**ASSIGNMENT 1**

Problem statement : Creating & visualizing neural network for the given data(Python)

Theory:

What is Neural Network ?

Neural networks, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are a subset of [machine learning](https://www.ibm.com/in-en/cloud/learn/machine-learning) and are at the heart of [deep learning](https://www.ibm.com/in-en/cloud/learn/deep-learning) algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

Artificial neural networks (ANNs) are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

Neural networks rely on training data to learn and improve their accuracy over time. However, once these learning algorithms are fine-tuned for accuracy, they are powerful tools in computer science and [artificial intelligence](https://www.ibm.com/in-en/cloud/learn/what-is-artificial-intelligence), allowing us to classify and cluster data at a high velocity. Tasks in speech recognition or image recognition can take minutes versus hours when compared to the manual identification by human experts. One of the most well-known neural networks is Google’s search algorithm.

**Types of Neural Networks**

Feed-Forward Neural Networks

Feed-forward neural networks are one of the more simple types of neural networks. It conveys information in one direction through input nodes; this information continues to be processed in this single direction until it reaches the output mode. Feed-forward neural networks may have hidden layers for functionality, and this type of most often used for facial recognition technologies.

Recurrent Neural Networks

A more complex type of neural network, recurrent neural networks take the output of a processing node and transmit the information back into the network. This results in theoretical "learning" and improvement of the network. Each node stores historical processes, and these historical processes are reused in the future during processing.

This becomes especially critical for networks in which the prediction is incorrect; the system will [attempt to learn](https://www.investopedia.com/terms/m/machine-learning.asp) why the correct outcome occurred and adjust accordingly. This type of neural network is often used in text-to-speech applications.

Convolutional Neural Networks

Convolutional neural networks, also called ConvNets or CNNs, have several layers in which data is sorted into categories. These networks have an input layer, an output layer, and a hidden multitude of convolutional layers in between. The layers create feature maps that record areas of an image that are broken down further until they generate valuable outputs. These layers can be pooled or entirely connected, and these networks are especially beneficial for image recognition applications.

Deconvolutional Neural Networks

Deconvolutional neural networks simply work in reverse of convolutional neural networks. The application of the network is to detect items that might have been recognized as important under a convolutional neural network. These items would likely have been discarded during the convolutional neural network execution process. This type of neural network is also widely used for image analysis or processing.

Modular Neural Networks

Modular neural networks contain several networks that work independently from one another. These networks do not interact with each other during an analysis process. Instead, these processes are done to allow complex, elaborate computing processes to be done more efficiently. Similar to other modular industries such as [modular real estate](https://www.investopedia.com/modular-vs-manufactured-home-insurance-5074202), the goal of the network independence is to have each module responsible for a particular part of an overall bigger picture.

Plotting (visualize) a neural network with Graphviz

Here we are utilizing source code for execution which we find in the above models:

We should talk about the methodology:

* Make a digraph object.
* Characterize the bearing of the chart utilizing rankdir.
* Make a subgraphs with the accompanying things:
* Set tone.
* Set hub properties.
* Set the Level of the subgraphs
* Make the edge between the item with (- >).

# Visualizing Artificial Neural Networks (ANNs)

* ANN Visualizer is a python library that enables us to visualize an Artificial Neural Network using just a single line of code. It is used to work with [Keras](https://keras.io/" \t "_blank) and makes use of python’s [graphviz](https://www.graphviz.org/" \t "_blank) library to create a neat and presentable graph of the neural network you’re building.
* With advanced in deep learning, you can now visualise the entire deep learning process or just the Convolutional Neural Network you’ve built.
* We are going to build simple neural network using keras and then use ANNvisualizer to visualize our neural network.

We will need 3 libraries for this demo

* keras
* ANNvisualizer
* graphviz

You can install the library using the below commands:

pip3 install keraspip3 install ann\_visualizerpip install graphviz

To generate the visualization, you need to follow the below command structure:

## ann\_viz(model, view=True, filename=”network.gv”, title=”MyNeural Network”)

* model – Your Keras sequential model
* view – If set to true, it opens the graph preview after the command has been executed
* filename – Where to save the graph. (it’s saved in a ‘.gv’ file format)
* title – The title for the visualized ANN

You have just seen how you can easily create your first neural network model in Keras.

Let’s tie it **ann\_viz()** together into this code.

Run **index.py**using:

python3 index.py

Conclusion: IKMKB