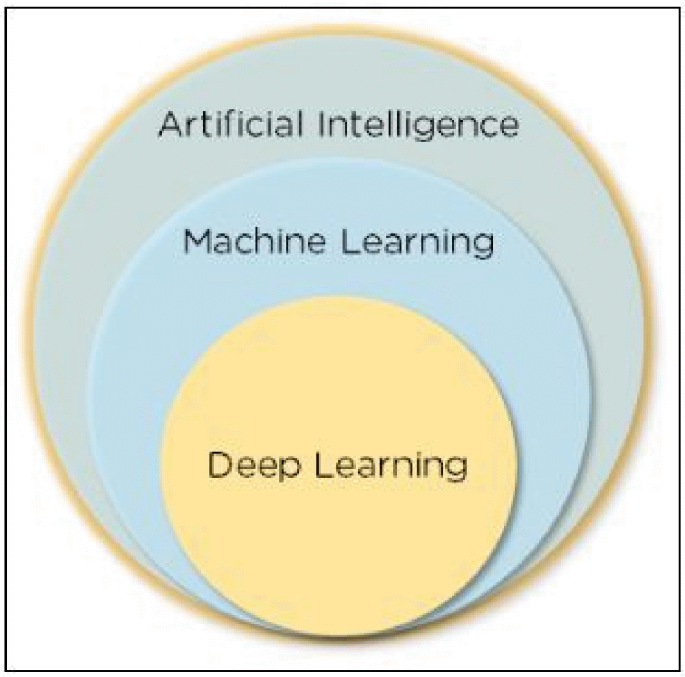
Data Science and Generative AI

by SATISH @

https://sathyatech.com

Deep Learning (DL)

Deep learning is a subset of machine learning, which is a part of artificial intelligence (AI).



Artificial intelligence is the ability of a machine to imitate intelligent human behavior. Machine learning allows a system to learn and improve from experience automatically. Deep learning is an application of machine learning that uses complex algorithms and deep neural nets to train a model.

Applications of Deep Learning

Next up in this introduction to deep learning tutorial, let's learn about some of the top applications of deep learning. Deep learning is widely used to make weather predictions about rain, earthquakes, and tsunamis. It helps in taking the necessary precautions. With deep learning, machines can comprehend speech and provide the required output. It enables the machines to recognize people and objects in the images fed to it.

Deep learning models also help advertisers leverage data to perform real-time bidding and targeted display advertising. In the next section introduction to deep learning tutorial, we will cover the need and importance of deep learning.

Importance of Deep Learning

- **Machine learning** works only with sets of **structured** and **semi-structured data**, while **deep learning** works with both **structured and unstructured data**
- Deep learning algorithms can perform **complex operations** efficiently, while machine learning algorithms cannot
- Machine learning algorithms use labeled sample data to extract patterns, while deep learning accepts **large volumes** of data as input and analyzes the input data to extract features out of an object
- The performance of machine learning algorithms decreases as the number of data increases; so to maintain the performance of the model, we need a deep learning

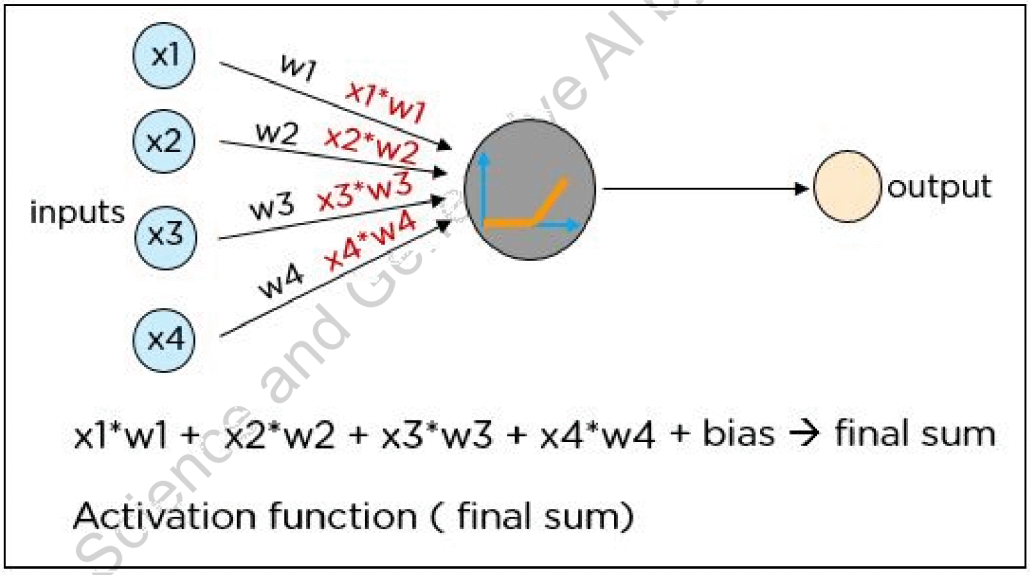
What are Neural Networks?

A neural network is a system modeled on the human brain, consisting of an input layer, multiple hidden layers, and an output layer.

Data is fed as input to the neurons. The information is transferred to the next layer using appropriate weights and biases. The output is the final value predicted by the artificial neuron.

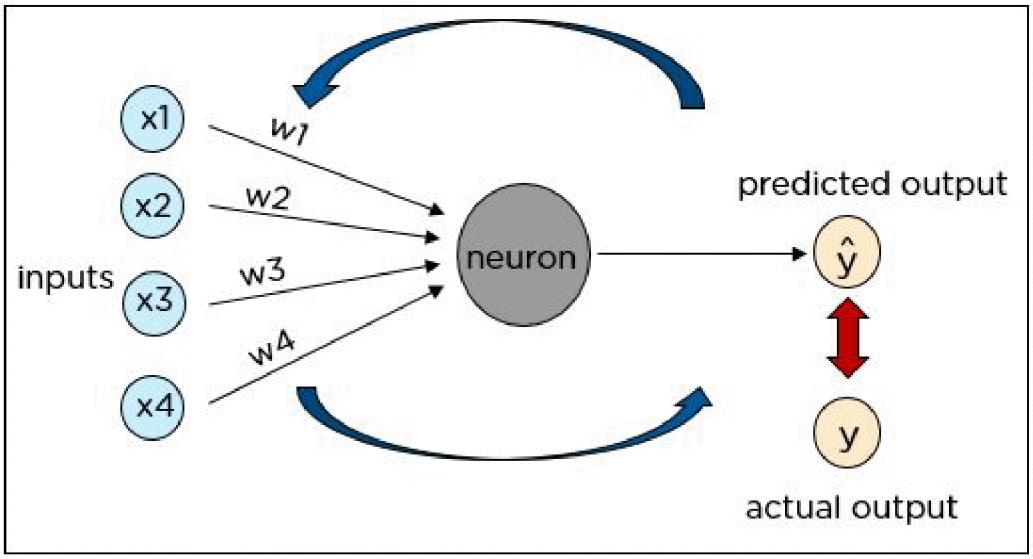
Each neuron in a neural network performs the following operations:

- The product of each input and the weight of the channel it is passed over is found
- The sum of the weighted products is computed, which is called the **weighted sum**
- A bias value of the neuron is added to the weighted sum
- The final sum is then subjected to a particular function known as the **activation function**



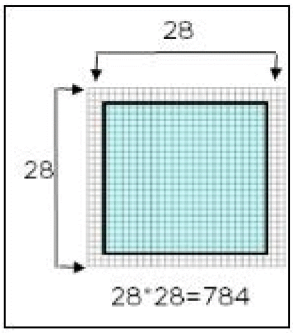
Cost Function

The cost function is one of the significant components of a neural network. The cost value is the difference between the neural nets predicted output and the actual output from a set of labeled training data. **The least-cost value is obtained by making adjustments to the weights and biases iteratively throughout the training process.**

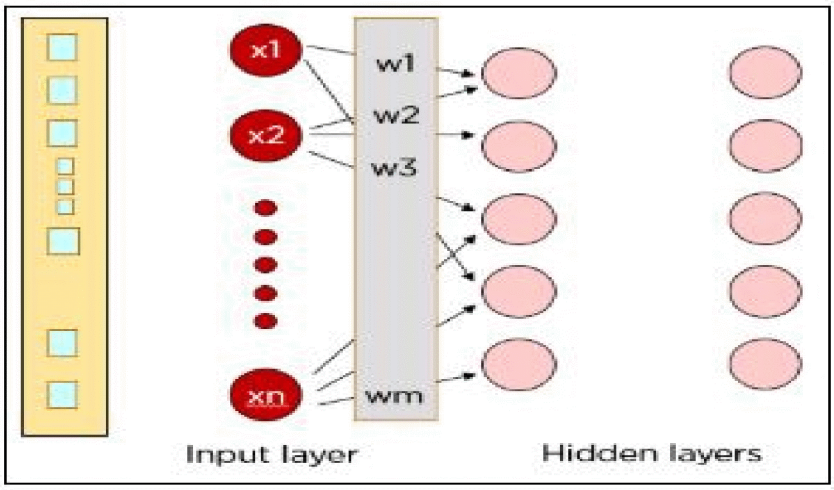


How Do Neural Networks Work?

In the next section of this introduction to deep learning the neural network will be trained to identify shapes. The shapes are images of 28*28 pixels.



Each pixel is fed as input to the neurons in the first layer. Hidden layers improve the accuracy of the output. Data is passed on from layer to layer overweight channels. Each neuron in one layer is weighted to each of the neurons in the next layer.

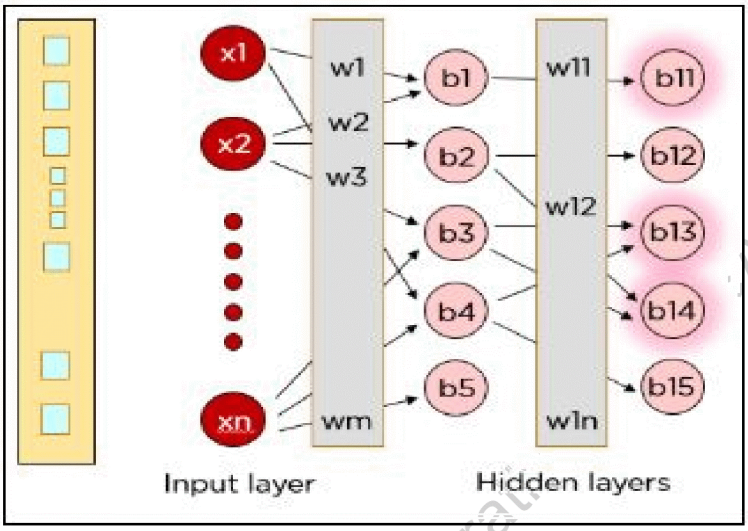


Each neuron in the first hidden layer takes a subset of the inputs and processes it. All the inputs are multiplied by their respective weights and a bias is added. The output of the weighted sum is applied to an activation function. The results of the activation function determine which neurons will be activated in the following layer.

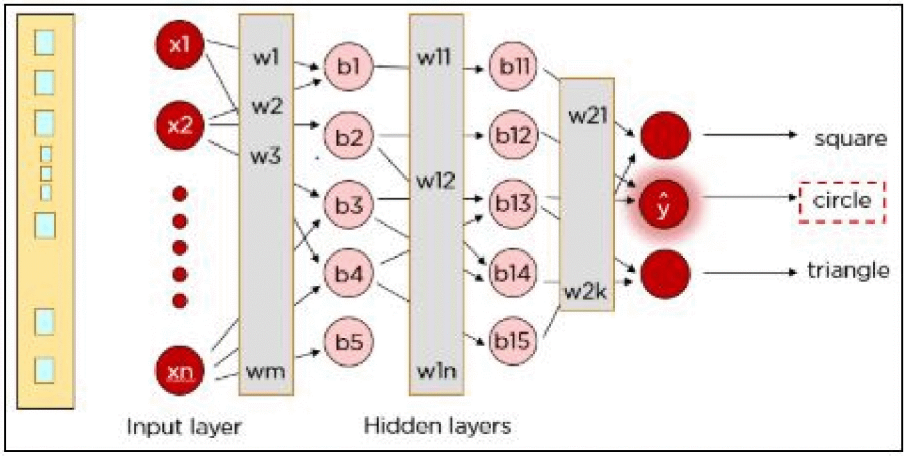
Step 1: x1*w1 + x2*w2 + b1

Step 2: Φ(x1* w1 + x2*w2 + b1)

where Φ is an activation function



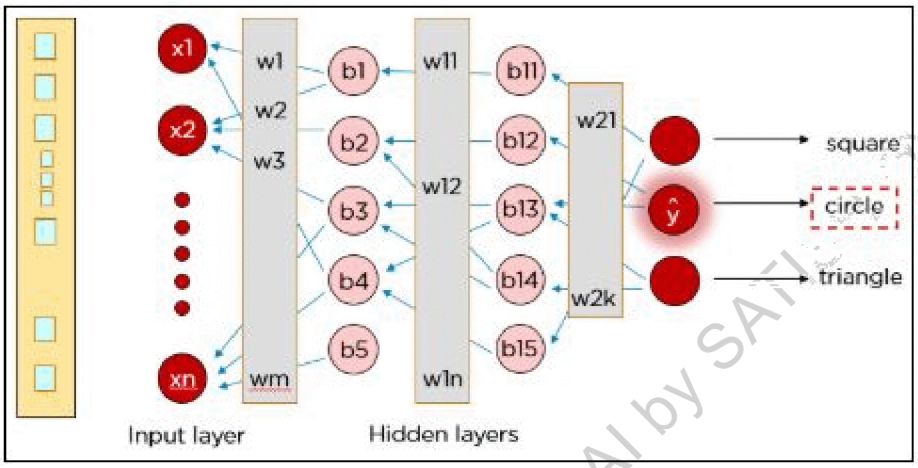
The above steps are performed again to ensure the information reaches the output layer, after which a single neuron in the output layer gets activated based on the activation function's value.



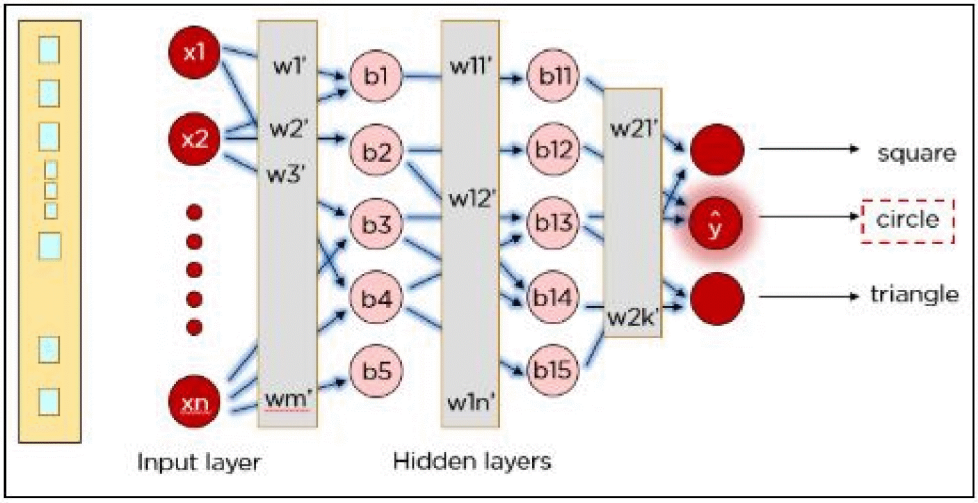
As you can see, our actual input was a square, but the neural network predicted the output as a circle. So, what went wrong?

The neural network has to be trained until the predicted output is correct and the predicted output is compared to the actual output by calculating the cost function.

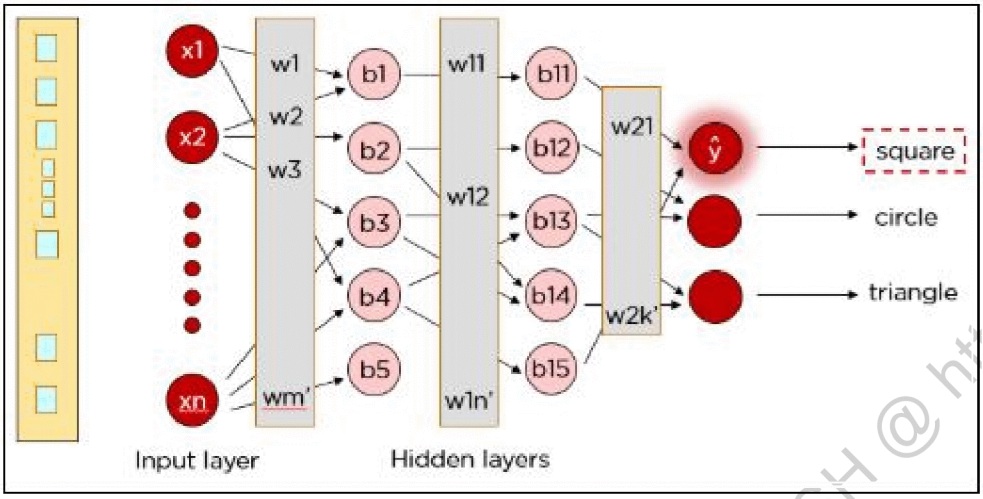
The cost function is calculated using the formula where Y is the actual value and Y hat is the predicted value. **The cost function determines the error in the prediction and reports it back to the neural network. This is called backpropagation.**



The weights are adjusted to reduce the error. The network is trained with the new weights.



Once again, the cost is determined and the backpropagation procedure is continued until the cost cannot be reduced any further.



Similarly, our network can be trained to predict circles and triangles too.

Now that you have a good understanding of how neural networks work, let's look at some of the important deep learning platforms.

Deep Learning Platforms

In the following section of the introduction to deep learning, you will learn about several deep learning platforms and when they are used.

Torch

The torch was developed using the LUA language with an implementation in C. **Torch's Python implementation is called PyTorch.**

Keras

Keras is a Python framework for deep learning. Its USP is reusability of code for CPU and GPU.

TensorFlow

TensorFlow is an open-source deep-learning library developed by Google. It's developed in C++ and has its implementation in Python. **Keras can now be run on top of TensorFlow**.

DL4J

Deep Learning for Java (DL4J) is the first deep learning library written for Java and Scala. It's integrated with Hadoop and Apache Spark.

Google's TensorFlow is currently the most popular learning library in the world. It's based on the concept of tensors, which are vectors or matrices of n dimensions.

Installation :

 # pip install keras

 # pip install tensorflow

 # pip install tensorflow==2.0.0

 # pip install tensorflow==2.1.0

 # pip install torch

 # pip install theano

TensorFlow

Google's Brain team developed a Deep Learning Framework called TensorFlow, which supports languages like Python and R, and uses dataflow graphs to process data. This is very important because as you build these neural networks, you can look at how the data flows through the neural network.

TensorFlow's machine learning models are easy to build, can be used for robust machine learning production, and allow powerful experimentation for research.

With TensorFlow, you also get TensorBoard for data visualization, which is a large package that generally goes unnoticed. TensorBoard simplifies the process for visually displaying data when working with your shareholders. You can use the R and Python visualization packages as well.

Initial release:	November 9, 2015
Stable release:	2.4.1 / January 21, 2021
Written in:	Python, C++, CUDA
Platform:	Linux, macOS, Windows, Android, JavaScript
Type:	Machine learning library
Repository	github.com/tensorflow/tensorflow
License:	Apache License 2.0
Website	www.tensorflow.org

Keras

Francois Chollet originally developed Keras, with 350,000+ users and 700+ open-source contributors, making it one of the fastest-growing deep learning framework packages.

Keras supports high-level neural network API, written in Python. What makes Keras interesting is that it runs on top of TensorFlow, Theano, and CNTK.

Keras is used in several startups, research labs, and companies including Microsoft Research, NASA, Netflix, and Cern.

Other Features of Keras:

- User-friendly, as it offers simple APIs and provides clear and actionable feedback upon user error
- Provides modularity as a sequence or a graph of standalone, fully-configurable modules that can be combined with as few restrictions as possible
- Easily extensible as new modules are simple to add, making Keras suitable for advanced research

Initial release:	March 27, 2015
Stable release:	2.4.0 / June 17, 2020
Platform:	Cross-platform
Type:	Neural networks
Repository	github.com/keras-team/keras
License:	Massachusetts Institute of Technology (MIT)

Website	https://keras.io/

PyTorch

Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan authored PyTorch and is primarily developed by Facebook's AI Research lab (FAIR). It's built on the Lua-based scientific computing framework for machine learning and deep learning algorithms. PyTorch employed Python, CUDA, along with C/C++ libraries, for processing and was designed to scale the production of building models and overall flexibility. If you're well-versed with C/C++, then PyTorch might not be too big of a jump for you.

PyTorch is widely used in large companies like Facebook, Twitter, and Google.

Other Features of the Deep Learning Framework Include:

- It provides flexibility and speed due to its hybrid front-end.
- Enables scalable distributed training and performance optimization in research and production using the "torch distributed" backend.
- Deep integration with Python allows popular libraries and packages to be quickly write neural network layers in Python.

Initial release:	September 2016
Stable release:	1.7.1 / December 10, 2020
Platform:	IA-32, x86-64
Type:	Library for machine learning and deep learning
Repository	github.com/pytorch/pytorch

License:	Berkeley Software Distribution (BSD)
Website	https://pytorch.org/

Theano

The University de Montreal developed Theano, written in Python and centers around NVIDIA CUDA, allowing users to integrate it with GPS. The Python library allows users to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays.

Initial release:	2007
Stable release:	1.0.5 / July 27, 2020
Platform:	Linux, macOS, Windows
Type:	Machine learning library
Repository	github.com/pytorch/pytorch
License:	The 3-Clause Berkeley Software Distribution (BSD)
Website	http://www.deeplearning.net/software/theano/

Deep learning has gained massive popularity in scientific analysis, and its algorithms are widely used by industry and ever increasingly in ecommerce now, as it solves complex problems. All deep learning algorithms use different types of neural networks to perform specific tasks.

What is Deep Learning and why do I need to know?

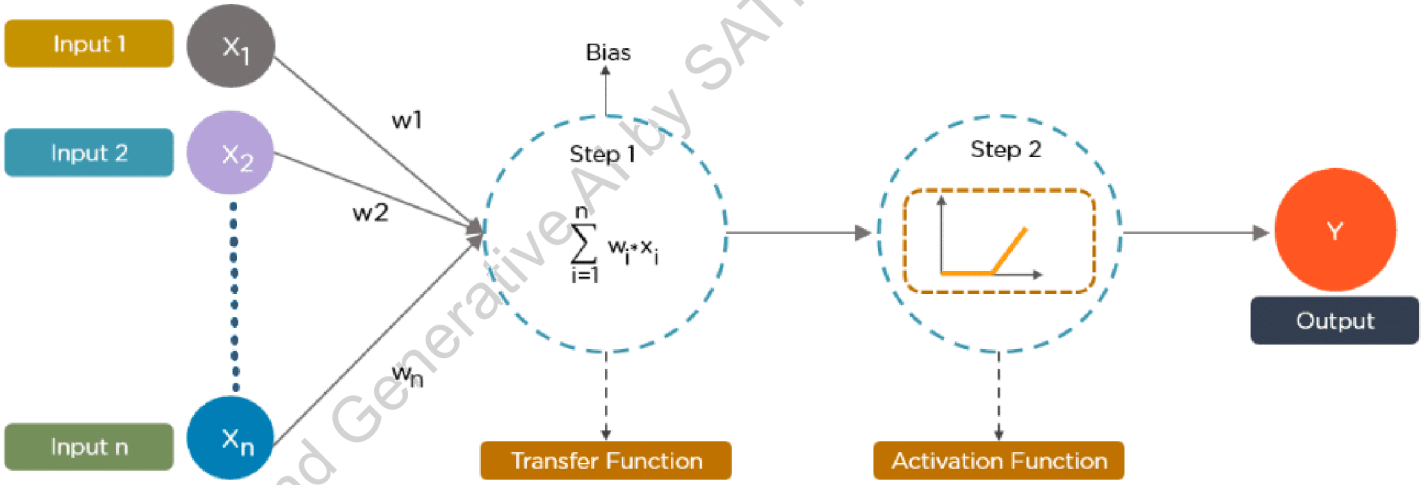
Deep learning uses artificial neural networks to perform sophisticated computations on large amounts of data. It is a type of machine learning that works based on the structure and function of the human brain.

Deep learning algorithms train machines by learning from examples. Industries such as individualised product selection software for ecommerce, media entertainment, and advertising commonly use deep learning.

Defining Neural Networks

A neural network is structured like the human brain and consists of artificial neurons, also known as nodes. These nodes are stacked next to each other in three layers:

- The input layer
- The hidden layer(s)
- The output layer



Data provides each node with information in the form of inputs. The node multiplies the inputs with random weights, calculates them, and adds a bias. Finally, nonlinear functions, also known as activation functions, are applied to determine which neuron to fire.

How Deep Learning Algorithms Work?

While deep learning algorithms feature self-learning representations, they depend upon ANNs that mirror the way the brain computes information. During the training process, algorithms use unknown elements in the input distribution to extract features, group objects, and discover useful data patterns. Much like training machines for self-learning, this occurs at multiple levels, using the algorithms to build the models.

Deep learning models make use of several algorithms. While no one network is considered perfect, some algorithms are better suited to perform specific tasks. To choose the right ones, it's good to gain a solid understanding of all primary algorithms.

Types of Algorithms Used in Deep Learning

Here is the list of the top 10 most popular deep learning algorithms:

1. Convolutional Neural Networks (CNNs)
2. Long Short Term Memory Networks (LSTMs)
3. Recurrent Neural Networks (RNNs)
4. Generative Adversarial Networks (GANs)
5. Radial Basis Function Networks (RBFNs)
6. Multilayer Perceptrons (MLPs)
7. Self Organizing Maps (SOMs)
8. Deep Belief Networks (DBNs)
9. Restricted Boltzmann Machines(RBMs)
10. Autoencoders

Deep learning algorithms work with almost any kind of data and require large amounts of computing power and information to solve complicated issues. Now, let us, deep-dive, into the top 10 deep learning algorithms.

1. Convolutional Neural Networks (CNNs)

CNNs, also known as ConvNets, consist of multiple layers and are mainly used for image processing and object detection.

NN's are widely used to identify satellite images, process medical images, forecast time series, and detect anomalies.

Yann LeCun developed the first CNN in 1988 when it was called LeNet. It was used for recognising characters like ZIP codes and digits

How Do CNNs Work?

CNN's have multiple layers that process and extract features from data:

Convolution Layer

- CNN has a convolution layer that has several filters to perform the convolution operation.

Rectified Linear Unit (ReLU)

- CNNs have a ReLU layer to perform operations on elements. The output is a rectified feature map.

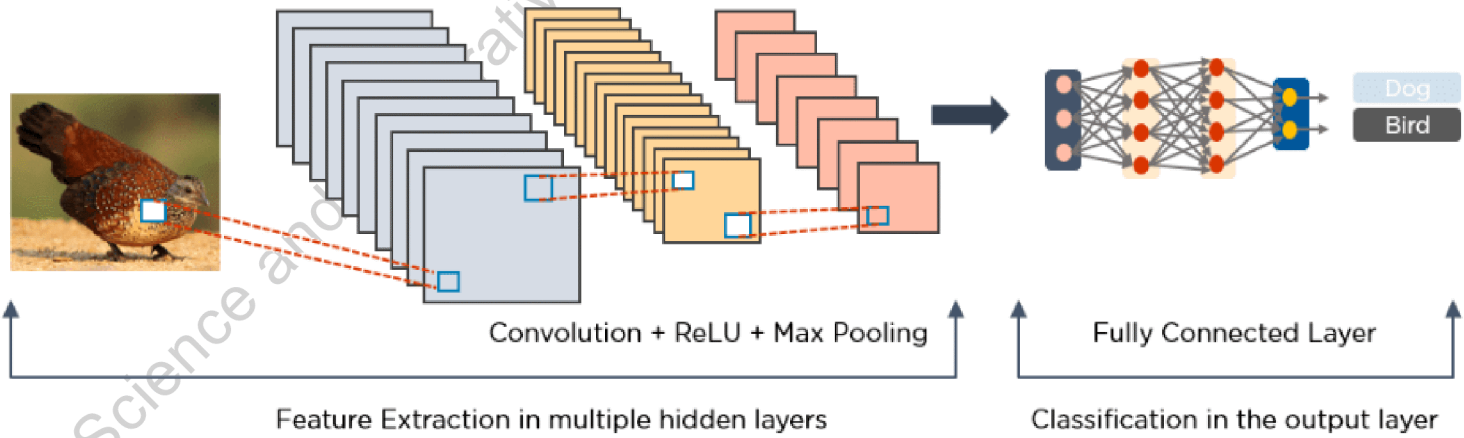
Pooling Layer

- The rectified feature map next feeds into a pooling layer. Pooling is a down-sampling operation that reduces the dimensions of the feature map.
- The pooling layer then converts the resulting two-dimensional arrays from the pooled feature map into a single, long, continuous, linear vector by flattening it.

Fully Connected Layer

- A fully connected layer forms when the flattened matrix from the pooling layer is fed as an input, which classifies and identifies the images.

Below is an example of an image processed via CNN.



2. Long Short Term Memory Networks (LSTMs)

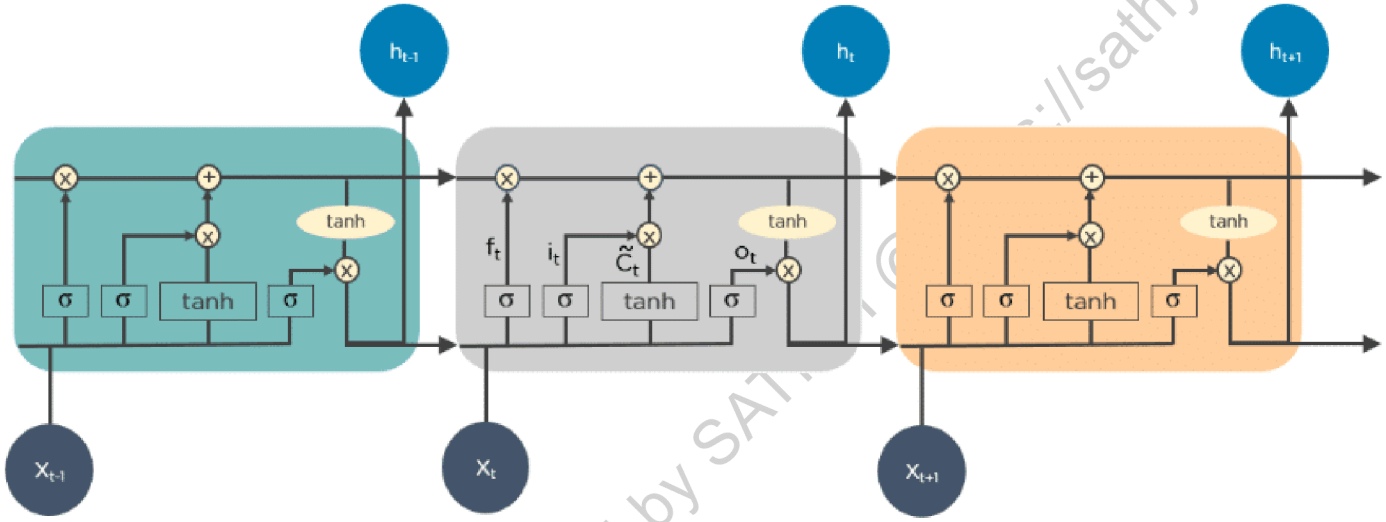
LSTMs are a type of Recurrent Neural Network (RNN) that can learn and memorize long-term dependencies. Recalling past information for long periods is the default behaviour.

LSTMs retain information over time. They are useful in time-series prediction because they remember previous inputs. LSTMs have a chain-like structure where four interacting layers communicate uniquely. Besides time-series predictions, LSTMs are typically used for speech recognition.

How Do LSTMs Work?

- First, they forget irrelevant parts of the previous state
- Next, they selectively update the cell-state values
- Finally, the output of certain parts of the cell state

Below is a diagram of how LSTMs operate:

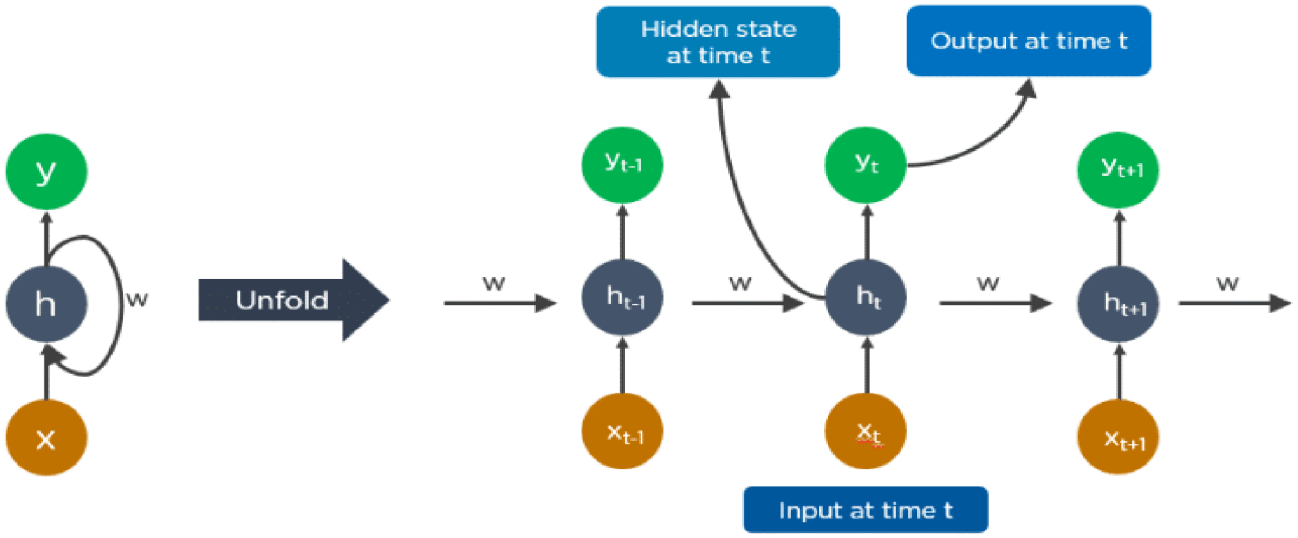


3. Recurrent Neural Networks (RNNs)

RNNs have connections that form directed cycles, which allow the outputs from the LSTM to be fed as inputs to the current phase.

The output from the LSTM becomes an input to the current phase and can memorize previous inputs due to its internal memory. RNNs are commonly used for image captioning, time-series analysis, natural language processing, handwriting recognition, and machine translation.

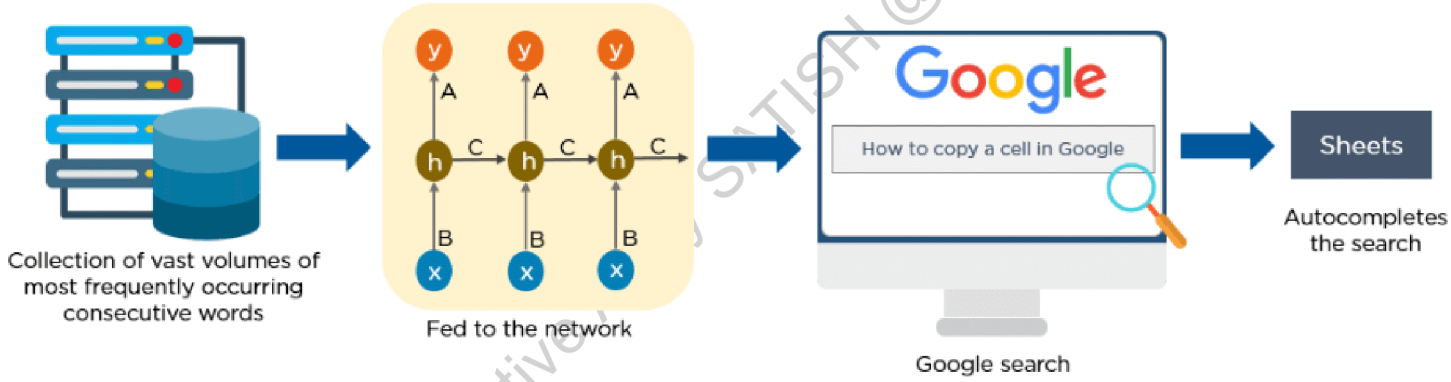
An unfolded RNN looks like this:



How Do RNNs work?

- The output at time t-1 feeds into the input at time t.
- Similarly, the output at time t feeds into the input at time t+1.
- RNNs can process inputs of any length.
- The computation accounts for historical information, and the model size does not increase with the input size.

Here is an example of how Google's autocompleting feature works:



4. Generative Adversarial Networks (GANs)

GANs are generative deep learning algorithms that create new data instances that resemble the training data. GAN has two components: a **generator**, which learns to generate fake data, and a **discriminator**, which learns from that false information.

The usage of GANs has increased over some time. They can be used to improve astronomical images and simulate gravitational lensing for dark-matter research. Video

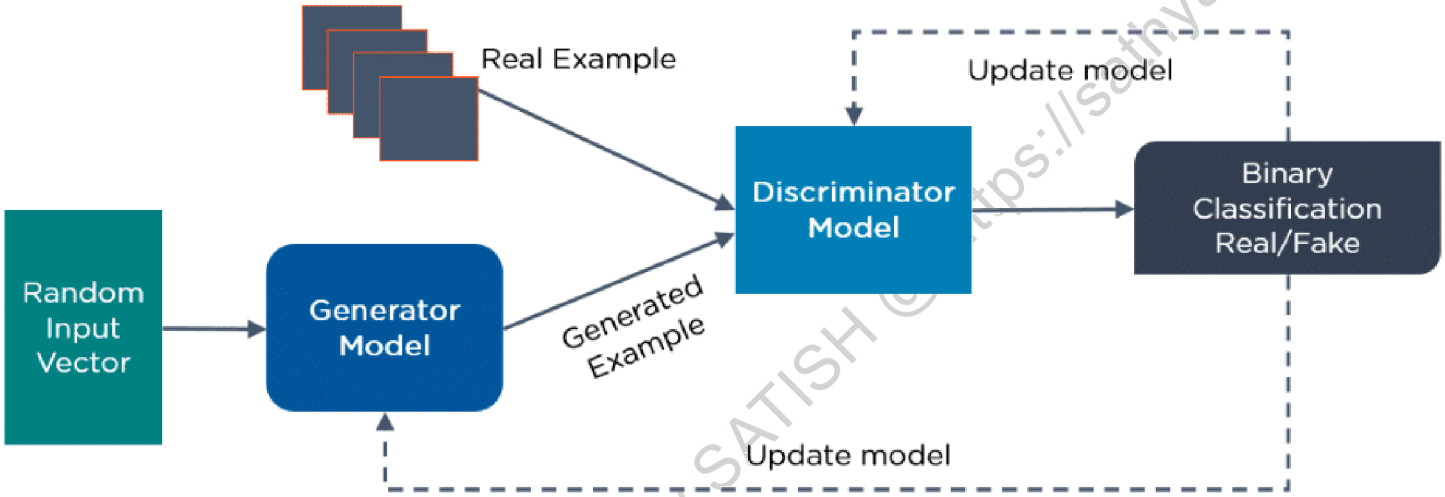
game developers use GANs to upscale low-resolution, 2D textures in old video games by recreating them in 4K or higher resolutions via image training.

GANs help generate realistic images and cartoon characters, create photographs of human faces, and render 3D objects.

How Do GANs Work?

- The discriminator learns to distinguish between the generator's fake data and the real sample data.
- During the initial training, the generator produces fake data, and the discriminator quickly learns to tell that it's false.
- The GAN sends the results to the generator and the discriminator to update the model.

Below is a diagram of how GANs operate:



5. Radial Basis Function Networks (RBFNs)

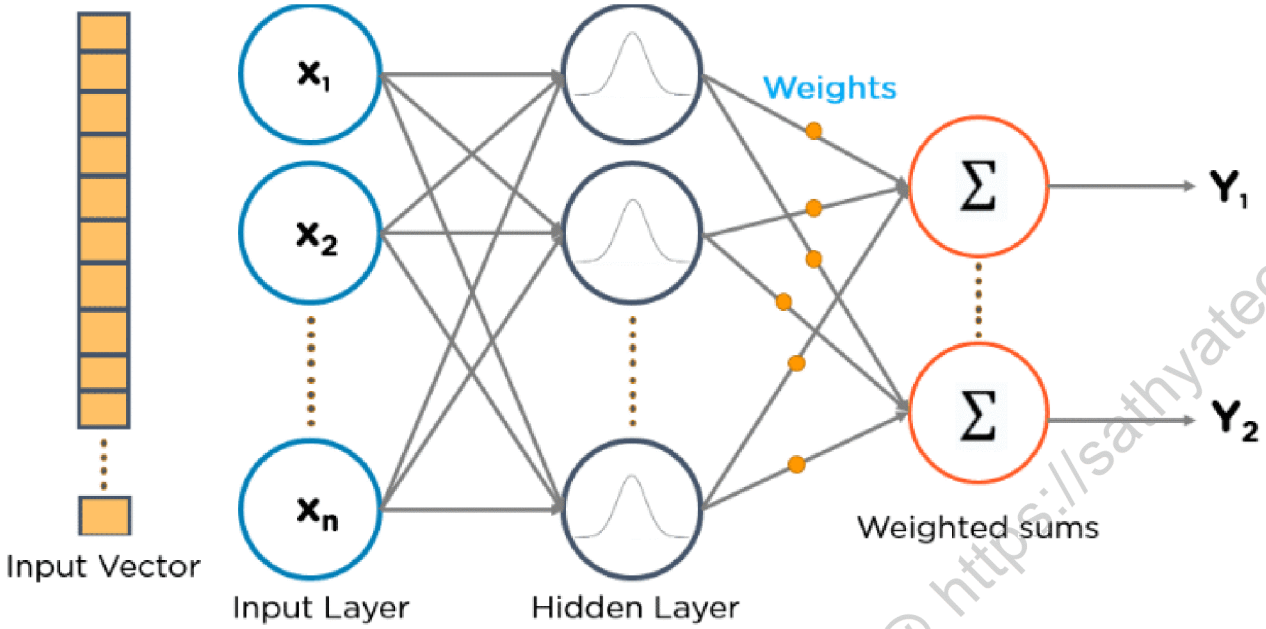
RBFNs are special types of feedforward neural networks that use radial basis functions as activation functions. They have an input layer, a hidden layer, and an output layer and are mostly used for classification, regression, and time-series prediction.

How Do RBFNs Work?

- RBFNs perform classification by measuring the input's similarity to examples from the training set.
- RBFNs have an input vector that feeds to the input layer. They have a layer of RBF neurons.

- The function finds the weighted sum of the inputs, and the output layer has one node per category or class of data.
- The neurons in the hidden layer contain the Gaussian transfer functions, which have outputs that are inversely proportional to the distance from the neuron's centre.
- The network's output is a linear combination of the input's radial-basis functions and the neuron's parameters.

See this example of an RBFN:



6. Multilayer Perceptrons (MLPs)

MLPs are an excellent place to start learning about deep learning technology.

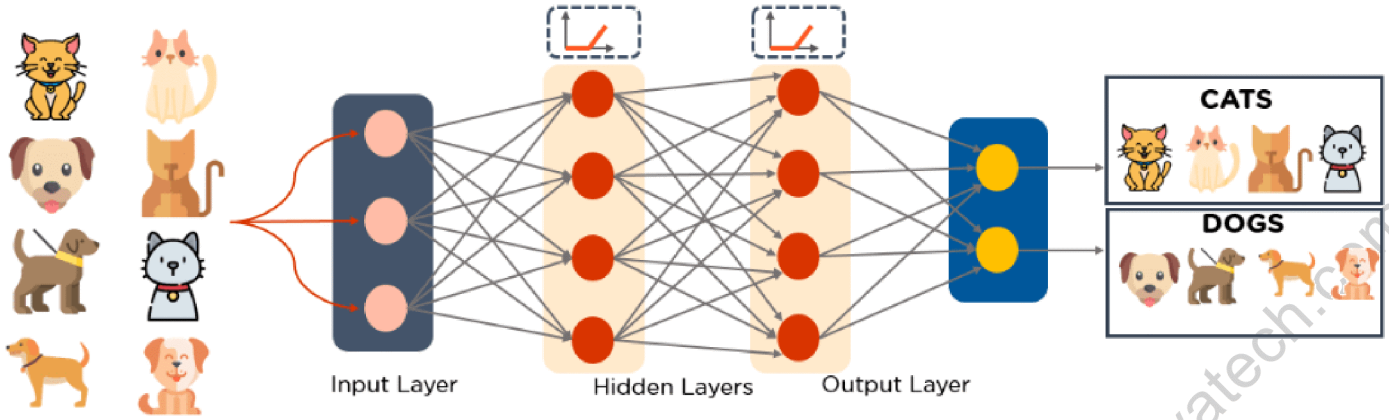
MLPs belong to the class of feedforward neural networks with multiple layers of perceptrons that have activation functions. MLPs consist of an input layer and an output layer that is fully connected. They have the same number of input and output layers but may have multiple hidden layers and can be used to build speech recognition, image recognition, and machine translation software.

How Do MLPs Work?

- MLPs feed the data to the input layer of the network. The layers of neurons connect in a graph so that the signal passes in one direction.
- MLPs compute the input with the weights that exist between the input layer and the hidden layers.

- MLPs use activation functions to determine which nodes to fire. Activation functions include ReLUs, sigmoid functions, and tanh.
- MLPs train the model to understand the correlation and learn the dependencies between the independent and the target variables from a training data set.

Below is an example of an MLP. The diagram computes weights and bias and applies suitable activation functions to classify images of cats and dogs.



7. Self Organizing Maps (SOMs)

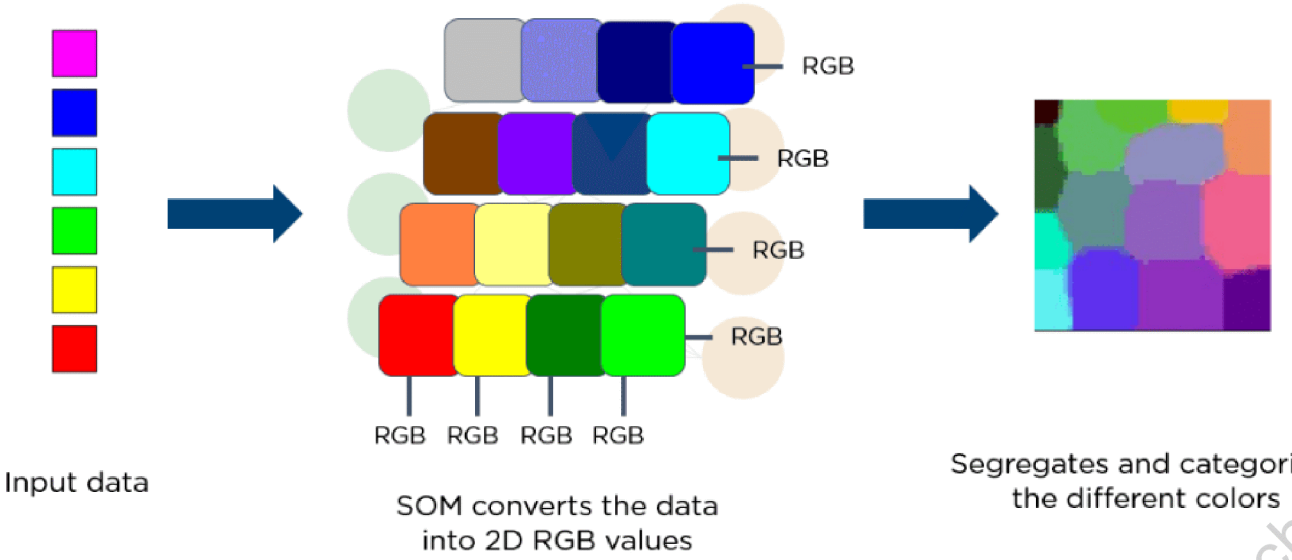
Professor Teuvo Kohonen invented SOMs**, which enable data visualisation to reduce the dimensions of data through self-organizing artificial neural networks.**

Data visualization attempts to solve the problem that humans cannot easily visualise high-dimensional data. SOMs are created to help users understand this high-dimensional information.

How Do SOMs Work?

- SOMs initialize weights for each node and choose a vector at random from the training data.
- SOMs examine every node to find which weights are the most likely input vector. The winning node is called the Best Matching Unit (BMU).
- SOMs discover the BMU's neighbourhood, and the amount of neighbours lessens over time.
- SOMs award a winning weight to the sample vector. The closer a node is to a BMU, the more its weight changes.
- The further the neighbour is from the BMU, the less it learns. SOMs repeat step two for N iterations.

Below, see a diagram of an input vector of different colours. This data feeds to a SOM, which then converts the data into 2D RGB values. Finally, it separates and categorises the different colours.



8. Deep Belief Networks (DBNs)

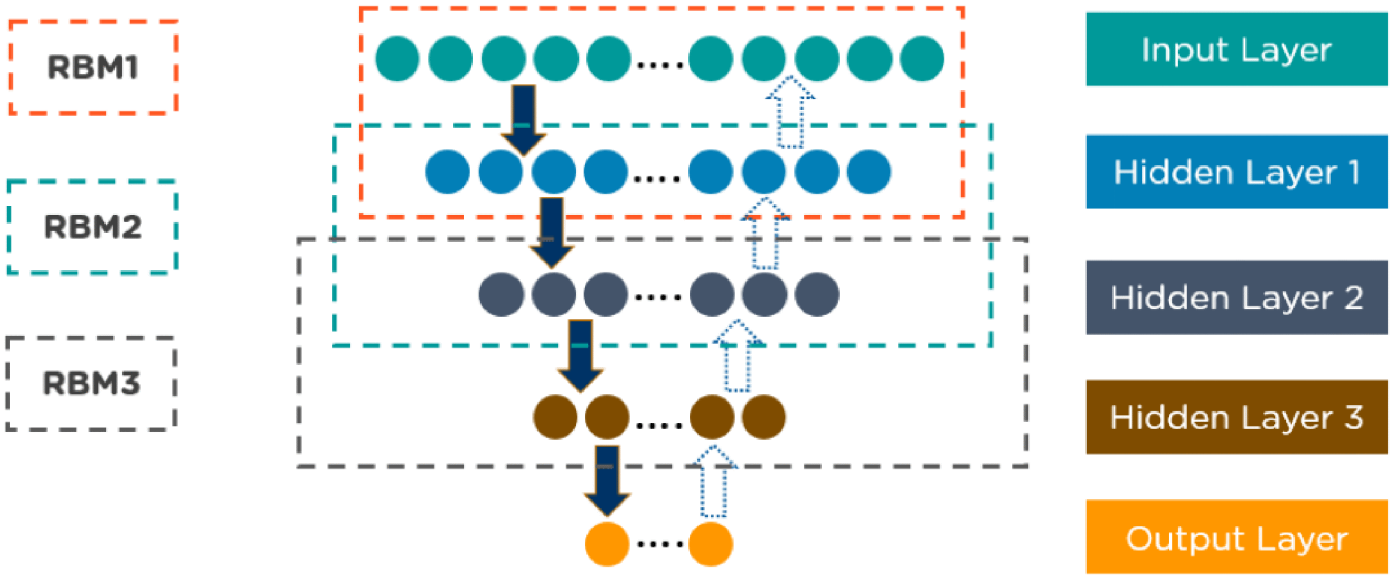
DBNs are generative models that consist of multiple layers of stochastic, latent variables. The latent variables have binary values and are often called hidden units.

DBNs are a stack of Boltzmann Machines with connections between the layers, and each RBM layer communicates with both the previous and subsequent layers. Deep Belief Networks (DBNs) are used for image recognition, video recognition, and motion-capture data.

How Do DBNs Work?

- Greedy learning algorithms train DBNs. The greedy learning algorithm uses a layer-by-layer approach for learning the top-down, generative weights.
- DBNs run the steps of Gibbs sampling on the top two hidden layers. This stage draws a sample from the RBM defined by the top two hidden layers.
- DBNs draw a sample from the visible units using a single pass of ancestral sampling through the rest of the model.
- DBNs learn that the values of the latent variables in every layer can be inferred by a single, bottom-up pass.

Below is an example of DBN architecture:



9. Restricted Boltzmann Machines (RBMs)

Developed by Geoffrey Hinton, RBMs are stochastic neural networks that can learn from a probability distribution over a set of inputs.

This deep learning algorithm is used for dimensionality reduction, classification, regression, collaborative filtering, feature learning, and topic modelling. RBMs constitute the building blocks of DBNs.

RBMs consist of two layers:

- Visible units
- Hidden units

Each visible unit is connected to all hidden units. RBMs have a bias unit that is connected to all the visible units and the hidden units, and they have no output nodes.

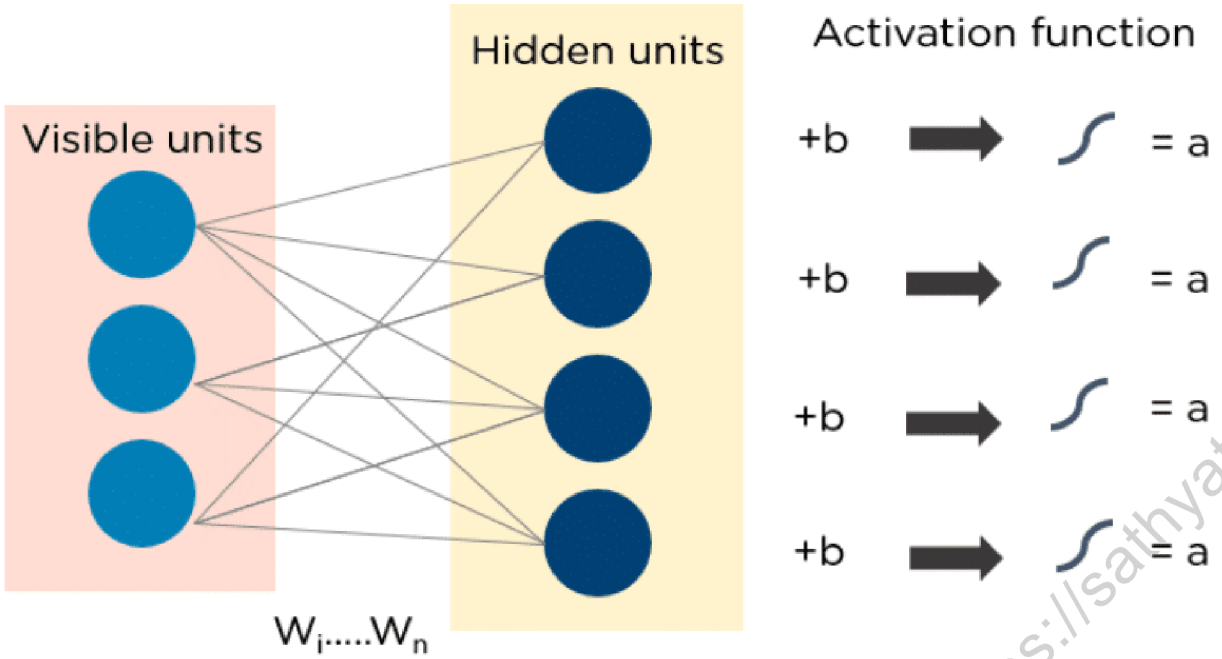
How Do RBMs Work?

RBMs have two phases: forward pass and backward pass.

- RBMs accept the inputs and translate them into a set of numbers that encode the inputs in the forward pass.
- RBMs combine every input with individual weight and one overall bias. The algorithm passes the output to the hidden layer.
- In the backward pass, RBMs take that set of numbers and translate them to form the reconstructed inputs.

- RBMs combine each activation with individual weight and overall bias and pass the output to the visible layer for reconstruction.
- At the visible layer, the RBM compares the reconstruction with the original input to analyse the quality of the result.

Below is a diagram of how RBMs function:



10. Autoencoders

Autoencoders are a specific type of feedforward neural network in which the input and output are identical. Geoffrey Hinton designed autoencoders in the 1980s to solve unsupervised learning problems. They are trained neural networks that replicate the data from the input layer to the output layer. **Autoencoders are used for purposes such as pharmaceutical discovery, popularity prediction, and image processing.**

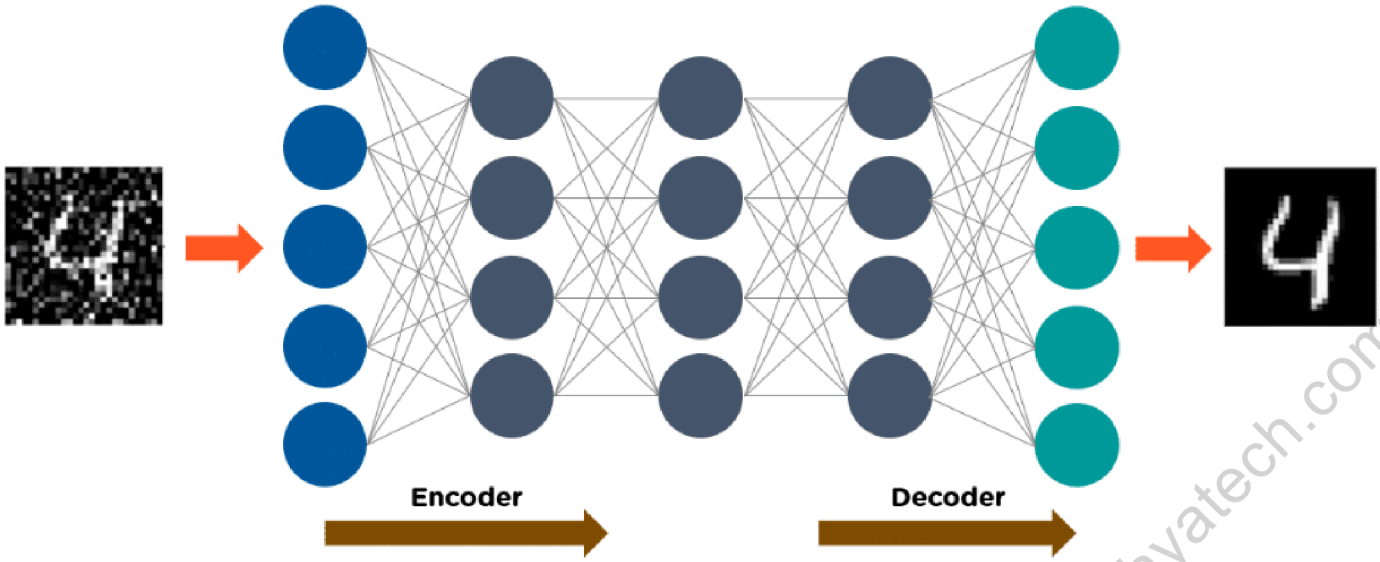
How Do Autoencoders Work?

An autoencoder consists of three main components: the encoder, the code, and the decoder.

- Autoencoders are structured to receive an input and transform it into a different representation. They then attempt to reconstruct the original input as accurately as possible.
- When an image of a digit is not visible, it feeds to an autoencoder neural network.

- Autoencoders first encode the image, then reduce the size of the input into a smaller representation.
- Finally, the autoencoder decodes the image to generate the reconstructed image.

The following image demonstrates how autoencoders operate:



Important Components of a Deep Neural Network:

1. Forward Propagation: In this process, input is passed forward from one layer of the network to the next until it passes through all layers and reaches the output.

2. Backpropagation: This is an iterative process that uses a chain rule to determine the contribution of each neuron to errors in the output. The error values are then propagated back through the network, and the weights of each neuron are adjusted accordingly.

3. Optimization: This technique is used to reduce errors generated during backpropagation in a deep neural network. Various algorithms, such as gradient descent and stochastic gradient descent, can be used to optimize the network.

4. Activation Functions: Activation functions are used to convert inputs into an output that can be recognized by the neural network. There are several types of activation functions, including linear, sigmoid, tanh, and ReLu (Rectified Linear Units).

5. Loss Functions: These functions are used to measure how well a neural network has performed after backpropagation and optimization. Common loss functions include mean squared error (MSE) and accuracy.

By combining all of these components, deep learning can take complex inputs and produce accurate predictions for a variety of tasks.

Deep Learning Algorithms

The three most popular deep learning algorithms are

- convolutional neural networks (CNNs),
- recurrent neural networks (RNNs)
- long short-term memory networks (LSTMs).

CNNs are used for image recognition, object detection, and classification.

RNNs are used for sequence modeling, such as language translation and text generation.

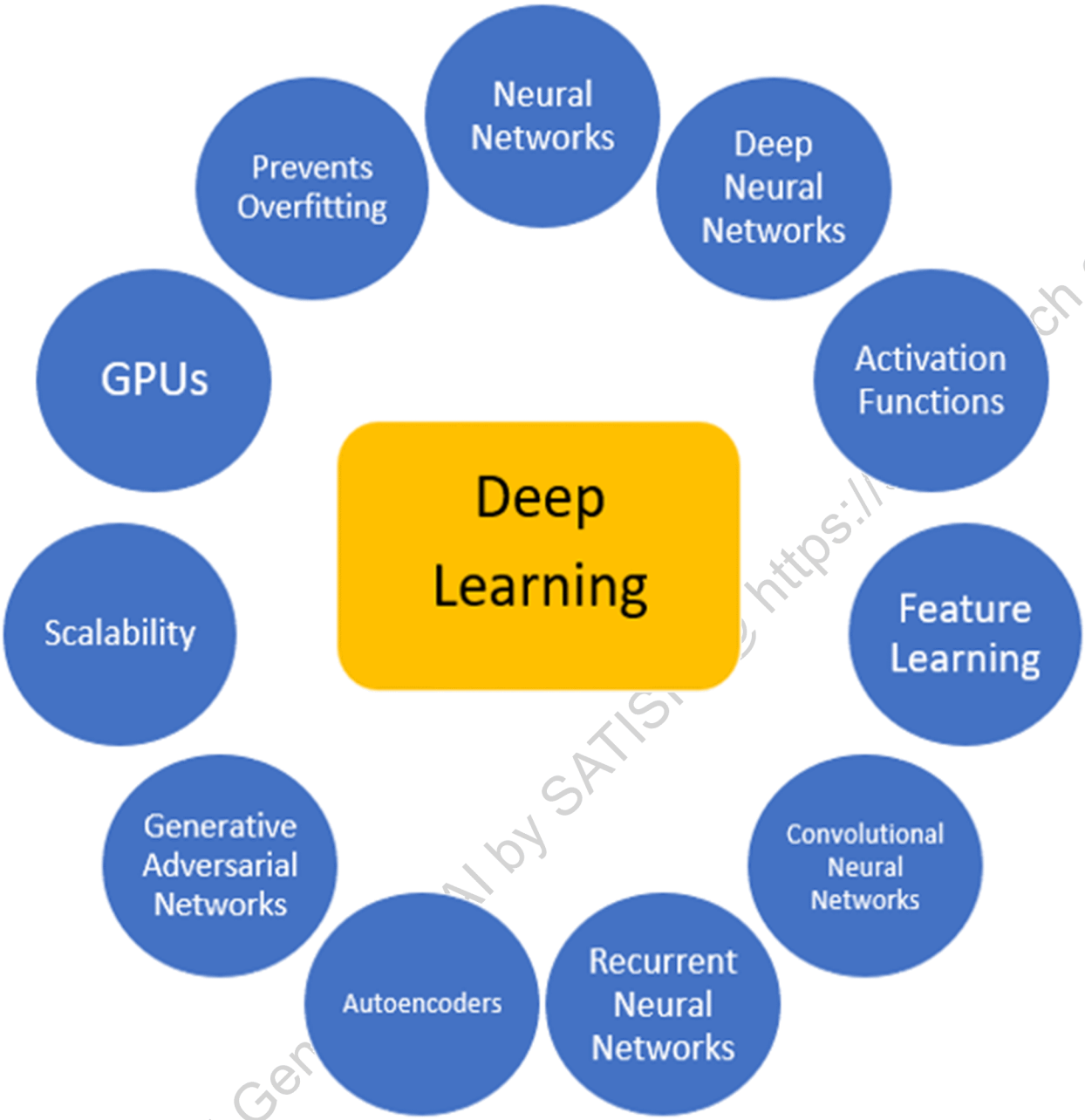
LSTMs use a special type of memory cell that enables them to remember longer sequences and are used for tasks such as recognizing handwriting and predicting stock prices.

Some less common, but still powerful deep learning algorithms include generative adversarial networks (GANs), autoencoders, reinforcement learning, deep belief networks (DBNs), and transfer learning.

- **GANs** can be used for image generation, text-to-image synthesis, and video colorization.
- **Autoencoders** are helpful for data compression and dimensionality reduction.
- **Reinforcement learning** is a type of machine learning in which agents learn to perform tasks by interacting with the environment.
- **DBNs** are primarily used for unsupervised feature learning.
- **Transfer learning** allows models trained on one problem to be reused for another.

Features of Deep Learning

Let us see the features of Deep Learning. It is a subset of Machine Learning and is useful in solving complex problems. One of the most trending topics, Generative AI is also a part of Deep Learning:



1. **Neural Networks:** Neural Networks are the backbone of Deep Learning. It is a series of algorithms that attempt to recognize underlying re0 Neural networks inspired by biological neurons.
2. **Deep Neural Networks (DNNs):** Deep Neural Networks (DNNs) have multiple hidden layers between the input and output layers. The "deep" in deep neural networks refers to the depth, i.e., the number of layers, which can range from a few to hundreds. DLL is a type of artificial neural network.
3. **Activation Functions:** Activation functions determine whether a neuron should be activated or not by introducing non-linearity into the model. This enabled it to learn complex patterns. It is a part of neural networks in deep learning.
4. **Feature Learning:** Feature learning includes automatically discovering and extracting the relevant features or representations needed for a task straight from raw data. Feature Learning is also known as representation learning,
5. **Convolutional Neural Networks (CNNs):** Convolutional Neural Networks (CNNs) are used for image and video processing. They mimic how the human brain processes visual information using layers of convolutional filters. This is done to detect various features in images. CNNs are a type of deep learning model.
6. **Recurrent Neural Networks (RNNs):** Recurrent Neural Networks (RNNs) were designed for processing sequential data. RNNs are a class of neural networks.
7. **Autoencoders:** Autoencoders are used to learn efficient representations of data, mostly for dimensionality reduction or feature learning. They have an encoder and a decoder. Autoencoders are a type of neural network.
8. **Generative Adversarial Networks (GANs):** Generative Adversarial Networks (GANs) are a class of machine learning frameworks designed for unsupervised learning. GANs consist of two neural networks, a generator, and a discriminator.
9. **Scalability:** Deep learning models can handle large volumes of data and complex architectures.
10. **GPUs:** Deep learning takes advantage of modern GPUs (Graphics Processing Units) for parallel computation. This speeds up the training process.
11. **Prevents Overfit:** Regularization techniques like dropout, batch normalization, and weight regularization help prevent overfitting, improving model performance on unseen data.

AI vs ML vs DL

We learned about Deep Learning in the previous lessons. Let us see the difference between _Artificial Intelligence, Machine Learning, and Deep Learning.

	Artificial Intelligence (AI)	Machine Learning (ML)	Deep Learning (DL)
What?	Artificial Intelligence means creating smart machines to mimic human behavior	In Machine Learning, the computer is fed with data and information. This gives these systems the ability to learn and enhance from experiences.	Deep Learning uses neural networks to model and solve complex problems
Subset/ Superset	AI is a superset of both ML and DL	ML is a subset of AI.	DL is a subset of ML.
Easy or Complex	Easier rules to advanced algorithms	Uses algorithms like decision trees, SVM, etc.	Uses complex neural networks with many layers.
Data	Can work with structured data and predefined rules	Needs structured data for training	Needs large amounts of labeled data for effective training
Human Intervention	Human Intervention is high, especially in rule-based systems	Human Intervention is moderate, requires feature extraction	Human Intervention is low, since it has automatic feature extraction
Hardware Requirements	Generally lower, depends on application	Moderate, can run on standard hardware	High, requires GPUs for training deep networks. Modern GPUs (Graphics Processing Units) are used for parallel computation.
Applications/ Uses	Robotics, game playing, etc.	Email filtering, healthcare, recommendation systems, etc.	Image and speech recognition, language processing, etc.
Efficiency	Varies based on complexity and design	Generally good with sufficient data	Often superior performance with large datasets
Learning Techniques	Rule-based learning, expert systems, etc.	Supervised, unsupervised, and reinforcement learning	Primarily supervised and unsupervised learning with deep networks

Deep Learning Advantages & Disadvantages

Deep Learning models achieve higher accuracy than traditional models. It has various other advantages. With that, the complexity of Deep Learning can also be expensive. Let us now see the advantages and disadvantages of deep learning.

Advantages of Deep Learning

The following are the advantages of Deep Learning:

1. **Automatic Feature Extraction**:
   Deep learning models can automatically extract relevant features from raw data, reducing the need for manual feature engineering.
2. **Better Performance**
   Deep learning models, particularly deep neural networks, often achieve higher accura cy and performance compared to traditional machine learning models, especially wit h large datasets.
3. **Processes Unstructured Data**:
   Capable of processing unstructured data such as text, images, and audio, in a better way than traditional models.
4. **Large datasets**:
   Can be scaled to handle large datasets and complex models, with modern GPU or TPUs. TPUs are much faster than GPUs.

Disadvantages of Deep Learning

The following are the disadvantages of Deep Learning:

1. **Require large amounts of data**:
   Deep learning models require large amounts of labeled data to perform well, which c an be expensive and time-consuming to obtain.
2. **Powerful Hardware Requirements**:
   Training deep learning models can be resource-
   intensive, requiring powerful hardware like GPUs or TPUs.
3. **Complexity**:
   Deep learning models are often more complex and harder to interpret compared to t raditional machine learning models. This can be a challenge for understanding the de cision-making process.
4. **Risk of Overfitting**:
   With high complexity, there is a risk of overfitting, where the model performs well on training data but poorly on new, unseen data. Regularization techniques such as dropout, and batch normalization, are needed to mitigate this.

5. **Long Training Time**:
   Training deep learning models can take too much time, especially with large datasets and complex architectures.

Deep Learning Applications

Deep learning has revolutionized various fields with its ability to process vast amounts of data and uncover complex patterns. Here are some key applications:

1. **Image Recognition:** Deep Learning is useful for not only facial recognition but also for object detection, Nowadays, CCTVs can also detect the movement of objects. Here are the applications:**Object Detection:** Identifying and labeling objects within an image. Used in security systems, autonomous vehicles, and medical imaging.**Facial Recognition:** Recognizing and verifying individual faces. Used in social media tagging, security, and user authentication.
2. **Natural Language Processing (NLP):** NL is a branch of artificial intelligence (AI) that uses machine learning to help computers understand, interpret, and manipulate human language. Here are its applications for deep learning:**Machine Translation:** Translating text from one language to another. Used in apps like Google Translate.**Chatbots and Virtual Assistants:** Powering conversational agents like Siri, Alexa, and chatbots for customer service.
3. **Speech Recognition:** Converting spoken language into text. Used in voice-activated assistants, transcription services, and voice commands.
4. **Healthcare:** Deep Learning revolutionized the healthcare industry with disease detection, identifying health risks, etc.**Medical Imaging:** Analyzing X-rays, MRIs, and CT scans to detect diseases. Used in diagnostics and treatment planning.**Predictive Analytics:** Predicting patient outcomes and identifying potential health risks. Used in personalized medicine.
5. **Self-driving Cars:** Enabling self-driving cars to understand and navigate their environment. Used in lane detection, obstacle avoidance, and traffic sign recognition, and considered one of the best applications of Deep Learning in autonomous vehicles.
6. **Finance:** Deep Learning applications also detect frauds, assist in trading decisions, etc.**Fraud Detection:** Identifying fraudulent transactions. Used in banking and online payment systems.
   Algorithmic Trading: Making automated trading decisions based on market data. Used in stock trading.
7. **Entertainment:** Deep Learning is used in recommending videos to Netflix lovers, also in developing game characters, etc.**Content Recommendation:** Suggesting movies, songs, or articles based on user preferences. Used in streaming services like Netflix, Amazon Prime, Spotify, etc.

Game Development: Creating realistic in-game characters and environments. Used in the gaming industry.

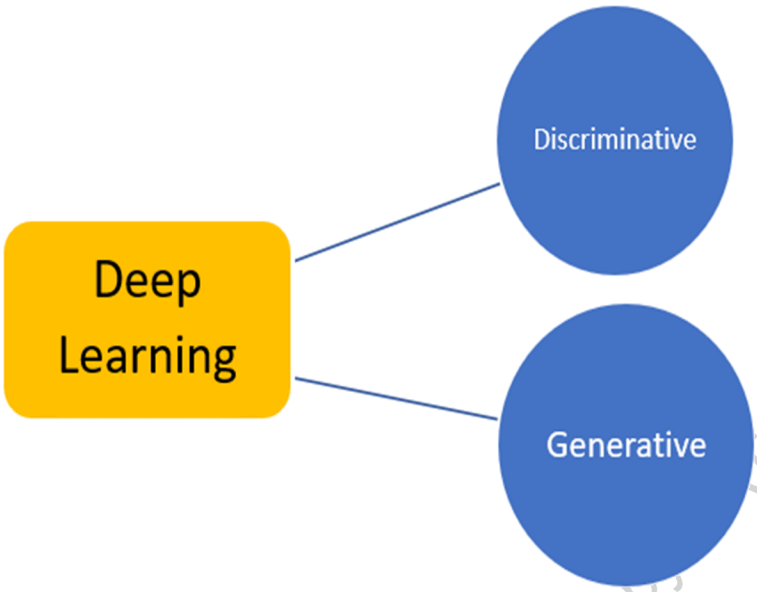
8. **Manufacturing:** Easily find defects in a product, predict failures in equipment, save it from a disaster and in turn save time and money.**Quality Control:** Inspecting products for defects using image recognition. Used in production lines.

Predictive Maintenance: Predicting equipment failures to perform maintenance before a breakdown. Used in factories and industrial settings.

9. **Robots:** Enabling robots to understand and interact with their environment, including object recognition, avoiding obstacles, etc. Deep learning has revolutionized the field of robotics, enabling robots to perform complex tasks with increased accuracy and autonomy.

Types of Deep Learning

We learned about Deep Learning in the previous lessons. Now, we will see the types of Deep Learning: Discriminative and Generative. Let us understand them one by one:



Discriminative

This type of Deep Learning is used to classify or predict. It discriminates between different kinds of data instances. It requires labeled data for training

- **Applications**: Classification tasks, e.g., spam detection, image recognition
- **Algorithms:** Logistic Regression, Support Vector Machines (SVM), Neural Networks
- **Discriminative model algorithm application:** Logistic regression predicts if an email is spam or not by learning the boundary between spam and non-spam emails.
- **Example:** Classify images as a dog or a cat.

Generative AI

Under this type of Deep Learning, new data is generated that is like the data it was trained on. Generates new data instances. It can process both labeled and unlabeled data using supervised, unsupervised, and semi-supervised methods. Generative AI is a subset of Deep Learning. It uses AI neural networks

- **Applications:** Data generation, e.g., creating images, text, music
- **Algorithms:** Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), Naive Bayes
- **Generative Model algorithm application**: GANs can generate realistic-looking images of faces by learning the distribution of real face images.
- **Example:** Generate a dog image.

Deep Learning Example:

```
# Importing the libraries

  import numpy as np

  import matplotlib.pyplot as plt

  import pandas as pd


# Importing the database

  dataset = pd.read_csv('Churn_Modelling.csv')
```

Step 2

We create matrices of the features of dataset and the target variable, which is column 14, labeled as “Exited”.

The initial look of data is as shown below −

In[]:

```
X = dataset.iloc[:, 3:13].values

Y = dataset.iloc[:, 13].values

X
```

Output

```
array([[619, 'France', 'Female', ..., 1, 1, 101348.88],
       [608, 'Spain', 'Female', ..., 0, 1, 112542.58],
       [502, 'France', 'Female', ..., 1, 0, 113931.57],
       ...,
       [709, 'France', 'Female', ..., 0, 1, 42085.58],
       [772, 'Germany', 'Male', ..., 1, 0, 92888.52],
       [792, 'France', 'Female', ..., 1, 0, 38190.78]], dtype=object)
```

Step 3

Y

Output

array([1, 0, 1, ..., 1, 1, 0], dtype = int64)

Step 4

We make the analysis simpler by encoding string variables. We are using the ScikitLearn function 'LabelEncoder' to automatically encode the different labels in the columns with values between 0 to n_classes-1.

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

labelencoder_X_1 = LabelEncoder()

X[:,1] = labelencoder_X_1.fit_transform(X[:,1])

labelencoder_X_2 = LabelEncoder()

X[:, 2] = labelencoder_X_2.fit_transform(X[:, 2])

X
```

Output

```
array([[619, 0, 0, ..., 1, 1, 101348.88],
       [608, 2, 0, ..., 0, 1, 112542.58],
       [502, 0, 0, ..., 1, 0, 113931.57],
       ...,
       [709, 0, 0, ..., 0, 1, 42085.58],
       [772, 1, 1, ..., 1, 0, 92888.52],
       [792, 0, 0, ..., 1, 0, 38190.78]], dtype=object)
```

In the above output,country names are replaced by 0, 1 and 2; while male and female are replaced by 0 and 1.

Step 5

Labelling Encoded Data

We use the same **ScikitLearn** library and another function called the **OneHotEncoder** to just pass the column number creating a dummy variable.

onehotencoder = OneHotEncoder(categorical features = [1])

X = onehotencoder.fit_transform(X).toarray()

X = X[:, 1:]

X

Now, the first 2 columns represent the country and the 4th column represents the gender.

Output

```
array([[0.0000000e+00, 0.0000000e+00, 6.1900000e+02, ..., 1.0000000e+00,
        1.0000000e+00, 1.0134888e+05],
       [0.0000000e+00, 1.0000000e+00, 6.0800000e+02, ..., 0.0000000e+00,
        1.0000000e+00, 1.1254258e+05],
       [0.0000000e+00, 0.0000000e+00, 5.0200000e+02, ..., 1.0000000e+00,
        0.0000000e+00, 1.1393157e+05],
       ...,
       [0.0000000e+00, 0.0000000e+00, 7.0900000e+02, ..., 0.0000000e+00,
        1.0000000e+00, 4.2085580e+04],
       [1.0000000e+00, 0.0000000e+00, 7.7200000e+02, ..., 1.0000000e+00,
        0.0000000e+00, 9.2888520e+04],
       [0.0000000e+00, 0.0000000e+00, 7.9200000e+02, ..., 1.0000000e+00,
        0.0000000e+00, 3.8190780e+04]])
```

We always divide our data into training and testing part; we train our model on training data and then we check the accuracy of a model on testing data which helps in evaluating the efficiency of model.

Step 6

We are using ScikitLearn's **train_test_split** function to split our data into training set and test set. We keep the train- to-test split ratio as 80:20.

#Splitting the dataset into the Training set and the Test Set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

Some variables have values in thousands while some have values in tens or ones. We scale the data so that they are more representative.

Step 7

In this code, we are fitting and transforming the training data using the **StandardScaler** function. We standardize our scaling so that we use the same fitted method to transform/scale test data.

```
# Feature Scaling

fromsklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_train = sc.fit_transform(X_train)

X_test = sc.transform(X_test)
```

Output

```
array([[-0.5698444 ,  1.74309049,  0.16958176, ...,  0.64259497,
        -1.03227043,  1.10643166],
       [ 1.75486502, -0.57369368, -2.30455945, ...,  0.64259497,
         0.9687384 , -0.74866447],
       [-0.5698444 , -0.57369368, -1.19119591, ...,  0.64259497,
        -1.03227043,  1.48533467],
       ...,
       [-0.5698444 , -0.57369368,  0.9015152 , ...,  0.64259497,
        -1.03227043,  1.41231994],
       [-0.5698444 ,  1.74309049, -0.62420521, ...,  0.64259497,
         0.9687384 ,  0.84432121],
       [ 1.75486502, -0.57369368, -0.28401079, ...,  0.64259497,
        -1.03227043,  0.32472465]])
```

The data is now scaled properly. Finally, we are done with our data pre-processing. Now,we will start with our model.

Step 8

We import the required Modules here. We need the Sequential module for initializing the neural network and the dense module to add the hidden layers.

```
# Importing the Keras libraries and packages

import keras

from keras.models import Sequential

from keras.layers import Dense
```

Step 9

We will name the model as Classifier as our aim is to classify customer churn. Then we use the Sequential module for initialization.

```
#Initializing Neural Network

classifier = Sequential()
```

Step 10

We add the hidden layers one by one using the dense function. In the code below, we will see many arguments.

Our first parameter is **output_dim**. It is the number of nodes we add to this layer. **init** is the initialization of the Stochastic Gradient Decent. In a Neural Network we assign weights to each node. At initialization, weights should be near to zero and we randomly initialize weights using the uniform function. The **input_dim** parameter is needed only for first layer, as the model does not know the number of our input variables. Here the total number of input variables is 11. In the second layer, the model automatically knows the number of input variables from the first hidden layer.

Execute the following line of code to addthe input layer and the first hidden layer −

```
classifier.add(Dense(units = 6, kernel_initializer = 'uniform',

activation = 'relu', input_dim = 11))
```

Execute the following line of code to add the second hidden layer −

```
classifier.add(Dense(units = 6, kernel_initializer = 'uniform',

activation = 'relu'))
```

Execute the following line of code to add the output layer −

```
classifier.add(Dense(units = 1, kernel_initializer = 'uniform',

activation = 'sigmoid'))
```

Step 11

Compiling the ANN

We have added multiple layers to our classifier until now. We will now compile them using the **compile** method. Arguments added in final compilation control complete the neural network.So,we need to be careful in this step.

Here is a brief explanation of the arguments.

First argument is **Optimizer**.This is an algorithm used to find the optimal set of weights. This algorithm is called the **Stochastic Gradient Descent (SGD)**. Here we are using one among several types, called the 'Adam optimizer'. The SGD depends on loss, so our second parameter is loss. If our dependent variable is binary, we use logarithmic loss function called **'binary_crossentropy'**, and if our dependent variable has more than two categories in output, then we use **'categorical_crossentropy'**. We want to improve performance of our neural network based on **accuracy**, so we add **metrics** as accuracy.

```
# Compiling Neural Network

classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

Step 12

A number of codes need to be executed in this step.

Fitting the ANN to the Training Set

We now train our model on the training data. We use the **fit** method to fit our model. We also optimize the weights to improve model efficiency. For this, we have to update the weights. **Batch size** is the number of observations after which we update the weights. **Epoch** is the total number of iterations. The values of batch size and epoch are chosen by the trial and error method.

```
classifier.fit(X_train, y_train, batch_size = 10, epochs = 50)
```

Making predictions and evaluating the model

```
# Predicting the Test set results

y_pred = classifier.predict(X_test)

y_pred = (y_pred> 0.5)
```

Predicting a single new observation

```
# Predicting a single new observation

"""Our goal is to predict if the customer with the following data will leave the bank:
```

Geography: Spain

Credit Score: 500

Gender: Female

Age: 40

Tenure: 3

Balance: 50000

Number of Products: 2

Has Credit Card: Yes

Is Active Member: Yes

Step 13

Predicting the test set result

The prediction result will give you probability of the customer leaving the company. We will convert that probability into binary 0 and 1.

```
# Predicting the Test set results

y_pred = classifier.predict(X_test)

y_pred = (y_pred> 0.5)

new_prediction = classifier.predict(sc.transform

(np.array([[0.0, 0, 500, 1, 40, 3, 50000, 2, 1, 1, 40000]])))

new_prediction = (new_prediction> 0.5)
```

Step 14

This is the last step where we evaluate our model performance. We already have original results and thus we can build confusion matrix to check the accuracy of our model.