

Deep Convolutional Networks

Aarti Singh

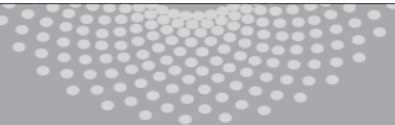
Machine Learning 10-701

Feb 15, 2023

Slides Courtesy: Barnabas Poczos, Ruslan Salakhutdinov, Joshua Bengio,
Geoffrey Hinton, Yann LeCun, Pat Virtue



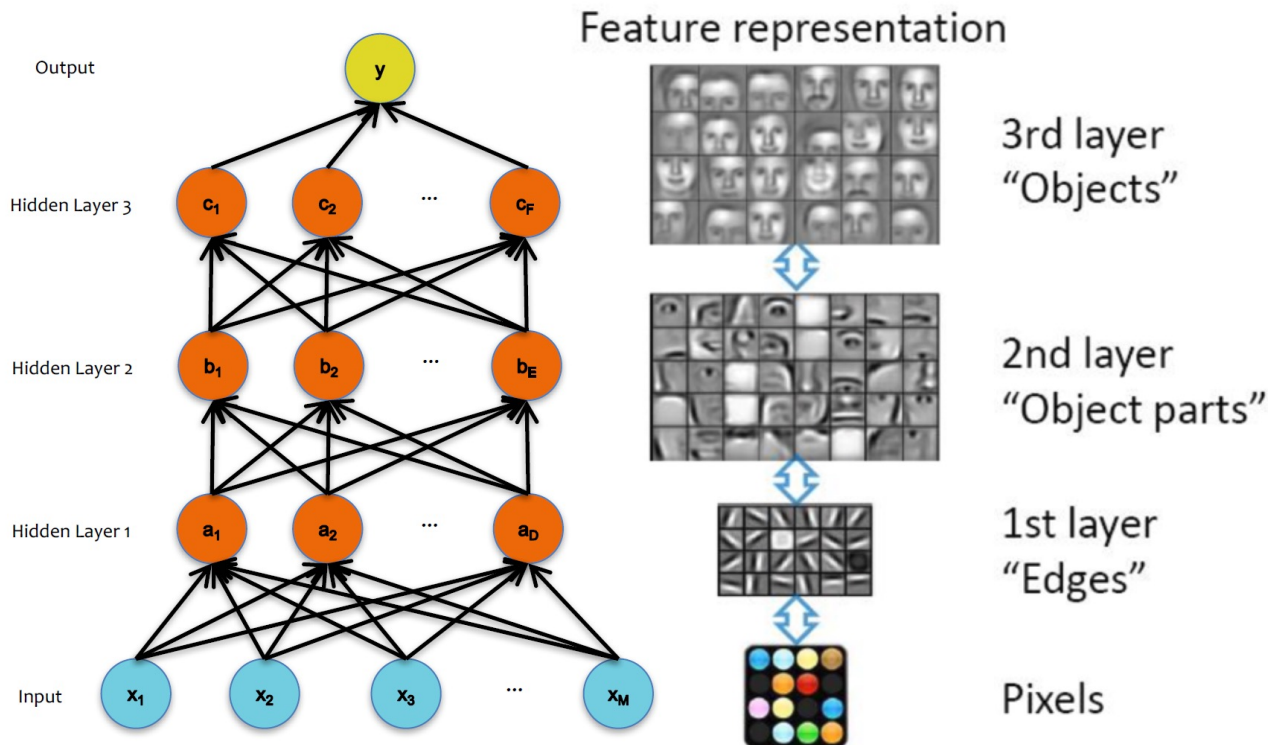
MACHINE LEARNING DEPARTMENT



Carnegie Mellon.
School of Computer Science

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.

Example from Honglak Lee (NIPS 2010)

- ❑ Neurobiological motivation: The mammal brain is organized in a deep architecture (Serre, Kreiman, Kouh, Cadieu, Knoblich, & Poggio, 2007) (E.g. visual system has 5 to 10 levels)

Deep Learning History

- ❑ **Inspired** by the architectural depth of the brain, researchers wanted for decades to train deep multi-layer neural networks.
- ❑ **No** very **successful** attempts were reported before 2006 ...
 - Researchers reported positive experimental results with typically two or three levels (i.e. one or two hidden layers), but training deeper networks consistently yielded poorer results.
- ❑ **SVM**: Vapnik and his co-workers developed the Support Vector Machine (1993). It is a shallow architecture.
- ❑ **Digression**: In the 1990's, many researchers abandoned neural networks with multiple adaptive hidden layers because SVMs worked better, and there was no successful attempts to train deep networks.
- ❑ **GPUs + Large datasets -> Breakthrough in 2006**

Breakthrough

Deep Belief Networks (DBN)

Hinton, G. E, Osindero, S., and Teh, Y. W. (2006).
A fast learning algorithm for deep belief nets.
Neural Computation, 18:1527-1554.

Autoencoders

Bengio, Y., Lamblin, P., Popovici, P., Larochelle, H. (2007).
Greedy Layer-Wise Training of Deep Networks,
Advances in Neural Information Processing Systems 19

Convolutional neural networks running on GPUs (2012)

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

Deep Convolutional Networks

Convolutional Neural Networks

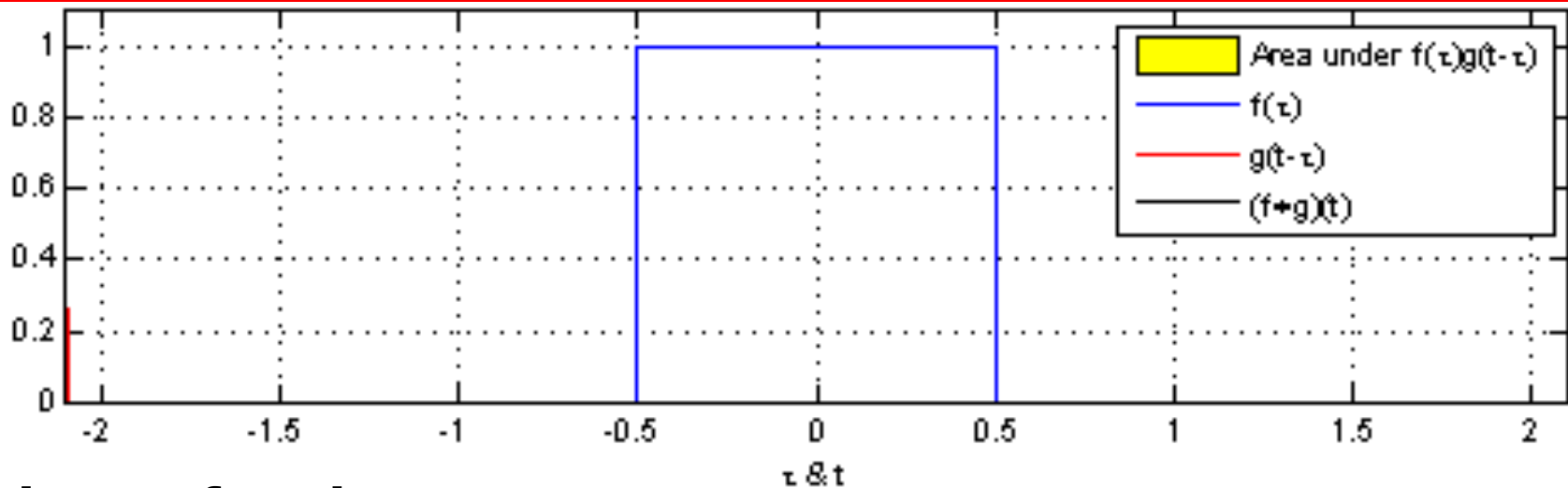
Compared to standard feedforward neural networks with similarly-sized layers,

- CNNs have much fewer connections and parameters
- and so they are easier to train,
- while their performance is likely to be only slightly worse, particularly for images as inputs.

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:

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

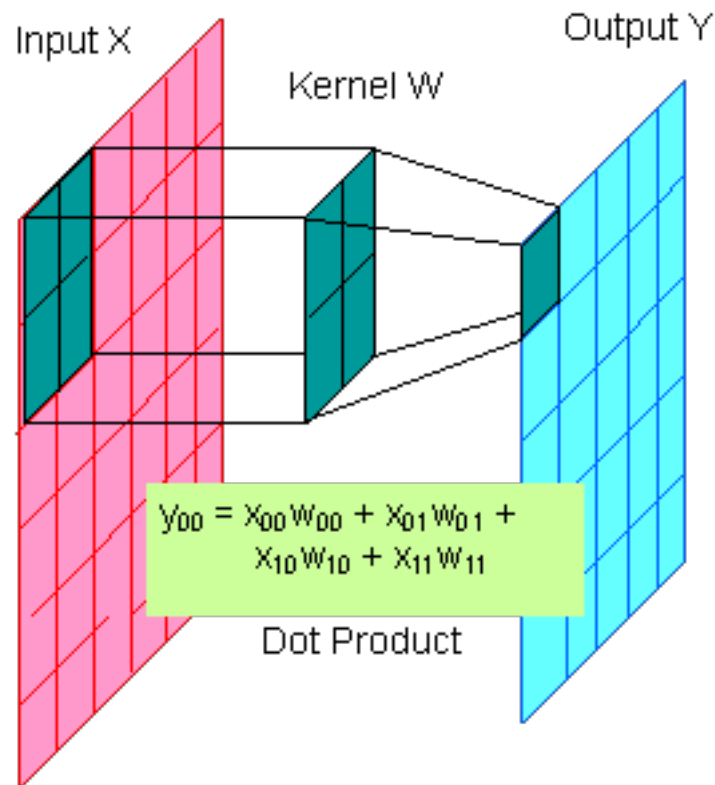
Discrete functions:

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

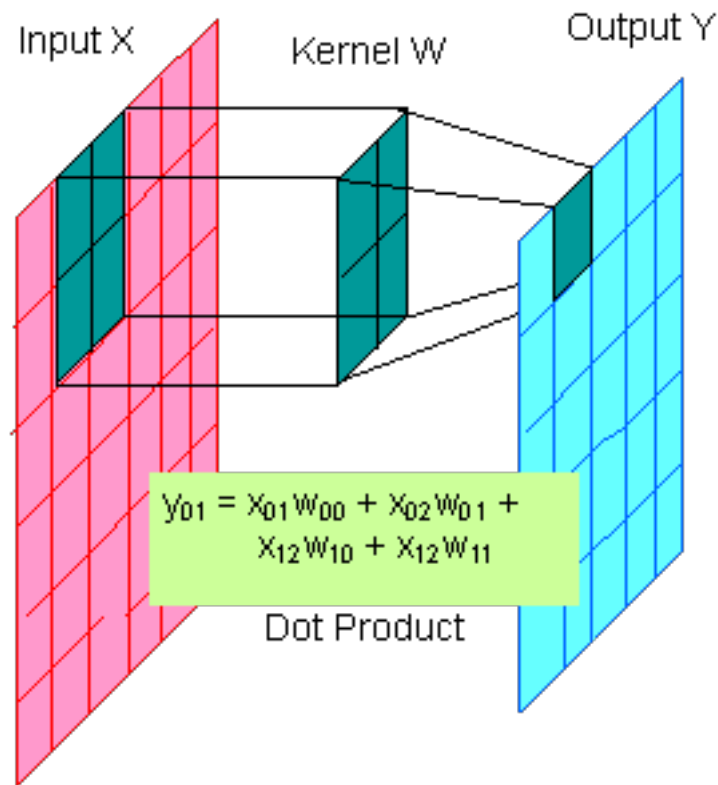
If discrete g has support on $\{-M, \dots, M\}$:

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

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

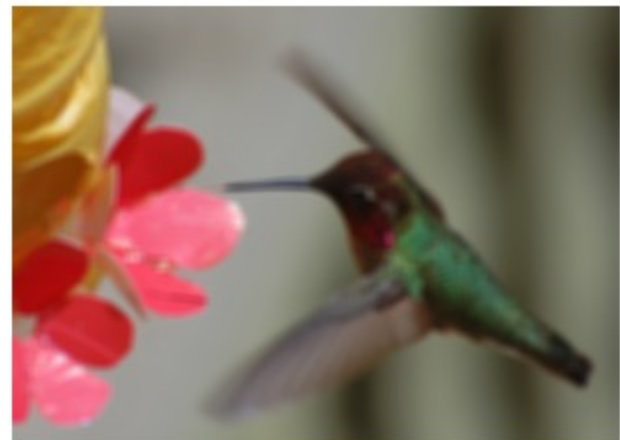


Filter (=kernel)

| | | | | |
|------|-------|-------|-------|------|
| 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 |

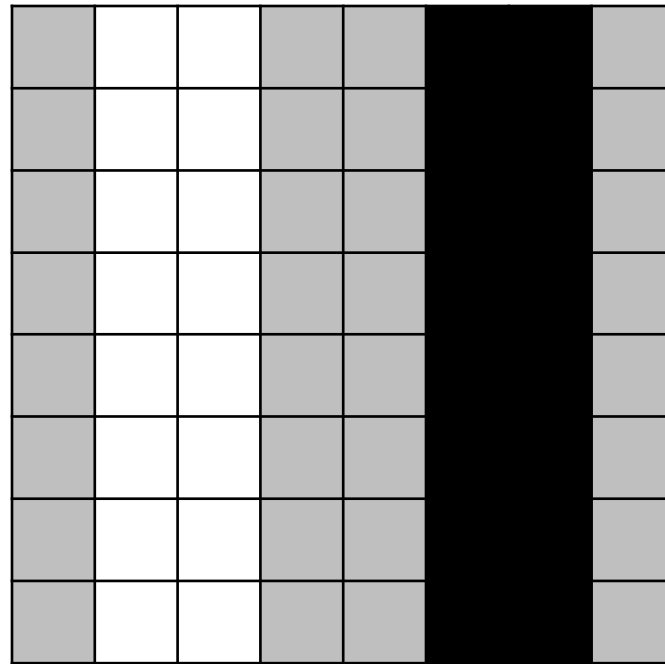
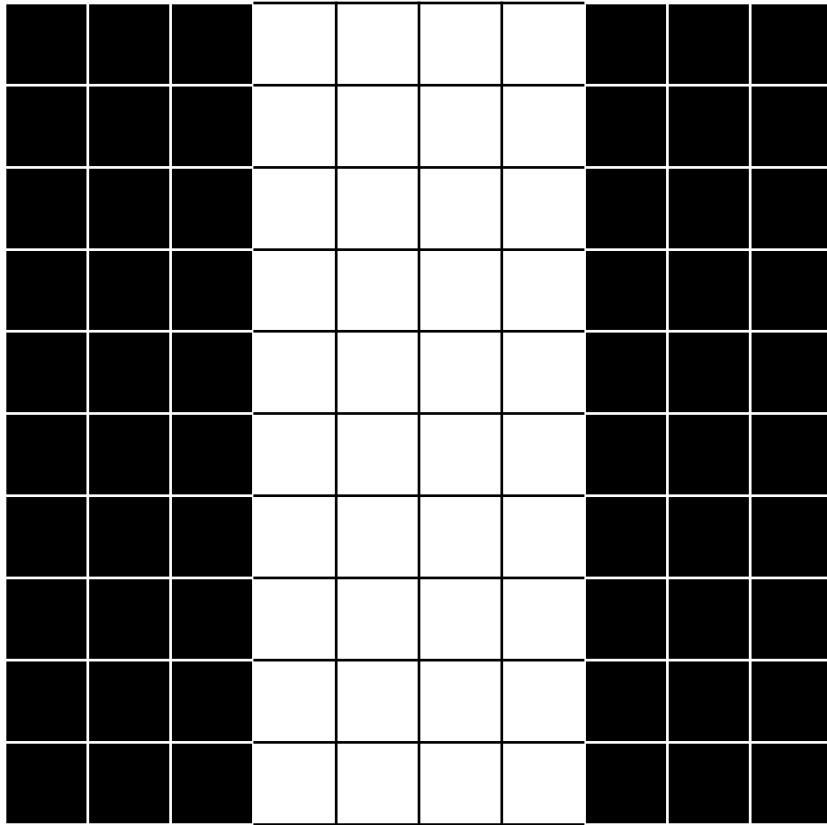


Convolution

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

| | | |
|----|---|---|
| -1 | 0 | 1 |
| -1 | 0 | 1 |
| -1 | 0 | 1 |

Convolution



| | | |
|----|---|---|
| -1 | 0 | 1 |
| -1 | 0 | 1 |
| -1 | 0 | 1 |

Poll: Which kernel goes with which output image?

Input



K1

| | | |
|----|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

K2

| | | |
|----|----|----|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

K3

| | | | |
|----|----|----|---|
| 0 | 0 | -1 | 0 |
| 0 | -2 | 0 | 1 |
| -1 | 0 | 2 | 0 |
| 0 | 1 | 0 | 0 |

Im1



Im2



Im3



Poll: Which kernel goes with which output image?

Input



K1

| | | |
|----|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

K2

| | | |
|----|----|----|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

K3

| | | | |
|----|----|----|---|
| 0 | 0 | -1 | 0 |
| 0 | -2 | 0 | 1 |
| -1 | 0 | 2 | 0 |
| 0 | 1 | 0 | 0 |

Im1



Im2

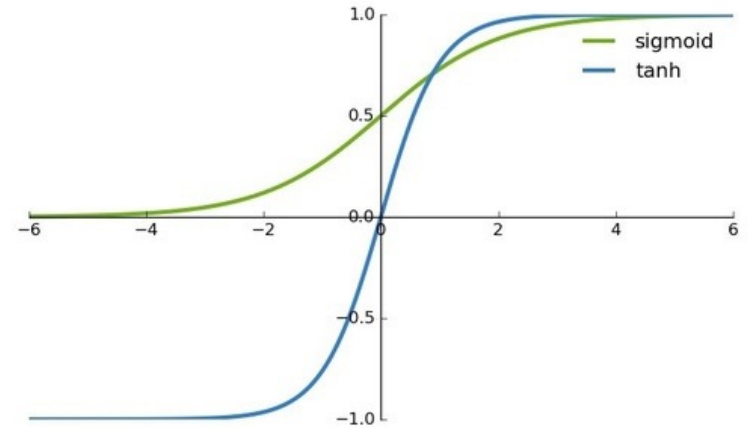
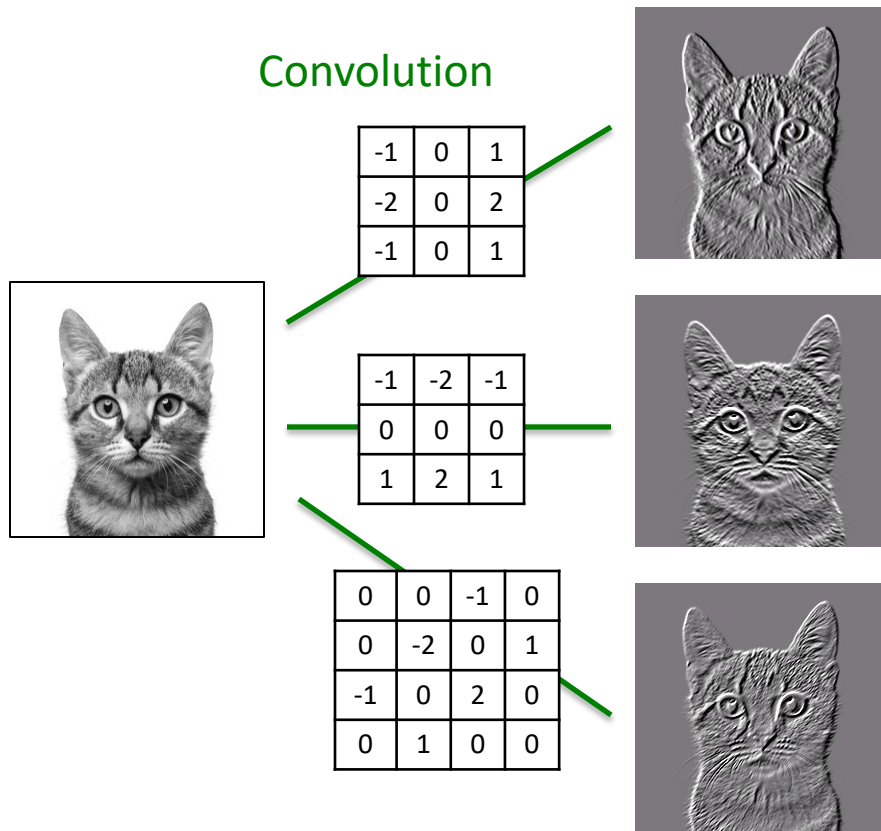


Im3



Convolutional Neural Networks

[Convolution + Nonlinear activation] + Pooling

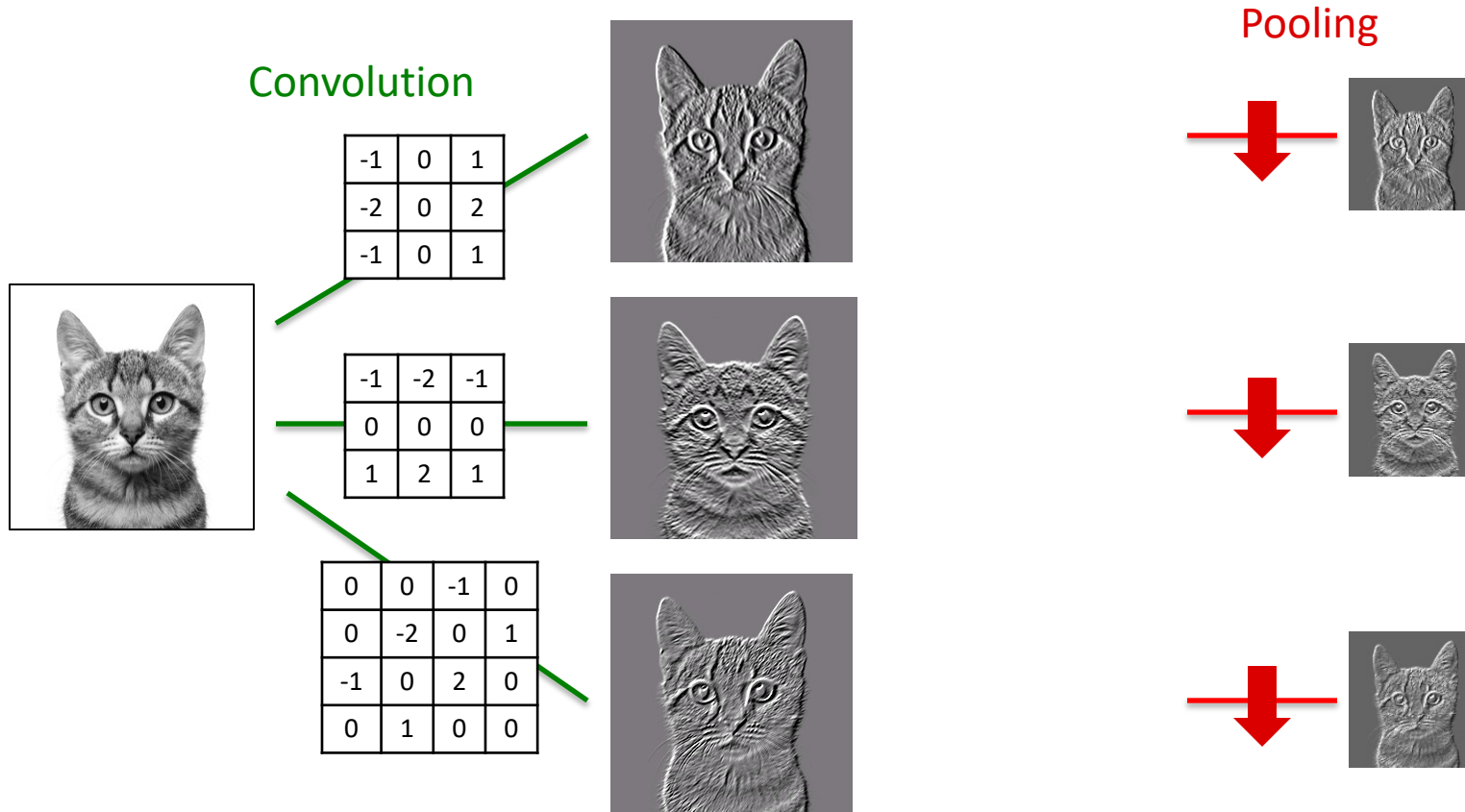


LeNet – tanh activation

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

Convolutional Neural Networks

[Convolution + Nonlinear activation] + Pooling



Pooling = Down-sampling

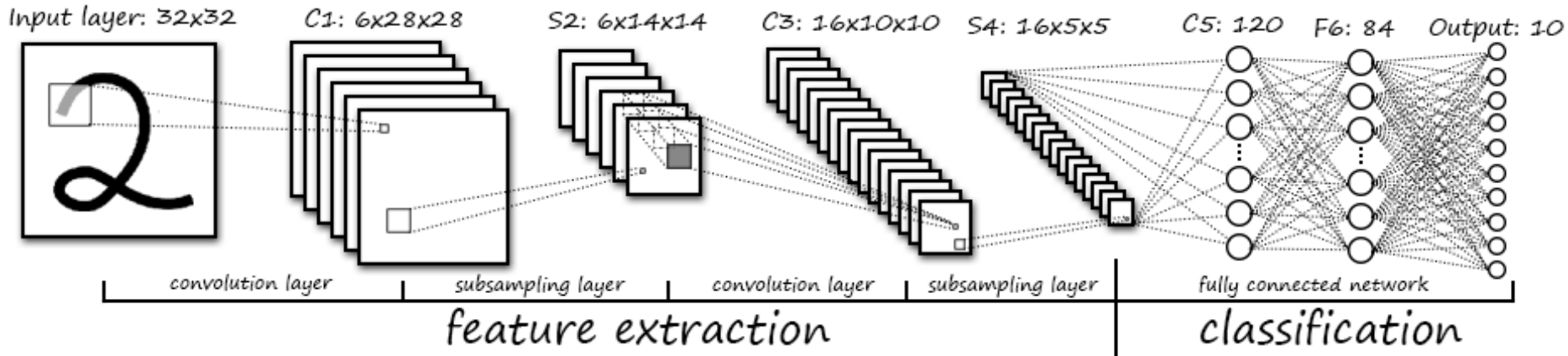
Reduce size to reduce number of parameters

Average pooling: convolution with stride = filter size

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

| | |
|-----|-----|
| .25 | .25 |
| .25 | .25 |

LeNet 5, LeCun et al 1998



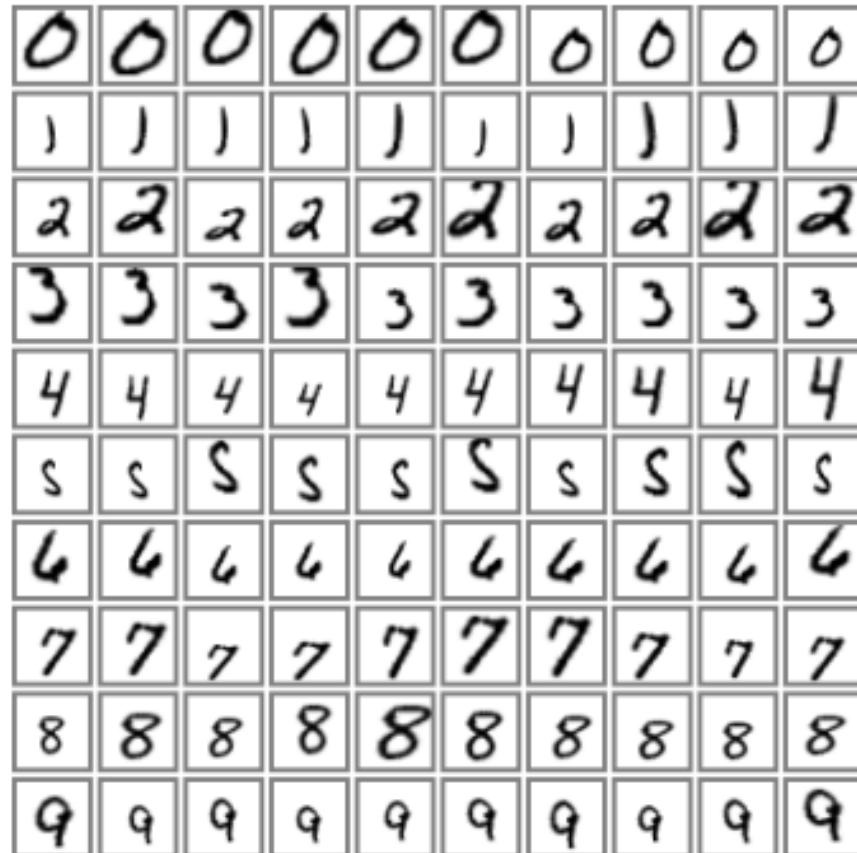
- **Input:** 32x32 pixel image. Largest character is 20x20 (All important info should be in the center of the receptive fields of the highest level feature detectors)
- **Cx:** Convolutional layer (C1, C3, C5) tanh nonlinear units
- **Sx:** Subsample layer (S2, S4) average pooling
- **Fx:** Fully connected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

MINIST Dataset



60,000 original dataset

Test error: 0.95%



540,000 artificial distortions

+ 60,000 original

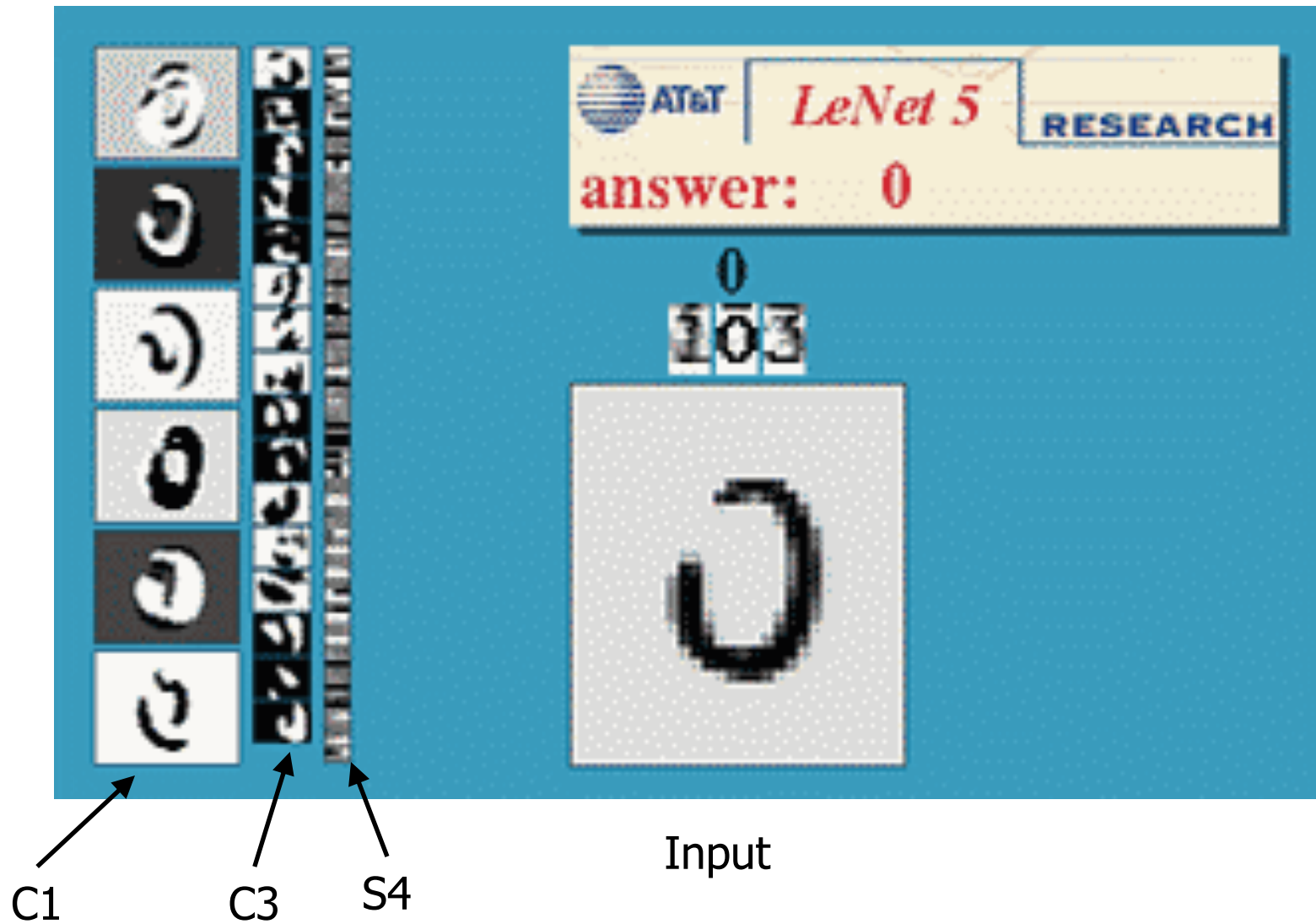
Test error: 0.8%

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



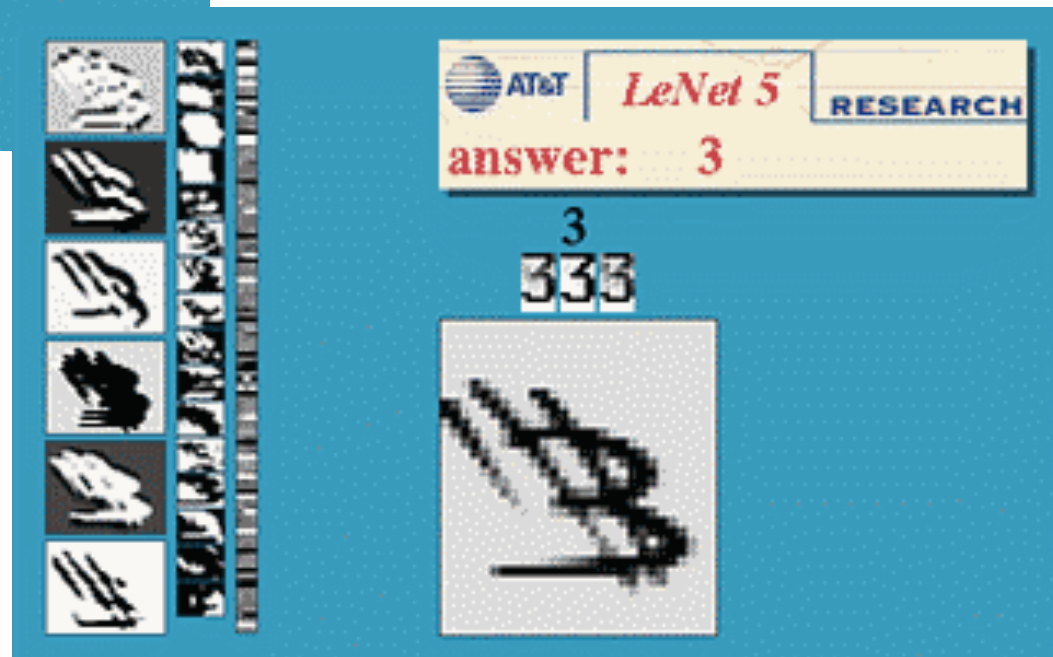
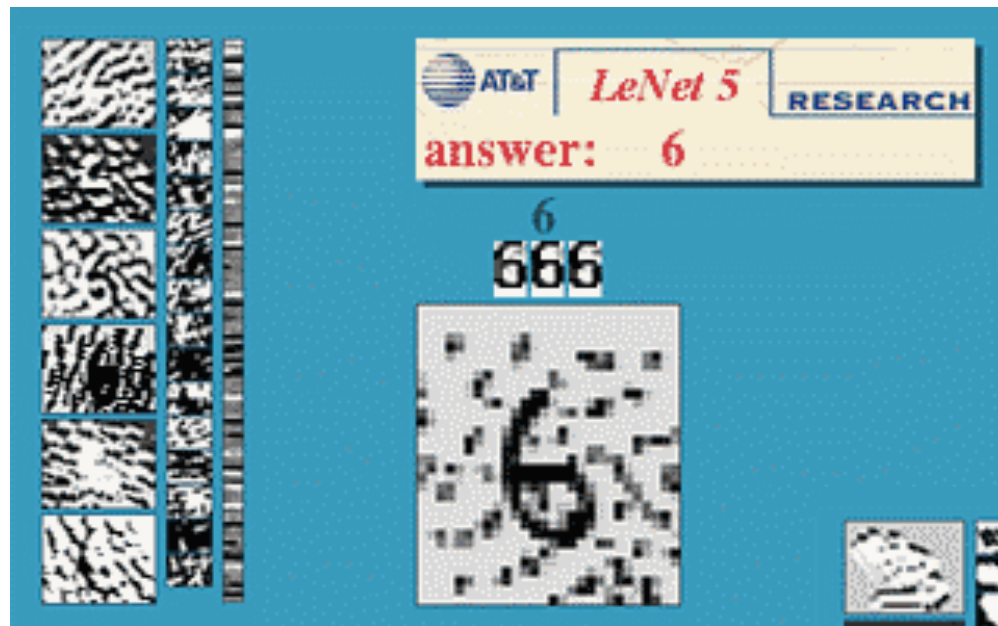
LeNet 5, Shift invariance



LeNet 5, Rotation invariance



LeNet 5, Noise resistance



LeNet 5, Unusual Patterns

AT&T *LeNet 5* RESEARCH
answer: 2

2
222

AT&T *LeNet 5* RESEARCH
answer: 3

3
333

AT&T *LeNet 5* RESEARCH
answer: 4

4
444

AT&T *LeNet 5* RESEARCH
answer: 6

6
666

ImageNet Classification with Deep Convolutional Neural Networks

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

Alex Net

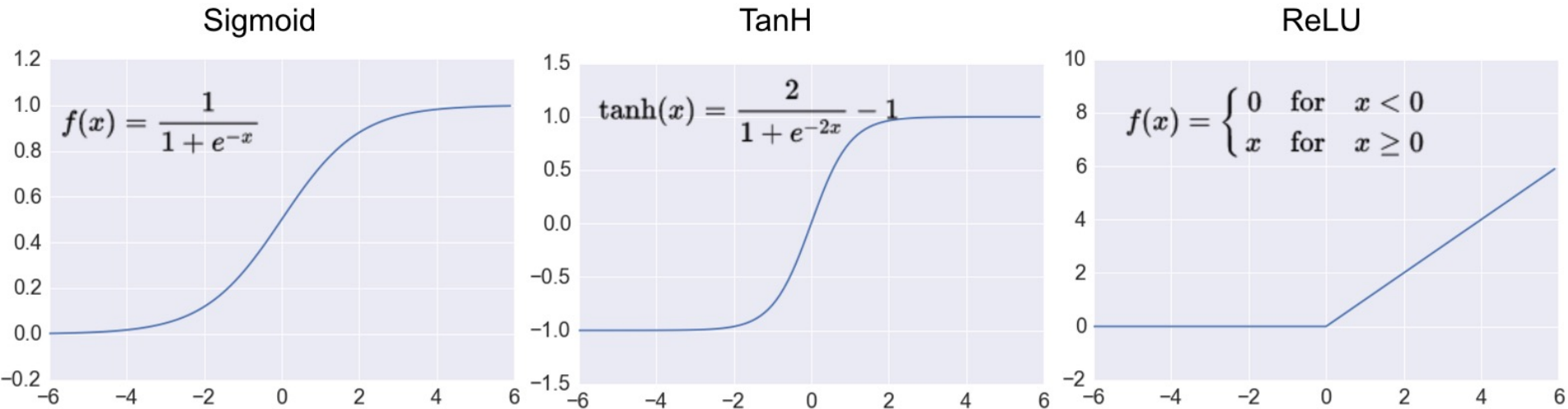
The Architecture

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

Non-saturating / Gradients don't vanish – faster training



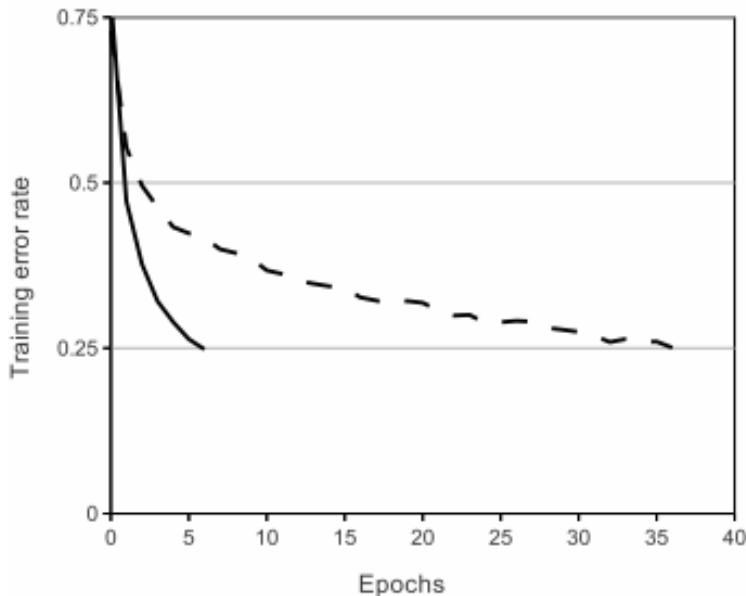
The Architecture

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

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

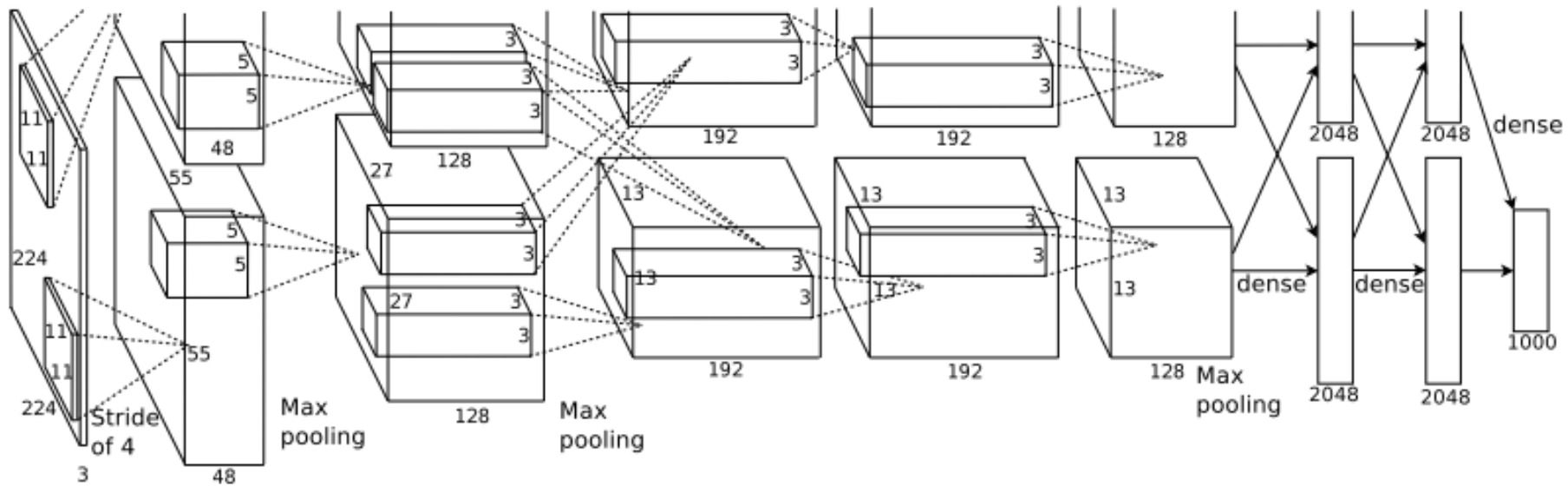
Here, **Rectified Linear Units (ReLU)** are used: $f(x) = \max(0, x)$

Non-saturating / Gradients don't vanish – faster training



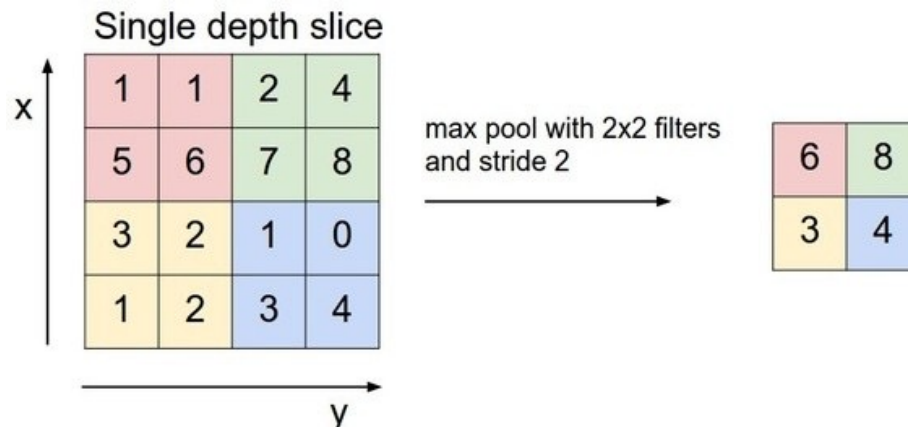
A four-layer convolutional neural network with ReLUs (solid line) reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with tanh neurons (dashed line)

The Architecture

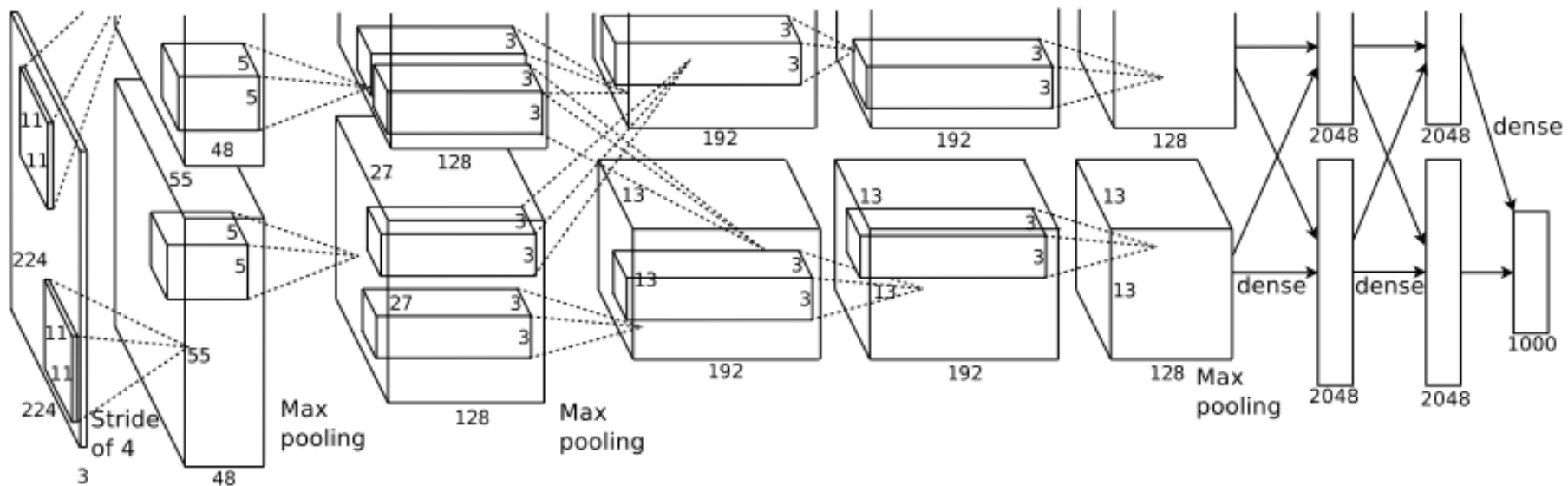


5 convolution layers (ReLU)

3 overlapping max pooling – nonlinear downsampling (max value of regions)



The Architecture



5 convolution layers (ReLU)

3 overlapping max pooling – nonlinear downsampling (max value of regions)

2 fully connected layers

output softmax

Training

- 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 with Rectified Linear Units (ReLUs), 3 overlapping max pooling, 2 fully connected layer
- ❑ Final feature layer: 4096-dimensional

- ❑ Prevent overfitting – data augmentation, dropout trick
- ❑ Randomly extracted 224x224 patches for more data

Preventing overfitting

1) **Data augmentation:** The easiest and most common method to reduce overfitting on image data is to artificially enlarge the dataset using label-preserving transformations:

- image translation
- horizontal reflections
- changing RGB intensities

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

- 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.
- forced to learn more robust features that are useful in conjunction with many different random subsets of the other neurons.

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

Results



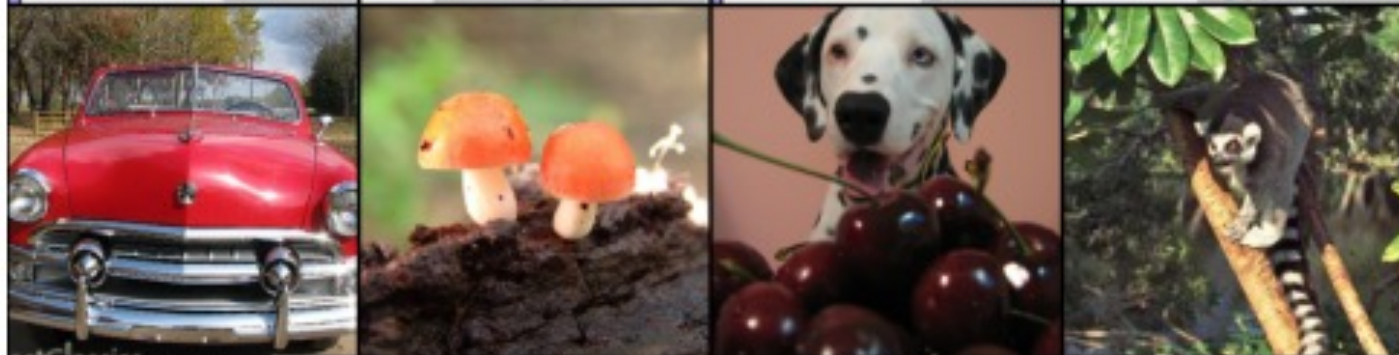
mite

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 |

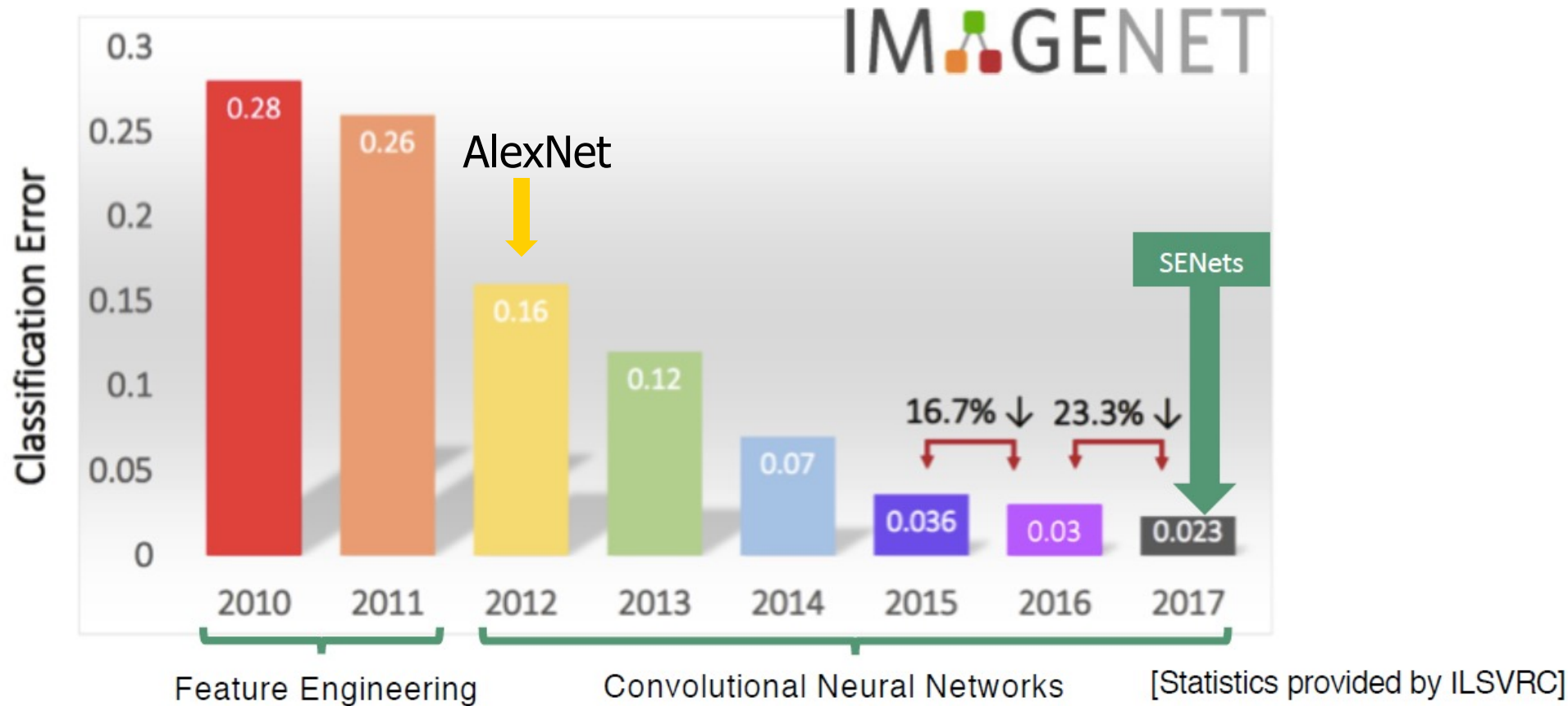
Results: Image similarity



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.

Results



Other optimization tips and tricks

- **Momentum:** use exponentially weighted sum of previous gradients

$$\bar{\nabla}_{\theta}^{(t)} = \nabla_{\theta} l(\mathbf{f}(\mathbf{x}^{(t)}), y^{(t)}) + \beta \bar{\nabla}_{\theta}^{(t-1)}$$

can get pass plateaus more quickly, by “gaining momentum”

- **Initialization:** cannot initialize to same value, all units in a hidden layer will behave same; randomly initialize unif[-b,b]

- **Adaptive learning rates:** one learning rate per parameter

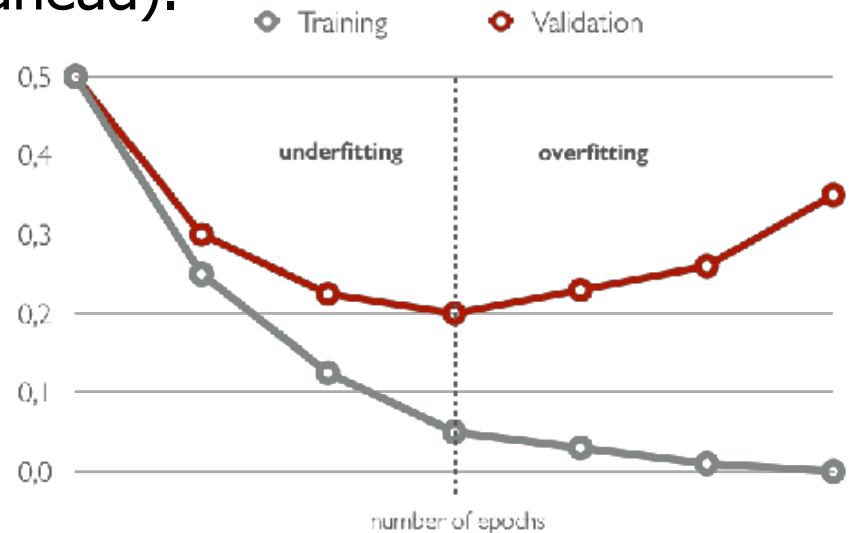
e.g. RMSProp uses exponentially weighted average of squared gradients

$$\gamma^{(t)} = \beta \gamma^{(t-1)} + (1 - \beta) \left(\nabla_{\theta} l(\mathbf{f}(\mathbf{x}^{(t)}), y^{(t)}) \right)^2 \quad \bar{\nabla}_{\theta}^{(t)} = \frac{\nabla_{\theta} l(\mathbf{f}(\mathbf{x}^{(t)}), y^{(t)})}{\sqrt{\gamma^{(t)} + \epsilon}}$$

Adam combines RMSProp with momentum

Tips and tricks for preventing overfitting

- **Dropout**
- **Data augmentation**
- **Early stopping:** stop training when validation set error increases (with some look ahead).



- **Neural Architecture search:** tune number of layers and neurons per layer using grid search or clever optimization

Tips and Tricks for training deep NNs

- First hypothesis (**underfitting**): better optimize
 - Increase the capacity of the neural network
 - Check initialization
 - Check gradients (saturating units and vanishing gradients)
 - Tune learning rate
- Second hypothesis (**overfitting**): use better regularization
 - Dropout
 - Data augmentation
 - Early stopping
 - Architecture search
- For many large-scale practical problems, you will need to use both: better optimization and better regularization!

Artificial Neural Networks: Summary

- Used to mimic distributed computation in brain
- Highly non-linear regression/classification
- Vector-valued inputs and outputs
- Potentially millions of parameters to estimate - overfitting
- Hidden layers learn intermediate representations – how many to use?

- Prediction – Forward propagation
- Gradient descent (Back-propagation), local minima problems

- Coming back in new form as deep networks