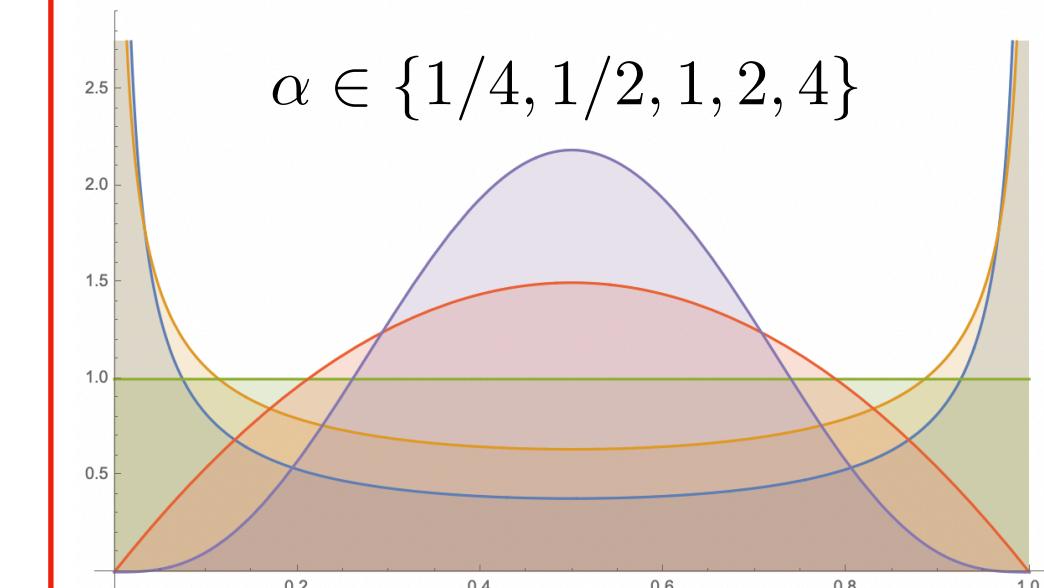
$$\mathcal{D}_\nu := \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^m$$

**mixup**

$$\mu(\tilde{x}, \tilde{y} | x_i, y_i) = \frac{1}{n} \sum_j^n \mathbb{E} [\delta(\tilde{x} = \lambda \cdot x_i + (1 - \lambda) \cdot x_j, \tilde{y} = \lambda \cdot y_i + (1 - \lambda) \cdot y_j)]$$

$\lambda \sim \text{Beta}(\alpha, \alpha)$ , for  $\alpha \in (0, \infty)$

$\alpha \rightarrow 0 \implies$  recovering ERM



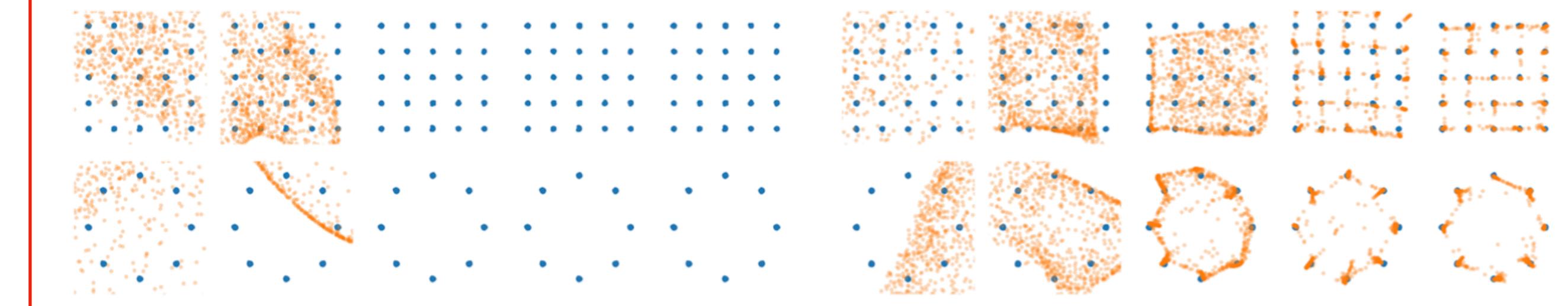
**mixup GANs**

$$\max_g \min_d \mathbb{E}_{x,z} [\ell(d(x), 1) + \ell(d(g(z)), 0)]$$

**ERM GAN**

$$\max_g \min_d \mathbb{E}_{x,z,\lambda} [\ell(d(\lambda x + (1 - \lambda)g(z)), \lambda)]$$

**mixup GAN ( $\alpha = 0.2$ )**





Boulder

# Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour

## Stochastic Gradient Descent (SGD)

$$L(w) = \frac{1}{|X|} \sum_{x \in X} l(x; w)$$

$L(w)$  → loss function

$w$  → weights

$X$  → labeled training set

$l(x; w)$  → loss computed from sample  $x$  and its corresponding label (e.g., cross-entropy + a regularization on  $w$ )

$$w_{t+1} = w_t - \eta \frac{1}{|\mathcal{B}|} \sum_{x \in \mathcal{B}} \nabla l(x; w_t) \rightarrow \text{SGD}$$

$\mathcal{B}$  → mini-batch sampled from  $X$

$|\mathcal{B}|$  → mini-batch size

$k$  → number of GPUs in a server

$n$  → number of samples per GPU

Baseline:  $k = 8$ ,  $n = 32$ , and  $|\mathcal{B}| = kn = 256$ .

$\eta = 0.1$  → learning rate

## Learning rates for large mini-batches

data parallelism (multiple servers)

$$\text{Linear Scaling Rule: } \eta = 0.1 \frac{|\mathcal{B}|}{256} = 0.1 \frac{kn}{256}$$

**Linear Scaling Rule:** When the minibatch size is multiplied by  $k$ , multiply the learning rate by  $k$ .

$\mathcal{B}_j, 0 \leq j < k \rightarrow$  sequence of  $k$  minibatches of size  $n$

$$\bigcup_{0 \leq j < k} \mathcal{B}_j \rightarrow \text{large minibatch of size } kn$$

$w_{t+k} = w_t - \eta \frac{1}{n} \sum_{0 \leq j < k} \sum_{x \in \mathcal{B}_j} \nabla l(x; w_{t+j}) \rightarrow k$  iterations of SGD with learning rate  $\eta$  and a mini-batch size of  $n$

$\hat{w}_{t+1} = w_t - \hat{\eta} \frac{1}{kn} \sum_{0 \leq j < k} \sum_{x \in \mathcal{B}_j} \nabla l(x; w_t) \rightarrow$  one iteration of SGD with learning rate  $\hat{\eta}$  and large mini-batch size of  $kn$

$$\hat{w}_{t+1} \neq w_{t+k}$$

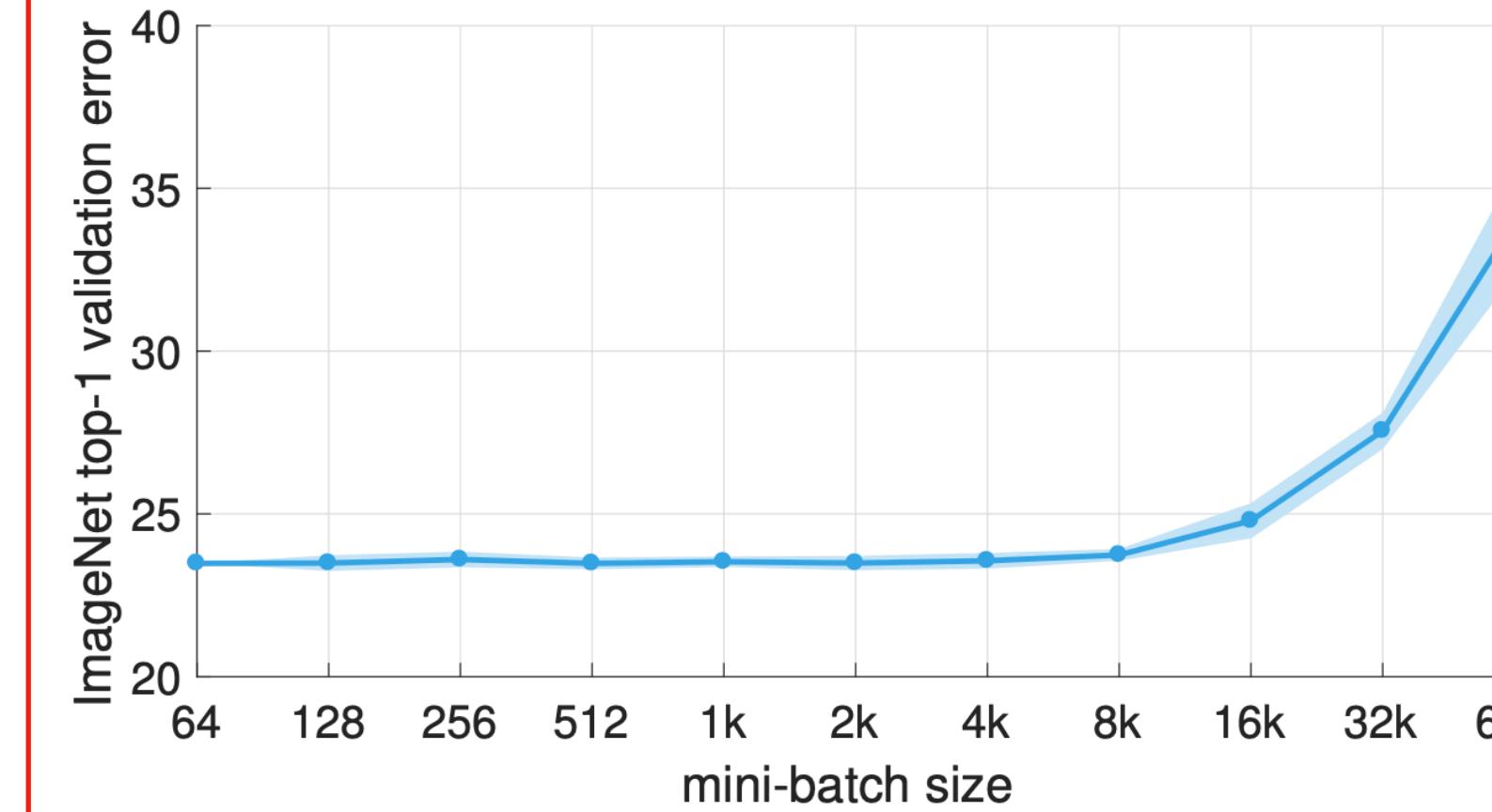
$\nabla l(x; w_t) \approx \nabla l(x; w_{t+j}) \rightarrow$  assume

$$\hat{\eta} = k\eta \implies \hat{w}_{t+1} \approx w_{t+k}$$

## Gradual Warmup

$\nabla l(x; w_t) \approx \nabla l(x; w_{t+j}) \rightarrow$  does not hold initially during training when the network is changing rapidly

Start with a learning rate of  $\eta$  and increment it by a constant amount at each iteration until it reaches  $\hat{\eta} = kn$  after 5 epochs



## Batch normalization

$l_{\mathcal{B}}(x; w) \rightarrow$  loss of a single sample  $x$  depends on the statistic of all samples in its mini-batch  $\mathcal{B}$

$$L(w) = \frac{1}{|X^n|} \sum_{\mathcal{B} \in X^n} L(\mathcal{B}, w) = \frac{1}{n} \sum_{x \in \mathcal{B}} l_{\mathcal{B}}(x, w)$$

$X^n \rightarrow$  all distinct subsets of size  $n$  drawn from the original training set  $X$

$L(w)$  changes as  $n$  changes!

Subtleties and Pitfalls of Distributed SGD (see the paper!)

## Batch normalization with large mini-batches

The BN statistics should not be computed across all workers, not only for the sake of reducing communication, but also for maintaining the same underlying loss function being optimized.



Boulder

# SGDR: Stochastic Gradient Descent with Warm Restarts

$n \rightarrow$  number of free parameters

$f : \mathbb{R}^n \rightarrow \mathbb{R}$

↳ function to be minimized

$x_t \in \mathbb{R}^n \rightarrow$  parameter vector at time step  $t$

$\nabla f_t(x_t) \rightarrow$  gradient information obtained on  
relatively small  $t$ -th batch of  $b$  data-points

Stochastic Gradient Descent (SGD)

$x_{t+1} = x_t - \eta_t \nabla f_t(x_t)$

$\eta_t \rightarrow$  learning rate

Stochastic Gradient Descent  
with Warm Restarts (SGDR)

$i \rightarrow$  indexes a new warm-started run

Restart SGD once  $T_i$  epochs are performed

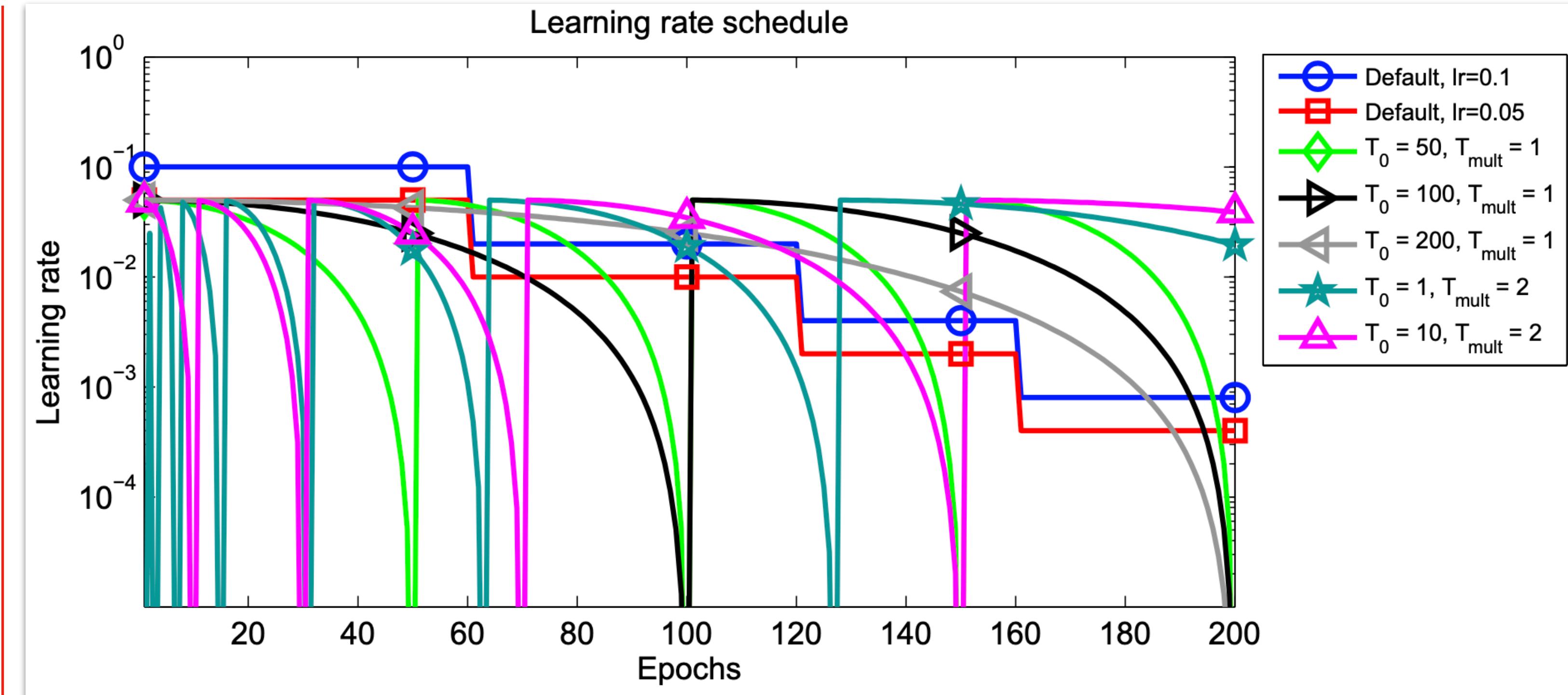
$$\eta_t = \eta_{min}^i + \frac{1}{2}(\eta_{max}^i - \eta_{min}^i)(1 + \cos(\frac{T_{cur}}{T_i}\pi))$$

$T_{cur} \rightarrow$  number of epochs performed since the  
last restart

$T_{cur}$  is updated at each batch iteration  $t$ , it  
can take fractional values such as 0.1, 0.2, etc.

$$t = 0 \implies \eta_t = \eta_{max}^i$$

$$T_{cur} = T_i \implies \eta_t = \eta_{min}^i$$



	depth- $k$	# params	# runs	CIFAR-10	CIFAR-100
WRN (ours)					
default with $\eta_0 = 0.1$	28-10	36.5M	med. of 5	4.24	20.33
default with $\eta_0 = 0.05$	28-10	36.5M	med. of 5	4.13	20.21
$T_0 = 50, T_{mult} = 1$	28-10	36.5M	med. of 5	4.17	19.99
$T_0 = 100, T_{mult} = 1$	28-10	36.5M	med. of 5	4.07	19.87
$T_0 = 200, T_{mult} = 1$	28-10	36.5M	med. of 5	3.86	19.98
$T_0 = 1, T_{mult} = 2$	28-10	36.5M	med. of 5	4.09	19.74
$T_0 = 10, T_{mult} = 2$	28-10	36.5M	med. of 5	4.03	19.58



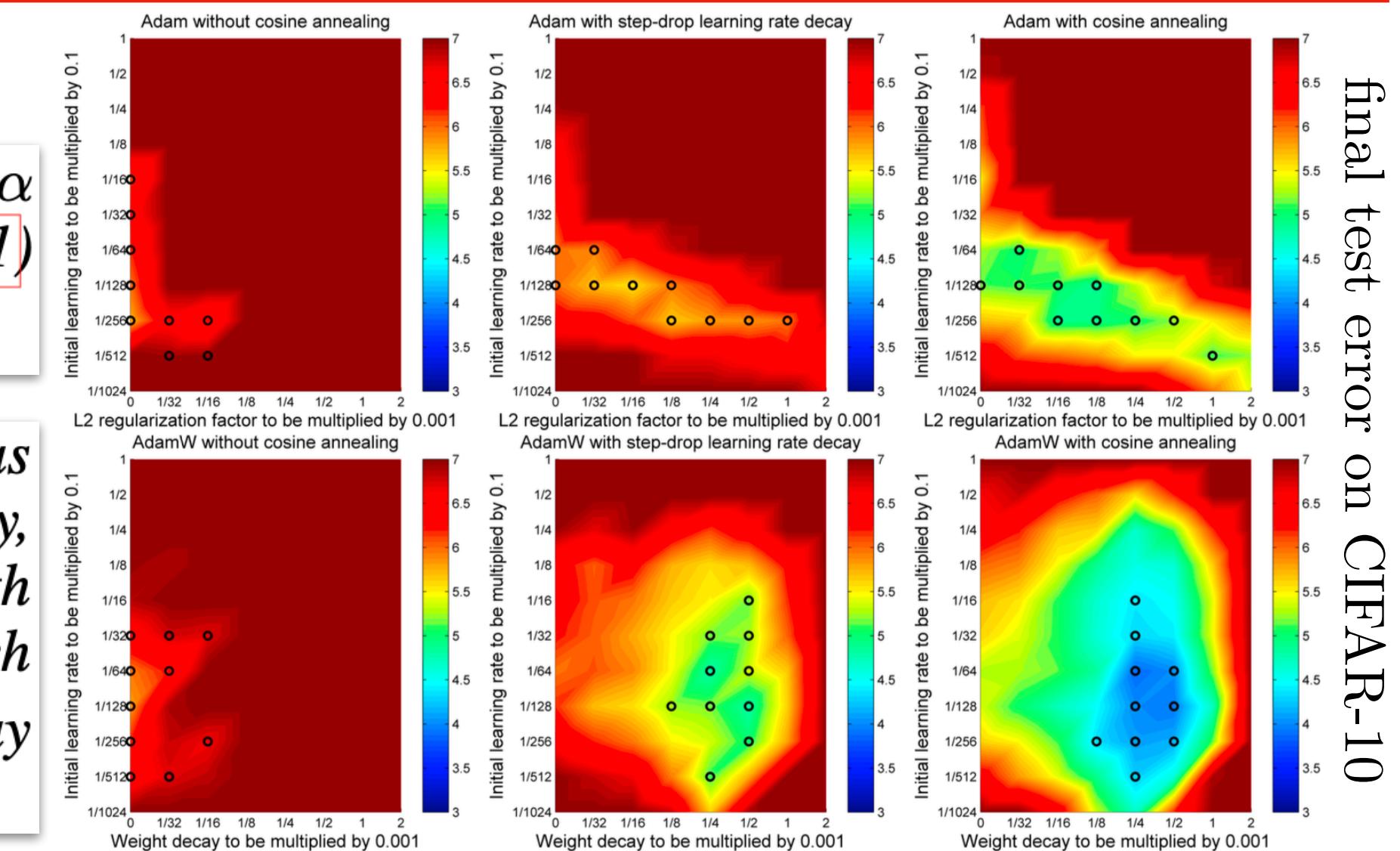
Boulder

# Decoupled Weight Decay Regularization

$$\theta_{t+1} = (1 - \lambda)\theta_t - \alpha \nabla f_t(\theta_t) \rightarrow \text{weight decay}$$

**Proposition 1** (Weight decay = L<sub>2</sub> reg for standard SGD). *Standard SGD with base learning rate  $\alpha$  executes the same steps on batch loss functions  $f_t(\theta)$  with weight decay  $\lambda$  (defined in Equation 1) as it executes without weight decay on  $f_t^{\text{reg}}(\theta) = f_t(\theta) + \frac{\lambda'}{2} \|\theta\|_2^2$ , with  $\lambda' = \frac{\lambda}{\alpha}$ .*

**Proposition 2** (Weight decay  $\neq$  L<sub>2</sub> reg for adaptive gradients). *Let  $O$  denote an optimizer that has iterates  $\theta_{t+1} \leftarrow \theta_t - \alpha M_t \nabla f_t(\theta_t)$  when run on batch loss function  $f_t(\theta)$  without weight decay, and  $\theta_{t+1} \leftarrow (1 - \lambda)\theta_t - \alpha M_t \nabla f_t(\theta_t)$  when run on  $f_t(\theta)$  with weight decay, respectively, with  $M_t \neq kI$  (where  $k \in \mathbb{R}$ ). Then, for  $O$  there exists no L<sub>2</sub> coefficient  $\lambda'$  such that running  $O$  on batch loss  $f_t^{\text{reg}}(\theta) = f_t(\theta) + \frac{\lambda'}{2} \|\theta\|_2^2$  without weight decay is equivalent to running  $O$  on  $f_t(\theta)$  with decay  $\lambda \in \mathbb{R}^+$ .*



**Algorithm 1** SGD with L<sub>2</sub> regularization and SGD with decoupled weight decay (SGDW), both with momentum

```

1: given initial learning rate  $\alpha \in \mathbb{R}$ , momentum factor  $\beta_1 \in \mathbb{R}$ , weight decay/L2 regularization factor  $\lambda \in \mathbb{R}$ 
2: initialize time step  $t \leftarrow 0$ , parameter vector  $\theta_{t=0} \in \mathbb{R}^n$ , first moment vector  $m_{t=0} \leftarrow \theta$ , schedule
   multiplier  $\eta_{t=0} \in \mathbb{R}$ 
3: repeat
4:    $t \leftarrow t + 1$ 
5:    $\nabla f_t(\theta_{t-1}) \leftarrow \text{SelectBatch}(\theta_{t-1})$                                  $\triangleright$  select batch and return the corresponding gradient
6:    $\mathbf{g}_t \leftarrow \nabla f_t(\theta_{t-1}) + \lambda \theta_{t-1}$                                 $\triangleright$  can be fixed, decay, be used for warm restarts
7:    $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$ 
8:    $\mathbf{m}_t \leftarrow \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \mathbf{g}_t$ 
9:    $\theta_t \leftarrow \theta_{t-1} - \mathbf{m}_t - \eta_t \lambda \theta_{t-1}$                                  $\triangleright$  select batch and return the corresponding gradient
10:  until stopping criterion is met
11:  return optimized parameters  $\theta_t$ 

```

**Algorithm 2** Adam with L<sub>2</sub> regularization and Adam with decoupled weight decay (AdamW)

```

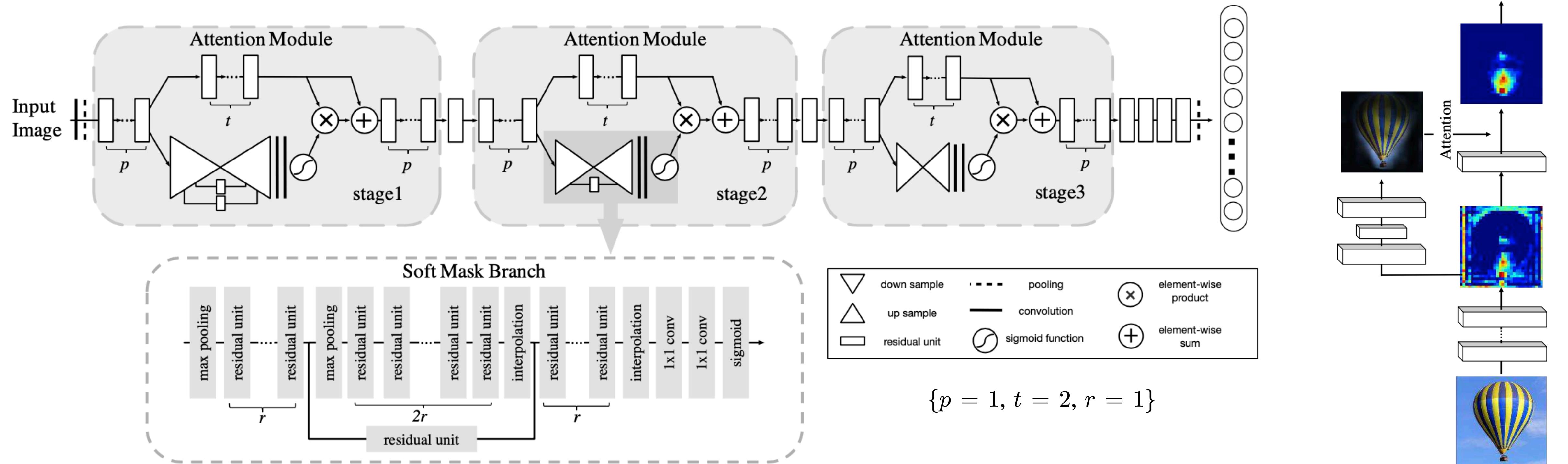
1: given  $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, \lambda \in \mathbb{R}$ 
2: initialize time step  $t \leftarrow 0$ , parameter vector  $\theta_{t=0} \in \mathbb{R}^n$ , first moment vector  $\mathbf{m}_{t=0} \leftarrow \theta$ , second moment
   vector  $\mathbf{v}_{t=0} \leftarrow \theta$ , schedule multiplier  $\eta_{t=0} \in \mathbb{R}$ 
3: repeat
4:    $t \leftarrow t + 1$ 
5:    $\nabla f_t(\theta_{t-1}) \leftarrow \text{SelectBatch}(\theta_{t-1})$                                  $\triangleright$  select batch and return the corresponding gradient
6:    $\mathbf{g}_t \leftarrow \nabla f_t(\theta_{t-1}) + \lambda \theta_{t-1}$                                 $\triangleright$  here and below all operations are element-wise
7:    $\mathbf{m}_t \leftarrow \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \mathbf{g}_t$                                  $\triangleright$   $\beta_1$  is taken to the power of  $t$ 
8:    $\mathbf{v}_t \leftarrow \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) \mathbf{g}_t^2$                                  $\triangleright$   $\beta_2$  is taken to the power of  $t$ 
9:    $\hat{\mathbf{m}}_t \leftarrow \mathbf{m}_t / (1 - \beta_1^t)$ 
10:   $\hat{\mathbf{v}}_t \leftarrow \mathbf{v}_t / (1 - \beta_2^t)$ 
11:   $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$                                  $\triangleright$  can be fixed, decay, or also be used for warm restarts
12:   $\theta_t \leftarrow \theta_{t-1} - \eta_t \left( \alpha \hat{\mathbf{m}}_t / (\sqrt{\hat{\mathbf{v}}_t} + \epsilon) + \lambda \theta_{t-1} \right)$ 
13:  until stopping criterion is met
14:  return optimized parameters  $\theta_t$ 

```



Boulder

# Residual Attention Network for Image Classification



## Attention Module

- Mask branch (bottom-up top-down structure)
- Trunk branch (performs feature processing)

$$H_{i,c}(x) = (1 + M_{i,c}(x)) * F_{i,c}(x) \rightarrow \text{Attention Residual Learning (ARL)}$$

$i \rightarrow$  ranges over all spatial positions

$c \rightarrow$  index of the channel

$$f_1(x_{i,c}) = \frac{1}{1 + \exp(-x_{i,c})} \rightarrow \text{Mixed Spatial and Channel Attention}$$

Network	params $\times 10^6$	FLOPs $\times 10^9$	Test Size	Top-1 err. (%)	Top-5 err. (%)
ResNet-152 [10]	60.2	11.3	224 $\times$ 224	22.16	6.16
Attention-56	31.9	6.3	224 $\times$ 224	<b>21.76</b>	<b>5.9</b>
ResNeXt-101 [36]	44.5	7.8	224 $\times$ 224	21.2	5.6
AttentionNeXt-56	31.9	6.3	224 $\times$ 224	<b>21.2</b>	<b>5.6</b>
Inception-ResNet-v1 [32]	-	-	299 $\times$ 299	21.3	5.5
AttentionInception-56	31.9	6.3	299 $\times$ 299	<b>20.36</b>	<b>5.29</b>
ResNet-200 [11]	64.7	15.0	320 $\times$ 320	20.1	4.8
Inception-ResNet-v2	-	-	299 $\times$ 299	19.9	4.9
Attention-92	51.3	10.4	320 $\times$ 320	<b>19.5</b>	<b>4.8</b>

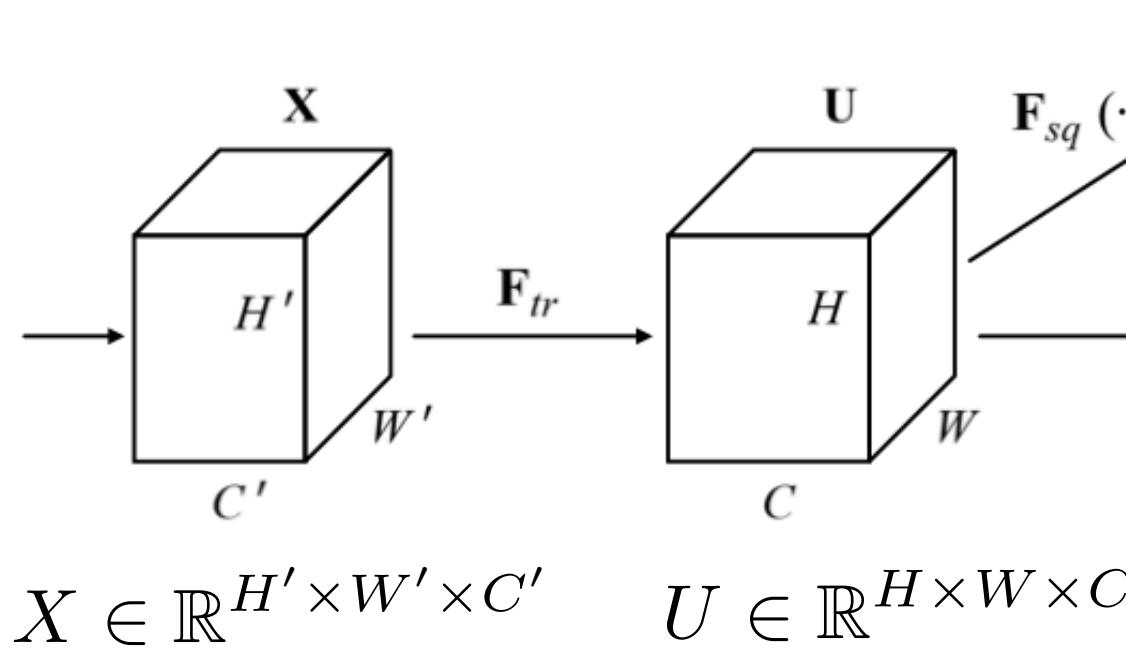


Boulder



[YouTube Video](#)

# Squeeze-and-Excitation Networks



$$X \in \mathbb{R}^{H' \times W' \times C'} \quad U \in \mathbb{R}^{H \times W \times C}$$

$$F_{tr} : X \mapsto U$$

$V = [v_1, \dots, v_C]$  → learned set of filter kernels

$v_c$  → parameters of the  $c$ -th filter

$$U = [u_1, \dots, u_C]$$

$$u_c = v_c * X = \sum_{s=1}^{C'} v_c^s * x^s$$

$$v_c = [v_c^1, \dots, v_c^{C'}]$$

$$X = [x^1, \dots, x^{C'}]$$

**Squeeze**

$$z \in \mathbb{R}^C \quad z_c = F_{sq}(u_c) = \frac{1}{HW} \sum_{i=1}^H \sum_{j=1}^W u_c(i, j)$$

**Excitation**

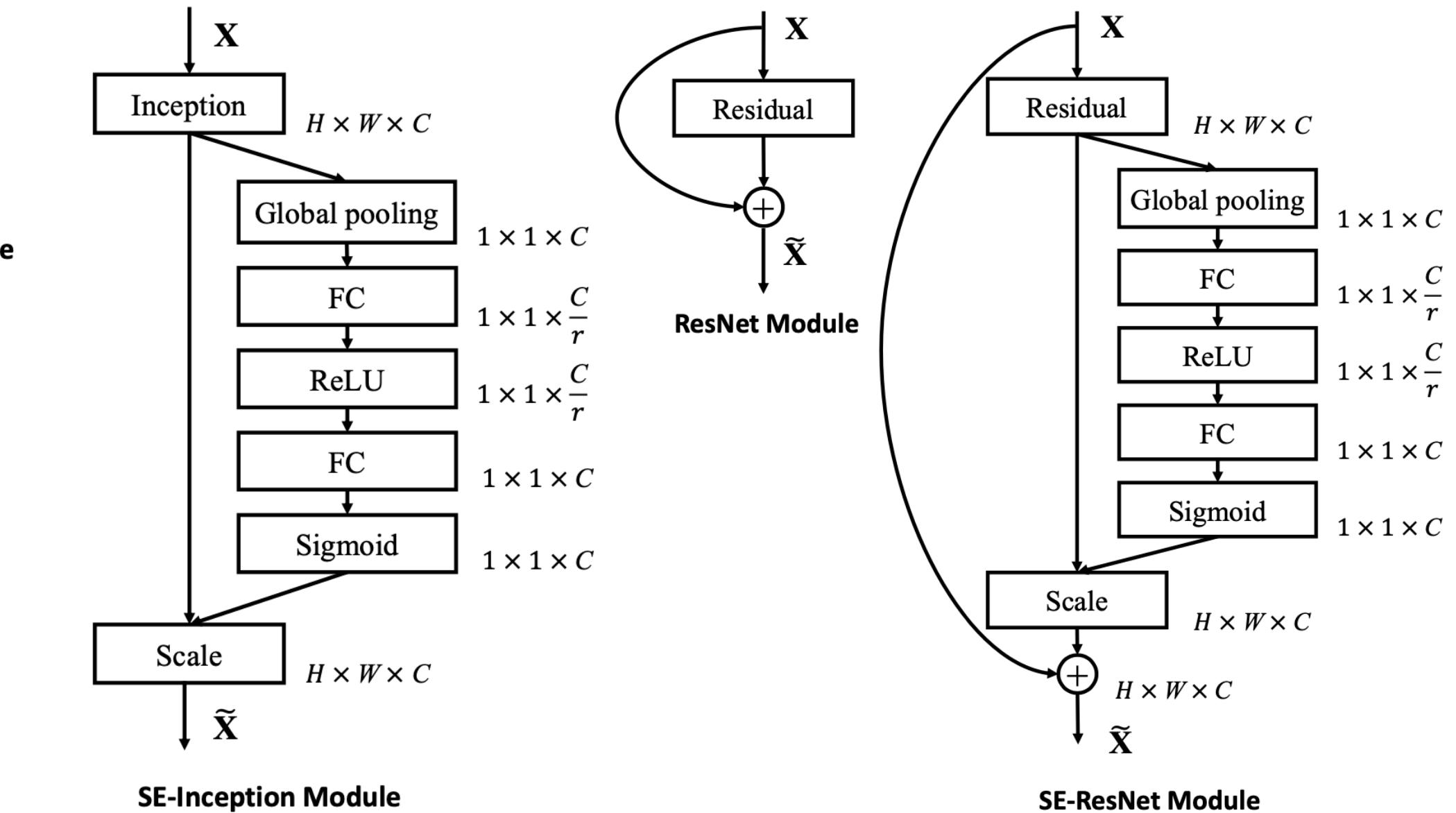
$$s = F_{ex}(z, W) = \sigma(g(z, W)) = \sigma(W_2 \delta(W_1 z))$$

$$W_1 \in \mathbb{R}^{\frac{C}{r} \times C}$$

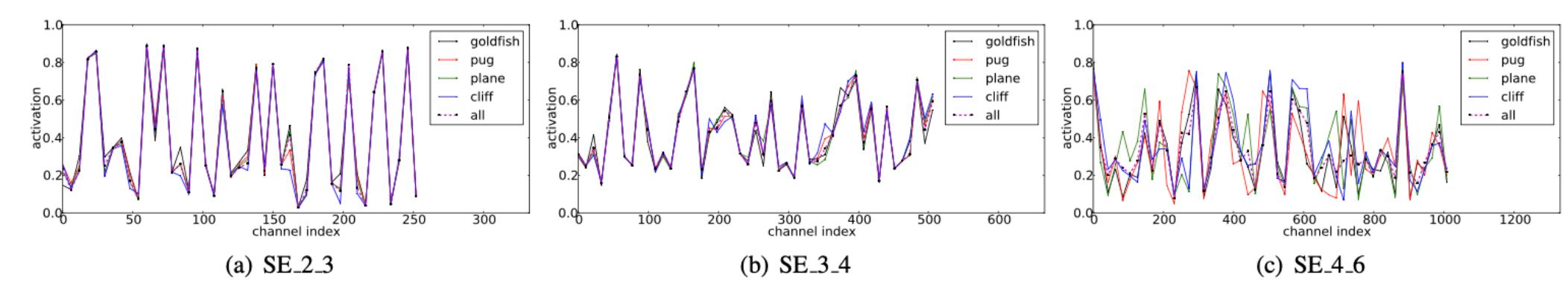
$$W_2 \in \mathbb{R}^{C \times \frac{C}{r}}$$

$$\begin{aligned} \tilde{x}_c &= F_{scale}(u_c, s_c) = s_c u_c \\ u_c &\in \mathbb{R}^{H \times W} \\ \tilde{X} &= [\tilde{x}_1, \dots, \tilde{x}_C] \end{aligned}$$

	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] <sup>†</sup>	-	-	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
<b>SENet-154</b>	<b>18.68</b>	<b>4.47</b>	<b>17.28</b>	<b>3.79</b>
NASNet-A (6@4032) [55] <sup>†</sup>	-	-	17.3 <sup>‡</sup>	3.8 <sup>‡</sup>
<b>SENet-154 (post-challenge)</b>	-	-	<b>16.88<sup>‡</sup></b>	<b>3.58<sup>‡</sup></b>



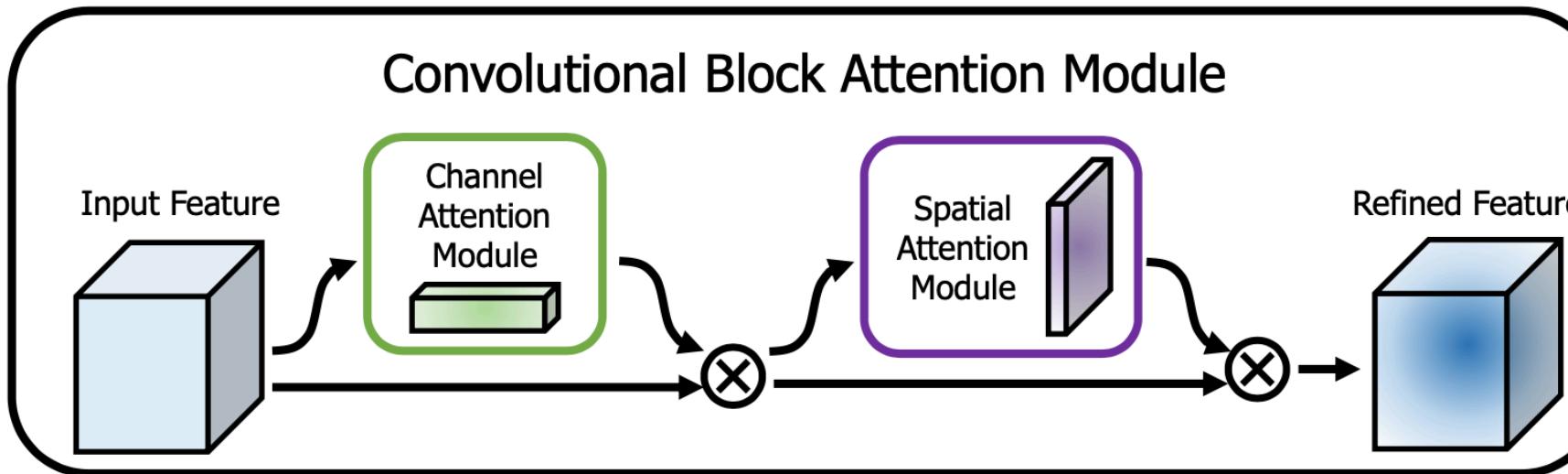
	original		re-implementation			SENet		
	top-1 err.	top-5 err.	top-1 err.	top-5 err.	GFLOPs	top-1 err.	top-5 err.	GFLOPs
ResNet-50 [10]	24.7	7.8	24.80	7.48	3.86	23.29 <sub>(1.51)</sub>	6.62 <sub>(0.86)</sub>	3.87
ResNet-101 [10]	23.6	7.1	23.17	6.52	7.58	22.38 <sub>(0.79)</sub>	6.07 <sub>(0.45)</sub>	7.60
ResNet-152 [10]	23.0	6.7	22.42	6.34	11.30	21.57 <sub>(0.85)</sub>	5.73 <sub>(0.61)</sub>	11.32
ResNeXt-50 [47]	22.2	-	22.11	5.90	4.24	21.10 <sub>(1.01)</sub>	5.49 <sub>(0.41)</sub>	4.25
ResNeXt-101 [47]	21.2	5.6	21.18	5.57	7.99	20.70 <sub>(0.48)</sub>	5.01 <sub>(0.56)</sub>	8.00
VGG-16 [39]	-	-	27.02	8.81	15.47	25.22 <sub>(1.80)</sub>	7.70 <sub>(1.11)</sub>	15.48
BN-Inception [16]	25.2	7.82	25.38	7.89	2.03	24.23 <sub>(1.15)</sub>	7.14 <sub>(0.75)</sub>	2.04
Inception-ResNet-v2 [42]	19.9 <sup>†</sup>	4.9 <sup>†</sup>	20.37	5.21	11.75	19.80 <sub>(0.57)</sub>	4.79 <sub>(0.42)</sub>	11.76





Boulder

# CBAM: Convolutional Block Attention Module



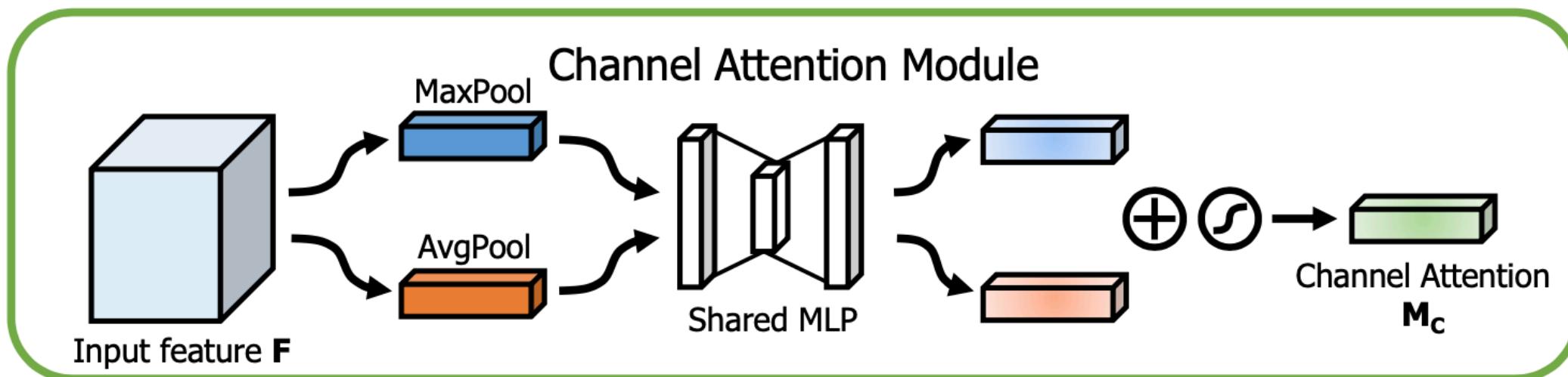
$$F \in \mathbb{R}^{C \times H \times W} \rightarrow \text{intermediate feature map}$$

$$M_c \in \mathbb{R}^{C \times 1 \times 1} \rightarrow \text{1D channel attention map}$$

$$M_s \in \mathbb{R}^{1 \times H \times W} \rightarrow \text{2D spatial attention map}$$

$$F' = M_c(F) \odot F$$

$$F'' = M_s(F') \odot F'$$

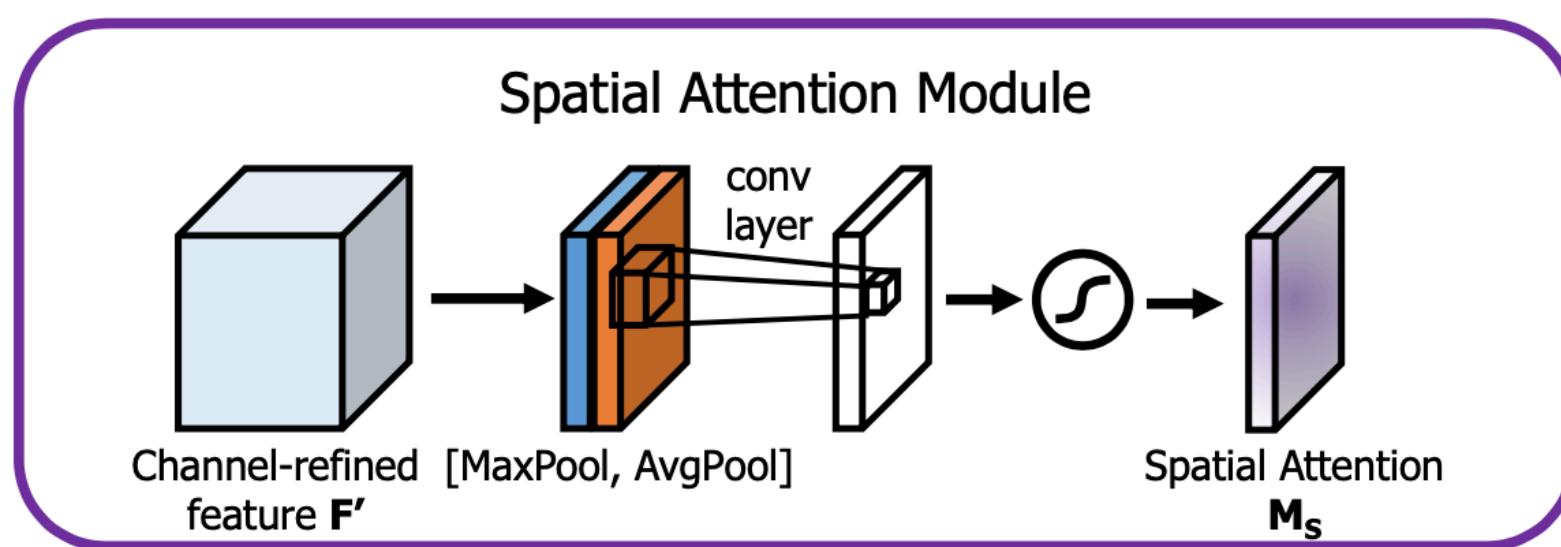


$$M_c(F) = \sigma(MLP(\text{AvgPool}(F)) + MLP(\text{MaxPool}(F)))$$

$$= \sigma(\mathbf{W}_1(\mathbf{W}_0(\mathbf{F}_{\text{avg}}^c)) + \mathbf{W}_1(\mathbf{W}_0(\mathbf{F}_{\text{max}}^c))),$$

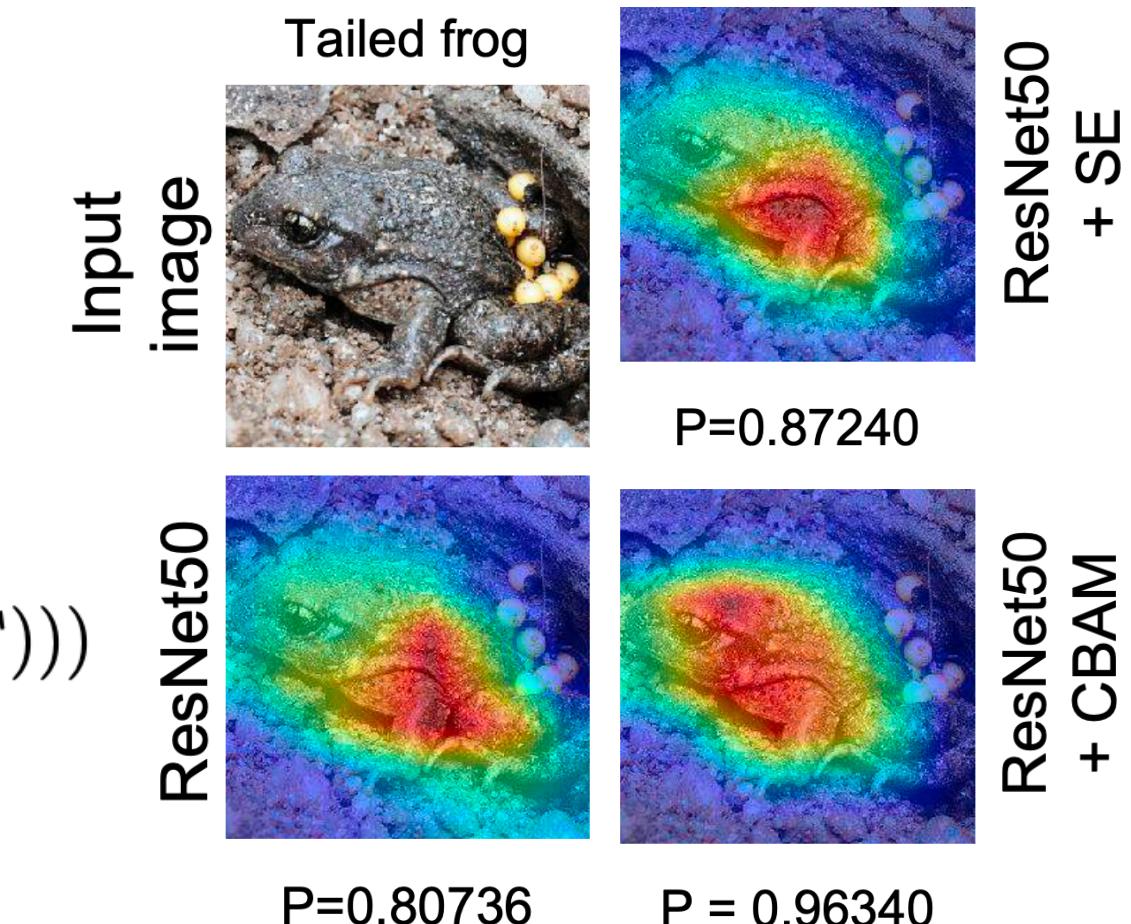
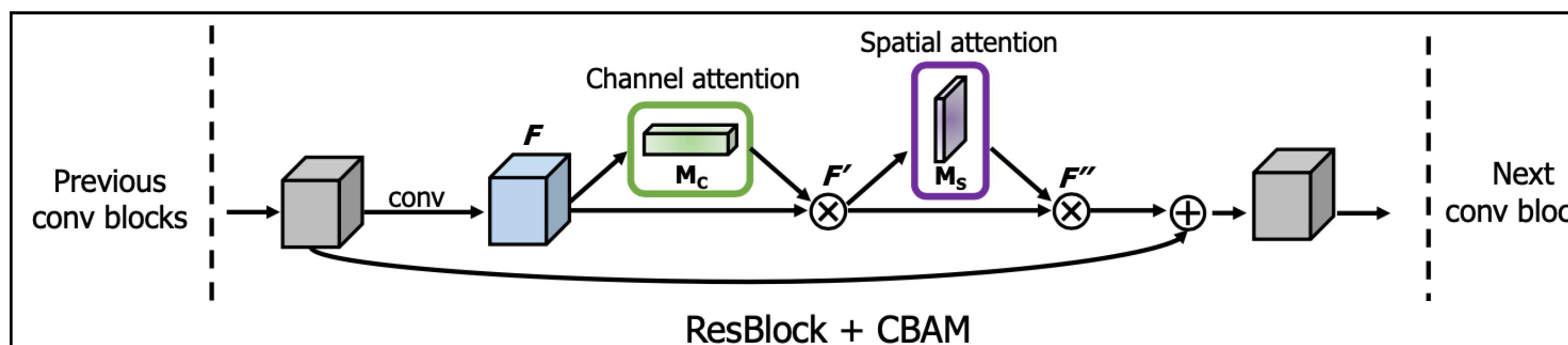
$$\mathbf{W}_0 \in \mathbb{R}^{C/r \times C}, \text{ and } \mathbf{W}_1 \in \mathbb{R}^{C \times C/r}$$

$r \rightarrow$  reduction ratio

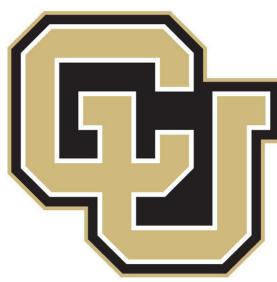


$$M_s(F) = \sigma(f^{7 \times 7}([\text{AvgPool}(F); \text{MaxPool}(F)]))$$

$$= \sigma(f^{7 \times 7}([\mathbf{F}_{\text{avg}}^s; \mathbf{F}_{\text{max}}^s])),$$

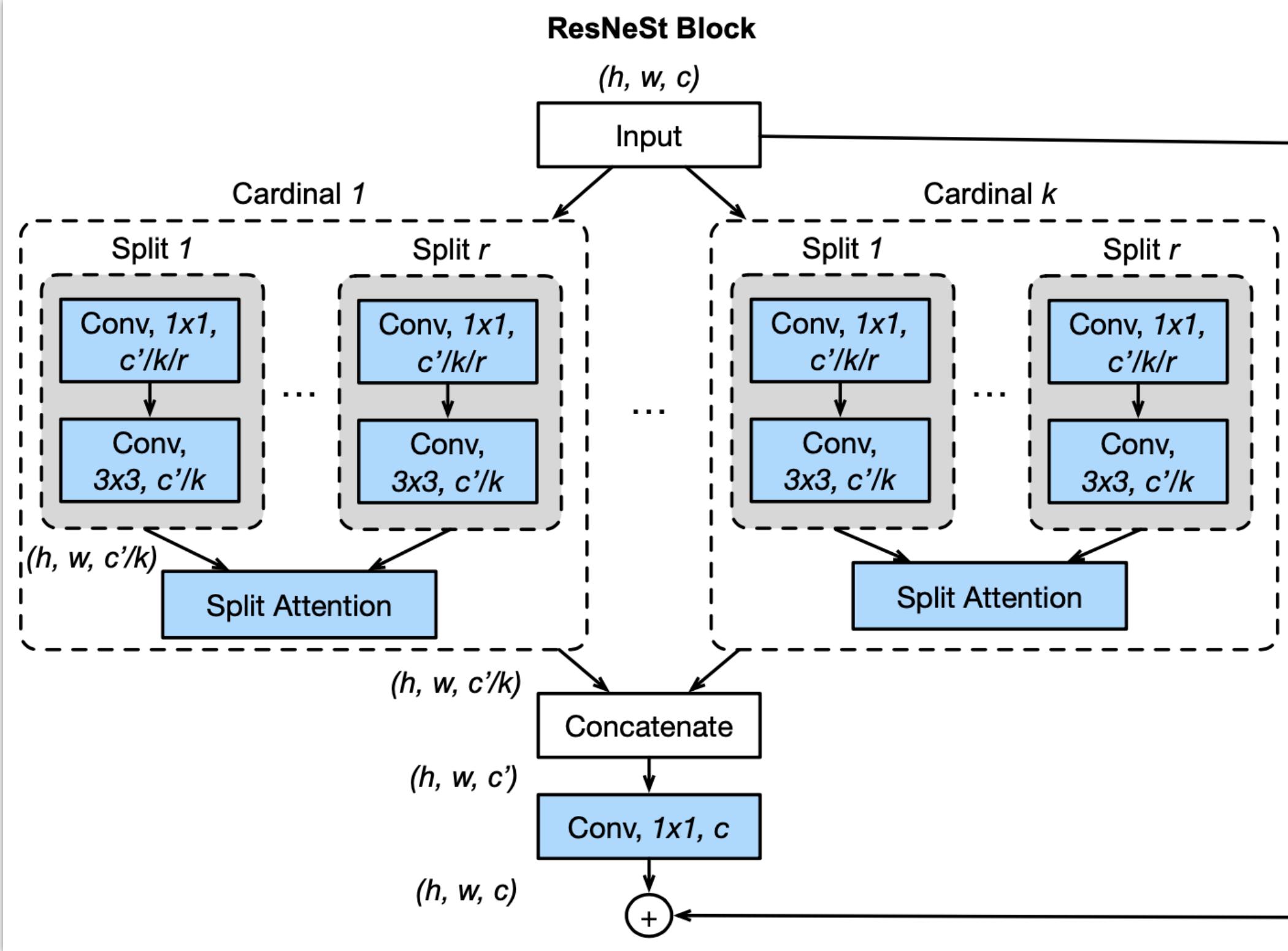


Architecture	Param.	GFLOPs	Top-1 Error (%)	Top-5 Error (%)
ResNet18 [5]	11.69M	1.814	29.60	10.55
ResNet18 [5] + SE [28]	11.78M	1.814	29.41	10.22
ResNet18 [5] + CBAM	11.78M	1.815	<b>29.27</b>	<b>10.09</b>
ResNet34 [5]	21.80M	3.664	26.69	8.60
ResNet34 [5] + SE [28]	21.96M	3.664	26.13	8.35
ResNet34 [5] + CBAM	21.96M	3.665	<b>25.99</b>	<b>8.24</b>
ResNet50 [5]	25.56M	3.858	24.56	7.50
ResNet50 [5] + SE [28]	28.09M	3.860	23.14	6.70
ResNet50 [5] + CBAM	28.09M	3.864	<b>22.66</b>	<b>6.31</b>
ResNet101 [5]	44.55M	7.570	23.38	6.88
ResNet101 [5] + SE [28]	49.33M	7.575	22.35	6.19
ResNet101 [5] + CBAM	49.33M	7.581	<b>21.51</b>	<b>5.69</b>
WideResNet18 [6] (widen=1.5)	25.88M	3.866	26.85	8.88
WideResNet18 [6] (widen=1.5) + SE [28]	26.07M	3.867	26.21	8.47
WideResNet18 [6] (widen=1.5) + CBAM	26.08M	3.868	<b>26.10</b>	<b>8.43</b>
WideResNet18 [6] (widen=2.0)	45.62M	6.696	25.63	8.20
WideResNet18 [6] (widen=2.0) + SE [28]	45.97M	6.696	24.93	7.65
WideResNet18 [6] (widen=2.0) + CBAM	45.97M	6.697	<b>24.84</b>	<b>7.63</b>
ResNeXt50 [7] (32x4d)	25.03M	3.768	22.85	6.48
ResNeXt50 [7] (32x4d) + SE [28]	27.56M	3.771	<b>21.91</b>	6.04
ResNeXt50 [7] (32x4d) + CBAM	27.56M	3.774	21.92	<b>5.91</b>
ResNeXt101 [7] (32x4d)	44.18M	7.508	21.54	5.75
ResNeXt101 [7] (32x4d) + SE [28]	48.96M	7.512	21.17	5.66
ResNeXt101 [7] (32x4d) + CBAM	48.96M	7.519	<b>21.07</b>	<b>5.59</b>



Boulder

# ResNeSt: Split-Attention Networks



global contextual information with embedded channel-wise statistics

$\mathcal{G}_i(s^j) \in \mathbb{R}^c \rightarrow$  two fully connected layers with ReLU activation

$a_i^j \in \mathbb{R}^c \rightarrow$  soft assignment weights

$$a_i^j = \exp(\mathcal{G}_i(s^j)) / \sum_{i'=1}^r \exp(\mathcal{G}_{i'}(s^j)) \text{ if } r > 1$$

$$a_i^j = 1 / (1 + \exp(-\mathcal{G}_i(s^j))) \text{ if } r = 1$$

$V^j = \sum_{i=1}^r a_i^j U_{r(j-1)+i} \in \mathbb{R}^{h \times w \times c} \rightarrow$  weighted fusion of cardinal group representation

See the paper for an efficient radix-major implementation!

ResNeSt = ResNeXt + Squeeze-and-Excitation

$X \in \mathbb{R}^{h \times w \times c} \rightarrow$  input to the split-attention block

$k \rightarrow$  cardinality hyperparameter (number of feature map groups)

$r \rightarrow$  radix hyperparameter (number of splits within a cardinal group)

$g = kr \rightarrow$  total number of feature groups

$U_i = \mathcal{F}_i(X), i = 1, \dots, g \rightarrow$  intermediate representation of each group

$\mathcal{F}_i \rightarrow 1 \times 1$  conv followed by a  $3 \times 3$  conv

$U_i \in \mathbb{R}^{h \times w \times (c'/k)}$

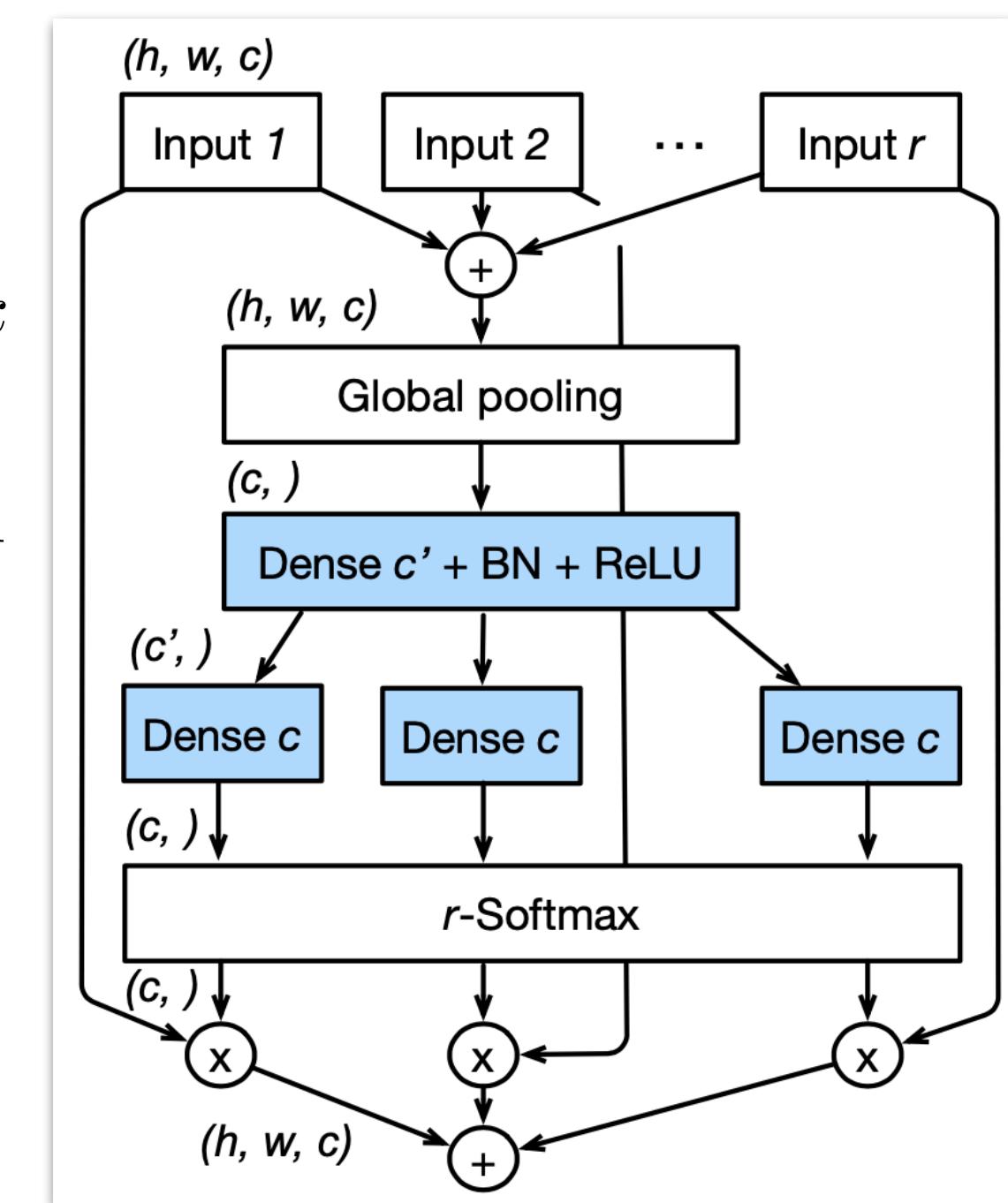
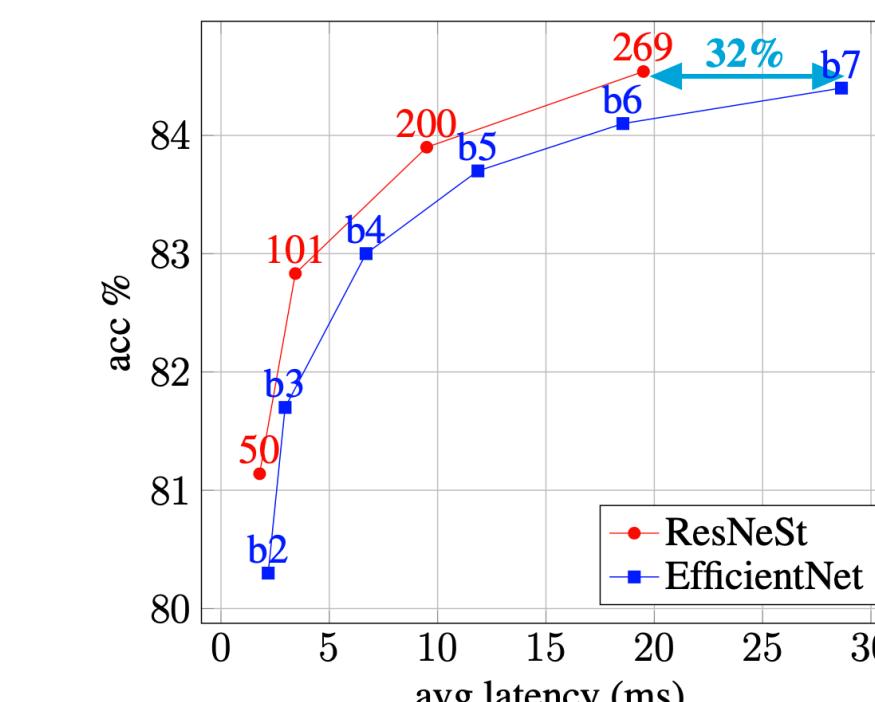
## Split Attention

For notation convenience, let us reuse  $c = c'/k$  so that  $U_i \in \mathbb{R}^{h \times w \times c}$

$\hat{U}^j = \sum_{i=1}^r U_{r(j-1)+i} \rightarrow$  representation for the  $j$ -th cardinal group

$\hat{U}^j \in \mathbb{R}^{h \times w \times c}, j = 1, \dots, k$

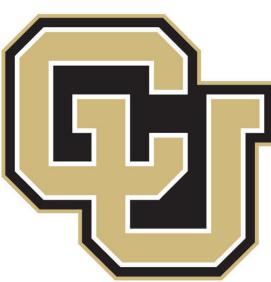
$$s^j = \frac{1}{hw} \sum_{i=1}^h \sum_{j=1}^w \hat{U}^j(i, j) \in \mathbb{R}^c$$



## ResNeSt Block

$$V = \text{concat}\{V^j, j = 1, \dots, k\}$$

$$Y = V + X$$



Boulder

# Random Erasing Data Augmentation

## Algorithm 1: Random Erasing Procedure

**Input** : Input image  $I$ ;  
 Image size  $W$  and  $H$ ;  
 Area of image  $S$ ;  
 Erasing probability  $p$ ;  
 Erasing area ratio range  $s_l$  and  $s_h$ ;  
 Erasing aspect ratio range  $r_1$  and  $r_2$ .

**Output:** Erased image  $I^*$ .

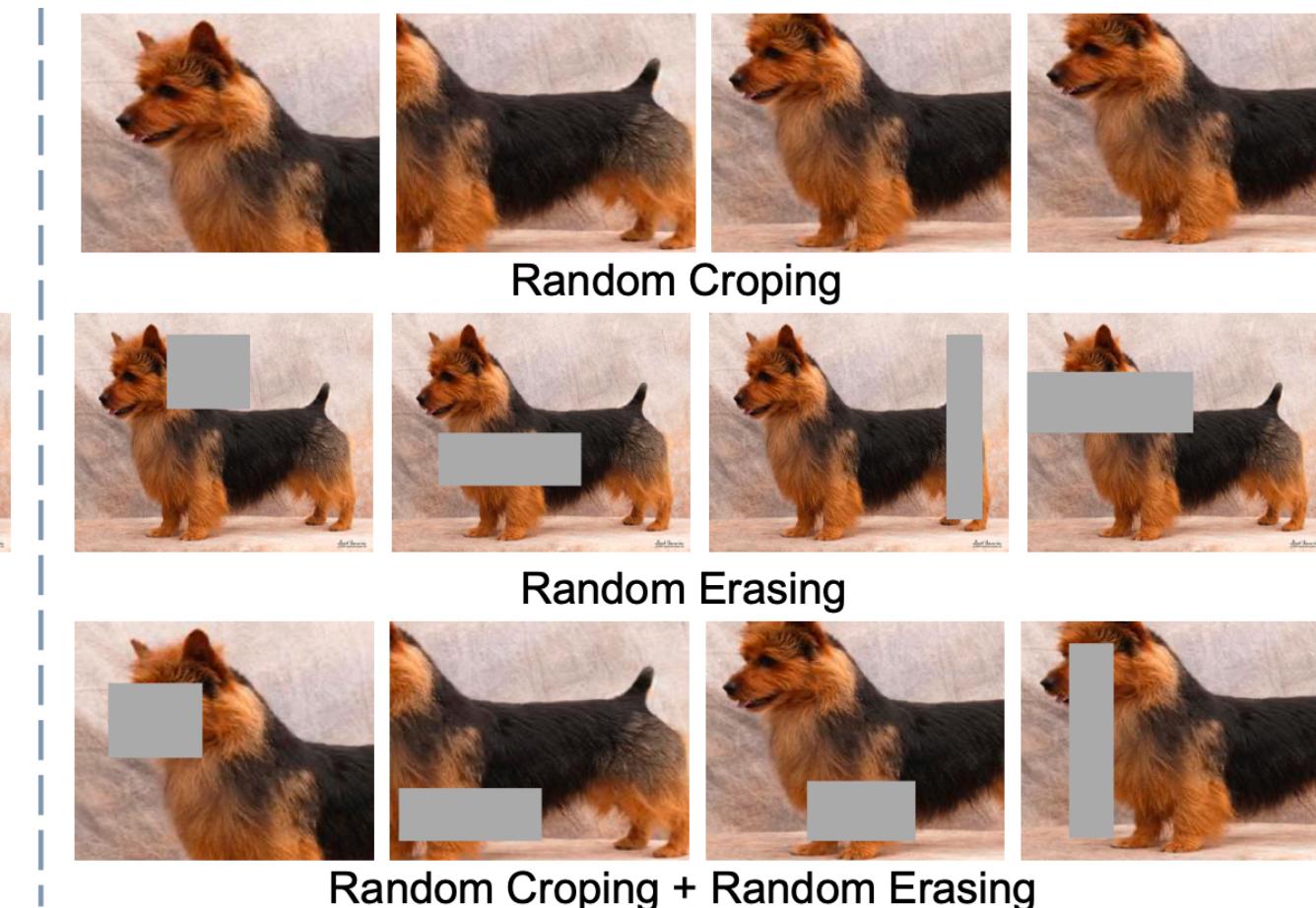
**Initialization:**  $p_1 \leftarrow \text{Rand}(0, 1)$ .

```

1 if  $p_1 \geq p$  then
2    $I^* \leftarrow I$ ;
3   return  $I^*$ .
4 else
5   while True do
6      $S_e \leftarrow \text{Rand}(s_l, s_h) \times S$ ;
7      $r_e \leftarrow \text{Rand}(r_1, r_2)$ ;
8      $H_e \leftarrow \sqrt{S_e \times r_e}$ ,  $W_e \leftarrow \sqrt{\frac{S_e}{r_e}}$ ;
9      $x_e \leftarrow \text{Rand}(0, W)$ ,  $y_e \leftarrow \text{Rand}(0, H)$ ;
10    if  $x_e + W_e \leq W$  and  $y_e + H_e \leq H$  then
11       $I_e \leftarrow (x_e, y_e, x_e + W_e, y_e + H_e)$ ;
12       $I(I_e) \leftarrow \text{Rand}(0, 255)$ ;
13       $I^* \leftarrow I$ ;
14      return  $I^*$ .
15    end
16  end
17 end
  
```



Model	CIFAR-100	
	Baseline	Random Erasing
ResNet-20	$30.84 \pm 0.19$	$29.97 \pm 0.11$
ResNet-32	$28.50 \pm 0.37$	$27.18 \pm 0.32$
ResNet-44	$25.27 \pm 0.21$	$24.29 \pm 0.16$
ResNet-56	$24.82 \pm 0.27$	$23.69 \pm 0.33$
ResNet-110	$23.73 \pm 0.37$	$22.10 \pm 0.41$
ResNet-20-PreAct	$30.58 \pm 0.16$	$30.18 \pm 0.13$
ResNet-32-PreAct	$29.04 \pm 0.25$	$27.82 \pm 0.28$
ResNet-44-PreAct	$25.22 \pm 0.19$	$24.10 \pm 0.26$
ResNet-56-PreAct	$24.14 \pm 0.25$	$22.93 \pm 0.27$
ResNet-110-PreAct	$22.11 \pm 0.20$	$20.99 \pm 0.11$
ResNet-18-PreAct	$24.50 \pm 0.29$	$24.03 \pm 0.19$
WRN-28-10	<b><math>18.49 \pm 0.11</math></b>	<b><math>17.73 \pm 0.15</math></b>
ResNeXt-8-64	$19.27 \pm 0.30$	$18.84 \pm 0.18$

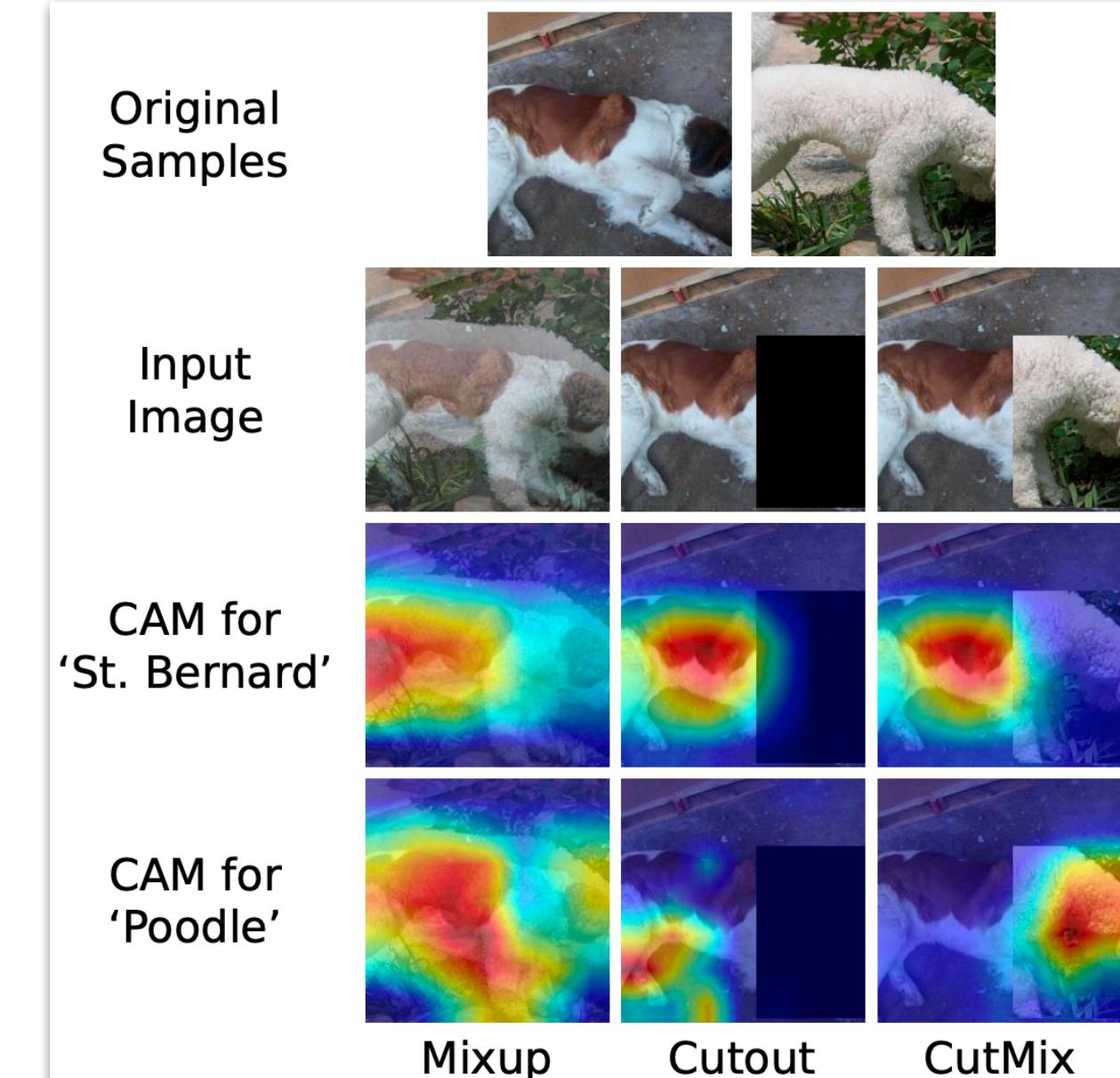
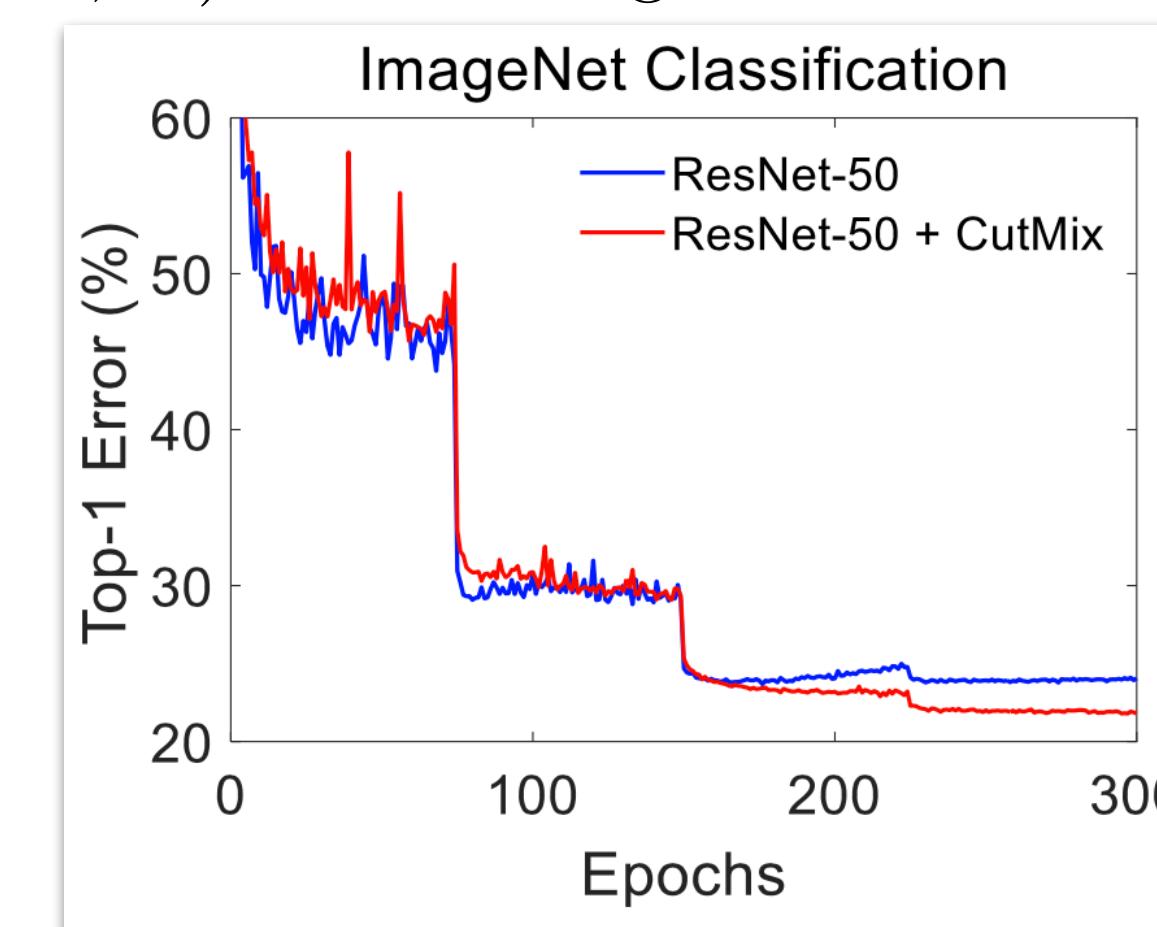


# CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features

	ResNet-50	Mixup [47]	Cutout [3]	CutMix	
Image					Black pixels or random noise: information loss and inefficiencies during training CutMix: Patches are cut and pasted among training images where the ground truth labels are also mixed proportionally to the area of the patches $x \in \mathbb{R}^{W \times H \times C}$ → training image $y \rightarrow$ corresponding label
Label	Dog 1.0	Dog 0.5 Cat 0.5	Dog 1.0	Dog 0.6 Cat 0.4	Generate a new training example $(\tilde{x}, \tilde{y})$ by combining two training samples $(x_A, y_A)$ and $(x_B, y_B)$ . $\tilde{x} = M \odot x_A + (1 - M) \odot x_B$ $\tilde{y} = \lambda y_A + (1 - \lambda) y_B$
ImageNet Cls (%)	76.3 (+0.0)	77.4 (+1.1)	77.1 (+0.8)	<b>78.6</b> <b>(+2.3)</b>	$M \in \{0, 1\}^{W \times H}$ → a binary mask indicating where to drop out and fill in from two images $\lambda \sim \text{Beta}(\alpha, \alpha)$ → similar to mixup
ImageNet Loc (%)	46.3 (+0.0)	45.8 (-0.5)	46.7 (+0.4)	<b>47.3</b> <b>(+1.0)</b>	$\alpha = 1 \implies \lambda \sim \text{Unif}(0, 1)$
Pascal VOC Det (mAP)	75.6 (+0.0)	73.9 (-1.7)	75.1 (-0.5)	<b>76.7</b> <b>(+1.1)</b>	$B = (r_x, r_y, r_w, r_h)$ → bounding box coordinates

$r_x \sim \text{Unif}(0, W), r_y \sim \text{Unif}(0, H), r_w = W\sqrt{1 - \lambda}, r_h = H\sqrt{1 - \lambda}$   
 $\frac{r_w r_h}{WH} = 1 - \lambda \rightarrow$  cropped area ratio  
 $M \rightarrow$  filling zeros within the bounding box  $B$  and ones otherwise

	Mixup	Cutout	CutMix
Usage of full image region	✓	✗	✓
Regional dropout	✗	✓	✓
Mixed image & label	✓	✗	✓





Boulder

# Neural Ordinary Differential Equations

$h_{t+1} = h_t + f_\theta(h_t) \rightarrow$  residual networks, recurrent neural networks, and normalizing flows (density estimation)

Euler discretization of a continuous transformation!

$$\dot{h} = f_\theta(h, t)$$

$h(0) \rightarrow$  input backpropagation through ODE solvers

$h(T) \rightarrow$  output (black-box differential equation solver)

**Reverse-mode automatic differentiation of ODE solutions**

Adjoint sensitivity method

$L \rightarrow$  scalar-valued loss function

$h(t) \rightarrow$  hidden state at time  $t$

$$a(t) := \frac{\partial L}{\partial h(t)} \rightarrow \text{adjoint}$$

$$\dot{a} = -a^T \frac{\partial f_\theta(h, t)}{\partial h}$$

$$a(T) = \frac{\partial L}{\partial h(T)} \rightarrow \text{known}$$

$$a(0) = \frac{\partial L}{\partial h(0)} \rightarrow \text{black-box differential equation solver}$$

Constant memory cost as a function of depth!

Recompute  $h(t)$  backward in time together with its adjoint  $a(t)$ , starting from  $h(T)$  and  $a(T)$ .

$$\frac{\partial L}{\partial \theta} = - \int_T^0 a(t)^T \frac{\partial f_\theta(h(t), t)}{\partial \theta} dt \rightarrow \text{ODE Solve}$$

$$a^T \frac{\partial f_\theta(h, t)}{\partial h}, a^T \frac{\partial f_\theta(h, t)}{\partial \theta} \rightarrow \text{vector-Jacobian product (standard automatic differentiations)}$$

Chen, Ricky TQ, et al. "Neural ordinary differential equations." *arXiv preprint arXiv:1806.07366* (2018).

**Continuous Depth Residual Networks**

Replace the 6 standard residual blocks in ResNet with ODESolve modules!

	Test Error	# Params	Memory	Time
1-Layer MLP <sup>†</sup>	1.60%	0.24 M	-	-
ResNet	0.41%	0.60 M	$\mathcal{O}(L)$	$\mathcal{O}(L)$
RK-Net	0.47%	0.22 M	$\mathcal{O}(\tilde{L})$	$\mathcal{O}(\tilde{L})$
ODE-Net	0.42%	0.22 M	$\mathcal{O}(1)$	$\mathcal{O}(\tilde{L})$

Performance on MNIST.

**Normalizing Flows**

Change of variable theorem

$$z_1 = f(z_0), f \rightarrow \text{bijective} \implies \log p(z_1) = \log p(z_0) - \log \left| \det \frac{\partial f}{\partial z_0} \right|$$

$$z(t+1) = z(t) + uh(w^T z(t) + b) \rightarrow \text{planar normalizing flow}$$

$$\implies \log p(z(t+1)) = \log p(z(t)) - \log \left| 1 + u^T \frac{\partial h}{\partial z} \right|$$

$$\text{KL}(q(x) \parallel p(x)) \rightarrow \text{loss function}$$

target density  
normalizing flow

**Continuous Normalizing Flows**

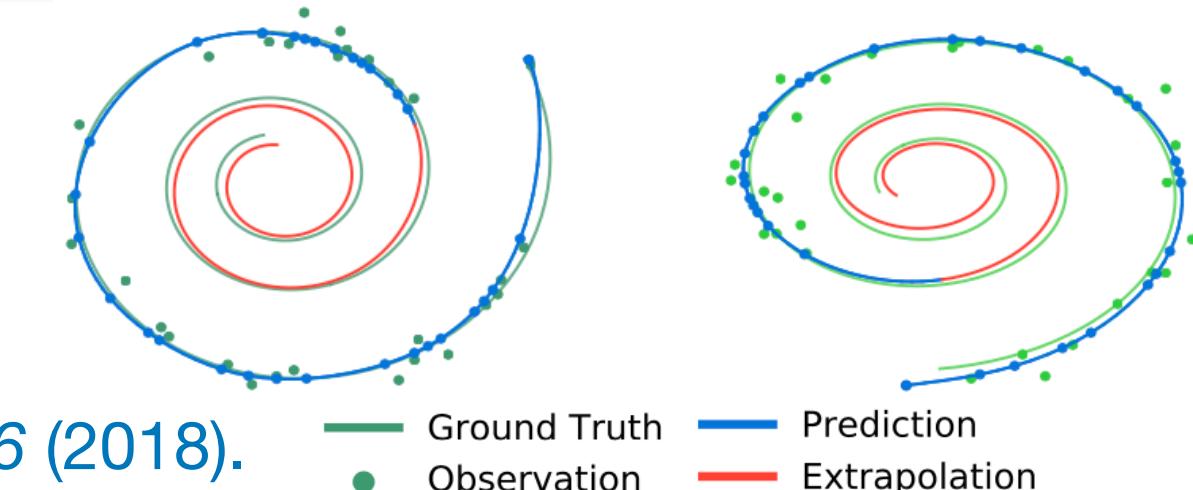
$$\dot{z} = f(z, t) \implies \frac{d \log p(z(t))}{dt} = -\text{tr}\left(\frac{\partial f}{\partial z(t)}\right)$$

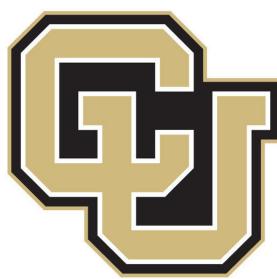
$f$  does not need to be bijective!

**Time Series**  $\mathbf{z}_{t_0} \sim p(\mathbf{z}_{t_0})$

$\mathbf{z}_{t_1}, \mathbf{z}_{t_2}, \dots, \mathbf{z}_{t_N} = \text{ODESolve}(\mathbf{z}_{t_0}, f, \theta_f, t_0, \dots, t_N)$

each  $\mathbf{x}_{t_i} \sim p(\mathbf{x} | \mathbf{z}_{t_i}, \theta_x)$



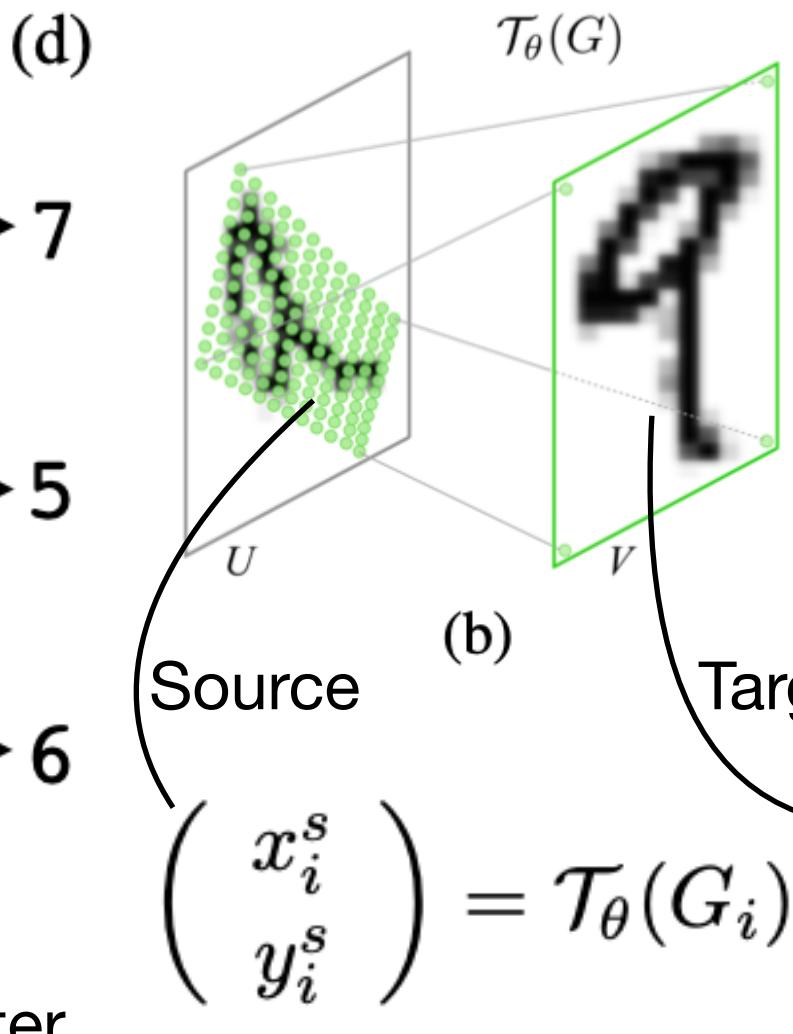
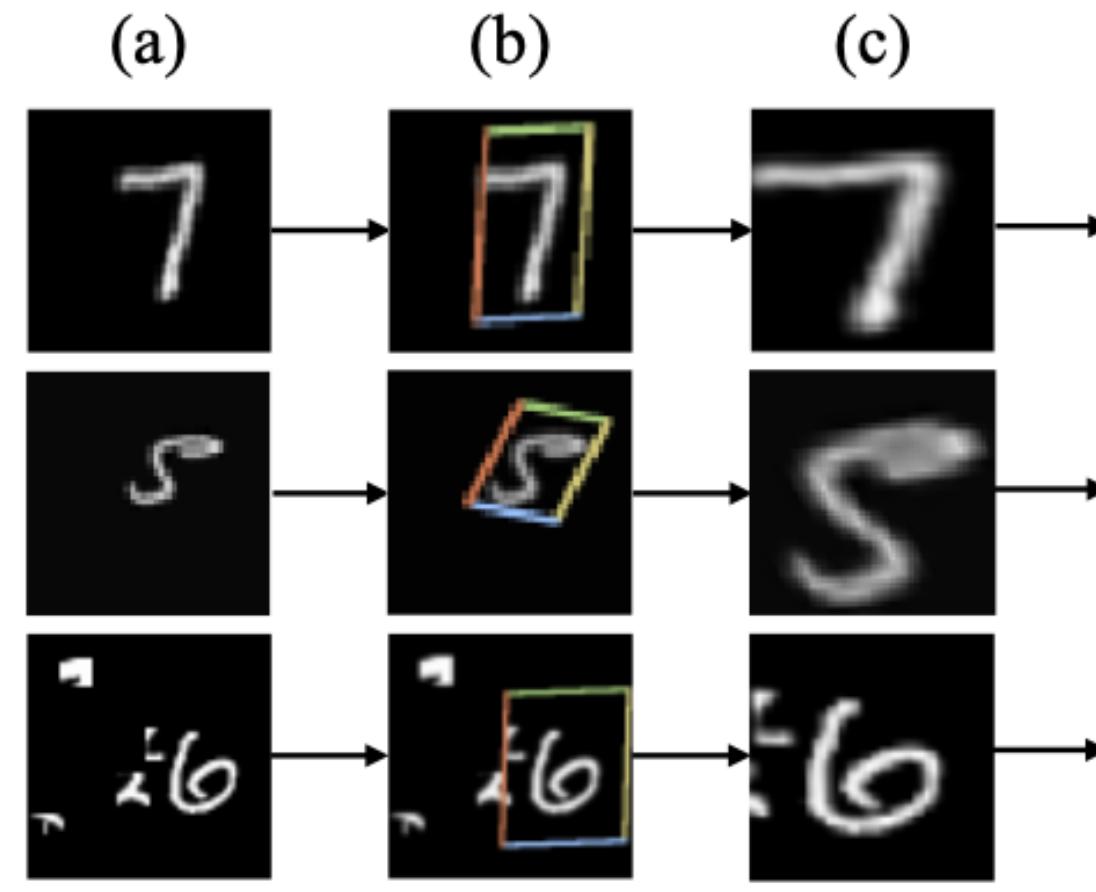


Boulder

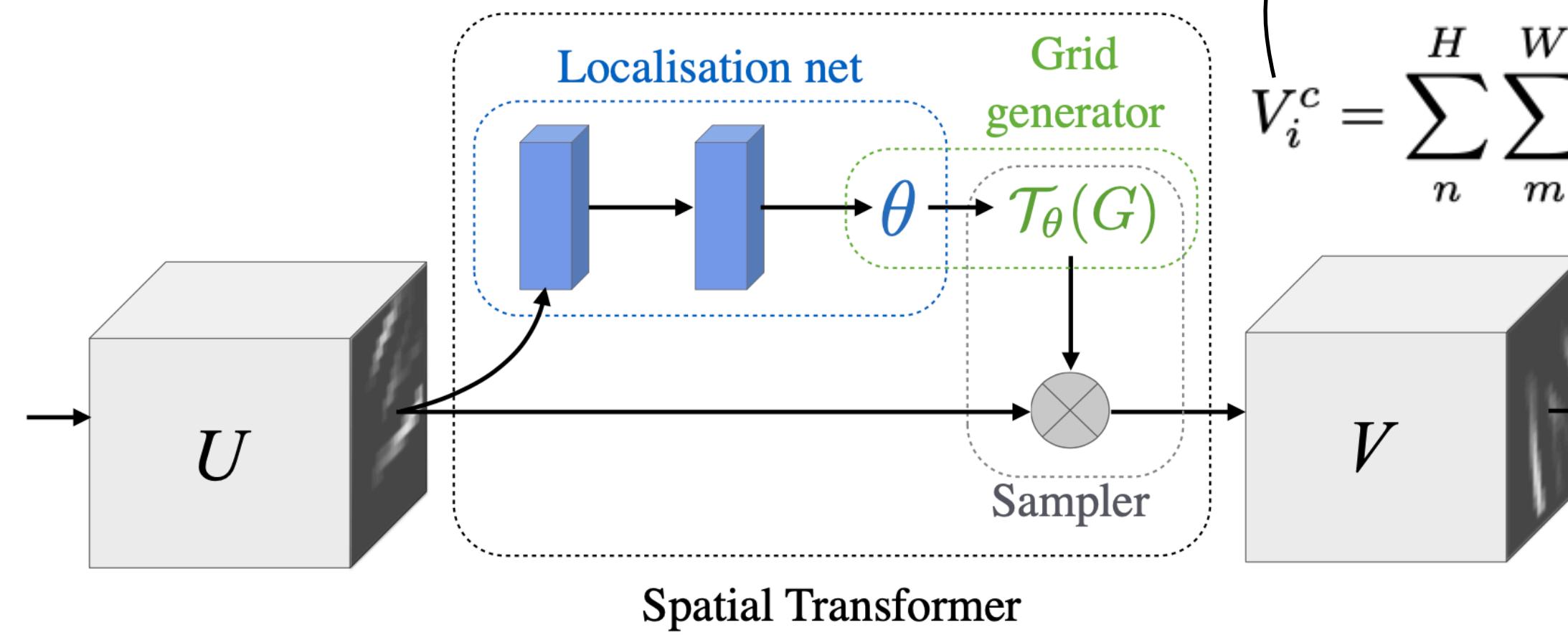


YouTube Video

# Spatial Transformer Networks



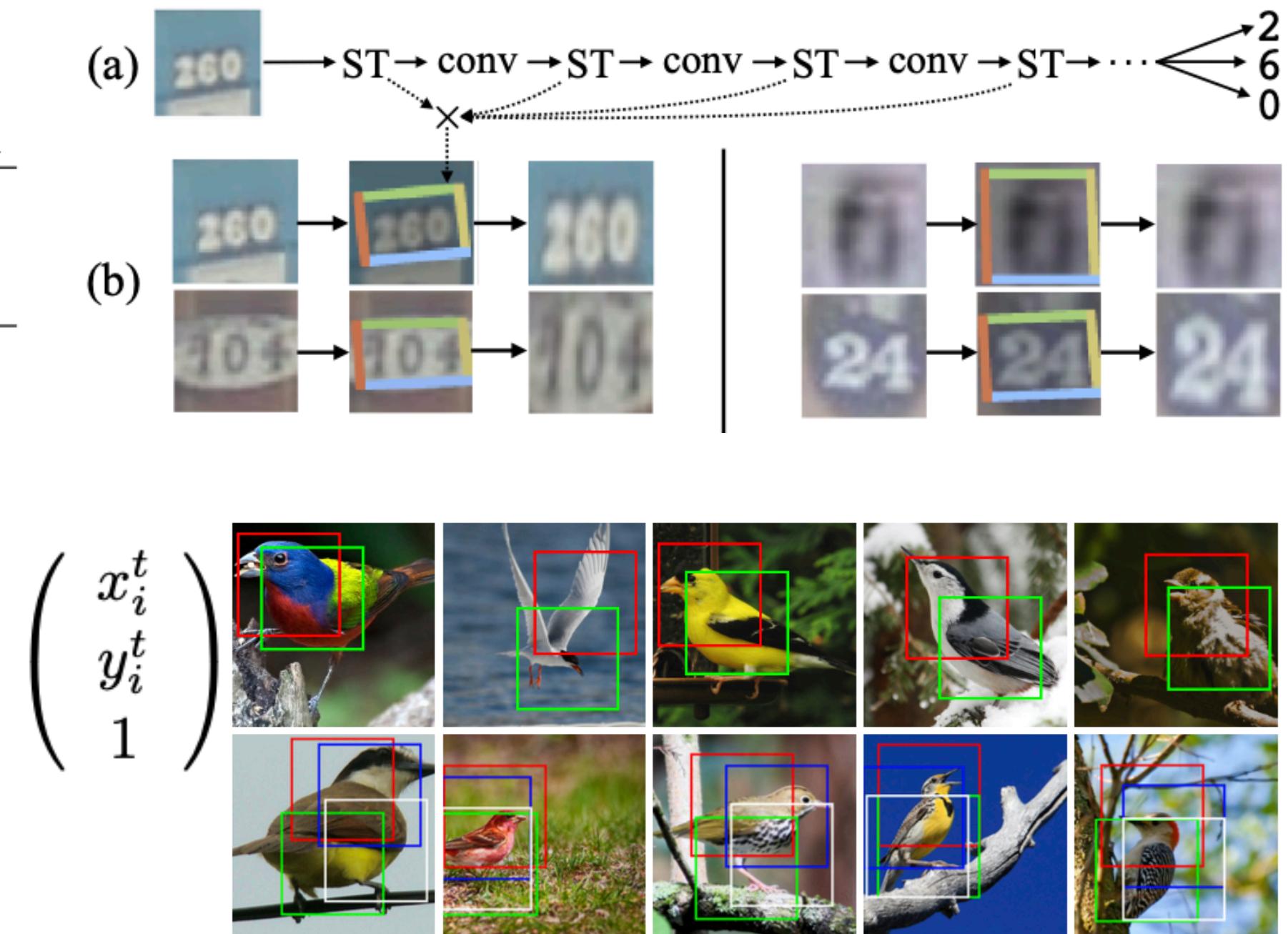
Bilinear sampling



$$\frac{\partial V_i^c}{\partial U_{nm}^c} = \sum_n^H \sum_m^W \max(0, 1 - |x_i^s - m|) \max(0, 1 - |y_i^s - n|)$$

$$\frac{\partial V_i^c}{\partial x_i^s} = \sum_n^H \sum_m^W U_{nm}^c \max(0, 1 - |y_i^s - n|) \begin{cases} 0 & \text{if } |m - x_i^s| \geq 1 \\ 1 & \text{if } m \geq x_i^s \\ -1 & \text{if } m < x_i^s \end{cases}$$

Model	Size	
	64px	128px
Maxout CNN [10]	4.0	-
CNN (ours)	4.0	5.6
DRAM* [1]	3.9	4.5
ST-CNN	Single Multi	3.7 <b>3.6</b> <b>3.9</b>



Model	
Cimpoi '15 [4]	66.7
Zhang '14 [30]	74.9
Branson '14 [2]	75.7
Lin '15 [20]	80.9
Simon '15 [24]	81.0
CNN (ours) 224px	82.3
2×ST-CNN 224px	83.1
2×ST-CNN 448px	83.9
4×ST-CNN 448px	<b>84.1</b>



Boulder



[YouTube Video](#)

# Dynamic Routing Between Capsules

neuron →  $\underbrace{\text{capsule}}$  → layer  
group of neurons

$s_j$  → total input to capsule  $j$

$v_j$  → vector output of capsule  $j$

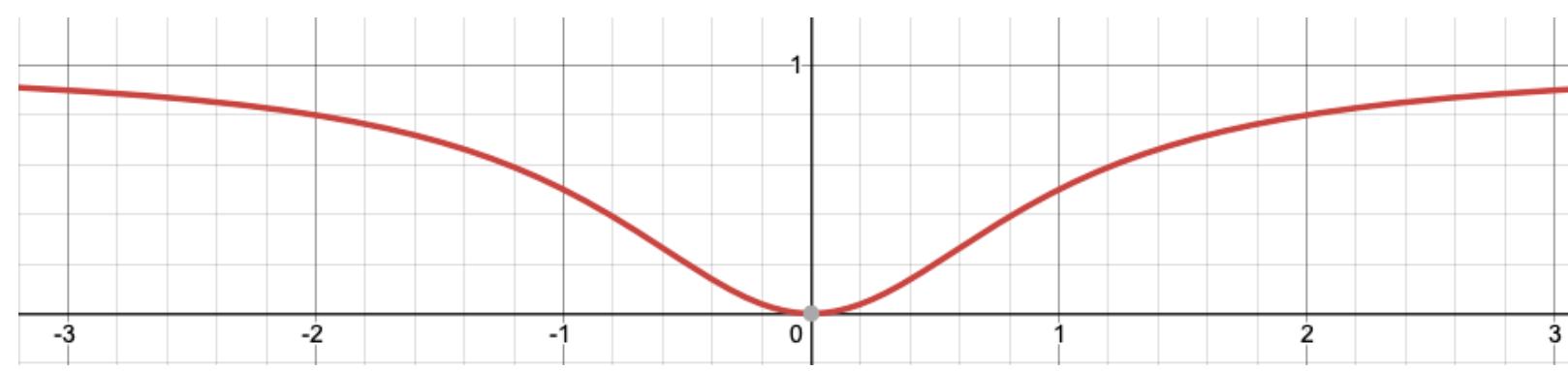
$\|v_j\|$  → probability that the entity represented by the capsule is present in the current input

$\frac{v_j}{\|v_j\|}$  → properties of the entity

entity: an object or an object part

$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \frac{s_j}{\|s_j\|} \leftarrow \text{squash}$$

squashing function



$$s_j = \sum_i c_{ij} u_{j|i}$$

$c_{ij}$  → coupling coefficients

$$u_{j|i} = W_{ij} u_i$$

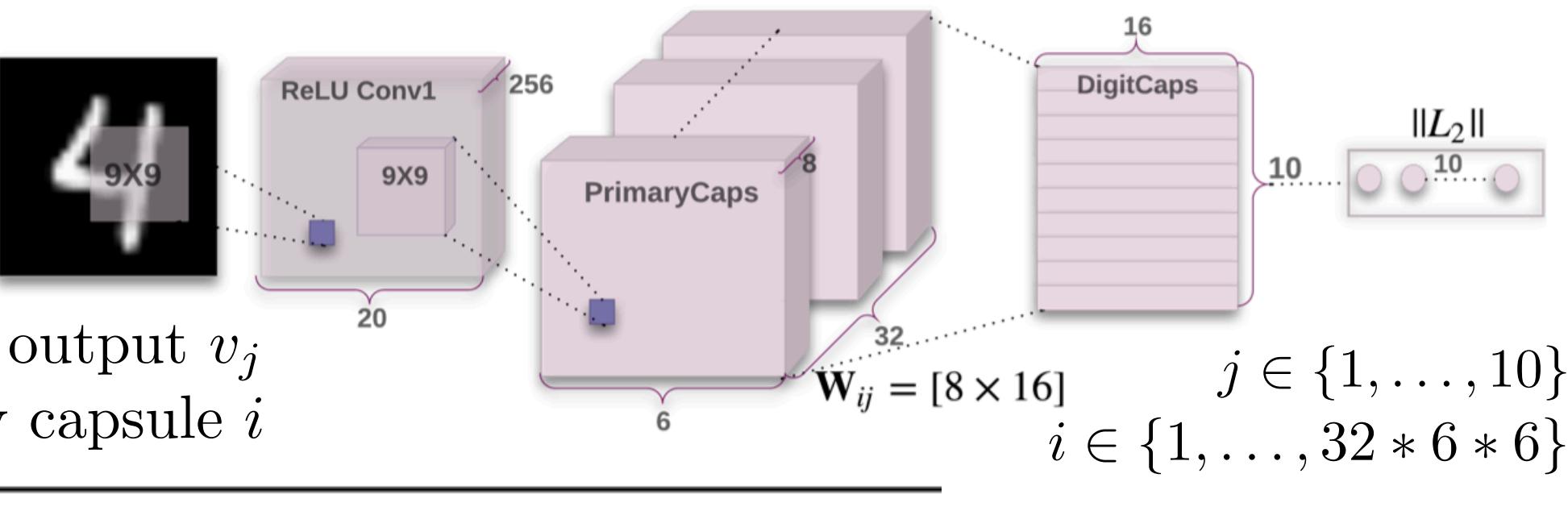
$u_{j|i}$  → prediction vectors from the capsules in the layer below

$u_i$  → output of a capsule in the layer below

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \leftarrow \text{softmax}$$

$b_{ij}$  → log prob. that capsule  $i$  should be coupled to capsule  $j$

$$a_{ij} = v_j \cdot u_{j|i} \rightarrow \text{agreement btw current output } v_j \text{ and the predictions } u_{j|i} \text{ made by capsule } i$$

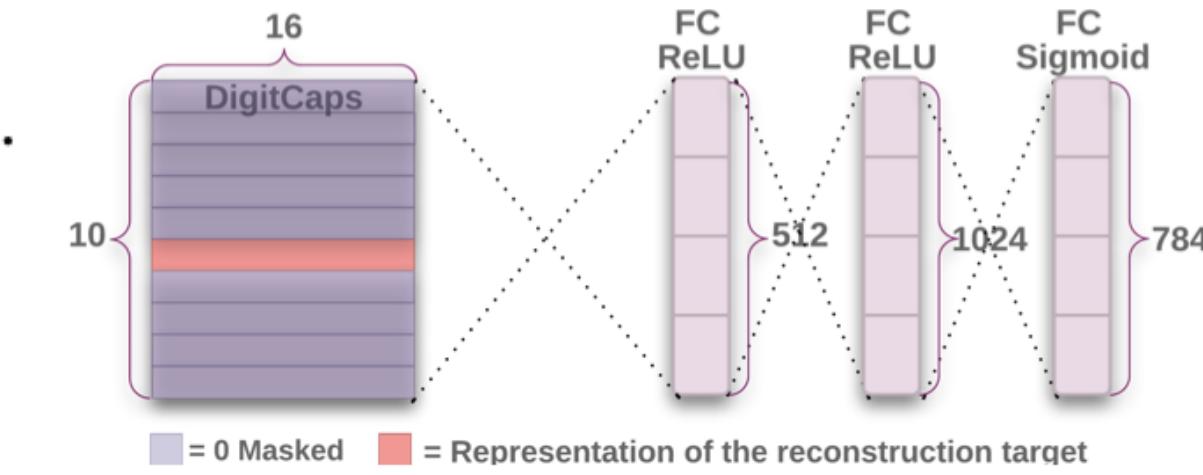


## Procedure 1 Routing algorithm.

```

1: procedure ROUTING( $u_{j|i}, r, l$ )
2:   for all capsule  $i$  in layer  $l$  and capsule  $j$  in layer  $(l + 1)$ :  $b_{ij} \leftarrow 0$ .
3:   for  $r$  iterations do
4:     for all capsule  $i$  in layer  $l$ :  $\mathbf{c}_i \leftarrow \text{softmax}(\mathbf{b}_i)$ 
5:     for all capsule  $j$  in layer  $(l + 1)$ :  $\mathbf{s}_j \leftarrow \sum_i c_{ij} \mathbf{u}_{j|i}$ 
6:     for all capsule  $j$  in layer  $(l + 1)$ :  $\mathbf{v}_j \leftarrow \text{squash}(\mathbf{s}_j)$ 
7:     for all capsule  $i$  in layer  $l$  and capsule  $j$  in layer  $(l + 1)$ :  $b_{ij} \leftarrow b_{ij} + \mathbf{u}_{j|i} \cdot \mathbf{v}_j$ 
return  $\mathbf{v}_j$ 

```



## Margin loss for digit existence

$$L_k = T_k \max(0, m^+ - \|\mathbf{v}_k\|)^2 + \lambda (1 - T_k) \max(0, \|\mathbf{v}_k\| - m^-)^2$$

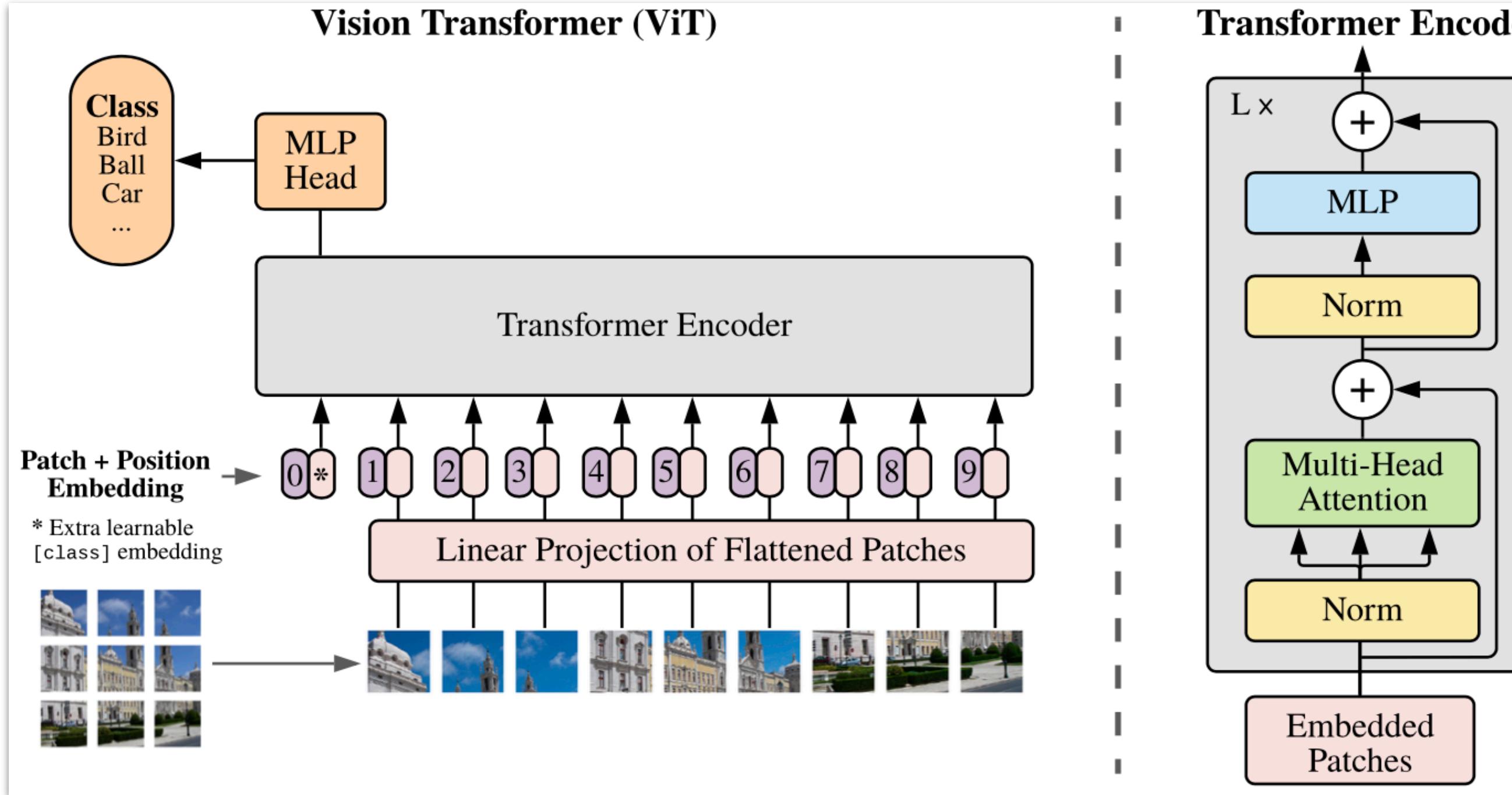
$T_k = 1$  iff a digit of class  $k$  is present

$$m^+ = 0.9 \text{ and } m^- = 0.1 \quad \lambda = 0.5$$

Method	Routing	Reconstruction	MNIST (%)	MultiMNIST (%)
Baseline	-	-	0.39	8.1
CapsNet	1	no	$0.34 \pm 0.032$	-
CapsNet	1	yes	$0.29 \pm 0.011$	7.5
CapsNet	3	no	$0.35 \pm 0.036$	-
CapsNet	3	yes	$0.25 \pm 0.005$	5.2

Scale and thickness	4 4 4 4 6 6 6 6 6 6
Localized part	6 6 6 6 6 6 6 6 6 6
Stroke thickness	5 5 5 5 5 5 5 5 5 5
Localized skew	4 4 4 4 4 4 4 4 4 4
Width and translation	3 3 3 3 3 3 3 3 3 3
Localized part	2 2 2 2 2 2 2 2 2 2

# An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale



Inductive Bias in CNNs: translation equivariance and locality

$\text{ImageNet Top1 Accuracy} [\%]$

Pre-training dataset	BiT	ViT-L/32	ViT-B/32	ViT-L/16	ViT-B/16	ViT-H/14
ImageNet	~81.5%	~71.5%	~76.5%	~73.5%	~77.5%	~78.5%
ImageNet-21k	~80.0%	~81.0%	~84.0%	~85.5%	~84.0%	~88.0%
JFT-300M	~79.0%	~84.0%	~81.0%	~87.0%	~84.0%	~88.5%

Image Patches  $\equiv$  Tokens (Words) in NLP

$x \in \mathbb{R}^{H \times W \times C} \rightarrow \text{image}$

$x_p \in \mathbb{R}^{N \times (P^2 C)} \rightarrow \text{sequence of flattened 2D patches (reshape } x\text{)}$

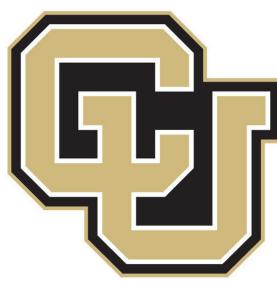
$P \times P \rightarrow \text{resolution of each image patch}$

$N = \frac{HW}{P^2} \rightarrow \text{resulting number of patches}$

$D \rightarrow \text{latent vector size}$

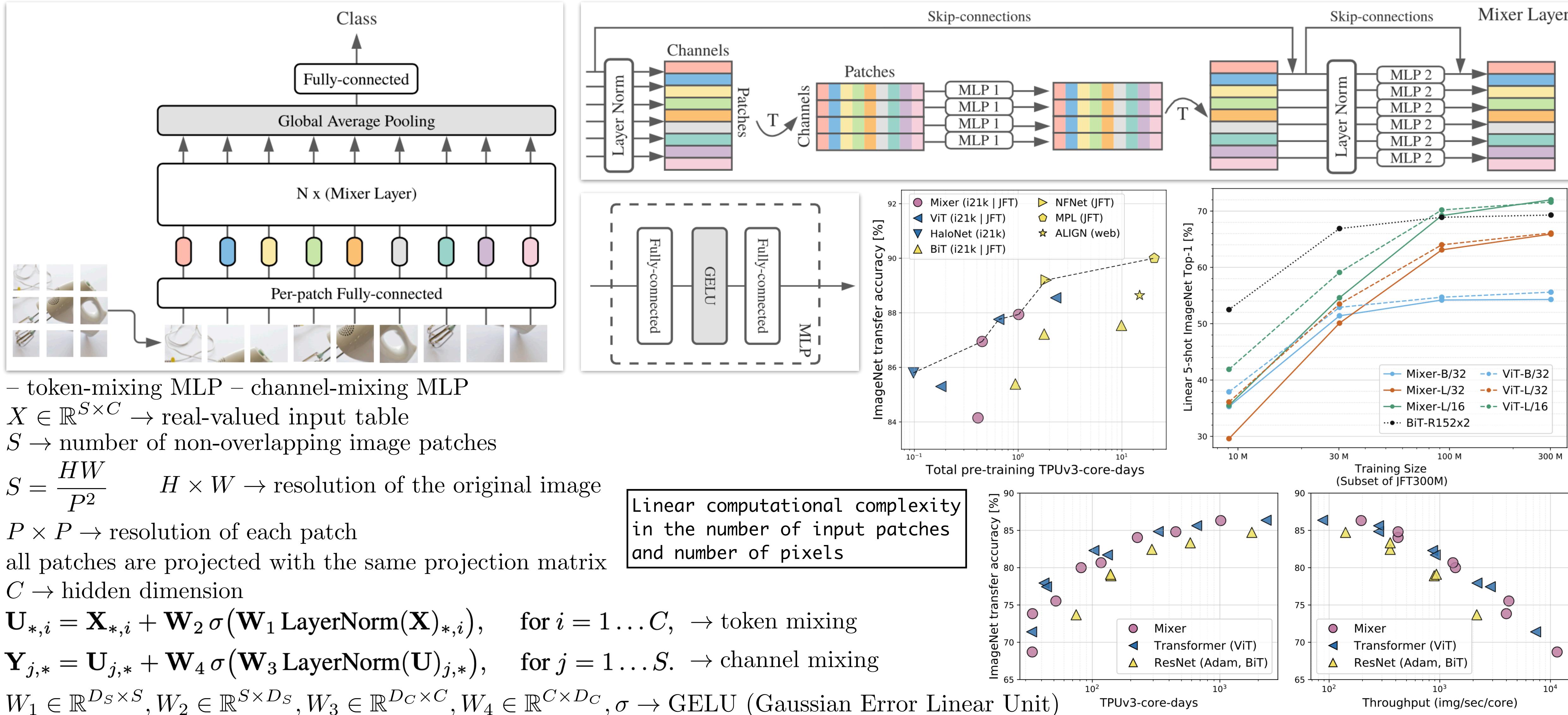
$z_0 = [x_{\text{class}}; x_p^1 E; x_p^2 E; \dots; x_p^N E] + E_{\text{pos}}, E \in \mathbb{R}^{(P^2 C) \times D}, E_{\text{pos}} \in \mathbb{R}^{(N+1) \times D}$

$z'_l = \text{MSA}(\text{LN}(z_{l-1}) + z_{l-1}), l = 1, \dots, L \quad z_l = \text{MLP}(\text{LN}(z'_l)) + z'_l, l = 1, \dots, L \quad y = \text{LN}(z_L^0)$



Boulder

# MLP-Mixer: An all-MLP Architecture for Vision





Boulder

# High-Performance Large-Scale Image Recognition Without Normalization

## Gradient Clipping

$L \rightarrow$  loss

$\theta \rightarrow$  model parameters

$G = \frac{\partial L}{\partial \theta} \rightarrow$  gradient vector

$G \rightarrow \begin{cases} \lambda \frac{G}{\|G\|} & \text{if } \|G\| > \lambda, \\ G & \text{otherwise.} \end{cases}$

$\lambda \rightarrow$  clipping threshold hyper-parameter

## Adaptive Gradient Clipping for Efficient Large-Batch Training

$W^\ell \in \mathbb{R}^{N \times M} \rightarrow$  weight matrix of the  $\ell$ -th layer

$G^\ell \in \mathbb{R}^{N \times M} \rightarrow$  gradient with respect to  $W^\ell$

$\|\cdot\|_F \rightarrow$  Frobenius norm

$$\|W^\ell\|_F = \sum_{i=1}^N \sum_{j=1}^M (W_{ij}^\ell)^2$$

$\frac{\|W^\ell\|_F}{\|G^\ell\|_F}$  provides a simple measure of how much a single gradient step will change the original weights  $W^\ell$

$\Delta W^\ell = -hG^\ell, h \rightarrow$  learning rate

$$\frac{\|\Delta W^\ell\|_F}{\|W^\ell\|_F} = h \frac{\|G^\ell\|_F}{\|W^\ell\|_F}$$

If  $\frac{\|\Delta W^\ell\|_F}{\|W^\ell\|_F}$  is large, we expect the training to become unstable!

$G_i^\ell \rightarrow i\text{-th row of matrix } G$  (unit  $i$  of the  $\ell$ -th layer)

$$G_i^\ell \rightarrow \begin{cases} \lambda \frac{\|W_i^\ell\|_F^*}{\|G_i^\ell\|_F} G_i^\ell & \text{if } \frac{\|G_i^\ell\|_F}{\|W_i^\ell\|_F^*} > \lambda, \\ G_i^\ell & \text{otherwise.} \end{cases}$$

$$\|W_i\|_F^* = \max(\|W_i\|_F, \epsilon), \text{ with default } \epsilon = 10^{-3}$$

## Normalizer-Free ResNets

$$h^{\ell+1} = h^\ell + \alpha f^\ell(h^\ell / \beta^\ell) \rightarrow \text{residual block}$$

$h^\ell \rightarrow$  input to the  $\ell$ -th residual block

$f^\ell \rightarrow$  function computed by the  $\ell$ -th residual branch  
(parametrized to be variance preserving at initialization)

$$\text{Var}(f^\ell(z)) = \text{Var}(z), \forall \ell$$

$\alpha = 0.2 \rightarrow$  rate at which the variance of the activations increases after each residual block (at initialization)

$$\beta^\ell = \sqrt{\text{Var}(h^\ell)}$$

## Scaled Weight Standardization

$$\hat{W}_{ij} = \frac{W_{ij} - \mu_i}{\sqrt{N} \sigma_i}$$

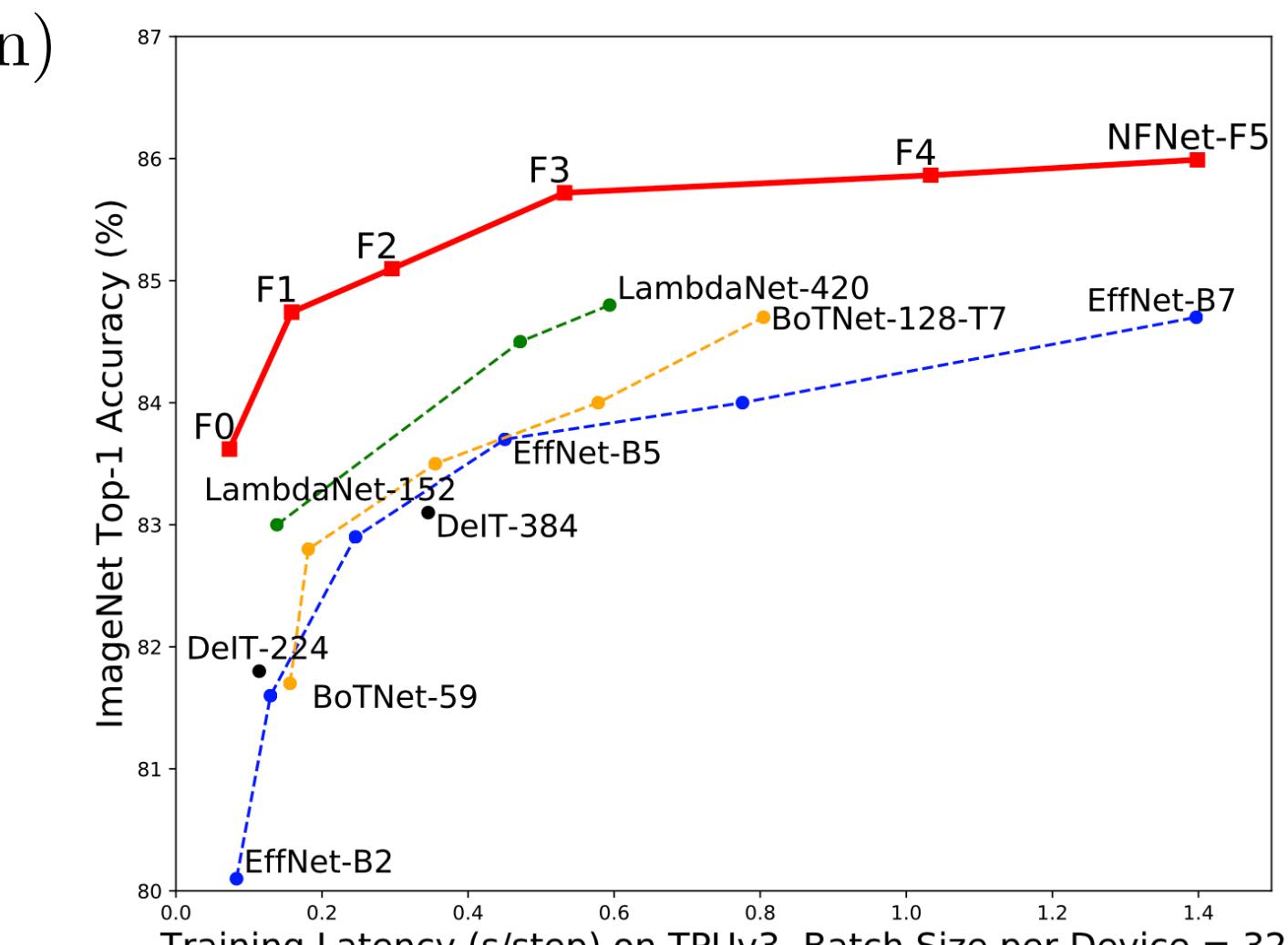
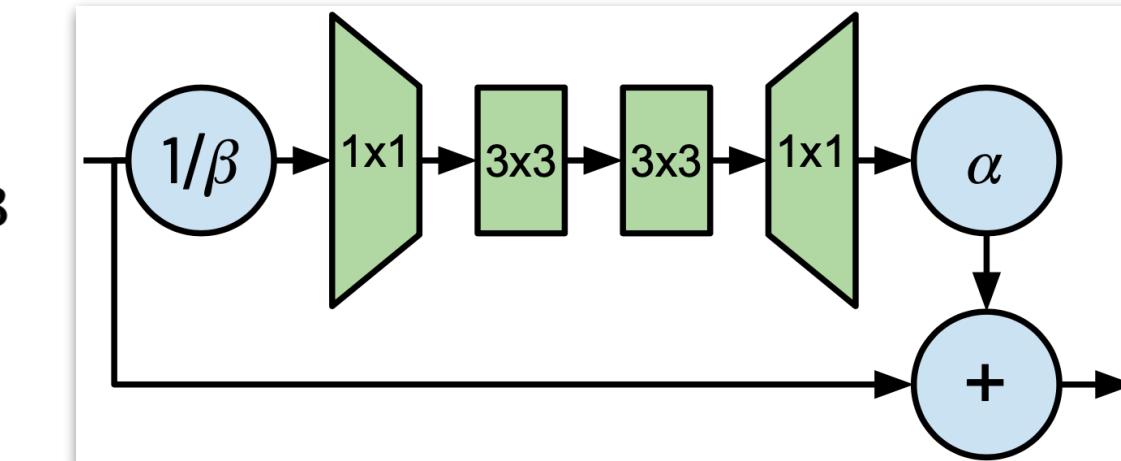
$$\mu_i = (1/N) \sum_j W_{ij}$$

$$\sigma_i^2 = (1/N) \sum_j (W_{ij} - \mu_i)^2$$

The activation functions are also scaled by a non-linearity specific scalar gain  $\gamma$ .

$$\text{For ReLUs, } \gamma = \sqrt{2/(1 - (1/\pi))}$$

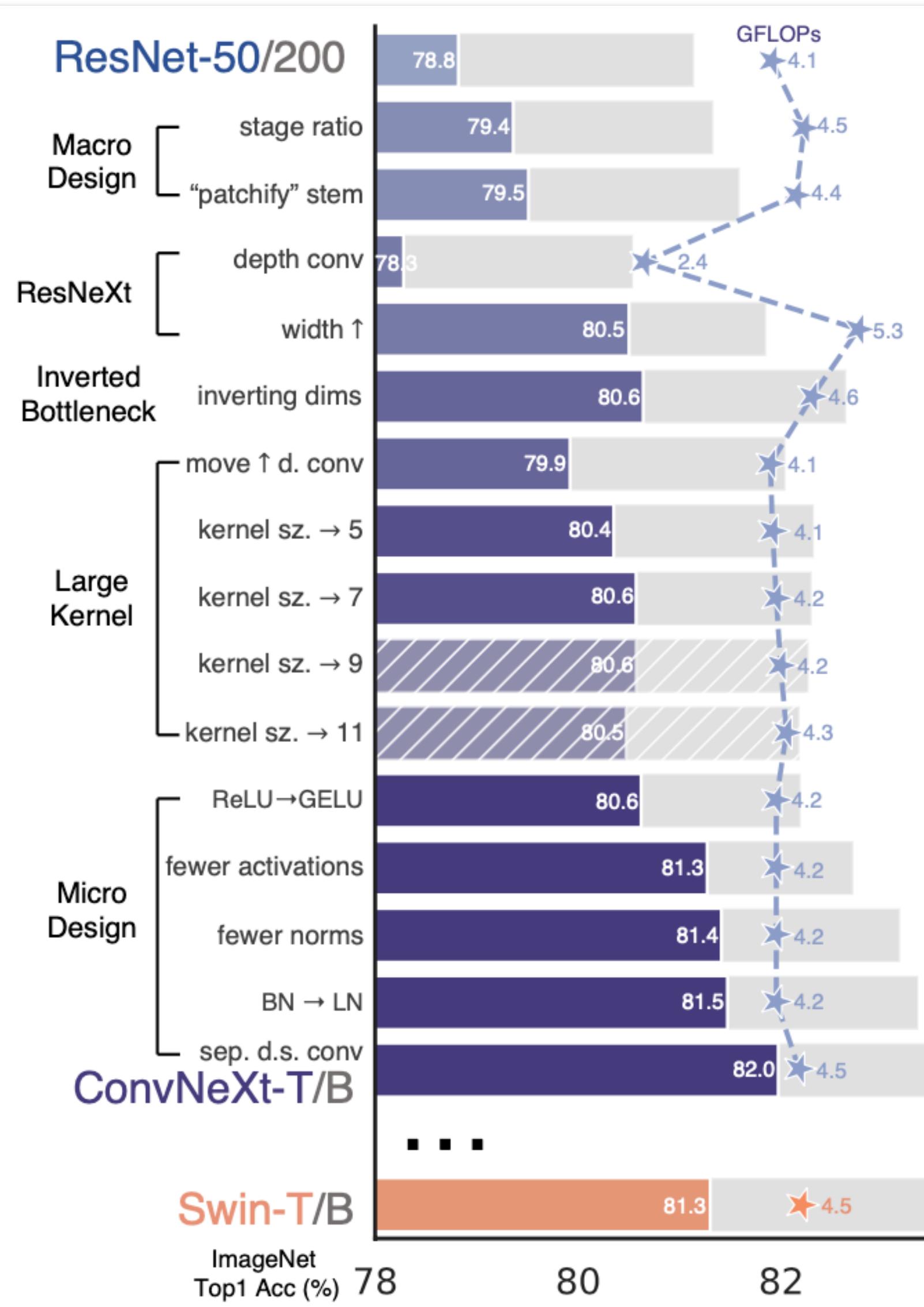
Ensures that the combination of the  $\gamma$ -scaled activation function and a Scaled Weight Standardized layer is variance preserving.



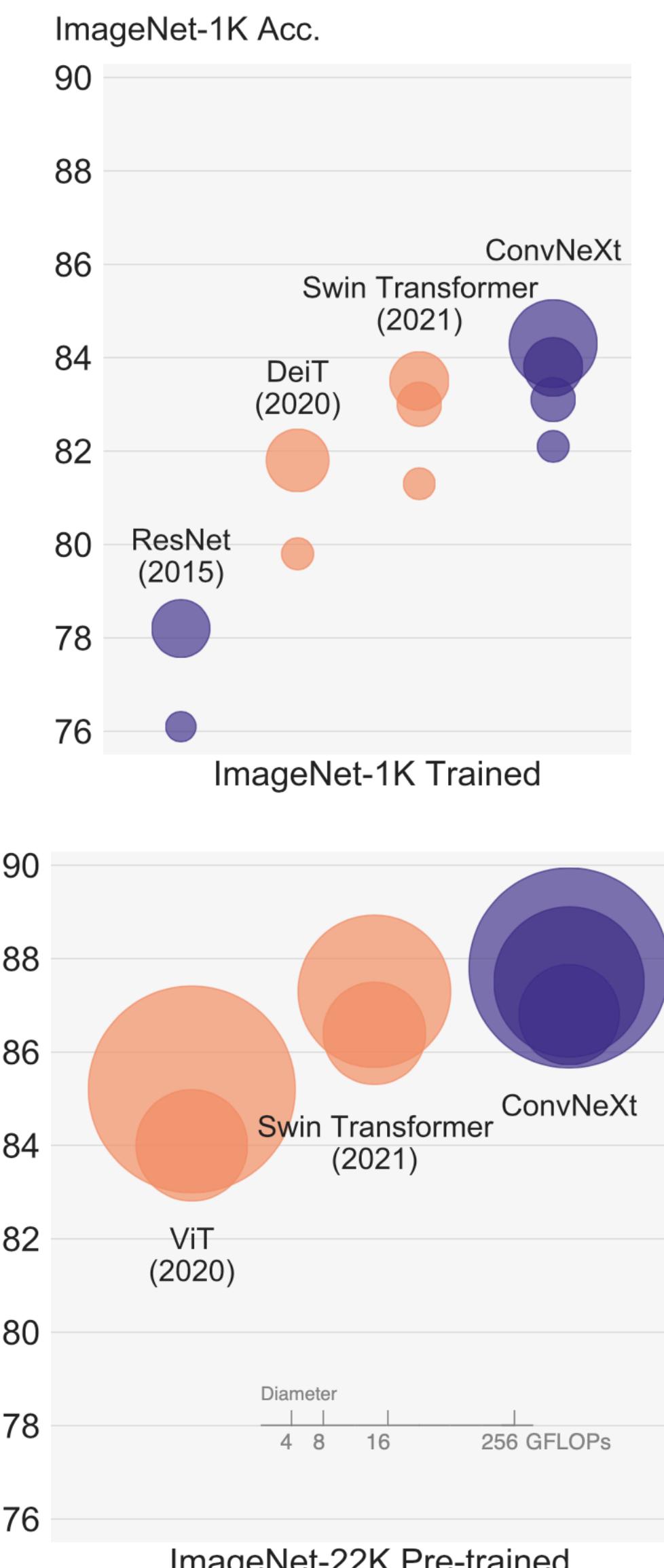


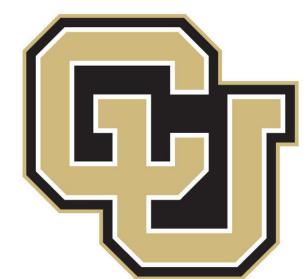
Boulder

# A ConvNet for the 2020s



	output size	• ResNet-50	• ConvNeXt-T	○ Swin-T
stem	56×56	7×7, 64, stride 2 3×3 max pool, stride 2	4×4, 96, stride 4	4×4, 96, stride 4
res2	56×56	$\begin{bmatrix} 1\times 1, 64 \\ 3\times 3, 64 \\ 1\times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} d7\times 7, 96 \\ 1\times 1, 384 \\ 1\times 1, 96 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times 1, 96\times 3 \\ \text{MSA, w7}\times 7, H=3, \text{rel. pos.} \\ 1\times 1, 96 \\ 1\times 1, 384 \\ 1\times 1, 96 \end{bmatrix} \times 2$
res3	28×28	$\begin{bmatrix} 1\times 1, 128 \\ 3\times 3, 128 \\ 1\times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} d7\times 7, 192 \\ 1\times 1, 768 \\ 1\times 1, 192 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times 1, 192\times 3 \\ \text{MSA, w7}\times 7, H=6, \text{rel. pos.} \\ 1\times 1, 192 \\ 1\times 1, 768 \\ 1\times 1, 192 \end{bmatrix} \times 2$
res4	14×14	$\begin{bmatrix} 1\times 1, 256 \\ 3\times 3, 256 \\ 1\times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} d7\times 7, 384 \\ 1\times 1, 1536 \\ 1\times 1, 384 \end{bmatrix} \times 9$	$\begin{bmatrix} 1\times 1, 384\times 3 \\ \text{MSA, w7}\times 7, H=12, \text{rel. pos.} \\ 1\times 1, 384 \\ 1\times 1, 1536 \\ 1\times 1, 384 \end{bmatrix} \times 6$
res5	7×7	$\begin{bmatrix} 1\times 1, 512 \\ 3\times 3, 512 \\ 1\times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} d7\times 7, 768 \\ 1\times 1, 3072 \\ 1\times 1, 768 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times 1, 768\times 3 \\ \text{MSA, w7}\times 7, H=24, \text{rel. pos.} \\ 1\times 1, 768 \\ 1\times 1, 3072 \\ 1\times 1, 768 \end{bmatrix} \times 2$
FLOPs		$4.1 \times 10^9$	$4.5 \times 10^9$	$4.5 \times 10^9$
# params.		$25.6 \times 10^6$	$28.6 \times 10^6$	$28.3 \times 10^6$
(pre-)training config	ConvNeXt-T/S/B/L ImageNet-1K 224 <sup>2</sup>			
optimizer	AdamW		layer-wise lr decay [6, 10]	None
base learning rate	4e-3		randaugment [12]	(9, 0.5)
weight decay	0.05		label smoothing [65]	0.1
optimizer momentum	$\beta_1, \beta_2 = 0.9, 0.999$		mixup [85]	0.8
batch size	4096		cutmix [84]	1.0
training epochs	300		stochastic depth [34]	0.1/0.4/0.5/0.5
learning rate schedule	cosine decay		layer scale [69]	1e-6
warmup epochs	20		gradient clip	None
warmup schedule	linear		exp. mov. avg. (EMA) [48]	0.9999





Boulder



# Questions?

[YouTube Playlist](#)

---