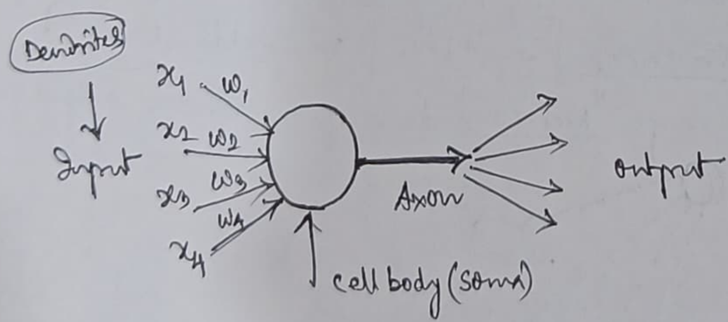
Artificial Neural Network (ANN)

- ANN
  - imitates human brain behaviour
  - neurons in human brain are connected as directed graph network.
  - neuron are the processing units (collects data, process and tranfers to the next neuron).
  - used to solve non-linear and complex problem.
  - neurons are working in parallel
  - applied in NLP, pattern recognition, face recognition, speech recognition, character recognition, text processing, stock prediction, computer vision etc.

- Artificial Neurons
- are like biological neurons called as nodes.
- nodes can receive one or more information and process it.
- nodes are connected with connection links. The links are associated with synaptic weight.



- Model of Artificial Neuron

Steps ⟹
① receives weighted inputs from other neurons
② operates with a threshold function or activation function.

Let, inputs are $= [x_1, x_2, \dots, x_n]$
weights associated are $= [w_1, w_2, \dots, w_n]$

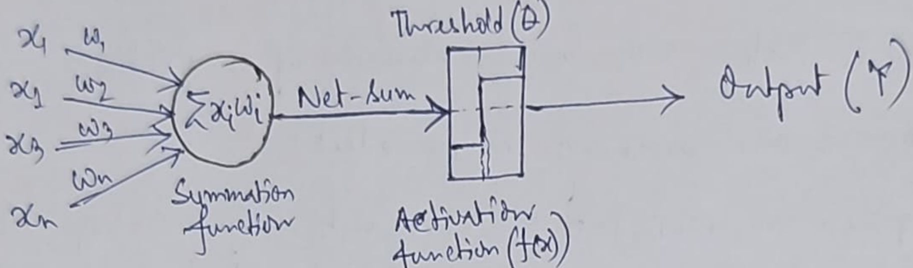
$$\boxed{\text{Net-sum} = \sum_{i=1}^{n} x_i w_i}$$

ANN cont.

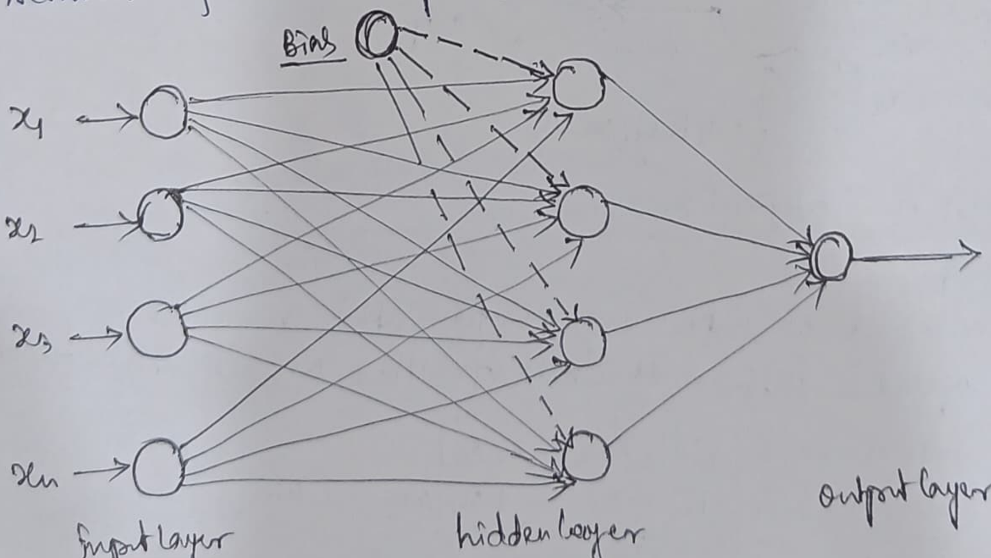
- Activation function is a binary step function which outputs a value '1' if the Net-sum $\geq$ threshold value (θ) and '0' if Net-sum $<$ threshold value (θ). So activation function applied on 'Net-sum'.

$$f(x) = \text{Activation function (Net-Sum)}.$$

So, output of neuron $Y = \begin{cases} 1 & \text{if } f(x) \geq \theta \\ 0 & \text{if } f(x) < \theta \end{cases}$



- **ANN Structure**
- neuron nodes are connected through edges and can work in parallel.
- ANN has three layer: Input Layer, Hidden Layer, Output Layer.
- each neuron collects inputs with associated weights and process.
- each neuron employs an activation function which determines output of the neuron. If net-sum $>$ threshold $\Rightarrow$ Neuron Fires.
- Neuron transforms linearly $\left[\sum_{i=1}^{n} x_i w_i\right]$ and adds biases.
- Activation function maps "Net-sum" to a non-linear output value.



Activation Function

- This mathematical function associated with each neuron, and map input signals to output signals.
- This function normalize output value of each neuron either between (0 and 1) or between (−1 and +1).
- This function can be linear or non-linear.

Linear Activation Fun^n

- This is useful when the input values are classified in any ~~one~~ of two groups and used in binary perceptron.

Non-Linear Function (Activation)

- This is continuous function, map the input in the range of (0,1) or (−1,1) etc.
- Useful in learning high-dimensional data or complex data like audio, video, images.

Activation Function in ANN

① Identity Function or Linear Function: $f(x) = x \ \forall x$

- $f(x)$ increases linearly/proportionally with x.
- useful when threshold is not applied.
- output is $\sum_{i=1}^{n} x_i w_i$ and ranges $-\alpha$ to $+\alpha$.

② Binary Step Function:

$$f(x) = \begin{cases} 1 & f(x) \geq \theta \\ 0 & f(x) < \theta \end{cases}$$

- output value is binary (0,1) w.r.t. 'θ'.

③ Bipolar Step function

$$f(x) = \begin{cases} 1 & f(x) \geq \theta \\ -1 & f(x) < \theta \end{cases}$$

④ Sigmoidal Function or Logistic Function

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

- widely used non-linear activation function.
- produced 'S' shaped curve with range 0 to 1.

(✱) (✱) it has vanishing gradient problem ⇒ no change in prediction for very low I/P value or very high input value.

(5) Bipolar Sigmoid function

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

value ranges (-1 to +1)

(6) Ramp function

- Linear function
- upper and lower limits are fixed.

$$f(x) = \begin{cases} 1 & x > 1 \\ x & 0 \leq x \leq 1 \\ 0 & x < 0 \end{cases}$$

(7) Tanh – Hyperbolic Tangent function

- scaled version of Sigmoid fun^n.
- non-linear
- (*) suffers from vanishing gradient problem.
- output value ranges (-1 to 1)

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

(*) (8) ReLu – Rectified Linear Unit Function

- used in Deep Learning Neural N/w model in Hidden Layer
- (*) avoids or reduces vanishing gradient problem.
- gives o/p '0' for ⊖ i/p value.
- works linear function like for ⊕ i/p value.

$$r(x) = \max(0,x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

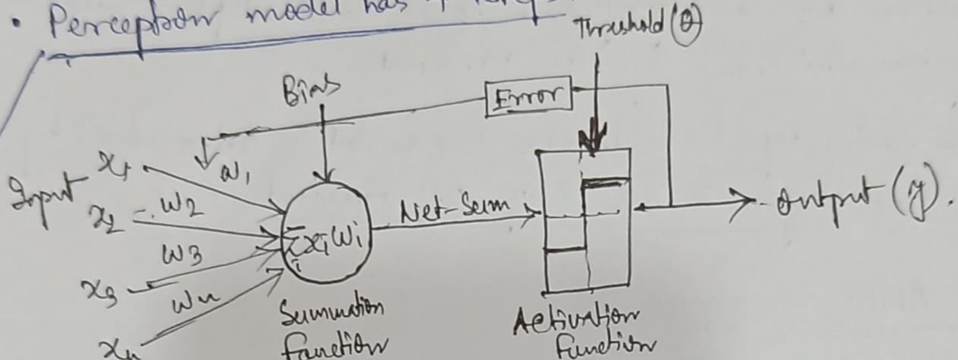
(9) Softmax Function

- non-linear function
- used in o/p layer
- handle multiple classes

$$S(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{k} e^{x_j}}$$ where $i = 0 \ldots k$

Perceptron and Learning Theory

- Perceptron is the first Neural Network (NN) model.
- is a Linear binary classifier.
- used for supervised Learning.
- it contains concept of "Artificial Neuron" and "Learning rule of adjusting weights" and extra input "bias"
- Artificial Neuron Learn from data the correct weights from variable weight values as a part of supervised learning.
- Perceptron model has 4 steps:



① **Inputs and Weights**

Inputs $(x_1, x_2, \dots x_n)$ with associated weights $(w_1, w_2, \dots w_n)$

$$\text{Net-Sum} = \sum_{i=1}^{n} x_i w_i$$

② **Bias**

$$f(x) = \text{Activation function } (\text{Net-Sum} + \text{bias})$$

③ **Net-Sum** : Sum of $x_i w_i$

④ **Activation function** :

$$y = \begin{cases} 1 & f(x) \geq \theta \\ 0 & f(x) < \theta \end{cases}$$

Error Calculation in Perceptron

$\boxed{\text{Sums Practice}}$

$$\text{error } e(t) = Y_{desired} - Y_{estimated}$$

if $e(t) = (+)$, increase Y
$= (-)$, decrease Y

then **update weights**

$$\Delta w_i = \alpha \times e(t) \times x_i$$
$$w_i = w_i + \Delta w_i$$

x_i = input value,
$e(t)$ = error at step t

α = Learning rate
Δw_i = difference in weight that has to be added to w_i

Delta Learning Rule and Gradient Descent

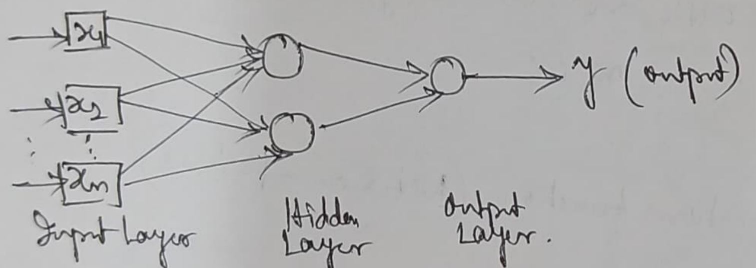
- Learning in NN does by adjusting N/w weights to minimize the difference between desired and estimated outputs.
  → this difference is error function (Cost Function).
- Cost function is linear, continuous and differentiable.

$$\text{Training error} = \frac{1}{2} \sum_{d \in T} \left(O_{Desired} - O_{Estimated} \right)^2$$

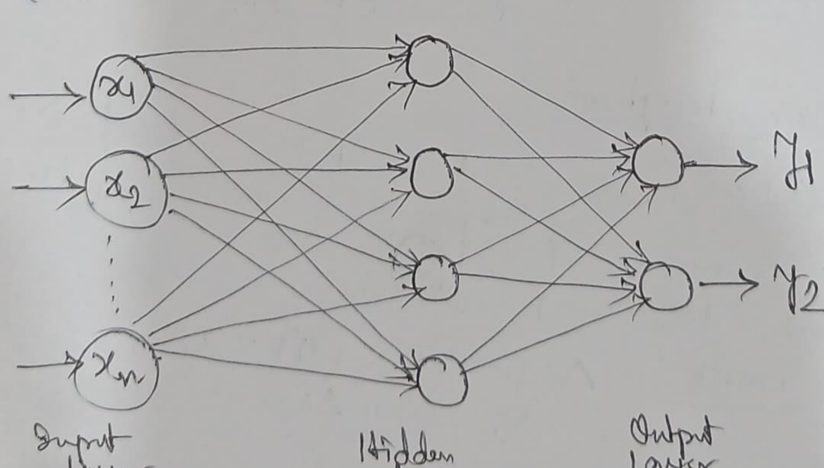
- Gradient Descent is an optimization approach used to minimize Cost function by converging to a local minimal point moving in the negative direction of the gradient.
  ↳ each step size during movement is determined by <u>Learning rate and slope of gradient</u>.

Types of ANN

① Feed Forward NN



② Fully Connected NN



ALgo
SUMS
NI

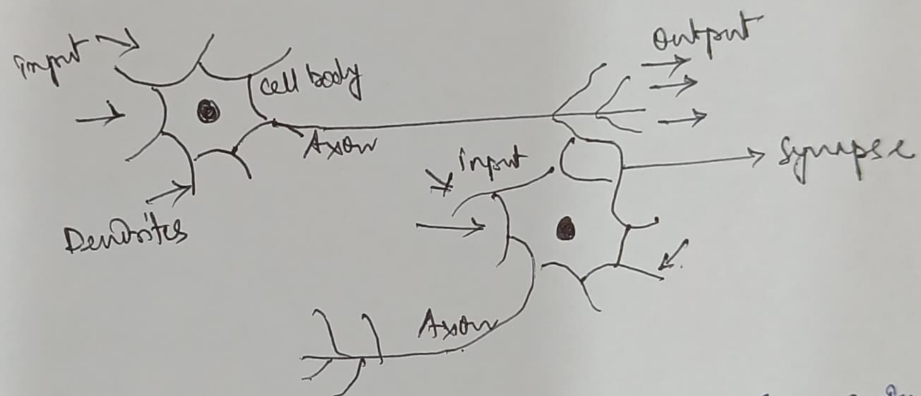
3 Multi Layer Perceptron (MLP)

- For ANN structure, if o/p is incorrect, then in backward direction, error is back propagated to adjust weights and biases to get correct o/p.
  - → N/W learns with training data.
  - → This ANN used in Deep learning.
  - → $\boxed{\text{Multiple Hidden Layers}}$ (Multi-Layer Perceptron) MLP

4 Feedback Neural N/W

O/P signals can be sent back to neurons in the same layer or to the neurons in the preceding layers.

Biological Neuron



- Typical biological neuron has 4 parts → dendrites, Soma, axon, synapse.
- The body of the neuron is called Soma.
- Dendrites accept i/p info^n and process in Soma.
- A single neuron is connected by axons to around 10,000 neurons and through these axons information is passed from one neuron to another neuron.
- If i/p information > threshold, neuron gets fired and transmits signal to another neuron through synapse.
- A synapse gets fired with an electrical impulse (spikes) and are transmitted to another neuron.
- A single neuron can receive synaptic i/ps from one neuron or multiple neuron.

⑧ ANN Cont.

- Advantages of ANN

① can solve complex problems involving non-linear processes.

② Can learn and recognize complex patterns and solve problems as humans solve problem

③ have a parallel processing capability and can predict in less time.

④ have an ability to work with inadequate knowledge.
Can handle incomplete and noisy data.

⑤ Can scale well to larger data sets and outperforms other learning mechanisms.

DEEP LEARNING

- DL is extension of ANN.
- NN with two or more hidden layers are called DNN.

- Shallow NN

NN with only one hidden layer.

- Deep NN
- NN with two or more hidden layers, one i/p, one o/p layer
- It can extract features automatically.
- It create higher abstraction of i/p at every layer.

- $$\text{Net sum } (u) = \sum_{i=1}^{n} x_i w_i + b$$
$$y = f(u)$$

Loss Function

Measured with MSE, Likelihood Loss, Log Loss or Cross Entropy Loss.

$\hat{y}$ = predicted value, y = actual value.

$\boxed{\text{MSE}}$ $$J = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y)^2$$

N = total number of neurons.

Error more ⇒ value of loss function is more.

$\boxed{\text{Likelihood Loss}}$
- used for classification.
- computed as multiplication of predicted probabilities of i/ps.

$\boxed{\text{Log Loss or Cross Entropy Loss}}$

$$E = -\left(y \log(p) + (1-y)\log(1-p)\right)$$ [for two classes]

$$E = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$ [more than 2 classes]

M is the number of classes,

p is the predicted probability of observation 'o' of class 'c'.

output is in the range (0, 1).

(2) DL Contd.

Regularization

- Overfitting is a major problem in DL.
  → performing well on training data but poorly on test data.
  → Lack of Generalization
- opposite to overfitting is underfitting (is generalization) but model failed to understand basic underlying relationship.
- Regularization aim to reduce overfitting in DL models.
  → combines loss functions and regularization constant.

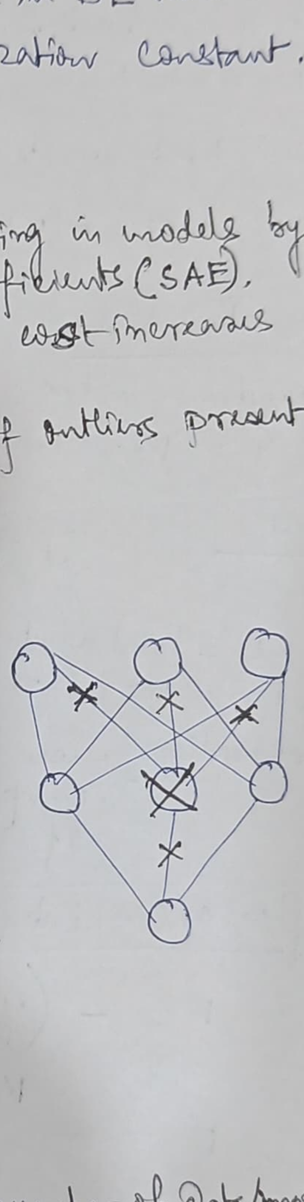
Regularization schemes

① **L1 & L2 regularization:**

- L1 regularization technique reduces overfitting in models by penalizing absolute size of regression coefficients (SAE).
- L1 takes absolute values of weights, so cost increases linearly.
- L2 takes square of weights, so cost of outliers present in the data increases exponentially.
- L2 also reduces overfitting by using (SSE).

② **Dropout Regularization:**

- used to solve overfitting
- remove neuron randomly and continue to training pass.
- drop same neuron in forward and back propagation, but enabled during testing.



(★) → It forces DNN to learn alternative or redundant representations.

(★) → Dropout of 0.5% is used.

③ **Data Augmentation**

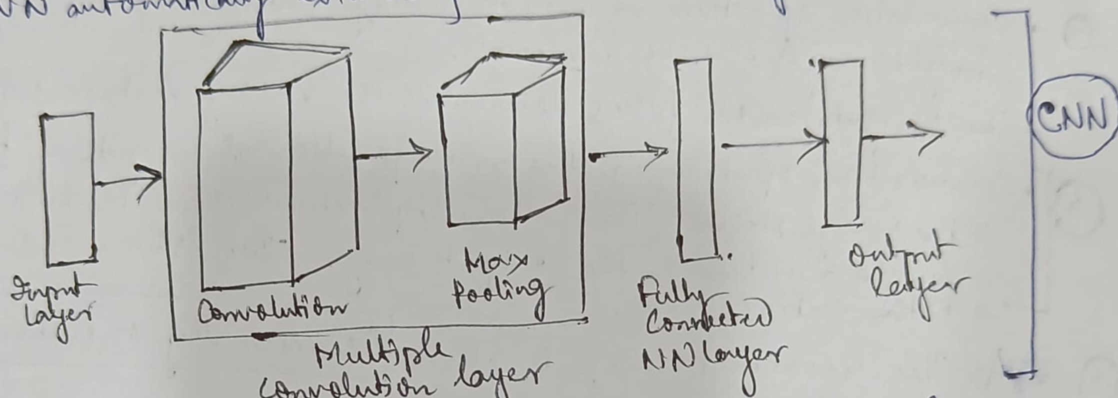
- Aim is to increase dataset by increasing number of data/images.
- For image, number can be increased by translation, rotation, scaling.
  Translation (shifting image in horizontal/vertical directions).
  Rotation (clock wise / anticlockwise).
  scaling (enlarging / shrinking).
- Transformation (mirroring, cropping etc).

4 Early stopping

5 Adding Noise

CNN (Convolutional NN)

- CNN are multilayer NN.
- used to recognize visual patterns directly from images.
- CNN performs image classification → taking image as I/P (array of numbers) and specifying class as O/P. (probability)
- CNN automatically extracts features like edges, curves.



- Major layers of CNN are convolutional layer, pooling layer, ReLU layer, fully connected layer.

WI
- Input layer is 2 Dimensional.
- Output is number of classes → So it is 1-Dimensional.

- It provides hierarchical learning.

LeNet CNN Architecture

- CNN can be implemented in many ways.
- for this architecture steps / layers can be:

Input layer → Convolutional layer → Average pooling → Convolutional layer → Average pooling → Fully Connected Network → output layer.

④ DNN Contd.

Input Layer

- Input for CNN is image.
- Image is 2D signal, varies over spatial coordinate $x, y \Rightarrow f(x,y)$.

Convolutional layer

- Convolution / CONVNET is implemented here.
- It has set of filters known as MASKS/KERNELS.
- Filters are convolved with i/p to give activation map/feature map.
- This layer examines small area using filters.
  (receptive field)
- Filters are small matrix (3x3 or 5x5 or 7x7 size) with weights or parameters.
- Convolution operation is used to identify features of the image like straight edges, curves, colours.
- The weights in the filters determine the kind of feature extracted.
- Convolution operation is the multiplication of weights of filters with the image pixels. Then all multiplications are summed to give a single number in the output image. Then filter moves throughout the entire image.

⟶ The resultant image is called $\boxed{\text{Activation Map or Feature Map}}$

Parameters of Convolutional operation

① Number of Filters: It is called depth.

② Filter size: 3x3 or 5x5 or 7x7

③ Stride: It is an hyperparameter, defines the shifts of filters. Default value of it is 1 pixel, but can be 2/3/4 pixels. with more stride value, overlapping pixels can be avoided, speed can be enhanced, but many image regions may be skipped.

Example i/p data [1,3,3,2,1,0,1], filter [1,1,1]

Stride = 1

| 1 | 3 | 3 | 2 | 1 | 0 | 1 |

| 1 | 1 | 1 | → (7)
   | 1 | 1 | 1 | → (8)
      | 1 | 1 | 1 | → (6)
         | 1 | 1 | 1 | → (3)
            | 1 | 1 | 1 | → (2)

Stride = 2

| 1 | 3 | 3 | 2 | 1 | 0 | 1 |

| 1 | 1 | 1 | → (7)
      | 1 | 1 | 1 | → (6)
            | 1 | 1 | 1 | → (2)

④ **Padding**: another hyperparameter, used to reduce the spatial volume.
But to preserve much info^n 'zero' can be padded and to have o/p image dimension same as input/image.

1	2
3	4

one zero padding →

0	0	0	0
0	1	2	0
0	3	4	0
0	0	0	0

two-zero padding →

0	0	0	0	0	0
0	0	0	0	0	0
0	0	1	2	0	0
0	0	3	4	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Formula For Activation Map/Feature Map

$$\text{Activation Map/Feature Map} = \frac{D-F+2P}{S} + 1$$

D = Dimension of image, F = Filter size, P = amount of padding,
S = Stride Length.

Q1 Image size = 28×28, filter size = 5×5, Stride = 1, padding is zero, what is the size of feature map?

⇒ $D=28, F=5, S=1, P=0.$

$$\text{Activation/Feature Map} = \frac{28-5+2\times 0}{1} + 1 = 24.$$

So the size of feature map = 24×24.

Q2 Consider the Data & mask, find results of Convolution process.

1	2	3	4
5	7	8	9
10	11	12	11
8	6	4	3

Data

1	0
0	1

mark

⇒

1	2		
5	7		

⇒

8	?	?
?	?	?
?	?	?

steps will follow with stride or sliding window concept.

← find → (by processing all locations.)

8	10	12
16	19	19
16	15	15

← Final feature Map.

⑥ DNN (contd)

RELU Layer

- After each convolution layer, an activation layer or RELU layer is used to introduce non-linearity.
- Activation like sigmoid, tanh can be used.
- RELU is good, because it alleviate problem of vanishing gradients.
- RELU uses $f(x) = \max(0, x)$ to all inputs.
  All ⊖ (negative) activations are reduced to '0'.

Pooling Layer / Down Sampling layer

- used to reduce spatial dimension of I/P.
- normally pooling mask size is 2×2.
- pooling layer has no parameters.

$$\left[\begin{array}{cc|cc} 10 & 11 & 12 & 13 \\ 3 & 4 & 6 & 7 \\ \hline 8 & 9 & 10 & 11 \\ 11 & 12 & 13 & 14 \end{array}\right] \xrightarrow{\text{average pooling}} \begin{array}{|c|c|} \hline 7 & 10 \\ \hline 10 & 12 \\ \hline \end{array}$$

$$\xrightarrow{\text{max pooling}} \begin{array}{|c|c|} \hline 11 & 13 \\ \hline 12 & 14 \\ \hline \end{array}$$

Dropout Layers

Solves the problem of overfitting.

Fully Connected Layer and Output Layer

- Transfer Learning.
- Application of DL

Robotic Control, Classification, Parameter estimation, State estimation, Data Mining, Autonomous Navigation, Bioinformatics, Speech Recognition, Text Analysis.