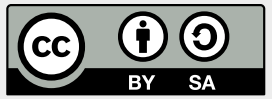
Convolutional Neural Networks II

Milan Straka

April 01, 2019

EUROPEAN UNION
European Structural and Investment Fund
Operational Programme Research,
Development and Education

Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics



Designing and Training Neural Networks

Designing and training a neural network is not a one-shot action, but instead an iterative procedure.

- When choosing hyperparameters, it is important to verify that the model does not underfit and does not overfit.
- Underfitting can be checked by increasing model capacity or training longer.
- Overfitting can be tested by observing train/dev difference and by trying stronger regularization.

Specifically, this implies that:

- We need to set number of training epochs so that training loss/performance no longer increases at the end of training.
- Generally, we want to use a large batchsize that does not slow us down too much (GPUs sometimes allow larger batches without slowing down training). However, with increasing batch size we need to increase learning rate, which is possible only to some extent. Also, small batch size sometimes work as regularization (especially for vanilla SGD algorithm).

Loading and Saving Models

- Using `tf.keras.Model.save`, both the architecture and model weights are saved. But saving the architecture is currently quite brittle:
  - `tf.keras.layers.InputLayer` does not work correctly
  - object losses (inherited from `tf.losses.Loss`) cannot be loaded
  - TensorFlow specific functions (not in `tf.keras.layers`) works only sometimes
  - ...

  Of course, the bugs are being fixed.
- Using `tf.keras.Model.save_weights`, only the weights of the model are saved. If the model is constructed again by the script (which usually required specifying the same hyperparameters as during model training), weights can be loaded using `tf.keras.Model.load_weights`.

Main Takeaways From Previous Lecture

- Convolutions can provide
  - local interactions in spacial/temporal dimensions
  - shift invariance
  - *much* less parameters than a fully connected layer
- Usually repeated 3×3 convolutions are enough, no need for larger filter sizes.
- When pooling is performed, double number of channels.
- Final fully connected layers are not needed, global average pooling is usually enough.
- Batch normalization is a great regularization method for CNNs.

ResNet – 2015 (3.6% error)

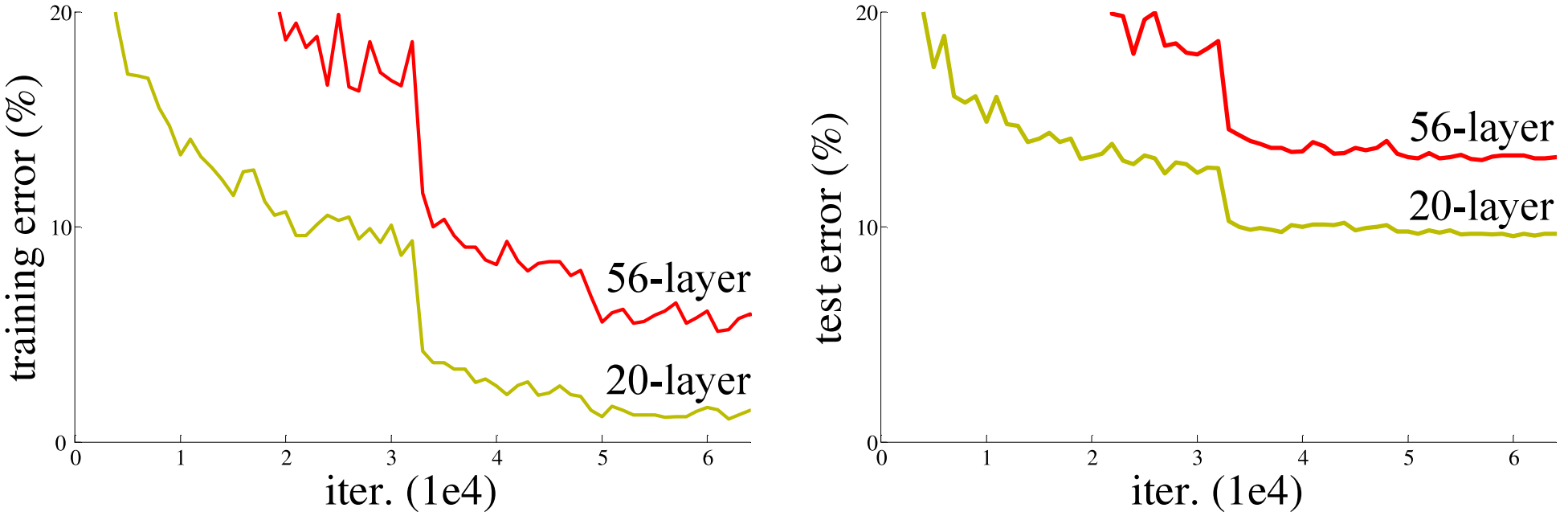


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Figure 1 of paper "Deep Residual Learning for Image Recognition", https://arxiv.org/abs/1512.03385.

ResNet – 2015 (3.6% error)

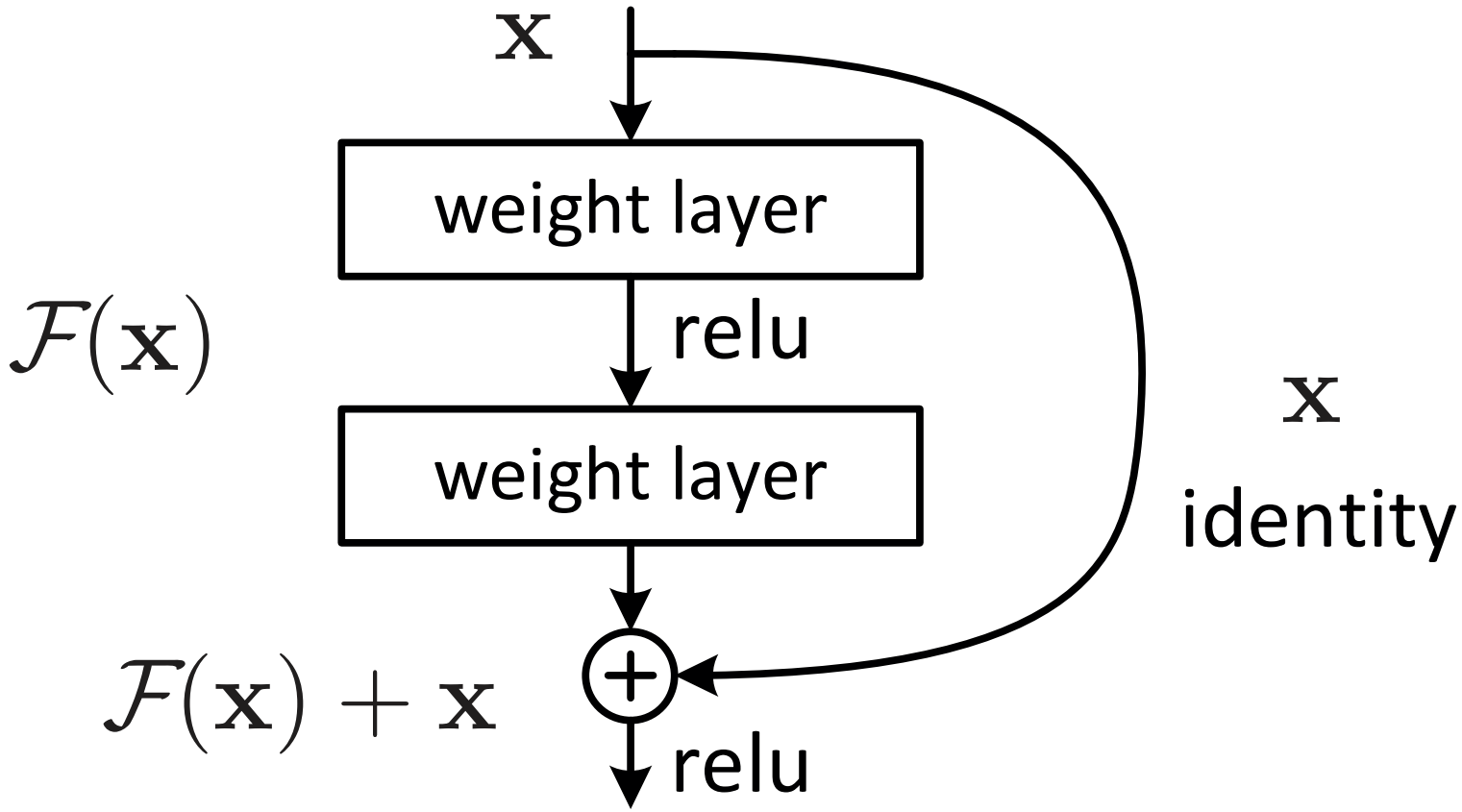


Figure 2. Residual learning: a building block.

Figure 2 of paper "Deep Residual Learning for Image Recognition", https://arxiv.org/abs/1512.03385.

ResNet – 2015 (3.6% error)

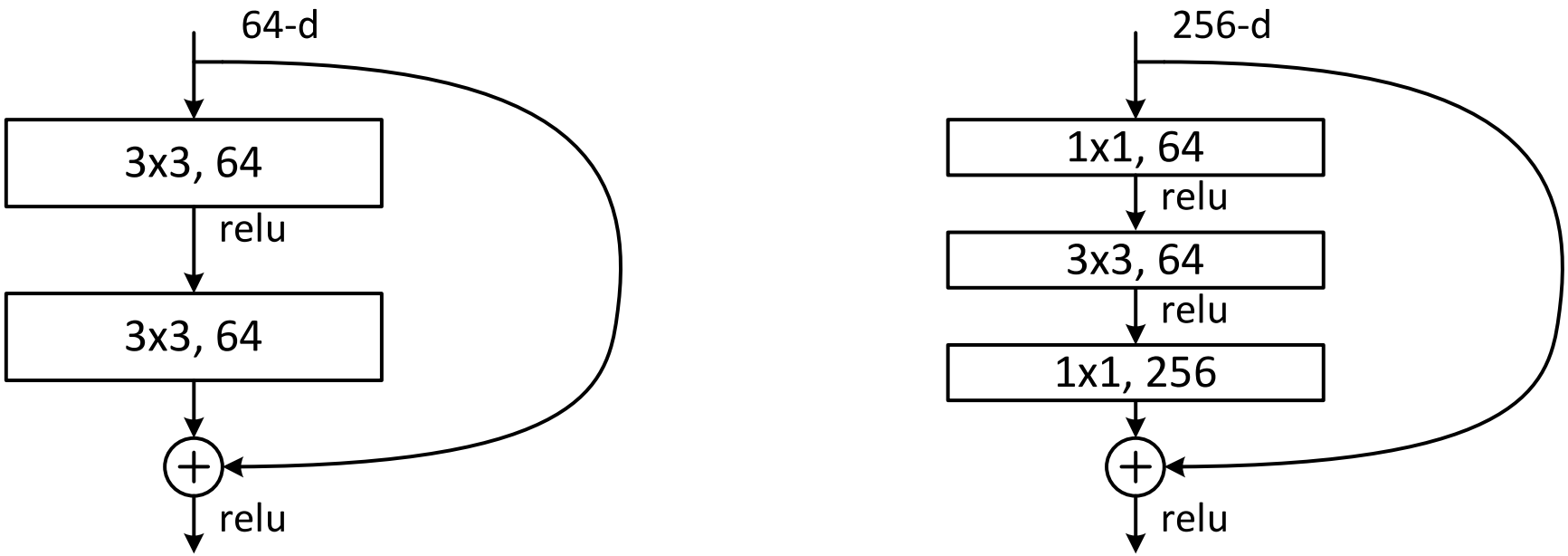


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

Figure 5 of paper "Deep Residual Learning for Image Recognition", https://arxiv.org/abs/1512.03385.

ResNet – 2015 (3.6% error)

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

Table 1 of paper "Deep Residual Learning for Image Recognition", https://arxiv.org/abs/1512.03385.

ResNet – 2015 (3.6% error)

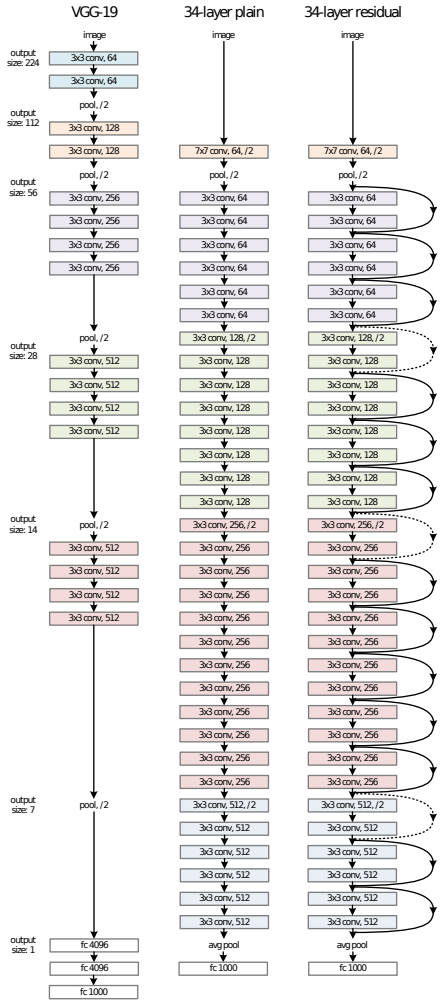


Figure 3 of paper "Deep Residual Learning for Image Recognition", https://arxiv.org/abs/1512.03385.

The residual connections cannot be applied directly when number of channels increase.

The authors considered several alternatives, and chose the one where in case of channels increase a 1×1 convolution is used on the projections to match the required number of channels.

ResNet – 2015 (3.6% error)

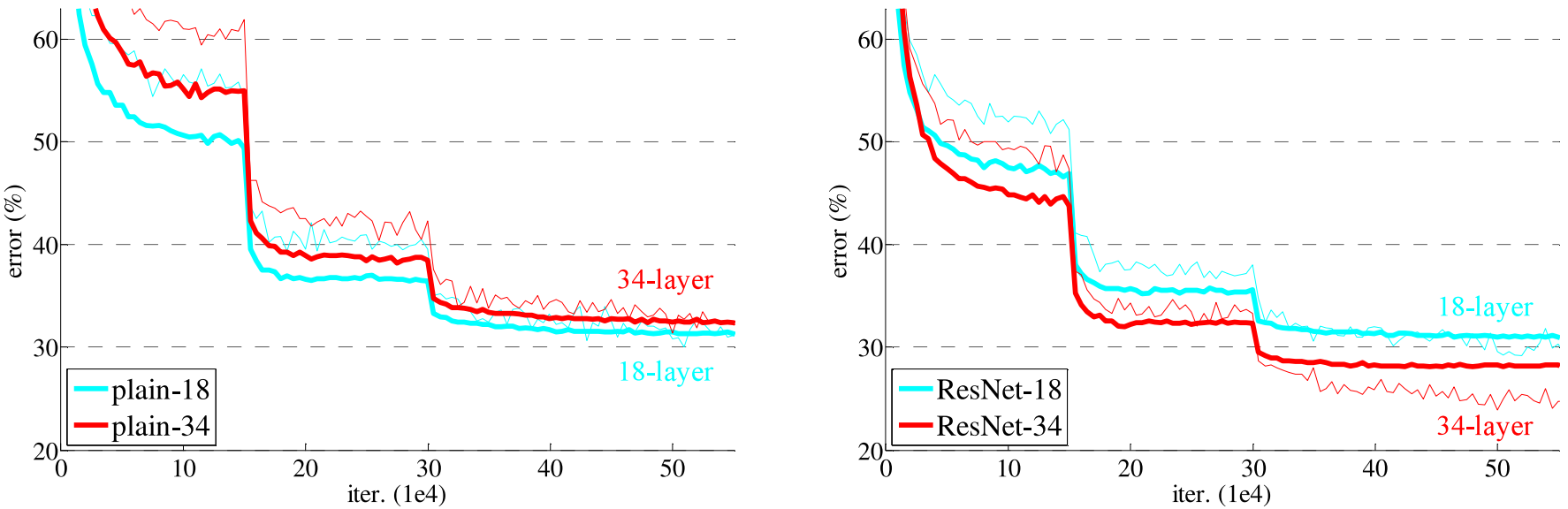


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 laye s. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

Figure 4 of paper "Deep Residual Learning for Image Recognition", https://arxiv.org/abs/1512.03385.

ResNet – 2015 (3.6% error)

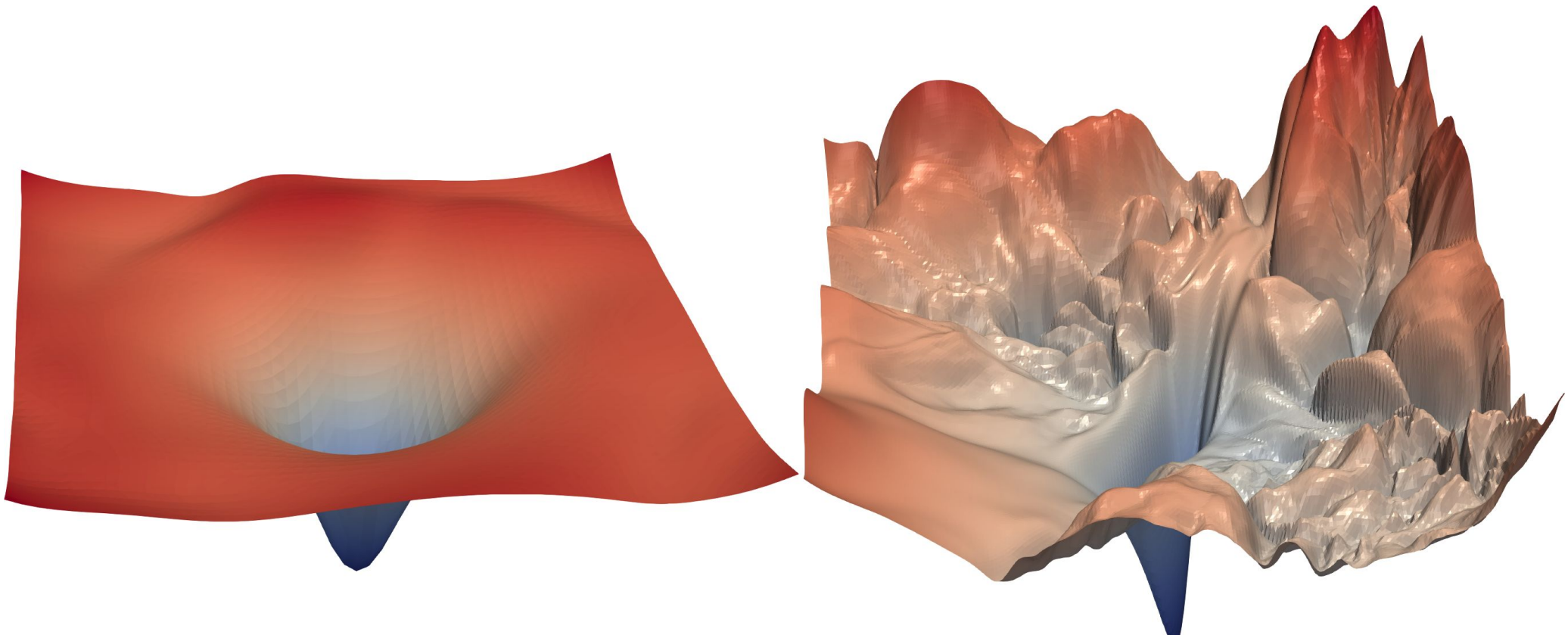


Figure 1 of paper "Visualizing the Loss Landscape of Neural Nets", https://arxiv.org/abs/1712.09913.

ResNet – 2015 (3.6% error)

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

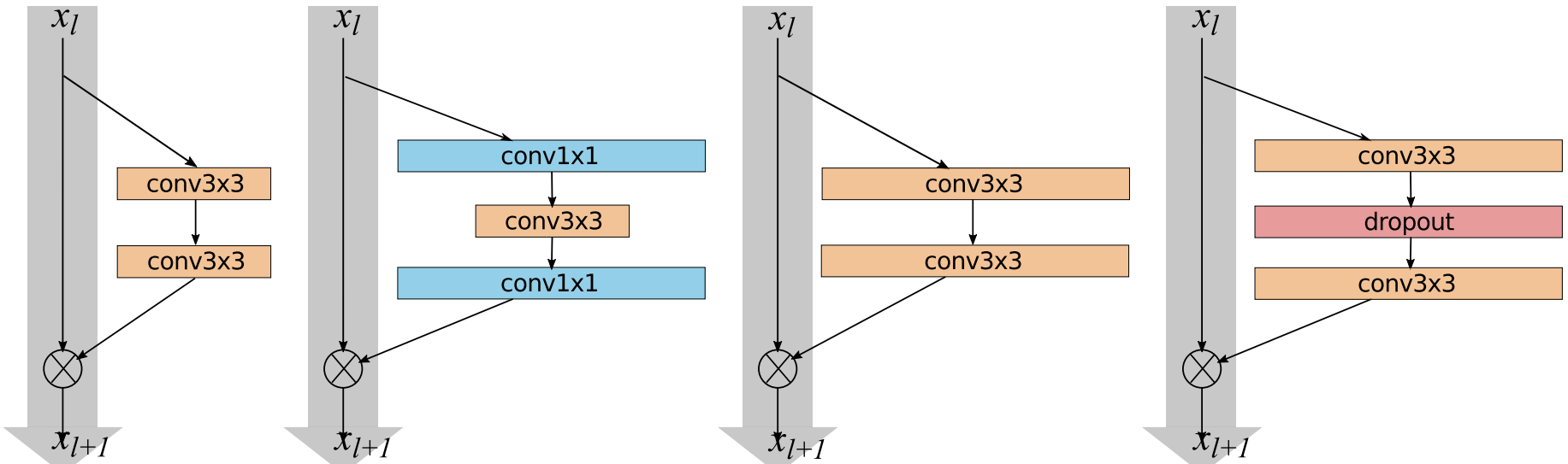
Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except † reported on the test set).

Table 4 of paper "Deep Residual Learning for Image Recognition", https://arxiv.org/abs/1512.03385.

method	top-5 err. (**test**)
VGG [41] (ILSVRC'14)	7.32
GoogLeNet [44] (ILSVRC'14)	6.66
VGG [41] (v5)	6.8
PReLU-net [13]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

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

Table 5 of paper "Deep Residual Learning for Image Recognition", https://arxiv.org/abs/1512.03385.



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

Figure 1: Various residual blocks used in the paper. Batch normalization and ReLU precede each convolution (omitted for clarity)

Figure 1 of paper "Wide Residual Networks", https://arxiv.org/abs/1605.07146

- Authors do not consider bottleneck blocks. Instead, they experiment with different *block types*, e.g., $B(1,3,1)$ of $B(3,3)$

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

Table 2 of paper "Wide Residual Networks", https://arxiv.org/abs/1605.07146

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	$[8 \times 8]$

Table 1 of paper "Wide Residual Networks", https://arxiv.org/abs/1605.07146

- Authors evaluate various *widening factors* k

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

Table 4 of paper "Wide Residual Networks", https://arxiv.org/abs/1605.07146

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	$[8 \times 8]$

Table 1 of paper "Wide Residual Networks", https://arxiv.org/abs/1605.07146

- Authors measure the effect of *dropping out* inside the residual block (but not the residual connection itself)

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

Table 6 of paper "Wide Residual Networks", https://arxiv.org/abs/1605.07146

group name	output size	block type = $B(3,3)$
conv1	32×32	$[3\times3,\ 16]$
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	$[8 \times 8]$

Table 1 of paper "Wide Residual Networks", https://arxiv.org/abs/1605.07146

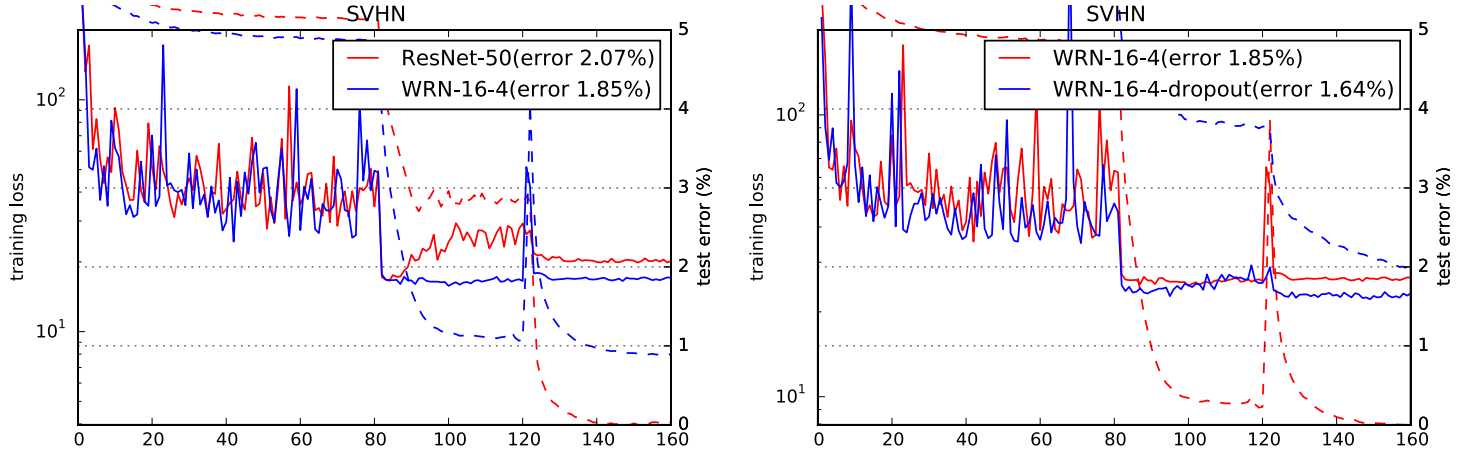


Figure 3 of paper "Wide Residual Networks", https://arxiv.org/abs/1605.07146

WideNet – CIFAR Results

	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 (ours)	40-4	8.9M	4.53	21.18
	16-8	11.0M	4.27	20.43
	28-10	36.5M	**4.00**	**19.25**

Table 5 of paper "Wide Residual Networks", https://arxiv.org/abs/1605.07146

DenseNet

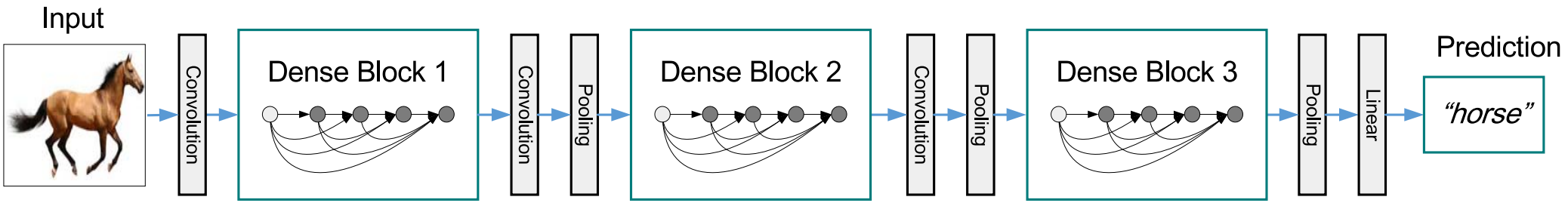


Figure 2 of paper "Densely Connected Convolutional Networks", https://arxiv.org/abs/1608.06993

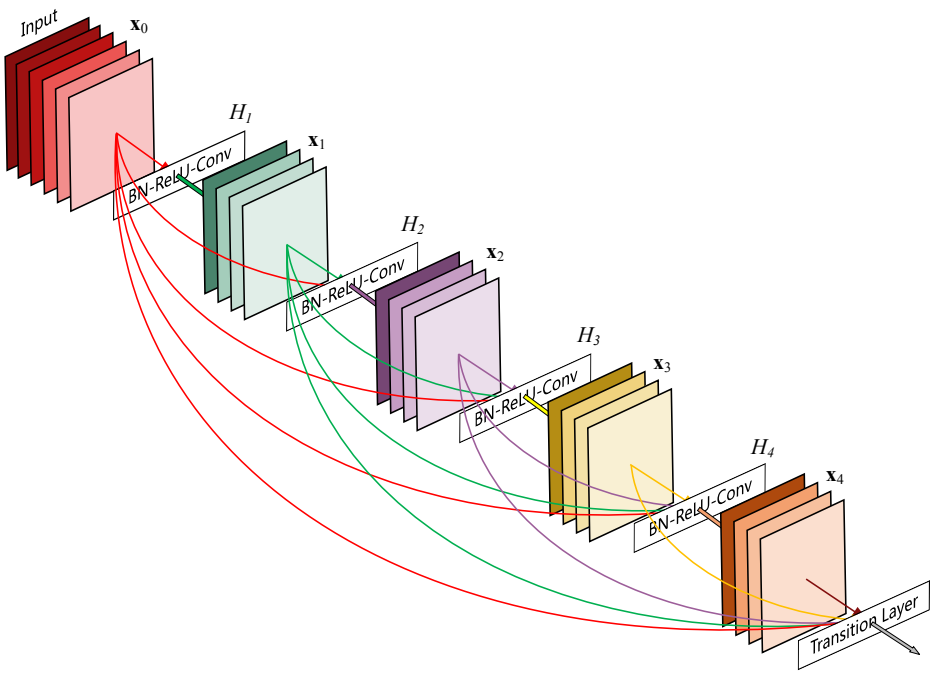


Figure 1 of paper "Densely Connected Convolutional Networks", https://arxiv.org/abs/1608.06993

DenseNet – Architecture

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112×112	7×7 conv, stride 2			
Pooling	56×56	3×3 max pool, stride 2			
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	1×1 conv			
	28×28	2×2 average pool, stride 2			
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28×28	1×1 conv			
	14×14	2×2 average pool, stride 2			
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14×14	1×1 conv			
	7×7	2×2 average pool, stride 2			
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1×1	7×7 global average pool			
		1000D fully-connected, softmax			

Table 1 of paper "Densely Connected Convolutional Networks", https://arxiv.org/abs/1608.06993

DenseNet – Results

Method	Depth	Params	C10	C10+	C100	C100+	SVHN
Network in Network [22]	-	-	10.41	8.81	35.68	-	2.35
All-CNN [32]	-	-	9.08	7.25	-	33.71	-
Deeply Supervised Net [20]	-	-	9.69	7.97	-	34.57	1.92
Highway Network [34]	-	-	-	7.72	-	32.39	-
FractalNet [17]	21	38.6M	10.18	5.22	35.34	23.30	2.01
with Dropout/Drop-path	21	38.6M	7.33	4.60	28.20	23.73	1.87
ResNet [11]	110	1.7M	-	6.61	-	-	-
ResNet (reported by [13])	110	1.7M	13.63	6.41	44.74	27.22	2.01
ResNet with Stochastic Depth [13]	110	1.7M	11.66	5.23	37.80	24.58	1.75
	1202	10.2M	-	4.91	-	-	-
Wide ResNet [42]	16	11.0M	-	4.81	-	22.07	-
	28	36.5M	-	4.17	-	20.50	-
with Dropout	16	2.7M	-	-	-	-	1.64
ResNet (pre-activation) [12]	164	1.7M	11.26*	5.46	35.58*	24.33	-
	1001	10.2M	10.56*	4.62	33.47*	22.71	-
DenseNet ($k = 12$)	40	1.0M	**7.00**	5.24	**27.55**	24.42	1.79
DenseNet ($k = 12$)	100	7.0M	**5.77**	**4.10**	**23.79**	**20.20**	1.67
DenseNet ($k = 24$)	100	27.2M	**5.83**	**3.74**	**23.42**	**19.25**	**1.59**
DenseNet-BC ($k = 12$)	100	0.8M	**5.92**	4.51	**24.15**	22.27	1.76
DenseNet-BC ($k = 24$)	250	15.3M	**5.19**	**3.62**	**19.64**	**17.60**	1.74
DenseNet-BC ($k = 40$)	190	25.6M	-	**3.46**	-	**17.18**	-

Table 2 of paper "Densely Connected Convolutional Networks", https://arxiv.org/abs/1608.06993

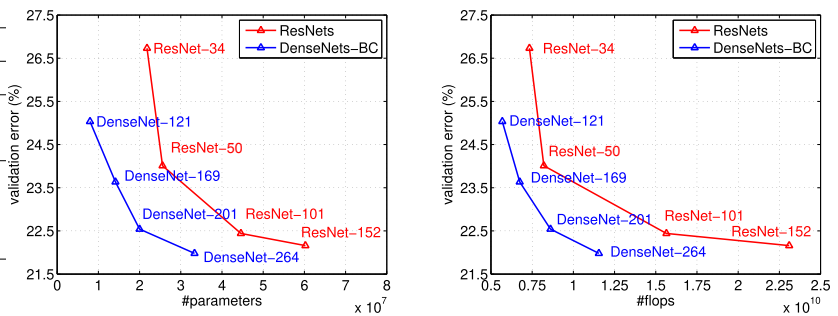
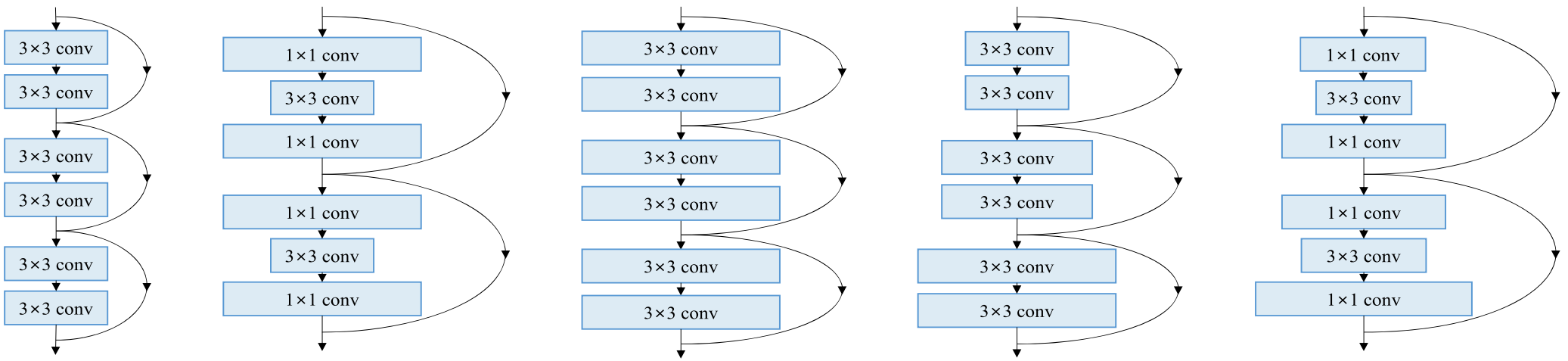


Figure 3 of paper "Densely Connected Convolutional Networks", https://arxiv.org/abs/1608.06993

PyramidNet



(a) basic (b) bottleneck (c) wide (d) pyramidal (e) pyramidal bottleneck

Figure 1 of paper "Deep Pyramidal Residual Networks", https://arxiv.org/abs/1610.02915

PyramidNet – Growth Rate

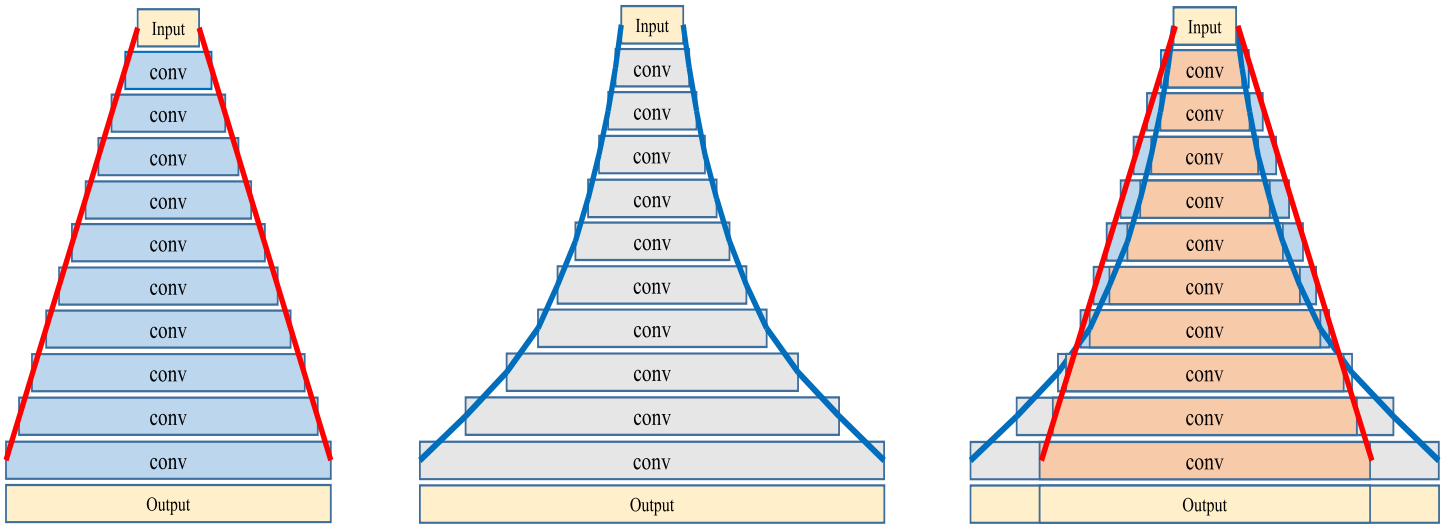


Figure 2 of paper "Deep Pyramidal Residual Networks", https://arxiv.org/abs/1610.02915

In architectures up until now, number of filters doubled when spacial resolution was halved.

Such exponential growth would suggest gradual widening rule $D_k = \lfloor D_{k-1} \cdot \alpha^{1/N} \rfloor$.

However, the authors employ a linear widening rule $D_k = \lfloor D_{k-1} + \alpha/N \rfloor$, where D_k is number of filters in the k-th out of N convolutional block and α is number of filters to add in total.

PyramidNet – Residual Connections

No residual connection can be a real identity – the authors propose to zero-pad missing channels, where the zero-pad channels correspond to newly computed features.

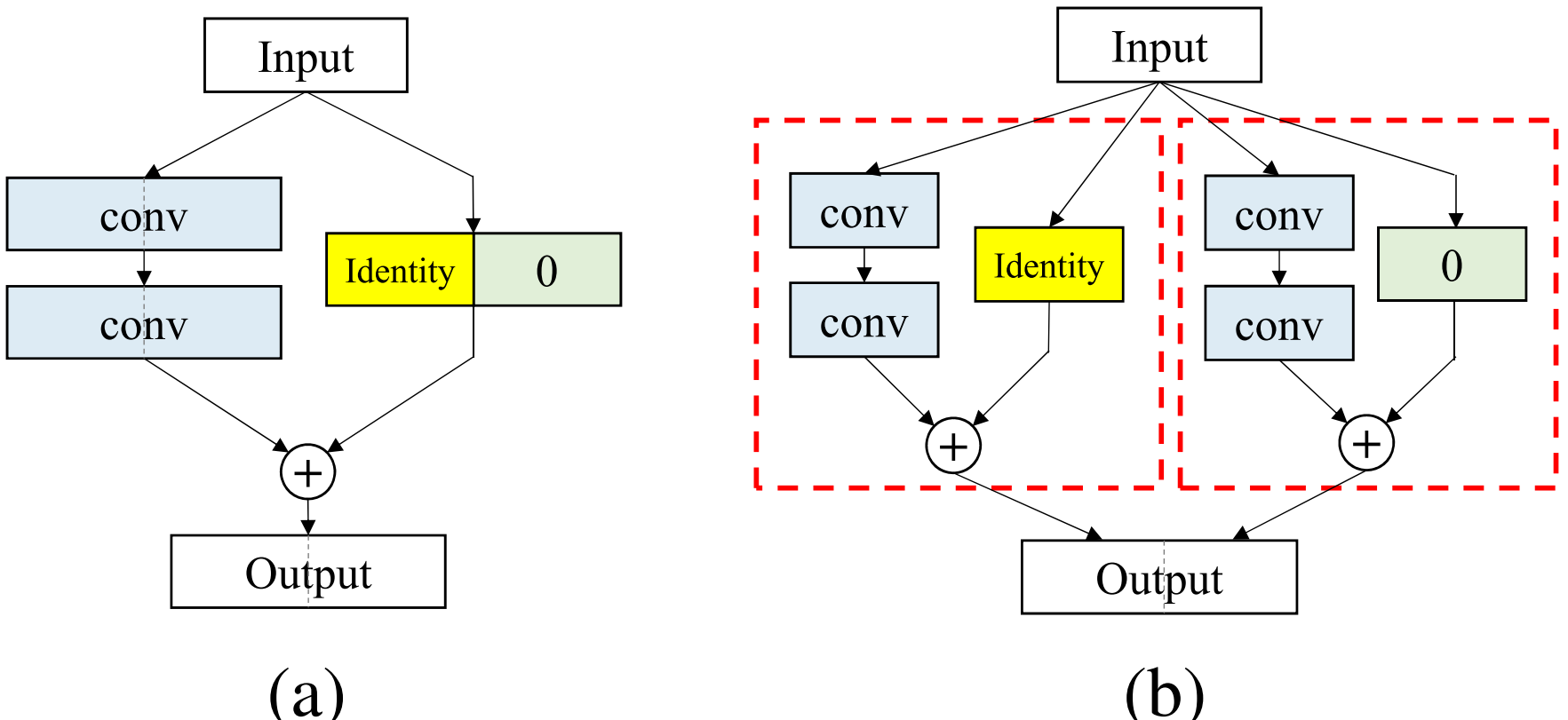


Figure 5 of paper "Deep Pyramidal Residual Networks", https://arxiv.org/abs/1610.02915

PyramidNet – CIFAR Results

Network	# of Params	Output Feat. Dim.	Depth	Training Mem.	CIFAR-10	CIFAR-100
NiN [18]	-	-	-	-	8.81	35.68
All-CNN [27]	-	-	-	-	7.25	33.71
DSN [17]	-	-	-	-	7.97	34.57
FitNet [21]	-	-	-	-	8.39	35.04
Highway [29]	-	-	-	-	7.72	32.39
Fractional Max-pooling [4]	-	-	-	-	4.50	27.62
ELU [29]	-	-	-	-	6.55	24.28
ResNet [7]	1.7M	64	110	547MB	6.43	25.16
ResNet [7]	10.2M	64	1001	2,921MB	-	27.82
ResNet [7]	19.4M	64	1202	2,069MB	7.93	-
Pre-activation ResNet [8]	1.7M	64	164	841MB	5.46	24.33
Pre-activation ResNet [8]	10.2M	64	1001	2,921MB	4.62	22.71
Stochastic Depth [10]	1.7M	64	110	547MB	5.23	24.58
Stochastic Depth [10]	10.2M	64	1202	2,069MB	4.91	-
FractalNet [14]	38.6M	1,024	21	-	4.60	23.73
SwapOut v2 (width×4) [26]	7.4M	256	32	-	4.76	22.72
Wide ResNet (width×4) [34]	8.7M	256	40	775MB	4.97	22.89
Wide ResNet (width×10) [34]	36.5M	640	28	1,383MB	4.17	20.50
Weighted ResNet [24]	19.1M	64	1192	-	5.10	-
DenseNet ($k = 24$) [9]	27.2M	2,352	100	4,381MB	3.74	19.25
DenseNet-BC ($k = 40$) [9]	25.6M	2,190	190	7,247MB	3.46	17.18
PyramidNet ($\alpha = 48$)	1.7M	64	110	655MB	4.58±0.06	23.12±0.04
PyramidNet ($\alpha = 84$)	3.8M	100	110	781MB	4.26±0.23	20.66±0.40
PyramidNet ($\alpha = 270$)	28.3M	286	110	1,437MB	3.73±0.04	18.25±0.10
PyramidNet (bottleneck, $\alpha = 270$)	27.0M	1,144	164	4,169MB	3.48±0.20	17.01±0.39
PyramidNet (bottleneck, $\alpha = 240$)	26.6M	1,024	200	4,451MB	3.44±0.11	16.51±0.13
PyramidNet (bottleneck, $\alpha = 220$)	26.8M	944	236	4,767MB	3.40±0.07	16.37±0.29
PyramidNet (bottleneck, $\alpha = 200$)	26.0M	864	272	5,005MB	**3.31**±0.08	**16.35**±0.24

Table 4 of paper "Deep Pyramidal Residual Networks", https://arxiv.org/abs/1610.02915

Group	Output size	Building Block
conv 1	32×32	$[3 \times 3, 16]$
conv 2	32×32	$\begin{bmatrix} 3 \times 3, \lfloor 16 + \alpha(k-1)/N \rfloor \\ 3 \times 3, \lfloor 16 + \alpha(k-1)/N \rfloor \end{bmatrix} \times N_2$
conv 3	16×16	$\begin{bmatrix} 3 \times 3, \lfloor 16 + \alpha(k-1)/N \rfloor \\ 3 \times 3, \lfloor 16 + \alpha(k-1)/N \rfloor \end{bmatrix} \times N_3$
conv 4	8×8	$\begin{bmatrix} 3 \times 3, \lfloor 16 + \alpha(k-1)/N \rfloor \\ 3 \times 3, \lfloor 16 + \alpha(k-1)/N \rfloor \end{bmatrix} \times N_4$
avg pool	1×1	$[8 \times 8, 16 + \alpha]$

Table 1 of paper "Deep Pyramidal Residual Networks", https://arxiv.org/abs/1610.02915

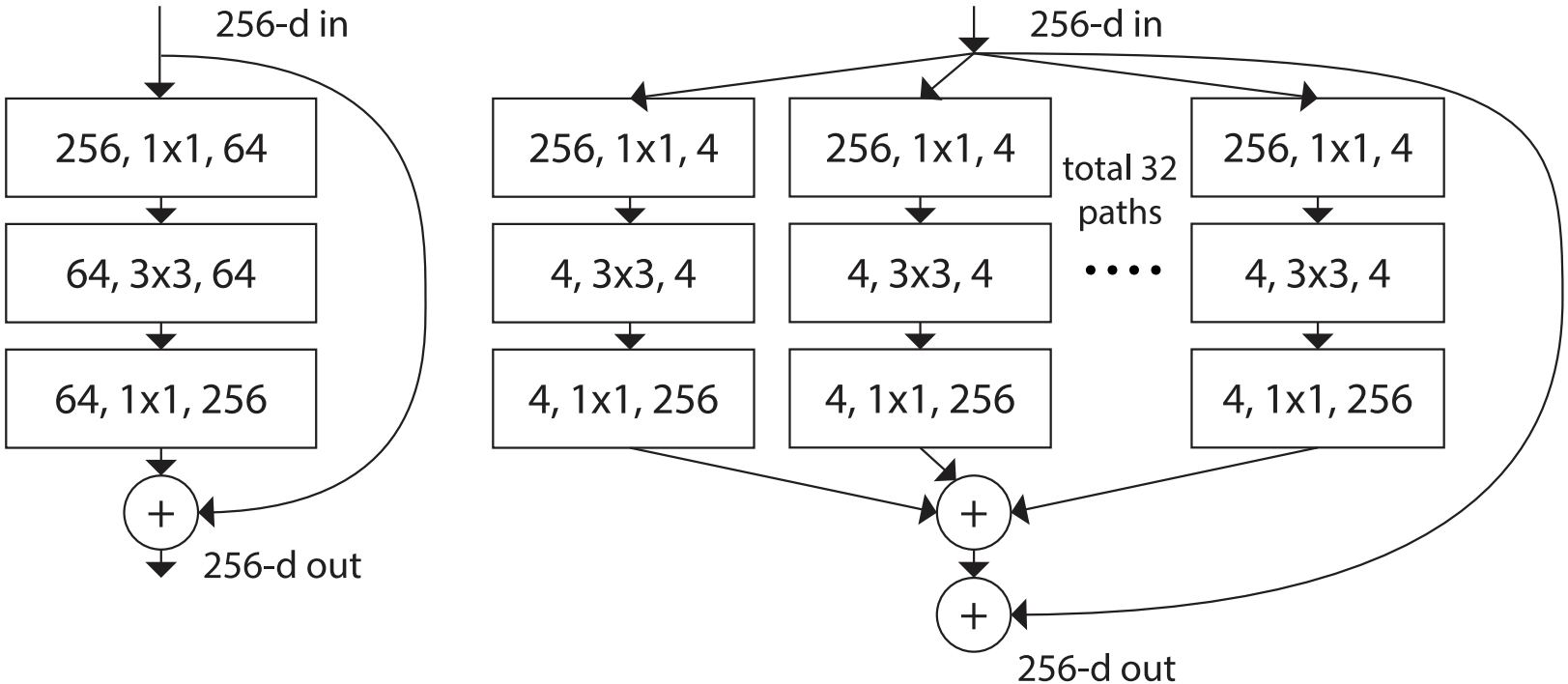


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

Figure 1 of paper "Aggregated Residual Transformations for Deep Neural Networks", https://arxiv.org/abs/1611.05431

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
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix}\times4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix}\times6$
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 5\ \ 2 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		$\mathbf{25.5}\times10^6$	$\mathbf{25.0}\times10^6$
FLOPs		$\mathbf{4.1}\times10^9$	$\mathbf{4.2}\times10^9$

Table 1 of paper "Aggregated Residual Transformations for Deep Neural Networks", https://arxiv.org/abs/1611.05431

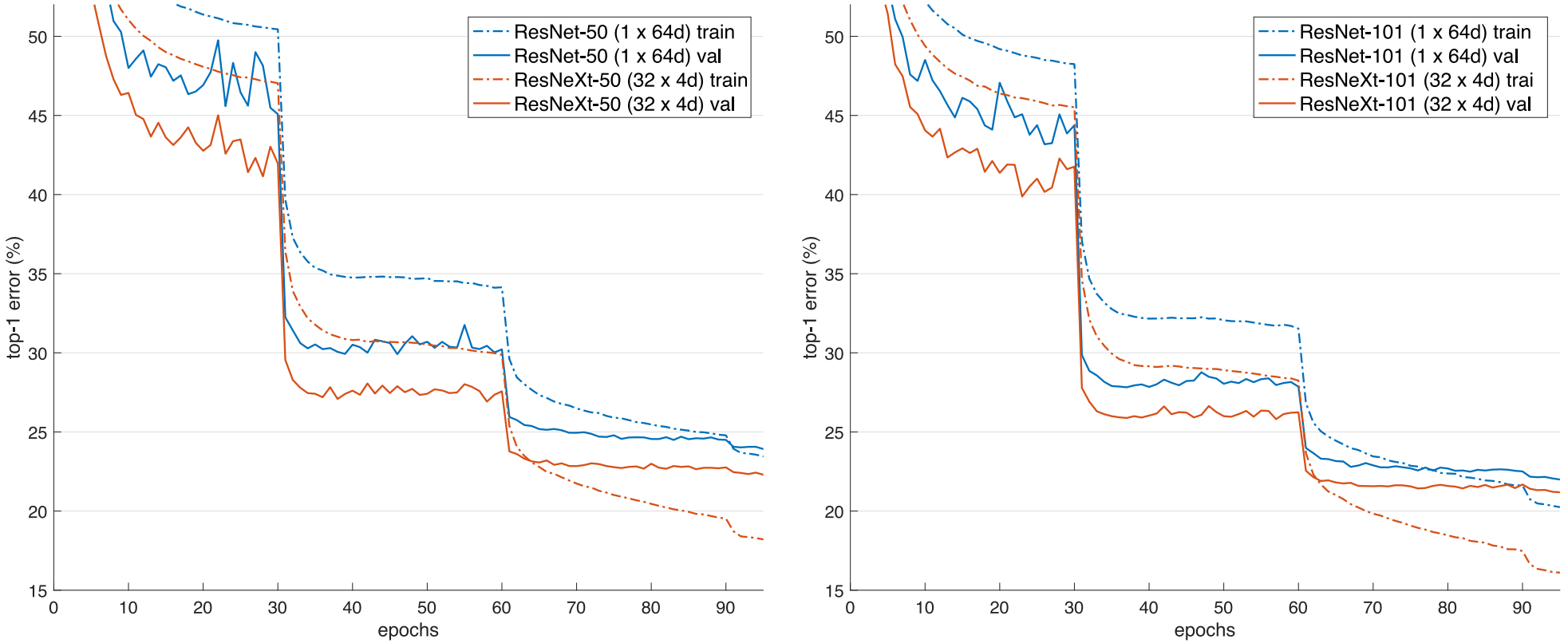


Figure 5. Training curves on ImageNet-1K. (**Left**): ResNet/ResNeXt-50 with preserved complexity (∼4.1 billion FLOPs, ∼25 million parameters); (**Right**): ResNet/ResNeXt-101 with preserved complexity (∼7.8 billion FLOPs, ∼44 million parameters).

Figure 5 of paper "Aggregated Residual Transformations for Deep Neural Networks", https://arxiv.org/abs/1611.05431

Deep Networks with Stochastic Depth

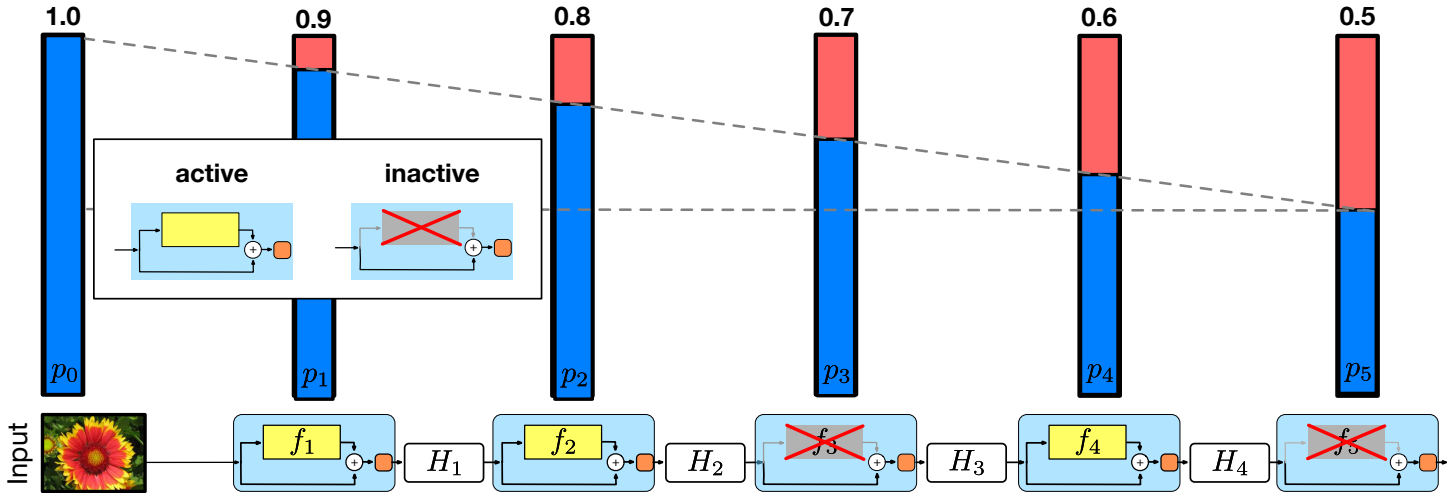
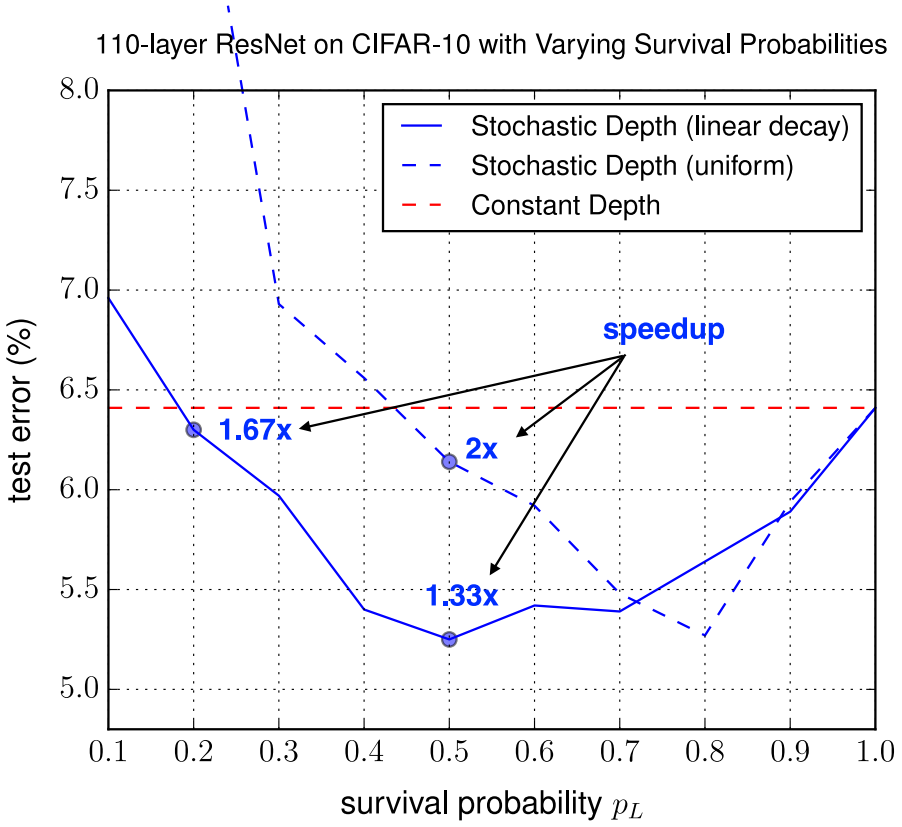


Figure 2 of paper "Deep Networks with Stochastic Depth", https://arxiv.org/abs/1603.09382

We drop a whole block (but not the residual connection) with probability $1 - p_l$. During inference, we multiply the block output by p_l to compensate.

All p_l can be set to a constant, but more effective is to use a simple linear decay $p_l = 1 - l/L(1 - p_L)$ where p_L is the final probability of the last layer, motivated by the intuition that the initial blocks extract low-level features utilized by the later layers and should therefore be present.

Deep Networks with Stochastic Depth



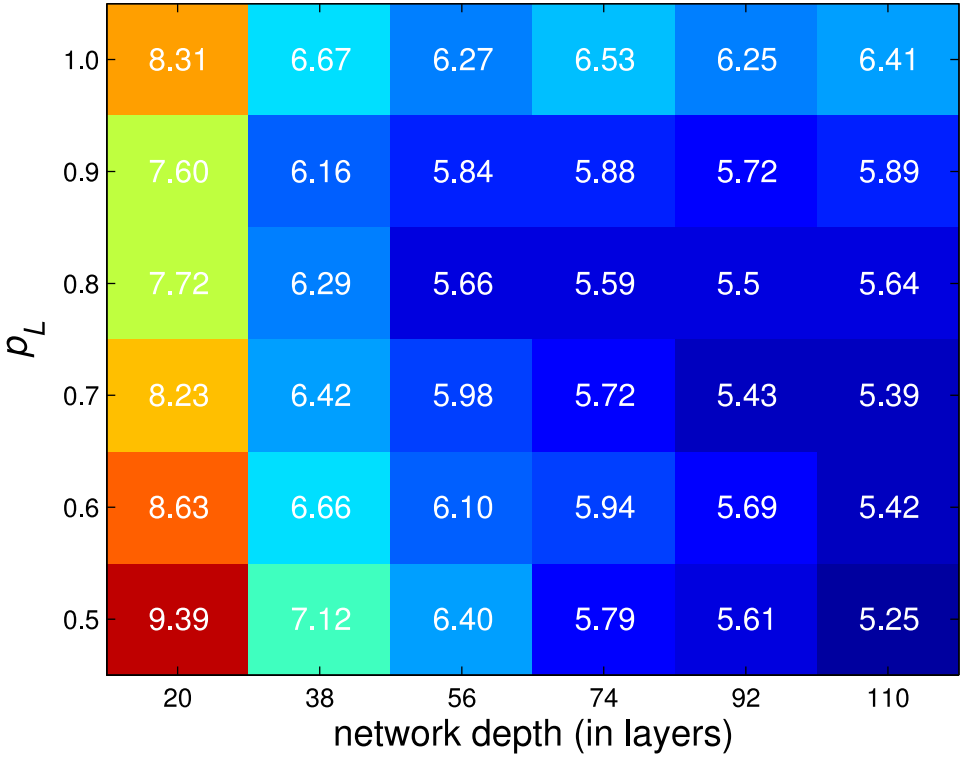
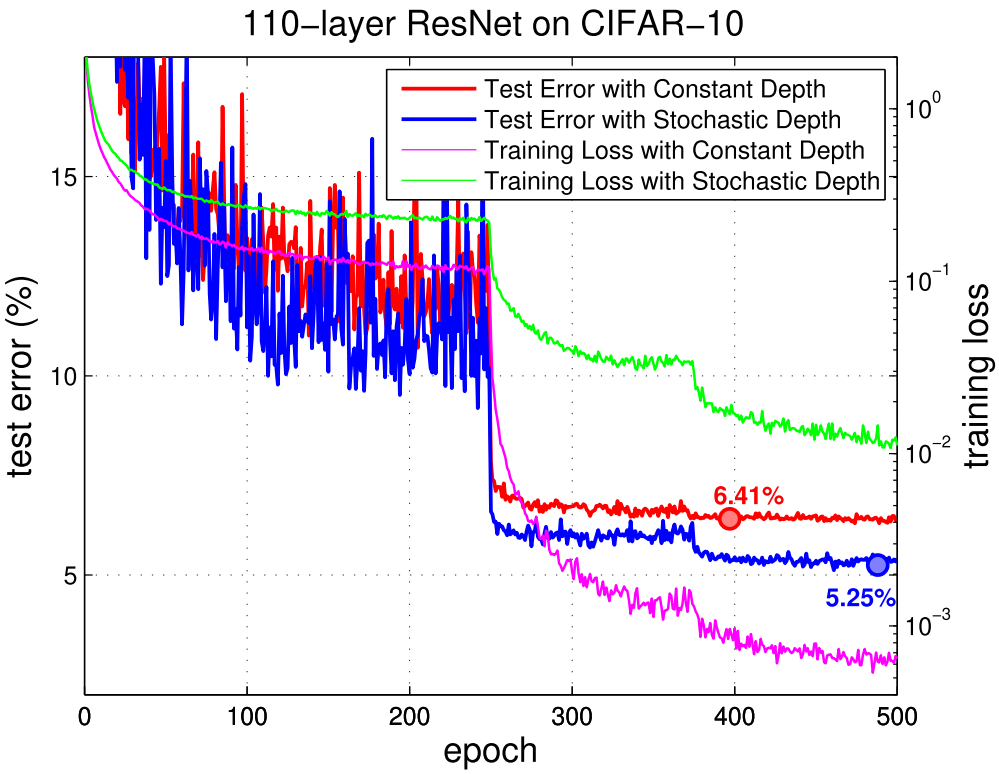


Figure 8 of paper "Deep Networks with Stochastic Depth", https://arxiv.org/abs/1603.09382

Deep Networks with Stochastic Depth



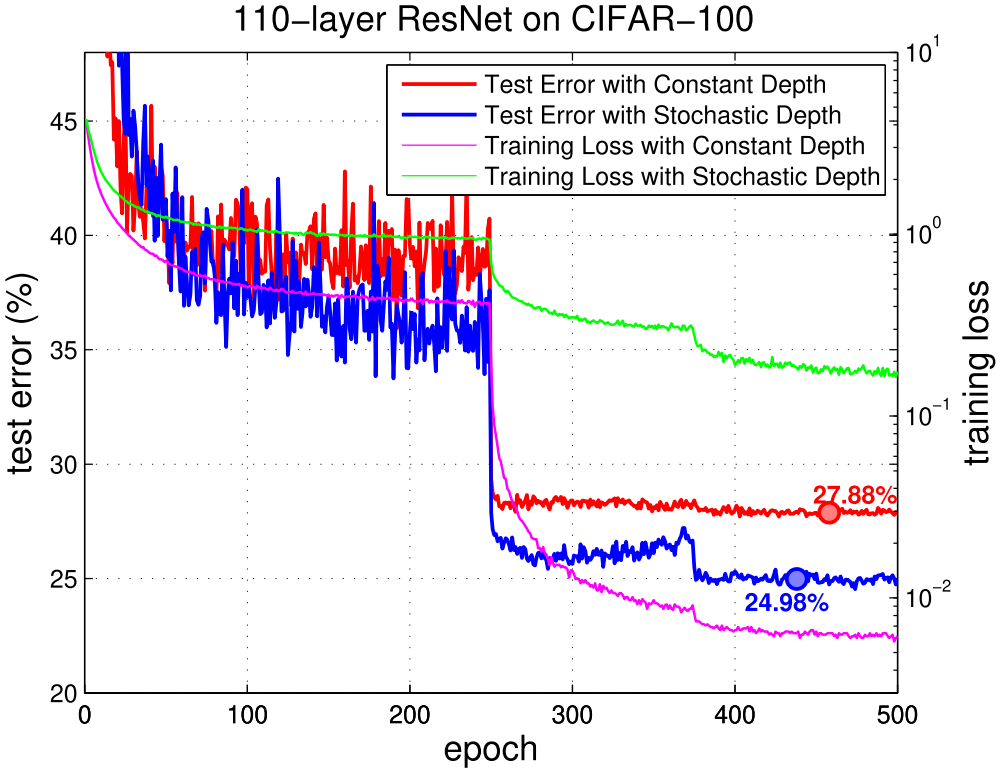


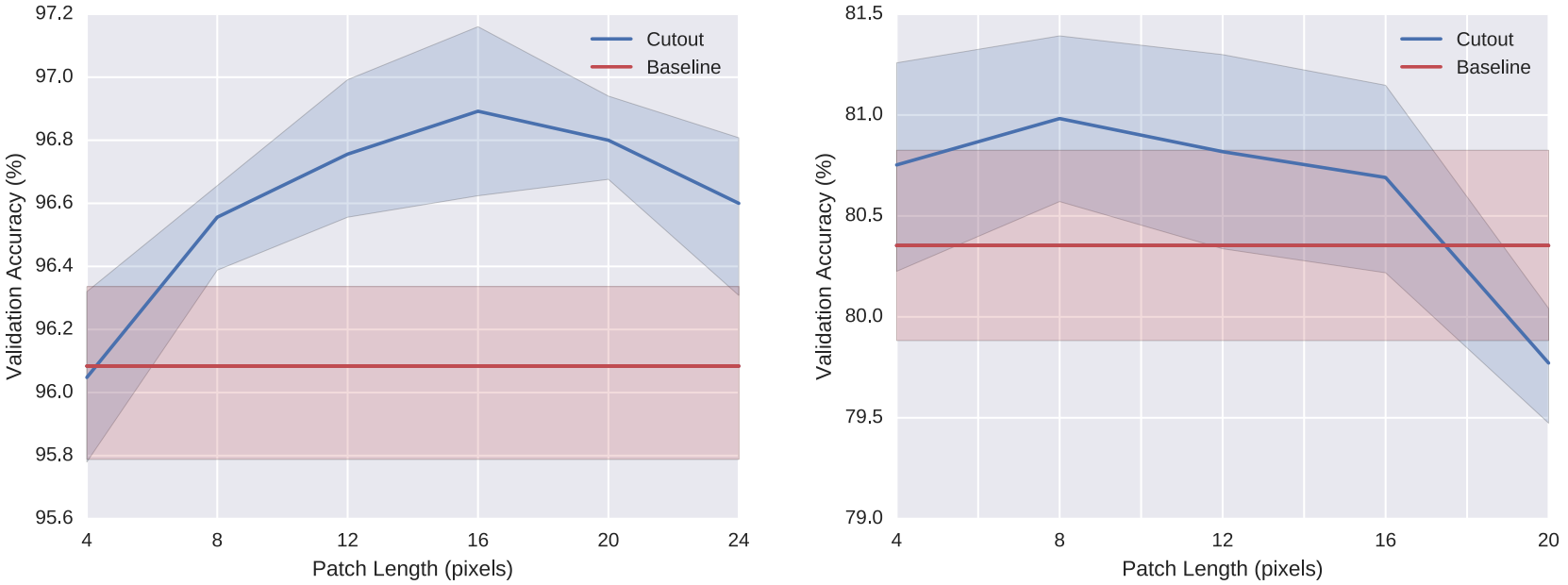
Figure 3 of paper "Deep Networks with Stochastic Depth", https://arxiv.org/abs/1603.09382

Cutout

Figure 1 of paper "Improved Regularization of Convolutional Neural Networks with Cutout", https://arxiv.org/abs/1708.04552

Drop 16×16 square in the input image, with randomly chosen center. The pixels are replaced by a their mean value from the dataset.

Cutout



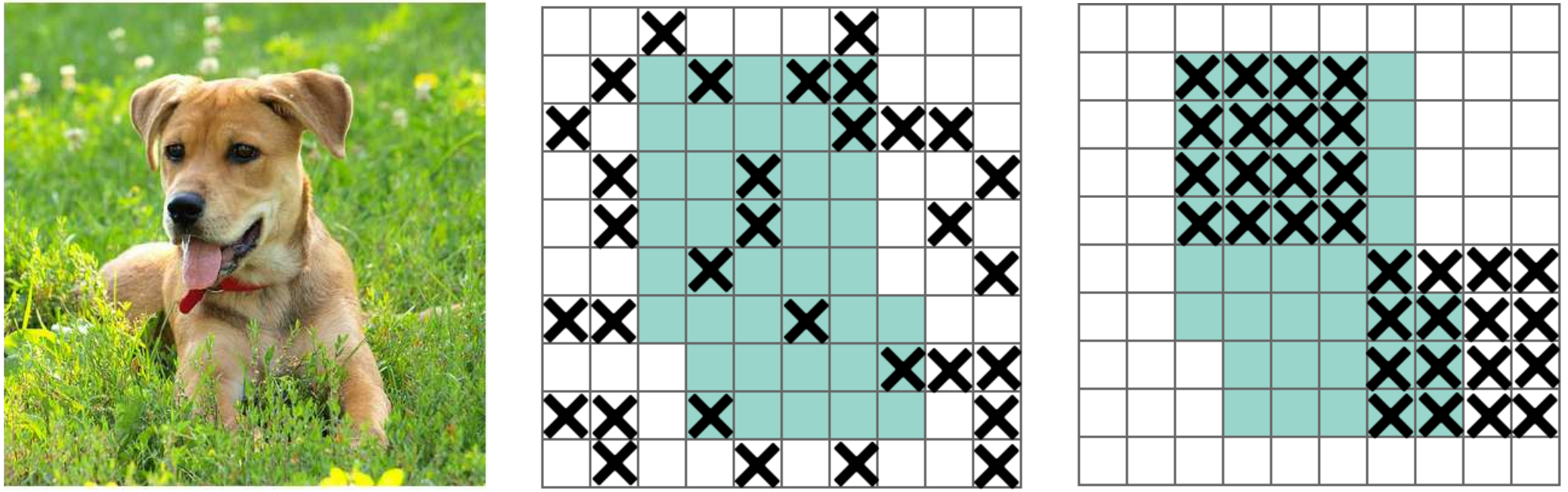
(a) CIFAR-10 (b) CIFAR-100

Figure 3 of paper "Improved Regularization of Convolutional Neural Networks with Cutout", https://arxiv.org/abs/1708.04552

Method	C10	C10+	C100	C100+	SVHN
ResNet18 [5]	10.63 ± 0.26	4.72 ± 0.21	36.68 ± 0.57	22.46 ± 0.31	-
ResNet18 + cutout	9.31 ± 0.18	3.99 ± 0.13	34.98 ± 0.29	21.96 ± 0.24	-
WideResNet [22]	6.97 ± 0.22	3.87 ± 0.08	26.06 ± 0.22	18.8 ± 0.08	1.60 ± 0.05
WideResNet + cutout	$\mathbf{5.54 \pm 0.08}$	3.08 ± 0.16	$\mathbf{23.94 \pm 0.15}$	18.41 ± 0.27	$\mathbf{1.30 \pm 0.03}$
Shake-shake regularization [4]	-	2.86	-	15.85	-
Shake-shake regularization + cutout	-	$\mathbf{2.56 \pm 0.07}$	-	$\mathbf{15.20 \pm 0.21}$	-

Table 1 of paper "Improved Regularization of Convolutional Neural Networks with Cutout", https://arxiv.org/abs/1708.04552

DropBlock

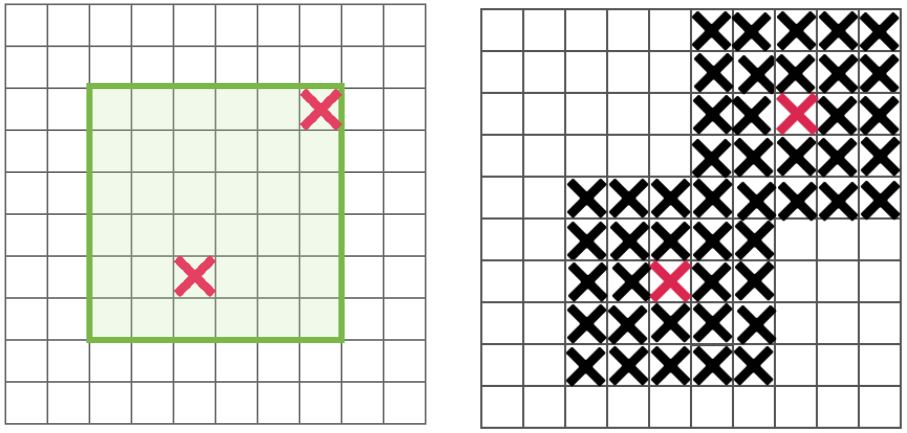


(a) (b) (c)

Figure 1 of paper "DropBlock: A regularization method for convolutional networks", https://arxiv.org/abs/1810.12890

Algorithm 1 DropBlock

1: **Input:**output activations of a layer (A), $block\_size$, γ, $mode$
2: **if** $mode$ == *Inference* **then**
3: return A
4: **end if**
5: Randomly sample mask M: $M_{i,j} \sim Bernoulli(\gamma)$
6: For each zero position $M_{i,j}$, create a spatial square mask with the center being $M_{i,j}$, the width, height being $block\_size$ and set all the values of M in the square to be zero (see Figure 2).
7: Apply the mask: $A = A \times M$
8: Normalize the features: $A = A \times$ **count**(M)/**count_ones**(M)



(a) (b)

Figure 2 of paper "DropBlock: A regularization method for convolutional networks", https://arxiv.org/abs/1810.12890

DropBlock

Model	top-1(%)	top-5(%)
ResNet-50	76.51 ± 0.07	93.20 ± 0.05
ResNet-50 + dropout (kp=0.7) [1]	76.80 ± 0.04	93.41 ± 0.04
ResNet-50 + DropPath (kp=0.9) [17]	77.10 ± 0.08	93.50 ± 0.05
ResNet-50 + SpatialDropout (kp=0.9) [20]	77.41 ± 0.04	93.74 ± 0.02
ResNet-50 + Cutout [23]	76.52 ± 0.07	93.21 ± 0.04
ResNet-50 + AutoAugment [27]	77.63	93.82
ResNet-50 + label smoothing (0.1) [28]	77.17 ±0.05	93.45 ±0.03
ResNet-50 + DropBlock, (kp=0.9)	78.13 ± 0.05	94.02 ± 0.02
ResNet-50 + DropBlock (kp=0.9) + label smoothing (0.1)	78.35 ± 0.05	94.15 ± 0.03

Table 1 of paper "DropBlock: A regularization method for convolutional networks", https://arxiv.org/abs/1810.12890

Beyond Image Classification

Beyond Image Classification

- Object detection (including location)

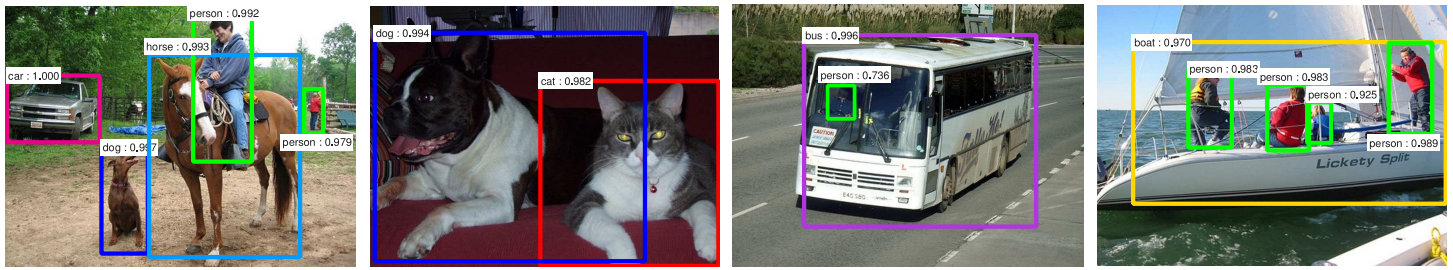


Figure 3 of paper "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", https://arxiv.org/abs/1506.01497

- Image segmentation

Figure 2 of paper "Mask R-CNN", https://arxiv.org/abs/1703.06870.

- Human pose estimation

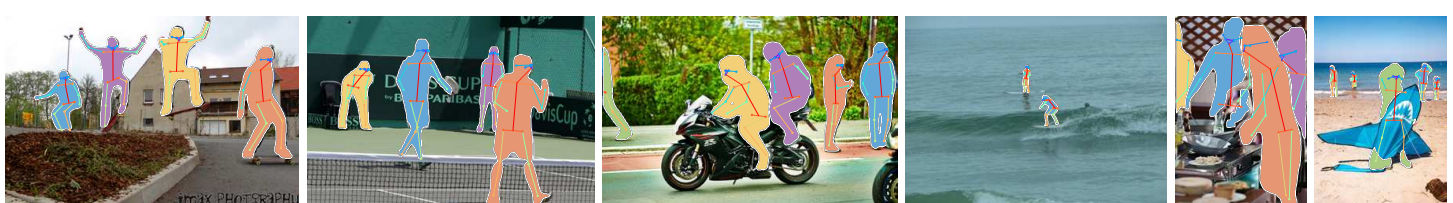


Figure 7 of paper "Mask R-CNN", https://arxiv.org/abs/1703.06870.

- Start with a network pre-trained on ImageNet (VGG-16 is used in the original paper).

RoI Pooling

- Crucial for fast performance.
- The last max-pool layer ($14 \times 14 \rightarrow 7 \times 7$ in VGG) is replaced by a RoI pooling layer, producing output of the same size. For each output sub-window we max-pool the corresponding values in the output layer.
- Two sibling layers are added, one predicting $K + 1$ categories and the other one predicting 4 bounding box parameters for each of K categories.

Fast R-CNN

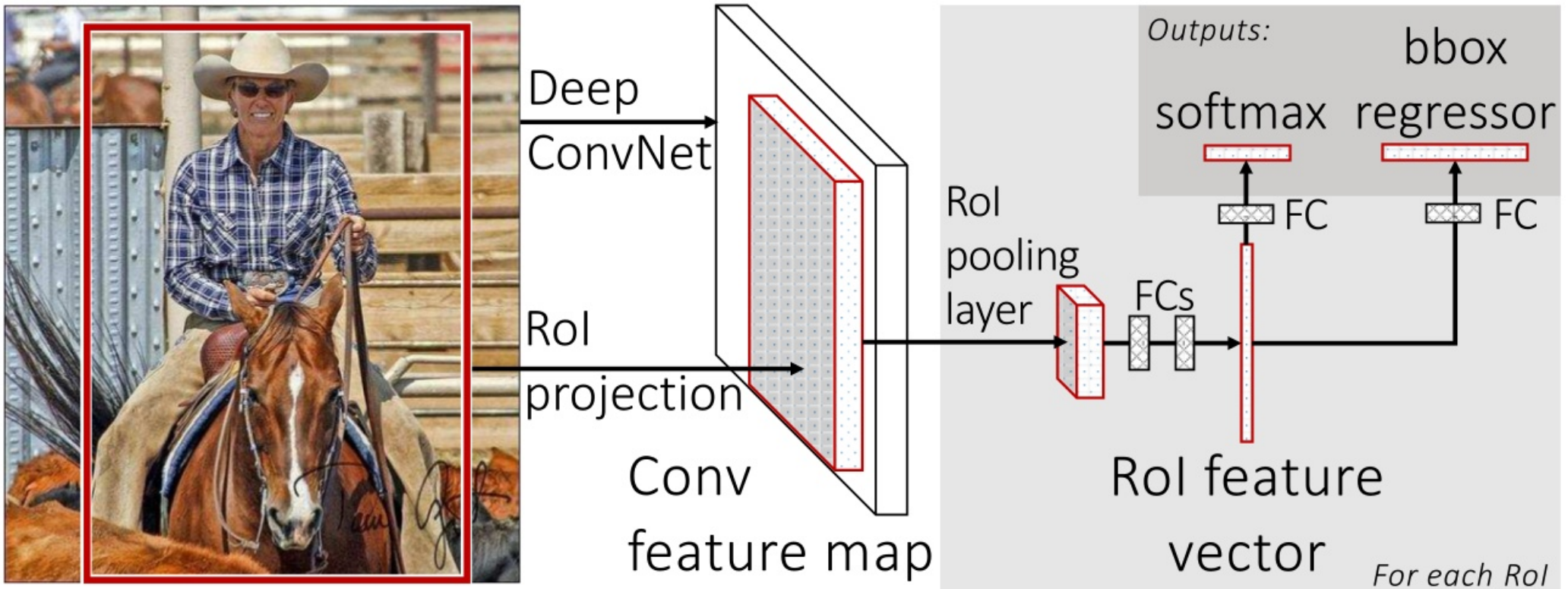


Figure 1 of paper "Fast R-CNN", https://arxiv.org/abs/1504.08083.

The bounding box is parametrized as follows. Let x_r, y_r, w_r, h_r be center coordinates and width and height of the RoI, and let x, y, w, h be parameters of the bounding box. We represent them as follows:

$$t_x = (x - x_r)/w_r, \quad t_y = (y - y_r)/h_r$$
$$t_w = \log(w/w_r), \quad t_h = \log(h/h_r)$$

Usually a smooth_{L_1} loss, or *Huber loss*, is employed for bounding box parameters

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

The complete loss is then

$$L(\hat{c}, \hat{t}, c, t) = L_{\text{cls}}(\hat{c}, c) + \lambda[c \geq 1] \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}(\hat{t}_i - t_i).$$

Intersection over union

For two bounding boxes (or two masks) the *intersection over union* (*IoU*) is a ration of the intersection of the boxes (or masks) and the union of the boxes (or masks).

Choosing RoIs for training

During training, we use 2 images with 64 RoIs each. The RoIs are selected so that 25% have intersection over union (IoU) overlap with ground-truth boxes at least 0.5; the others are chosen to have the IoU in range $[0.1, 0.5)$.

Choosing RoIs during inference

Single object can be found in multiple RoIs. To choose the most salient one, we perform *non-maximum suppression* -- we ignore RoIs which have an overlap with a higher scoring RoI of the same type, where the IoU is larger than a given threshold (usually, 0.3 is used). Higher scoring RoI is the one with higher probability from the classification head.

Average Precision

Evaluation is performed using *Average Precision* (*AP*).

We assume all bounding boxes (or masks) produced by a system have confidence values which can be used to rank them. Then, for a single class, we take the boxes (or masks) in the order of the ranks and generate precision/recall curve, considering a bounding box correct if it has IoU at least 0.5 with any ground-truth box. We define *AP* as an average of precisions for recall levels $0, 0.1, 0.2, \ldots, 1$.

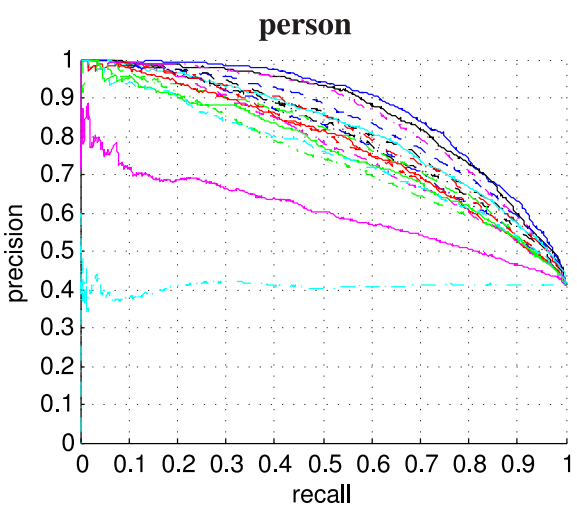


Figure 6 of paper "The PASCAL Visual Object Classes (VOC) Challenge", http://homepages.inf.ed.ac.uk/ckiw/postscript/ijcv_voc09.pdf.

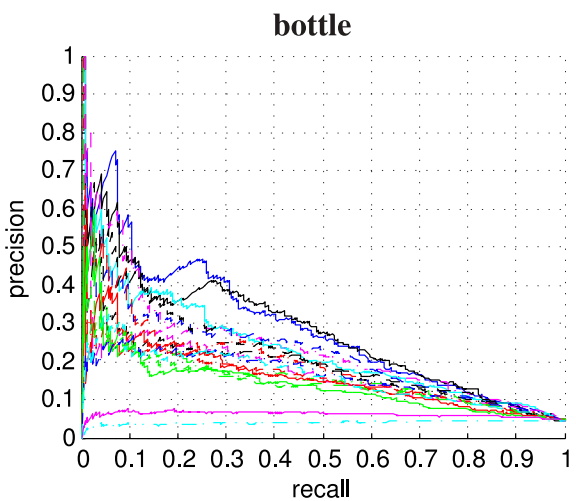


Figure 6 of paper "The PASCAL Visual Object Classes (VOC) Challenge", http://homepages.inf.ed.ac.uk/ckiw/postscript/ijcv_voc09.pdf.

Faster R-CNN

For Fast R-CNN, the most time consuming part is generating the RoIs.

Therefore, Faster R-CNN jointly generates *regions of interest* using a *region proposal network* and performs object detection.

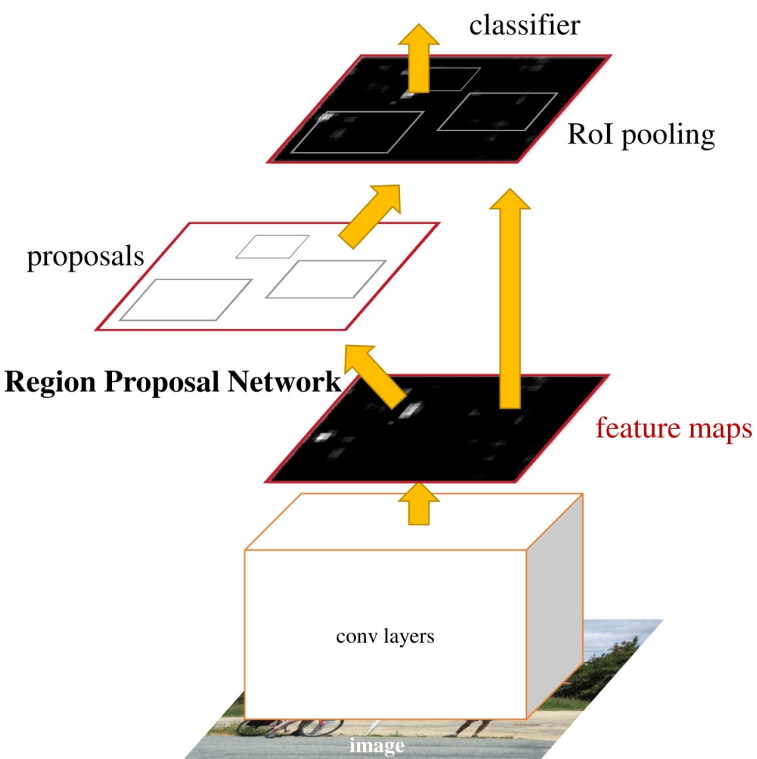


Figure 2 of paper "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", https://arxiv.org/abs/1506.01497

Faster R-CNN

The region proposals are generated using a 3×3 sliding window, with 3 different scales (128^2, 256^2 and 512^2) and 3 aspect ratios ($1:1$, $1:2$, $2:1$). For every anchor, there is a Fast-R-CNN-like object detection head – a classification into two classes (background, object) and a boundary regressor.

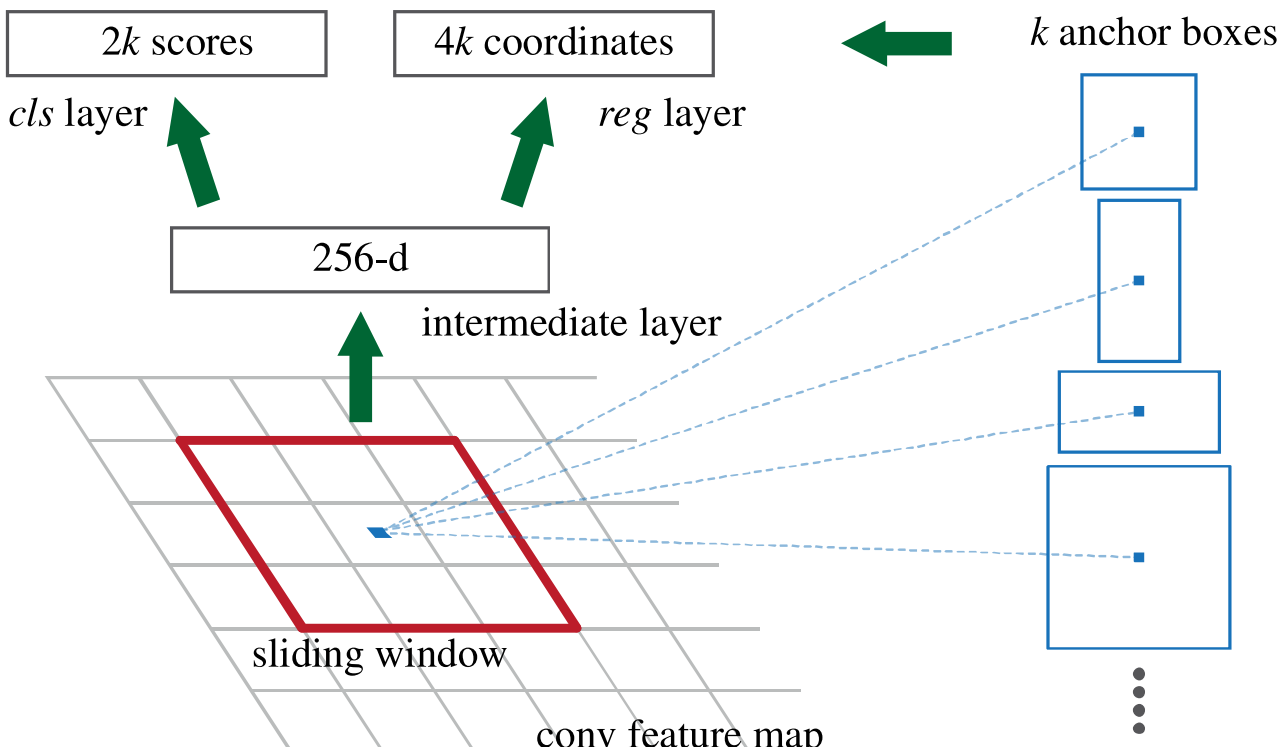


Figure 3 of paper "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", https://arxiv.org/abs/1506.01497

Faster R-CNN

During training, we generate

- positive training examples for every anchor that has highest IoU with a ground-truth box;
- furthermore, a positive example is also any anchor with IoU at least 0.7 for any ground-trugh box;
- negative training examples for every anchor that has IoU at most 0.3 with all ground-truth boxes.

During inference, we consider all predicted non-background regions, run non-maximum suppresion on them using a 0.7 IoU threshold, and then take N top-scored regions (i.e., the ones with highest probability from the classification head) – the paper uses 300 proposals, compared to 2000 in the Fast R-CNN.

Table 3: Detection results on **PASCAL VOC 2007 test set**. The detector is Fast R-CNN and VGG-16. Training data: "07": VOC 2007 trainval, "07+12": union set of VOC 2007 trainval and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2000. †: this number was reported in [2]; using the repository provided by this paper, this result is higher (68.1).

method	# proposals	data	mAP (%)
SS	2000	07	66.9†
SS	2000	07+12	70.0
RPN+VGG, unshared	300	07	68.5
RPN+VGG, shared	300	07	69.9
RPN+VGG, shared	300	07+12	**73.2**
RPN+VGG, shared	300	COCO+07+12	**78.8**

Table 4: Detection results on **PASCAL VOC 2012 test set**. The detector is Fast R-CNN and VGG-16. Training data: "07": VOC 2007 trainval, "07++12": union set of VOC 2007 trainval+test and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2000. †: http://host.robots.ox.ac.uk:8080/anonymous/HZJTQA.html. ‡: http://host.robots.ox.ac.uk:8080/anonymous/YNPLXB.html. §: http://host.robots.ox.ac.uk:8080/anonymous/XEDH10.html.

method	# proposals	data	mAP (%)
SS	2000	12	65.7
SS	2000	07++12	68.4
RPN+VGG, shared†	300	12	67.0
RPN+VGG, shared‡	300	07++12	**70.4**
RPN+VGG, shared§	300	COCO+07++12	**75.9**

Tables 3 and 4 of paper "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", https://arxiv.org/abs/1506.01497

Mask R-CNN

"Straightforward" extension of Faster R-CNN able to produce image segmentation (i.e., masks for every object).

Figure 2 of paper "Mask R-CNN", https://arxiv.org/abs/1703.06870.

Mask R-CNN

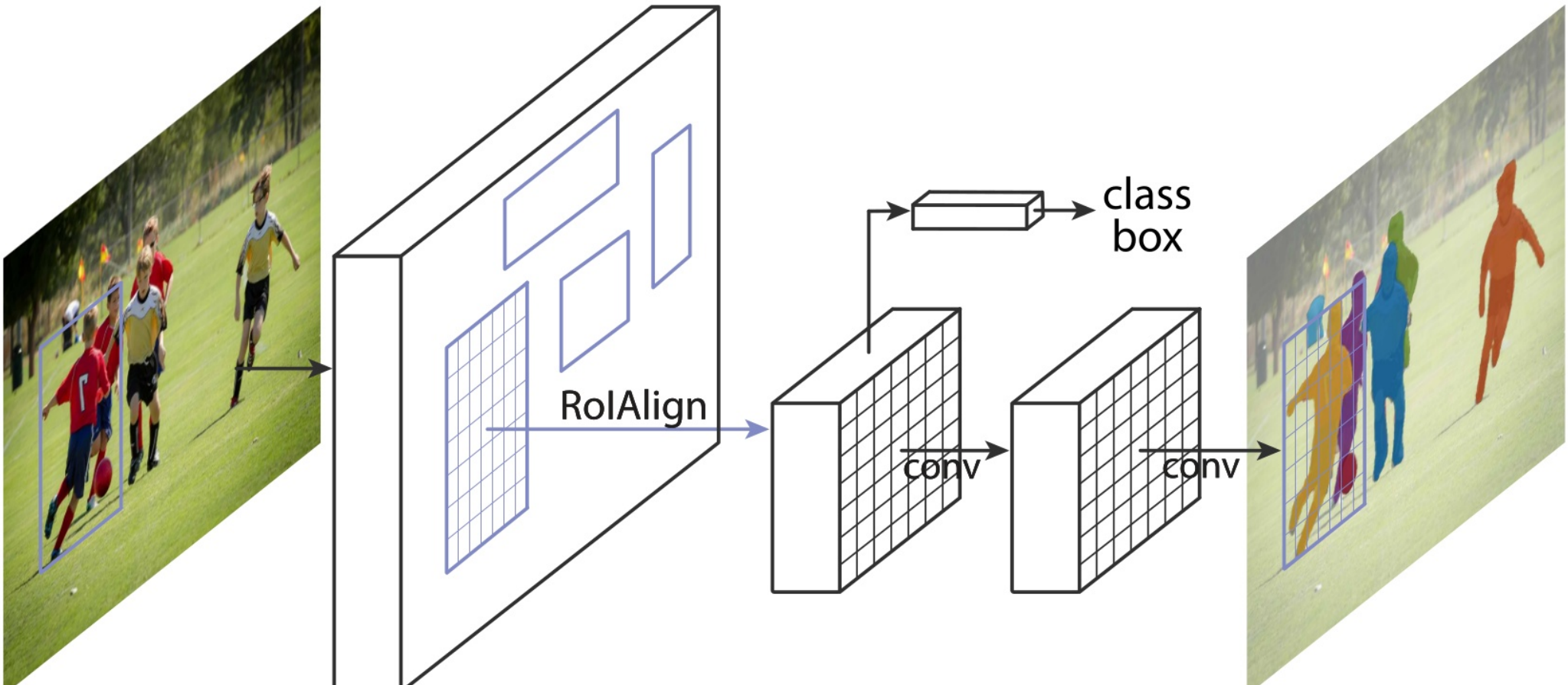


Figure 1 of paper "Mask R-CNN", https://arxiv.org/abs/1703.06870.

RoIAlign

More precise alignment is required for the RoI in order to predict the masks. Therefore, instead of max-pooling used in the RoI pooling, RoIAlign with bilinear interpolation is used.

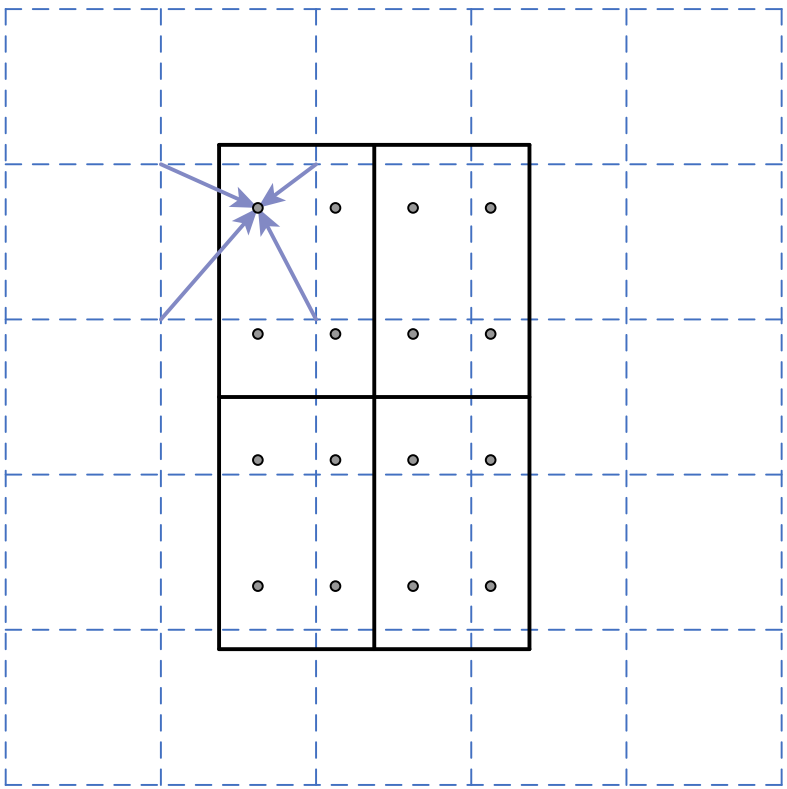


Figure 3 of paper "Mask R-CNN", https://arxiv.org/abs/1703.06870.

Mask R-CNN

Masks are predicted in a third branch of the object detector.

- Usually higher resolution is needed (14×14 instead of 7×7).
- The masks are predicted for each class separately.
- The masks are predicted using convolutions instead of fully connected layers.

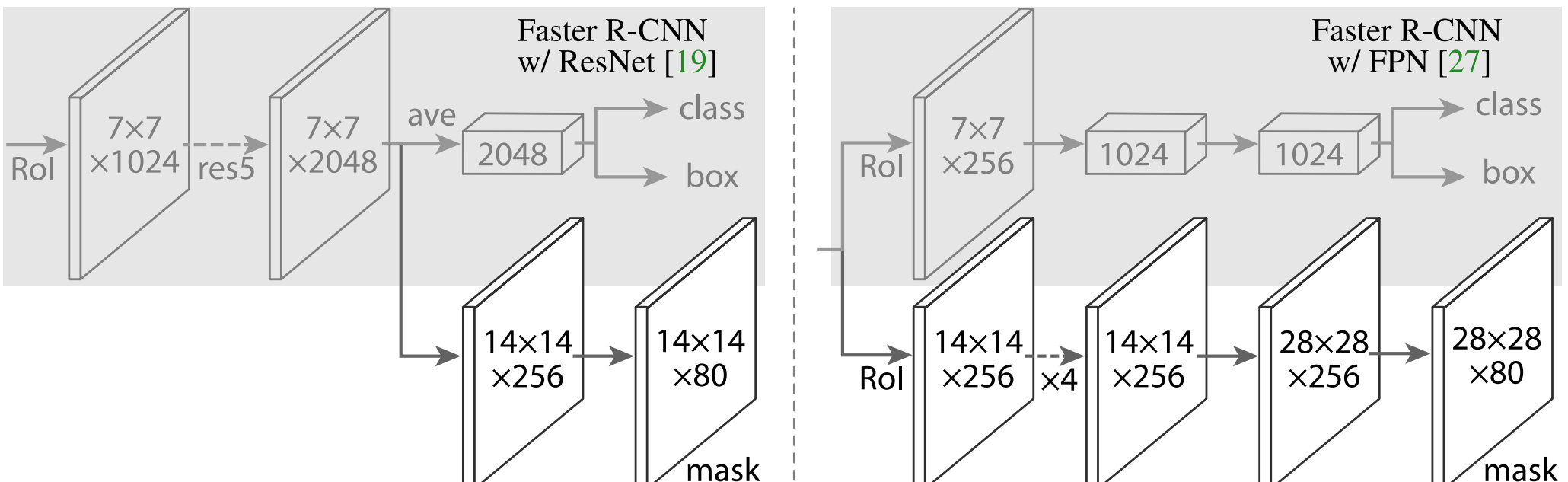


Figure 4 of paper "Mask R-CNN", https://arxiv.org/abs/1703.06870.

net-depth-features	AP	AP_{50}	AP_{75}
ResNet-50-C4	30.3	51.2	31.5
ResNet-101-C4	32.7	54.2	34.3
ResNet-50-FPN	33.6	55.2	35.3
ResNet-101-FPN	35.4	57.3	37.5
ResNeXt-101-FPN	**36.7**	**59.5**	**38.9**

(a) **Backbone Architecture**: Better backbones bring expected gains: deeper networks do better, FPN outperforms C4 features, and ResNeXt improves on ResNet.

	AP	AP_{50}	AP_{75}
softmax	24.8	44.1	25.1
sigmoid	**30.3**	**51.2**	**31.5**
	+5.5	*+7.1*	*+6.4*

(b) **Multinomial *vs.* Independent Masks** (ResNet-50-C4): *Decoupling* via per-class binary masks (sigmoid) gives large gains over multinomial masks (softmax).

	align?	bilinear?	agg.	AP	AP_{50}	AP_{75}
RoIPool [12]			max	26.9	48.8	26.4
RoIWarp [10]		✓	max	27.2	49.2	27.1
		✓	ave	27.1	48.9	27.1
RoIAlign	✓	✓	max	**30.2**	**51.0**	**31.8**
	✓	✓	ave	**30.3**	**51.2**	**31.5**

(c) **RoIAlign** (ResNet-50-C4): Mask results with various RoI layers. Our RoIAlign layer improves AP by ~3 points and AP_{75} by ~5 points. Using proper alignment is the only factor that contributes to the large gap between RoI layers.

	AP	AP_{50}	AP_{75}	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}
RoIPool	23.6	46.5	21.6	28.2	52.7	26.9
RoIAlign	**30.9**	**51.8**	**32.1**	**34.0**	**55.3**	**36.4**
	+7.3	*+ 5.3*	*+10.5*	*+5.8*	*+2.6*	*+9.5*

(d) **RoIAlign** (ResNet-50-**C5**, *stride 32*): Mask-level and box-level AP using *large-stride* features. Misalignments are more severe than with stride-16 features (Table 2c), resulting in big accuracy gaps.

	mask branch	AP	AP_{50}	AP_{75}
MLP	fc: $1024 \to 1024 \to 80 \cdot 28^2$	31.5	53.7	32.8
MLP	fc: $1024 \to 1024 \to 1024 \to 80 \cdot 28^2$	31.5	54.0	32.6
FCN	conv: $256 \to 256 \to 256 \to 256 \to 256 \to 80$	**33.6**	**55.2**	**35.3**

(e) **Mask Branch** (ResNet-50-FPN): Fully convolutional networks (FCN) *vs.* multi-layer perceptrons (MLP, fully-connected) for mask prediction. FCNs improve results as they take advantage of explicitly encoding spatial layout.

Table 2. **Ablations**. We train on `trainval35k`, test on `minival`, and report *mask* AP unless otherwise noted.

Table 2 of paper "Mask R-CNN", https://arxiv.org/abs/1703.06870.