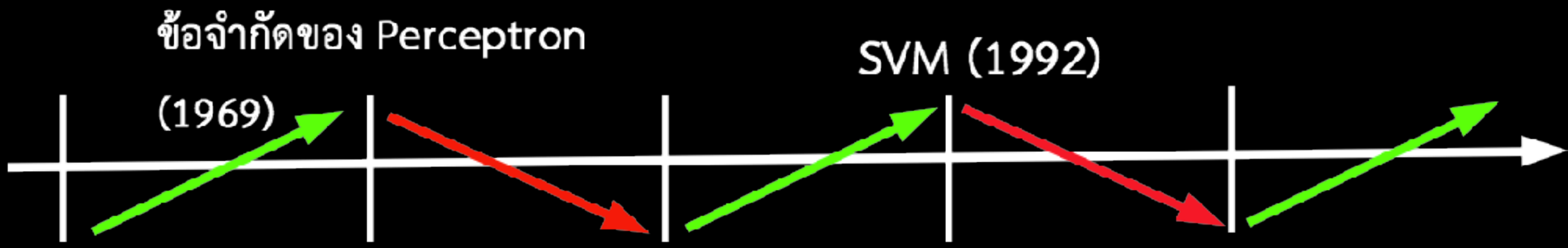
Deep Learning

sanparith.marukatat@nectec.or.th

Neural Networks Timeline

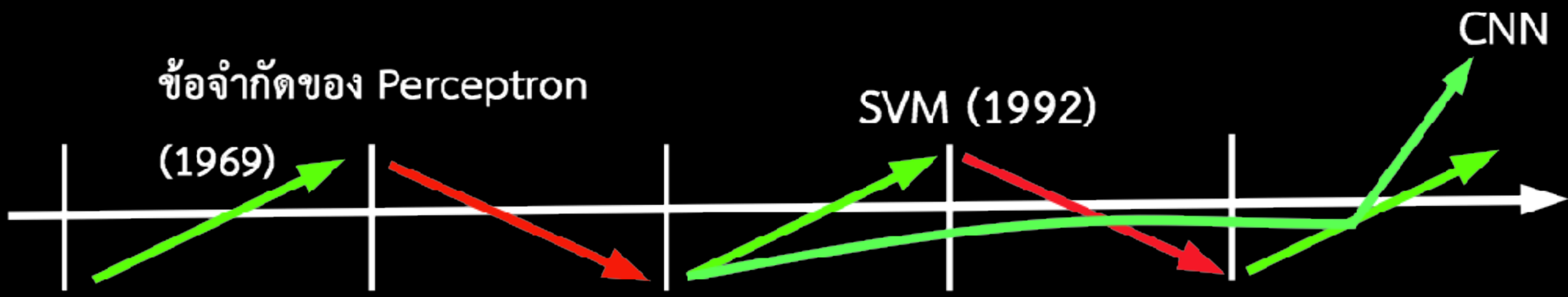


แบบจำลองเซลล์ประสาท
McCulloch และ Pitts
(1943)

Backprop algorithm
มีการค้นพบตั้งแต่ทศวรรษที่ 70
แต่มาถูกพบใหม่และมีการใช้งาน
อย่างแพร่หลายในช่วง 80

Deep learning
(2006)

Neural Networks Timeline



แบบจำลองเซลล์ประสาท
McCulloch และ Pitts
(1943)

Backprop algorithm
มีการค้นพบตั้งแต่ทศวรรษที่ 70
แต่มาถูกพบใหม่และมีการใช้งาน
อย่างแพร่หลายในช่วง 80

Deep learning
(2006)

- Return of Neural Networks
- Focus on deep structure
- Take advantage of today computing power

Neural Networks Timeline

แบบจำลองเซลล์ประสาท
McCulloch และ Pitts
(1943)

Backprop algorithm
มีการค้นพบตั้งแต่ทศวรรษที่ 70
แต่มาถูกพบใหม่และมีการใช้งาน
อย่างแพร่หลายในช่วง 80

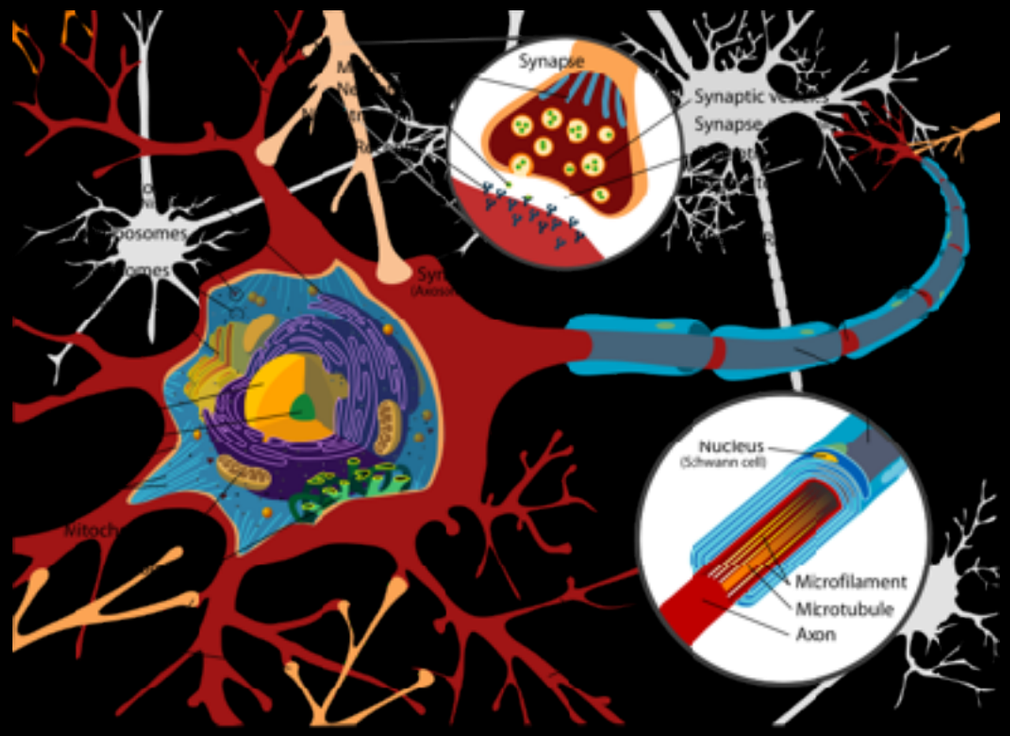
Deep learning
(2006)

- AI
  - Philosophy, Symbolic, Statistics, Connectionist, ...
- Machine Learning
  - Focus on learning algorithm
  - Formal framework
  - Learning problems: Supervised, Unsupervised, Semi-supervised, Transductive, Active
- Datamining
  - Discover useful rules or structures

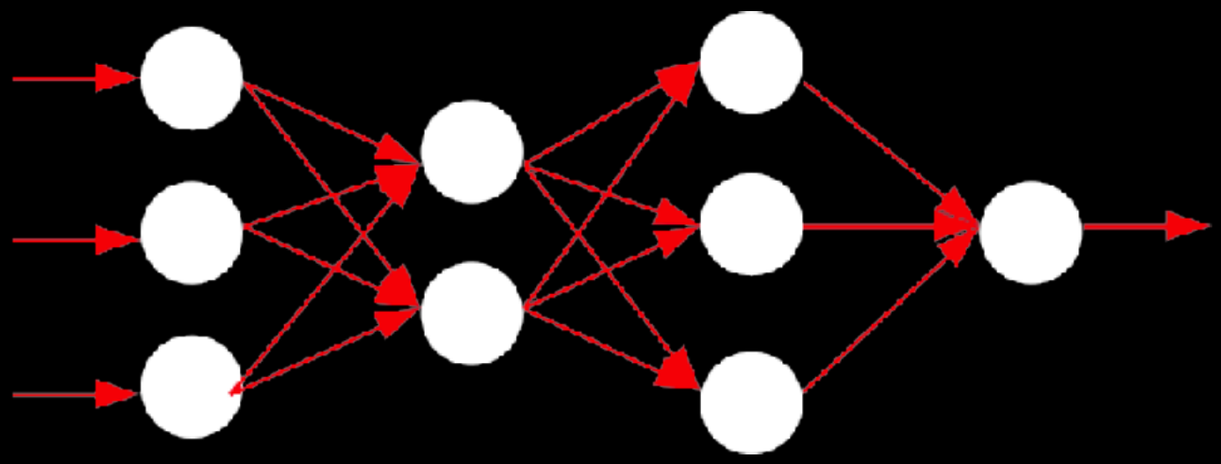
Basic concept

Neural Networks

- Neurons are connected via synapse
- A neuron receives activations from other neurons
- When these activations reach a threshold, it fires an electronics signal to other neurons



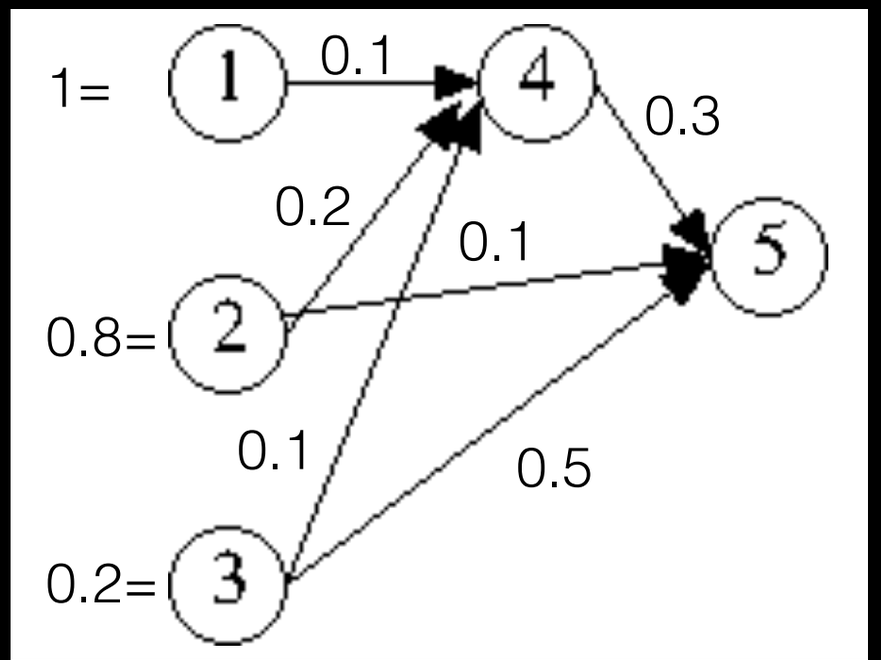
http://en.wikipedia.org/wiki/Neuron



Artificial Neural Networks (1)

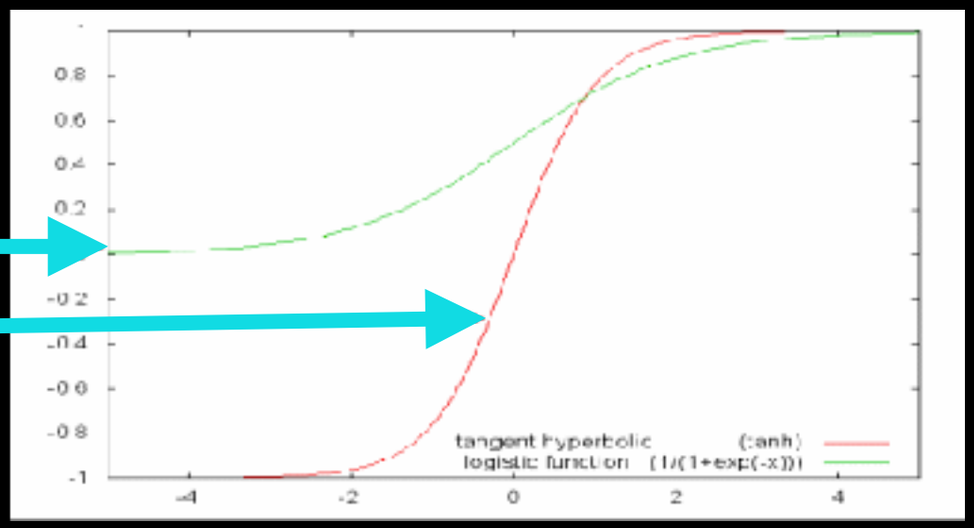
$$o_i^l = \sum_{j=1}^{h_{l-1}} w_{ij}^l s_j^{l-1}$$

$$s_i^l = f(o_i^l)$$



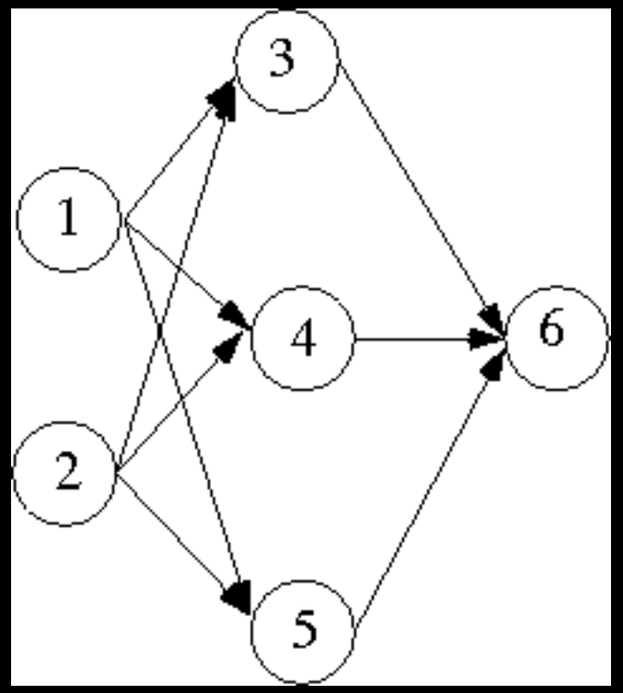
Activation function

- logistic(x) = 1/(1+exp(-x))
- tanh
- ReLU(x) = max{0, x}



Multi-Layer Perceptron

- Number of input nodes = number of features
- 1 hidden layer
- Full connection between consecutive layers
- 2-class
  - 1 output node with class label +1 and -1
- more than 2 classes
  - Number of output nodes = number of classes **(WHY?)**
  - Neural network produces a vector of output values
  - Each output node is associated with a single class
  - Classification rule: put the input pattern in the class whose corresponding output node gives maximal value

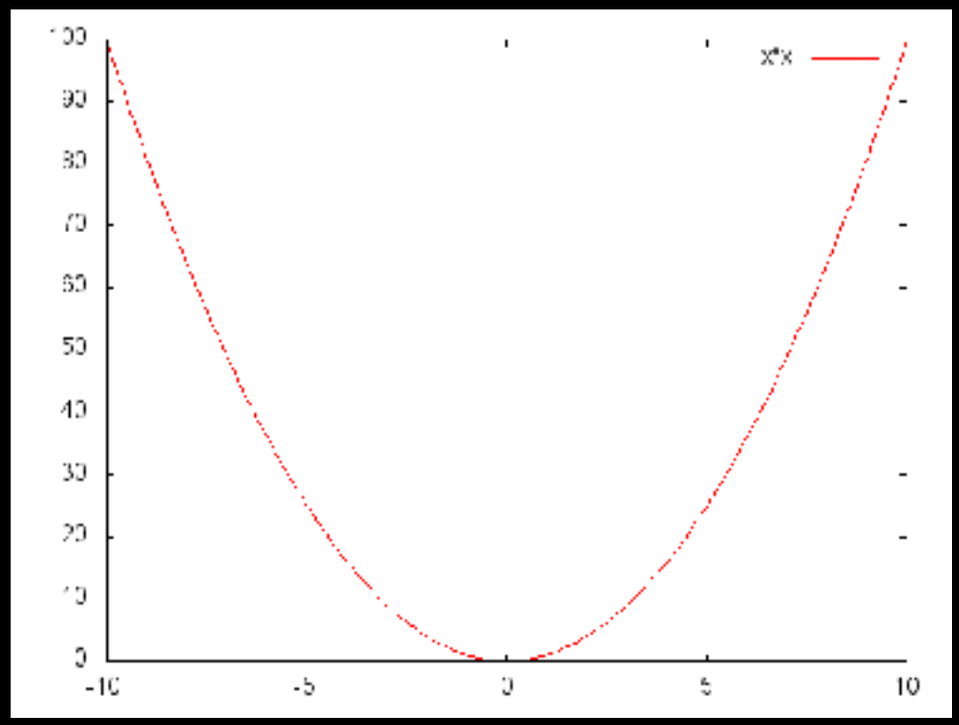


Artificial Neural Networks (2)

- Universal Approximator
  - Classification, Regression, Density estimation, …
- Parameters
  - Inputs
  - Weights
- Training = Adjusting weights
  - How? -> Use **Gradient**

Gradient (1)

- Basic idea
  - f(x) = x*x
  - How to find x that minimizes f?
  - Given value of x (e.g. =5) how to find new value of x with smaller value of f?



Gradient (2)

- **Gradient of a function f** having a set of parameters θ is **a vector of partial derivatives** of f with respect to each parameter θ_i

- **Gradient** indicates the **direction of change** for θ which greatest **increases** f(θ)

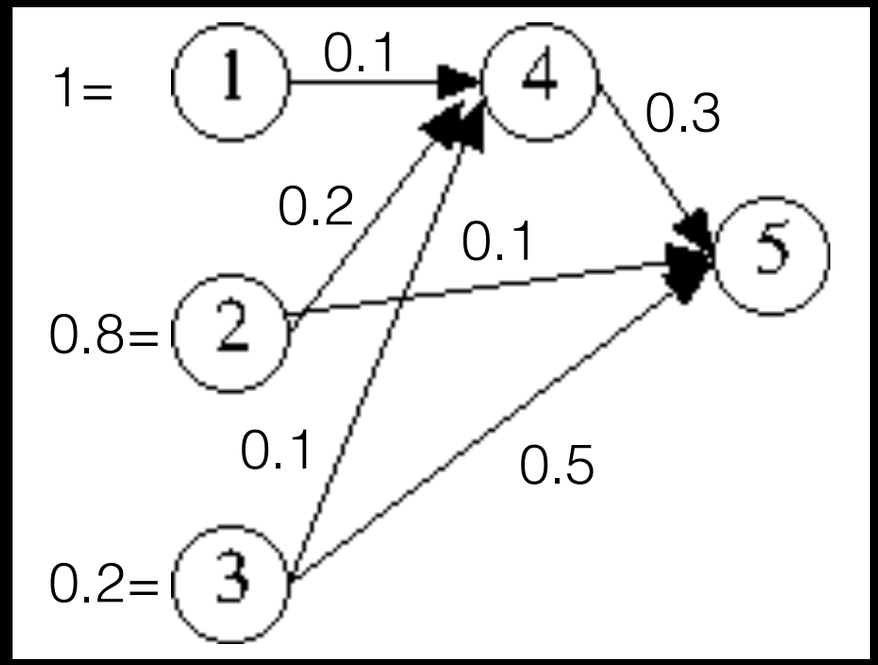
- Question: How can we use the Gradient to train the neural networks?

Supervised training

- NN Input: vector x
- NN output: vector s^L
- Target output: vector s^*
- Classical objective function
  - Squared error $E = (s-s^*)^2$

Exercise

- What is the function implemented by this network?
  - Activation function = tanh
- How many weights to be adjusted?
- What is the dimension of the Gradient?



Error Back-propagation (Backprop)

- Squared error

$$E = \sum_{i=1}^{h_L} (s_i^L - s_i^*)^2$$

- Gradient points to direction of increased E -> **So what?**
- Use **chain rule**
  - h(x) = f(g(x))
  - h'(x) = ?

$$\frac{\partial E}{\partial w_{ij}^l} = \frac{\partial E}{\partial a_i^l} \frac{\partial a_i^l}{\partial w_{ij}^l}$$

Backprop (1)

- If j is on output layer

$$\frac{\partial E}{\partial a_j^L} = 2(s_j^L - s_j^*)\tanh'(a_j^L)$$

- If j is on hidden layer

$$\frac{\partial E}{\partial a_j^l} = \sum_{k=1}^{h_{l+1}} \frac{\partial E}{\partial a_k^{l+1}} \frac{\partial a_k^{l+1}}{\partial s_j^l} \frac{\partial s_j^l}{\partial a_j^l}$$

$$\frac{\partial a_k^l}{\partial s_j^l} = w_{jk}^l$$

$$\frac{\partial s_j^l}{\partial a_j^l} = \tanh'(a_j^l)$$

Backprop (2)

- Calculation backward from output layers
- Change objective function affects output nodes
  - **Cross entropy** for classification problem
- Change activation function affects partial diff s'_j
- Can be applied to any NN structures

Weights update

$$w_{ij} \leftarrow w_{ij} + \eta \Delta w_{ij}$$

learning rate

- Basic update

$$\Delta w_{ij} = -\frac{\partial E}{\partial w_{ij}}$$

- Common update today

momentum weight decay

$$\Delta w_{ij}(t+1) = -\frac{\partial E}{\partial w_{ij}} + \alpha \Delta w_{ij}(t) - \beta w_{ij}$$

Learning rate

- Large learning rate
  - Error reduces fast at the beginning
  - May miss **local optimum**
- Small learning rate
  - Error reduces very slow
- Several adaptive learning rate methods
- Common practice today
  - Fixed learning rate until stable error then reduce (/2 or /10)

Universal approximator

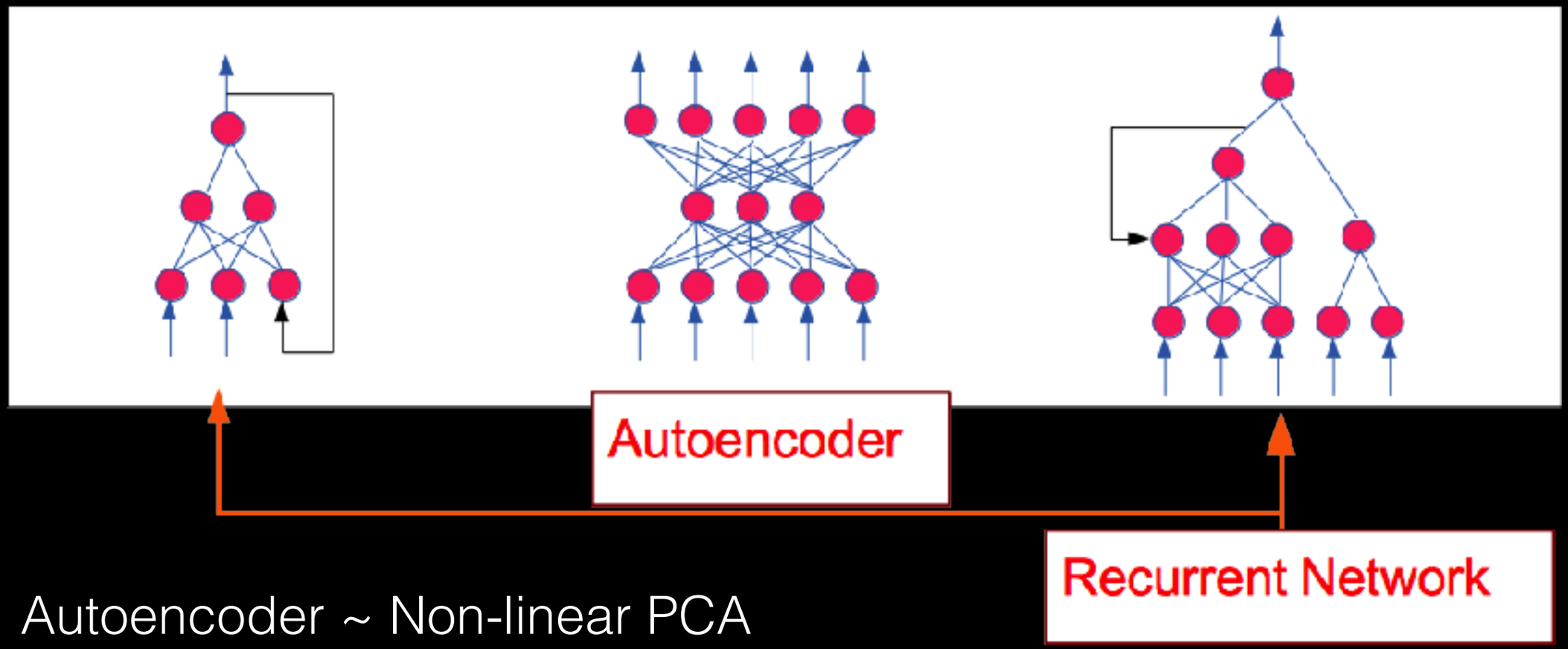
- Cybenko (1989), Hornik et al. (1989) and Funahashi (1989)

Feed-forward neural network with

- any sigmoid transfer function
- one hidden layer with enough hidden nodes

can approximate any continuous real-valued function on $[0,1]^d$ or any compact subset of R^d

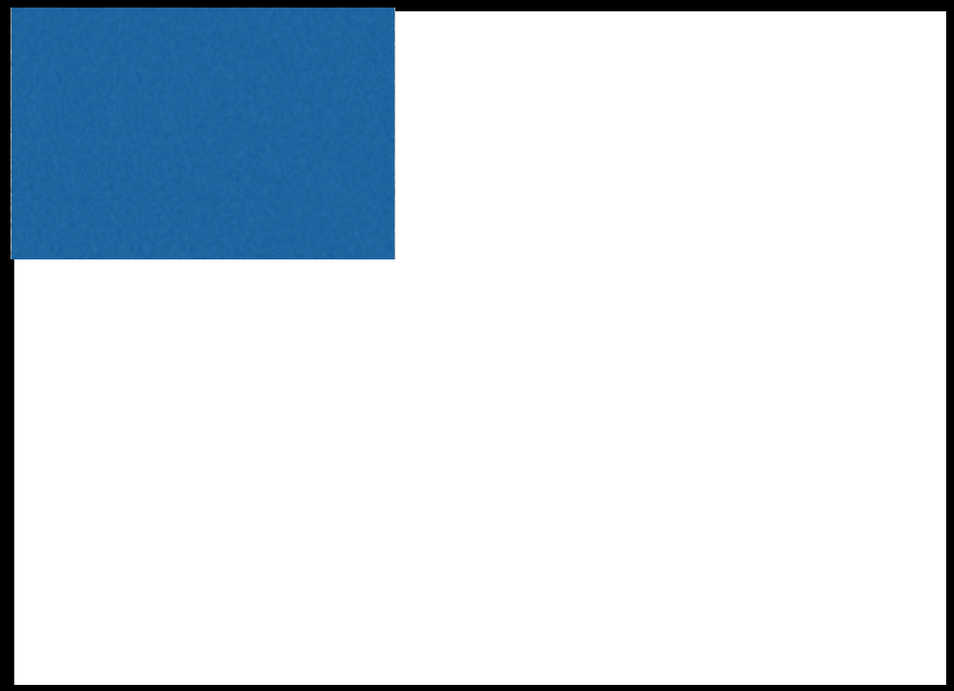
Some NN structures

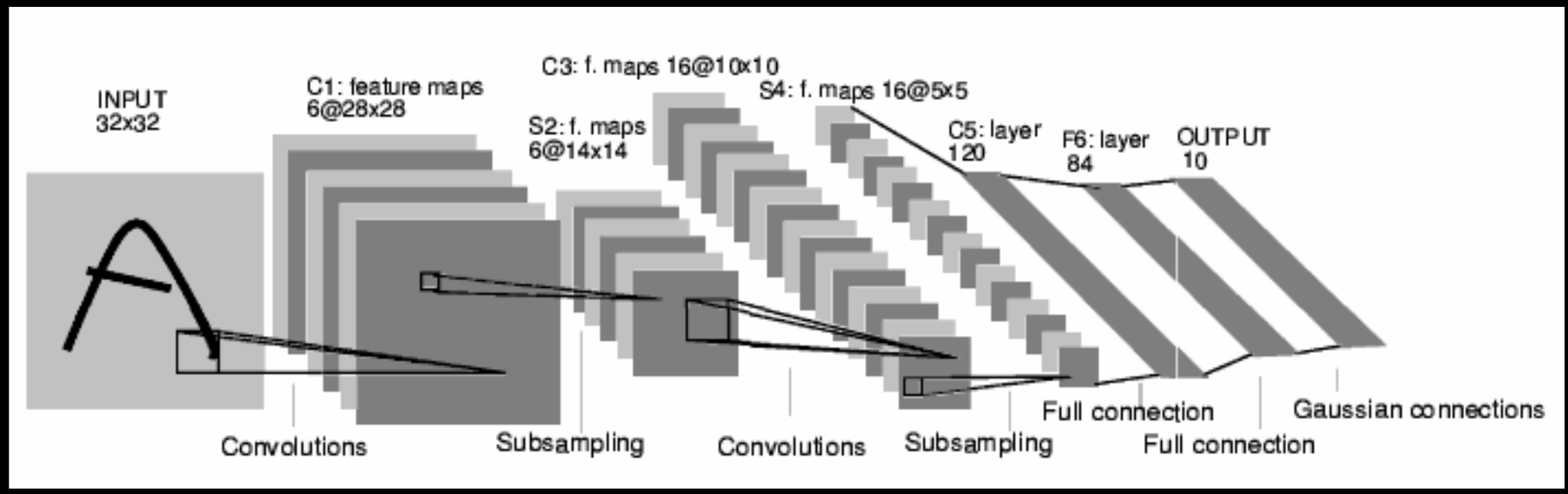


- Autoencoder ~ Non-linear PCA
- Recurrent Network ~ Sequence modeling
  - Long-term dependency problem
  - Outperformed by Markovian model (HMM, State-space models)

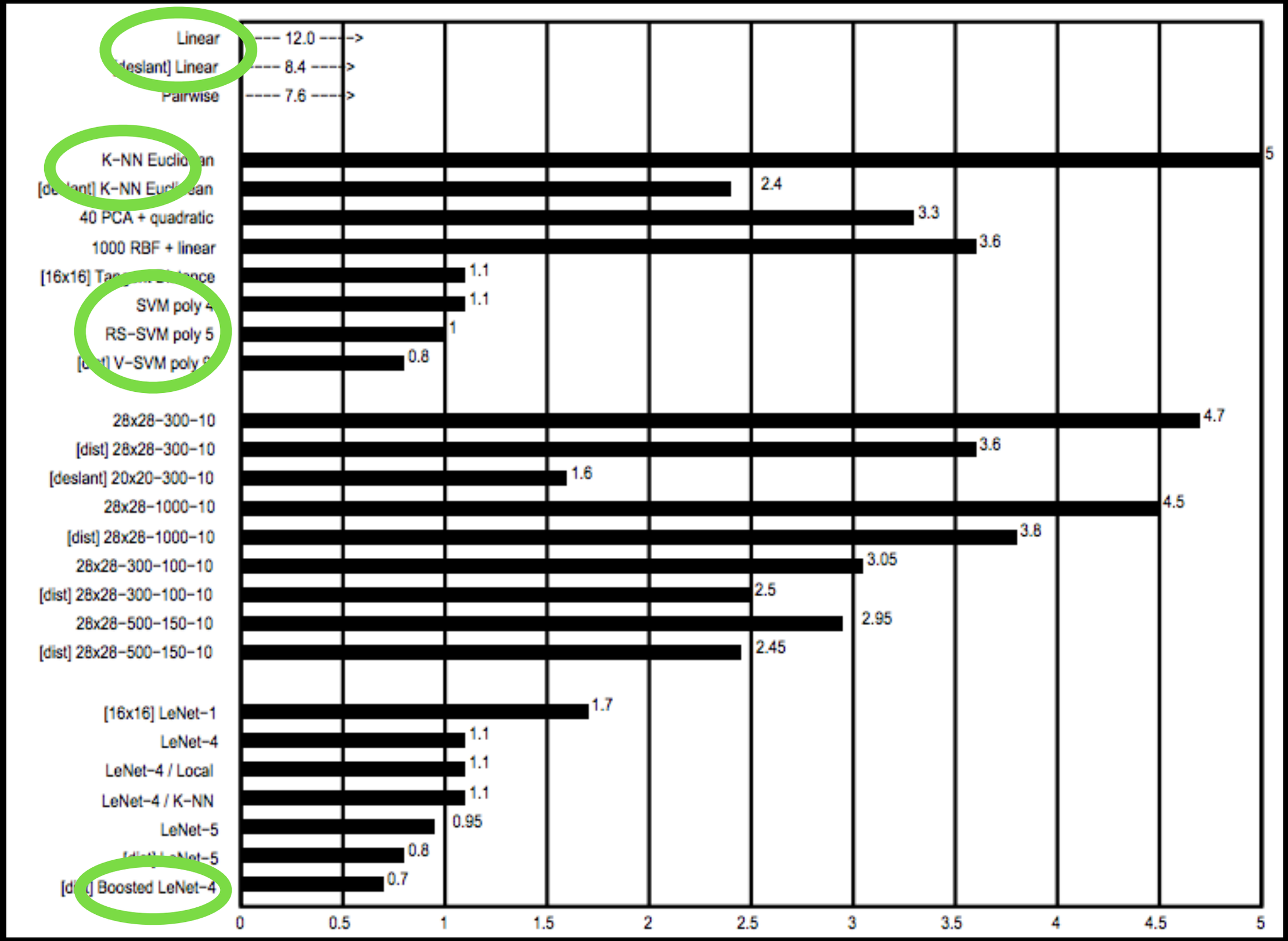
Convolutional NN (CNN)

- Image Convolution
- Feature extractor + Classifier





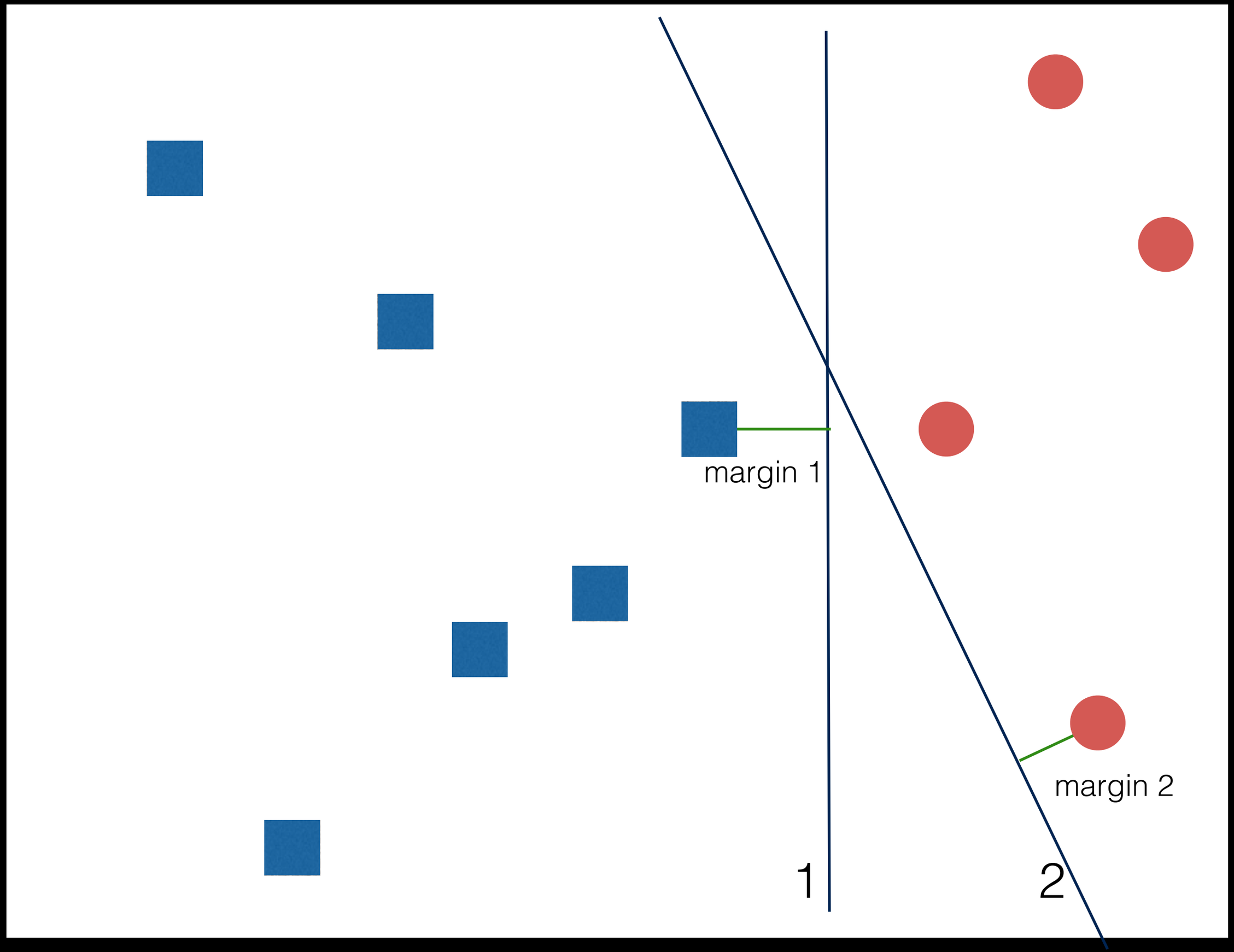
Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition", Proc. Of the IEEE, November 1998



Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition", Proc. Of the IEEE, November 1998

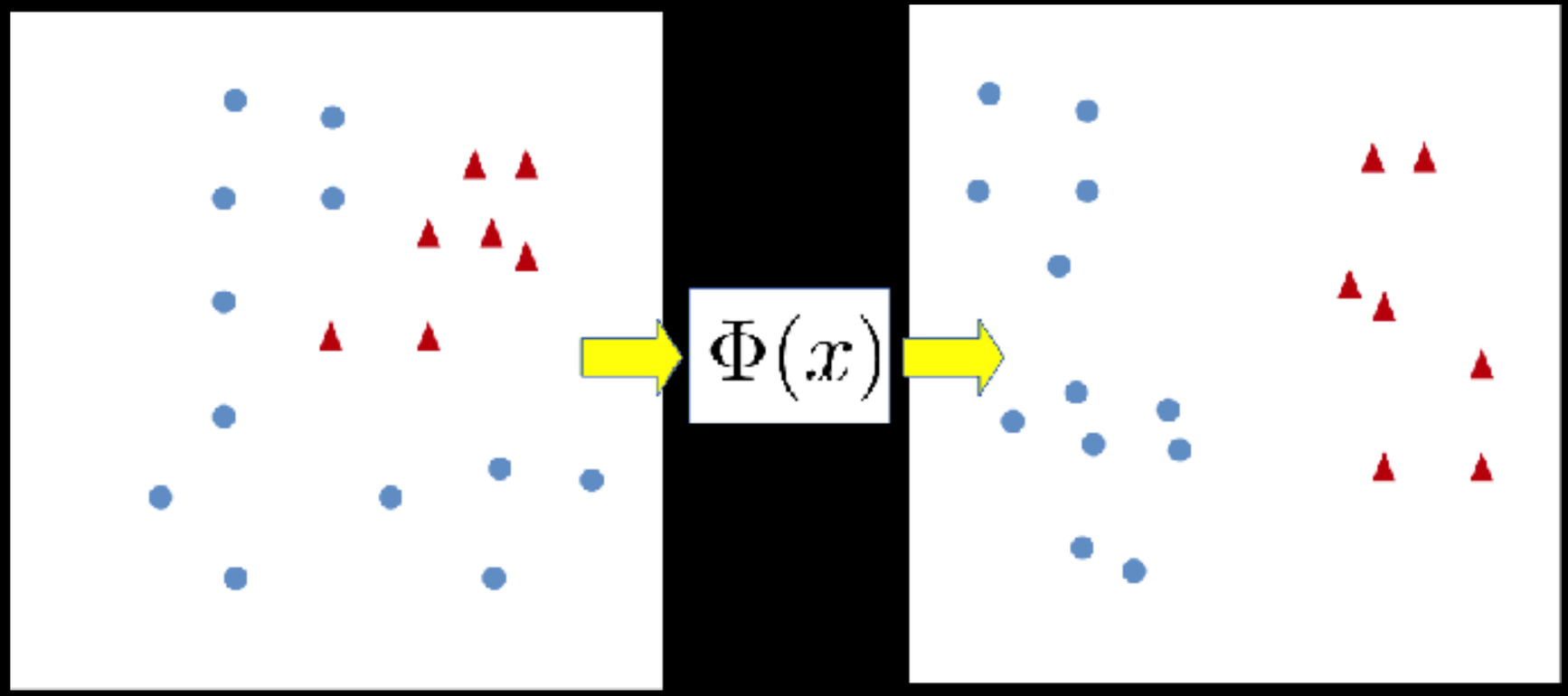
Support Vector Machine

- SVM:
  - **Linear classifier** in a space induced by a **kernel** function
  - Trained with **margin** as **regularization** term using **constrained optimization techniques**
- **Regularization** indicates preference when multiple solutions are possible
  - Image enhancement → enhance edge (high frequency) but not noise (also high frequency)
  - $(A + \lambda I)^{-1}$


margin 1
margin 2
1
2

Kernel function

- k is a valid kernel if there exists a function $\mathbf{\Phi}$ such that $k(x,y) = \mathbf{\Phi}(x)^T\mathbf{\Phi}(y)$



SVM

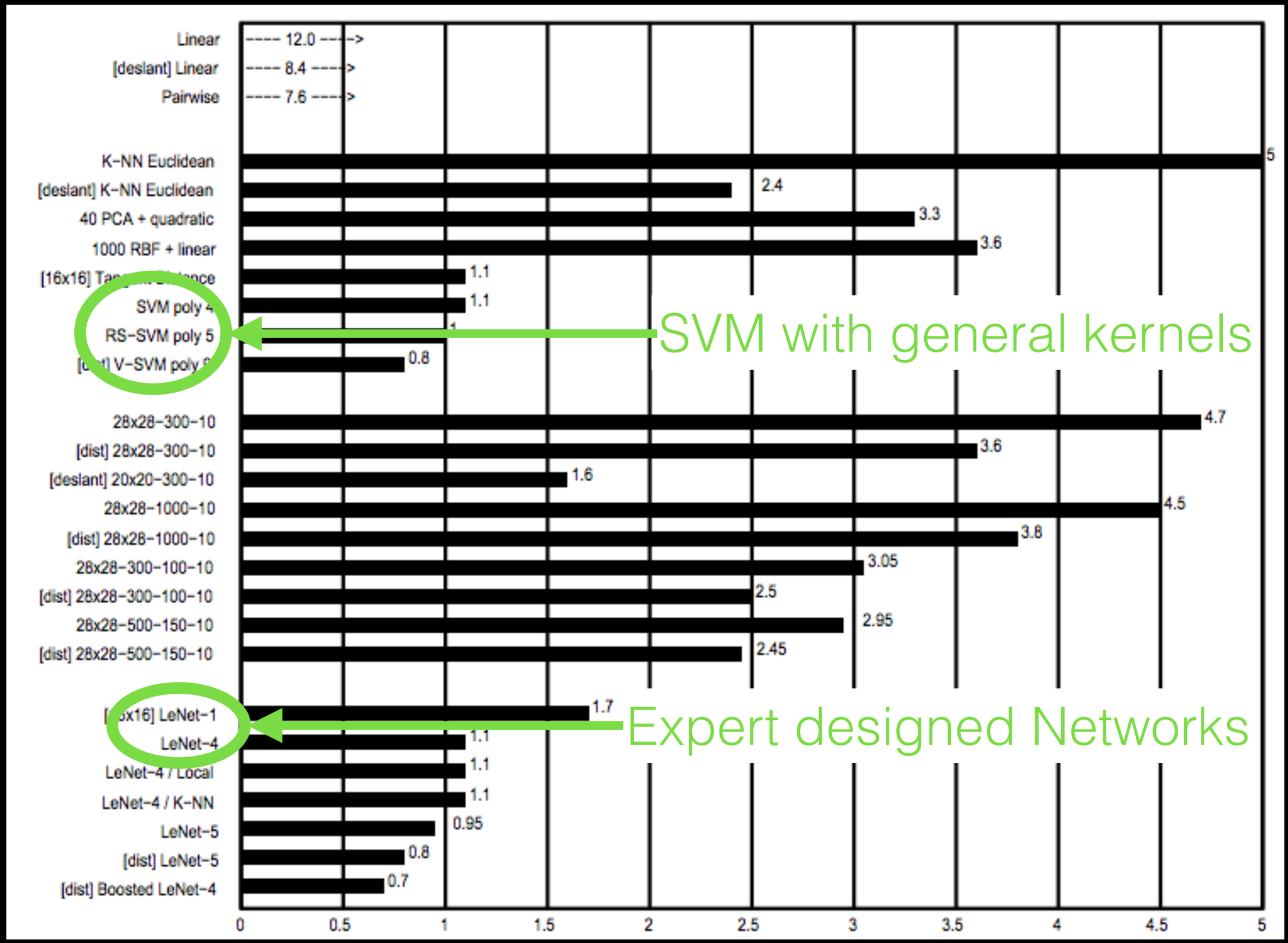
- Classification rule

$$f(x) = \text{SIGN}(\sum_i \alpha_i y_i k(x_i, x) + b)$$

- Training problem

$$\min_\alpha \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j k(x_i, x_j) - \sum_i \alpha_i$$
$$\text{subject to} \sum_i \alpha_i y_i = 0, 0 \leq \alpha_i \leq C$$

Single optimum global!!



Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition", Proc. Of the IEEE, November 1998

Comparison

- NN
  - User defined structure
  - Simple training algorithm
  - Difficult to adjust
- SVM
  - Structure discovered from training procedure
  - Complex training algorithm
  - Easy to adjust

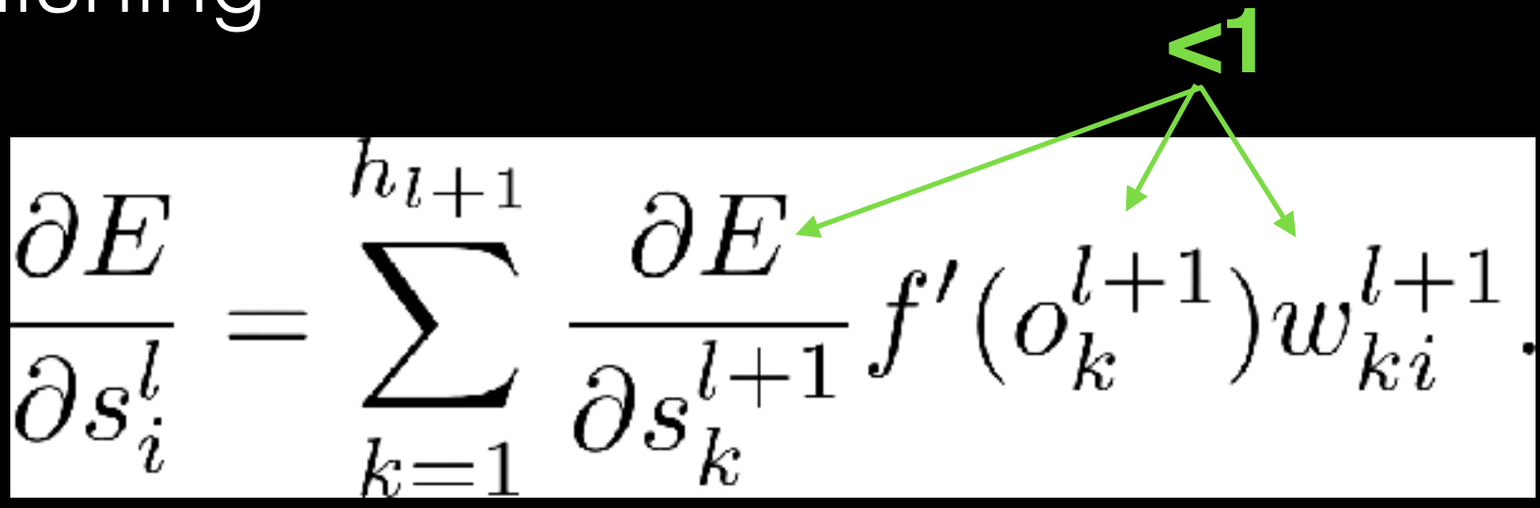
Notes

- NN and SVM are **shallowed structures**
  - SVM with tanh kernel ~ an MLP
  - MLP with one hidden layer is widely used
    - Parameter tuning → number of hidden nodes
- Universal approximator but with how many nodes?
- Biological neural networks have deep structure
  - Complex tasks: Vision and language

Deep learning

Practical problem

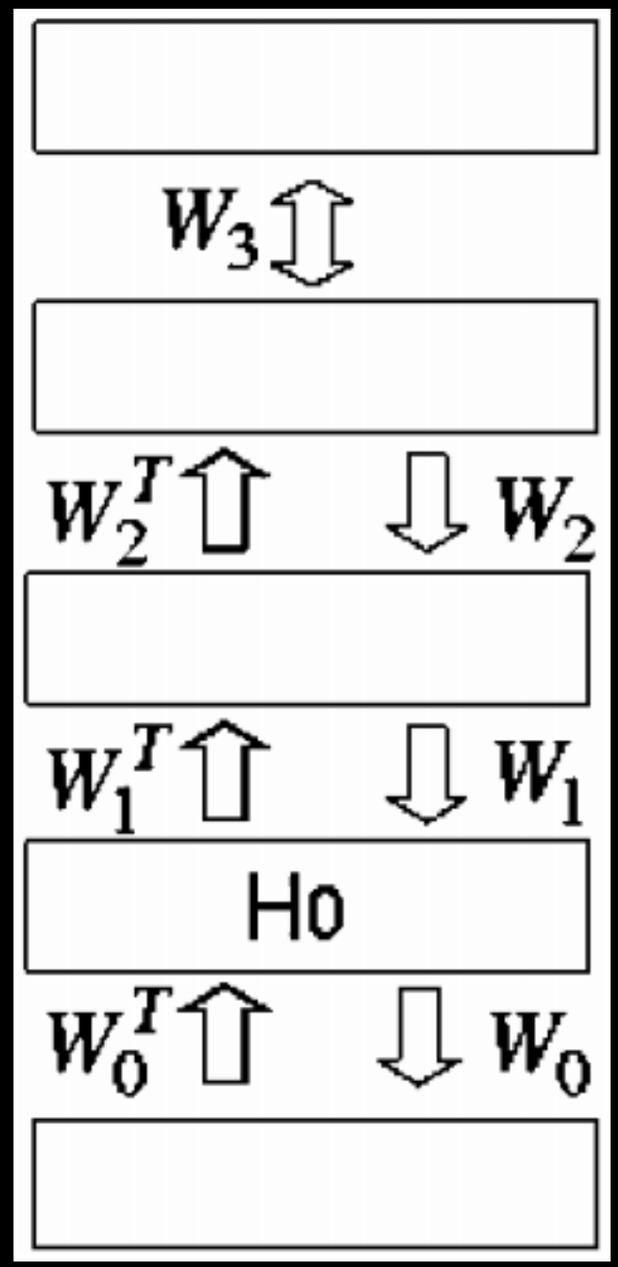
- Gradient Vanishing
  - Backprop



- Some still works
  - Random projections
  - Extreme learning machine

Solution

- Hinton, Osindero and Teh (2006)
- Greedy train a new layer consecutively
- A single layer training is based on an unsupervised training algorithm called the **Restricted Boltzmann Machine (RBM)**
- Supervised learning is used in the last layer+eventually fine tuning on lower layers



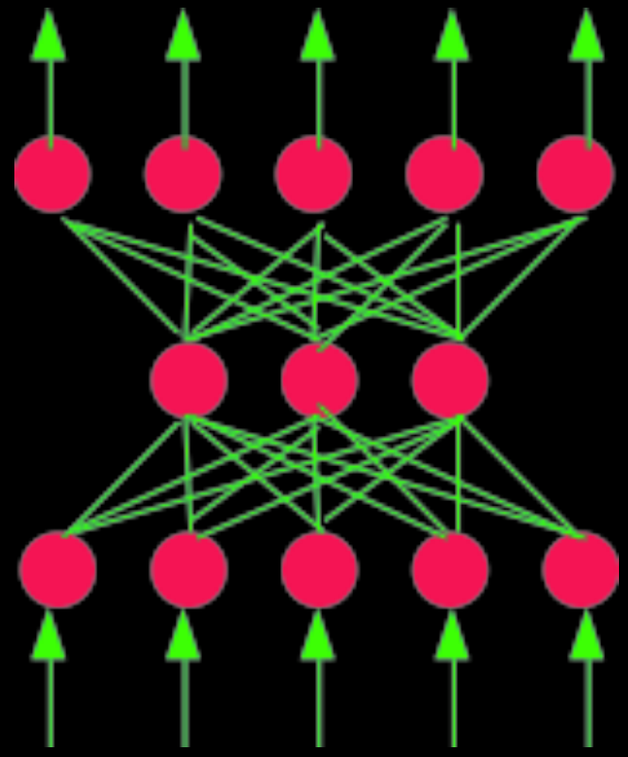
G. Hinton, S. Osindero, and Y.-W. Teh,
"A Fast Learning Algorithm for Deep Belief Nets",
In Neural Computation, 18, pp. 1527-1554, 2006

Deep Belief Nets

- Stack of RBMs
  - The hidden units of one layer become visible units of the next layer
  - Train layers consecutively
- Why stacking RBMs?
  - Improve lower variation bound on probability of training data under the model [Hinton, Osindero and Teh 2006]
- Another interpretation
  - RBM = data reconstruction
  - Use Autoencoder instead of RBM

Stack of Denoising Autoencoders

- Bengio's team
  - Train Autoencoder with corrupted input
  - Discard output layer. The hidden units of this Autoencoder become input units of the next layer
  - Supervised fine-tuning the whole structure
- Example: error on MNIST
  - 500-500-500-10 : 5.71%
  - 1000-1000-1000-10 : 1.57%
- Still lose against CNN : 0.3% (best)

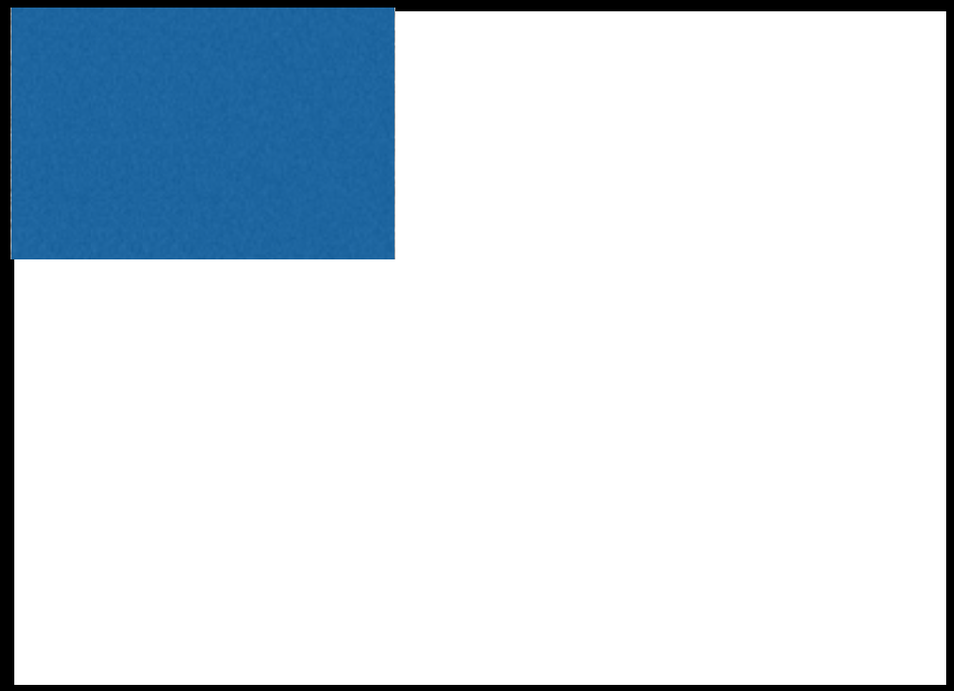


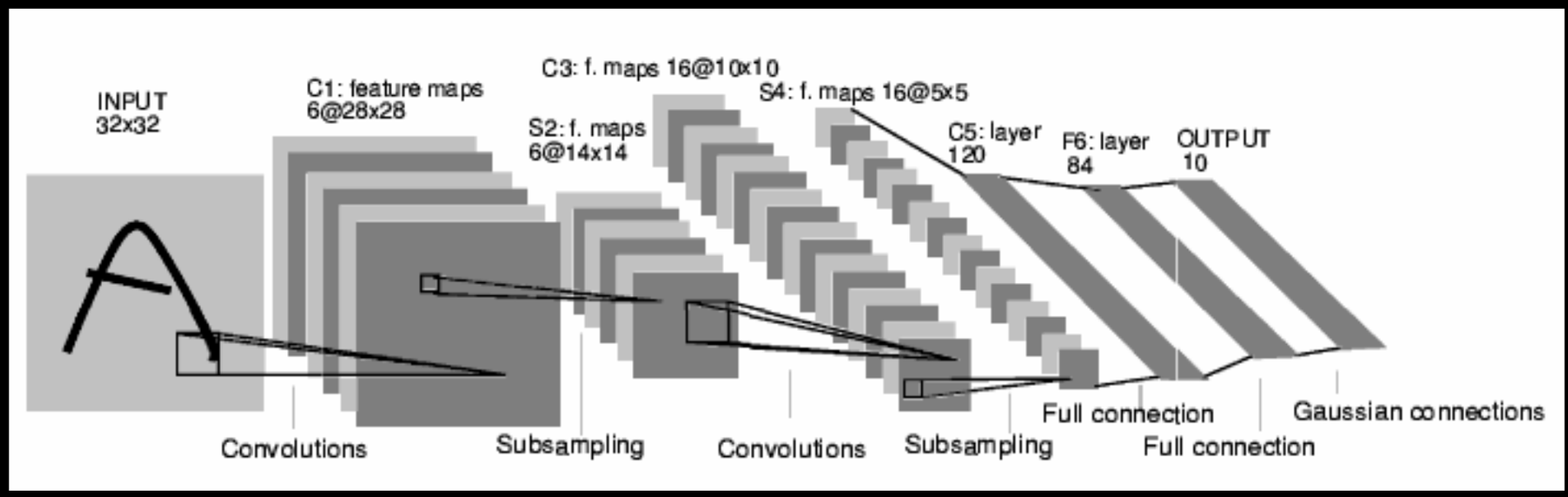
Pre-training

- Build network layer by layer successively
- Initialize the weights instead of random init
- **Does not solve the gradient vanishing problem**

Convolutional NN (CNN)

- **Image Convolution = Weights Sharing**
  - **Sum of Gradient**
- Problem with computational cost
- **GPU programming**!! (2010,2012-now)





Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition", Proc. Of the IEEE, November 1998

Rectified Linear Unit (ReLU)

- What if we have f'=1

$$\frac{\partial E}{\partial s_i^l} = \sum_{k=1}^{h_{l+1}} \frac{\partial E}{\partial s_k^{l+1}} f'(o_k^{l+1}) w_{ki}^{l+1}.$$

- ReLU(x) = max{ 0, x }

Learning technique

for deep network

Computing power:

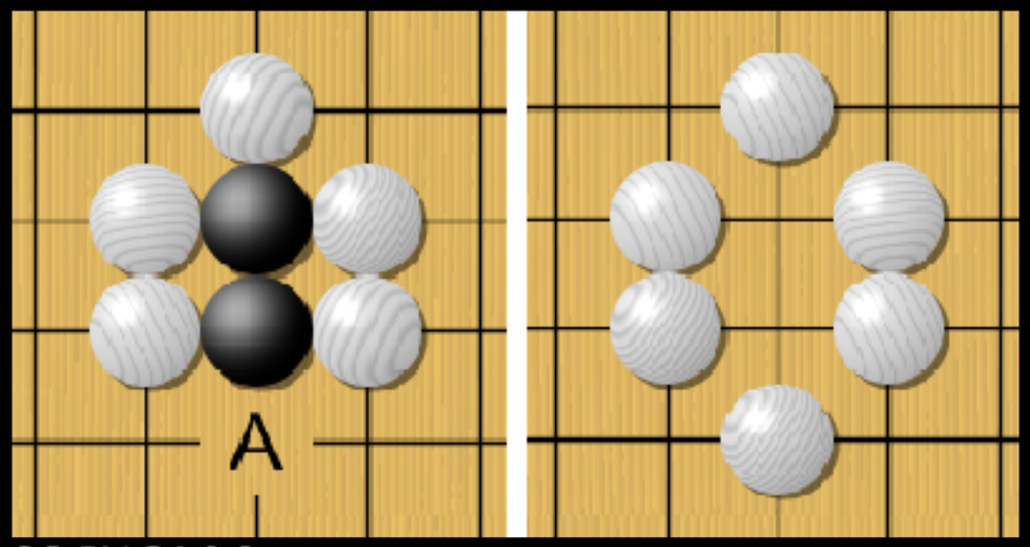
GPU, parallel, ...

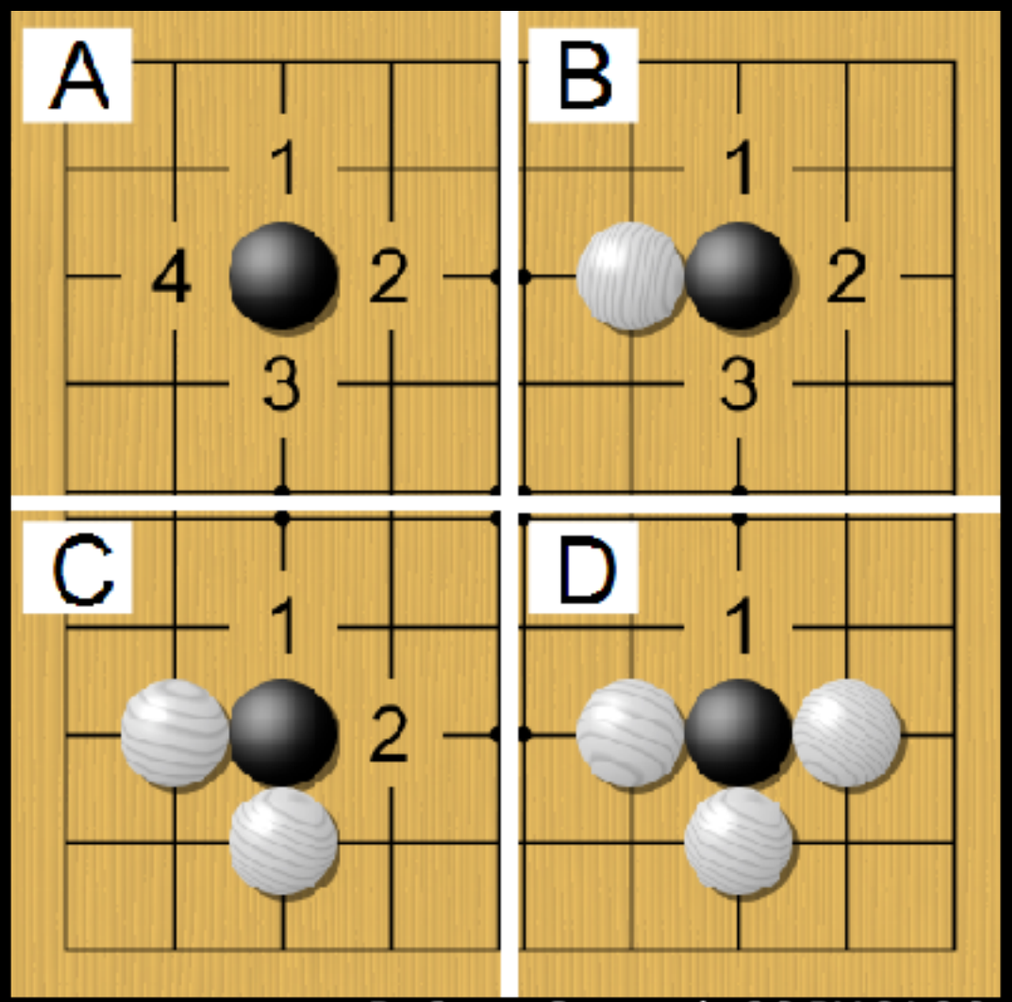
Big Data

Some applications

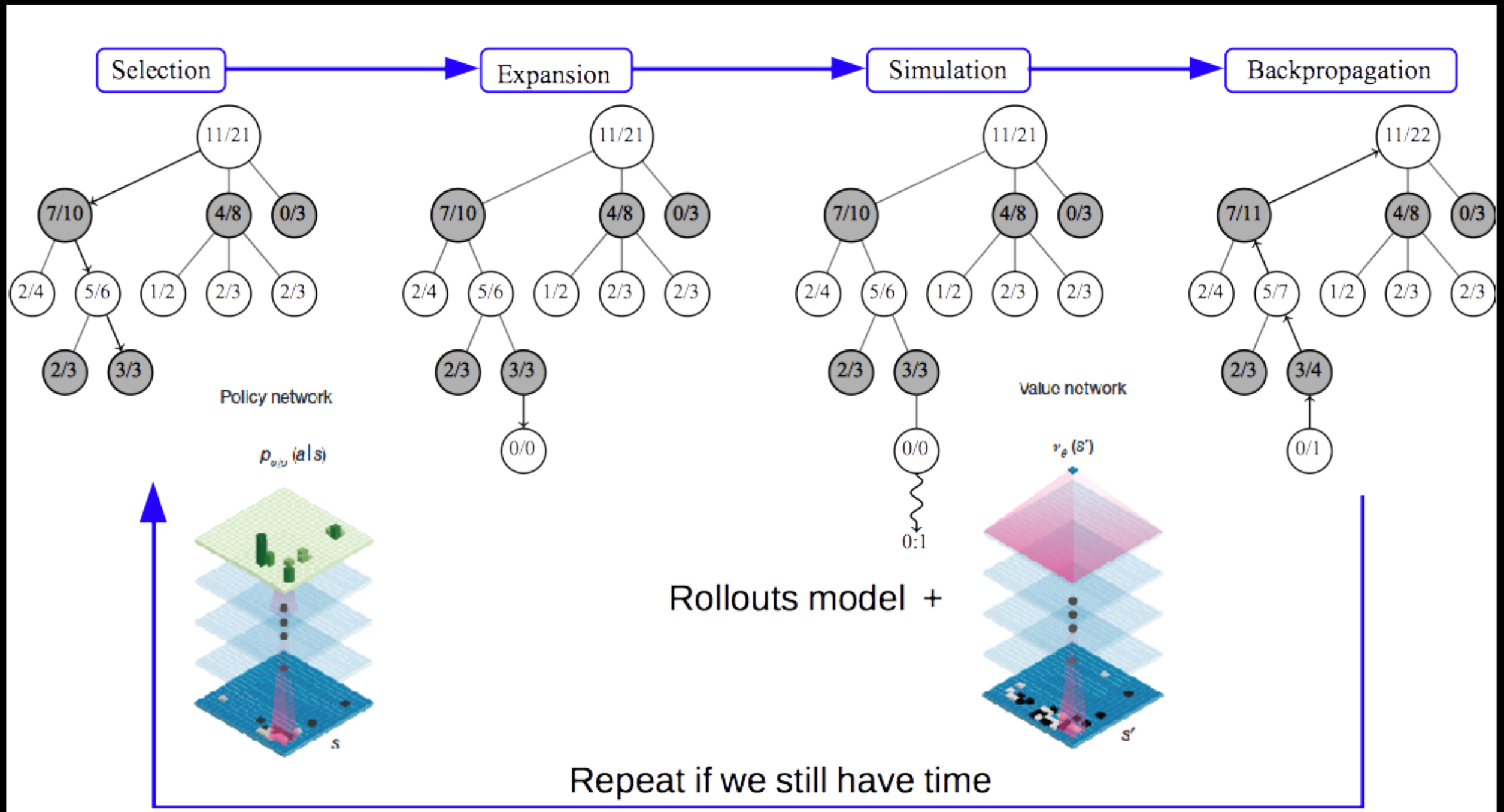
AlphaGO

- GO
  - 19x19 board, Play on intersection
  - Black first
    - Bigger area win
- AlphaGO VS. Lee Sedol



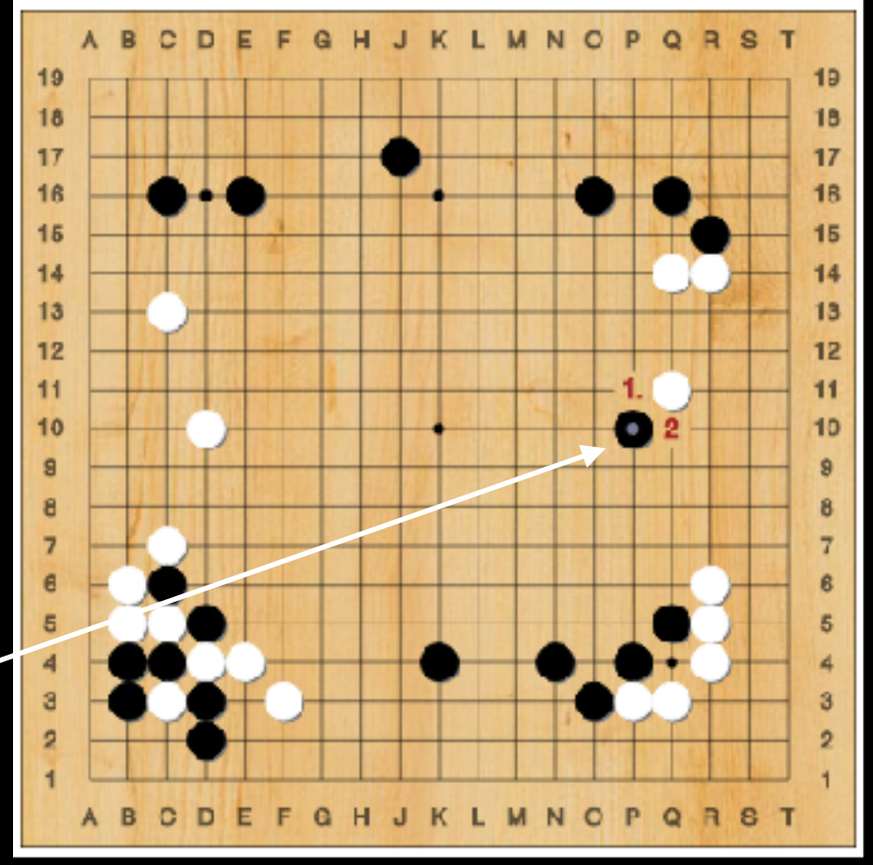


Monte-Carlo Tree Search + Policy Network and Value Network



Lesson learned

- 1 in 10,000 moves
  - AlphaGO: Match 2, Black Move 37
  - Lee Sedol: Match 4, White Move 78
- AlphaGO considers 27 moves ahead
- Rollouts ~rule-based model
  (response patterns, non-response patterns)
- MCTS does not use previous decision but AlphaGO moves are still consistent
  - Temporal encoding



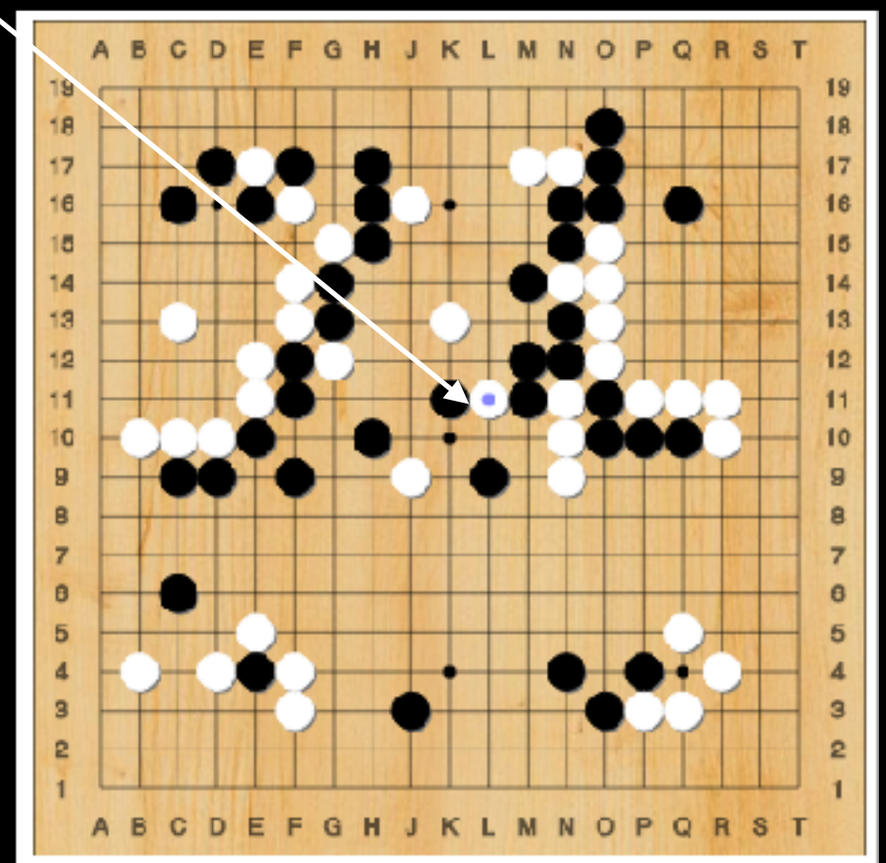


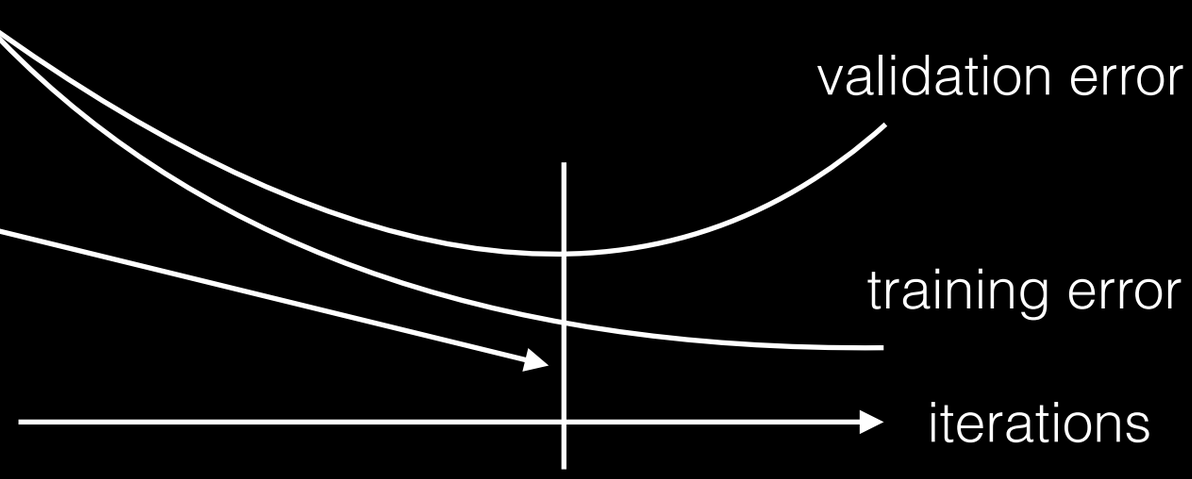
Image processing

ImageNet challenge

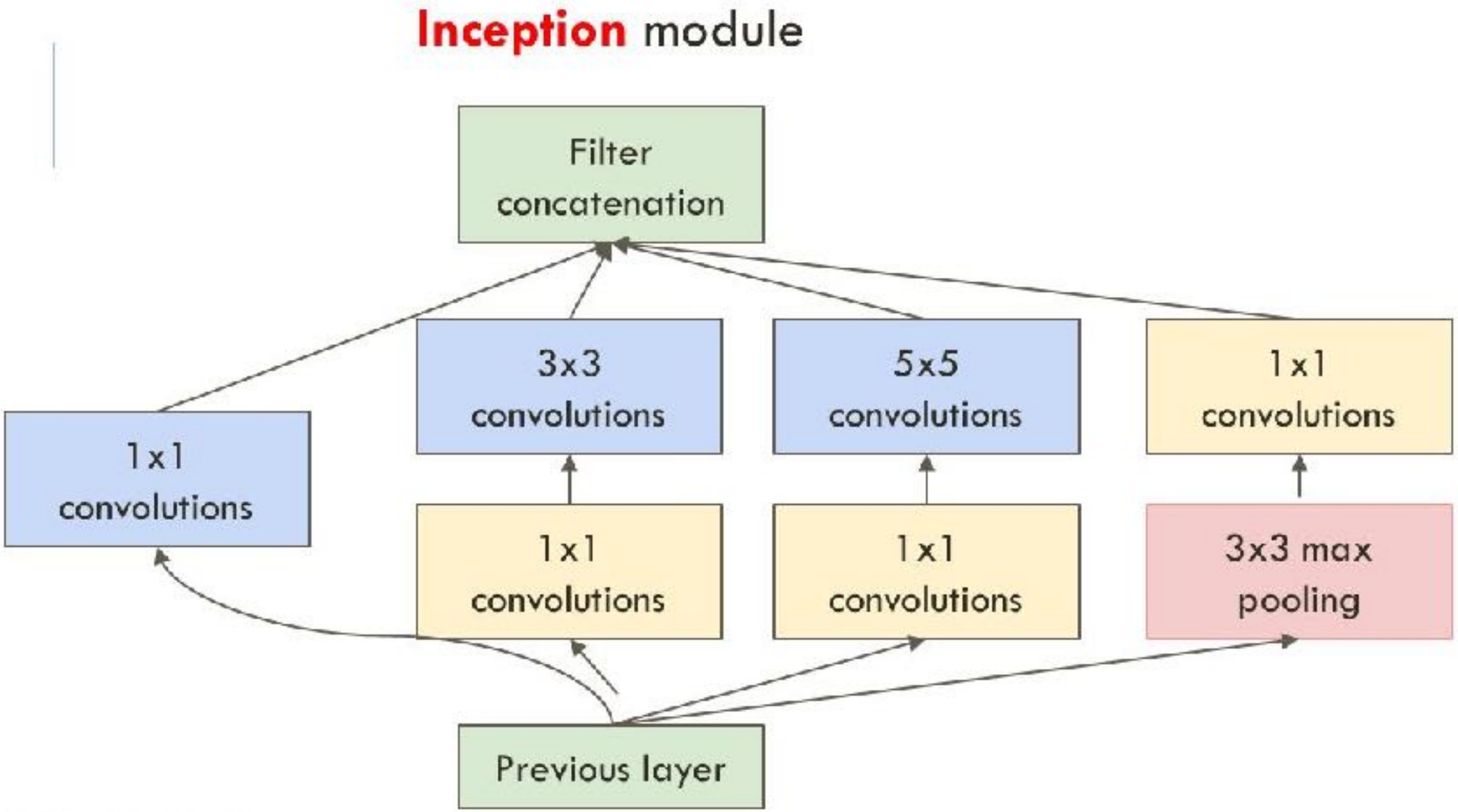
- 2010-2012: SVM + Spatial Pyramid + local features
- 2012: AlexNet (7 layers, 60M params, **Drop-out, ReLU, GPU**)
- 2013: OverFeat (8 layers, **bounding box regression**)
- 2014: GooLeNet (20 “layers”, 1M params, **Inception module**), VGG (3x3 kernel, 20 layers)
- 2015: ResNet (150 layers, **skip connection**)
- 2016: Combined model (ResNet, Inception, Inception-ResNet, Wide-ResNet, …)

Overfit problem

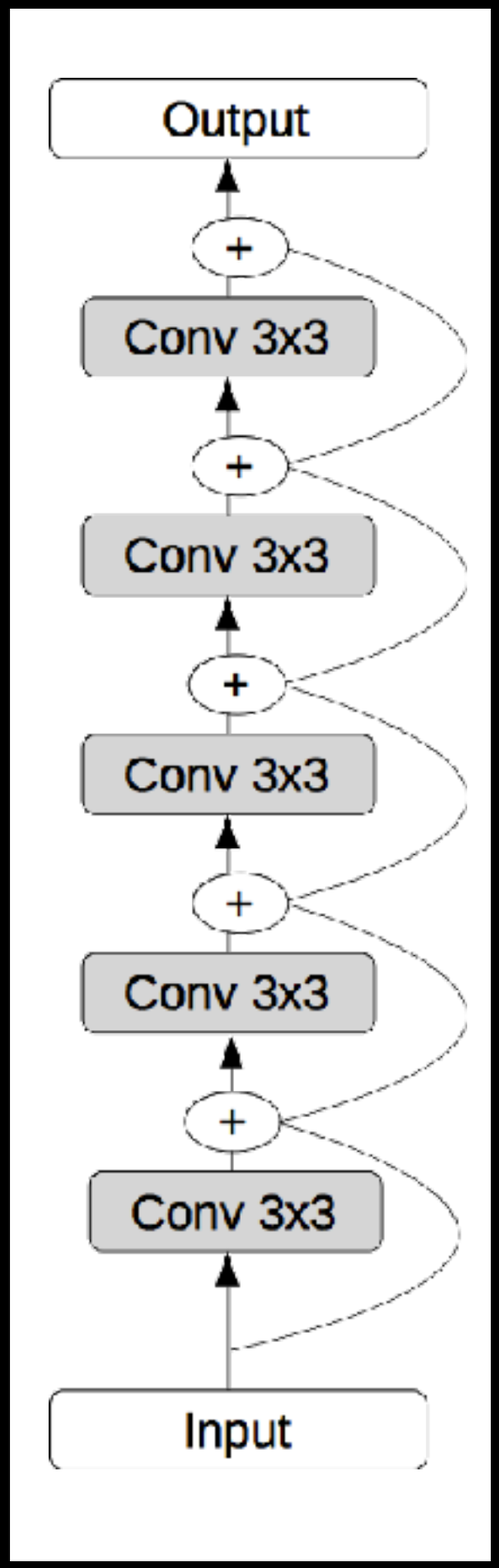
- Understand VS memorizing
- Rule of thumbs: when #params is large the model tends to be overfit
- Problem: NN structure is defined first!
- Solution
  - Early stopping
  - Weights decay
  - Optimal brain damage
  - **Drop-out** ~simulated brain damage



Inception module

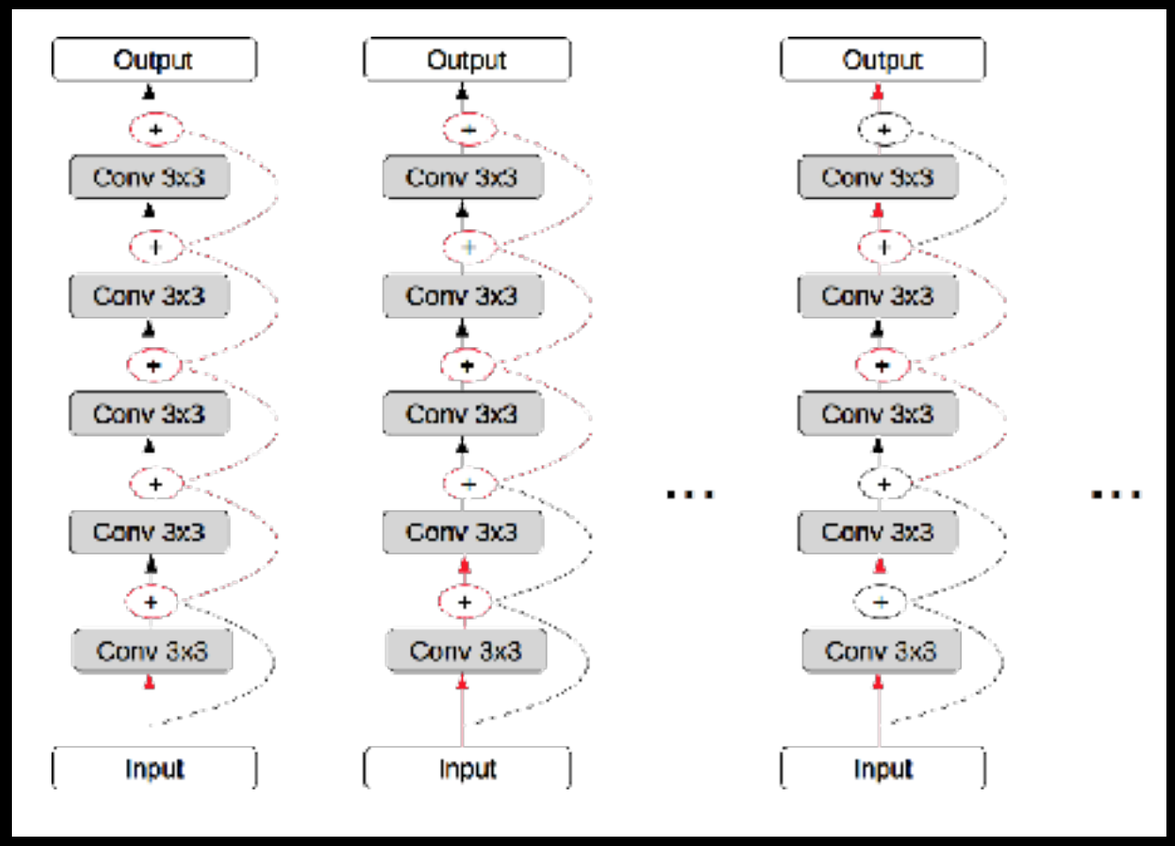


SLIDE CREDIT:GOOGLE INC



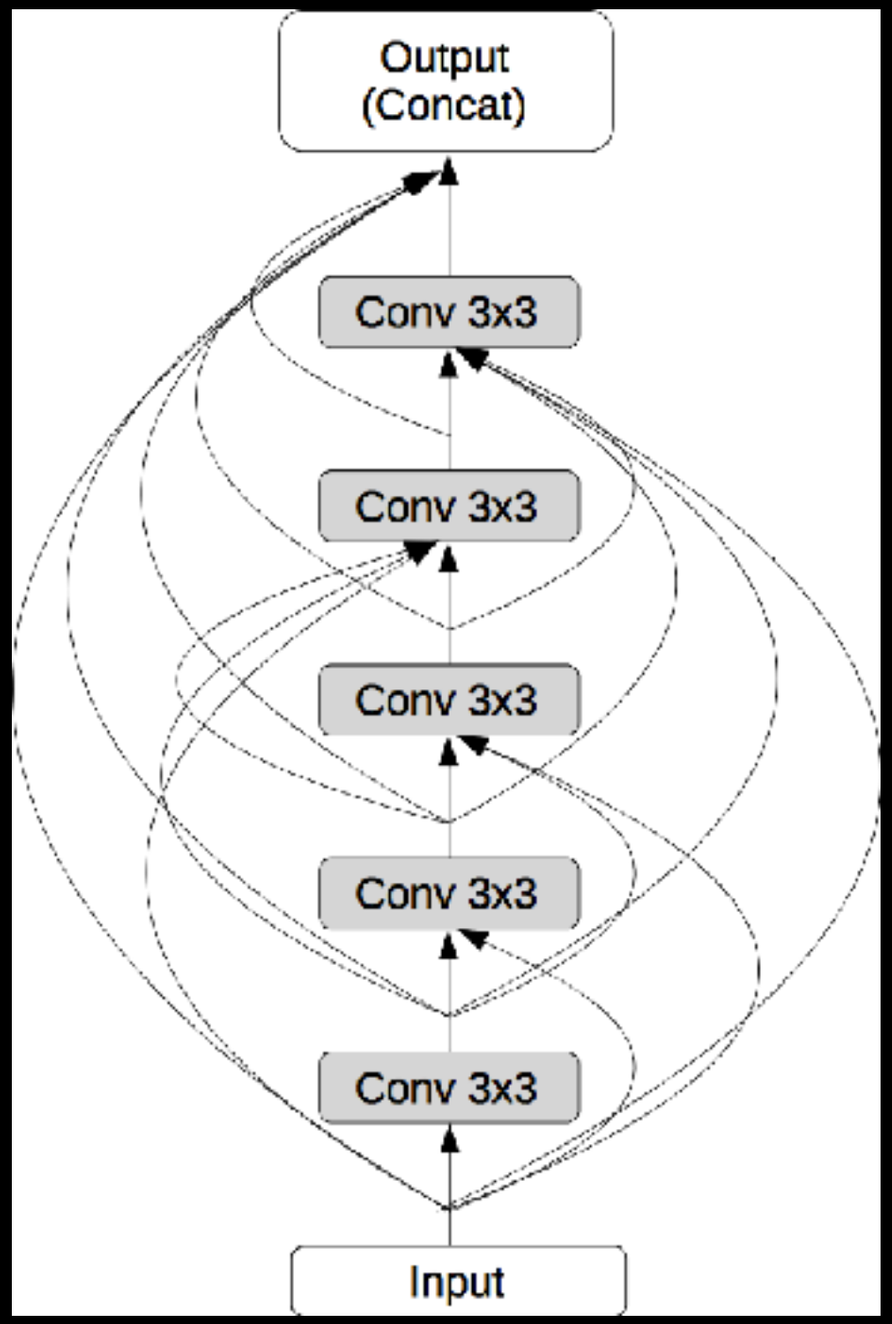
ResNet

- Add skip connections
- Weights of unnecessary blocks will be driven toward zeros -> residual
- Acts like mixture of several shallower networks



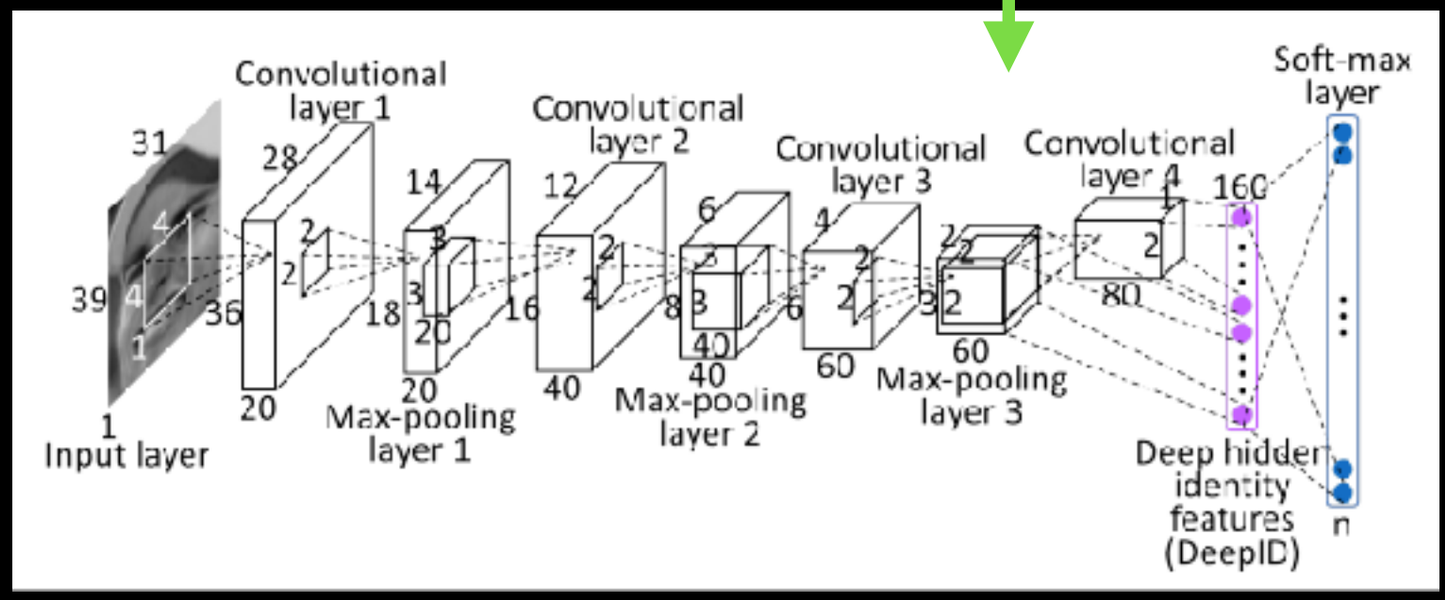
ResNet variations

- Skip connections
  - Wide ResNet
  - ResNet in ResNet
  - Inception ResNet
- DenseNet



Labeled faces in the wild

- Human ~97.5%
- DeepFace >97% (Facebook)
- FaceNet > 99% (Google)
- Face++ > 99% (Commercial, China)
- DeepID > 99% (Commercial, China)

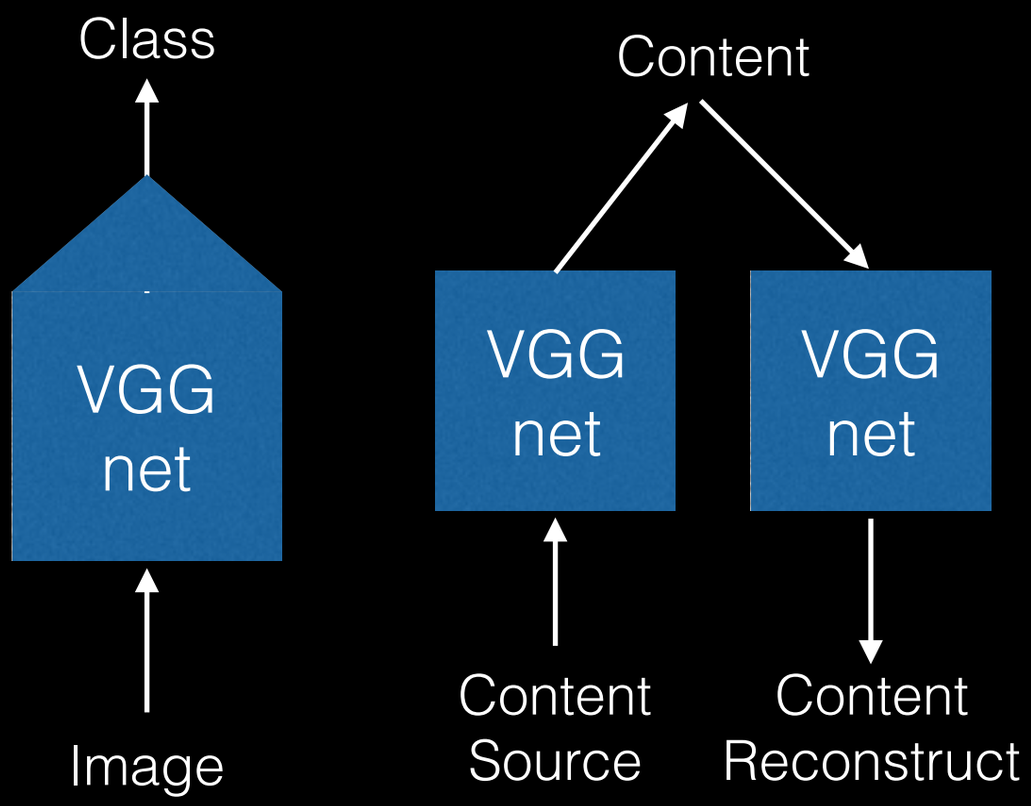


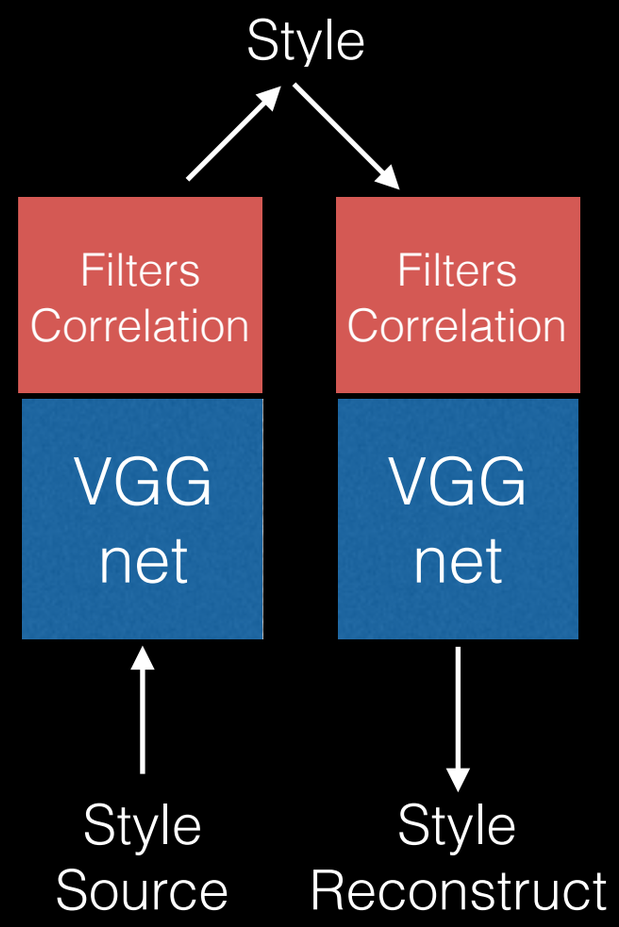
Style transfer

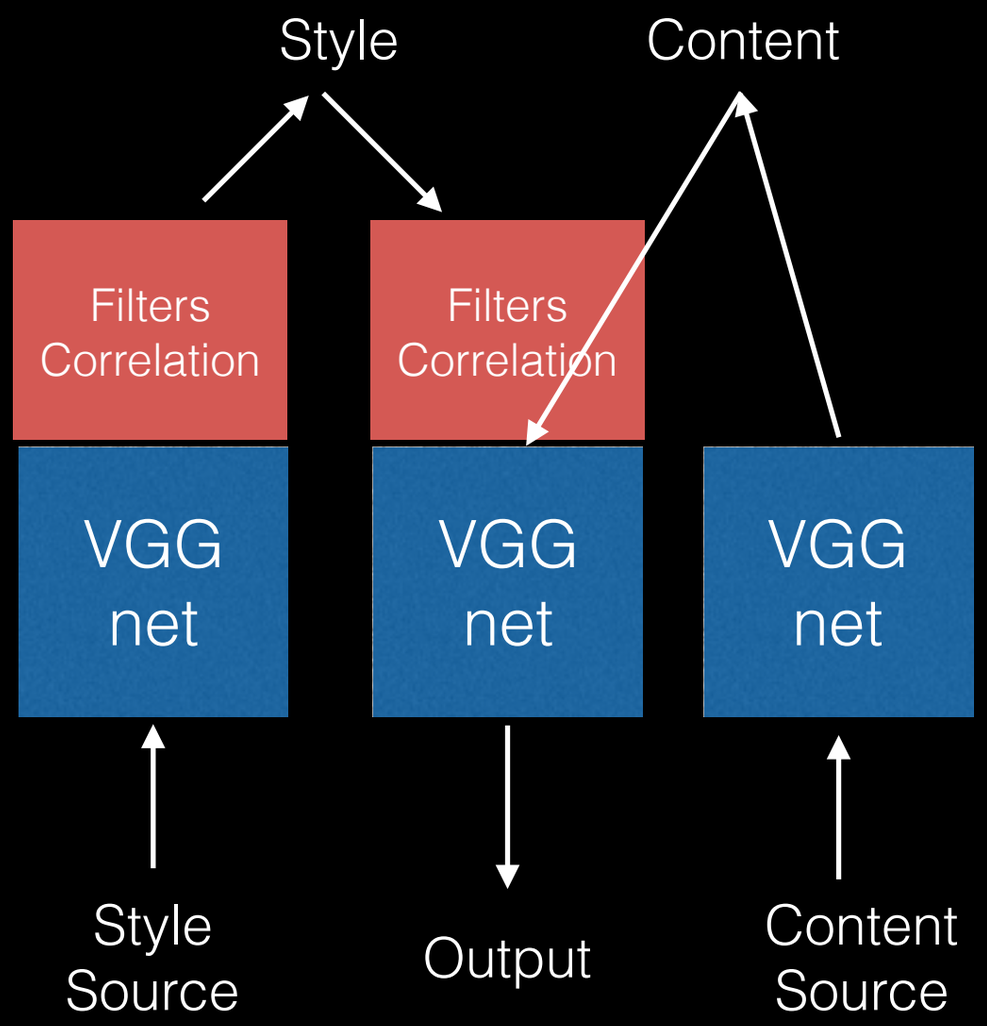
L.A. Gatys, A.S. Ecker, and M. Bethge, “Image Style Transfer Using Convolutional Neural Networks” CVPR 2016

Style transfer

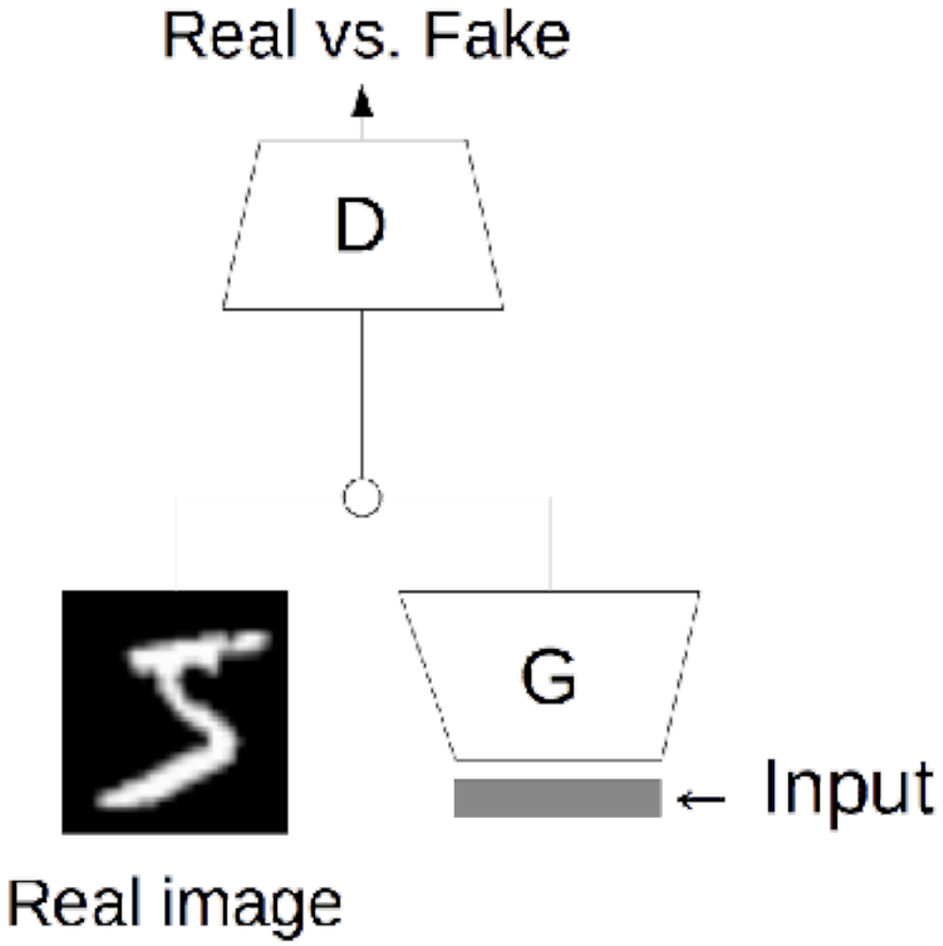
- Basic idea
  - Internal representation = content
  - Convolution layer = filter banks
  - Correlation between filters output = style
  - Use Gradient backprop to recon image


Class
VGG net
Image
Content
VGG net
VGG net
Content Source
Content Reconstruct


Style
Filters Correlation
Filters Correlation
VGG net
VGG net
Style Source
Style Reconstruct


Style
Content
Filters Correlation
Filters Correlation
VGG net
VGG net
VGG net
Style Source
Output
Content Source

Generative Adversarial Networks (GAN)



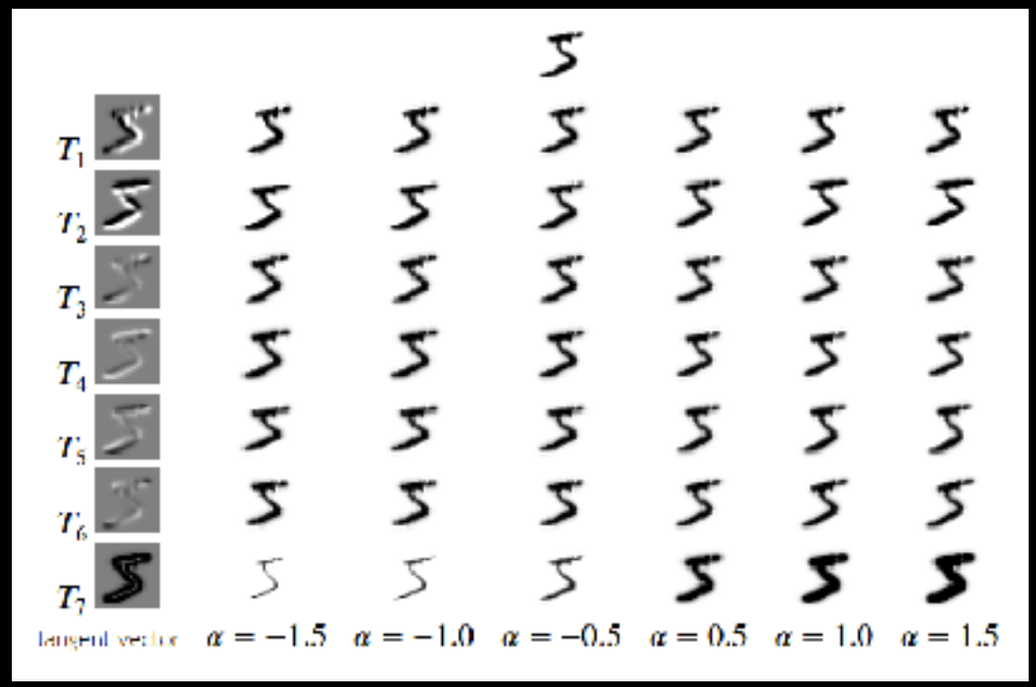
Generative Adversarial Nets

Ian J. Goodfellow, Jean Pouget-Abadie,* Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair,† Aaron Courville, Yoshua Bengio‡
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

https://arxiv.org/pdf/1406.2661.pdf

Data augmentation

- Tangent vectors for handwritten
  - MNIST test error 0%!!
- Random crop + transform
- **New solution Generator**



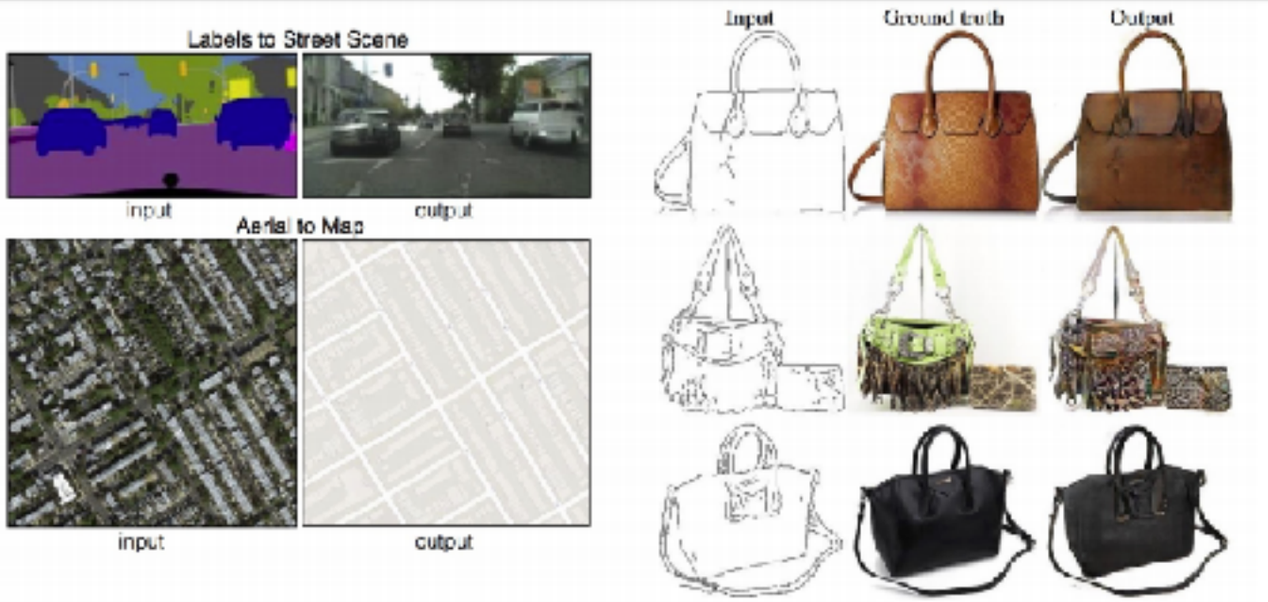


Figure 7: Isola *et al.* (2016) created a concept they called image to image translation,

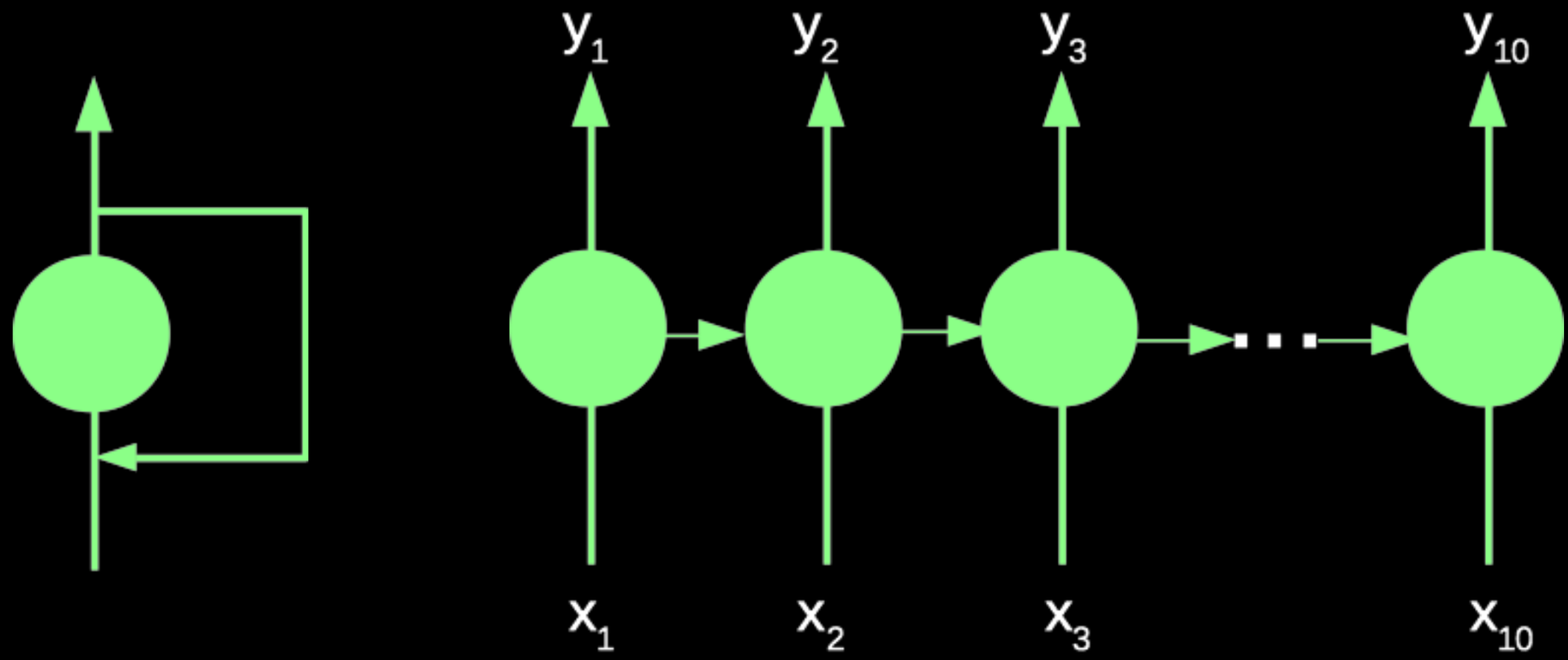
Figure 5: Zhu *et al.* (2016) developed an interactive application called *interactive generative adversarial networks* (iGAN). A user can draw a rough sketch of an image, and iGAN uses a GAN to produce the most similar realistic image. In this example, a

Figure 4: Ledig *et al.* (2016) demonstrate excellent single-image superresolution results

Ian Goodfellow, “NIPS 2016 Tutorial: Generative Adversarial Networks”
https://arxiv.org/pdf/1701.00160.pdf

Text processing

Recurrent Networks (1)



1 layer

1 layer when backprop from y_1

10 layers when backprop from y_{10}

Gradient Vanishing too!!

Recurrent Networks (2)

- Conceptual : input x_t, output y_t, hidden state h_t
  - $h_t = f_1(x_t, y_{t-1}, h_{t-1})$ and $y_t = f_2(x_t, y_{t-1}, h_{t-1})$
- Friendly path for Gradient backprop
  - $h_t = h_{t-1} + \tanh(Ux_t + Wy_{t-1})$ and $y_t = \tanh(h_t)$
- Some information should be **forgotten**, some information are **noise**, some information should be **filtered** out before output

Recurrent Networks (3)

- LSTM: the above idea with **learnable gates**
  - $f = sigm(U^f x_t + W^f y_{t-1})$
  - $i = sigm(U^i x_t + W^i y_{t-1})$
  - $o = sigm(U^o x_t + W^o y_{t-1})$
- GRU
  - 2 gates: **reset** and **update**

Word2Vec

- Transform word into numerical vector
- Bag of Words (BoW)
- Autoencoder + BoW + reconstruct **next word**
- Use output of hidden nodes as output vector
- Surprised!!:
  - vec(King) - vec(Man) + vec(Woman) = vec(Queen)
  - vec(Vietnam) + vec(Capital) = vec(Hanoi)

Useful tools

- Caffe (Python, C++)
- Tensorflow (Python, C++)
- Theano (Python)
- Keras (Python)
- Torch (Lua)