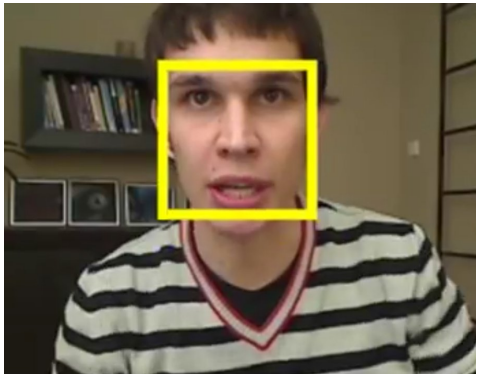
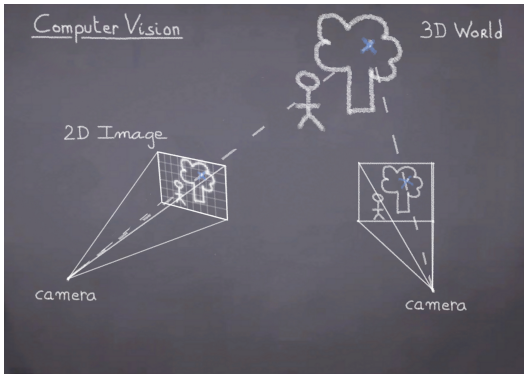
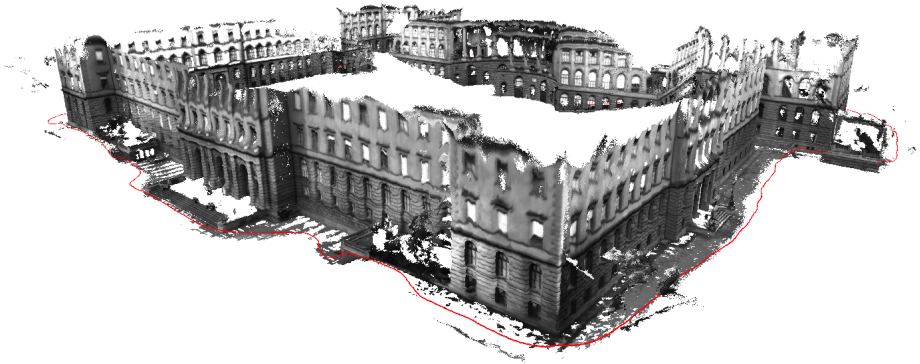
Computer vision and CNN architectures

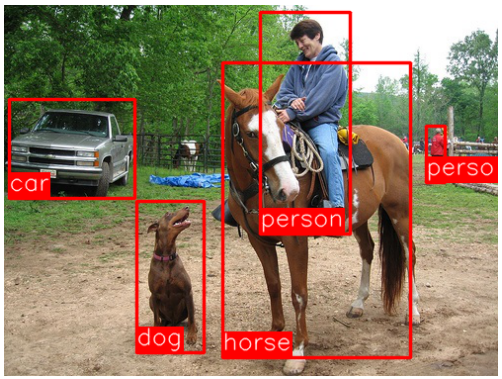
Ronan Sicre
Credits to Yannis Avrithis https://sif-dlv.github.io/

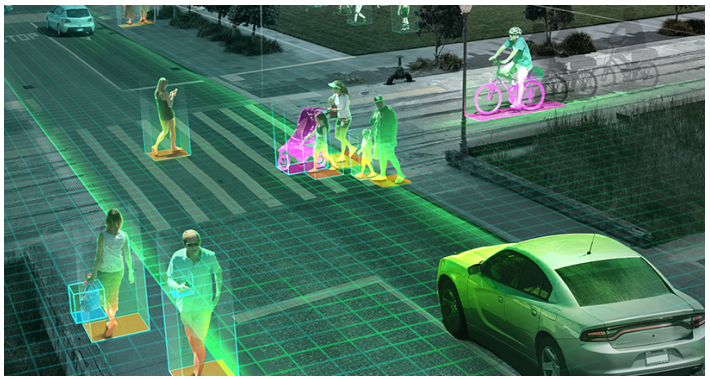
computer vision in images



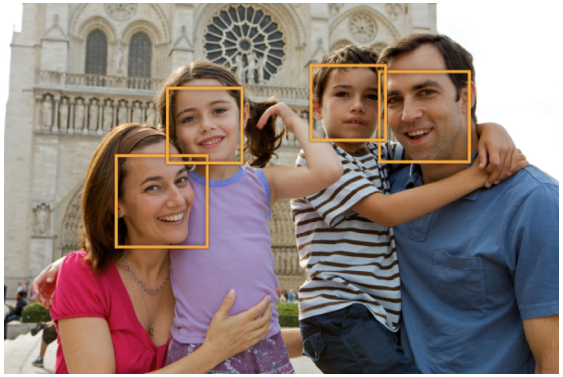


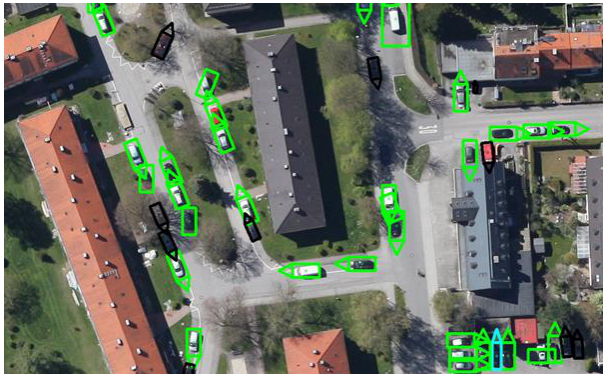




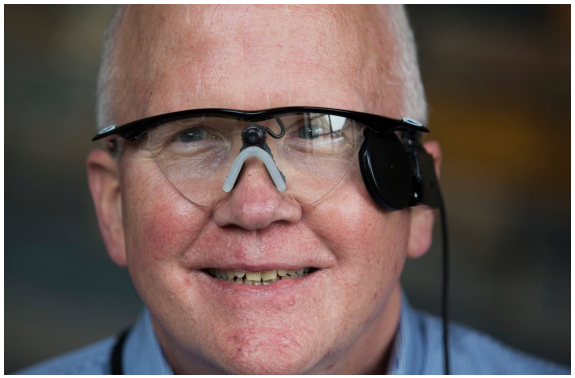
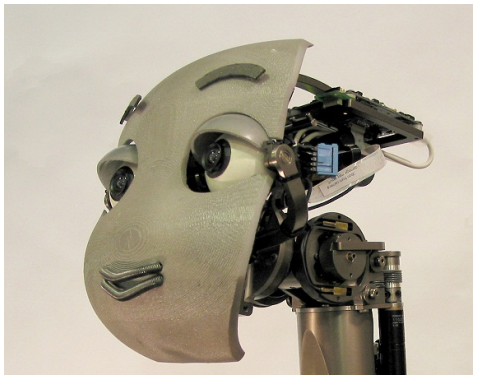


computer vision in images

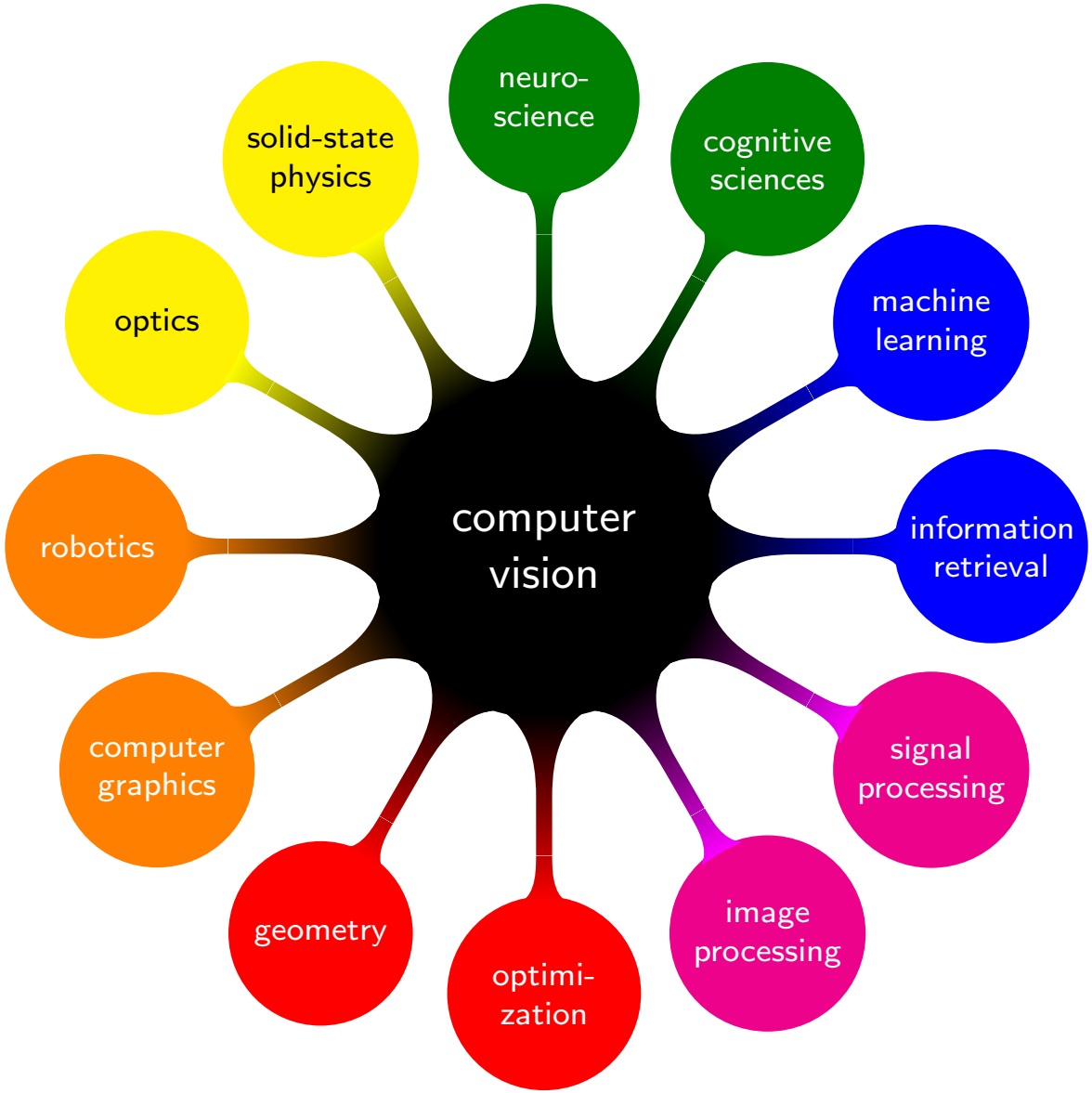




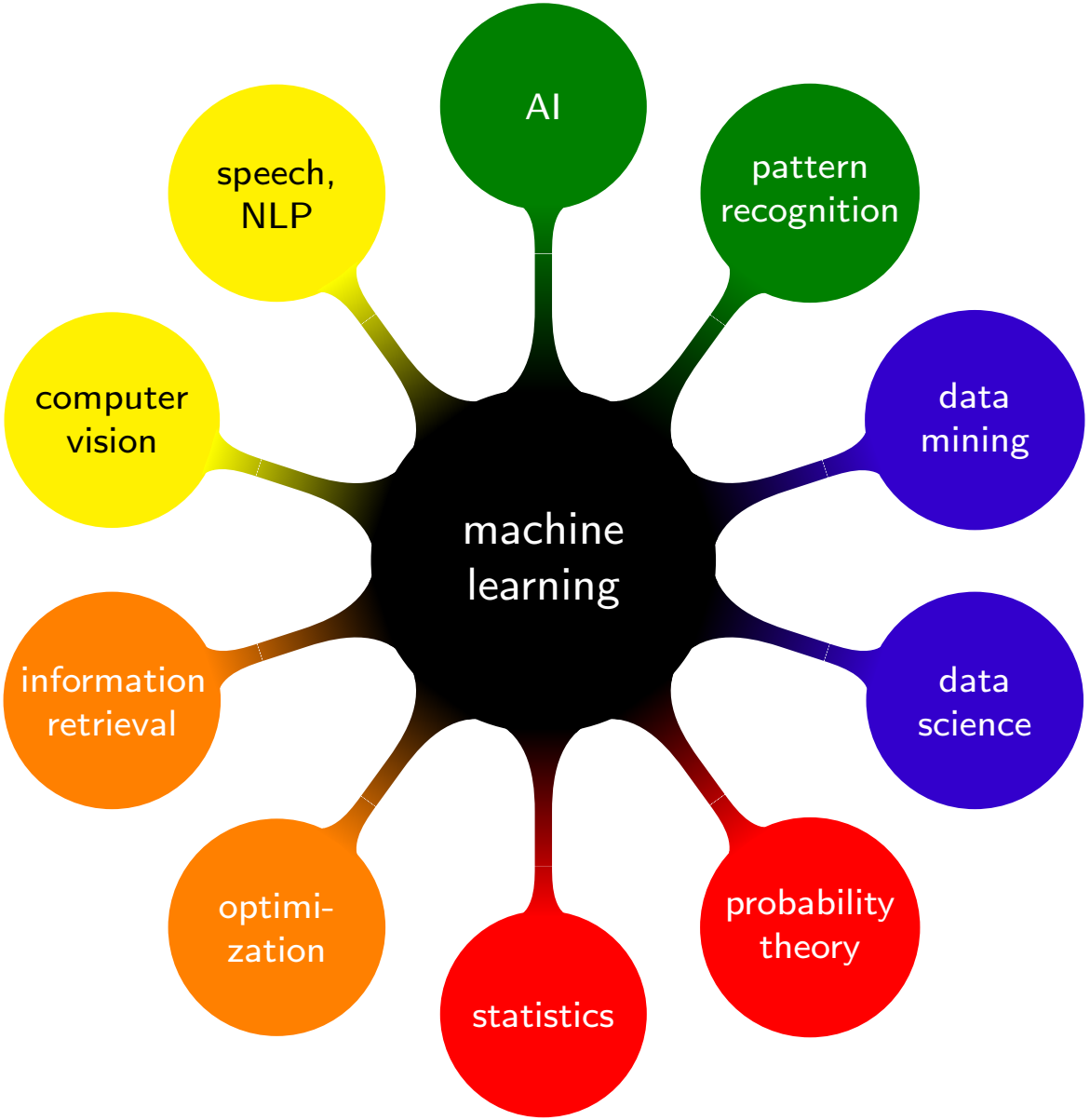


computer vision—related fields



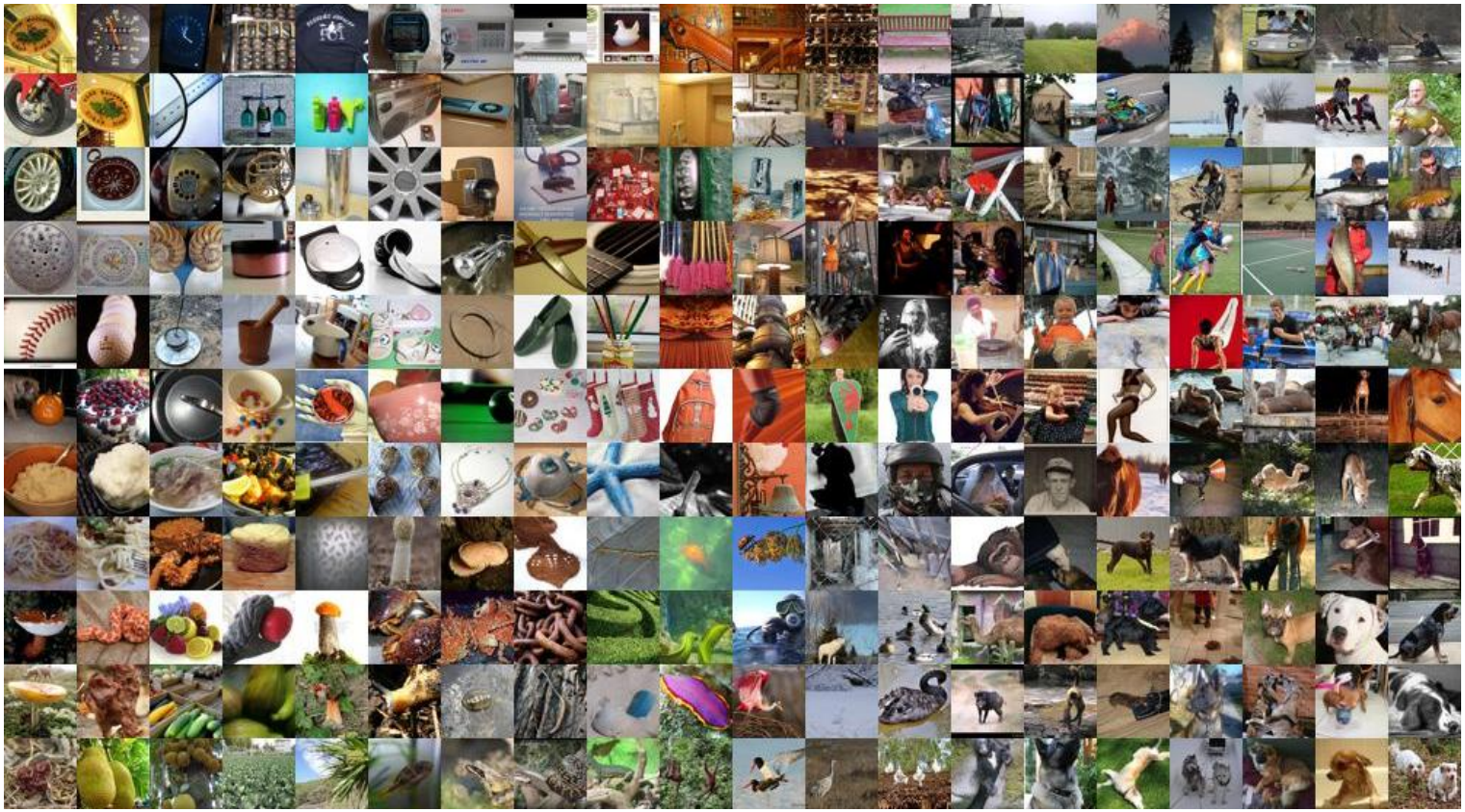
machine learning—related fields



modern deep learning

ImageNet

[Russakovsky et al. 2014]

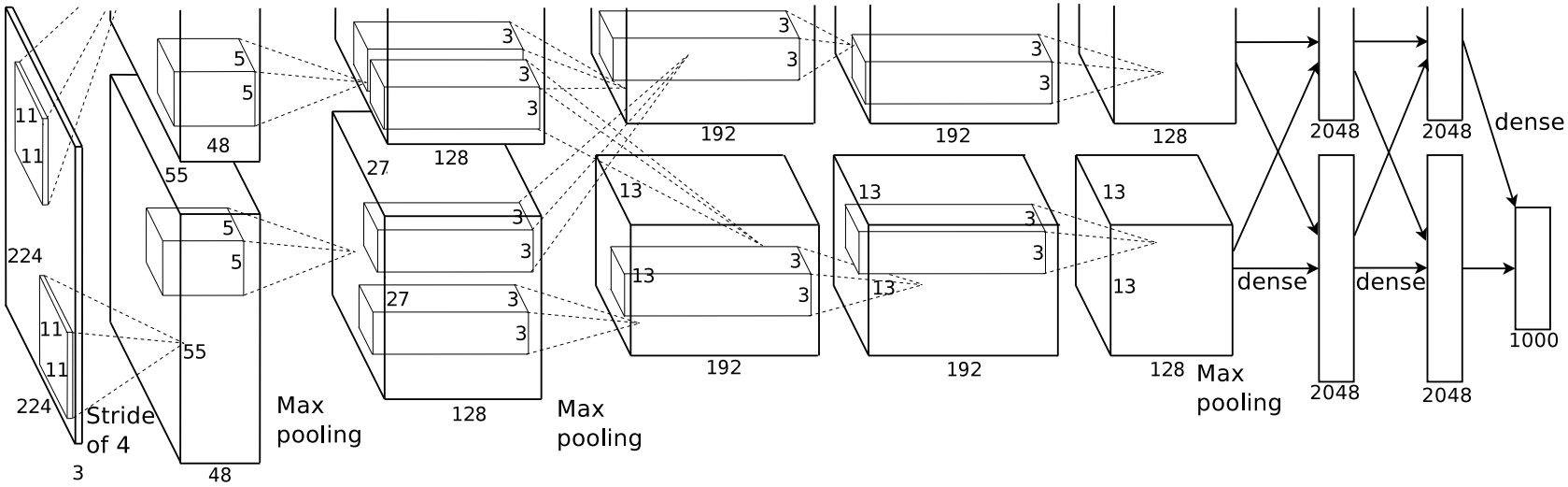


- 22k classes, 15M samples
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC): 1000 classes, 1.2M training images, 50k validation images, 150k test images

Russakovsky, Deng, Su, Krause, *et al.* 2014. Imagenet Large Scale Visual Recognition Challenge.

AlexNet

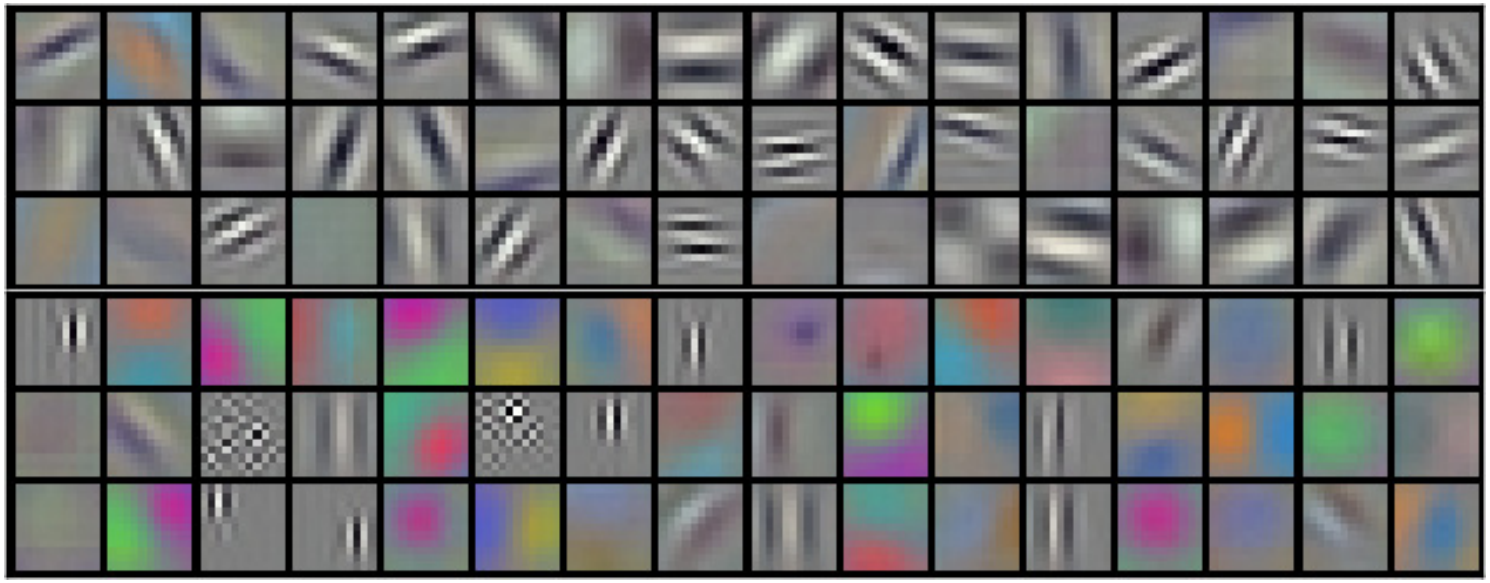
[Krizhevsky et al. 2012]



- implementation on two GPUs; connectivity between the two subnetworks is limited
- ReLU, data augmentation, local response normalization, dropout
- outperformed all previous models on ILSVRC by 10%

Krizhevsky, Sutskever, Hinton. NIPS 2012. Imagenet Classification with Deep Convolutional Neural Networks.

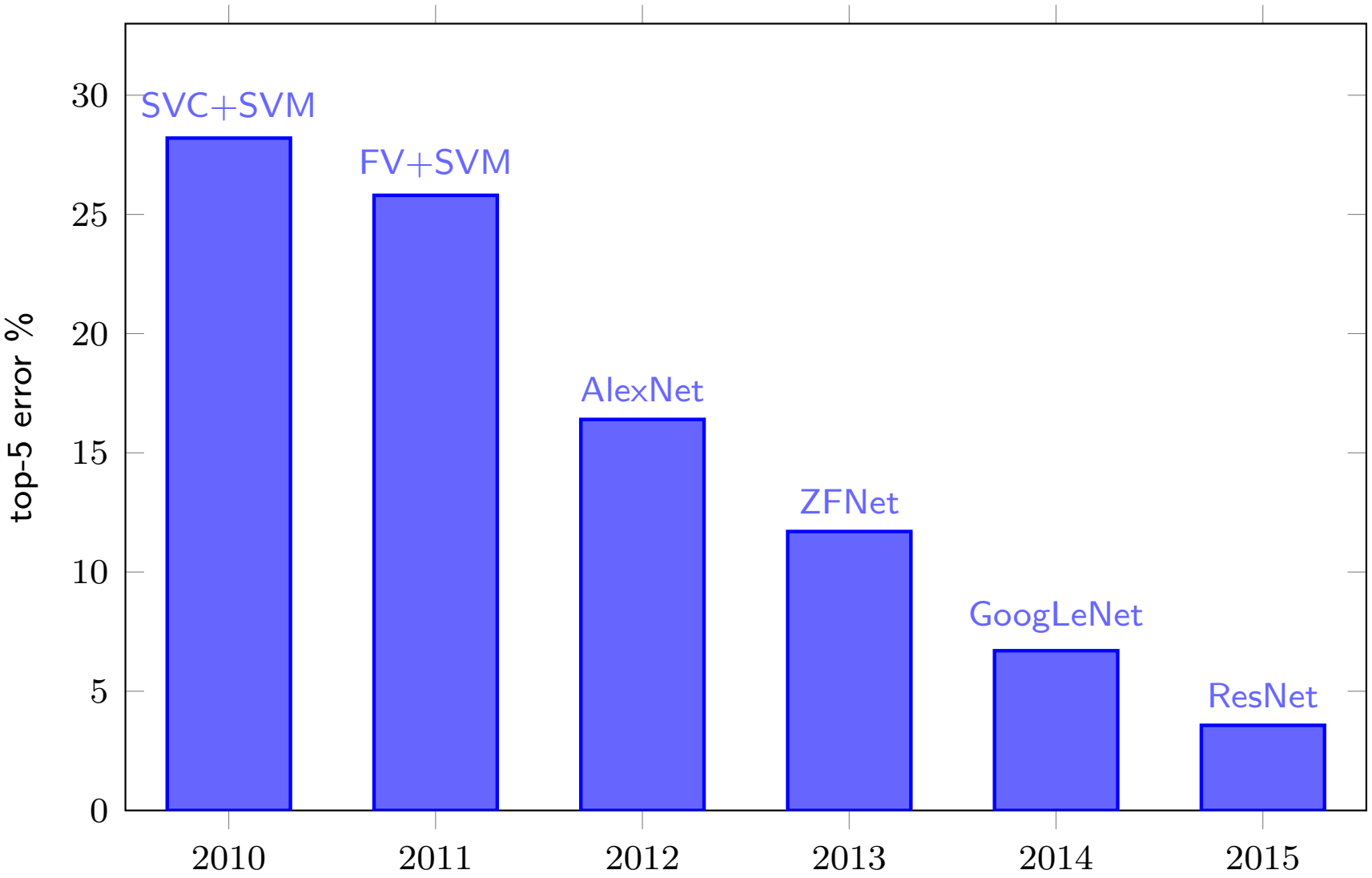
learned layer 1 kernels



- 96 kernels of size $11 \times 11 \times 3$
- top: 48 GPU 1 kernels; bottom: 48 GPU 2 kernels

Krizhevsky, Sutskever, Hinton. NIPS 2012. Imagenet Classification with Deep Convolutional Neural Networks.

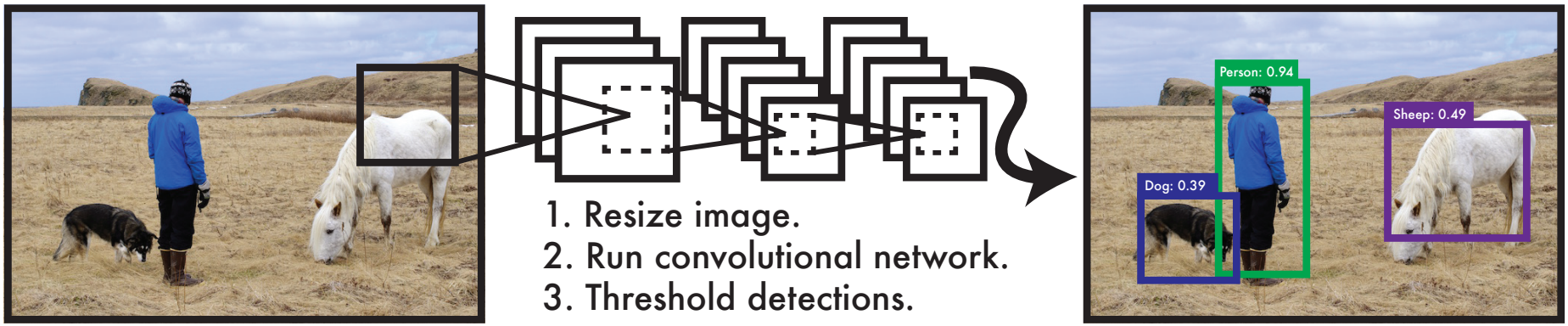
ImageNet classification performance



Russakovsky, Deng, Su, Krause, *et al*. 2014. Imagenet Large Scale Visual Recognition Challenge.

object detection

[Redmon et al. 2016]

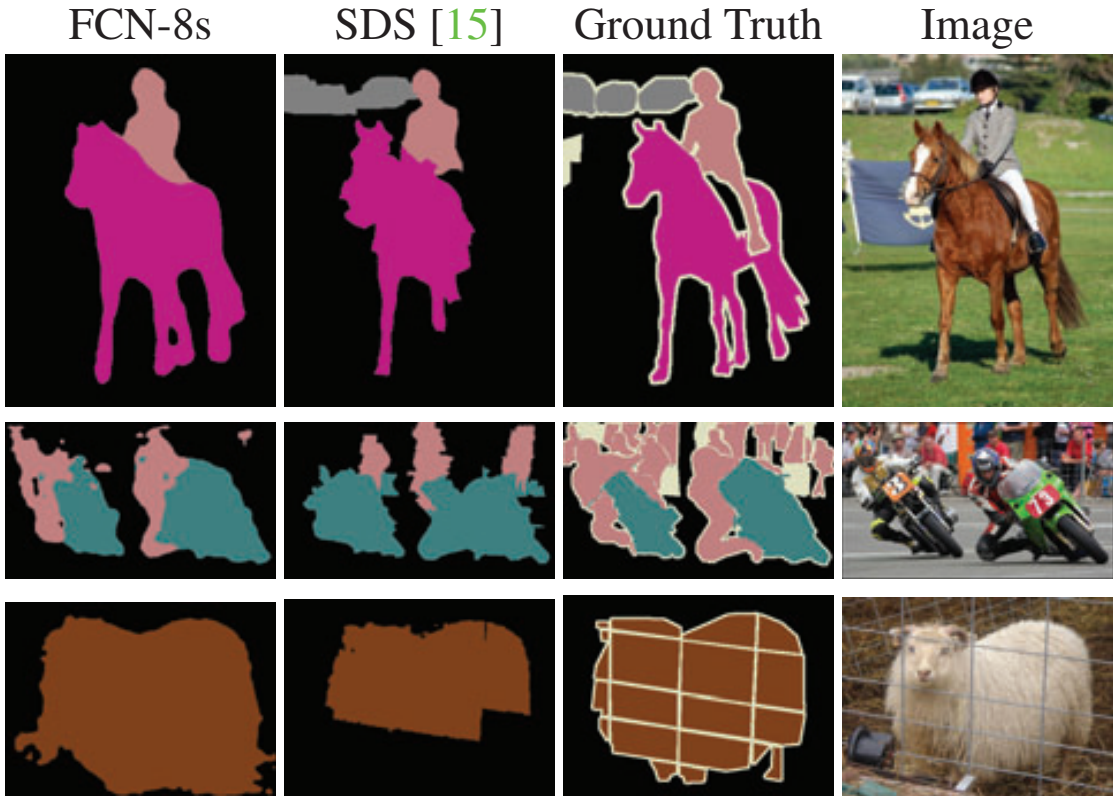


- learn to detect objects as a single classification and regression task, without scanning the image or detecting candidate regions
- first object detector to operate at 45fps

Redmon, Divvala, Girshick, Farhadi. CVPR 2016. You Only Look Once: Unified, Real-Time Object Detection.

semantic segmentation

[Long et al. 2015]



- learn to upsample
- apply to pixel-dense prediction tasks

instance segmentation and pose estimation

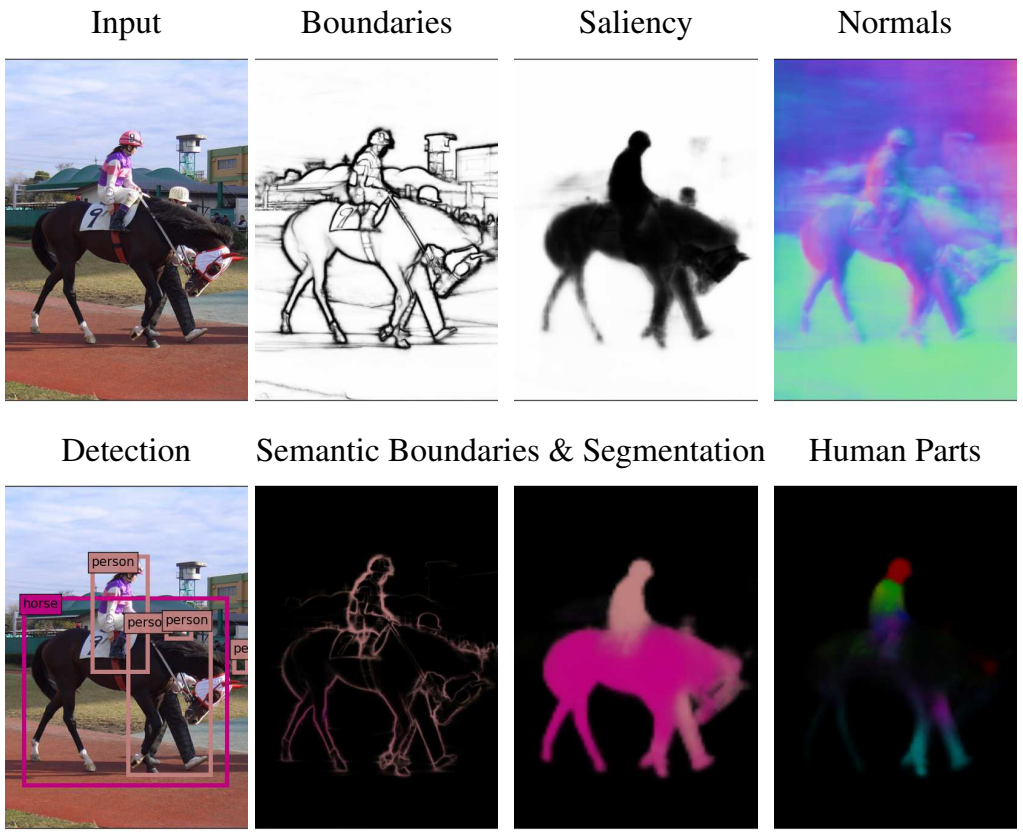
[He et al. 2017]

- semantic segmentation per detected region
- pose estimation as regression

He, Gkioxari, Dollar, Girshick. ICCV 2017. Mask R-CNN.

multi-task learning

[Kokkinos 2017]



- learn several vision tasks with a joint network architecture including task-specific skip layers

Kokkinos. CVPR 2017. Ubernet: Training a Universal Convolutional Neural Network for Low-, Mid-, and High-Level Vision Using Diverse Datasets and Limited Memory.

geometric matching

[Rocco et al. 2017]

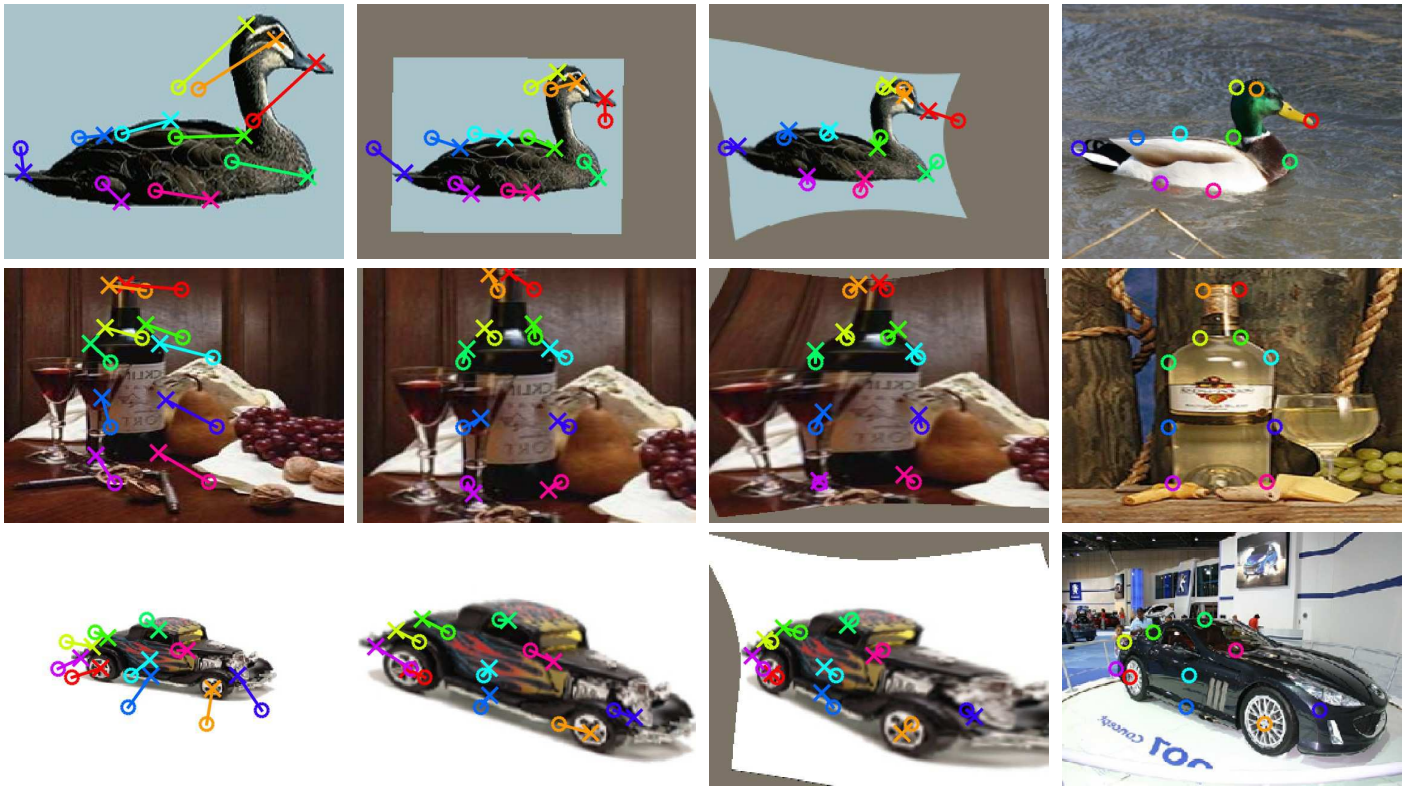


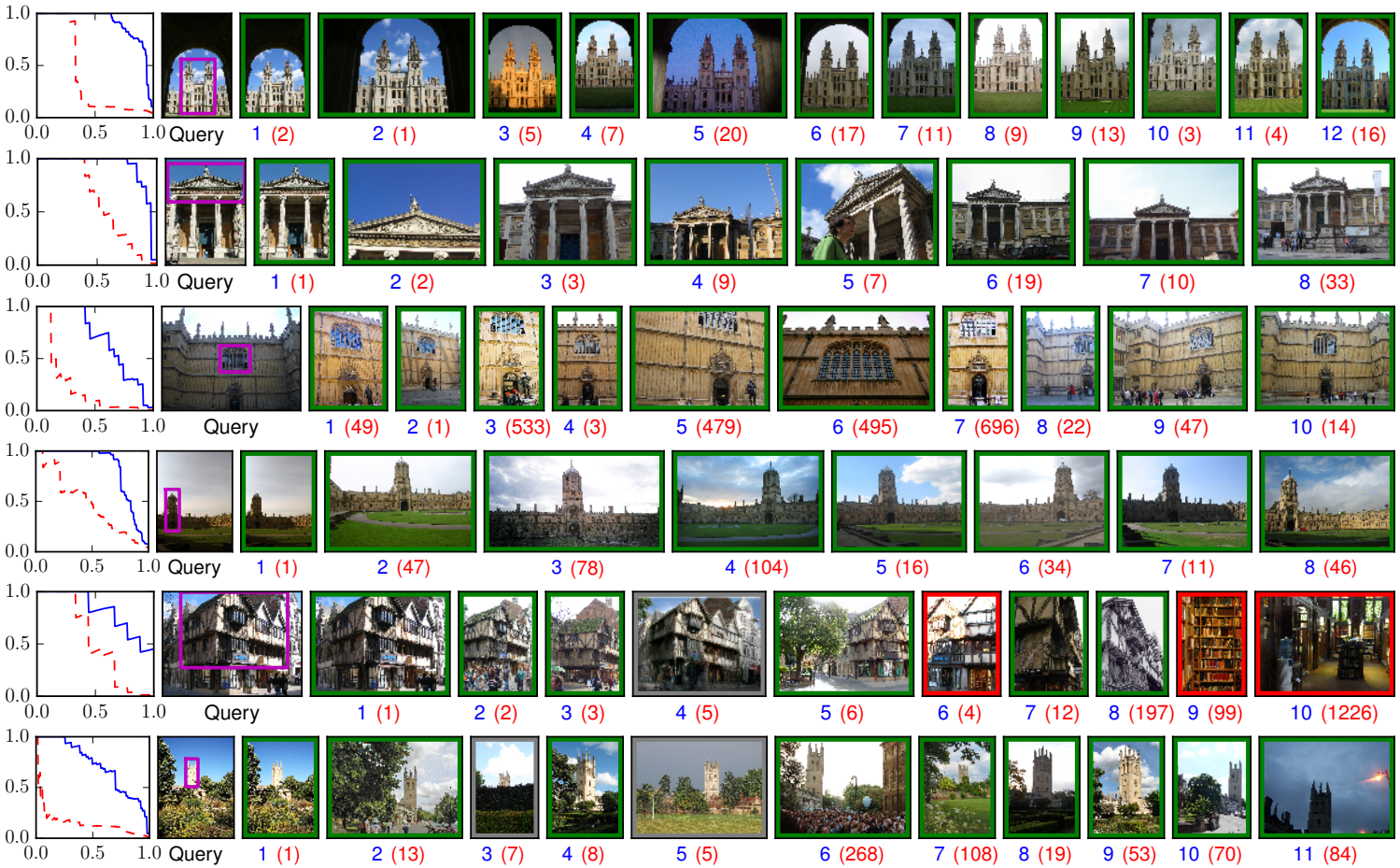
Image A | Aligned A (affine) | Aligned A (affine+TPS) | Image B

- mimic the standard steps of feature extraction, matching and simultaneous inlier detection and model parameter estimation
- still trainable end-to-end

Rocco, Arandjelovic, Sivic. CVPR 2017. Convolutional Neural Network Architecture for Geometric Matching.

image retrieval

[Gordo et al. 2016]

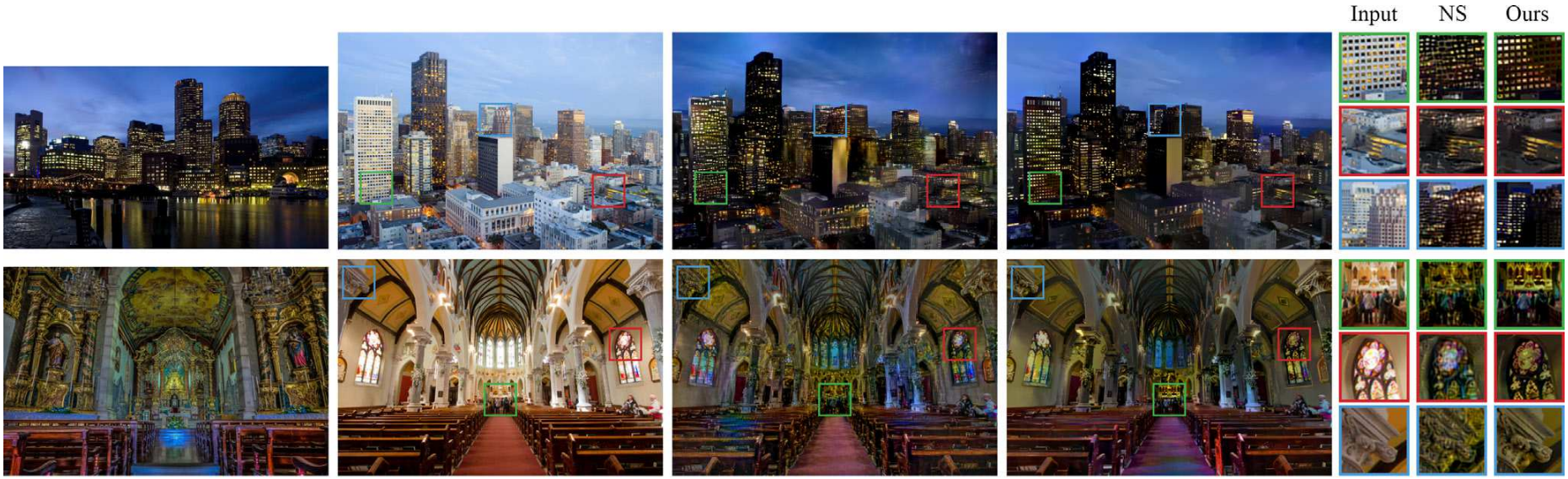


- learn to match
- apply as generic feature extractor

Gordo, Almazan, Revaud, Larlus. ECCV 2016. Deep Image Retrieval: Learning Global Representations for Image Search.

photorealistic style transfer

[Luan et al. 2017]



(a) Reference style image (b) Input image (c) Neural Style (distortions) (d) Our result (e) Insets

- generate same scene as input image
- transfer style from reference image
- photorealism regularization

Luan, Paris, Shechtman, Bala. CVPR 2017. Deep Photo Style Transfer.

image captioning

[Vinyals et al. 2017]

- image description by deep CNN
- language generation by RNN

Vinyals, Toshev, Bengio and Erhan. PAMI 2017. Show and Tell: Lessons Learned From the 2015 MSCOCO Image Captioning Challenge.

image retrieval challenges

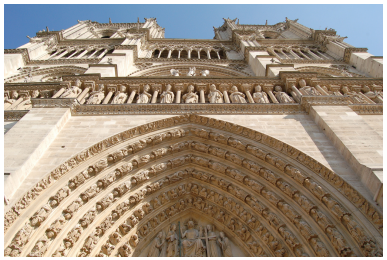
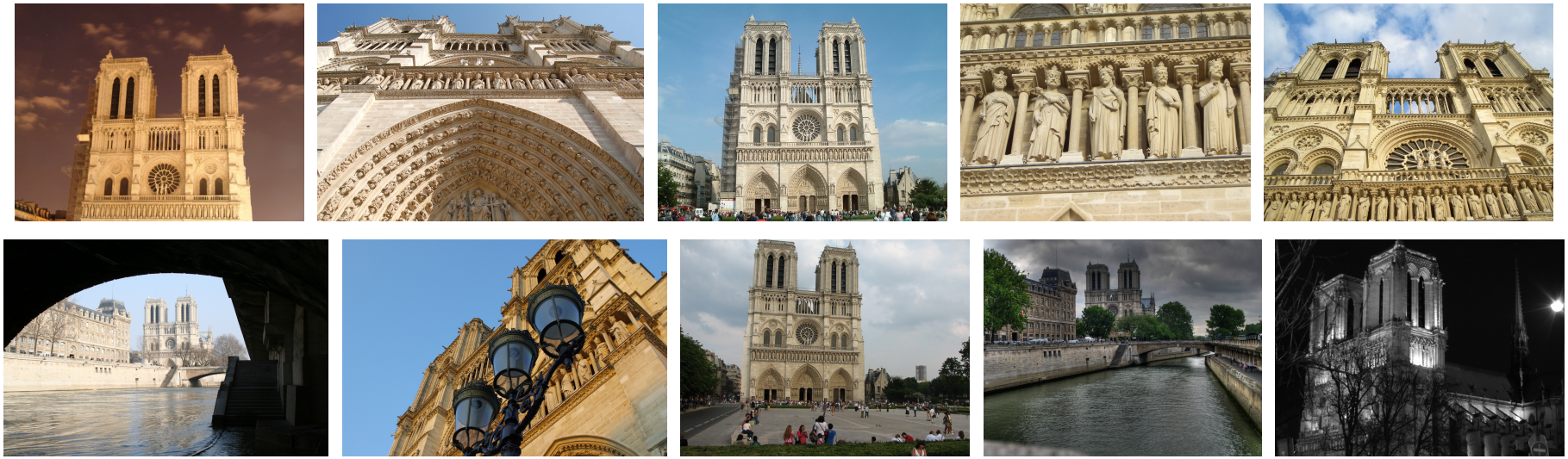









image retrieval challenges



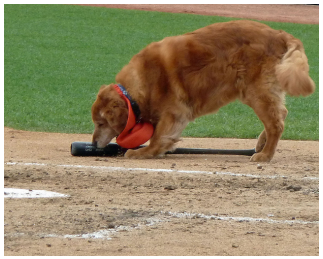
- scale
- viewpoint
- occlusion
- clutter
- lighting

- distinctiveness
- distractors

image classification challenges







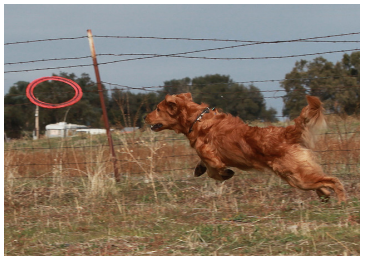


image classification challenges

- scale
- viewpoint
- occlusion
- clutter
- lighting

- number of instances
- texture/color
- pose
- deformability
- intra-class variability

Convolution

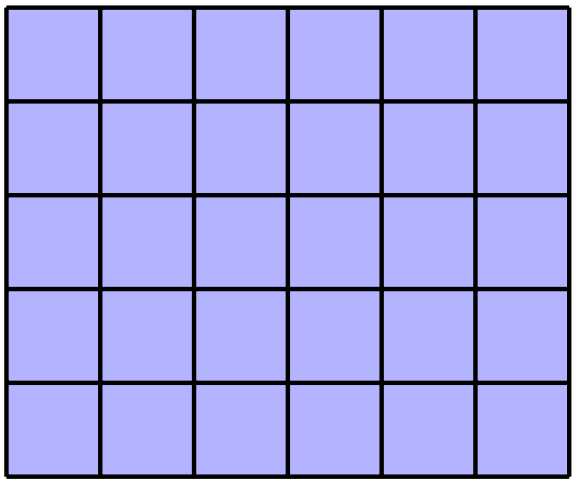
2d convolution

1	2	3
4	5	6
7	8	9

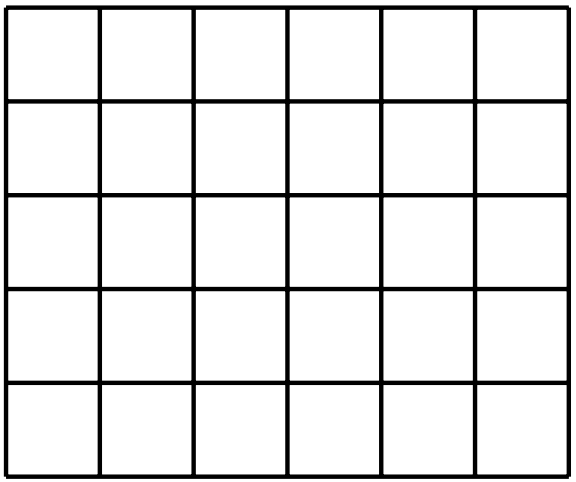
h

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} x[\mathbf{k}]h[\mathbf{n} - \mathbf{k}]$$

$$= \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$



x



$x * h$

2d convolution

1	2	3
4	5	6
7	8	9

h

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} x[\mathbf{k}]h[\mathbf{n} - \mathbf{k}]$$

$$= \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$

9	8	7			
6	5	4			
3	2	1			

x

$x * h$

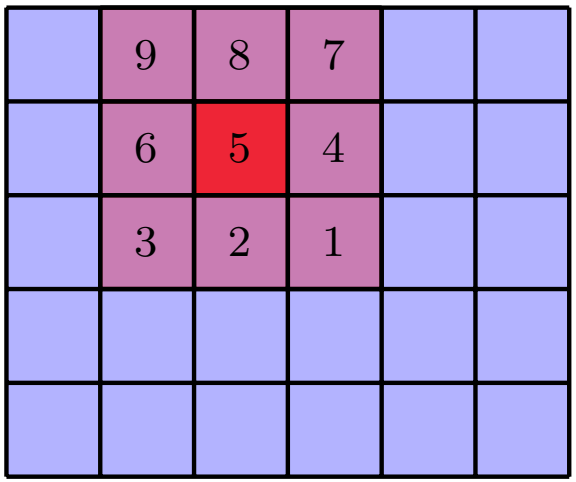
2d convolution

1	2	3
4	5	6
7	8	9

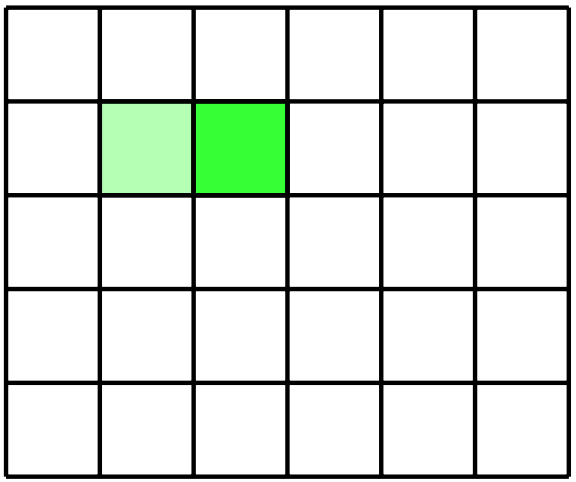
h

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} x[\mathbf{k}]h[\mathbf{n} - \mathbf{k}]$$

$$= \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$



x



$x * h$

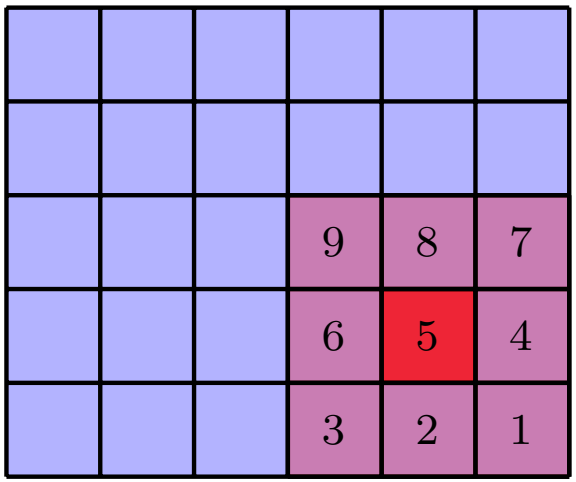
2d convolution

1	2	3
4	5	6
7	8	9

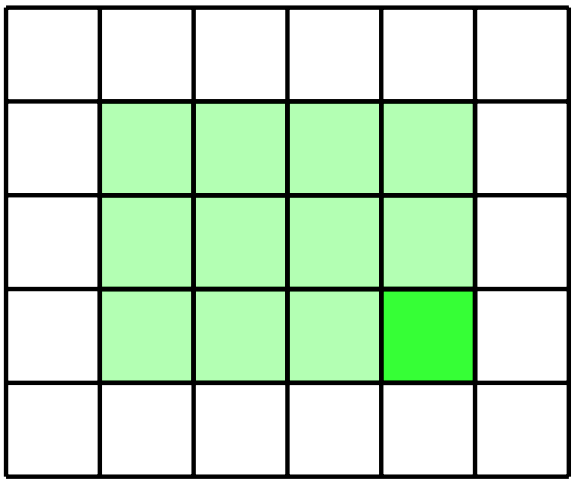
h

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} x[\mathbf{k}]h[\mathbf{n} - \mathbf{k}]$$

$$= \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$

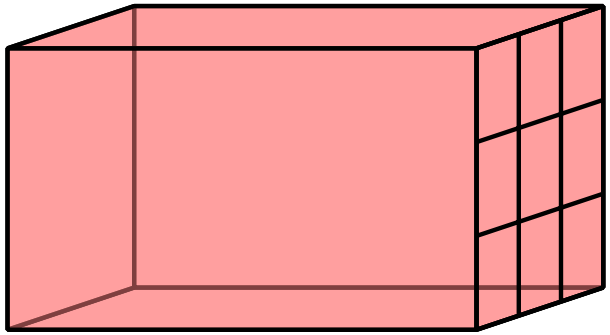


x



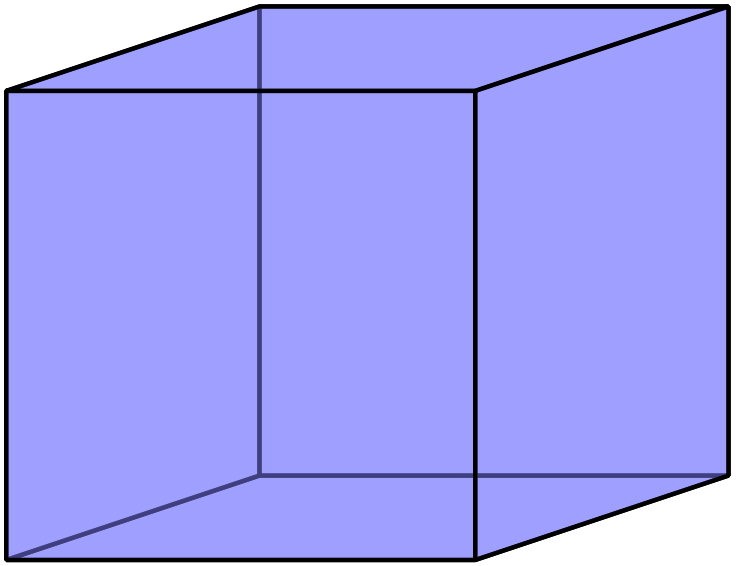
$x * h$

convolution on feature maps

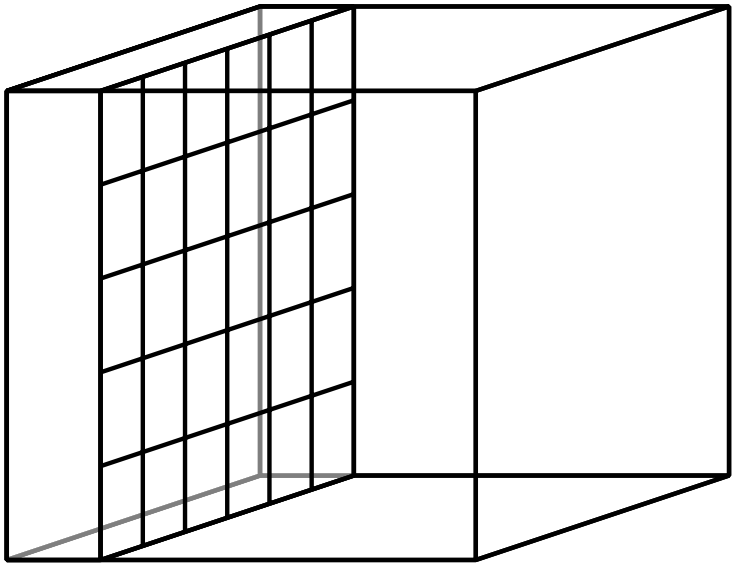


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

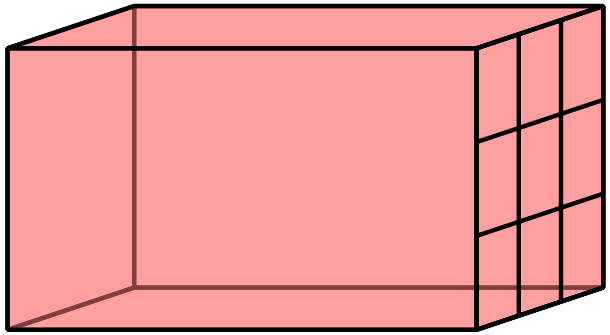


input $\mathbf{x}$



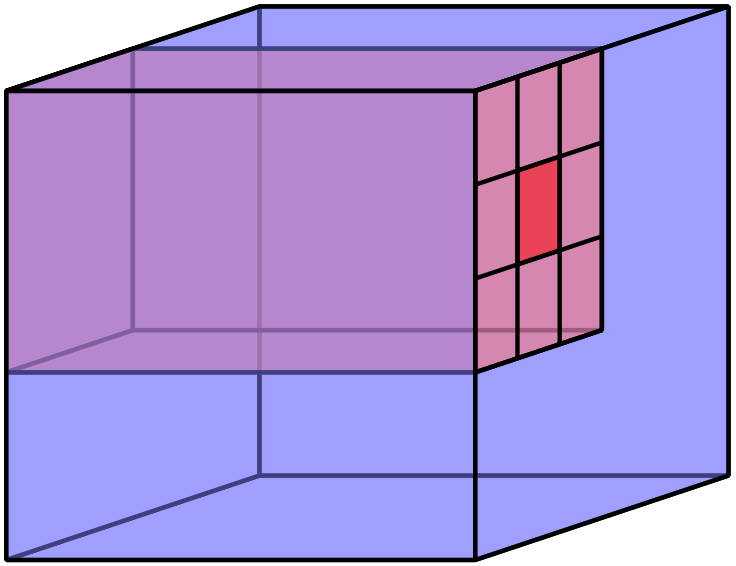
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

convolution on feature maps

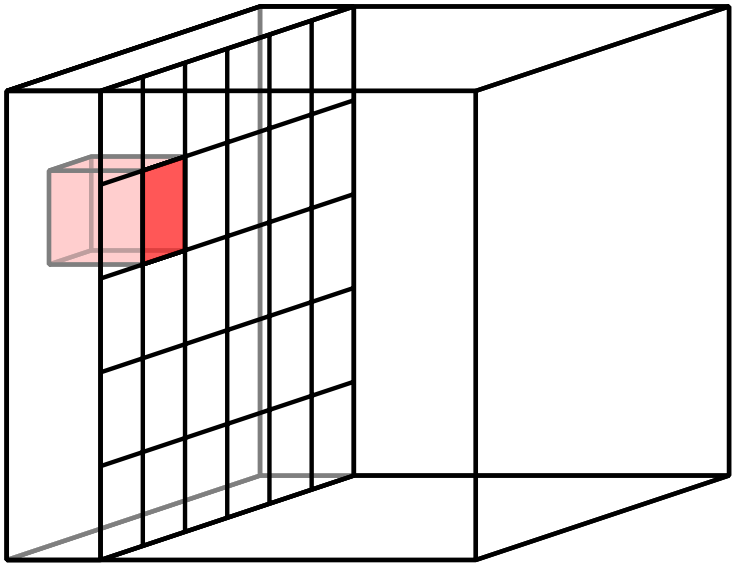


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

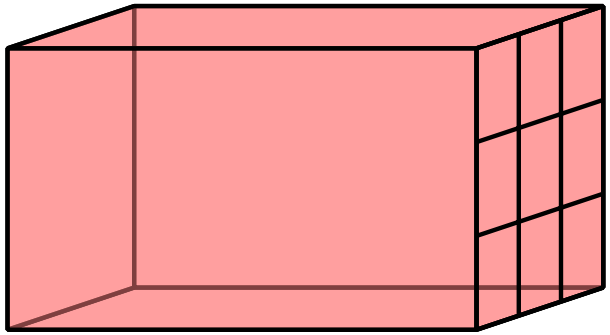


input $\mathbf{x}$



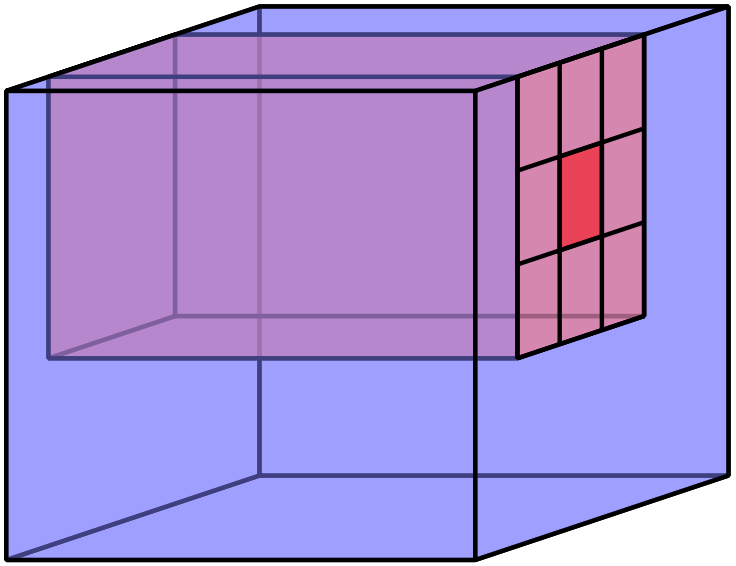
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

convolution on feature maps

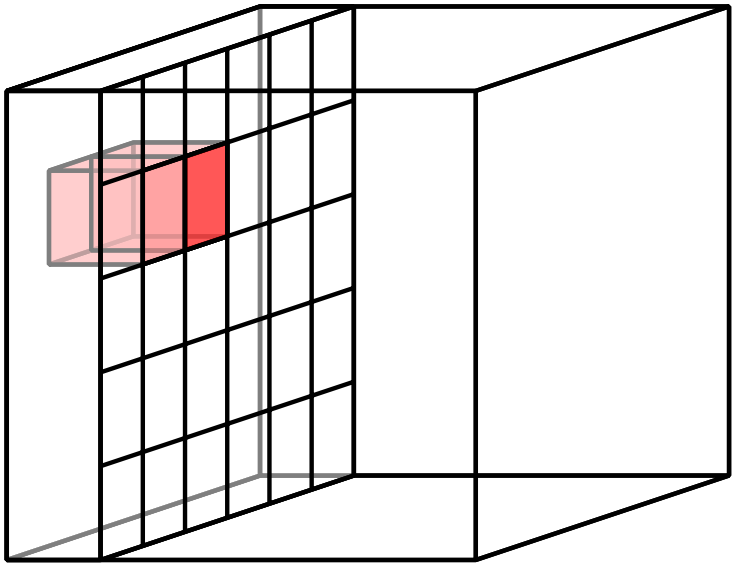


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

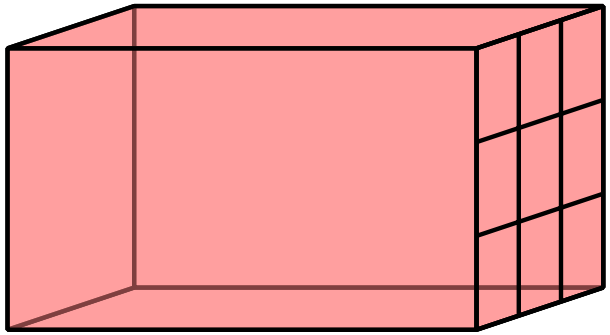


input $\mathbf{x}$



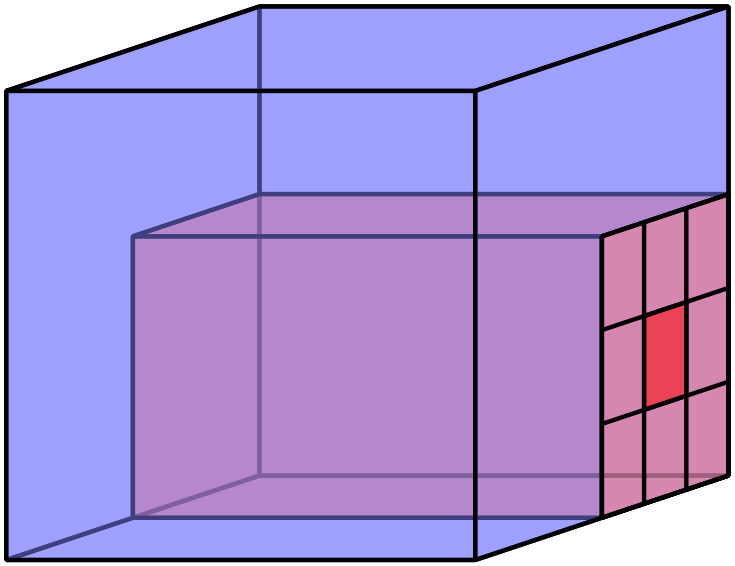
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

convolution on feature maps

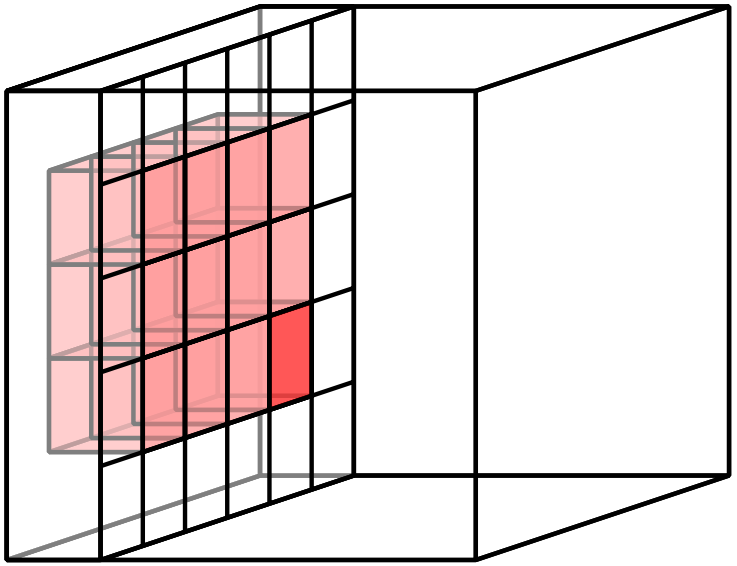


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

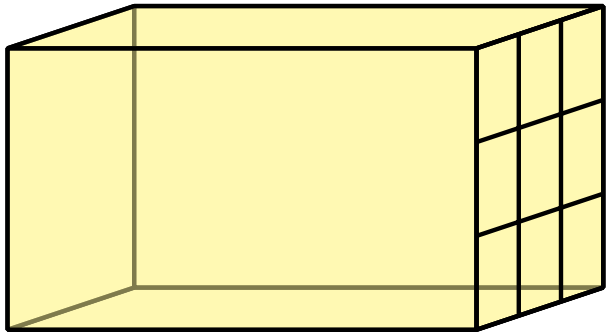


input $\mathbf{x}$



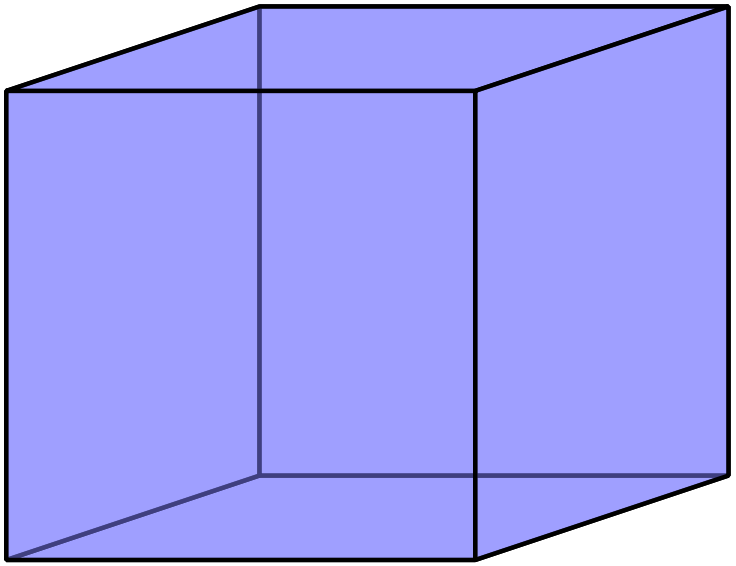
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

convolution on feature maps

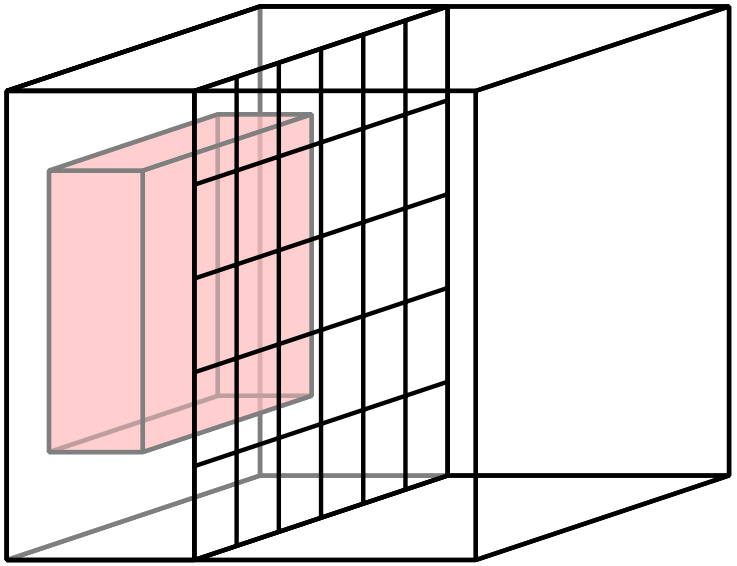


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

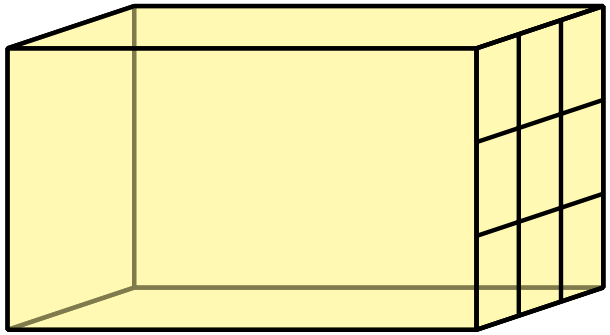


input $\mathbf{x}$



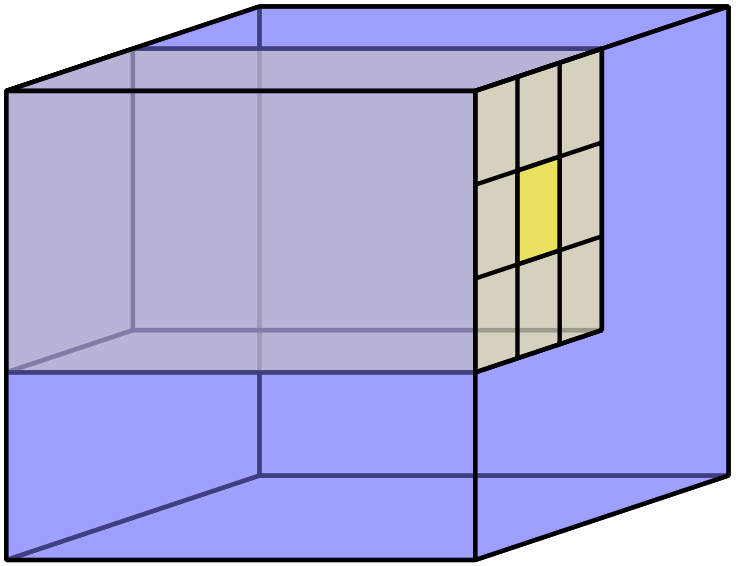
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

convolution on feature maps

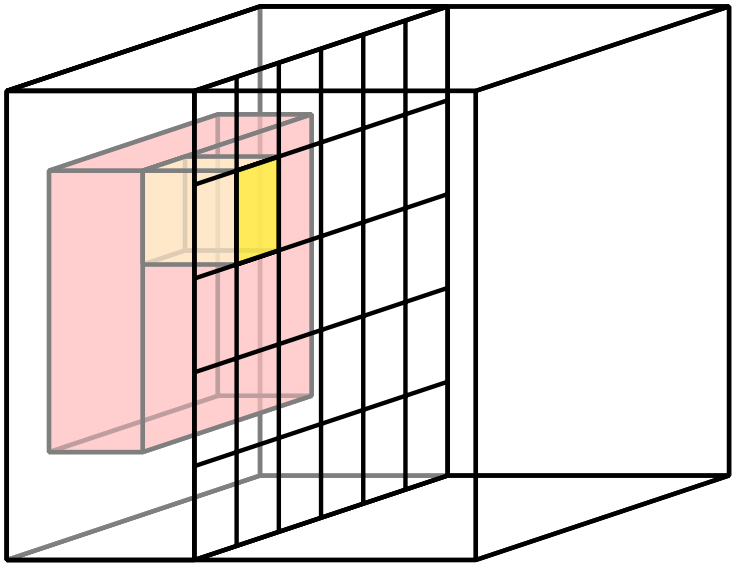


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

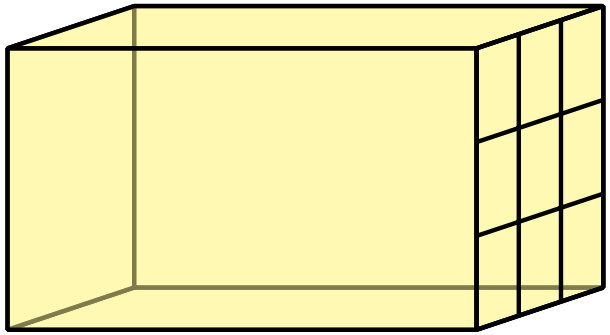


input $\mathbf{x}$



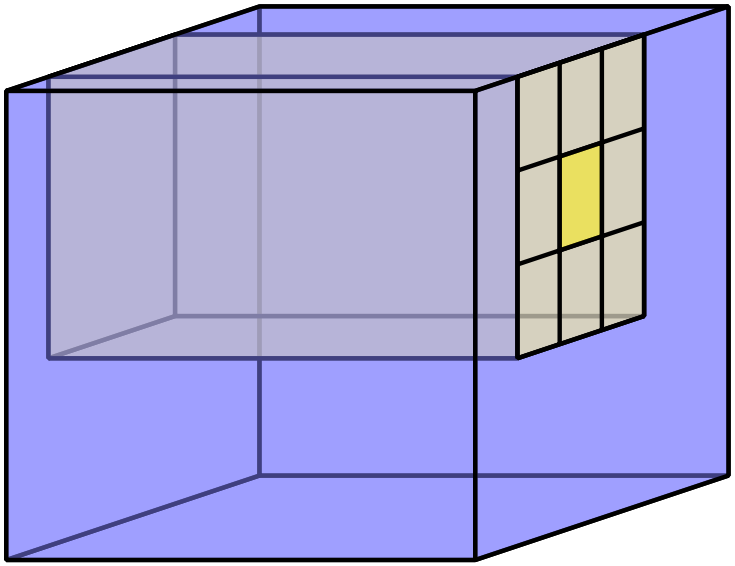
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

convolution on feature maps

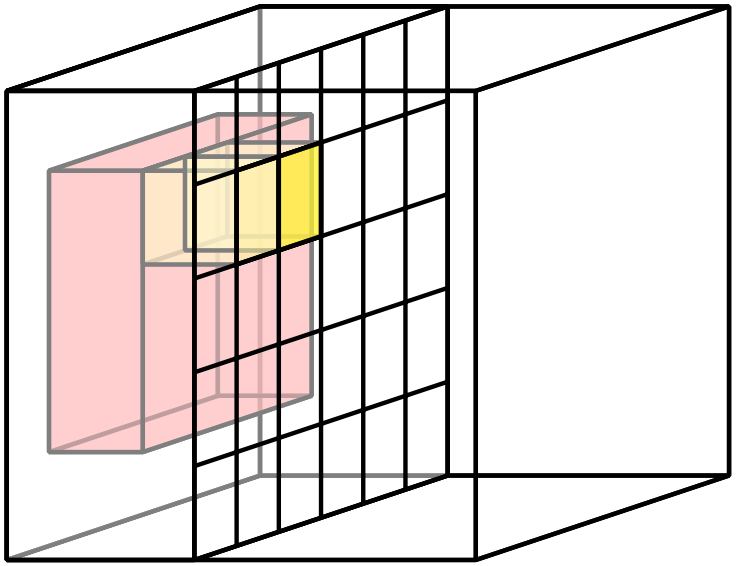


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

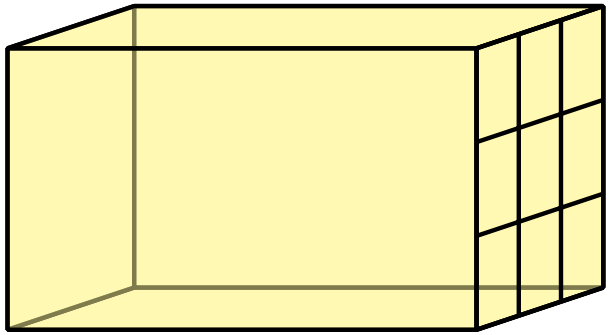


input $\mathbf{x}$



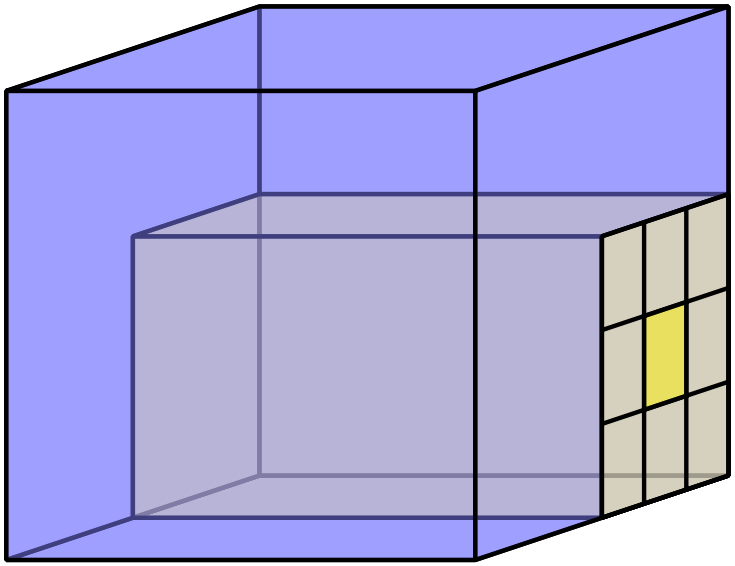
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

convolution on feature maps

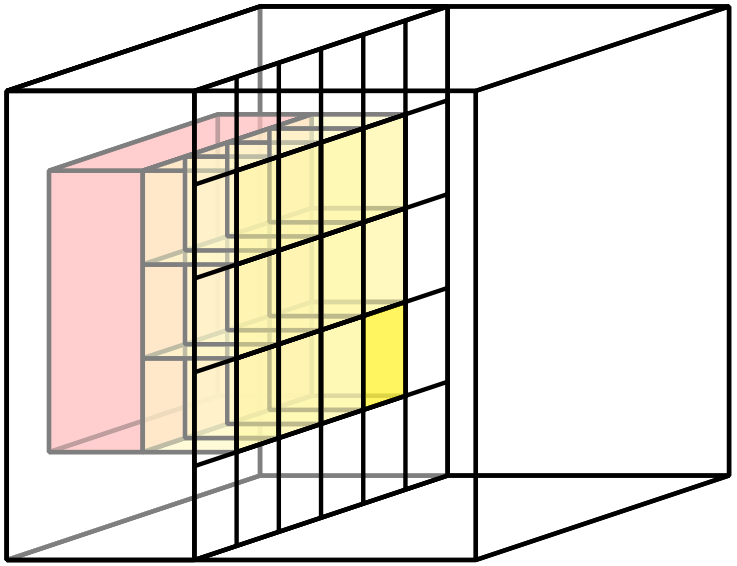


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

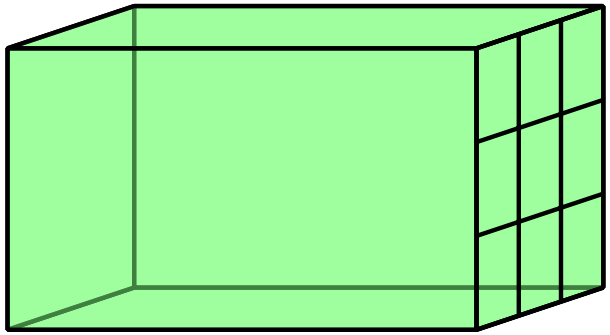


input $\mathbf{x}$



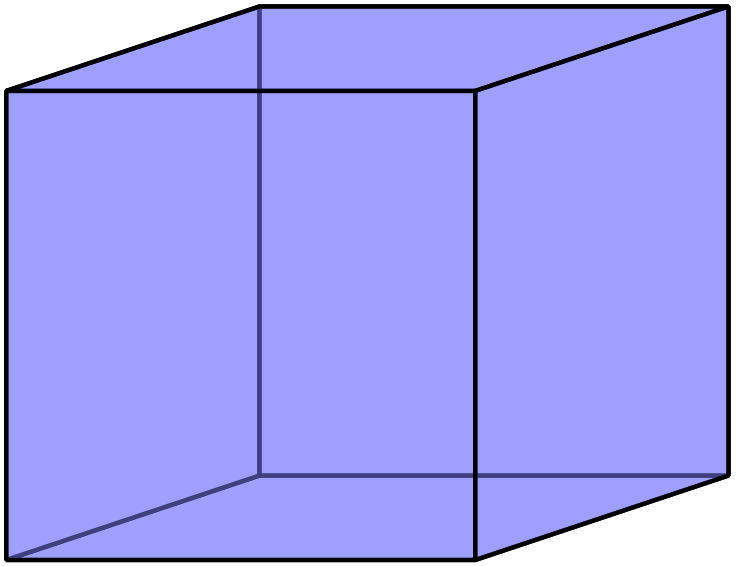
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

convolution on feature maps

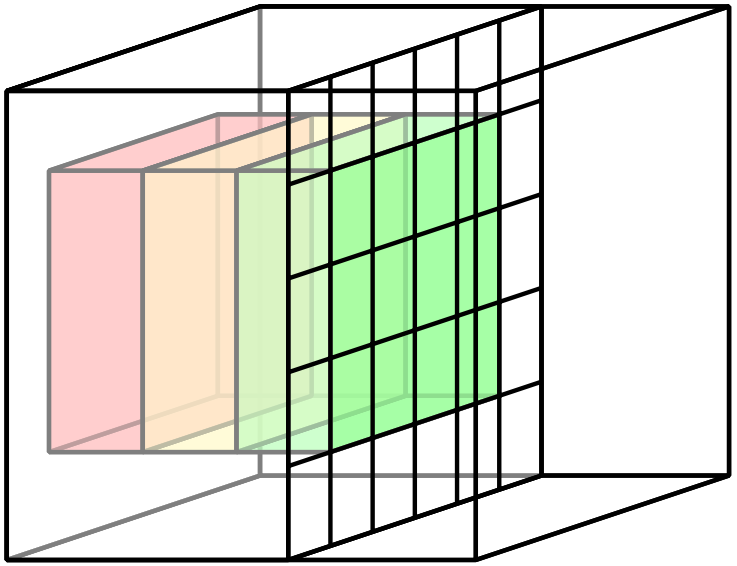


kernel $\mathbf{w}_3$

different kernel for
each output dimension



input $\mathbf{x}$

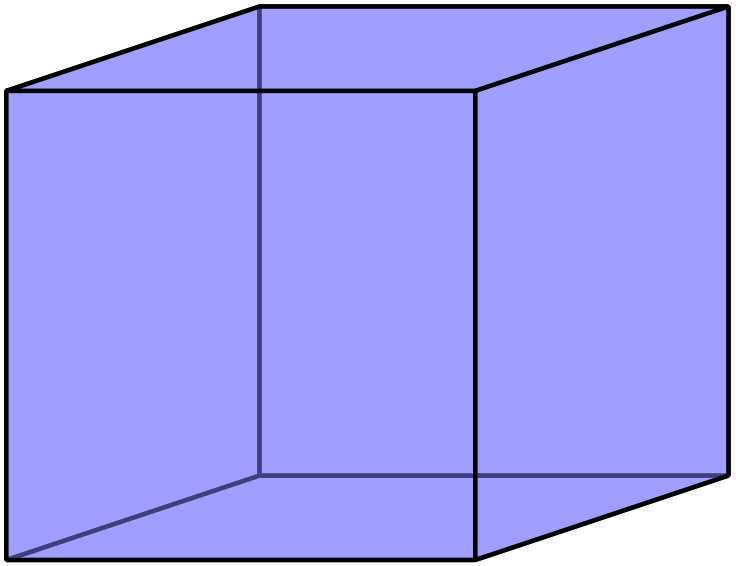


output $y_3 = h(\mathbf{w}_3^\top \star \mathbf{x} + b_3)$

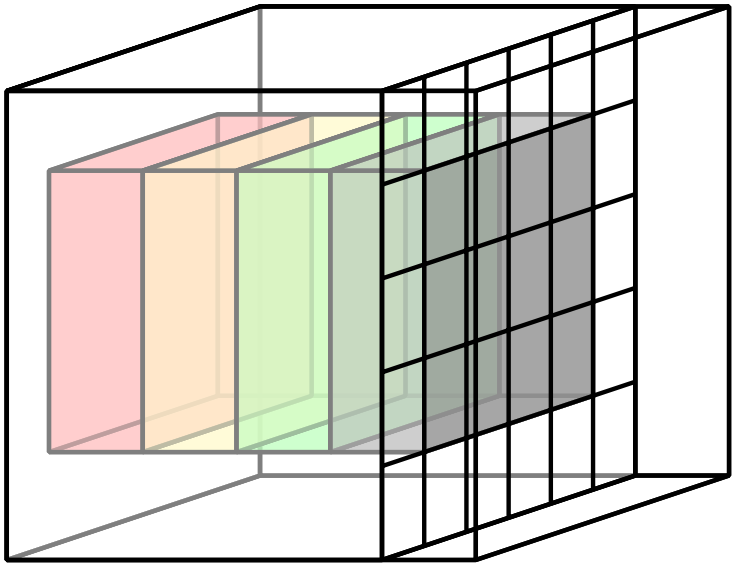
convolution on feature maps

kernel $\mathbf{w}_4$

different kernel for
each output dimension

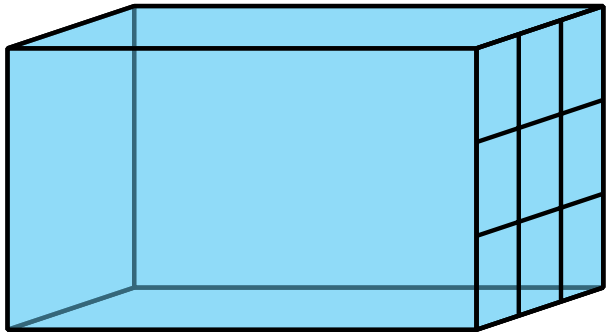


input $\mathbf{x}$



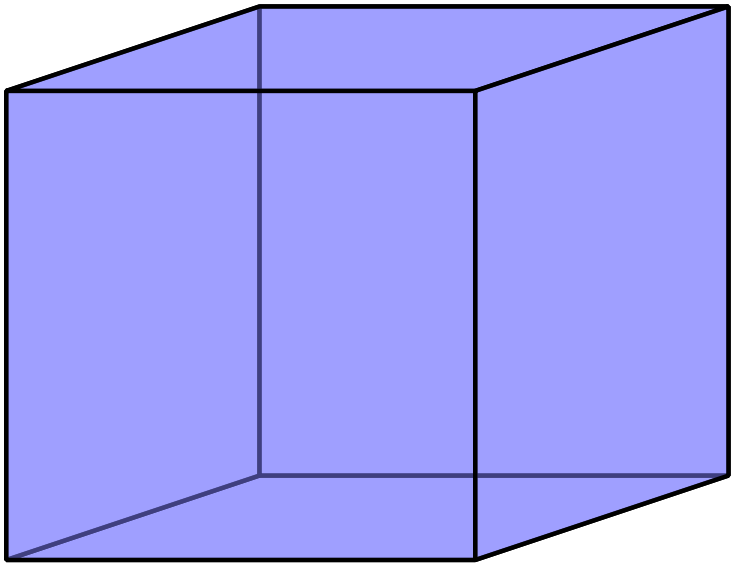
output $y_4 = h(\mathbf{w}_4^\top \star \mathbf{x} + b_4)$

convolution on feature maps

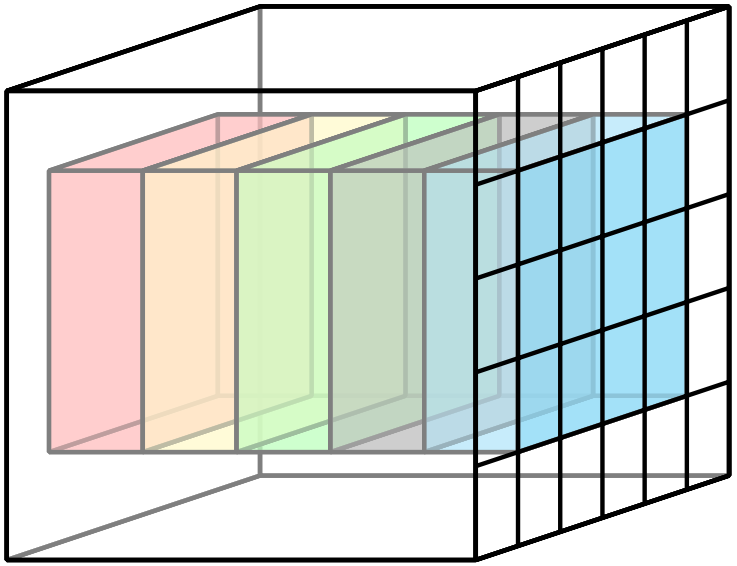


kernel $\mathbf{w}_5$

different kernel for
each output dimension

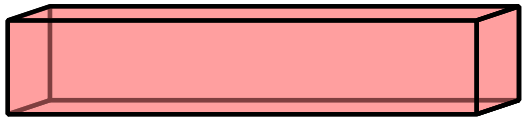


input $\mathbf{x}$



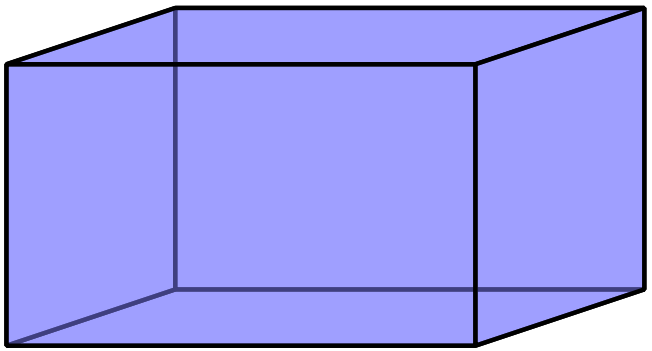
output $y_5 = h(\mathbf{w}_5^\top \star \mathbf{x} + b_5)$

1×1 convolution

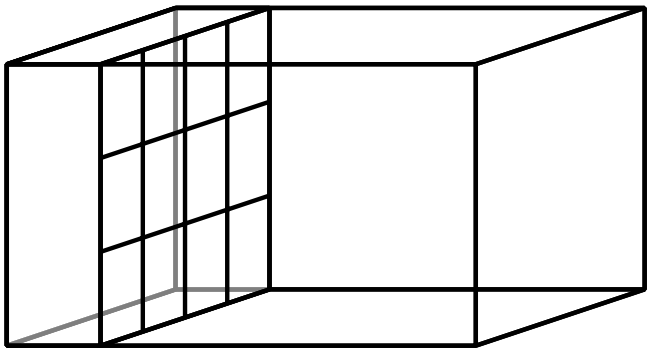


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

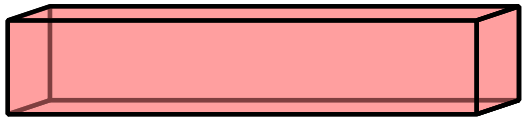


input $\mathbf{x}$



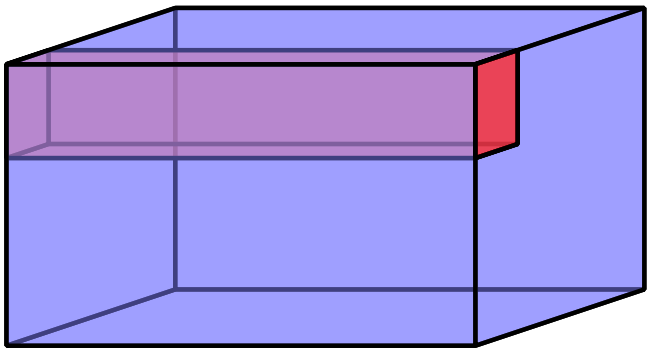
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

1×1 **convolution**

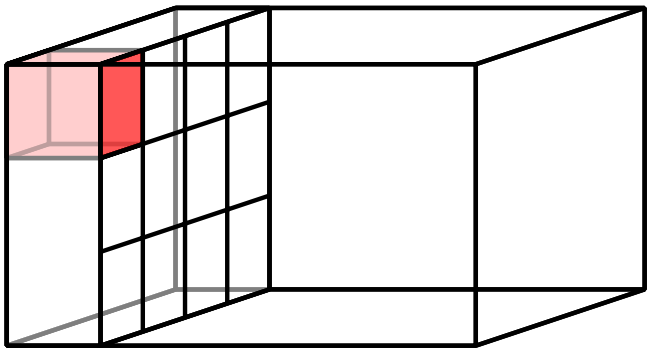


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

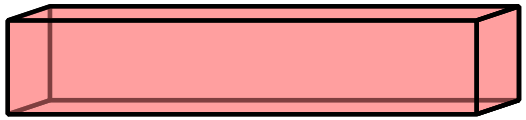


input $\mathbf{x}$



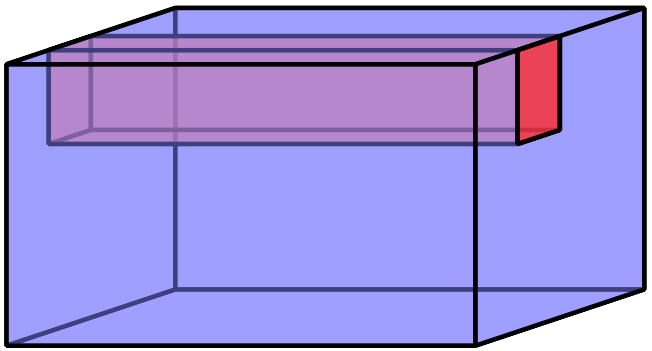
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

1×1 convolution

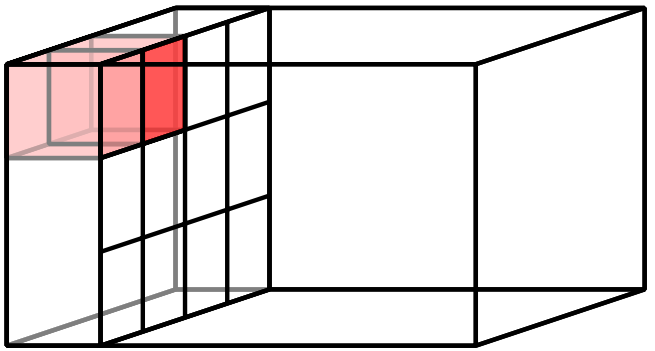


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

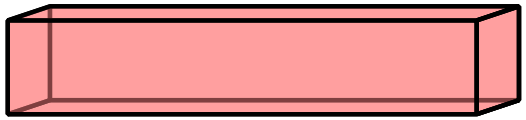


input $\mathbf{x}$



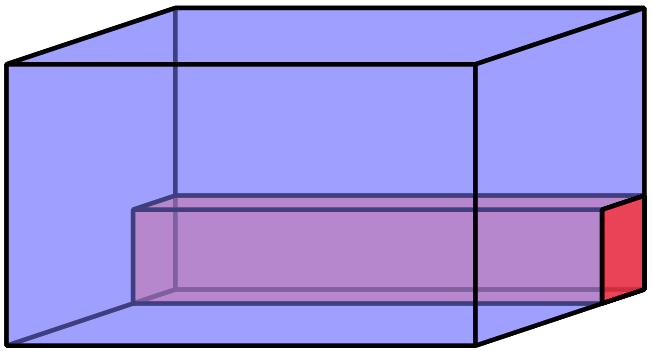
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

1×1 convolution

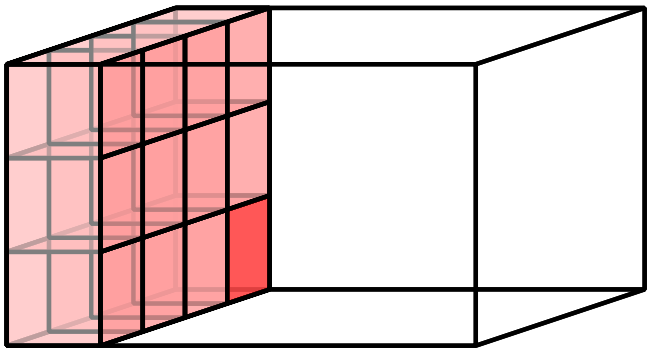


kernel $\mathbf{w}_1$

kernel weights shared
among all spatial positions

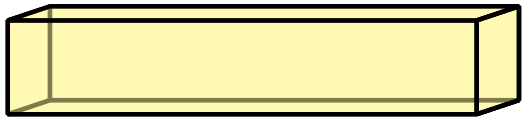


input $\mathbf{x}$



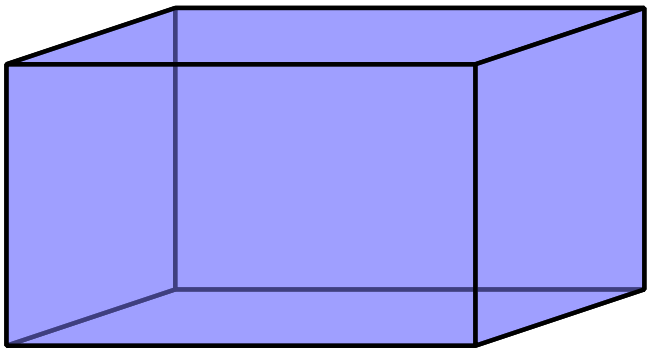
output $y_1 = h(\mathbf{w}_1^\top \star \mathbf{x} + b_1)$

1×1 convolution

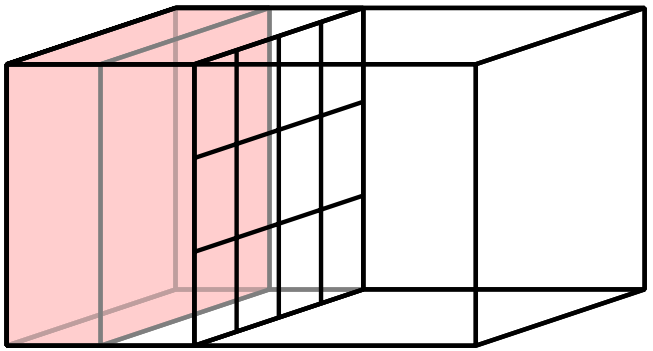


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

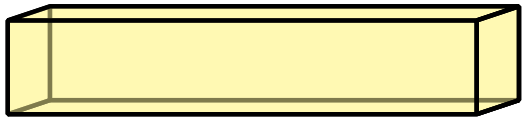


input $\mathbf{x}$



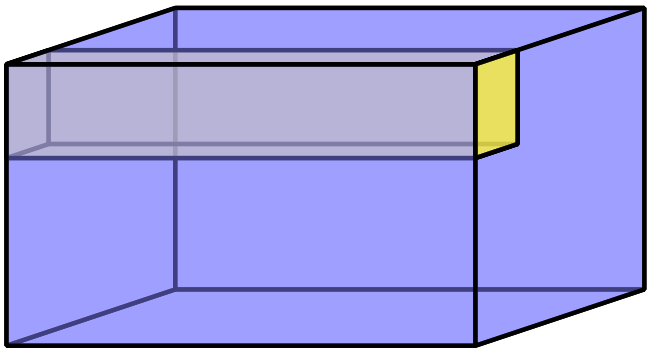
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

1×1 convolution

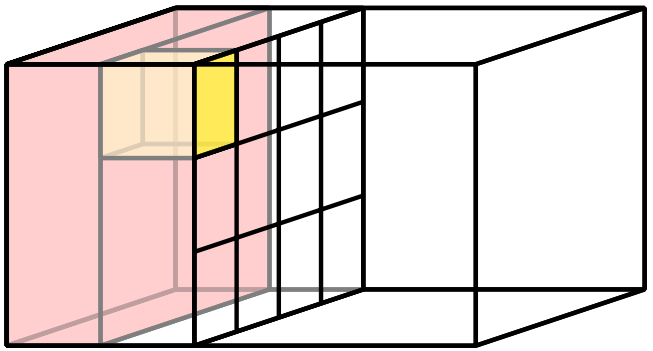


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

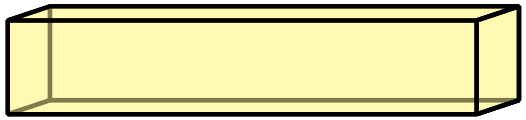


input $\mathbf{x}$



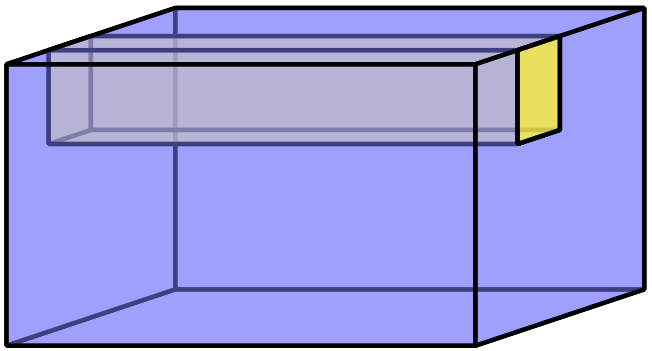
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

1×1 convolution

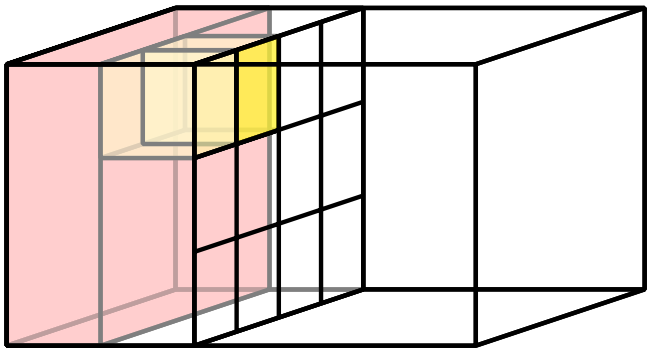


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

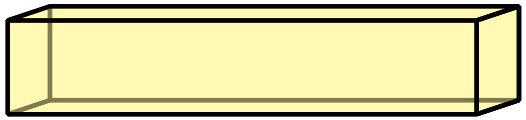


input $\mathbf{x}$



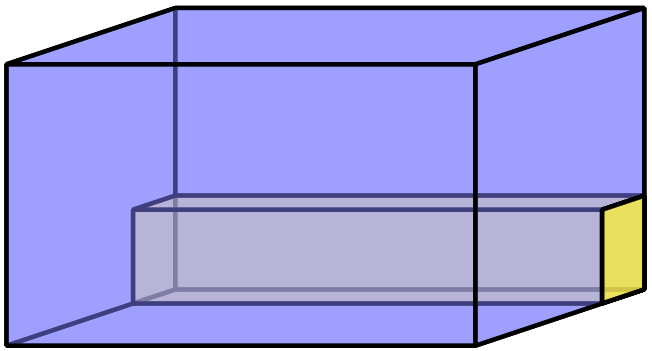
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

1×1 convolution

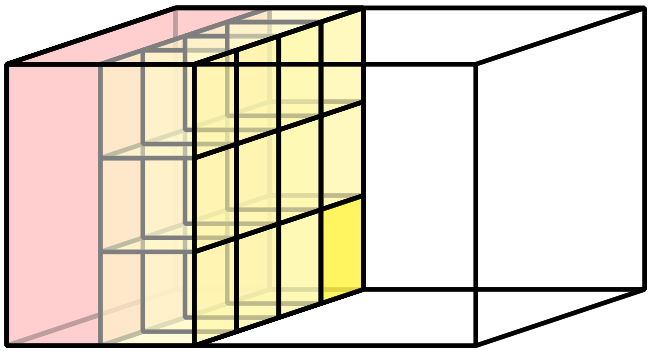


kernel $\mathbf{w}_2$

new kernel, but still shared among all spatial positions

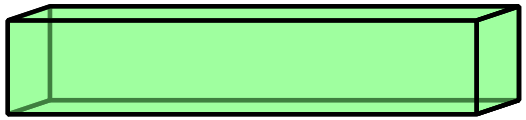


input $\mathbf{x}$



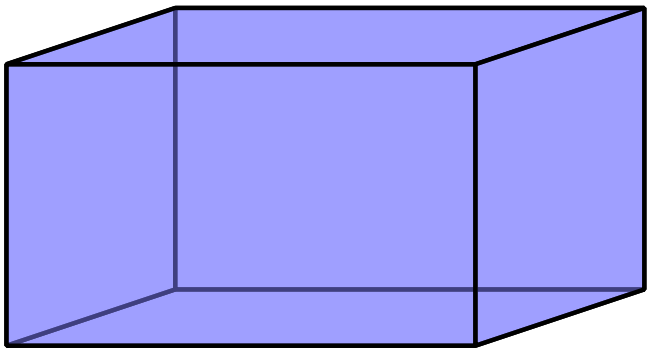
output $y_2 = h(\mathbf{w}_2^\top \star \mathbf{x} + b_2)$

1×1 **convolution**

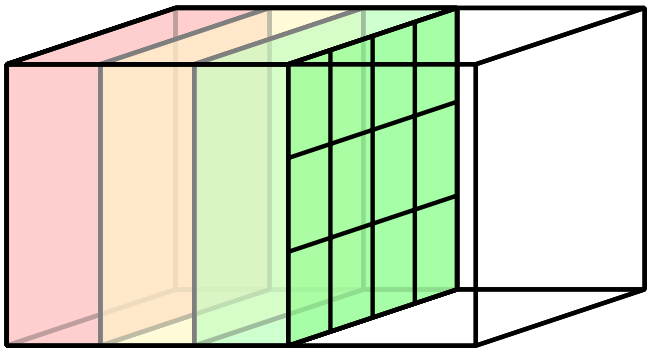


kernel $\mathbf{w}_3$

different kernel for
each output dimension



input $\mathbf{x}$

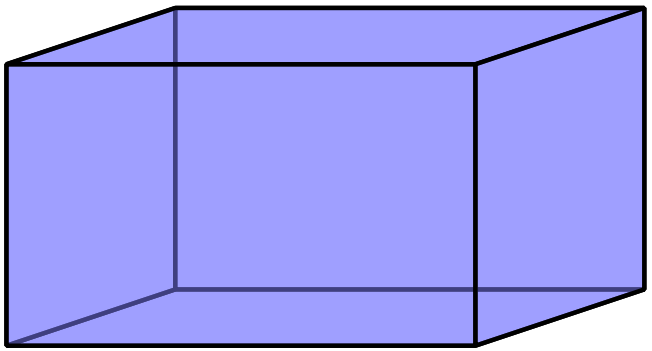


output $y_3 = h(\mathbf{w}_3^\top \star \mathbf{x} + b_3)$

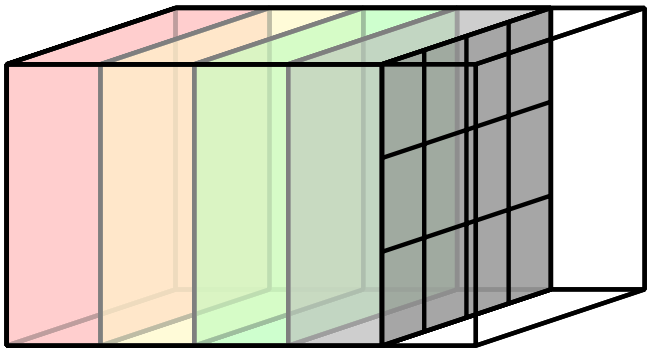
1×1 convolution

kernel $\mathbf{w}_4$

different kernel for
each output dimension



input $\mathbf{x}$

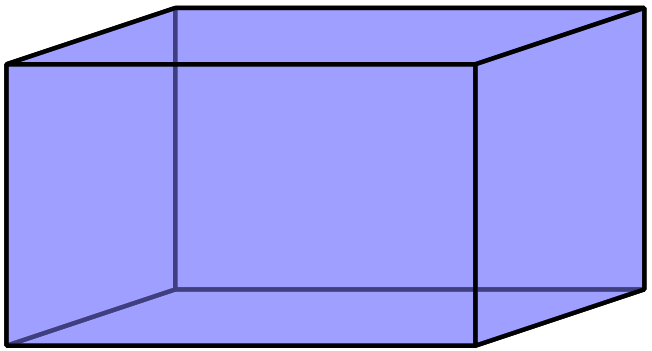


output $y_4 = h(\mathbf{w}_4^\top \star \mathbf{x} + b_4)$

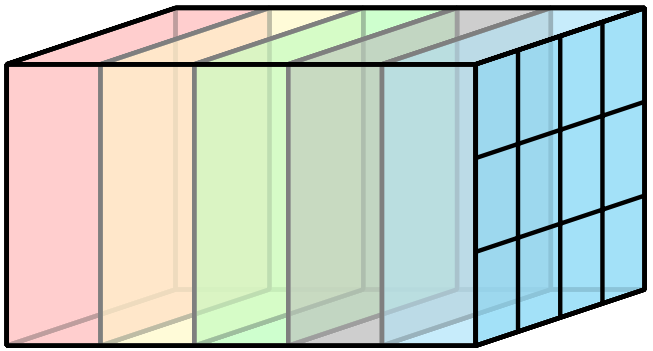
1×1 convolution

kernel $\mathbf{w}_5$

different kernel for
each output dimension



input $\mathbf{x}$



output $y_5 = h(\mathbf{w}_5^\top \star \mathbf{x} + b_5)$

network architectures

convolutional network

			MNIST			CIFAR10		
			param	ops	volume	param	ops	volume
$\mathbf{x}$ =	input		0	0	$28 \times 28 \times 1$	0	0	$32 \times 32 \times 3$
$\mathbf{z}_1$ =	conv$(5, 32)$	$(\mathbf{x})$	832	479232	$24 \times 24 \times 32$	2432	1906688	$28 \times 28 \times 32$
$\mathbf{p}_1$ =	pool(2)	$(\mathbf{z}_1)$	0	18432	$12 \times 12 \times 32$	0	25088	$14 \times 14 \times 32$
$\mathbf{z}_2$ =	conv$(5, 64)$	$(\mathbf{p}_1)$	51264	3280896	$8 \times 8 \times 64$	51264	5126400	$10 \times 10 \times 64$
$\mathbf{p}_2$ =	pool(2)	$(\mathbf{z}_2)$	0	4096	$4 \times 4 \times 64$	0	6400	$5 \times 5 \times 64$
$\mathbf{z}_3$ =	fc(100)	$(\mathbf{p}_2)$	102500	102500	100	160100	160100	100
$\mathbf{a}_4$ =	fc(10)	$(\mathbf{z}_3)$	1010	1010	10	1010	1010	10
$\mathbf{y}$ =	softmax	$(\mathbf{a}_4)$	0	0	10	0	0	10

- most parameters in first fully connected layer
- most operations in second convolutional layer
- most memory in first convolutional layer

conv$(r, k'[, p = 0][, s = 1])$; (max)- pool$(r[, s = r][, p = 0])$;

convolutional network

	MNIST			CIFAR10		
	param	ops	volume	param	ops	volume
input	0	0	$28 \times 28 \times 1$	0	0	$32 \times 32 \times 3$
conv$(5, 32)$	832	479232	$24 \times 24 \times 32$	2432	1906688	$28 \times 28 \times 32$
pool(2)	0	18432	$12 \times 12 \times 32$	0	25088	$14 \times 14 \times 32$
conv$(5, 64)$	51264	3280896	$8 \times 8 \times 64$	51264	5126400	$10 \times 10 \times 64$
pool(2)	0	4096	$4 \times 4 \times 64$	0	6400	$5 \times 5 \times 64$
fc(100)	102500	102500	100	160100	160100	100
fc(10)	1010	1010	10	1010	1010	10
softmax	0	0	10	0	0	10

- most parameters in first fully connected layer
- most operations in second convolutional layer
- most memory in first convolutional layer

conv$(r, k'[, p = 0][, s = 1])$; (max)- pool$(r[, s = r][, p = 0])$;

convolutional network

	MNIST			CIFAR10		
	param	ops	volume	param	ops	volume
input	0	0	$28 \times 28 \times 1$	0	0	$32 \times 32 \times 3$
conv$(5, 32)$	832	479232	$24 \times 24 \times 32$	2432	1906688	$28 \times 28 \times 32$
pool(2)	0	18432	$12 \times 12 \times 32$	0	25088	$14 \times 14 \times 32$
conv$(5, 64)$	51264	3280896	$8 \times 8 \times 64$	51264	5126400	$10 \times 10 \times 64$
pool(2)	0	4096	$4 \times 4 \times 64$	0	6400	$5 \times 5 \times 64$
fc(100)	102500	102500	100	160100	160100	100
fc(10)	1010	1010	10	1010	1010	10
softmax	0	0	10	0	0	10

- most parameters in first fully connected layer
- most operations in second convolutional layer
- most memory in first convolutional layer

conv$(r, k'[, p = 0][, s = 1])$; (max)- pool$(r[, s = r][, p = 0])$;

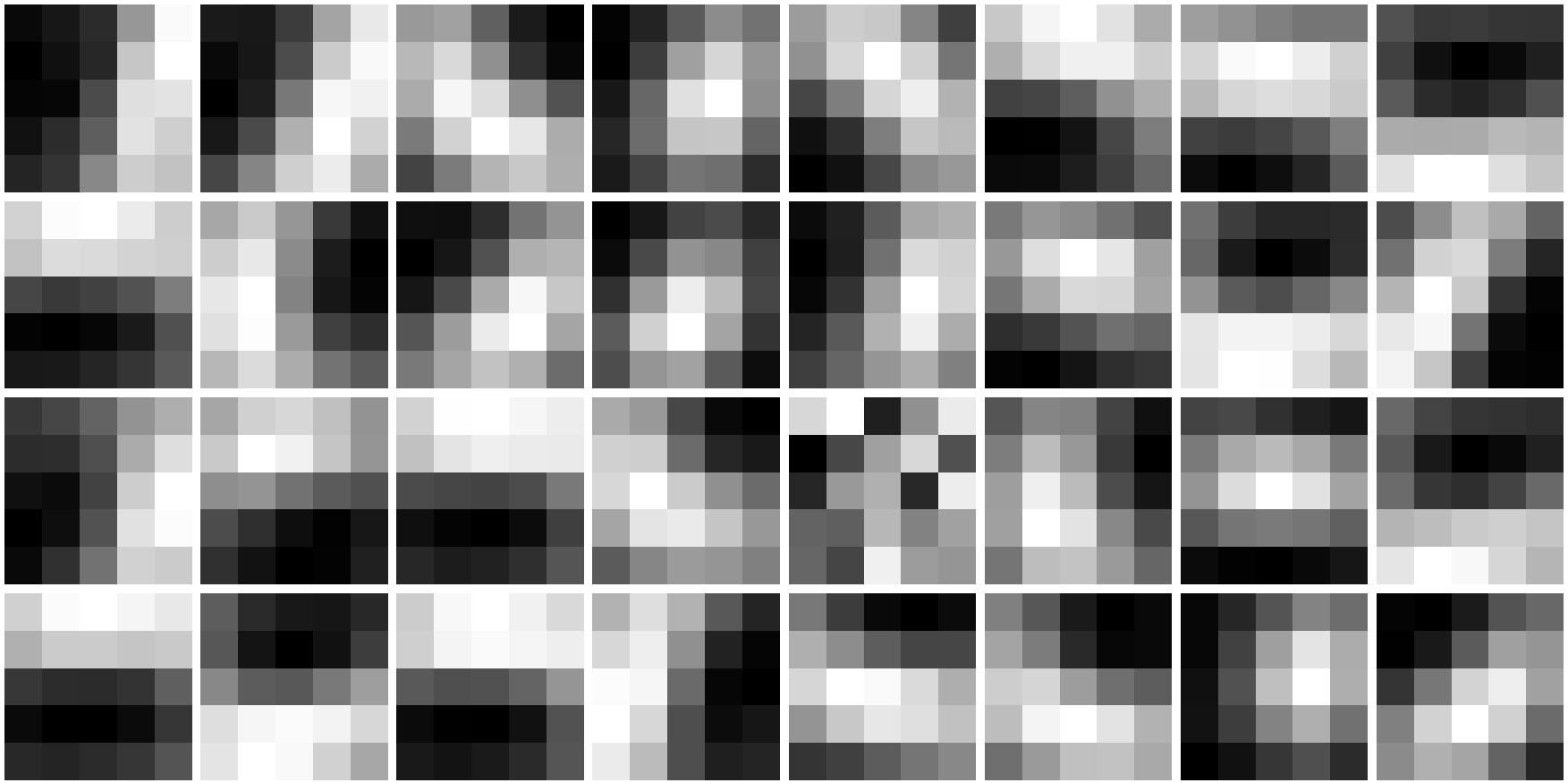
convolutional network

	MNIST			CIFAR10		
	param	ops	volume	param	ops	volume
input	0	0	$28 \times 28 \times 1$	0	0	$32 \times 32 \times 3$
conv$(5, 32)$	832	479232	$24 \times 24 \times 32$	2432	1906688	$28 \times 28 \times 32$
pool(2)	0	18432	$12 \times 12 \times 32$	0	25088	$14 \times 14 \times 32$
conv$(5, 64)$	51264	3280896	$8 \times 8 \times 64$	51264	5126400	$10 \times 10 \times 64$
pool(2)	0	4096	$4 \times 4 \times 64$	0	6400	$5 \times 5 \times 64$
fc(100)	102500	102500	100	160100	160100	100
fc(10)	1010	1010	10	1010	1010	10
softmax	0	0	10	0	0	10

- most parameters in first fully connected layer
- most operations in second convolutional layer
- most memory in first convolutional layer

conv$(r, k'[, p = 0][, s = 1])$; (max)- pool$(r[, s = r][, p = 0])$;

MNIST layer 1 filters



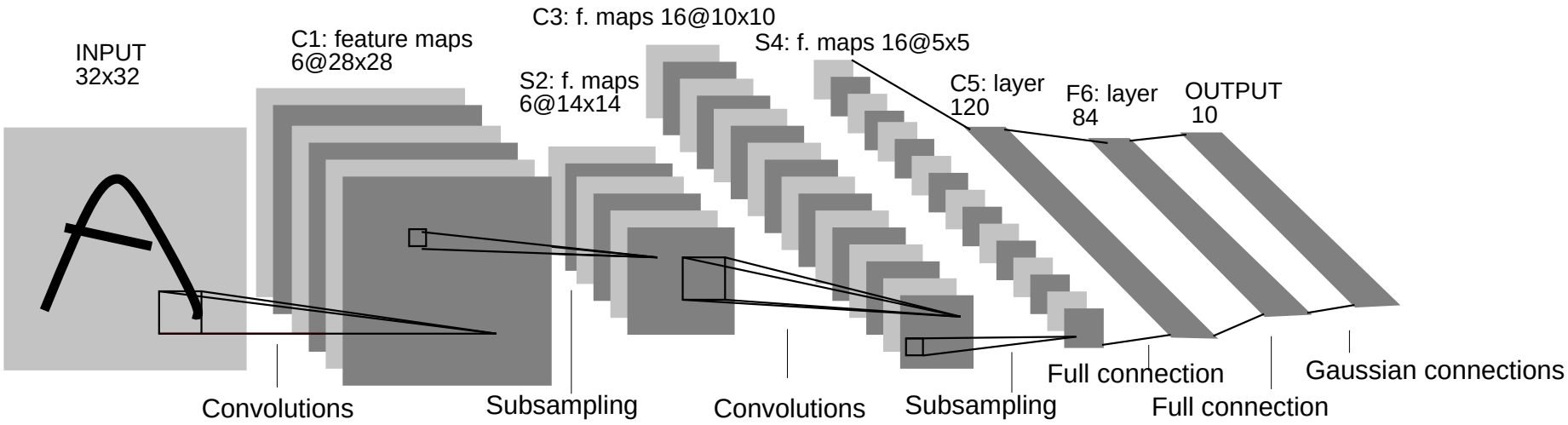
- mini-batch $m = 128$, learning rate $\epsilon = 10^{-2}$, regularization strength $\lambda = 10^{-2}$, Gaussian initialization $\sigma = 0.1$
- test error: 1.2%

CIFAR10 layer 1 filters

- mini-batch $m = 128$, learning rate $\epsilon = 10^{-2}$, regularization strength $\lambda = 10^{-2}$, Gaussian initialization $\sigma = 0.1$
- test error: 28%

LeNet-5

[LeCun et al. 1998]



- first convolutional neural network to use back-propagation
- applied to character recognition

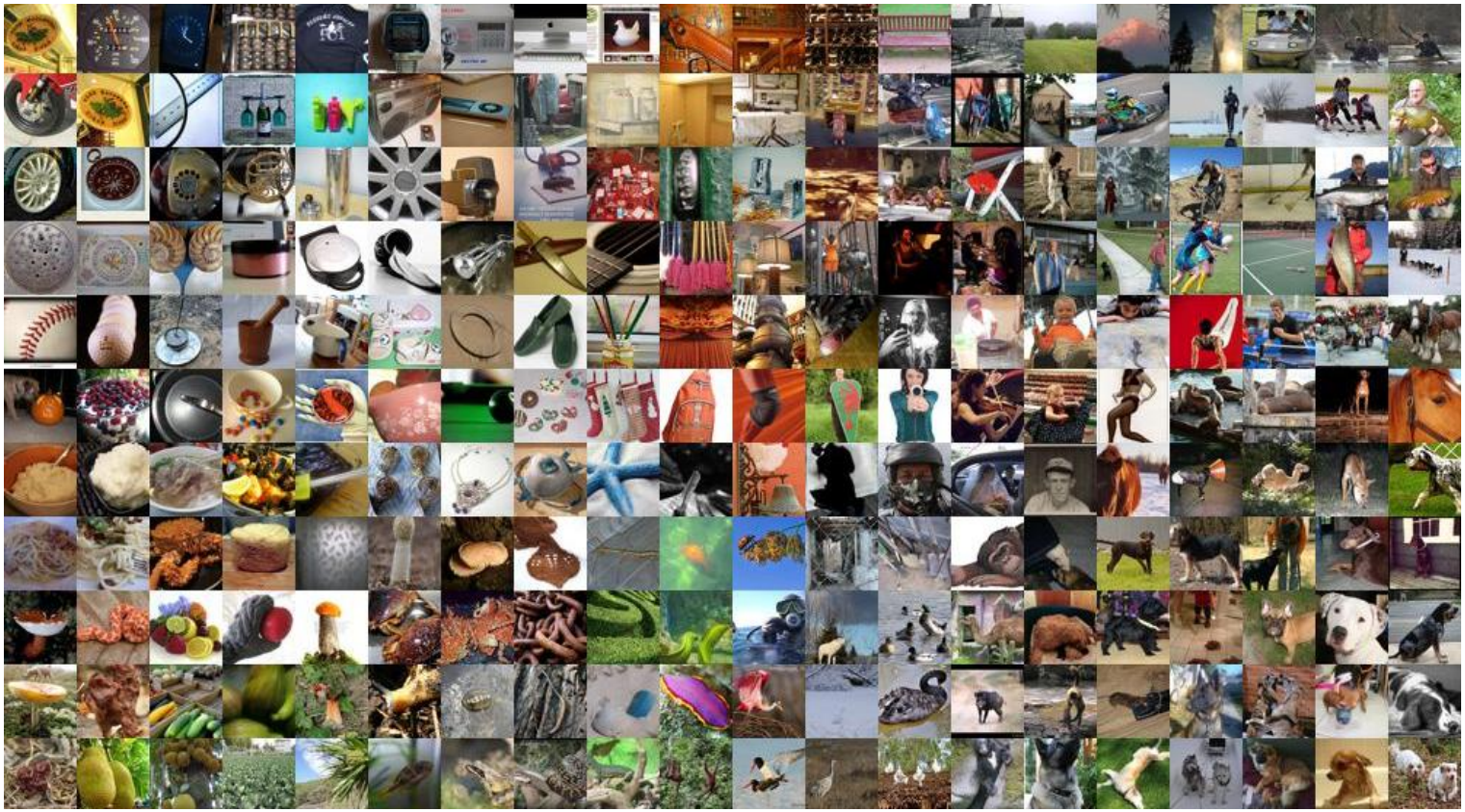
LeNet-5

	parameters	operations	volume
input$(32, 1)$	0	0	$32 \times 32 \times 1$
conv$(5, 6)$	156	$122,304$	$28 \times 28 \times 6$
avg(2)	0	$4,704$	$14 \times 14 \times 6$
conv$(5, 16)$	$2,416$	$241,600$	$10 \times 10 \times 16$
avg(2)	0	$1,600$	$5 \times 5 \times 16$
conv$(5, 120)$	$48,120$	$48,120$	$1 \times 1 \times 120$
fc(84)	$10,164$	$10,164$	84
RBF(10)	850	850	10
softmax	0	10	10

- subsampling by average pooling with learnable global weight and bias
- scaled $\tanh$ function after first pooling layer and FC layer
- last convolutional layer allows variable-sized input
- output RBF units: Euclidean distance to 7×12 distributed codes
- $\mathrm{softmax}$-like loss function

ImageNet

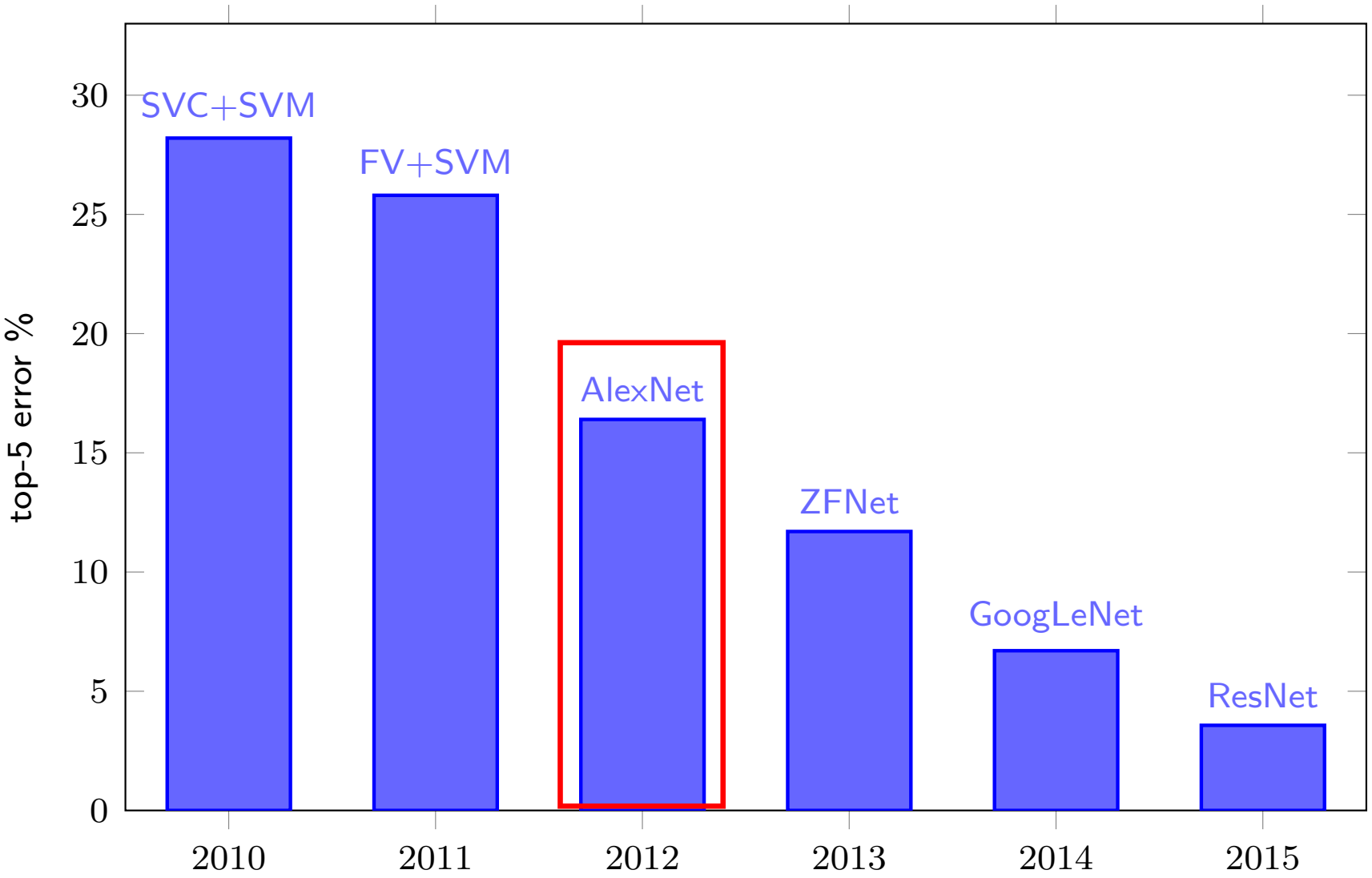
[Russakovsky et al. 2014]



- 22k classes, 15M samples
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC): 1000 classes, 1.2M training images, 50k validation images, 150k test images

Russakovsky, Deng, Su, Krause, *et al.* 2014. Imagenet Large Scale Visual Recognition Challenge.

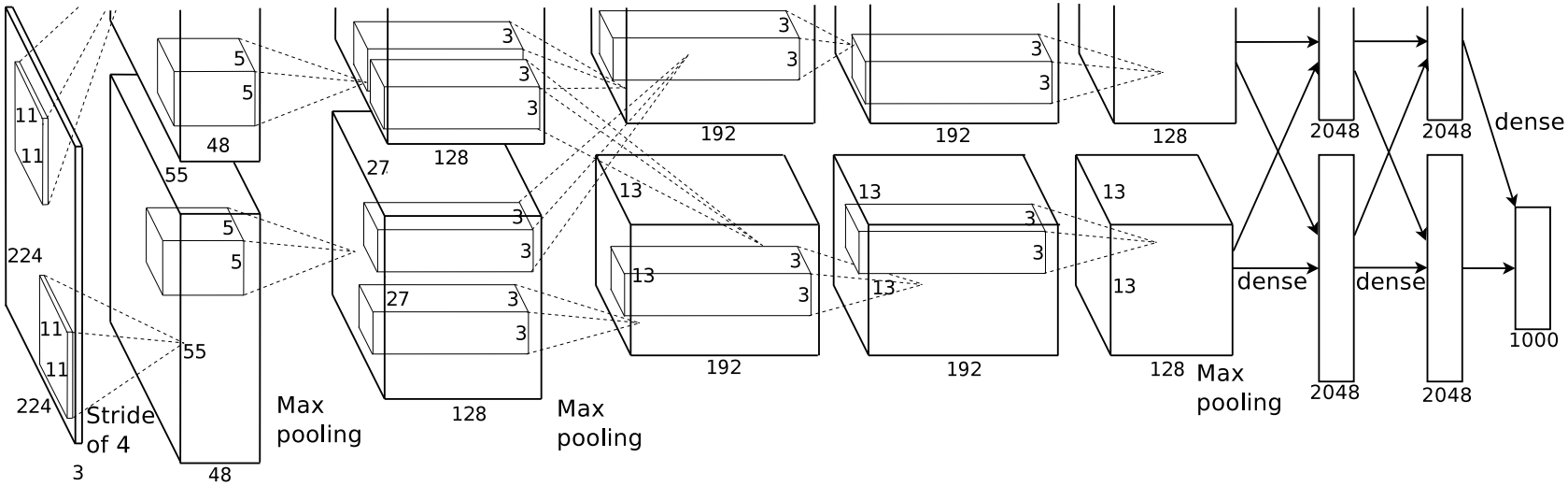
ImageNet classification performance



Russakovsky, Deng, Su, Krause, *et al*. 2014. Imagenet Large Scale Visual Recognition Challenge.

AlexNet

[Krizhevsky et al. 2012]

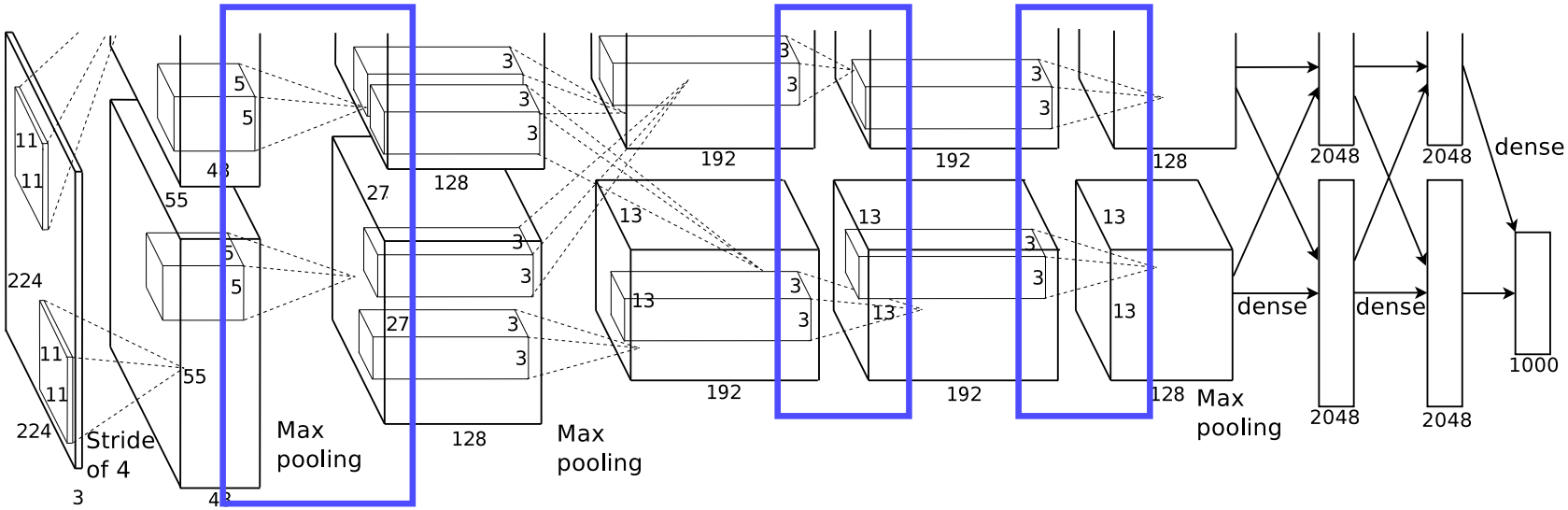


- 16.4% top-5 error on on ILSVRC'12, outperformed all by 10%
- 8 layers
- ReLU, local response normalization, data augmentation, dropout
- stochastic gradient descent with momentum
- implementation on two GPUs; connectivity between the two subnetworks is limited

Krizhevsky, Sutskever, Hinton. NIPS 2012. Imagenet Classification with Deep Convolutional Neural Networks.

AlexNet

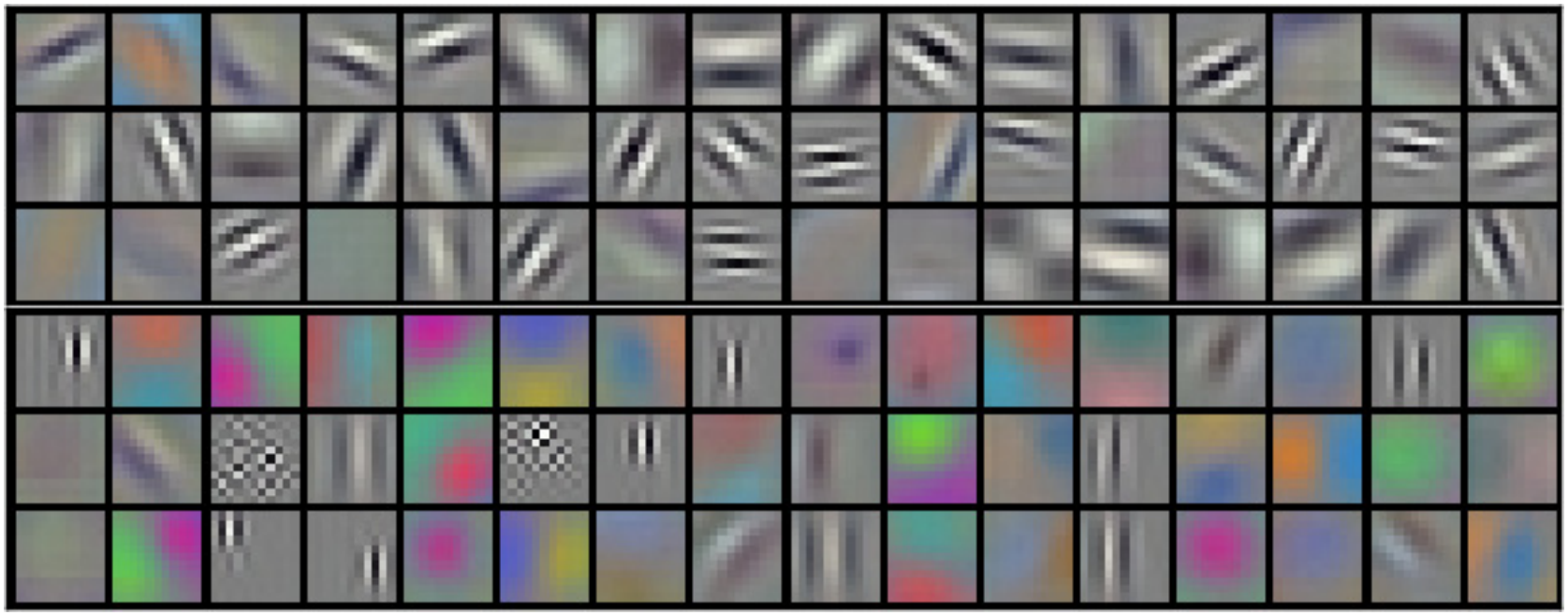
[Krizhevsky et al. 2012]



- 16.4% top-5 error on on ILSVRC'12, outperformed all by 10%
- 8 layers
- ReLU, local response normalization, data augmentation, dropout
- stochastic gradient descent with momentum
- implementation on two GPUs; connectivity between the two subnetworks is limited

Krizhevsky, Sutskever, Hinton. NIPS 2012. Imagenet Classification with Deep Convolutional Neural Networks.

learned layer 1 kernels



- 96 kernels of size $11 \times 11 \times 3$
- top: 48 GPU 1 kernels; bottom: 48 GPU 2 kernels

Krizhevsky, Sutskever, Hinton. NIPS 2012. Imagenet Classification with Deep Convolutional Neural Networks.

AlexNet (CaffeNet)

	parameters	operations	volume
input$(227, 3)$	0	0	$227 \times 227 \times 3$
conv$(11, 96, s4)$	$34,944$	$105,705,600$	$55 \times 55 \times 96$
pool$(3, 2)$	0	$290,400$	$27 \times 27 \times 96$
norm	0	$69,984$	$27 \times 27 \times 96$
conv$(5, 256, p2)$	$614,656$	$448,084,224$	$27 \times 27 \times 256$
pool$(3, 2)$	0	$186,624$	$13 \times 13 \times 256$
norm	0	$43,264$	$13 \times 13 \times 256$
conv$(3, 384, p1)$	$885,120$	$149,585,280$	$13 \times 13 \times 384$
conv$(3, 384, p1)$	$1,327,488$	$224,345,472$	$13 \times 13 \times 384$
conv$(3, 256, p1)$	$884,992$	$149,563,648$	$13 \times 13 \times 256$
pool$(3, 2)$	0	$43,264$	$6 \times 6 \times 256$
fc(4096)	$37,752,832$	$37,752,832$	$4,096$
fc(4096)	$16,781,312$	$16,781,312$	$4,096$
fc(1000)	$4,097,000$	$4,097,000$	$1,000$
softmax	0	$1,000$	$1,000$

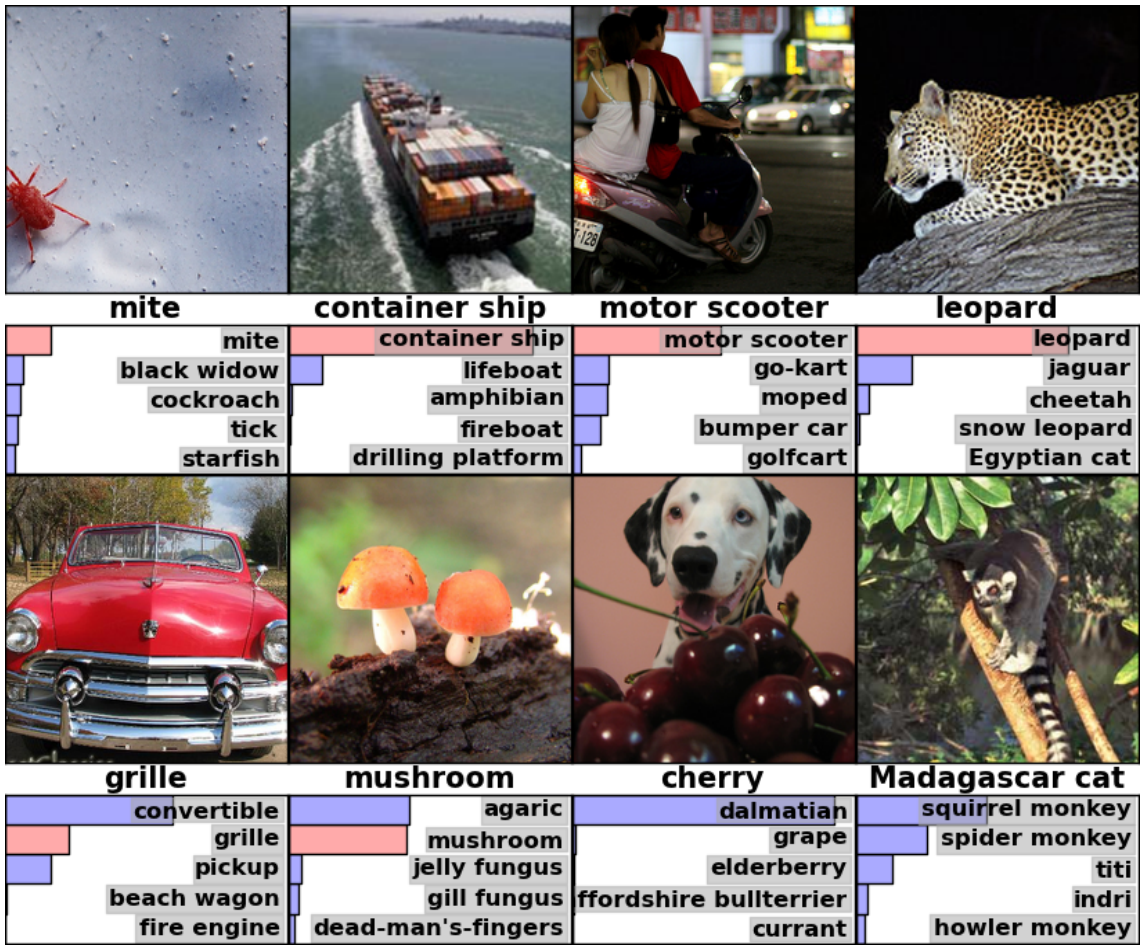
- ReLU follows each convolutional and fully connected layer
- CaffeNet: input size modified from 224×224, pool/norm switched

AlexNet (CaffeNet)

	parameters	operations	volume
input$(227, 3)$	0	0	$227 \times 227 \times 3$
conv$(11, 96, s4)$	$34,944$	$105,705,600$	$55 \times 55 \times 96$
pool$(3, 2)$	0	$290,400$	$27 \times 27 \times 96$
norm	0	$69,984$	$27 \times 27 \times 96$
conv$(5, 256, p2)$	$614,656$	$448,084,224$	$27 \times 27 \times 256$
pool$(3, 2)$	0	$186,624$	$13 \times 13 \times 256$
norm	0	$43,264$	$13 \times 13 \times 256$
conv$(3, 384, p1)$	$885,120$	$149,585,280$	$13 \times 13 \times 384$
conv$(3, 384, p1)$	$1,327,488$	$224,345,472$	$13 \times 13 \times 384$
conv$(3, 256, p1)$	$884,992$	$149,563,648$	$13 \times 13 \times 256$
pool$(3, 2)$	0	$43,264$	$6 \times 6 \times 256$
fc(4096)	$37,752,832$	$37,752,832$	$4,096$
fc(4096)	$16,781,312$	$16,781,312$	$4,096$
fc(1000)	$4,097,000$	$4,097,000$	$1,000$
softmax	0	$1,000$	$1,000$

- ReLU follows each convolutional and fully connected layer
- CaffeNet: input size modified from 224×224, pool/norm switched

AlexNet: classification examples



- correct label on top; its predicted probability with red if visible

Krizhevsky, Sutskever, Hinton. NIPS 2012. Imagenet Classification with Deep Convolutional Neural Networks.

VGG

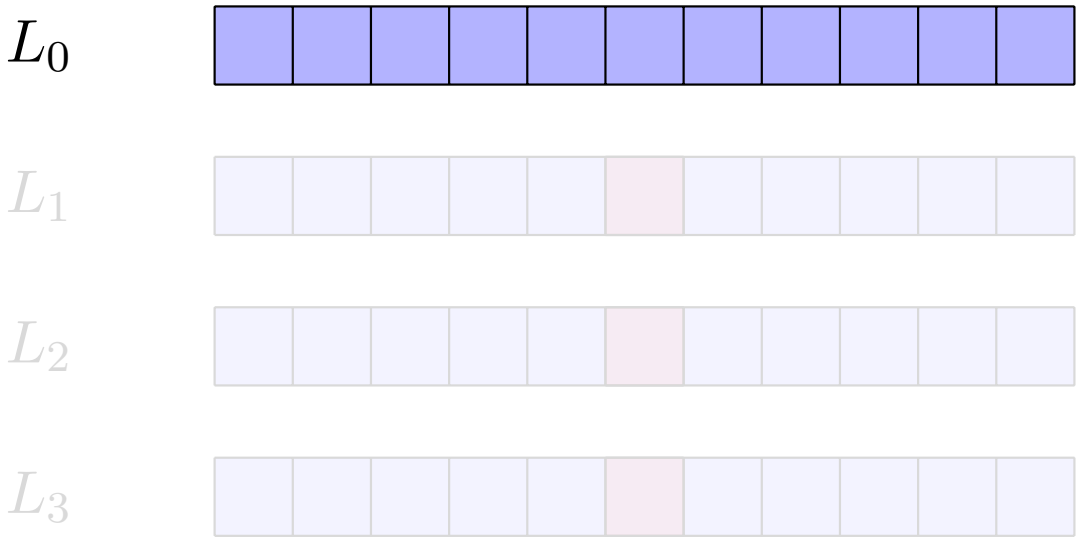
[Simonyan and Zisserman 2014]

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**
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**
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**
maxpool					

- 7.3% top-5 error on ILSVRC'14
- depth increased up to 19 layers, kernel sizes (strides) reduced to $3(1)$
- local response normalization doesn't do anything
- top/bottom layers of deep models pre-initialized by trained model A

Simonyan and Zisserman 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition.

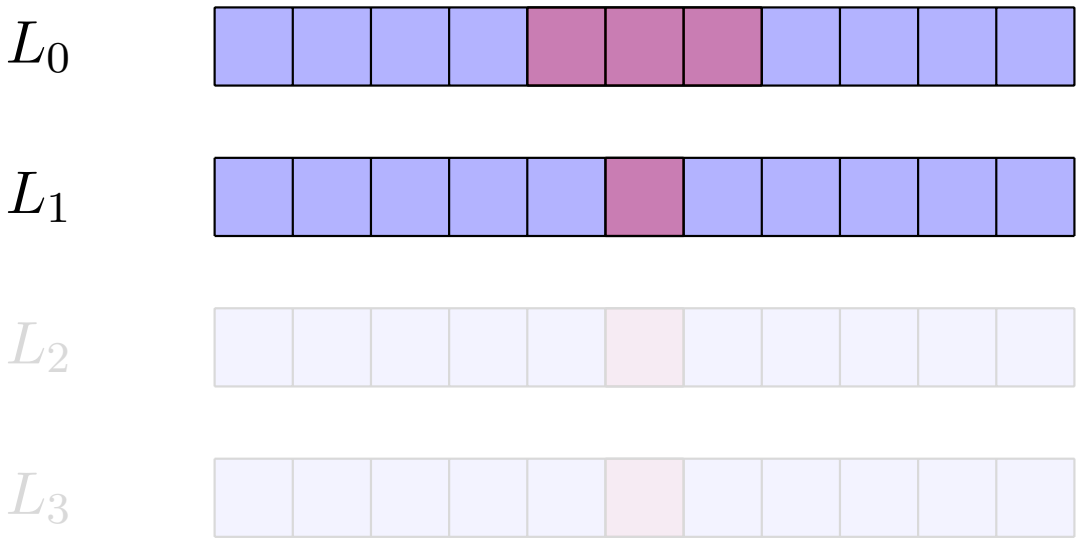
effective receptive field



- is the part of the visual input that affects a given cell indirectly through previous layers
- grows linearly with depth
- stack of three 3×3 kernels of stride 1 has the same effective receptive field as a single 7×7 kernel, but fewer parameters

Simonyan and Zisserman 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition.

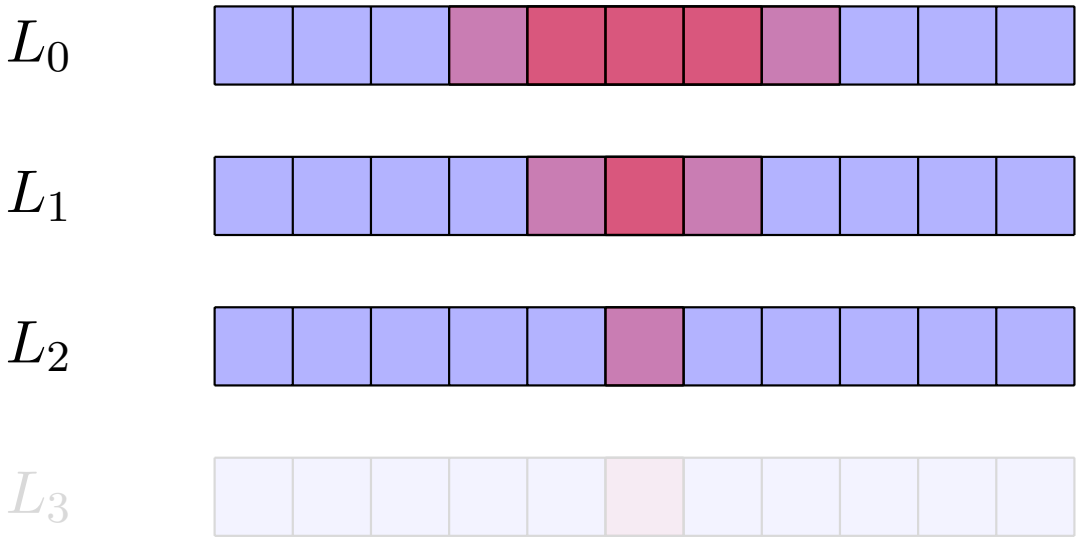
effective receptive field



- is the part of the visual input that affects a given cell indirectly through previous layers
- grows linearly with depth
- stack of three 3×3 kernels of stride 1 has the same effective receptive field as a single 7×7 kernel, but fewer parameters

Simonyan and Zisserman 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition.

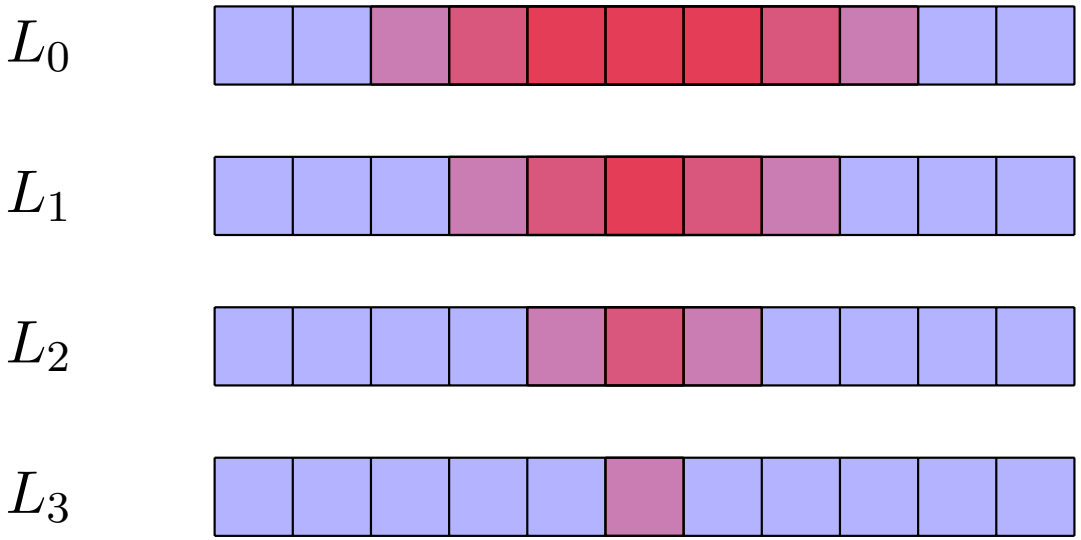
effective receptive field



- is the part of the visual input that affects a given cell indirectly through previous layers
- grows linearly with depth
- stack of three 3×3 kernels of stride 1 has the same effective receptive field as a single 7×7 kernel, but fewer parameters

Simonyan and Zisserman 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition.

effective receptive field



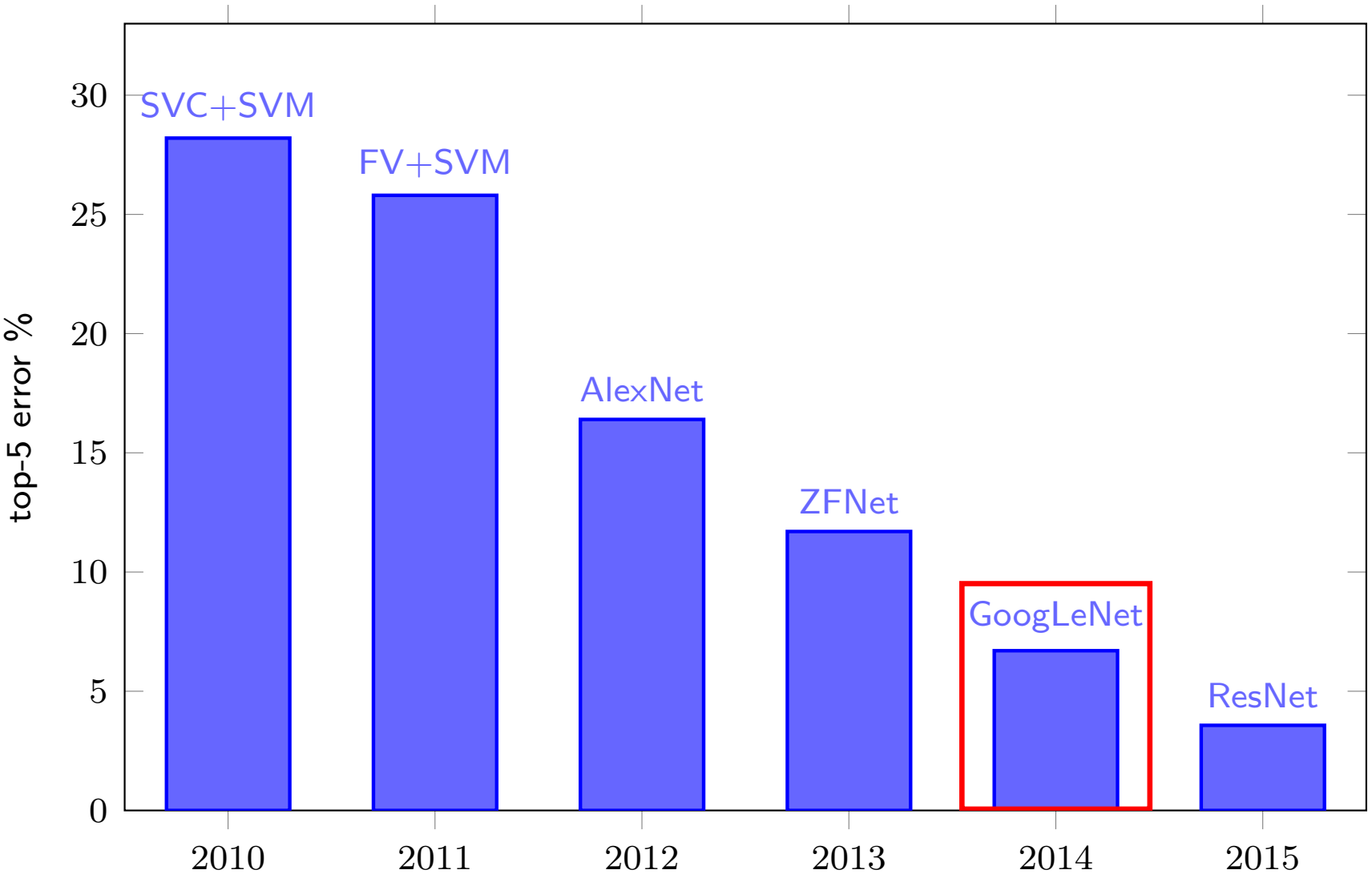
- is the part of the visual input that affects a given cell indirectly through previous layers
- grows linearly with depth
- stack of three 3×3 kernels of stride 1 has the same effective receptive field as a single 7×7 kernel, but fewer parameters

Simonyan and Zisserman 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition.

VGG-16

	parameters	operations	volume
input$(224, 3)$	0	0	$224 \times 224 \times 3$
conv$(3, 64, p1)$	$1,792$	$89,915,390$	$224 \times 224 \times 64$
conv$(3, 64, p1)$	$36,928$	$1,852,899,328$	$224 \times 224 \times 64$
pool(2)	0	$3,211,264$	$112 \times 112 \times 64$
conv$(3, 128, p1)$	$73,856$	$926,449,664$	$112 \times 112 \times 128$
conv$(3, 128, p1)$	$147,584$	$1,851,293,696$	$112 \times 112 \times 128$
pool(2)	0	$1,605,632$	$56 \times 56 \times 128$
conv$(3, 256, p1)$	$295,168$	$925,646,848$	$56 \times 56 \times 256$
conv$(3, 256, p1)$	$590,080$	$1,850,490,880$	$56 \times 56 \times 256$
conv$(3, 256, p1)$	$590,080$	$1,850,490,880$	$56 \times 56 \times 256$
pool(2)	0	$802,816$	$28 \times 28 \times 256$
conv$(3, 512, p1)$	$1,180,160$	$925,245,440$	$28 \times 28 \times 512$
conv$(3, 512, p1)$	$2,359,808$	$1,850,089,472$	$28 \times 28 \times 512$
conv$(3, 512, p1)$	$2,359,808$	$1,850,089,472$	$28 \times 28 \times 512$
pool(2)	0	$401,408$	$14 \times 14 \times 512$
conv$(3, 512, p1)$	$2,359,808$	$462,522,368$	$14 \times 14 \times 512$
conv$(3, 512, p1)$	$2,359,808$	$462,522,368$	$14 \times 14 \times 512$
conv$(3, 512, p1)$	$2,359,808$	$462,522,368$	$14 \times 14 \times 512$
pool(2)	0	$100,352$	$7 \times 7 \times 512$
fc(4096)	$102,764,544$	$102,764,544$	$4,096$
fc(4096)	$16,781,312$	$16,781,312$	$4,096$
fc(1000)	$4,097,000$	$4,097,000$	$1,000$
softmax	0	$1,000$	$1,000$

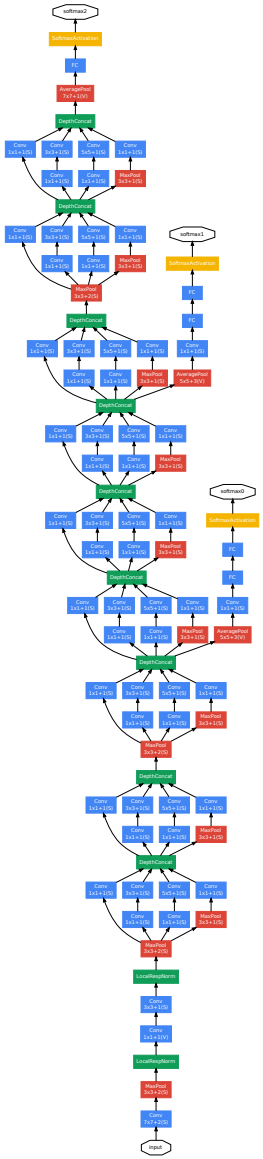
ImageNet classification performance



Russakovsky, Deng, Su, Krause, *et al*. 2014. Imagenet Large Scale Visual Recognition Challenge.

GoogLeNet

[Szegedy et al. 2015]



- 6.7% top-5 error on ILSVRC'14
- depth increased to 22 layers, kernel sizes 1×1 to 5×5
- inception module repeated 9 times
- 1×1 kernels used as "bottleneck" layers (dimensionality reduction)
- 25 times less parameters and faster than AlexNet
- auxiliary classifiers

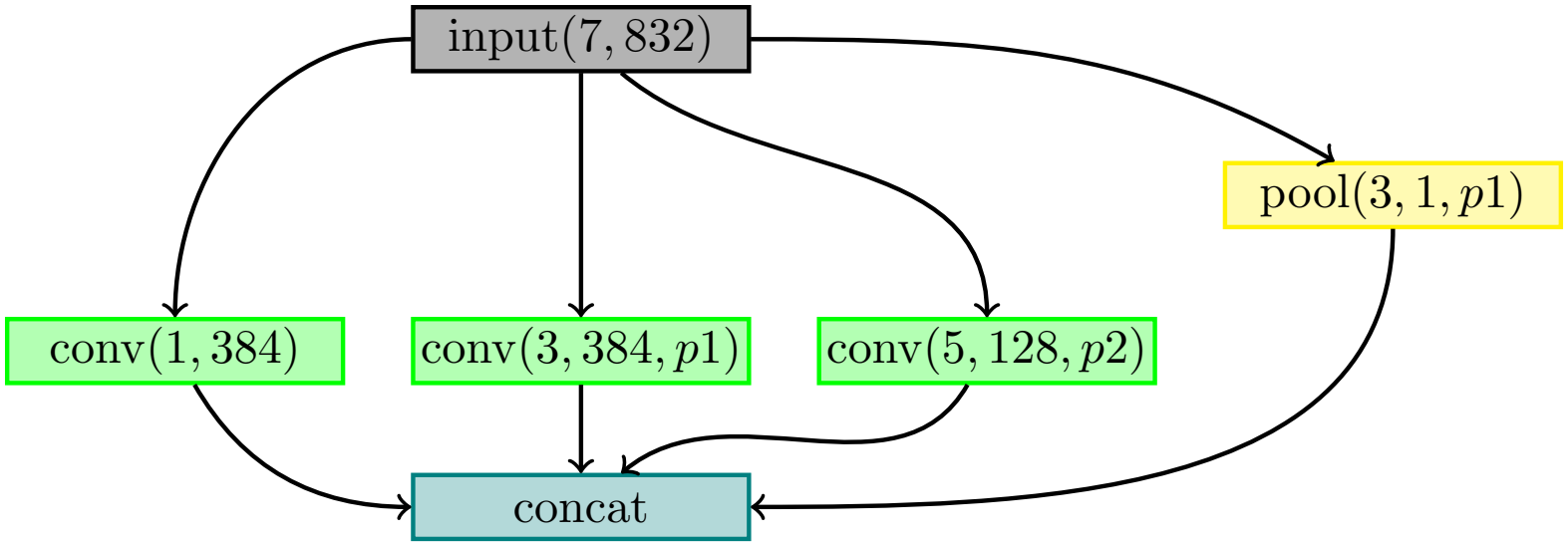
Szegedy, Liu, Jia, Sermanet, Reed, Anguelov, Erhan, Vanhoucke and Rabinovich. CVPR 2015. Going Deeper with Convolutions.

convolutional features are sparse

- deep layers have more features (*e.g.* 1024) and lower resolutions (*e.g.* 7×7)
  - detected patterns in many cases are as small as 5×5, 3×3 or even 1×1
  - the convolution operation resembles more (sparse) matrix multiplication than convolution
- sparse matrix multiplication
  - is not as efficient as dense on parallel hardware
  - modern methods use coarse partitioning of nonzero elements
  - this is aligned with the Hebbian rule "cells that fire together wire together"

Szegedy, Liu, Jia, Sermanet, Reed, Anguelov, Erhan, Vanhoucke and Rabinovich. CVPR 2015. Going Deeper with Convolutions.

inception module

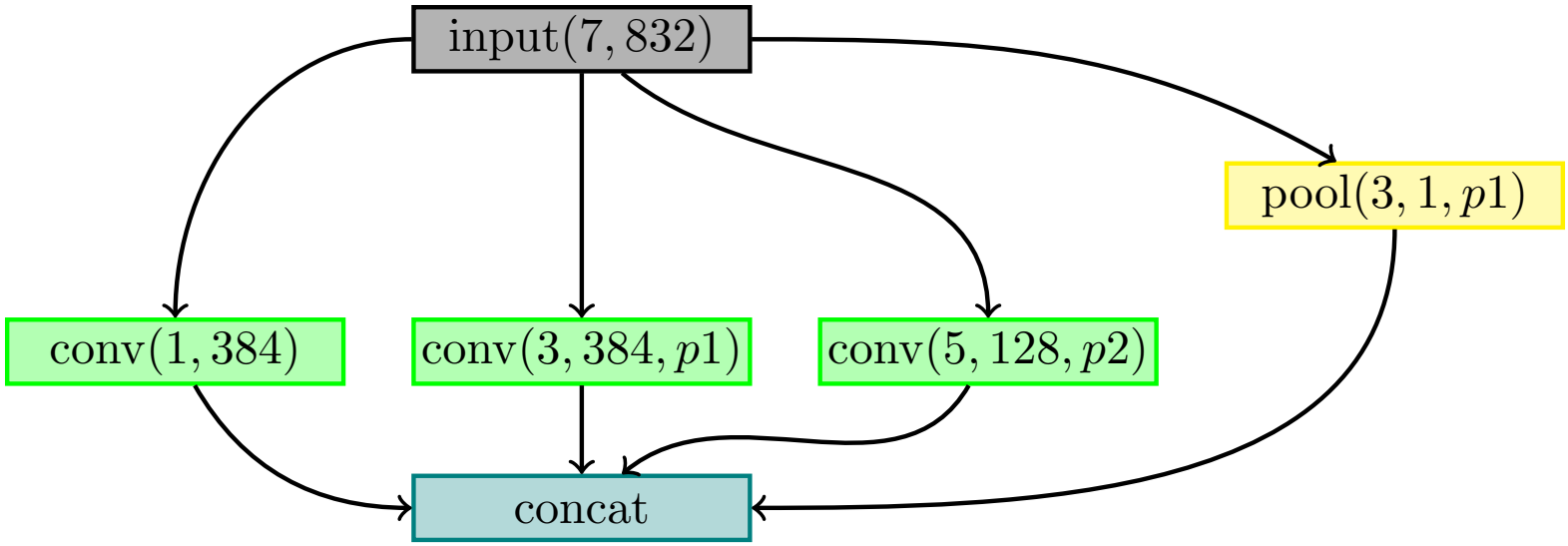


- naive inception module simply concatenates (feature-wise) three convolutions and one max-pooling
- but this expensive and dimension keeps increasing
- add dimension reduction to control cost, dimensions, and sparsity
- this is referred to as inception module

Szegedy, Liu, Jia, Sermanet, Reed, Anguelov, Erhan, Vanhoucke and Rabinovich. CVPR 2015. Going Deeper with Convolutions.

inception module

$271,418,048$ operations

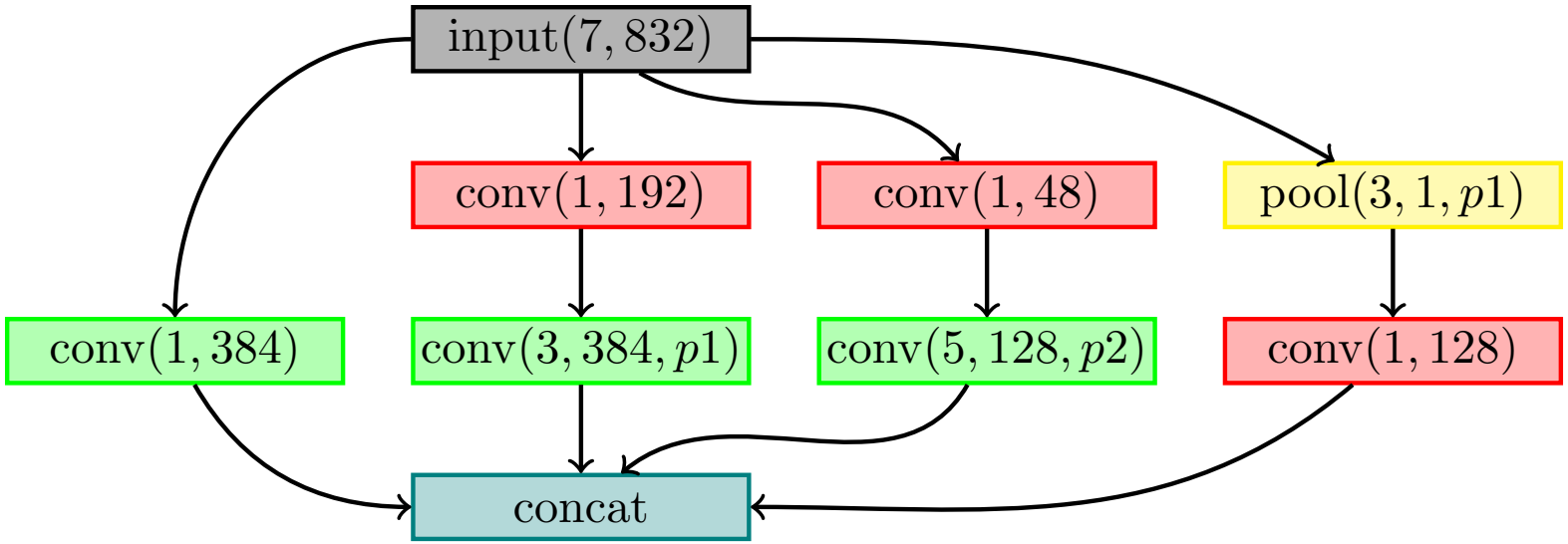


- naive inception module simply concatenates (feature-wise) three convolutions and one max-pooling
- but this expensive and dimension keeps increasing
- add dimension reduction to control cost, dimensions, and sparsity
- this is referred to as inception module

Szegedy, Liu, Jia, Sermanet, Reed, Anguelov, Erhan, Vanhoucke and Rabinovich. CVPR 2015. Going Deeper with Convolutions.

inception module

$70,800,688$ operations

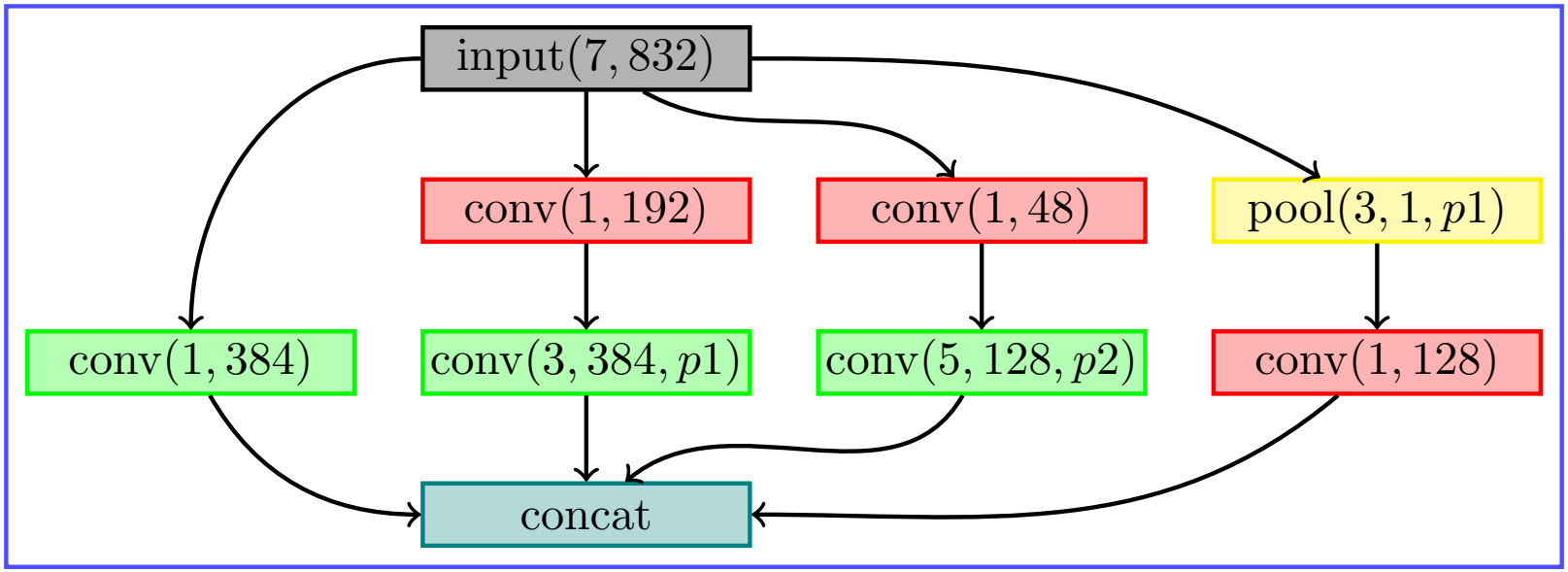


- naive inception module simply concatenates (feature-wise) three convolutions and one max-pooling
- but this expensive and dimension keeps increasing
- add dimension reduction to control cost, dimensions, and sparsity
- this is referred to as inception module

Szegedy, Liu, Jia, Sermanet, Reed, Anguelov, Erhan, Vanhoucke and Rabinovich. CVPR 2015. Going Deeper with Convolutions.

inception module

$70,800,688$ operations

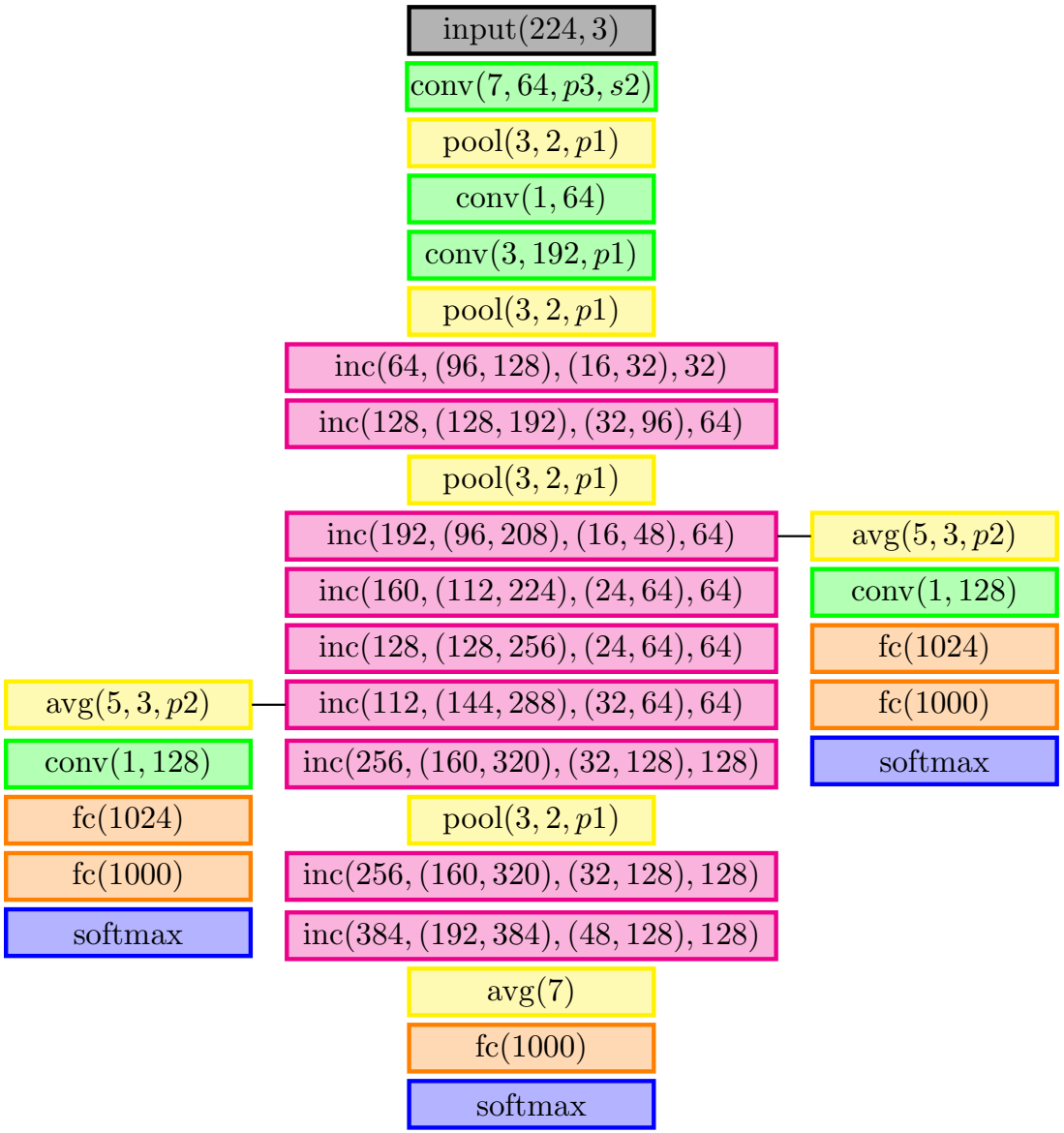


$\text{inc}(384, (192, 384), (48, 128), 128)$

- naive inception module simply concatenates (feature-wise) three convolutions and one max-pooling
- but this expensive and dimension keeps increasing
- add dimension reduction to control cost, dimensions, and sparsity
- this is referred to as inception module

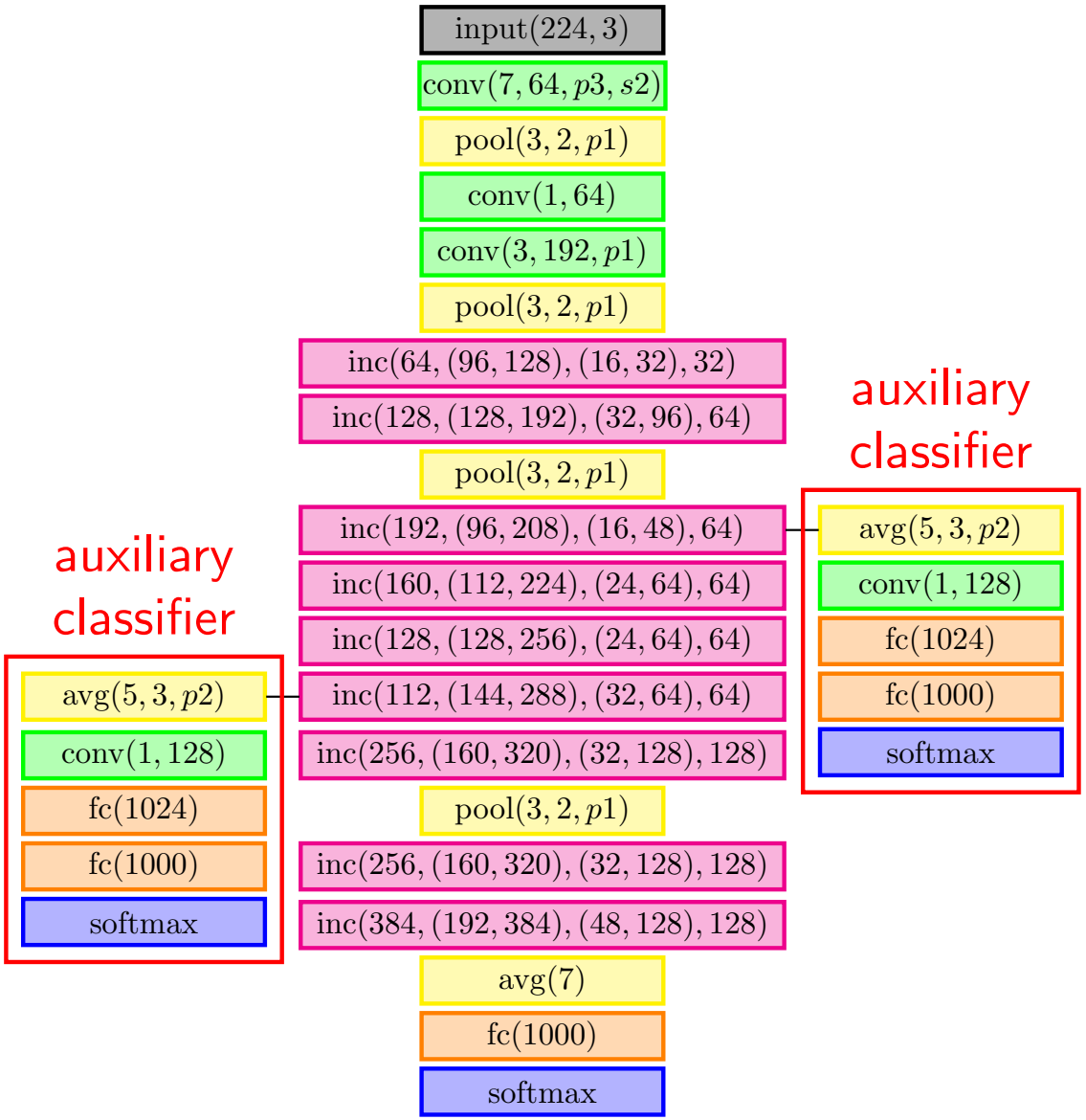
Szegedy, Liu, Jia, Sermanet, Reed, Anguelov, Erhan, Vanhoucke and Rabinovich. CVPR 2015. Going Deeper with Convolutions.

GoogLeNet



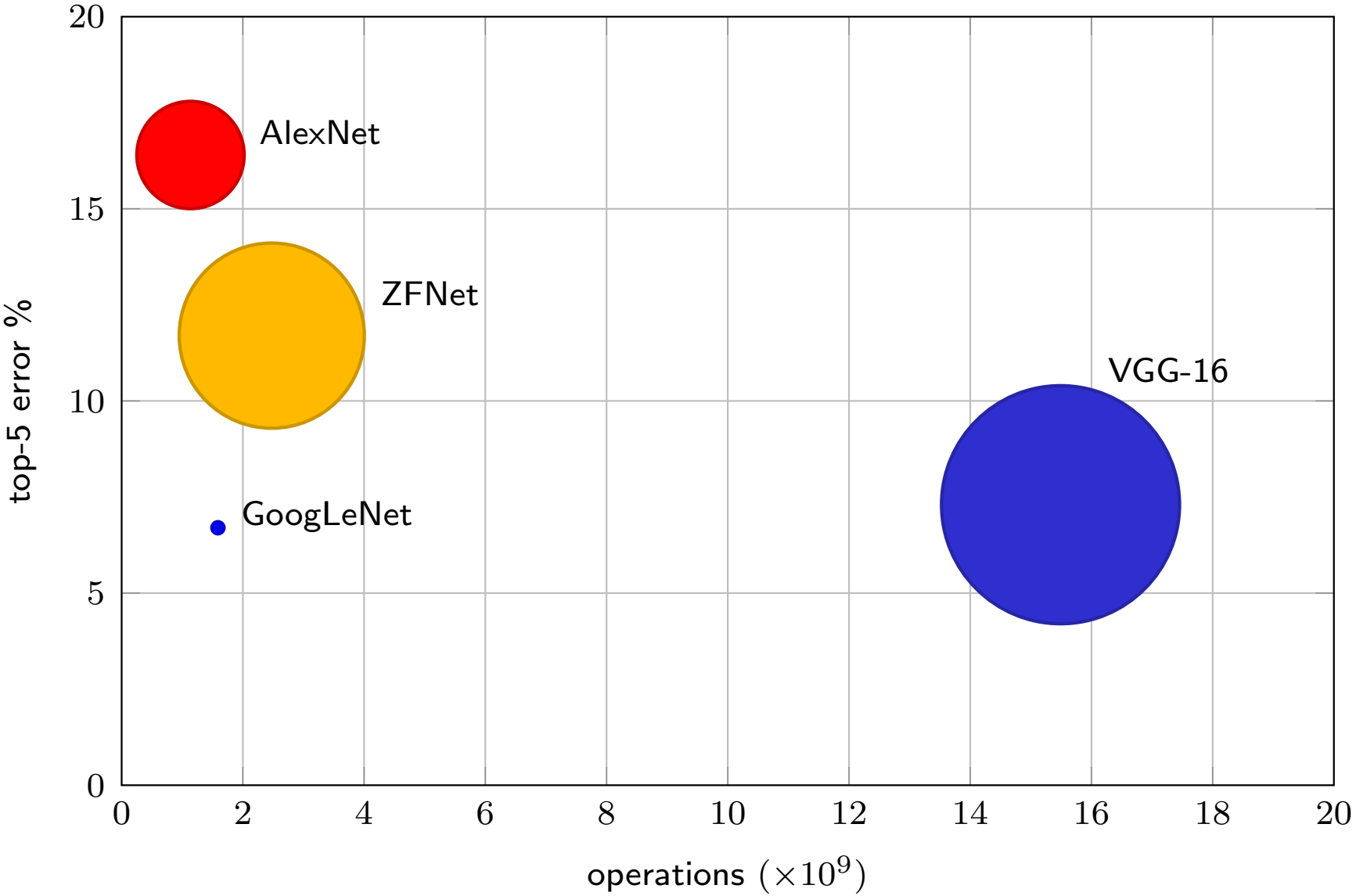
	parameters	operations	volume
input	0	0	$224 \times 224 \times 3$
conv(7, 64, p3, s2)	9,472	118,816,768	$112 \times 112 \times 64$
pool(3, 2, p1)	0	802,816	$56 \times 56 \times 64$
conv(1, 64)	4,160	13,045,760	$56 \times 56 \times 64$
conv(3, 192, p1)	110,784	347,418,624	$56 \times 56 \times 192$
pool(3, 2, p1)	0	602,112	$28 \times 28 \times 192$
inc(64, (96, 128), (16, 32), 32)	163,696	128,488,192	$28 \times 28 \times 256$
inc(128, (128, 192), (32, 96), 64)	388,736	304,969,728	$28 \times 28 \times 480$
pool(3, 2, p1)	0	376,320	$14 \times 14 \times 480$
inc(192, (96, 208), (16, 48), 64)	376,176	73,824,576	$14 \times 14 \times 512$
inc(160, (112, 224), (24, 64), 64)	449,160	88,135,712	$14 \times 14 \times 512$
inc(128, (128, 256), (24, 64), 64)	510,104	100,080,736	$14 \times 14 \times 512$
inc(112, (144, 288), (32, 64), 64)	605,376	118,754,048	$14 \times 14 \times 528$
inc(256, (160, 320), (32, 128), 128)	868,352	170,300,480	$14 \times 14 \times 832$
pool(3, 2, p1)	0	163,072	$7 \times 7 \times 832$
inc(256, (160, 320), (32, 128), 128)	1,043,456	51,170,112	$7 \times 7 \times 832$
inc(384, (192, 384), (48, 128), 128)	1,444,080	70,800,688	$7 \times 7 \times 1024$
avg(7)	0	50,176	$1 \times 1 \times 1024$
fc(1000)	1,025,000	1,025,000	1,000
softmax	0	1,000	1,000

GoogLeNet



parameters	operations	volume
0	0	$224 \times 224 \times 3$
9,472	118,816,768	$112 \times 112 \times 64$
0	802,816	$56 \times 56 \times 64$
4,160	13,045,760	$56 \times 56 \times 64$
110,784	347,418,624	$56 \times 56 \times 192$
0	602,112	$28 \times 28 \times 192$
163,696	128,488,192	$28 \times 28 \times 256$
388,736	304,969,728	$28 \times 28 \times 480$
0	376,320	$14 \times 14 \times 480$
376,176	73,824,576	$14 \times 14 \times 512$
449,160	88,135,712	$14 \times 14 \times 512$
510,104	100,080,736	$14 \times 14 \times 512$
605,376	118,754,048	$14 \times 14 \times 528$
868,352	170,300,480	$14 \times 14 \times 832$
0	163,072	$7 \times 7 \times 832$
1,043,456	51,170,112	$7 \times 7 \times 832$
1,444,080	70,800,688	$7 \times 7 \times 1024$
0	50,176	$1 \times 1 \times 1024$
1,025,000	1,025,000	1,000
0	1,000	1,000

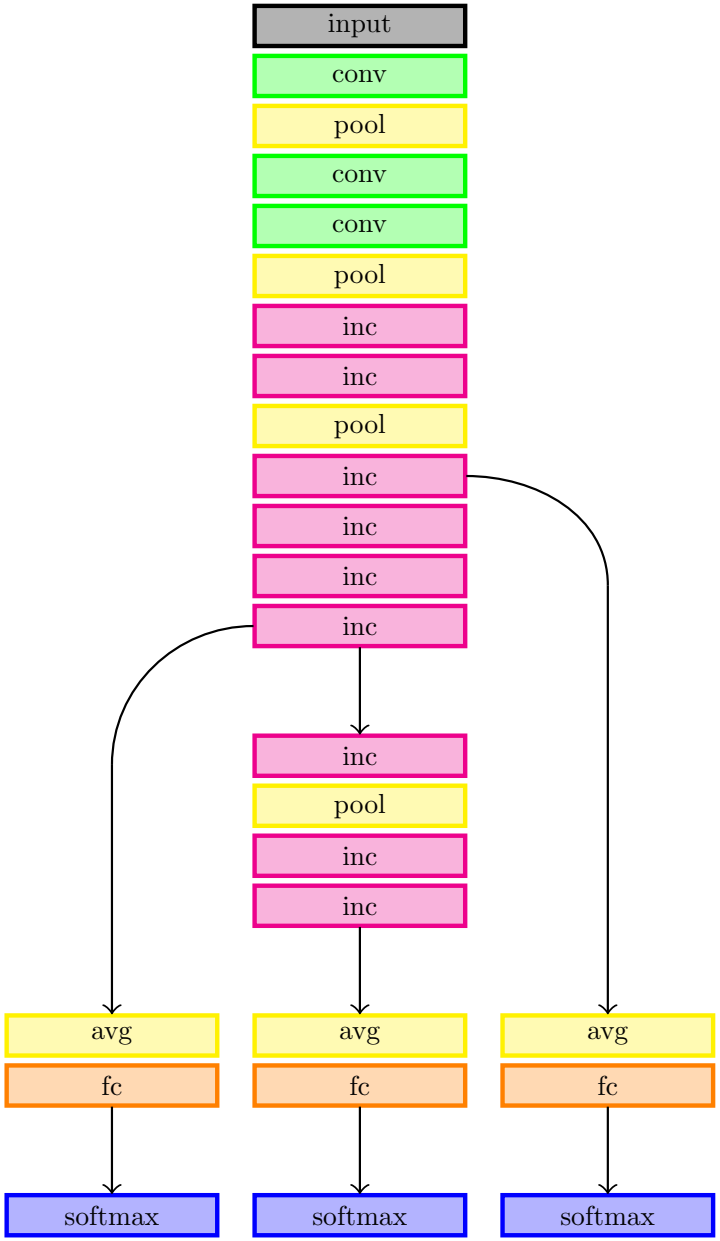
network performance



Canziani, Culurciello and Paszke. 2016. An Analysis of Deep Neural Network Models for Practical Applications.

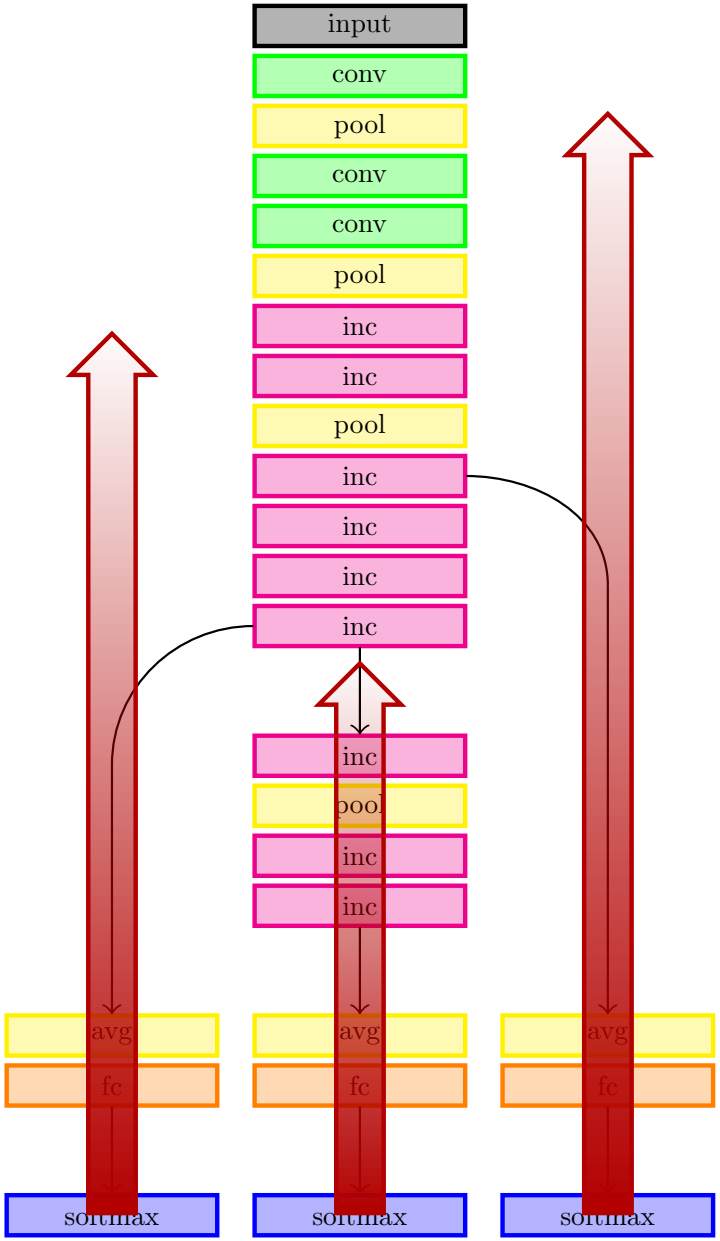
deeper architectures

remember GoogLeNet auxiliary classifiers?



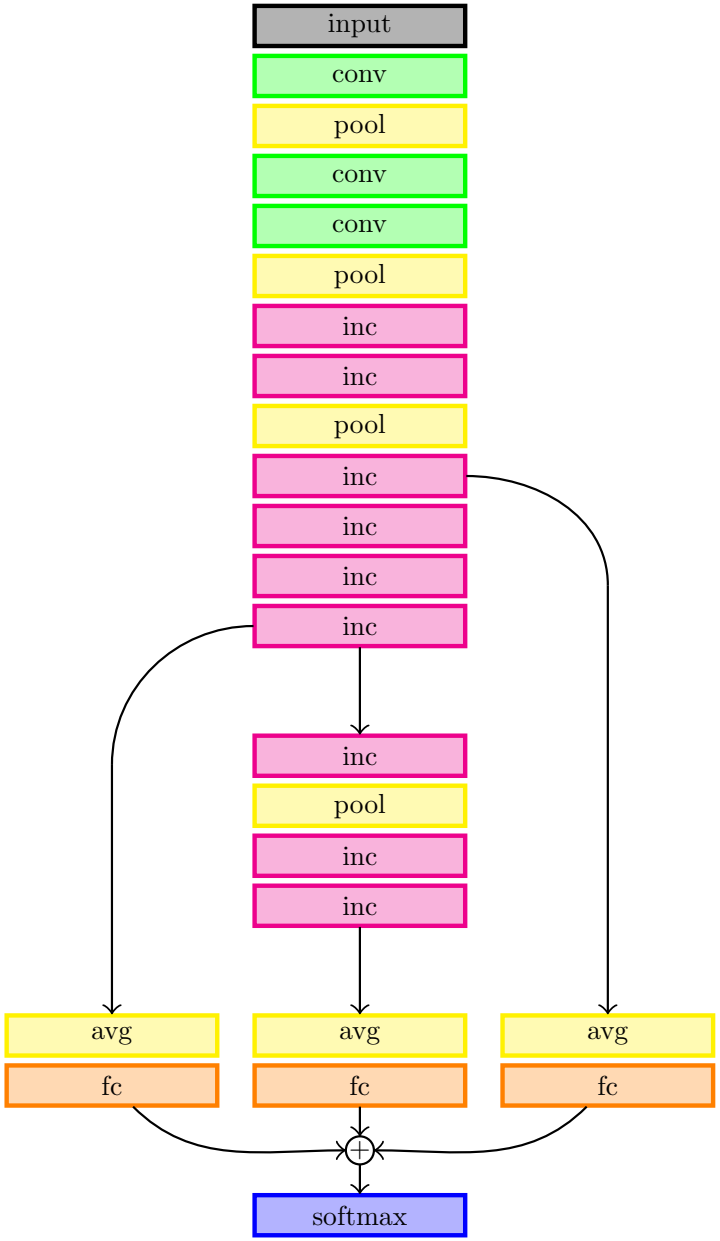
- GoogLeNet has two auxiliary classifiers that are discarded at inference
- these classifiers inject gradient signal deeper backwards
- we now transform the network in ways that are not necessarily equivalent, but maintain this backward flow pattern
- the result is two skip connections that can be maintained at inference

remember GoogLeNet auxiliary classifiers?



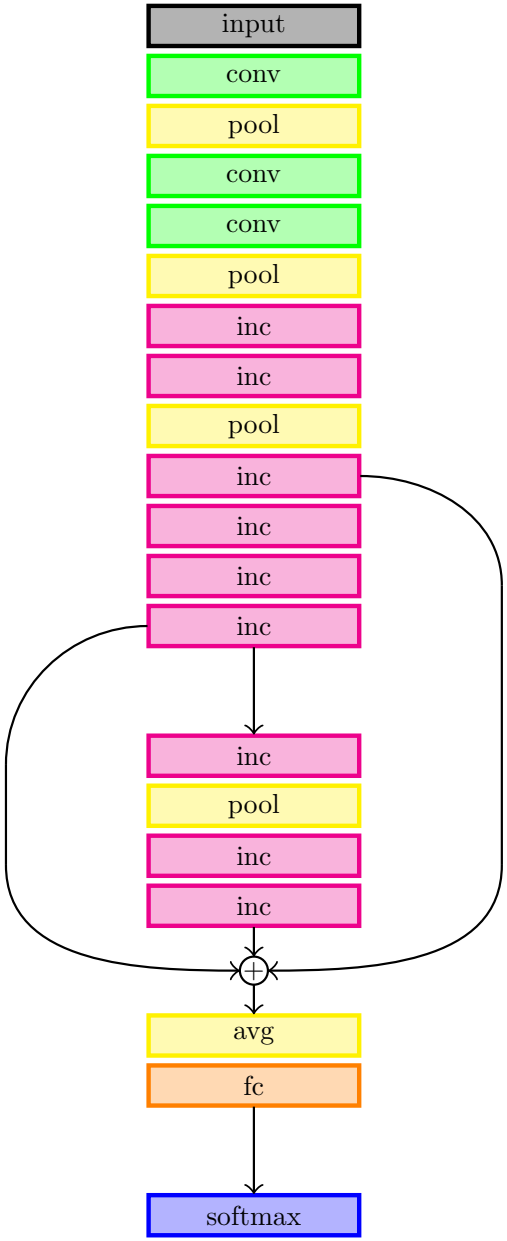
- GoogLeNet has two auxiliary classifiers that are discarded at inference
- these classifiers inject gradient signal deeper backwards
- we now transform the network in ways that are not necessarily equivalent, but maintain this backward flow pattern
- the result is two skip connections that can be maintained at inference

remember GoogLeNet auxiliary classifiers?



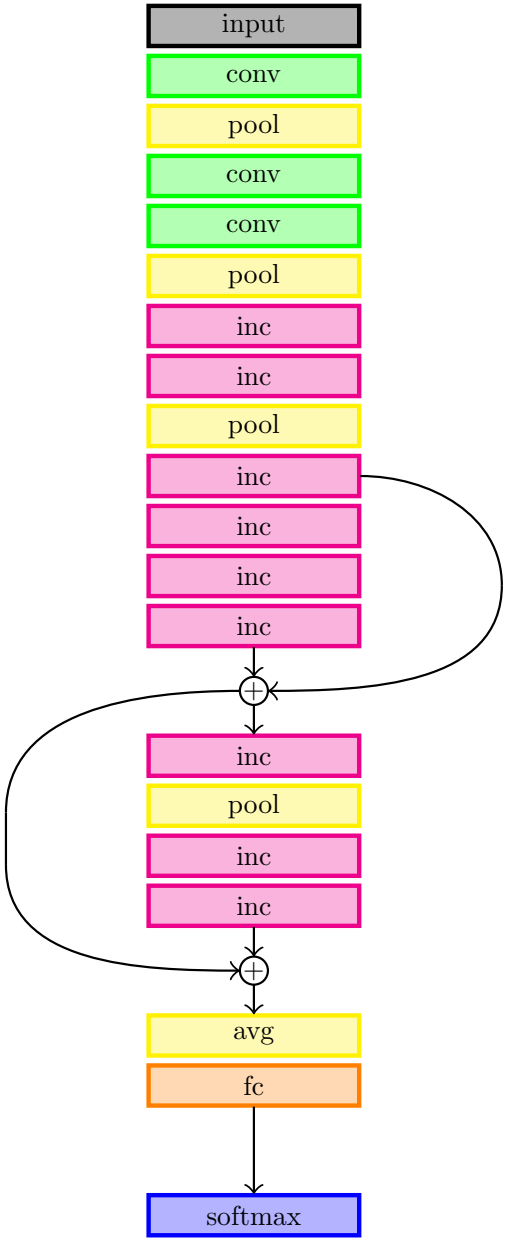
- GoogLeNet has two auxiliary classifiers that are discarded at inference
- these classifiers inject gradient signal deeper backwards
- we now transform the network in ways that are not necessarily equivalent, but maintain this backward flow pattern
- the result is two skip connections that can be maintained at inference

remember GoogLeNet auxiliary classifiers?



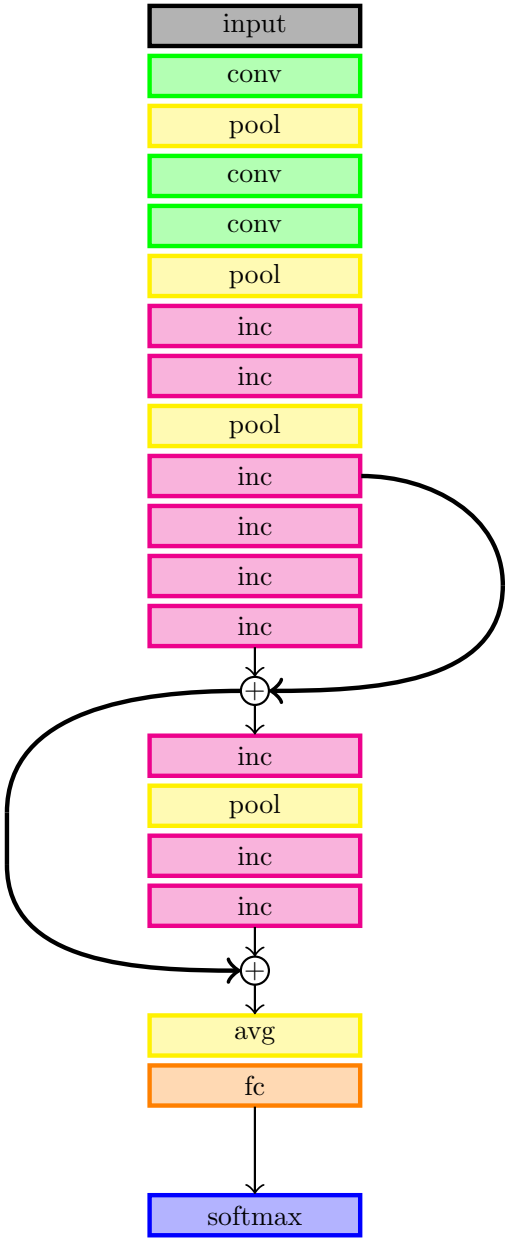
- GoogLeNet has two auxiliary classifiers that are discarded at inference
- these classifiers inject gradient signal deeper backwards
- we now transform the network in ways that are not necessarily equivalent, but maintain this backward flow pattern
- the result is two skip connections that can be maintained at inference

remember GoogLeNet auxiliary classifiers?



- GoogLeNet has two auxiliary classifiers that are discarded at inference
- these classifiers inject gradient signal deeper backwards
- we now transform the network in ways that are not necessarily equivalent, but maintain this backward flow pattern
- the result is two skip connections that can be maintained at inference

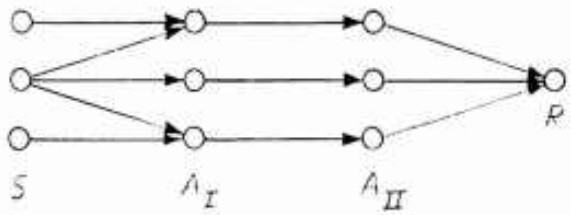
remember GoogLeNet auxiliary classifiers?



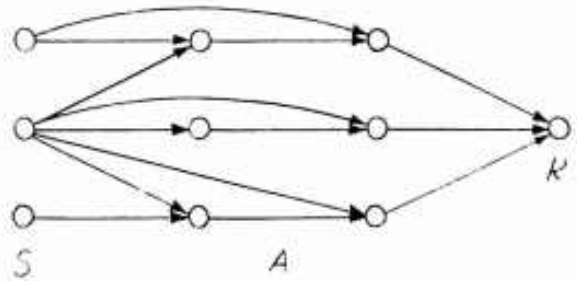
- GoogLeNet has two auxiliary classifiers that are discarded at inference
- these classifiers inject gradient signal deeper backwards
- we now transform the network in ways that are not necessarily equivalent, but maintain this backward flow pattern
- the result is two skip connections that can be maintained at inference

skip connections are not new

the network diagram:



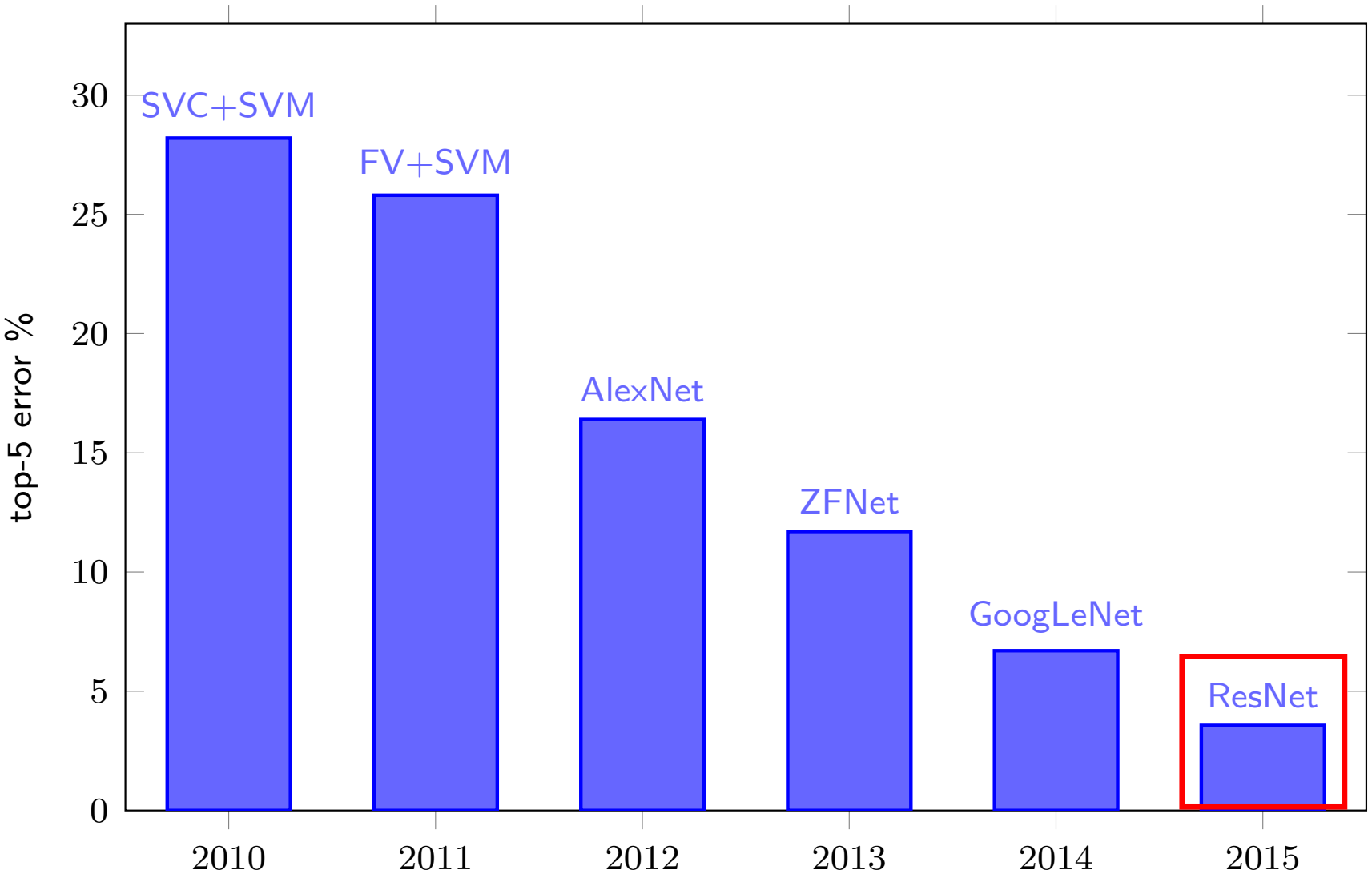
represents a four-layer series-coupled system, whereas the diagram



represents a three-layer cross coupled system, since all A-units are at least the same logical distance from the sensory units (see Definition 18,

Rosenblatt 1962. Principles of Neurodynamics.

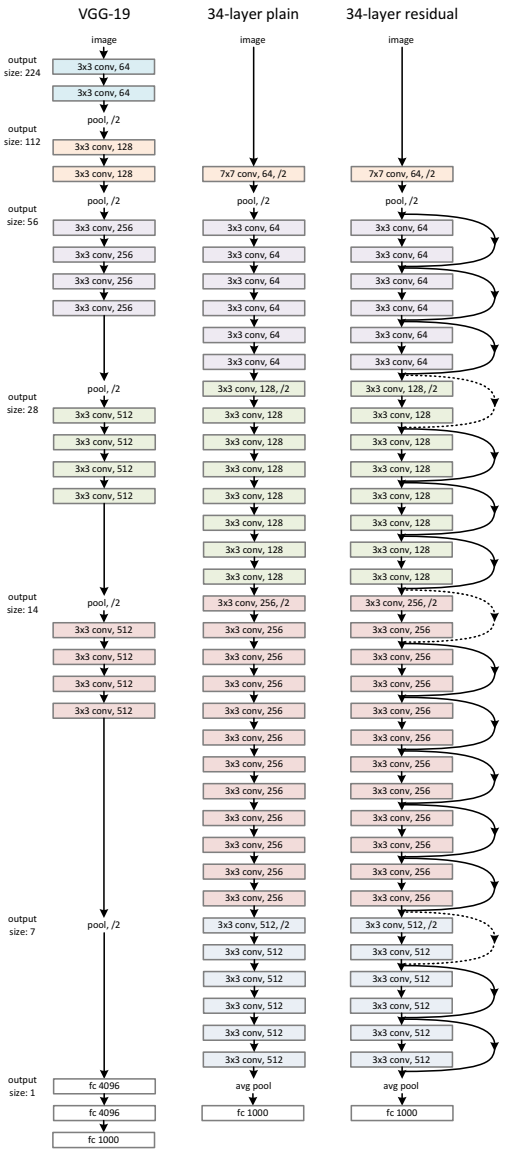
ImageNet classification performance



Russakovsky, Deng, Su, Krause, *et al*. 2014. Imagenet Large Scale Visual Recognition Challenge.

residual networks

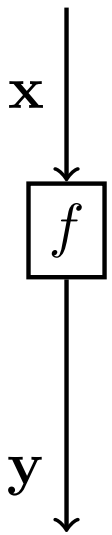
[He et al. 2016]



- 3.57% top-5 error on ILSVRC'15
- won first place on several ILSVRC and COCO 2015 tasks
- depth increased to 152 layers, kernel size mostly 3×3
- residual unit repeated up to 50 times
- 1×1 kernels used as "bottleneck" layers
- up to $10\times$ more operations but same parameters as AlexNet

He, Zhang, Ren, Sun. CVPR 2016. Deep Residual Learning for Image Recognition.

skip connections and residual



- "plain" unit: f is the mapping

$$\mathbf{y} = f(\mathbf{x})$$

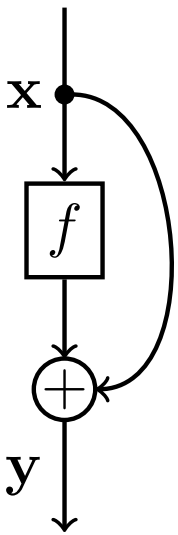
- residual unit: f is the residual

$$\mathbf{y} = \mathbf{x} + f(\mathbf{x})$$

- by copying the features of a shallow model and setting the new mapping to the identity, a deeper model performs at least as well as the shallow one
- "if an identity mapping were optimal, it would be easier to push a residual to zero than to fit an identity mapping by a stack of nonlinear layers"

He, Zhang, Ren, Sun. CVPR 2016. Deep Residual Learning for Image Recognition.

skip connections and residual



- "plain" unit: f is the mapping

$$\mathbf{y} = f(\mathbf{x})$$

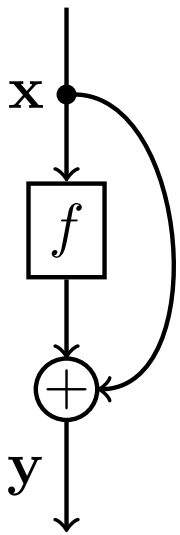
- residual unit: f is the residual

$$\mathbf{y} = \mathbf{x} + f(\mathbf{x})$$

- by copying the features of a shallow model and setting the new mapping to the identity, a deeper model performs at least as well as the shallow one
- "if an identity mapping were optimal, it would be easier to push a residual to zero than to fit an identity mapping by a stack of nonlinear layers"

He, Zhang, Ren, Sun. CVPR 2016. Deep Residual Learning for Image Recognition.

skip connections and residual



- "plain" unit: f is the mapping

$$\mathbf{y} = f(\mathbf{x})$$

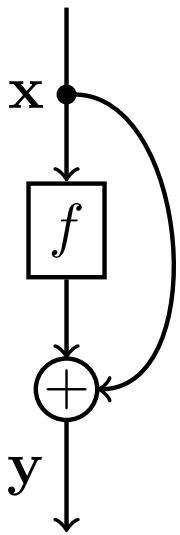
- residual unit: f is the residual

$$\mathbf{y} = \mathbf{x} + f(\mathbf{x})$$

- by copying the features of a shallow model and setting the new mapping to the identity, a deeper model performs at least as well as the shallow one
- "if an identity mapping were optimal, it would be easier to push a residual to zero than to fit an identity mapping by a stack of nonlinear layers"

He, Zhang, Ren, Sun. CVPR 2016. Deep Residual Learning for Image Recognition.

skip connections and residual



- "plain" unit: f is the mapping

$$\mathbf{y} = f(\mathbf{x})$$

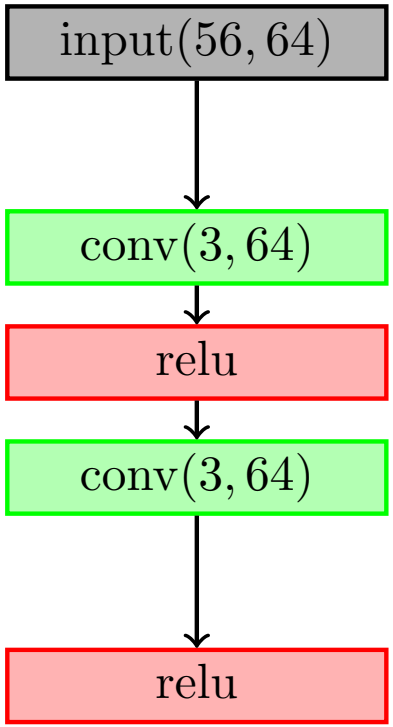
- residual unit: f is the residual

$$\mathbf{y} = \mathbf{x} + f(\mathbf{x})$$

- by copying the features of a shallow model and setting the new mapping to the identity, a deeper model performs at least as well as the shallow one
- "if an identity mapping were optimal, it would be easier to push a residual to zero than to fit an identity mapping by a stack of nonlinear layers"

He, Zhang, Ren, Sun. CVPR 2016. Deep Residual Learning for Image Recognition.

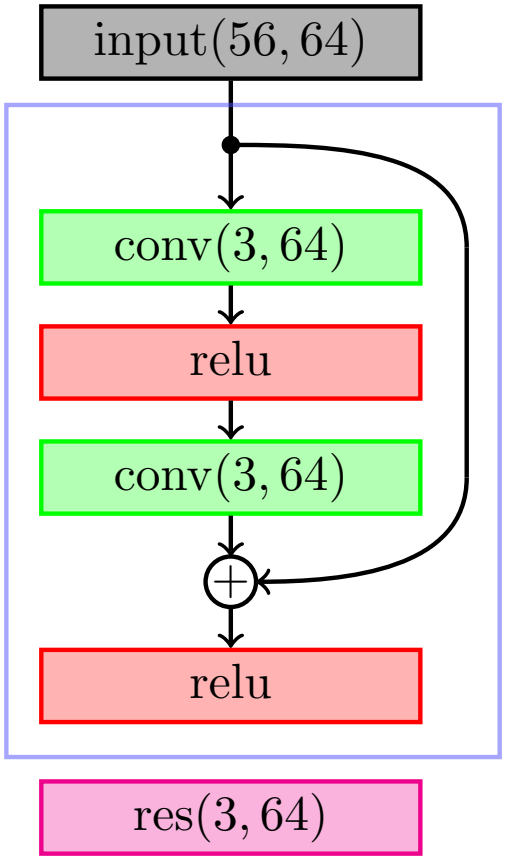
residual unit



- "plain" unit, with nonlinearities shown separately, and batch normalization included in each convolutional layers

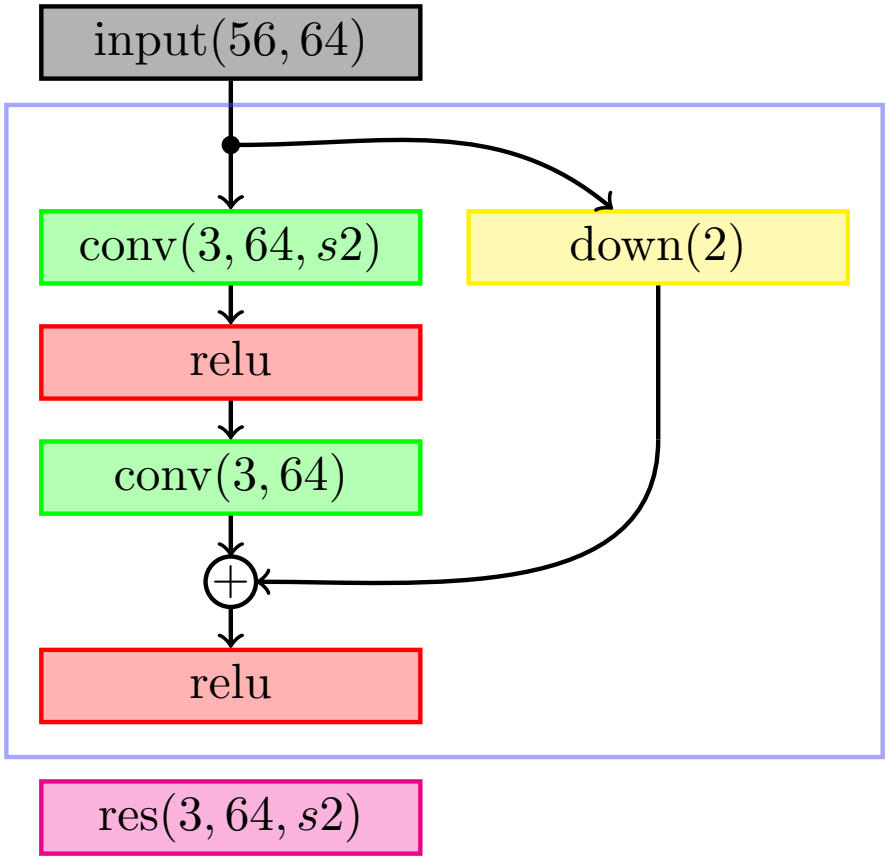
He, Zhang, Ren, Sun. CVPR 2016. Deep Residual Learning for Image Recognition.

residual unit



- residual unit, with a skip connection over the two convolutional layers and the relu between them

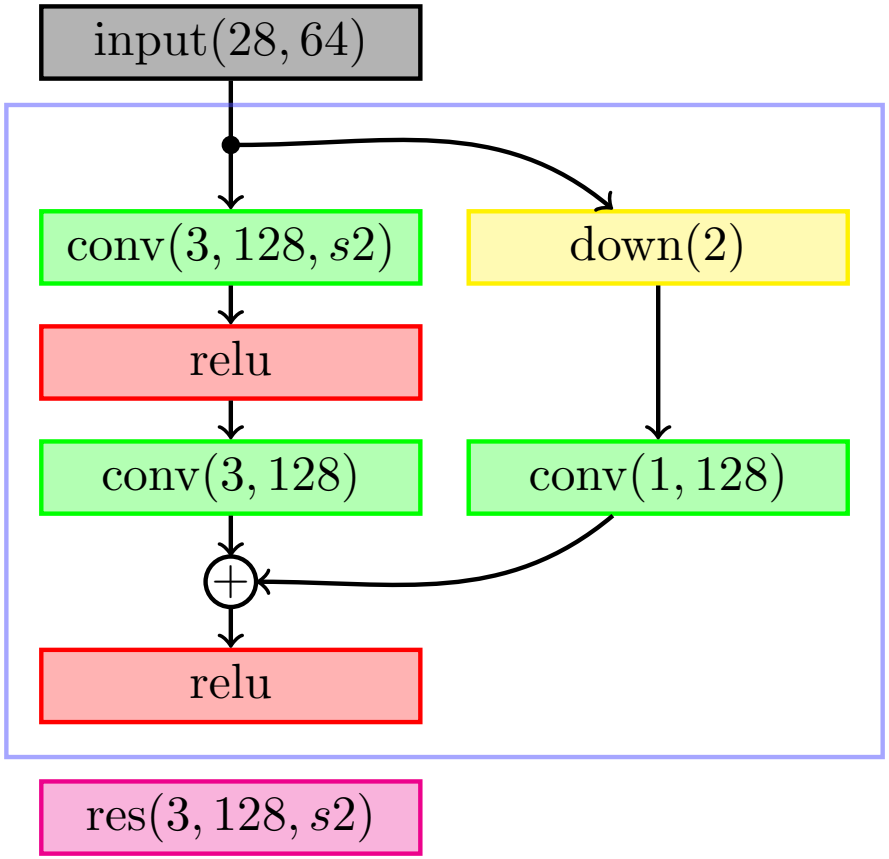
residual unit



- stride 2 in the first convolutional layer, along with downsampling on the skip connection

He, Zhang, Ren, Sun. CVPR 2016. Deep Residual Learning for Image Recognition.

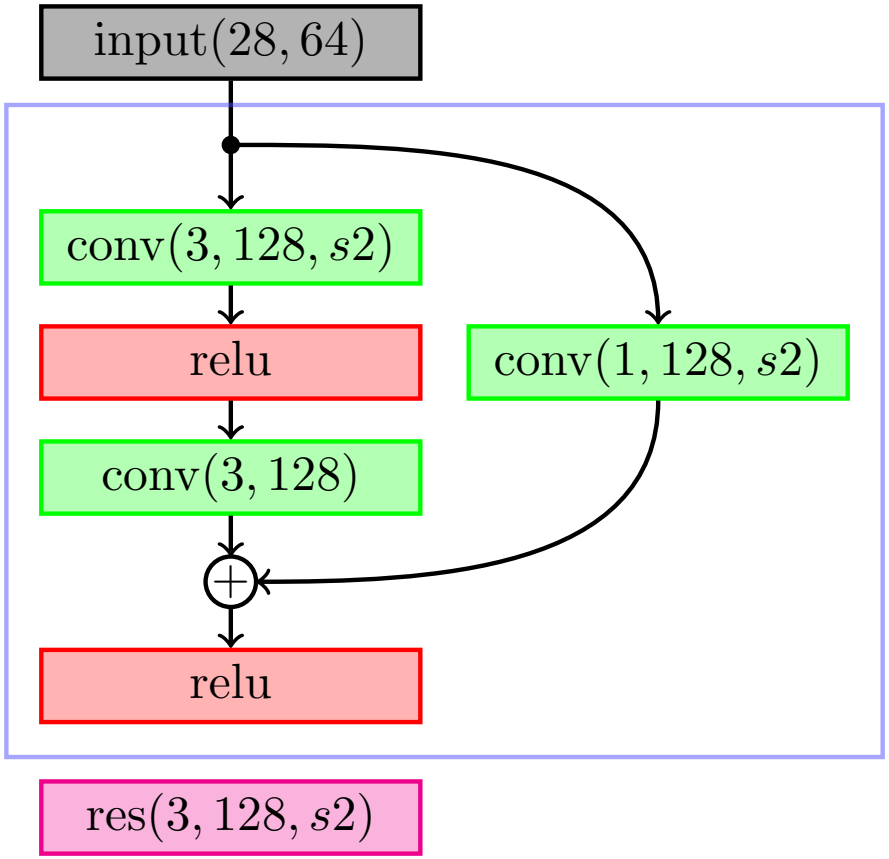
residual unit



- increasing the number of features, along with a 1×1 convolution on the skip connection to project to the new feature space

He, Zhang, Ren, Sun. CVPR 2016. Deep Residual Learning for Image Recognition.

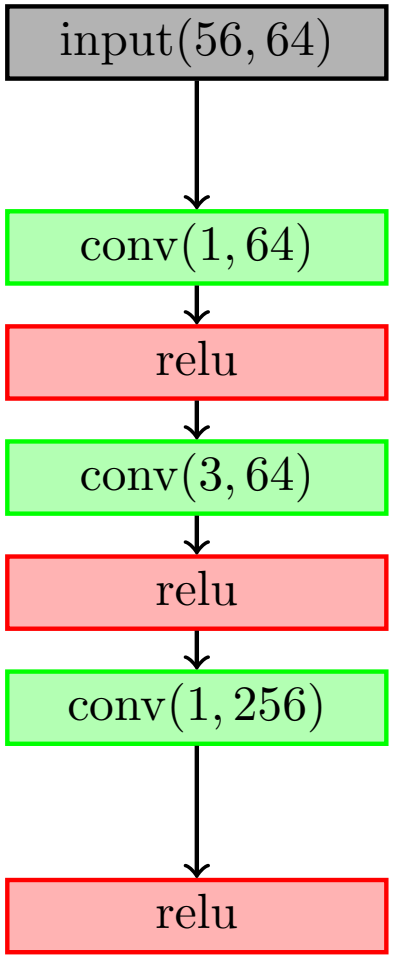
residual unit



- which is the same as a single 1×1 convolution with stride 2, both downsampling and projecting

He, Zhang, Ren, Sun. CVPR 2016. Deep Residual Learning for Image Recognition.

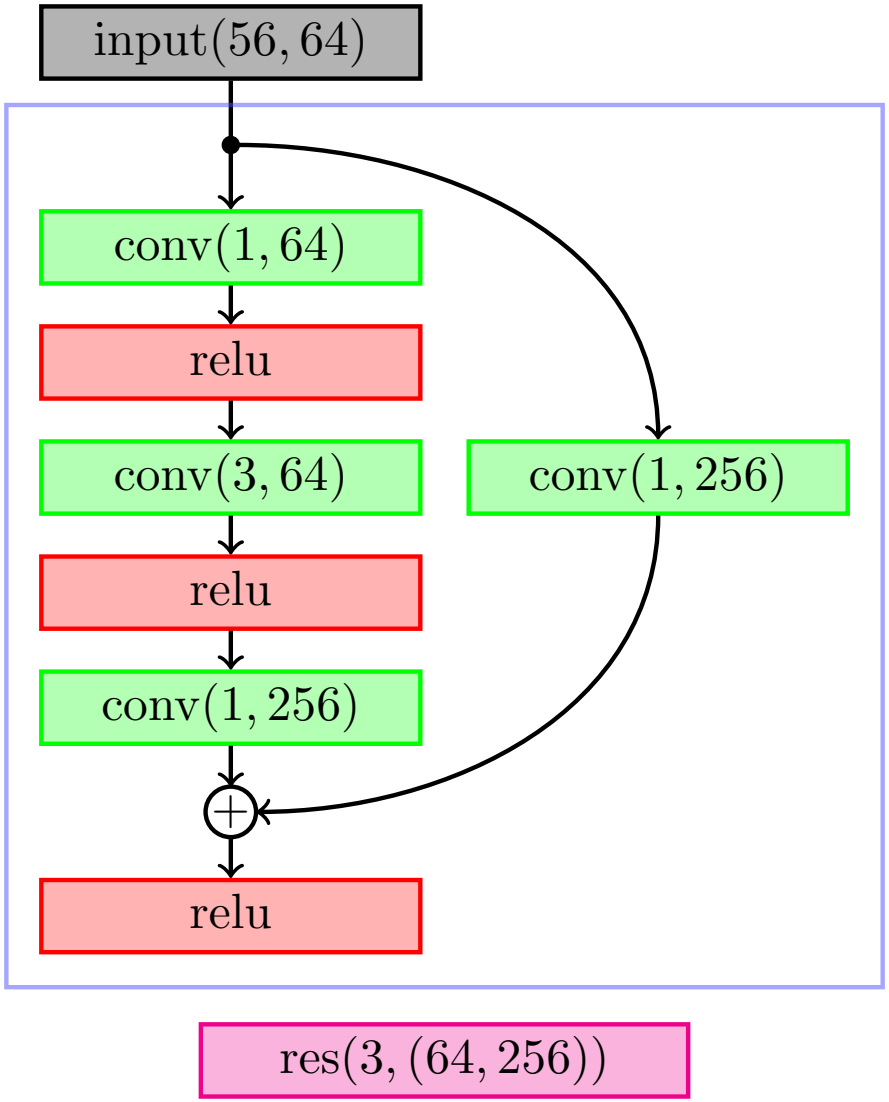
residual bottleneck unit



- "plain" bottleneck unit, with 1×1 convolutions

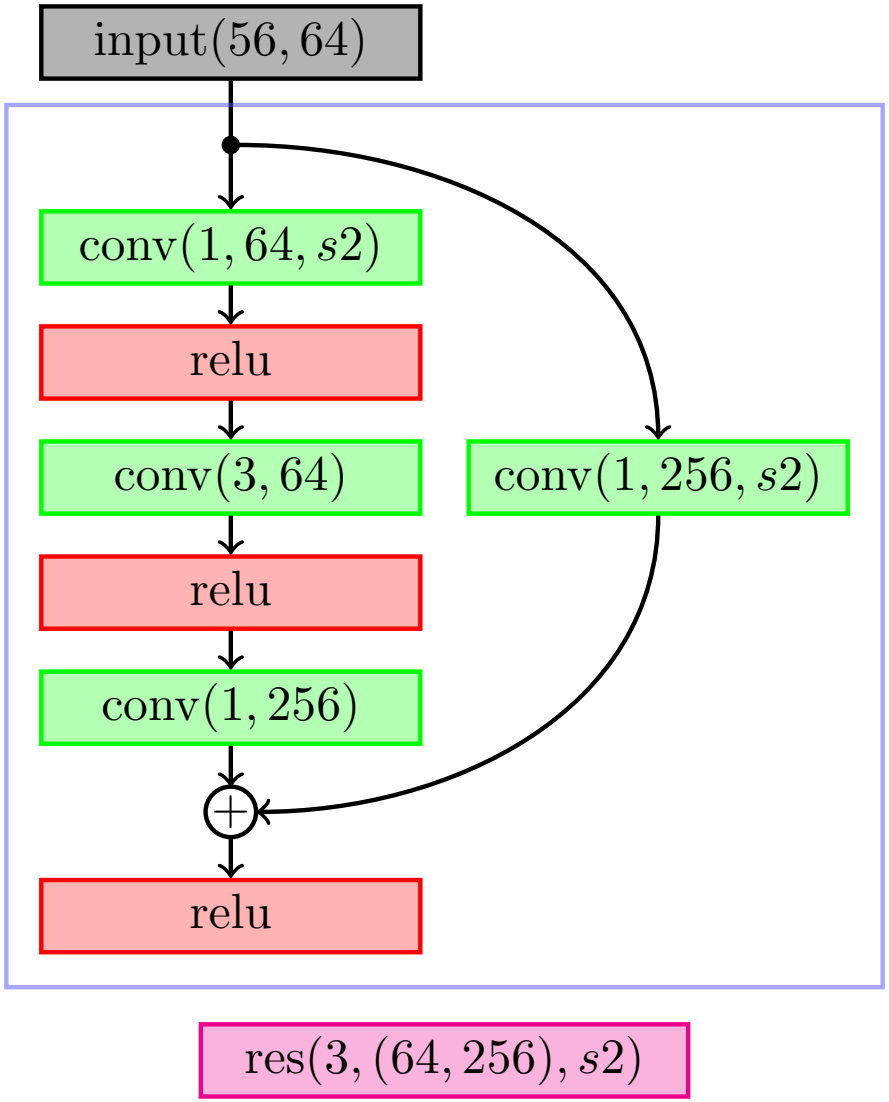
He, Zhang, Ren, Sun. CVPR 2016. Deep Residual Learning for Image Recognition.

residual bottleneck unit



- residual bottleneck unit with a skip connection, always projecting

residual bottleneck unit



- stride 2 in the first convolutional and the skip layer

He, Zhang, Ren, Sun. CVPR 2016. Deep Residual Learning for Image Recognition.

ResNet-34

		parameters	operations	volume
	input$(224, 3)$	0	0	$224 \times 224 \times 3$
	conv$(7, 64, p3, s2)$	$9,472$	$118,816,768$	$112 \times 112 \times 64$
	pool$(3, 2, p1)$	0	$802,816$	$56 \times 56 \times 64$
$3\times$	res$(3, 64)$	$221,568$	$694,837,248$	$56 \times 56 \times 64$
	res$(3, 128, s2)$	$229,760$	$180,182,016$	$28 \times 28 \times 128$
$3\times$	res$(3, 128)$	$885,504$	$694,235,136$	$28 \times 28 \times 128$
	res$(3, 256, s2)$	$918,272$	$180,006,400$	$14 \times 14 \times 256$
$5\times$	res$(3, 256)$	$5,900,800$	$1,156,556,800$	$14 \times 14 \times 256$
	res$(3, 512, s2)$	$3,671,552$	$179,918,592$	$7 \times 7 \times 512$
$2\times$	res$(3, 512)$	$9,439,232$	$462,522,368$	$7 \times 7 \times 512$
	avg(7)	0	$25,088$	512
	fc(1000)	$513,000$	$513,000$	1000
	softmax	0	$1,000$	1000

- $3\times$ more operations but $3\times$ less parameters comparing to AlexNet

He, Zhang, Ren, Sun. CVPR 2016. Deep Residual Learning for Image Recognition.

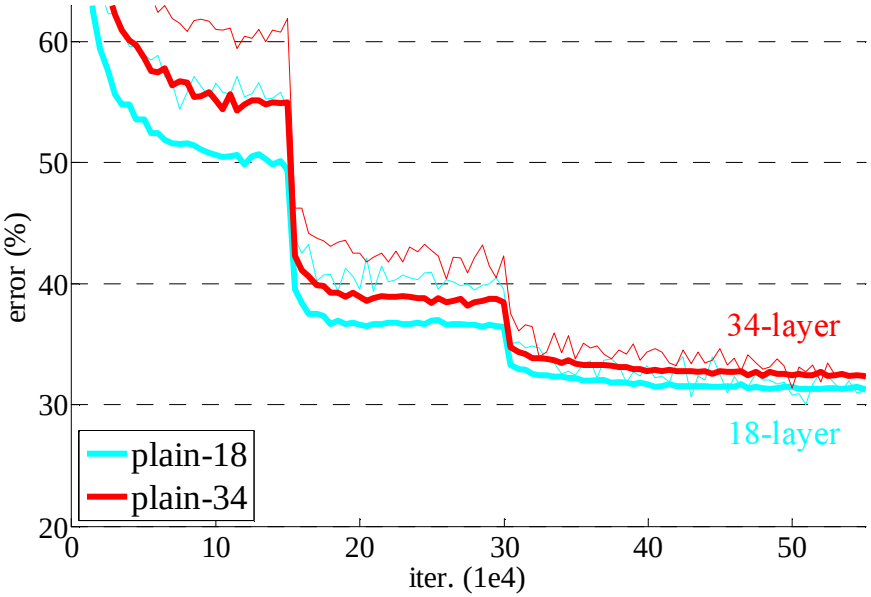
ResNet-101

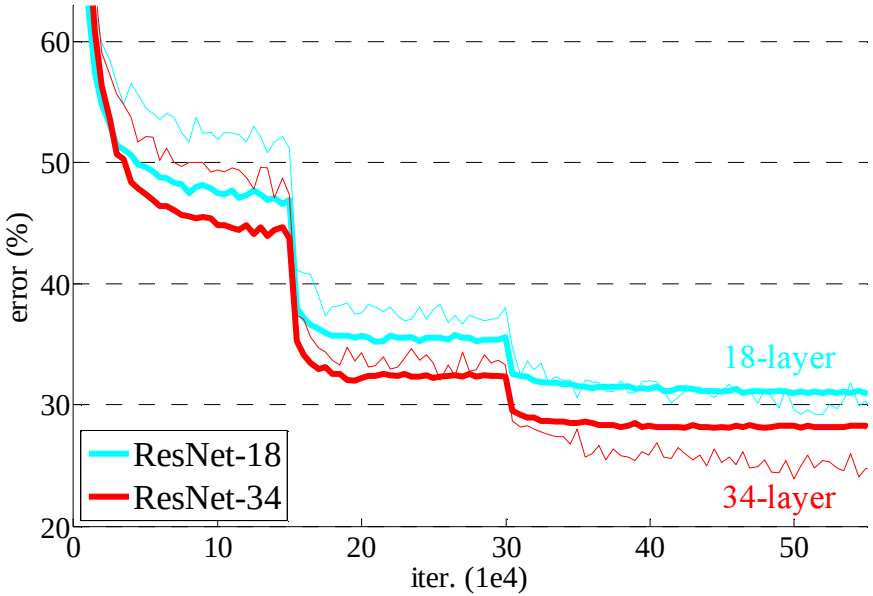
		parameters	operations	volume
	input$(224, 3)$	0	0	$224 \times 224 \times 3$
	conv$(7, 64, p3, s2)$	$9,472$	$118,816,768$	$112 \times 112 \times 64$
	pool$(3, 2, p1)$	0	$802,816$	$56 \times 56 \times 64$
$3\times$	res$(3, (64, 256))$	$214,400$	$672,358,400$	$56 \times 56 \times 256$
	res$(3, (128, 512), s2)$	$378,112$	$296,640,512$	$28 \times 28 \times 512$
$3\times$	res$(3, (128, 512))$	$837,888$	$656,904,192$	$28 \times 28 \times 512$
	res$(3, (256, 1024), s2)$	$1,509,888$	$296,038,400$	$14 \times 14 \times 1024$
$22\times$	res$(3, (256, 1024))$	$24,544,256$	$4,810,674,176$	$14 \times 14 \times 1024$
	res$(3, (512, 2048), s2)$	$6,034,432$	$295,737,344$	$7 \times 7 \times 2048$
$2\times$	res$(3, (512, 2048))$	$8,919,040$	$437,032,960$	$7 \times 7 \times 2048$
	avg(7)	0	$100,352$	2048
	fc(1000)	$2,049,000$	$2,049,000$	1000
	softmax	0	$1,000$	1000

- $7\times$ more operations but $1.5\times$ less parameters comparing to AlexNet

He, Zhang, Ren, Sun. CVPR 2016. Deep Residual Learning for Image Recognition.

ResNet-34: ImageNet





- a plain network exhibits degradation with increasing depth
- while a residual network gains from increasing depth

He, Zhang, Ren, Sun. CVPR 2016. Deep Residual Learning for Image Recognition.

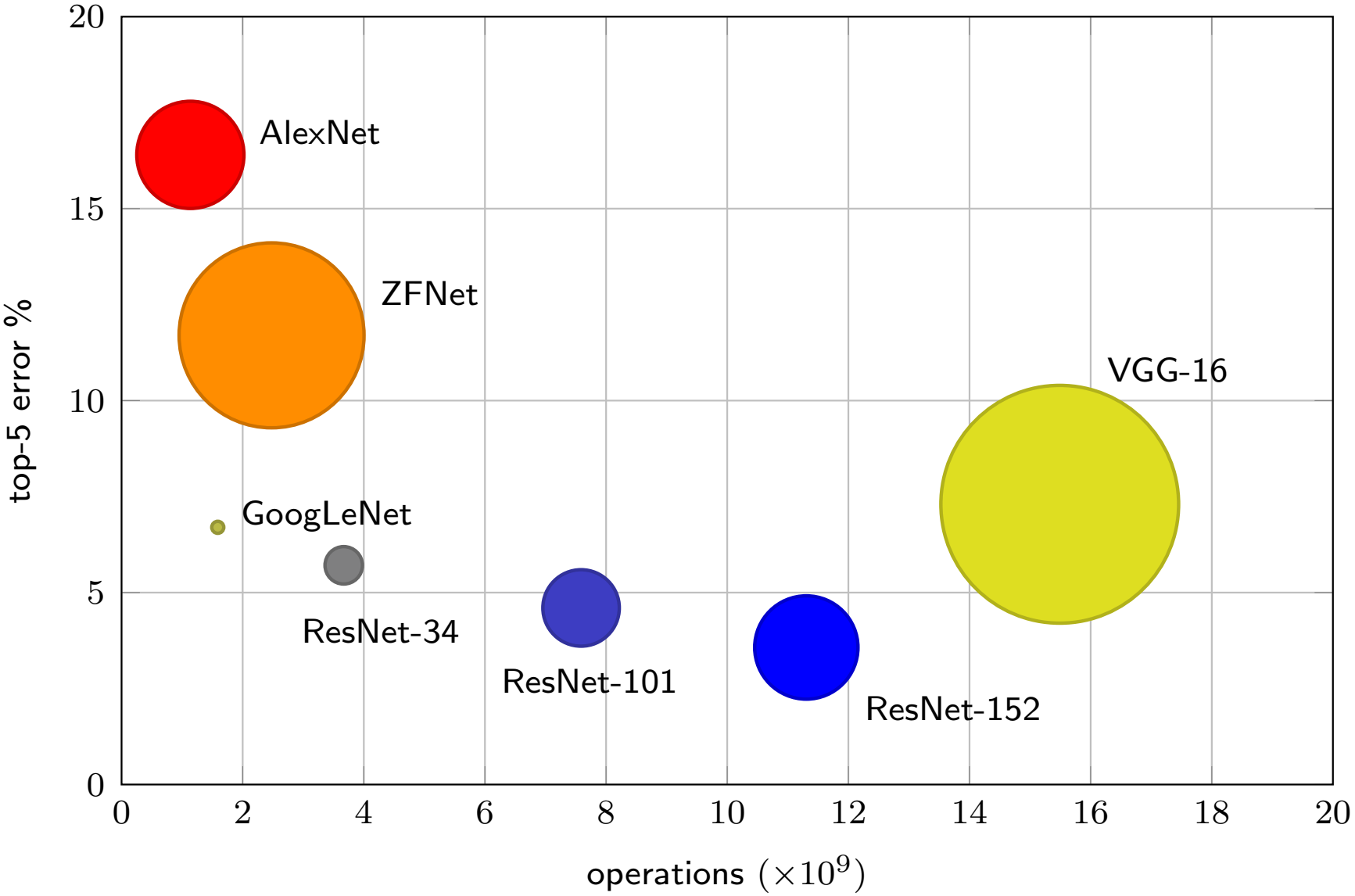
ResNet models

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

- downsampling by 2 at layers conv3_1, conv4_1, conv5_1

He, Zhang, Ren, Sun. CVPR 2016. Deep Residual Learning for Image Recognition.

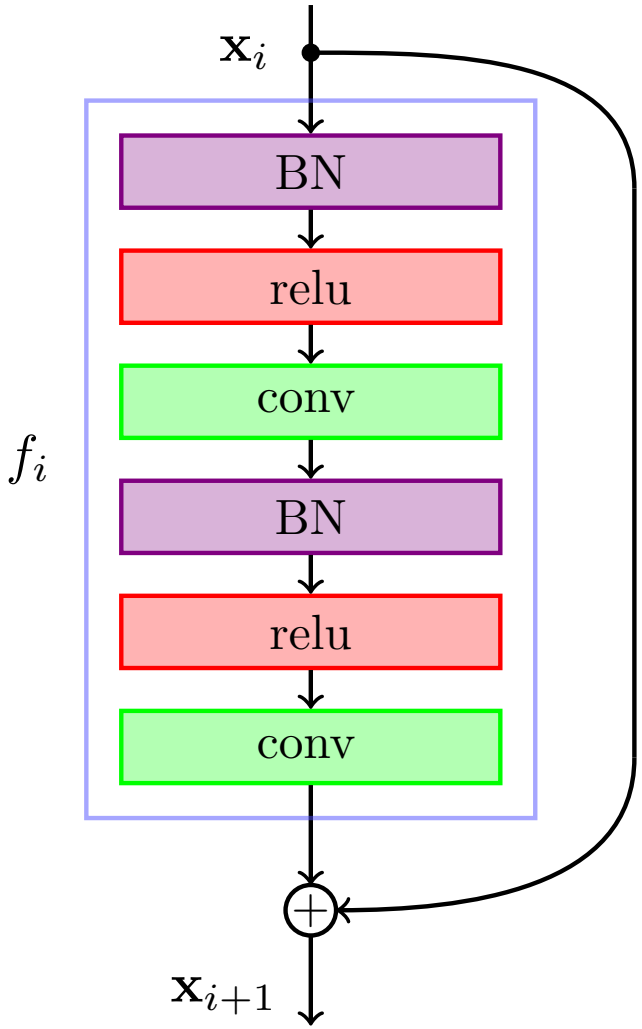
network performance



Canziani, Culurciello and Paszke. 2016. An Analysis of Deep Neural Network Models for Practical Applications.

densely connected networks

[Huang et al. 2017]



- residual unit with identity mapping: add

$$\mathbf{x}_{i+1} = \mathbf{x}_i + f_i(\mathbf{x}_i)$$

- densely connected unit: concatenate

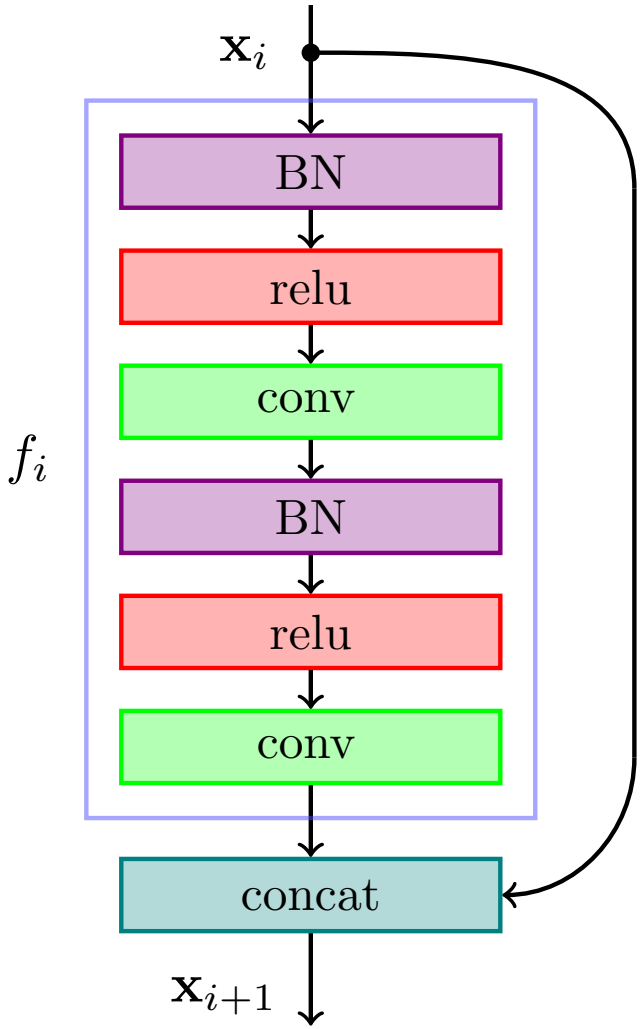
$$\mathbf{x}_{i+1} = (\mathbf{x}_i, f_i(\mathbf{x}_i))$$

- feature map dimension increases by growth rate k at each unit
- a dense block is a chain of densely connected units
- a transition layer reduces feature map dimension by a factor $\theta = 2$

Huang, Liu, van der Maaten and Weinberger. CVPR 2017. Densely Connected Convolutional Networks.

densely connected networks

[Huang et al. 2017]



- residual unit with identity mapping: add

$$\mathbf{x}_{i+1} = \mathbf{x}_i + f_i(\mathbf{x}_i)$$

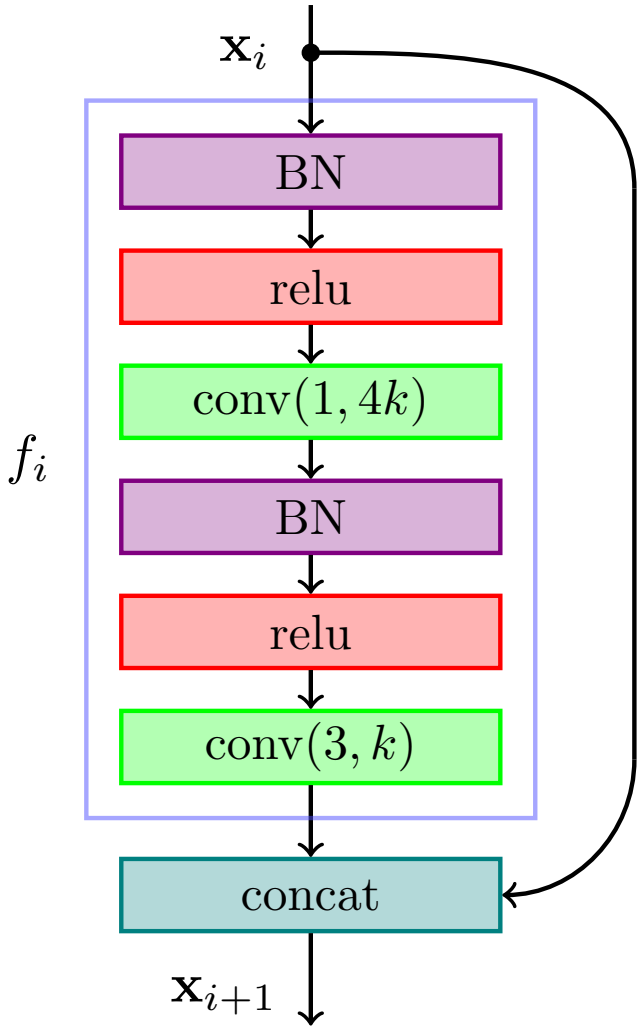
- densely connected unit: concatenate

$$\mathbf{x}_{i+1} = (\mathbf{x}_i, f_i(\mathbf{x}_i))$$

- feature map dimension increases by growth rate k at each unit
- a dense block is a chain of densely connected units
- a transition layer reduces feature map dimension by a factor $\theta = 2$

densely connected networks

[Huang et al. 2017]



- residual unit with identity mapping: add

$$\mathbf{x}_{i+1} = \mathbf{x}_i + f_i(\mathbf{x}_i)$$

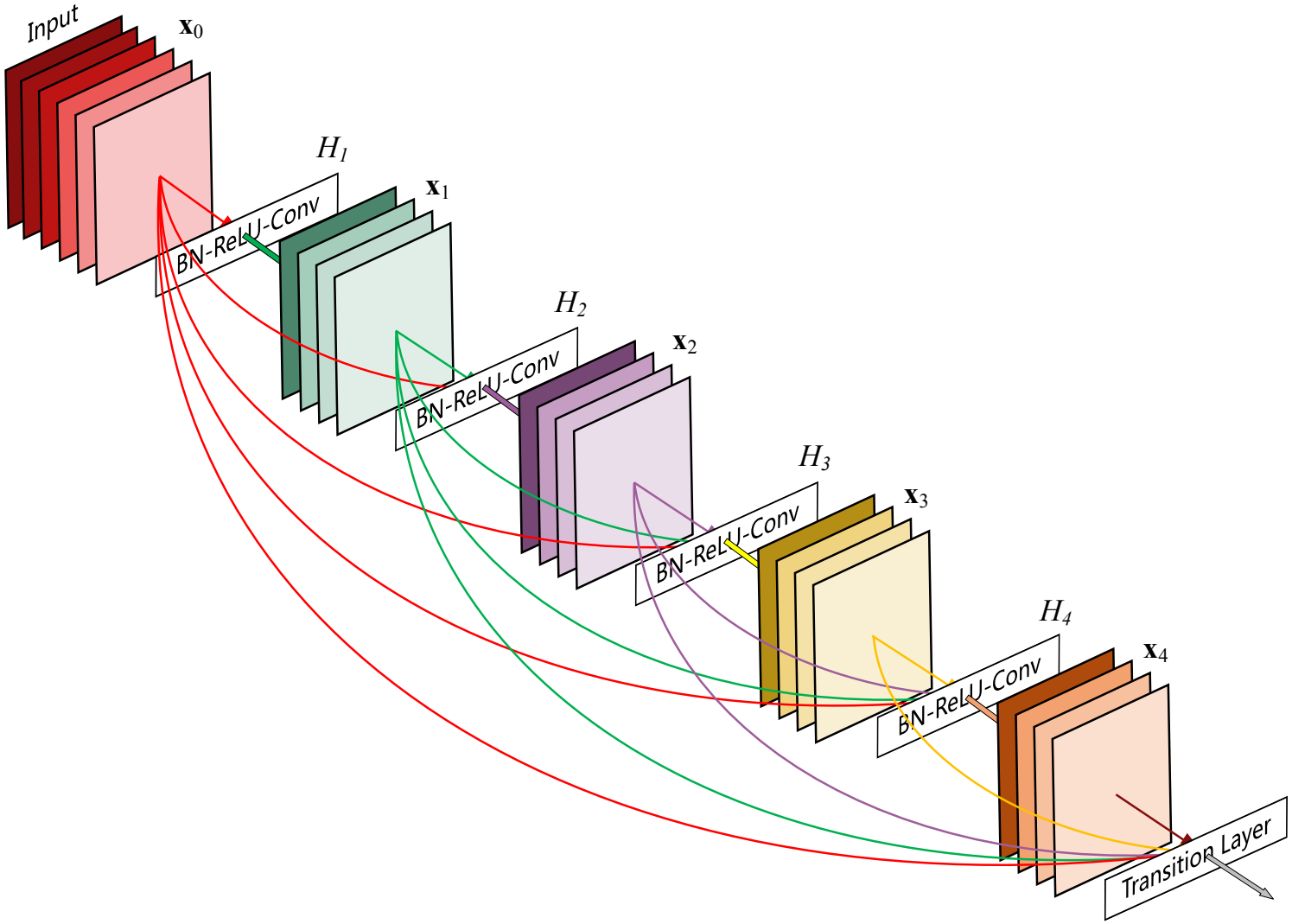
- densely connected unit: concatenate

$$\mathbf{x}_{i+1} = (\mathbf{x}_i, f_i(\mathbf{x}_i))$$

- feature map dimension increases by growth rate k at each unit
- a dense block is a chain of densely connected units
- a transition layer reduces feature map dimension by a factor $\theta = 2$

Huang, Liu, van der Maaten and Weinberger. CVPR 2017. Densely Connected Convolutional Networks.

densely connected networks



- dense block followed by transition layer

Huang, Liu, van der Maaten and Weinberger. CVPR 2017. Densely Connected Convolutional Networks.

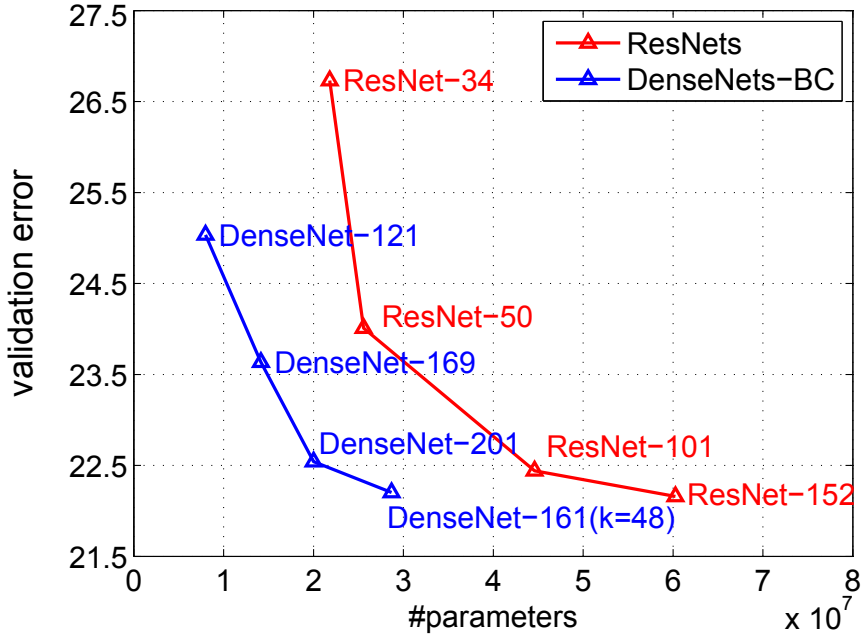
DenseNet models

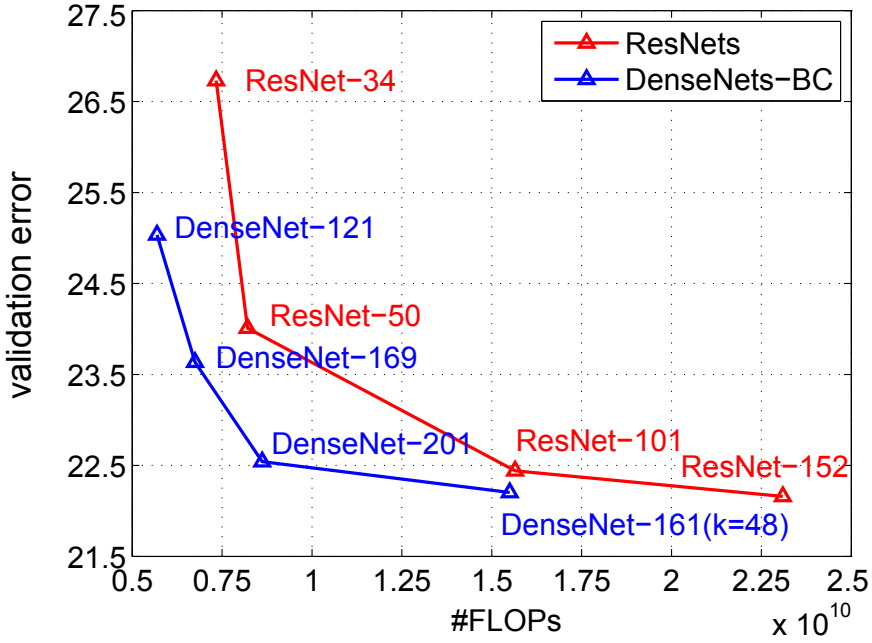
Layers	Output Size	DenseNet-121($k = 32$)	DenseNet-169($k = 32$)	DenseNet-201($k = 32$)	DenseNet-161($k = 48$)
Convolution	112×112	7×7 conv, stride 2			
Pooling	56×56	3×3 max pool, stride 2			
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
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	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 36$
Transition Layer (3)	14×14	1×1 conv			
	7×7	2×2 average pool, stride 2			
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$
Classification Layer	1×1	7×7 global average pool			
		1000D fully-connected, softmax			

- input is 224×224; first convolutional layer produces $2k$ features; transition layer reduces dimension and resolution by 2

Huang, Liu, van der Maaten and Weinberger. CVPR 2017. Densely Connected Convolutional Networks.

DenseNet vs. ResNet: ImageNet

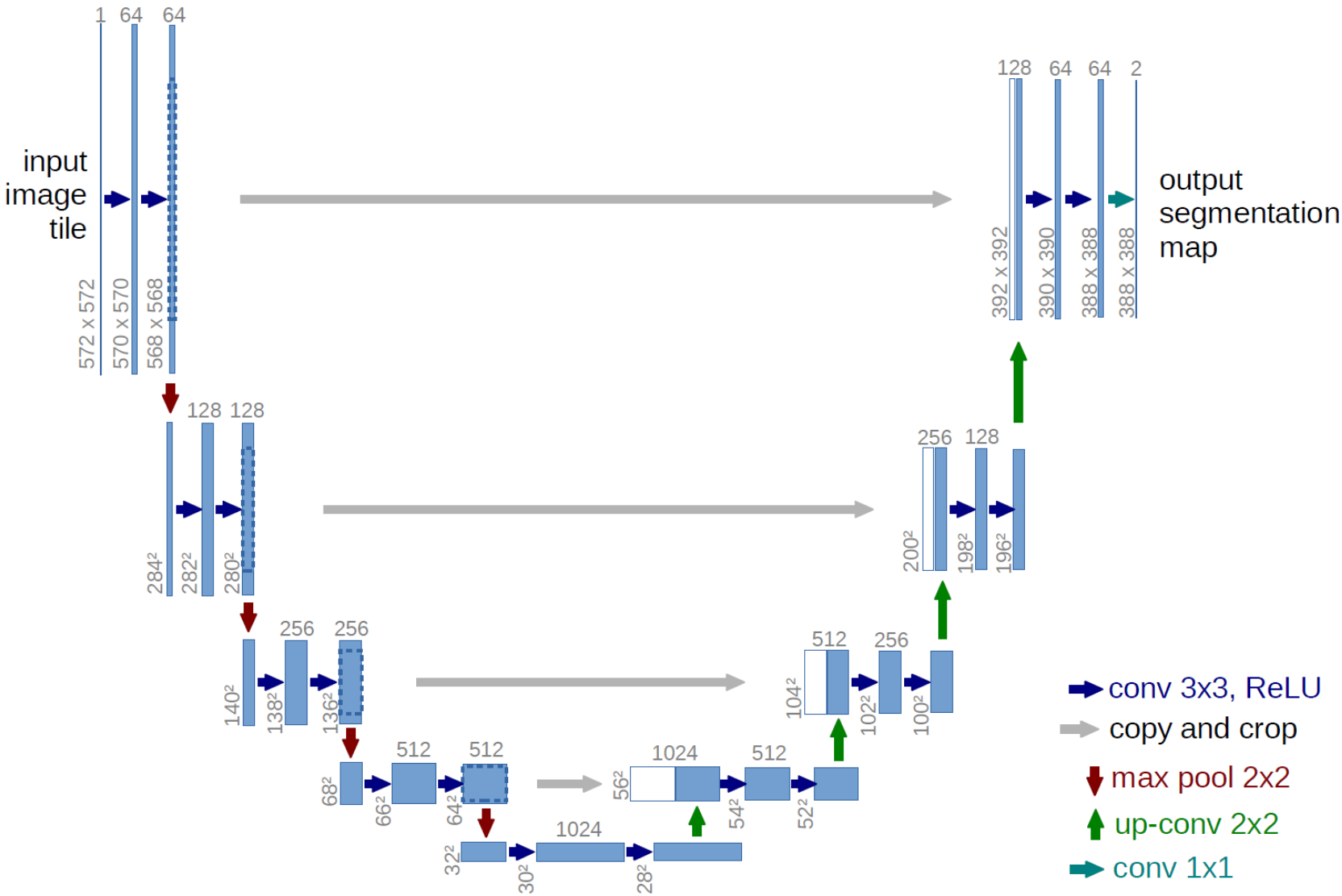




- top-1 single-crop ImageNet validation error
- encourages feature re-use and reduces the number of parameters

Huang, Liu, van der Maaten and Weinberger. CVPR 2017. Densely Connected Convolutional Networks.

U-net: a segmentation convolutional network


input image tile
output segmentation map
1 64 64
572 x 572
570 x 570
568 x 568
128 128
284²
282²
280²
256 256
140²
138²
136²
512 512
68²
66²
64²
1024
32²
30²
28²
1024 512
56²
54²
52²
512 256
104²
102²
100²
256 128
200²
198²
196²
128 64 64 2
392 x 392
390 x 390
388 x 388
388 x 388
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

Conclusion

Convolution: filter size (1 and rest), stride, padding.
Convolution and fully connected layers.
Depth and vanishing gradient
Multi-scale information
Re-use information in early layers
Parameters, memory, and operations