Artificial Neural Networks: Basis

人工智能引论 第10课

主讲人：刘家瑛

2019年4月8日

Slides Credit

- *Slides modified from Geoffrey Hinton and Feifei Li.*
- http://www.cs.toronto.edu/~hinton/coursera_slides.html
- http://cs231n.stanford.edu/

Research Group

STRUCT

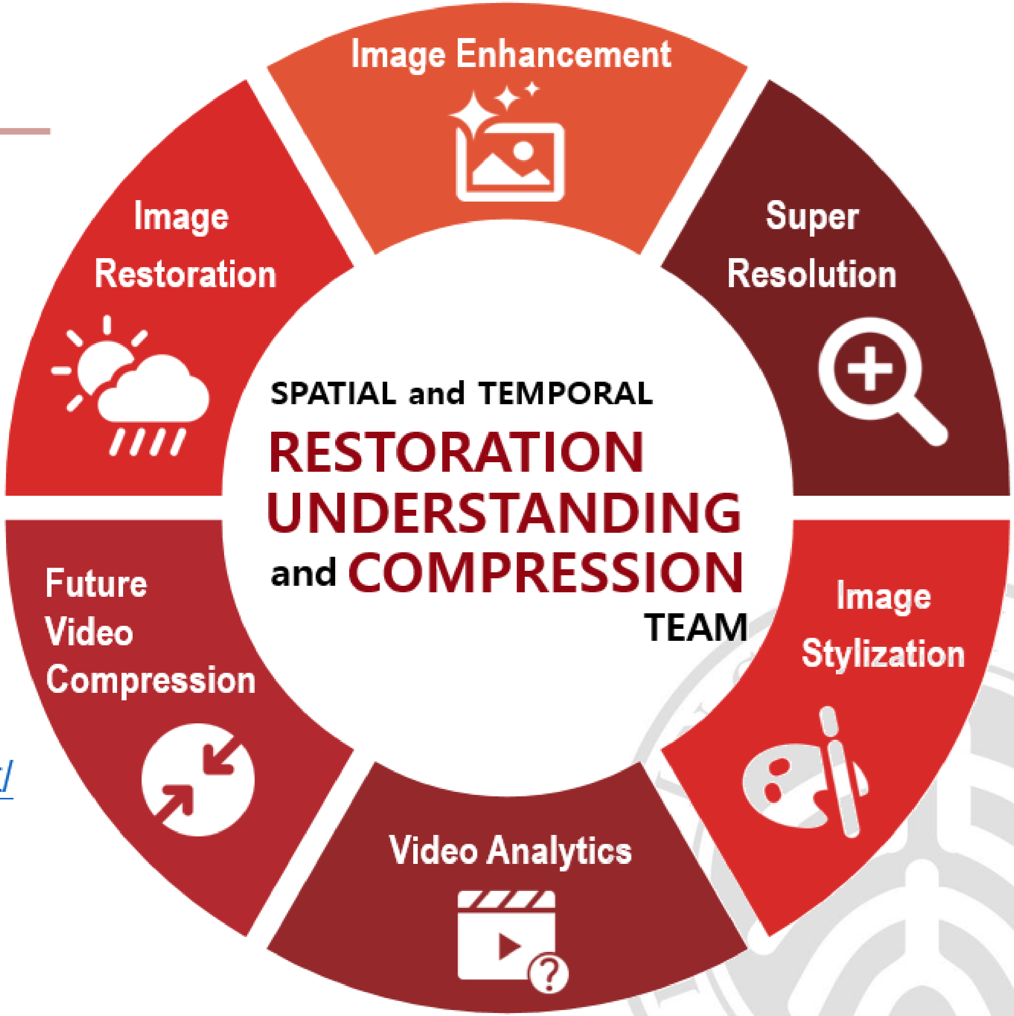
智能影像计算

北京大学 计算机科学技术研究所

视频信息处理研究组

Spatial and Temporal **Restoration**,
Understanding and **Compression** Team

- PI: 刘家瑛
- 邮箱: liujiaying@pku.edu.cn
- 网页: http://www.icst.pku.edu.cn/struct/



Outline

- **General ANN**
- CNN Network
- RNN Network and Gated Network
- Beyond CNN and RNN

TECHNOLOGY

The New York Times

SUBSCRIBE NOW

LOG IN

Three Pioneers in Artificial Intelligence Win Turing Award

From left, Yann LeCun, Geoffrey Hinton and Yoshua Bengio. The researchers worked on key developments for

Source: https://www.nytimes.com/2019/03/27/technology/turing-award-hinton-lecun-bengio.html

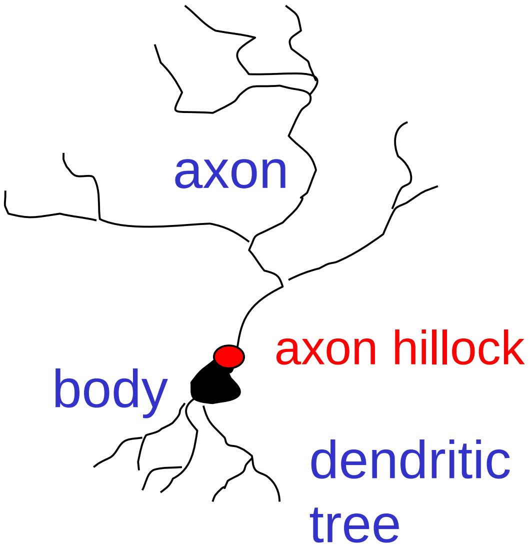
General ANN

ANN — Artificial Neural Network

- Biological Basis
- Perceptron
- Activation Function
- Multi-Layer Perceptrons
- Back-Propagation
- Loss Functions

A Typical Cortical Neuron

- **Physical structure:**
  - one ***axon*** that branches
  - a ***dendritic tree*** that collects input from other neurons
- **Axons typically contact dendritic trees at *synapses***
  - A ***spike of activity*** in the axon causes charge to be injected into the post-synaptic neuron
- **Spike generation:**
  - There is an axon hillock that generates outgoing spikes whenever enough charge has flowed in at synapses to depolarize the cell membrane

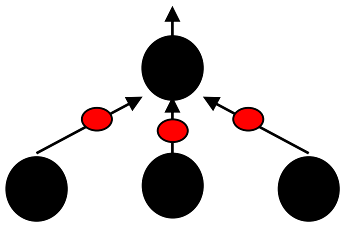


Synapses

- When a spike of activity travels along an axon and arrives at a synapse it causes vesicles of transmitter chemical to be released.
  - There are several kinds of transmitter.
- The transmitter molecules diffuse across the synaptic cleft and bind to receptor molecules in the membrane of the post-synaptic neuron thus changing their shape.
  - This opens up holes that allow specific ions in or out.

How the brain works

- Each neuron receives inputs from other neurons
  - A few neurons also connect to receptors.
  - Cortical neurons use spikes to communicate.



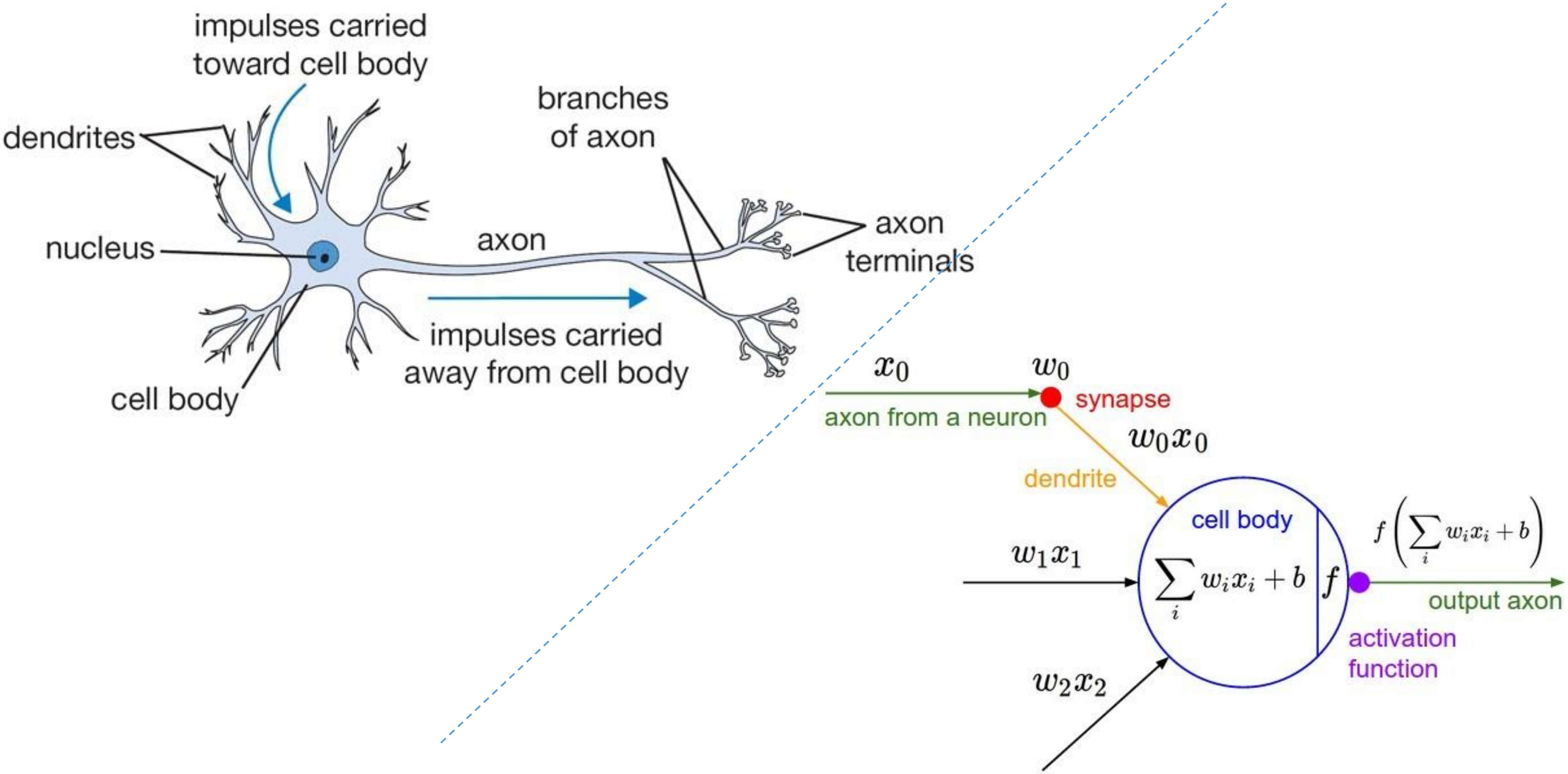
- The effect of each input line on the neuron is controlled by a synaptic weight
  - The weights can be positive or negative.
- The synaptic weights adapt so that the whole network learns to perform useful computations
  - Recognizing objects, understanding language, making plans, controlling the body.
- You have about 10^{11} neurons each with about 10^{4} weights.
  - A huge number of weights can affect the computation in a very short time. Much better bandwidth than a workstation.

Modularity and the brain

- Different bits of the cortex do different things.
  - Local damage to the brain has specific effects.
  - Specific tasks increase the blood flow to specific regions.
- But cortex looks pretty much the same all over.
  - Early brain damage makes functions relocate.
- Cortex is made of general purpose stuff that has the ability to turn into special purpose hardware in response to experience.
  - This gives rapid parallel computation plus flexibility.
  - Conventional computers get flexibility by having stored sequential programs, but this requires very fast central processors to perform long sequential computations.

Biological and Mathematical

- Comparing the biological Vs. the mathematical structure

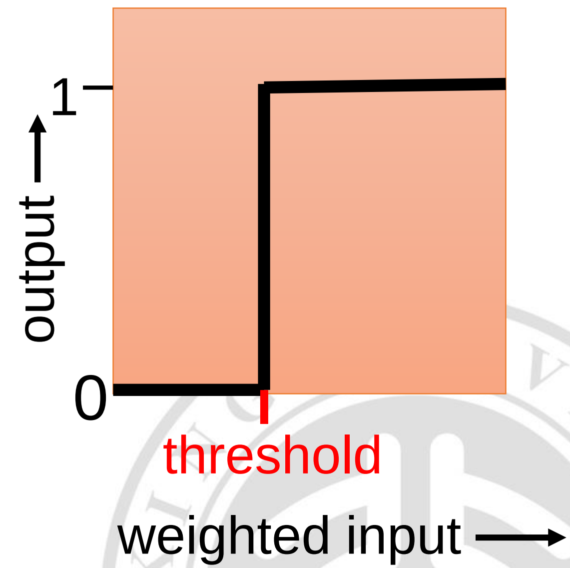


Binary Threshold Neurons

- McCulloch-Pitts [1943]:
  - Compute a weighted sum of the inputs
  - Send out a *fixed size spike* of activity, if the weighted sum exceeds a threshold
  - McCulloch and Pitts thought that

"Each spike is like the truth value of a proposition,

and each neuron combines truth values to compute the truth value of another proposition!"



Linear Neurons

- These are simple but computationally limited
  - If we can make them learn, we may get insight into more complicated neurons.

$$z = \sum_i x_i w_i \qquad \xrightarrow{\theta = -b} \qquad z = b + \text{å}_i\, x_i w_i$$

$$y = \begin{cases} 1 \text{ if } z \geq \theta \\ 0 \text{ otherwise} \end{cases} \qquad\qquad y = \begin{cases} 1 \text{ if } z^3\ 0 \\ 0 \text{ otherwise} \end{cases}$$

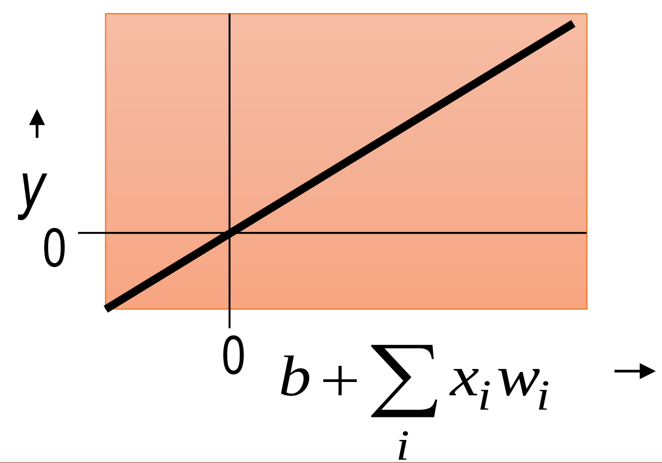
bias, i th input

$$y = b + \sum_i x_i w_i$$

output

index over input connections

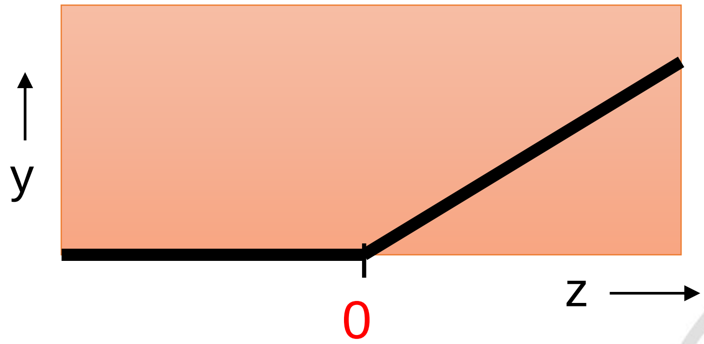
weight on i th input



Rectified Linear Neurons

- Sometimes called linear threshold neurons
- They compute a linear weighted sum of their inputs.
- The output is a non-linear function of the total input.

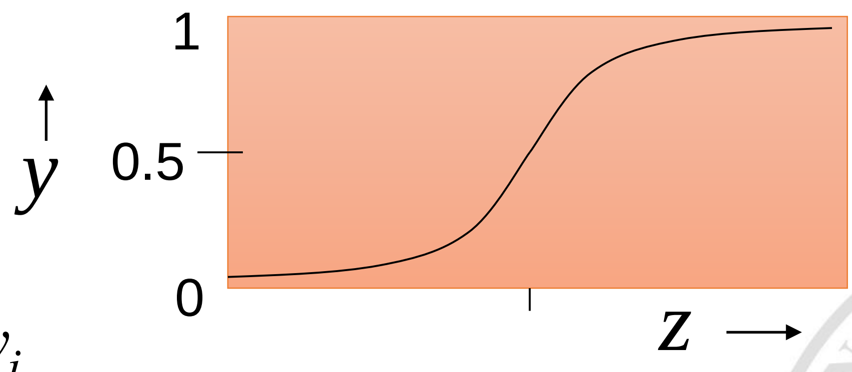
$$z = b + \sum_i x_i w_i$$



$$y = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$$

Sigmoid Neurons

- These give a real-valued output that is a smooth and bounded function of their total input.
  - Typically they use the logistic function
  - They have nice derivatives which make learning easy



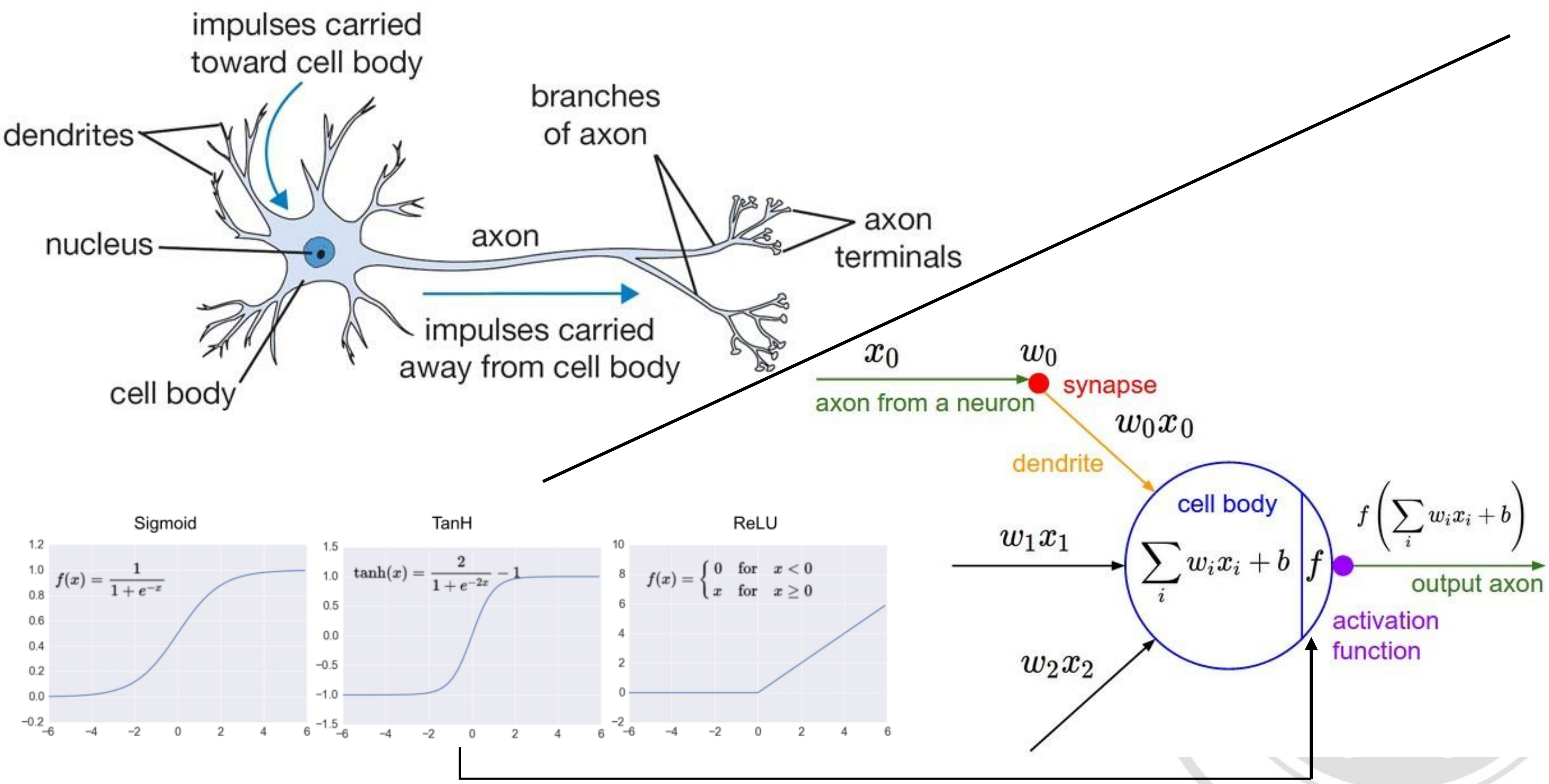
$$z = b + \sum_i x_i w_i$$

$$y = \frac{1}{1 + e^{-z}}$$

0

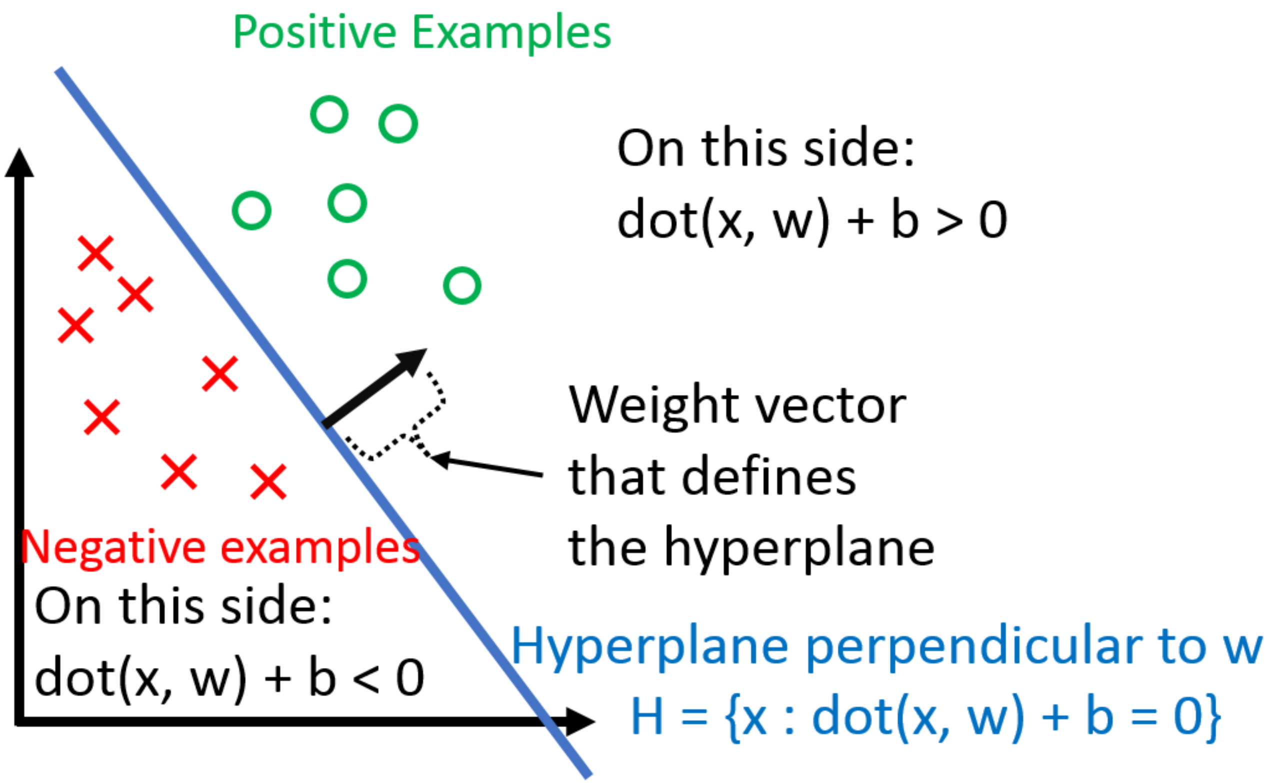
Biological and Mathematical

- Comparing the biological and the mathematical structure



What binary neurons can do

- A binary neuron can be used as a simple classifier



http://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote03.html

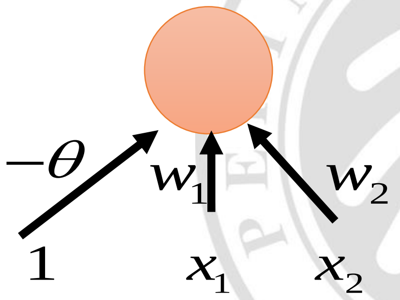
What binary neurons cannot do

- The XOR problem
- A binary threshold output unit cannot even tell if two single bit features are the same!

  Positive cases (same): (1,1) → 1; (0,0) → 1

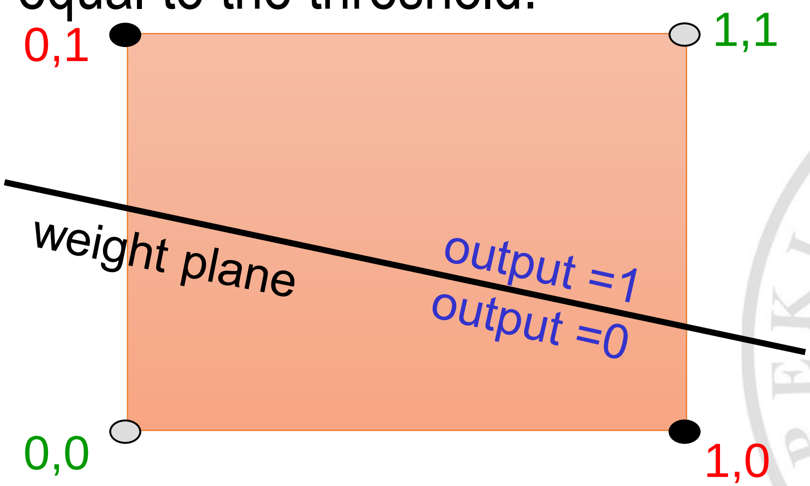
  Negative cases (different): (1,0) → 0; (0,1) → 0
- The four input-output pairs give four inequalities that are impossible to satisfy:

$$w_1 + w_2 \geq \theta, \quad 0 \geq \theta$$
$$w_1 < \theta, \quad w_2 < \theta$$



A geometric view

- Imagine "data-space" in which the axes correspond to components of an input vector.
- Each input vector is a point in this space.
- A weight vector defines a plane in data-space.
- The weight plane is perpendicular to the weight vector and misses the origin by a distance equal to the threshold.



The positive and negative cases cannot be separated by a plane

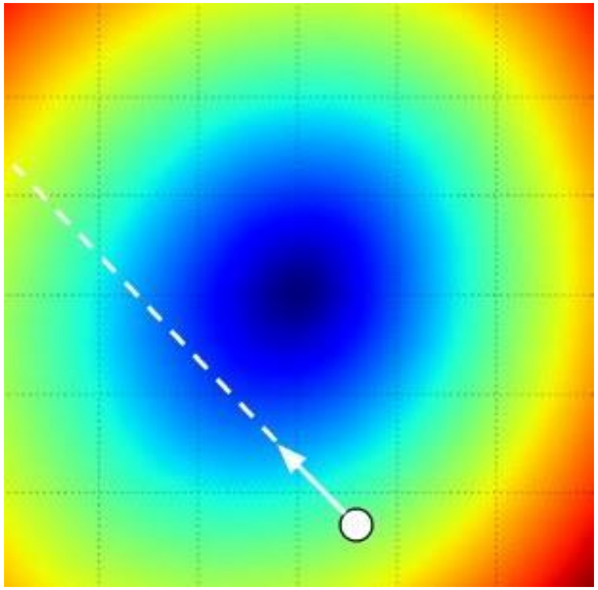
Learning with hidden units

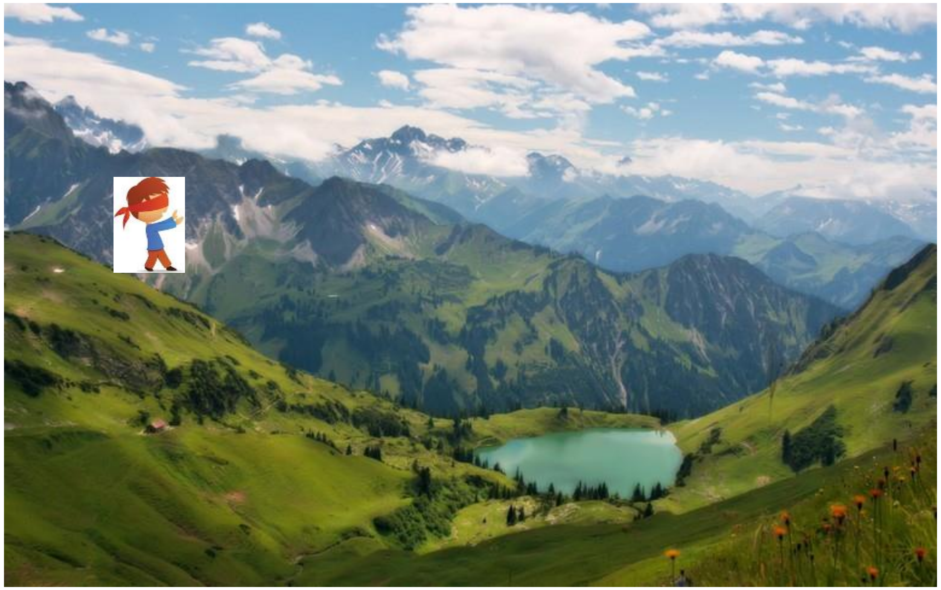
- Networks without hidden units are very limited in the input-output mappings they can learn to model.
  - More layers of linear units do not help. Its still linear.
  - Fixed output non-linearities are not enough.
- We need multiple layers of adaptive, non-linear hidden units. But how can we train such nets?
  - We need an efficient way of adapting all the weights, not just the last layer. This is hard.
  - Learning the weights going into hidden units is equivalent to learning features.
  - This is difficult because nobody is telling us directly what the hidden units should do.

Optimization Landscape

- **Gradient Descent**

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$





Example of Gradients

$$f(x,y) = xy \qquad \rightarrow \qquad \frac{\partial f}{\partial x} = y \qquad \frac{\partial f}{\partial y} = x$$

Example: x = 4, y = -3 => f(x,y) = -12

$$\boxed{\frac{\partial f}{\partial x} = -3} \qquad \boxed{\frac{\partial f}{\partial y} = 4} \qquad \boxed{\nabla f = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}]}$$

partial derivatives

gradient

Compound Expressions:

- **Example of Gradients**

$$f(x, y, z) = (x + y)z$$

$$\boxed{q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1}$$

$$\boxed{f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q}$$

Compound Expressions:

- **Example of Gradients**

$$f(x, y, z) = (x + y)z$$

$$\boxed{q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1}$$

$$\boxed{f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q}$$

Chain rule:

$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}}$$

Compound Expressions:

- **Example of Gradients**

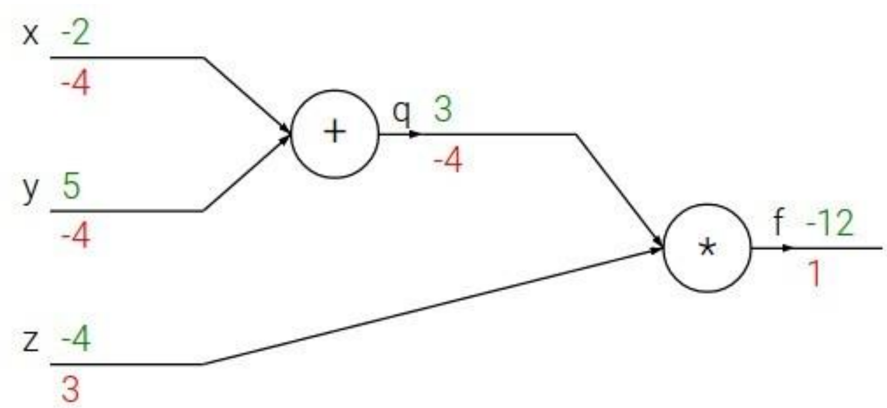
$$f(x,y,z) = (x+y)z$$

$$\boxed{q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1}$$

$$\boxed{f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q}$$

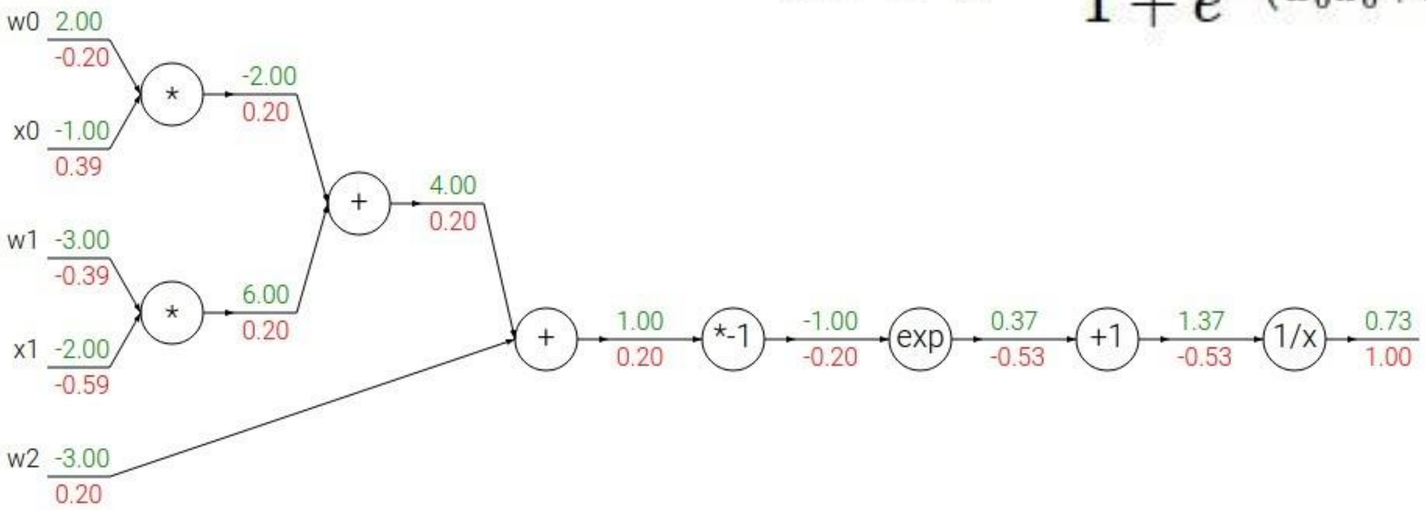
Chain rule:

$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}}$$



BP: Another Example

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



$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

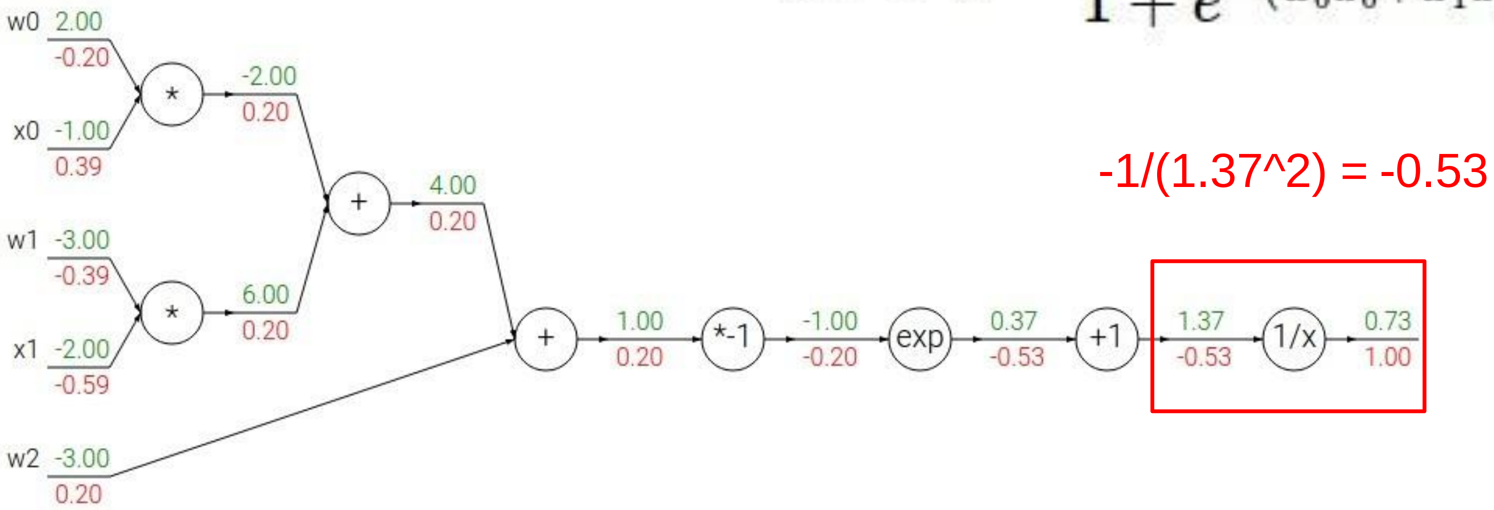
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

BP: Another Example

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



$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

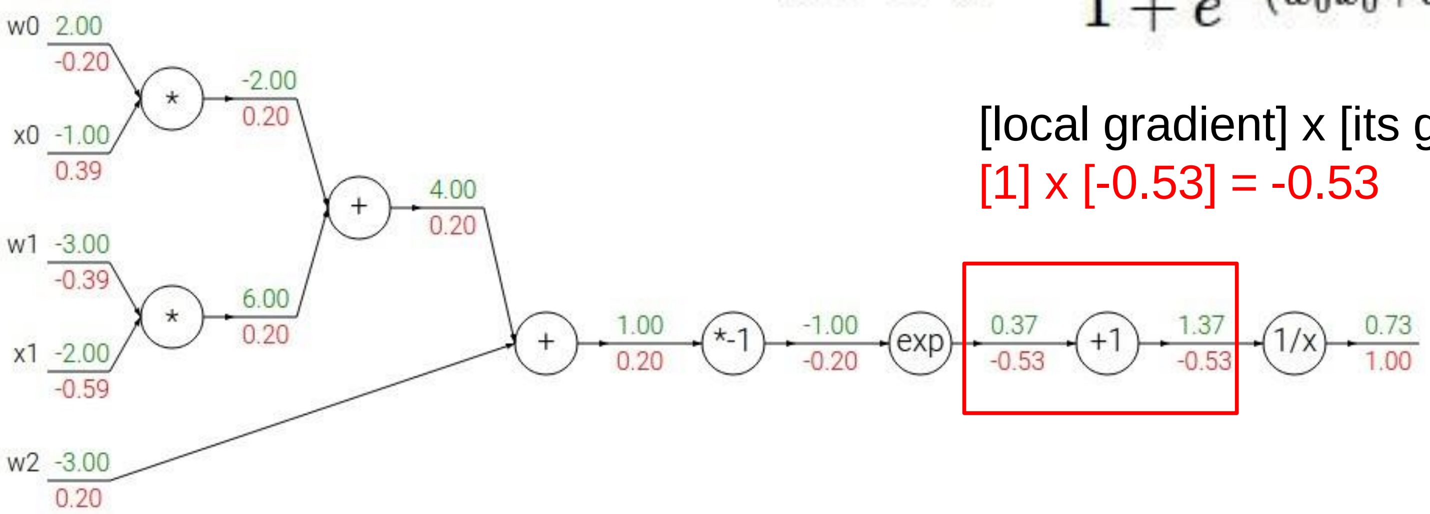
$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

BP: Another Example

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

[local gradient] x [its gradient]

[1] x [-0.53] = -0.53



$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

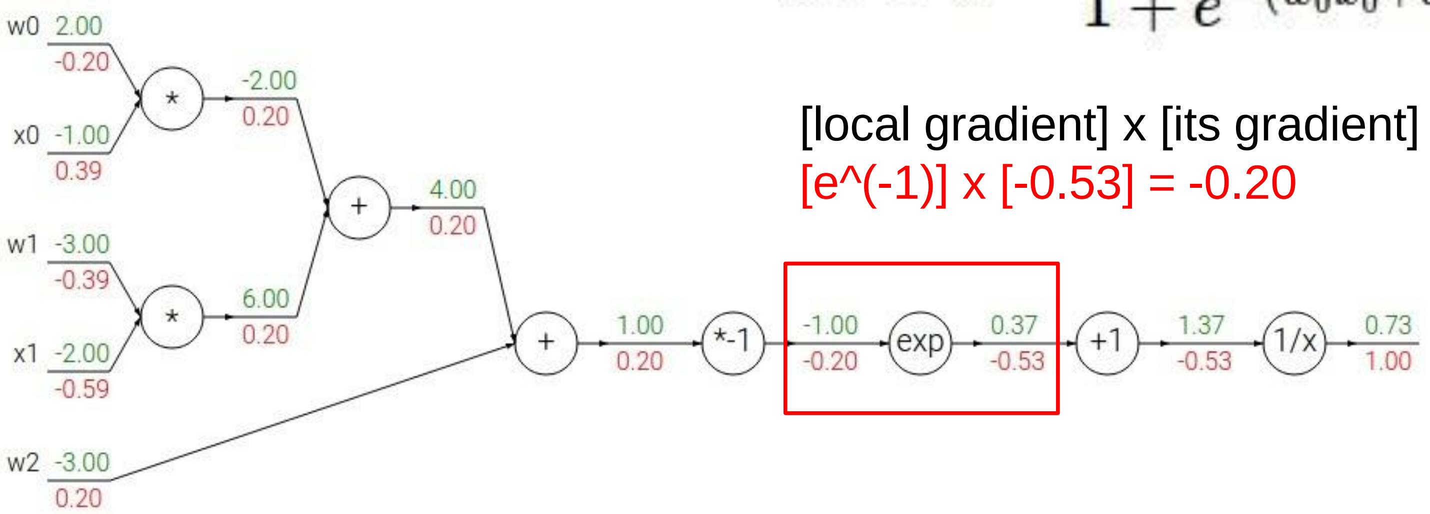
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

BP: Another Example

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



[local gradient] x [its gradient]

[e^(-1)] x [-0.53] = -0.20

$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

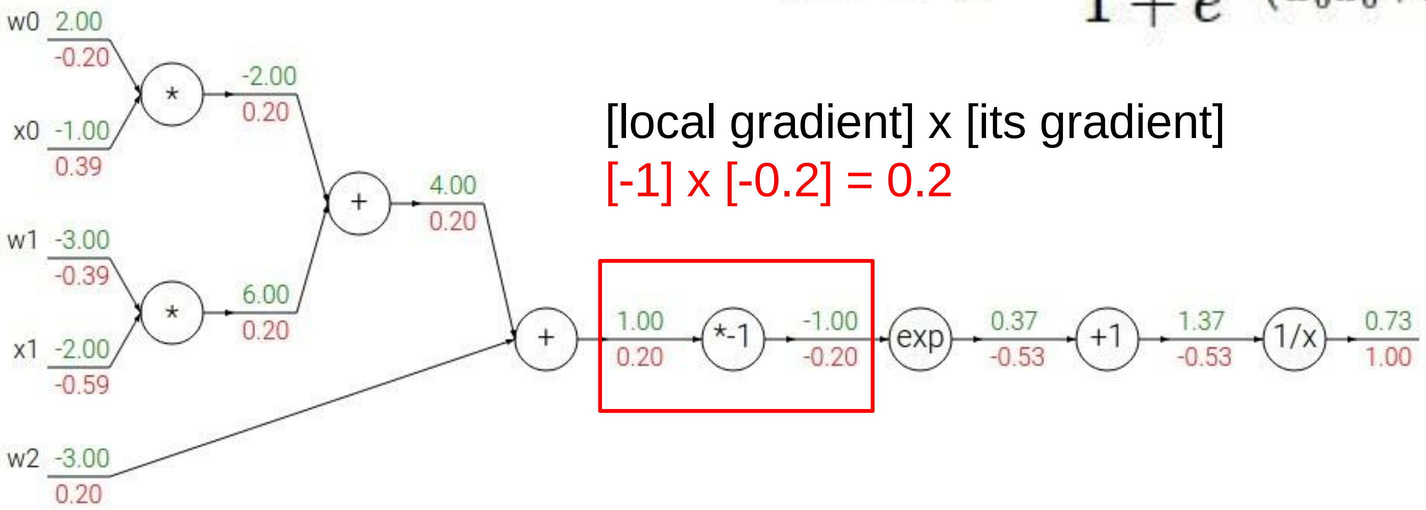
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

BP: Another Example

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

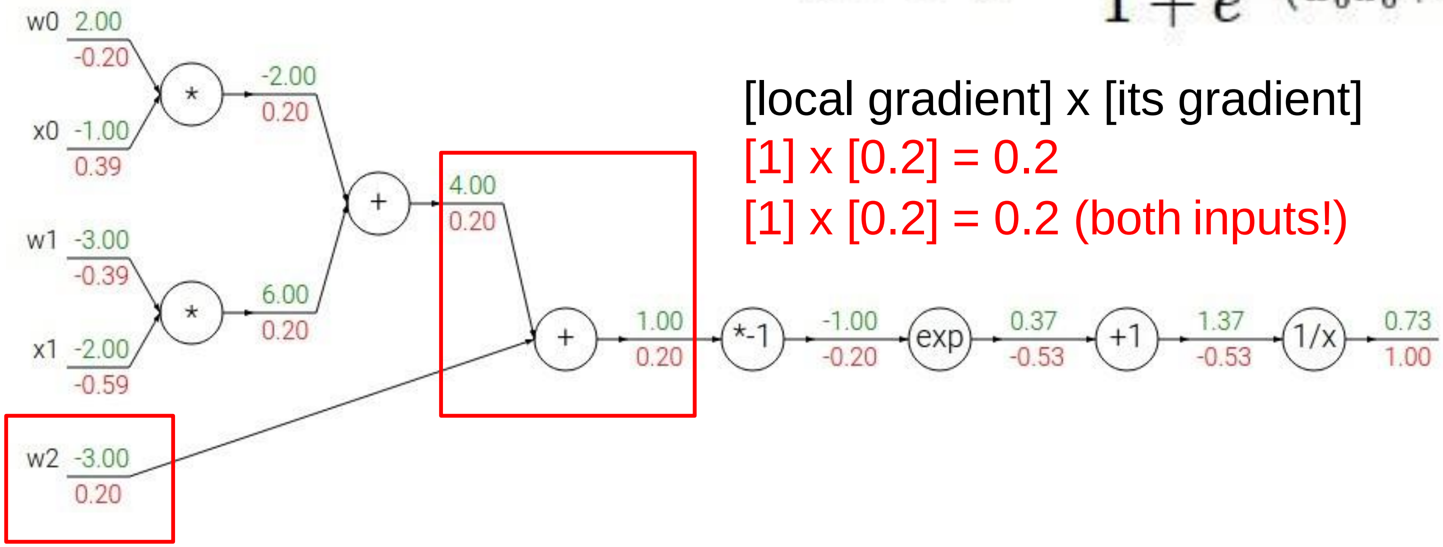


[local gradient] x [its gradient]
[-1] x [-0.2] = 0.2

$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$	$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$
$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$	$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$

BP: Another Example

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



[local gradient] x [its gradient]

[1] x [0.2] = 0.2

[1] x [0.2] = 0.2 (both inputs!)

$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

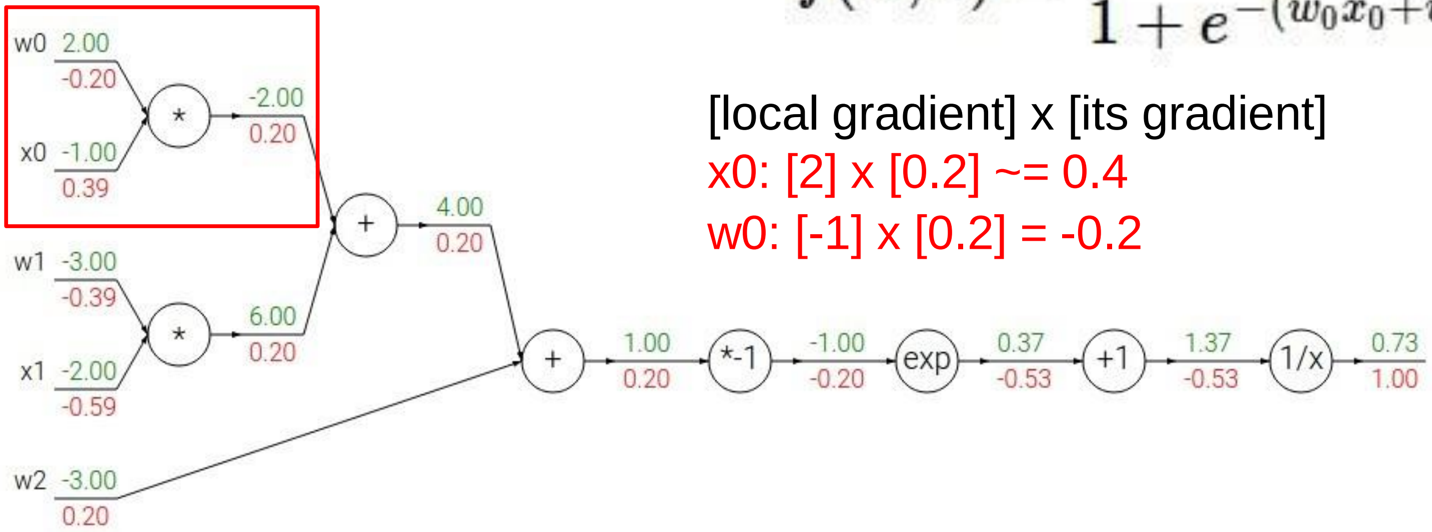
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

BP: Another Example

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



[local gradient] x [its gradient]

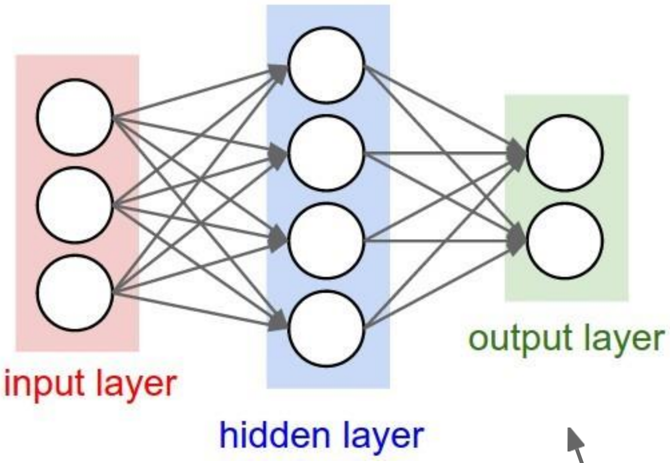
x0: [2] x [0.2] ~= 0.4

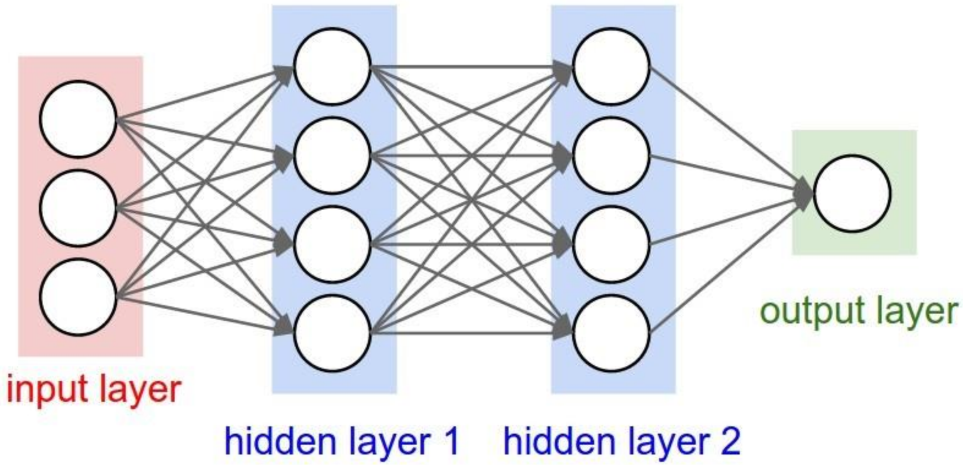
w0: [-1] x [0.2] = -0.2

$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x \qquad f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a \qquad f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Neural Networks: Architectures





"Fully-connected" layers

Loss function: Squared Error

- Expression $\mathcal{L}_{MSE} = \sum (y_i - \tilde{y}_i)^2$
- Convexity $\frac{\partial \mathcal{L}}{\partial y} = 2y$

$$\frac{\partial^2 \mathcal{L}}{\partial y^2} = 2 > 0$$

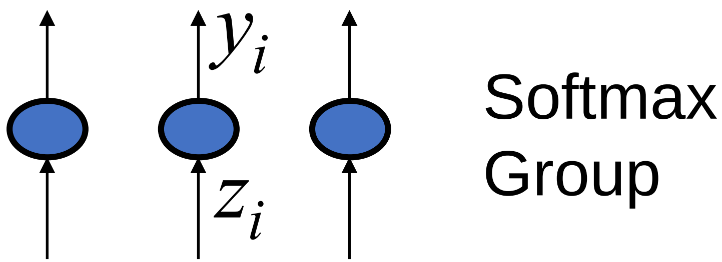
- Decrease as y_i get closer to the ground truth

Problems with Squared Error

- The squared error measure has some drawbacks:
  - If the desired output is 1 and the actual output is 0.00000001 there is almost no gradient for a logistic unit to fix up the error.
  - If we are trying to assign probabilities to mutually exclusive class labels, we know that the outputs should sum to 1, but we are depriving the network of this knowledge.
- Is there a different cost function that works better?
  - Yes: Force the outputs to represent a probability distribution across discrete alternatives.

Softmax

- The output units in a softmax group use a non-local non-linearity:



$$y_i = \frac{e^{z_i}}{\sum_{j \in group} e^{z_j}}$$

$$\frac{\partial y_i}{\partial z_i} = y_i \left(1 - y_i\right)$$

Cross-entropy: *right cost function to use with softmax*

- The right cost function is the negative log probability of the right answer.
- C has a very big gradient when the target value is 1 and the output is almost 0.

  - A value of 0.000001 is much better than 0.000000001

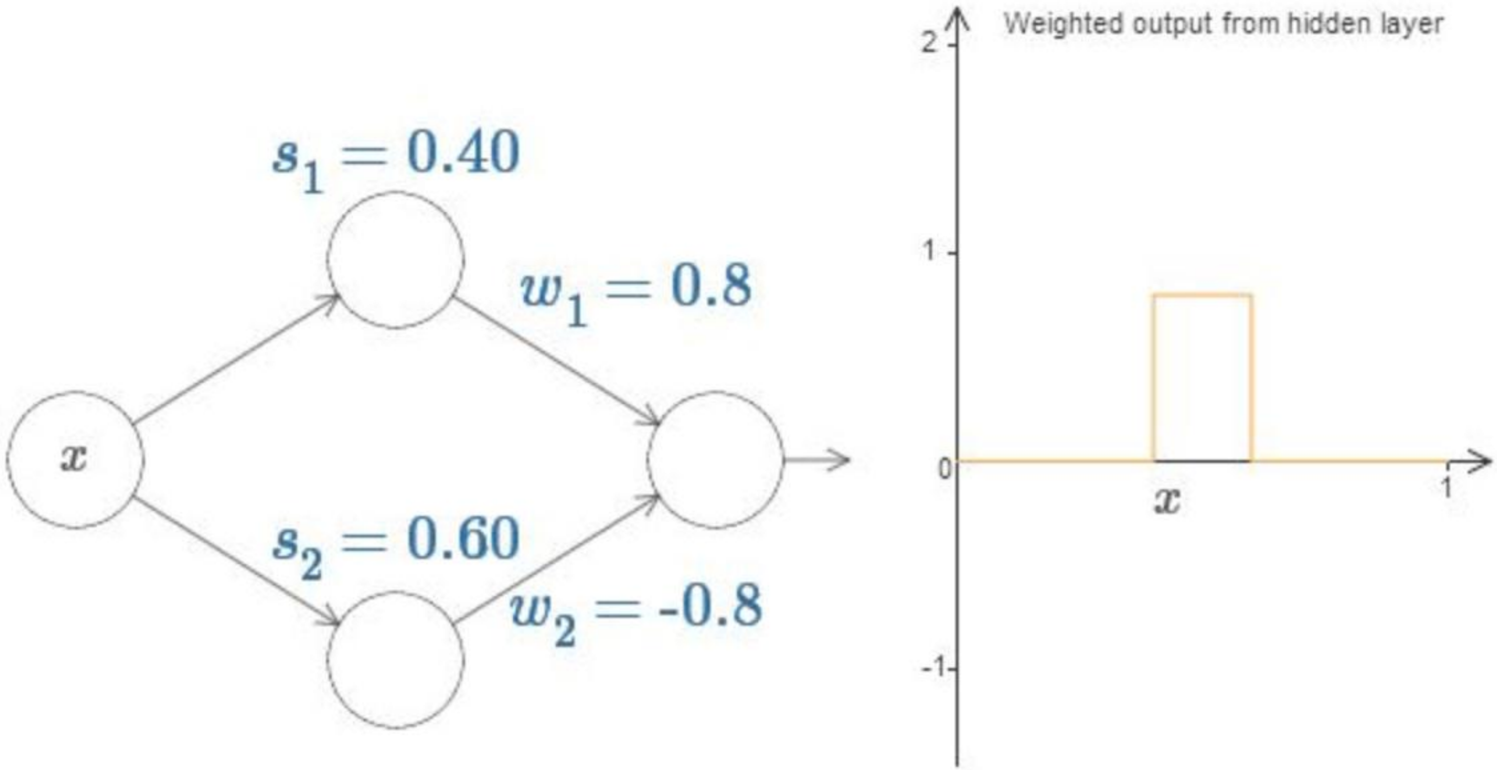
  - The steepness of dC/dy exactly balances the flatness of dy/dz

$$C = -\sum_j t_j \log y_j$$

target value

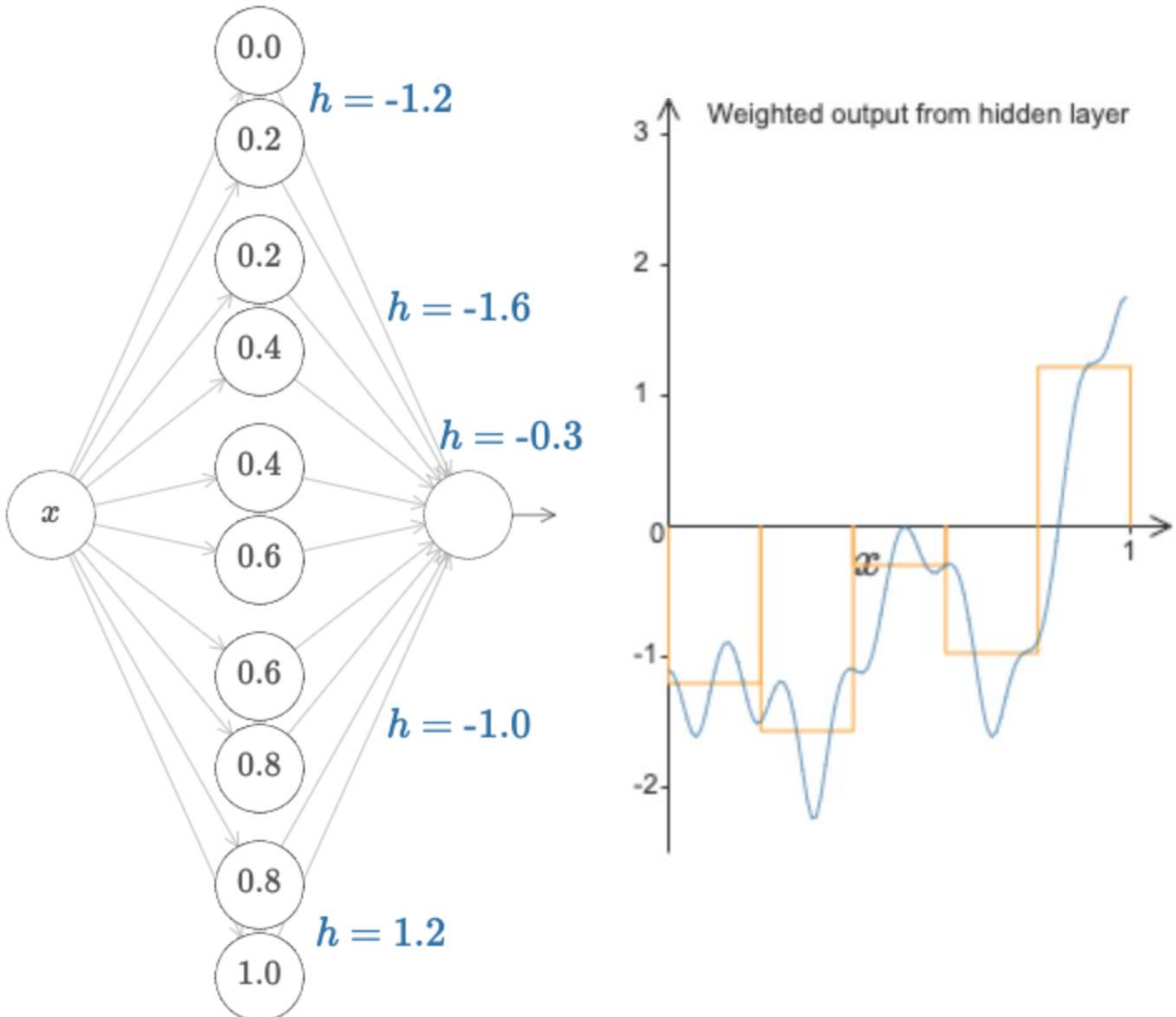
$$\frac{\partial C}{\partial z_i} = \sum_j \frac{\partial C}{\partial y_j} \frac{\partial y_j}{\partial z_i} = y_i - t_i$$

What kinds of functions can a NN represent



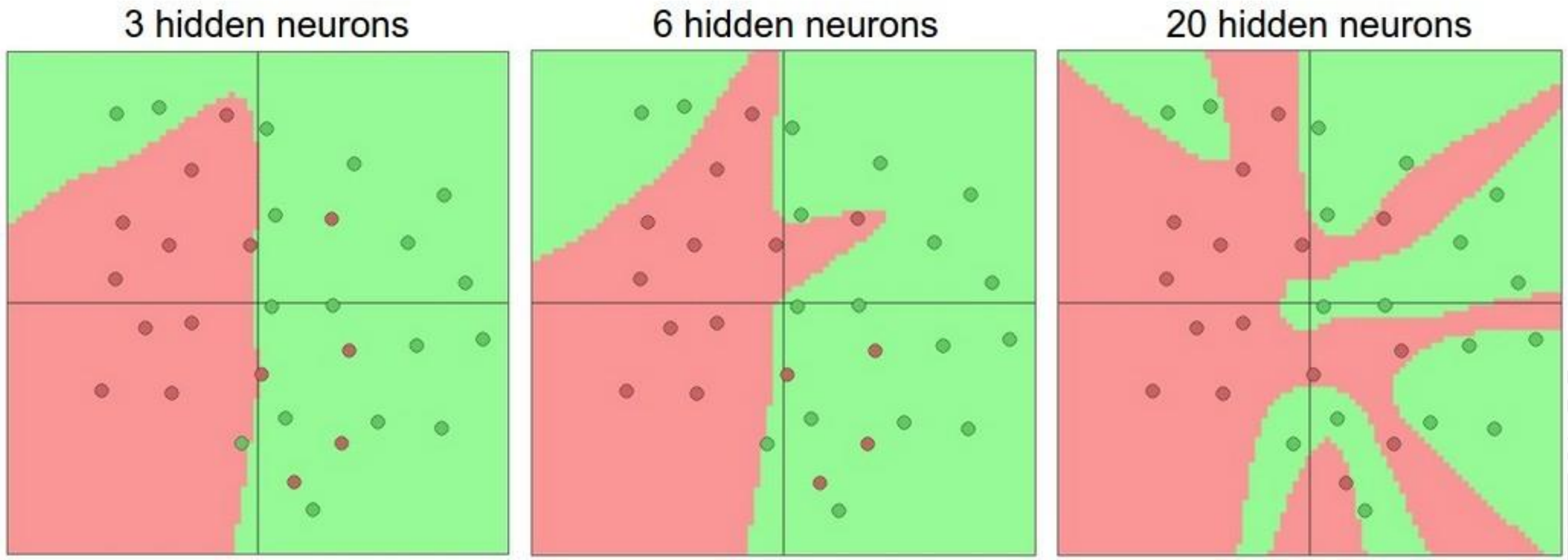
[http://neuralnetworksanddeeplearning.com/chap4.html]

What kinds of functions can a NN represent



[http://neuralnetworksanddeeplearning.com/chap4.html]

What kinds of functions can a NN represent



more neurons = more capacity

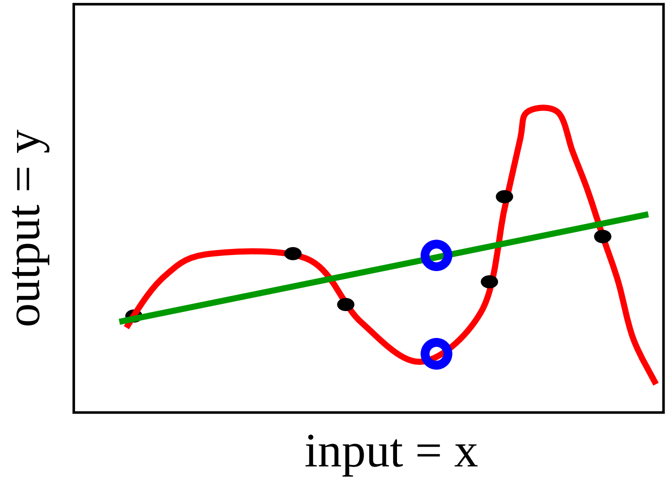
Overfitting: The downside of using powerful models

- The training data contains information about the regularities in the mapping from input to output. But it also contains two types of noise.
  - The target values may be unreliable (usually only a minor worry).
  - There is sampling error. There will be accidental regularities just because of the particular training cases that were chosen.
- When we fit the model, it cannot tell which regularities are real and which are caused by sampling error.
  - So it fits both kinds of regularity.
  - If the model is very flexible it can model the sampling error really well. This is a disaster.

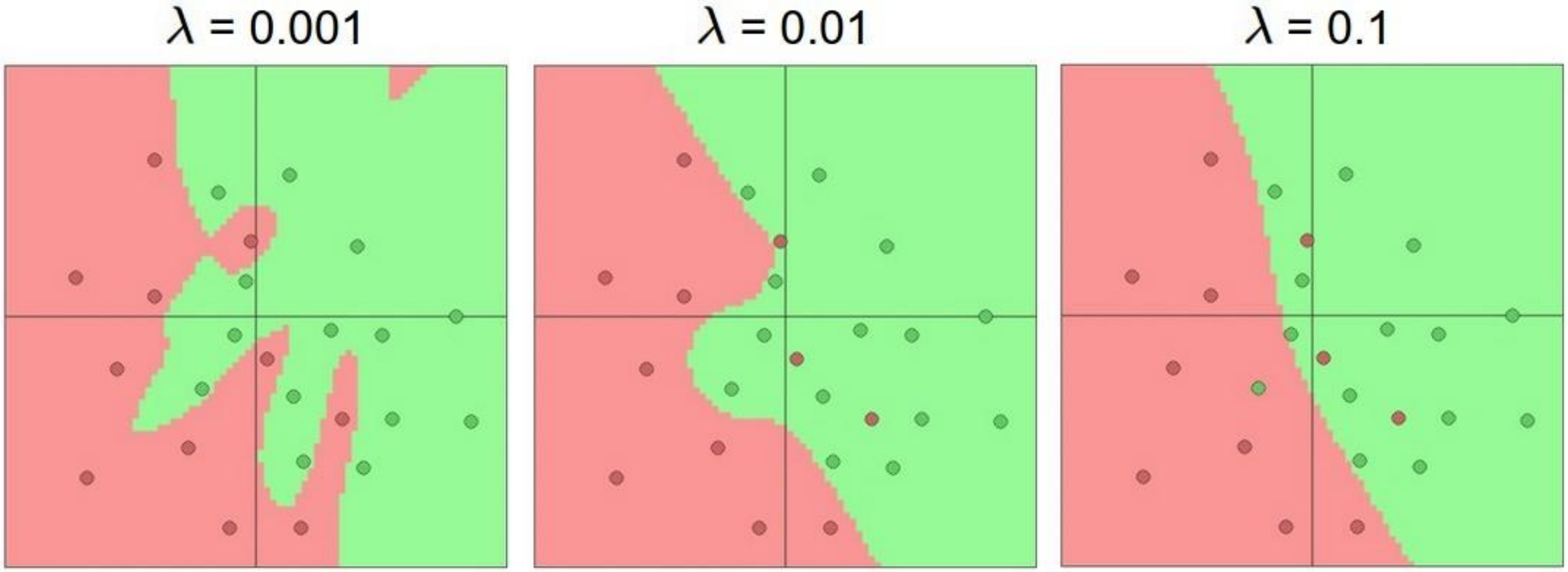
Simple Example of Overfitting

- Which model do you trust?
  - The complicated model fits the data better.
  - But it is not economical.
- A model is convincing when it fits a lot of data surprisingly well.
  - It is not surprising that a complicated model can fit a small amount of data well.

Which output value should you predict for this test input?

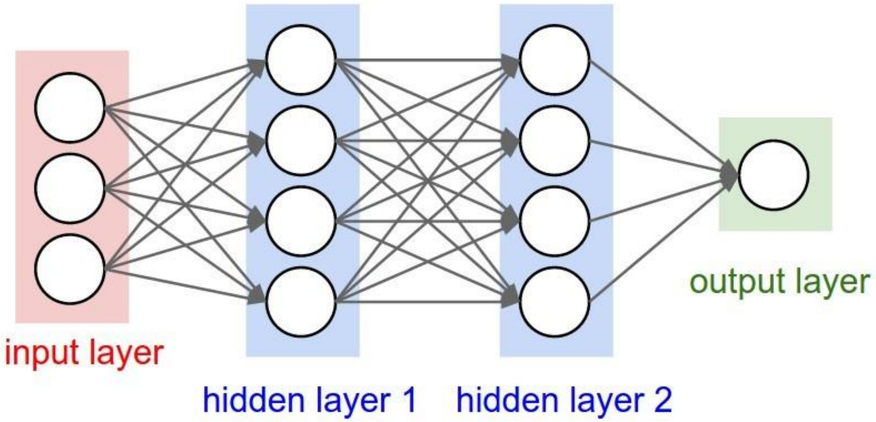


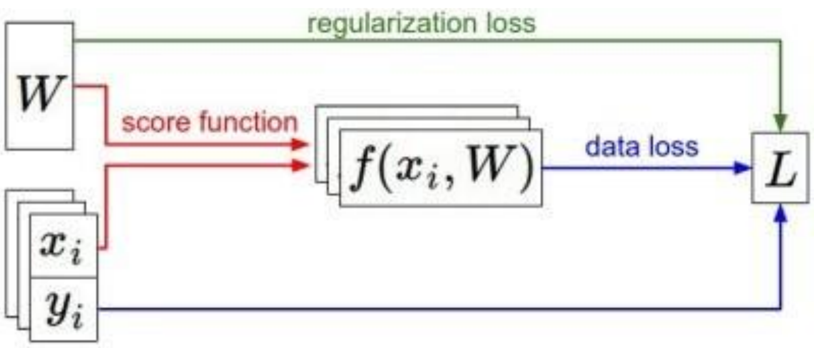
Use Regularization



Where we are right now...

- A multi-layer network
  - Approximating a wide range of functions
  - Loss function
  - Gradient Descent
  - Back-Propagation





Outline

- General ANN
- **CNN Network**
- RNN Network and Gated Network
- Beyond CNN and RNN

A bit of History

Hubel & Wiesel,

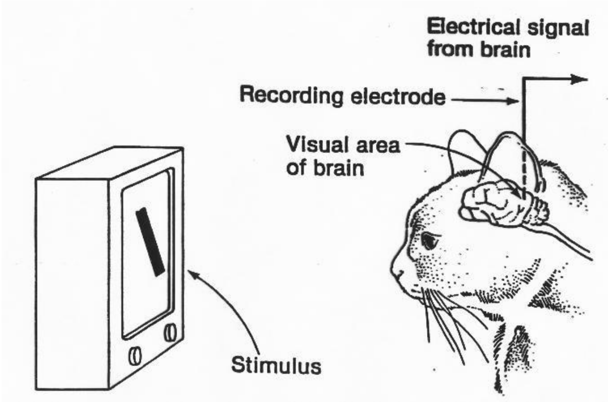
1959

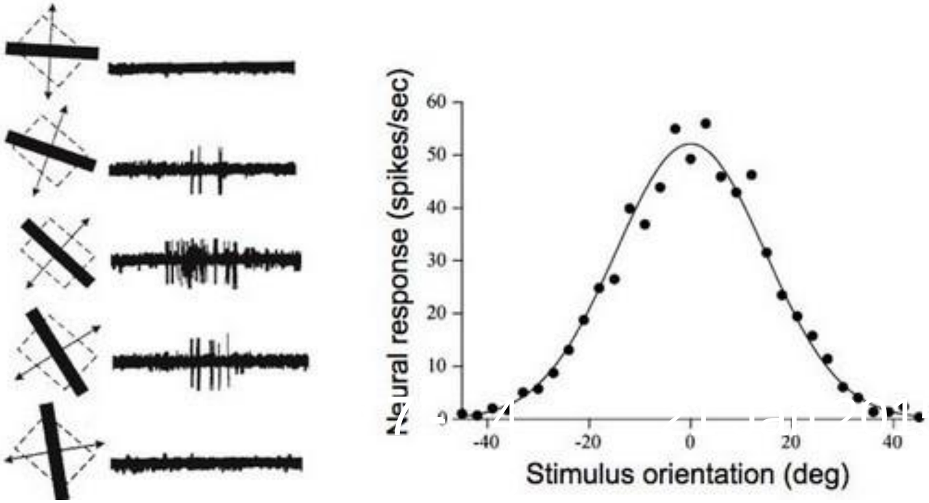
RECEPTIVE FIELDS OF SINGLE NEURONES IN
THE CAT'S STRIATE CORTEX

1962

RECEPTIVE FIELDS, BINOCULAR INTERACTION
AND FUNCTIONAL ARCHITECTURE IN
THE CAT'S VISUAL CORTEX

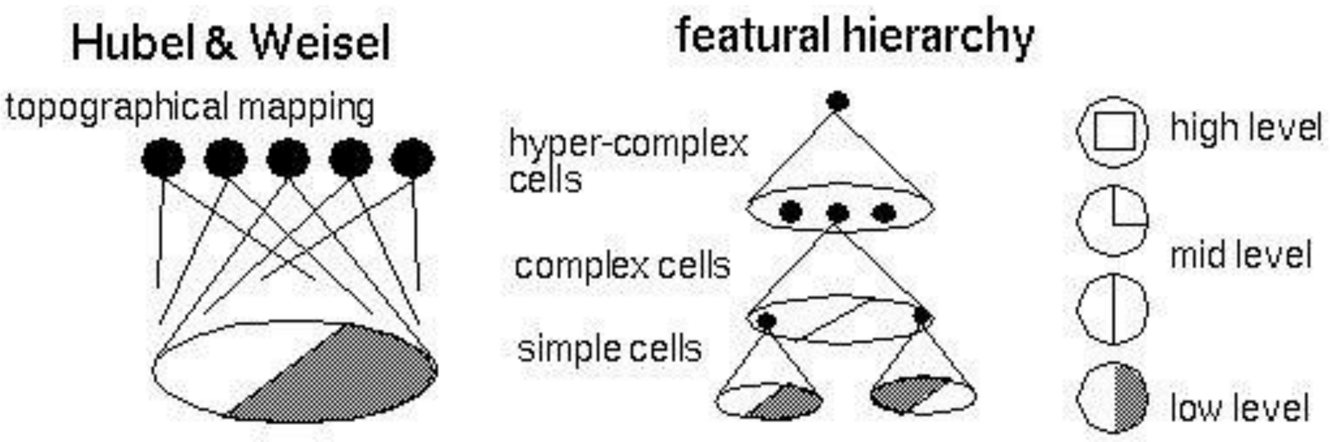
1968...

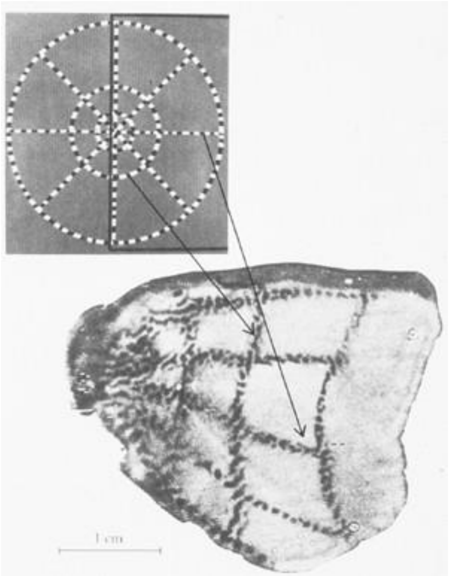




A bit of History

- **Topographical mapping in the cortex:** nearby cells in cortex represented nearby regions in the visual field.

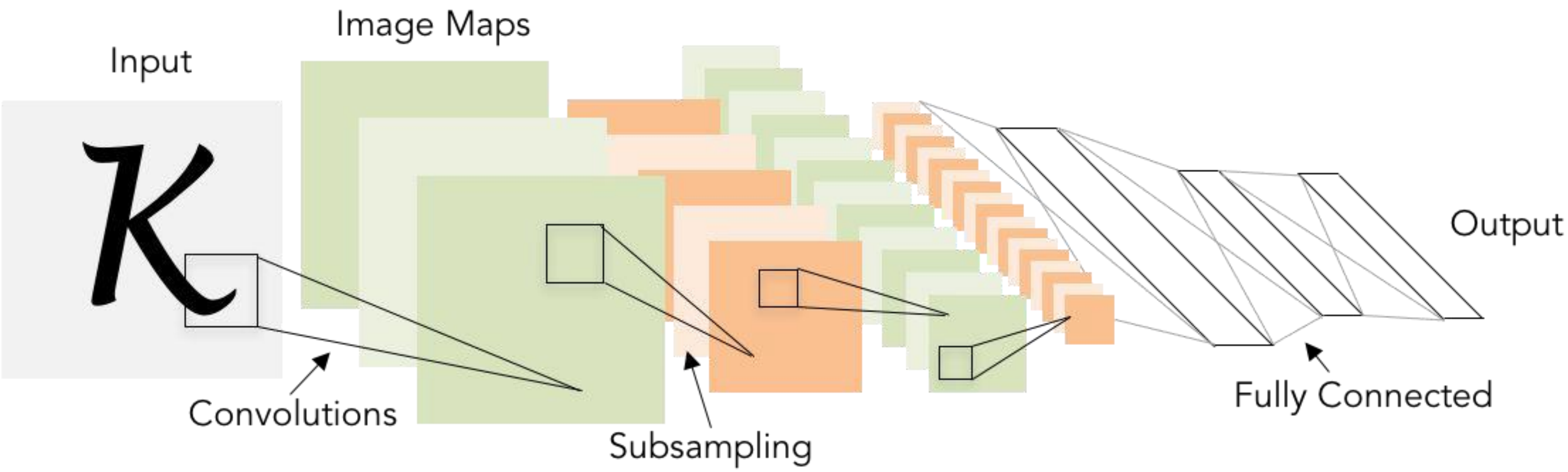




A bit of History

- **Gradient-based learning applied to document recognition**

[LeCun, Bottou, Bengio, Haffner 1998]

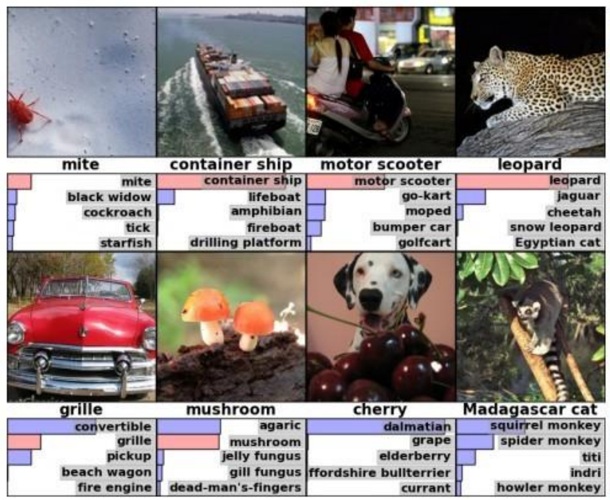


LeNet-5

Fast-forward to today: ConvNets are everywhere

[Goodfellow 2014]

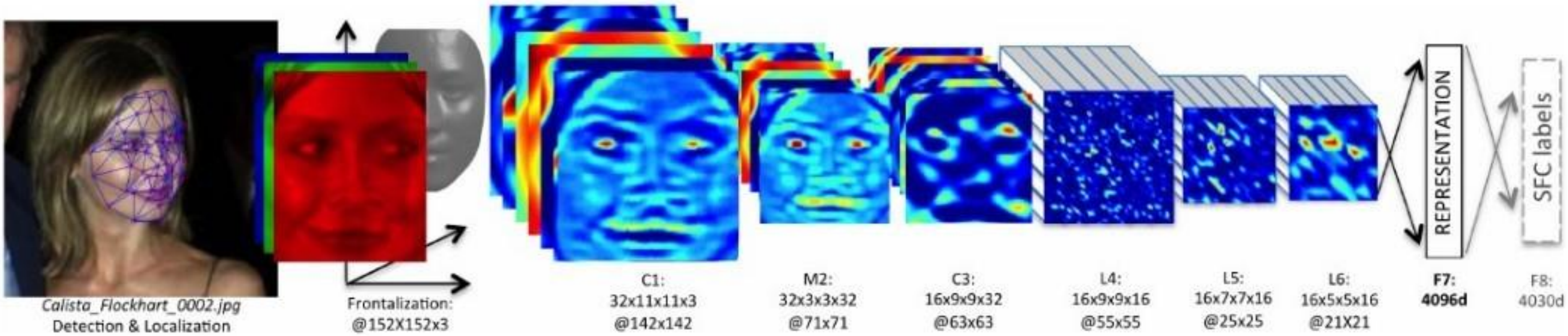
Classification



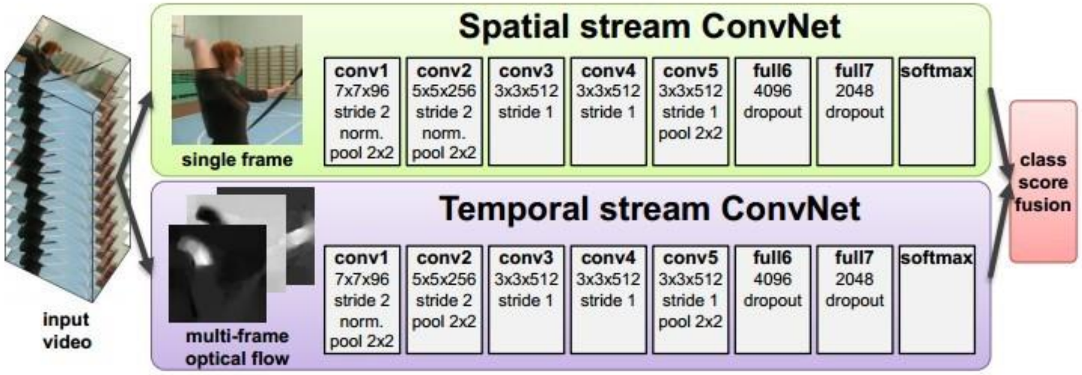
Retrieval

[Krizhevsky 2012]

Fast-forward to today: ConvNets are everywhere



[Taigman et al. 2014]



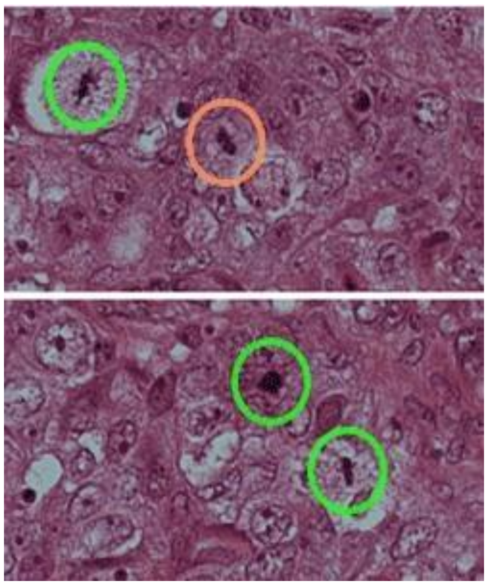
[Simonyan et al. 2014]

Fast-forward to today: ConvNets are everywhere

[Toshev, Szegedy 2014]

[Mnih 2013]

Fast-forward to today: ConvNets are everywhere

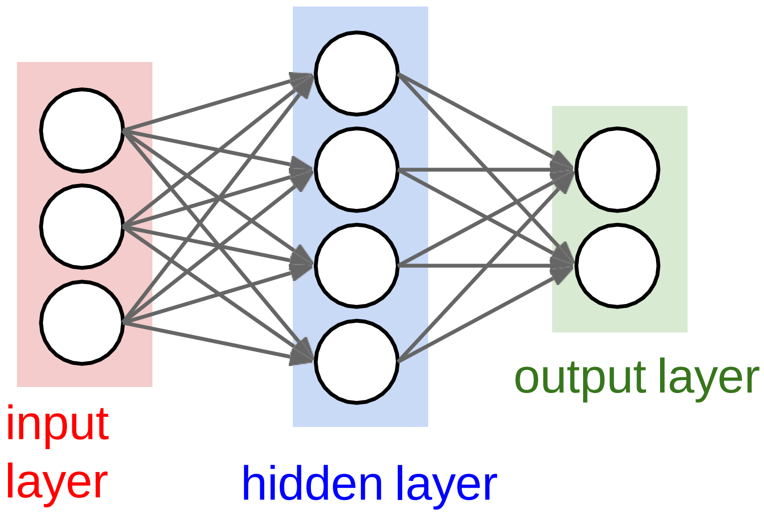


[Ciresan et al. 2013]

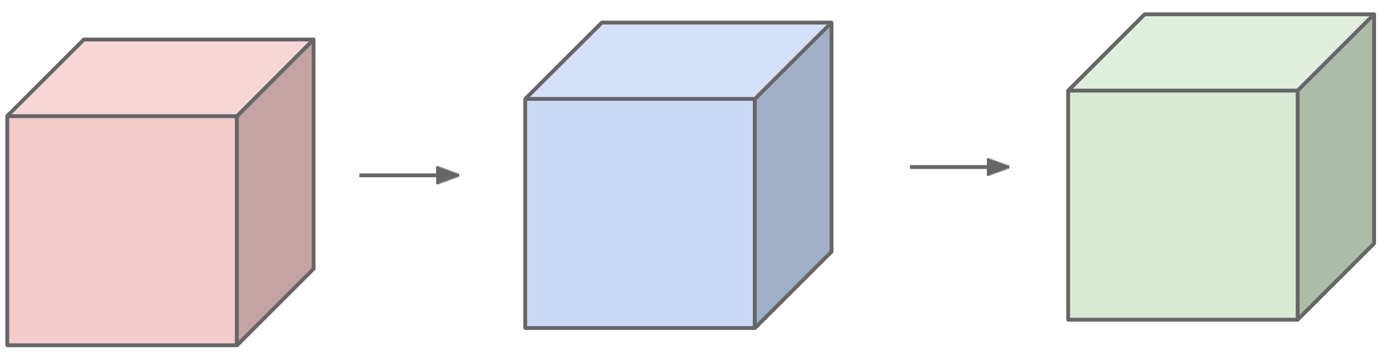
[Sermanet et al. 2011]
[Ciresan et al.]

Convolutional Neural Network

before:

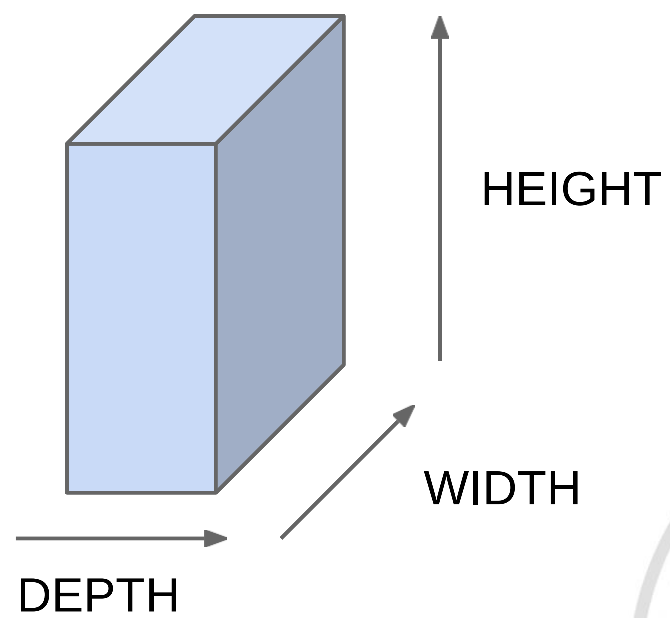


now:



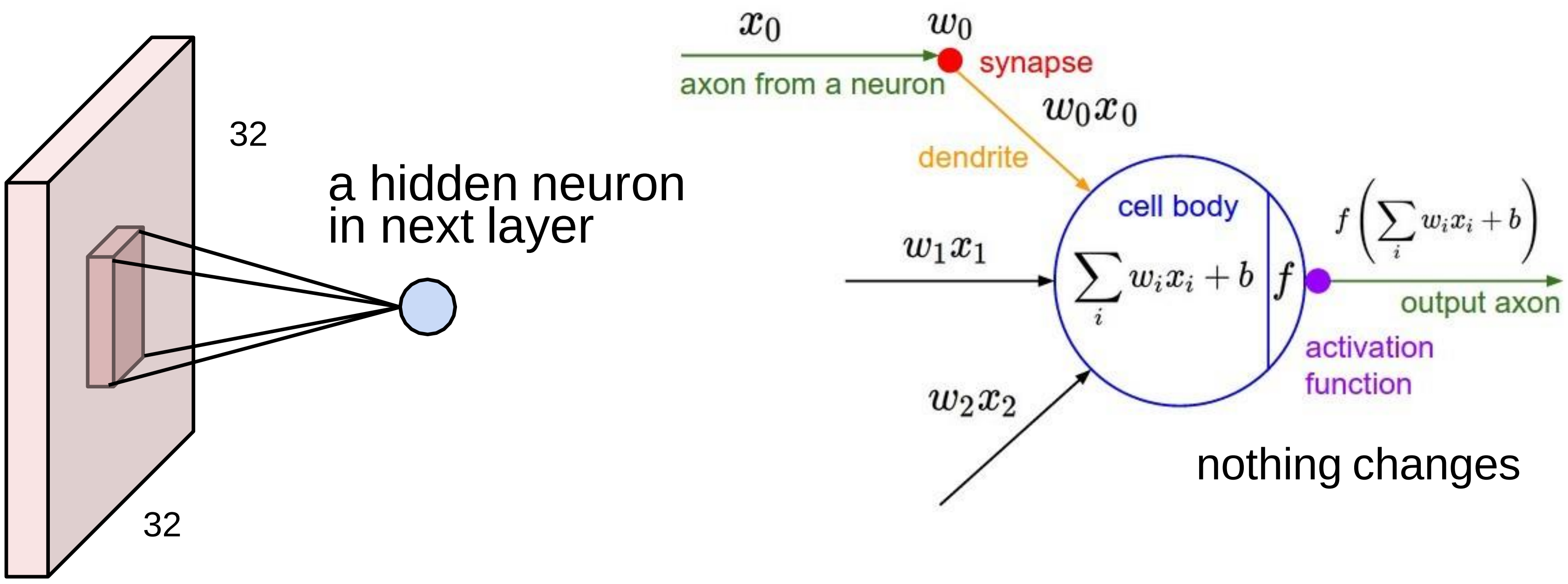
Convolutional Neural Network

- All Neural Net activations arranged in **3 dimensions:**



For example, a CIFAR-10 image is a 32x32x3 volume
32 width, 32 height, 3 depth (RGB channels)

Local connectivity

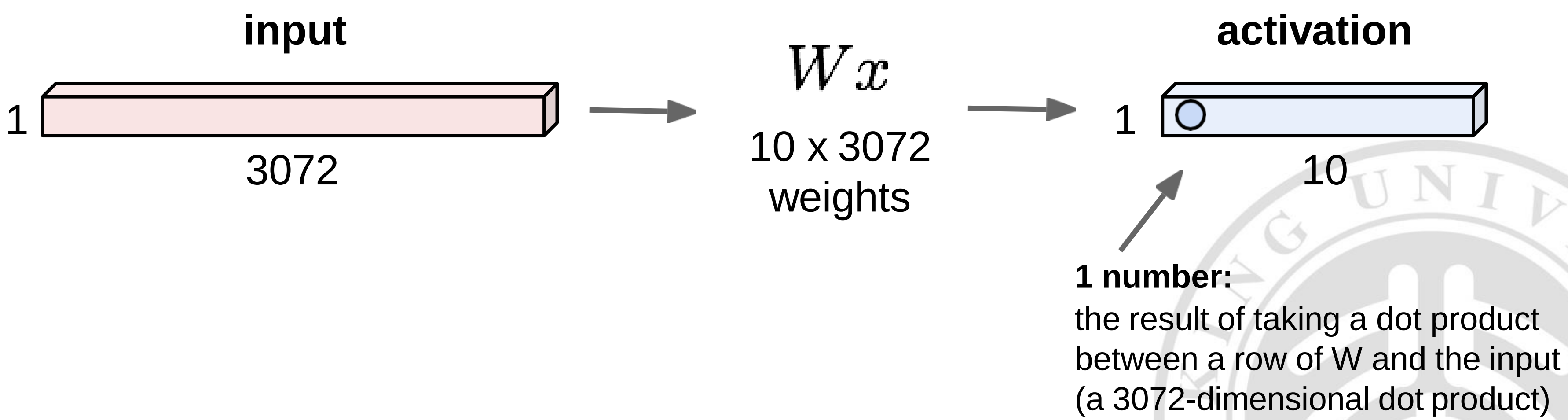


nothing changes

ConvNet are just Neural Networks BUT:

- Fully Connected Layer:

32x32x3 image -> stretch to 3072 x 1

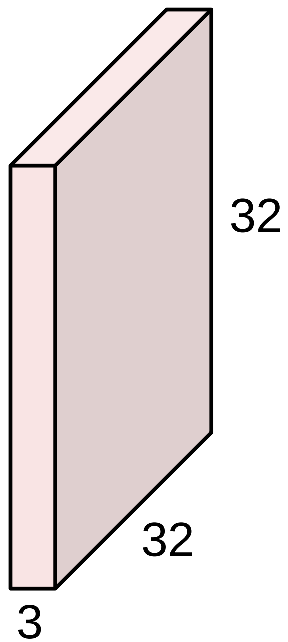


ConvNet are just Neural Networks BUT:

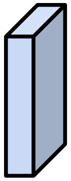
- Convolutional Layer:

Filters always extend the full depth of the input volume

32x32x3 image



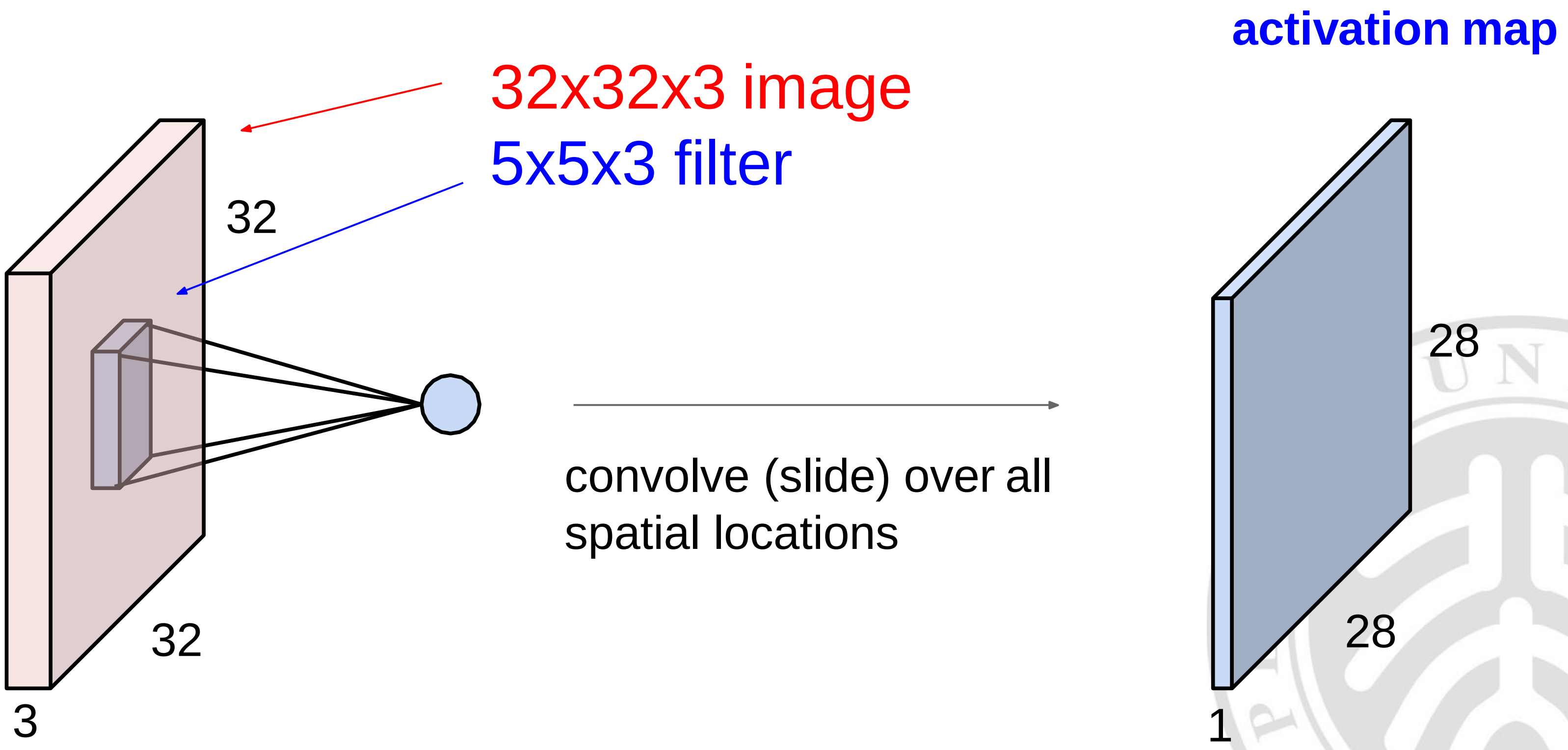
5x5x3 filter



Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

ConvNet are just Neural Networks BUT:

- Convolutional Layer:



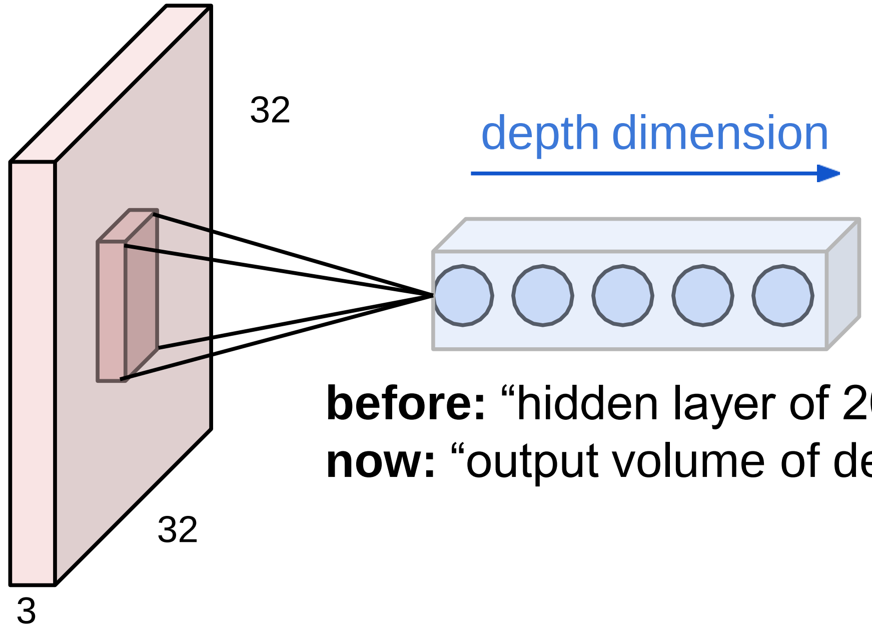
ConvNet are just Neural Networks BUT:

- Convolutional Layer:

consider a second, green filter

ConvNet are just Neural Networks BUT:

Depth Dimension

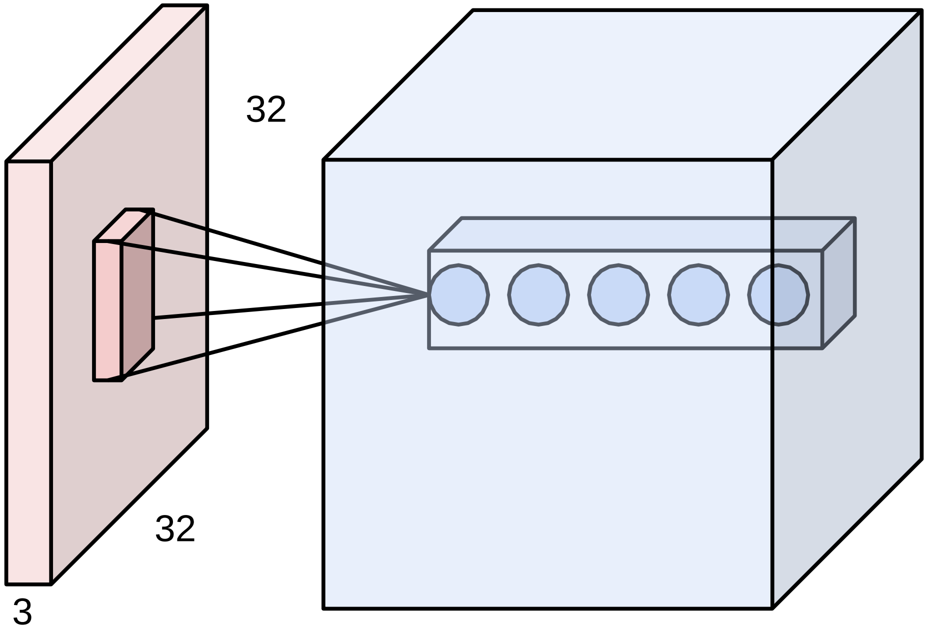


before: "hidden layer of 200 neurons"
now: "output volume of depth 200"

Multiple neurons all looking at the same region of the input volume, stacked along depth.

ConvNet are just Neural Networks BUT:

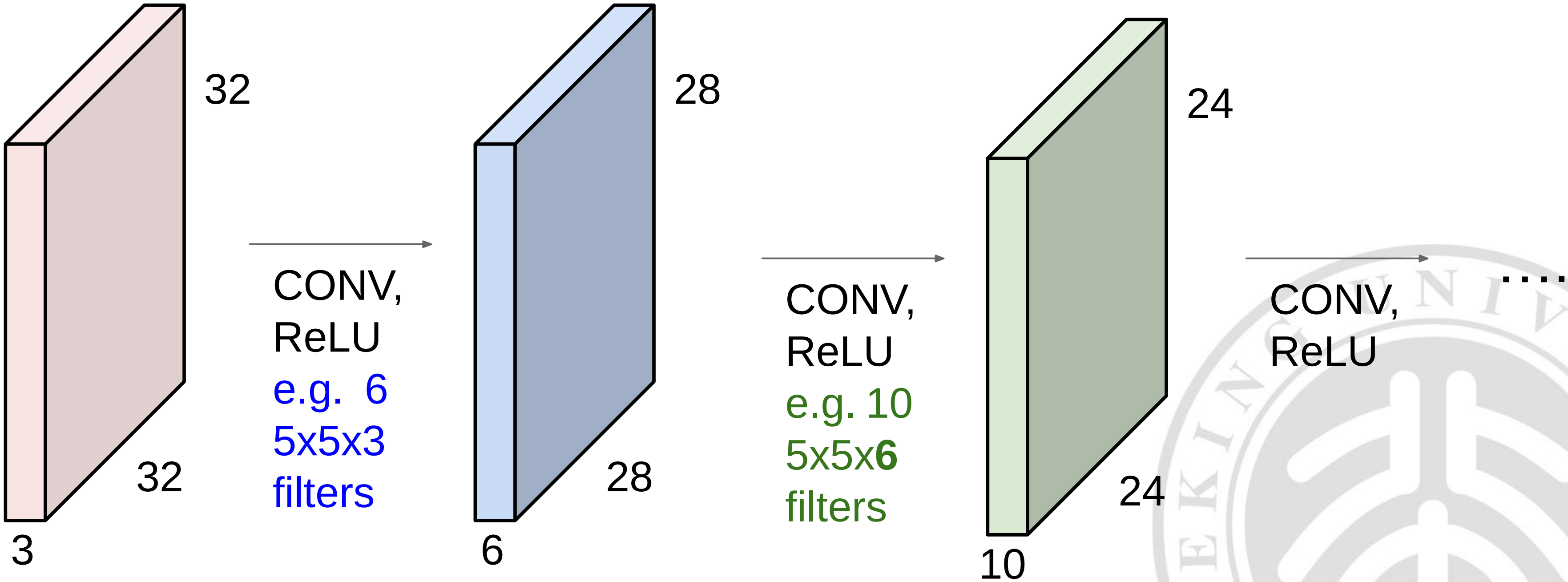
Feature maps



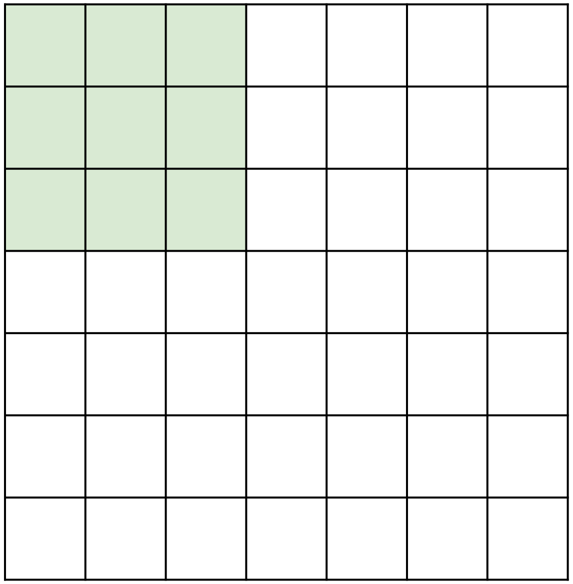
These form a single [1 x 1 x depth] "depth column" in the output volume

ConvNet are just Neural Networks BUT:

- ConvNet is a sequence of Convolution Layers, interspersed with activation functions

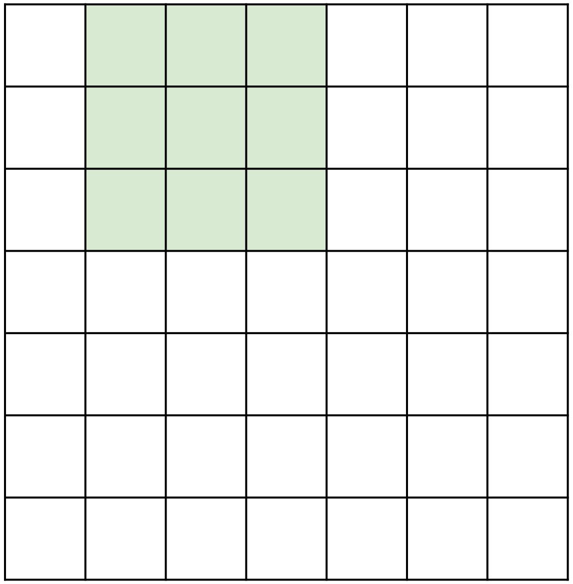


Replicate this column of hidden neurons across space, with some stride.



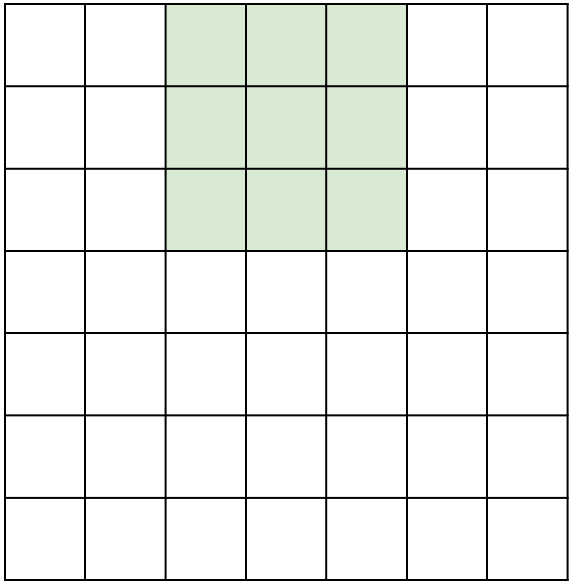
7x7 input
assume 3x3 connectivity, stride 1

Replicate this column of hidden neurons across space, with some stride.



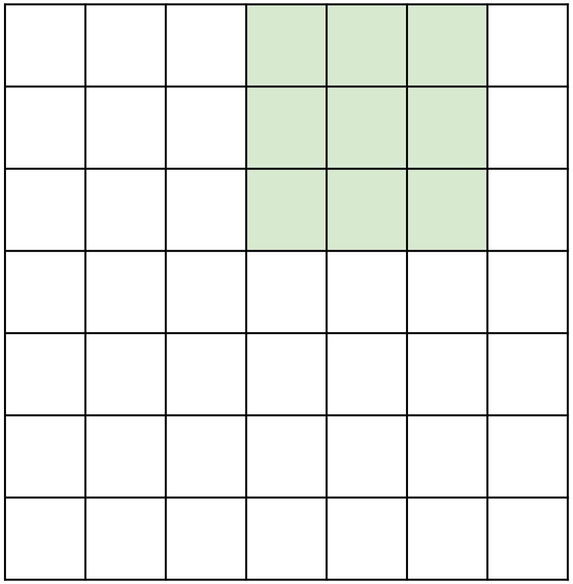
7x7 input
assume 3x3 connectivity, stride 1

Replicate this column of hidden neurons across space, with some stride.



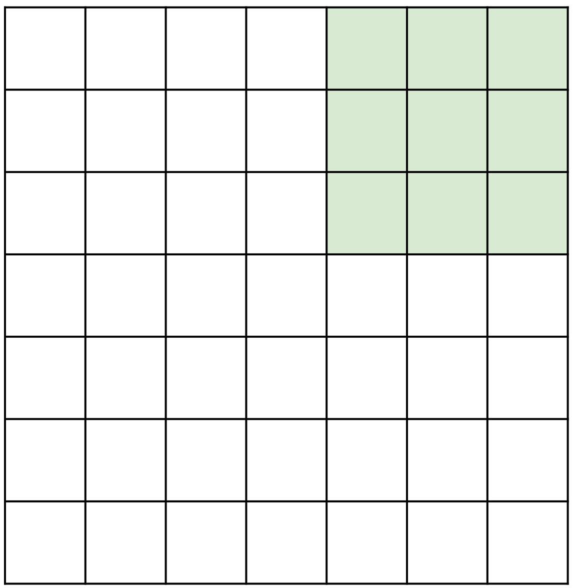
7x7 input
assume 3x3 connectivity, stride 1

Replicate this column of hidden neurons across space, with some stride.



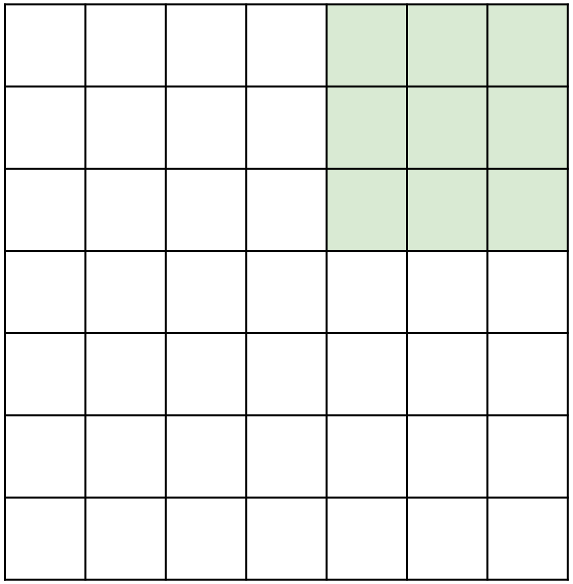
7x7 input
assume 3x3 connectivity, stride 1

Replicate this column of hidden neurons across space, with some stride.



7x7 input
assume 3x3 connectivity, stride 1
$\Rightarrow$ **5x5 output**

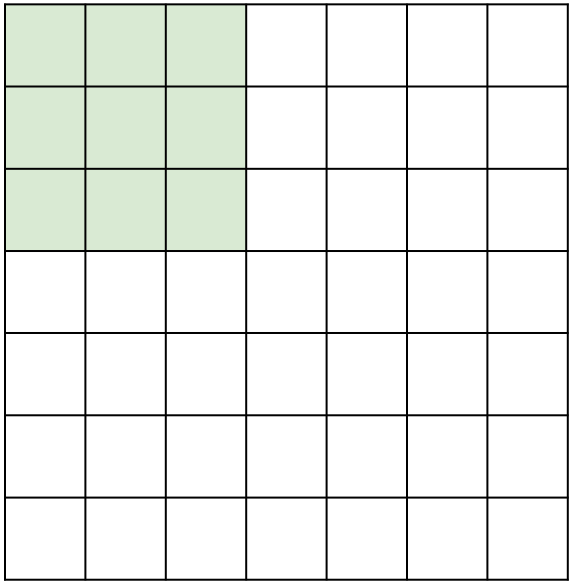
Replicate this column of hidden neurons across space, with some stride.



7x7 input
assume 3x3 connectivity, stride 1
$\Rightarrow$ **5x5 output**

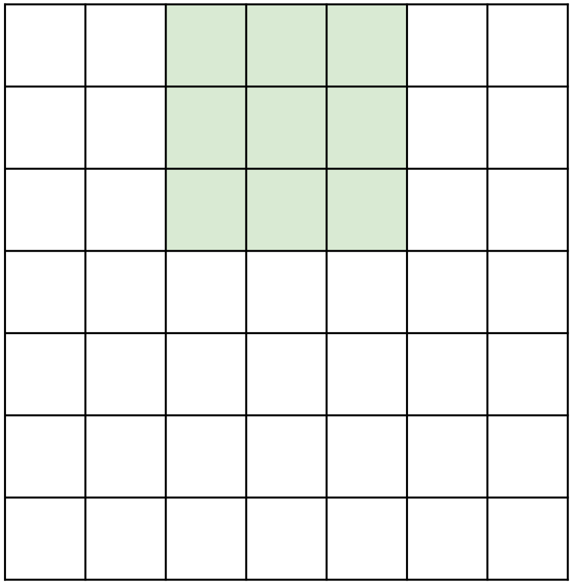
what about stride 2?

Replicate this column of hidden neurons across space, with some stride.



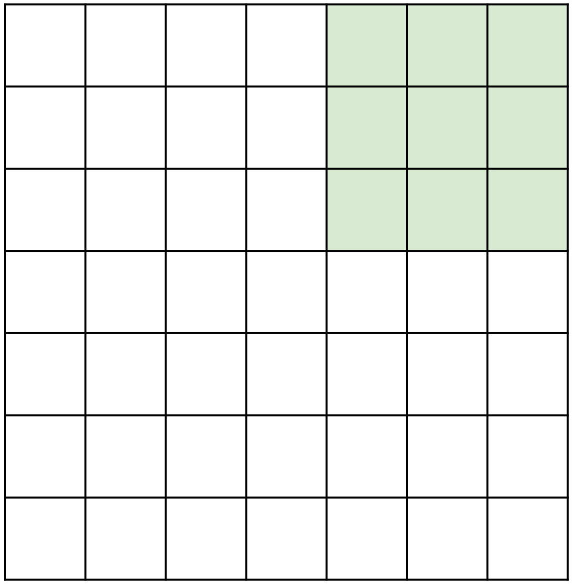
7x7 input
assume 3x3 connectivity, stride 2

Replicate this column of hidden neurons across space, with some stride.



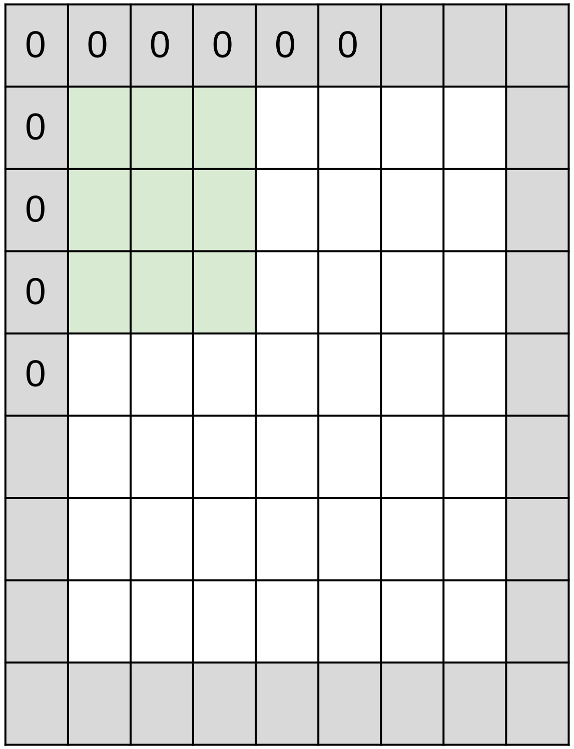
7x7 input
assume 3x3 connectivity, stride 2

Replicate this column of hidden neurons across space, with some stride.



7x7 input
assume 3x3 connectivity, stride 2
$\Rightarrow$ **3x3 output**

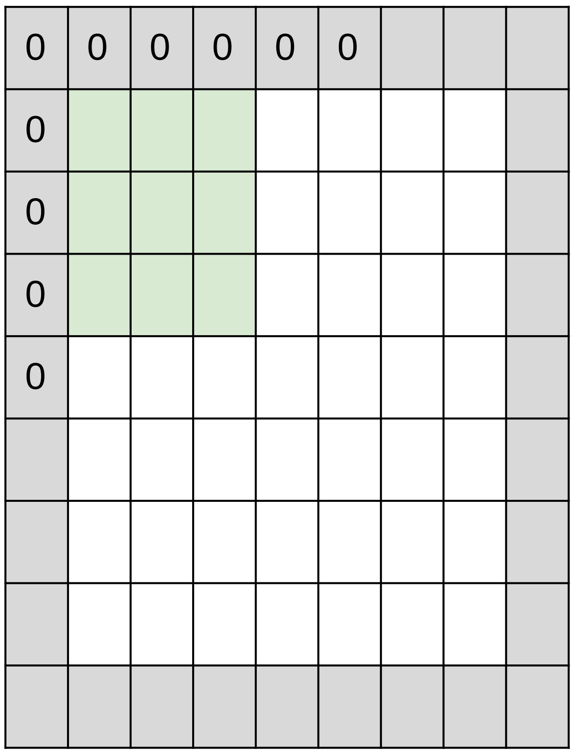
In practice: Common to zero pad the border



(in each channel)
e.g. input 7x7
neuron with receptive field 3x3, stride 1
pad with 1 pixel border

what is the output?

In practice: Common to zero pad the border



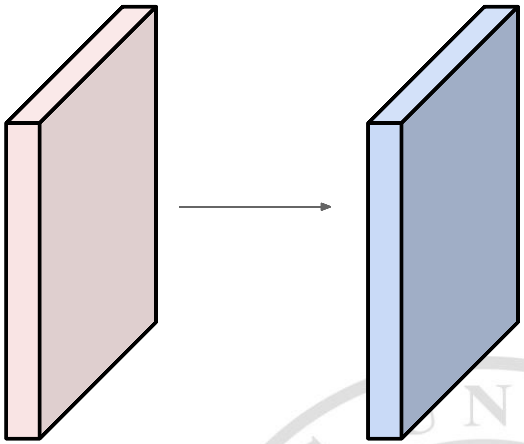
(in each channel)
e.g. input 7x7
neuron with receptive field 3x3, stride 1
pad with 1 pixel border

what is the output?
7x7 => preserved size!

Example

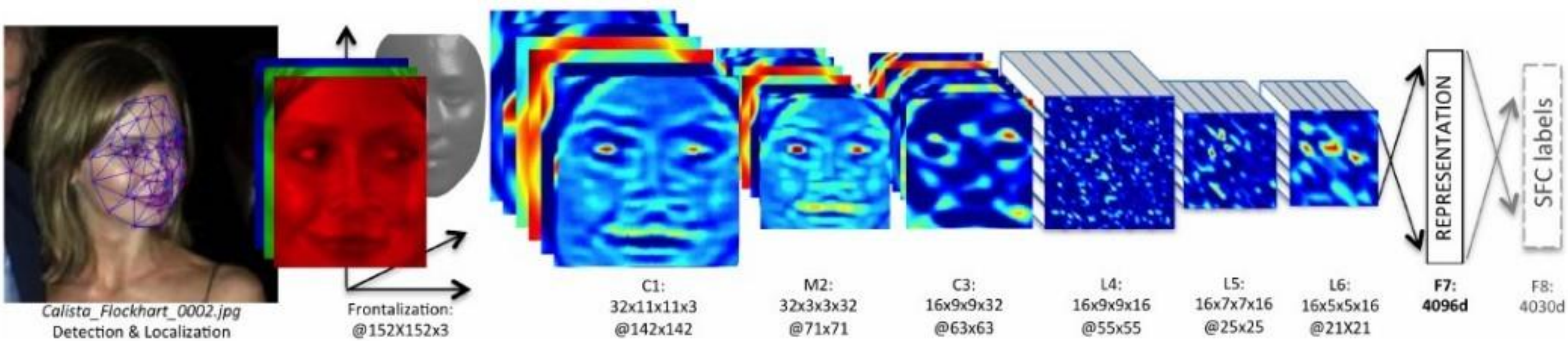
Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2



Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so **32x32x10**

Learning Multi-Layer Features

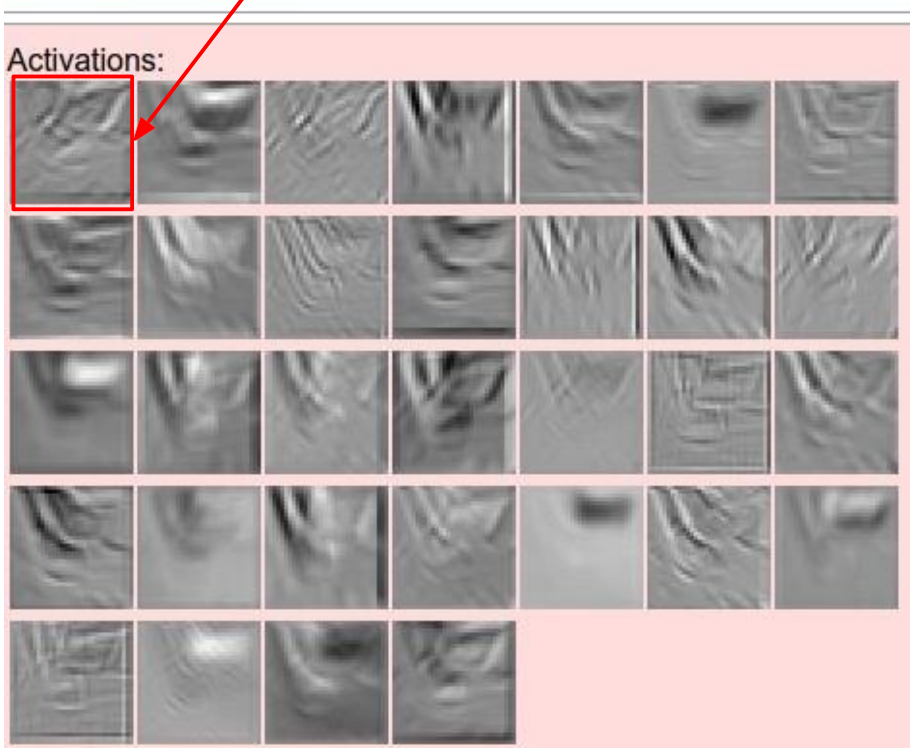


We'd like to be able to learn different things at different spatial positions

Learning Multi-Layer Features

- one filter = one depth slice (or activation map)

5x5 filters



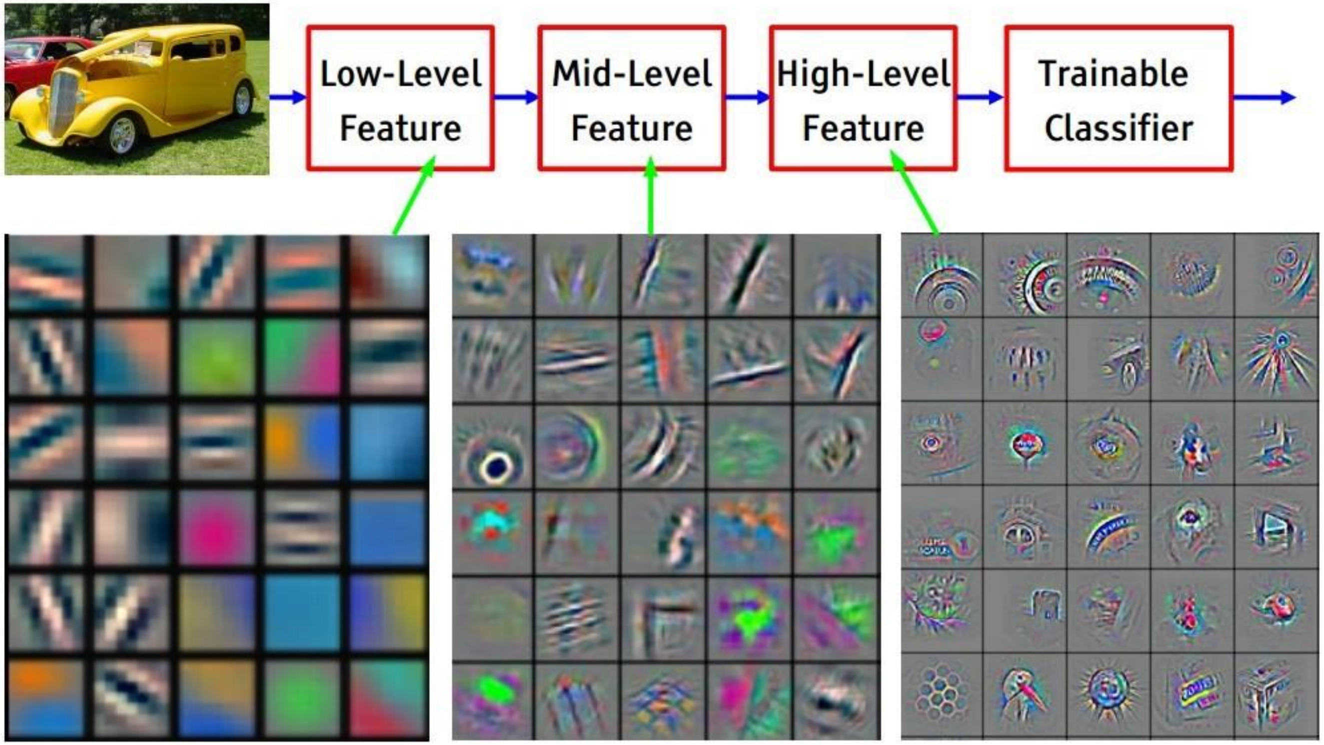
Can call the neurons “filters”

We call the layer convolutional because it is related to convolution of two signals (kind of):

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

elementwise multiplication and sum o
a filter and the signal (image)
= np.dot(w,x) + b

Learning Multi-Layer Features



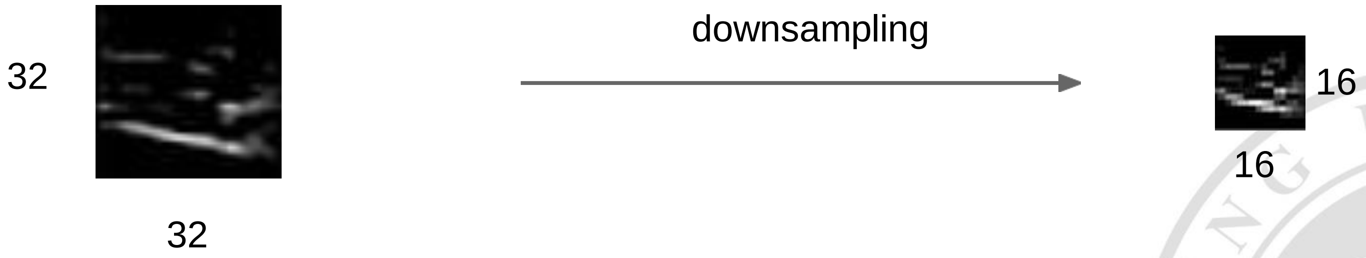
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

[From recent Yann LeCun slides]

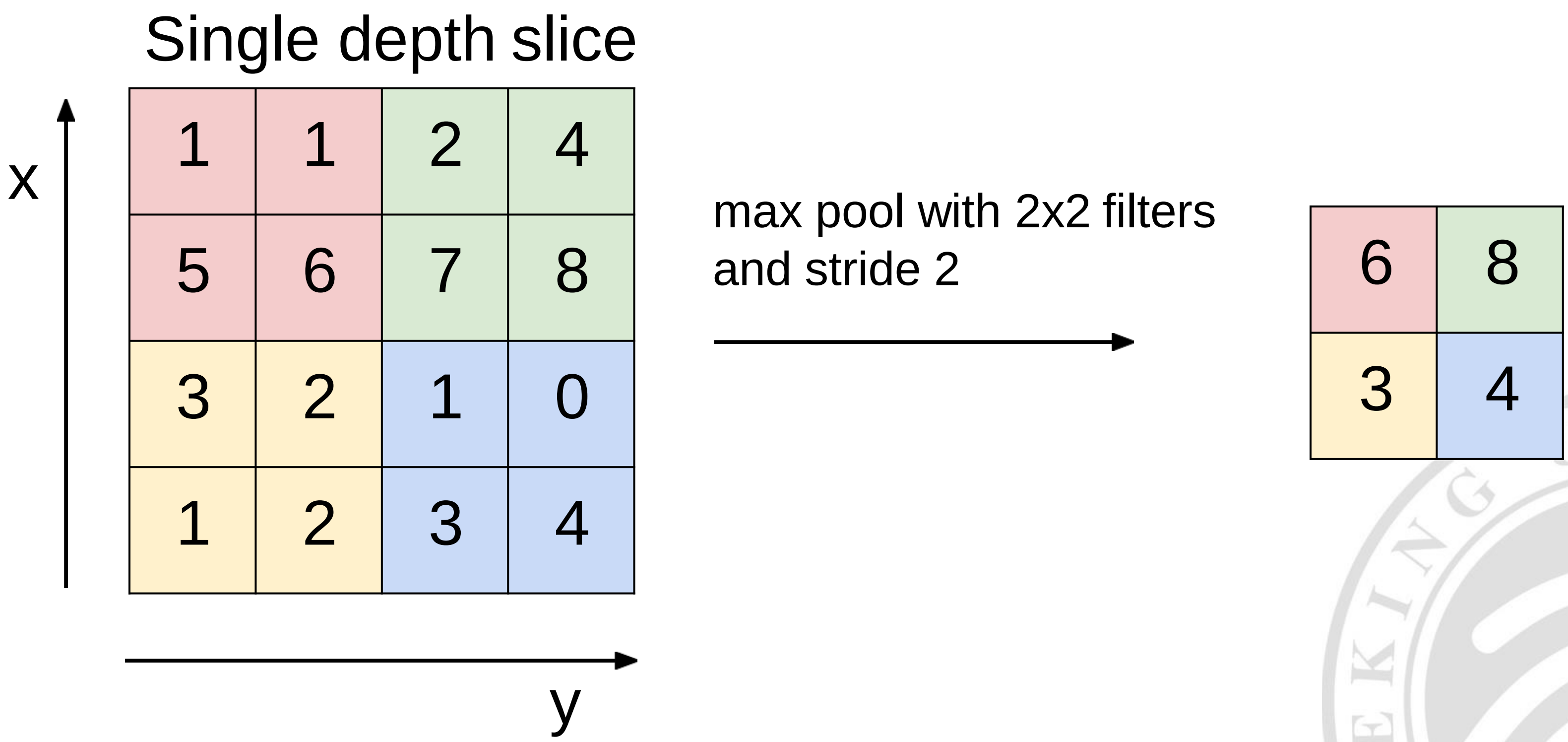
Learning Multi-Layer Features

Pool Layers

- Convenience layer: makes the representations smaller and more manageable without losing too much information
- Computes MAX operation (most common)

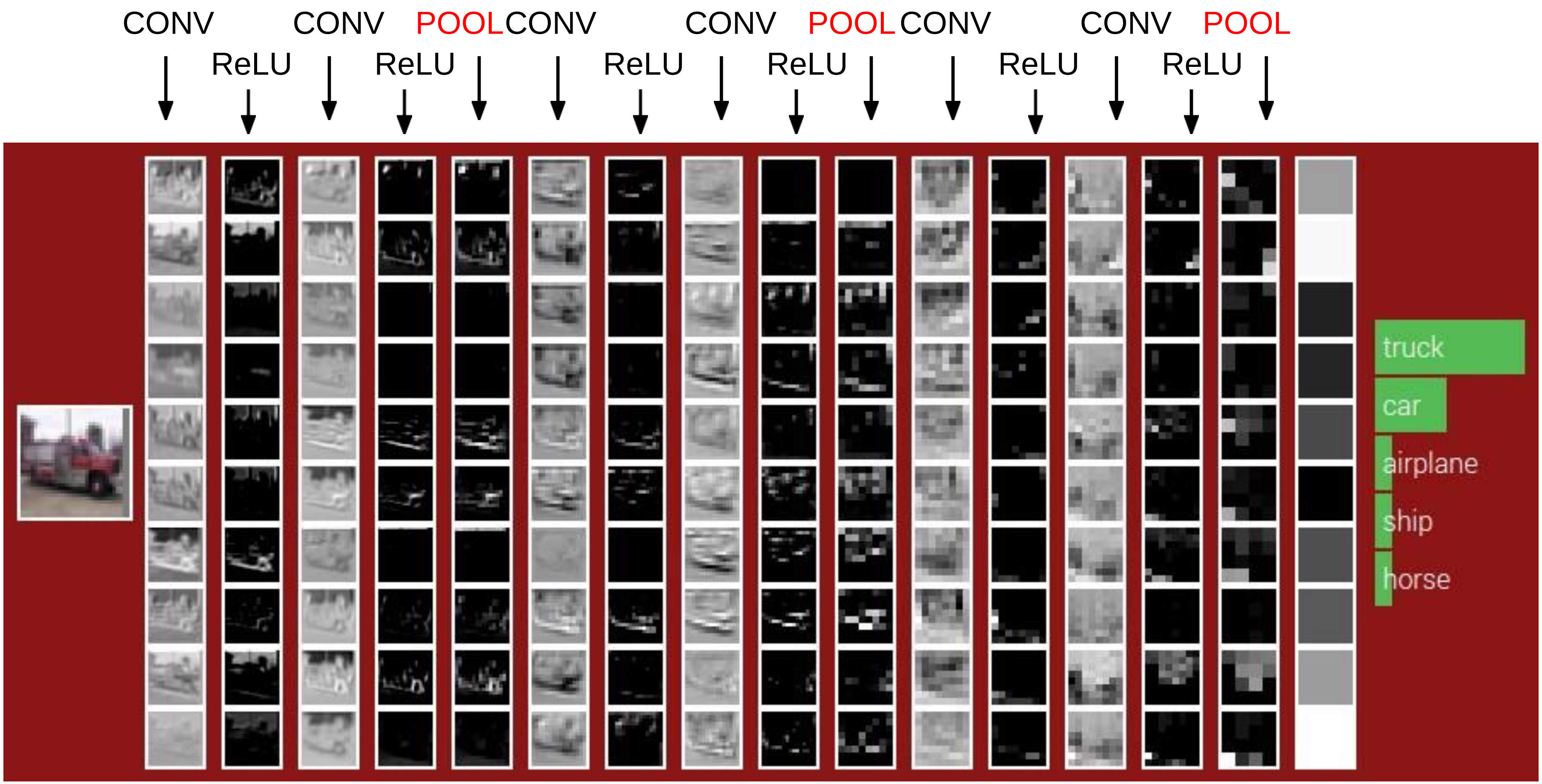


MAX Pooling



Single depth slice
x
y
1
1
2
4
5
6
7
8
3
2
1
0
1
2
3
4
max pool with 2x2 filters
and stride 2
6
8
3
4

Pool Layers


CONV
ReLU
CONV
ReLU
POOL
CONV
ReLU
CONV
ReLU
POOL
CONV
ReLU
CONV
ReLU
POOL
truck
car
airplane
ship
horse