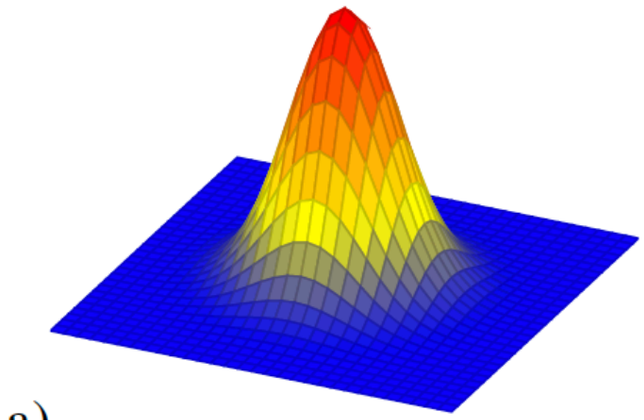
Learning Based Vision I

Computer Vision
Fall 2019
Columbia University

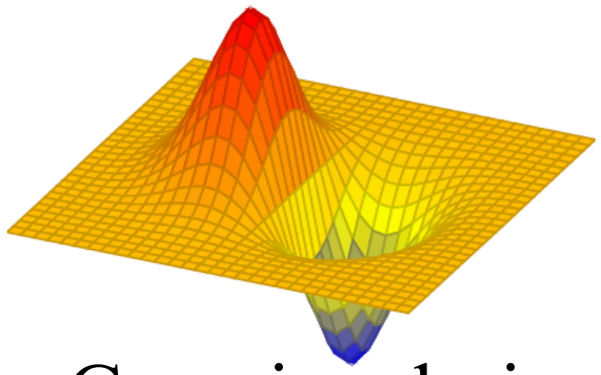
We need translation invariance

Slide credit: Antonio Torralba

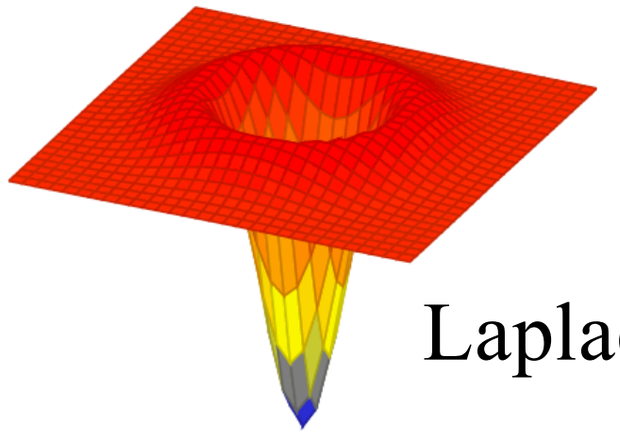
Lots of useful linear filters…



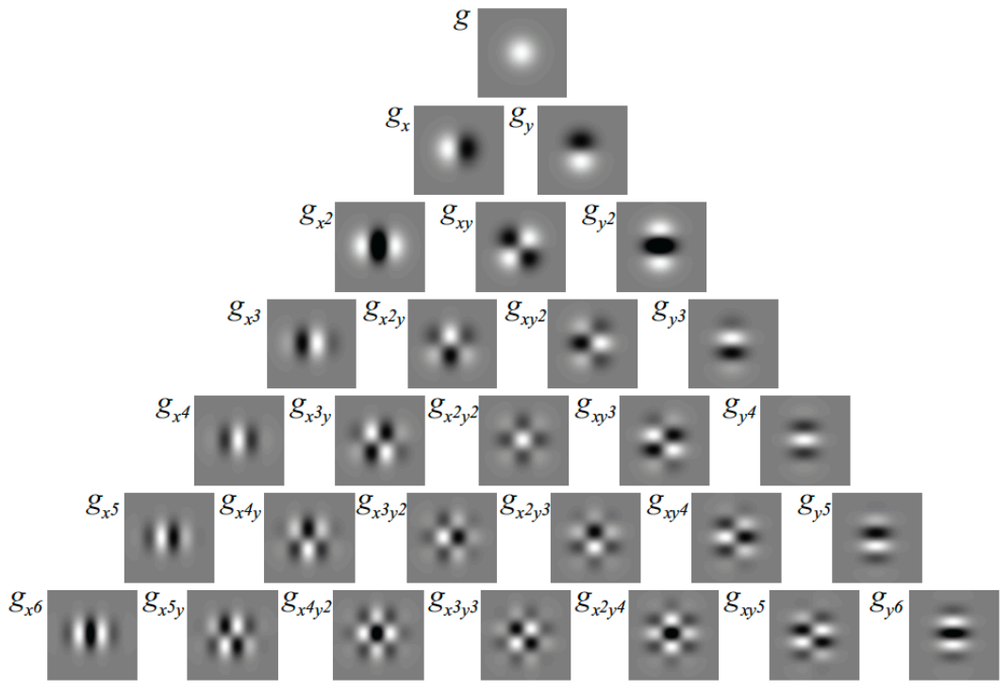
a) Gaussian



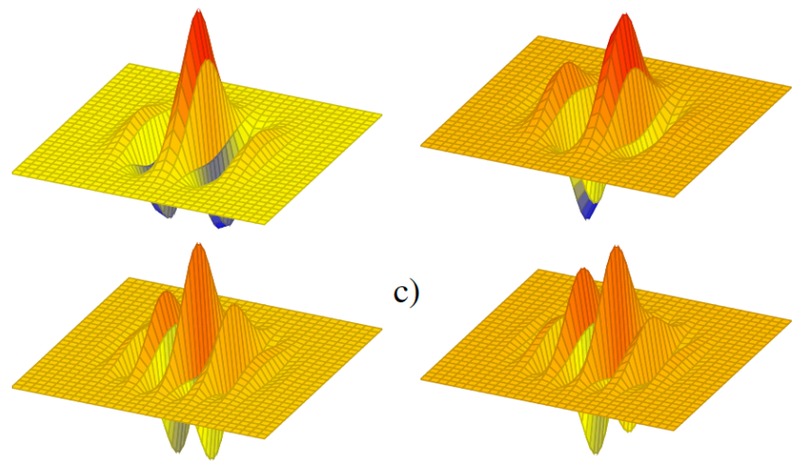
b) Gaussian derivative



c) Laplacian



High order Gaussian derivatives



Gabor

And many more…

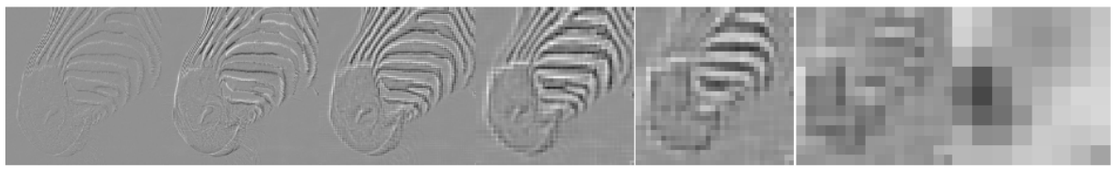
Slide credit: Antonio Torralba

We need translation and scale invariance

Slide credit: Antonio Torralba

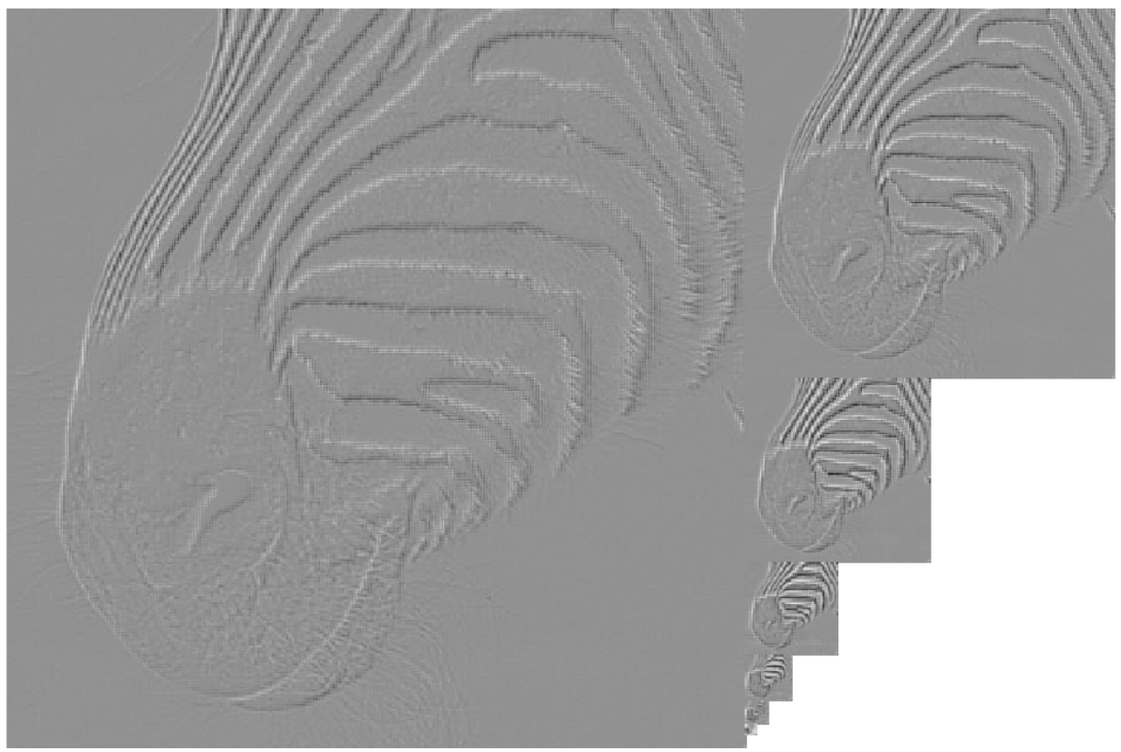
Lots of image pyramids…

512 256 128 64 32 16 8



512 256 128 64 32 16 8

Gaussian Pyr



Laplacian Pyr

And many more: QMF, steerable, …

Slide credit: Antonio Torralba

We need …

Slide credit: Antonio Torralba

What is the best representation?

- All the previous representation are manually constructed.
- Could they be learnt from data?

Slide credit: Antonio Torralba

A brief history of Neural Networks

Slide credit: Antonio Torralba

Where did this all start?

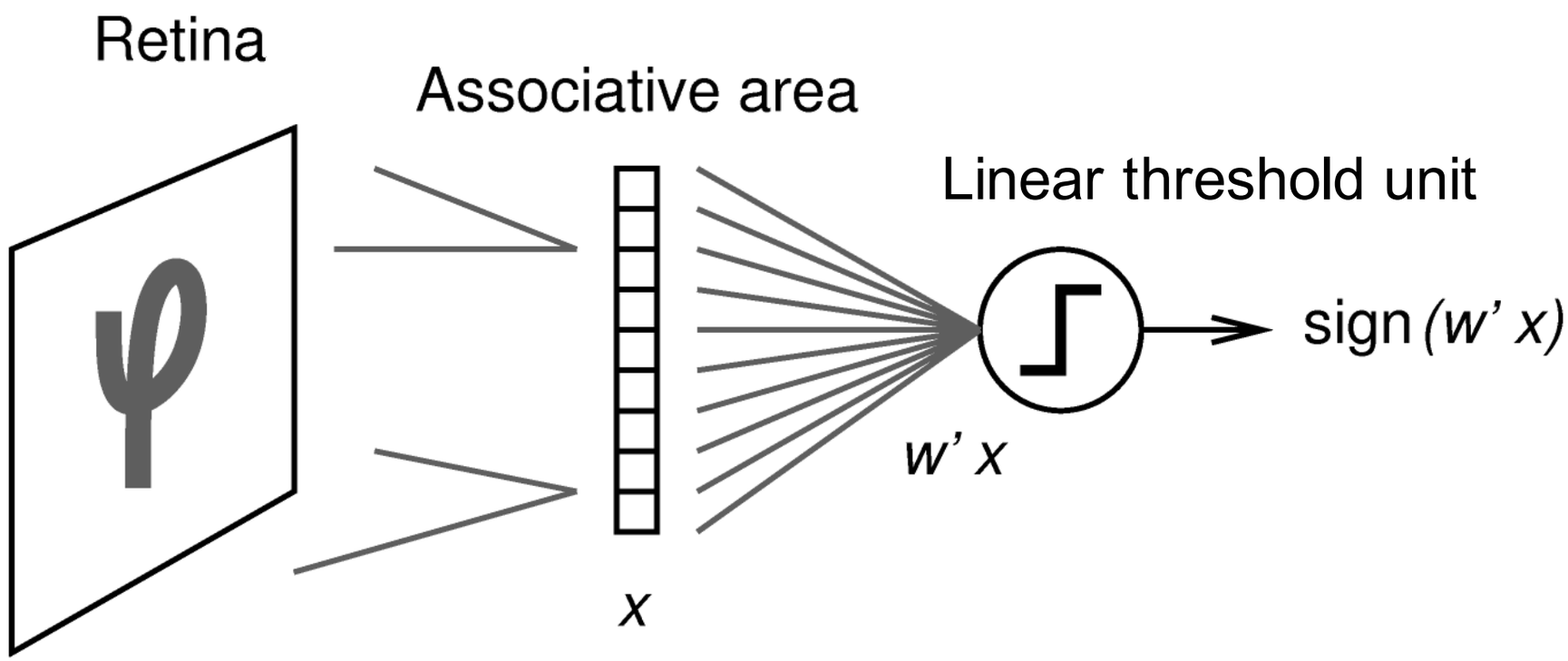
Hubel & Wiesel (1962)

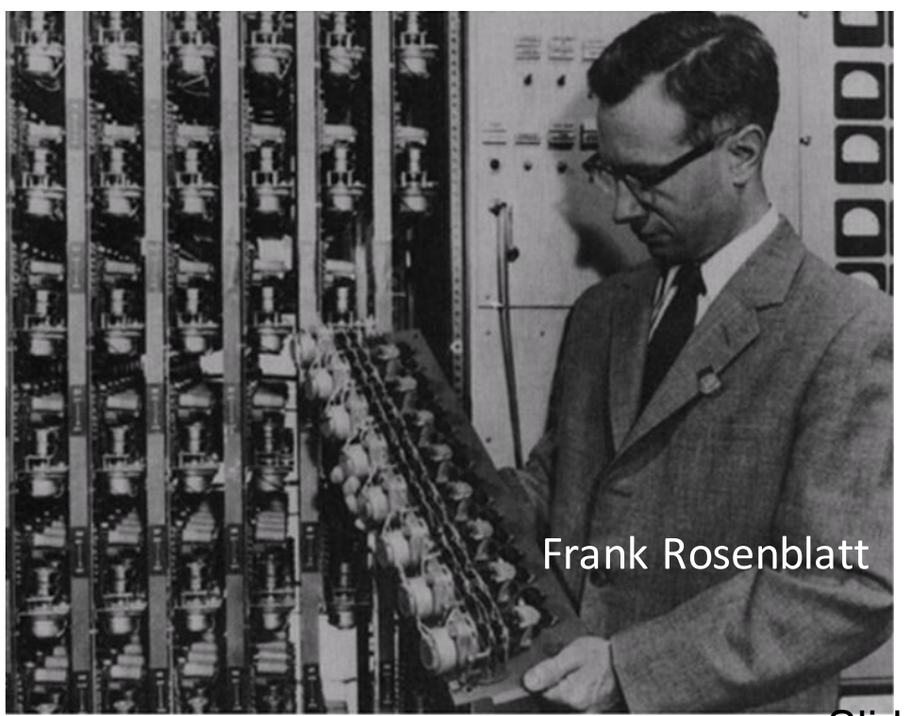
Insights about early image processing in the brain.

- Simple cells detect local features
- Complex cells pool local features in a retinotopic neighborhood

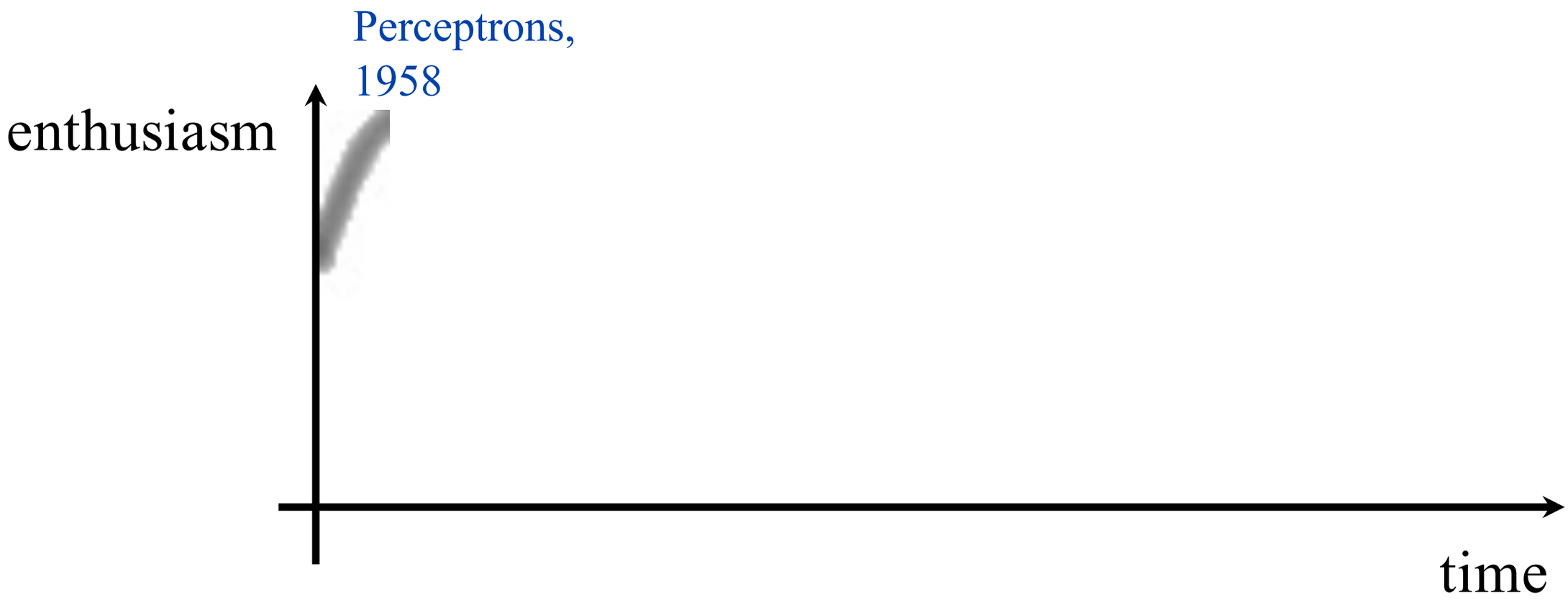
Slide credit: Deva Ramanan

The perceptron





Slide credit: Deva Ramanan

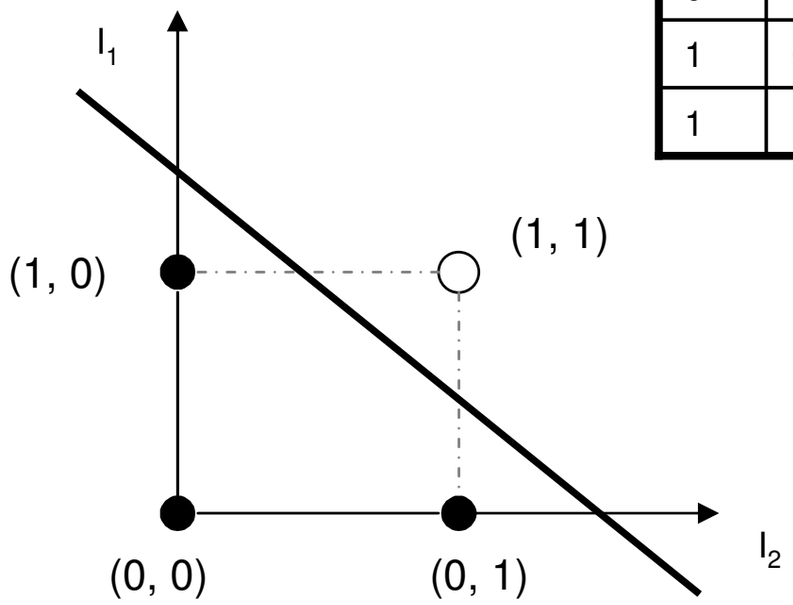


Slide credit: Antonio Torralba

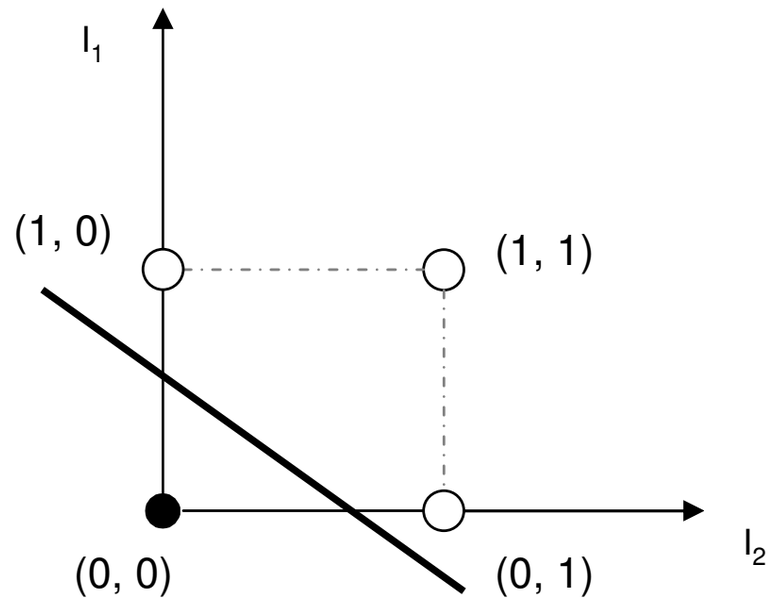
Percepton

If input features are binary, can model logical “and”s and “or”s

AND		
I_1	I_2	out
0	0	0
0	1	0
1	0	0
1	1	1

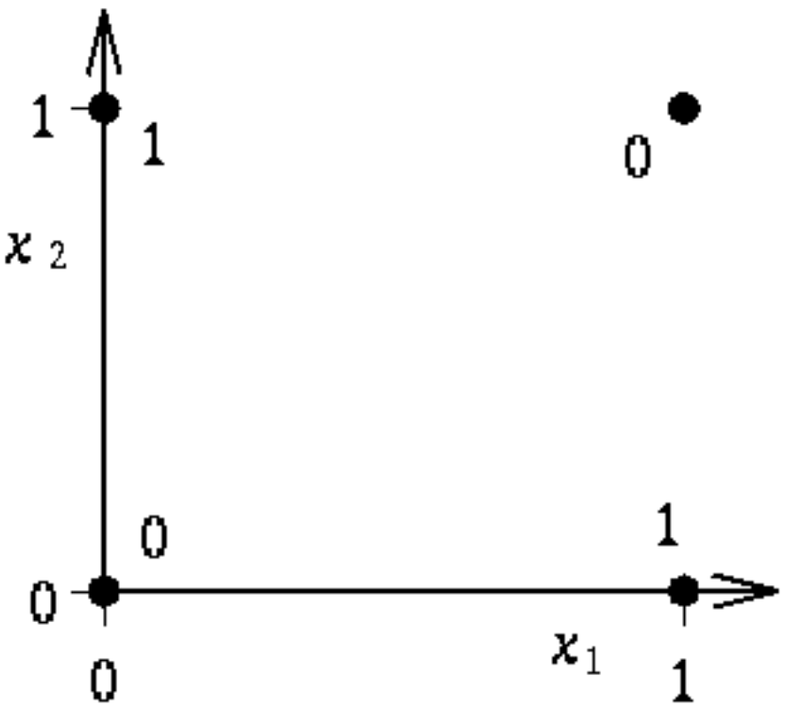


OR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	1



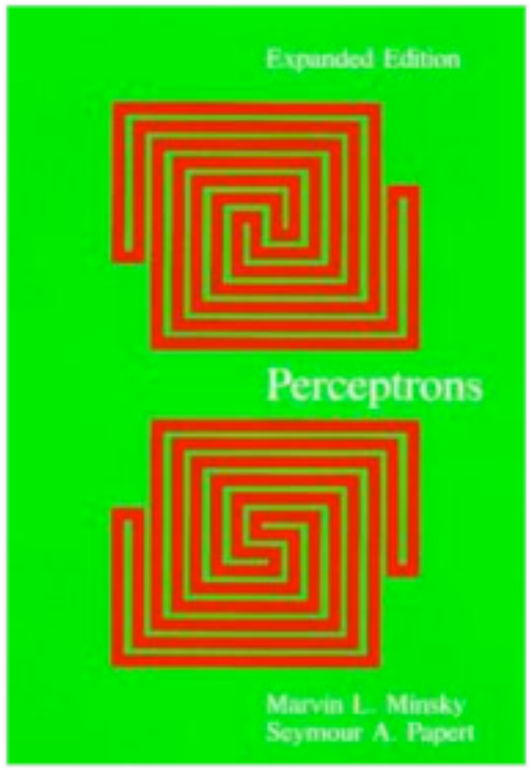
Slide credit: Deva Ramanan

What about “xors”?



Slide credit: Deva Ramanan

Minsky and Papert, Perceptrons, 1972



Perceptrons, expanded edition

An Introduction to Computational Geometry

By Marvin Minsky and Seymour A. Papert

Overview

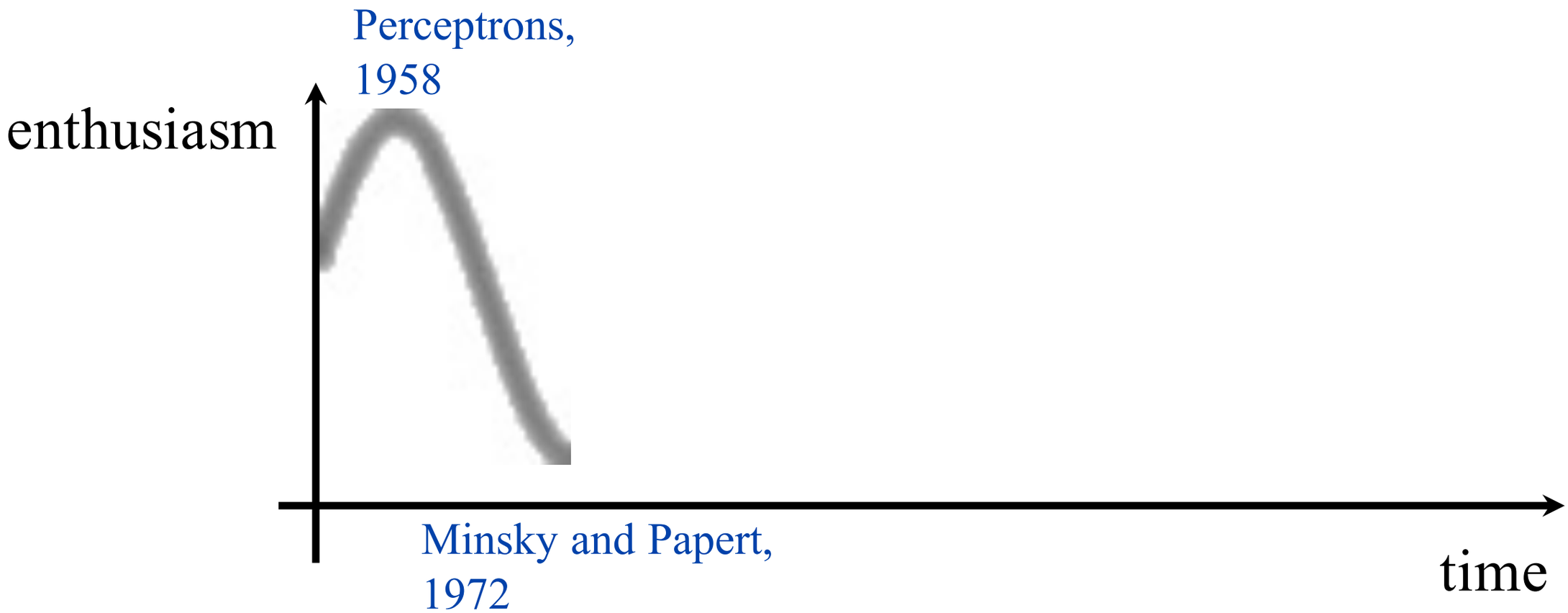
Perceptrons - the first systematic study of parallelism in computation - has remained a classical work on threshold automata networks for nearly two decades. It marked a historical turn in artificial intelligence, and it is required reading for anyone who wants to understand the connectionist counterrevolution that is going on today.

Artificial-intelligence research, which for a time concentrated on the programming of ton Neumann computers, is swinging back to the idea that intelligence might emerge from the activity of networks of neuronlike entities. Minsky and Papert's book was the first example of a mathematical analysis carried far enough to show the exact limitations of a class of computing machines that could seriously be considered as models of the brain. Now the new developments in mathematical tools, the recent interest of physicists in the theory of disordered matter, the new insights into and psychological models of how the brain works, and the evolution of fast computers that can simulate networks of automata have given *Perceptrons* new importance.

Witnessing the swing of the intellectual pendulum, Minsky and Papert have added a new chapter in which they discuss the current state of parallel computers, review developments since the appearance of the 1972 edition, and identify new research directions related to connectionism. They note a central theoretical challenge facing connectionism: the challenge to reach a deeper understanding of how "objects" or "agents" with individuality can emerge in a network. Progress in this area would link connectionism with what the authors have called "society theories of mind."

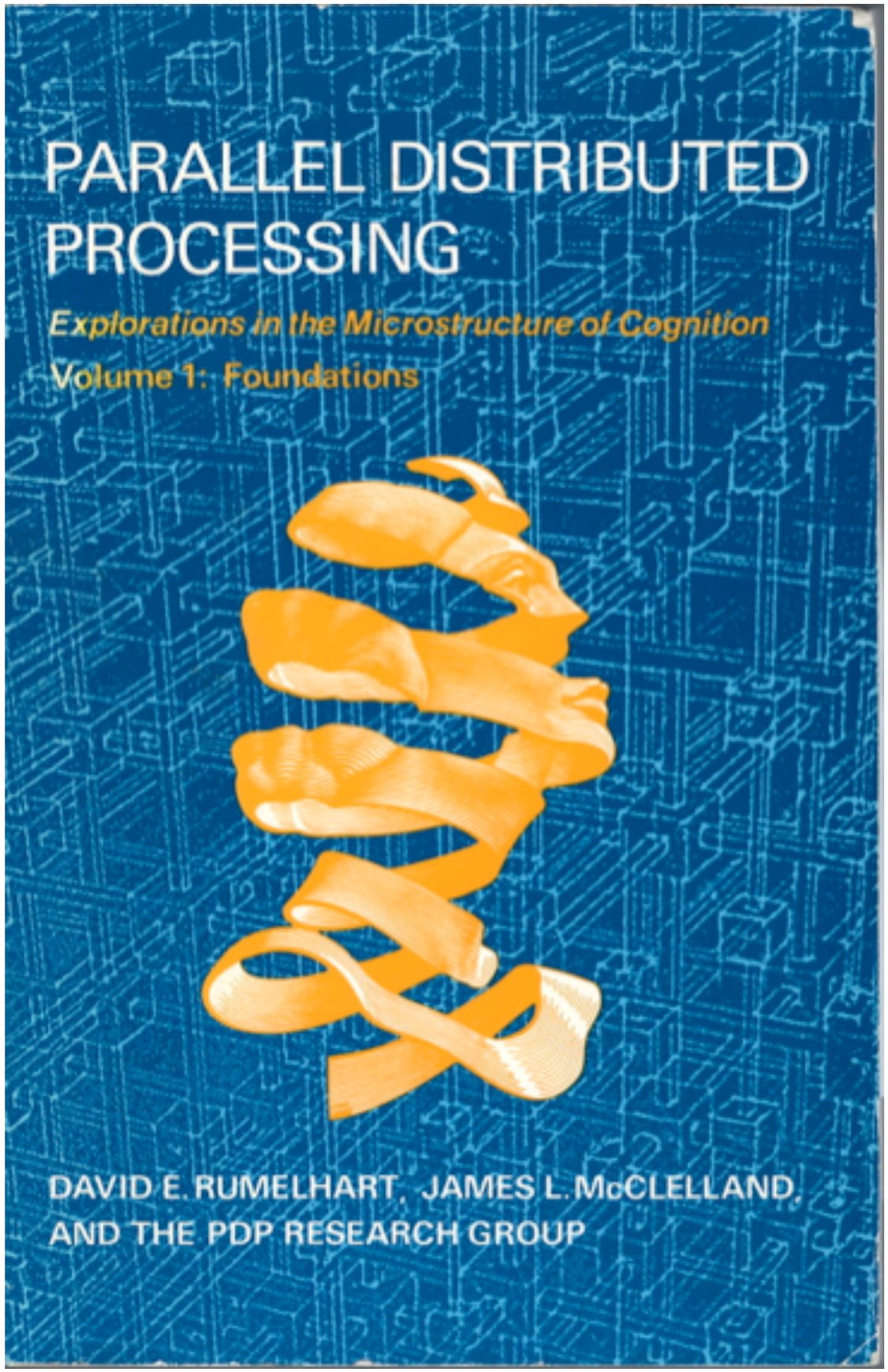
Paperback | $35.00 Short | £24.95 | ISBN: 9780262631112 | 308 pp. | 6 x 8.9 in | December 1987

Slide credit: Antonio Torralba



Slide credit: Antonio Torralba

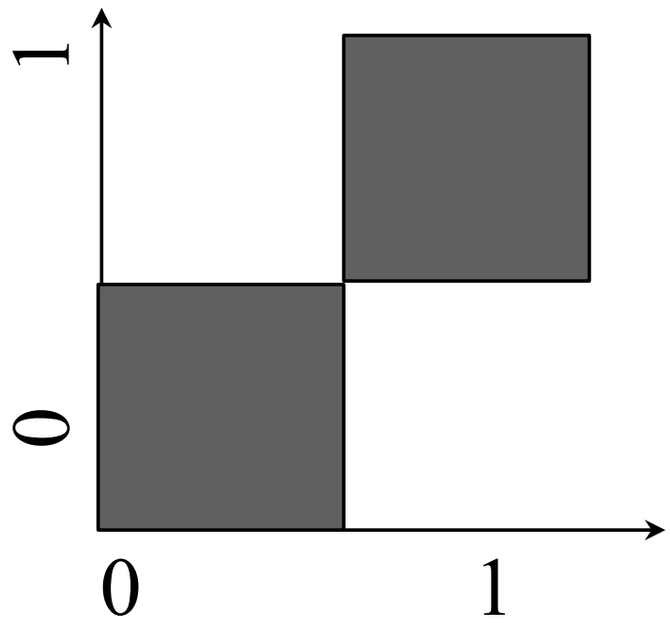
Parallel Distributed Processing (PDP), 1986



Slide credit: Antonio Torralba

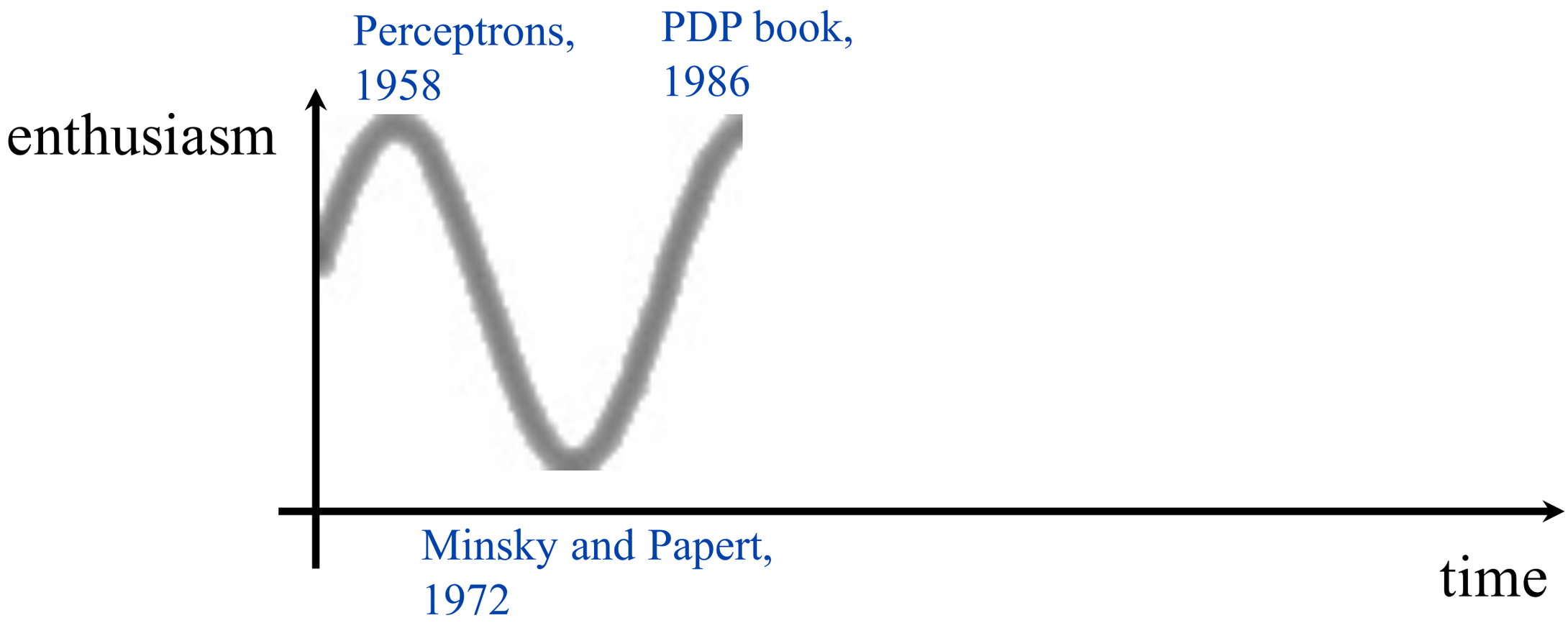
XOR problem

Inputs		Output
0	0	0
1	0	1
0	1	1
1	1	0



PDP authors pointed to the backpropagation algorithm as a breakthrough, allowing multi-layer neural networks to be trained. Among the functions that a multi-layer network can represent but a single-layer network cannot: the XOR function.

Slide credit: Antonio Torralba



Slide credit: Antonio Torralba

LeCun conv nets, 1998

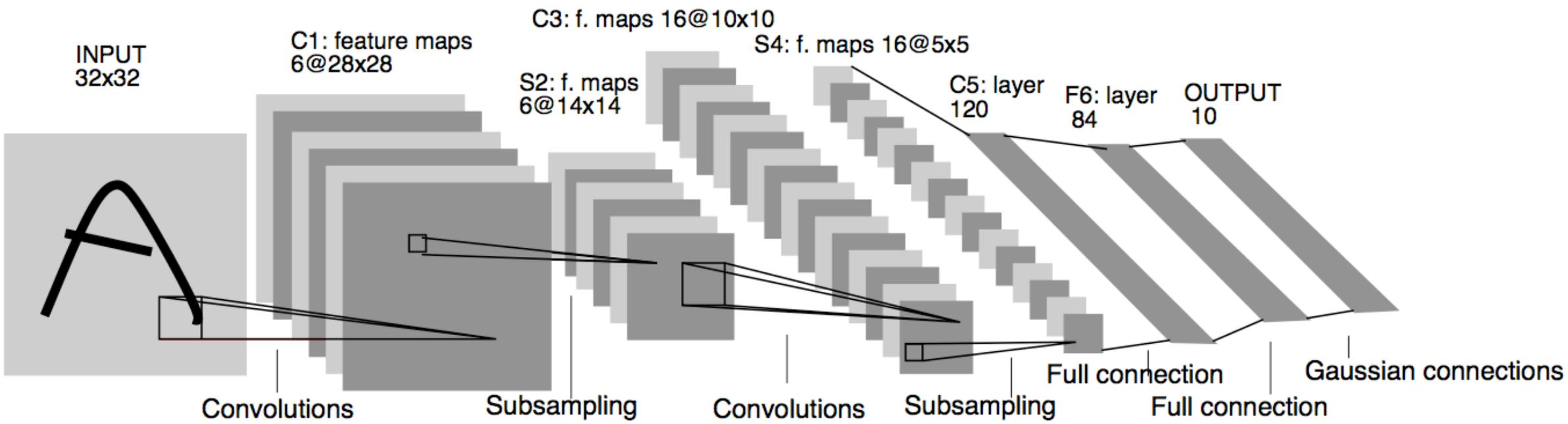


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Demos:

http://yann.lecun.com/exdb/lenet/index.html

Slide credit: Antonio Torralba

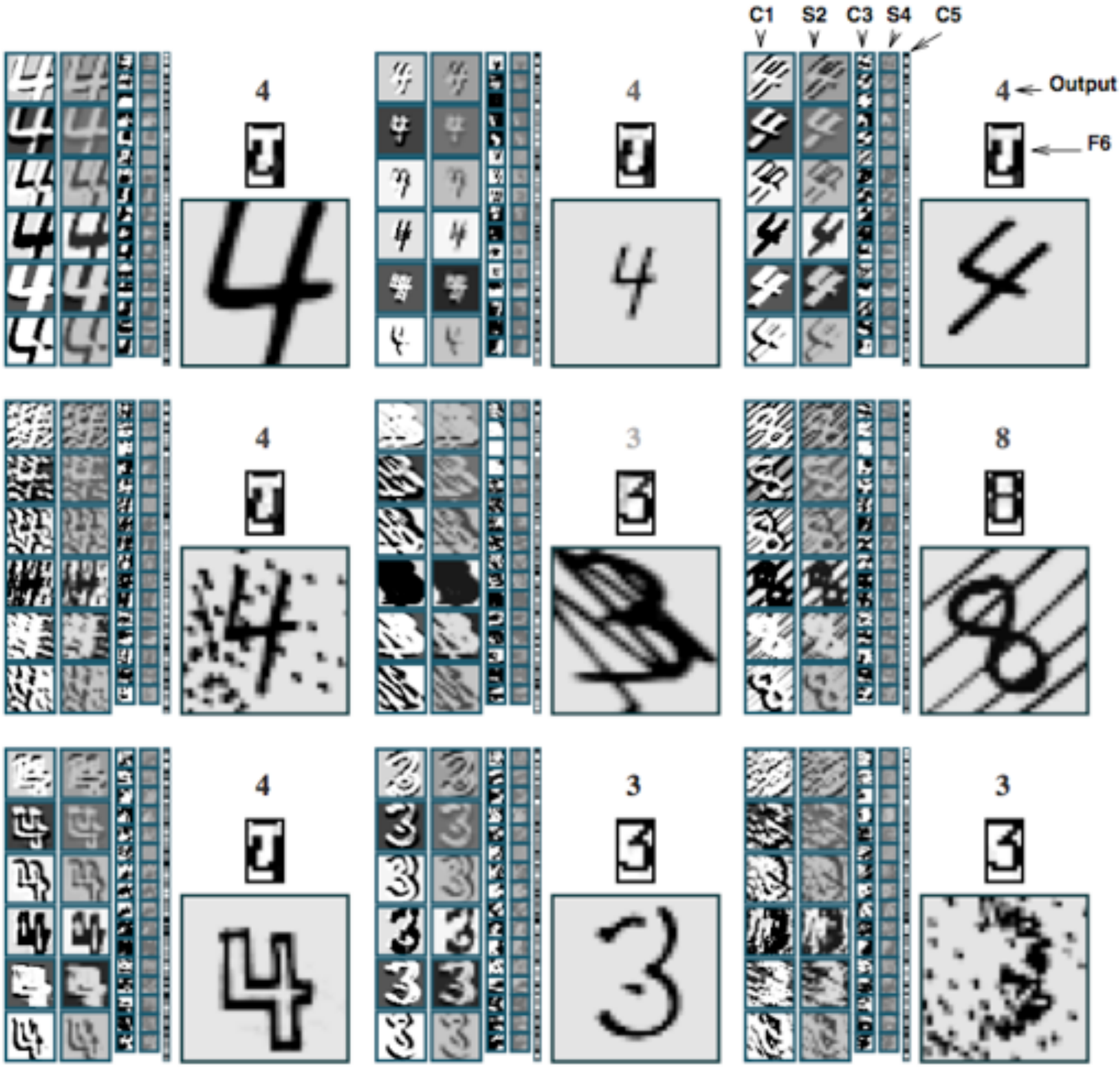
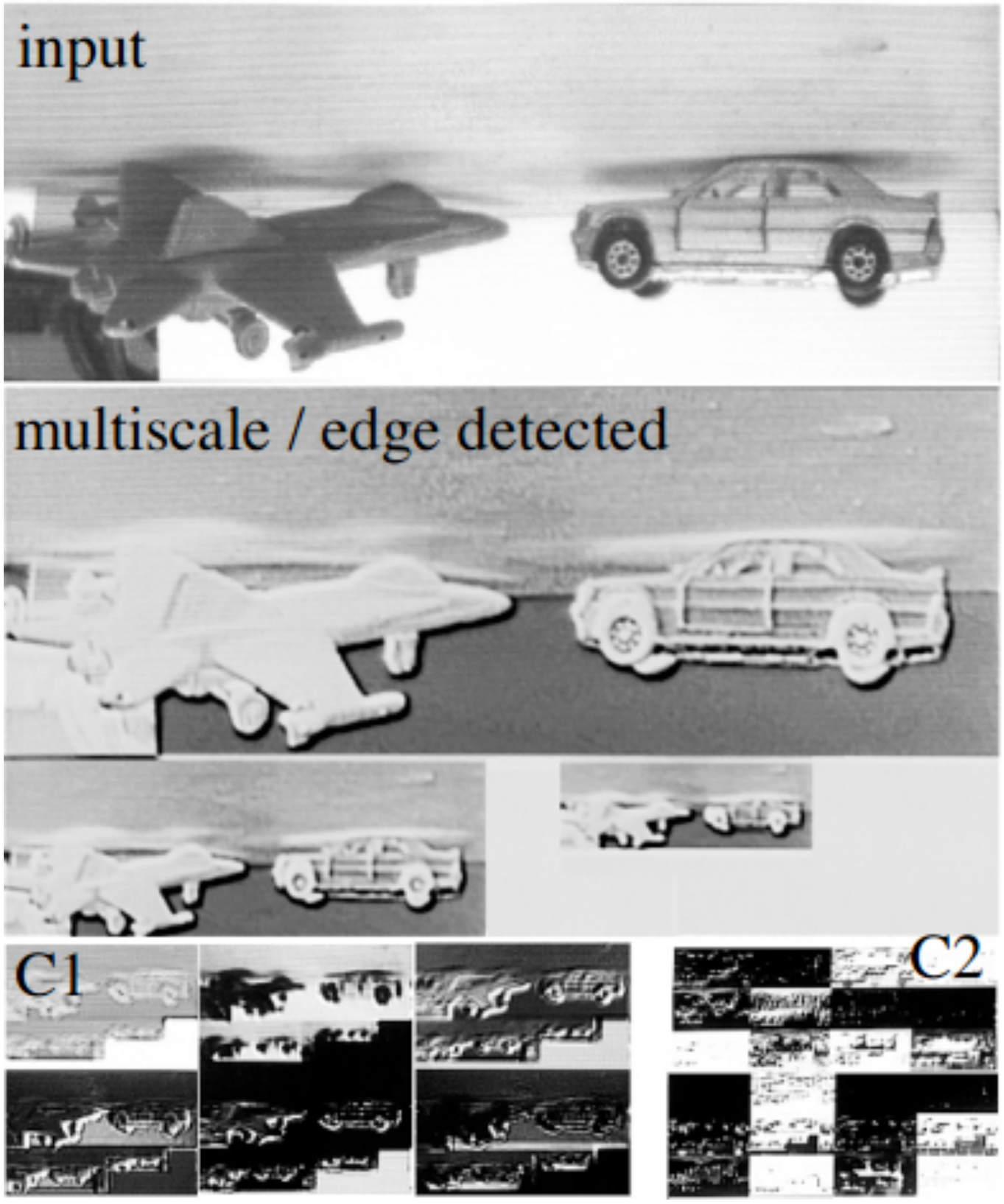


Fig. 13. Examples of unusual, distorted, and noisy characters correctly recognized by LeNet-5. The grey-level of the output label represents the penalty (lighter for higher penalties).

Slide credit: Antonio Torralba



Neural networks to recognize handwritten digits? yes

Neural networks for tougher problems? not really

http://pub.clement.farabet.net/ecvw09.pdf

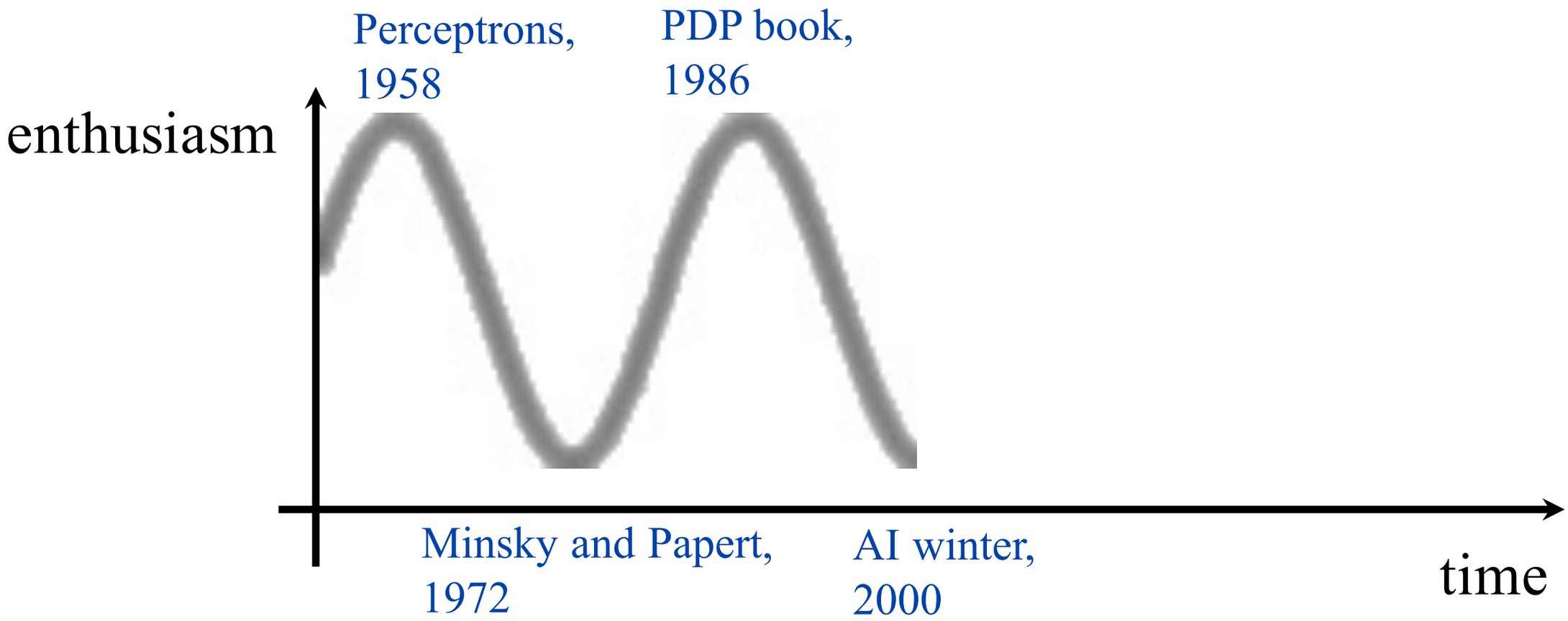
Slide credit: Antonio Torralba

NIPS 2000

- NIPS, Neural Information Processing Systems, is the premier conference on machine learning. Evolved from an interdisciplinary conference to a machine learning conference.

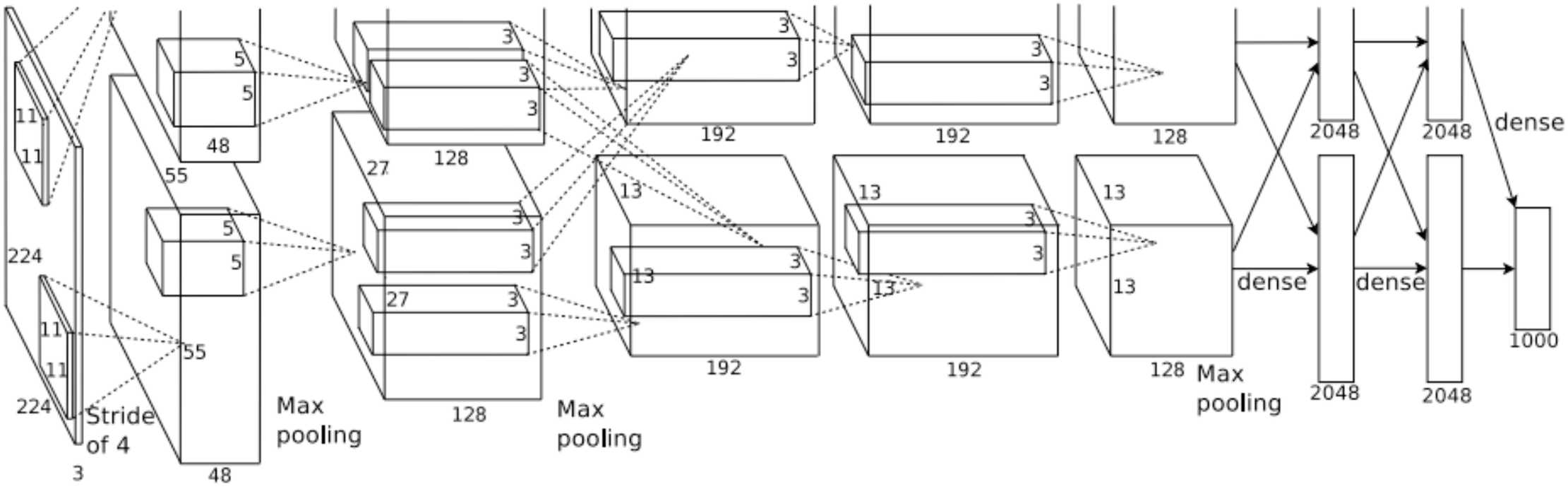
- For the NIPS 2000 conference:
  - title words predictive of paper acceptance: "Belief Propagation" and "Gaussian".
  - title words predictive of paper rejection: "Neural" and "Network".

Slide credit: Antonio Torralba



Slide credit: Antonio Torralba

Krizhevsky, Sutskever, and Hinton, NIPS 2012

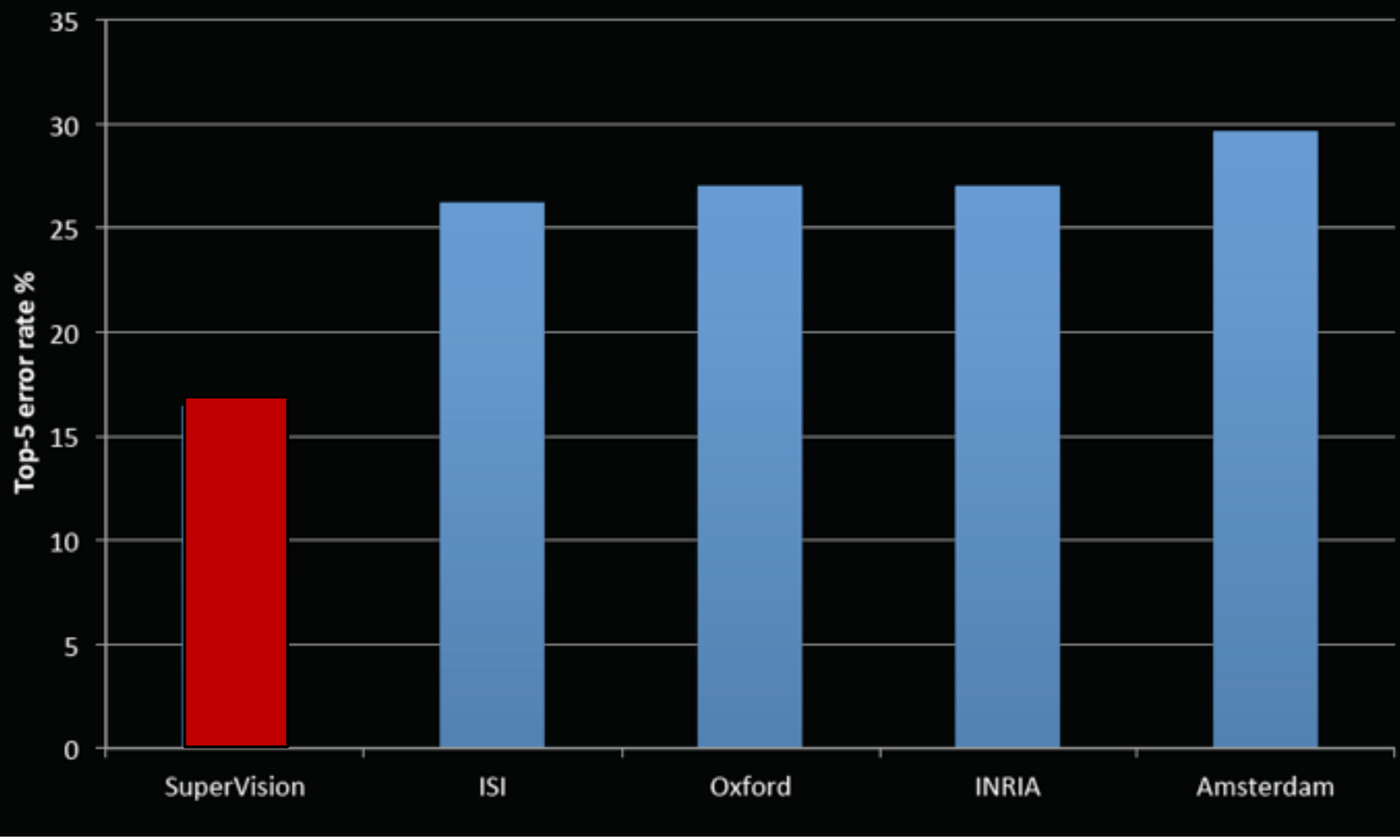


Slide credit: Antonio Torralba

Slide from Rob Fergus, NYU

ImageNet Classification 2012

- Krizhevsky et al. -- 16.4% error (top-5)
- Next best (non-convnet) – 26.2% error



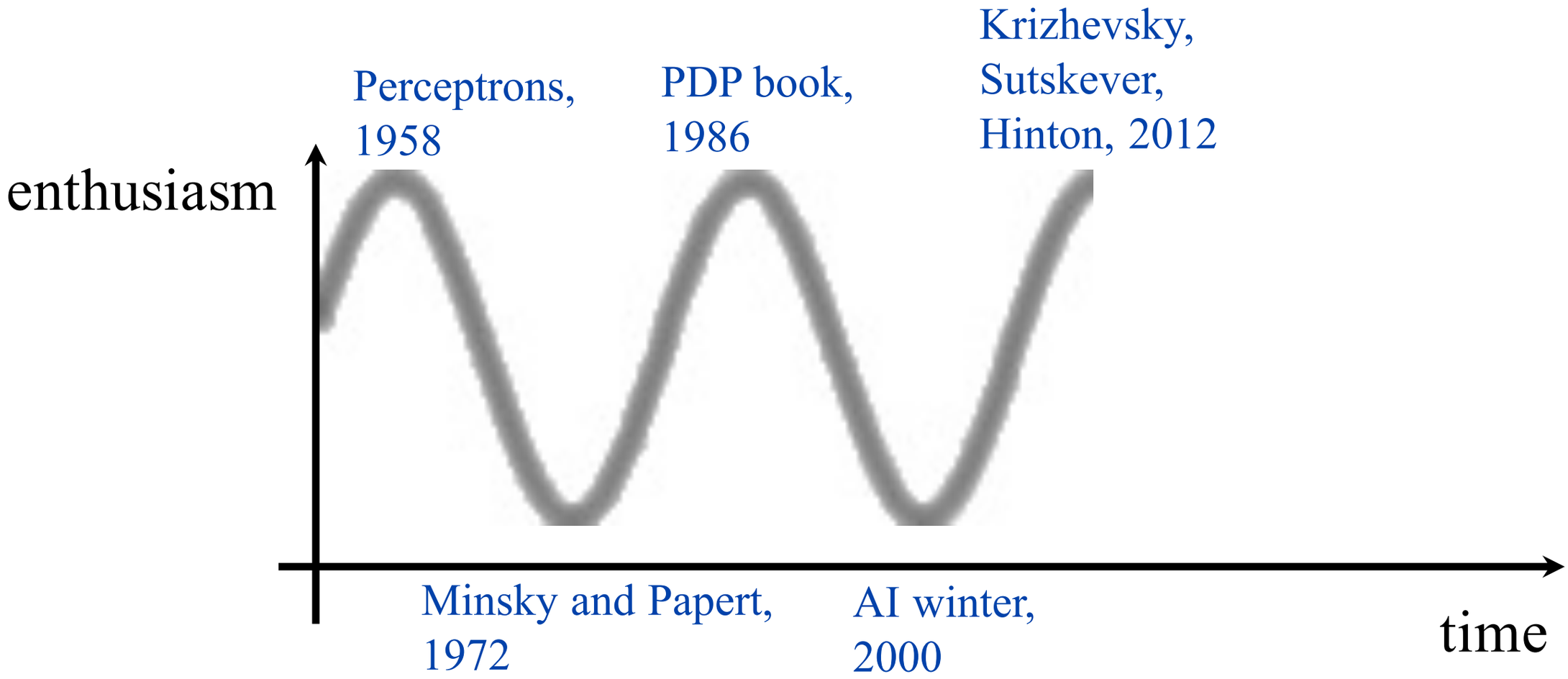
Krizhevsky, Sutskever, and Hinton, NIPS 2012

Slide credit: Antonio Torralba

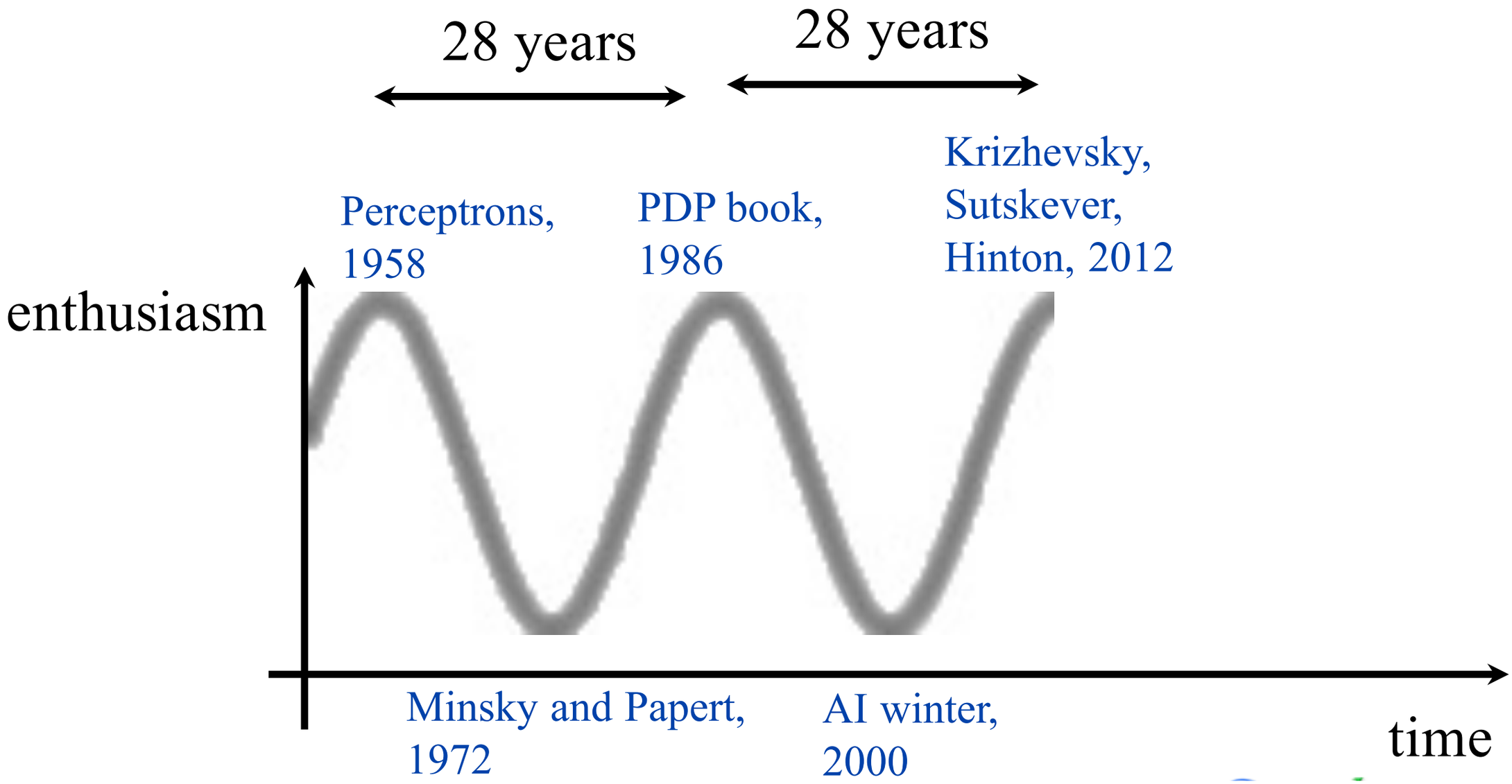
Test Nearby images, according to NN features

Krizhevsky, Sutskever, and Hinton, NIPS 2012

Slide credit: Antonio Torralba

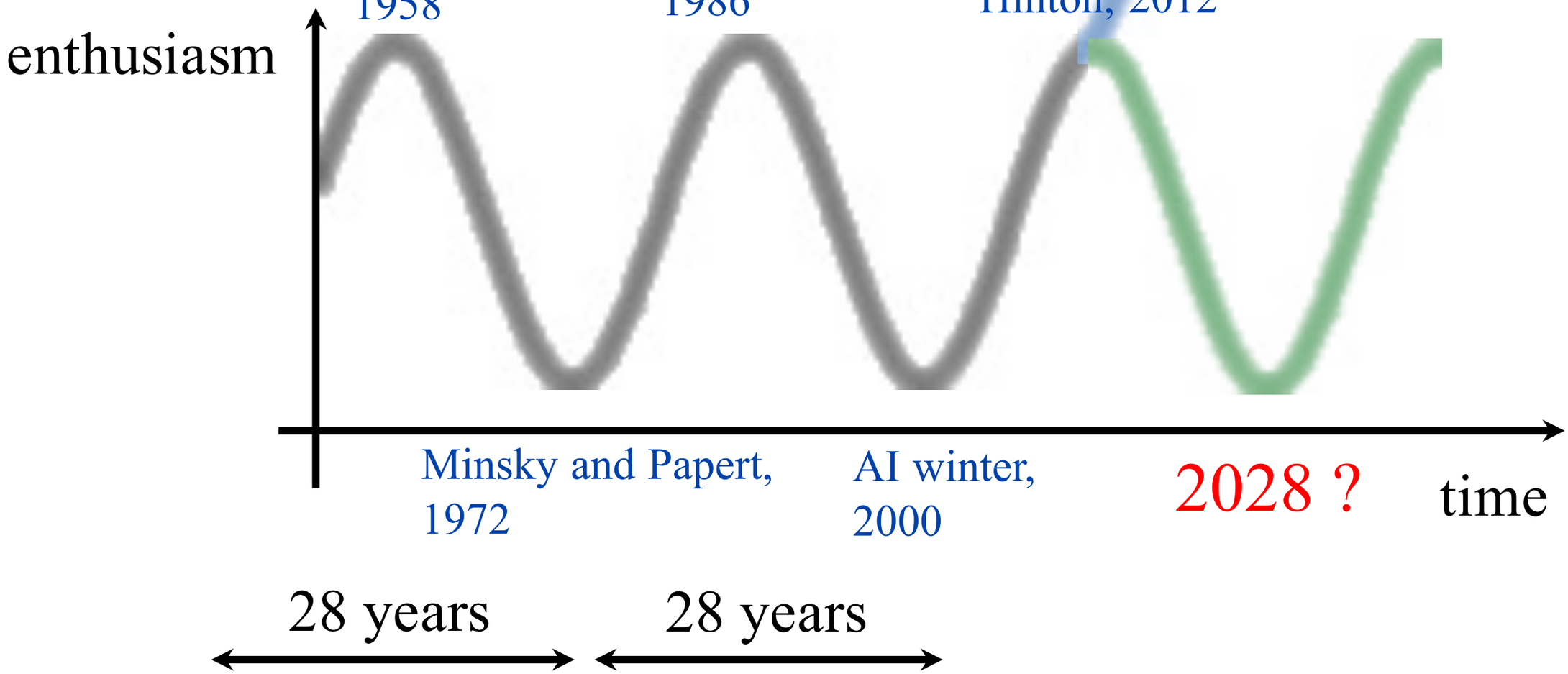


Slide credit: Antonio Torralba

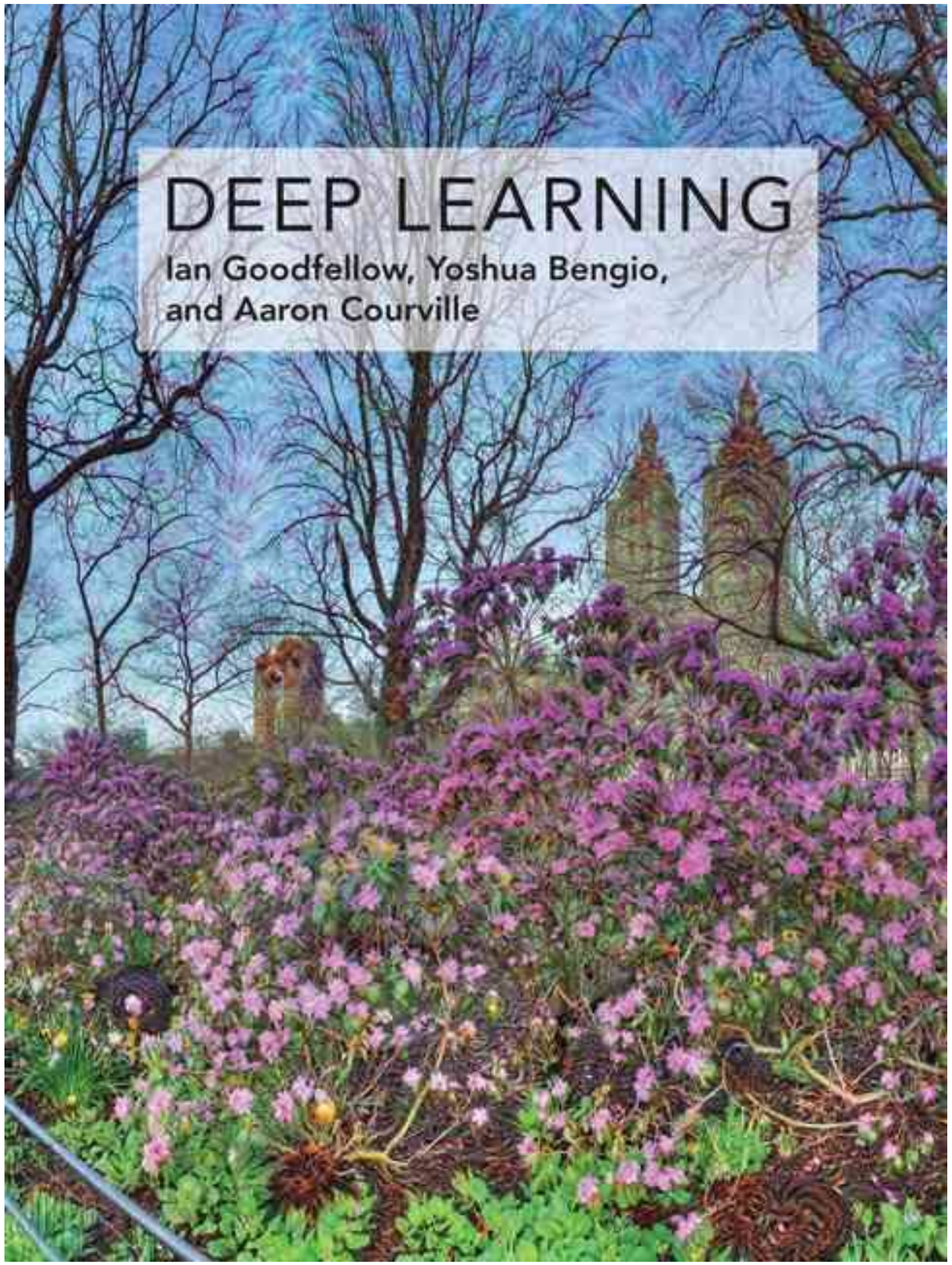


Slide credit: Antonio Torralba

What comes next?



Slide credit: Antonio Torralba



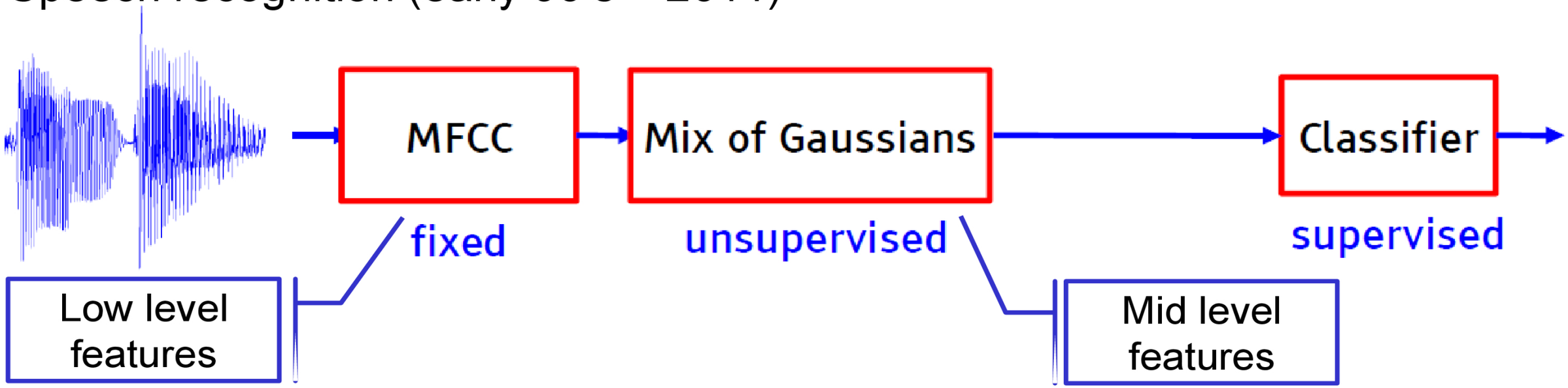
http://www.deeplearningbook.org/

By Ian Goodfellow, Yoshua Bengio and Aaron Courville

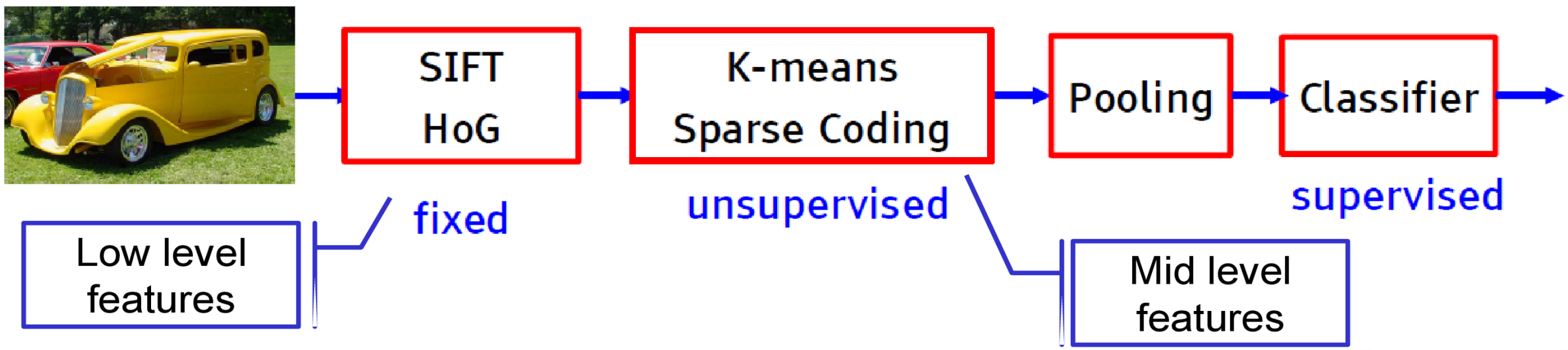
November 2016

“Classical” Recognition

Speech recognition (early 90’s – 2011)



Object recognition (2006 – 2012)

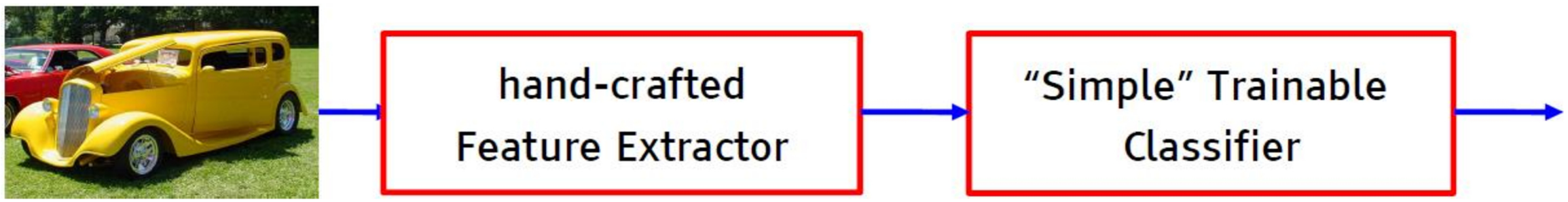


Slide credit: LeCun

End-to-end

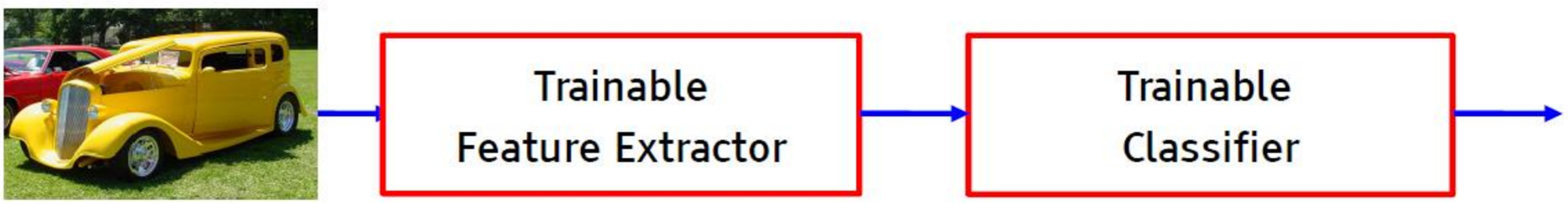
Deep learning can be summarized as learning both the representation and the classifier out of it

- Fixed engineered features (or kernels) + trainable classifier



VS.

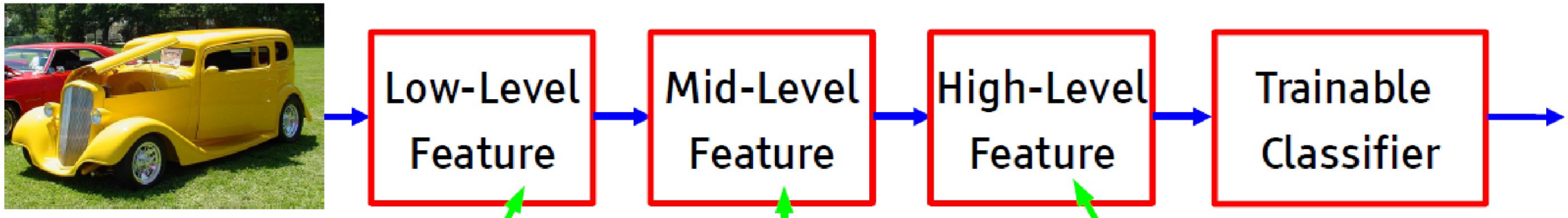
- End-to-end learning / feature learning / deep learning



Slide credit: LeCun

End-to-end

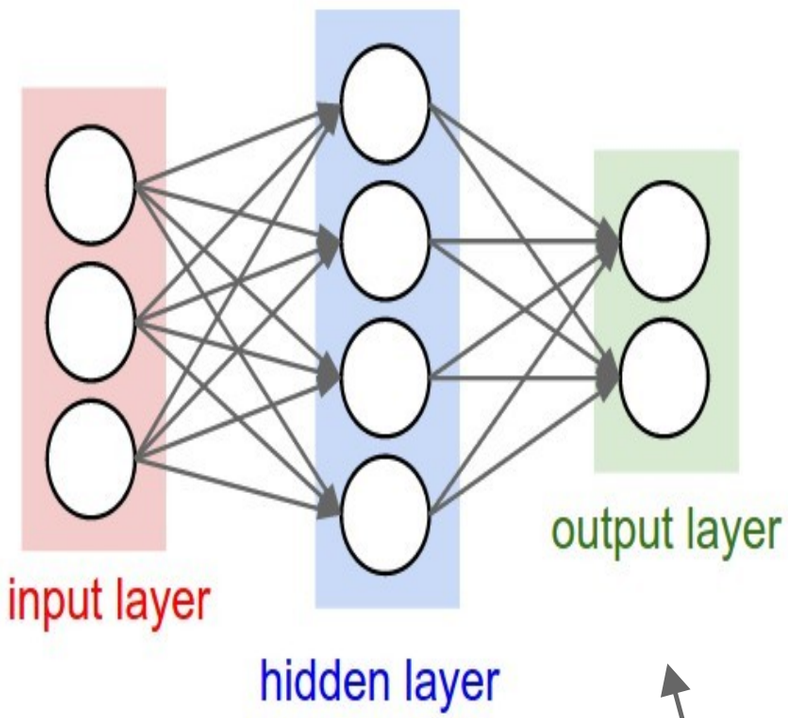
In deep learning we have multiple stages of non linear feature transformation

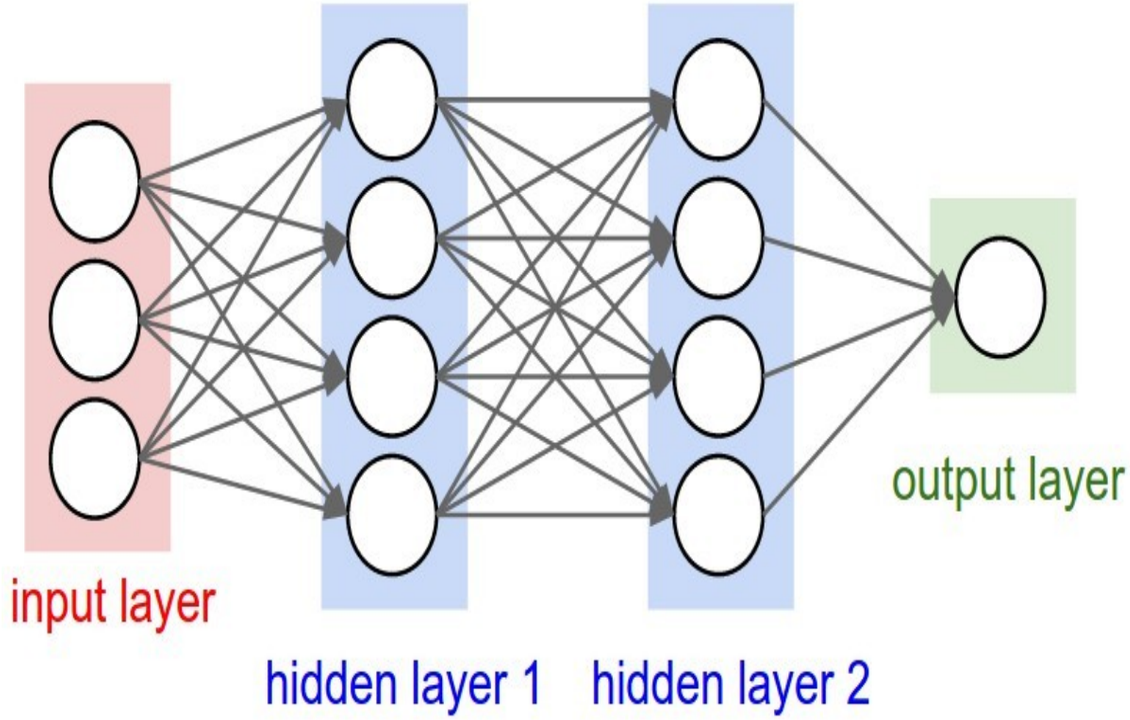


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Slide credit: LeCun

Artificial Neural Networks





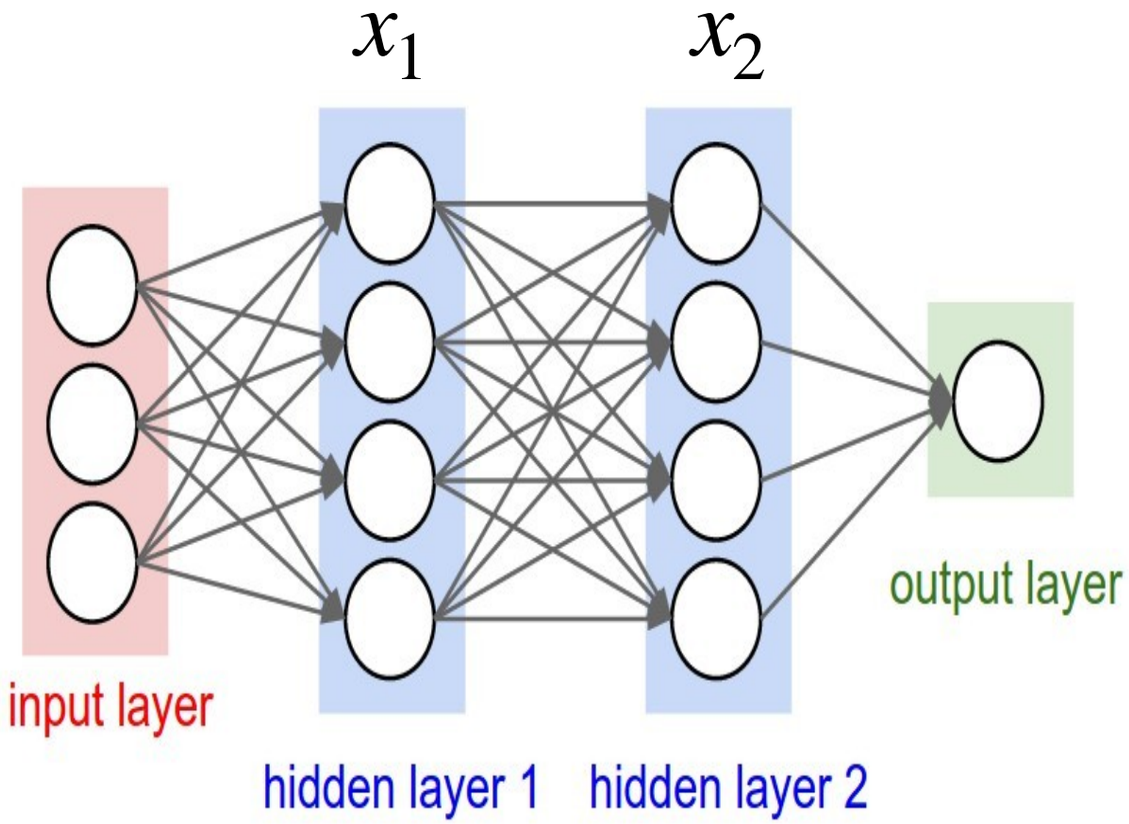
"2-layer Neural Net", or
"1-hidden-layer Neural Net"

"3-layer Neural Net", or
"2-hidden-layer Neural Net"

"Fully-connected" layers

Slide credit: Andrej Karpathy

Artificial Neural Networks



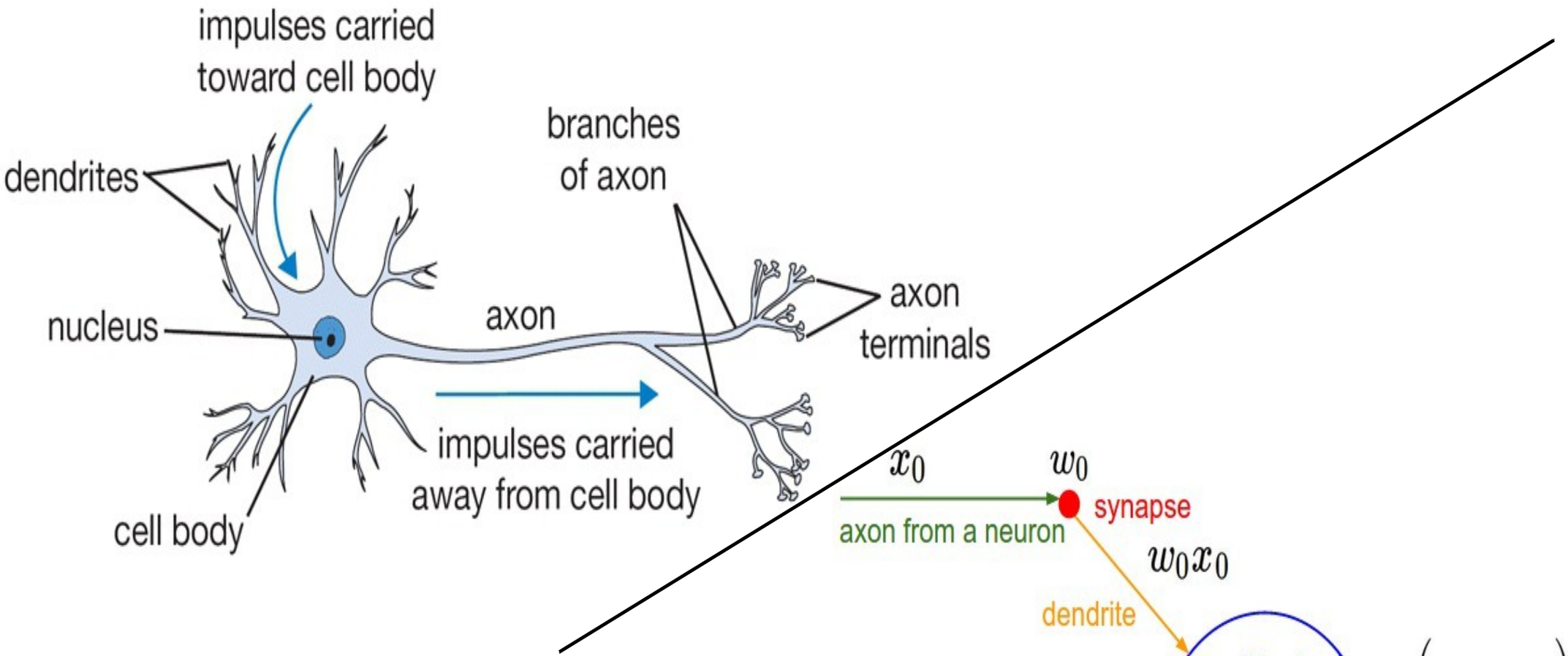
$$x_i \in \mathbb{R}^{H \times 1}$$

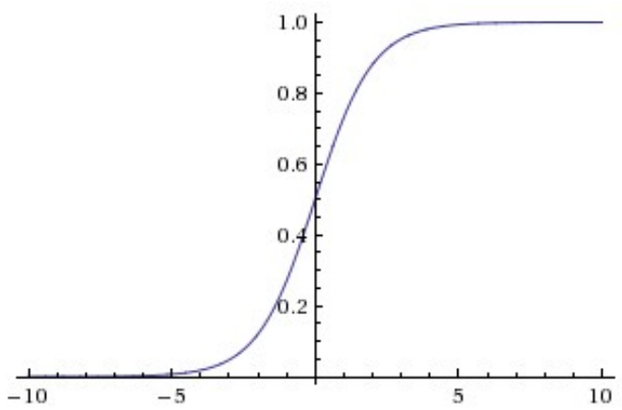
$$W_i \in \mathbb{R}^{D \times H}$$

$$b_i \in \mathbb{R}^{D \times 1}$$

$$x_{i+1} = f\left(W_i x_i + b_i\right)$$

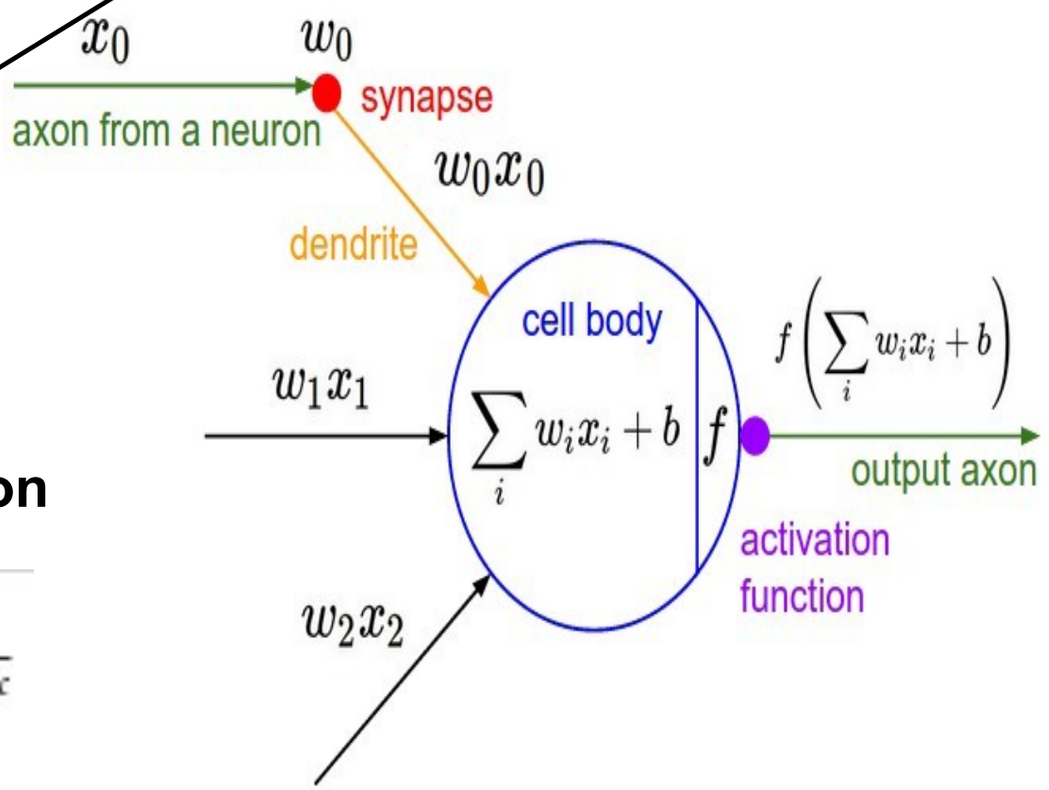
Bio/Artificial Neurons





sigmoid activation function

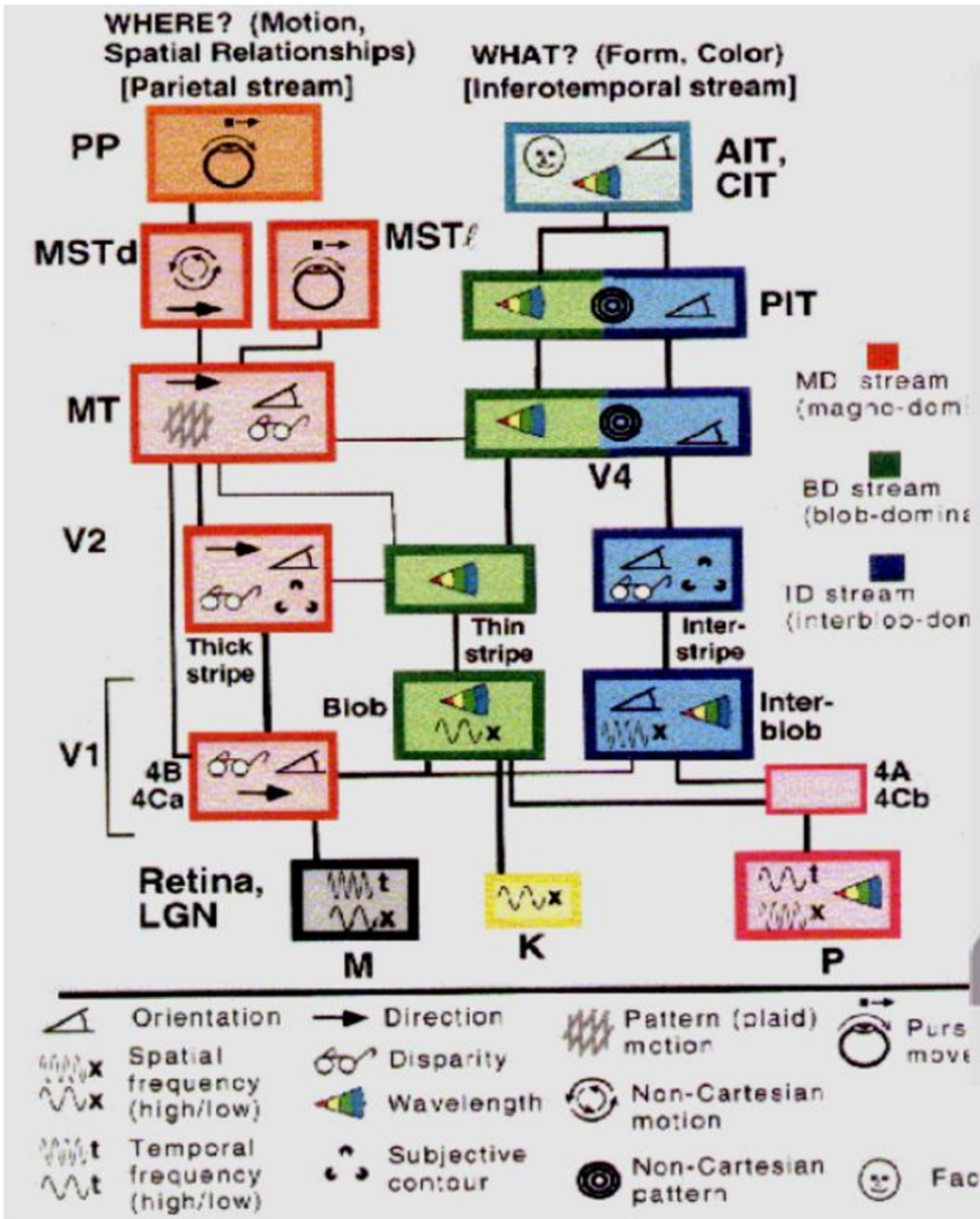
$$\frac{1}{1+e^{-x}}$$

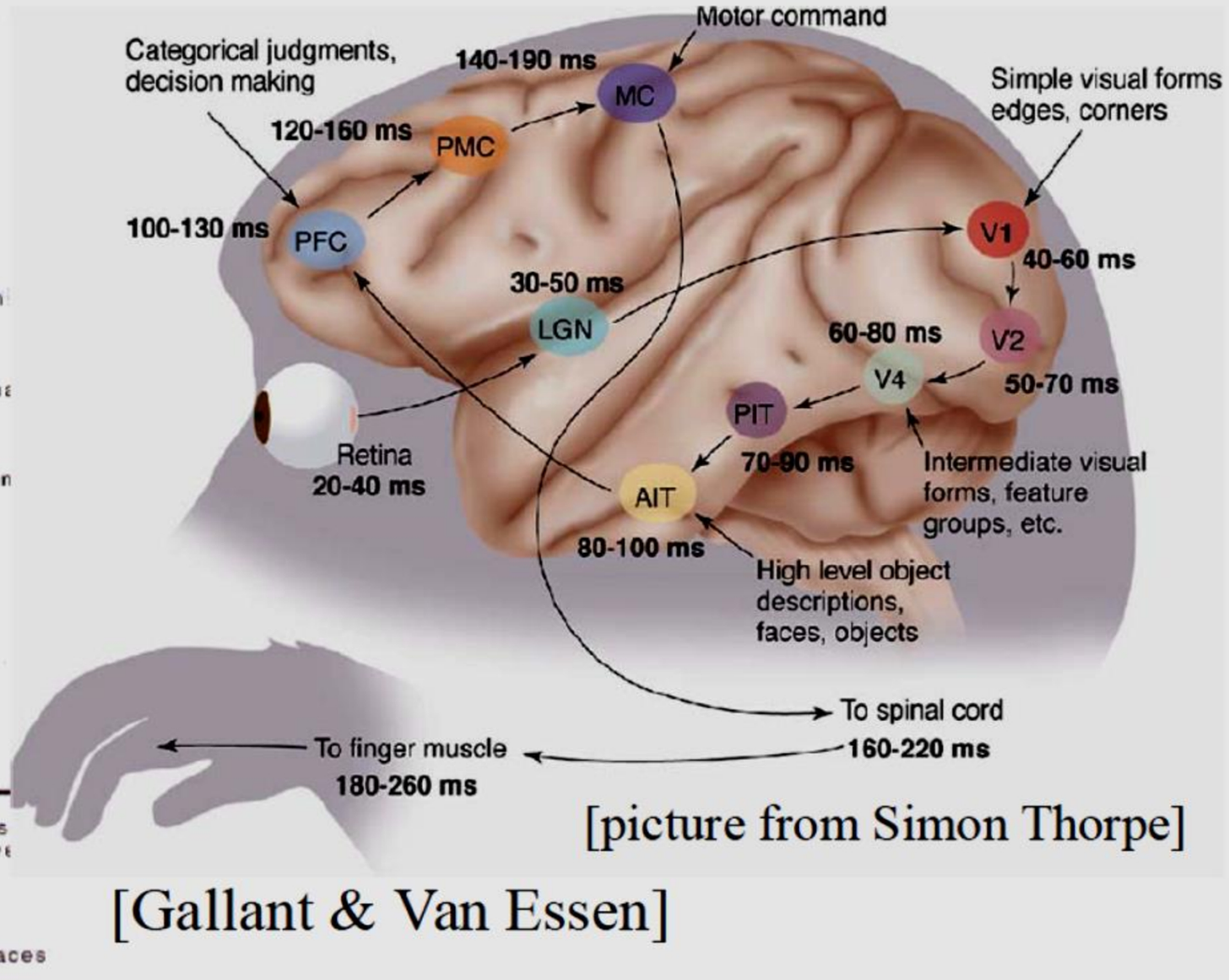


Slide credit: Andrej Karpathy

Bio/Artificial Networks

The ventral (recognition) pathway in the visual cortex has multiple stages



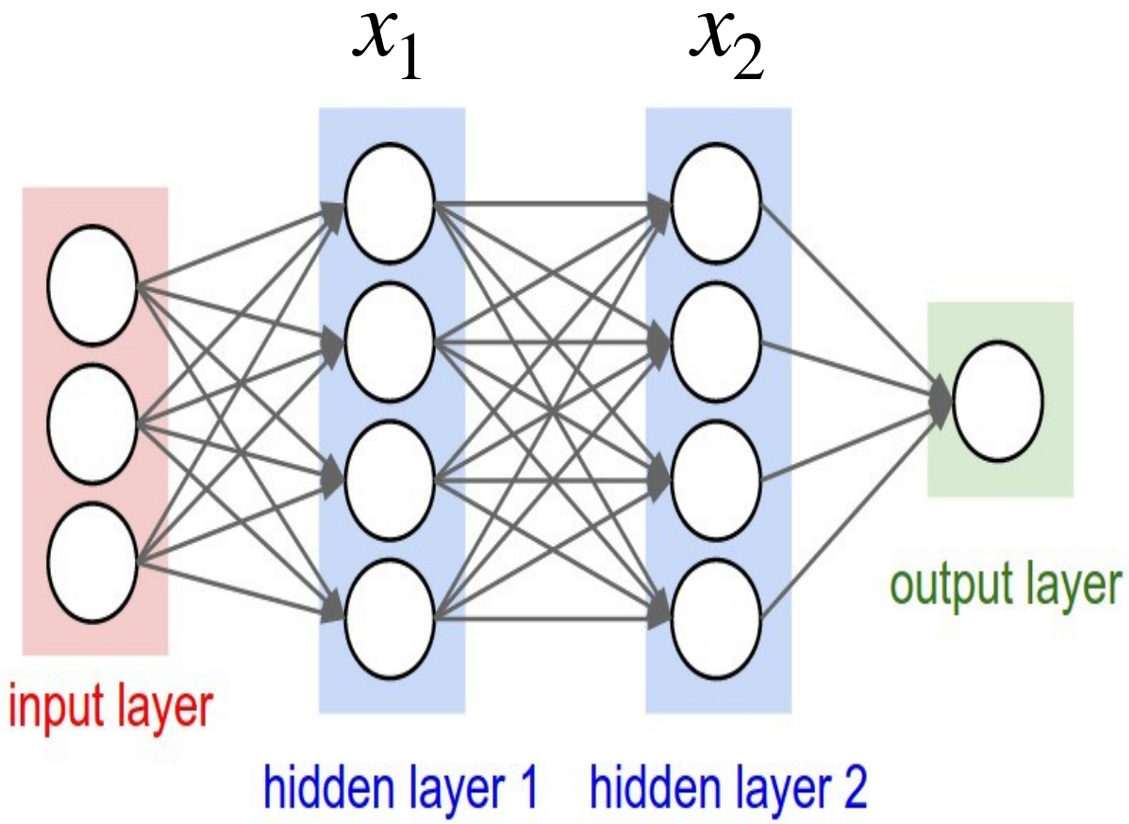


[picture from Simon Thorpe]

[Gallant & Van Essen]

Slide credit: LeCun

Artificial Neural Networks



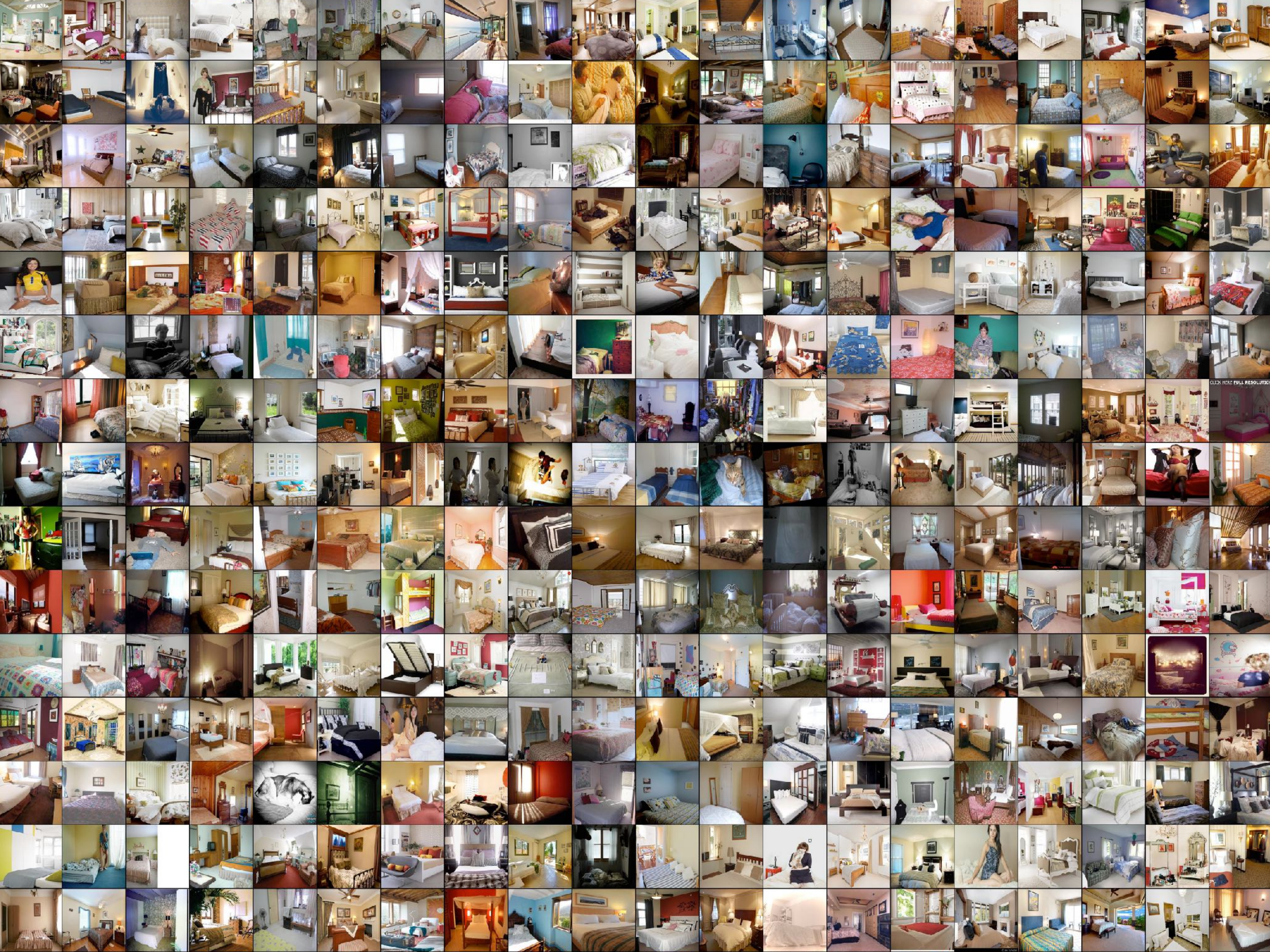
$$x_i \in \mathbb{R}^{H \times 1}$$

$$W_i \in \mathbb{R}^{D \times H}$$

$$b_i \in \mathbb{R}^{D \times 1}$$

$$x_{i+1} = f\left(W_i x_i + b_i\right)$$

How do we find W and b?


CLICK HERE FULL RESOLUTION

Park

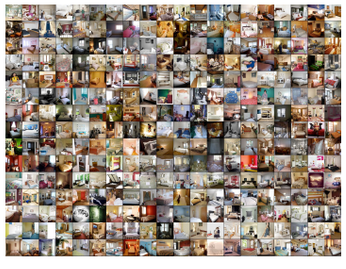
Kitchens

Beaches

Forests

Theatres

Playroom



Bedrooms

Pools

Loss Functions

x_i Input (image)

y_i Target (labels)

$\mathscr{L}$ Loss Function

θ Parameters

$f(x_i; \theta)$ Prediction

The objective of learning:

$$\min_{\theta} \sum_{i} \mathscr{L}\left(f(x_i; \theta), y_i\right)$$

Common Loss Functions

Squared error:

$$\mathscr{L}(x, y) = \|x - y\|_2^2$$

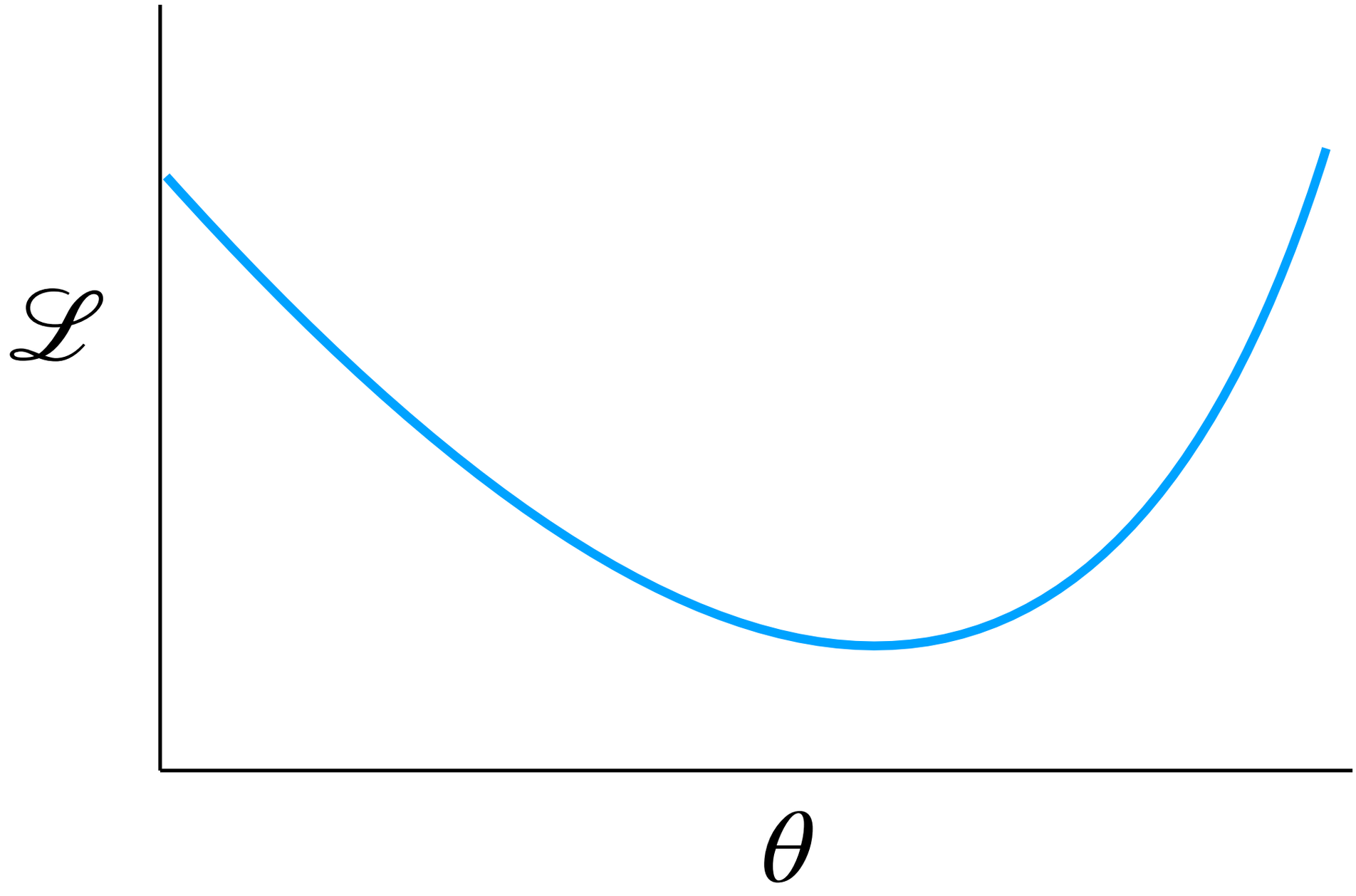
Hinge loss:

$$\mathscr{L}(x, y) = \max(0, 1 - x \cdot y)$$

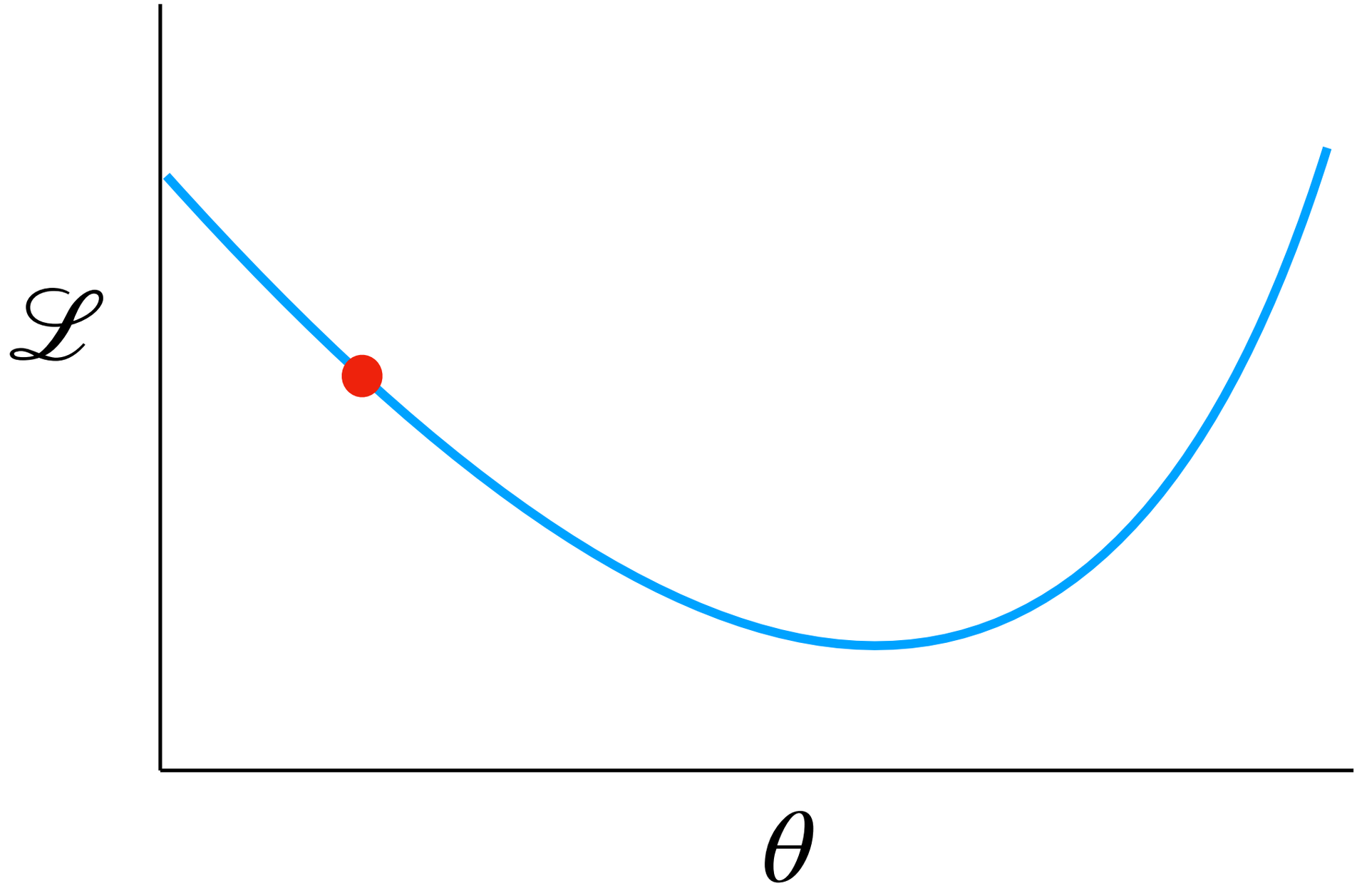
Cross entropy:

$$\mathscr{L}(x, y) = -\sum_i y_i \log x_i$$

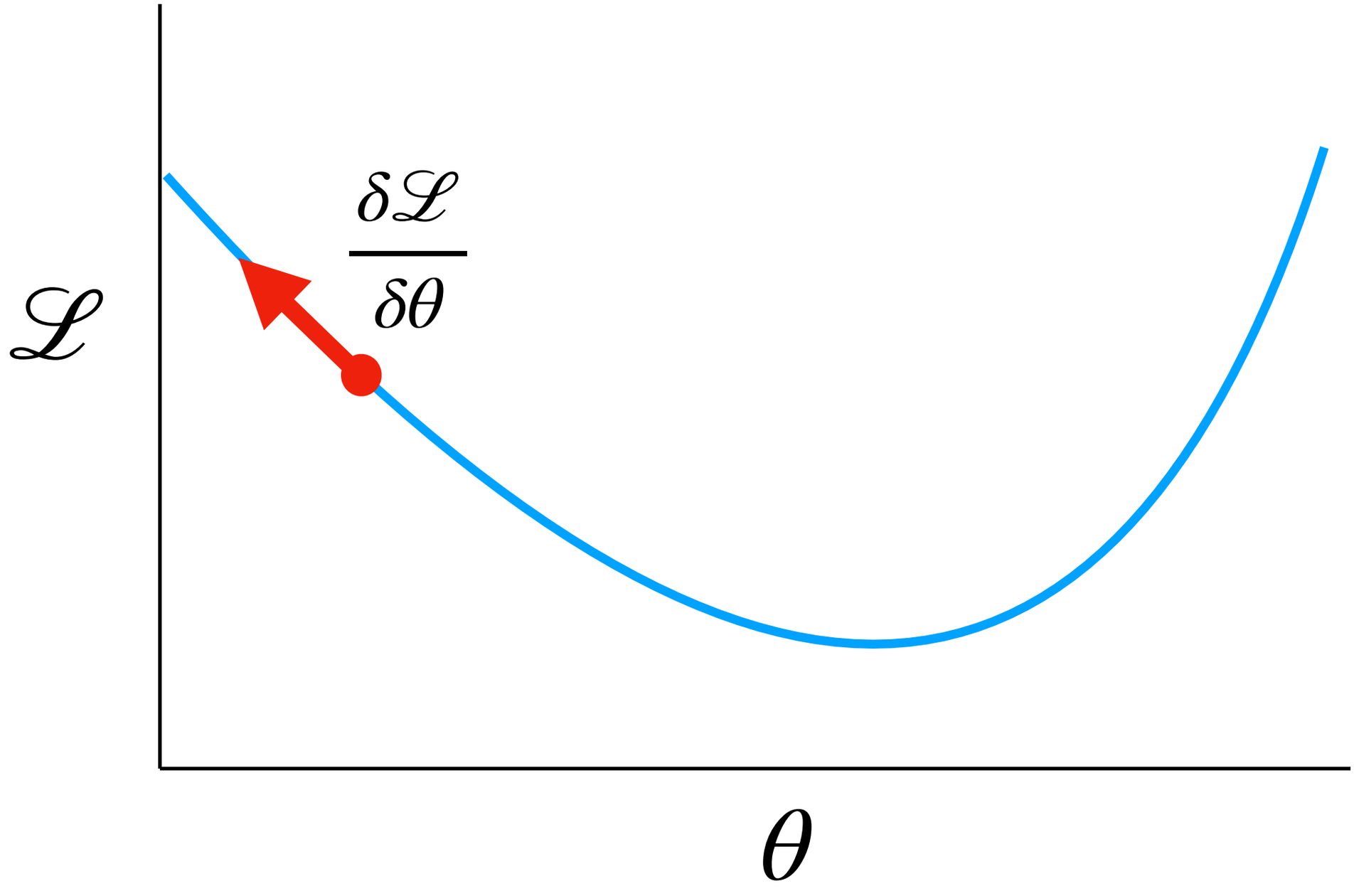
Loss Surface



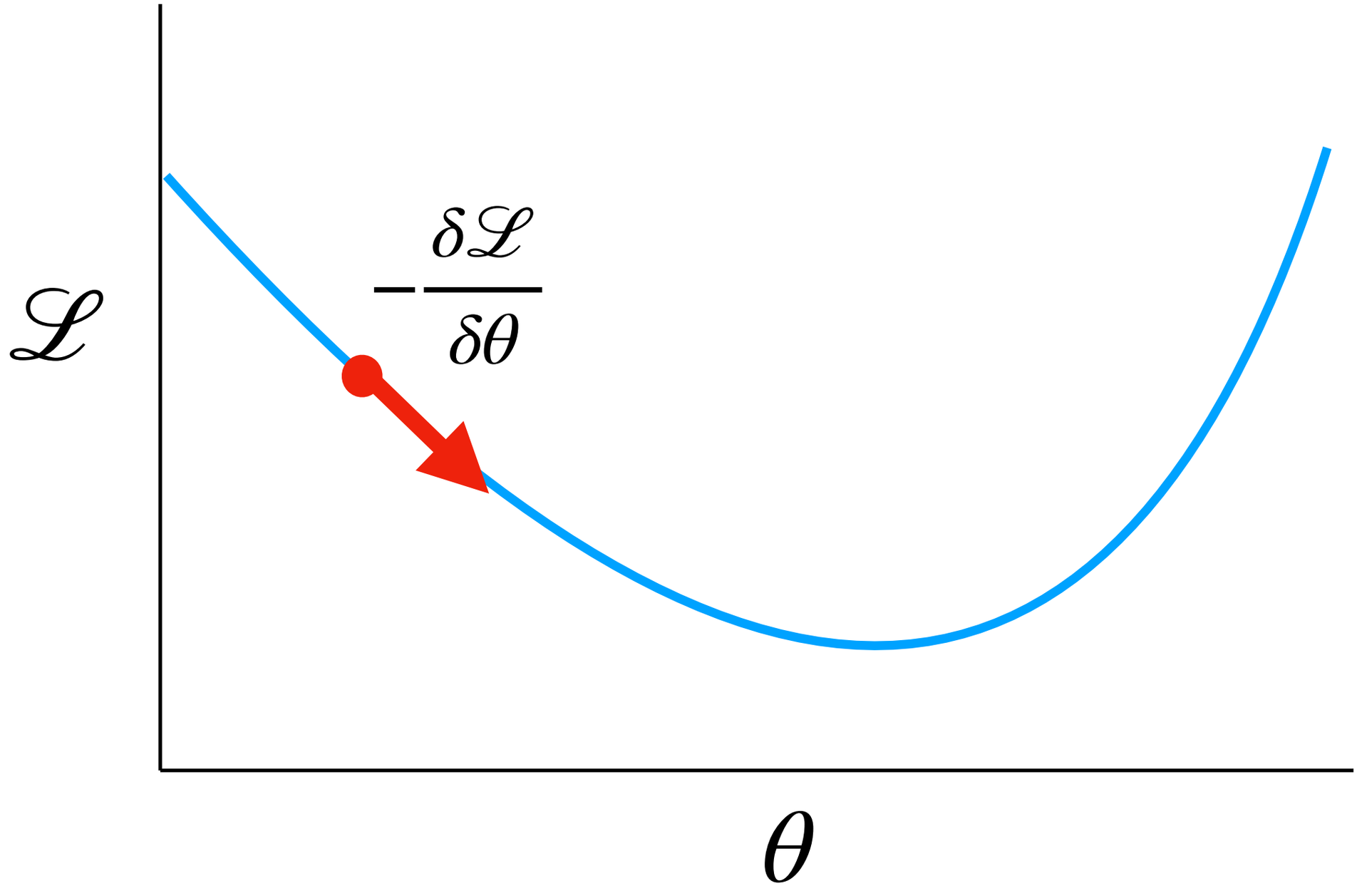
Loss Surface



Loss Surface

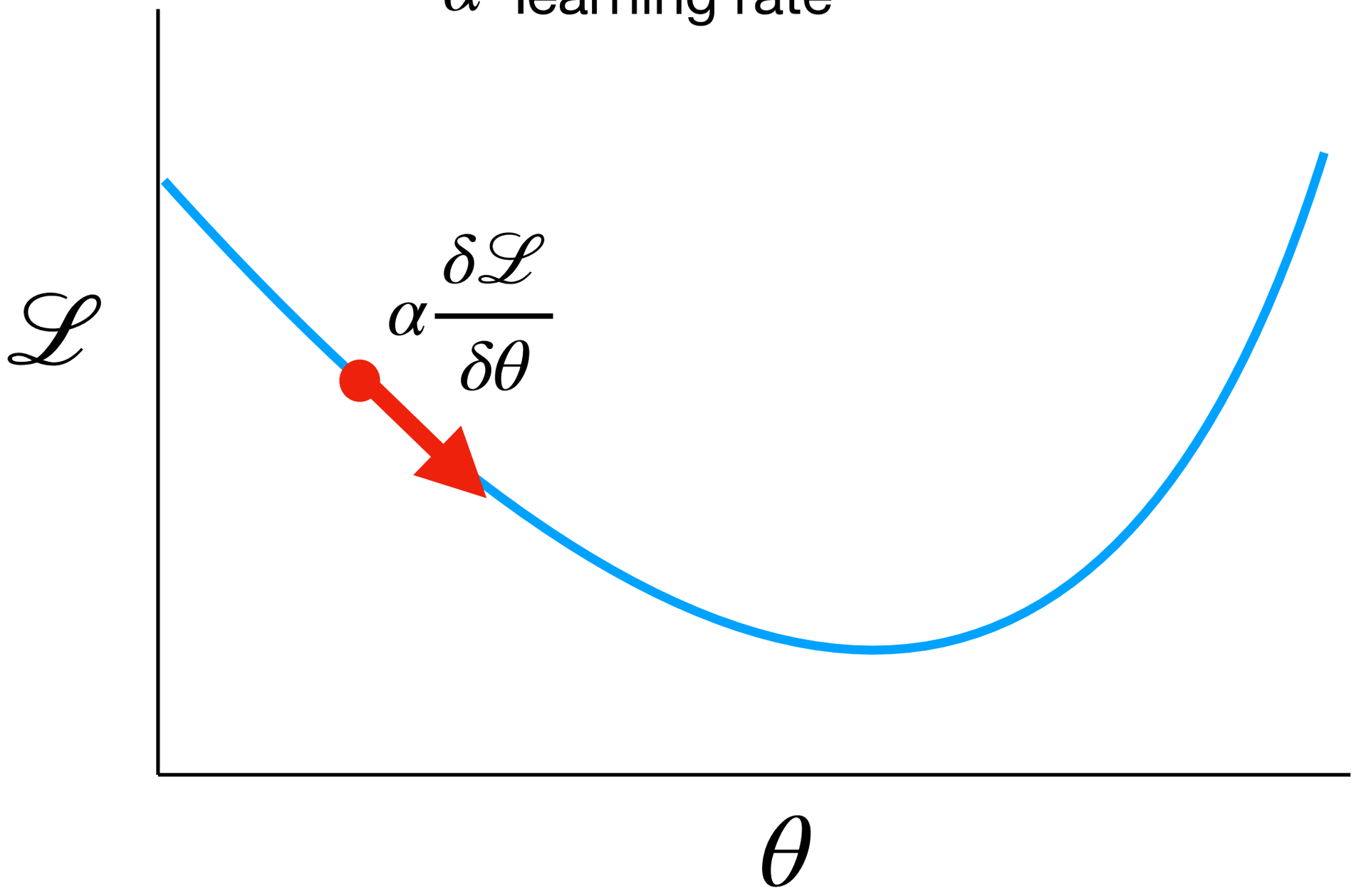


Gradient Descent

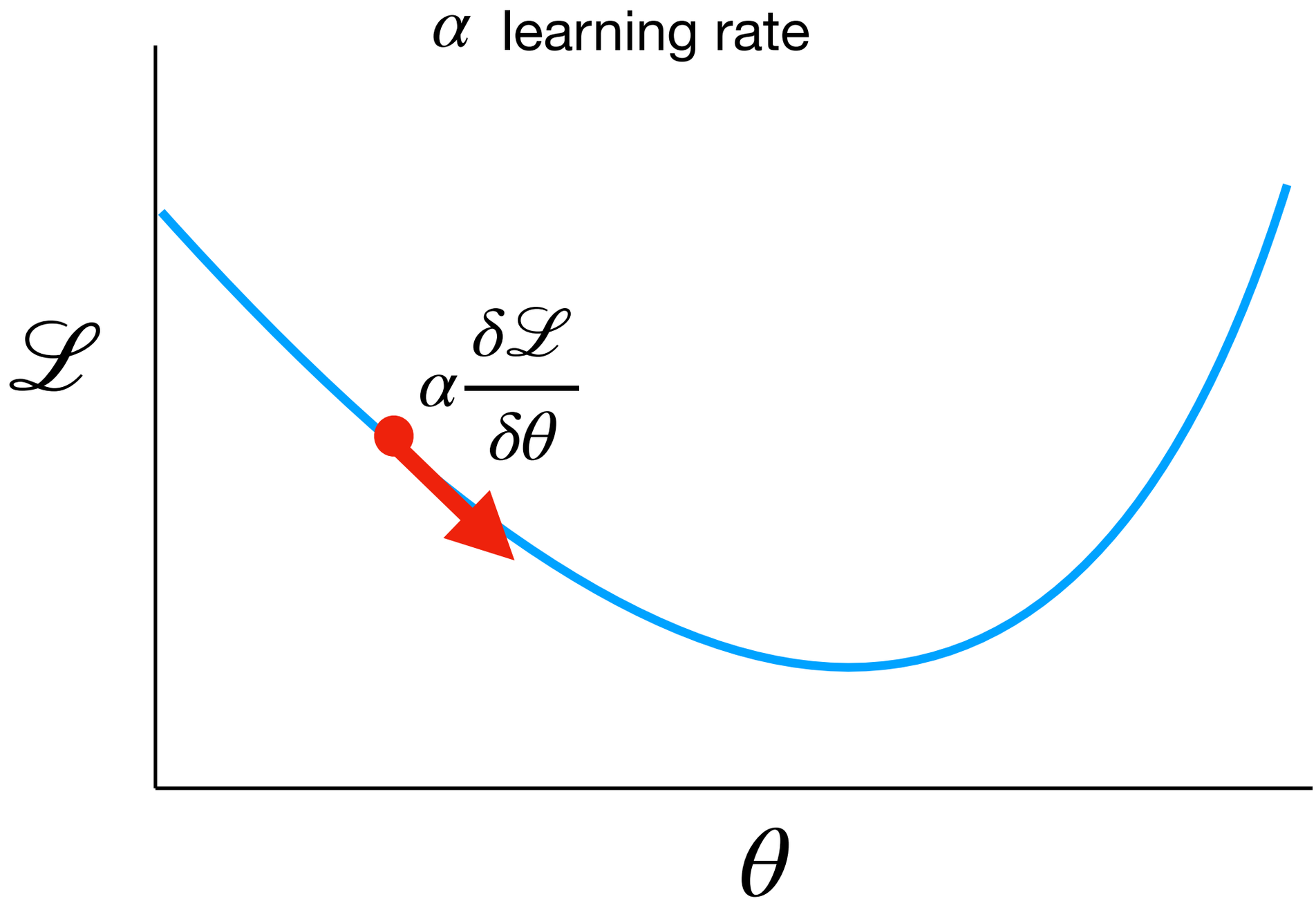


Gradient Descent

α learning rate

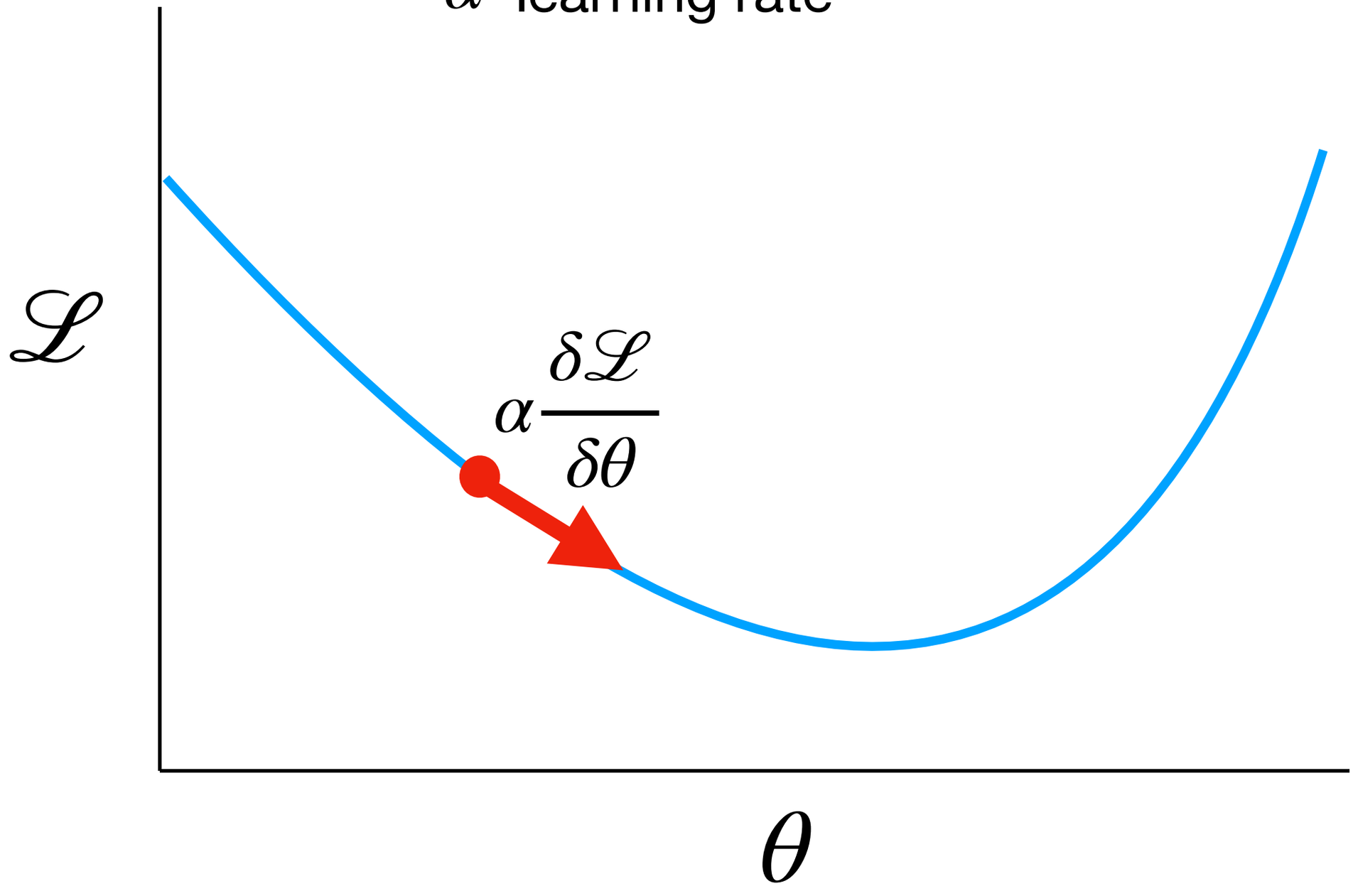


Gradient Descent



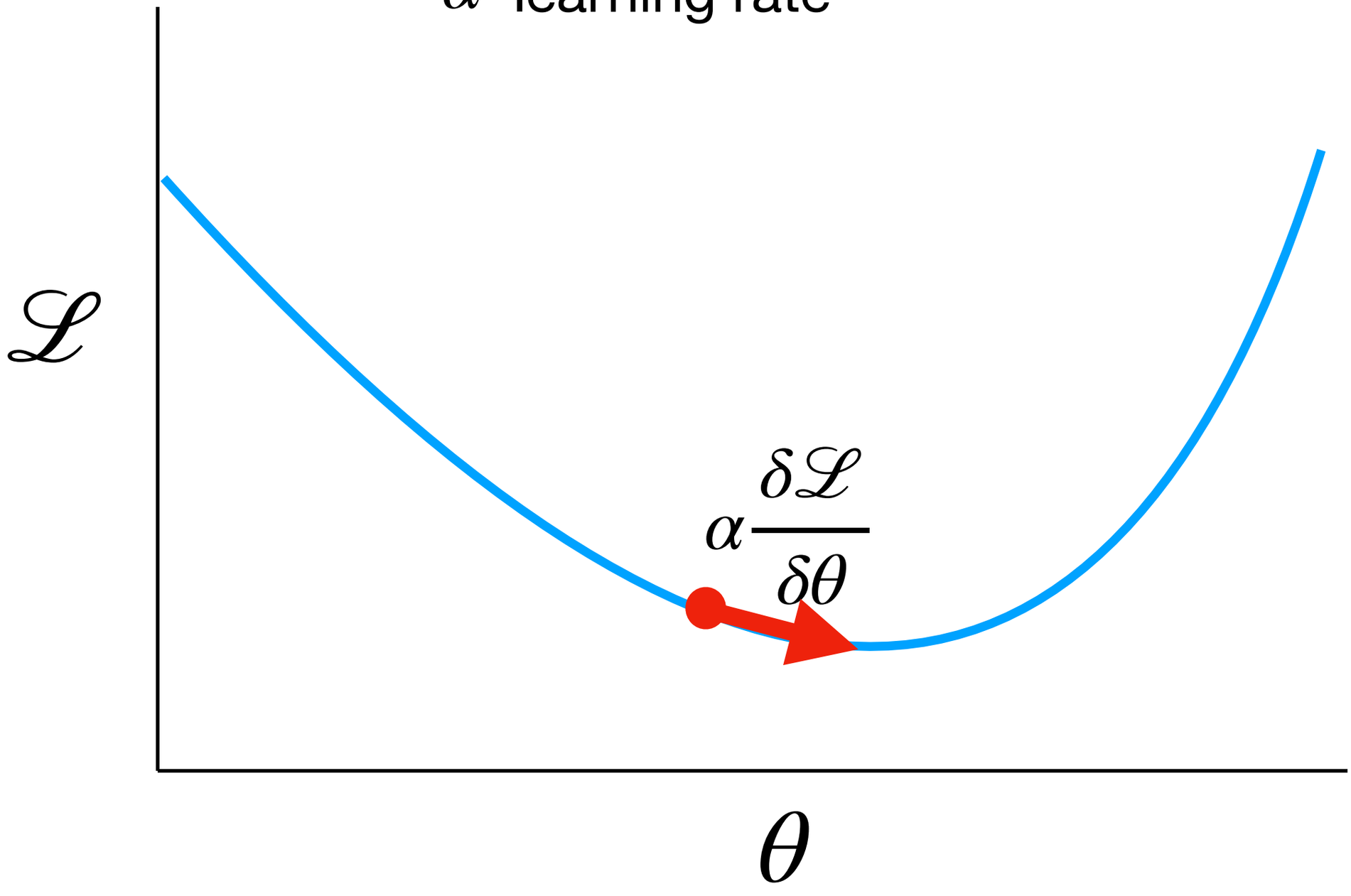
Gradient Descent

α learning rate



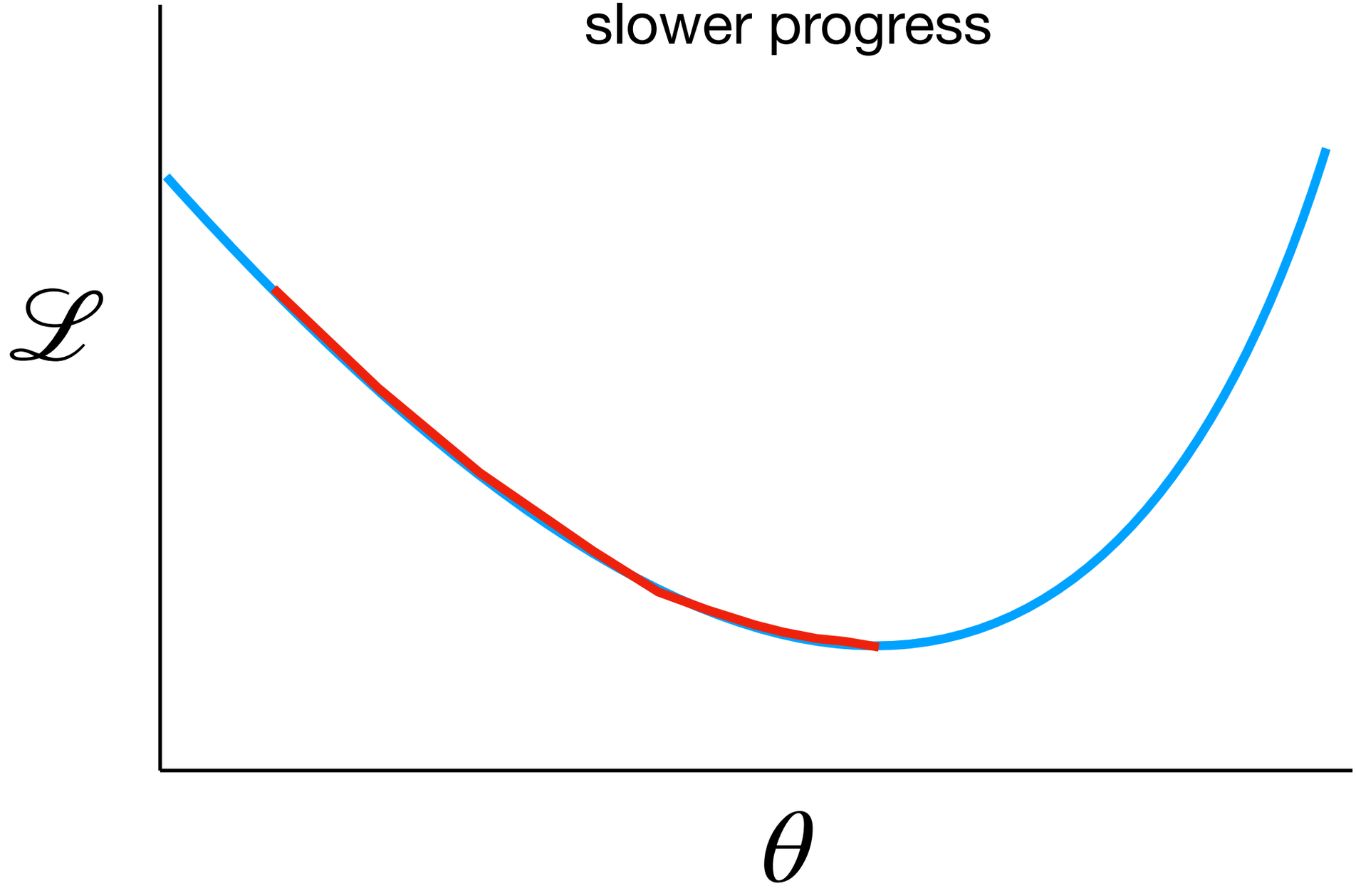
Gradient Descent

α learning rate



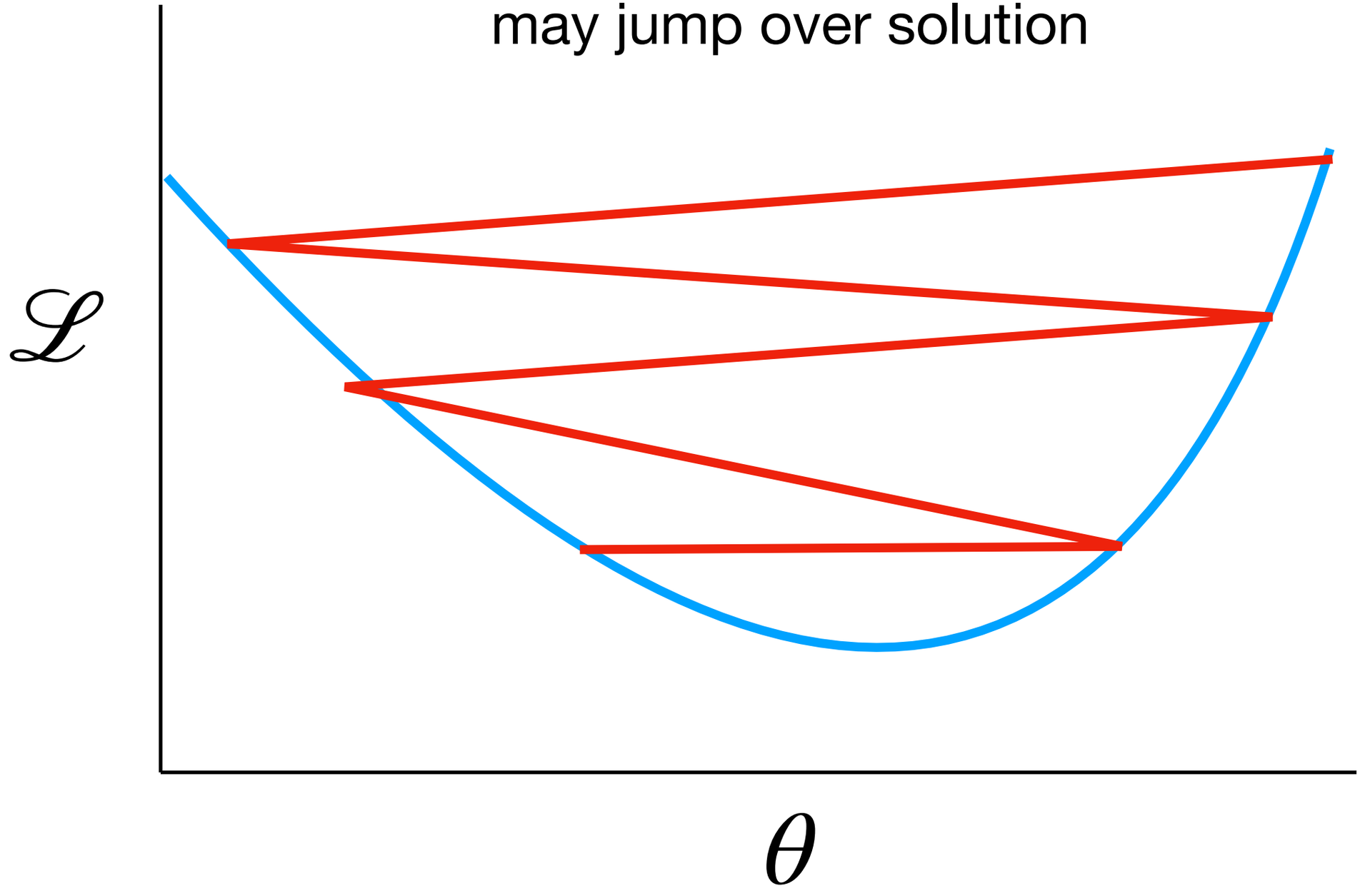
Gradient Descent

Small learning rate:
slower progress

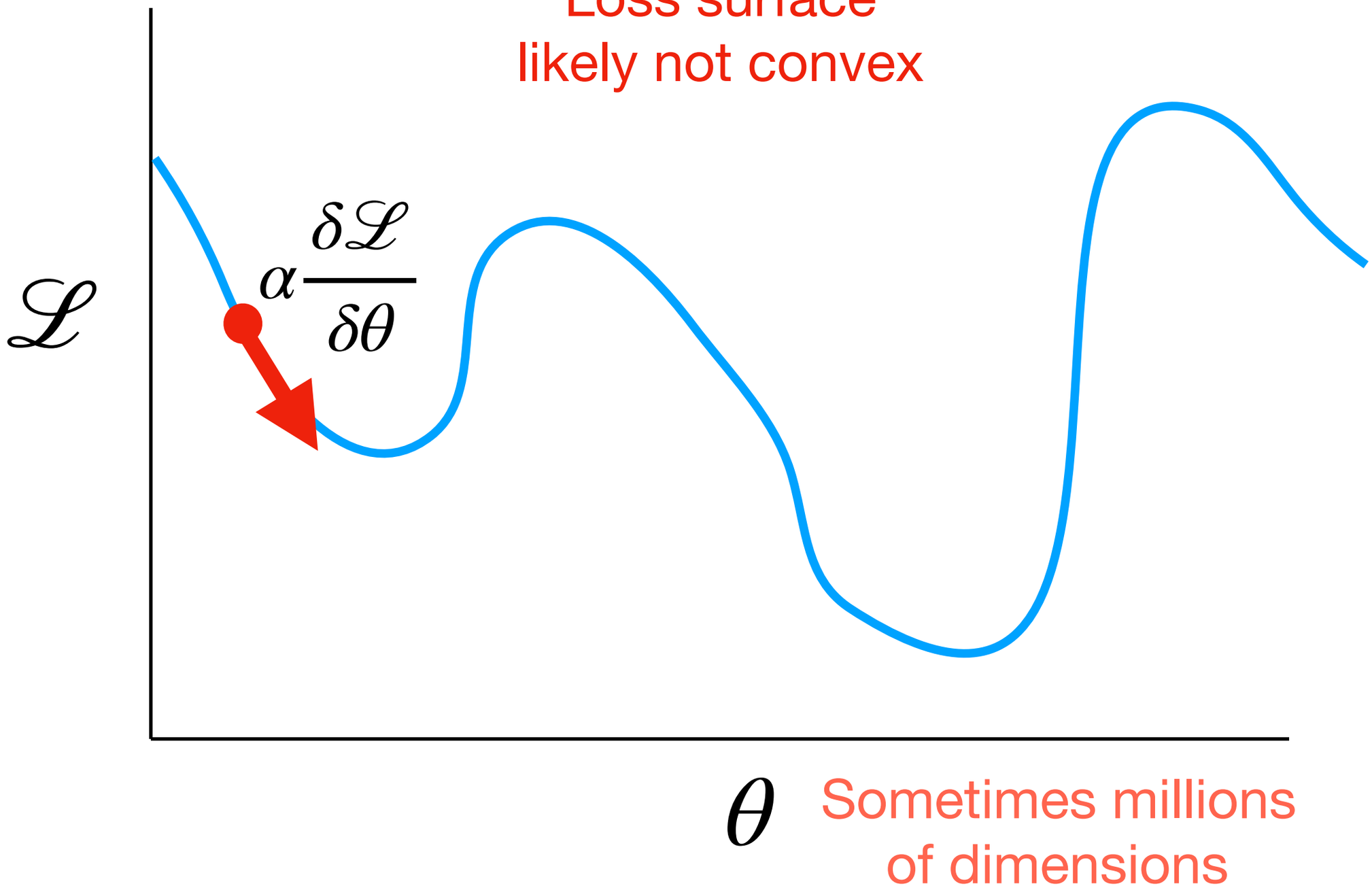


Gradient Descent

Large learning rate:
may jump over solution



Gradient Descent



Flavors of Gradient Descent

Gradient descent:

$$\theta_{t+1} = \theta_t + \alpha \frac{\delta \mathscr{L}}{\delta \theta_t}$$

Gradient descent with momentum:

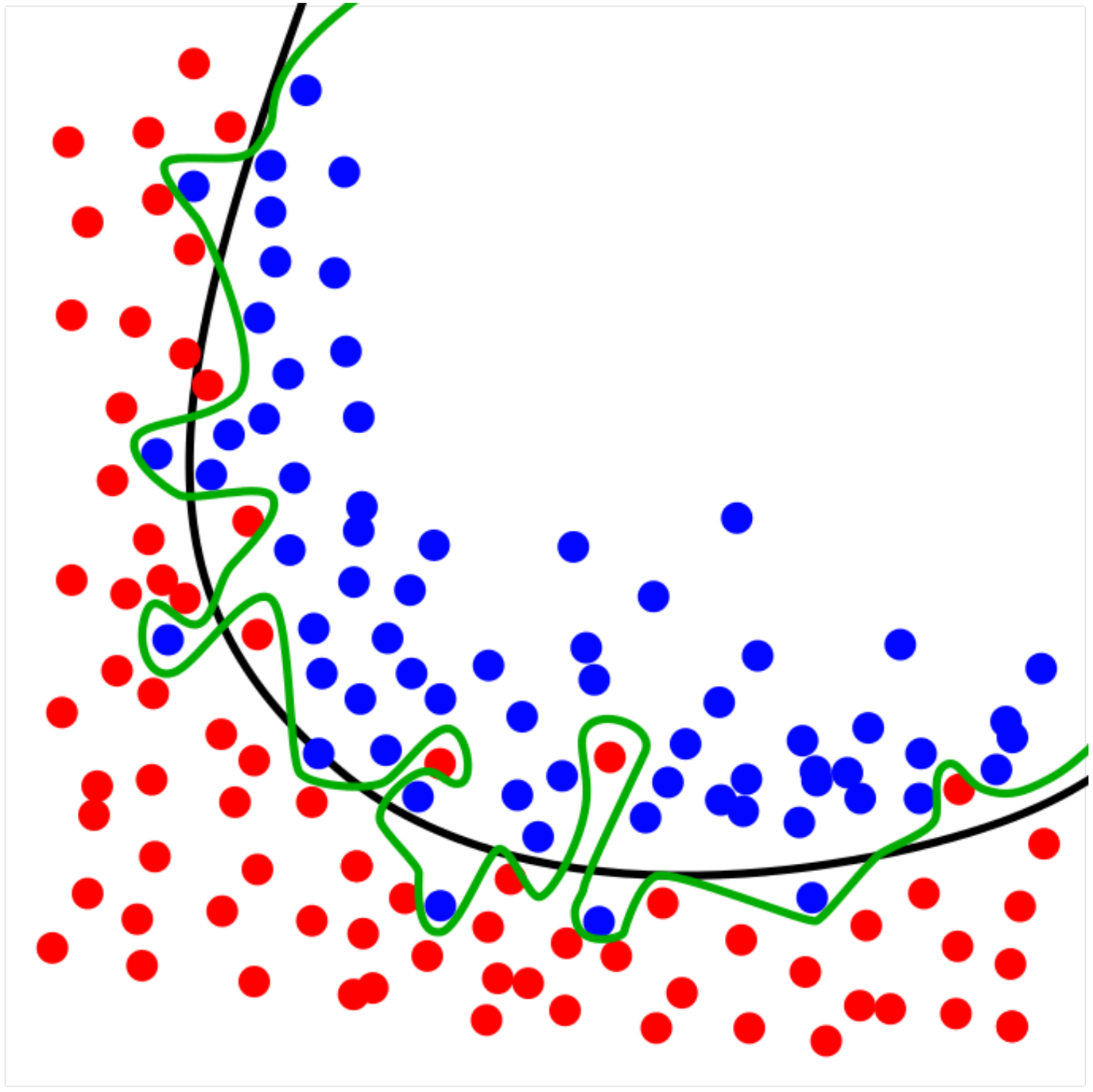
$$z_{t+1} = \beta z_t + \frac{\delta \mathscr{L}}{\delta \theta_t}$$

$$\theta_{t+1} = \theta_t + \alpha z_{t+1}$$

Stochastic gradient descent:

$$\theta_{t+1} = \theta_t + \alpha \mathbb{E}_x \left[\frac{\delta \mathscr{L}}{\delta \theta_t} \right]$$

Are we done?



Wikipedia

Regularization

x_i Input (image)

y_i Target (labels)

$\mathscr{L}$ Loss Function

θ Parameters

$f(x_i; \theta)$ Prediction

$\mathscr{R}$ Regularization

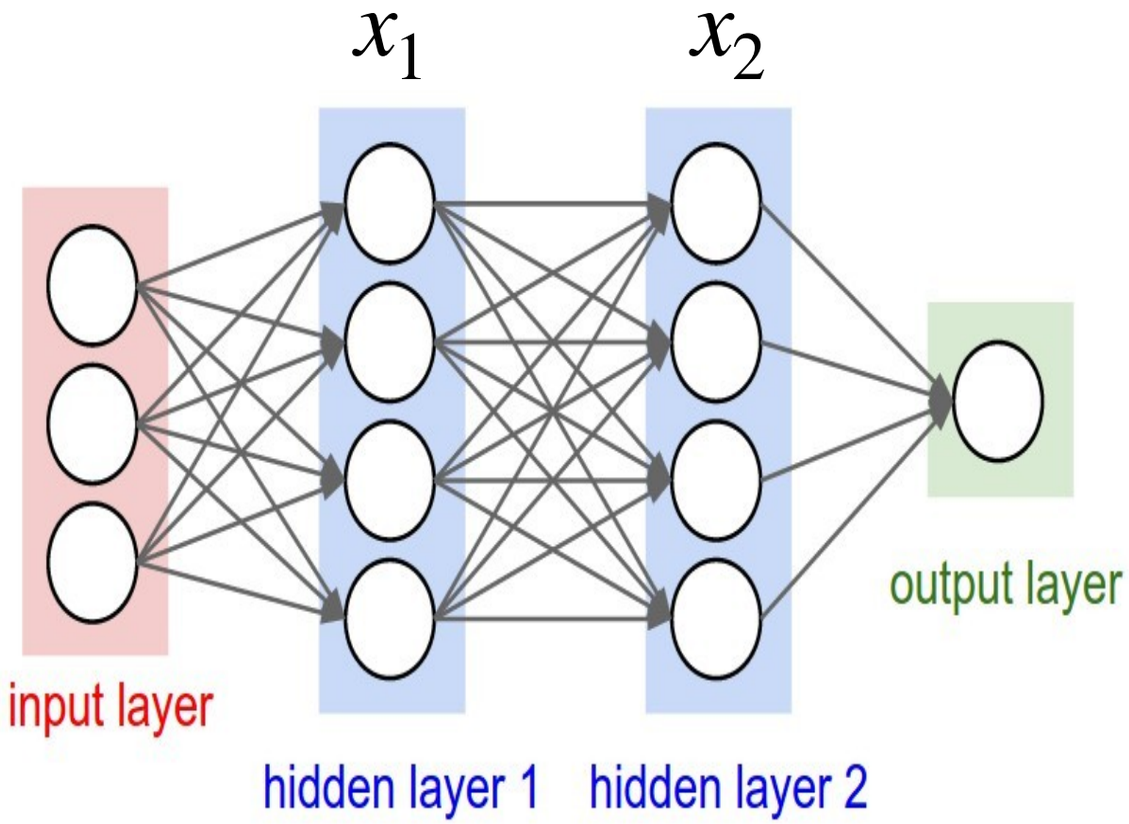
The objective of learning:

$$\min_{\theta} \sum_{i} \mathscr{L}\left(f(x_i), y_i\right) + \lambda R(\theta)$$

Common regularization:

$$R(\theta) = \|\theta\|_2^2$$

Artificial Neural Networks



$$x_i \in \mathbb{R}^{H \times 1}$$

$$W_i \in \mathbb{R}^{D \times H}$$

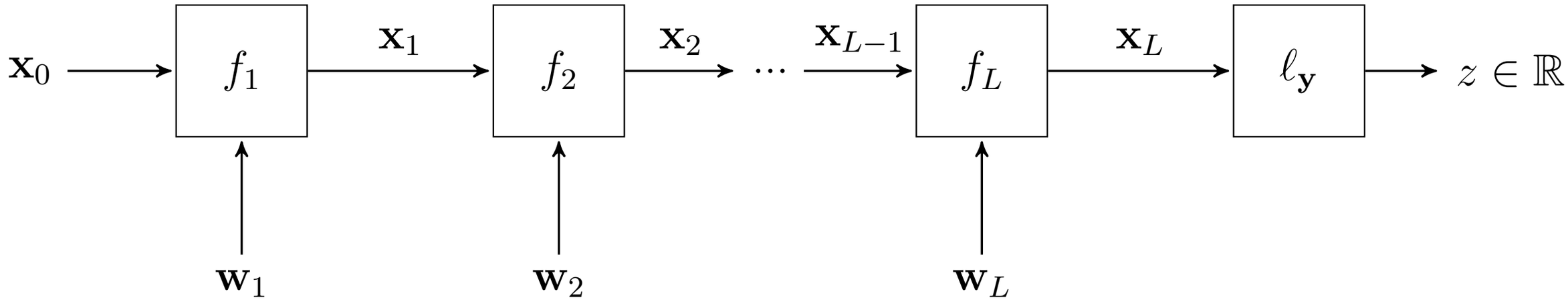
$$b_i \in \mathbb{R}^{D \times 1}$$

$$x_{i+1} = f\left(W_i x_i + b_i\right)$$

How do we compute gradients?

$$\min_{\theta} \sum_{j} \mathscr{L}\left(x_{last}^{j}, y^{j}\right)$$

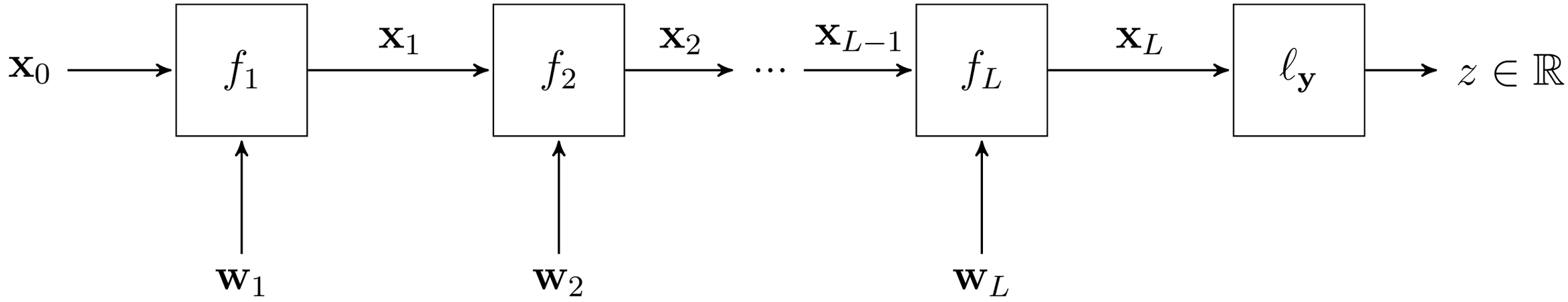
Where we are headed



- Let $\mathbf{x_L}$ be the output of the L^{th} layer.
- We then apply a loss given a target label **y**
- Ultimate goal: compute **dz/dw**

Slide credit: Deva Ramanan

Composable units



$$g(\mathbf{x}_0, \mathbf{w}_1, \ldots) = f_L(f_{L-1}(f_{L-2}(\ldots, \mathbf{w}_{L-2}), \mathbf{w}_{L-1}), \mathbf{w}_L)$$
$$= f_L(\cdot, \mathbf{w}_L) \circ f_{L-1}(\cdot, \mathbf{w}_{L-1}) \ldots$$

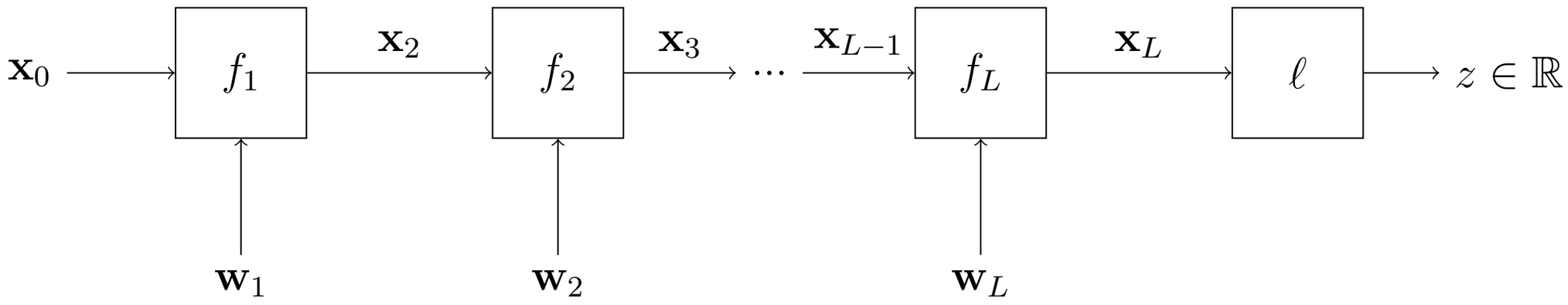
[Shorthand notation for *composition* that we'll use in next slides]

Slide credit: Deva Ramanan

Review: Chain Rule

$$\frac{\delta}{\delta x}\left[f \circ g(x)\right] = f'\left(g(x)\right) g'(x)$$

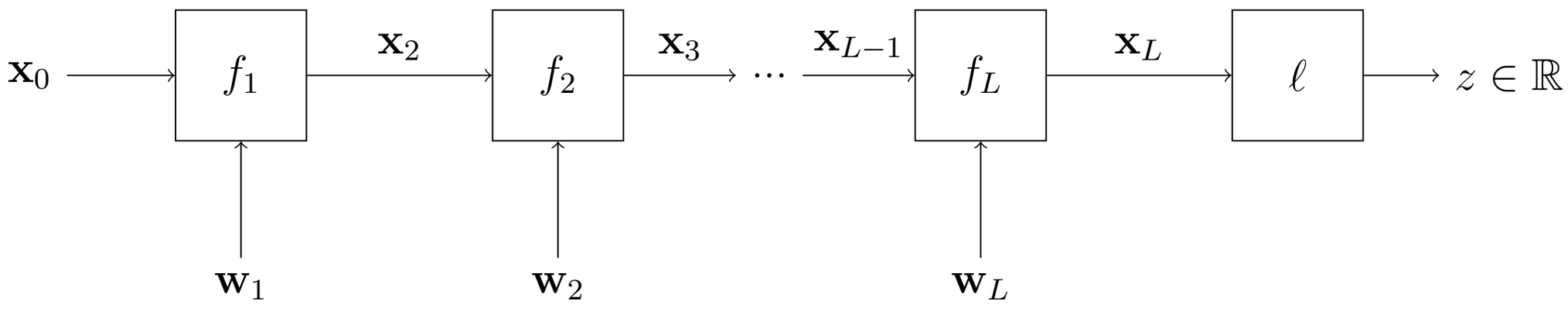
Backprop



$$\frac{dz}{d\mathbf{w}_l} = \frac{d}{d\mathbf{w}_l}\left[\ell_{\mathbf{y}} \circ f_L(\cdot; \mathbf{w}_L) \circ ... \circ f_2(\cdot; \mathbf{w}_2) \circ f_1(\mathbf{x}_0; \mathbf{w}_1)\right]$$

Slide credit: Deva Ramanan

Backprop

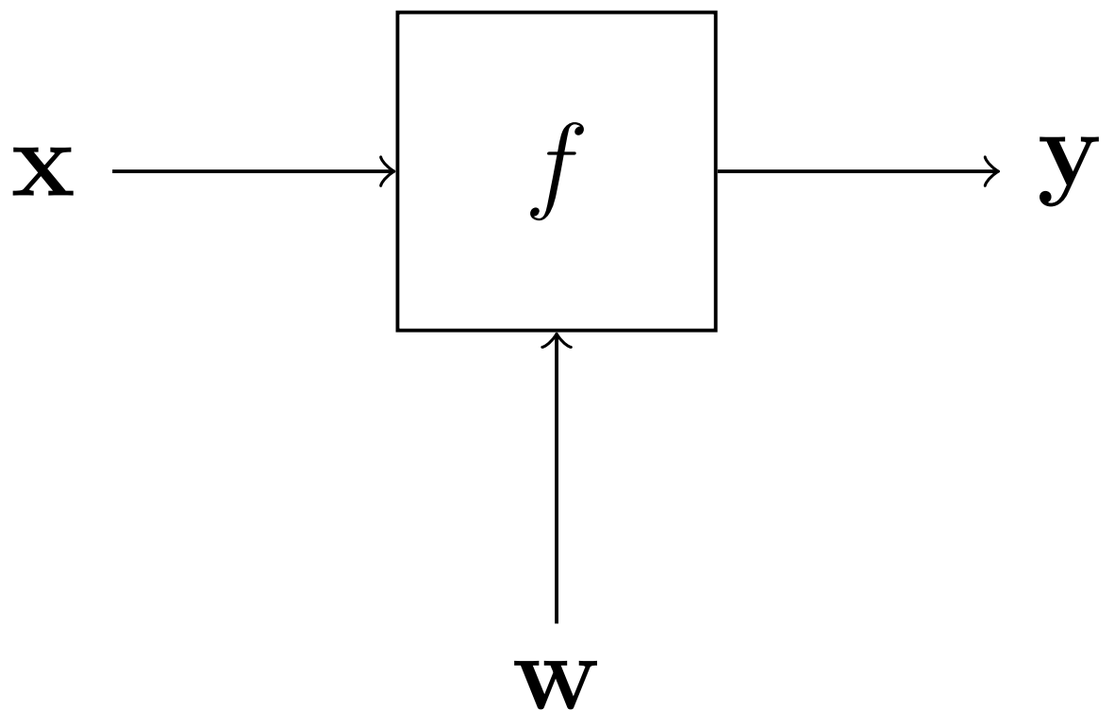


$$\frac{dz}{d\mathbf{w}_l} = \frac{d}{d\mathbf{w}_l}\left[\ell_{\mathbf{y}} \circ f_L(\cdot;\mathbf{w}_L) \circ \ldots \circ f_2(\cdot;\mathbf{w}_2) \circ f_1(\mathbf{x}_0;\mathbf{w}_1)\right]$$

$$\frac{dz}{d\mathbf{w}_l} = \frac{dz}{d(\operatorname{vec}\mathbf{x}_L)^\top}\frac{d\operatorname{vec}\mathbf{x}_L}{d(\operatorname{vec}\mathbf{x}_{L-1})^\top}\cdots\frac{d\operatorname{vec}\mathbf{x}_{l+1}}{d(\operatorname{vec}\mathbf{x}_l)^\top}\frac{d\operatorname{vec}\mathbf{x}_l}{d\mathbf{w}_l^\top}$$

We can add *any* functional module *f* so long as its interface provides:
(1) derivative of output wrt to input
(2) derivative of output wrt parameter

Slide credit: Deva Ramanan

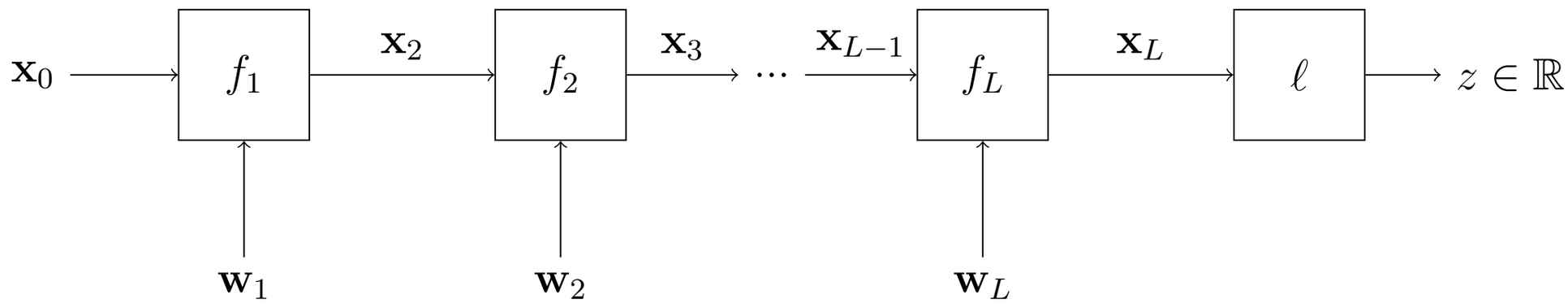


$\frac{\partial y}{\partial w}$: needed to compute gradiant updates for this block

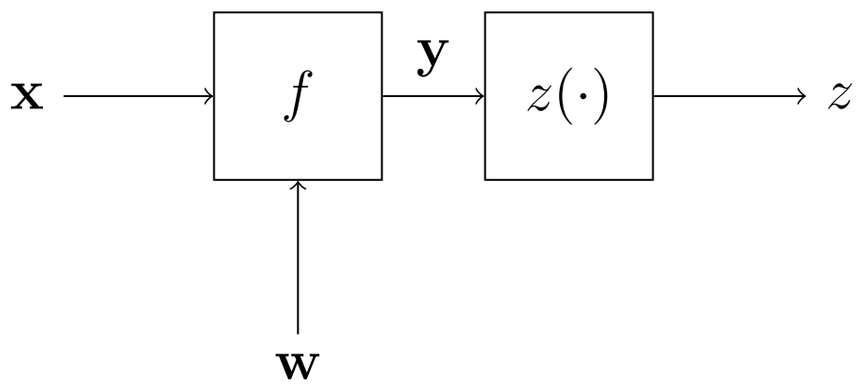
$\frac{\partial y}{\partial x}$: needed to compute gradiant updates next block down stream

Slide credit: Deva Ramanan

Computational savings: cache intermediate gradients

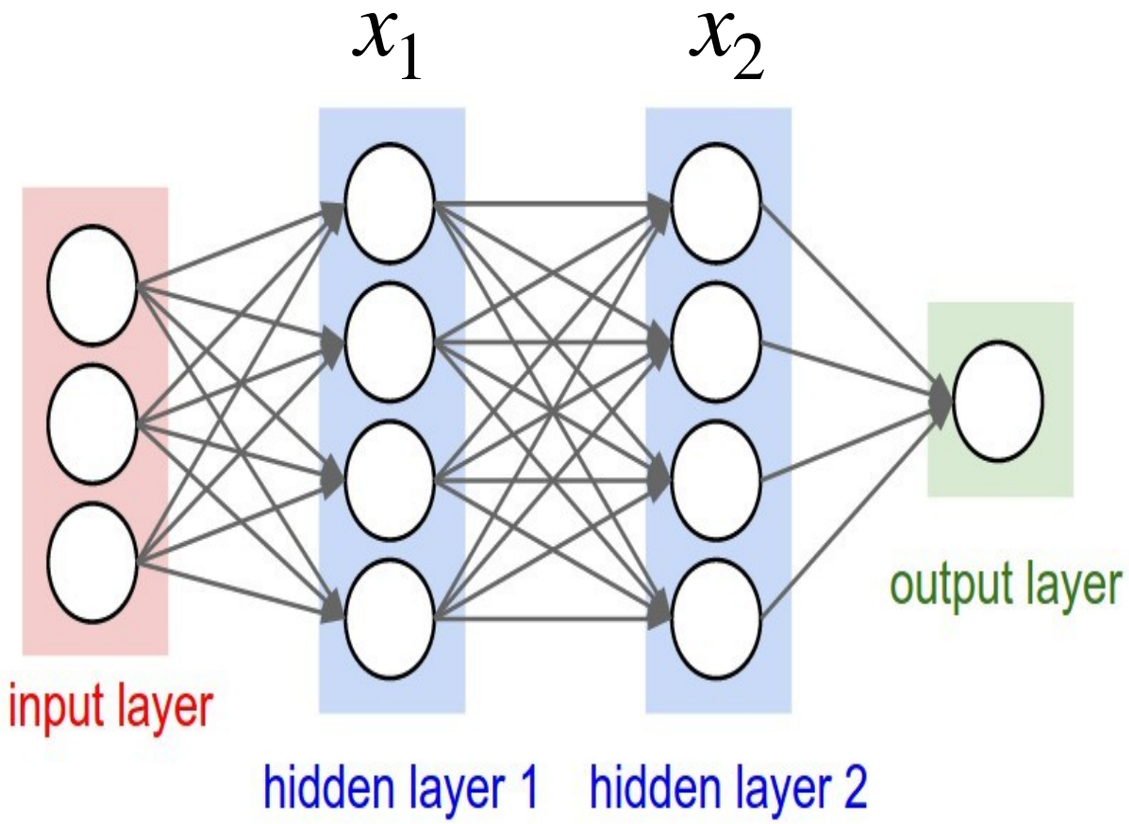


$$\underbrace{\ell \circ f_L(\cdot, \mathbf{w}_L) \circ f_{L-1}(\cdot, \mathbf{w}_{L-1}) \cdots \circ f_{l+1}(\cdot, \mathbf{w}_{l+1})}_{z(\cdot)} \circ f_l(\mathbf{x}_l, \mathbf{w}_l) \circ \ldots$$



Slide credit: Deva Ramanan

Artificial Neural Networks



$$x_i \in \mathbb{R}^{H \times 1}$$

$$W_i \in \mathbb{R}^{D \times H}$$

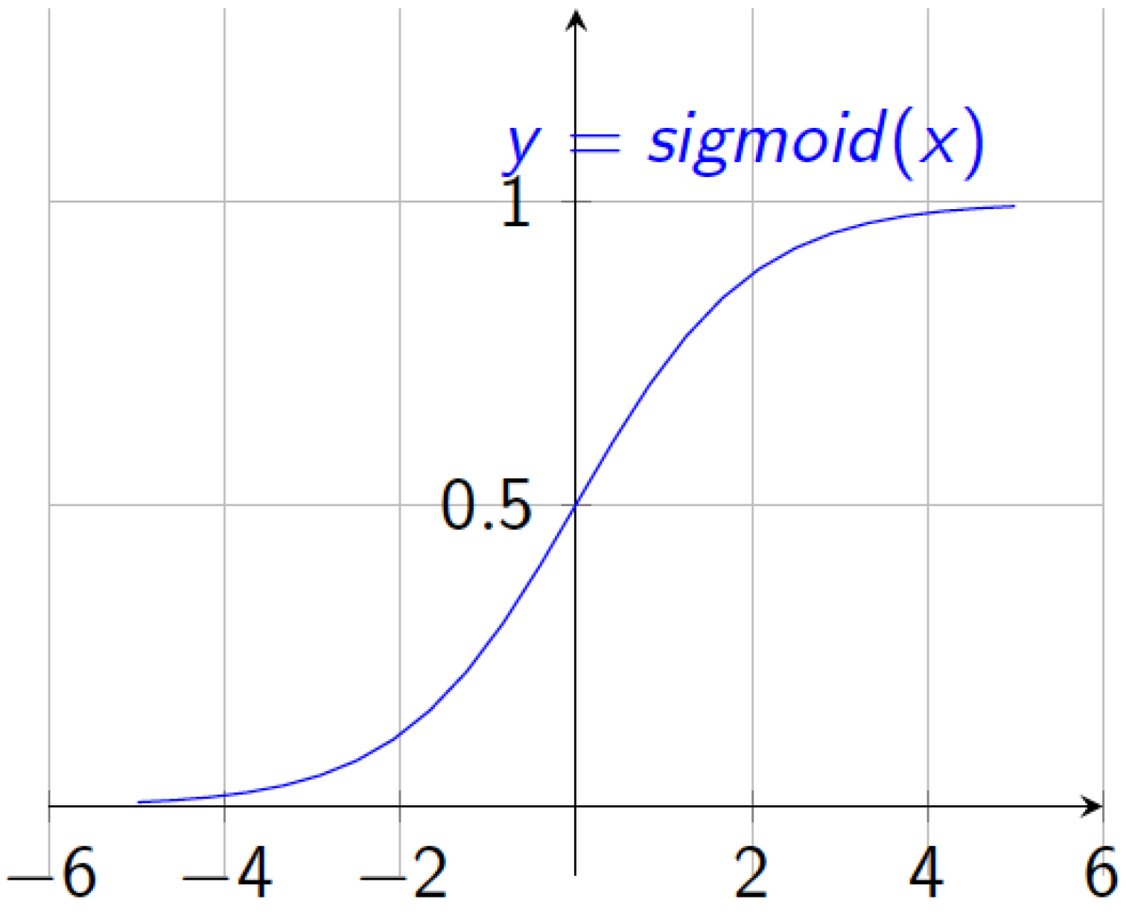
$$b_i \in \mathbb{R}^{D \times 1}$$

$$x_{i+1} = f\left(W_i x_i + b_i\right)$$

$$\min_{\theta} \sum_{j} \mathcal{L}\left(x_{last}^{j}, y^{j}\right)$$

Non-linearities: sigmoid

$$f(a) = sigmoid(a) = \frac{1}{1 + e^{-a}}$$

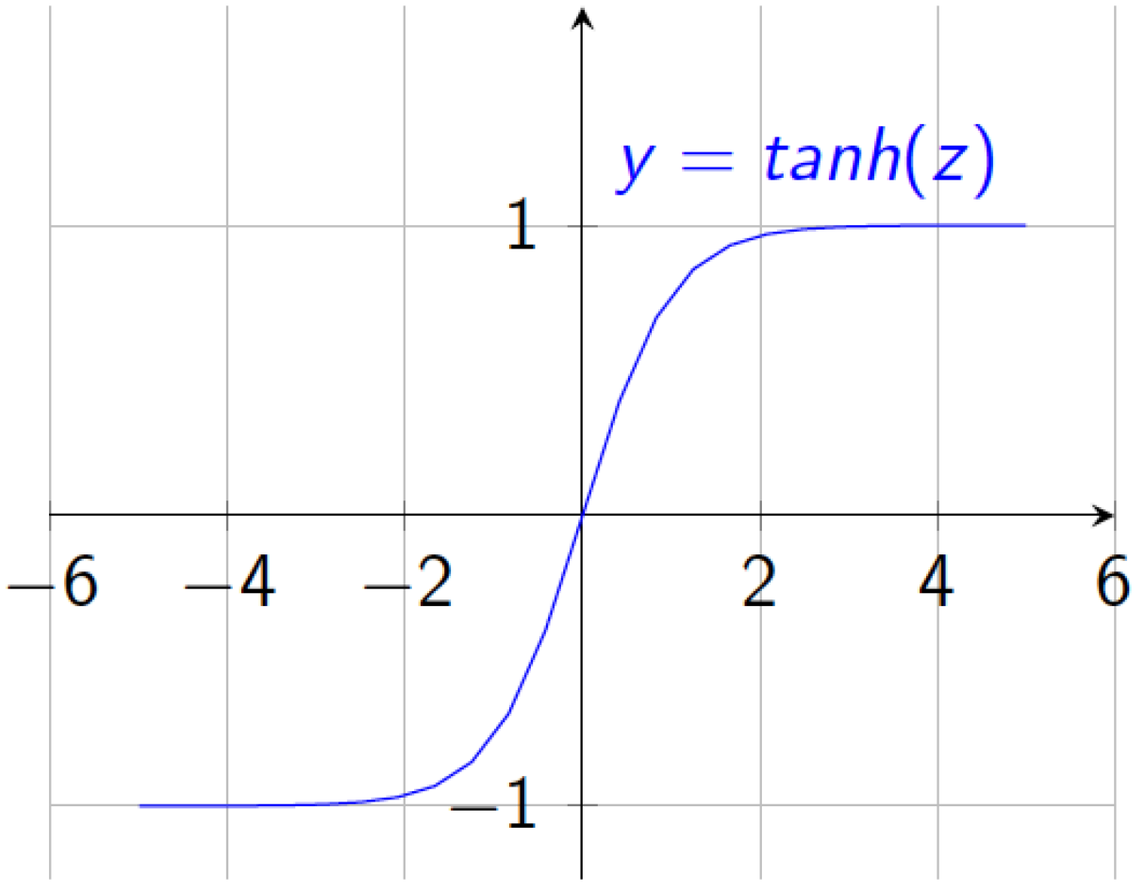


- Interpretation as firing rate of neuron
- Bounded between [0,1]
- Saturation for large +ve,-ve inputs
- Gradients go to zero
- Outputs centered at 0.5 (poor conditioning)
- Not used in practice

Slide credit: Antonio Torralba

Non-linearities: tanh

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



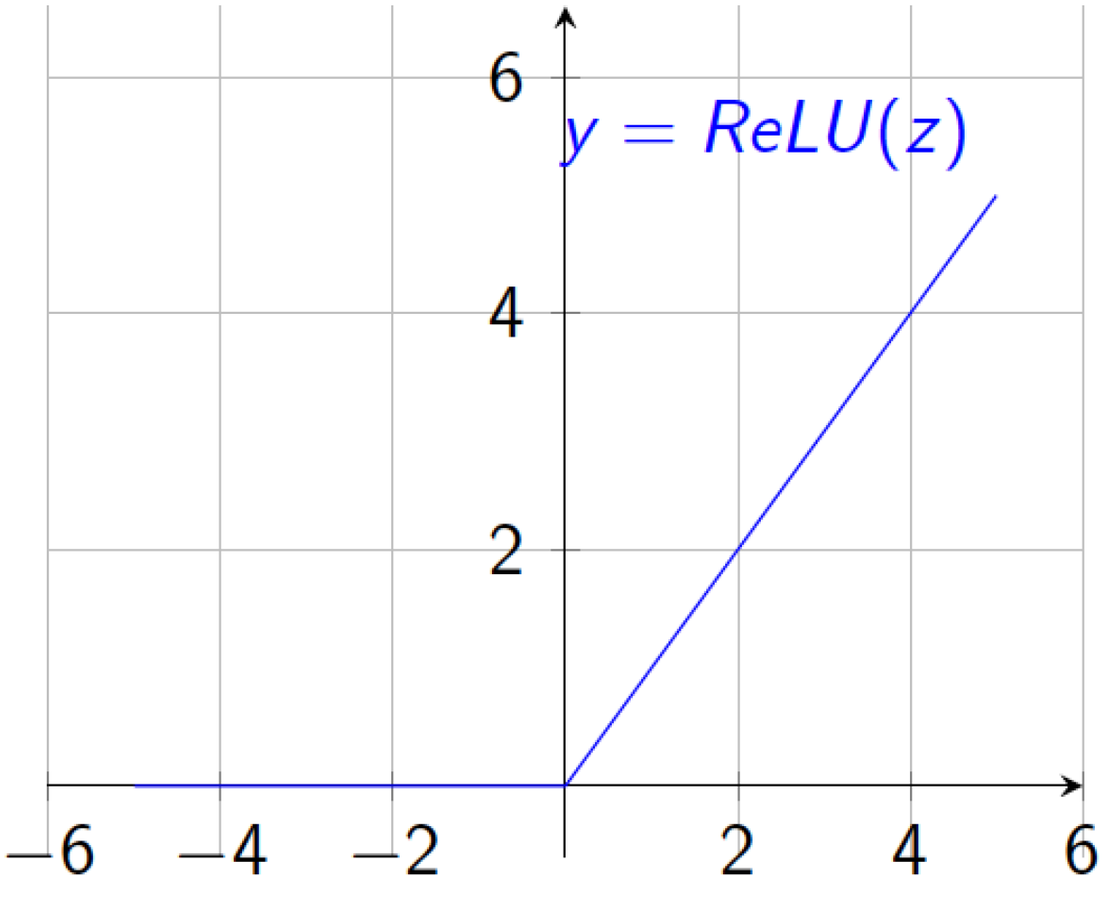
- Bounded between [-1,+1]
- Saturation for large +ve,-ve inputs
- Gradients go to zero
- Outputs centered at 0
- Preferable to sigmoid

$$\tanh(x) = 2\,\text{sigmoid}(2x) - 1$$

Slide credit: Antonio Torralba

Non-linearities: rectified linear (ReLU)

$$f(a) = \max(a, 0)$$



- Unbounded output (on positive side)
- Efficient to implement:

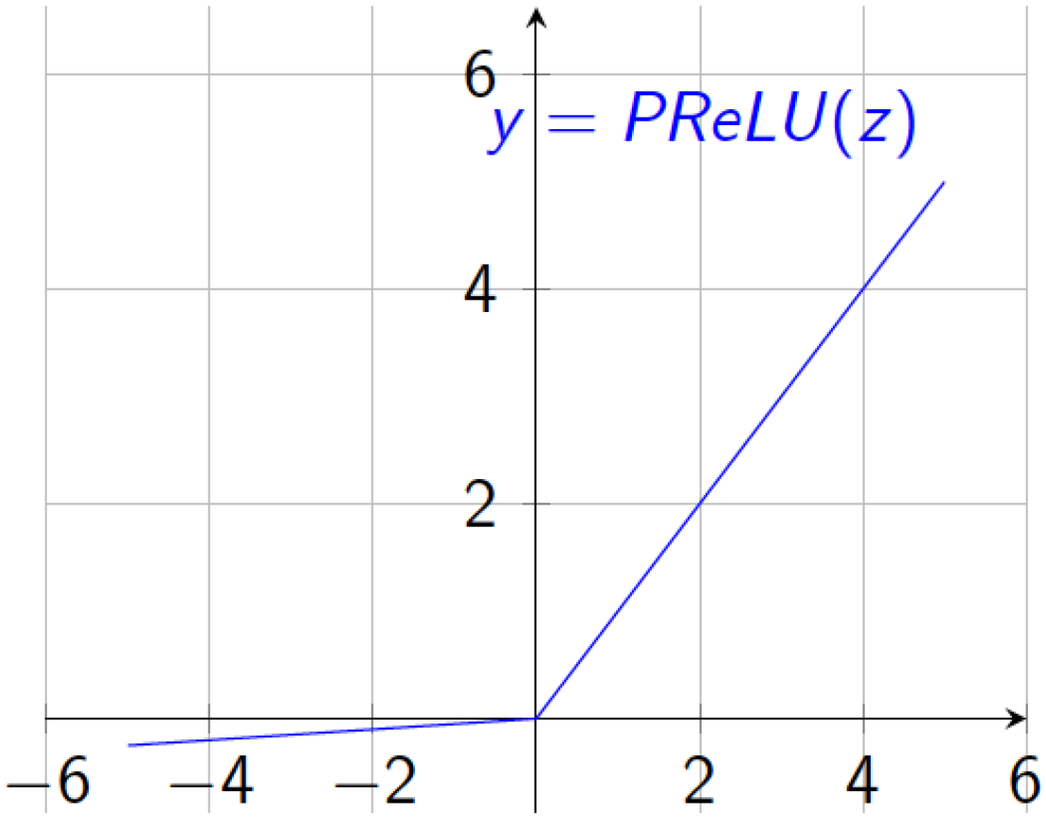
$$f'(a) = \frac{df}{da} = \begin{cases} 0 & a < 0 \\ 1 & a \geq 0 \end{cases}$$

- Also seems to help convergence (see 6x speedup vs tanh in [Krizhevsky et al.])
- Drawback: if strongly in negative region, unit is dead forever (no gradient).
- Default choice: widely used in current models.

Slide credit: Antonio Torralba

Non-linearities: Leaky ReLU

$$f(a) = \begin{cases} \max(0,a) & a > 0 \\ a \min(0,a) & a < 0 \end{cases}$$



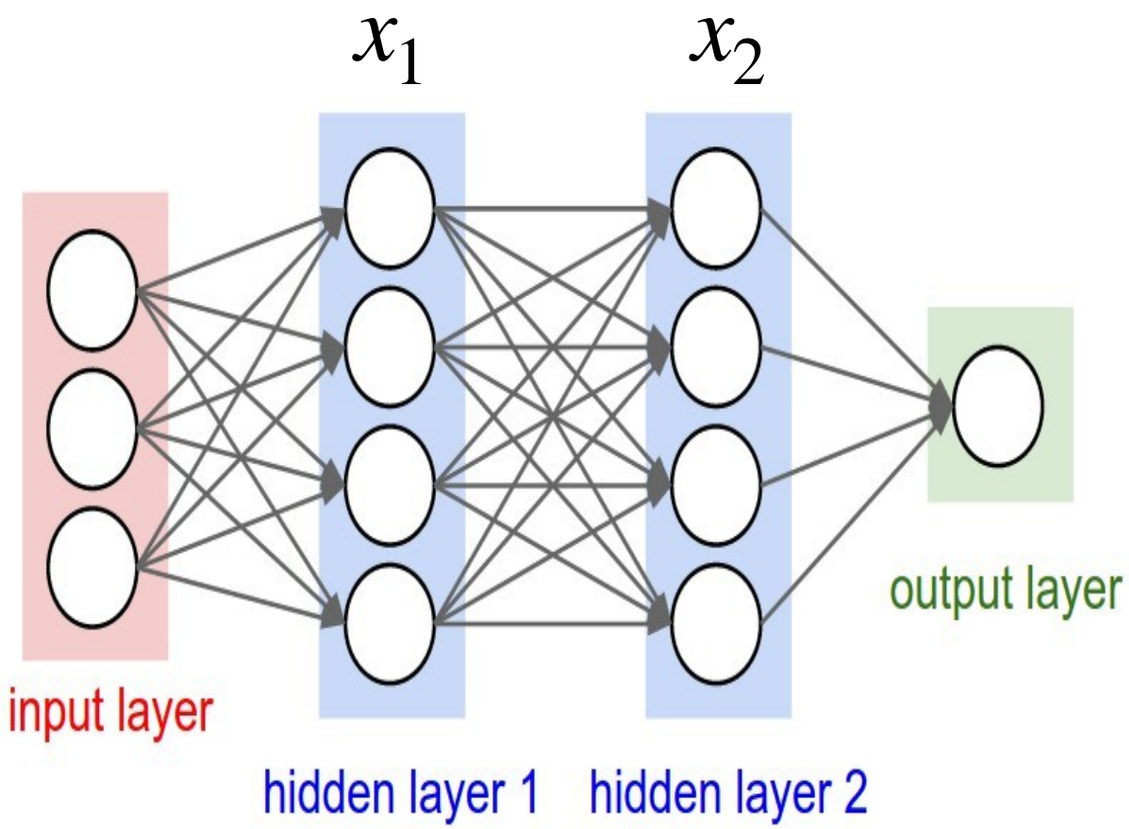
- where α is small (e.g. 0.02)
- Efficient to implement:

$$f'(a) = \frac{df}{da} = \begin{cases} -a & a < 0 \\ 1 & a > 0 \end{cases}$$

- Also known as probabilistic ReLU (PReLU)
- Has non-zero gradients everywhere (unlike ReLU)
- α can also be learned (see Kaiming He et al. 2015).

Slide credit: Antonio Torralba

Multilayer Perceptron (MLP)



$$x_i \in \mathbb{R}^{H \times 1}$$

$$W_i \in \mathbb{R}^{D \times H}$$

$$b_i \in \mathbb{R}^{D \times 1}$$

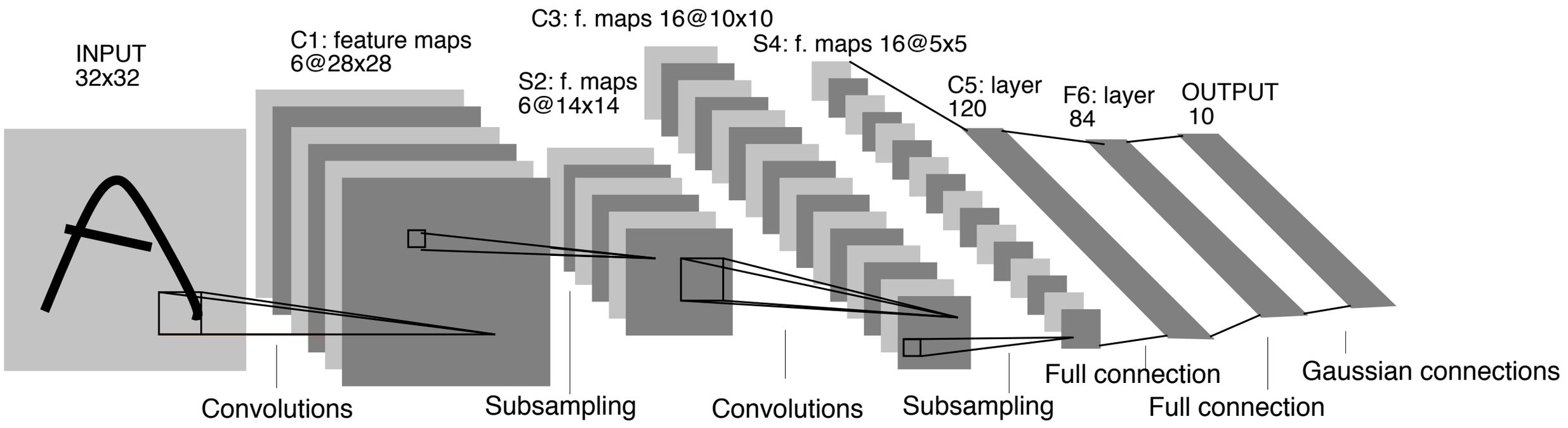
$$x_{i+1} = f\left(W_i x_i + b_i\right)$$

$$\min_{\theta} \sum_{j} \mathscr{L}\left(x_{last}^{j}, y^{j}\right)$$

Convolutional Networks

Gradient-Based Learning Applied to Document Recognition

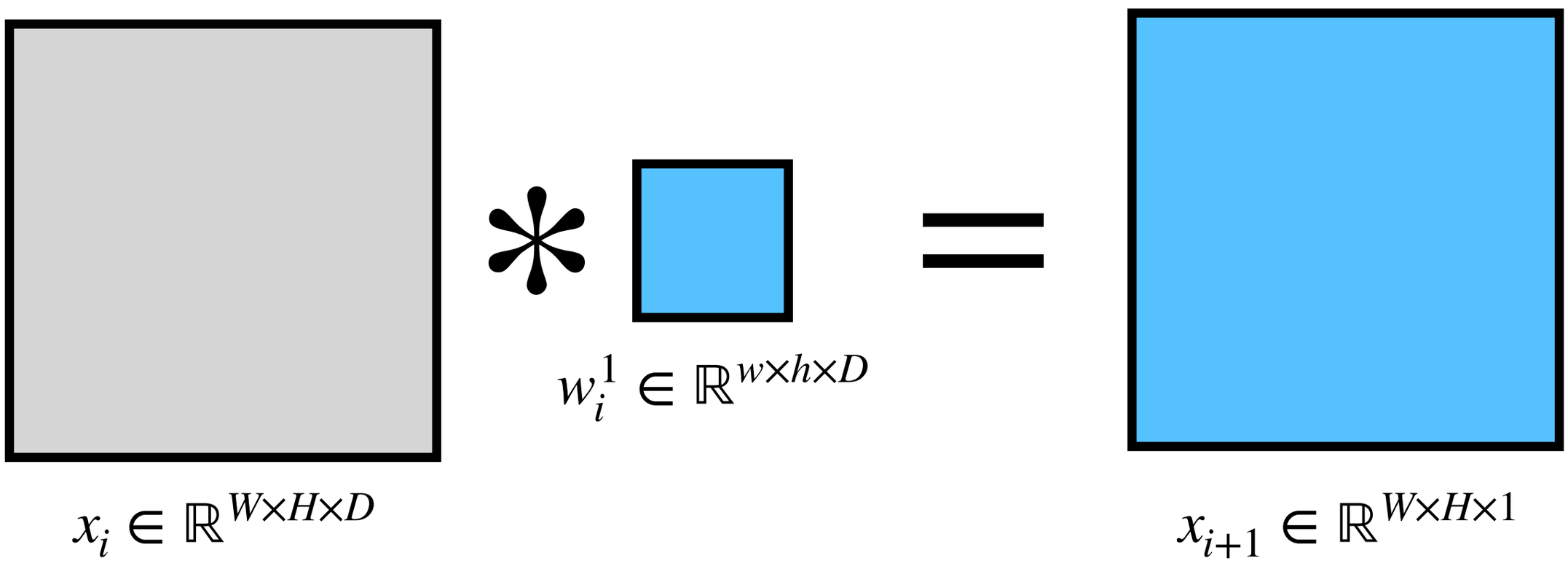
Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner



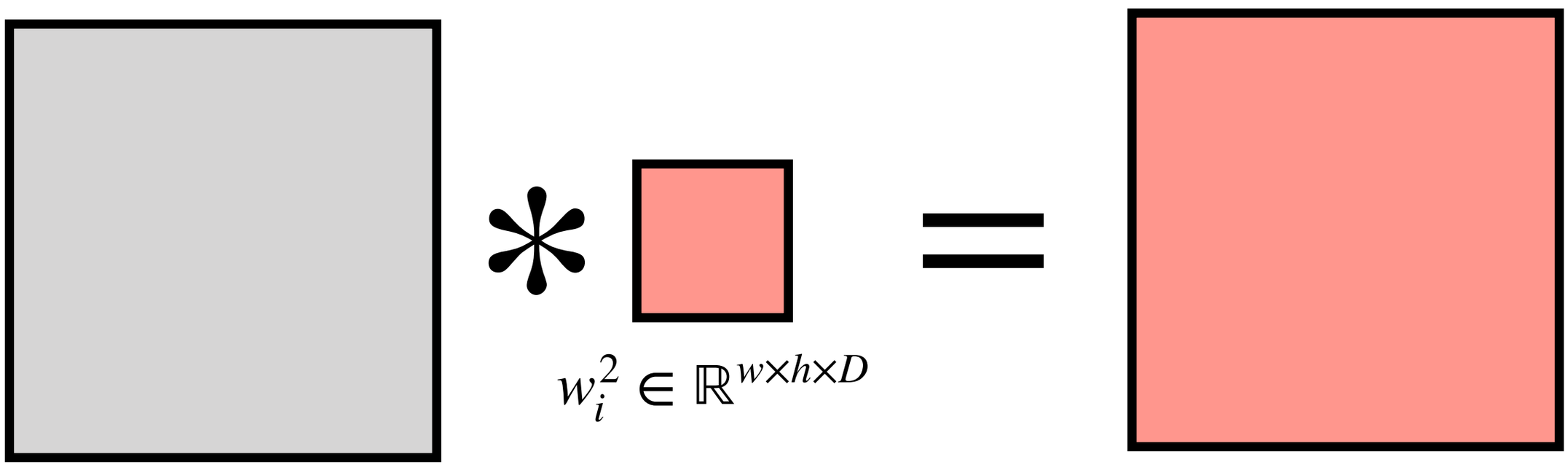
Stack together convolution and pooling (avg + subsample) operations.

Slide credit: Deva Ramanan

Convolutional Layer



Convolutional Layer

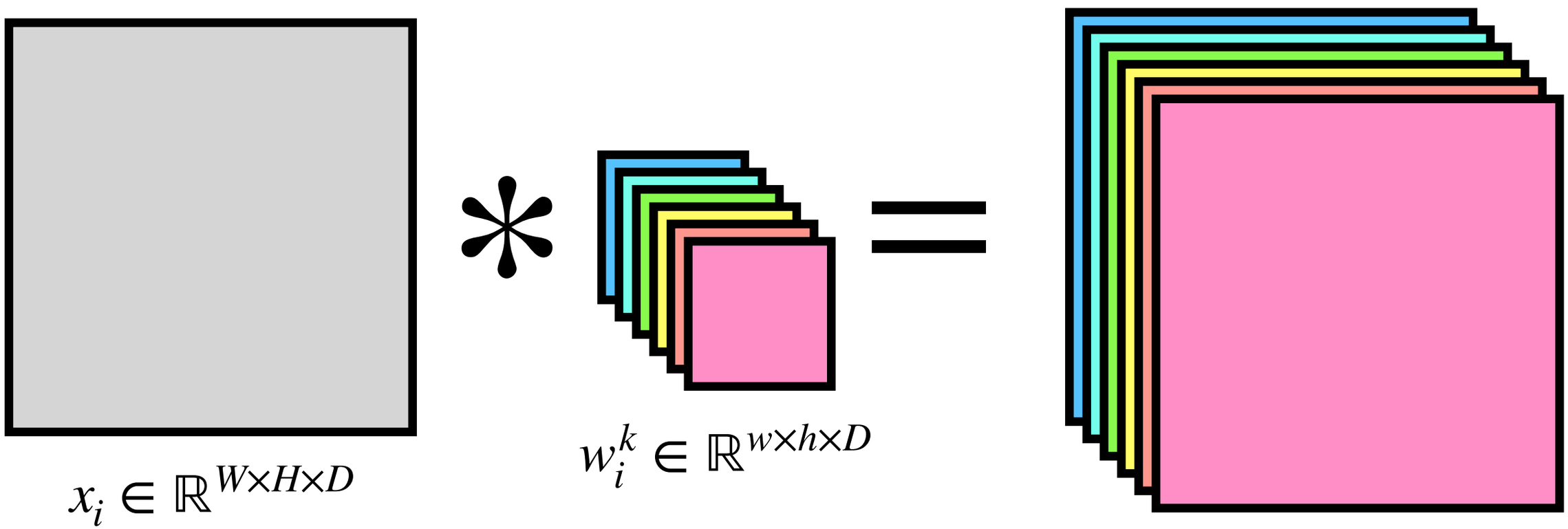


$x_i \in \mathbb{R}^{W \times H \times D}$

$w_i^2 \in \mathbb{R}^{w \times h \times D}$

$x_{i+1} \in \mathbb{R}^{W \times H \times 1}$

Convolutional Layer

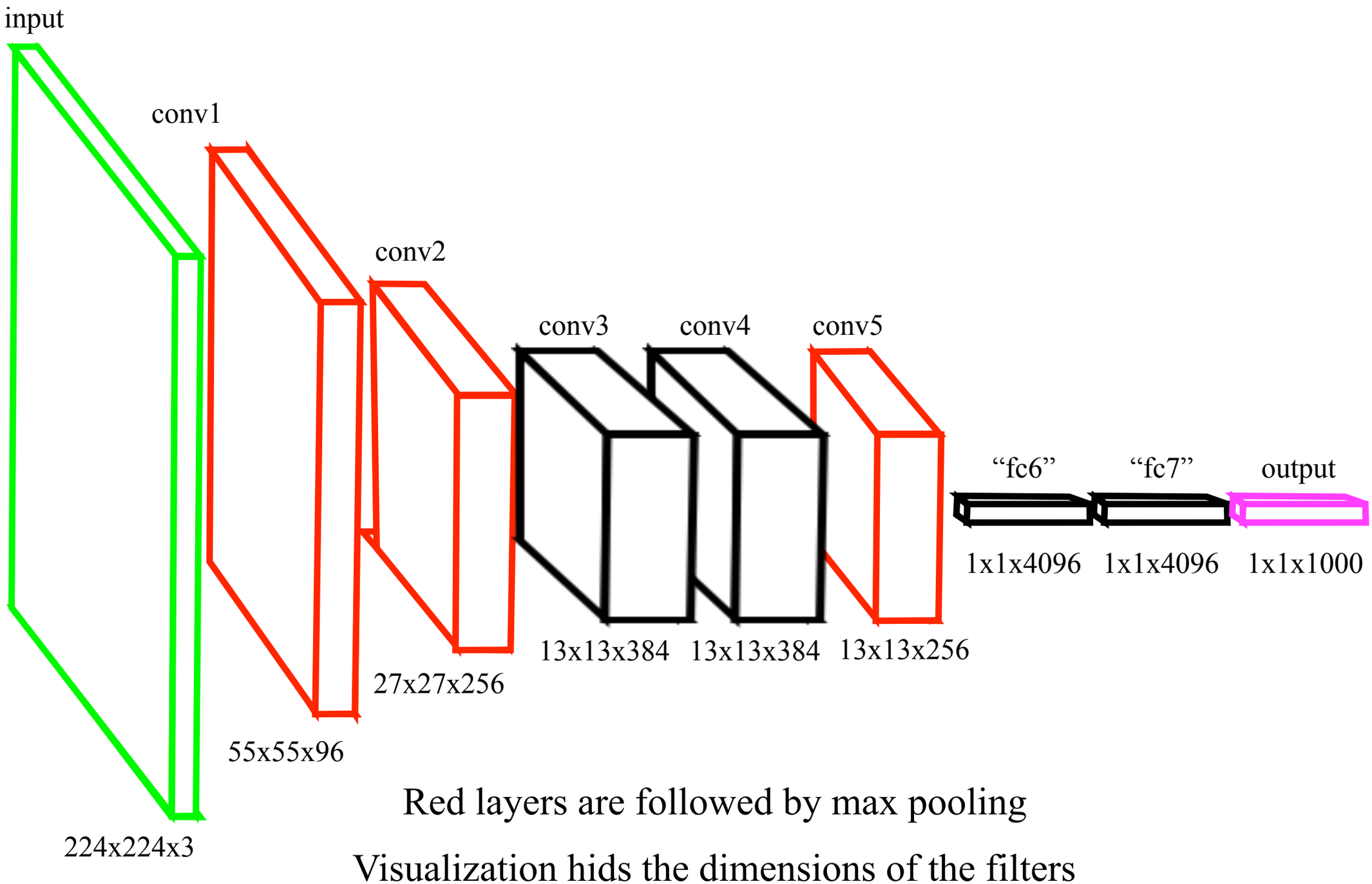


$x_i \in \mathbb{R}^{W \times H \times D}$

$w_i^k \in \mathbb{R}^{w \times h \times D}$

$x_{i+1} \in \mathbb{R}^{W \times H \times K}$

A better visualization of AlexNet



Red layers are followed by max pooling

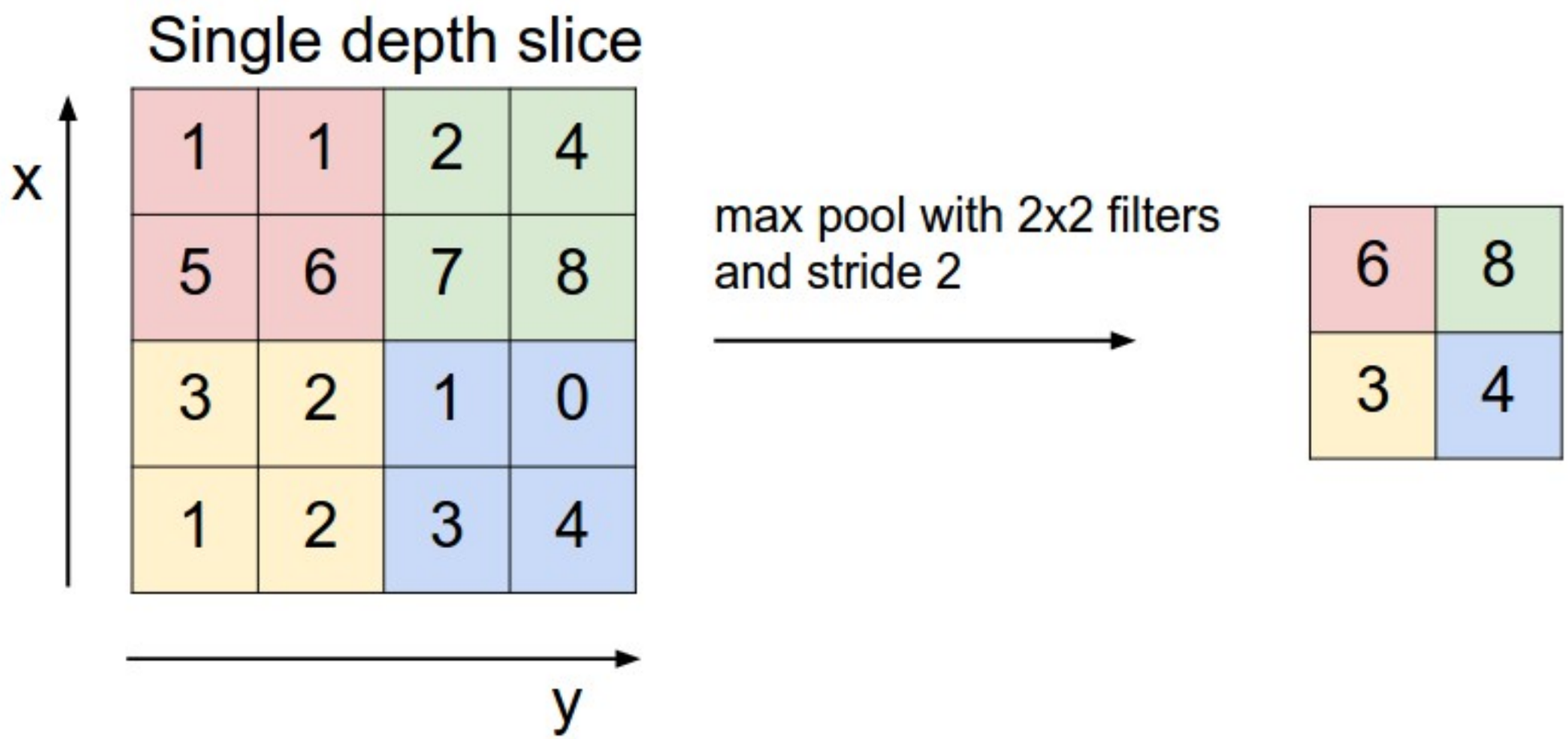
Visualization hids the dimensions of the filters

Slide credit: Deva Ramanan

Max Pooling

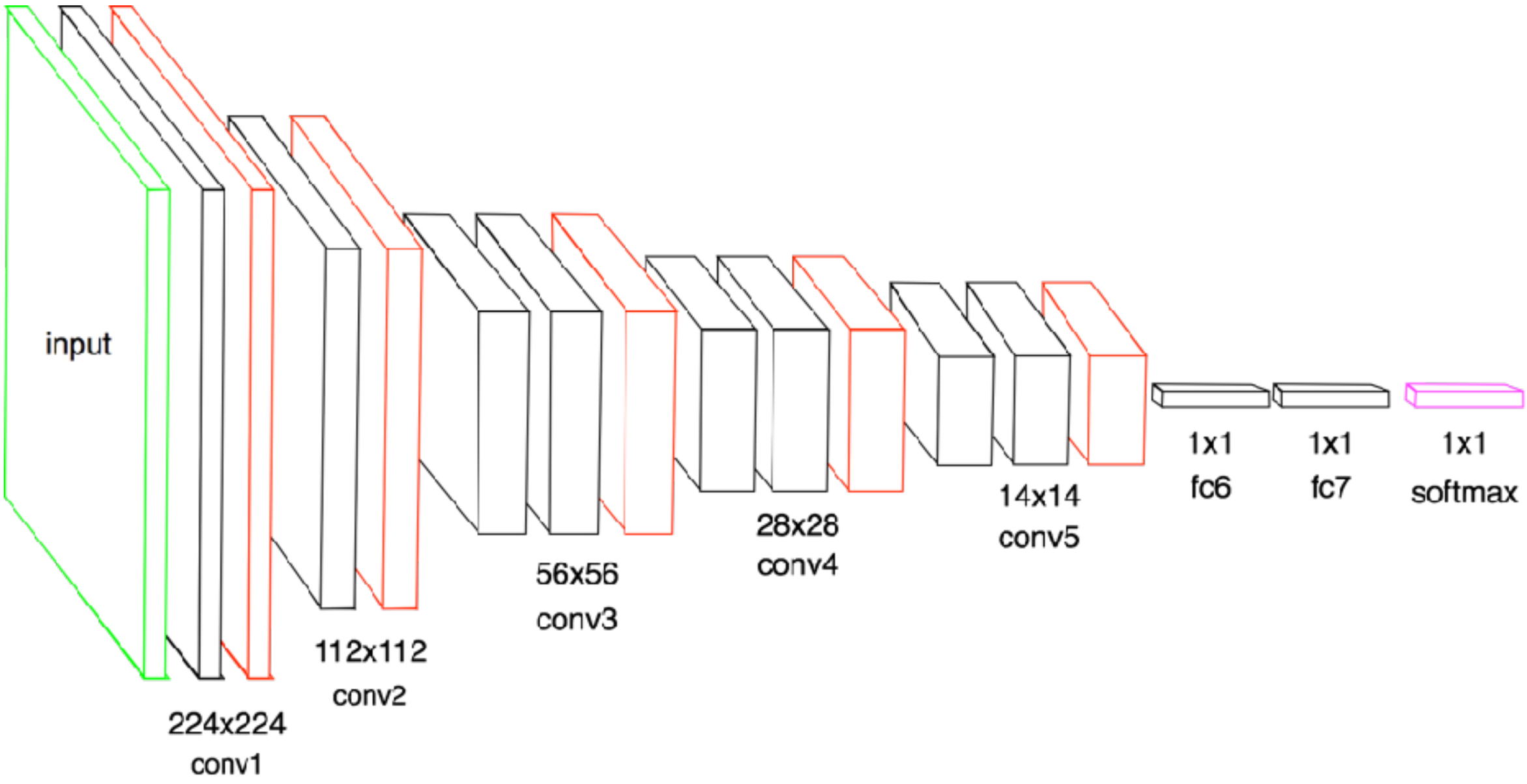
Fast way to resize an “image”

What’s the derivative?



Slide credit: Andrej Karpathy

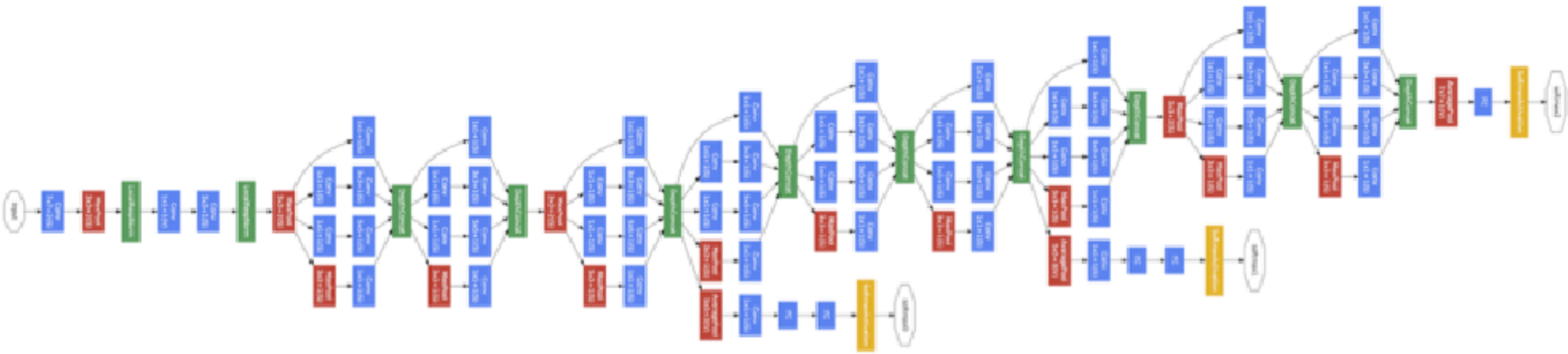
VGG19



All filter dimensions 3x3 except fc6 (which uses 7x7)

Slide credit: Deva Ramanan

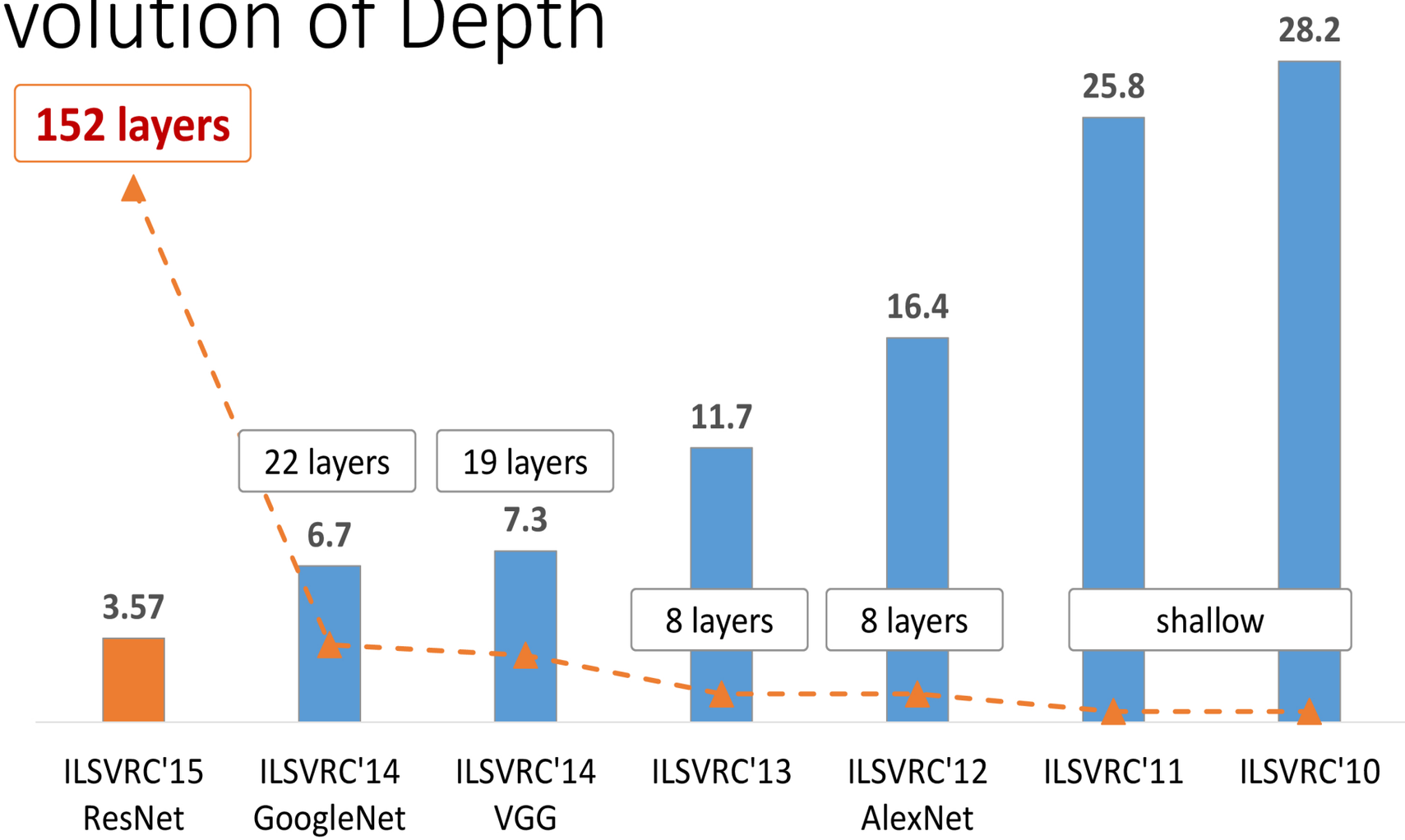
Inception / resnet



Convolution
Pooling
Softmax
Other

Slide credit: Deva Ramanan

Revolution of Depth



ImageNet Classification top-5 error (%)

ICCV15 International Conference on Computer Vision

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.

Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)

VGG, 19 layers
(ILSVRC 2014)

ResNet, 152 layers
(ILSVRC 2015)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.

How many layers?

- 1 Layer = Linear
- 2 Layer = theoretically can approximate any function, if each layer is wide enough
- In practice, deeper models are better than wider models
- Active area of research: what is the best architecture?
- Architecture matters for performance, but machines will learn architectures soon… I hope…

Examples

- From Clarifai.com

Predicted Tags:

food	(16.00%)
dinner	(3.10%)
bbq	(2.90%)
market	(2.50%)
meal	(1.40%)
turkey	(1.40%)
grill	(1.30%)
pizza	(1.30%)
eat	(1.10%)
holiday	(1.00%)

Stats:

Size: 247.24 KB

Time: 110 ms

Examples

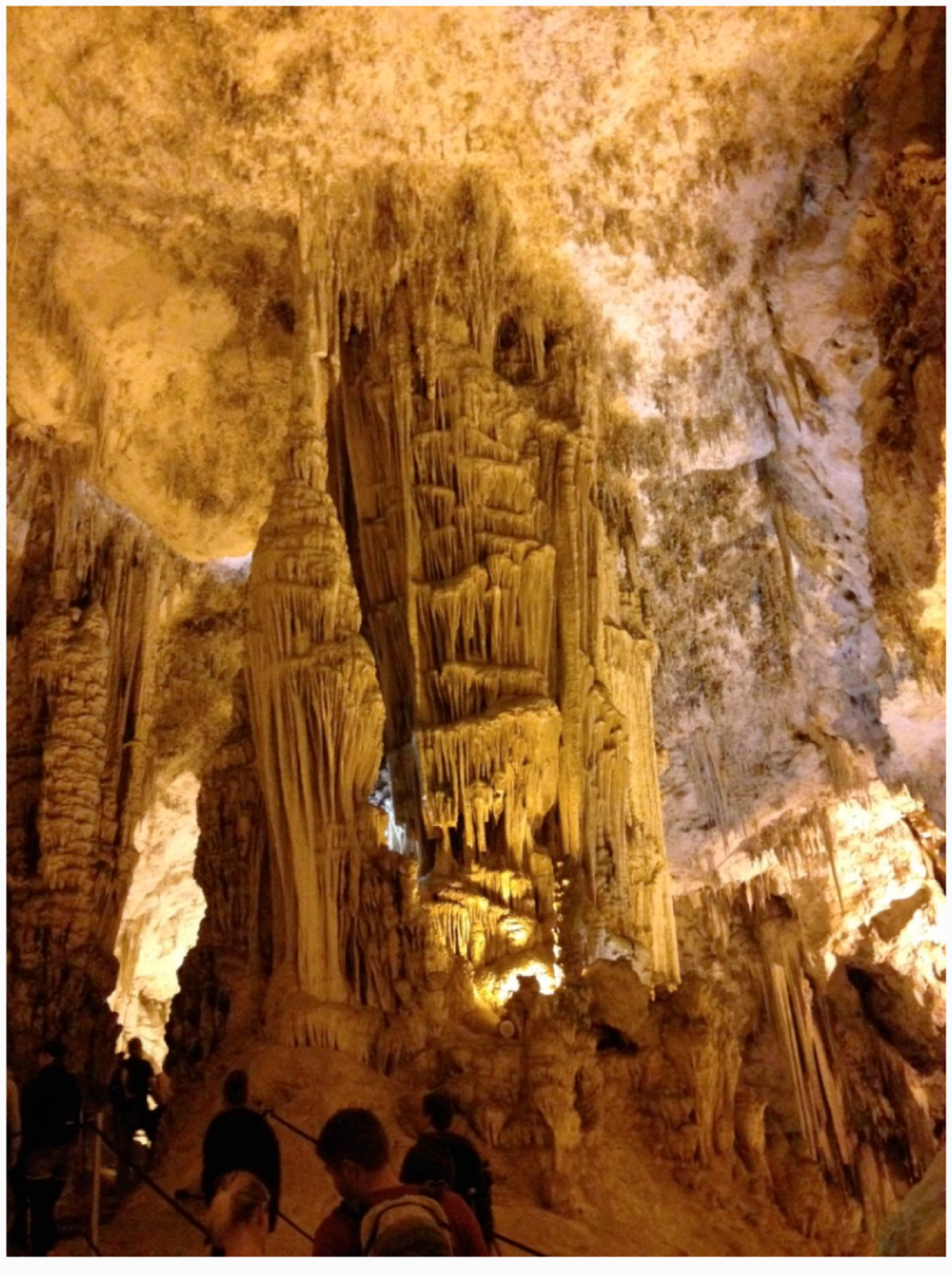
- From Clarifai.com

Predicted Tags:

ship	(2.30%)
helsinki	(1.80%)
fish	(1.40%)
port	(1.10%)
istanbul	(1.10%)
beach	(1.00%)
denmark	(1.00%)
copenhagen	(0.90%)
sea	(0.80%)
boat	(0.80%)

Examples

- From Clarifai.com



Predicted Tags:

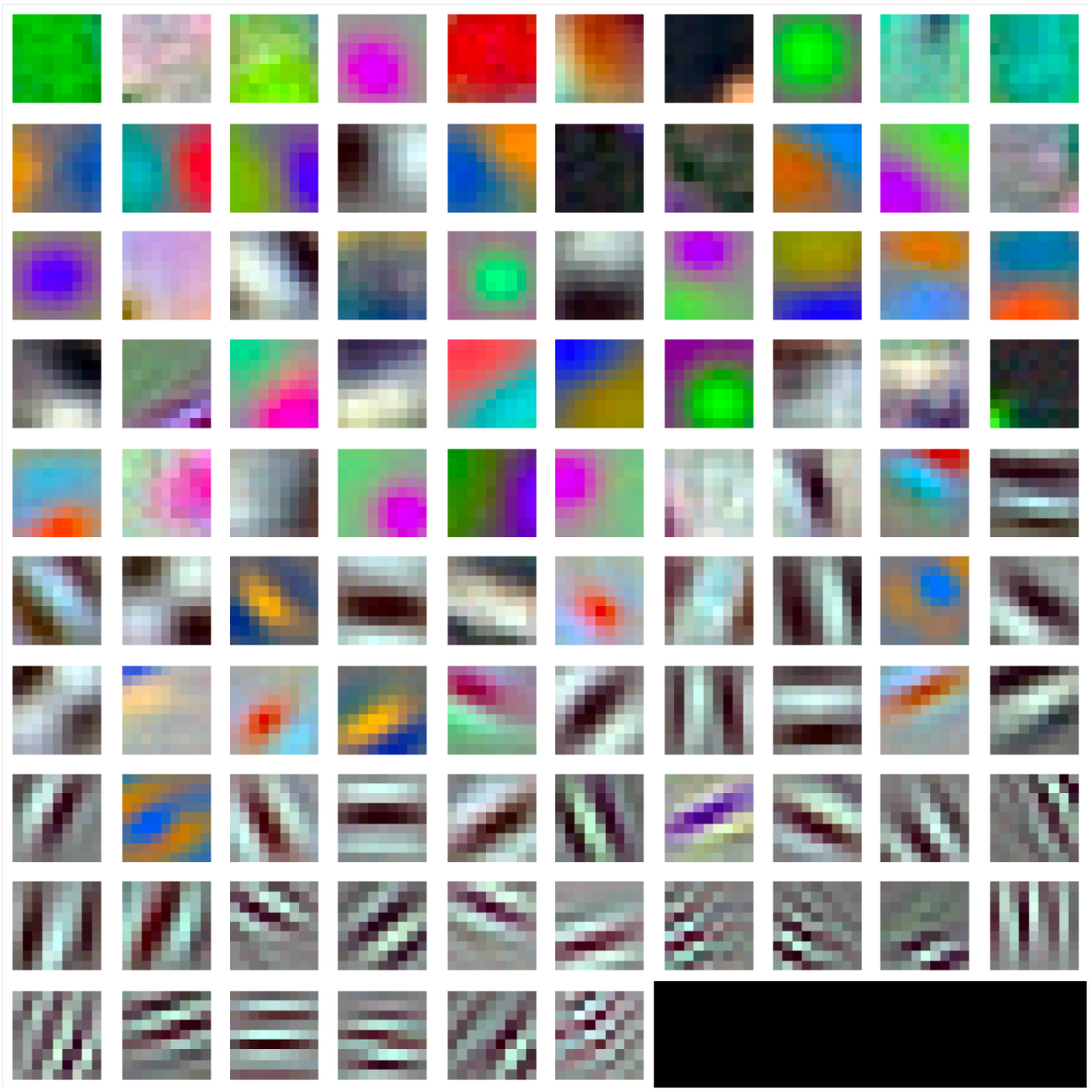
barcelona	(6.50%)
street	(3.00%)
cave	(2.20%)
sagrada	(1.90%)
old	(1.80%)
night	(1.40%)
familia	(1.40%)
jerusalem	(1.40%)
guanajuato	(1.10%)
alley	(1.00%)

Stats:

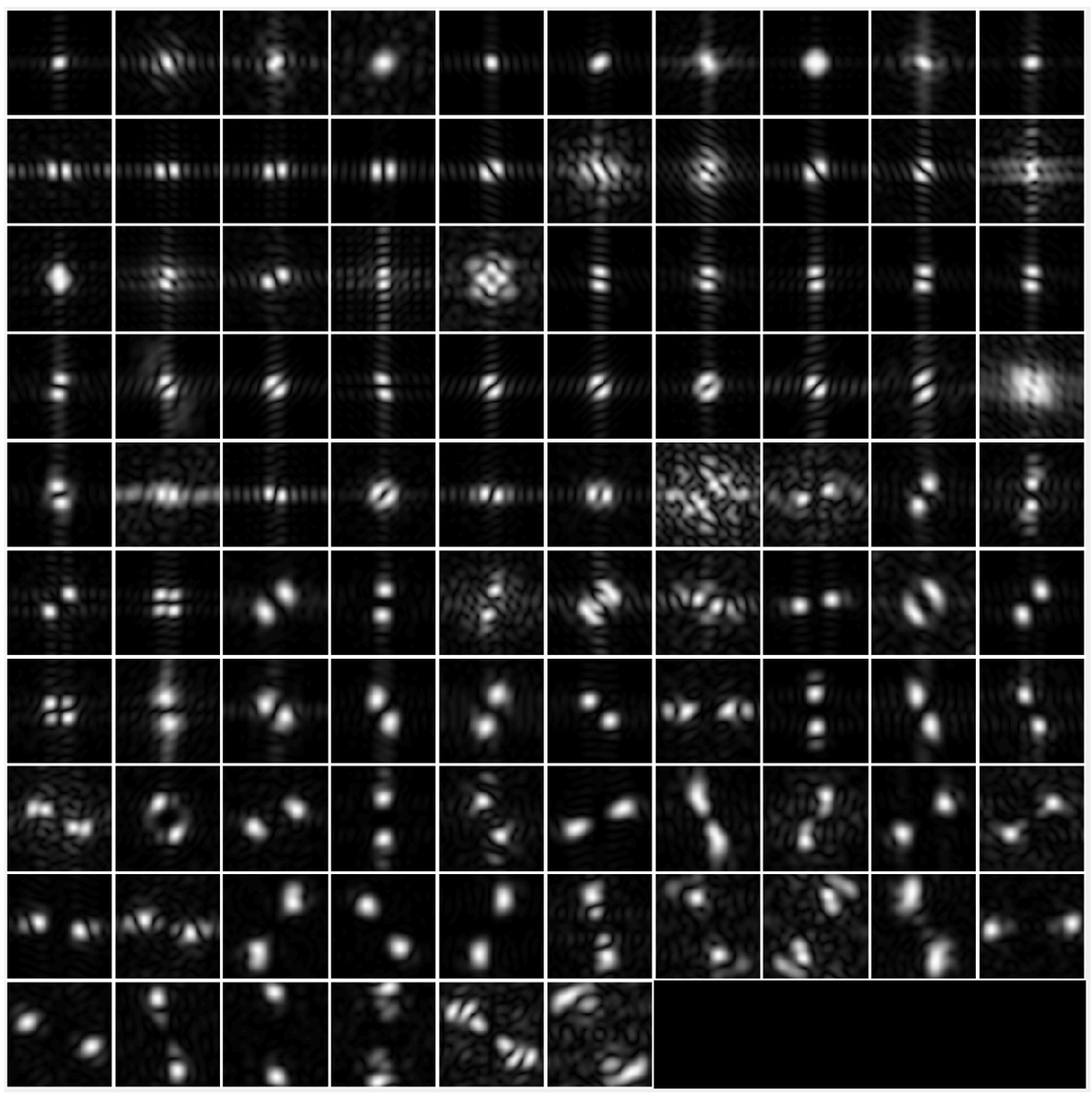
Size: 278.96 KB

Time: 113 ms

Get to know your units

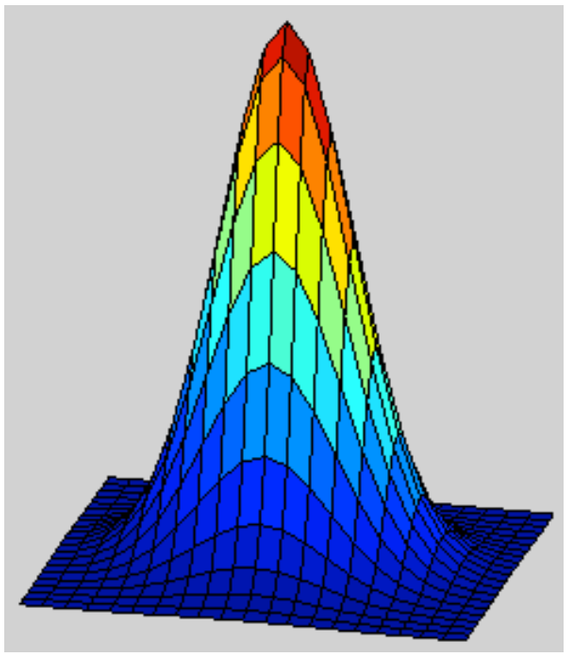


96 Units in conv1

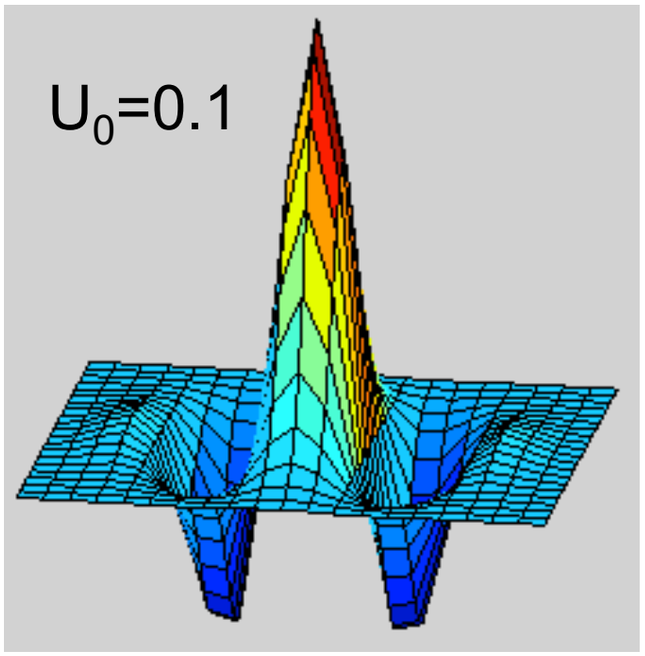


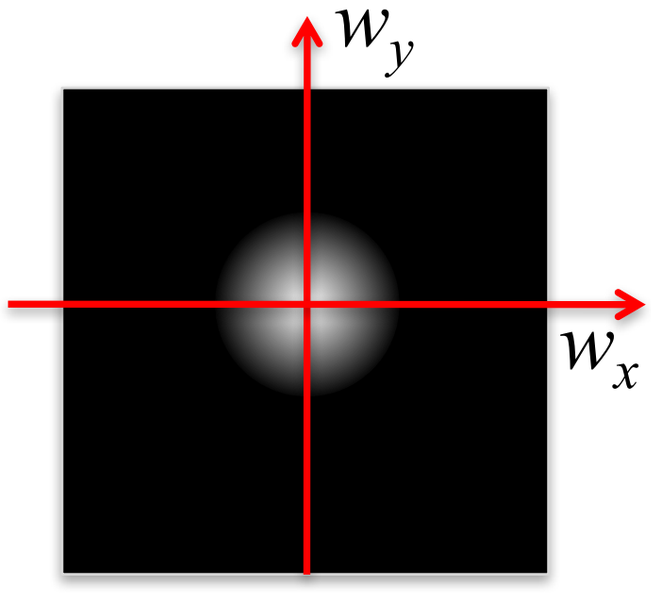
Slide credit: Antonio Torralba

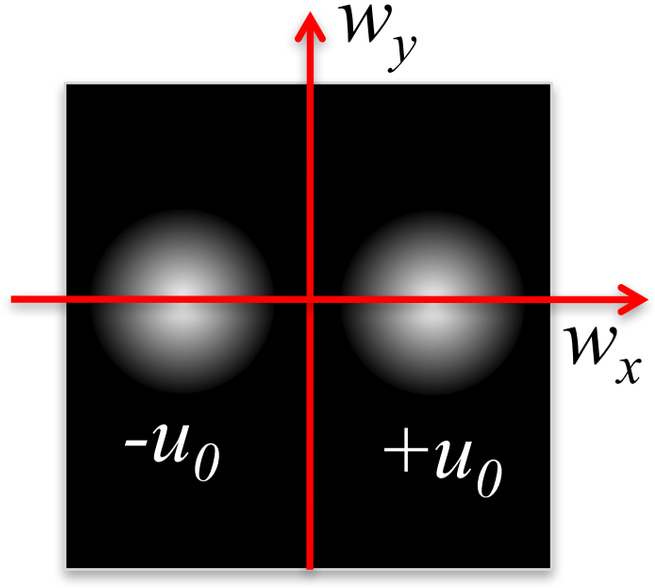
Fourier transform of a Gabor wavelet



$$\psi_c(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \cos(2\pi u_0 x)$$







Slide credit: Antonio Torralba

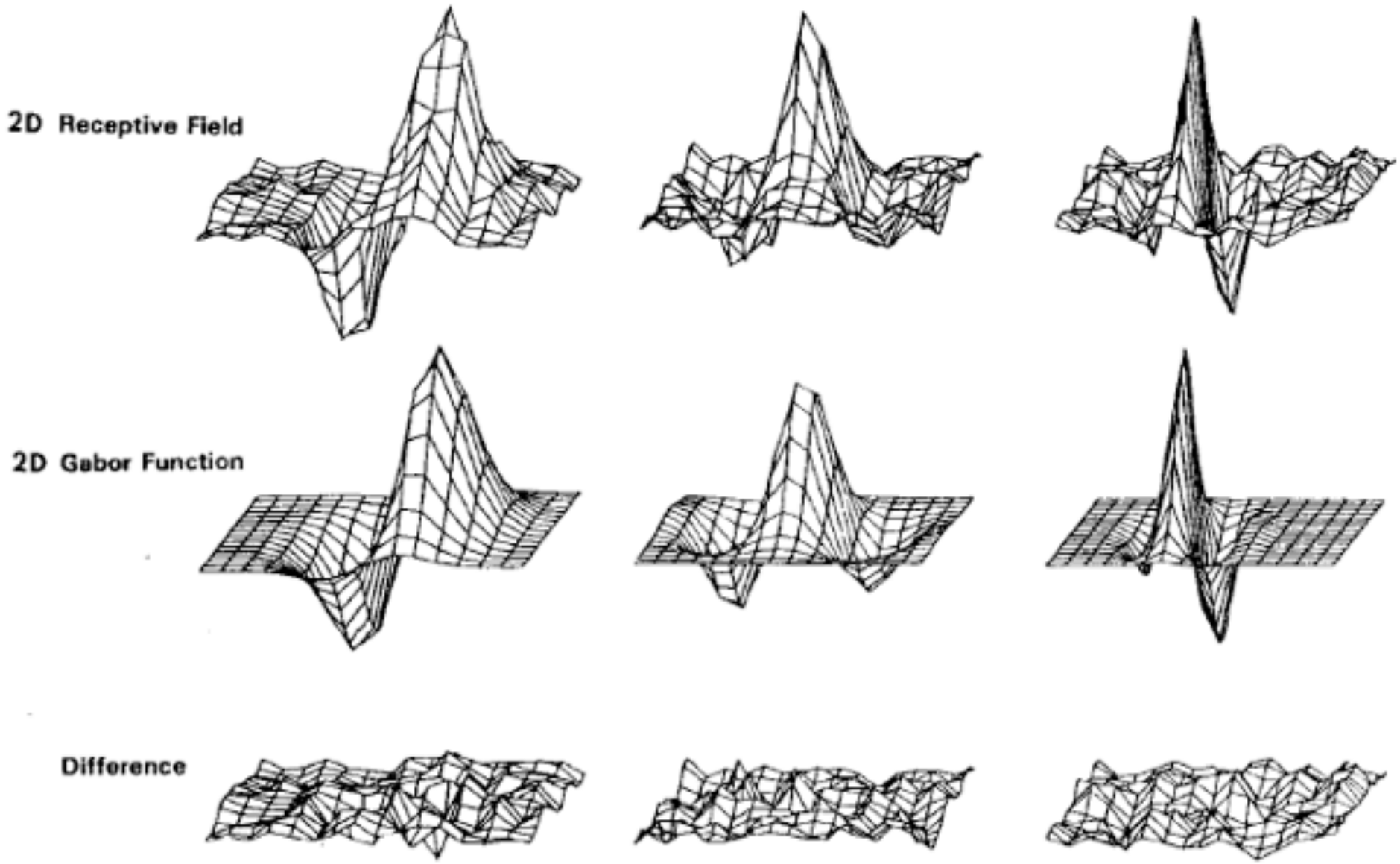
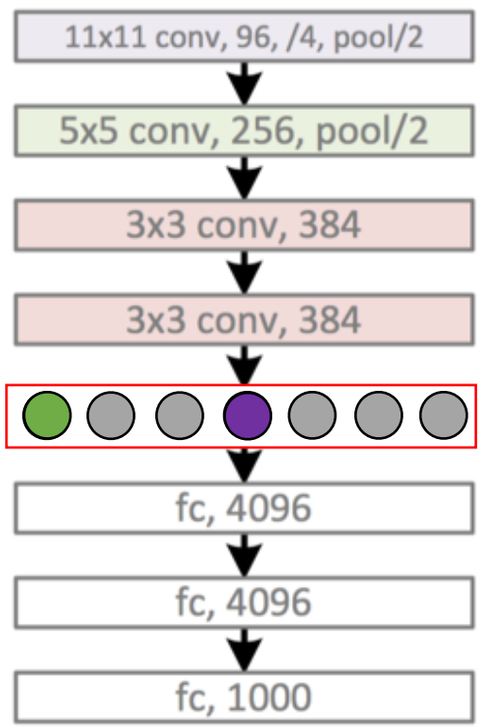


Fig. 5. Top row: illustrations of empirical 2-D receptive field profiles measured by J. P. Jones and L. A. Palmer (personal communication) in simple cells of the cat visual cortex. Middle row: best-fitting 2-D Gabor elementary function for each neuron, described by (10). Bottom row: residual error of the fit, indistinguishable from random error in the Chi-squared sense for 97 percent of the cells studied.

73

John Daugman, 1988

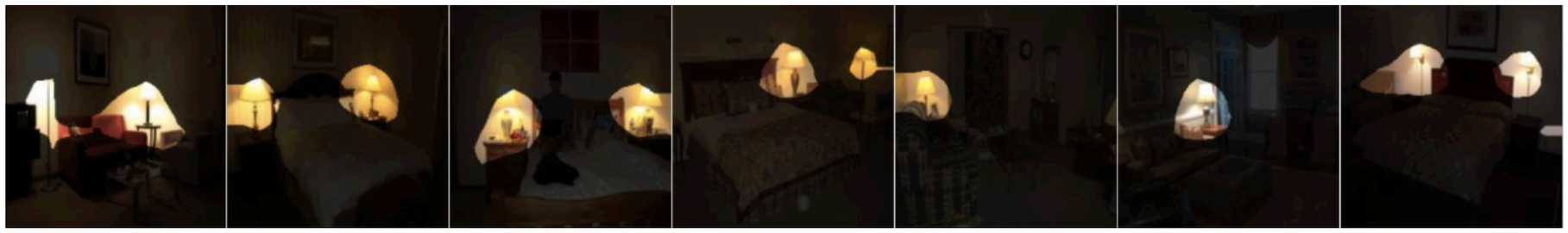
Slide credit: Antonio Torralba



Top Activated Images

Interpretation: lamp

Unit 1



Top Activated Images

Interpretation: car

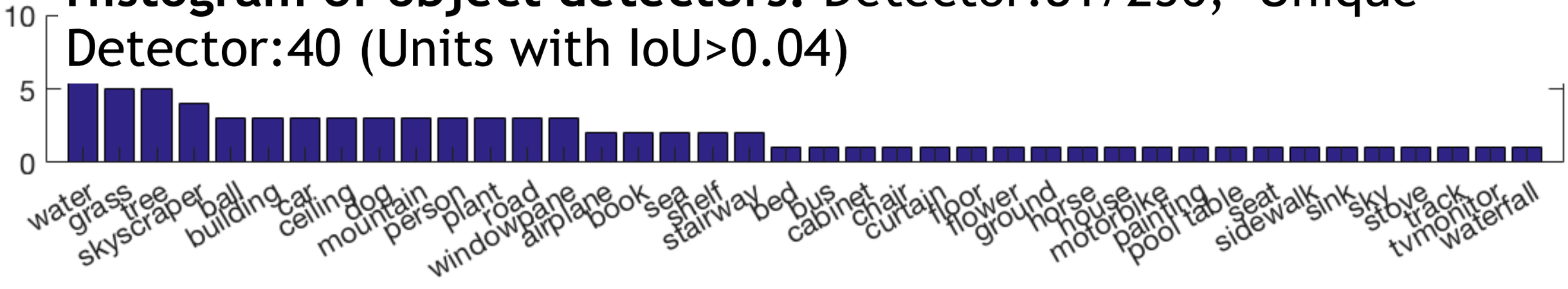
Unit 4

Slide credit: Bolei Zhou

conv5 unit 79 car (object) IoU=0.13

conv5 unit 107 road (object)

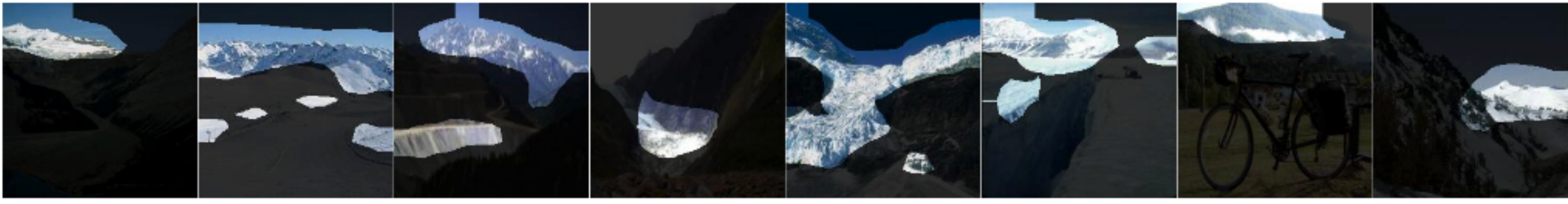
IoU=0.15



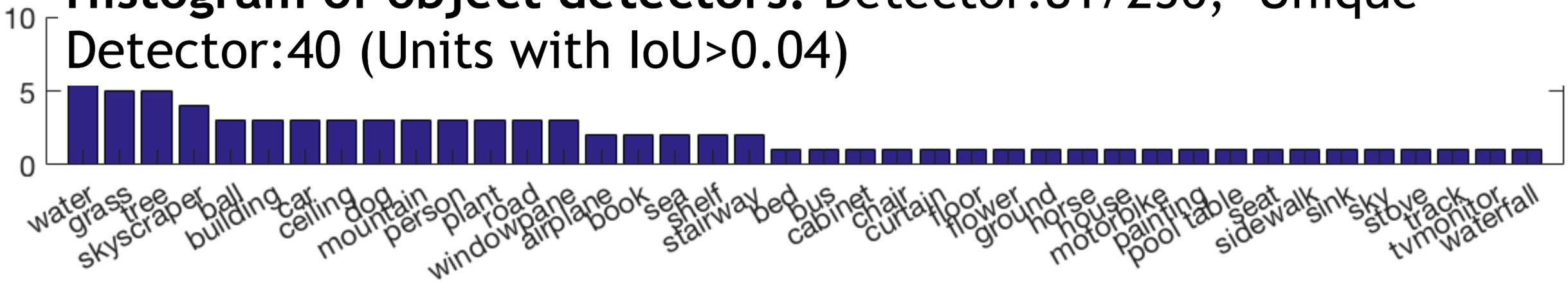
Slide credit: Bolei Zhou

conv5 unit 144 mountain (object) IoU=0.13

conv5 unit 200 mountain (object)



Histogram of object detectors: Detector:81/256, Unique Detector:40 (Units with IoU>0.04)



Slide credit: Bolei Zhou

House

Airplane

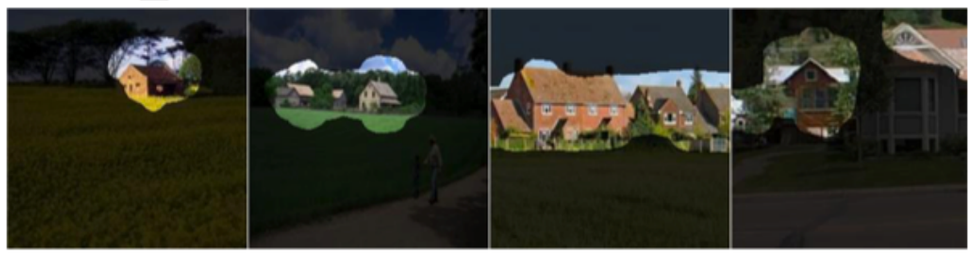
AlexNet

conv5 unit 36 IoU=0.053

conv5 unit 13 IoU=0.101

VGG

conv5_3 unit 243 IoU=0.070



conv5_3 unit 151 IoU=0.150

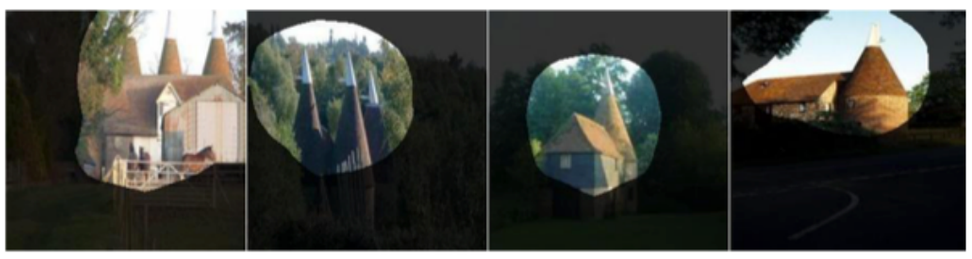
GoogLeNet

inception_4e unit 789 IoU=0.137

inception_4e unit 92 IoU=0.164

ResNet

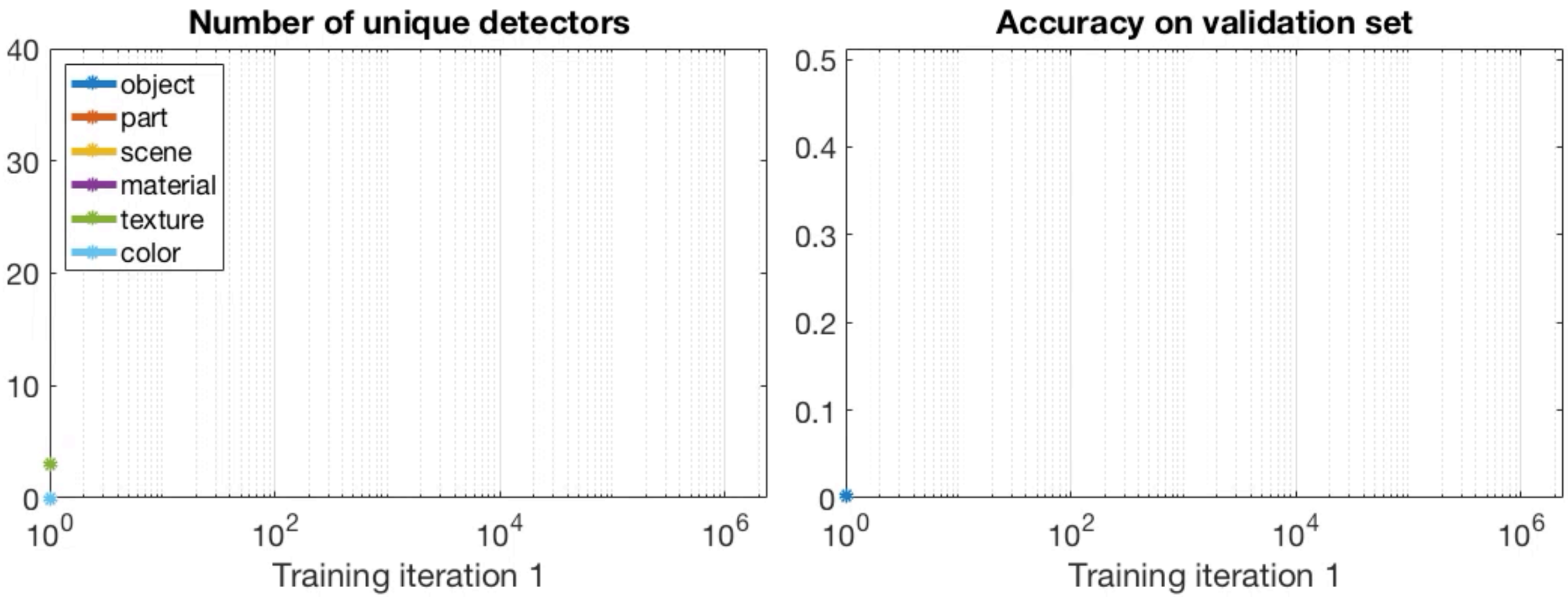
res5c unit 1410 IoU=0.142



res5c unit 1243 IoU=0.172

Slide credit: Bolei Zhou

Emergence of Interpretable Units during Training

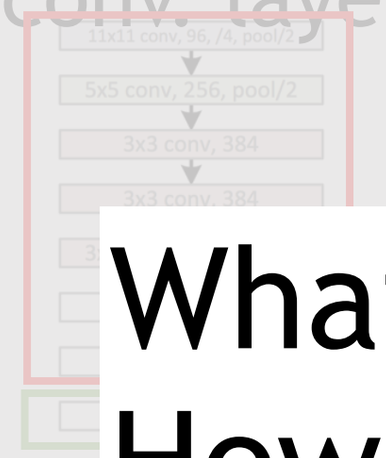


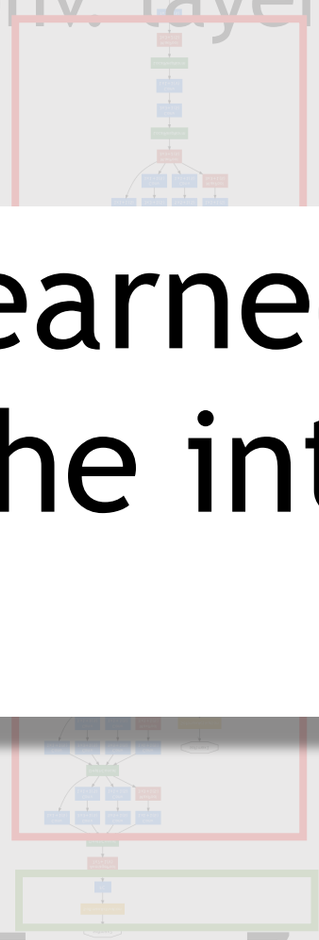
Slide credit: Bolei Zhou

Deep ConvNet for Visual Recognition

2012: AlexNet 2014: VGG 2015: GoogLeNet 2016: ResNet

5 conv. layers 16 conv. layers 22 conv. layers >100 conv. layers



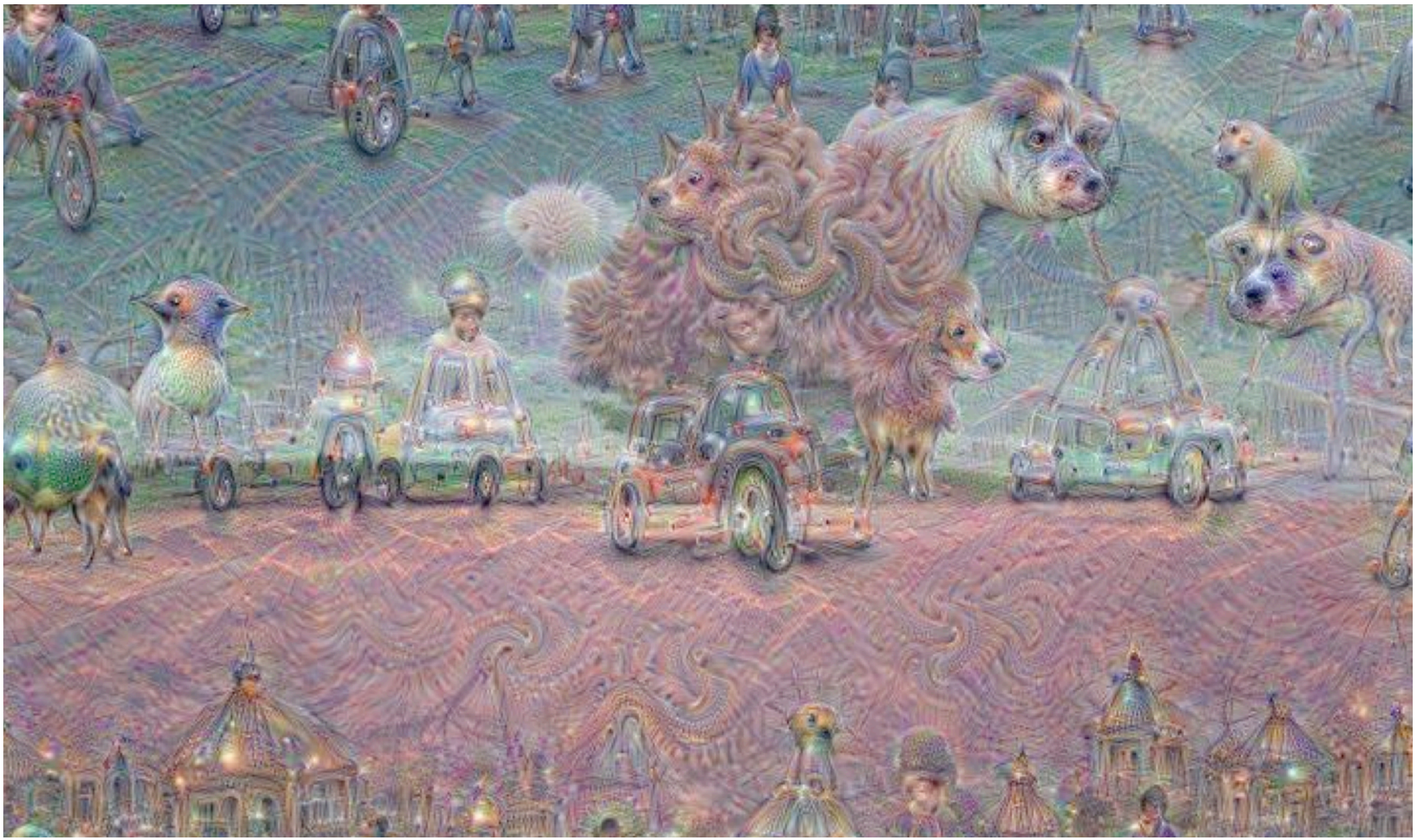


Error: 8.5% Error: 7.8% Error: 4.4%

What have been learned inside?
How to compare the internal representations?

Slide credit: Bolei Zhou

Next Class:
Art, Surprises, and More



Optimization

Appears to be a significant hurdle for training

Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

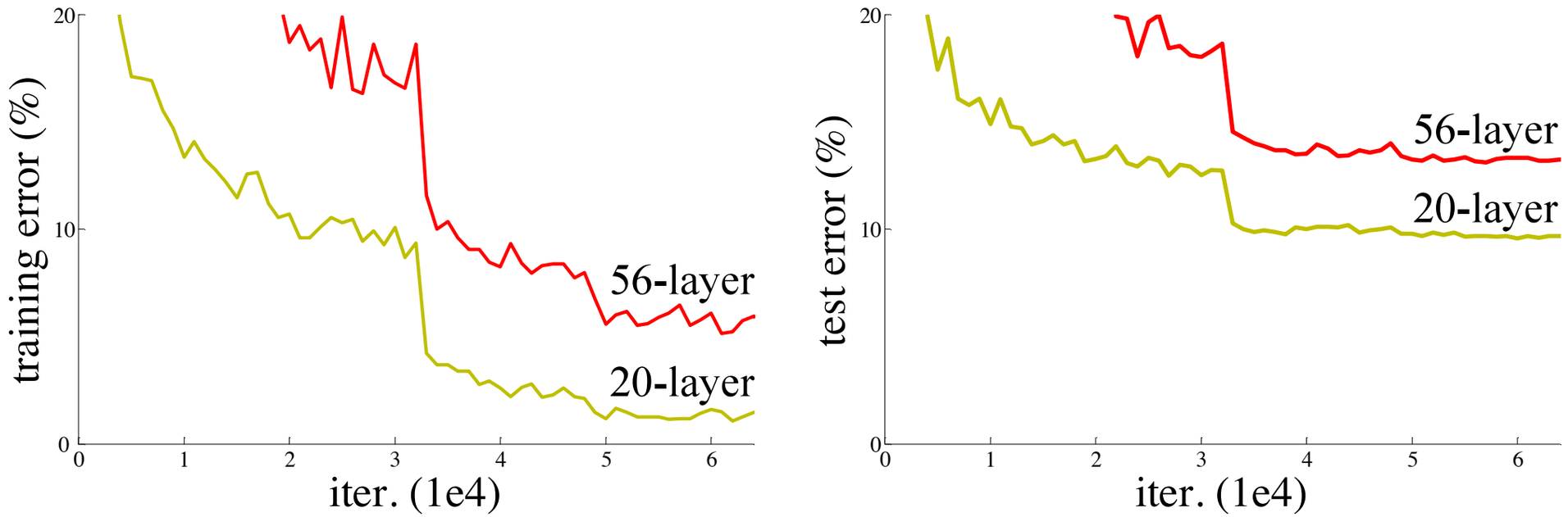
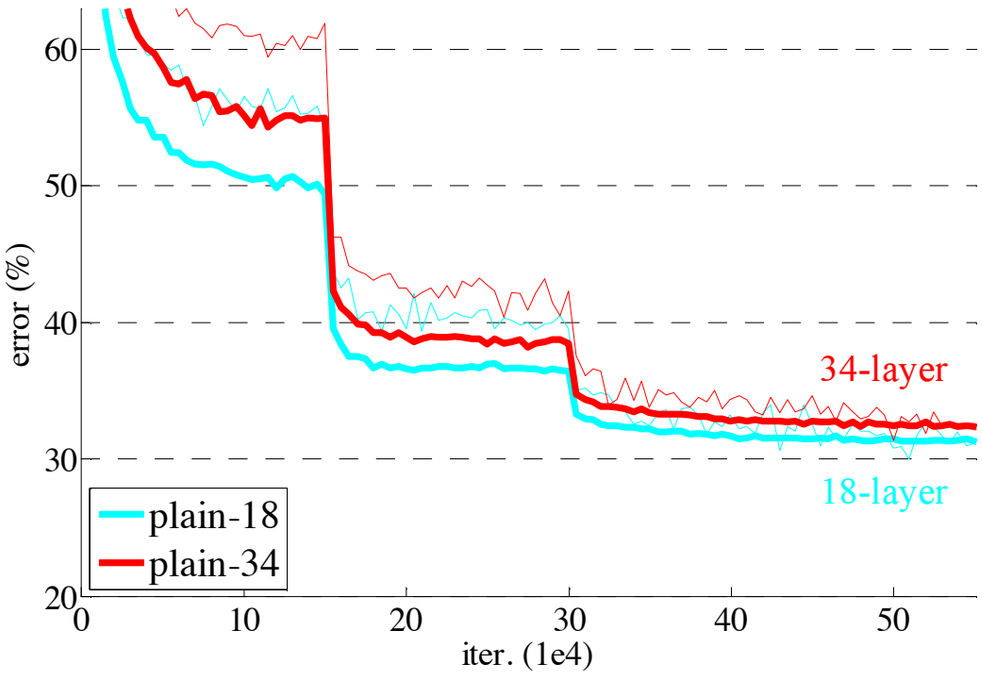


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Slide credit: Deva Ramanan

Intuition: bias deep models to behave like shallow models during learning



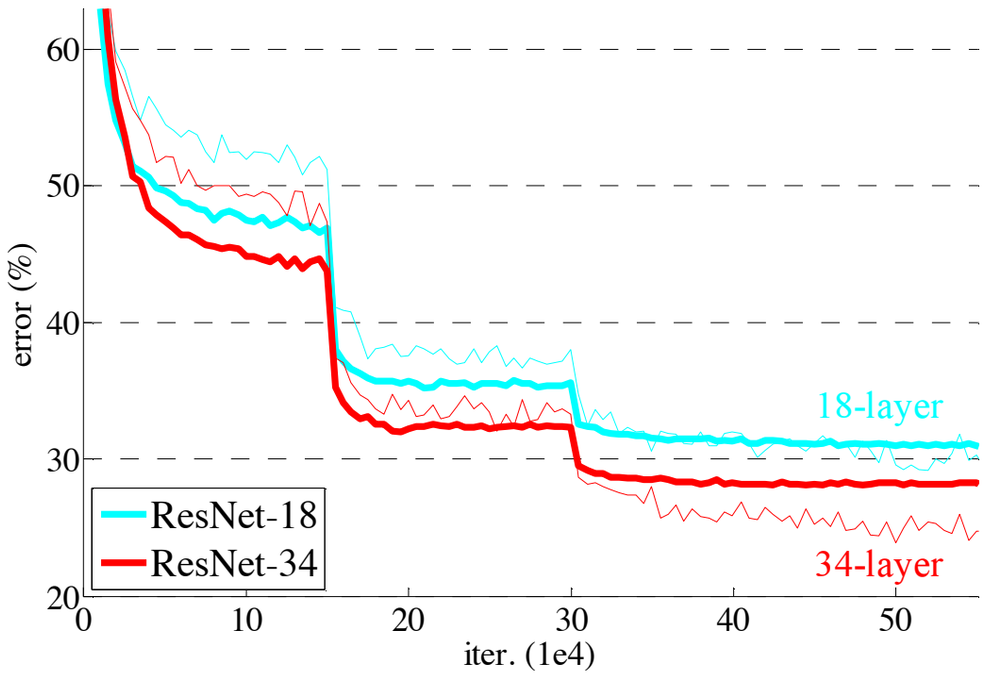
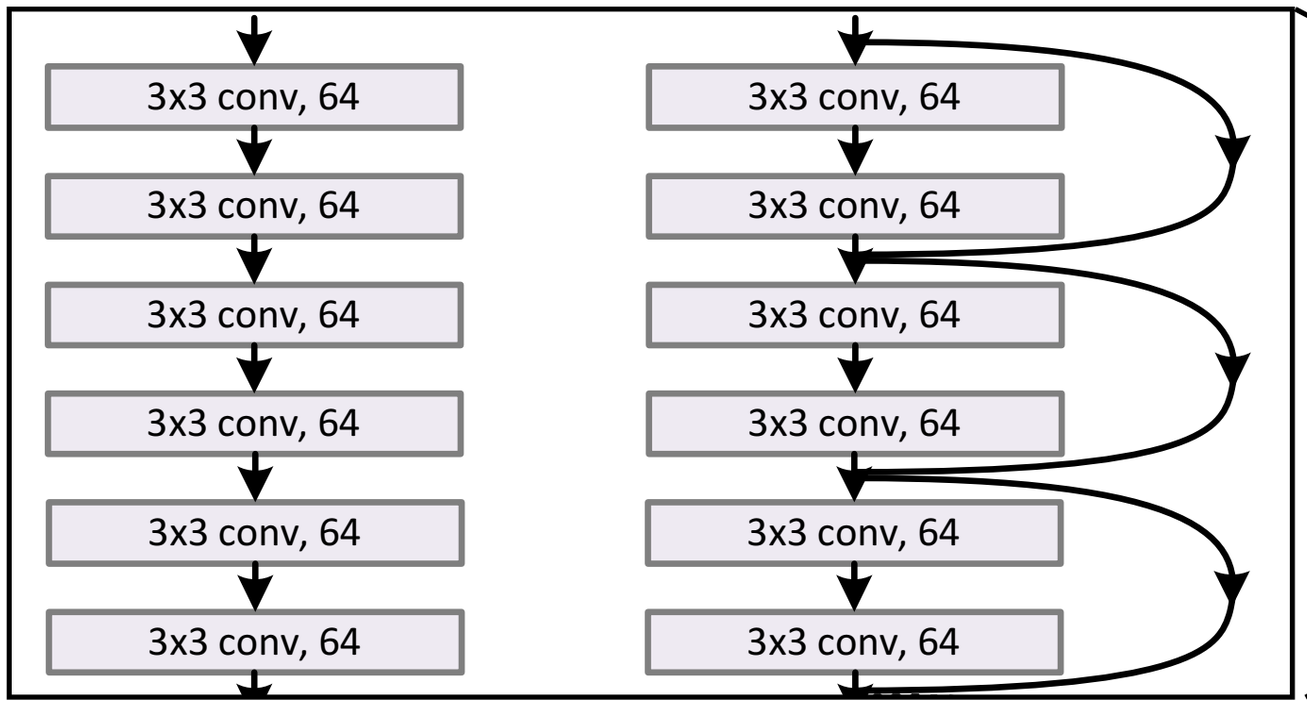
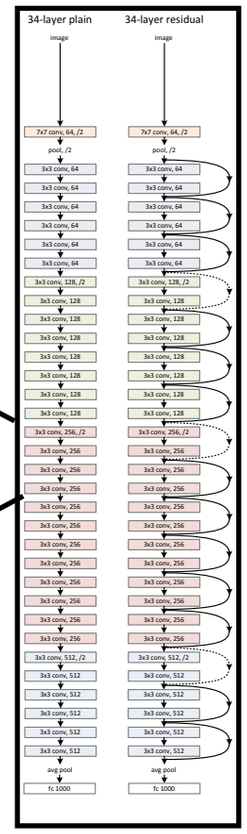


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.



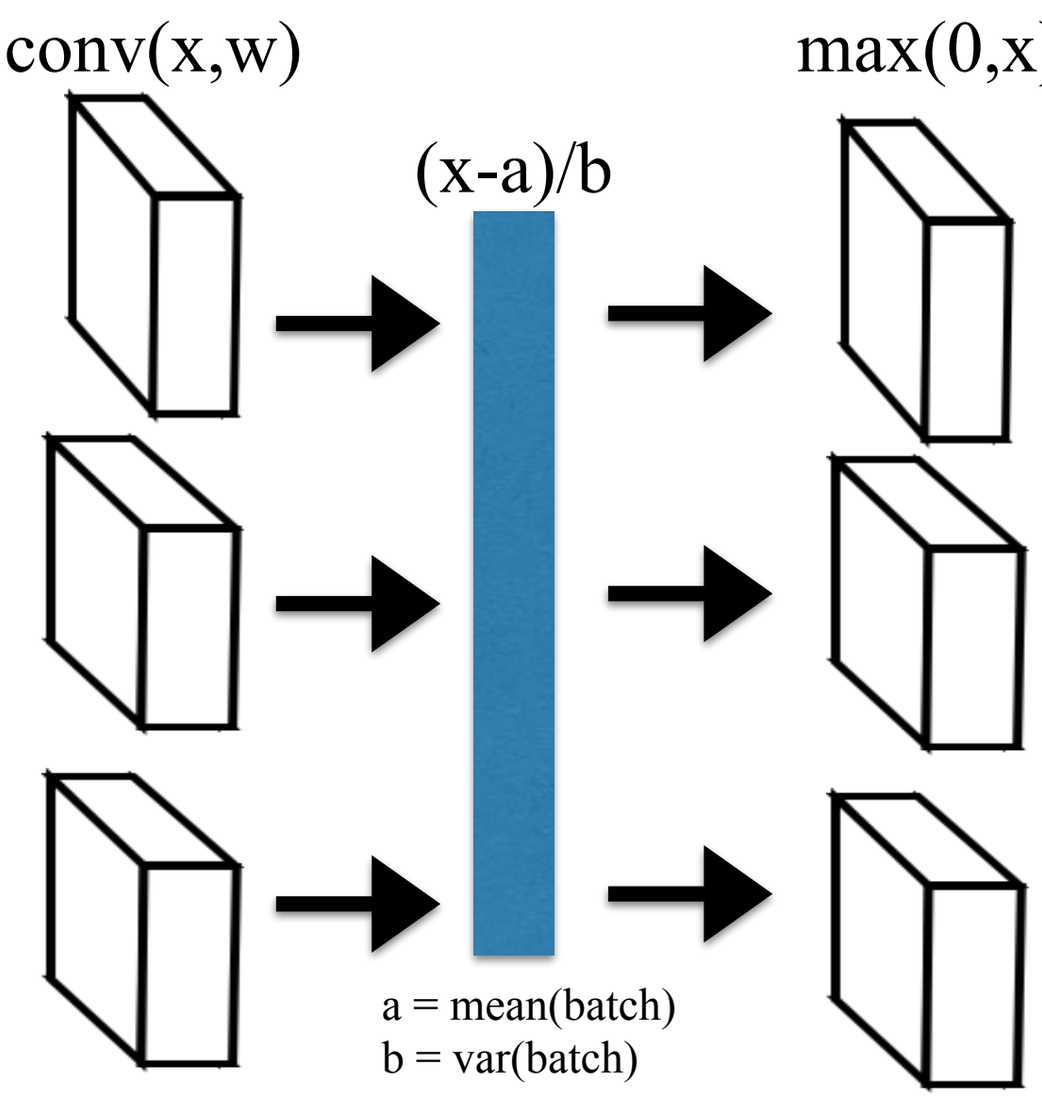


Slide credit: Deva Ramanan

Batch normalization

[Ioffe et al]

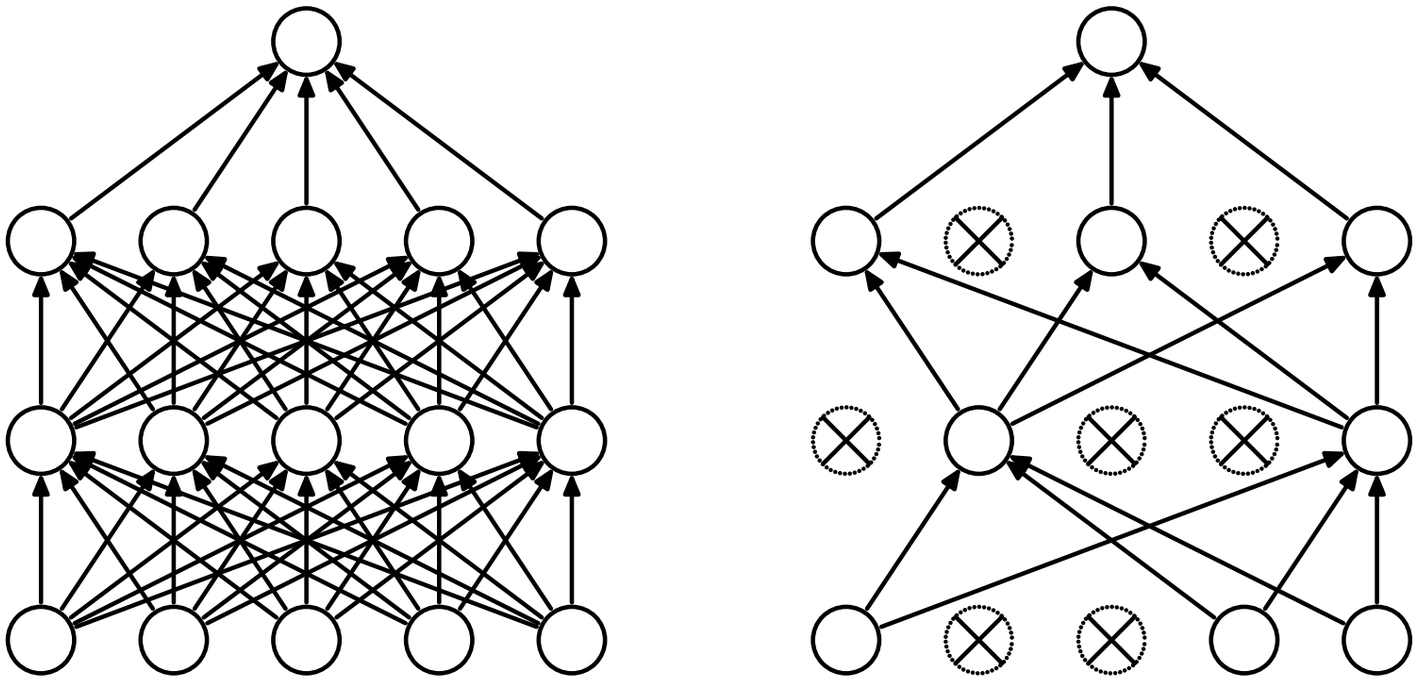
Intuition: build second-order behaviour into SGD by normalizing variables (zero-mean, identity covariance) before nonlinearity



Many (if not most) contemporary networks make use of this

Slide credit: Deva Ramanan

Drop-out regularization



(a) Standard Neural Net

(b) After applying dropout.

Intuition: we should really train a family of models with different architectures and average their predictions (c.f. model averaging from machine learning)

Practical implementation: learn a single “superset” architecture that randomly removes nodes (by randomly zero’ing out activations) during gradient updates

Slide credit: Deva Ramanan