By Shweta Tiwari from IT Department

Notes of "Artificial Intelligence for Engineering/Engineers(KMC-101)"

January, 2022

Notes Part-1

MS. SHWETA TIWARI

Rajkiya Engineering College | Ambedkar Nagar, UP, India

Faculty Name: Miss. Shweta Tiwari, Subject: Artificial Intelligence for Engineering (KMC101)
Year- 1st Year, Semester- 1st Sem, Branch- EE and IT, Session- Odd Semester (2021-22)

Artificial Intelligence for Engineering

UNIT-4

January, 2022

Notes Part-1

MS. SHWETA TIWARI

UNIT-IV

UNIT 4: ARTIFICIAL NEURAL NETWORKS

January, 2022

Notes Part-1

MS. SHWETA TIWARI

Artificial Neural Networks

Artificial intelligence (AI), also known as machine intelligence, is a branch of computer science that aims to imbue software with the ability to analyze its environment using either predetermined rules and search algorithms, or pattern recognizing machine learning models, and then make decisions based on those analyses.

Basic Structure of ANNs

The idea of ANNs is based on the belief that working of human brain by making the right connections, can be imitated using silicon and wires as living neurons and dendrites.

The human brain is composed of 86 billion nerve cells called neurons. They are connected to other thousand cells by Axons. Stimuli from external environment or inputs from sensory organs are accepted by dendrites. These inputs create electric impulses, which quickly travel through the neural network. A neuron can then send the message to other neuron to handle the issue or does not send it forward.

January, 2022

Notes Part-1
MS. SHWETA TIWARI

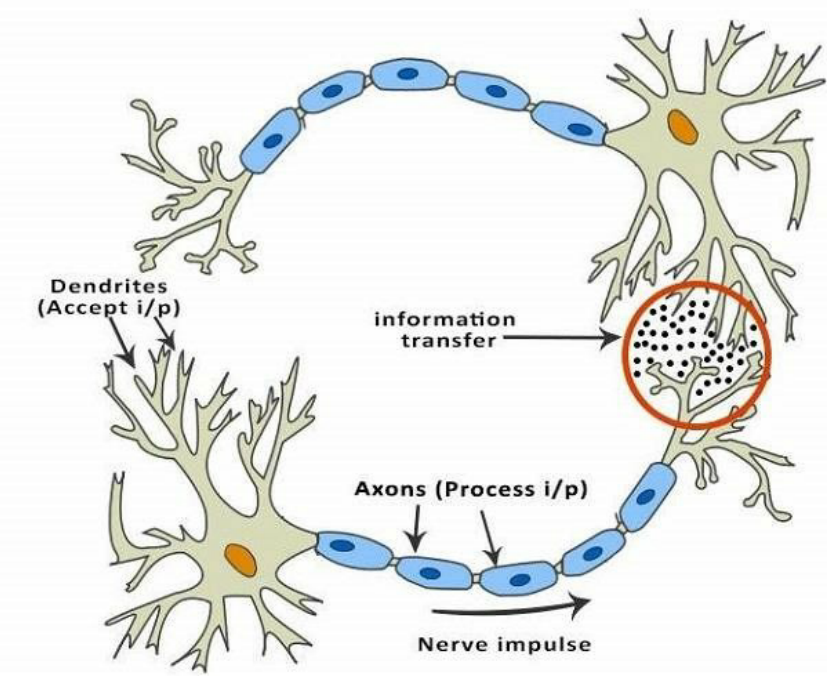


Figure 4.1: **Neuron**

ANNs are composed of multiple **nodes**, which imitate biological **neurons** of human brain. The neurons are connected by links and they interact with each other. The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons. The output at each node is called its **activation** or **node value.**

Each link is associated with **weight.** ANNs are capable of learning, which takes place by altering weight values. The following illustration shows a simple ANN –

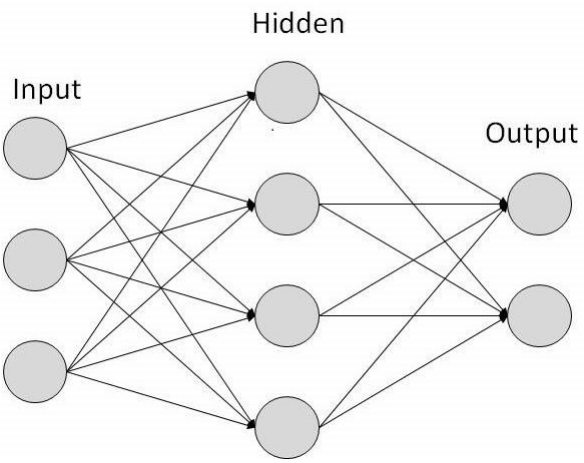


Figure 4.2: ANN

Types of Artificial Neural Networks

There are two Artificial Neural Network topologies − **FeedForward** and **Feedback.**

FeedForward ANN

In this ANN, the information flow is unidirectional. A unit sends information to other unit from which it does not receive any information. There are no feedback loops. They are used in pattern generation/recognition/classification. They have fixed inputs and outputs.

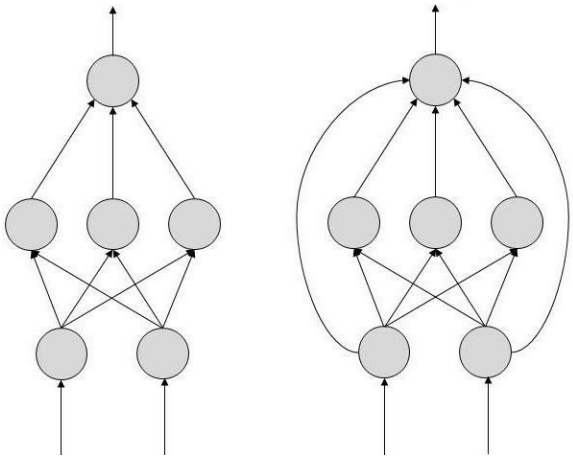


Figure 4.2: FeedForward ANN

FeedBack ANN

Here, feedback loops are allowed. They are used in content addressable memories.

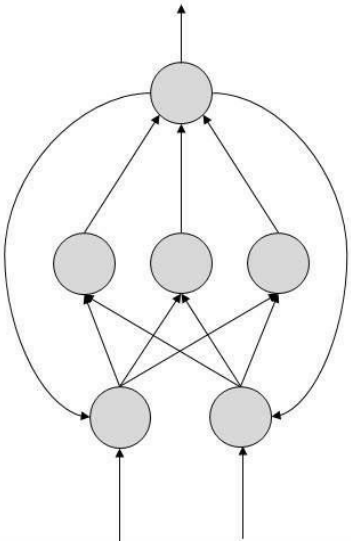


Figure 4.3: FeedBack ANN

4.1. Deep Learning

Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. In deep learning, we don't need to explicitly program everything. The concept of deep learning is not new. It has been around for a couple of years now. It's on hype nowadays because earlier we did not have that much processing power and a lot of data

Architectures :

1. **Deep Neural Network** – It is a neural network with a certain level of complexity (having multiple hidden layers in between input and output layers). They are capable of modeling and processing non-linear relationships.

2. **Deep Belief Network(DBN)** – It is a class of Deep Neural Network. It is multi-layer belief networks.

Steps for performing DBN :

a. Learn a layer of features from visible units using Contrastive Divergence algorithm.

b. Treat activations of previously trained features as visible units and then learn features

of features.

c. Finally, the whole DBN is trained when the learning for the final hidden layer is achieved.

3. **Recurrent** (perform same task for every element of a sequence) **Neural Network** – Allows for parallel and sequential computation. Similar to the human brain (large feedback network of connected neurons). They are able to remember important things about the input they received and hence enables them to be more precise.

BLANK PAGE

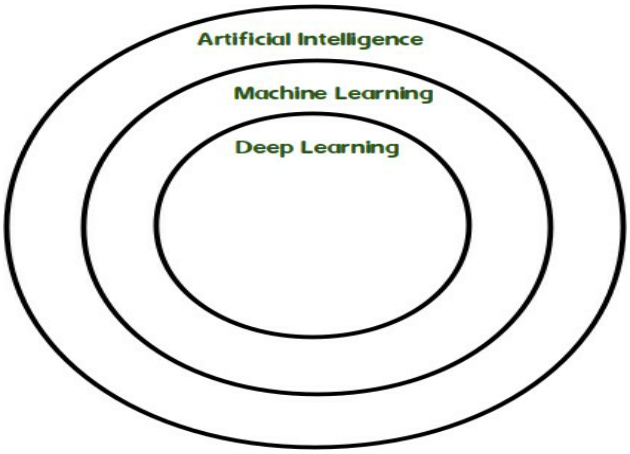


Figure 4.4: Deep Learning

Working:

First, we need to identify the actual problem in order to get the right solution and it should be understood, the feasibility of the Deep Learning should also be checked (whether it should fit Deep Learning or not). Second, we need to identify the relevant data which should correspond to the actual problem and should be prepared accordingly. Third, Choose the Deep Learning Algorithm appropriately. Fourth, Algorithm should be used while training the dataset. Fifth, Final testing should be done on the dataset.

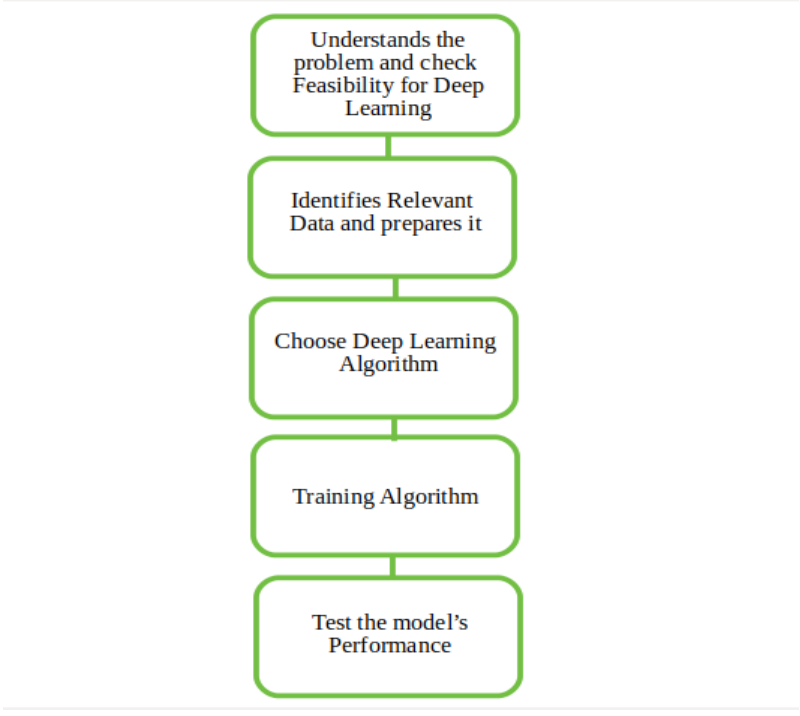


Figure 4.5: Deep Learning Working

January, 2022

Notes Part-1

MS. SHWETA TIWARI

4.2. Recurrent Neural Networks

Recurrent Neural Network(RNN) are a type of Neural Network where the **output from previous step are fed as input to the current step**. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is **Hidden state**, which remembers some information about a sequence.

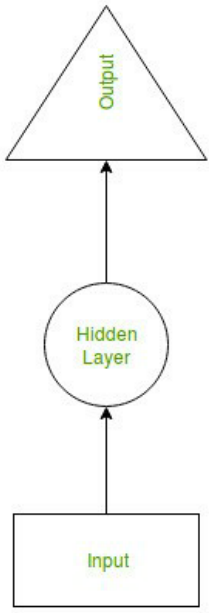


Figure 4.6: Recurrent Neural Networks

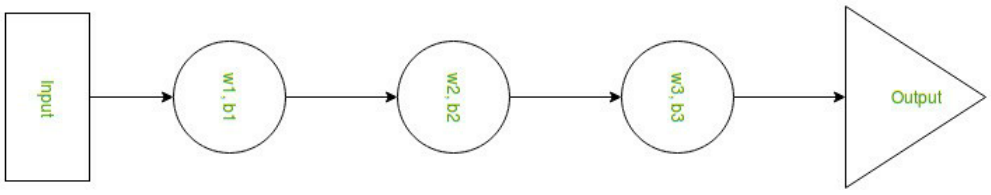
RNN have a **"memory"** which remembers all information about what has been calculated. It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks.

How RNN works

The working of a RNN can be understood with the help of below example:

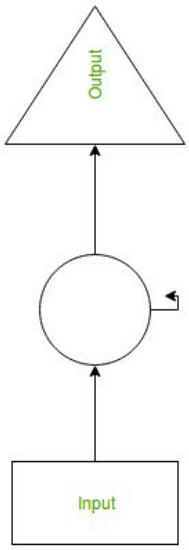
Example:

Suppose there is a deeper network with one input layer, three hidden layers and one output layer. Then like other neural networks, each hidden layer will have its own set of weights and biases, let's say, for hidden layer 1 the weights and biases are (w1, b1), (w2, b2) for second hidden layer and (w3, b3) for third hidden layer. This means that each of these layers are independent of each other, i.e. they do not memorize the previous outputs.



Now the RNN will do the following:

- RNN converts the independent activations into dependent activations by providing the same weights and biases to all the layers, thus reducing the complexity of increasing parameters and memorizing each previous outputs by giving each output as input to the next hidden layer.
- Hence these three layers can be joined together such that the weights and bias of all the hidden layers is the same, into a single recurrent layer.



- **Formula for calculating current state:**

$$h_t = f(h_{t-1}, x_t)$$

where:

h_t -> current state

h_{t-1} -> previous

statex_t -> input

- **Formula for applying Activation function(tanh):**

$$h_t = \tanh (W_{hh}h_{t-1} + W_{xh}x_t)$$

where:

w_{hh} -> weight at recurrent

neuronw_{xh} -> weight at input

- **Formula for calculating output:**

$$y_t = W_{hy}h_t$$

Y_t -> output

W_{hy} -> weight at output layer

4.3. Convolutional Neural Networks

Convolutional Neural Networks are very similar to ordinary Neural Networks from the previous chapter: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply.

Architecture Overview

Recall: Regular Neural Nets. As we saw in the previous chapter, Neural Networks receive an input (a single vector), and transform it through a series of hidden layers. Each hidden layer is made up of a set of neurons, where each neuron is fully connected to all neurons in the previous layer, and where neurons in a single layer function completely independently and do not share any connections. The last fully-connected layer is called the "output layer" and in classification settings it represents the class scores.

Regular Neural Nets don't scale well to full images. In CIFAR-10, images are only of size 32x32x3 (32 wide, 32 high, 3 color channels), so a single fully-connected neuron in a first hidden layer of a regular Neural Network would have 32*32*3 = 3072 weights. This amount still seems manageable, but clearly this fully-connected structure does not scale to larger images. For example, an image of more respectable size, e.g. 200x200x3, would lead to neurons that have 200*200*3 = 120,000 weights. Moreover, we would almost certainly want to have several such neurons, so the parameters would add up quickly! Clearly, this full connectivity is wasteful and the huge number of parameters would quickly lead to overfitting.

3D volumes of neurons.

Convolutional Neural Networks take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth. (Note that the word depth here refers to the third dimension of an activation volume, not to the depth of a full Neural Network, which can refer to the total number of layers in a network.) For example, the input images in CIFAR-10 are an input volume of activations, and the volume has dimensions 32x32x3 (width, height, depth respectively). As we will soon see, the neurons in a layer will only be connected to a small region of the layer before it, instead of all of the neurons in a fully-

January, 2022

Notes Part-1

MS. SHWETA TIWARI

connected manner. Moreover, the final output layer would for CIFAR-10 have dimensions 1x1x10, because by the end of the ConvNet architecture we will reduce the full image into a single vector of class scores, arranged along the depth dimension. Here is a visualization

Figure 4.7: Convolutional Neural Networks

4.4. The Universal Approximation Theorem

Mathematically speaking, any neural network architecture aims at finding any mathematical function y= f(x) that can map attributes(x) to output(y). The accuracy of this function i.e. mapping differs depending on the distribution of the dataset and the architecture of the network employed. The function f(x) can be arbitrarily complex. The Univeral Approximation Theorem tells us that Neural Networks has a kind of **universality** i.e. no matter what f(x) is, there is a network that can approximately approach the result and do the job! This result holds for any number of inputs and outputs.

If we observe the neural network above, considering the input attributes provided as height and width, our job is to predict the gender of the person. If we exclude all the activation layers from the above network, we realize that h_1 is a linear function of both weight and height with parameters w_1, w_2, and the bias term b_1. Therefore mathematically,

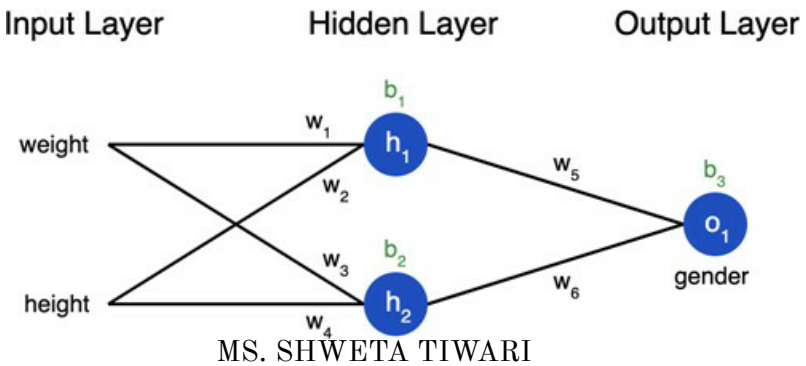


Figure 4.8: Universal Approximation

$h_1 = w_1 * \text{weight} + w_2 * \text{height} + b_1$

Similarily,

$h2 = w_3 * \text{weight} + w_4 * \text{height} + b_2$

Going along these lines we realize that o1 is also a linear function of h_1 and h2, and therefore depends linearly on input attributes weight and height as well. This essentially boils down to a linear regression model. Does a linear function suffice at approaching the Universal Approximation Theorem? The answer is NO. This is where activation layers come into play. An activation layer is applied right after a linear layer in the Neural Network to provide non-linearities. Non-linearities help Neural Networks perform more complex tasks. An activation layer operates on activations (h_1, h2 in this case) and modifies them according to the activation function provided for that particular activation layer. Activation functions are generally non-linear except for the identity function. Some commonly used activation functions are ReLu, sigmoid, softmax, etc. With the introduction of non- linearity's along with linear terms, it becomes possible for a neural network to model any given function approximately on having appropriate parameters(w_1, w_2, b_1, etc in this case). The parameters converge to appropriateness on training suitably. You can get better acquainted mathematically with the Universal Approximation theorem from here.

4.5. Generative Adversarial Networks

Generative Adversarial Networks (GANs) are a powerful class of neural networks that are used for unsupervised learning. It was developed and introduced by Ian J. Goodfellow in 2014. GANs are basically made up of a system of two competing neural network models which compete with each other and are able to analyze, capture and copy the variations within a dataset.

Why were GANs developed in the first place?

It has been noticed most of the mainstream neural nets can be easily fooled into misclassifying things by adding only a small amount of noise into the original data. Surprisingly, the model after adding noise has higher confidence in the wrong prediction than when it predicted correctly. The reason for such adversary is that most machine learning models learn from a limited amount of data, which is a huge drawback, as it is prone to overfitting. Also, the mapping between the input and the output is almost linear. Although, it may seem that the boundaries of separation between the various classes are linear, but in reality, they are composed of linearities and even a small change in a point in the feature space might lead to misclassification of data.

January, 2022

Notes Part-1

MS. SHWETA TIWARI

How does GANs work?

Generative Adversarial Networks (GANs) can be broken down into three parts:

- **Generative:** To learn a generative model, which describes how data is generated in terms of a probabilistic model.
- **Adversarial:** The training of a model is done in an adversarial setting.
- **Networks:** Use deep neural networks as the artificial intelligence (AI) algorithms for training purpose.

In GANs, there is a **generator** and a **discriminator**. The Generator generates fake samples of data(be it an image, audio, etc.) and tries to fool the Discriminator. The Discriminator, on the other hand, tries to distinguish between the real and fake samples. The Generator and the Discriminator are both Neural Networks and they both run in competition with each other in the training phase. The steps are repeated several times and in this, the Generator and Discriminator get better and better in their respective jobs after each repetition. The working can be visualized by the diagram given below:

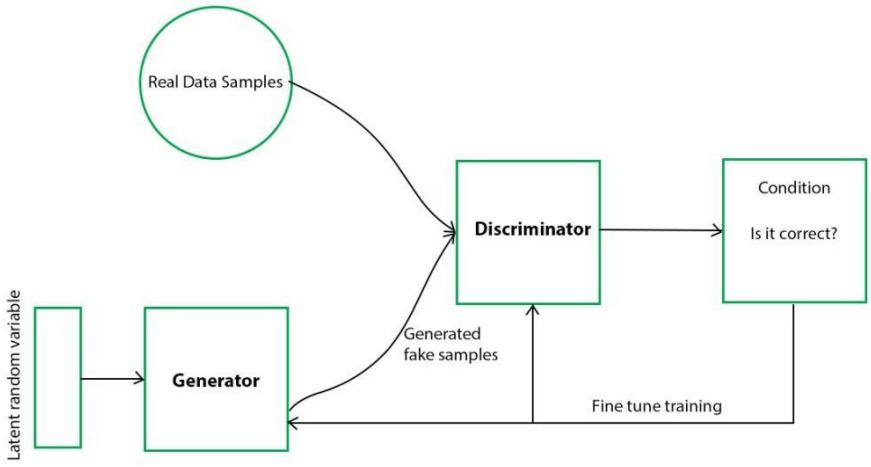


Figure 4.9: Generative Adversarial Networks

Here, the generative model captures the distribution of data and is trained in such a manner that it tries to maximize the probability of the Discriminator in making a mistake. The Discriminator, on the other hand, is based on a model that estimates the probability that the sample that it got is received from the training data and not from the Generator.

The GANs are formulated as a minimax game, where the Discriminator is trying to minimize its reward **V(D, G)** and the Generator is trying to minimize the Discriminator's reward or in other words, maximize its loss. It can be mathematically described by the formula below:

$$\min_G \max_D V(D,G)$$

$$V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z))]$$

where,

G = Generator

D = Discriminator

Pdata(x) = distribution of real data

P(z) = distribution of generator

x = sample from Pdata(x) z

= sample from P(z)

D(x) = Discriminator network

G(z) = Generator network

So, basically, training a GAN has two parts:

- **Part 1:** The Discriminator is trained while the Generator is idle. In this phase, the network is only forward propagated and no back-propagation is done. The Discriminator is trained on real data for n epochs, and see if it can correctly predict them as real. Also, in this phase, the Discriminator is also trained on the fake generated data from the Generator and see if it can correctly predict them as fake.
- **Part 2:** The Generator is trained while the Discriminator is idle. After the Discriminator is trained by the generated fake data of the Generator, we can get its predictions and use the results for training the Generator and get better from the previous state to try and fool the Discriminator.

The above method is repeated for a few epochs and then manually check the fake data if it seems genuine. If it seems acceptable, then the training is stopped, otherwise, its allowed to continue for few

more epochs.

Different types of GANs:

GANs are now a very active topic of research and there have been many different types of GAN implementation. Some of the important ones that are actively being used currently are described below:

1. **Vanilla GAN:** This is the simplest type GAN. Here, the Generator and the Discriminator are simple multi-layer perceptrons. In vanilla GAN, the algorithm is really simple, it tries to optimize the mathematical equation using stochastic gradient descent.

2. **Conditional GAN (CGAN):** CGAN can be described as a deep learning method in which some conditional parameters are put into place. In CGAN, an additional parameter 'y' is added to the Generator for generating the corresponding data. Labels are also put into the input to the Discriminator in order for the Discriminator to help distinguish the real data from the fake generated data.

3. **Deep Convolutional GAN (DCGAN):** DCGAN is one of the most popular also the most successful implementation of GAN. It is composed of ConvNets in place of multi-layer perceptrons. The ConvNets are implemented without max pooling, which is in fact replaced by convolutional stride. Also, the layers are not fully connected.

4. **Laplacian Pyramid GAN (LAPGAN):** The Laplacian pyramid is a linear invertible image representation consisting of a set of band-pass images, spaced an octave apart, plus a low-frequency residual. This approach uses multiple numbers of Generator and Discriminator networks and different levels of the Laplacian Pyramid. This approach is mainly used because it produces very high-quality images. The image is down-sampled at first at each layer of the pyramid and then it is again up-scaled at each layer in a backward pass where the image acquires some noise from the Conditional GAN at these layers until it reaches its original size.

5. **Super Resolution GAN (SRGAN):** SRGAN as the name suggests is a way of designing a GAN in which a deep neural network is used along with an adversarial network in order to produce higher resolution images. This type of GAN is particularly useful in optimally up- scaling native low-resolution images to enhance its details minimizing errors while doing so.

Questions:

1. Define ANN and Neural computing.
2. List some applications of ANNs.
3. What are the design parameters of ANN?

4. Explain the three classifications of ANNs based on their functions. Explain them in brief.
5. Write the differences between conventional computers and ANN.
6. What are the applications of Machine Learning .When it is used.
7. What is deep learning , Explain its uses and application and history.
8. What Are the Applications of a Recurrent Neural Network (RNN)?
9. What Are the Different Layers on CNN?
10. Explain Generative Adversarial Network.

January, 2022

Notes Part-1

MS. SHWETA TIWARI