

# Lecture 1: Introduction to Deep Learning

# Overview



This class is about:

- deep learning
- application in computer vision and graphics
- applications in natural language processing

It will include:

- 12 lectures
- 6 seminars
- 4 homeworks
- A final project

2006

CAPTCHA



Computer vision = 60%

$$0.6^{12} = 0.00217$$

# 2014



Completed • Swag • 215 teams

## Dogs vs. Cats

Wed 25 Sep 2013 – Sat 1 Feb 2014 (8 months ago)

Dashboard ▼

### Private Leaderboard - Dogs vs. Cats

This competition has completed. This leaderboard reflects the final standings.

See someone's profile

#	Δ1w	Team Name <small>* in the money</small>	Score <small>🔍</small>	Entries	Last Submission UTC (Best - Last)
1	—	Pierre Sermanet *	0.98914	5	Sat, 01 Feb 2014 21:43:19 (-)
2	↑26	orchid *	0.98309	17	Sat, 01 Feb 2014 23:52:30
3	—	Owen	0.98171	15	Sat, 01 Feb 2014 17:04:40 (-)
4	new	Paul Covington	0.98171	3	Sat, 01 Feb 2014 23:05:20
5	↓3	Maxim Milakov	0.98137	24	Sat, 01 Feb 2014 18:20:58

$$0.989^{12} = 0.875$$

2014

# Microsoft Research

Search Microsoft Research

[Our research](#)

[Connections](#)

[Careers](#)

[About us](#)

[All](#)

[Downloads](#)

[Events](#)

[Groups](#)

[News](#)

[People](#)

[Projects](#)

[Publications](#)

## ASIRRA



After 8 years of operation, Asirra is shutting down effective October 1, 2014. Thank you to all of our users!

# How to tell a cat from a dog?

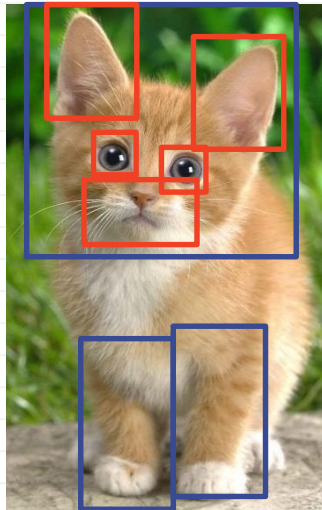
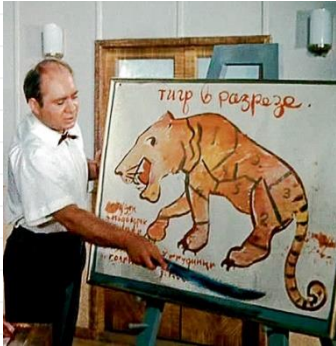


The cat or domestic cat (*Felis catus*) is a small carnivorous mammal.[1][2] It is the only domesticated species in the family Felidae.[4] The cat is either a house cat, kept as a pet; or a feral cat, freely ranging and avoiding human contact.[5] A house cat is valued by humans for companionship and for its ability to hunt rodents. About 60 cat breeds are recognized by various cat registries...



The domestic dog (*Canis lupus familiaris* when considered a subspecies of the wolf or *Canis familiaris* when considered a distinct species)[4] is a member of the genus *Canis* (canines), which forms part of the wolf-like canids,[5] and is the most widely abundant terrestrial carnivore.[6][7][8][9][10] The dog and the extant gray wolf are sister taxa[11][12][13] as modern wolves are not closely related to the wolves that were first domesticated,[12][13] which implies that the direct ancestor of the dog is extinct...

# High level vision is part-based



# How to tell a cat from a dog: ML approach

- Does it have pointed ears?
- Does it have floppy ears?
- Does it have curvy tail?
- Does it have elongated muzzle?
- Does it have legs longer than 25cm?
- ....



➡ *ML classifier*

Easier questions, but still very hard  
*(feature engineering needed!)*



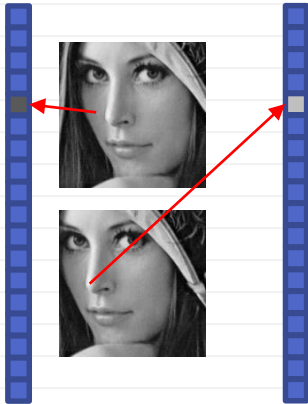
# Face detection challenge



?



$x$

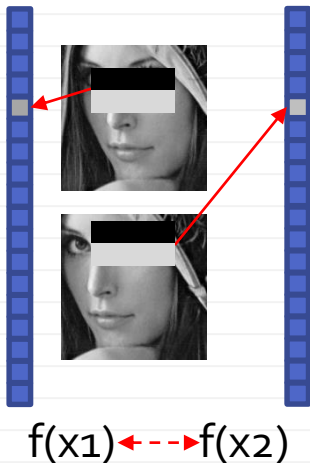


Natural feature mapping:

- Highly non-smooth w.r.t. jitter
- Require lots of training samples

$f(x_1)$    $f(x_2)$

# Haar features



Viola-Jones features:

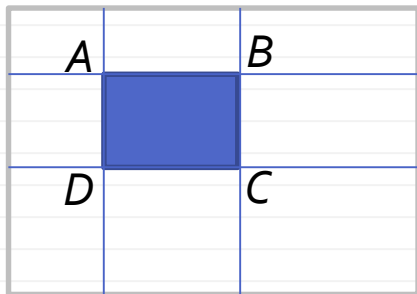
- Smoother w.r.t. jitter
- Less training examples needed
- *(also fast to compute)*

[Viola Jones, CVPR'01]

# Haar features



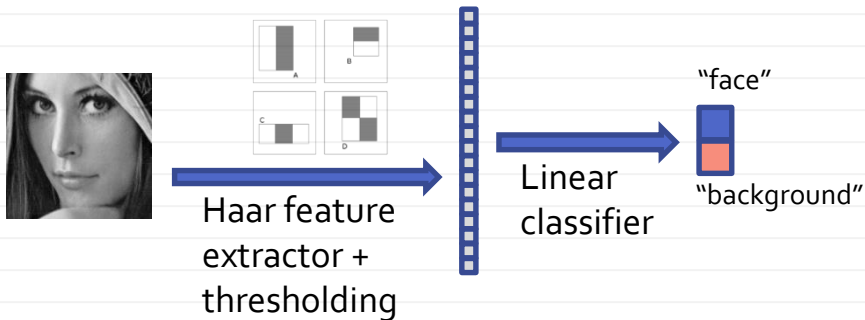
$$F(A) = \iint_{\subseteq A} f(x, y) dx dy$$



$$\iint_{ABCD} f(x, y) dx dy = F(C) + F(A) - F(B) - F(D)$$

[Viola Jones, CVPR'01]

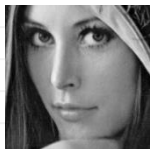
# Viola-Jones detector



- **Non-shallow**, learnable representation (AdaBoost greedy algorithm)
- Cascaded detector for speed
- One of the most impactful papers in CV history

[Viola Jones, CVPR'01]

# From face detection to pedestrian detection



VS.

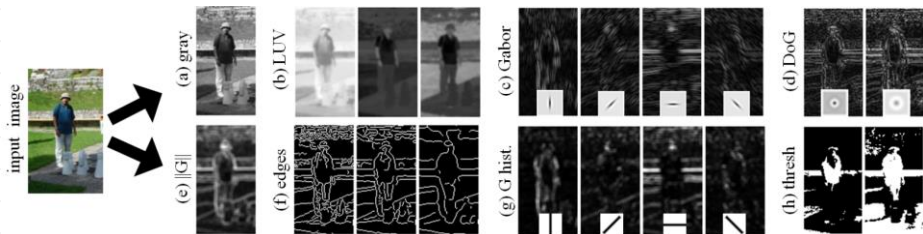


Good industry-grade performance by  
Viola-Jones (for frontal faces)



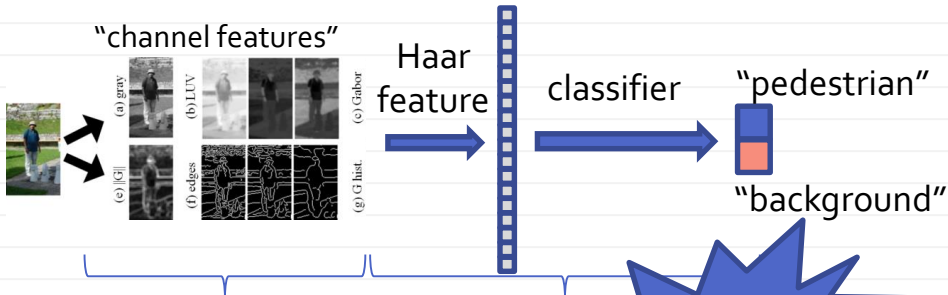
Viola-Jones detector not good enough

# Improving pedestrian detection



[Dollar et al. BMVCog]

# Improved pedestrian detector

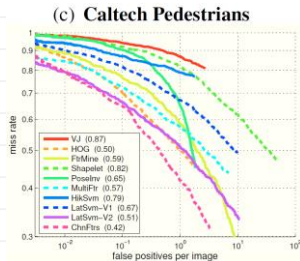


Hand-crafted

Trained using

Deep

[Dollar, Tu, Perona, Belongie. *Integral Channel Features*. BMVC09]

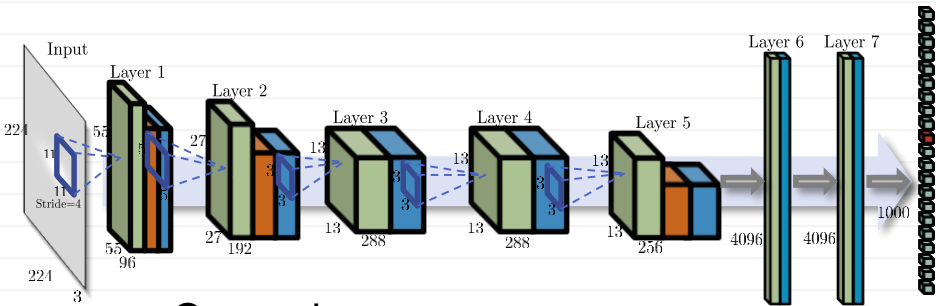


# Then, what is “deep learning”?

- Previous CV systems were “deep”, they used multiple layers of representation with success
- The main “novelty” in modern age *deep learning*: **end-to-end joint learning** of multiple (10+) layers



# The winner: convolutional networks



Operations:

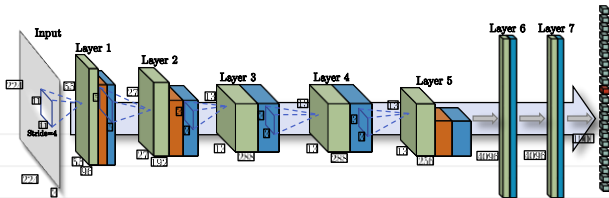
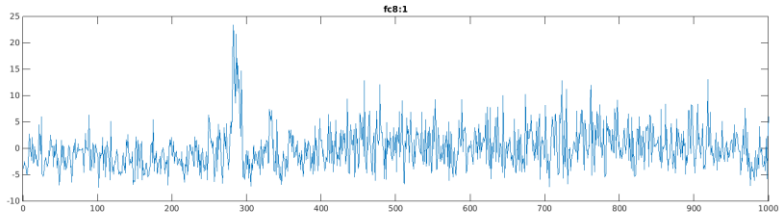
generalized convolutions

pooling (image resizing)

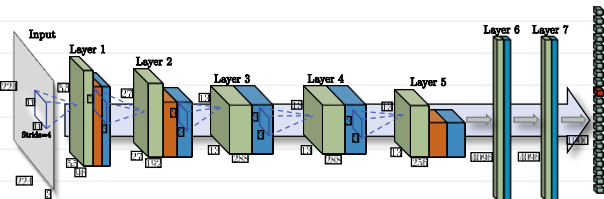
elementwise non-linearity

matrix multiplication

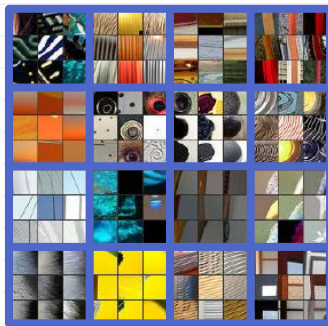
# Representations



# Left-to-right = "smarter"



Layer 1



Layer 2



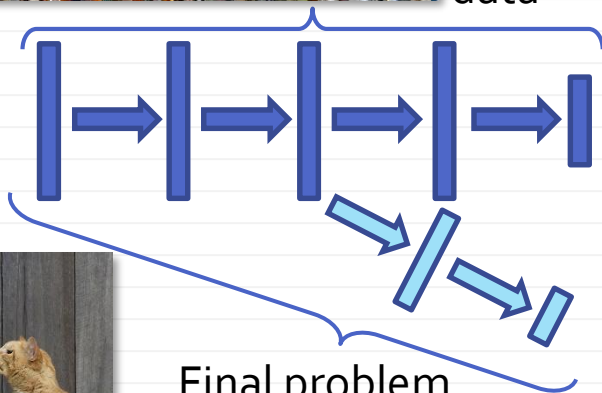
Layer 5

[Zeiler Fergus 14]

# Transfer learning



Task where we  
have a lot of  
data



# Learning intermediate representations

- The essence of modern “deep learning”
- Is essential for intelligence
- Can be done via supervised, unsupervised and other types of learning
- Has been done all along before “deep learning” revolution

# Gazoob world

“tufa”



“tufa”

“tufa”

[Tenenbaum et al. Science 2011]

# Then, what is “deep learning”?

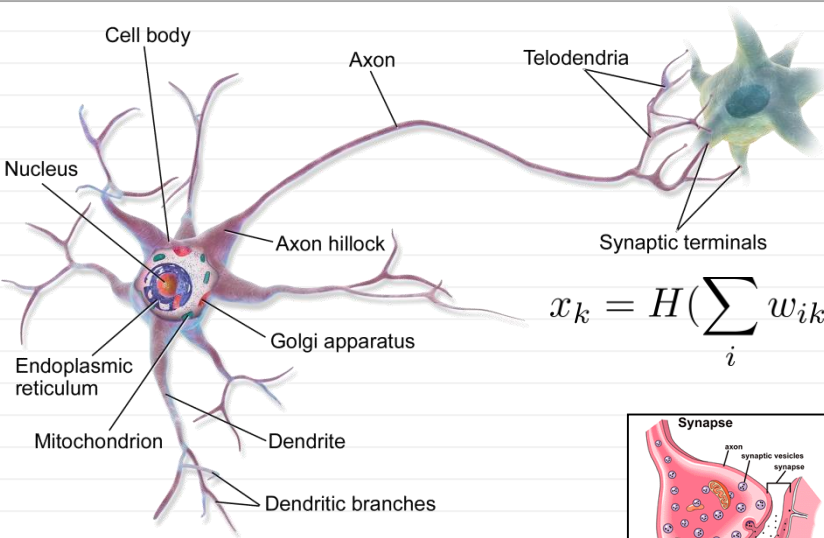
## End-to-end joint learning of all layers:

- multiple assemblable blocks
- each block is piecewise-differentiable
- gradient-based optimization
- gradients computed by *backpropagation*

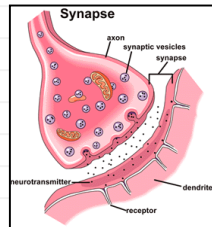
Deep learning “revolution”  
(2012? – now): rapid  
engineering improvements  
following these principles



# Neuron model

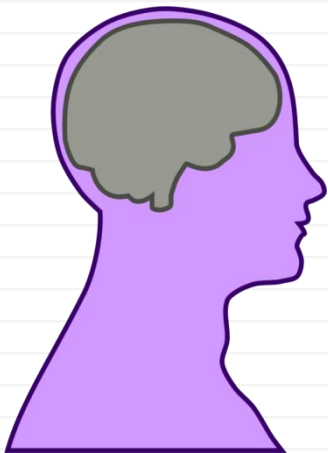


$$x_k = H\left(\sum_i w_{ik} x_i - \tau\right)$$





# Brain statistics



Human brain:

- 100 billion neurons
- average neuron is connected to 1000-10000 other neurons
- 100 trillion synapses
- 10-25% is in visual cortex

# Perceptron

[Rosenblatt 1957]: an “artificial neuron”

$$y = H(w^T x)$$

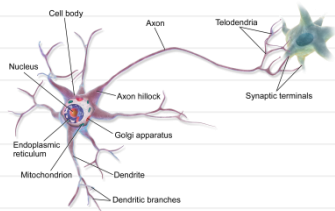
**loop** over examples

$$y = H(w^T x_i);$$

$$w = w + 1/2 x_i * (y_i - y);$$

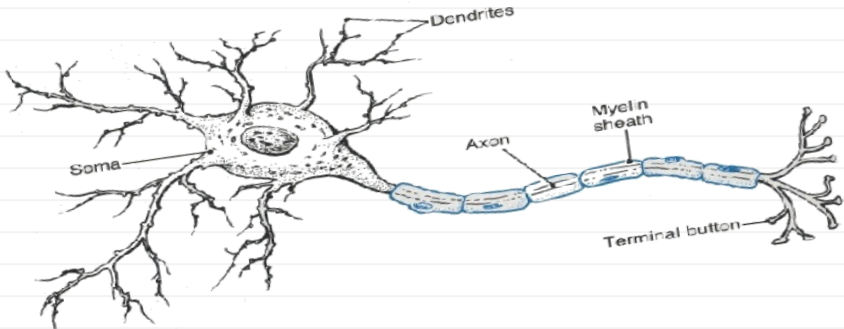
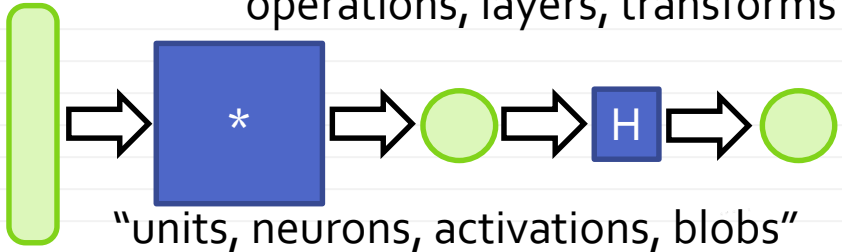
**end**

Converges to linear separator of the training data if it exists.



# Terminology and graphical language

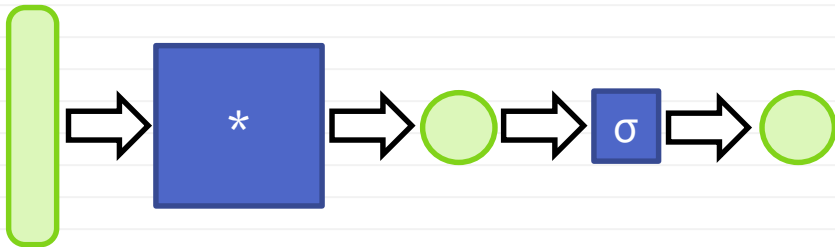
“operations, layers, transforms”



# Logistic regression

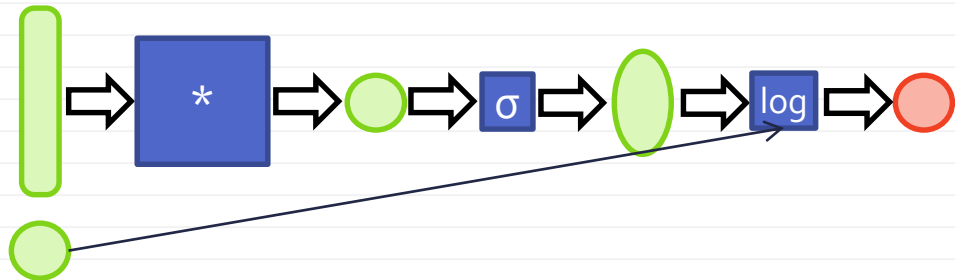
$$P(y(x) = y_i | w) = \frac{1}{1 + e^{-y_i w^T x_i}} = \sigma(y_i w^T x_i)$$

Same diagram/network:



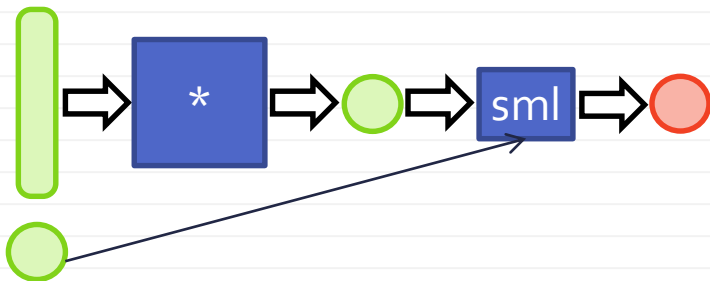
# Training logistic regression

$$E(w) = - \sum_{i=1}^N \log P(y(x) = y_i | w) = \sum_{i=1}^N \log(1 + e^{-y_i w^T x_i})$$



# Logistic regression: simplifying training

$$E(w) = - \sum_{i=1}^N \log P(y(x) = y_i | w) = \sum_{i=1}^N \log(1 + e^{-y_i w^T x_i})$$



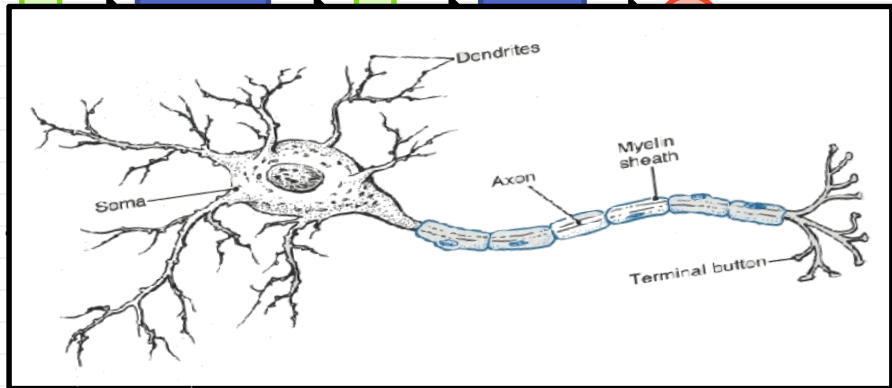
Softmax loss = log loss over softmax/logistic

# Multinomial logistic regression

Training:

$$w_i^T x$$

"Loss"



# Biological neuron layers

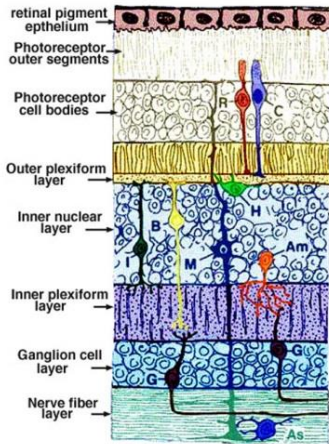


Fig. 5. Scheme of the layers of the developing retina around 5 months' gestation (Modified from Odgen, 1989).

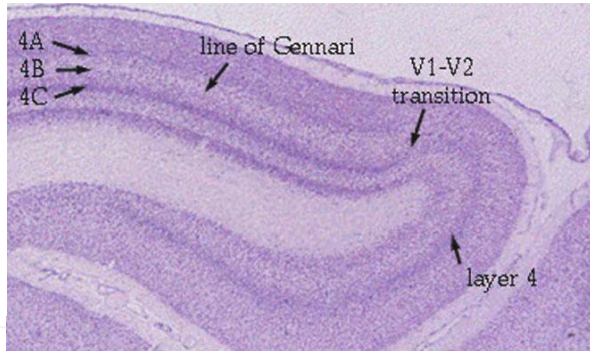
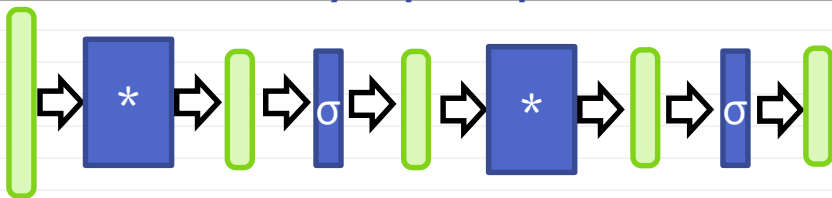


Figure 9. Nissl stained section of the visual cortex to show the border between area 17 (V1) and area 18 (V2).

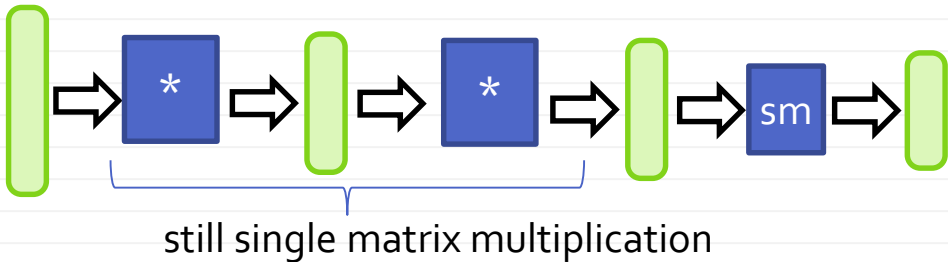


# Multi-layer perceptron idea

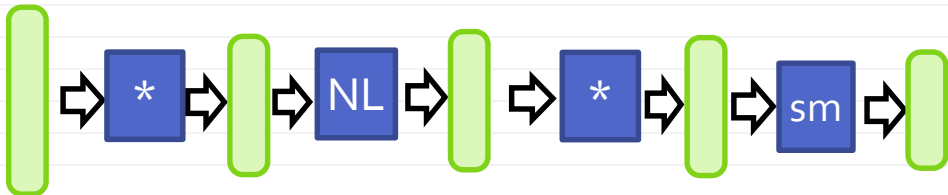


- First layer: parallel logistic regression
- Each predicts presence of some feature in the input
- Second layer is a logistic regression that “weighs” the input of the first layer

# Artificial multilayer networks

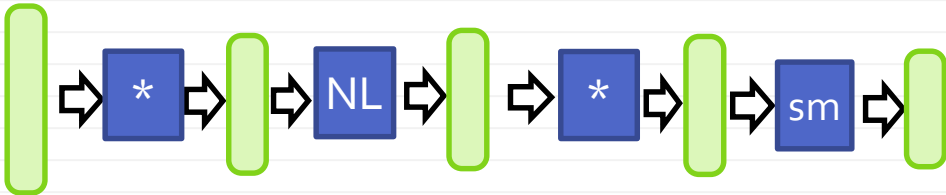


To get more powerful model need non-linearity:



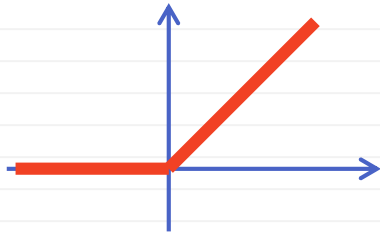
# Adding non-linearities

To get more powerful model need non-linearity:



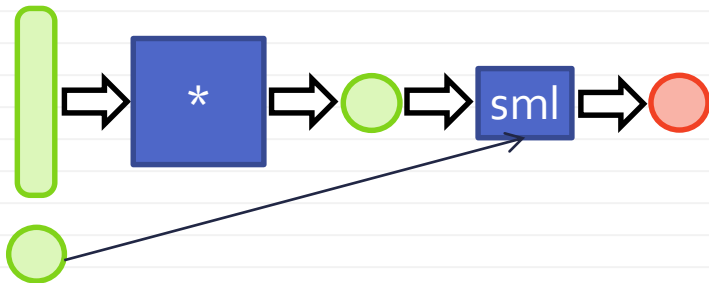
Possible *elementwise* non-linearities:

- Heaviside
- Sigmoid(logistic)/tanh
- More recently:  
 $ReLU(x) = \max(0, x)$



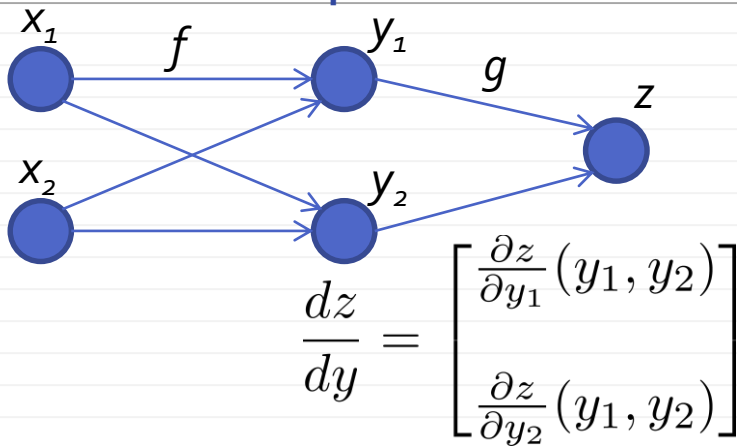
# Training logistic regression

$$E(w) = - \sum_{i=1}^N \log P(y(x) = y_i | w) = \sum_{i=1}^N \log(1 + e^{-y_i w^T x_i})$$



$$\frac{dE}{dw} \Big|_{x_i} = (\sigma(y_i w^T x_i) - 1) y_i x_i$$

## Recap: chainrule



$$\frac{\partial z}{\partial x_1} = \frac{\partial z}{\partial y_1} \cdot \frac{\partial y_1}{\partial x_1} + \frac{\partial z}{\partial y_2} \cdot \frac{\partial y_2}{\partial x_1}$$

## Recap: chainrule

$$\frac{\partial z}{\partial x_1} = \frac{\partial z}{\partial y_1} \cdot \frac{\partial y_1}{\partial x_1} + \frac{\partial z}{\partial y_2} \cdot \frac{\partial y_2}{\partial x_1}$$

$$\frac{\partial z}{\partial x_2} = \frac{\partial z}{\partial y_1} \cdot \frac{\partial y_1}{\partial x_2} + \frac{\partial z}{\partial y_2} \cdot \frac{\partial y_2}{\partial x_2}$$

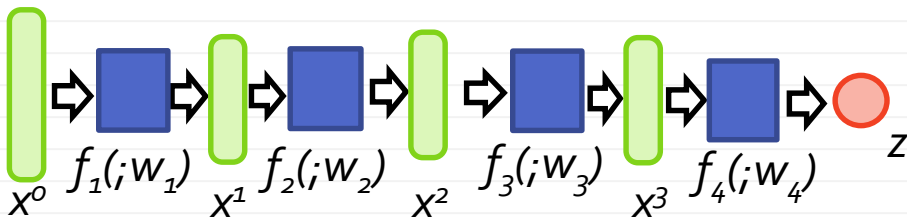
$$\frac{dz}{dx} = \begin{bmatrix} \frac{\partial z}{\partial x_1} \\ \frac{\partial z}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\partial z}{\partial y_1} \\ \frac{\partial z}{\partial y_2} \end{bmatrix}$$

## Recap: chainrule

$$\frac{dz}{dx} = \begin{bmatrix} \frac{\partial z}{\partial x_1} \\ \frac{\partial z}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\partial z}{\partial y_1} \\ \frac{\partial z}{\partial y_2} \end{bmatrix}$$

$$\frac{dz}{dx} = \left( \frac{dy}{dx} \right)^T \frac{dz}{dy}$$

# Computing deeper derivatives



$$z = f_4(f_3(f_2(f_1(x; w_1); w_2); w_3); w_4)$$



# Sequential computation: *backpropagation*

$\frac{dz}{dx^3}, \frac{dz}{dw_4}$  can be computed

$$\frac{dz}{dw_3} = \frac{dx^3}{dw_3}^T \cdot \frac{dz}{dx^3}$$

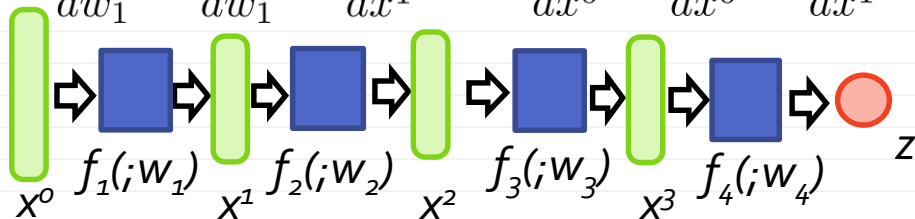
$$\frac{dz}{dx^2} = \frac{dx^3}{dx^2}^T \cdot \frac{dz}{dx^3}$$

$$\frac{dz}{dw_2} = \frac{dx^2}{dw_2}^T \cdot \frac{dz}{dx^2}$$

$$\frac{dz}{dx^1} = \frac{dx^2}{dx^1}^T \cdot \frac{dz}{dx^2}$$

$$\frac{dz}{dw_1} = \frac{dx^1}{dw_1}^T \cdot \frac{dz}{dx^1}$$

$$\frac{dz}{dx^0} = \frac{dx^1}{dx^0}^T \cdot \frac{dz}{dx^1}$$



# Layer abstraction



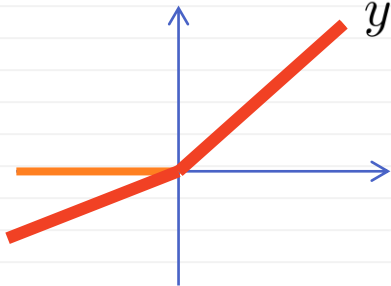
Each layer is defined by:

- forward performance:  $y = f(x; w)$
- backward performance:

$$z(x) = z(f(x; w))$$

$$\frac{dz}{dx} = \frac{dy}{dx}^T \cdot \frac{dz}{dy} \qquad \frac{dz}{dw} = \frac{dy}{dw}^T \cdot \frac{dz}{dy}$$

# Example: “leaky ReLu”



$$y = f(x; \alpha) = \max(x, \alpha x)$$

$$\frac{dz}{dx} =$$

$$\frac{\partial z}{\partial \alpha} =$$



Cornell University  
Library

[arXiv.org](https://arxiv.org) > [cs](#) > [arXiv:1502.01852](#)

Computer Science > Computer Vision and Pattern Recognition

**Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification**

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun

(Submitted on 6 Feb 2015)

# OOP pseudocode of deep learning

```
abstract class Layer {  
    params w,dzdw;  
    virtual y = forward(x);  
    virtual dzdx = backward(dzdy,x,y);  
    // should compute dzdw as well  
  
    void update (tau) {  
        w = w+tau*dzdw;  
    }  
};
```

Efficient implementations have to use vector/matrix instructions and work efficiently for minibatches!

# Computing the partial derivatives

$$z(x) = z(f(x; w))$$

$$\frac{dz}{dx} = \frac{dy^T}{dx} \cdot \frac{dz}{dy}$$

$$\frac{dz}{dw} = \frac{dy^T}{dw} \cdot \frac{dz}{dy}$$

Options for partial derivatives:

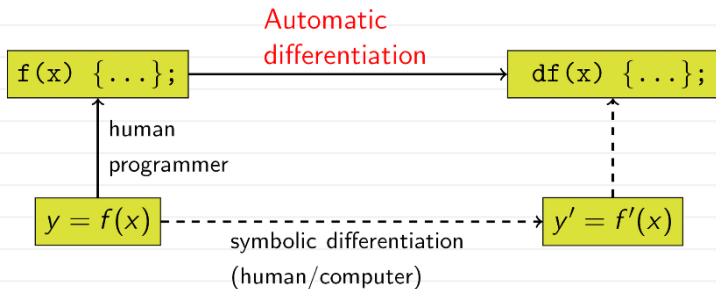
- Finite differences
- Derive gradients analytically

Debugging is hard

Gradient checking is a good idea!

# Deep learning/symbolic comp. packages

- All packages facilitate stacking layers and defining new layers
- Differ on languages/levels of granularity
- Some allow **symbolic differentiation**
- Some allow **automatic differentiation**



# Recap

Deep learning:

- Define each layer
- Assemble layers into an architecture
- Loop over minibatches
- For each minibatch find the stochastic gradient and update the parameters (use momentum, etc.)

# Zoo of layers

Multiplicative layer  
Convolutional layer

Copy layer  
Split layer  
Cat layer  
Merge layer

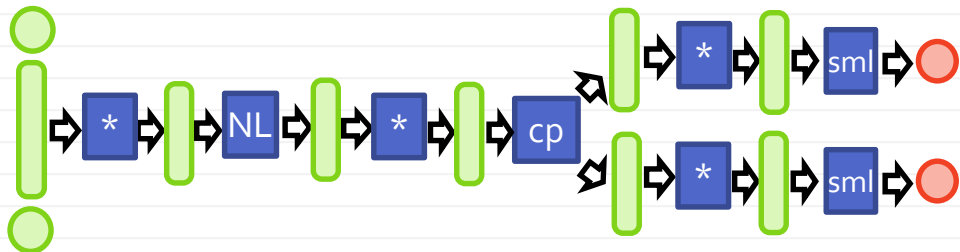
ReLu layer  
Sigmoid layer  
Softmax layer  
Normalization layer  
Max-pooling layer

Data providers

Log-loss layer  
Softmax loss layer  
Hinge loss layer  
L2-loss layer  
Contrastive loss layer



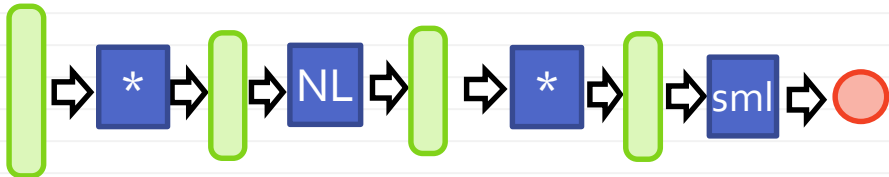
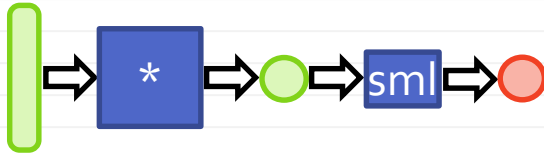
## Example: multitask learning



Typical usecase:

- Two related tasks
- Limited labeled data for the main task
- Lots of labeled data for auxiliary task

## Back to regularization



- Overfitting is severe for deep models (why?)
- The progress on deep learning was “delayed” till huge amount of data

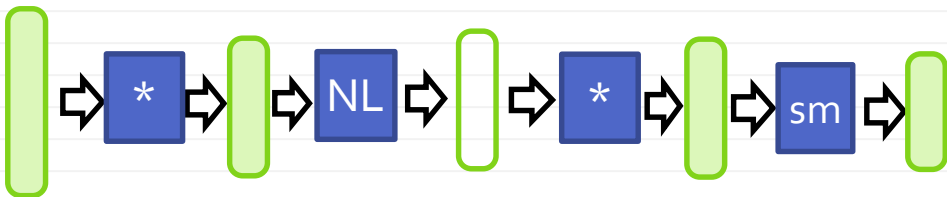
## Recap: regularization

Strategies to avoid overfitting (aka *regularize learning*):

- Pick a “simpler” model (e.g. conv nets)
- Stop optimization early (always keep checking validation loss/error)
- Impose smoothness (weight decay)
- Inject noise (equivalent to smoothness)
- Bag (average) multiple models

# Dropout regularization

Regularization with a special type of noise:



- At training time, define which units are active at random (*mask*) and which ones are dropped. Divide active unit values by the drop-out probability

[Srivastava et al. 2011]

# How to implement dropout

## Define it as a layer!

Forward propagation (train-time only):

$$n \sim \text{Bernouli}(p) \qquad y = \frac{1}{p} x \odot n$$

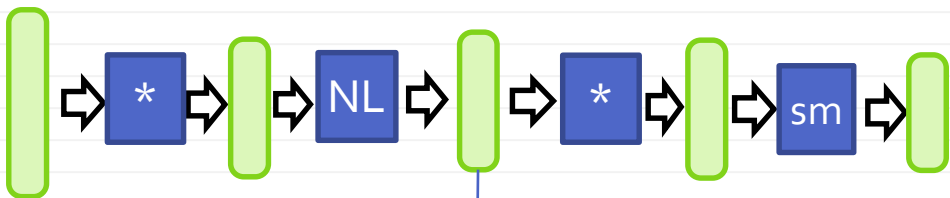
Backward propagation:

$$\frac{dz}{dx} = \frac{1}{p} \frac{dz}{dy} \odot n$$

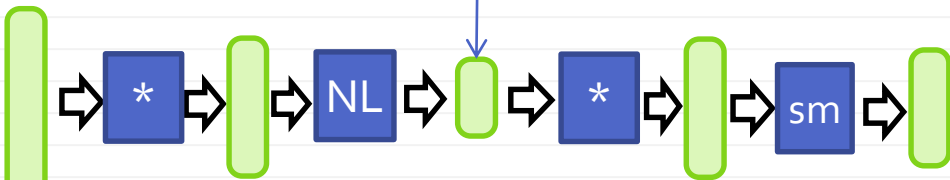
This layer is switched off during test

# Dropout idea: ensemble interpretation

Pseudo-ensemble training:



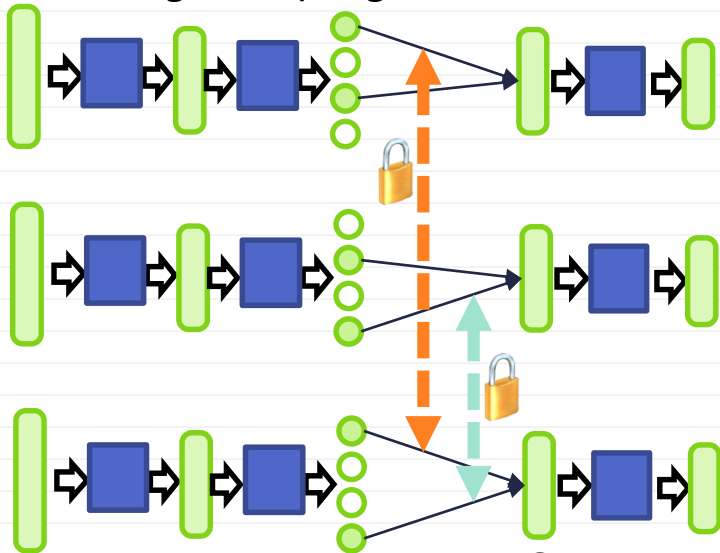
A derived model: *subsampling*



Goal: “train” an exponential number of such reduced models [Srivastava et al. 2011]

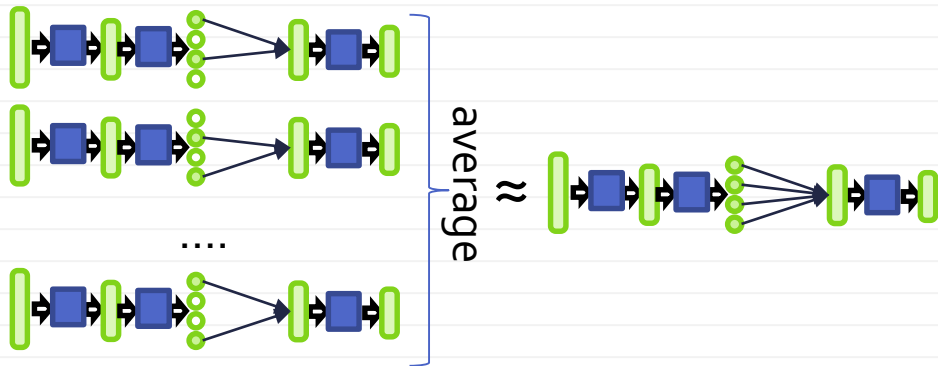
# Dropout idea: ensemble interpretation

Training a very big ensemble of models:



[Srivastava et al. 2011]

# Dropout idea: ensemble interpretation



- Approximation is not exact
- ...but works well in practice

[Srivastava et al. 2011]



# Deep learning: recap

## End-to-end joint learning of all layers:

- multiple assembleable blocks
- each block is piecewise-differentiable
- gradient-based optimization
- gradients computed by backpropagation

