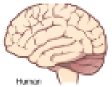
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
FACULTY OF ENGINEERING

Cognitive Neuroscience for AI Developers

Week 4– Structure and Function of the Nervous System: Plasticity + Neurodevo

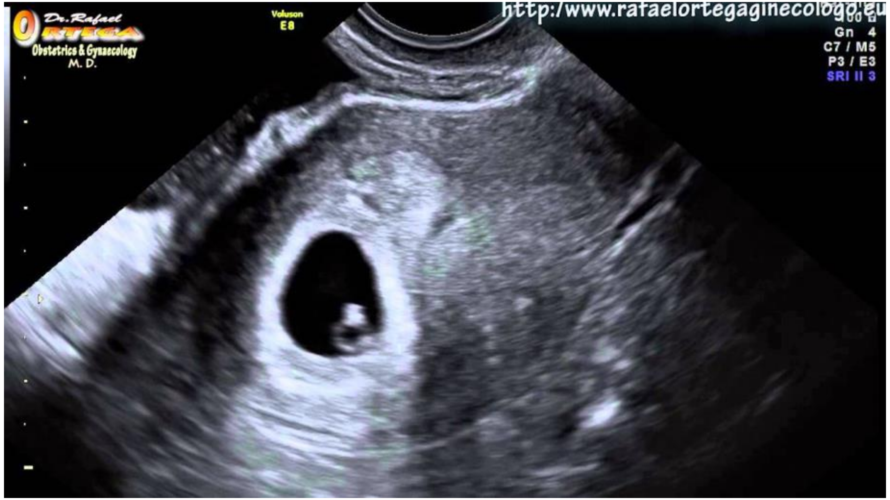
The brain


Pexels.com (CC0)



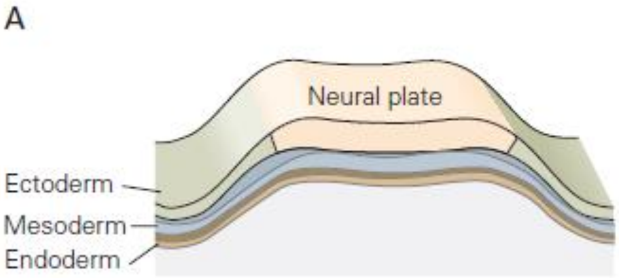
How does it emerge?

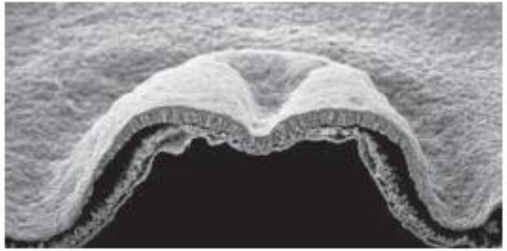
Back in the days…

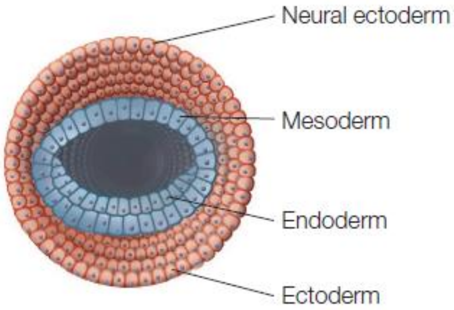


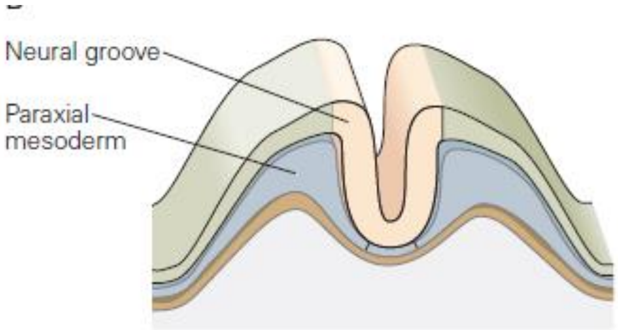
© Pauline Breijer

What is happening there?

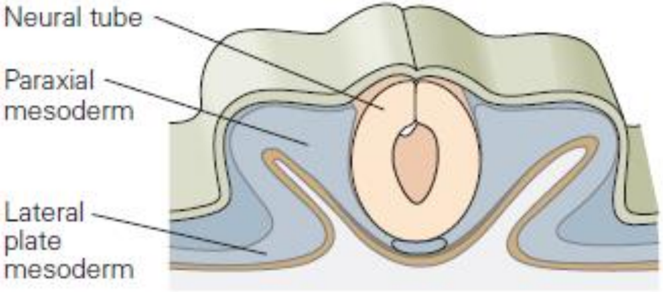


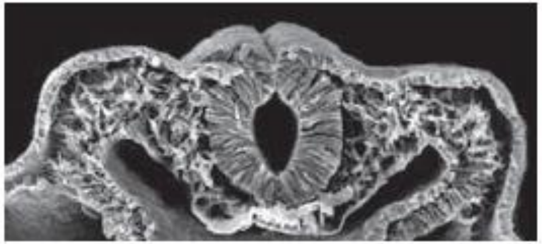


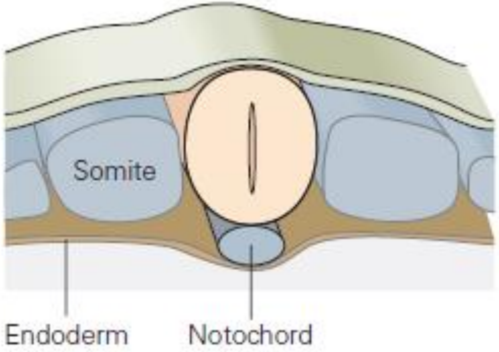


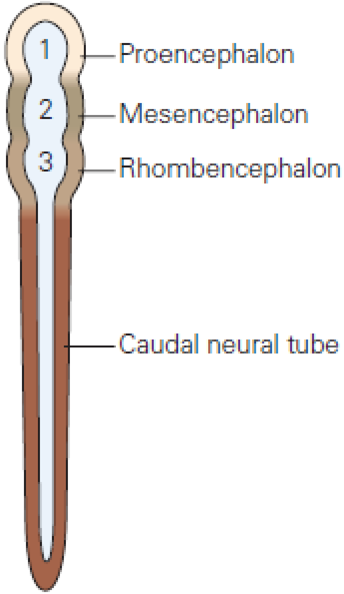


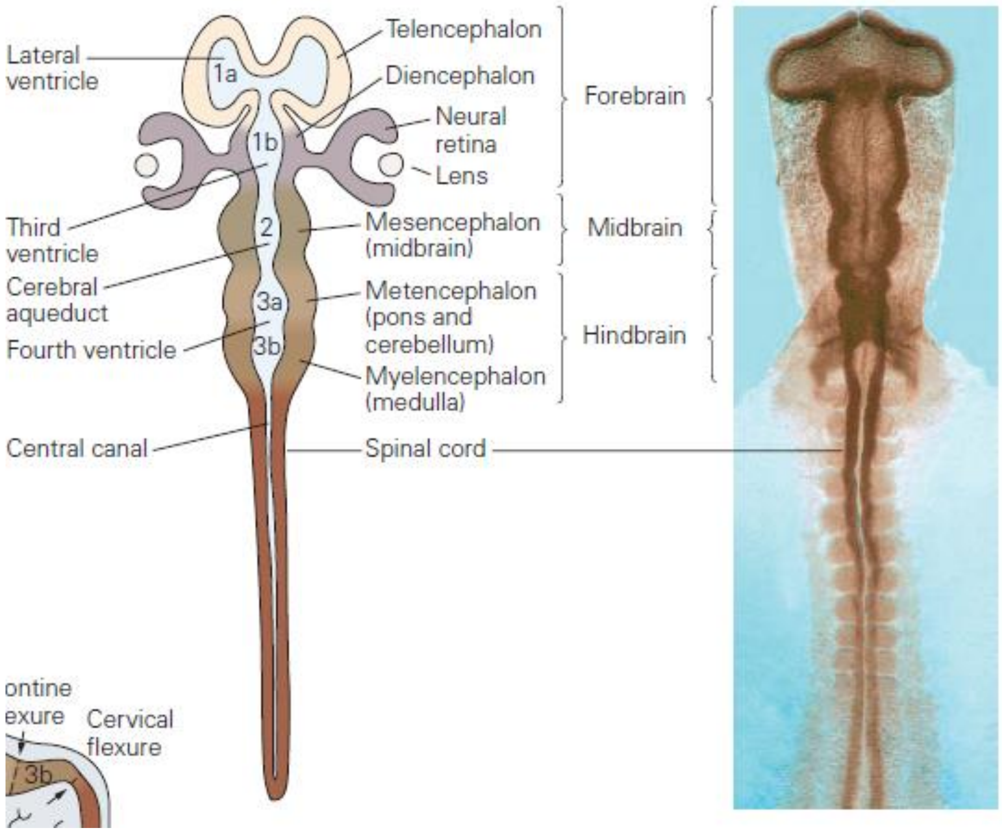
What is happening there?



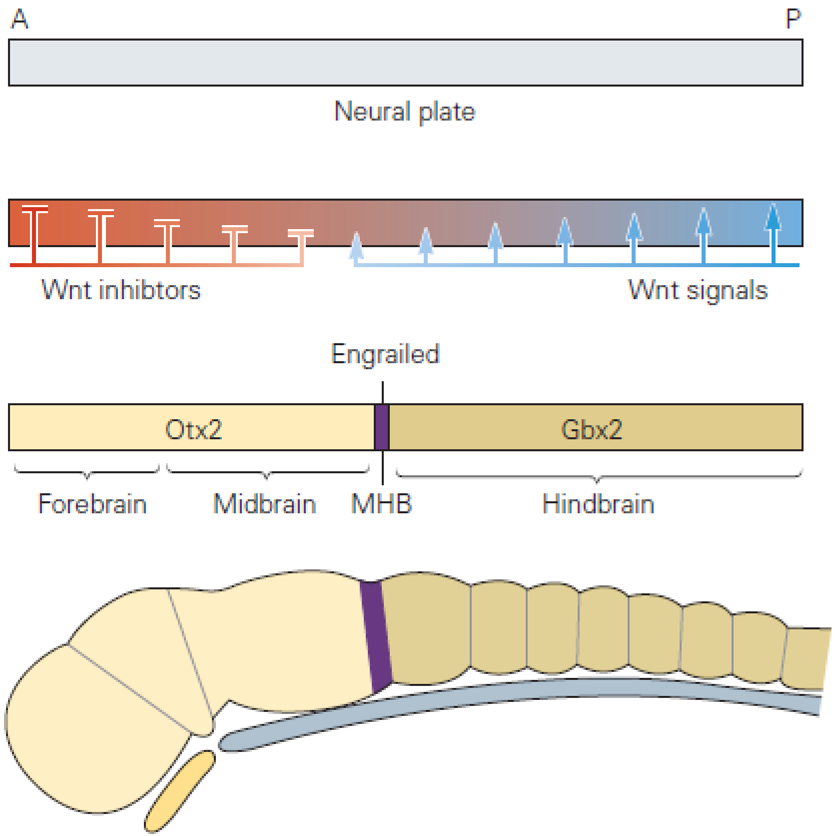


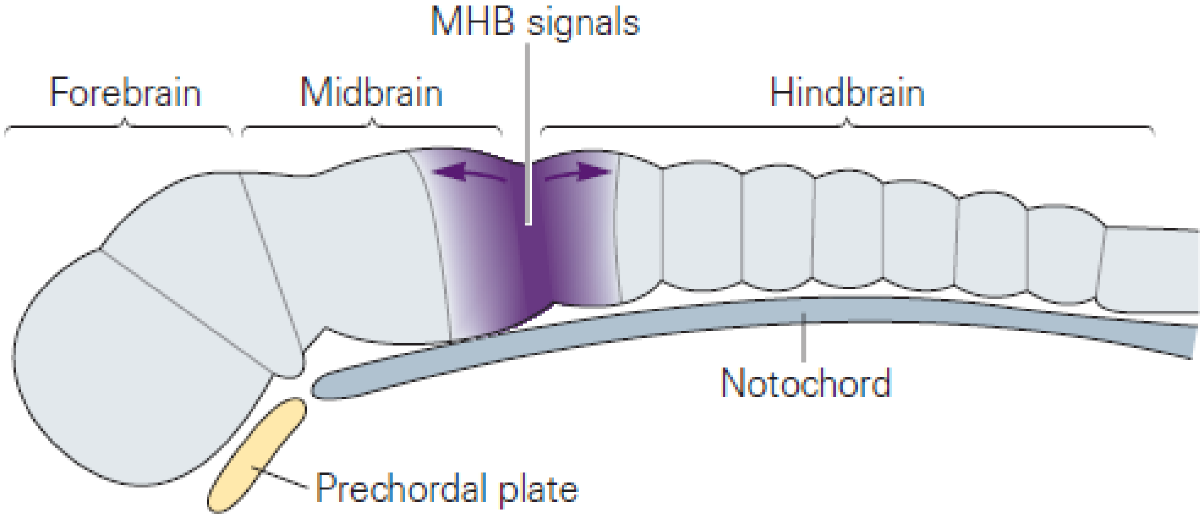



1
2
3
Proencephalon
Mesencephalon
Rhombencephalon
Caudal neural tube

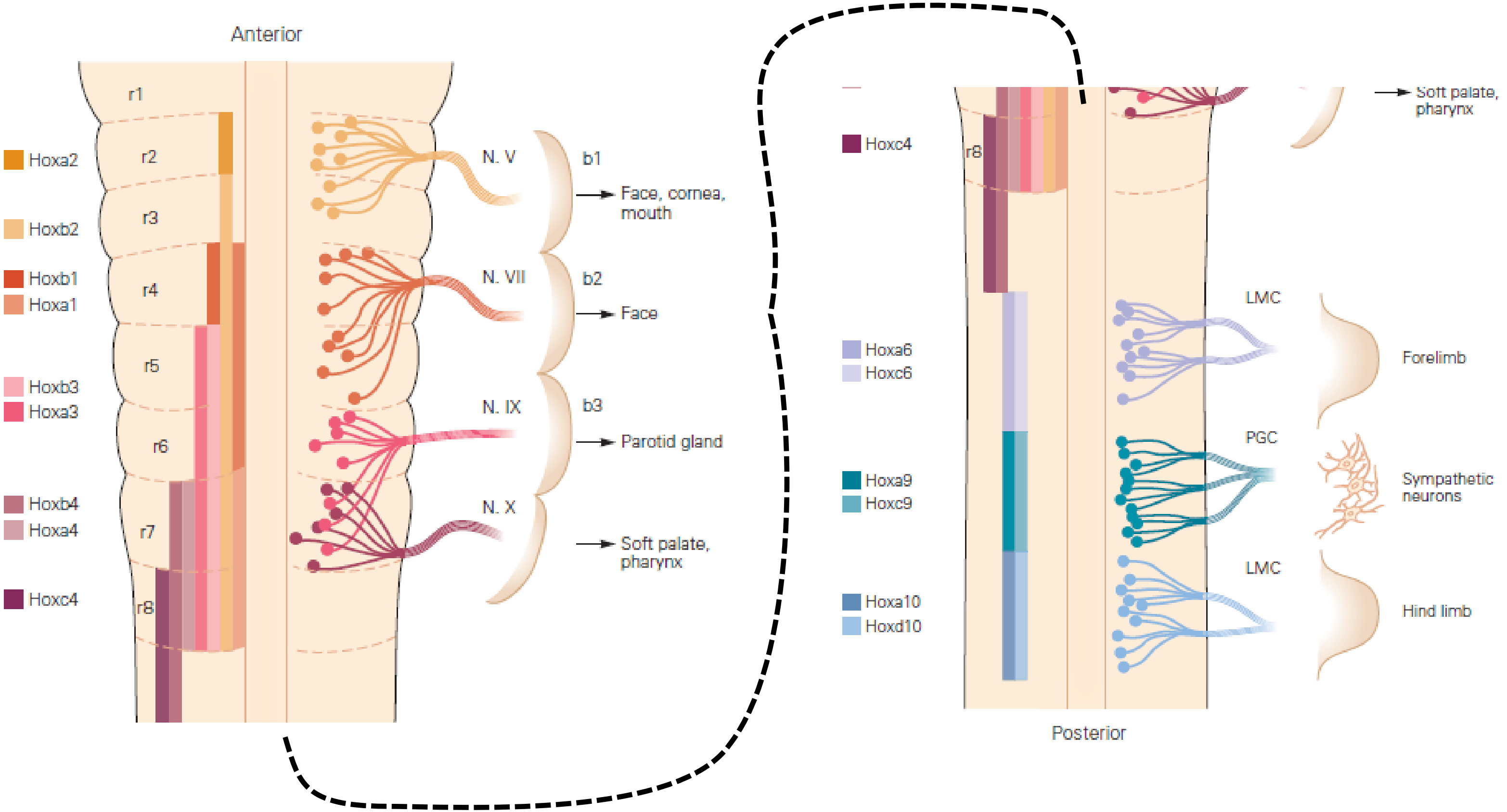

Lateral ventricle
1a
1b
2
3a
3b
Third ventricle
Cerebral aqueduct
Fourth ventricle
Central canal
Telencephalon
Diencephalon
Neural retina
Lens
Mesencephalon (midbrain)
Metencephalon (pons and cerebellum)
Myelencephalon (medulla)
Spinal cord
Forebrain
Midbrain
Hindbrain
ontine
exure
Cervical
flexure

Signaling pathways define neural development

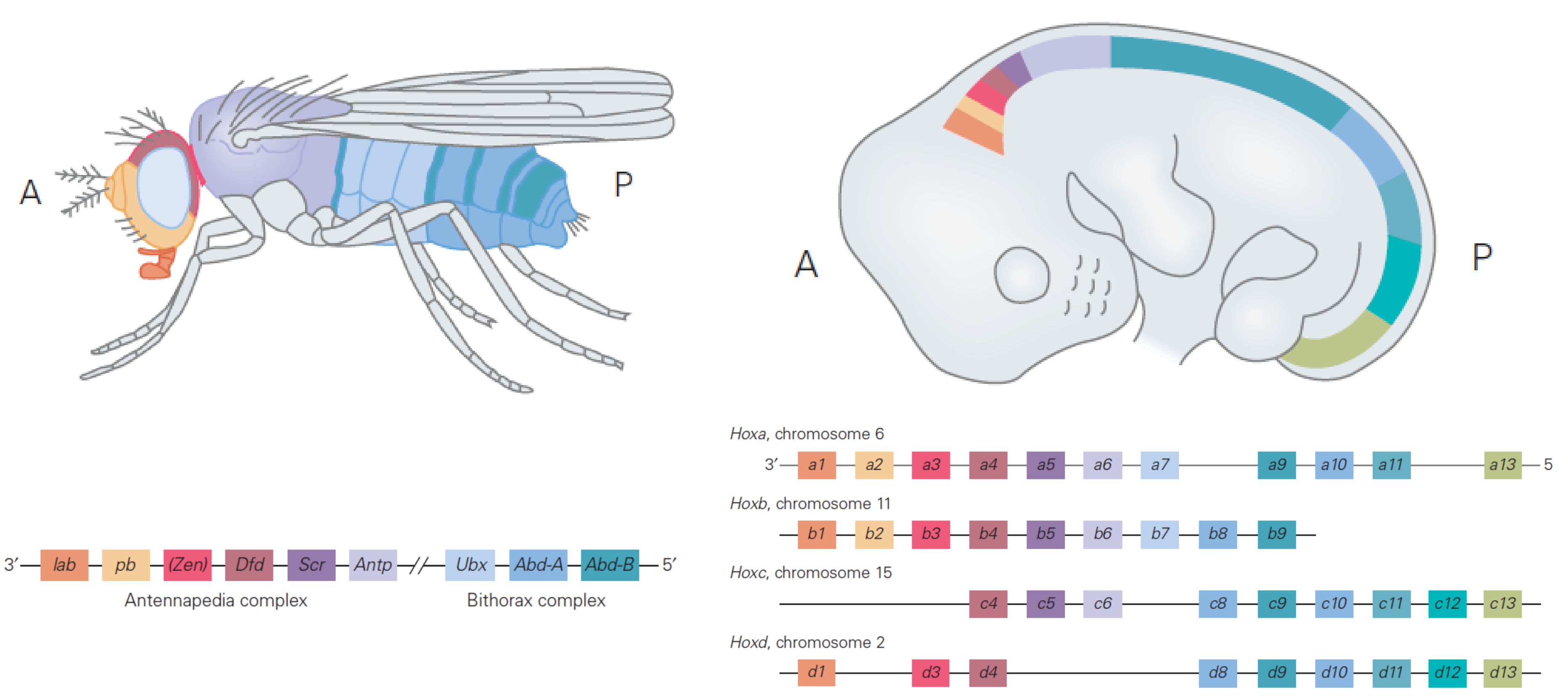



MHB signals
Forebrain
Midbrain
Hindbrain
Notochord
Prechordal plate

Hox genes determines motor neurons



Positional development is highly conserved

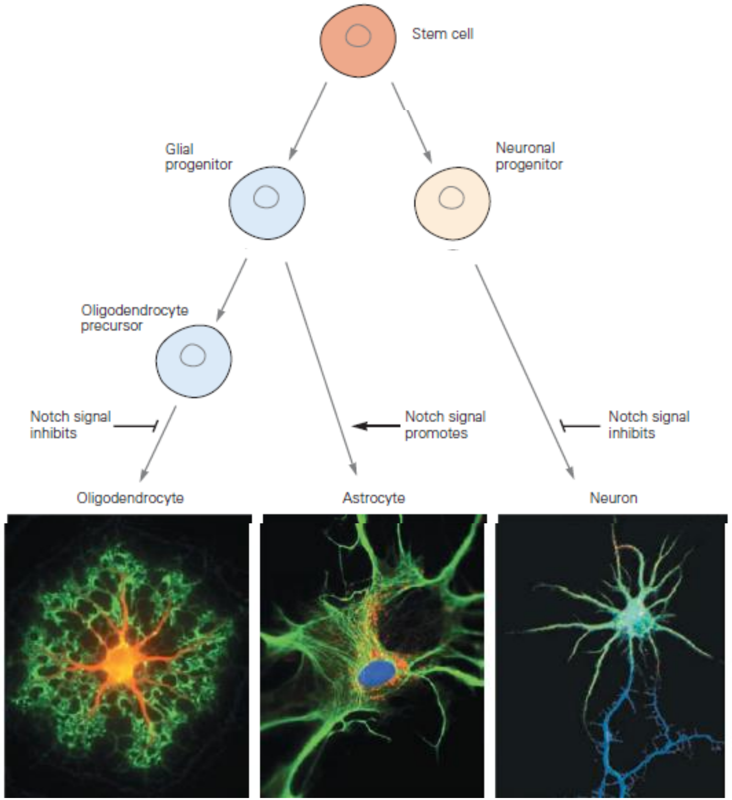


Mammals have very similar development

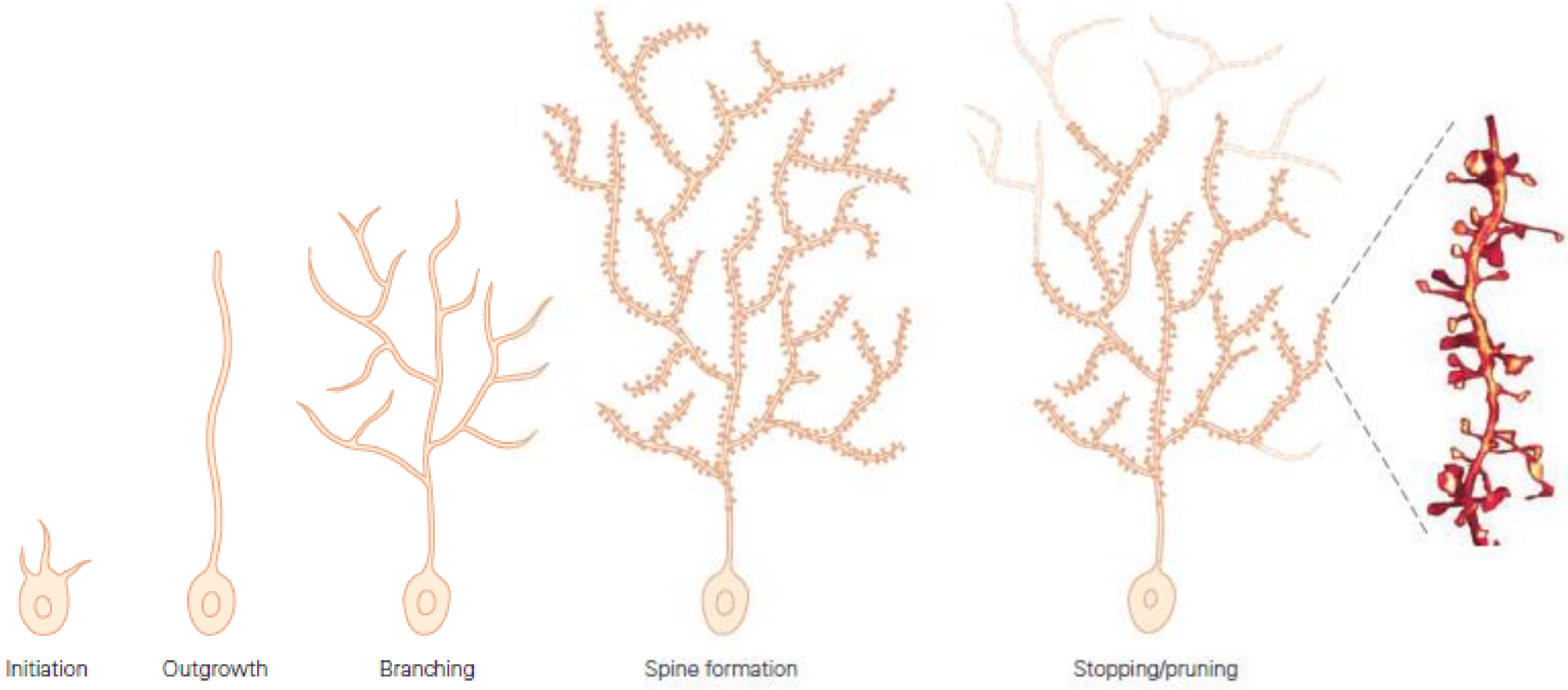
Human

Pig

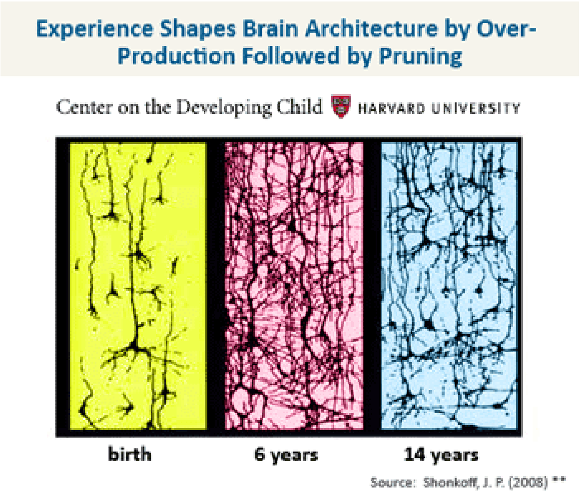
Brain cells have a common ancestor

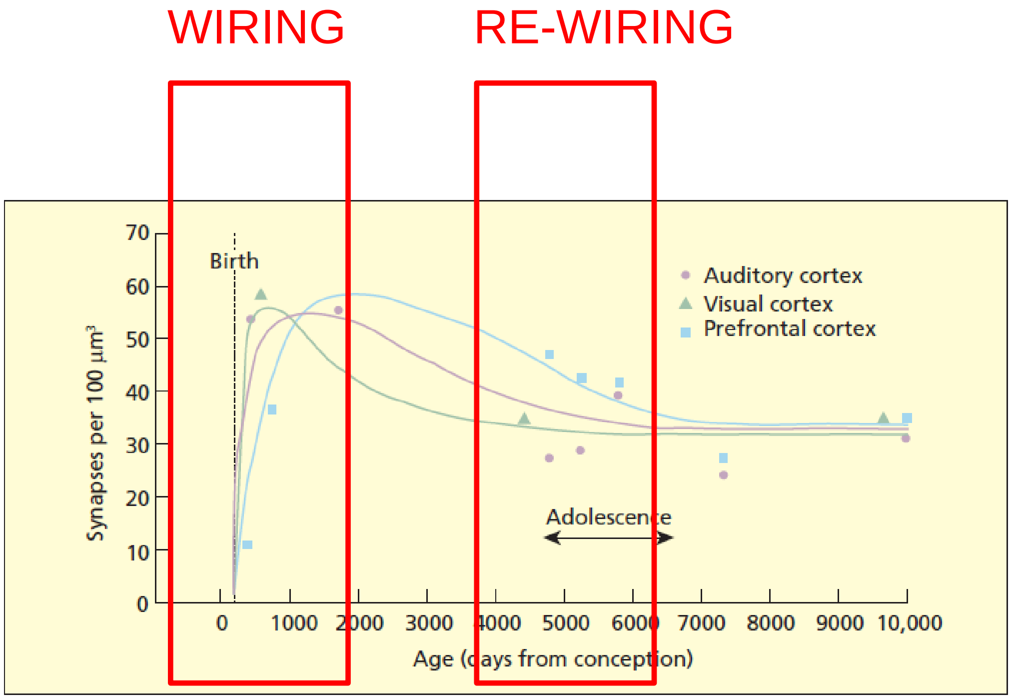


Neuron maturation



Pruning over lifetime





From Huttenlocher and Dabholkar, 1997. Reprinted with permission of John Wiley & Sons Inc.

Pruning in deep neural networks

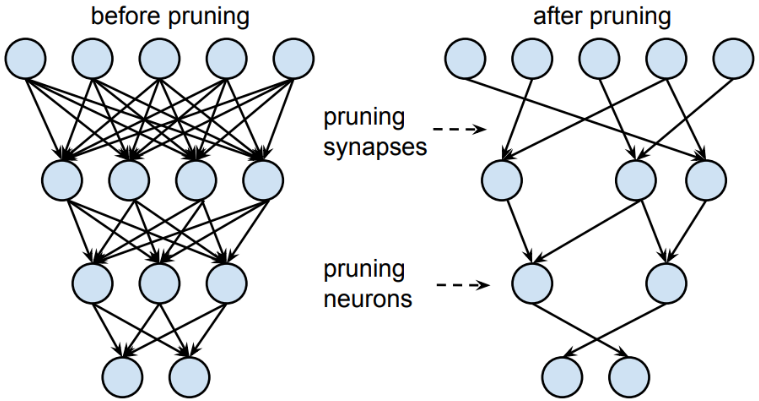
Optimal Brain Damage

Yann Le Cun, John S. Denker and Sara A. Solla
AT&T Bell Laboratories, Holmdel, N. J. 07733

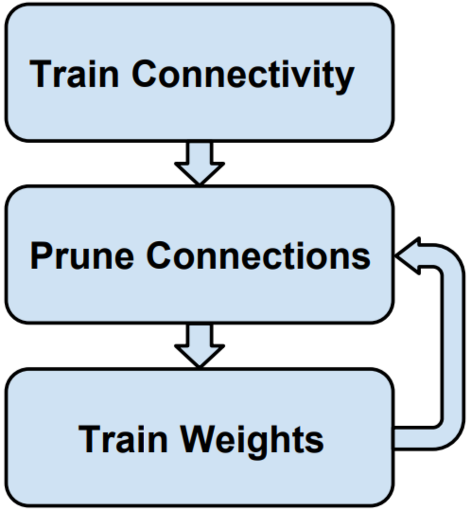
ABSTRACT

We have used information-theoretic ideas to derive a class of practical and nearly optimal schemes for adapting the size of a neural network. By removing unimportant weights from a network, several improvements can be expected: better generalization, fewer training examples required, and improved speed of learning and/or classification. The basic idea is to use second-derivative information to make a tradeoff between network complexity and training set error. Experiments confirm the usefulness of the methods on a real-world application.

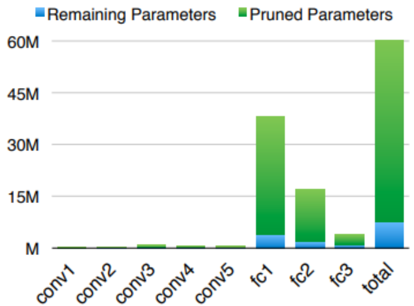
Le Cun et al., 1990



Pruning synapses:
making network sparse
Pruning neurons:
Making network dense



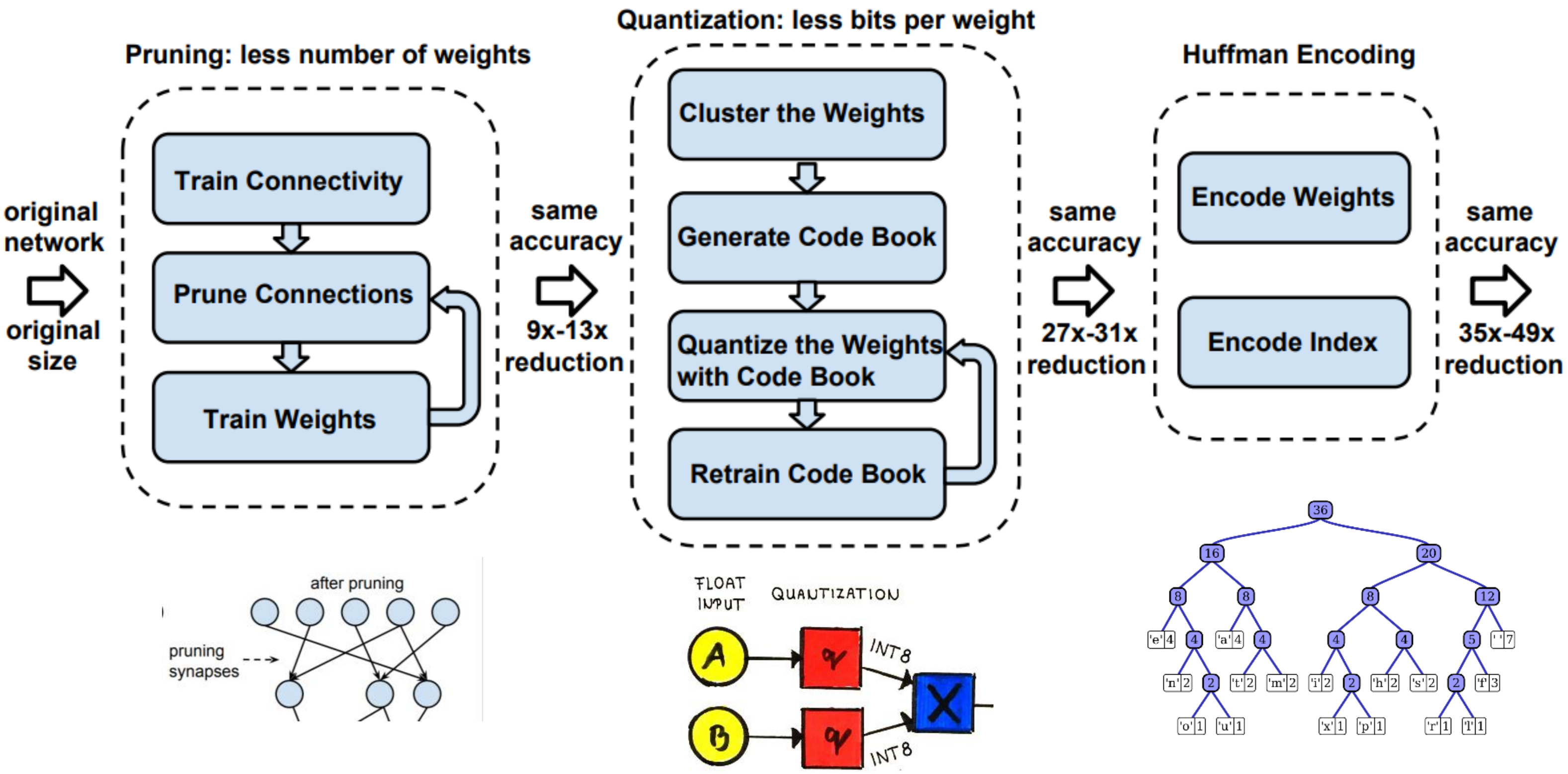
Layer	Weights	FLOP	Act%	Weights%	FLOP%
conv1	35K	211M	88%	84%	84%
conv2	307K	448M	52%	38%	33%
conv3	885K	299M	37%	35%	18%
conv4	663K	224M	40%	37%	14%
conv5	442K	150M	34%	37%	14%
fc1	38M	75M	36%	9%	3%
fc2	17M	34M	40%	9%	3%
fc3	4M	8M	100%	25%	10%
Total	61M	1.5B	54%	**11%**	**30%**



Layer	Weights	FLOP	Act%	Weights%	FLOP%
conv1_1	2K	0.2B	53%	58%	58%
conv1_2	37K	3.7B	89%	22%	12%
conv2_1	74K	1.8B	80%	34%	30%
conv2_2	148K	3.7B	81%	36%	29%
conv3_1	295K	1.8B	68%	53%	43%
conv3_2	590K	3.7B	70%	24%	16%
conv3_3	590K	3.7B	64%	42%	29%
conv4_1	1M	1.8B	51%	32%	21%
conv4_2	2M	3.7B	45%	27%	14%
conv4_3	2M	3.7B	34%	34%	15%
conv5_1	2M	925M	32%	35%	12%
conv5_2	2M	925M	29%	29%	9%
conv5_3	2M	925M	19%	36%	11%
fc6	103M	206M	38%	4%	1%
fc7	17M	34M	42%	4%	2%
fc8	4M	8M	100%	23%	9%
total	138M	30.9B	64%	**7.5%**	**21%**

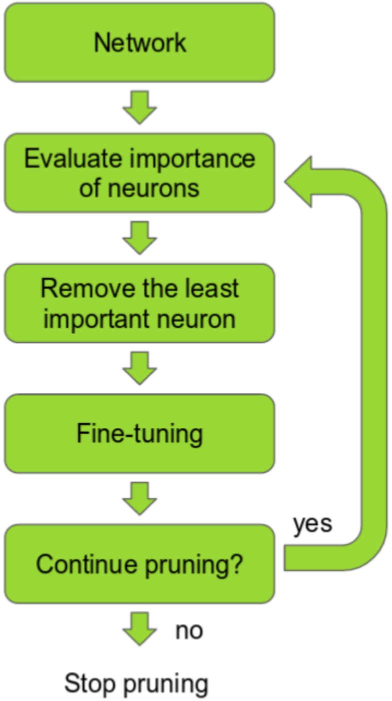
Han et al., NeurIPS 2015

Pruning in deep neural networks



Han et al., Deep Compression ICLR 2016

Bruteforce NVIDIA study



2.1 Oracle pruning

Minimizing the difference in accuracy between the full and pruned models depends on the criterion for identifying the "least important" parameters, called *saliency*, at each step. The best criterion would be an exact empirical evaluation of each parameter, which we denote the *oracle* criterion, accomplished by ablating each non-zero parameter $w \in \mathcal{W}'$ in turn and recording the cost's difference.

To compute the oracle, we evaluate the change in loss caused by removing each individual feature map from the fine-tuned VGG-16 network. (See Appendix A.3 for additional analysis.) We rank

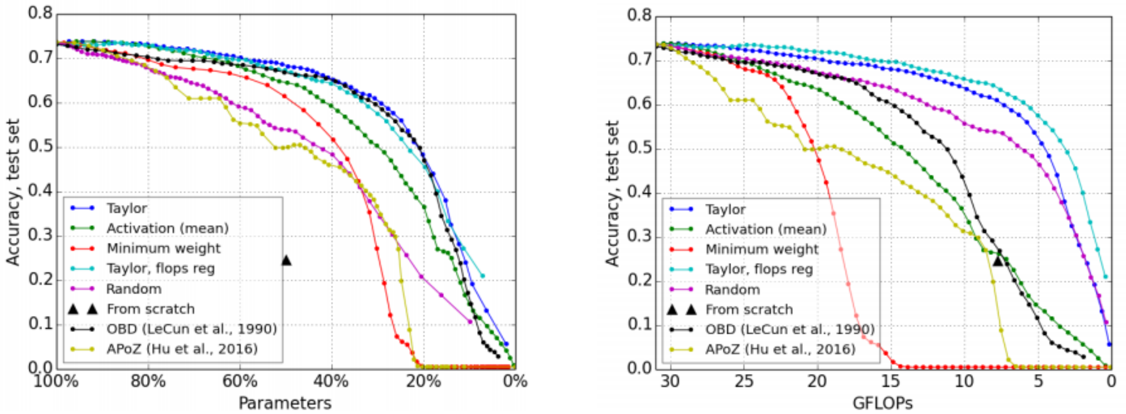


Figure 4: Pruning of feature maps in VGG-16 fine-tuned on the Birds-200 dataset.

Molchanov et al., ICLR 2017

State of NN pruning

MLSys 2020

What is the State of Neural Network Pruning?

Davis Blalock [*1] Jose Javier Gonzalez Ortiz [*1] Jonathan Frankle [1] John Guttag [1]

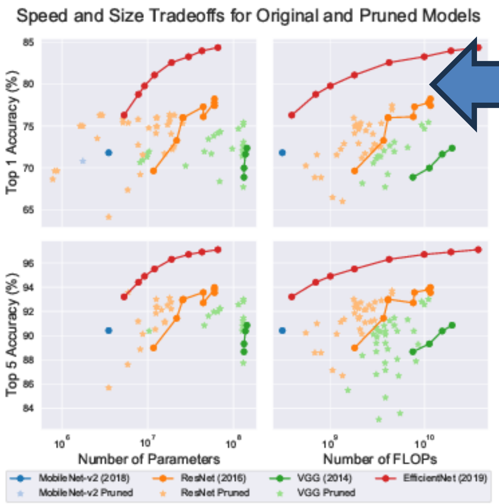
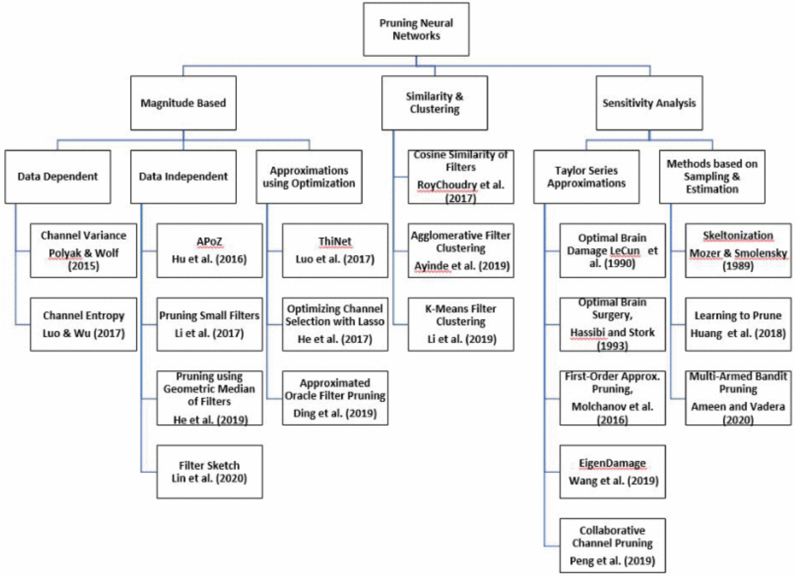


Figure 1: Size and speed vs accuracy tradeoffs for different pruning methods and families of architectures. Pruned models sometimes outperform the original architecture, but rarely outperform a better architecture.

4.5 Confounding Variables

Even when comparisons include the same datasets, models, ... d operating points, other confounding variables ... meaningful comparisons difficult. Some vari- ... particular interest include:

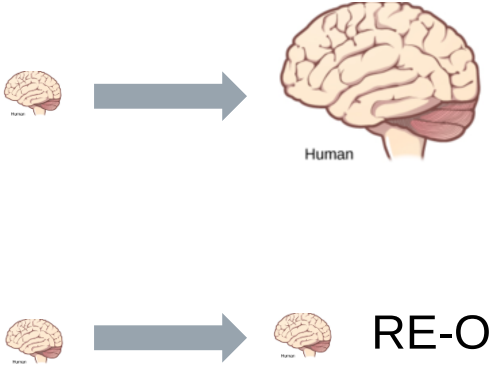
- Accuracy and efficiency of the initial model
- Data augmentation and preprocessing
- Random variations in initialization, training, and fine-tuning. This includes choice of optimizer, hyperparameters, and learning rate schedule.
- Pruning and fine-tuning schedule
- Deep learning library. Different libraries are known to



Vadeera and Ameen, IEEE Access 2022

The brain is dynamic


Pexels.com (CC0)



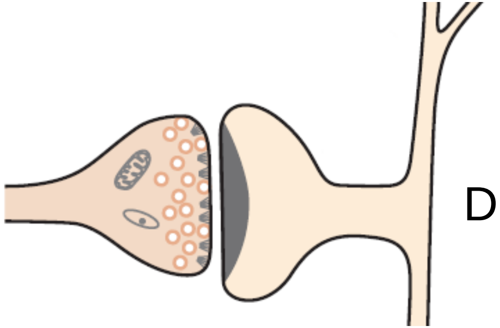
GROWTH

RE-ORGANIZATION

Neural plasticity

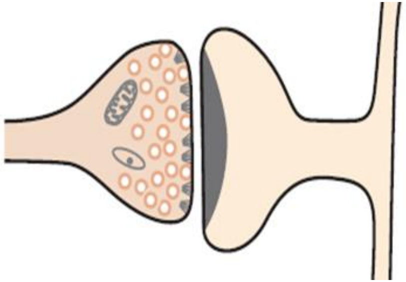
Changing the synapse strength

Input from other neuron

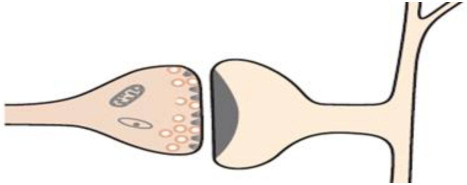


Dendrite

pre post



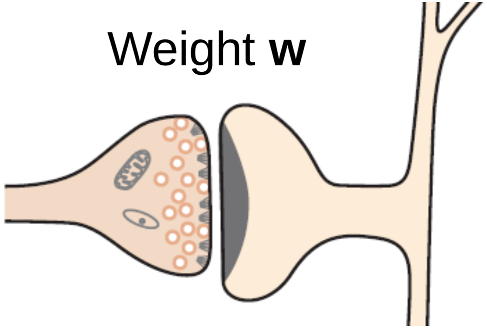
Intensify connection:
Long-term potentiation (LTP)



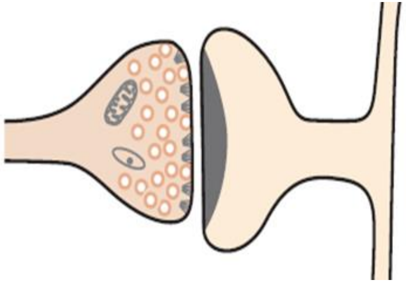
Loosen connection:
Long-term depression (LTD)

Changing the synapse strength

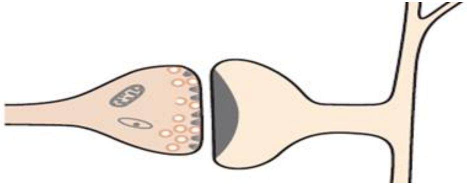
Weight **w**



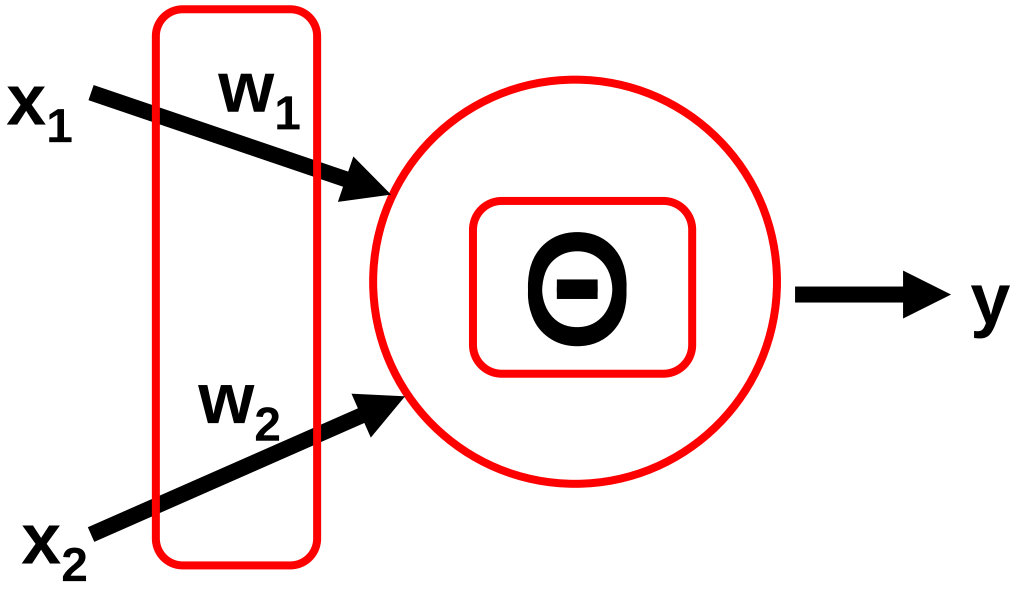
Weight **w**↑



Weight **w**↓



Perceptrons



Frank Rosenblatt

How do we change…

Weights: Hebbian learning, Delta Rule
Synaptic anatomy

Threshold: Learnable w/ SGD

Learnable Extended Activation Function for Deep Neural Networks

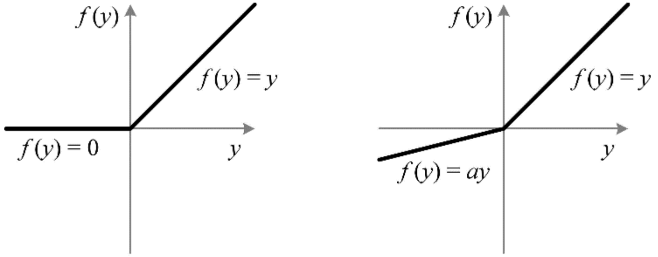
YEVGENIY BODYANSKIY[1], SERHII KOSTIUK[2]
[1]Control Systems Research Laboratory, National University of Radio Electronics, Nauky av. 14, Kharkiv, 61166, Kharkiv, Ukraine, (e-mail: yevgeniy.bodyanskiy@nure.ua)
[2]Dept. of Artificial Intelligence, National University of Radio Electronics, Nauky av. 14, Kharkiv, 61166, Kharkiv, Ukraine, (e-mail: serhii.kostiuk@nure.ua)
Corresponding author: Serhii Kostiuk (e-mail: serhii.kostiuk@nure.ua).

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

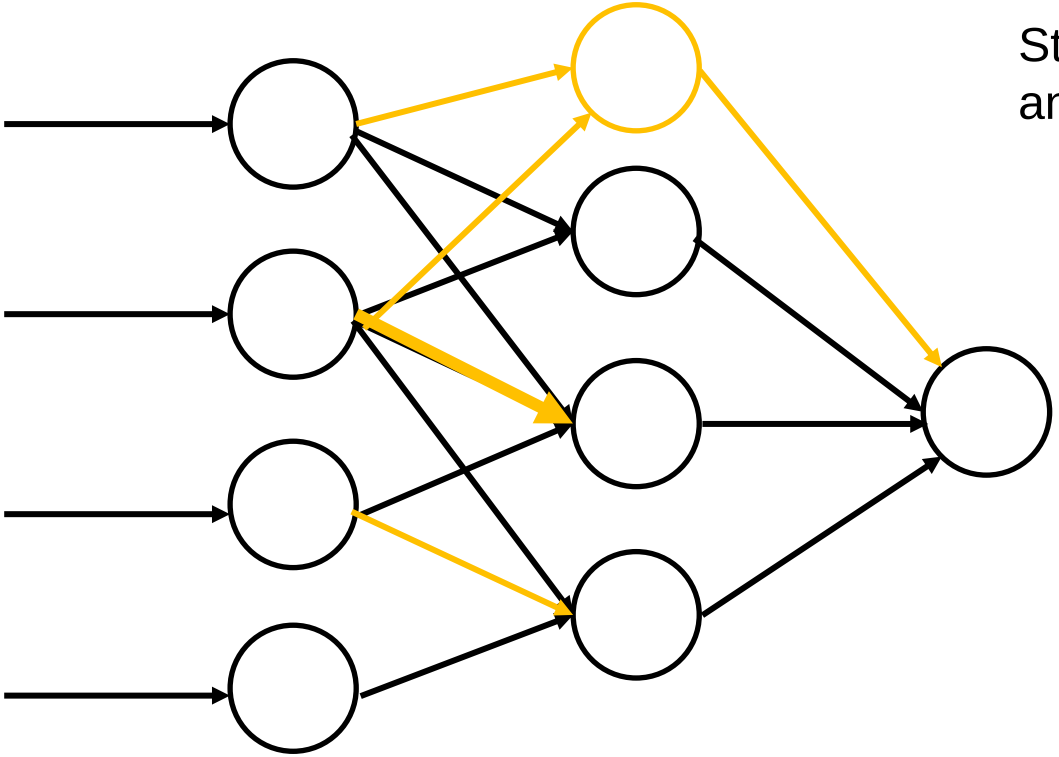
Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun

Activation function:

Parametric ReLU
(Earlier layers more linear,
Later more variable)



Multilayer perceptrons

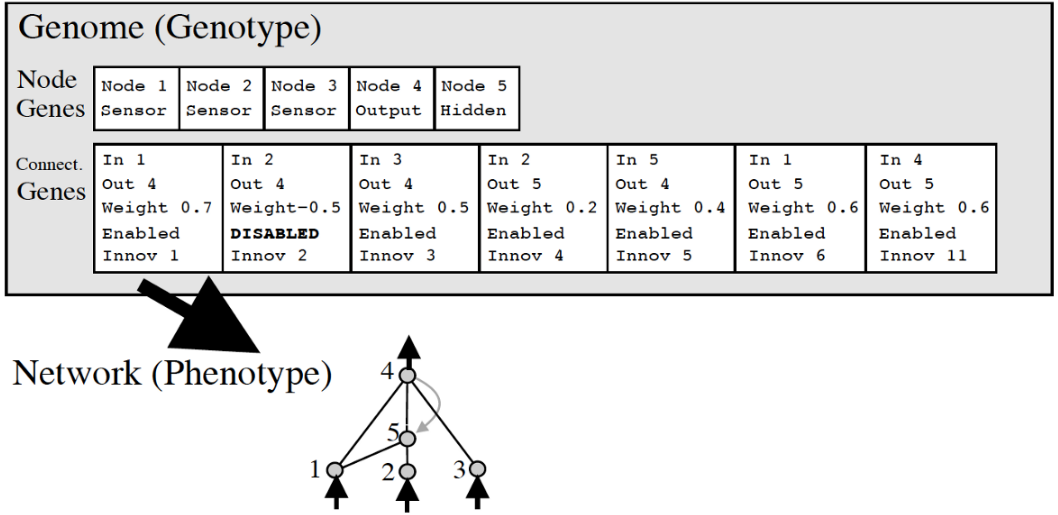


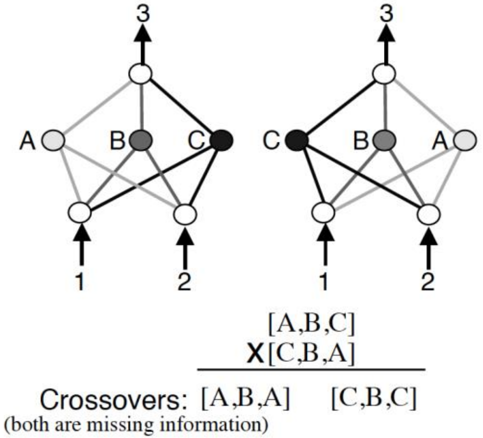
Structural
and synaptic plasticity

→ Typical learning paradigms work with synaptic plasticity

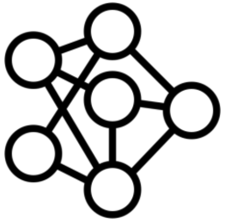
NEAT

Evolving neural networks through augmenting topologies
KO Stanley, R Miikkulainen - Evolutionary computation, 2002 - MIT Press
An important question in neuroevolution is how to gain an advantage from evolving neural network topologies along with weights. We present a method, NeuroEvolution of Augmenting Topologies (NEAT), which outperforms the best fixed-topology method on a challenging benchmark reinforcement learning task. We claim that the increased efficiency is due to (1) employing a principled method of crossover of different topologies,(2) protecting structural innovation using speciation, and (3) incrementally growing from minimal structure ...





NEAT vs. the human brain



No topology enforced

Overall topology enforced

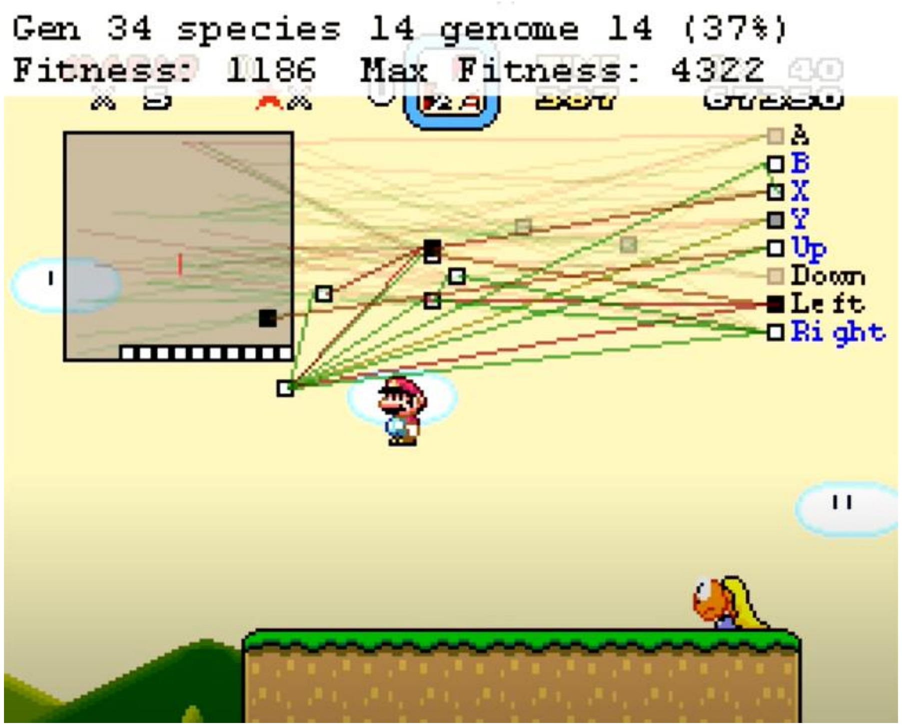
Minimal set to tackle task

One-fits-all solution

Hard to extrapolate to
Similar, yet slightly different tasks

Very good in abstraction

Applied NEAT



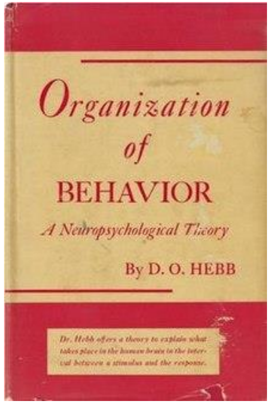
„**MarI/O** is a program made of neural networks and genetic algorithms that kicks butt at Super Mario World.”

© Seth Bling, YouTube https://www.youtube.com/watch?v=qv6UVOQ0F44

How to change weights?

Hebb‘s rule:

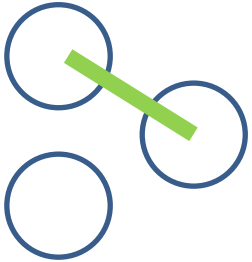
„Neurons that fire together, wire together.“



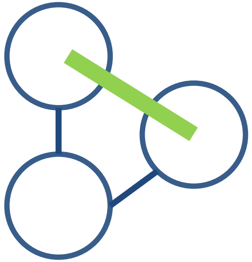
$$w_{ij} = x_i x_j$$

x_i	x_j	w_{ij}
0	0	0
1	0	0
0	1	0
1	1	1

Hebb‘s rule



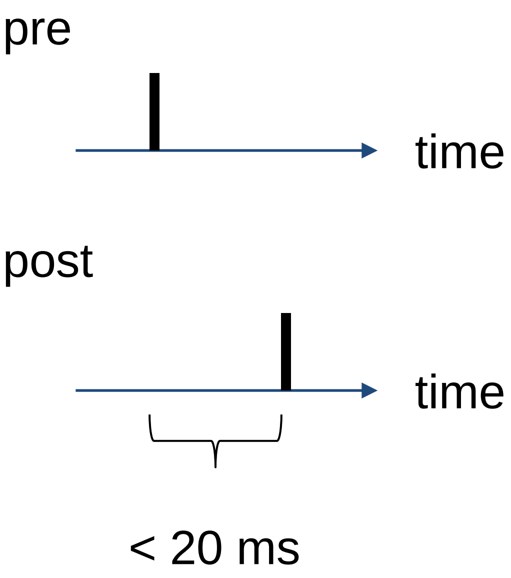
Structural
plasticity



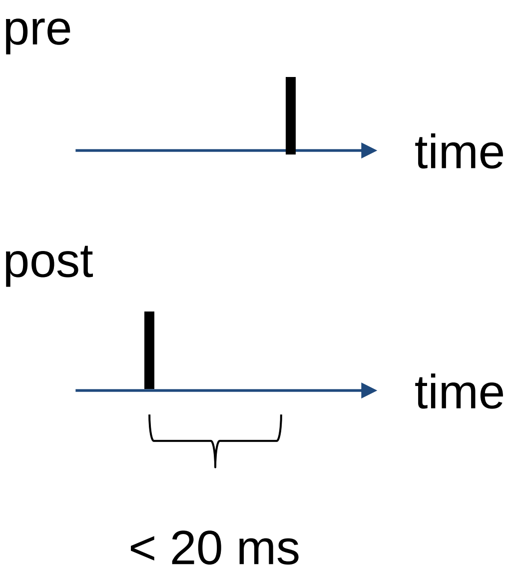
Synaptic
plasticity

Spike timing dependent plasticity (STDP)

Hebb‘s rule – cntd.



Synaptic **w**↑ → **LTP**



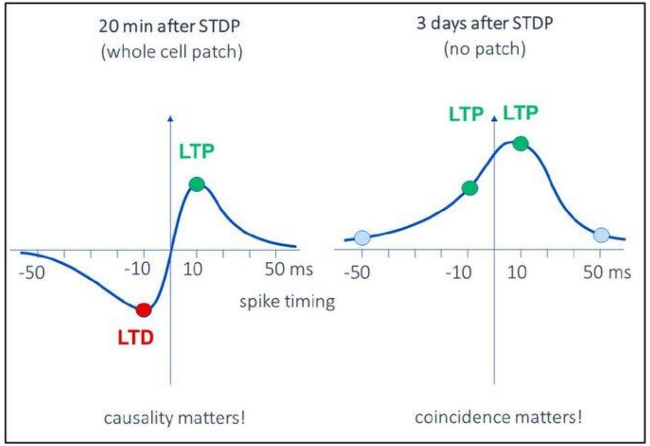
Synaptic **w**↓ → **LTD**

Spike timing dependent plasticity

Recent Research

Spike-timing-dependent plasticity rewards synchrony rather than causality

Margarita Anisimova[1], Bas van Bommel[1], Marina Mikhaylova[1,2], J. Simon Wiegert, Thomas G. Oertner[1*], Christine E. Gee[1,3*]



“Our results confirm that neurons wire together if they fire together, but suggest that synaptic depression after anti-causal activation (tLTD) is a transient phenomenon.”

https://doi.org/10.1101/863365

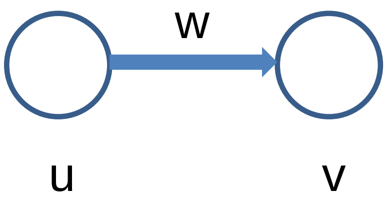
biorxiv, 2021

Synaptic plasticity rules

The basic Hebb‘s rule:

$$v = w \cdot u$$

$$\tau_w \frac{d\boldsymbol{w}}{dt} = v \cdot \boldsymbol{u}$$



$$\tau_w \frac{d\boldsymbol{w}}{dt} = \boldsymbol{w} \cdot \boldsymbol{u} \cdot \boldsymbol{u}^T$$

Correlation matrix

$$\tau_w \frac{d\boldsymbol{w}}{dt} = \boldsymbol{Q} \cdot \boldsymbol{w}^T$$

$$v = \boldsymbol{w} \cdot \boldsymbol{u}^T$$

Correlation-based plasticity rule

$$\boldsymbol{w} \rightarrow \boldsymbol{w} + \epsilon \boldsymbol{Q} \cdot \boldsymbol{w}^T \qquad \epsilon = \frac{1}{\tau_w}$$

Synaptic plasticity rules

Correlation-based plasticity rule

$$\tau_w \frac{d\boldsymbol{w}}{dt} = \boldsymbol{Q} \cdot \boldsymbol{w}^{\boldsymbol{T}}$$

→ Basic Hebb only allows LTP

Postsynaptic
LTD/LTP switch

$$\tau_w \frac{d\boldsymbol{w}}{dt} = (v - \theta_v) \cdot \boldsymbol{u}$$

$$\tau_w \frac{d\boldsymbol{w}}{dt} = v \cdot (\boldsymbol{u} - \boldsymbol{\theta_u})$$

Presynaptic
LTD/LTP switch

Covariance matrix

$$v = \boldsymbol{w} \cdot \boldsymbol{u}^{\boldsymbol{T}} \Rightarrow \tau_w \frac{d\boldsymbol{w}}{dt} = \boldsymbol{w} \cdot \boldsymbol{u} \cdot (\boldsymbol{u} - \boldsymbol{\theta_u})^{\boldsymbol{T}} \Rightarrow \tau_w \frac{d\boldsymbol{w}}{dt} = \boldsymbol{C} \cdot \boldsymbol{w}^{\boldsymbol{T}}$$

Covariance-based plasticity rule

BCM rule

Hebbian learning suffers from instability

$$\tau_w \frac{d\boldsymbol{w}}{dt} = v \cdot \boldsymbol{u} \cdot (v - \theta_v)$$ If constant → unstable

➔ Threshold of postsynaptic activity that determins if synapse is strengthened or weakened.

Adapt threshold θ_v: $$\tau_\theta \frac{d\theta_v}{dt} = v^2 - \theta_v$$

Synaptic Normalization

$$\tau_\theta \frac{d\theta_v}{dt} = v^2 - \theta_v$$

→ Stabilize weights through postsynaptic activity

Can we use penalty terms directly on the weight vector?

$$\tau_w \frac{d\boldsymbol{w}}{dt} = v \cdot \boldsymbol{u} - \frac{v(\boldsymbol{n} \cdot \boldsymbol{u}^T)\boldsymbol{n}}{N_u}$$

Normalize by subtracting the same quantity

Oja‘s rule

$$\tau_w \frac{d\boldsymbol{w}}{dt} = v \cdot \boldsymbol{u} - \alpha \cdot v^2 \cdot \boldsymbol{w} \qquad \alpha > 0$$

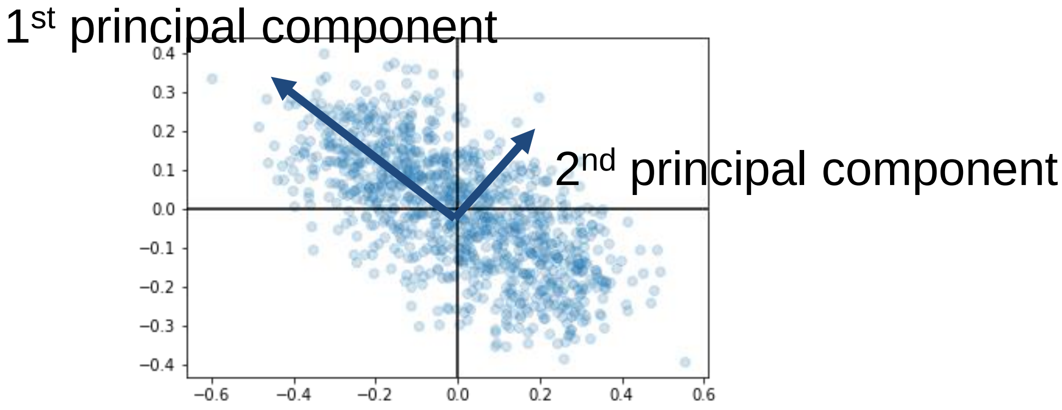
$$\tau_w \frac{d\boldsymbol{w}}{dt} = v \cdot (\boldsymbol{u} - \alpha \cdot v \cdot \boldsymbol{w})$$

https://www.ae-info.org/ae/Member/Oja_Erkki

Oja‘s rule

Unsupervised learning

PRINCIPAL COMPONENT ANALYSIS (PCA)



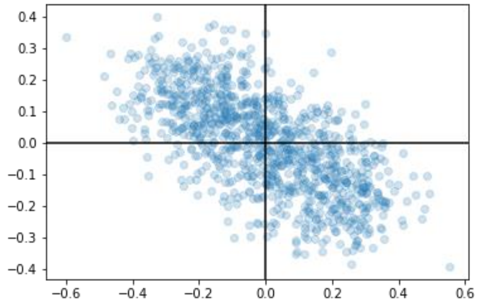
Oja‘s rule

Simulation

Steps:

1. Data generation
2. Variable initialization
3. Iterate through data
    1. Compute pre-synaptic input
    2. Compute post-synaptic activation
    3. Compute Δw and update
4. Plot result

```python
# That you see the same what I see
np.random.seed(42)
N = 1000

# Generate random data
x = np.linspace(-.3, .3, N)
np.random.shuffle(x)
y = -.7 * x

x += np.random.randn(x.size) / 10
y += np.random.randn(y.size) / 10
```



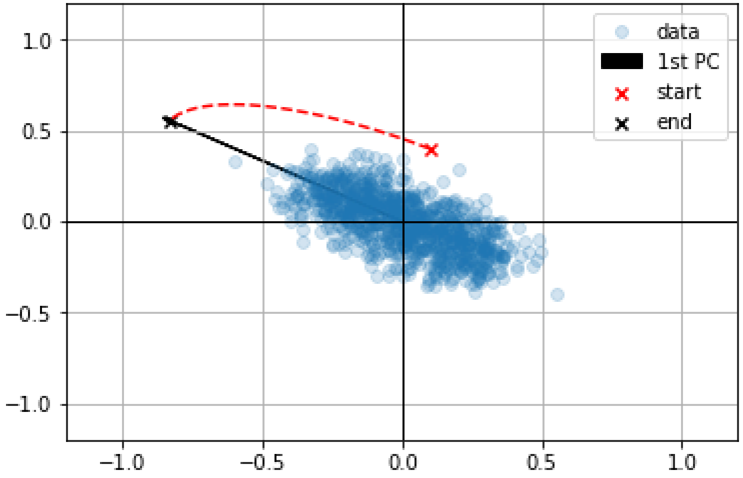

```python
# Initialize some random weights
w = np.array([0.1, 0.4])
# # Training iterations
N = 1000
eta = 0.1
ws = []

# Training
for i in range(N):
    rPre  = np.asarray([x[ix], y[ix]])
    rPost = w @ rPre
    w = w + eta * rPost * (rPre - rPost * w)
```

Oja‘s rule

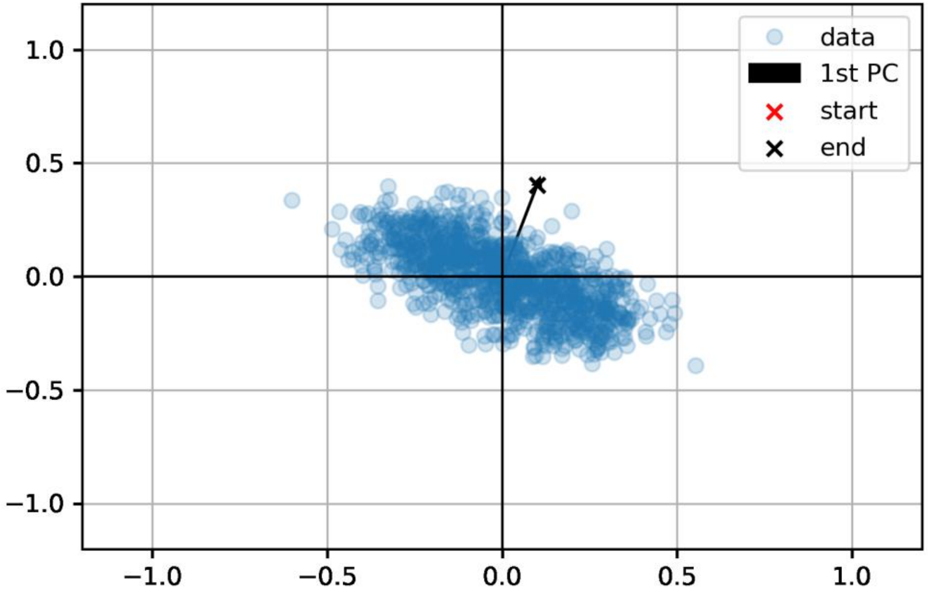
Simulation

Steps:

1. Data generation
2. Variable initialization
3. Iterate through data
   1. Compute pre-synaptic input
   2. Compute post-synaptic activation
   3. Compute Δw and update
4. Plot result

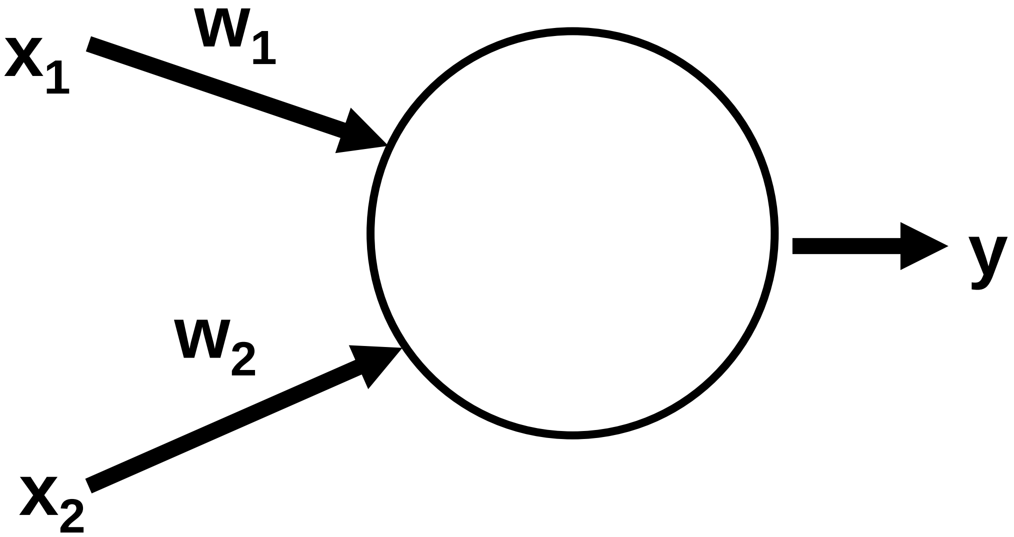


Simulation



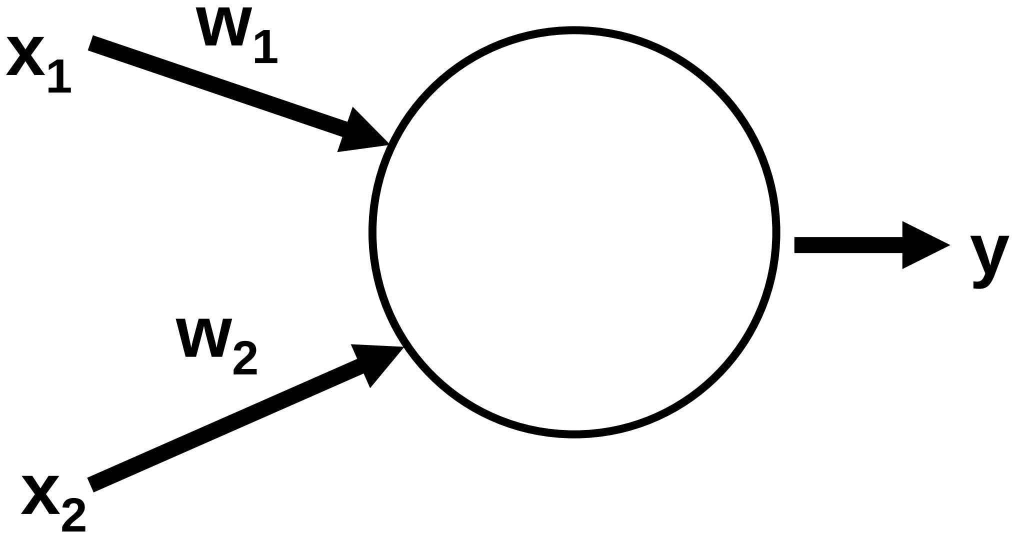
Supervised Learning

Perceptron



Unsupervised learning → no target emposed
Supervised learning → target emposed

Supervised Learning



$$\hat{y} = \sum_i w_i \cdot x_i$$

Target y: 2

Prediction $\hat{y}$: 1

Error = „Target – Prediction“ =
$= \frac{1}{2}(y - \hat{y})^2$

$$\frac{dE}{dw_i}E = \frac{dE}{dw_i}\frac{1}{2}(y - \hat{y})^2 = -(y - \hat{y})\frac{d\hat{y}}{dw_i}\sum_i w_i \cdot x_i = -(y - \hat{y}) \cdot x_i$$

Delta rule

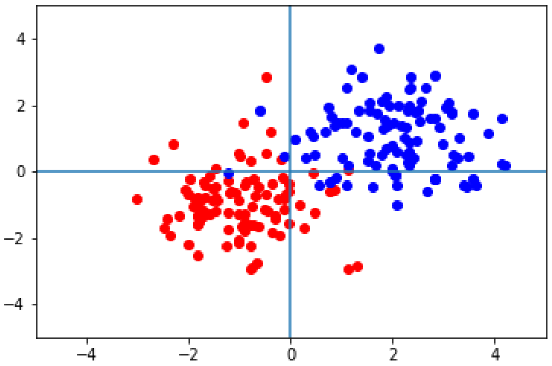
$$\Delta w_i = \alpha \cdot (y - \hat{y}) \cdot x_i$$

Simulation Delta Rule

Steps:

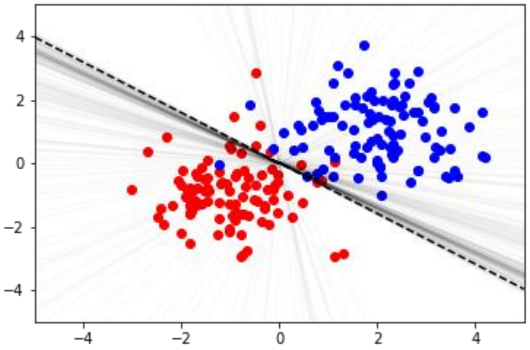
1. Data generation
2. Variable initialization
3. Iterating
    1. Compute prediction
    2. Update weights
4. Plotting

```python
np.random.seed(42)

N = 100

xs = []

for _ in range(N):
    x0 = np.random.randn()-1
    x1 = np.random.randn()-1
    xs.append((x0, x1))

    x0 = np.random.randn()+2
    x1 = np.random.randn()+1
    xs.append((x0, x1))

xs = np.asarray(xs)
```

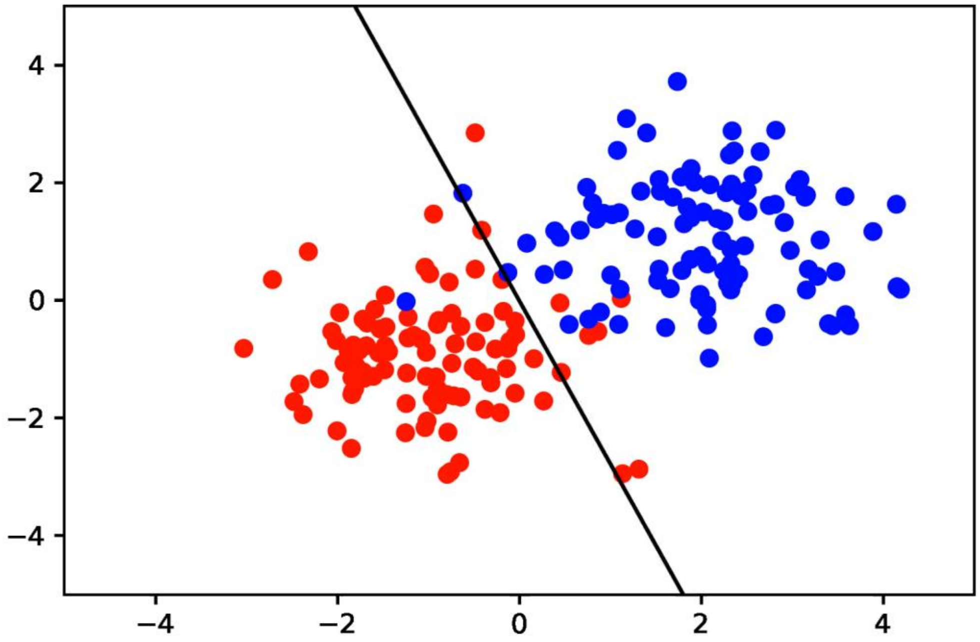



```python
w = np.random.randn(2)
eta = 0.01
```

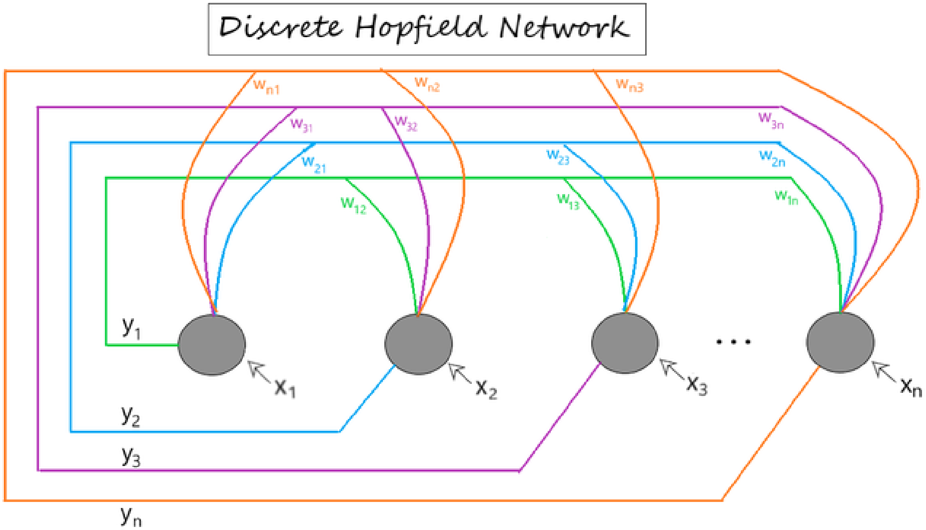
```python
for i in range(2*N):
    # Determine class/target
    t = -1 if i % 2 == 0 else 1

    # Compute prediction
    pred = w @ xs[i]

    # Update weights
    w = w + eta * (t-pred) * xs[i]

```



Simulation over time



Applied in deep neural networks



Binary (0,1)
Or bipolar (-1,1)
activations

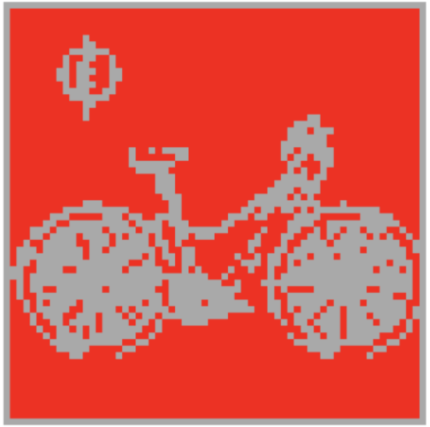
Learning rules should incorporate:
Local and incremental
→ e.g. Hebbian learning

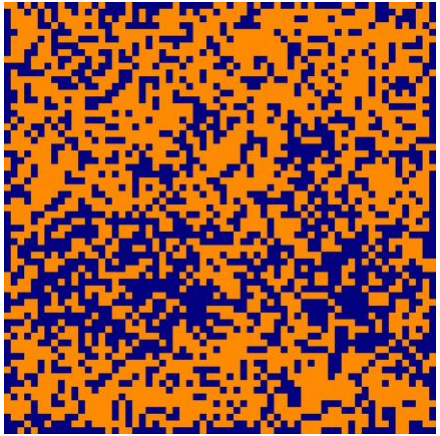
Minimizing the energy in the system

Hopfield networks

Capacity:
0.16 * n (n=units in Network)

$$C \cong \frac{n}{2\log_2 n}$$





Large images (1024x1024) → require 8M neurons (!!)

Storing continuous space?

64x64 px
→ Trained on 4096 nodes
→ 7.2% (256) nodes were updated per step

https://towardsdatascience.com/hopfield-networks-neural-memory-machines-4c94be821073

Hopfield networks - now

HOPFIELD NETWORKS IS ALL YOU NEED

Hubert Ramsauer* **Bernhard Schäfl*** **Johannes Lehner*** **Philipp Seidl***
Michael Widrich* **Thomas Adler*** **Lukas Gruber*** **Markus Holzleitner***
Milena Pavlović‡,§ **Geir Kjetil Sandve**§ **Victor Greiff**‡ **David Kreil**†
Michael Kopp† **Günter Klambauer*** **Johannes Brandstetter*** **Sepp Hochreiter***,†

*ELLIS Unit Linz, LIT AI Lab, Institute for Machine Learning, Johannes Kepler University Linz, Austria
†Institute of Advanced Research in Artificial Intelligence (IARAI)
‡Department of Immunology, University of Oslo, Norway
§Department of Informatics, University of Oslo, Norway

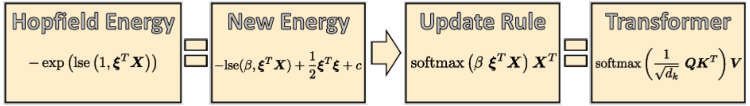


Figure 1: We generalize the energy of binary modern Hopfield networks to continuous states while keeping fast convergence and storage capacity properties. We also propose a new update rule that minimizes the energy. The new update rule is the attention mechanism of the transformer. Formulae are modified to express softmax as row vector. "="-sign means "keeps the properties".

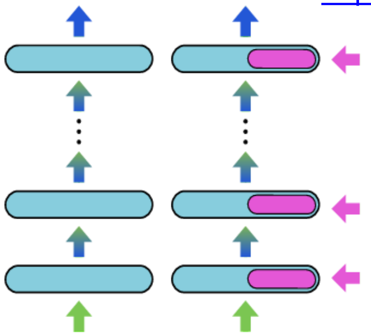
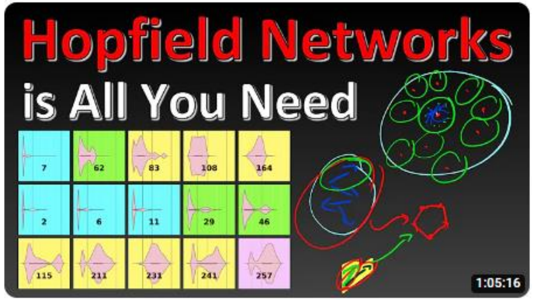


Figure 2: Left: A standard deep network with layers (■) propagates either a vector or a set of vectors from the input to the output. Right: A deep network, where layers (■) are equipped with associative memories via Hopfield layers (■).

https://www.youtube.com/watch?v=nv6oFDp6rNQ



Explanation
by Yannic Kilcher

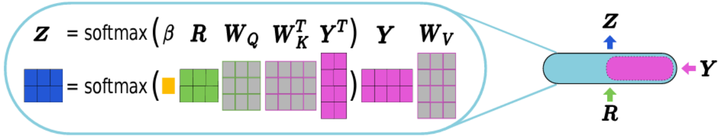
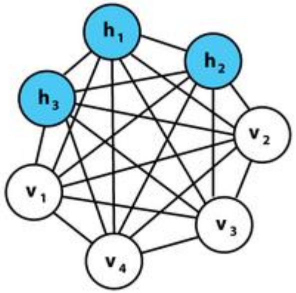


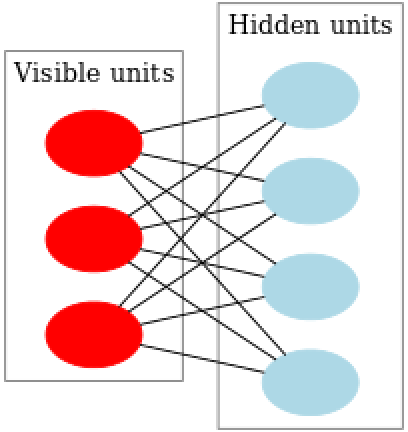
Figure 3: The layer `Hopfield` allows the association of two sets $\boldsymbol{R}$ (■) and $\boldsymbol{Y}$ (■). It can be integrated into deep networks that propagate sets of vectors. The Hopfield memory is filled with a set from either the input or previous layers. The output is a set of vectors $\boldsymbol{Z}$ (■).

Deep Belief Networks



Boltzmann
distribution
for sampling

Boltzmann machine

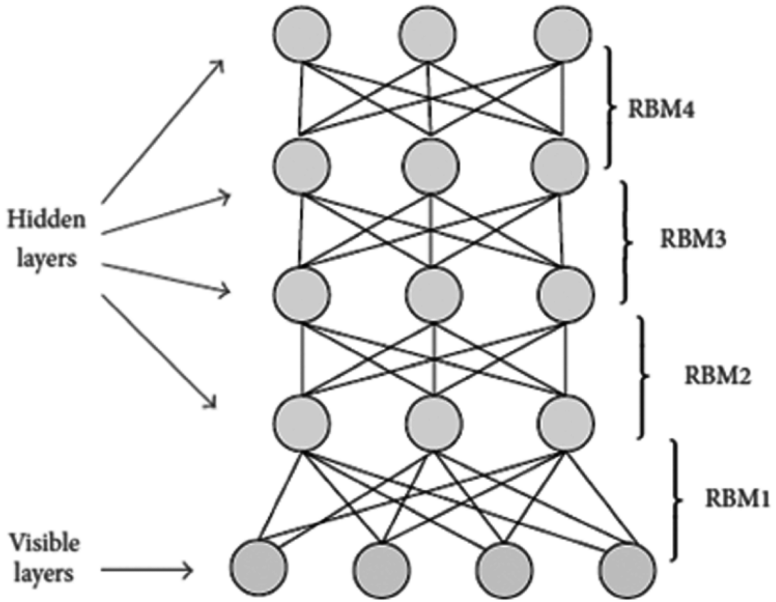


Restricted Boltzmann machine
(RBM) – no visible-visible and hidden-hidden connections
Training w/ contrastive divergence

Everything is connected to everything,
Energy-based model (EBM),
Similar to Hopfields.

Training:
minimizing KL-divergence

Deep Belief Networks



First round of training:
unsupervised w/ CD
RBM by RBM

Connected to output,
then **supervised** classification