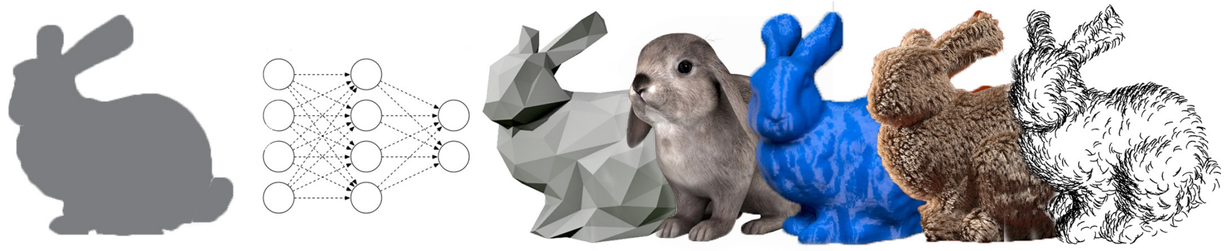


CreativeAI: Deep Learning for Computer Graphics

Deep Learning for Computer Graphics

Niloy Mitra UCL/Adobe

Iasonas Kokkinos UCL/Ariel AI

Paul Guerrero UCL/Adobe

Vladimir Kim Adobe

Nils Thuerey TU Munich

Leonidas Guibas Stanford University/FAIR

Timetable

		Niloy	Iasonas	Paul	Nils	Leonidas
Introduction	9:00	X				
Neural Network Basics	~9:15		X			
Supervised Learning in CG	~9:50	X				
Unsupervised Learning in CG	~10:20			X		
Learning on Unstructured Data	~10:55					X
Learning for Simulation/Animation	~11:35				X	
Discussion	12:05	X	X	X	X	X

Problems in Computer Graphics

- Feature detection (image features, point features) $\mathbb{R}^{m \times m} \rightarrow \mathbb{Z}$
- Denoising, Smoothing, etc. $\mathbb{R}^{m \times m} \rightarrow \mathbb{R}^{m \times m}$
- Embedding, Metric learning $\mathbb{R}^{m \times m, m \times m} \rightarrow \mathbb{R}^{d}$

analysis

- Rendering $\mathbb{R}^{m \times m} \rightarrow \mathbb{R}^{m \times m}$
- Animation $\mathbb{R}^{3m \times t} \rightarrow \mathbb{R}^{3m}$
- Physical simulation $\mathbb{R}^{3m \times t} \rightarrow \mathbb{R}^{3m}$
- Generative models $\mathbb{R}^{d} \rightarrow \mathbb{R}^{m \times m}$

synthesis

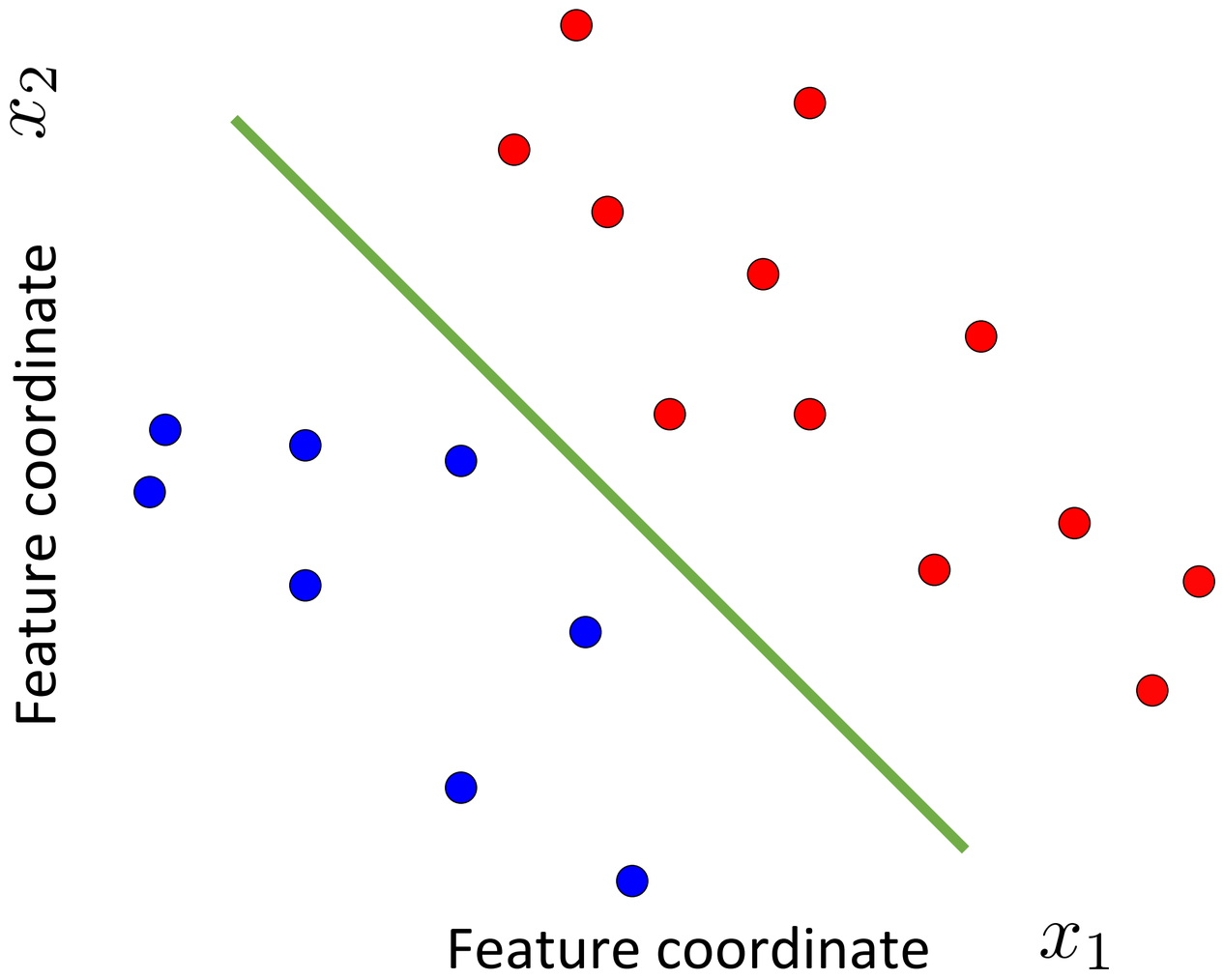
Goal: Learn a Parametric Function

$$f_\theta : \mathbb{X} \longrightarrow \mathbb{Y}$$

θ : function parameters $\mathbb{X}$: source domain $\mathbb{Y}$: target domain

Neural networks: choice of functional form

Linear classification problem



$$f_\theta : \mathbb{R}^n \longrightarrow \{0, 1\}$$

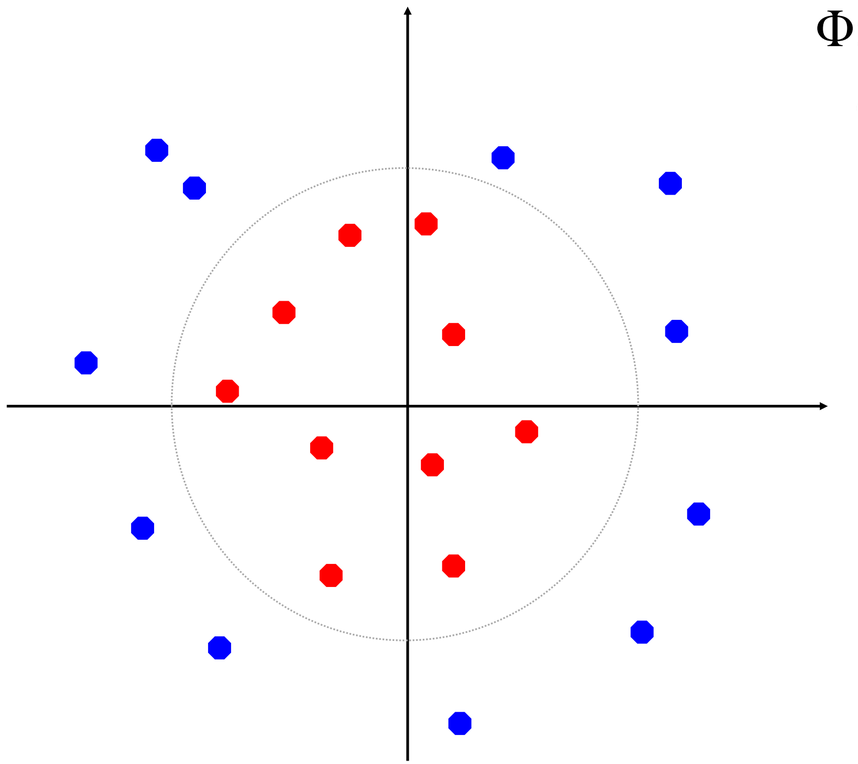
$$f_\theta(x) = \begin{cases} 1 & \text{if } wx + b \geq 0 \\ 0 & \text{if } wx + b < 0 \end{cases}$$

$$\theta = \{w, b\}$$

Each data point has a class label:

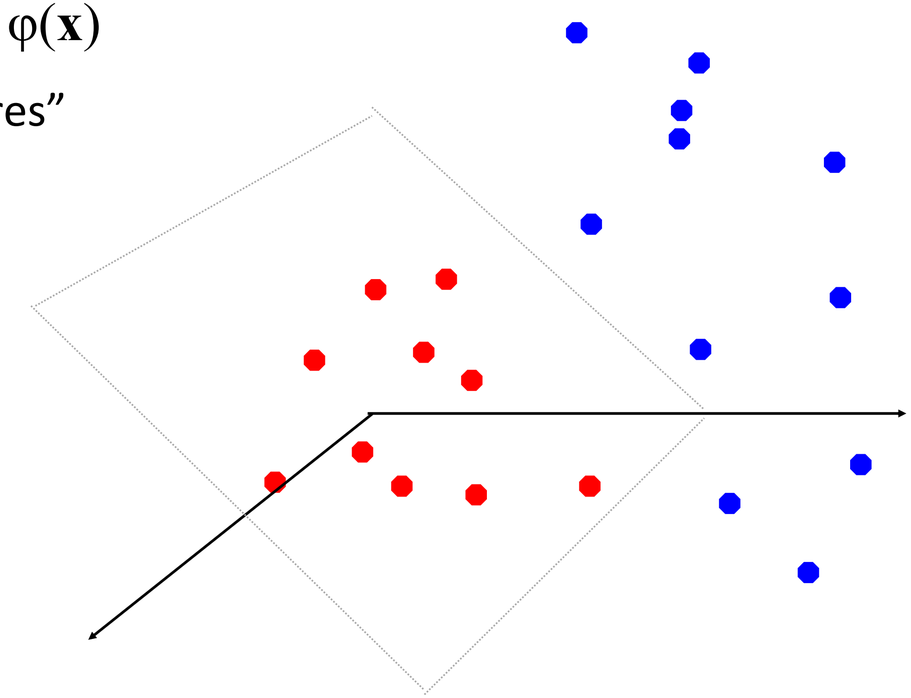
$$y^i = \begin{cases} 1 & (\bullet) \\ 0 & (\bullet) \end{cases}$$

Beyond linear functions



$\Phi: \mathbf{x} \rightarrow \varphi(\mathbf{x})$

"features"

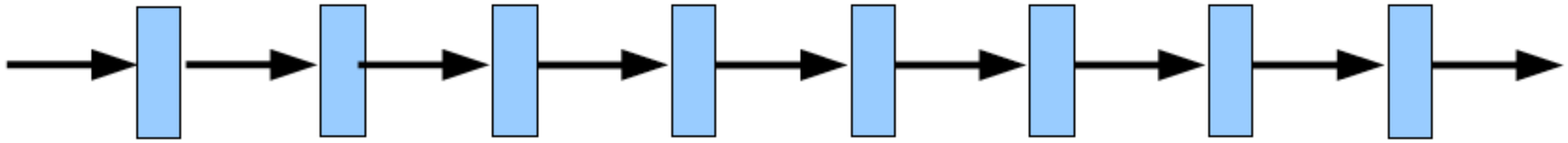


Not linearly-separable in R^2

Linearly-separable in R^3

Neural networks: learn the features!

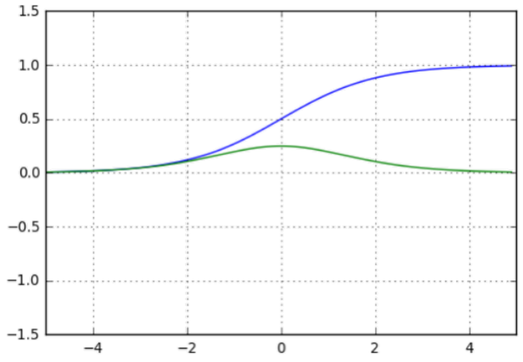
Neural networks: function composition



basic building block

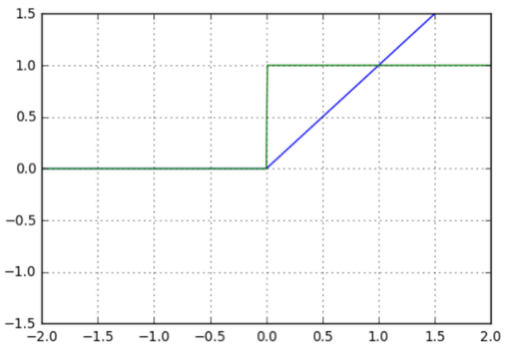
‘Neuron’: Cascade of Linear and Nonlinear Function

function
derivative



Sigmoidal (“logistic”)

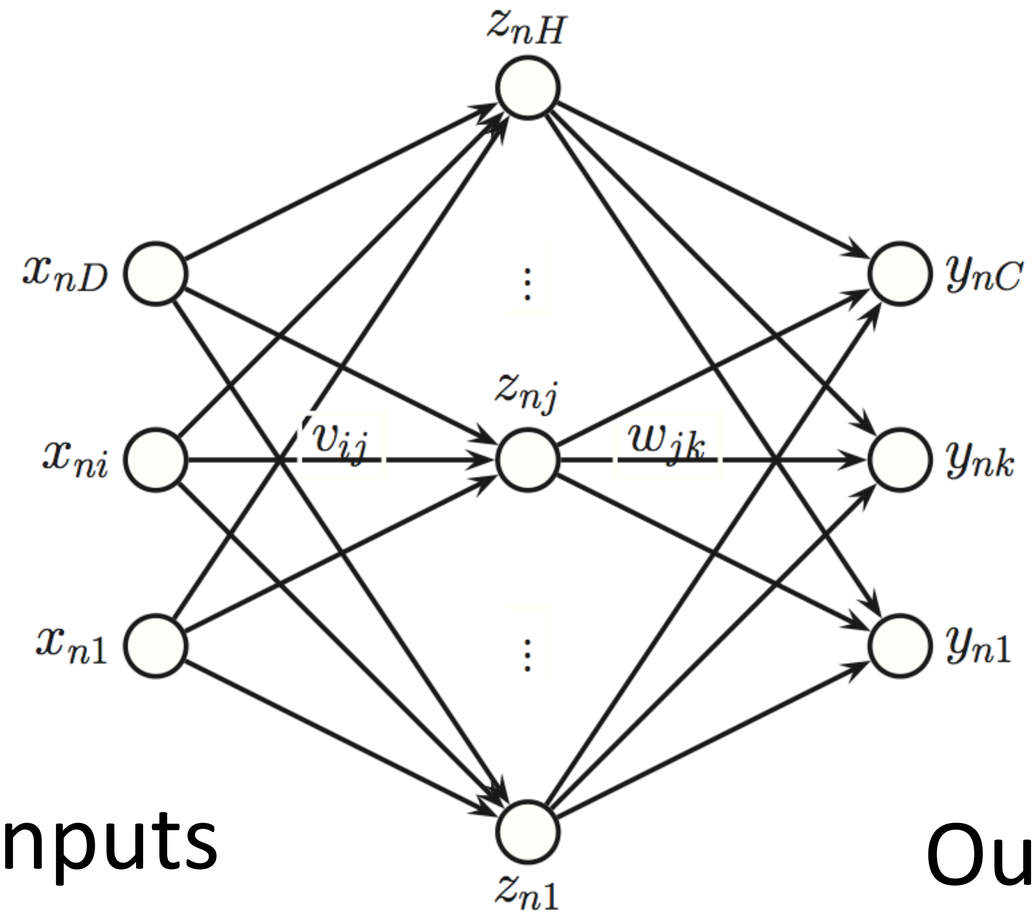
$$g(a) = \frac{1}{1 + \exp(-a)}$$



Rectified Linear Unit (RELU)

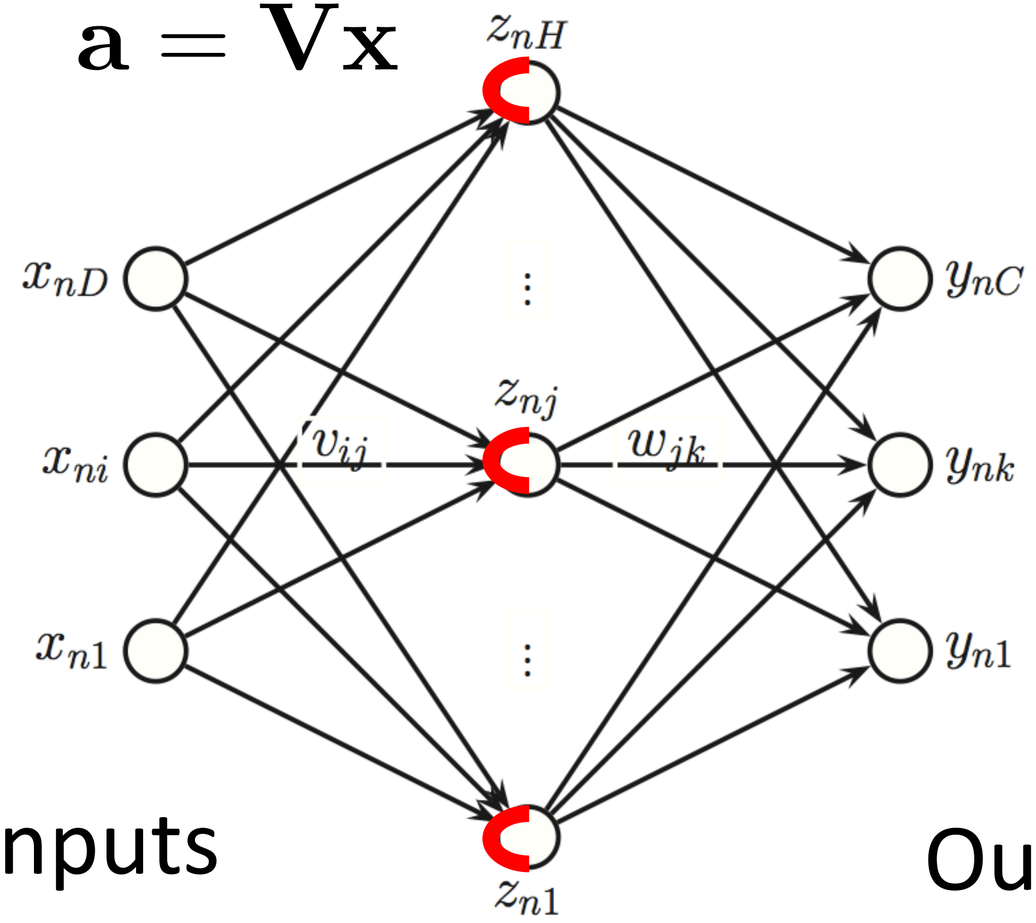
$$g(a) = \max(0, a)$$

A two-layer neural network



A two-layer neural network

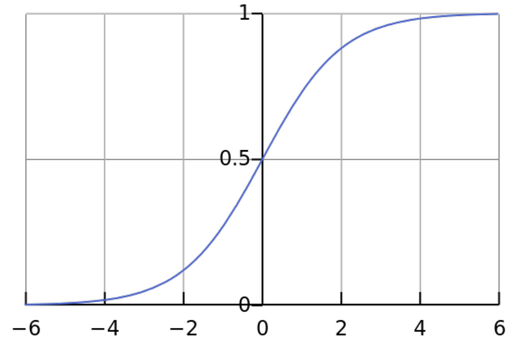
$$\mathbf{a} = \mathbf{V}\mathbf{x}$$

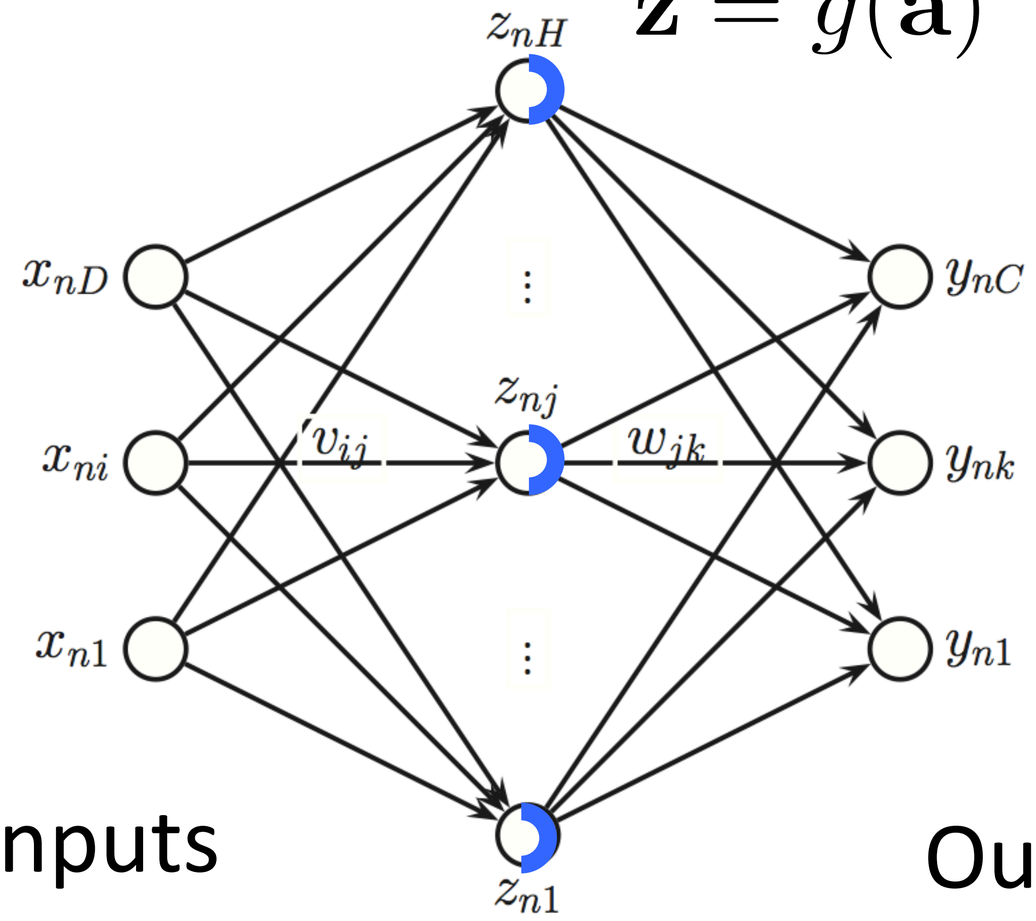


A two-layer neural network

$$\mathbf{z} = g(\mathbf{a})$$

$$z_k = \frac{1}{1 + \exp(-a_k)}$$





Inputs

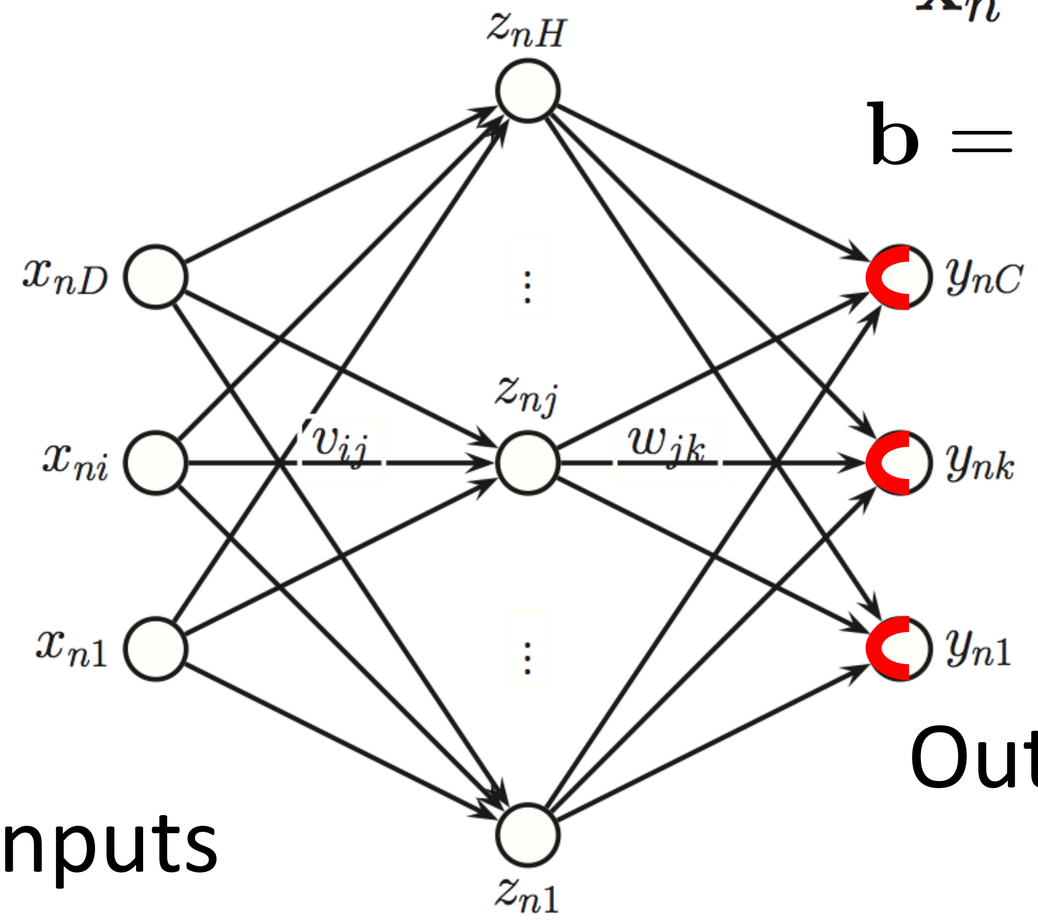
Hidden layer

Outputs

A two-layer neural network

$$\mathbf{x}_n \xrightarrow{\mathbf{V}} \mathbf{a}_n \xrightarrow{g} \mathbf{z}_n \xrightarrow{\mathbf{W}} \mathbf{b}_n \xrightarrow{h} \hat{\mathbf{y}}_n$$

$$\mathbf{b} = \mathbf{W}\mathbf{z}$$



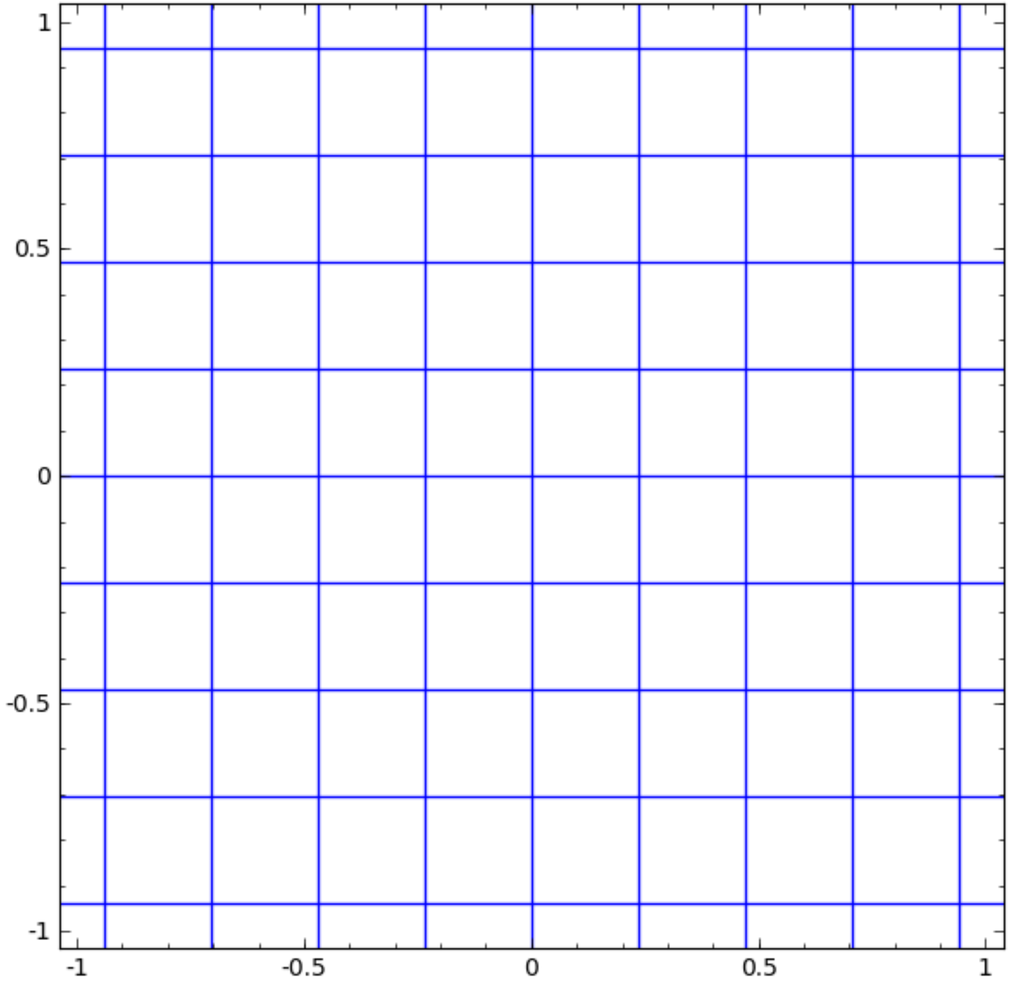
Inputs

Hidden layer

Outputs

Nonlinear mapping $\mathbb{R}^2 \rightarrow \mathbb{R}^2$

Evolution of isocontours as parameters change



$$y_1 = g(w_{1,1}x_1 + w_{1,2}x_2 + w_{1,3})$$
$$y_2 = g(w_{2,1}x_1 + w_{2,2}x_2 + w_{2,3})$$

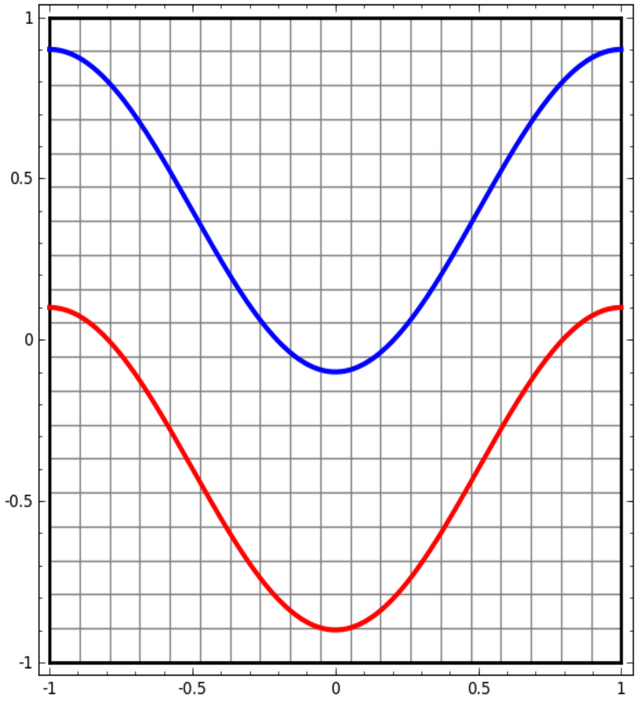
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$$

$$\mathbf{y} = g(\mathbf{W}\mathbf{x})$$

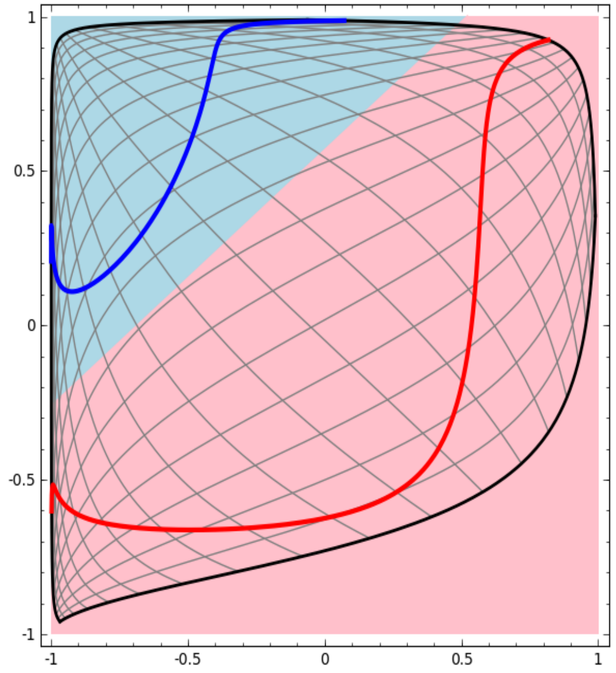
http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

From non-separable to linearly separable

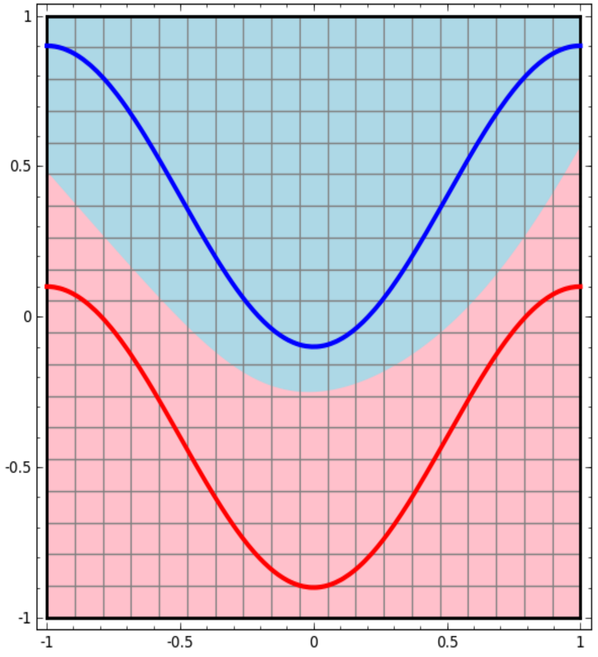
Non-linearly separable data



Data mapped to learned space

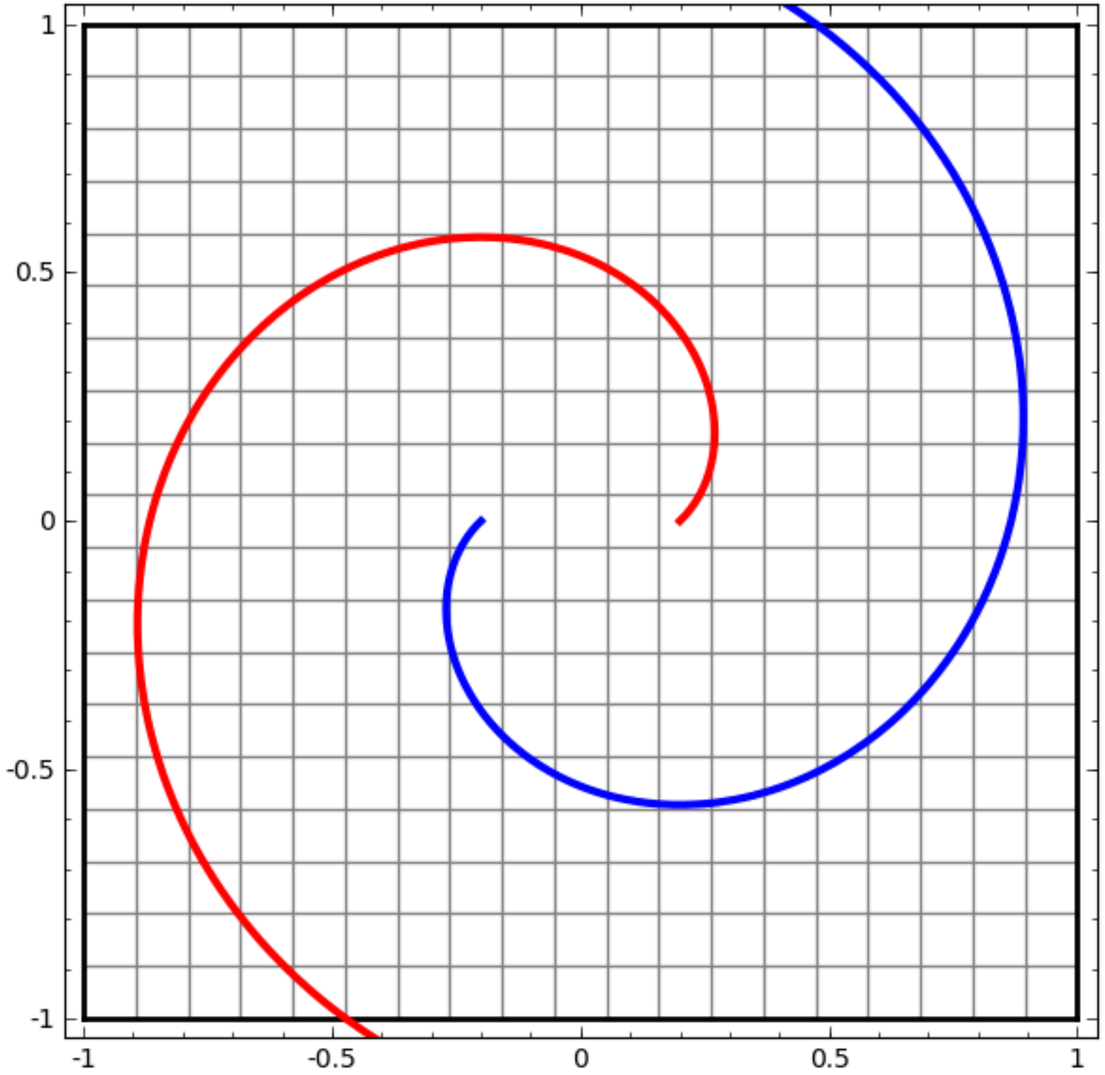


Decision function



http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Linearizing a 2D classification task (4 hidden layers)



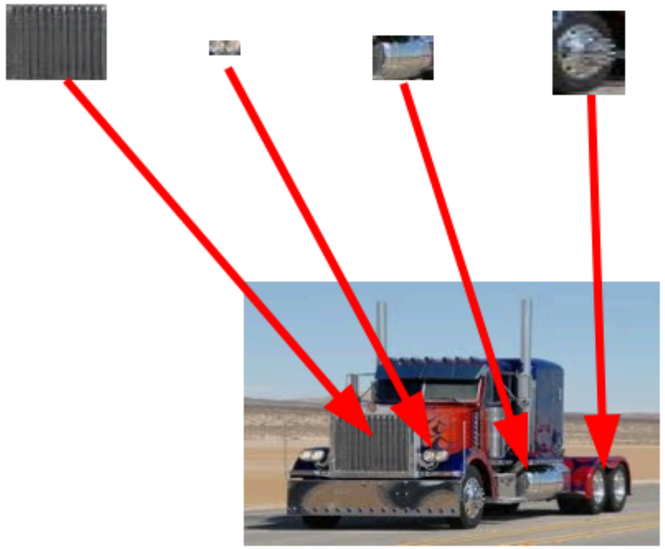
http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Hidden Layers: intuitively, what do they do?

Intuition: learn “dictionary” for objects

“Distributed representation”:
represent (and classify) objects by mixing & mashing reusable parts

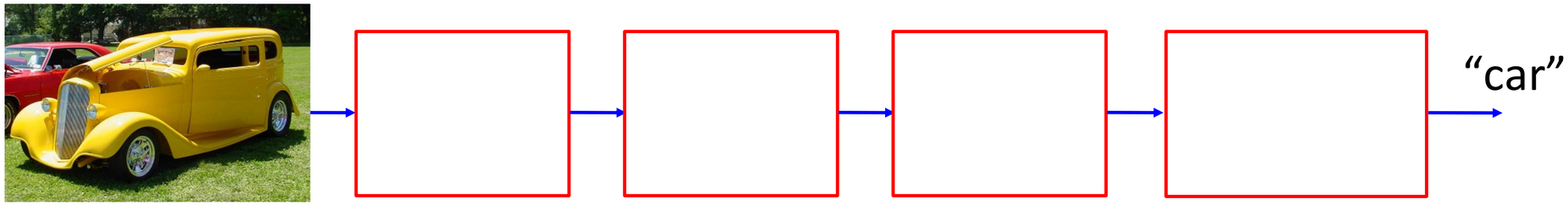
[0 0 **1** 0 0 0 0 **1** 0 0 **1** **1** 0 0 **1** 0 …] truck feature



Creativ

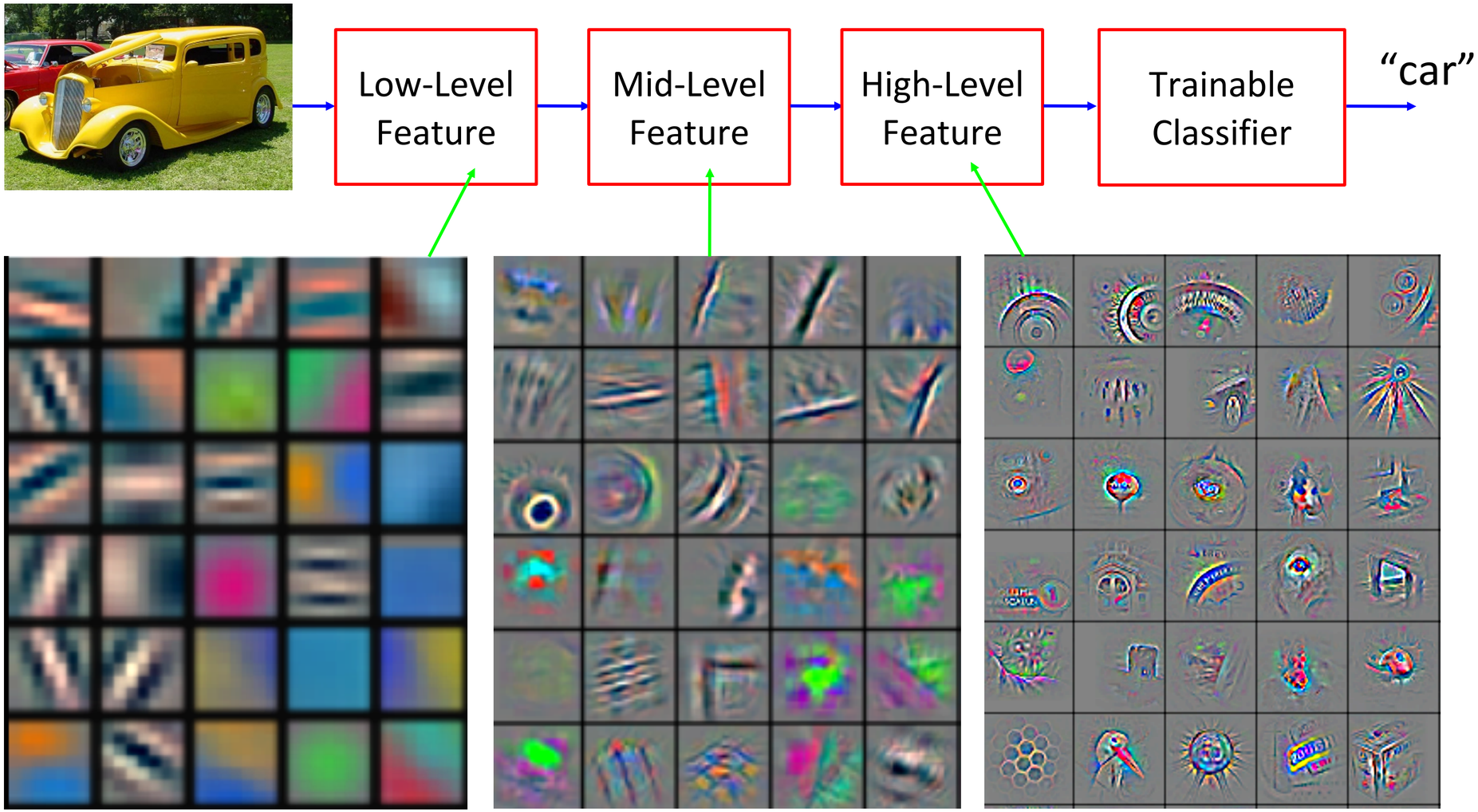
Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Deep Learning = Hierarchical Compositionality



Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Deep Learning = Hierarchical Compositionality



Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Training and Optimization

Training Goal

Our network implements a parametric function:

$$f_\theta : \mathbb{X} \longrightarrow \mathbb{Y} \qquad \hat{y} = f(x;\theta)$$

During training, we search for parameters that minimize a loss:

$$\min_\theta L_f(\theta)$$

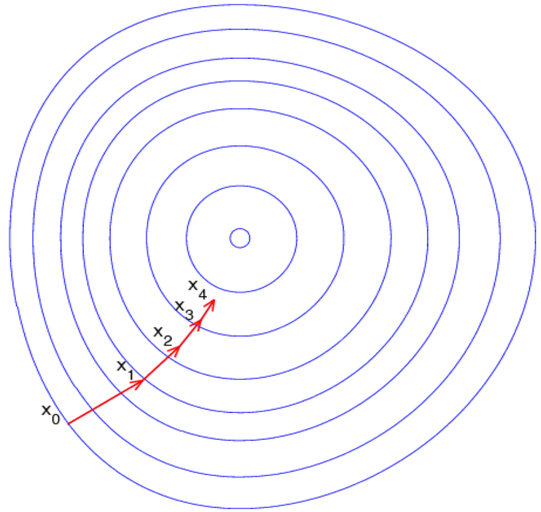
Example: L2 regression loss given target (x^i, y^i) pairs :

$$L_f(\theta) = \sum_i \| f(x^i;\theta) - y^i \|_2^2$$

Gradient Descent Minimization Method

Initialize: θ_0

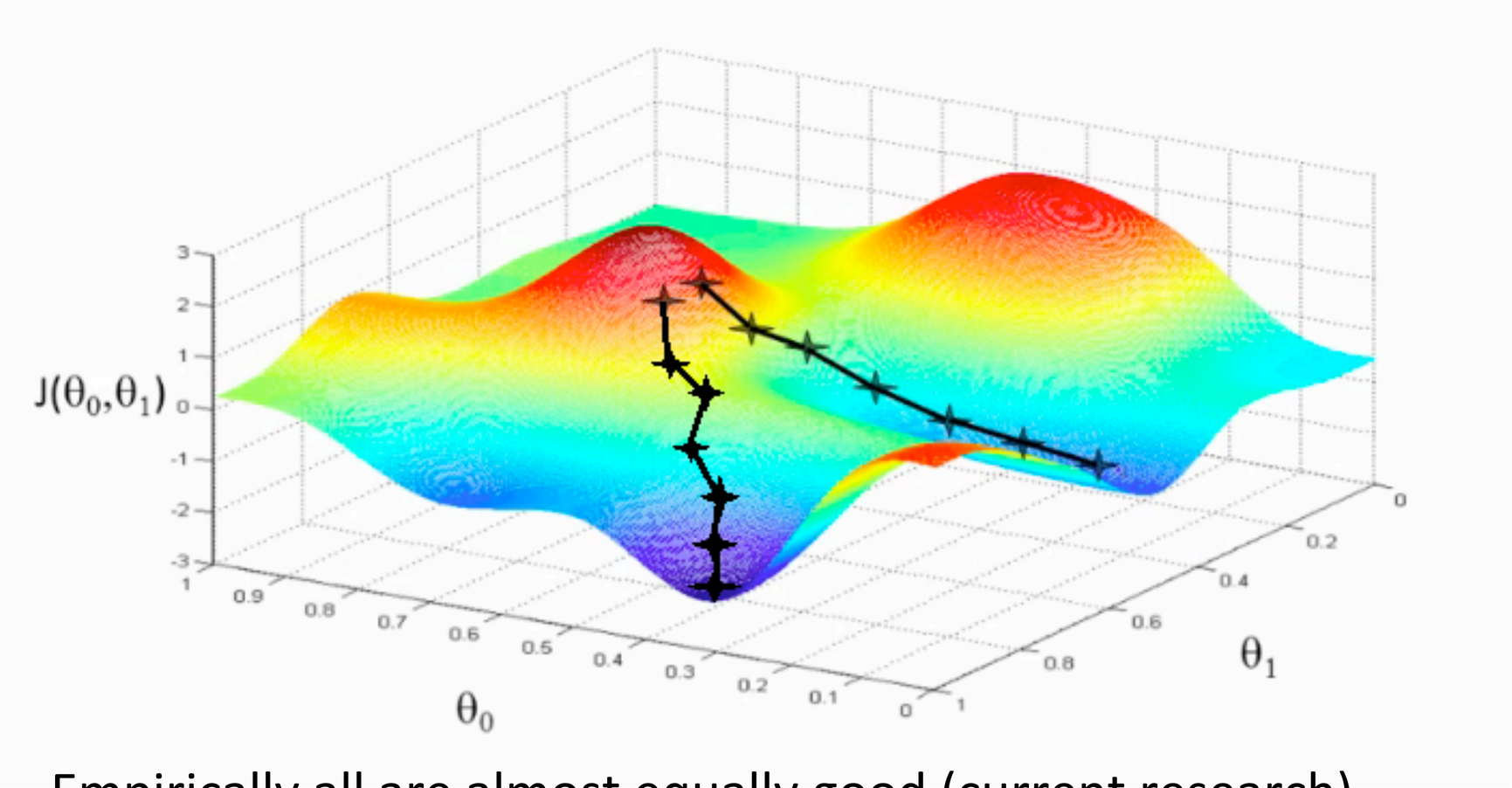
Update: $\theta_{i+1} = \theta_i - \alpha \nabla f(\theta_i)$



We can always make it converge for a convex function

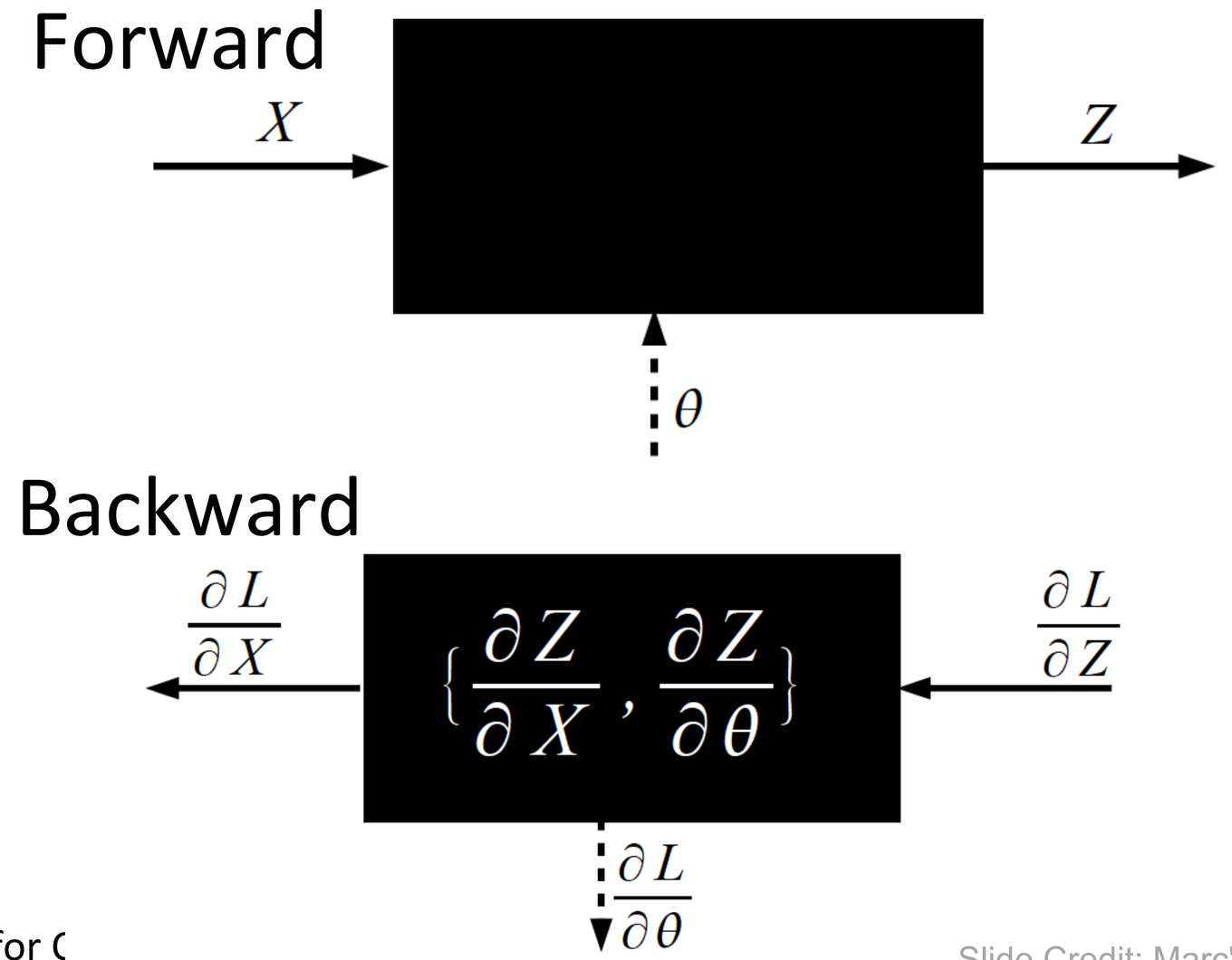
Neural network's loss: non convex objective

Multiple Local Minima, based on initialization



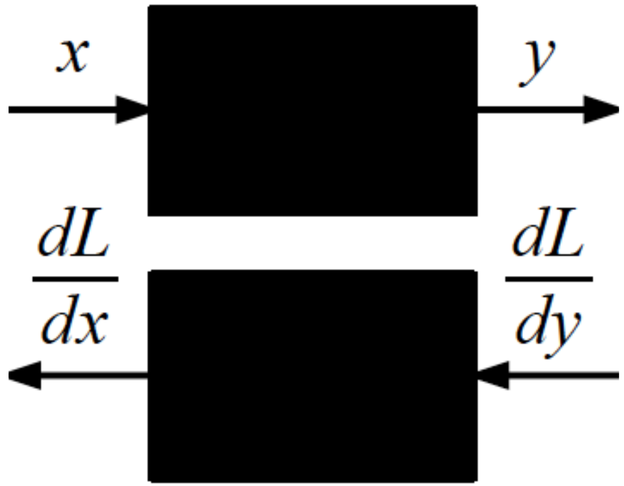
Empirically all are almost equally good (current research)

All you need is gradients



Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Chain Rule

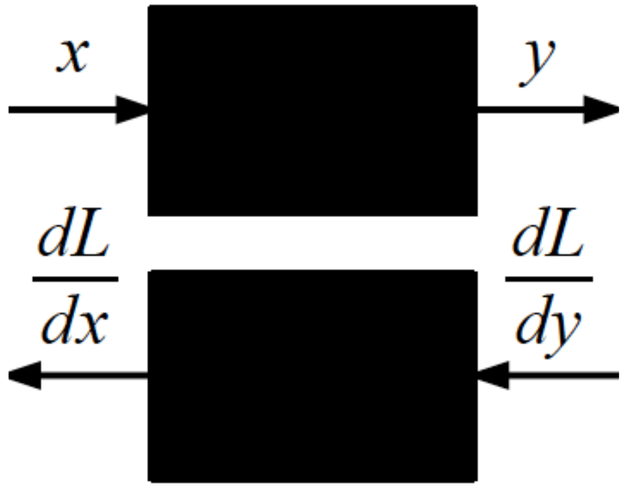


Given $y(x)$ and dL/dy,

What is dL/dx ?

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Chain Rule



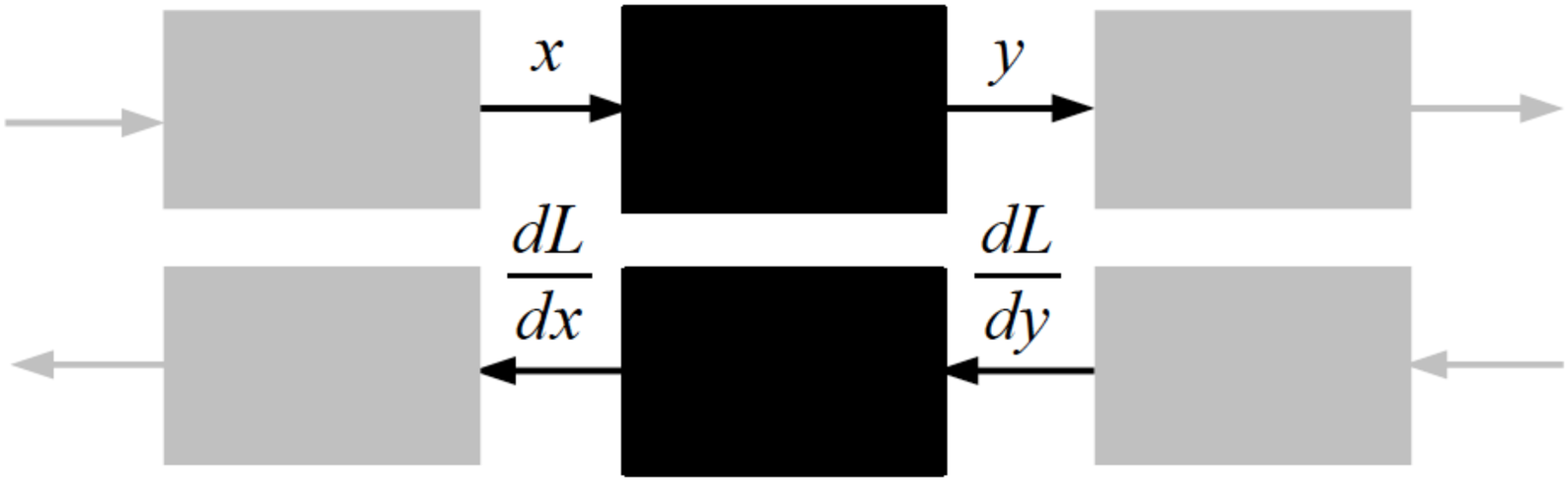
Given $y(x)$ and dL/dy,

What is dL/dx ?

$$\frac{dL}{dx} = \frac{dL}{dy} \cdot \frac{dy}{dx}$$

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Chain Rule



Given $y(x)$ and dL/dy,

What is dL/dx ?

$$\frac{dL}{dx} = \frac{dL}{dy} \cdot \frac{dy}{dx}$$

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

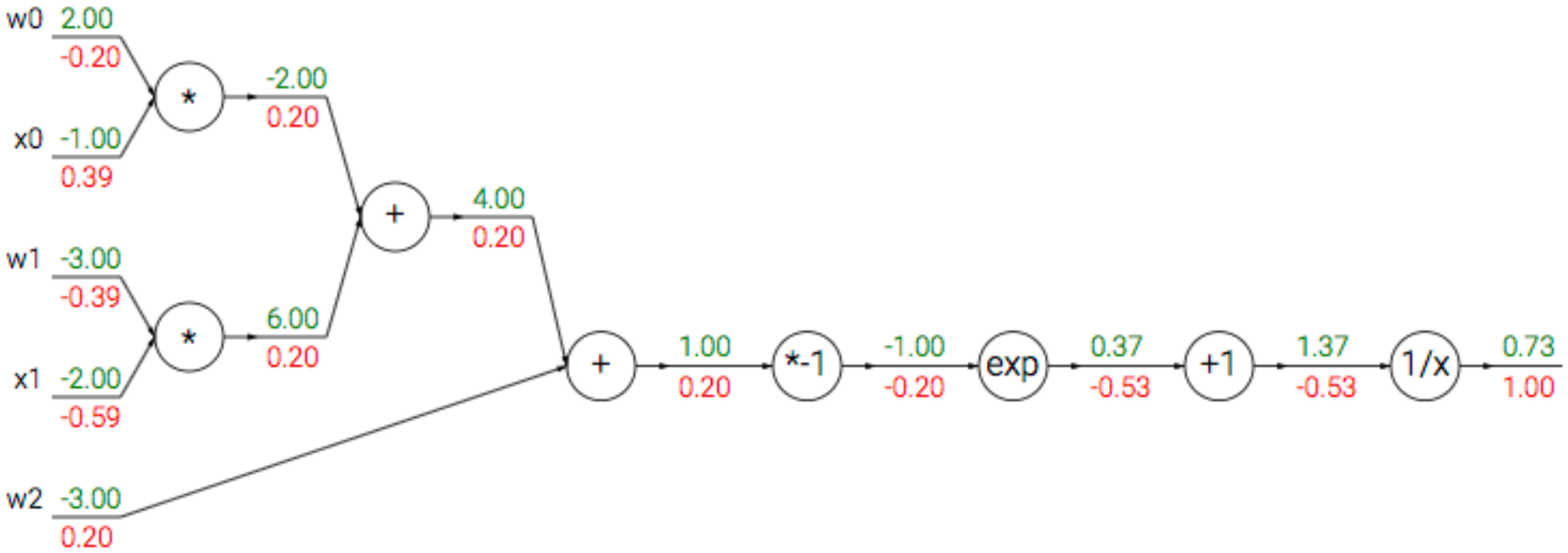
Toy example: single sigmoidal unit

$$f(w,x) = \frac{1}{1+\exp(-(w_0x_0 + w_1x_1 + w_2))}$$

Composition of differentiable blocks:

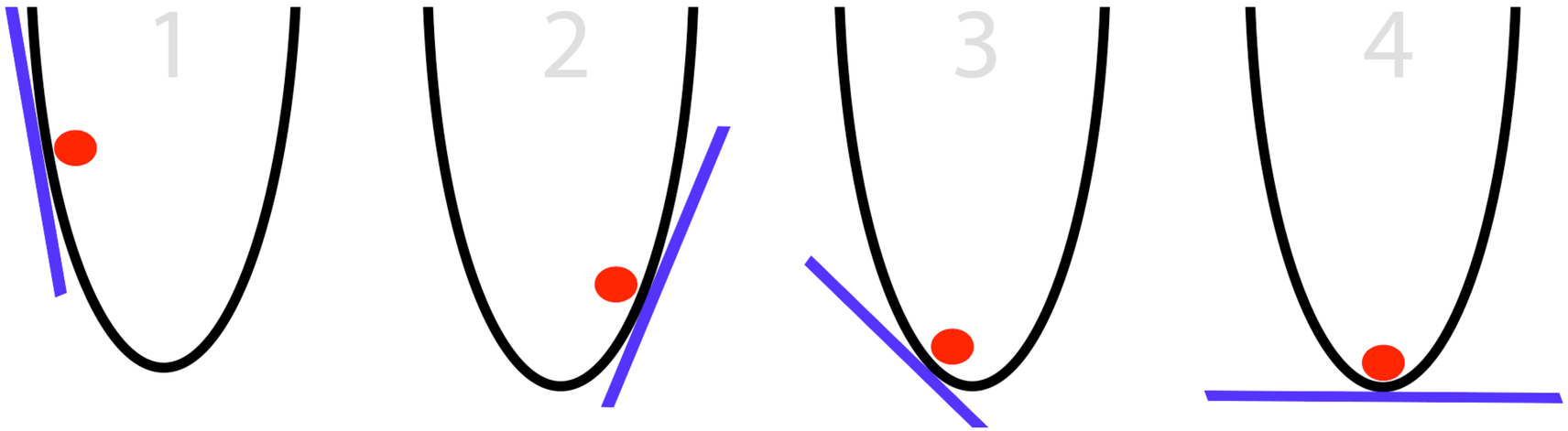
$$\begin{aligned}
f(x) = \frac{1}{x} \quad &\rightarrow \quad f'(x) = -\frac{1}{x^2} \\
f_c(x) = c + x \quad &\rightarrow \quad f'(x) = 1 \\
f(x) = e^x \quad &\rightarrow \quad f'(x) = e^x \\
f_a(x) = ax \quad &\rightarrow \quad f'(x) = a
\end{aligned}$$

Computation graph & automatic differentiation

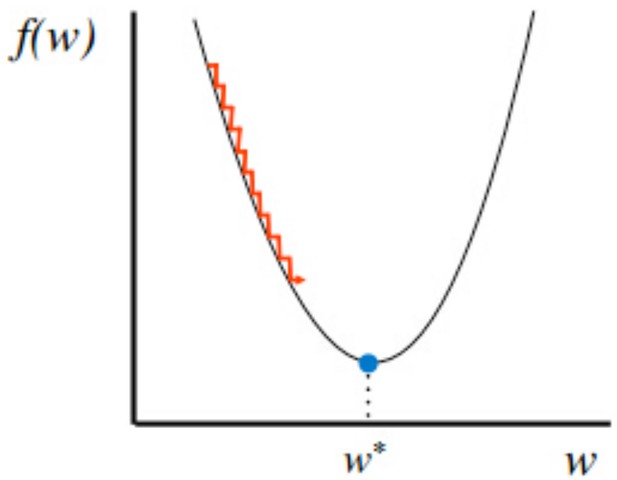


Slide Credit: Justin Johnson

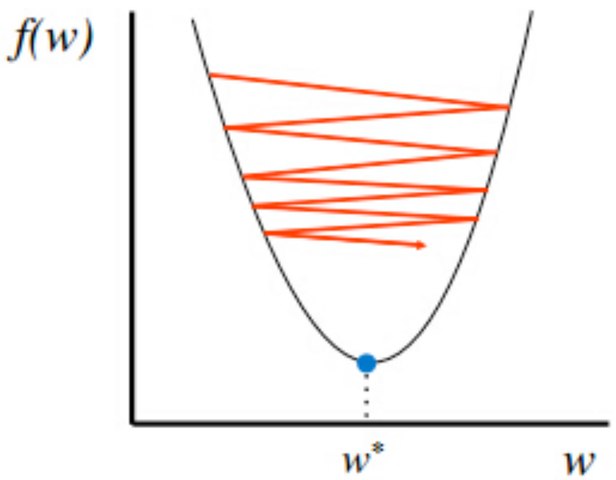
Parameter Updates & Convergence



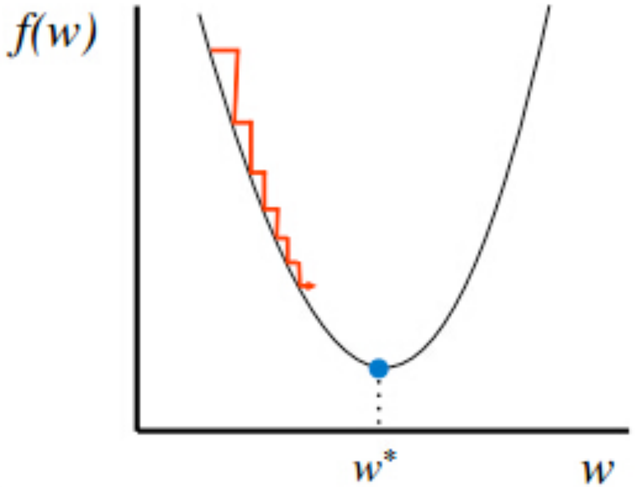
(S)GD with adaptable stepsize



Too small: converge very slowly



Too big: overshoot and even diverge

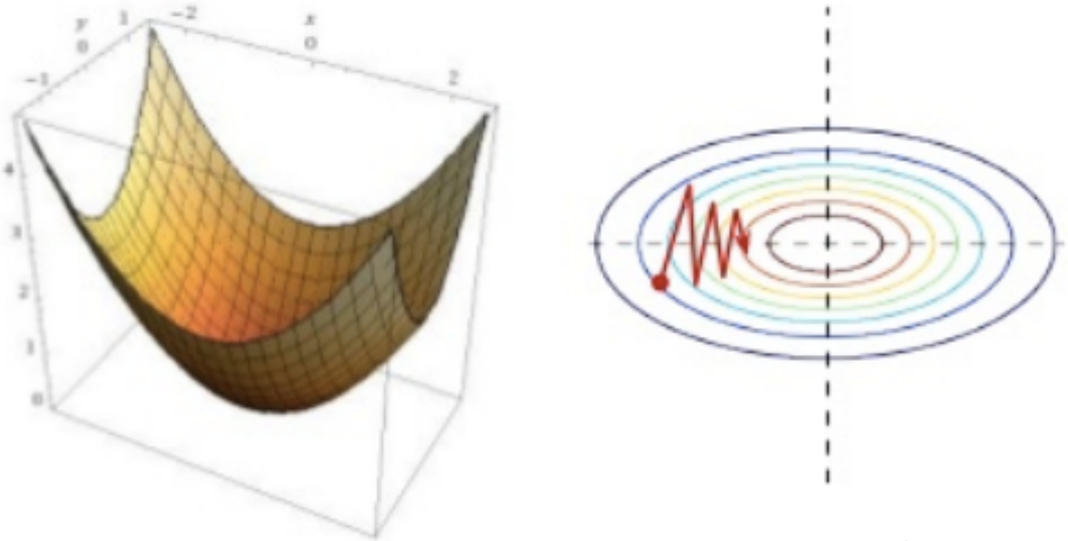


Reduce size over time

e.g $\epsilon_t = \frac{c}{t}$

.

(S)GD with momentum



Main idea: retain long-term trend of updates, drop oscillations

(S)GD $$\mathbf{W}_{t+1} = \mathbf{W}_t - \epsilon_t \nabla_{\mathbf{W}} L(\mathbf{W})$$

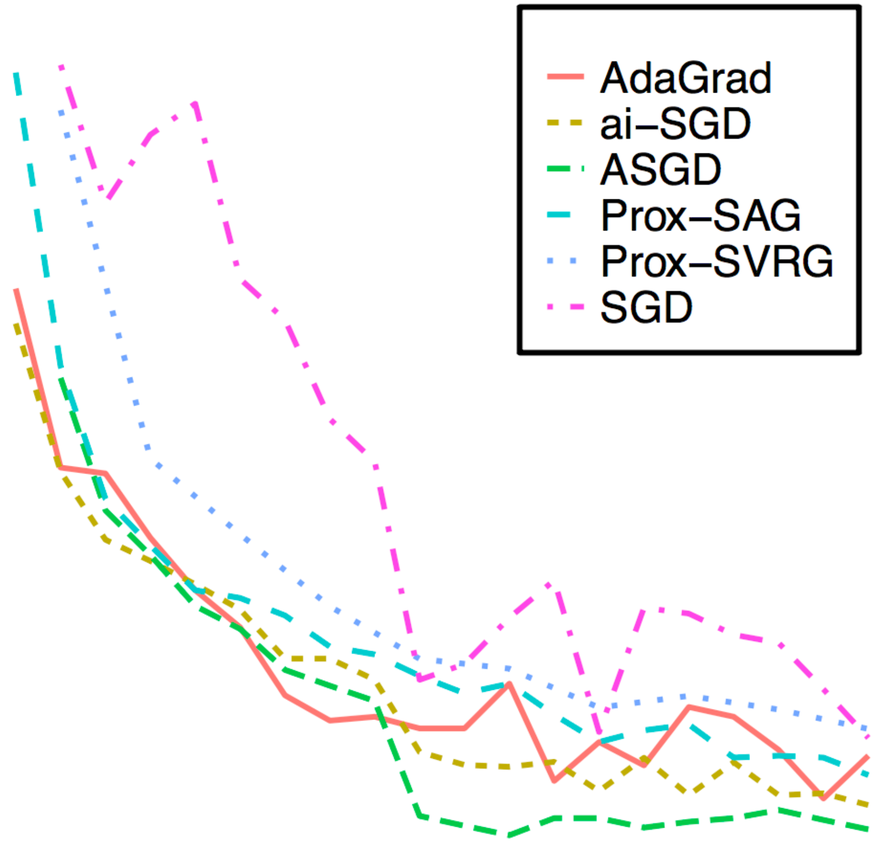
(S)GD + momentum

$$\mathbf{V}_{t+1} = \mu \mathbf{V}_t + (1-\mu) \nabla_{\mathbf{W}} L(\mathbf{W}_t)$$

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \epsilon_t \mathbf{V}_{t+1}$$

Step-size Selection & Optimizers: research problem

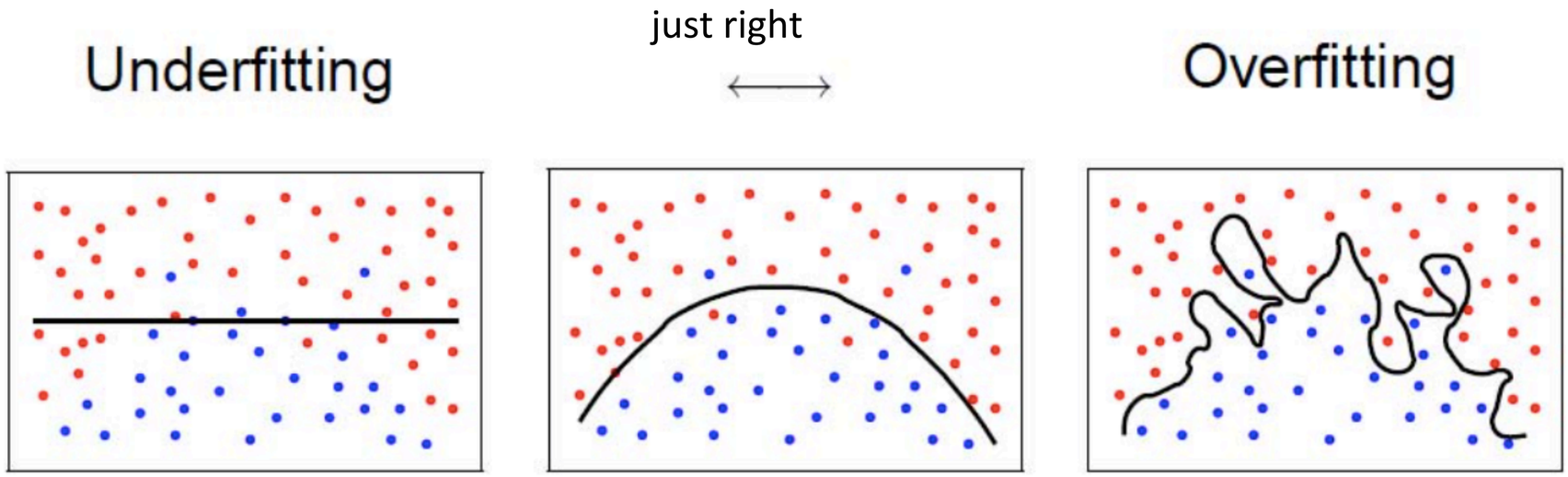
- Nesterov's Accelerated Gradient (NAG)
- R-prop
- AdaGrad
- RMSProp
- AdaDelta
- Adam
- …



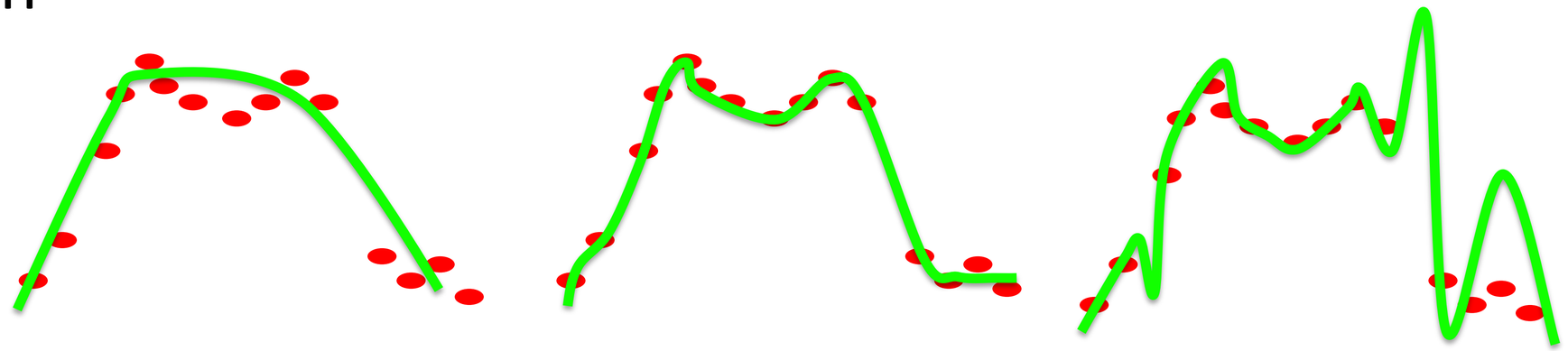
Overfitting and Regularization

Overfitting, in images

Classification



Regression



Slide Credit: Rob Fergus

1980’s solution: l2 Regularization

Regularization: preference for small parameters

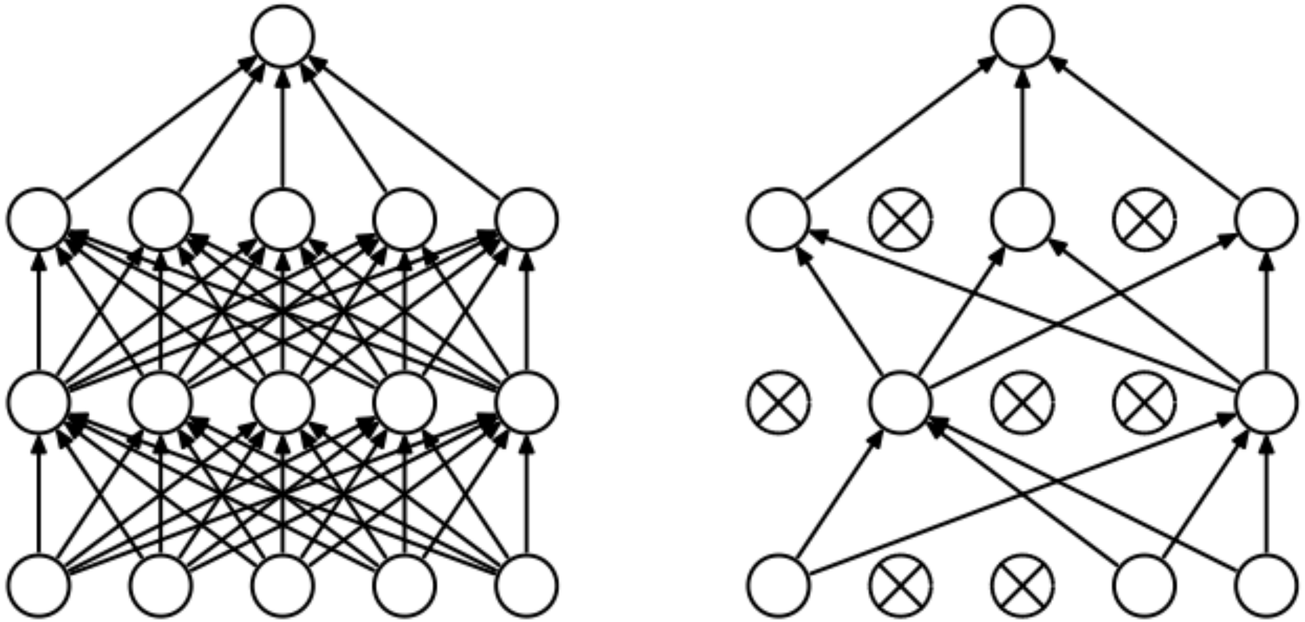
$$L(\mathbf{W}) = \frac{1}{N}\sum_{i=1}^{N} l(\mathbf{y}^i, \hat{\mathbf{y}}^i) + \sum_l \lambda_l \sum_{k,m} (\mathbf{W}_{k,m}^l)^2$$

Per-sample loss

Per-layer regularization

Integration with gradient-descent: “weight decay”

2010’s solution: Dropout

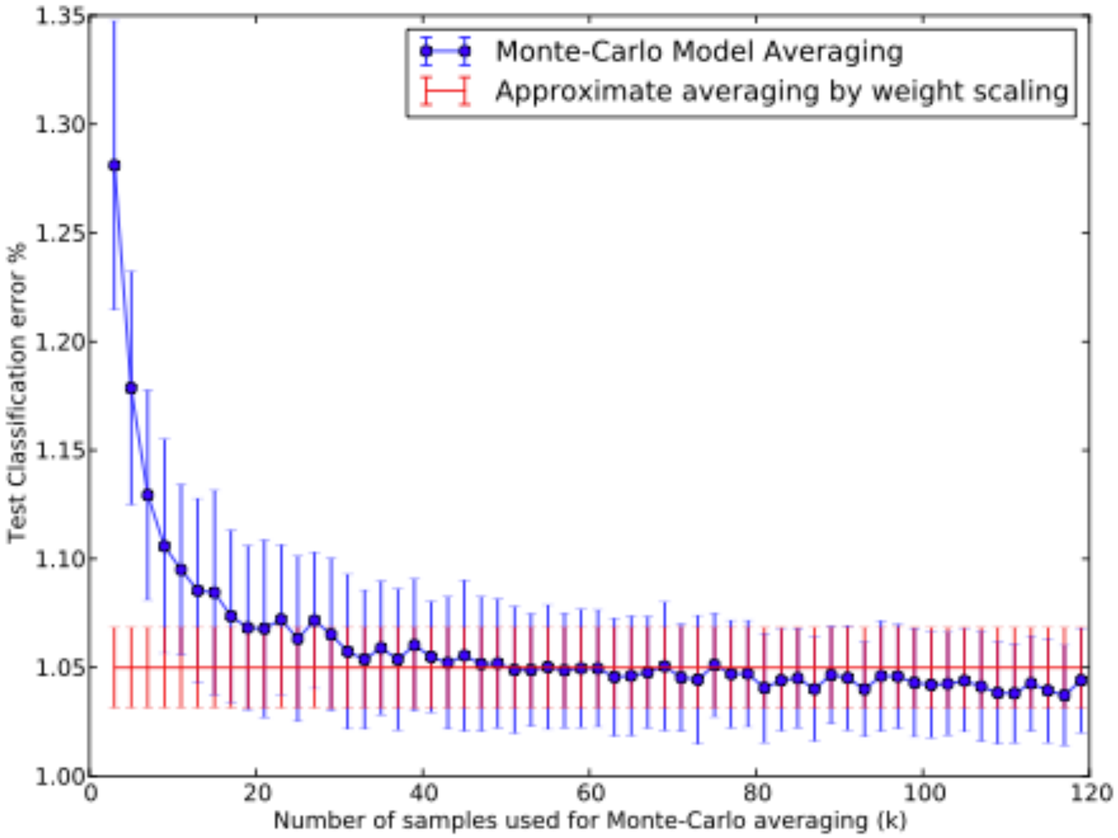


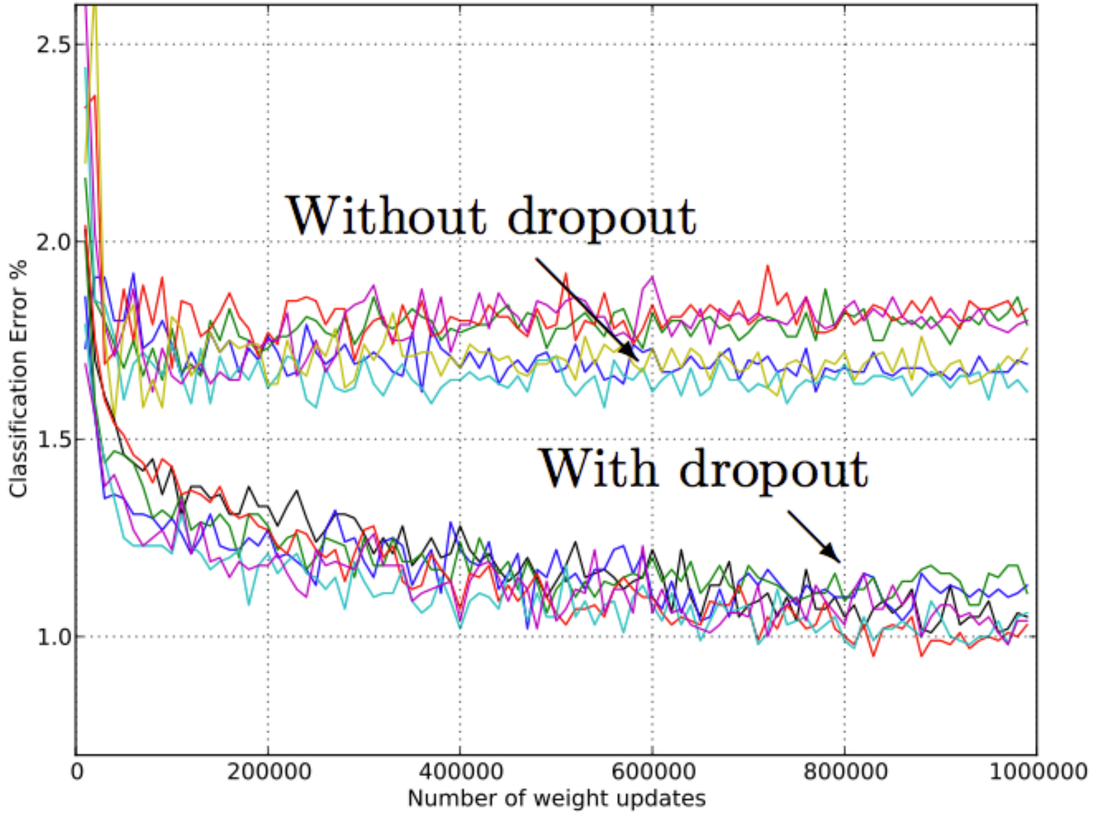
(a) Standard Neural Net

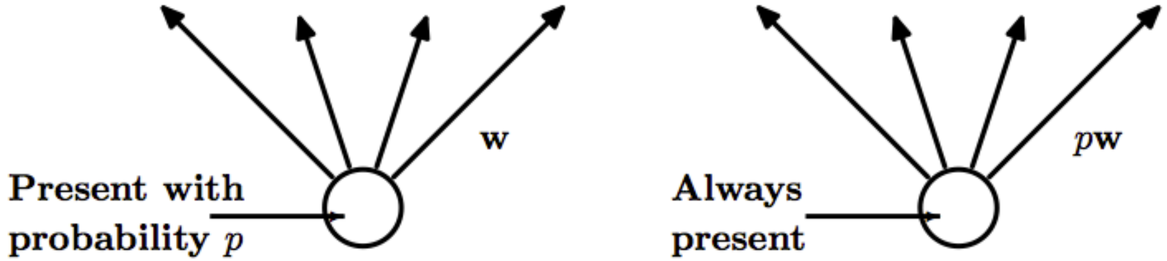
(b) After applying dropout.

During training, process each sample by a ‘decimated’ net

Test time: Deterministic Approximation

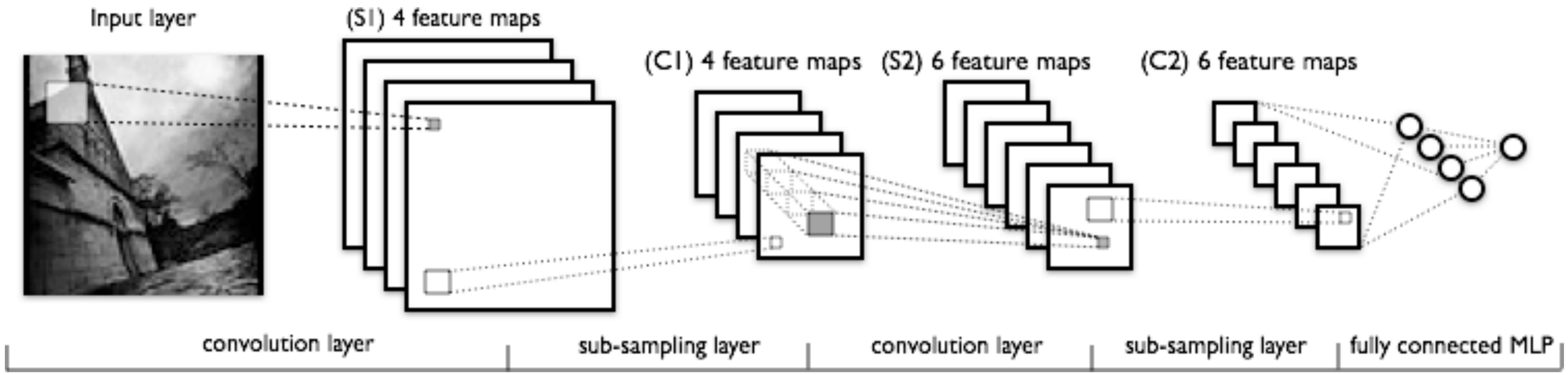


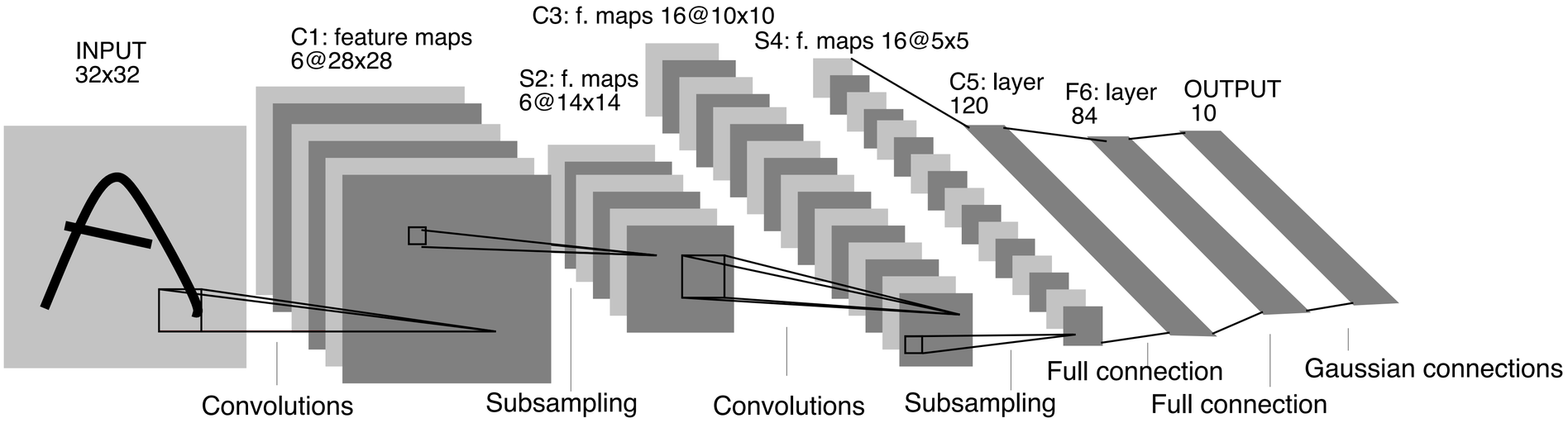




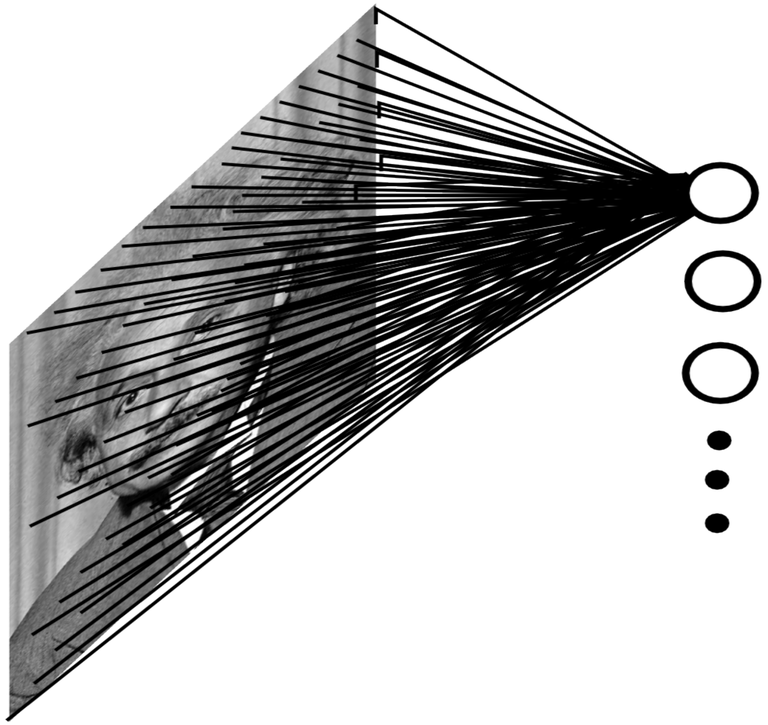
(a) At training time (b) At test time

Convolutional Neural Networks (CNNs/ConvNets)





“Fully-connected” layer



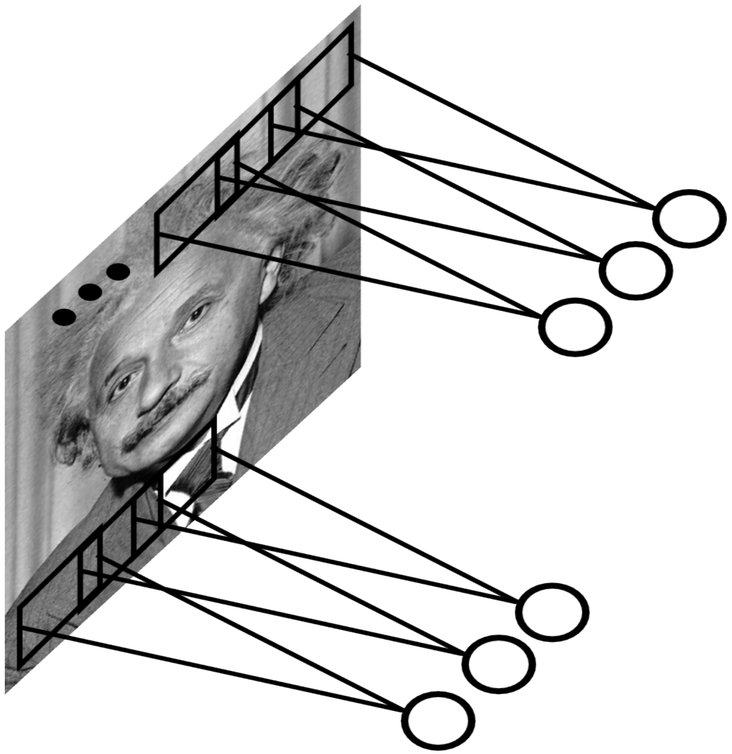
#of parameters: K^2

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \\ y_K \end{bmatrix} = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} & \dots & w_{1,K} \\ w_{2,1} & w_{2,2} & w_{2,3} & w_{2,4} & \dots & w_{2,K} \\ w_{3,1} & w_{3,2} & w_{3,3} & w_{3,4} & \dots & w_{3,K} \\ w_{4,1} & w_{4,2} & w_{4,3} & w_{4,4} & \dots & w_{4,K} \\ & & \vdots & & & \\ w_{K,1} & w_{K,2} & w_{K,3} & w_{K,4} & \dots & w_{K,K} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_K \end{bmatrix}$$

slow, big, prone to overfitting

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Idea: repeat a local operation (image is stationary)



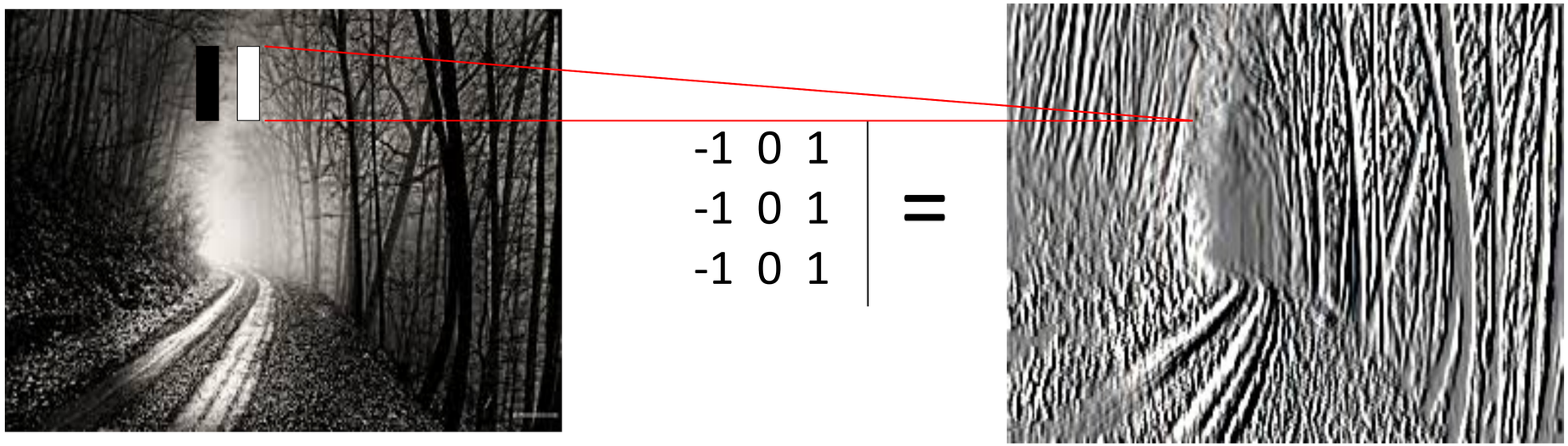
#of parameters: size of window

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \\ y_K \end{bmatrix} = \begin{bmatrix} w_0 & w_1 & w_2 & 0 & \dots & 0 \\ 0 & w_0 & w_1 & w_2 & \dots & 0 \\ 0 & 0 & w_0 & w_1 & \dots & 0 \\ 0 & 0 & 0 & w_0 & \dots & 0 \\ & & \vdots & & & \\ 0 & 0 & 0 & 0 & \dots & w_0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_K \end{bmatrix}$$

fast, small, regularized

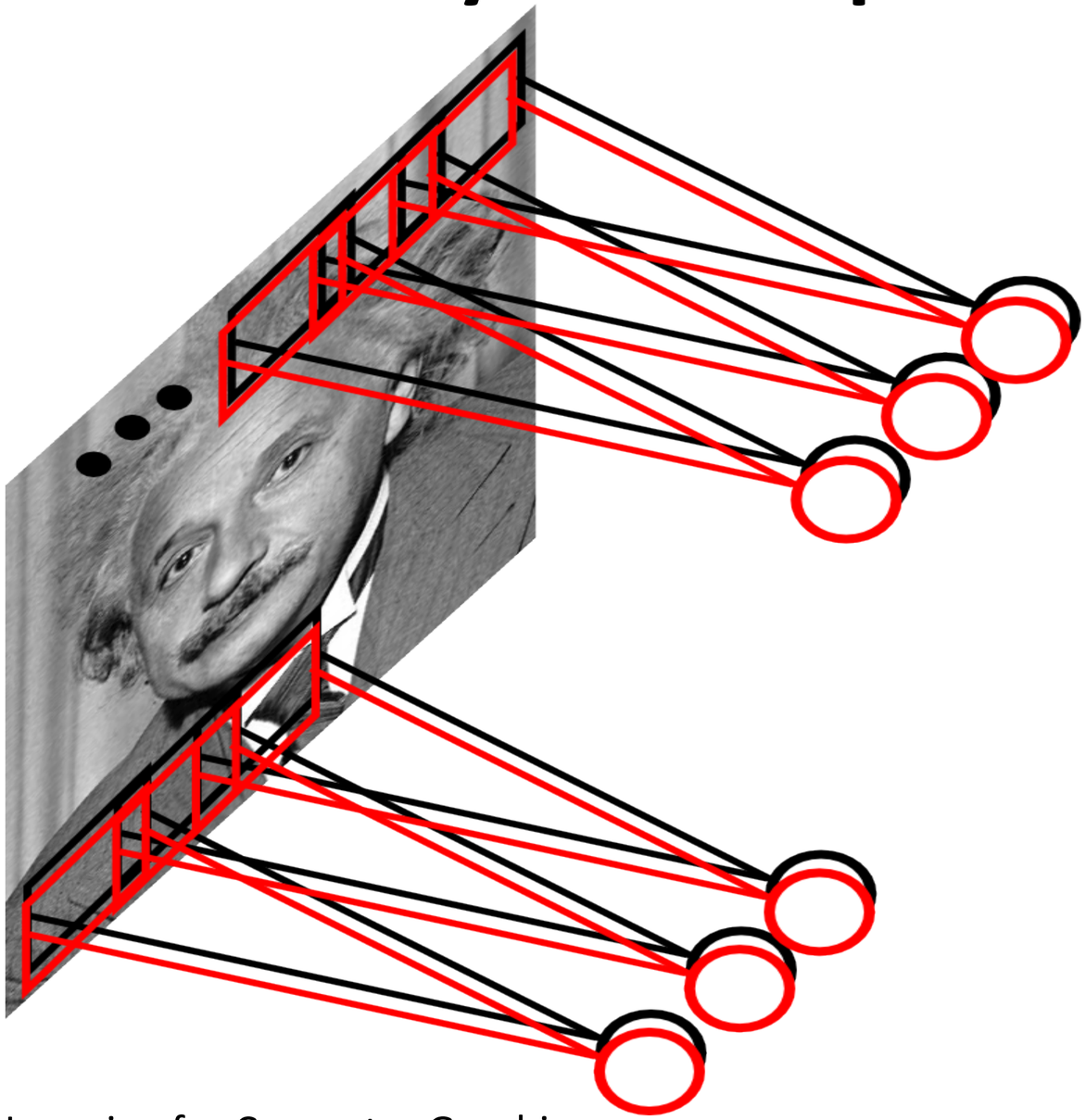
Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Identical to convolution operation



But with learnable weights!

Convolutional layer: multiple output channels



Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

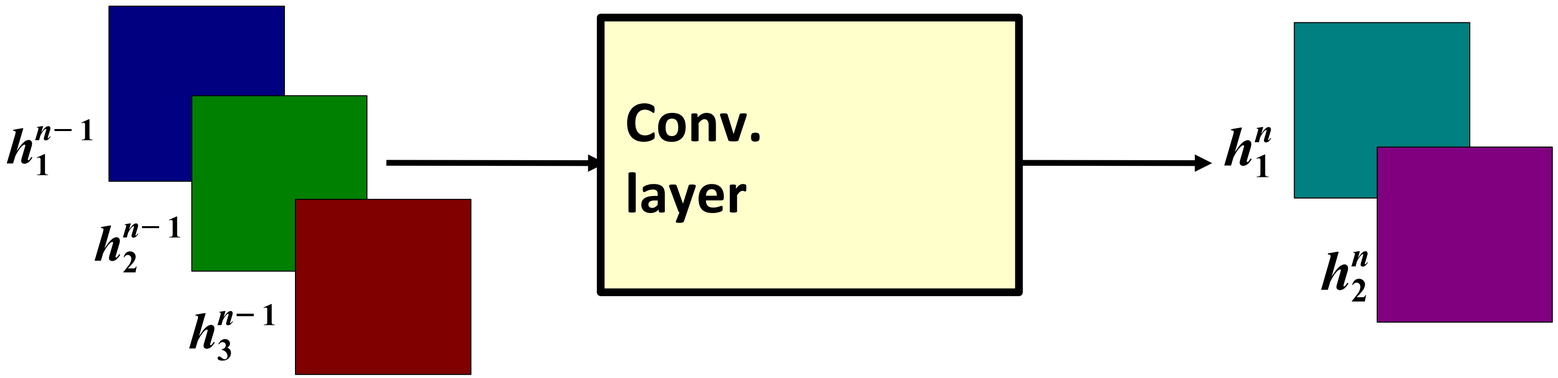
Convolutional layer composition

$$h_i^n = \max\left\{0, \sum_{j=1}^{\#\text{input channels}} h_j^{n-1} * w_{ij}^n\right\}$$

output feature map

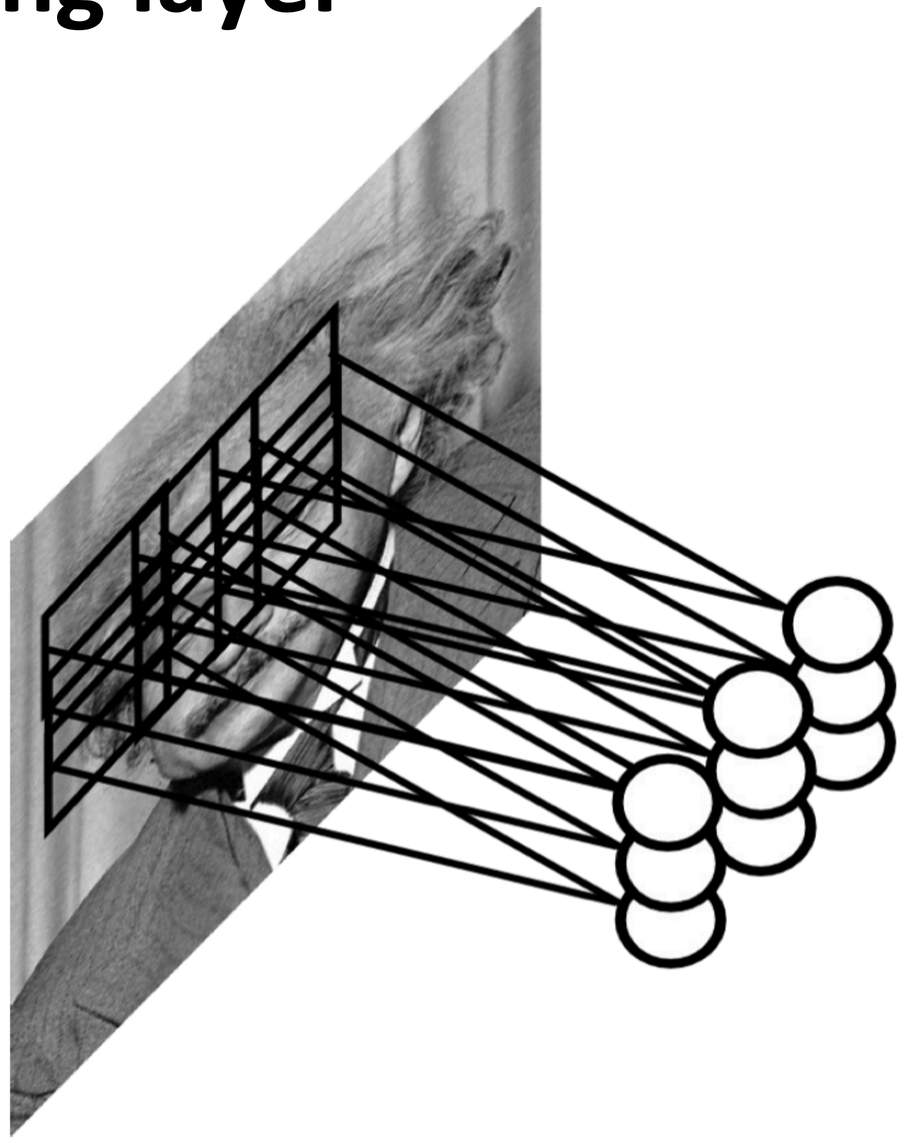
input feature map

kernel



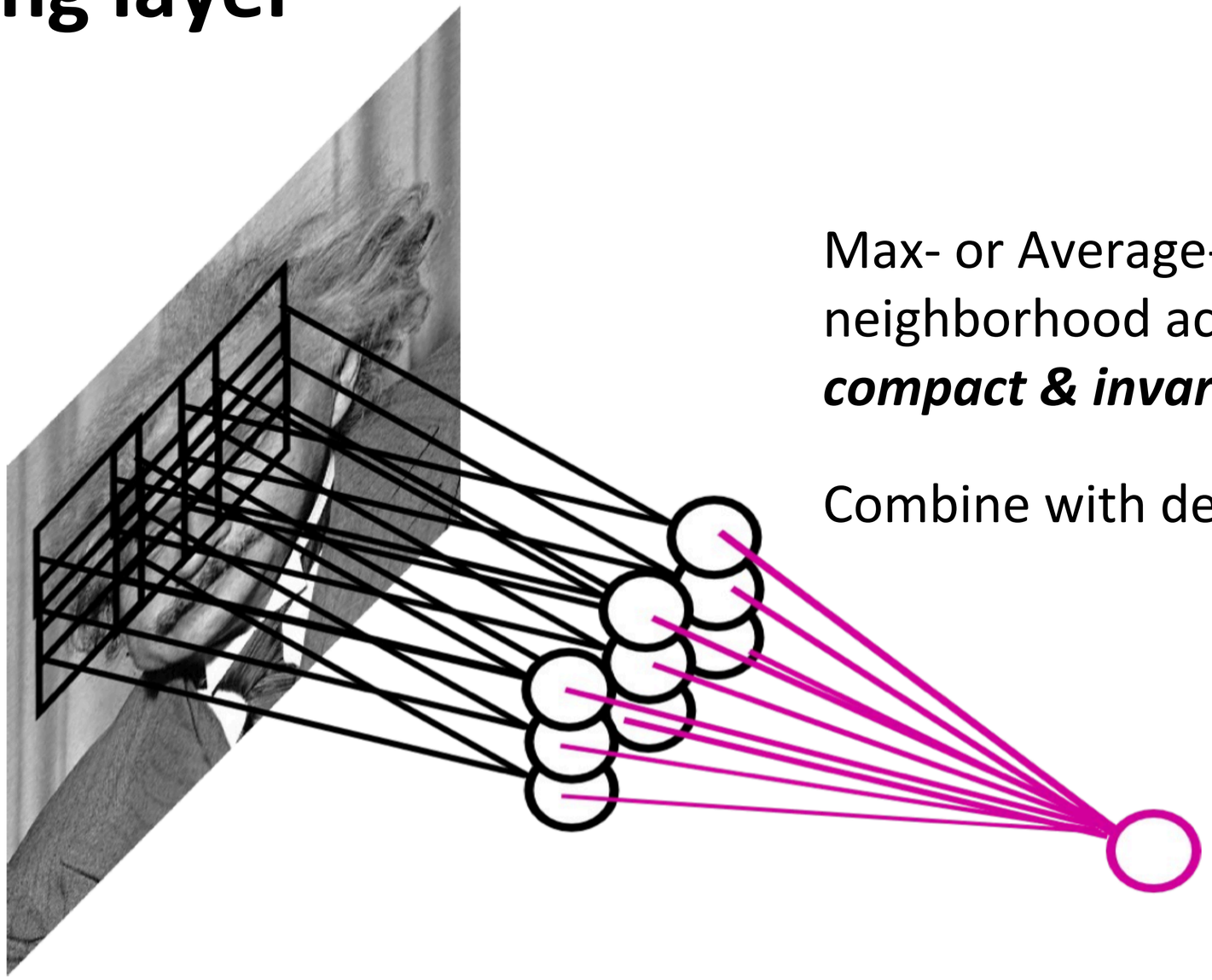
Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Pooling layer



Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Pooling layer



Max- or Average-based neighborhood activation summary:
compact & invariant

Combine with decimation!

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Receptive field

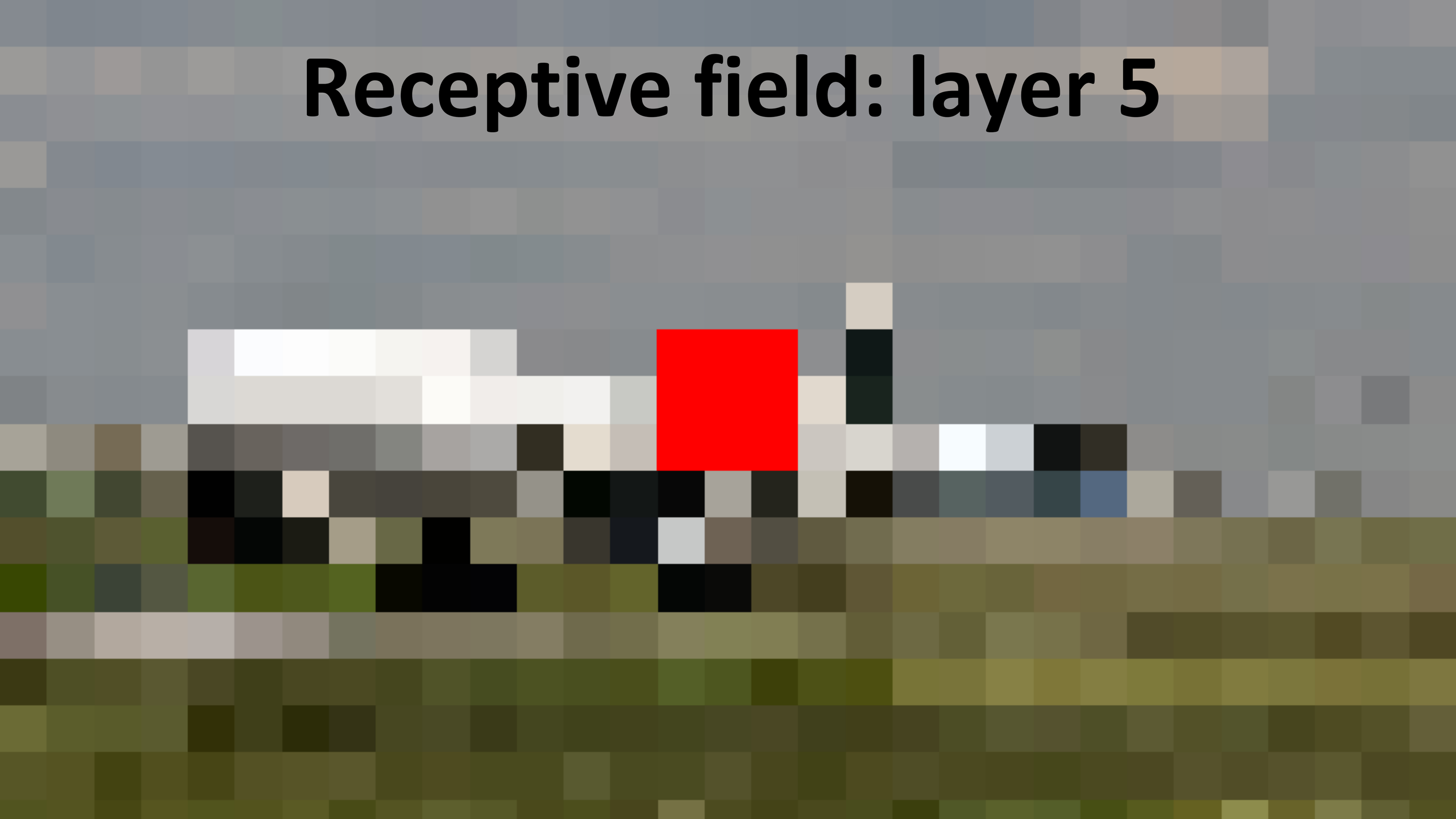
Receptive field: layer 1

Receptive field: layer 2

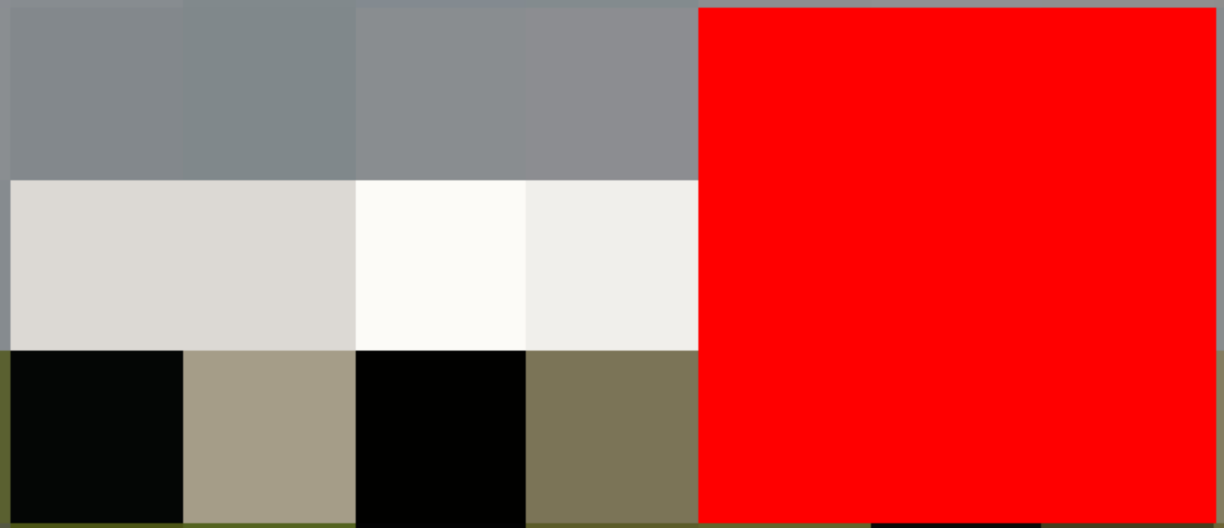
Receptive field: layer 3

Receptive field: layer 4

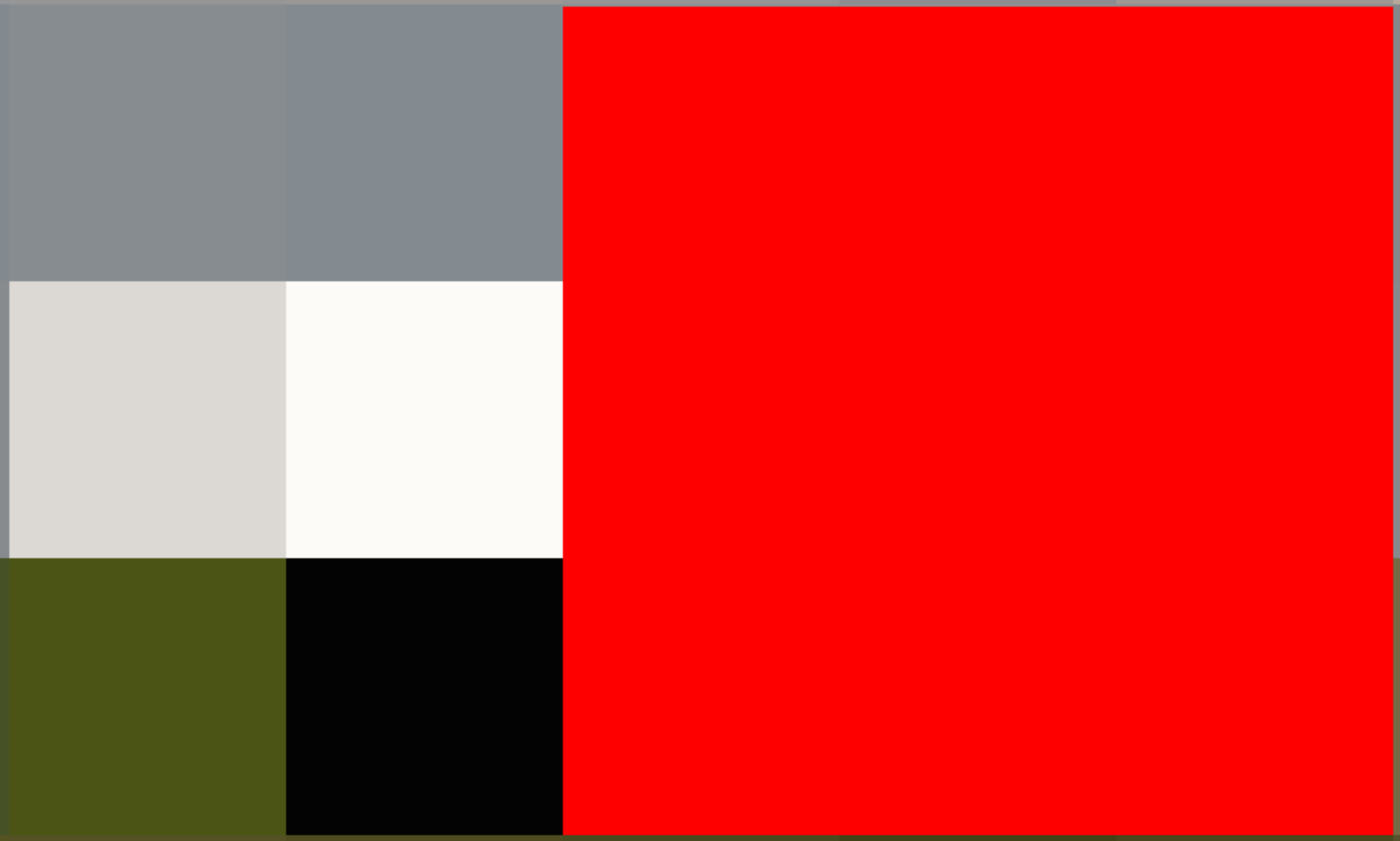
Receptive field: layer 5



Receptive field: layer 6

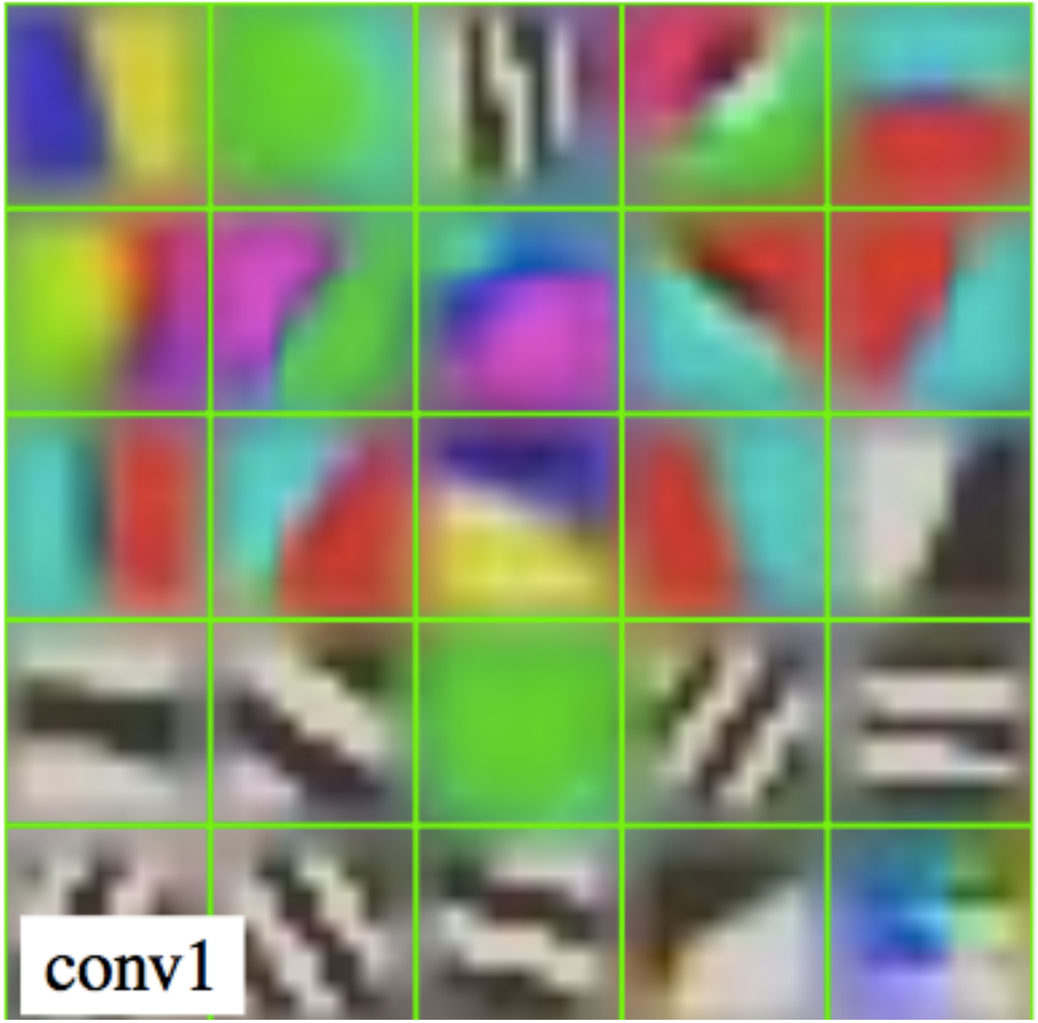


Receptive field: layer 7



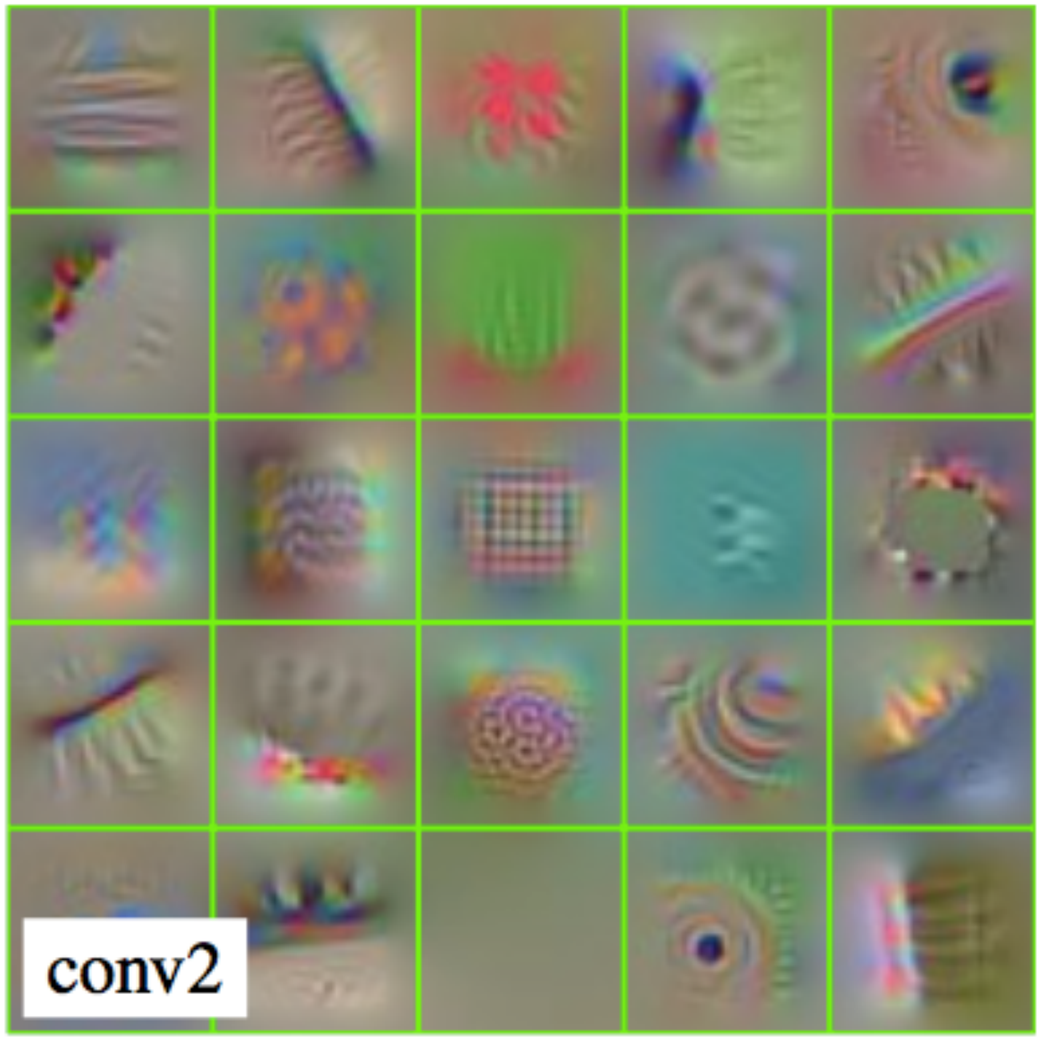
Receptive field: layer 8

What is in the network?



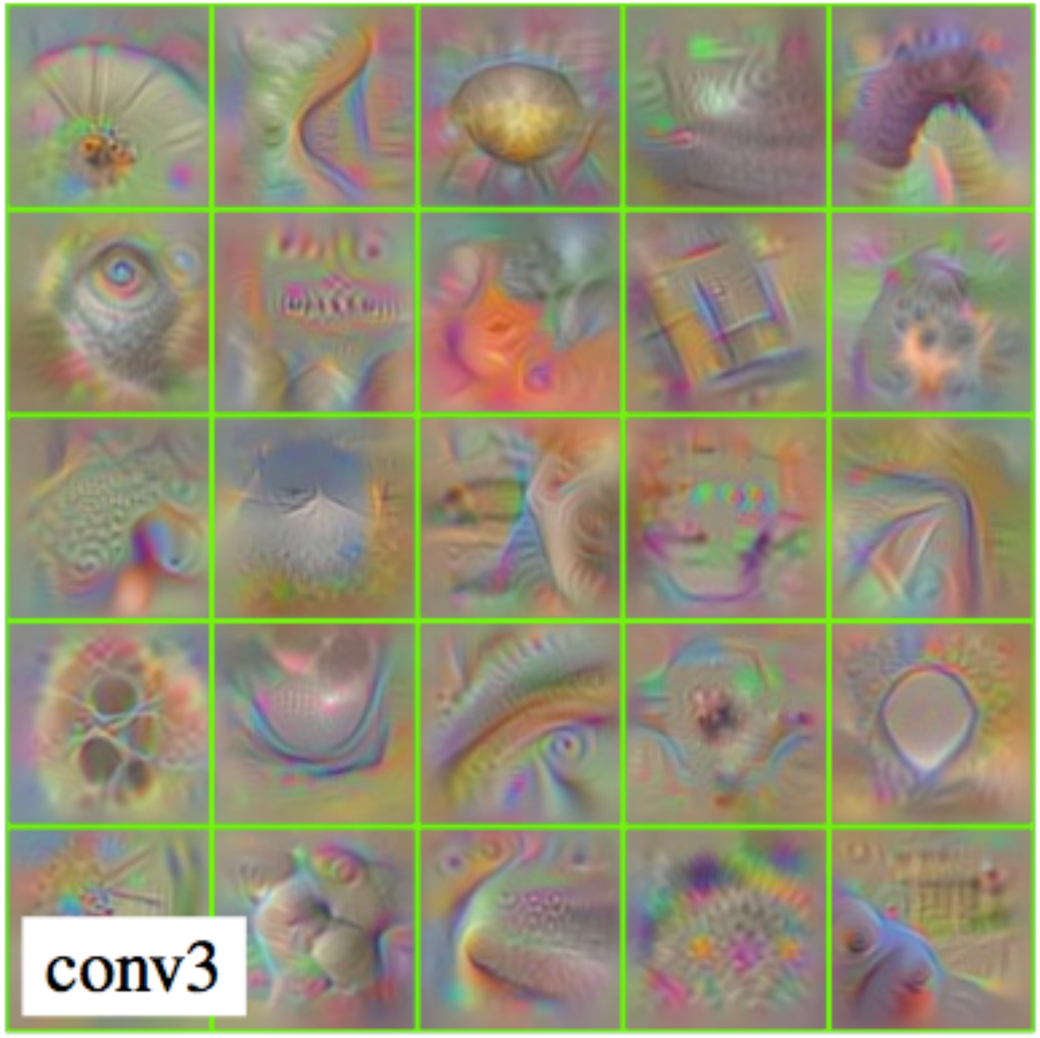
Visualizing deep convolutional neural networks using natural pre-images, A. Mahendra and A. Vedaldi

What is in the network?



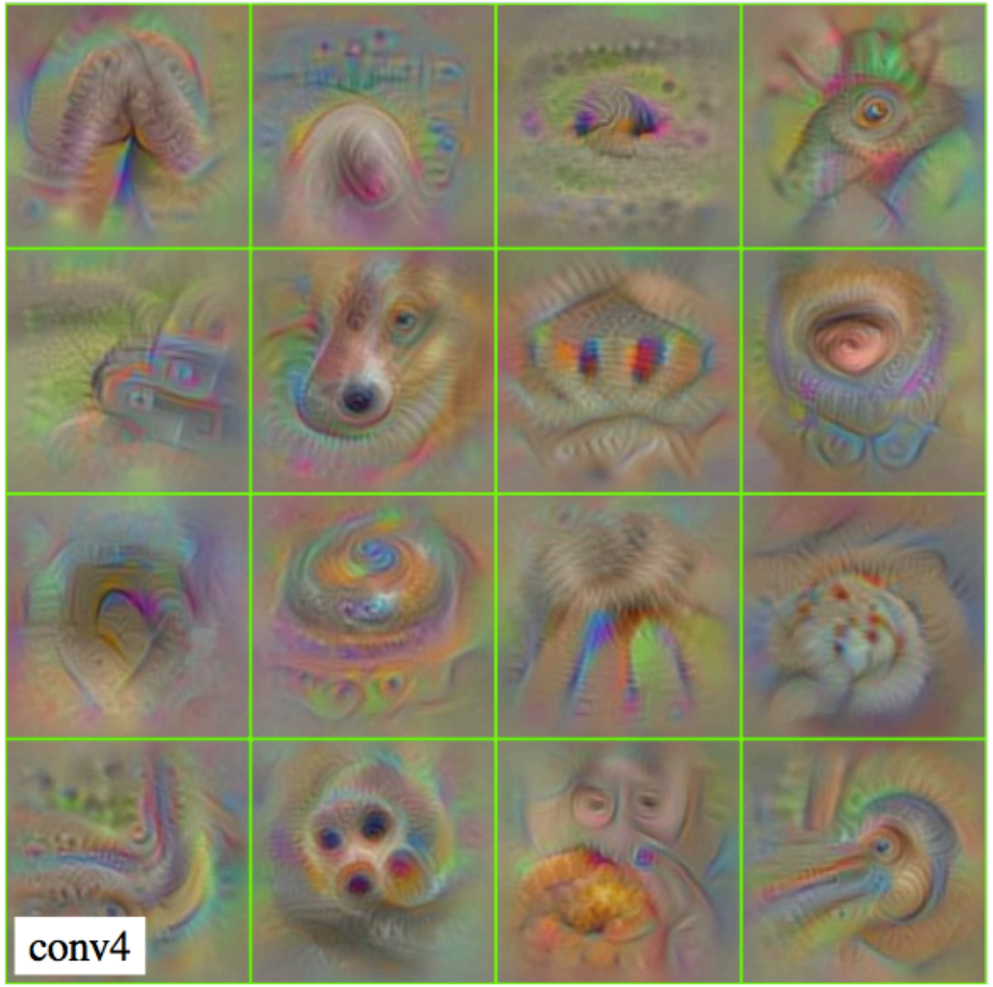
Visualizing deep convolutional neural networks using natural pre-images, A. Mahendra and A. Vedaldi

What is in the network?



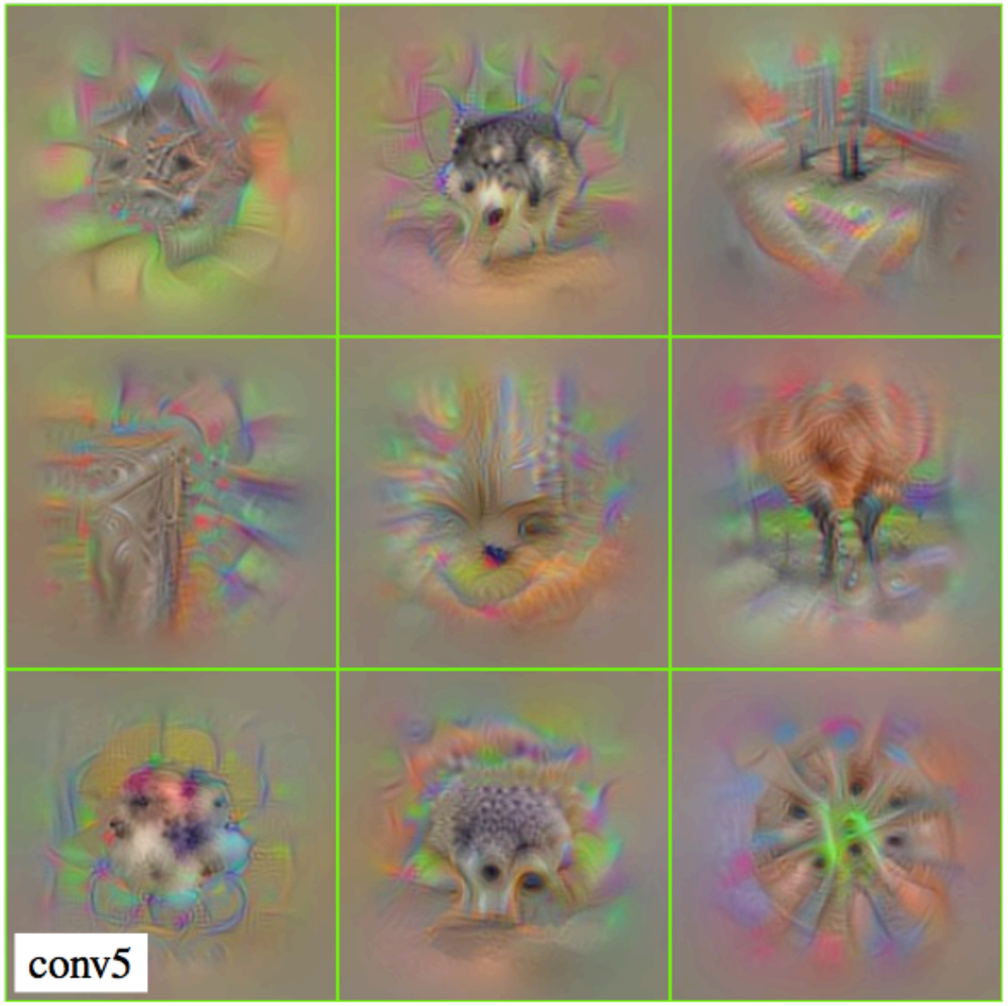
Visualizing deep convolutional neural networks using natural pre-images,
A. Mahendra and A. Vedaldi

What is in the network?



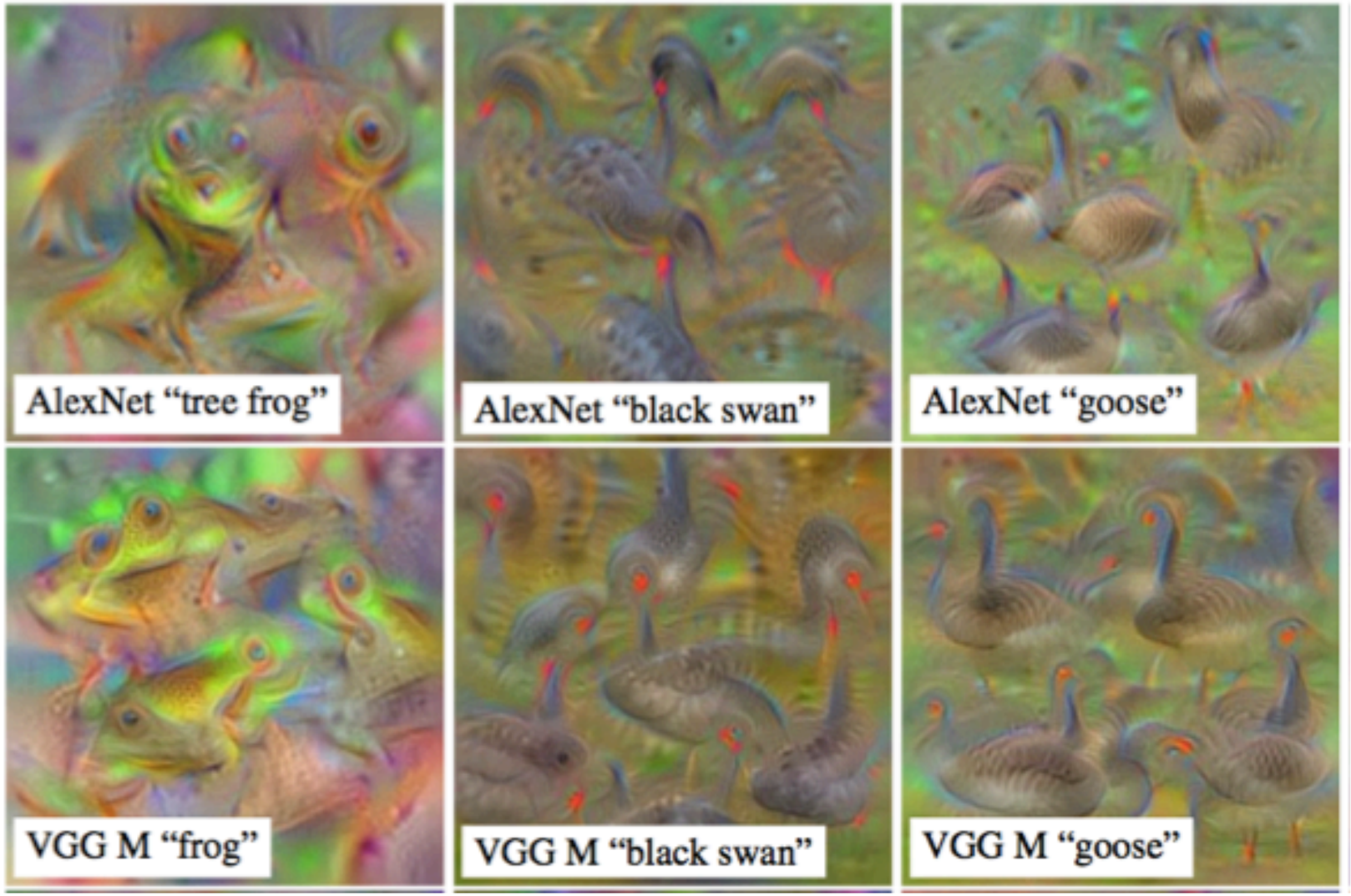
Visualizing deep convolutional neural networks using natural pre-images, A. Mahendra and A. Vedaldi

What is in the network?



Visualizing deep convolutional neural networks using natural pre-images,
A. Mahendra and A. Vedaldi

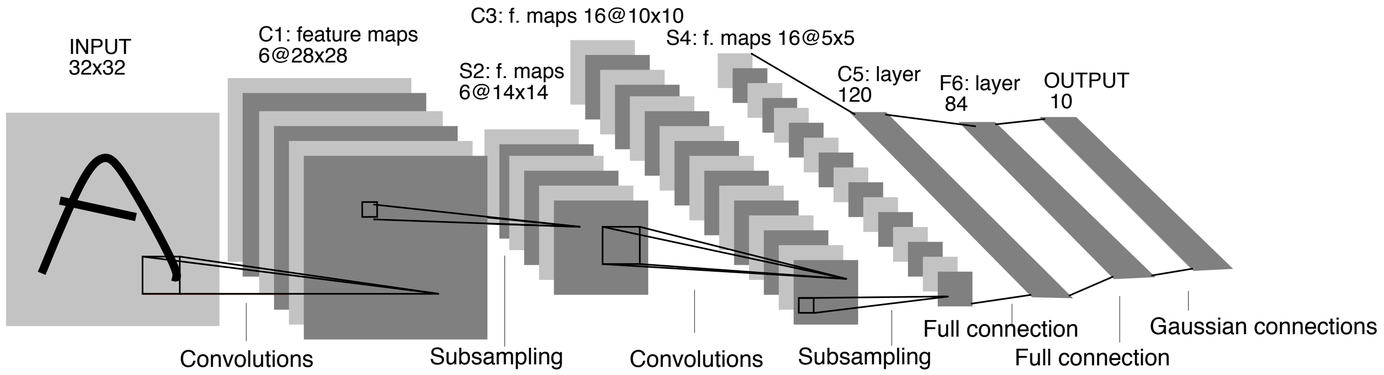
What is in the network?



Visualizing deep convolutional neural networks using natural pre-images, A. Mahendra and A. Vedaldi

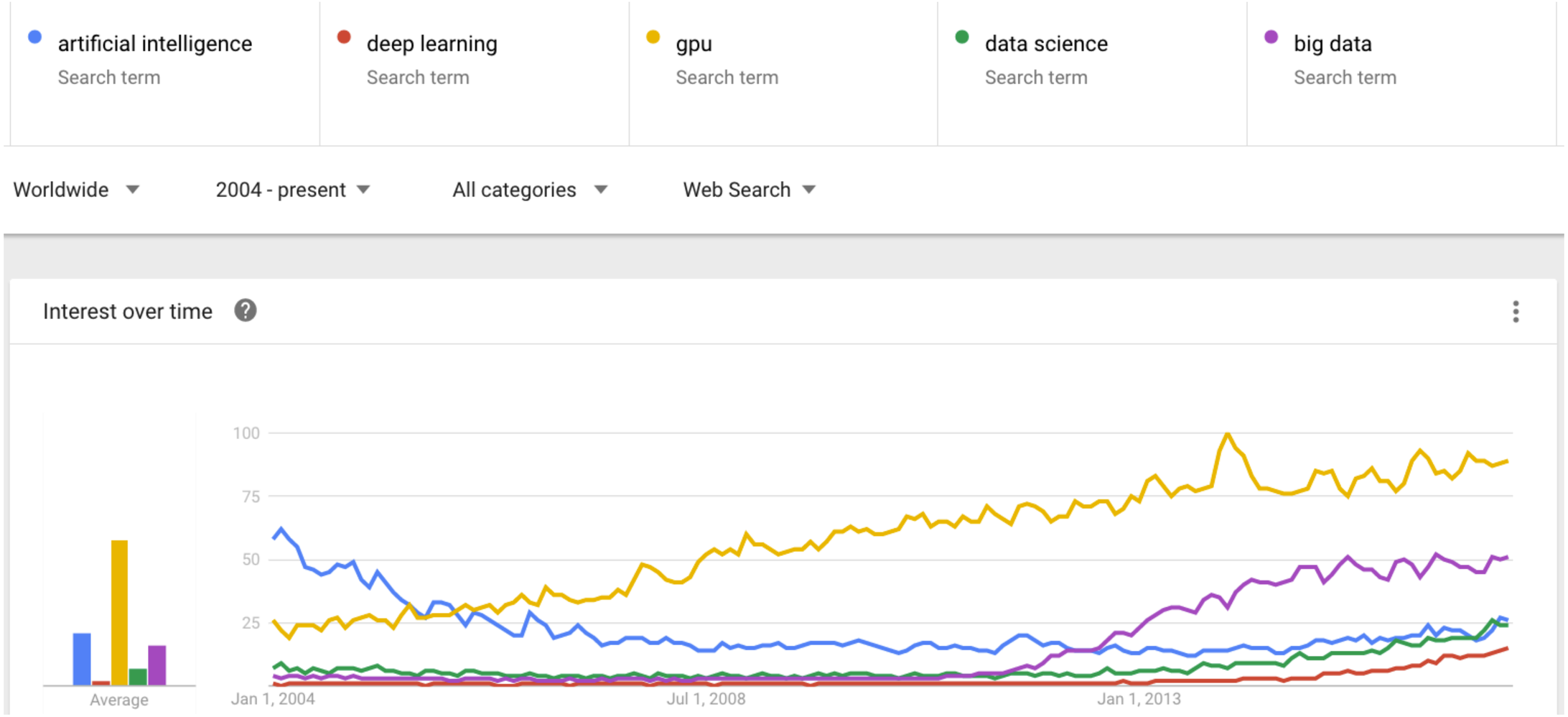
Modern Architectures: going deeper

CNNs, late 1980's: LeNet



Gradient-based learning applied to document recognition.
Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998

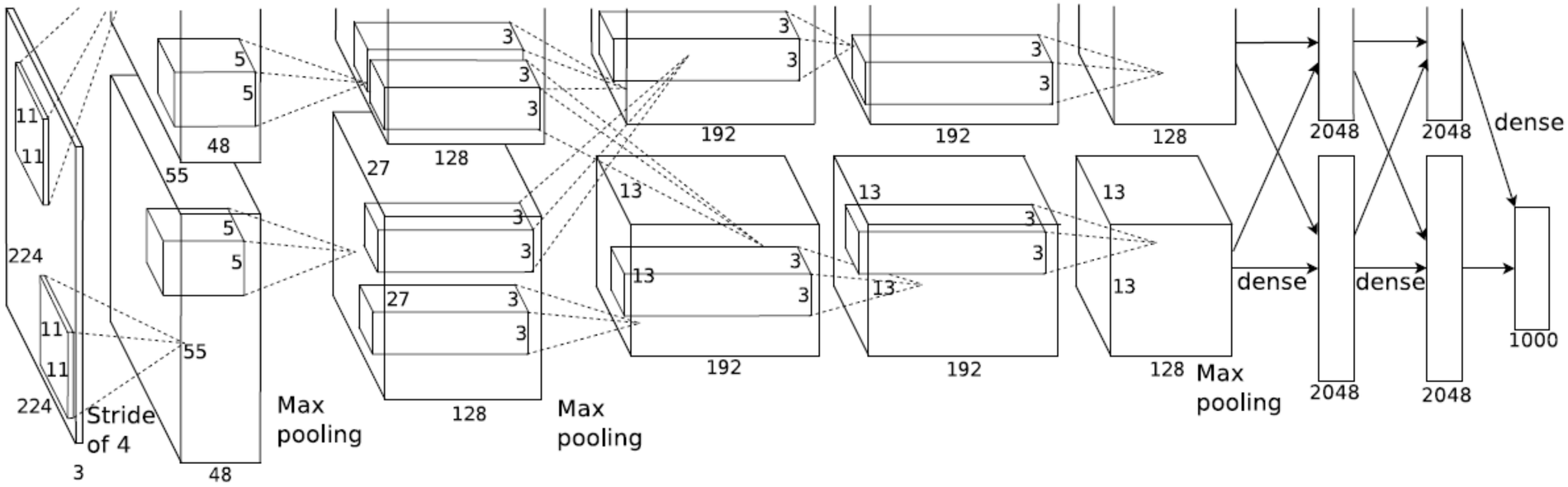
What happened in between?



deep learning = neural networks (+ big data + GPUs)

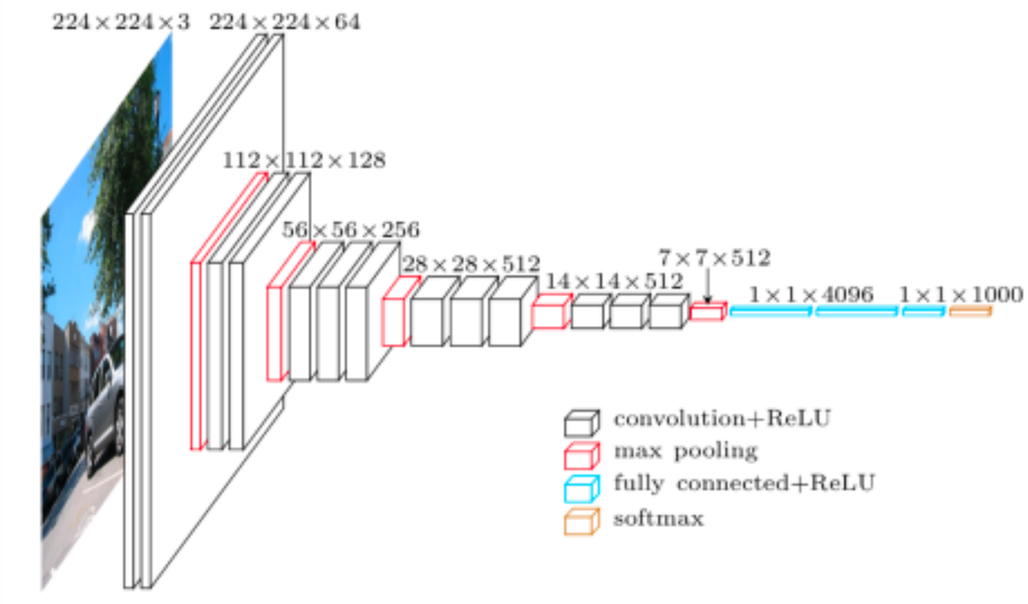
+ a few more recent tricks!

CNNs, 2012



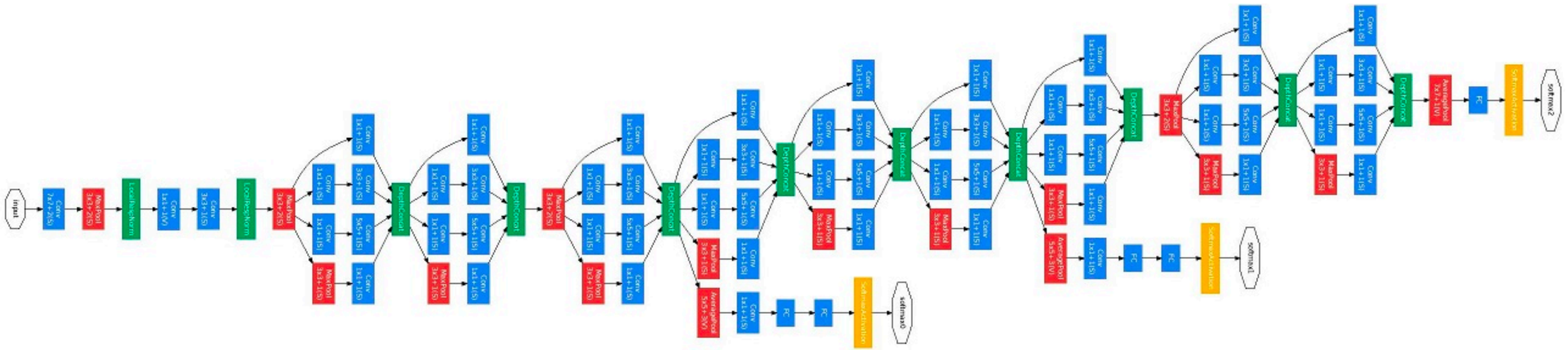
AlexNet
Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton:
ImageNet classification with deep convolutional neural networks. Commun. ACM 60(6): 84-90 (2017)

CNNs, 2014: VGG



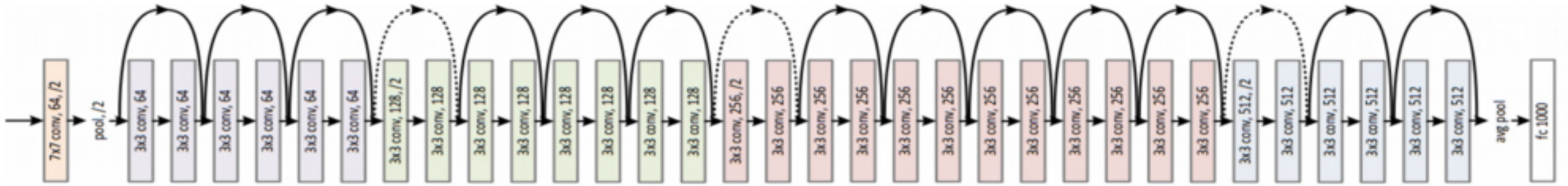
Karen Simonyan, Andrew Zisserman (=Visual Geometry Group)
Very Deep Convolutional Networks for Large-Scale Image Recognition, arxiv, 2014.

CNNs, 2014: GoogLeNet



Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich
Going Deeper with Convolutions, CVPR 2015

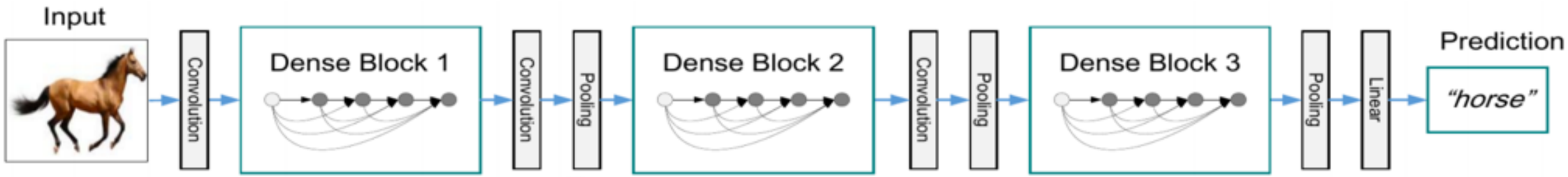
CNNs, 2015: ResNet



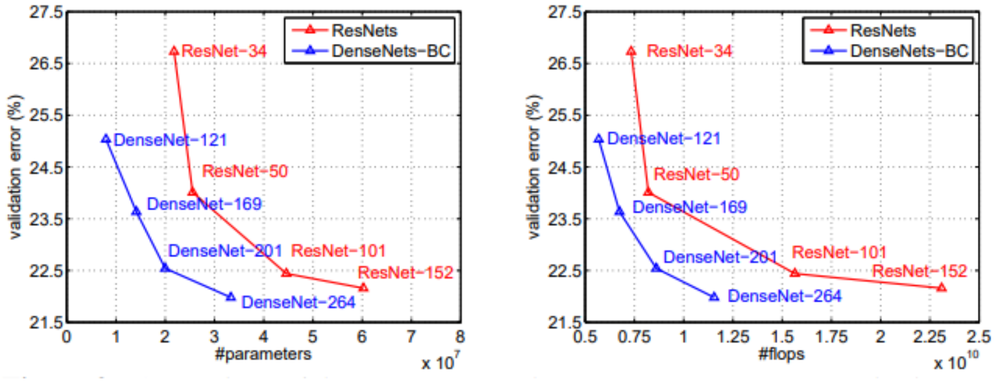
ResNet
Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun,
Deep Residual Learning for Image Recognition
CVPR 2016

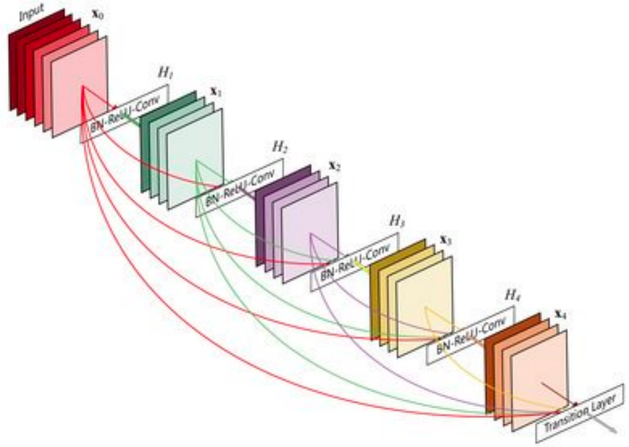
CNNs, 2017: DenseNet

Densely Connected Convolutional Networks, CVPR 2017
Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger



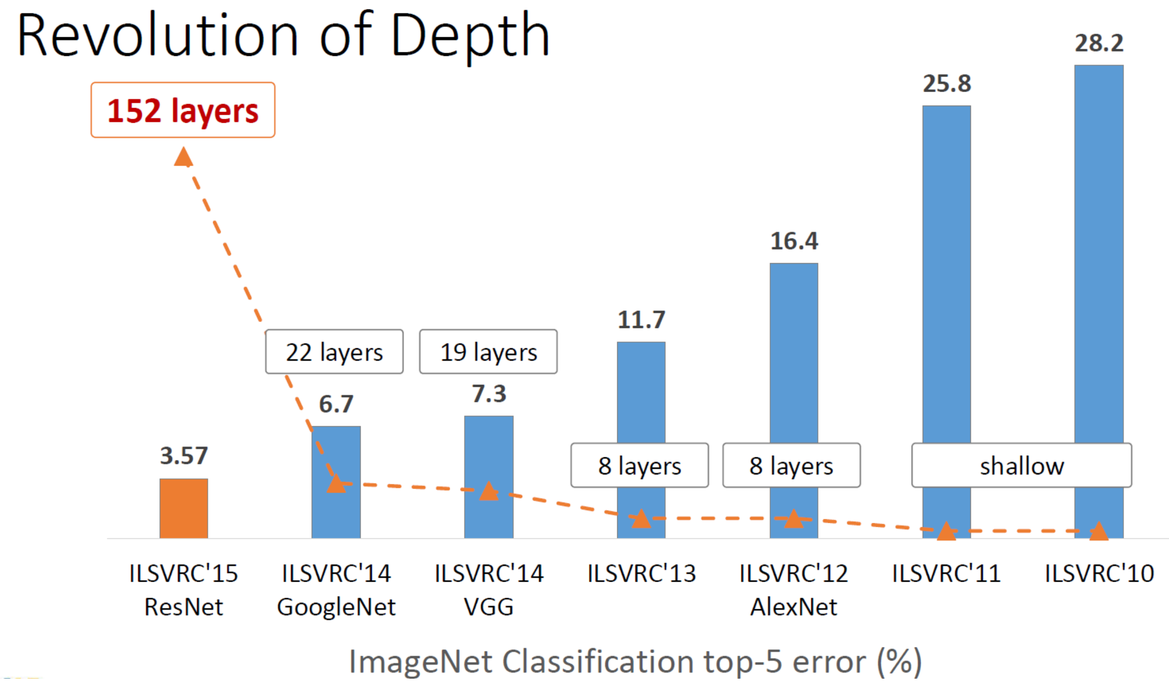
better performance/parameter ratio





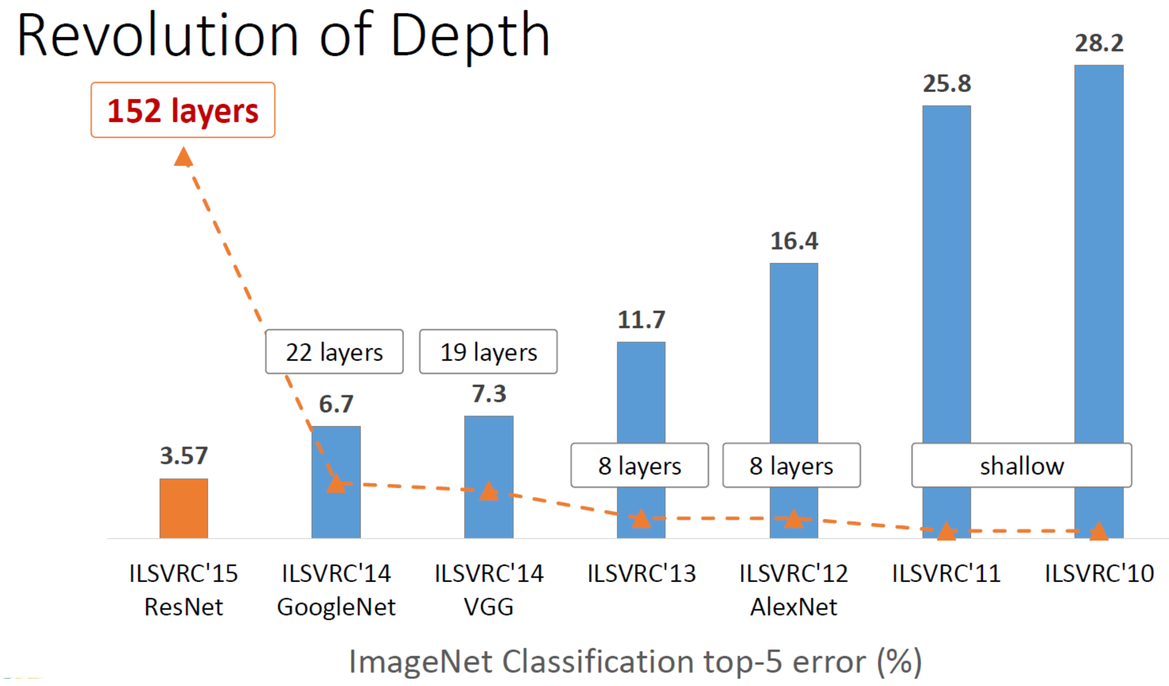
The Deeper, the Better

- Deeper networks can cover more complex problems
  - Increasingly large receptive field size & rich patterns



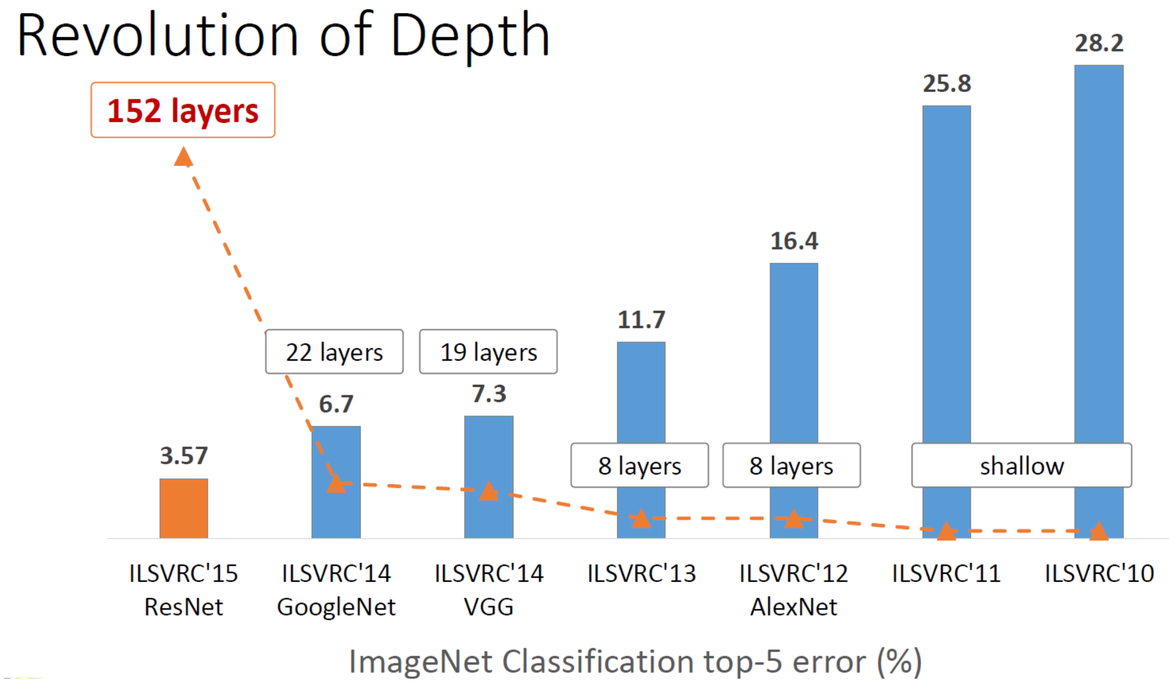
Going Deeper

- From 2 to 10: 2010-2012
  - ReLUs
  - Dropout
  - …



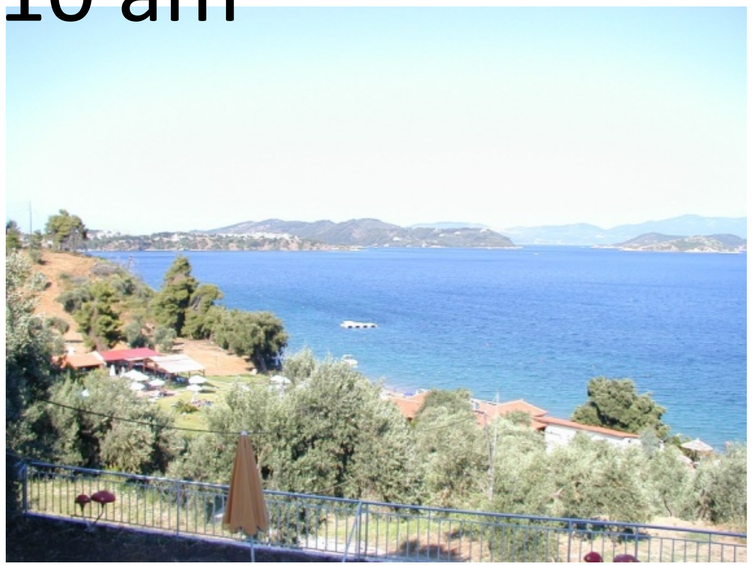
Going Deeper

- From 10 to 20: 2015
  - Batch Normalization



External Covariate Shift: your input changes

10 am

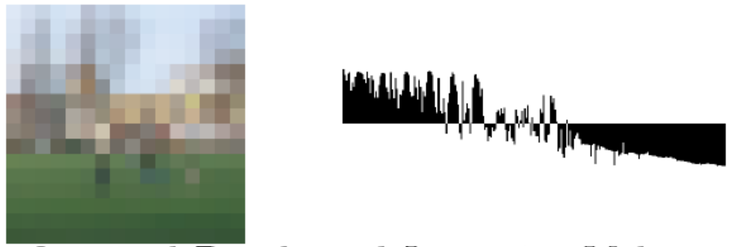


2pm

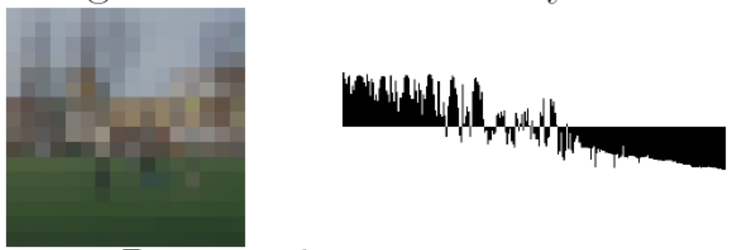
7pm

“Whitening”: Set Mean = 0, Variance = 1

Photometric transformation: $I \rightarrow a\,I + b$



Original Patch and Intensity Values



Brightness Decreased

Contrast increased,

- Make each patch have zero mean:

$$\mu = \frac{1}{N}\sum_{x,y} I(x,y)$$

$$Z(x,y) = I(x,y) - \mu$$

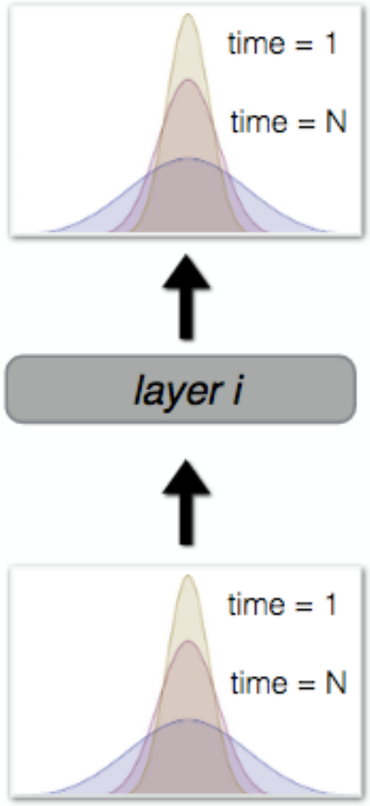
- Then make it have unit variance:

$$\sigma^2 = \frac{1}{N}\sum_{x,y} Z(x,y)^2$$

$$ZN(x,y) = \frac{Z(x,y)}{\sigma}$$

Internal Covariate Shift

Neural network activations during training: moving target



Batch Normalization

Whiten-as-you-go:

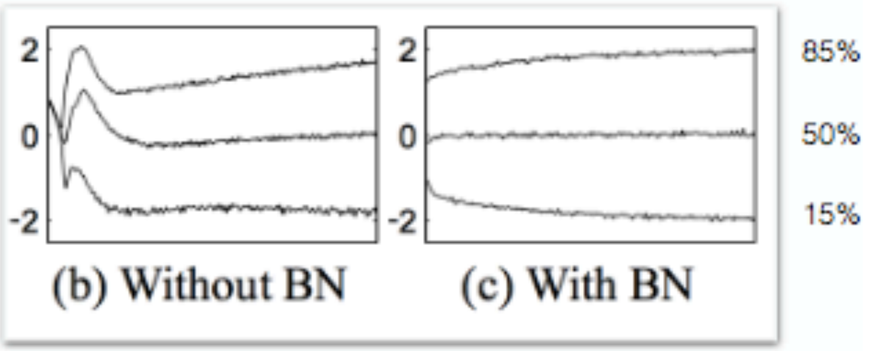
- Normalize the activations in each layer within a mini-batch.
- Learn the mean and variance (γ, β) of each layer as parameters

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

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

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

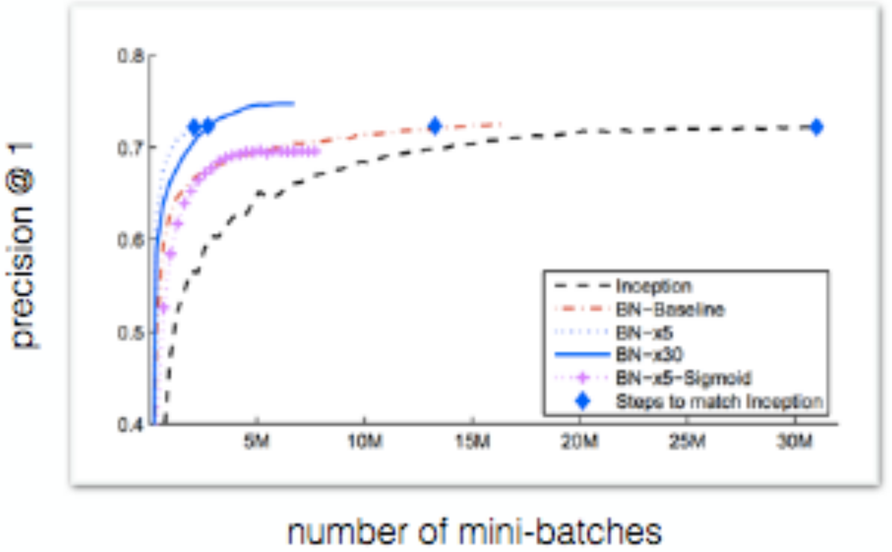
$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$



Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift
S Ioffe and C Szegedy (2015)

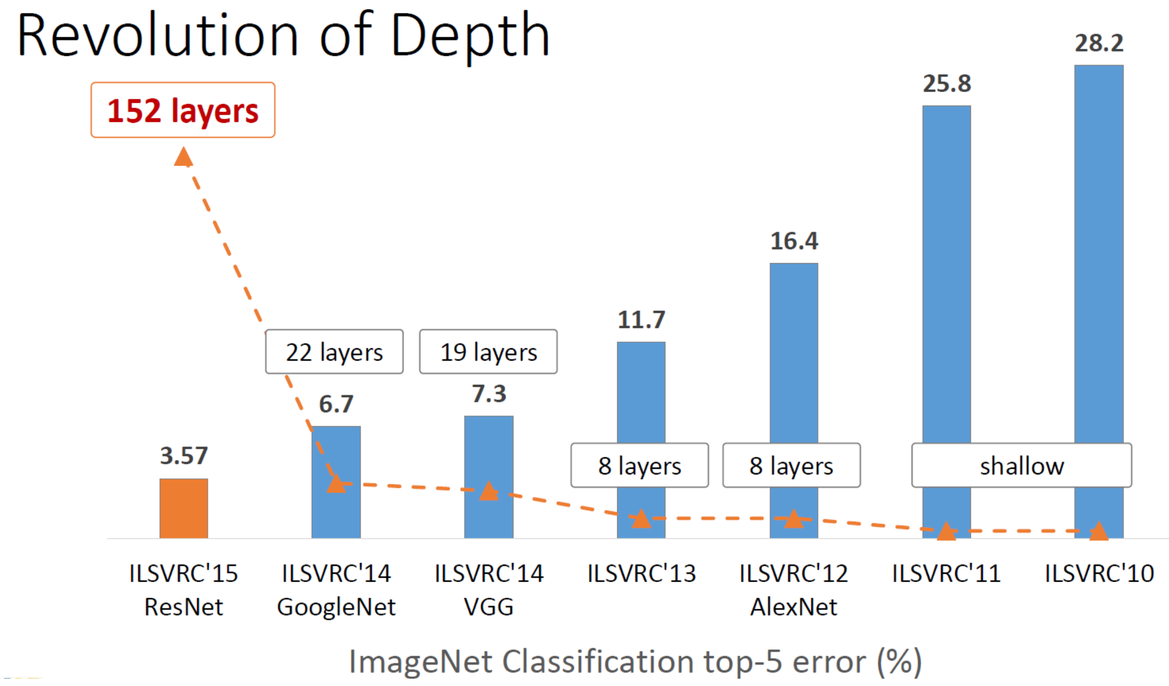
Batch Normalization: used in all current systems

- Multi-layer CNN's train faster with fewer data samples (15x).
- Employ faster learning rates and less network regularizations.
- Achieves state of the art results on ImageNet.



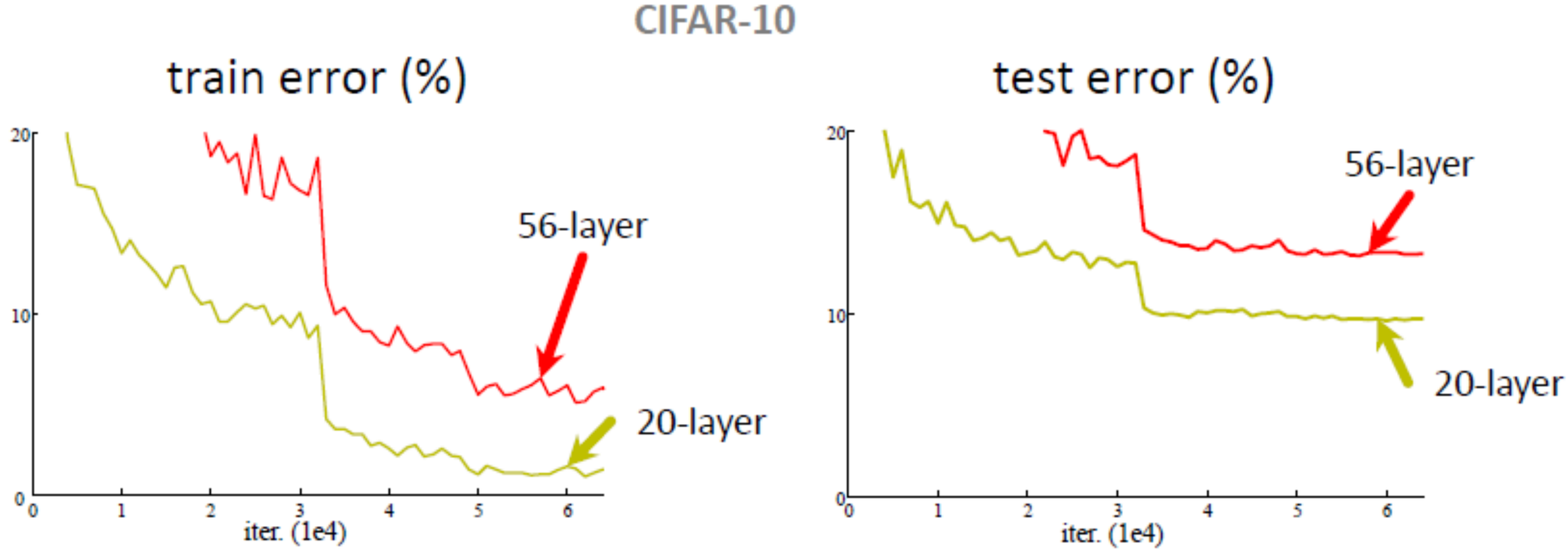
Going Deeper

- From 20 to 100/1000
  - Residual networks



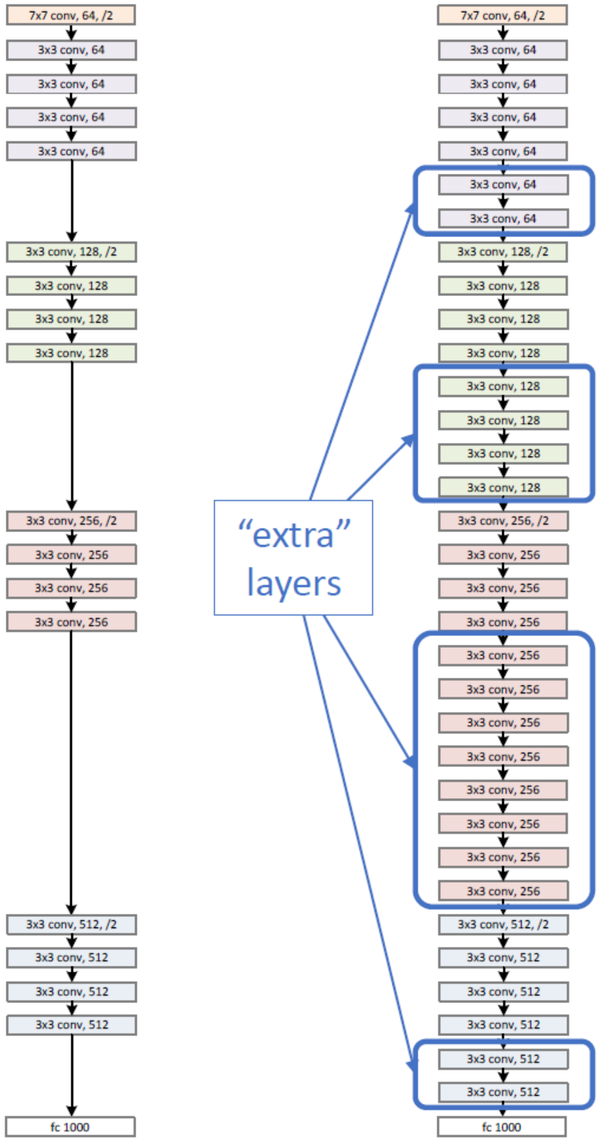
Plain network: deeper is not necessarily better

- Plain nets: stacking 3x3 conv layers
- 56-layer net has higher training error and test error than 20-layer net



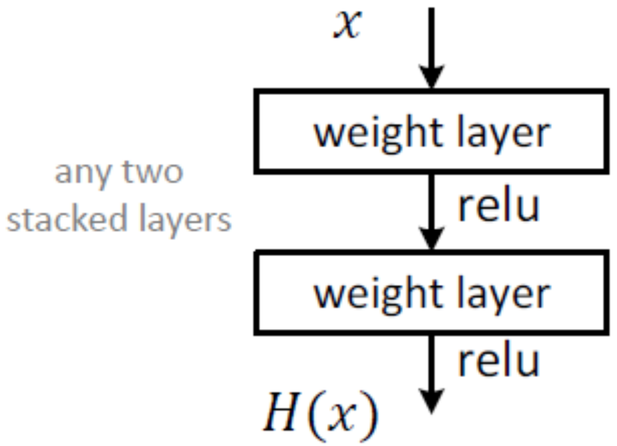
Residual Network

- Naïve solution
  - If extra layers are an identity mapping, then training errors can not increase



Residual Network

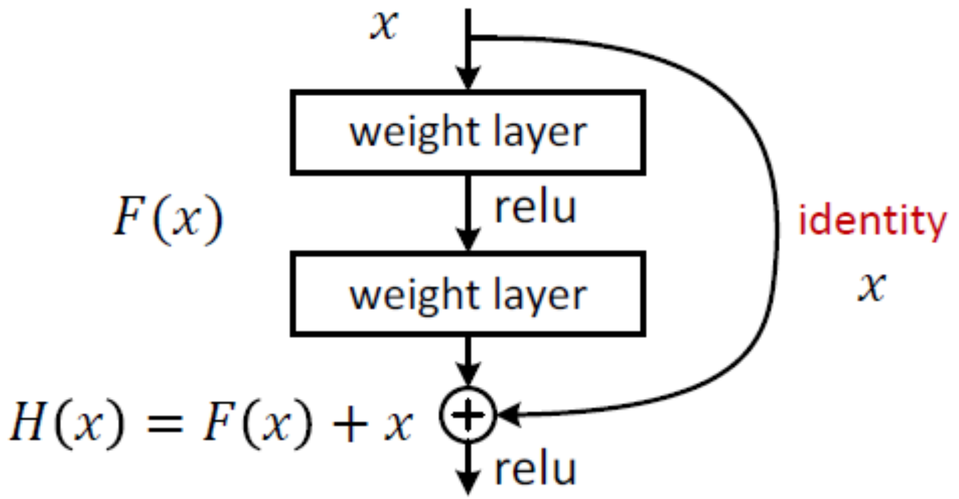
- Plain block
  - Difficult to make identity mapping because of multiple non-linear layers



Residual Network

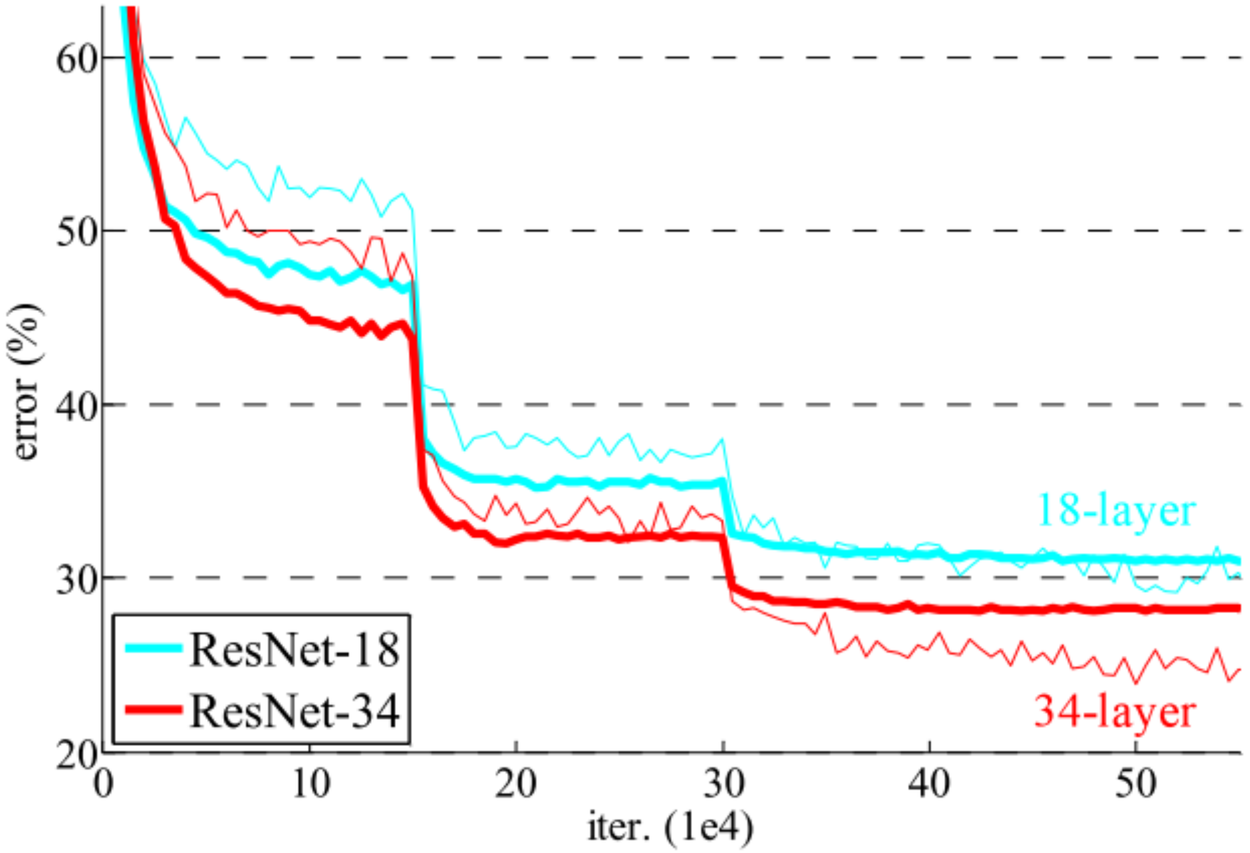
- Residual block
  - If identity were optimal, easy to set weights as 0
  - If optimal mapping is closer to identity, easier to find small fluctuations

Appropriate for treating perturbation as keeping a base information

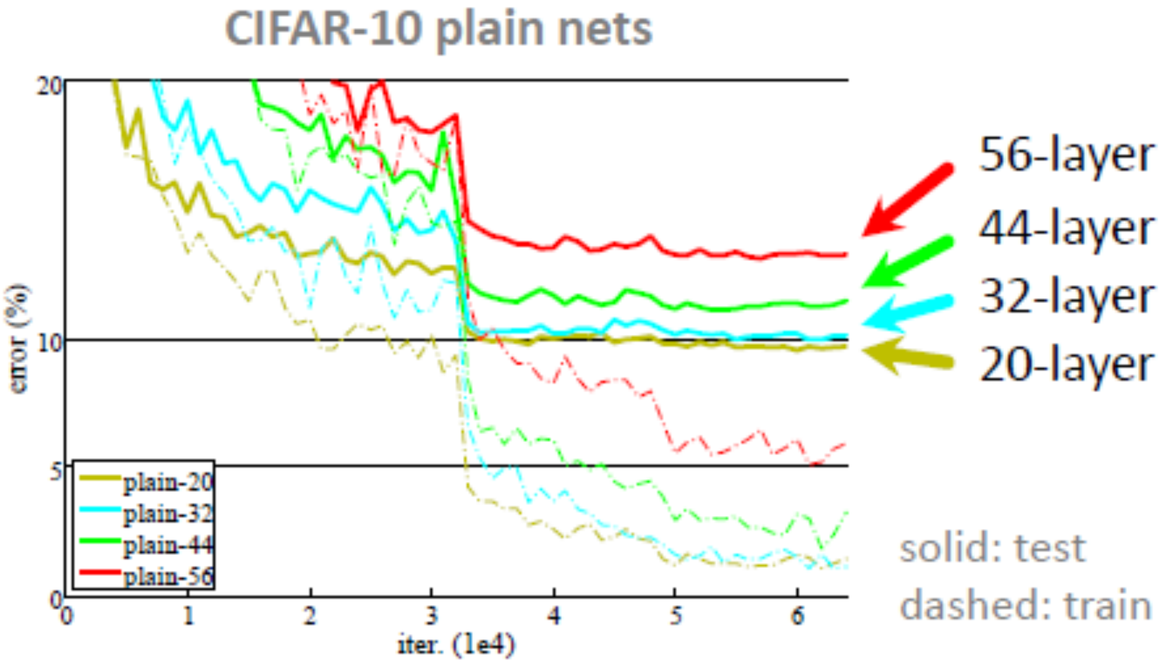


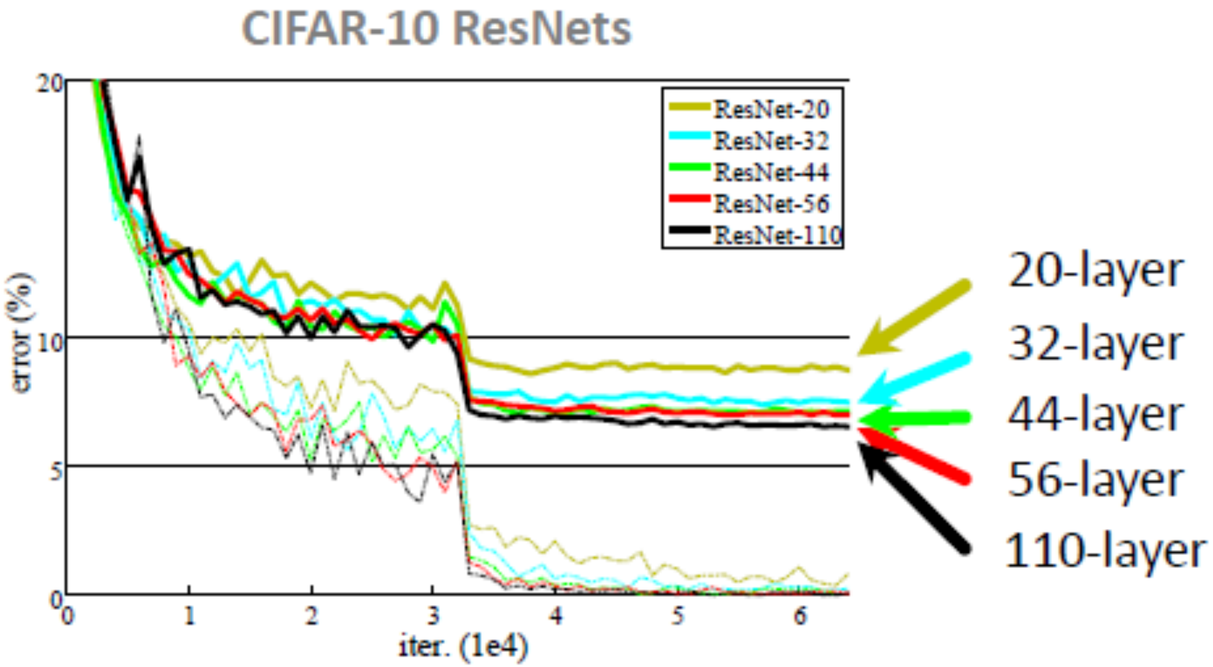
Residual Network

- Deeper ResNets have lower training error



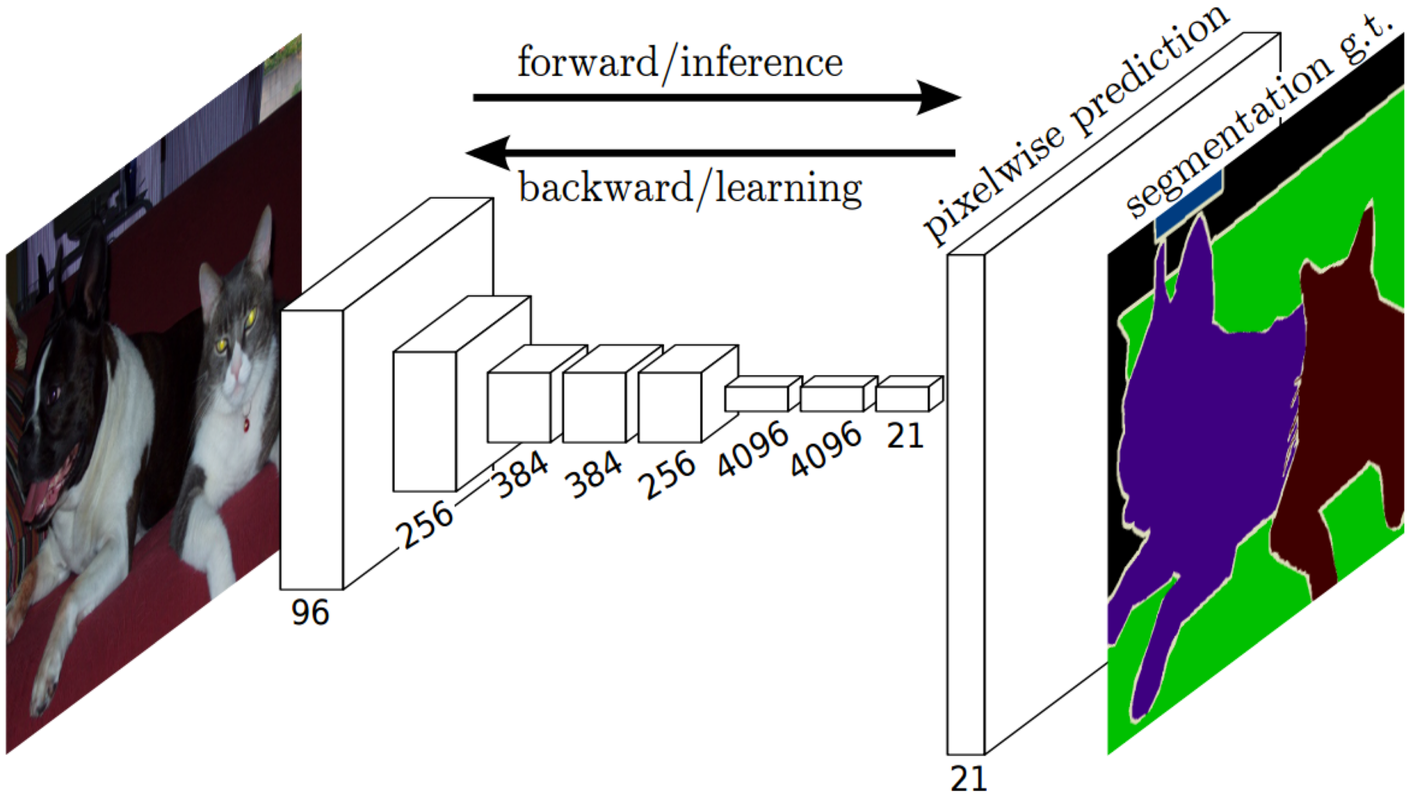
Residual Network: deeper is better





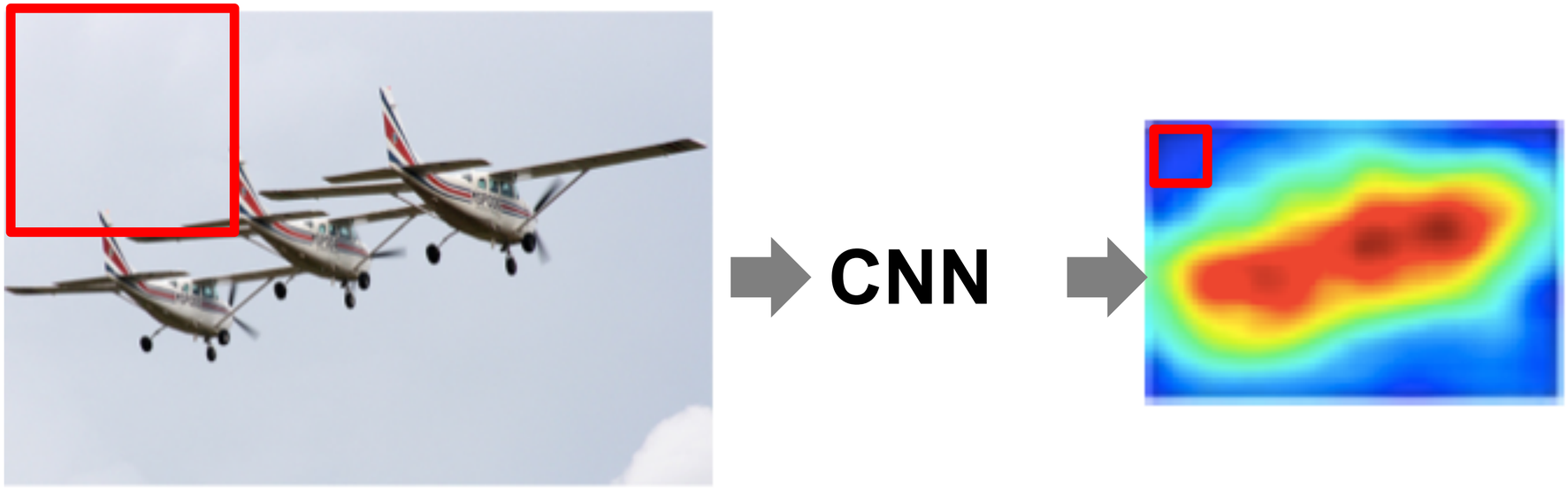
CNNs for Image Understanding

Fully-convolutional Neural Networks for X

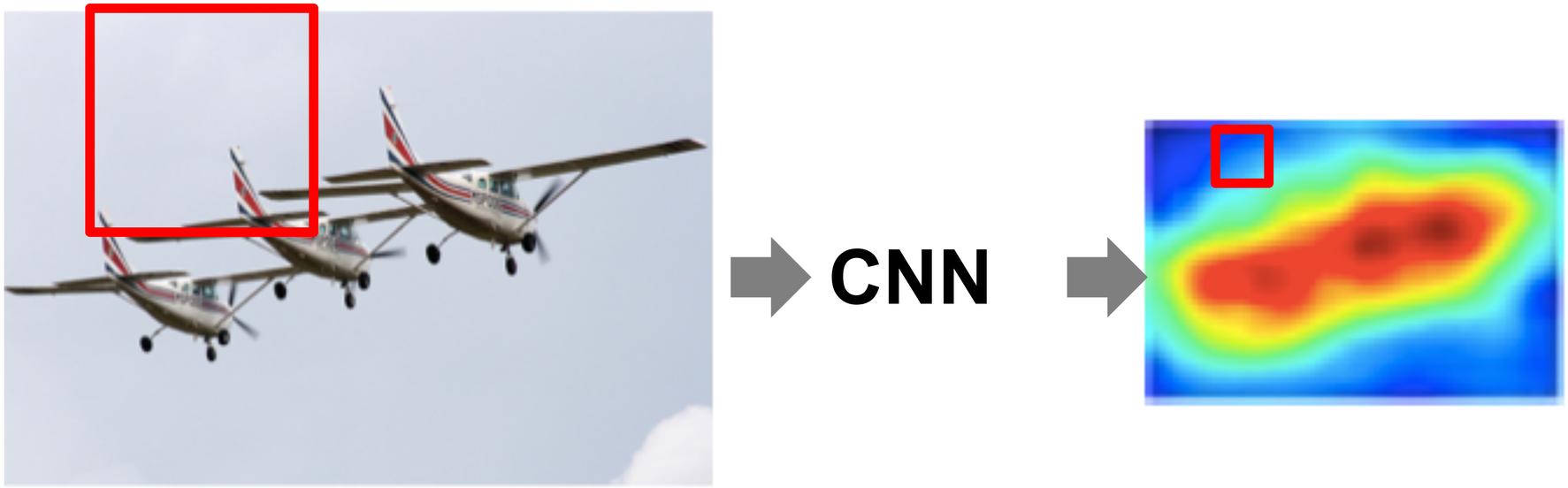


J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation.*CVPR*, 2015

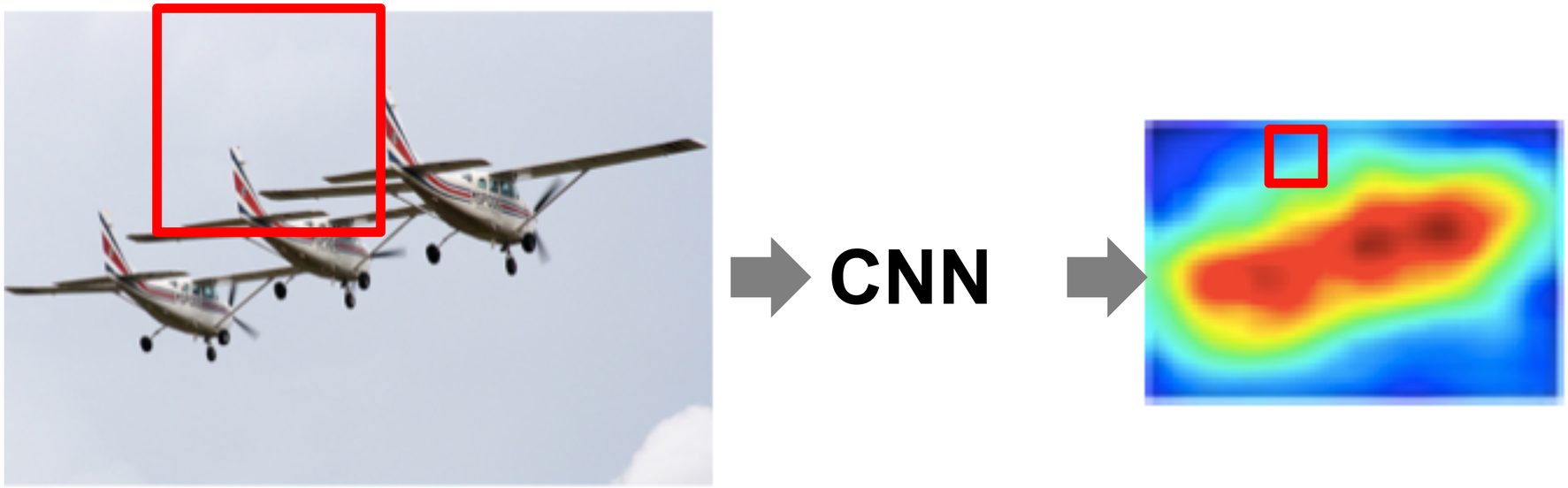
Fully-convolutional Neural Networks



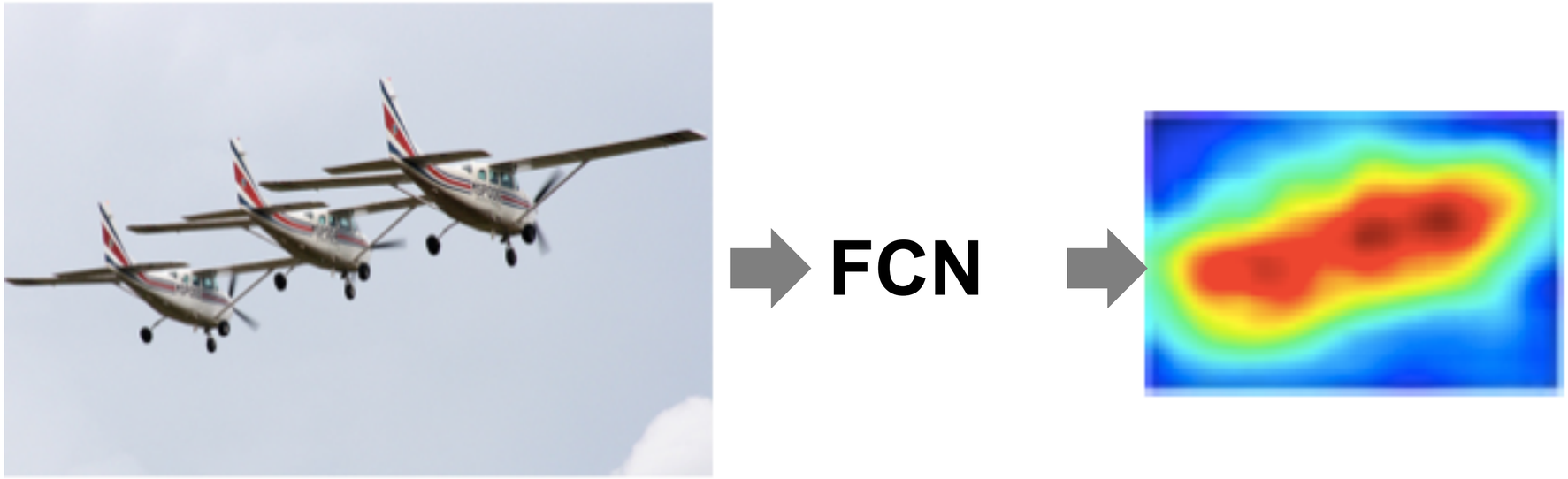
Fully-convolutional Neural Networks



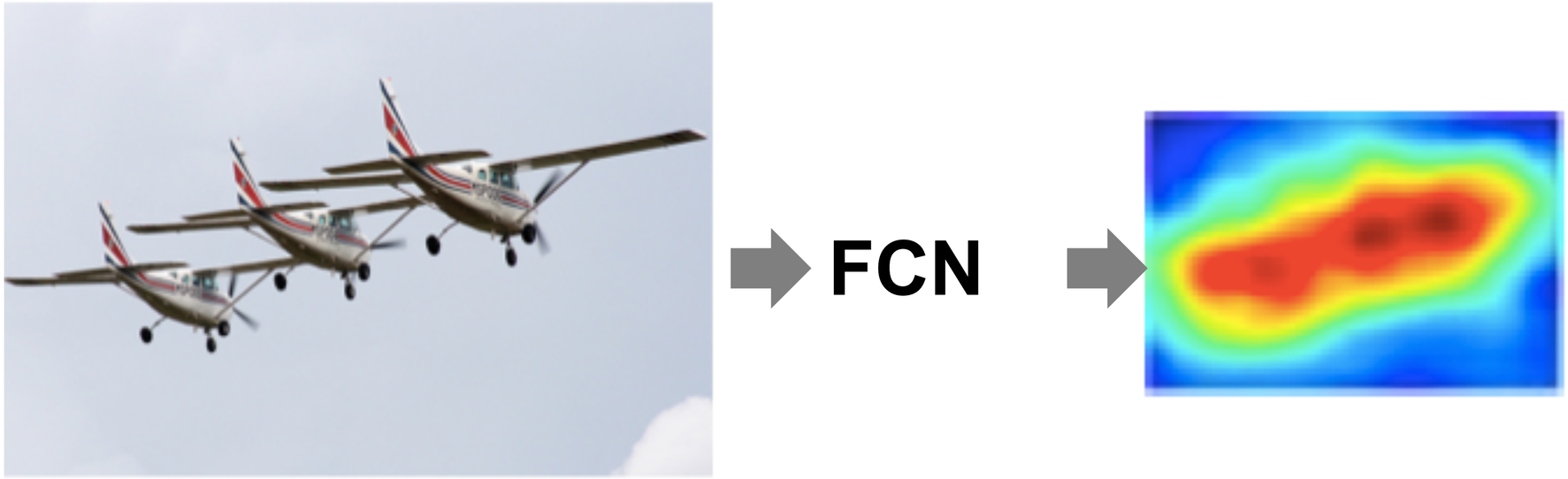
Fully-convolutional Neural Networks



Fully-convolutional Neural Networks



Fully-convolutional Neural Networks



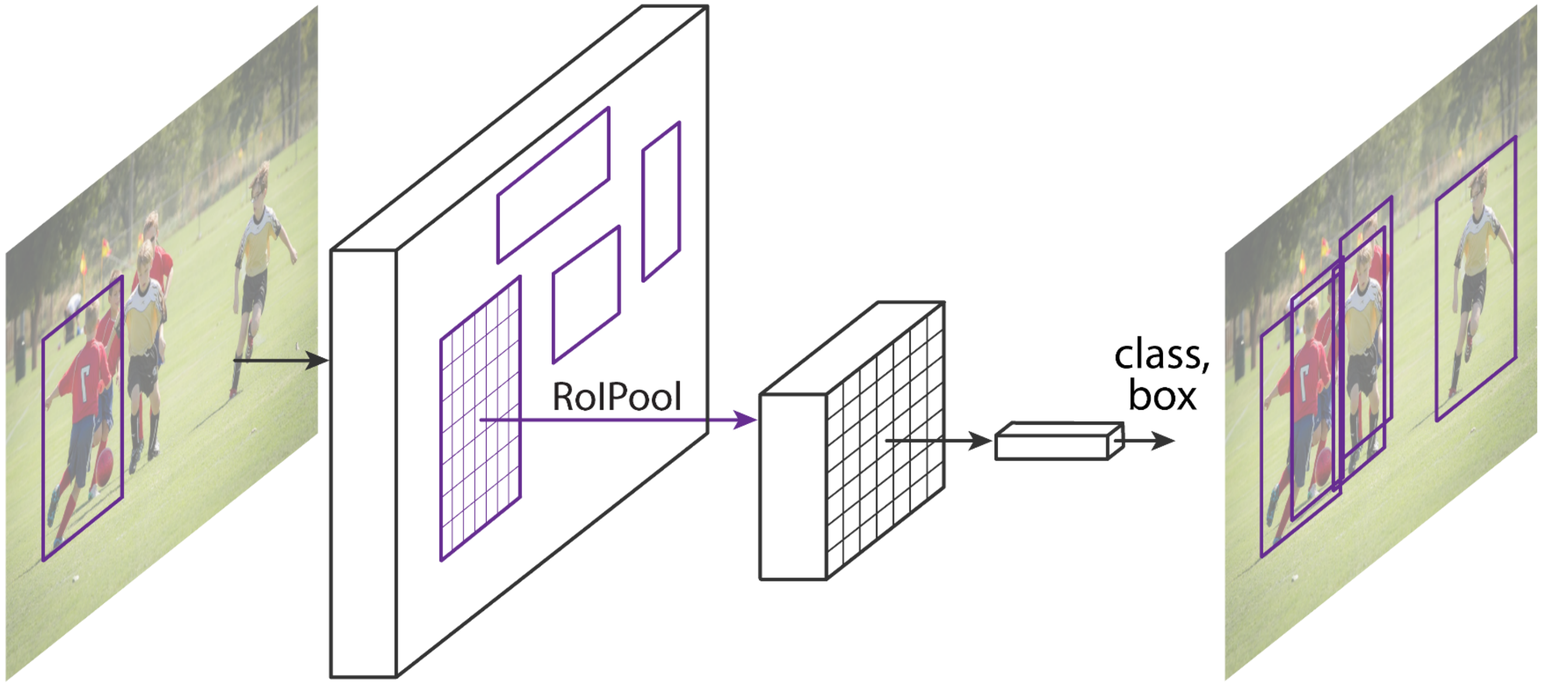
Fast (shared convolutions)
Simple (dense)

Deeplab v1,v2,v3: FCN-based semantic segmentation

DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs
Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, Alan L. Yuille, ICLR 2015

Object Detection: Fast(er)-RCNN

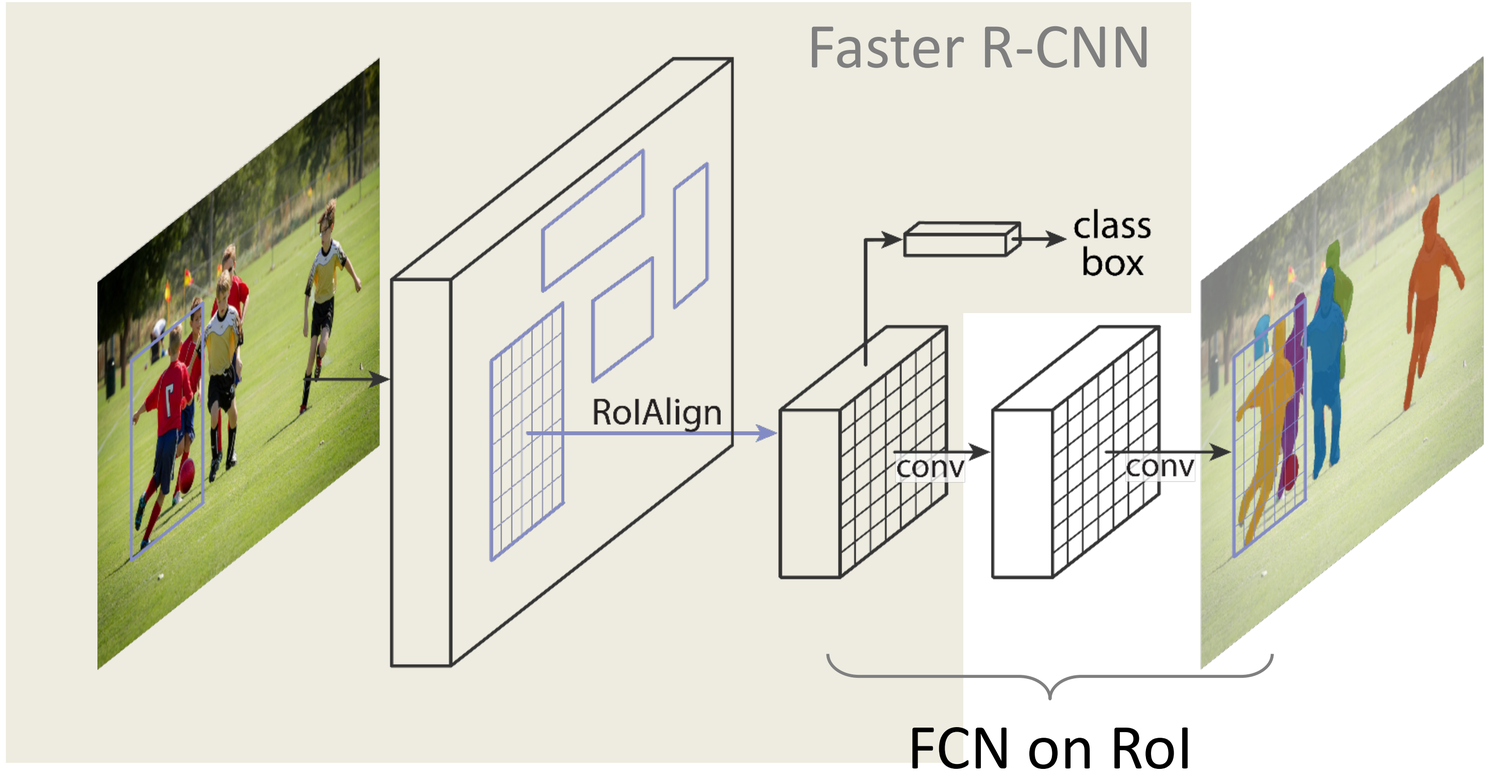
- Fast/Faster R-CNN
  - ✓ Good speed
  - ✓ Good accuracy
  - ✓ Intuitive
  - ✓ Easy to use



Ross Girshick. “Fast R-CNN”. ICCV 2015.
Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. NIPS 2015.

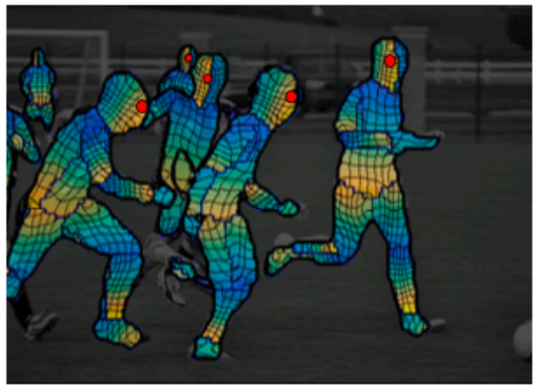
Mask R-CNN

- Mask R-CNN = **Faster R-CNN** with **FCN** on RoIs



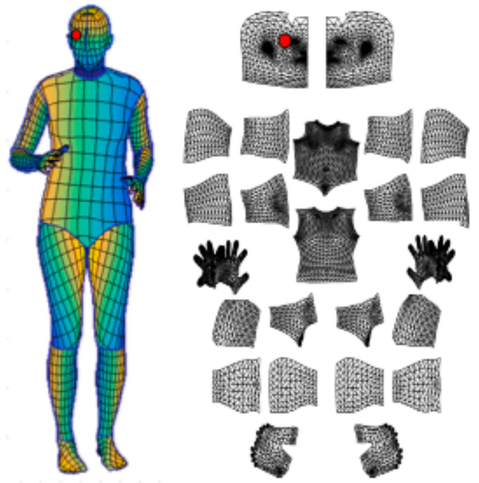
Mask R-CNN frame-by-frame

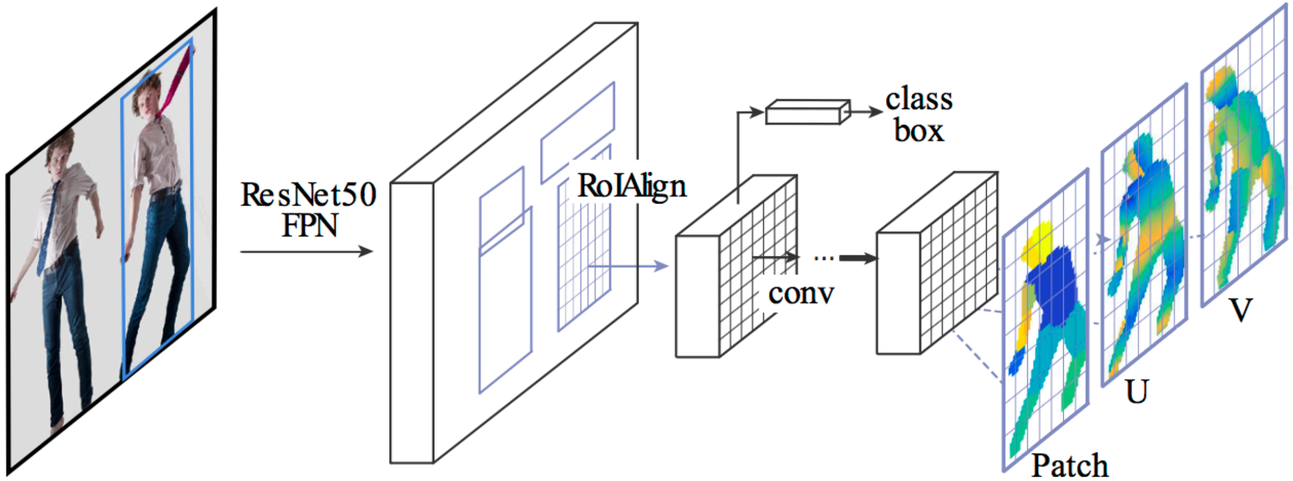
DensePose: dense image-to-body correspondence



DensePose-RCNN Results

DensePose COCO Dataset

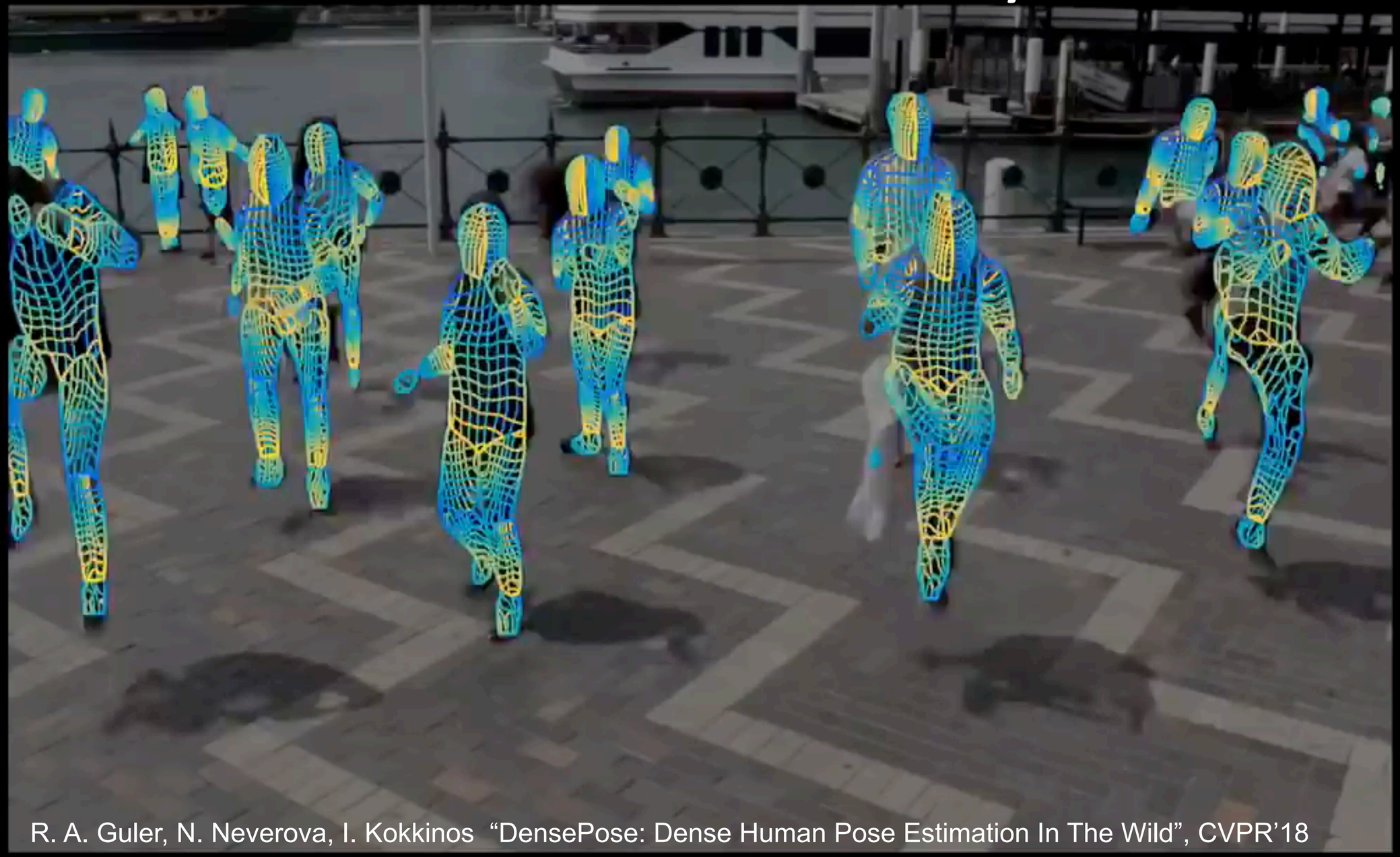




DensePose-RCNN: ~25 FPS

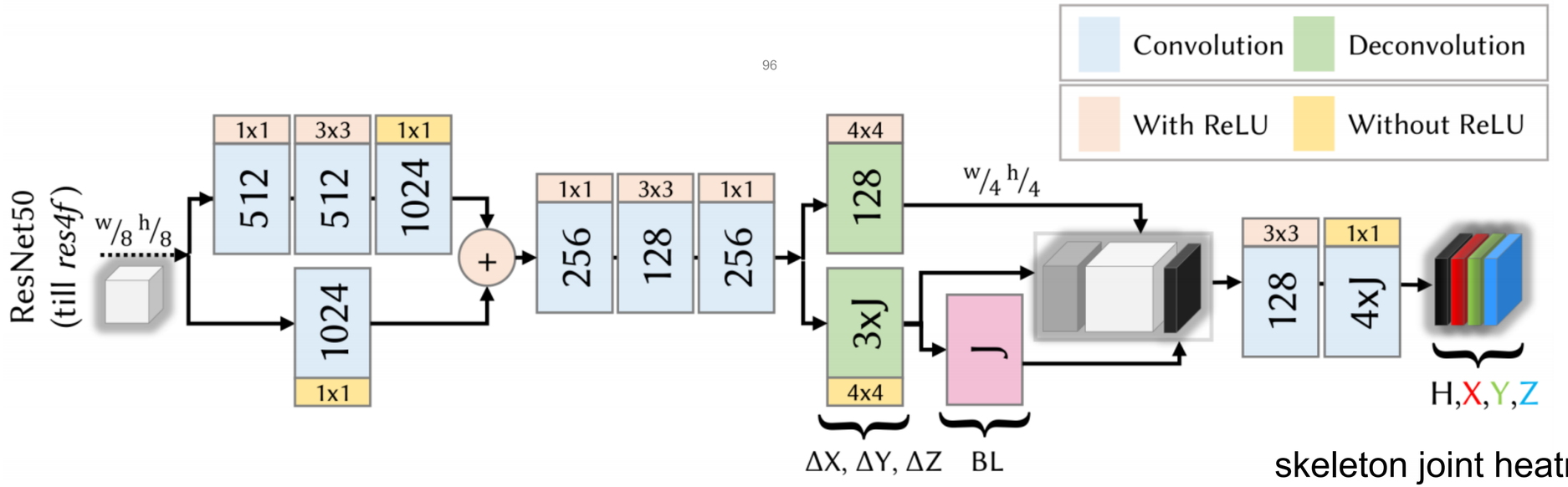
R. A. Guler, N. Neverova, I. Kokkinos
“DensePose: Dense Human Pose Estimation In The Wild”, CVPR’18

DensePose estimation frame-by-frame



R. A. Guler, N. Neverova, I. Kokkinos “DensePose: Dense Human Pose Estimation In The Wild”, CVPR’18

3D Pose Estimation: *VNECT*



VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera, Mehta et al., **SIGGRAPH 2017**

HoloPose estimation frame-by-frame

Ariel AI

loPose: Holistic 3D Human Reconstruction In-the-Wild, A. Guler and I. Kokkinos, CVPR 2019

Main frameworks

(Python, C++, Java)

(Python, backends support other languages)

(Python)

(C++, Python, Matlab)

Currently less frequently used

(Python)

(Python)

(Python, C++)

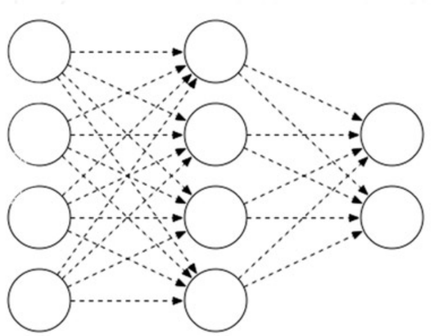
(Python, C++, C#)

(Matlab)

(Python, Java, Scala)

(Python, C++, and others)

Course Information (slides/code/comments)



http://geometry.cs.ucl.ac.uk/creativeai/

Scan me

Back-propagation Algorithm

Chain rule

$$x \xrightarrow{g} u \xrightarrow{f} y$$

y is affected by x through intermediate quantity, u:

$$u = g(x) \quad y = f(u)$$

Calculus: $(f(g(x)))' = f'(g(x))g'(x)$

Rewrite:

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$$

Chain rule of differentiation– multiple variables

x(t),y(t) coordinates: given by GPS

z=f(x,y) given by map

Q: what is your speed in the vertical direction?

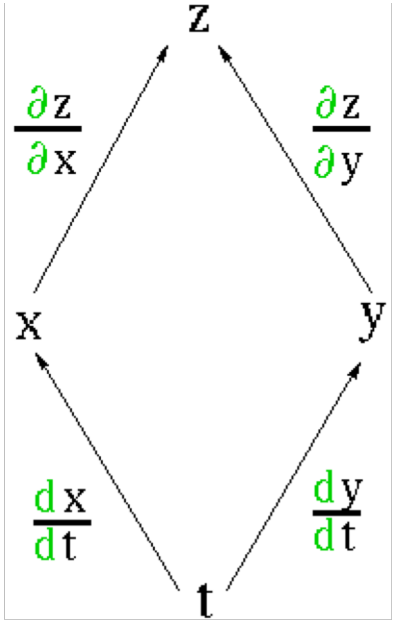
https://www.math.hmc.edu/calculus/tutorials/multichainrule/

Chain rule of differentiation– multiple variables

x(t),y(t) coordinates: given by GPS

z=f(x,y) given by map

Q: what is your speed in the vertical direction?

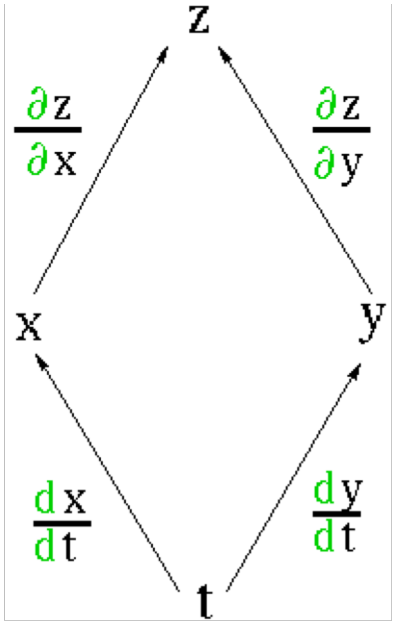


https://www.math.hmc.edu/calculus/tutorials/multichainrule/

Chain rule of differentiation– multiple variables

Let $x = x(t)$ and $y = y(t)$ be differentiable at t and suppose that $z = f(x, y)$ is differentiable at $(x(t), y(t))$. Then $z = f(x(t), y(t))$ is differentiable at t and

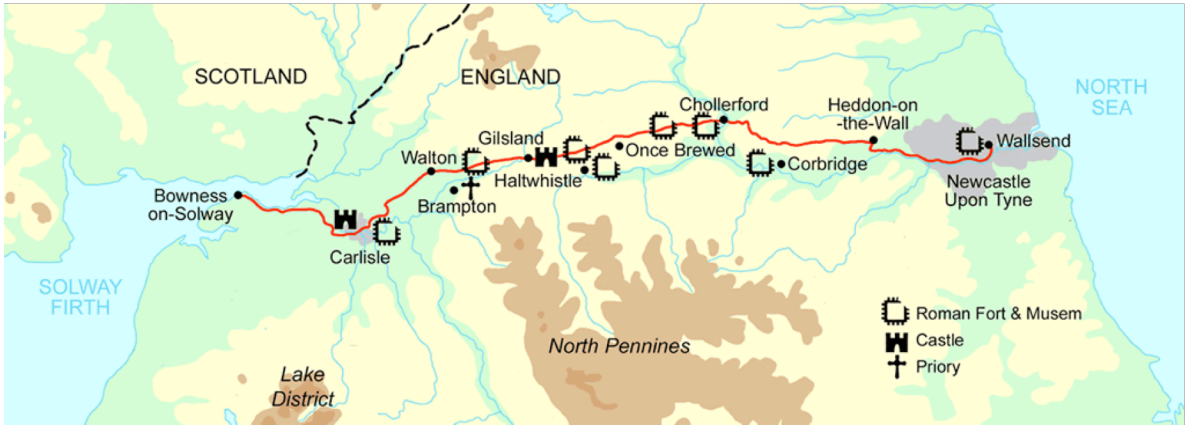
$$\frac{dz}{dt} = \frac{\partial z}{\partial x}\frac{dx}{dt} + \frac{\partial z}{\partial y}\frac{dy}{dt}.$$



x(t),y(t) coordinates: given by GPS

z=f(x,y) given by map

Q: what is your speed in the vertical direction?

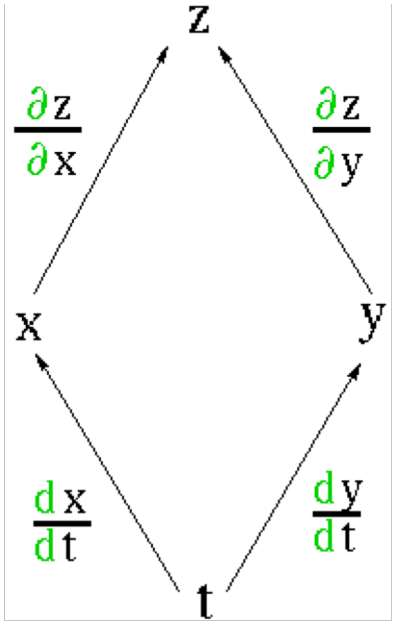


https://www.math.hmc.edu/calculus/tutorials/multichainrule/

Chain rule of differentiation– multiple variables

Let $x = x(t)$ and $y = y(t)$ be differentiable at t and suppose that $z = f(x, y)$ is differentiable at $(x(t), y(t))$. Then $z = f(x(t), y(t))$ is differentiable at t and

$$\frac{dz}{dt} = \frac{\partial z}{\partial x}\frac{dx}{dt} + \frac{\partial z}{\partial y}\frac{dy}{dt}.$$



Let $z = x^2y - y^2$ where $x = t^2$ and $y = 2t$. Then

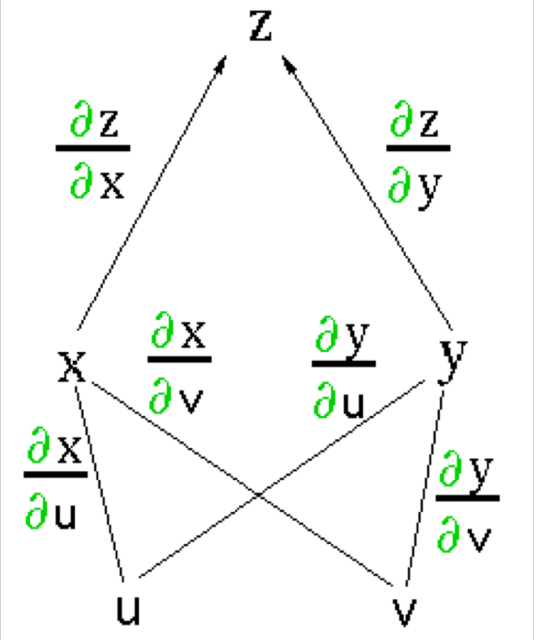
$$\begin{aligned}
\frac{dz}{dt} &= \frac{\partial z}{\partial x}\frac{dx}{dt} + \frac{\partial z}{\partial y}\frac{dy}{dt} \\
&= (2xy)(2t) + (x^2 - 2y)(2) \\
&= (2t^2 \cdot 2t)(2t) + \left((t^2)^2 - 2(2t)\right)(2) \\
&= 8t^4 + 2t^4 - 8t \\
&= 10t^4 - 8t.
\end{aligned}$$

https://www.math.hmc.edu/calculus/tutorials/multichainrule/

Chain rule of derivative – multiple variables

Let $x = x(u,v)$ and $y = y(u,v)$ have first-order partial derivatives at the point (u,v) and suppose that $z = f(x,y)$ is differentiable at the point $(x(u,v), y(u,v))$. Then $f(x(u,v), y(u,v))$ has first-order partial derivatives at (u,v) given by

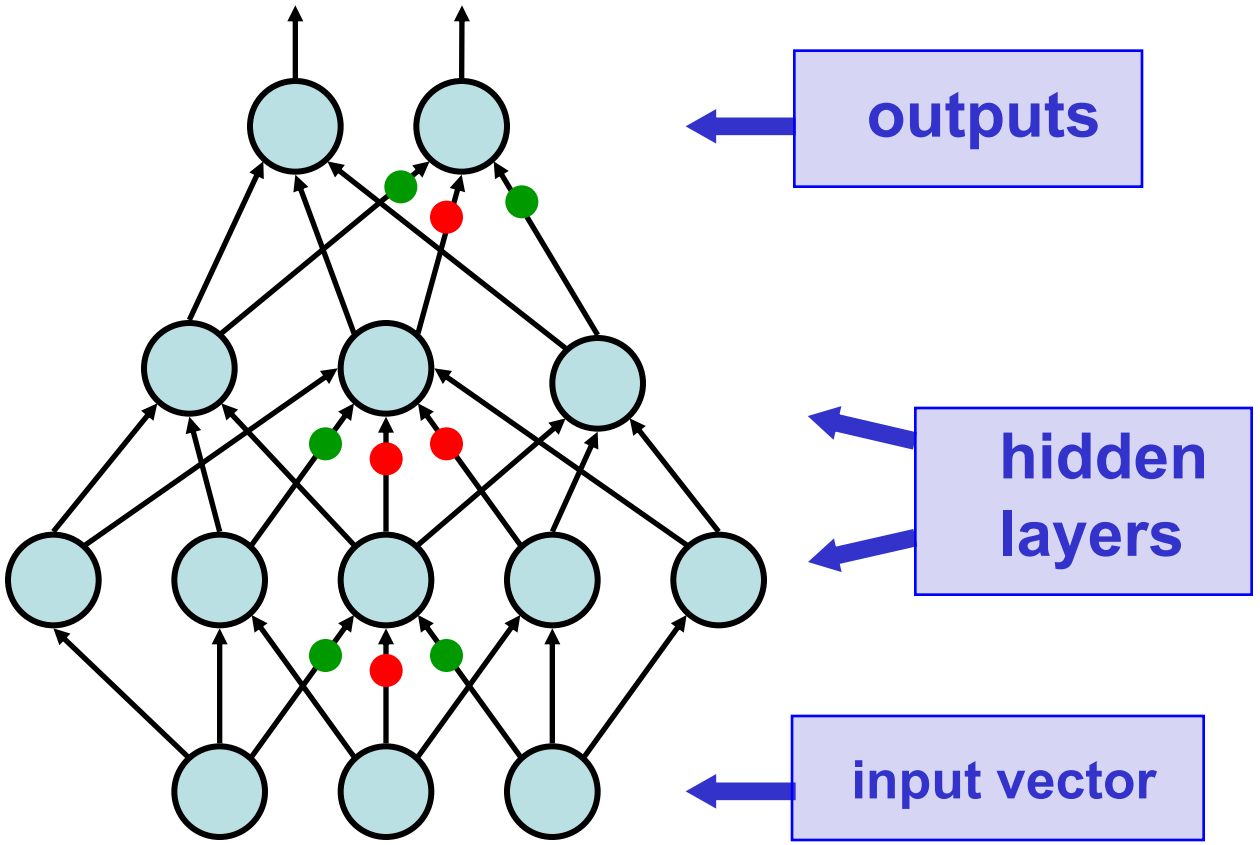
$$\frac{\partial z}{\partial u} = \frac{\partial z}{\partial x}\frac{\partial x}{\partial u} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial u}$$
$$\frac{\partial z}{\partial v} = \frac{\partial z}{\partial x}\frac{\partial x}{\partial v} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial v}.$$



https://www.math.hmc.edu/calculus/tutorials/multichainrule/

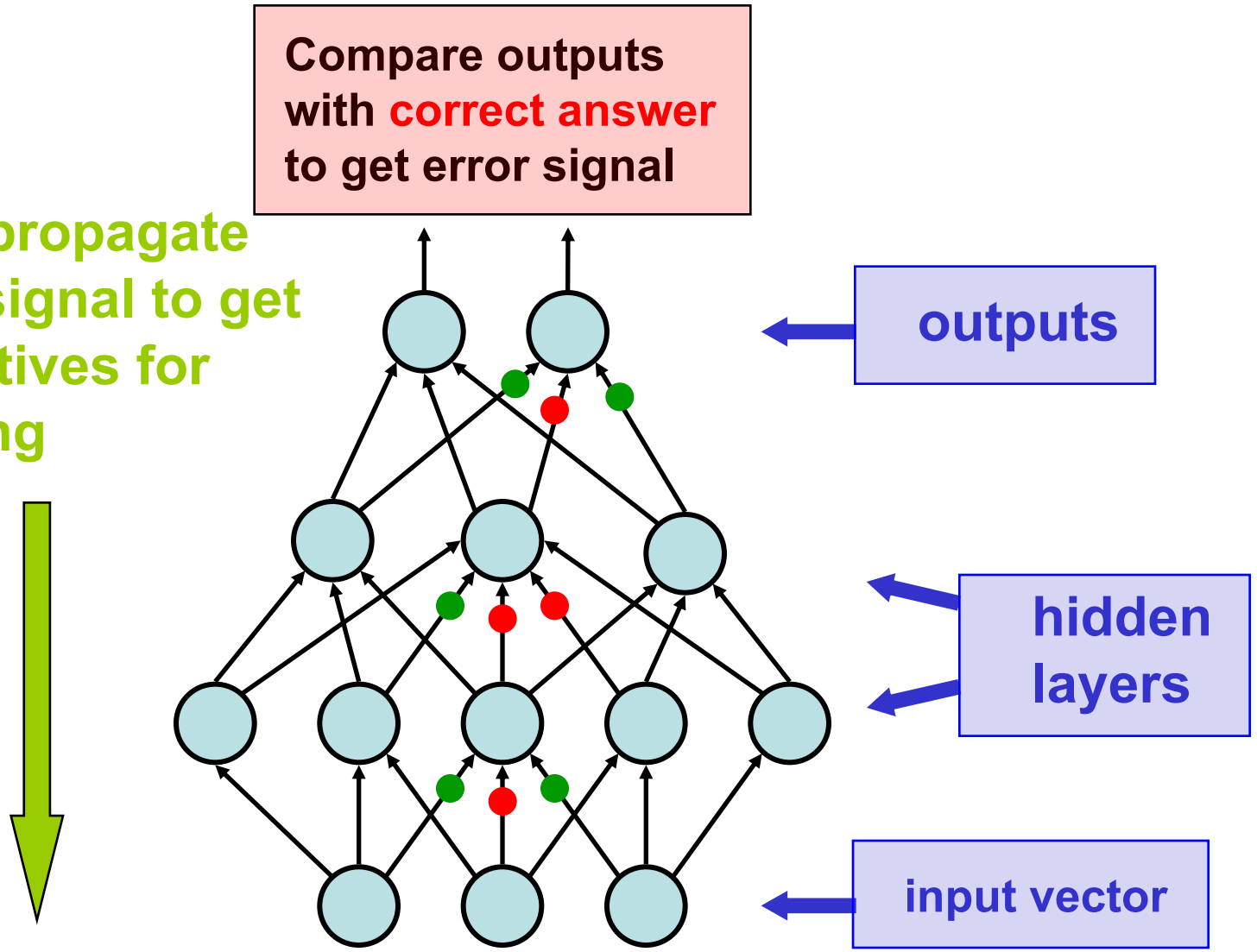
Multi-Layer Perceptrons

$$u_i = g\left(\sum_{k\in\mathcal{N}(i)} w_{k,i}\, g\left(\sum_{m\in\mathcal{N}(k)} w_{m,k}u_m + b_k\right) + b_i\right)$$



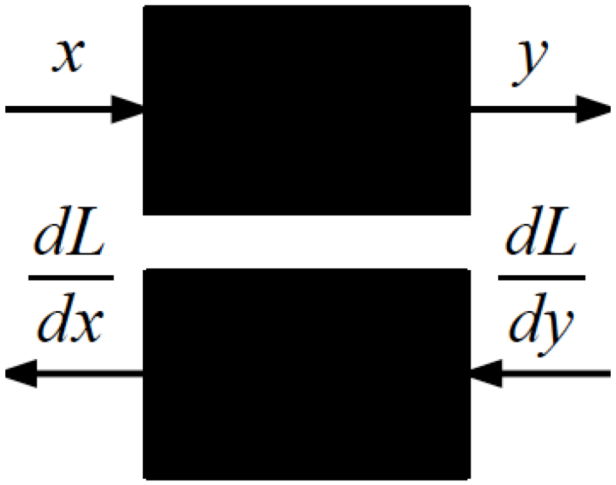
Slide credits: G. Hinton

Multi-Layer Perceptrons (~1985)



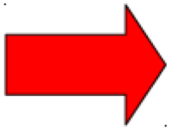
Slide credits: G. Hinton

Chain Rule

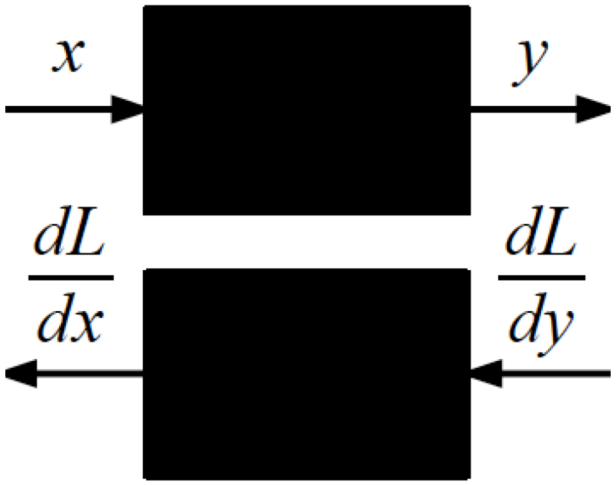


Given $y(x)$ and dL/dy,

What is dL/dx ?



Chain Rule

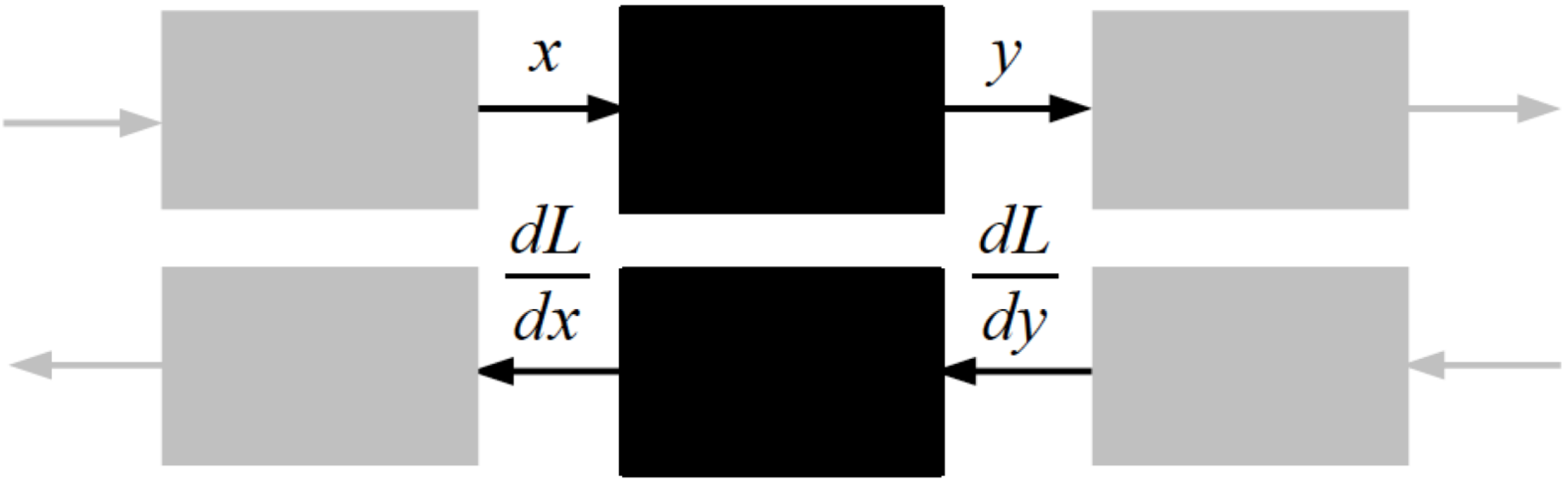


Given $y(x)$ and dL/dy,

What is dL/dx ?

$$\frac{dL}{dx} = \frac{dL}{dy} \cdot \frac{dy}{dx}$$

‘another brick in the wall’



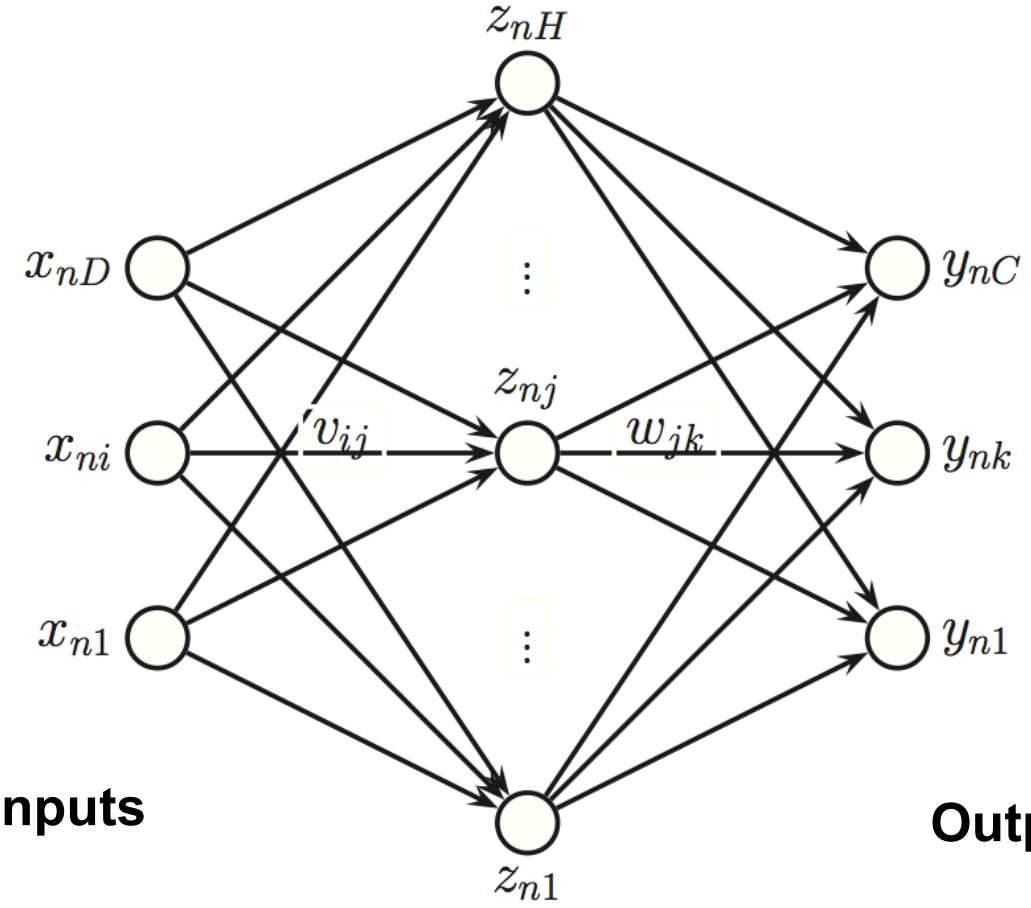
Given $y(x)$ and dL/dy,

What is dL/dx ?

$$\frac{dL}{dx} = \frac{dL}{dy} \cdot \frac{dy}{dx}$$

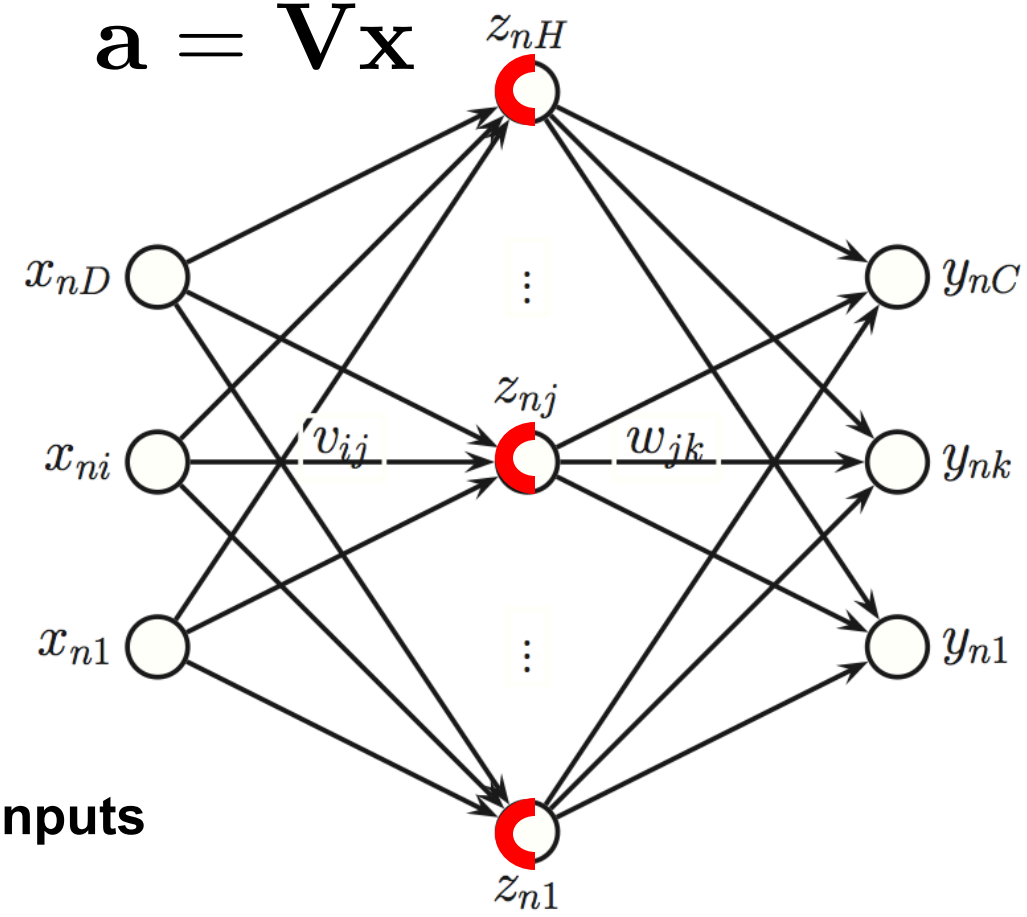
A neural network for multi-way classification

$$\mathbf{x}_n \xrightarrow{\mathbf{V}} \mathbf{a}_n \xrightarrow{g} \mathbf{z}_n \xrightarrow{\mathbf{W}} \mathbf{b}_n \xrightarrow{h} \hat{\mathbf{y}}_n$$



A neural network in forward mode:

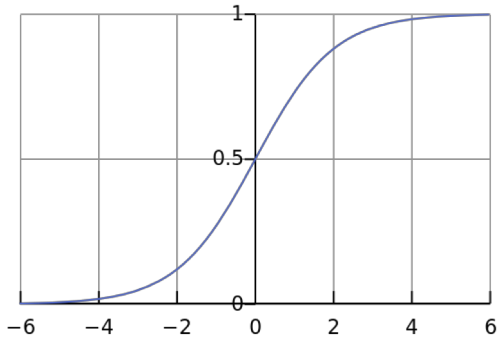
$$\mathbf{a} = \mathbf{V}\mathbf{x}$$

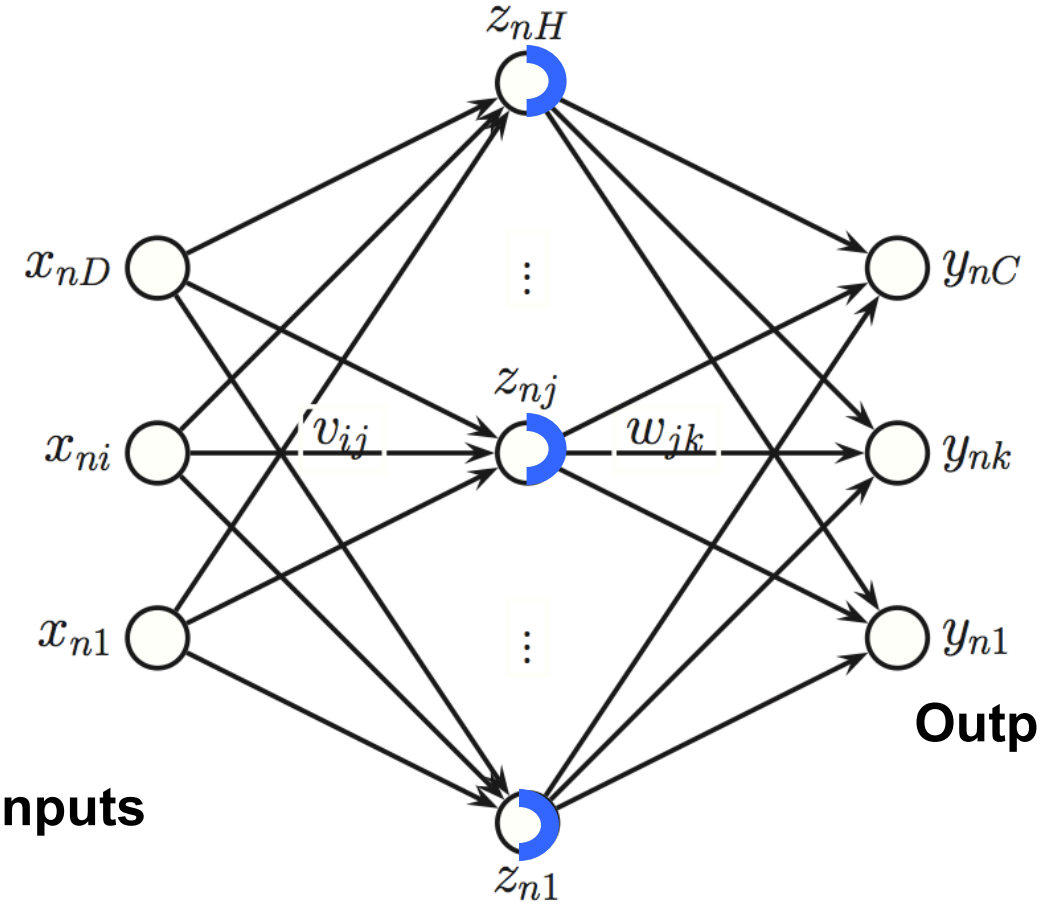


A neural network in forward mode:

$$\mathbf{z} = g(\mathbf{a})$$

$$z_k = \frac{1}{1 + \exp(-a_k)}$$



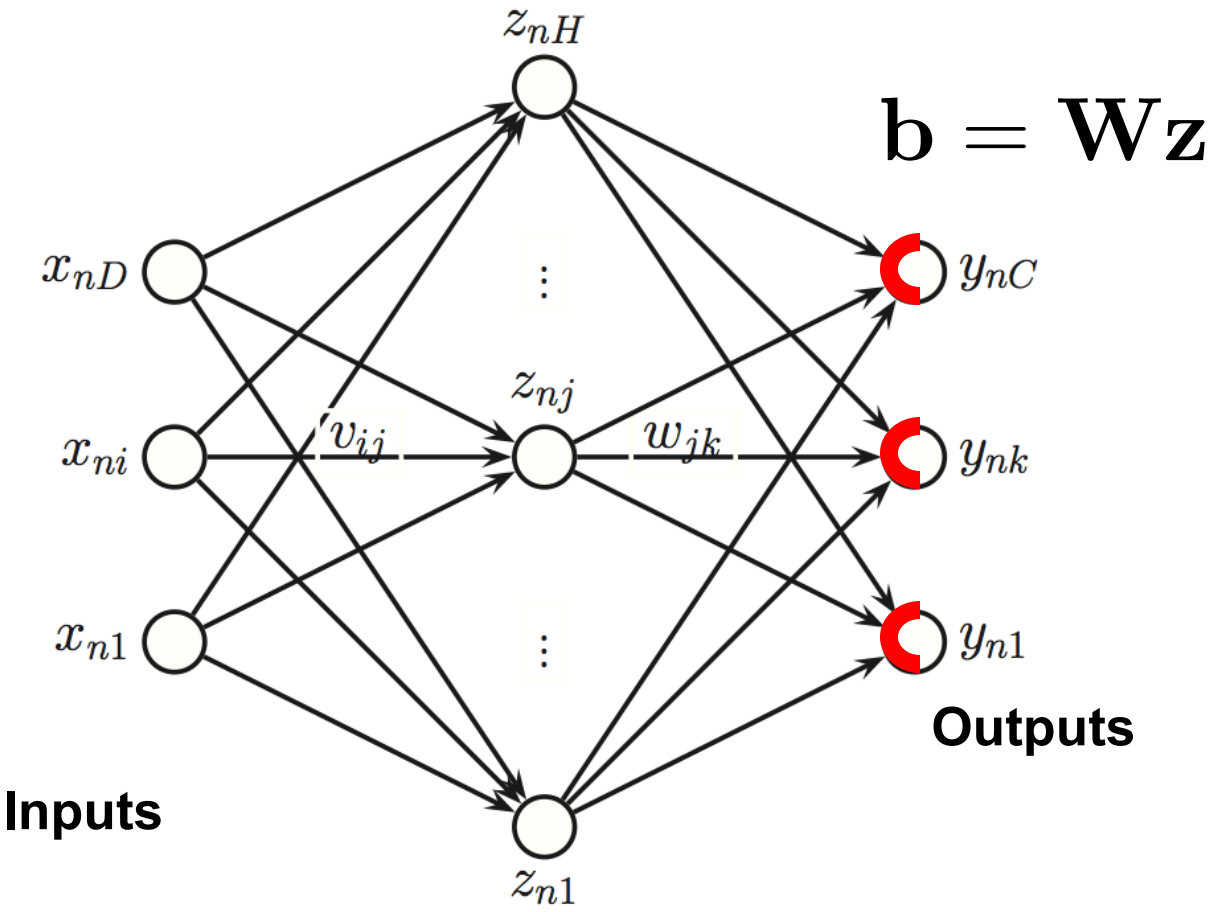


Outputs

Inputs

Hidden layer

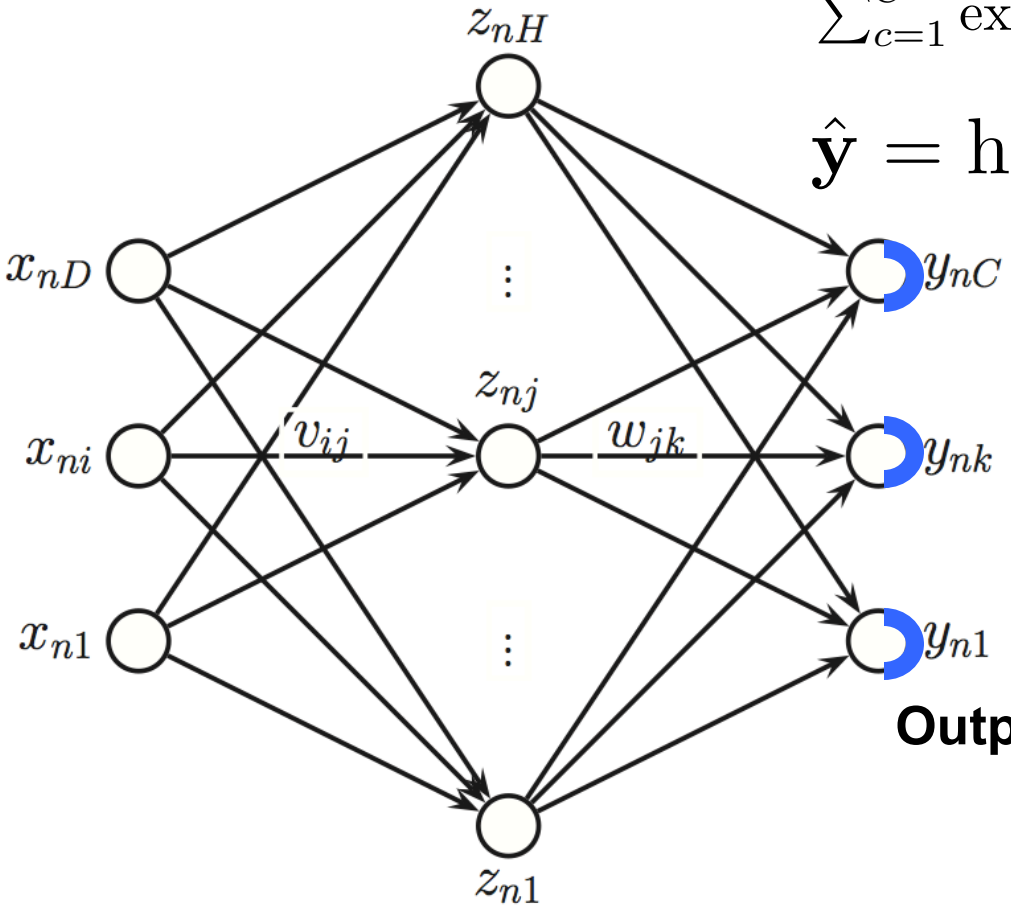
A neural network in forward mode:

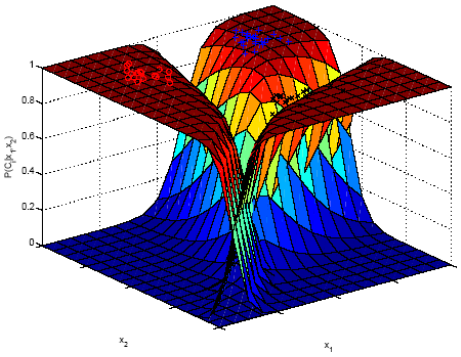


A neural network in forward mode:

$$\hat{y}_k = \frac{\exp(b_k)}{\sum_{c=1}^{C} \exp(b_c)}$$

$$\hat{\mathbf{y}} = \mathrm{h}(\mathbf{b})$$





Outputs

Hidden layer

Training objective, multi-class classification

One-hot label encoding: $\mathbf{y}^i = (0, 0, 1, 0)$

Likelihood of training sample: $(\mathbf{y}^i, \mathbf{x}^i)$

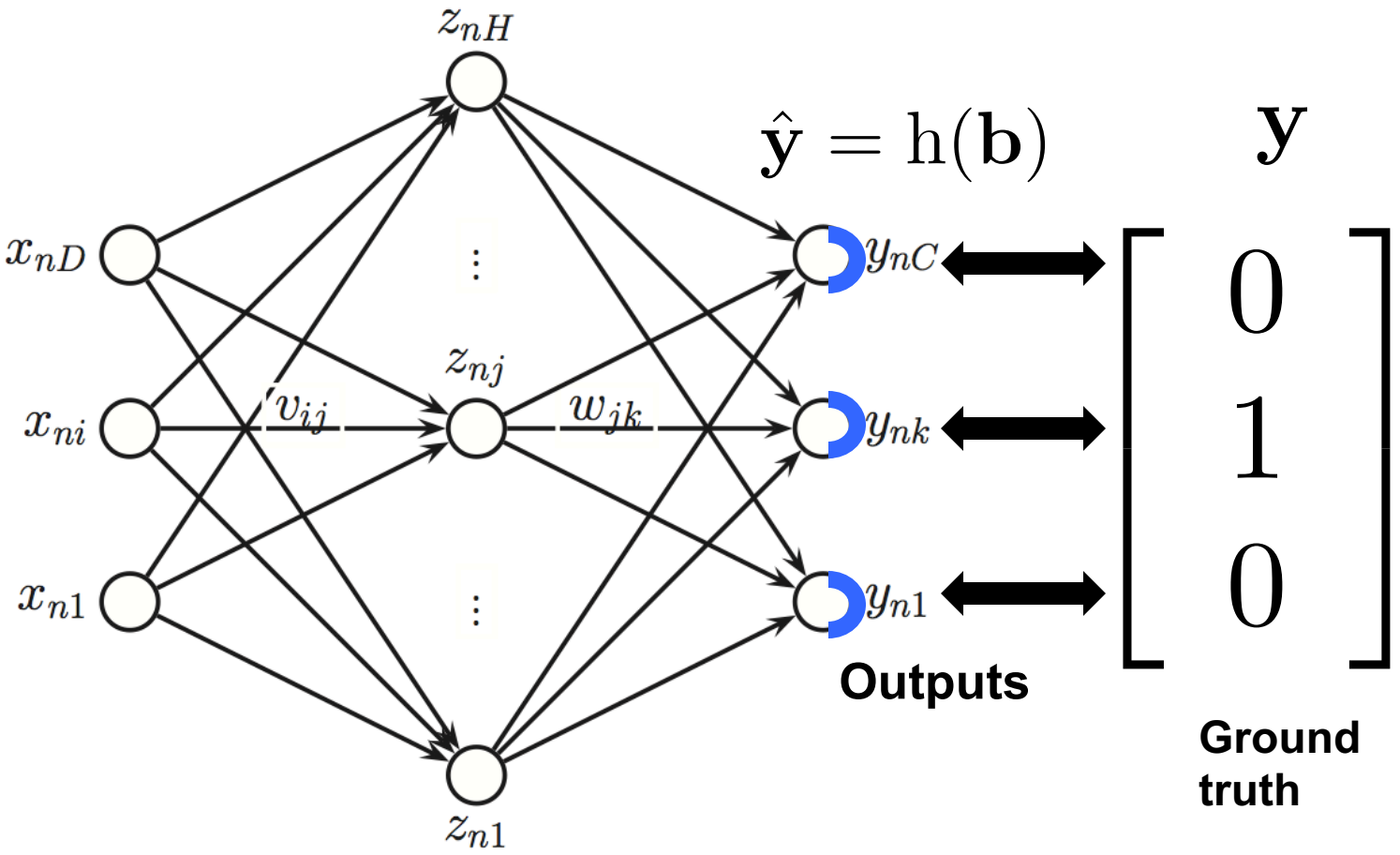
$$P(\mathbf{y}^i | \mathbf{x}^i; \mathbf{w}) = \prod_{c=1}^{C} (g_c(\mathbf{x}, \mathbf{W}))^{\mathbf{y}_c^i}$$

Optimization criterion:

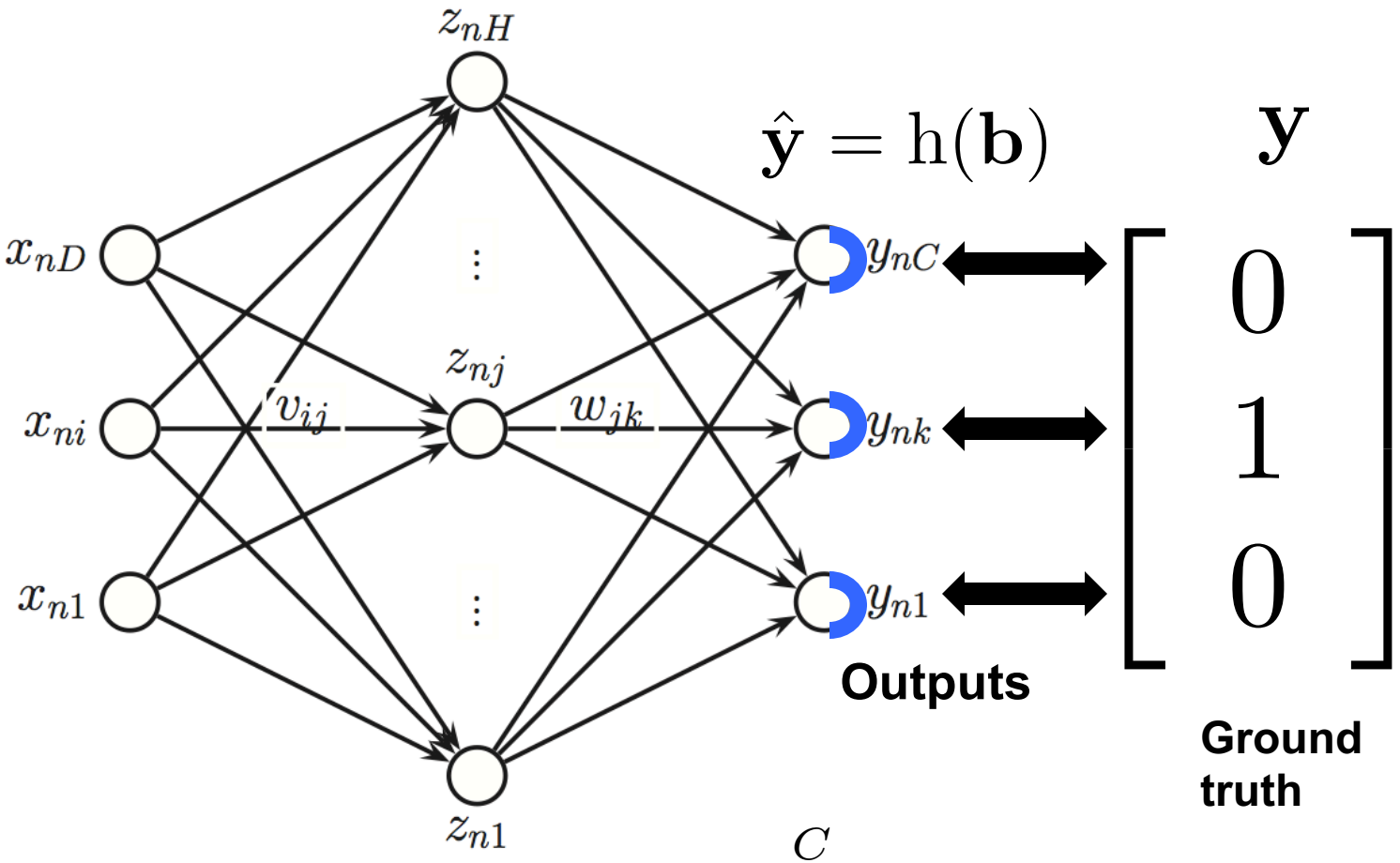
$$L(\mathbf{W}) = -\sum_{i=1}^{N} \sum_{c=1}^{C} \mathbf{y}_c^i \log (g_c(\mathbf{x}, \mathbf{W}))$$

Parameter estimation: Gradient of L with respect to **W**

Objective for multi-class classification

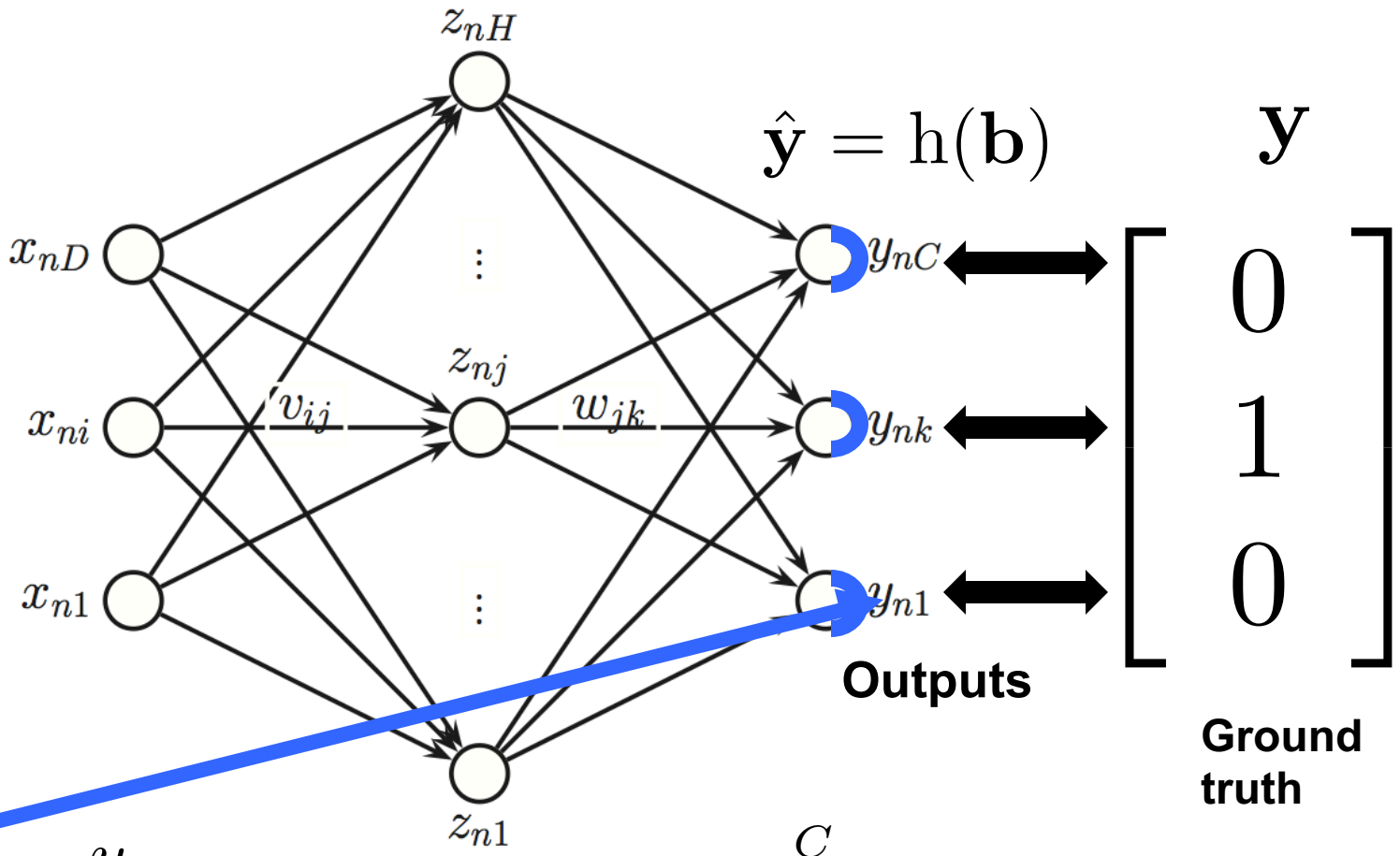


Objective for multi-class classification



$$L(\mathbf{W}) = -\sum_{c=1}^{C} y_c \log\left(\hat{y}_c(\mathbf{x}; \mathbf{W})\right)$$

Derivative of loss w.r.t. top-layer neurons

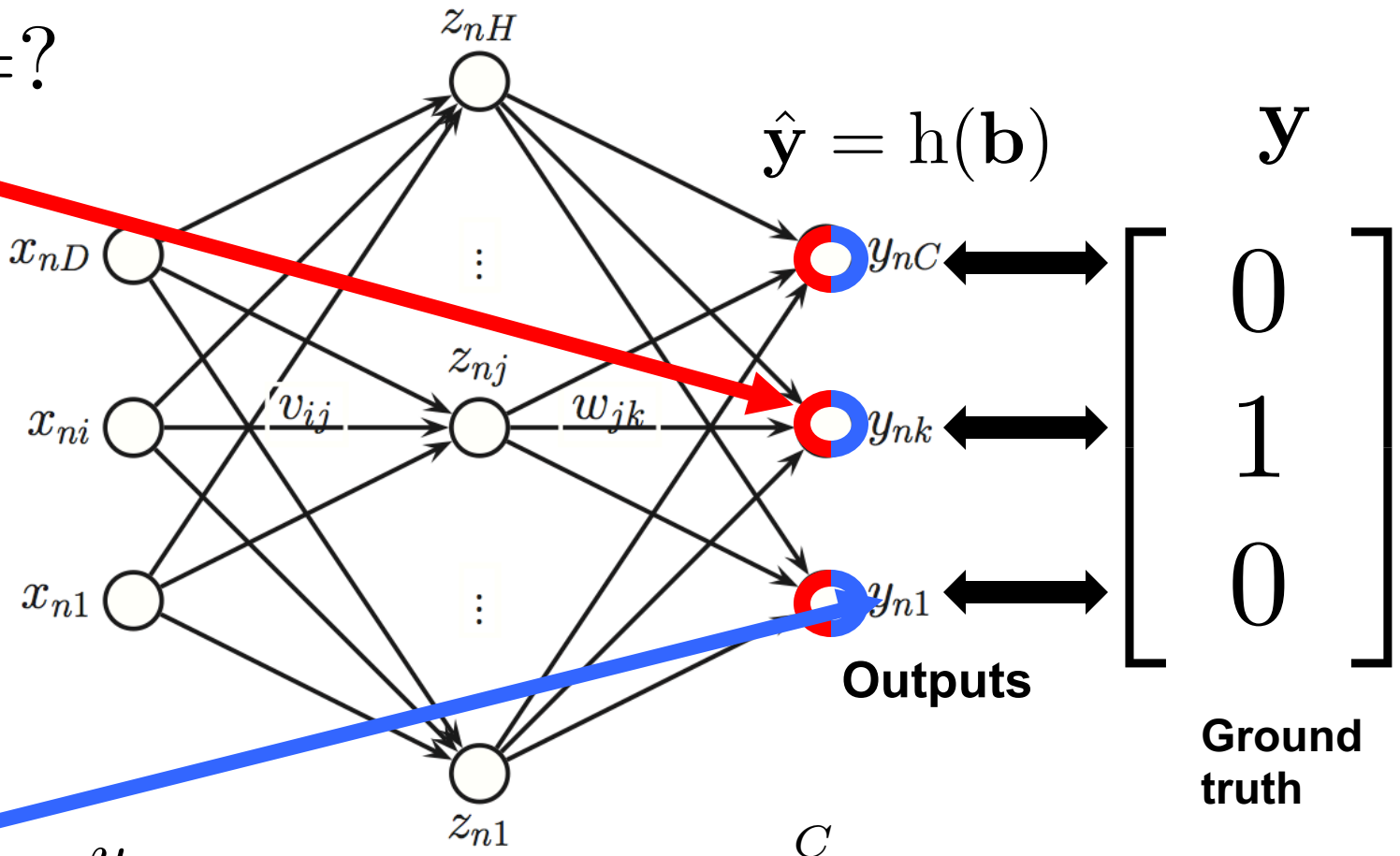


$$\frac{\partial L}{\partial \hat{y}_c} = -\frac{y_c}{\hat{y}_c}$$

$$L(\mathbf{W}) = -\sum_{c=1}^{C} y_c \log\left(\hat{y}_c(\mathbf{x}; \mathbf{W})\right)$$

A neural network in backward mode:

$$\frac{\partial L}{\partial b_k} = ?$$

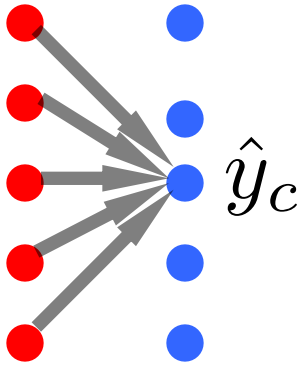


$$\frac{\partial L}{\partial \hat{y}_c} = -\frac{y_c}{\hat{y}_c}$$

$$L(\mathbf{W}) = -\sum_{c=1}^{C} y_c \log\left(\hat{y}_c(\mathbf{x}; \mathbf{W})\right)$$

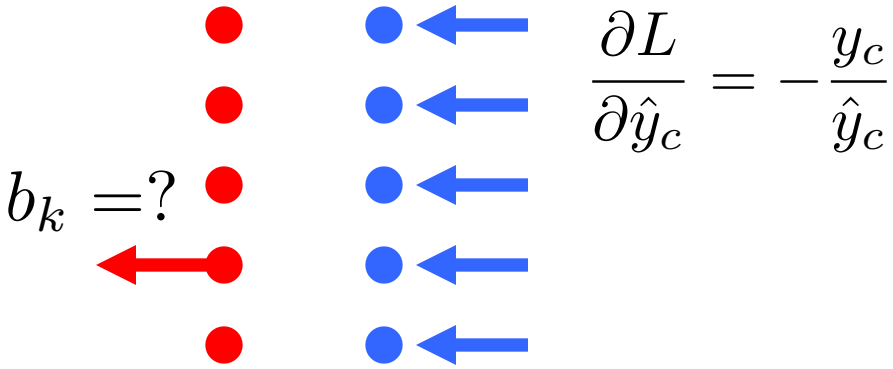
Softmax in forward mode: all for one

$$\hat{y}_c = \frac{\exp(b_c)}{\sum_{c'=1}^{C} \exp(b'_c)}$$



Softmax in backward mode?

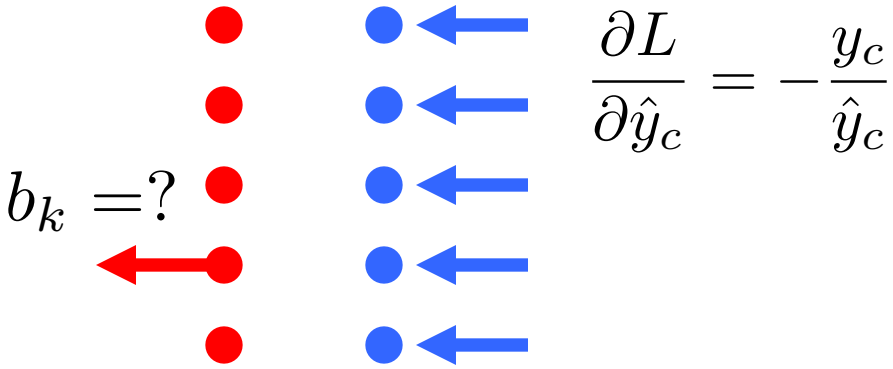
$$\hat{y}_c = \frac{\exp(b_c)}{\sum_{c'=1}^{C} \exp(b'_c)}$$



$$\frac{\partial L}{\partial \hat{y}_c} = -\frac{y_c}{\hat{y}_c}$$

Softmax in backward mode?

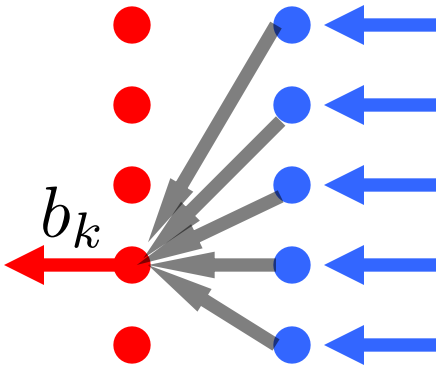
$$\hat{y}_c = \frac{\exp(b_c)}{\sum_{c'=1}^{C} \exp(b'_c)}$$



$$\frac{\partial L}{\partial \hat{y}_c} = -\frac{y_c}{\hat{y}_c}$$

Softmax in backward mode: one from all

$$\hat{y}_c = \frac{\exp(b_c)}{\sum_{c'=1}^{C} \exp(b'_c)}$$



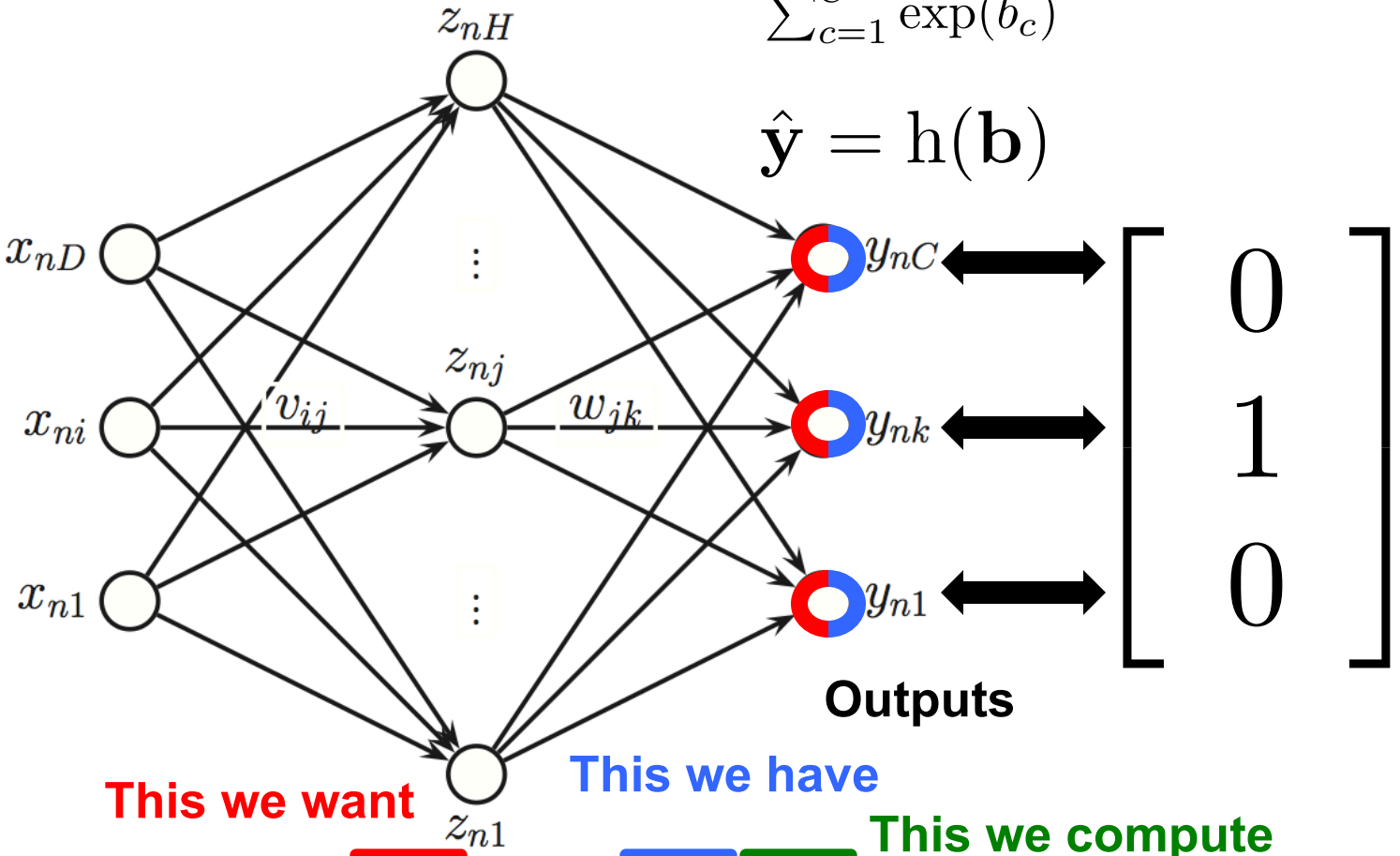
$$\frac{\partial L}{\partial \hat{y}_c} = -\frac{y_c}{\hat{y}_c}$$

$$\frac{\partial L}{\partial b_k} = \sum_c \frac{\partial L}{\partial \hat{y}_c} \frac{\partial \hat{y}_c}{\partial b_k}$$

A neural network in backward mode:

$$\hat{y}_k = \frac{\exp(b_k)}{\sum_{c=1}^{C} \exp(b_c)}$$

$$\hat{\mathbf{y}} = \mathrm{h}(\mathbf{b})$$



This we want

This we have

This we compute

$$\frac{\partial L}{\partial \hat{y}_c} = -\frac{y_c}{\hat{y}_c}$$

$$\frac{\partial L}{\partial b_k} = \sum_c \frac{\partial L}{\partial \hat{y}_c} \frac{\partial \hat{y}_c}{\partial b_k} = \hat{y}_k - y_k$$

In backward mode?

$$\frac{\partial L}{\partial b_k} = \sum_c \frac{\partial L}{\partial \hat{y}_c} \frac{\partial \hat{y}_c}{\partial b_k} \qquad \hat{y}_c = \frac{\exp(b_c)}{\sum_{c'=1}^{C} \exp(b'_c)} \qquad \frac{\partial L}{\partial \hat{y}_c} = -\frac{y_c}{\hat{y}_c}$$

$$\frac{\partial y_c}{\partial b_k} = \frac{\frac{\partial \exp(b_c)}{\partial b_k}}{\sum_{c'=1}^{C} \exp(b_{c'})} - \frac{\exp(b_c) \frac{\partial \sum_{c'=1}^{C} \exp(b_{c'})}{\partial b_k}}{(\sum_{c'=1}^{C} \exp(b_{c'}))^2}$$

$$= \frac{[c=k]\exp(b_k)}{\sum_{c'} \exp(b'_c)} - \frac{\exp(b_c)}{\sum_{c'=1}^{C} \exp(b_{c'})} \frac{\exp(b_k)}{\sum_{c'=1}^{C} \exp(b_{c'})}$$

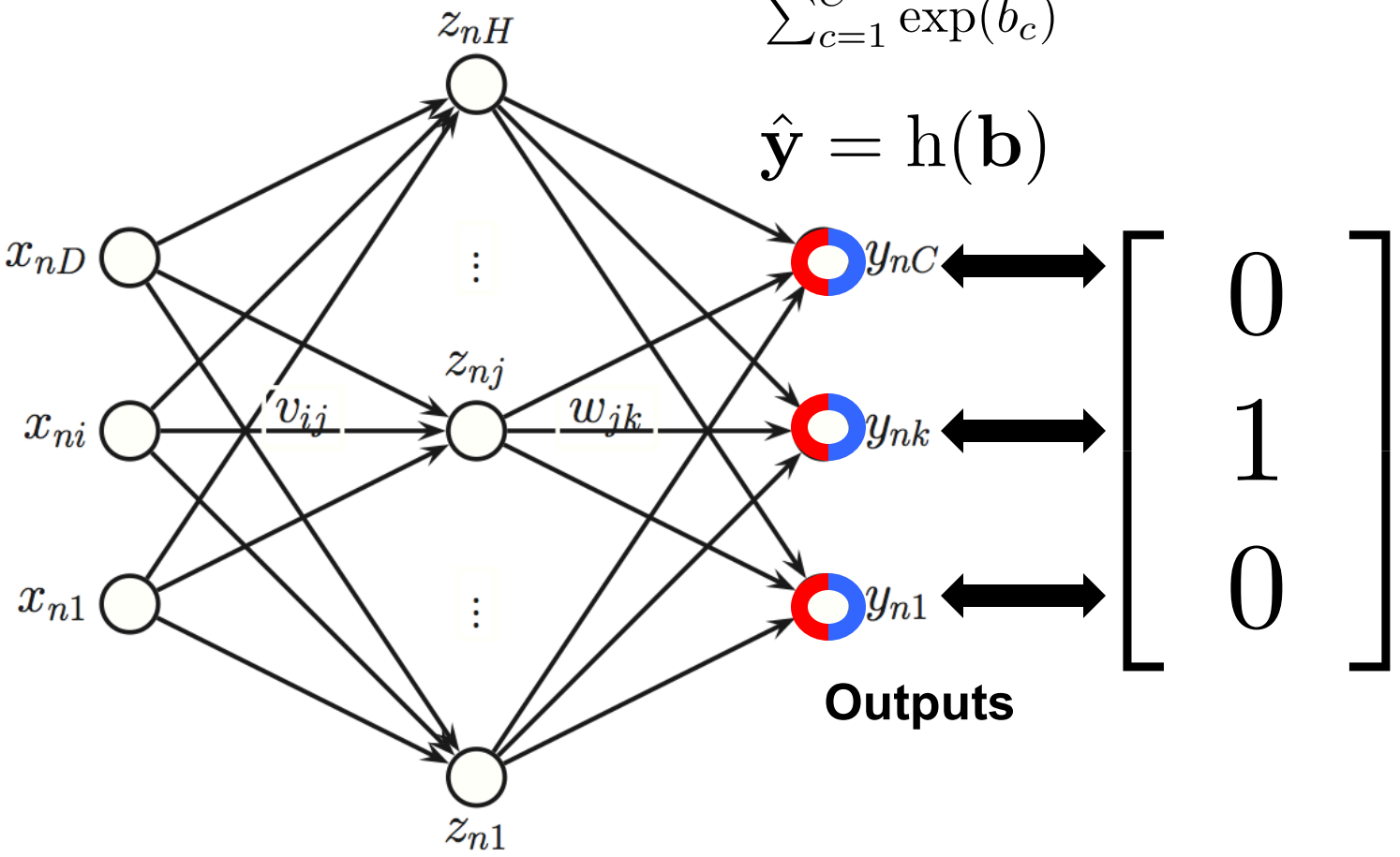
$$= [c=k]\hat{y}_k - \hat{y}_c\hat{y}_k \quad = ([c=k] - \hat{y}_c)\hat{y}_k$$

$$\frac{\partial L}{\partial b_k} = \sum_{c=1}^{C} -\frac{y_c}{\hat{y}_c} ([c=k]\hat{y}_k - \hat{y}_c\hat{y}_k) = -y_k - \sum_{c=1}^{C} (-y_c)\hat{y}_k \; = \hat{y}_k - y_k$$

A neural network in backward mode:

$$\hat{y}_k = \frac{\exp(b_k)}{\sum_{c=1}^{C} \exp(b_c)}$$

$$\hat{\mathbf{y}} = \mathrm{h}(\mathbf{b})$$

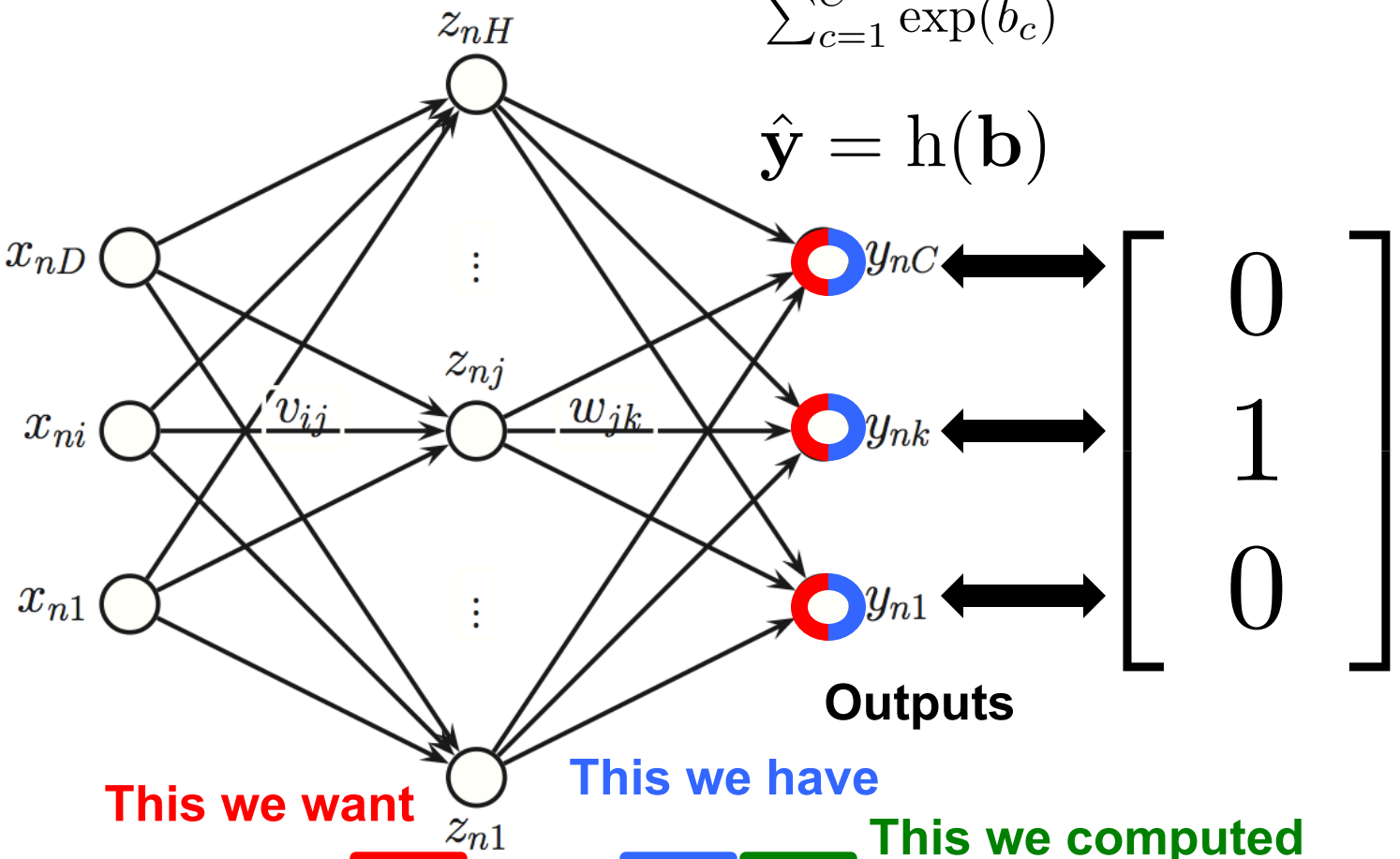


$$\frac{\partial L}{\partial \hat{y}_c} = -\frac{y_c}{\hat{y}_c} \qquad \frac{\partial L}{\partial b_k} = \sum_c \frac{\partial L}{\partial \hat{y}_c} \frac{\partial \hat{y}_c}{\partial b_k}$$

A neural network in backward mode:

$$\hat{y}_k = \frac{\exp(b_k)}{\sum_{c=1}^{C} \exp(b_c)}$$

$$\hat{\mathbf{y}} = \mathrm{h}(\mathbf{b})$$



This we want

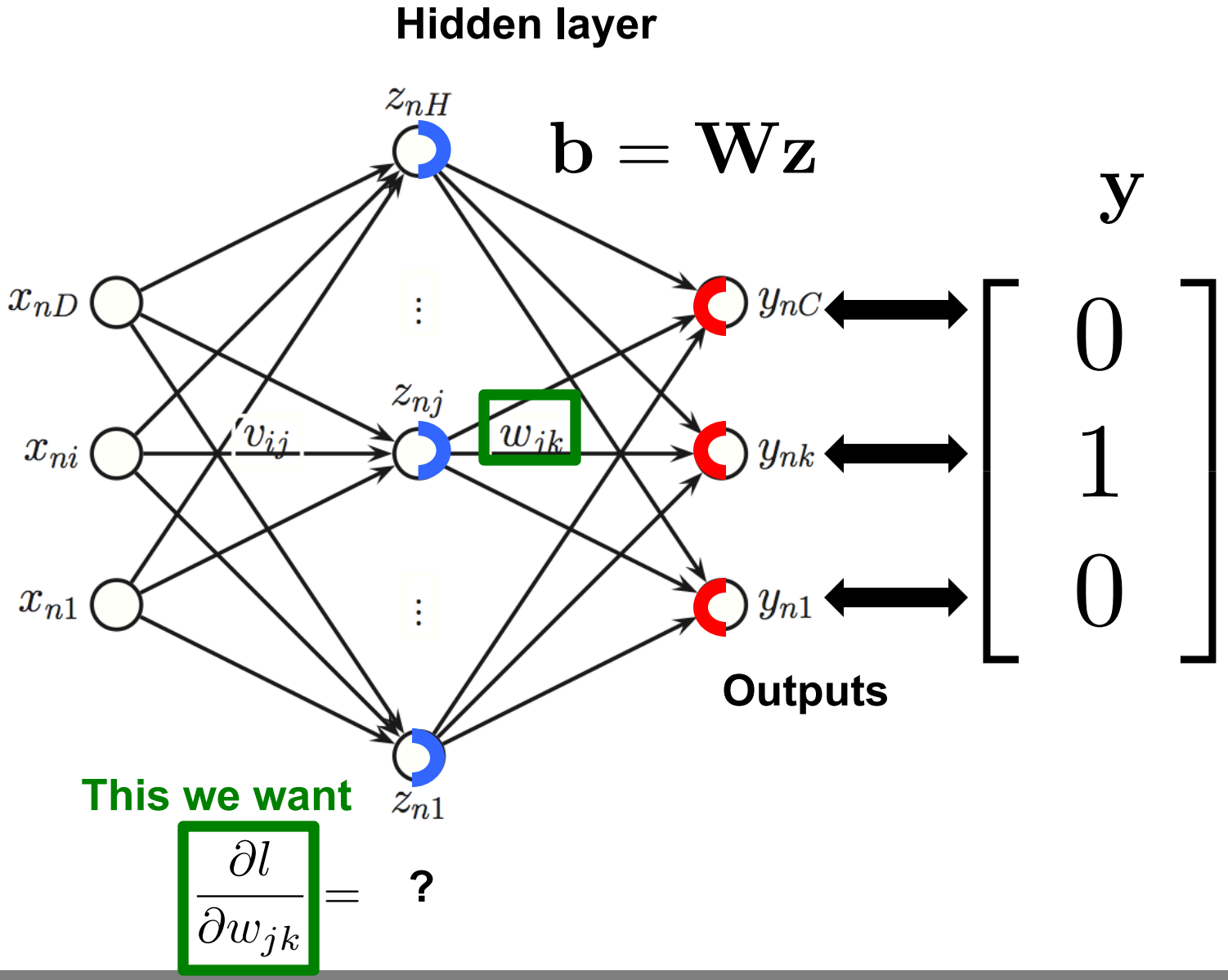
This we have

This we computed

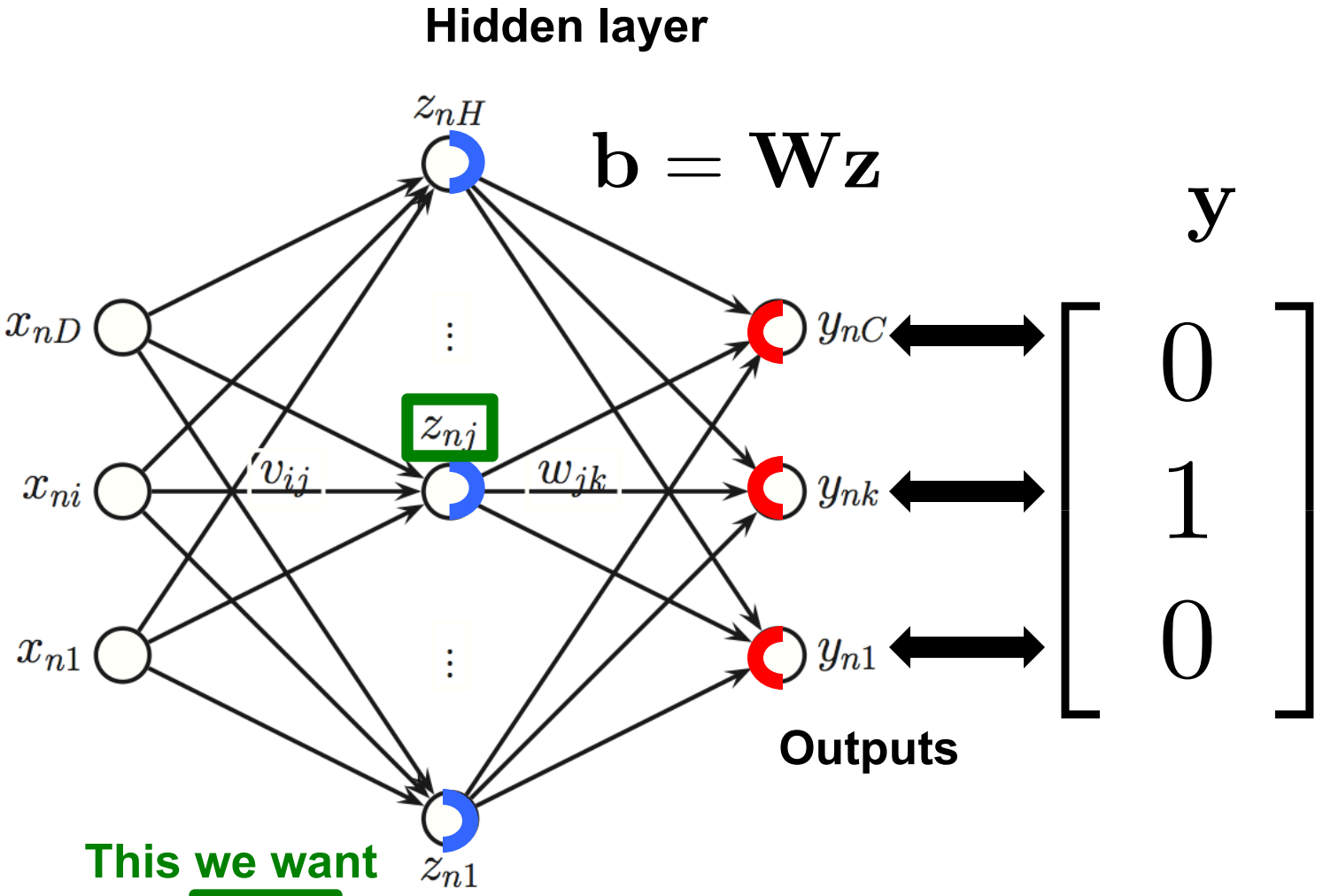
$$\frac{\partial L}{\partial \hat{y}_c} = -\frac{y_c}{\hat{y}_c}$$

$$\frac{\partial L}{\partial b_k} = \sum_c \frac{\partial L}{\partial \hat{y}_c} \frac{\partial \hat{y}_c}{\partial b_k} = \hat{y}_k - y_k$$

A neural network in backward mode:

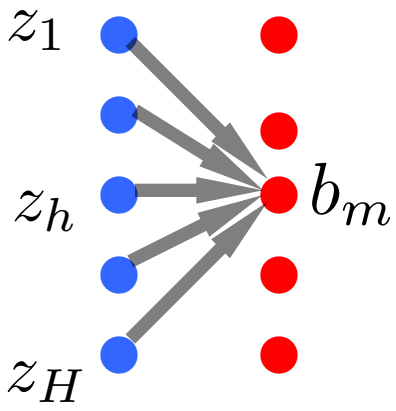


A neural network in backward mode:



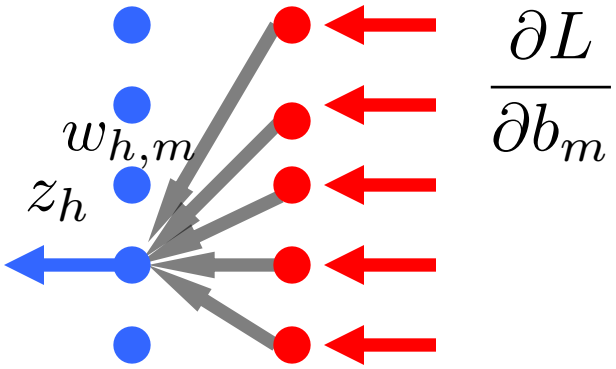
Linear layer in forward mode: all for one

$$b_m = \sum_{h=1}^{H} z_h w_{h,m}$$



Linear layer in backward mode: one from all

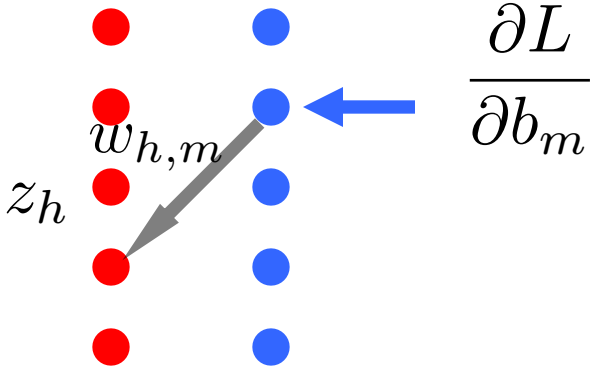
$$b_m = \sum_{h=1}^{H} z_h w_{h,m}$$



$$\frac{\partial L}{\partial z_h} = \sum_{c=1}^{C} \frac{\partial L}{\partial b_c} \cdot \frac{\partial b_c}{\partial z_h} = \sum_{c=1}^{C} \frac{\partial L}{\partial b_c} w_{h,c}$$

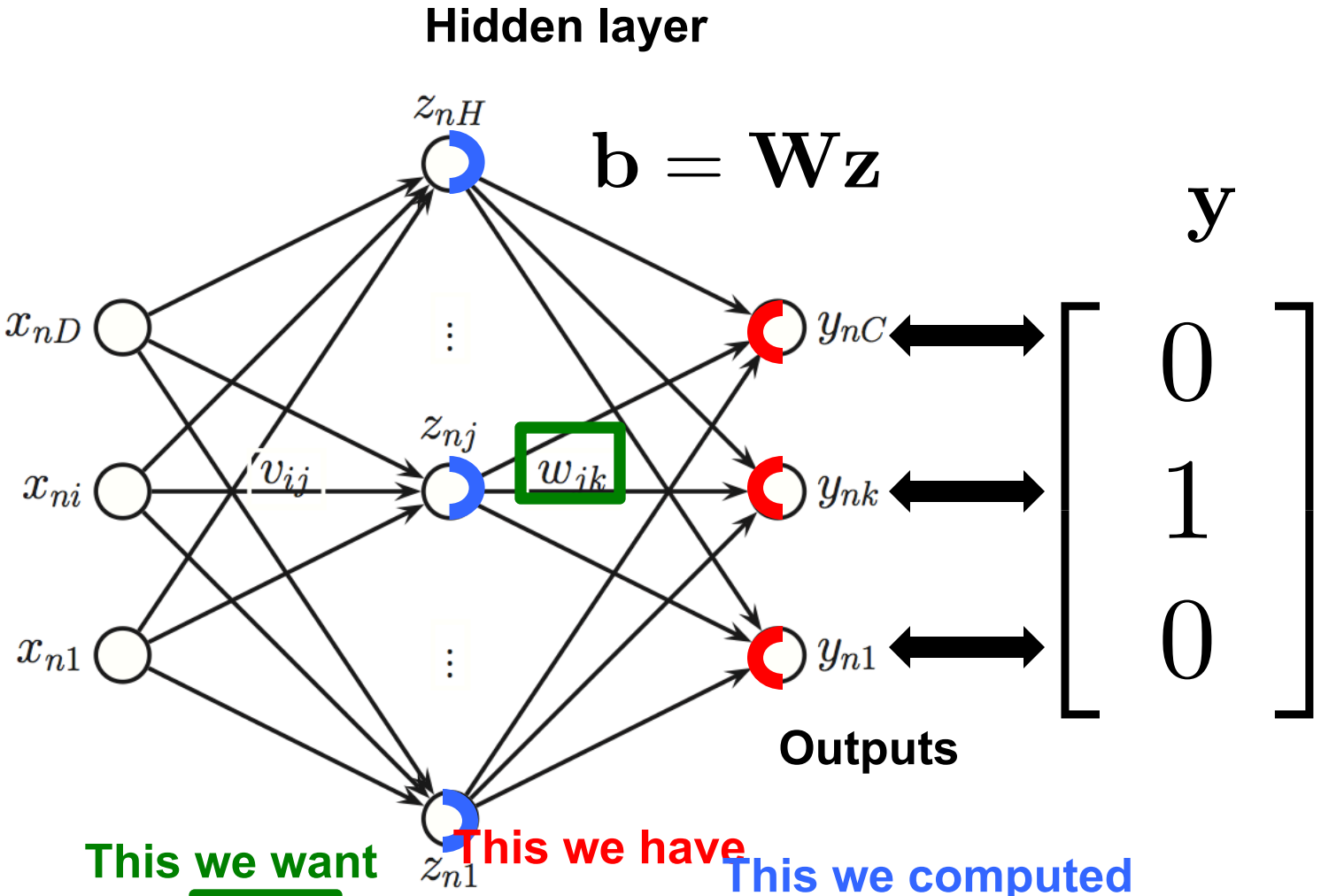
Linear layer parameters in backward: 1-to-1

$$b_m = \sum_{h=1}^{H} z_h w_{h,m}$$

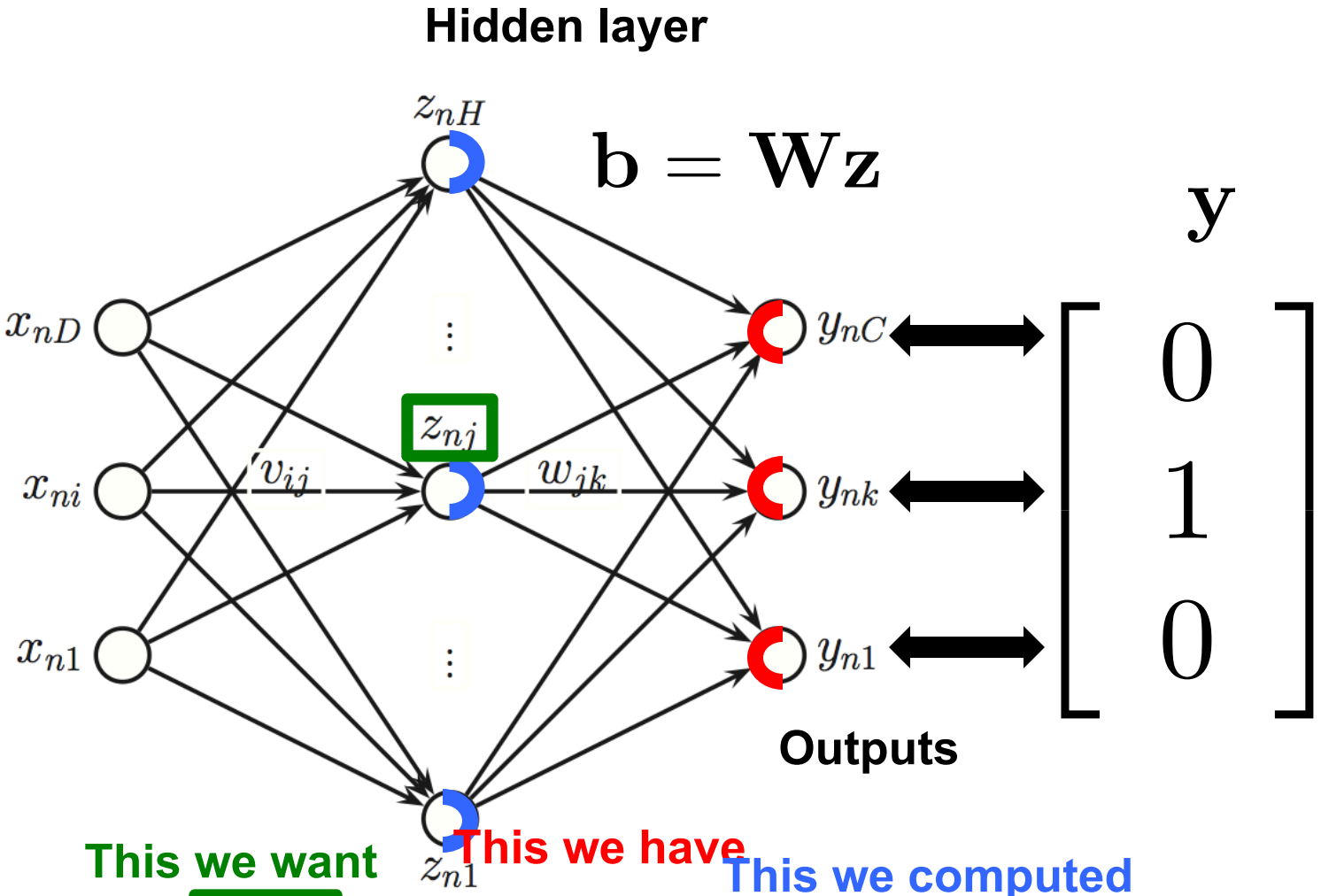


$$\frac{\partial L}{\partial w_{h,m}} = \sum_{c=1}^{C} \frac{\partial L}{\partial b_c} \cdot \frac{\partial b_c}{\partial w_{h,m}} = \frac{\partial L}{\partial b_m} z_h$$

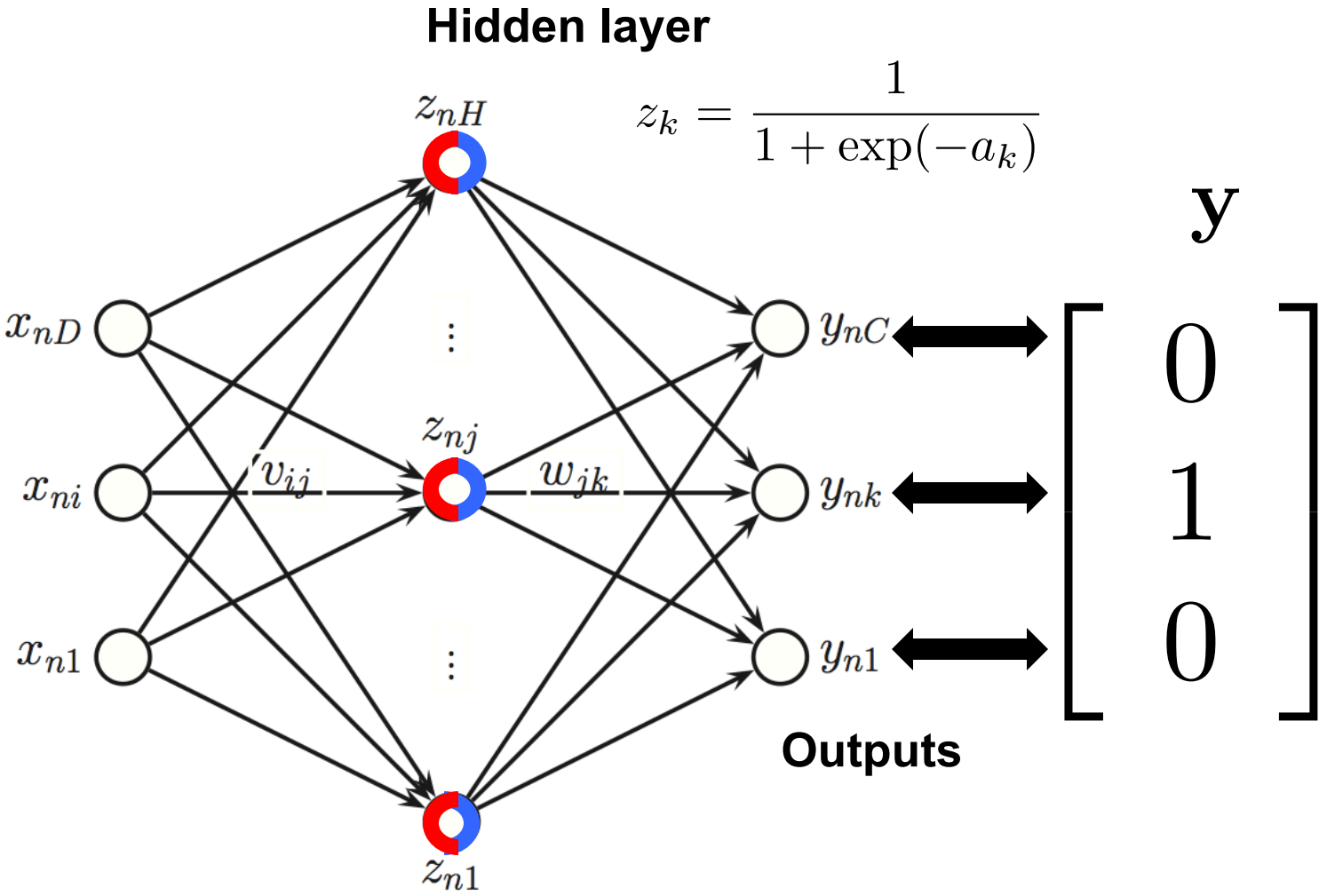
A neural network in backward mode: ⏪



A neural network in backward mode: ⏪

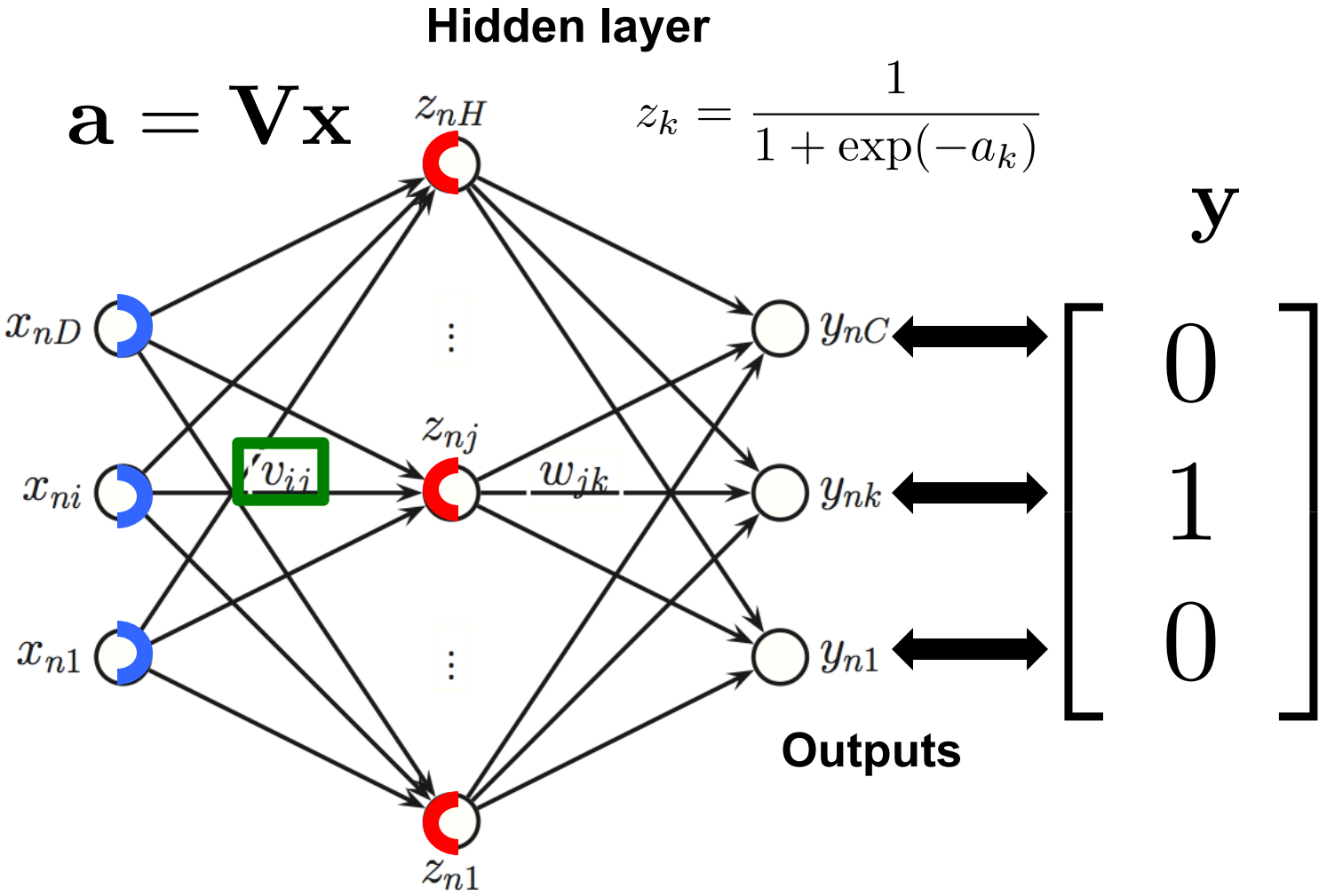


A neural network in backward mode: ⏪



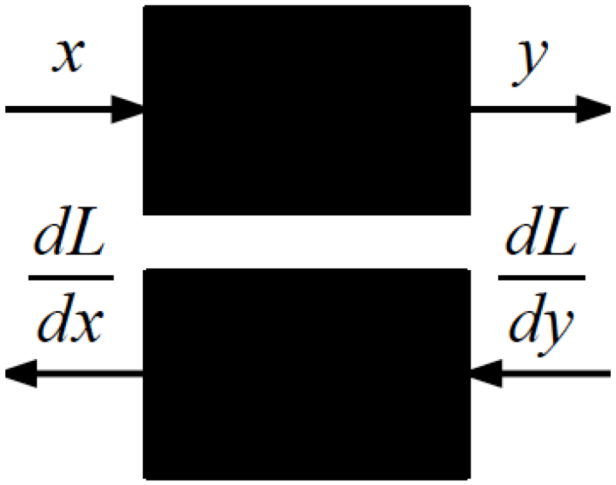
$$\frac{\partial l}{\partial a_k} = \sum_m \frac{\partial l}{\partial z_m}\frac{\partial z_m}{\partial a_k} = \frac{\partial l}{\partial z_k} g'(a_k) = \frac{\partial l}{\partial z_k} g(a_k)(1-g(a_k))$$

A neural network in backward mode:



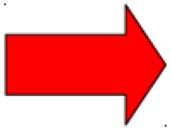
$$\frac{\partial l}{\partial v_{ij}} = \sum_k \frac{\partial l}{\partial a_k}\frac{\partial a_k}{\partial v_{ij}} = \frac{\partial l}{\partial a_j} x_i$$

Chain Rule

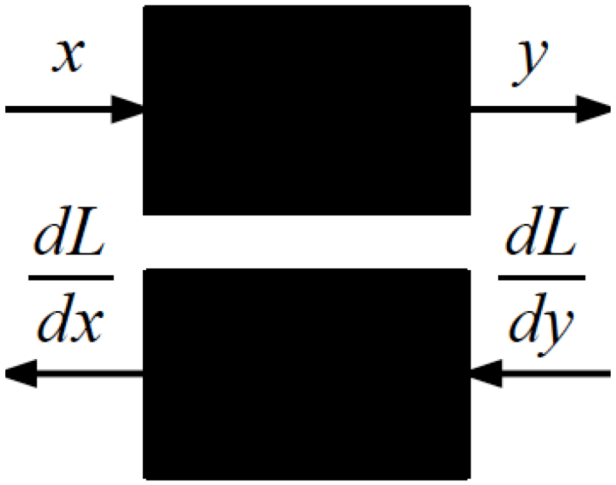


Given $y(x)$ and dL/dy,

What is dL/dx ?



Chain Rule

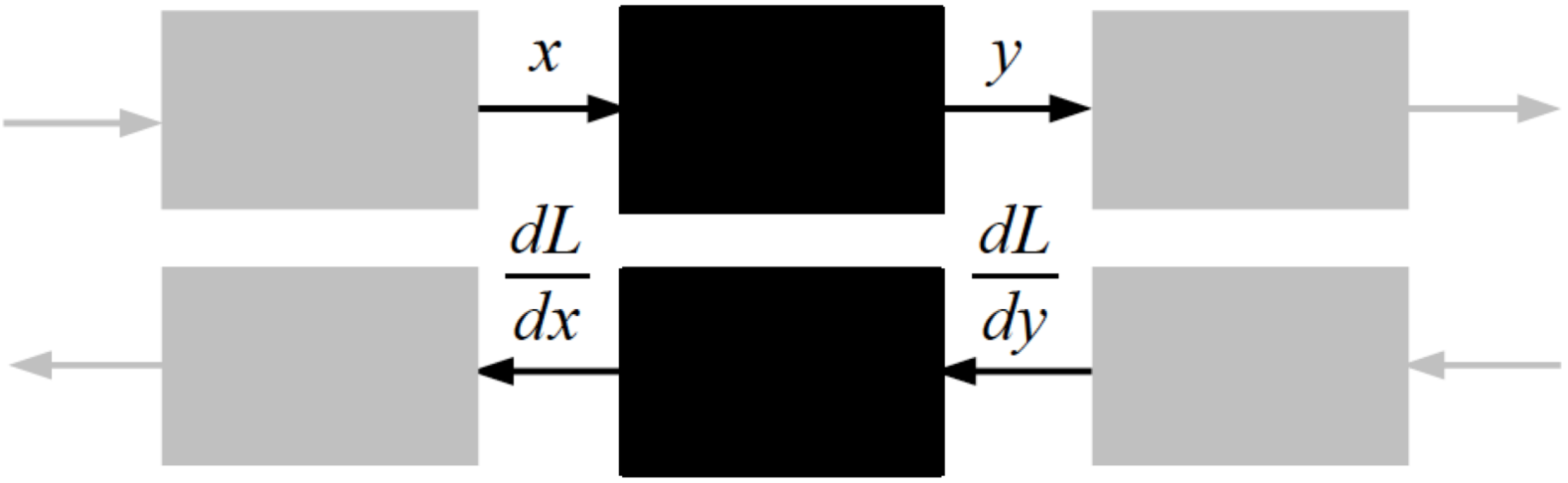


Given $y(x)$ and dL/dy,

What is dL/dx ?

$$\frac{dL}{dx} = \frac{dL}{dy} \cdot \frac{dy}{dx}$$

‘another brick in the wall’



Given $y(x)$ and dL/dy,

What is dL/dx ?

$$\frac{dL}{dx} = \frac{dL}{dy} \cdot \frac{dy}{dx}$$