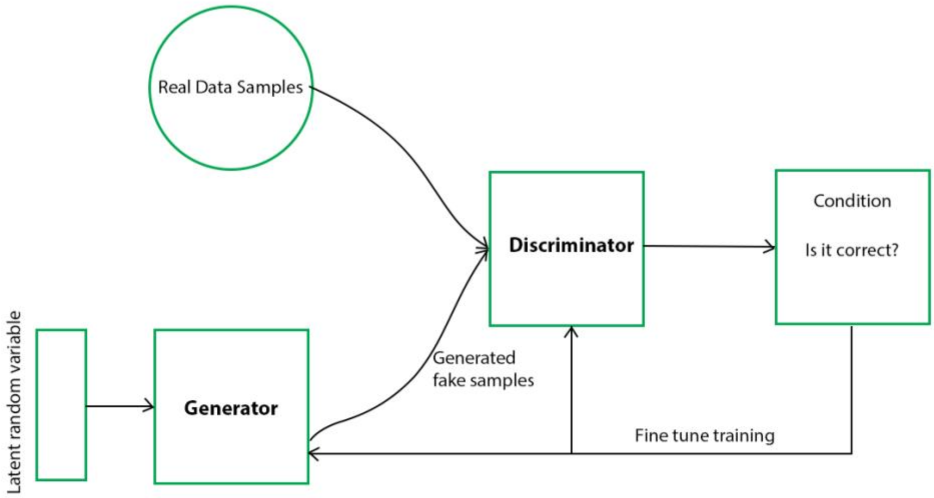
GAN

GAN stands for Generative Adversarial Network, which is a deep learning architecture composed of two neural networks, the generator and the discriminator.

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.



Components:

Generator network: The generator is responsible for generating fake data samples that mimic the characteristics of the real data.

Discriminator network: differentiate's between real and fake data.

Training:

The generator network produces synthetic data and the discriminator network evaluates it.

The generator is trained to fool the discriminator and the discriminator is trained to correctly identify real and fake data.

This process continues until the generator produces data that is indistinguishable from real data.

Types of GAN:-

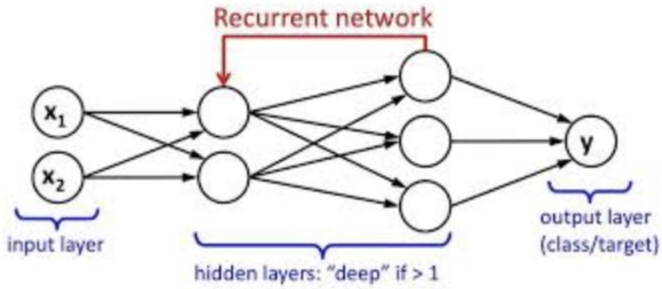
Vanilla GAN	Conditional GAN	Wasserstein GAN	Cycle-Consistent GAN
consists of a generator and a discriminator network trained through an adversarial process.	In a conditional GAN, both the generator and discriminator networks take additional input, such as a class label or a reference image, to generate or evaluate specific types of data.	This is a variant of GAN that uses the Wasserstein distance metric to measure the distance between the generated and real data distributions	In a cycle-consistent GAN, there are two GANs trained in opposite directions, with the goal of learning a mapping between two different domains of data

Applications:

- Image synthesis
- Text-to-Image synthesis
- Image-to-Image translation
- Anomaly detection

- Data augmentation

Recurrent Neural Network
Recurrent Neural Networks (RNNs) are a class of neural networks that are designed to work with sequential data.
Unlike feedforward neural networks that process input data in a single pass, RNNs have loops in their architecture that allow them to maintain an internal state or memory of past inputs.
This makes RNNs well-suited for tasks that involve sequential data, such as natural language processing, speech recognition, and time series analysis.



The key features of RNNs include:

- Recurrent connections: RNNs have recurrent connections that allow the output from one time step to be fed back into the network as input to the next time step. This creates a feedback loop that allows the network to maintain an internal state and keep track of past inputs.
- Hidden state: The internal state of an RNN is stored in a hidden vector that is updated at each time step based on the current input and the previous hidden state.
- Time-dependency: The output of an RNN at each time step depends not only on the current input but also on the previous hidden state, which creates a time-dependency that allows the network to process sequential data.
- Backpropagation through time: RNNs are trained using backpropagation through time, which is a variant of backpropagation that takes into account the time-dependency of the network.

Variations of RNN:-

Vanilla RNN	Long Short-Term Memory (LSTM)	Gated Recurrent Unit (GRU)
This is the simplest type of RNN, which uses a simple hidden state to maintain information about past inputs	LSTM is a type of RNN that uses memory cells and gating mechanisms to selectively remember or forget past inputs.	GRU is a simplified version of LSTM that uses fewer parameters and can be easier to train.

Applications: -
- Natural language processing
- Time series analysis
- Robotics

CNN
Convolutional Neural Networks (CNNs) are a type of neural network that are commonly used for image recognition and classification tasks.
CNNs are able to automatically learn hierarchical representations of image features from raw pixel values, without the need for manual feature extraction.

The key features of CNNs include:
- Convolutional layers: The core building block of a CNN is the convolutional layer, which applies a set of filters (also called kernels or weights) to the input image to extract features at different scales and orientations.

- Pooling layers: Pooling layers are used to downsample the output of convolutional layers, reducing the spatial dimensions of the feature maps while retaining the most salient features
- Activation functions: CNNs use activation functions like ReLU (Rectified Linear Unit) to introduce nonlinearity into the network, allowing it to learn more complex representations of image features.
- Fully connected layers: After the convolutional and pooling layers, CNNs often include one or more fully connected layers, which are used to generate the final output (e.g., a classification label).
- Backpropagation: Like other neural networks, CNNs are trained using backpropagation, which involves computing gradients of the loss function with respect to the network weights and adjusting the weights using an optimization algorithm such as stochastic gradient descent (SGD).

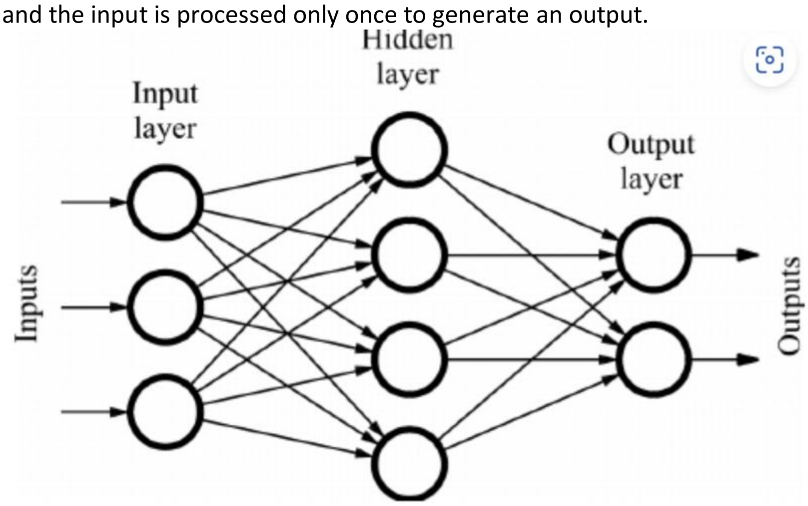
LeNet-5	AlexNet	VGGNet
LeNet-5 is one of the earliest and most popular CNN architectures, used primarily for handwritten digit recognition.	It consists of five convolutional layers, followed by two fully connected layers.	VGGNet is known for its simplicity and has been shown to achieve high accuracy on a wide range of image recognition tasks.

Applications: -

- Autonomous vehicles
- Facial recognition
- AR/VR

FeedForward Neural Network

A feedforward neural network (FFNN) is a type of artificial neural network where information flows in only one direction, from input layer to output layer. This means that there are no loops or cycles in the network, and the input is processed only once to generate an output.



The key components of a FFNN include:

- Input layer: The input layer is the first layer of the network, and it receives the input data. Each input neuron corresponds to one feature or variable in the input data.
- Hidden layers: Hidden layers are intermediate layers between the input and output layers. Each hidden layer contains a set of neurons that perform computations on the input data to extract features and generate higher-level representations.
- Output layer: The output layer is the final layer of the network, and it generates the output based on the computations performed in the hidden layers. The number of output neurons depends on the type of task, such as binary classification, multiclass classification, or regression.
- Activation functions: Each neuron in the network applies an activation function to the weighted sum of its inputs. Popular activation functions include sigmoid, ReLU, and tanh.
- Weights and biases: Each neuron has a set of weights and biases that determine its contribution to the output. These weights and biases are learned during the training process using optimization algorithms such as backpropagation.

Applications:

- Image and video processing
- Natural language processing
- Medical diagnosis and analysis