



Wenmin Wang

# Principles of Machine Learning

The Three Perspectives

 Springer

Principles of Machine Learning

## Part II Frameworks

- 3 Probabilistic Framework
- 4 Statistical Framework
- 5 Connectionist Framework
- 6 Symbolic Framework
- 7 Behavioral Framework

# 5 Connectionist Framework

# Contents

5.1 Overview

5.2 Connectionist Learning Theories

5.3 Units of Neural Networks

5.4 Structures of Neural Networks

5.5 Optimizations of Neural Networks

5.6 Agents of Neural Networks

## About the Connectionist Framework

- Connectionist framework is based on connectionist approaches and models. Its manifestation is known as artificial neural networks, used to simulate the brain's information processing and learning process.
- Artificial neural network was pioneered in 1940s, gradually attracting the attention of some researchers, and even becoming a theory and method in artificial intelligence and machine learning.
- Donald Hebb's connectionist view is known as Hebbian theory, an interpretation of the brain's learning process, and is therefore also known as Hebbian learning.
- The core of connectionism is the theory of connectionist learning, also known as the theory of connectionist learning.

# Connectionist Learning Theories

## History Related to Connectionist Learning Theories

| Year | Name                            | Pioneer(s) and Publication  |
|------|---------------------------------|---|
| 1949 | Hebbian Theory                  | Donald Hebb. <i>The Organization of Behaviour: A Neuropsychological Theory</i> , Jone Wiley & Sons, 1949.   |
| 1982 | Connectionist Models            | Jerome Feldman and Dana Ballard. “Connectionist Models and Their Properties”, <i>Cognitive Science</i> , 1982.  |
| 1986 | Parallel Distributed Processing | David Rumelhart, Geoffrey Hinton and James McClelland. “A general framework for parallel distributed processing.” <i>Parallel distributed processing: Explorations in the microstructure of cognition</i> , 1986. |
| 2007 | Neural Networks Theory          | Alexander I. Galushkin. <i>Neural networks theory</i> . Springer Berlin Heidelberg, 2007.   |

# Hebbian Theory

Hebbian theory was introduced by Donald Hebb in his book *The Organization of Behavior: A Neuropsychological Theory*, in 1949.

Hebb attempts to explain the adaptation of brain neurons during the learning process.

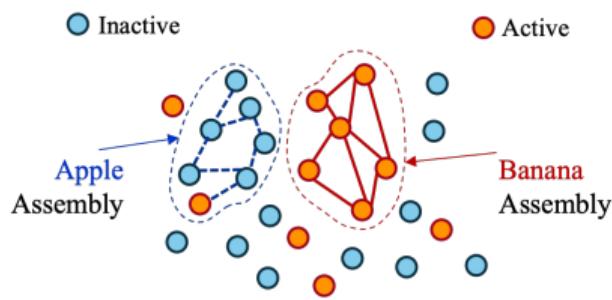
“When an axon of cell *A* is near enough to excite a cell *B* and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that *A*’s efficiency, as one of the cells firing *B*, is increased.”



Hebbian theory is also called Hebb’s rule, and cell-assembly theory.

# Hebb's Cell-Assembly Theory

"Any frequently repeated, particular stimulation will lead to the slow development of a 'cell-assembly,' a diffuse structure comprising cells in the cortex and diencephalon (and also, perhaps in the basal ganglia of the cerebrum), capable of acting briefly as a closed system, delivering facilitation to other such systems and usually having a specific motor facilitation."



"A series of such events constitutes a 'phase sequence' – the thought process. Each assembly action may be aroused by a preceding assembly, by a sensory event, or-normally-by both. The central facilitation from one of these activities on the next is the prototype of 'attention'."

## Connectionist Models

Connectionism attempts to explain mental phenomena using artificial neural networks (ANNs). Inspired by the brain, using neuron-like computing units, and synapse-like connections.

The core of connectionism is referred to as *connectionist models*, that was proposed by Jerome Feldman and Dana Ballard in a paper titled “Connectionist Models and Their Properties” in 1982.

The Models are designed as an information processing models (IPM).

They also discussed the stability and noise sensitivity of the model, distributed decision-making, issues of time and sequence, representation of complex concepts, and application in cognitive science.

# Parallel Distributed Processing

Parallel Distributed Processing (PDP) began in the 1970s and made significant progress in the 1980s.

PDP emphasizes the parallel processing nature of neural networks and the distributed nature of neurons.

It became popular in 1986, and 1987 with the release of the books:

*Parallel Distributed Processing: Explorations in the Microstructure of Cognition.*

*Volume 1: Foundations*

*Volume 2: Psychological and Biological Models.*

Those books are now considered seminal connectionist works, and it is now common to fully equate PDP and connectionism.

# Framework of Parallel Distributed Processing

In the article “A General Framework for Parallel Distributed Processing”, the *eight important attributes* of PDP was listed:

- (1) A set of processing units.
- (2) A state of activation.
- (3) An output function for each unit.
- (4) A pattern of connectivity among units.
- (5) A propagation rule for propagating patterns of activities through the network of connectivities.
- (6) An activation rule for combining the inputs impinging on a unit with the current state of that unit to produce a new level of activation for the unit.
- (7) A learning rule whereby patterns of connectivity are modified by experience.
- (8) An environment within which the system must operate.

# Neural Network Theory

The concept of neural networks can be traced back to a classic McCulloch-Pitts Neuron Model, proposed in 1943.

The perceptron, composed of single-layer neurons, is the earliest artificial neural network.

Based on the single-layer perceptron, a multilayer perceptron (MLP) appeared: it consists of an input layer, an output layer, and multiple hidden layers, also known as a multilayer artificial neural network.

In 1997, Philippe de Wilde published a book titled *Neural Network Models: Theory and Projects*, which deeply explained neural network models.

In 2007, Alexander Galushkin published a book titled *Neural Networks Theory*, elevating neural networks to a theoretical level.

## About the Units of Neural Networks

Biological neuron is the unit in biological neural networks.

Artificial neuron was proposed as the unit in artificial neural networks in 1943.

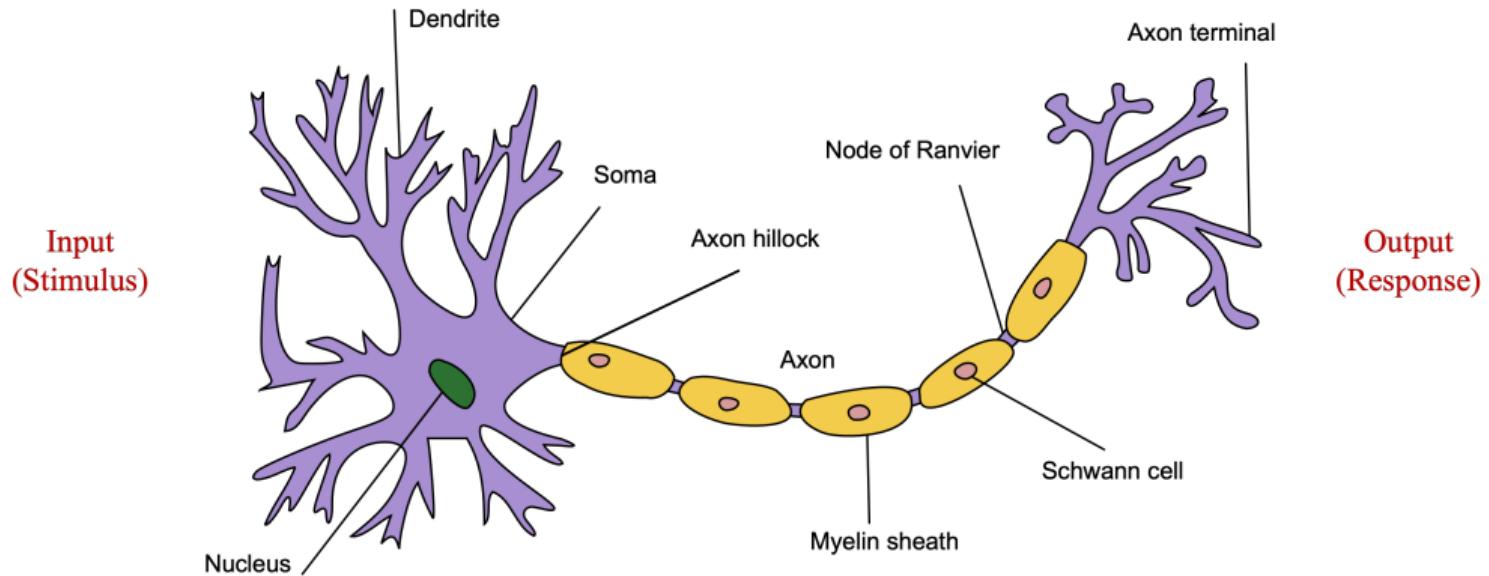
Spiking neuron was proposed in 1952.

Long short-term memory (LSTM) was proposed as the unit of recurrent neural networks in 1997.

Gated recurrent unit (GRU) was introduced in 2014.

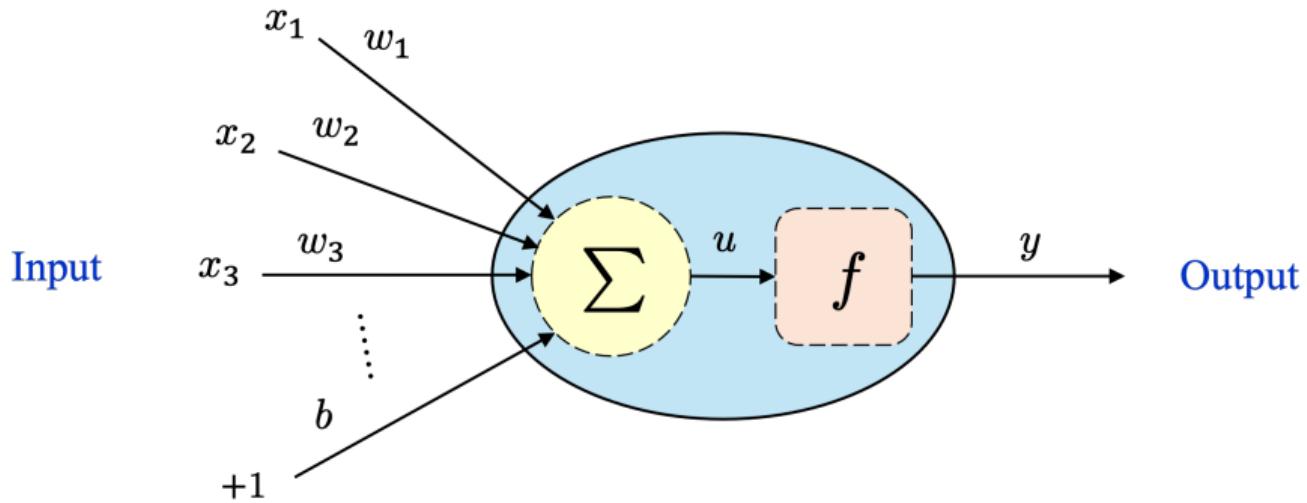
Capsule neuron was proposed as the unit of capsule neural network in 2017.

# Biological Neuron



<https://en.wikipedia.org/wiki/Neuron>

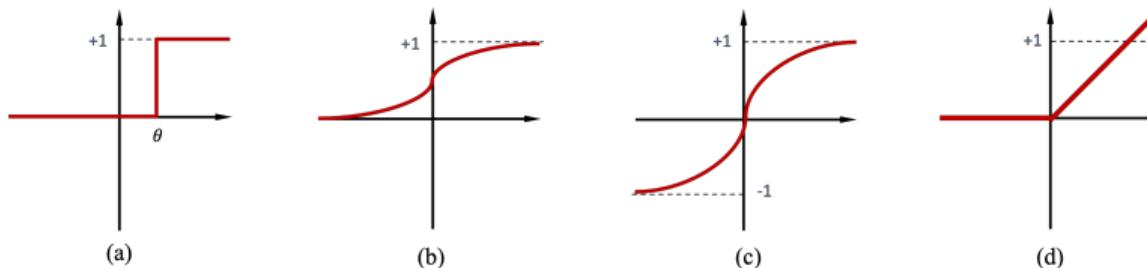
# Artificial Neuron



$$u = \sum_i w_i x_i + b, \quad \text{activation function} y = f(u).$$

# Activation Functions

| Name                             | Function   | Derivative   |
|----------------------------------|--|--|
| (a) Binary threshold             | $f(u) = \begin{cases} 0 & \text{if } u < \theta \\ 1 & \text{otherwise} \end{cases}$ | $\frac{d\sigma(u)}{du} = 0$  |
| (b) Sigmoid                      | $f(u) = \sigma(u) = \frac{1}{1+e^{-u}}$  | $\frac{d\sigma(u)}{du} = \sigma(u) \cdot (1 - \sigma(u))$                  |
| (c) Hyperbolic Tangent (TanH)    | $f(u) = \tanh(u) = \frac{(e^u - e^{-u})}{(e^u + e^{-u})}$                            | $\frac{d\tanh(u)}{du} = 1 - f(u)^2$  |
| (d) Rectified Linear Unit (ReLU) | $f(u) = \max(0, u) = \begin{cases} 0, & u \leq 0 \\ u, & u > 0 \end{cases}$          | $\frac{df(u)}{du} = \begin{cases} 0, & u \leq 0 \\ 1, & u > 0 \end{cases}$ |

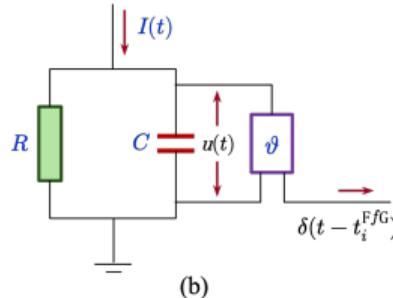
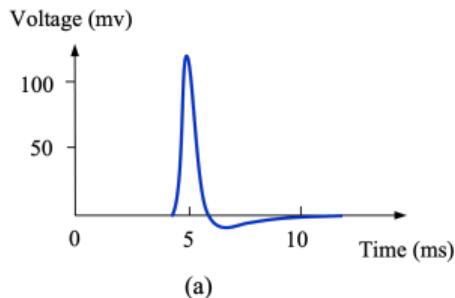


# Spiking Neuron

First spiking neuron was proposed in 1952, inspired by biological neuron. Based on it, spiking neural network (SNN) was presented.

In neuroscience, *spike* is used to represent the electrical pulse signal received by the dendrites of biological neurons, its waveform is extremely sharp.

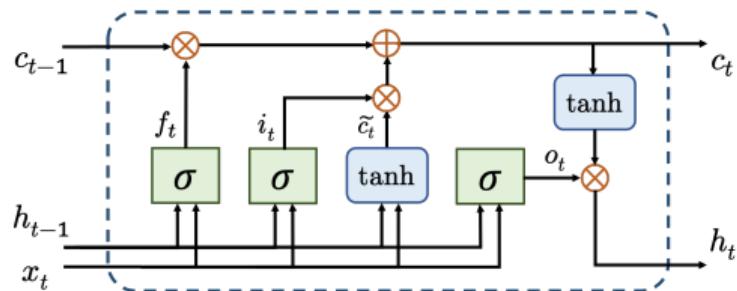
- It transmits information through a discrete spike train.
- Generating spikes increases/decreases with excitatory/inhibitory input.
- When internal state reaches a given threshold, spikes will be generated.



- (a) Action potential.  
(b) Leaky integrate-and-fire model.

# Long Short-Term Memory Unit

Long short-term memory (LSTM) unit was proposed in 1997, which can serve as a unit of a recurrent neural network (RNN).

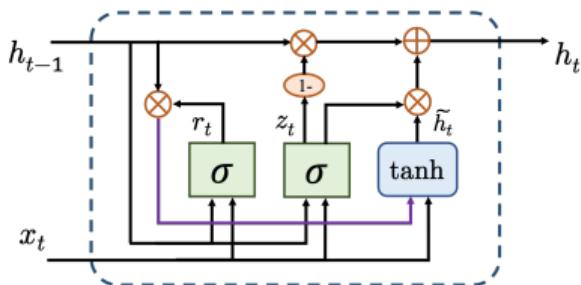


$$\begin{aligned} f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f), \\ i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i), \\ \tilde{c}_t &= \tanh(W_c x_t + U_c h_{t-1} + b_c), \\ c_t &= f_t * c_{t-1} + i_t * \tilde{c}_t, \\ o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o), \\ h_t &= o_t * \tanh(c_t). \end{aligned}$$

A neural network composed of LSTM units is called an LSTM network.

## Gated Recurrent Unit

Gated recurrent unit (GRU) is a variant of LSTM unit, proposed in 2014, can also serve as a unit of RNN.

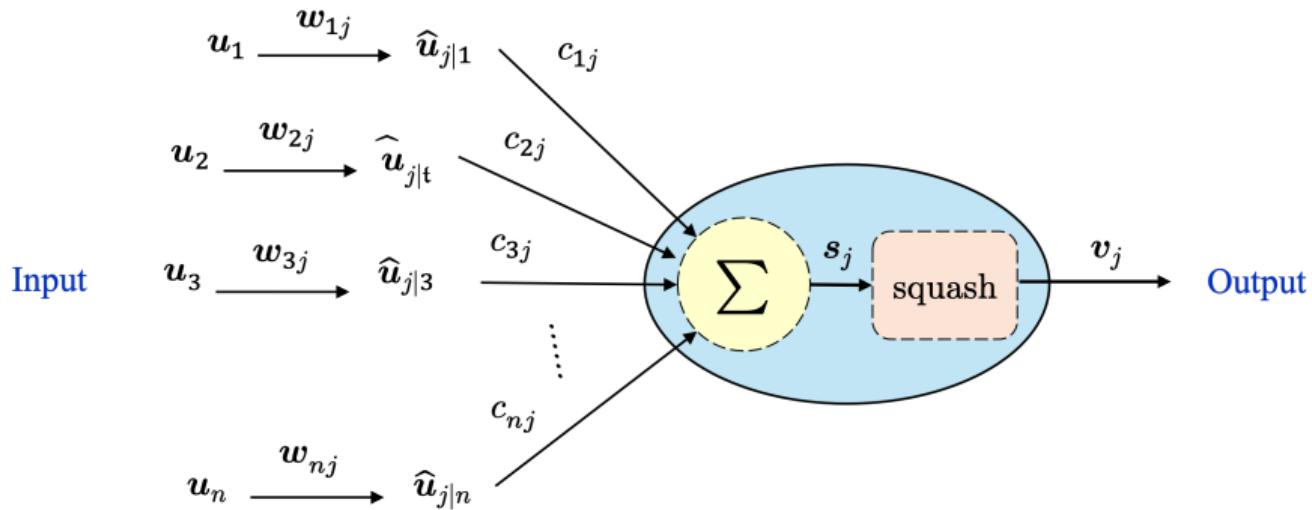


$$\begin{aligned} z_t &= \sigma(W_z \cdot [h_{t-1}, x_t]), \\ r_t &= \sigma(W_r \cdot [h_{t-1}, x_t]), \\ \tilde{h}_t &= \tanh(W \cdot [r_t * h_{t-1}, x_t]), \\ h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t. \end{aligned}$$

GRU is more concise and performs similarly to LSTM in some tasks, and shows better performance on some smaller datasets.

But some studies suggest that LSTM is more powerful than GRU.

# Capsule Unit



$$\hat{\mathbf{u}}_{j|i} = \mathbf{w}_{ij}\mathbf{u}_i. \quad \mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}. \quad \mathbf{v}_j = \frac{\|\mathbf{s}_j\|^2}{1 + \|\mathbf{s}_j\|^2} \frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}.$$

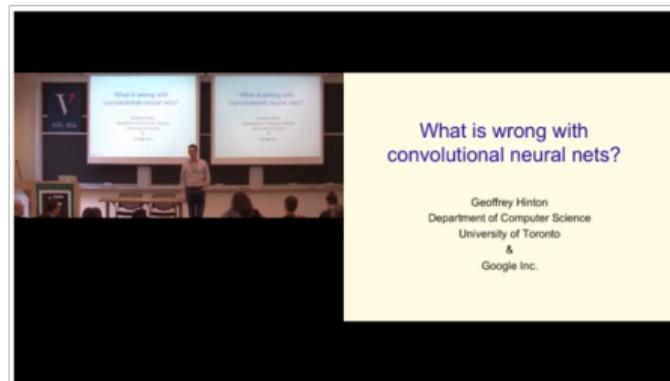
# Limitation with Convolutional Neural Nets (CNNs)

Geoffrey Hinton from the University of Toronto presented “What is wrong with convolutional neural nets?” on August 20, 2017.

CNNs have too few levels of structure:  
neurons, layers, and whole nets.

It needs to group the neurons in each  
layer into “capsules”.

Capsules can do a lot of internal  
computation and then output a compact  
result.

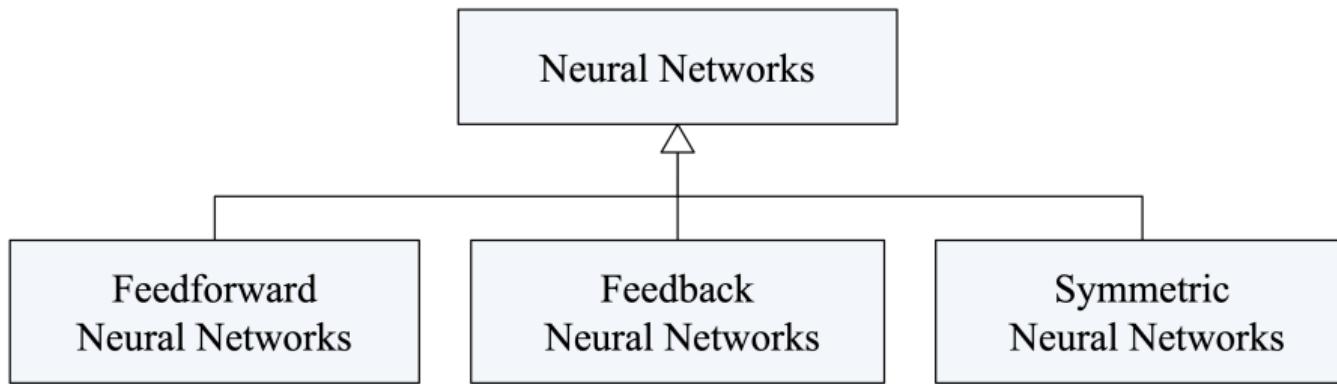


<https://www.youtube.com/watch?v=Jv1VDdl4vy4>

The network consisting capsules is called capsule neural network (CapsNet).

# About the Structures of Neural Networks

From the viewpoints on the structures of neural networks, which can be divided into three main types:



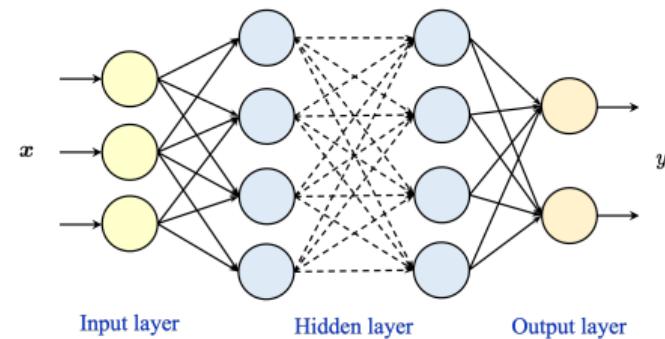
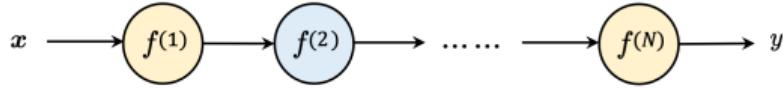
# Feedforward Neural Networks

In feedforward neural networks, the information moves in only one direction, forward from input layer, through hidden layers and to the output layer.

A feedforward model with  $N$  layers can be represented by

$$y = f^{(N)} \circ \dots \circ f^{(2)} \circ f^{(1)} (\mathbf{x}; \theta^{(1)}) .$$

where  $f^{(i)} (\mathbf{x}; \theta^{(i)})$  is the function of  $i$ th layer.

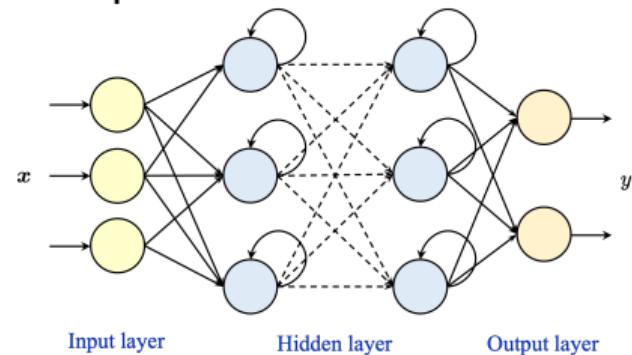
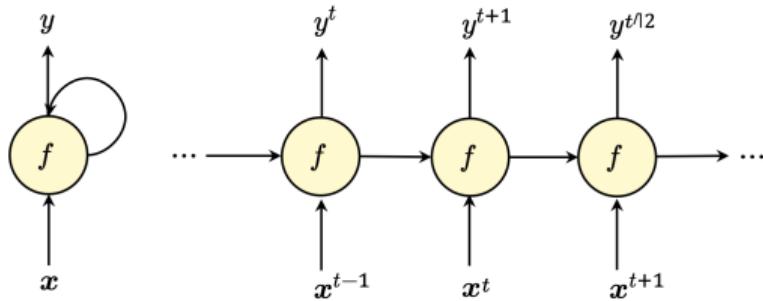


It's a multi-layer feedforward neural network if using artificial neural network to implement the multi-layer feedforward model.

# Recurrent Neural Networks

Recurrent neural networks (RNNs) also include backward connections between their neurons or layers, forming directed cycles.

RNNs can retain internal state to show dynamic temporal characteristics.



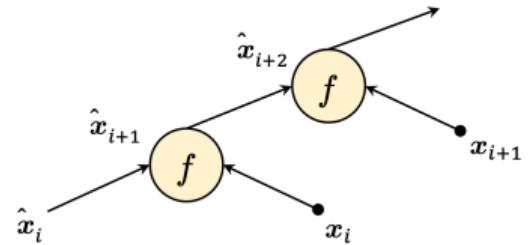
$$y^{(t+1)} = f(y^{(t)}, \mathbf{x}^{(t)}; \theta) = f(f(y^{(t-1)}, \mathbf{x}^{(t-1)}) , \mathbf{x}^{(t)}; \theta) ; \theta).$$

# Recursive Neural Networks

Recursive neural networks are a type of tree-like network.

Applying the same set of weights recursively and traverses the network in topological order.

$$\hat{x}_{i+2} = f(\hat{x}_{i+1}, x_{i+1}; \theta) = f(f(\hat{x}_i, x_i; \theta), x_{i+1}; \theta).$$



Recursive neural networks are different from recurrent neural networks:

- recursive neural networks are structurally recursive
- recurrent neural networks are cyclic in time.

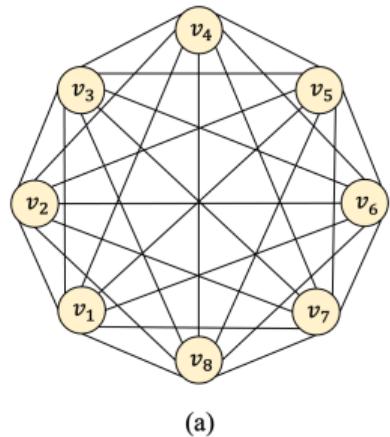
# Recurrent vs. Recursive Neural Networks

|                    | <i>Recurrent neural networks</i>                                    | <i>Recursive neural networks</i>             |
|--------------------|---|--|
| Special vs general | A special case of recursive neural networks                         | A generalization of recurrent neural network |
| Structure          | Chain-like neural network structure                                 | Tree-like neural network structure           |
| Features           | Recurring over time   | Structurally recursive                       |
| Applications       | Nature language processing, computer vision, and speech recognition | Nature language processing                   |

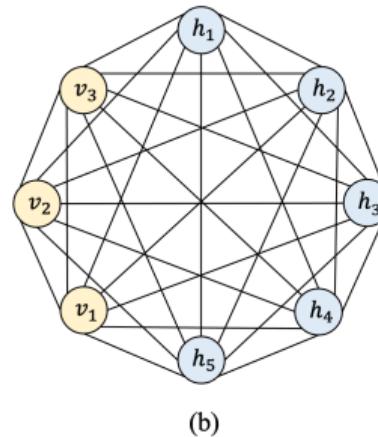
Recurrent neural networks include RNNs, LSTM networks, and GRU networks.

# Symmetric Neural Networks

In symmetric Neural Networks, the connections between units are symmetrical with the same weight in both directions.



(a)



(b)

Two types of symmetric neural networks: (a) Hopfield networks (without hidden units),  
(b) Boltzmann machines (with hidden units).

# Optimizations of Neural Networks

Multilayer artificial neural networks are composed of many artificial neurons connected in a certain way and layer, each with a weight between two layers of neurons.

The weight can be adjusted in some way, and then get the corresponding output value through the activation function, so that the neural network can adapt to the input and learn.

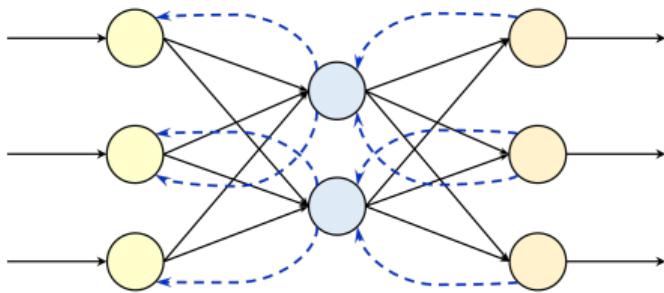
This is known as the *optimization* of neural networks, and is also often referred to as the *training* of neural networks.

# Backpropagation

Backpropagation is short for “backward propagation of errors”.

Be a common method for training multilayer neural networks and adjusting network weight errors.

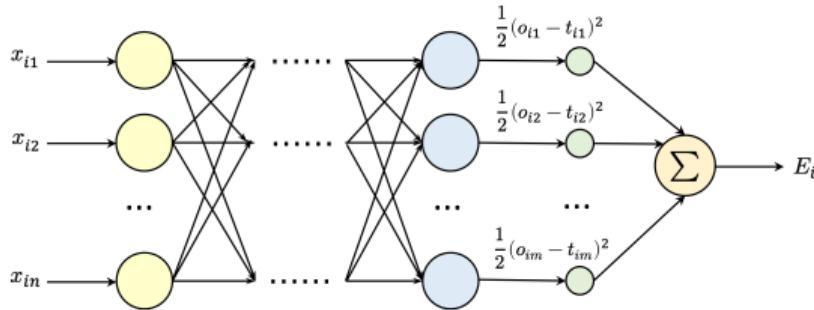
It usually combines with the gradient descent optimization method.



The blue dashed line is the path of backpropagation in the neural network.

# Backpropagation

Consider a neural network with  $n$  dimensional input and  $m$  dimensional output.



Given a training set composed of  $p$  ordered pairs:  $\{(x_1, t_1), (x_2, t_2), \dots, (x_p, t_p)\}$ , to minimize the error between actual output  $\mathbf{o}_i$  and target output  $\mathbf{t}_i$ ,

$$E = \frac{1}{2} \sum_{i=1}^p \|\mathbf{o}_i - \mathbf{t}_i\|^2 = \sum_{i=1}^p \left\{ \frac{1}{2} \sum_{j=1}^m (o_{ij} - t_{ij})^2 \right\}.$$

## Backpropagation

Because  $E$  is specifically calculated by the combination of expanding network node functions, it is a continuous differentiable function of the  $l$  weights  $W_1, W_2, \dots, W_l$  in the network. In this way, it is possible to minimize  $E$  by using the iterative process of gradient descent, for which it is necessary to calculate its gradient in the following way:

$$\nabla E = \left\{ \frac{\partial E}{\partial W_1}, \frac{\partial E}{\partial W_2}, \dots, \frac{\partial E}{\partial W_l} \right\}.$$

Each weight is updated using the following increment:

$$\Delta W_i = -\gamma \frac{\partial E}{\partial W_i} \quad \text{for } i = 1, \dots, l.$$

Where,  $\gamma$  represents the learning constant, which is a proportionality parameter. It defines the step length in the negative gradient direction for each iteration.

# Evolutionary Methods

Another approach to neural network optimization, mainly including:

- **Evolutionary Algorithms:**

are population-based general metaheuristic optimization algorithms, inspired by biological evolution mechanisms such as reproduction, mutation, recombination, and selection. There are several types of evolutionary algorithms, including genetic algorithms, genetic programming, evolutionary programming, gene expression programming, differential evolution, and evolution strategy. Some researchers have tried to use those algorithms for the optimization of ANNs.

- **Neuroevolution:**

is a method that combines evolutionary algorithms with ANNs, that is, using evolutionary algorithms to generate ANNs, including their parameters, topology, and rules.

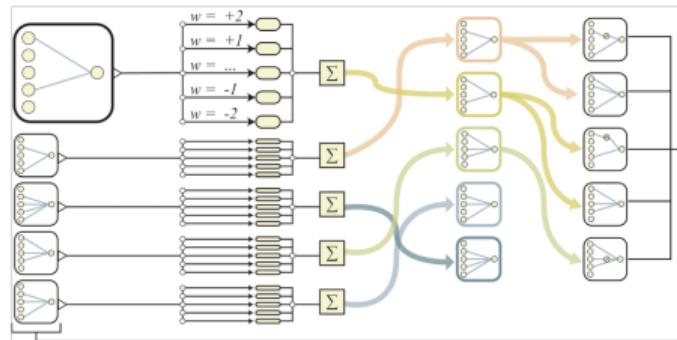
# Weight-Agnostic Method

This method was proposed in “Weight Agnostic Neural Networks” at NeurIPS 2019.

“Weight agnostic” refers to such ANNs with strong inductive biases, which can perform various tasks with randomly given initial weights, without training like traditional ANNs.

The algorithm is called neural architecture search (NAS):

- **Initialize:** creating an initial population of minimal neural network topologies.
- **Evaluate:** evaluating each network over multiple rollouts.
- **Rank:** ranking the networks according to their performance and complexity.
- **Vary:** creating a new population by varying the best network topologies.



<https://arxiv.org/abs/1906.04358>

# Single- and Multi-Agent Networks

## **Single-agent networks:**

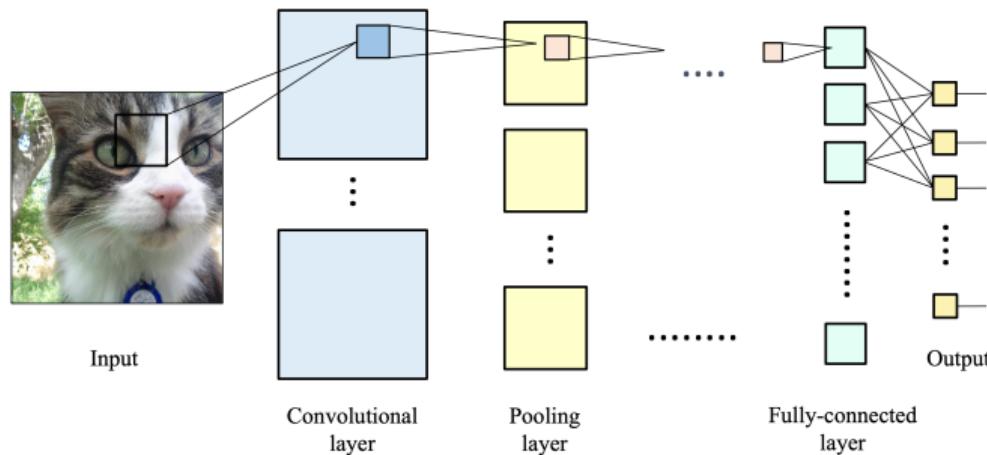
refer to a network that has the logical function of only one agent. For example, perceptron, multilayer perceptron, convolutional neural network (CNN), LSTM network, GRU network, deep Boltzmann machine (DBM), deep belief network (DBN), residual networks (ResNet), and spiking neural networks (SNN).

## **Multi-agent networks:**

refer to the networks that have the logical functions of at least two agents, with the logical relationships between those agents, include autoencoder (AE), generative adversarial network (GAN), collaborative neural network, and knowledge distillation network.

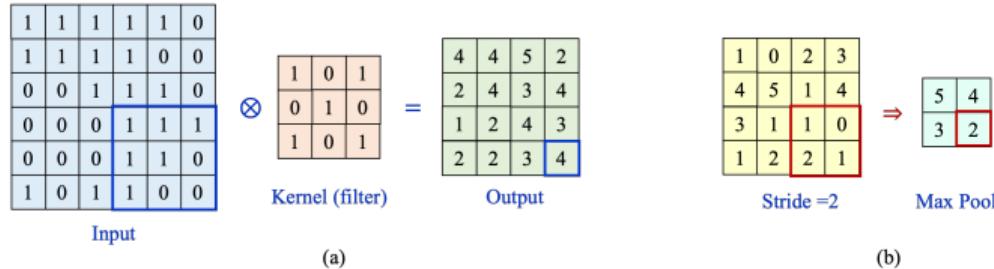
# Case Study of Single-Agent Networks

CNN is a deep network, in addition to the input layer and output layer, it includes several convolutional layers, pooling layers, and fully connected layers.



CNN mainly uses the following four strategies to reduce the number of parameters:  
(i) local connection, (ii) shared weights, (iii) pooling, and (iv) multilayer.

# Convolution and Pooling in CNN



## Convolution:

using convolutional kernel (filter). (a) is an example on a  $6 \times 6$  image, the size of filter is  $3 \times 3$ , and the filter moves across the image in a sliding window manner, obtaining different local area features.

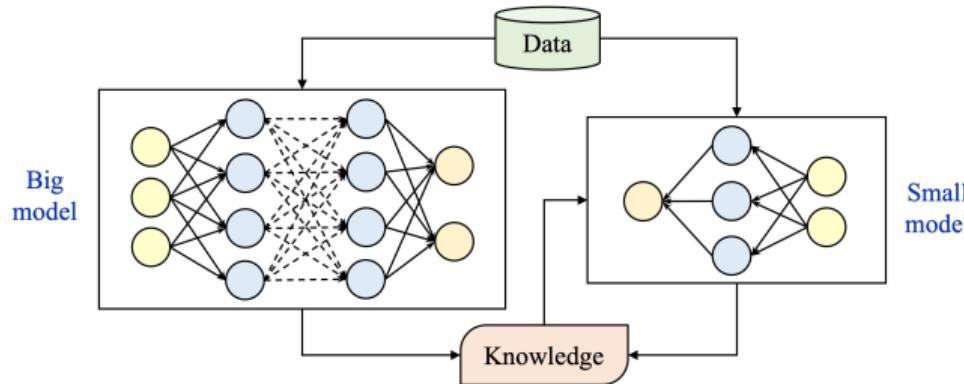
## Pooling:

including max pooling and average pooling. (b) is an example of max pooling, with the stride is equal to 2.

# Case Study of Multi-Agent Networks

Knowledge distillation network (KDN) has two agents: big model and small model.

KDN uses the knowledge distillation method to transfer the knowledge from pre-trained large model to the small model.



The large/small model can be regarded as teacher/student model.

# Thank You