[Tsinghua Big Data Summer Camp, 2016]

Deep Learning

Jun Zhu
dcszj@mail.tsinghua.edu.cn
http://bigml.cs.tsinghua.edu.cn/~jun
State Key Lab of Intelligent Technology & Systems
Tsinghua University

July 28, 2016

Why going deep?

- Data are often high-dimensional.

- There is a huge amount of structure in the data, but the structure is too complicated to be represented by a simple model.

- Insufficient depth can require more computational elements than architectures whose depth matches the task.

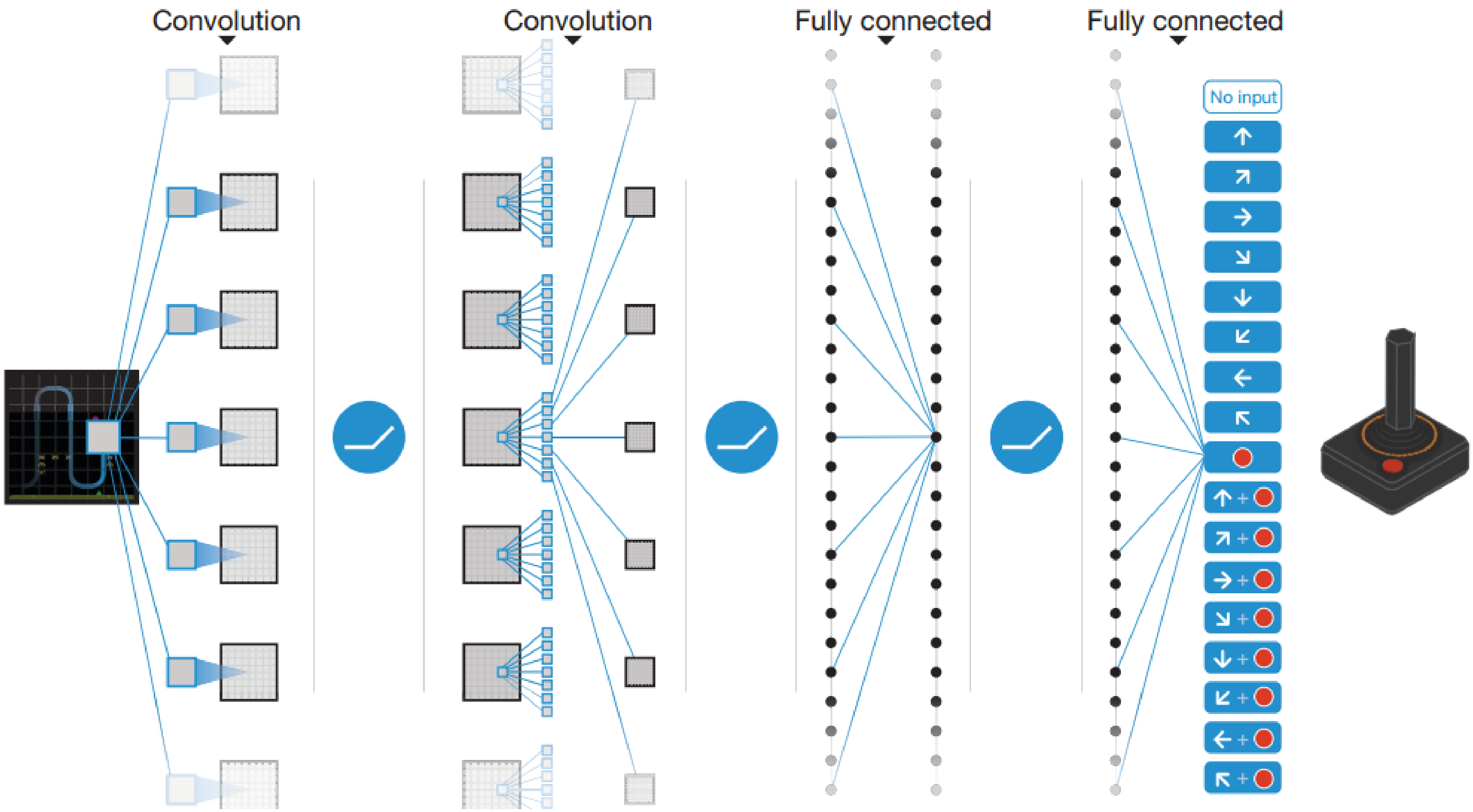
- Deep nets provide simpler but more descriptive models of many problems.

Microsoft's speech recognition system

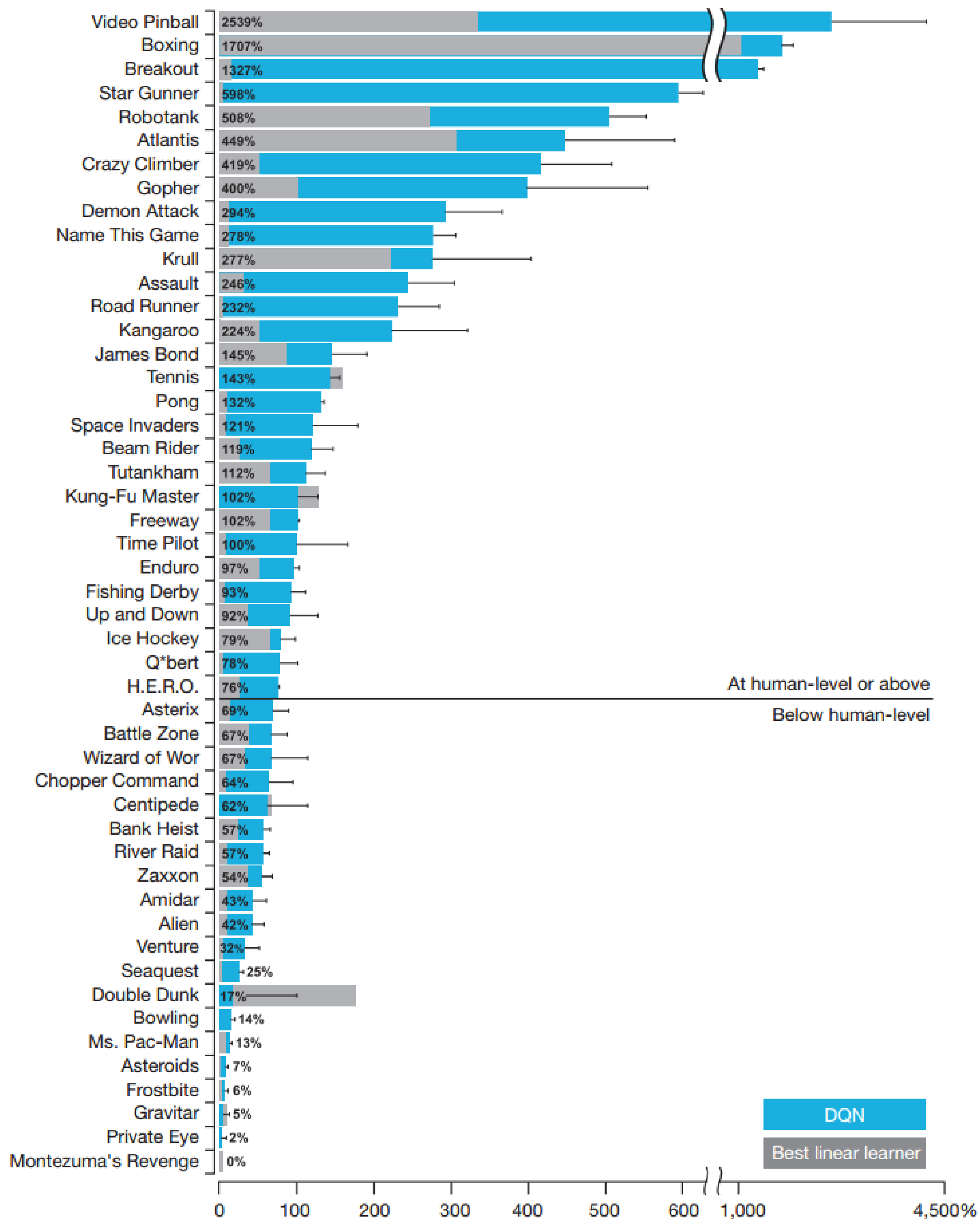
- http://v.youku.com/v_show/id_XNDc0MDY4ODI0.html

Human-Level Control via Deep RL

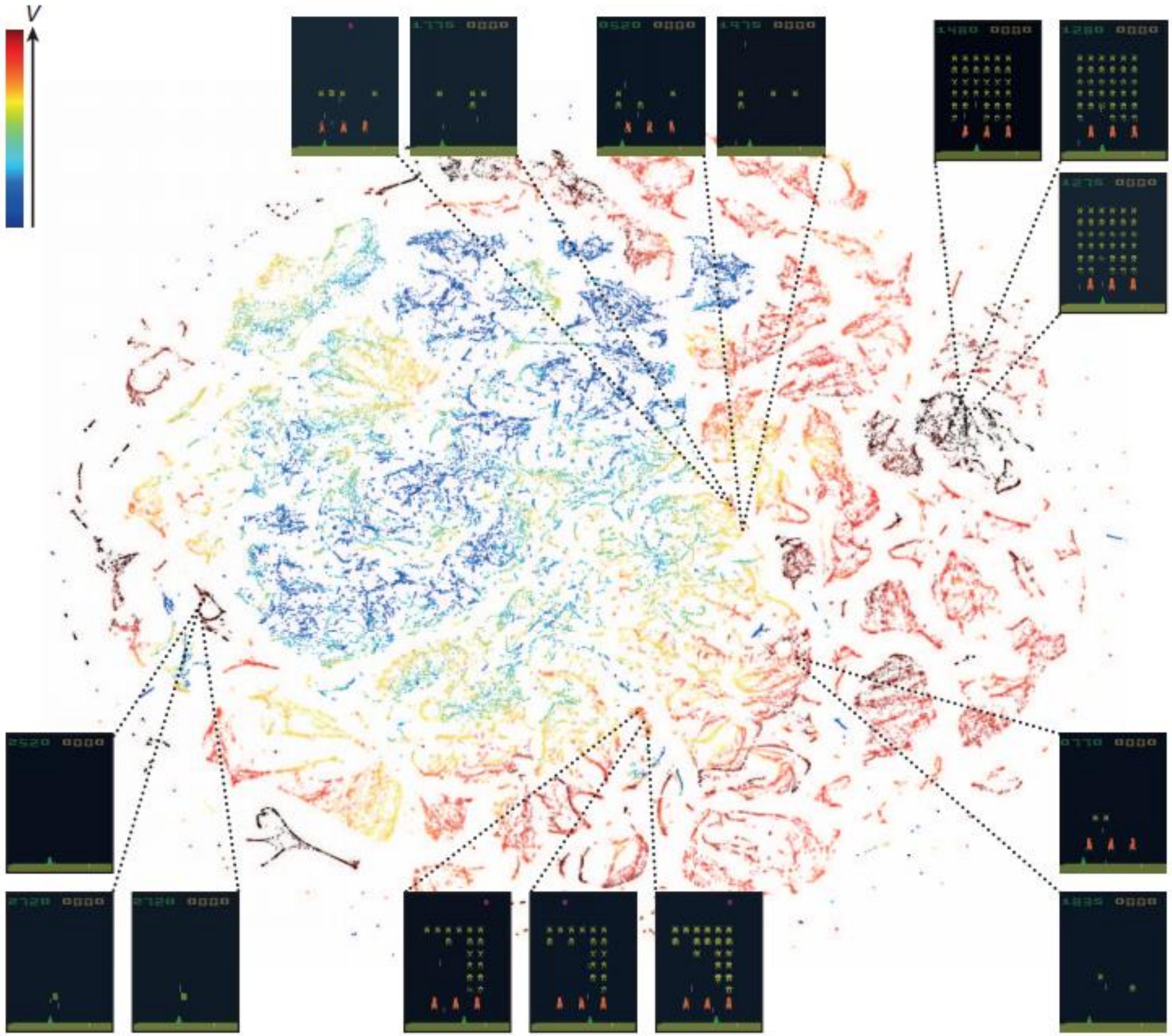
- Deep Q-network with human-level performance on A



[Mnih et al., Nature 518, 529–533, 2015]

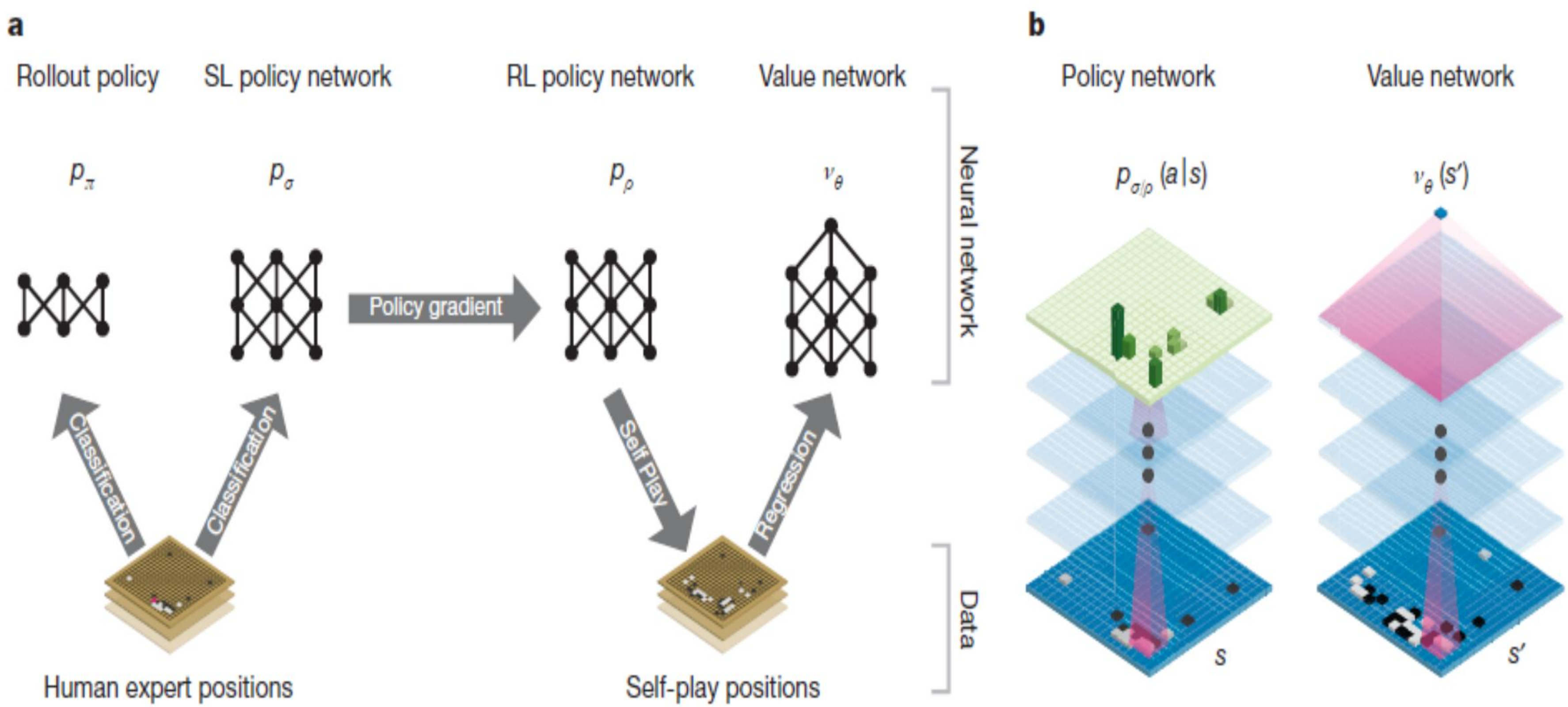


Video Pinball
Boxing
Breakout
Star Gunner
Robotank
Atlantis
Crazy Climber
Gopher
Demon Attack
Name This Game
Krull
Assault
Road Runner
Kangaroo
James Bond
Tennis
Pong
Space Invaders
Beam Rider
Tutankham
Kung-Fu Master
Freeway
Time Pilot
Enduro
Fishing Derby
Up and Down
Ice Hockey
Q*bert
H.E.R.O.
Asterix
Battle Zone
Wizard of Wor
Chopper Command
Centipede
Bank Heist
River Raid
Zaxxon
Amidar
Alien
Venture
Seaquest
Double Dunk
Bowling
Ms. Pac-Man
Asteroids
Frostbite
Gravitar
Private Eye
Montezuma's Revenge
2539%
1707%
1327%
598%
508%
449%
419%
400%
294%
278%
277%
246%
232%
224%
145%
143%
132%
121%
119%
112%
102%
102%
100%
97%
93%
92%
79%
78%
76%
69%
67%
67%
64%
62%
57%
57%
54%
43%
42%
32%
25%
17%
14%
13%
7%
6%
5%
2%
0%
At human-level or above
Below human-level
DQN
Best linear learner
0
100
200
300
400
500
600
1,000
4,500%


V

AlphaGo

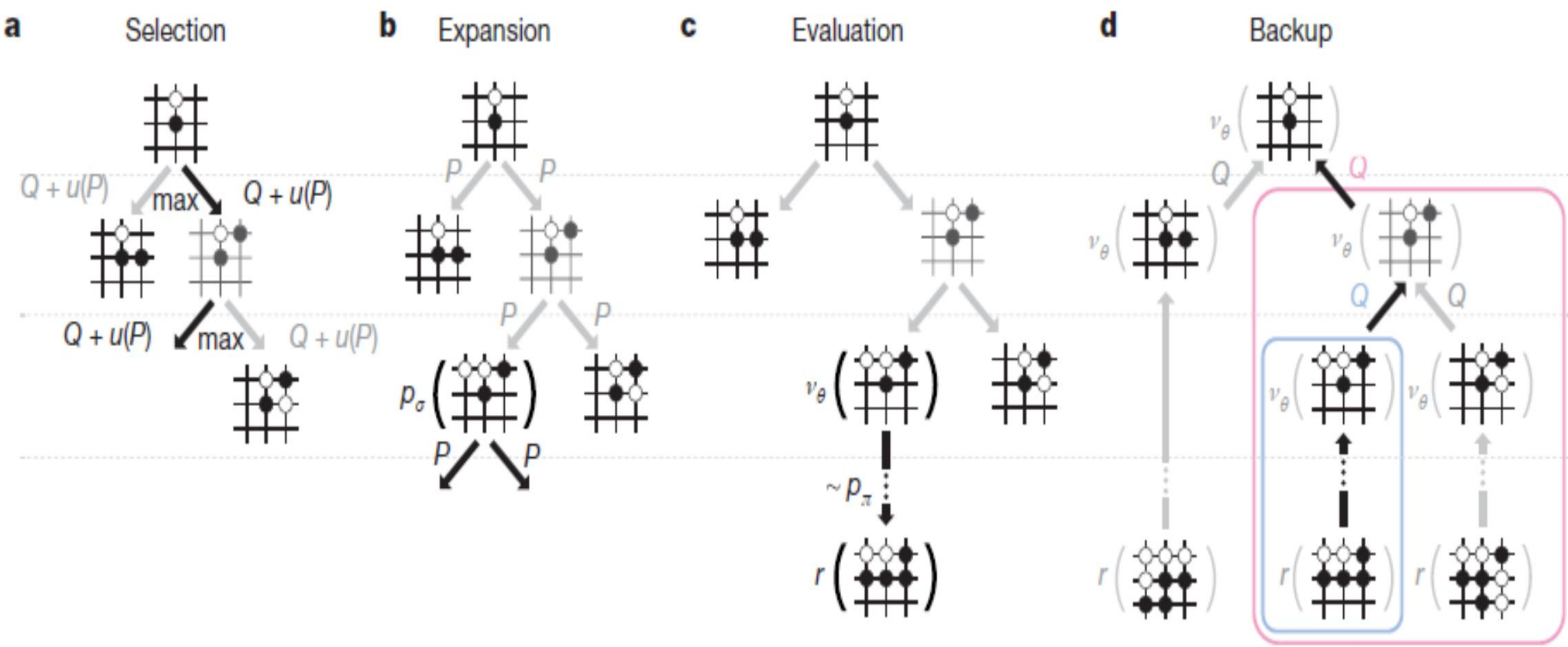
◈ Neural network training pipeline and architecture



[Silver et al., Mastering the game of Go with deep neural networks and tree search. Nature, 484(529), 2016]

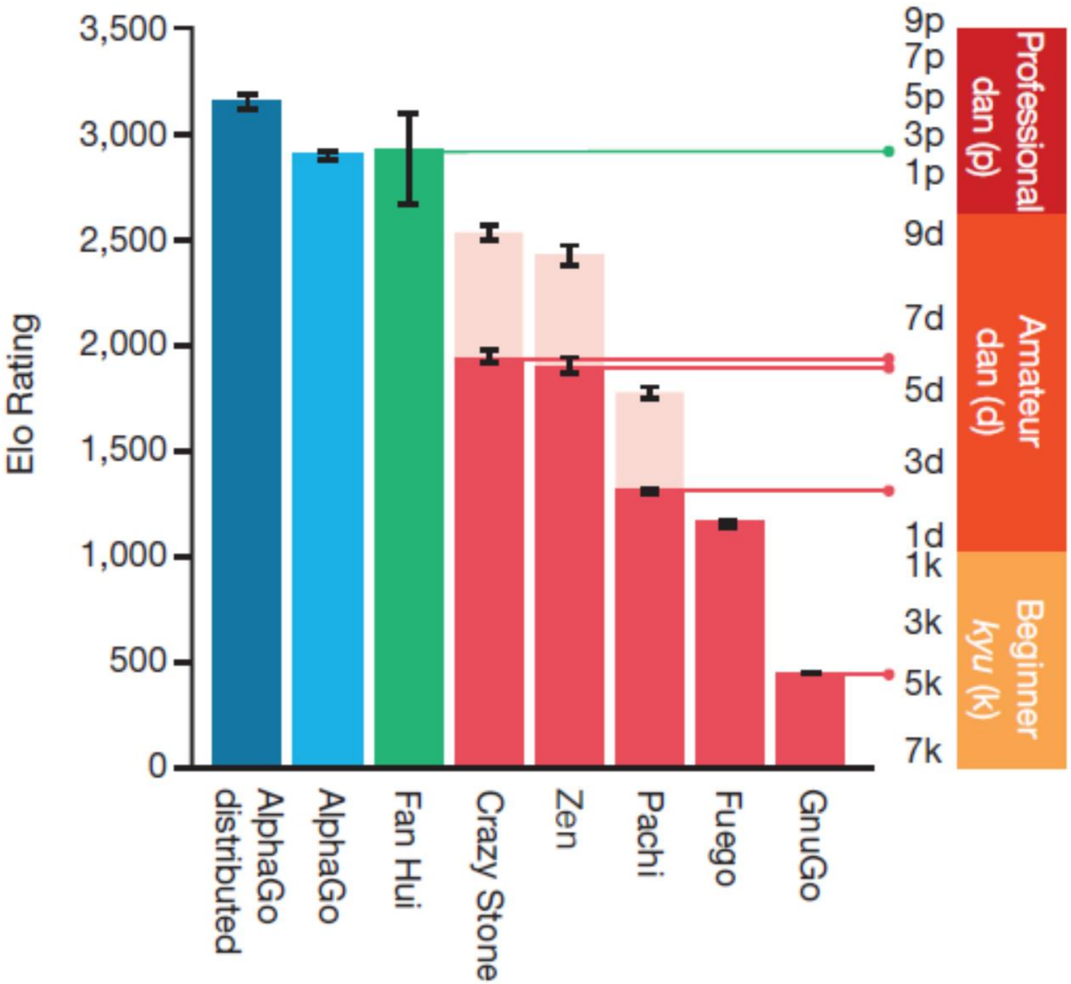
AlphaGo

- Monte Carlo tree search



[Silver et al., Mastering the game of Go with deep neural networks and tree search. Nature, 484(529), 2016]

AlphaGo

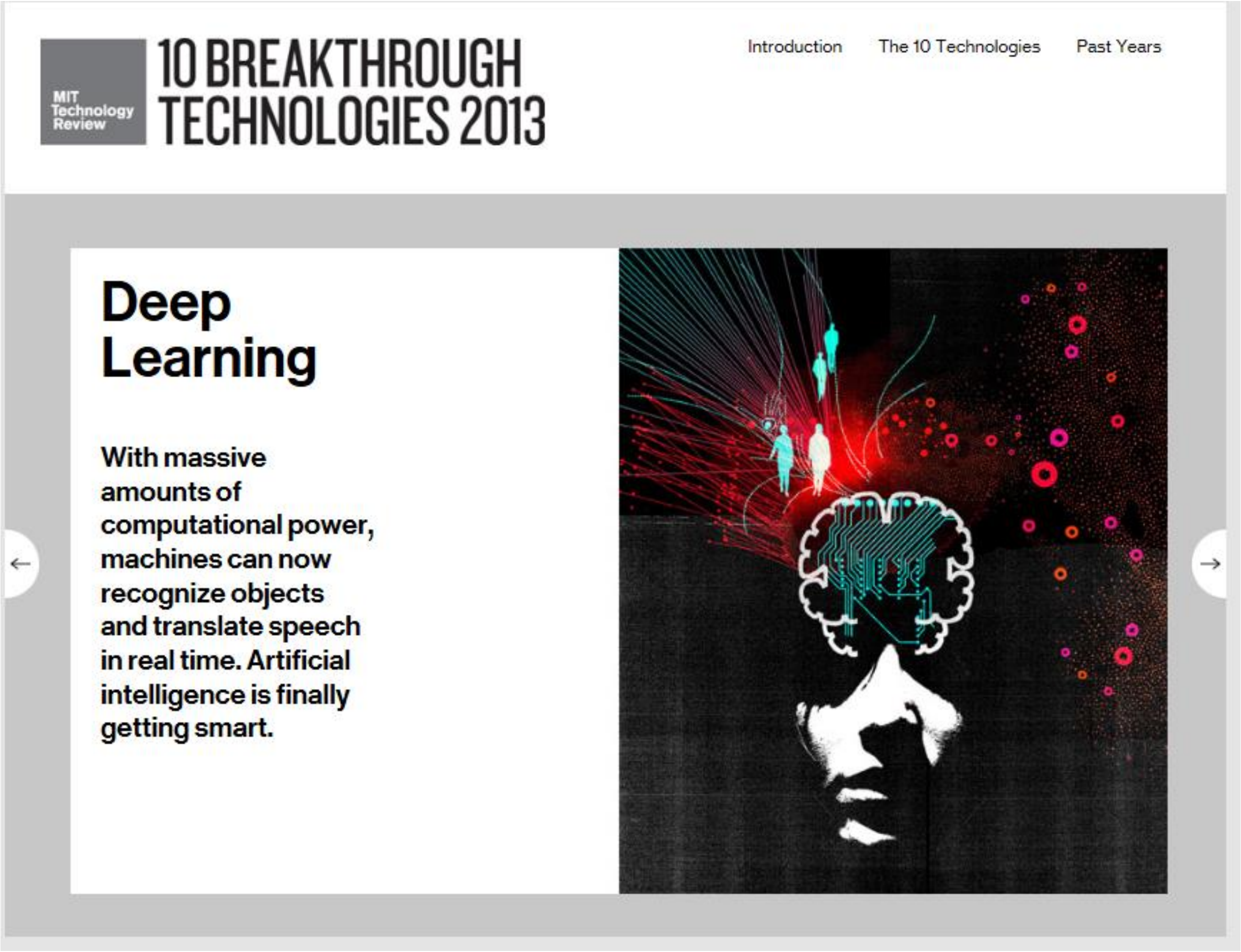


[Silver et al., Mastering the game of Go with deep neural networks and tree search. Nature, 484(529), 2016]

Go

- A combinatorial game
  - Two players, deterministic, zero-sum, perfect information

MIT 10 Breakthrough Tech 2013



http://www.technologyreview.com/featuredstory/513696/deep-learning/

Deep Learning in industry


Driverless car


Face identification


Speech recognition

Web search

…

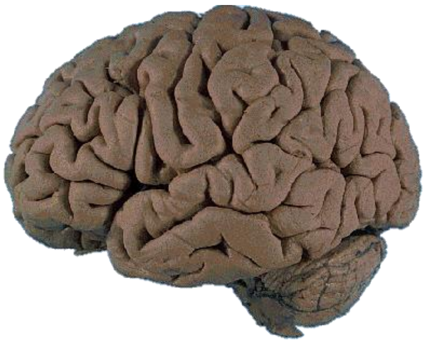
Google Alibaba.com HUAWEI intel

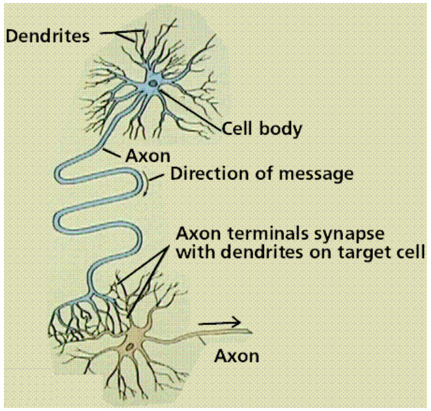
Microsoft Baidu 百度 Tencent 腾讯 AMD

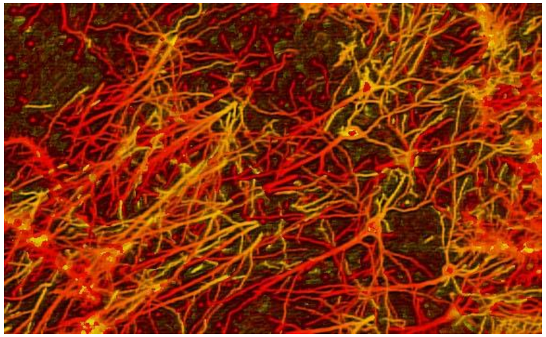
…

Deep Learning Models

How brains seem to do computing?





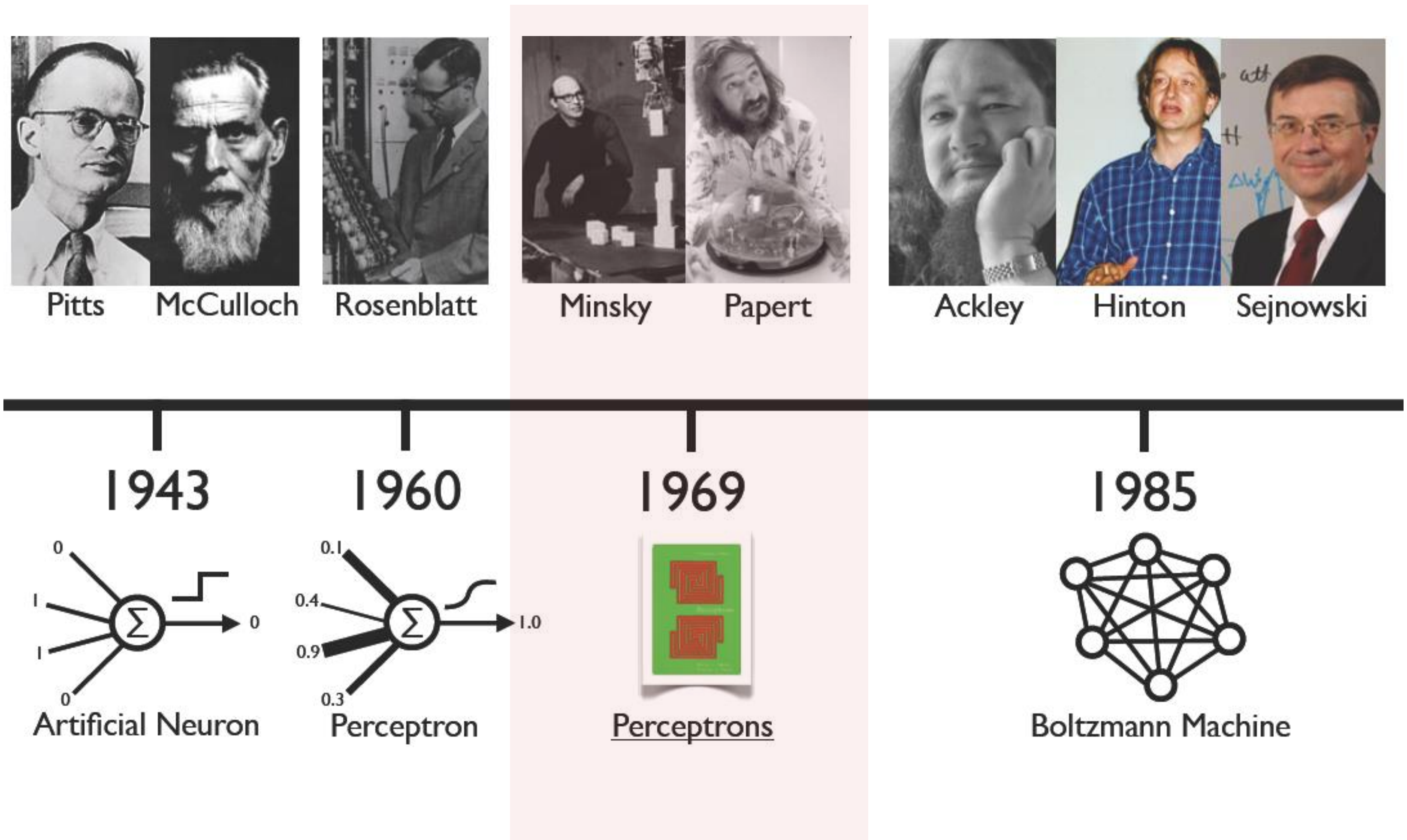


The business end of this is made of lots of these joined in networks like this

Much of our own "computations" are performed in/by this network

Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes

History of neural networks



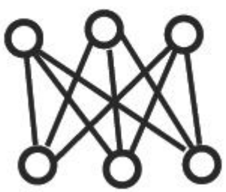
History of neural networks

Smolensky

Hinton

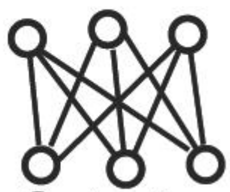
Hinton et al.

1986



Harmoniums
(Restricted Boltzmann Machine)

2002

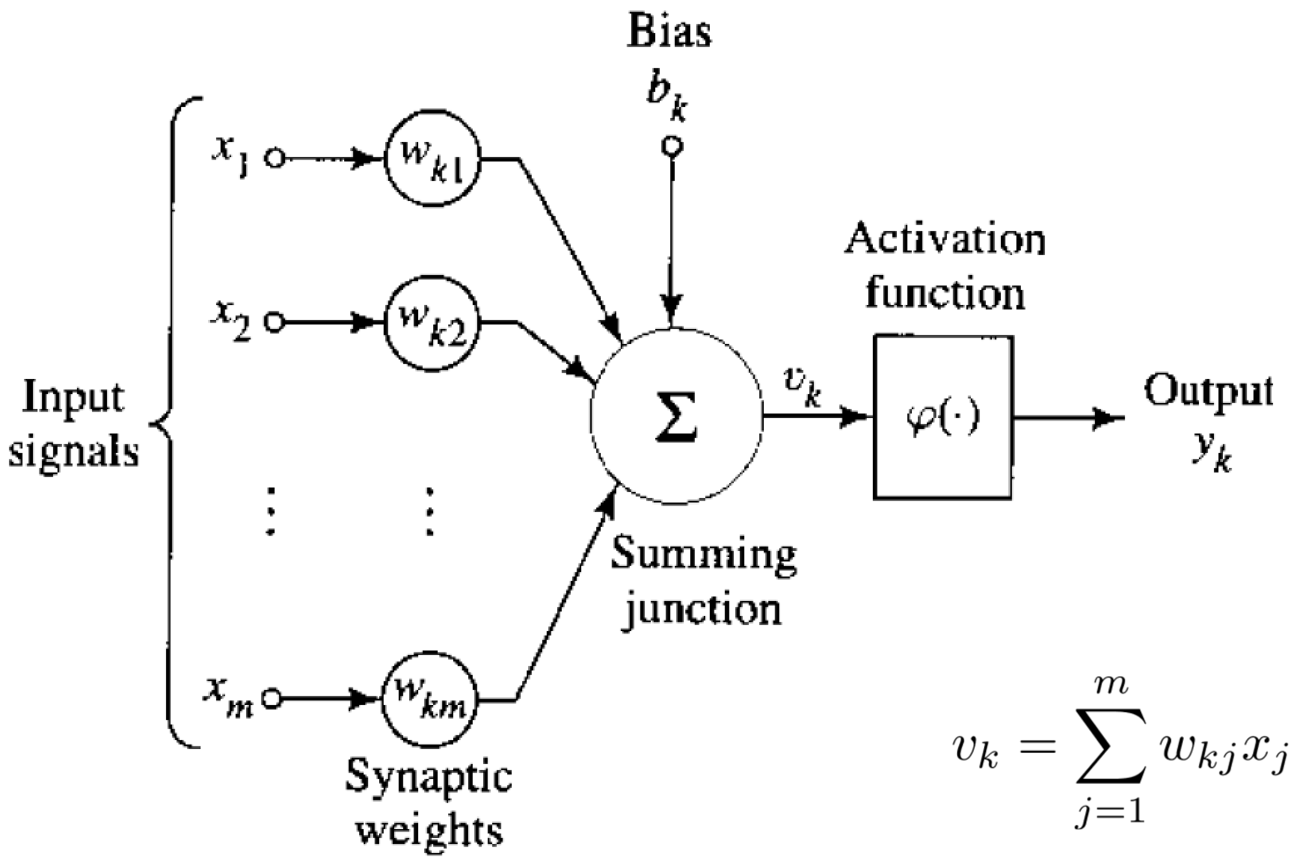


Contrastive
Divergence

2006

Deep Belief
Networks

Model of a neuron

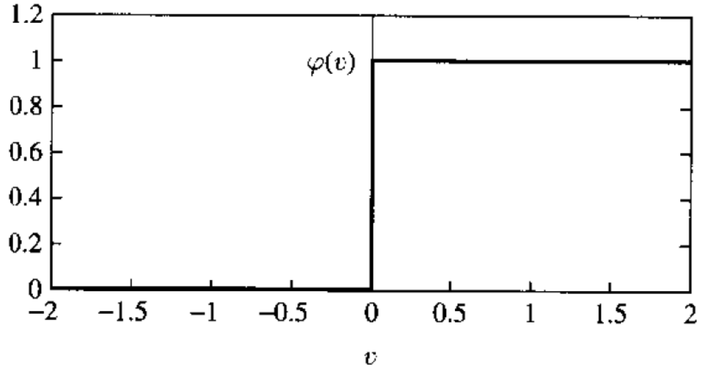


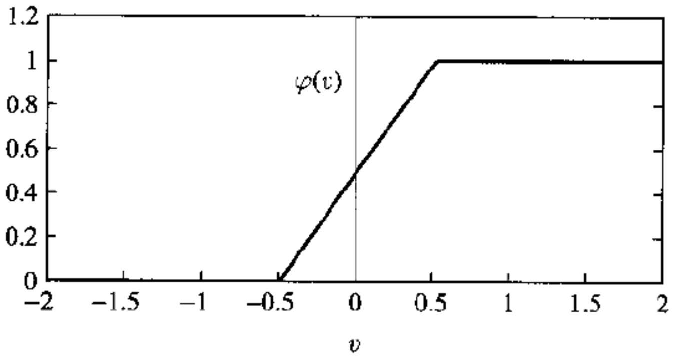
$$v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k$$

$$y_k = \psi(v_k)$$

Activation function

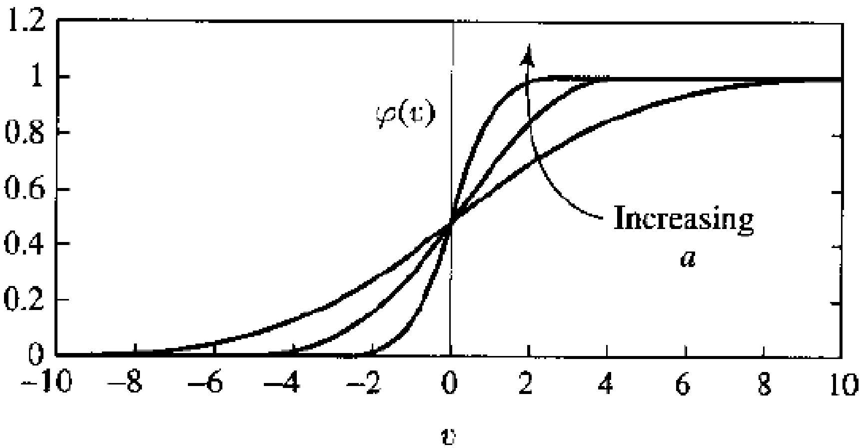
- Threshold function & piecewise linear function:





- Sigmoid function

$$\psi_\alpha(v) = \frac{1}{1+\exp(-\alpha v)}$$



$a \to \infty$: step function

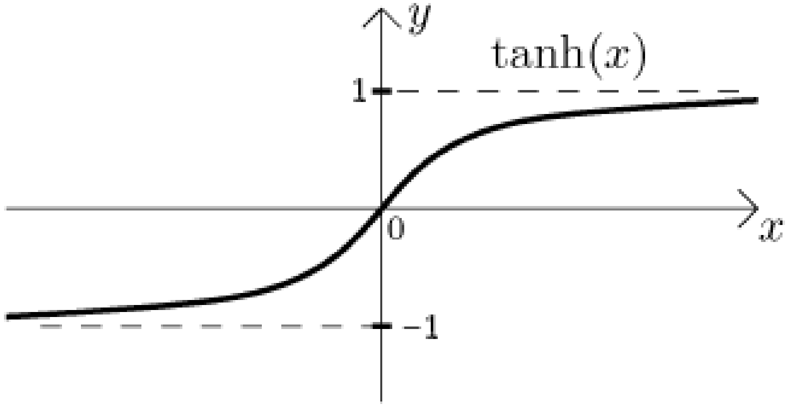
Activation function with negative values

- Threshold function & piecewise linear function:

$$\mathrm{sgn}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

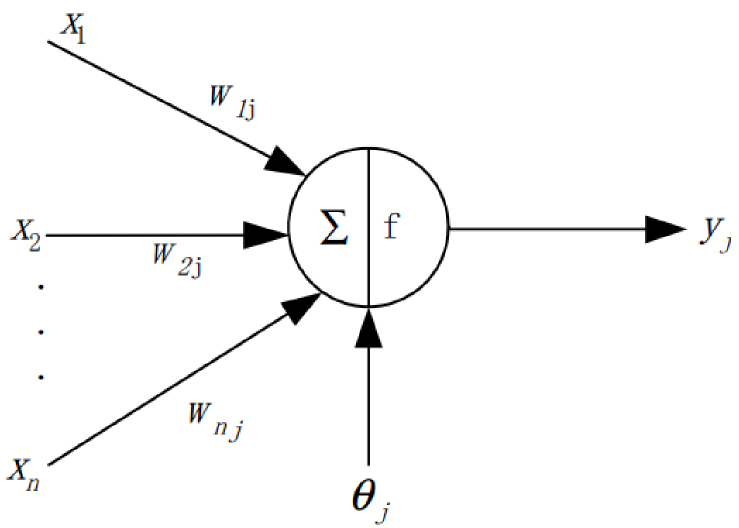
- Hyperbolic tangent function

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



McCulloch & Pitts's Artificial Neuron

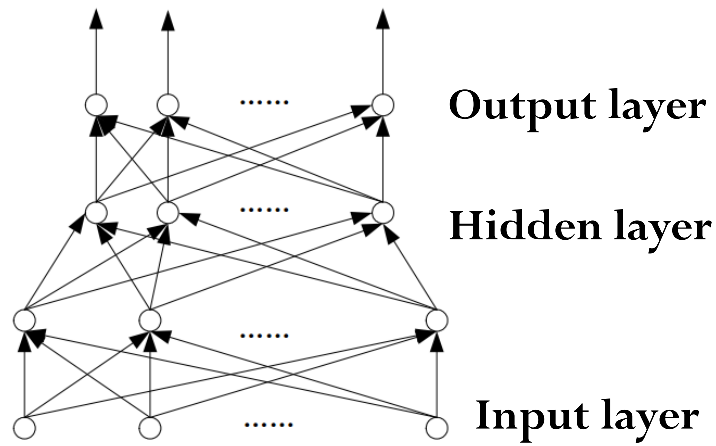
- The first model of artificial neurons in 1943
  - Activation function: a threshold function



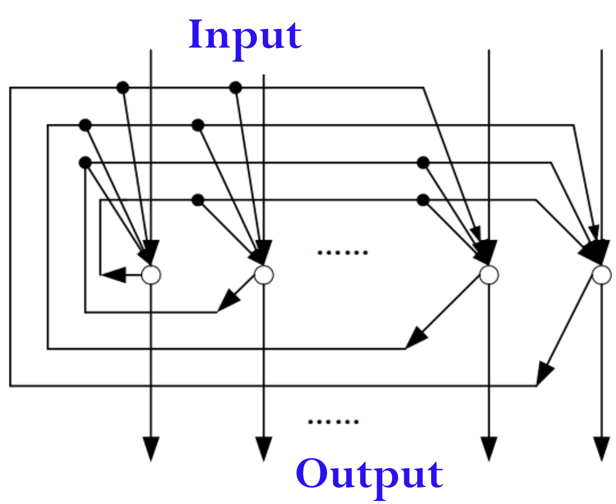
$$y_j = \mathrm{sgn}\left(\sum_i w_{ij} x_i - \theta_j \right)$$

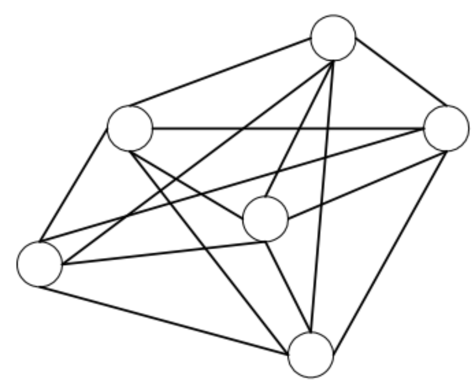
Network Architecture

- Feedforward networks



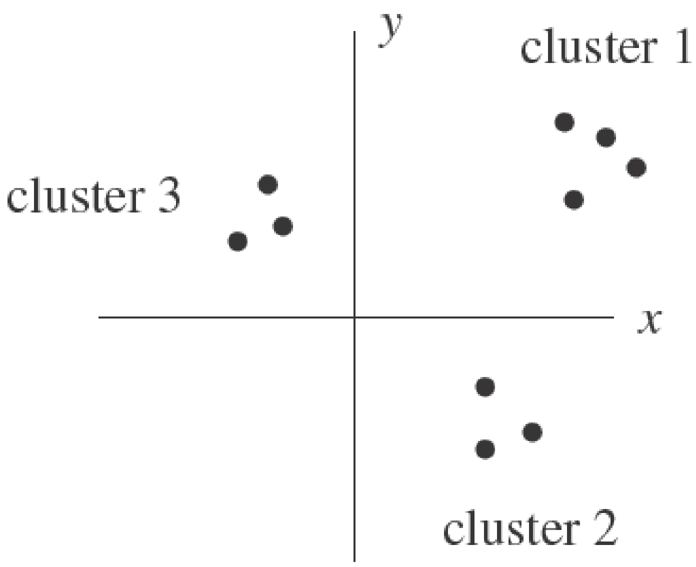
- Recurrent networks

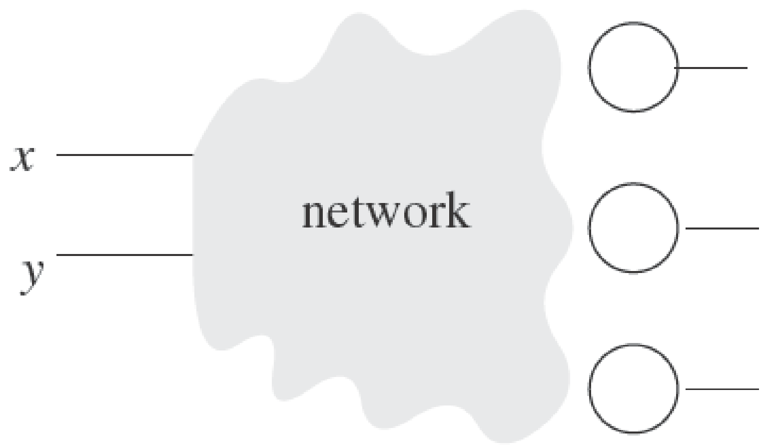




Learning Paradigms

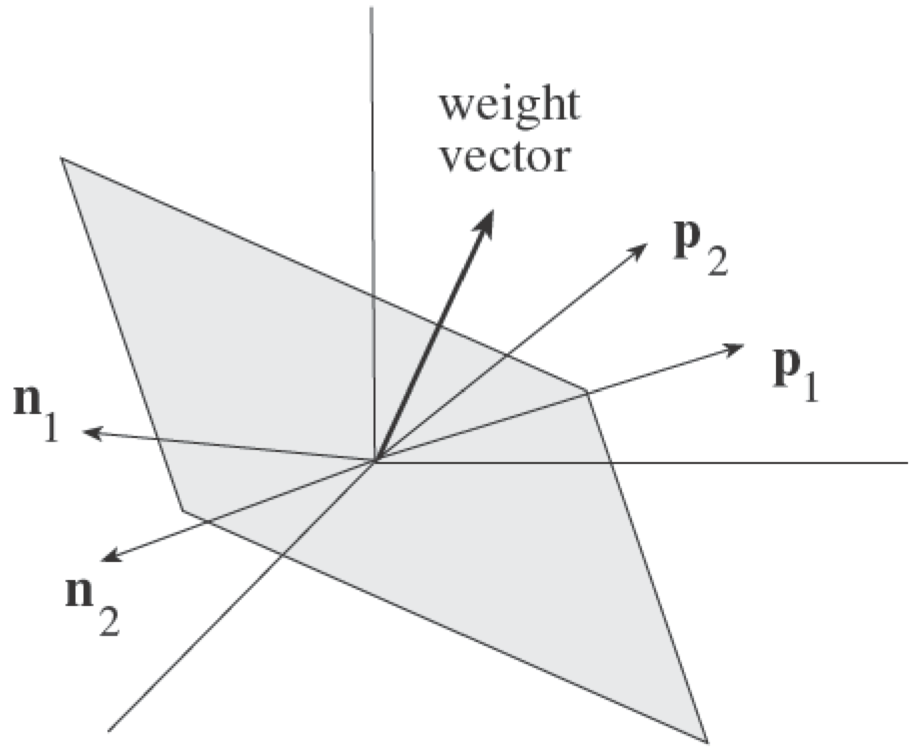
- Unsupervised learning (learning without a teacher)
  - Example: clustering





Learning Paradigms

- Supervised Learning (learning with a teacher)
  - For example, classification: learns a separation plane

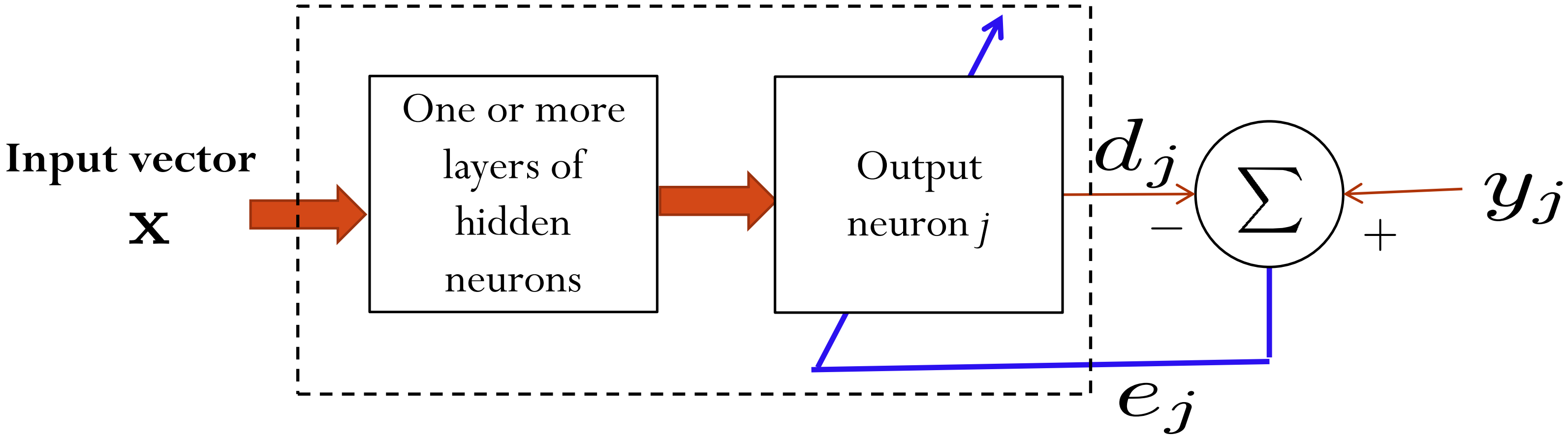


Learning Rules

- Error-correction learning
- Competitive learning
- Hebbian learning
- Boltzmann learning
- Memory-based learning
  - Nearest neighbor, radial-basis function network

Error-correction learning

- The generic paradigm:



- Error signal:

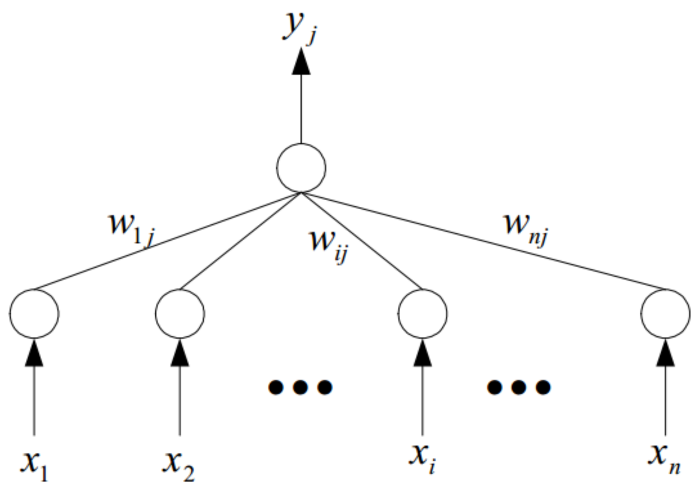
$$e_j = y_j - d_j$$

- Learning objective:

$$\min_{\mathbf{w}} R(\mathbf{w};\mathbf{x}) := \frac{1}{2}\sum_j e_j^2$$

Example: Perceptron

- One-layer feedforward network based on error-correction learning (no hidden layer):



  - Current output (at iteration t):

$$d_j = (\mathbf{w}_t^j)^\top \mathbf{x}$$

  - Update rule (*exercise?*):

$$\mathbf{w}_{t+1}^j = \mathbf{w}_t^j + \eta(y_j - d_j)\mathbf{x}$$

Perceptron for classification

- Consider a single output neuron

- Binary labels:

$$y \in \{+1, -1\}$$

- Output function:

$$d = \mathrm{sgn}\left(\mathbf{w}_t^\top \mathbf{x}\right)$$

- Apply the error-correction learning rule, we get … (next slide)

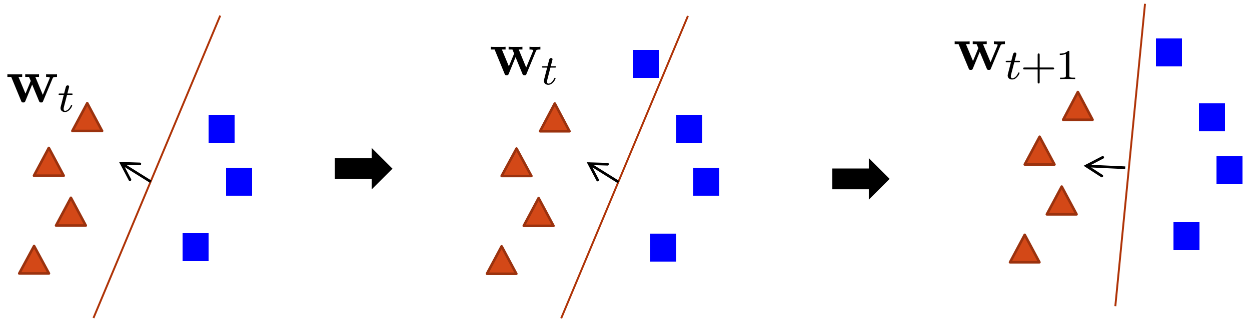
Perceptron for Classification

- Set $\mathbf{w}_1 = 0$ and t=1; scale all examples to have length 1 (doesn't affect which side of the plane they are on)
- Given example x, predict positive *iff*

$$\mathbf{w}_t^\top \mathbf{x} > 0$$

- If a mistake, update as follows
  - Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta_t \mathbf{x}$
  - Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \mathbf{x}$

$t \leftarrow t + 1$



Convergence Theorem

- For linearly separable case, the perceptron algorithm will converge in a finite number of steps

Mistake Bound

- Theorem:
  - *Let* $\mathcal{S}$ *be a sequence of labeled examples consistent with a linear threshold function* $\mathbf{w}_*^\top \mathbf{x} > 0$ *, where* $\mathbf{w}_*$ *is a unit-length vector.*
  - *The number of mistakes made by the online Perceptron algorithm is at most* $(1/\gamma)^2$*, where*

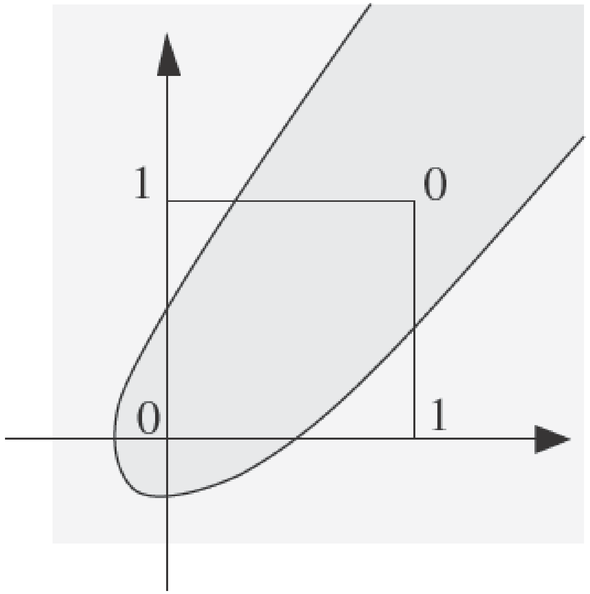
$$\gamma = \min_{\mathbf{x} \in \mathcal{S}} \frac{|\mathbf{w}_*^\top \mathbf{x}|}{\|\mathbf{x}\|}$$

  - *i.e.: if we scale examples to have length 1, then* γ *is the minimum distance of any example to the plane* $\mathbf{w}_*^\top \mathbf{x} = 0$
  - γ *is often called the "margin" of* $\mathbf{w}_*$ *; the quantity* $\frac{\mathbf{w}_*^\top \mathbf{x}}{\|\mathbf{x}\|}$ *is the cosine of the angle between* $\mathbf{x}$ *and* $\mathbf{w}_*$

Deep Nets

- Multi-layer Perceptron
- CNN
- Auto-encoder
- RBM
- Deep belief nets
- Deep recurrent nets

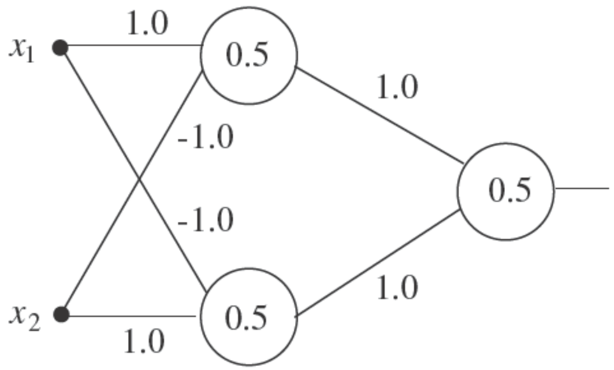
XOR Problem



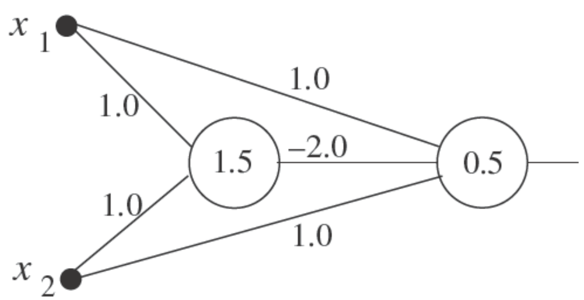
- Single-layer perceptron can't solve the problem

XOR Problem

- A network with 1-layer of 2 neurons works for XOR:
  - threshold activation function



  - Many alternative networks exist (not layered)



Multilayer Perceptrons

- Computatonal limitations of single-layer Perceptron by Minsky & Papert (1969)

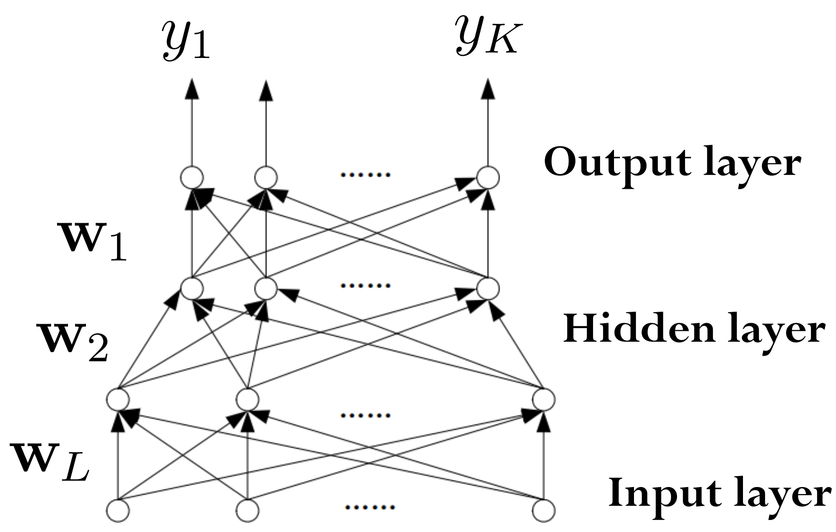
- Multilayer Perceptrons:
  - Multilayer feedforward networks with an error-correction learning algorithm, known as error ***back-propagation***
  - A generalization of single-layer percetron to allow nonlinearity

Backpropagation

- Learning as loss minimization

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{2} \sum_j e_j^2(\mathbf{x})$$

$$e_j = y_j - d_j$$



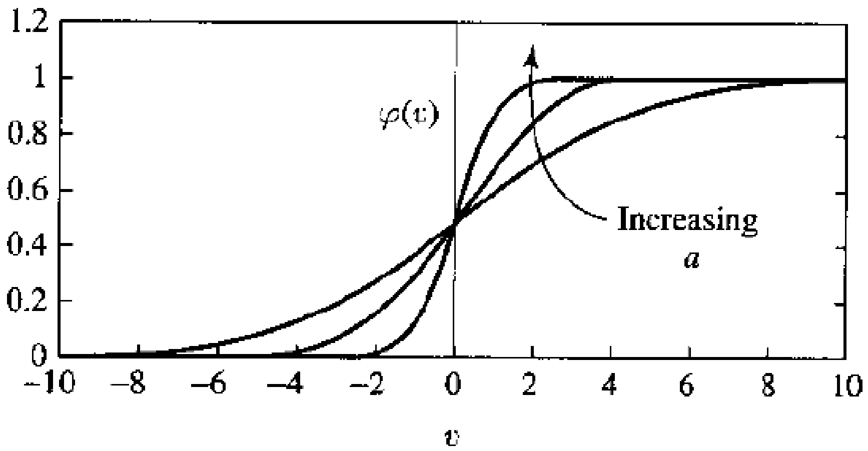
- Learning with gradient descent

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \lambda_t \nabla R(\mathbf{w}; \mathcal{D})$$

Backpropagation

- Step function in perceptrons is non-differentiable
- Differentiable activation functions are needed to calculate gradients, e.g., sigmoid:

$$\psi_\alpha(v) = \frac{1}{1 + \exp(-\alpha v)}$$



Backpropagation

- Derivative of a sigmoid function ($\alpha = 1$)

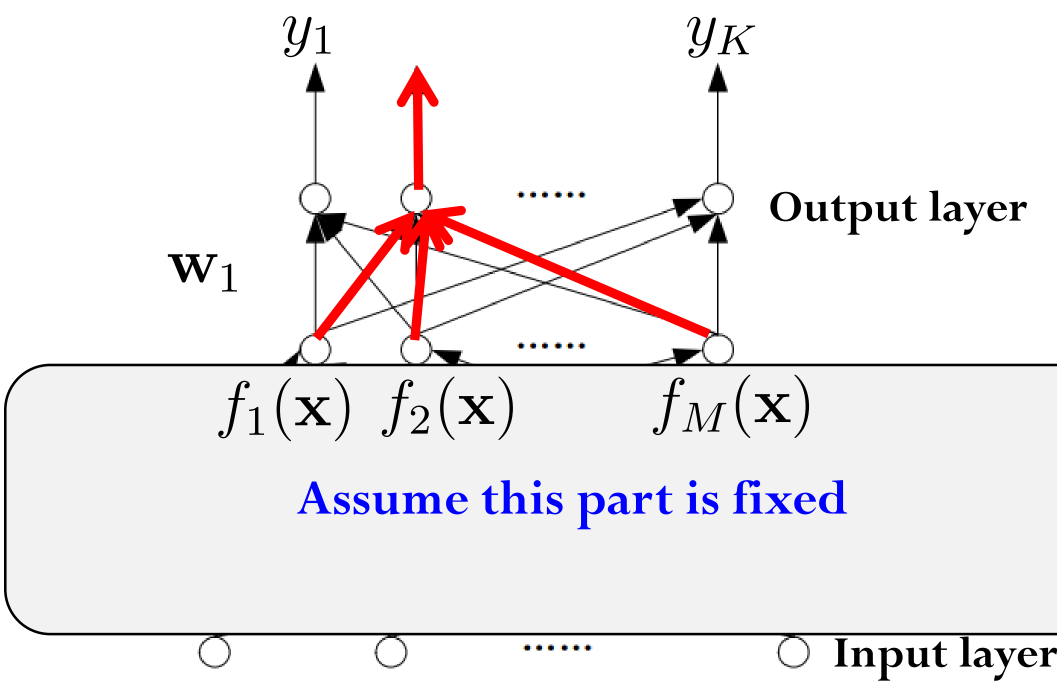
$$\nabla_v \psi(v) = \frac{e^{-v}}{(1+e^{-v})^2} = \psi(v)(1-\psi(v))$$

  - Notice about the small scale of the gradient
  - Gradient vanishing issue

- Many other activation functions examined

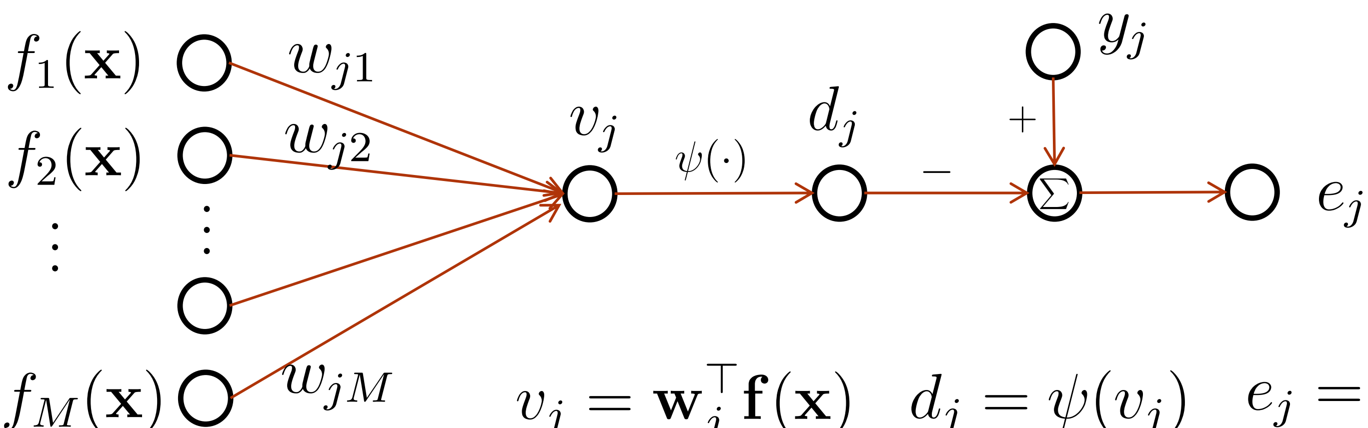
Gradient computation at output layer

- Output neurons are separate:



Gradient computation at output layer

- Signal flow:



$$v_j = \mathbf{w}_j^\top \mathbf{f}(\mathbf{x}) \quad d_j = \psi(v_j) \quad e_j = y_j - d_j$$

$$R_j = \frac{1}{2} e_j^2$$

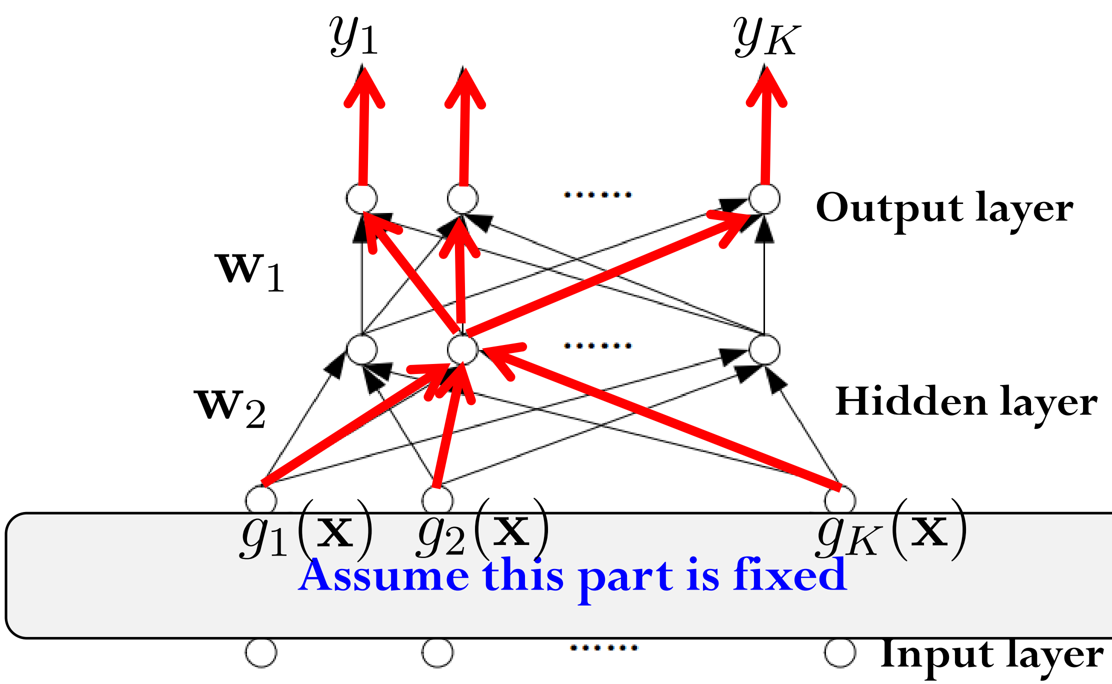
$$R = \frac{1}{2} \sum_j e_j^2$$

$$\begin{aligned} \nabla_{w_{ji}} R &= \frac{\partial R_j}{\partial e_j} \frac{\partial e_j}{\partial d_j} \frac{\partial d_j}{\partial v_j} \frac{\partial v_j}{\partial w_{ji}} \\ &= e_j \cdot (-1) \cdot \psi'(v_j) \cdot f_i(\mathbf{x}) \\ &= -\boxed{e_j \psi'(v_j)} f_i(\mathbf{x}) \end{aligned}$$

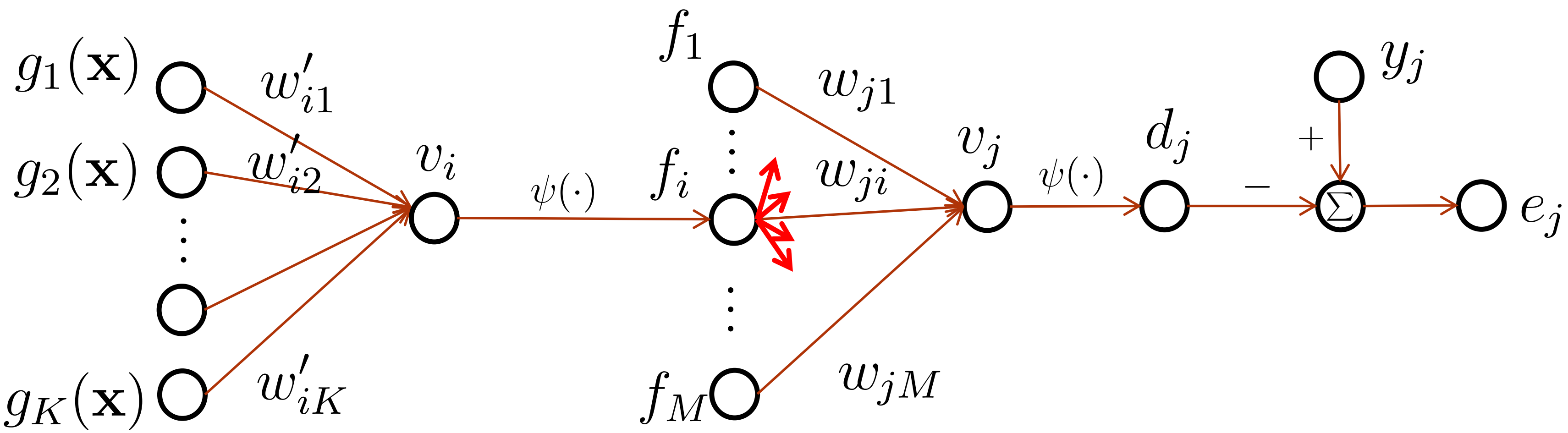
Local gradient: $\delta_j = -\dfrac{\partial R}{\partial v_j}$

Gradient computation at hidden layer

- Output neurons are NOT separate:



Gradient computation at hidden layer



$v_i = (\mathbf{w}'_i)^\top \mathbf{g} \quad f_i = \psi(v_i) \quad v_j = \mathbf{w}_j^\top \mathbf{f} \quad d_j = \psi(v_j) \quad e_j = y_j - d_j$

$$R_j = \frac{1}{2} e_j^2$$

$$R = \frac{1}{2} \sum_j e_j^2$$

$$\nabla_{w'_{ik}} R = \sum_j \frac{\partial R_j}{\partial e_j} \frac{\partial e_j}{\partial d_j} \frac{\partial d_j}{\partial v_j} \frac{\partial v_j}{\partial f_i} \frac{\partial f_i}{\partial v_i} \frac{\partial v_i}{\partial w'_{ik}}$$

$$= -\sum_j e_j \psi'(v_j) w_{ji} \psi'(v_i) g_k(\mathbf{x})$$

$$= -\boxed{\sum_j \delta_j w_{ji} \psi'(v_i)} g_k(\mathbf{x})$$

Local gradient:

$$\delta_i = -\frac{\partial R}{\partial v_i}$$

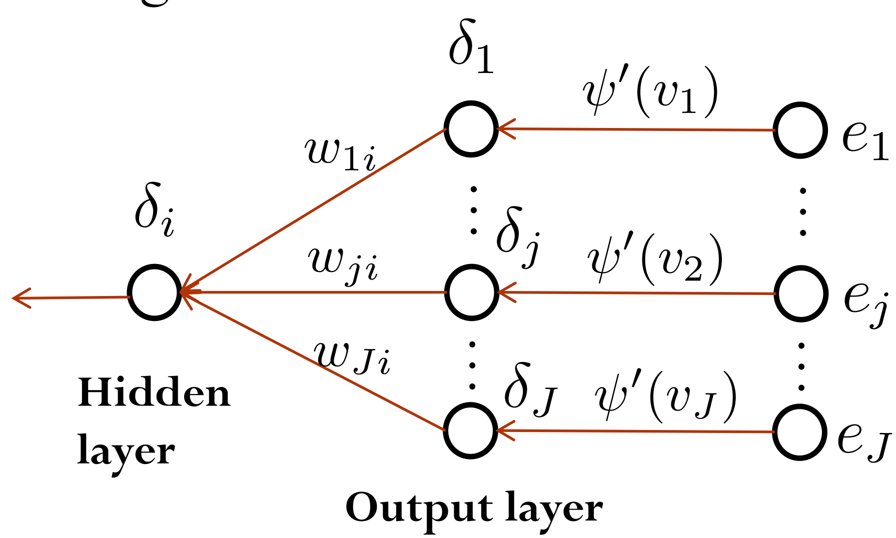
Back-propagation formula

- The update rule of **local gradients**:
  - for hidden neuron *i*:

$$\delta_i = \psi'(v_i) \sum_j \delta_j w_{ji}$$

Only depends on the activation function at hidden neuron i

- Flow of error signal:



Back-propagation formula

- The update rule of weights:
  - Output neuron:

$$\Delta w_{ji} = \lambda \cdot \delta_j \cdot f_i(\mathbf{x})$$

  - Hidden neuron:

$$\Delta w'_{ik} = \lambda \cdot \delta_i \cdot g_k(\mathbf{x})$$

$$\begin{pmatrix} Weight \\ correction \\ \Delta w_{ji} \end{pmatrix} = \begin{pmatrix} learning \\ rate \\ \lambda \end{pmatrix} \cdot \begin{pmatrix} local \\ gradient \\ \delta_j \end{pmatrix} \cdot \begin{pmatrix} input\ signal \\ of\ neuron\ j \\ v_i \end{pmatrix}$$

Two Passes of Computation

- Forward pass
  - Weights fixed
  - Start at the first hidden layer
  - Compute the output of each neuron
  - End at output layer

- Backward pass
  - Start at the output layer
  - Pass error signal backward through the network
  - Compute local gradients

Stopping Criterion

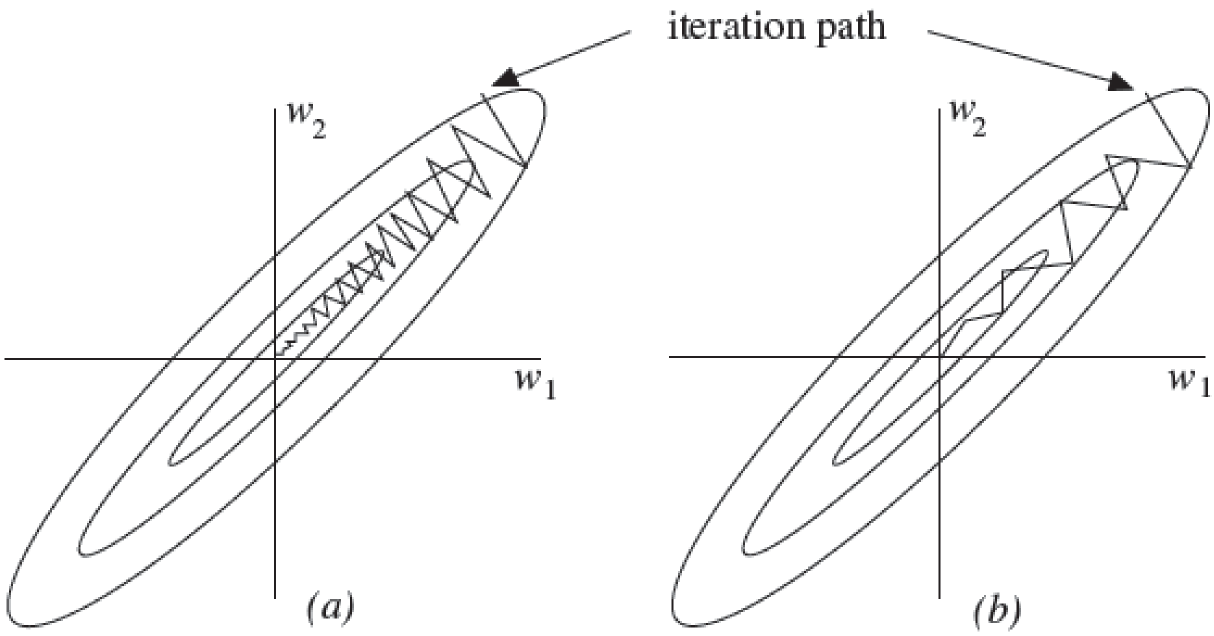
- No general rules

- Some reasonable heuristics:
  - The norm of gradient is small enough
  - The number of iterations is larger than a threshold
  - The training error is stable
  - …

Improve Backpropagation

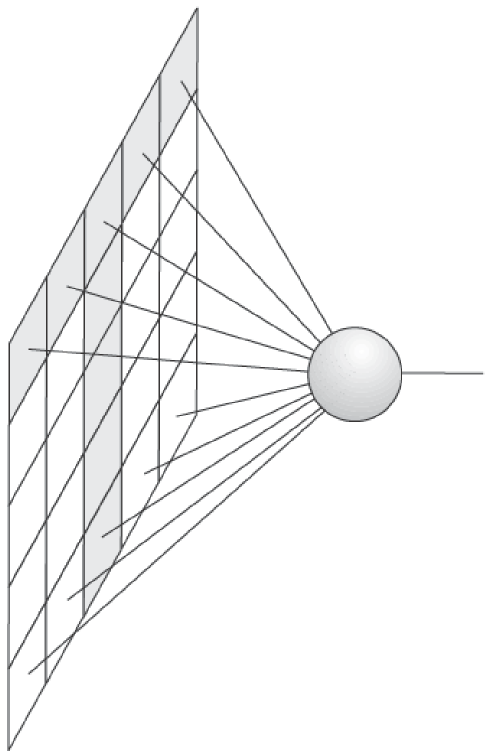
- Many methods exist to improve backpropagation
- E.g., backpropagation with momentum

$$\Delta w_{ij}^{t} = -\lambda \frac{\partial R}{\partial w_{ij}} + \alpha \Delta w_{ij}^{t-1}$$

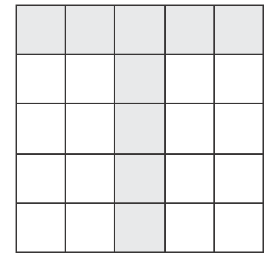


Neurons as Feature Extractor

- Compute the similarity of a pattern to the ideal pattern of a neuron
- Threshold is the minimal similarity required for a pattern
- Reversely, it visualizes the connections of a neuron



pattern



weights

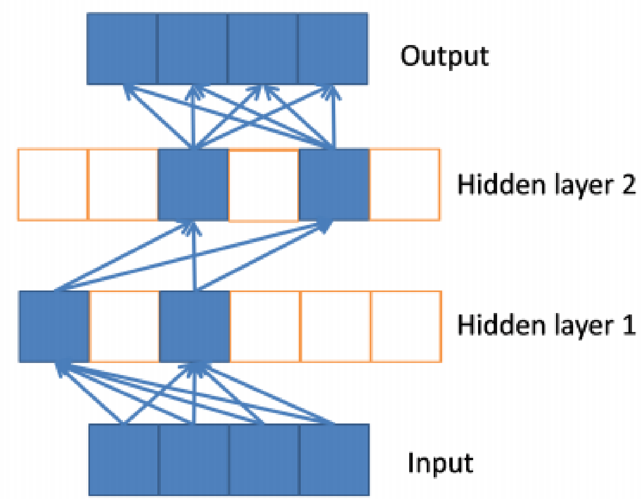
1	1	1	1	1
–1	–1	1	–1	–1
–1	–1	1	–1	–1
–1	–1	1	–1	–1
–1	–1	1	–1	–1

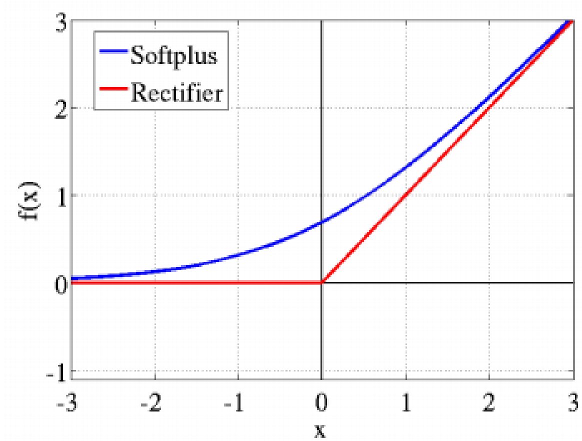
Vanishing gradient problem

- The gradient can decrease exponentially during back-prop
- Solutions:
  - Pre-training + fine tuning
  - Rectifier neurons (sparse gradients)
- Ref:
  - Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. Hochreiter, Bengio, & Frasconi, 2001

Deep Rectifier Nets

- Sparse representations without gradient vanishing



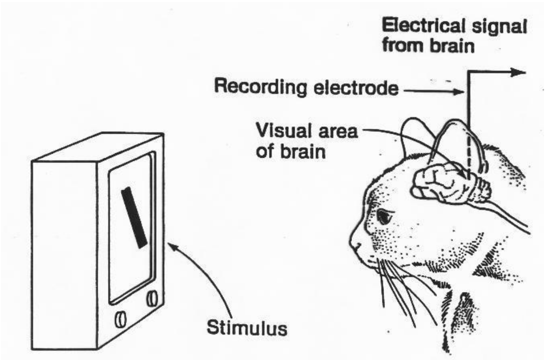


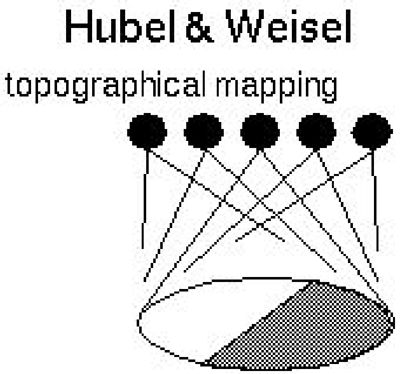
- Non-linearity comes from the path selection
  - Only a subset of neurons are active for a given input
- Can been seen as a model with an exponential number of linear models that share weights

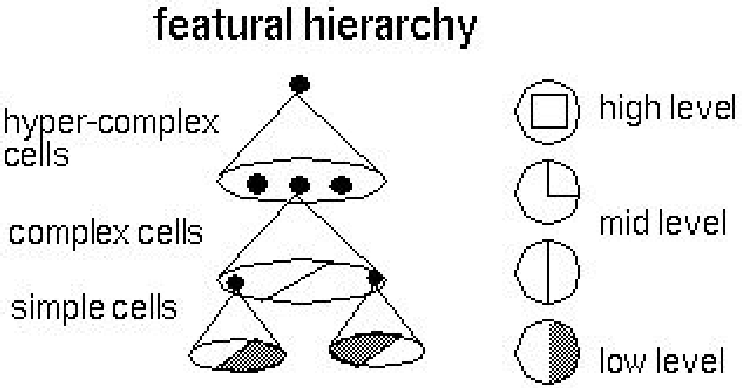
[Deep sparse rectifier neural networks. Glorot, Bordes, & Bengio, 2011]

CNN

- Hubel and Wiesel’s study on annimal’s visual cortex:
  - Cells that are sensitive to small sub-regions of the visual field, called a *receptive field*
  - Simple cells respond maximally to specific edge-like patterns within their receptive field. Complex cells have larger receptive fields and are locally invariant to the exact position of the pattern.

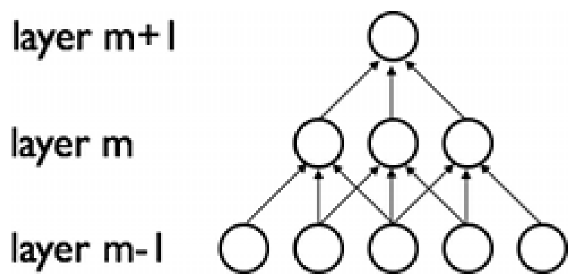




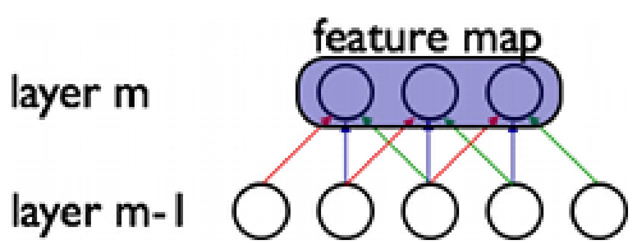


Convolutional Neural Networks

- Sparse local connections (spatially contiguous receptive fields)

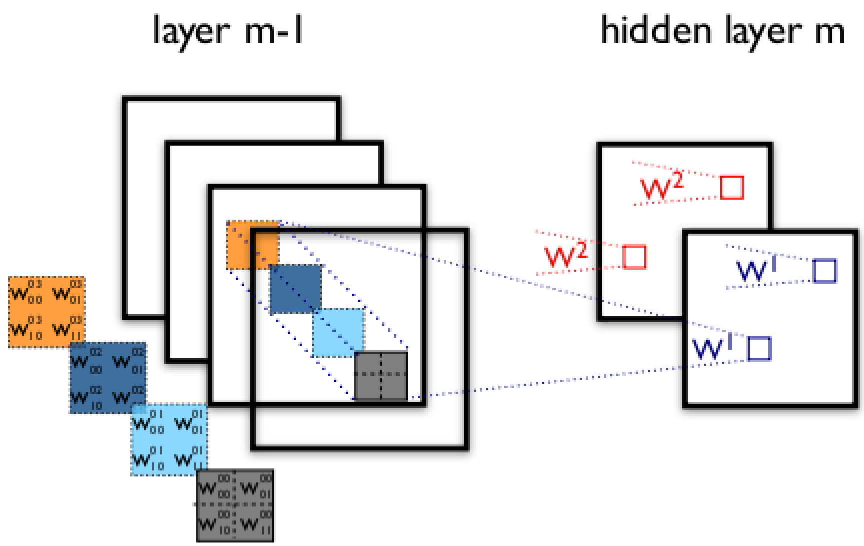


- Shared weights: each filter is replicated across the entire visual field, forming a feature map



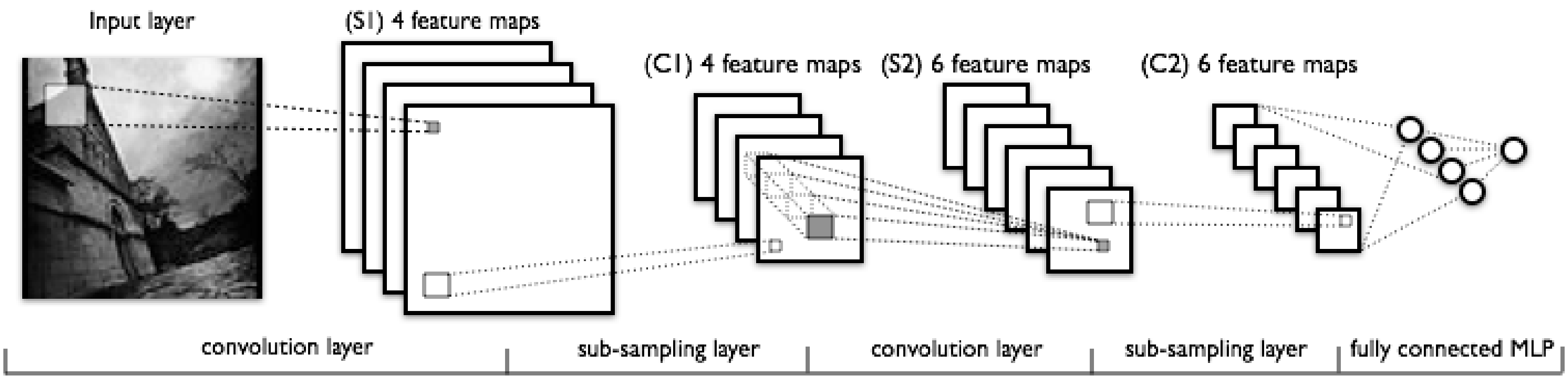
CNN

- Each layer has multiple feature maps



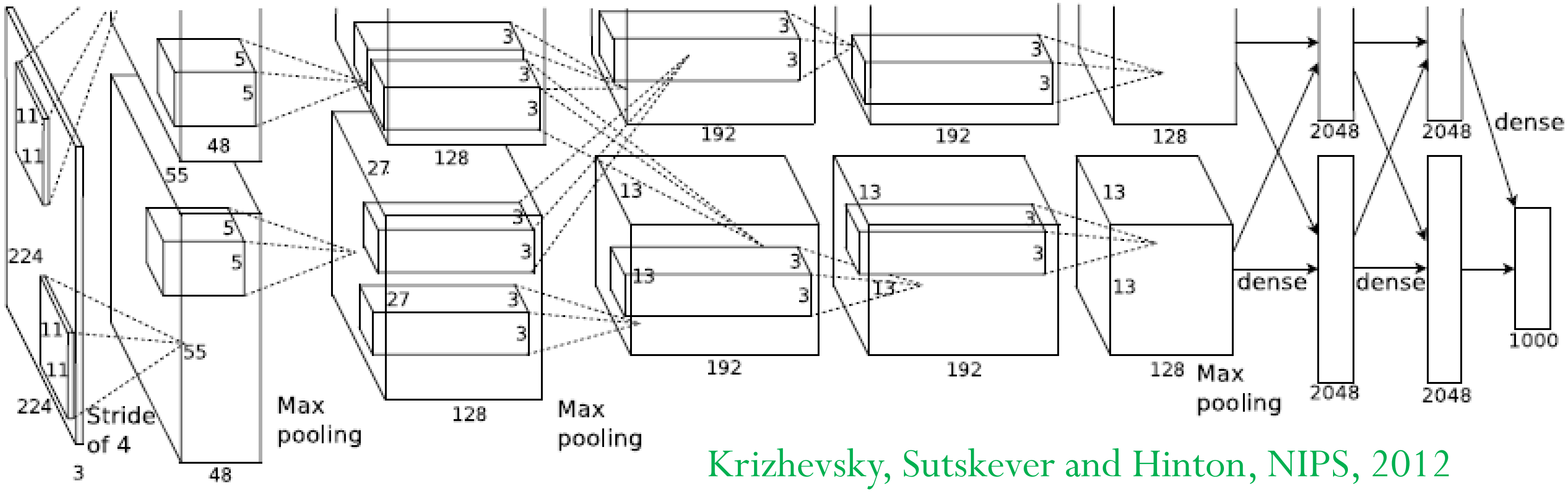
CNN

- The full model



- ***Max-pooling***, a form of non-linear down-sampling.
  - Max-pooling partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum value.

Example: CNN for image classification



Krizhevsky, Sutskever and Hinton, NIPS, 2012

- Network dimension: 150,528(input)-253,440–186,624–64,896–64,896–43,264–4096–4096–1000(output)
  - In total: 60 million parameters
  - Task: classify 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes
  - Results: state-of-the-art accuracy on ImageNet

IMAGENET

14,197,122 images, 21841 synsets indexed

Explore Download Challenge New People Publication About

Issues with CNN

- Computing the activations of a single convolutional filter is much more expensive than with traditional MLPs

- Many tuning parameters
  - # of filters:
    - Model complexity issue (overfitting vs underfitting)
  - Filter shape:
    - the right level of "granularity" in order to create abstractions at the proper scale, given a particular dataset
    - Usually 5x5 for MNIST at 1^{st} layer
  - Max-pooling shape:
    - typical: 2x2; maybe 4x4 for large images

Auto-Encoder

- Encoder: (a distributed code)

$$\mathbf{y} = s(\mathbf{W}\mathbf{x} + \mathbf{b})$$

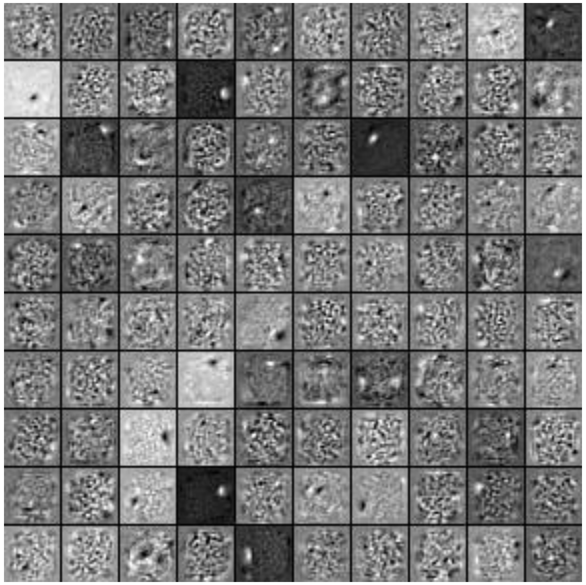
- Decoder:

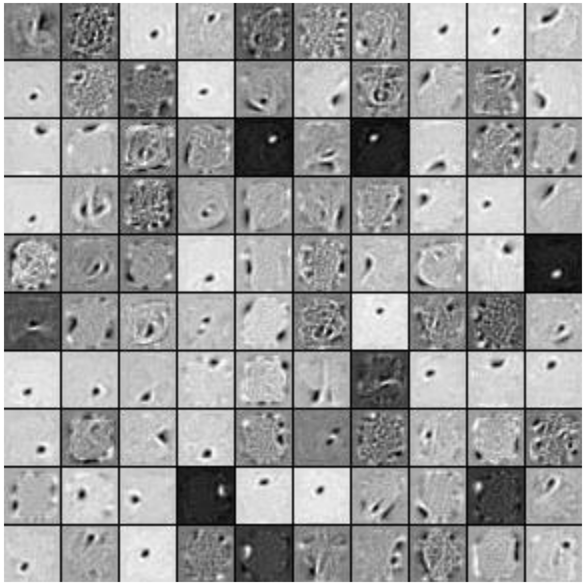
$$\mathbf{z} = s(\mathbf{W}'\mathbf{y} + \mathbf{b}')$$

- Minimize reconstruction error

- Connection to PCA
  - PCA is linear projection, which Auto-Encoder is nonlinear
  - Stacking PCA with nonlinear processing may perform as well (MaYi's work)
- Denoising Auto-Encoder
  - A stochastic version with corrupted noise to discover more robust features
  - E.g., randomly set some inputs to zero

- Left: no noise;   right: 30 percent noise





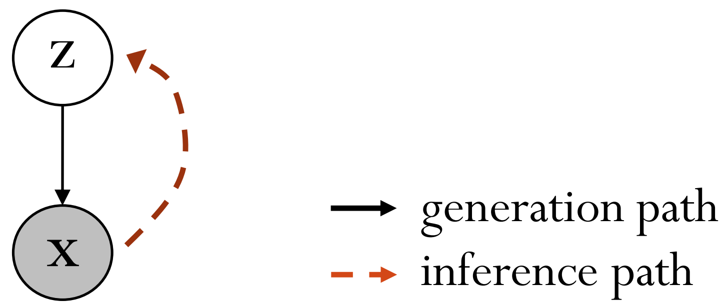
Deep Generative Models

Probabilistic Generative Models

- **Assumption**: data is described by some factors, which are often hidden

$$z \sim p(z|\alpha)$$

$$x \sim p(x|z,\beta)$$



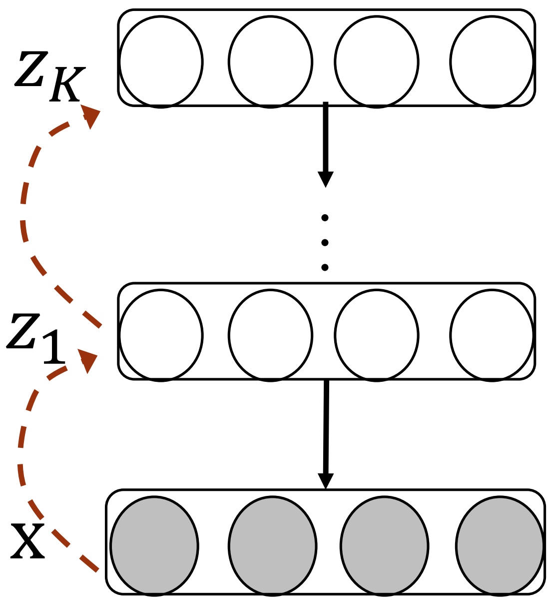
- **Inference with top-down & bottom-up cues**: infer the posterior distribution $p(z|x) \propto p(z|\alpha)p(x|z,\beta)$
- **Learning**: estimate the parameters

$$\hat{\theta} = \operatorname{argmax} p(D|\theta)$$

- **Bayesian inference**: infer the posterior distribution of parameters $p(\theta|D) \propto p_0(\theta)p(D|\theta)$

Deep Generative Models

- Multi-layer ***latent-feature*** representations with nonlinear transformations



high-level representations

a horse sitting on the grass

⋮

slightly higher-level representations

low-level representations

$$z \sim p(z|\alpha) \qquad x \sim p(x|z, \beta)$$

- Many variants by combining different building blocks

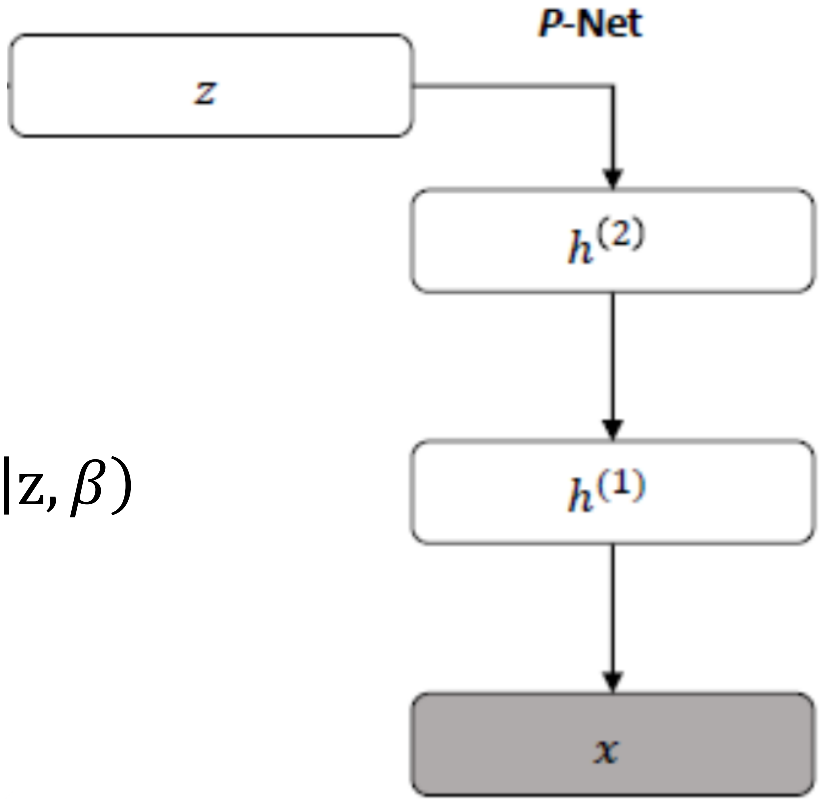
Recent Advances on DGMs

- Models:
  - Deep belief networks (Salakhutdinov & Hinton, 2009)
  - Autoregressive models (Larochelle & Murray, 2011; Gregor et al., 2014)
  - Stochastic variations of neural networks (Bengio et al., 2014)
  - …
- Applications:
  - Image recognition (Ranzato et al., 2011)
  - Inference of hidden object parts (Lee et al., 2009)
  - Semi-supervised learning (Kingma et al., 2014)
  - Multimodal learning (Srivastava & Salakhutdinov, 2014; Karpathy et al., 2014)
  - …
- Learning algorithms
  - Stochastic variational inference (Kingma & Welling, 2014; Rezende et al., 2014)
  - …

Symmetric Q-P Network

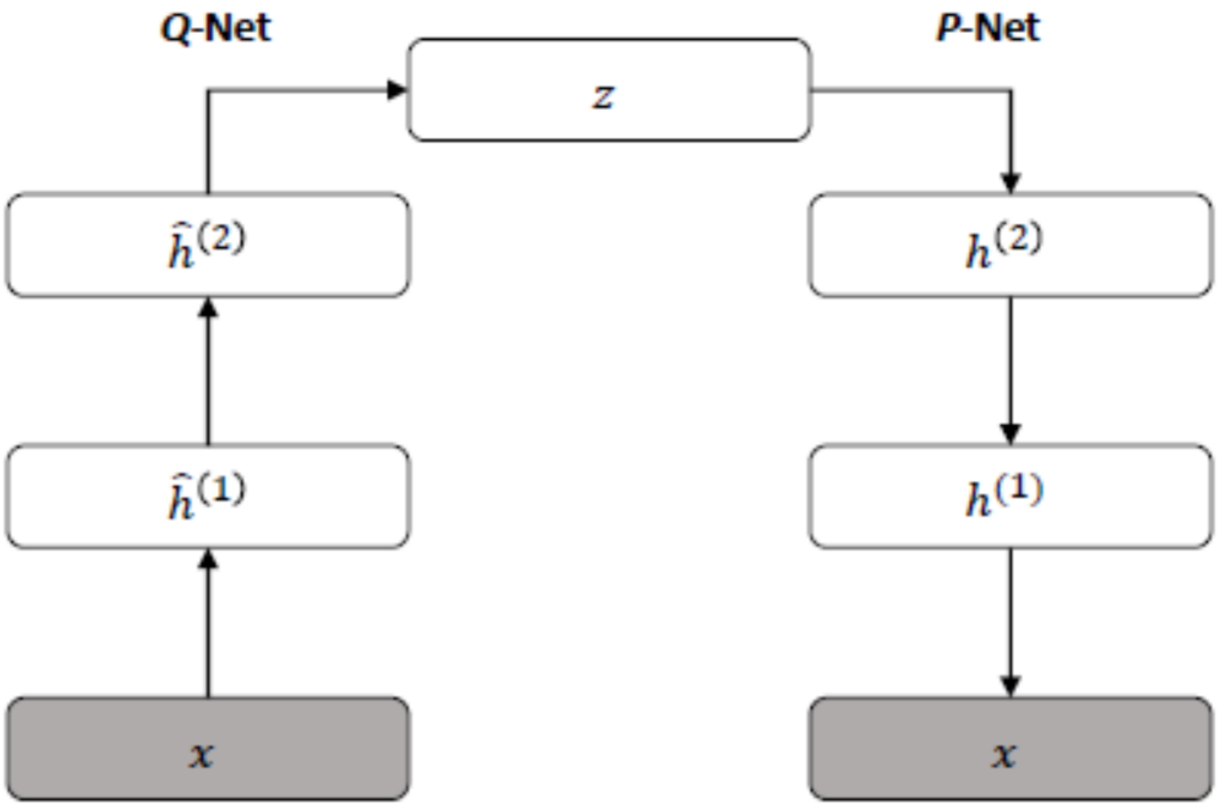
- **P-network** with two deterministic layers

$$p(\mathrm{z}|\mathrm{x}) \propto p(\mathrm{z}|\alpha)p(\mathrm{x}|\mathrm{z},\beta)$$



Symmetric Q-P Network

- **Q-network** approximates the posterior (Kingma & Welling, 2014; Rezende et al. 2014)



$$q(z|x) \approx p(z|x)$$

Neural Evidence?

- Our visual systems contain multilayer generative models

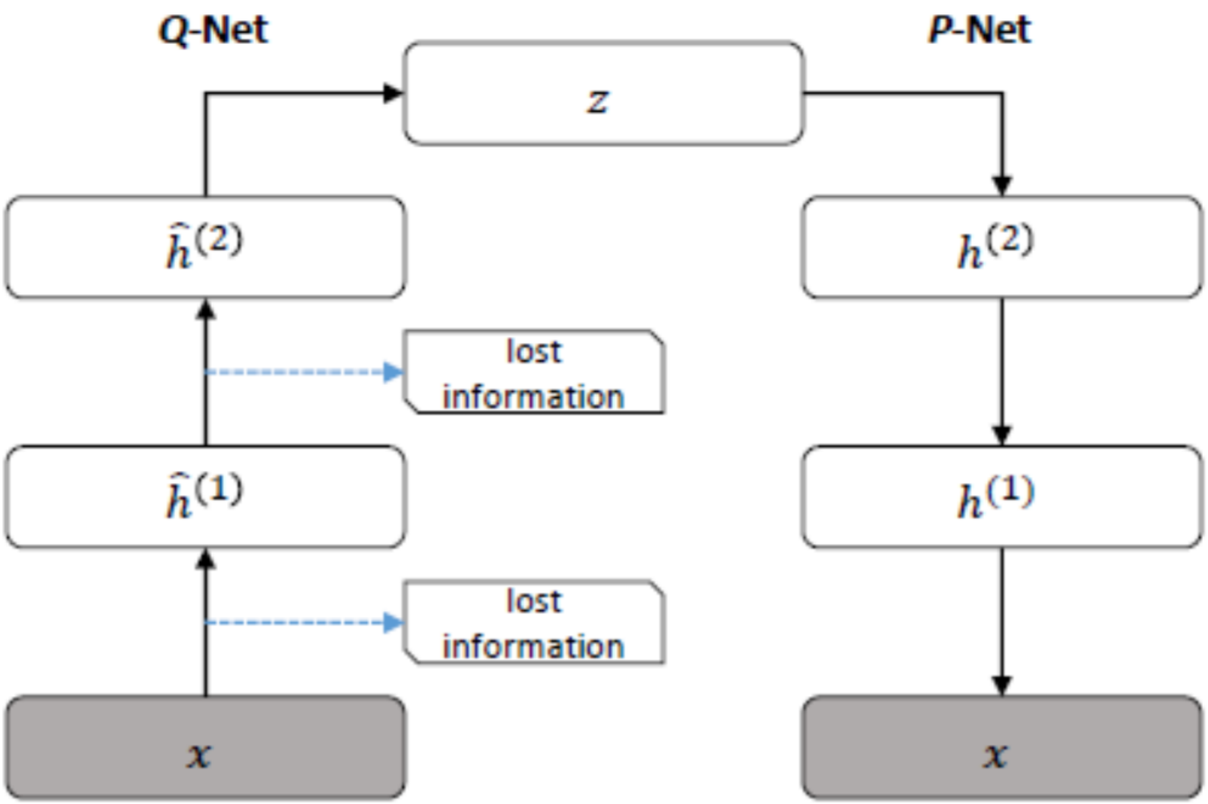
- Top-down connections:
  - Generate low-level features of images from high-level representations
  - Visual imagery, dreaming?

- Bottom-up connections:
  - Infer the high-level representations that would have generated an observed set of low-level features

[Hinton, Trends in Cognitive Science, 11(10), 2007]

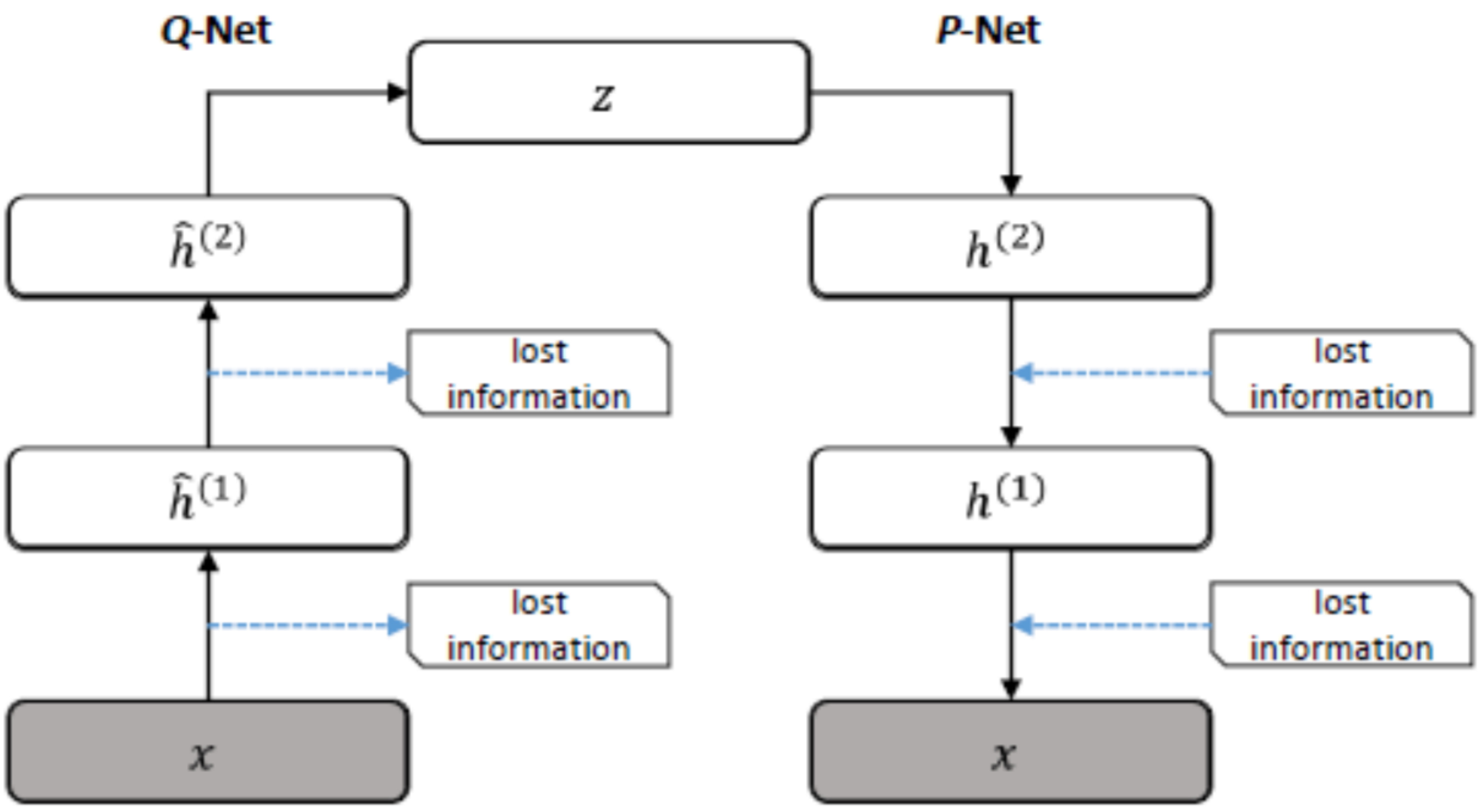
Symmetric Q-P Network

- **Problem**: detail information is lost during abstraction

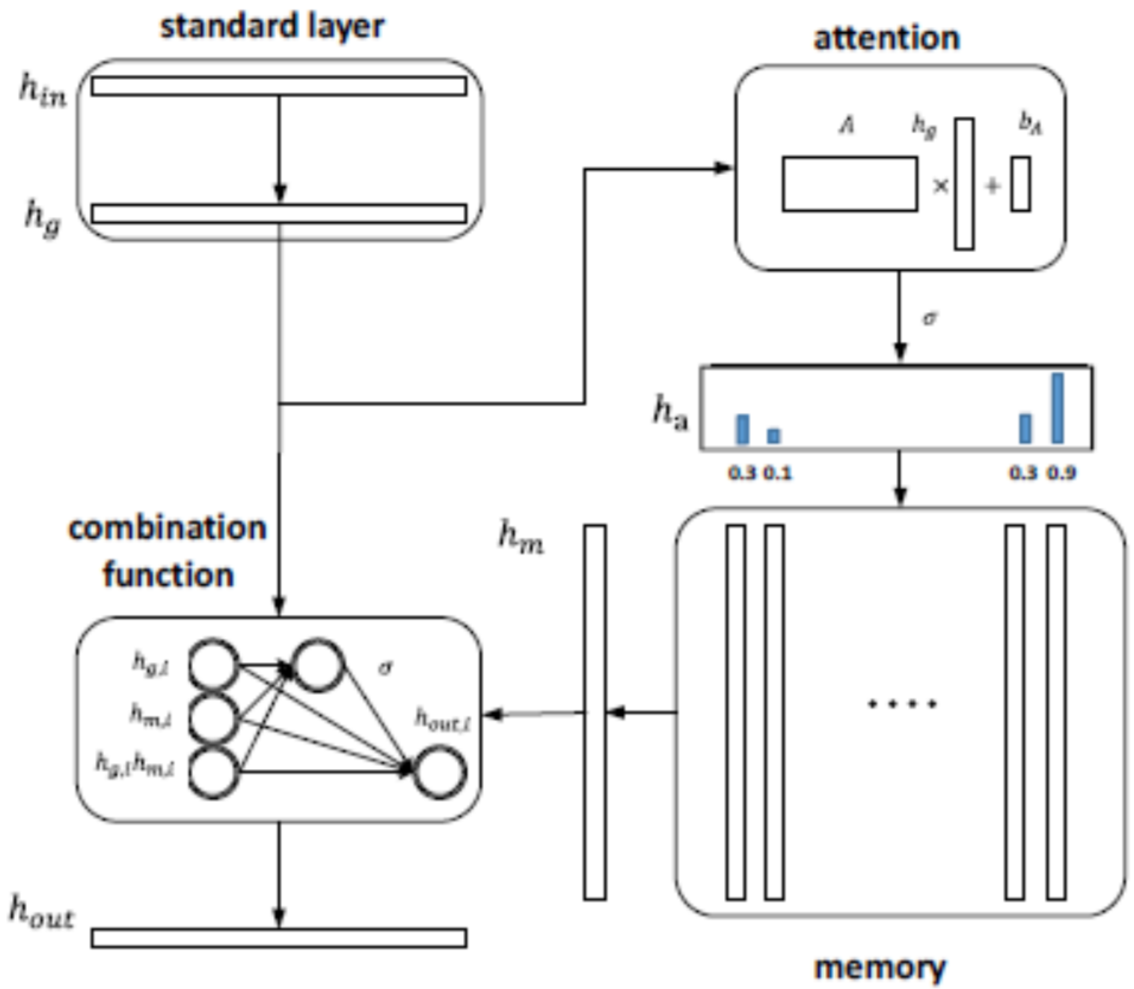


Symmetric Q-P Network

- **Ideal case**: get the lost information back!



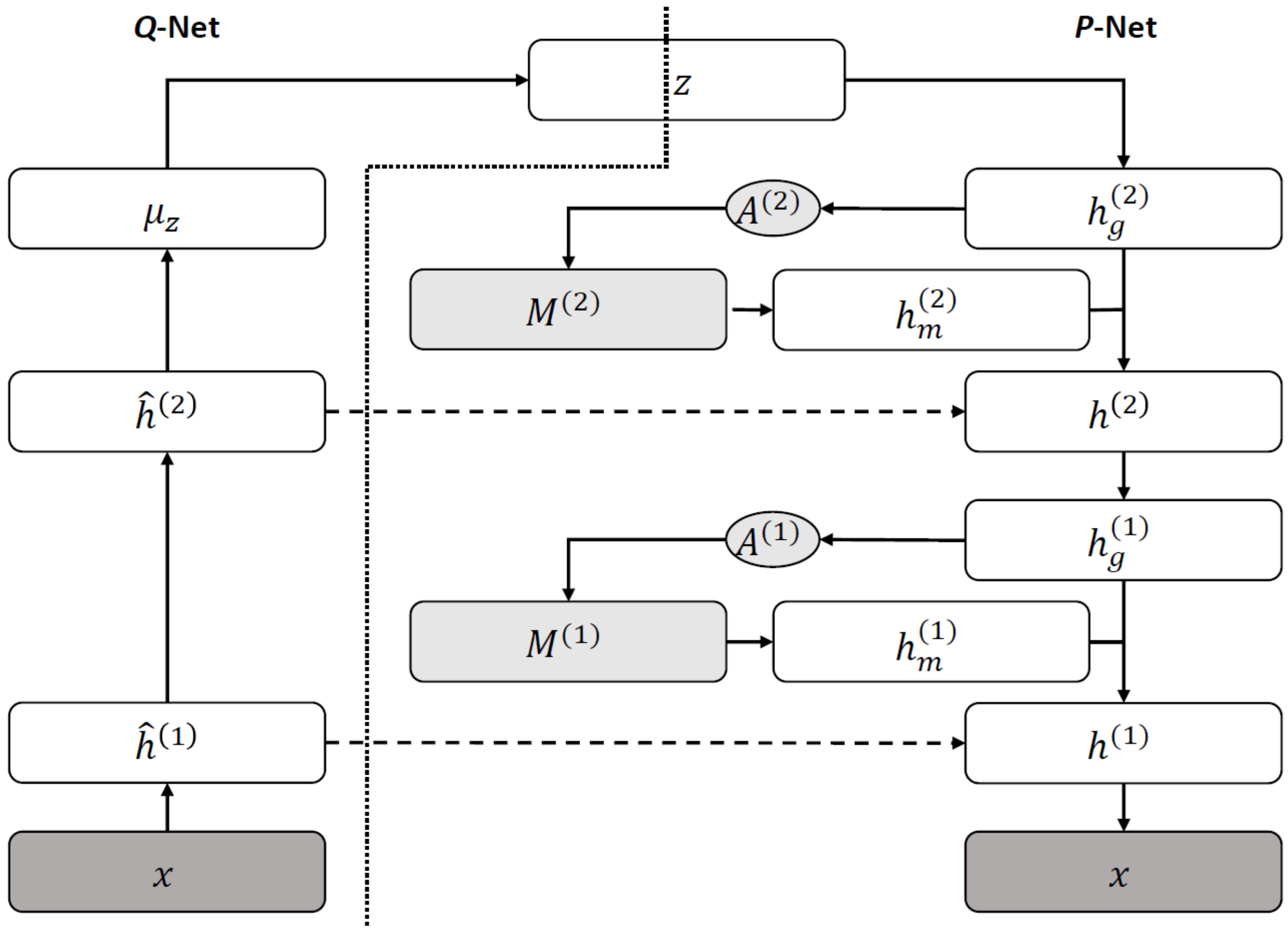
A Layer with Memory and Attention



[Li, Zhu & Zhang. Learning to Generate with Memory, ICML 2016]

A Stacked Deep Model with Memory

- Asymmetric architecture



Some Results

- Density estimation

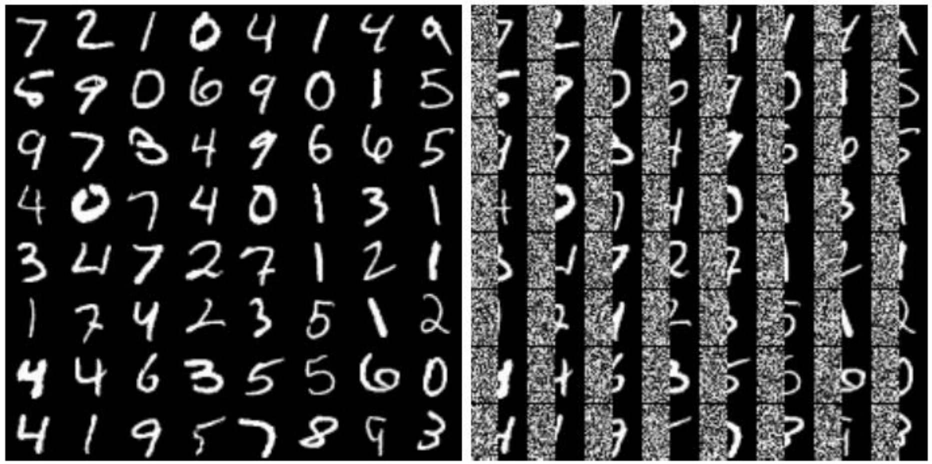
MODELS	MNIST	OCR-LETTERS
VAE	-85.69	-30.09
MEM-VAE(ours)	**-84.41**	**-29.09**
IWAE-5	-84.43	-28.69
MEM-IWAE-5(ours)	**-83.26**	**-27.65**
IWAE-50	-83.58	-27.60
MEM-IWAE-50(ours)	**-82.84**	**-26.90**

Some Results

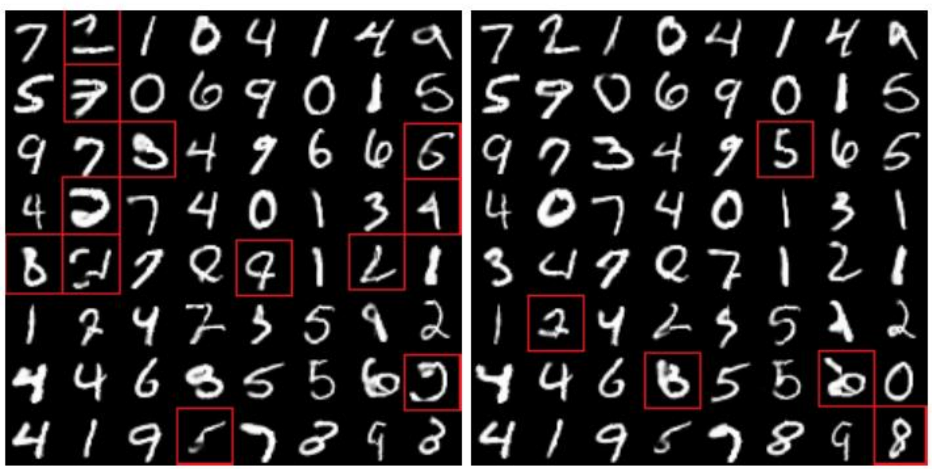
- Density estimation

Models	MNIST	OCR-letters
VAE	-85.69	-30.09
MEM-VAE(ours)	-84.41	-29.09
IWAE-5	-84.43	-28.69
MEM-IWAE-5(ours)	-83.26	-27.65
IWAE-50	-83.58	-27.60
MEM-IWAE-50(ours)	**-82.84**	**-26.90**
DBN	-84.55	-
S2-IWAE-50	**-82.90**	-
*RWS-SBN/SBN**	-85.48	-29.99
*RWS-NADE/NADE**	-85.23	**-26.43**
*NADE**	-88.86	-27.22
*DARN**	**-84.13**	-28.17

Missing Value Imputation



(a) Data (b) Noisy data



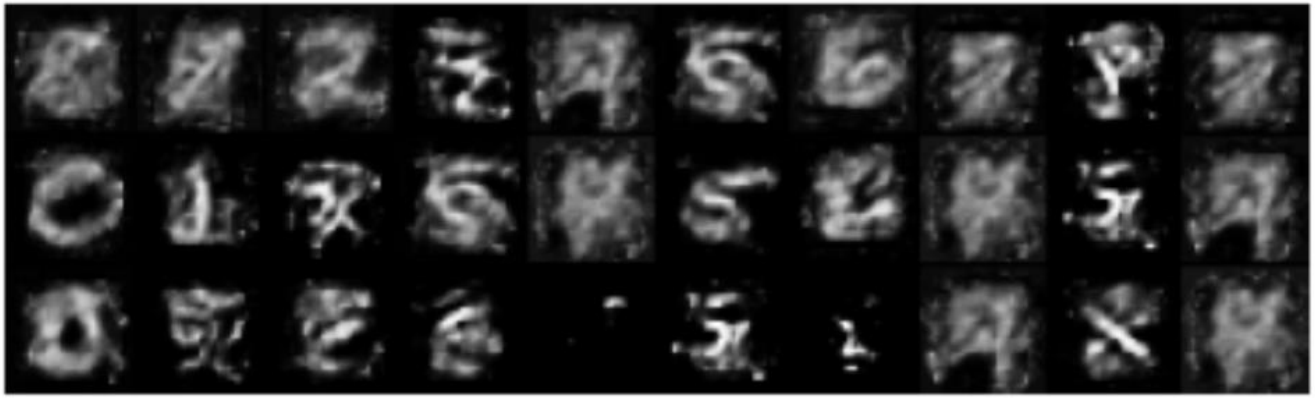
(c) Results of VAE (d) Results of MEM-VAE

Learnt Memory Slots

- Average preference over classes of the first 3 slots:

"0"	"1"	"2"	"3"	"4"	"5"	"6"	"7"	"8"	"9"
0.27	0.82	0.33	0.11	0.34	0.15	0.49	0.27	0.09	0.28
0.24	0.09	0.06	0.11	0.30	0.13	0.12	0.27	0.09	0.21
0.18	0.05	0.06	0.11	0.07	0.07	0.05	0.11	0.09	0.18

- Corresponding images:



Discussions

Some Counter-intuitive Properties of DL

- Stability w.r.t small perturbations to inputs
  - Imperceptible non-random perturbation can arbitrarily change the prediction (adversarial examples exist!)

(a) (b)

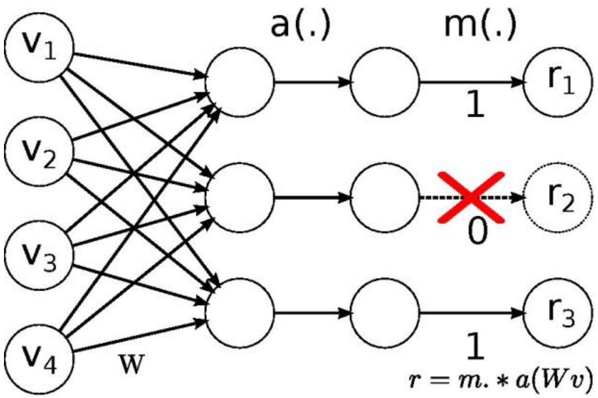
10x of differences

[Szegedy et al., Intriguing properties of neural nets, 2013]

Overfitting in Big Data

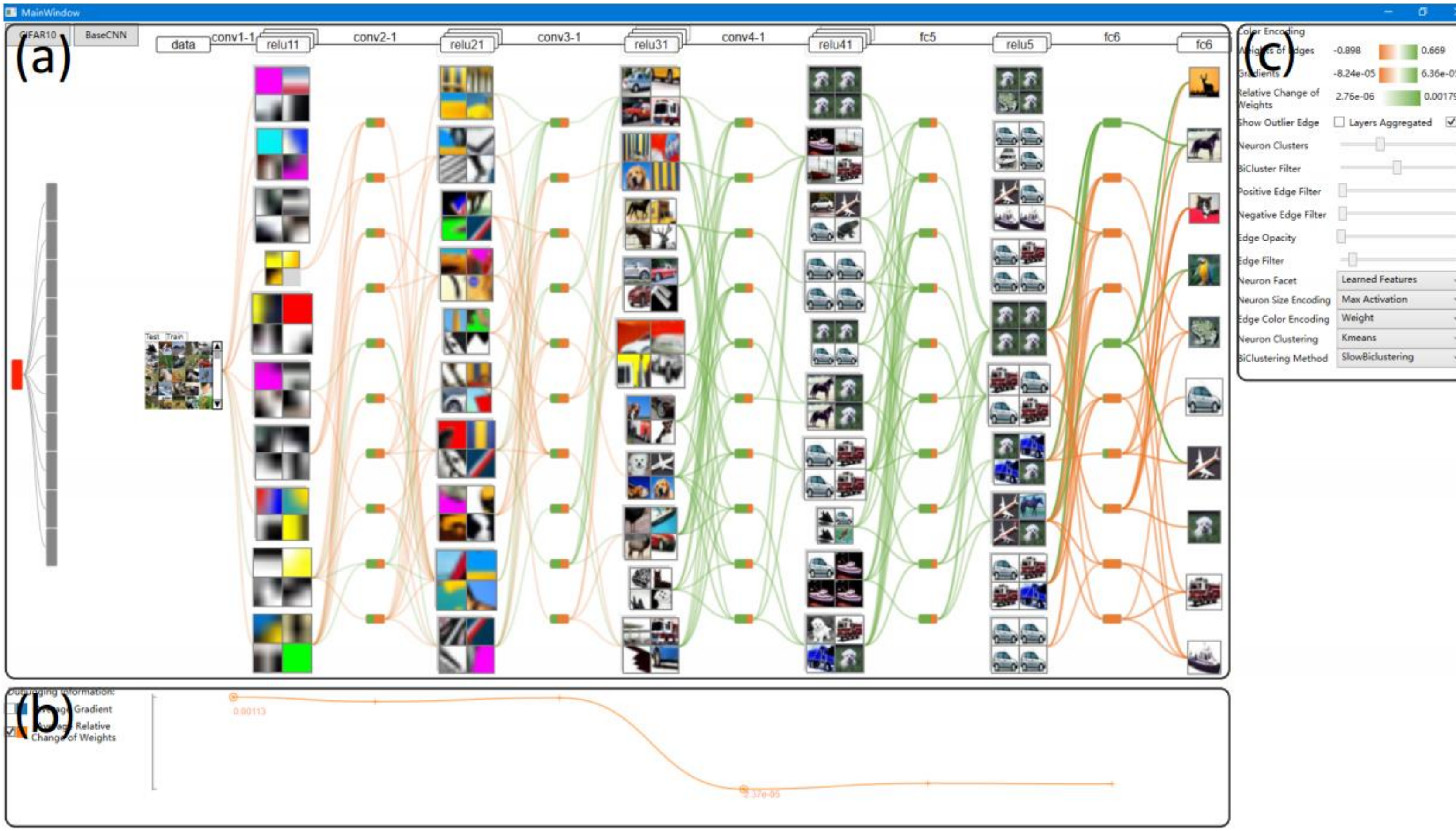
- Surprisingly, regularization to prevent overfitting is ***increasingly important***, rather than increasingly irrelevant!

- Increasing research attention, e.g., dropout training (Hinton, 2012)



- More theoretical understanding and extensions
  - Dropout as a Bayesian approximation (Gal & Ghahramani, 2016)
  - MCF (van der Maaten et al., 2013); Logistic-loss (Wager et al., 2013);
  - Dropout SVM (Chen, et al., 2014; Zhuo et al., 2015)

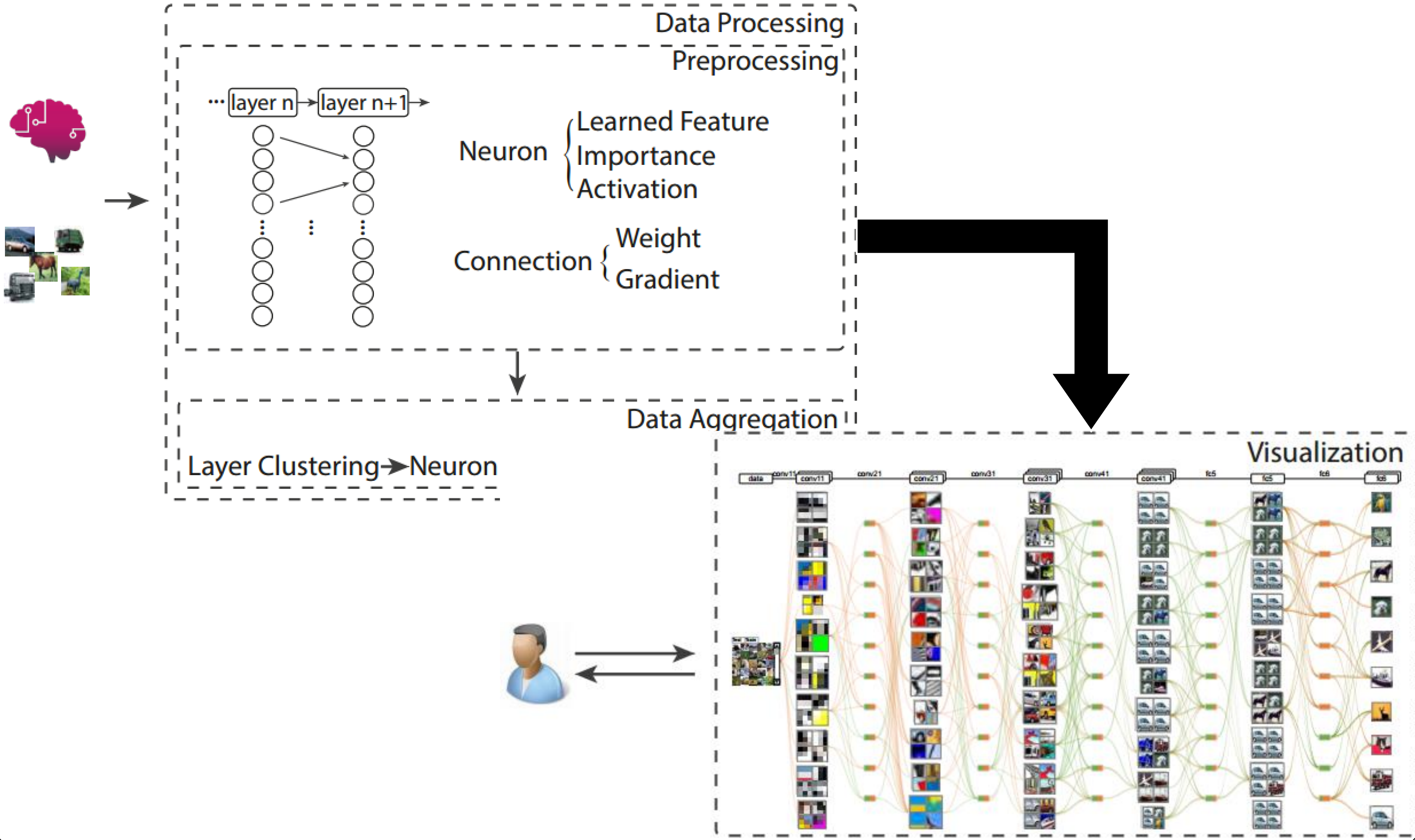
CNNVis: Turn black-box into gray



http://cgcad.thss.tsinghua.edu.cn/mengchen/video/CNNVis-final.mp4

[Liu, Shi, Li, Li, Zhu & Liu. Towards Better Analysis of Deep CNNs, IEEE VIS 2016]

CNNVis: Turn black-box into gray



Thank You!