Unit-7: Representations, Description and Recognition

Representations and Description

Introduction to Some Descriptors

Image analysis and image understanding are two important systems that constitute a computer vision system. The first step in image analysis system is segmentation. This step is followed by representation and description schemes. The output of the segmentation process is the boundary pixels of objects.

The representation of the object boundaries is important for the analysis of shape. Shape synthesis is useful in computer-aided design of parts and assemblies. In general, the representation schemes can be carried out in two ways:

- First, the representation can be done using external characteristics of the objects or regions in the image. When the focus is on shape characteristics then it is necessary to consider external representation.
- Second, we can use internal characteristics. When the primary focus is on texture or patterns or colors then it is necessary to consider the internal representation.

Once the representation is over, the next process is to describe the region based on representation.

A region can be described by extracting the features such as orientation of the edge lines joining the extreme points and the number of concavities in the boundary and length of boundary.

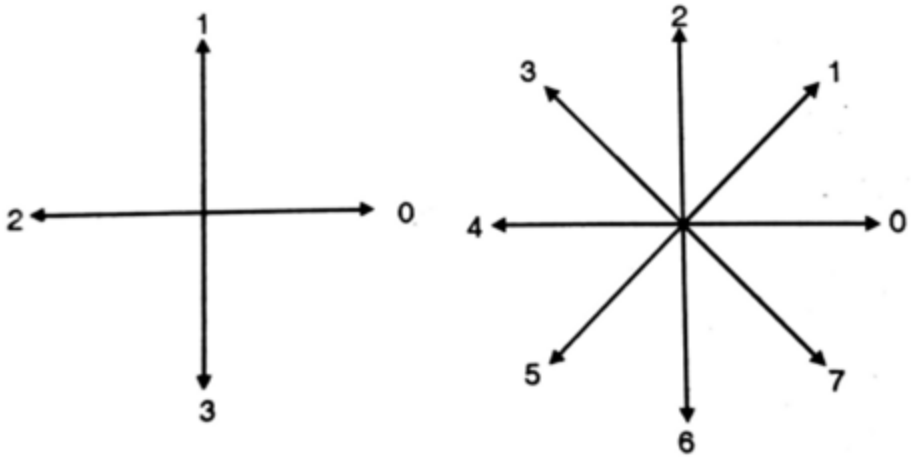
Some of the basic descriptors are given bellow:

simple feature extraction is done using chain code and fourrier descriptors

Chain Code:

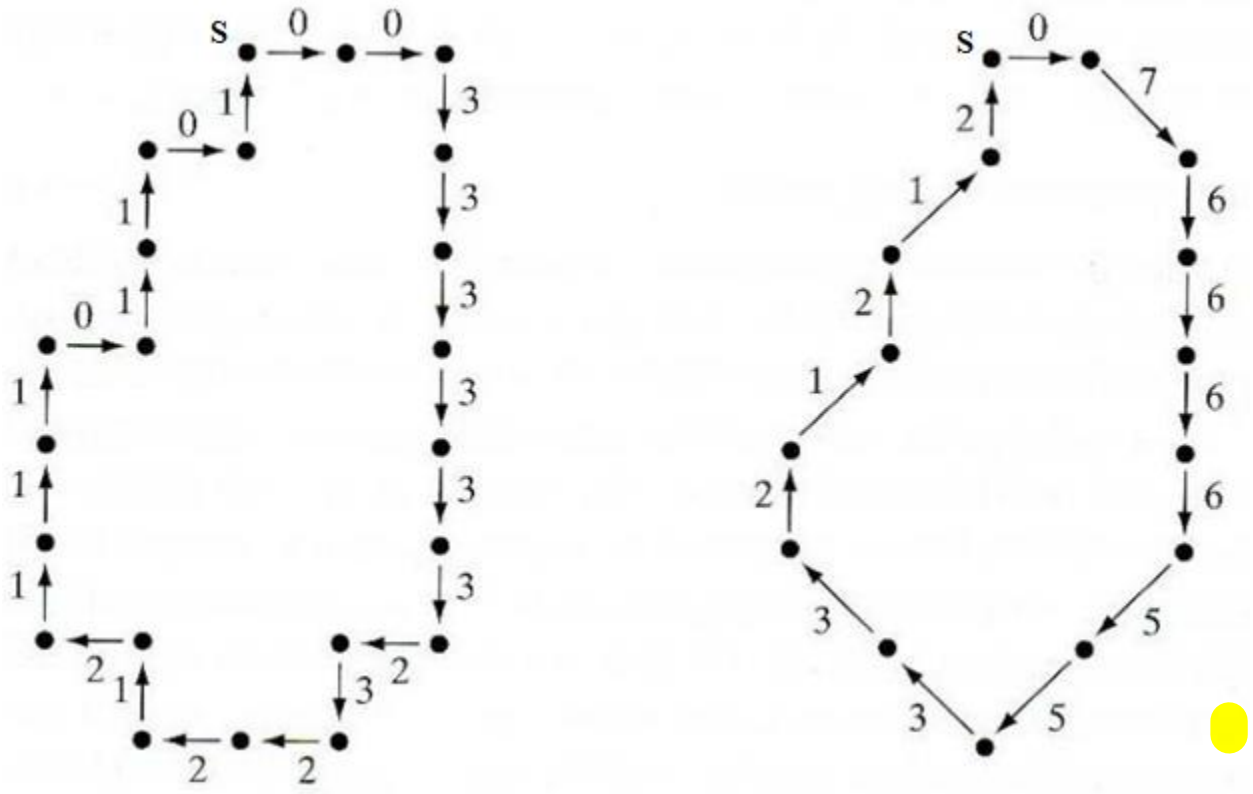
The chain code technique is simple and effectively used to represent the boundary of an object. There are two different chain code approaches.

In the first approach 4-directions are used whereas in the second approach 8-directions are used. The 4 and 8 directions are shown in figure bellow



4-directional chain code 8-directional chain code

The direction of each segment in the image can be coded using the numbers allotted to the directions. Let us see the steps involved in coding an arbitrary boundary shown in figure bellow



1. Boundary using 4-directional chain code 2. Boundary using 8-directional chain code

To represent the digital boundary, consider a starting point **'S'** then from the starting point, move to the next point in the boundary using the 4-directional or 8-directional code.

The boundary given in figure 1 can be represented using the 4-directional chain code **00333333232212111011 01**.

Similarly, the boundary shown in figure 2 can be represented using the 8-directional chain code **076666553321212**.

The accuracy of the resulting code depends on the grid spacing. Closer the grid, larger the accuracy and length of the chain code. The chain code for a boundary is not unique and depends on the starting point selection.

Limitation of chain code

Chain code will be varies or change when

1. Starting point of image is change
2. Image is reflected
3. Image is rotated

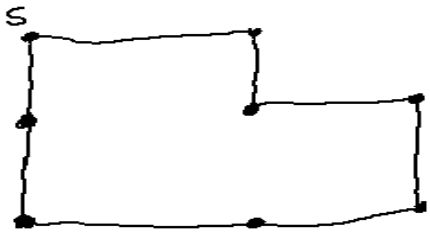
To overcome this limitation chain code normalization is used. The normalized chain code also called Shape Number will be same even stating point is changed, reflection, or rotation of object is done.

Procedure for normalization is

1. Find the normal chain code of given image
2. Find the first difference of corresponding two code
3. Start writing the code from first 0 position

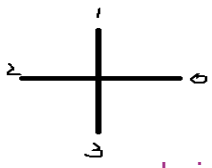
Let's take example of normalized chain code to overcome above mentioned limitation of chain code.

Given an object with starting point S



Chain code (CC): **03032211**

Now, find first difference of corresponding codes (*Count the difference between two codes in anticlockwise direction of 4-directional chain code symbol*)



chain code ko last ma euta 0 jsto aafae assume gaerw nikalinxa hae differnce

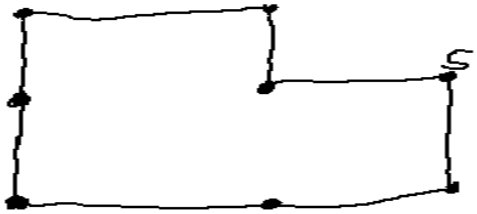
Chain code of first difference (CFD): **31330303**

Then find normalize chain code or shape number (*Start writing code from first zero)*

Shape Number: **03033133**

Starting point variation

Let's change the starting point

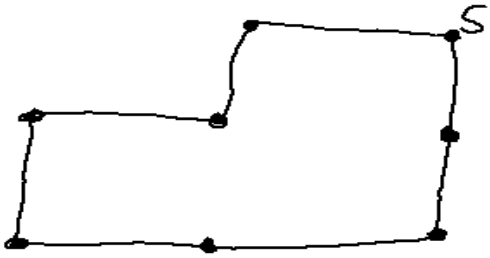


CC: **32211030**

CFD: **30303313**

Shape Number: **03033133**

Reflection of object

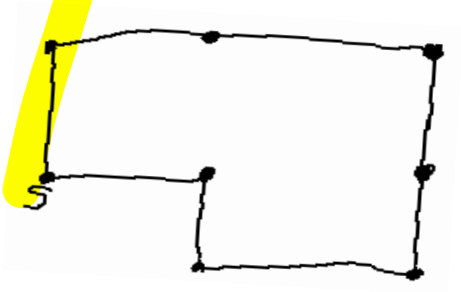


CC: **33221010**

CFD: **03033133**

Shape Number: **03033133**

Rotation of object



CC: **10033212**

CFD: **30303313**

Shape Number: **03033133**

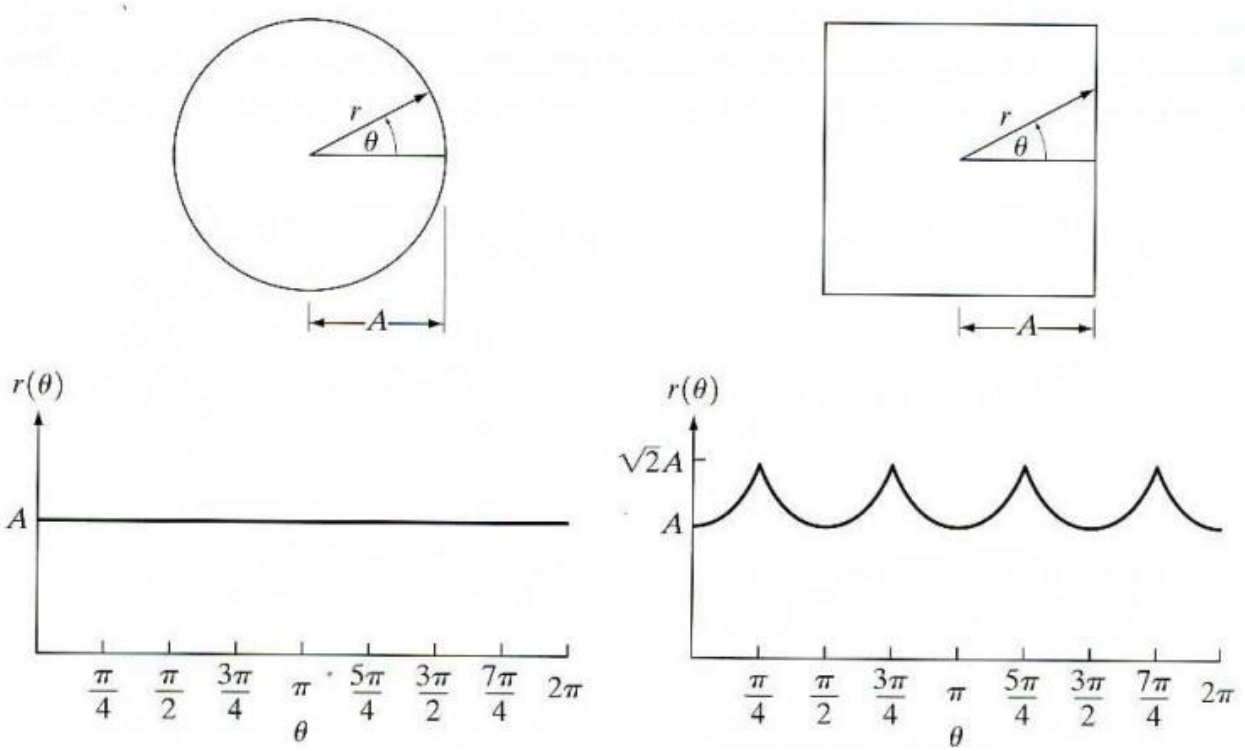
Shape Number

Based on 4-directional-chain code or 8- directional-chain code, the shape number is the difference with the smallest magnitude. The number of digits in the shape number is the called the order.

Shape Number is actually a normalized chain code.

Signature:

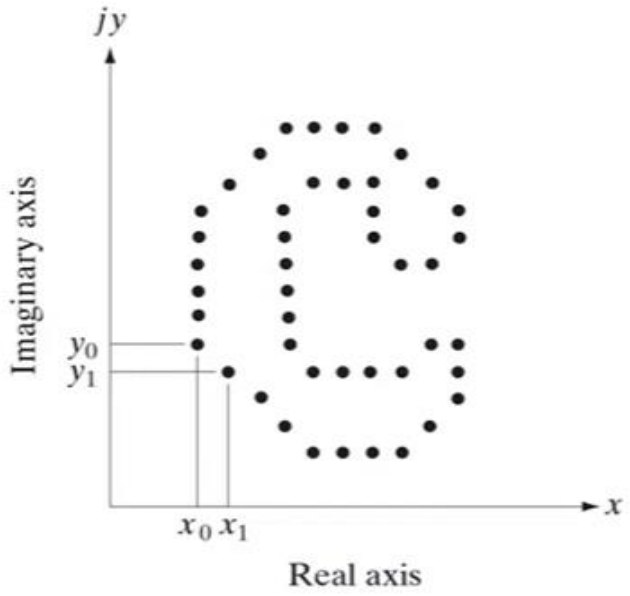
A signature is a 1-D functional representation of a boundary. Any boundary can be represented in a simple way using the signature. Any boundary can be described using the two-dimensional function. In order to reduce the dimensionality, the signature is generated and boundary can be described using the one-dimensional function. The signature generation for two typical boundaries is illustrated in Figures bellow.



Last two graphs are one dimensional representation or signature of first two shapes.

Fourier Descriptors

A common method of describing the contour (Outline) of an object is by using 1-Dimesnional Fourier Transform.



The figure shows a N-point digital boundary in the spatial Domain. Each of these edges pixels can be defined by its **x** and **y** coordinates. Starting at an arbitrary point (x0,y0),(x1,y1),(x2,y2)…….(x_{k-1},y_{k-1}) points are encountered as we move in the counter clockwise direction.

The coordinates can be expressed in the form **x(k) = x_k** and **y(k) = y_k** .With this notation, the boundary itself can be represented as the sequence of coordinates

S(k)= [x(k), y(k)] each point ko coordinate didae represent vairaxani tw

For, k=0,1, 2,.....,K-1

These coordinates values can be used to generate a complex function of the form,

S(k) = x(k) + jy(k) jailai yo second walae term ma lagne raexa j uta fourirer series mani cos(theta)+jsin(theta)

For, k=0,1, 2,.....,K-1

Hence the x-axis is treated as real axis and y-axis as the imaginary axis of the sequence of complex number.

The Fourier transform of this function S(k) yields the frequency components that describe the given edge. teha mathi ko ma points mtra xa ni edge tw chaiyoni

The discrete Fourier transform (DFT) of S(k) is

$$a(u) = \sum_{k=0}^{K-1} s(k) e^{-j2\pi uk/K}$$

@uk

uta fourier series ko T jsto eha K raexa descriptor ma chai

Where, u = 0,1,2……..K-1

The complex coefficients of a(u) from 0 to K-1 are called Fourier Descriptors of the boundaries.

The inverse Fourier Transform of a(u) restore the corffients of S(k)

$$s(k) = \frac{1}{K} \sum_{u=0}^{K-1} a(u) e^{j2\pi uk/K}$$

Where, k = 0,1,2……..K-1

However instead of using all the Fourier coefficients, we only use few of them say first P in the range 0 to P-1, while remaining terms are made zero,

a(u) = 0 if u > P-1

The result of the approximation of S(k) is

$$\hat{s}(k) = \frac{1}{P} \sum_{u=0}^{P-1} a(u) e^{j2\pi uk/P}$$

Where, k = 0,1,2……..K-1

Although only P terms are used to obtain each component of $\hat{s}(k)$, k still ranges from 0 to K-1. That is the same number of points exits in the approximate boundary, but not as many terms are used in the reconstruction of each point.

Although only P terms are used for a(u), a(k) still has 0 to N-1 values. That is the same number of points exist in the new approximated boundary but not as many terms are used in the reconstruction of each point.

suruma boundary represt garda chai almost sabai points herinxa tarw pxi inverse garda chai sabai heridenw teivyerw 0 to p-1 matra

Following example shows the variation of boundary points of chromosomes which varies the shape of chromosome.

dherae number of boundary points rakhyo vne ksto aauxa thorae rakhe ksto auxa testo

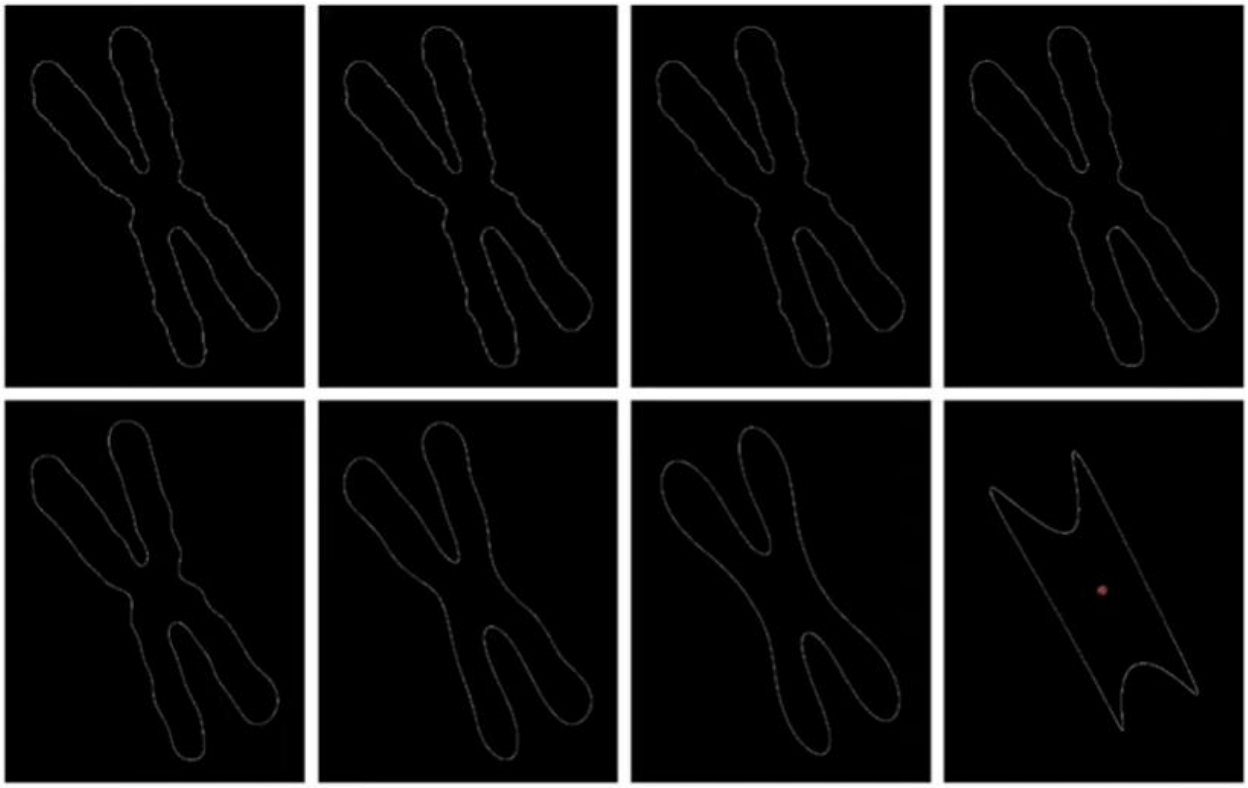


Fig: (a): Boundary of human chromosome (2868) points (b-h): Boundary reconstructed using 1434, 286, 144, 72, 36, 18, and 8 Fourier descriptors respectively. These numbers are approximately 50%, 10%, 5%, 2.5%, 1.25%, 0.63%, and 0.28% of 2868 points respectively.

Image Recognition

Patterns and Pattern Classes

A pattern is the arrangement of descriptors. The name *feature is* used often in the pattern recognition to denote a descriptor.

pattern class ko naam animal huna skxa teha aba dog ko cat ko patterns parne vyo etiniharu ko kei kei kura tw same hunxa teskae basis ma eutae class ma rakhxa

A *pattern class is a family* of patterns that share some common properties. Pattern classes are denoted ω1,ω2,……,ωW, where W is the number of classes. Pattern recognition by machine involves techniques for assigning patterns to their respective classes automatically and with as little human intervention as possible. Three common pattern arrangements used in practice are vectors (for quantitative descriptions), strings and trees (for structural descriptions).

Pattern vectors are denoted by bold lowercase letters, such as **x, y**, and **z**, and represented as columns (that is, n x 1 matrices). Hence a pattern vector can be expressed in the form

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix}$$

pattern=dog,pattern=cat esto hunxa

x1 x2 different descriptors ho x chai pattern ho
aba eha tyo x lai define grne descriptors hunxa ani w1 ma prxaki w2 ma parxa ki herne ho tyo match garerw

Where each component, x_i, represents the i^{th} descriptor and n is the total number of such descriptors associated with the pattern.

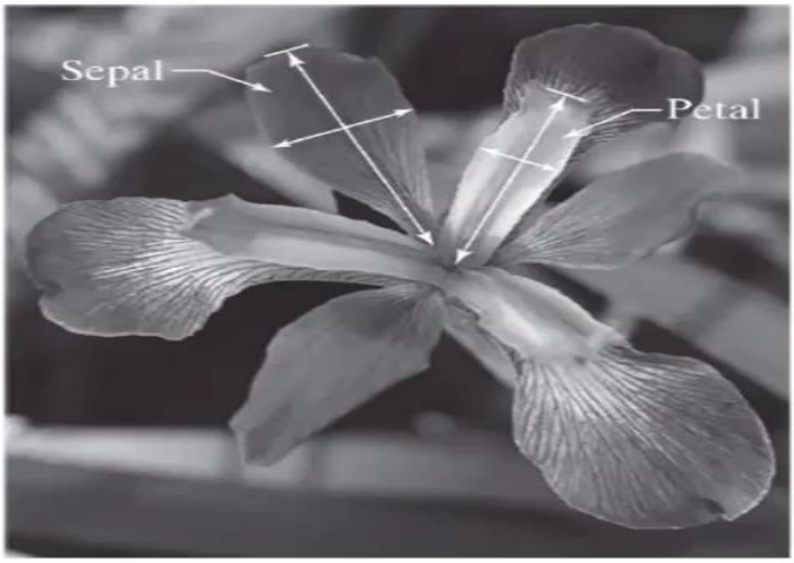
It can also express in the equivalent form

$$\boldsymbol{x = (x1, x2, , xn)^T}$$

Where T indicates transposition of this matrix

The nature of the components of a pattern vector **x** depends on the approach used to describe the physical pattern itself.

Let us illustrate with an example that is both simple and gives a sense of history in the area of classification of measurements.



$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

x_1 = *Petal Width*

x_2 = *Petal Length*

x_3 = *Sepal Width*

x_4 = *Sepal Length*

x=[1.5,2,3,2] esto hola re ani esko basis ma chai tyo kun class ma prxa tw vnerw decide garinxa

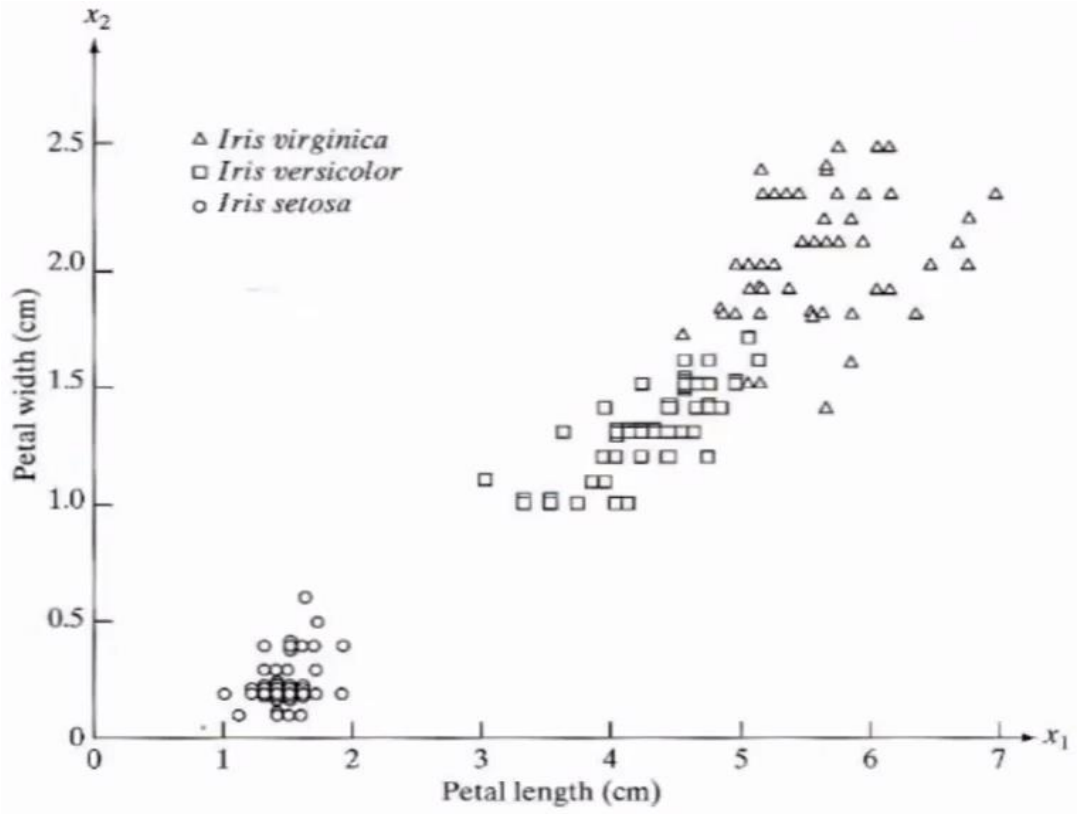
Here in this vector, **x** is the pattern class and x_1, x_2, x_3, x_4 are different patterns in that class.

In our present terminology, each flower is described by two measurements, which leads to a 2-D pattern vector of the form

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Where x_1 and x_2 correspond to petal length and width, respectively

Let's take some sample measurement for three types of flowers *setosa, virginica,* and *versicolor*. Then plot the two measurement petal length and petal width in graph and analyze the result.



In this graph three types iris flower describe by two measurements

The three pattern classes in this case, denoted ω1, ω2, and ω3, correspond to the varieties *setosa, virginica,* and *versicolor, r*espectively. Because the petals of flowers vary in width and length, the pattern vectors describing these flowers also will vary, not only between different classes, but also within a class.

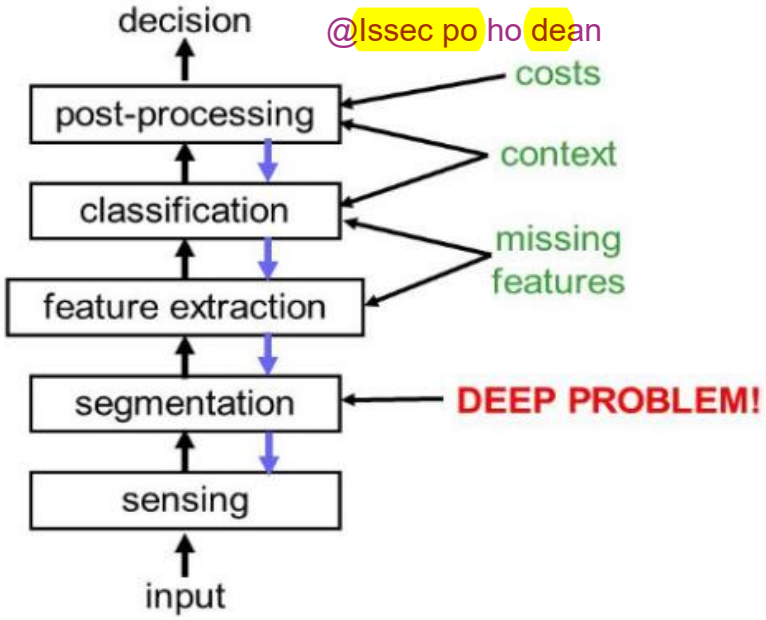
The above Figure shows length and width measurements for several samples of each type of iris. After a set of measurements has been selected (two in this case), the components of a pattern vector become the entire description of each physical sample.

Thus each flower in this case becomes a point in 2-D Euclidean space. We note also that measurements of petal width and length in this case adequately separated

the class of *Iris setosa* from the other two but did not separate as successfully the *virginica* and *versicolor* types from each other. This result illustrates the classic *feature selection* problem, in which the degree of class seperability depends strongly on the choice of descriptors selected for an application.

Overview of Pattern Recognition

Pattern Recognition can be defined as the classification of the data on the basis of the knowledge gained or on the basis of statistical information extracted from patterns and their representations.



Block Diagram of Pattern Recognition

- The sensor converts images or sounds or other physical input into signal data.
- Segmentation is the part which enables segregating sensed object from the background noise.
- The feature extractor measures object properties and select those features that are important for classification.
- The classifier uses these features to assign the sensed object to a category.

- Finally, the post-processor can take account of other considerations such as the cost of the error to decide appropriate action.

Decision-Theoretic Methods

Decision theoretic approaches to recognition are based on the use of decision functions

$$\boldsymbol{x = (x_1, x_1, x_1 \ldots\ldots x_n)}$$

It represents an n-dimensional pattern vector.

For W pattern classes (ω_1, ω_2, ω_3,….ω_w), the basic problem in decision-theoretic pattern recognition is to find W decision functions $d_1(x)$, $d_2(x)$, …… $d_w(x)$ with the property that if a pattern x belongs to class ω_i then

$$\boldsymbol{d_i(x) > d_j(x)\ , j=1,2,3\ldots\ldots W;\ j \neq i} \qquad \boldsymbol{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ 1}$$

In another word an unknown pattern x is said to belongs to the i^{th} pattern class ω_i, if upon substitution of x into all decision functions, $d_i(x)$ yield the largest numerical value.

The decision boundary separating class ω_i from ω_j is given by the value of x for which

$d_i(x) = d_j(x)$ or, $d_i(x) - d_j(x) = 0$

Common practice is to identify the decision boundary between two classes by the single function

$$\boldsymbol{d_{ij}(x) = d_i(x) - d_j(x) = 0} \qquad \boldsymbol{\ldots\ldots\ldots\ldots\ldots\ldots 2}$$

di(x) dj(x) chai tala minimum distance classifier bata calculate grne

Thus $d_{ij}(x) > 0$ for patterns of class ω_i and $d_{ij}(x) < 0$ for pattern of class ω_j

jun thulo hunxa tesmae prxa tei ho simply

Matching

Matching is the one of the popular decision theoretic method. Recognition techniques based on matching represent each class by a prototype pattern vector. An unknown pattern is assigned to the class to which, it is closest in terms of a predefined metric.

Minimum Distance Classifier

The simplest approach in matching method is the Minimum Distance Classifier, which, as its name implies, computes the (Euclidean) distance between the unknown and each of the prototype vectors. It chooses the smallest distance to make a decision.

Suppose that we define the prototype of each pattern class ω_j to be the mean vector of the patterns of that class:

$$m_j = \frac{1}{N_j}\sum_{x\epsilon\omega_j} x_j \qquad j = 1,2,\ldots\ldots\ldots\ldots,W \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ 3$$

Where N_j is the number of pattern vectors from class ω_j and the summation is taken over these vectors. W is the number of pattern classes.

One way to determine the class membership of an unknown pattern vector x is assign it to the class of its closest prototype.

If we use the Euclidean distance to determine similarity, the minimum distance classifier computes the distance as

$$d_j(x) = ||x - m_j||,\ j = 1,2,\ldots\ldots\ldots,w \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ 4$$

Where, x is an unknown pattern vector, and m_j is mean vector of a class.

$||x|| = (x^T x)^{1/2}$ *is the Euclidean norm*

Then assign x to class ω_j, if $d_j(x)$ is smallest distance. Hence, smallest distance implies to best match in this formulation.

Selecting the smallest distance is equivalent to evaluation of this equation

$$d_j(\mathbf{x}) = \mathbf{x}^T\mathbf{m}_j - \frac{1}{2}\mathbf{m}_j^T\mathbf{m}_j \qquad j = 1, 2, \ldots\ldots, W \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ 5$$

And assign x to class $\omega_{j,,}$ if $d_j(x)$ yields the larges numerical value.

i.e.

$d_j(x) > d_i(x)$; j = 1,2,……,w and $j \neq i$

This formulation agrees the concept of decision function as defined in Equation 1.

Note: *If there are two classes then boundary is perpendicular bisection line, if three classes then boundary is a plane like triangle, square and if more than there classes then boundary is hyperplane.*

Example 1: The two classes, Iris-versicolor and Iris-setosa, denoted ω_1 and ω_2 respectively, have sample mean vectors are

$m_1 = (4.3, 1.3)^T$, and $m_2 = (1.5, 0.3)^T$, find the equation of decision boundary.

Solution:

The decision function is

$d_1(x) = x^T m_1 - ½\ m_1{}^T m_1$

$= [x_1, x_2] \begin{bmatrix} 4.3 \\ 1.3 \end{bmatrix} - ½\ [4.3, 1.3] \begin{bmatrix} 4.3 \\ 1.3 \end{bmatrix}$

$= x_1\ 4.3 + x_2\ 1.3 - ½\ (4.3x4.3 + 1.3x1.3)$

$= x_1\ 4.3 + x_2\ 1.3 - ½\ (18.49 + 1.69)$

$= x_1\ 4.3 + x_2\ 1.3 - ½\ (20.18)$

$= 4.3x_1 + 1.3x_2 - 10.1$

$d_2(x) = x^T m_2 - ½\ m_2{}^T m_2$

$= 1.5x_1 + 0.3x_2 - 1.17$

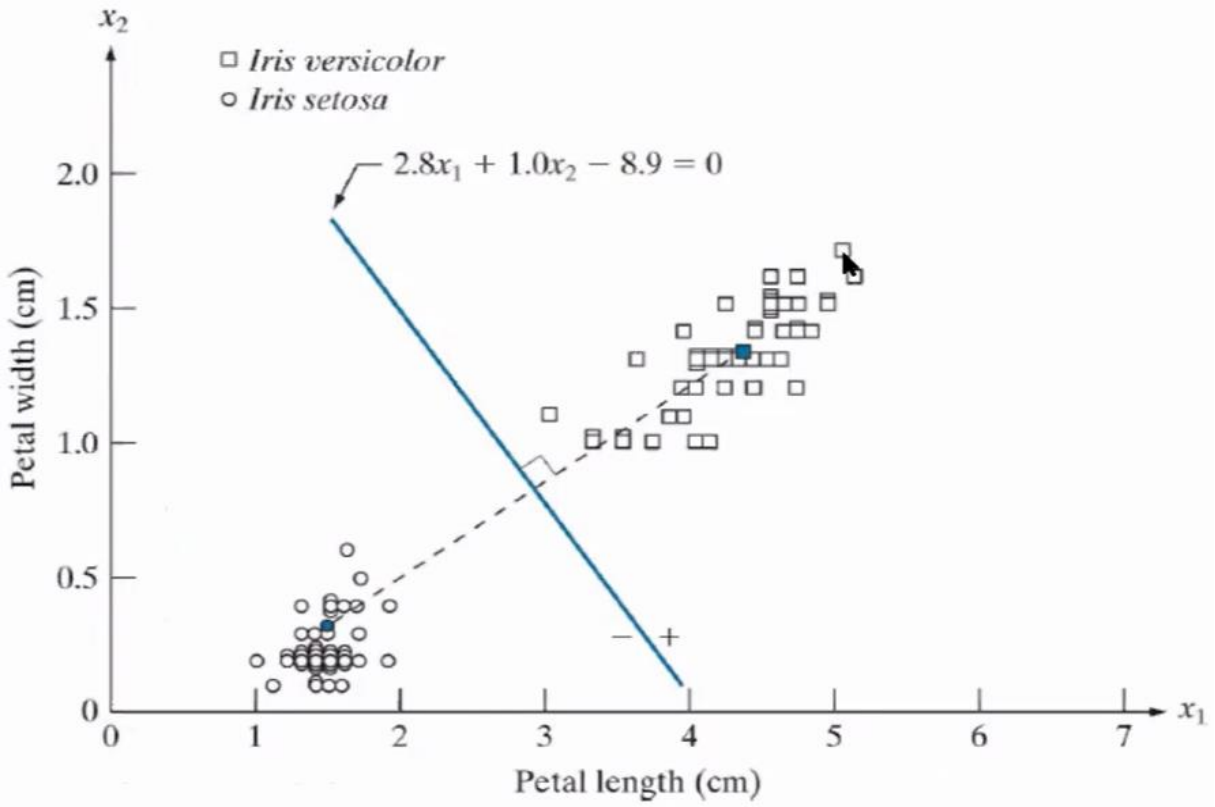
The Equation of decision boundary is

$D_{12}(x) = d_1(x) – d_2(x) = 0$

$= 4.3x_1 + 1.3x_2 – 10.1 - (1.5x_1 + 0.3x_2 – 1.17)$

$= 4.3x_1 + 1.3x_2 – 10.1 - 1.5x_1 - 0.3x_2 + 1.17$

$= 2.8x_1 + 1.0x_2 – 8.9 = 0$



Example 2: The two classes, Iris-versicolor and Iris-setosa, denoted ω_1 and ω_2 respectively, have sample mean vectors are $m_1 = (4.3, 1.3)^T$, and $m_2 = (1.5, 0.3)^T$. An unknown pattern P with vector $P = (4.4, 1.5)^T$ is found, find in which class pattern P belongs?

Solution:

Given, mean vector of two classes are

$m_1 = (4.3, 1.3)^T$

$m_2 = (1.5, 0.3)^T$

Unknown pattern is

$P = (4.4, 1.5)^T$

We know the decision function is

$d_n(x) = x^T m_n – ½ m_n^T m_n$

For first class i.e. $m_1 = (4.3, 1.3)^T$

$d_1(p) = p^T m_1 – ½ m_1^T m_1$

$= [4.4, 1.5] \begin{bmatrix} 4.3 \\ 1.3 \end{bmatrix} – ½ [4.3, 1.3] \begin{bmatrix} 4.3 \\ 1.3 \end{bmatrix}$

$= 4.4x4.3 + 1.5x1.3 – ½ (4.3x4.3 + 1.3x1.3)$

$= 18.9 + 1.9 – ½ (18.49 + 1.69)$

$= 35.9 – ½ (20.18)$

$= 35.9 – 10.1$

$= 25.8$

For second class i.e. $m_2 = (1.5, 0.3)^T$

$d_2(p) = p^T m_2 – ½ m_2^T m_2$

$= [4.4, 1.5] \begin{bmatrix}1.5\\0.3\end{bmatrix} – ½ [1.5, 0.3] \begin{bmatrix}1.5\\0.3\end{bmatrix}$

$= 4.4x1.5 + 1.5x0.3 – ½ (1.5x1.5 + 0.3x0.3)$

$= 6.6 + 0.45 – ½ (2.25 + 0.09)$

$= 7.05 – ½ (2.34)$

$= 7.05 – 1.17$

$= 5.88$

Now the $d_1(P) > d_2(P)$

Hence, Pattern P is belongs to first class i.e. Iris-versicolor.

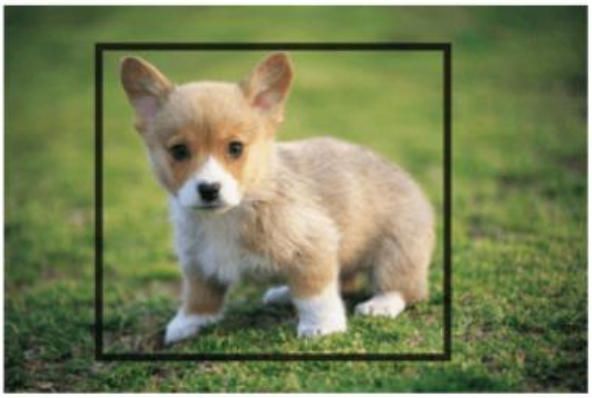
Neural Network Based Image Recognition

Image recognition

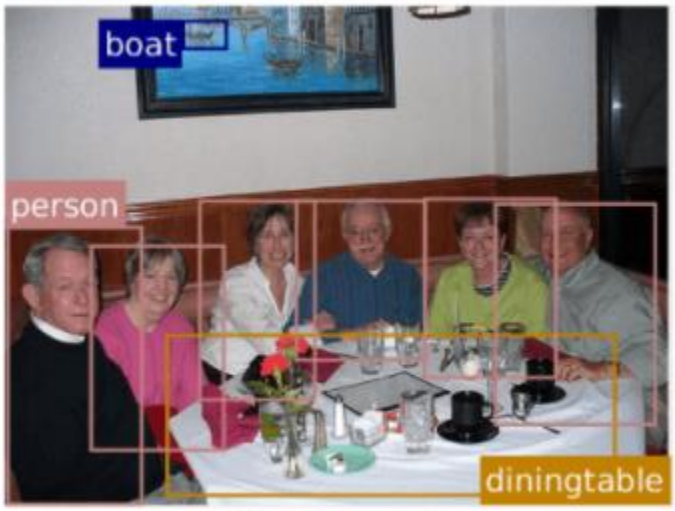
Image recognition (or image classification) is the task of identifying images and categorizing them in one of several predefined distinct classes. So, image recognition software and apps can define what's depicted in a picture and distinguish one object from another.

The field of study aimed at enabling machines with this ability is called ***computer vision.*** Being one of the computer vision (CV) tasks, image classification serves as the foundation for solving different CV problems, including:

Image classification with localization: placing an image in a given class and drawing a bounding box around an object to show where it's located in an image.



Object detection: categorizing multiple different objects in the image and showing the location of each of them with bounding boxes. So, it's a variation of the image classification with localization tasks for numerous objects.



Object Detection

Object (semantic) segmentation: identifying specific pixels belonging to each object in an image instead of drawing bounding boxes around each object as in object detection.

Semantic Segmentation

Instance segmentation: differentiating multiple objects (instances) belonging to the same class (each person in a group).

Instance Segmentation

Neural Network midterm ko paper

Researchers can use deep learning models for solving computer vision tasks. Deep learning is a machine learning technique that focuses on teaching machines to learn

by example. Since most deep learning methods use neural network architectures, deep learning models are frequently called deep neural networks.

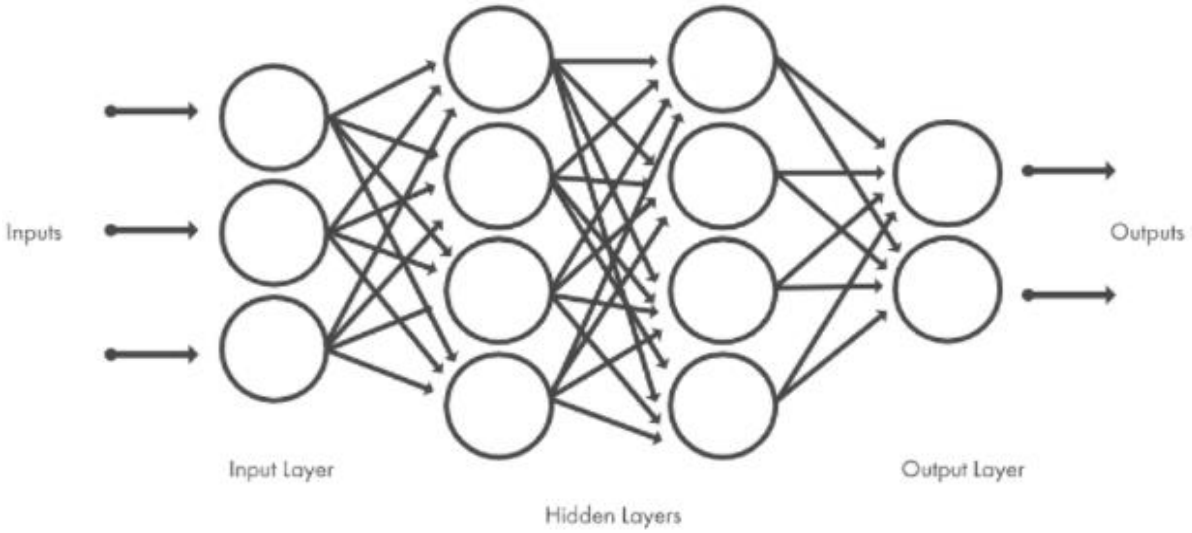
Image recognition is one of the tasks in which ***deep neural networks (DNNs)***. Neural networks are computing systems designed to recognize patterns. Their architecture is inspired by the human brain structure, hence the name.

They consist of three types of layers:

- Input Layers
- Hidden Layers
- Output Layers

The input layer receives a signal, the hidden layer processes it, and the output layer makes a decision or a forecast about the input data. Each network layer consists of interconnected nodes (*artificial neurons*) that do the computation.

While traditional neural networks have up to three hidden layers, deep networks may contain hundreds of them.



The architecture of a neural network, each layer consists of nodes

How neural networks learn to recognize patterns?

How do we understand whether a person passing by on the street is a familiar or a stranger? We look at them, subconsciously analyze their appearance, and if some

inherent features – face shape, eye color, hairstyle, body type, gait, or even fashion choices match with a specific person we know, we recognize this individual. This brainwork takes just a moment.

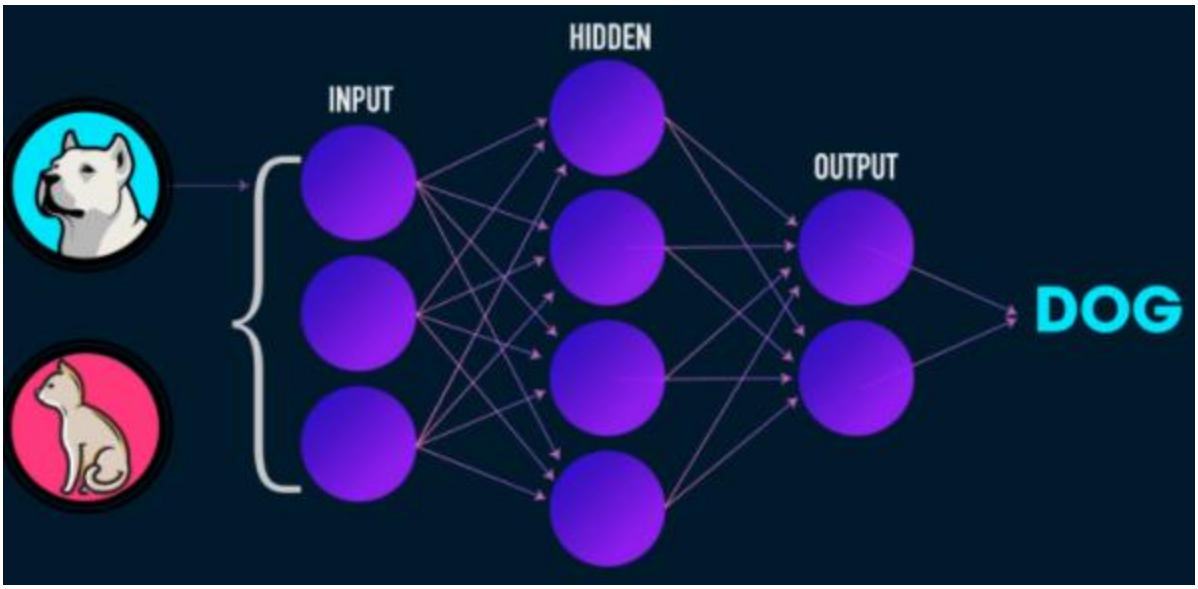
So, to be able to recognize faces, a system must learn their features first. It must be trained to predict whether an object is X or Y. Deep learning models learn these characteristics in a different way from machine learning (ML) models. That's why model training approaches are different as well.

Training deep learning models (such as neural networks)

To build an ML model that can, for instance, predict customer character, data scientists must specify what input features (problem properties) the model will consider in predicting a result. That may be a customer's education, income, lifecycle stage, product features, or modules used, number of interactions with customer support and their outcomes. The process of constructing features using domain knowledge is called *feature engineering.*

If we were to train a deep learning model to see the difference between a dog and a cat using feature engineering. Well, imagine gathering characteristics of billions of cats and dogs that live on this planet. We can't construct accurate features that will work for each possible image while considering such complications as viewpoint-dependent object variability, background clutter, lighting conditions, or image deformation. There should be another approach, which is the nature of neural networks. Neural networks learn features directly from data with which they are trained, so specialists don't need to extract features manually.

feature engineering bata garna garo vyerw neural networks lyako ho jaha aafae machine le learn garxa examples haru bbata natra feature engineering ma multiple objects haru liyerw aune ani aliali grdae feature tipdae note grdae basira hunxa



Each image is labeled with a category it belongs to – a cat or dog. The algorithm explores these examples, learns about the visual characteristics of each category, and eventually learns how to recognize each image class. This model training style is called *supervised learning*. suruam supervised learningmethod bata sikxa ani able hunxa learn garxa afae xutauna

Each layer of nodes trains on the output (feature set) produced by the previous layer. So, nodes in each successive layer can recognize more complex, detailed features visual representations of what the image depicts. Such a hierarchy of increasing complexity and abstraction is known as *feature hierarchy*.

So, the more layers the network has, the greater it's predictive capability.

Convolutional Neural Networks (CNNs)

The leading architecture used for image recognition and detection tasks is Convolutional Neural Networks (CNNs). Convolutional neural networks consist of several layers with small neuron collections, each of them perceiving small parts of an image. The results from all the collections in a layer partially overlap in a way to create the entire image representation. The layer below then repeats this process on the new image representation, allowing the system to learn about the image composition.

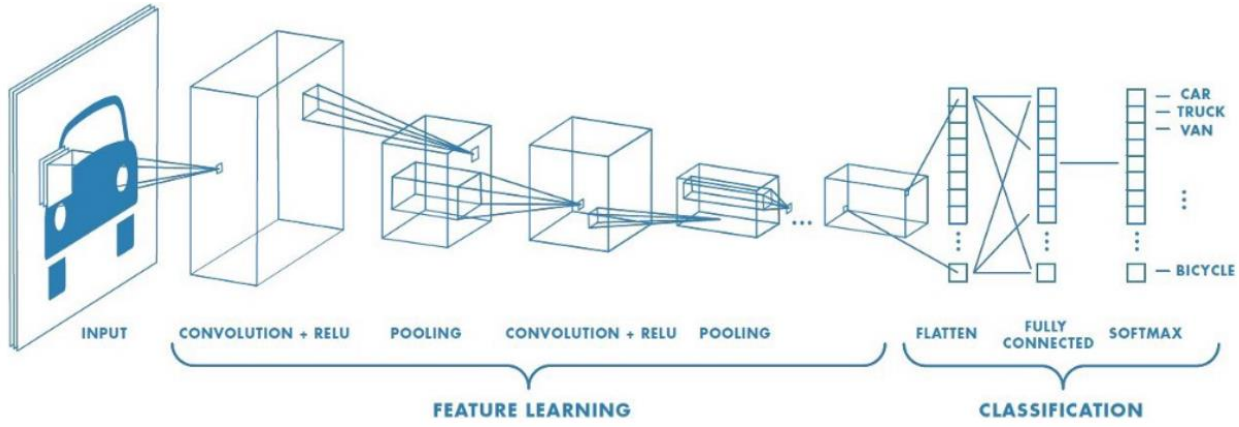
CNN as a machine learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other.

CNN works by extracting features from the images. Any CNN consists of the following:

1. The input layer which is a grayscale image
2. The Output layer which is a binary or multi-class labels
3. Hidden layers consisting of convolution layers, ReLU (rectified linear unit) layers, the pooling layers, and a fully connected Neural Network

It is very important to understand that ANN or Artificial Neural Networks, made up of multiple neurons is not capable of extracting features from the image. This is where a combination of convolution and pooling layers comes into the picture. Similarly, the convolution and pooling layers can't perform classification hence we need a fully connected Neural Network.

Let's consider that we have access to multiple images of different vehicles, each labeled into a truck, car, van, bicycle, etc. Now the idea is to take these pre-label/classified images and develop a machine learning algorithm that is capable of accepting a new vehicle image and classify it into its correct category or label.



When the image is read from accusation tool then it goes to convolution layer.

The convolution layer consists of one or more Kernels with different weights that are used to extract features from the input image. When we slide the Kernel over

the input image based on the weights of the Kernel we end up calculating features for different pixels based on their surrounding/neighboring pixel values.

ReLU or rectified linear unit is a process of applying an activation function to increase the non-linearity of the network without affecting the receptive fields of convolution layers. ReLU allows faster training of the data, whereas Leaky ReLU can be used to handle the problem of vanishing gradient.

The pooling layer applies a non-linear down-sampling on the convolved feature often referred to as the activation maps. This is mainly to reduce the computational complexity required to process the huge volume of data linked to an image. Pooling is not compulsory and is often avoided. Usually, there are two types of pooling, Max Pooling, that returns the maximum value from the portion of the image covered by the Pooling Kernel and the Average Pooling that averages the values covered by a Pooling Kernel.

Once the pooling is done the output needs to be converted to a tabular structure that can be used by an artificial neural network to perform the classification which is called image flattering.

After flattering the image result is goes to final classification and this process recognize the image.

End of Unit-7