[Spring, 2017]

Short Introduction to Deep Learning

Pattern Recognition (BRI623)

Heung-Il Suk

hisuk@korea.ac.kr

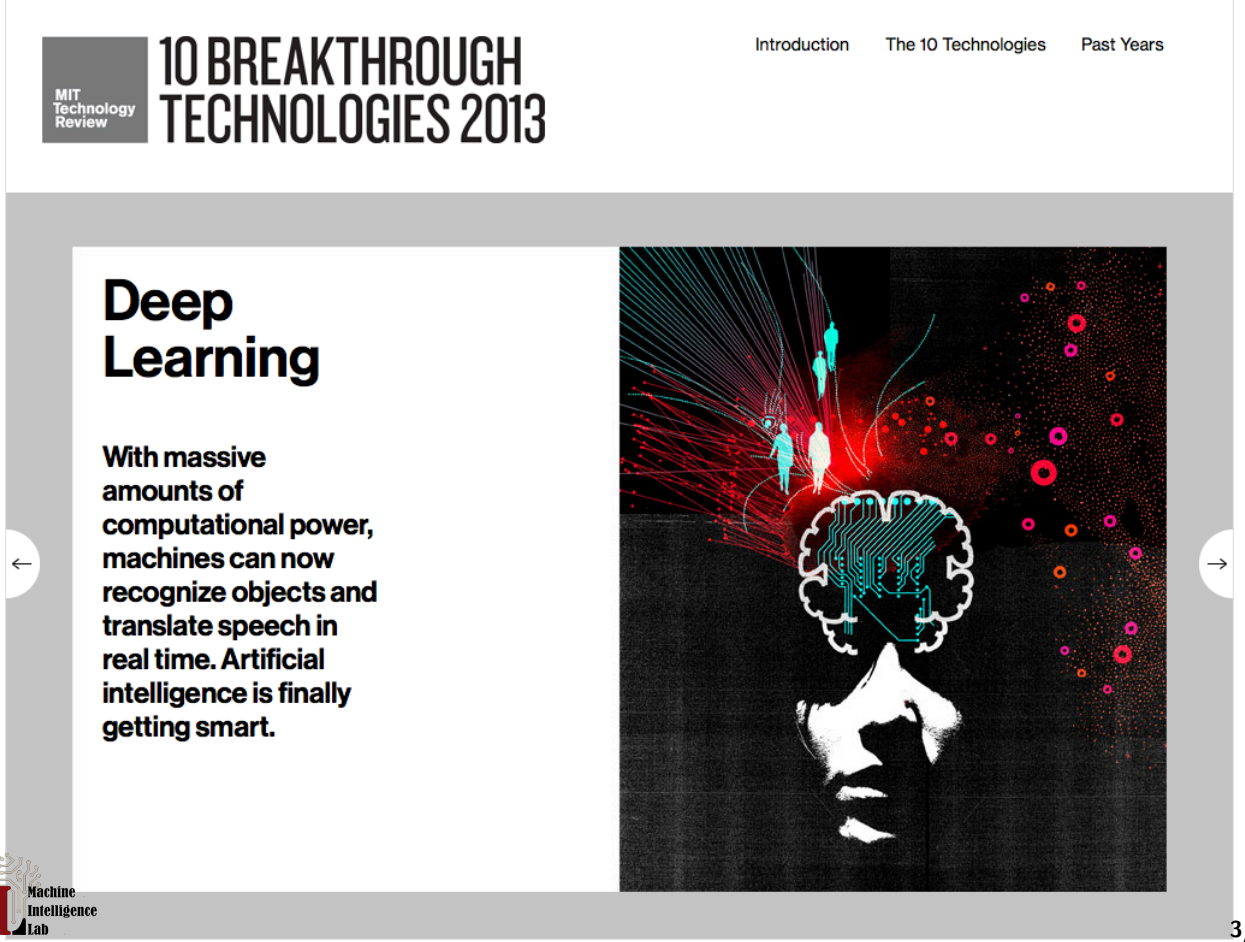
http://www.ku-milab.org

Department of Brain and Cognitive Engineering,
Korea University

Contents

1. Introduction
2. What is Deep Learning?
3. Difficulties in Deep Learning
4. Deep Models
5. Stacked Auto-Encoder
6. Algorithmic Advances for Better Learning

Introduction

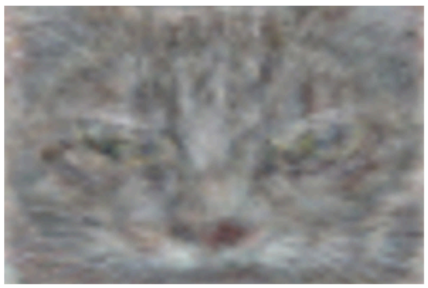


MIT Technology Review

Google Puts Its Virtual Brain Technology to Work

A powerful new approach to artificial intelligence is ready to improve many Google products.

By Tom Simonite on October 5, 2012



Platonic ideal: This composite image represents the ideal stimulus that Google's neural network recognizes as a cat face.

This summer Google set a new landmark in the field of artificial intelligence with software that learned how to recognize cats, people, and other things simply by watching YouTube videos (see "Self-Taught Software"). That technology, modeled on how brain cells operate, is now being put to work making Google's products smarter, with speech recognition being the first service to benefit.

Google's learning software is based on simulating groups of connected brain cells that communicate and influence one another. When such a neural network, as it's called, is exposed to data, the relationships between different neurons can change. That causes the network to develop the ability to react in certain ways to incoming data of a particular kind – and the network is said to have learned something.

Machine Intelligence Lab

The New York Times

Scientists See Promise in Deep-Learning Programs

Hao Zhang/The New York Times

A voice recognition program translated a speech given by Richard F. Rashid, Microsoft's top scientist, into Mandarin Chinese.

By JOHN MARKOFF
Published: November 23, 2012

Using an artificial intelligence technique inspired by theories about how the brain recognizes patterns, technology companies are reporting startling gains in fields as diverse as computer vision, speech recognition and the identification of promising new molecules for designing drugs.

FACEBOOK
TWITTER
GOOGLE+
SAVE

Machine Intelligence Lab

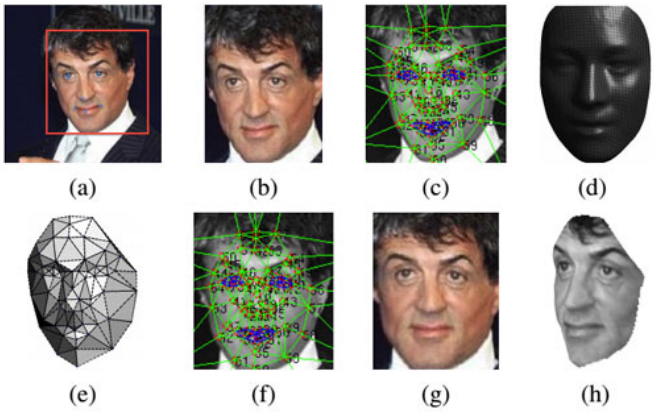
MIT Technology Review

Facebook Creates Software That Matches Faces Almost as Well as You Do

Facebook's new AI research group reports a major improvement in face-processing software.

By Tom Simonite on March 17, 2014

Asked whether two unfamiliar photos of faces show the same person, a human being will get it right 97.53 percent of the time. New software developed by researchers at Facebook can score 97.25 percent on the same challenge, regardless of variations in lighting or whether the person in the picture is directly facing the camera.



What is Deep Learning?

What is Deep Learning?

Part of a broader family of machine learning methods
based on learning representations of data

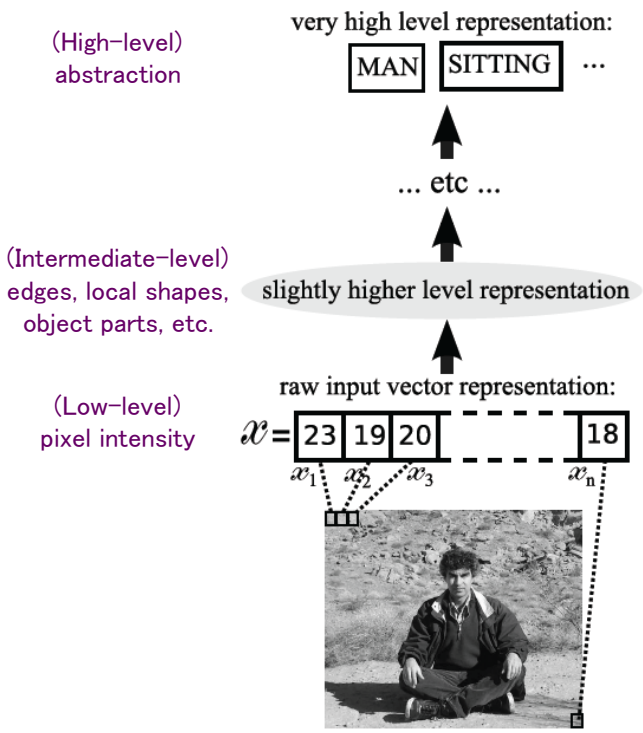
In terms of a model architecture...

- Most current learning algorithms are shallow architectures
  - *e.g.*, support vector machine

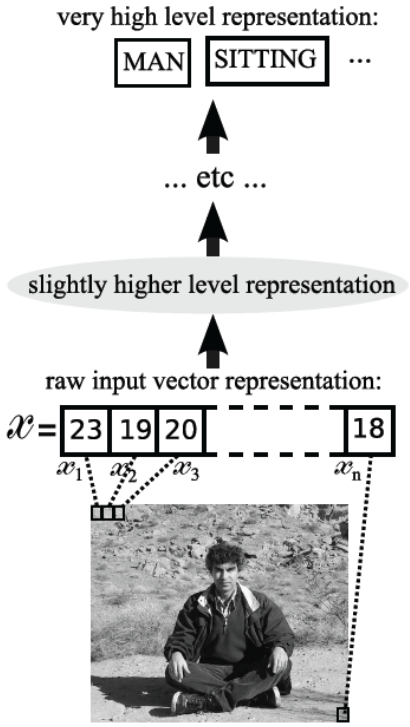
$$\hat{f}(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x})\right)$$

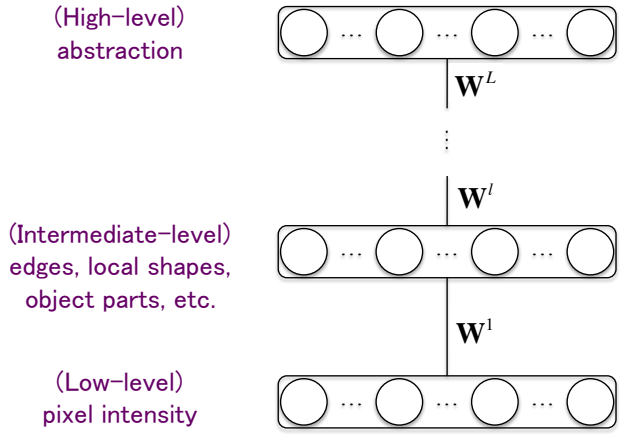
- Deep architecture
  - Composed of multiple levels of *non-linear* operations, such as neural nets with many hidden layers

Goal of deep architectures: *to learn feature hierarchies*



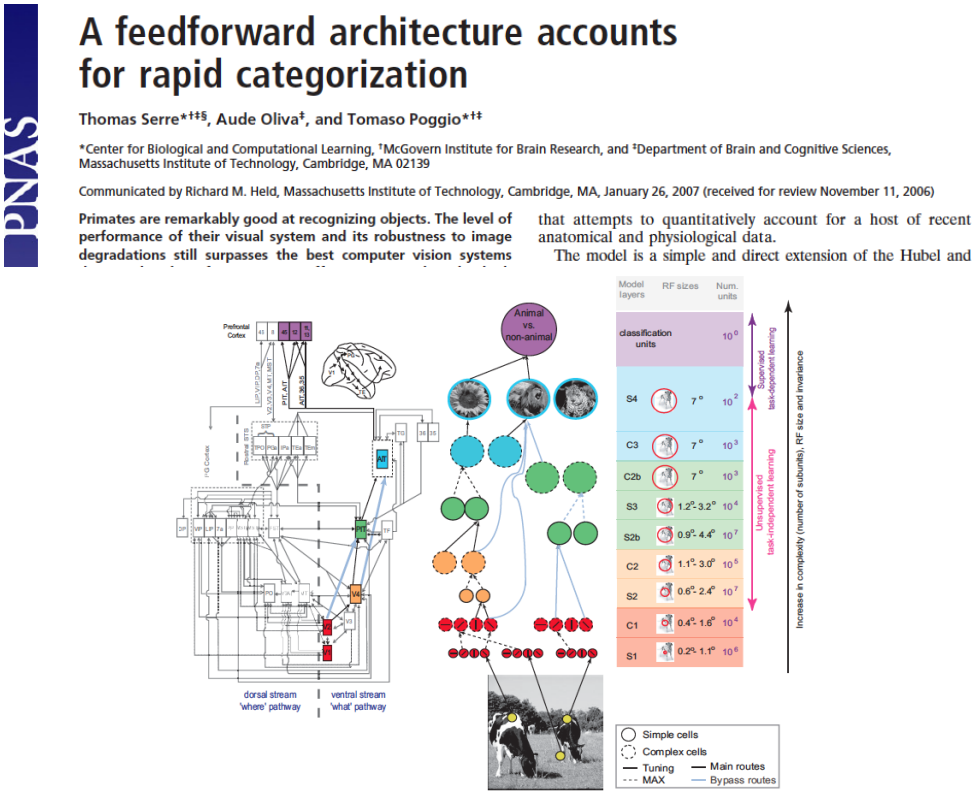
(Image credit: Bengio, 2009)





Deep architecture
for hierarchical feature representation

Biological Motivation: Deep Architecture



PNAS

A feedforward architecture accounts for rapid categorization

Thomas Serre*†‡§, Aude Oliva‡, and Tomaso Poggio*†‡

*Center for Biological and Computational Learning, †McGovern Institute for Brain Research, and ‡Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA 02139

Communicated by Richard M. Held, Massachusetts Institute of Technology, Cambridge, MA, January 26, 2007 (received for review November 11, 2006)

Primates are remarkably good at recognizing objects. The level of performance of their visual system and its robustness to image degradations still surpasses the best computer vision systems

that attempts to quantitatively account for a host of recent anatomical and physiological data.

The model is a simple and direct extension of the Hubel and

(Serre *et al.*, 2007)

Biological Motivation: Feature Representation

nature

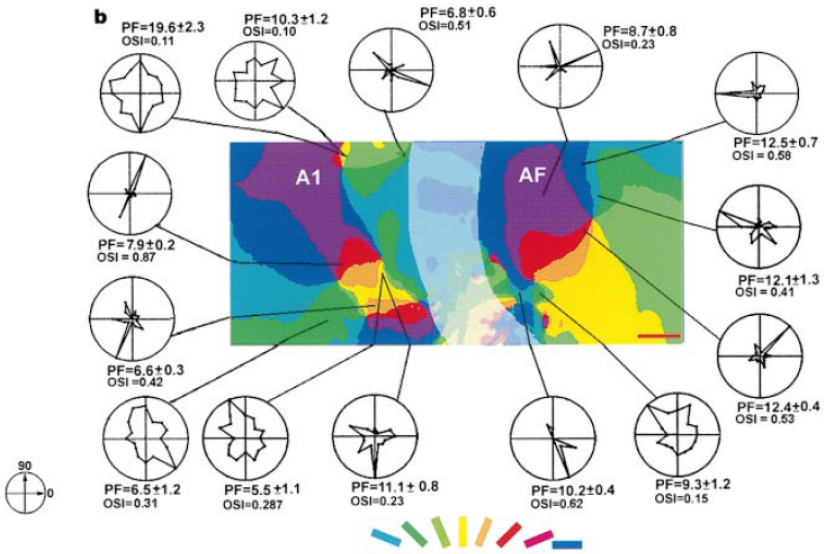
articles

Induction of visual orientation modules in auditory cortex

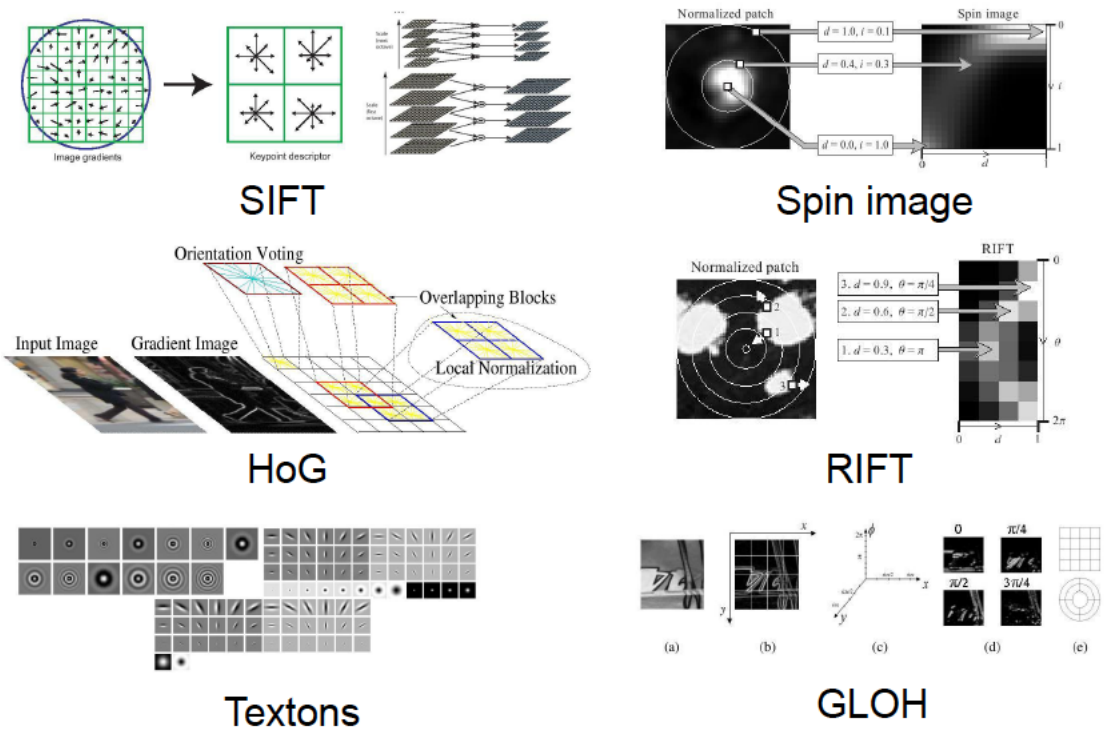
Jitendra Sharma, Alessandra Angelucci* & Mriganka Sur

Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA

Modules of neurons sharing a common property are a basic organizational feature of mammalian sensory cortex. Primary visual cortex (V1) is characterized by orientation modules—groups of cells that share a preferred stimulus orientation—which are organized into a highly ordered orientation map. Here we show that in ferrets in which retinal projections are routed into the auditory pathway, visually responsive neurons in 'rewired' primary auditory cortex are also organized into orientation modules. The orientation tuning of neurons within these modules is comparable to the tuning of cells in V1 but the orientation map is less orderly. Horizontal connections in rewired cortex are more patchy and periodic than connections in normal auditory cortex, but less so than connections in V1. These data show that afferent activity has a profound influence on diverse components of cortical circuitry, including thalamocortical and local intracortical connections, which are involved in the generation of orientation tuning, and long-range horizontal connections, which are important in creating an orientation map.

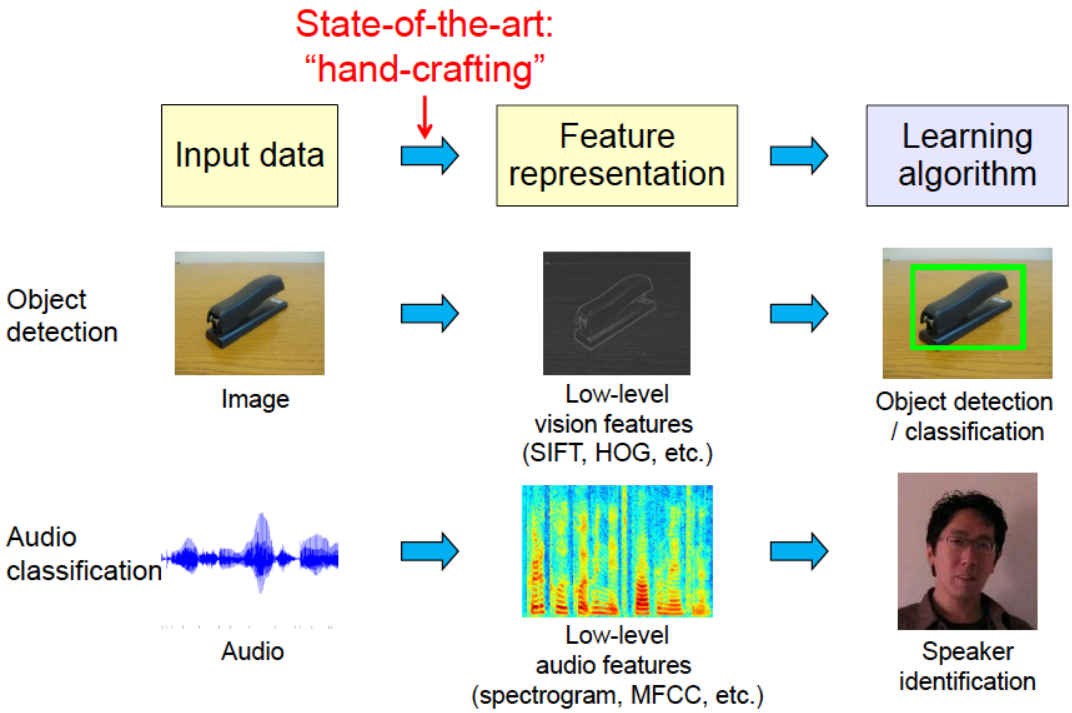


Hand-crafted Feature Representation



(Image credit: H. Lee)

Hand-crafted features in different applications



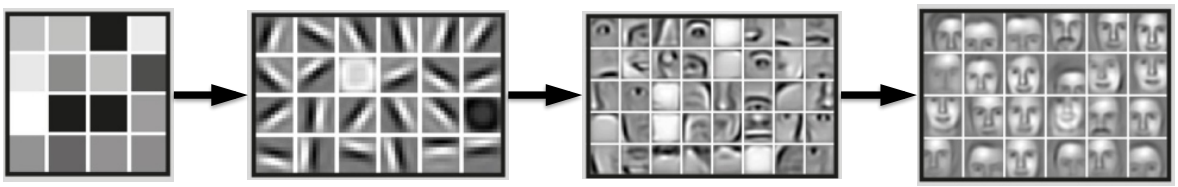
(Image credit: H. Lee)

[Limitations of Hand-crafted Features]

- Needs expert knowledge
- Requires time-consuming hand-tuning process
- Not equally helpful for target tasks
- (Possibly) other types of unknown features, which may be much more important

Let the computer learn the feature representations from data autonomously!!!

Deep-learning neural networks use layers of increasingly complex rules to categorize complicated shapes such as faces.



Layer 1: The computer identifies pixels of light and dark.

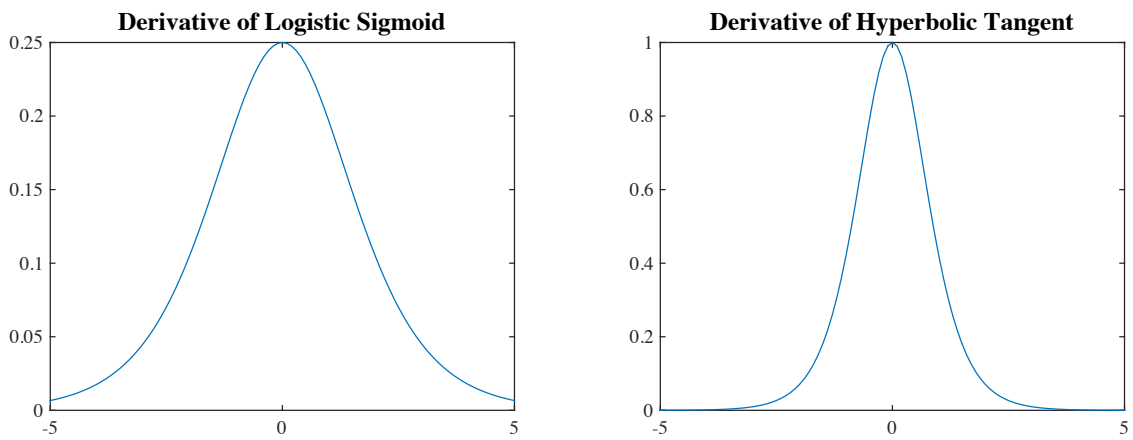
Layer 2: The computer learns to identify edges and simple shapes.

Layer 3: The computer learns to identify more complex shapes and objects.

Layer 4: The computer learns which shapes and objects can be used to define a human face.

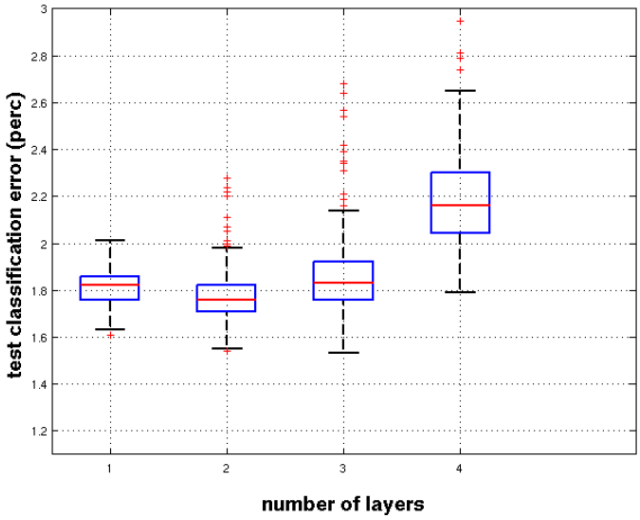
(Modified, N. Jones "The Learning Machines," Nature, 2014)

Difficulties in Deep Learning



Difficulties in Deep Learning

- Empirical Results: Backprop on Deep Neural Networks
  - Digit classification on MNIST
  - 400 different initialization seeds
  - Increasing depth seems to increase the probability of finding poor local minima

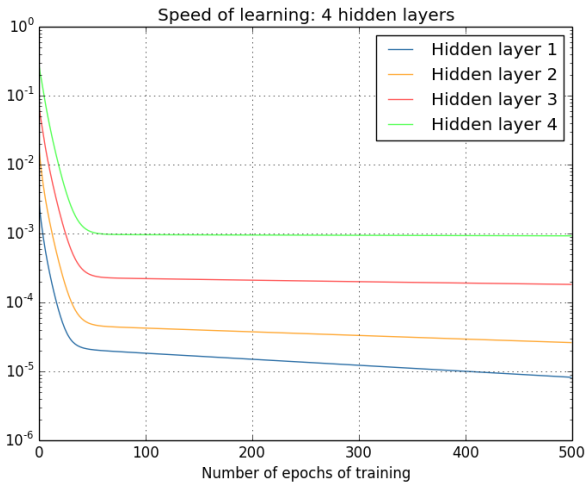


(Erhan *et al.*, AISTATS 2009)

[Vanishing Gradient Problem]

- Backpropagation becomes ineffective due to vanishing gradients after repeated multiplication.
- The gradient tends to get smaller as we propagate backward through the hidden layers.
- Neurons in the earlier layers learn much more slowly then neurons in later laters.

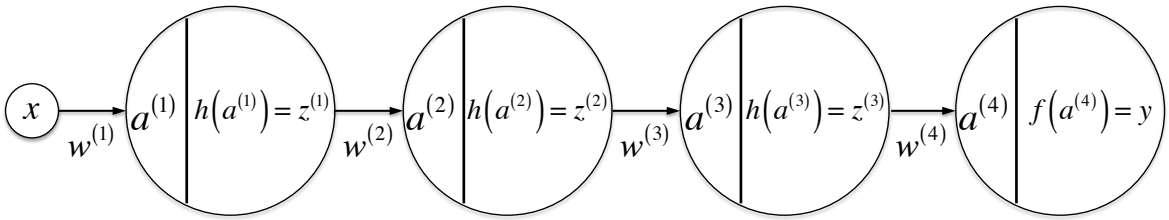
- We can think of the gradient $\delta^{(l)}$ as a vector whose entries determine how quickly the l-th hidden layer learns.
- Use the magnitude of the vector $\|\delta^{(l)}\|$ as (roughly) global measure of the speed at which the l-th layer is learning.



MNIST dataset, (784-30-30-30-30-10) network

(M. Nielson, *Neural Networks and Deep Learning*, 2015)

[Cause of Vanishing Gradient Problem]



(Recep.) $$\frac{\partial E_n}{\partial w_{ji}} = \underbrace{\frac{\partial E_n}{\partial a_j}}_{\equiv \delta_j} \underbrace{\frac{\partial a_j}{\partial w_{ji}}}_{z_i} = \left\{ h'(a_j) \sum_k w_{kj} \delta_k \right\} z_i$$

$$\frac{\partial E_n}{\partial w^{(1)}} = \overbrace{x\, h'\left(a^{(1)}\right) w^{(2)} \underbrace{h'\left(a^{(2)}\right) w^{(3)} \overbrace{h'\left(a^{(3)}\right) w^{(4)} \underbrace{f'\left(a^{(4)}\right)(y_n - t_n)}_{\delta^{(4)} = \frac{\partial E_n}{\partial a^{(4)}}}}^{\delta^{(3)}}}_{\delta^{(2)}}}^{\delta^{(1)}}$$

In a general feed-forward neural network: multiple units in each layer

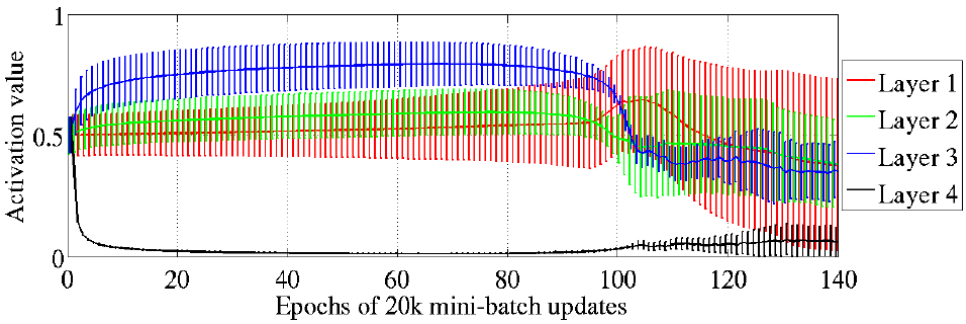
$$\boldsymbol{\delta}^{(l)} = H'\left(\mathbf{a}^{(l)}\right)\left(\mathbf{W}^{(l+1)}\right)^{\top} H'\left(\mathbf{a}^{(l-1)}\right)\left(\mathbf{W}^{(l+2)}\right)^{\top} \cdots H'\left(\mathbf{a}^{(L)}\right)\frac{\partial E_n}{\partial \mathbf{a}^{(L)}}$$

where $H'\left(\mathbf{a}^{(l)}\right) = \begin{bmatrix} h'\left(a_1^{(l)}\right) & & & & 0 \\ & \ddots & & & \\ & & h'\left(a_j^{(l)}\right) & & \\ & & & \ddots & \\ 0 & & & & h'\left(a_M^{(l)}\right) \end{bmatrix}$

- In general, the weights are initialized by using a Gaussian with mean 0 and standard deviation 1.

$$\left|w_j^{(l)}\right| < 1$$

- Logistic sigmoid activation function
  - Cause the activations in the final hidden layer to saturate near 0 early in training, substantially slowing down learning



(Glorot and Bengio, AISTATS 2010)

  - Suggested some alternative activation functions (*e.g.*, 'tanh' and 'softsign') with ways to initialize weights

Breakthrough by Hinton et al.

- Greedy layer-wise pre-training (Neural Computation, 2006)

LETTER Communicated by Yann Le Cun

A Fast Learning Algorithm for Deep Belief Nets

Geoffrey E. Hinton
hinton@cs.toronto.edu
Simon Osindero
osindero@cs.toronto.edu
Department of Computer Science, University of Toronto, Toronto, Canada M5S 3G4

Yee-Whye Teh
tehyw@comp.nus.edu.sg
Department of Computer Science, National University of Singapore, Singapore 117543

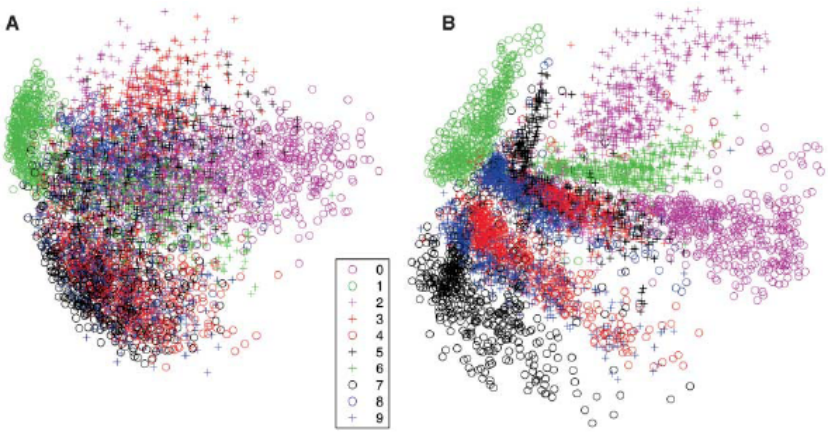
Reducing the Dimensionality of Data with Neural Networks

Science AAAS

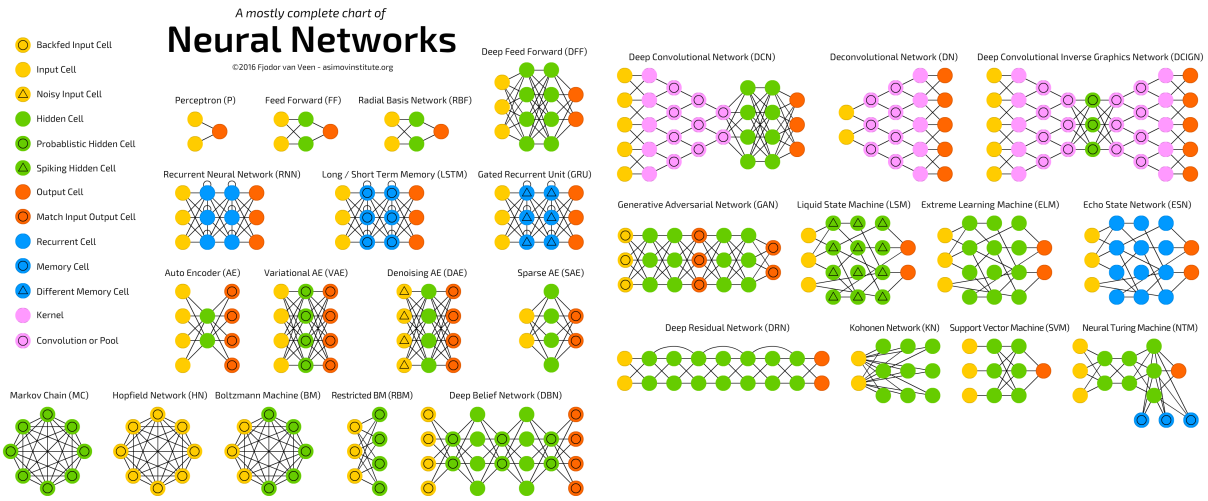
G. E. Hinton* and R. R. Salakhutdinov

High-dimensional data can be converted to low-dimensional codes by training a multilayer neural network with a small central layer to reconstruct high-dimensional input vectors. Gradient descent can be used for fine-tuning the weights in such "autoencoder" networks, but this works well only if the initial weights are close to a good solution. We describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes that work much better than principal components analysis as a tool to reduce the dimensionality of data.

Fig. 3. **(A)** The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. **(B)** The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (*8*).



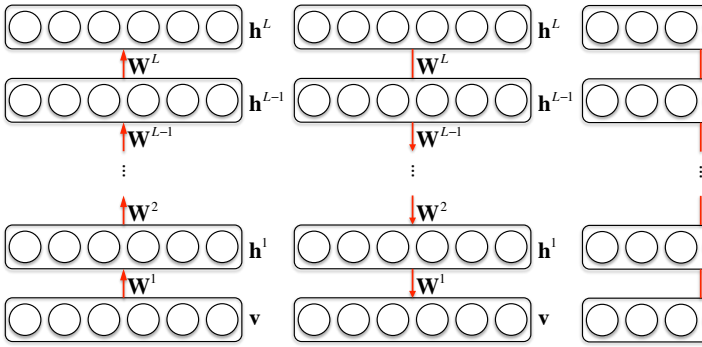
Deep Models

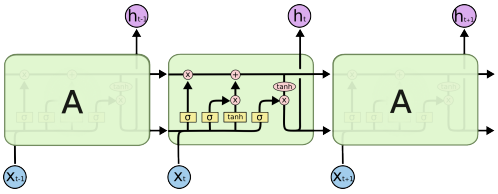


Source: http://www.asimovinstitute.org/neural-network-zoo/

Deep Models

- Stacked Auto-Encoder (SAE)
- Deep Belief Network (DBN)
- Deep Boltzmann Machine (DBM)
- Convolutional NN (CNN)
- Recurrent NN (RNN)

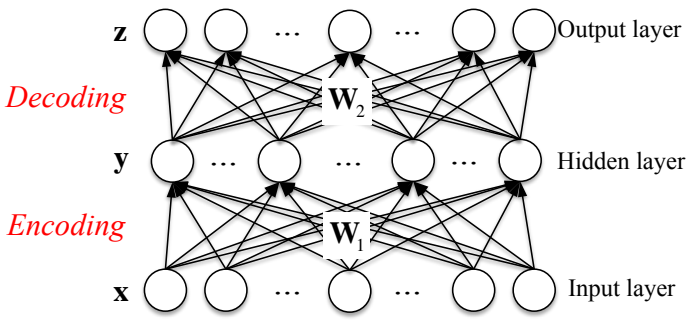




Stacked Auto-Encoder

Auto-Encoder

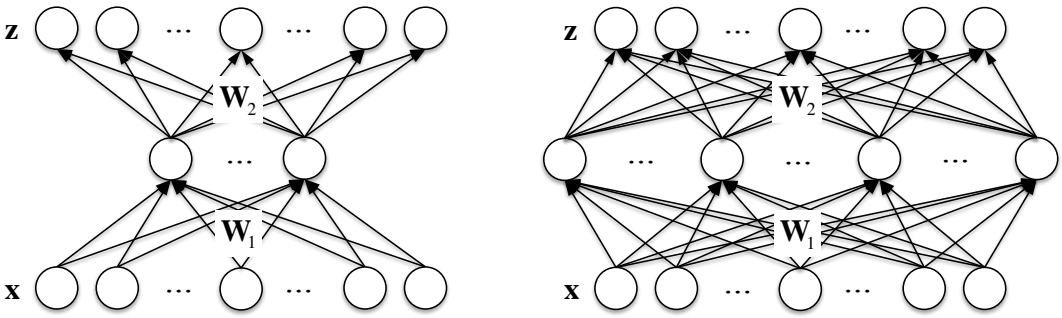
Learning a latent/compressed representation of an input $\mathbf{x}$



$$\mathbf{z} = \mathbf{W}_2\mathbf{x} + \mathbf{b}_2 \approx \mathbf{x}$$

$$\mathbf{y} = f(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$

- $\mathbf{W}_1 \in \mathbb{R}^{D_H \times D_I}$, $\mathbf{W}_2 \in \mathbb{R}^{D_I \times D_H}$: weight matrices
- $\mathbf{b}_1 \in \mathbb{R}^{D_H}$, $\mathbf{b}_2 \in \mathbb{R}^{D_I}$: bias vectors
- D_H and D_I: number of hidden and input units
- $f(a) = 1/\left(1 + exp(-a)\right)$: sigmoid function



- Any number of hidden units, even larger than the input dimension (sparsity constraint on the hidden units; sparse auto-encoder)

$$E(\mathbf{X}, \mathbf{Z}) + \gamma \sum_{j}^{D_H} \mathrm{KL}(\rho || \hat{\rho}_j)$$

$$E(\mathbf{X}, \mathbf{Z}) = \frac{1}{2} \sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{z}_i\|_2^2$$

$$\mathrm{KL}(\rho || \hat{\rho}_j) = \rho log \frac{\rho}{\hat{\rho}_j} + (1-\rho) log \frac{1-\rho}{1-\hat{\rho}_j}$$

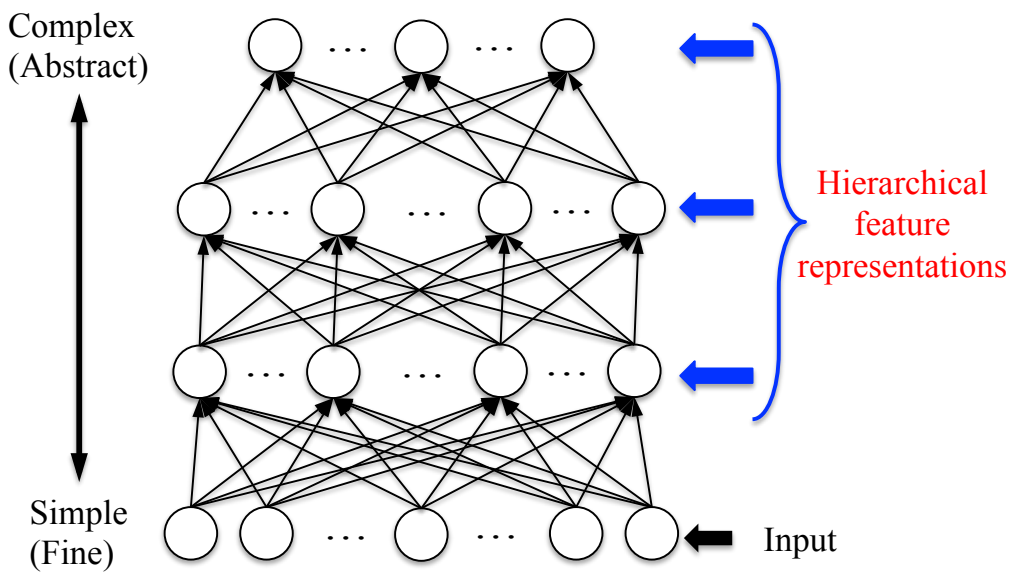
Stacked Auto-Encoder

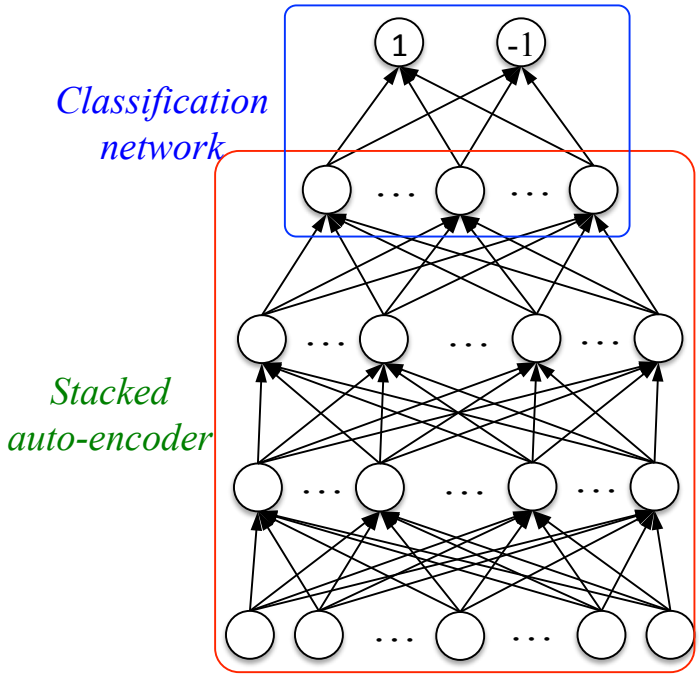
- Auto-encoder: a building block for feature representation in neuroimaging or biological data
- Discovering patterns such as nonlinear relations among features
- Any number of hidden units, even larger than the input dimension (sparsity constraint on the hidden units; sparse auto-encoder)

$$E(\mathbf{X}, \mathbf{Z}) + \gamma \sum_{j}^{D_H} \mathsf{KL}(\rho || \hat{\rho}_j)$$

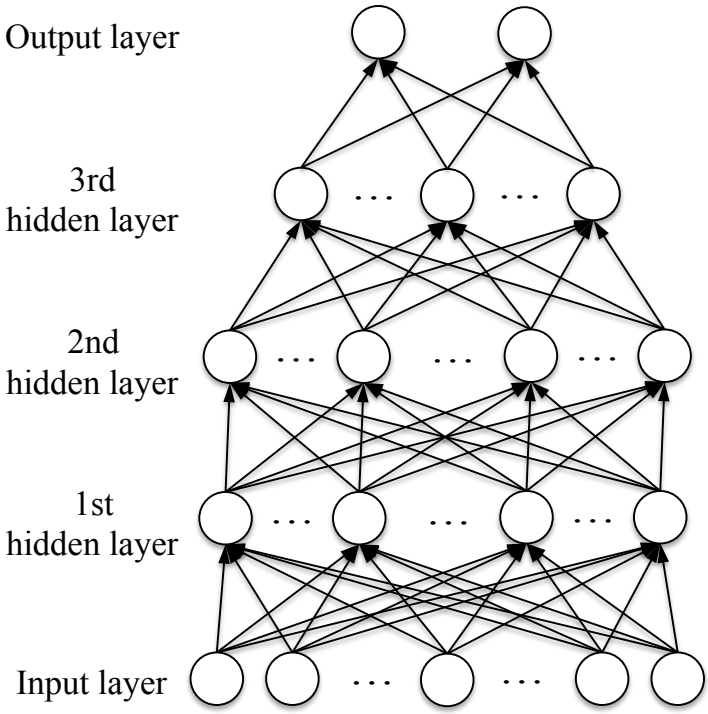
$$E(\mathbf{X}, \mathbf{Z}) = \frac{1}{2} \sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{z}_i\|_2^2$$

$$\mathsf{KL}(\rho || \hat{\rho}_j) = \rho log \frac{\rho}{\hat{\rho}_j} + (1-\rho) log \frac{1-\rho}{1-\hat{\rho}_j}$$

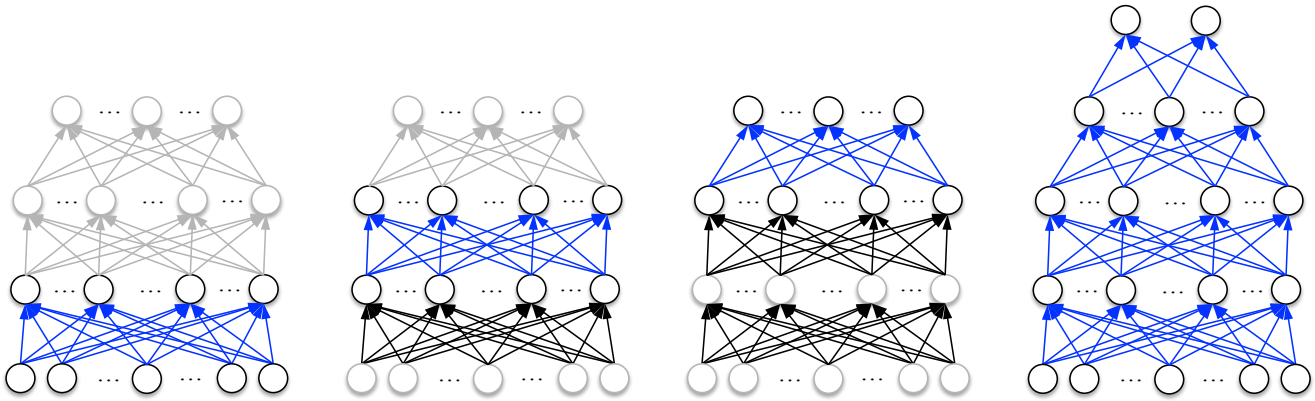




Greedy Layer-Wise Pre-Training

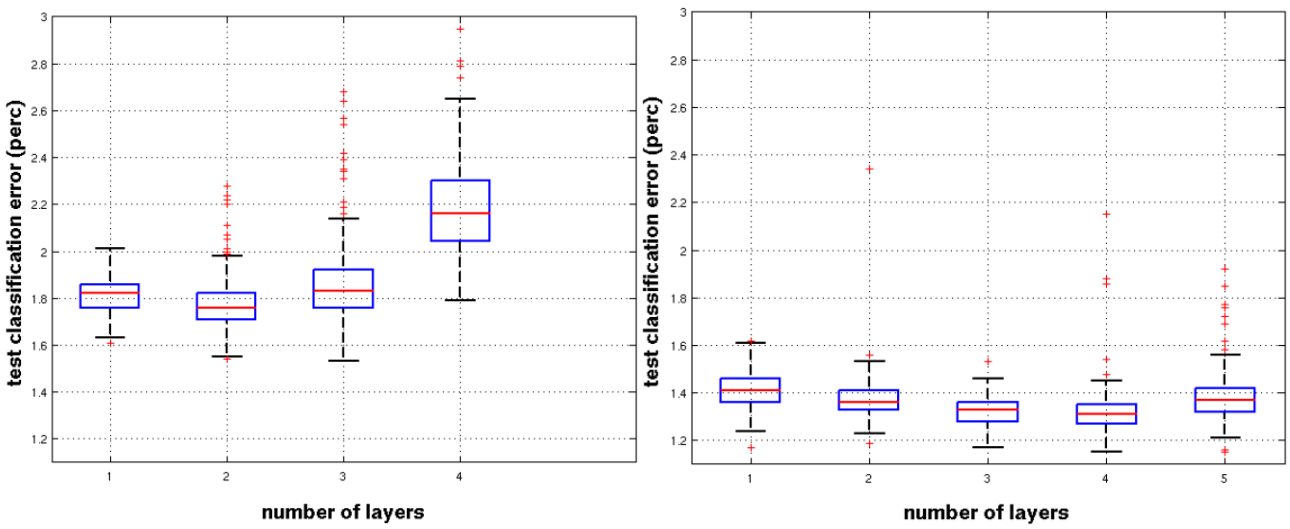


- Train layer 1 in an unsupervised manner
- Train layer 2 while keeping layer 1 fixed in an unsupervised manner
- Train layer 3 while keeping layer 1 & 2 fixed in an unsupervised manner
- Fine-tune the whole network in a supervised manner



(Pre-train 1st hidden layer) (Pre-train 2nd hidden layer) (Pre-train 3rd hidden layer) (Fine-tuning whole network)

Effects of Pre-Training: Empirical Results



(Erhan *et al.*, AISTATS 2009)

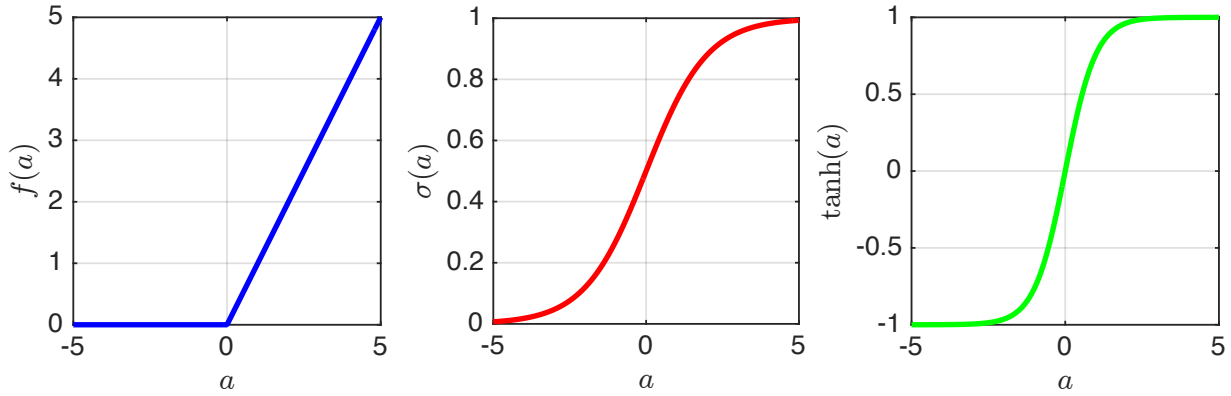
Algorithmic Advances for Better Learning

Algorithmic Advances for Better Learning

- Greedy layer-wise pretraining [Hinton and Salakhutdinov, 2006]
- Rectified Linear Unit (ReLU) [Nair and Hinton, 2010]
- Denoising [Vincent *et al.*, 2008]
- Dropout [Hinton *et al.*, 2012]
- Batch normalization [Ioffe and Szegedy, 2015]
- Variational Auto-Encoder [Kingma and Welling, 2014]
- Generative Adversarial Networks [Goodfellow *et al.*, 2016]

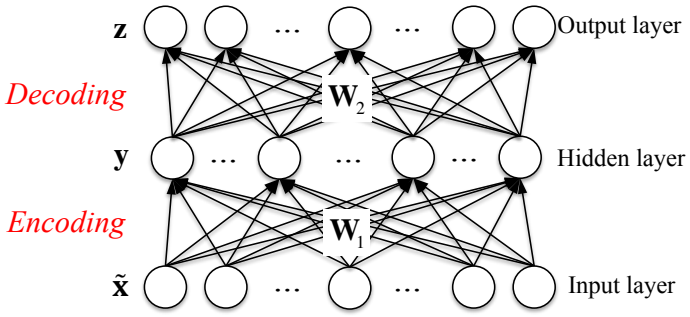
ReLU [Glorot et al., 2011]

- Rectified linear function: $f(a) = \max(0, a)$
- Logistic sigmoid: $\sigma(a) = \frac{1}{1+e^{-a}}$
- Hyperbolic tangent: $\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$



- Non-differentiable at $a = 0$
  - Highly unlikely that it will be at exactly $a = 0$ at any time
  - In practice, set either 0 or 1
- Unbounded on the positive side
  - *Regularization* to limit the magnitude of weights
  - ReLU+ℓ_2-regularization without pre-training: slightly superior or similar performance to pre-training [Glorot *et al.*, 2011]

Denoising Auto-Encoder [Vincent et al., 2008]



$$\mathbf{z} = \mathbf{W}_2\mathbf{x} + \mathbf{b}_2 \approx \mathbf{x}$$

$$\mathbf{y} = f(\mathbf{W}_1\tilde{\mathbf{x}} + \mathbf{b}_1)$$

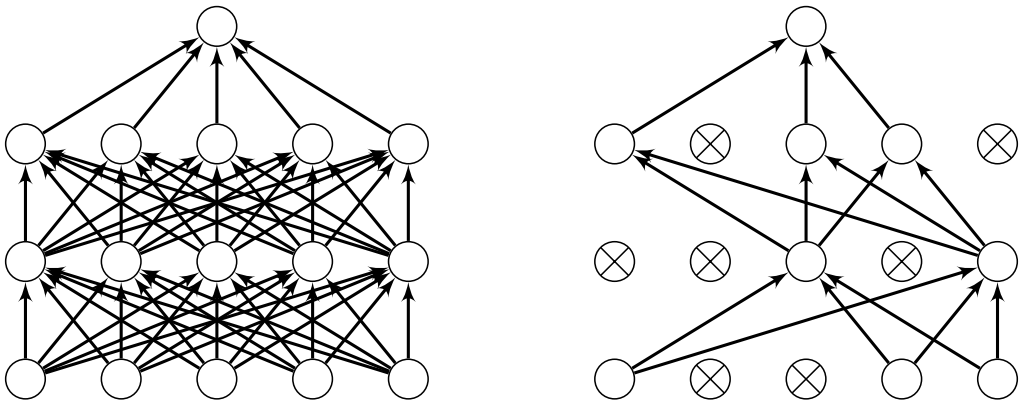
- In order to force the hidden layer to discover more robust features and prevent it from simply learning the identity, train the auto-encoder to reconstruct the input from a corrupted version of it.
- $\tilde{\mathbf{x}}$: corrupted input $\mathbf{x}$ by adding noise
- Train parameters so that the output $\mathbf{z}$ to be equal to $\mathbf{x}$

Bayesian interpretation of denoising

S. Mohamed, "A Statistical View of Deep Learning," TR, 2015

Dropout [Hinton et al., 2012]

- Preventing co-adaptation of neurons: a neuron cannot rely on the presence of particular other neurons [Krizhevsky *et al.*, 2012]
- Randomly deactivate a set (*e.g.*, 50%) of the neurons in a network on each training iteration



(a) Standard DNN (b) After applying dropout

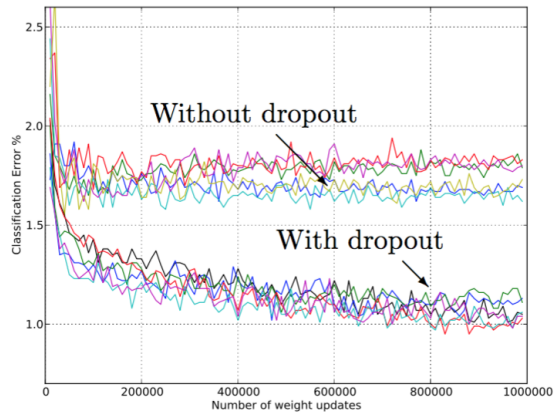
- In training: for *each training sample* in a mini-batch

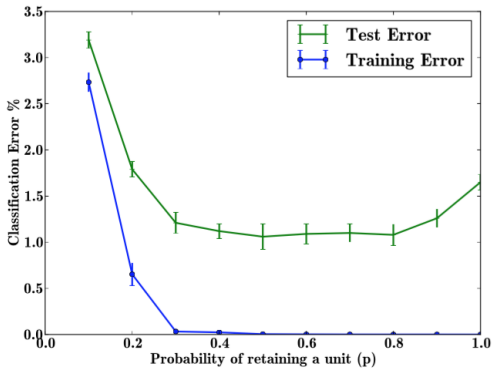
$$
\begin{aligned}
r_j^{(l)} &\sim \text{Bernoulli}\,(p) \\
\tilde{\mathbf{z}}^{(l)} &= \mathbf{r}^{(l)} \odot \mathbf{z}^{(l)} \\
a_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \tilde{\mathbf{z}}^{(l)} + b_i^{(l+1)} \\
z_i^{(l+1)} &= f\left(a_i^{(l+1)}\right)
\end{aligned}
$$

- At test time: scaling the weights
  - Infeasible to explicitly average the predictions from exponentially many thinned models
  - Approximation: to use a single neural net at test time without dropout with the weights scaled-down

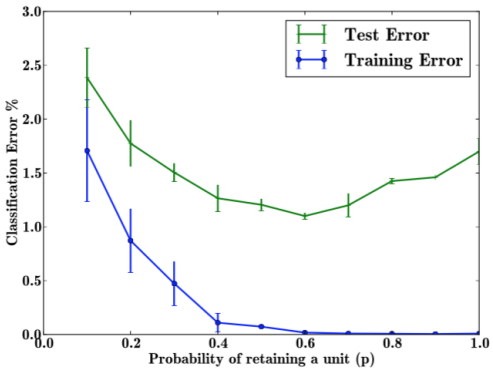
$$\mathbf{W}_{\text{test}}^{(l)} = p\mathbf{W}^{(l)}$$

- Experiments on MNIST dataset
  - Networks having 2 to 4 hidden layers each with 1,024 to 2,048 units (→)
  - Effector of dropout rate p (↓)
    - **(a)** 784-2048-2048-2048-10
    - **(b)** n: number of hidden units





(a) Keeping n fixed.



(b) Keeping pn fixed.

Bayesian interpretation of dropout

S. Mohamed, "A Statistical View of Deep Learning," TR, 2015

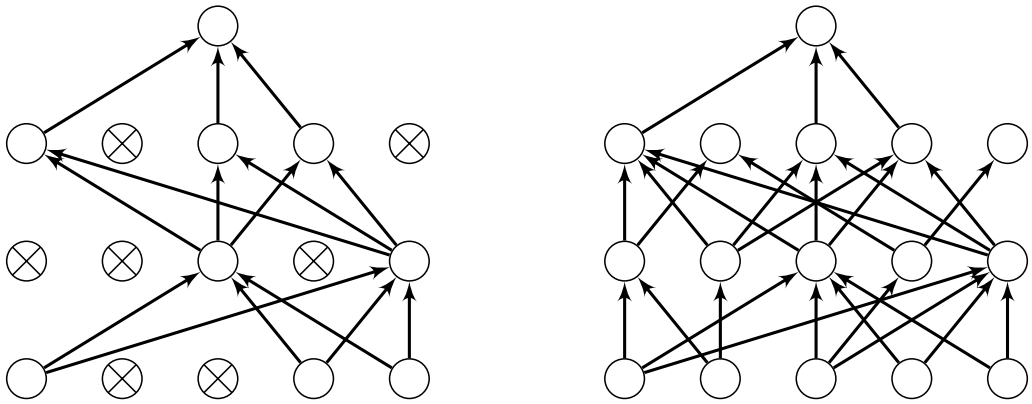
Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," arXiv, 2015

- With n units, a collection of 2^n possible thinned neural networks
- Training a neural network with dropout can be seen as training a collection of 2^ns thinned networks with extensive weight sharing

Dropout can be seen as a way of doing an equally-weighted averaging of exponentially many models with shared weights.

Dropconnect [Wan et al., 2013]

- Randomly remove a set (*e.g.*, 50%) of the weights and biases within the network



(a) After applying dropout (b) After applying dropconnect

- In training: for *each training sample* in a mini-batch

$$
\begin{aligned}
R_{ij}^{(l+1)} &\sim \text{Bernoulli}\,(p) \\
\tilde{\mathbf{W}}^{(l+1)} &= \mathbf{R}^{(l+1)} \odot \left[\mathbf{W}^{(l+1)}\ \mathbf{b}^{(l+1)}\right] \\
\mathbf{a}^{(l+1)} &= \tilde{\mathbf{W}}^{(l+1)} \left[\mathbf{z}^{(l)}; 1\right] \\
\mathbf{z}^{(l+1)} &= f\left(\mathbf{a}^{(l+1)}\right)
\end{aligned}
$$

- At test time: by means of Gaussian sampling
  - Weighted sum of Bernoulli variables $R_{ij}^{(l+1)}$

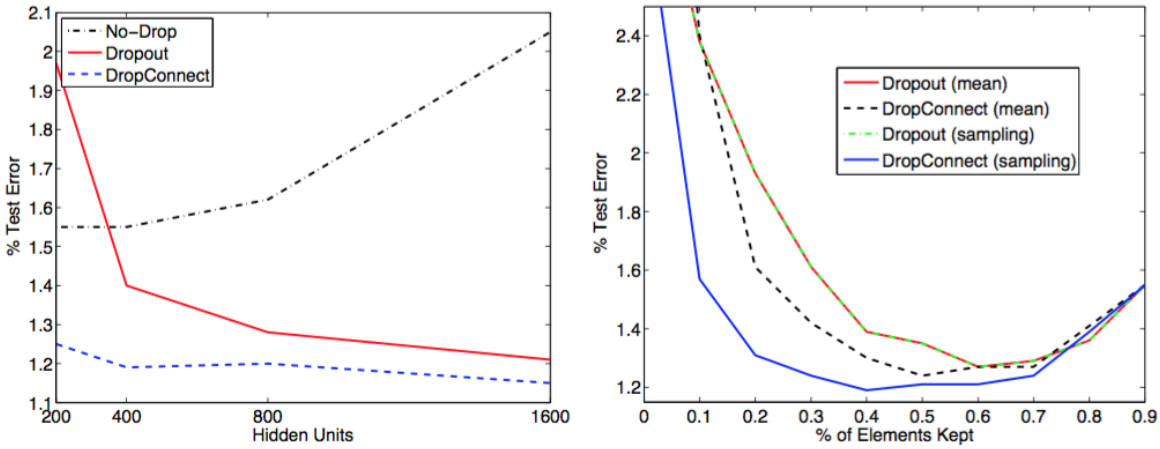
$$a_i^{(l+1)} = \sum_j \left(W_{ij}^{(l+1)} z_j^{(l)} + b_i^{(l+1)}\right) R_{ij}^{(l+1)}$$

  - Approximating preactivations by a Gaussian via moment matching

$$a_i^{(l+1)} \sim \mathcal{N}\left(p\mathbf{W}_{i\rightarrow}^{(l+1)}\mathbf{z}^{(l)}, p\,(1-p)\left(\mathbf{W}_{i\rightarrow}^{(l+1)}\mathbf{z}^{(l)} \odot \mathbf{W}_{i\rightarrow}^{(l+1)}\mathbf{z}^{(l)}\right)\left(\mathbf{z}^{(l)} \odot \mathbf{z}^{(l)}\right)\right)$$

$$z_i^{(l+1)} = f\left(a_i^{(l+1)}\right)$$

Experiment with MNIST dataset using 2-layer fully connected neural network



Batch Normalization [Ioffe and Szegedy, 2015]

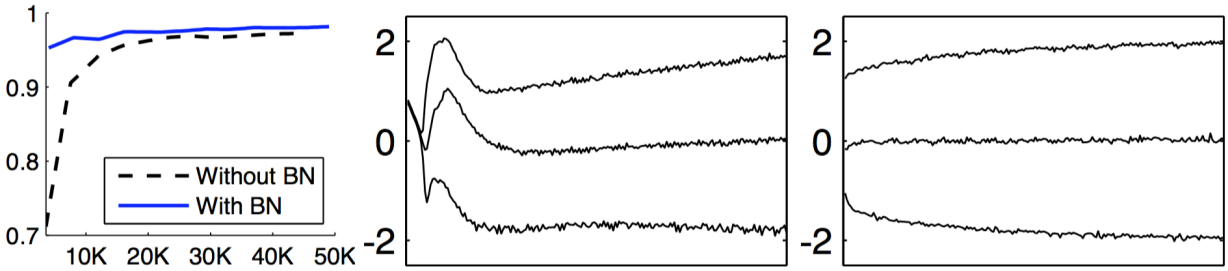
- Internal covariate shift: change in the distribution of network activations due to the change in network parameters during training makes the training time slow
- Performing normalization for each mini-batch and backpropagating the gradients through the normalization paramerters (*i.e.*, scale and shift)
- For each unit in a layer l, their value is normalized

$$\hat{x}_k^{(l)} = \frac{x_k^{(l)} - \mathbb{E}[x_k^{(l)}]}{\sqrt{\mathrm{Var}[x_k^{(l)}]}}$$

  where $k = 1, \ldots, F^{(l)}$ and $F^{(l)}$: number of units in the layer l.
- A pair of learnable parameters $\gamma_k^{(l)}$ and $\beta_k^{(l)}$ are then introduced to scale and shift the normalized values to restore the representation power of the network

$$y_k^{(l)} = \gamma_k^{(l)} \hat{x}_k^{(l)} + \beta_k^{(l)}.$$

- Reduce the training time and the need for dropout



(Left) Test accuracy of the MNIST network trained with and without Batch Normalization, vs. the number of training steps

(MR) The evolution of input distributions to a typical sigmoid, over the course of training, shown as $\{15, 50, 85\}$th percentiles.

Open Software for DL

- Caffe (https://github.com/BVLC/caffe)
- Theano (http://deeplearning.net/software/theano)
- TensorFlow (https://www.tensorflow.org)
- Torch (http://torch.ch/)
- Pylearn2 (http://deeplearning.net/software/pylearn2/)
- MatConvNet (http://www.vlfeat.org/matconvnet/)
- etc.

CHAPTER

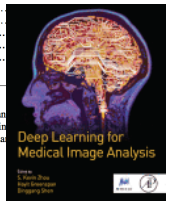
An Introduction to Neural Networks and Deep Learning

1

Heung-Il Suk
Korea University, Seoul, Republic of Korea

CHAPTER OUTLINE

1.1 Introduction ... 3
1.2 Feed-Forward Neural Networks ... 4
1.2.1 Perceptron ... 4
1.2.2 Multi-Layer Neural Network ... 5
1.2.3 Learning in Feed-Forward Neural Networks ... 6
1.3 Convolutional Neural Networks ... 8
1.3.1 Convolution and Pooling Layer ... 8
1.3.2 Computing Gradients ... 9
1.4 Deep Models ... 11
1.4.1 Vanishing Gradient Problem ... 11
1.4.2 Deep Neural Networks ... 12
1.4.2.1 Auto-Encoder ... 12
1.4.2.2 Stacked Auto-Encoder ... 13
1.4.3 Deep Generative Models ... 14
1.4.3.1 Restricted Boltzmann Machine ... 15
1.4.3.2 Deep Belief Network ... 17
1.4.3.3 Deep Boltzmann Machine ... 18
1.5 Tricks for Better Learning ... 20
1.5.1 Rectified Linear Unit (ReLU) ... 20
1.5.2 Dropout ...
1.5.3 Batch Normalization ...
1.6 Open-Source Tools for Deep Learning ...
References ...
Notes ...



1.1 INTRODUCTION

A brain or biological neural network is considered as the most well-organ
tem that processes information from different senses such as sight, hearin
taste, and smell in an efficient and intelligent manner. One of the key mecha

Deep Learning for Medical Image Analysis. DOI: 10.1016/B978-0-12-810408-8.00002-X
Copyright © 2017 Elsevier Inc. All rights reserved.

BE19CH09-Shen ARI 1 March 2017 13:31

REVIEWS IN ADVANCE

Review in Advance first posted online on March 9, 2017. (Changes may still occur before final publication online and in print.)

Deep Learning in Medical Image Analysis

Dinggang Shen,[1,2] Guorong Wu,[1] and Heung-Il Suk[2]

[1]Department of Radiology, University of North Carolina, Chapel Hill, North Carolina 27599; email: dgshen@med.unc.edu

[2]Department of Brain and Cognitive Engineering, Korea University, Seoul 02841, Republic of Korea; email: hisuk@korea.ac.kr

Annu. Rev. Biomed. Eng. 2017.10. Downloaded from www.annualreviews.org
Access provided by Korea University on 03/27/17. For personal use only.

Annu. Rev. Biomed. Eng. 2017. 19:221–48

The *Annual Review of Biomedical Engineering* is online at bioeng.annualreviews.org

https://doi.org/10.1146/annurev-bioeng-071516-044442

Copyright © 2017 by Annual Reviews.
All rights reserved

Keywords

medical image analysis, deep learning, unsupervised feature learning

Abstract

This review covers computer-assisted analysis of images in the field of medical imaging. Recent advances in machine learning, especially with regard to deep learning, are helping to identify, classify, and quantify patterns in medical images. At the core of these advances is the ability to exploit hierarchical feature representations learned solely from data, instead of features designed by hand according to domain-specific knowledge. Deep learning is rapidly becoming the state of the art, leading to enhanced performance in various medical applications. We introduce the fundamentals of deep learning methods and review their successes in image registration, detection of anatomical and cellular structures, tissue segmentation, computer-aided disease diagnosis and prognosis, and so on. We conclude by discussing research issues and suggesting future directions for further improvement.

221

Changes may still occur before final publication online and in print

Machine Intelligence Lab