CNN & Deep Learning

Pierre Alliez

Inria Sophia Antipolis
pierre.alliez@inria.fr

Slide credit: Niloy Mitra

Representations in Computer Graphics

- Images (e.g., pixel grid)
- Volume (e.g., voxel grid)
- Meshes (e.g., vertices/edges/faces)
- Point clouds (e.g., collection of points)
- Animation (e.g., skeletal positions over time; cloth dynamics over time)
- Physics simulations (e.g., fluid flow over space-time, multi body interaction)

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, these are learned

$\mathbb{X}$: source domain

$\mathbb{Y}$: target domain

Examples:

Image Classification:

$$f_\theta : \mathbb{R}^{w \times h \times c} \longrightarrow \{0, 1, \ldots, k-1\}$$

$w \times h \times c$: image dimensions

k: class count

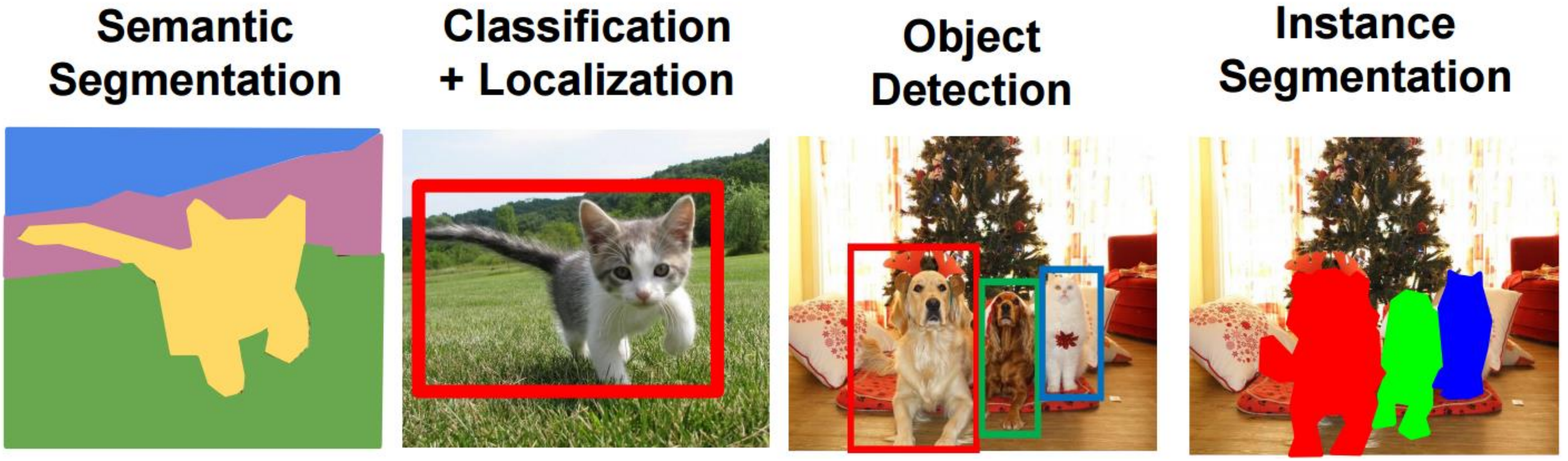
Image Synthesis:

$$f_\theta : \mathbb{R}^{n} \longrightarrow \mathbb{R}^{w \times h \times c}$$

n : latent variable count

$w \times h \times c$: image dimensions

Semantic Segmentation



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf

The Legend of Tarzan

Pose Detection using CNNs

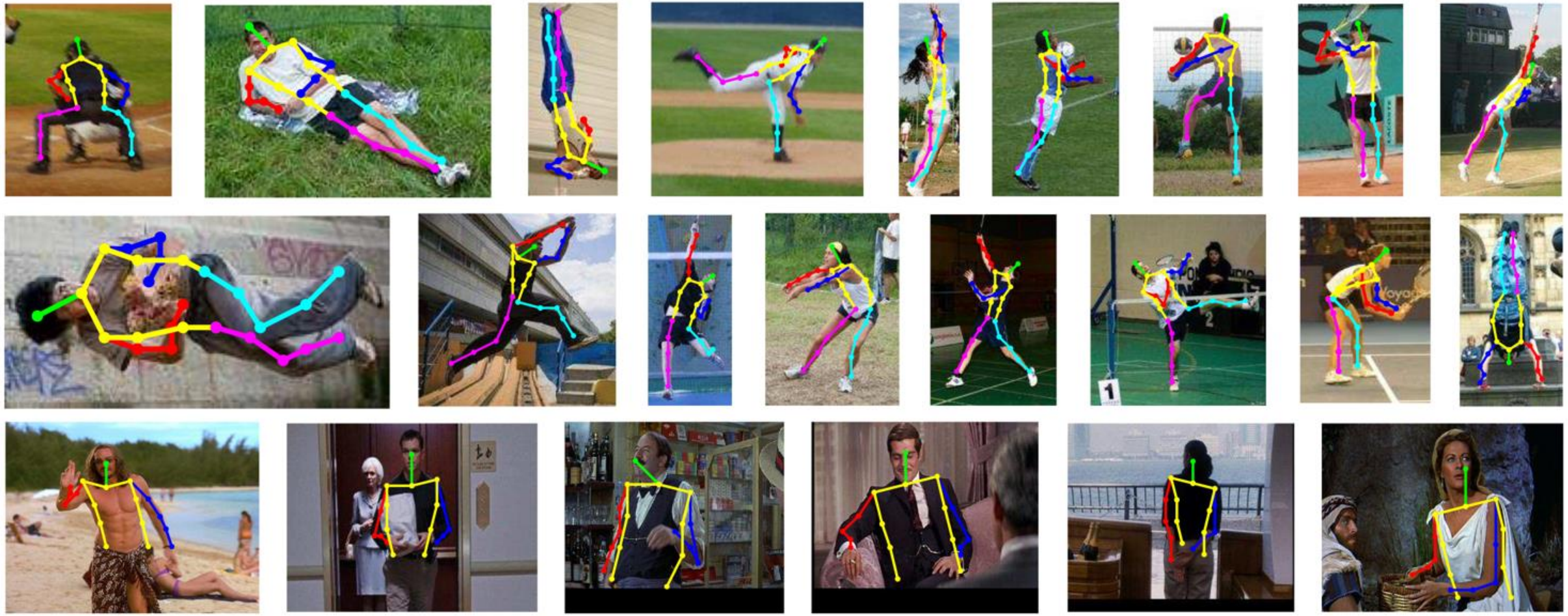
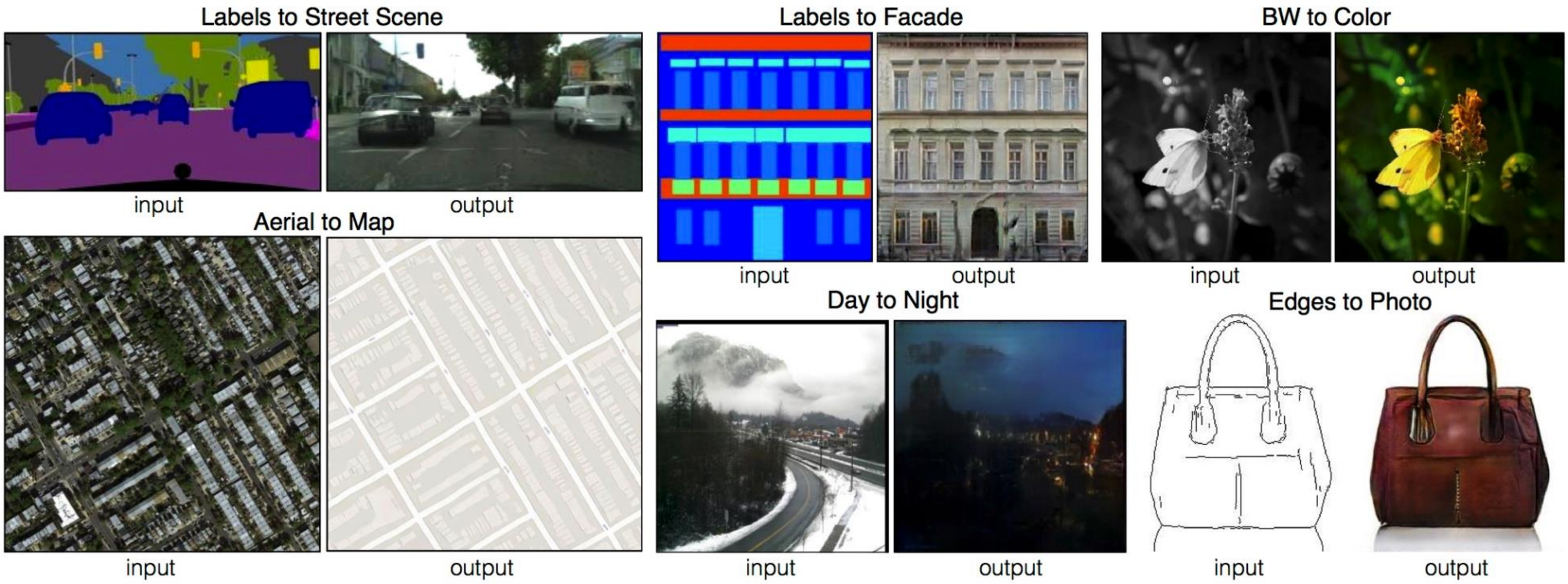


Image Denoising

[Chaitanya et al. 2017, Siggraph]

Image Translation Problems

[Isola et al. 2017, CVPR]



Sketch to Face!

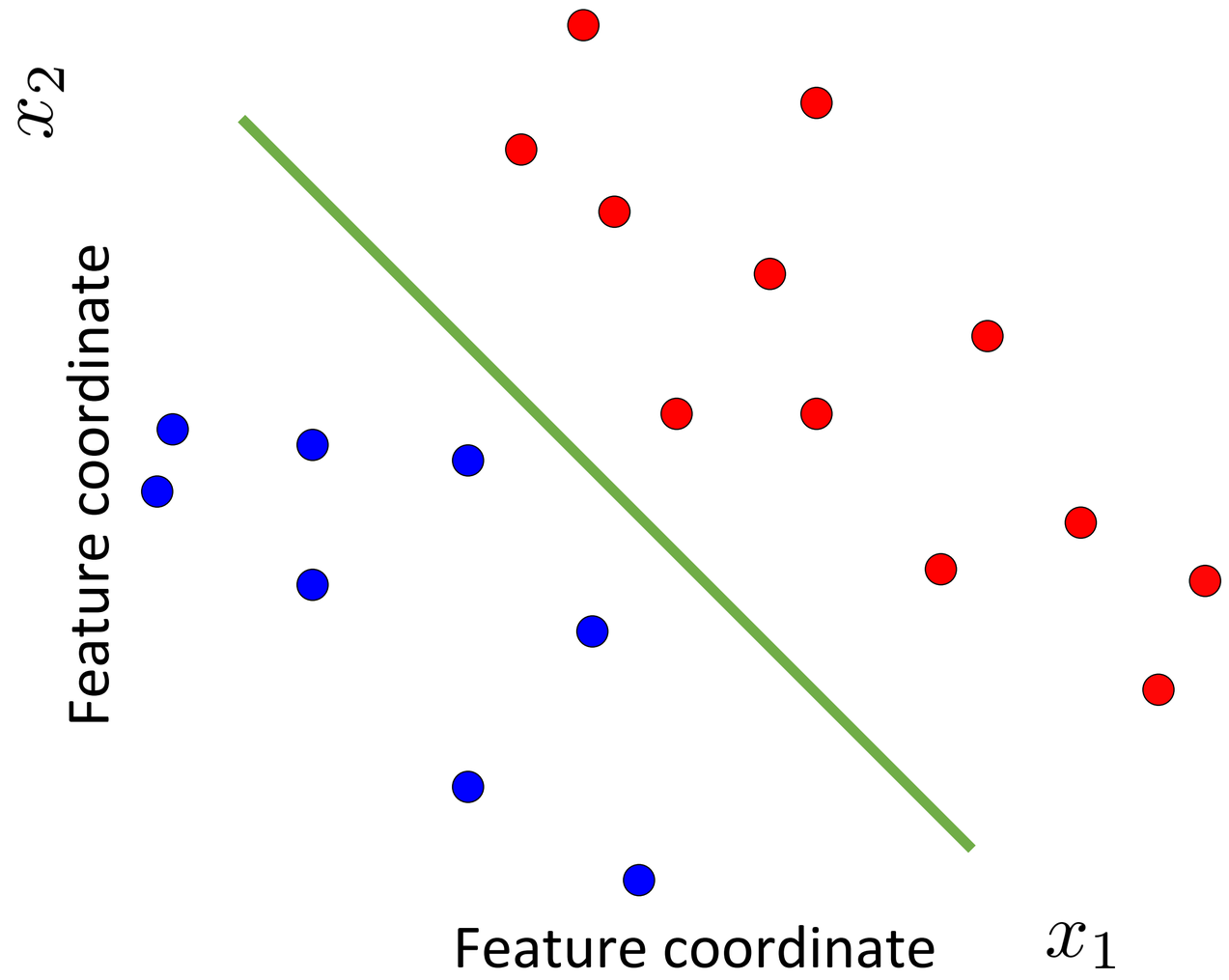
[Han et al. 2017, Siggraph]

DeepSketch2Face: A Deep Learning Based Sketching System for 3D Face and Caricature Modeling

[Wang et al. 2018, Siggraph Asia]

Real Images

Machine Learning 101: Linear Classifier



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

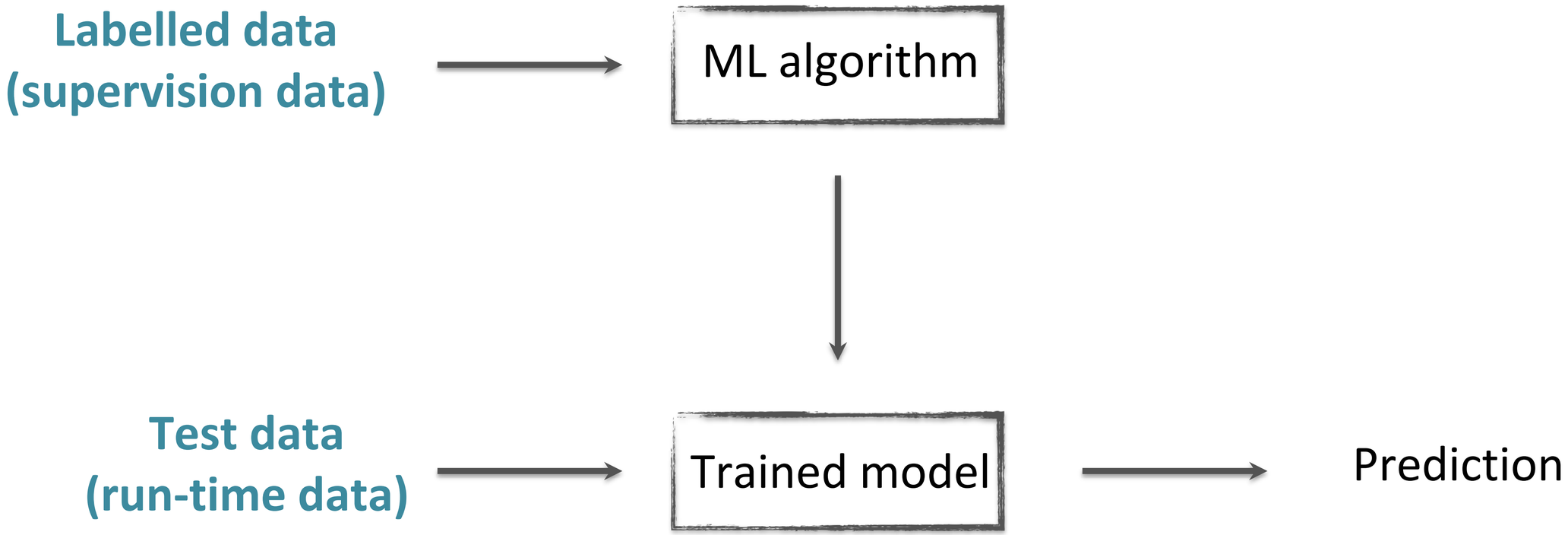
$$f_\theta(x) = \begin{cases} 1 & \text{if } wx + b \geq 0 \\ 0 & \text{if } wx + b < 0 \end{cases}$$

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

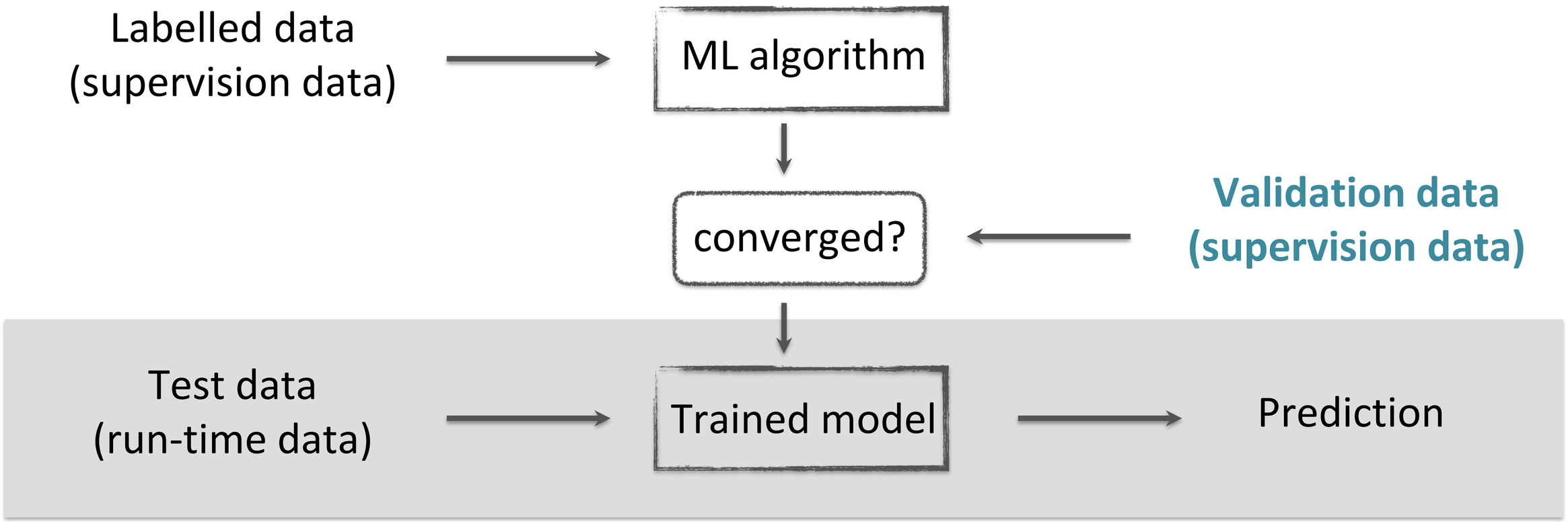
Each data point has a class label:

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

Data-driven Algorithms (Supervised)

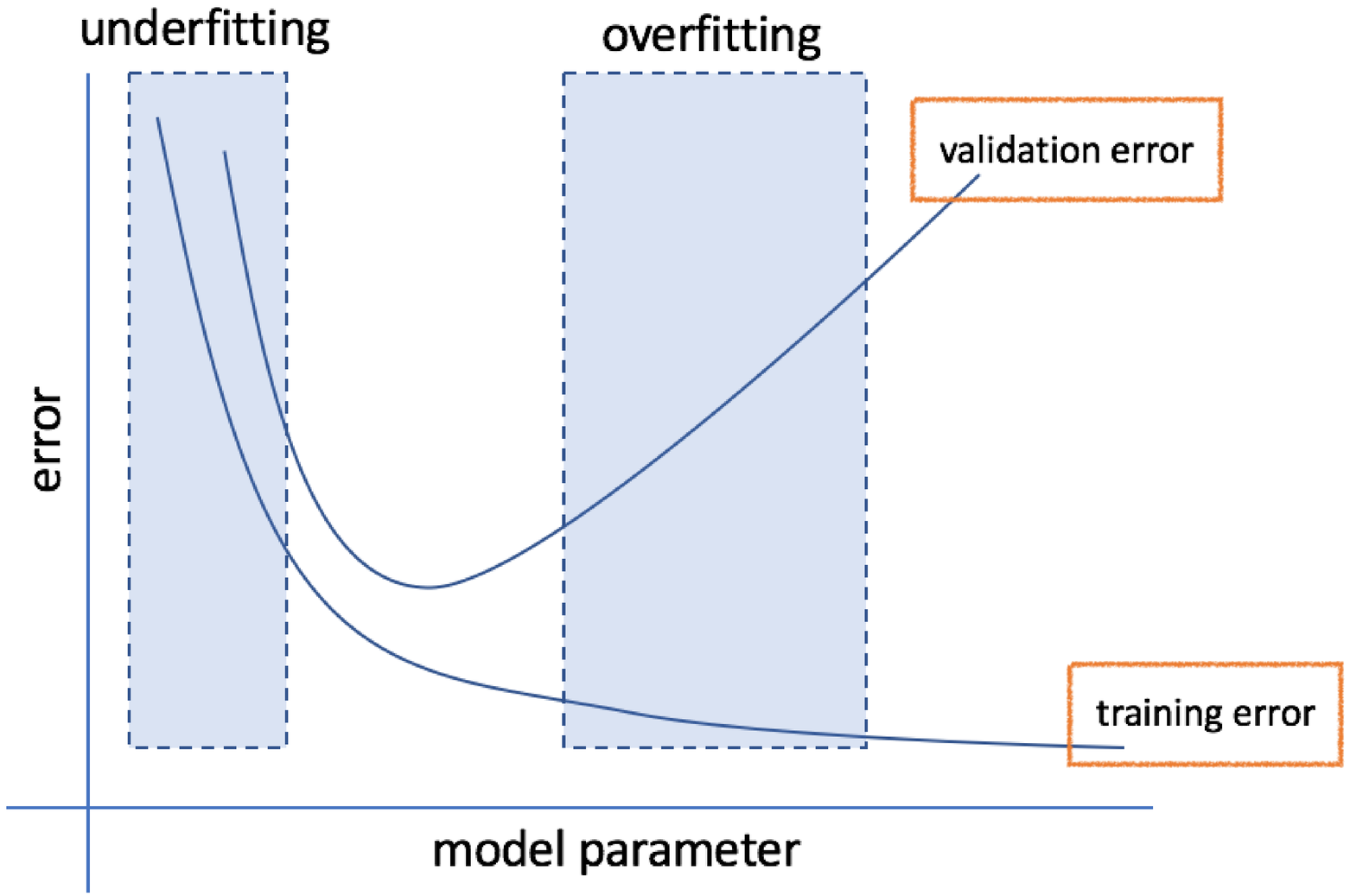


Data-driven Algorithms (Supervised)

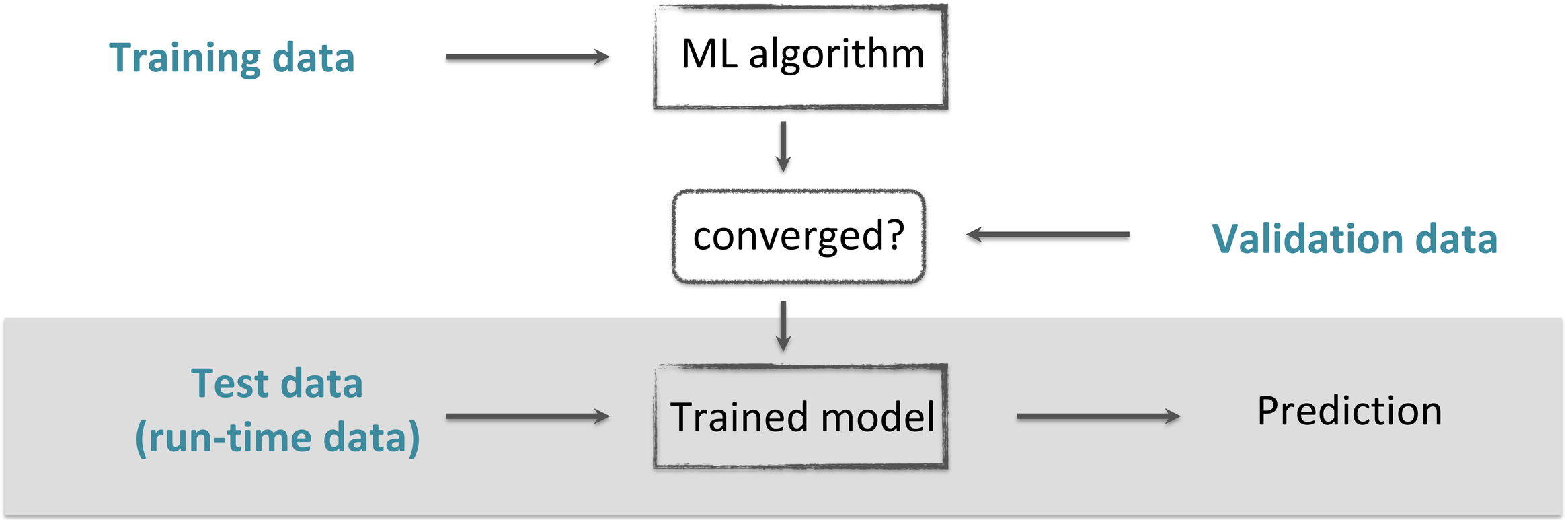


Implementation Practice: Training: 70%; Validation: 15%; Test 15%

Training versus Validation Loss/Accuracy

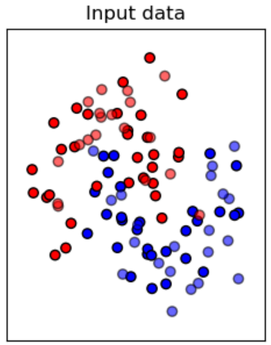


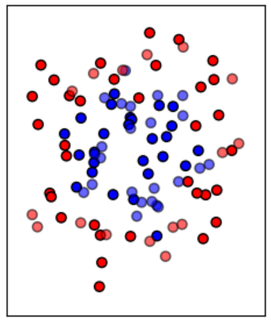
Data-driven Algorithms (Unsupervised)

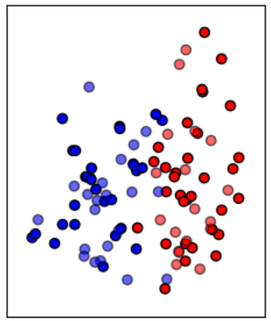


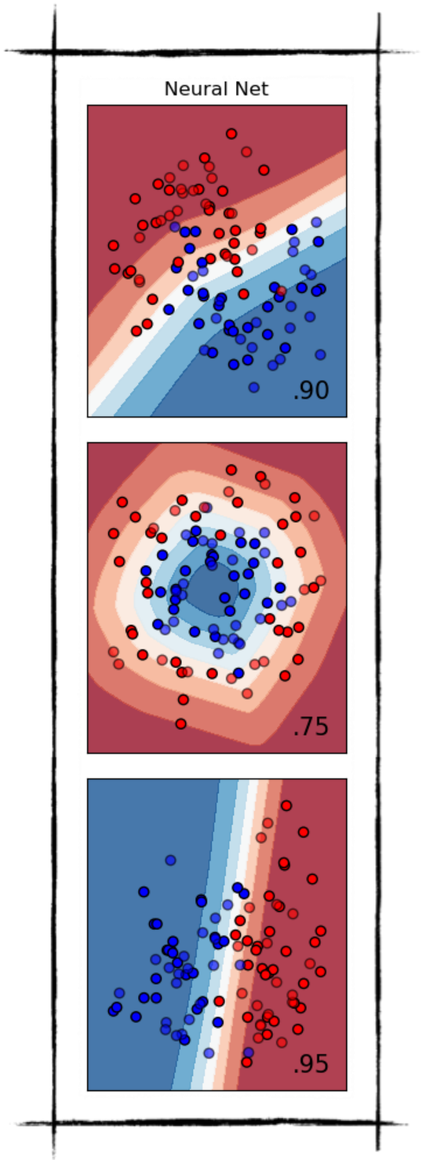
Implementation Practice: Training: 70%; Validation: 15%; Test 15%

Various ML Approaches (Supervised approaches)









http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

Rise of Learning

- 1958: Perceptron
- 1974: Backpropagation
- 1981: Hubel & Wiesel wins Nobel prize for ‘visual system’
- 1990s: SVM era
- 1998: CNN used for handwriting analysis
- **2012: AlexNet wins ImageNet**

What is Special about CG?

1. **Regular data structure** and easy to parallelize
   (e.g., image translation)

2. Many sources of input data — **model building**
   (e.g., images, scanners, motion capture)

3. Many sources of **synthetic data** — can serve as supervision data
   (e.g., rendering, animation)

4. Many problems in **generative models** and need for **user-control**

Main Challenges and Scope for Innovation

1. **Representation:** How is the data organised and structured?

2. **Training data:** Is it synthetic or real, or mixed?

3. **User control:** End-to-end or in small steps?

4. **Loss functions:** Hand-crafted or learned from data?

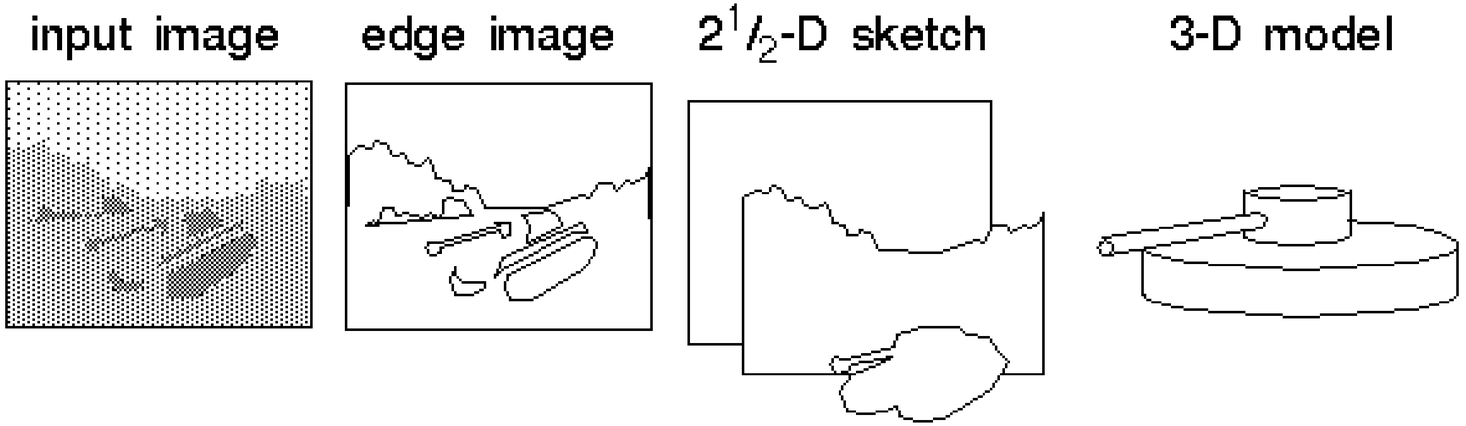
Data is the New Currency

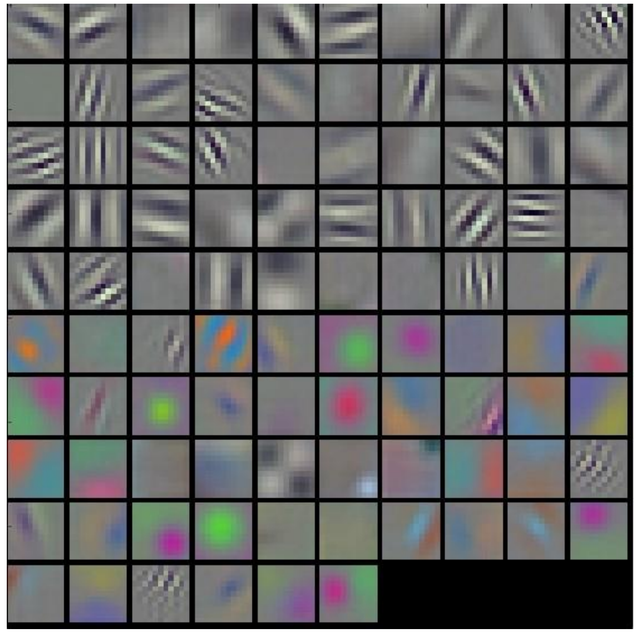
- **Synthetic** data
  - **Generative model** + photo-realistic **rendering**
  - Object geometry + physical **simulation**
  - Object geometry + synthetic materials + realistic simulations

- **Real** data
  - Collected from images, scans, mocap sessions
  - Collected using specialized equipments (e.g., light-field, pressure gloves)

End-to-end: Learned Features

- *Before*
  - Handcrafted feature extraction, e.g., edges or corners (hand-crafted)
  - Mostly with linear models (PCA)
- *Now*
  - End-to-end
  - Move away from hand-crafted representations





End-to-end: Learned Loss

- *Before*
  - Evaluation came after
  - It was a bit optional
    - You might still have a good algorithm without a good way of quantifying it
    - Evaluation helped publishing
- *Now*
  - It is essential and build-in
  - If the loss is not good, the result is not good
  - (Extensive) Evaluation happens automatically

- While still much is left to do, this makes graphics much more reproducible

End-to-end Training: Real/Generated Data

- *Before*
  - Test with some toy examples
  - Deploy on real stuff
  - Maybe collect some performance data later

- *Now*
  - Test and deploy need to be as identical **(in distribution)**
  - Need to collect data first
  - No two steps

Examples in Graphics

Geometry

Image
manipulation

Animation

Rendering

Examples in Graphics

Geometry

Procedural modelling

Mesh segmentation

Learning deformations

Colorization

Sketch simplification

Image manipulation

Animation

BRDF estimation

Boxification

Fluid

Real-time rendering

Animation

Rendering

Denoising

Facial animation

PCD processing

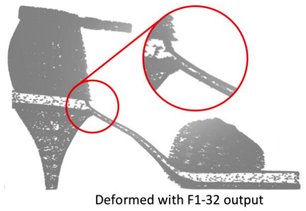
Examples in Graphics

Sketch simplification

Colorization

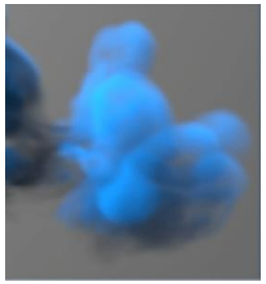
Procedural modelling

Mesh segmentation

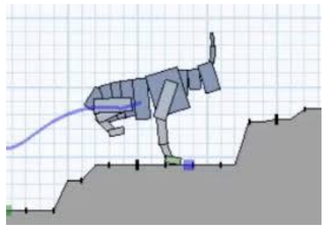


Learning deformations

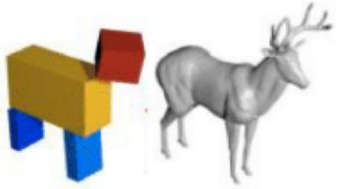
BRDF estimation



Fluid



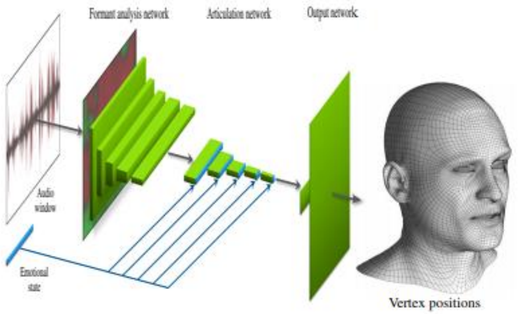
Animation



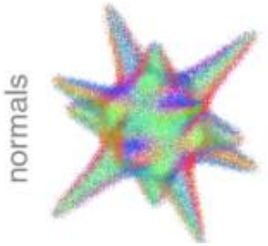
Boxification

Real-time rendering

Denoising



Facial animation



PCD processing

Deep Learning

Problems eg 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

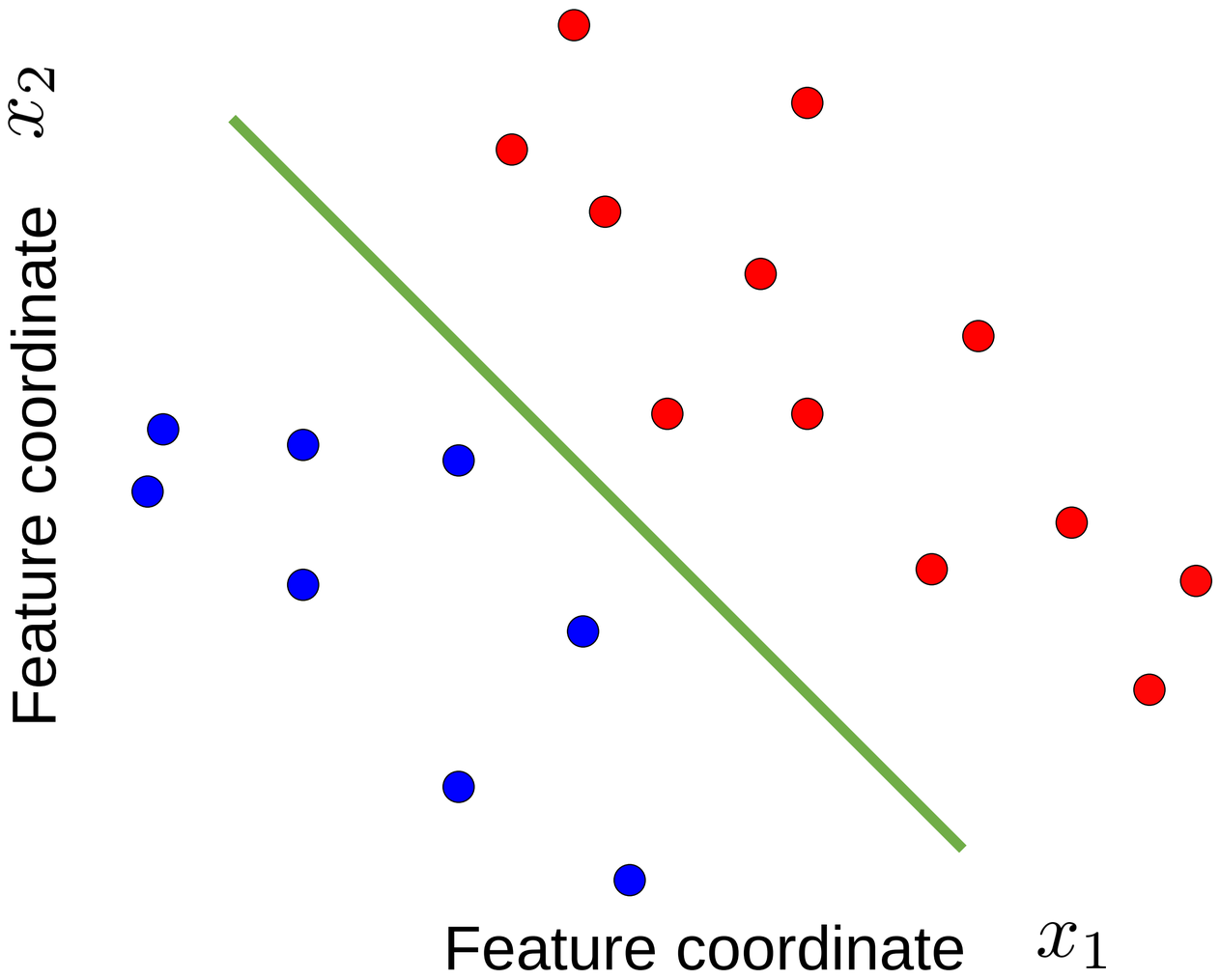
Goal: Learn a Parametric Function

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

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

Neural networks: choice of functional form

Linear classification problem



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

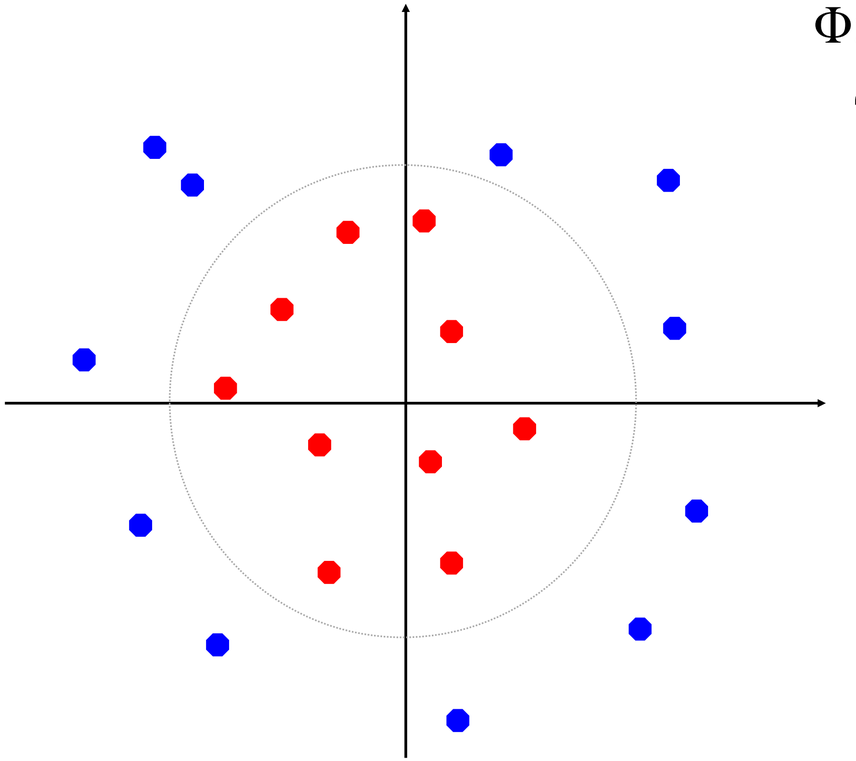
$$f_\theta(x) = \begin{cases} 1 & \text{if } wx + b \geq 0 \\ 0 & \text{if } wx + b < 0 \end{cases}$$

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

Each data point has a class label:

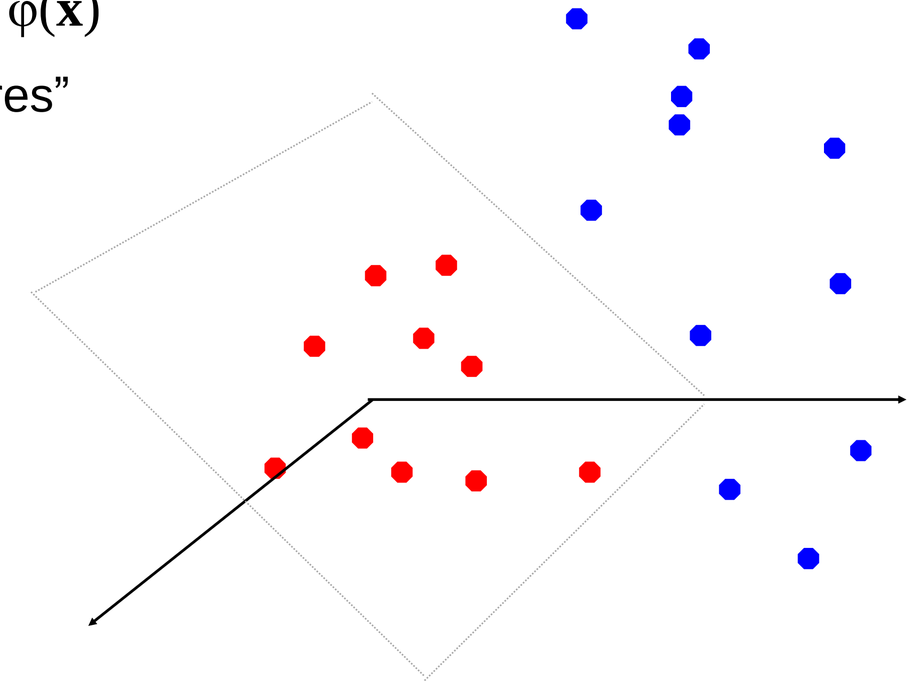
$$y^i = \begin{cases} 1 & (\bullet) \\ 0 & (\bullet) \end{cases}$$

Beyond linear functions



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

"features"

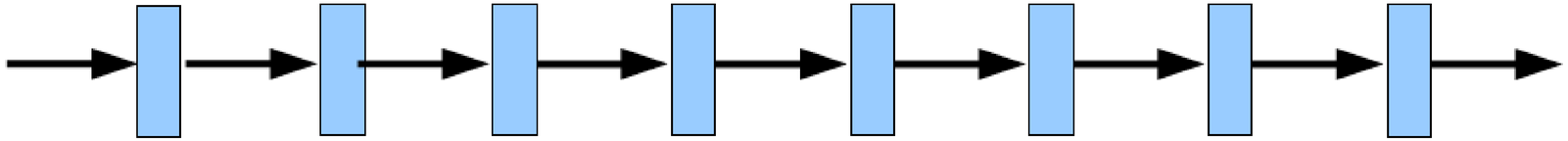


Not linearly-separable in R^2

Linearly-separable in R^3

Neural networks: learn the features!

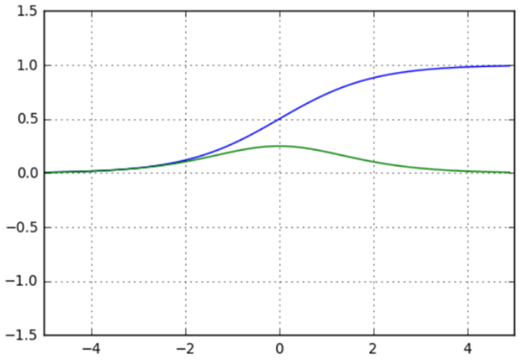
Neural networks: function composition



basic building block

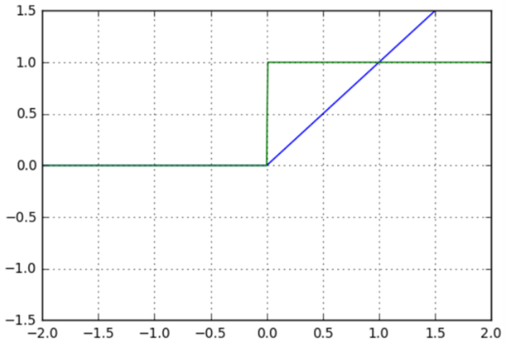
‘Neuron’: Cascade of Linear and Nonlinear Function

function
derivative



Sigmoidal (“logistic”)

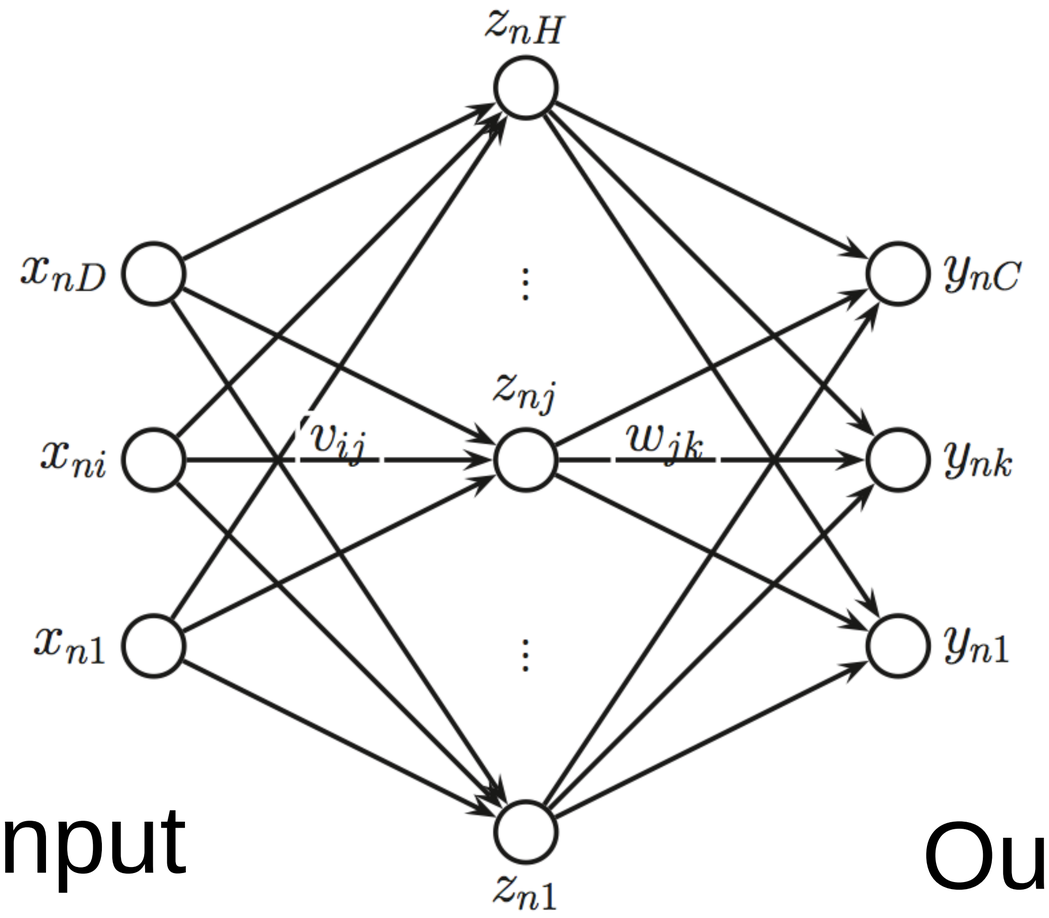
$$g(a) = \frac{1}{1 + \exp(-a)}$$



Rectified Linear Unit (RELU)

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

A two-layer neural network



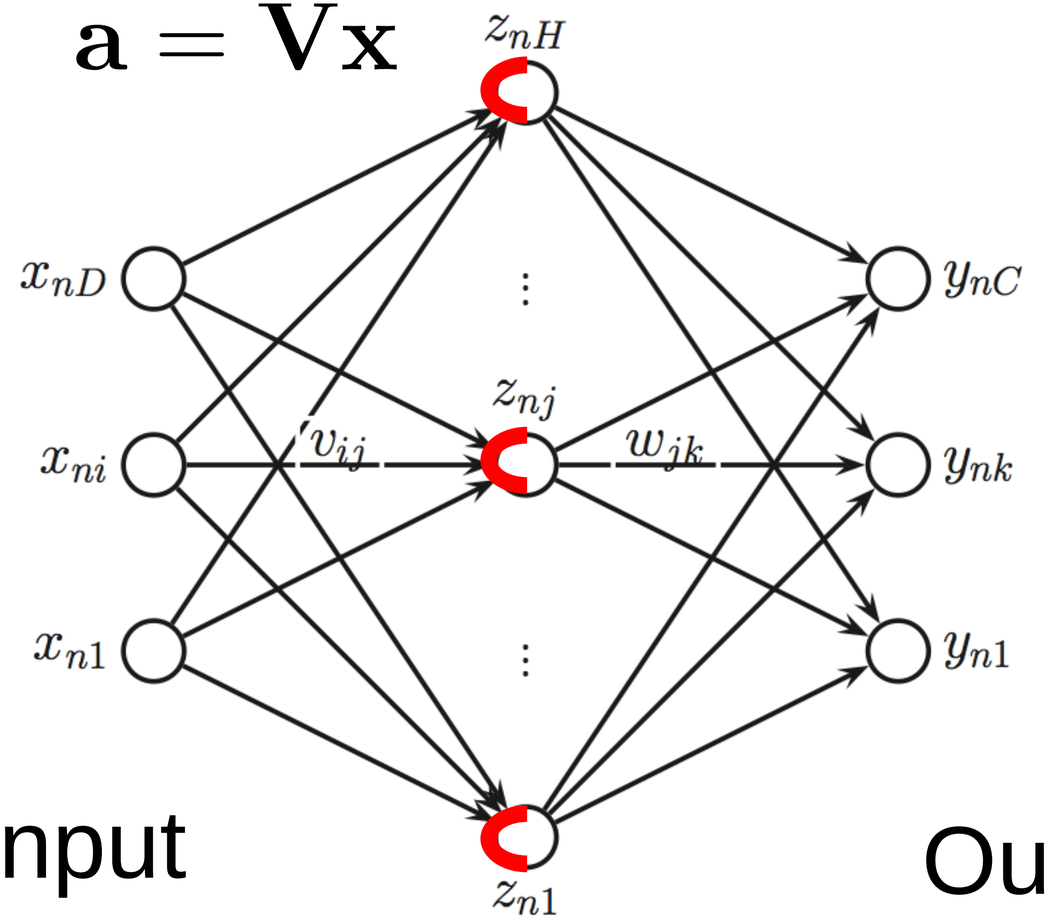
Inputs

Hidden layer

Outputs

A two-layer neural network

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



Inputs

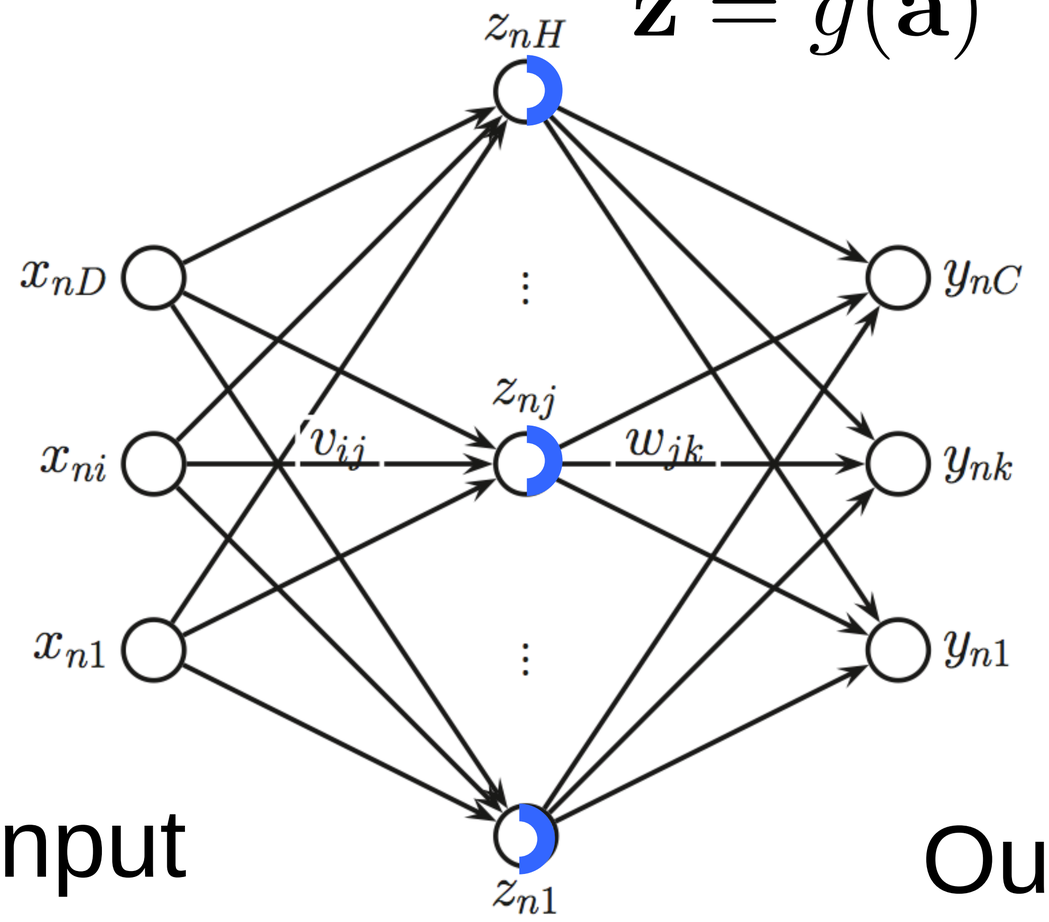
Hidden layer

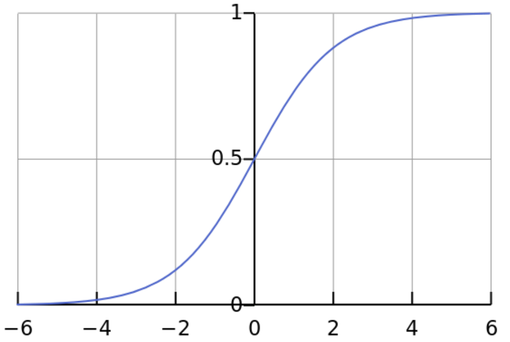
Outputs

A two-layer neural network

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

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





Inputs

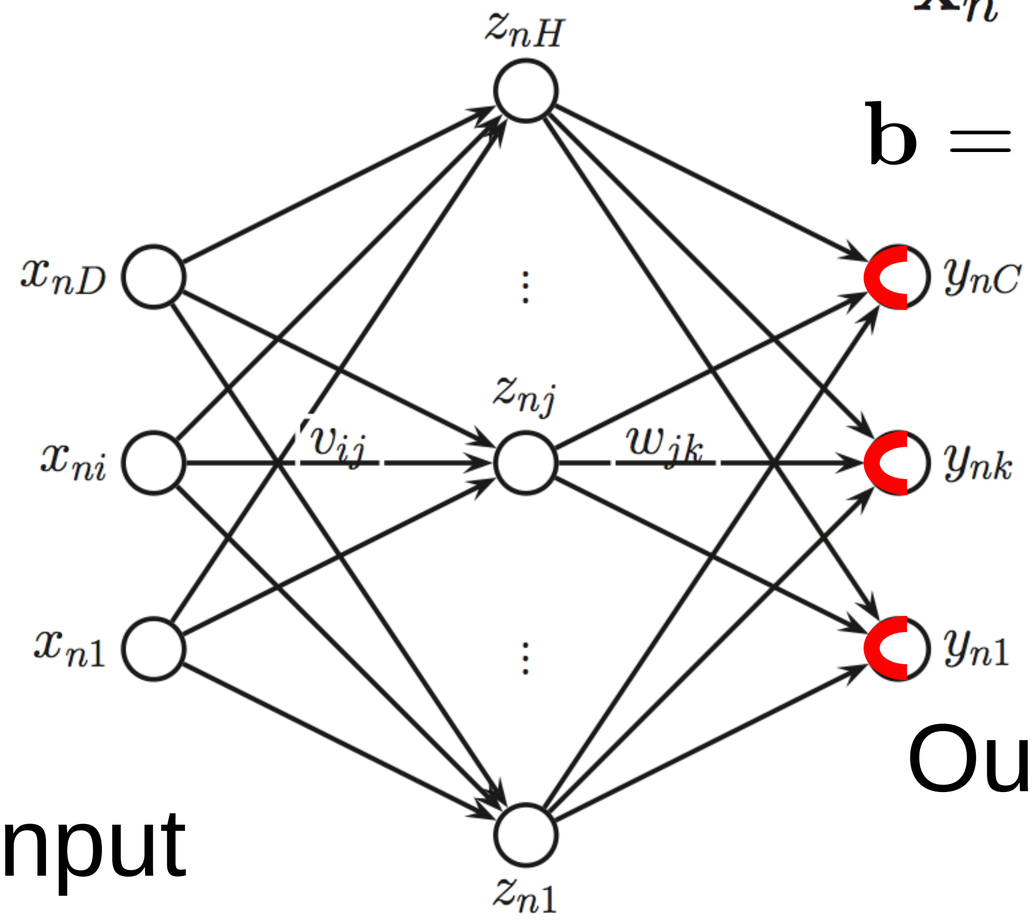
Hidden layer

Outputs

A two-layer neural network

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

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



Outputs

Inputs

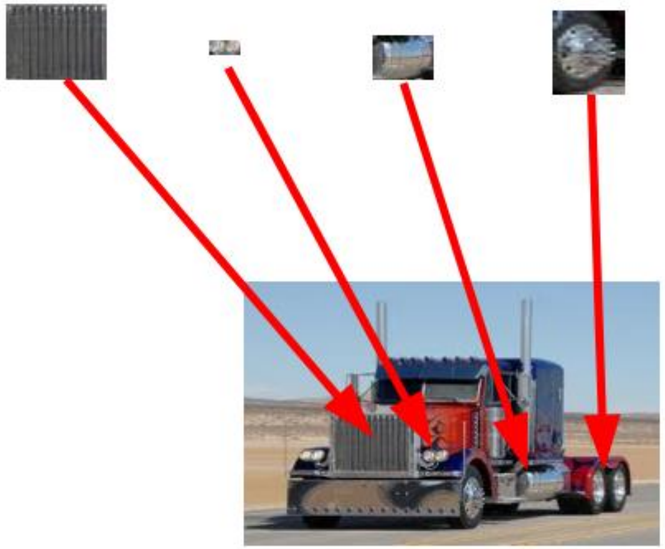
Hidden layer

Hidden Layers: intuitively, what do they do?

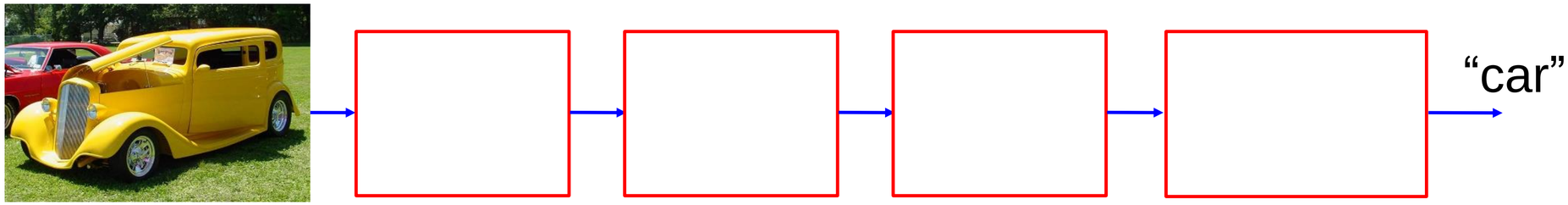
Intuition: learn “dictionary” for objects

“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

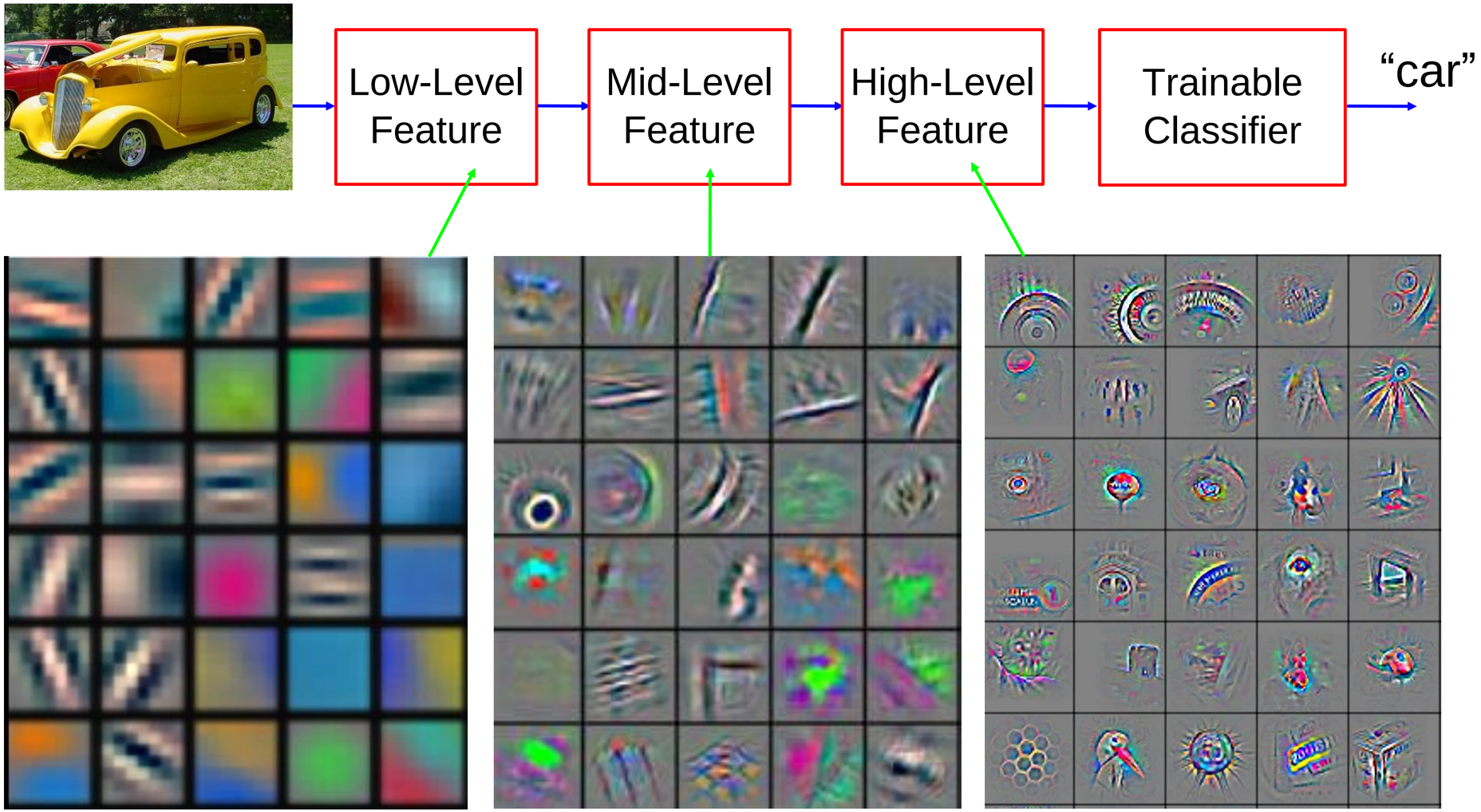


Deep Learning = Hierarchical Compositionality



Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Deep Learning = Hierarchical Compositionality



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:

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

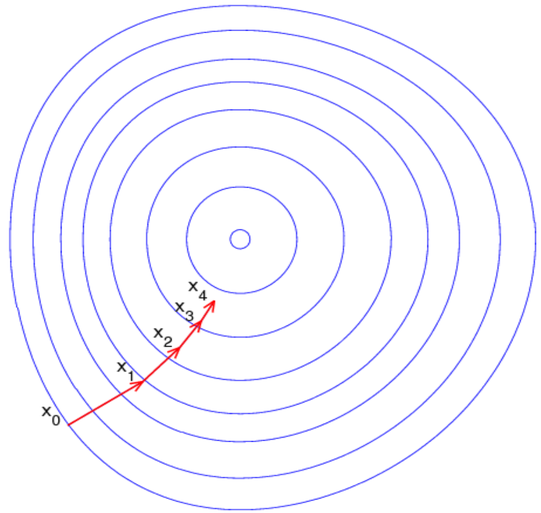
Example: L2 regression loss given target (x^i, y^i) pairs :

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

Gradient Descent Minimization Method

Initialize: θ_0

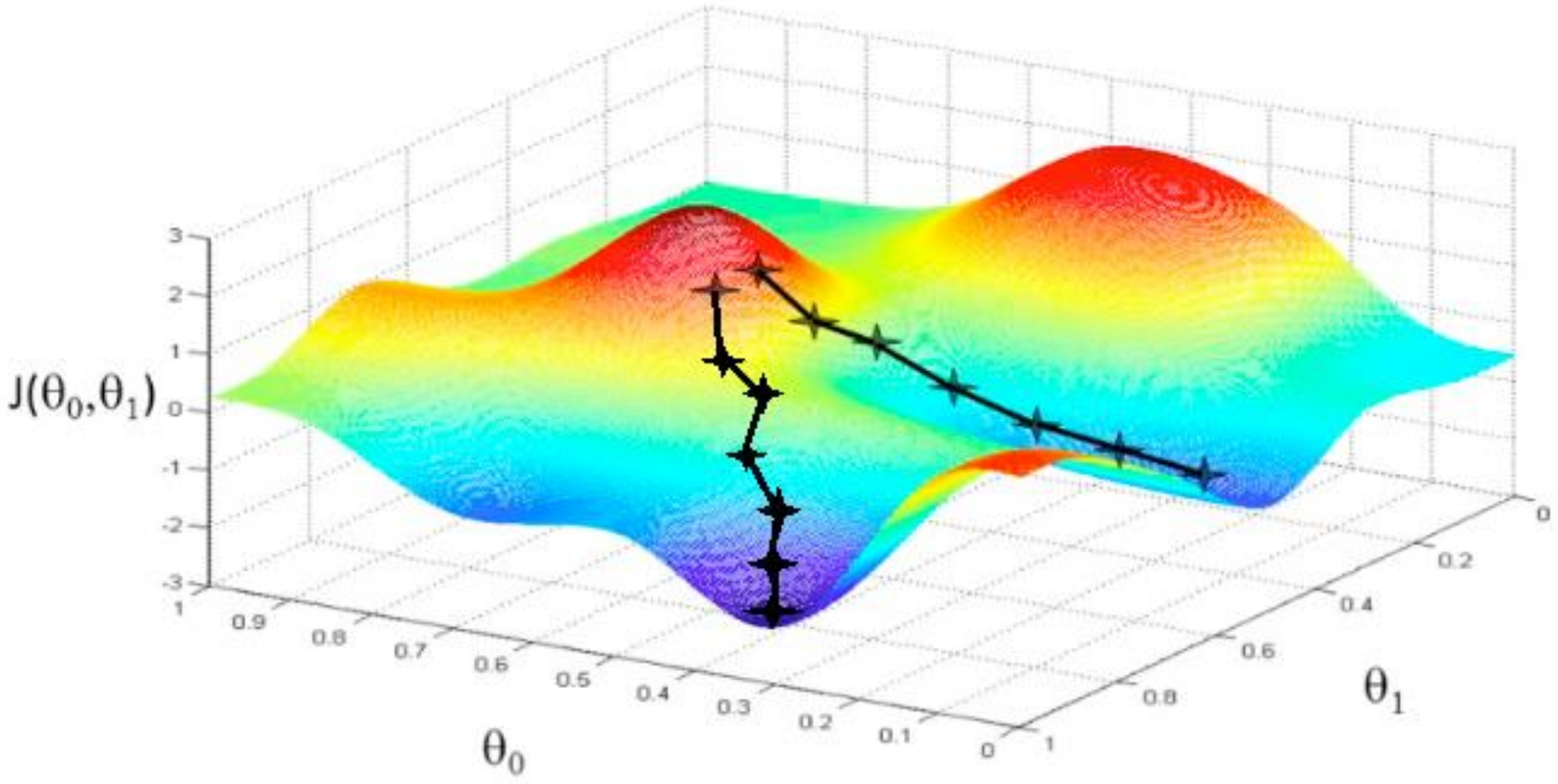
Update: $\theta_{i+1} = \theta_i - \alpha \nabla f(\theta_i)$



We can always make it converge for a convex function

Neural network's loss: non convex objective

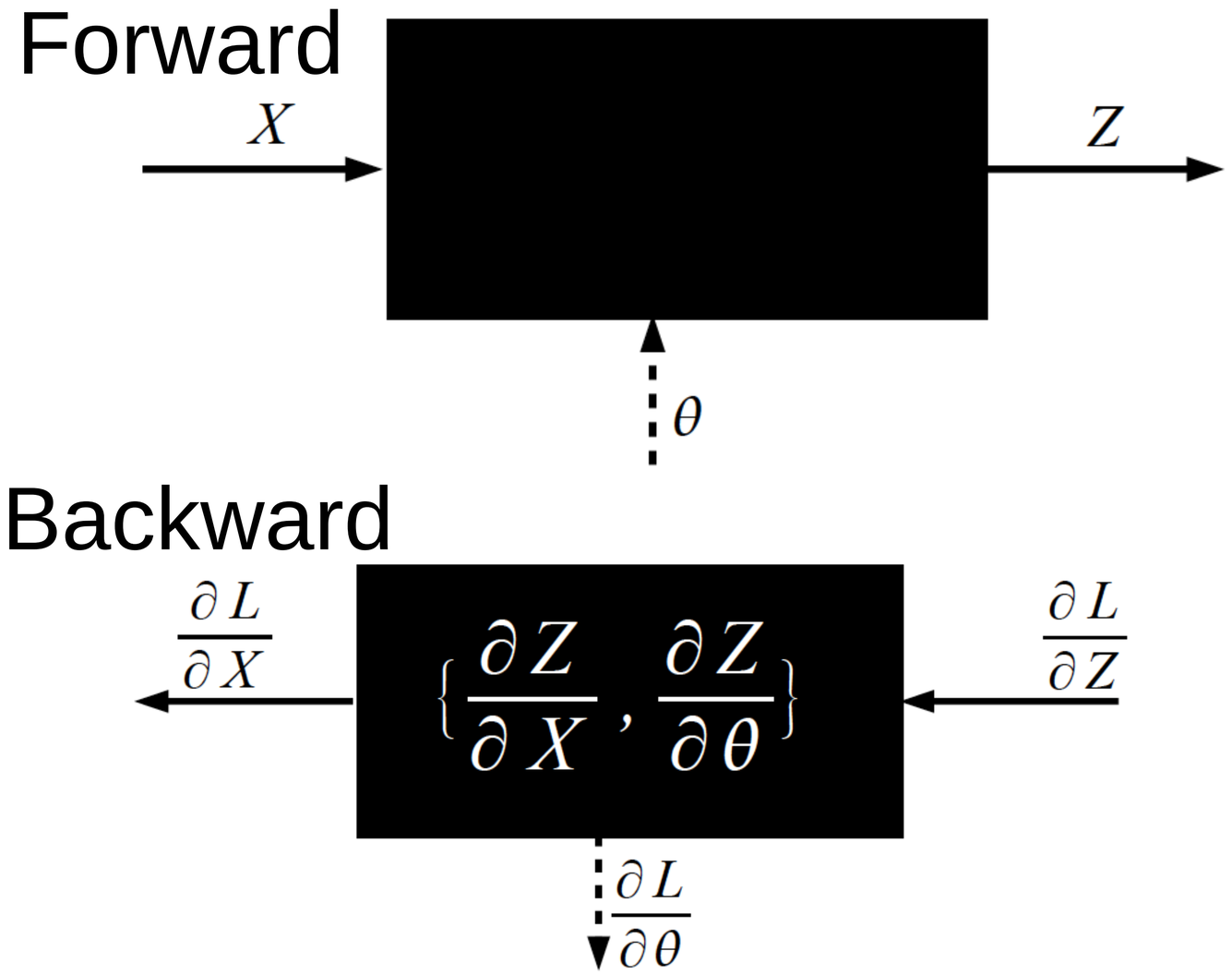
Multiple Local Minima, based on initialization



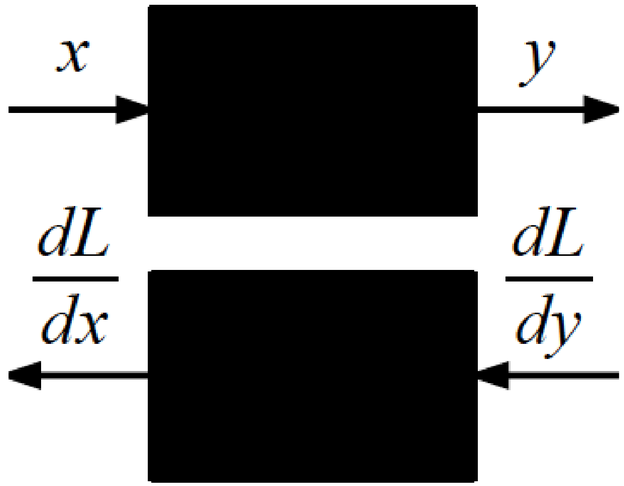
Empirically all are almost equally good

Central research topic: how can this happen? On to the gradients!

All you need is gradients



Chain Rule

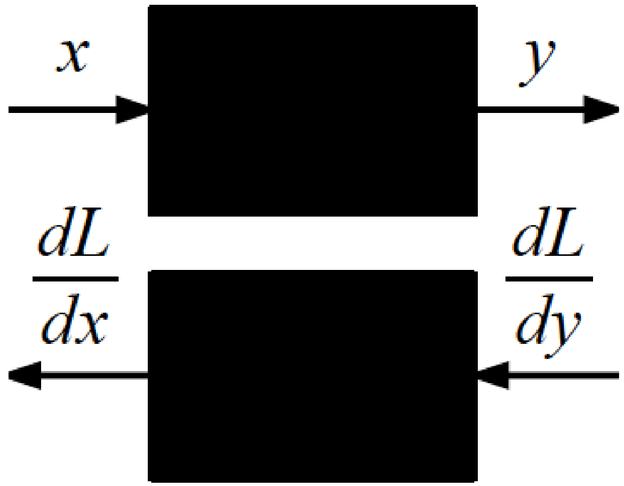


Given $y(x)$ and dL/dy,

What is dL/dx ?

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Chain Rule



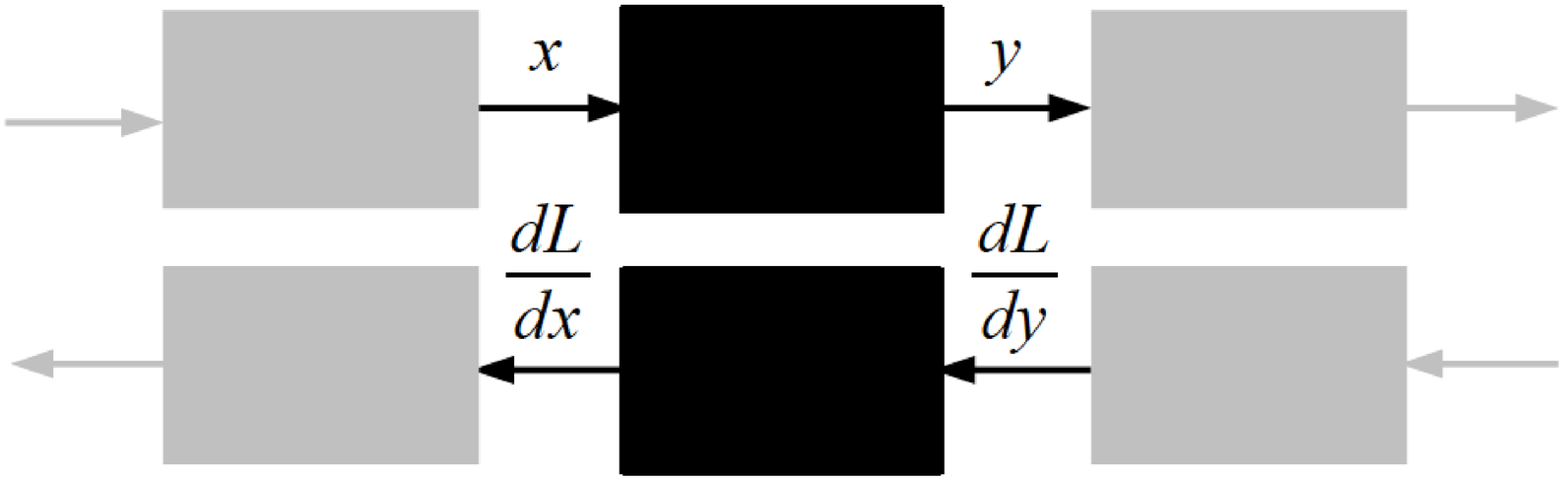
Given $y(x)$ and dL/dy,

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

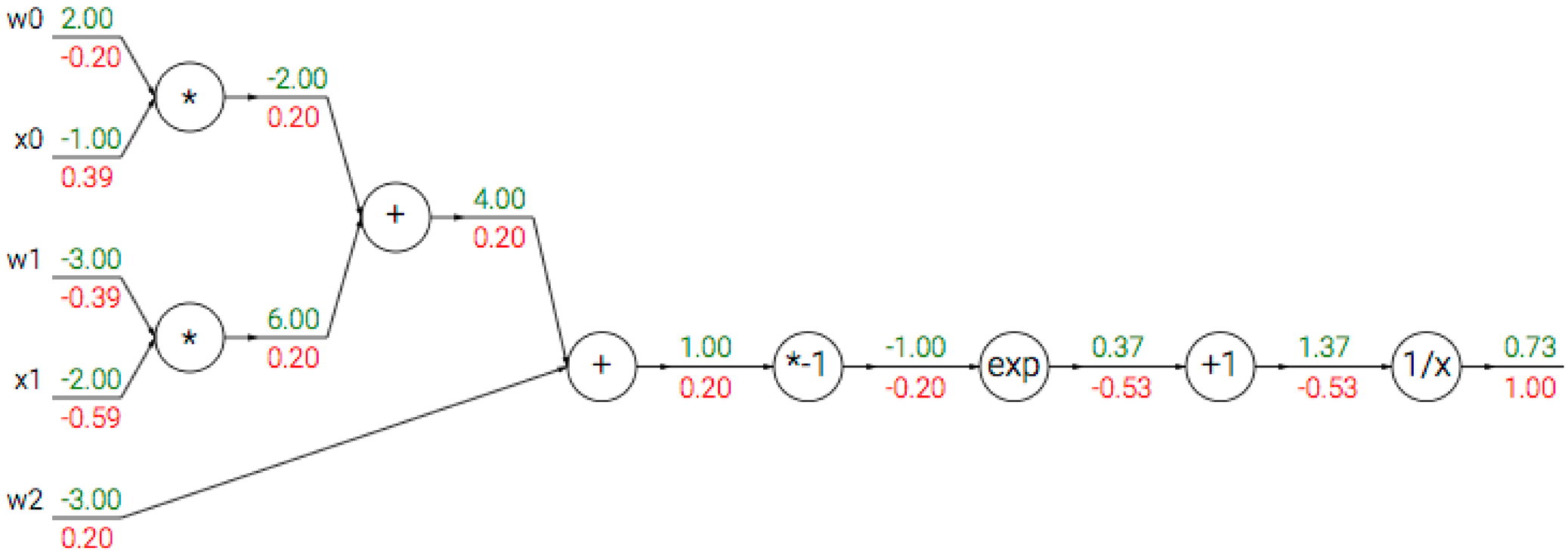
Toy example: single sigmoidal unit

$$f(w, x) = \frac{1}{1 + \exp(-(w_0 x_0 + w_1 x_1 + w_2))}$$

Composition of differentiable blocks:

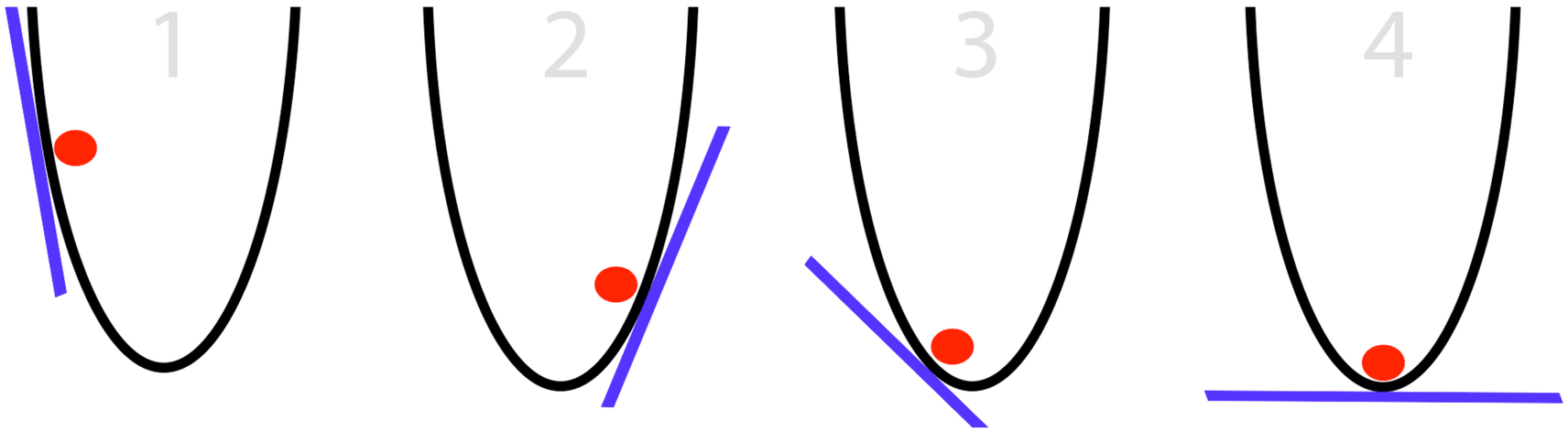
$$\begin{aligned}
f(x) = \frac{1}{x} \quad &\rightarrow \quad f'(x) = -\frac{1}{x^2} \\
f_c(x) = c + x \quad &\rightarrow \quad f'(x) = 1 \\
f(x) = e^x \quad &\rightarrow \quad f'(x) = e^x \\
f_a(x) = ax \quad &\rightarrow \quad f'(x) = a
\end{aligned}$$

Computation graph & automatic differentiation

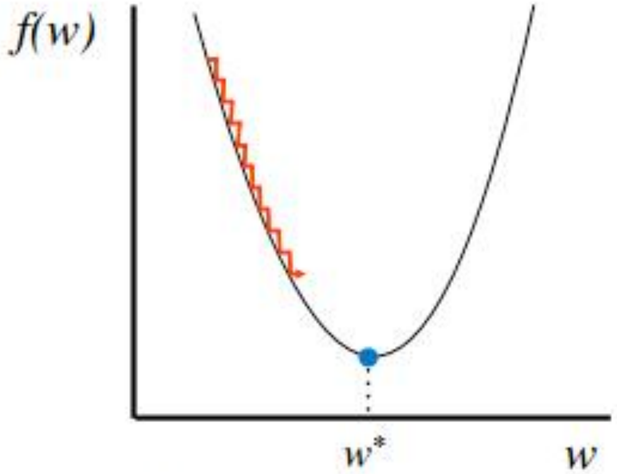


Slide Credit: Justin Johnson

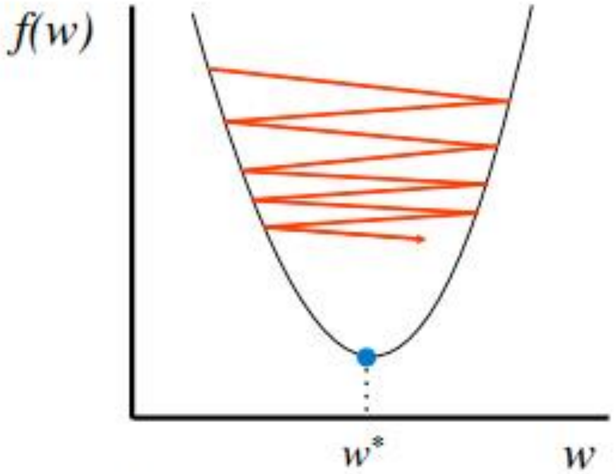
Parameter Updates & Convergence



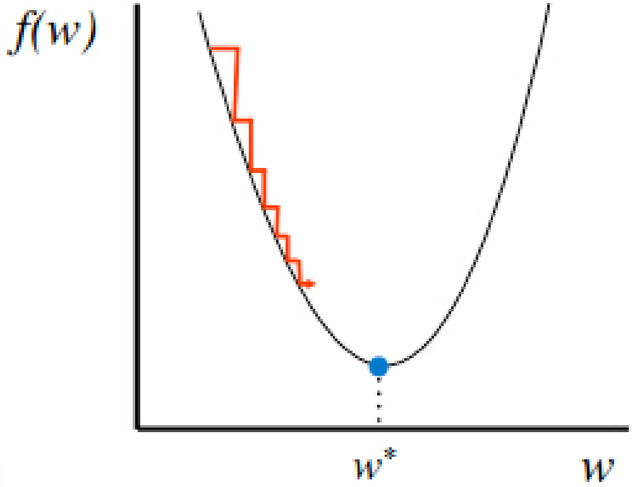
(S)GD with adaptable stepsize



Too small: converge very slowly



Too big: overshoot and even diverge

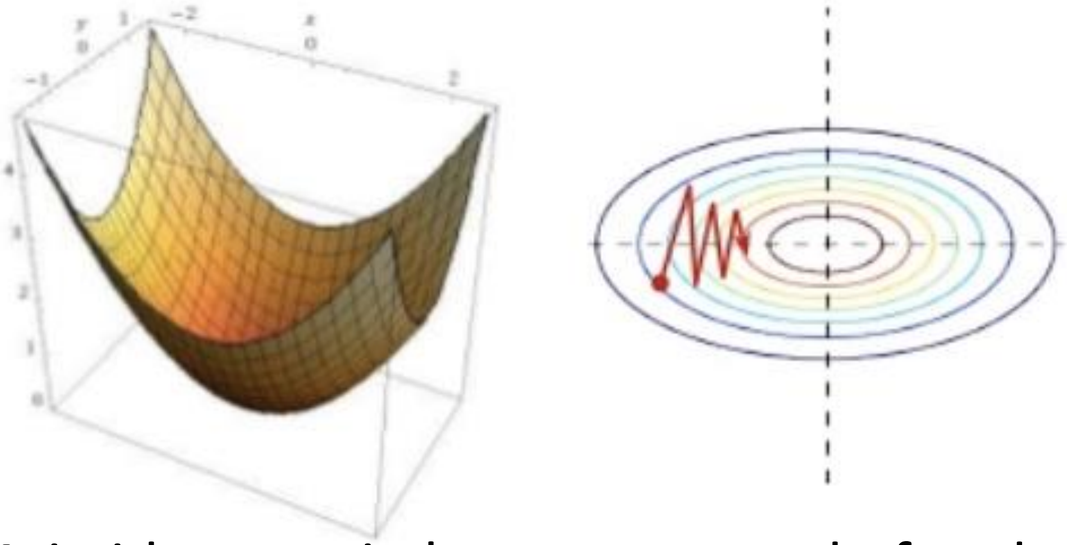


Reduce size over time

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

.

(S)GD with momentum



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

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

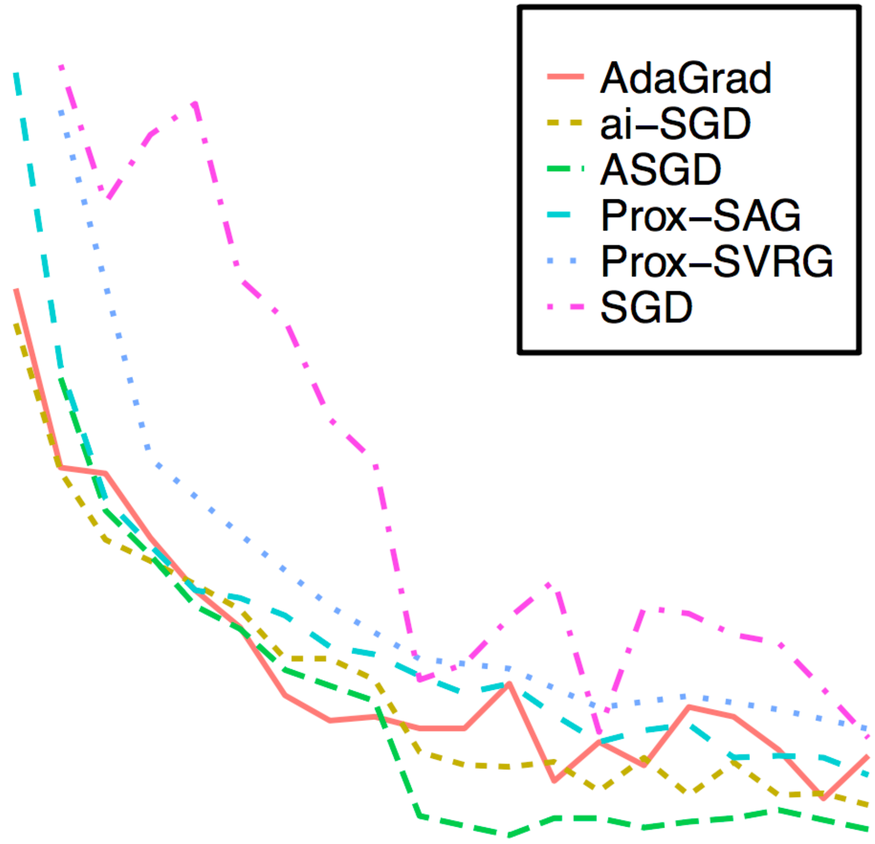
(S)GD + momentum

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

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

Step-size Selection & Optimizers: research problem

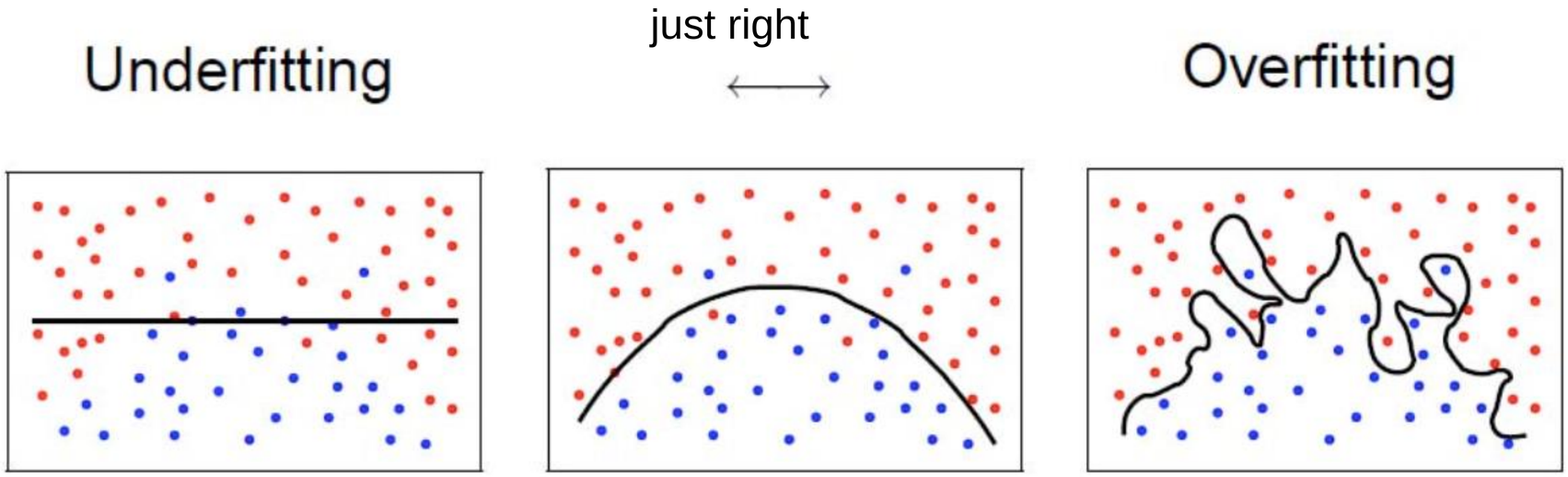
- Nesterov's Accelerated Gradient (NAG)
- R-prop
- AdaGrad
- RMSProp
- AdaDelta
- Adam
- …



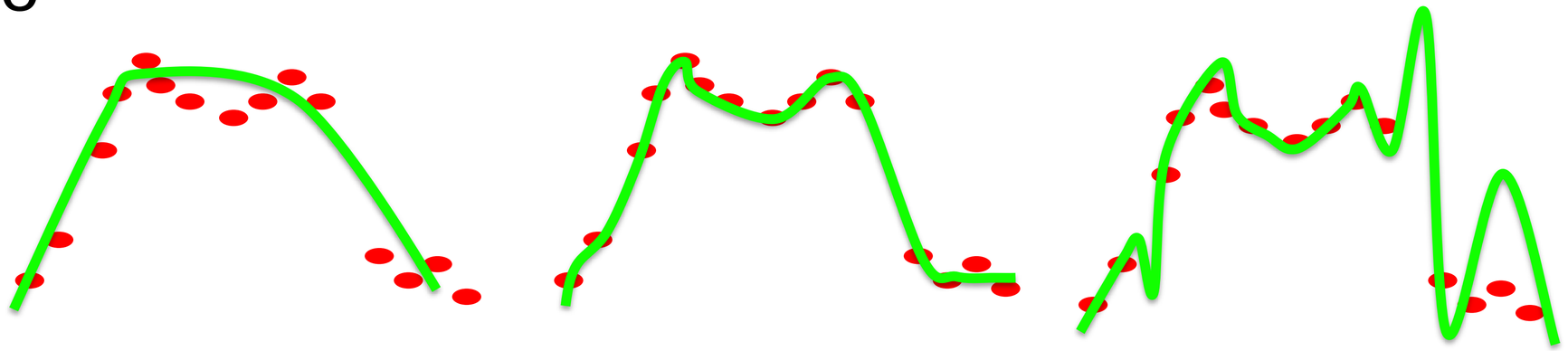
Overfitting and Regularization

Overfitting, in images

Classification



Regression



1980’s solution: l2 Regularization

Regularization: preference for small parameters

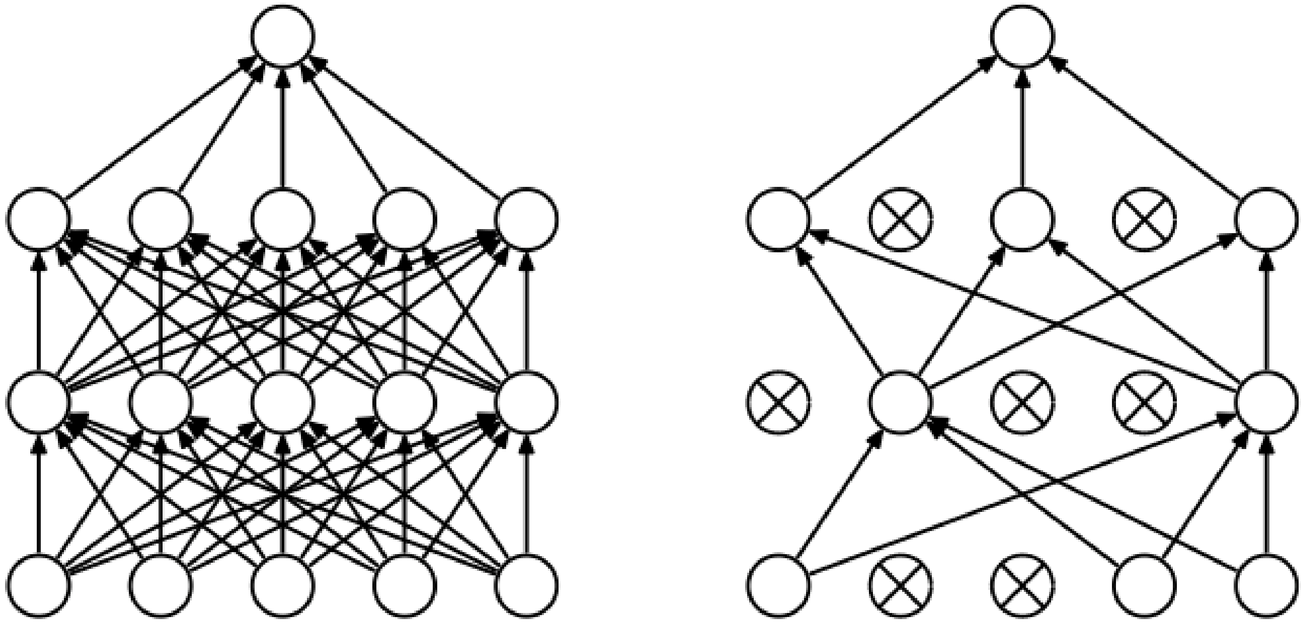
$$L(\mathbf{W}) = \frac{1}{N}\sum_{i=1}^{N} l(\mathbf{y}^i, \hat{\mathbf{y}}^i) + \sum_l \lambda_l \sum_{k,m} (\mathbf{W}^l_{k,m})^2$$

Per-sample loss

Per-layer regularization

Integration with gradient-descent: “weight decay”

2010’s solution: Dropout

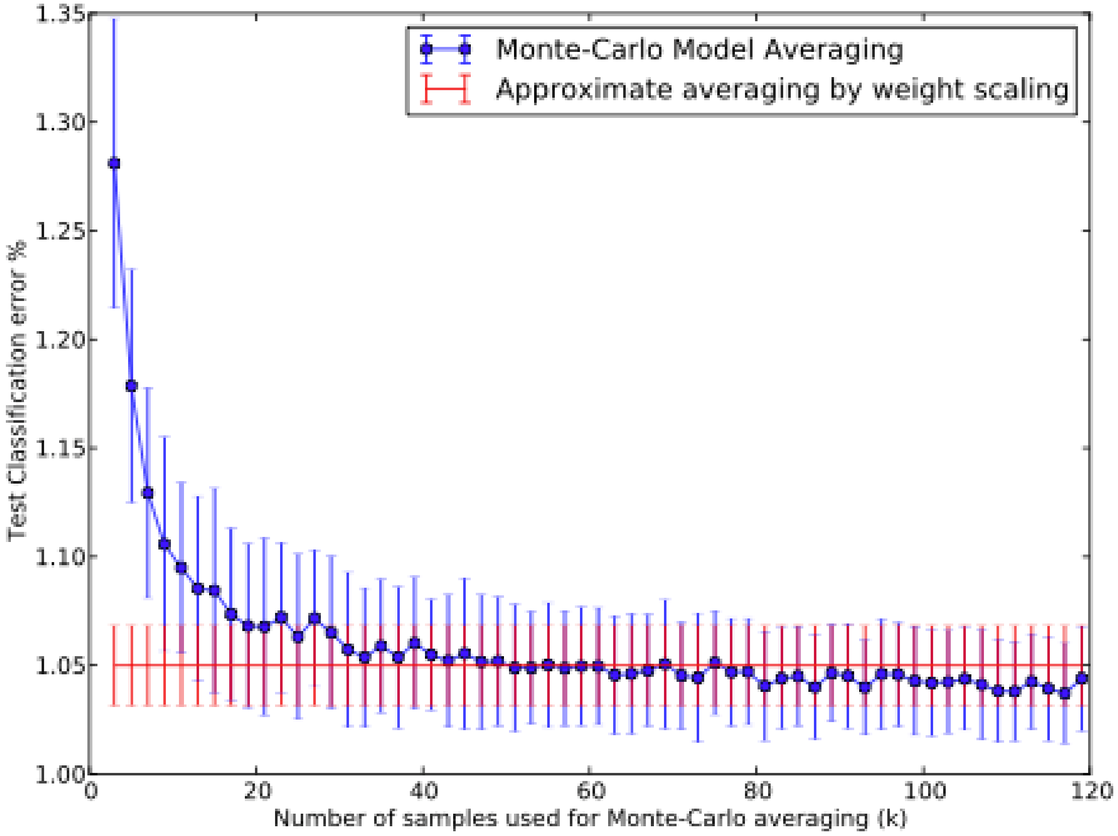


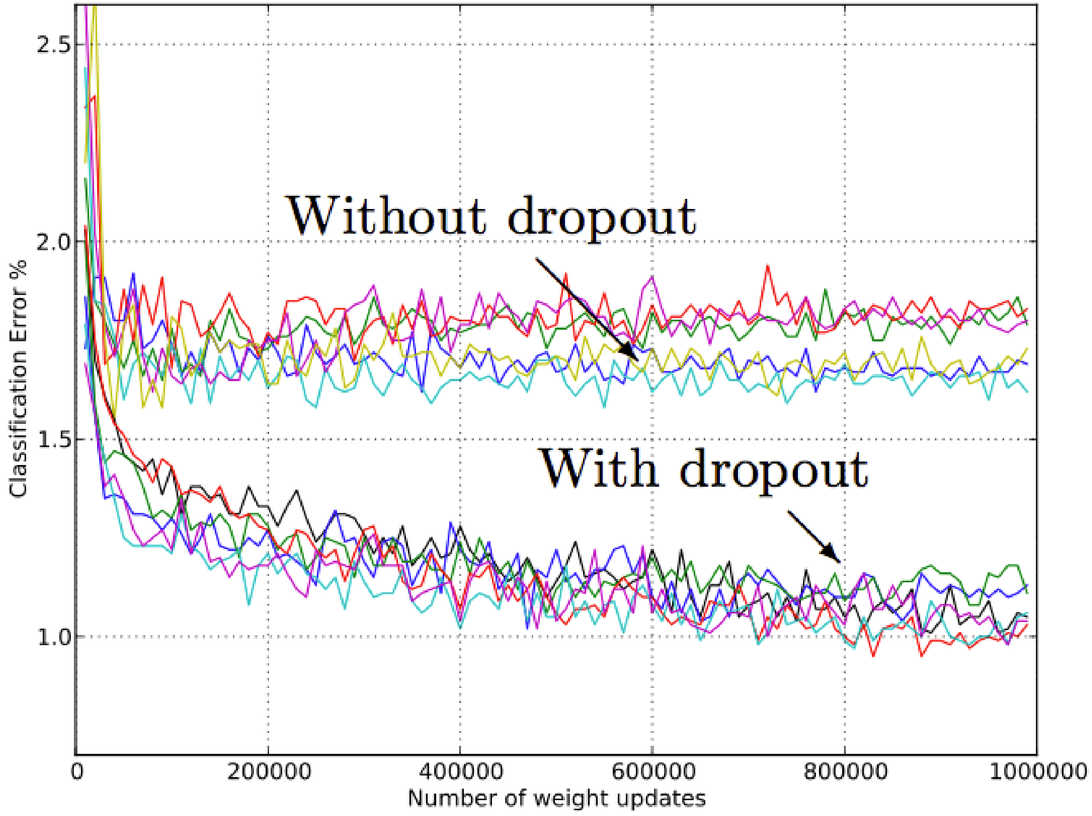
(a) Standard Neural Net

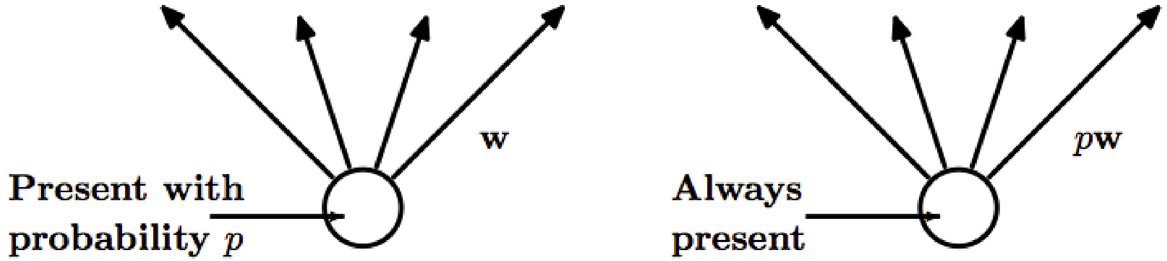
(b) After applying dropout.

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

Test time: Deterministic Approximation

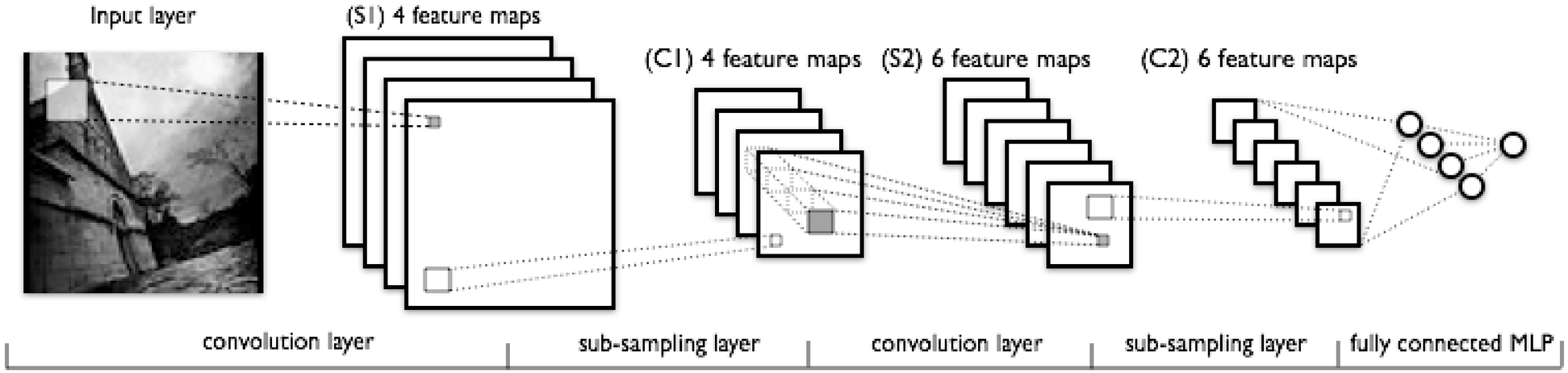


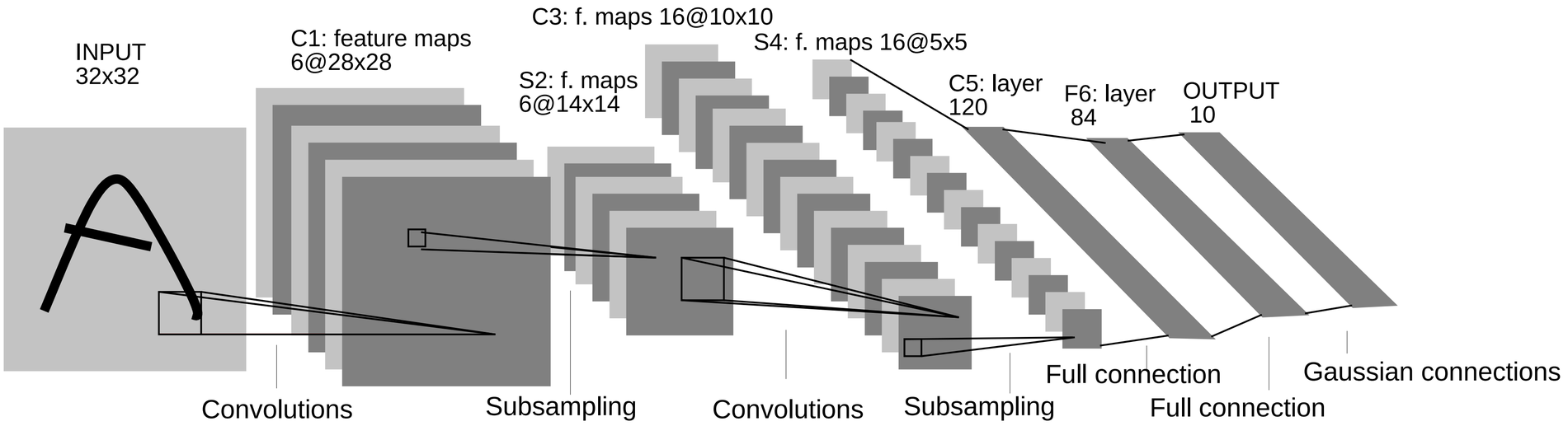




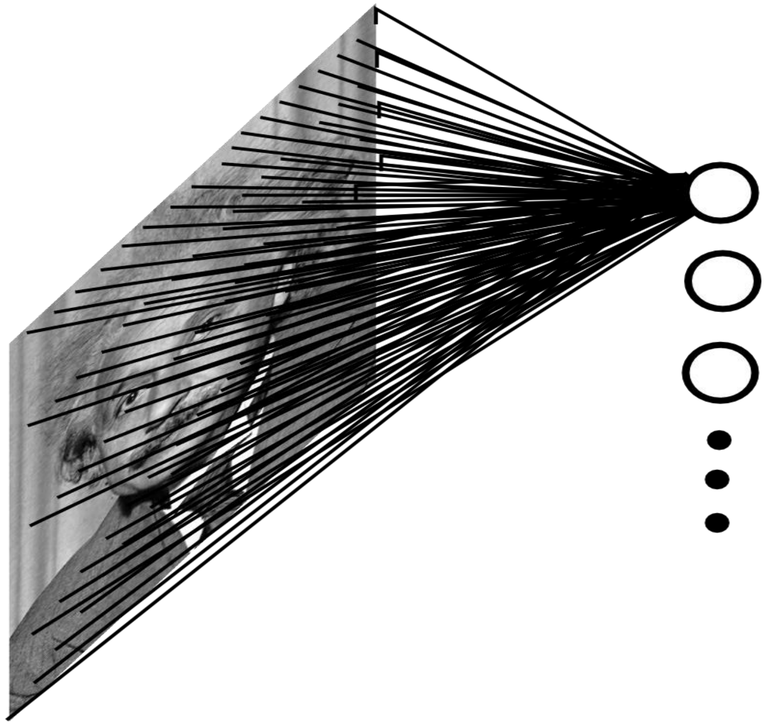
(a) At training time (b) At test time

Convolutional Neural Networks (CNNs/ConvNets)





“Fully-connected” layer

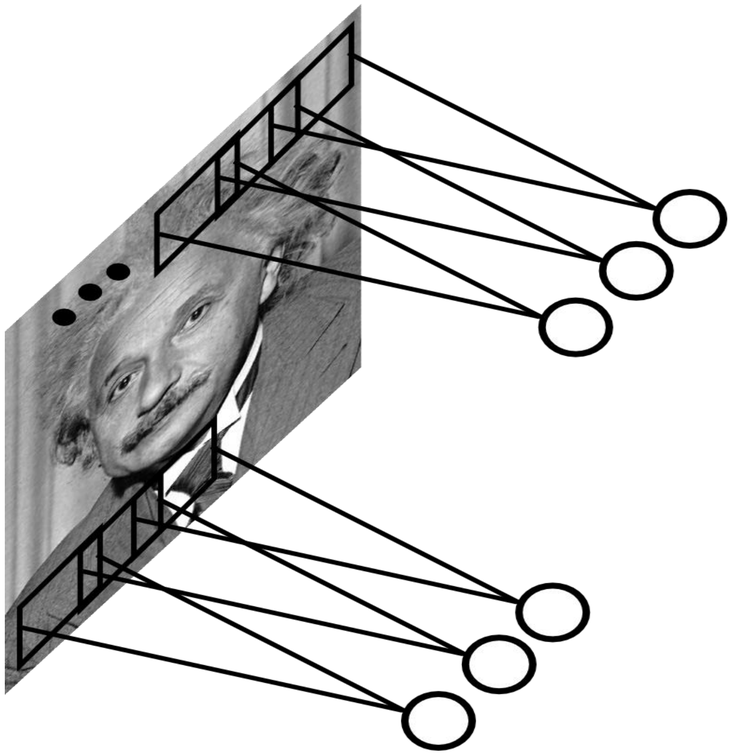


#of parameters: K^2

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

slow, big, prone to overfitting

Idea: repeat a local operation (image is stationary)

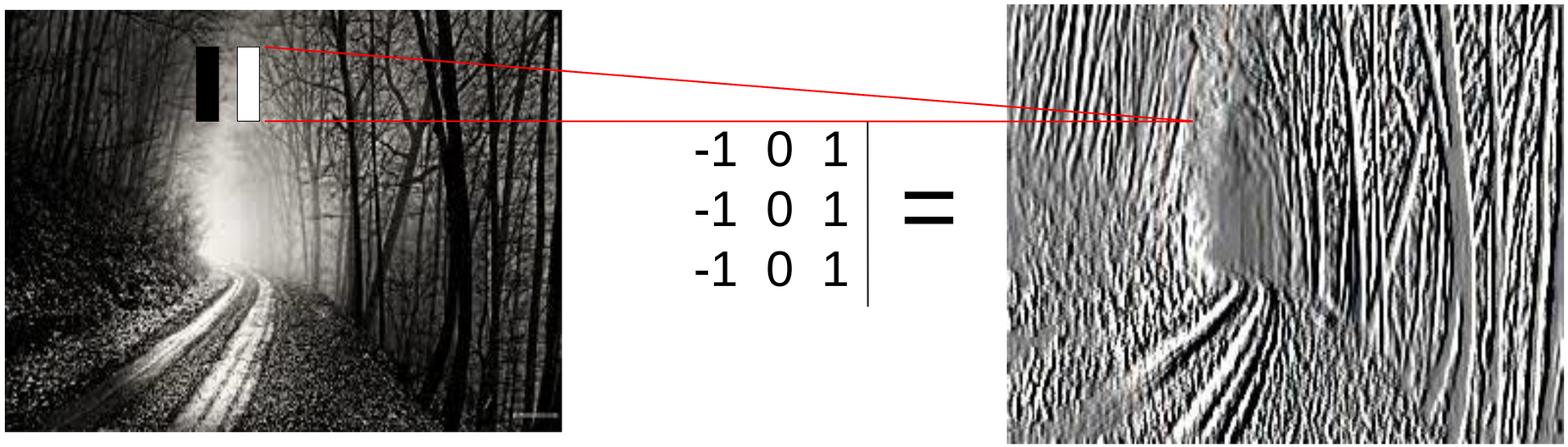


#of parameters: size of window

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

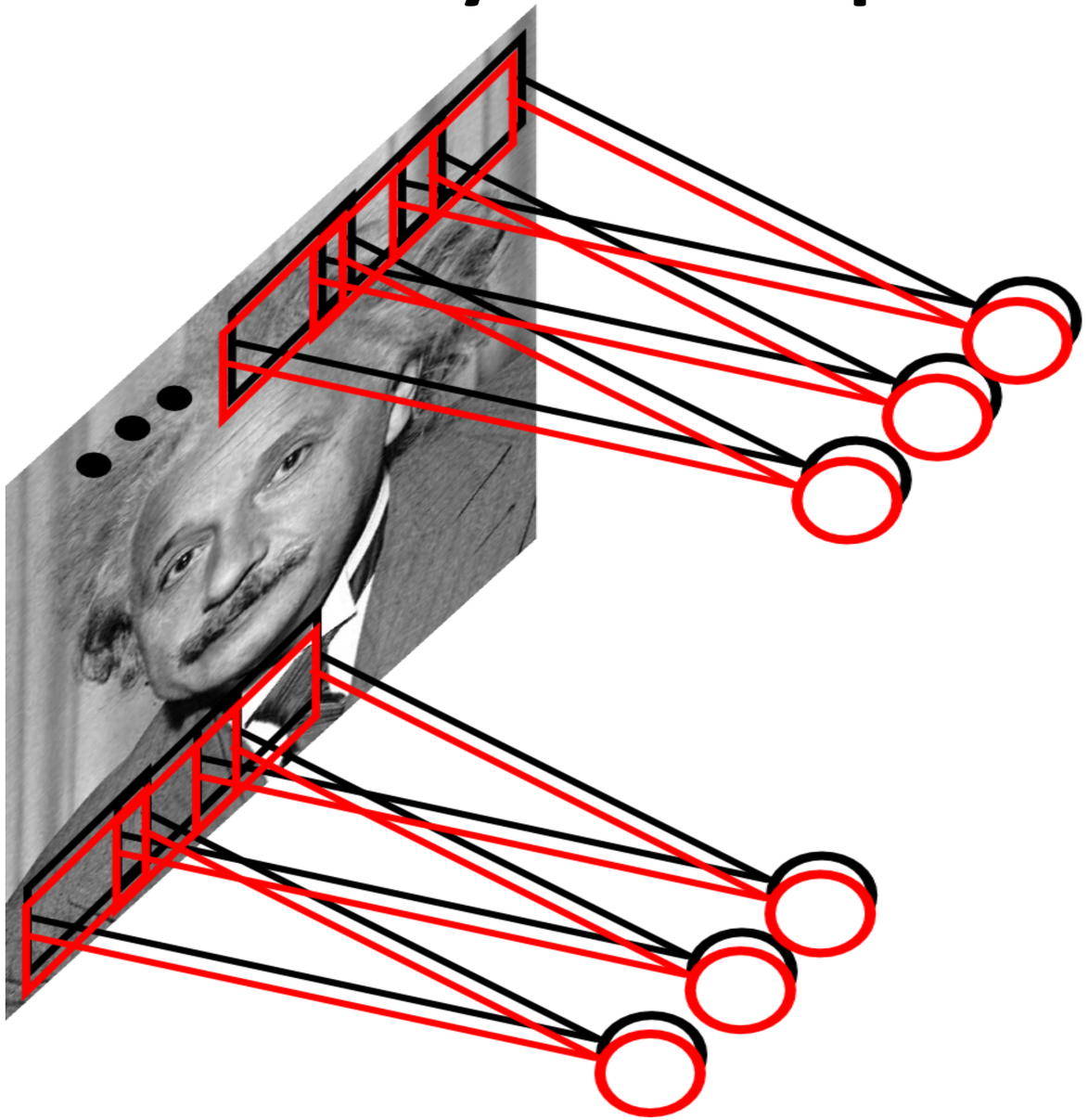
fast, small, regularized

Identical to convolution operation



But with learnable weights!

Convolutional layer: multiple output channels



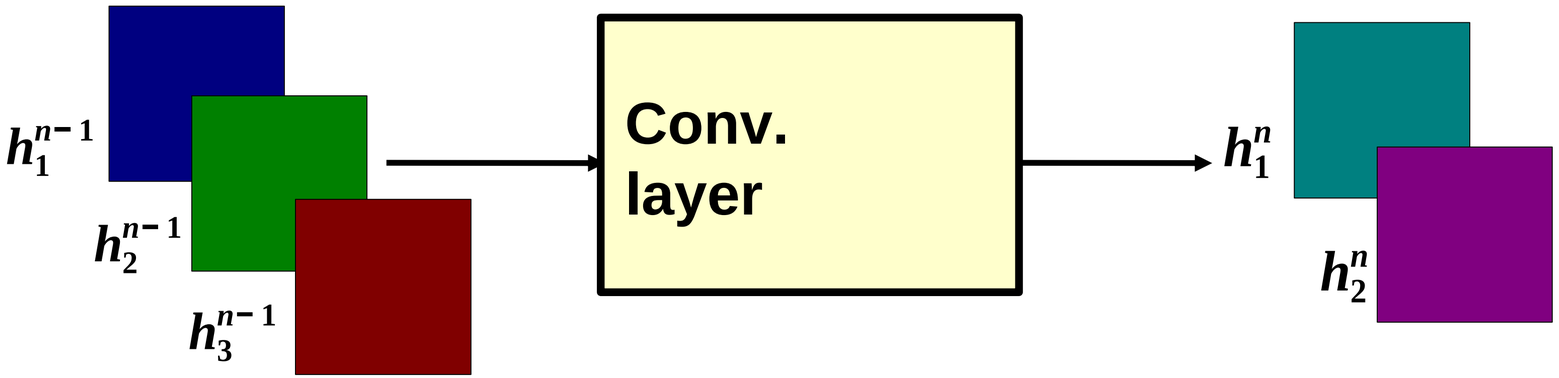
Convolutional layer composition

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

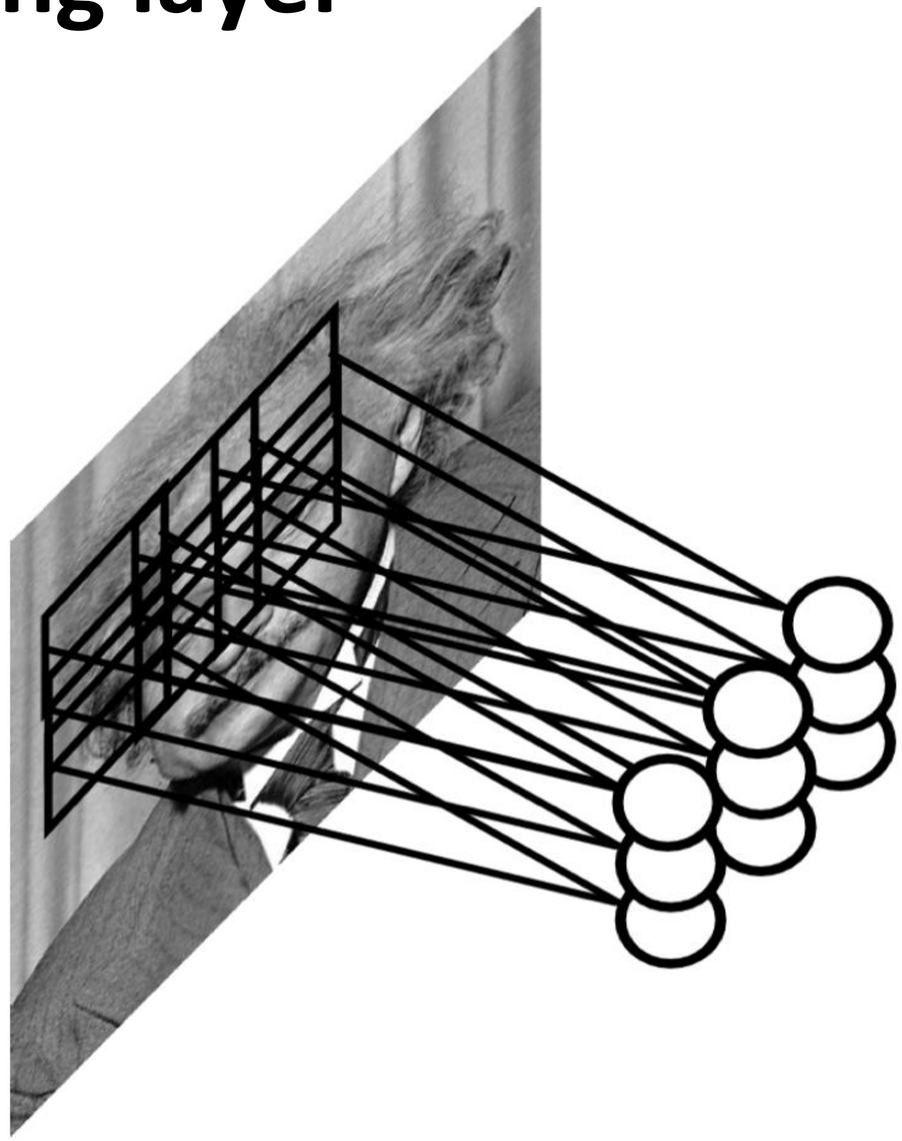
output feature map

input feature map

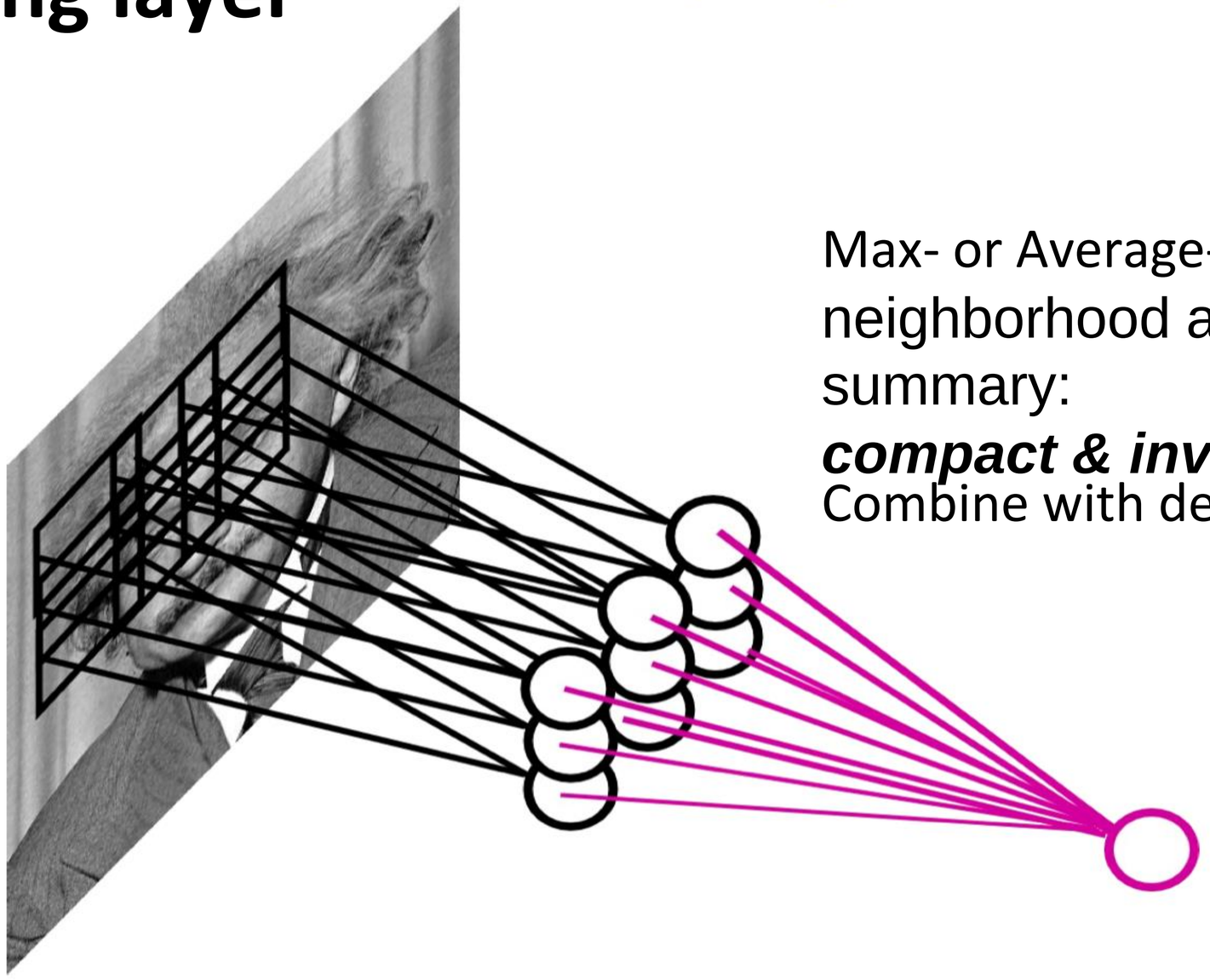
kernel



Pooling layer



Pooling layer



Max- or Average-based neighborhood activation summary:

compact & invariant

Combine with decimation!

Receptive field

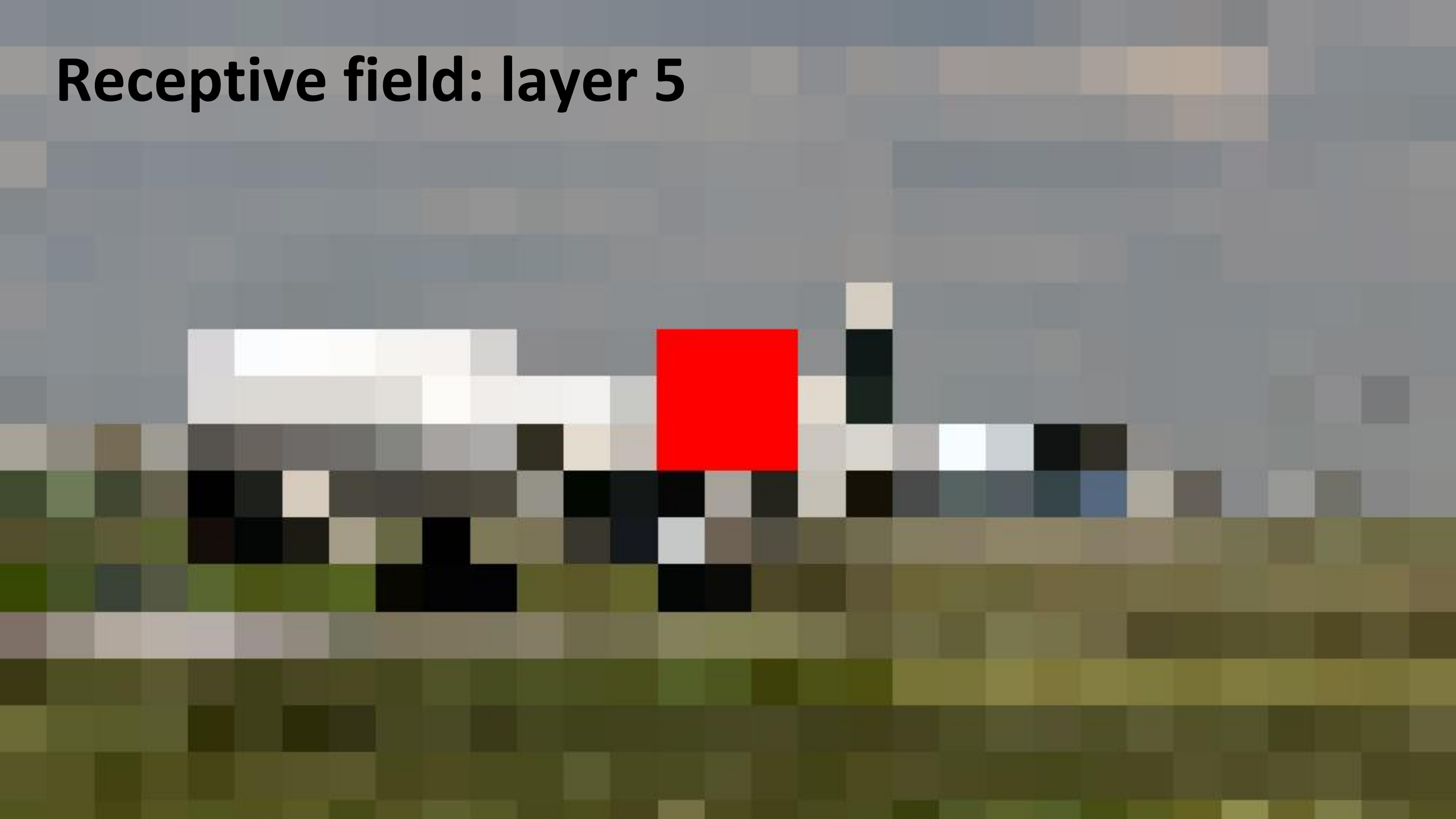
Receptive field: layer 1

Receptive field: layer 2

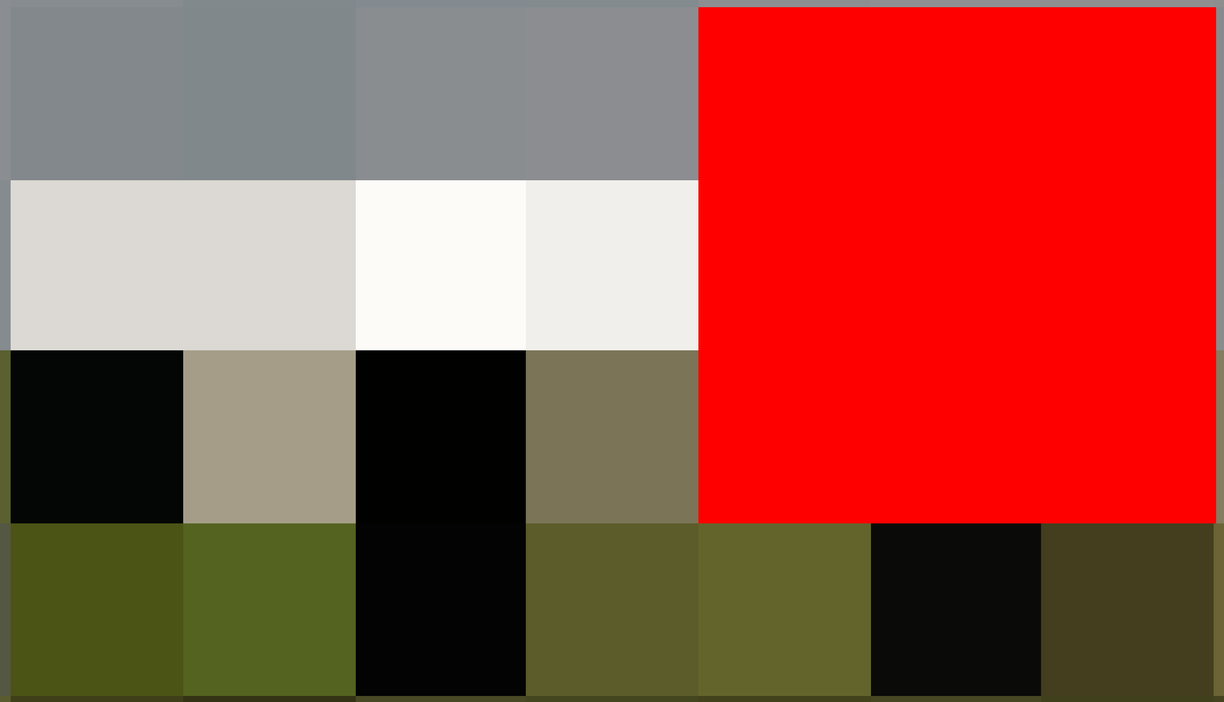
Receptive field: layer 3

Receptive field: layer 4

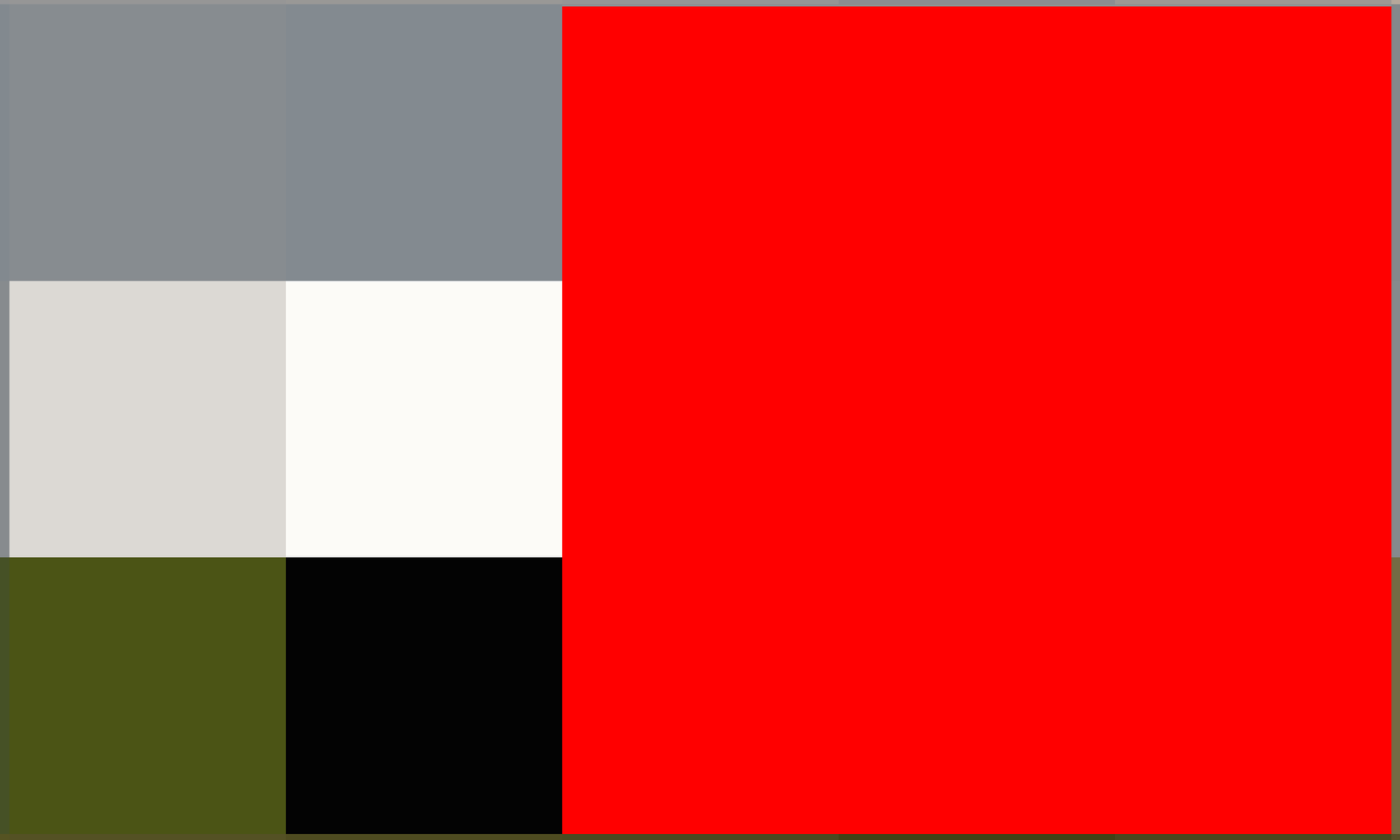
Receptive field: layer 5



Receptive field: layer 6



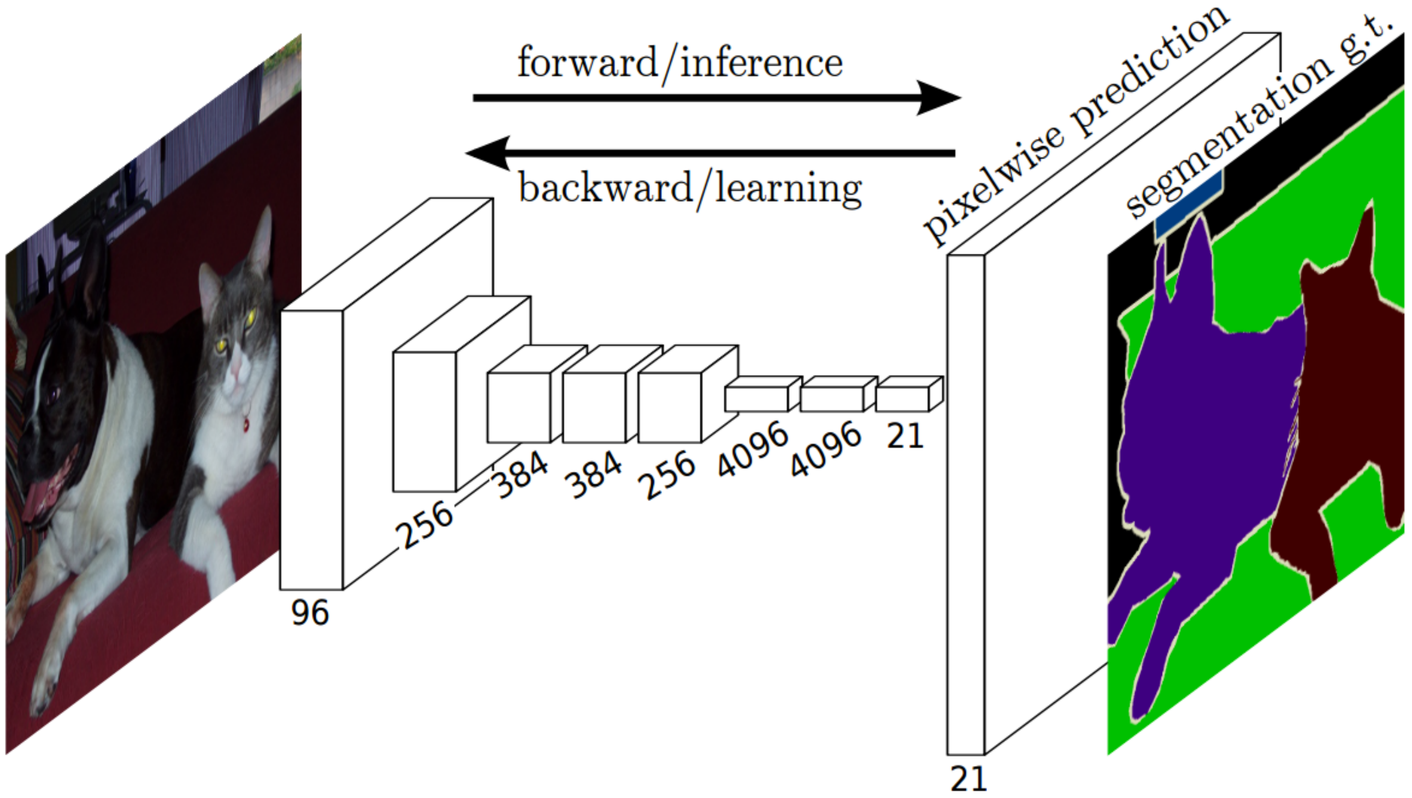
Receptive field: layer 7



Receptive field: layer 8

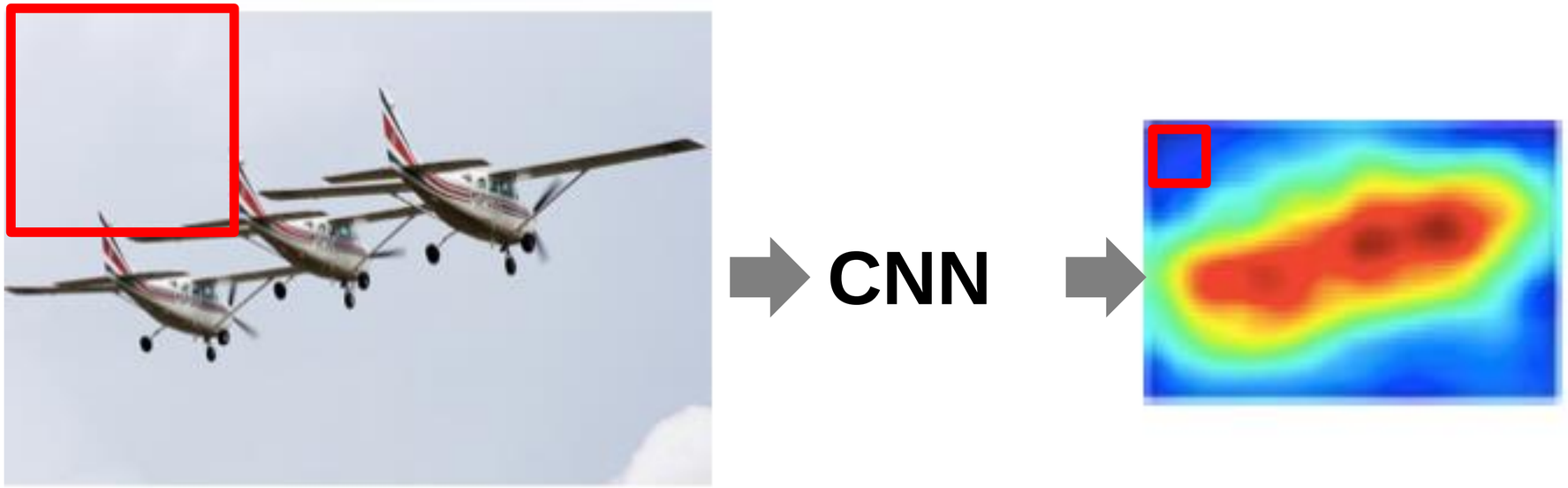
CNNs for Image Understanding

Fully-convolutional Neural Networks for X

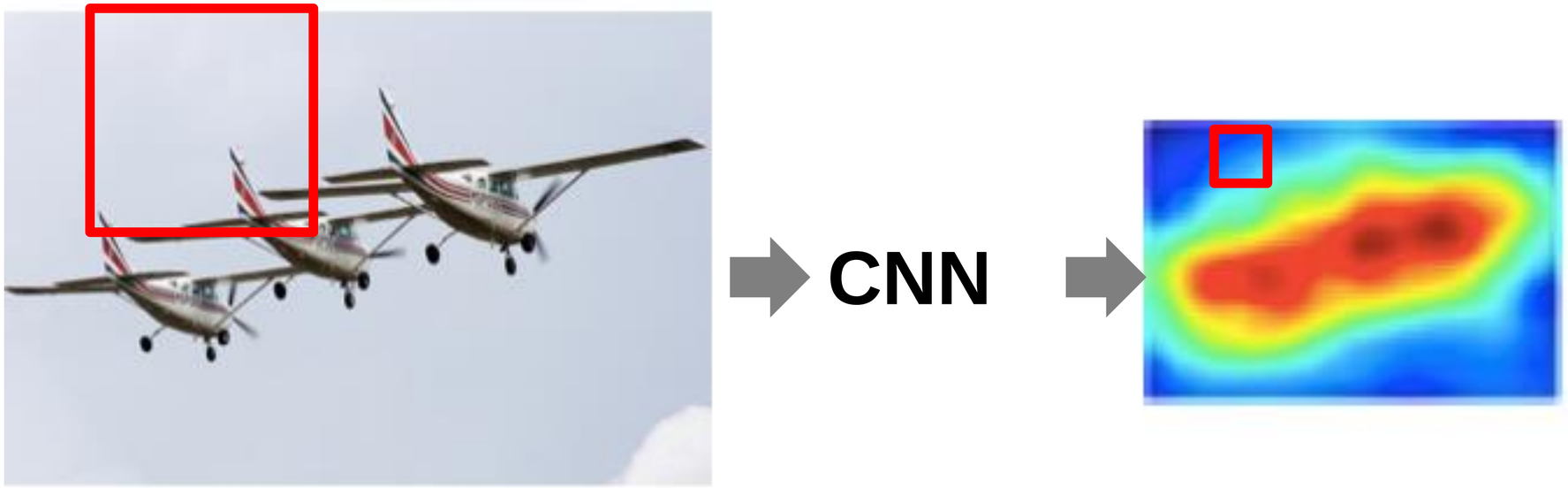


J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation.*CVPR*, 2015

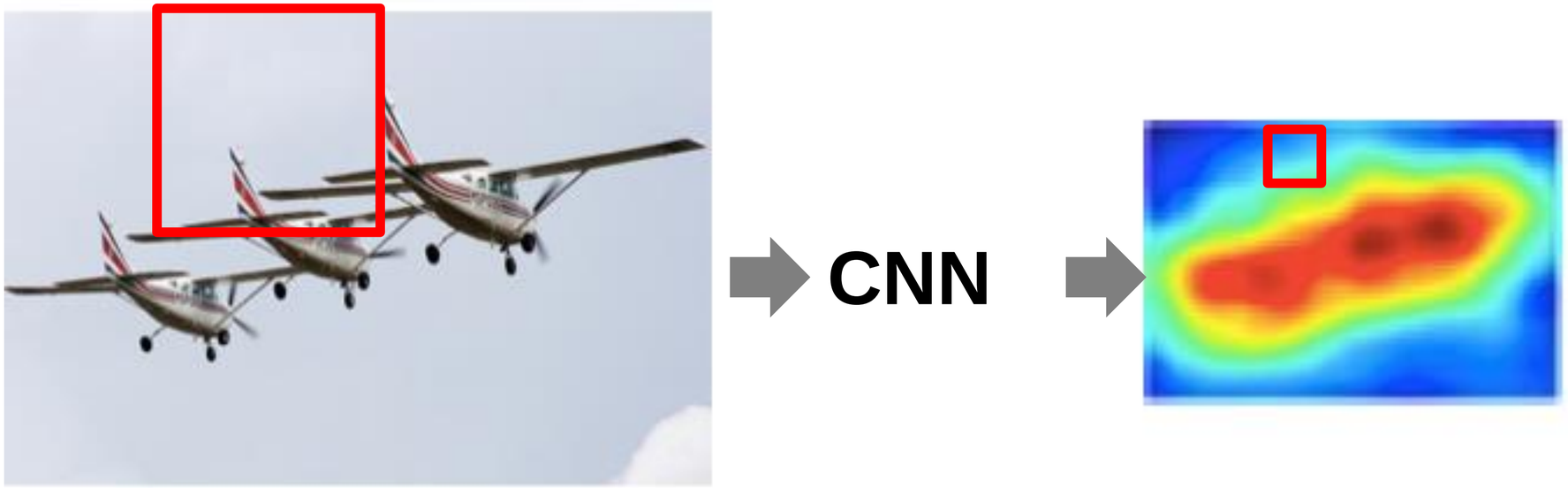
Fully-convolutional Neural Networks



Fully-convolutional Neural Networks

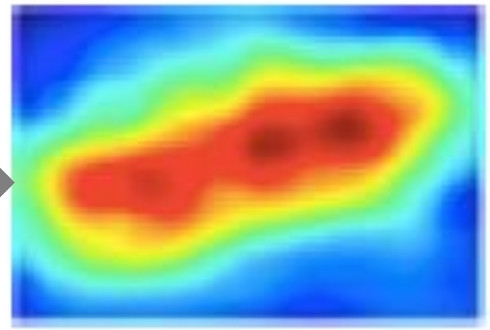


Fully-convolutional Neural Networks

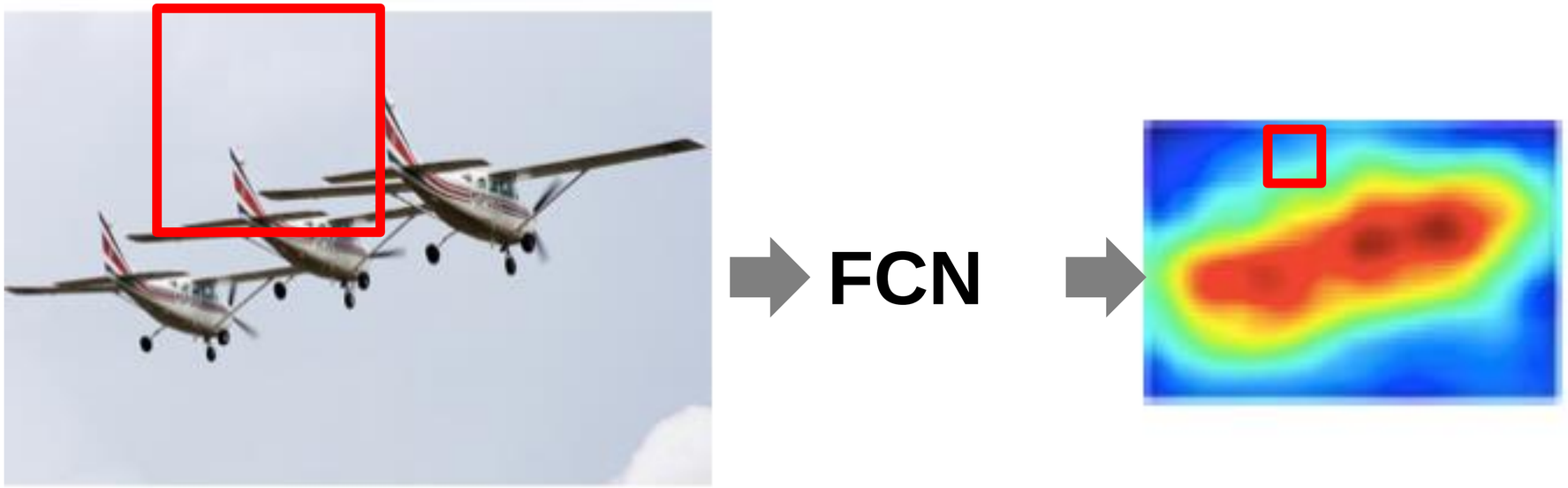


Fully-convolutional Neural Networks

FCN



Fully-convolutional Neural Networks

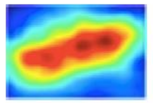


Fast (shared convolutions)
Simple (dense)

Fully Convolutional Neural Networks in Practice

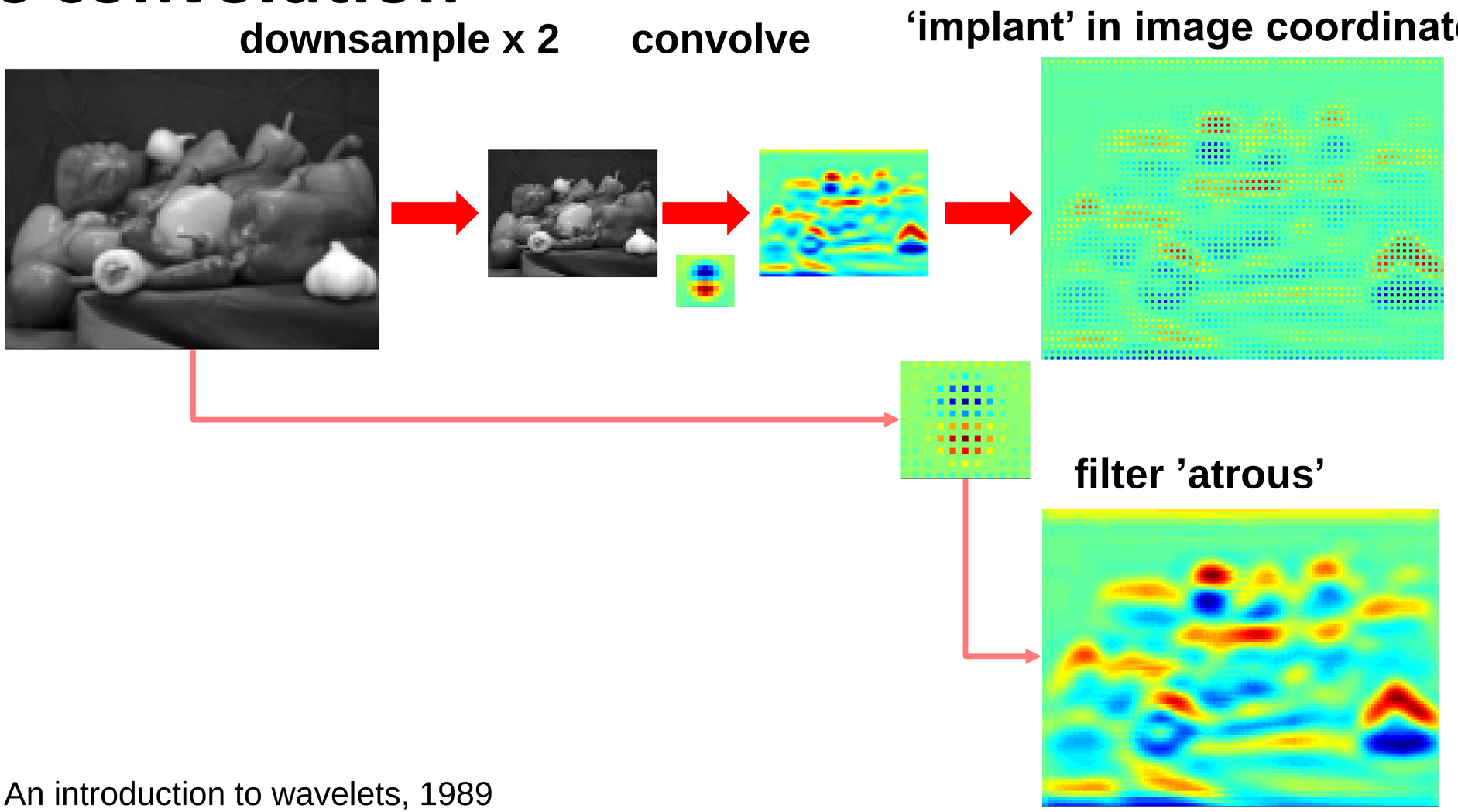
32-fold decimation
224x224 to 7x7

FCN



Fast (shared convolutions)
Simple (dense)
Low resolution

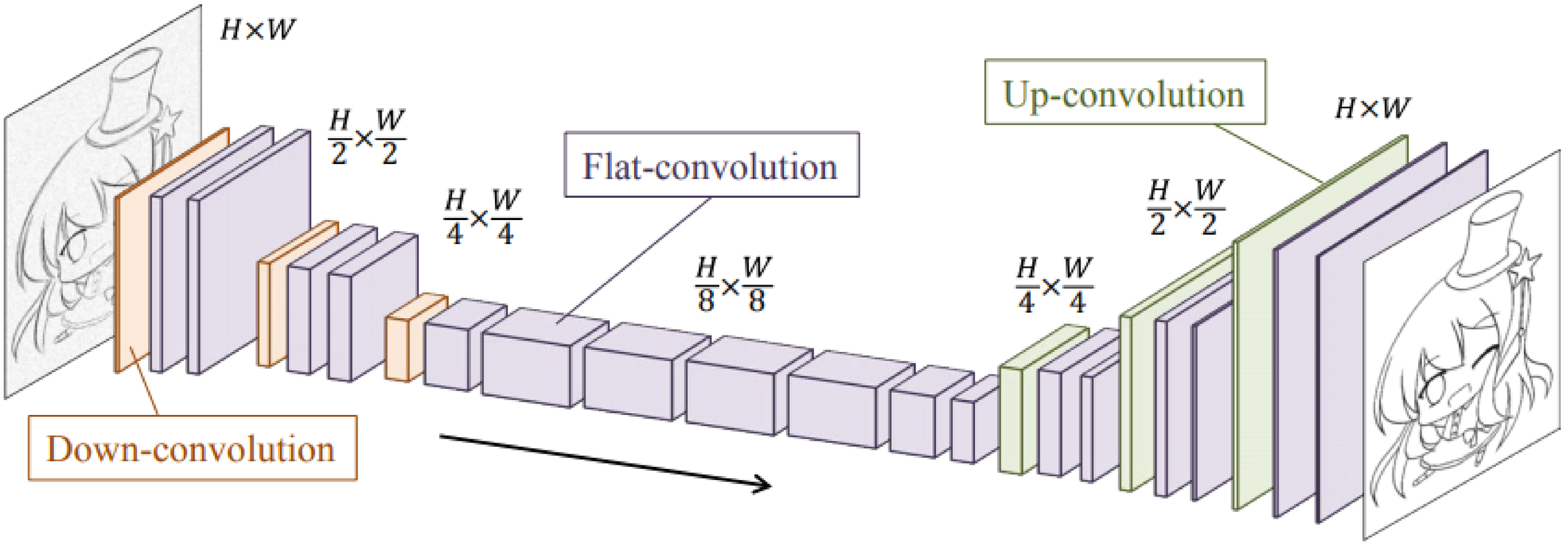
Atrous convolution



S. Mallat, An introduction to wavelets, 1989

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

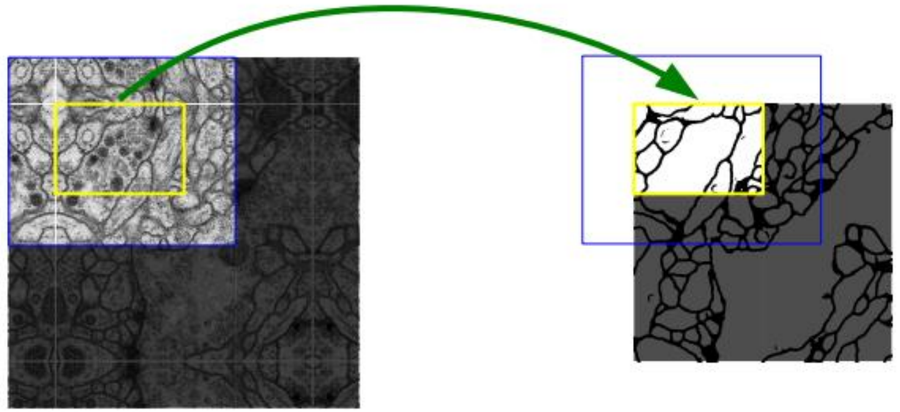
Encoder-decoder

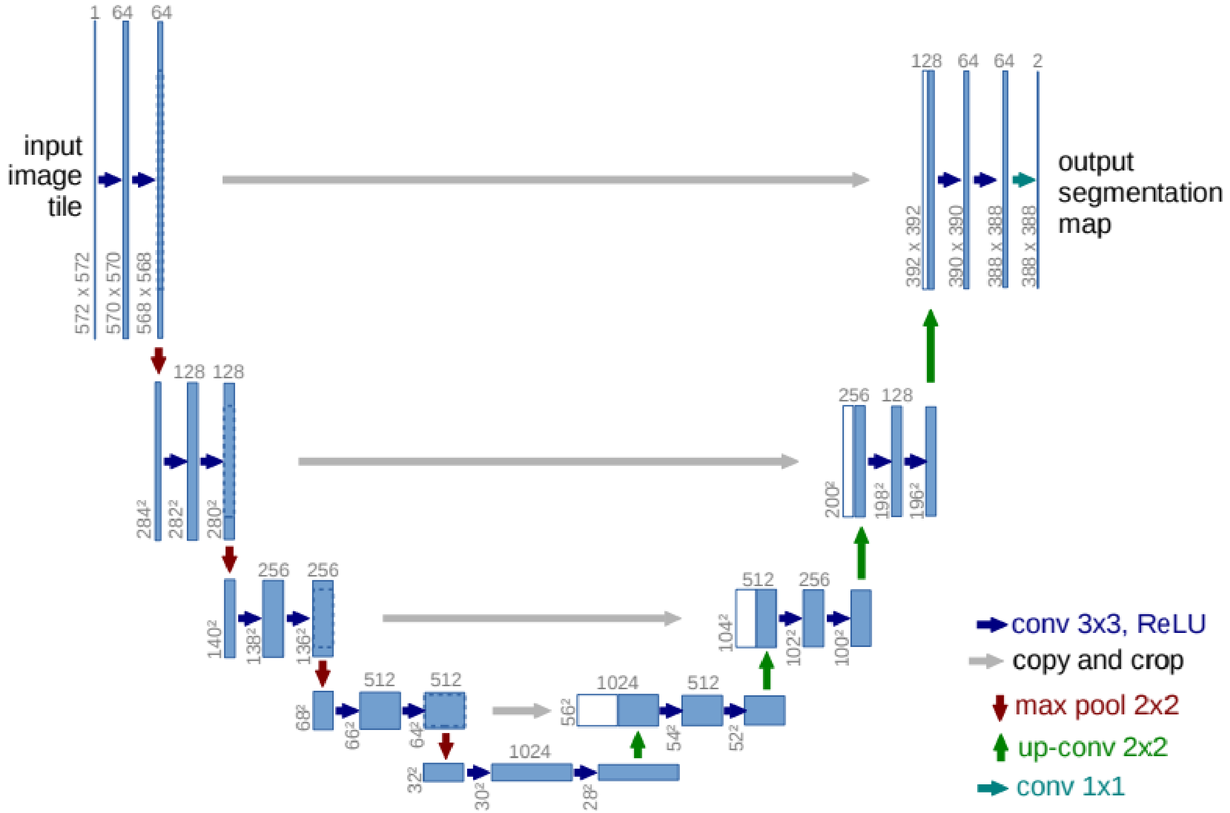


Learning to simplify. Simo-Serra et al. 2016

Encoder-decoder + Skip connections

- 1st: Reduce resolutions as before
- 2nd: Increase resolution
- Transposed convolutions



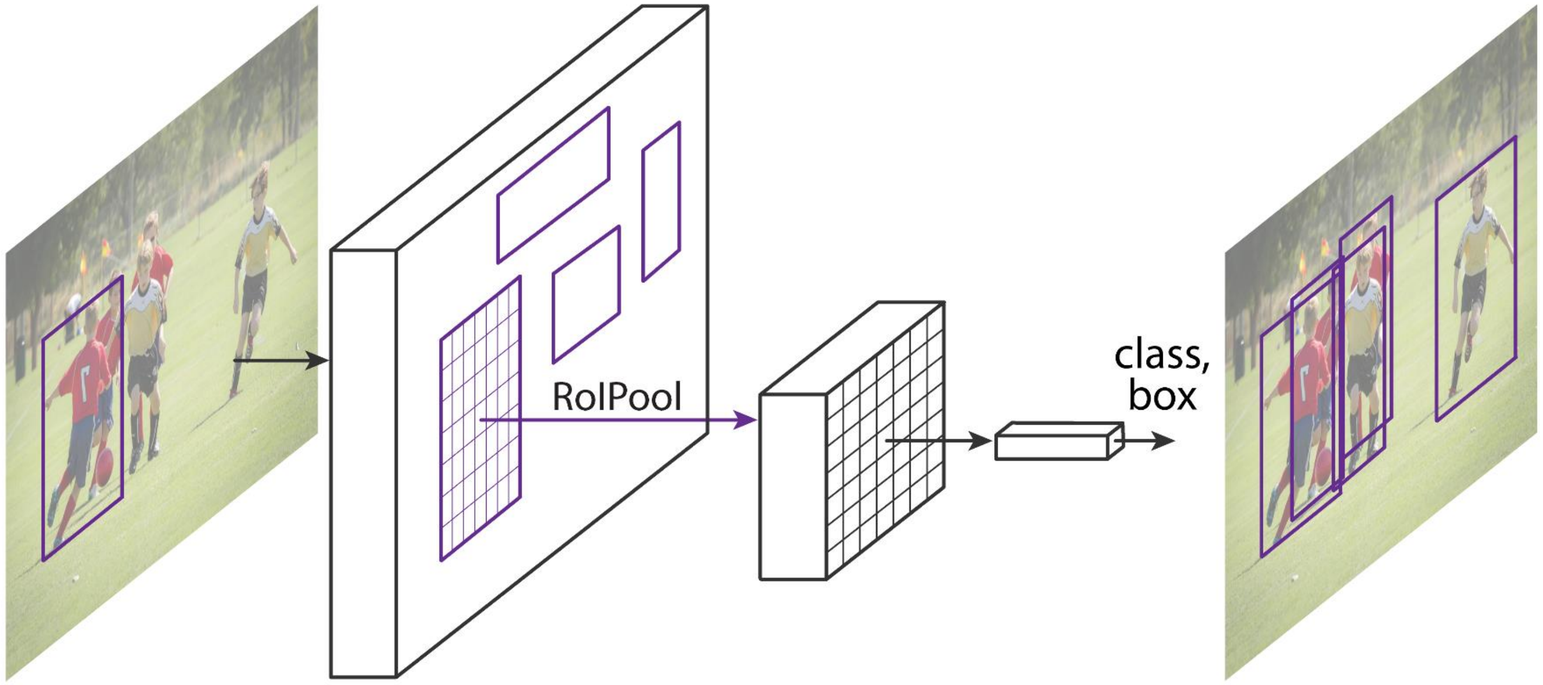


U-Net: Convolutional Networks for Biomedical Image Segmentatio. Ronneberger et al. 2015

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

Object Detection: Fast(er)-RCNN

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

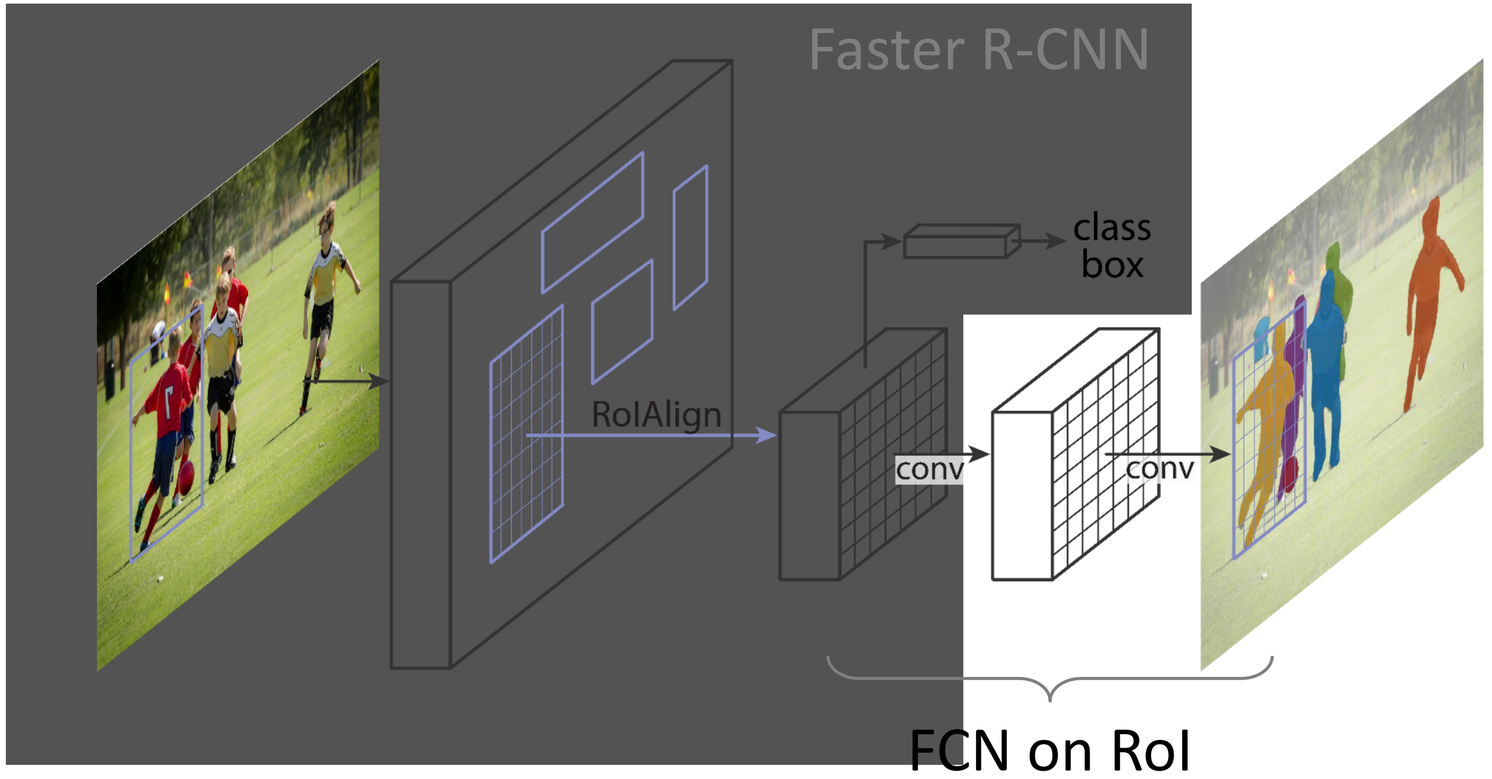


Ross Girshick. “Fast R-CNN”. ICCV 2015.

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

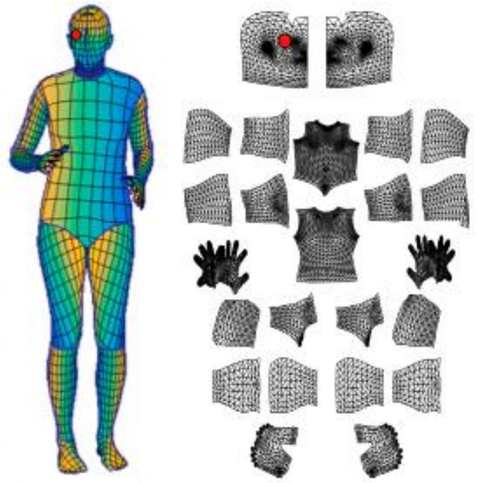
- Mask R-CNN = **Faster R-CNN** with **FCN** on RoIs



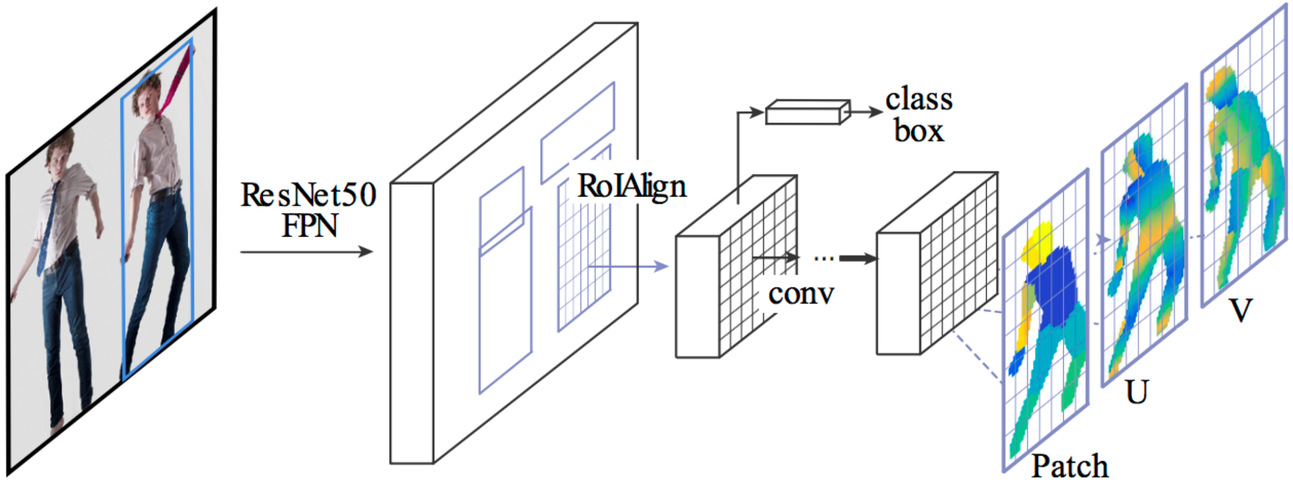
Mask R-CNN frame-by-frame

DensePose: dense image-to-body correspondence

DensePose-RCNN Results



DensePose COCO Dataset

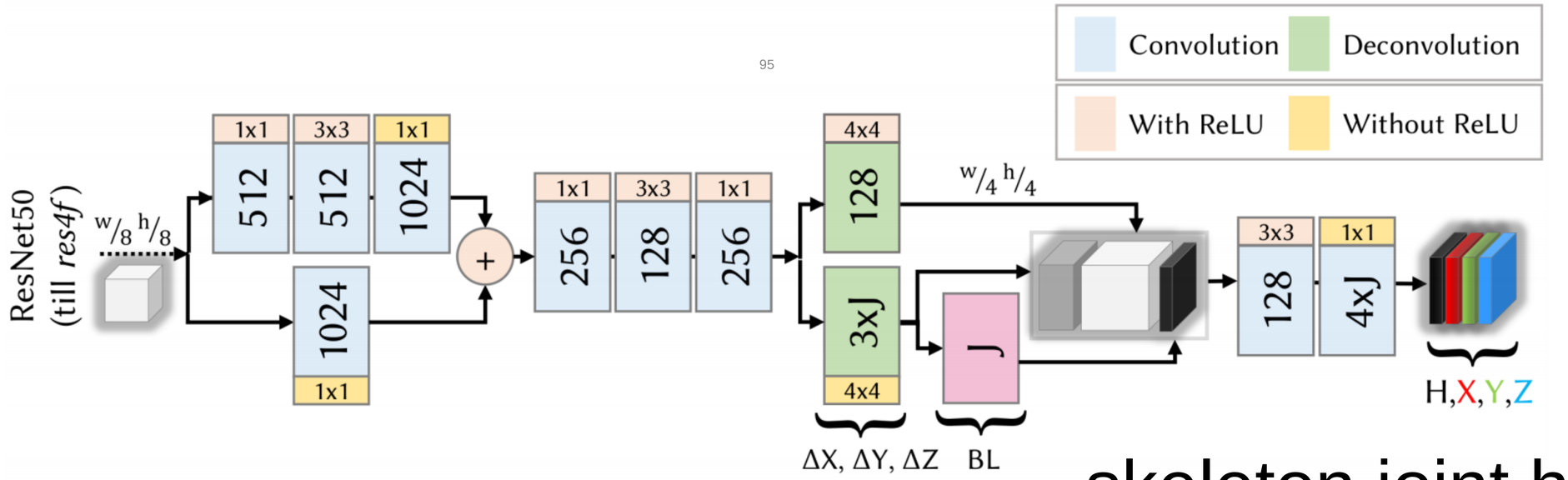


DensePose-RCNN: ~25 FPS

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

DensePose estimation frame-by-frame

3D Pose Estimation: *VNECT*



skeleton joint heatmap and 3d positions

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

HoloPose estimation frame-by-frame

Ariel AI

HoloPose: Holistic 3D Human Reconstruction In-the-Wild, A. Guler and I. Kokkinos, CVPR 201

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)

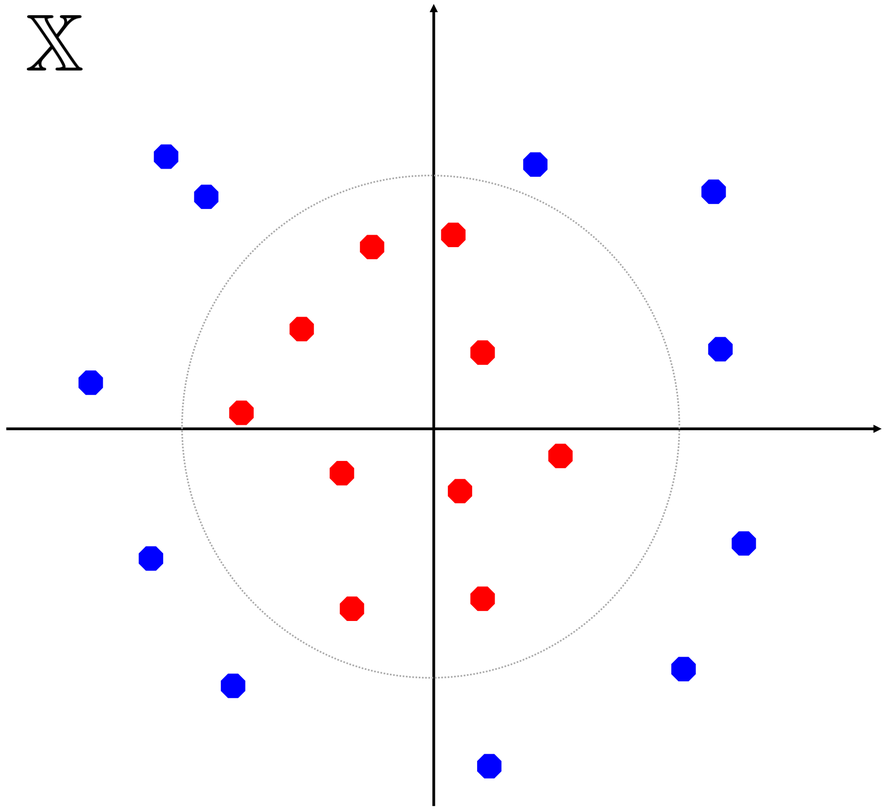
(Python, Java, Scala)

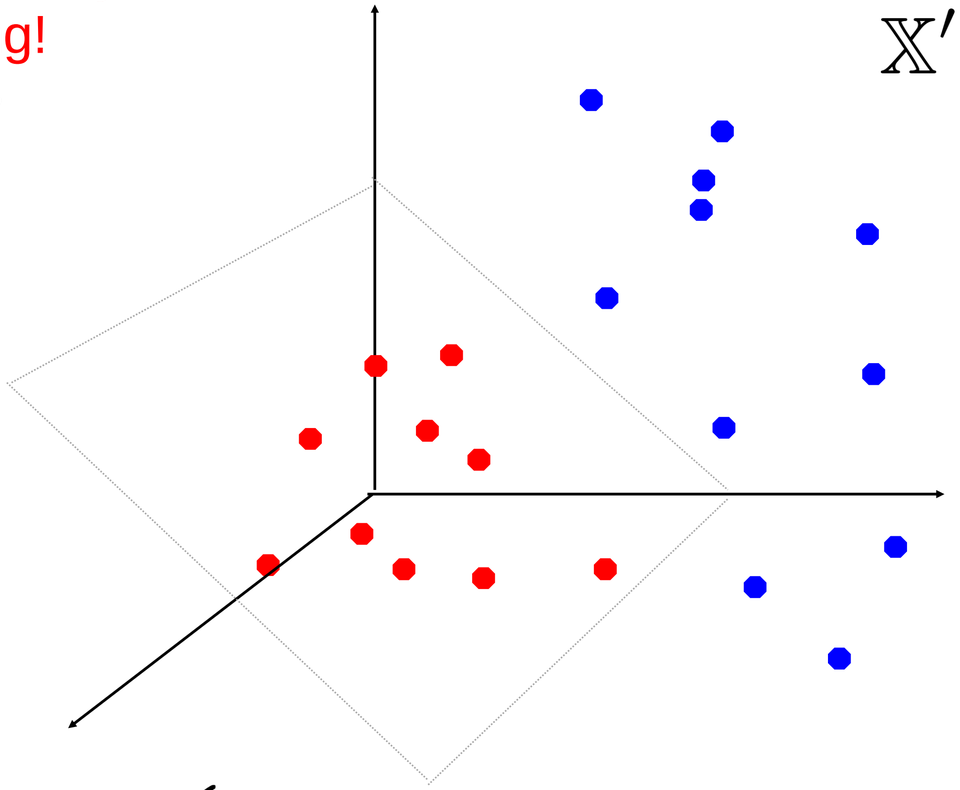
(Python, C++, and others)

Reminder: Non-linear decision boundaries

This is what the hidden layers should be doing!

$$g : \mathbb{X} \longrightarrow \mathbb{X}'$$

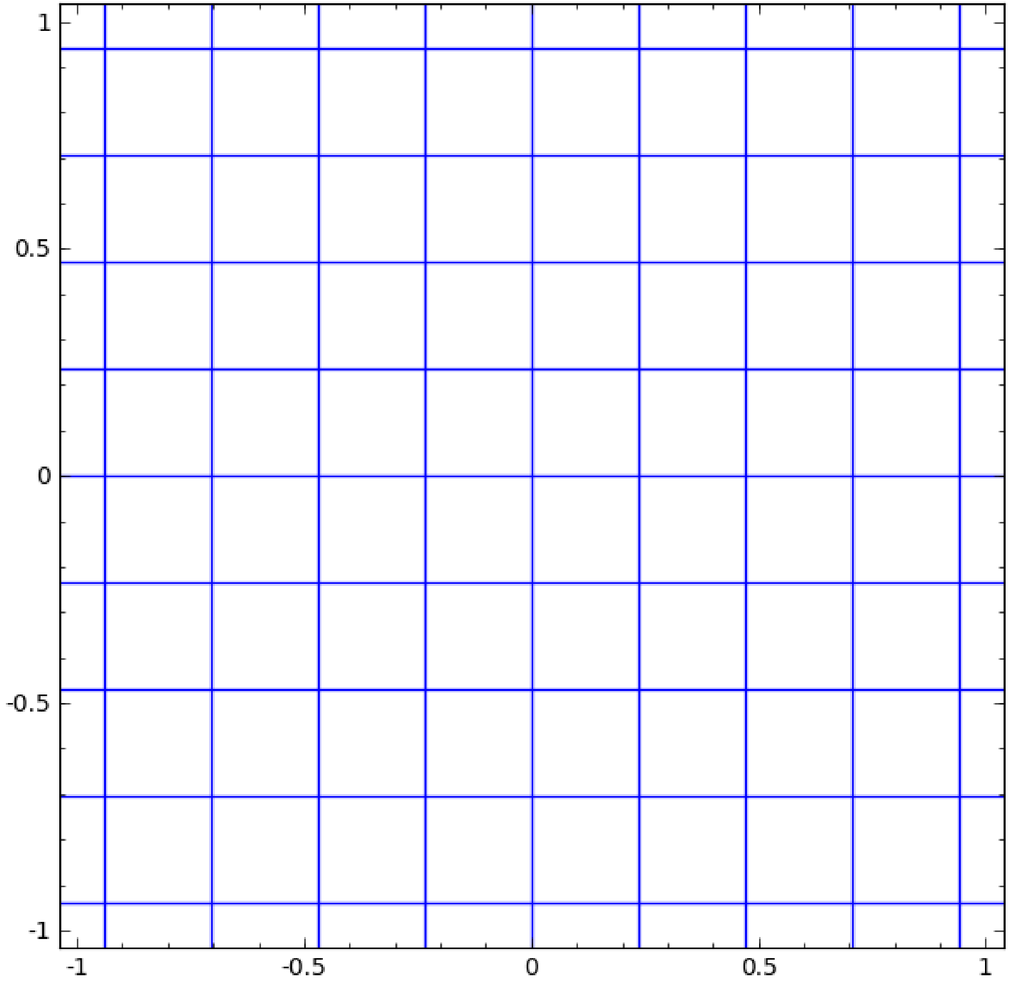




$$f_\theta(x) = \begin{cases} 1 & \text{if } w\ g(x) + b \geq 0 \\ 0 & \text{if } w\ g(x) + b < 0 \end{cases}$$

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

Evolution of isocontours as parameters change



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

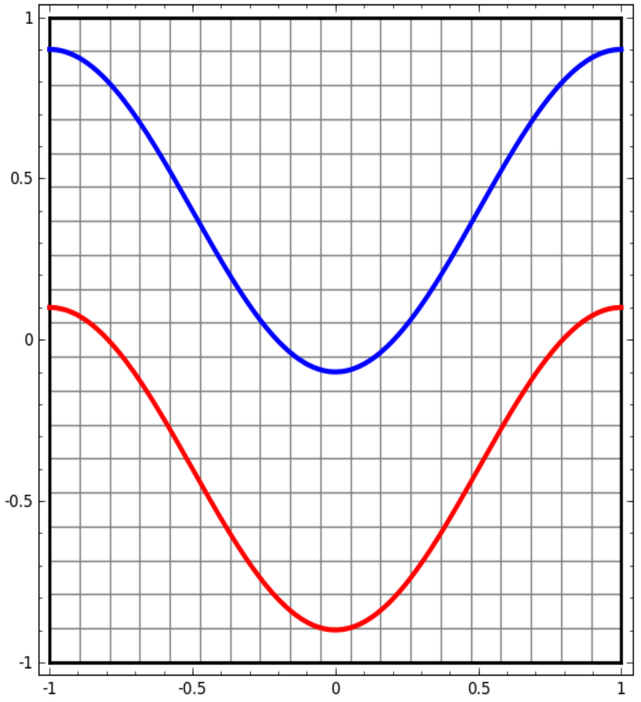
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$$

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

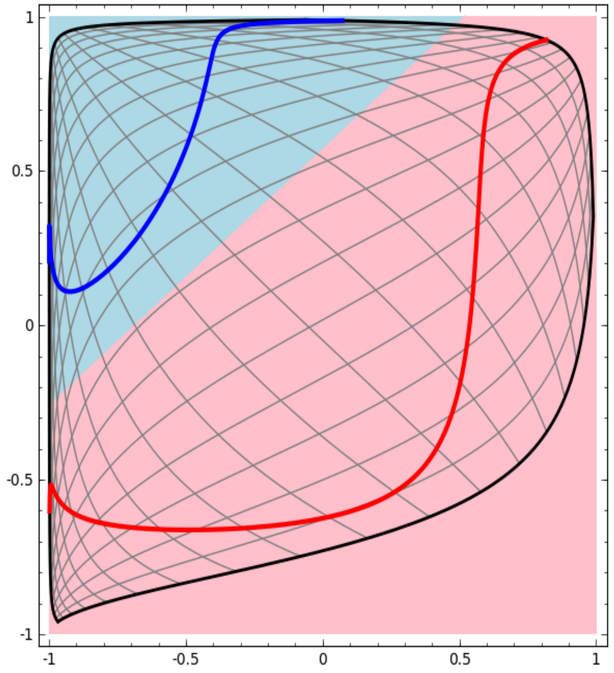
http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

From non-separable to linearly separable

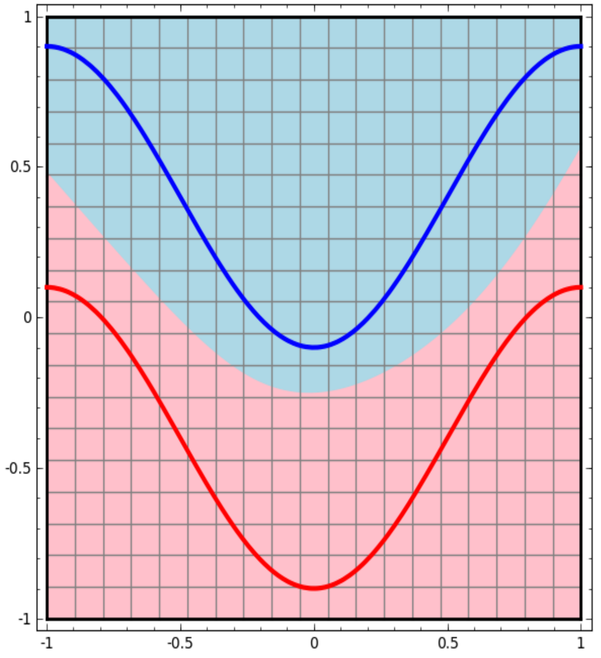
Non-linearly separable data



Data mapped to learned space

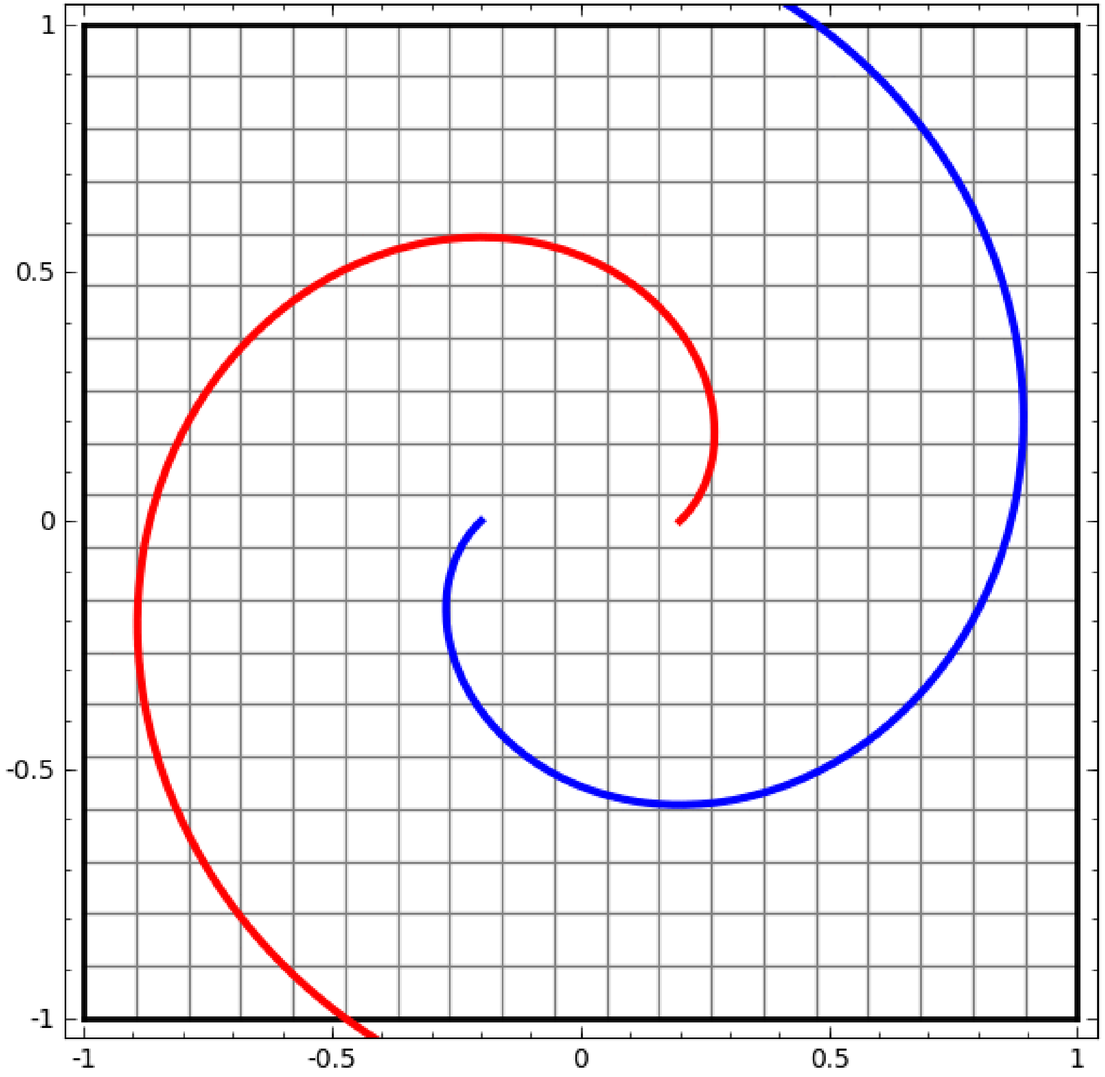


Decision function



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/

Dropout Performance

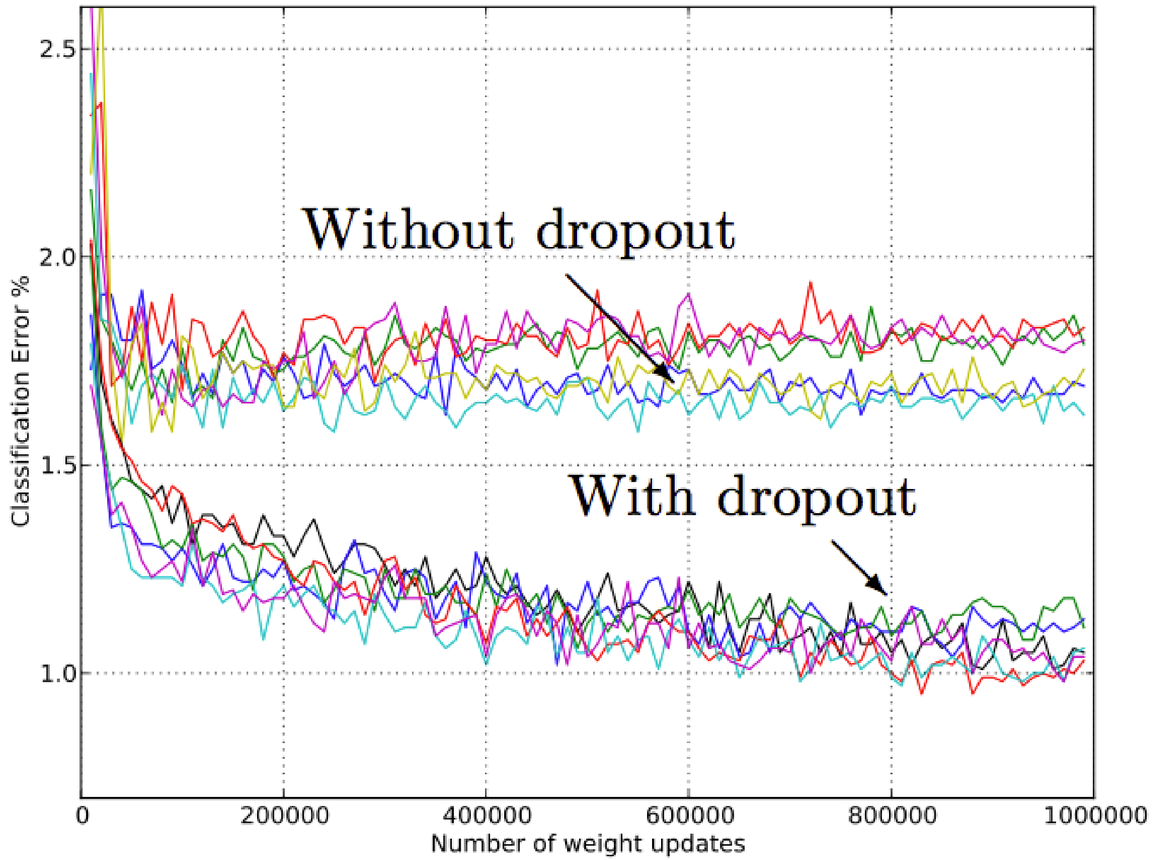


Figure 4: Test error for different architectures with and without dropout. The networks have 2 to 4 hidden layers each with 1024 to 2048 units.

Neural Network Training: Old & New Tricks

Old: (80's)

Stochastic Gradient Descent, Momentum, "weight decay"

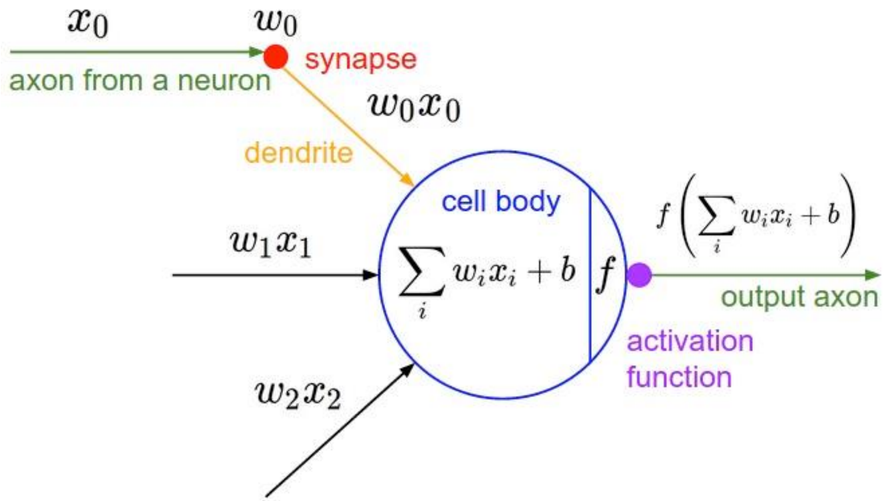
New: (last 5-6 years)

Dropout

ReLUs

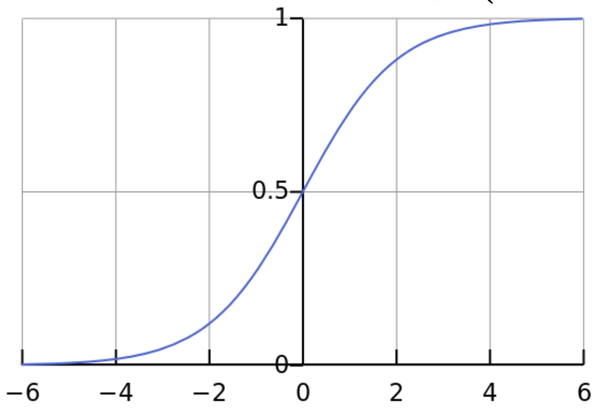
Batch Normalization

‘Neuron’: Cascade of Linear and Nonlinear Function



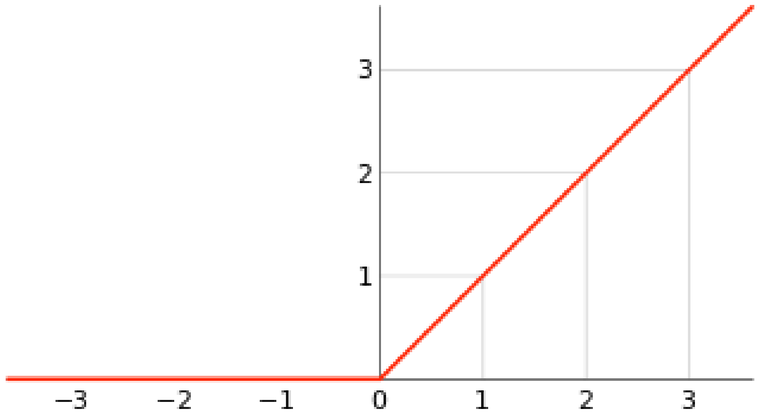
Sigmoidal (“logistic”)

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



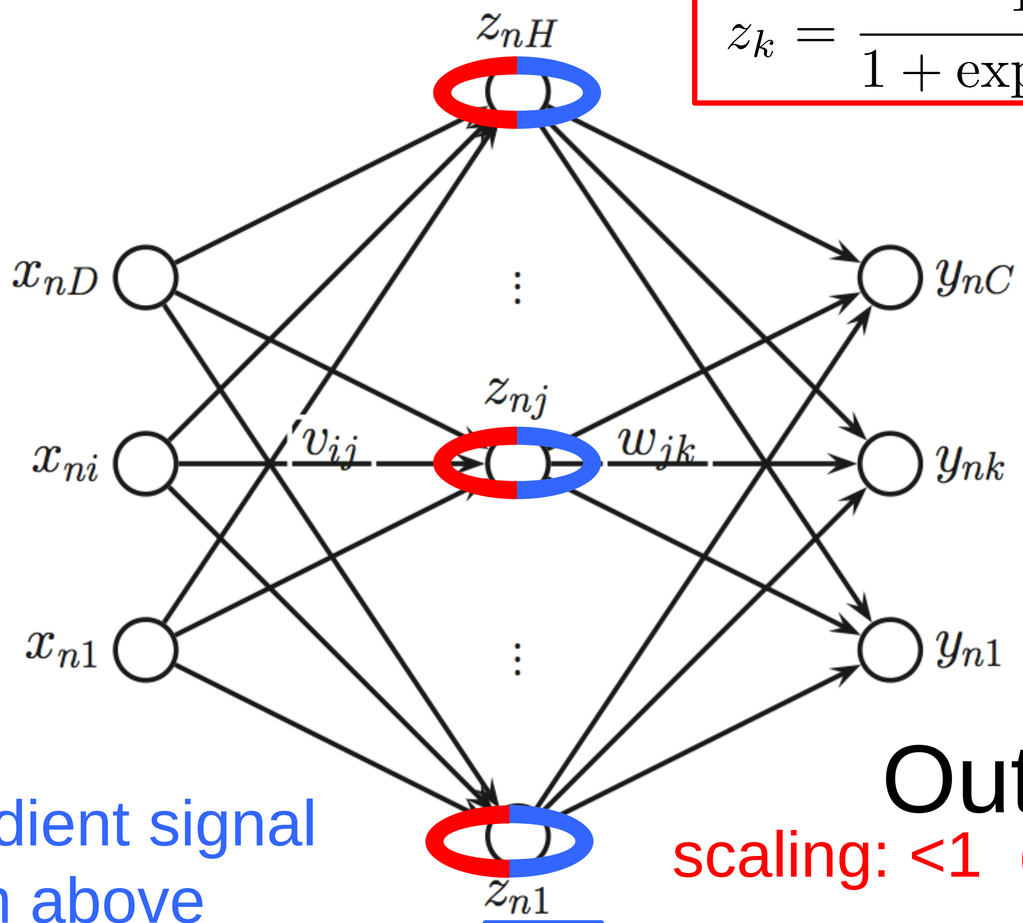
Rectified Linear Unit (REL

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



Reminder: a network in backward mode

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



$$\mathbf{y} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Outputs

Gradient signal from above

scaling: <1 (actually <0.25)

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

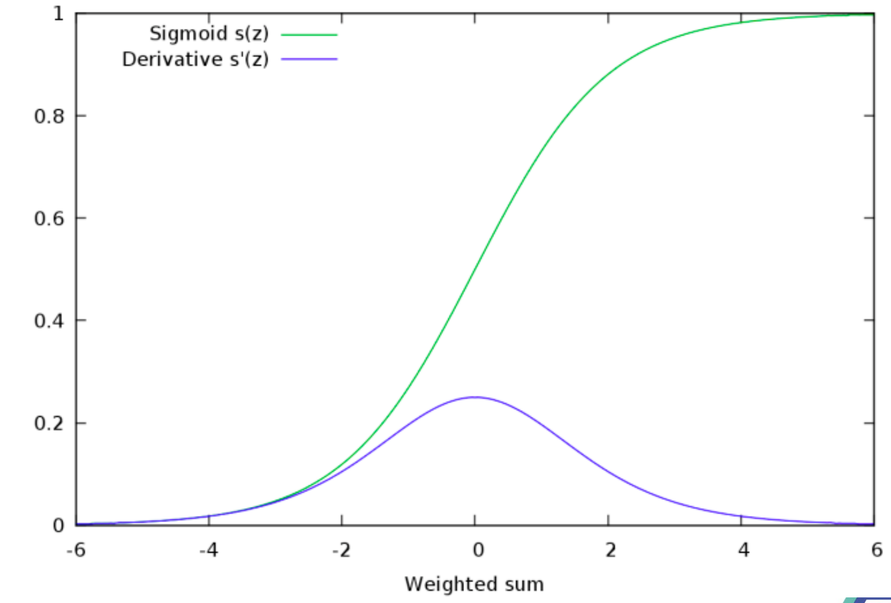
Vanishing Gradients Problem

Gradient signal from above

scaling: <1 (actually <0.25)

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

10 times: updates in the first layers get m
er knows what to do, lower layers “don’t
Sigmoidal Unit: Signal is not
getting through!



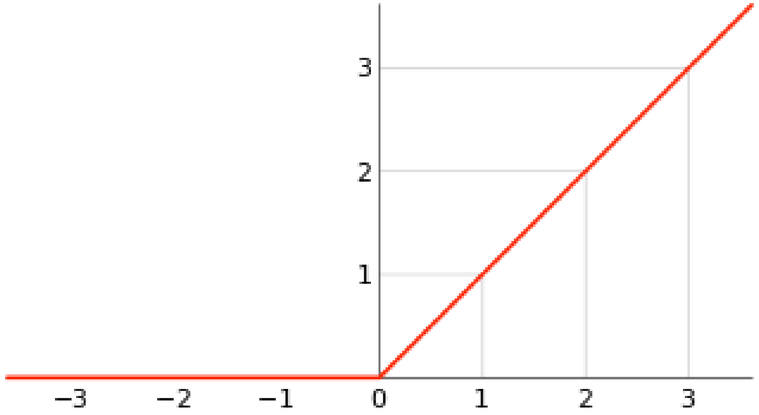
Vanishing Gradients Problem: ReLU Solves It

Gradient signal from above

Scaling: {0,1}

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

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



$$g'(a) = \begin{cases} 1 & a > 0 \\ 0 & a < 0 \end{cases}$$

Neural Network Training: Old & New Tricks

Old: (80’s)

- Stochastic Gradient Descent, Momentum, “weight decay”

New: (last 5-6 years)

- Dropout
- ReLUs
- Batch Normalization

External Covariate Shift: your input changes

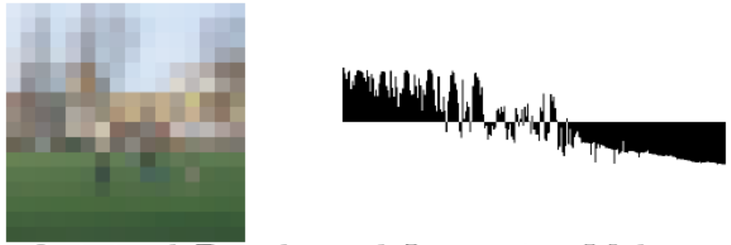
10 am

2pm

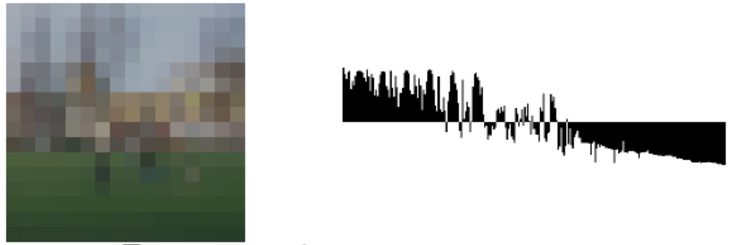
7pm

“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:

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

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

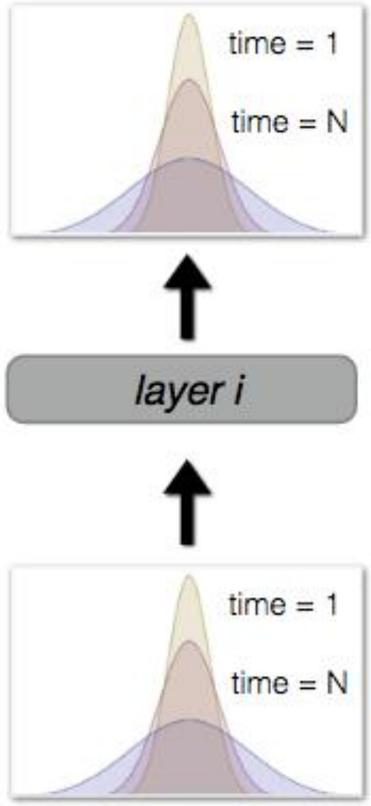
- Then make it have unit variance:

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

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

Internal Covariate Shift

Neural network activations during training: moving target



Batch Normalization

Whiten-as-you-go:

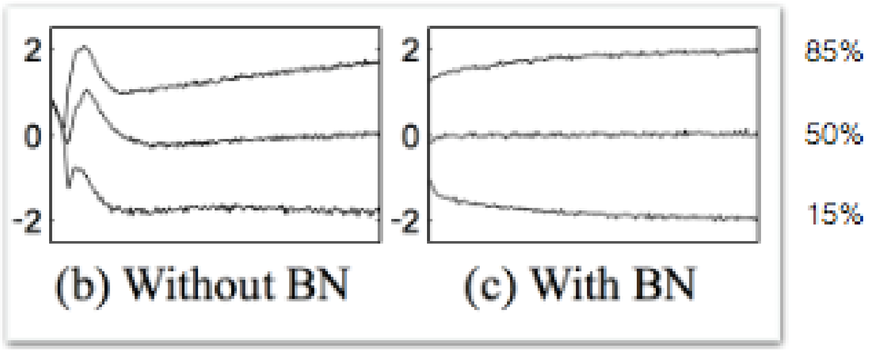
- Normalize the activations in each layer within a mini-batch.
- Learn the mean and variance (γ, β) of each layer as parameters

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

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

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

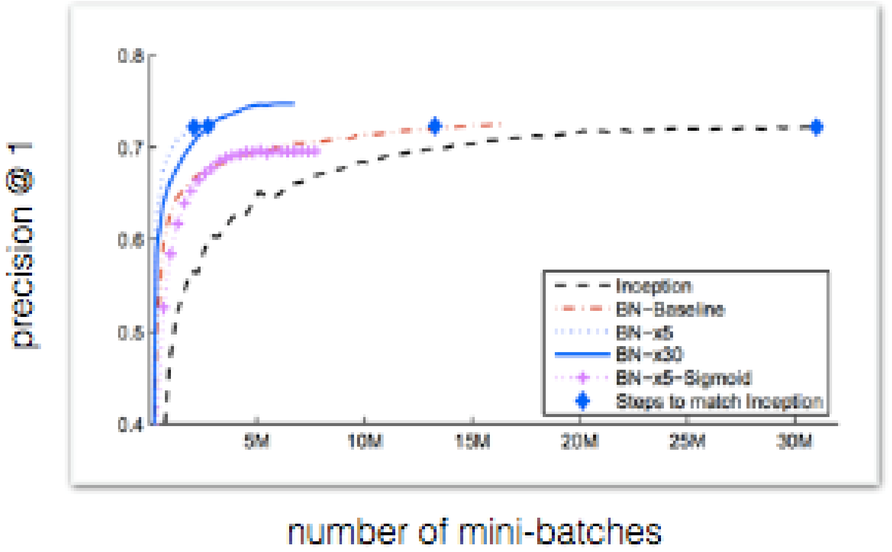
$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$



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

Batch Normalization: used in all current systems

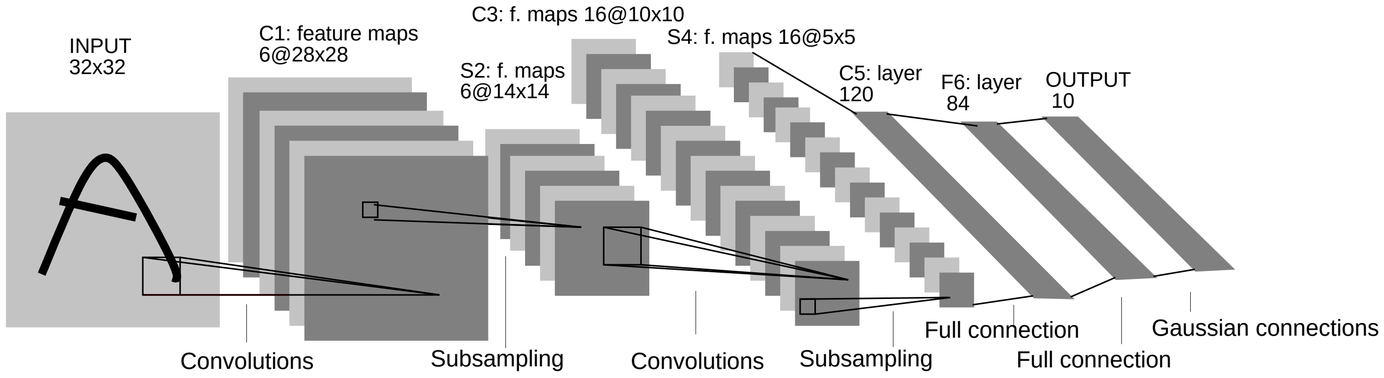
- Multi-layer CNN's train faster with fewer data samples (15x).
- Employ faster learning rates and less network regularizations.
- Achieves state of the art results on ImageNet.



Convolutional Neural Networks

Modern Architectures

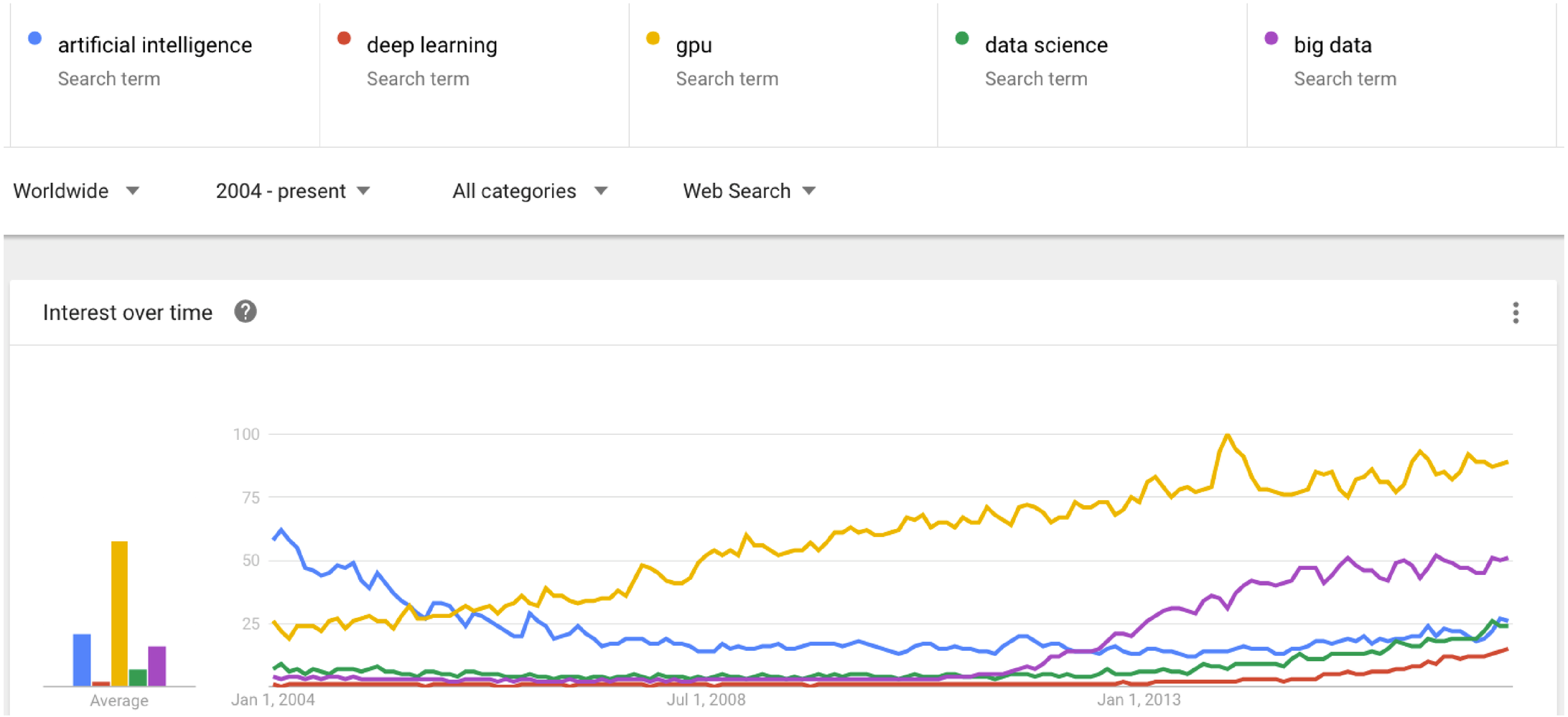
CNNs, late 1980's: LeNet



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

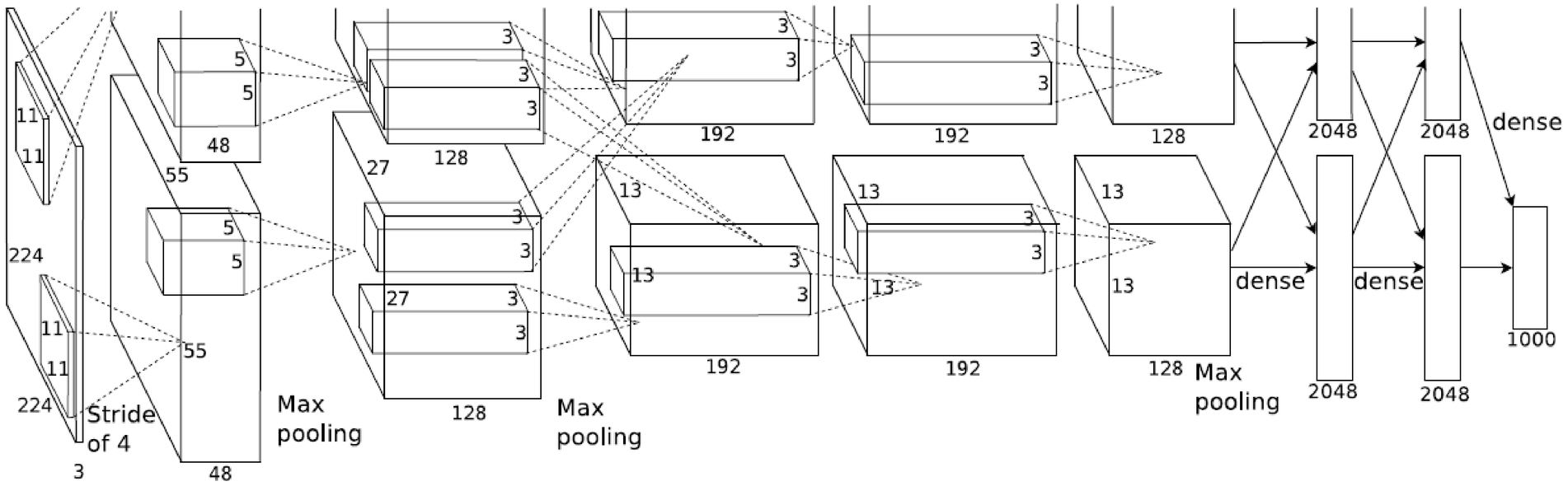
https://www.youtube.com/watch?v=FwFduRA_L6Q

What happened in between?



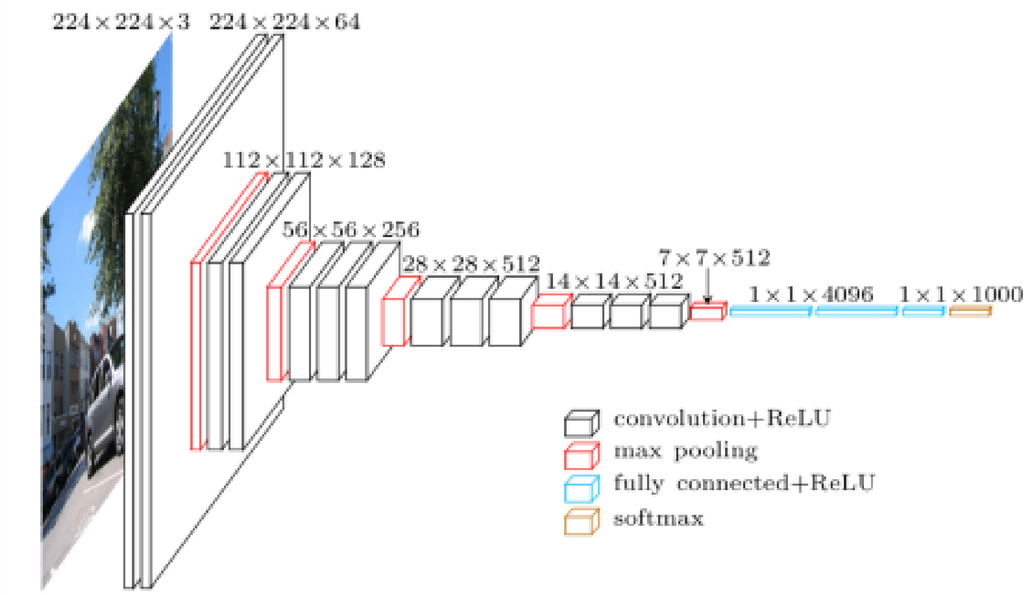
deep learning = neural networks (+ a few more ~~big data + GPUs~~ recent tricks!

CNNs, 2012



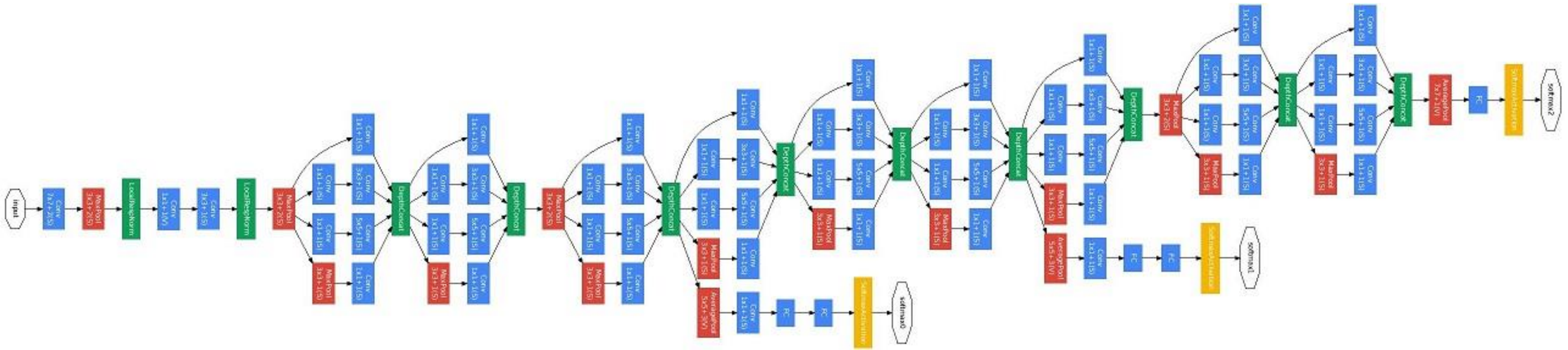
AlexNet
Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton: ImageNet classification with deep convolutional neural networks. Commun. ACM 60(6): 84-90 (2017)

CNNs, 2014: VGG



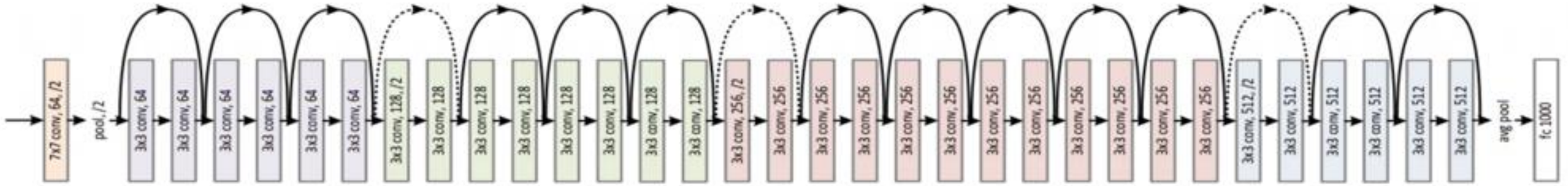
Karen Simonyan, Andrew Zisserman (=Visual Geometry Group)
Very Deep Convolutional Networks for Large-Scale Image Recognition, arxiv, 2014.

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

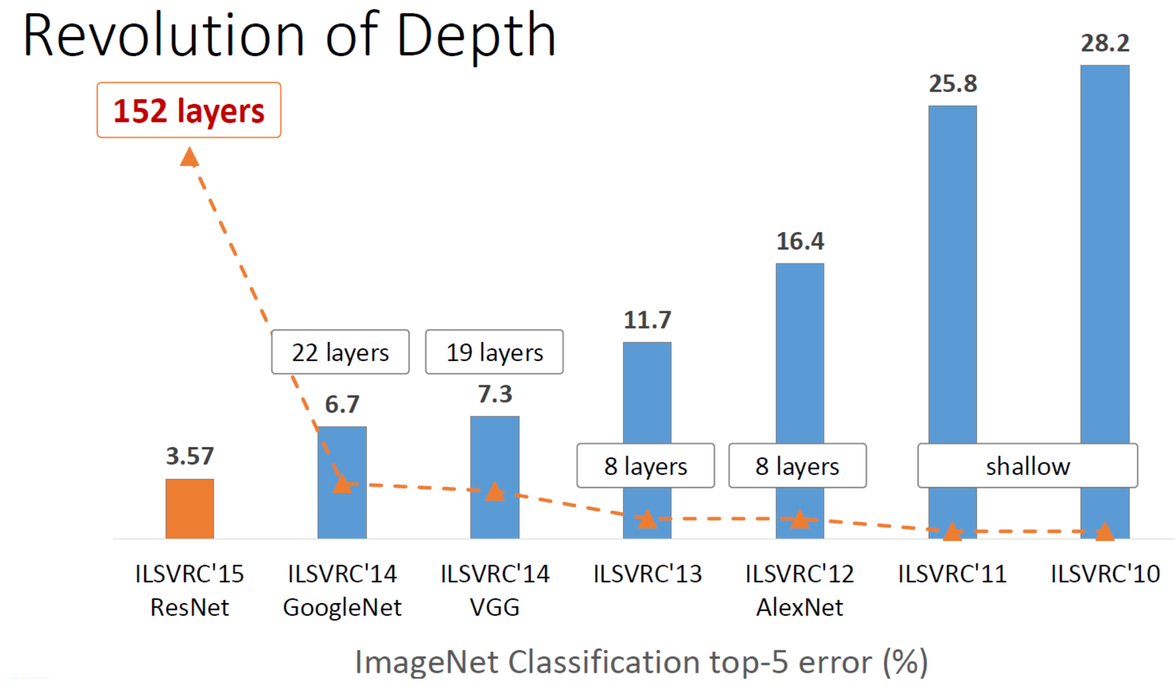
CNNs, 2015: ResNet



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

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
  - …