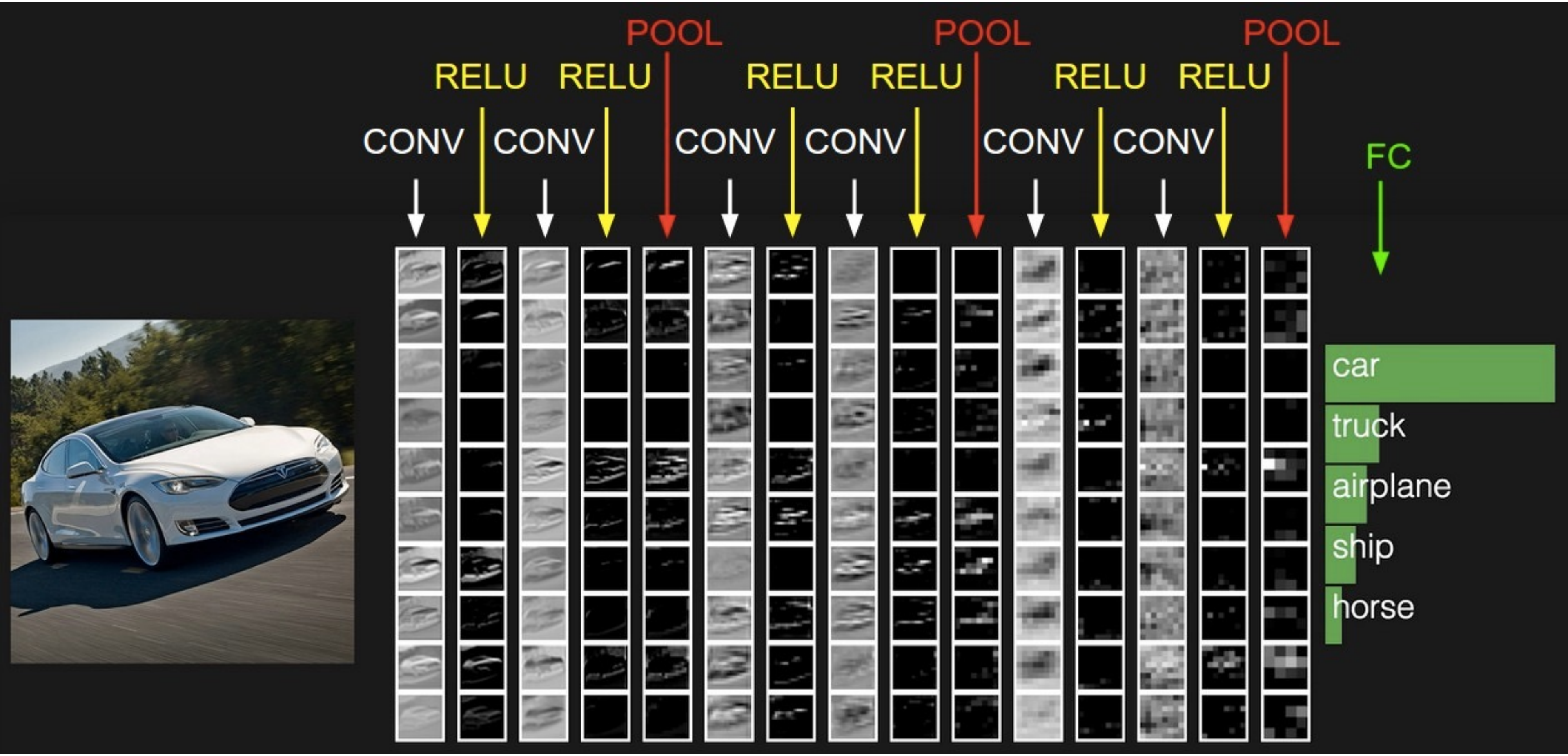
Convolutional neural networks

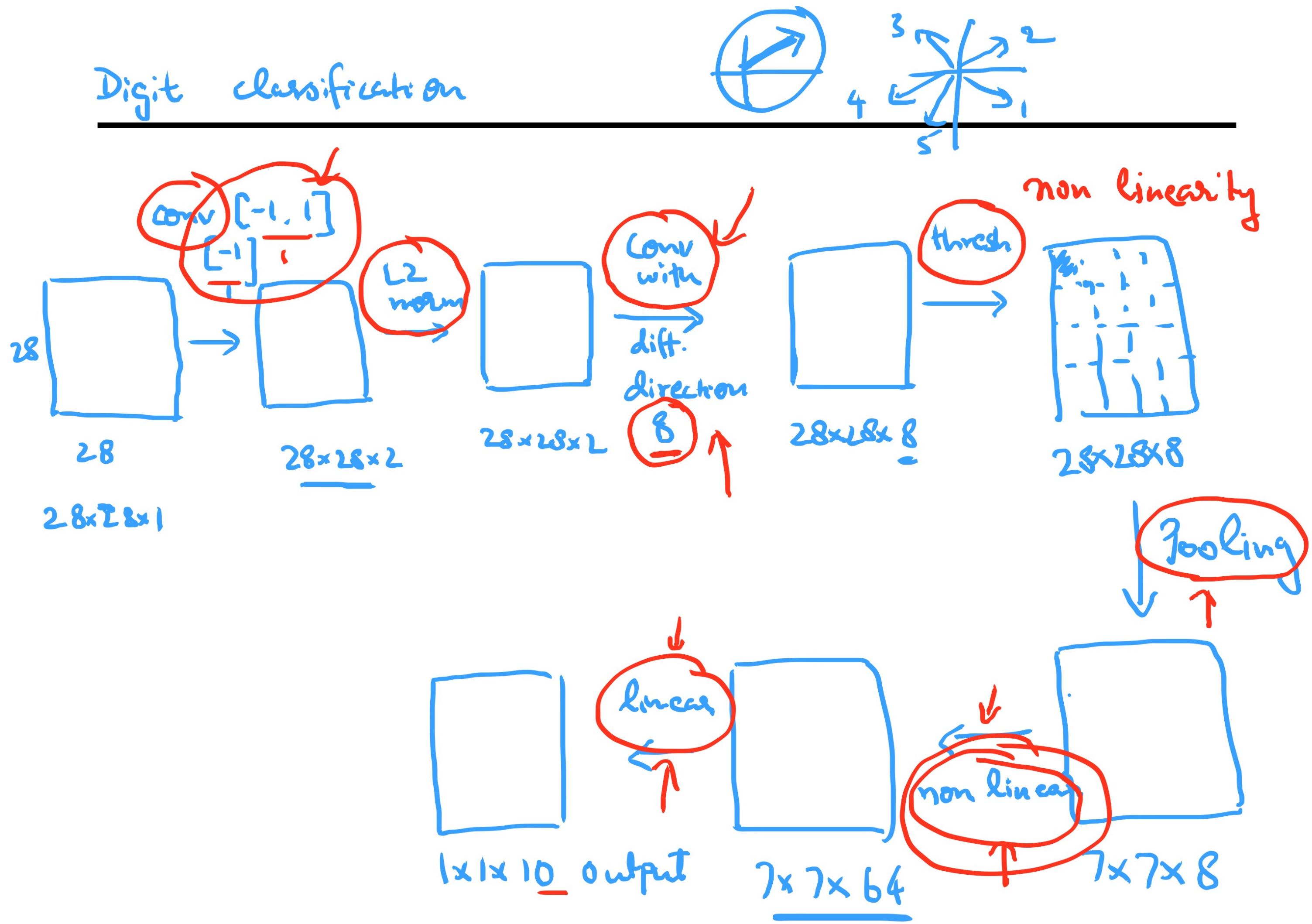


Slides from L. Lazebnik, Rob Fergus, Andrej Karpathy

Outline

- Building blocks for CNNs
- Motivation and history
- Alexnet
- Since Alexnet

Compare: Digit Classification using SVMs

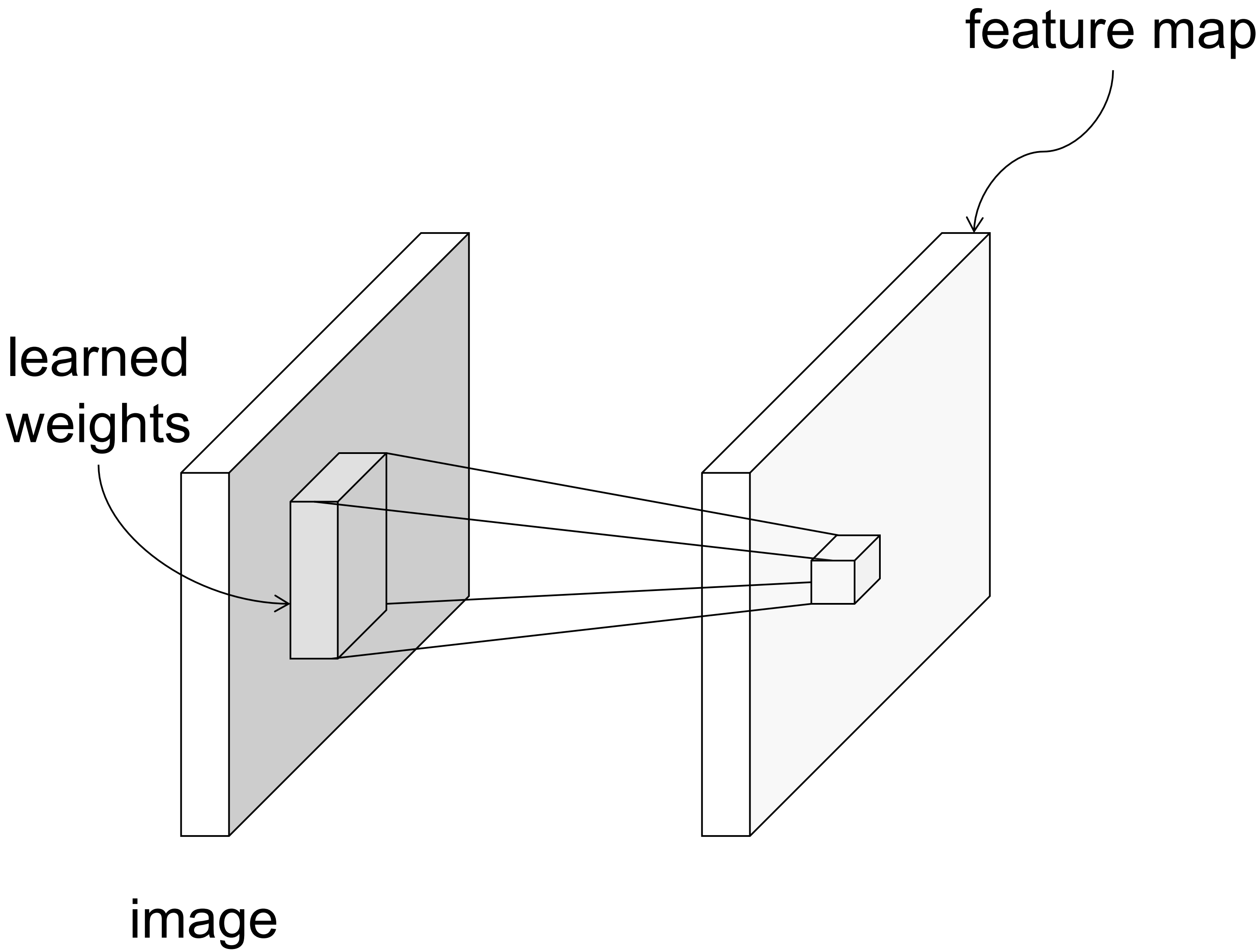


Components of a CNN architecture

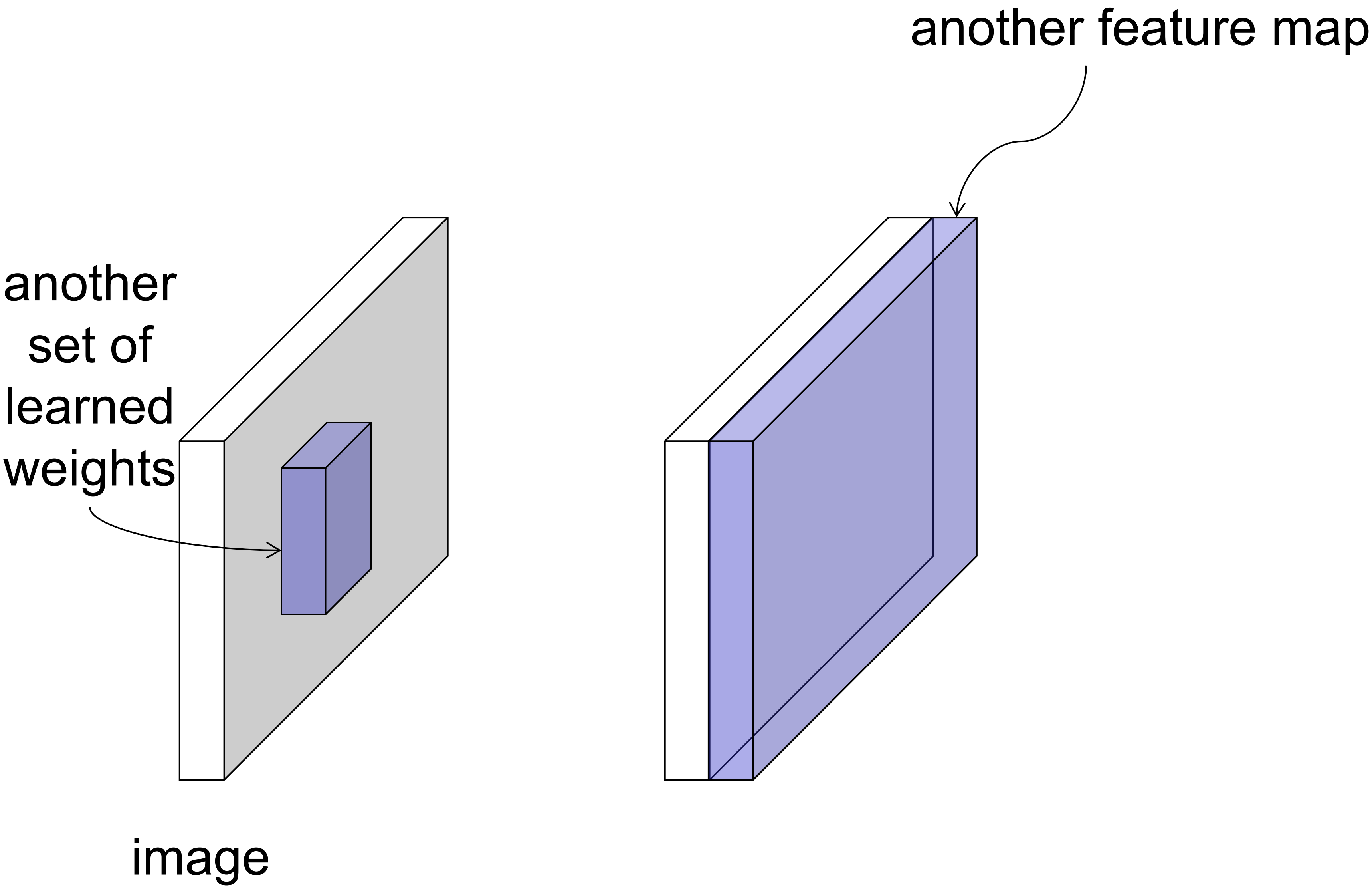
- Convolutional Layers
- Non-linearities
- Pooling
- Fully-connected Layers
- Normalization Layers

Rationale?

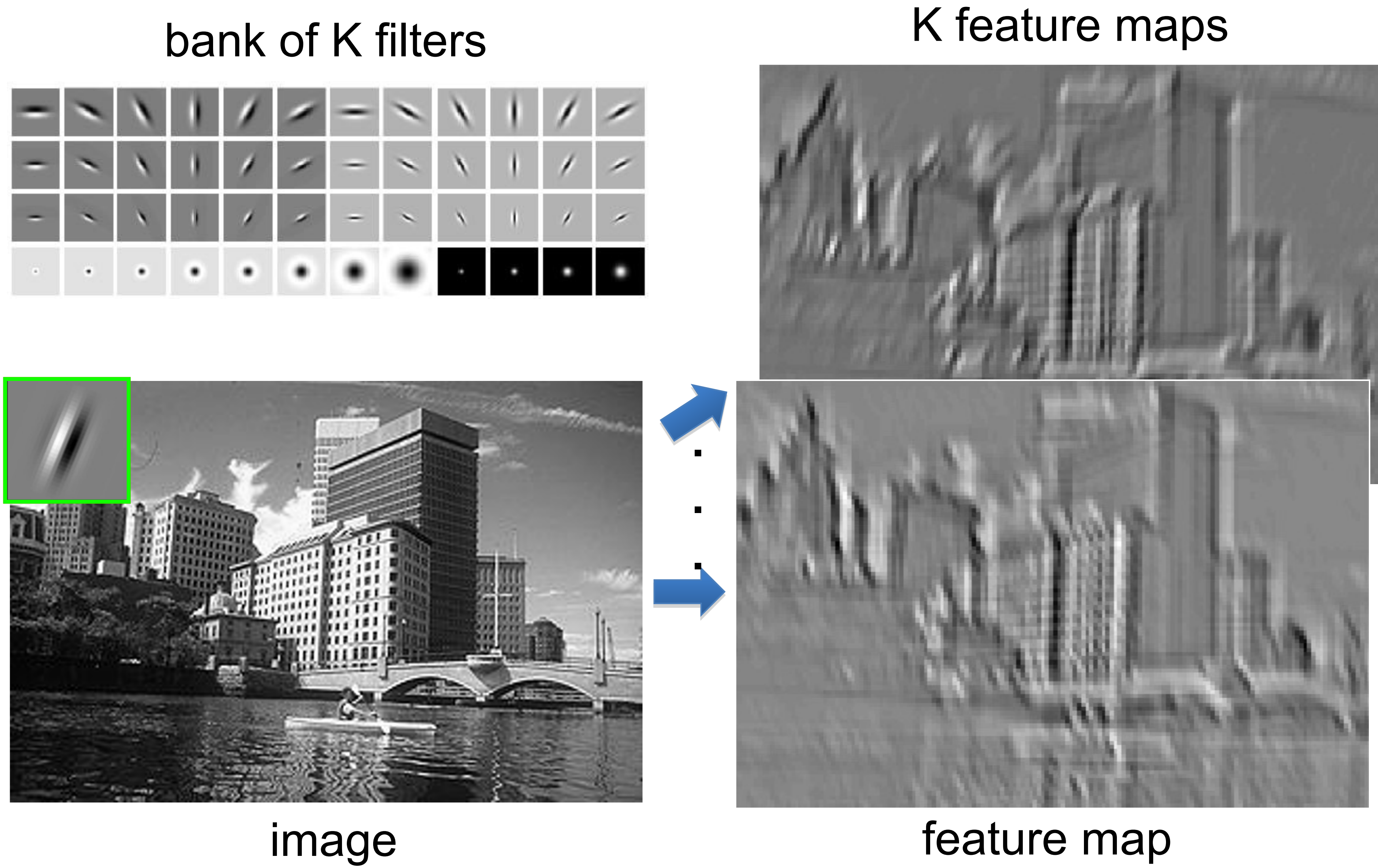
Neural networks for images



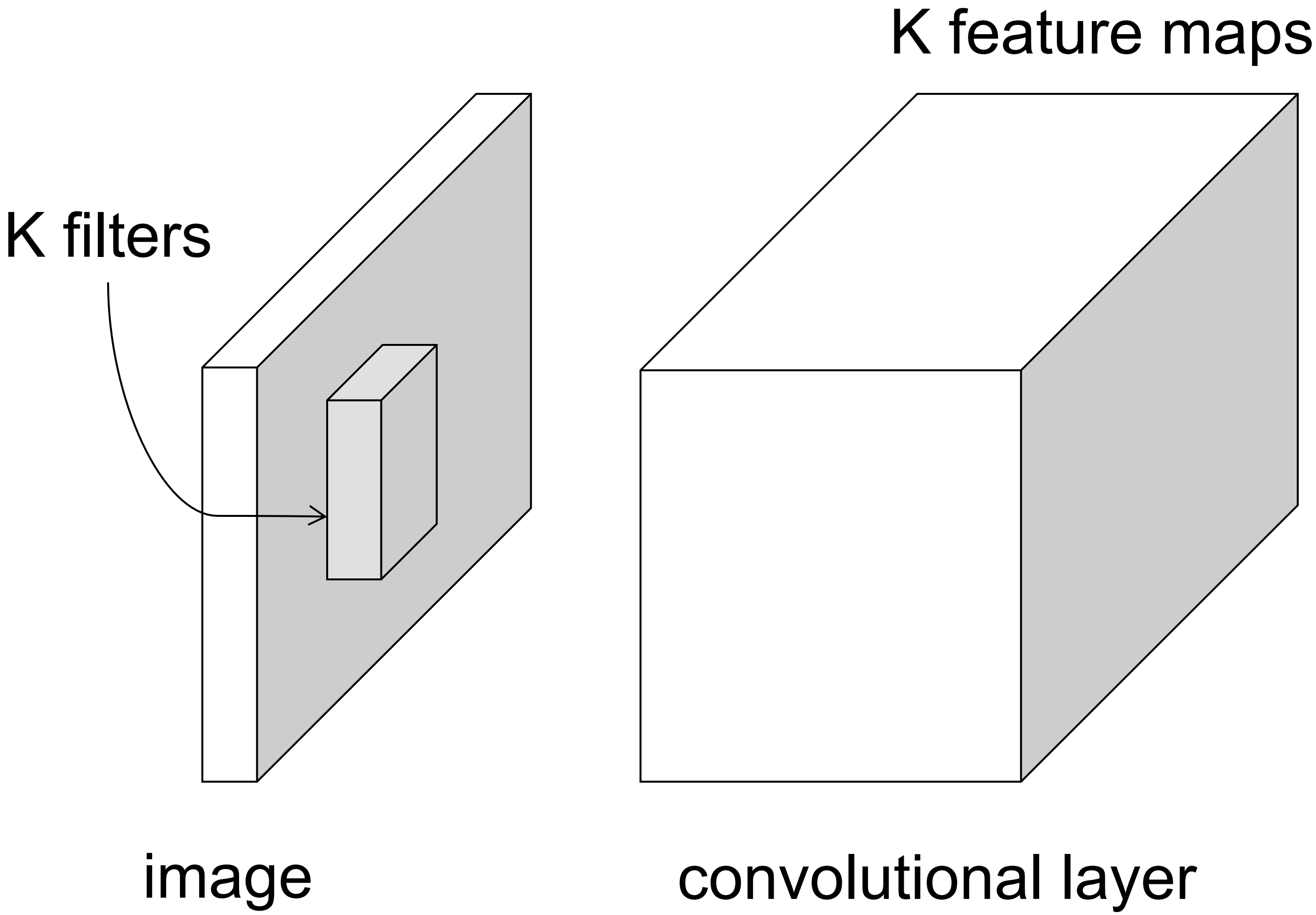
Neural networks for images



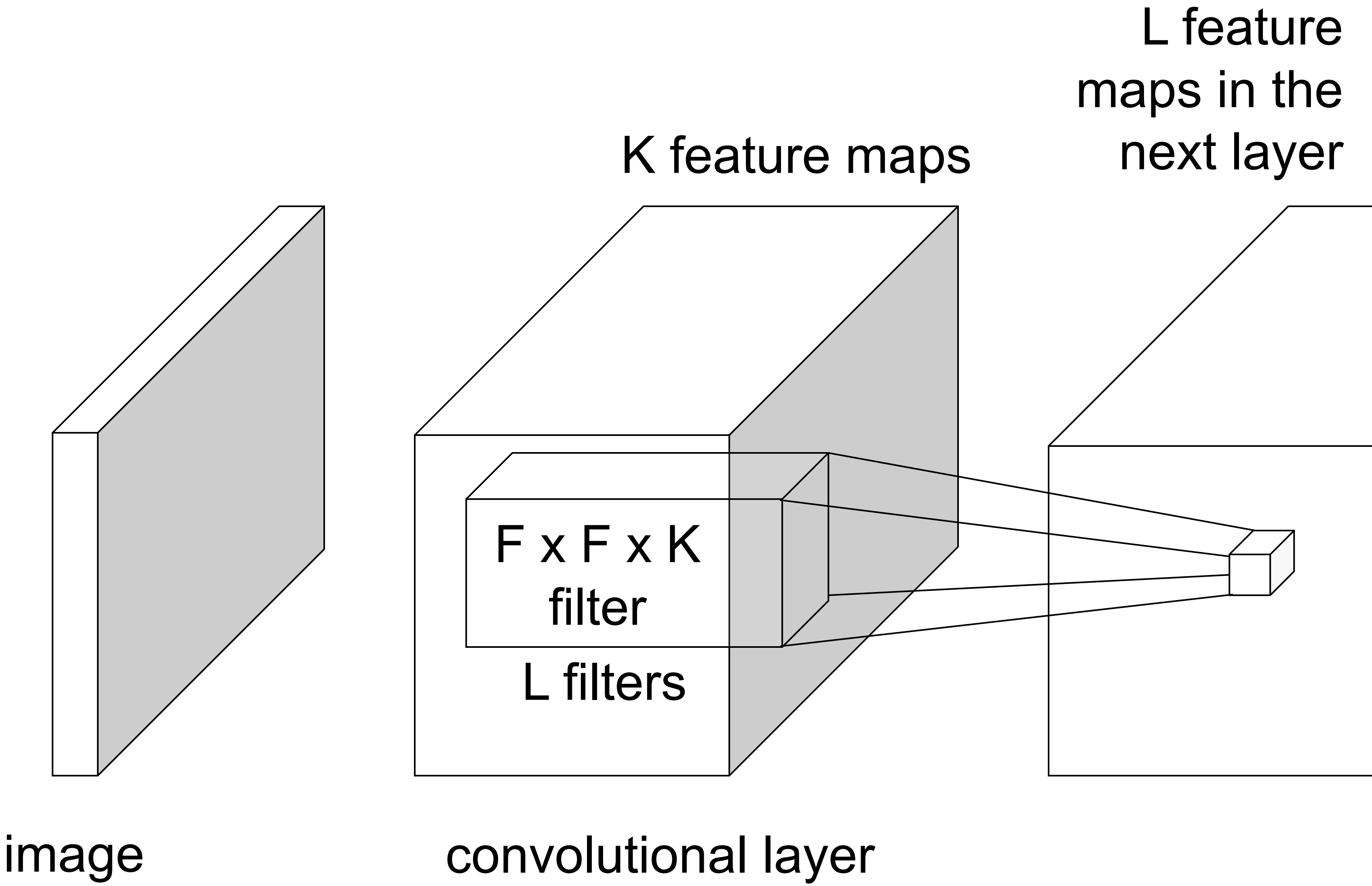
Convolution as feature extraction



Convolutional layer

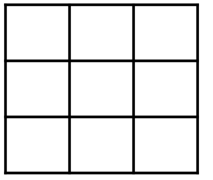


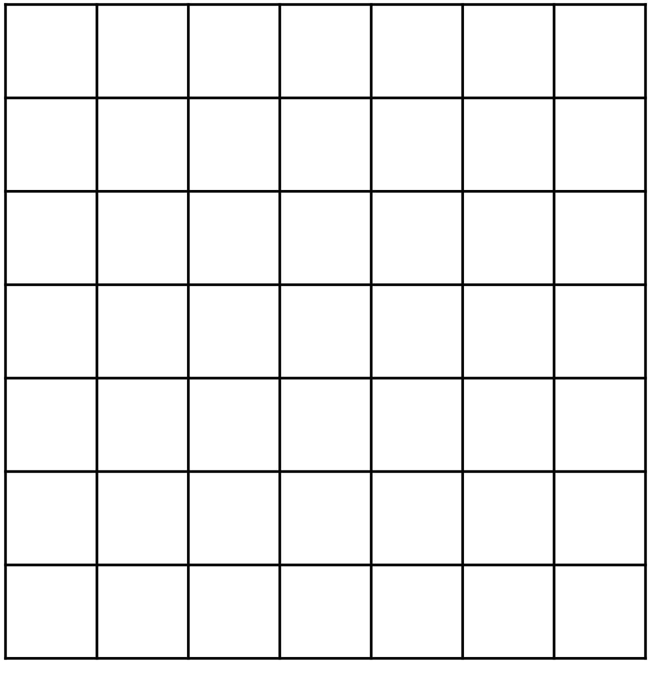
Convolutional layer



Convolutional layer

- Input
- Convolutional Hyper-Parameters
    - Kernel Size
    - Number of Filters
    - Padding
    - Stride
- Parameters
    - Weights
    - Biases
- Output Size





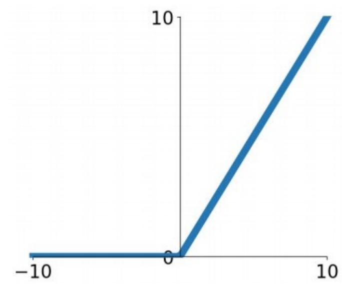
Components of a CNN architecture

- Convolutional Layers
- Non-linearities
- Pooling
- Fully-connected Layers
- Normalization Layers

Non-Linearities

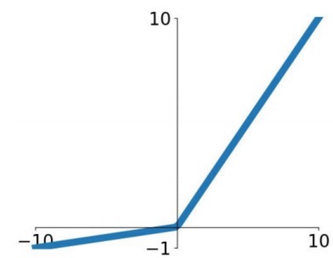
ReLU

$$\max(0, x)$$



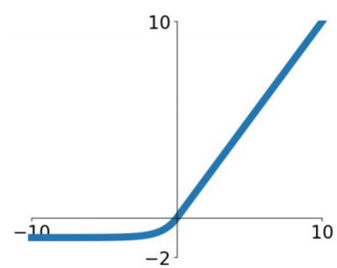
Leaky ReLU

$$\max(0.1x, x)$$



ELU

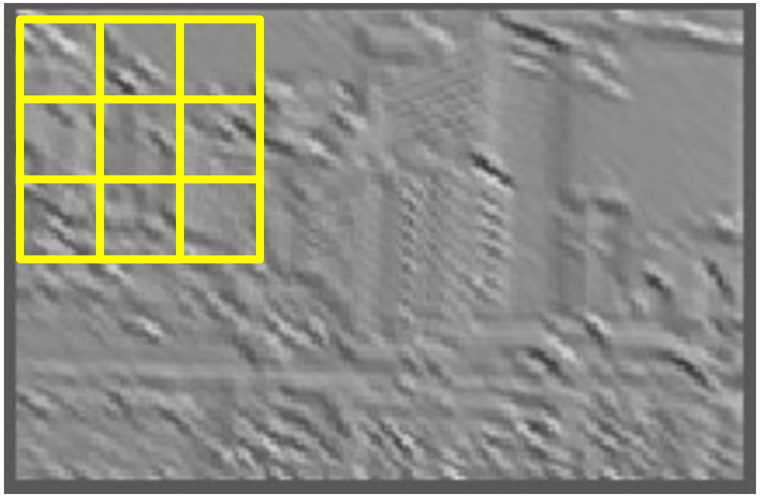
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



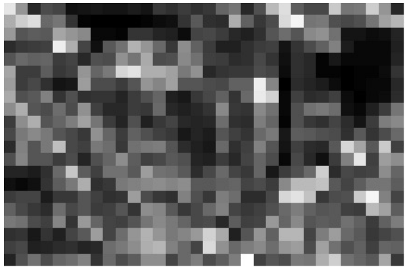
Source: R. Fergus, Y. LeCun

Source: [Stanford 231n]

Pooling Layers

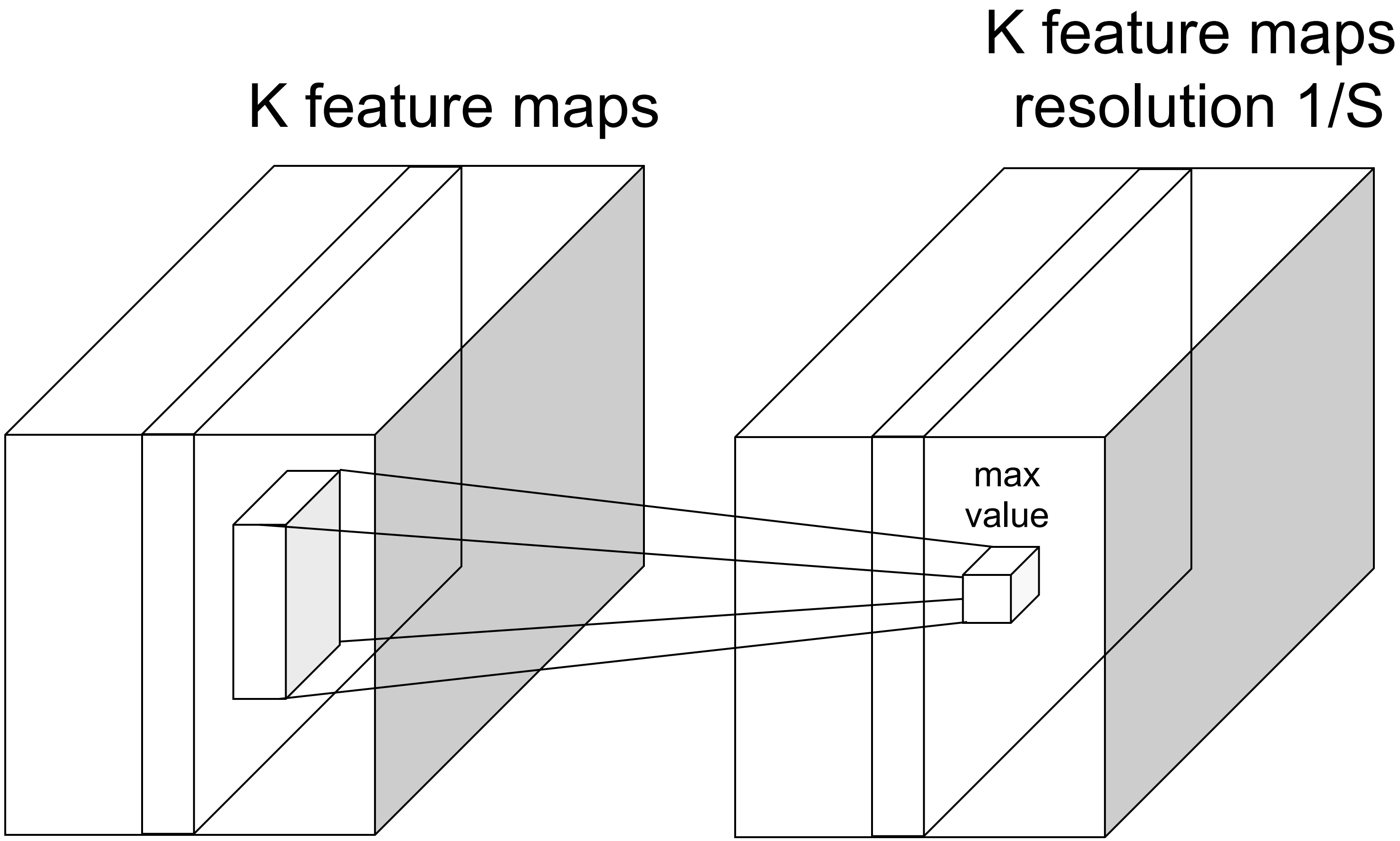


Max (or Avg)



Source: R. Fergus, Y. LeCun

Pooling Layers

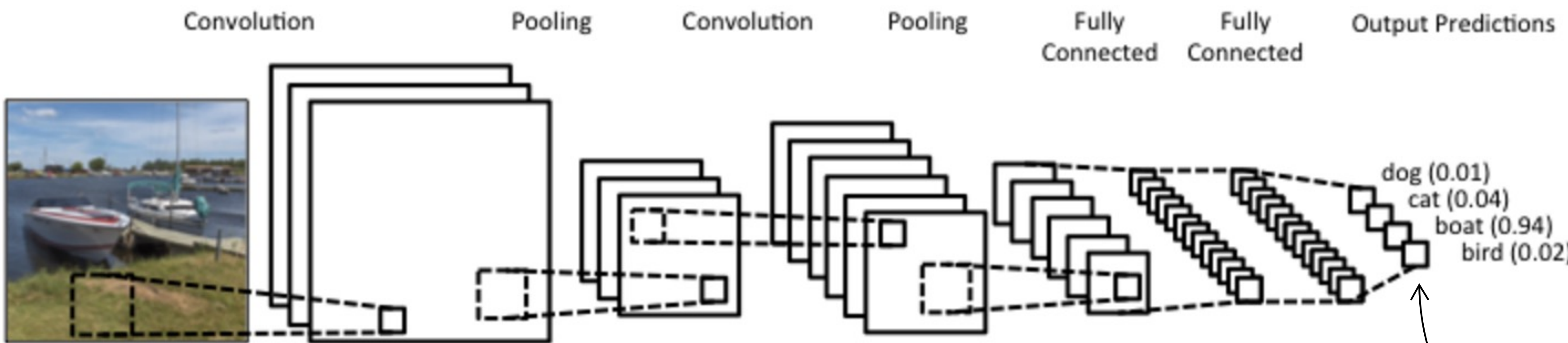


F x F pooling filter,
stride S
Usually: F=2 or 3, S=2

Components of a CNN architecture

- Convolutional Layers
- Non-linearities
- Pooling
- Fully-connected Layers
- Normalization Layers (in just a bit)

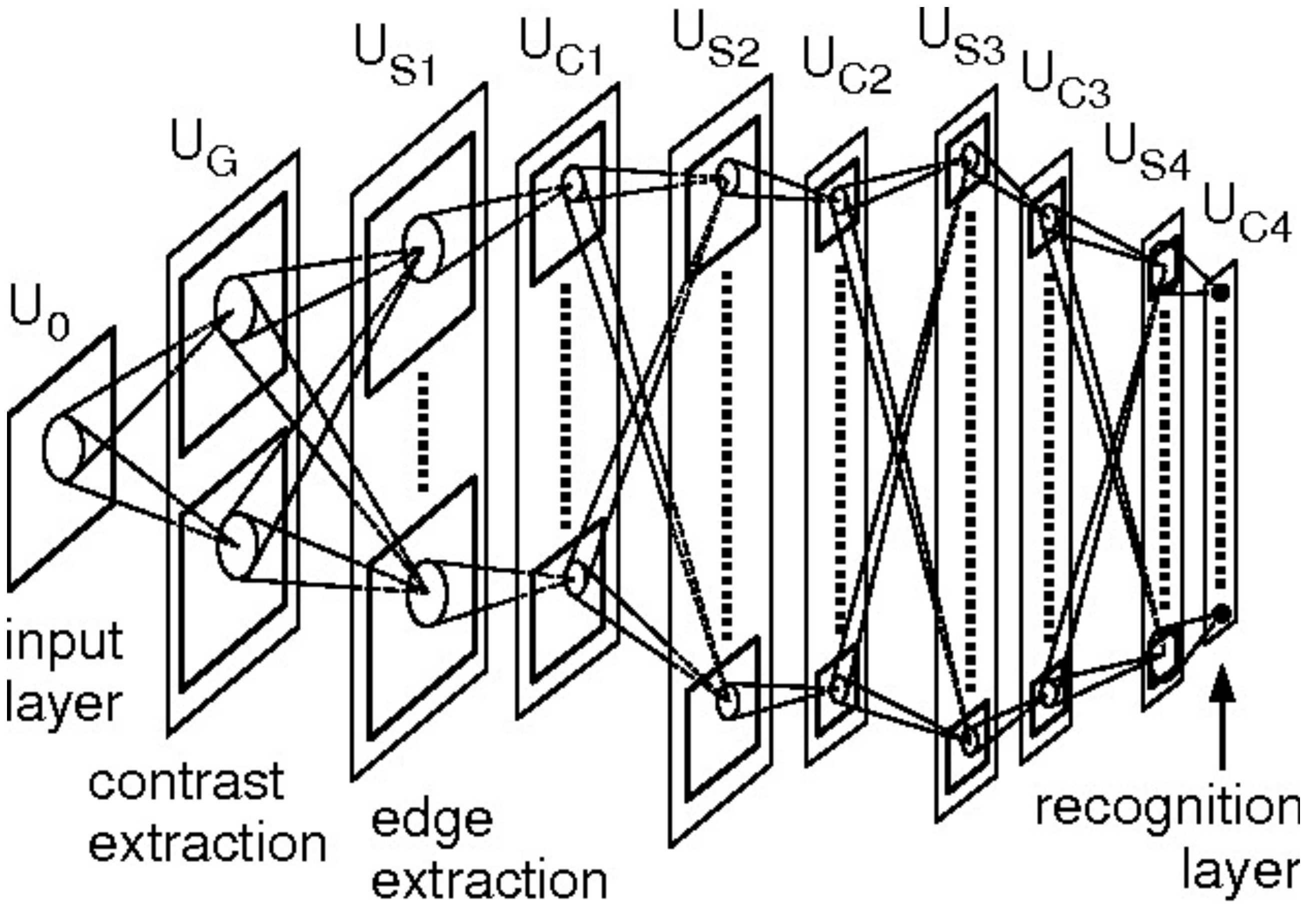
Putting it together



Softmax layer:

$$P(c \mid \mathbf{x}) = \frac{\exp(\mathbf{w}_c \cdot \mathbf{x})}{\sum_{k=1}^{C} \exp(\mathbf{w}_k \cdot \mathbf{x})}$$

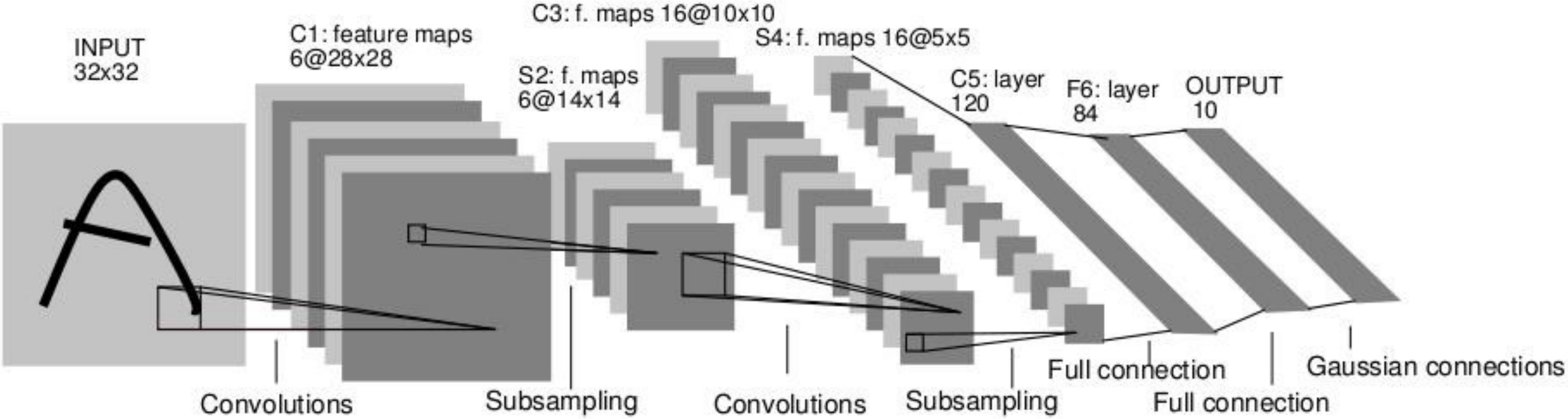
History: Neocognitron



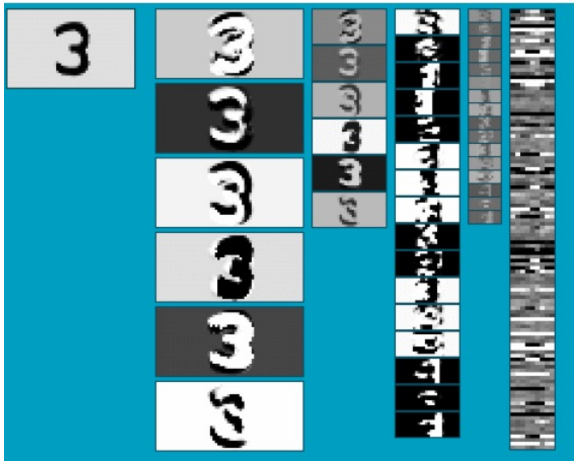
K. Fukushima, 1980s

https://en.wikipedia.org/wiki/Neocognitron

History: LeNet-5

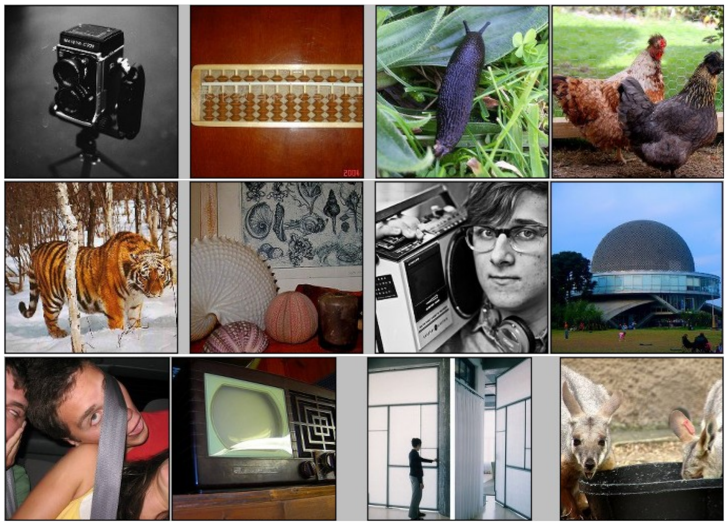


- Average pooling
- Sigmoid or tanh nonlinearity
- Fully connected layers at the end
- Trained on MNIST digit dataset with 60K training examples



Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, [Gradient-based learning applied to document recognition](), Proc. IEEE 86(11): 2278–2324, 1998.

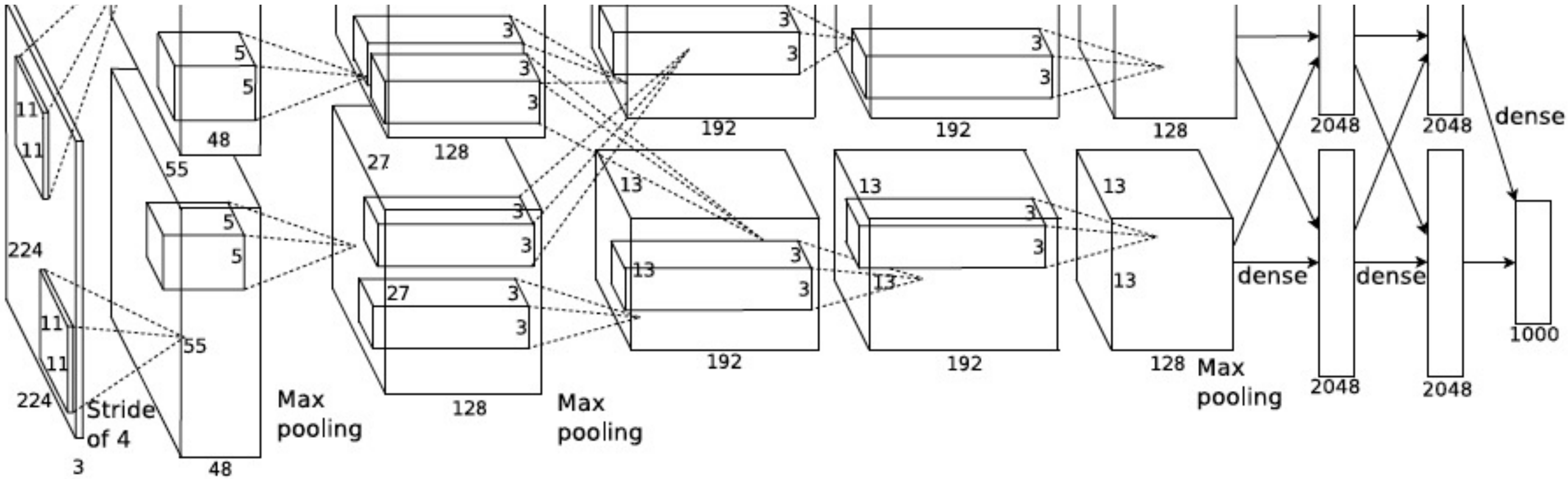
ImageNet Challenge



- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC):
  1.2 million training images, 1000 classes

www.image-net.org/challenges/LSVRC/

AlexNet: ILSVRC 2012 winner



- Similar framework to LeNet but:
  - Max pooling, ReLU nonlinearity
  - More data and bigger model (7 hidden layers, 650K units, 60M params)
  - GPU implementation (50x speedup over CPU)
    - Trained on two GPUs for a week
  - Dropout regularization

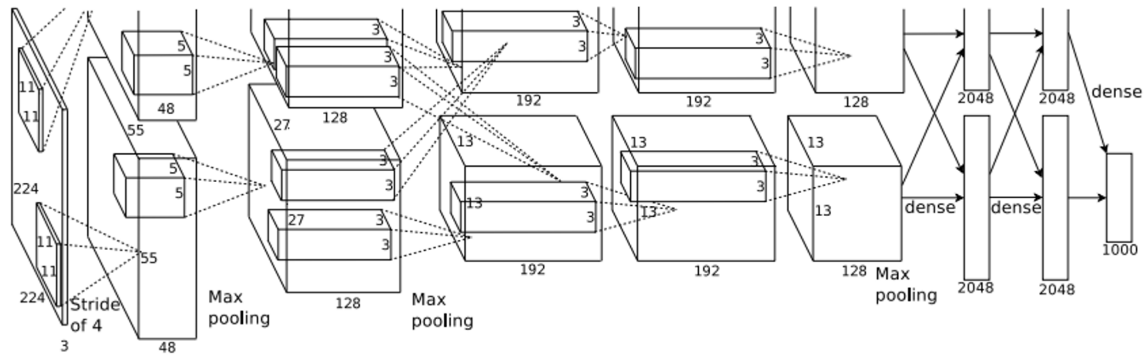
A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

ImageNet Challenge 2012-2014

Team	**Year**	**Place**	**Error (top-5)**	**External data**
XRCE	2011		25.8%	no
SuperVision – Toronto (7 layers)	2012	-	16.4%	no
SuperVision	2012	1st	15.3%	ImageNet 22k

http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/

AlexNet

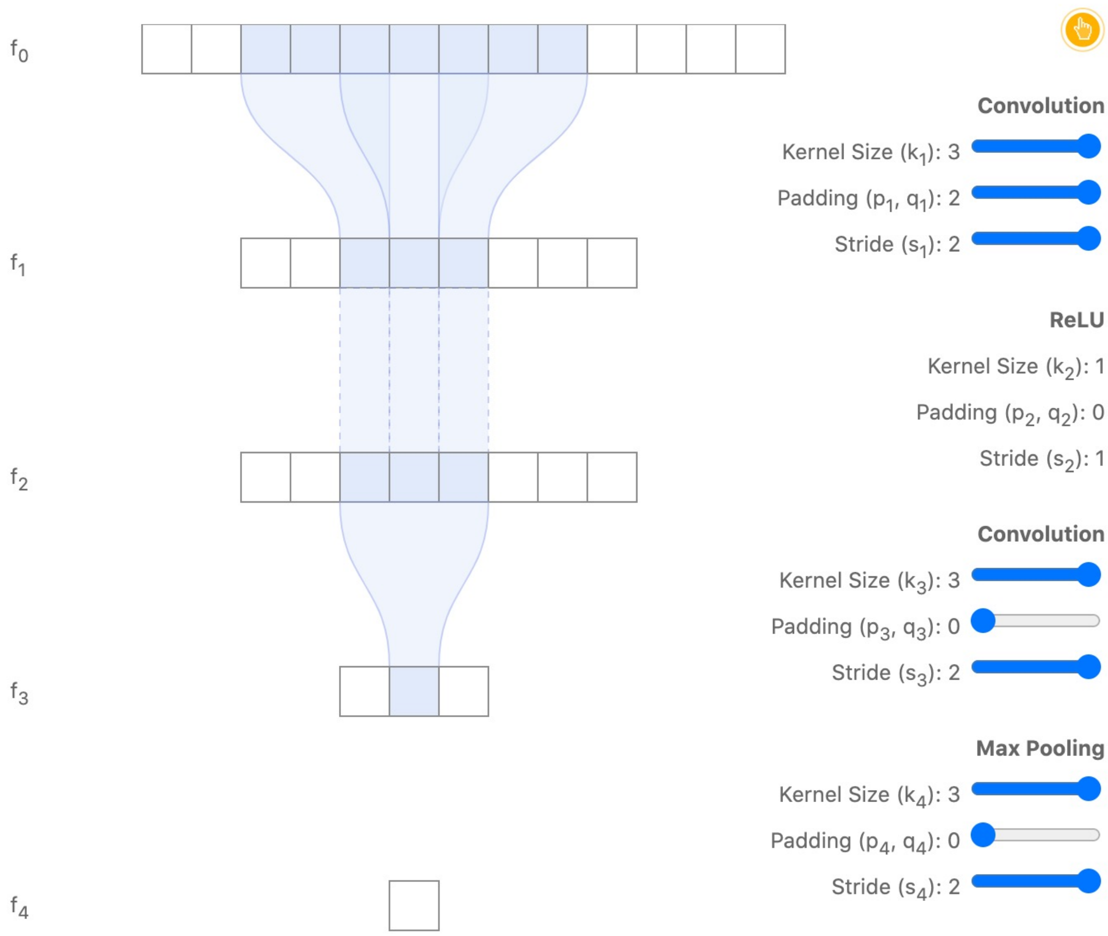


	Input size		Layer				Output size	
Layer	C	H / W	filters	kernel	stride	pad	C	H / W
conv1	3	227	64	11	4	2	64	56
pool1	64	56		3	2	0	64	27
conv2	64	27	192	5	1	2	192	27
pool2	192	27		3	2	0	192	13
conv3	192	13	384	3	1	1	384	13
conv4	384	13	256	3	1	1	256	13
conv5	256	13	256	3	1	1	256	13
pool5	256	13		3	2	0	256	6
flatten	256	6					9216	
fc6	9216		4096				4096	
fc7	4096		4096				4096	
fc8	4096		1000				1000	

Source: Justin Johnson, David Fouhey.

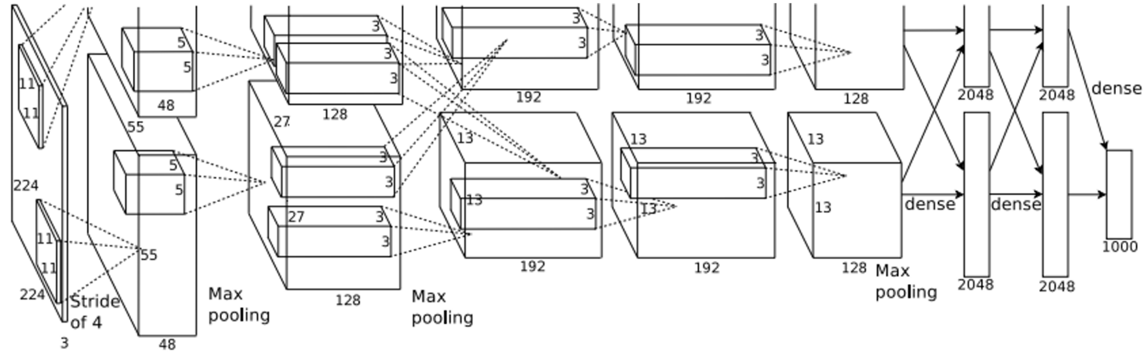
Receptive Field

Deep Nets with striding have large receptive fields



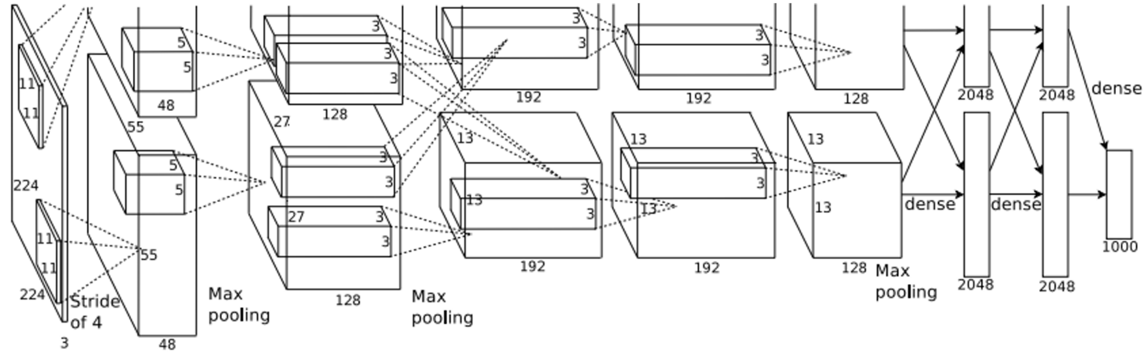
Source: https://distill.pub/2019/computing-receptive-fields/

Receptive Field



	Input size		Layer				Output size		Receptive Field	Effective Stride	Effective Padding
Layer	C	H / W	filters	kernel	stride	pad	C	H / W			
conv1	3	227	64	11	4	2	64	56	11	4	2
pool1	64	56		3	2	0	64	27	19	8	2
conv2	64	27	192	5	1	2	192	27	51	8	18
pool2	192	27		3	2	0	192	13	67	16	34
conv3	192	13	384	3	1	1	384	13	99	16	50
conv4	384	13	256	3	1	1	256	13	131	16	66
conv5	256	13	256	3	1	1	256	13	163	16	66
pool5	256	13		3	2	0	256	6	195	32	66
flatten	256	6					9216		259	32	66
fc6	9216		4096				4096		259	32	66
fc7	4096		4096				4096		259	32	66
fc8	4096		1000				1000		259	32	66

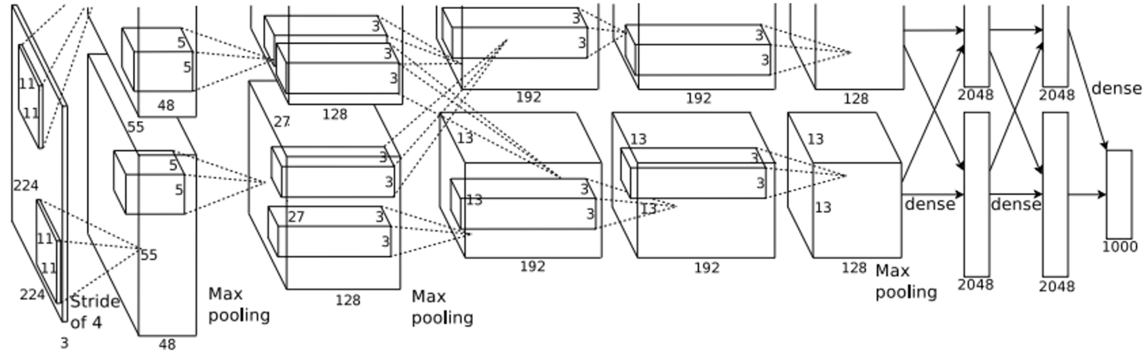
Other Stats



	Input size		Layer				Output size				
Layer	C	H / W	filters	kernel	stride	pad	C	H / W	memory (KB)	params (k)	flop (M)
conv1	3	227	64	11	4	2	64	56	784	23	73
pool1	64	56		3	2	0	64	27	182	0	0
conv2	64	27	192	5	1	2	192	27	547	307	224
pool2	192	27		3	2	0	192	13	127	0	0
conv3	192	13	384	3	1	1	384	13	254	664	112
conv4	384	13	256	3	1	1	256	13	169	885	145
conv5	256	13	256	3	1	1	256	13	169	590	100
pool5	256	13		3	2	0	256	6	36	0	0
flatten	256	6					9216		36	0	0
fc6	9216		4096				4096		16	37,749	38
fc7	4096		4096				4096		16	16,777	17
fc8	4096		1000				1000		4	4,096	4

Source: Justin Johnson, David Fouhey.

AlexNet

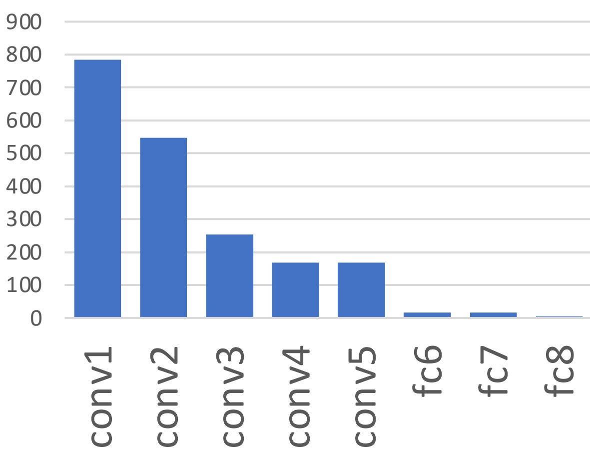


Most of the **memory usage** is in the early convolution layers

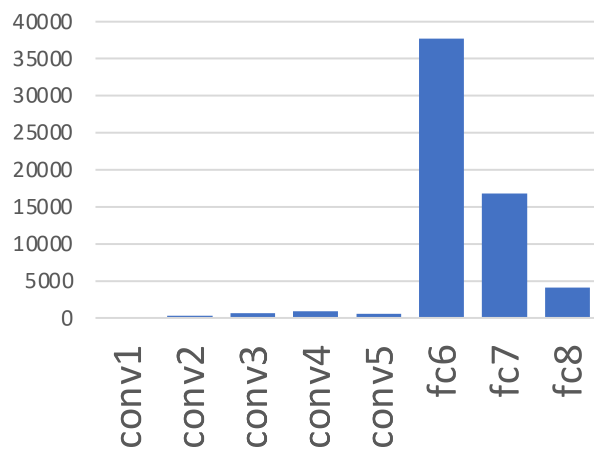
Nearly all **parameters** are in the fully-connected layers

Most **floating-point ops** occur in the convolution layers

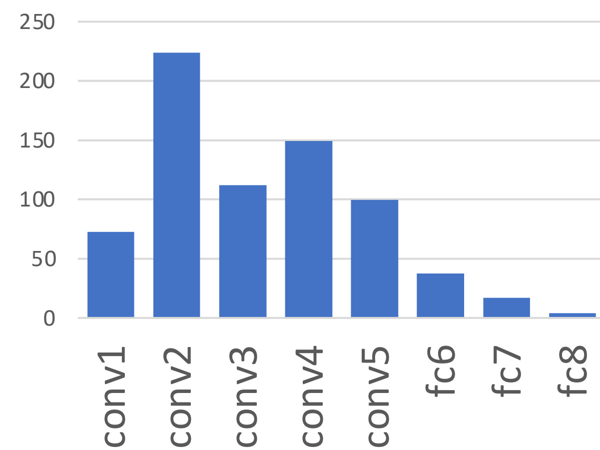
Memory (KB)



Params (K)

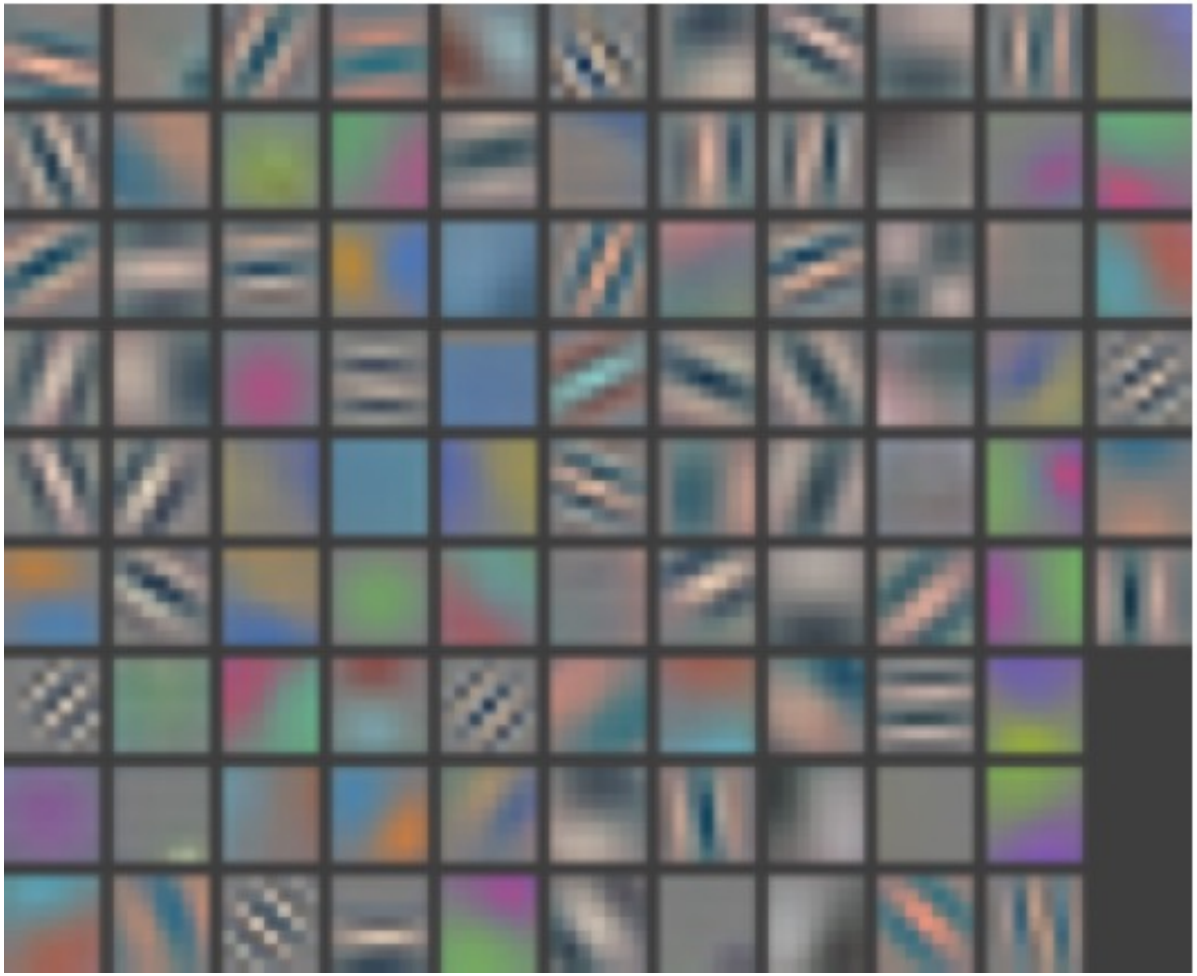


MFLOP



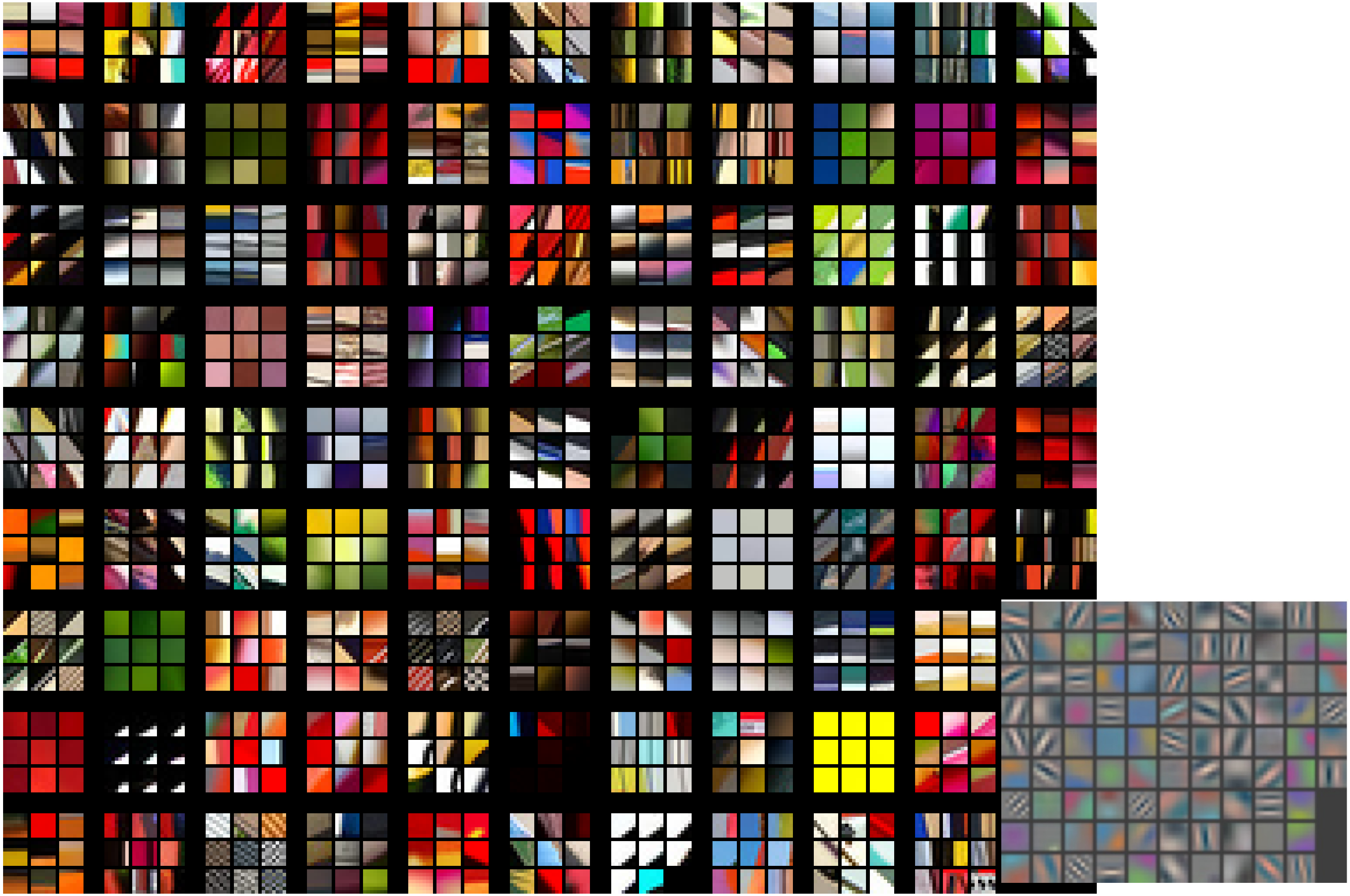
Source: Justin Johnson, David Fouhey.

Layer 1 Filters

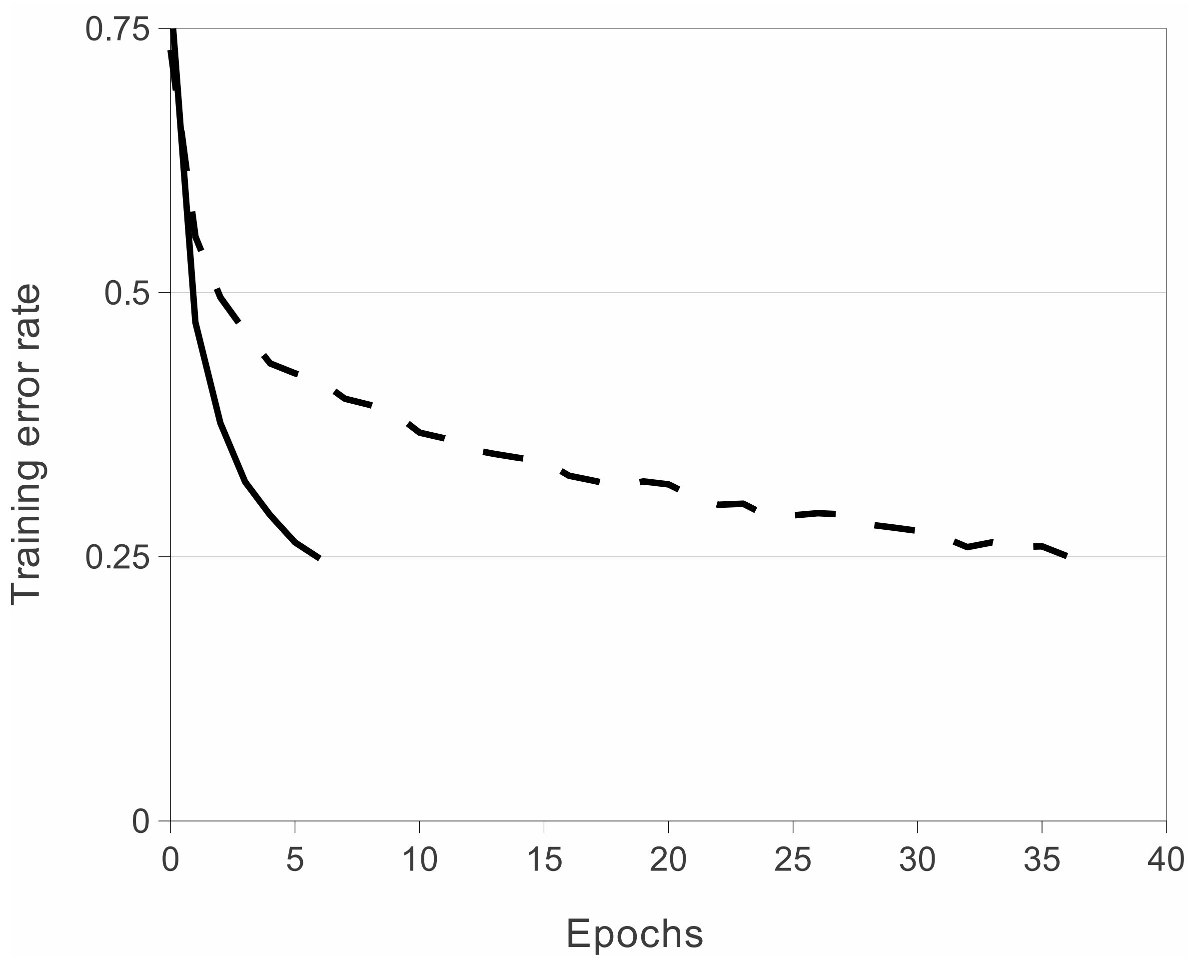


M. Zeiler and R. Fergus, [Visualizing and Understanding Convolutional Networks](), ECCV 2014 (Best Paper Award winner)

Layer 1: Top-9 Patches



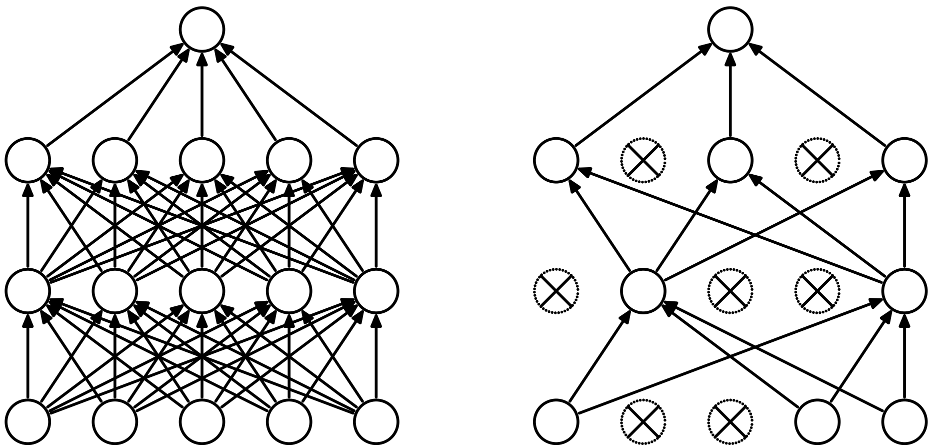
ReLU vs tanh



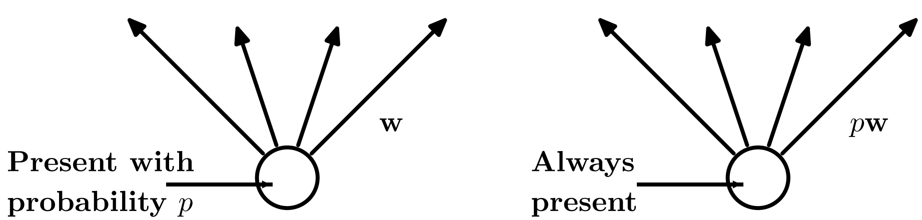
A. Krizhevsky, I. Sutskever, and G. Hinton, [ImageNet Classification with Deep Convolutional Neural Networks](), NIPS 2012

Dropout

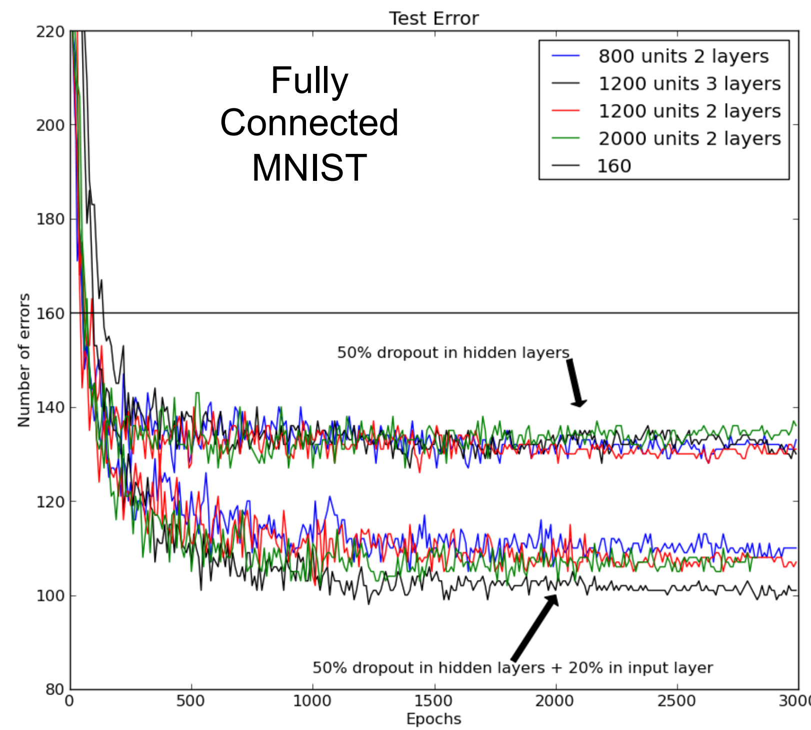
- Randomly drop units in training
- Prevents co-adaptation
- Thought to sample from an exponential number of thinned networks
- Acts as a regularizer



(a) Standard Neural Net (b) After applying dropout.



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



N. Srivastava, G. Hinton et al., [Dropout: A Simple Way to Prevent Neural Networks from Overfitting](), JMLR 2014

Outline

- Building blocks for CNNs
- Motivation and history
- Alexnet
- Since Alexnet

Components of a CNN architecture

- Convolutional Layers
- Non-linearities
- Pooling
- Fully-connected Layers
- Normalization Layers

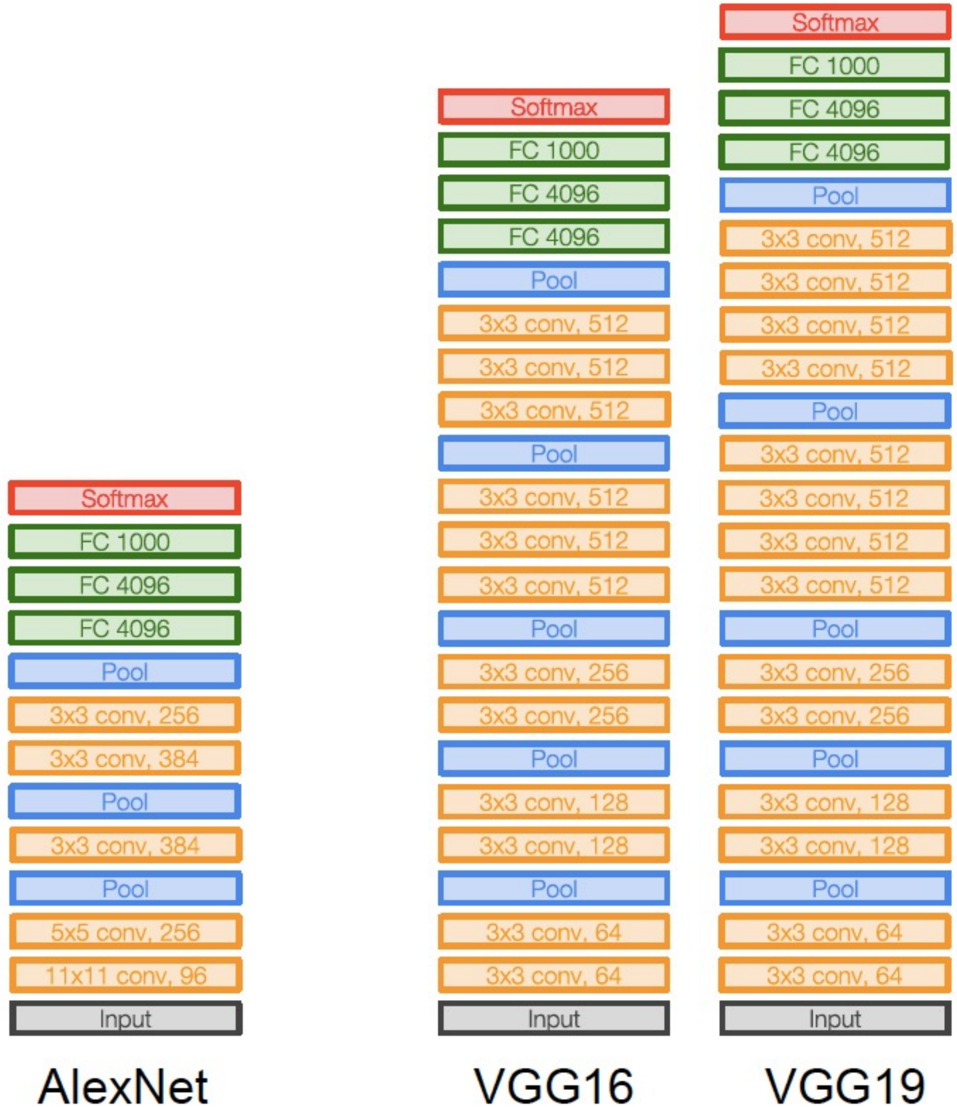
Rationale?

Since Alexnet

- More efficient use of parameters
  - No FC layers
  - Smaller kernels
- Normalization layers
  - LRN layers don't improve performance as much
  - Batch Normalization
- Deeper networks
  - 7 layers -> 19 layers -> 150 layers
  - Residual connections
  - Batch normalization
- Self-attention

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

VGGNet: ILSVRC 2014 2nd place



[Image source]

K. Simonyan and A. Zisserman, [Very Deep Convolutional Networks for Large-Scale Image Recognition], ICLR 2015

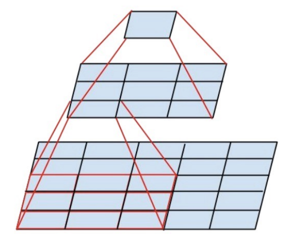
VGGNet: ILSVRC 2014 2^{nd} place

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

- Sequence of deeper networks trained progressively
- Large receptive fields replaced by successive layers of 3x3 convolutions (with ReLU in between)

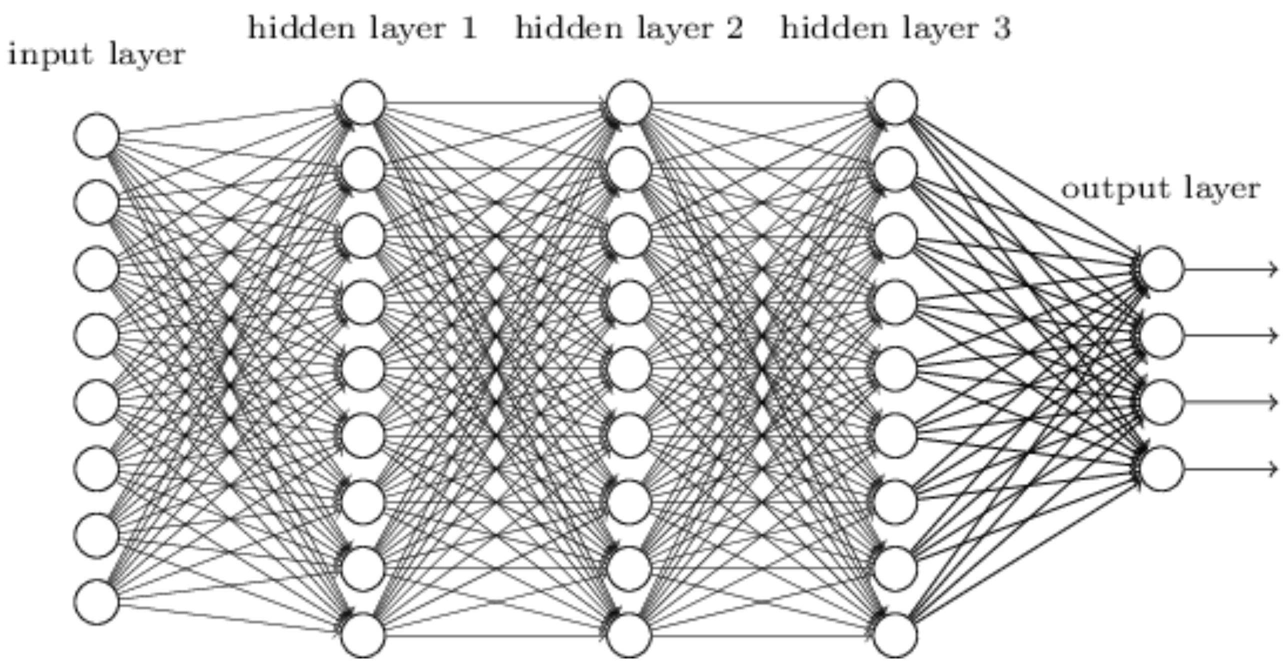


- One 7x7 conv layer with K feature maps needs $49K^2$ weights, three 3x3 conv layers need only $27K^2$ weights
- Experimented with 1x1 convolutions

K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015

Batch Normalization

- Multi-layer training can suffer from “covariate shift”
- Distribution of hidden layer 2 input’s changes over time.



S. Iofffe and C. Szegedy, [Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift](), arXiv 2015

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

At test time:

Use μ and σ obtained from training set (typically done via running average).

$$\mathrm{z} = g(W\mathrm{u} + \mathrm{b}) \longrightarrow \mathrm{z} = g(\mathbf{BN}(W\mathrm{u}))$$

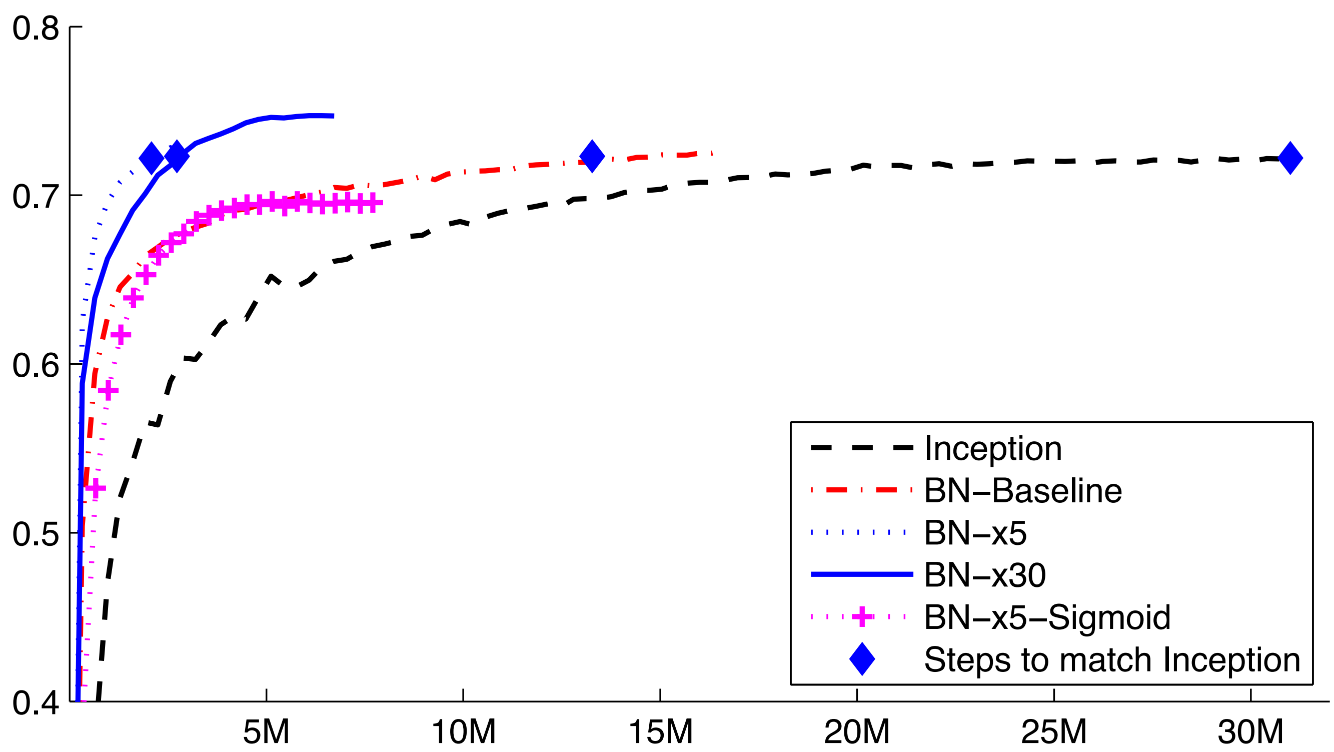
Single biggest source of bugs in my code!!

S. Iofffe and C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, arXiv 2015

Batch Normalization

- Multi-layer training can suffer from “covariate shift”

$$\mathrm{z} = g(W\mathrm{u} + \mathrm{b}) \longrightarrow \mathrm{z} = g(\mathrm{BN}(W\mathrm{u}))$$



S. Iofffe and C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, arXiv 2015

Batch Normalization

- Multi-layer training can suffer from “covariate shift”
- Accelerates training
- Regularizes the model
- Less sensitive to initialization
- See also: How Does Batch Normalization Help Optimization?

S. Iofffe and C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, arXiv 2015

ResNet: ILSVRC 2015 winner

AlexNet, 8 layers
(ILSVRC 2012)

VGG, 19 layers
(ILSVRC 2014)

K. He, X. Zhang, S. Ren, and J. Sun, [Deep Residual Learning for Image Recognition](), CVPR 2016 (Best Paper)

ResNet: ILSVRC 2015 winner

AlexNet, 8 layers
(ILSVRC 2012)

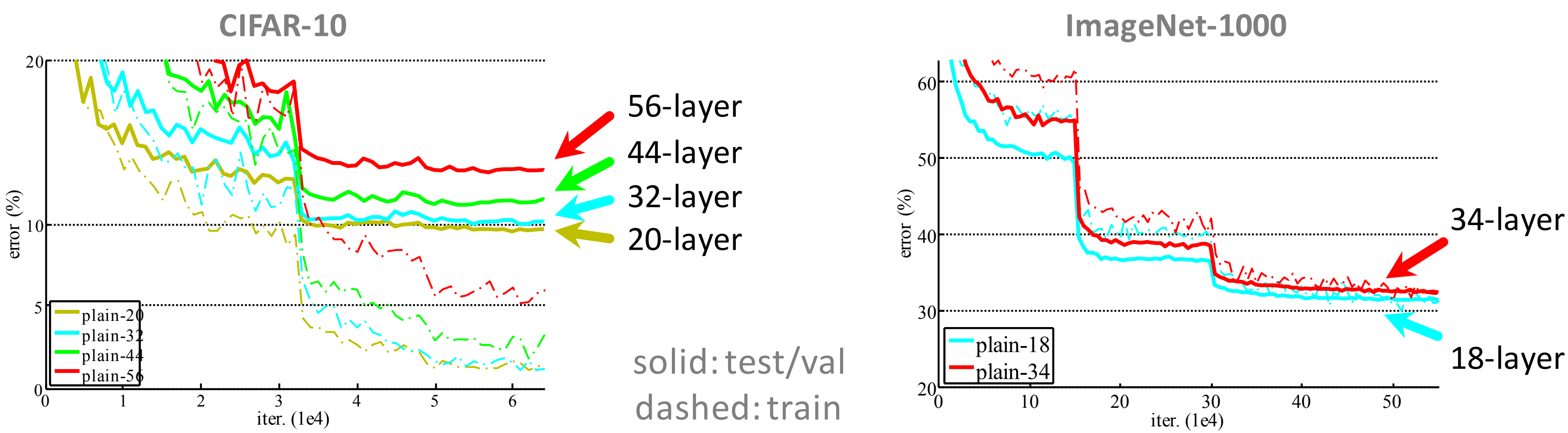
VGG, 19 layers
(ILSVRC 2014)

ResNet, 152 layers
(ILSVRC 2015)

K. He, X. Zhang, S. Ren, and J. Sun, [Deep Residual Learning for Image Recognition](), CVPR 2016 (Best Paper)


I WAS WINNING IMAGENET
UNTIL A DEEPER MODEL CAME ALONG
imgflip.com

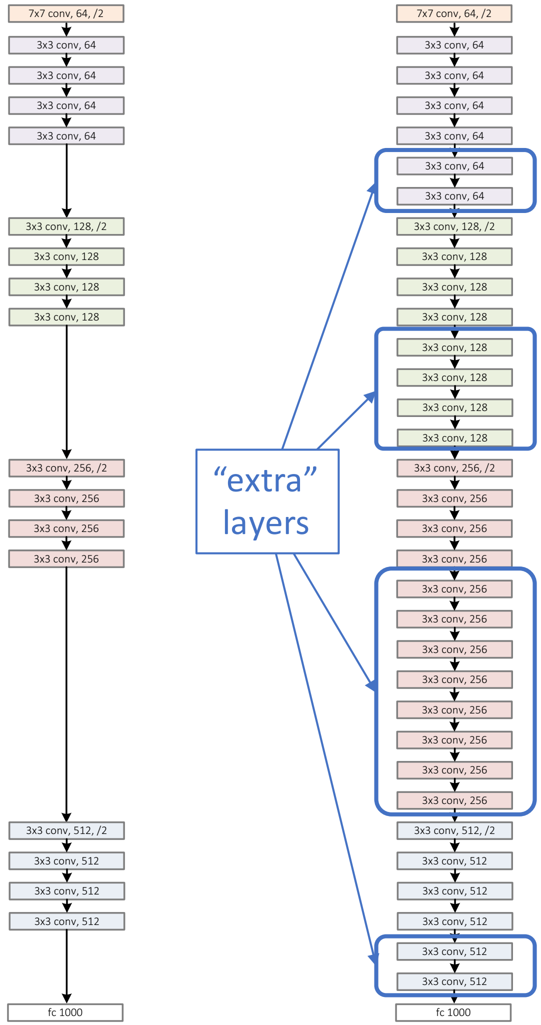
Source (?)



- “Overly deep” plain nets have **higher training error**
- A general phenomenon, observed in many datasets

Slide from Kaiming He.

a shallower model (18 layers)



a deeper counterpart (34 layers)

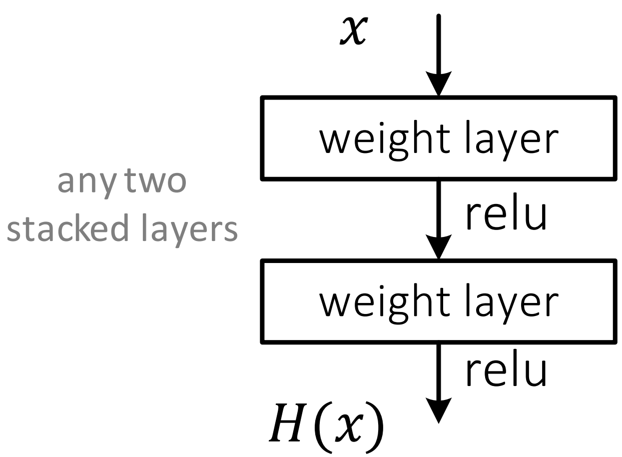
- Richer solution space
- A deeper model should not have **higher training error**
- A solution *by construction*:
  - original layers: copied from a learned shallower model
  - extra layers: set as identity
  - at least the same training error
- Optimization difficulties: solvers cannot find the solution when going deeper…

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

Slide from Kaiming He.

Deep Residual Learning

- Plaint net



$H(x)$ is any desired mapping,

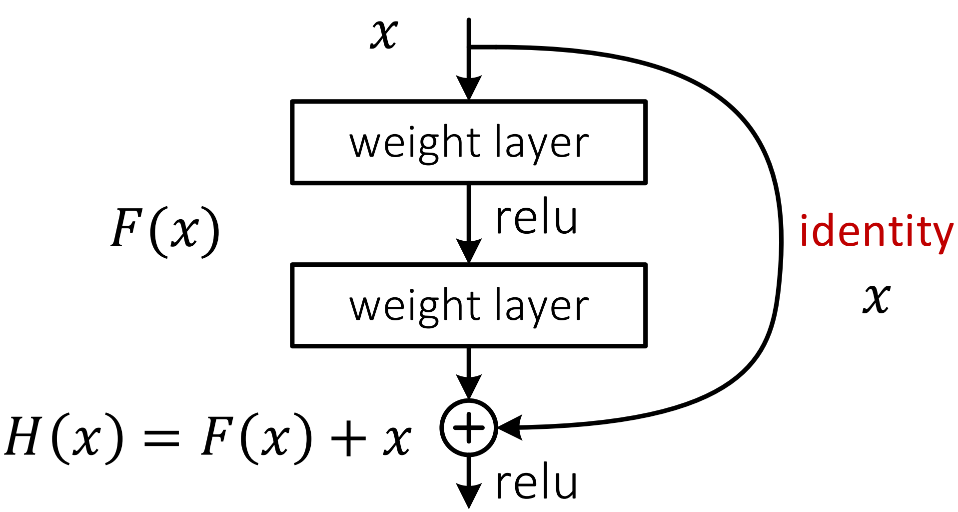
hope the 2 weight layers fit $H(x)$

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

Slide from Kaiming He.

Deep Residual Learning

- Residual net



$H(x)$ is any desired mapping,

~~hope the 2 weight layers fit $H(x)$~~

hope the 2 weight layers fit $F(x)$

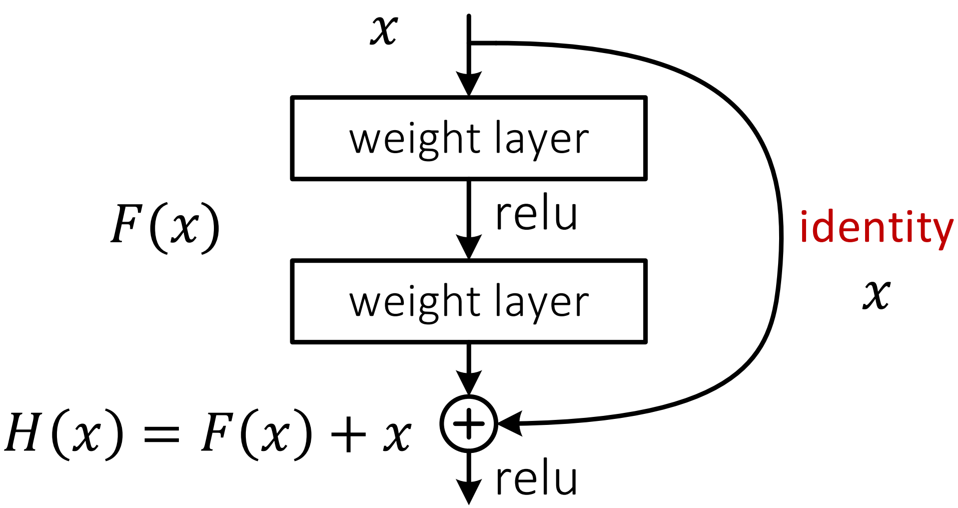
let $H(x) = F(x) + x$

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

Slide from Kaiming He.

Deep Residual Learning

- $F(x)$ is a residual mapping w.r.t. identity

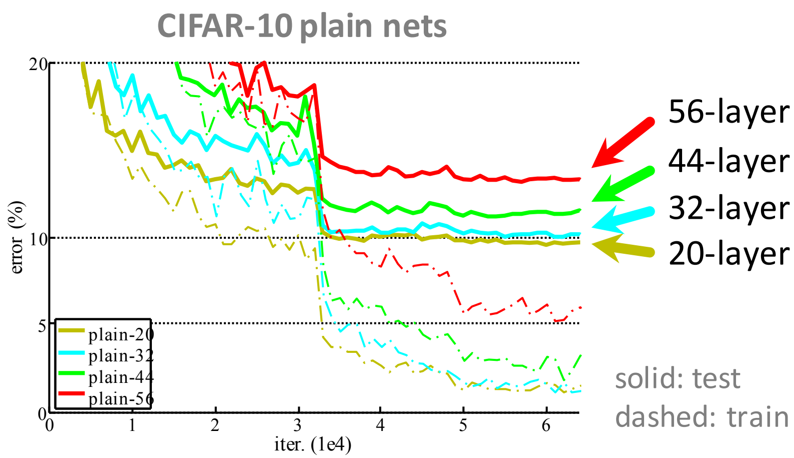


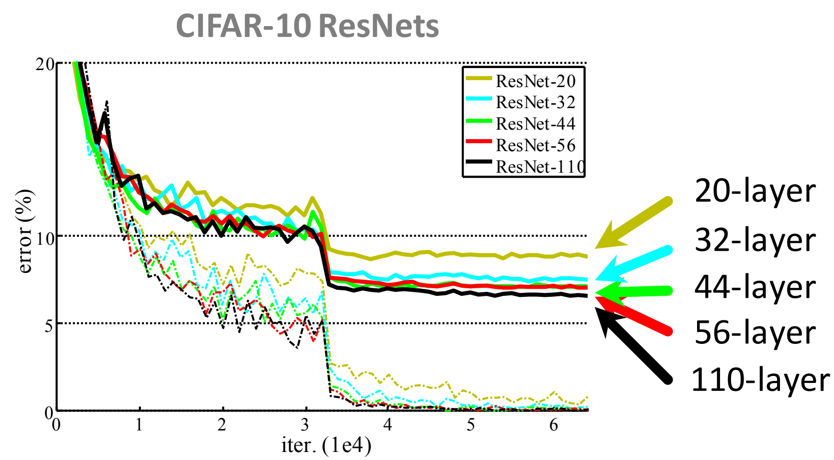
- If identity were optimal, easy to set weights as 0
- If optimal mapping is closer to identity, easier to find small fluctuations

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

Slide from Kaiming He.

CIFAR-10 experiments





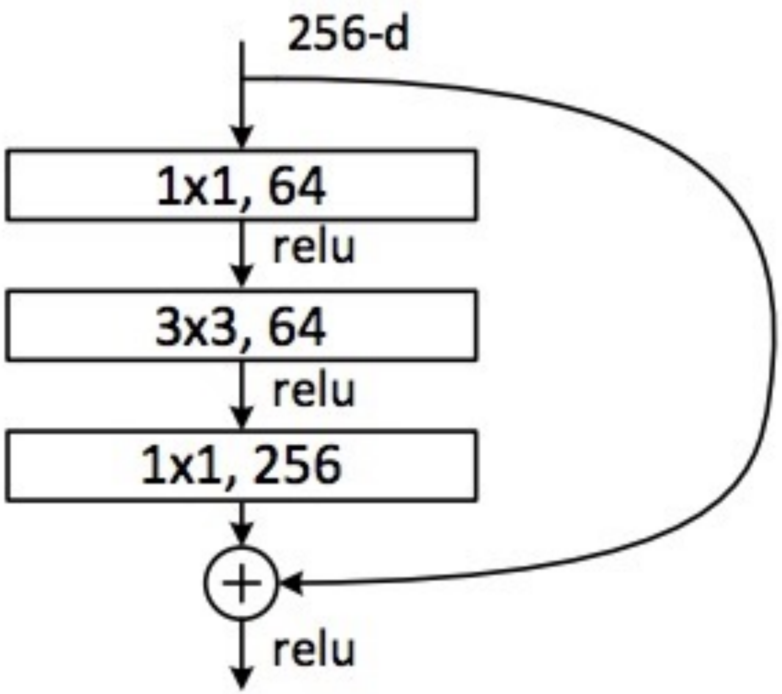
- Deep ResNets can be trained without difficulties
- Deeper ResNets have **lower training error**, and also lower test error

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

Slide from Kaiming He.

ResNet

Deeper residual module (bottleneck)



- Directly performing 3x3 convolutions with 256 feature maps at input and output:
  256 x 256 x 3 x 3 ~ 600K operations
- Using 1x1 convolutions to reduce 256 to 64 feature maps, followed by 3x3 convolutions, followed by 1x1 convolutions to expand back to 256 maps:
  256 x 64 x 1 x 1 ~ 16K
  64 x 64 x 3 x 3 ~ 36K
  64 x 256 x 1 x 1 ~ 16K
  Total: ~70K

K. He, X. Zhang, S. Ren, and J. Sun, [Deep Residual Learning for Image Recognition](), CVPR 2016 (Best Paper)

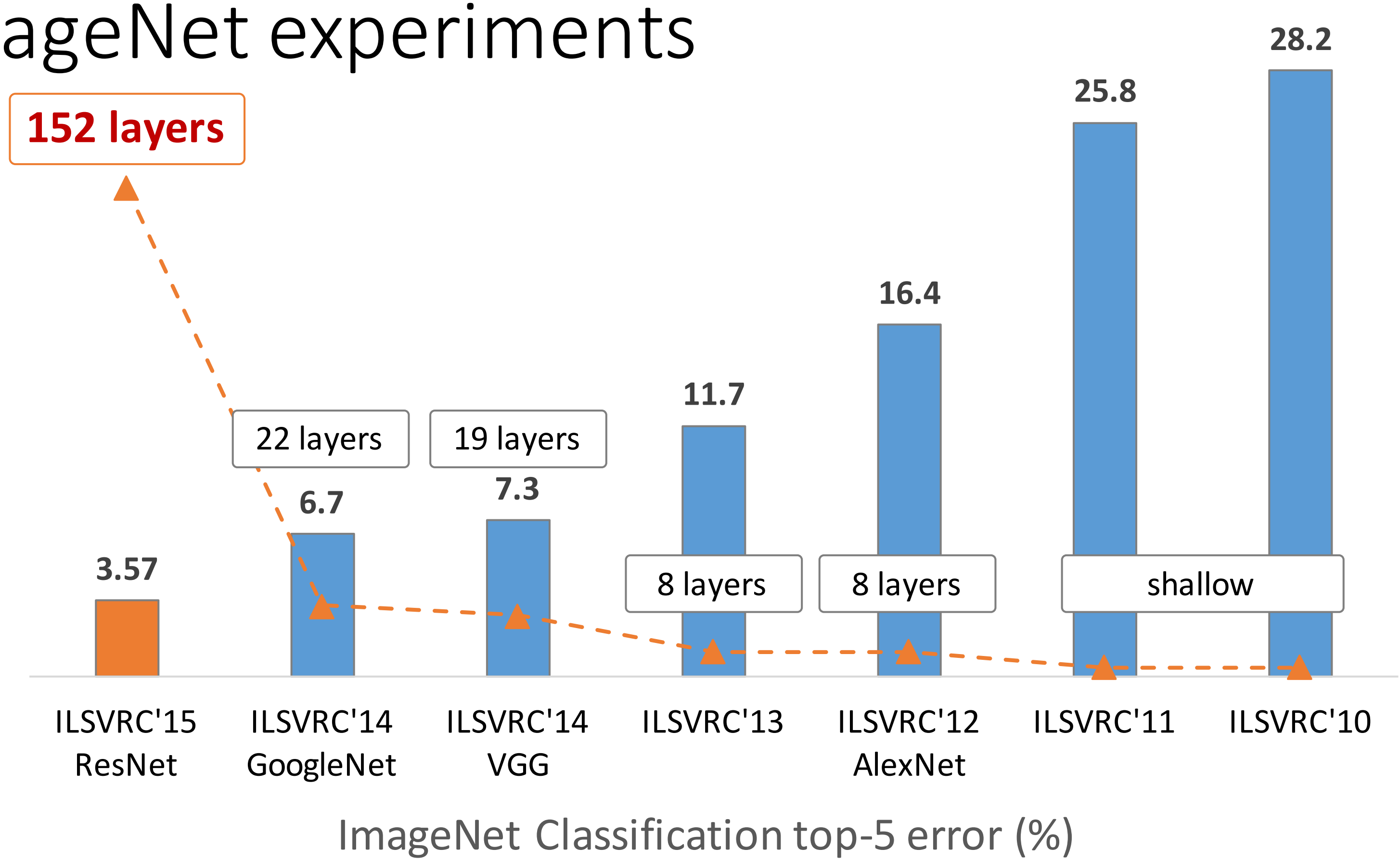
ResNet

Architectures for ImageNet:

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

K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)

ImageNet experiments



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

Slide from Kaiming He.

Summary: ILSVRC 2012-2015

Team	Year	Place	Error (top-5)	External data
SuperVision – Toronto (AlexNet, 7 layers)	2012	-	16.4%	no
SuperVision	2012	1st	15.3%	ImageNet 22k
Clarifai – NYU (7 layers)	2013	-	11.7%	no
Clarifai	2013	1st	11.2%	ImageNet 22k
VGG – Oxford (16 layers)	2014	2nd	7.32%	no
GoogLeNet (19 layers)	2014	1st	6.67%	no
ResNet (152 layers)	2015	1st	3.57%	
Human expert*			5.1%	

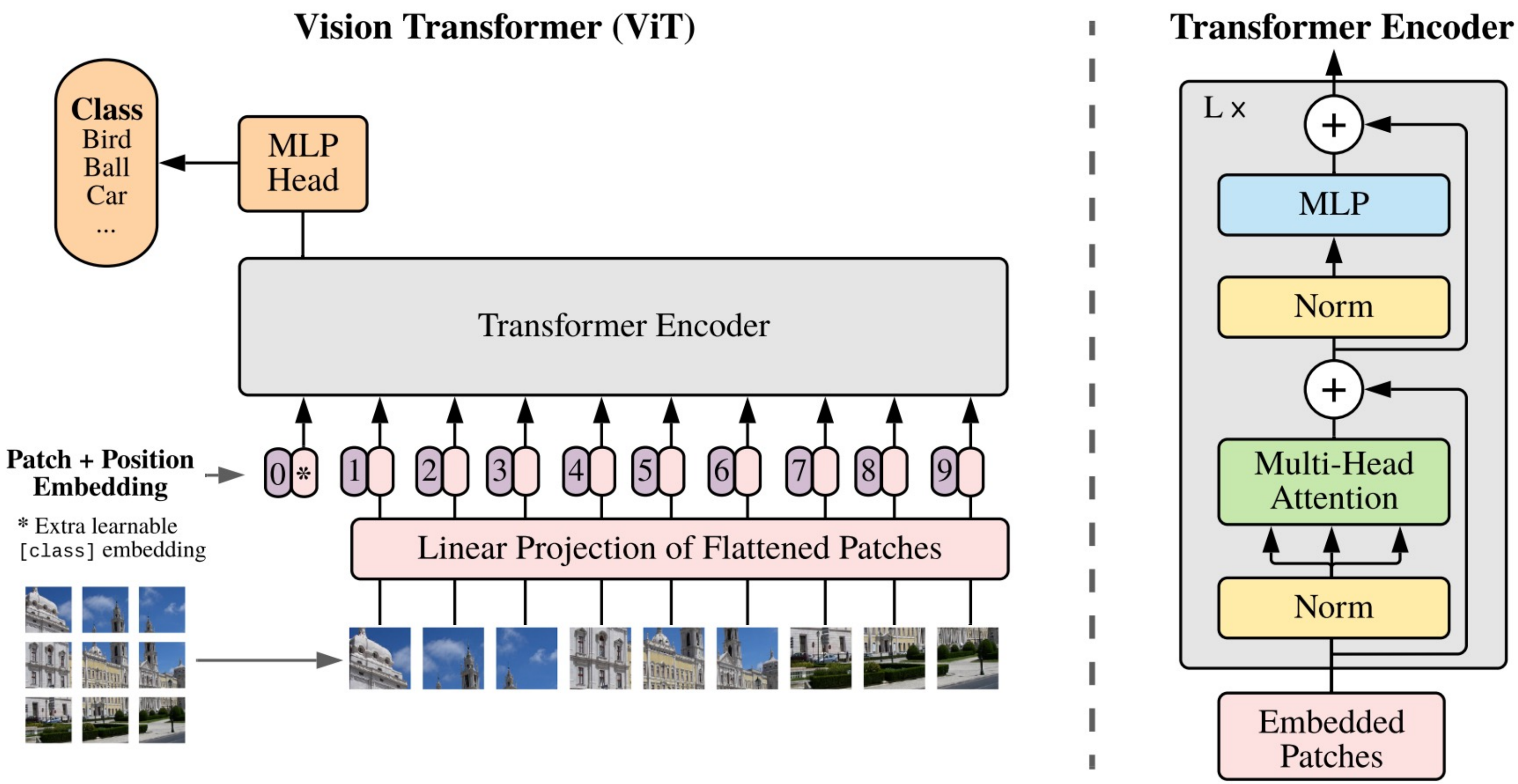
http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/

Other Things

- Training data augmentation
- Averaging classifier outputs over multiple crops/flips
- Ensembles of networks

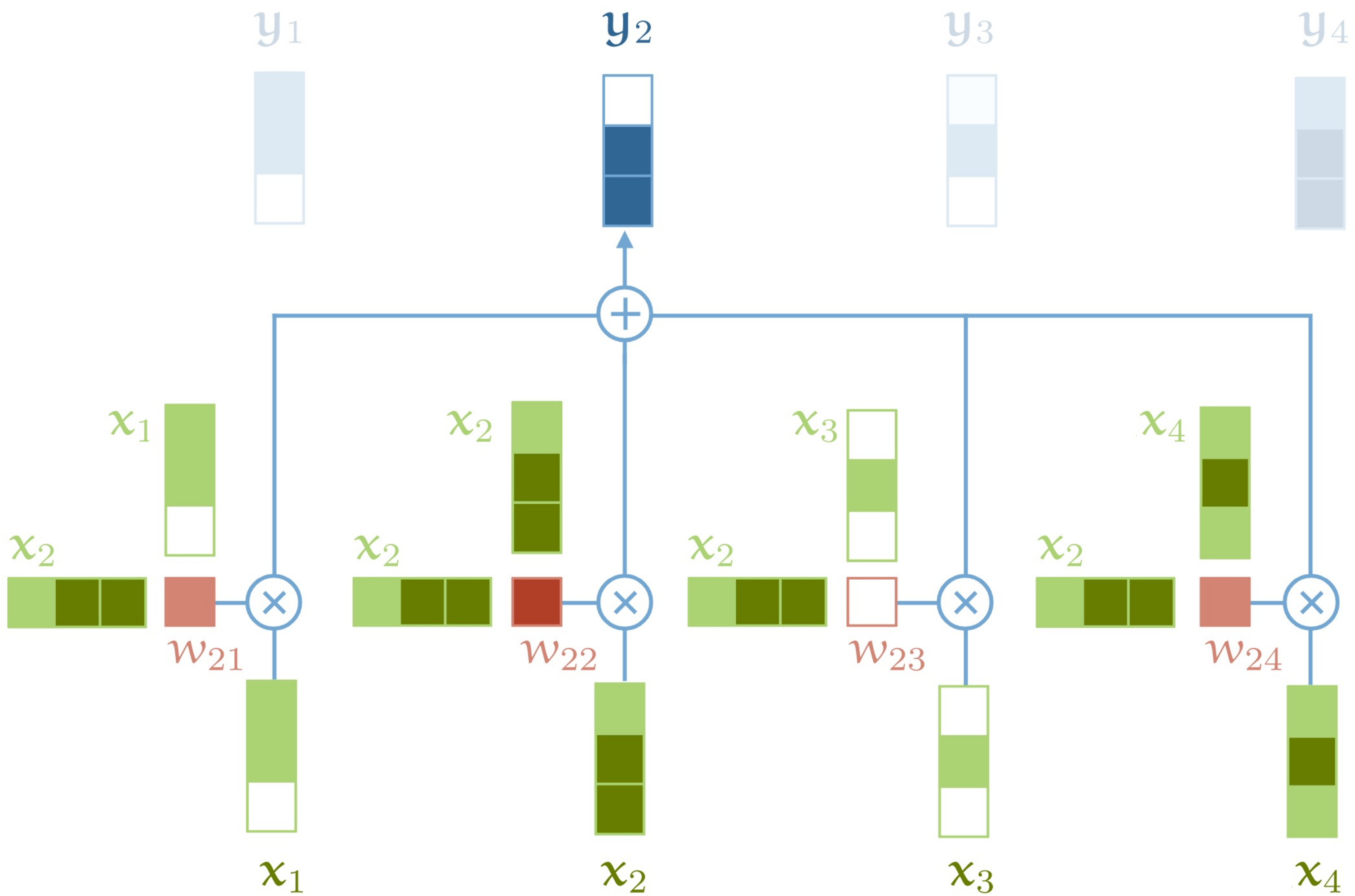
- Officially, starting with 2015, image classification is not part of ILSVRC challenge, but people continue to benchmark on the data

Attention (Vision Transformers)



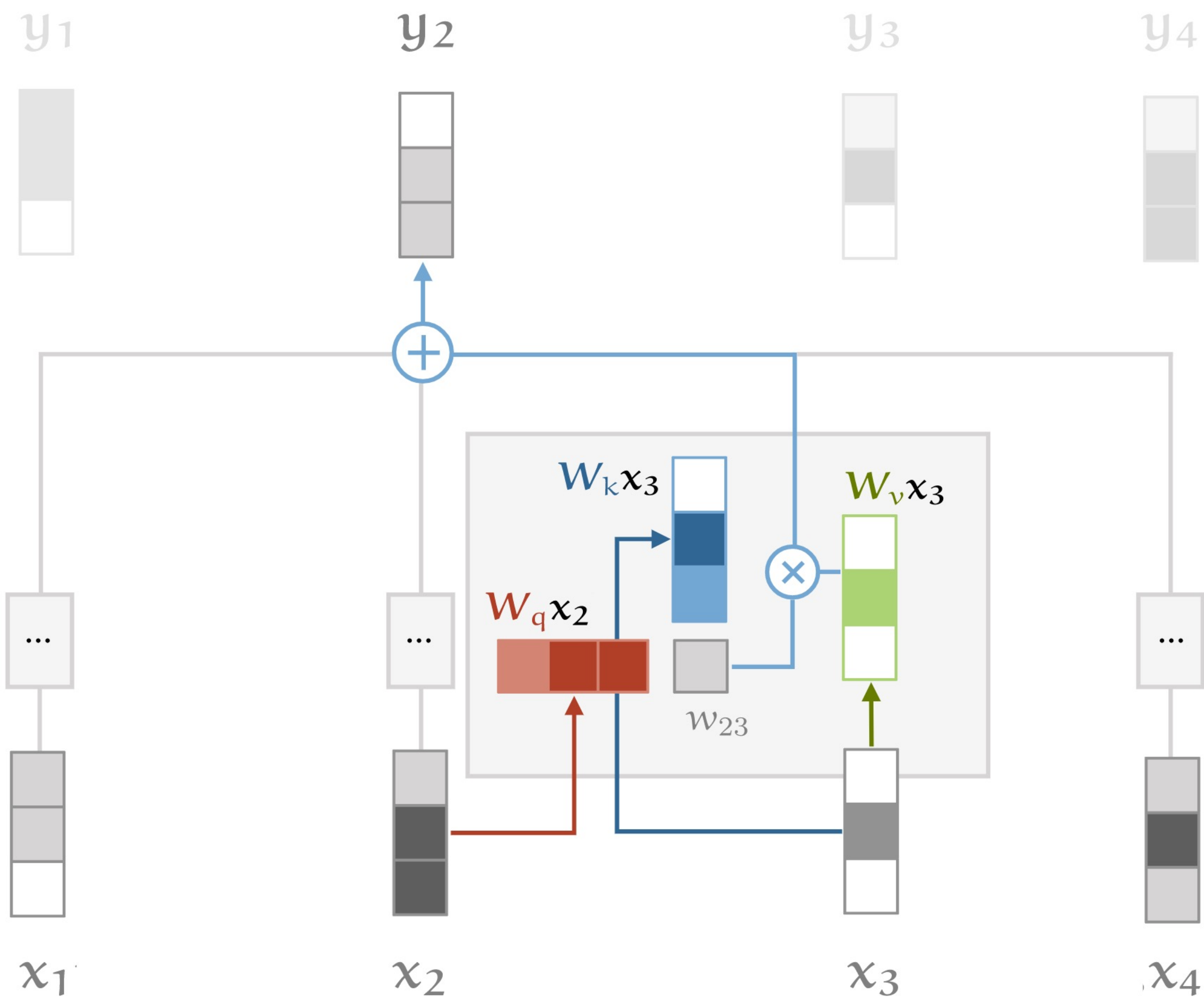
A. Dosovitskiy et al., [An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale](#).

Attention



Source: http://peterbloem.nl/blog/transformers
See also: [Attention is all you need]()

Attention (with key, query and value)



Source: http://peterbloem.nl/blog/transformers
See also: Attention is all you need

Representing Positions

- Positional Embeddings
  - Learn embeddings for different positions
- Positional Encodings
  - Explicitly encode positions using sin, cos terms

See also: [Attention is all you need]

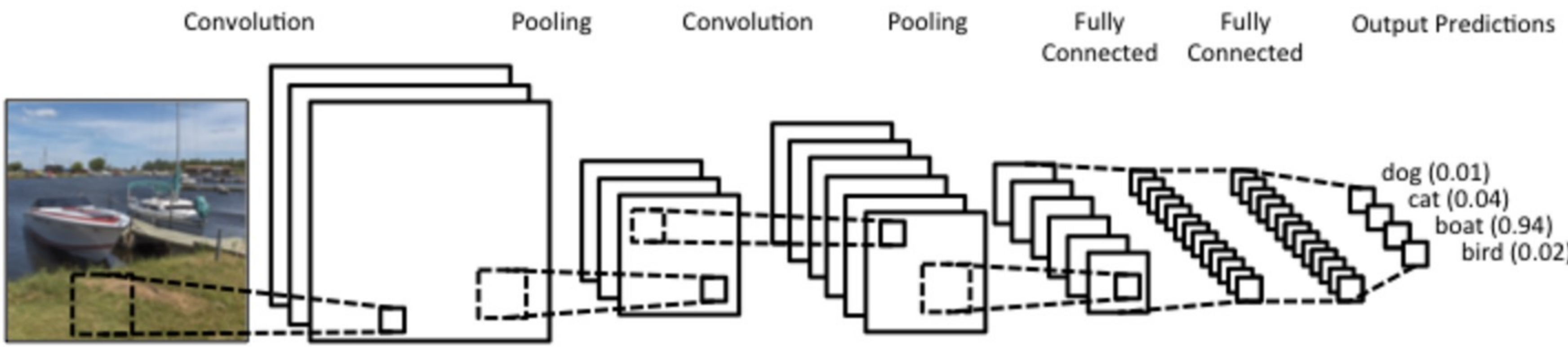
Attention (Vision Transformers)

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21K (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** $\pm$ 0.04	87.76 $\pm$ 0.03	85.30 $\pm$ 0.02	87.54 $\pm$ 0.02	88.4/88.5*
ImageNet ReaL	**90.72** $\pm$ 0.05	90.54 $\pm$ 0.03	88.62 $\pm$ 0.05	90.54	90.55
CIFAR-10	**99.50** $\pm$ 0.06	99.42 $\pm$ 0.03	99.15 $\pm$ 0.03	99.37 $\pm$ 0.06	–
CIFAR-100	**94.55** $\pm$ 0.04	93.90 $\pm$ 0.05	93.25 $\pm$ 0.05	93.51 $\pm$ 0.08	–
Oxford-IIIT Pets	**97.56** $\pm$ 0.03	97.32 $\pm$ 0.11	94.67 $\pm$ 0.15	96.62 $\pm$ 0.23	–
Oxford Flowers-102	99.68 $\pm$ 0.02	**99.74** $\pm$ 0.00	99.61 $\pm$ 0.02	99.63 $\pm$ 0.03	–
VTAB (19 tasks)	**77.63** $\pm$ 0.23	76.28 $\pm$ 0.46	72.72 $\pm$ 0.21	76.29 $\pm$ 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Table 2: Comparison with state of the art on popular image classification benchmarks. We report mean and standard deviation of the accuracies, averaged over three fine-tuning runs. Vision Transformer models pre-trained on the JFT-300M dataset outperform ResNet-based baselines on all datasets, while taking substantially less computational resources to pre-train. ViT pre-trained on the smaller public ImageNet-21k dataset performs well too. *Slightly improved 88.5% result reported in Touvron et al. (2020).

A. Dosovitskiy et al., An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.

Learned Representations are Useful in General



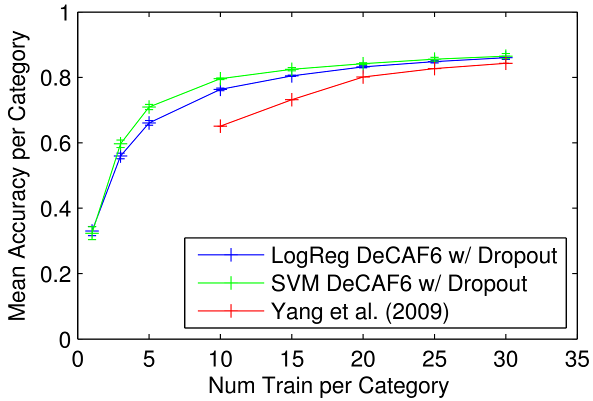
1. Features extracted from CNNs trained on ImageNet were effective for many CV tasks.
2. Furthermore, learned network weights serve as an excellent starting point for other tasks.

J. Donahue, Y. Jia et al. [DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition]. ICML 2014

How to use a trained network for a new task?

	$DeCAF_5$	$DeCAF_6$	$DeCAF_7$
LogReg	63.29 ± 6.6	84.30 ± 1.6	84.87 ± 0.6
LogReg with Dropout	-	86.08 ± 0.8	85.68 ± 0.6
SVM	77.12 ± 1.1	84.77 ± 1.2	83.24 ± 1.2
SVM with Dropout	-	$\mathbf{86.91 \pm 0.7}$	85.51 ± 0.9
Yang et al. (2009)		84.3	
Jarrett et al. (2009)		65.5	

Caltech 101



Caltech 101

	Amazon → Webcam		
	SURF	$DeCAF_6$	$DeCAF_7$
Logistic Reg. (S)	9.63 ± 1.4	48.58 ± 1.3	53.56 ± 1.5
SVM (S)	11.05 ± 2.3	52.22 ± 1.7	53.90 ± 2.2
Logistic Reg. (T)	24.33 ± 2.1	72.56 ± 2.1	74.19 ± 2.8
SVM (T)	51.05 ± 2.0	78.26 ± 2.6	78.72 ± 2.3
Logistic Reg. (ST)	19.89 ± 1.7	75.30 ± 2.0	76.32 ± 2.0
SVM (ST)	23.19 ± 3.5	80.66 ± 2.3	79.12 ± 2.1
Daume III (2007)	40.26 ± 1.1	$\mathbf{82.14 \pm 1.9}$	81.65 ± 2.4
Hoffman et al. (2013)	37.66 ± 2.2	80.06 ± 2.7	80.37 ± 2.0
Gong et al. (2012)	39.80 ± 2.3	75.21 ± 1.2	77.55 ± 1.9
Chopra et al. (2013)		58.85	

Domain Adaptation

Method	Accuracy
$DeCAF_6$	58.75
DPD + $DeCAF_6$	**64.96**
DPD (Zhang et al., 2013)	50.98
POOF (Berg & Belhumeur, 2013)	56.78

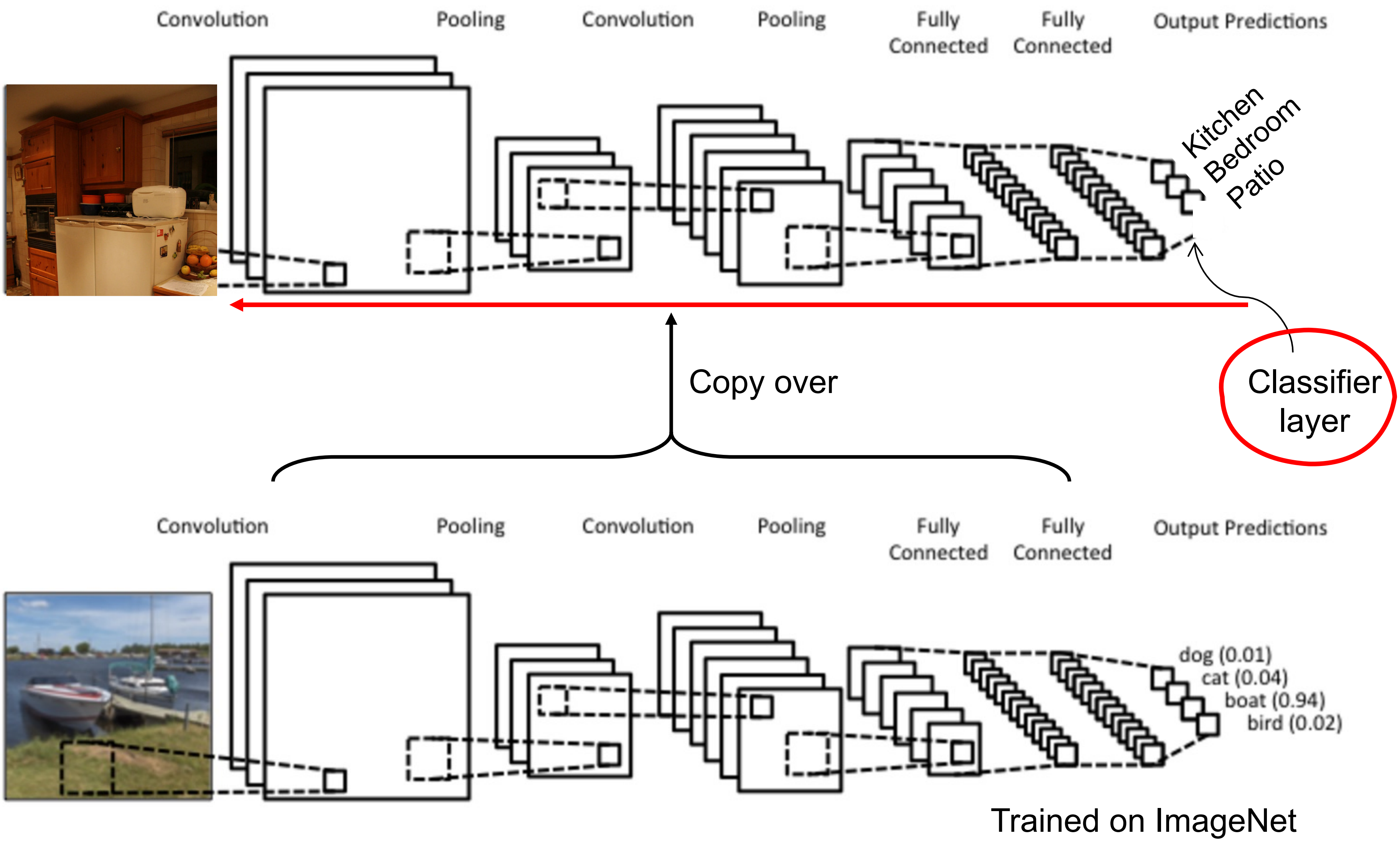
Fine-grained Classification

	$DeCAF_6$	$DeCAF_7$
LogReg	$\mathbf{40.94 \pm 0.3}$	40.84 ± 0.3
SVM	39.36 ± 0.3	40.66 ± 0.3
Xiao et al. (2010)	38.0	

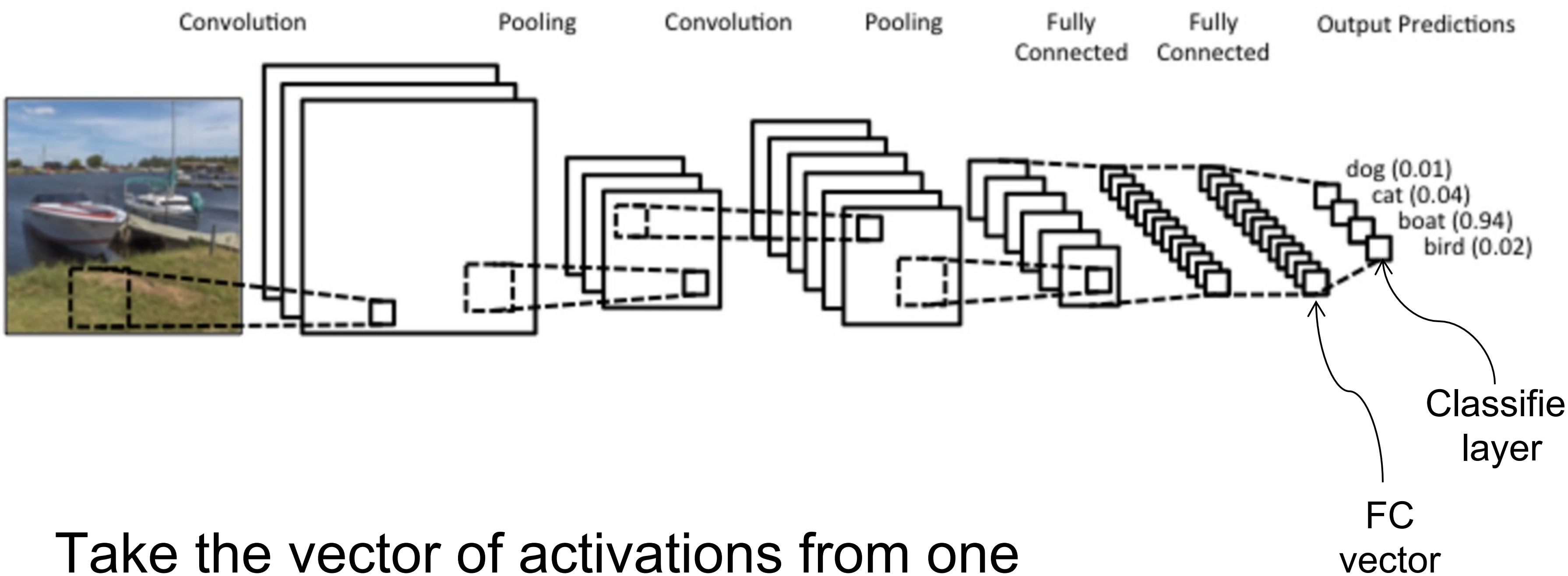
Scene Classification

J. Donahue, Y. Jia et al. [DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition](). ICML 2014

How to use a trained network for a new task?



How to use a trained network for a new task?



- Take the vector of activations from one of the fully connected (FC) layers and treat it as an off-the-shelf feature
  - Train a new classifier layer on top of the FC layer
- *Fine-tune* the whole network