Lecture 7

COGNITIVE COMPUTING

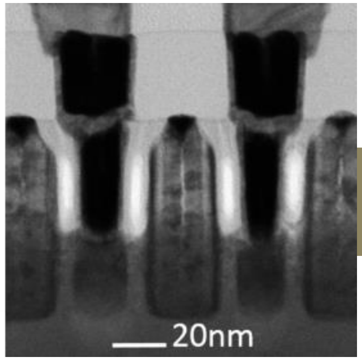
Outline

- Cognitive computing
  - Why and what is it?
- Neural network structure
  - The brain
  - Neuron models
  - Network models
- Learning in Neural networks
  - Gradient descent and backpropagation
  - Deep learning
- Hardware for Neural networks

- Hand-in assignment
- Oral exams

Recap on course

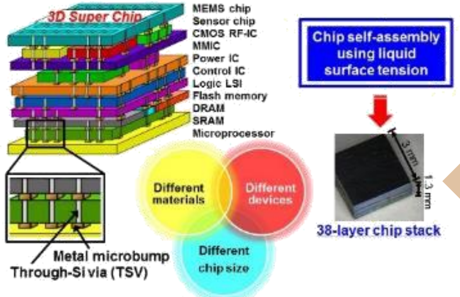
Traditional CMOS



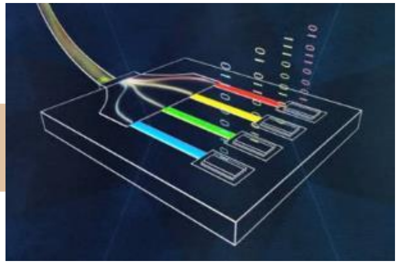
Dennard scaling
"smaller, faster, cheaper"

More than Moore applications

"Versatile, efficient, connected"

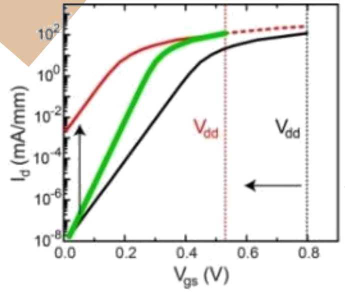


3D circuits and heterogeneous materials

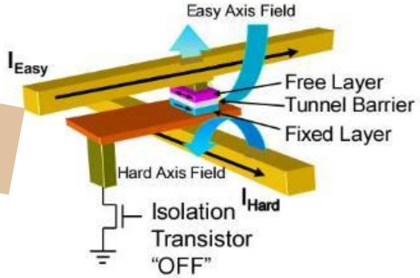


Optical interconnects

Integrated sensors



Low-power



New types of Memory

Why cognitive computing?

- Peta-bytes of constant data generation
  - Mostly unstructured
- Ubiquitous computing power

”Versatile, efficient, connected”

How to handle the data flow and utilize this power?

We need cognitive systems that:

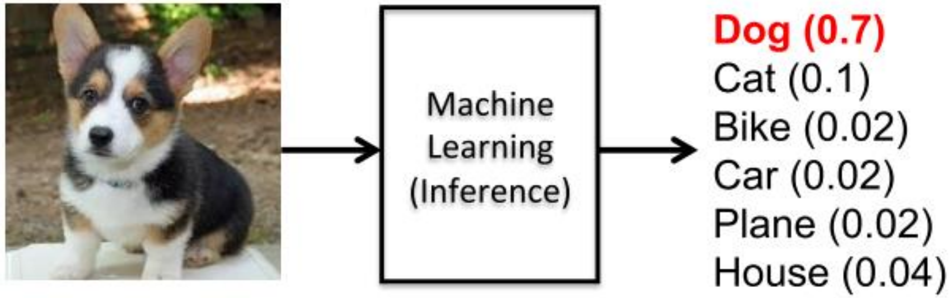
1. Identify data patterns
2. Identifies anomalies in data
3. Can find optimal decision based on data

Pattern recognition

- **Image recognition**
  - Computer vision → data from images, surveillance, autonomous cars,...

- **Speech recognition**
  - Personal assistants, surveillance
- **Language**
  - Data mining**,** translation

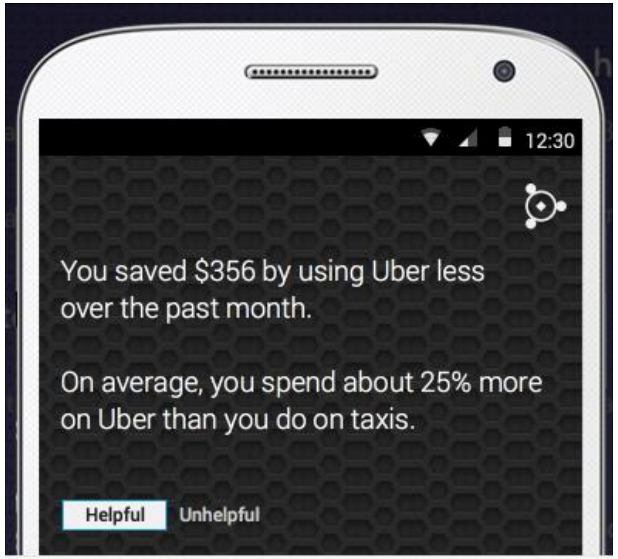
- Unstructured → Structured data



Decision making

- **Health care**
  - Diagnosis
  - Identifying individual treatments for patients
  - Drugs: Choosing optimal trials to reduce time to market

- **Finance**
  - Stock analysis:
    - Analyse earnings statements, news reports and regulatory filings looking for clues on how to view a stock.
  - Personal Financial Advisor
    - Analyse what impacts personal economy and suggest solutions



Wallet.ai

Anomaly detection

Monitor processes and detect issues/trends early on and alert/make decision

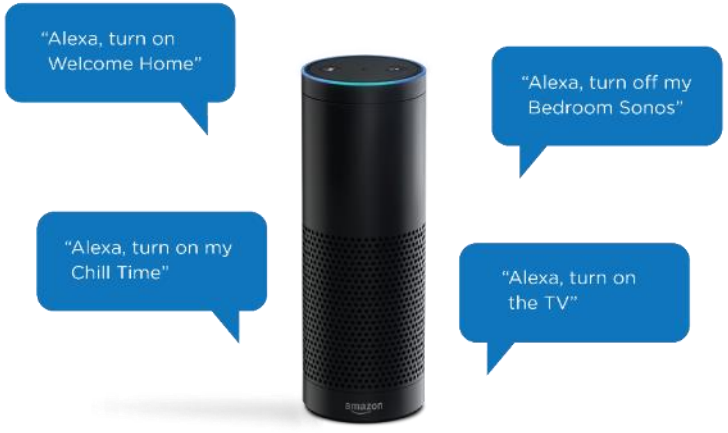
- **Finding illegal behavior**
  - Observe traffic (monetary, wares, transactions)
- **Failures**
  - Monitor sensor status and detect irregularities before break-down
- **Health care**
  - Monitoring of ECG/EEG signals can detect cardiovascular deceases, early onset of seizure in epilepsia

Who is working on it?

IBM (Watson)

Data mining
Health
IoT
Business
...

Nvidia
GPU AI Hardware...



Visual, Speech, Text recognition
Smarter search
Knowledge mining

Personal Assistant
Smarter search
Autonomous driving
...

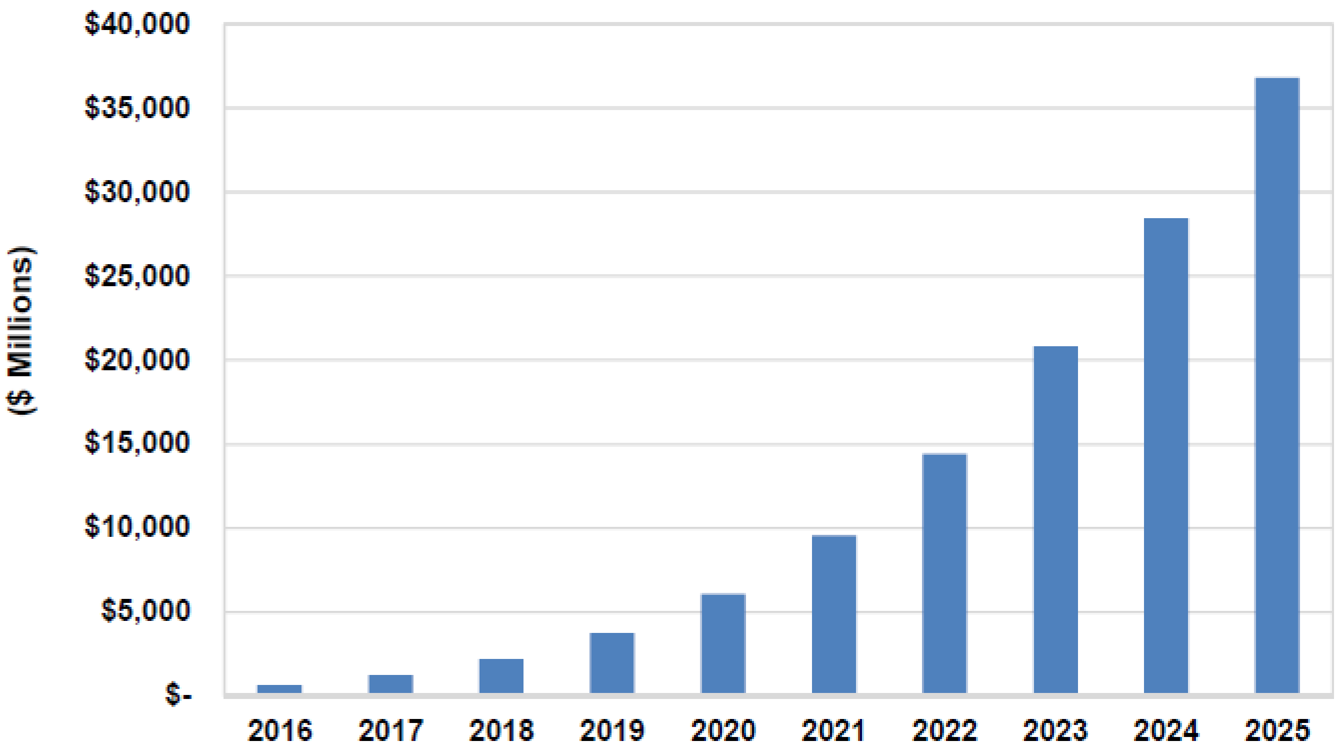
Amazon Alexa
”Personal assistent”

Giant new industry

NANO ELECTRONICS GROUP

AI has applications and use cases in almost every industry vertical and is considered the **next big technological shift**, similar to past shifts like the industrial revolution, the computer age, and the smartphone revolution

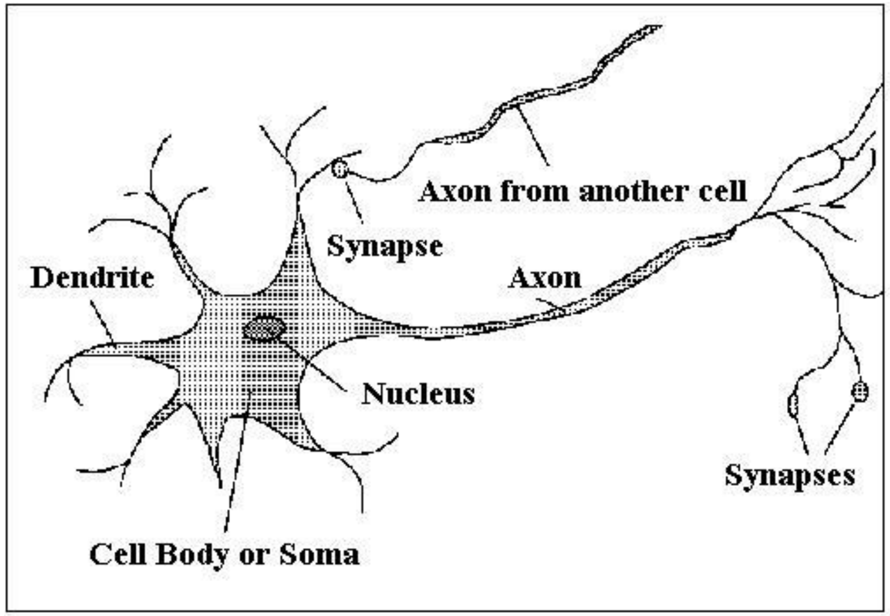
Chart 1.1 Artificial Intelligence Revenue, World Markets: 2016-2025

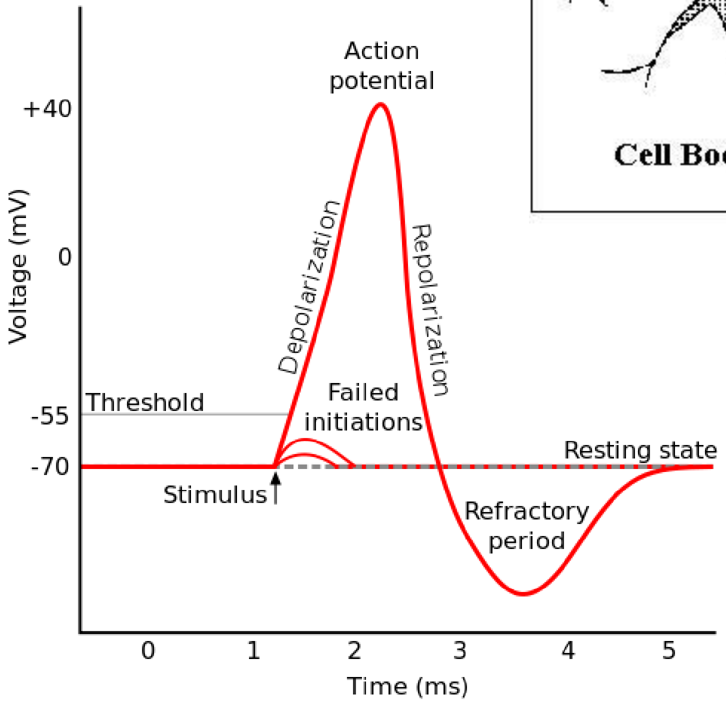


(Source: Tractica)

How the brain works (”much simplified”)

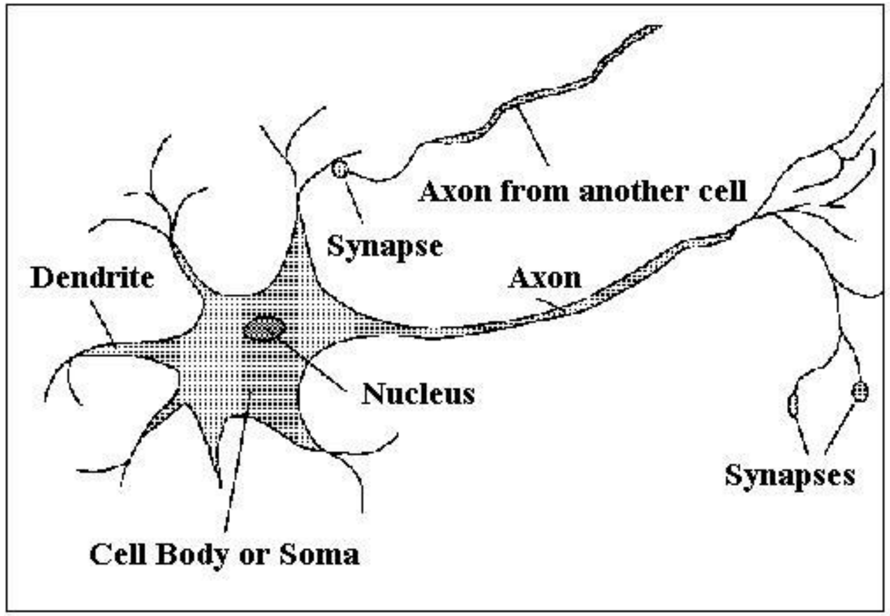
- Neurons ~ 10^{11}
- Connected by synapses with varying ”resistance”/”weight”
  ~ 10^{15} synapses
- Electrical stimuli above threshold close in time causes them to fire a signal
- Signals propagate through network
- Connections encode logic/memory
- Hierarcial ”layered” structure

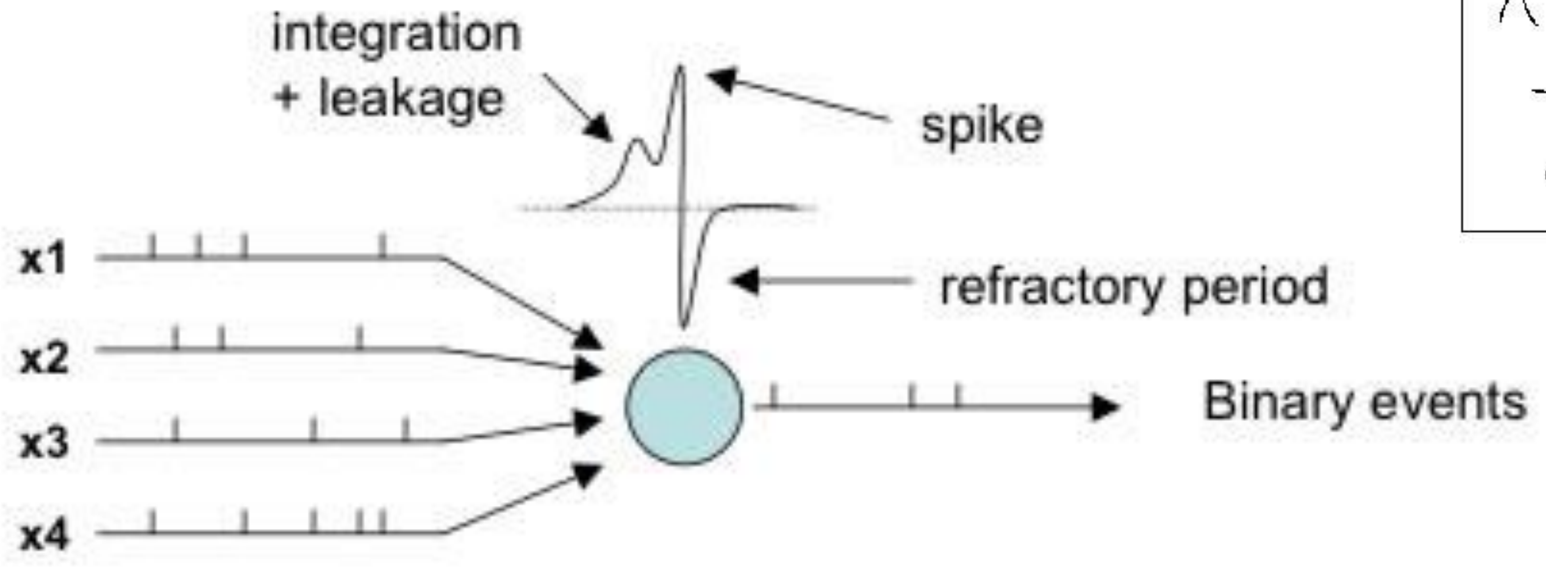




Spiking neurons

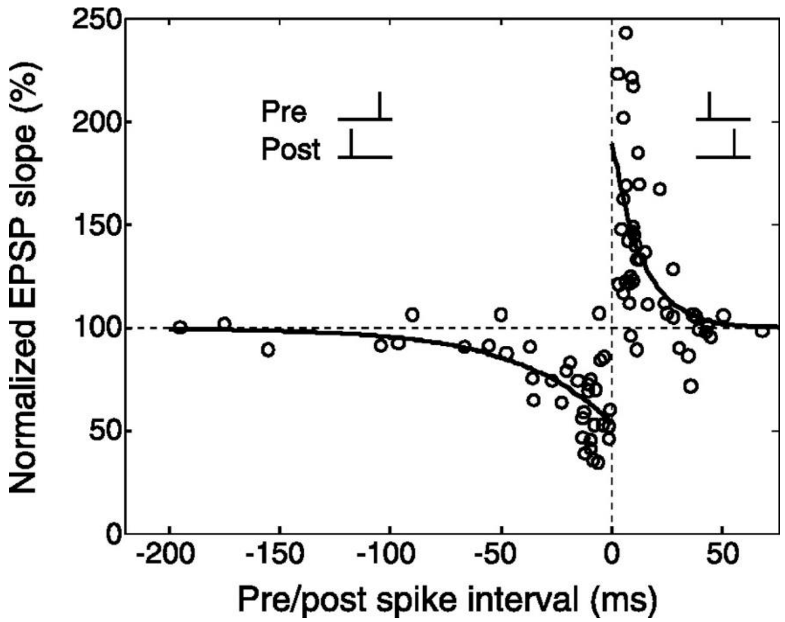
- Input signals in dendrites are integrated in the neuron
- Many inputs in short time interval
  → a threshold is overcome
  → the neuron will fire a signal into the axon

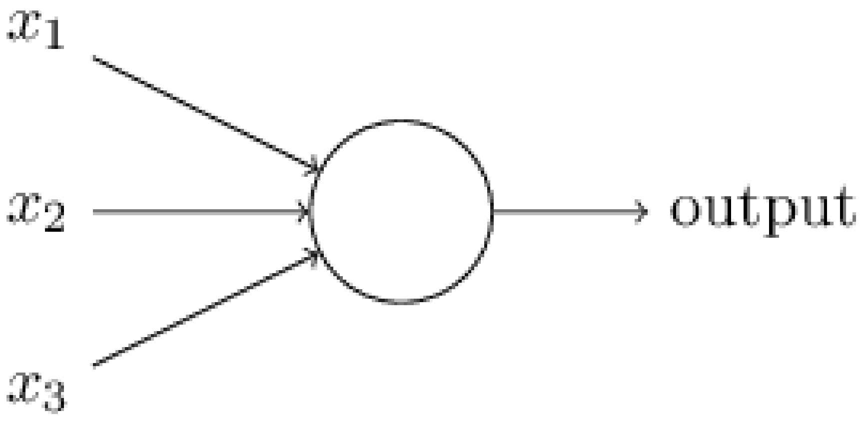


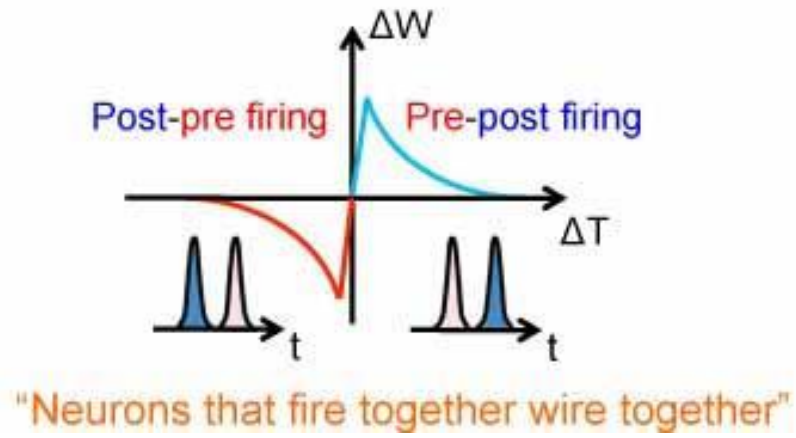


Spike-timing dependent plasticity

- How the brain "learns"
- Inputs prior to neuron fires are strengthened
- Inputs after the neuron fires are weakened
- Neurons tend to fire when many inputs occur at the same time (threshold) → Subset of correlated inputs remain



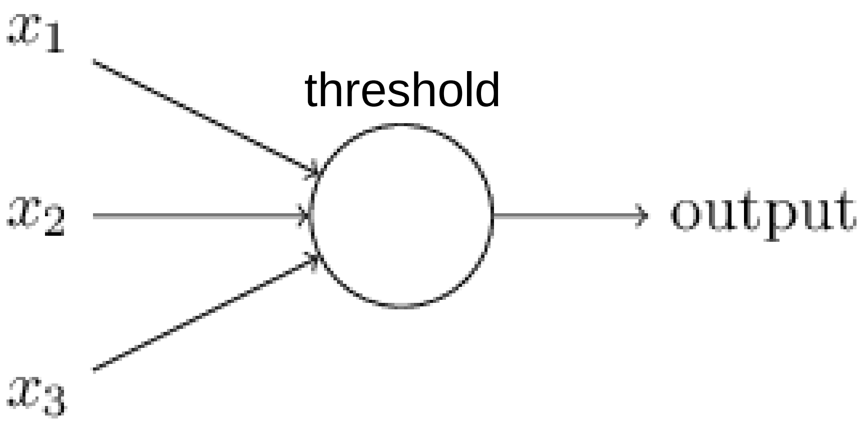




The perceptron

- A simplified representation of a neuron
  - A number of inputs, one output
  - Threshold determines fire/not.
    - Also called *bias*

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

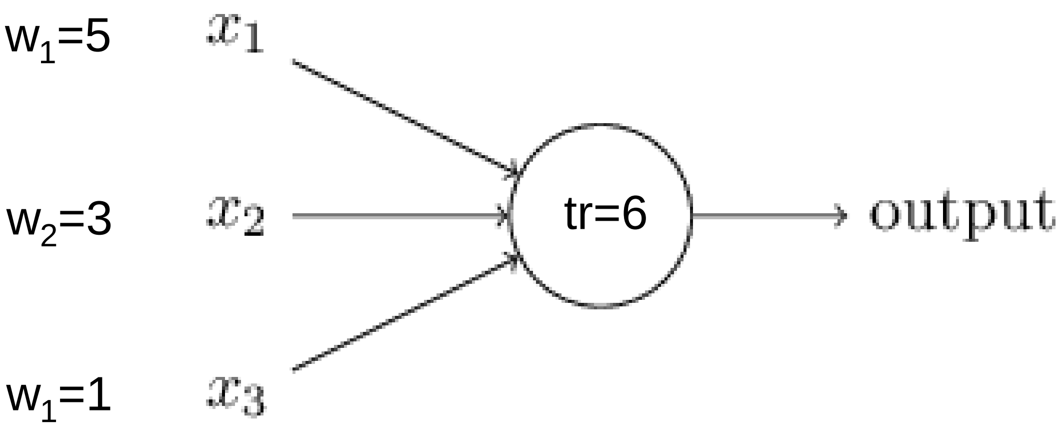


Perceptron example

Cheese festival in another town and you love cheese

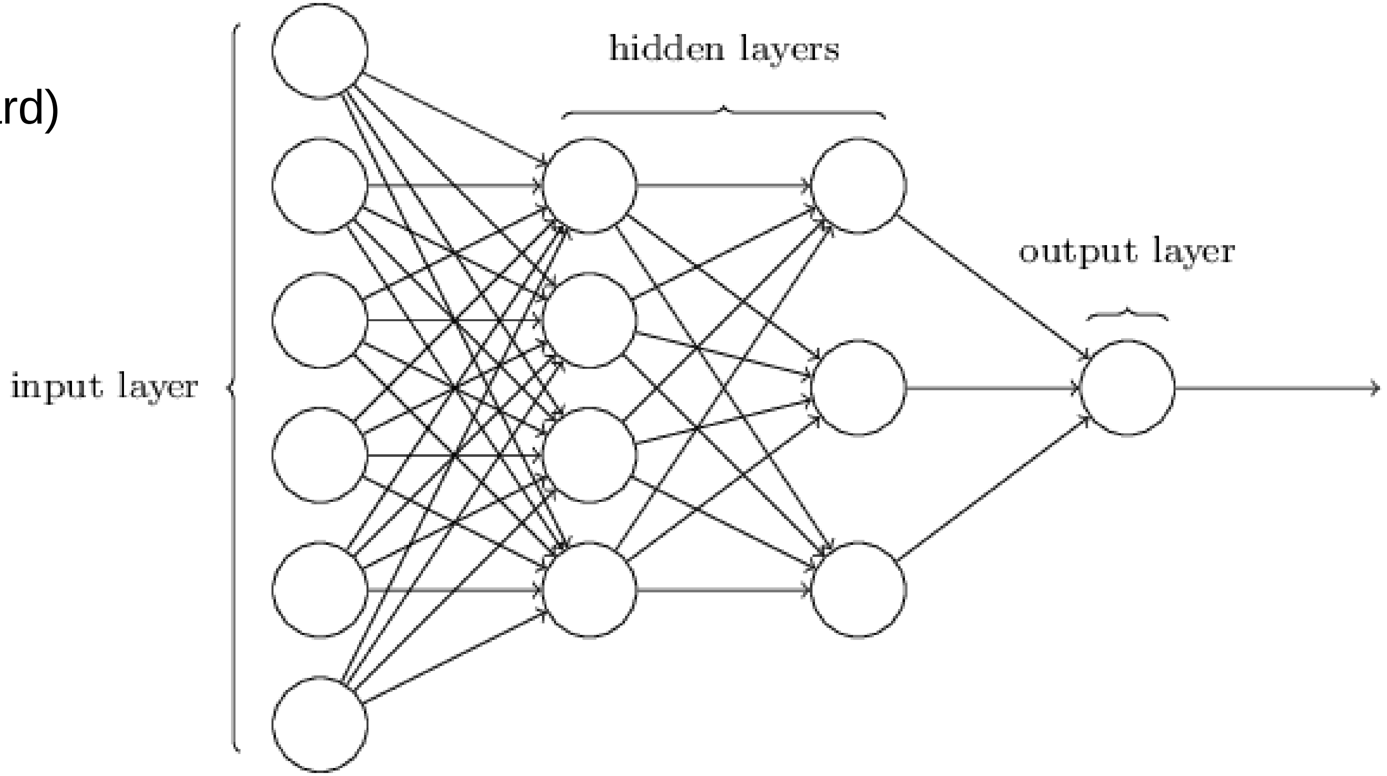
But should you go there? Three factors to decide:

1. Is the weather good?
2. Does your boyfriend/girlfriend want to join?
3. Is it easy to get there?



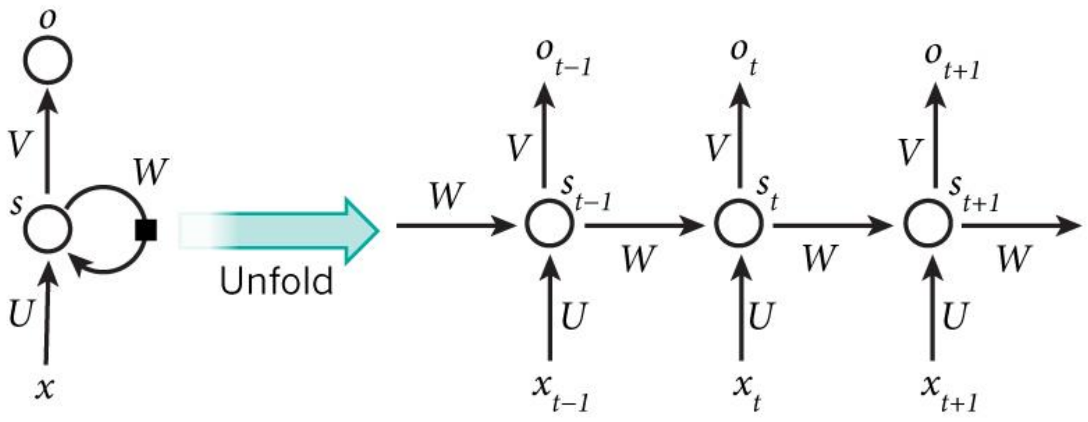
Layered network

- First layer – Input layer
- Last layer – Output layer
- Intermediate layers – Hidden layers
  - For more subtle and sophisticated ”decision making”
- Connections usually go forward (feed-forward)

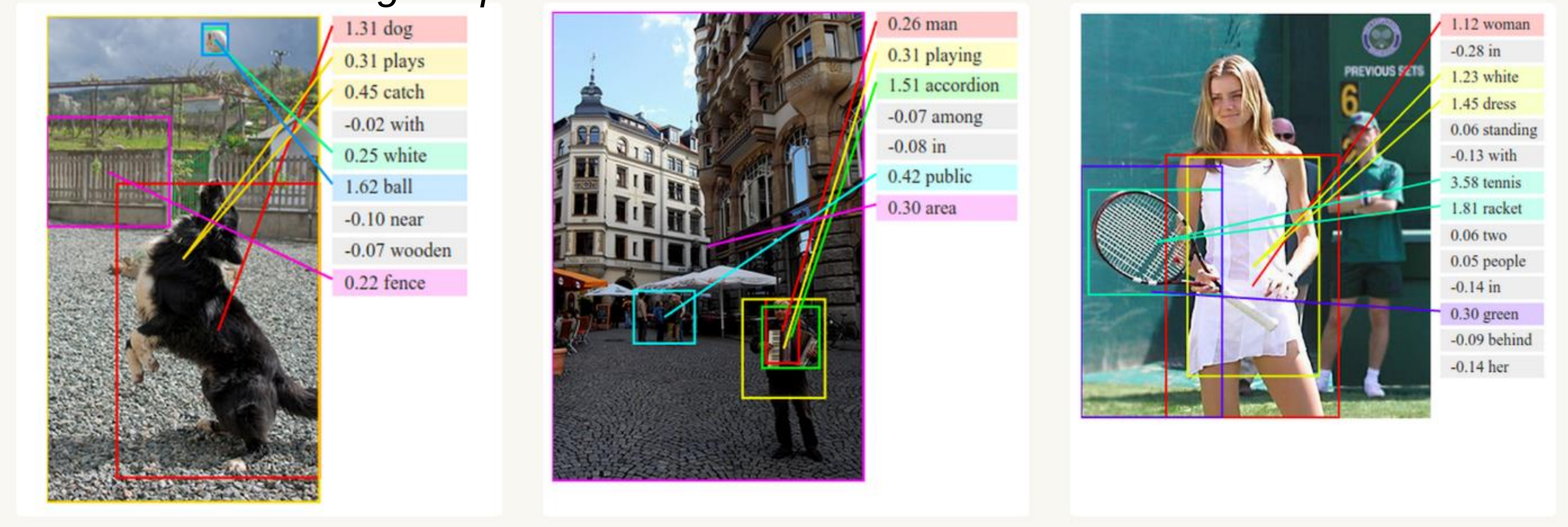


Recurrent neural networks

- Allowing for connections within and to previous layers
- The network obtains ”memory” of previous states
- Useful for analysing flows of temporally connected data
  - Speech, language, ...
- Best implemented with spiking neurons

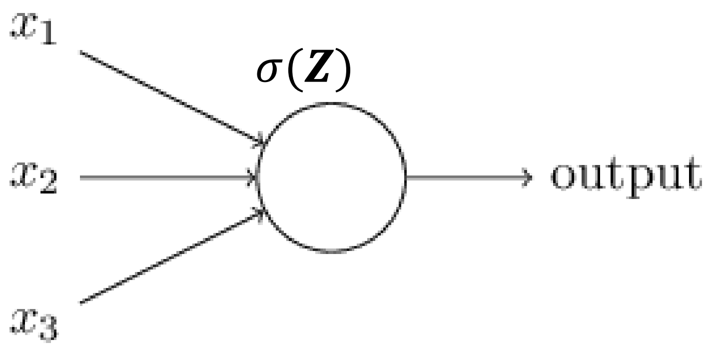


Generation of image caption text

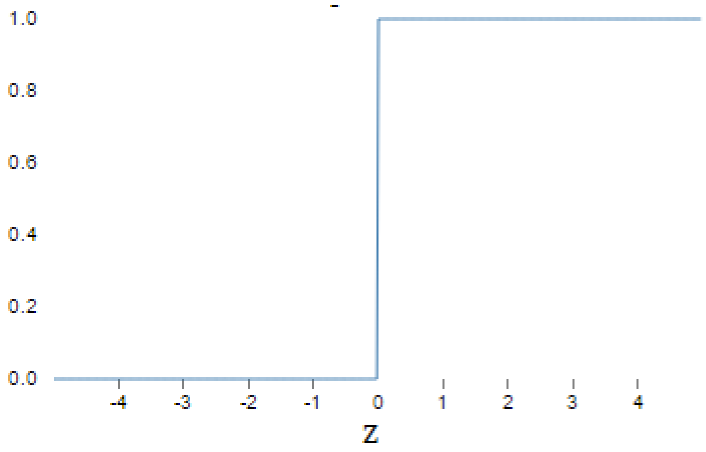


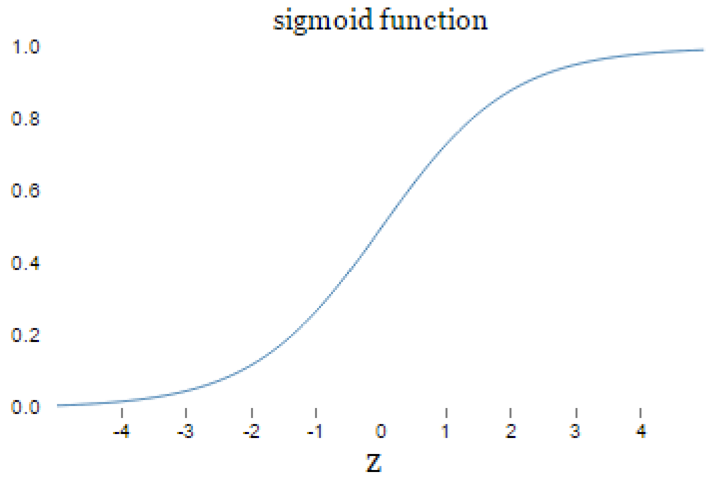
Sigmoid neuron

- Smooth activation → To facilitate learning algorithms
- Sigmoid function:
- $\sigma(\boldsymbol{Z}) = \frac{1}{1+\exp(-\boldsymbol{Z})}$



perceptron





Supervised Learning

- Idea:
  - Feed network with known examples x that should give output y(x)
  - Look at the outcome and define how good was the result
    - The cost function
      A smooth function in terms of weights and thresholds to evaluate how well trained the network is
  - Adjust the weights (**w**) and biases (**b**) to improve
  - Repeat until good enough

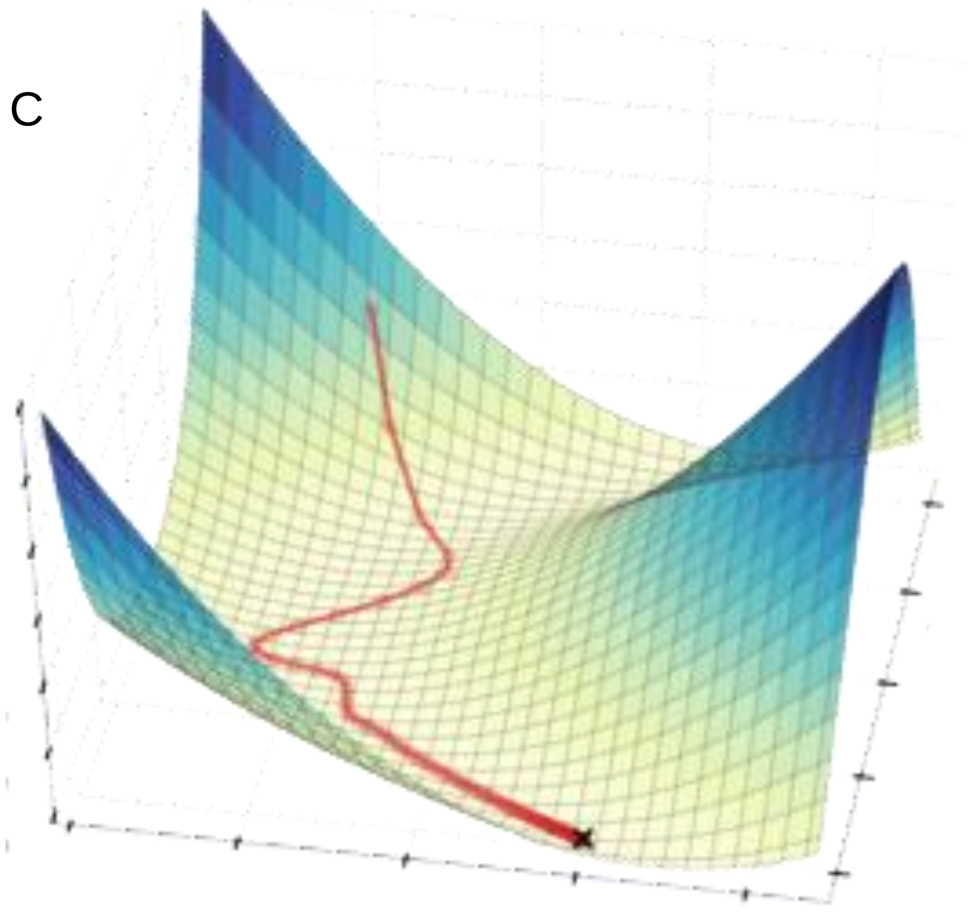
- Quadratic cost function
  - $C(w,b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$
    - y(x) – correct output, a – output of network, n – nbr of training inputs

Gradient Descent Algorithm

- A general method to minimize a multivariable function (cost function)
- Change parameters *v* in the direction of maximum gradient of C
- $\nabla C = \left(\frac{\partial C}{\partial v_1}, \ldots \frac{\partial C}{\partial v_m}\right)^T$
- $\Delta v = -\eta \nabla C$, η is the *learning rate*

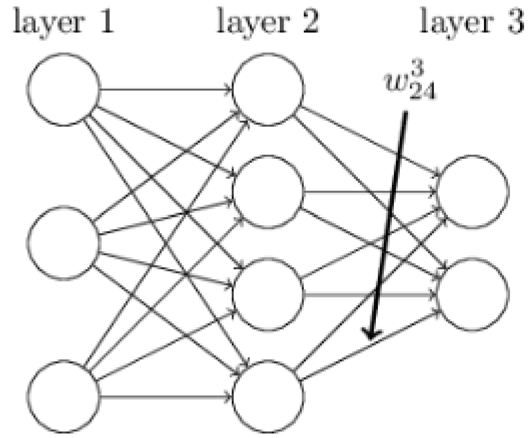
$$w_k \rightarrow w'_k = w_k - \eta \frac{\partial C}{\partial w_k}$$

$$b_l \rightarrow b'_l = b_l - \eta \frac{\partial C}{\partial b_l}.$$



Matrix representation

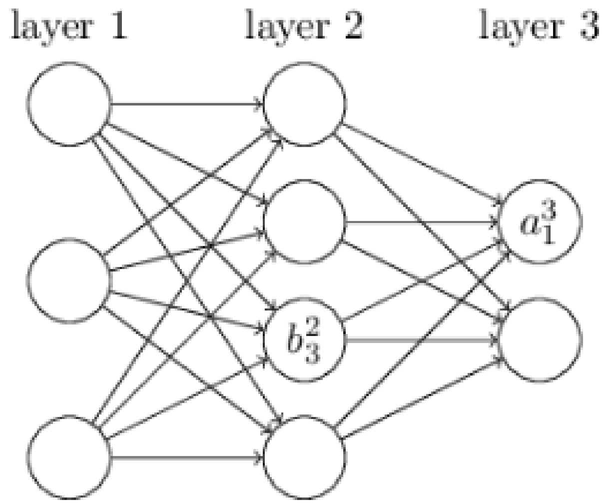
$$\boldsymbol{W}^l = \begin{bmatrix} w_{11} & \cdots & w_{1,n} \\ \vdots & \ddots & \vdots \\ w_{m,1} & \cdots & w_{m,n} \end{bmatrix}$$



w^l_{jk} is the weight from the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer

$$\boldsymbol{A}^l = \begin{bmatrix} a^l_1 \\ \vdots \\ a^l_m \end{bmatrix}$$

$$\boldsymbol{B}^l = \begin{bmatrix} b^l_1 \\ \vdots \\ b^l_m \end{bmatrix}$$



The weighted input of the l^{th} layer:

$$\boldsymbol{Z}^l = \boldsymbol{W}^l \boldsymbol{A}^{l-1} + \boldsymbol{B}^l$$

The activation of the l^{th} layer is thus:

$$\boldsymbol{A}^l = \sigma(\boldsymbol{Z}^l)$$

The error function

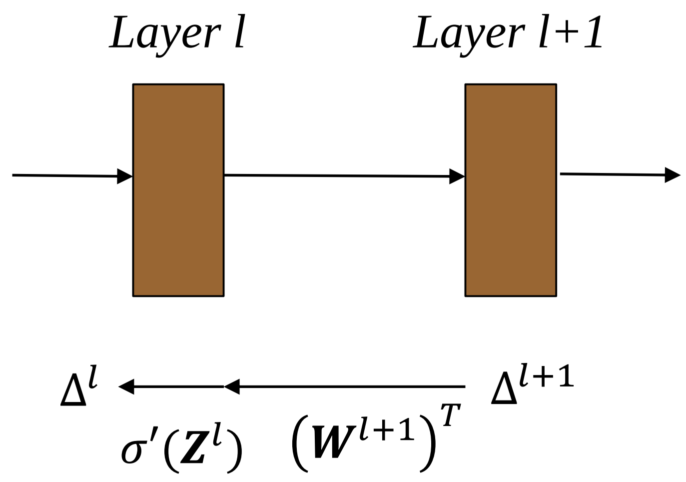
- A way to quantify the error of the estimation due to a change in the network parameters

- The error in the output layer:
- $\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'\left(z_j^L\right) \rightarrow \Delta^L = \nabla_a C .* \sigma'(\boldsymbol{Z}^L)$

How fast the activation changes at z_j

How fast the cost is changing with change of activation at j

- For the previous layers:
- $\Delta^l = \left[\left(\boldsymbol{W}^{l+1}\right)^T \Delta^{l+1}\right] .* \sigma'(\boldsymbol{Z}^l)$

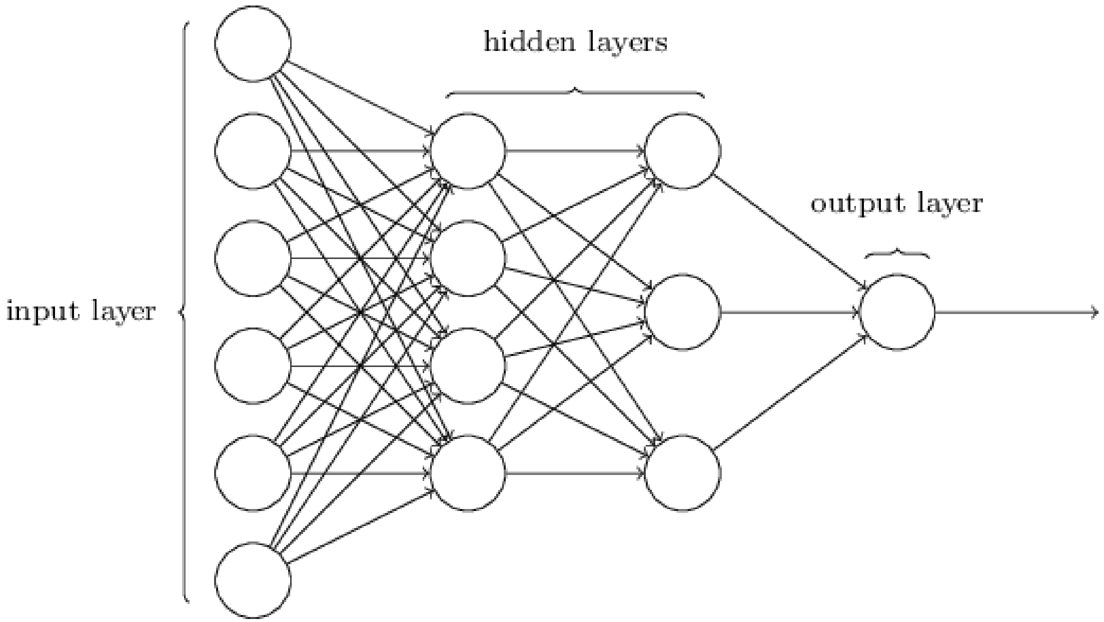


Backpropagation algoritm

- A way to calculate ∇C

1. **Input** *x*: Set the activation $\boldsymbol{A}^l$ at the input layer
2. **Feed-forward**: for each *l* = *2,3,...,L* compute $\boldsymbol{Z}^l$ and $\boldsymbol{A}^l$
3. **Output error:** Compute Δ^L of the output layer
4. **Back-propagate error:** For each layer starting with output, calculate $\Delta^l = \left[\left(\boldsymbol{W}^{l+1}\right)^T \Delta^{l+1}\right] .* \, \sigma'(\boldsymbol{Z}^l)$
5. **Output:** $\frac{\partial C}{\partial \boldsymbol{B}^l} = \boldsymbol{\Delta}^{\mathrm{l}}, \frac{\partial C}{\partial \boldsymbol{w}_{jk}^l} = a_k^{l-1} \delta_j^{\mathrm{l}}$

A¹



Improving speed at calculating ∇C

- Need to compute the gradient for each training example x and then average
  - $C = \frac{1}{n}\sum_x C_x \rightarrow \nabla C = \frac{1}{n}\sum_x \nabla C_X$
- With n > 10000 this is extremely slow

→ **<u>Stoichastic gradient descent</u>**

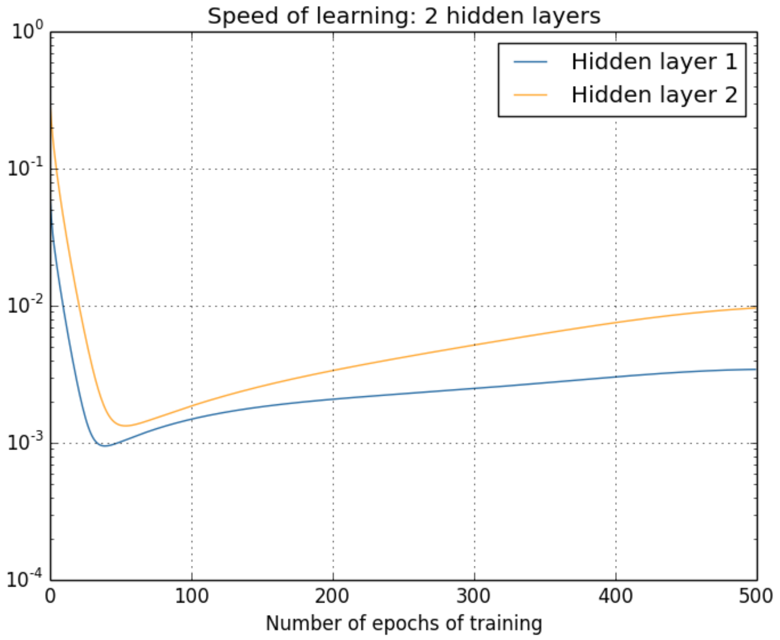
- Randomly choose sample m of x (mini-batch)
- $\nabla C \approx \frac{1}{m}\sum_{j=1}^{m} \nabla C_{x_j}$
- Train on this mini-batch, then choose a new batch until having trained with all (1 epoch)
- Then start over and redo until finished

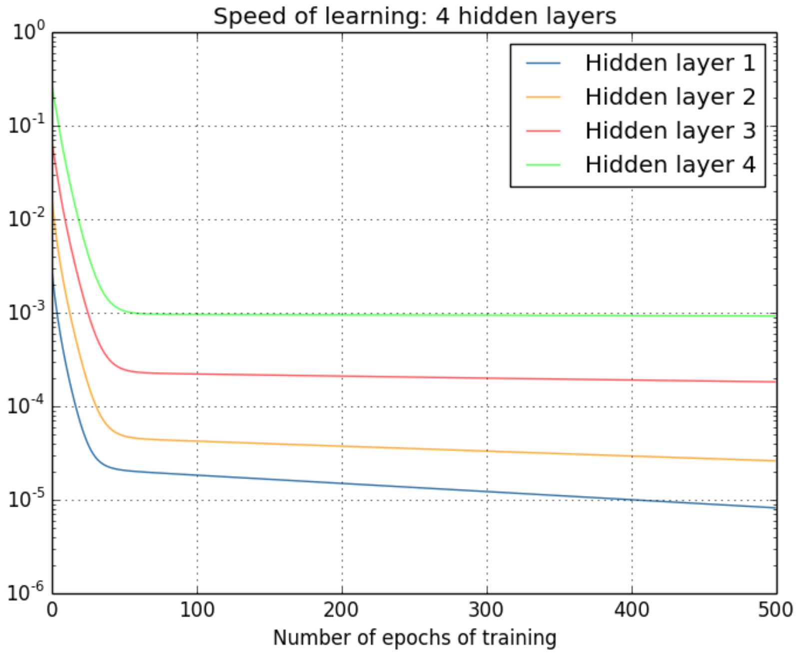
$$w_k \rightarrow w'_k = w_k - \frac{\eta}{m}\sum_j \frac{\partial C_{X_j}}{\partial w_k}$$

$$b_l \rightarrow b'_l = b_l - \frac{\eta}{m}\sum_j \frac{\partial C_{X_j}}{\partial b_l},$$

Deep networks

- More than one hidden layer → much improved abstraction power
- Different layers learn at vastly different speed → backprop and gradient descent works poorly
  - Vanishing gradient problem: Built-in instability with gradient descent techniques.

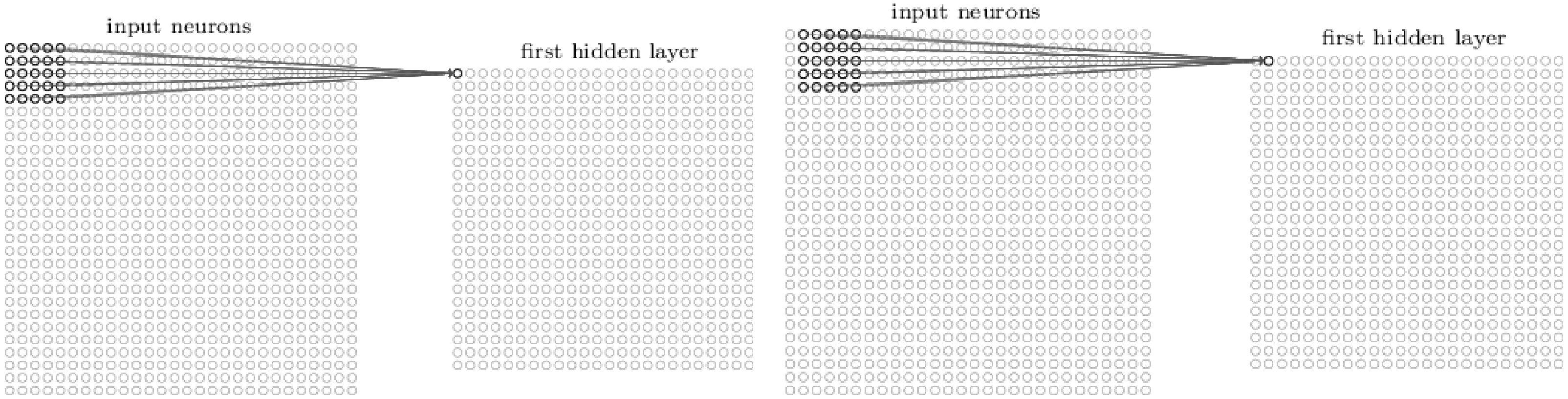




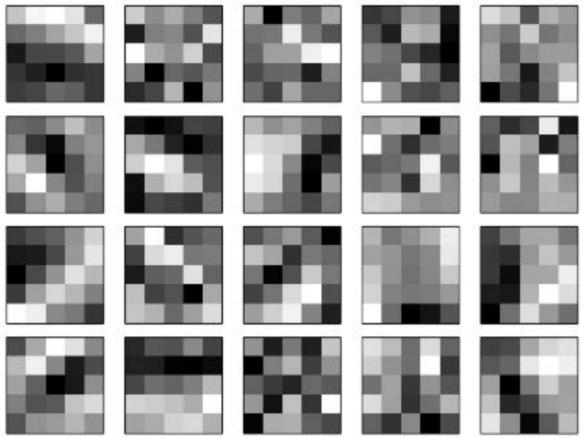
Convolutional deep neural networks

- Utilizing the fact that neighboring inputs are related to each other → great for image analysis
- Create hidden layer: Each neuron connects to a subset of neurons in the previous layer
  - A *local receptive field*
- All neurons in first hidden layer use same *shared bias* and *shared weights*!
  - i.e. This layer can detect the same feature **anywhere** in the image
  - Drastically reduces number of parameters: 5x5+1=26
- A convolutional network uses many parallel feature maps to build up an understanding of images

$$\sigma\left(b+\sum_{l=0}^{4}\sum_{m=0}^{4} w_{l,m}a_{j+l,k+m}\right)$$

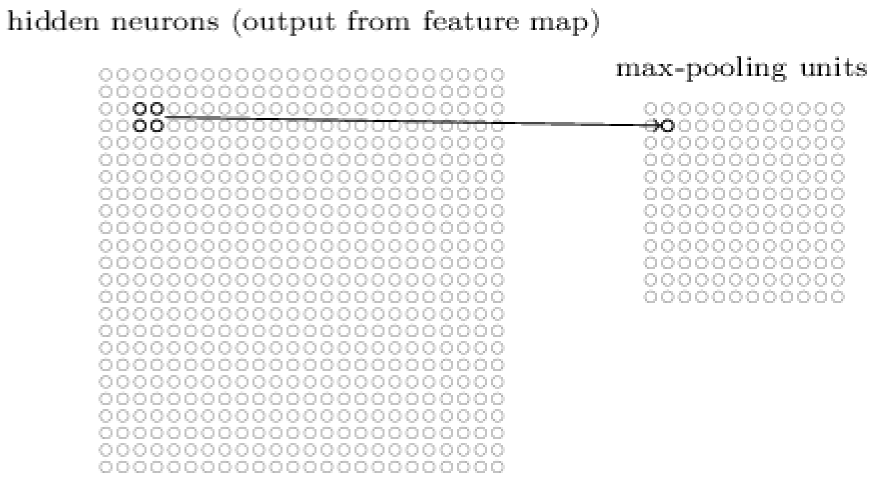


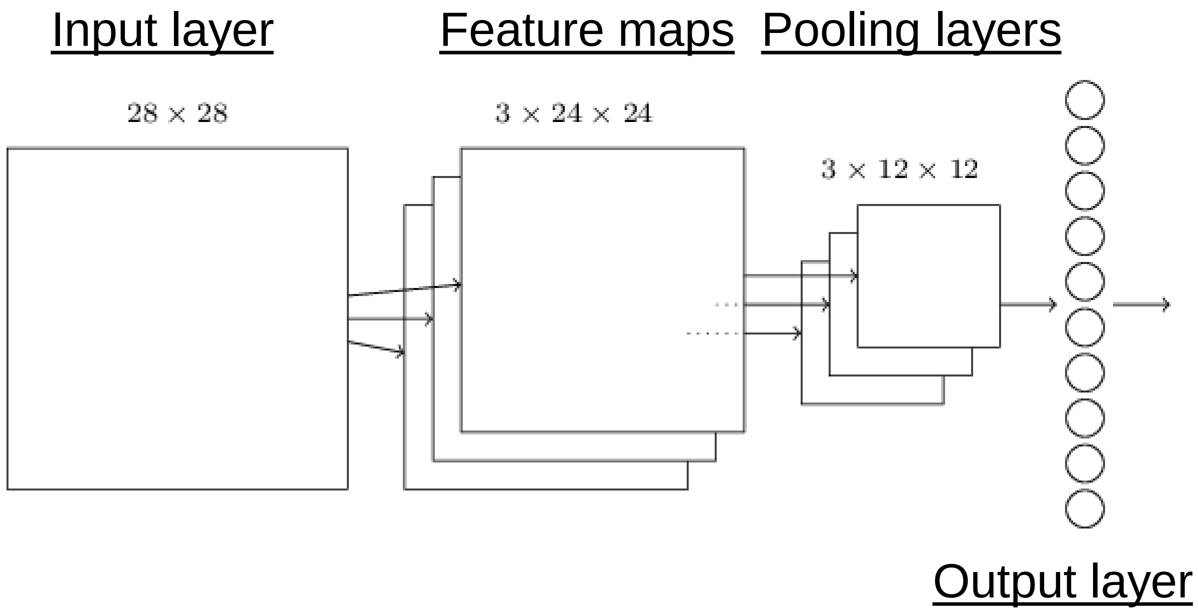
20 feature maps used for recognizing handwritten numbers



Complete Convolutional network

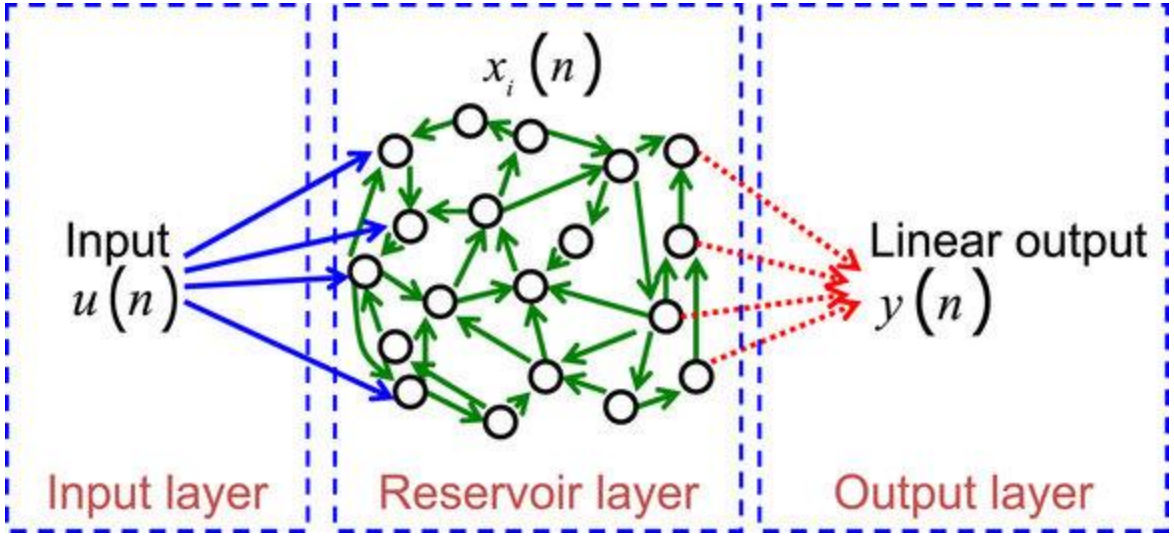
- One pooling layer per feature map
  - To condense information from feature maps
  - Max pooling: activation = the max of the neurons in the connected neurons in feature map
  - L2 pooling: root of the sum of squares
- The Output layer is a normal completely connected layer to all neurons in the pooling layers
- Avoids learning issues by reducing parameters...





Reservoir computing

- Viewed as an extension of neural networks
- Non-linear randomly connected nodes form a reservoir
  - Reservoir nodes and connections are usually <u>constant</u>
- The dynamic behavior of the reservoir creates the logic
- Read out by linear combination of the reservoir output
  - Training by linear regression of the output to known inputs

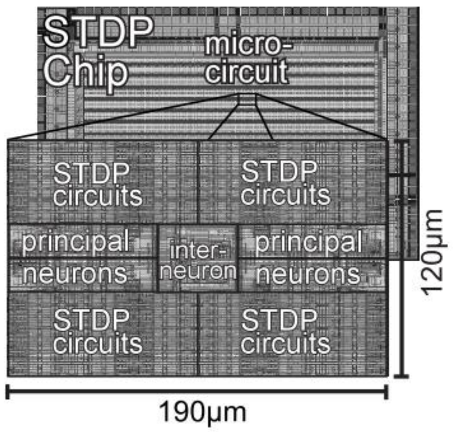


Hardware for Neural networks

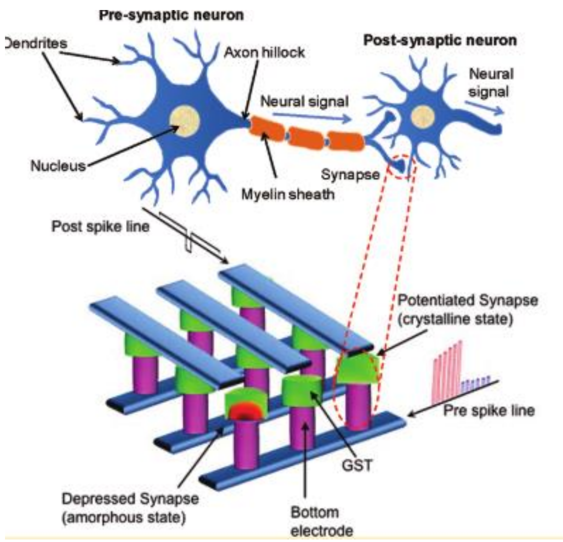
- The brain is very energy efficient 10 W
  - Simulating 5s of brain activity with supercomputer → 500s and 1.4 MW
- Specialized hardware needed to improve energy efficiency

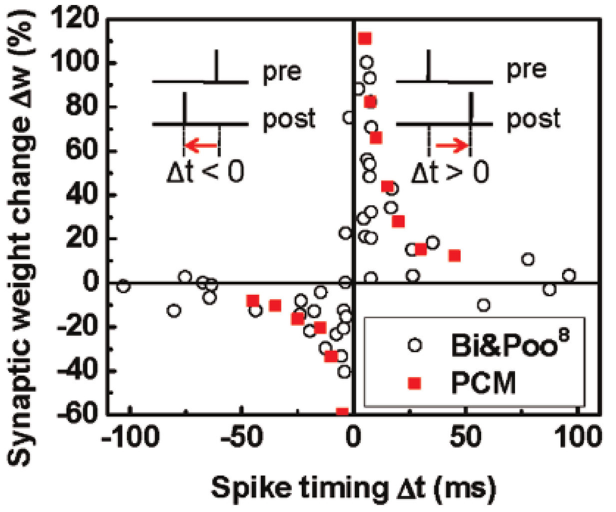
- Using CMOS as synapse → ~10 transistors/synapse
- PCMs as synapses → dense and energy efficient
- GPU instead of CPU → Faster at matrix operations
  - Nvidia leads this market

J. Arthur et al. 2006



Nvidia Tesla P100
$2 Billion in R&D
15 billion transistors



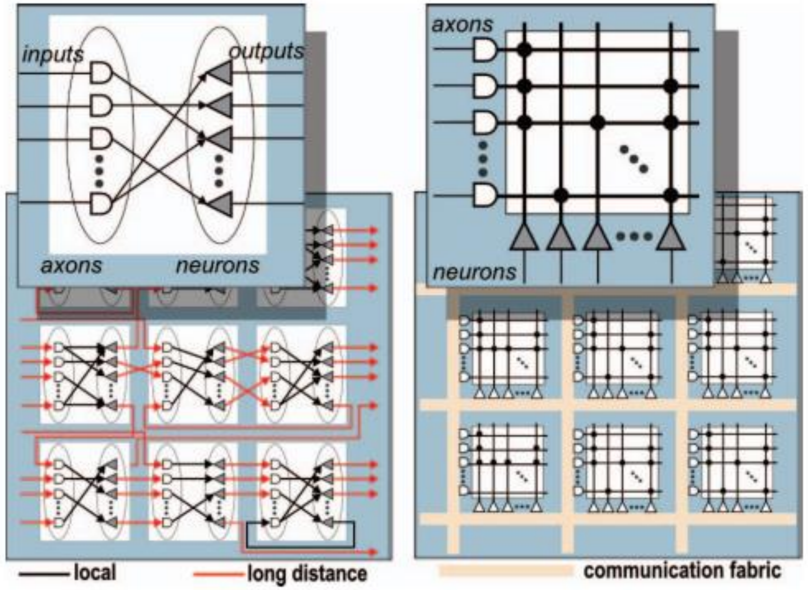


Kuzum et al Nano Lett. 2011

TrueNorth

- The first (?) commercial neuromorphic processing chip
- Neurosynaptic cores of 256 output neurons and 256 input axons, connected by 256x256 synapses
- Cores are connected by network-on-chip
  → large neural network
- 4096 cores as 64x64 array
- >1x10^6 neurons, >256x10^6 synapses
- 5.4 billion transistors (!!)
- Uses spiking neurons
- Each TrueNorth consumes ~ 0.23 W in active mode
  - A one chip system: 3.5 W

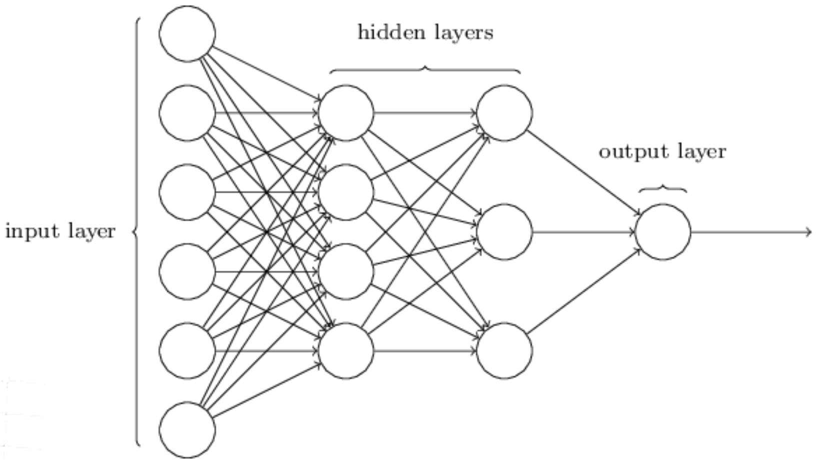
- Currently scaled up to 4x4 TrueNorth system (NS16e)

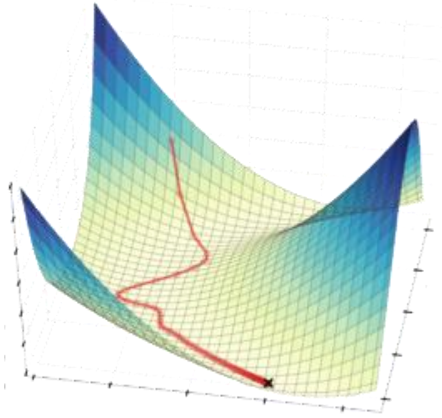


NS16e

Summary – Cognitive computing

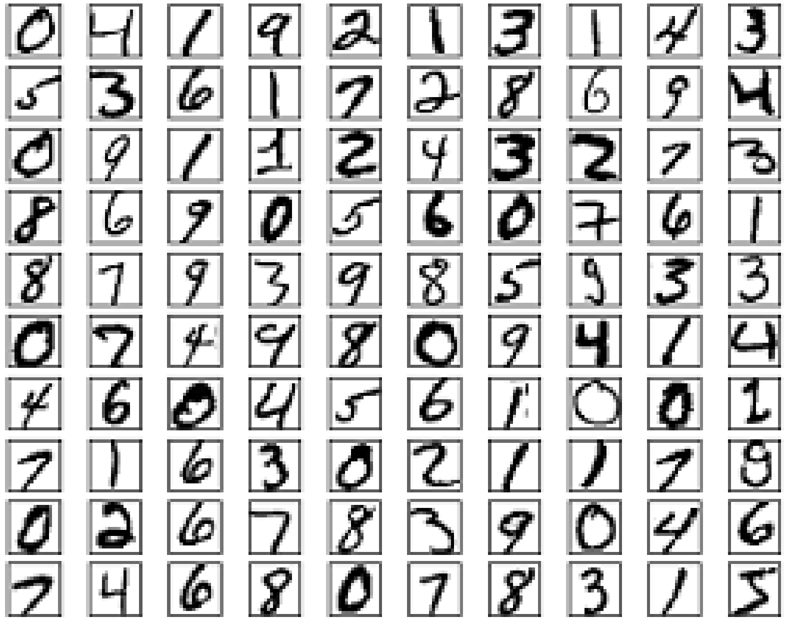
- A new revolution in computing
  - Data-centered computing
  - Big business → industry is leaping at it
- Based on the neural network of the brain
- Simplified neuron models
  - Sigmoid neuron
  - Spiking neurons
- Learning by:
  - gradient descent
  - Back-propagation
- Convolutional deep networks
- Reservoir computing
- GPU instead of CPU
- Specialized hardware for energy efficiency
  - PCM...or something else?





Hand-in assignment

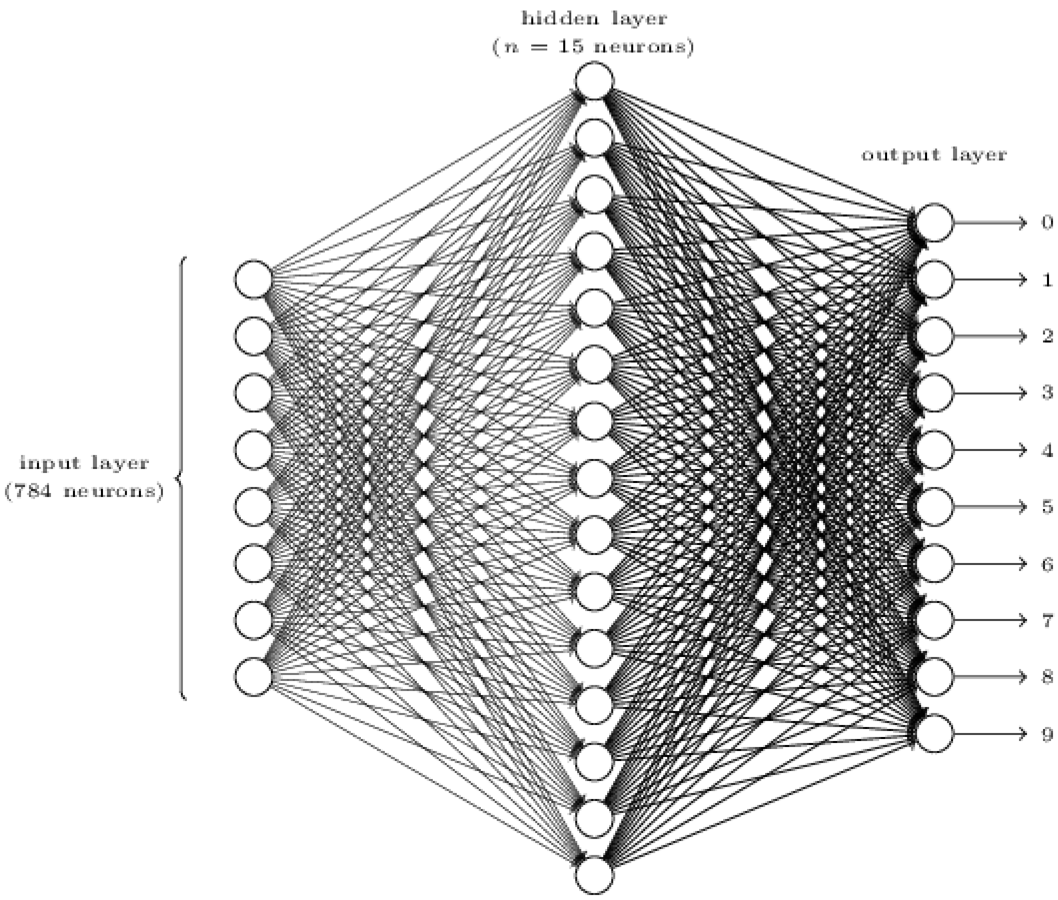
- Task: ”**Design and implement a neural network that can identify handwritten numbers 0-9**”
- MNIST data base (http://yann.lecun.com/exdb/mnist/): 60 000 examples of handwritten numbers
  - 28x28 pixels grey-scale
  - Target: > 95% accuracy (but who will be the best?)
- 1-2 persons per project
- Written report: 5-10 pages
  - What kind of network architecture was used and why?
  - How was learning done and why did you do it that way?
  - Include code as Appendix
- Deadline: **12 March**
- Use literature list as an aid
- Detailed instructions come tomorrow...



Some tips

- Start with designing your network
  - # neurons in input layer: one/pixel
  - # neurons in output layer: 10
  - # hidden layers: experiment
  - # neurons in hidden layer: experiment
  - Use matrix representation!
  - Feed-forward network
  - Use gradient descent and back-propagation

- <u>Important to consider:</u>
- Choice of cost function
- Size of mini-batch
- Learning step size



Oral exams

- 7-10 March
- 30 min discussion 1 on 1
  - Based on content from lectures and literature list

1. Be able to describe devices and phenomena that were brought up on the lectures
2. Be able to describe the impact on society/industry of emerging technologies

- Sign up on Doodle to decide on exact times.