Computer Vision; Image Classification; Large Networks

YouTube Playlist

Maziar Raissi

Assistant Professor

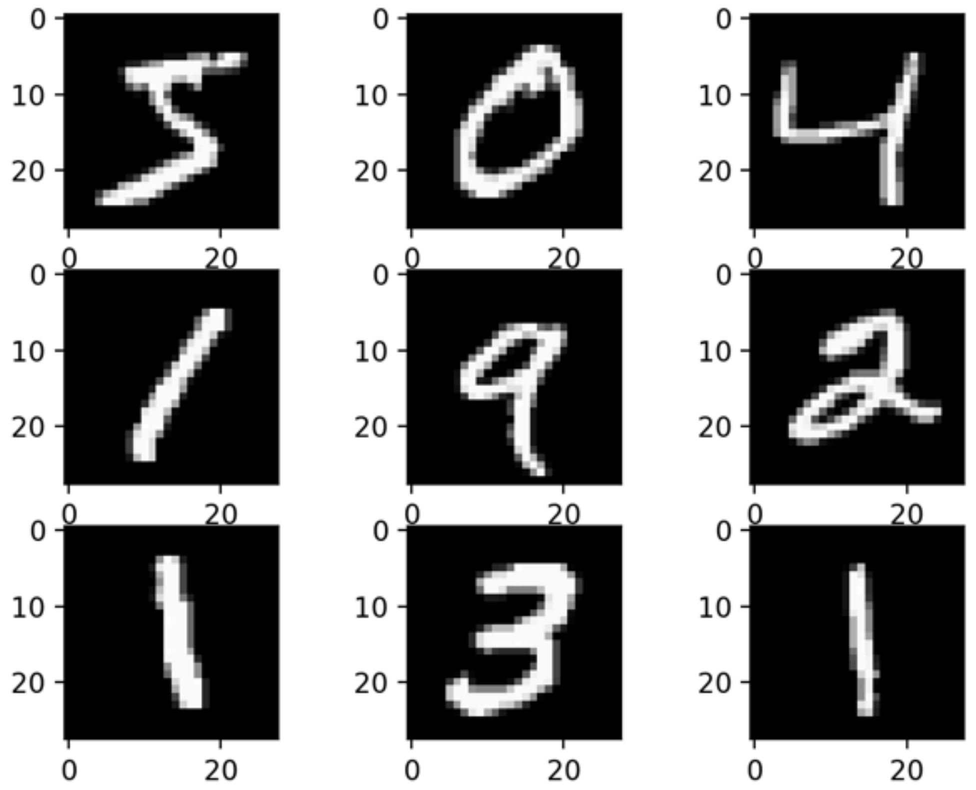
Department of Applied Mathematics

University of Colorado Boulder

maziar.raissi@colorado.edu

Introduction to Convolutional Neural Networks

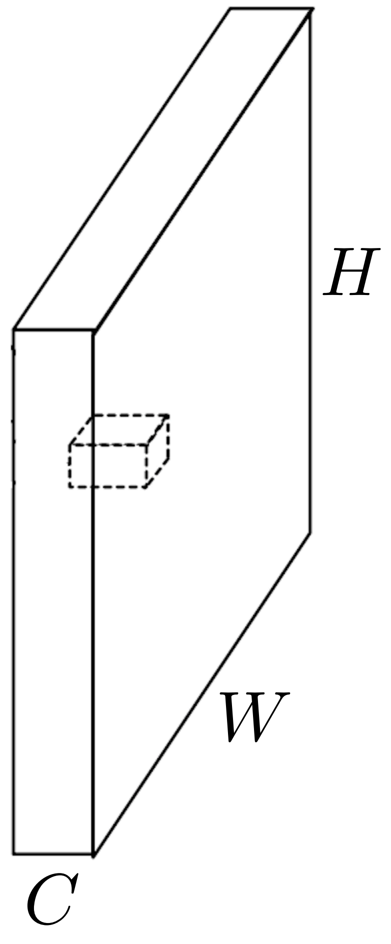
Boulder



Question: What is an image?

Answer: An image is a tensor.

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



$C = 3$ (RGB)

$C = 1$ (Black & White)

Question: What is a pixel?

Answer: Each pixel is a vector.

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

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

$X = \mathrm{X}/255 \rightarrow$ normalize

Training Data

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

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

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

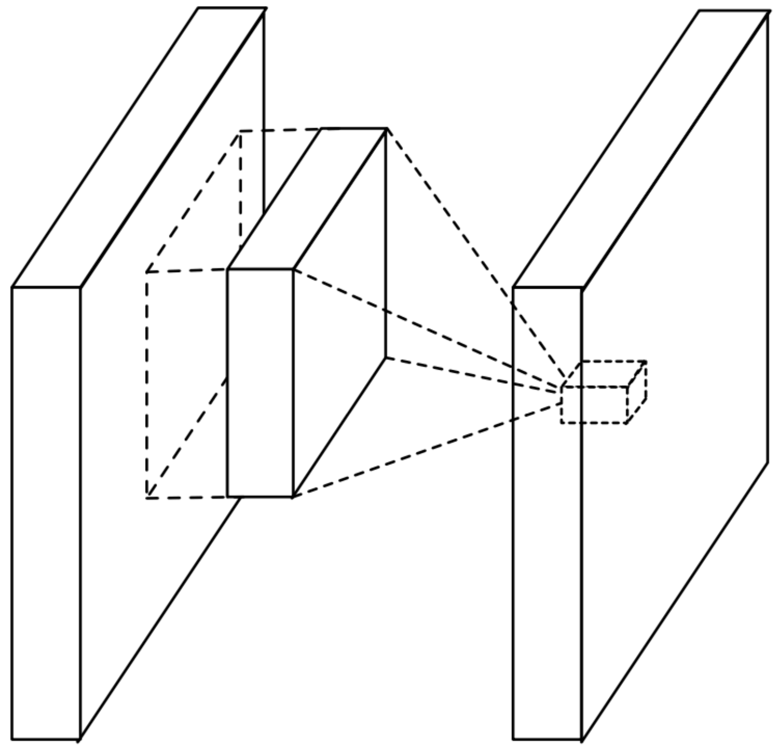
Feature Learning

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

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

$g_\theta \rightarrow$ CNN

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



1×1 Convolution

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

3×3 Convolution

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

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

no additional parameters

Vector Representation

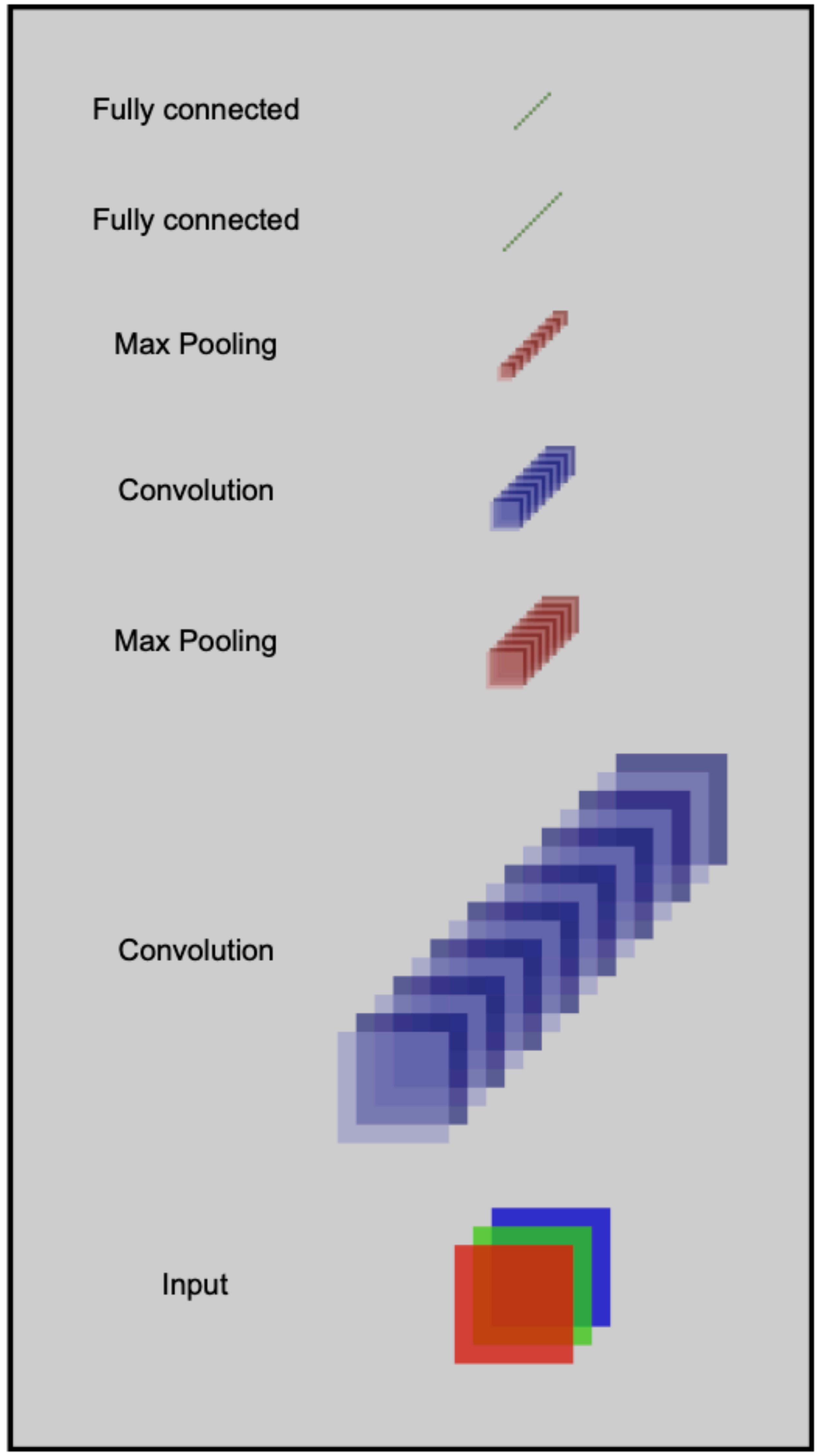
$x \rightarrow$ global average pooling

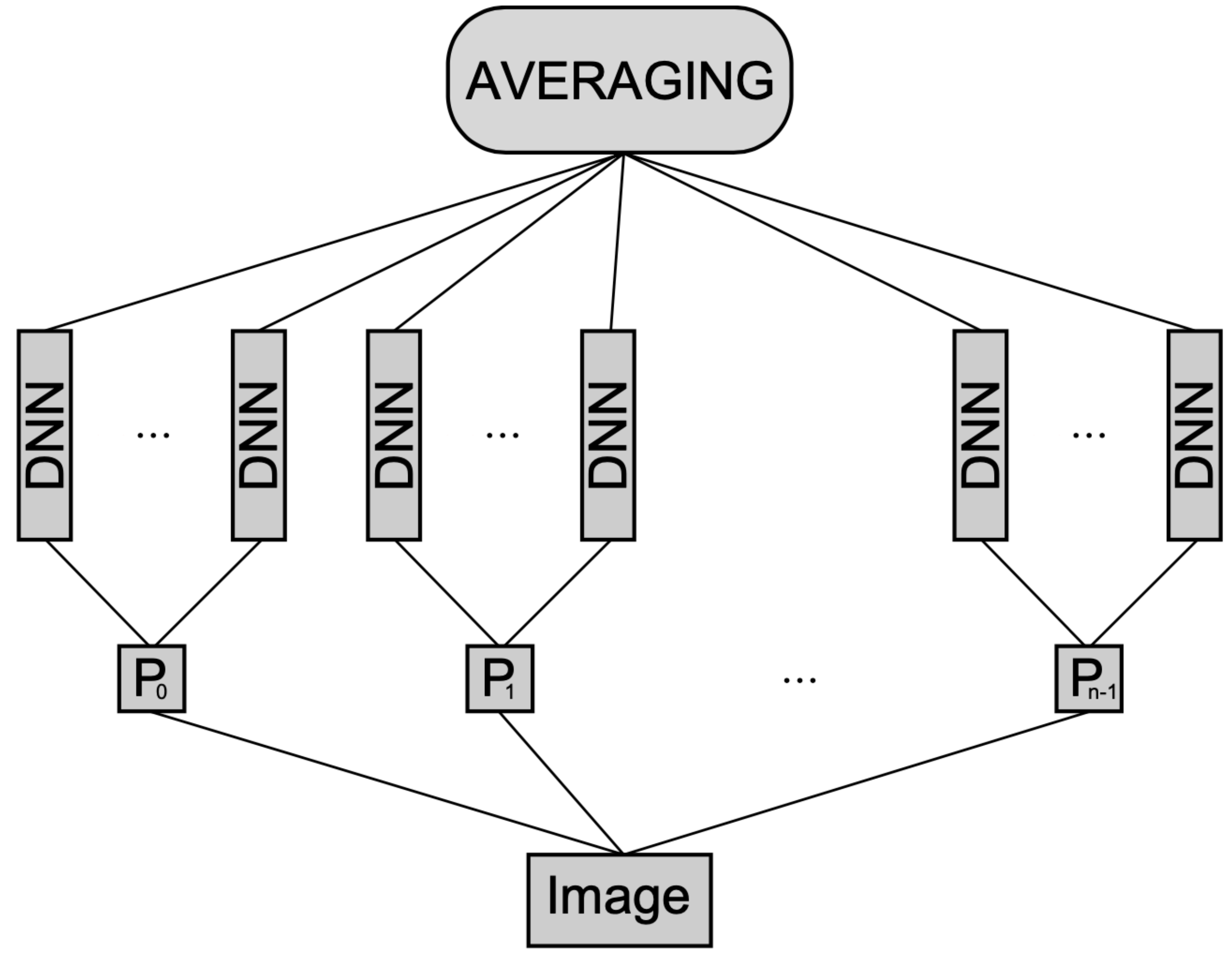
Multi-Layer Perceptron (MLP)

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

Play Dough

Multi-column Deep Neural Networks for Image Classification



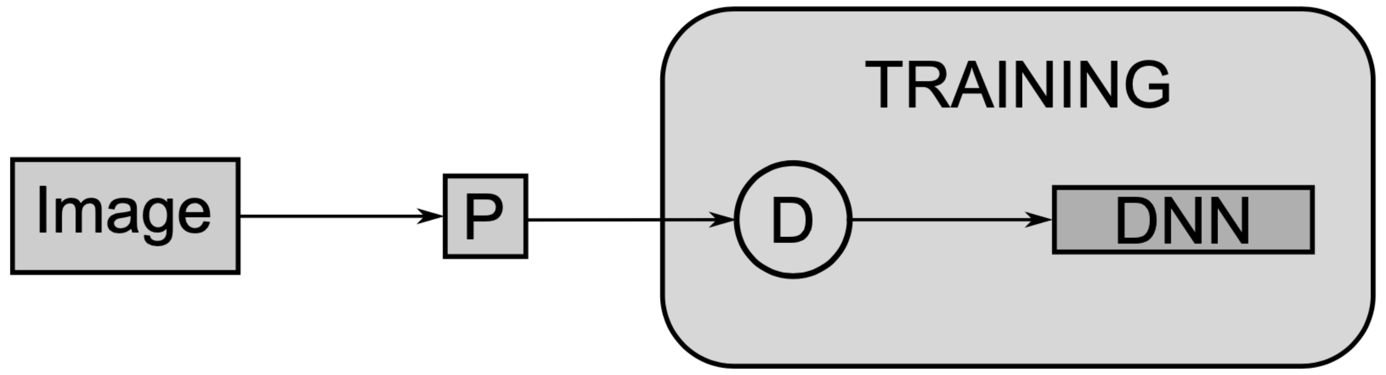


input image can be preprocessed by $P_0 - P_{n-1}$ blocks

$$y^i_{MCDNN} = \frac{1}{N} \sum_{j}^{\#columns} y^i_{DNN_j}$$

prediction of column j

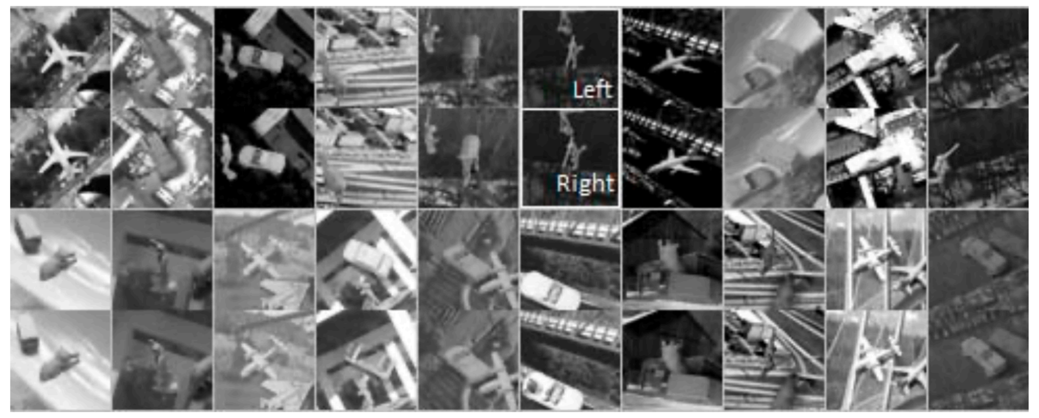
i corresponds to the i^{th} class



dataset is preprocessed before training
every epoch, the images are distorted (D block)

Results on MNIST dataset.

Method	Paper	Error rate[%]
CNN	[32]	0.40
CNN	[26]	0.39
MLP	[5]	0.35
CNN committee	[6]	0.27
MCDNN	this	**0.23**



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)	**11.63**		
digits (10)	**0.77**	3.71 [12]	1.88 [23]
letters (52)	**21.01**	30.91[16]	
letters* (26)	**7.37**	13.00 [4]	13.66[16]
merged (37)	**7.99**		
uppercase (26)	**1.83**	10.00 [4]	6.44 [9]
lowercase (26)	**7.47**	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

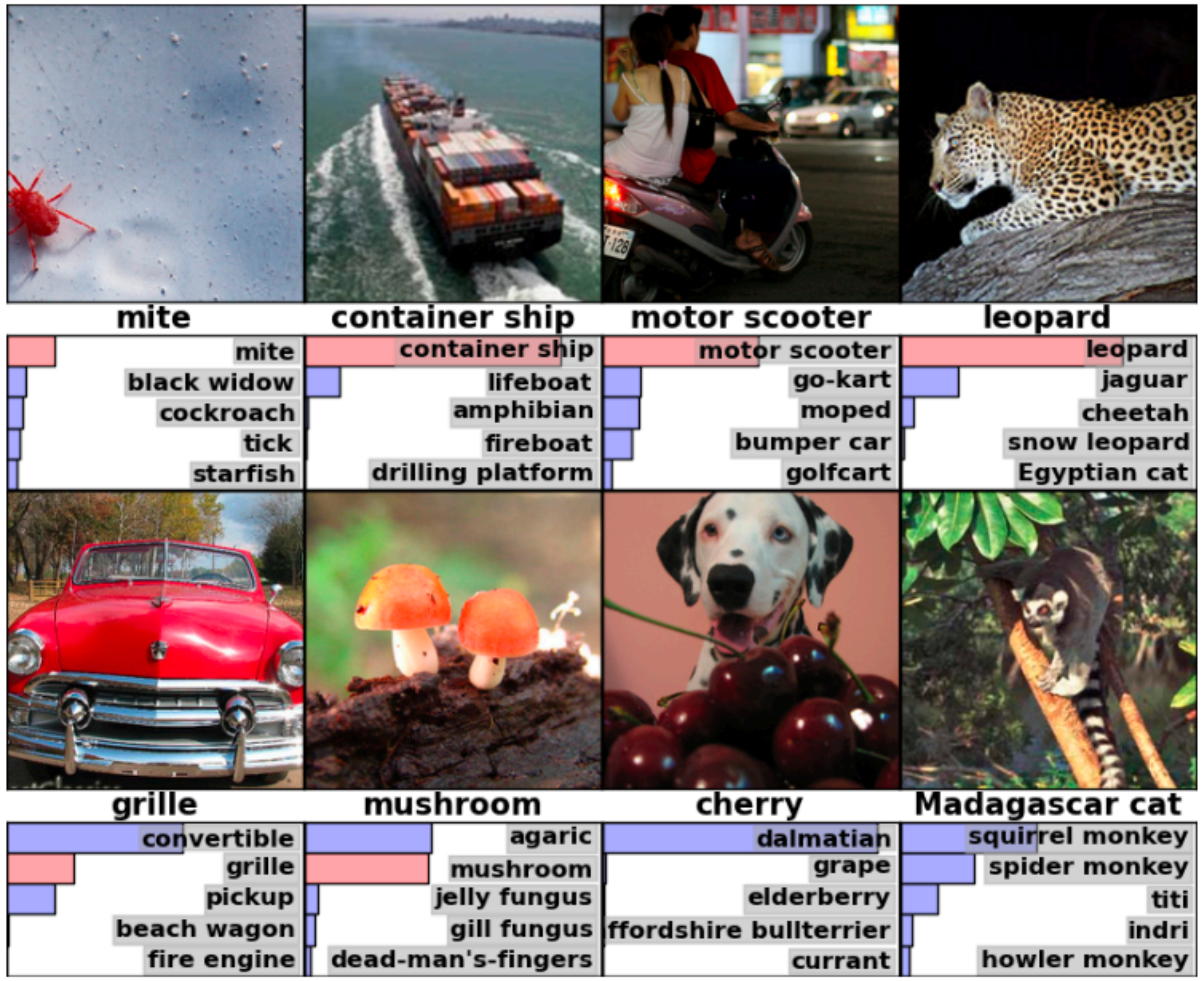
→ Chinese characters

Ciregan, Dan, Ueli Meier, and Jürgen Schmidhuber. "Multi-column deep neural networks for image classification." *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012.

Boulder

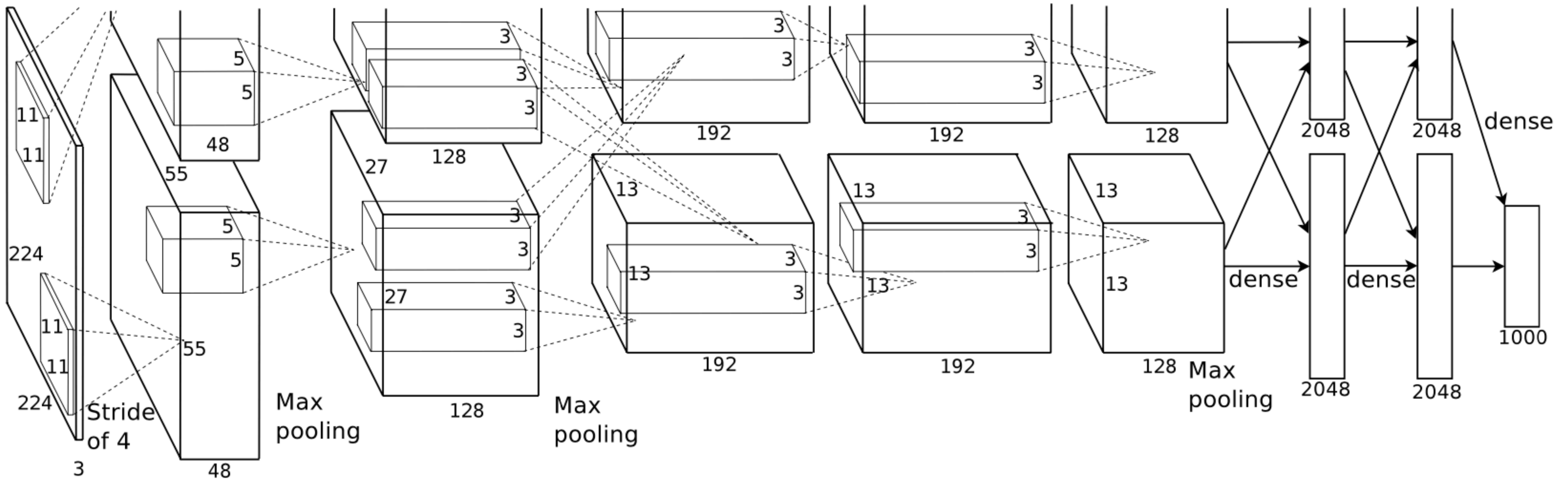
ImageNet Classification with Deep Convolutional Neural Networks

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: $I \in \mathbb{R}^{224\times224\times3} \mapsto p \in \{p \in \mathbb{R}^{1000} : \sum_i p_i = 1, p_i \geq 0 \ \forall i\}$

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

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

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

3x3 Conv: $Y(x,y) = \sum_{i=1,2,3}\sum_{j=1,2,3} W(i,j)X(s(x-2)+i, s(y-2)+j), W \in \mathbb{R}^{3\times3\times F\times C}$

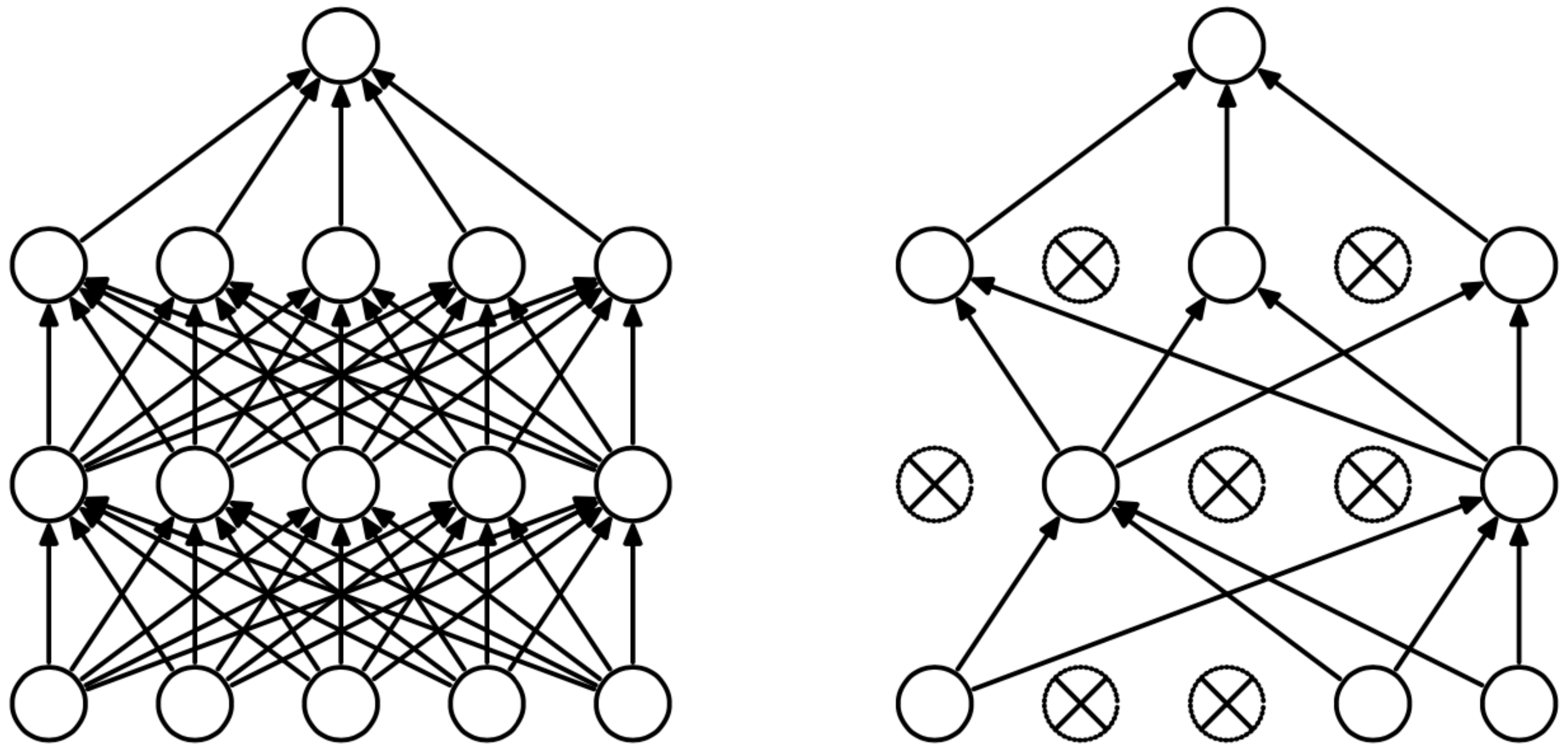
Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

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

Boulder

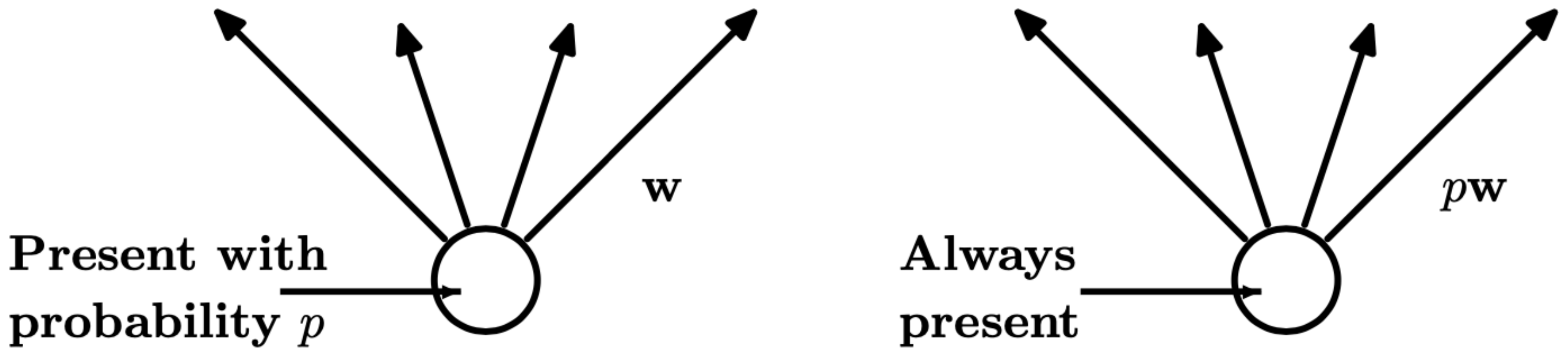
Dropout: A Simple Way to Prevent Neural Networks from Overfitting

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



(a) At training time (b) At test time

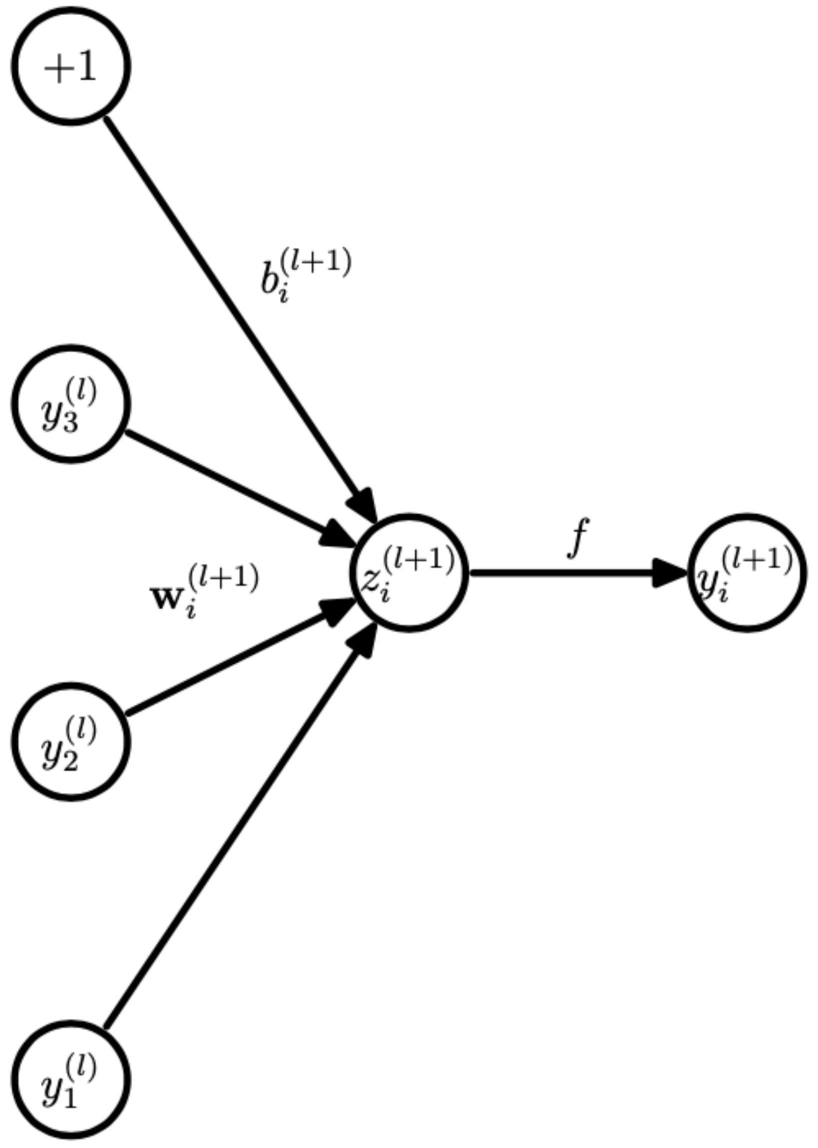
Each hidden unit in a neural network trained with dropout must learn to work with a randomly chosen sample of other units

$L \rightarrow$ number of hidden layers

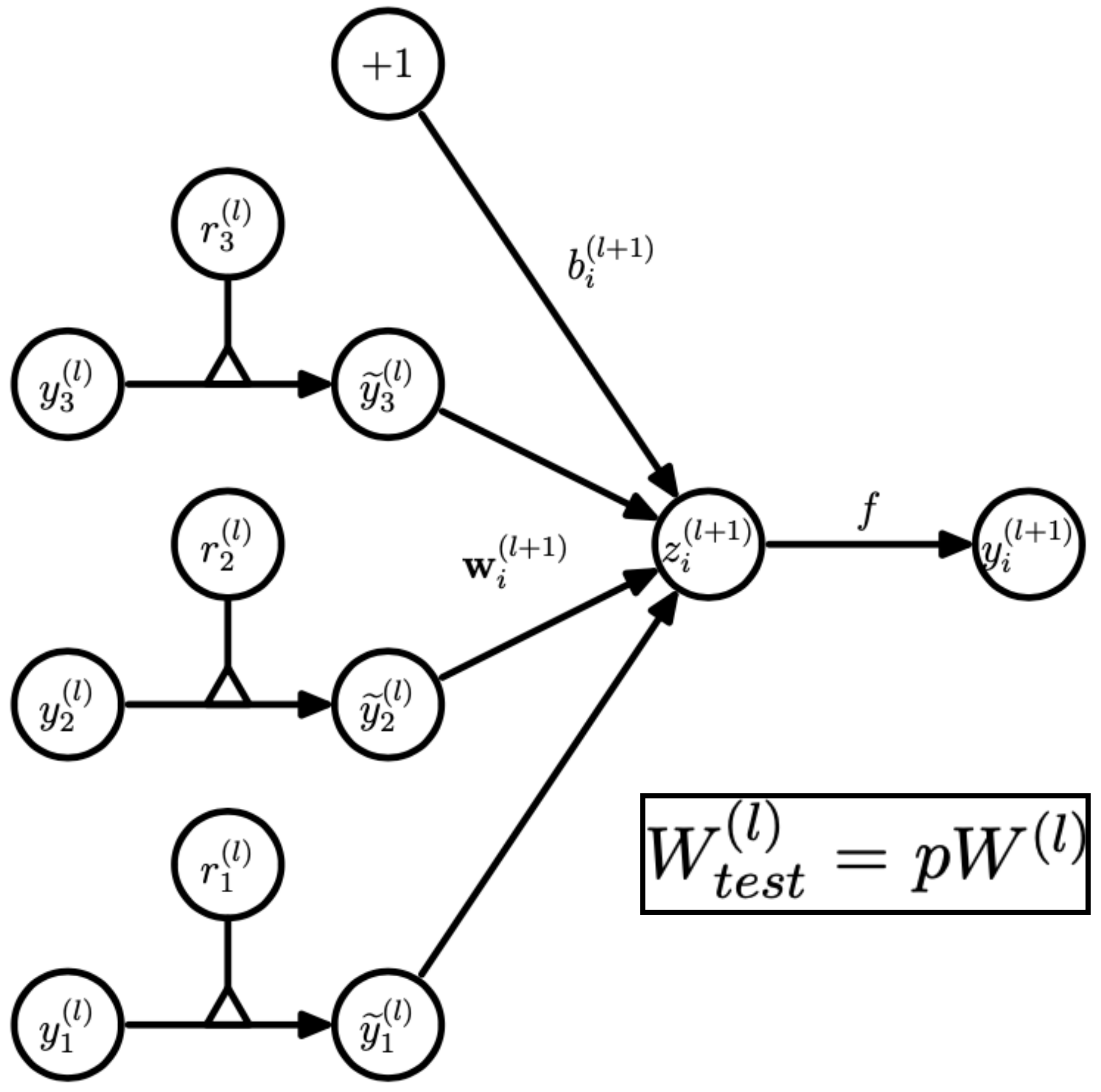
$l \in \{1, \ldots, L\}$

$$\begin{aligned} z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \mathbf{y}^l + b_i^{(l+1)}, \\ y_i^{(l+1)} &= f(z_i^{(l+1)}), \end{aligned}$$

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



(a) Standard network



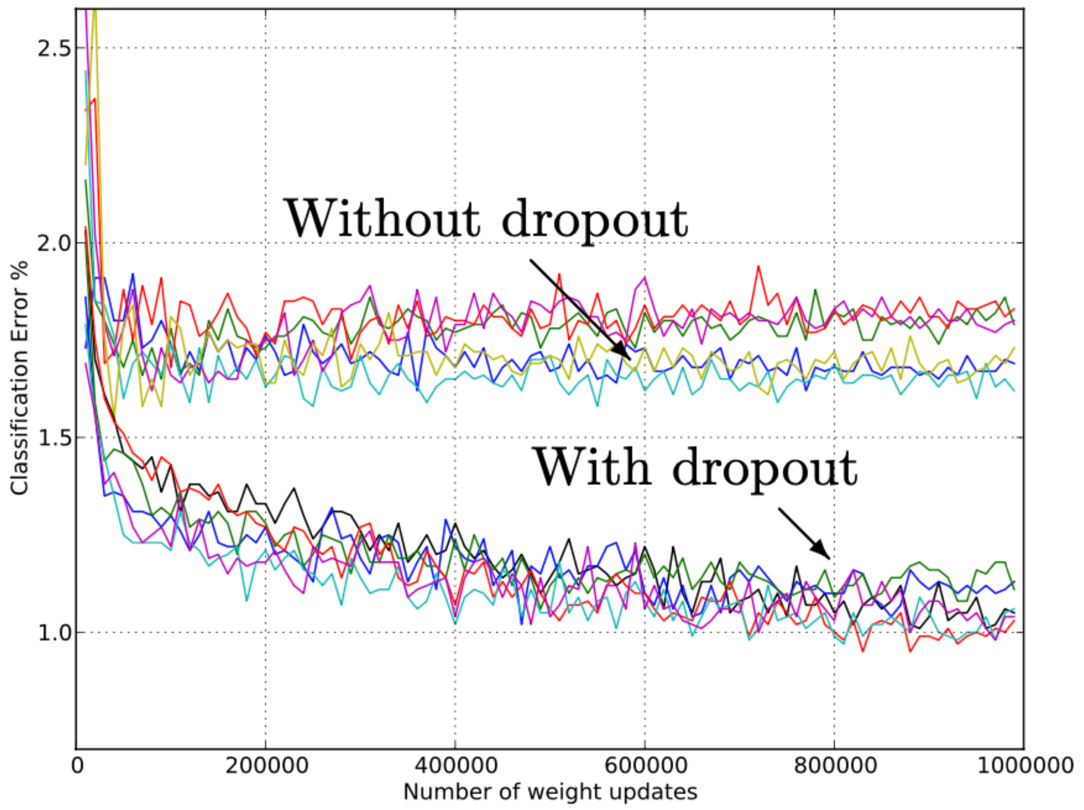
(b) Dropout network

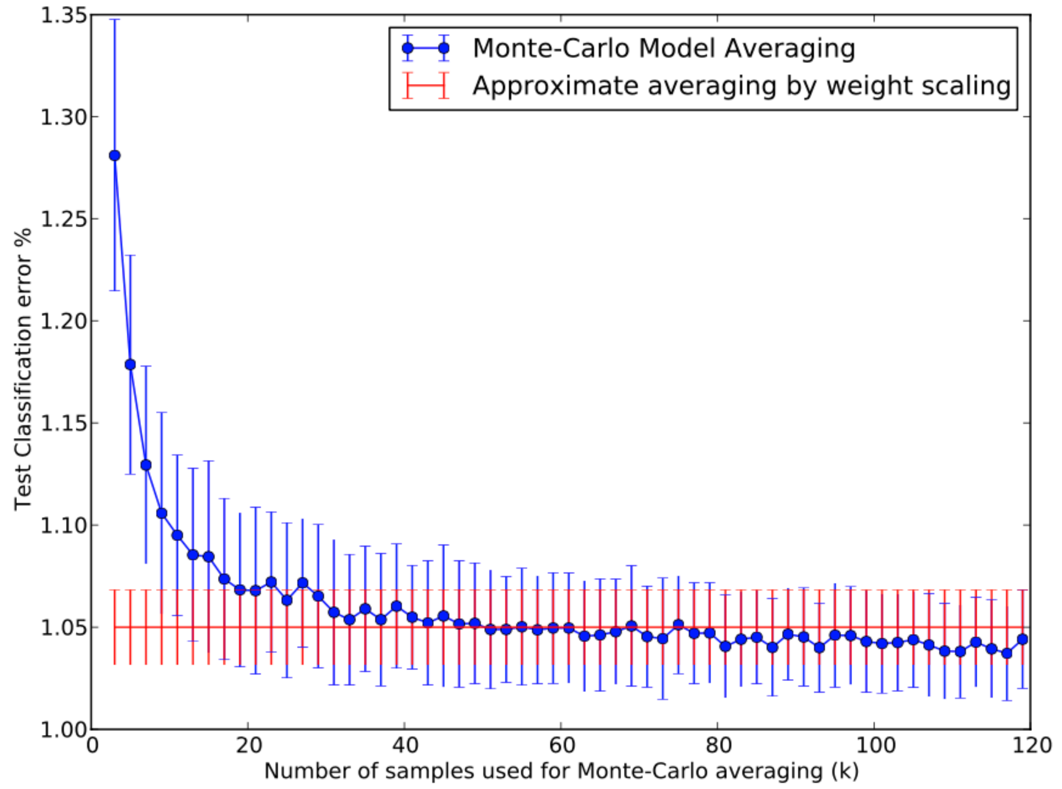
$$W_{test}^{(l)} = pW^{(l)}$$

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

max-norm regularization

Dropout with linear regression is equivalent to ridge regression.





Data Set	Domain	Dimensionality	Training Set	Test Set
MNIST	Vision	784 (28 × 28 grayscale)	60K	10K
SVHN	Vision	3072 (32 × 32 color)	600K	26K
CIFAR-10/100	Vision	3072 (32 × 32 color)	60K	10K
ImageNet (ILSVRC-2012)	Vision	65536 (256 × 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).

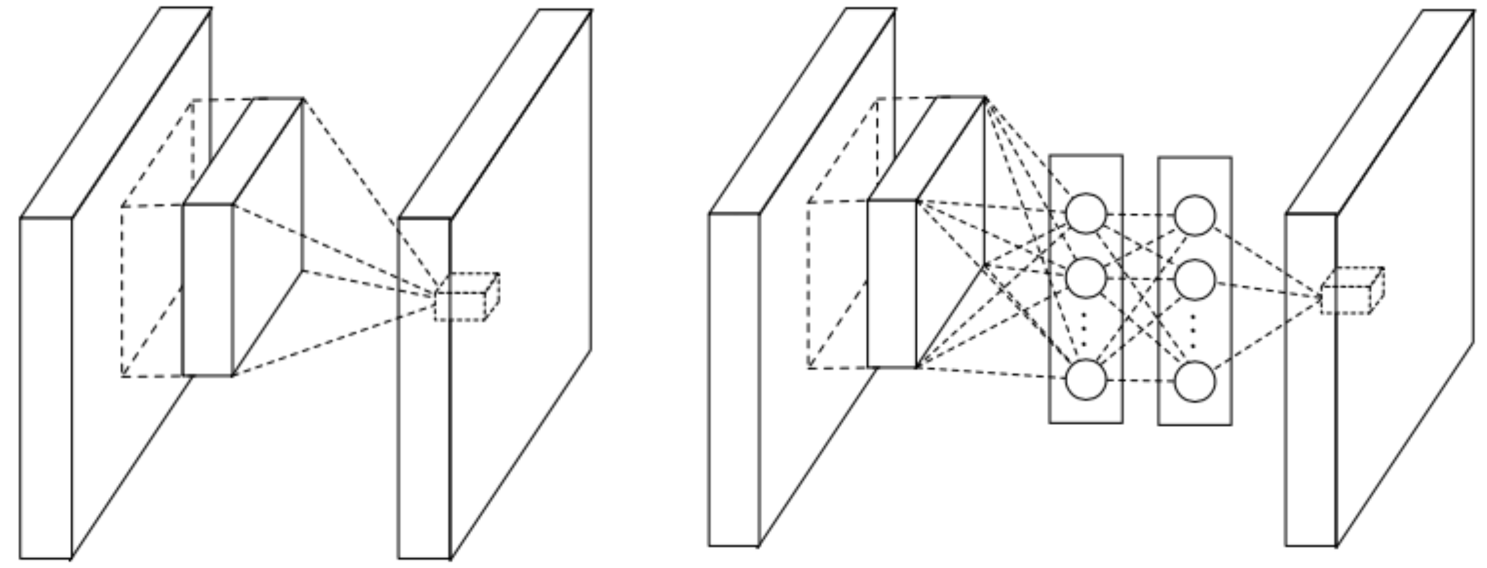
Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *The journal of machine learning research* 15.1 (2014): 1929-1958.

Network In Network

Boulder

YouTube Playlist

1) MLP Convolution Layers (1x1 Convolutions)



(a) Linear convolution layer (b) Mlpconv layer

$$f^1_{i,j,k_1} = \max({w^1_{k_1}}^T x_{i,j} + b_{k_1}, 0).$$

Input patch centered at location (i, j)

ReLU activation function

$$\vdots$$

$$f^n_{i,j,k_n} = \max({w^n_{k_n}}^T f^{n-1}_{i,j} + b_{k_n}, 0).$$

k_ℓ is used to index the channels of the feature map

2) Global Average Pooling

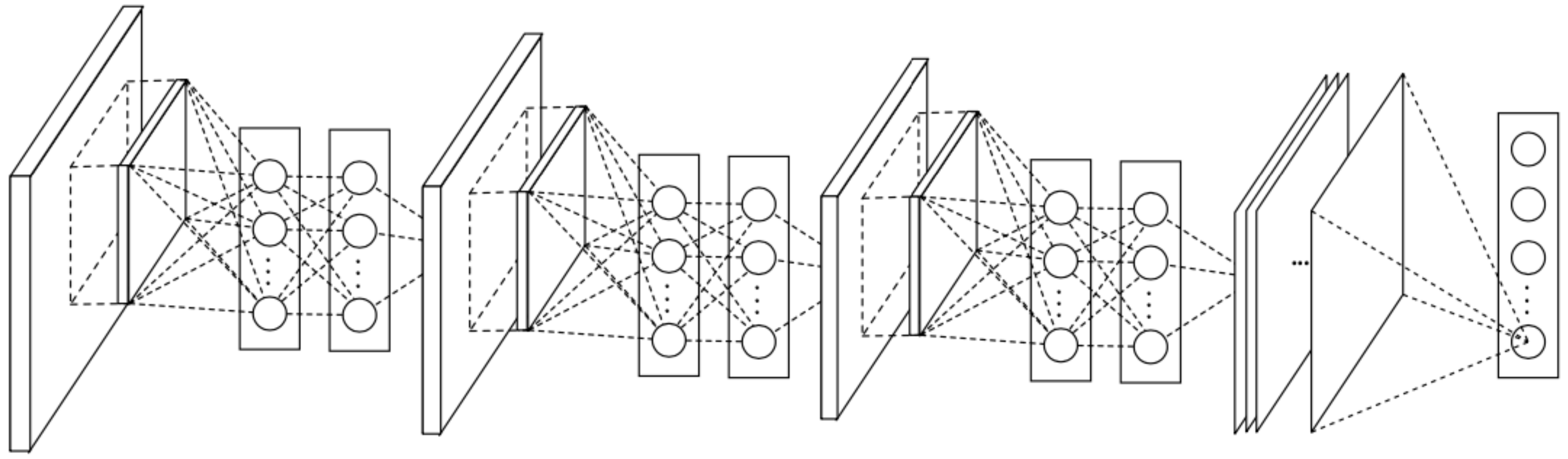


Table 1: Test set error rates for CIFAR-10 of various methods.

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

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%
NIN + Dropout	**35.68%**

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%
DropConnect [15]	**1.94%**

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%
Conv. maxout + Dropout [8]	**0.45%**

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%

Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." *arXiv preprint arXiv:1312.4400* (2013).

Very Deep Convolutional Networks for Large-Scale Image Recognition

Boulder

[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 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Local Response Normalization (LRN):

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

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

response-normalized activity

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)

random variable drawn from a Gaussian with mean zero and standard deviation 0.1

$$I_{xy} = [I^R_{xy}, I^G_{xy}, I^B_{xy}]^T + [p_1, p_2, p_3][\alpha_1\lambda_1, \alpha_2\lambda_2, \alpha_3\lambda_3]^T$$

eigenvector and eigenvalue of the 3 × 3 covariance matrix of RGB pixel values

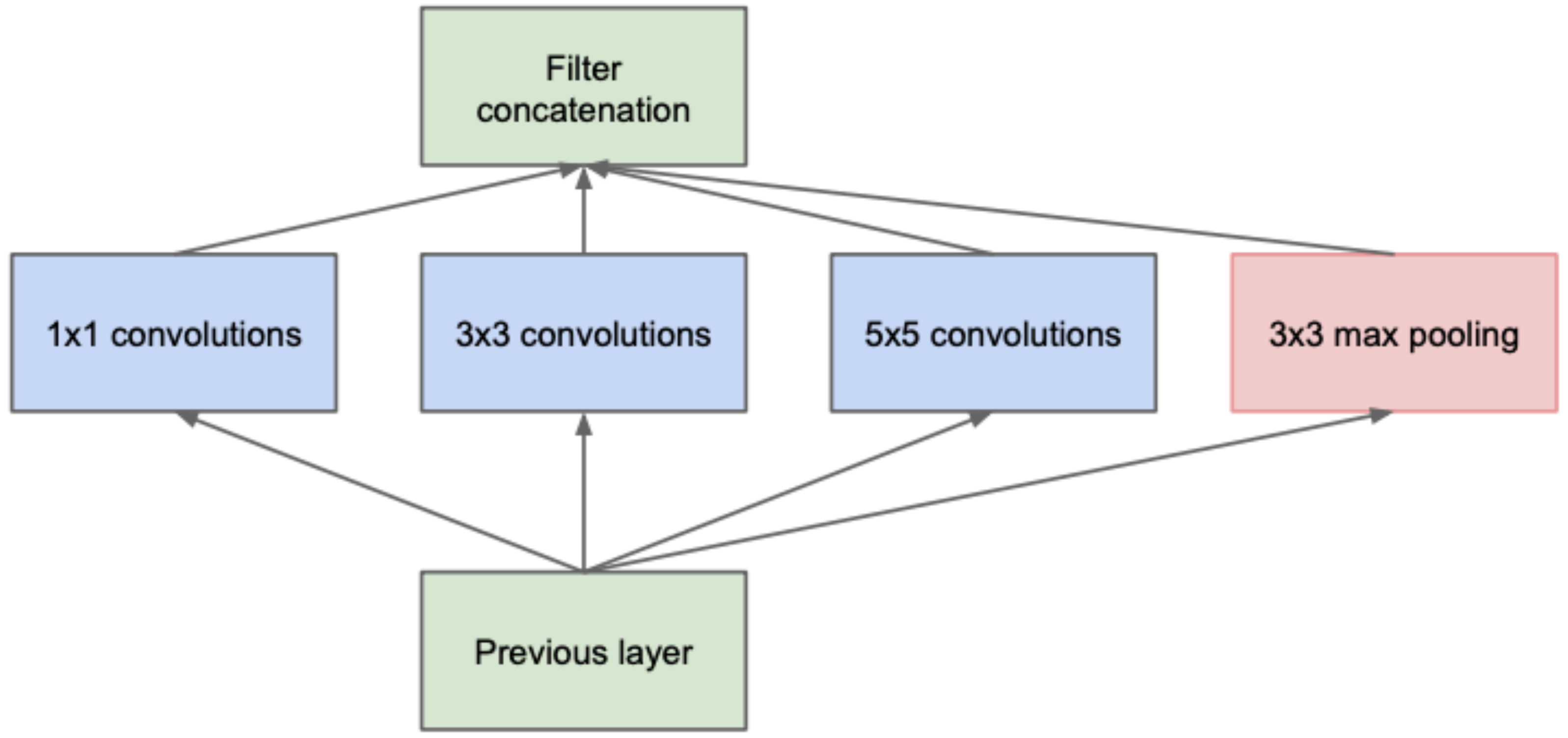
3) Scale jittering

Each training image is individually rescaled by randomly sampling S from a certain range $[S_{\min}, S_{\max}]$

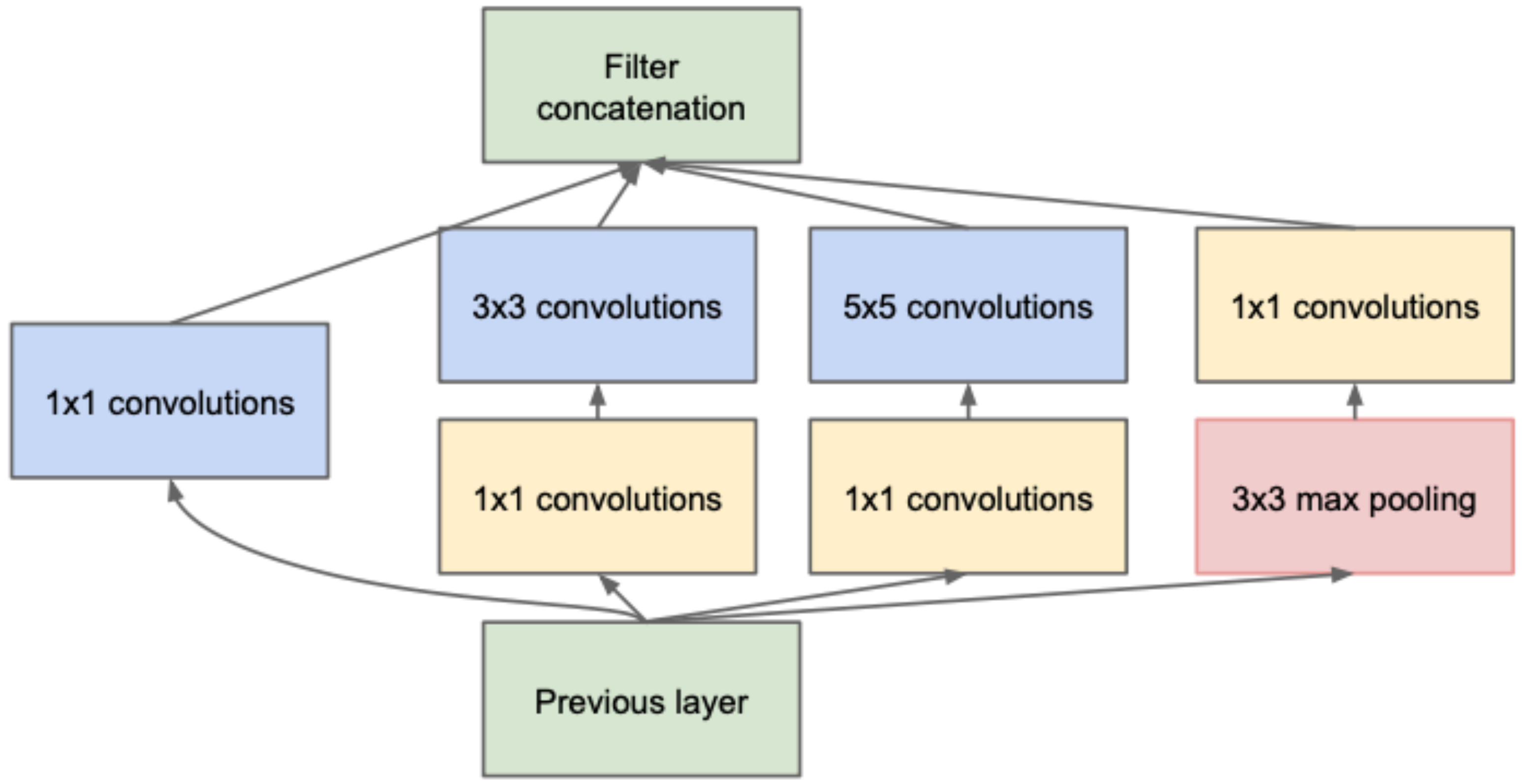
S is the smallest side of an isotropically-rescaled training image.

Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

Going Deeper with Convolutions



(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.

Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

Boulder

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

$z = g(\mathrm{BN}(\underbrace{Wu}_{x}))$

Training

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

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

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

All operations are element-wise (dimension-wise)

The gradients flow through μ and σ^2

Inference

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

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

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

Convolution

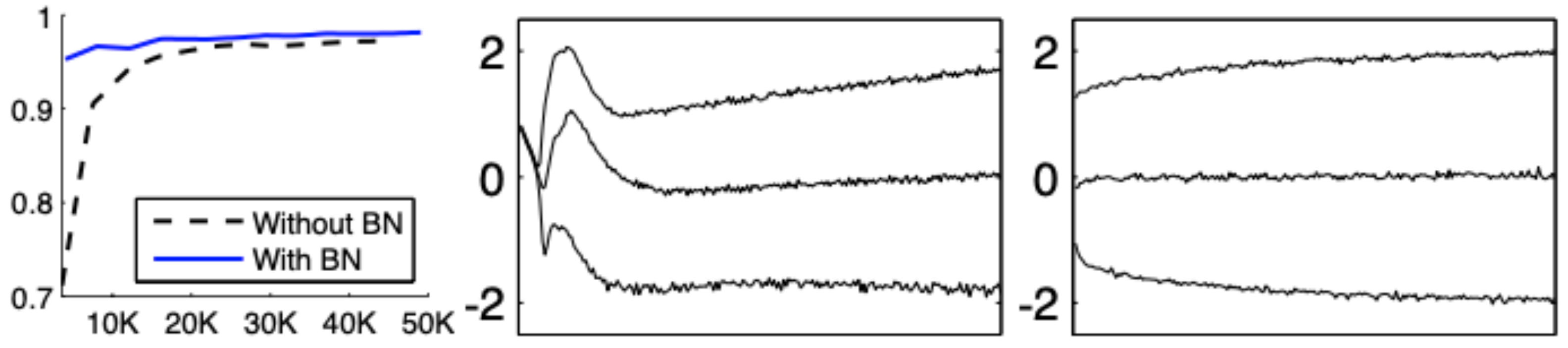
$X \in \mathbb{R}^{p\times q\times d}$

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

Batch-norm is done per feature map

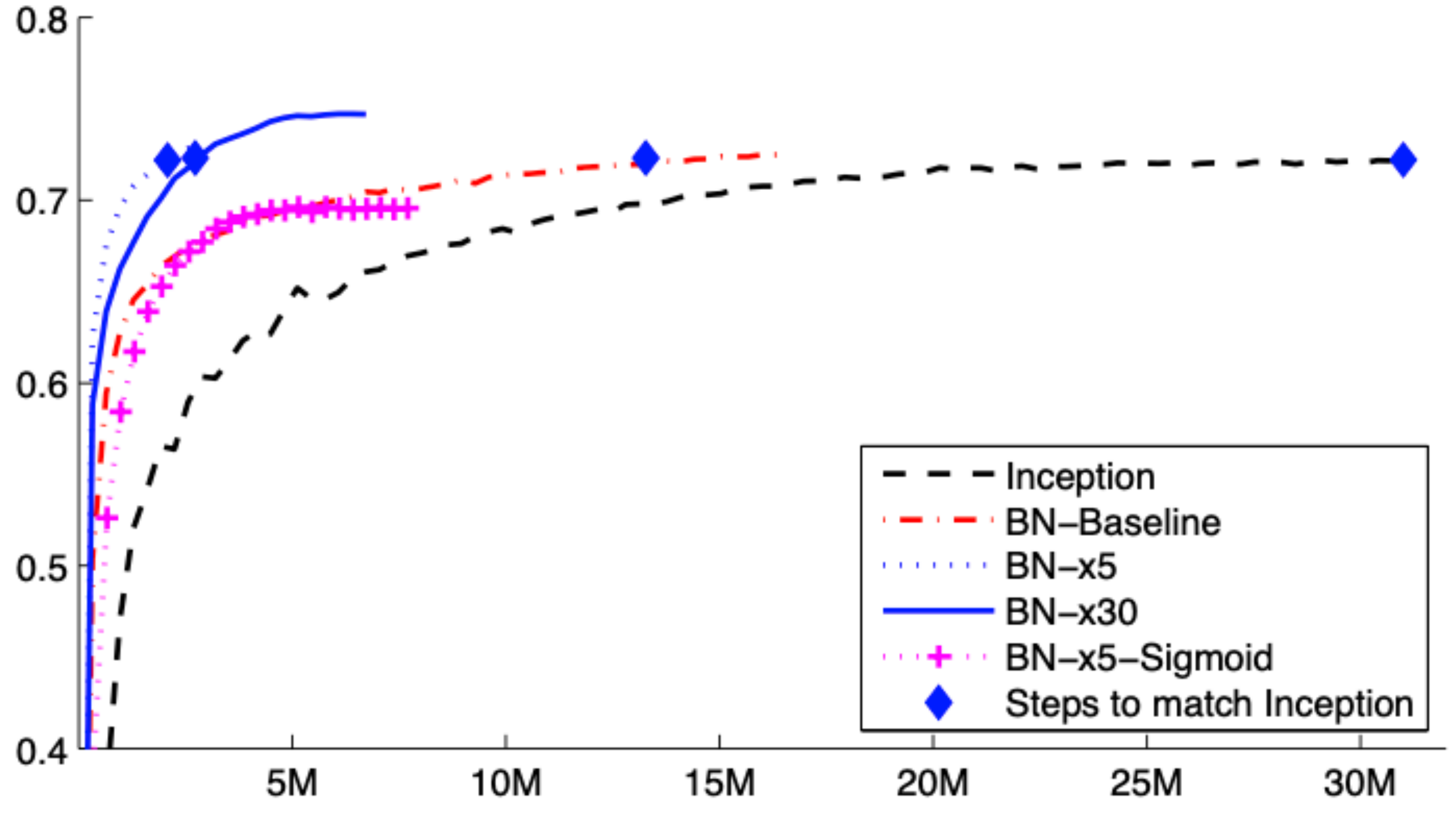
Benefits of batch normalization:

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



(a) (b) Without BN (c) With BN

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%***

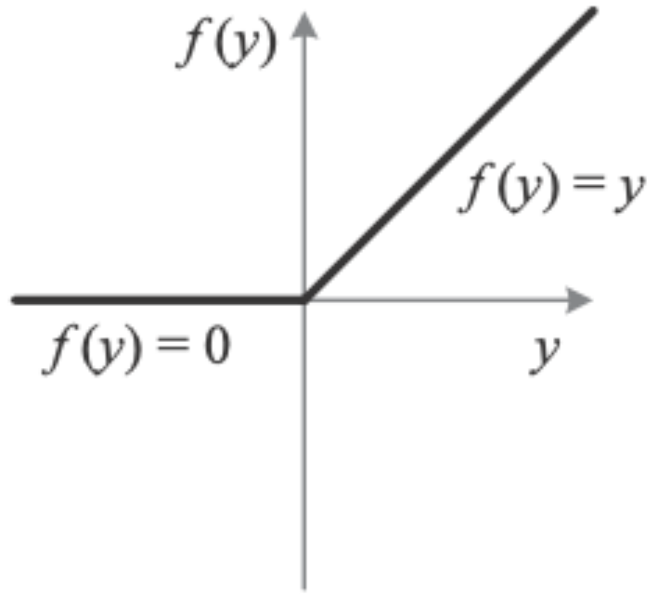
Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *arXiv preprint arXiv:1502.03167* (2015).

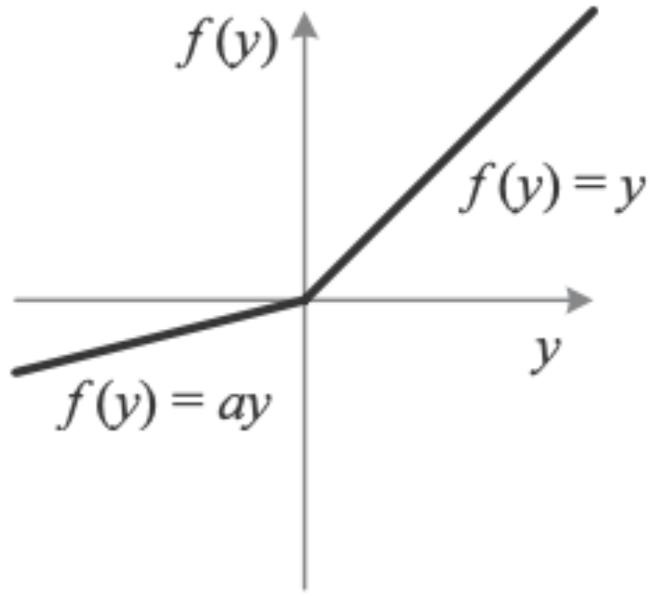
Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification

Boulder

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

$\Delta a_i \leftarrow \mu \Delta a_i + \epsilon \frac{\partial \mathcal{E}}{\partial a_i}$ 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	**32.64**	**12.75**

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

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

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

$$\mathrm{Var}[y_\ell] = n_\ell \mathrm{Var}[w_\ell x_\ell] \underbrace{=}_{\mathbb{E}[w_\ell]=0} n_\ell \underbrace{\mathrm{Var}[w_\ell]}_{=\mathbb{E}[w_\ell^2]} \underbrace{\mathbb{E}[x_\ell^2]}_{\neq \mathrm{Var}[x_\ell]}$$

$y_\ell \rightarrow$ random variable of each element in y_ℓ

$w_\ell \rightarrow$ random variable of each element in W_ℓ

$x_\ell \rightarrow$ random variable of each element in x_ℓ

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

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

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

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

Backward Propagation

$$\boxed{\begin{array}{l} \frac{1}{2}(1+a^2) n_\ell \mathrm{Var}[w_\ell] = 1 \\ \frac{1}{2}(1+a^2) \hat{n}_\ell \mathrm{Var}[w_\ell] = 1 \end{array}}$$

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

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

gradient at a pixel of this layer

$\Delta y_\ell = \frac{\partial \mathcal{E}}{\partial y_\ell} \in \mathbb{R}^{\hat{n}}$ $\quad \hat{n} = k^2 d$

$k \times k$ pixels in d channels

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

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

$\mathbb{E}[\Delta y_\ell] = \frac{1}{2}\mathbb{E}[\Delta x_{\ell+1}] = 0$

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

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

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

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

He, Kaiming, et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." *Proceedings of the IEEE international conference on computer vision*. 2015.

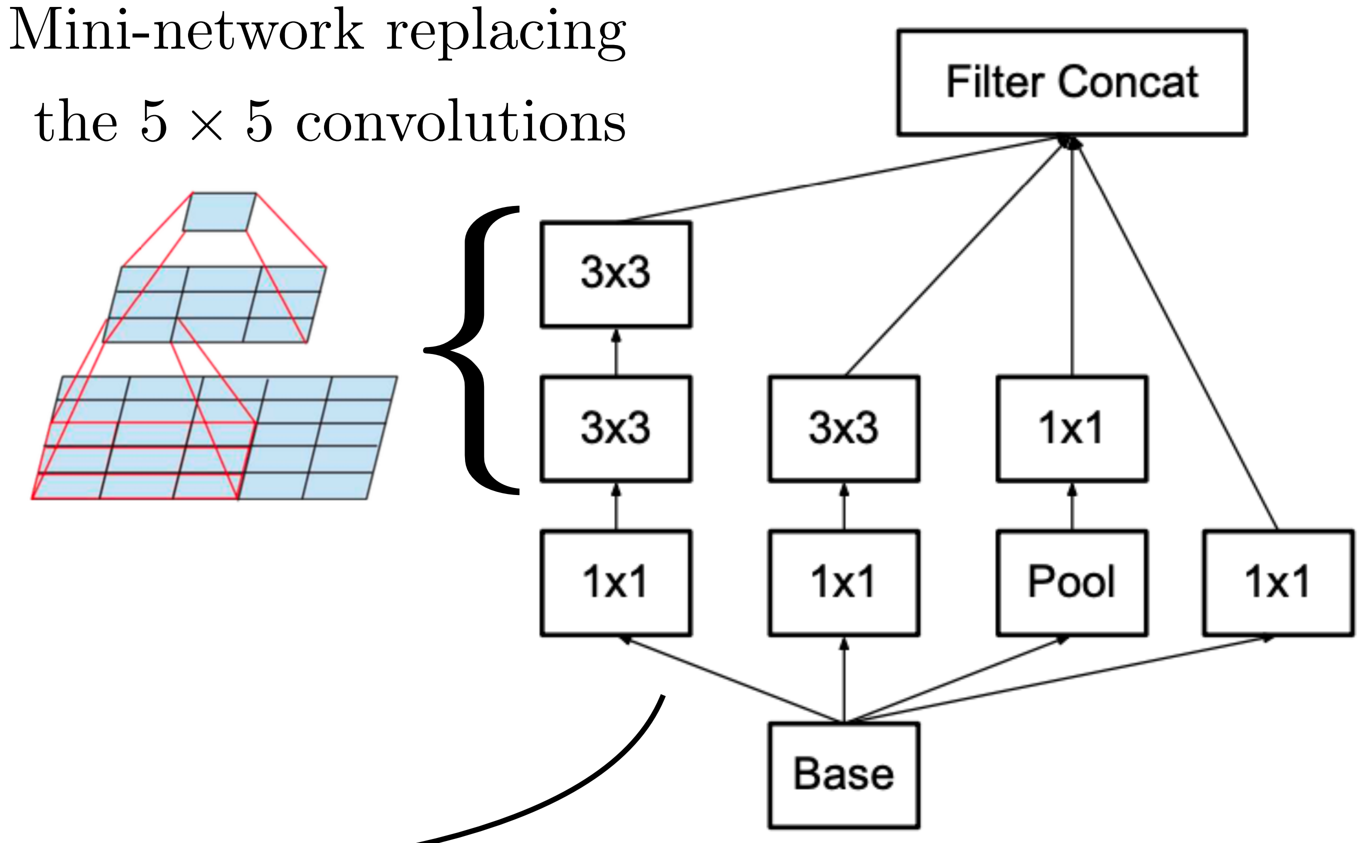
Boulder

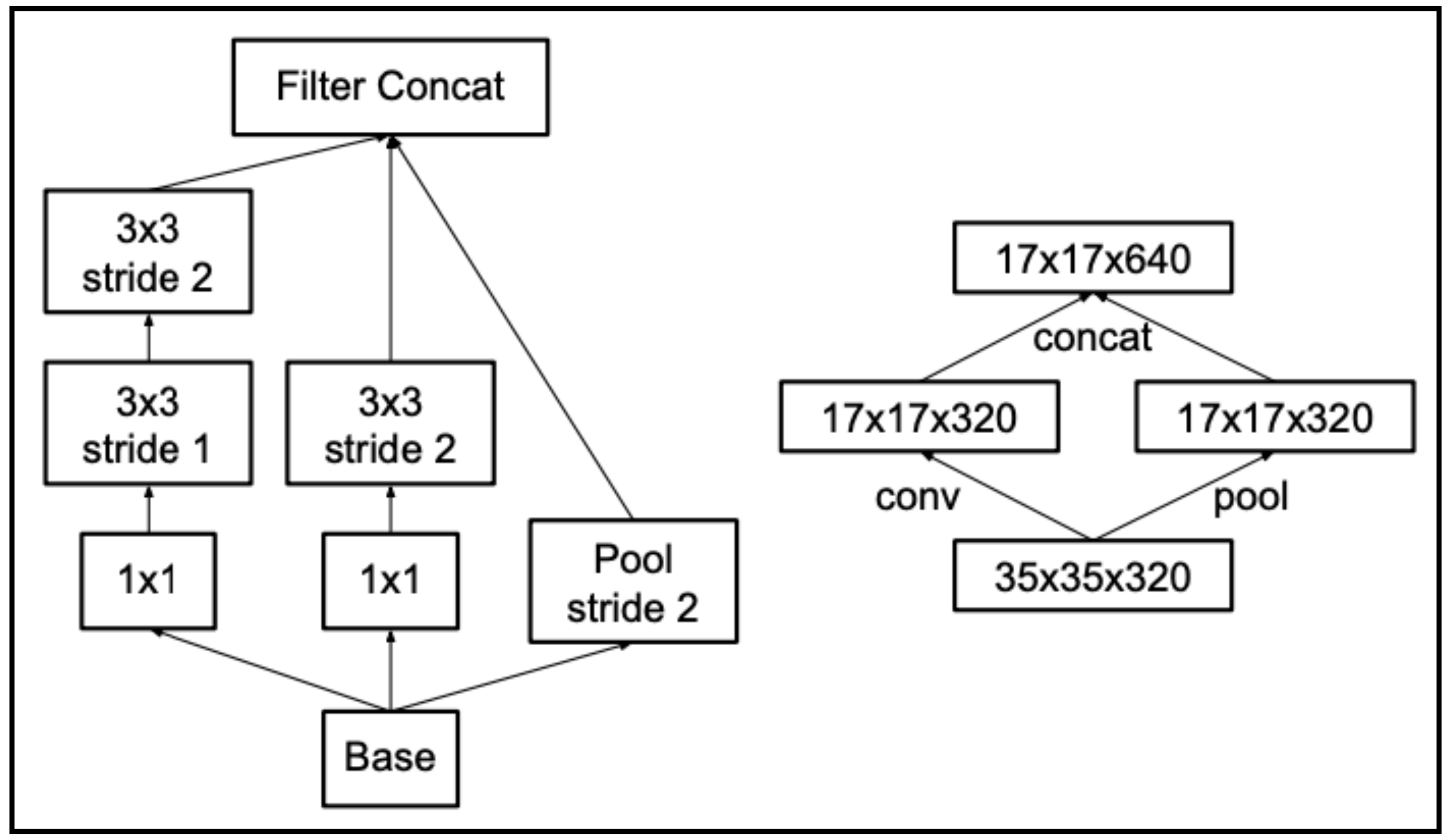
Rethinking the Inception Architecture for Computer Vision

YouTube Playlist

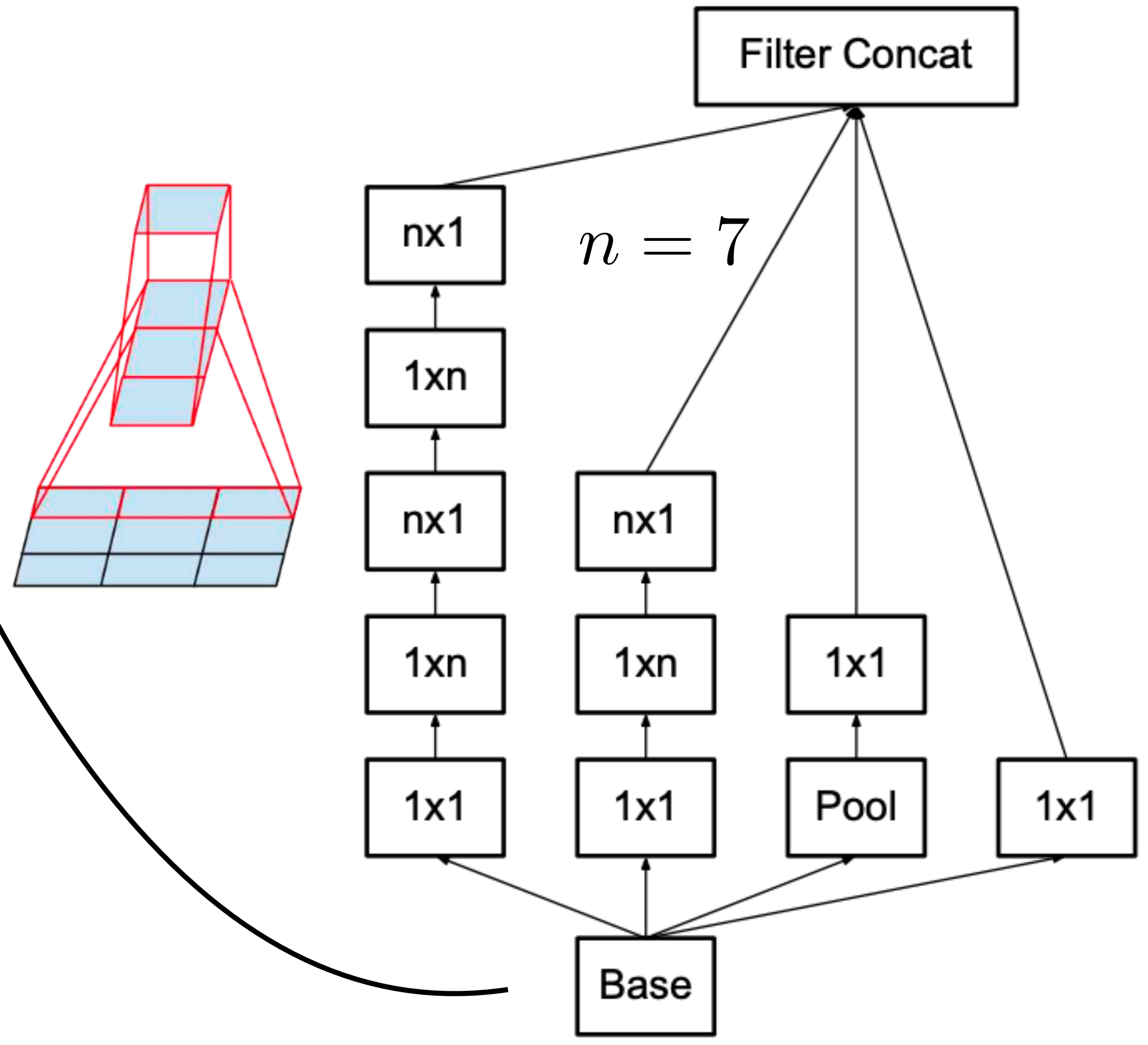
type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7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

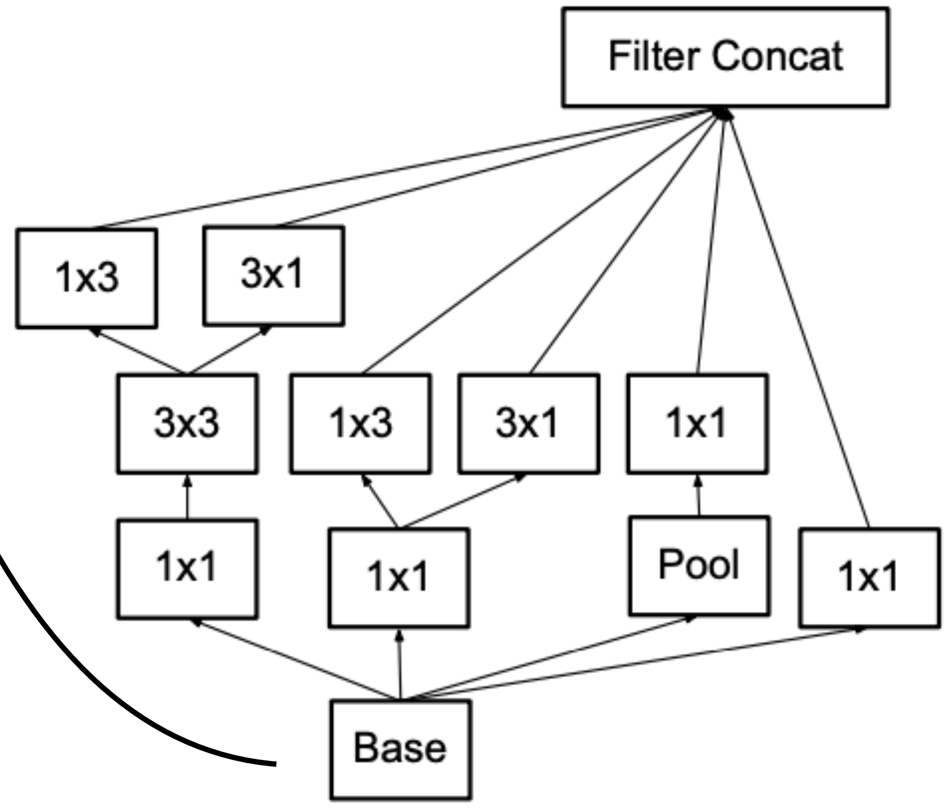
Mini-network replacing the 5×5 convolutions





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

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

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

$u = 1/K$

$x \to$ training example

$k \in \{1, \dots, K\} \to$ label

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

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

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

$k^* \to$ ground truth label

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

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

Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

Boulder

Training Very Deep Networks

Highway Networks

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

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

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

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

element-wise multiplication

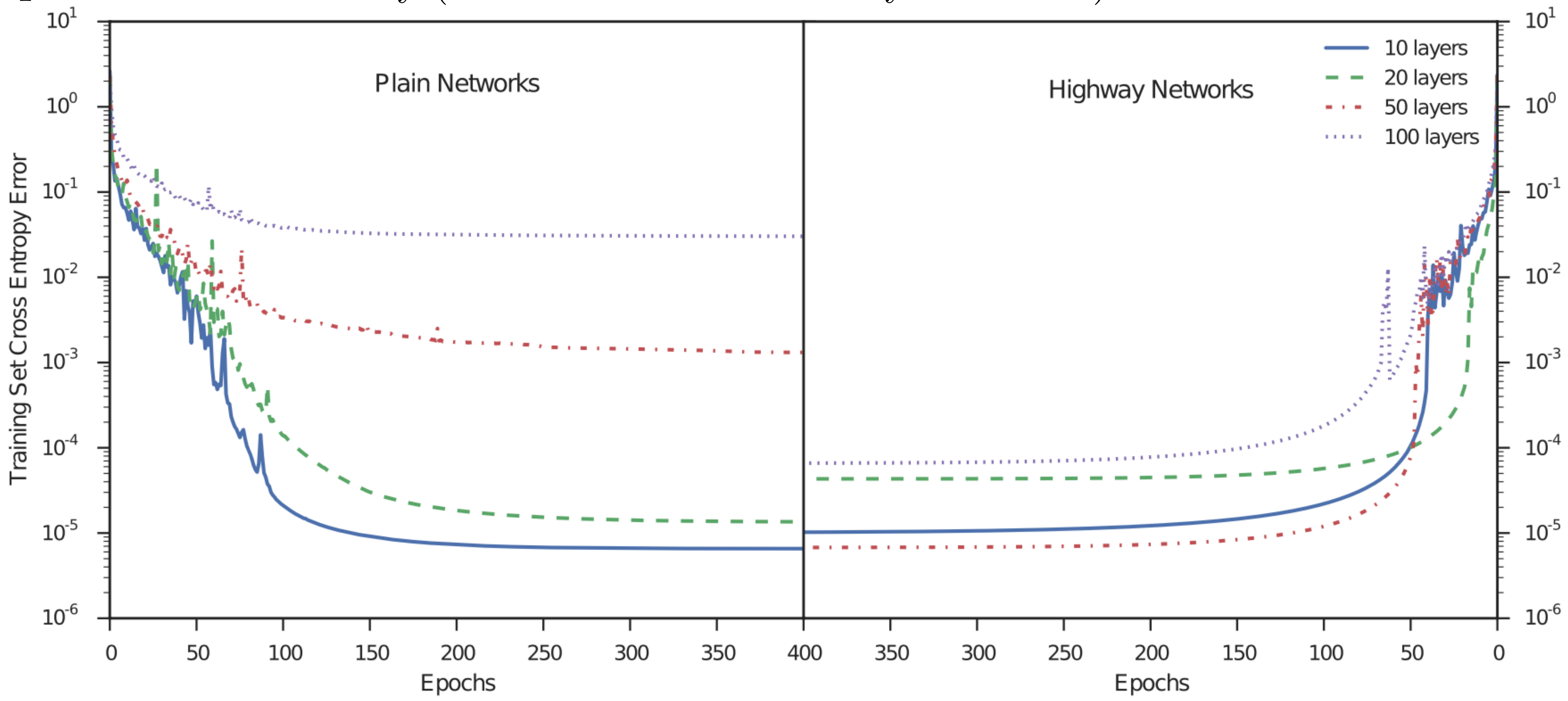
$T \rightarrow$ transform gate

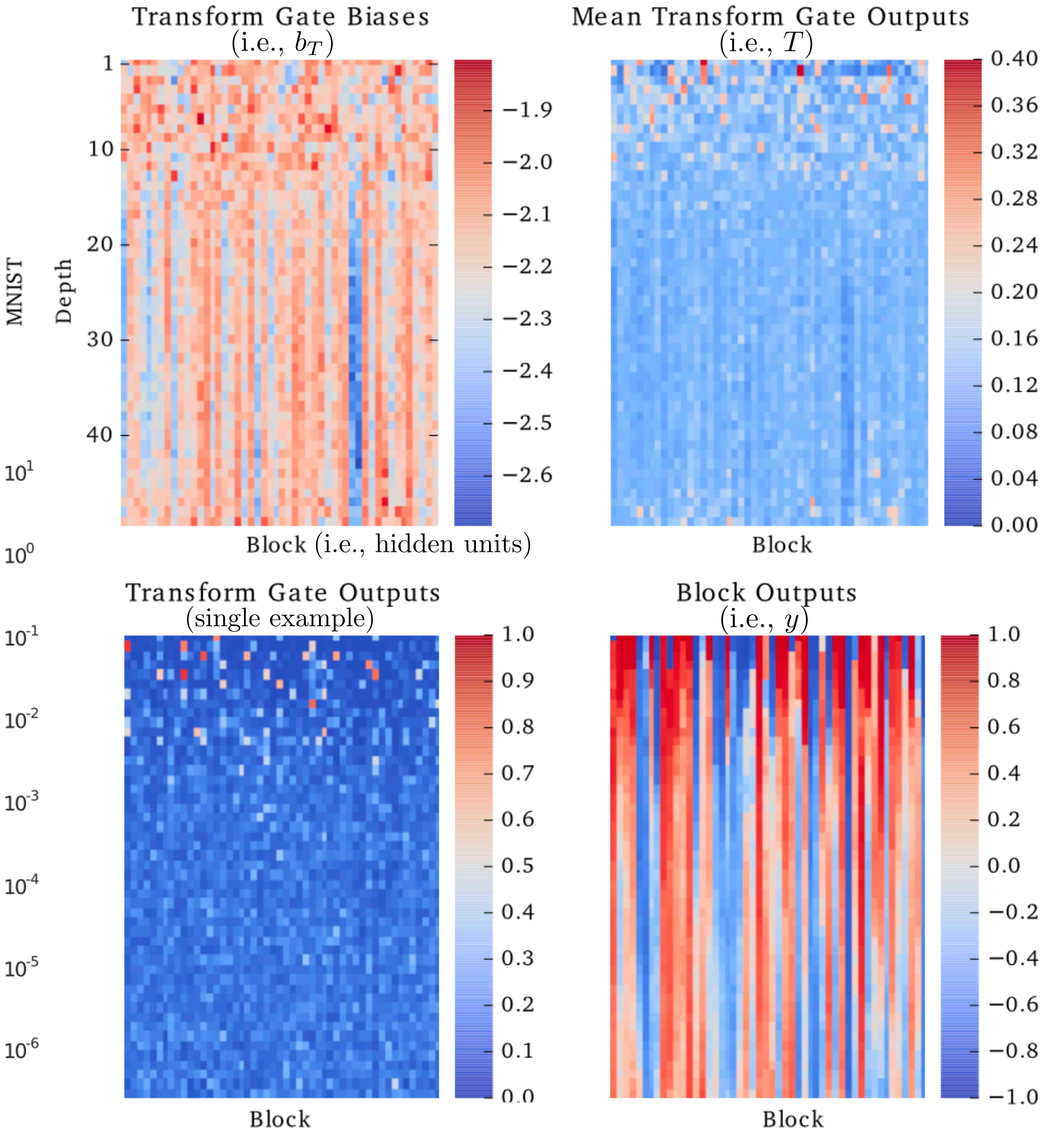
$C \rightarrow$ carry gate

$C = 1 - T$

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

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





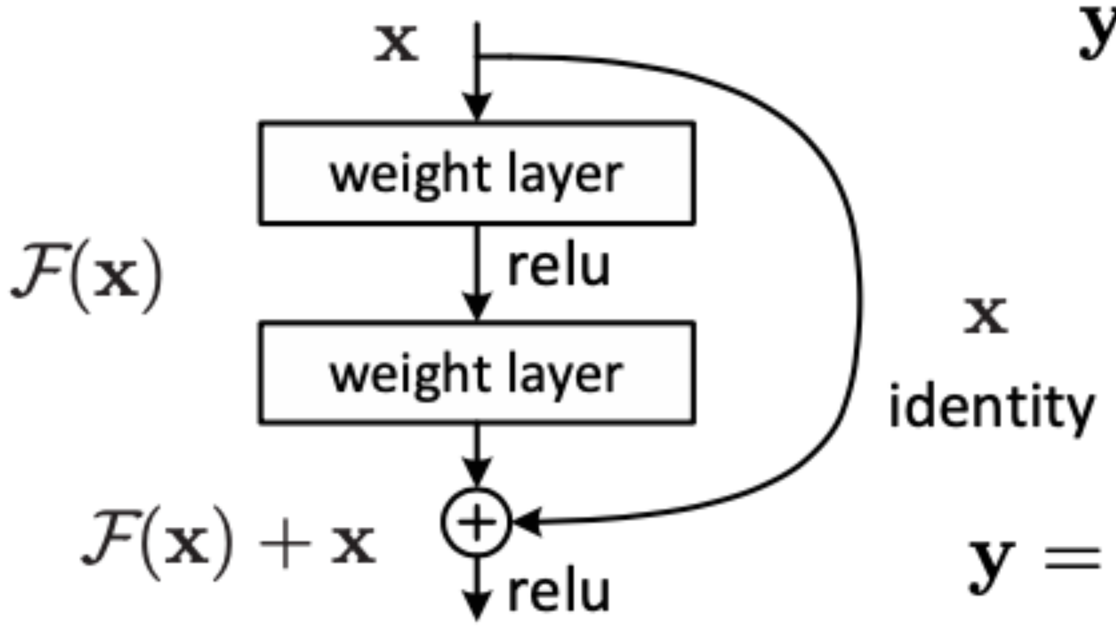
Srivastava, Rupesh Kumar, Klaus Greff, and Jürgen Schmidhuber. "Training very deep networks." *arXiv preprint arXiv:1507.06228* (2015).

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

Deep Residual Learning for Image Recognition

Boulder

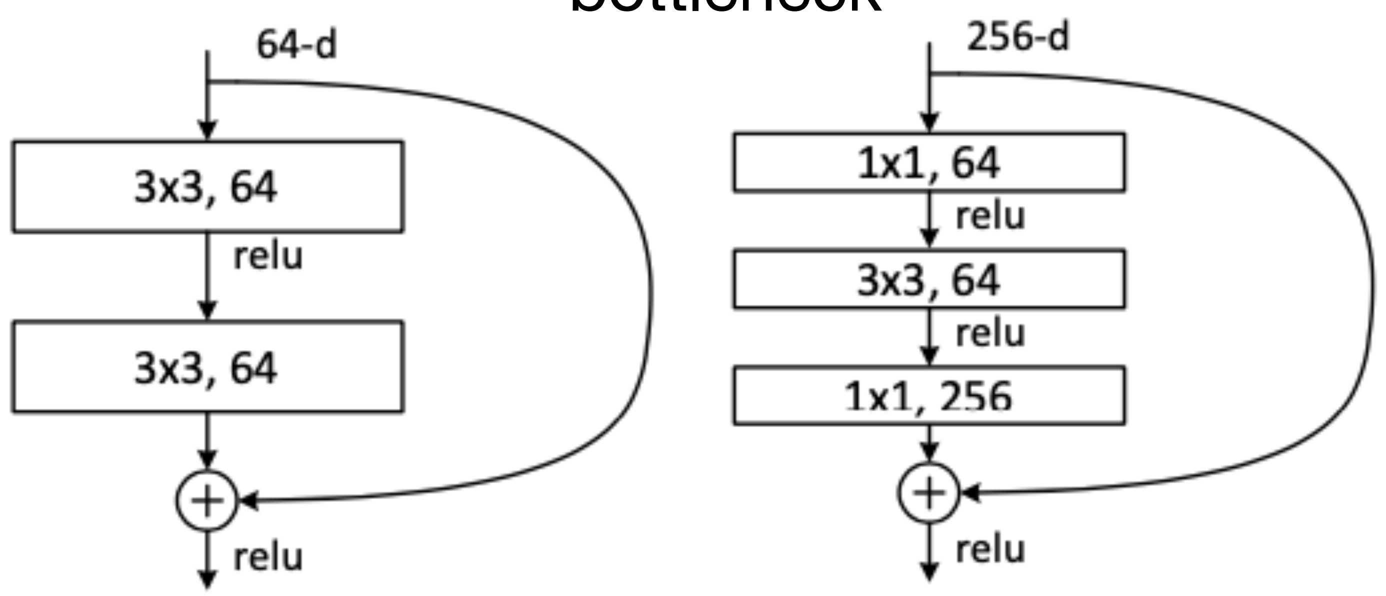
[YouTube Video]



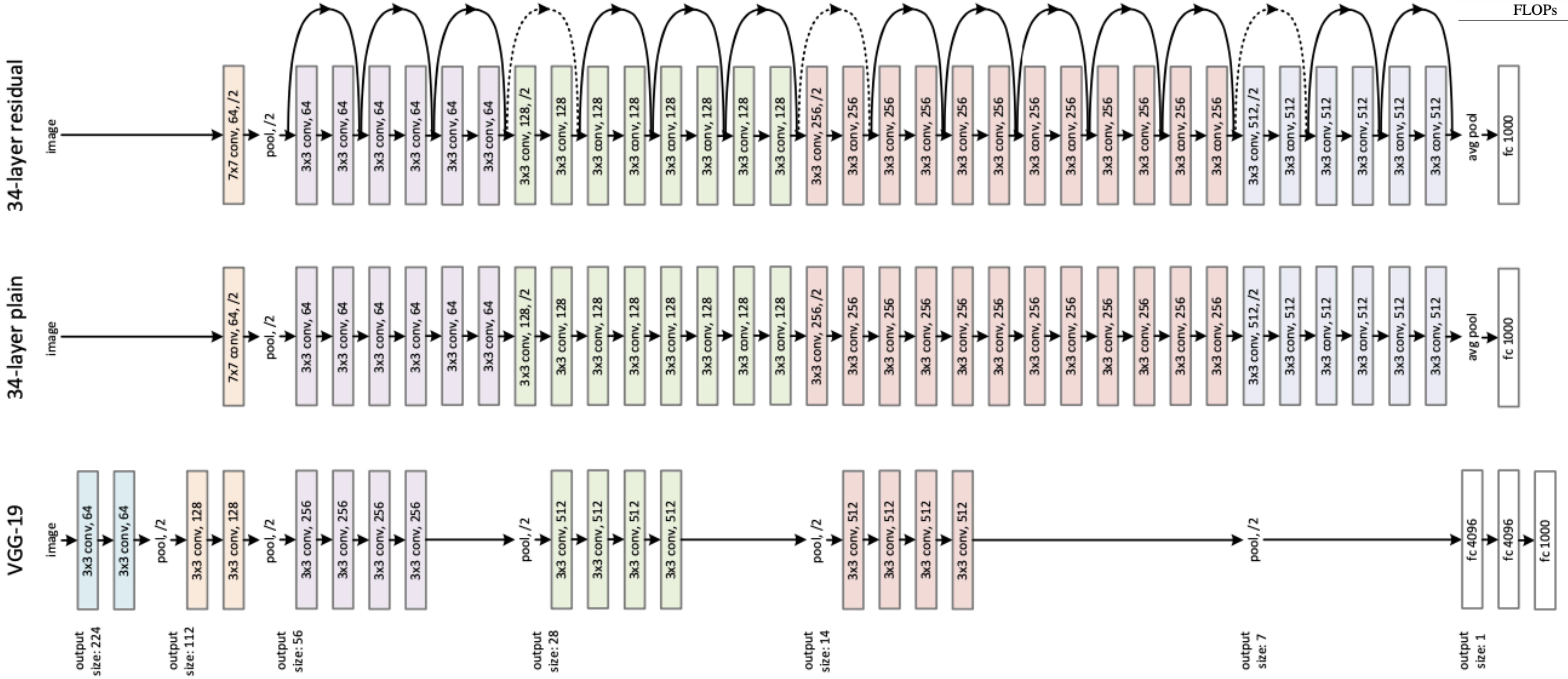
$$\mathbf{y} = \underbrace{\mathcal{F}(\mathbf{x}, \{W_i\})}_{W_2\sigma(W_1\mathbf{x})} + \mathbf{x}.$$

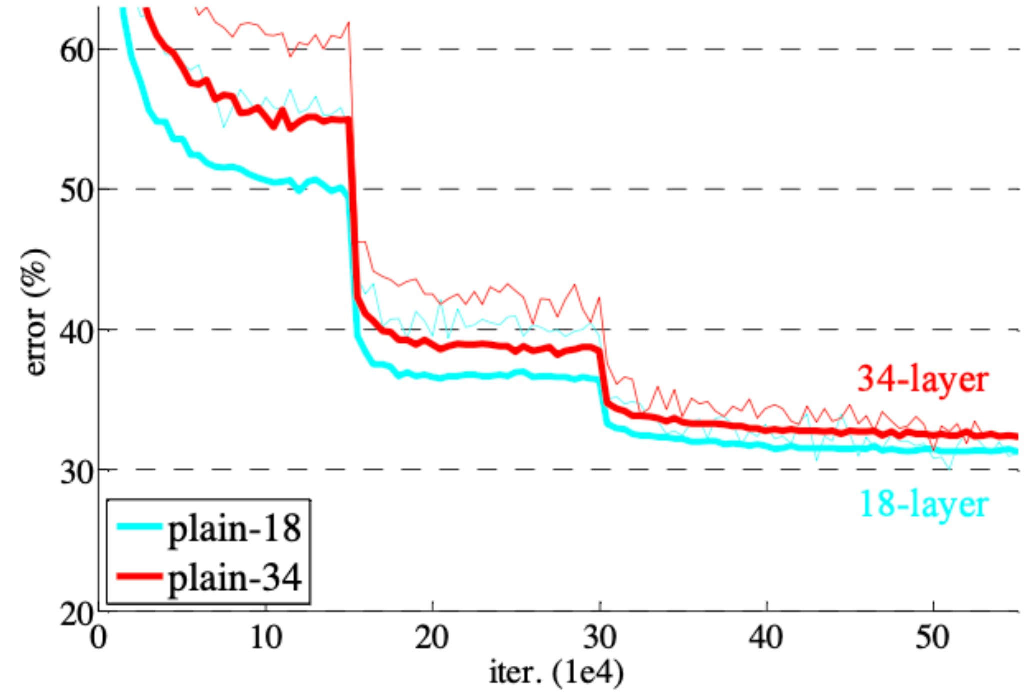
$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}.$$

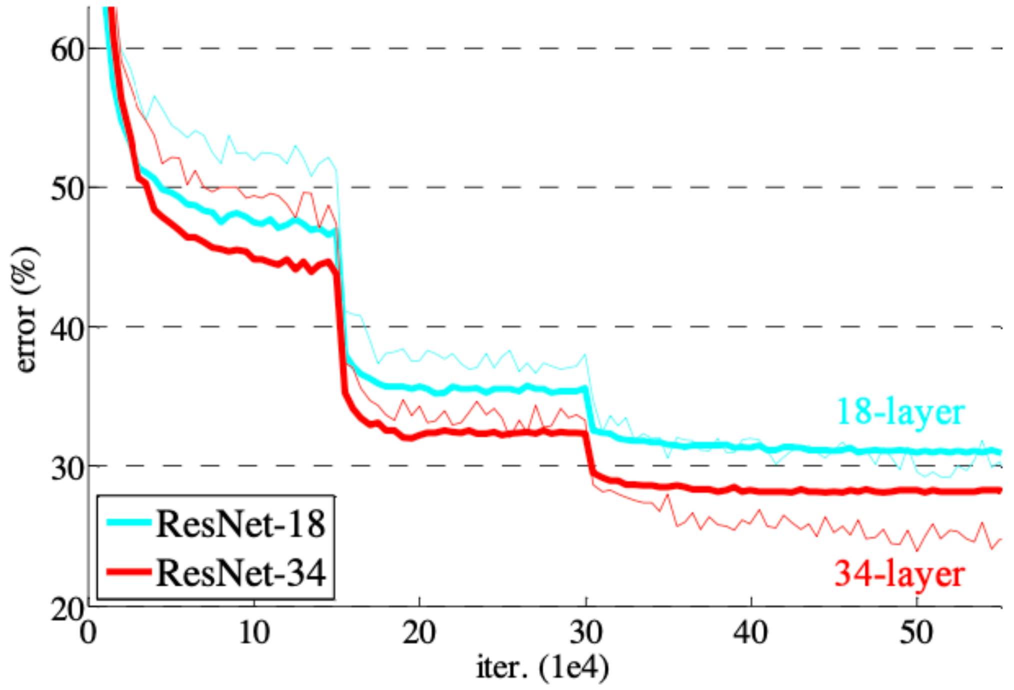
bottleneck



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9



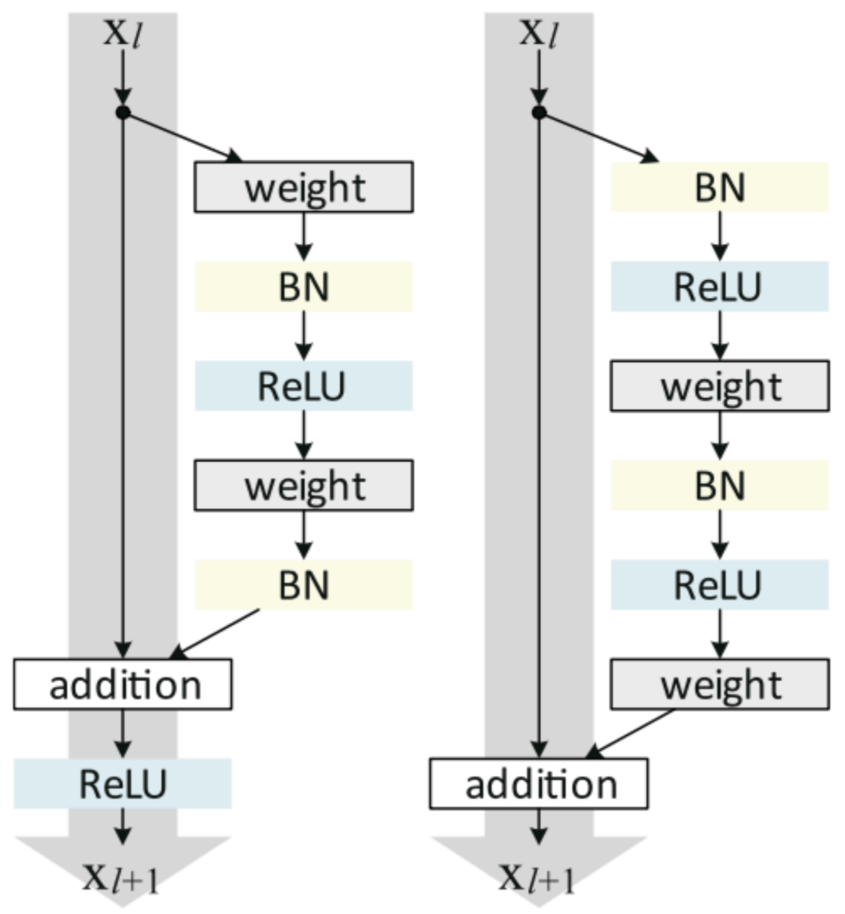




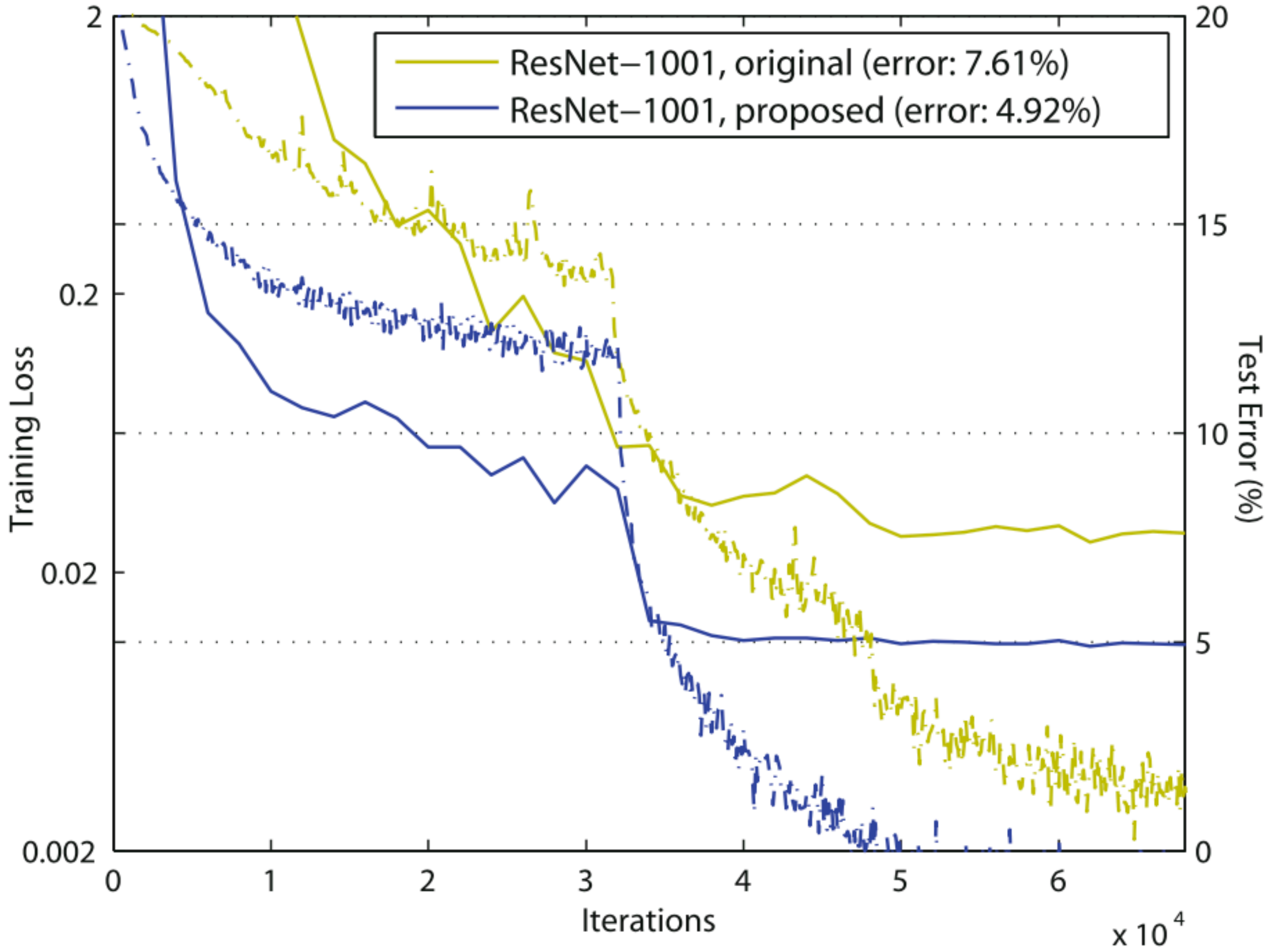
He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

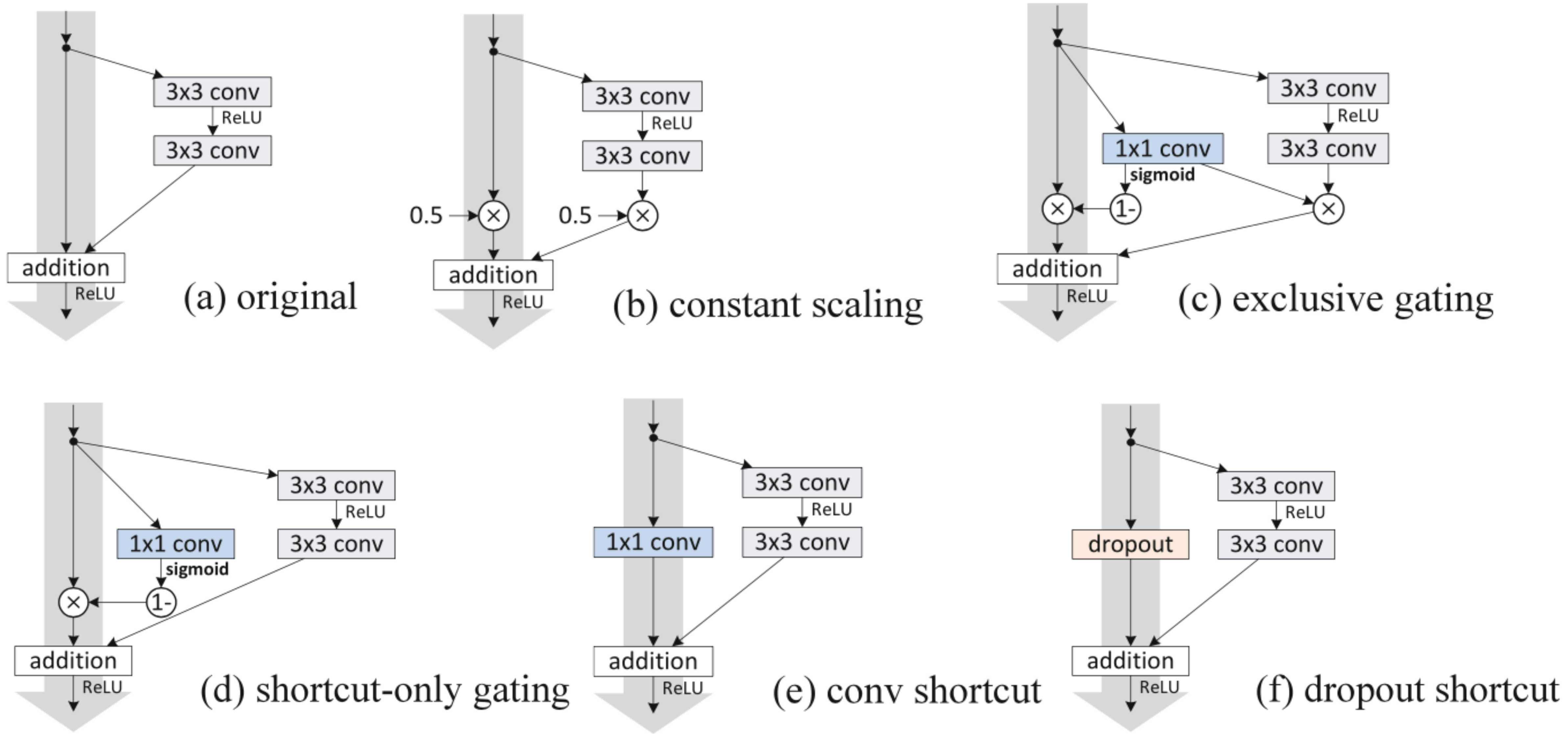
Identity Mappings in Deep Residual Networks

[YouTube Video](#)



(a) original (b) proposed





(a) original (b) constant scaling (c) exclusive gating

(d) shortcut-only gating (e) conv shortcut (f) dropout shortcut

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

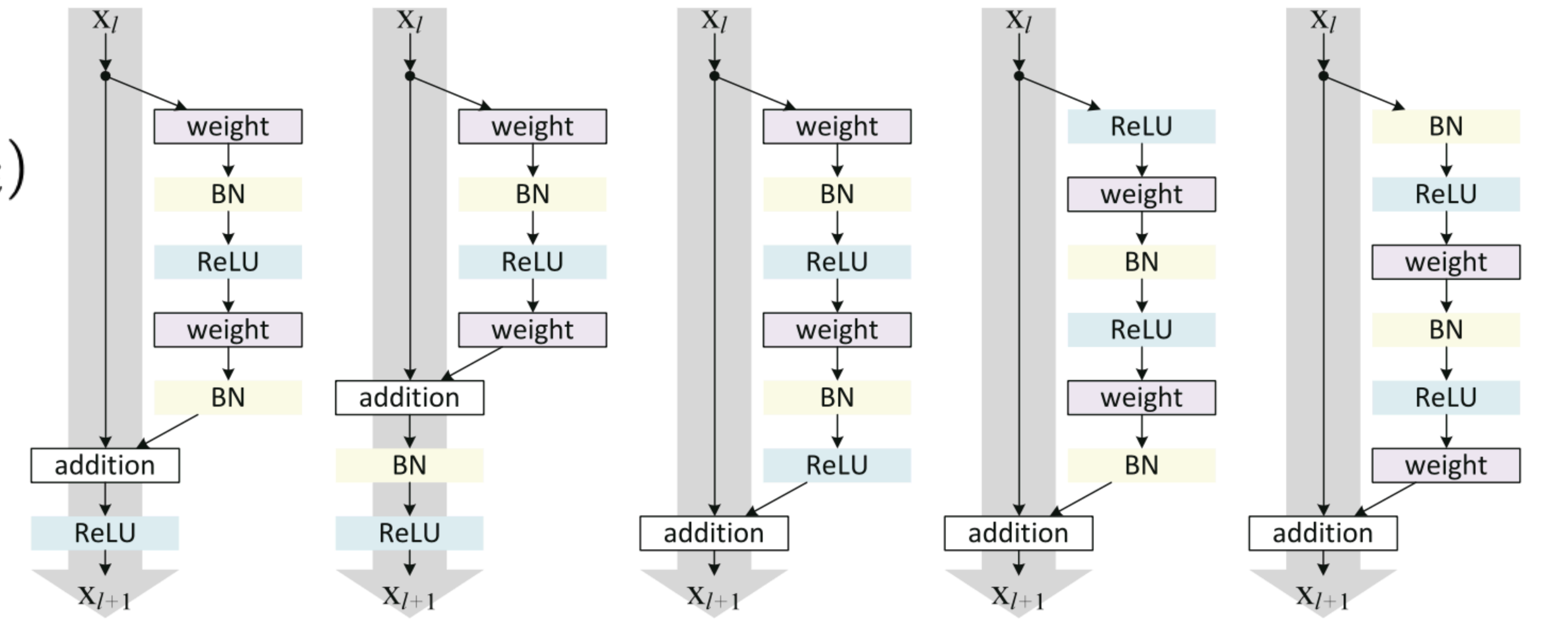
$$\left.\begin{array}{l} h(\mathbf{x}_l) = \mathbf{x}_l \\ f \text{ is ReLU} \end{array}\right\} \text{Original}$$

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

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

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

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



(a) original (b) BN after addition (c) ReLU before addition (d) ReLU-only pre-activation (e) **full pre-activation**

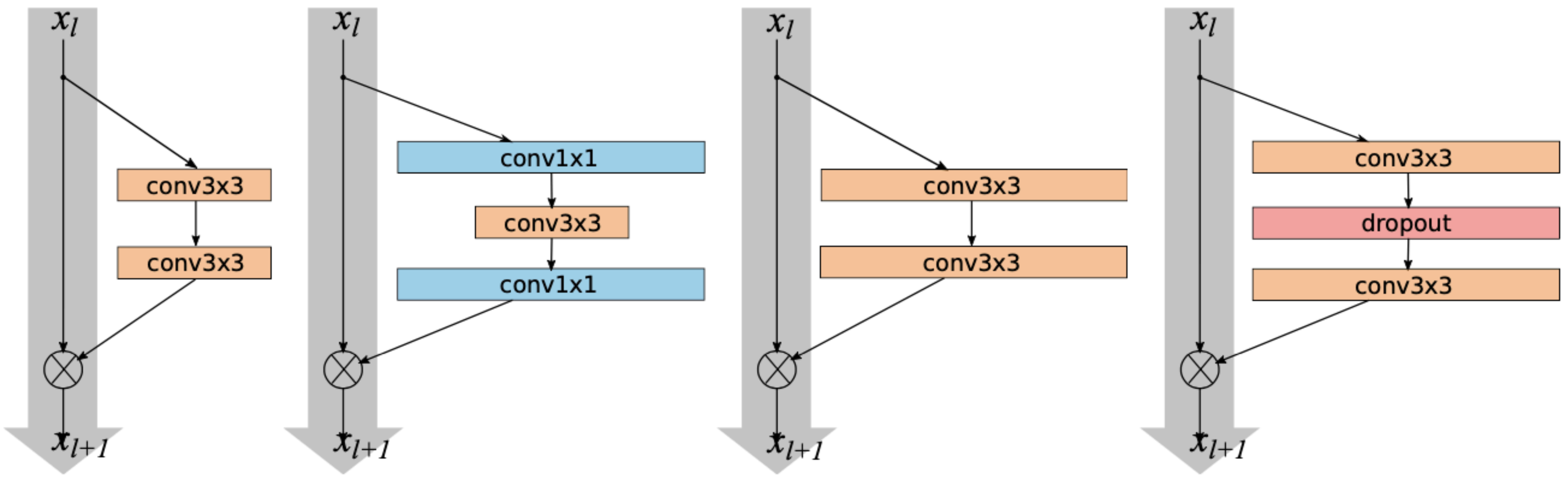
He, Kaiming, et al. "Identity mappings in deep residual networks." *European conference on computer vision*. Springer, Cham, 2016.

Wide Residual Networks

[YouTube Video]

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

$$\underbrace{\text{Conv-BN-ReLU}}_{\text{ResNet}} \rightarrow \underbrace{\text{BN-ReLU-Conv}}_{\text{Identity Mapping}}$$



(a) basic (b) bottleneck (c) basic-wide (d) wide-dropout

$n \rightarrow$ 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 1×1 layer
3. $B(1,3,1)$ - with the same dimensionality of all convolutions, «straightened» bottleneck
4. $B(1,3)$ - the network has alternating 1×1 - 3×3 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

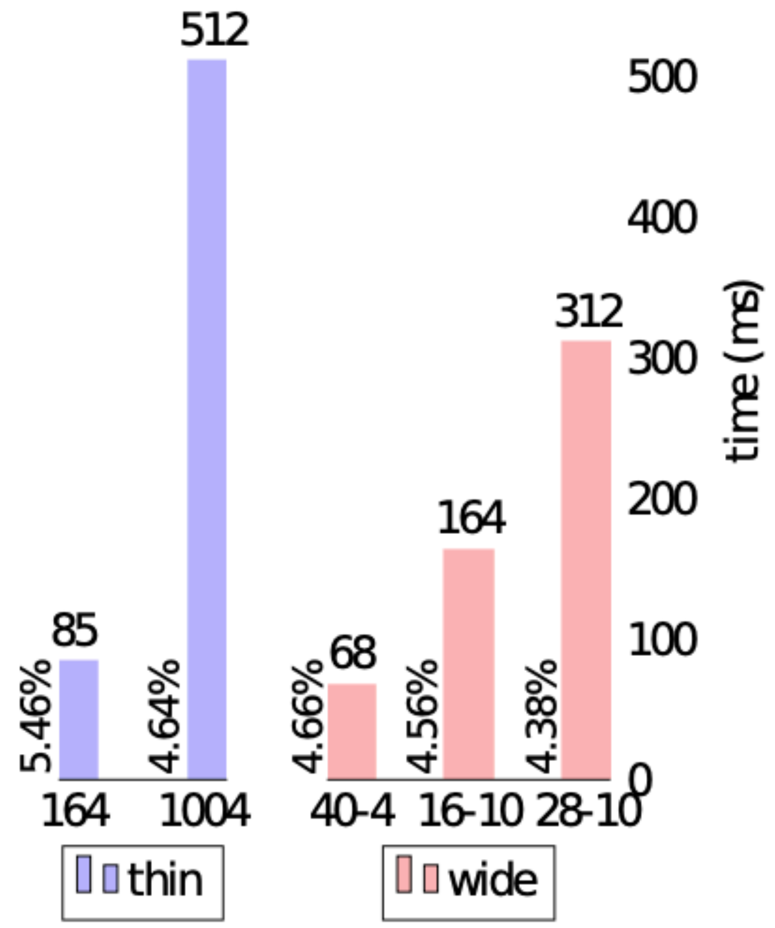
group name	output size	block type = $B(3,3)$
conv1	32×32	[3×3, 16]
conv2	32×32	$\begin{bmatrix} 3\times3, 16\times k \\ 3\times3, 16\times k \end{bmatrix} \times N$
conv3	16×16	$\begin{bmatrix} 3\times3, 32\times k \\ 3\times3, 32\times k \end{bmatrix} \times N$
conv4	8×8	$\begin{bmatrix} 3\times3, 64\times k \\ 3\times3, 64\times k \end{bmatrix} \times N$
avg-pool	1×1	[8 × 8]

depth	k	# params	CIFAR-10	CIFAR-100
40	1	0.6M	6.85	30.89
40	2	2.2M	5.33	26.04
40	4	8.9M	4.97	22.89
40	8	35.7M	4.66	-
28	10	36.5M	**4.17**	20.50
28	12	52.5M	4.33	**20.43**
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
$B(1,3,1)$	40	1.4M	85.8	6.06
$B(3,1)$	40	1.2M	67.5	5.78
$B(1,3)$	40	1.3M	72.2	6.42
$B(3,1,1)$	40	1.3M	82.2	5.86
$B(3,3)$	28	1.5M	67.5	5.73 ⟵
$B(3,1,3)$	22	1.1M	59.9	5.78

depth	k	dropout	CIFAR-10	CIFAR-100	SVHN
16	4		5.02	24.03	1.85
16	4	✓	5.24	23.91	1.64
28	10		4.00	19.25	-
28	10	✓	**3.89**	**18.85**	-
52	1		6.43	29.89	2.08
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	-
pre-act-ResNet[13]	110	1.7M	6.37	-
	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	**4.00**	**19.25**

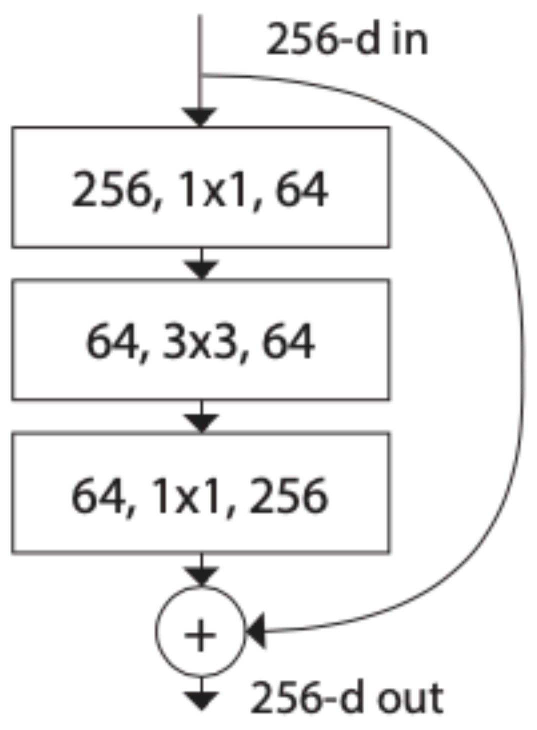


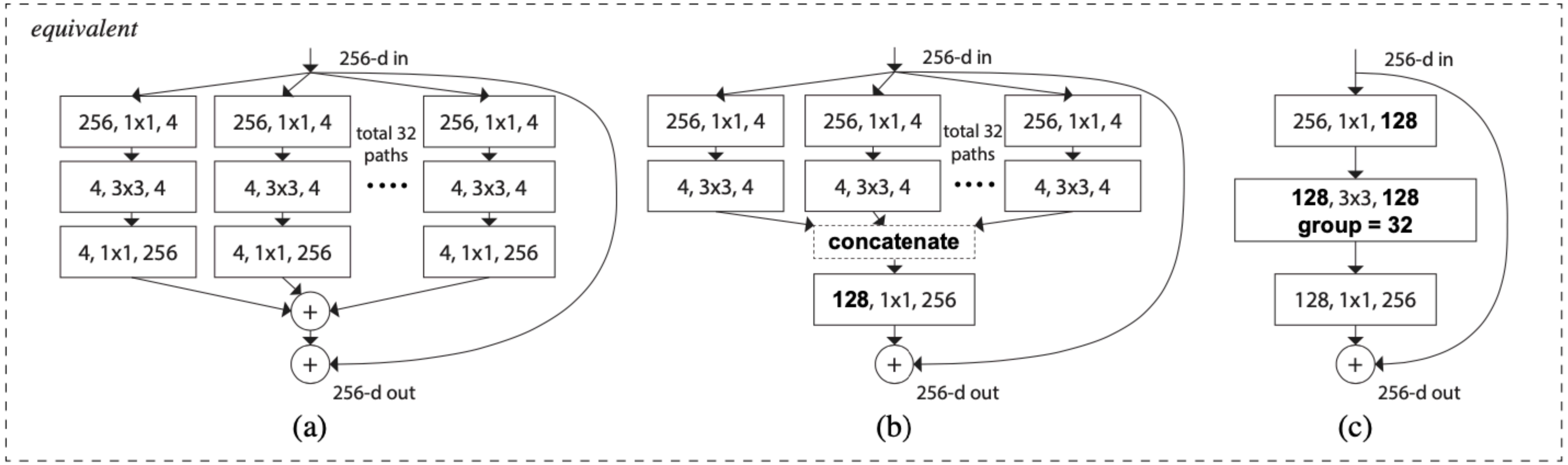
Zagoruyko, Sergey, and Nikos Komodakis. "Wide residual networks." *arXiv preprint arXiv:1605.07146* (2016).

Aggregated Residual Transformations for Deep Neural Networks

Boulder

YouTube Video





stage	output	ResNet-50		ResNeXt-50 (32×4d)	
conv1	112×112	7×7, 64, stride 2		7×7, 64, stride 2	
conv2	56×56	3×3 max pool, stride 2		3×3 max pool, stride 2	
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}$	×3	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix}$	×3
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}$	×4	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix}$	×4
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}$	×6	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix}$	×6
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}$	×3	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix}$	×3
	1×1	global average pool 1000-d fc, softmax		global average pool 1000-d fc, softmax	
# params.		**25.5**×10^6		**25.0**×10^6	
FLOPs		**4.1**×10^9		**4.2**×10^9	

$$\sum_{i=1}^{32} \underbrace{W_i}_{\in\mathbb{R}^{256\times4}} \underbrace{x_i}_{\in\mathbb{R}^4} = \underbrace{[W_1,\ldots,W_{32}]}_{\in\mathbb{R}^{256\times128}} \underbrace{[x_1;\ldots;x_{32}]}_{\mathbb{R}^{128}}$$

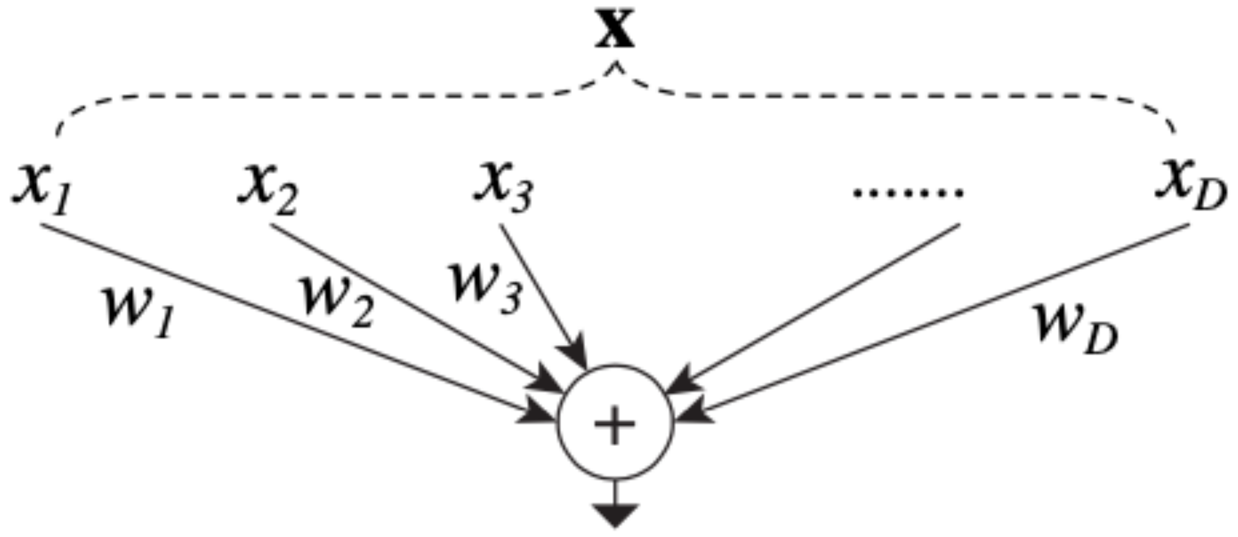
$$\underbrace{\text{split}}_{1\times1 \text{ conv}} - \underbrace{\text{transform}}_{3\times3,5\times5,\text{ etc. conv}} - \underbrace{\text{merge}}_{\text{concatenation}} \quad \text{strategy} \to \text{inception}$$

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

$C \to$ cardinality

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

cardinality: size of the set of transformations



A simple neuron that performs inner product.

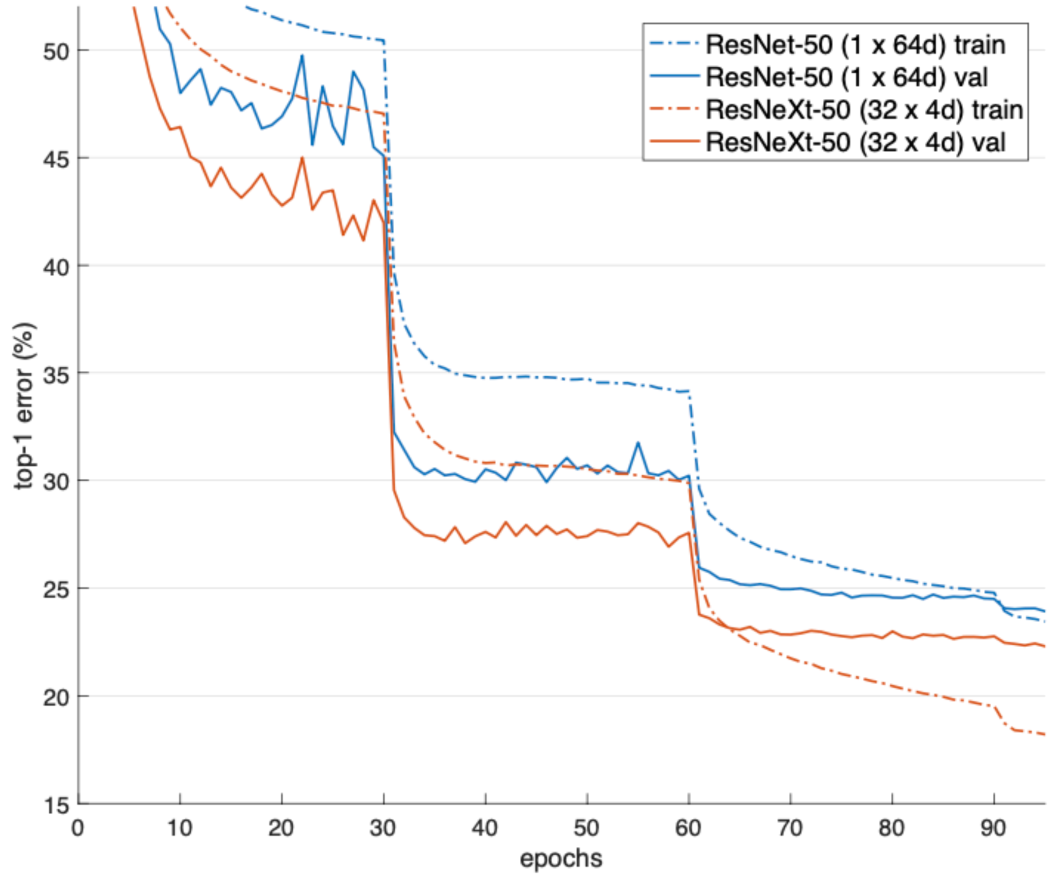
$$\sum_{i=1}^{D} w_i x_i \to \text{neuron}$$

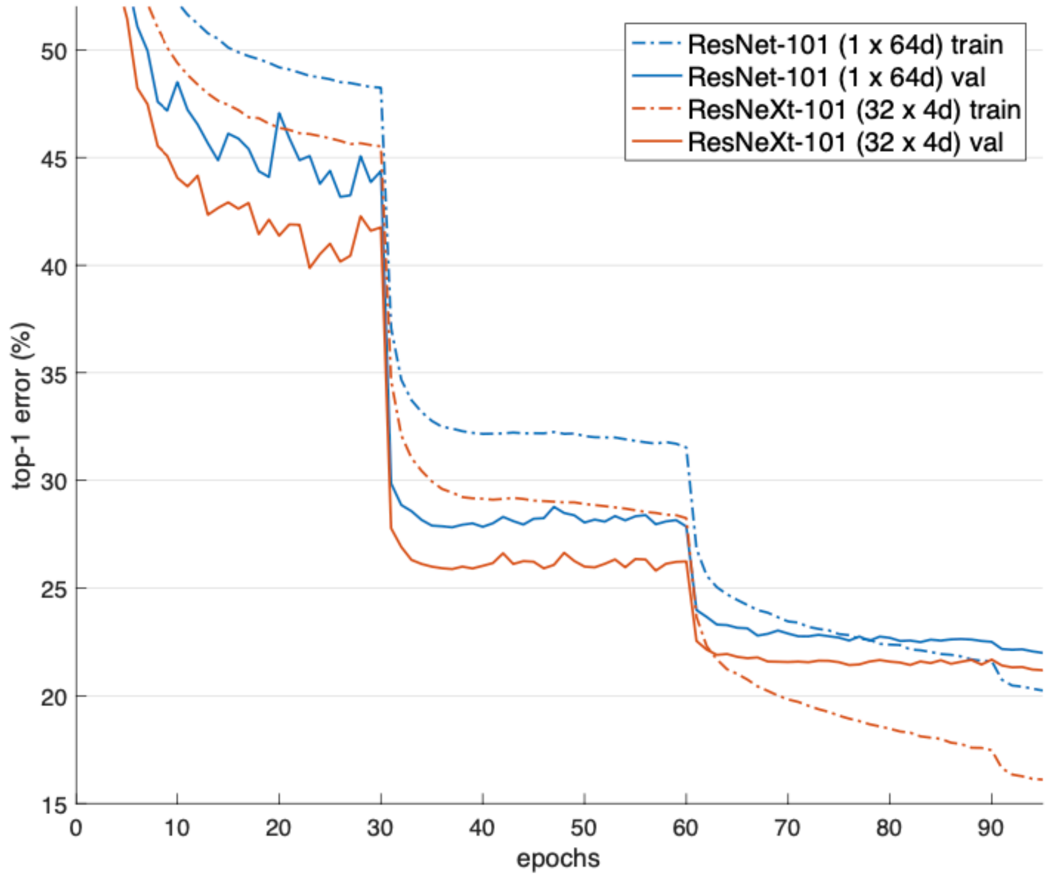
$$x = [x_1, x_2, \ldots, x_D]$$

split: $x \to x_i$

transform: $x_i w_i$

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

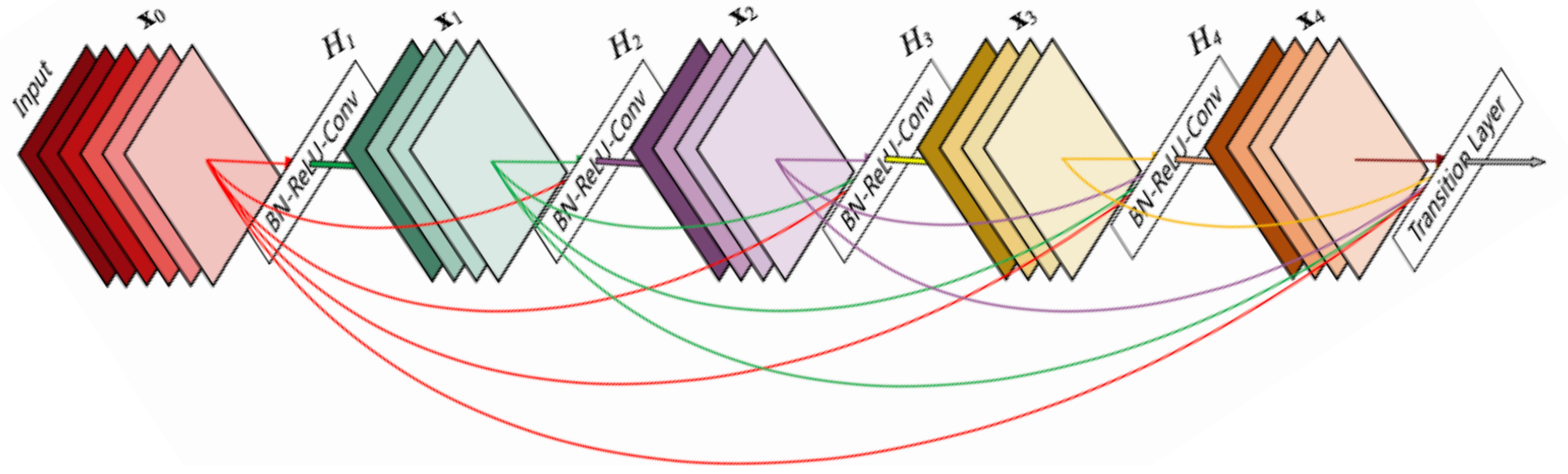


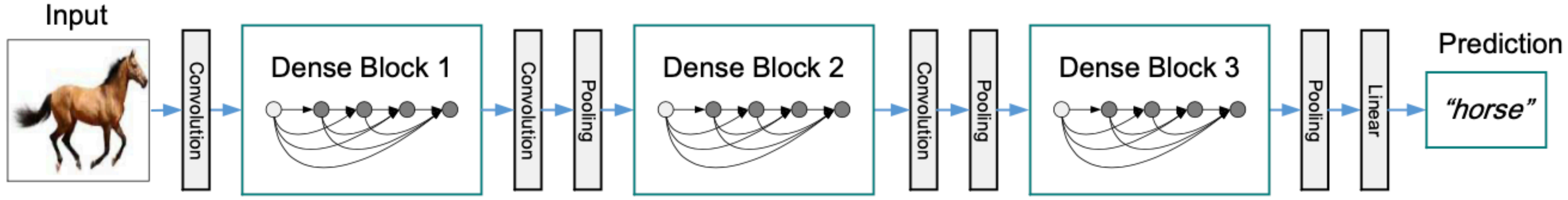


Xie, Saining, et al. "Aggregated residual transformations for deep neural networks." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

Densely Connected Convolutional Networks

YouTube Video





Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-265
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

$x_0 \rightarrow$ 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 ℓ-th layer

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

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

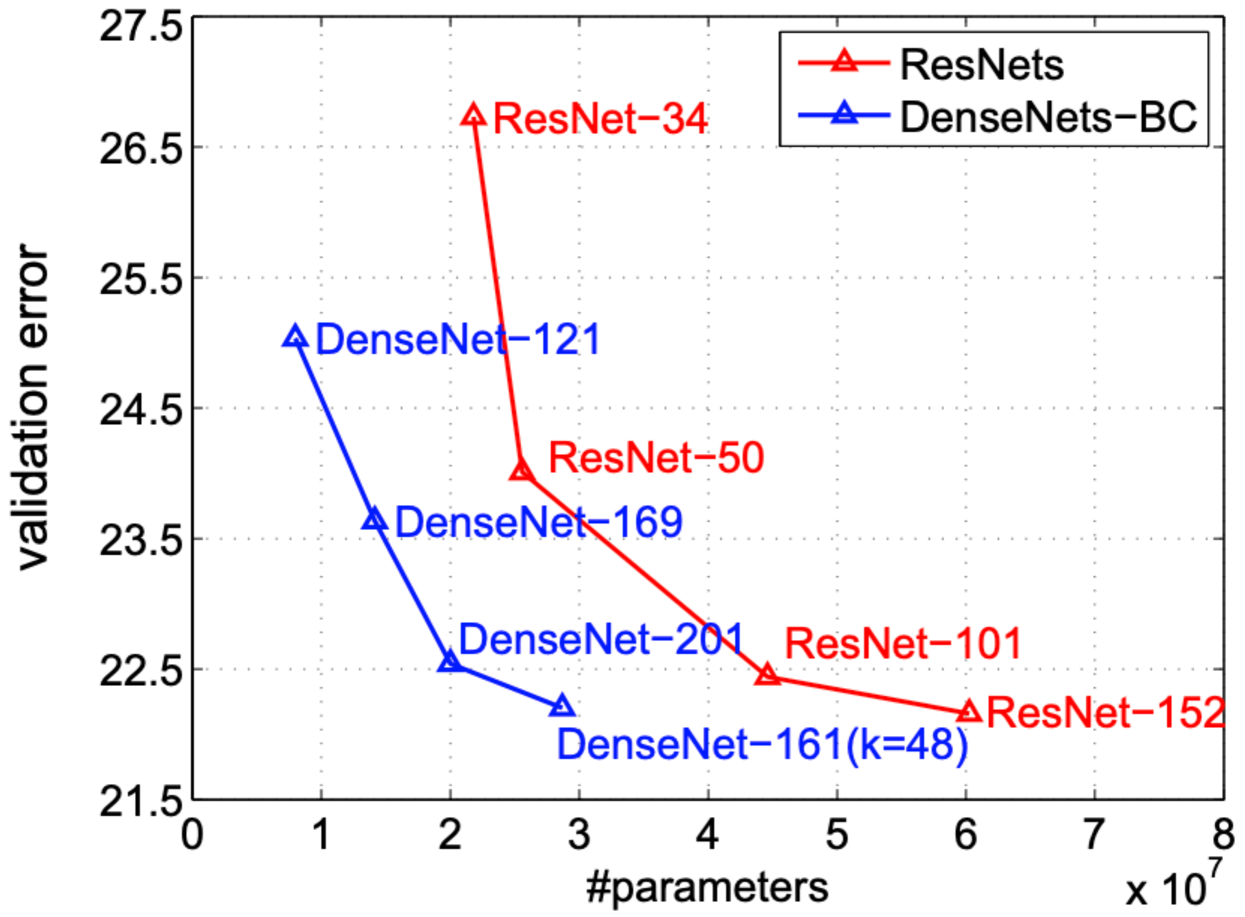
$x_\ell = H_\ell(\underbrace{[x_0, x_1, \dots, x_{\ell-1}]}_{\text{concatenate}})$

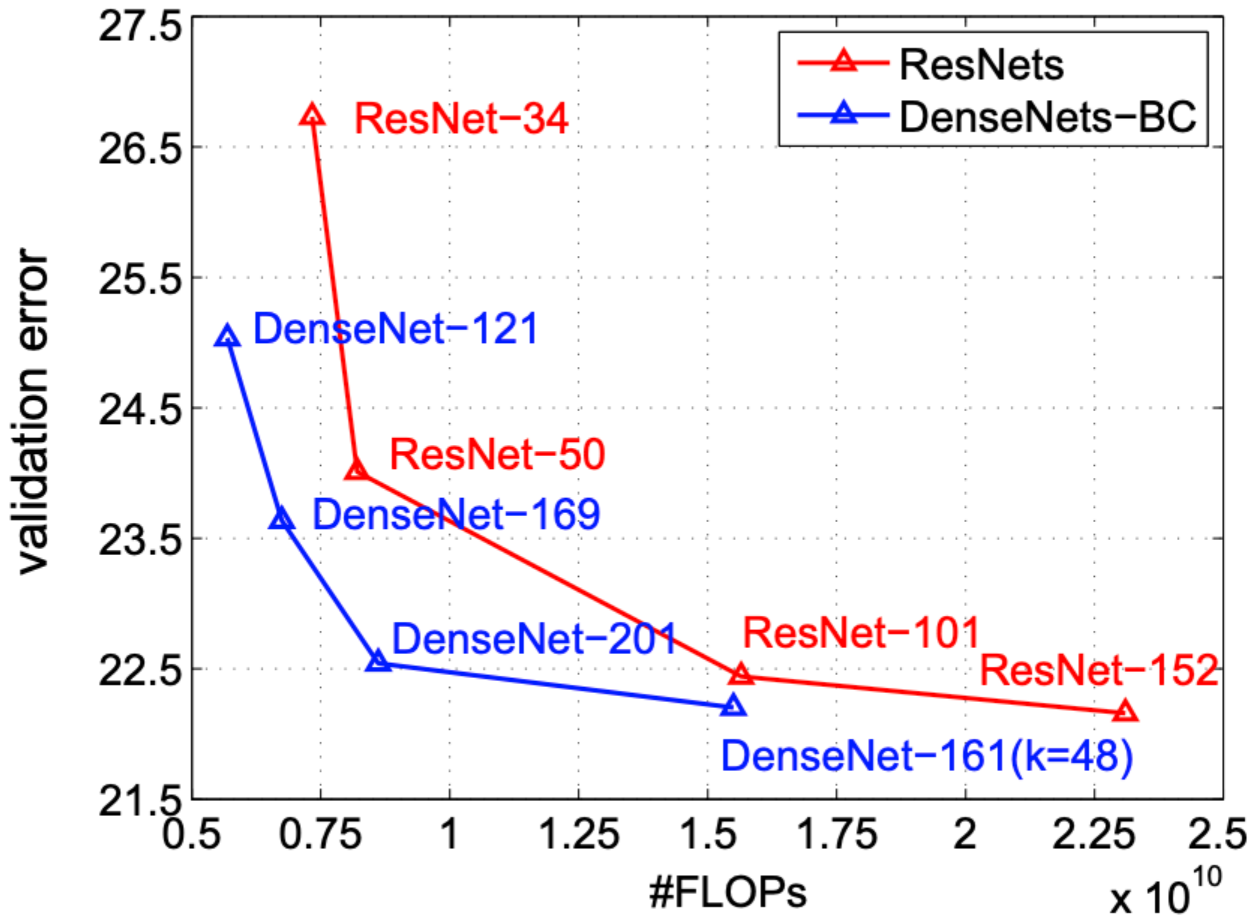
BN-ReLU-Conv(1×1)-BN-ReLU-Conv(3×3)

$4k$ feature maps

growth rate of the network
$k = 12$

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





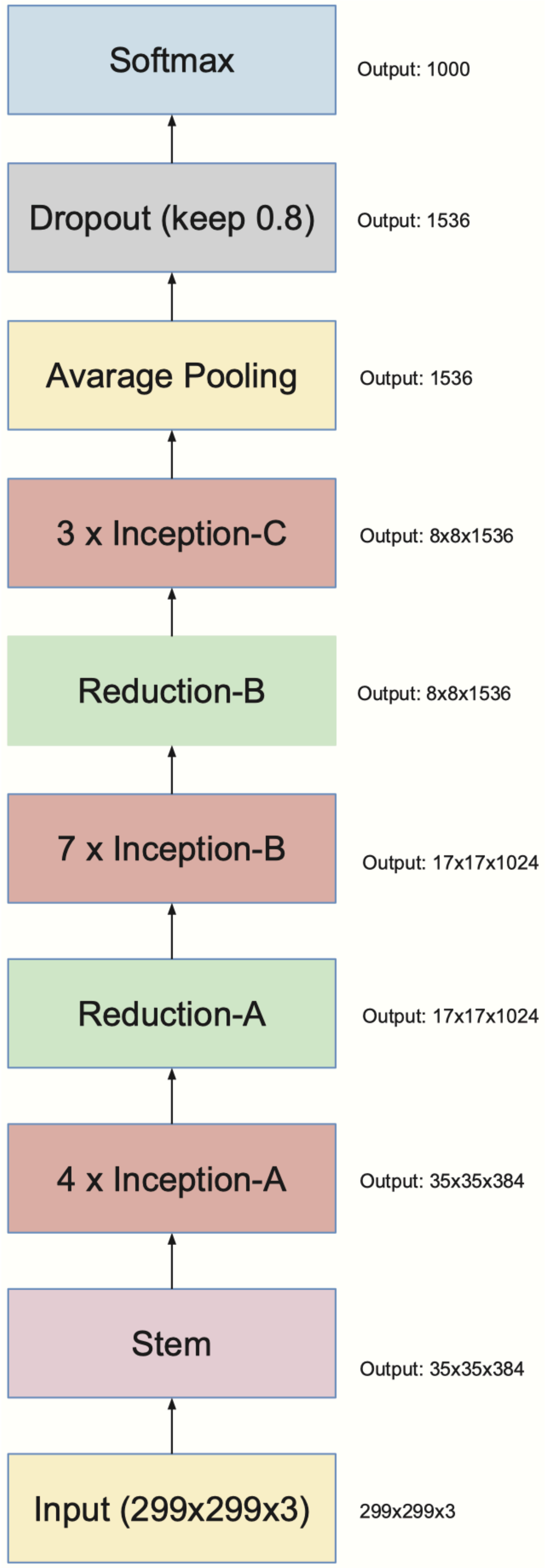
Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
Huang, Gao, et al. "Convolutional networks with dense connectivity." *IEEE transactions on pattern analysis and machine intelligence* (2019).

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

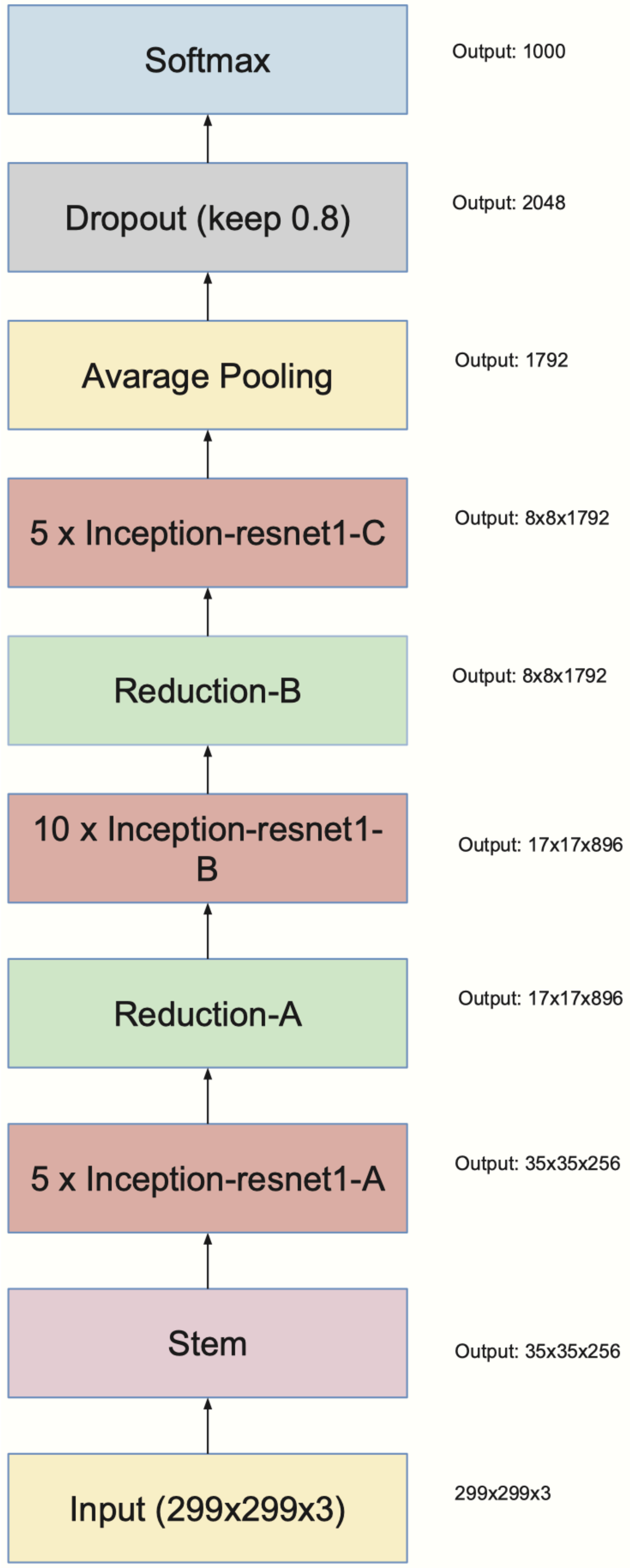
Boulder

[YouTube Video](#)

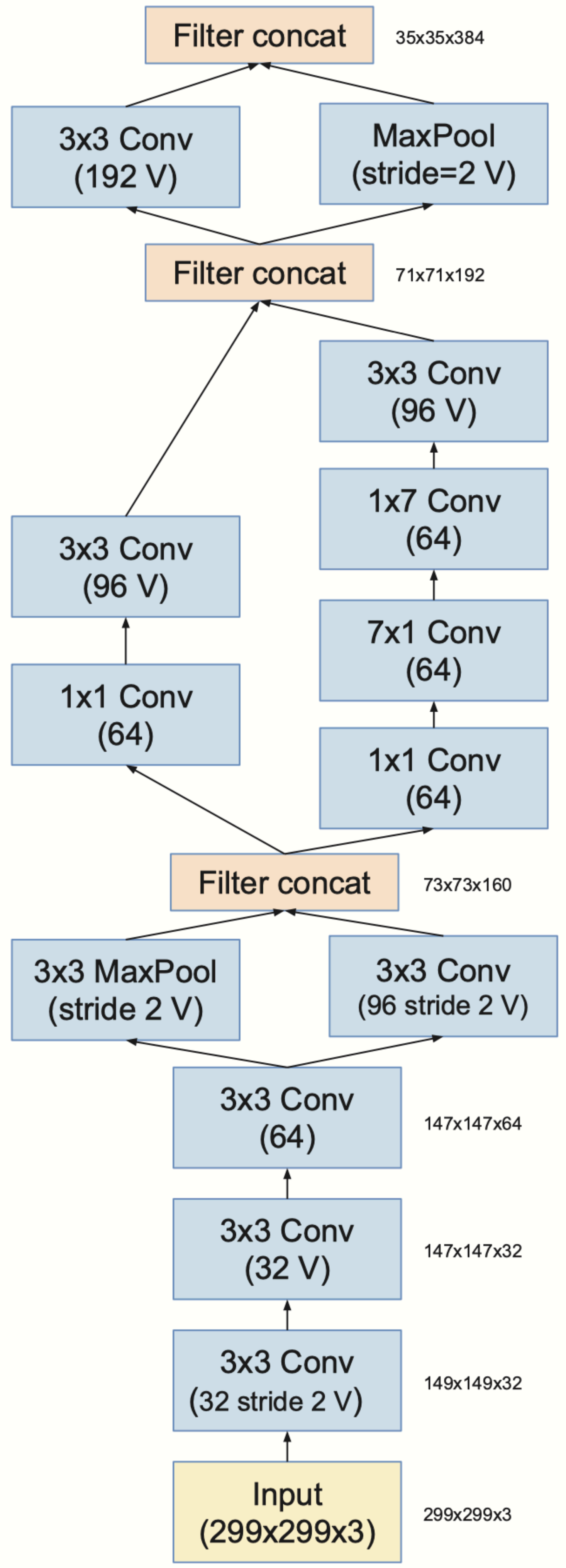
Inception-v4



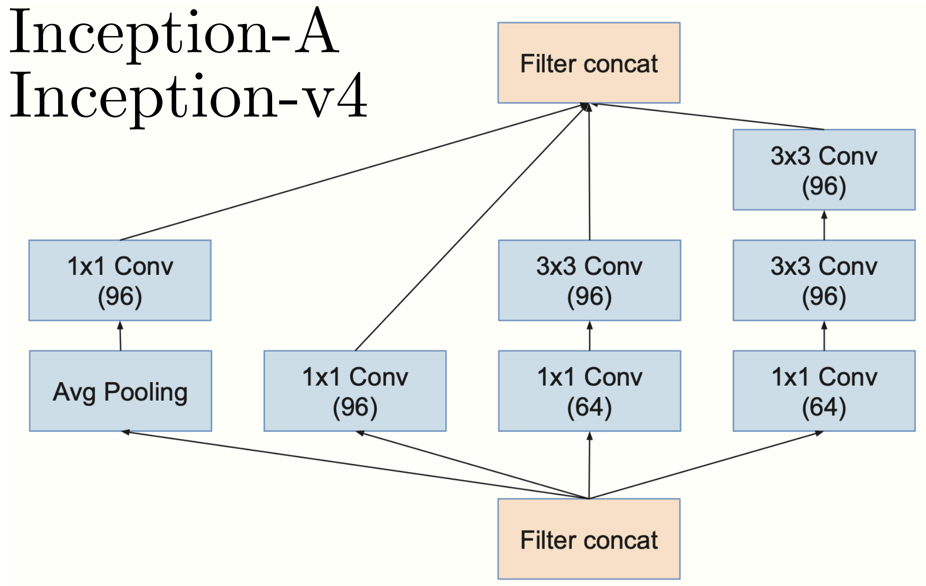
Inception-Resnet-v1 & -v2



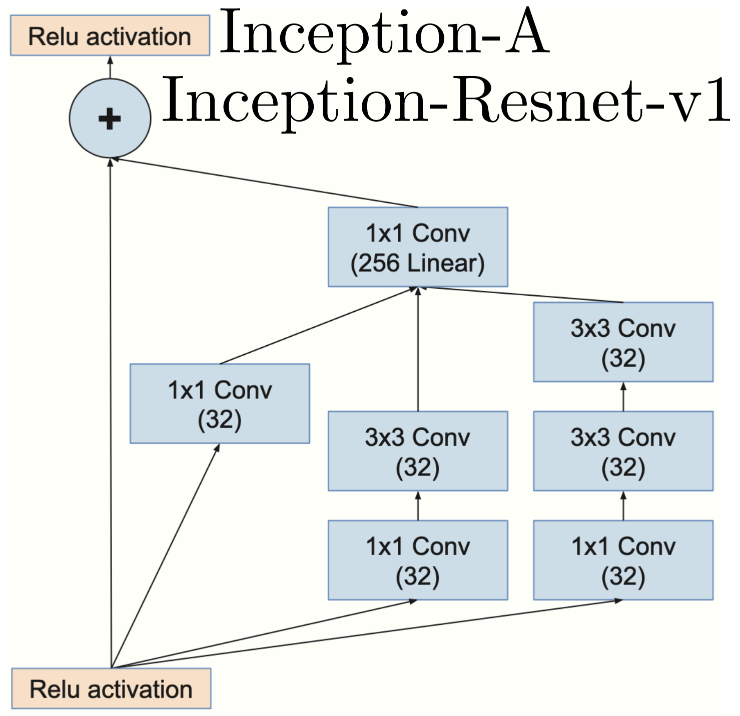
Stem: Inception-v4 & Inception-Resnet-v2



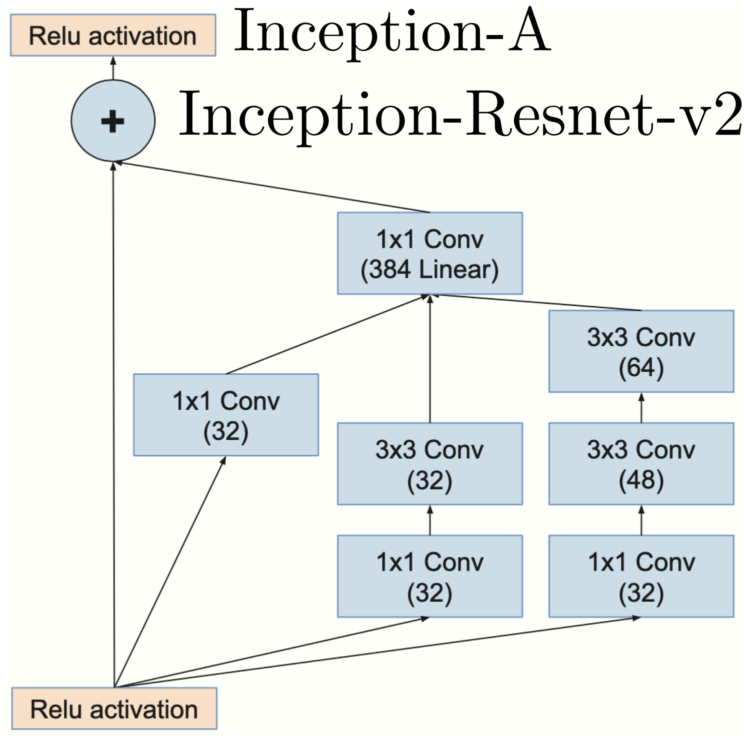
Inception-A Inception-v4



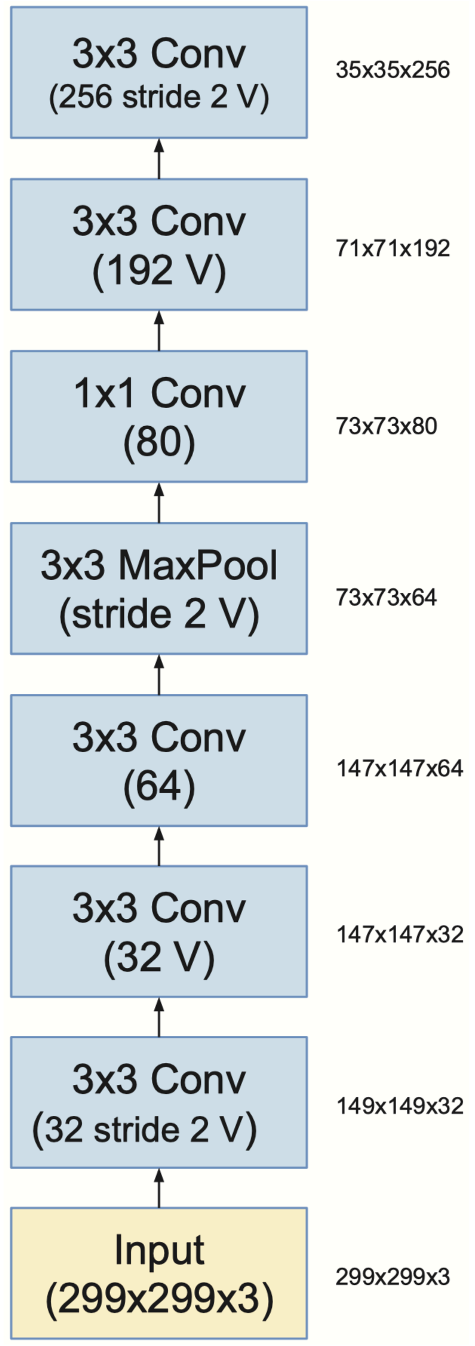
Inception-A Inception-Resnet-v1



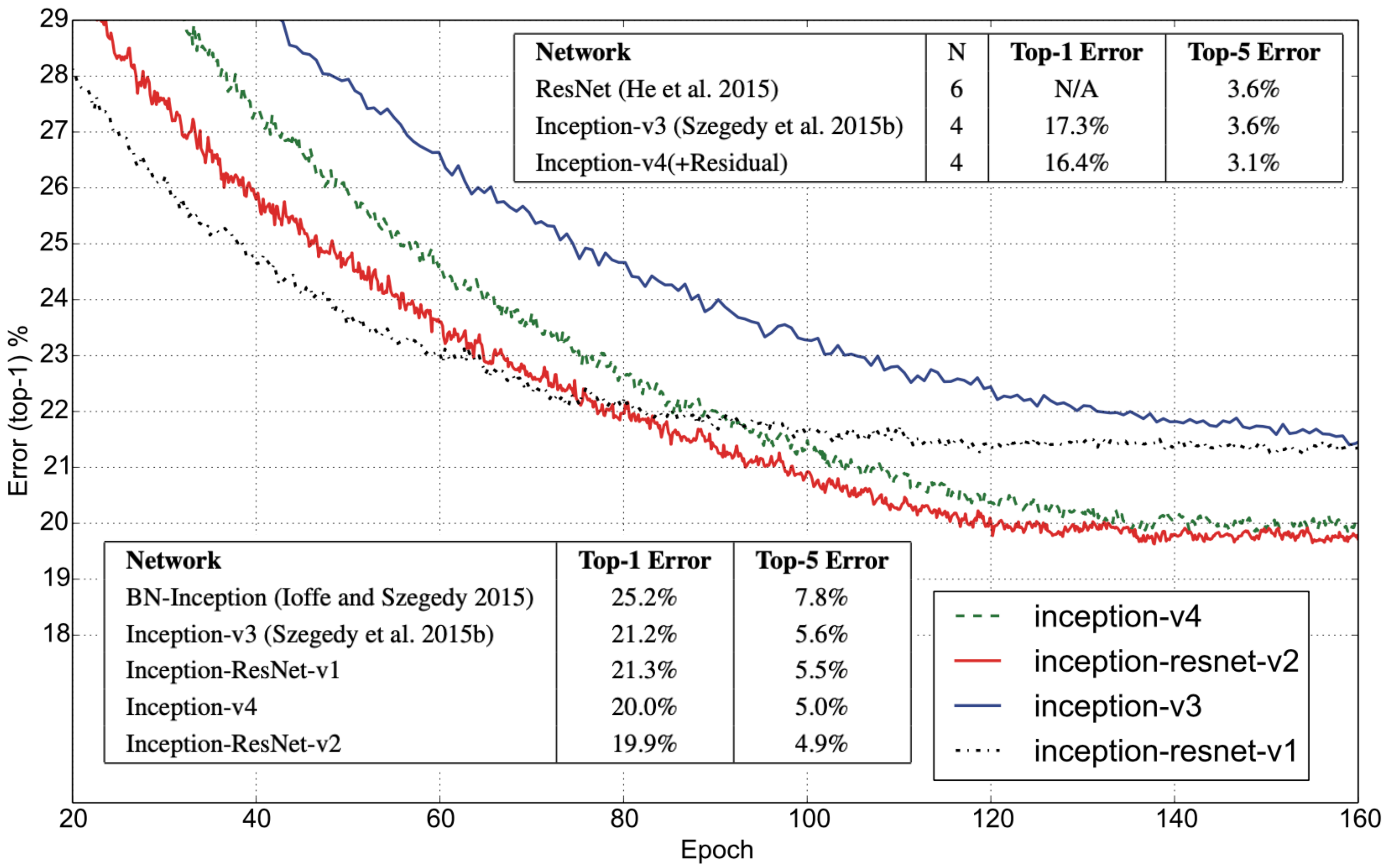
Inception-A Inception-Resnet-v2



Stem: Inception-Resnet-v1



Refer to the paper for the exact forms of Inception-B and Inception-C blocks



Network	N	Top-1 Error	Top-5 Error
ResNet (He et al. 2015)	6	N/A	3.6%
Inception-v3 (Szegedy et al. 2015b)	4	17.3%	3.6%
Inception-v4(+Residual)	4	16.4%	3.1%

Network	Top-1 Error	Top-5 Error
BN-Inception (Ioffe and Szegedy 2015)	25.2%	7.8%
Inception-v3 (Szegedy et al. 2015b)	21.2%	5.6%
Inception-ResNet-v1	21.3%	5.5%
Inception-v4	20.0%	5.0%
Inception-ResNet-v2	19.9%	4.9%

Szegedy, Christian, et al. "Inception-v4, inception-resnet and the impact of residual connections on learning." *arXiv preprint arXiv:1602.07261* (2016).

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 $(x,y) \sim P$)

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

P is unknown!

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

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

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

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

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)

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

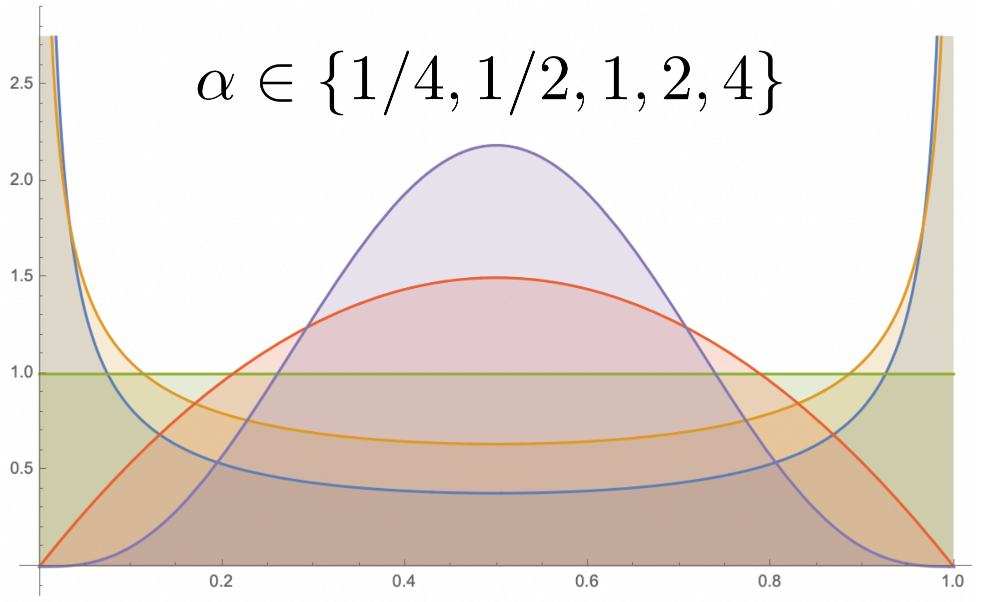
$\mathcal{D}_\nu := \{(\tilde{x}_i,\tilde{y}_i)\}_{i=1}^{m}$

mixup

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

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

$\alpha \rightarrow 0 \implies$ recovering ERM

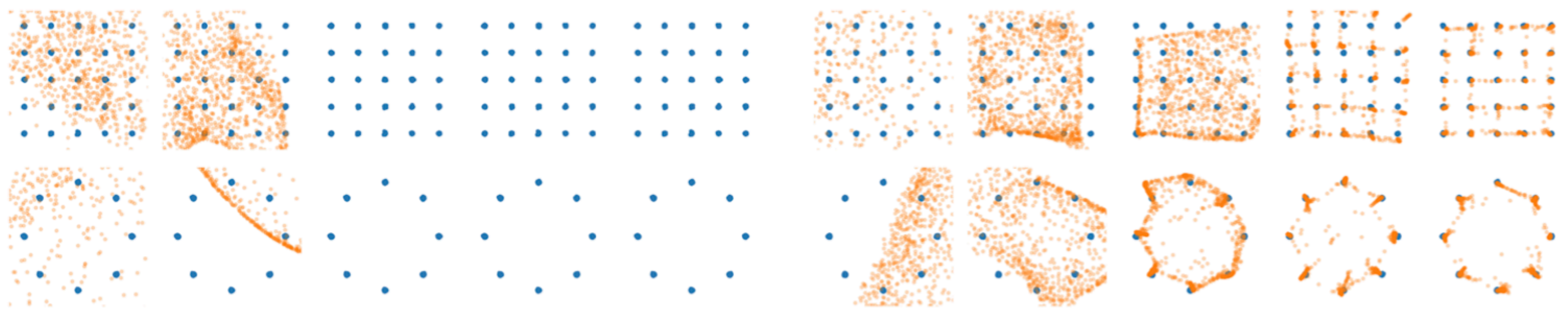


Model	Method	Epochs	Top-1 Error	Top-5 Error
ResNet-50	ERM (Goyal et al., 2017)	90	23.5	-
	mixup $\alpha = 0.2$	90	**23.3**	**6.6**
ResNet-101	ERM (Goyal et al., 2017)	90	22.1	-
	mixup $\alpha = 0.2$	90	**21.5**	**5.6**
ResNeXt-101 32*4d	ERM (Xie et al., 2016)	100	21.2	-
	ERM	90	21.2	5.6
	mixup $\alpha = 0.4$	90	**20.7**	**5.3**
ResNeXt-101 64*4d	ERM (Xie et al., 2016)	100	20.4	5.3
	mixup $\alpha = 0.4$	90	**19.8**	**4.9**
ResNet-50	ERM	200	23.6	7.0
	mixup $\alpha = 0.2$	200	**22.1**	**6.1**
ResNet-101	ERM	200	22.0	6.1
	mixup $\alpha = 0.2$	200	**20.8**	**5.4**
ResNeXt-101 32*4d	ERM	200	21.3	5.9
	mixup $\alpha = 0.4$	200	**20.1**	**5.0**

mixup GANs

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

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



Zhang, Hongyi, et al. "mixup: Beyond empirical risk minimization." *arXiv preprint arXiv:1710.09412* (2017).

Boulder

Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour

Stochastic Gradient Descent (SGD)

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

$L(w) \rightarrow$ loss function
$w \rightarrow$ weights
$X \rightarrow$ labeled training set
$l(x;w) \rightarrow$ loss computed from sample x and its corresponding label (e.g., cross-entropy + a regularization on w)

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

$\mathcal{B} \rightarrow$ mini-batch sampled from X
$|\mathcal{B}| \rightarrow$ mini-batch size
$k \rightarrow$ number of GPUs in a server
$n \rightarrow$ number of samples per GPU
Baseline: $k = 8$, $n = 32$, and $|\mathcal{B}| = kn = 256$.
$\eta = 0.1 \rightarrow$ learning rate

Learning rates for large mini-batches

data parallelism (multiple servers)
Linear Scaling Rule: $\eta = 0.1\frac{|\mathcal{B}|}{256} = 0.1\frac{kn}{256}$

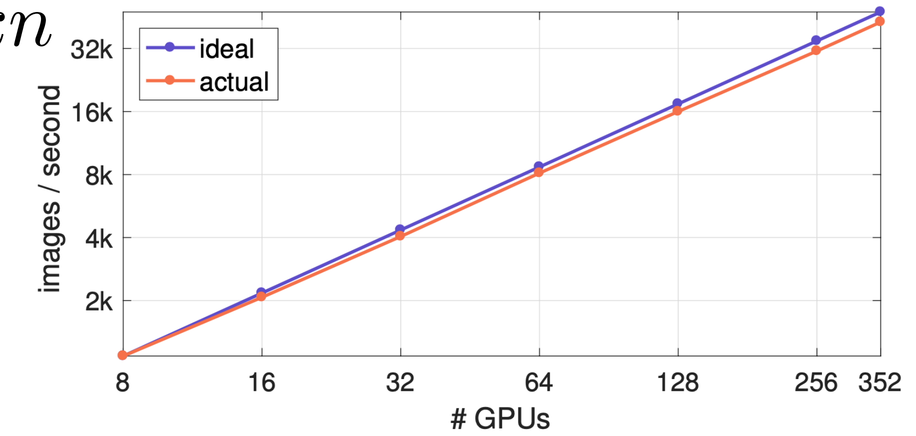
> ***Linear Scaling Rule:*** *When the minibatch size is multiplied by k, multiply the learning rate by k.*

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

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

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

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

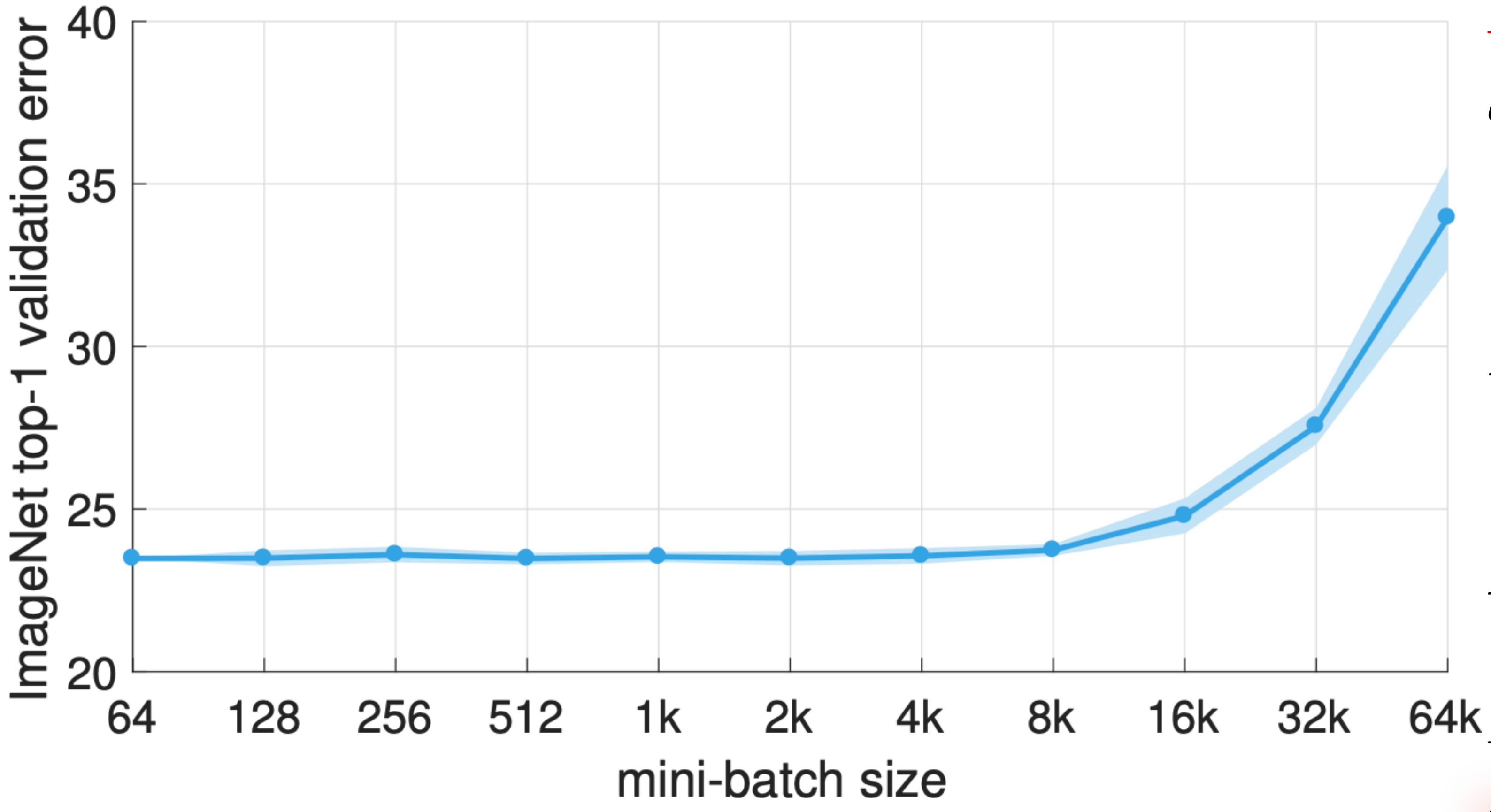


$\hat{w}_{t+1} \neq w_{t+k}$
$\nabla l(x;w_t) \approx \nabla l(x;w_{t+j}) \rightarrow$ assume
$\hat{\eta} = k\eta \implies \hat{w}_{t+1} \approx w_{t+k}$

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 η and increment it by a constant amount at each iteration until it reaches $\hat{\eta} = k\eta$ 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 $\mathcal{B}$

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

$X^n \rightarrow$ all distinct subsets of size n drawn from the original training set X
$L(w)$ changes as n changes!

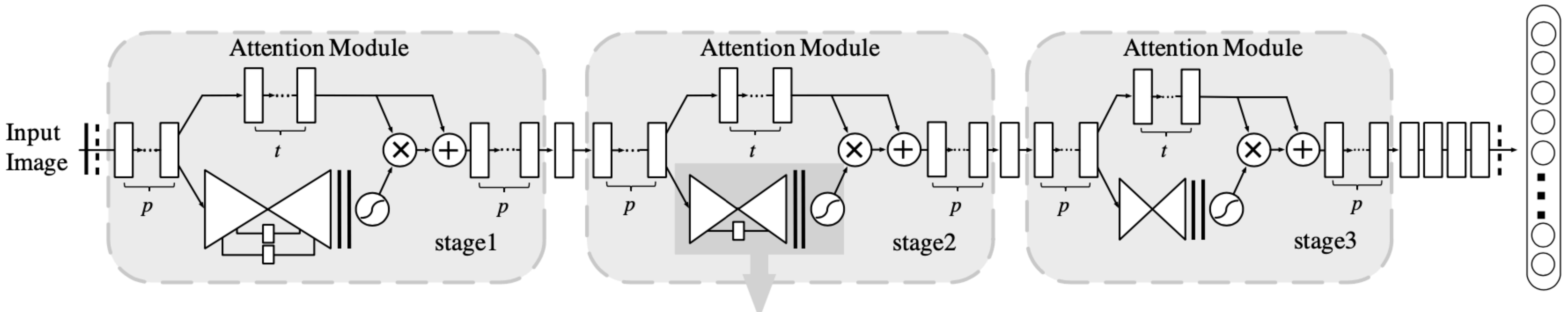
Subtleties and Pitfalls of Distributed SGD (see the paper!)

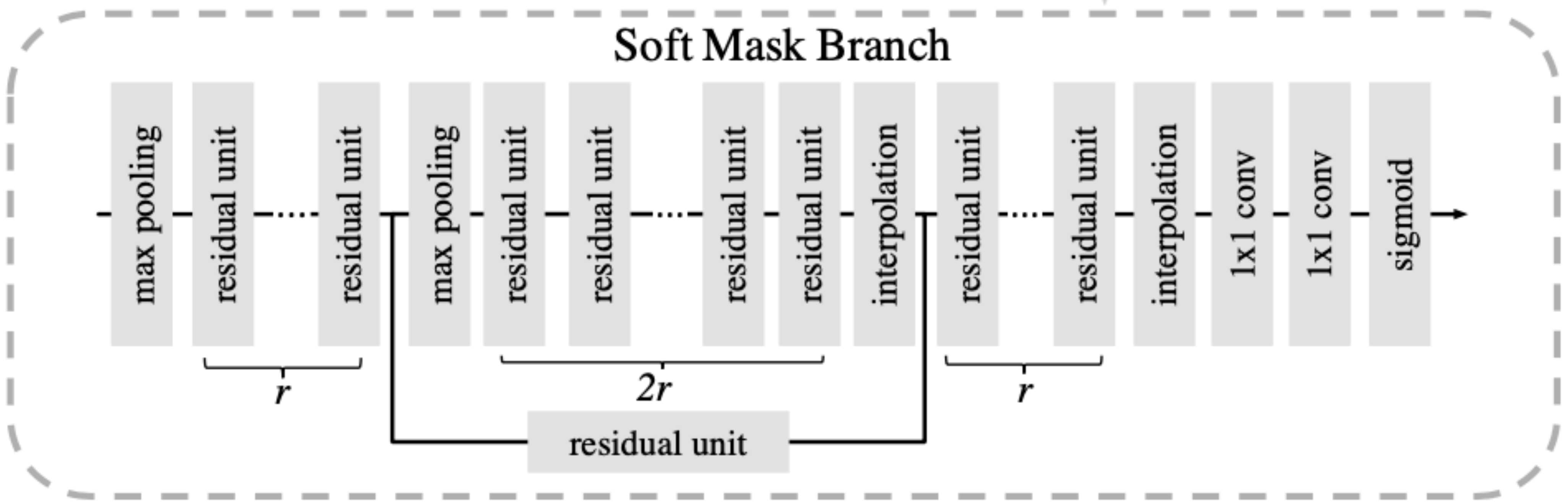
Batch normalization with large mini-batches

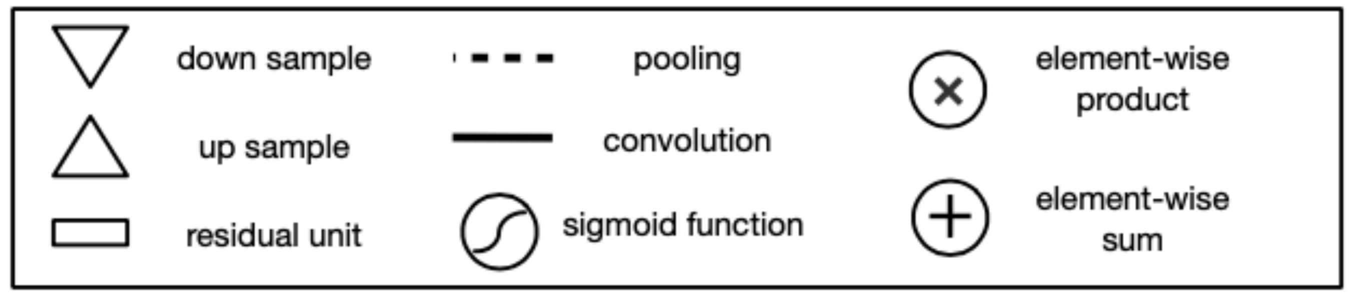
```
The BN statistics should not be computed across all workers, not only for the sake of reducing
communication, but also for maintaining the same underlying loss function being optimized.
```


Goyal, Priya, et al. "Accurate, large minibatch sgd: Training imagenet in 1 hour." *arXiv preprint arXiv:1706.02677* (2017).

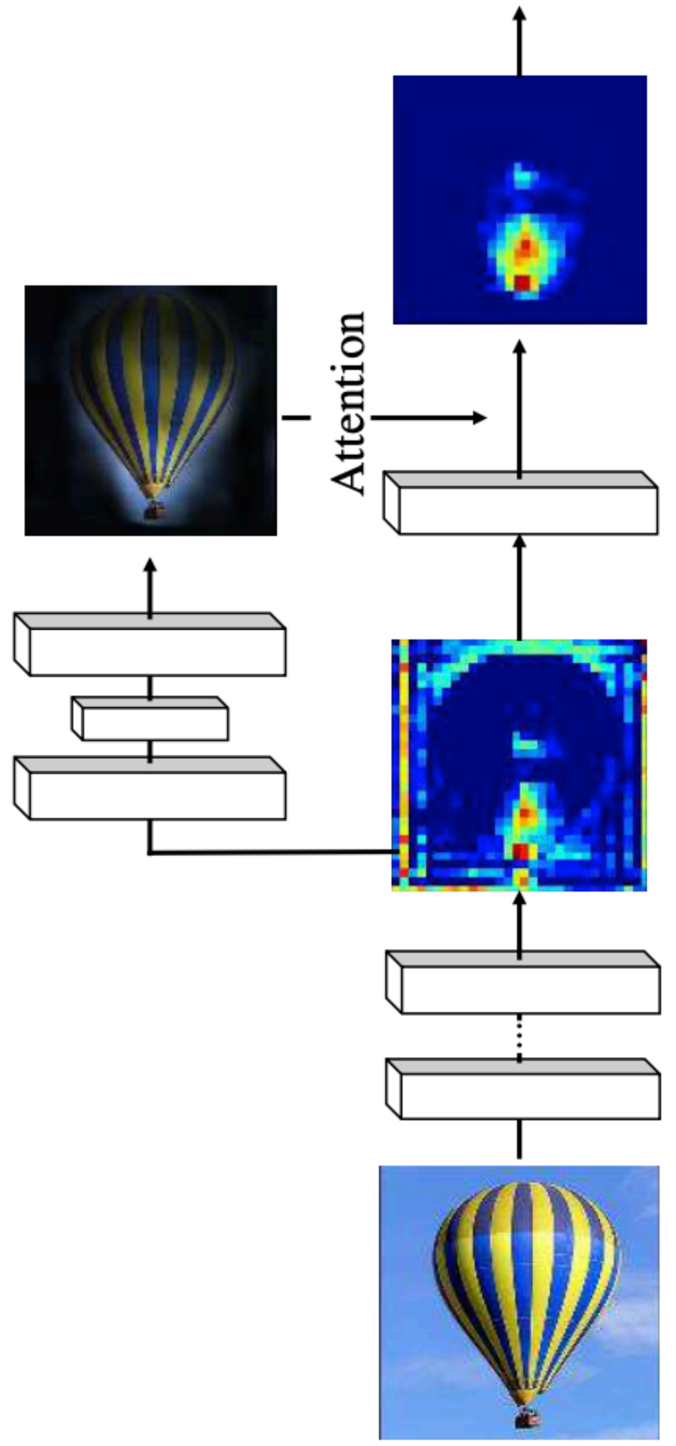
Residual Attention Network for Image Classification







$$\{p = 1, t = 2, r = 1\}$$



Attention Module

– Mask branch (bottom-up top-down structure)

– Trunk branch (performs feature processing)

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

$i \rightarrow$ ranges over all spatial positions

$c \rightarrow$ index of the channel

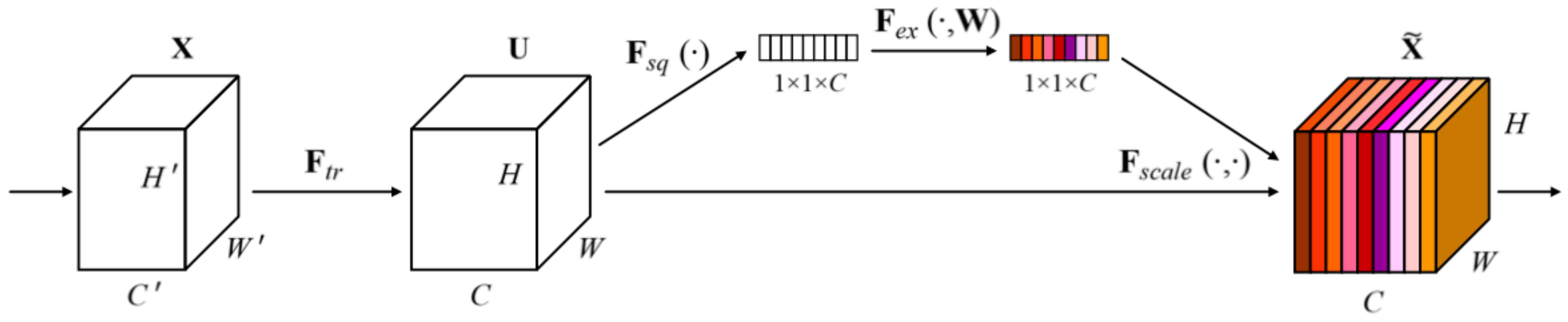
$$f_1(x_{i,c}) = \frac{1}{1 + exp(-x_{i,c})} \rightarrow \text{Mixed Spatial and Channel Attention}$$

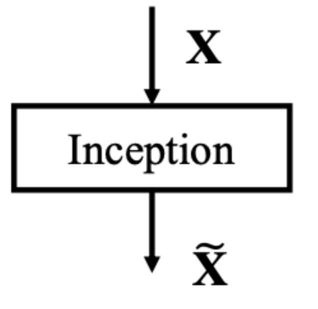
Network	params×10^6	FLOPs×10^9	Test Size	Top-1 err. (%)	Top-5 err. (%)
ResNet-152 [10]	60.2	11.3	224 × 224	22.16	6.16
Attention-56	31.9	6.3	224 × 224	**21.76**	**5.9**
ResNeXt-101 [36]	44.5	7.8	224 × 224	21.2	5.6
AttentionNeXt-56	31.9	6.3	224 × 224	**21.2**	**5.6**
Inception-ResNet-v1 [32]	-	-	299 ×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

Wang, Fei, et al. "Residual attention network for image classification." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

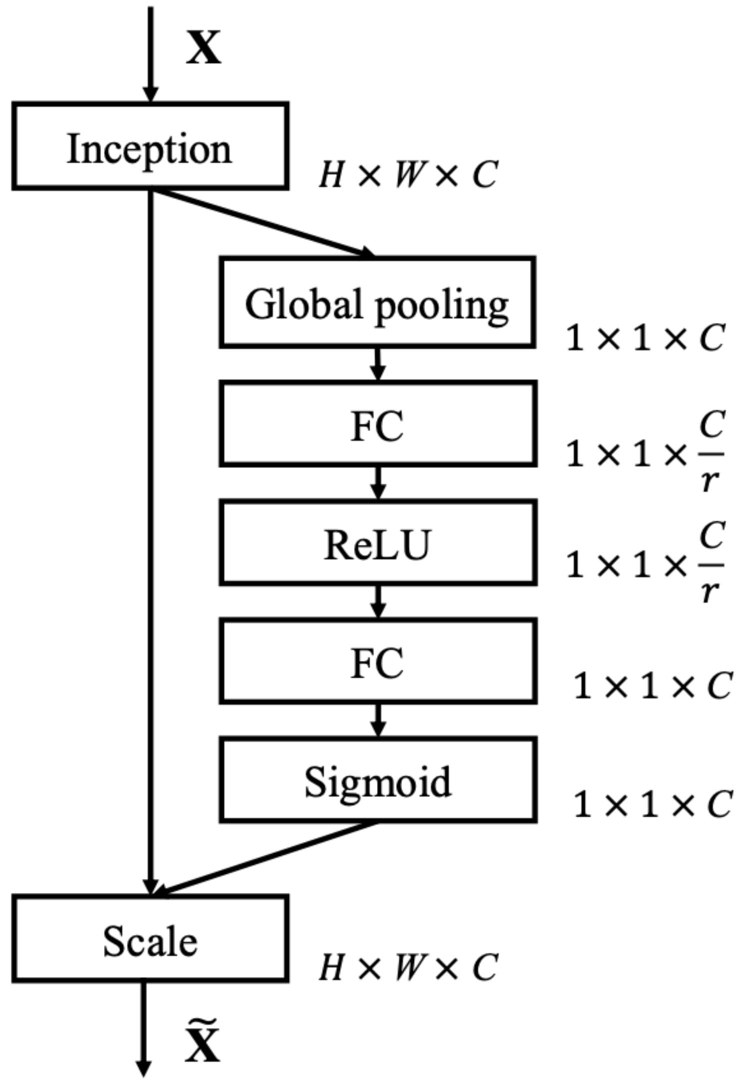
Squeeze-and-Excitation Networks

YouTube Video

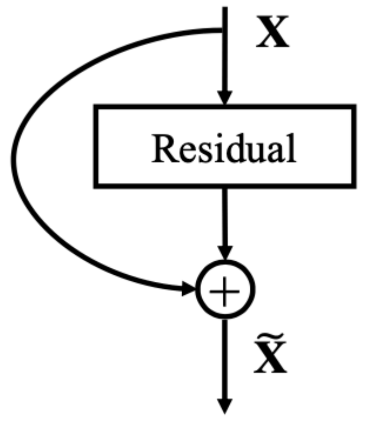




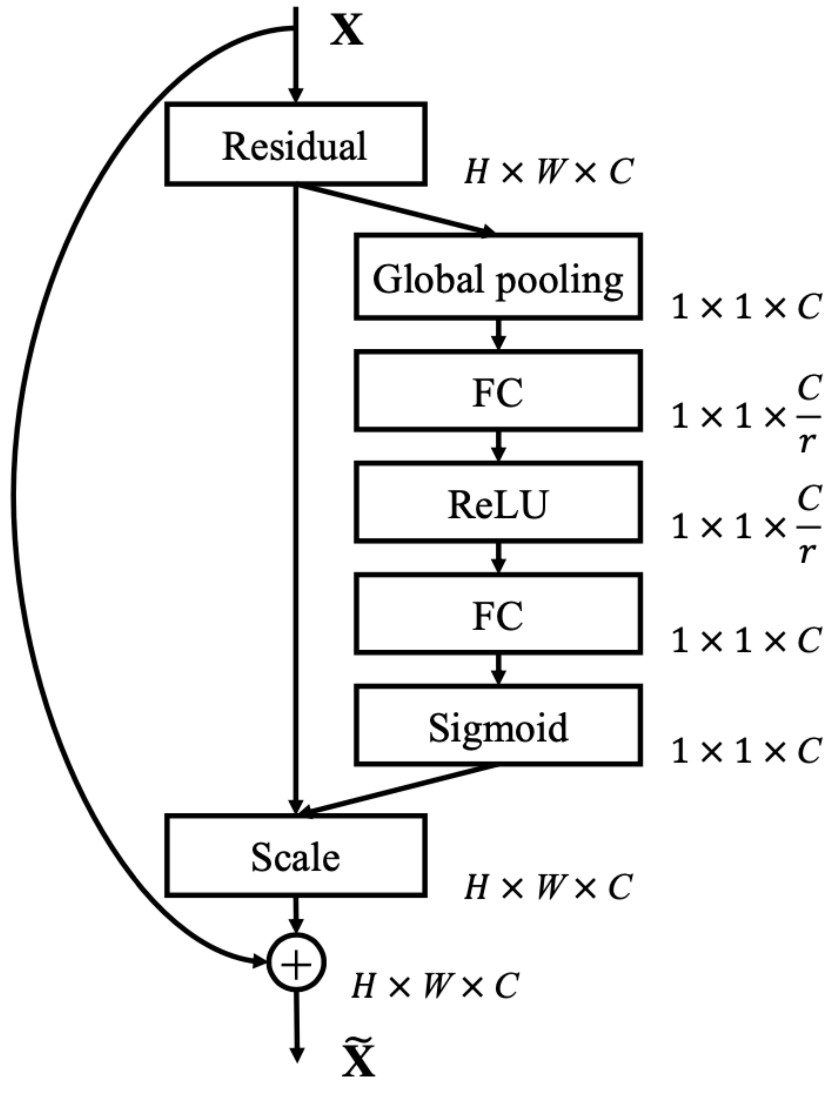
Inception Module



SE-Inception Module



ResNet Module



SE-ResNet Module

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

$F_{tr} : X \mapsto U$

$V = [v_1, \ldots, v_C] \rightarrow$ learned set of filter kernels

$v_c \rightarrow$ parameters of the c-th filter

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

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

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

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

Squeeze

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

Excitation

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

$W_1 \in \mathbb{R}^{\frac{C}{r} \times C}$ $\qquad W_2 \in \mathbb{R}^{C \times \frac{C}{r}}$

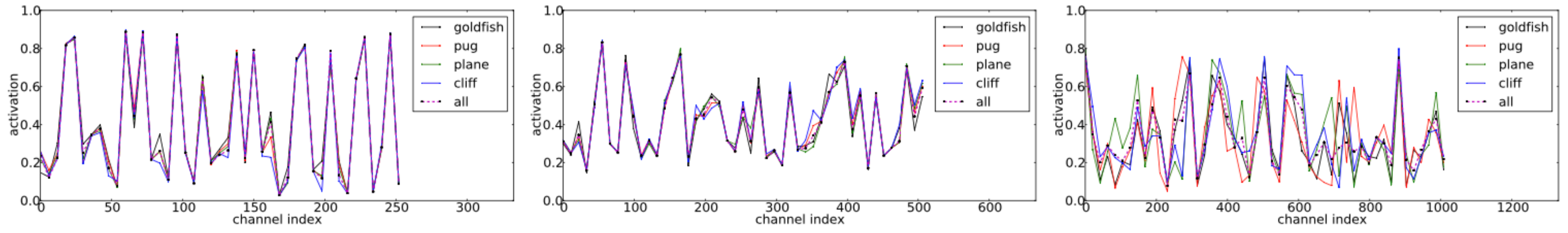
$\tilde{x}_c = F_{scale}(u_c, s_c) = s_c u_c$

$u_c \in \mathbb{R}^{H \times W}$

$\tilde{X} = [\tilde{x}_1, \ldots, \tilde{x}_C]$

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

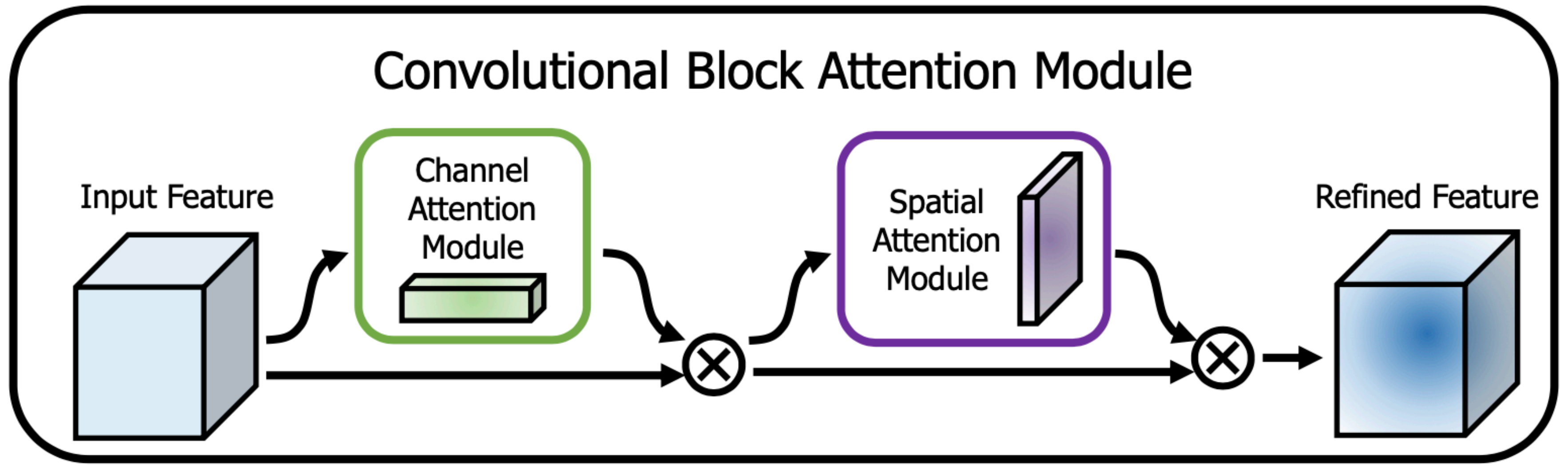
	original		re-implementation			SENet		
	top-1 err.	top-5 err.	top-1err.	top-5 err.	GFLOPs	top-1 err.	top-5 err.	GFLOPs
ResNet-50 [10]	24.7	7.8	24.80	7.48	3.86	$23.29_{(1.51)}$	$6.62_{(0.86)}$	3.87
ResNet-101 [10]	23.6	7.1	23.17	6.52	7.58	$22.38_{(0.79)}$	$6.07_{(0.45)}$	7.60
ResNet-152 [10]	23.0	6.7	22.42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

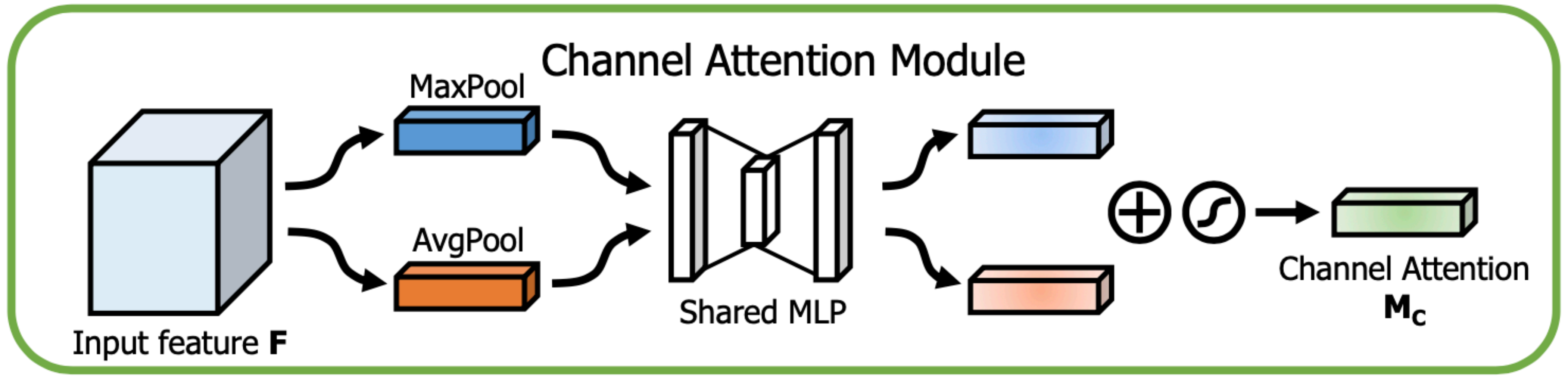


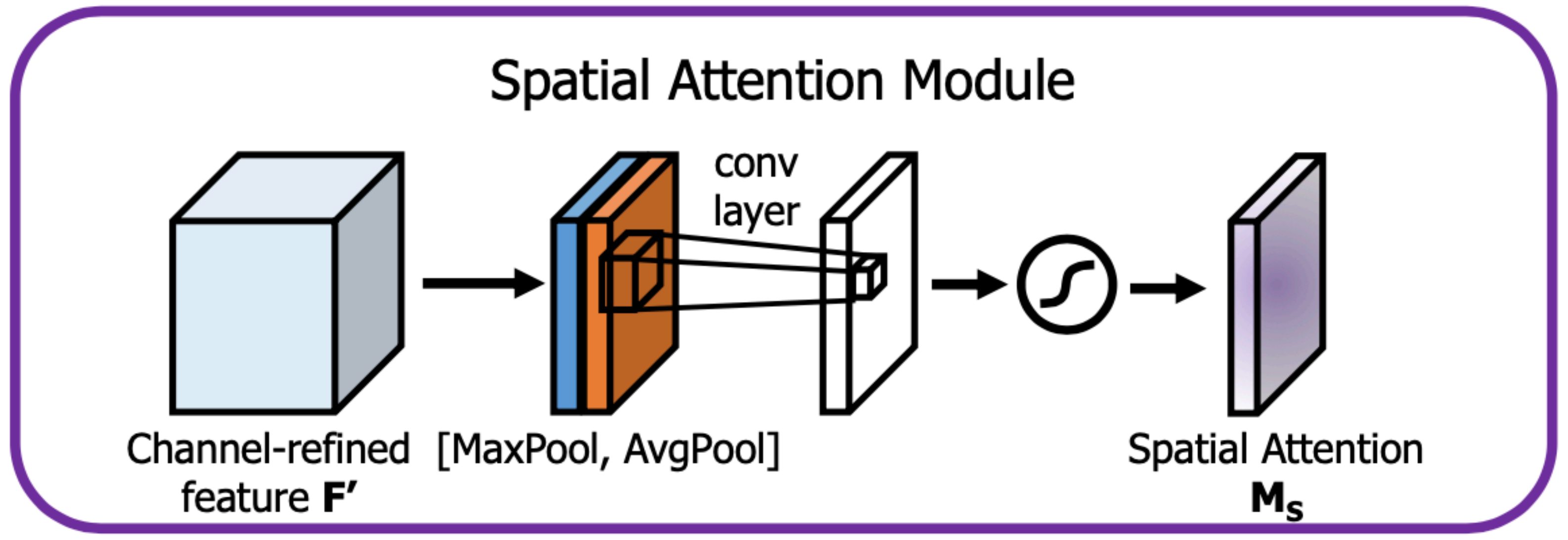
(a) SE_2_3 (b) SE_3_4 (c) SE_4_6

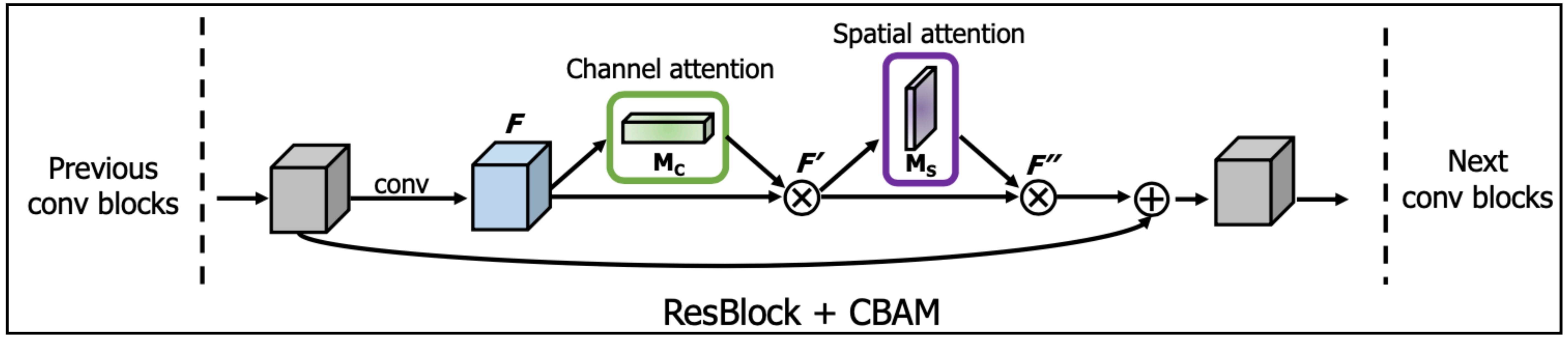
Hu, Jie, Li Shen, and Gang Sun. "Squeeze-and-excitation networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

CBAM: Convolutional Block Attention Module









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

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

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

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

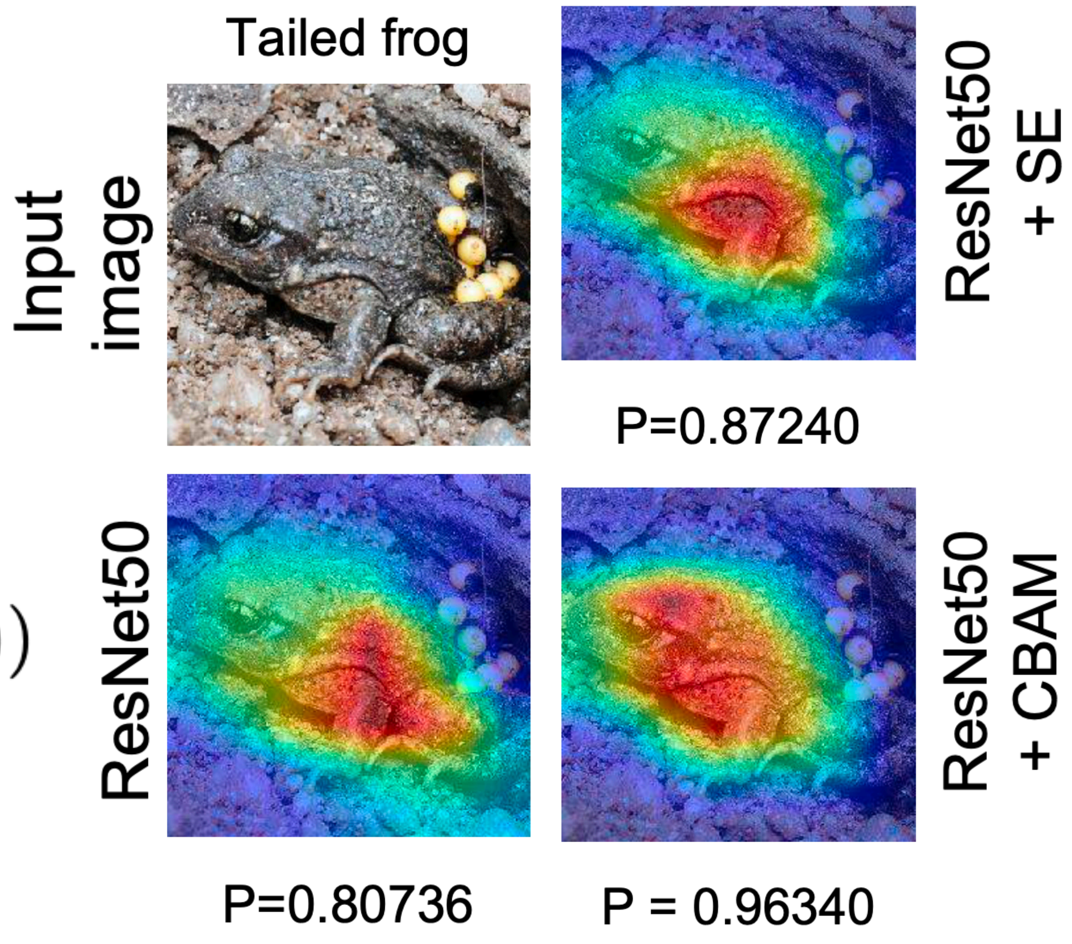
$F'' = M_s(F') \odot F'$

$$\mathbf{M_c}(\mathbf{F}) = \sigma(MLP(AvgPool(\mathbf{F})) + MLP(MaxPool(\mathbf{F})))$$
$$= \sigma(\mathbf{W_1}(\mathbf{W_0}(\mathbf{F^c_{avg}})) + \mathbf{W_1}(\mathbf{W_0}(\mathbf{F^c_{max}}))),$$

$\mathbf{W_0} \in \mathbb{R}^{C/r\times C}$, and $\mathbf{W_1} \in \mathbb{R}^{C\times C/r}$

$r \rightarrow$ reduction ratio

$$\mathbf{M_s}(\mathbf{F}) = \sigma(f^{7\times 7}([AvgPool(\mathbf{F}); MaxPool(\mathbf{F})]))$$
$$= \sigma(f^{7\times 7}([\mathbf{F^s_{avg}}; \mathbf{F^s_{max}}])),$$

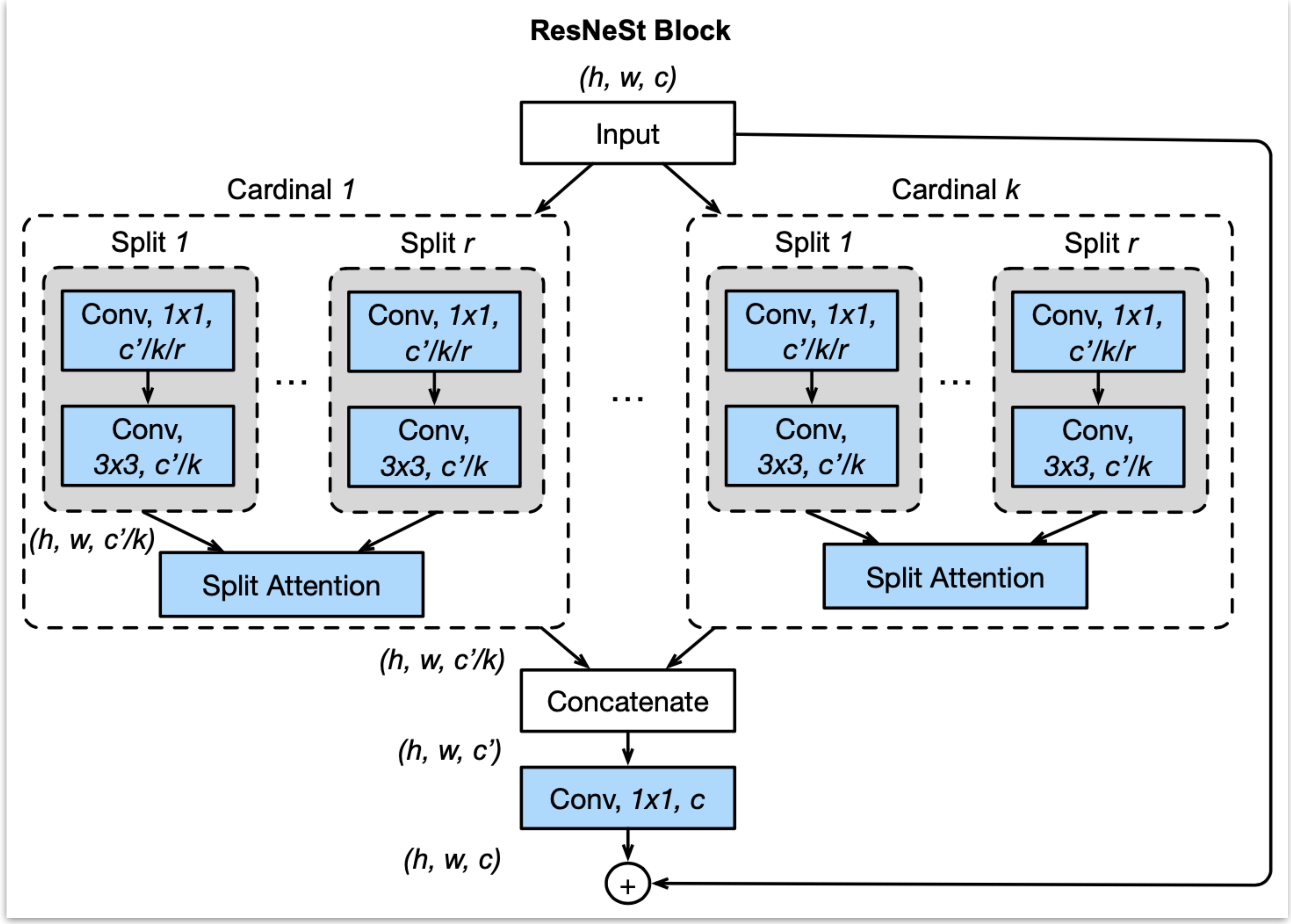


Architecture	Param.	GFLOPs	Top-1 Error (%)	Top-5 Error (%)
ResNet18 [5]	11.69M	1.814	29.60	10.55
ResNet18 [5] + SE [28]	11.78M	1.814	29.41	10.22
ResNet18 [5] + CBAM	11.78M	1.815	**29.27**	**10.09**
ResNet34 [5]	21.80M	3.664	26.69	8.60
ResNet34 [5] + SE [28]	21.96M	3.664	26.13	8.35
ResNet34 [5] + CBAM	21.96M	3.665	**25.99**	**8.24**
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	**22.66**	**6.31**
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	**21.51**	**5.69**
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	**26.10**	**8.43**
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	**24.84**	**7.63**
ResNeXt50 [7] (32x4d)	25.03M	3.768	22.85	6.48
ResNeXt50 [7] (32x4d) + SE [28]	27.56M	3.771	**21.91**	6.04
ResNeXt50 [7] (32x4d) + CBAM	27.56M	3.774	21.92	**5.91**
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	**21.07**	**5.59**

Woo, Sanghyun, et al. "Cbam: Convolutional block attention module." *Proceedings of the European conference on computer vision (ECCV)*. 2018.

Boulder

ResNeSt: Split-Attention Networks



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

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

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

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

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

global contextual information with embedded channel-wise statistics

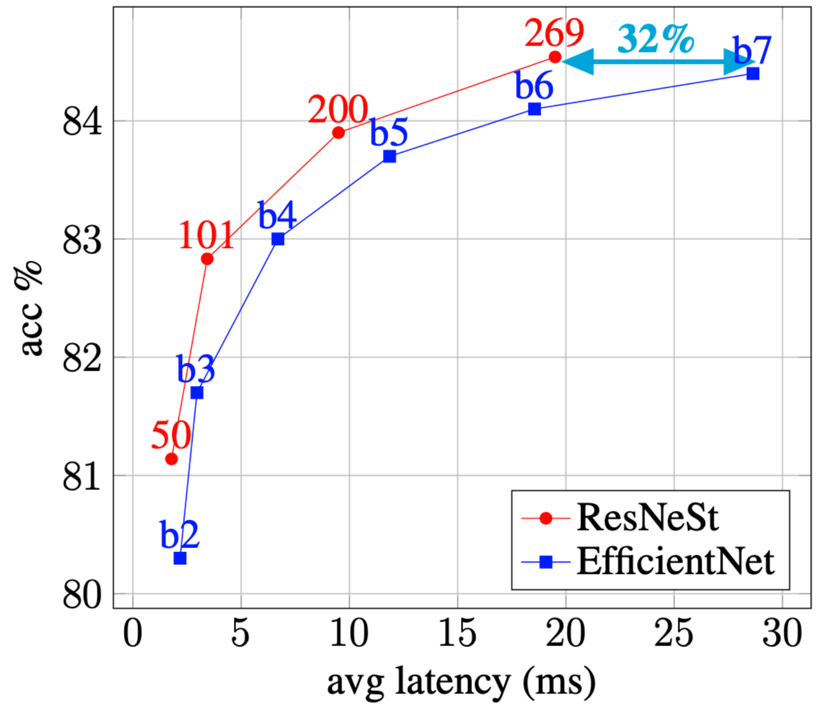
$\mathcal{G}_i(s^j) \in \mathbb{R}^c \rightarrow$ two fully connected layers with ReLU activation

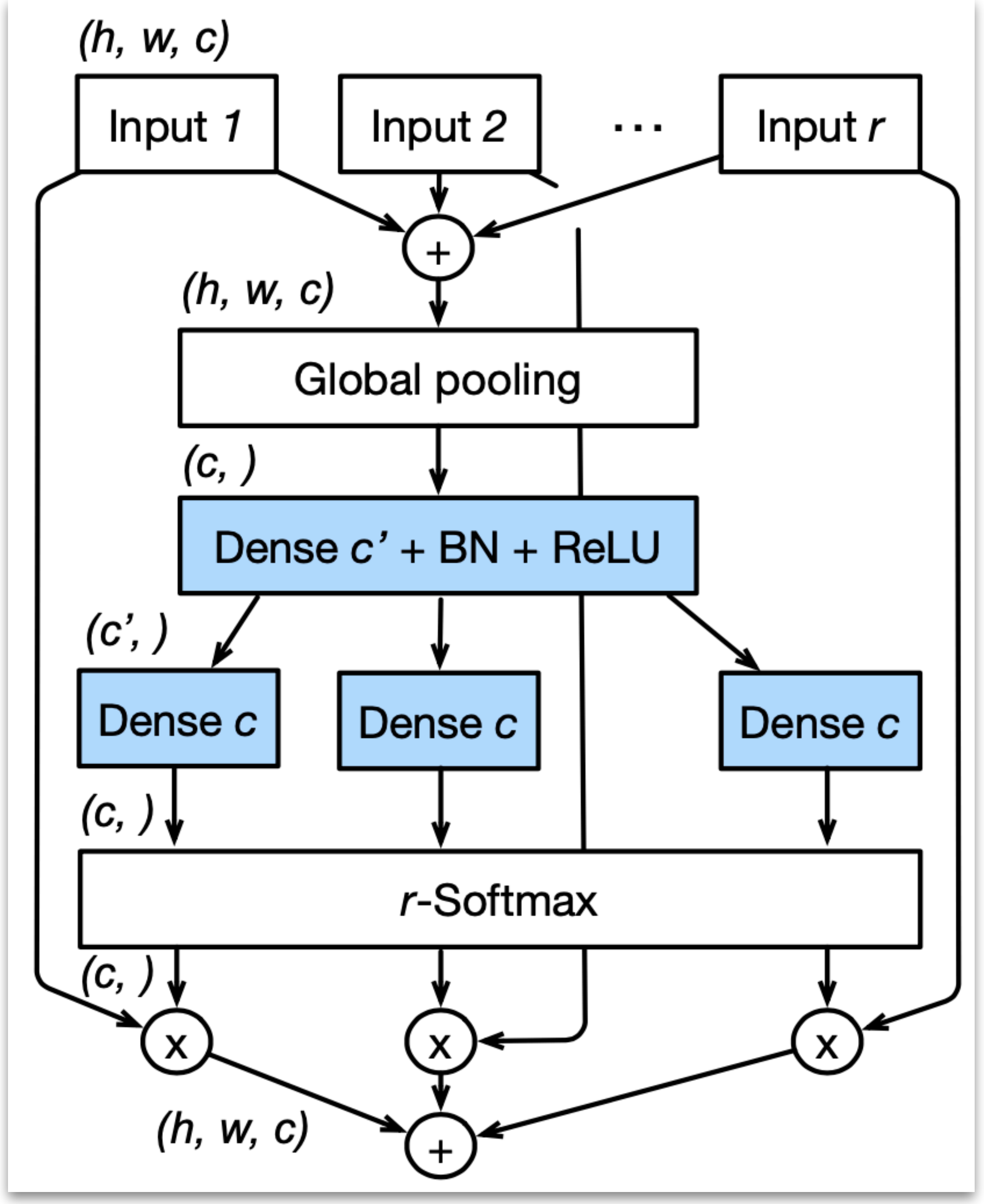
$a_i^j \in \mathbb{R}^c \rightarrow$ soft assignment weights

See the paper for an efficient radix-major implementation!

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





ResNeSt Block

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

$\text{Y} = \text{V} + \text{X}$

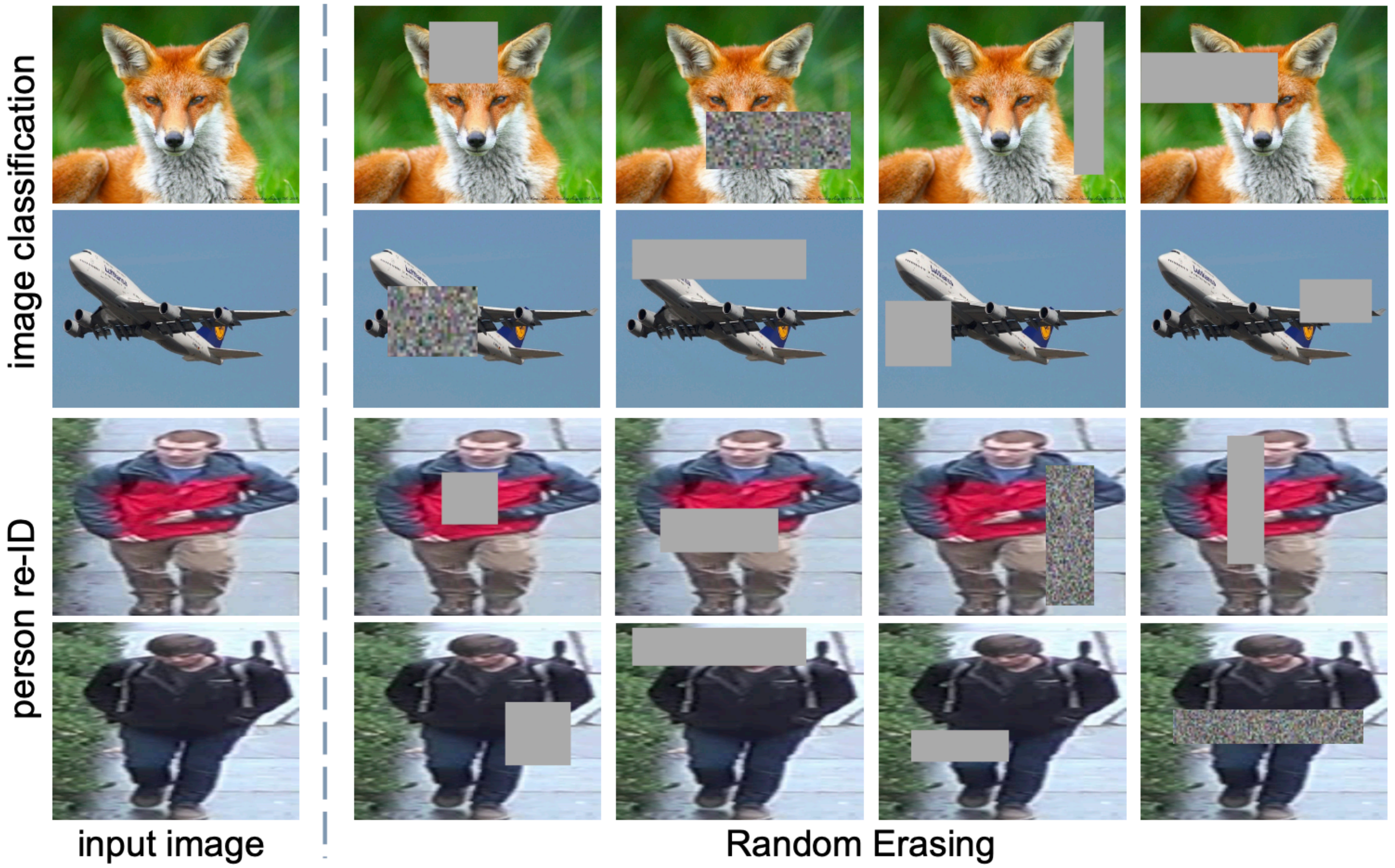
Zhang, Hang, et al. "Resnest: Split-attention networks." *arXiv preprint arXiv:2004.08955* (2020).

Random Erasing Data Augmentation

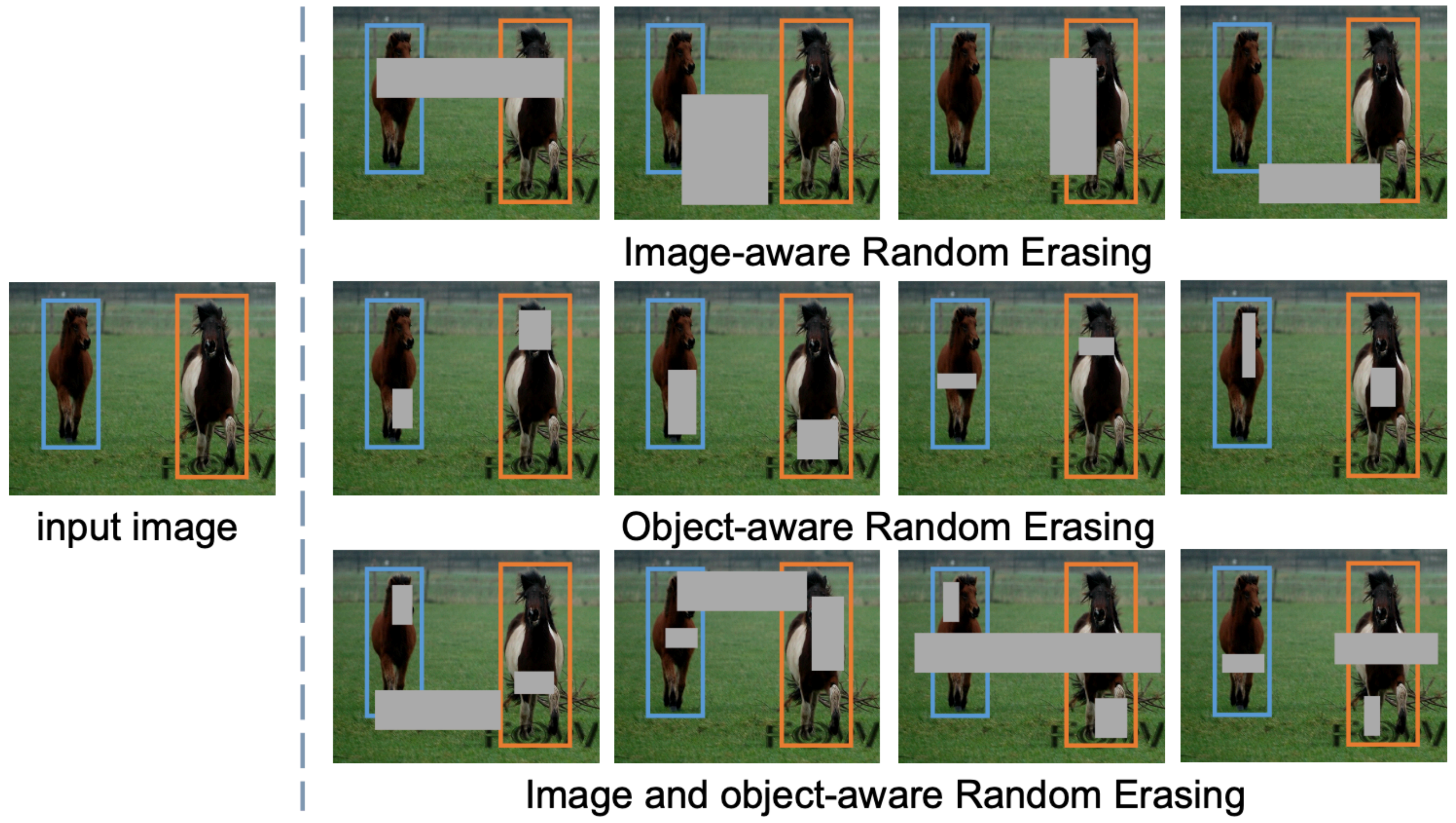
Algorithm 1: Random Erasing Procedure

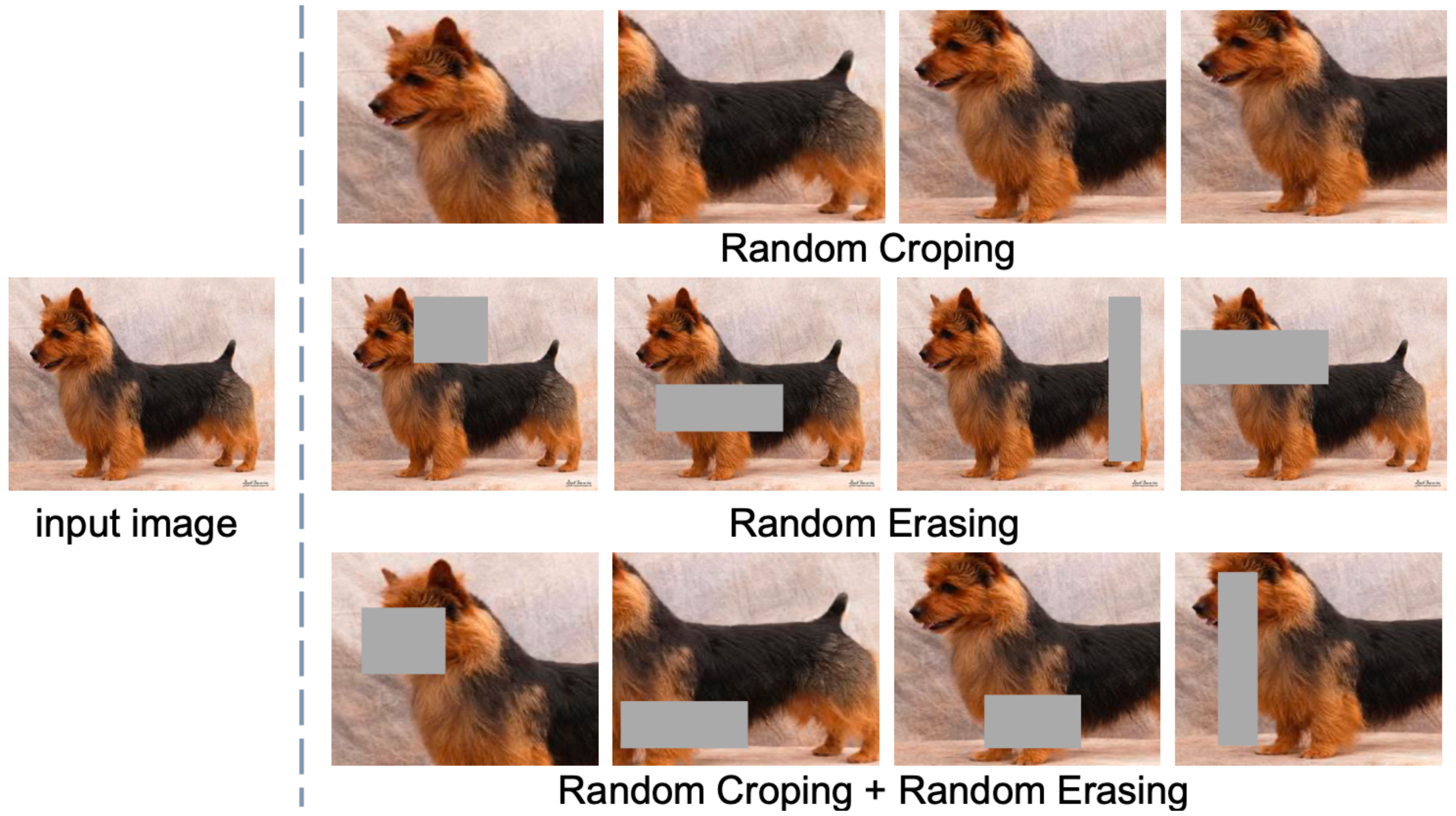
Input : Input image I;
Image size W and H;
Area of image S;
Erasing probability p;
Erasing area ratio range s_l and s_h;
Erasing aspect ratio range r_1 and r_2.
Output: Erased image I^*.
Initialization: $p_1 \leftarrow$ Rand $(0, 1)$.

if $p_1 \geq p$ **then**
  $I^* \leftarrow I$;
  return I^*.
else
  while *True* **do**
    $S_e \leftarrow \text{Rand}\ (s_l, s_h) \times S$;
    $r_e \leftarrow \text{Rand}\ (r_1, r_2)$;
    $H_e \leftarrow \sqrt{S_e \times r_e},\ W_e \leftarrow \sqrt{\frac{S_e}{r_e}}$;
    $x_e \leftarrow \text{Rand}\ (0, W),\ y_e \leftarrow \text{Rand}\ (0, H)$;
    if $x_e + W_e \leq W$ *and* $y_e + H_e \leq H$ **then**
      $I_e \leftarrow (x_e, y_e, x_e + W_e, y_e + H_e)$;
      $I(I_e) \leftarrow \text{Rand}\ (0, 255)$;
      $I^* \leftarrow I$;
      return I^*.
    end
  end
end



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





Zhong, Zhun, et al. "Random Erasing Data Augmentation." *AAAI*. 2020.

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

$\underbrace{h(T) \rightarrow \text{output}}$ 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 $a(T)$.

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

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

Continuous Depth Residual Networks

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

	Test Error	# Params	Memory	Time
1-Layer MLP†	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	$\mathbf{\mathcal{O}(1)}$	$\mathcal{O}(\tilde{L})$

Performance on MNIST.

Normalizing Flows

Change of variable theorem

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

cubic cost!

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

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

$\mathrm{KL}(q(x) \,||\, p(x)) \rightarrow$ loss function

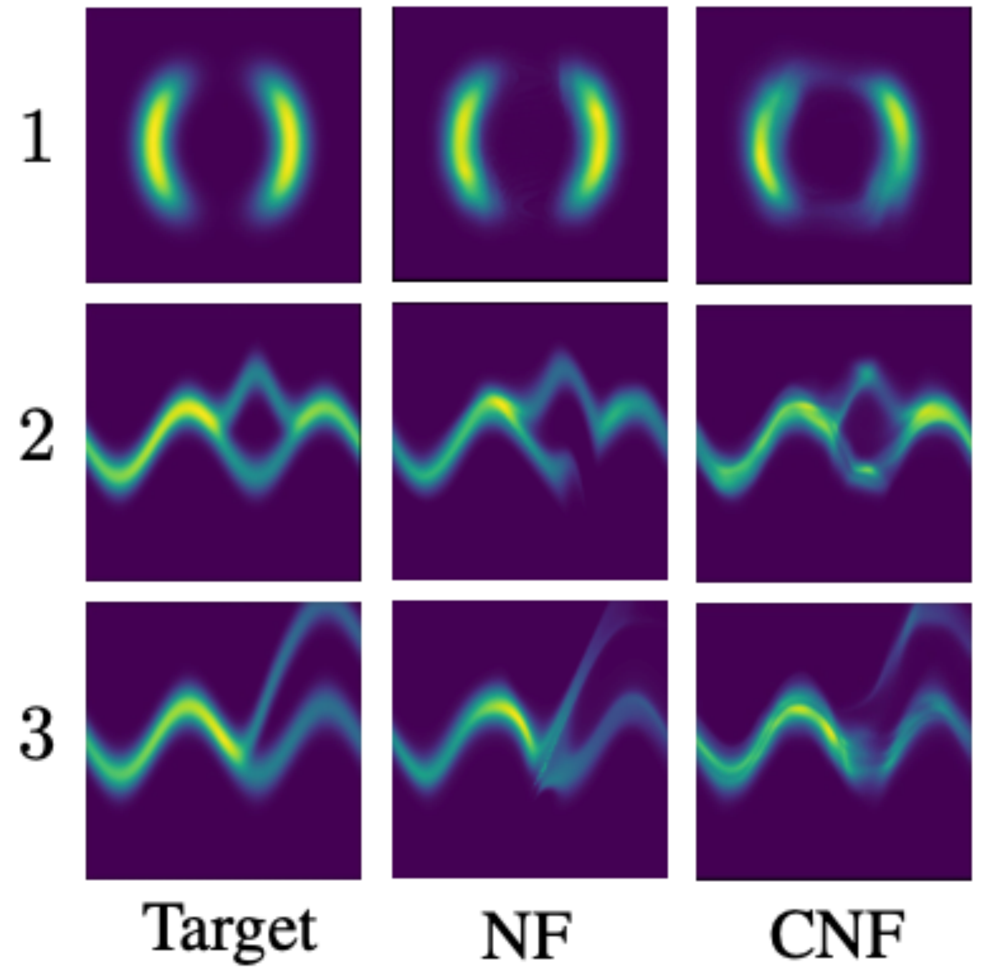
target density

normalizing flow

Continuous Normalizing Flows

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

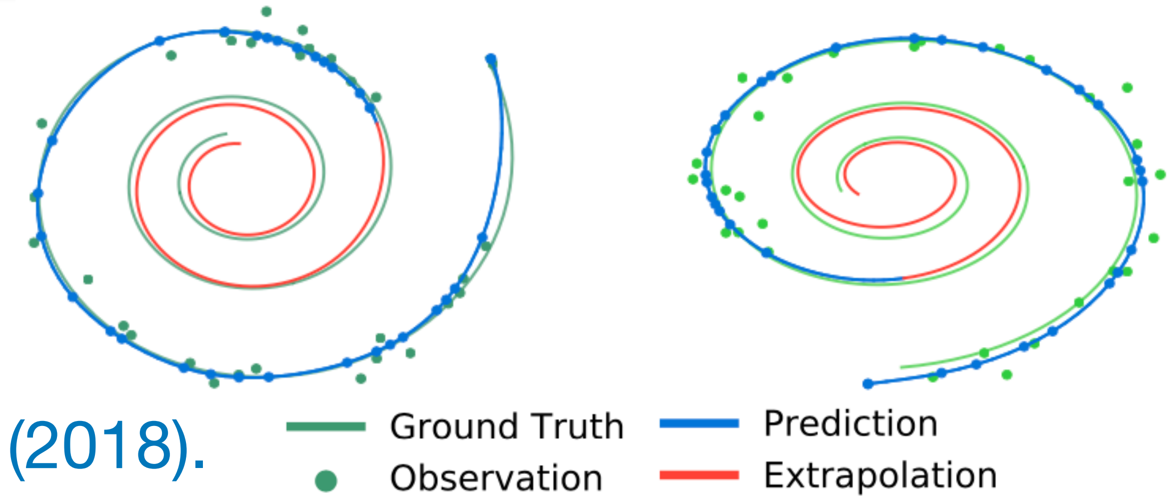
f does not need to be bijective!



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

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

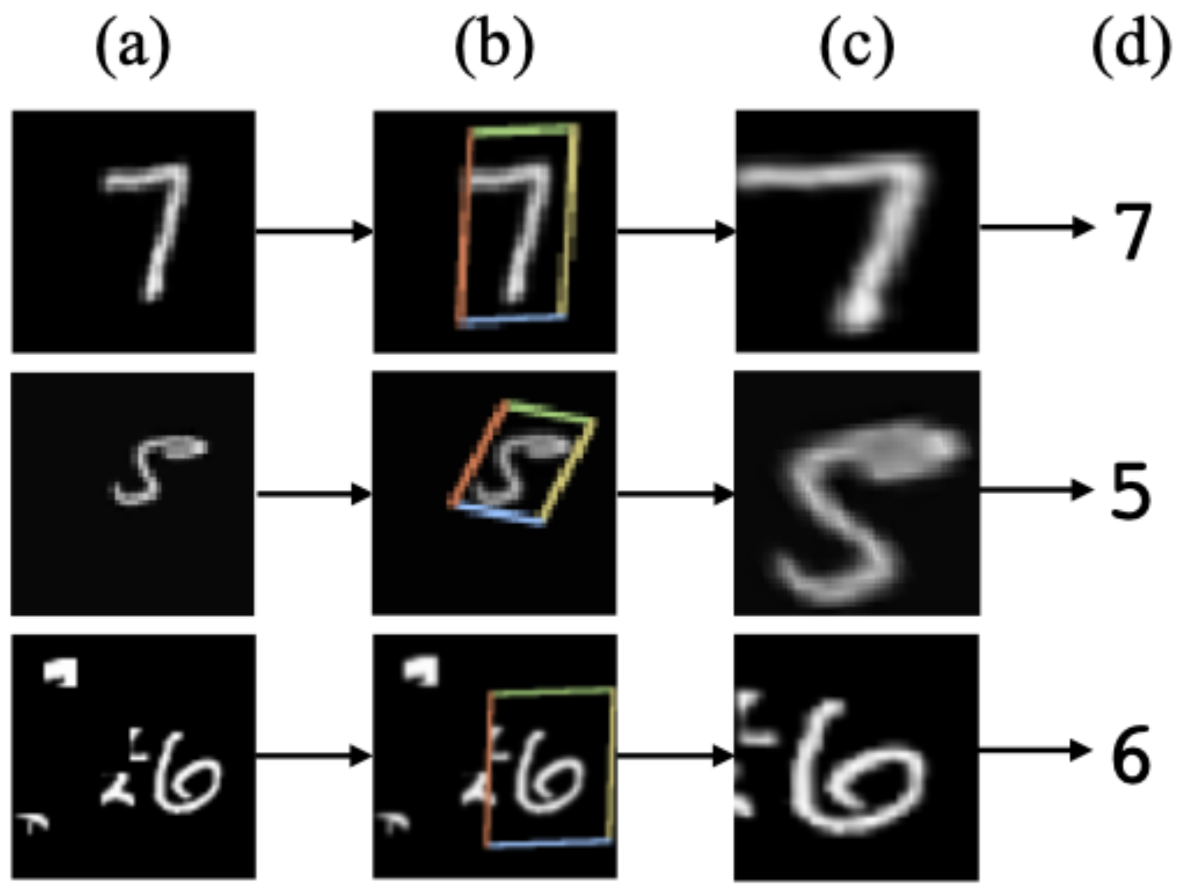
each $\mathbf{x}_{t_i} \sim p(\mathbf{x} | \mathbf{z}_{t_i}, \theta_\mathbf{x})$



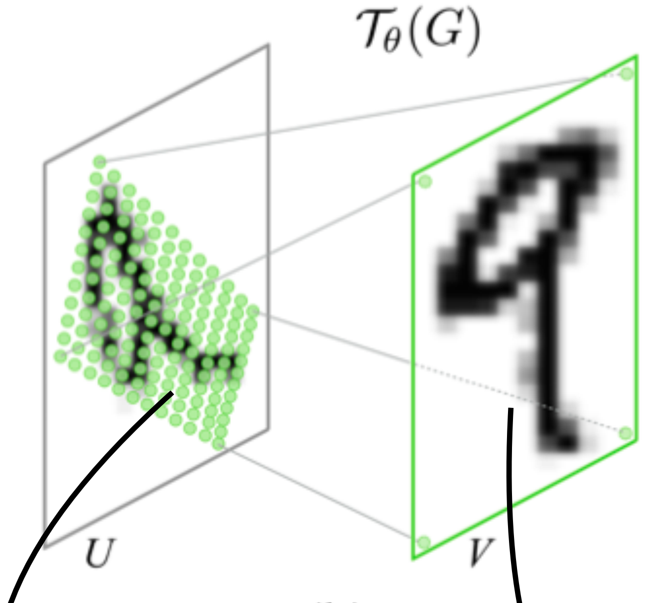
Chen, Ricky TQ, et al. "Neural ordinary differential equations." *arXiv preprint arXiv:1806.07366* (2018).

Spatial Transformer Networks

YouTube Video



(a) MNIST distorted with random translation, scale, rotation, and clutter.

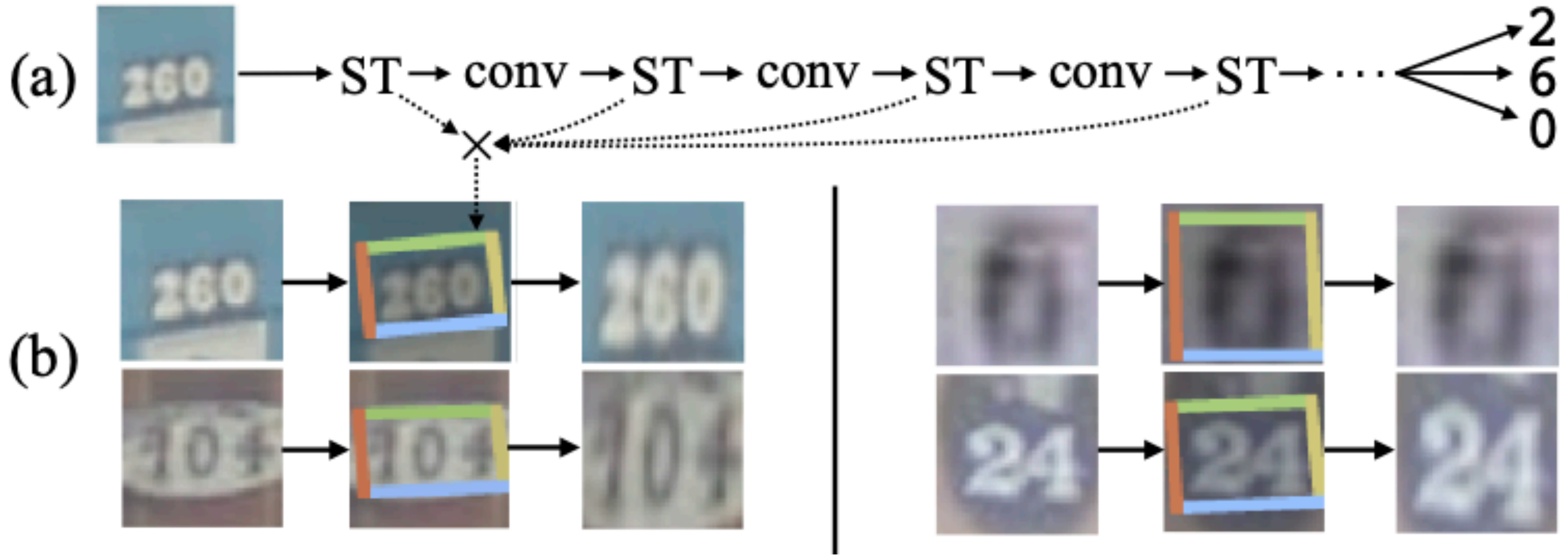


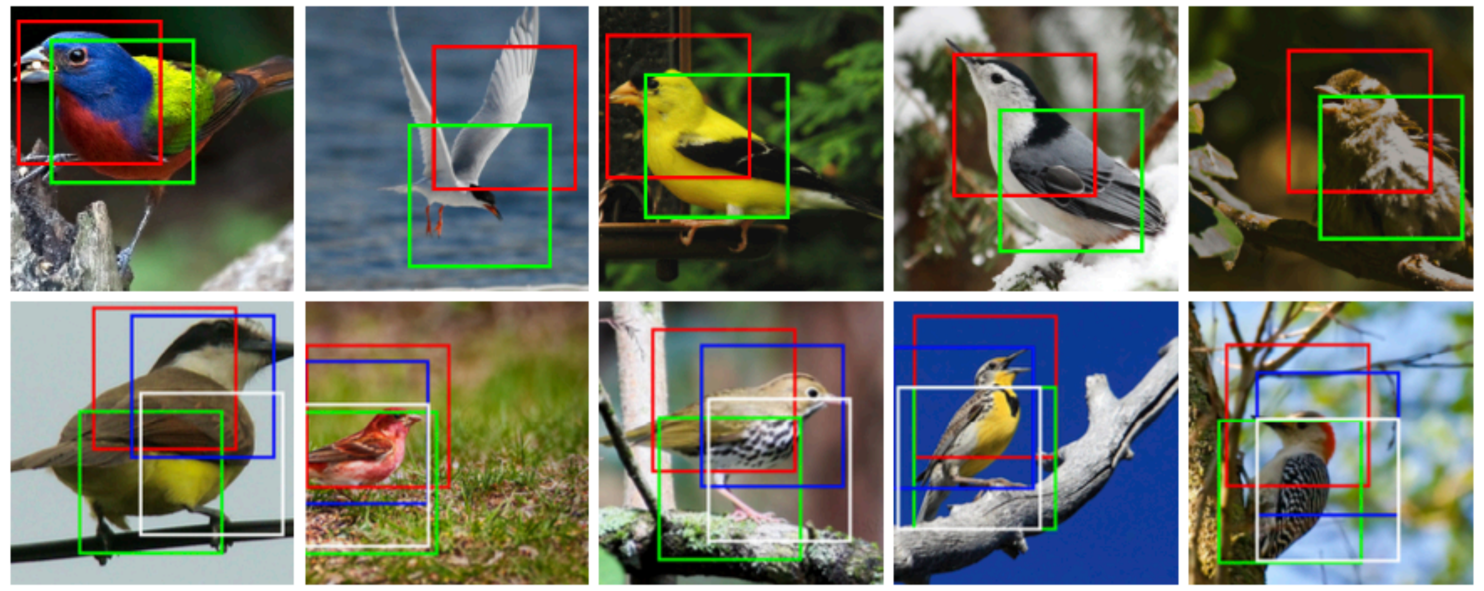
Source

Target

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = \mathtt{A}_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$

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





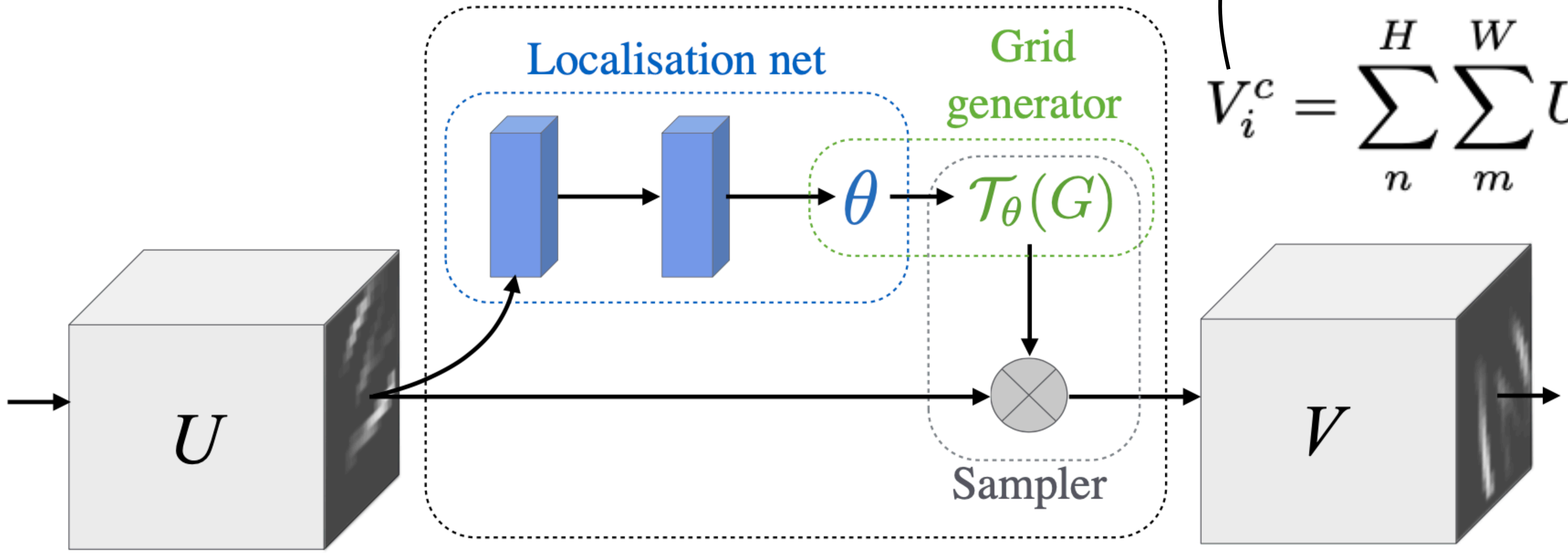
Model		
Cimpoi '15 [4]		66.7
Zhang '14 [30]		74.9
Branson '14 [2]		75.7
Lin '15 [20]		80.9
Simon '15 [24]		81.0
CNN (ours)	224px	82.3
2×ST-CNN	224px	83.1
2×ST-CNN	448px	83.9
4×ST-CNN	448px	**84.1**

Bilinear sampling

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

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

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



Spatial Transformer

Jaderberg, Max, Karen Simonyan, and Andrew Zisserman. "Spatial transformer networks." *Advances in neural information processing systems*. 2015.

Dynamic Routing Between Capsules

YouTube Video

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

$s_j \rightarrow$ total input to capsule j

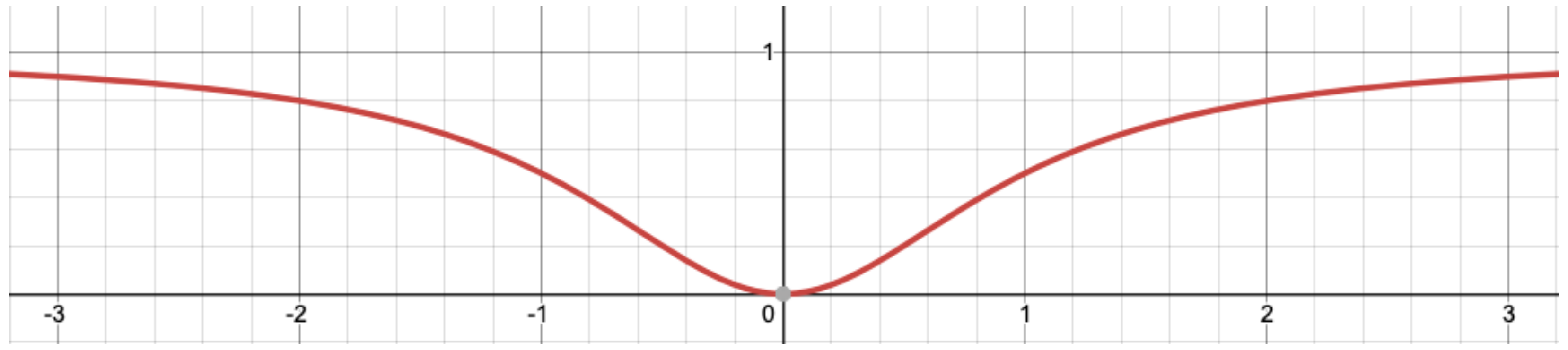
$v_j \rightarrow$ vector output of capsule j

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

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

entity: an object or an object part

$$v_j = \underbrace{\frac{\|s_j\|^2}{1+\|s_j\|^2}}_{\text{squashing function}} \frac{s_j}{\|s_j\|} \leftarrow \texttt{squash}$$



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

$c_{ij} \rightarrow$ coupling coefficients

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

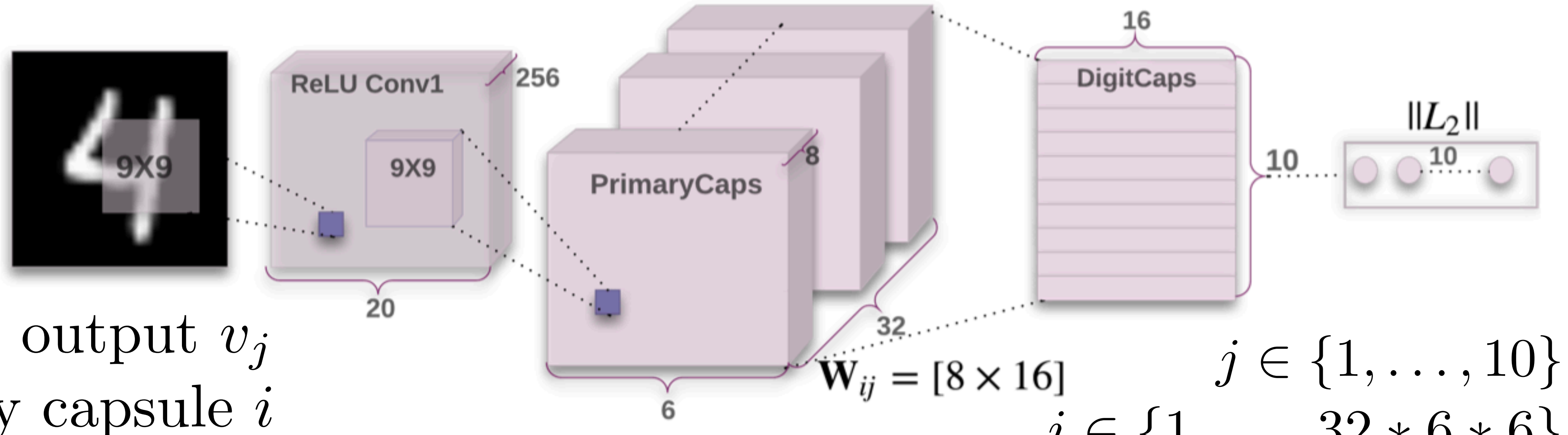
$u_{j|i} \rightarrow$ prediction vectors from the capsules in the layer below

$u_i \rightarrow$ output of a capsule in the layer below

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

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

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

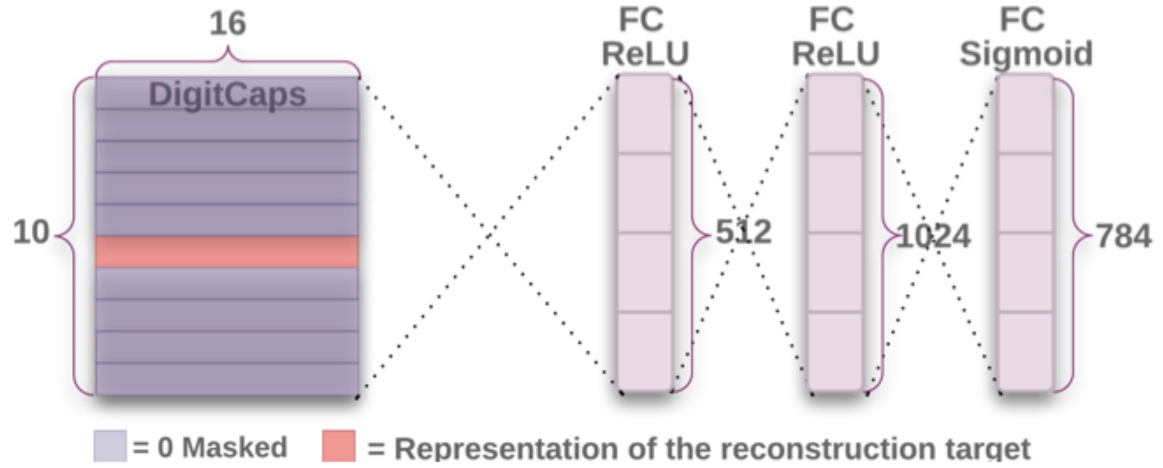


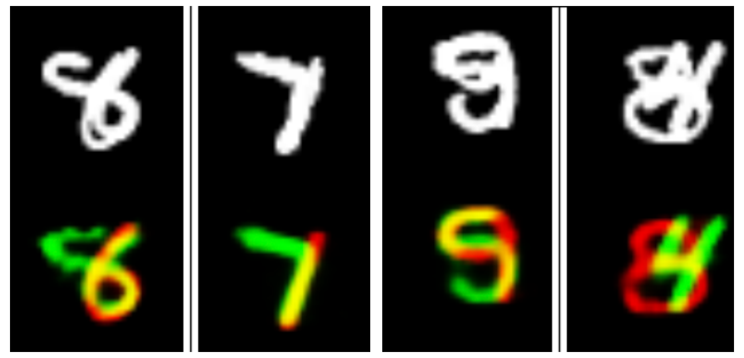
$j \in \{1, \dots, 10\}$

$i \in \{1, \dots, 32*6*6\}$

Procedure 1 Routing algorithm.

procedure ROUTING($\boldsymbol{u}_{j|i}, r, l$)
  for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
  for r iterations **do**
    for all capsule i in layer l: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$
    for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij}\mathbf{u}_{j|i}$
    for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$
    for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \mathbf{u}_{j|i}.\mathbf{v}_j$
  return $\mathbf{v}_j$





Margin loss for digit existence

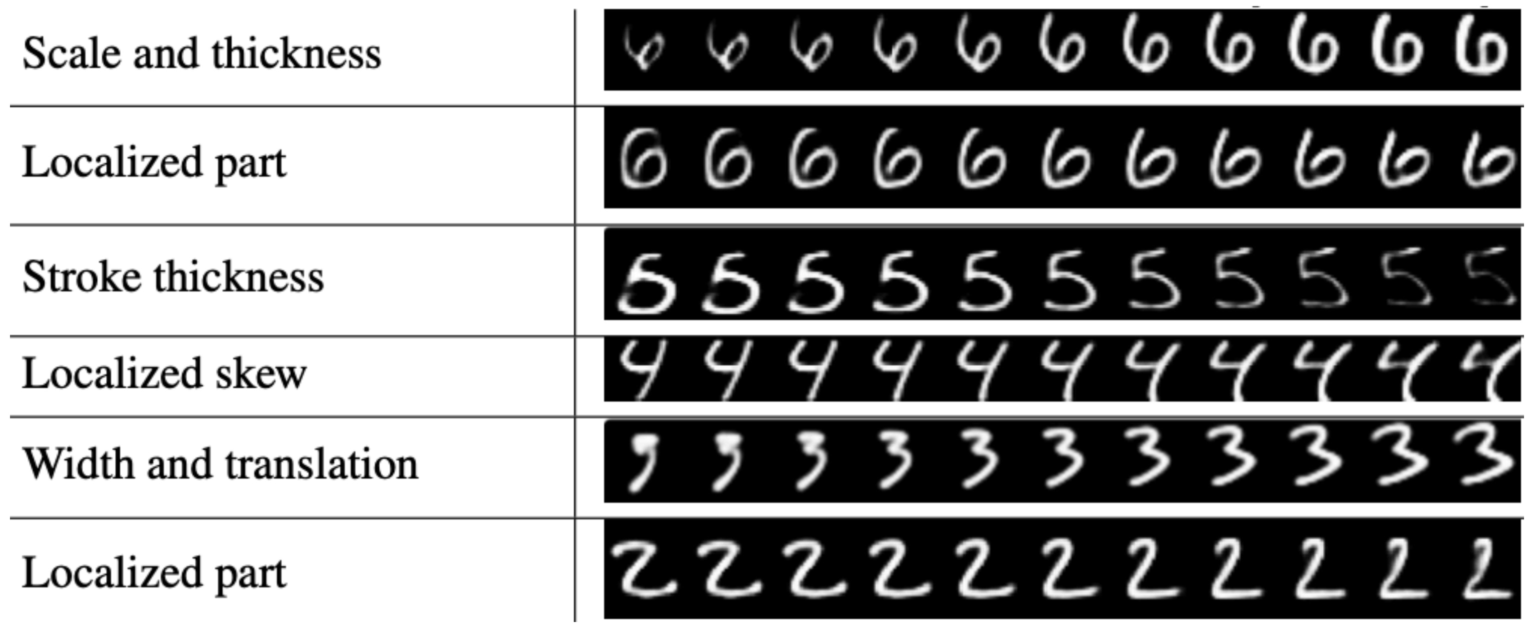
$$L_k = T_k \max(0, m^+ - \|\mathbf{v}_k\|)^2 + \lambda(1-T_k)\max(0, \|\mathbf{v}_k\| - m^-)^2$$

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

$\lambda = 0.5$

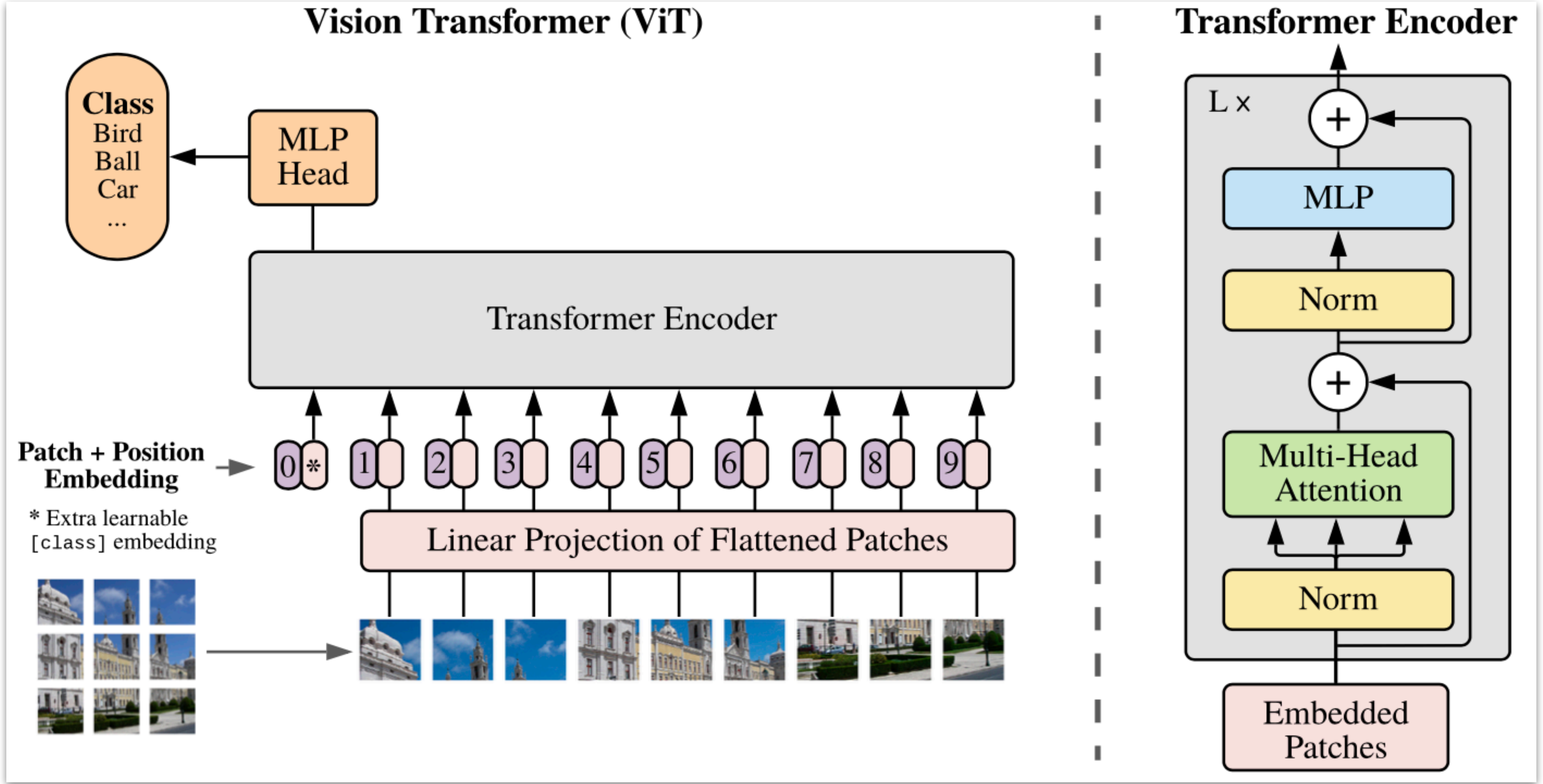
$m^+ = 0.9$ and $m^- = 0.1$

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	$\mathbf{0.25}_{\pm 0.005}$	**5.2**



Sabour, Sara, Nicholas Frosst, and Geoffrey E. Hinton. "Dynamic routing between capsules." *Advances in neural information processing systems*. 2017.

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



LN $\rightarrow$ layer normalization

MSA $\rightarrow$ multiheaded self-attention

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

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

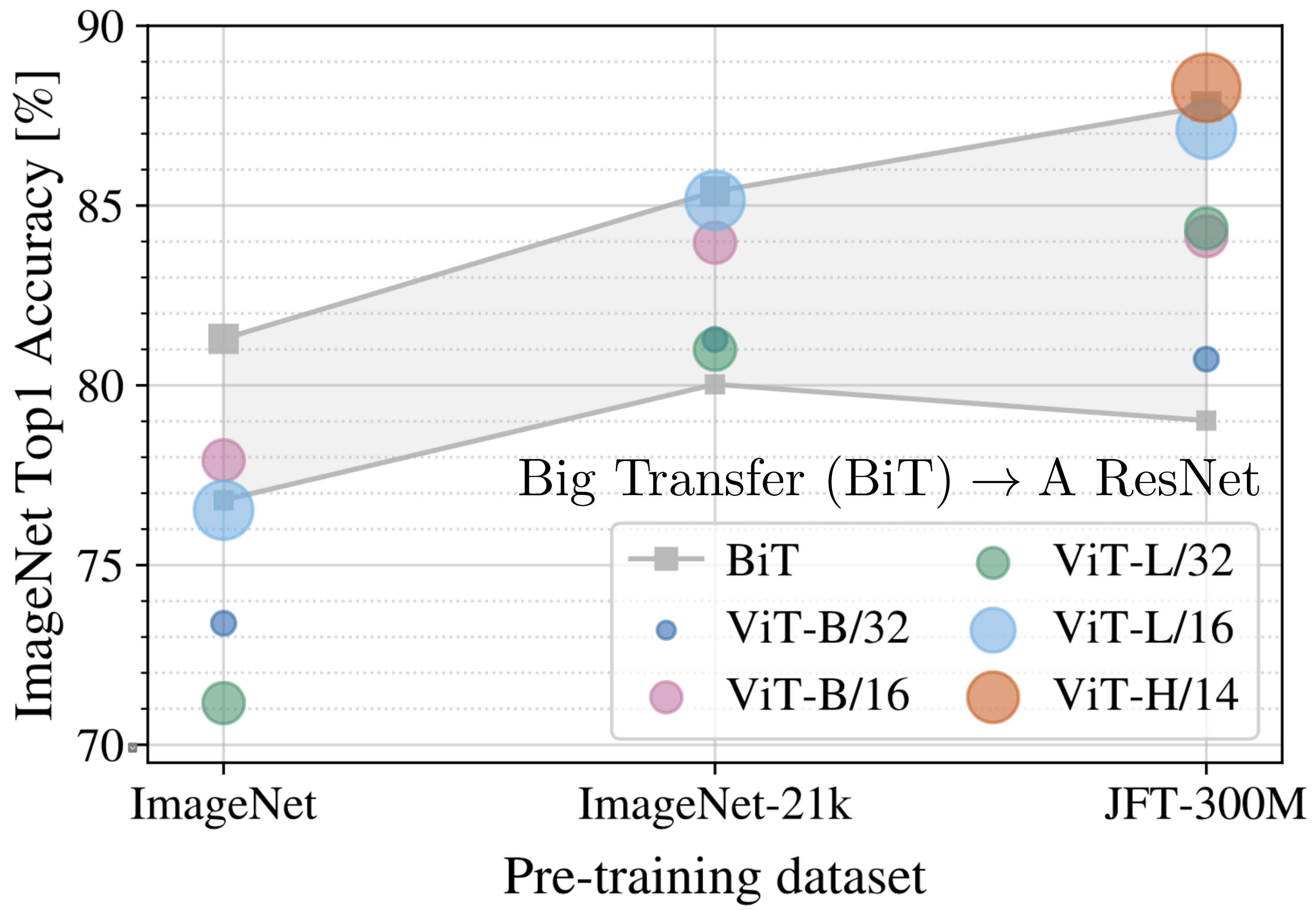
$A = \mathrm{softmax}(qk^T / \sqrt{D_h}), A \in \mathbb{R}^{N \times N}$

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

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

$D_h = D/k$

Inductive Bias in CNNs: translation equivariance and locality



Big Transfer (BiT) $\rightarrow$ A ResNet

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

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

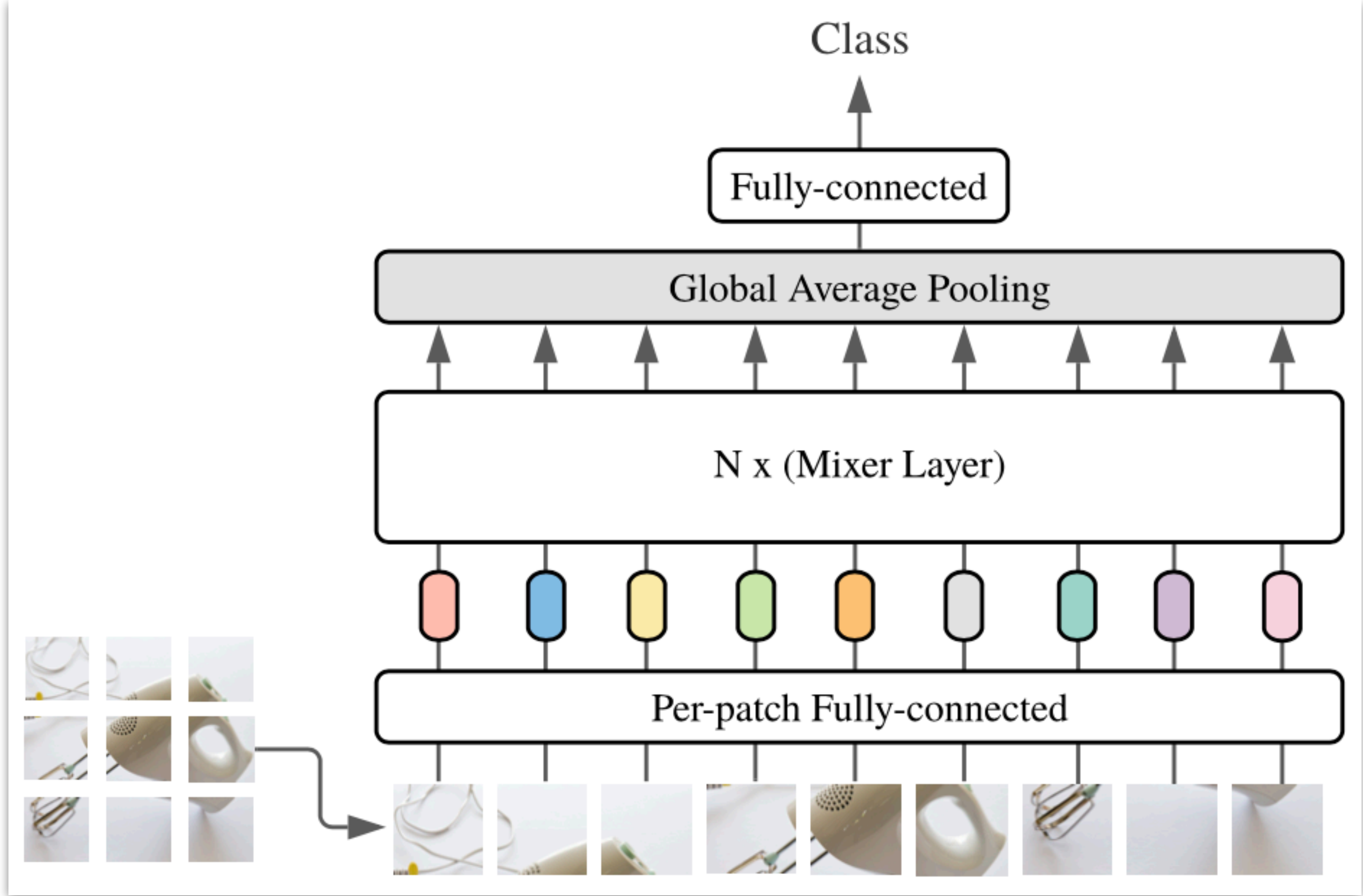
$D \rightarrow$ latent vector size

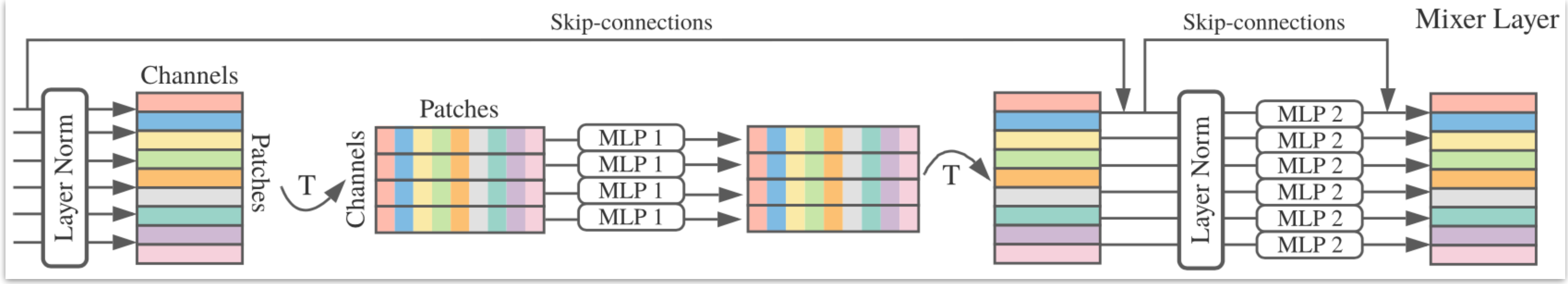
$z_0 = [x_{\text{class}}; x_p^1 E; x_p^2 E; \ldots; x_p^N E] + E_{\text{pos}}, E \in \mathbb{R}^{(P^2 C) \times D}, E_{\text{pos}} \in \mathbb{R}^{(N+1) \times D}$

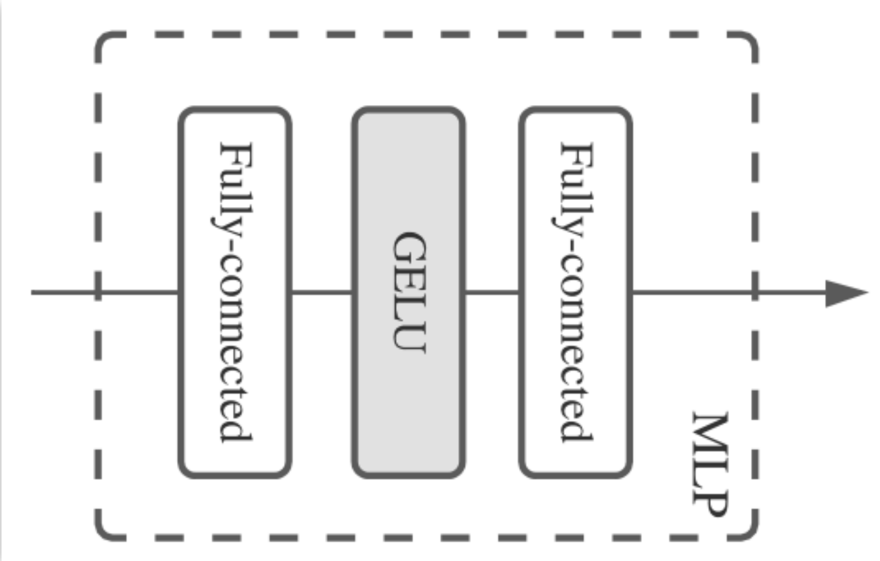
$z'_l = \mathrm{MSA}(\mathrm{LN}(z_{l-1}) + z_{l-1}), l = 1, \ldots, L \qquad z_l = \mathrm{MLP}(\mathrm{LN}(z'_l)) + z'_l, l = 1, \ldots, L \qquad y = \mathrm{LN}(z_L^0)$

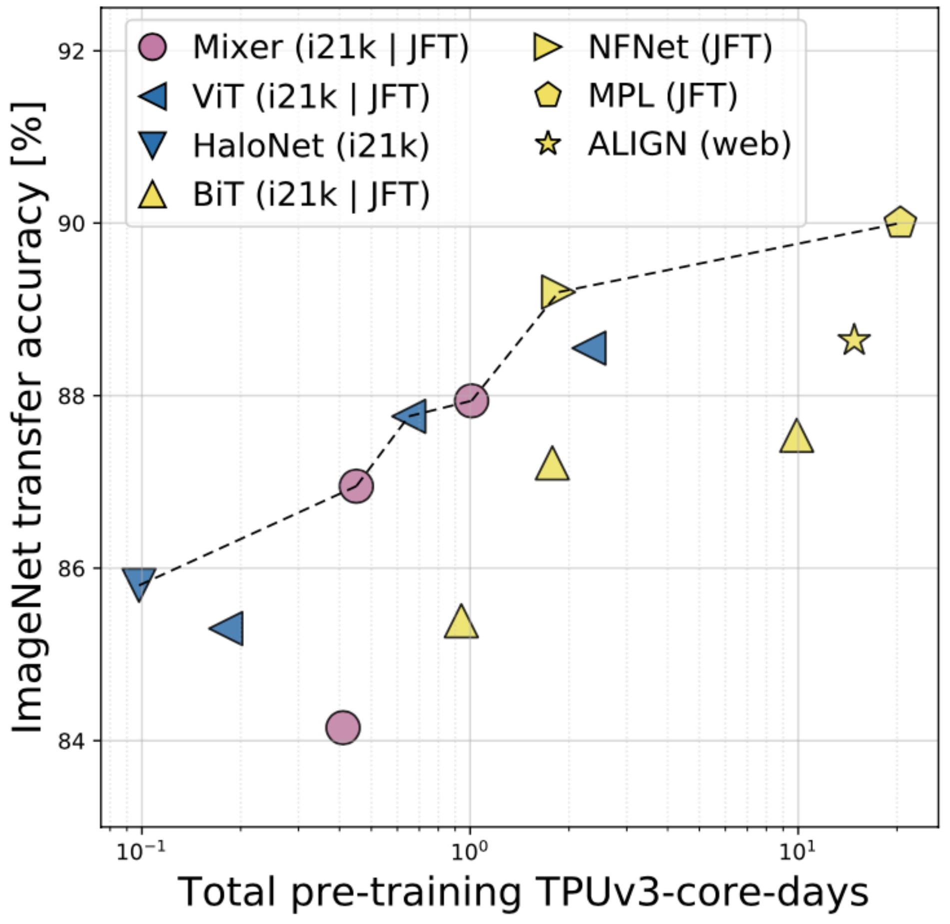
Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929* (2020).

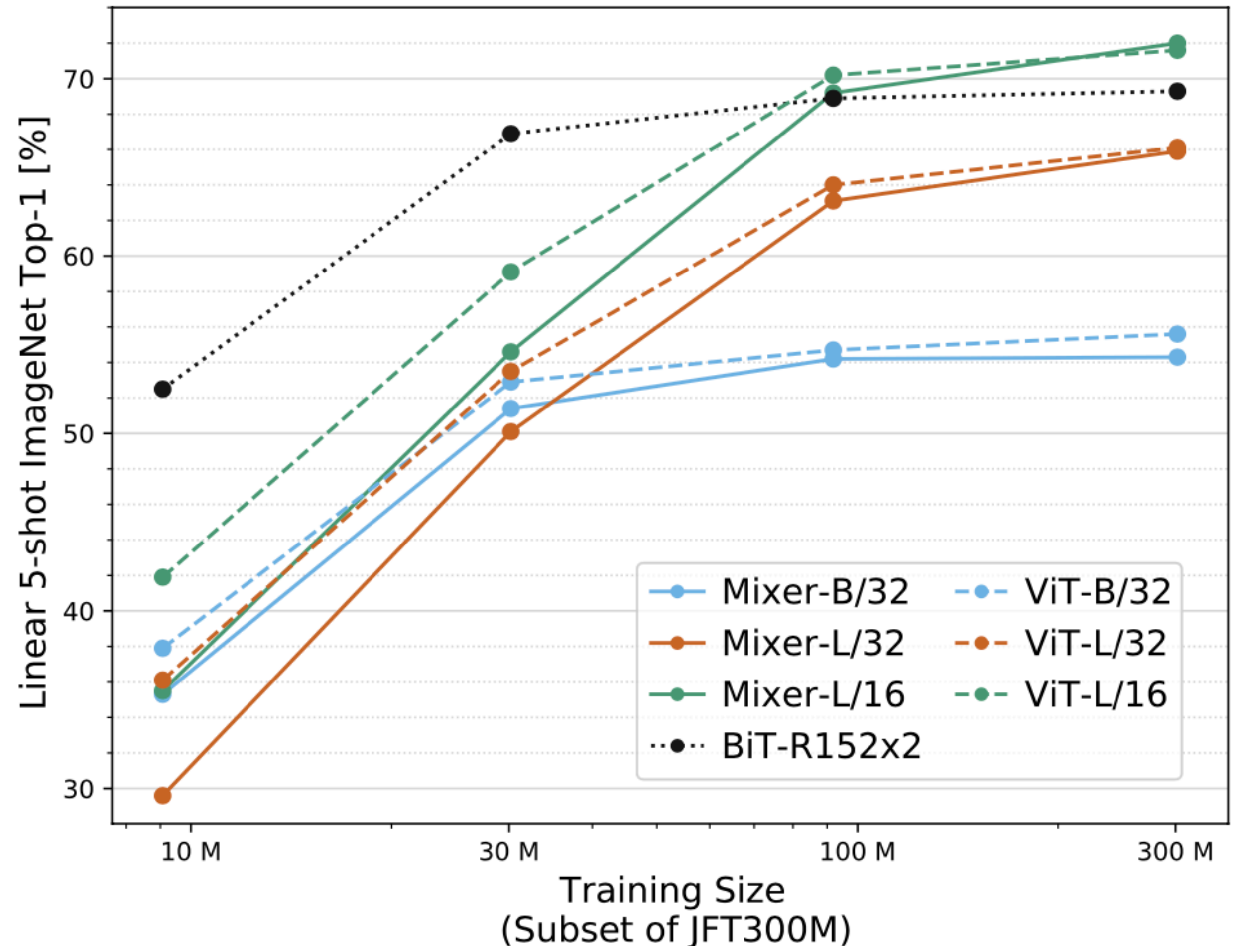
MLP-Mixer: An all-MLP Architecture for Vision











– token-mixing MLP – channel-mixing MLP

$X \in \mathbb{R}^{S \times C} \rightarrow$ real-valued input table

$S \rightarrow$ number of non-overlapping image patches

$S = \dfrac{HW}{P^2}$ $\quad H \times W \rightarrow$ resolution of the original image

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

all patches are projected with the same projection matrix

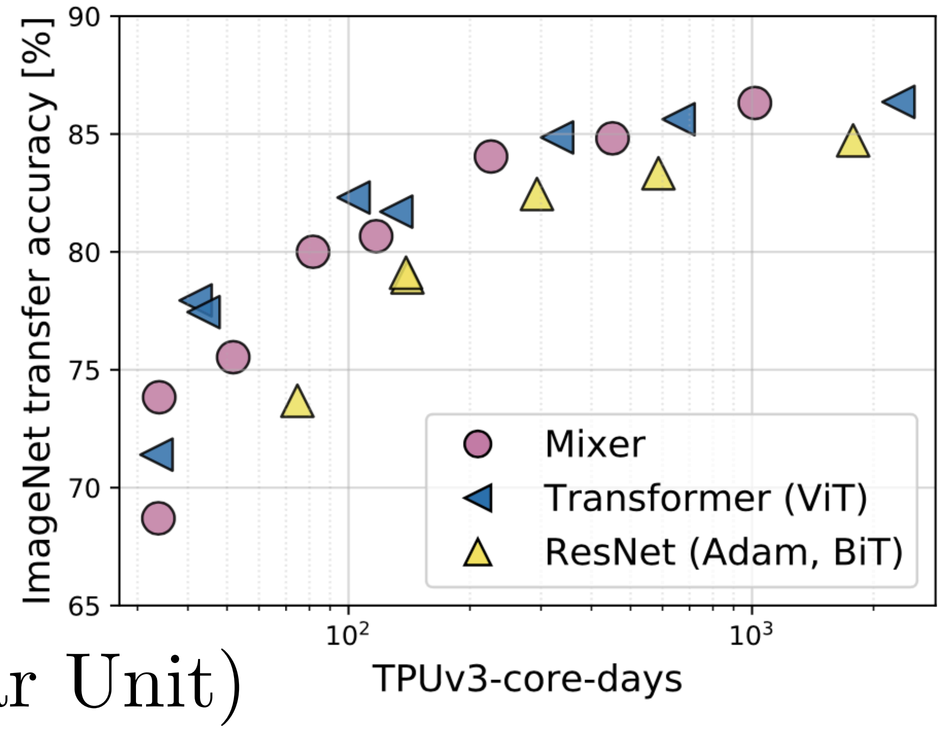
$C \rightarrow$ hidden dimension

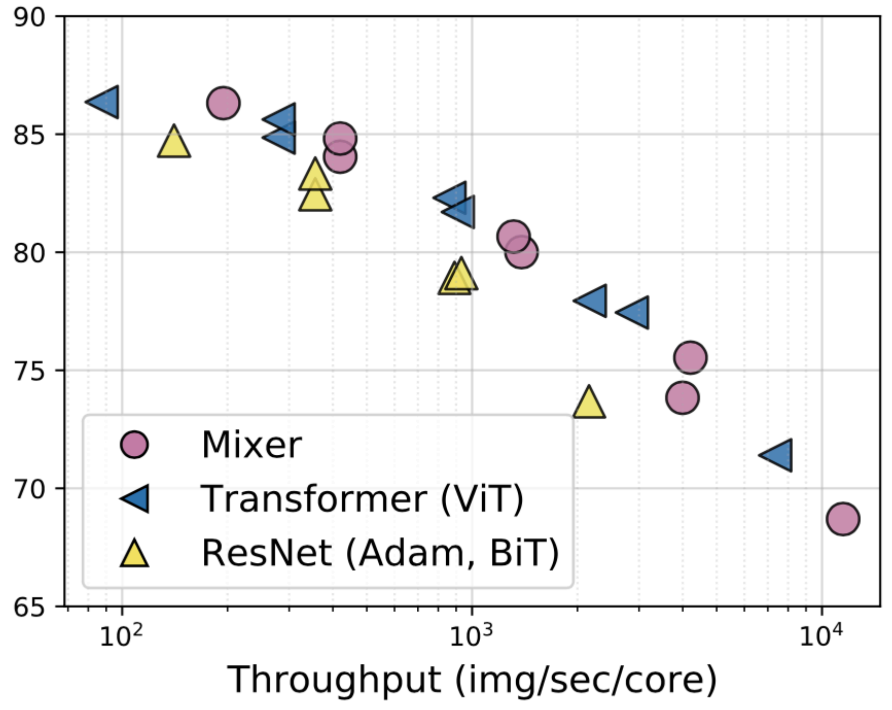
```
Linear computational complexity
in the number of input patches
and number of pixels
```

$$\mathbf{U}_{*,i} = \mathbf{X}_{*,i} + \mathbf{W}_2\, \sigma\big(\mathbf{W}_1\, \mathrm{LayerNorm}(\mathbf{X})_{*,i}\big), \quad \text{for } i = 1 \ldots C, \ \rightarrow \text{token mixing}$$

$$\mathbf{Y}_{j,*} = \mathbf{U}_{j,*} + \mathbf{W}_4\, \sigma\big(\mathbf{W}_3\, \mathrm{LayerNorm}(\mathbf{U})_{j,*}\big), \quad \text{for } j = 1 \ldots S. \ \rightarrow \text{channel mixing}$$

$W_1 \in \mathbb{R}^{D_S \times S}, W_2 \in \mathbb{R}^{S \times D_S}, W_3 \in \mathbb{R}^{D_C \times C}, W_4 \in \mathbb{R}^{C \times D_C}, \sigma \rightarrow$ GELU (Gaussian Error Linear Unit)





Tolstikhin, Ilya, et al. "MLP-Mixer: An all-MLP Architecture for Vision." *arXiv preprint arXiv:2105.01601* (2021).

High-Performance Large-Scale Image Recognition Without Normalization

Gradient Clipping

$L \rightarrow$ loss

$\theta \rightarrow$ model parameters

$G = \dfrac{\partial L}{\partial \theta} \rightarrow$ 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 ℓ-th layer

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

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

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

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

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

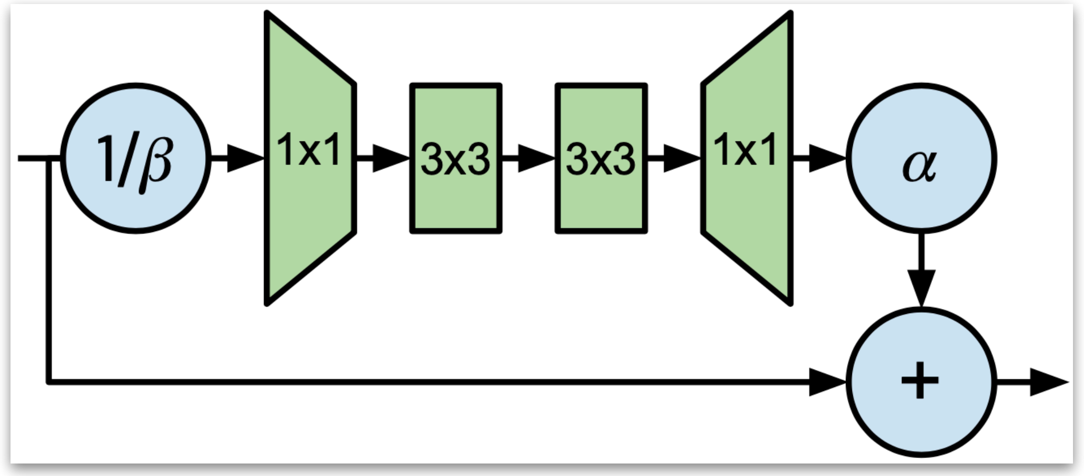
$$\frac{\|\Delta W^\ell\|_F}{\|W^\ell\|_F} = h \frac{\|G^\ell\|_F}{\|W^\ell\|_F}$$

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

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

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

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



Normalizer-Free ResNets

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

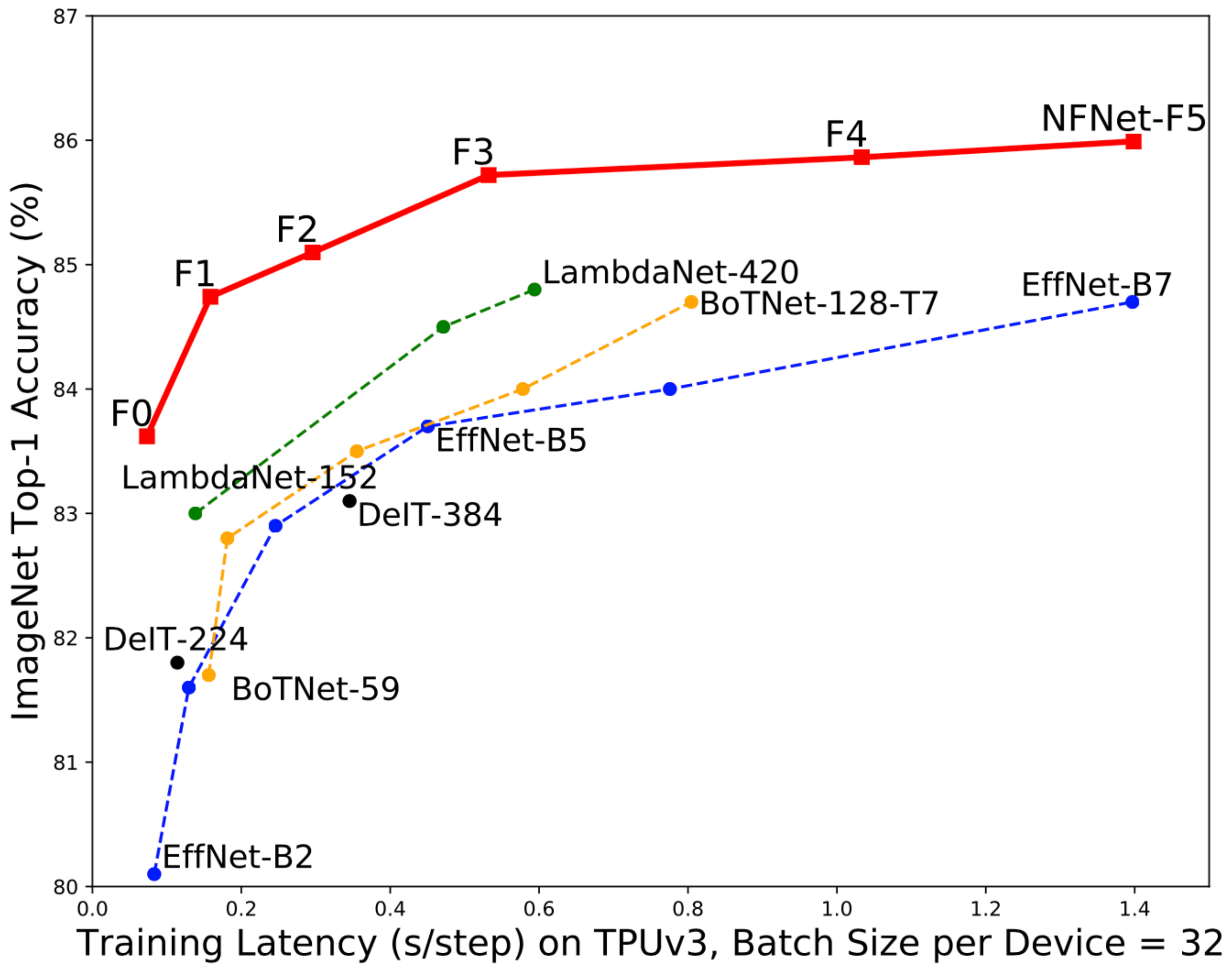
$h^\ell \rightarrow$ input to the ℓ-th residual block

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

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

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

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



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 γ.

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

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

Brock, Andrew, et al. "High-performance large-scale image recognition without normalization." *arXiv preprint arXiv:2102.06171* (2021).

Boulder

Questions?

[YouTube Playlist]