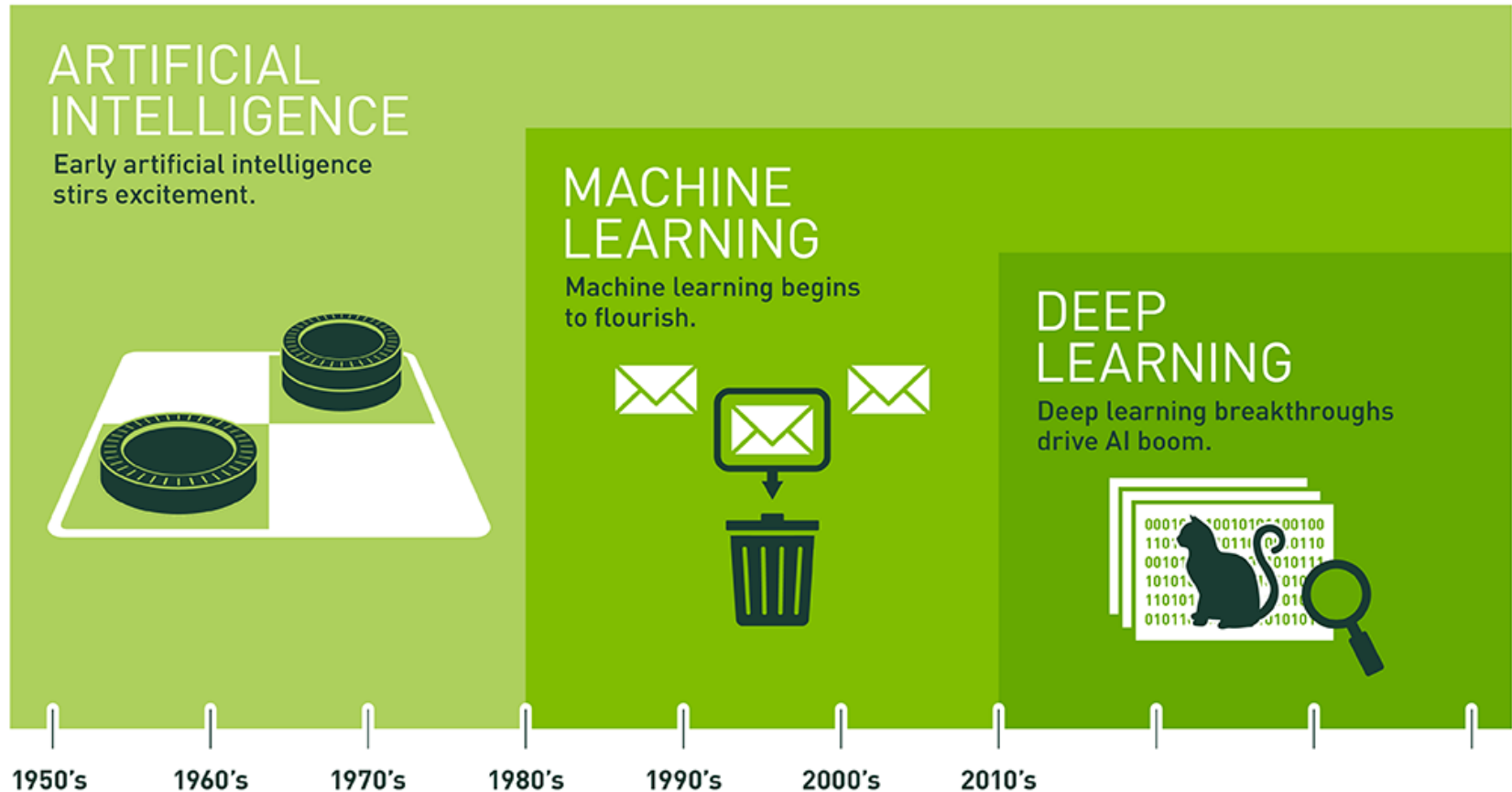# Deep Learning
Based Recommendation (Part A)

Topic 9A

Deep Learning Based Recommendation

- Introduction to Deep Learning
  - Key concepts
  - Breakthrough
  - Deep learning models
    - Multi-layer Perceptron (MLP)
    - Convolutional Neural Network (CNN)
    - Recurrent Neural Network (RNN)
    - Autoencoder

Deep Learning and AI

ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.

MACHINE LEARNING
Machine learning begins to flourish.

DEEP LEARNING
Deep learning breakthroughs drive AI boom.

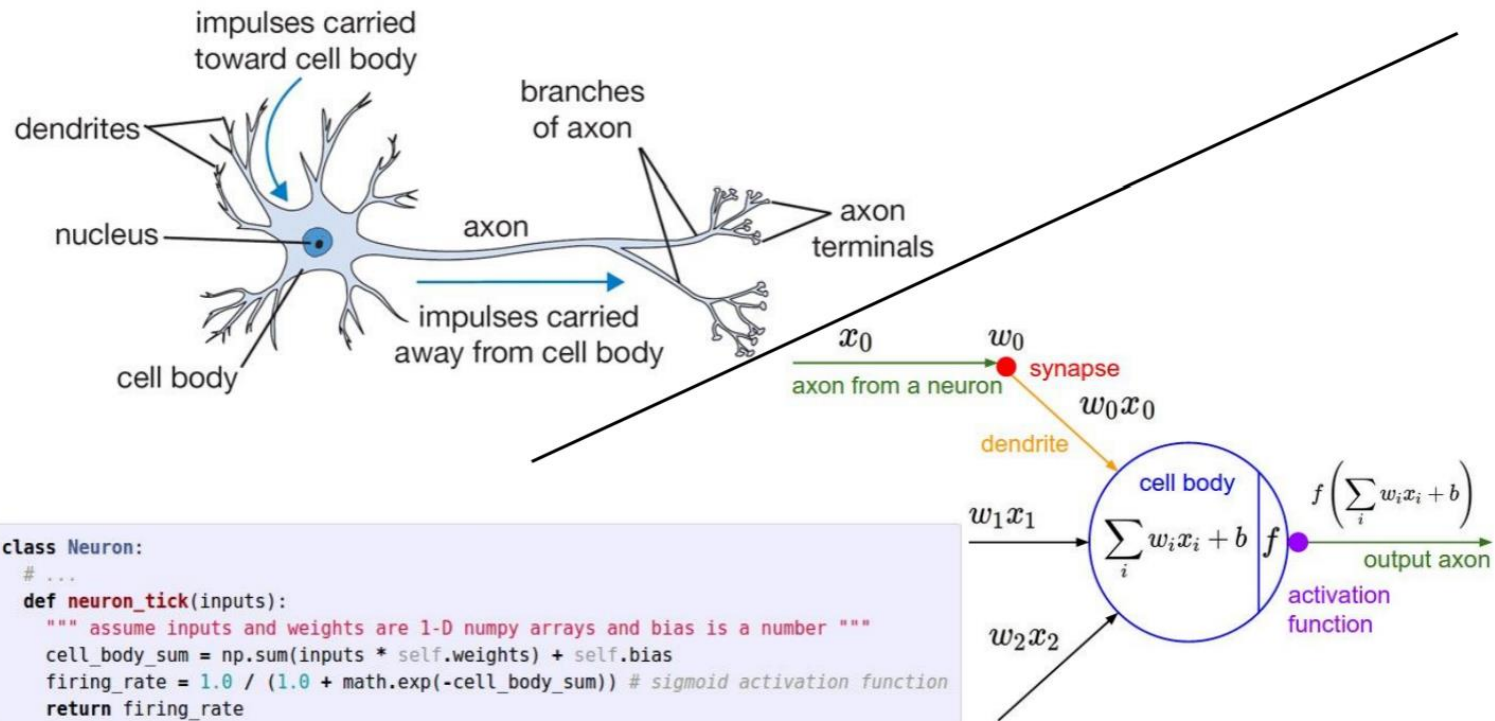1950's 1960's 1970's 1980's 1990's 2000's 2010's

Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Deep Learning

- A part of machine learning using artificial neural networks

- Deep learning methods are also considered as representation learning methods as a machine is fed with raw data and the representations needed for detection or classification are automatically discovered.
  - We can say deep learning as
    - Layered representations learning.
    - Hierarchical representations learning.

- What "Deep" means?
  - The number of layers used to create a model from the data.
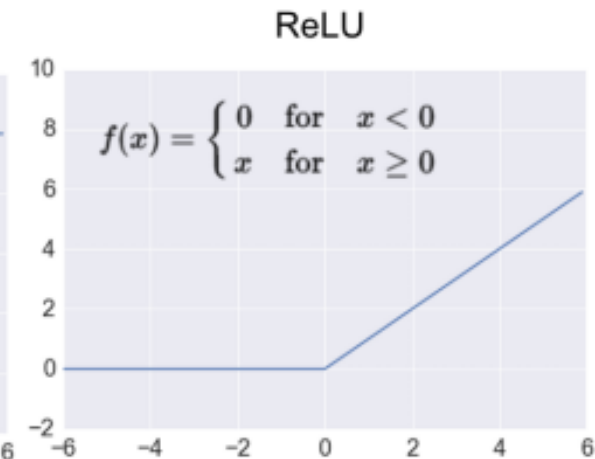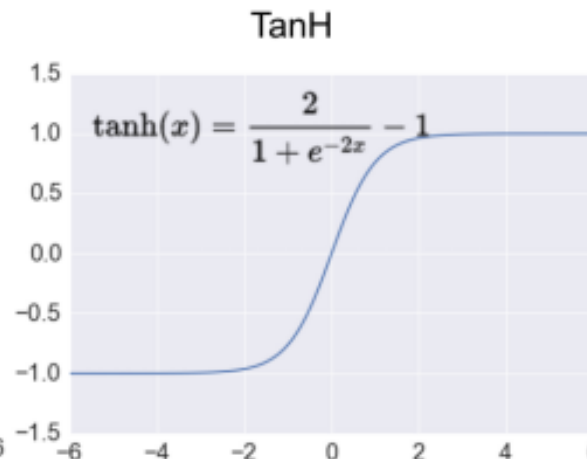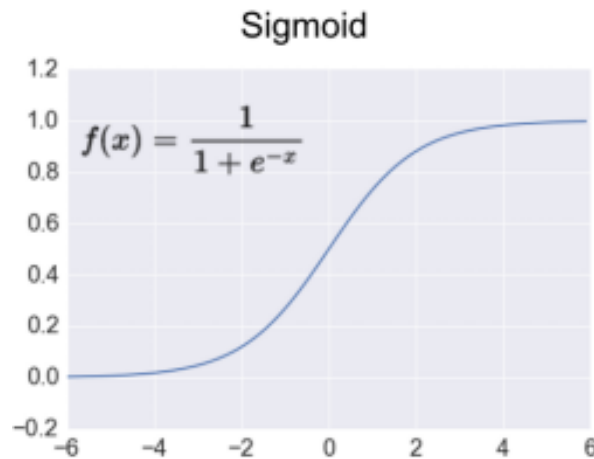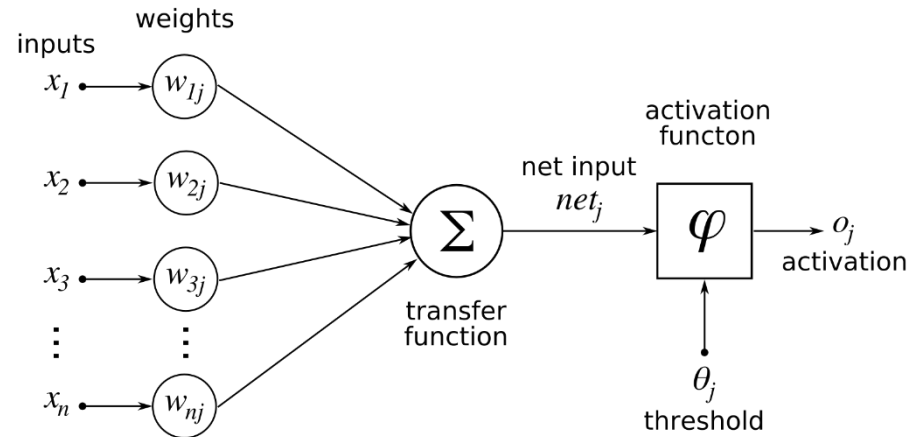   - Usually, two or more hidden layered neural network.

Two Neural Networks

□ Some of the key concepts are inspired by understanding the brain structure, but deep learning models are not brain modelling.


```
class Neuron:
    # ...
    def neuron_tick(inputs):
        """ assume inputs and weights are 1-D numpy arrays and bias is a number """
        cell_body_sum = np.sum(inputs * self.weights) + self.bias
        firing_rate = 1.0 / (1.0 + math.exp(-cell_body_sum)) # sigmoid activation function
        return firing_rate
```
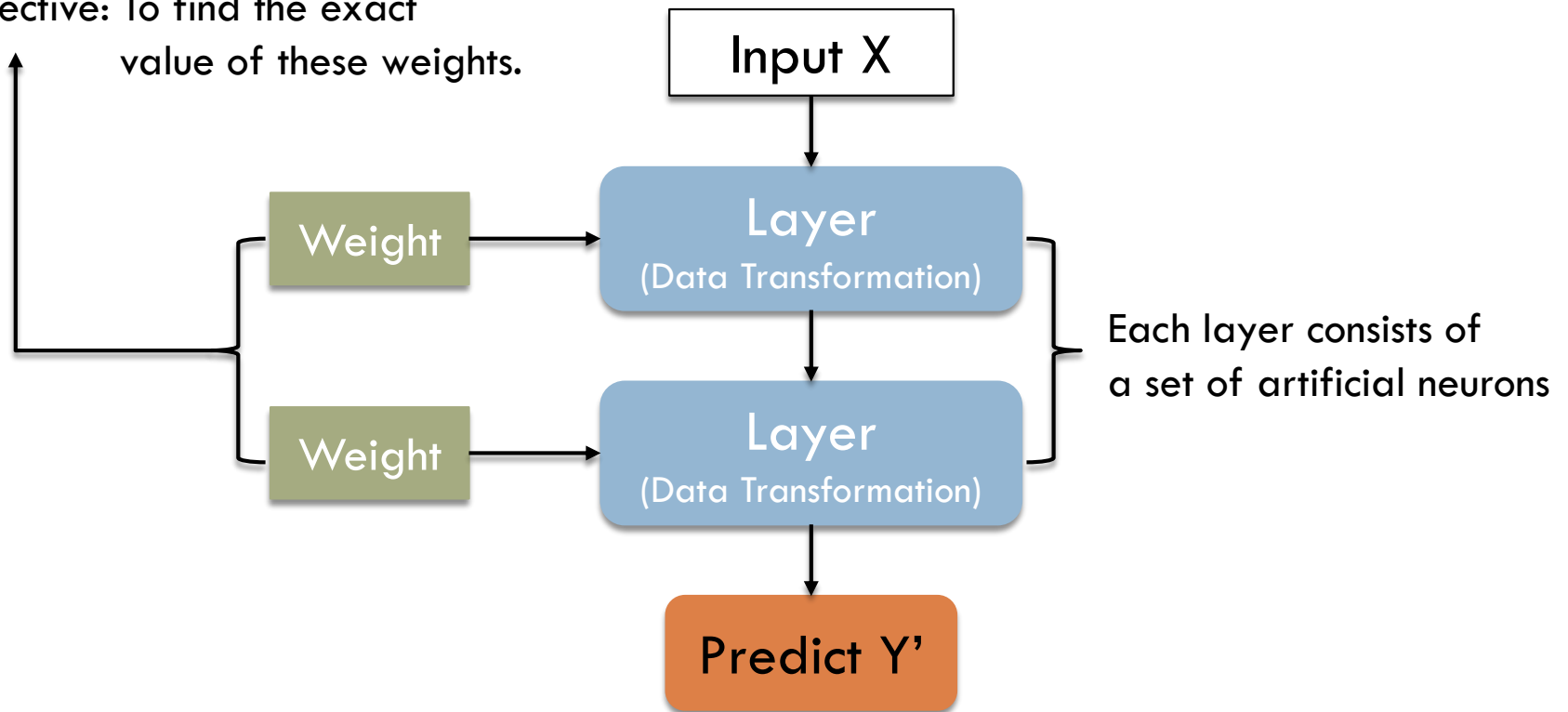
Activation Functions

Activation functions are responsible for transforming the summed weighted input (X) from the node into the output (Y).

inputs

weights

x_1 — w_{1j}

x_2 — w_{2j}

x_3 — w_{3j}

⋮ ⋮

x_n — w_{nj}

Σ

transfer function

net input net_j

activation funcion

φ → o_j activation

θ_j threshold

Sigmoid

$$f(x) = \frac{1}{1+e^{-x}}$$

TanH

$$\tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

ReLU

$$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$$

Overall Approach

Objective: To find the exact
value of these weights.

Input X

Weight

Layer
(Data Transformation)

Weight

Layer
(Data Transformation)

Each layer consists of
a set of artificial neurons

Predict Y'

Overall Approach

Objective: To find the exact
value of these weights.

```
                    ┌─────────────┐
                    │   Input X   │
                    └─────────────┘
                           │
                           ▼
  ┌──────────┐      ┌──────────────────────┐
  │  Weight  │─────▶│       Layer          │
  └──────────┘      │ (Data Transformation)│
                    └──────────────────────┘
                           │
                           ▼
  ┌──────────┐      ┌──────────────────────┐
  │  Weight  │─────▶│       Layer          │
  └──────────┘      │ (Data Transformation)│
                    └──────────────────────┘
                           │
                           ▼
              ┌─────────────┐      ┌─────────────┐
              │  Predict Y' │      │  Target Y   │
              └─────────────┘      └─────────────┘
                        │               │
                        ▼               ▼
                     ( Loss function )
                           │
                           ▼
                      ┌────────────┐
                      │ Loss score │
                      └────────────┘
```
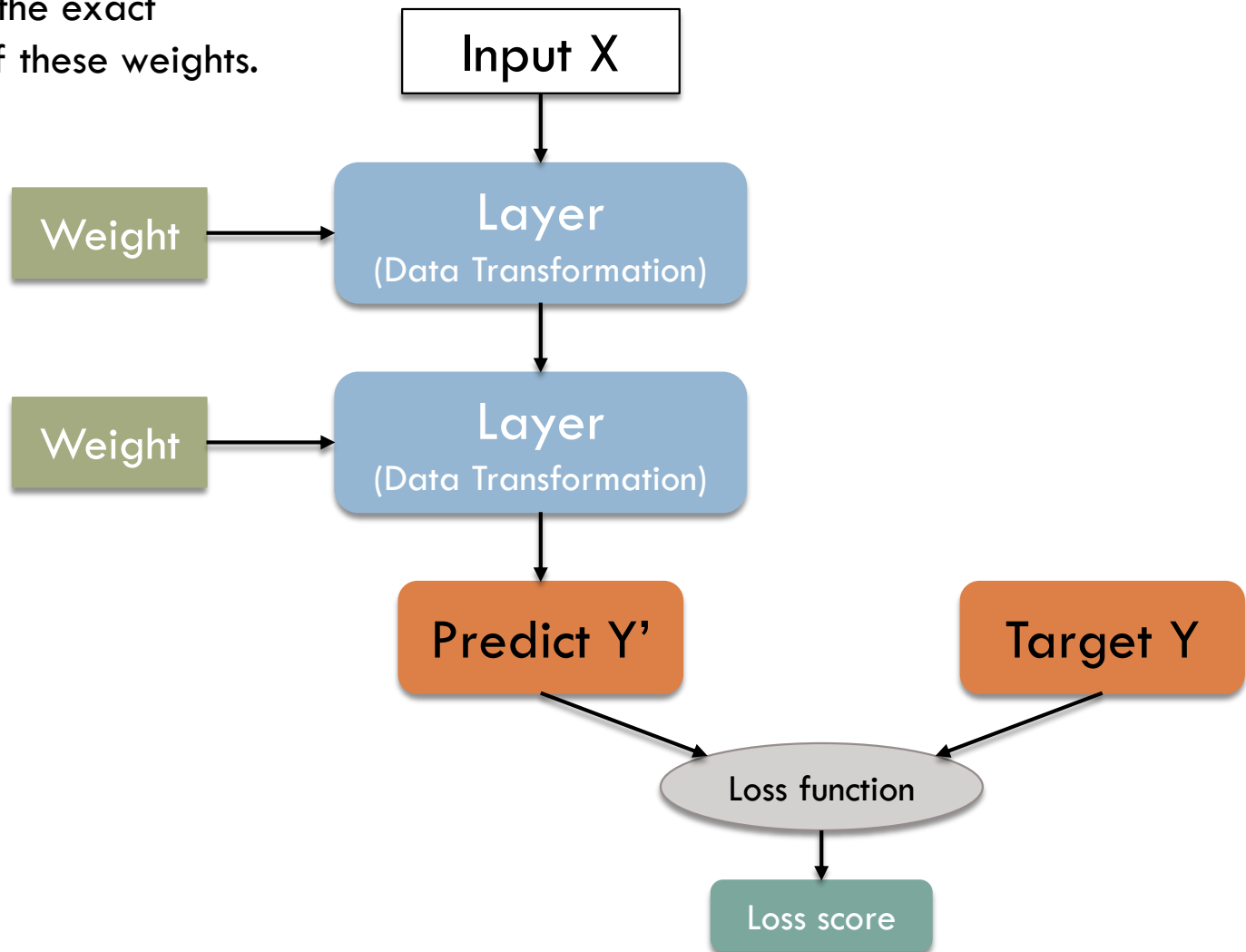
Overall Approach

Objective: To find the exact value of these weights.
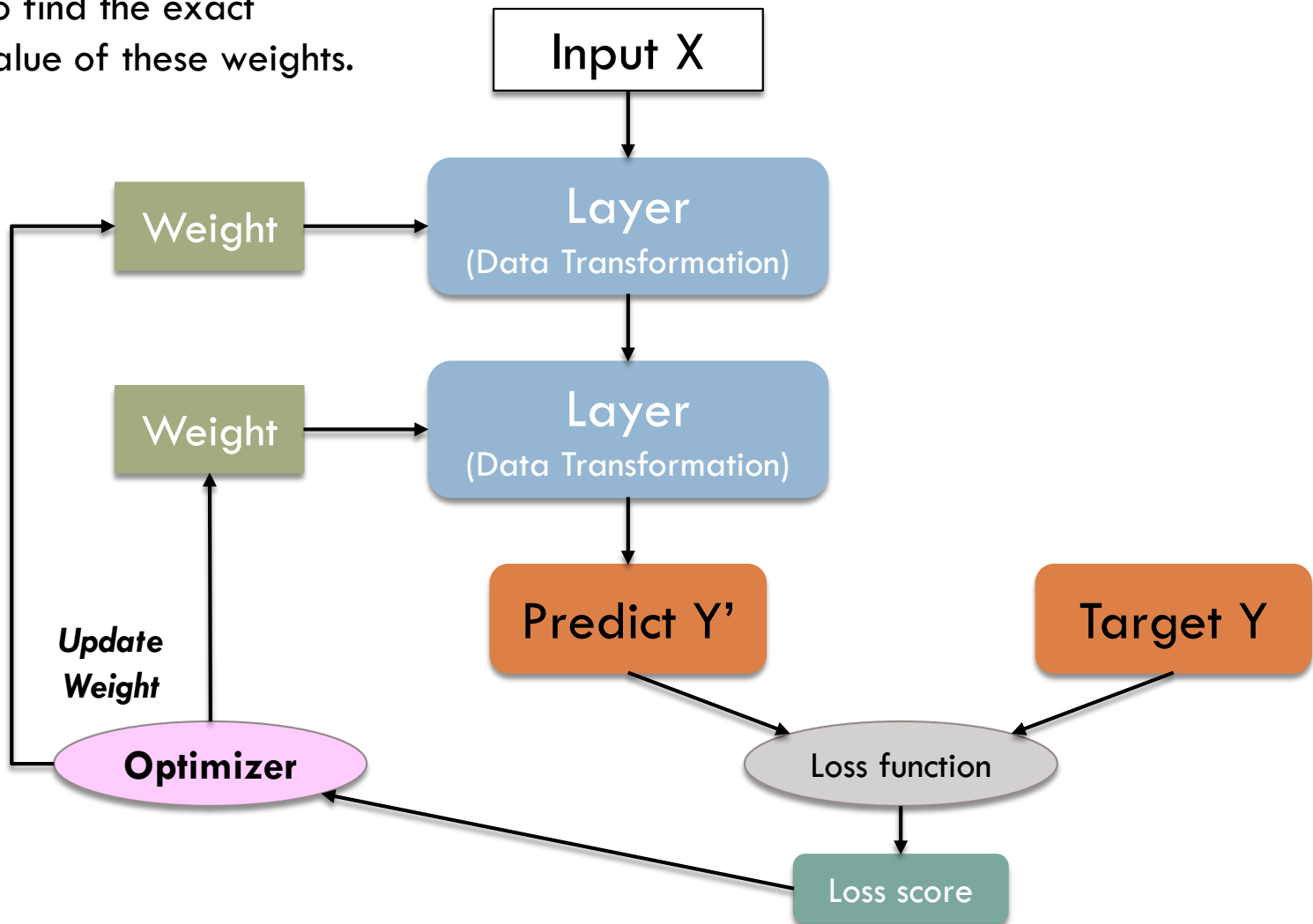
Loss Function

1.Binary Classification

NN model

input
activate
output

Layers

input → Dense → Dense → Sigmoid → output

y=0.9

error

label
t=1.0

Activation

sigmoid function

$$output = \frac{1}{1 + e^{-v}}$$
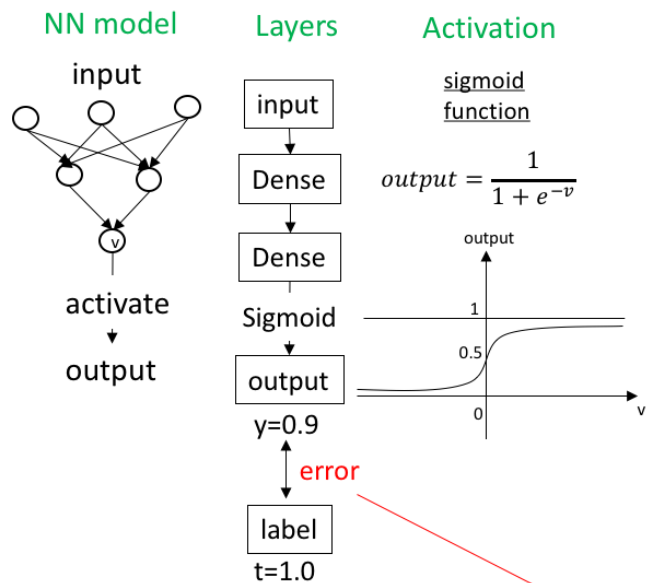
2.Multiclass Classification

NN model

input
activate
output

Layers

input → Dense → Dense → Softmax → output

y1=0.2 y2=0.1 y3=0.7

error

label
t1=0.0 t2=0.0 t3=1.0

Activation

softmax function

$$output_k = \frac{e^k}{\sum_{i=1}^{n} e^i}$$

Cross Entropy(CE)

$$L = -\sum t_i log y_i$$

3.Regression

NN model

input
output

Layers

input → Dense → output

y=417

error

label
t=520
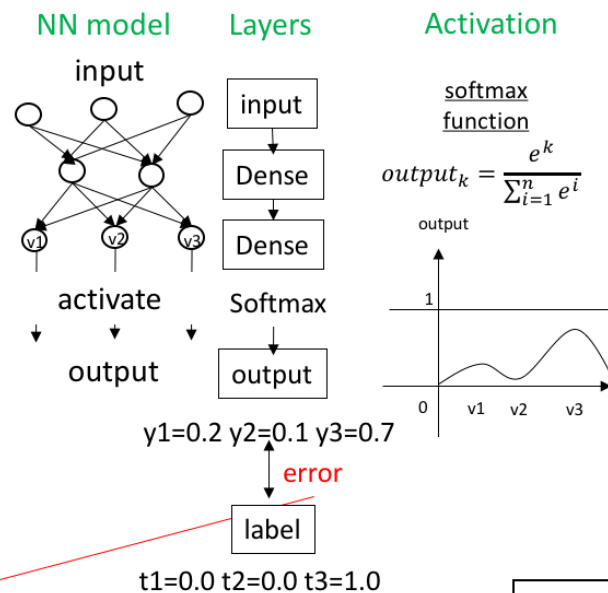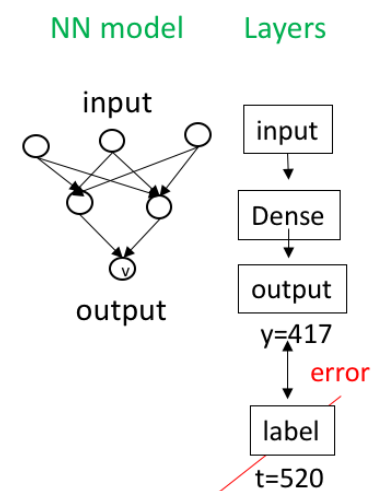
Mean Squared Error(MSE)
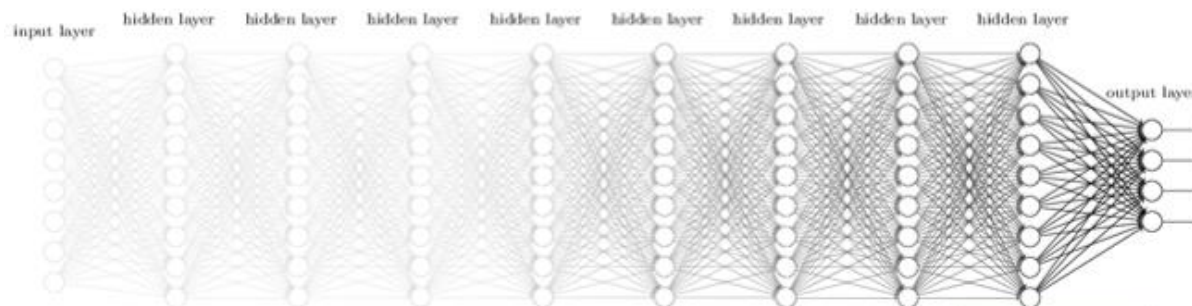
$$L = \frac{1}{2}(t - y)^2$$

Backpropagation
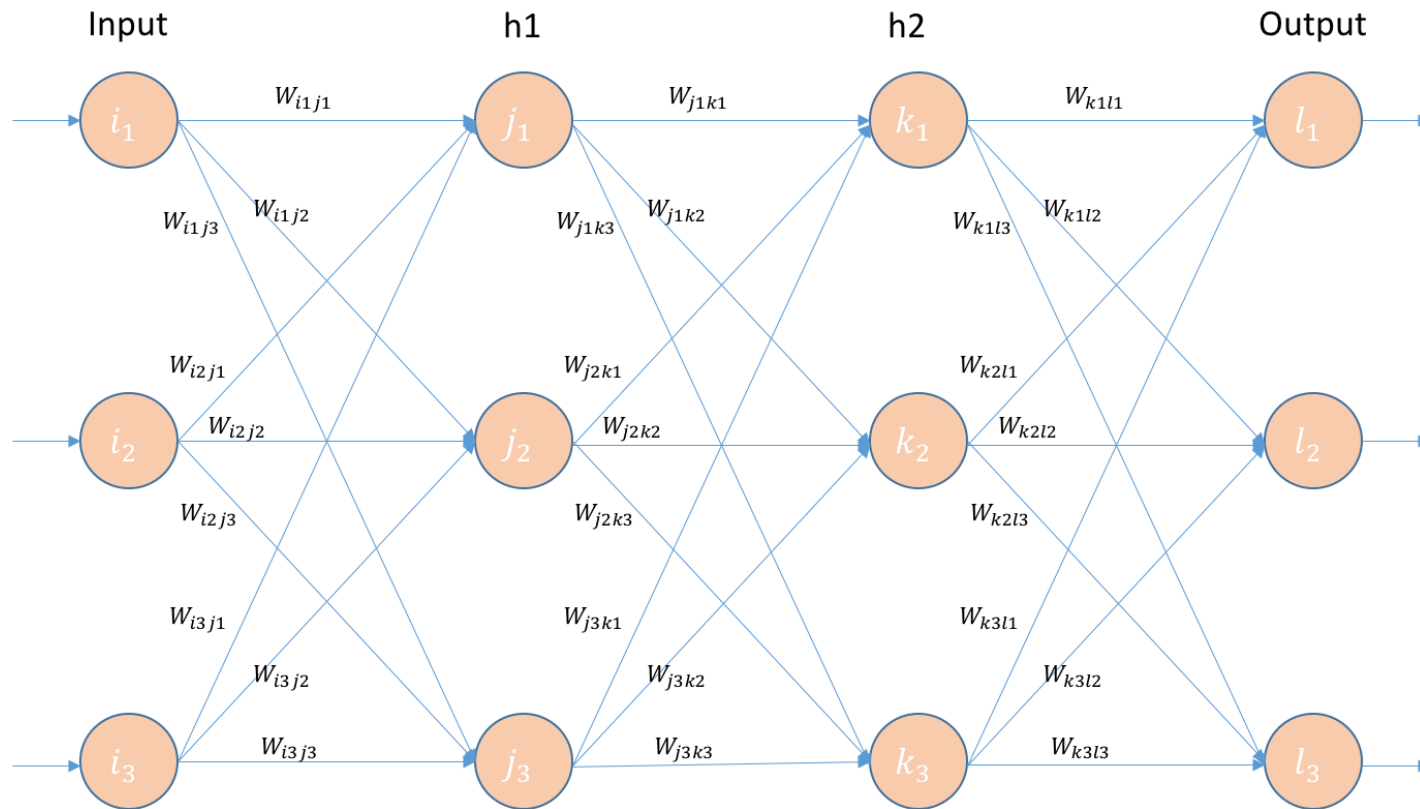
Breakthrough

- Problem of Deep neural network
  - Vanishing gradient: feedback signal(loss) is faded as the layer increases.
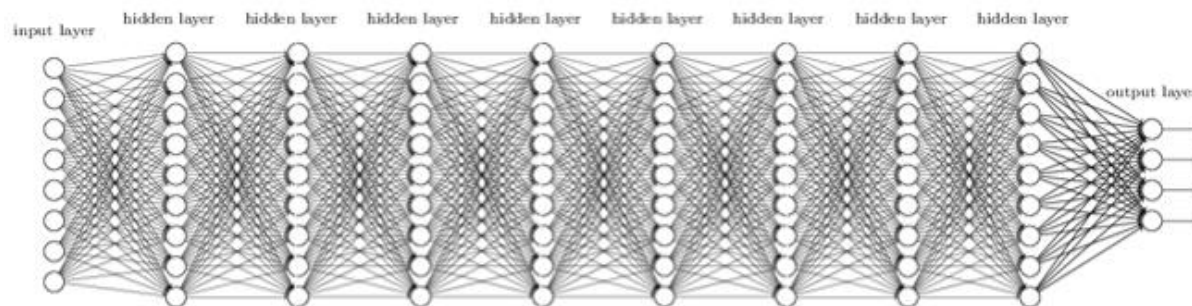
Deep Neural Network

Vanishing Gradient

Breakthrough

- Solutions in 2009 ~ 2010
  - Activation function better suited to the layer of the neural network: Relu activation.
  - Weighted initialization method that makes pre-training existing layer unnecessary: Xavier or He's initialization.
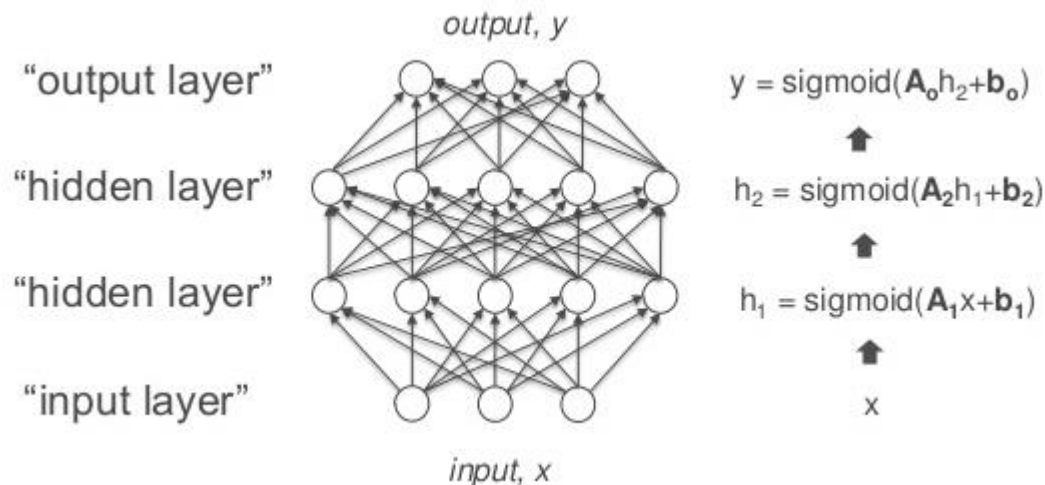 - Optimizer: Adam.

```
# Xavier initialization
# Glorot et al. 2010
W = np.random.randn(fan_in, fan_out)/np.sqrt(fan_in)

# He et al. 2015
W = np.random.randn(fan_in, fan_out)/np.sqrt(fan_in/2)
```
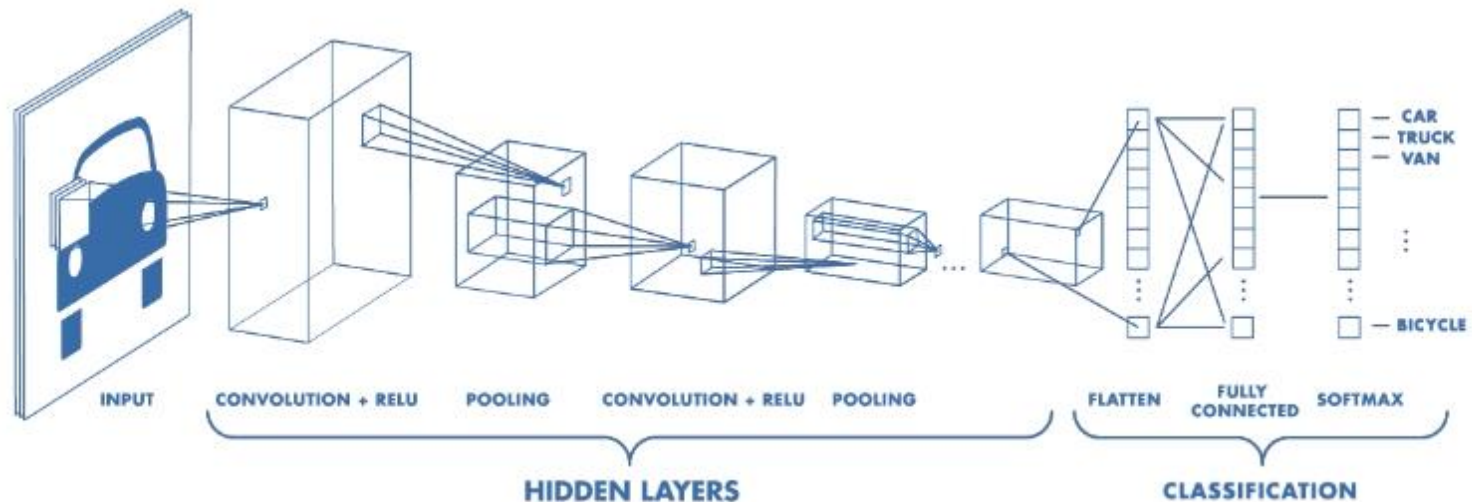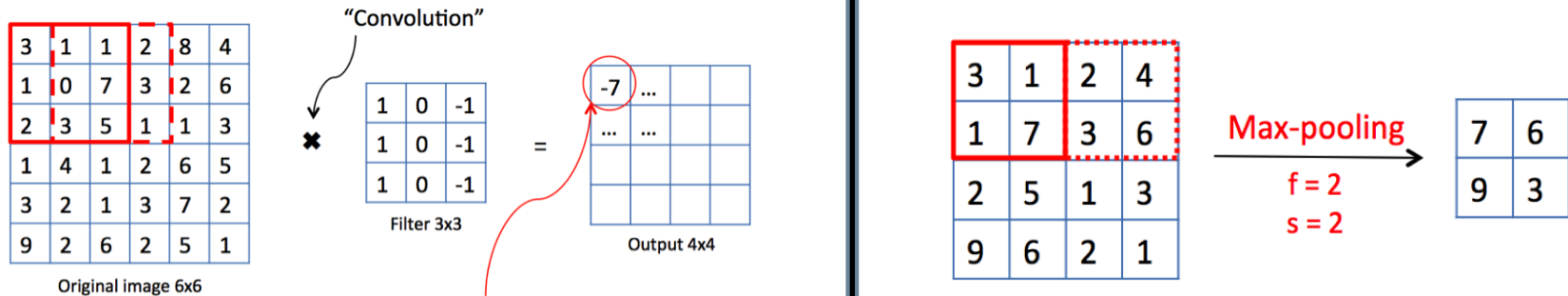
- Solutions in 2014 ~ 2016
  - Batch normalization.
  - Residual connection.
  - Depthwise separable convolution.

Deep Learning - MLP

- Multi-Layer Perceptron (MLP)
  - A feed-forward artificial neural network with one or more layers between input and output layer.
  - The input is transformed using a learnt non-linear transformation.
 - Fully-connected layer (FC) or Dense layer

output, y

"output layer"   ◯ ◯ ◯ $y = sigmoid(\mathbf{A_o}h_2 + \mathbf{b_o})$

"hidden layer"   ◯ ◯ ◯ ◯ $h_2 = sigmoid(\mathbf{A_2}h_1 + \mathbf{b_2})$

"hidden layer"   ◯ ◯ ◯ ◯ $h_1 = sigmoid(\mathbf{A_1}x + \mathbf{b_1})$

"input layer"   ◯ ◯ ◯   x

input, x

Deep Learning - CNN

Deep Learning - RNN

☐ Recurrent networks have two sources of input, **the current input** and **the recent past**, which gets combined to determine how they respond to new data.

☐ This model is usually used to language modeling.
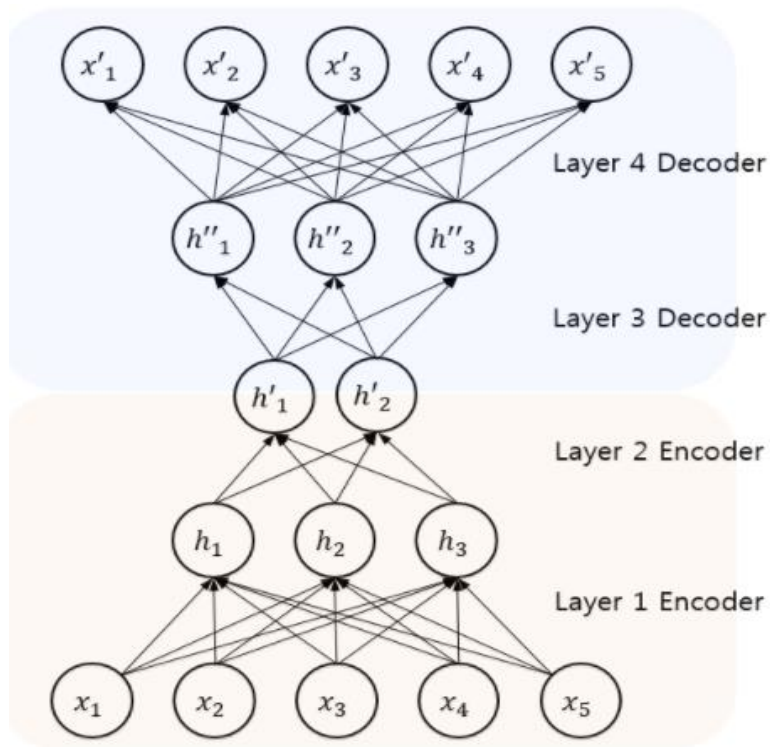

Recent past   Current input
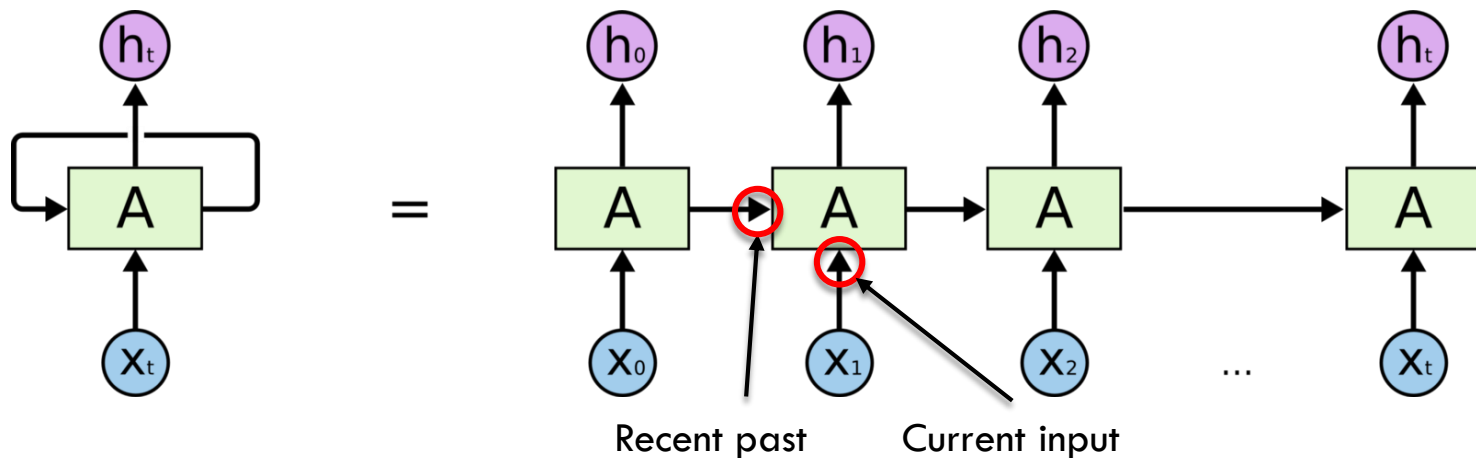
Deep Learning - Autoencoder

- An autoencoder is a type of neural network used to learn efficient latent representation in unsupervised learning method.
  - Input and output of autoencoder are same dimension.
- The objective is to reduce dimension in data.

Decoder : $z = g_{\theta'}(y) = s(W'y + b')$

Encoder : $y = f_\theta(x) = s(Wx + b)$
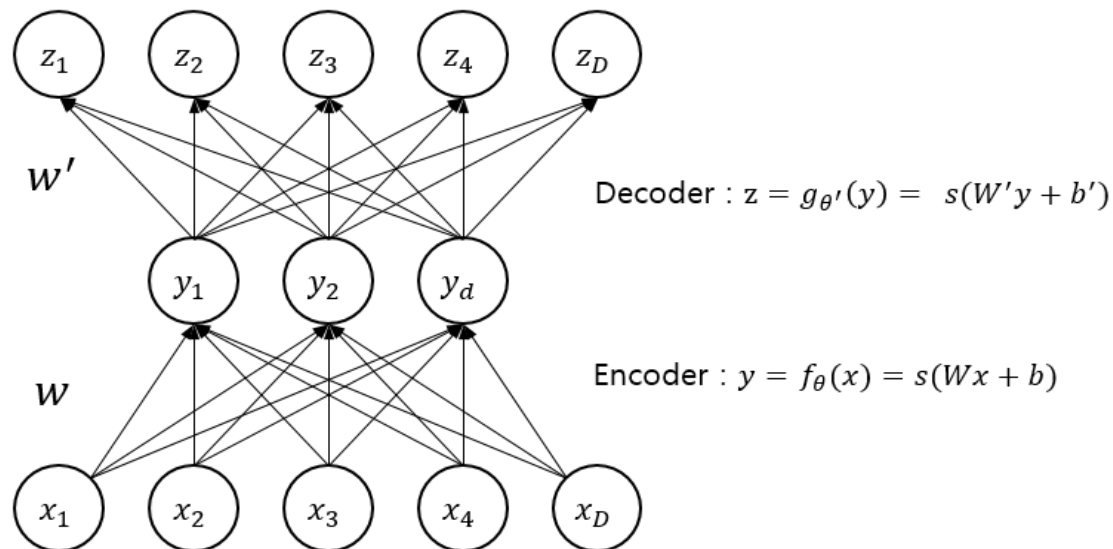
Deep Learning - Autoencoder

☐ Stacked Autoencoder.   ☐ Denoising Autoencoder.