Artificial Neural Networks
Lecture 5: Convolutional Nets

Tudor Berariu
tudor.berariu@gmail.com

Faculty of Automatic Control and Computers
University Politehnica of Bucharest

Lecture : 4th of November, 2015
Last Updated: 1st of December, 2015

Today's Outline

1. Object recognition
2. Convolutional Networks
3. The Breakthrough on ImageNet
4. Understanding and Visualizing Convolutional Networks
5. Pushing further the State-of-the-Art
6. Fooling Convolutional Nets
7. Spatial Transformer Networks

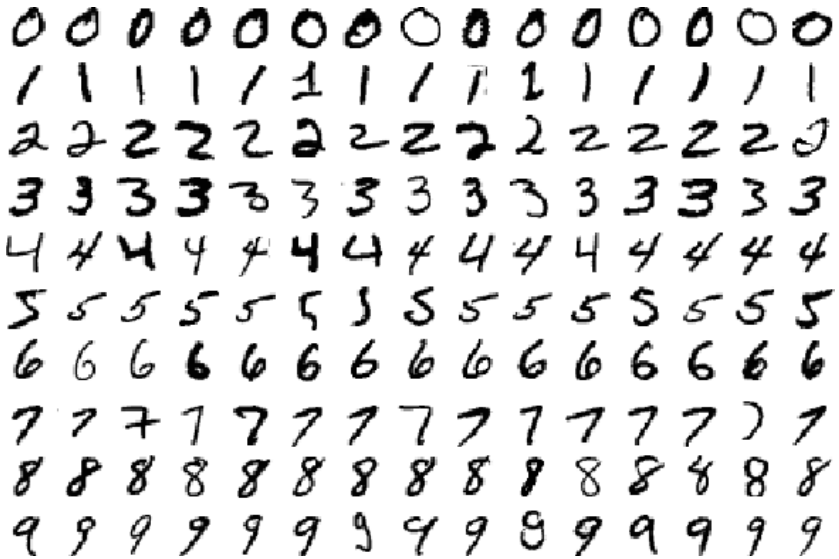
Today's Outline

The Problem

- Classic challenges in images:
  - object identification in a scene (classification)
  - object localization
- Why is object recongition a **difficult** problem?
  - segmentation (lots of objects, overlaped objects)
  - scale, rotation and other deformations
  - lighting
  - variability
  - objects classified by their function

Popular Data Sets

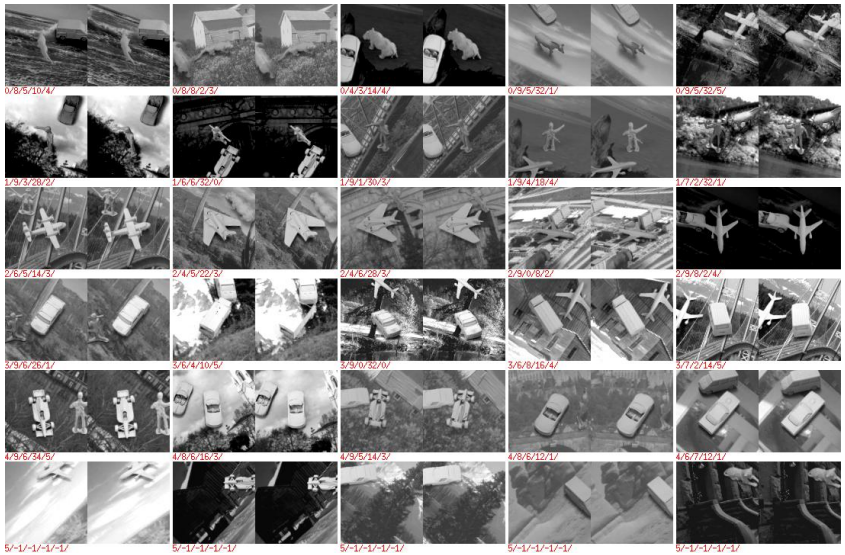
- MNIST
- http://yann.lecun.com/exdb/mnist/



Popular Data Sets

- MNIST
- NORB

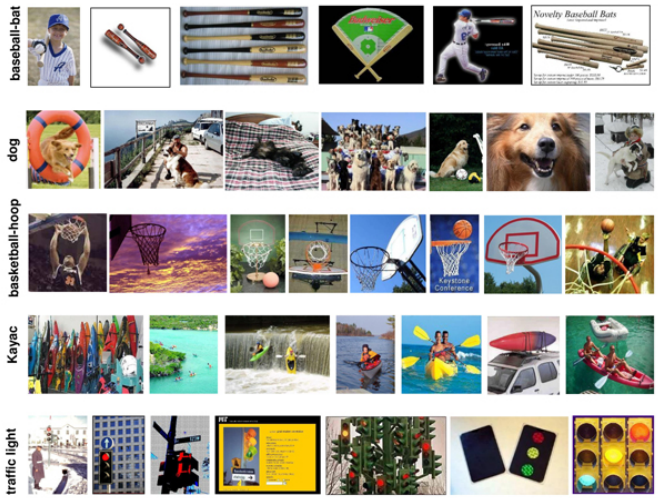
- binocular images of toy figurines under various illumination and pose
- `http://www.cs.nyu.edu/~ylclab/data/norb-v1.0/`



Popular Data Sets

- MNIST
- NORB
- Caltech-101/256

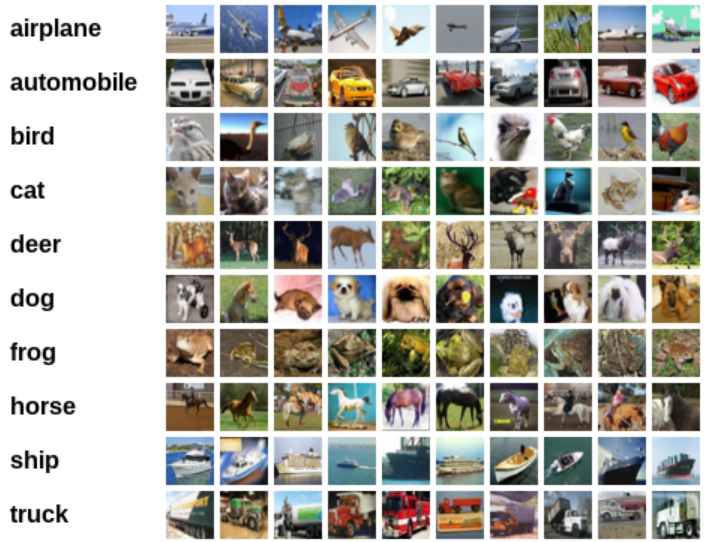
- 30607 images in 256 categories
- manually collected from Google Images



Popular Data Sets

- MNIST
- NORB
- Caltech-101/256
- CIFAR-10/100

- 60000 32x32 colour images
- 10 / 100 classes



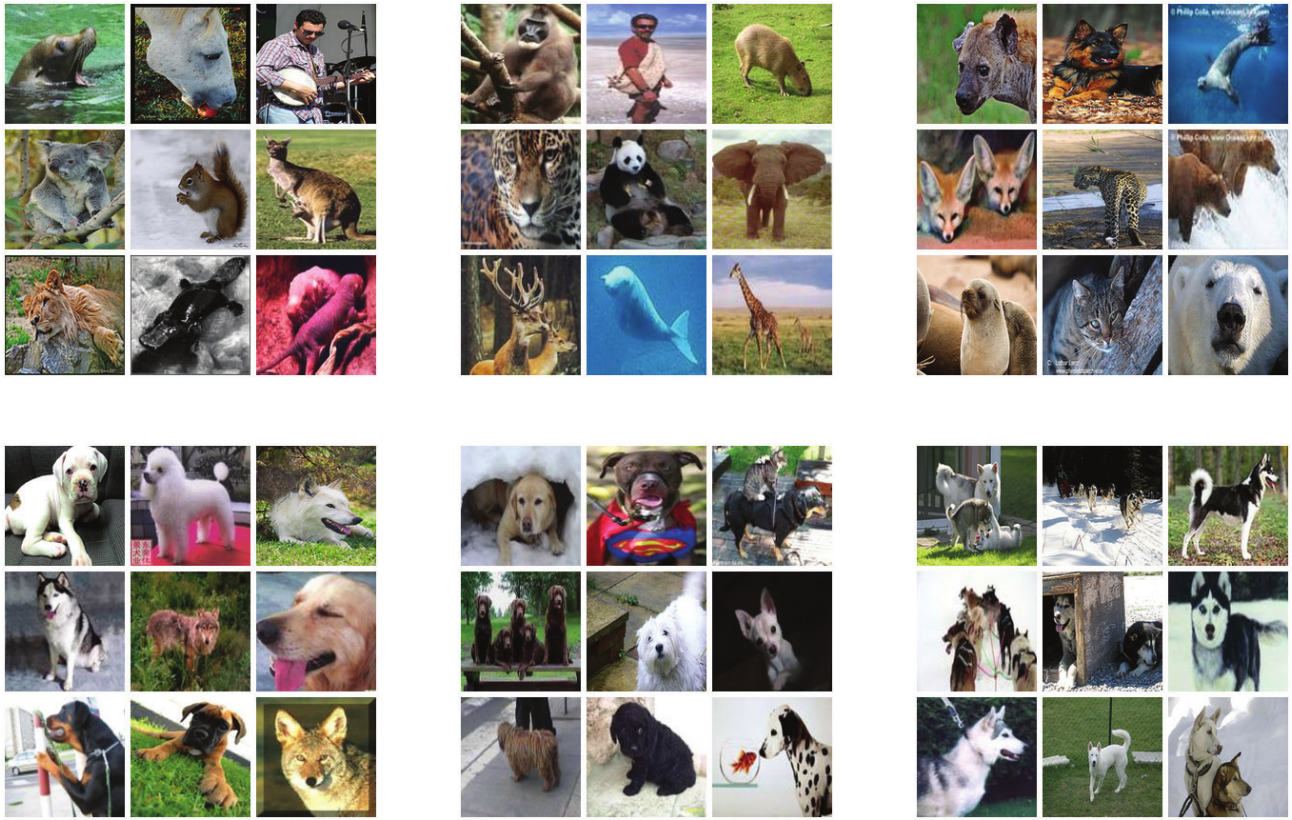
Popular Data Sets

- MNIST
- NORB
- Caltech-101/256
- CIFAR-10/100
- SVHN

- `http://ufldl.stanford.edu/housenumbers/`
- 73257 digits for training
- 26032 digits for testing

ImageNet

- 15 million high-resolution images [DDS+09]
- 22000 hierarchical classes



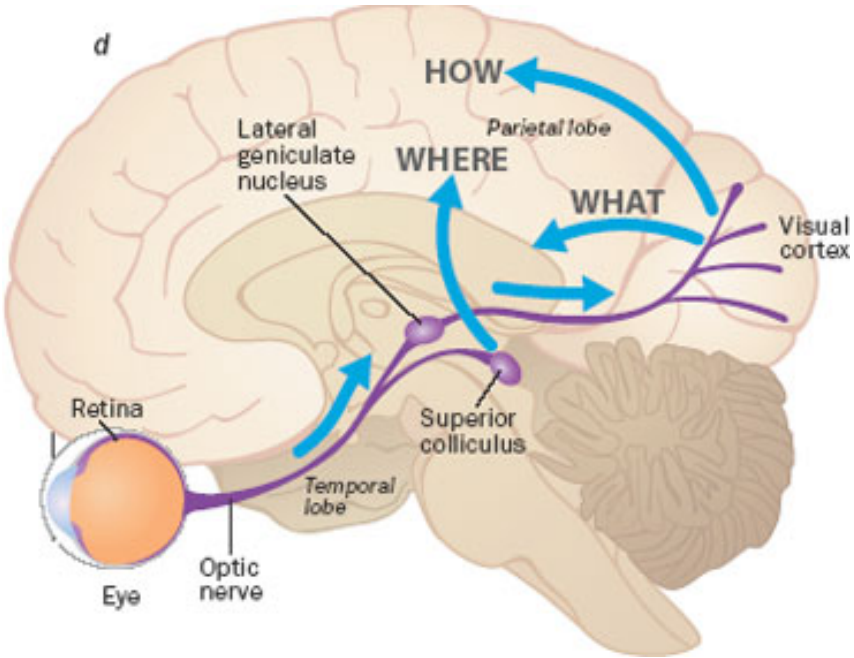
- labeled by humans using Amazon's Mechanical Turk

Today's Outline

Today's Outline

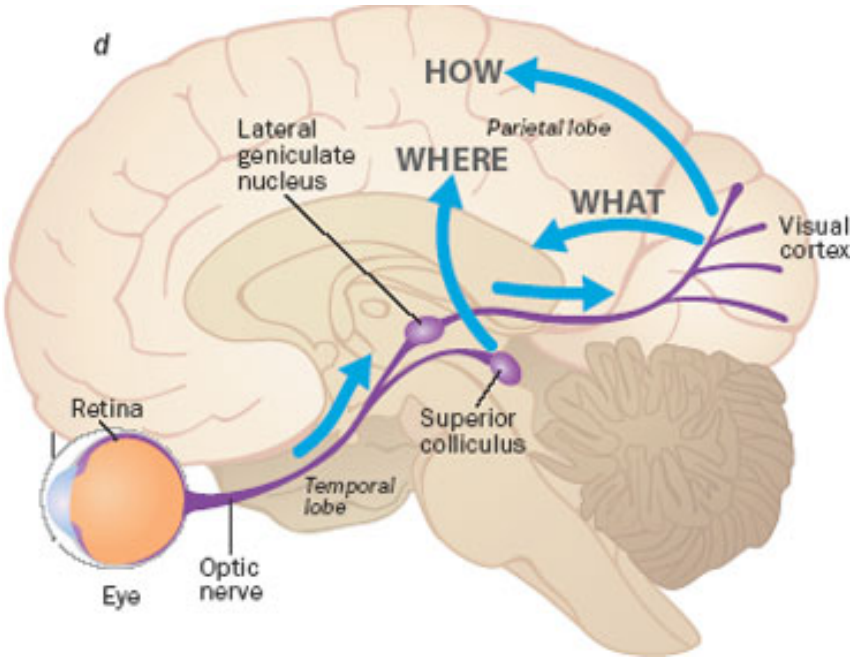
Neuroscientific Basis for Convolutional Networks

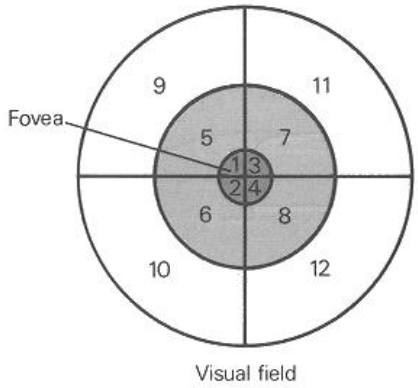
- the visual pathways: retina - LGN - V1

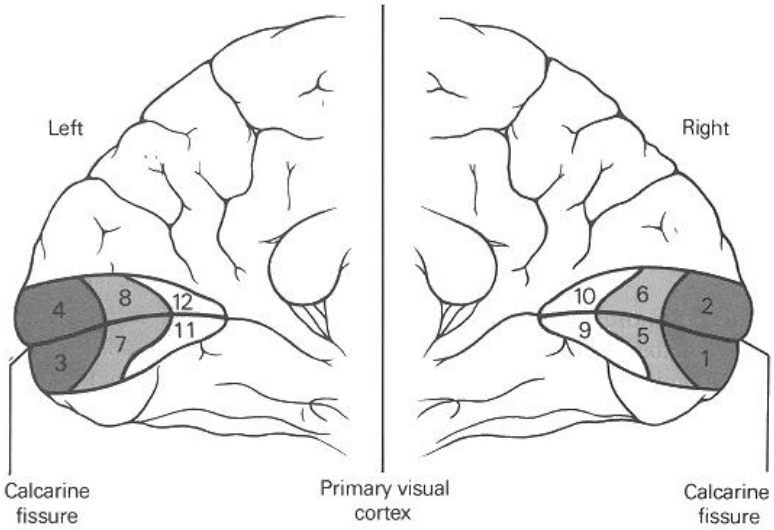


Neuroscientific Basis for Convolutional Networks

- the visual pathways: retina - LGN - V1
- V1 has a retinotopic organization







All starts in the retina

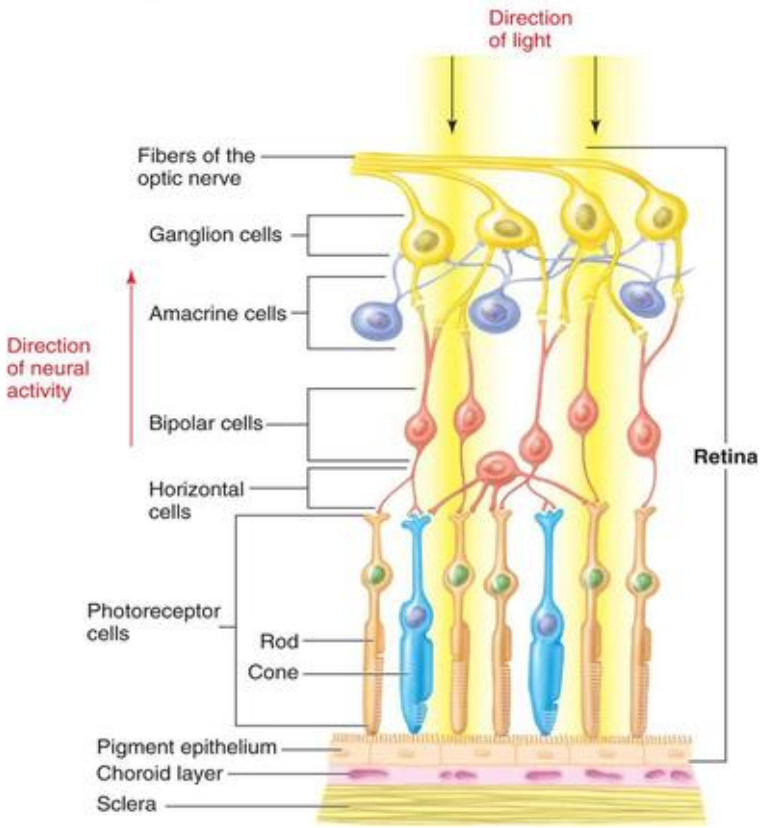


Figure: Layers of the retina

Hubel's cat

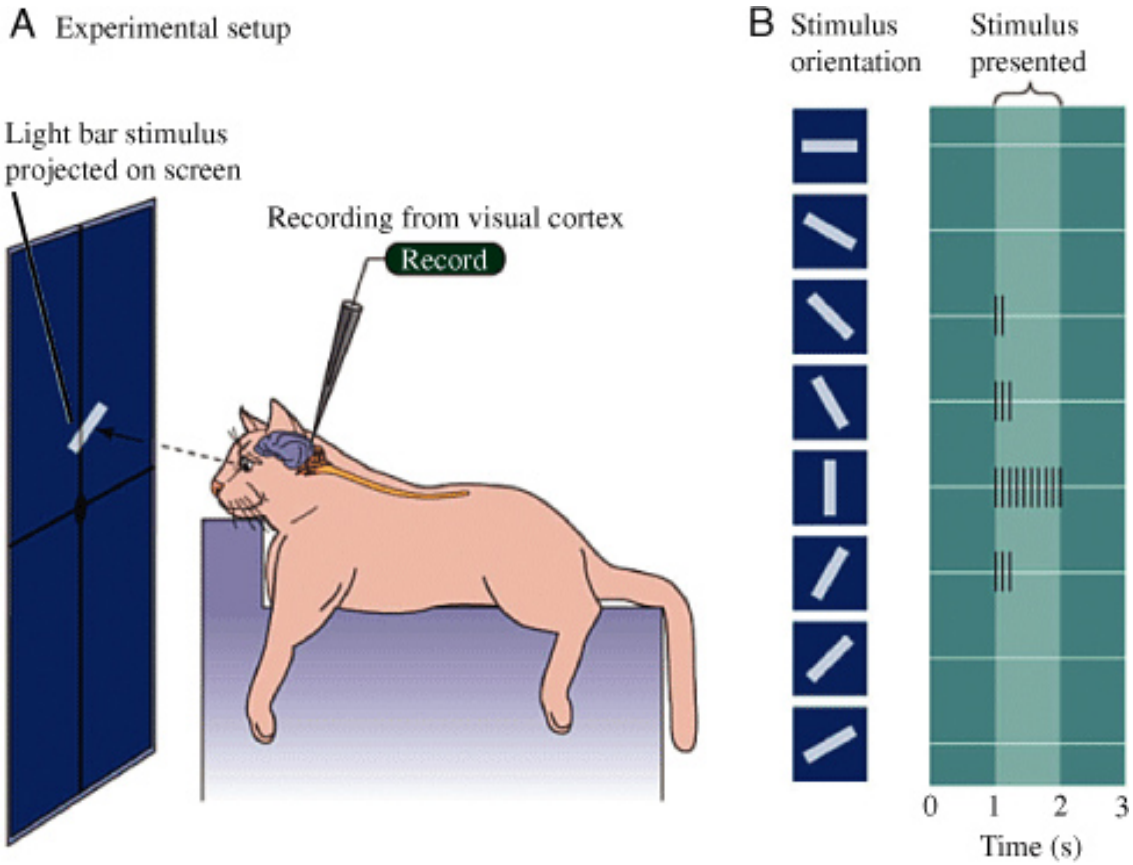


Figure: Layers of the retina

From simple to complex cells

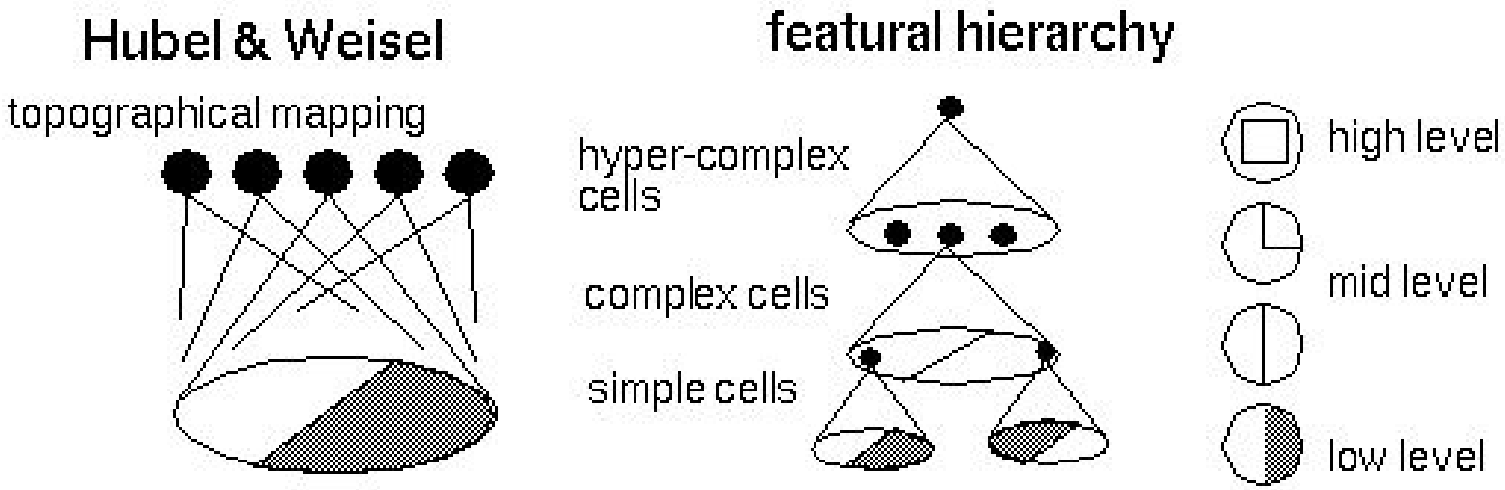


Figure: From simple to complex cells

Today's Outline

1. Object recognition
2. Convolutional Networks
   - Neuromotivation
   - Architecture
   - LeNet5
3. The Breakthrough on ImageNet
4. Understanding and Visualizing Convolutional Networks
5. Pushing further the State-of-the-Art
6. Fooling Convolutional Nets

3D Volumes of Neurons

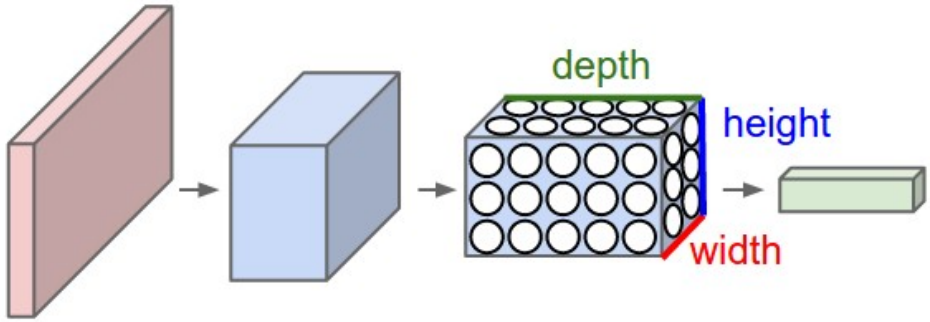


Figure: 3D volumes (image from `http://cs231n.github.io/`)

Architectures

- layers:
  - convolutional
  - ReLU
  - pooling / subsampling

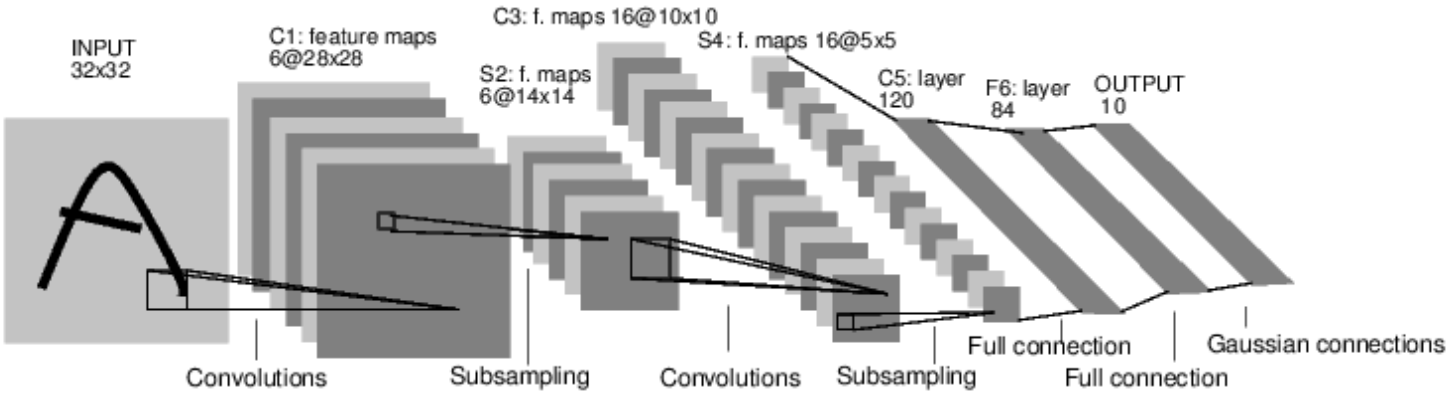


Figure: LeNet 5 Architecture

Convolution layers

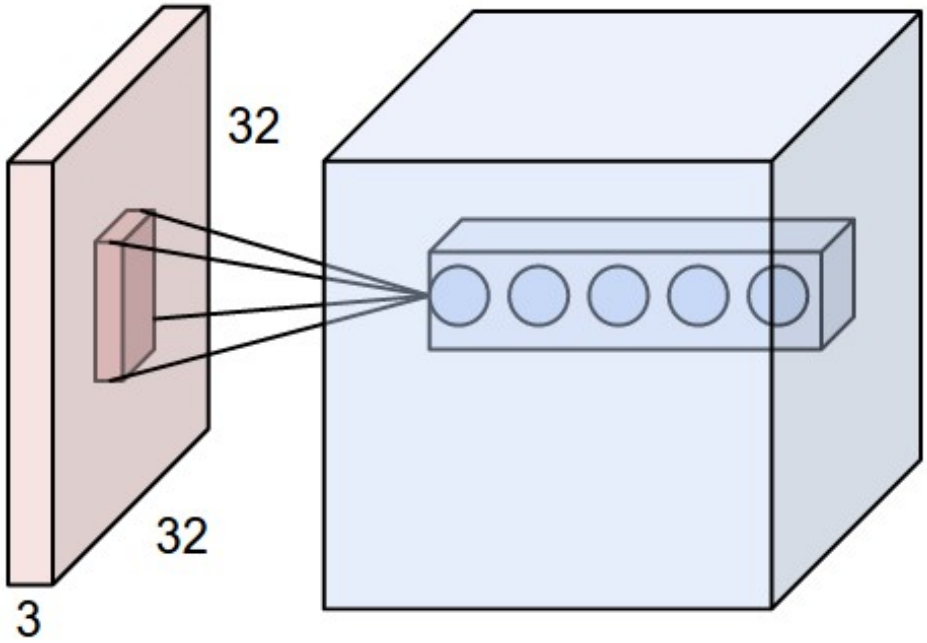


Figure: Local connectivity (image from `http://cs231n.github.io`)

Principles of Convolution Layers

- local connectivity; sparse connectivity
- parameter sharing
- equivariant representation

Convolution Kernel

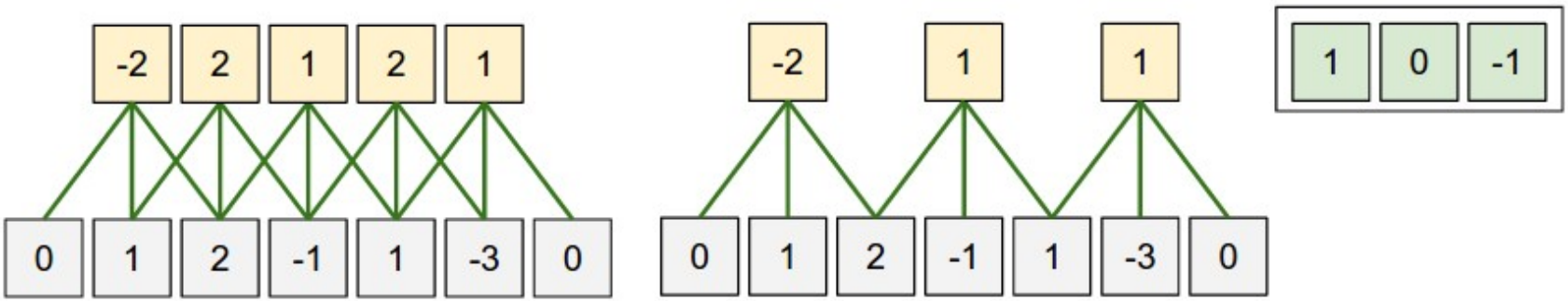
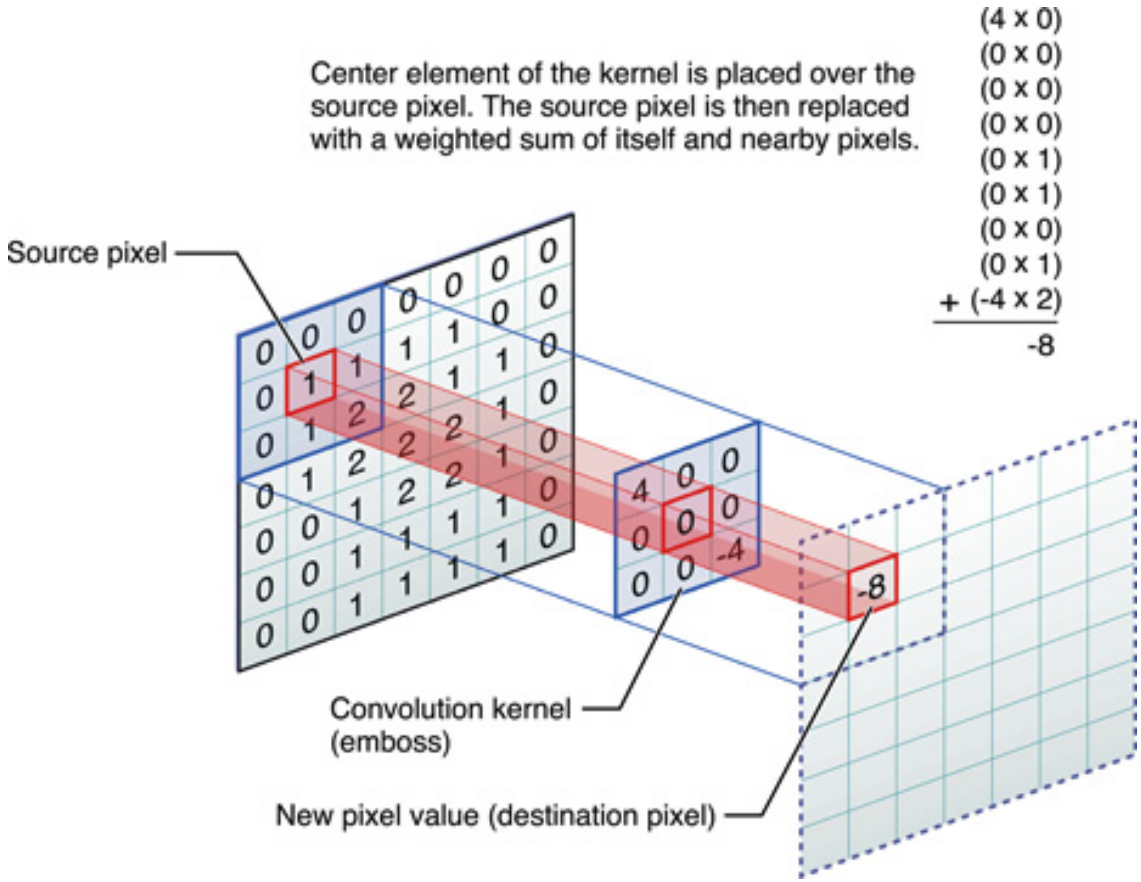


Figure: The convolution kernel

Architecture Parameters

- width $\times$ height
- stride



Pooling

- different types of pooling:
  - max pooling
  - average pooling
  - fractional pooling
- pooling layers have no parameters

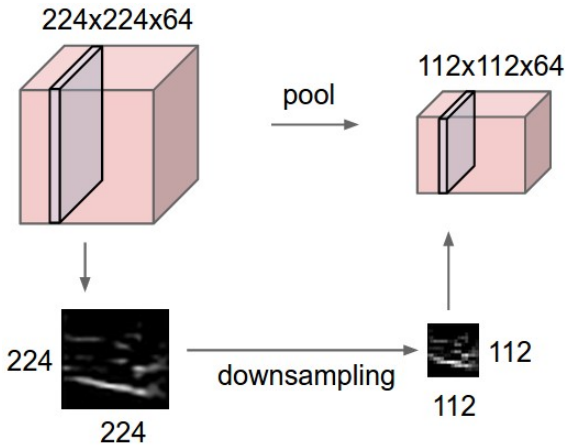


Figure: Pooling (image from `http:cs231n.github.io`)

Max pooling

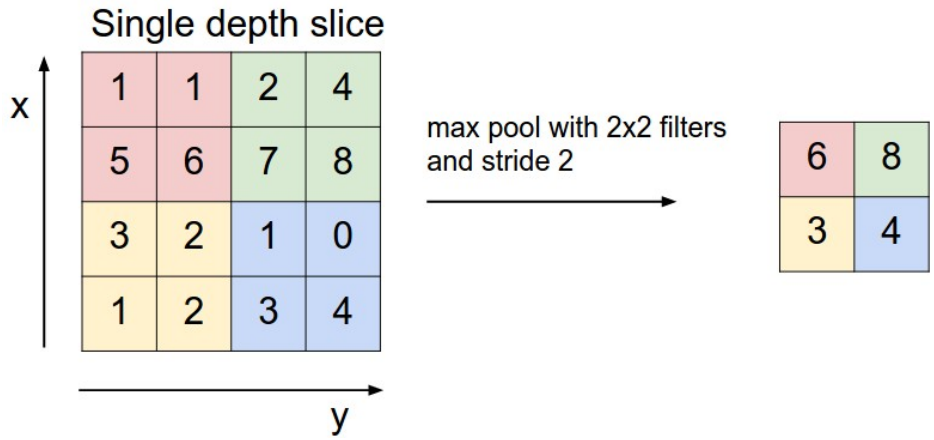
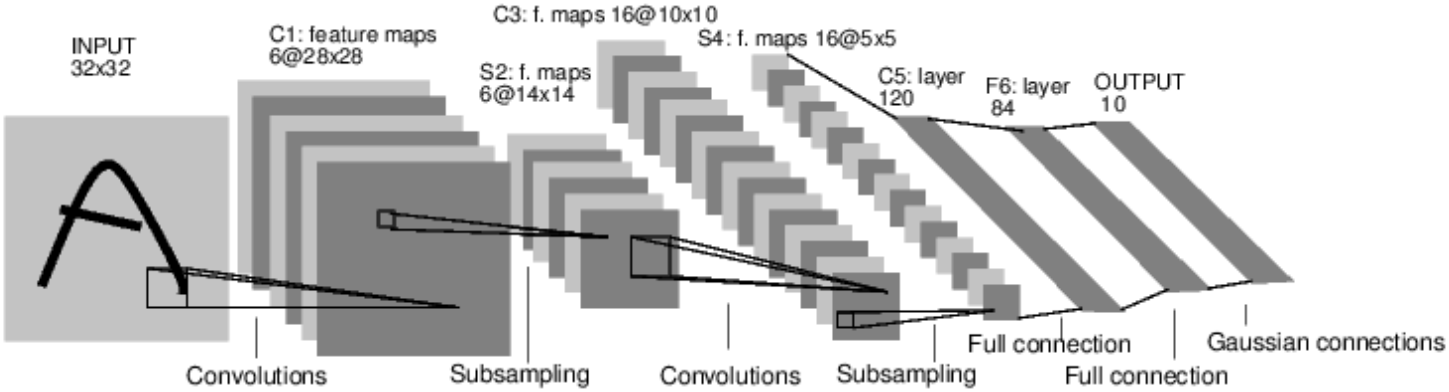


Figure: Max pooling (image from `http:cs231n.github.io`)

Today's Outline

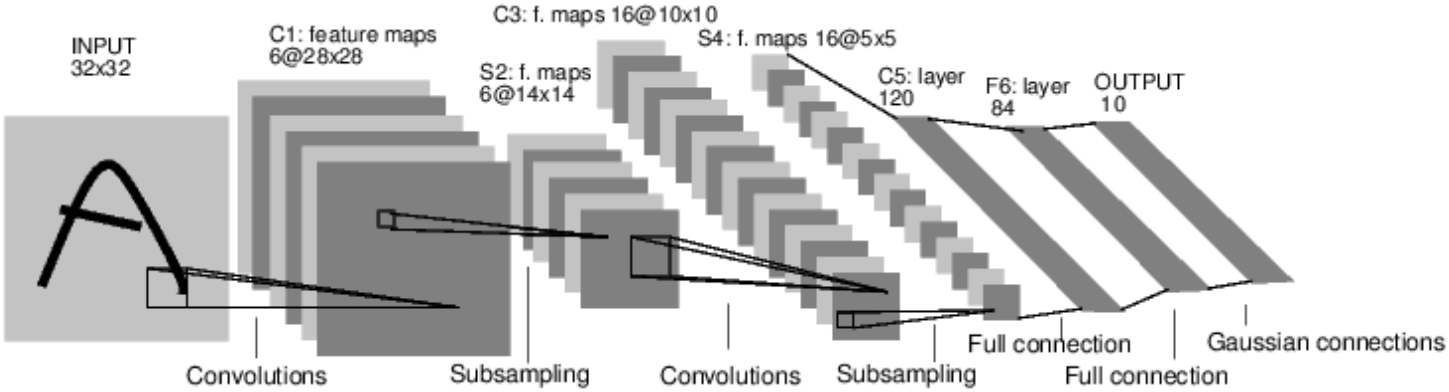
LeNet5



First layer: Convolutional

- input: 1 map of 32×32 pixels
- 6 feature maps with a 5×5 kernel
- number of parameters:
- output:

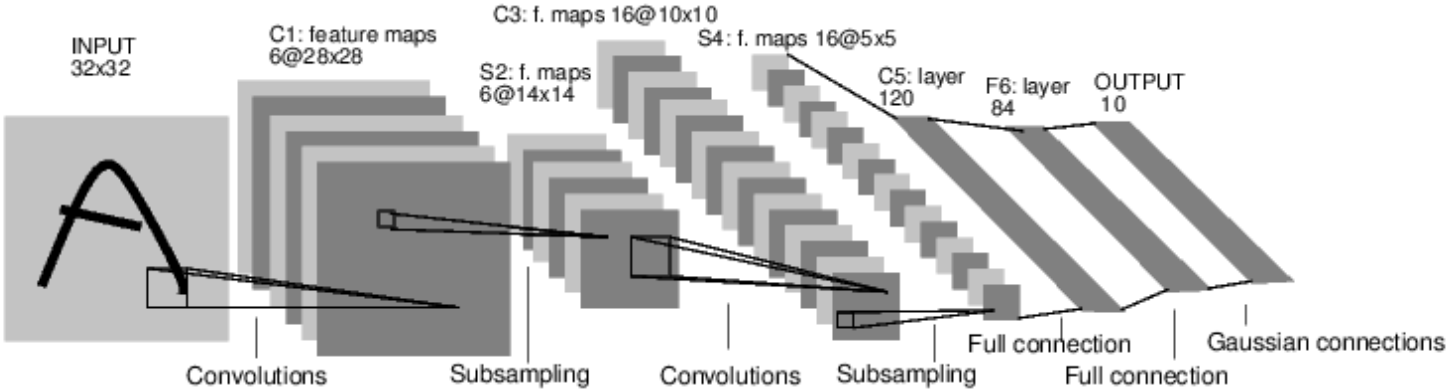
LeNet5



First layer: Convolutional

- input: 1 map of 32×32 pixels
- 6 feature maps with a 5×5 kernel
- number of parameters: $6 \times (5 \times 5 + 1)$
- output: 6 maps $\times 28 \times 28$

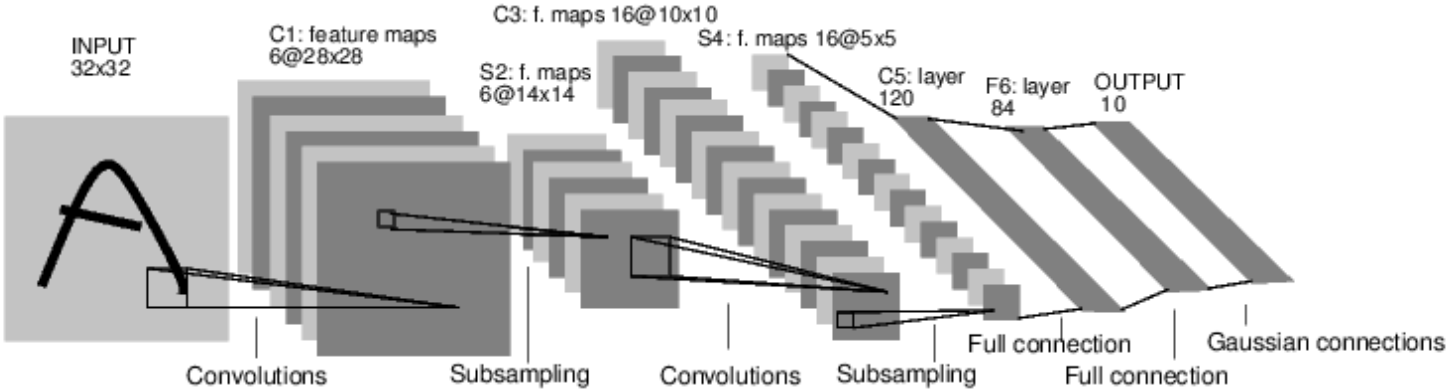
LeNet5



Second layer: Max pooling

- input: 6 maps of 28×28 pixels
- 2×2 non-overlapping receptive fields
- number of parameters:
- output:

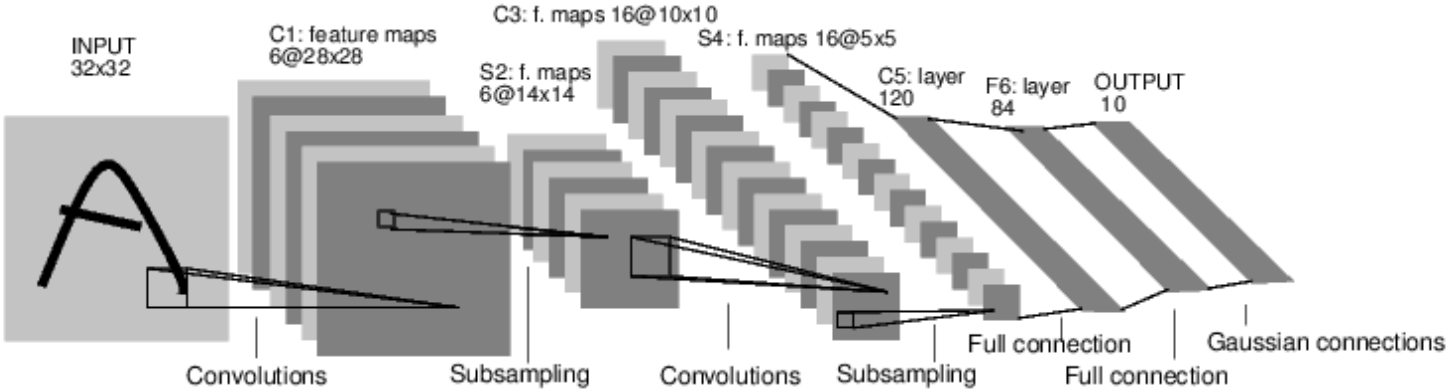
LeNet5



Second layer: Max pooling

- input: 6 maps of 28×28 pixels
- 2×2 non-overlapping receptive fields
- number of parameters: 0
- output: 6 maps $\times 14 \times 14$

LeNet5



Third layer: Convolutional

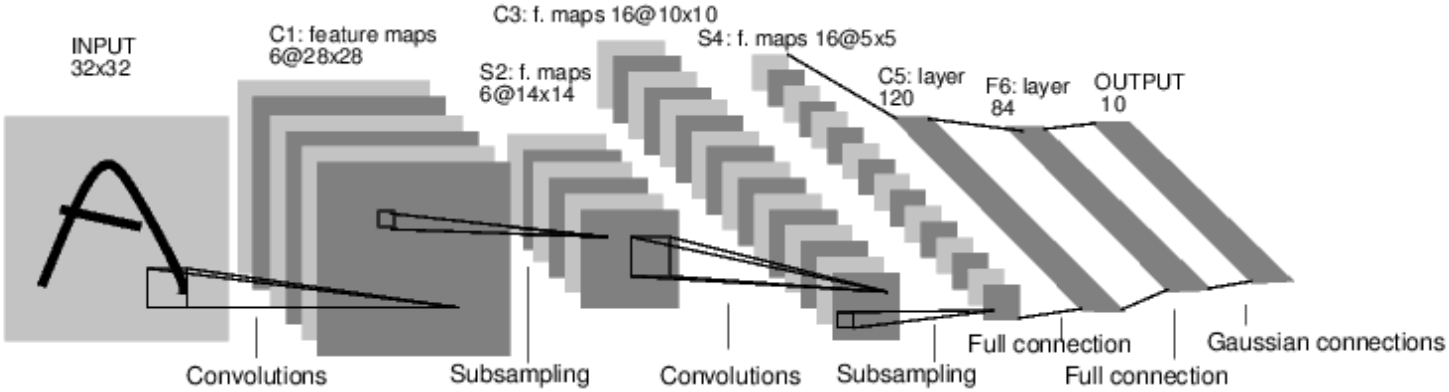
- input: 6 feature maps of 14×14 pixels
- 16 feature maps with a 5×5 kernel
- number of parameters: $\sum_{i=1}^{16} conn_i \times 5 \times 5 + 1$

Table 1. The Interconnection of the S2 Layer to C3 Layer [5]

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

- output:

LeNet5



Third layer: Convolutional

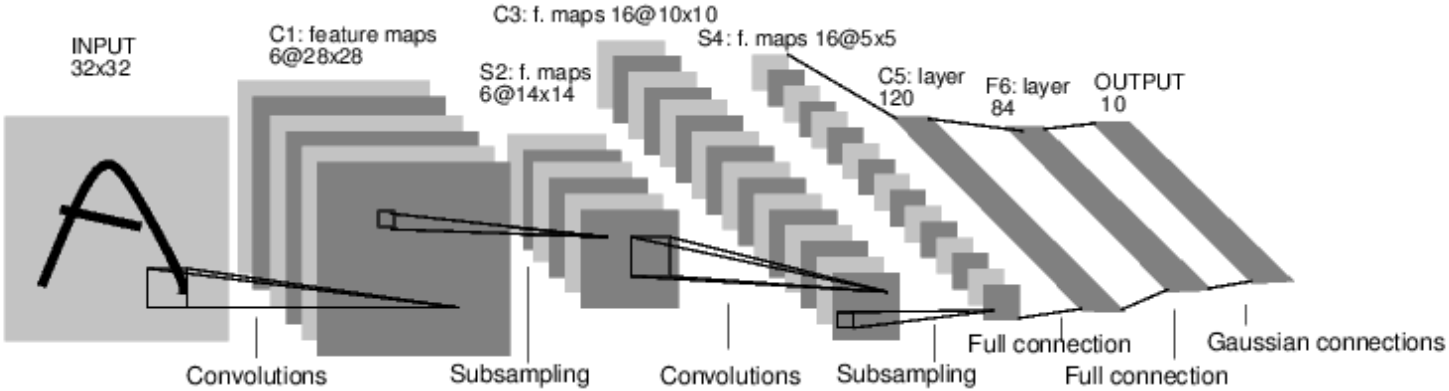
- input: 6 feature maps of 14×14 pixels
- 16 feature maps with a 5×5 kernel
- number of parameters: $\sum_{i=1}^{16} conn_i \times 5 \times 5 + 1$

Table 1. The Interconnection of the S2 Layer to C3 Layer [5]

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

- output: 16 maps $\times 10 \times 10$

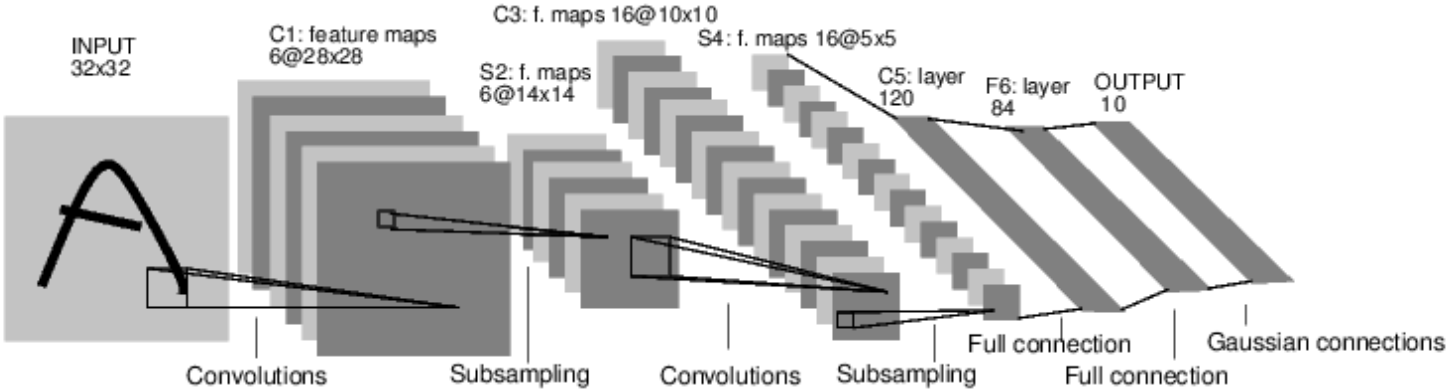
LeNet5



Fourth layer: Max pooling

- input: 16 maps of 10×10 pixels
- 2×2 non-overlapping receptive fields
- number of parameters:
- output:

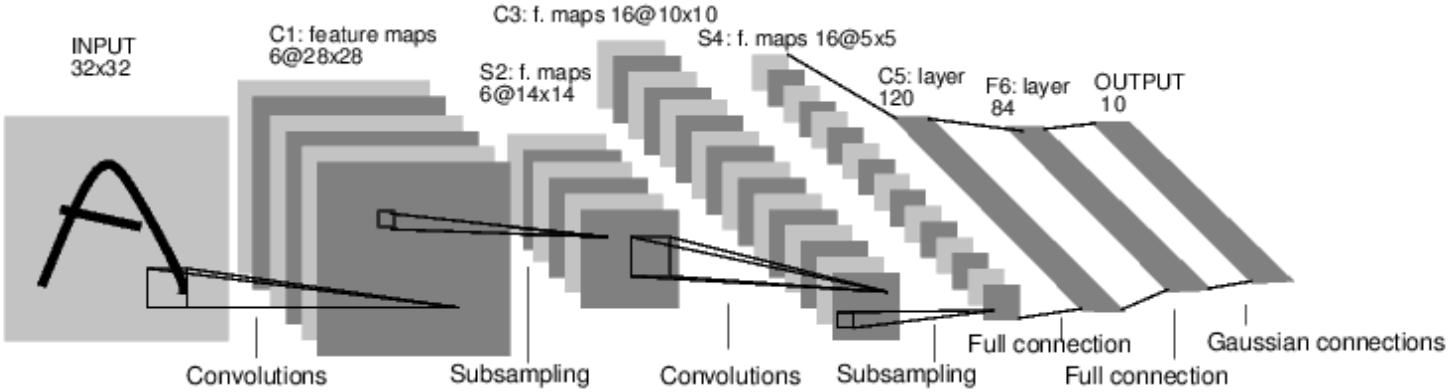
LeNet5



Fourth layer: Max pooling

- input: 16 maps of 10×10 pixels
- 2×2 non-overlapping receptive fields
- number of parameters: 0
- output: 16 maps $\times 5 \times 5 = 400 \textit{units}$

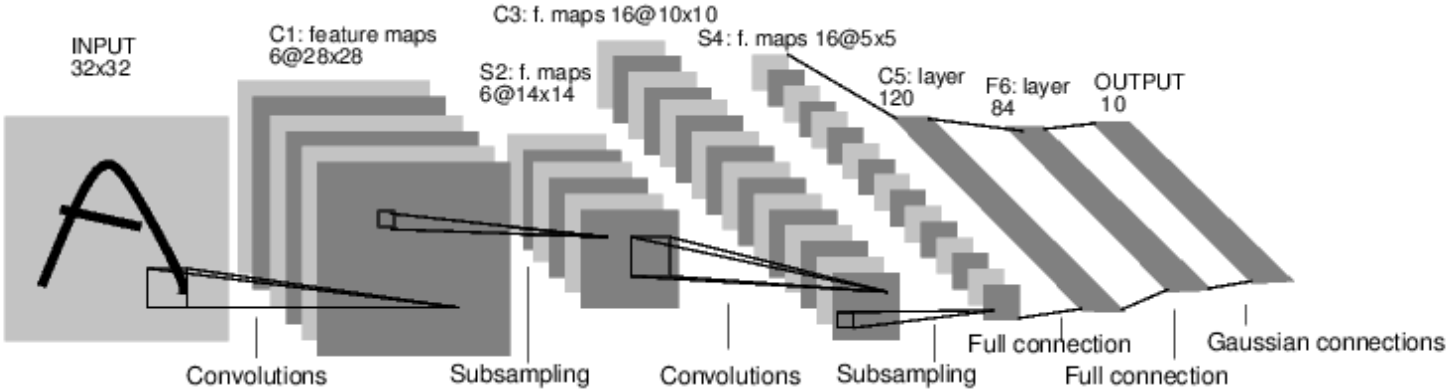
LeNet5



Fifth layer: Fully connected

- input: 16 maps of 5×5 pixels
- 120 fully connected units
- number of parameters:
- output:

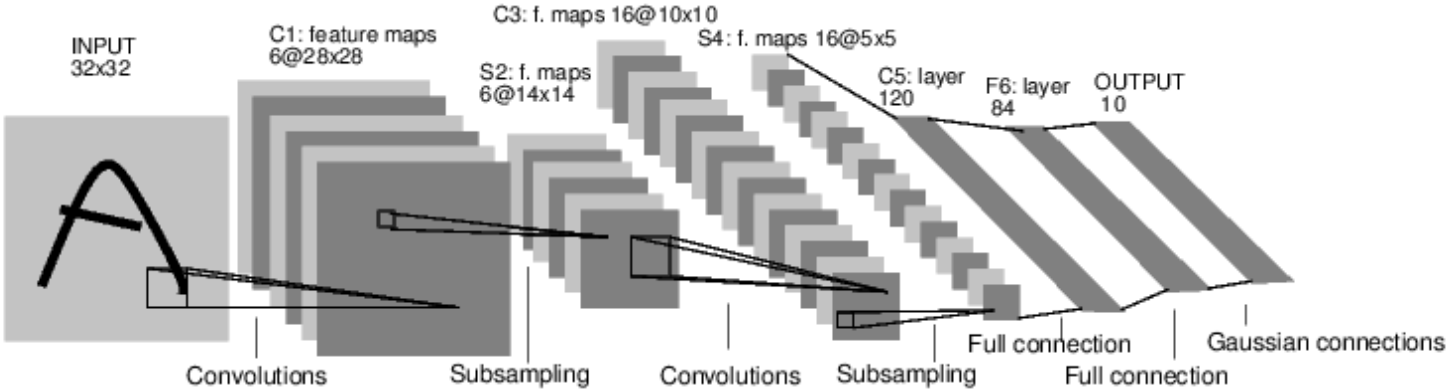
LeNet5



Fifth layer: Fully connected

- input: 16 maps of 5×5 pixels
- 120 fully connected units
- number of parameters: $120 \times (16 \times 5 \times 5 + 1)$
- output: 120 units

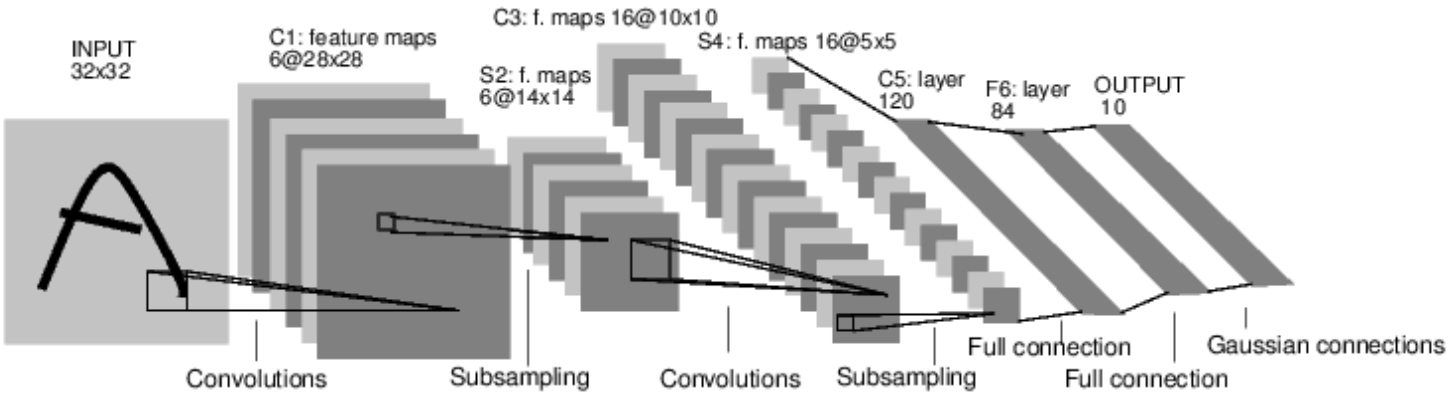
LeNet5



Sixth layer: Fully connected

- input: 120 units
- 84 fully connected units
- number of parameters:
- output:

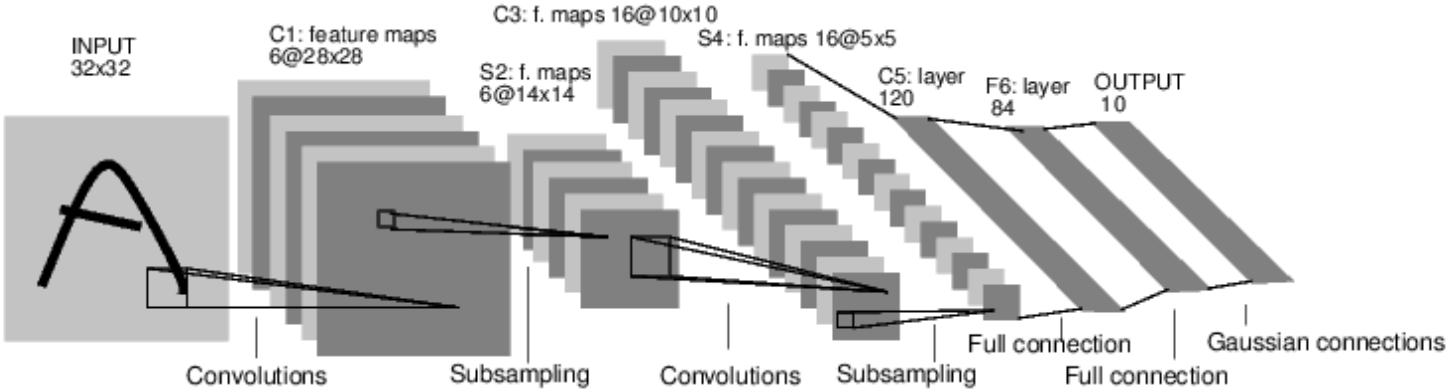
LeNet5



Sixth layer: Fully connected

- input: 120 units
- 84 fully connected units
- number of parameters: $84 \times (120 + 1)$
- output: 84 units

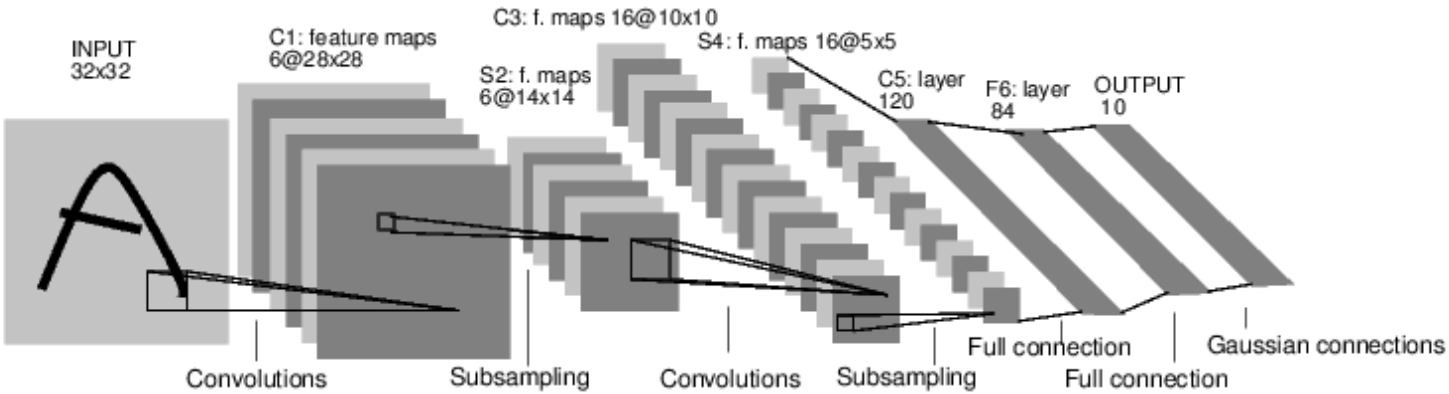
LeNet5



Seventh layer: Fully connected

- input: 84 units
- 10 fully connected units (one for each class)
- number of parameters:
- output:

LeNet5

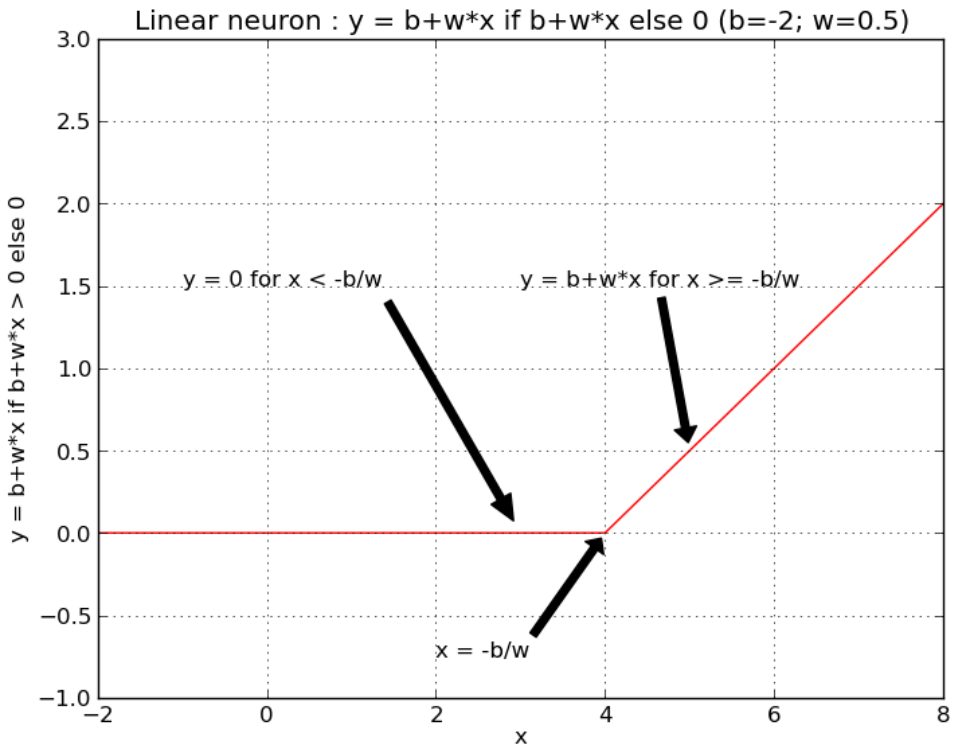


Seventh layer: Fully connected

- input: 84 units
- 10 fully connected units (one for each class)
- number of parameters: $10 \times (84 + 1)$
- output: 10 units

ReLU units

$$a_k = \begin{cases} z_k & \text{if } z_k \geq 0 \\ 0 & \text{if } z_k < 0 \end{cases} \tag{1}$$



What to read?

- [BGC15, Chapter 9]

Today's Objectives

Our objectives for today's lecture are to . . .

- . . . get a **deeper understanding** of how convolutional nets work;
- . . . get familiar with **recent research** on convolutional networks;
- . . . learn how to define and train convolutional networks in **Torch**.

Today's Outline

1. Object recognition
2. Convolutional Networks
3. The Breakthrough on ImageNet
4. Understanding and Visualizing Convolutional Networks
5. Pushing further the State-of-the-Art
6. Fooling Convolutional Nets
7. Spatial Transformer Networks

Today's Outline

Deep Convolutional Neural Networks

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, *Imagenet classification with deep convolutional neural networks*, Advances in neural information processing systems, 2012, pp. 1097–1105

First major result in deep learning.

The Architecture and Some Tricks

- similar to LeNet [LBBH98], but deeper
- 5 convolutional layers + 3 fully-connected layers
- ReLU leads to faster training than tanh
- trained on multiple GPUs
- local response normalization

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

  - implements a form of lateral inhibition
- overlapping pooling

AlexNet

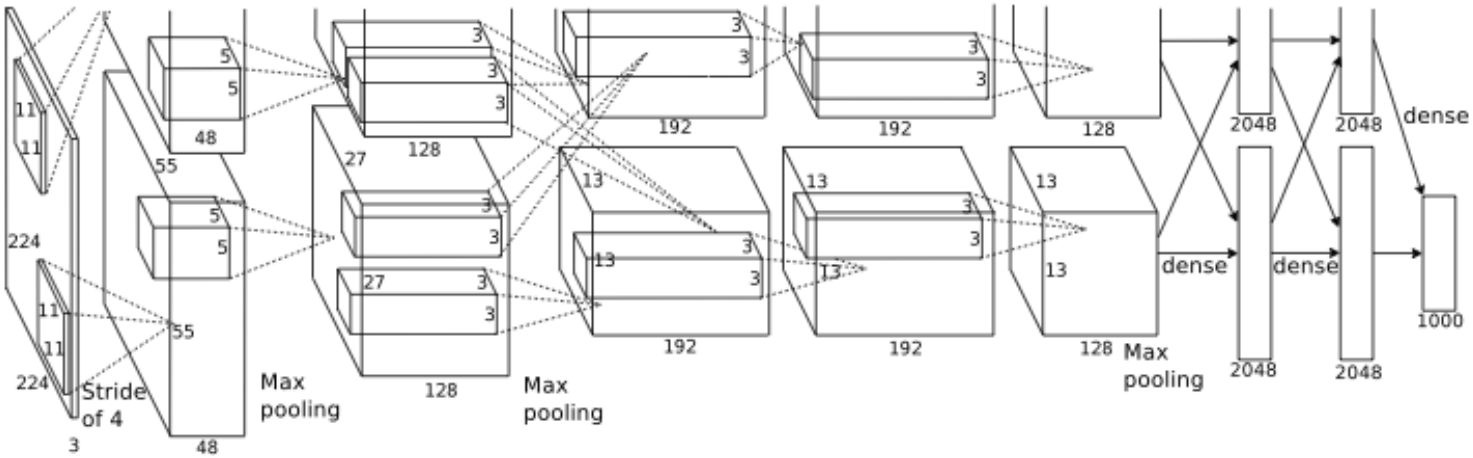
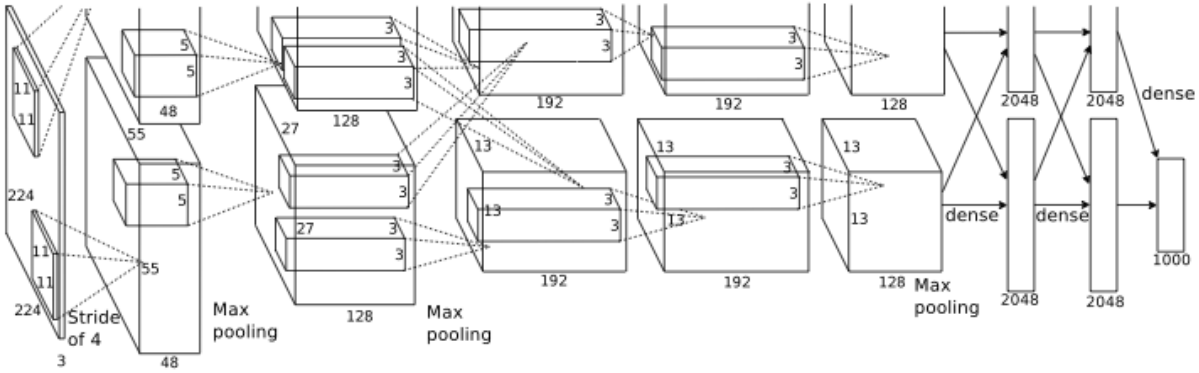


Figure: Image from [KSH12]

- 60 million parameters!!

AlexNet - details



- Input $3 \times 224 \times 224$
- Conv1: 96 kernels of size $3 \times 11 \times 11$, stride 4
- Conv2: 256 kernels of size $48 \times 5 \times 5$
- Conv3: 384 kernels of size $256 \times 3 \times 3$
- Conv4: 384 kernels of size $192 \times 3 \times 3$
- Conv5: 256 kernels of size $192 \times 3 \times 3$
- FC6, FC7: 4096 neurons

Reducing Overfitting

- data augmentation: patches
  - extracting random 224×224 patches
  - add horizontal reflections
  - for test: average predictions on 10 patches (four corners + center) + horizontal reflection
- altering the intensities of the RGB channels
  - perform PCA for all RGB pixels on the whole ImageNet
  - add multiples of the found principal components:

$$I'_{x,y} = I_{x,y} + [\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3]^\mathsf{T}[\alpha_1\lambda_1, \alpha_2\lambda_2, \alpha_3\lambda_3]$$

    where $\alpha_i \sim \mathcal{N}(0, 0.1)$ and $\mathbf{p}_i, \lambda_i$ are the eigenvalues and eigenvectors of the covariance matrix of the RGB pixel values
- dropout

Learning

- Stochastic Gradient Descent
- 128 example in each mini-batch
- 0.9 momentum
- 0.0005 weight decay

$$\Delta \mathbf{w}_{i+1} = 0.9 \cdot \Delta \mathbf{w}_i - 0.0005 \cdot \eta \cdot \mathbf{w}_i - \eta \cdot \left\langle \left. \frac{\partial E}{\partial \mathbf{w}} \right|_{\mathbf{w}_i} \right\rangle_{D_i}$$

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \Delta \mathbf{w}_{i+1}$$

- $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, 0.01 \cdot \mathbf{I})$
- *bias* $= 0$ for layers 2, 4, 5
- *bias* $= 1$ for the fully connected layers

Kernels on the first convolution

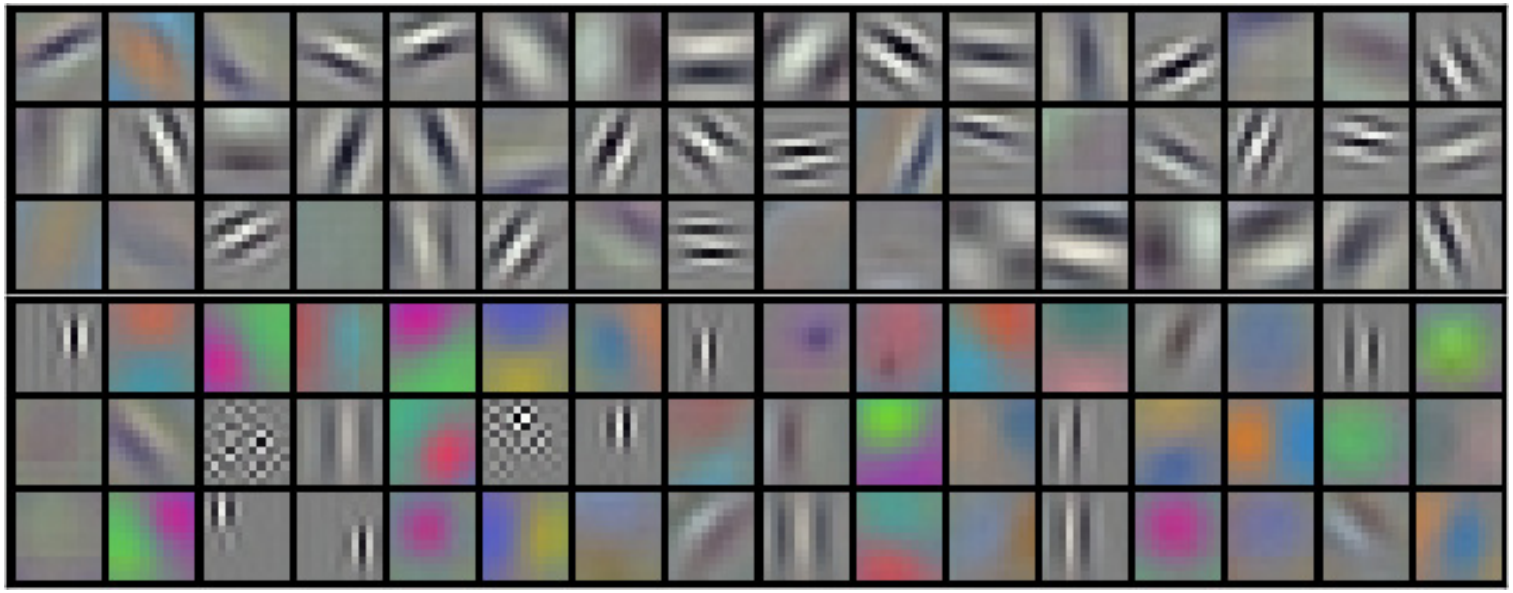


Figure: The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2. (Image from [KSH12])

Results

- ILSVRC 2012
  `http://image-net.org/challenges/LSVRC/2012/results`
- Task 1 (Classification, 5 guesses):

Solution	**Error**
AlexNet	0.16422
ISI (Tokyo)	0.26172

- Task 2 (Localization)

Solution	**Error**
AlexNet	0.341905
VGG (Oxford)	0.500342

Classification examples

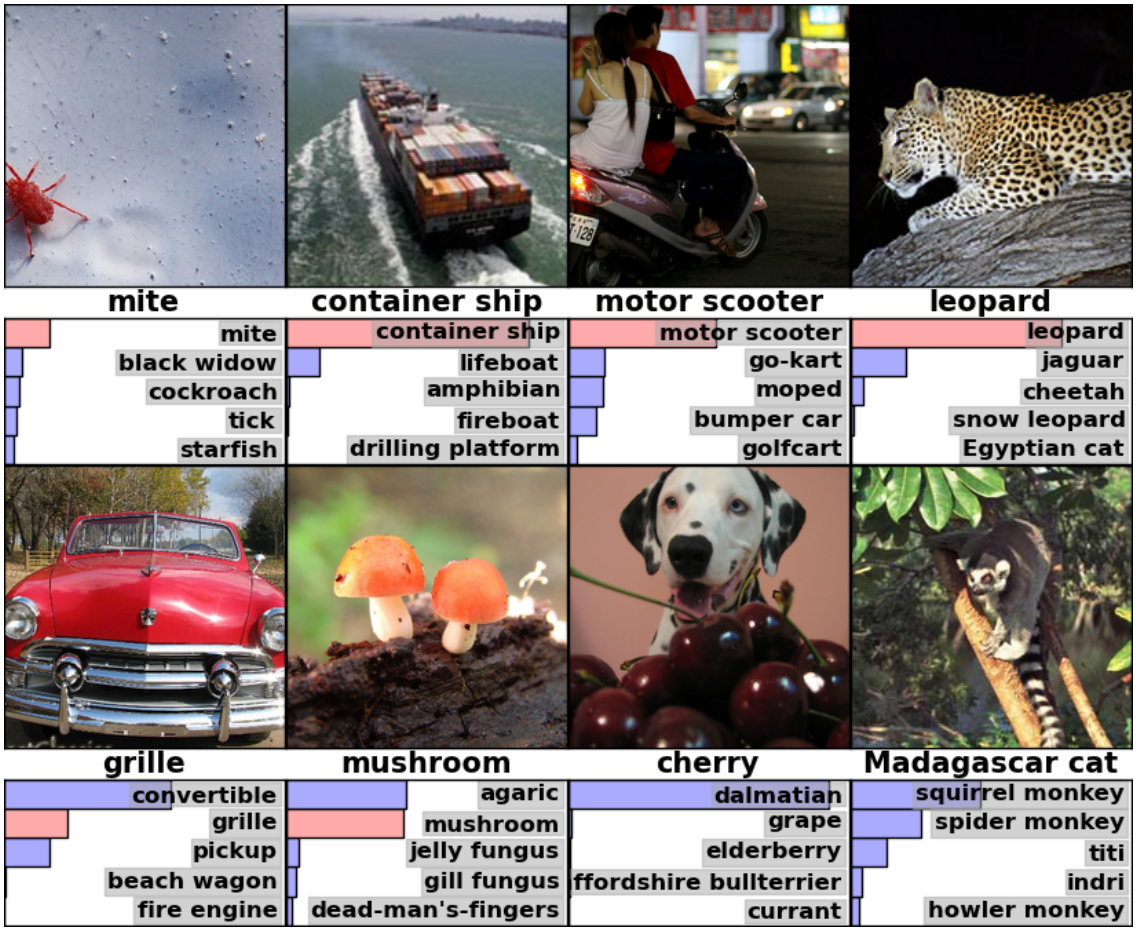
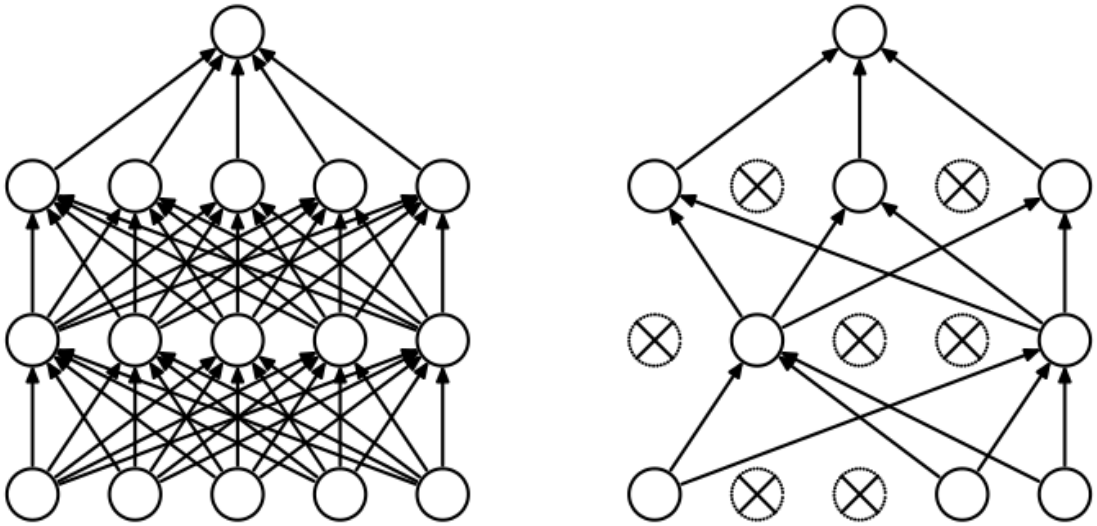


Figure: Top five most probable labels for some examples. (image from [KSH12])

Dropout Technique

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, *Dropout: A simple way to prevent neural networks from overfitting*, The Journal of Machine Learning Research **15** (2014), no. 1, 1929–1958

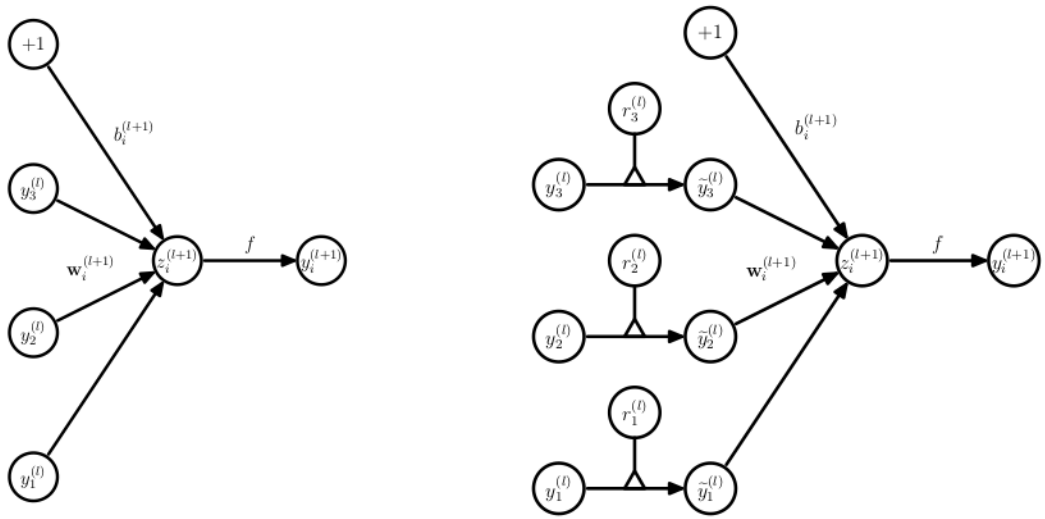
Dropout



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

Figure: Dropout Technique (image from [SHK+14])

Dropout Architecture



(a) Standard network

(b) Dropout network

Figure 3: Comparison of the basic operations of a standard and dropout network.

Figure: Dropout Technique (image from [SHK+14])

$$r_j^{(l)} \sim \text{Bernoulli}(p) \qquad \tilde{\mathbf{y}} = \mathbf{r}^{(l)} * \mathbf{y}^{(l)}$$

Effects of dropout

- reduces overfitting
- methods using dropout achieve state-of-the-art results on SVHN, ImageNet, CIFAR-100, MNIST
- each training case trains a different network
- drawbacks: increased training time

Comparison of different regularizers

Method	Test Classification error %
L2	1.62
L2 + L1 applied towards the end of training	1.60
L2 + KL-sparsity	1.55
Max-norm	1.35
Dropout + L2	1.25
Dropout + Max-norm	**1.05**

Table 9: Comparison of different regularization methods on MNIST.

Figure: (image from [SHK+14])

DropConnect

Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus, *Regularization of neural networks using dropconnect*, Proceedings of the 30th International Conference on Machine Learning (ICML-13), 2013, pp. 1058–1066

DropConnect

- DropConnect is a generalization of Dropout (if $f(0) = 0$)
- each connection is dropped with probability $1 - p$
- the sparsity is on the weights $\mathbf{W}$

$$\mathbf{a} = f\left((\mathbf{M} \star \mathbf{W})\,\mathbf{x}\right)$$

where $M_{ij} \sim Bernoulli(p)$

- DropConnect realizes a mixture of models (different architectures)

DropConnect

neuron	model	error(%) 5 network	voting error(%)
relu	No-Drop	1.62 ± 0.037	1.40
	Dropout	1.28 ± 0.040	1.20
	DropConnect	1.20 ± 0.034	**1.12**
sigmoid	No-Drop	1.78 ± 0.037	1.74
	Dropout	1.38 ± 0.039	**1.36**
	DropConnect	1.55 ± 0.046	1.48
tanh	No-Drop	1.65 ± 0.026	1.49
	Dropout	1.58 ± 0.053	1.55
	DropConnect	1.36 ± 0.054	**1.35**

Table 2. MNIST classification error rate for models with two fully connected layers of 800 neurons each. No data augmentation is used in this experiment.

Figure: Comparison of different regularization techniques on MNIST

Today's Outline

Matthew D. Zeiler and Rob Fergus, *Visualizing and understanding convolutional networks*, CoRR **abs/1311.2901** (2013)

Idea

- analyse the learned filters of a convolutional net
- analyse the activations on convolutional layers when an image is presented to the network
- attach a deconvolution network to the trained CNN
  - it reconstructs the activations from the last layer back to the input pixel space
  - unpool, rectify, filter

Deconvolution network

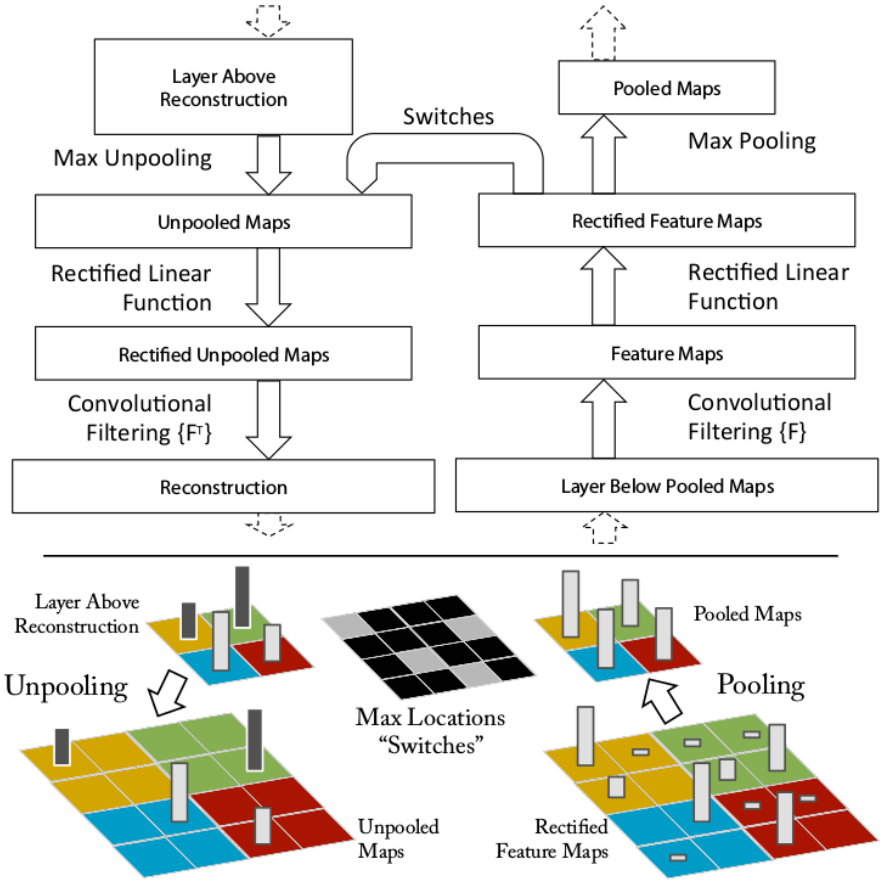
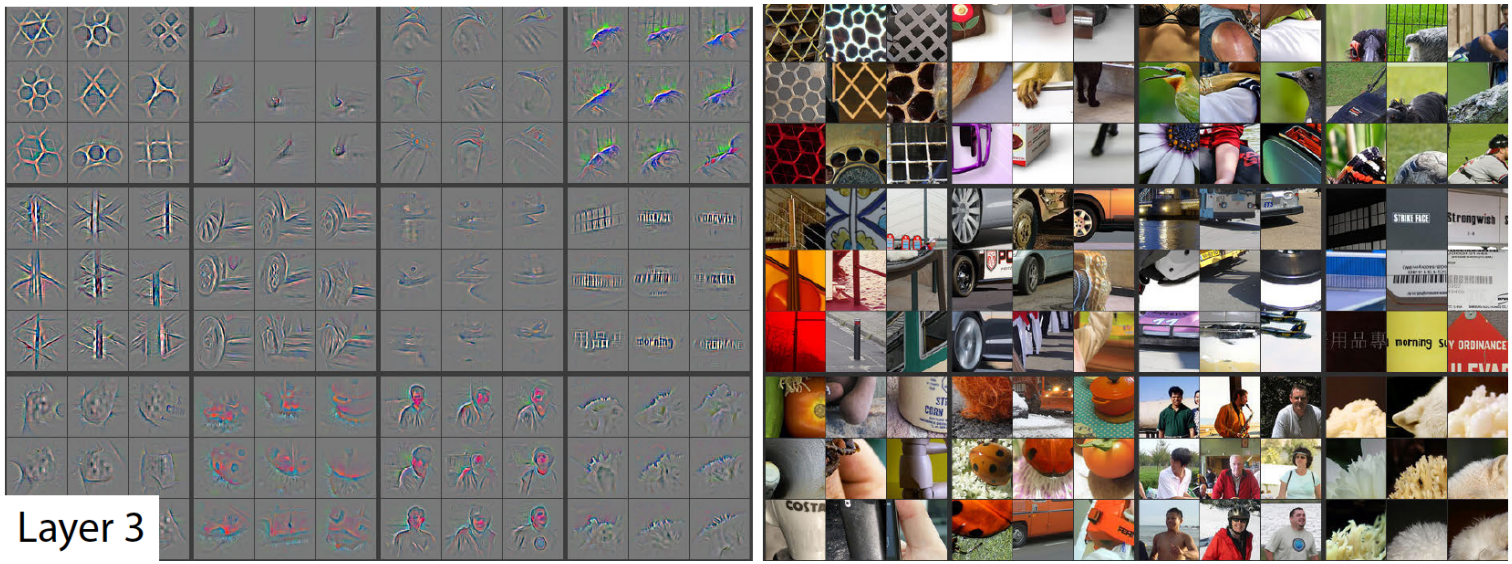


Figure: A CNN and its deconvolution network

Analysing AlexNet

- Problems:
  - visualizations of the first layer filters reveals that a few of them dominate
  - first layer filters are a mix of extremely high and low frequency information
  - second layer visualizations shows aliasing artifacts caused by the large stride in the first layer
- Solutions:
  - reduce the first layer filter size from 11x11 to 7x7
  - reduce the stride of the convolution

Visualization



Visualization

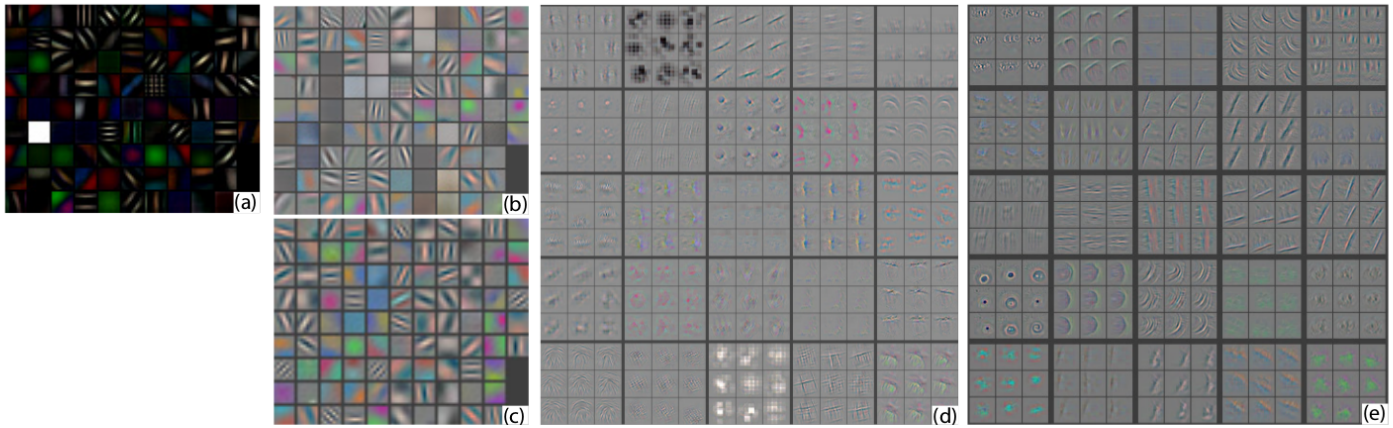


Figure 6. (a): 1st layer features without feature scale clipping. Note that one feature dominates. (b): 1st layer features from (Krizhevsky et al., 2012). (c): Our 1st layer features. The smaller stride (2 vs 4) and filter size (7x7 vs 11x11) results in more distinctive features and fewer "dead" features. (d): Visualizations of 2nd layer features from (Krizhevsky et al., 2012). (e): Visualizations of our 2nd layer features. These are cleaner, with no aliasing artifacts that are visible in (d).

Conclusions

- the CNN is classifying the object in the image, not by using the context of the image.
- the model implicitly establishes some form of correspondence between correlated features (eyes, mouth, in faces) in the layer 5 and shows lesser correlation at layer 7
- removing the fully connected layers 6 and 7 or changing their size only results in a small increase in error.

Today's Outline

Today's Outline

1. Object recognition
2. Convolutional Networks
3. The Breakthrough on ImageNet
4. Understanding and Visualizing Convolutional Networks
5. Pushing further the State-of-the-Art
   - Network in Network
   - Inception (GoogLeNet)
6. Fooling Convolutional Nets

Network in Network

Min Lin, Qiang Chen, and Shuicheng Yan, *Network in network*, CoRR **abs/1312.4400** (2013)

http://arxiv.org/abs/1312.4400

Nonlinear filter

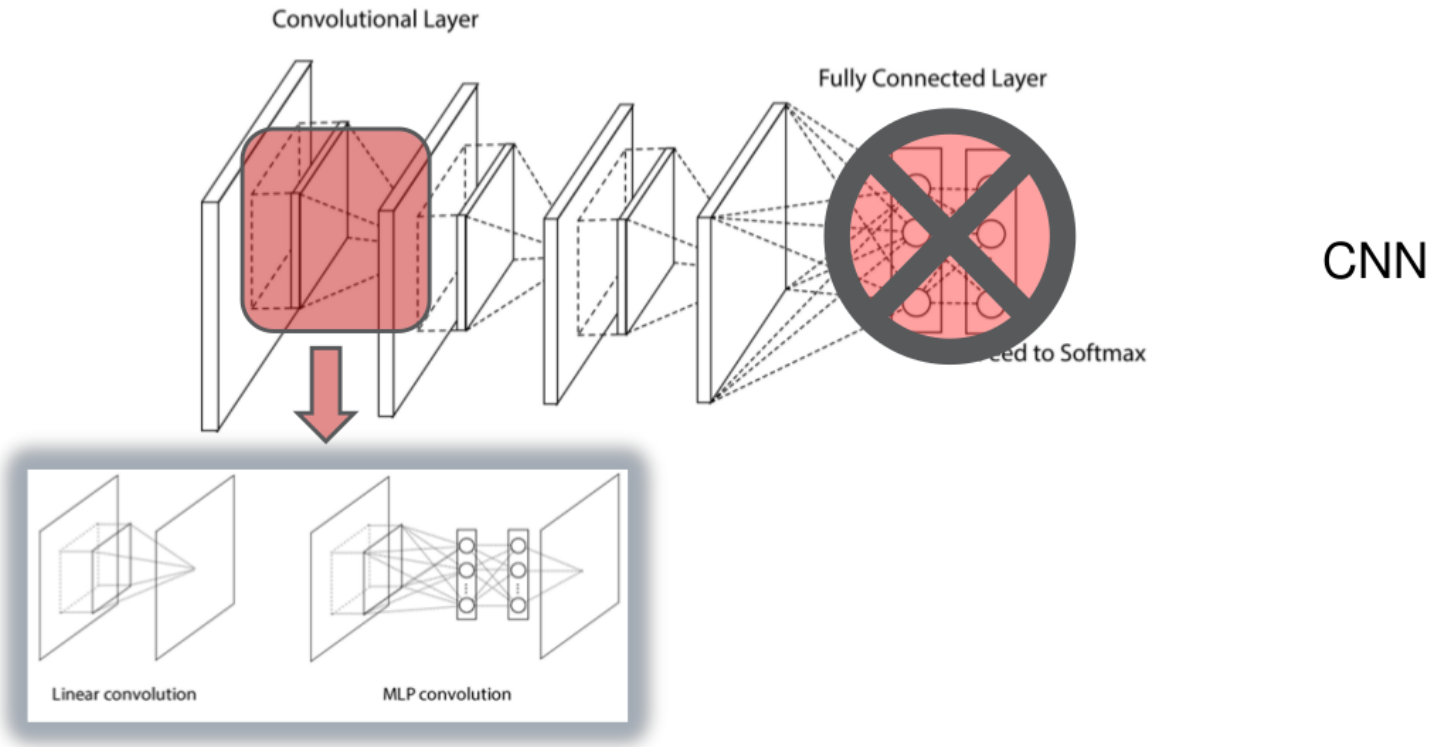


Figure: CNN with non-linear filters (image from authors' presentation at ILSVRC 2014)

Global Average Pooling

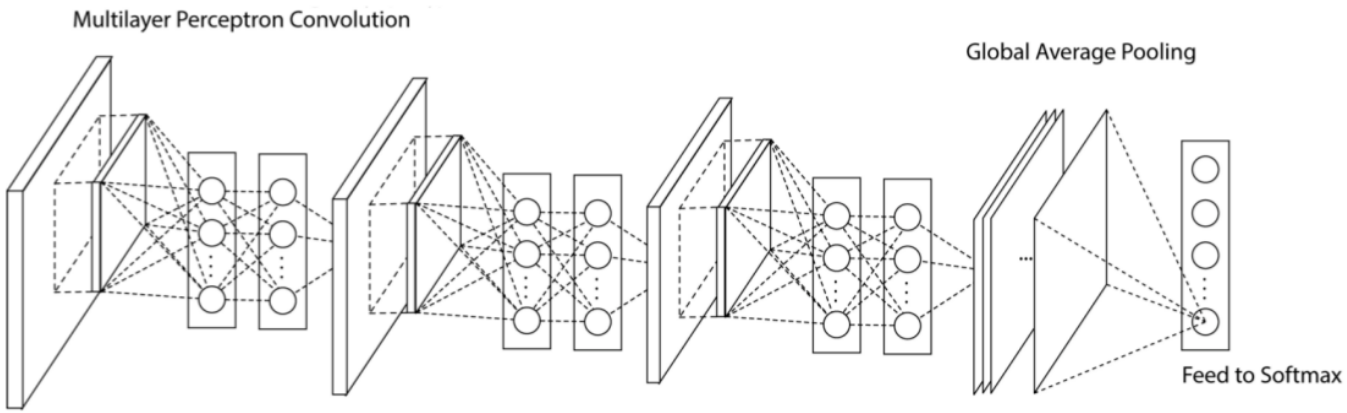


Figure: CNN with non-linear filters (image from authors' presentation at ILSVRC 2014)

- global average pooling replaces the fully connectd layers
- direct connection between categories and feature maps
- acts as a regularizer

Better abstraction (CCCP)

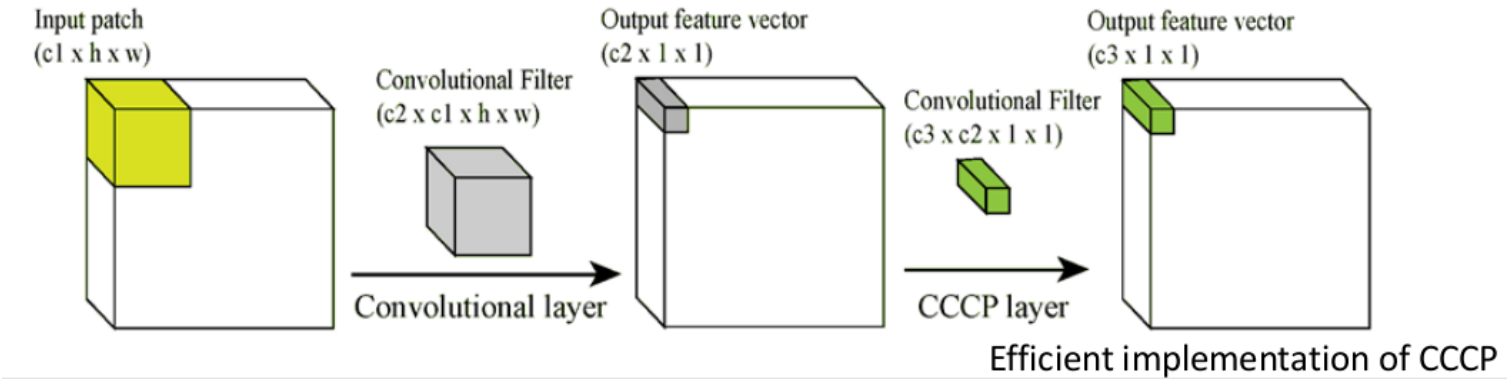


Figure: Efficient implementation of CCCP (image from authors' presentation at ILSVRC 2014)

Better abstraction (CCCP)

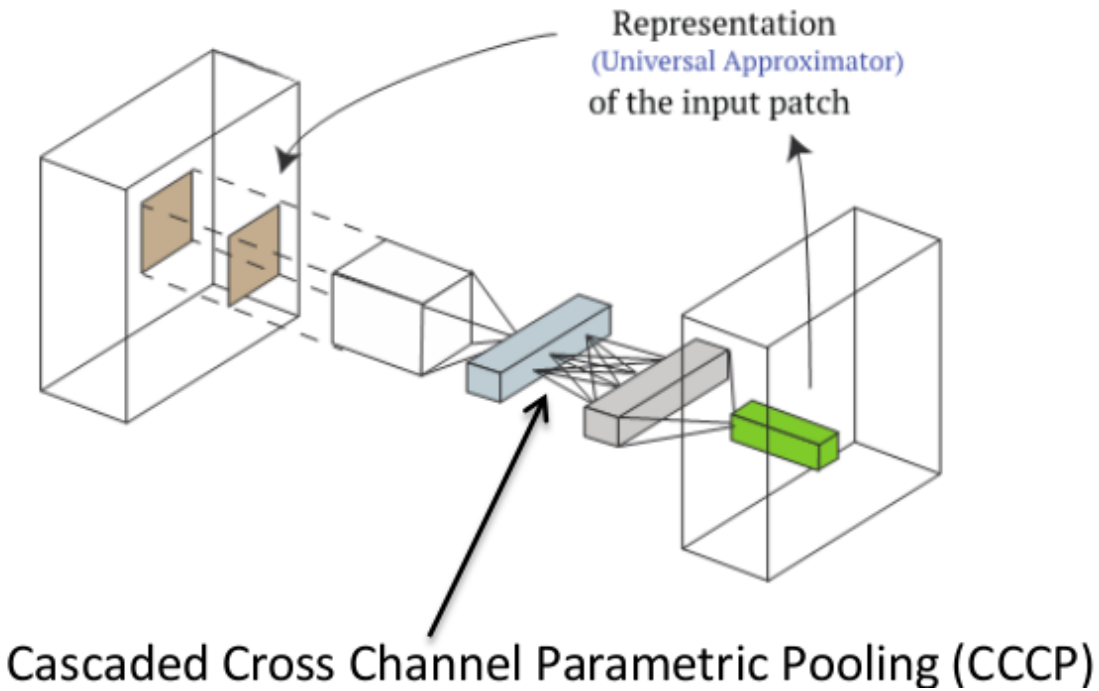


Figure: Cascaded Cross Chanel Parametric Pooling (image from authors' presentation at ILSVRC 2014)

Results

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

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

Table 2: Test set error rates for CIFAR-100 of various methods.

Method	Test Error
Learned Pooling [16]	43.71%
Stochastic Pooling [11]	42.51%
Conv. maxout + Dropout [8]	38.57%
Tree based priors [17]	36.85%
NIN + Dropout	**35.68%**

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

Method	Test Error
Stochastic Pooling [11]	2.80%
Rectifier + Dropout [18]	2.78%
Rectifier + Dropout + Synthetic Translation [18]	2.68%
Conv. maxout + Dropout [8]	2.47%
NIN + Dropout	2.35%
Multi-digit Number Recognition [19]	2.16%
DropConnect [15]	**1.94%**

Table 4: Test set error rates for MNIST of various methods.

Method	Test Error
2-Layer CNN + 2-Layer NN [11]	0.53%
Stochastic Pooling [11]	0.47%
NIN + Dropout	0.47%
Conv. maxout + Dropout [8]	**0.45%**

Today's Outline

1. Object recognition
2. Convolutional Networks
3. The Breakthrough on ImageNet
4. Understanding and Visualizing Convolutional Networks
5. Pushing further the State-of-the-Art
   - Network in Network
   - Inception (GoogLeNet)
6. Fooling Convolutional Nets

Going Deeper

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, *Going deeper with convolutions*, CoRR **abs/1409.4842** (2014)

`http://arxiv.org/abs/1409.4842`

Inception Module - Naive version

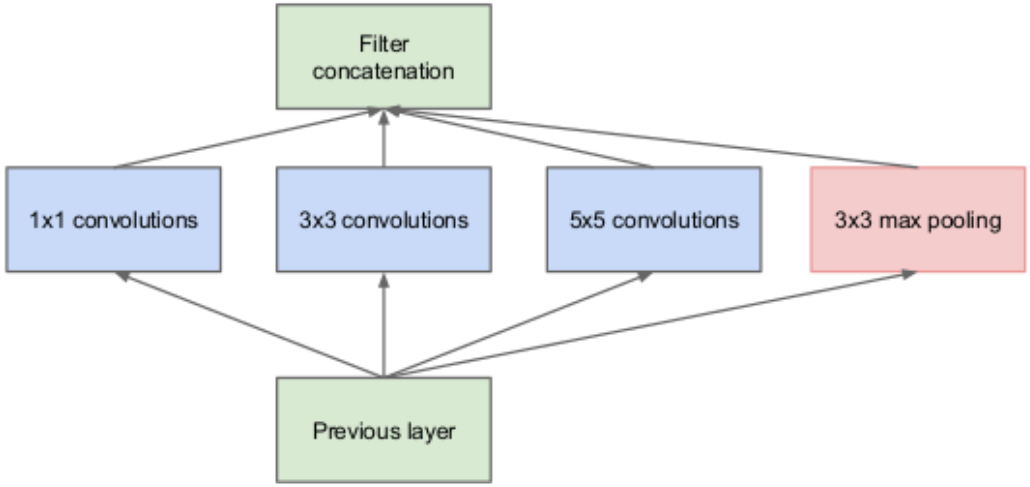


Figure: Inception module, naive version (image from [SLJ+14])

- the module uses 1×1, 3×3, 5×5 convolutions and a pooling in parallel
- needs large computational resources

Inception Module - With dimensionality reduction

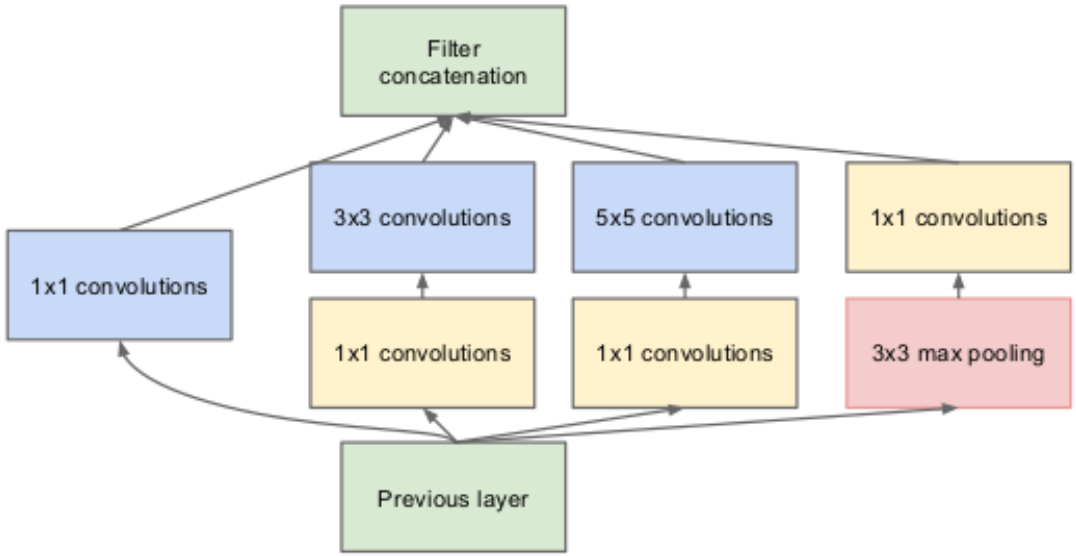


Figure: Inception module, naive version (image from [SLJ+14])

- use 1×1 convolutions for dimensionality reduction
- provide an added element of nonliniarity

GoogLeNet Architecture

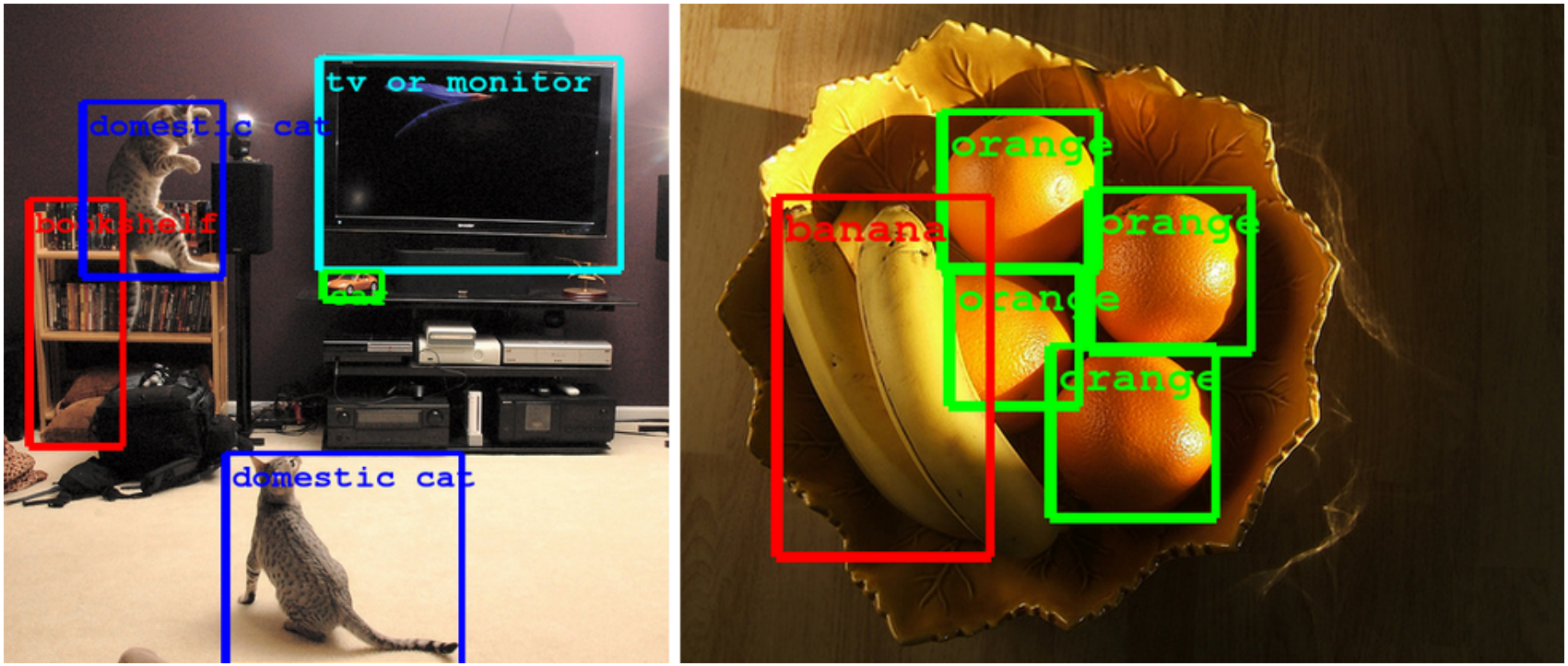


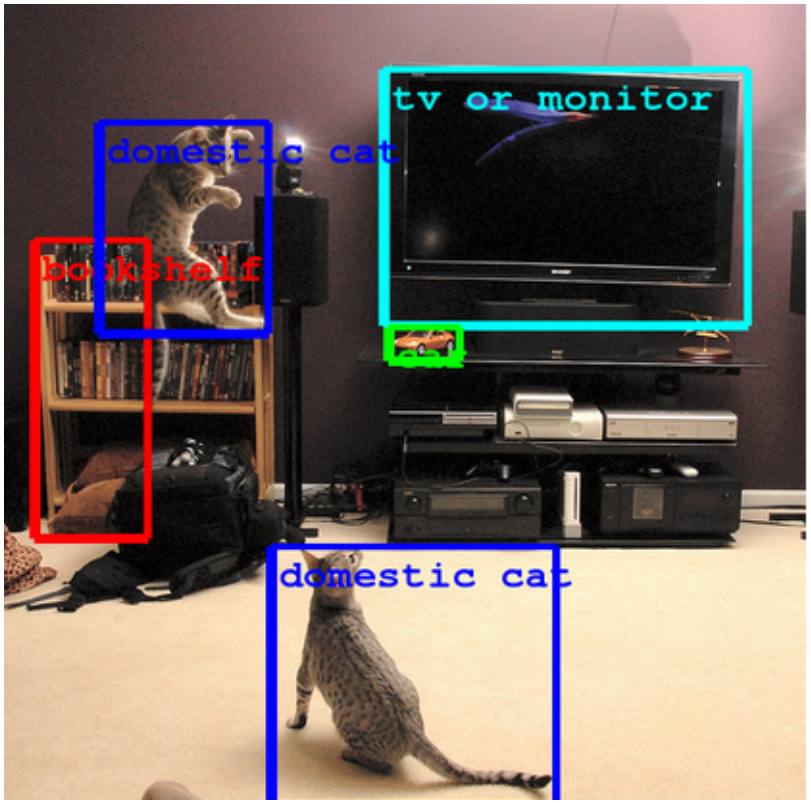
Figure: GoogLeNet Architecture (image from [SLJ+14])

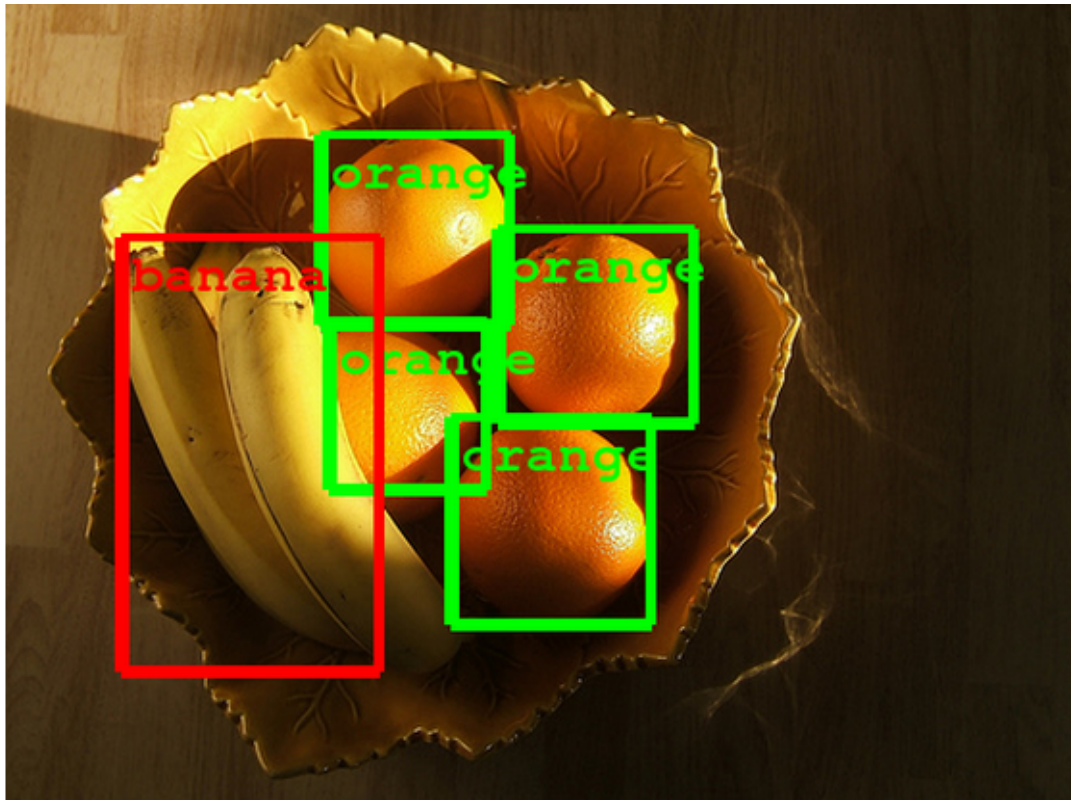
- 22 layers deep, but 12x fewer parameters than AlexNet
- no fully connected layers on top
- mid-network classification to improve backpropagated gradient

Results

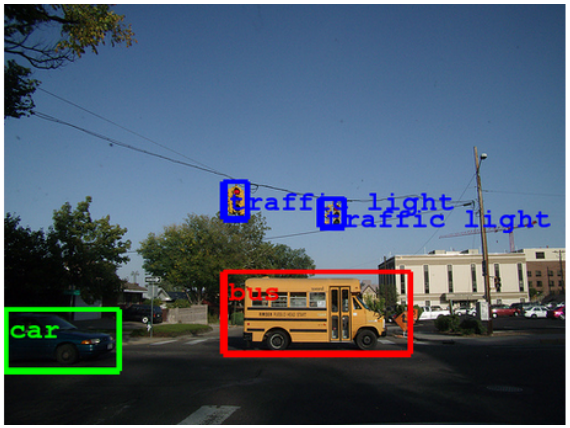
- GoogLeNet entered the ILSVRC2014 challenge
  - 1st place on Classification
  - 1st place on Object Detection with additional data

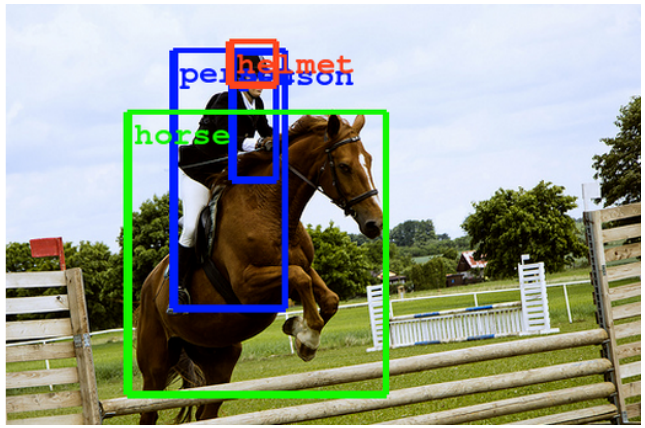
2014 - GoogLeNet results





2014 - GoogLeNet results





2014 results

ILSVRC classification task top-5 performance (with provided training data only).

Team	Year	Place	Top-5 error
SuperVision	2012	1	16.42%
ISI	2012	2	26.17%
VGG	2012	3	26.98%
Clarifai	2013	1	11.74%
NUS	2013	2	12.95%
ZF	2013	3	13.51%
GoogLeNet	2014	1	6.66%
VGG	2014	2	7.32%
MSRA	2014	3	8.06%
Andrew Howard	2014	4	8.11%
DeeperVision	2014	5	9.51%

Classification results on ImageNet 2012

Team	**Year**	**Place**	**Error**	**External data?**
SuperVision	2012		16.4%	no
SuperVision	2012	1st	15.3%	ImageNet 22k
Clarifai	2013		11.7 %	no
Clarifai	2013	1st	11.2%	no ImageNet 22k
MSRA	2014	3rd	7.35%	no
VGG	2014	2nd	7.32%	no
GoogLeNet	2014	1st	6.67%	no

Deeper and deeper

K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, CoRR **abs/1409.1556** (2014)

`http://arxiv.org/abs/1409.1556`

Deeper is better

- idea: stack many 1×1 and 3×3 convolutions one on each other
- increased discriminative ability due to extra nonliniarities (ReLU layers)
- small number of channels per layer
- fewer parameters

VGG Nets

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

VGG Results

Table 3: **ConvNet performance at a single test scale.**

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	**25.5**	**8.0**

- increased depth provide improvements to AlexNet

Ng goes to Baidu

TECH 5/19/2014 @ 1:02PM | 5,840 views

Baidu's Coup: Ng Aims To Build Silicon Valley's New Brain Trust

+ Comment Now + Follow Comments

Baidu Inc. is one of world's Internet heavyweights, sporting a $55 billion market capitalization and the status of being the planet's fifth most popular website, thanks to the popularity of its Chinese-language search engine. Until now, though, Baidu has been practically invisible in Silicon Valley.

That's about to change.

Last week, Baidu announced that it is hiring Andrew Ng, a renowned Stanford computer-science professor who also has led artificial intelligence projects at Google and has cofounded Coursera, the online-education company. Ng will take charge of a five-year, $300 million research initiative for Baidu, spanning both China and Silicon Valley. Ng will be based in Sunnyvale, Calif., where Baidu has big plans to build up its own brain trust.

Baidu builds supercomputer for deep learning

Better results on ImageNet

Chinese search engine company Baidu says it has built the world's most-accurate computer vision system, dubbed Deep Image, which runs on a supercomputer optimized for deep learning algorithms. Baidu claims a 5.98 percent error rate on the ImageNet object classification benchmark; a team from Google won the 2014 ImageNet competition with a 6.66 percent error rate.

In experiments, humans achieved an estimated error rate of 5.1 percent on the ImageNet dataset.

The computer used for image recognition

It is comprised of 36 server nodes, each with 2 six-core Intel Xeon E5-2620 processors. Each sever contains 4 Nvidia Tesla K40m GPUs and one FDR InfiniBand (56Gb/s) which is a high-performance low-latency interconnection and supports RDMA. The peak single precision floating point performance of each GPU is 4.29TFlops and each GPU has 12GB of memory.

... In total, Minwa has 6.9TB host memory, 1.7TB device memory, and about 0.6 [petaflops] theoretical single precision peak performance.

The Minwa supercomputer

- 36 nodes
  - 2 six-core Intel Xeon E5-2620
  - 4 Nvidia Tesla K40m GPUs
    - 12 GB Memory
    - 2880 cores
    - 4.29 TFLops The peak single precision floating point performance
  - 1 FDR Infiniband (56Gb/s)

Improvements

- data augmentation

Original image

Red casting

- multi-scale training ($256 \times 256 \longrightarrow 512 \times 512$)

Improvements

- data augmentation

Original image

Blue color casting

- multi-scale training ($256 \times 256 \longrightarrow 512 \times 512$)

Improvements

- data augmentation

Original image

Green color casting

- multi-scale training ($256 \times 256 \longrightarrow 512 \times 512$)

Improvements

- data augmentation

Original image

Horizontal Stretch

- multi-scale training ($256 \times 256 \longrightarrow 512 \times 512$)

Improvements

- data augmentation

Original image

Vertical stretch

- multi-scale training ($256 \times 256 \longrightarrow 512 \times 512$)

Improvements

- data augmentation

Original image

Vignette

- multi-scale training ($256 \times 256 \longrightarrow 512 \times 512$)

Improvements

- data augmentation

Original image

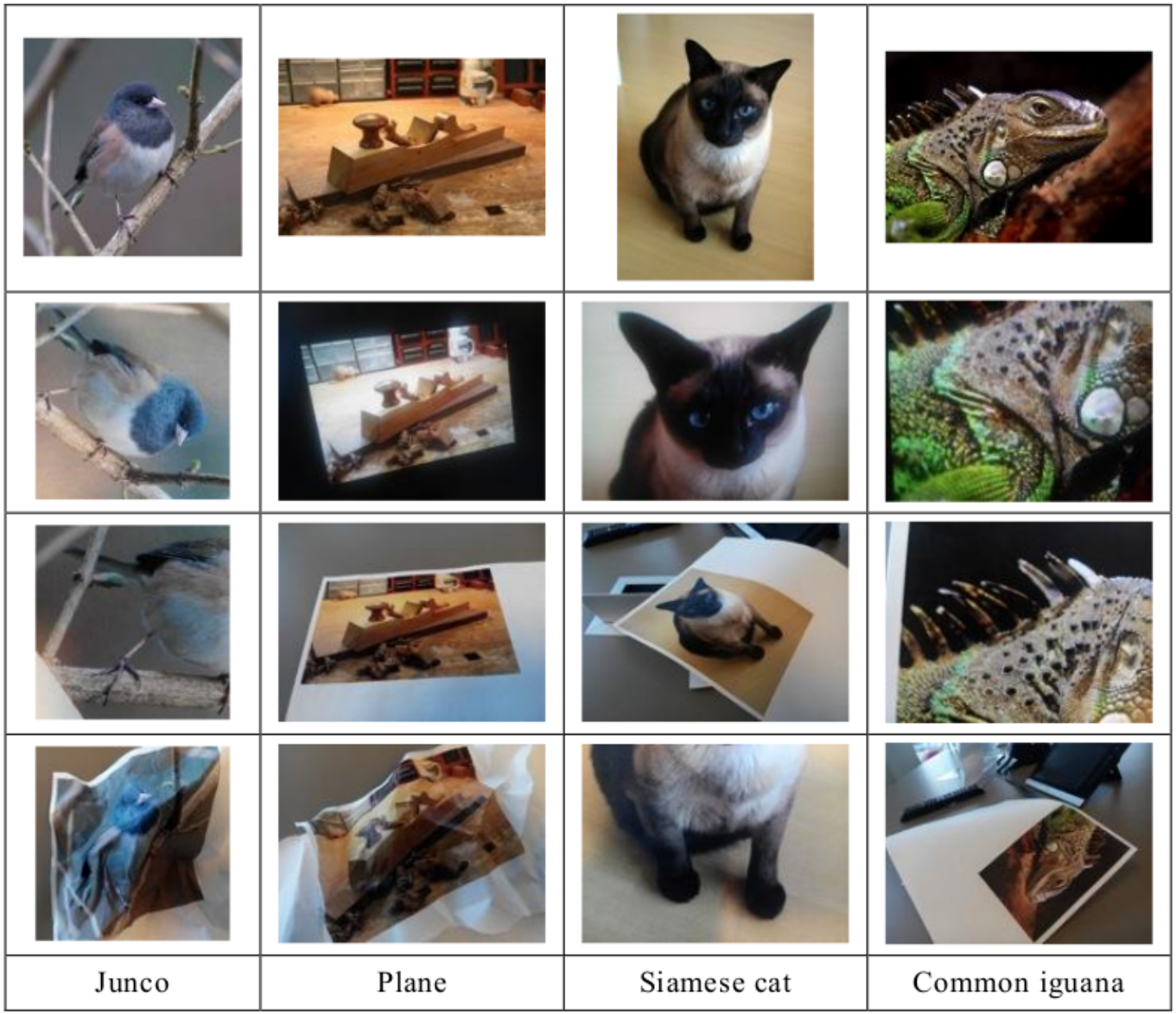
left rotation + crop

- multi-scale training ($256 \times 256 \longrightarrow 512 \times 512$)

DeepImage - the best score ever

Team	Year	Place	Top-5 error
SuperVision	2012	1	16.42%
ISI	2012	2	26.17%
VGG	2012	3	26.98%
Clarifai	2013	1	11.74%
NUS	2013	2	12.95%
ZF	2013	3	13.51%
GoogLeNet	2014	1	6.66%
VGG	2014	2	7.32%
MSRA	2014	3	8.06%
Andrew Howard	2014	4	8.11%
DeeperVision	2014	5	9.51%
Deep Image	-	-	**5.98%**

Testing robustness



Today's Outline

Spatial Transformer Networks

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus, *Intriguing properties of neural networks*, arXiv preprint arXiv:1312.6199 (2013)

`http://arxiv.org/abs/1506.02025`

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy, *Explaining and harnessing adversarial examples*, arXiv preprint arXiv:1412.6572 (2014)

`http://arxiv.org/abs/1412.6572`

Adversarial examples

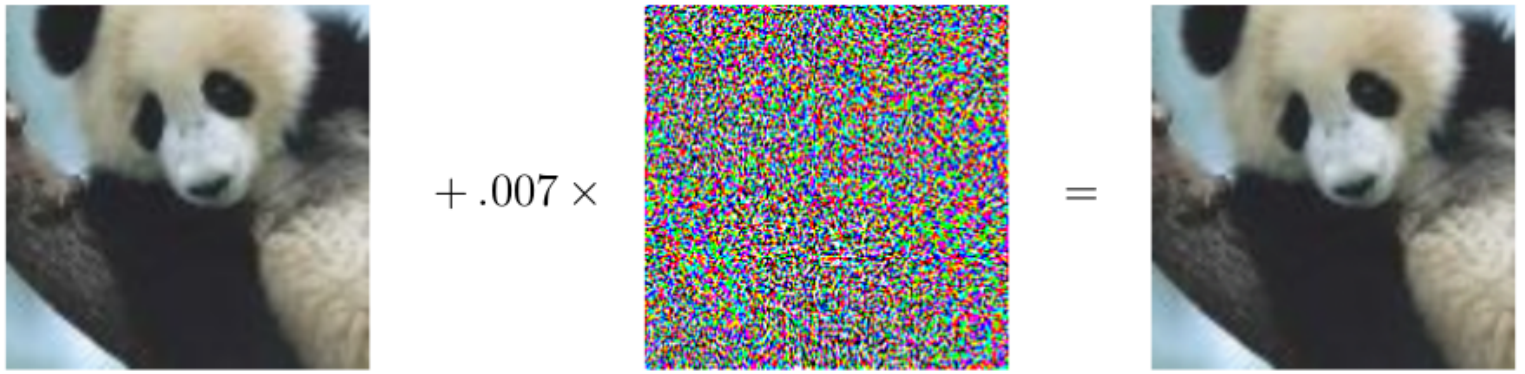


Figure: Adversarial example for GoogLeNet (image from [SZS+13])

Training against adversarial examples

- training with an adversarial objective function based on the fast gradient sign method is an **effective regularizer**

$$\tilde{J}(\theta, \mathbf{x}, y) = \alpha J(\theta, \mathbf{x}, y) + (1 - \alpha) J(\theta, \mathbf{x} + \epsilon \cdot \text{sign}(\nabla_x J(\theta, \mathbf{x}, y))$$

- using adversarial training and dropout reduced the error from 0.94% to 0.84%

Today's Outline

Spatial Transformer Networks

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu, *Spatial transformer networks*, arXiv preprint arXiv:1506.02025 (2015)

`http://arxiv.org/abs/1506.02025`

Spatial Invariance

- CNNs realize spatial invariance only through deep hierarchies of convolutions and max poolings
- STs perform (non-local) scaling, cropping or rotation
- STs can be trained with back-propagation
- STNs are useful for:
  - classification
  - co-localisation
  - spatial attention

STNs for distorted MNIST

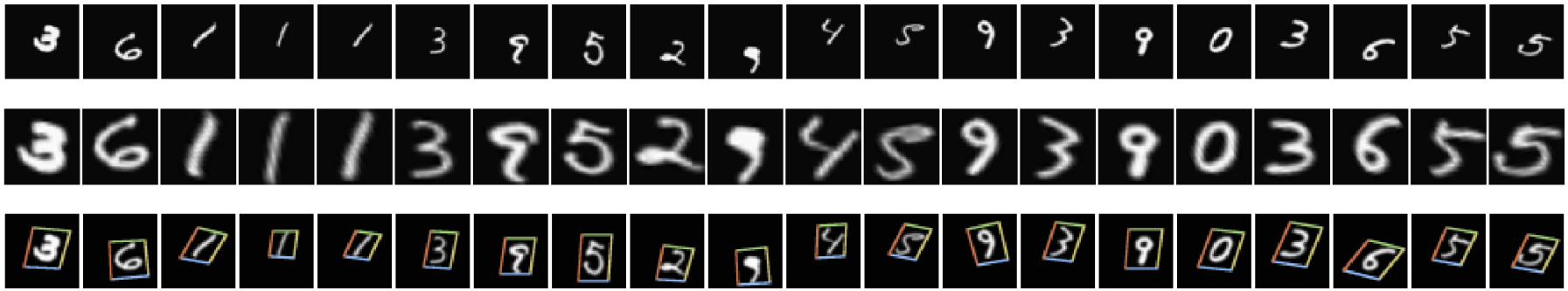


Figure: Spatial transformer as the first layer of a fully-connected network for distorted MNIST (image from [JSZK15])

The Spatial Transformer Module

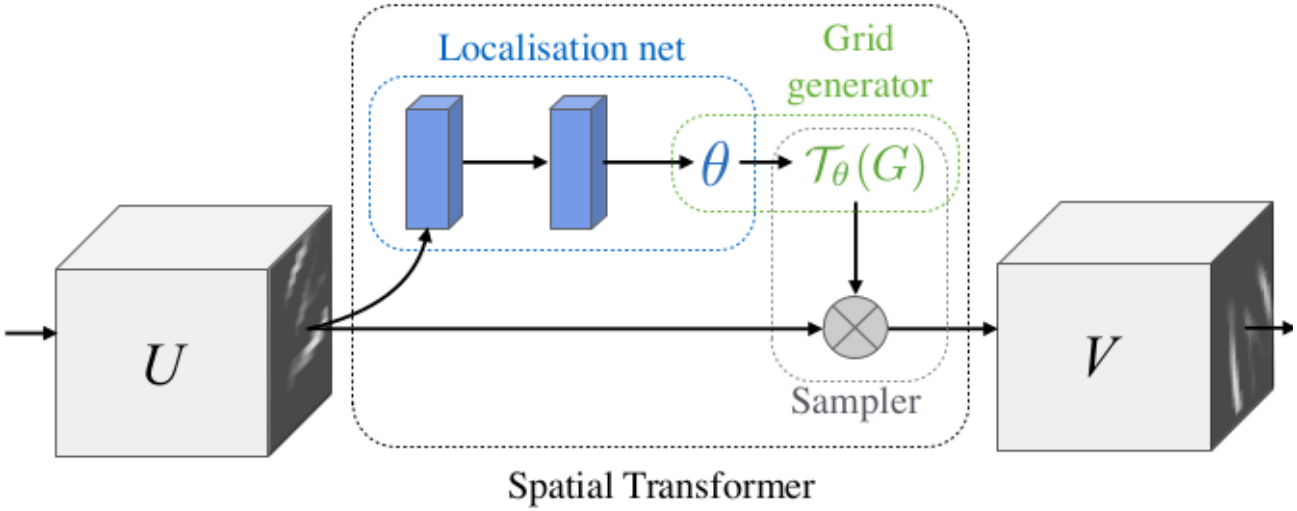


Figure: The architecture of a spatial transformer module (image from [JSZK15])

The Spatial Transformer Module

- applies a spatial transformation to a feature map during a single forward pass
- transformation is conditioned on the particular input
- for multi-channel inputs, the same transformation is applied to all channels
- architecture:
  - localisation network
  - grid generator
  - the sampler

The Localisation Network

- input: $U \in \mathbb{R}^{H \times W \times C}$
- output: $\theta = f_{\text{loc}}(U)$
  - e.g. 6 parameters matrix for affine transformations
  - any kind of network with a last regression layer

Parameterised Sampling Grid

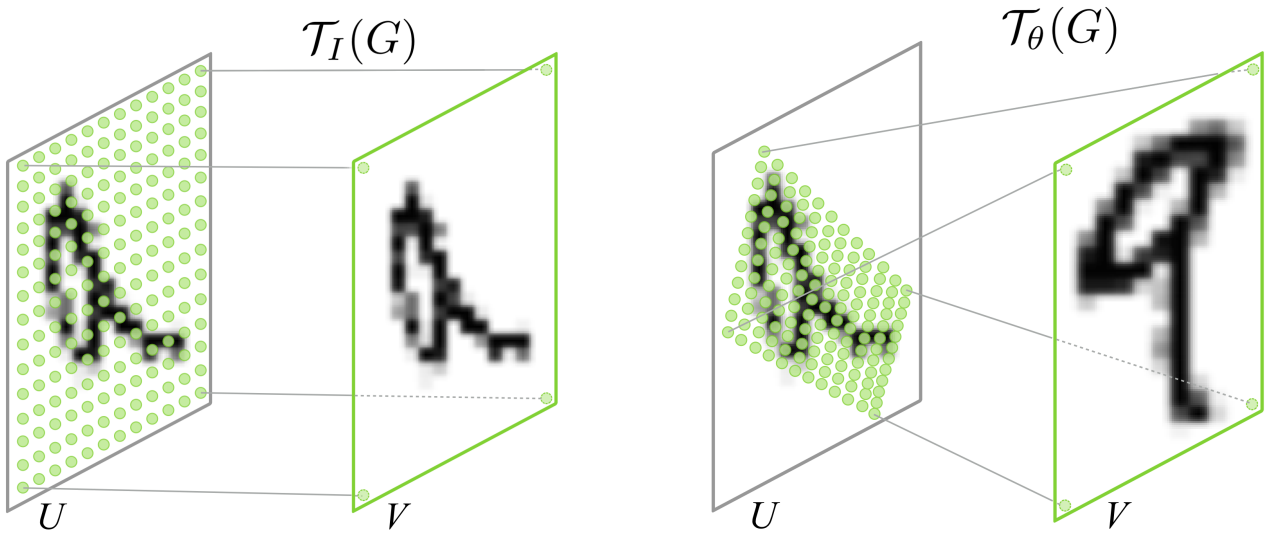


Figure: Applying the parameterised sampling grid (image from [JSZK15])

Parameterised Sampling Grid

- The output maps

$$G = \{G_i\}$$

$$G_i = \begin{pmatrix} x_i^t \\ y_i^t \end{pmatrix}$$

- a 2D affine transformation:

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

Differentiable Image Sampling

- takes the sampling points $\mathcal{T}_\theta(G)$ and U to produce V
- each point (x_i^s, y_i^s) defines the location where kernel is centered

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c k(x_i^s - m; \mathbf{\Phi}_x) k(y_i^s - n; \mathbf{\Phi}_y)$$
$$\forall i \in [1 \dots H'W'] \quad \forall c \in [1 \dots C]$$

- closest pixel

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c \delta(\lfloor x_i^s + 0.5 \rfloor - m) \delta(\lfloor y_i^s + 0.5 \rfloor - n)$$

- bilinear sampling kernel

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

Optimizing the STN for co-localisation

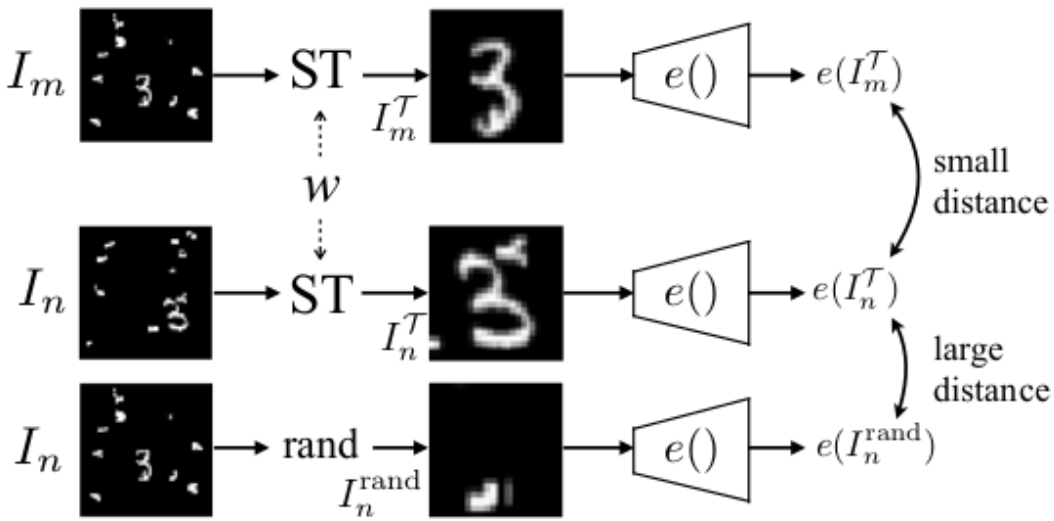


Figure: The optimisation architecture for co-localisation. (image from [JSZK15])

Today's Outline

References I


Yoshua Bengio, Ian J. Goodfellow, and Aaron Courville, *Deep learning*, Book in preparation for MIT Press, 2015.


Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, *Imagenet: A large-scale hierarchical image database*, Computer and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, 2009, pp. 248–255.


Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy, *Explaining and harnessing adversarial examples*, arXiv preprint arXiv:1412.6572 (2014).


Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu, *Spatial transformer networks*, arXiv preprint arXiv:1506.02025 (2015).

References II


Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, *Imagenet classification with deep convolutional neural networks*, Advances in neural information processing systems, 2012, pp. 1097–1105.


Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE **86** (1998), no. 11, 2278–2324.


Min Lin, Qiang Chen, and Shuicheng Yan, *Network in network*, CoRR **abs/1312.4400** (2013).


Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, *Dropout: A simple way to prevent neural networks from overfitting*, The Journal of Machine Learning Research **15** (2014), no. 1, 1929–1958.

References III


Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, *Going deeper with convolutions*, CoRR **abs/1409.4842** (2014).


K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, CoRR **abs/1409.1556** (2014).


Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus, *Intriguing properties of neural networks*, arXiv preprint arXiv:1312.6199 (2013).


Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus, *Regularization of neural networks using dropconnect*, Proceedings of the 30th International Conference on Machine Learning (ICML-13), 2013, pp. 1058–1066.

References IV


Matthew D. Zeiler and Rob Fergus, *Visualizing and understanding convolutional networks*, CoRR **abs/1311.2901** (2013).