Deep Learning

Md. Jalil Piran, PhD
Asst. Professor
Computer Science and Engineering
Sejong University
Spring, 2021

Outline

- Introduction to Deep Learning (DL)
- The History of DL
- Programming Tools
- Artificial Neural Networks

ARTIFICIAL NEURAL NETWORKS

Neural Networks

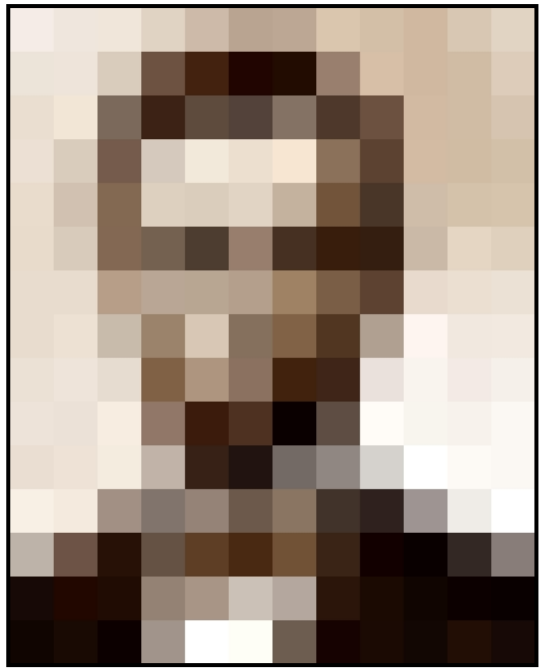
- Neural Networks can be :
  - **Biological** models
  - **Artificial** models

- **Goal**: to produce artificial systems capable of sophisticated **computations** similar to the **human brain**.

Classification

- **Computer vision**; photo classification example.

What humans see?

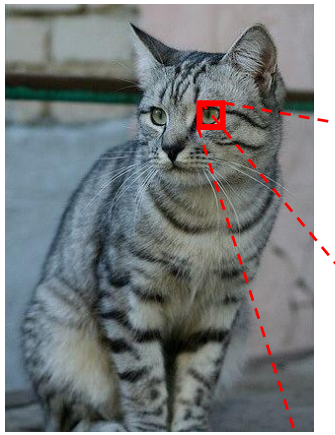


What computer see?

243	239	240	225	206	185	188	218	211	206	216	225
242	239	218	110	67	31	34	152	213	206	208	221
243	242	123	58	94	82	132	77	108	208	208	215
235	217	115	212	243	236	247	139	91	209	208	211
233	208	131	222	219	226	196	114	74	208	213	214
232	217	131	116	77	150	69	56	52	201	228	223
232	232	182	186	184	179	159	123	93	232	235	235
232	236	201	154	216	133	129	81	175	252	241	240
235	238	230	128	172	138	65	63	234	249	241	245
237	236	247	143	59	78	0	94	255	248	247	251
234	237	245	193	55	33	115	144	213	255	253	251
248	245	161	128	149	109	138	65	47	156	239	255
190	107	39	102	94	73	114	58	17	0	51	137
23	32	33	148	168	203	179	43	27	17	18	4
17	26	19	160	255	255	109	33	26	19	35	24

Classification

In real:



In a computer

```
[[105 112 108 111 104  99 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 128 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  93 101  94]
 [106  91  61  64  69  91  88  85 101 107 109  98  75  84  96  95]
 [114 108  85  55  55  69  64  54  64  87 112 129  98  74  84  91]
 [133 137 147 103  65  81  80  65  52  54  74  84 102  93  85  82]
 [128 137 144 140 109  95  86  70  62  65  63  63  60  73  86 101]
 [125 133 148 137 119 121 117  94  65  79  80  65  54  64  72  98]
 [127 125 131 147 133 127 126 131 111  96  89  75  61  64  72  84]
 [115 114 109 123 150 148 131 118 113 109 100  92  74  65  72  78]
 [ 89  93  90  97 108 147 131 118 113 114 113 109 106  95  77  80]
 [ 63  77  86  81  77  79 102 123 117 115 117 125 125 130 115  87]
 [ 62  65  82  89  78  71  80 101 124 126 119 101 107 114 131 119]
 [ 63  65  75  88  89  71  62  81 120 138 135 105  81  98 110 118]
 [ 87  65  71  87 106  95  69  45  76 130 126 107  92  94 105 112]
 [118  97  82  86 117 123 116  66  41  51  95  93  89  95 102 107]
 [164 146 112  80  82 120 124 104  76  48  45  66  88 101 102 109]
 [157 170 157 120  93  86 114 132 112  97  69  55  70  82  99  94]
 [130 128 134 161 139 100 109 118 121 134 114  87  65  53  69  86]
 [128 112  96 117 150 144 120 115 104 107 102  93  87  81  72  79]
 [123 107  96  86  83 112 153 149 122 109 104  75  80 107 112  99]
 [122 121 102  80  82  86  94 117 145 148 153 102  58  78  92 107]
 [122 164 148 103  71  56  78  83  93 103 119 139 102  61  69  84]]
```

Pixels intensity values, e.g. brightness

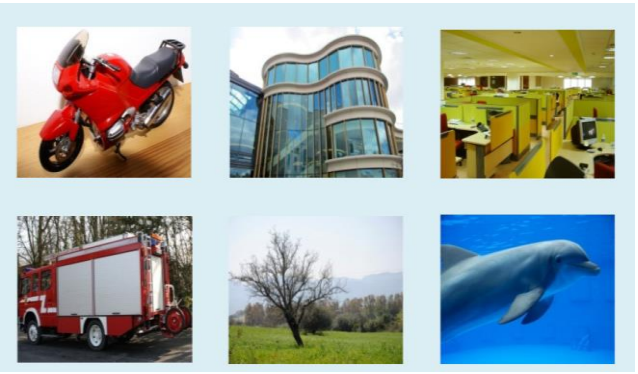
- An image is just a big grid of numbers between [0, 255]:
- e.g. $800 \times 600 \times 3$ (3 channels RGB)

Computer Vision

Classification

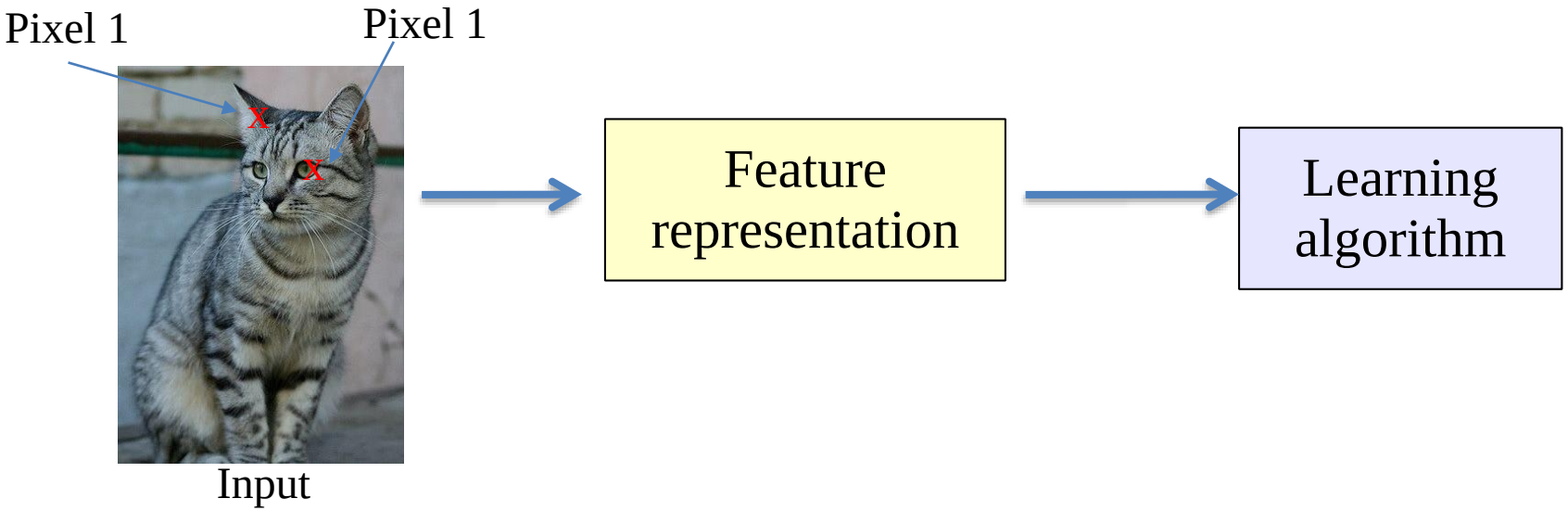
Cat detection

Not a cat



Testing:

Classification



Computer Vision

Classification

Images/video

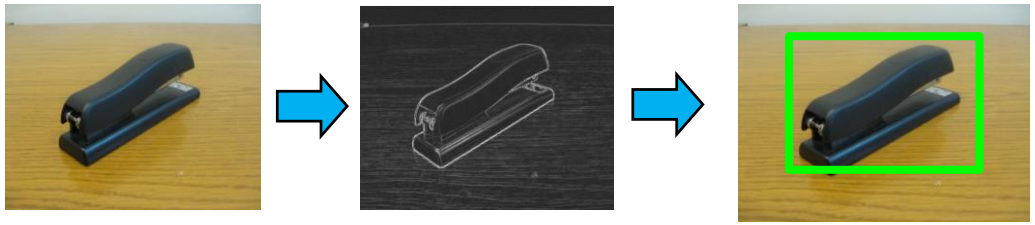
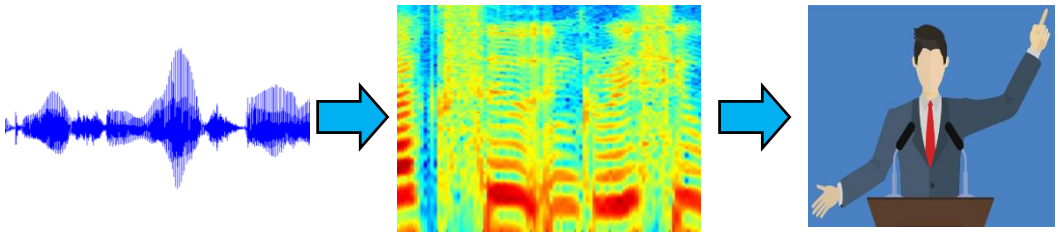


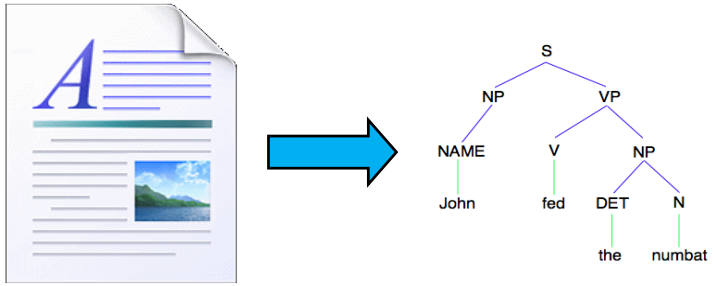
Image Vision features Detection

Audio



Audio Audio features Speaker ID

Text

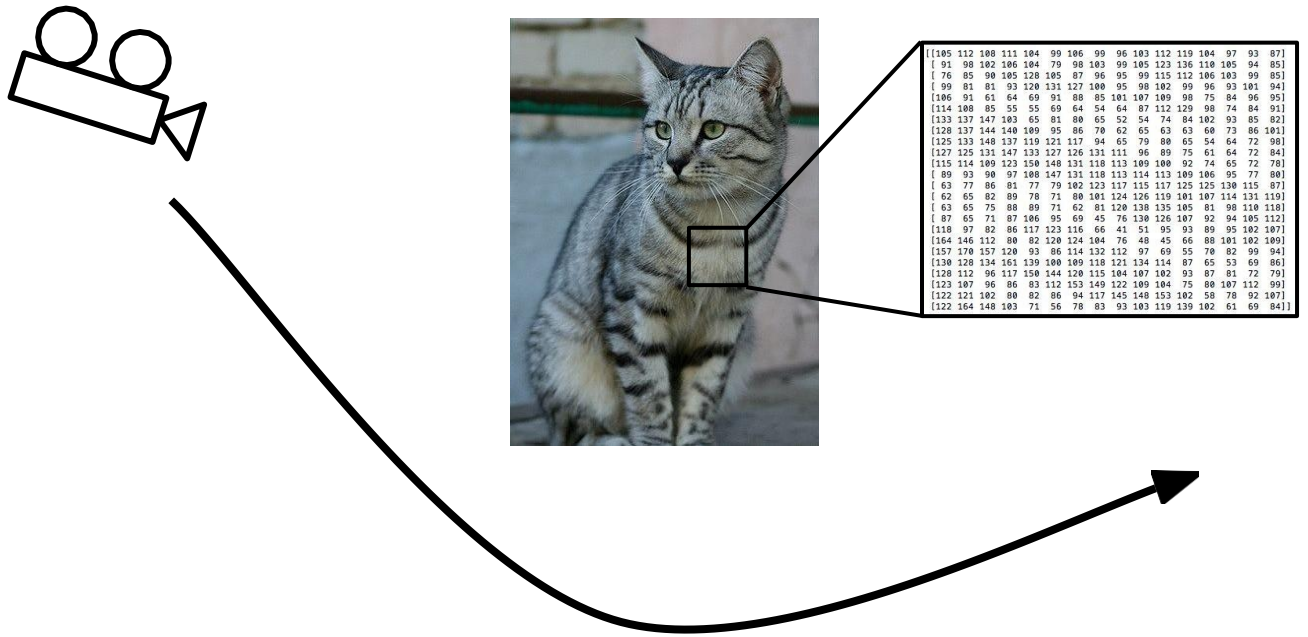


Text Text features

Text classification,
Machine translation,
Information retrieval, ...

Challenges

- Viewpoint variation

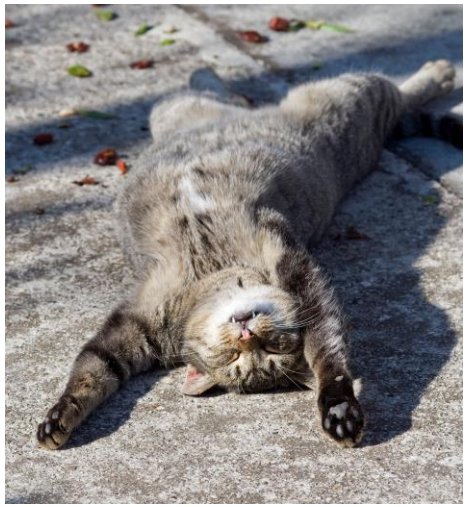


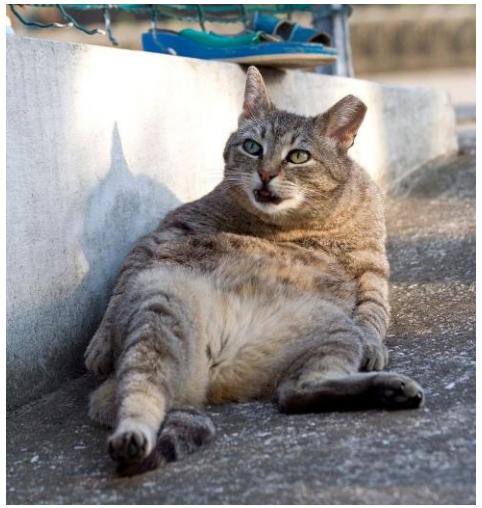
Challenges

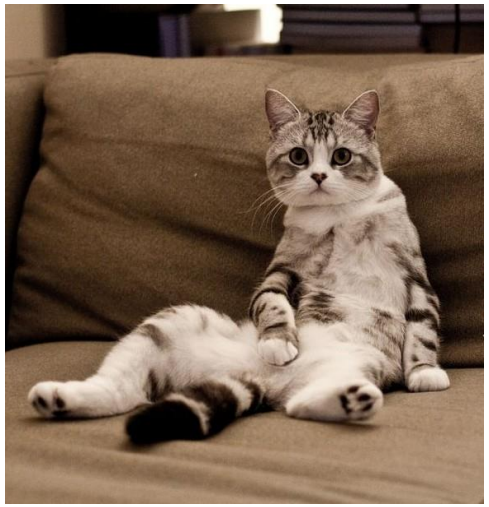
- Illumination

Challenges

- Deformation







Challenges

- Occlusion

Challenges

- Background Clutter

Challenges

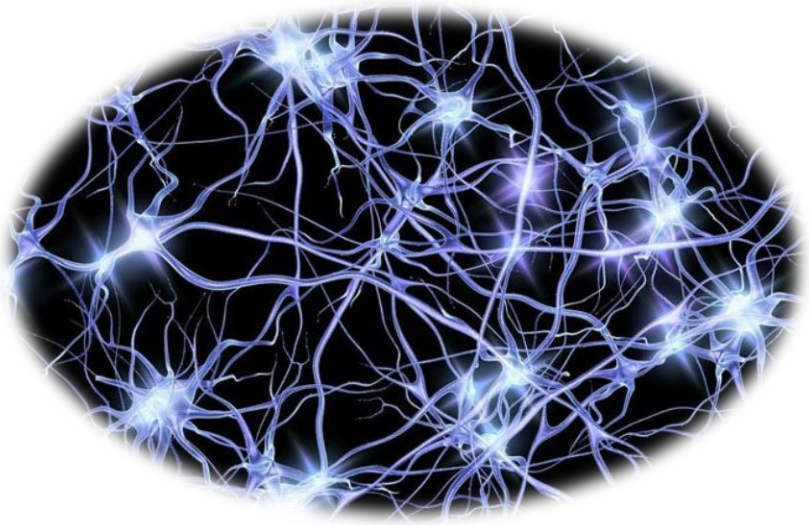
- Intra-class variation

The Brain

- Human brain "**computes**" in an entirely different way from conventional digital computers.
- The brain is highly **complex**, **nonlinear**, and **parallel**.
- Organization of neurons to perform tasks much faster than computers.
- Typical time taken in visual recognition tasks is 100–200 ms.
- Key features of the biological brain:
  - Experience shapes the wiring through plasticity,
  - Hence learning becomes the central issue in neural networks.

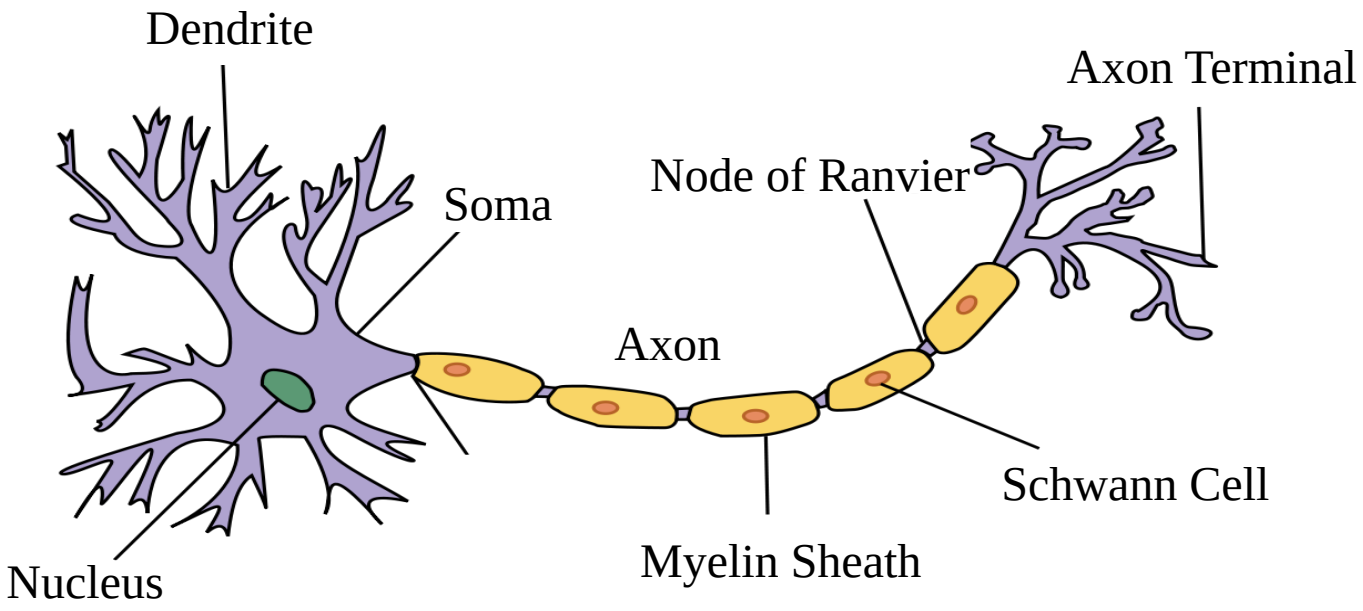
Human Neurons

- Number of **neurons**
  - ~ 10^{10}
- Neuron **switching time**
  - ~ 0.001 second
- **Connections** per neuron
  - ~ $10^{4\text{-}5}$
- **Scene recognition** time
  - ~ 0.1 second
- 100 inference steps doesn't seem like enough
  - much parallel computation



Human Neurons

- A biological neuron basically;
  - receives inputs from other sources (**dendrites**),
  - merges them in some way (**soma**),
  - performs an operation on the result (**axon**),
  - outputs the result - possibly to other neurons (**axon terminals**).



ANN

- An ANN is a massively **parallel distributed processor** made up of simple **processing units**.
- Each **unit** has a natural propensity for storing experimental knowledge and making it available for use.
- Neural networks **resemble the brain**:
  - **Knowledge** is acquired from the environment through a **learning process**.
  - Inner neuron connection **strengths**, known as synaptic **weights**, are used to store the acquired knowledge.
- Procedure used for learning:
  - **Learning algorithm**,
  - **Weights**,
  - or even the **topology** can be adjusted.

ANN

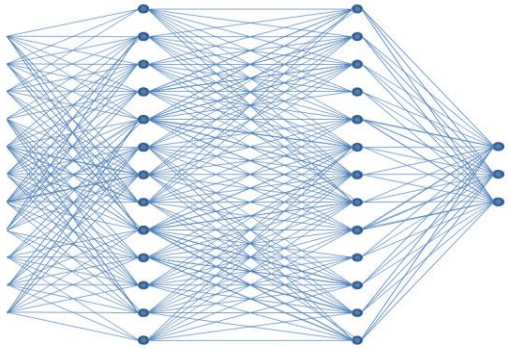
- ANN "**learns**" to perform tasks by considering **examples**, generally without being programmed with any task-specific rules.
- ANN is based on a collection of connected **nodes** called **Artificial Neurons**, “analogous to a biological brain”.
- Each connection can transmit a signal from one neuron to another, like the **synapses** in a biological brain.
- An artificial neuron that receives a signal can **process** it and then **signal** additional artificial neurons connected to it.

ANN

Definition

- "An **information processing system** that has been developed as a generalization of mathematical models of human cognition or neurobiology, based on the assumptions that:
  - Information processing occurs at many simple elements called **Neurons**.
  - Signals are passed between neurons over connection **Links / Channels**.
  - Each connection link has an associated **Weight**, which typically multiplies the signal transmitted.
  - Each neuron applies an **Activation Function** (usually non-linear) to its net input (sum of weighted input signals) to determine its output signal."

ANN



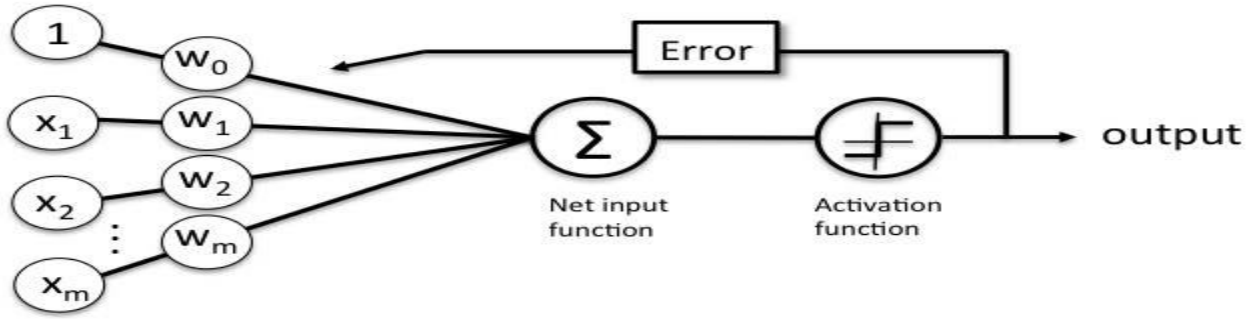
Dendrites	**Weights**,
Soma	**Input function** (summation function)
Axon	**Activation function** (transfer function)
Axon terminals	**Output**

ANN

- The procedure:
  - **Input values** enter the neuron via the **connections**.
  - The **inputs** are multiplied with the **weighting factor** of their respective connection.
    - There is often a separate **bias** connection, acts as a threshold for the neuron to produce some useful output.
  - The modified inputs are fed into a **summation function**.
    - Usually just sums the products.
  - The result from the summation function is sent to a **transfer function**.
    - Usually a **step function**, or a **sigmoid function**.
  - The neuron **outputs** the result of the transfer function into other neurons, or to an outside connection.

Types

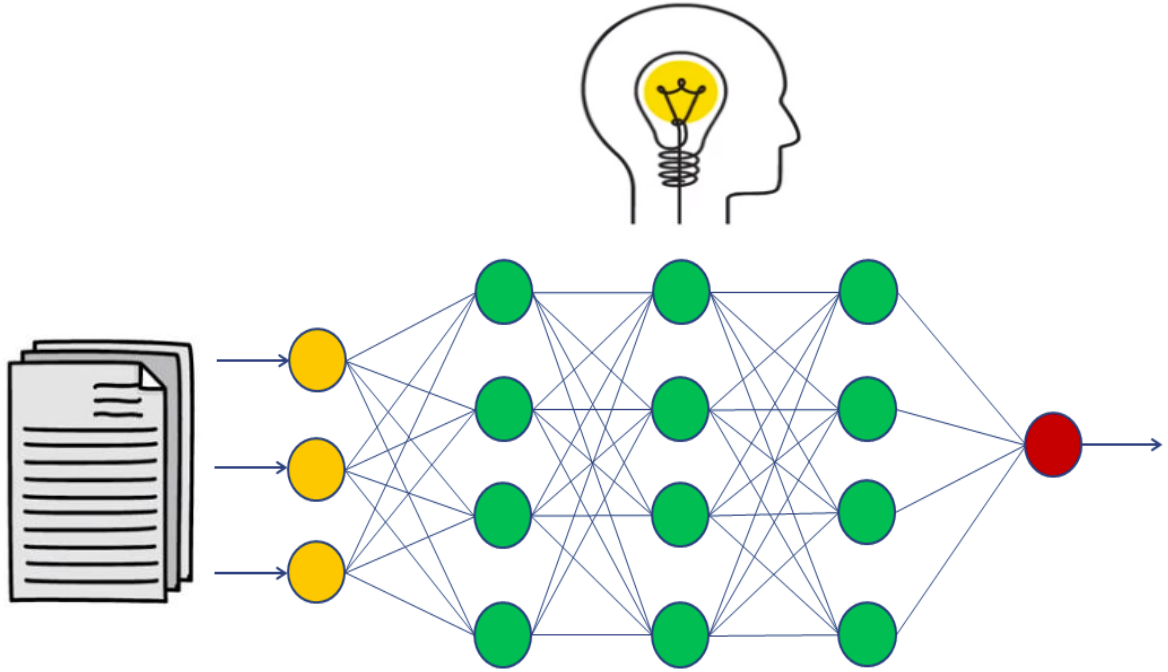
- **Single-layered NN**
  - Multiple input sources will be fed into the set of neurons, which produce the outputs to the NN, e.g. **perceptron**.
  - Can represents only linearly separable functions.



- **Multi-layered NN**
  - More powerful than single-layered NN
  - High complexity
  - High training time

ANN

- In nutshell, an ANN;
  takes in **data**,
  train itself to recognize the patterns in the data,
  and **predict** the output.



Output

ANN

- **Neurons** vs. **Units**
  - Each element of NN is a node called **unit**.
  - Units are connected by **links**.
  - Each link has a numeric **weight**.

ANN

Planning in Building an ANN

- The **number** of **units** to use.
- The **type** of **units** required.
- **Connection** between the units.

ANN

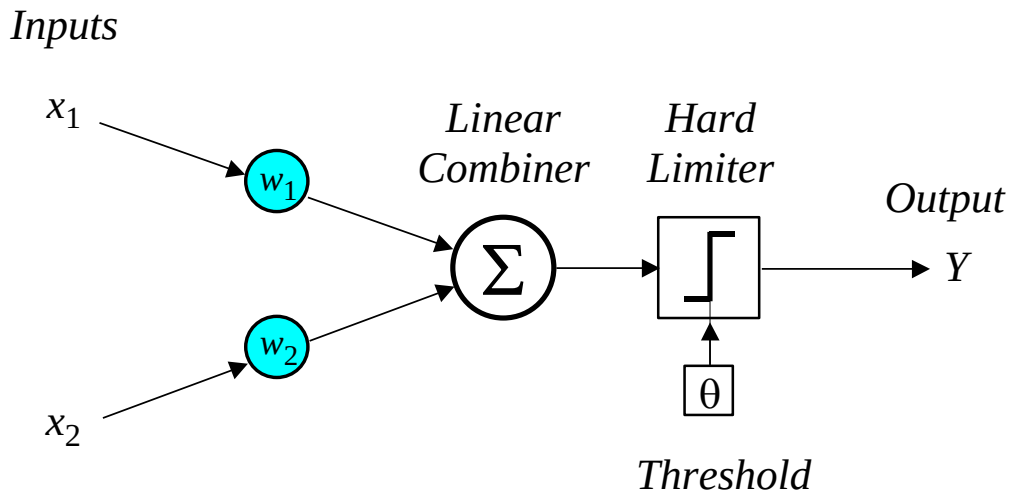
- **Batch**: dividing the dataset into number of small pieces instead of passing the entire dataset into the ANN at once.

- **Iteration**: running the procedure over each batch.
  - Example) having 10,000 images as dataset and a batch size of 200,
  - Then an epoch should contain 10,000/200 = 50 iterations.

- **Epoch**: one iteration over entire dataset.

Architecture

- **Input Layer**
- **Hidden Layer(s)**

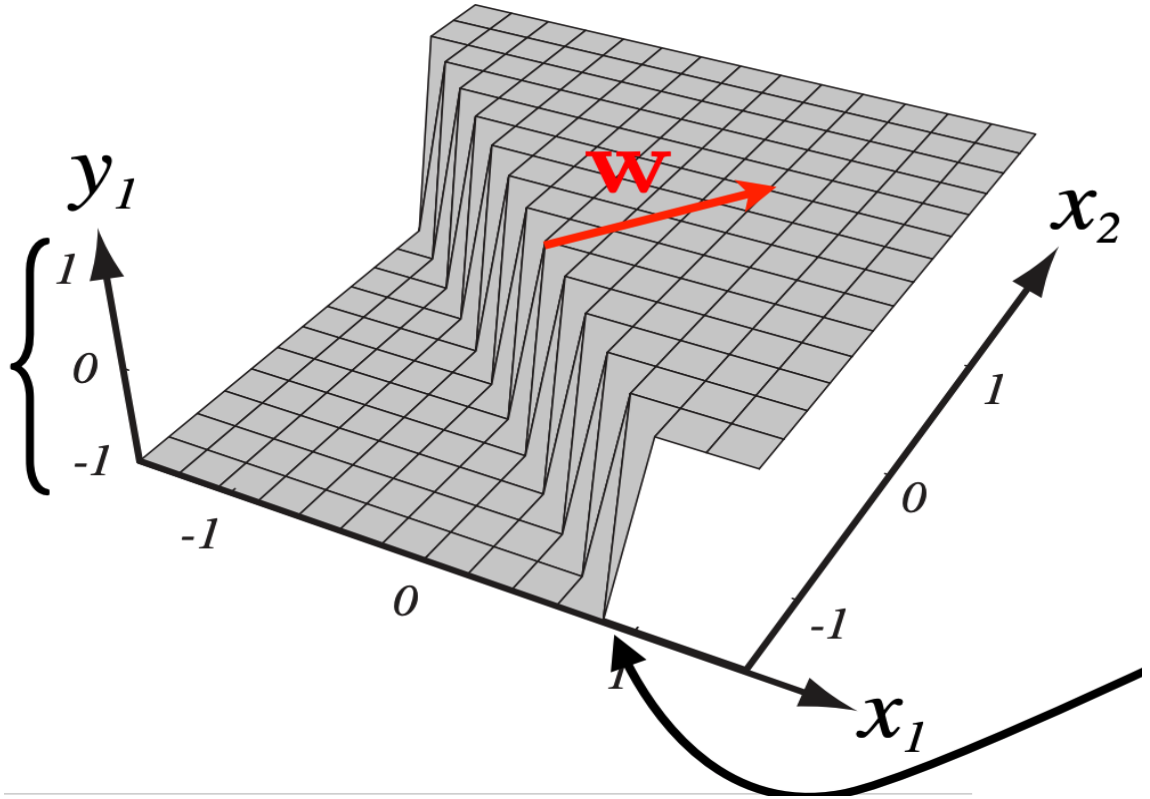


- **Activation Function:** defines the output of that node given an input or set of inputs.
- **Weights**, adjusts as learning proceeds.
  weight increases or decreases the strength of the signal at a connection.
- **Biases**; sometimes a bias term is added to the total weighted sum of inputs to serve as a **threshold** to shift the activation function.
- **Output Layer** represents the results.

Architecture

weights bias activation function

range determined by g(.)



bias only change the position of the hyperplane

Activation Function

- The activation function determines whether a particular node be **activated** or not.
- Only the activated neuron **transmits** data to the next layer.

"*Neurons that fire together wire together*"

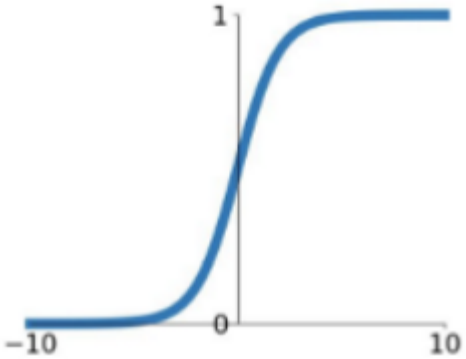
_Hebbian Theory

- We may try different set of functions and see which works best on the problem at hand:
  - Sigmoid Function
  - Rectified Linear Unit (ReLU)
  - Leaky Rectified Linear Unit
  - Hyperbolic Tangent (tanh)
  - Exponential Linear Units (ELU)
  - …

Activation Function

Sigmoid Function (σ)

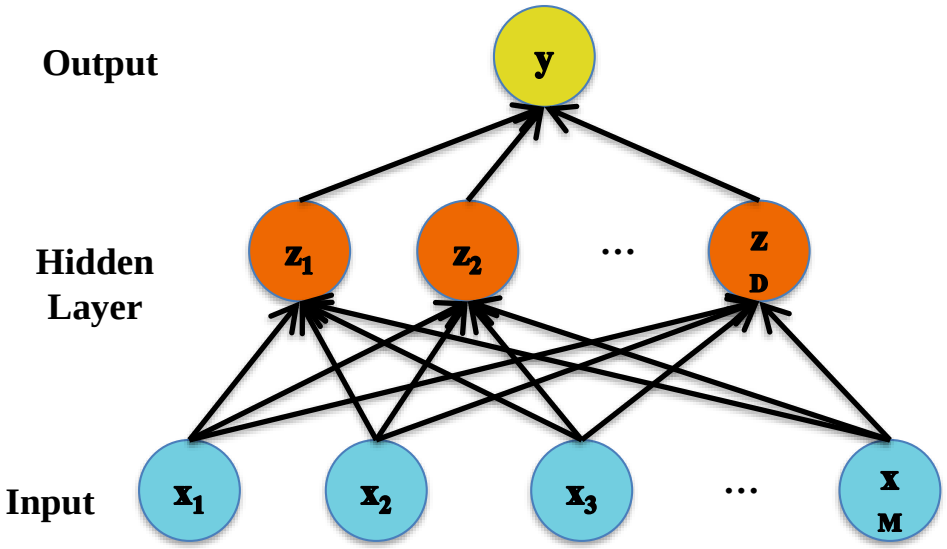
$$g(z) = \frac{1}{(1+e^{\{-z\}})}$$



- Range from [0,1].
- Not zero centered.
- Have exponential operations.
- Mostly for output layer where the output is binary.
- For the hidden layers, it is slow.
- Simplification during backpropagation.

Activation Function

Sigmoid Function (σ)



(F) **Loss**
$J = \frac{1}{2}(y - y^*)^2$

(E) **Output (sigmoid)**
$y = \frac{1}{1+\exp(-b)}$

(D) **Output (linear)**
$b = \sum_{j=0}^{D} \beta_j z_j$

(C) **Hidden (sigmoid)**
$z_j = \frac{1}{1+\exp(-a_j)}, \ \forall j$

(B) **Hidden (linear)**
$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i, \ \forall j$

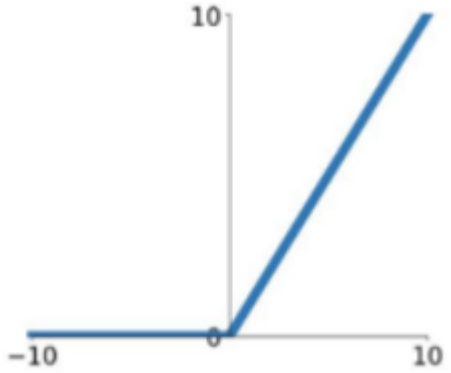
(A) **Input**
Given $x_i, \ \forall i$

Activation Function

Rectified Linear Unit (ReLU)

$$g(z) = \max\{0, z\}$$

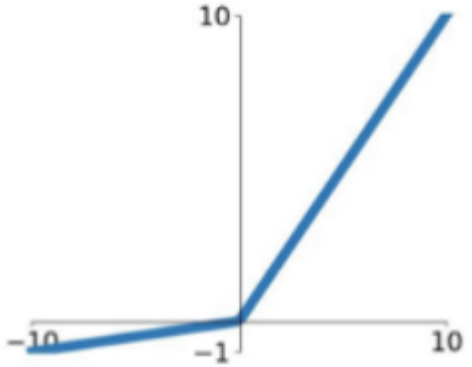
- Simple during backpropagation.
- It does not saturate.
- Converges faster than the others.
- Issue: dead ReLU.



Leaky Rectified Linear Unit

$$g(z) = \max\{a * z, z\}$$

- As an improvement over ReLU.
- Overcomes the dead problem.
- overcomes the zero gradient issue from ReLU and assigns α which is a small value for $z \leq 0$.

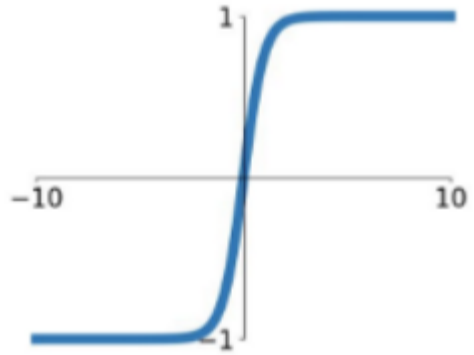


Activation Function

Hyperbolic Tangent (tanh)

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

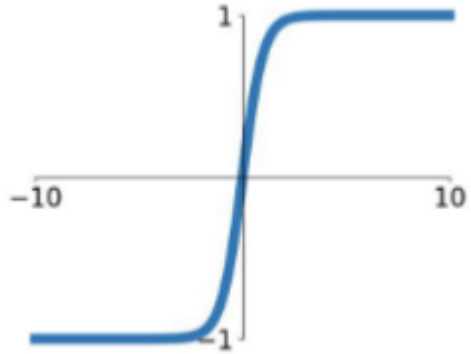
- Range between [-1,1].
- Zero centered.
- Good for the cases when input > 0.



Exponential Linear Units (ELU)

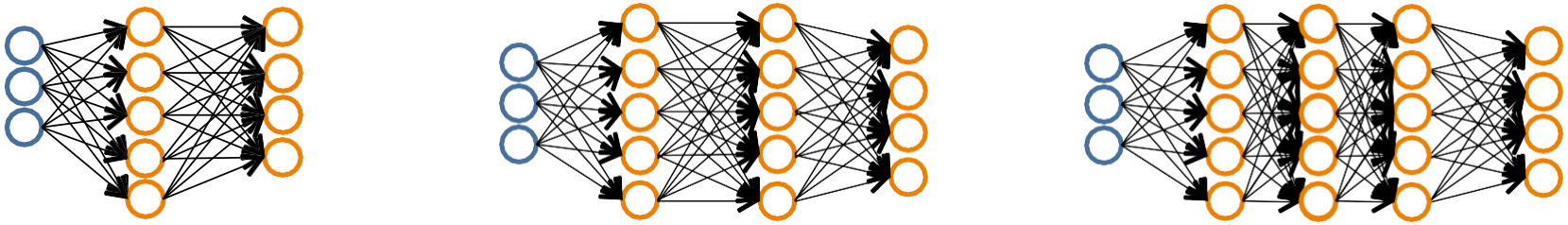
$$g(z) = t\begin{cases} x & x \geq 0 \\ \alpha(e^{x} - 1) & x < 0 \end{cases}$$

- An improvement of ReLU.
- No dead ReLU problem.
- Closer to zero mean outputs than Leaky ReLU.
- More computation because of Exponential function.



Putting it together

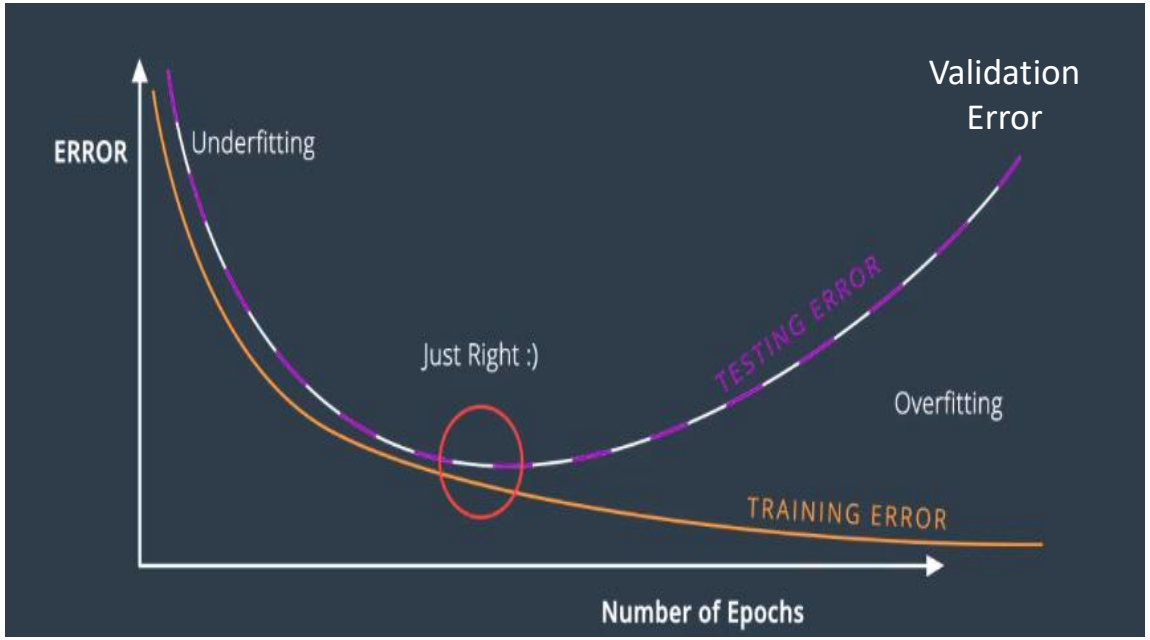
- Pick a network **architecture**

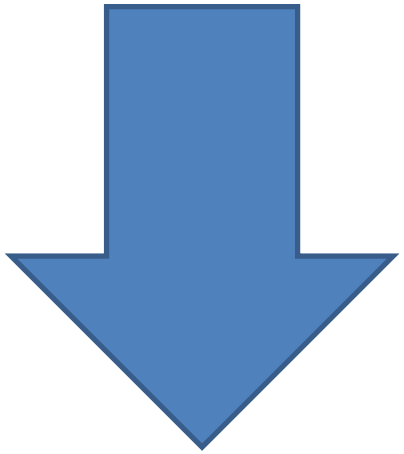


- Number of **input units**: Dimension of features
- Number **output units**: Number of classes
- Reasonable default: 1 **hidden layer**,
  or if >1 hidden layer, have same number of hidden units in every layer (usually the more the better)
- **Grid search,** a model hyperparameter optimization technique.

Putting it together

- Early stopping
- Use a **validation set** performance to select the best **configuration**
- To select the number of **epochs**, stop training when validation set error increases.





Learning in ANN

- **Learning**; the method of **modifying** the weights of connections between the neurons of a specified network.
- Goal: to find weights and biases that **minimize** the **cost function**.
- Types of Learning:
  - **Supervised Learning**
  - **Unsupervised learning**
  - **Reinforcement learning**

Learning in ANN

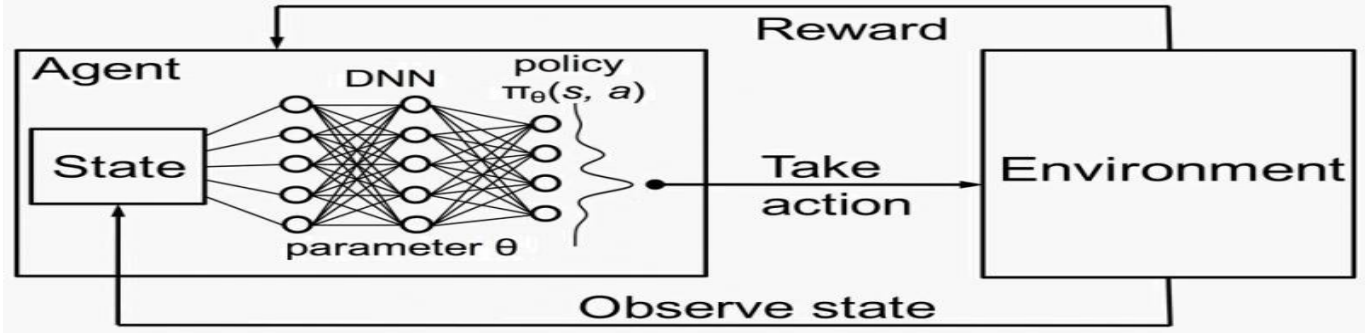
- **Supervised Learning**;
  - The **input** vector is presented to the network, which will give an output vector.
  - This **output** vector is compared with the desired output vector.
  - An **error signal** is generated, if there is a difference between the actual output and the desired output vector.
  - On the basis of this error signal, the **weights** are adjusted until the actual output is matched with the desired output.

Learning in ANN

- **Unsupervised Learning**;
  - The **input** vectors of **similar type** are combined to form **clusters**.
  - When a new input **pattern** is applied, then the neural network gives an output response indicating the **cluster** to which the input pattern belongs.
  - There is **no feedback** from the environment as to what should be the desired output and if it is correct or incorrect.
  - The network itself must discover the **patterns** and features from the input data, and the relation for the input data over the output.

Learning in ANN

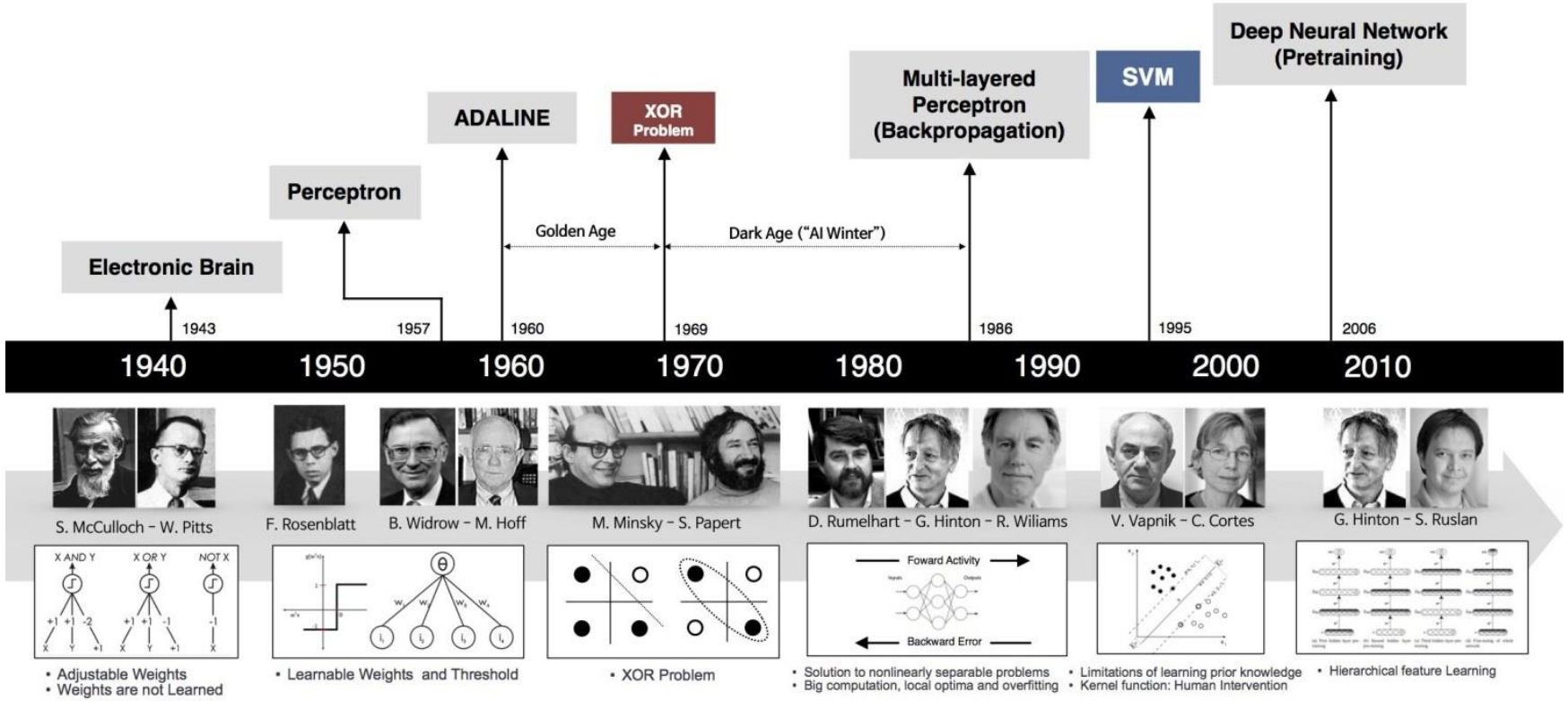
- **Reinforcement Learning**;
  - The network receives some **feedback** from the environment.
  - This makes it somewhat similar to supervised learning.
  - However, the feedback obtained here is evaluative not instructive, which means there is no teacher as in supervised learning.
  - After receiving the feedback, the network performs **adjustments** of the **weights** to get better critic information in future.



Deep Reinforcement Learning

History

- Origins: Algorithms that try to mimic the brain.
- ANN was very widely used in 80s and early 90s;
- ANN popularity diminished in late 90s.
- Recent resurgence: State-of-the-art technique for many applications



Applications

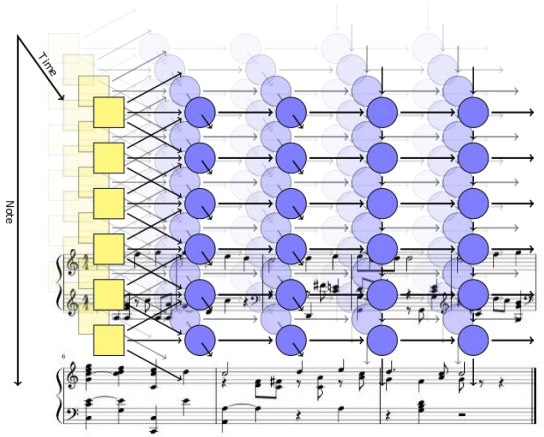
- Application areas
  - Wherever **features extraction** is required,
  - Language processing
  - Character recognition
  - Pattern recognition
  - Signal processing
  - Prediction
  - …

Applications

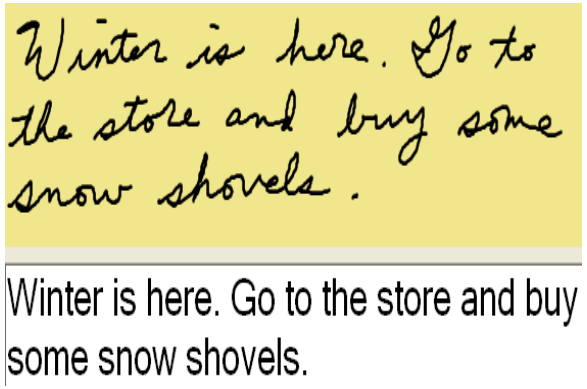
Facial Recognition

Real-time Translation

Image Compression



Music Composition



Handwriting Recognition

Stock Prediction

and many more….

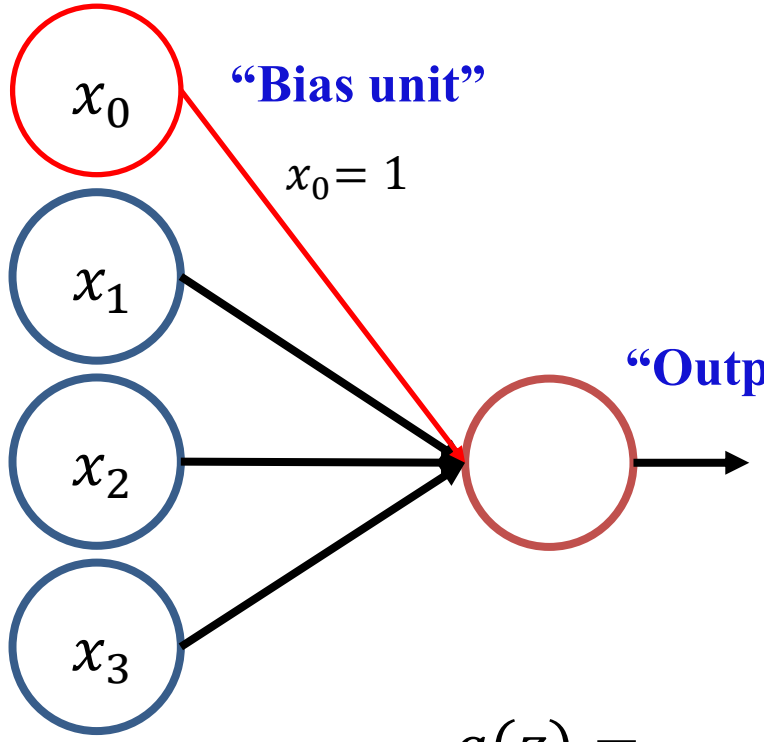
Advantages of ANN

- Nonlinearity
- Input-output mapping
- Adaptivity
- Evidential response
- Contextual information
- Fault tolerance
- VLSI implementability
- Uniformity of analysis and design
- Neurobiological analogy

An artificial neuron

Logistic unit

- Architecture of an artificial neuron:



$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

"Weights"
"Parameters"

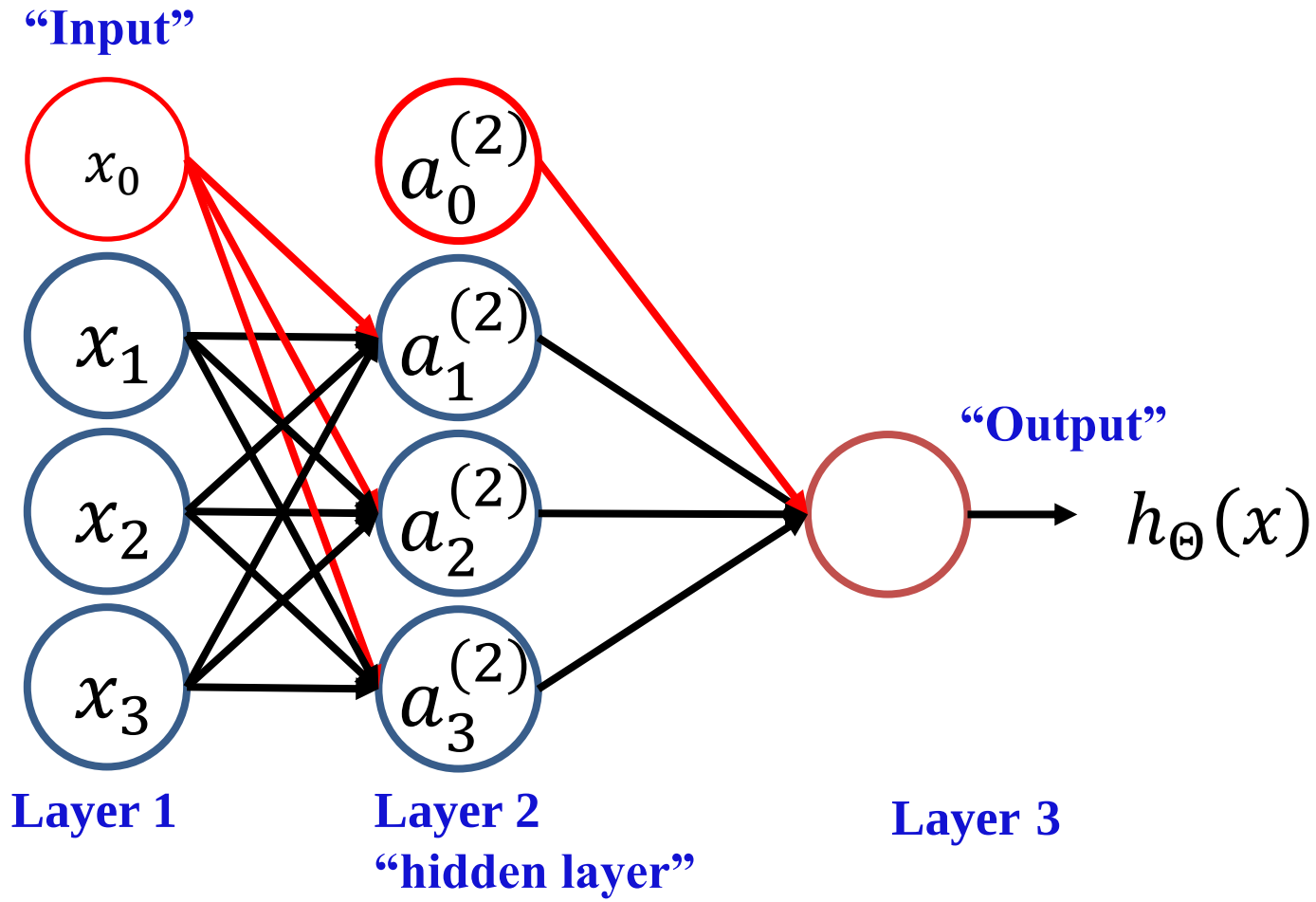
$$h_\theta(x) = \frac{1}{1 + e^{-\theta^\top x}}$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

Sigmoid (logistic) **activation function**

ANN

- An ANN composed of artificial **neurons**.



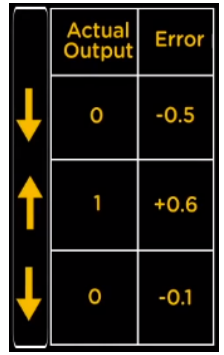
A network with only a single hidden layer is conventionally called "**shallow**".

Forward Propagation Algorithm

- The **input** is **multiplied** by the correspond **weight**.
- The **sum** of incoming values from the input layer is sent to the next **neuron** in the **hidden layer**.
- Each **neuron** is associated with a numerical value, i.e. the **bias**, which is added to the input sum.
- The result will be passed to the **activation function**.
- The result of activation function determines if the particular neuron will be **activated or not**.
- An activated neuron **transmits data** to the neurons of the **next layer** over the **channels**.
- In the **output layer**, the neuron with the **highest** value determines the **output**.
- In this manner the data are propagated through the network called "**Forward Propagation**" a.k.a. **Feed Forward Propagation**.

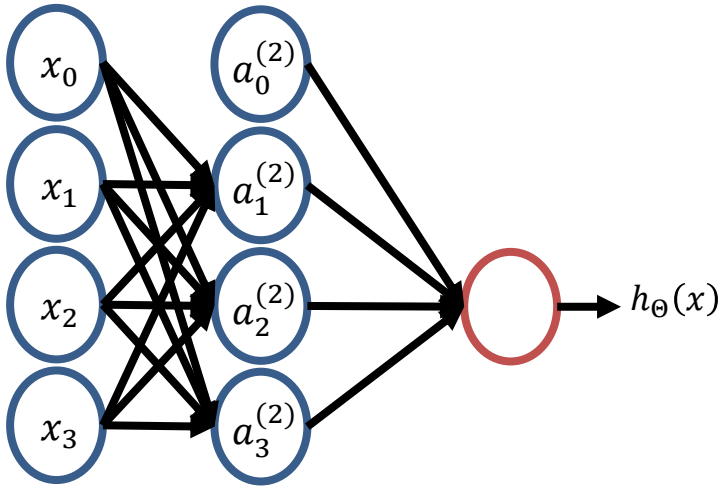
Backpropagation Algorithm

- The **predicted output** is compared with the **actual output** to realize the error e.g. the probability.
- The magnitude of **error** indicates how wrong the networks is.
- This information is then transferred **backward** through the network.
- This process is known as "**backpropagation**".
- Based on the received information, the **weights** are **adjusted**.
- The process of "**forward propagation**" and "**backpropagation**" is iteratively performed till the network can predict the output correctly in most of the cases.



- **Q**) how long it may takes time! Seconds, minutes, hours, months, …

Forward Propagation Representation



$a_i^{(j)}$ = "**activation**" of unit i in layer j

$\Theta^{(j)}$ = matrix of **weights**; controlling function mapping from layer j to layer $j+1$

Sigmoid function

$$a_1^{(2)} = g\left(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3\right)$$

$$a_2^{(2)} = g\left(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3\right)$$

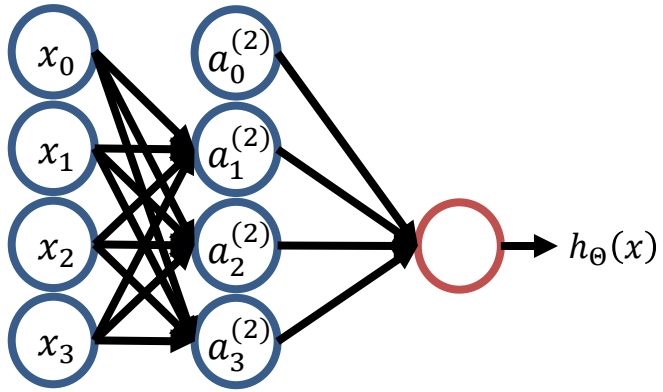
$$a_3^{(2)} = g\left(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3\right)$$

$$h_\Theta(x) = g\left(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(1)}a_1^{(2)} + \Theta_{12}^{(1)}a_2^{(2)} + \Theta_{13}^{(1)}a_3^{(2)}\right)$$

If the network has s_j units in layer j and s_{j+1} units in layer $j+1$, then $\Theta^{(j)}$ will be of dimension $s_{j+1} \times (s_j + 1)$.

Forward Propagation Representation

Vectorized Implementation



$$a_1^{(2)} = g\left(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3\right) = g(z_1^{(2)})$$

$$a_2^{(2)} = g\left(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3\right) = g(z_2^{(2)})$$

$$a_3^{(2)} = g\left(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3\right) = g(z_3^{(2)})$$

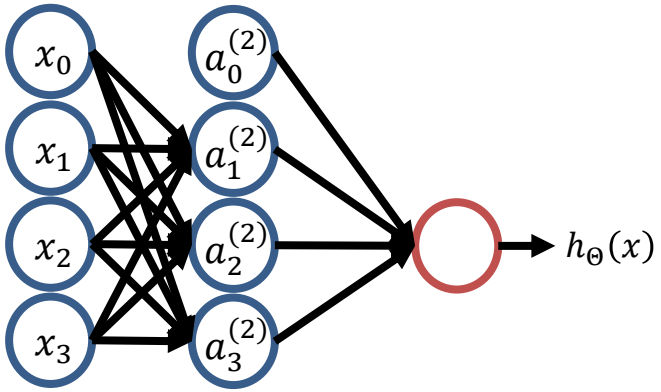
$$h_\Theta(x) = g\left(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(1)}a_1^{(2)} + \Theta_{12}^{(1)}a_2^{(2)} + \Theta_{13}^{(1)}a_3^{(2)}\right) = g(z^{(3)})$$

"Pre-activation"

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

Continued…

Forward Propagation Representation



"Pre-activation"

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

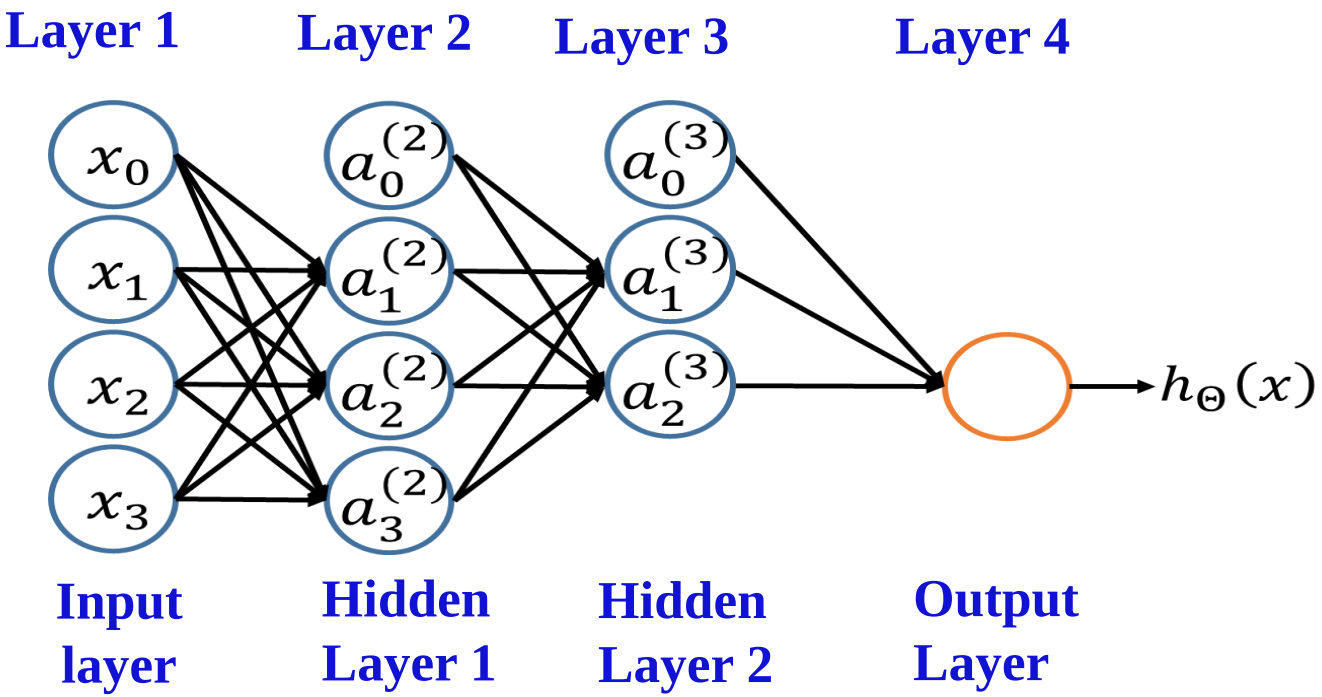
$$a_1^{(2)} = g(z_1^{(2)})$$
$$a_2^{(2)} = g(z_2^{(2)})$$
$$a_3^{(2)} = g(z_3^{(2)})$$
$$h_\Theta(x) = g(z^{(3)})$$

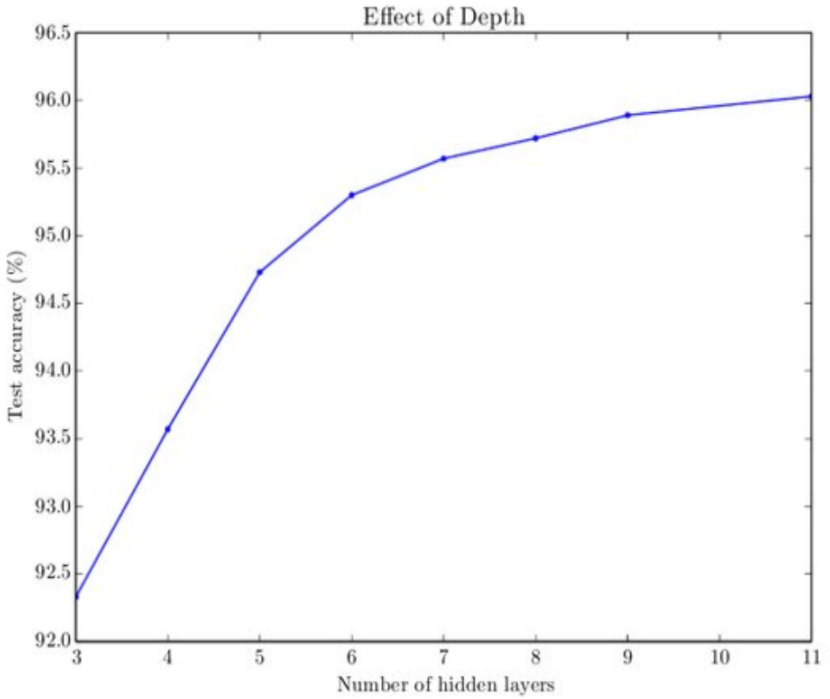
$$z^{(2)} = \Theta^{(1)} x = \Theta^{(1)} a^{(1)}$$
$$a^{(2)} = g(z^{(2)})$$

Add $a_0^{(2)} = 1$ **Bias unit**

$$z^{(3)} = \Theta^{(2)} a^{(2)}$$
$$h_\Theta(x) = a^{(3)} = g(z^{(3)})$$

Deep Neural Networks





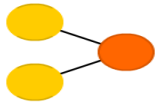
Empirical results showing that deeper networks generalize better.

An ANN having two or more hidden layers counts as "**Deep Neural Networks**".

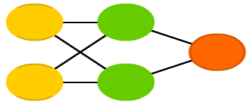
A lot of terminologies for NN!

- Backfed Input Cell
- Input Cell
- Noisy Input Cell
- Hidden Cell
- Probablistic Hidden Cell
- Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Different Memory Cell
- Kernel
- Convolution or Pool

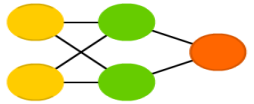
Perceptron (P)



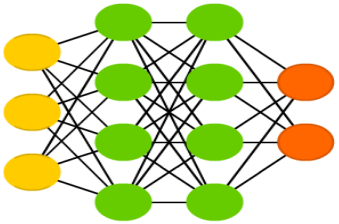
Fee Forward (FF)



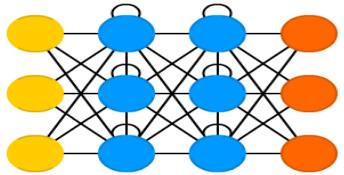
Radial Basis Network (RBF)



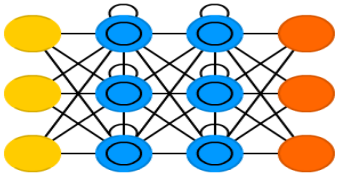
Deep Feed Forward (DDF)



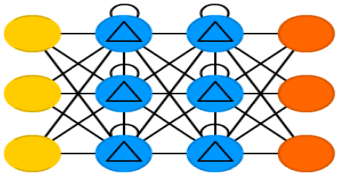
Recurrent Neural Networks



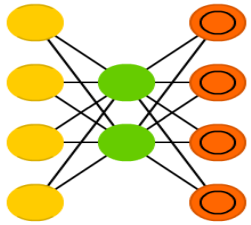
Long/Short Term Memory (LSTM)



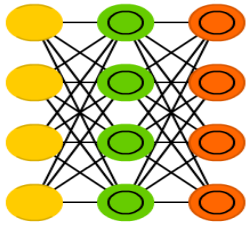
Gated Recurrent Unit (GRU)



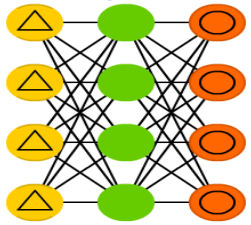
Auto Encoder (AE)



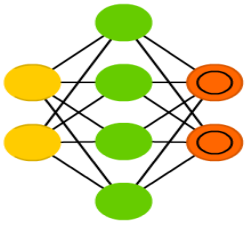
Variational AE (VAE)



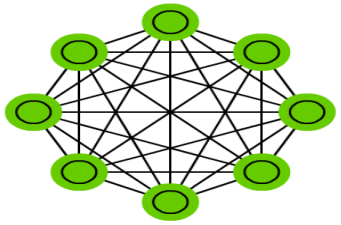
Denoising AE (DAE)



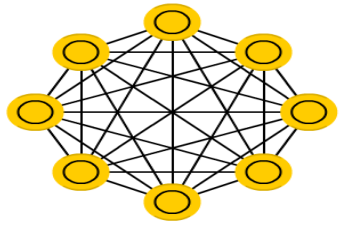
Sparse AE (SAE)



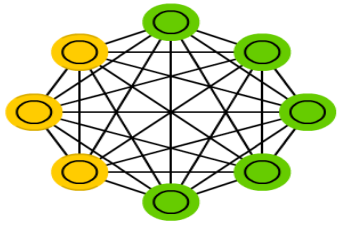
Markov Chain (MC)



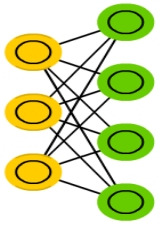
Hopfield Network (HN)



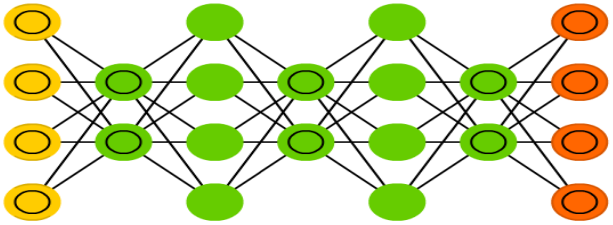
Boltzmann Machnie (BM)



Restricted BM

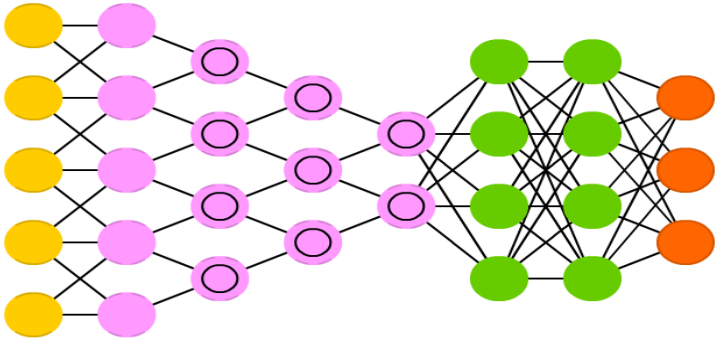


Deep Belief Network (DBN)

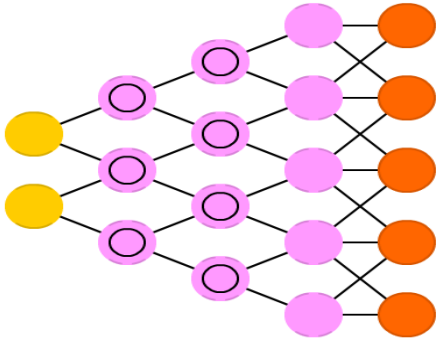


A lot of terminologies for NN!

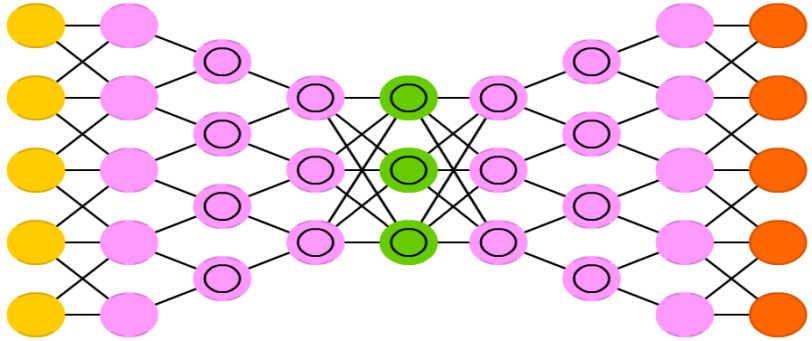
Deep Convolutional Network (DCN)



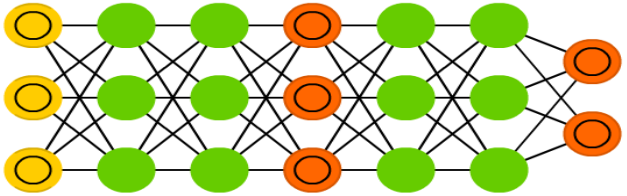
Deconvolutional Neural Networks



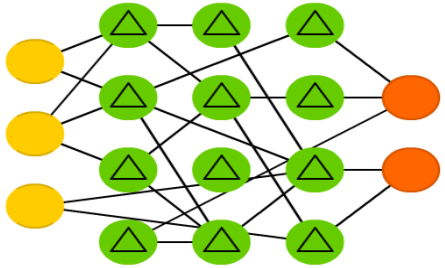
Deep Convolutional Inverse Graphics Network



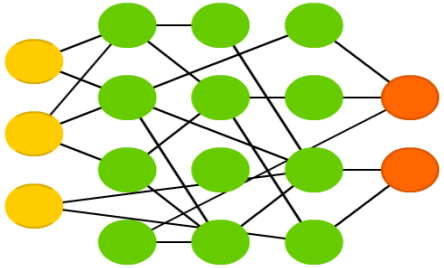
Generative Adversarial Network



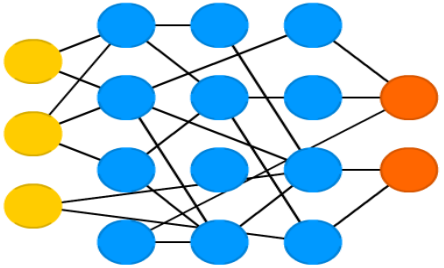
Liquid State Machine



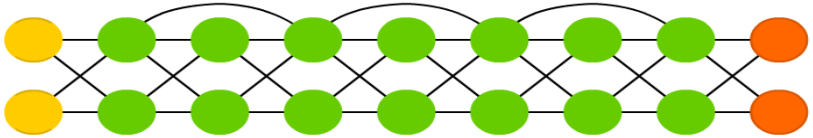
Extreme Learning Machine



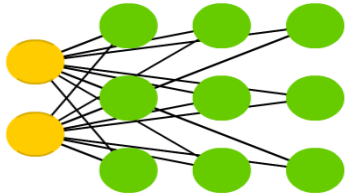
Echo State Network



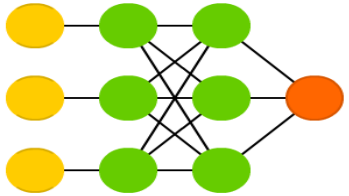
Deep Residual Network



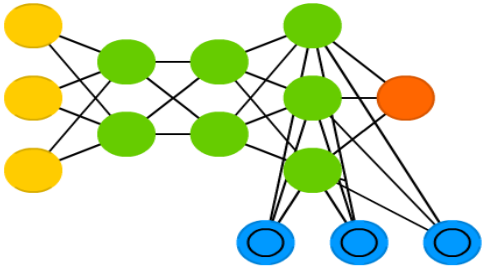
Kohonen Network



Support Vector Machine



Neural Turing Machine



Office: #432, Dayang AI Center, **Phone**: 010-8999-8586, **email**: *piran@sejong.ac.kr*