

Introduction to Machine Learning

Deep Learning

Barnabás Póczos



MACHINE LEARNING DEPARTMENT



Credits

Many of the pictures, results, and other materials are taken from:

- ☐ Ruslan Salakhutdinov
- ☐ Joshua Bengio
- ☐ Geoffrey Hinton
- ☐ Yann LeCun

Contents

- ❑ Definition and Motivation

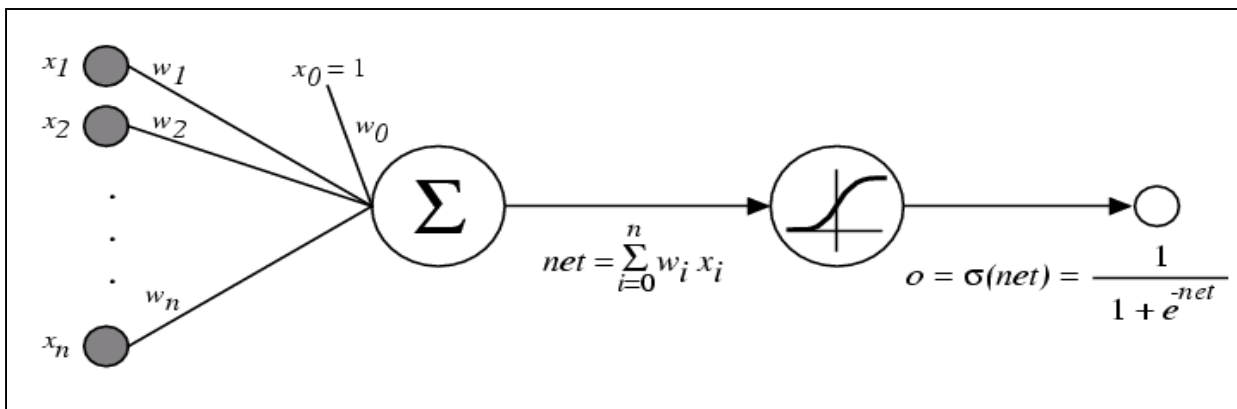
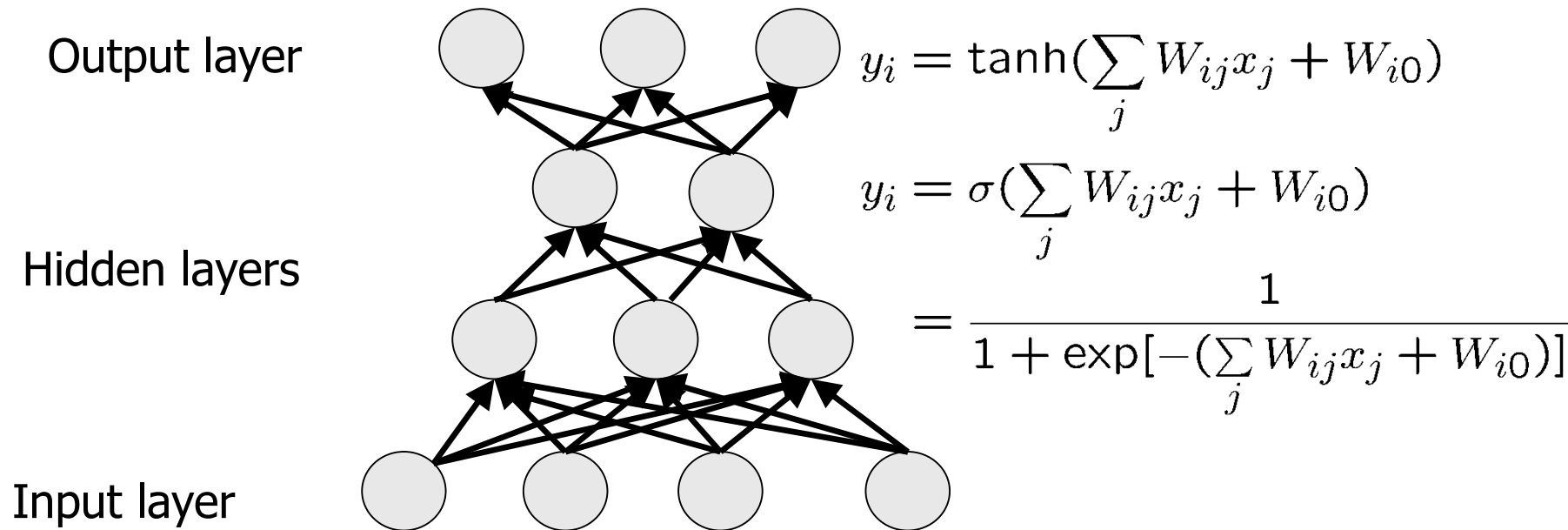
- ❑ Deep architectures

 - ❑ Convolutional networks

- ❑ Applications

Deep architectures

Defintion: Deep architectures are composed of *multiple levels* of non-linear operations, such as neural nets with many hidden layers.



Goal of Deep architectures

Goal: Deep learning methods aim at

- learning *feature hierarchies*
- where features from higher levels of the hierarchy are formed by lower level features.

edges, local shapes, object parts

Low level representation

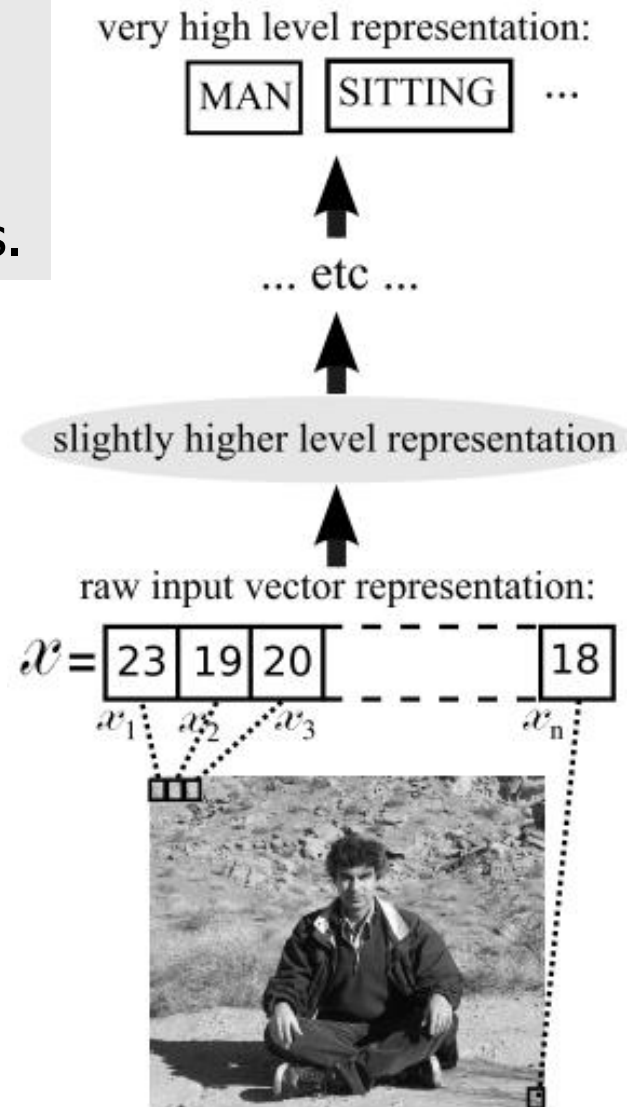


Figure is from Yoshua Bengio

Theoretical Advantages of Deep Architectures

- ❑ Some complicated functions cannot be efficiently represented (in terms of number of tunable elements) by architectures that are too shallow.
- ❑ Deep architectures might be able to represent some functions otherwise not efficiently representable.

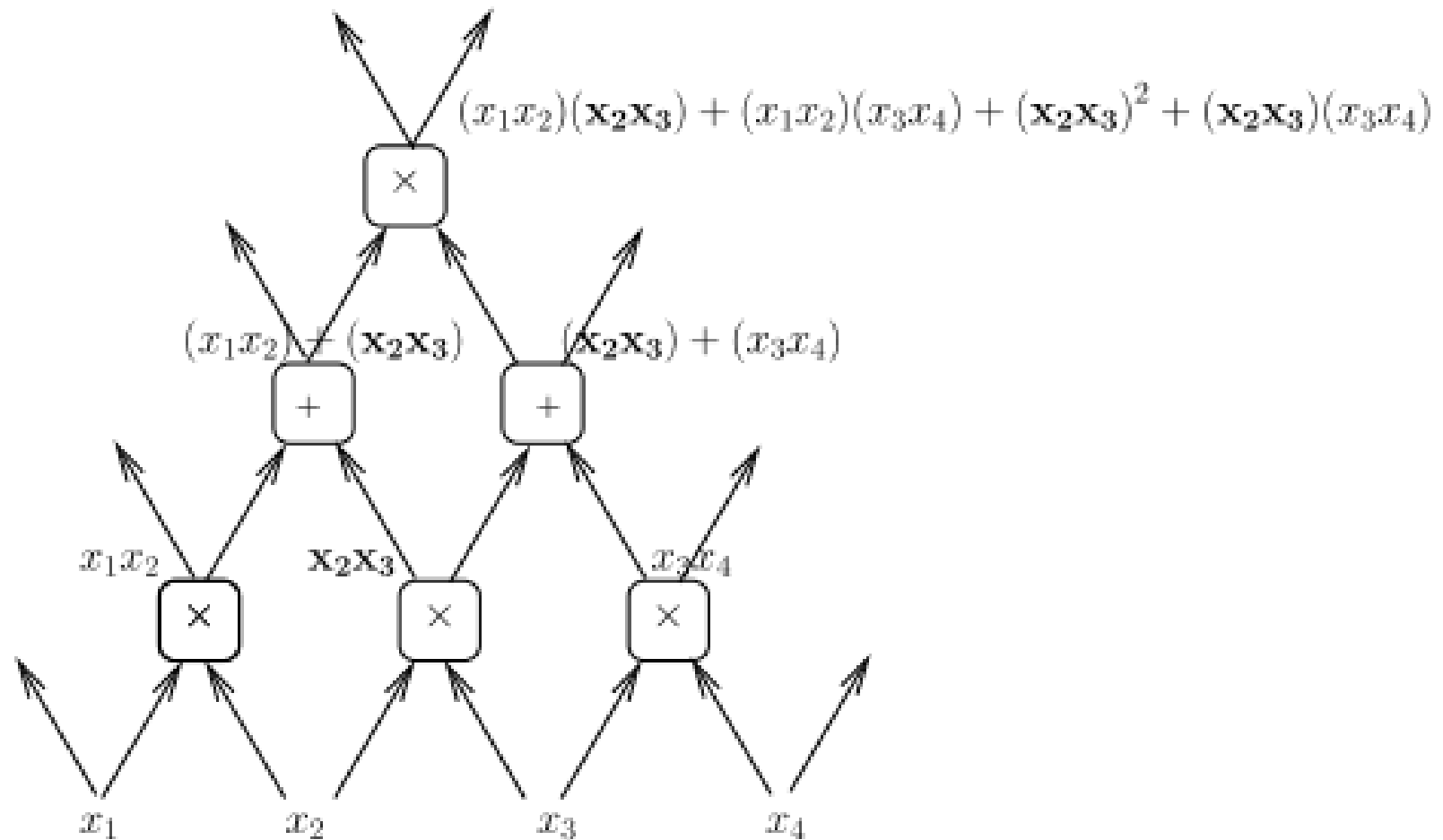
- ❑ **More formally:**

Functions that can be compactly represented by a depth k architecture might require an exponential number of computational elements to be represented by a depth $k - 1$ architecture

- ❑ The consequences are
 - **Computational:** We don't need exponentially many elements in the layers
 - **Statistical:** poor generalization may be expected when using an insufficiently deep architecture for representing some functions.

Theoretical Advantages of Deep Architectures

The Polynomial circuit:



Deep Convolutional Networks

Deep Convolutional Networks

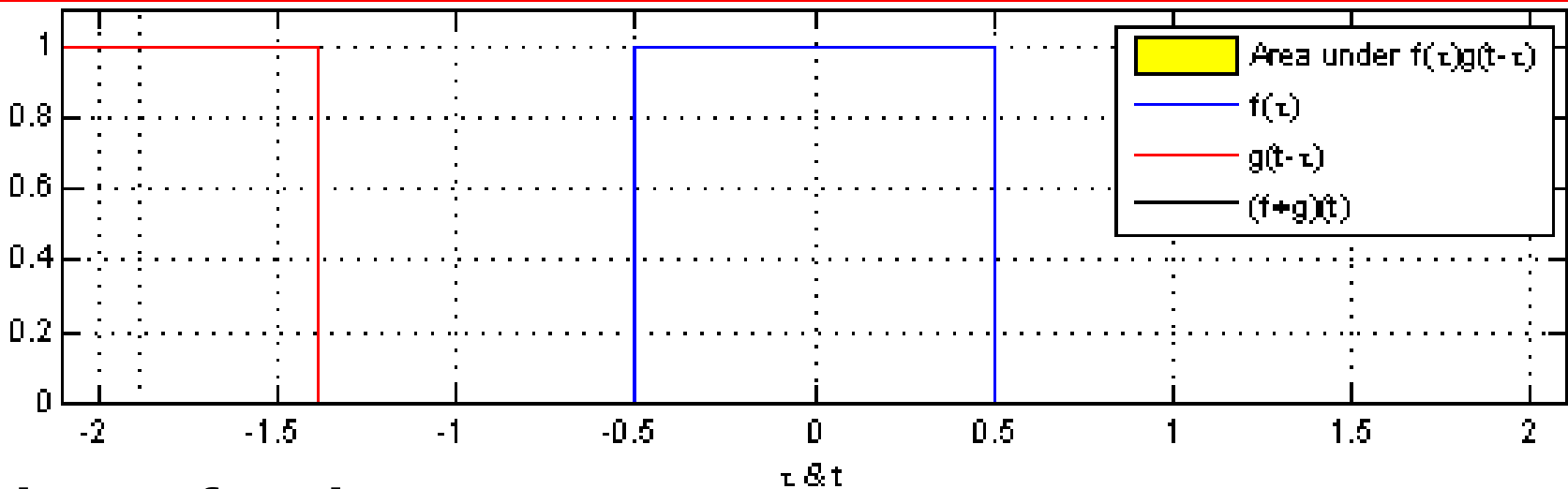
Compared to standard feedforward neural networks with similarly-sized layers,

- CNNs have much fewer connections and parameters
- and so they are easier to train,
- while their theoretically-best performance is likely to be only slightly worse.

LeNet 5

Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: **Gradient-Based Learning Applied to Document Recognition**, *Proceedings of the IEEE*, 86(11):2278-2324, November **1998**

Convolution



Continuous functions:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau = \int_{-\infty}^{\infty} f(t - \tau) g(\tau) d\tau.$$

Discrete functions:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] g[n - m] = \sum_{m=-\infty}^{\infty} f[n - m] g[m]$$


If discrete g has support on $\{-M, \dots, M\}$:

$$(f * g)[n] = \sum_{m=-M}^M f[n - m] g[m]$$

Convolution

If discrete g has support on $\{-M, \dots, M\}$:

$$(f * g)[n] = \sum_{m=-M}^M f[n-m]g[m]$$

kernel


Product of polynomials

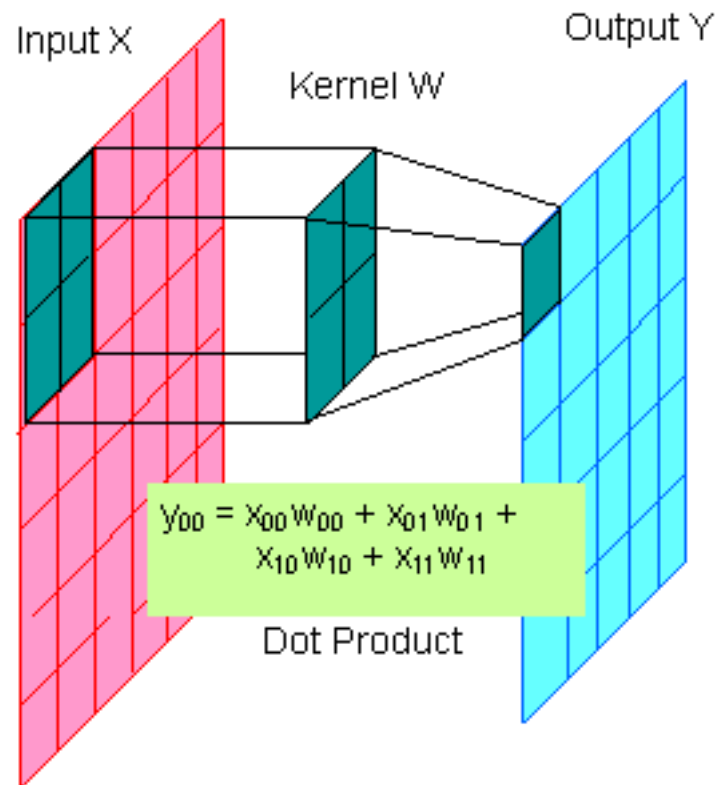
kernel of the convolution



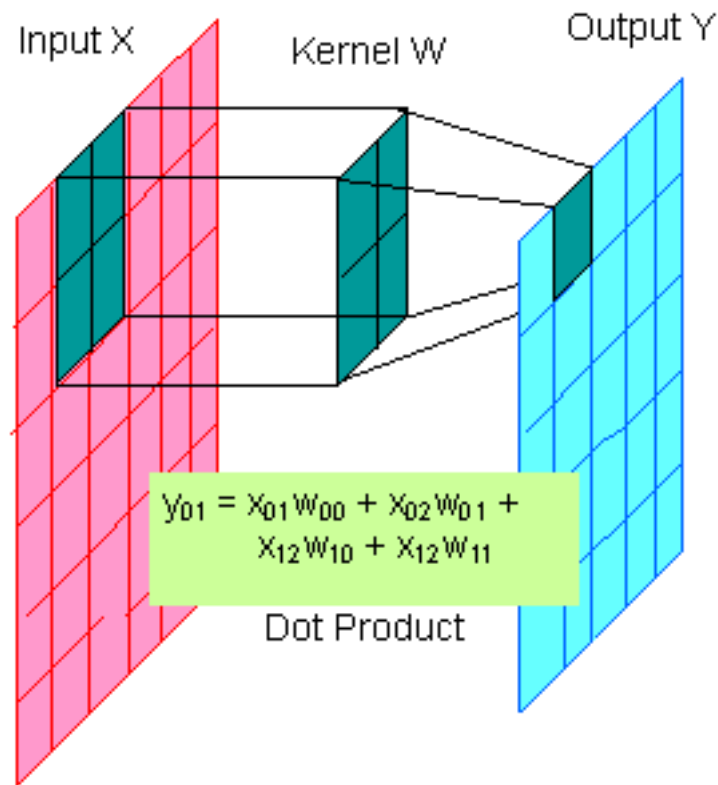
$$[1, 2] * [3, 2, 5] = (x + 2) * (3x^2 + 2x + 5) = 3x^3 + 8x^2 + 9x + 10$$

$$[1 \times 3 + 2 \times 0, 1 \times 2 + 2 \times 3, 1 \times 5 + 2 \times 2, 1 \times 0 + 2 \times 5] = [3, 8, 9, 10]$$

2-Dimensional Convolution



2-Dimensional Convolution



2-Dimensional Convolution

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1,y-n_2]$$

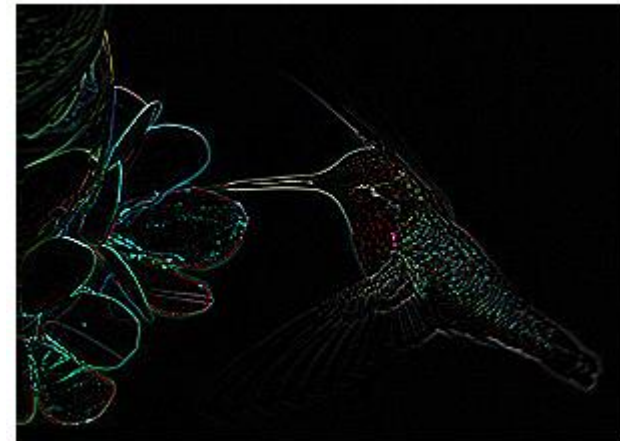
<https://graphics.stanford.edu/courses/cs178/applets/convolution.html>

Original



Filter (=kernel)

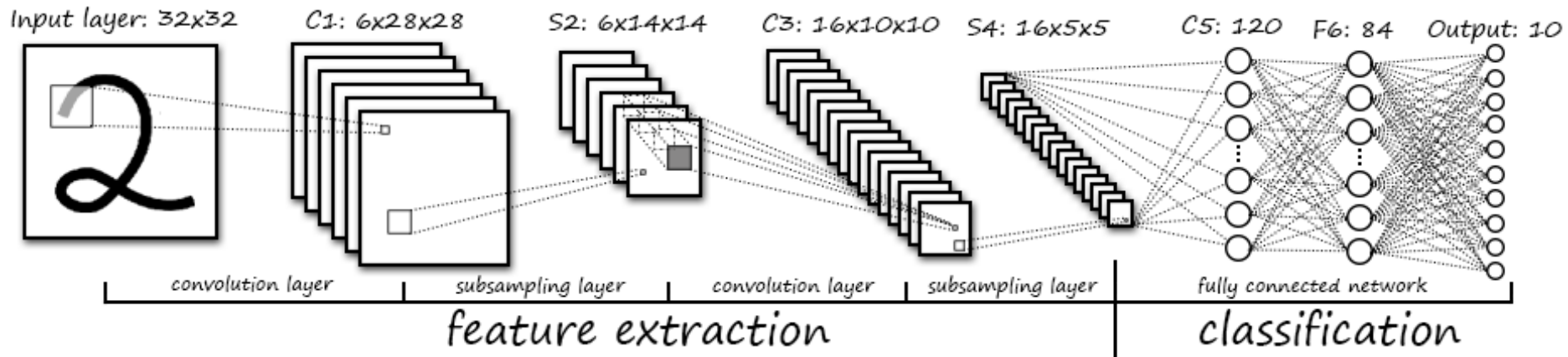
0.00	0.00	0.00	0.00	0.00
0.00	0.00	-2.00	0.00	0.00
0.00	-2.00	8.00	-2.00	0.00
0.00	0.00	-2.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00



0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04



LeNet 5, LeCun 1998

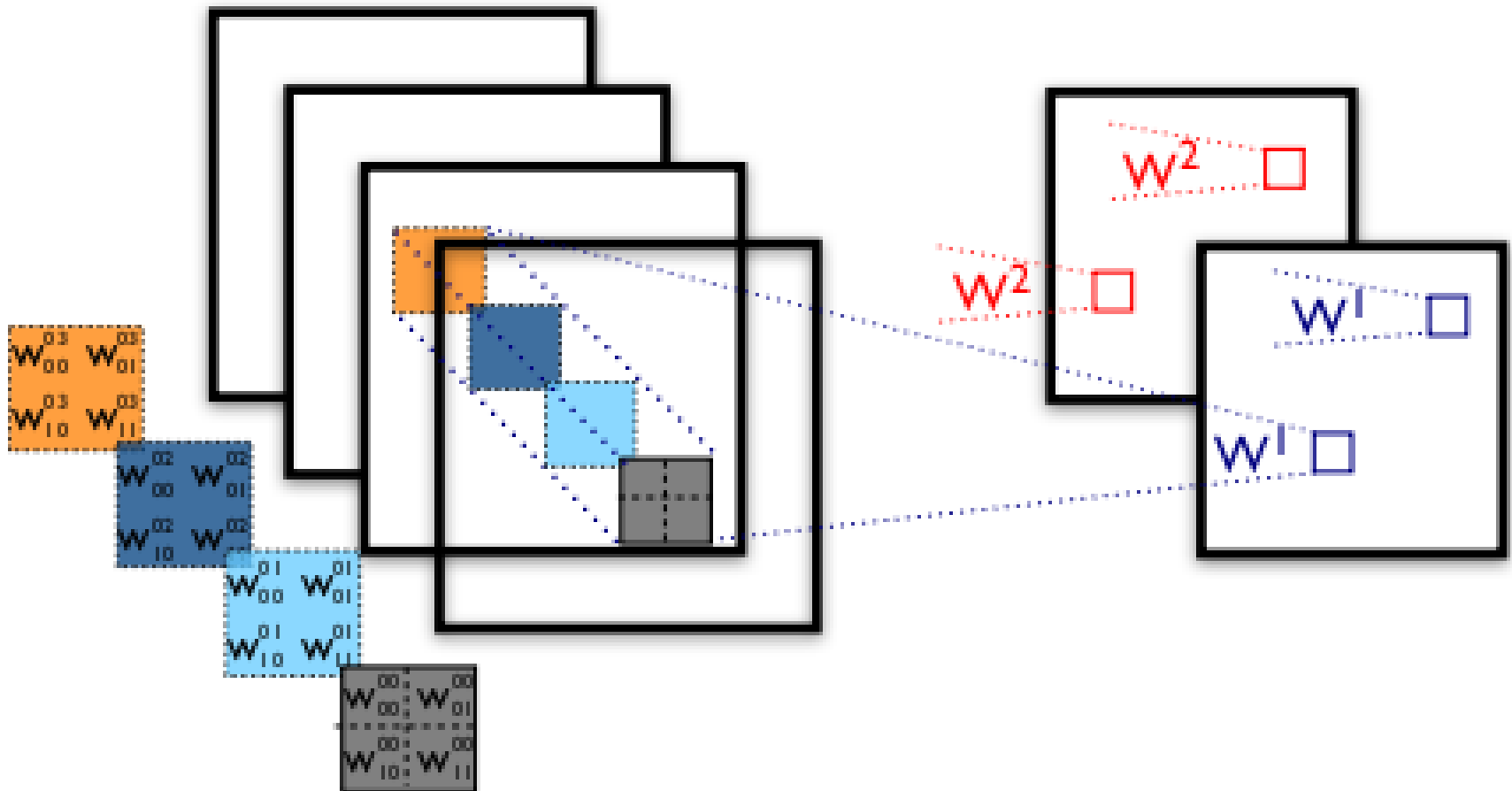


- **Input:** 32x32 pixel image. Largest character is 20x20
(All important info should be in the center of the receptive fields of the highest level feature detectors)
- **Cx:** Convolutional layer (C1, C3, C5)
- **Sx:** Subsample layer (S2, S4)
- **Fx:** Fully connected layer (F6)
- Black and White pixel values are normalized:
E.g. White = -0.1, Black = 1.175 (Mean of pixels = 0, Std of pixels = 1)

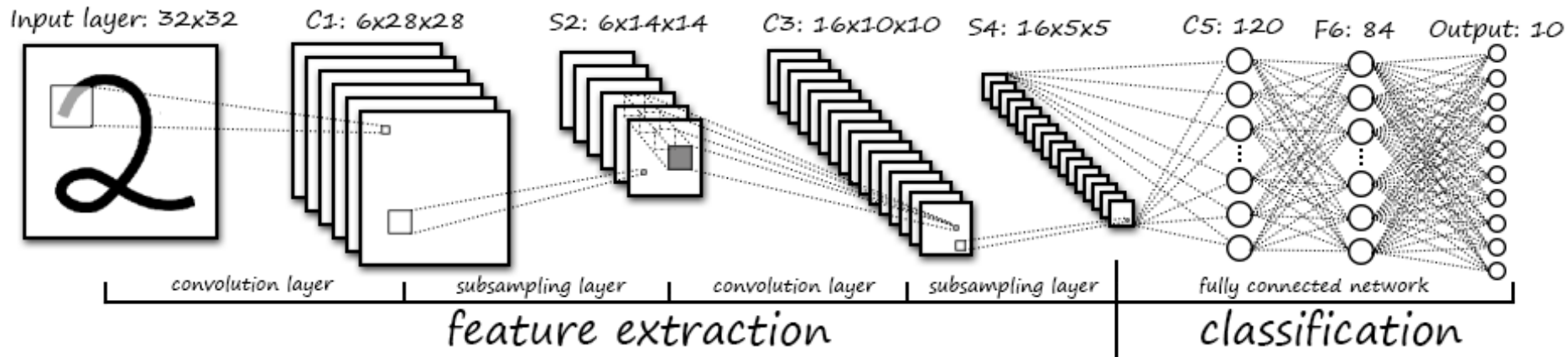
Convolutional Layer

layer $m-1$

hidden layer m



LeNet 5, Layer C1



C1: Convolutional layer with 6 feature maps of size 28x28. $C1^k, k = 1, \dots, 6$.

Each unit of C1 has a 5x5 receptive field in the input layer.

- Topological structure
- Sparse connections
- Shared weights

$$C1^k_{ij} = \tanh([W^k * x]_{ij} + b_k).$$

$$W_k \in \mathbb{R}^{5 \times 5}, b_k \in \mathbb{R}, x \in \mathbb{R}^{32 \times 32}$$

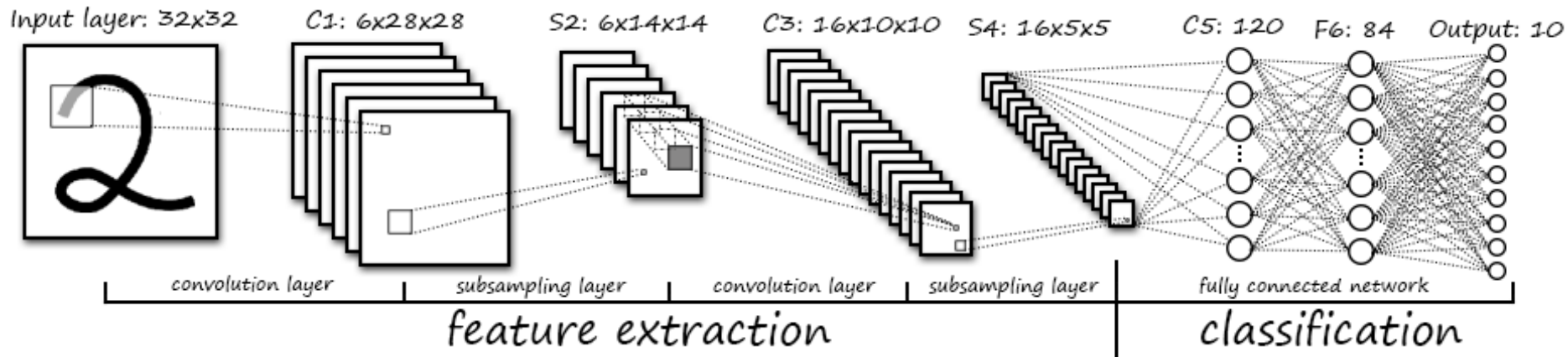
$$k=1, \dots, 6 \quad C1^k \in \mathbb{R}^{28 \times 28}.$$

$(5*5+1)*6=156$ parameters to learn

Connections: $(5*5+1)*28*28*6=122304$

If it was fully connected, we had $(32*32+1)*(28*28)*6$ parameters = connections²¹

LeNet 5, Layer S2



S2: Subsampling layer with 6 feature maps of size 14x14

2x2 nonoverlapping receptive fields in C1

$$w_1^k, w_2^k \in \mathbb{R}$$

$$S2_{ij}^k = \tanh(w_1^k \sum_{s,t=0}^1 C1_{2i-s,2j-t}^k + w_2^k).$$

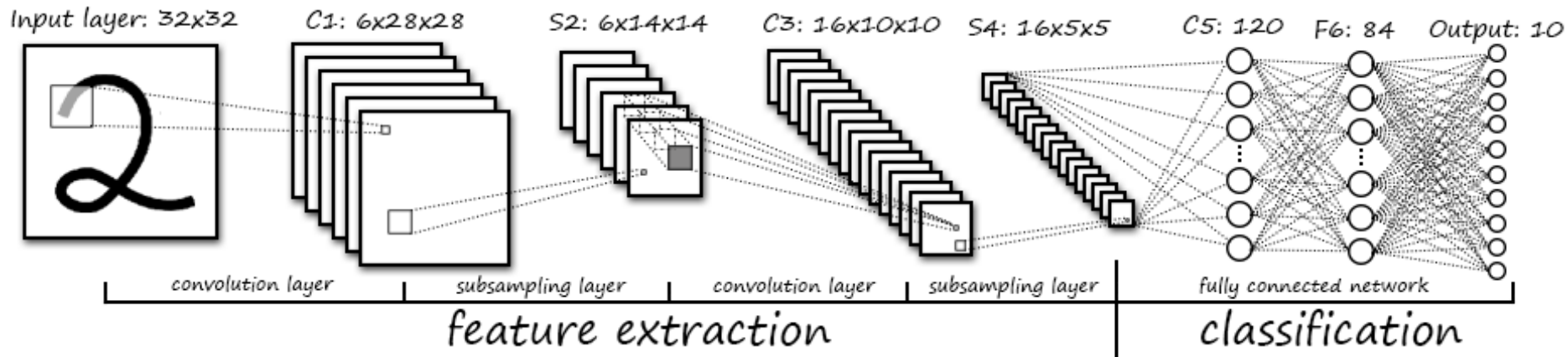
Layer S2: 6*2=12 trainable parameters.

$$k=1, \dots, 6, \quad i, j = 1, \dots, 14$$

Connections: $14 \times 14 \times (2 \times 2 + 1) \times 6 = 5880$

$$S2^k \in \mathbb{R}^{14 \times 14}.$$

LeNet 5, Layer C3



- C3: Convolutional layer with 16 feature maps of size 10x10
- Each unit in C3 is connected to several! 5x5 receptive fields at identical locations in S2

Layer C3:

1516 trainable parameters.

$$=(3*5*5+1)*6+(4*5*5+1)*9+(6*5*5+1)$$

Connections: 151600

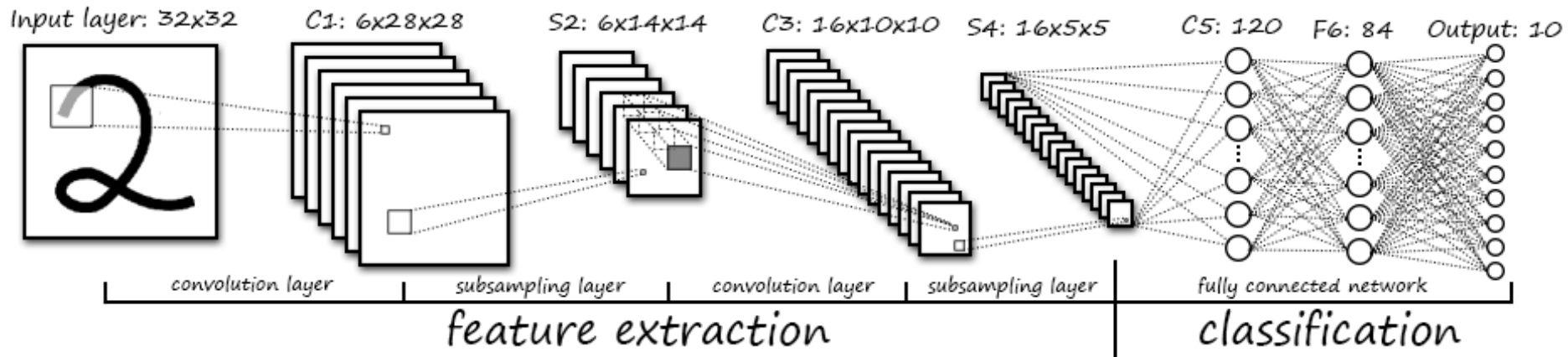
$$(3*5*5+1)*6*10*10+(4*5*5+1)*9*10*10+(6*5*5+1)*10*10$$

	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
4				X	X	X			X	X	X	X		X	X	X
5					X	X	X			X	X	X	X		X	X

TABLE I

EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.

LeNet 5, Layer S4



- S4: Subsampling layer with 16 feature maps of size 5x5
- Each unit in S4 is connected to the corresponding 2x2 receptive field at C3

$$S4_{ij}^k = \tanh(w_1^k \sum_{s,t=0}^1 C1_{2i-s,2j-t}^k + w_2^k).$$

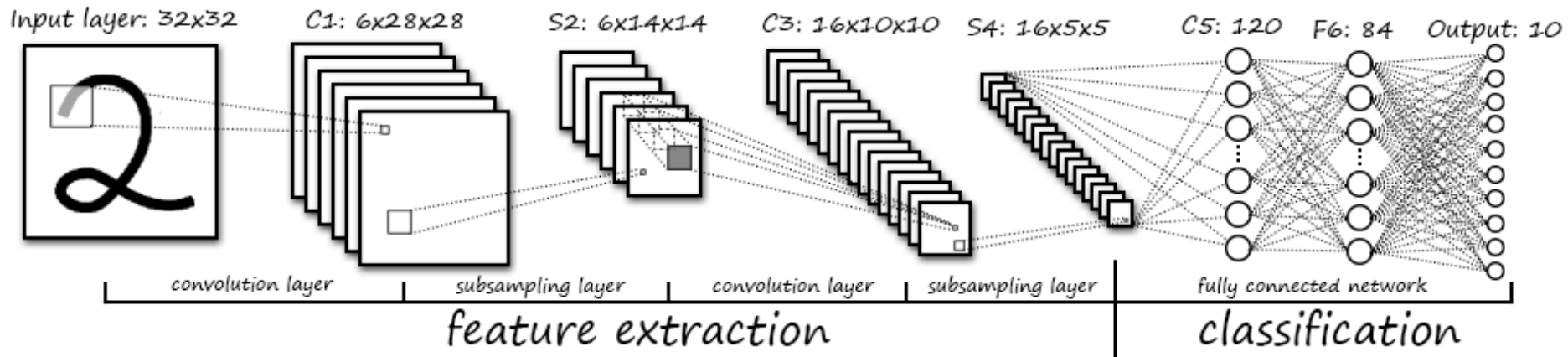
$$k=1, \dots, 16, i, j = 1, \dots, 5$$

Layer S4: 16*2=32 trainable parameters.

Connections: 5*5*(2*2+1)*16=2000

$$S4^k \in \mathbb{R}^{5 \times 5}.$$

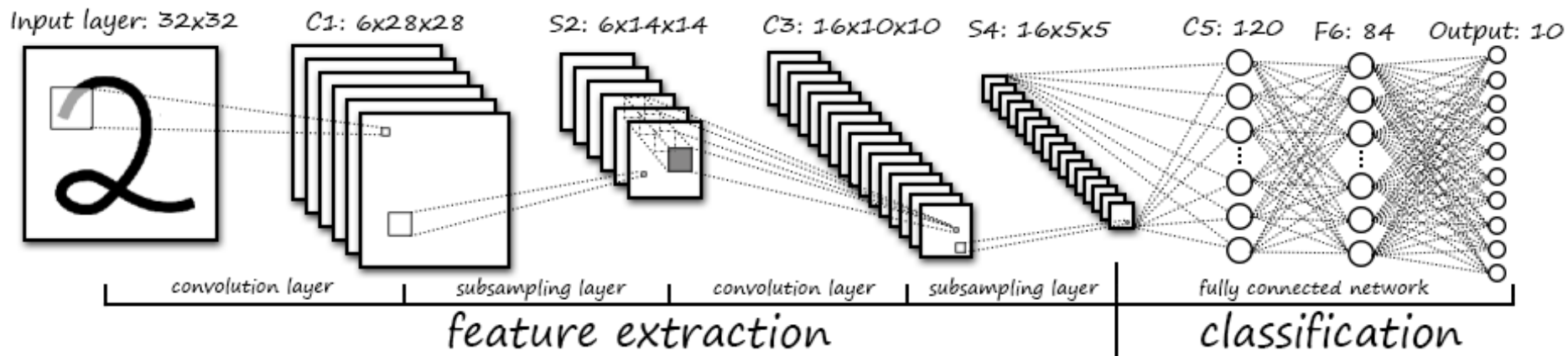
LeNet 5, Layer C5



- C5: Convolutional layer with 120 feature maps of size 1x1
- Each unit in C5 is connected to all 16 5x5 receptive fields in S4

Layer C5: $120 \times (16 \times 25 + 1) = 48120$ trainable parameters and connections
(Fully connected)

LeNet 5, Layer C5



Layer F6: 84 fully connected units. $84 \times (120 + 1) = 10164$ trainable parameters and connections.

Output layer: 10RBF (One for each digit)

$$y_i = \sum_{j=1}^{84} (x_j - w_{ij})^2, \quad i = 1, \dots, 10.$$

84=7x12, stylized image.

From F6

84 parameters, 84*10 connections

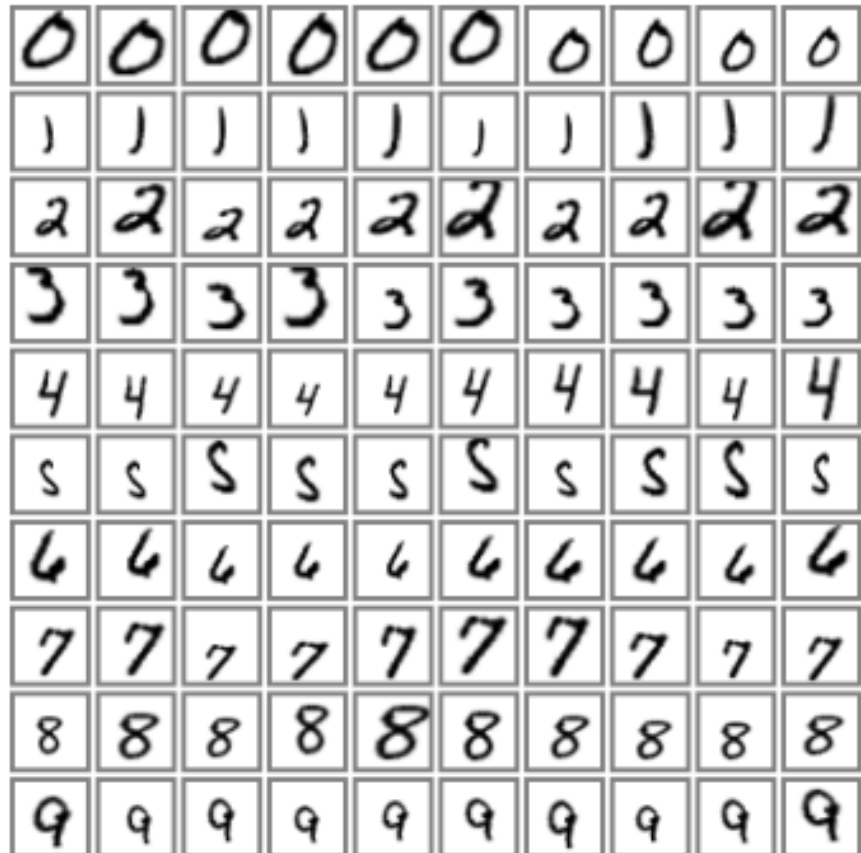
Weight update: Backpropagation

MINIST Dataset



60,000 original datasets

Test error: 0.95%



540,000 artificial distortions

+ 60,000 original

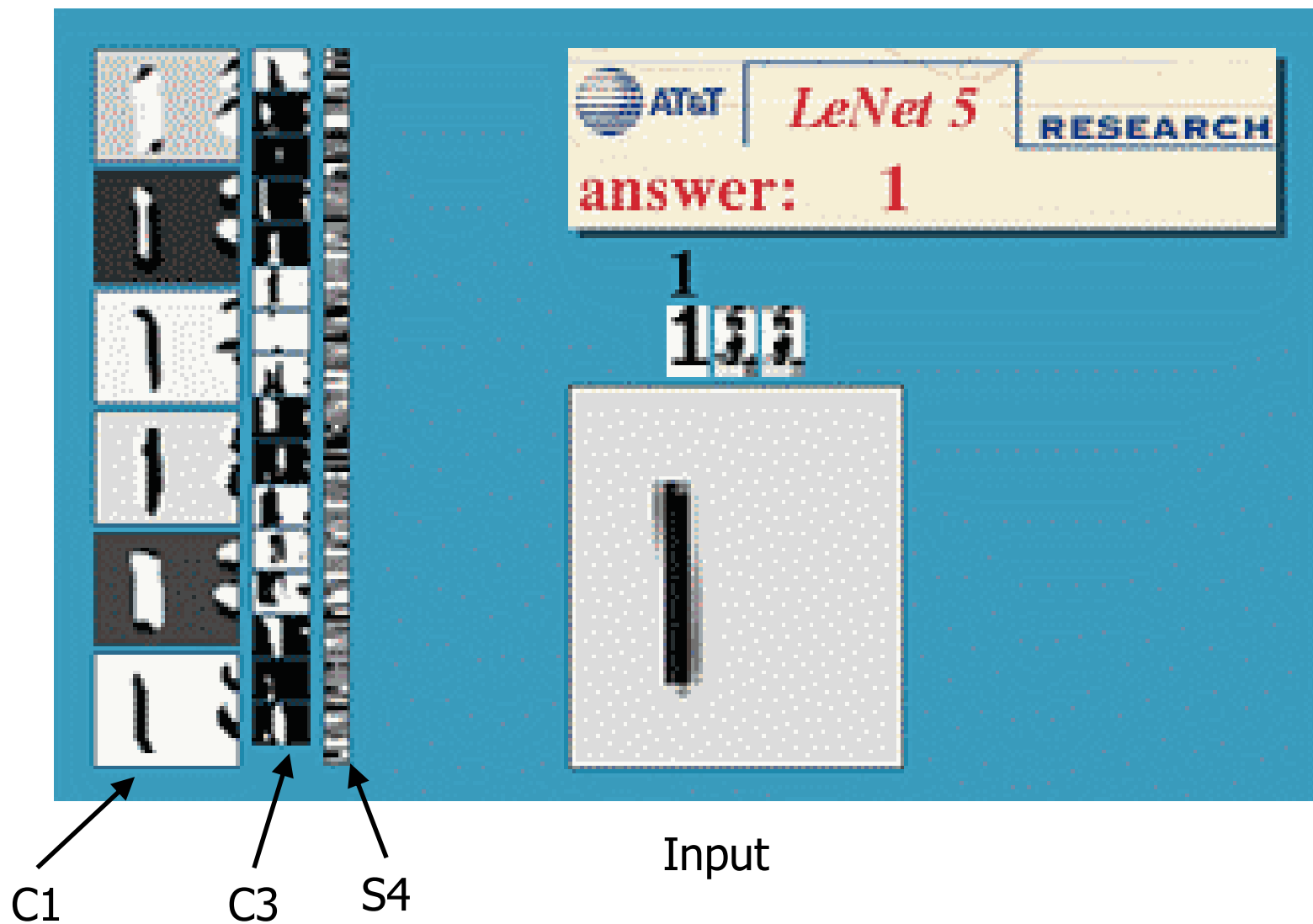
Test error: 0.8%

Misclassified examples

True label -> Predicted label

									
4->6	3->5	8->2	2->1	5->3	4->8	2->8	3->5	6->5	7->3
									
9->4	8->0	7->8	5->3	8->7	0->6	3->7	2->7	8->3	9->4
									
8->2	5->3	4->8	3->9	6->0	9->8	4->9	6->1	9->4	9->1
									
9->4	2->0	6->1	3->5	3->2	9->5	6->0	6->0	6->0	6->8
									
4->6	7->3	9->4	4->6	2->7	9->7	4->3	9->4	9->4	9->4
									
8->7	4->2	8->4	3->5	8->4	6->5	8->5	3->8	3->8	9->8
									
1->5	9->8	6->3	0->2	6->5	9->5	0->7	1->6	4->9	2->1
									
2->8	8->5	4->9	7->2	7->2	6->5	9->7	6->1	5->6	5->0
									
4->9	2->8								

LeNet 5 in Action



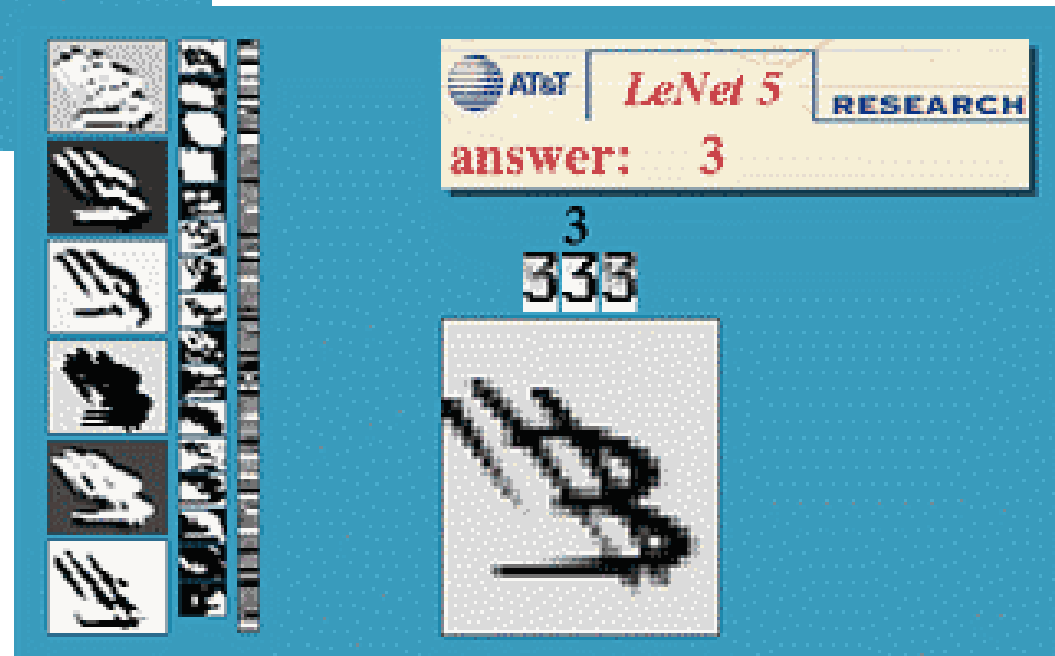
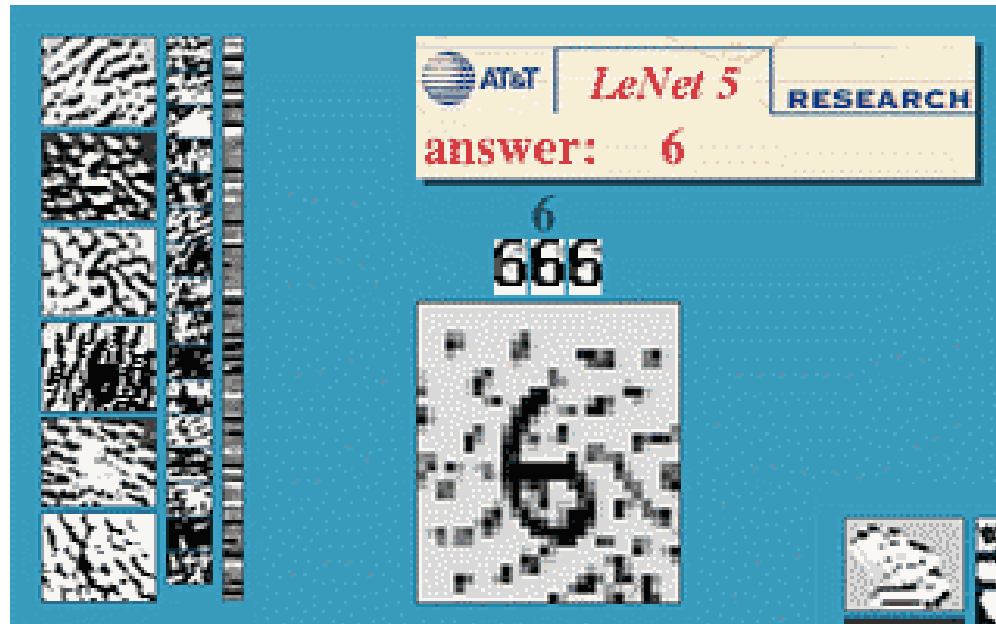
LeNet 5, Shift invariance



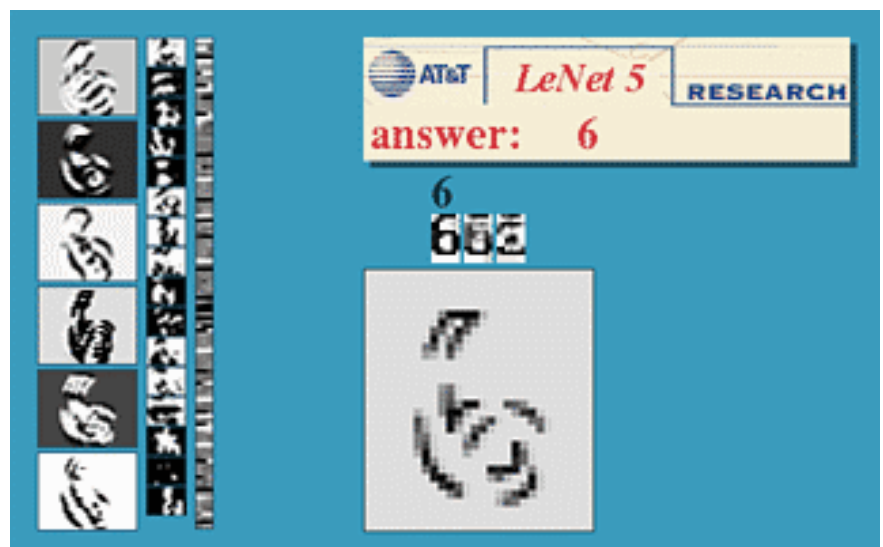
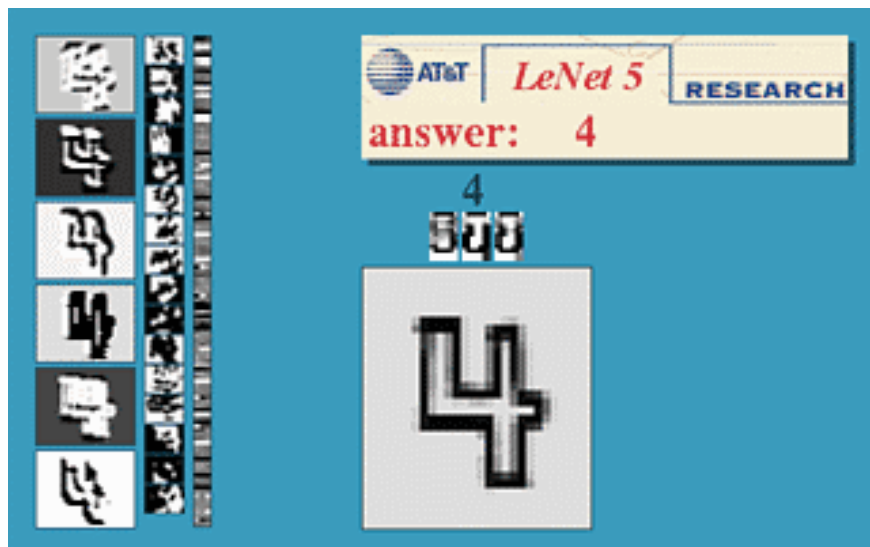
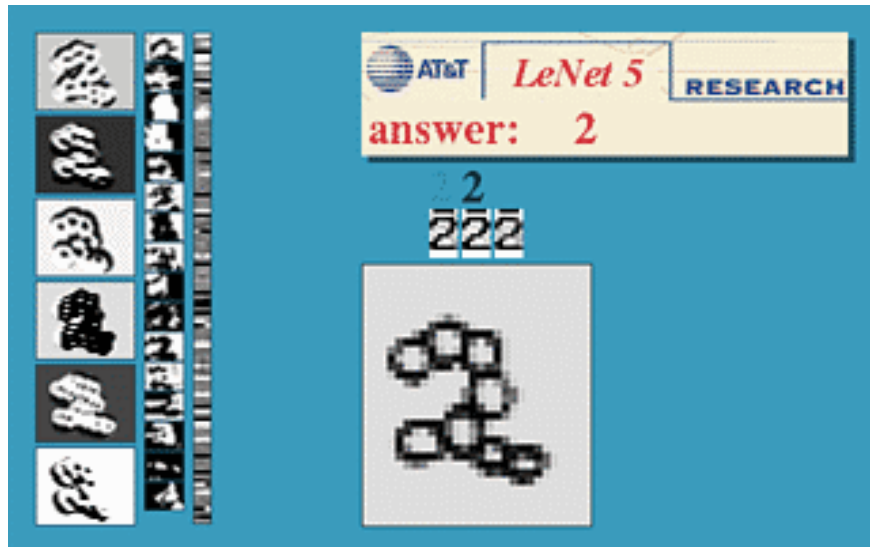
LeNet 5, Rotation invariance



LeNet 5, Noise resistance



LeNet 5, Unusual Patterns



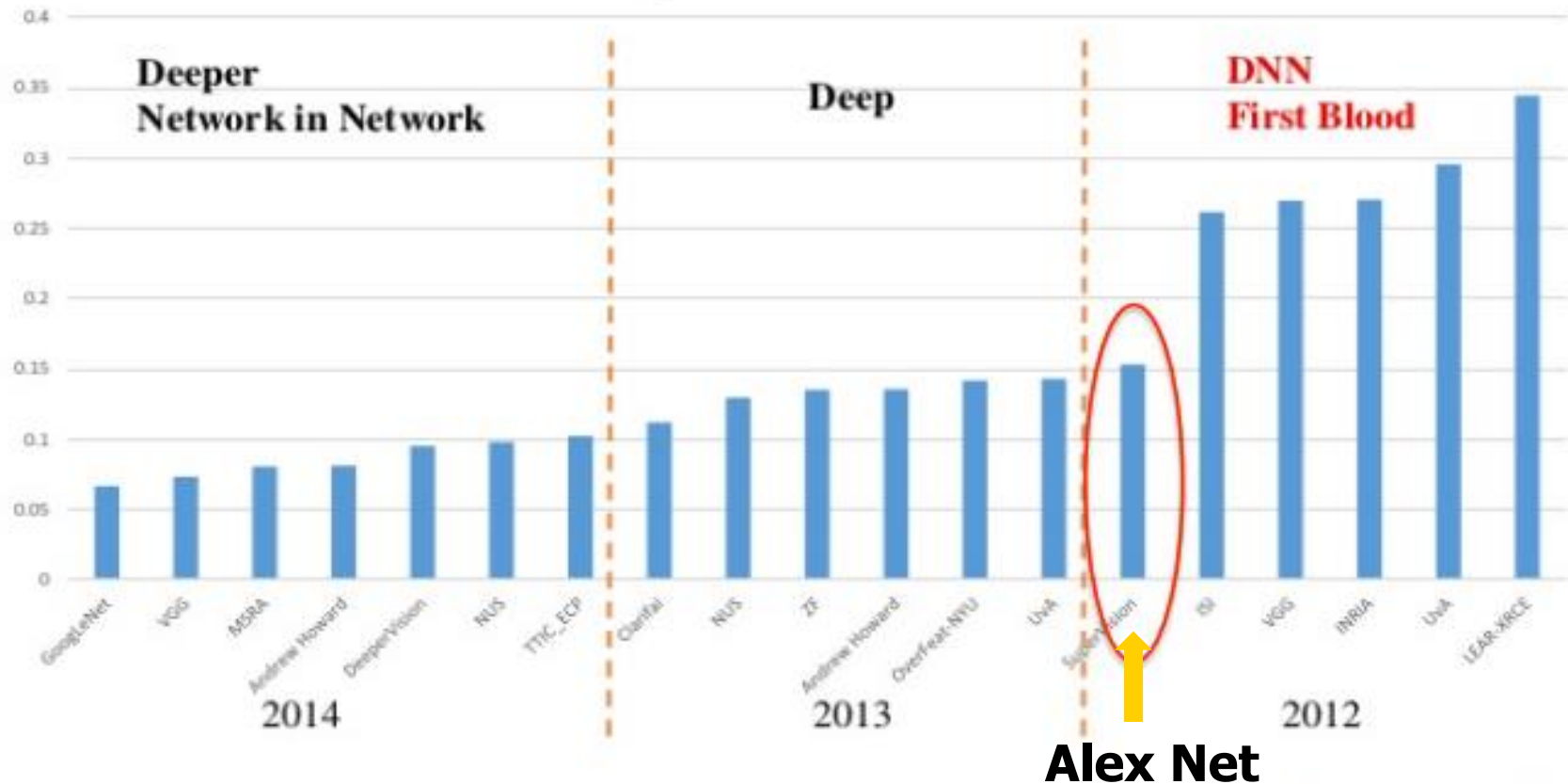
ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton,
Advances in Neural Information Processing Systems 2012

Alex Net

ILSVRC

ImageNet Classification error throughout years and groups



Li Fei-Fei: ImageNet Large Scale Visual Recognition Challenge, 2014 <http://image-net.org/>

ImageNet

- ❑ 15M images
- ❑ 22K categories
- ❑ Images collected from Web
- ❑ Human labelers (Amazon's Mechanical Turk crowd-sourcing)
- ❑ ImageNet Large Scale Visual Recognition Challenge (ILSVRC-2010)
 - 1K categories
 - 1.2M training images (~1000 per category)
 - 50,000 validation images
 - 150,000 testing images
- ❑ RGB images
- ❑ Variable-resolution, but this architecture scales them to 256x256 size

ImageNet

Classification goals:

- ❑ Make 1 guess about the label (Top-1 error)
- ❑ make 5 guesses about the label (Top-5 error)



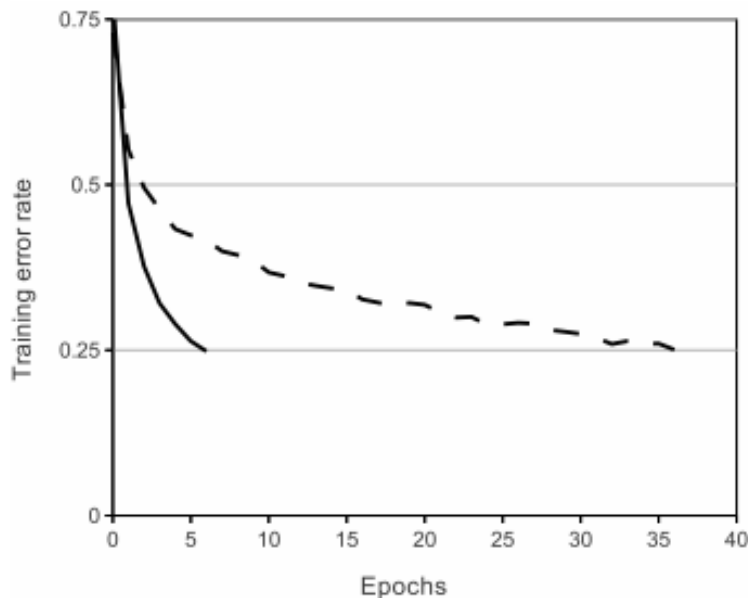
The Architecture

Typical nonlinearities: $f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

$$f(x) = (1 + e^{-x})^{-1} \quad (\text{logistic function})$$

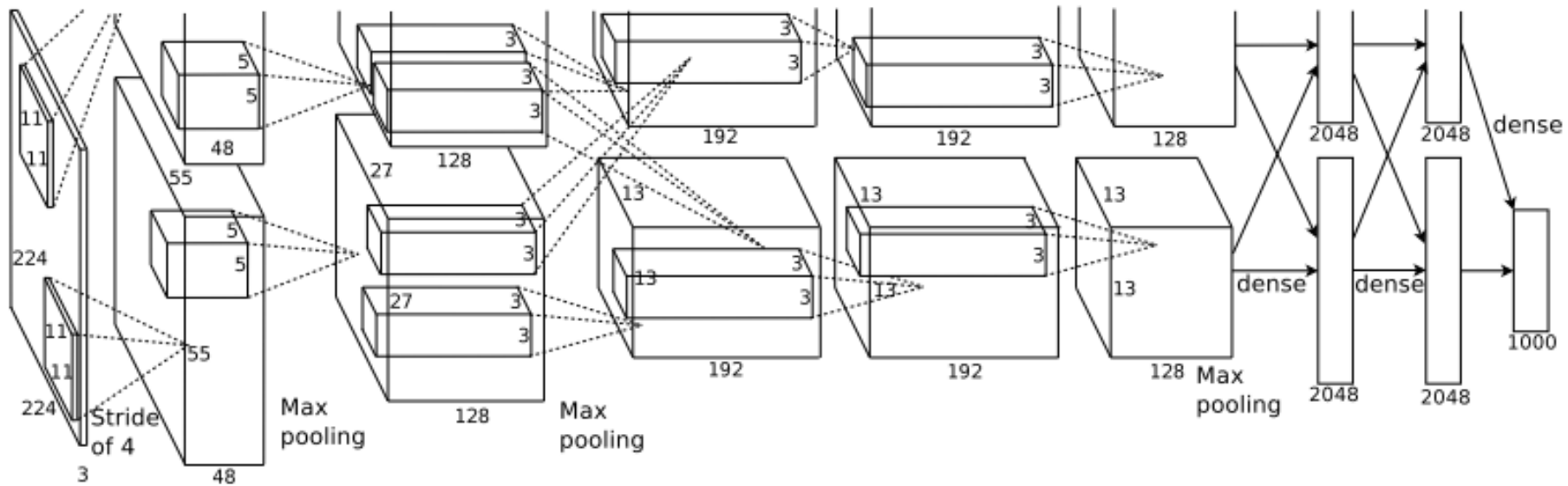
Here, however, Rectified Linear Units (ReLU) are used: $f(x) = \max(0, x)$

Empirical observation: Deep convolutional neural networks with ReLUs train several times faster than their equivalents with tanh units



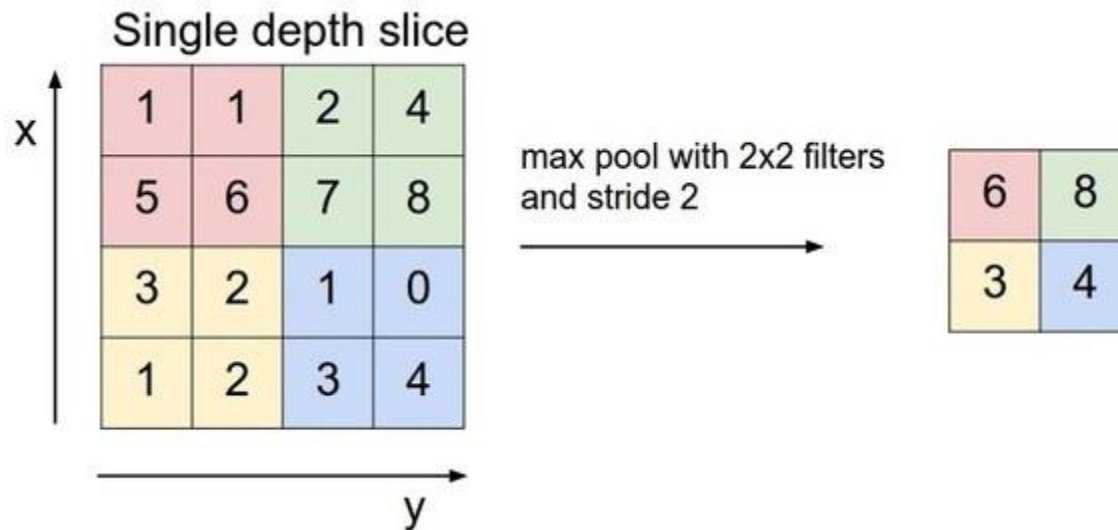
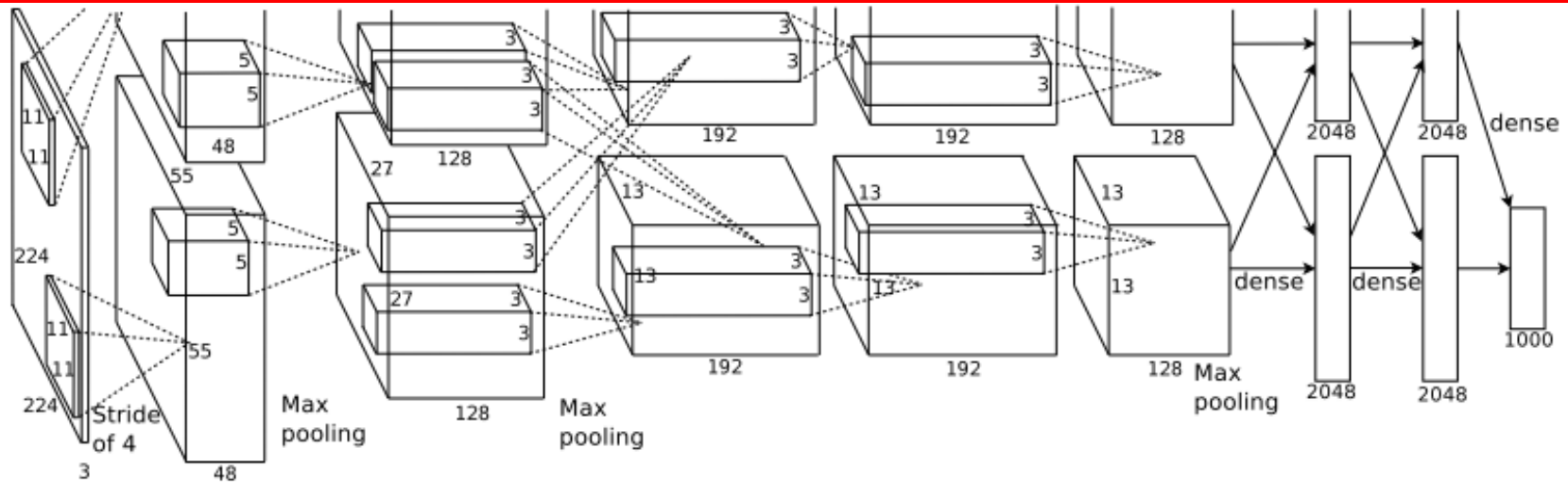
A four-layer convolutional neural network with ReLUs (solid line) reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with tanh neurons (dashed line)

The Architecture



The first convolutional layer filters the $224 \times 224 \times 3$ input image with $96 = 2 \times 48$ kernels of size $11 \times 11 \times 3$ with a stride of 4 pixels (this is the distance between the receptive field centers of neighboring neurons in the kernel map. $224/4 = 56$)

The Max-pooling Layer



The pooling layer: form of non-linear down-sampling. Max-pooling partitions the input image into a set of rectangles and, for each such sub-region, outputs the maximum value

The Architecture

- Trained with stochastic gradient descent
 - on two NVIDIA GTX 580 3GB GPUs
 - for about a week
-
- ❑ 650,000 neurons
 - ❑ 60,000,000 parameters
 - ❑ 630,000,000 connections
 - ❑ 5 convolutional layer, 3 fully connected layer
 - ❑ Final feature layer: 4096-dimensional
-
- ❑ Rectified Linear Units, overlapping pooling, dropout trick
 - ❑ Randomly extracted 224x224 patches for more data

Data Augmentation

The easiest and most common method to **reduce overfitting** on image data is to artificially **enlarge the dataset** using label-preserving transformations.

We employ two distinct forms of data augmentation:

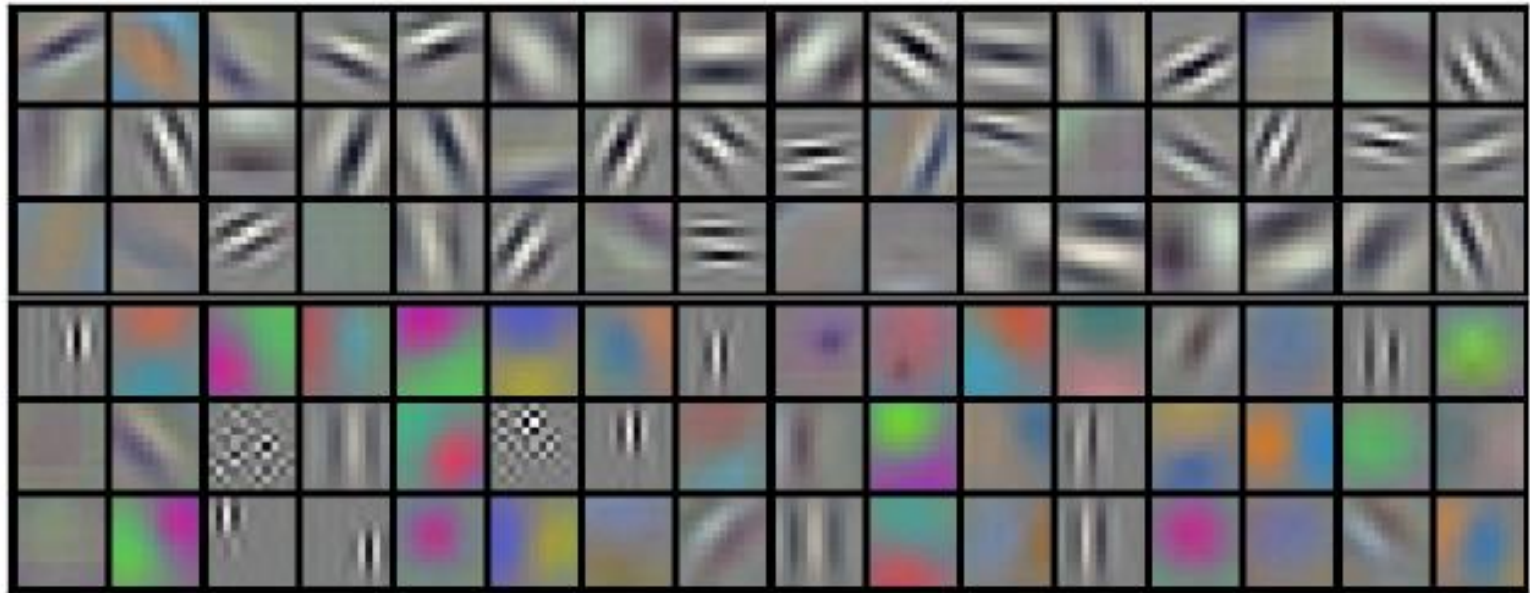
- image translation
- horizontal reflections
- changing RGB intensities

Dropout

Dropout: set the output of each hidden neuron to zero w.p. 0.5.

- The neurons which are “dropped out” in this way do not contribute to the forward pass and do not participate in backpropagation.
- So every time an input is presented, the neural network samples a different architecture, but all these architectures share weights.
- This technique reduces complex co-adaptations of neurons, since a neuron cannot rely on the presence of particular other neurons.
- It is, therefore, forced to learn more robust features that are useful in conjunction with many different random subsets of the other neurons.
- Without dropout, our network exhibits substantial overfitting.
- Dropout roughly doubles the number of iterations required to converge.

The first convolutional layer



96 convolutional kernels of size $11 \times 11 \times 3$ learned by the first convolutional layer on the $224 \times 224 \times 3$ input images.

The top 48 kernels were learned on GPU1 while the bottom 48 kernels were learned on GPU2

Looks like Gabor wavelets, ICA filters...

Results

Results on the test data:

top-1 error rate: 37.5%

top-5 error rate: 17.0%

ILSVRC-2012 competition:

15.3% classification error

2nd best team: 26.2% classification error

Results



mite

container ship

motor scooter

leopard

mite black widow cockroach tick starfish	container ship lifeboat amphibian fireboat drilling platform	motor scooter go-kart moped bumper car golfcart	leopard jaguar cheetah snow leopard Egyptian cat



grille

mushroom

cherry

Madagascar cat

convertible grille pickup beach wagon fire engine	agaric mushroom jelly fungus gill fungus dead-man's-fingers	dalmatian grape elderberry ffordshire bullterrier currant	squirrel monkey spider monkey titi indri howler monkey

Results: Image similarity



Test column

six training images that produce feature vectors in the last hidden layer with the smallest Euclidean distance from the feature vector for the test image.

Thanks for your Attention! 😊