MODULE 5

FEATURE ENGINEERING

- Variable-length data refers to data where the number of elements or observations within a feature can vary from one instance to another. This is in contrast to fixed-length data, where each feature has a consistent and predetermined number of elements.

- Variable-length data is common in various types of datasets, and handling it effectively is an important aspect of feature engineering. Here are some examples to illustrate variable-length data:

Text Data:

In natural language processing (NLP), the length of a text document varies. Feature engineering for text data involves converting variable-length sequences of words into fixed-length representations, such as using techniques like tokenization, padding, or embeddings.

Time Series Data:

Time series data often involves sequences of varying lengths. For example, stock prices recorded over different time intervals may result in variable-length time series data. Feature engineering may involve transforming these time series into fixed-length features using methods like rolling averages, moving windows, or statistical summaries.

Sequential Data:

Sequences of events or actions in data can have variable lengths. For instance, a sequence of user interactions on a website may vary in length for different users. Feature engineering in this context may involve creating fixed-length representations or summarizing the sequence in some way.

Sensor Data:

Data from sensors, especially in IoT (Internet of Things) applications, can have variable lengths depending on when and how frequently data is collected. Feature engineering for sensor data may involve aggregating or resampling the data to create fixed-length features.

Handling variable-length data in feature engineering often requires careful consideration of how to represent and process such data. Here are some common approaches:

- **Padding:** Adding zeros or some placeholder values to sequences to make them uniform in length.
- **Truncation:** Trimming sequences to a specified length.
- **Aggregation:** Summarizing variable-length sequences into fixed-length features, such as calculating means, medians, or other statistics.
- **Embeddings:** Transforming variable-length categorical data into fixed-length vector representations using techniques like word embeddings for text data.

Instance Based Engineering

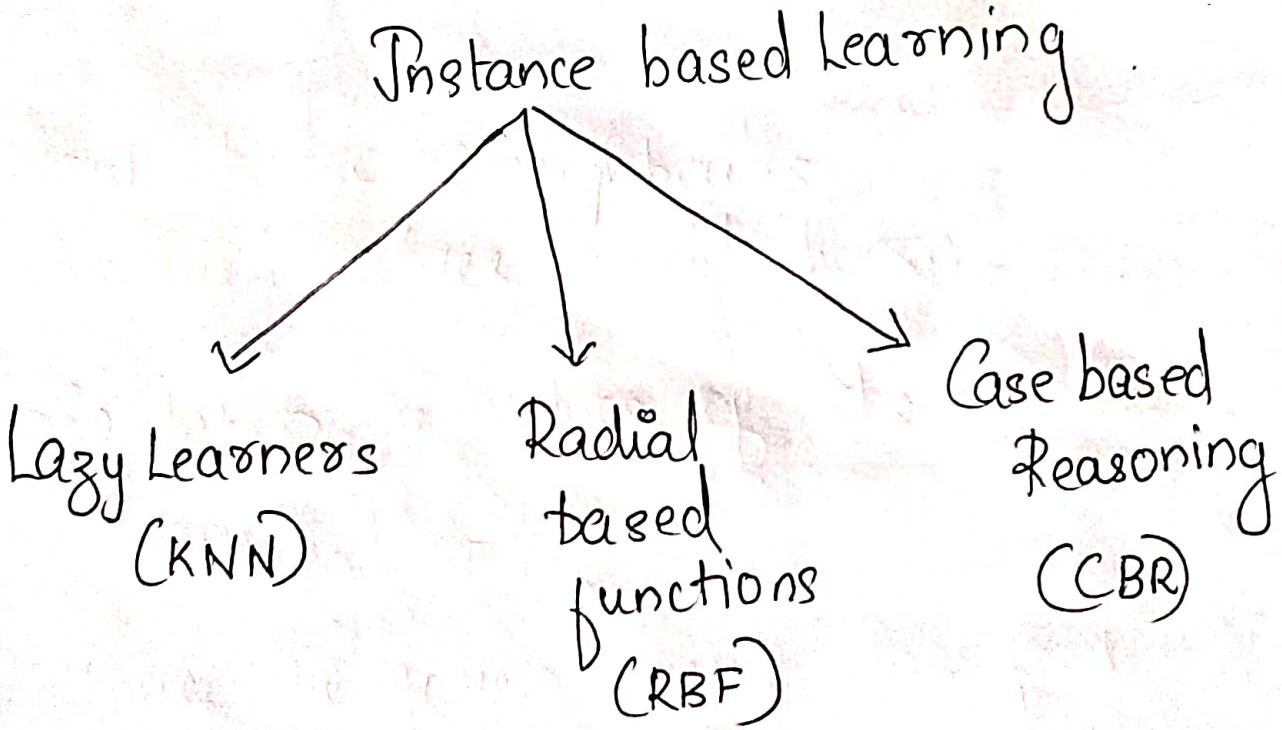
→ Memorize and then apply.

→ Instead of performing explicit generalisation, it compares new problems with instances in training, which are stored in memory.

Example:- Spam Mails → Spamfilter

→ Also called as memory based learning / Lazy learning.

→ Done with 3 different approaches.



KNN . K-Nearest neighbour algm

→ Example for lazy learning

Example .

Given data Query ⇒ x = (Maths = 6, CS = 8) and K = 3.

Classification depends on pass/fail on basis of the details of 5 students are given below:-

	Maths	CS	Result
1)	4	3	F
2)	6	7	P
3)	7	8	P
4)	5	5	F
5)	8	8	P

Euclidean distance (d)

$$d = \sqrt{|x_{O_1} - x_{A_1}|^2 + |x_{O_2} - x_{A_2}|^2}$$

O = observed value

A = Actual value

1) Calculate $d_1 = \sqrt{(6-4)^2 + (8-3)^2}$

$= \sqrt{2^2+5^2} = \sqrt{29} = 5.38$

2) $d_2 = \sqrt{(6-6)^2 + (8-7)^2} = \sqrt{0+1} = 1$

3) $d_3 = \sqrt{(6-7)^2 + (8-8)^2} = \sqrt{1+0} = 1$

4) $d_4 = \sqrt{(6-5)^2 + (8-5)^2} = \sqrt{1+9} = \sqrt{10}$

$= 3.16$

5) $d_5 = \sqrt{(6-8)^2 + (8-8)^2} = \sqrt{4+0} = \sqrt{4}$

$= 2$

Now choose 3 neighbours.

3 nearest values should choose

So taken the values of d_2, d_3 and d_5

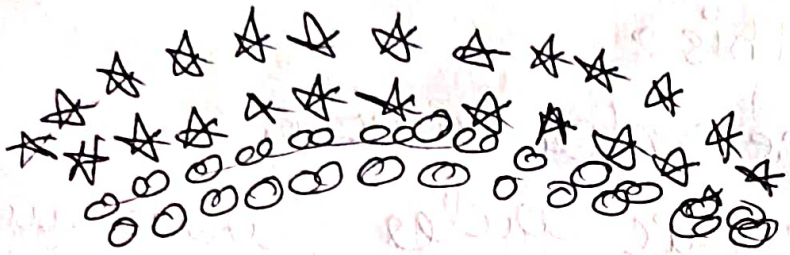
(3P) and (0F)

Thus the given query classified as pass

Radial Basis functions

→ used in ANN
→ has only one hidden nodes.

example



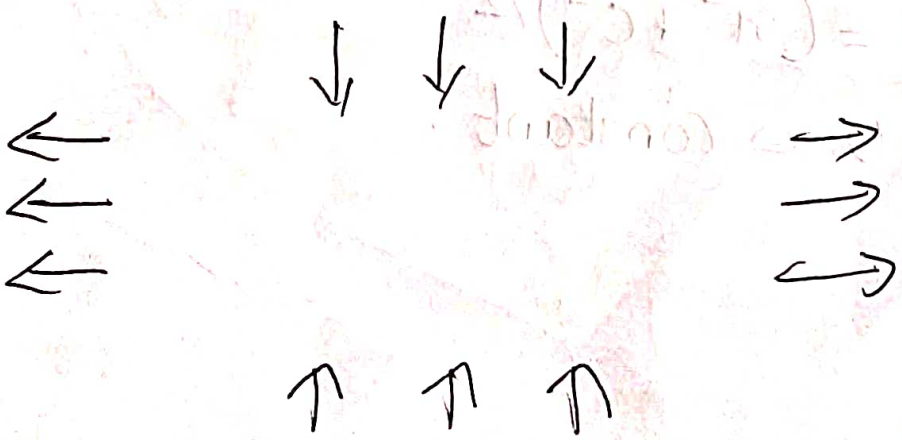
→ 2 classes (star & circle)

→ data is not linearly sepesable

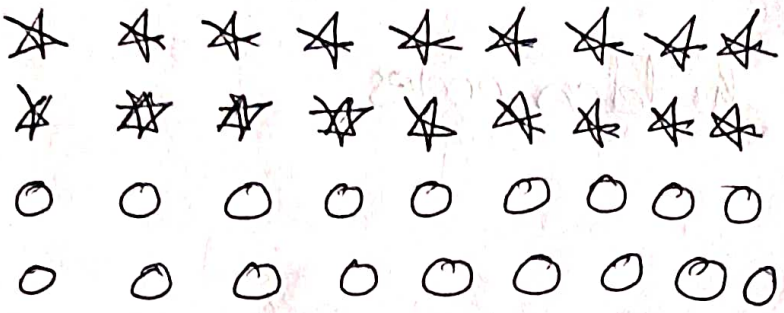
→ 2 steps

1. Increase the dimensionality (2D - 3D) but this step is not mandatory, only based on the requirement.

2. Expand the direction (Horizontal)
Compress the direction (Vertical)



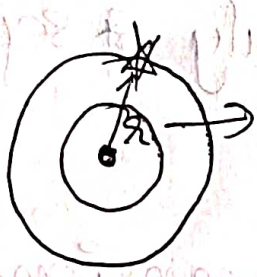
Resultant dataset :



How RBF does this?

→ Consider one center randomly

→ draw cocentric circles [circles with same units]



To expand/compress, we use 3 functions.

1) Multiquadric

$$\phi(r) = (r^2 + c^2)^{1/2}$$

$c > 0 \Rightarrow$ constant.

2) Inverse multiquadric

$$\phi(r) = \frac{1}{(r^2 + c^2)^{1/2}}$$

3) Gaussian function

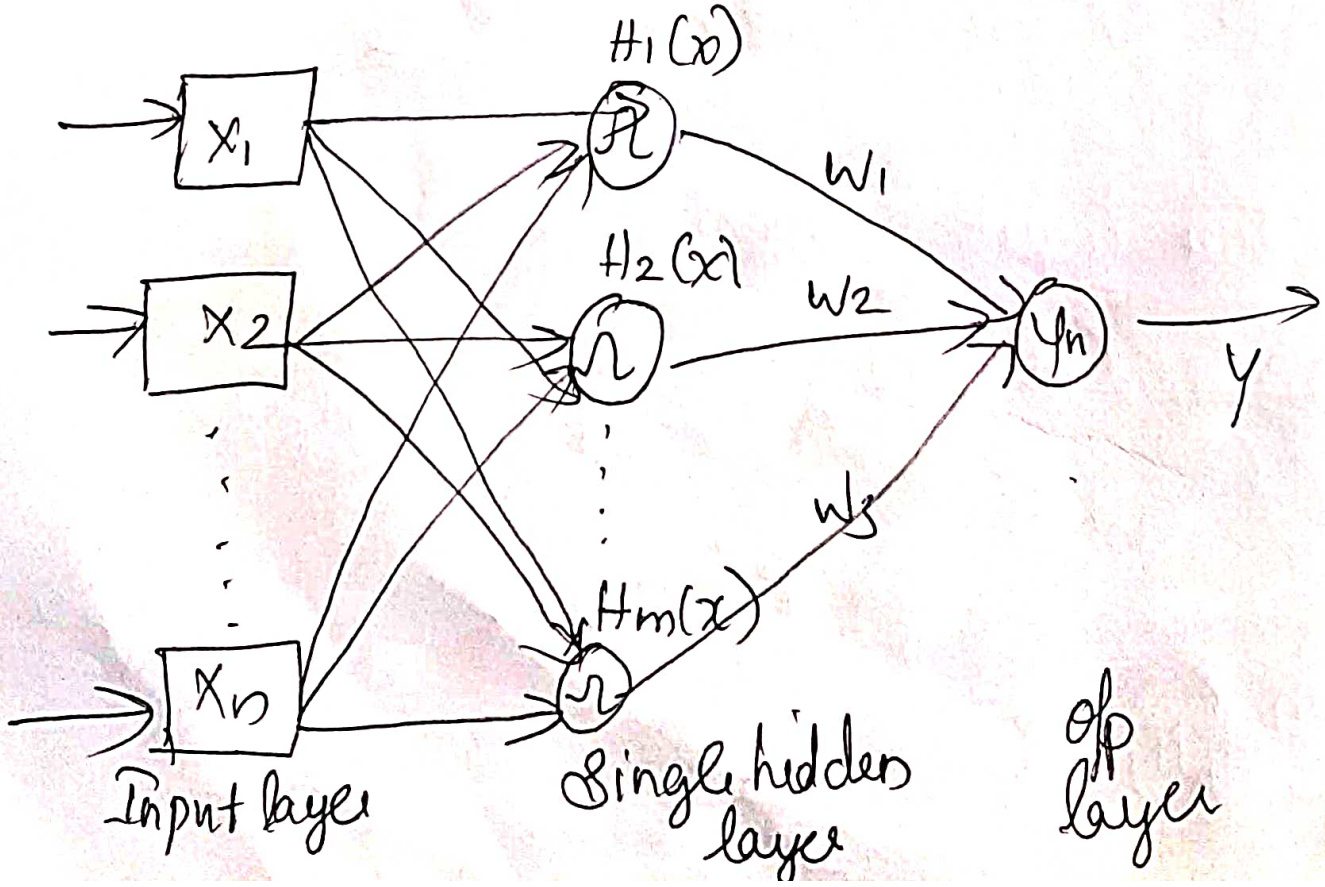
$$\phi(r) = \exp\left[\frac{-r^2}{2\sigma^2}\right]$$

σ is a constant, which is always > 0.

Radial Basis function in Neural n/w

→ Radial Basis function is a type of Multilayer Perceptron which has one input layer, one output layer and with strictly one hidden layer.

→ The hidden layer uses a non-linear radial basis function as the activation function, which converts the input parameters into high dimension space which is then fed into the network to linearly seperate the problems.

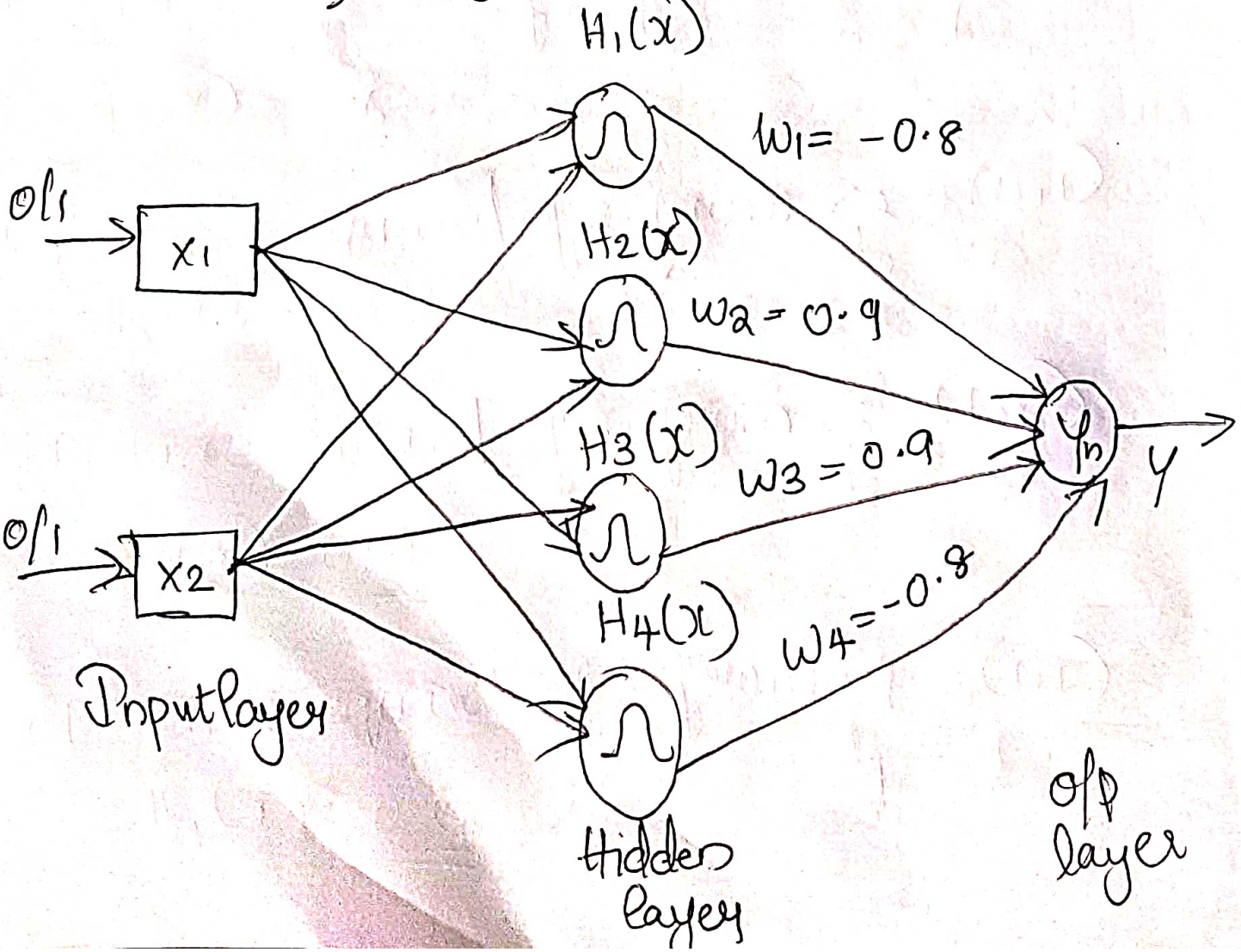


6) Consider the XOR Boolean function that has 4 patterns (0,0) (0,1) (1,0) and (1,1) in a 2-D input space.

$\sigma = 1$

Construct a RBFNN as shown in figure that classifies the input patterns.

- (0,0) → 0
- (0,1) → 1
- (1,0) → 1
- (1,1) → 0



Define 4 Hidden layer neurons with Gaussian RBF:-

$$H_1(x) = e^{\frac{-(x-c_1)^2}{2\sigma^2}} \; ; c_1 = (0,0)$$

$$H_2(x) = e^{\frac{-(x-c_2)^2}{2\sigma^2}} \; ; c_2 = (0,1)$$

$$H_3(x) = e^{\frac{-(x-c_3)^2}{2\sigma^2}} \; ; c_3 = (1,0)$$

$$H_4(x) = e^{\frac{-(x-c_4)^2}{2\sigma^2}} \; ; c_4 = (1,1)$$

$$d^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2$$

For input pattern (0,0)

* Distance squared of x from $c_1 = (0,0)$

$= (0-0)^2 + (0-0)^2 = 0$

$H_1(x) = e^{\frac{-(x-c_1)^2}{2\sigma^2}} = e^{\frac{0}{2}} = 1.0$

* Distance squared of x from $c_2 = (0,1)$

$= (0-0)^2 + (0-1)^2 = 1$

$H_2(x) = e^{\frac{-(x-c_2)^2}{2\sigma^2}} = e^{-\frac{1}{2}} = 0.6$

* Distance squared of x from $c_3 = (1,0)$

$= (0-1)^2 + (0-0)^2 = 1$

$H_3(x) = e^{-1/2} = 0.6$

Distance squared of x from $C_4 = (1,1)$

$= (0-1)^2 + (0-1)^2 = 2$

$H_4(x) = e^{\frac{-(x-C_4)^2}{2\sigma^2}} = e^{\frac{-2}{2}} = 0.4$

$$\sum_{j=1}^{m} w_j \cdot (H_j x) = -0.8 \times 1.0 + 0.9 \times 0.6 + 0.9 \times 0.6 + -0.8 \times 0.4 = -0.04$$

For input pattern (0,1)

* Distance squared of x from $C_1 = (0,0)$

$d_1 = 1$.

$H_1(x) = 0.6$.

* Distance squared of x from $C_2 = (0,1)$

$d_2 = 0$.

$H_2(x) = 1.0$.

* Distance squared of x from $C_3 = (1,0)$

$d_3 = 2$

$H_3(x) = 0.4$

* Distance squared of x from $C_4 = (1,1)$

$d_4 = 1$.

$H_4(x) = 0.6$

$\sum_{j=1}^{m} w_j \cdot H_j(x) = -0.8 \times 0.6 + 0.9 \times 1.0 + 0.9 \times 0.4 + -0.8 \times 0.6 = 0.3$

For input patteen (1,0)

$d_1 = 1$

$H_1(x) = 0.6$

$d_2 = 2$

$H_2(x) = 0.4$

$d_3 = 0$

$H_3(x) = 1.0$

$d_4 = 1$

$H_4(x) = 0.6$

$\sum_{j=1}^{m} w_j H_j(x) = -0.8 \times 0.6 + 0.9 \times 0.4 + 0.9 \times 1.0 + -0.8 \times 0.6 = 0.3$

For input patteen (1,1)

~~Distance so~~ $d_1 = 2$

$H_1(x) = 0.4$

$d_2 = 1$

$H_2(x) = 0.6$

$d_3 = 1$

$H_3(x) = 0.6$

$d_4 = 0$

$H_4(x) = 1.0$

$\sum_{j=1}^{m} w_j H_j(x) = -0.8 \times 0.4 + 0.9 \times 0.6 + 0.9 \times 0.6 + -0.8 \times 1.0 = -0.04$

Forward phase calculation

Input		$H_1(x)$	$H_2(x)$	$H_3(x)$	$H_4(x)$	$\sum_{j=1}^{n} W_j H_j(x)$	op
0	0	1.0	0.6	0.6	0.4	−0.04	0
0	1	0.6	1.0	0.4	0.6	0.3	1
1	0	0.6	0.4	1.0	0.6	0.3	1
1	1	0.4	0.6	0.6	1.0	−0.04	0
		$w_1 = -0.8$	$w_2 = 0.9$	$w_3 = 0.9$	$w_4 = -0.8$		

CASE-BASED REASONING

- In CBR the instances are not represented as real-valued points, but instead, they use a ***rich symbolic*** representation and the methods used to retrieve similar instances are correspondingly more elaborate.
- CBR has been applied to problems such as
  - conceptual design of mechanical devices based on a stored library of previous designs,
  - reasoning about new legal cases based on previous rulings, and

CASE-BASED REASONING

Case-based reasoning consists of a cycle of the following four steps:

1. **Retrieve -** Given a new case, retrieve similar cases from the case base.
2. **Reuse -** Adapt the retrieved cases to fit to the new case.
3. **Revise -** Evaluate the solution and revise it based on how well it works.
4. **Retain -** Decide whether to retain this new case in the case base.

CASE-BASED REASONING – Example

Example:

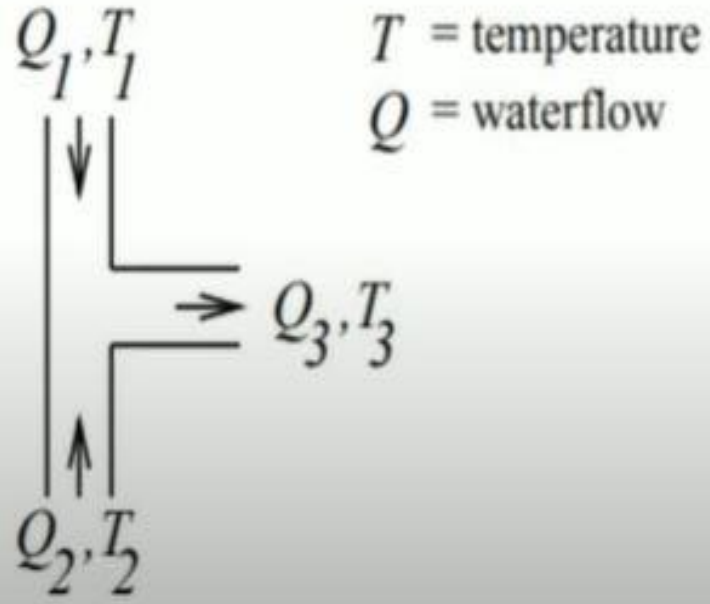
- Help desk that users call with problems to be solved.
- When users give a description of a problem, the closest cases in the case base are retrieved.
- The diagnostic assistant could recommend some of these to the user, adapting each case to the user's particular situation.
- If one of the adapted cases works, that case is added to the case base, to be used when another user asks a similar question.
- If none of the cases found works, some other method is attempted to solve the problem, perhaps by adapting other cases or having a human help diagnose the problem.
- When the problem is finally solved, the solution is added to the case base.

A prototypical example of a case-based reasoning

- The problem setting is illustrated in below figure

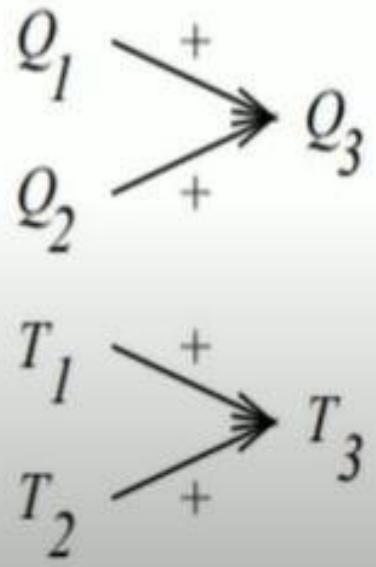
A stored case: T−junction pipe

Structure:



T = temperature
Q = waterflow

Function:



Case Based Reasoning

⇒ In CBR, everything is considered as case and based on the previous cases we propose a solution.

⇒ Instances represented as Symbols (not values)

CBR has 3 components

1. Similarity functions or distance measure
2. Approximation/Adjustment of instances
3. Symbolic representation of instances

→ For modelling CBR, we use CADET s/m.

⇓

[case based design tool]

⇒ has 15 predefined libraries.

* Suppose that the database of a CBR system contains the following four cases :-

Case	Monthly Income	Account balance	Home owner	Credit score
1	3	2	0	2
2	2	1	1	2
3	3	2	2	4
4	0	-1	0	0

→ The system is using the nearest neighbour retrieval algorithm with the following similarity function :

$$d(T,S) = \sum_{i=1}^{m} |T - S_i| * w_i$$

→ Where T is the Target case, S is the Source case, i is the number of a feature, and w_i are the weights. Cases with smaller values of $d(T,S)$ are considered to be more similar.

Consider the following new (tauget) case:-

Case	Monthly Income	Account Balance	Home owner	Credit score
5	3	1	2	?

Answer the following qns:-

a) Which case will the CBR s/m retrieve as the 'best match', if all the weights $w_i = 1$?

b) What can be changed in the similarity function to make feature 'Account Balance' three times more important than any other feature? Will this change influence the solution?

Solution

a) $d(T,S) = \sum_{i=1}^{m} |T - S_i| * w_i$

$d(T,S1) = |3-3| + |1-2| + |2-0| = 0+1+2 = 3$

$d(T,S2) = |3-2| + |1-1| + |2-1| = 1+0+1 = 2$

$d(T,S3) = |3-3| + |1-2| + |2-2| = 0+1+0 = 1$

$d(T,S4) = |3-0| + |1+1| + |2-0| = 3+2+2 = 7$

The most similar is case 3

b) What can be changed in the similarity function to make feature 'Account Balance' three times more important than any other feature? Will this change influence the solution?

- $d(T,S) = \sum_{i=1}^{m} |T - S_i| * w_i$

$d(T,S_1) = |3-3| * 1 + |1-2| * 3 + |2-0|$

$= 0+3+2 = 5$

$d(T, S_2) = |3-2| + |1-1| * 3 + |2-1| =$

$1 + 0 + 1 = 2$

$d(T, S_3) = |3-3| + |1-2| * 3 + |2-2|$

$= 0 + 3 + 0 = 3$

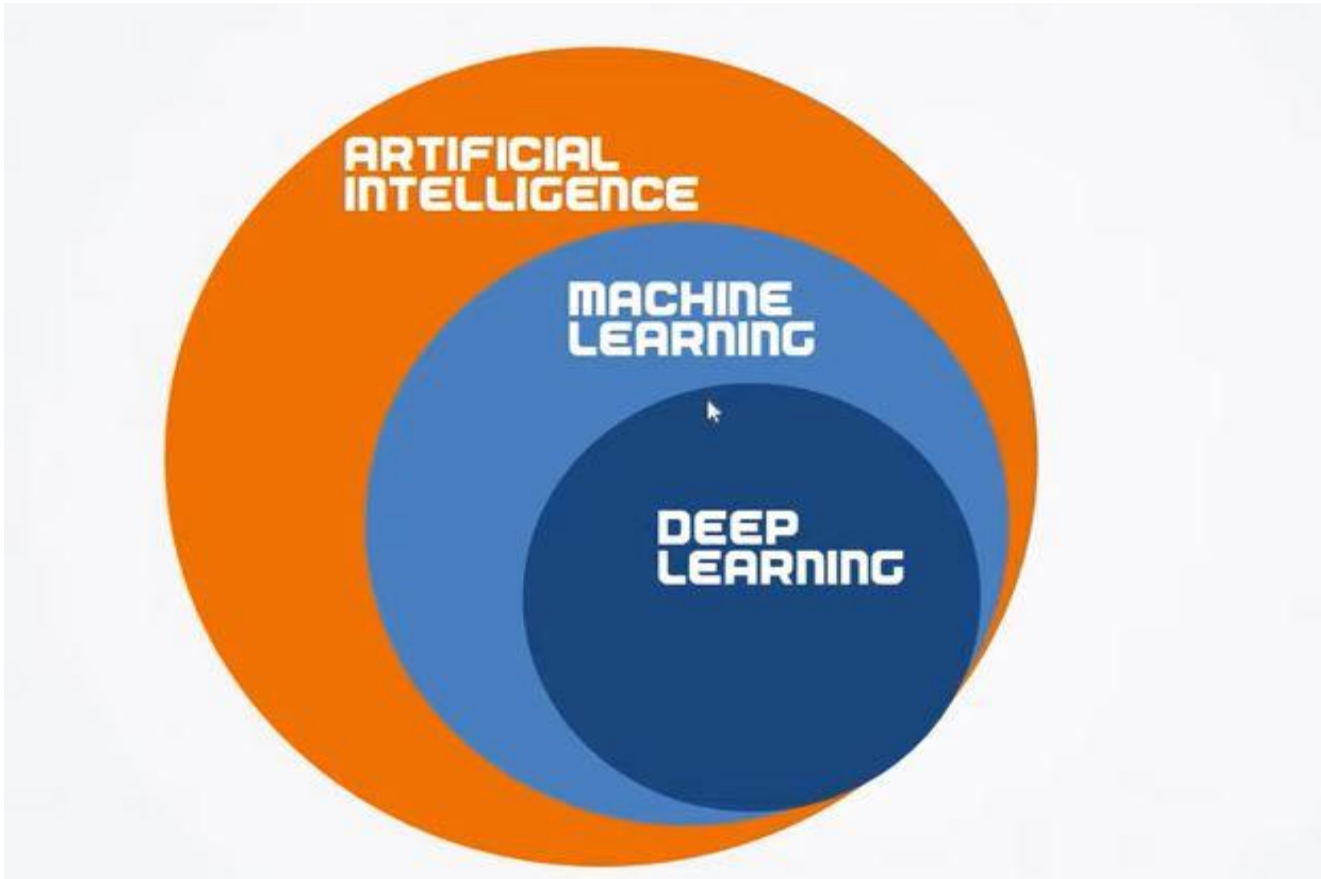
$d(T, S_4) = |3-0| + |1+1| * 3 + |2-0|$

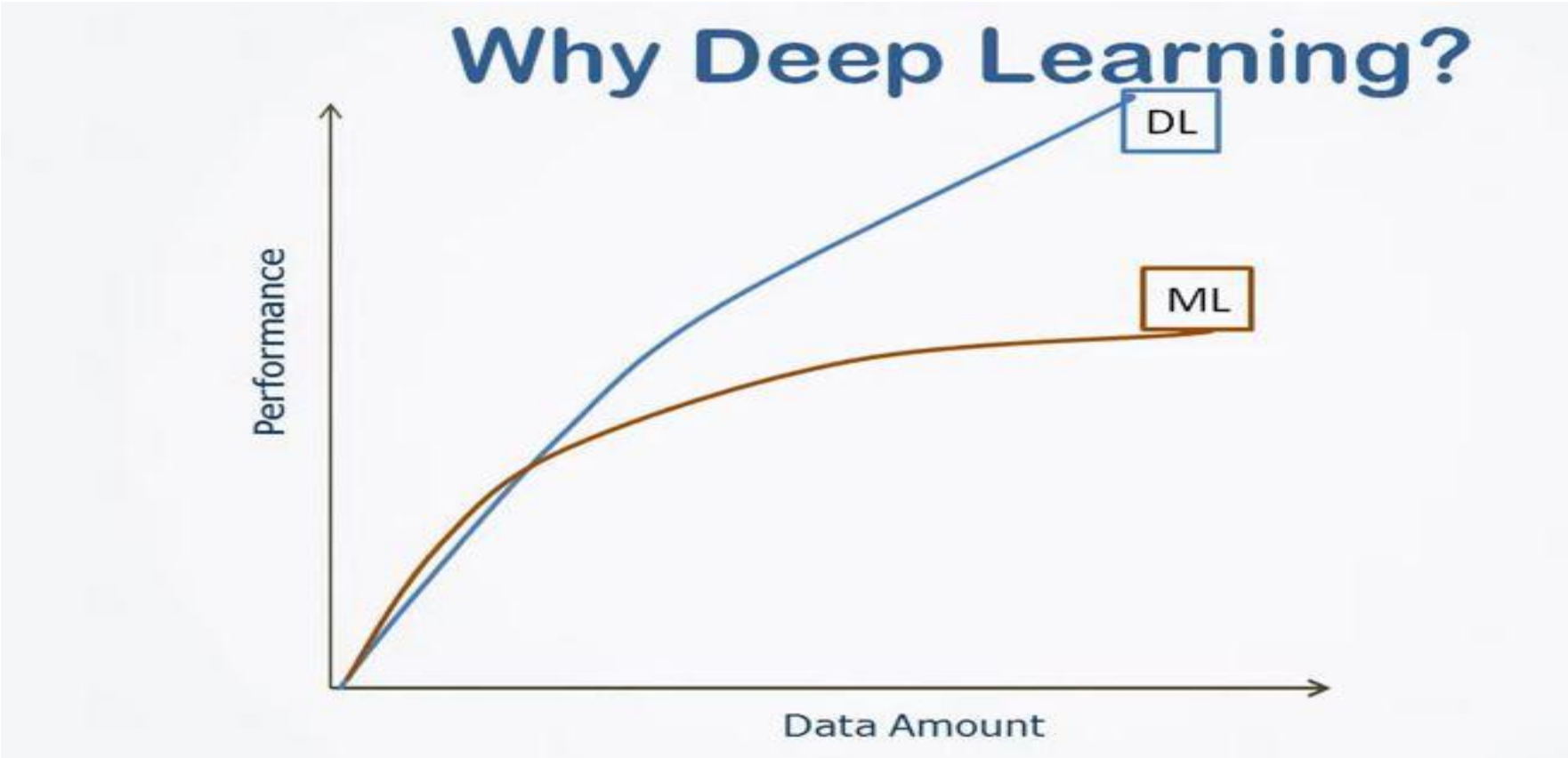
$= 3 + 6 + 2 = 11$

So Now the case 2 is similar.

DEEP LEARNING

Deep learning is a computer software that **mimics the network of neurons in a brain**. It is a subset of machine learning and is called deep learning because it makes use of deep **neural networks.**


ARTIFICIAL
INTELLIGENCE
MACHINE
LEARNING
DEEP
LEARNING


Why Deep Learning?
DL
ML
Performance
Data Amount


Deep learning use cases
Self Driving Car
Robotics & Labour Automation
Tumour Detection
Speech Recognition

Deep Learning

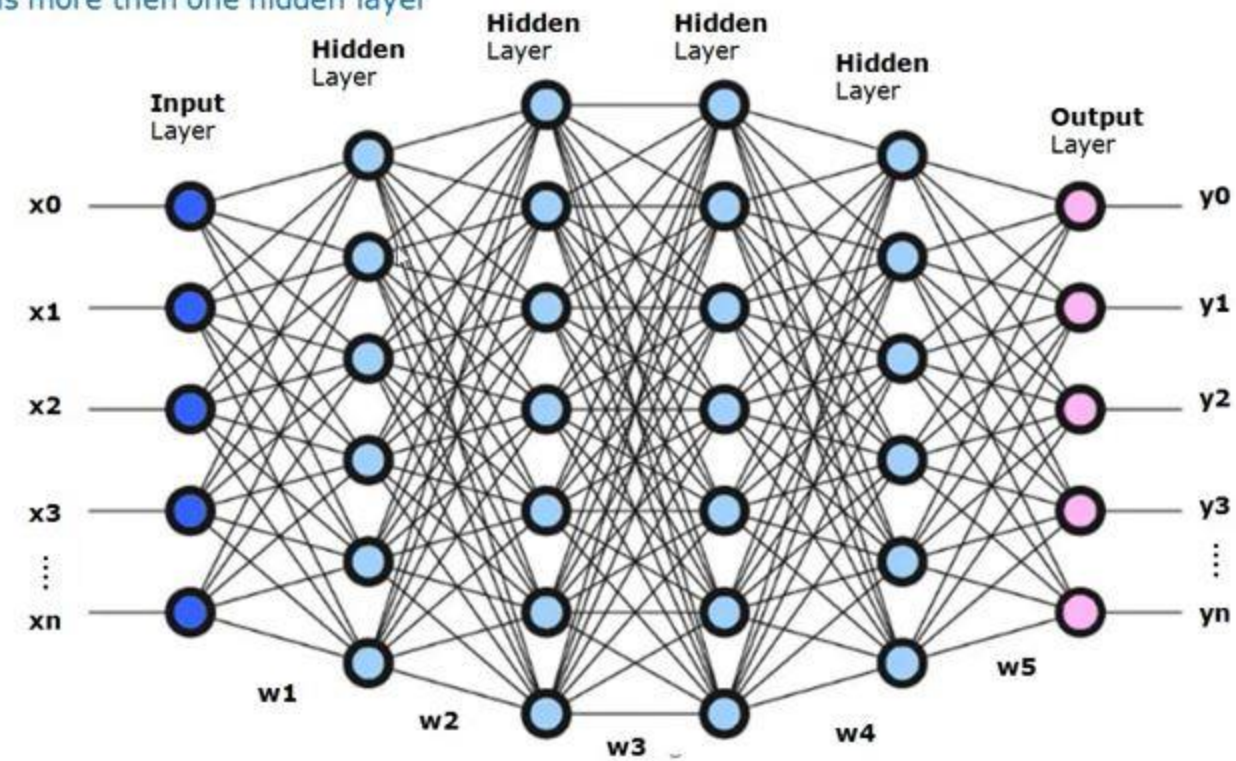
Deep learning algorithms are constructed with connected layers.

- ✓ The first layer - Input Layer
- ✓ The last layer - Output Layer
- ✓ All layers in between are called Hidden Layers. The word deep means the network join neurons in more than two layers.

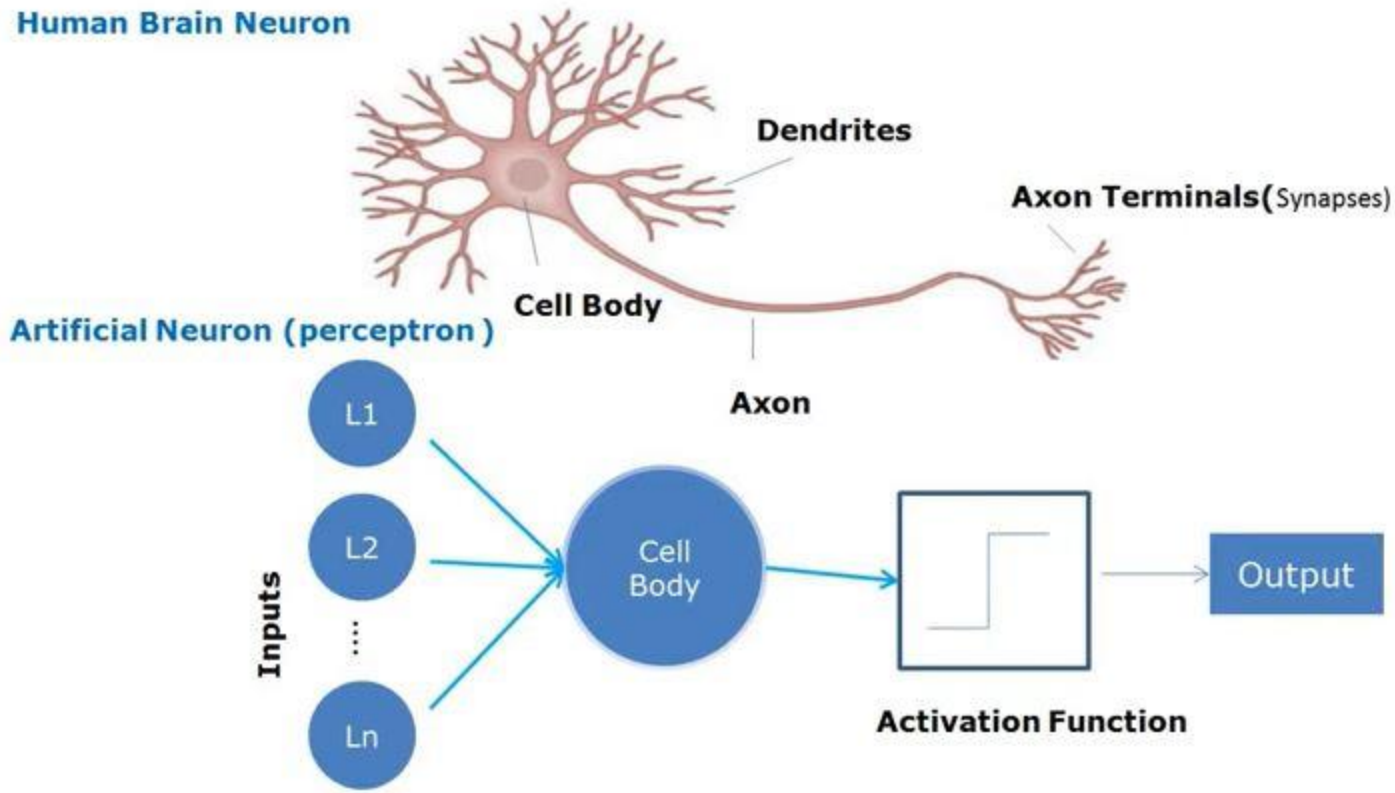
Deep Neural Network (DNN)

A deep neural network (DNN) is an artificial neural network (ANN or NN) with multiple layers between the input and output layers

DNN has more then one hidden layer



Human Brain and Artificial Neuron



Convolutional Neural Network (CNN)

CNN is a special type of Feed Forward Neural network

- ✓Images recognition
- ✓Images classifications
- ✓Objects detections
- ✓Faces Recognition

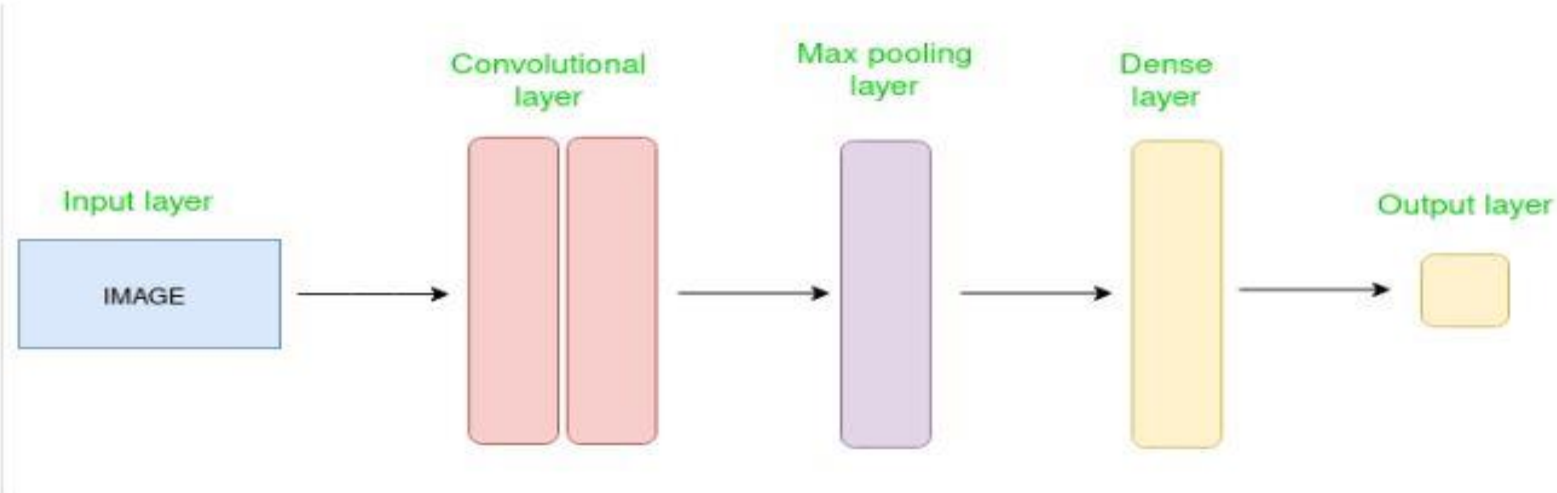
LAYERS

1.Input Layer: This layer represents the raw input data, which could be images, text, or any other form of structured data.
2.Convolutional Layer: This layer applies convolution operation to the input, using filters (also known as kernels) to extract features such as edges, textures, and patterns. The output of this layer is called feature maps.
3.Activation Layer: After each convolutional operation, an activation function like ReLU (Rectified Linear Unit) is usually applied element-wise to introduce non-linearity into the network, enabling it to learn complex patterns.
4.Pooling Layer: Pooling layers (commonly max pooling or average pooling) down-sample the feature maps by summarizing the presence of features in sub-regions, reducing the dimensionality of the data and making the network more computationally efficient.
5.Fully Connected (Dense) Layer: After several convolutional and pooling layers, the final feature maps are flattened into a vector and fed into one or more fully connected layers. These layers perform classification or regression tasks by learning global patterns in the feature maps.

Output Layer: The last layer of the network, it produces the final output. For classification tasks, this layer typically uses a softmax activation function to output probabilities for each class. For regression tasks, it may use a linear activation function.

CNN architecture

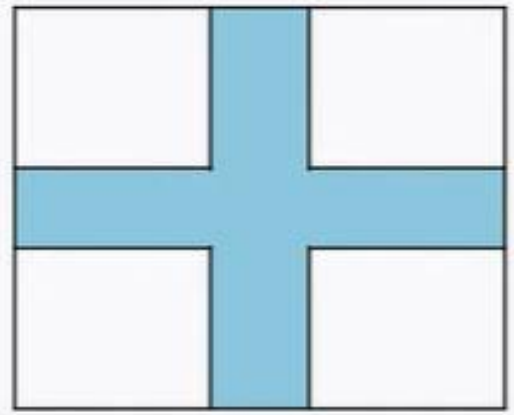
Convolutional Neural Network consists of multiple layers like the input layer, Convolutional layer, Pooling layer, and fully connected layers.



Simple CNN architecture

The Convolutional layer applies filters to the input image to extract features, the Pooling layer downsamples the image to reduce computation, and the fully connected layer makes the final prediction. The network learns the optimal filters through backpropagation and gradient descent.

Convolutional Neural Network (CNN)



Convolutional Neural Network (CNN)

Filters/Kernals

-1	1	-1
-1	1	-1
1	1	1

1	1	1
-1	1	-1
-1	1	-1

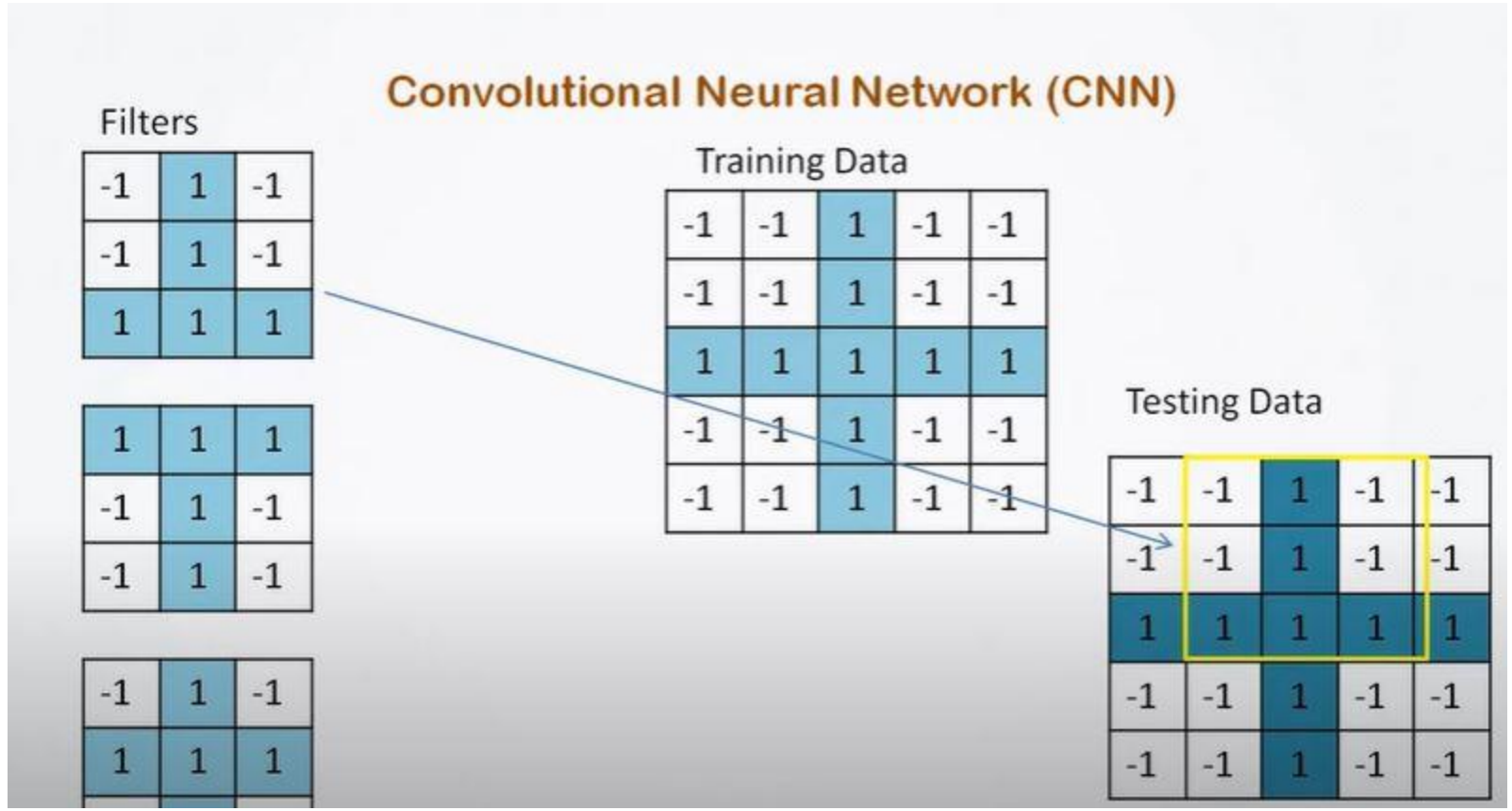
-1	1	-1
1	1	1
-1	1	-1

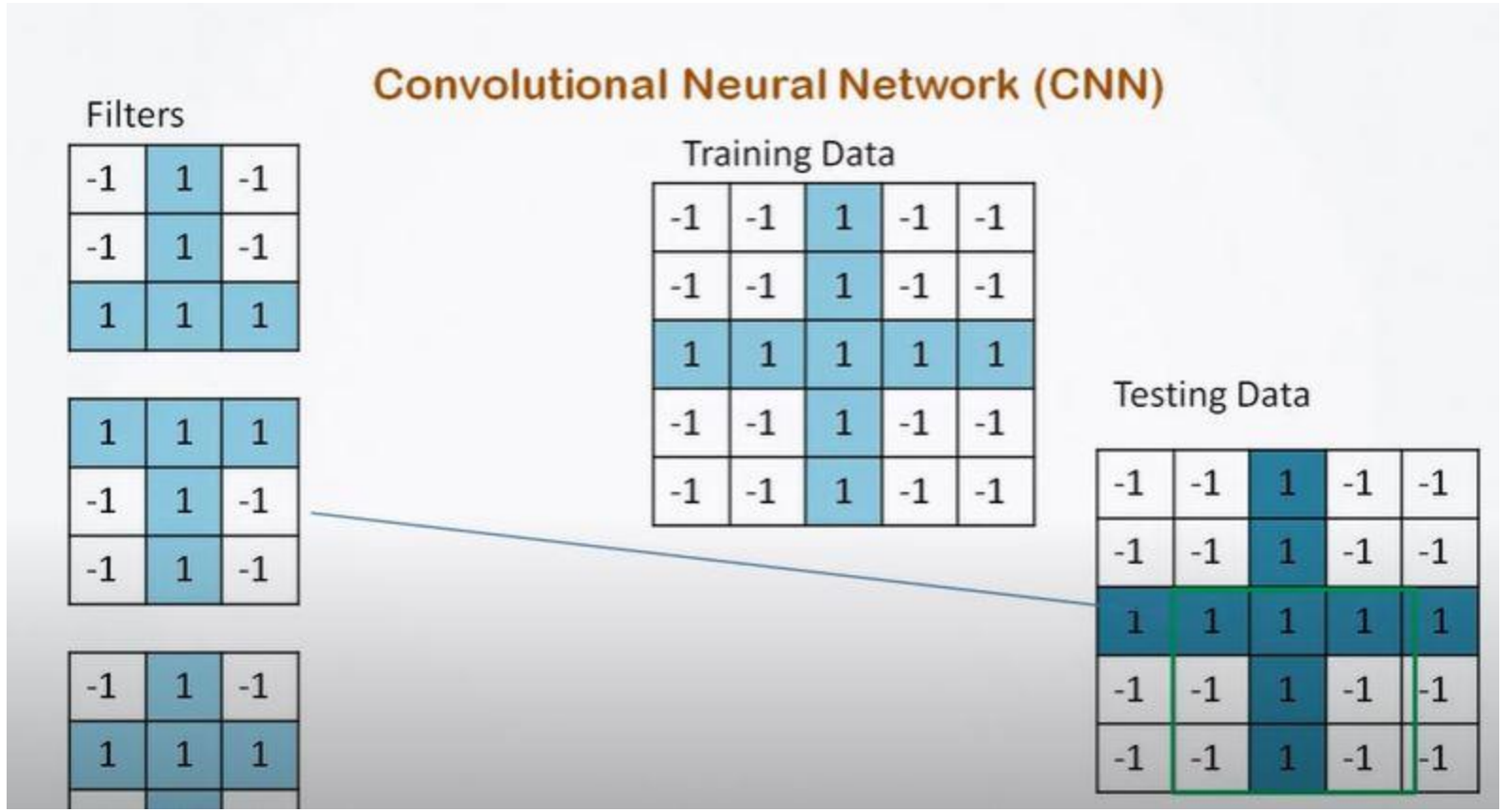
Training Data

-1	-1	1	-1	-1
-1	-1	1	-1	-1
1	1	1	1	1
-1	-1	1	-1	-1
-1	-1	1	-1	-1

Testing Data

-1	-1	1	-1	-1
-1	-1	1	-1	-1
1	1	1	1	1
-1	-1	1	-1	-1
-1	-1	1	-1	-1


Convolutional Neural Network (CNN)
Filters
-1 1 -1
-1 1 -1
1 1 1
1 1 1
-1 1 -1
-1 1 -1
-1 1 -1
1 1 1
Training Data
-1 -1 1 -1 -1
-1 -1 1 -1 -1
1 1 1 1 1
-1 -1 1 -1 -1
-1 -1 1 -1 -1
Testing Data
-1 -1 1 -1 -1
-1 -1 1 -1 -1
1 1 1 1 1
-1 -1 1 -1 -1
-1 -1 1 -1 -1


Convolutional Neural Network (CNN)
Filters
-1 1 -1
-1 1 -1
1 1 1
1 1 1
-1 1 -1
-1 1 -1
-1 1 -1
1 1 1
Training Data
-1 -1 1 -1 -1
-1 -1 1 -1 -1
1 1 1 1 1
-1 -1 1 -1 -1
-1 -1 1 -1 -1
Testing Data
-1 -1 1 -1 -1
-1 -1 1 -1 -1
1 1 1 1 1
-1 -1 1 -1 -1
-1 -1 1 -1 -1

Convolutional Neural Network (CNN)

Filters

-1	1	-1
-1	1	-1
1	1	1

1	1	1
-1	1	-1
-1	1	-1

-1	1	-1
1	1	1
-1	1	-1

(1 - 1 - 1 + 1 - 1 - 1 +1 +1 +1)/9 = .11

-1	-1	1	-1	-1
-1	-1	1	-1	-1
1	1	1	1	1
-1	-1	1	-1	-1
-1	-1	1	-1	-1

1	-1	-1
1	-1	-1
1	1	1

Feature Map

.11		

Featured Map is featured/filtered image

Convolutional Neural Network (CNN)

Filters

-1	1	-1
-1	1	-1
1	1	1

1	1	1
-1	1	-1
-1	1	-1

-1	1	-1
1	1	1
-1	1	-1

(1 + 1 + 1 + 1 + 1 + 1 +1 +1 +1)/9 = 1

-1	-1	1	-1	-1
-1	-1	1	-1	-1
1	1	1	1	1
-1	-1	1	-1	-1
-1	-1	1	-1	-1

1	1	1
1	1	1
1	1	1

Stride is the number of pixels shifts over the input matrix.
Here Stride = 1

Feature Map

.11	1	

Convolutional Neural Network (CNN)

Filters

-1	1	-1
-1	1	-1
1	1	1

1	1	1
-1	1	-1
-1	1	-1

-1	1	-1
1	1	1
-1	1	-1

-1	-1	1	-1	-1
-1	-1	1	-1	-1
1	1	1	1	1
-1	-1	1	-1	-1
-1	-1	1	-1	-1

to make the final filter more informative
we use padding in image matrix

Featured Images

.11	1	.33
-.33	.11	-.33
-.33	.11	-.11

-33	.11	-.33
-33	.11	-33
.11	1	-11

-.11	.11	-33
-33	1	.33
-33	.11	-11

Filters

-1	1	-1
-1	1	-1
1	1	1

1	1	1
-1	1	-1
-1	1	-1

-1	1	-1
1	1	1
-1	1	-1

ReLu Layer

-1	-1	1	-1	-1
-1	-1	1	-1	-1
1	1	1	1	1
-1	-1	1	-1	-1
-1	-1	1	-1	-1

$$f(x) = \begin{cases} 0 \text{ if } x < 0 \\ X \text{ if } x >= 0 \end{cases}$$

Featured Images

.11	1	.33
0	.11	0
0	.11	0

0	.11	0
0	.11	0
.11	1	0

0	.11	0
0	1	.33
0	.11	0

Convolutional Neural Network (CNN)

0	0	0	0	0	0	0
0	-1	-1	1	-1	-1	0
0	-1	-1	1	-1	-1	0
0	1	1	1	1	1	0
0	-1	-1	1	-1	-1	0
0	-1	-1	1	-1	-1	0
0	0	0	0	0	0	0

Padding : it refers to the amount of pixels added to an image when it is being processed by the kernel of a CNN

Padding is simply a process of adding layers of zeros to our input image

Pooling Layer

-1	-1	1	-1	-1
-1	-1	1	-1	-1
1	1	1	1	1
-1	-1	1	-1	-1
-1	-1	1	-1	-1

Featured Images

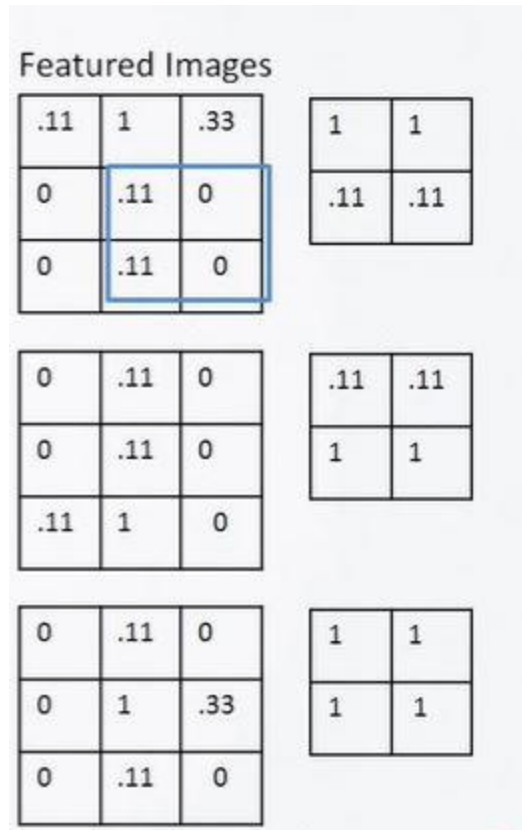
.11	1	.33
0	.11	0
0	.11	0

0	.11	0
0	.11	0
.11	1	0

0	.11	0
0	1	.33
0	.11	0

1	1
.11	

Shrink image in smaller size.
Move window size 2 across filtered image.
Pick highest value


Featured Images
.11 1 .33
0 .11 0
0 .11 0
1 1
.11 .11
0 .11 0
0 .11 0
.11 1 0
.11 .11
1 1
0 .11 0
0 1 .33
0 .11 0
1 1
1 1

Process Can be Repeated

-1	-1	1	-1	-1
-1	-1	1	-1	-1
1	1	1	1	1
-1	-1	1	-1	-1
-1	-1	1	-1	-1

Featured Images

.11	1	.33
0	.11	0
0	.11	0

1	1
.11	.11

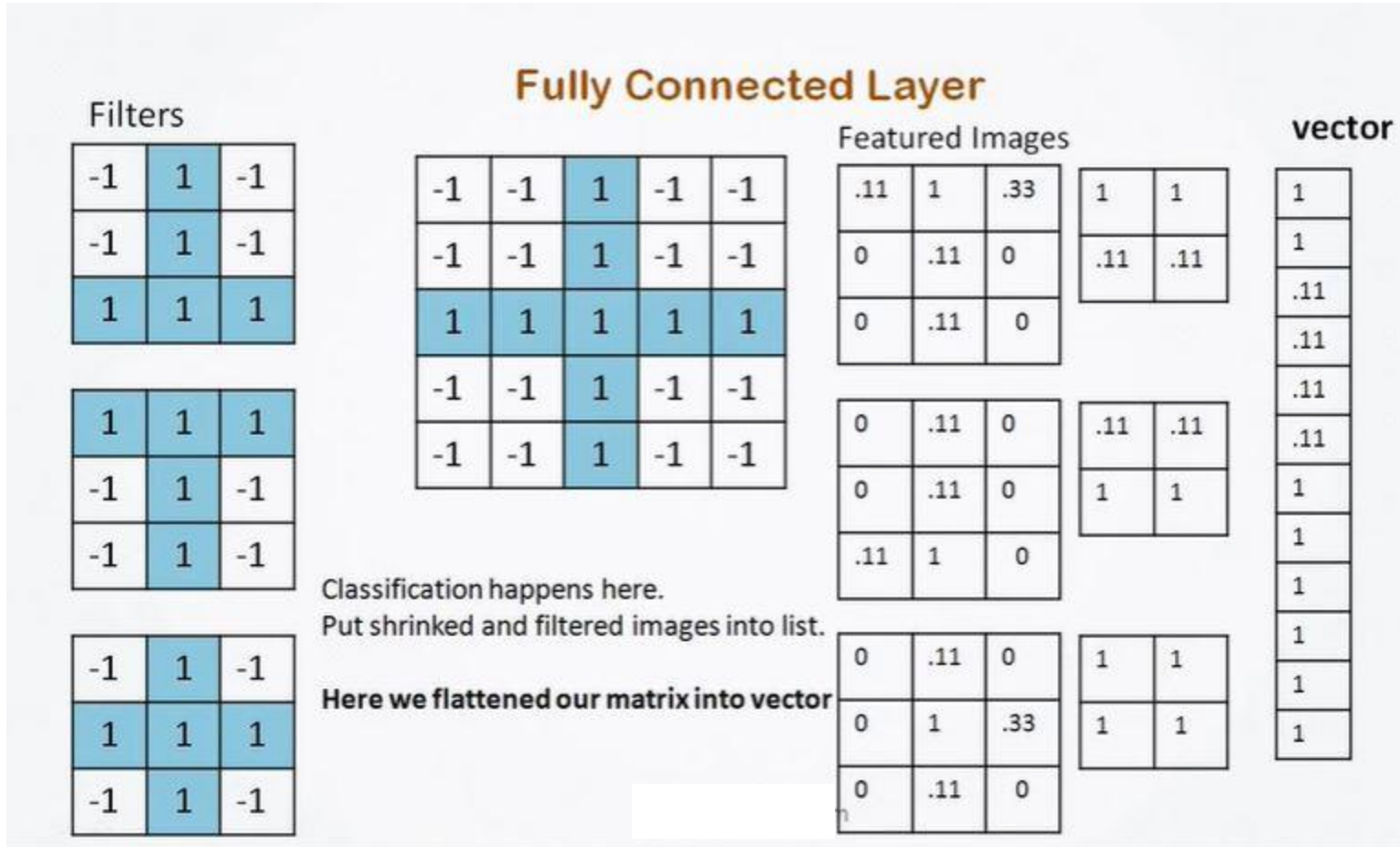
0	.11	0
0	.11	0
.11	1	0

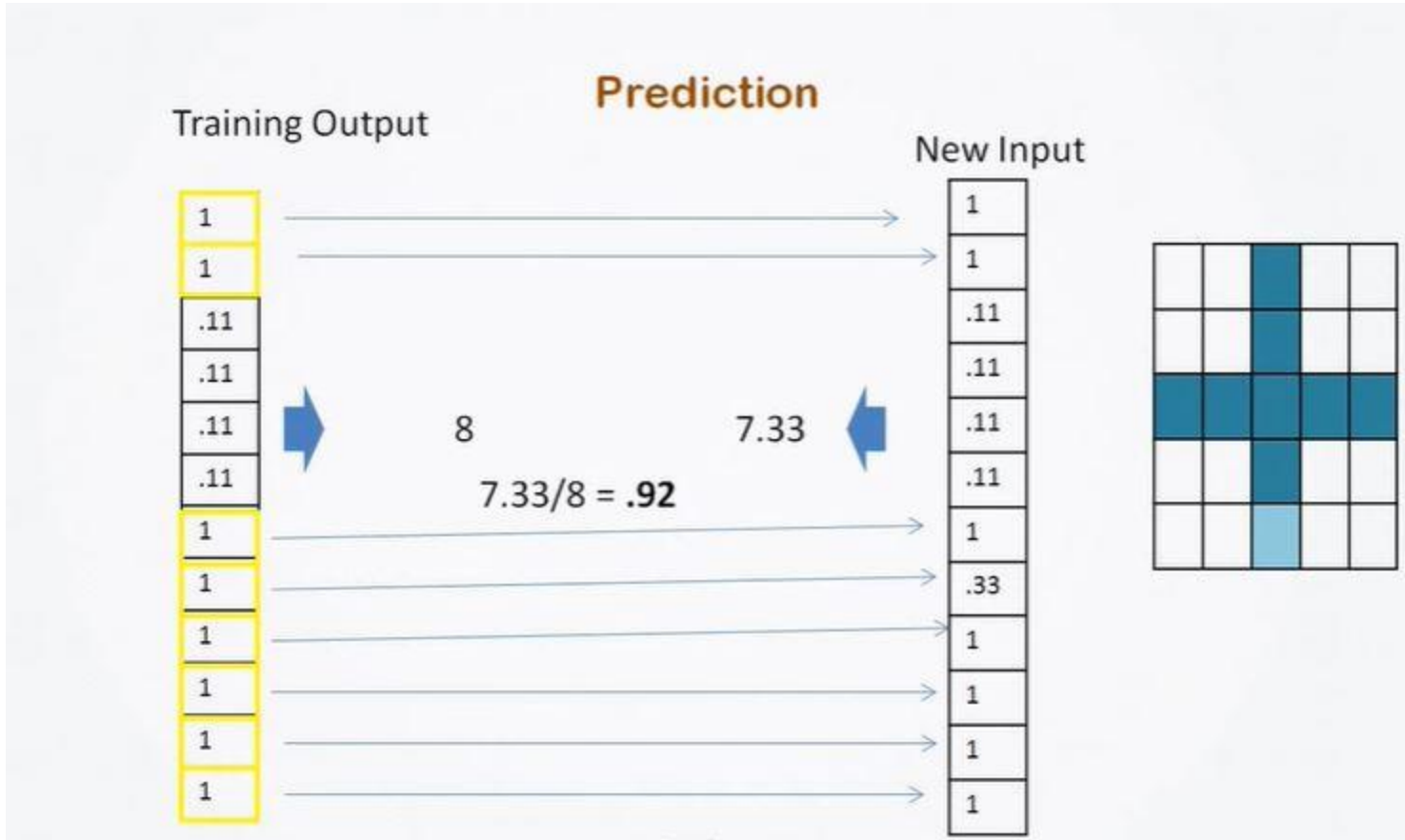
.11	.11
1	1

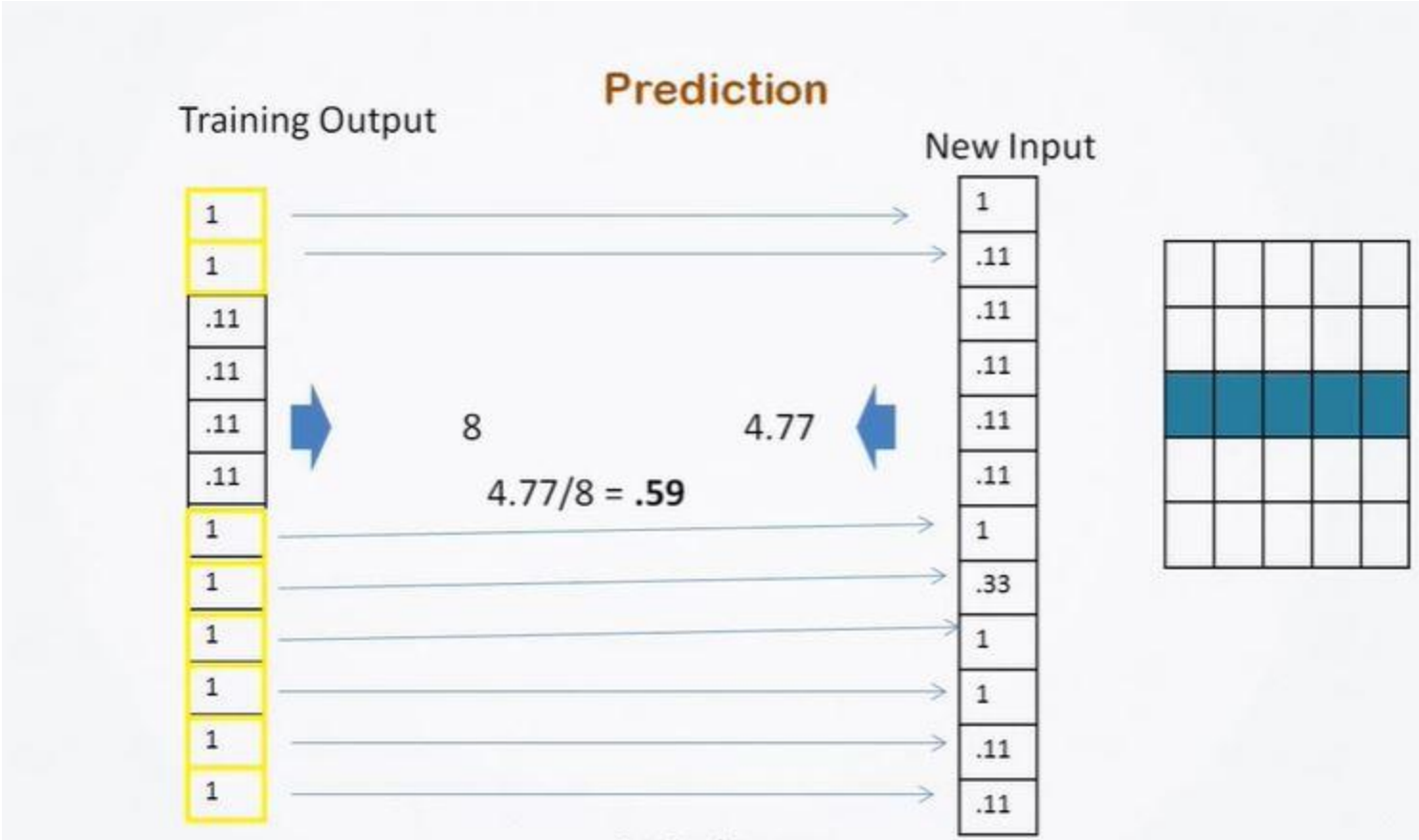
0	.11	0
0	1	.33
0	.11	0

1	1
1	1

We Can Add Convolution ReLu and Pulling Layer Again to shirnk to output images But for this example images already Shrinked into smaller sizes


Fully Connected Layer
Filters
-1 1 -1
-1 1 -1
1 1 1
1 1 1
-1 1 -1
-1 1 -1
-1 1 -1
1 1 1
-1 1 -1
-1 -1 1 -1 -1
-1 -1 1 -1 -1
1 1 1 1 1
-1 -1 1 -1 -1
-1 -1 1 -1 -1
Classification happens here.
Put shrinked and filtered images into list.
Here we flattened our matrix into vector
Featured Images
.11 1 .33
0 .11 0
0 .11 0
1 1
.11 .11
0 .11 0
0 .11 0
.11 1 0
.11 .11
1 1
0 .11 0
0 1 .33
0 .11 0
1 1
1 1
vector
1
1
.11
.11
.11
.11
1
1
1
1
1
1


Prediction
Training Output
New Input
1
1
.11
.11
.11
.11
1
1
1
1
1
1
1
1
.11
.11
.11
.11
1
.33
1
1
1
1
8
7.33
7.33/8 = .92


Prediction
Training Output
New Input
1
1
.11
.11
.11
.11
8
4.77
4.77/8 = .59
1
1
1
1
1
1
1
.11
.11
.11
.11
.11
.11
1
.33
1
1
.11
.11

Automated Feature Engineering

- Automated feature engineering refers to the process of automatically generating new features from existing data to improve the performance of machine learning models.
- Traditionally, feature engineering has been a manual and time-consuming task, requiring domain knowledge and experimentation to identify relevant features that capture important patterns in the data.
- Automated feature engineering techniques aim to streamline this process by automatically generating and selecting features based on predefined criteria or through machine learning algorithms.
- Automated feature engineering can be implemented using libraries and frameworks specifically designed for this purpose, such as Feature tools, TPOT, or auto-sklearn.

Featuretools is an open source library for performing automated feature engineering. It is a great tool designed to fast-forward the feature generation process, thereby giving more time to focus on other aspects of machine learning model building. In other words, it makes your data "machine learning ready".

- Feature Tools is a popular open-source Python framework for automated feature engineering. It works across multiple related tables and applies various transformations for feature generation.
- The entire process is carried out using a technique called “Deep Feature Synthesis”(DFS) which recursively applies transformations across entity sets to generate complex features.

Three major components of a Feature Tool Package

- Entities
- Deep Feature Synthesis (DFS)
- Feature primitives

- **Entity Set**
  An entity is a data table holding information. It is the most fundamental building block of the framework. A collection of such entities is called an Entity Set. The entity sets also include additional information like schemas, metadata, and various entities' relationships.

- **Feature Primitives**
  Primitives are the statistical functions applied to transform the data present in the entity set. The functions include aggregations, ratios, percentages, etc. Primitives may process multiple data entities to create a single value, such as sum, min, or max, or apply a transformation on entire columns to create a new feature.

- **Deep Feature Synthesis (DFS)**
  DFS is the algorithm used by the framework for the automated extraction of features. It uses a combination of primitives and applies them to the entity sets to generate the features. The primitives are applied so that the new features result from complex operations applied across various dataset parts.