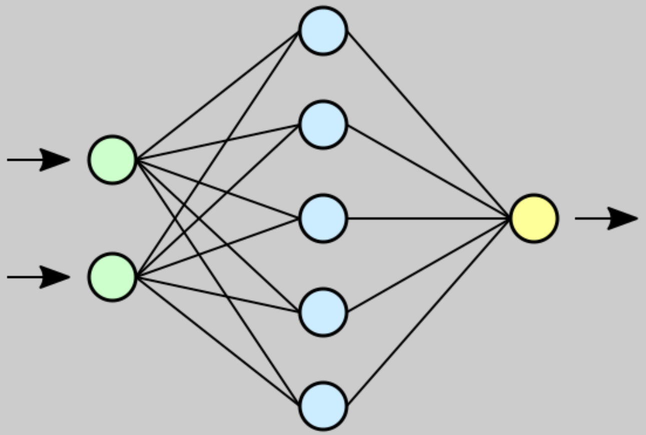
Deep Neural Networks

By:Pranav N Ghatigar
USN: ENG19CS0227
Dayananda Sagar University

Contents

Neural Networks

An artificial network consists of a pool of simple processing units which communicate by sending signals to each other over a large number of weighted connections.



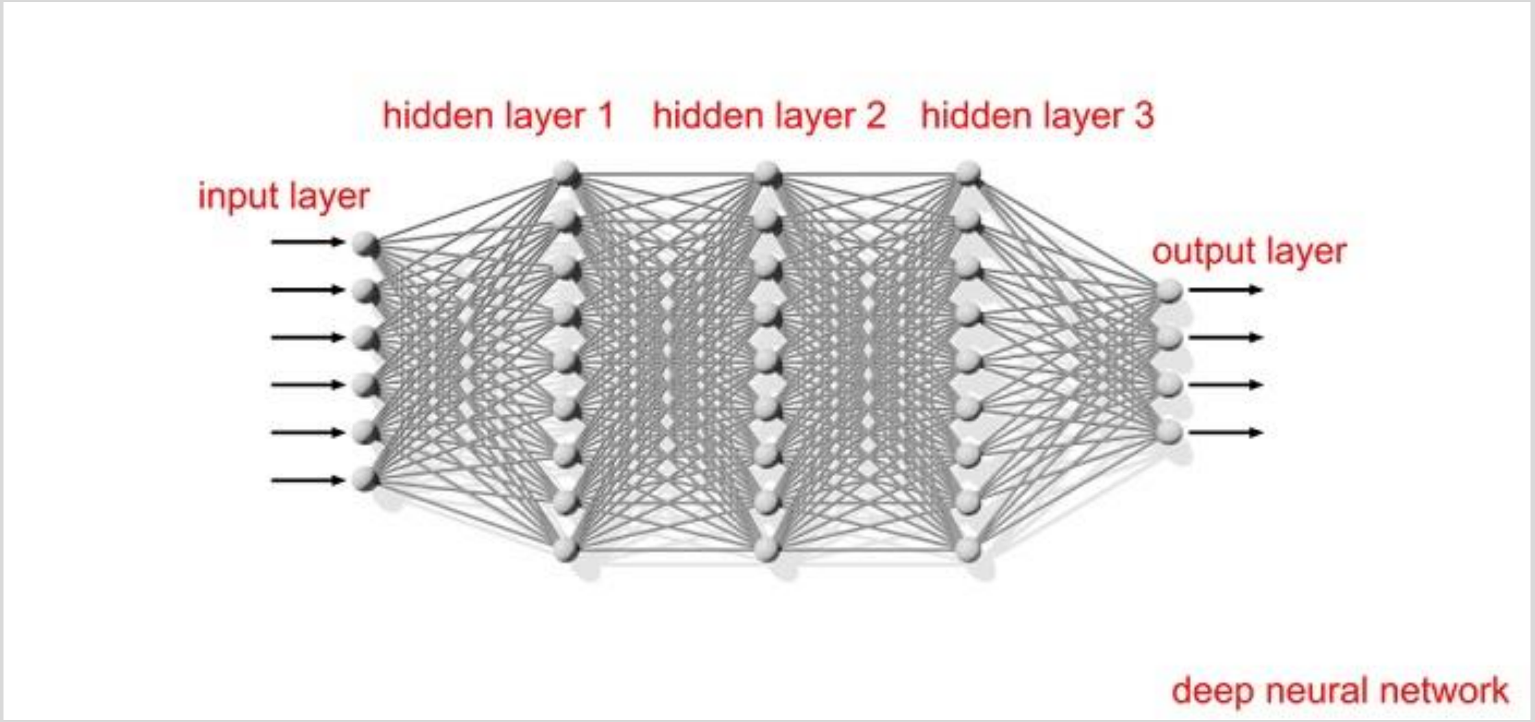
Deep Learning

- It is part of a broader family of Machine Learning.
- The adjective "deep" in deep learning refers to the use of multiple layers in the network.
- It uses various architectures:
    - Deep Neural Networks
    - Deep Reinforcement Learning
    - Recurrent Neural Networks
    - Convolution Neural Networks

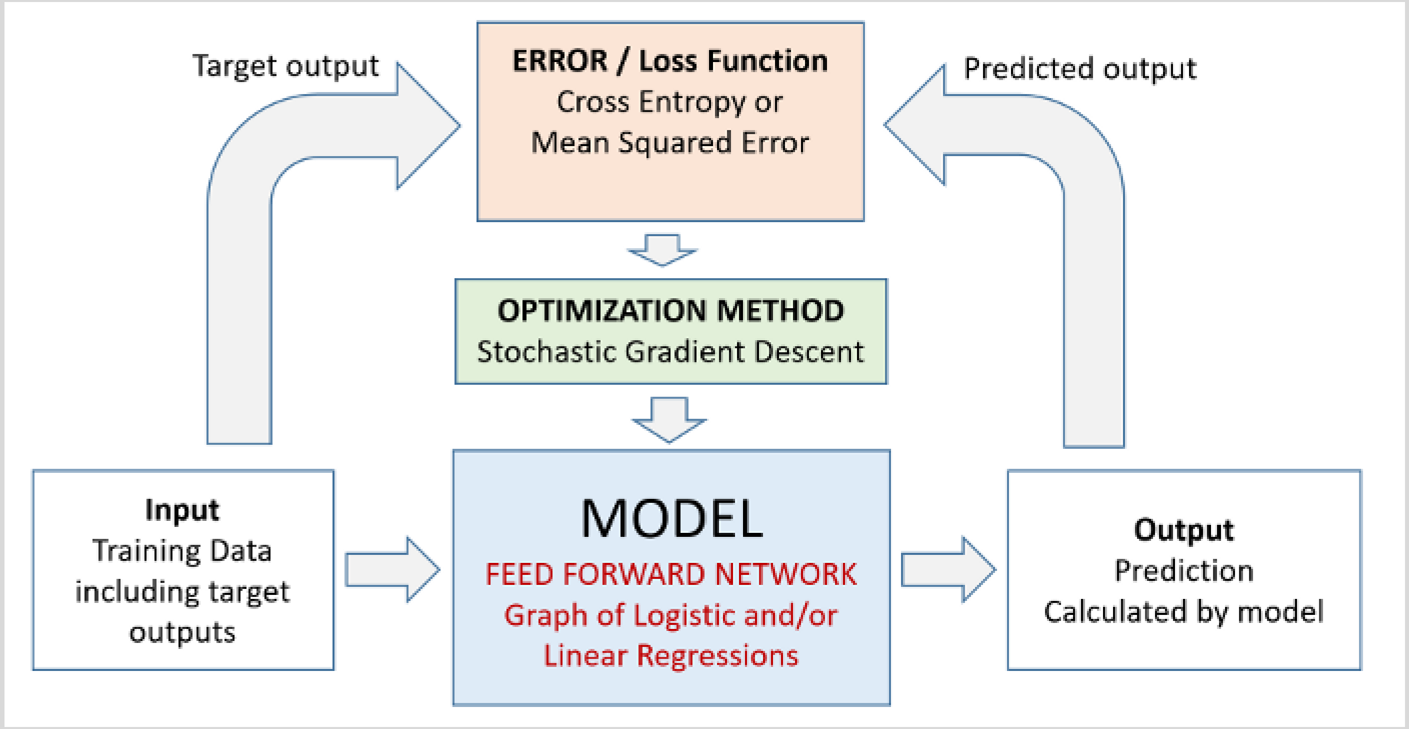
Deep Neural Networks

- A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers.
- There are different types of neural networks but they always consist of the same components:
  - neurons,
  - synapses,
  - weights,
  - biases,
  - functions.

DNN


hidden layer 1
hidden layer 2
hidden layer 3
input layer
output layer
deep neural network

How it works?


Target output
ERROR / Loss Function
Cross Entropy or
Mean Squared Error
Predicted output
OPTIMIZATION METHOD
Stochastic Gradient Descent
Input
Training Data
including target
outputs
MODEL
FEED FORWARD NETWORK
Graph of Logistic and/or
Linear Regressions
Output
Prediction
Calculated by model

Types

1. Forward feed NN
2. Convolution NN
3. Recurrent NN
4. Generative Adversarial Network

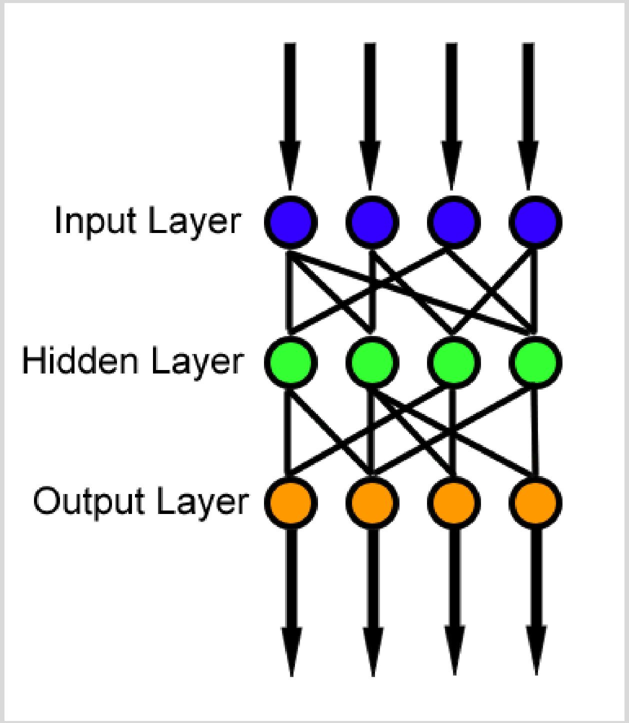
Feed Forward Network

A feedforward neural network is an artificial neural network wherein connections between the nodes do not form a cycle.

It is the most simplest and the first artificial neural network devised.

There are **no** cycles or loops in the network.

One direction



Convolution Neural Network

A convolutional neural network (CNN) is a class of artificial neural network, which is most commonly applied to analyze visual imagery.

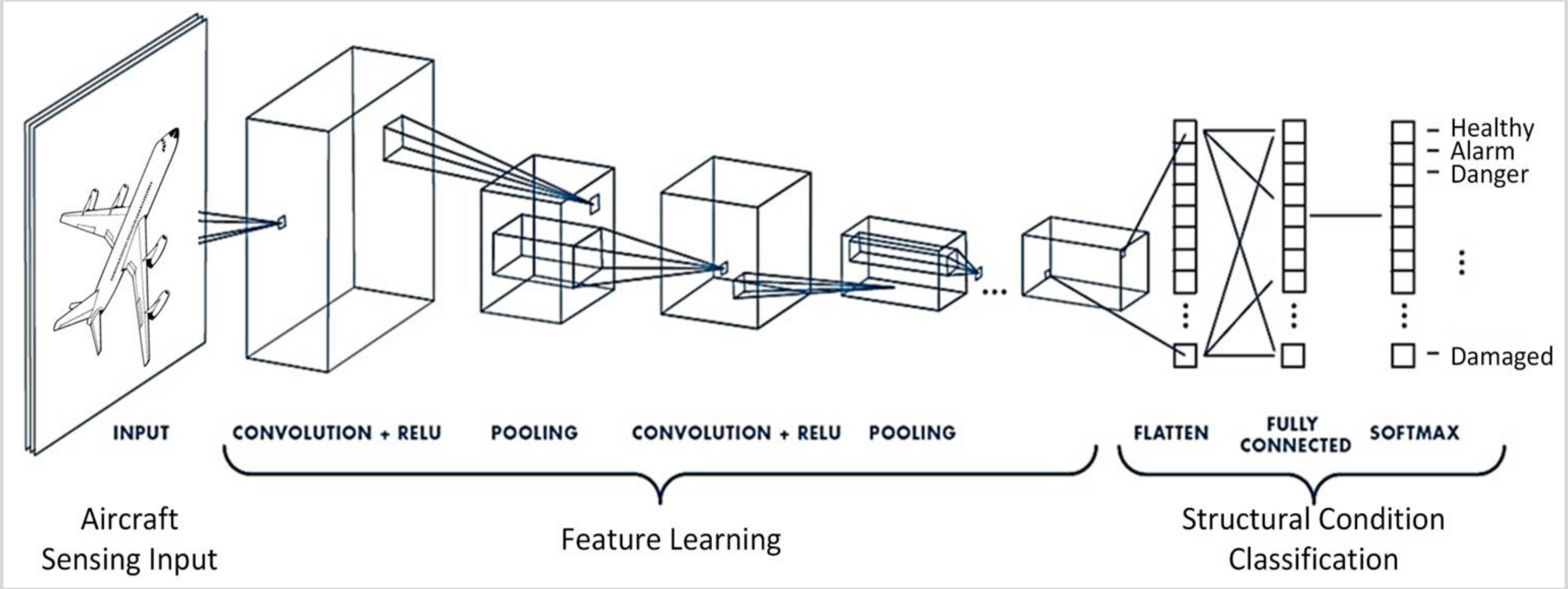
Convolutional networks are a specialized type of neural networks that use convolution in place of general matrix multiplication in at least one of their layers.

These networks are used in many fields but mostly computer vision.

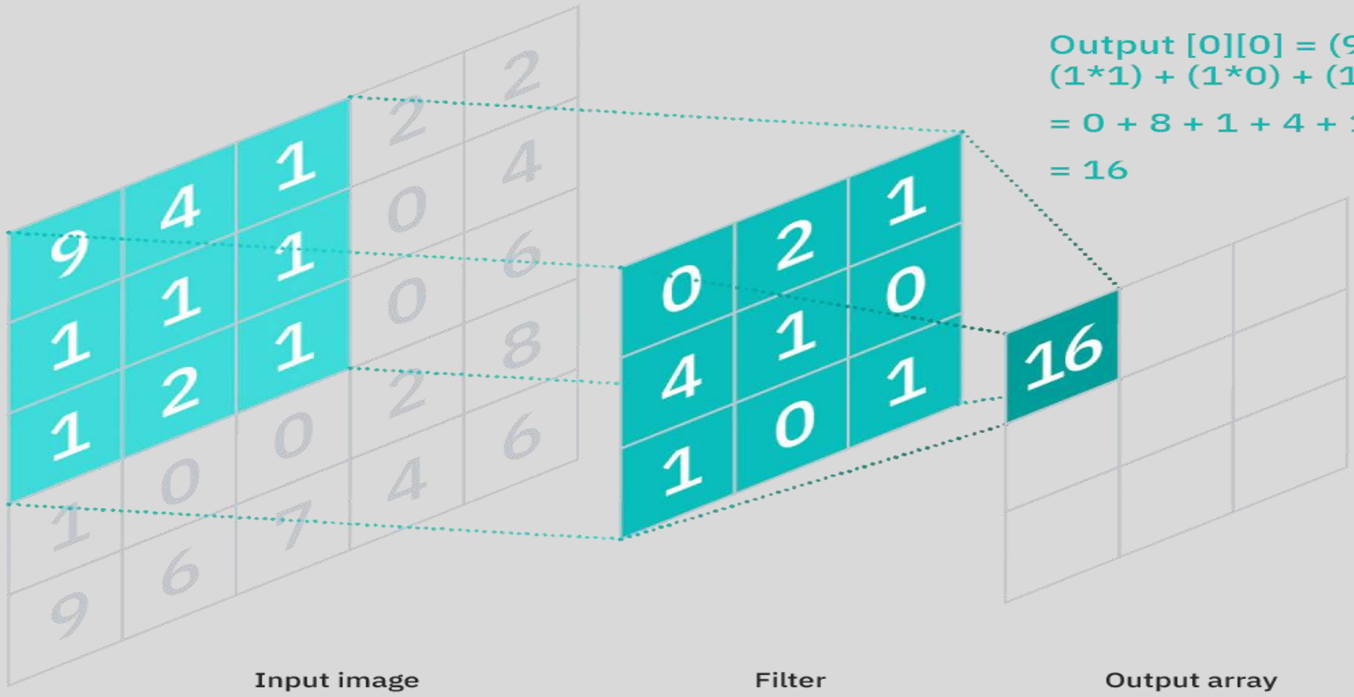
Architecture of CNN

There are five components:

I. Convolution: To extract features from an image
II. ReLU: To smoothen the image and make boundaries distinct
III. Pooling: Helps fix distorted images
IV. Flattening: To turn image into a suitable representation
V. Full Connection: To process the data into a neural network


INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
Healthy
Alarm
Danger
Damaged
Aircraft Sensing Input
Feature Learning
Structural Condition Classification

Convolution

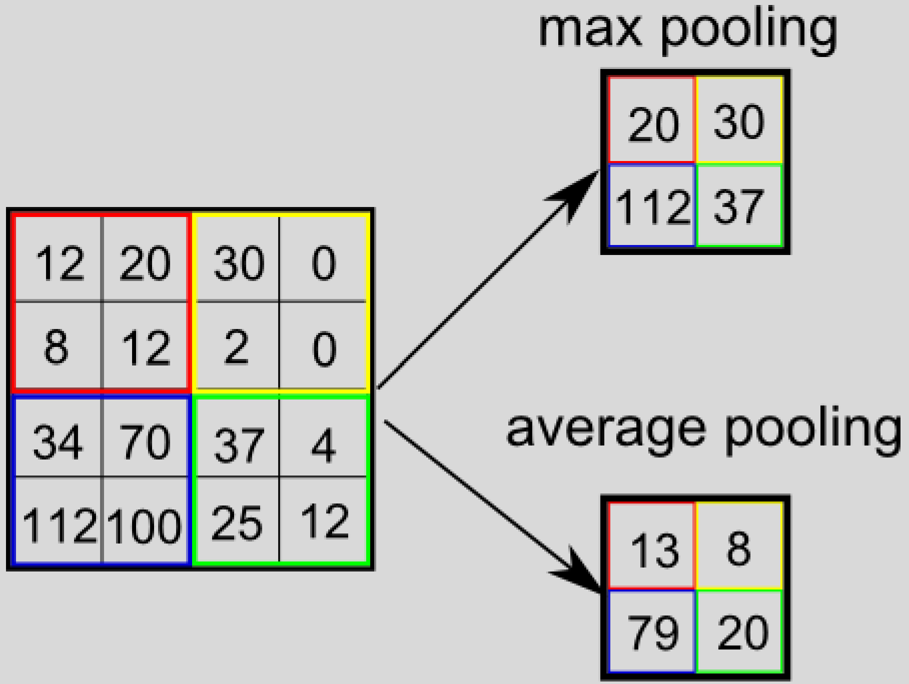


Output [0][0] = (9*0) + (4*2) + (1*4) + (1*1) + (1*0) + (1*1) + (2*0) + (1*1)

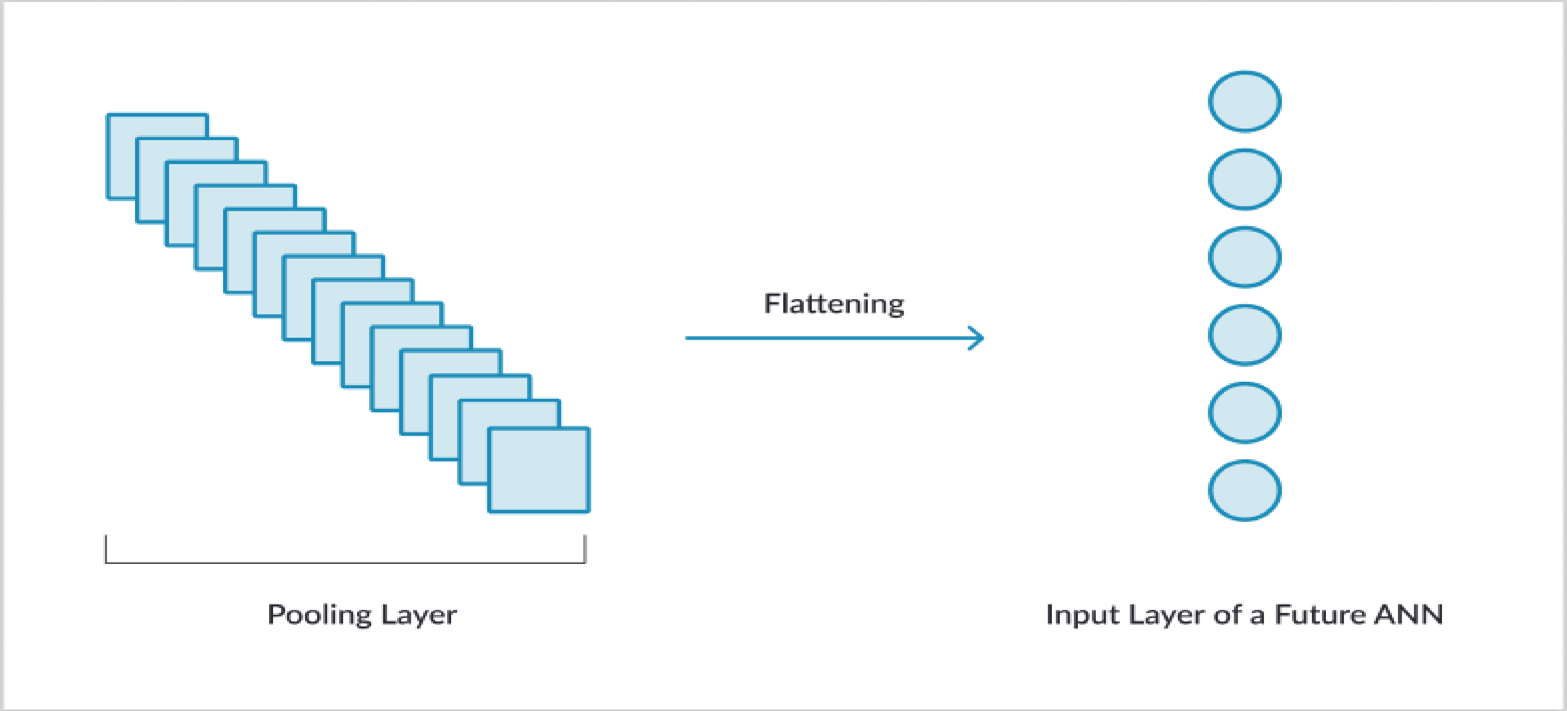
= 0 + 8 + 1 + 4 + 1 + 0 + 1 + 0 + 1

= 16

Pooling


max pooling
average pooling
12
20
30
0
8
12
2
0
34
70
37
4
112
100
25
12
20
30
112
37
13
8
79
20

Flattening



Applications of CNN

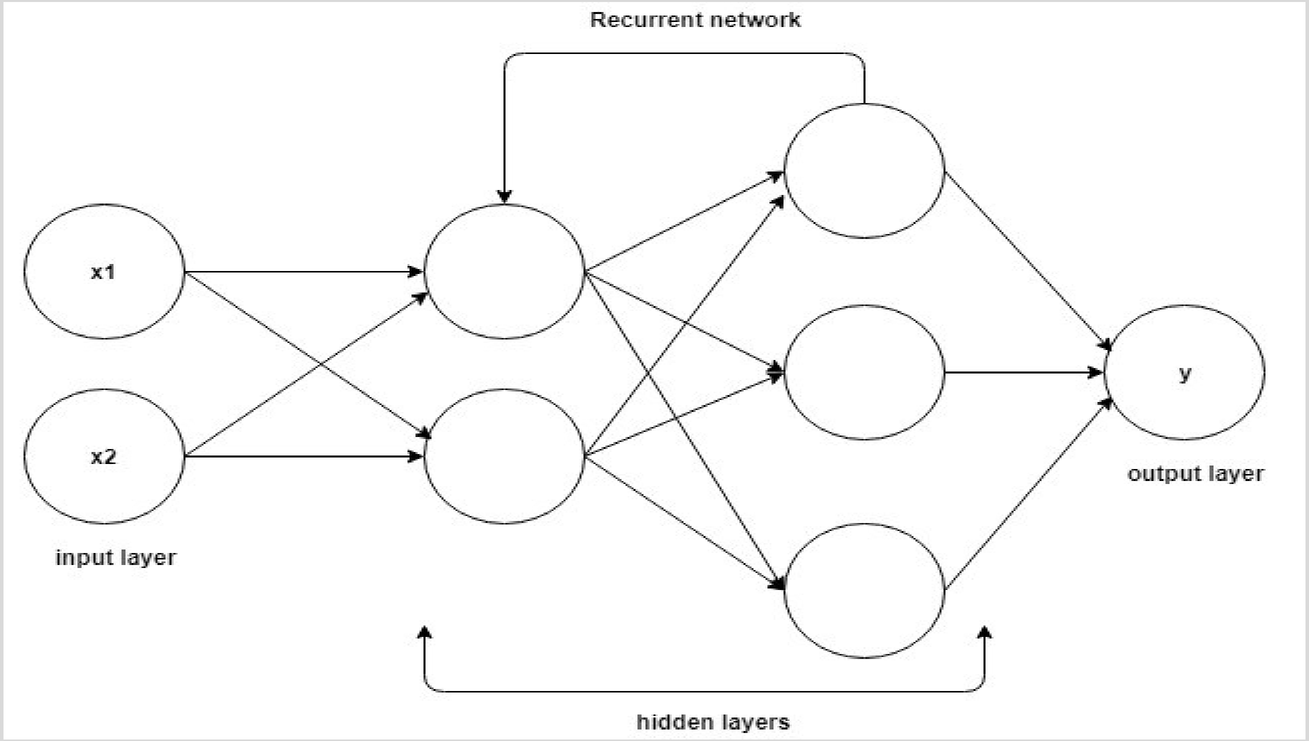
- ➔ Decoding Facial Recognition
- ➔ Computer Vision
- ➔ Image Tagging
- ➔ Driverless Vehicles

Recurrent Neural Network

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence.

RNNs can use their internal state (memory) to process variable length sequences of inputs.

The most widely used domain is Natural Language Processing


Recurrent network
x1
x2
input layer
y
output layer
hidden layers

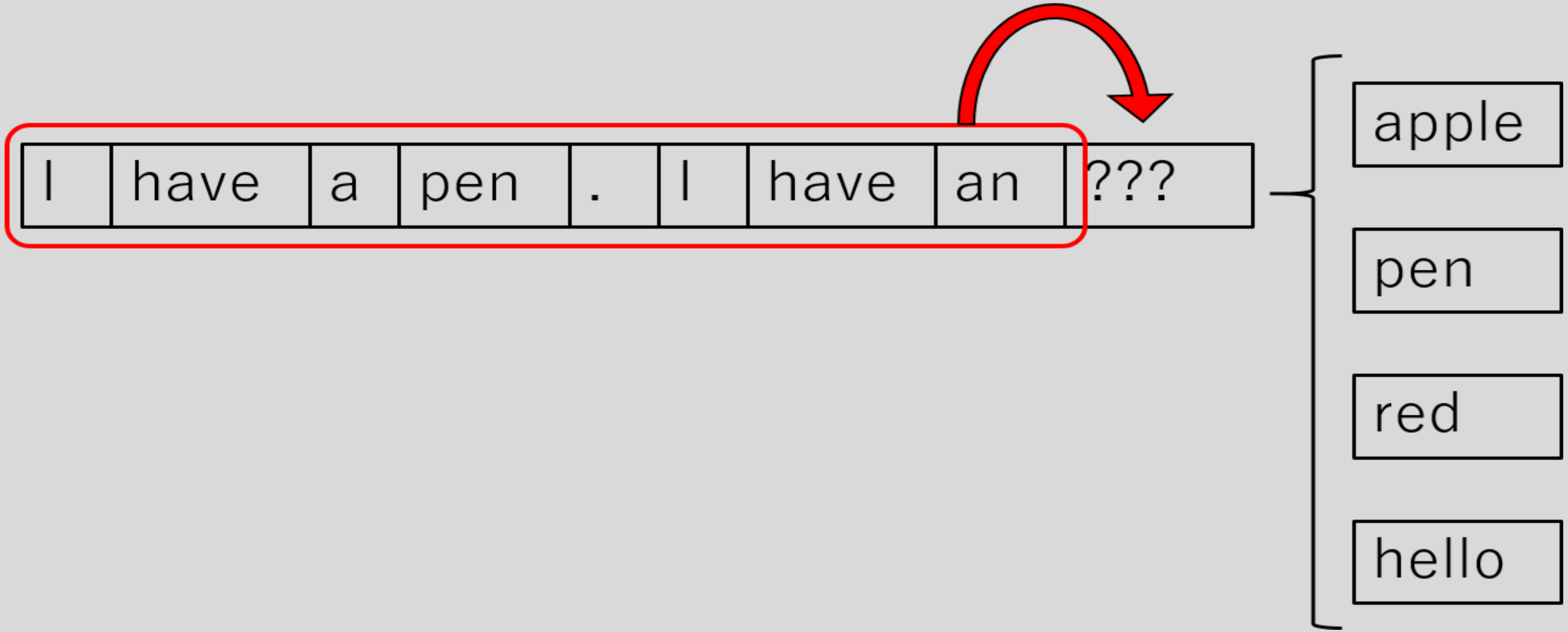
Example

Using LSTM, we can predict the next word in a sentence much like autocomplete feature.

Suppose I type, "I am from Bengaluru, I speak ________"

The algorithm will predict Kannada. This is done using LSTM and RNN.

Example



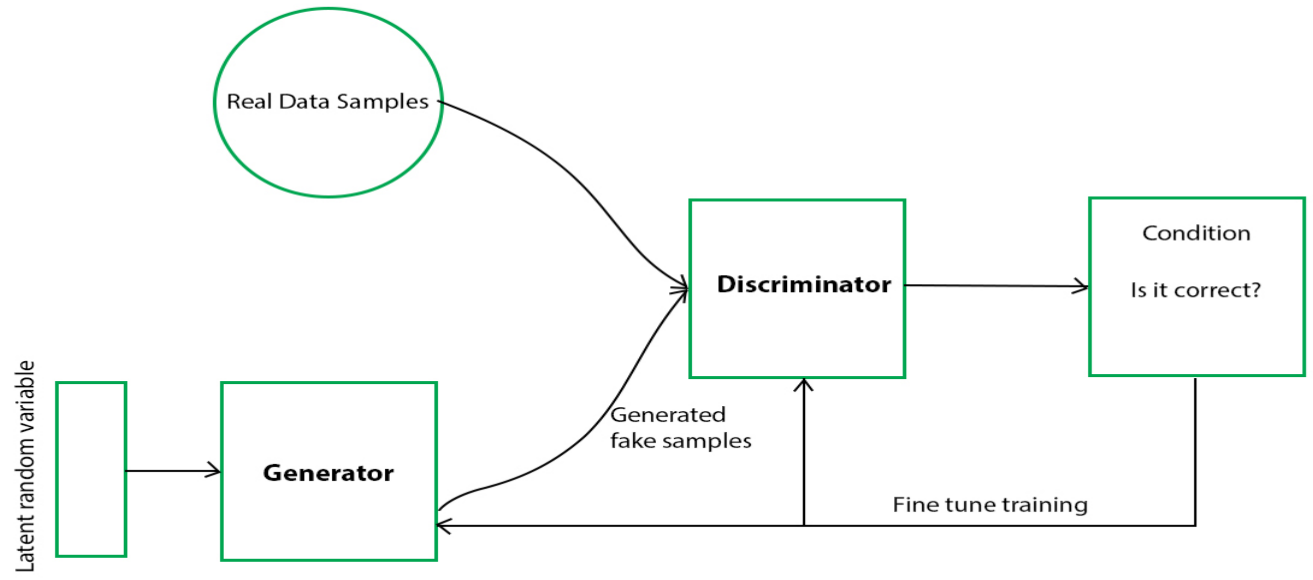
Applications of RNN

- ➔ Speech Detection
- ➔ Call Centre Analysis
- ➔ Generating Image Description
- ➔ Prediction Problems

Generative Adversarial Network

Generative Adversarial Networks, or GANs for short, are an approach to generative modeling using deep learning methods, such as convolutional neural networks.

Generative modeling is an unsupervised learning task in machine learning that involves automatically discovering and learning the regularities or patterns in input data in such a way that the model can be used to generate or output new examples that plausibly could have been drawn from the original dataset.


Real Data Samples
Discriminator
Condition
Is it correct?
Latent random variable
Generator
Generated fake samples
Fine tune training

GAN

GANs are a clever way of training a generative model by framing the problem as a supervised learning problem with two sub-models:

> The Generator model that we train to generate new examples, and
>
> The Discriminator model that tries to classify examples as either real or fake (generated).

The two models are trained together in a zero-sum game, adversarial, until the discriminator model is fooled about half the time, meaning the generator model is generating plausible examples.

Application of GAN

- ➔ Image-to-Image Translation
- ➔ Text-to-Image Translation
- ➔ 3D Object Generation
- ➔ Entertainment Industry with CGI

Handwritten Digits

Now we will be writing a code using CNN to recognise handwritten digits.

Firstly, we import all necessary libraries,

- Numpy
- Pandas
- Tensorflow
- Matplotlib
- Keras
- Sklearn

```
import numpy as np

import pandas as pd

import random

import tensorflow as tf

import matplotlib.pyplot as plt

from sklearn.metrics import accuracy_score

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Flatten, Conv2D, Dense, MaxPooling2D

from tensorflow.keras.optimizers import SGD

from tensorflow.keras.utils import to_categorical

from tensorflow.keras.datasets import mnist
```

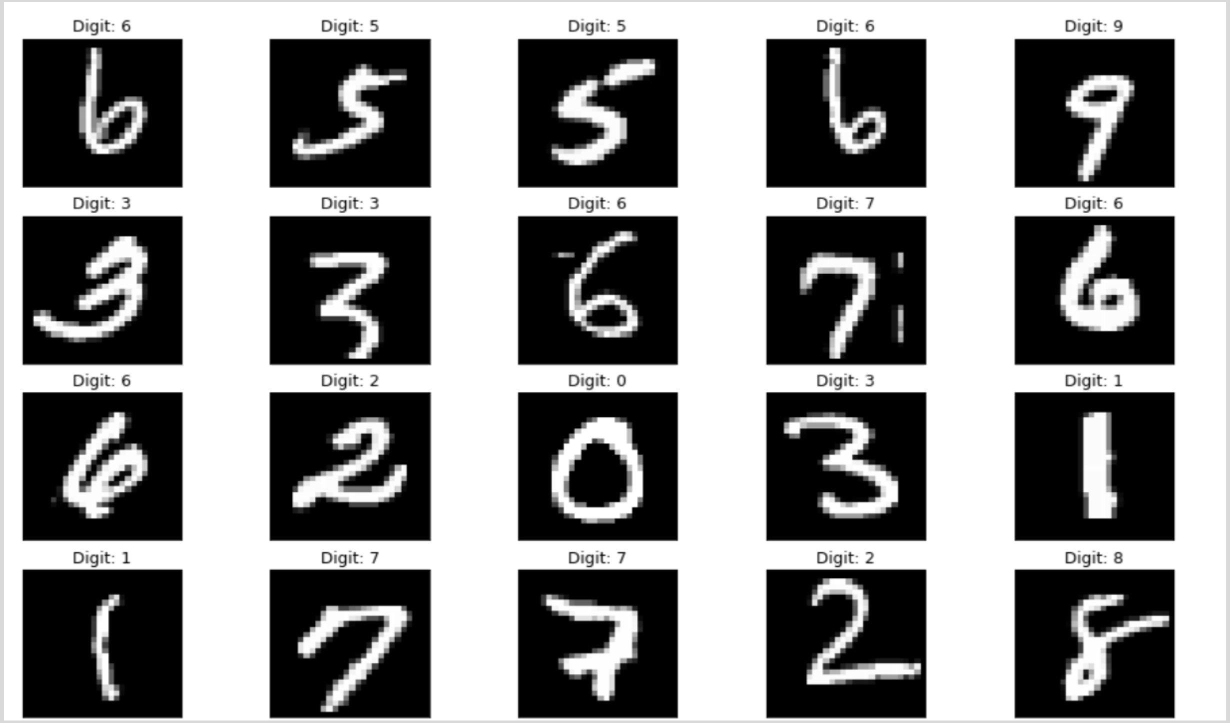
```
(X_train, y_train), (X_test, y_test) = mnist.load_data()

print(X_train.shape)

X_train[0].min(), X_train[0].max()

X_train = (X_train - 0.0) / (255.0 - 0.0)

X_test = (X_test - 0.0) / (255.0 - 0.0)


X_train[0].min(), X_train[0].max()
```

```
def plot_digit(image, digit, plt, i):
    plt.subplot(4, 5, i + 1)
    plt.imshow(image, cmap=plt.get_cmap('gray'))
    plt.title(f"Digit: {digit}")
    plt.xticks([])
    plt.yticks([])
plt.figure(figsize=(16, 10))
for i in range(20):
    plot_digit(X_train[i], y_train[i], plt, i)
plt.show()
X_train = X_train.reshape((X_train.shape + (1,)))
X_test = X_test.reshape((X_test.shape + (1,)))
y_train[0:20]
```

```
model = Sequential([
    Conv2D(32, (3, 3), activation="relu", input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(100, activation="relu"),
    Dense(10, activation="softmax")])
optimizer = SGD(learning_rate=0.01, momentum=0.9)
model.compile(
    optimizer=optimizer,
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"])
model.summary()
odel.fit(X_train, y_train, epochs=10, batch_size=32)
```

```
plt.figure(figsize=(16, 10))
for i in range(20):
    image = random.choice(X_test).squeeze()
    digit = np.argmax(model.predict(image.reshape((1, 28, 28, 1)))[0], axis=-1)
    plot_digit(image, digit, plt, i)
plt.show()
```

Output


Digit: 6
Digit: 5
Digit: 5
Digit: 6
Digit: 9
Digit: 3
Digit: 3
Digit: 6
Digit: 7
Digit: 6
Digit: 6
Digit: 2
Digit: 0
Digit: 3
Digit: 1
Digit: 1
Digit: 7
Digit: 7
Digit: 2
Digit: 8

Thank you