



# Computer Vision; Image Classification; Large Networks



[YouTube Playlist](#)

**Maziar Raissi**

**Assistant Professor**

Department of Applied Mathematics

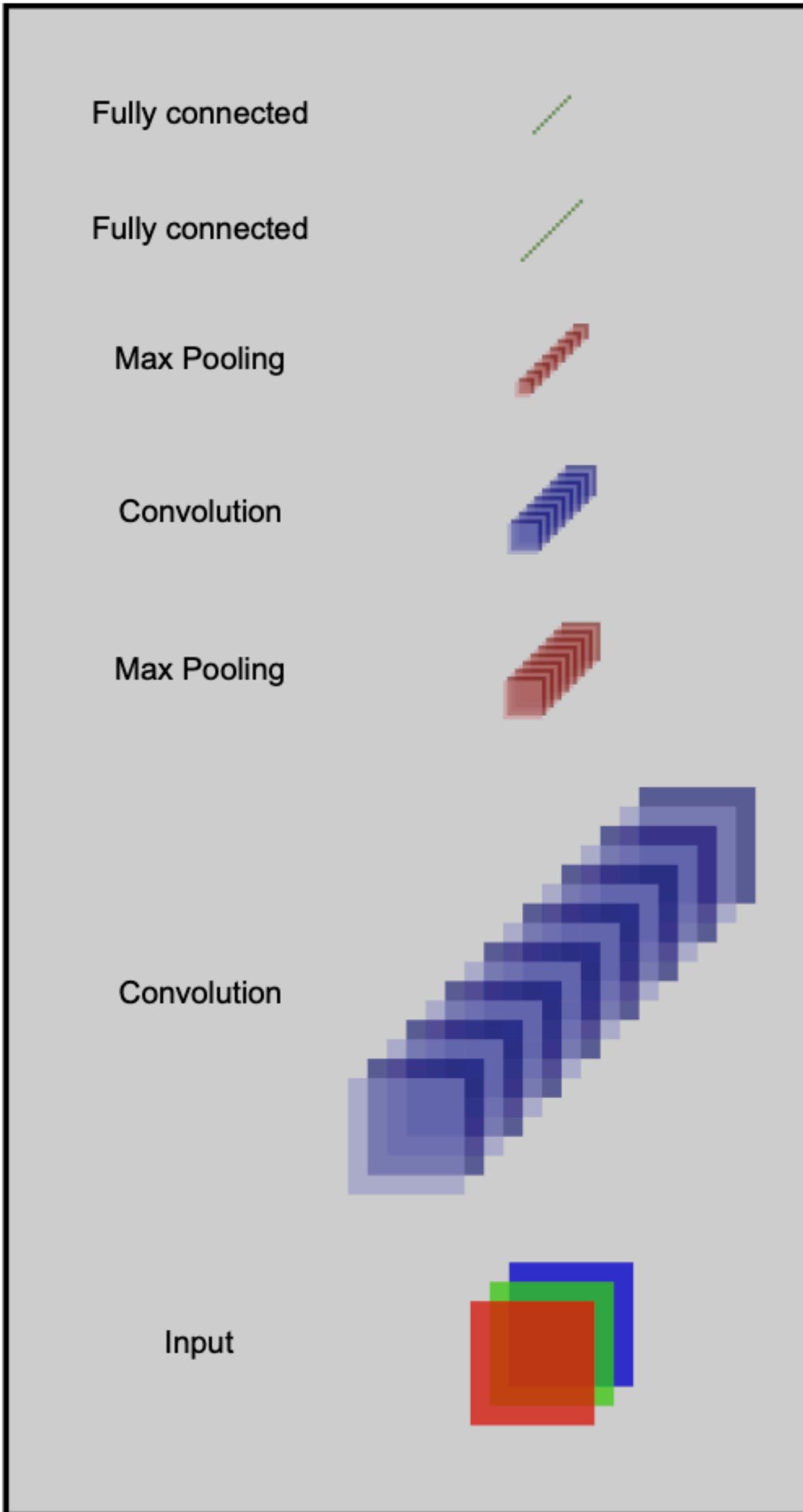
University of Colorado Boulder

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

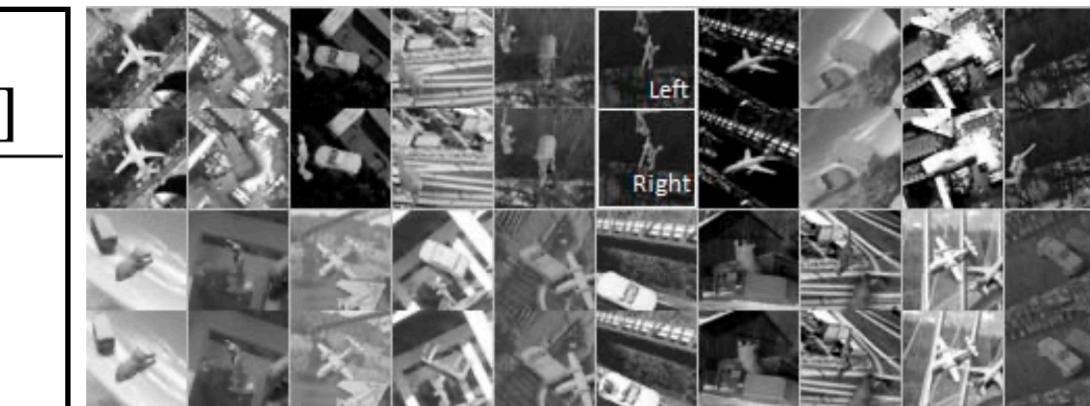


Boulder

# 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

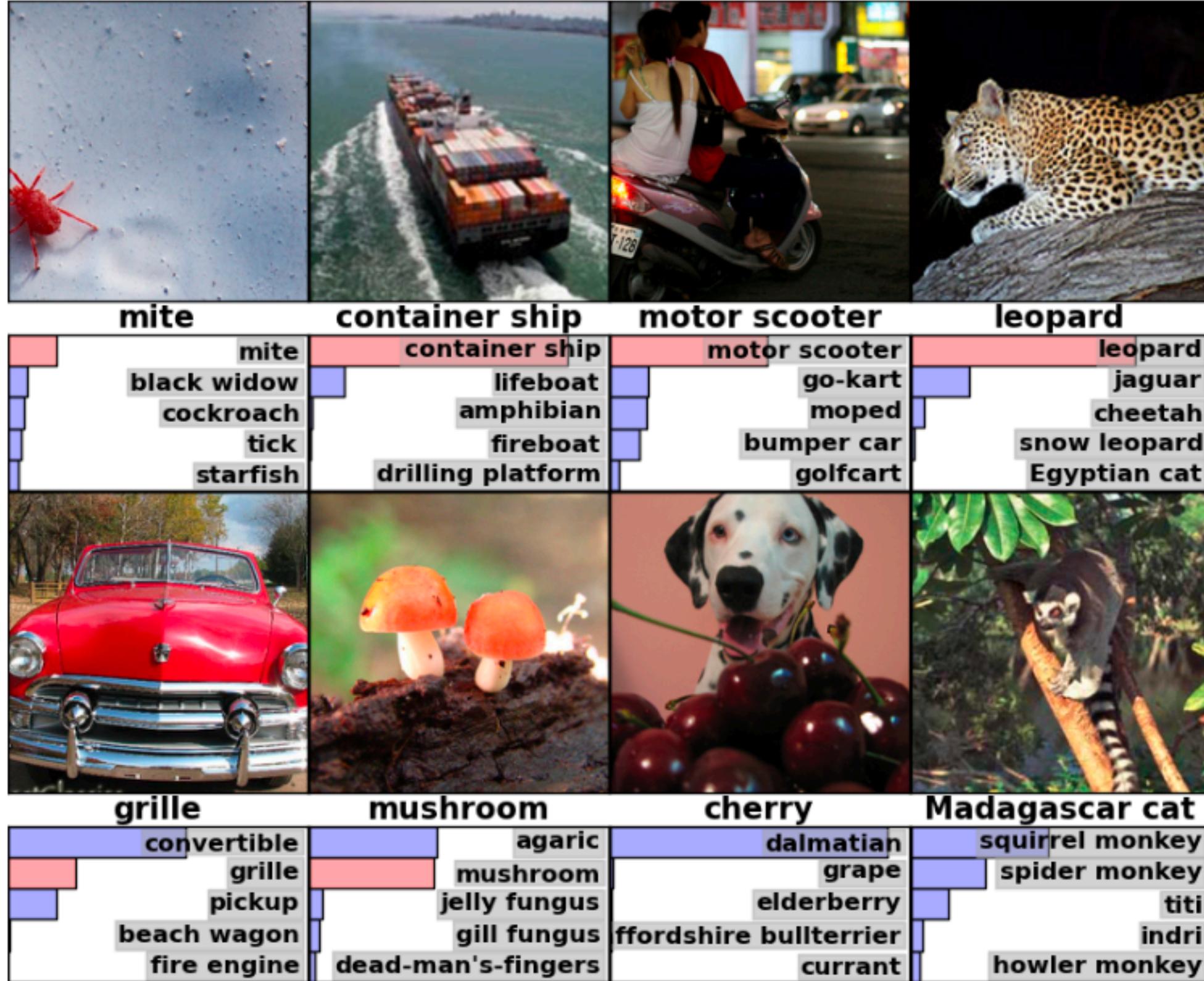


Boulder

# ImageNet Classification with Deep Convolutional Neural Networks

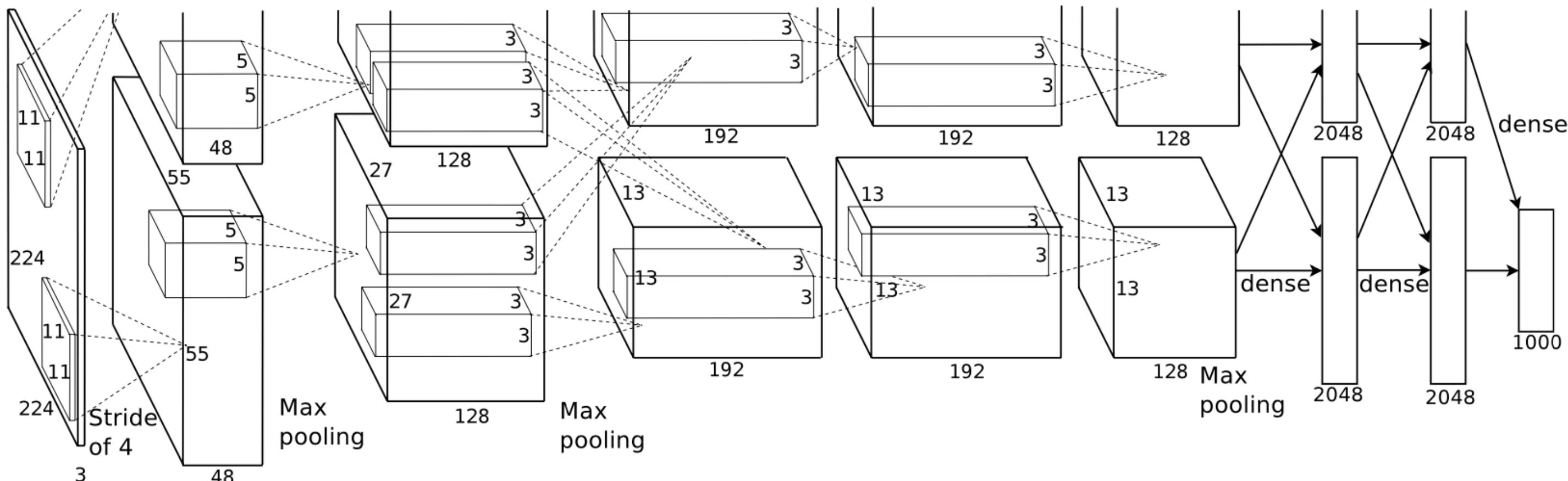


[YouTube Video](#)



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

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

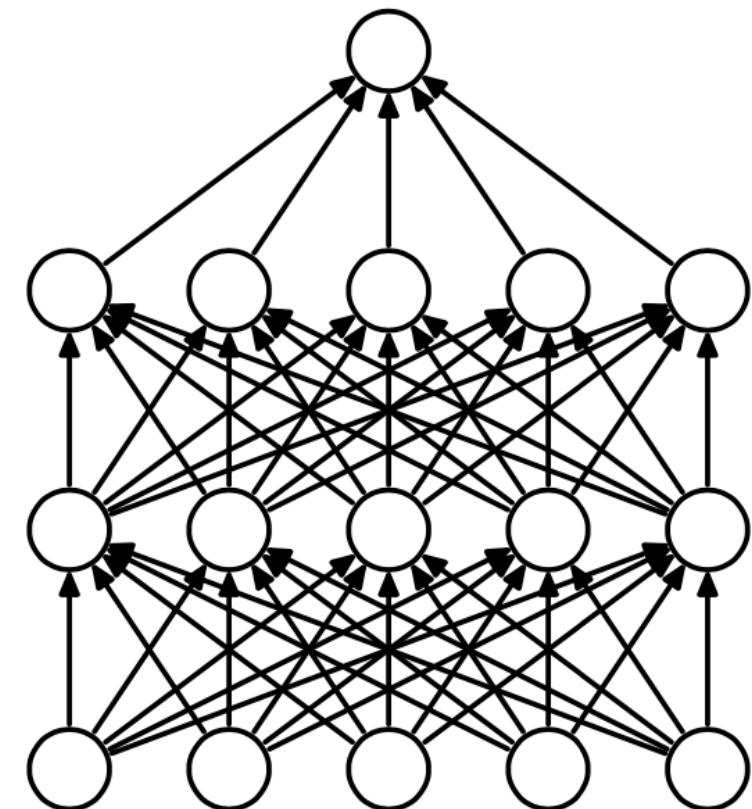
**3x3 Conv:**  $Y(x, y) = \sum_{i=1,2,3} \sum_{j=1,2,3} W(i, j)X(s(x-2)+i, s(y-2)+j), W \in \mathbb{R}^{3 \times 3 \times F \times C}$

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

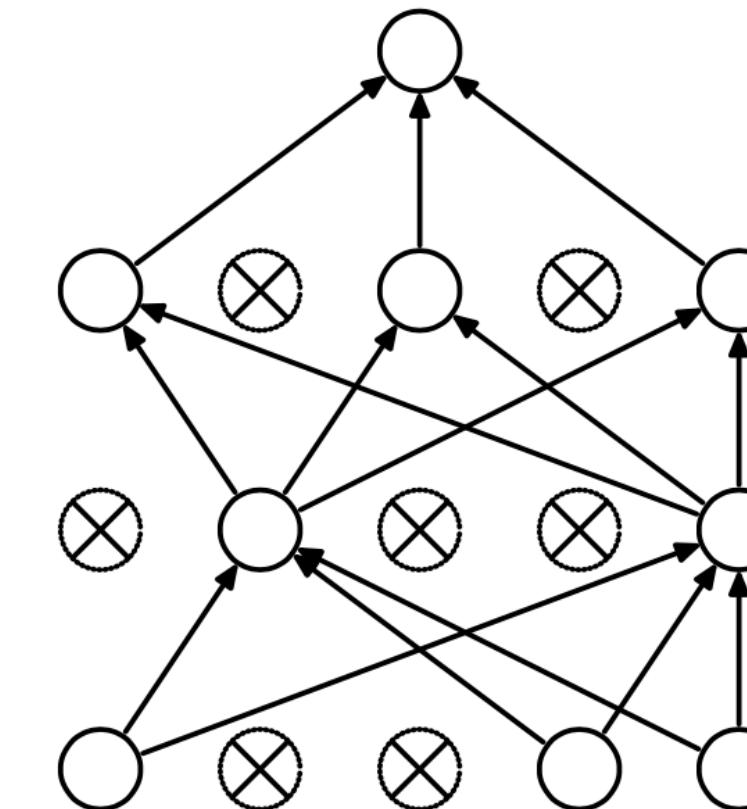
$$\text{Loss: } \mathcal{L}(\theta) = - \sum_{n=1}^N \log(p_{i_n}(I_n))$$

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>

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

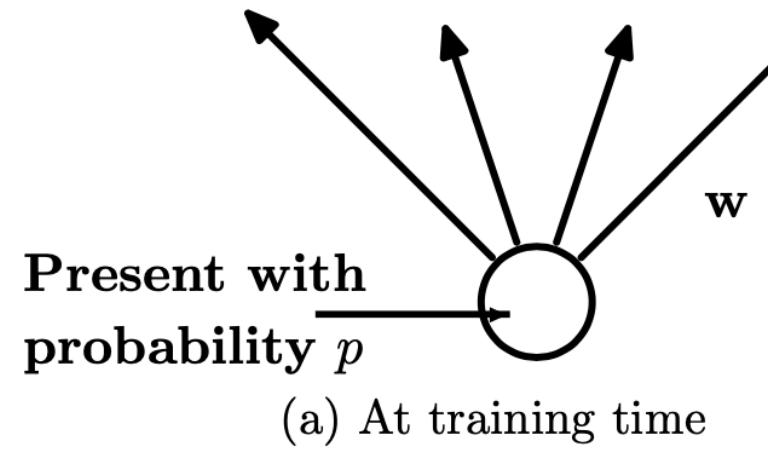

[YouTube Video](#)


(a) Standard Neural Net

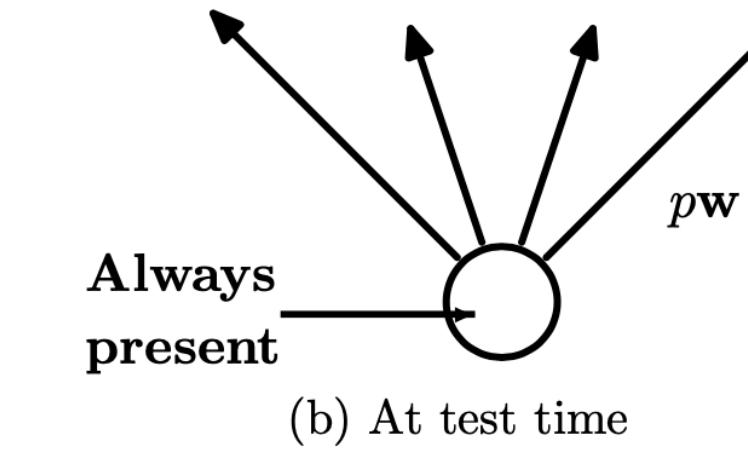


(b) After applying dropout.

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



(a) At training time

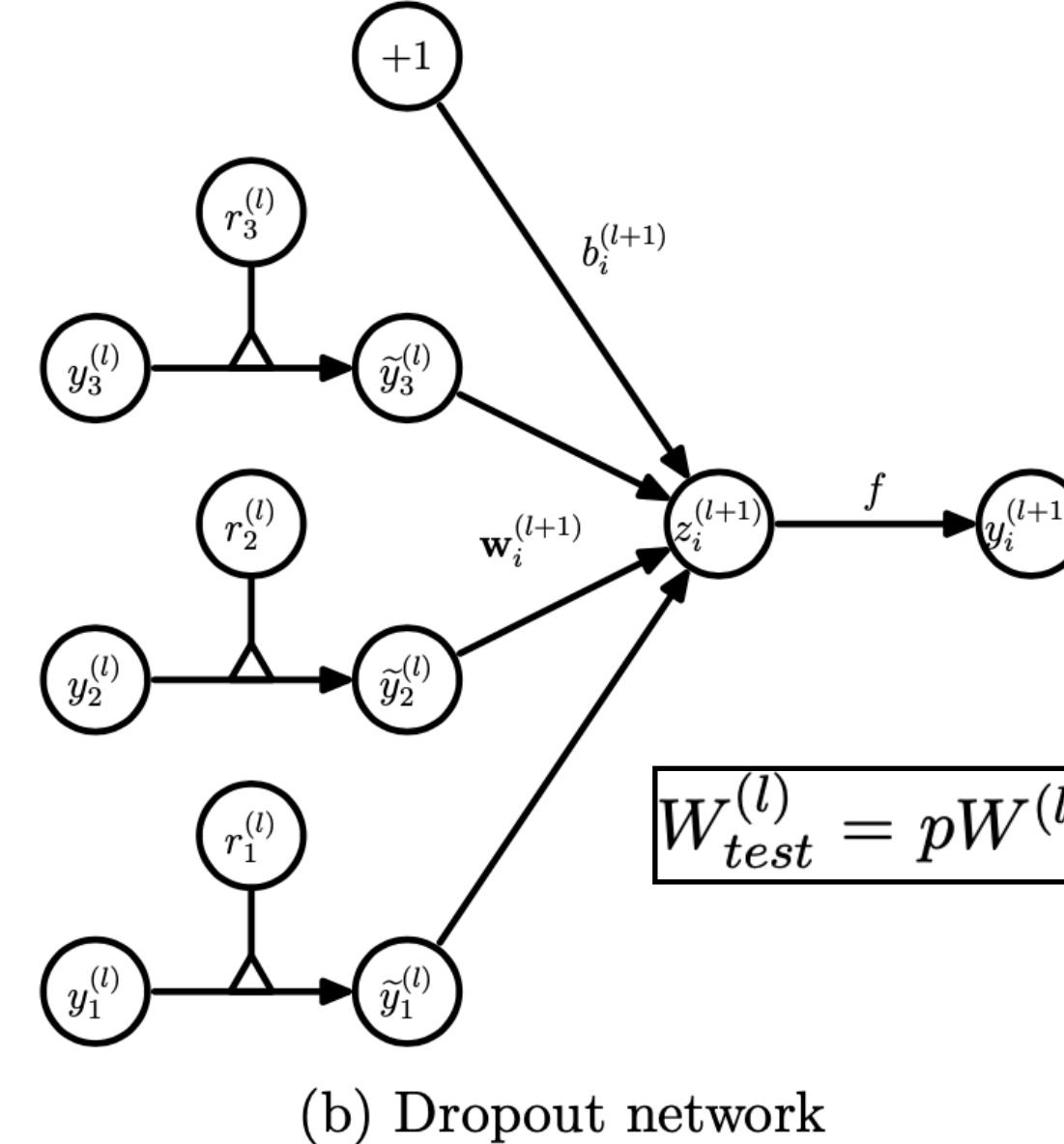
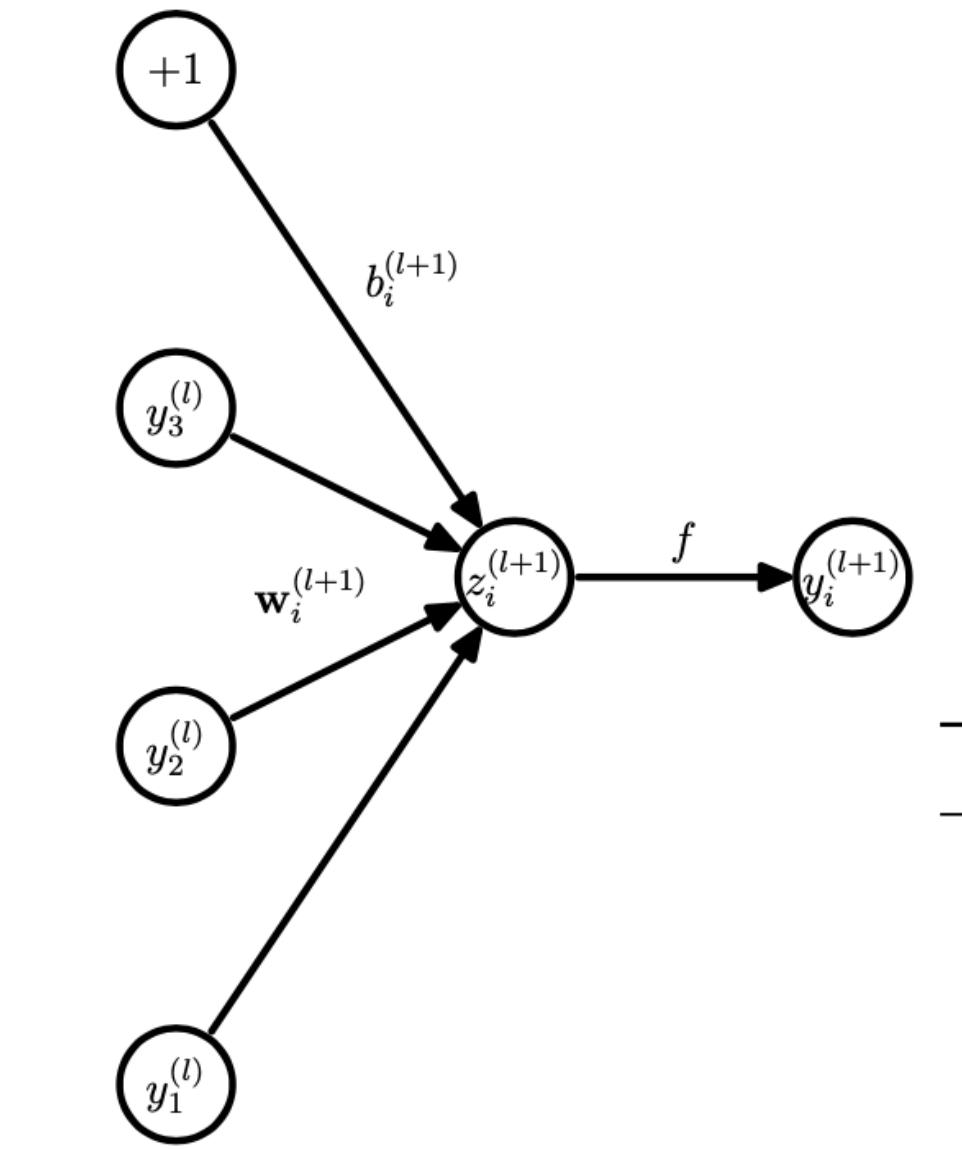


(b) At test time

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

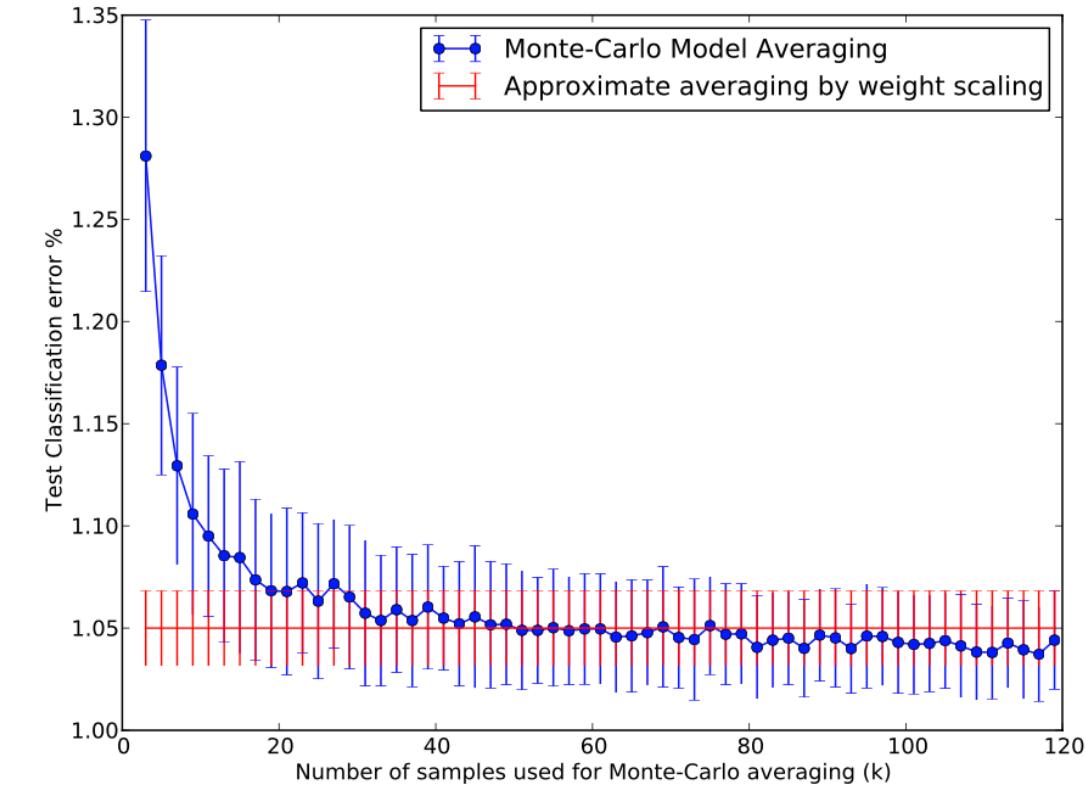
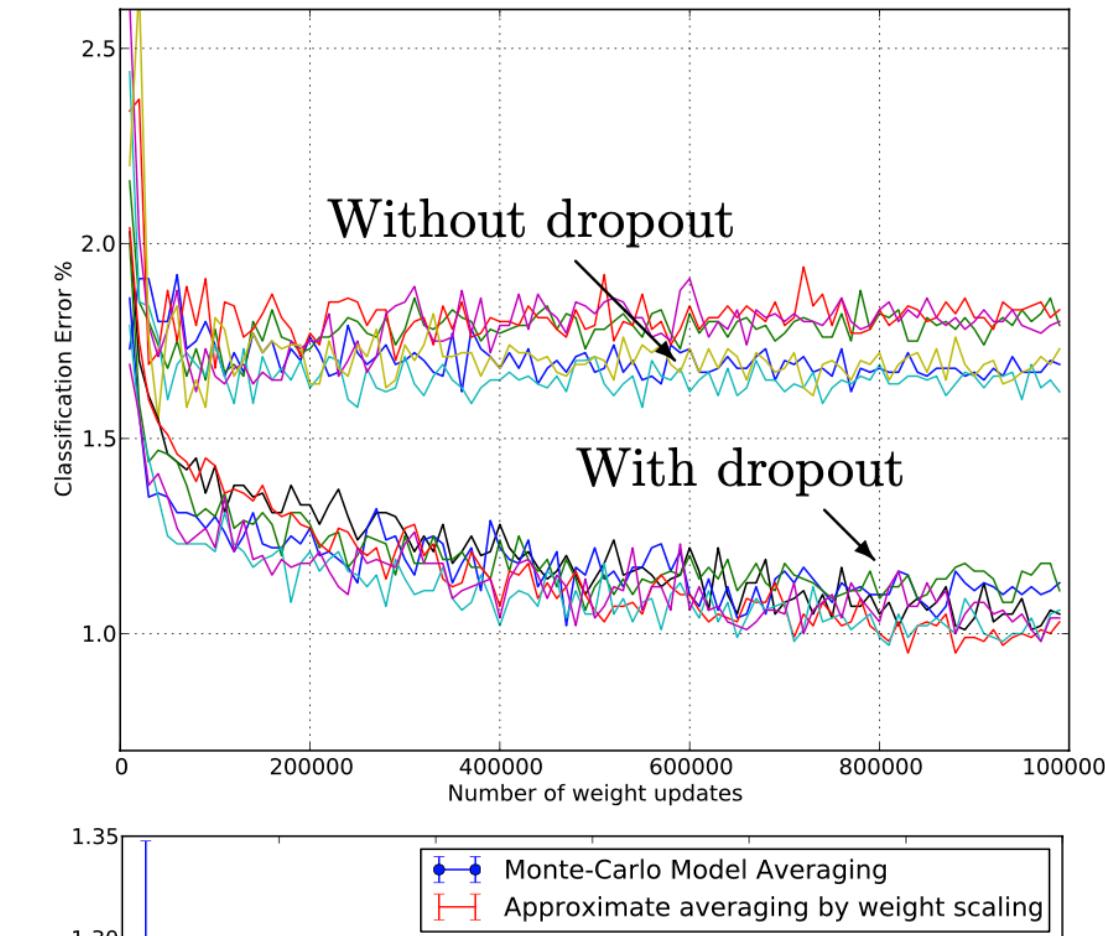
$$l \in \{1, \dots, L\}$$

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



$\|\mathbf{w}\|_2 \leq c$   
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.



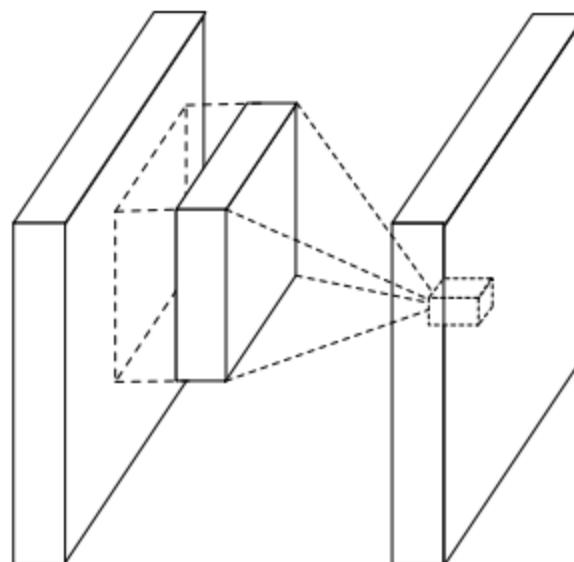
Boulder



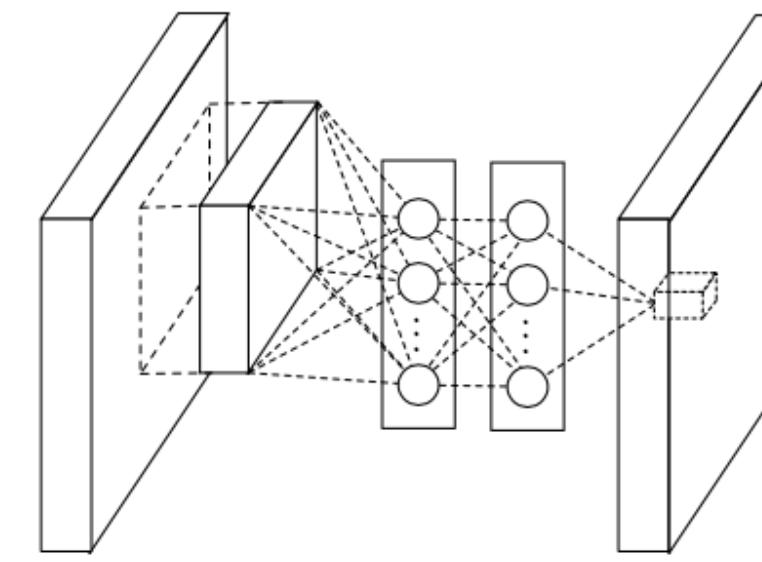
[YouTube Playlist](#)

# 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

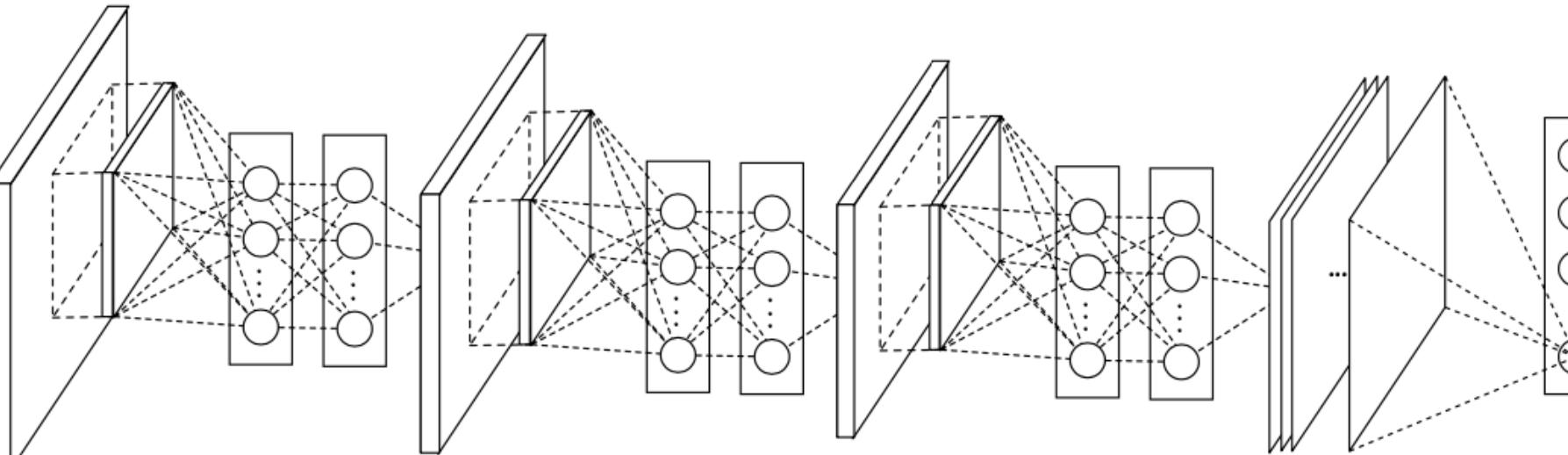


Table 3: Test set error rates for SVHN of various methods.

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.

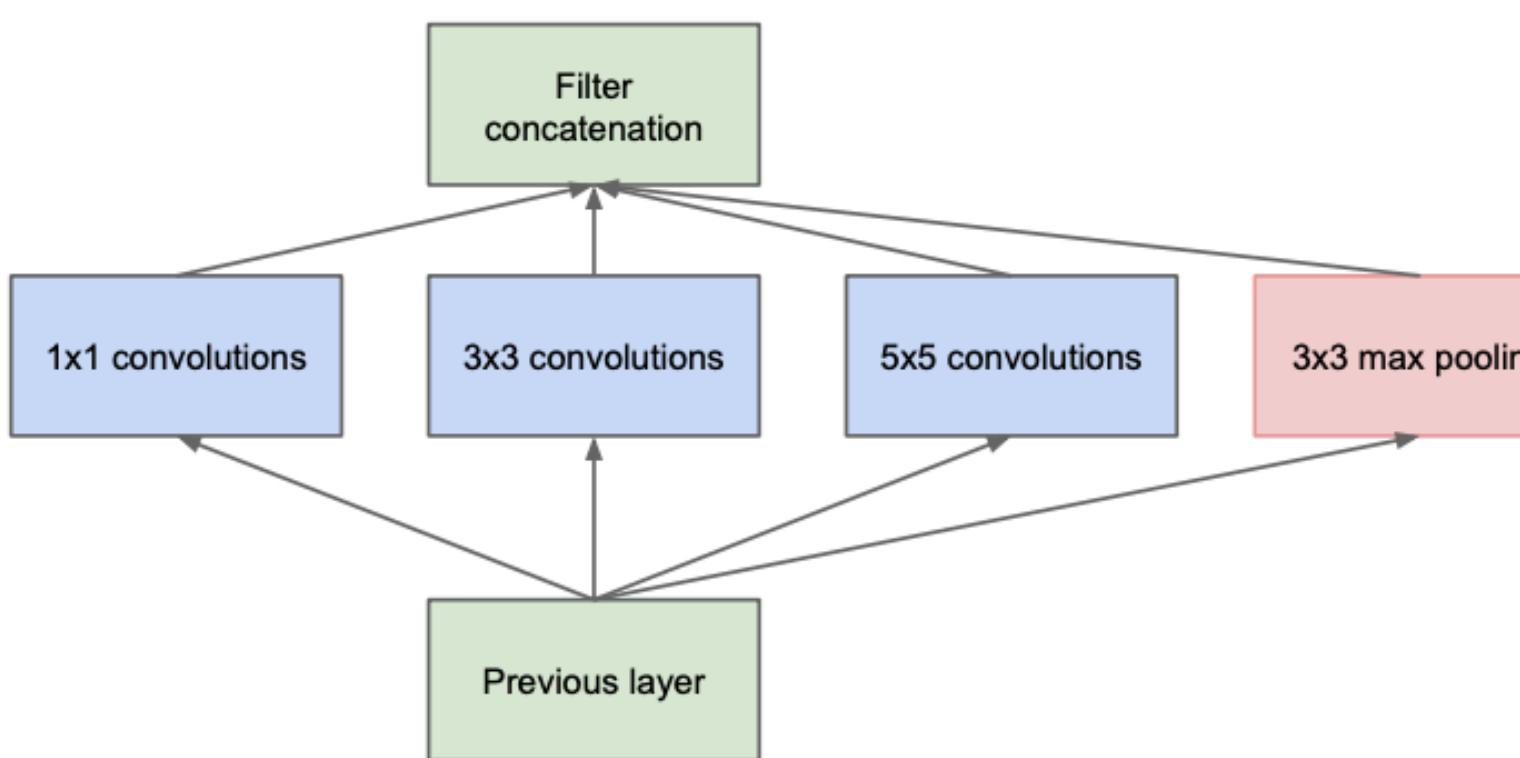


Boulder

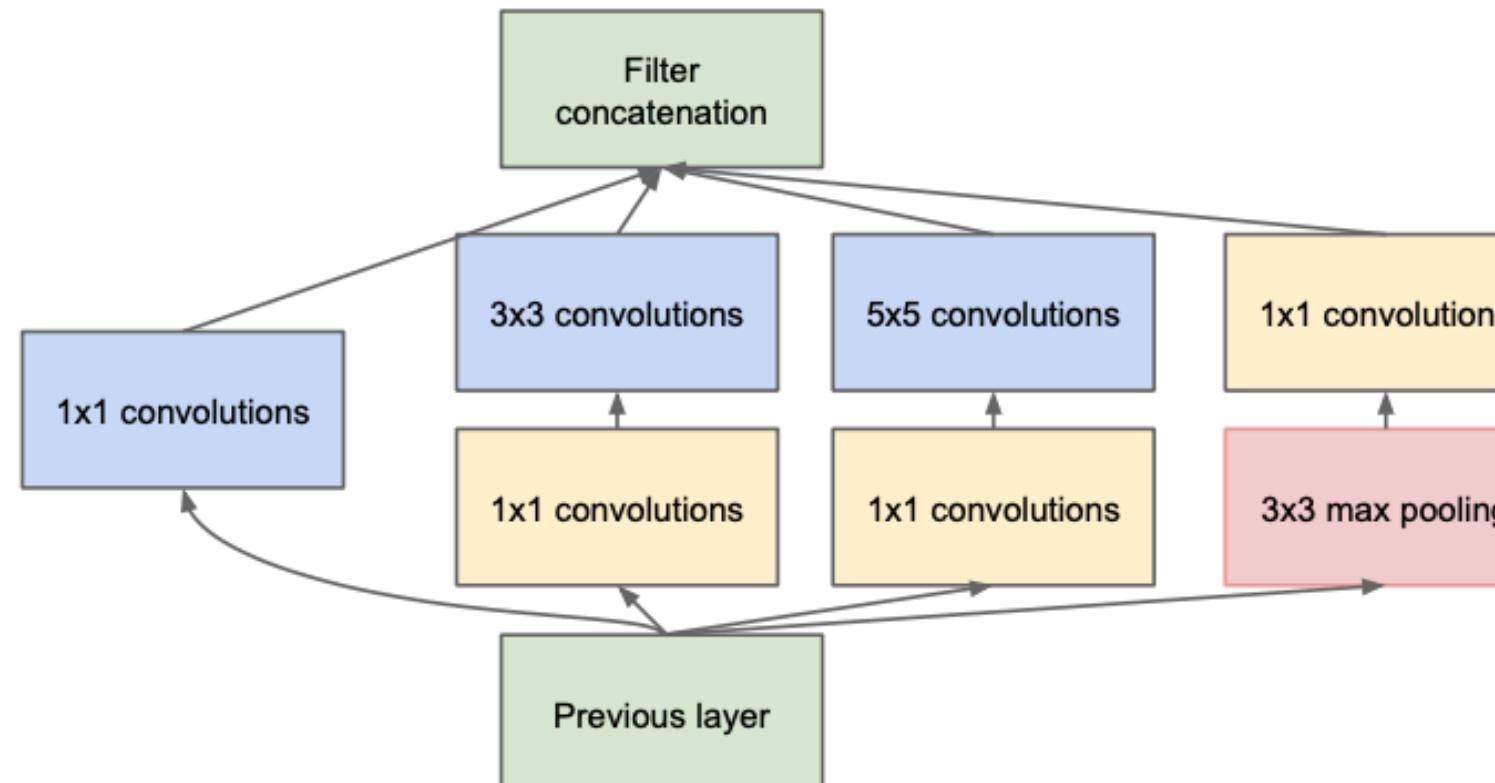


# Going Deeper with Convolutions

[YouTube Playlist](#)



(a) Inception module, naïve version



(b) Inception module with dimensionality reduction

type	patch size/stride	output size	depth	#1x1	#3x3 reduce	#3x3	#5x5 reduce	#5x5	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**

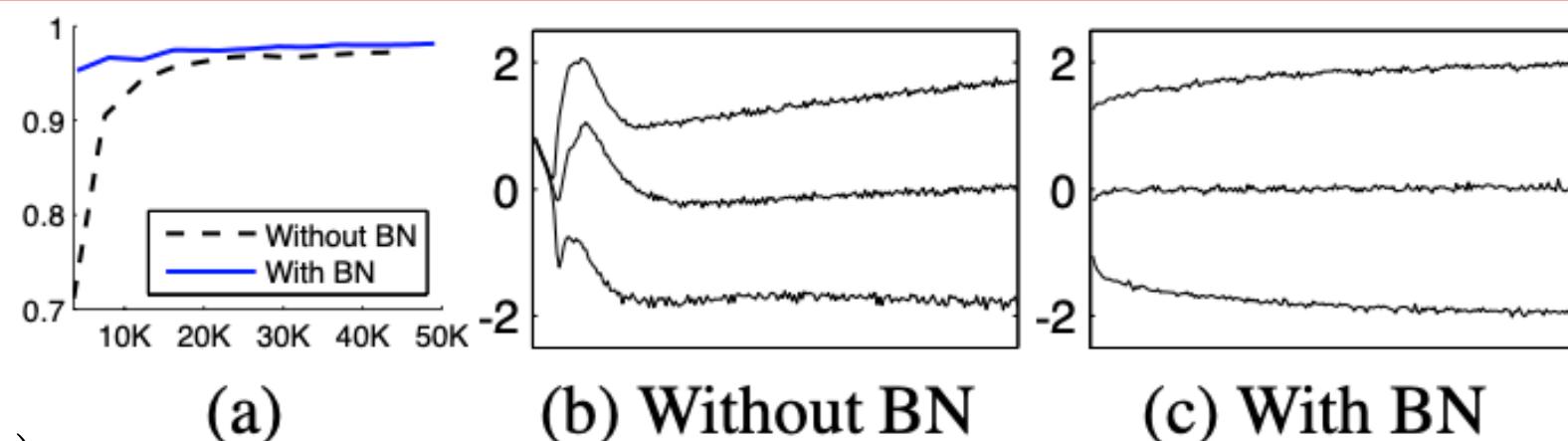
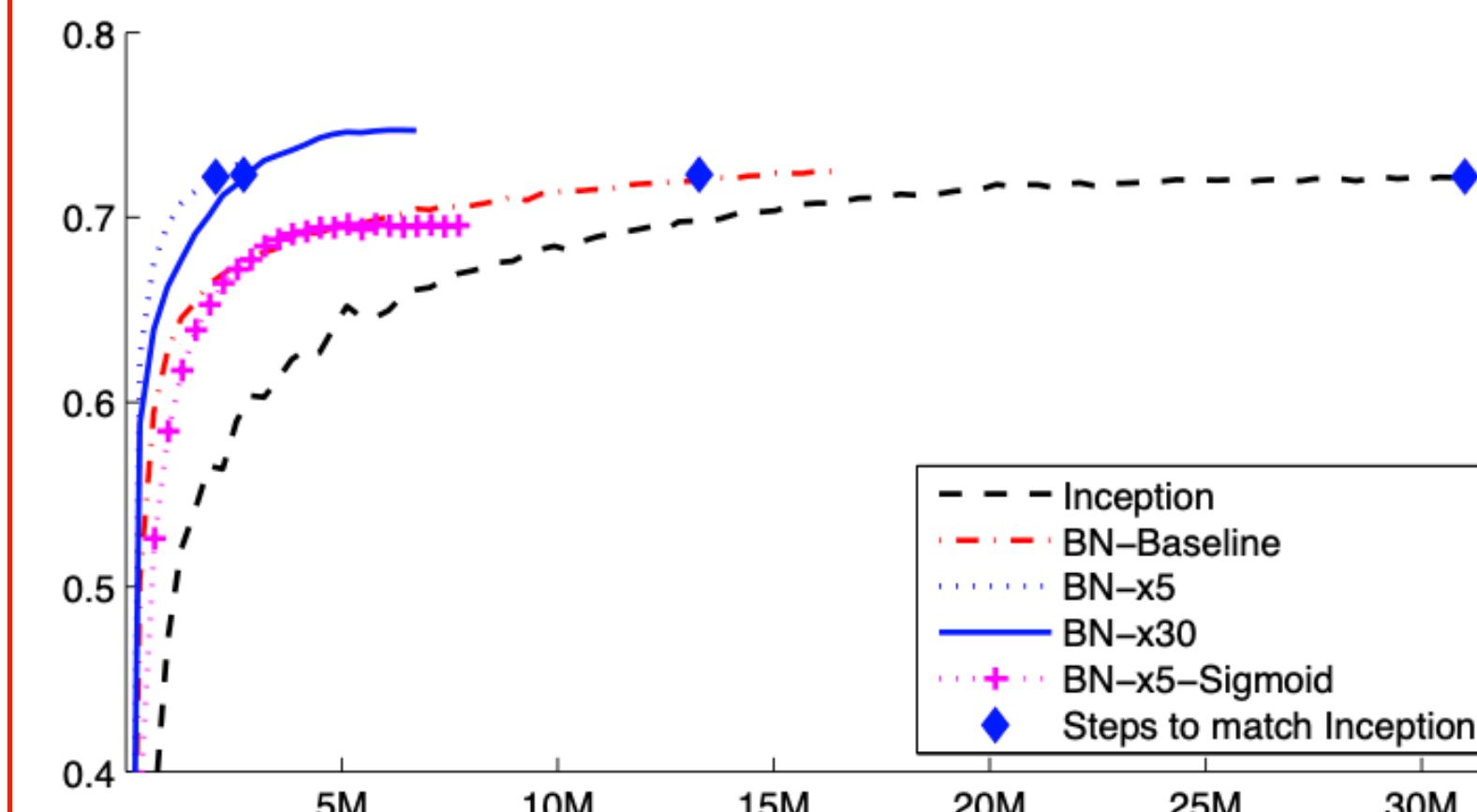
$$X \in \mathbb{R}^{p \times q \times d}$$

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

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

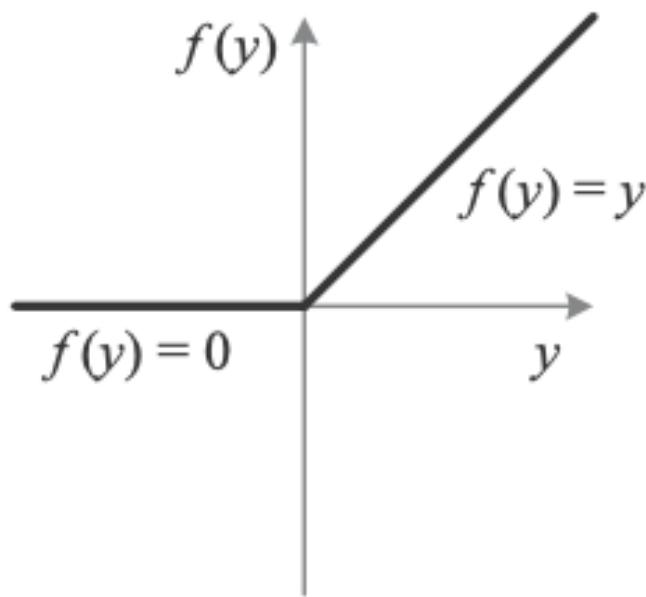
Shuffle training examples more thoroughly

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


 $f(y_i) = \max(0, y_i) + a_i \min(0, y_i) \rightarrow \text{Parametric ReLU}$ 
 $a_i = 0 \rightarrow \text{ReLU}$ 
 $a_i = 0.01 \rightarrow \text{Leaky ReLU}$ 
 $i \rightarrow \text{indicates different channels}$ 

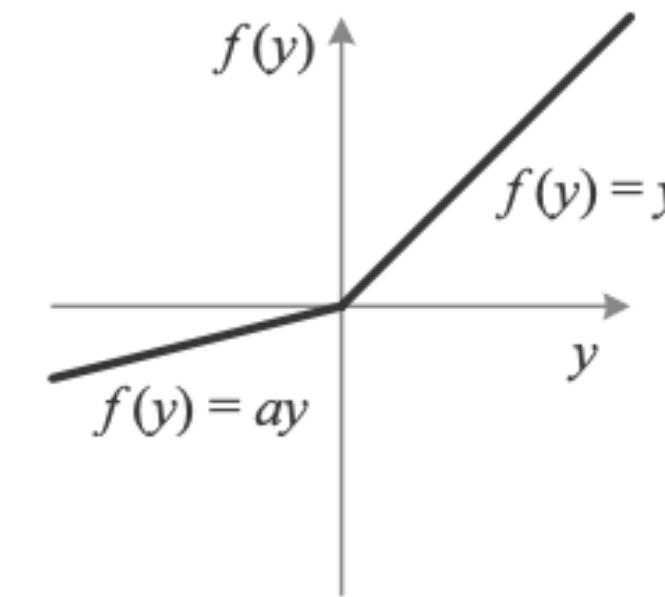
$$\Delta a_i \leftarrow \mu \Delta a_i + \epsilon \frac{\partial \mathcal{E}}{\partial a_i} \quad \text{momentum method}$$

	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

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

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

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

$x_\ell = f(y_{\ell-1})$

$c_\ell = d_{\ell-1}$

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

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

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

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

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

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

## 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

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

$\hat{n} = k^2 d$   
 $k \times k$  pixels in  $d$  channels

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

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

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

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  $\ell = 1, \forall \ell$

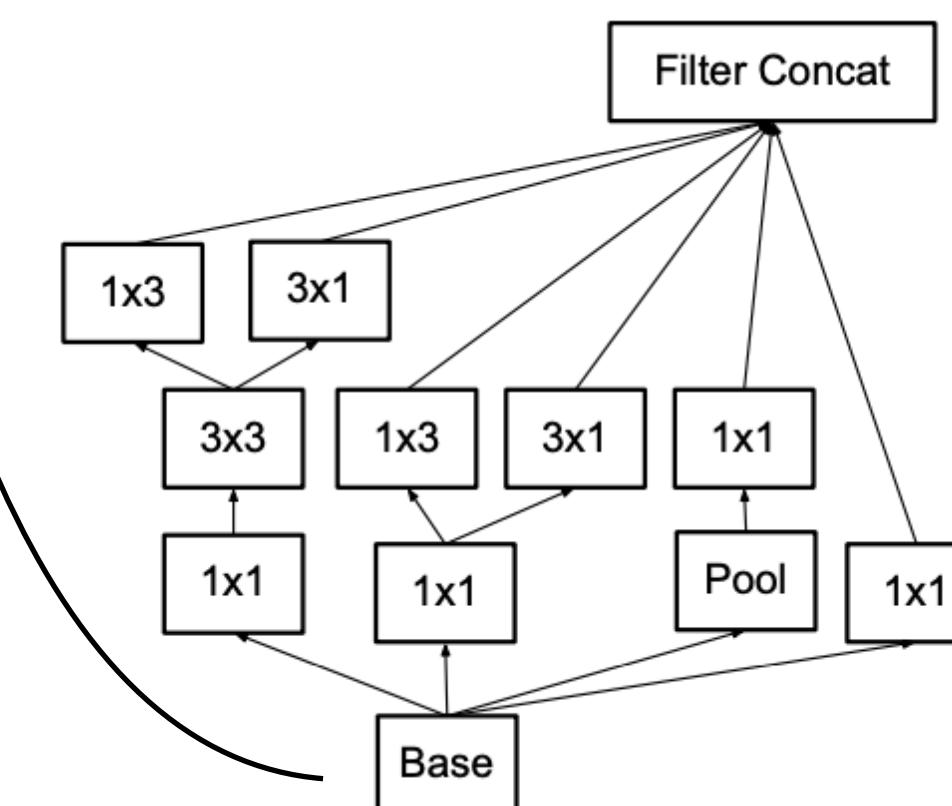


Boulder

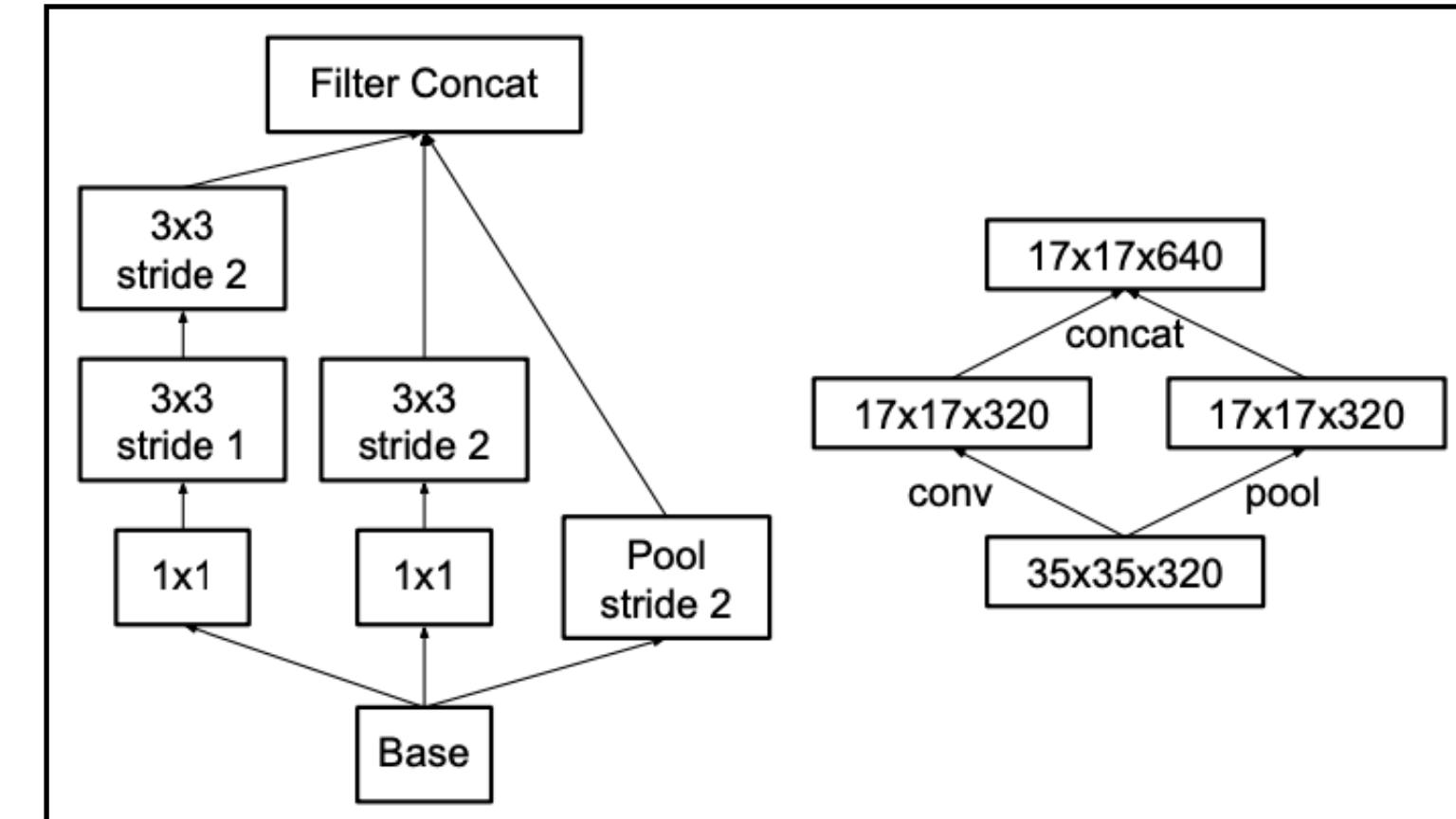
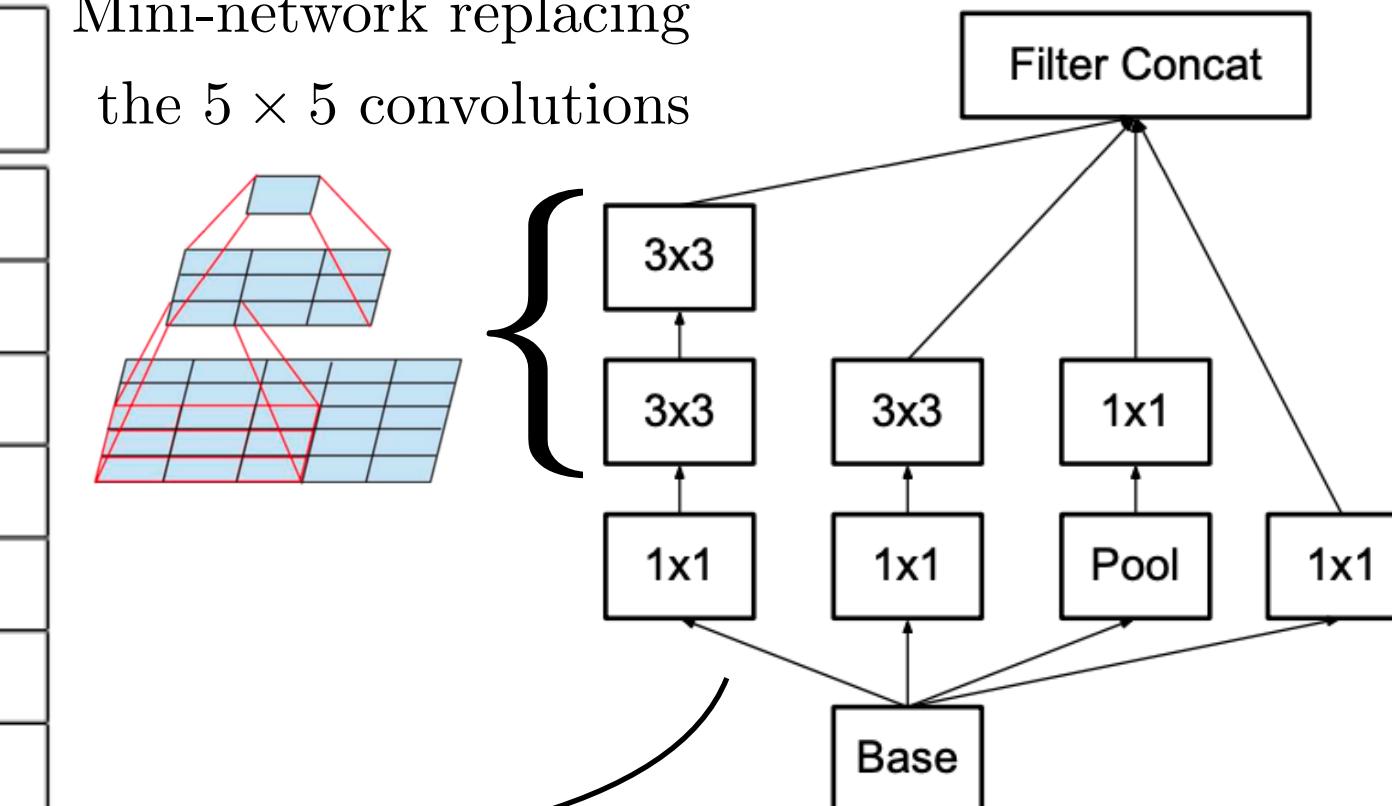
# Rethinking the Inception Architecture for Computer Vision

[YouTube Playlist](#)

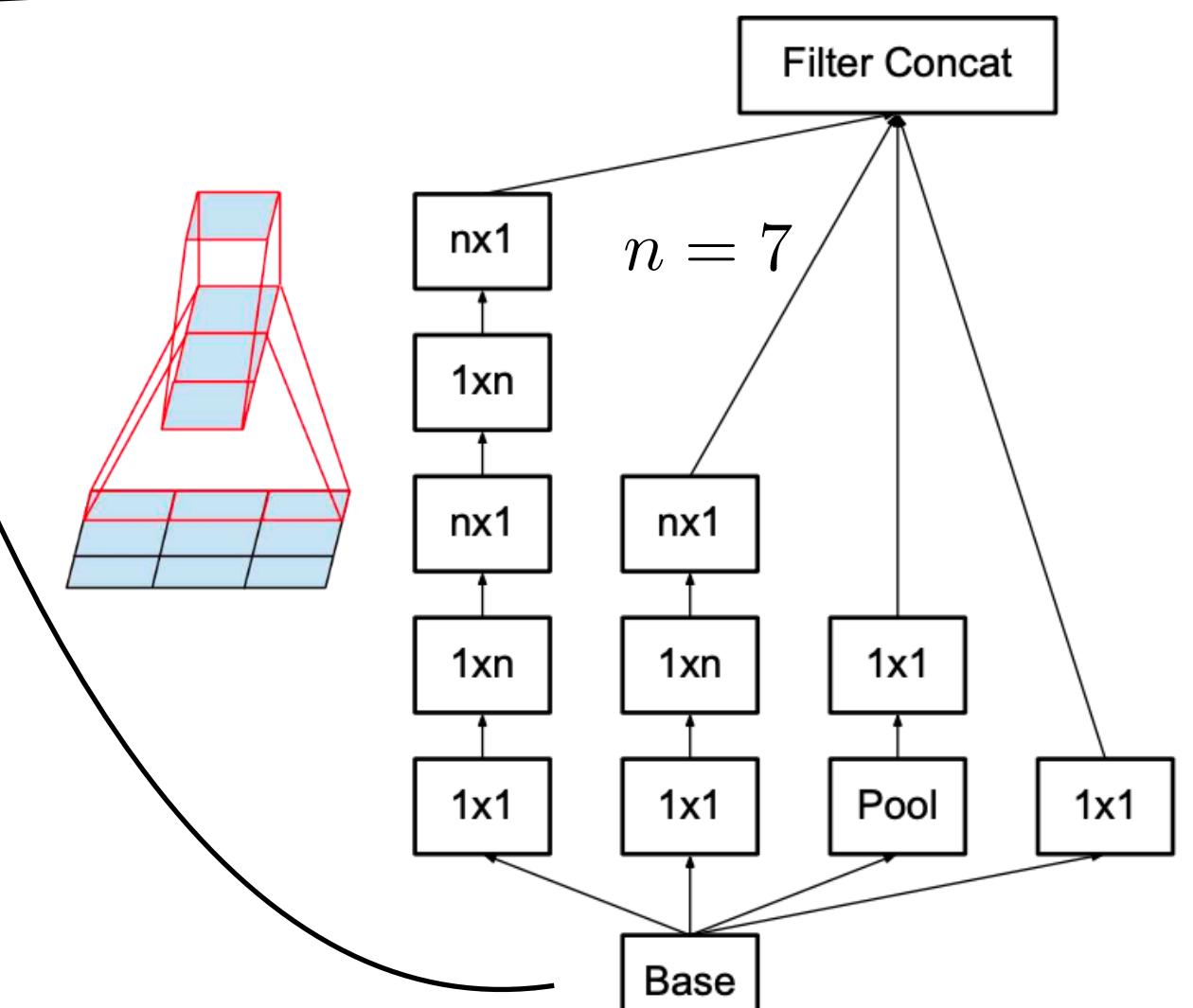
type	patch size/stride or remarks	input size
conv	$3 \times 3 / 2$	$299 \times 299 \times 3$
conv	$3 \times 3 / 1$	$149 \times 149 \times 32$
conv padded	$3 \times 3 / 1$	$147 \times 147 \times 32$
pool	$3 \times 3 / 2$	$147 \times 147 \times 64$
conv	$3 \times 3 / 1$	$73 \times 73 \times 64$
conv	$3 \times 3 / 2$	$71 \times 71 \times 80$
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



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



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

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)

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

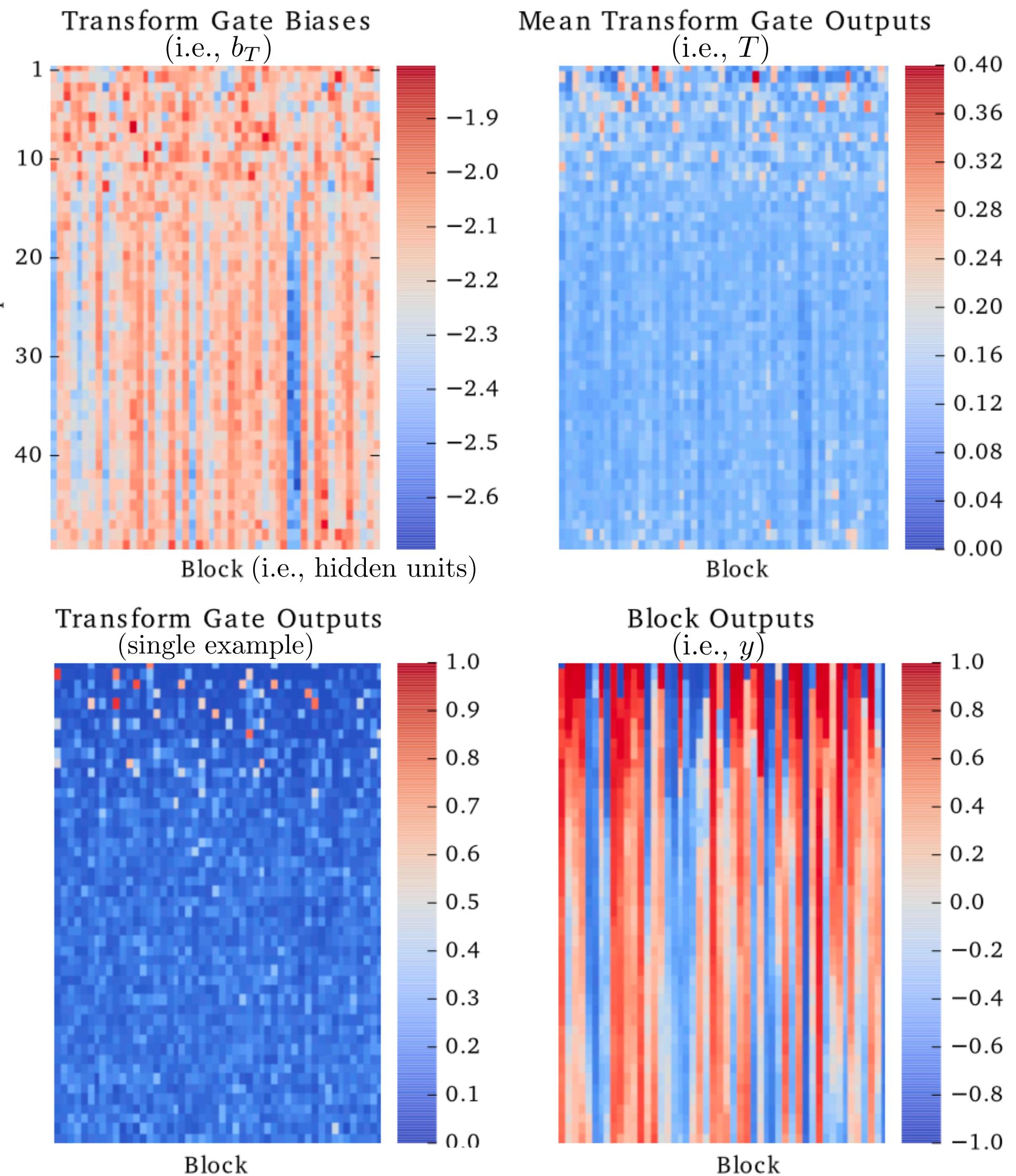
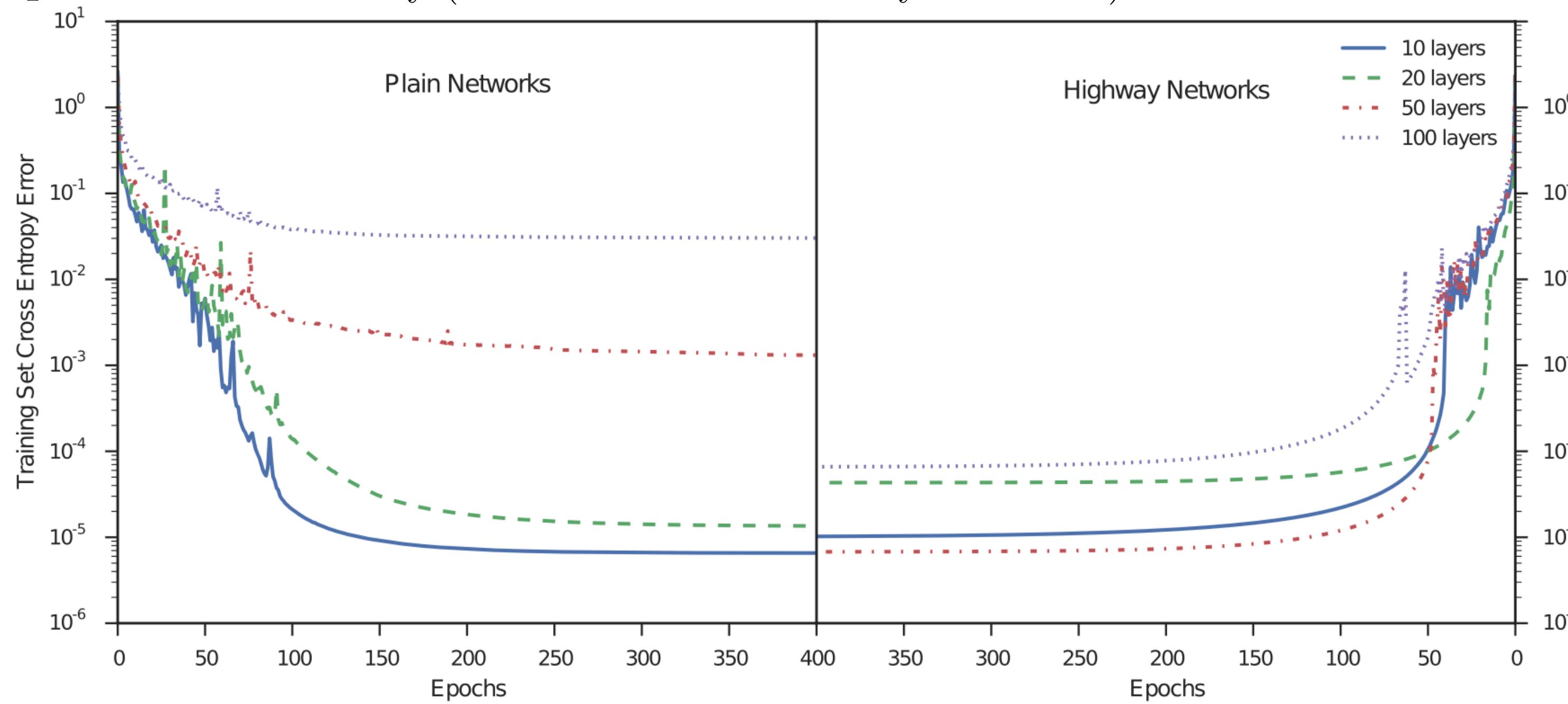
$T \rightarrow$  transform gate       $\underbrace{\cdot}_{\text{element-wise multiplication}}$

$C \rightarrow$  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).

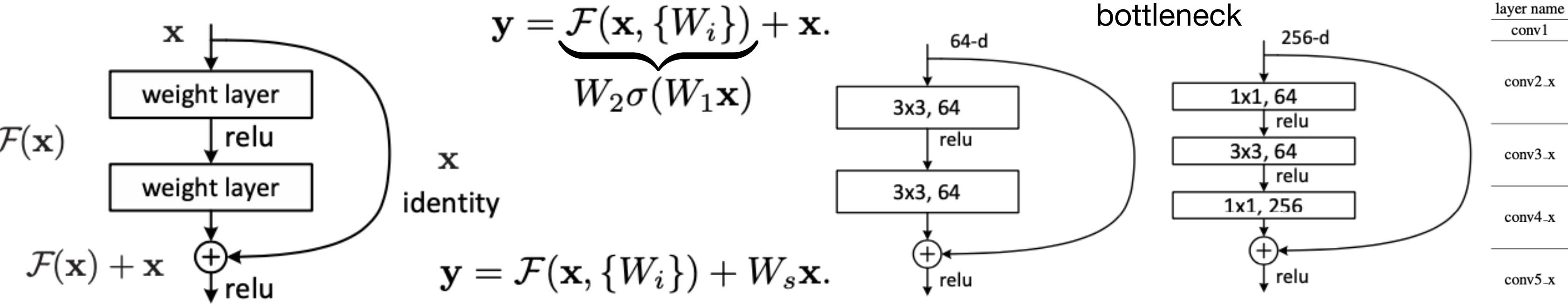


Boulder

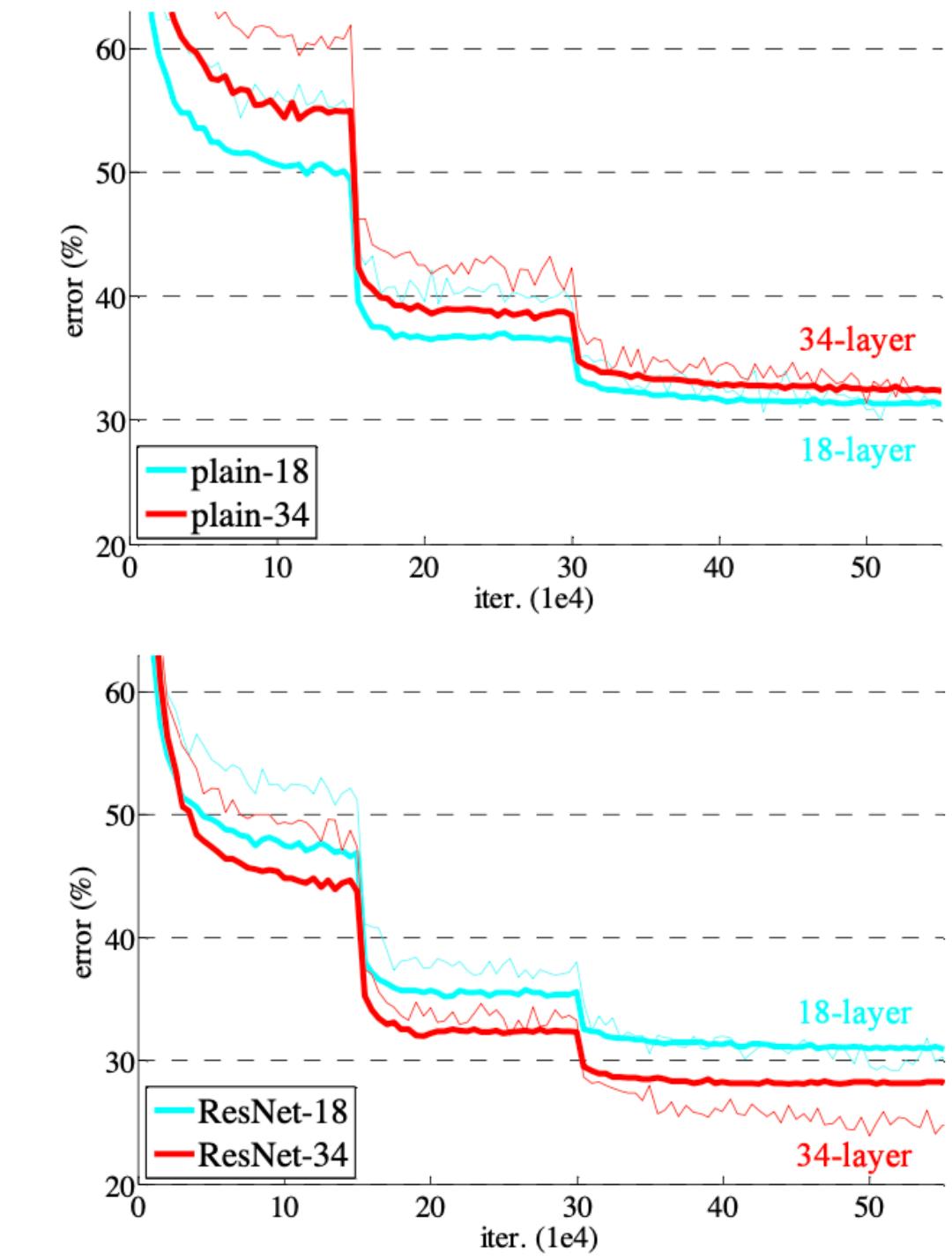
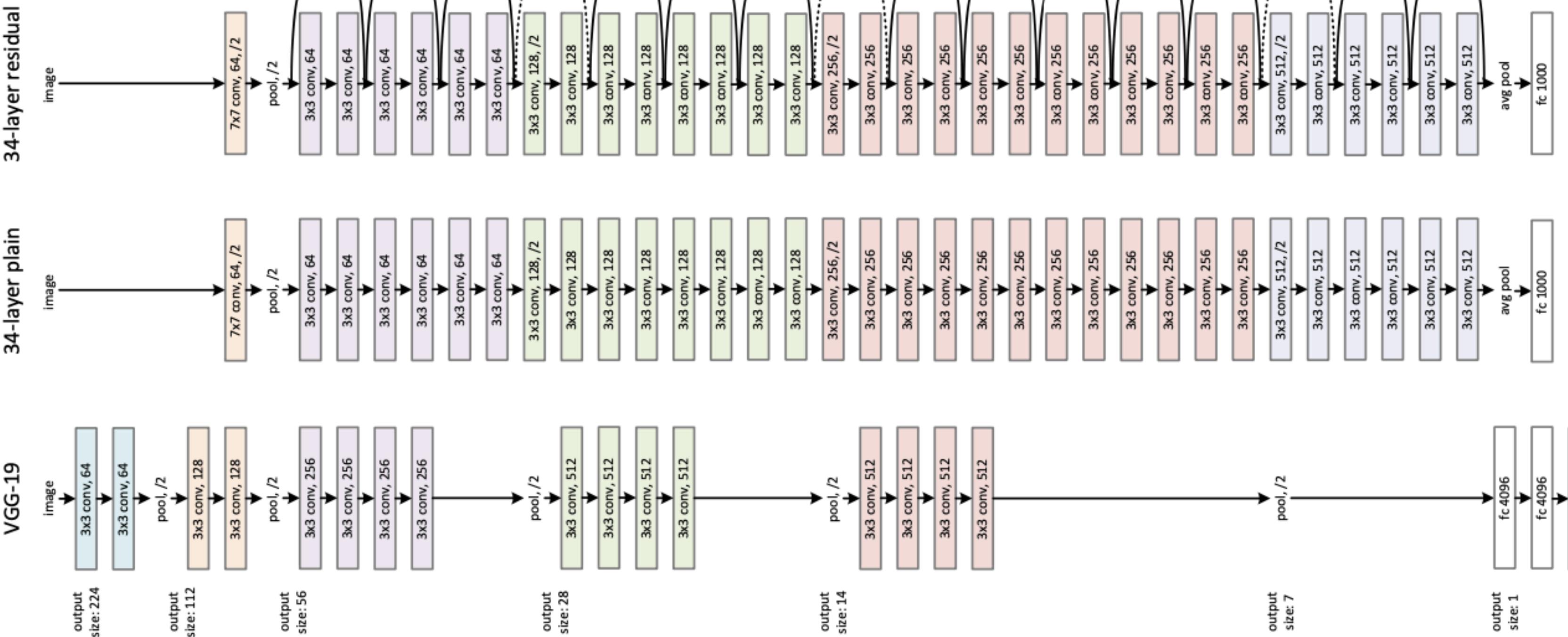


[YouTube Video](#)

# Deep Residual Learning for Image Recognition



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112x112					
conv2_x	56x56					
conv3_x	28x28					
conv4_x	14x14					
conv5_x	7x7					
		average pool, 1000-d fc, softmax				
	FLOPs	$1.8 \times 10^9$	$3.6 \times 10^9$	$3.8 \times 10^9$	$7.6 \times 10^9$	$11.3 \times 10^9$



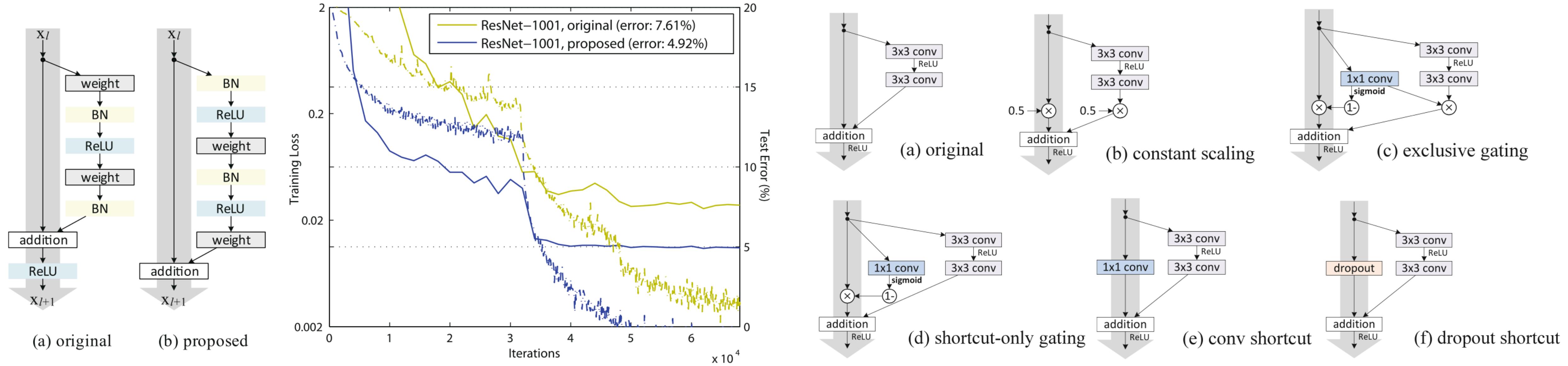


Boulder



[YouTube Video](#)

# Identity Mappings in Deep Residual Networks



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

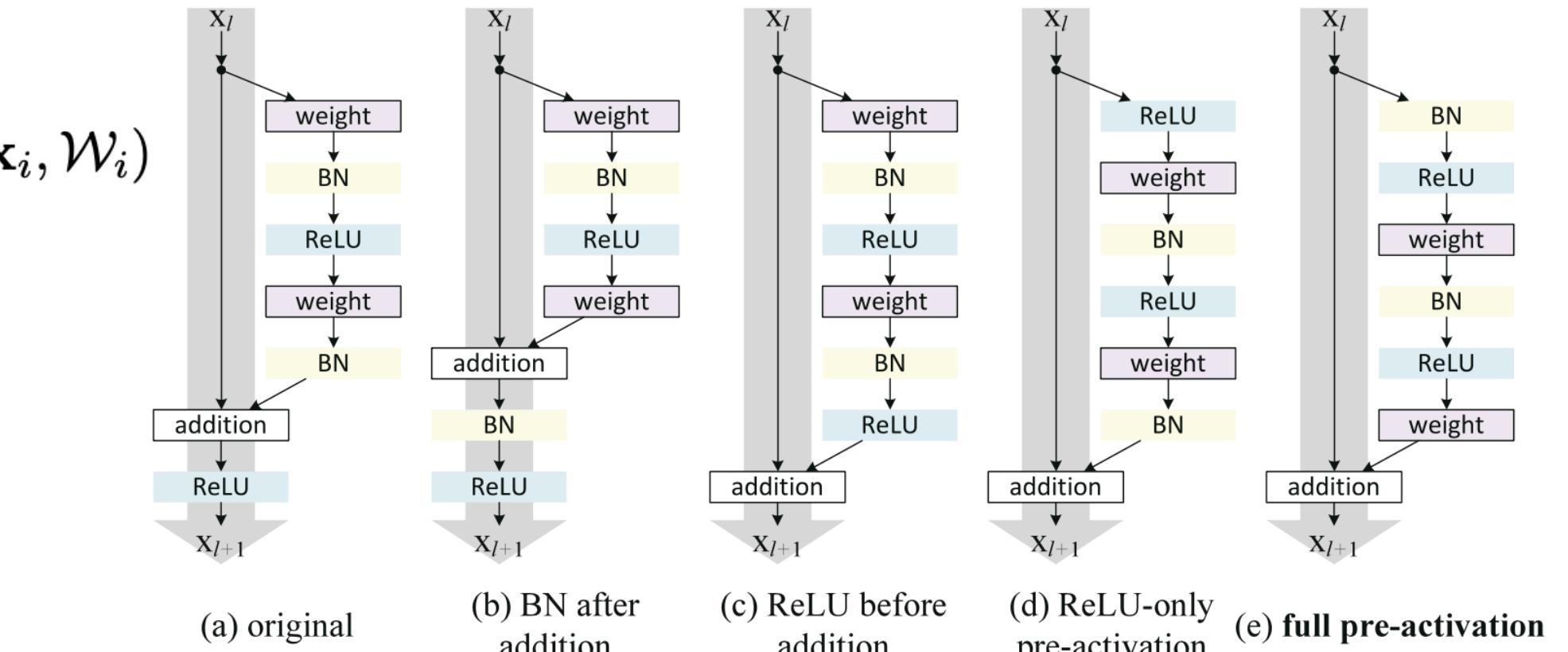
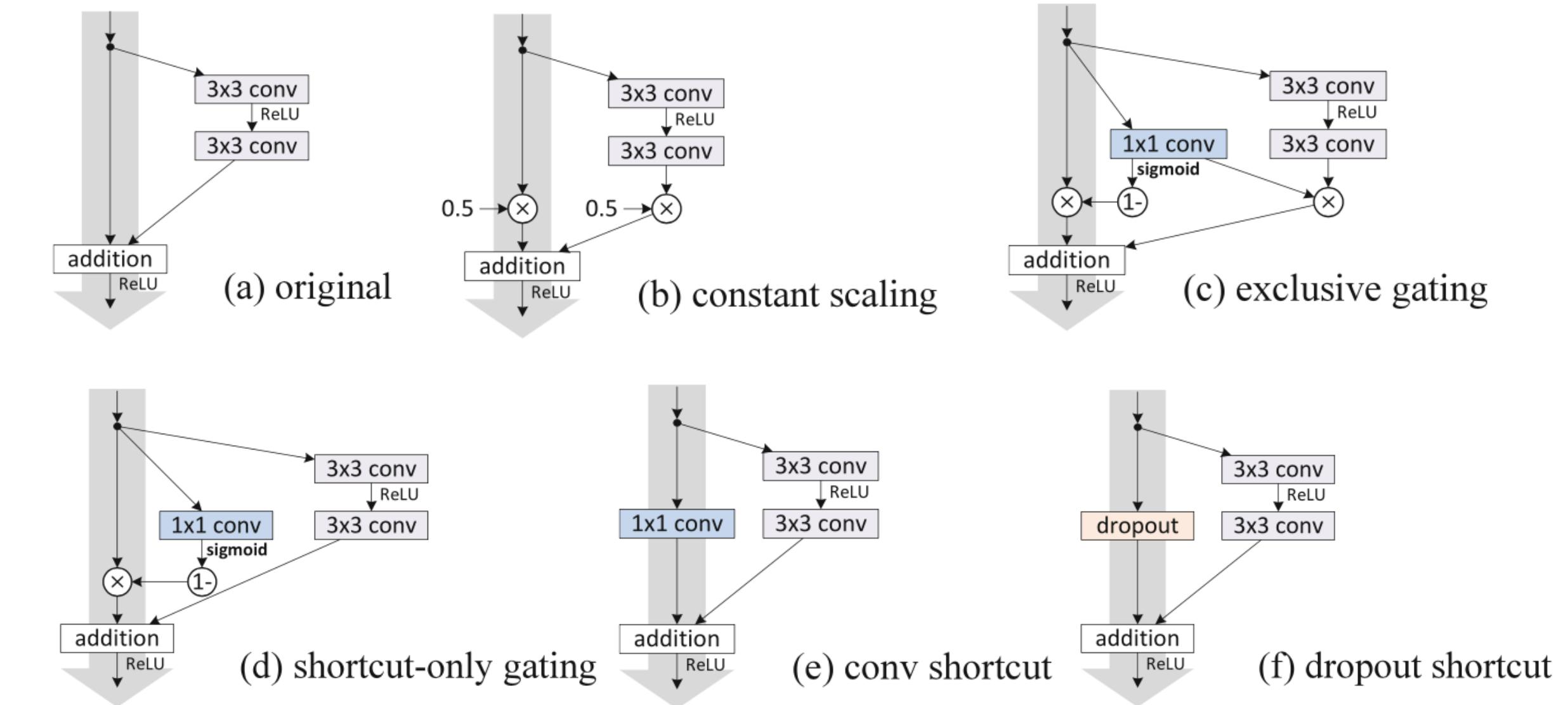
$$h(\mathbf{x}_l) = \mathbf{x}_l \\ f \text{ is ReLU} \quad \left. \right\} \text{Original}$$

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

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

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

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





Boulder

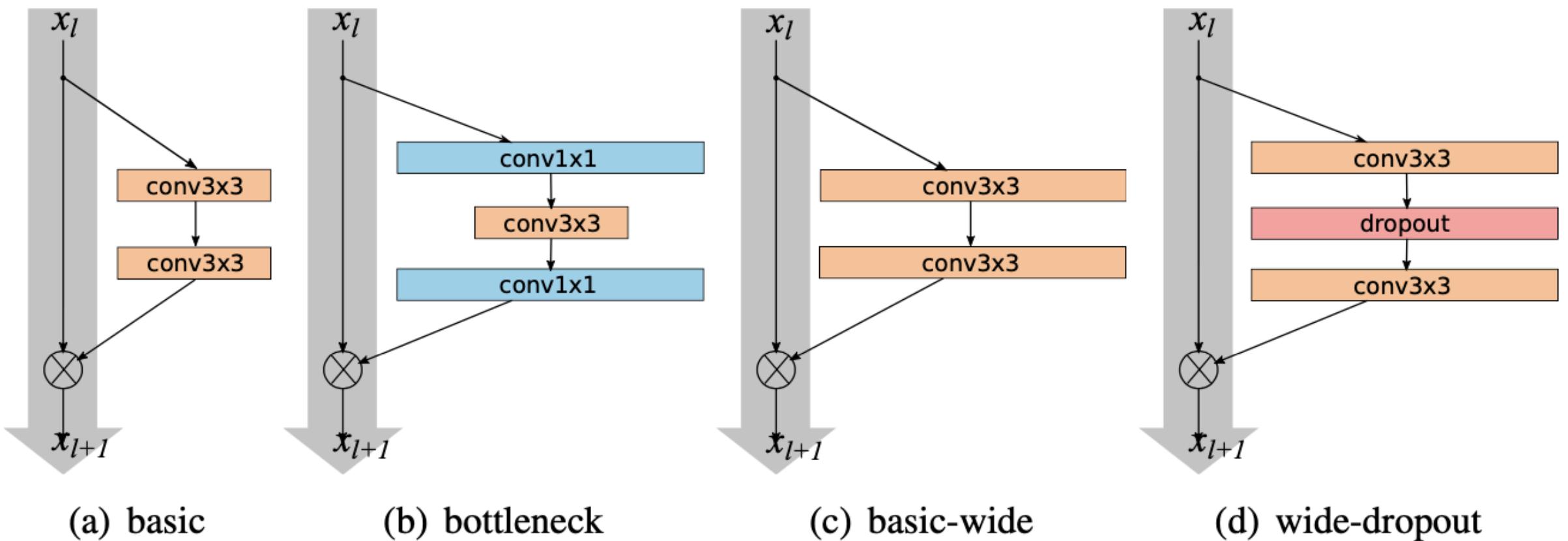


[YouTube Video](#)

# 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

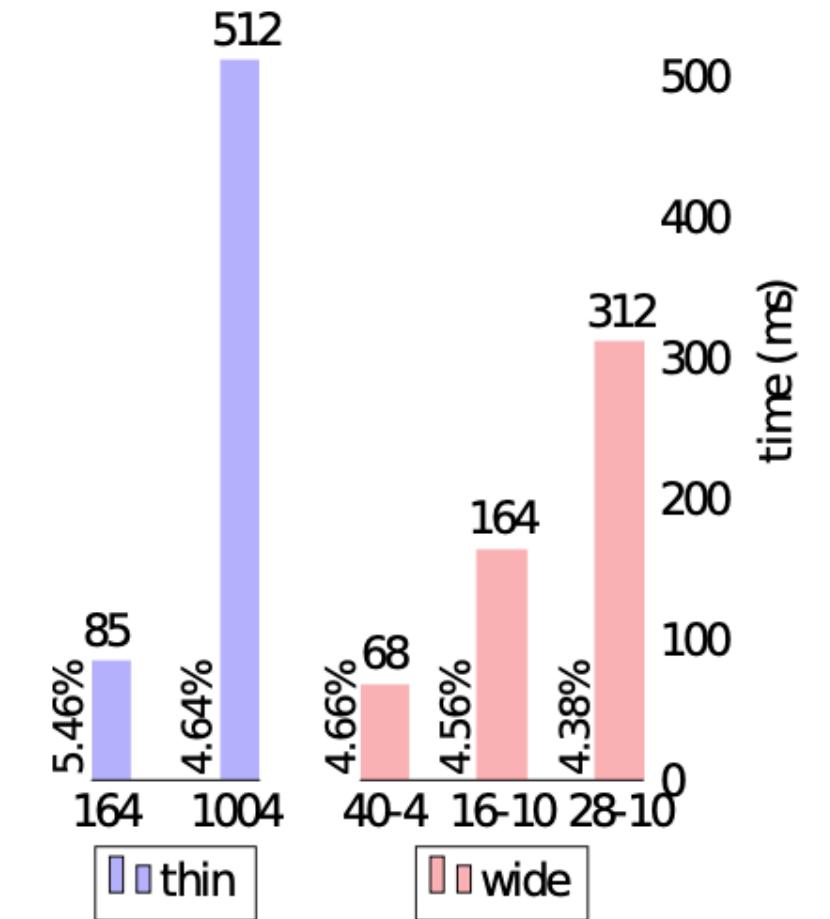
$\ell \rightarrow$  deepening factor (number of convs in a block)

$l$	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>



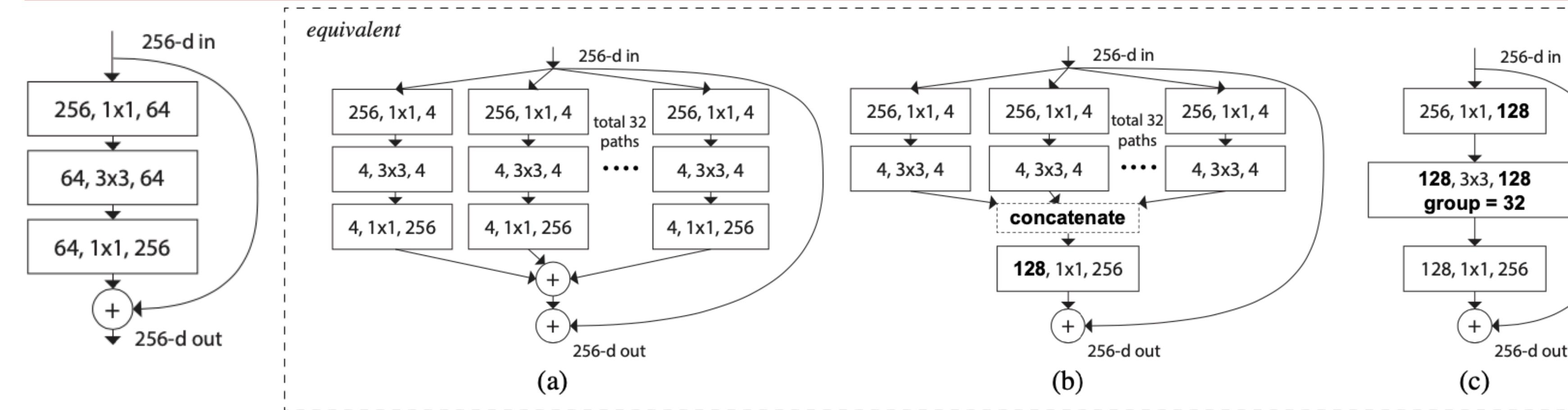


Boulder

# Aggregated Residual Transformations for Deep Neural Networks



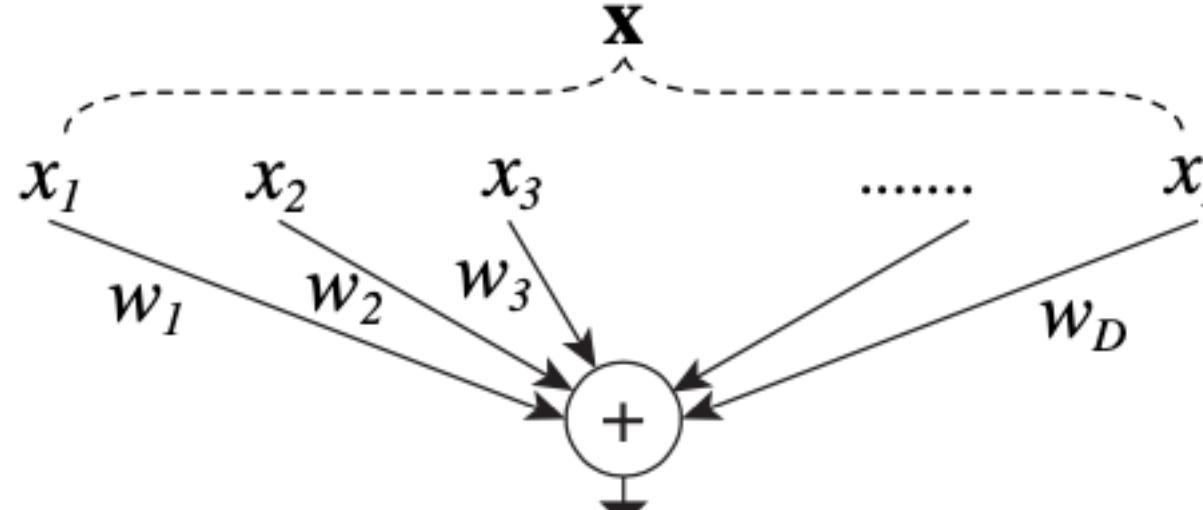
[YouTube Video](#)



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

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.

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

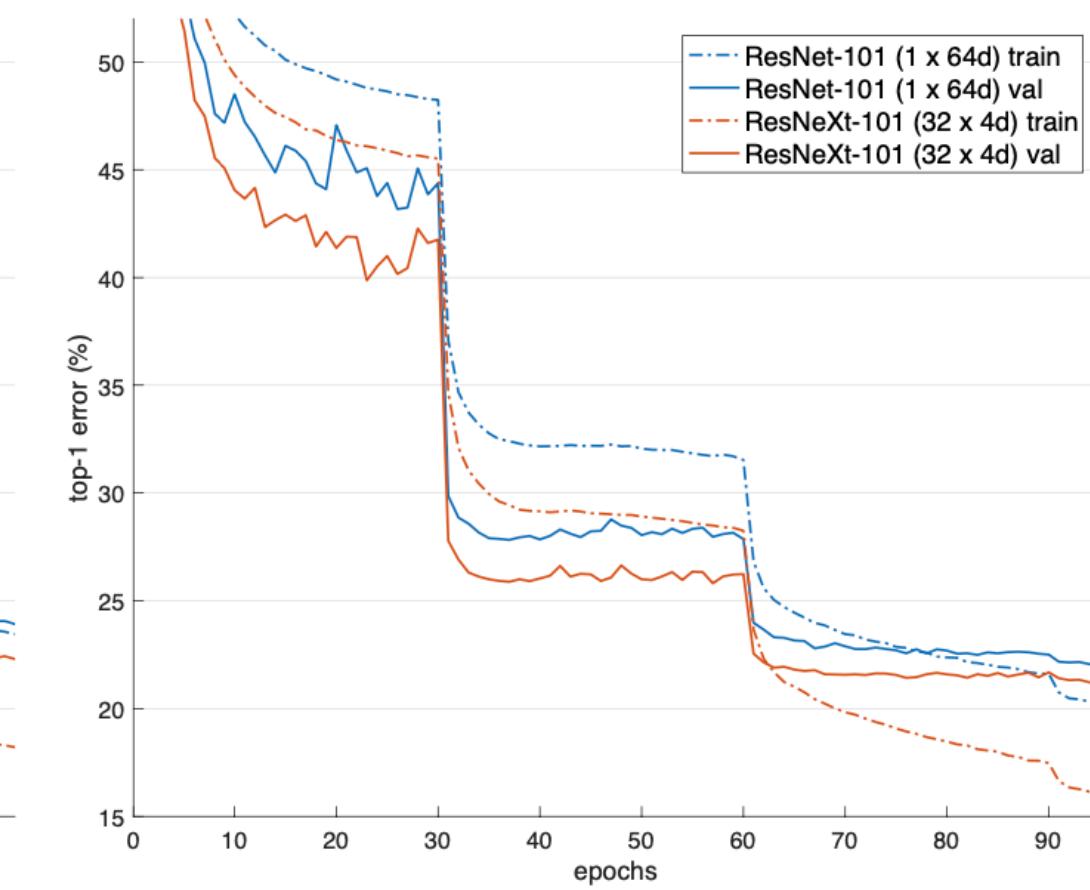
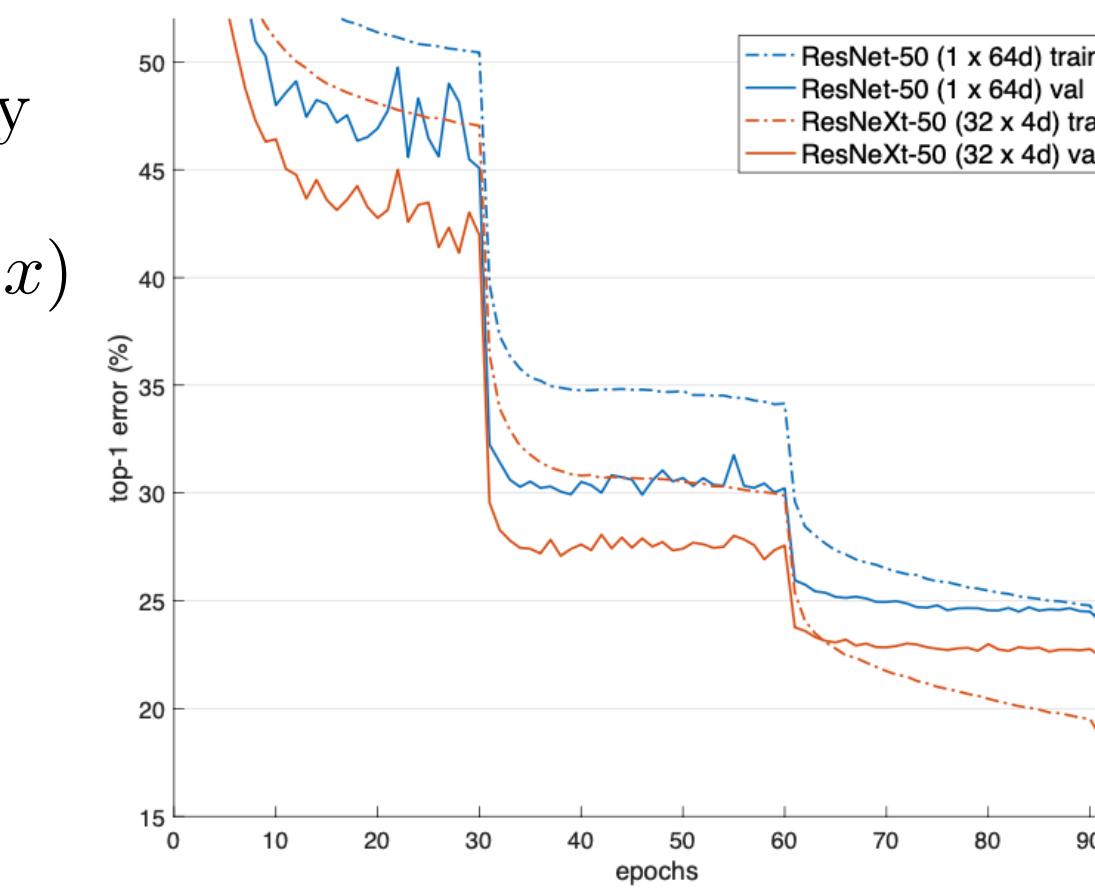
$C \rightarrow$  cardinality

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

split:  $x \rightarrow x_i$

transform:  $x_i w_i$

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



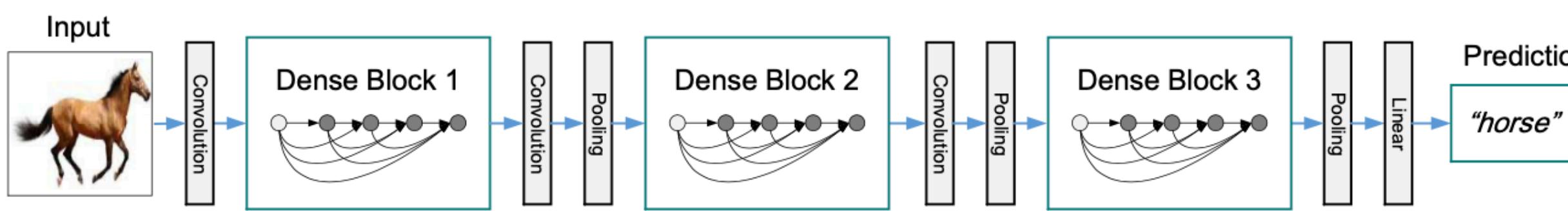
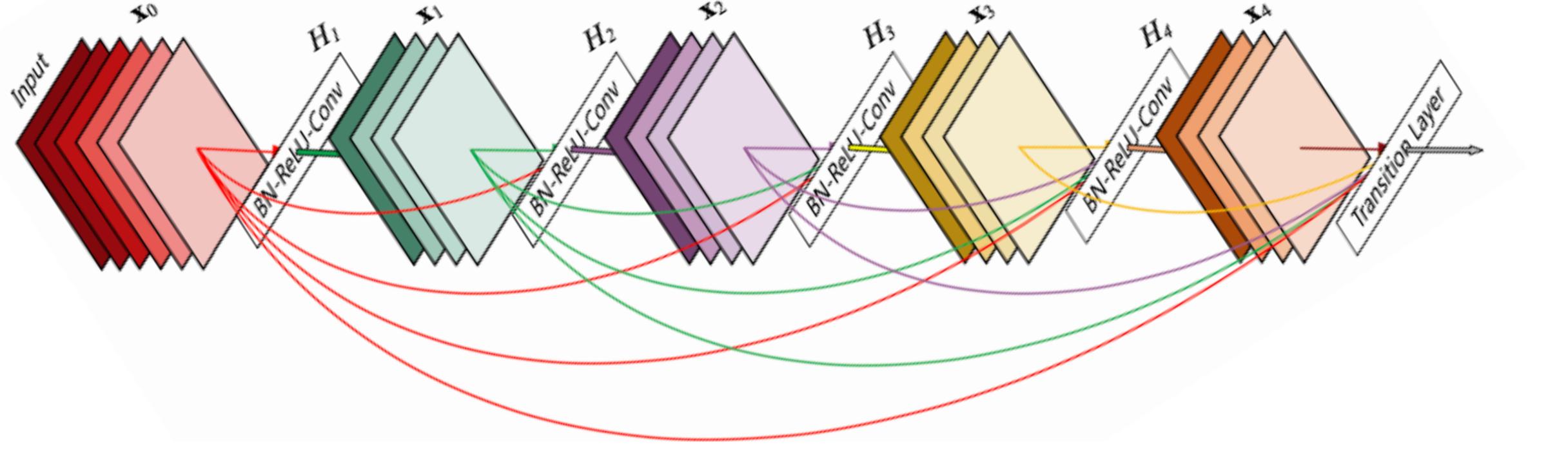


Boulder



[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

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

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

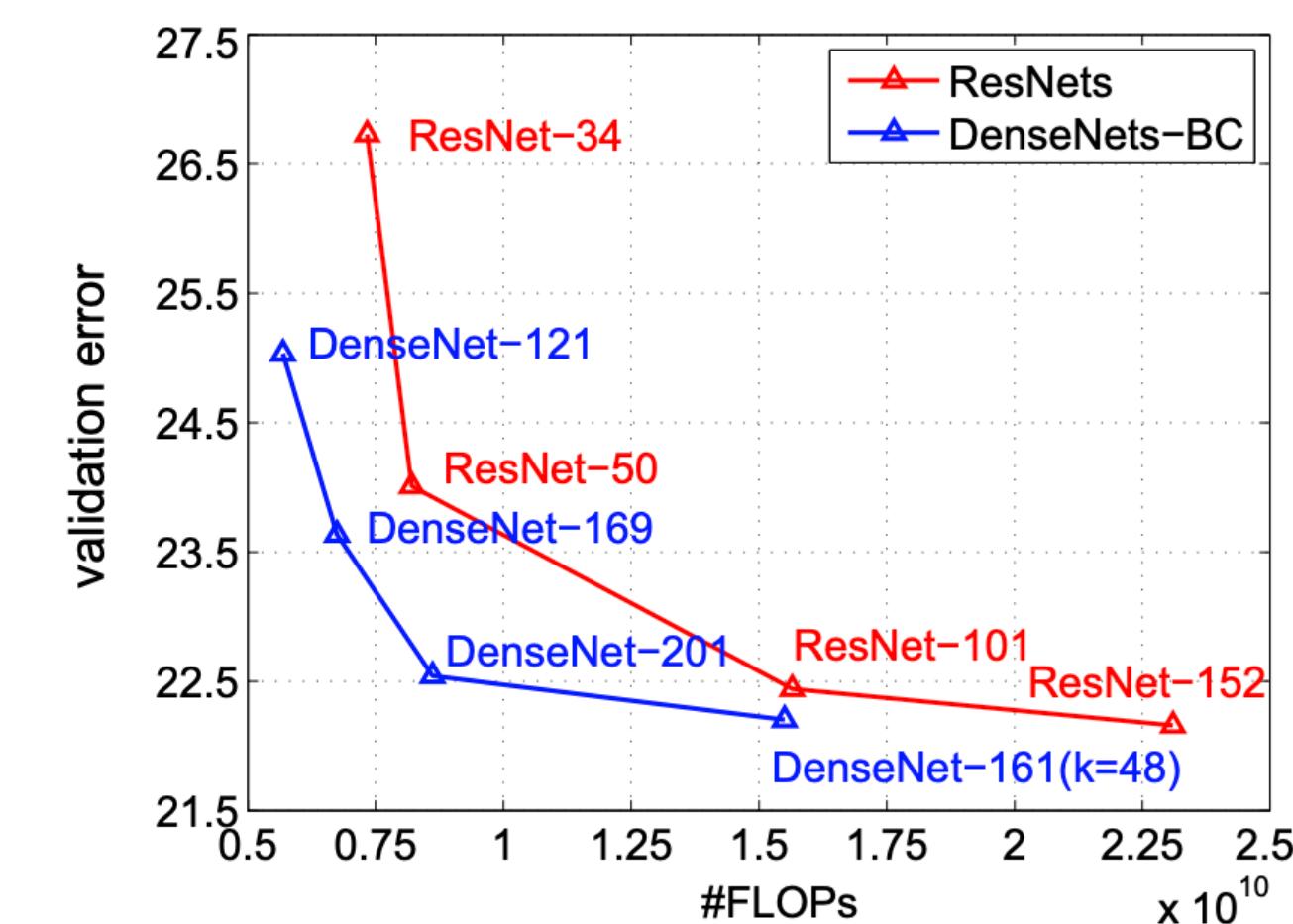
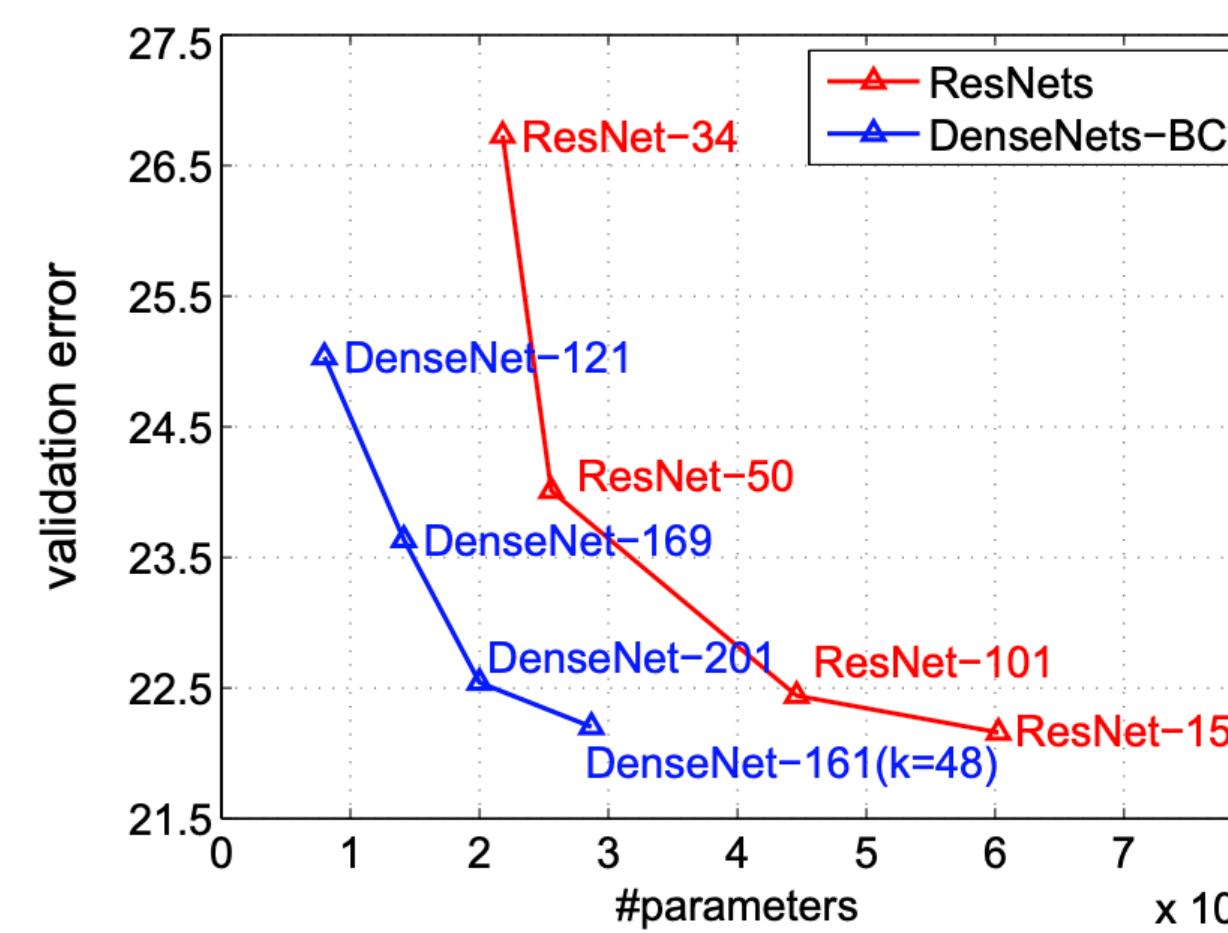
$x_\ell = H_\ell(\underbrace{[x_0, x_1, \dots, x_{\ell-1}]}_{\text{concatenate}})$

BN-ReLU-Conv(1x1)-BN-ReLU-Conv(3x3)  
↳ 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	$[1 \times 1 \text{ conv}, 3 \times 3 \text{ conv}] \times 6$	$[1 \times 1 \text{ conv}, 3 \times 3 \text{ conv}] \times 6$	$[1 \times 1 \text{ conv}, 3 \times 3 \text{ conv}] \times 6$	$[1 \times 1 \text{ conv}, 3 \times 3 \text{ conv}] \times 6$
	56 × 56				
Transition Layer (1)	56 × 56			$1 \times 1 \text{ conv}$	
Dense Block (2)	28 × 28			$2 \times 2 \text{ average pool, stride 2}$	
	28 × 28	$[1 \times 1 \text{ conv}, 3 \times 3 \text{ conv}] \times 12$	$[1 \times 1 \text{ conv}, 3 \times 3 \text{ conv}] \times 12$	$[1 \times 1 \text{ conv}, 3 \times 3 \text{ conv}] \times 12$	$[1 \times 1 \text{ conv}, 3 \times 3 \text{ conv}] \times 12$
Transition Layer (2)	28 × 28			$1 \times 1 \text{ conv}$	
Dense Block (3)	14 × 14			$2 \times 2 \text{ average pool, stride 2}$	
	14 × 14	$[1 \times 1 \text{ conv}, 3 \times 3 \text{ conv}] \times 24$	$[1 \times 1 \text{ conv}, 3 \times 3 \text{ conv}] \times 32$	$[1 \times 1 \text{ conv}, 3 \times 3 \text{ conv}] \times 48$	$[1 \times 1 \text{ conv}, 3 \times 3 \text{ conv}] \times 64$
Transition Layer (3)	14 × 14			$1 \times 1 \text{ conv}$	
Dense Block (4)	7 × 7			$2 \times 2 \text{ average pool, stride 2}$	
	7 × 7	$[1 \times 1 \text{ conv}, 3 \times 3 \text{ conv}] \times 16$	$[1 \times 1 \text{ conv}, 3 \times 3 \text{ conv}] \times 32$	$[1 \times 1 \text{ conv}, 3 \times 3 \text{ conv}] \times 32$	$[1 \times 1 \text{ conv}, 3 \times 3 \text{ conv}] \times 48$
Classification Layer	1 × 1			$7 \times 7 \text{ global average pool}$	1000D fully-connected, softmax



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

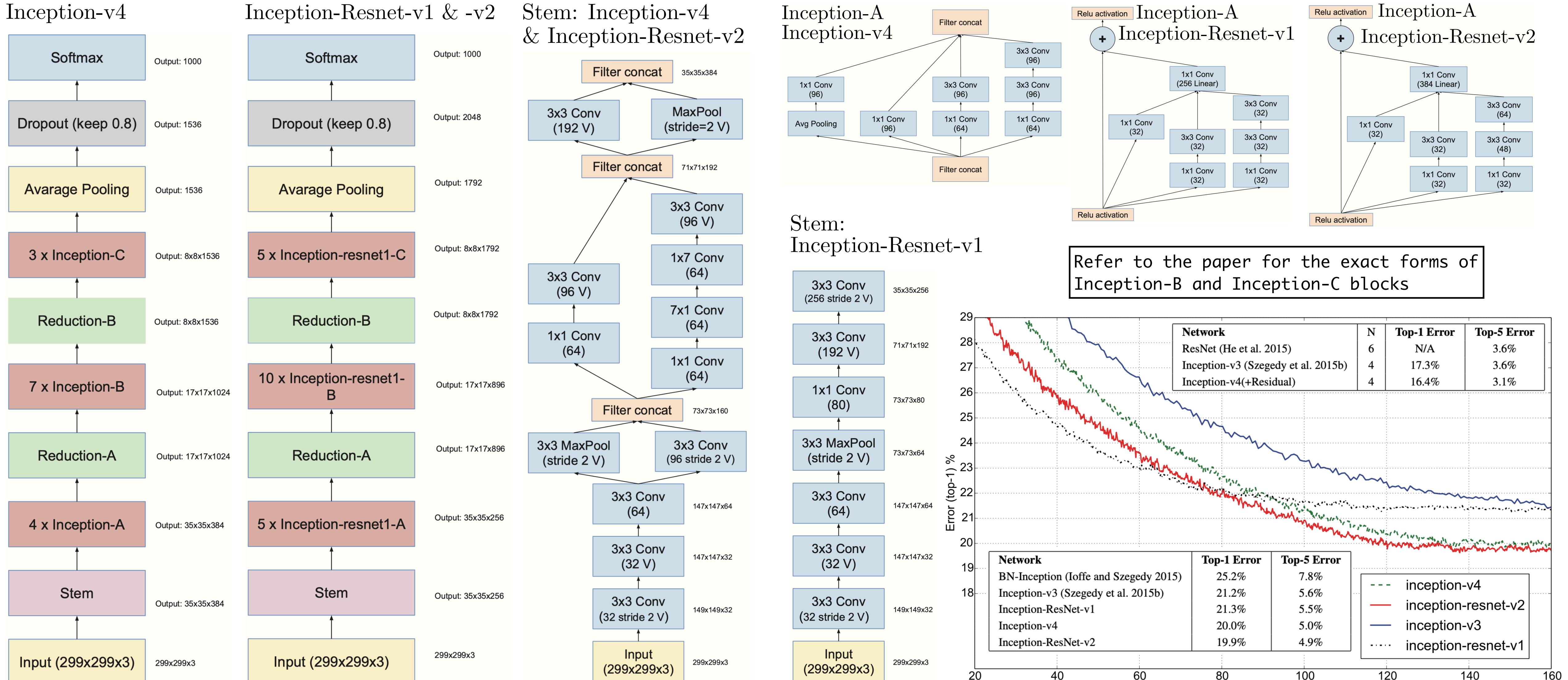
Huang, Gao, et al. "Convolutional networks with dense connectivity." IEEE transactions on pattern analysis and machine intelligence (2019).



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



[YouTube Video](#)



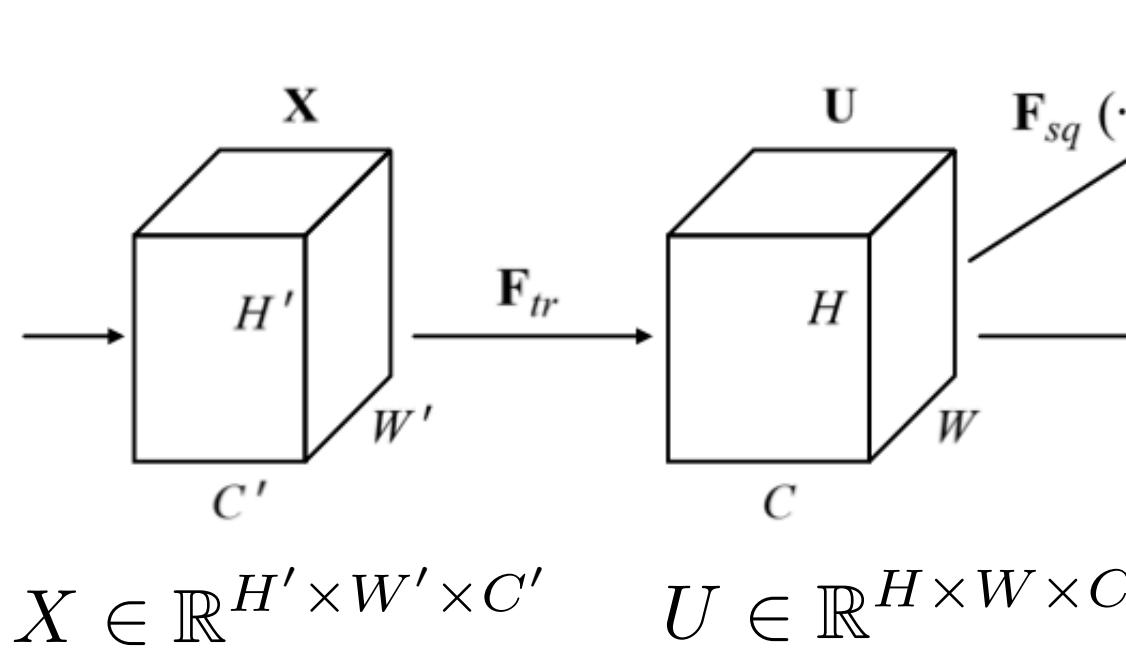


Boulder



[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**

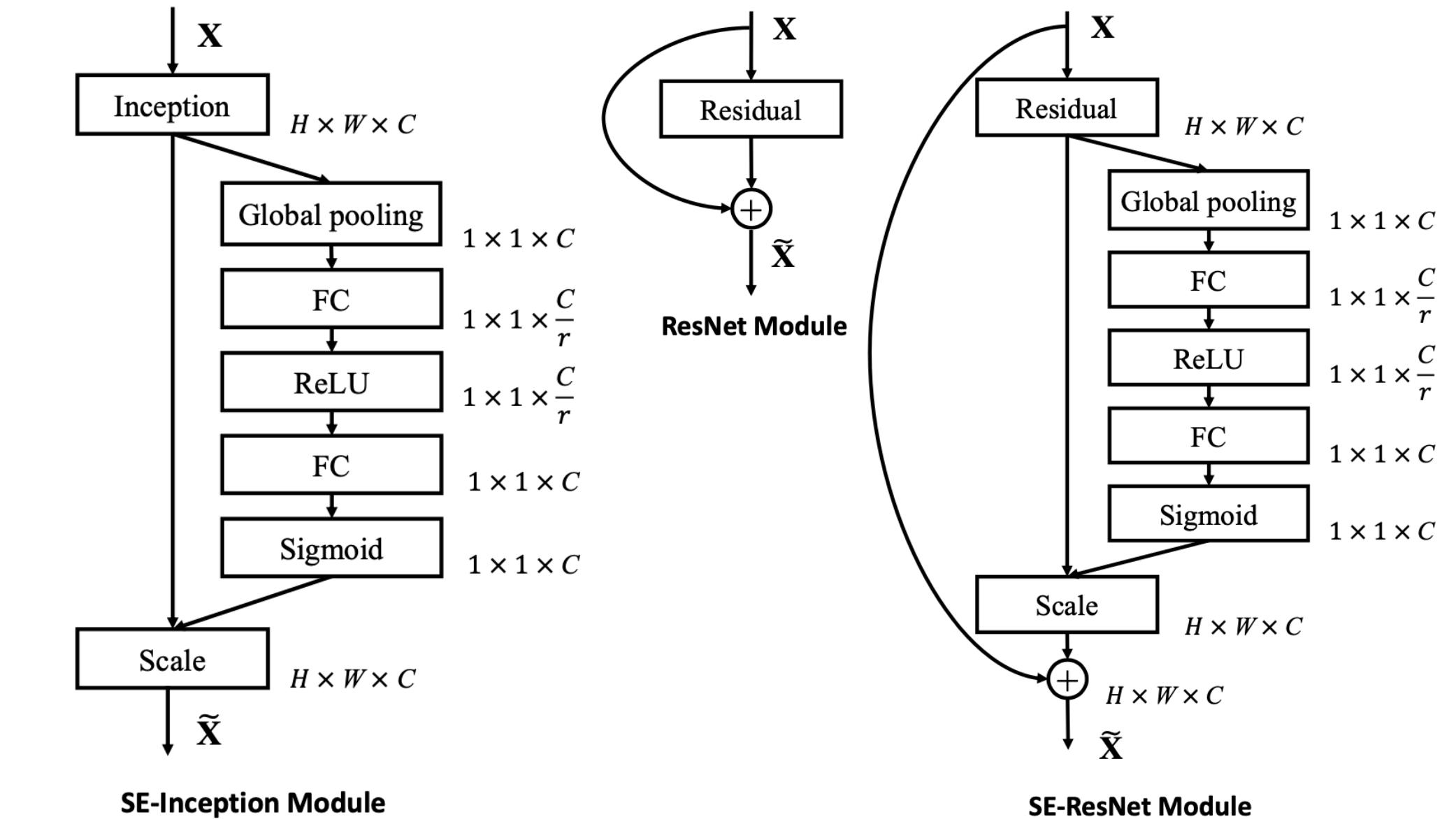
$$s = F_{ex}(z, W) = \sigma(g(z, W)) = \sigma(W_2 \delta(W_1 z))$$

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

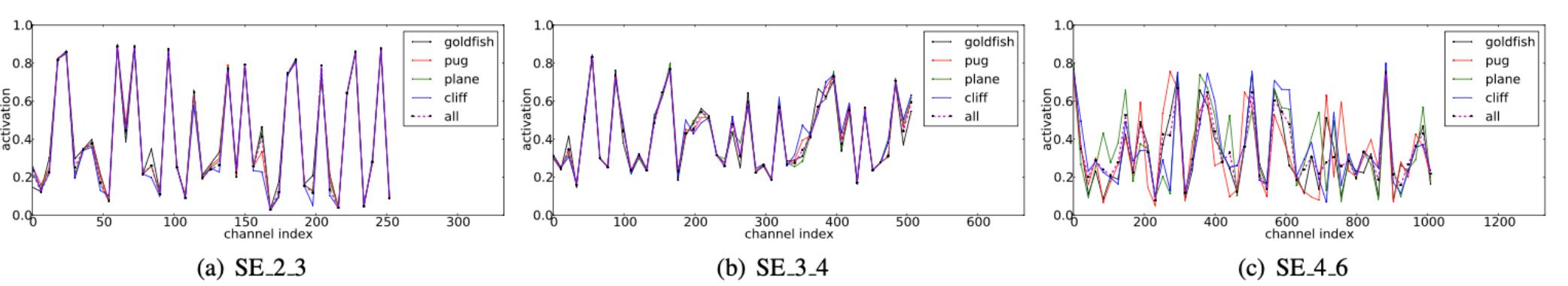
$$W_2 \in \mathbb{R}^{C \times \frac{C}{r}}$$

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

	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>



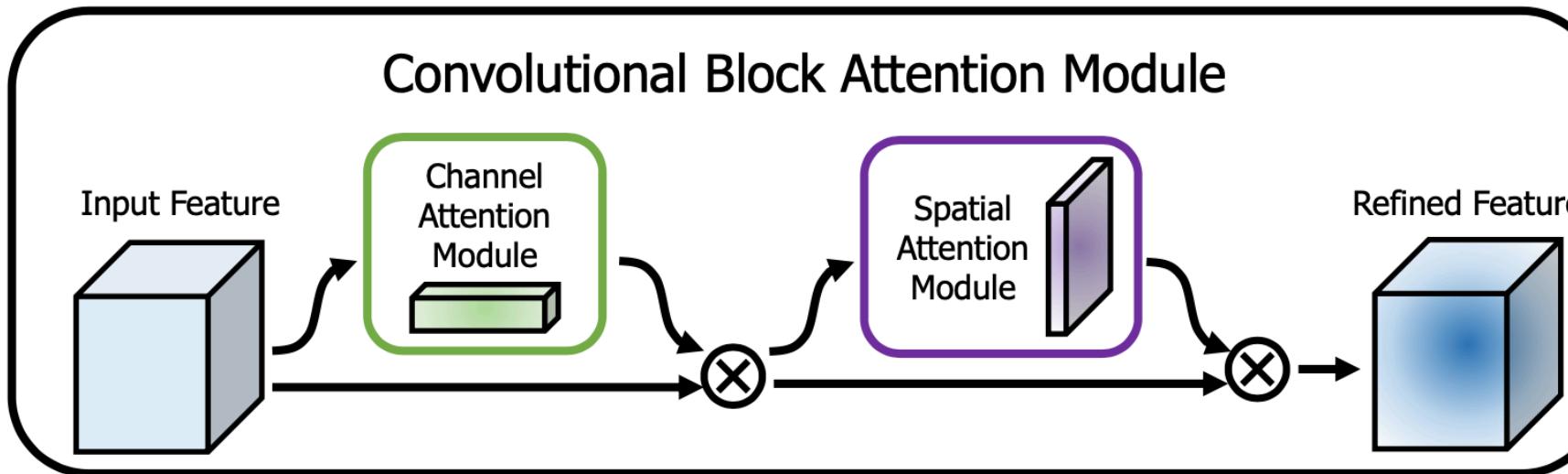
	original		re-implementation			SENet		
	top-1 err.	top-5 err.	top-1 err.	top-5 err.	GFLOPs	top-1 err.	top-5 err.	GFLOPs
ResNet-50 [10]	24.7	7.8	24.80	7.48	3.86	23.29 <sub>(1.51)</sub>	6.62 <sub>(0.86)</sub>	3.87
ResNet-101 [10]	23.6	7.1	23.17	6.52	7.58	22.38 <sub>(0.79)</sub>	6.07 <sub>(0.45)</sub>	7.60
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



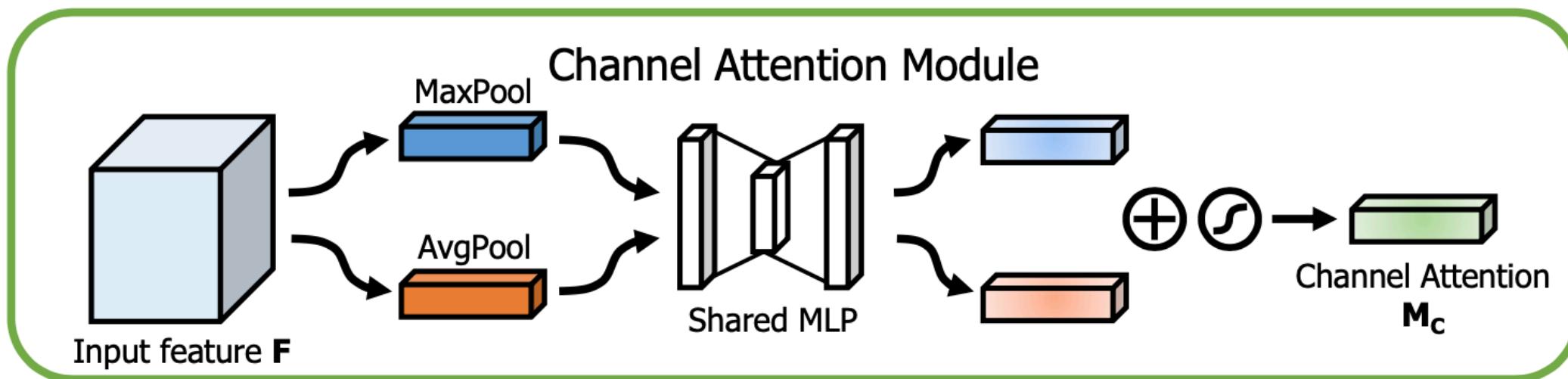
$$F \in \mathbb{R}^{C \times H \times W} \rightarrow \text{intermediate feature map}$$

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

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

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

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

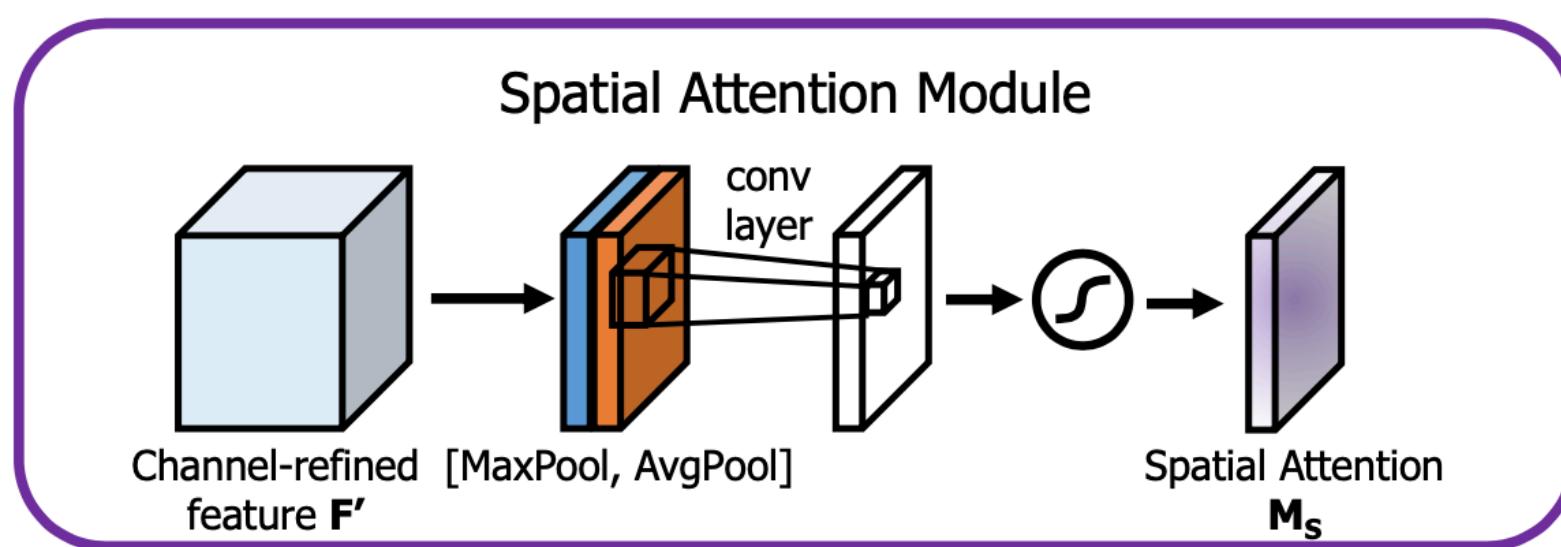


$$M_c(F) = \sigma(MLP(\text{AvgPool}(F)) + MLP(\text{MaxPool}(F)))$$

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

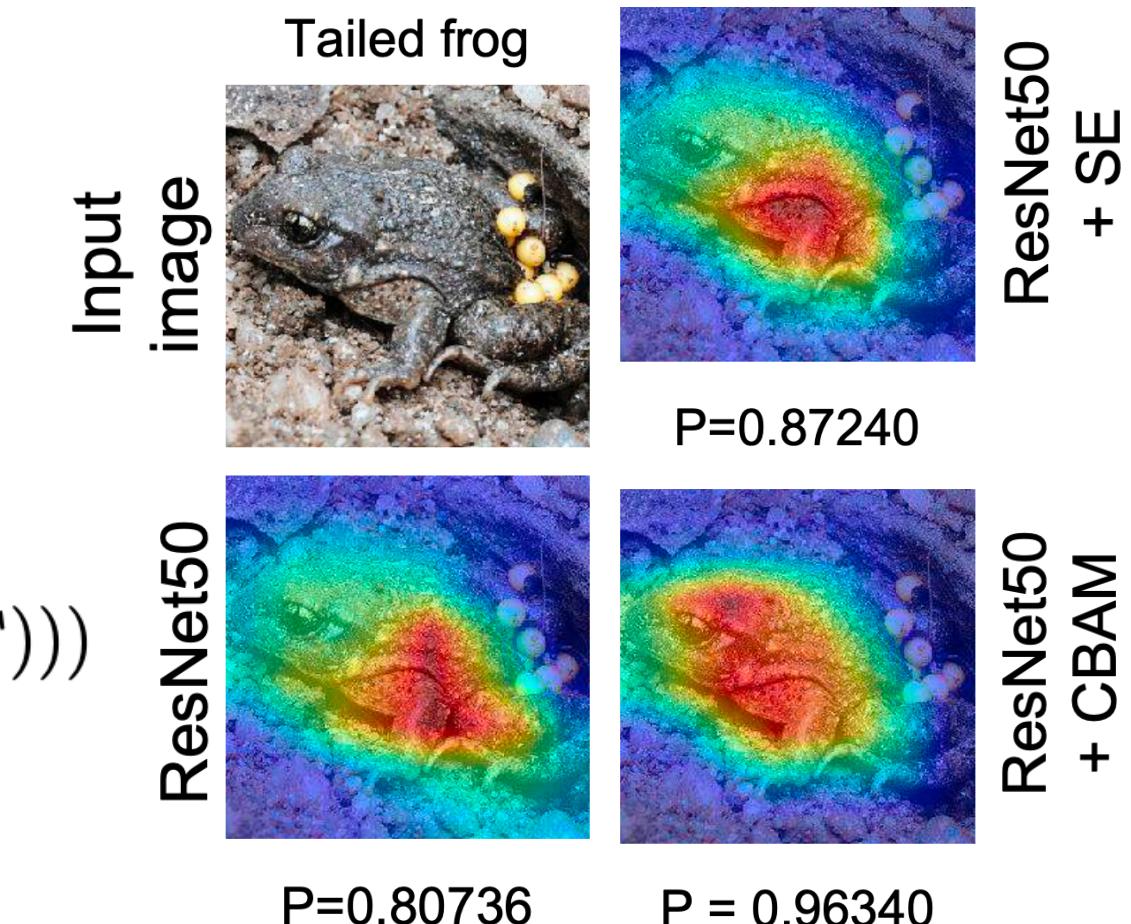
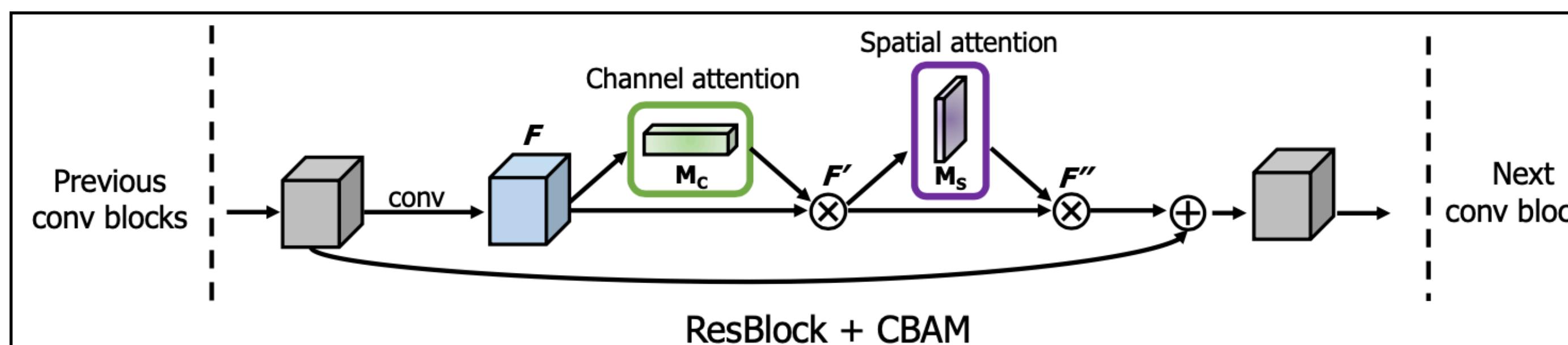
$$\mathbf{W}_0 \in \mathbb{R}^{C/r \times C}, \text{ and } \mathbf{W}_1 \in \mathbb{R}^{C \times C/r}$$

$r \rightarrow$  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

# 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  $I^*$ .

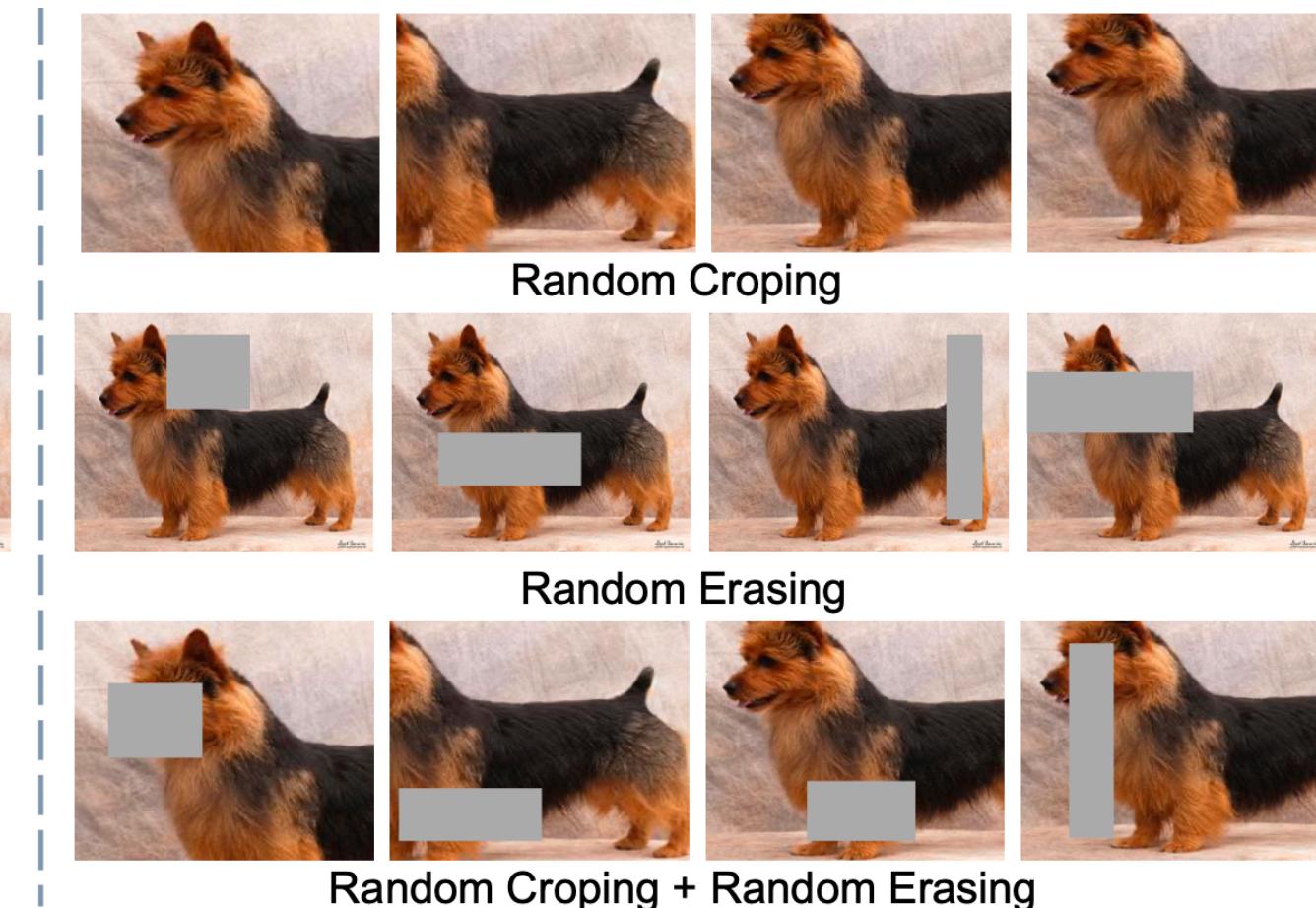
**Initialization:**  $p_1 \leftarrow \text{Rand}(0, 1)$ .

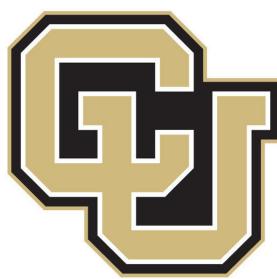
```

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	$19.27 \pm 0.30$	$18.84 \pm 0.18$



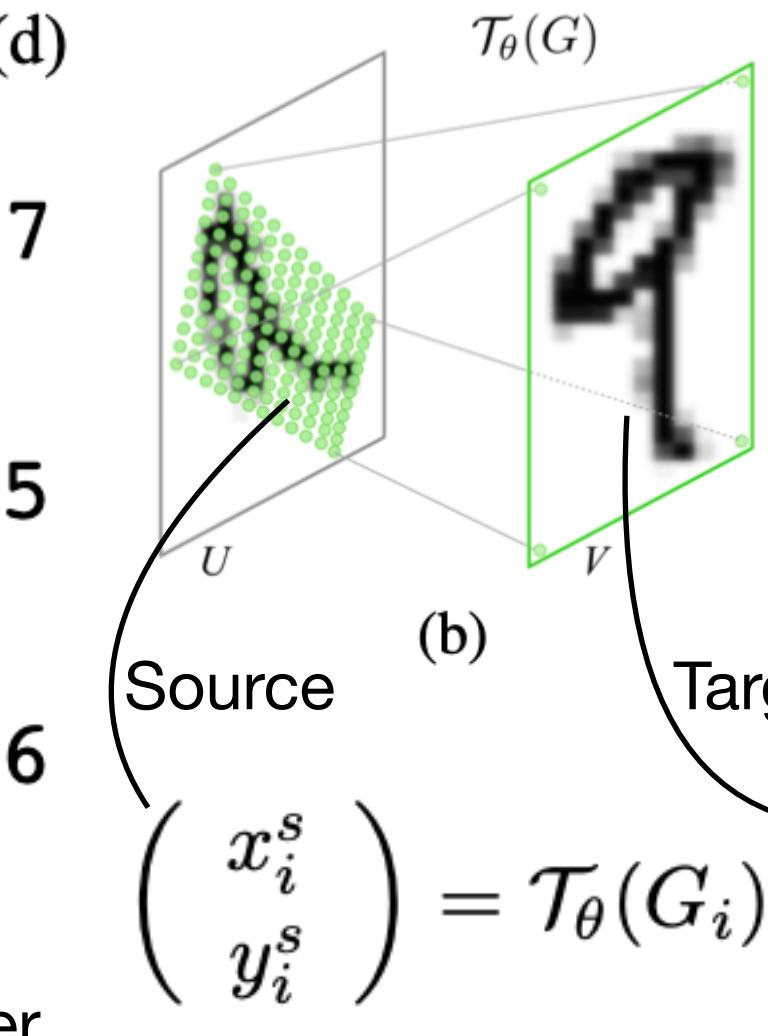
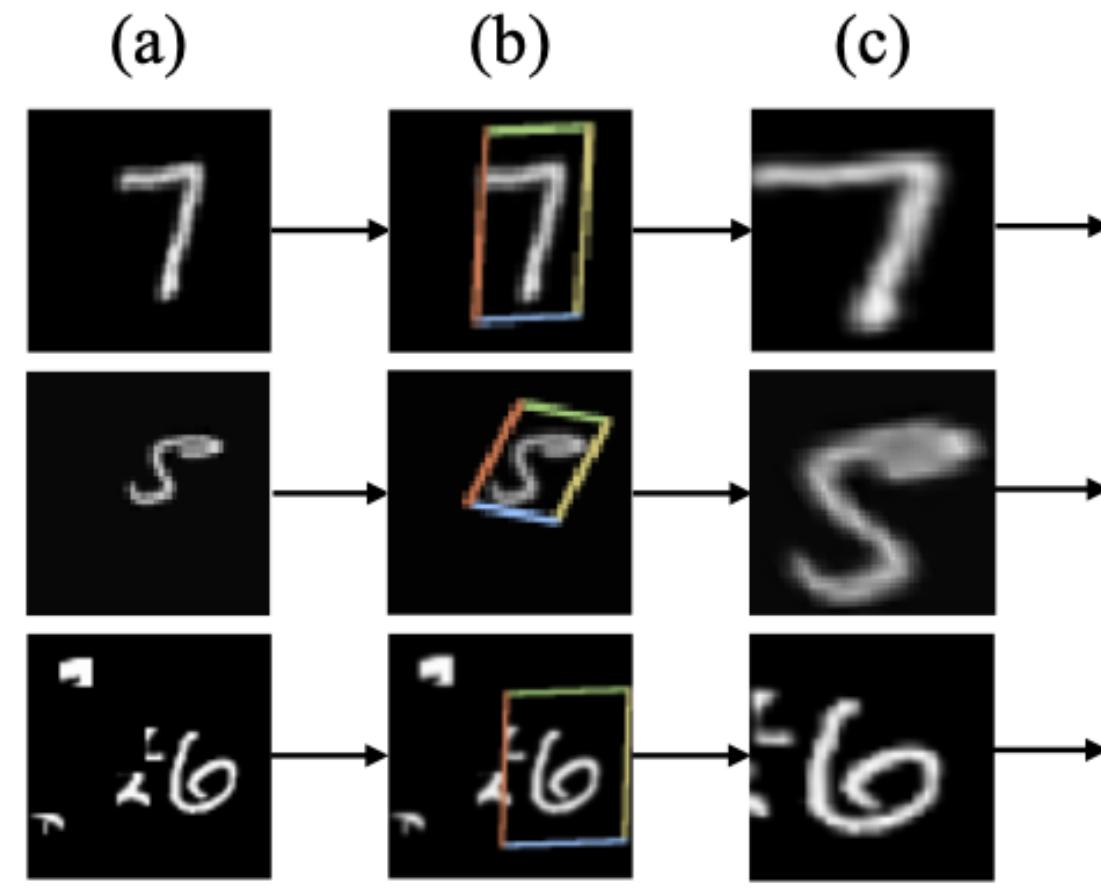


Boulder

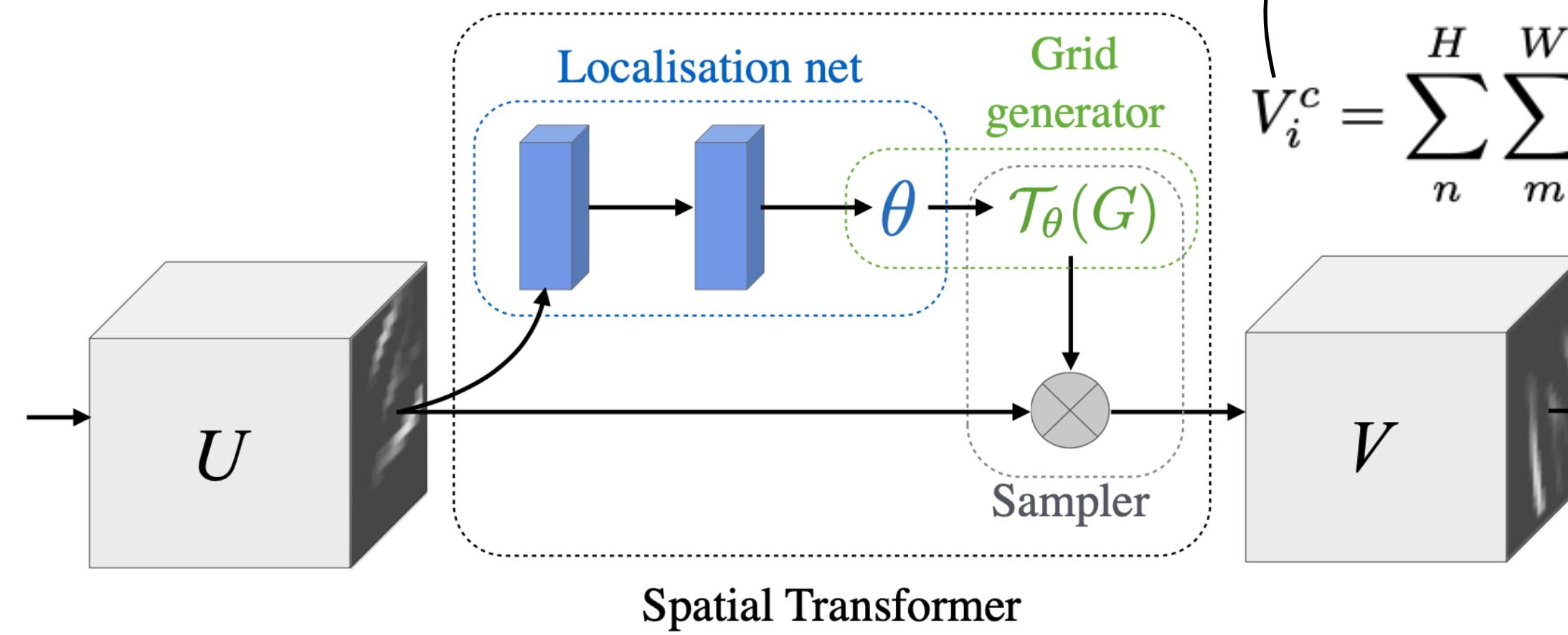


YouTube Video

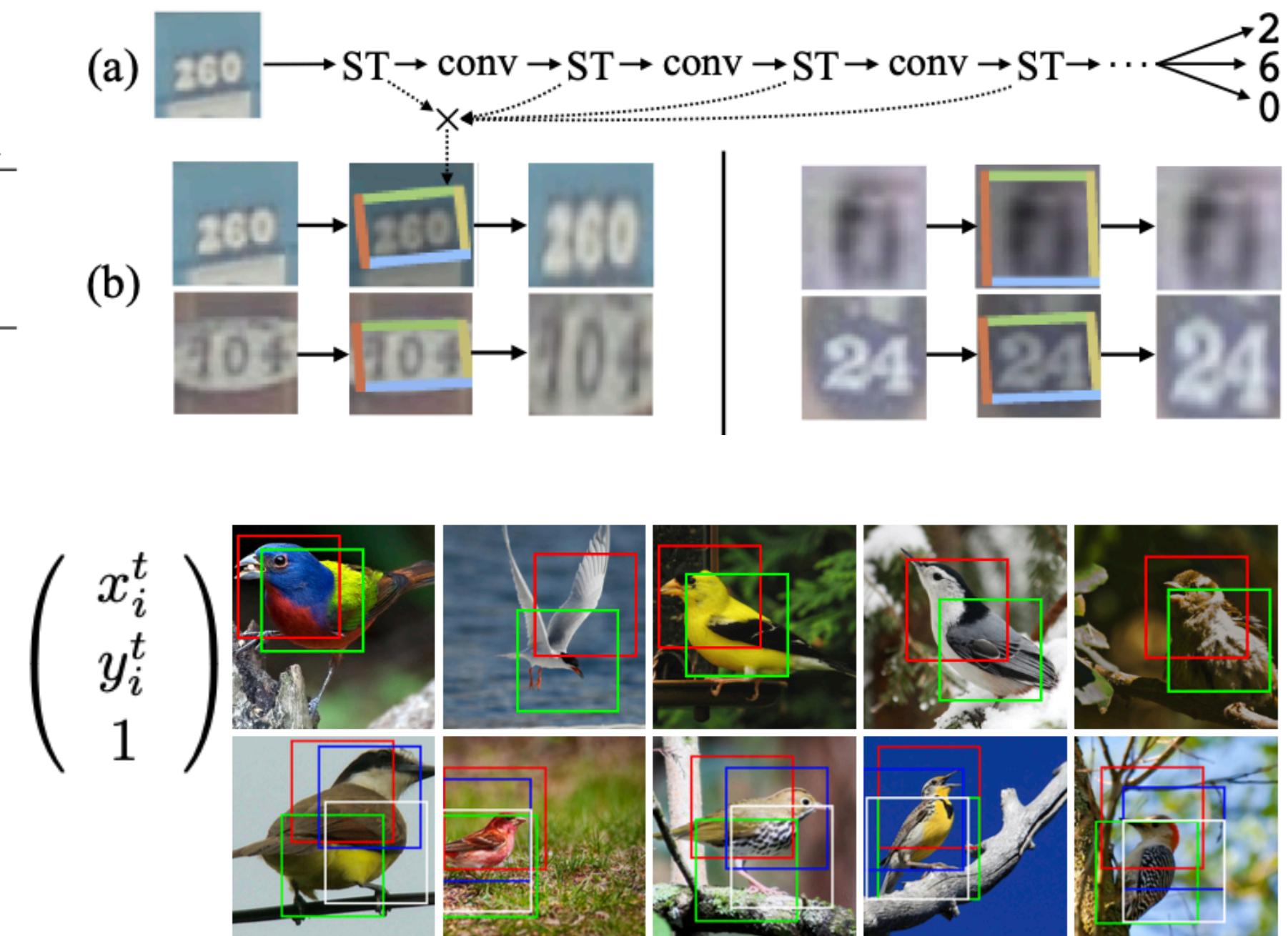
# Spatial Transformer Networks



Bilinear sampling



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>



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

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

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



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  $j$

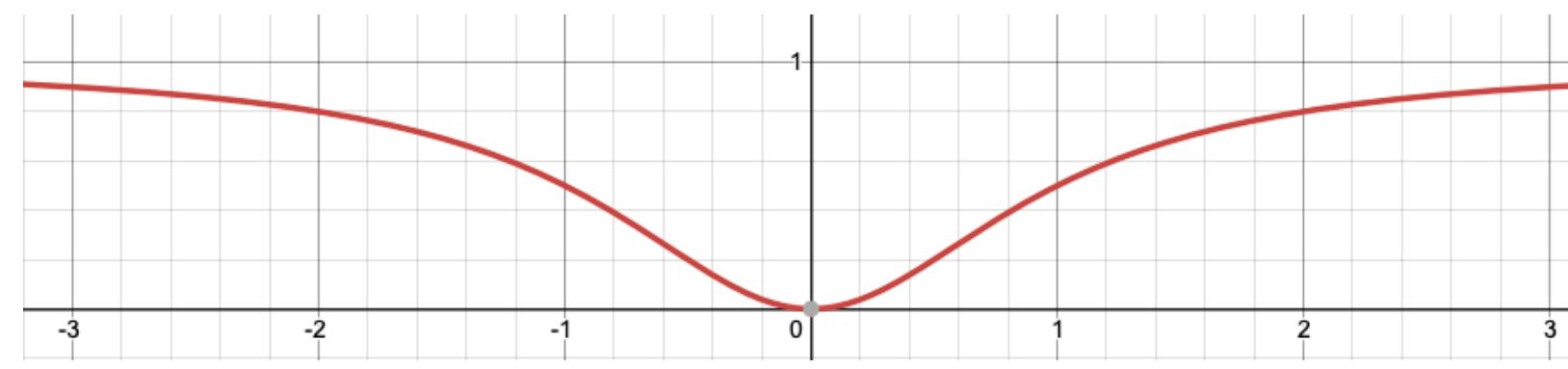
$\|v_j\|$  → probability that the entity represented by the capsule is present in the current input

$\frac{v_j}{\|v_j\|}$  → 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

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

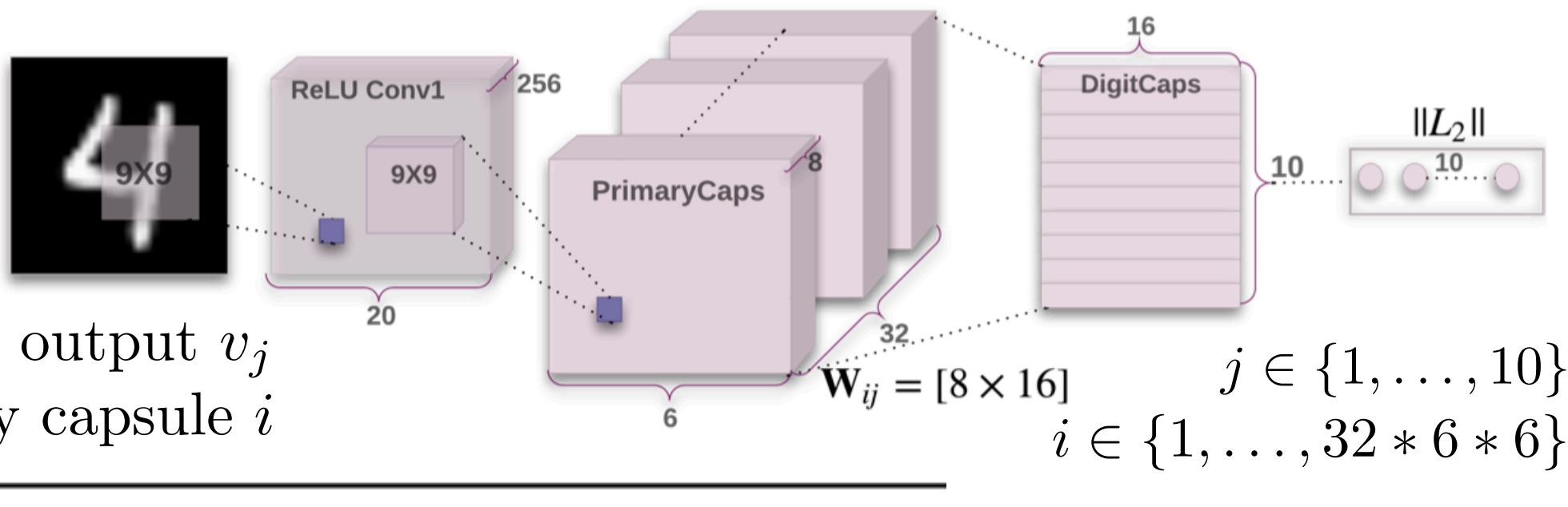
$u_{j|i}$  → prediction vectors from the capsules in the layer below

$u_i$  → 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  $j$

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

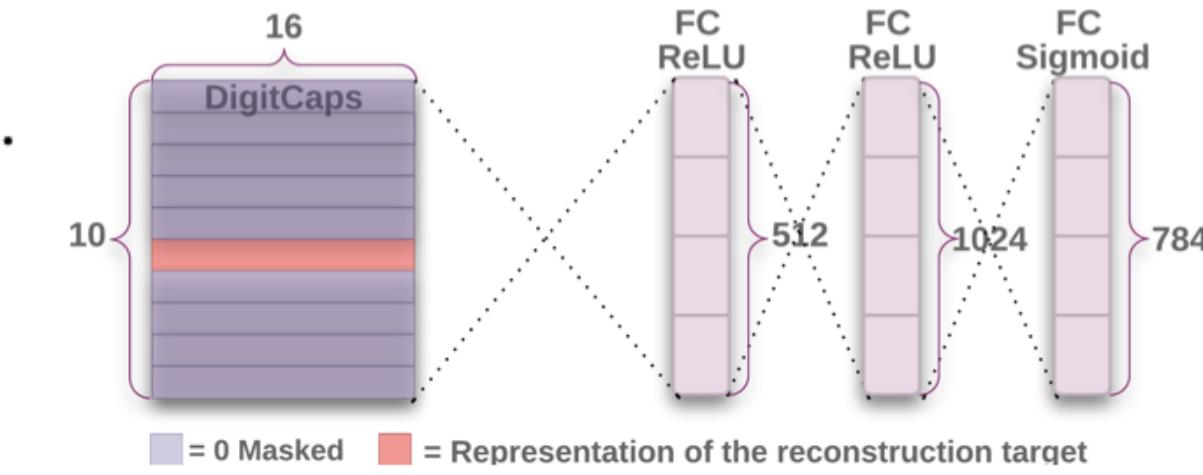


## Procedure 1 Routing algorithm.

```

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

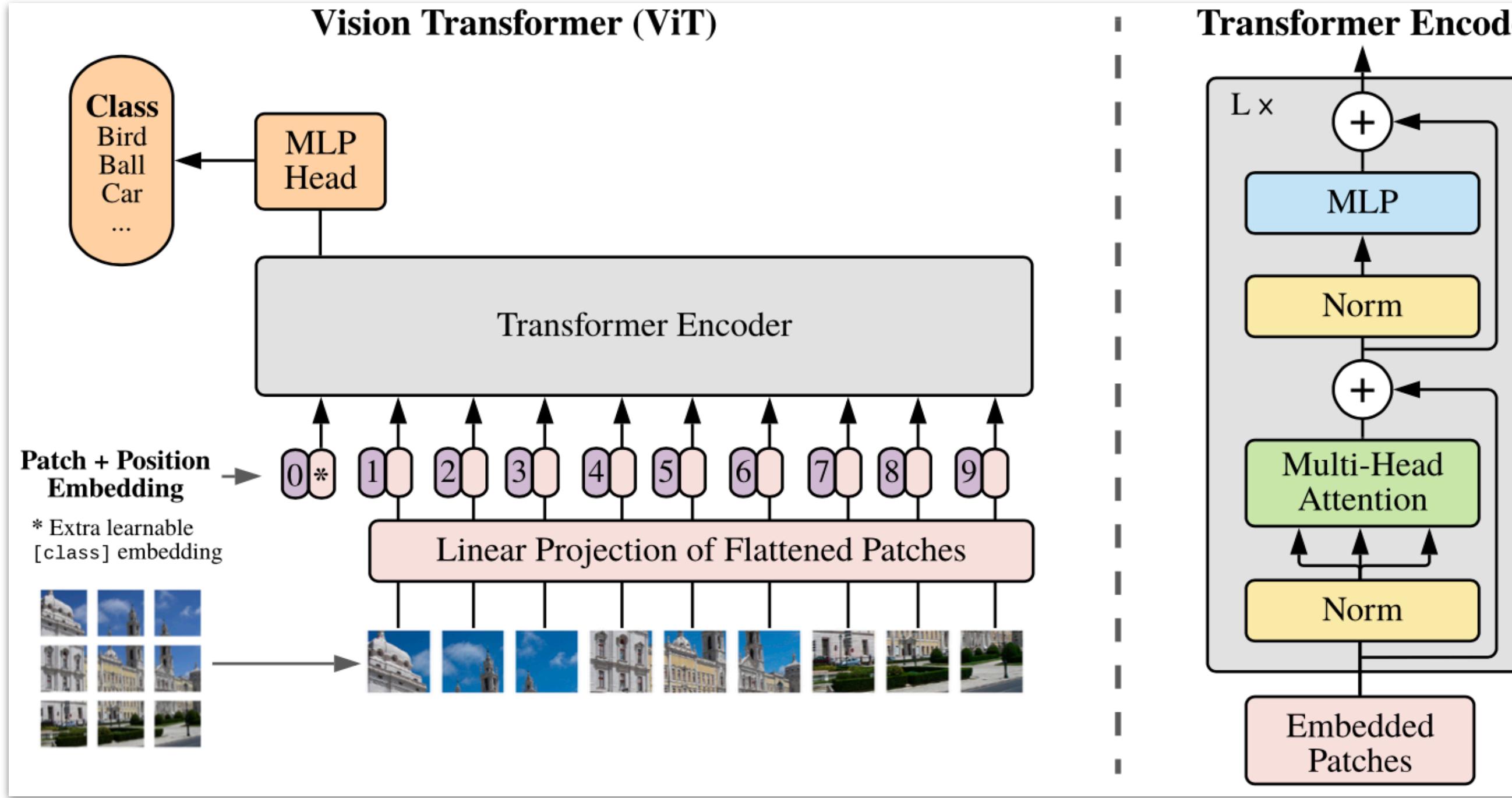


Image Patches  $\equiv$  Tokens (Words) in NLP

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

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

$P \times P \rightarrow$  resolution of each image patch

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

$D \rightarrow$  latent vector size

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

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

$\text{LN} \rightarrow$  layer normalization  
 $\text{MSA} \rightarrow$  multiheaded self-attention

$$z \in \mathbb{R}^{N \times D}$$

$$[q, k, v] = z U_{qkv}, U_{qkv} \in \mathbb{R}^{D \times 3D_h}$$

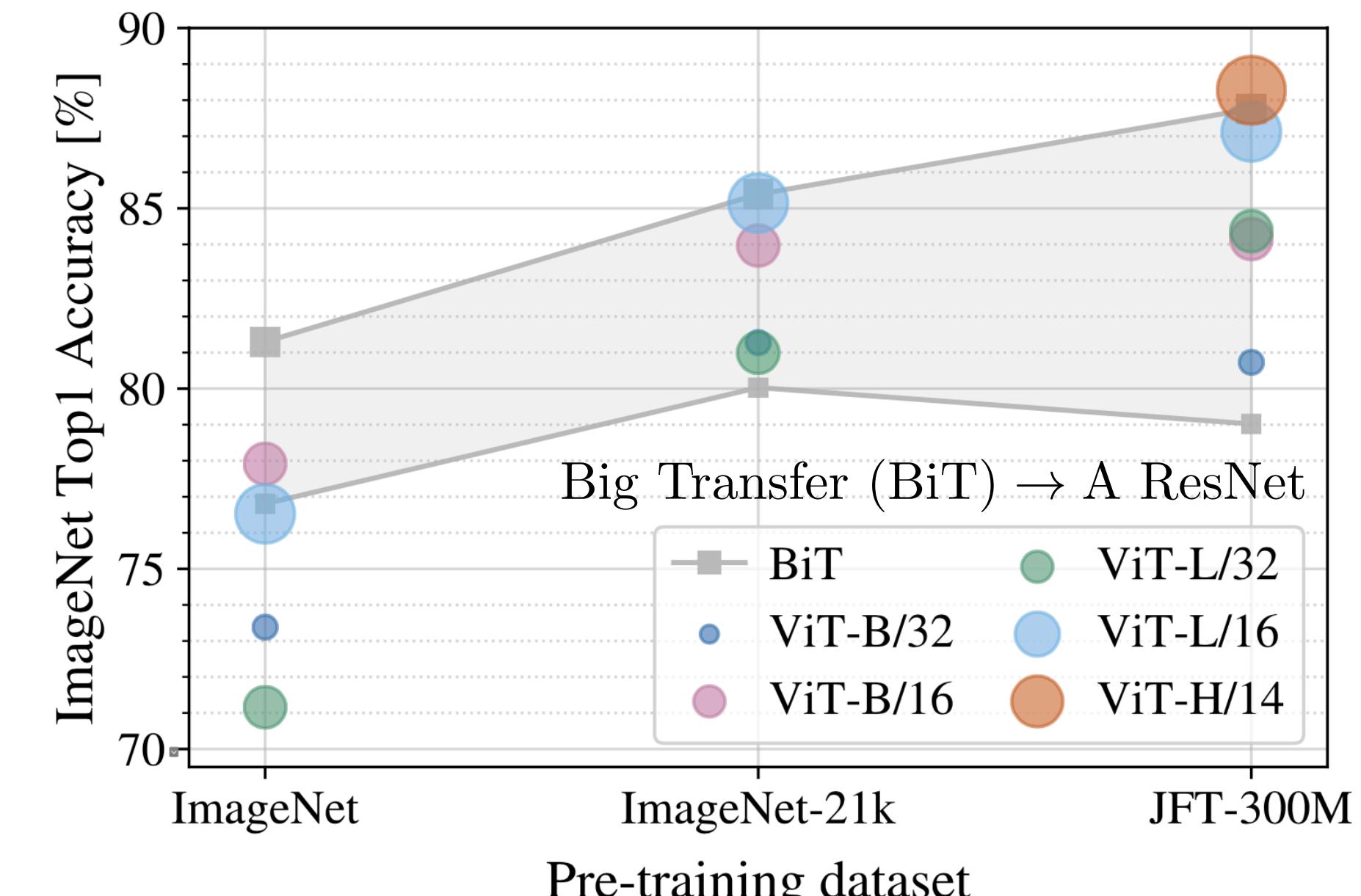
$$A = \text{softmax}(qk^T / \sqrt{D_h}), A \in \mathbb{R}^{N \times N}$$

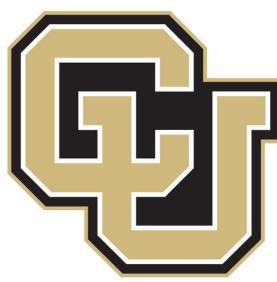
$$\text{SA}(z) = Av$$

$$\text{MSA}(z) = [SA_1(z); SA_2(z); \dots; SA_k(z)]U_{msa}, U_{msa} \in \mathbb{R}^{kD_h \times D}$$

$$D_h = D/k$$

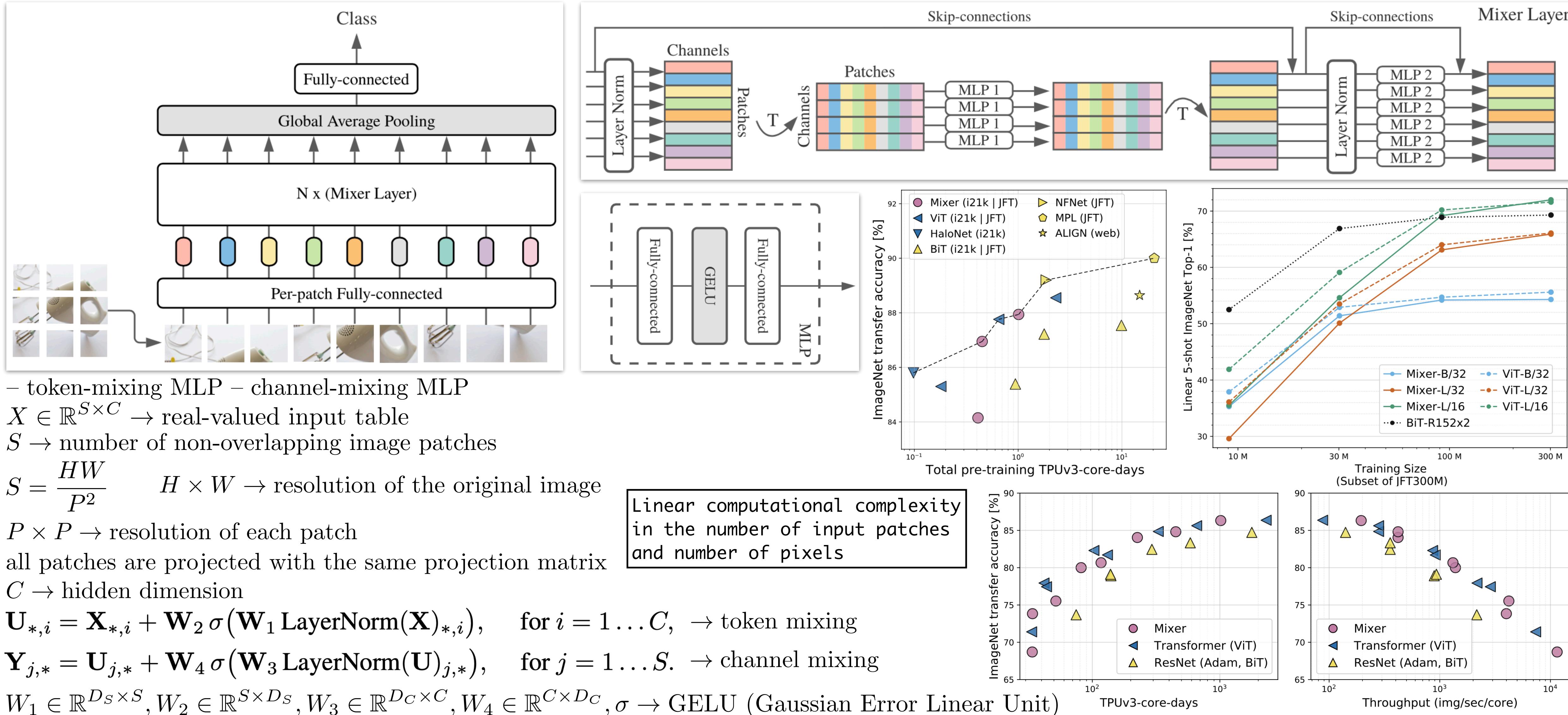
Inductive Bias in CNNs: translation equivariance and locality





Boulder

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





Boulder



# Questions?

[YouTube Playlist](#)

---