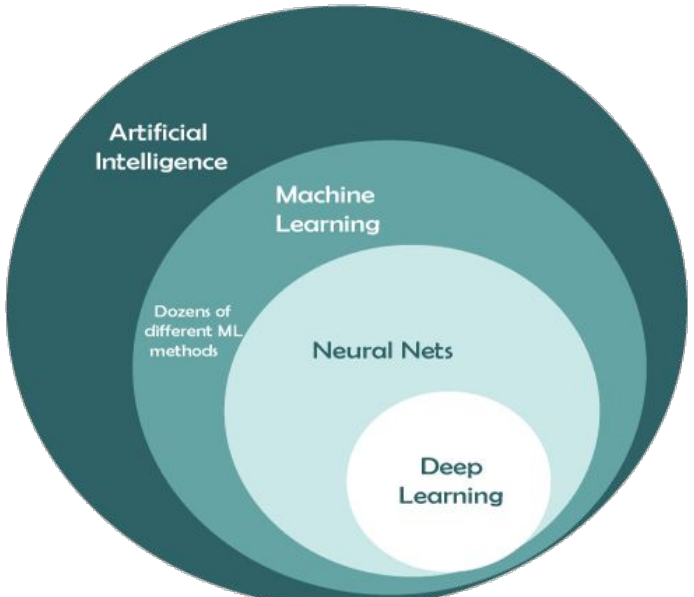
What is machine learning and deep learning?

- Deep learning is a subtype of machine learning. With machine learning, you manually extract the relevant features of an image. With deep learning, you feed the raw images directly into a deep neural network that learns the features automatically.
- Deep learning often requires hundreds of thousands or millions of images for the best results. It’s also computationally intensive and requires a high-performance GPU.

Deep learning vs. Machine learning vs. Artificial Intelligence

- Deep Learning, Machine Learning, and Artificial Intelligence are the most used terms on the internet for IT folks. However, all these three technologies are connected with each other.
- Artificial Intelligence (AI) can be understood as an umbrella that consists of both Machine learning and deep learning.



What is Artificial Intelligence (AI)?

- Artificial Intelligence is defined as a field of science and engineering that deals with making intelligent machines or computers to perform human-like activities.
- Mr. **John McCarthy** is known as the godfather of this amazing invention.

What is Machine Learning?

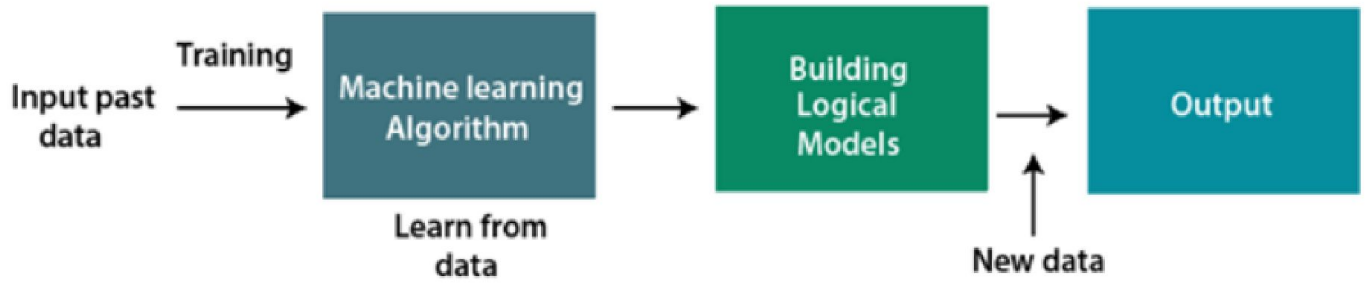
- Machine learning is a growing technology which enables computers to learn automatically from past data. Machine learning uses various algorithms for building mathematical models and making predictions using historical data or information. Currently, it is being used for various tasks such as image recognition, speech recognition, email filtering, Facebook auto-tagging, recommender system, and many more.
- In the real world, we are surrounded by humans who can learn everything from their experiences with their learning capability, and we have computers or machines which work on our instructions. But can a machine also learn from experiences or past data like a human does? So here comes the role of **Machine Learning**.

- Machine Learning is said as a subset of **artificial intelligence** that is mainly concerned with the development of algorithms which allow a computer to learn from the data and past experiences on their own. The term machine learning was first introduced by **Arthur Samuel** in **1959**. We can define it in a summarized way as:
- Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed.

How does Machine Learning work

- A Machine Learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it.
- The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.
- Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machine builds the logic as per the data and predict the output.
- Machine learning has changed our way of thinking about the problem.

- The below block diagram explains the working of Machine Learning algorithm:



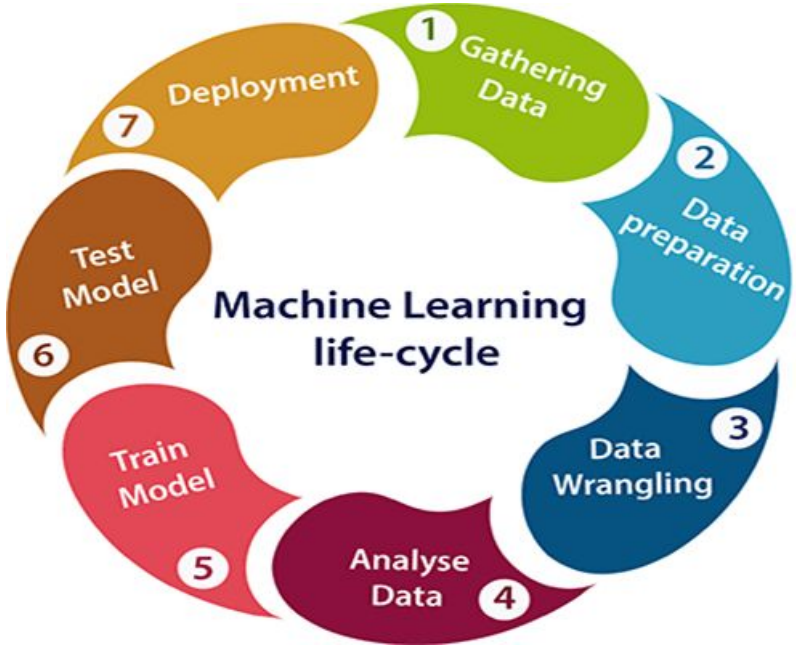
Features of Machine Learning:

- Machine learning uses data to detect various patterns in a given dataset.
- It can learn from past data and improve automatically.
- It is a data-driven technology.
- Machine learning is much similar to data mining as it also deals with the huge amount of the data.

Machine learning Life cycle

- Machine learning has given the computer systems the abilities to automatically learn without being explicitly programmed. But how does a machine learning system work? So, it can be described using the life cycle of machine learning. Machine learning life cycle is a cyclic process to build an efficient machine learning project. The main purpose of the life cycle is to find a solution to the problem or project.
- Machine learning life cycle involves seven major steps, which are given below:

- **Gathering Data**
- **Data preparation**
- **Data Wrangling**
- **Analyse Data**
- **Train the model**
- **Test the model**
- **Deployment**



1. Gathering Data:

- Data Gathering is the first step of the machine learning life cycle. The goal of this step is to identify and obtain all data-related problems.
- In this step, we need to identify the different data sources, as data can be collected from various sources such as **files**, **database**, **internet**, or **mobile devices**. It is one of the most important steps of the life cycle. The quantity and quality of the collected data will determine the efficiency of the output. The more will be the data, the more accurate will be the prediction.

This step includes the below tasks:

- **Identify various data sources**
- **Collect data**
- **Integrate the data obtained from different sources**
- By performing the above task, we get a coherent set of data, also called as a **dataset**. It will be used in further steps.

2. Data preparation

- After collecting the data, we need to prepare it for further steps. Data preparation is a step where we put our data into a suitable place and prepare it to use in our machine learning training.
- In this step, first, we put all data together, and then randomize the ordering of data.
- This step can be further divided into two processes:
- **Data exploration:**
  It is used to understand the nature of data that we have to work with. We need to understand the characteristics, format, and quality of data. A better understanding of data leads to an effective outcome. In this, we find Correlations, general trends, and outliers.
- **Data pre-processing:**
  Now the next step is preprocessing of data for its analysis.

3. Data Wrangling

- Data wrangling is the process of cleaning and converting raw data into a useable format. It is the process of cleaning the data, selecting the variable to use, and transforming the data in a proper format to make it more suitable for analysis in the next step. It is one of the most important steps of the complete process. Cleaning of data is required to address the quality issues.
- It is not necessary that data we have collected is always of our use as some of the data may not be useful. In real-world applications, collected data may have various issues, including:
- **Missing Values**
- **Duplicate data**
- **Invalid data**
- **Noise**

So, we use various filtering techniques to clean the data.

4. Data Analysis

- Now the cleaned and prepared data is passed on to the analysis step. This step involves:
- **Selection of analytical techniques**
- **Building models**
- **Review the result**
- The aim of this step is to build a machine learning model to analyze the data using various analytical techniques and review the outcome. It starts with the determination of the type of the problems, where we select the machine learning techniques such as **Classification**, **Regression**, **Cluster analysis**, **Association**, etc. then build the model using prepared data, and evaluate the model.

5. Train Model

- Now the next step is to train the model, in this step we train our model to improve its performance for better outcome of the problem.
- We use datasets to train the model using various machine learning algorithms. Training a model is required so that it can understand the various patterns, rules, and, features.

6. Test Model

- Once our machine learning model has been trained on a given dataset, then we test the model. In this step, we check for the accuracy of our model by providing a test dataset to it.
- Testing the model determines the percentage accuracy of the model as per the requirement of project or problem.

7. Deployment

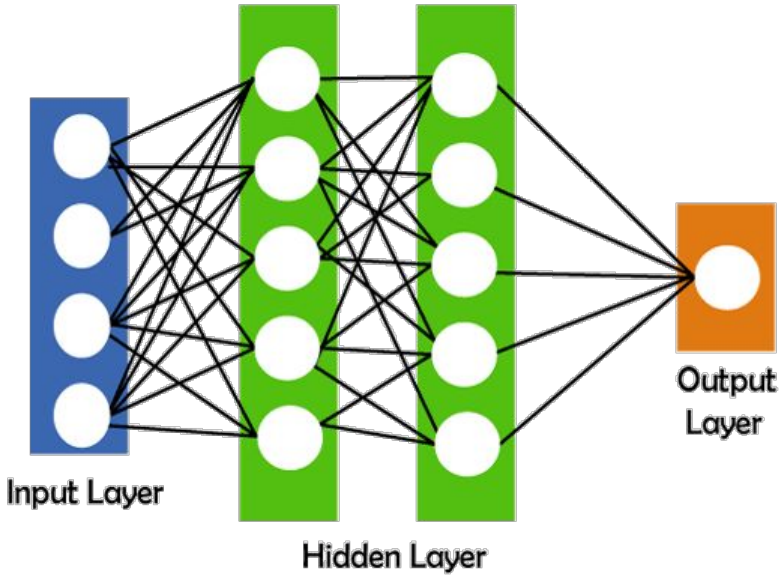
- The last step of machine learning life cycle is deployment, where we deploy the model in the real-world system.
- If the above-prepared model is producing an accurate result as per our requirement with acceptable speed, then we deploy the model in the real system. But before deploying the project, we will check whether it is improving its performance using available data or not. The deployment phase is similar to making the final report for a project.

What is Deep Learning?

- "Deep learning is defined as the subset of machine learning and artificial intelligence that is based on artificial neural networks". In deep learning, the deep word refers to the number of layers in a neural network.
- Deep Learning is a set of algorithms inspired by the structure and function of the human brain. It uses a huge amount of structured as well as unstructured data to teach computers and predicts accurate results.
- Deep learning can be useful to solve many complex problems with more accurate predictions such as image recognition, voice recognition, product recommendations systems, natural language processing (NLP), etc.

The basic structure of deep learning

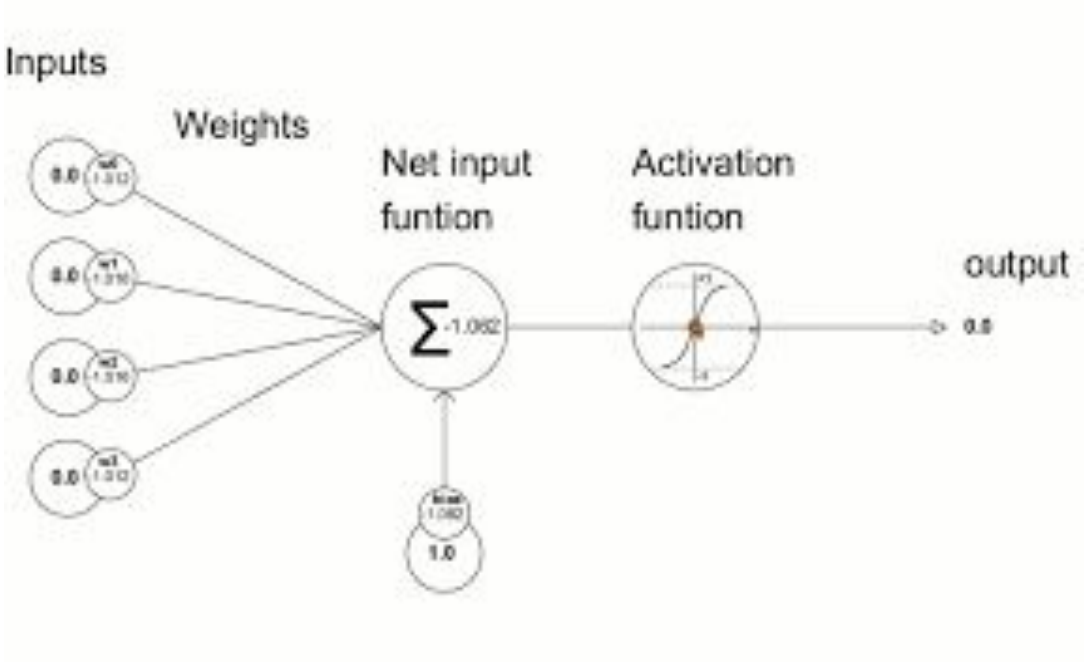
- Deep learning includes various neural networks that possess different layers, such as input layers, hidden layers, and output layers. The input layer accepts input data; hidden layers are used to find any hidden pattern and feature from the data, and output layers show the expected results.



How does deep learning work?

There are a few simple steps that deep learning follows.

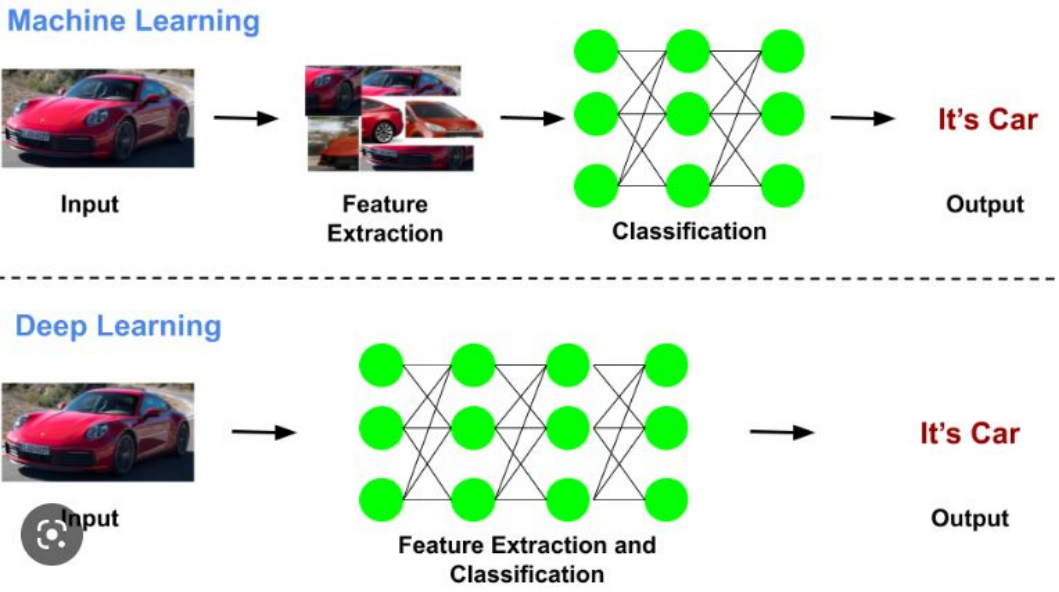
1. Calculate the weighted sum
2. Use this weighted sum in step1 as input for the activation function.
3. The activation function adds bias and decides whether the neuron should be triggered or not.
4. Predict output at the output layer.
5. Compare predicted output and actual output and accordingly use the backpropagation method for improving the performance of the model.



Types of deep neural networks

There are some different types of deep learning networks available. These are as follows:

- Feedforward neural network
- Radial basis function neural networks
- Multi-layer perceptron
- Convolution neural network (CNN)
- Recurrent neural network
- Modular neural network
- Sequence to sequence models



Machine Learning	Deep Learning
+ Good results with small data sets	– Requires very large data sets
+ Quick to train a model	– Computationally intensive
– Need to try different features and classifiers to achieve best results	+ Learns features and classifiers automatically
– Accuracy plateaus	+ Accuracy is unlimited

	Deep Learning	Machine Learning
Data	Needs a big dataset	Performs well with a small to a medium dataset
Hardware requirements	Requires machines with GPU	Works with low-end machines
Engineering peculiarities	Needs to understand the basic functionality of the data	Understands the features and how they represent the data
Training time	Long	Short
Processing time	A few hours or weeks	A few seconds or hours
Number of algorithms	Few	Many
Data interpetation	Difficult	Some ML algorithms are easy to interpret, whereas some are hardly possible

Supervised Learning

- Supervised learning, as the name indicates, has the presence of a supervisor as a teacher. Basically supervised learning is when we teach or train the machine using data that is well labelled. Which means some data is already tagged with the correct answer. After that, the machine is provided with a new set of examples(data) so that the supervised learning algorithm analyses the training data(set of training examples) and produces a correct outcome from labelled data.

For instance, suppose you are given a basket filled with different kinds of fruits. Now the first step is to train the machine with all the different fruits one by one like this:

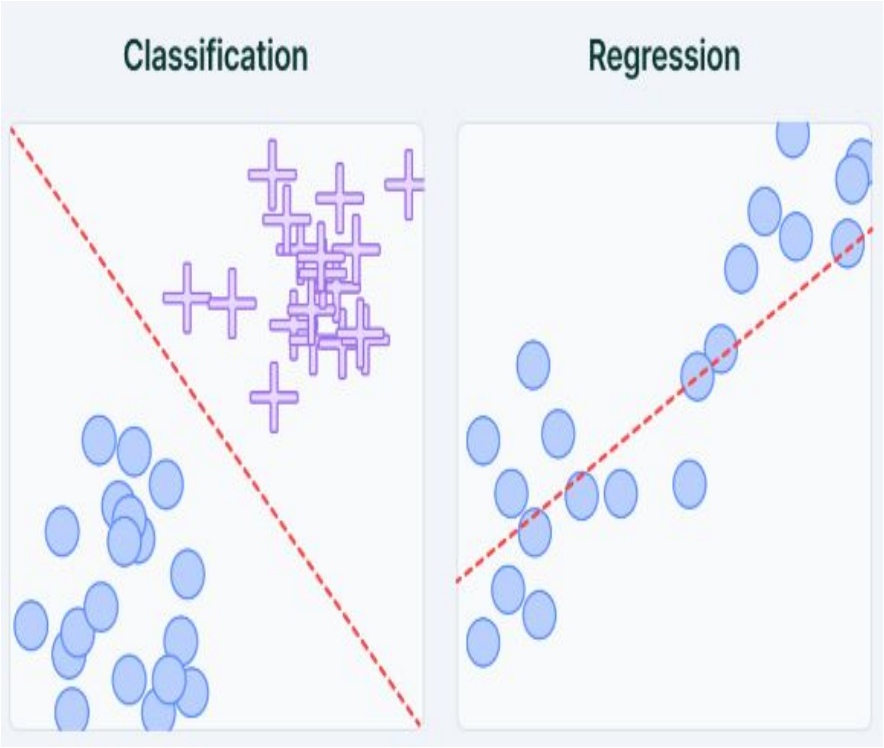
- If the shape of the object is rounded and has a depression at the top, is red in color, then it will be labeled as –**Apple**.
- If the shape of the object is a long curving cylinder having Green-Yellow color, then it will be labeled as –**Banana**.
- Now suppose after training the data, you have given a new separate fruit, say Banana from the basket, and asked to identify it.

- Since the machine has already learned the things from previous data and this time has to use it wisely.
- It will first classify the fruit with its shape and color and would confirm the fruit name as BANANA and put it in the Banana category.

- Thus the machine learns the things from training data(basket containing fruits) and then applies the knowledge to test data(new fruit).

- Image segmentation, medical diagnosis, fraud detection, spam detection, speech recognition, etc., are some important applications of supervised learning.

- Supervised learning can be further categorized into 2 types of problems as follows:
  Classification
  Regression



- **Classification** refers to taking an input value and mapping it to a discrete value. This could be things like trying to predict what objects are present in an image (a cat/ a dog) or whether it is going to rain today or not.
- **Regression** is related to continuous data (value functions). In Regression, the predicted output values are real numbers. It deals with problems such as predicting the price of a house or the trend in the stock price at a given time, etc.

Advantages of Supervised learning

- Supervised machine learning helps to predict output based on prior experience
- It helps to provide an exact idea about classes of objects.

Disadvantages of Supervised learning

- This method is not significant in solving complex problems.
- This method does not guarantee to give exact output as it contains both structured and unstructured data.
- It needs more computational time to teach models.

Unsupervised Learning

- Unsupervised learning is the training of a machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance.
- Here the task of the machine is to group unsorted information according to similarities, patterns, and differences without any prior training of data.

- Unlike supervised learning, no teacher is provided that means no training will be given to the machine. Therefore the machine is restricted to find the hidden structure in unlabeled data by itself.
- **For instance**, suppose it is given an image having both dogs and cats which it has never seen.

- Thus the machine has no idea about the features of dogs and cats so we can't categorize it as 'dogs and cats '.
- But it can categorize them according to their similarities, patterns, and differences, i.e., we can easily categorize the above picture into two parts.
- The first may contain all pics having **dogs** in them and the second part may contain all pics having **cats** in them.
- Here you didn't learn anything before, which means no training data or examples.

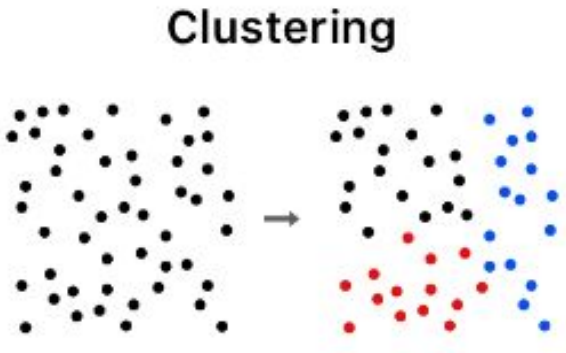
It allows the model to work on its own to discover patterns and information that was previously undetected. It mainly deals with unlabelled data.

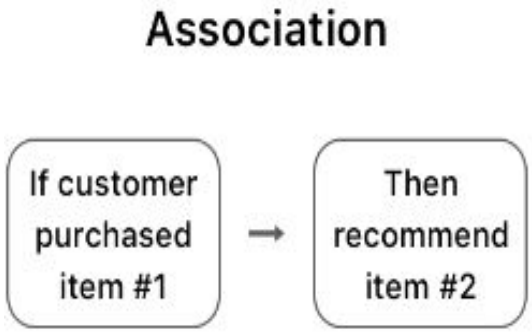
- Network analysis, recommendation system, anomaly detection, singular value decompositions, etc., are some important applications of unsupervised learning.
- Types of Unsupervised Learning:

  Clustering

  Associations

- Clustering is the type of Unsupervised Learning where we find hidden patterns in the data based on their similarities or differences. These patterns can relate to the shape, size, or color and are used to group data items or create clusters.
- The association rule is used to find the probability of co-occurrence of items in a collection. These techniques are often utilized in customer behavior analysis in e-commerce websites and OTT platforms.





Advantages of unsupervised learning

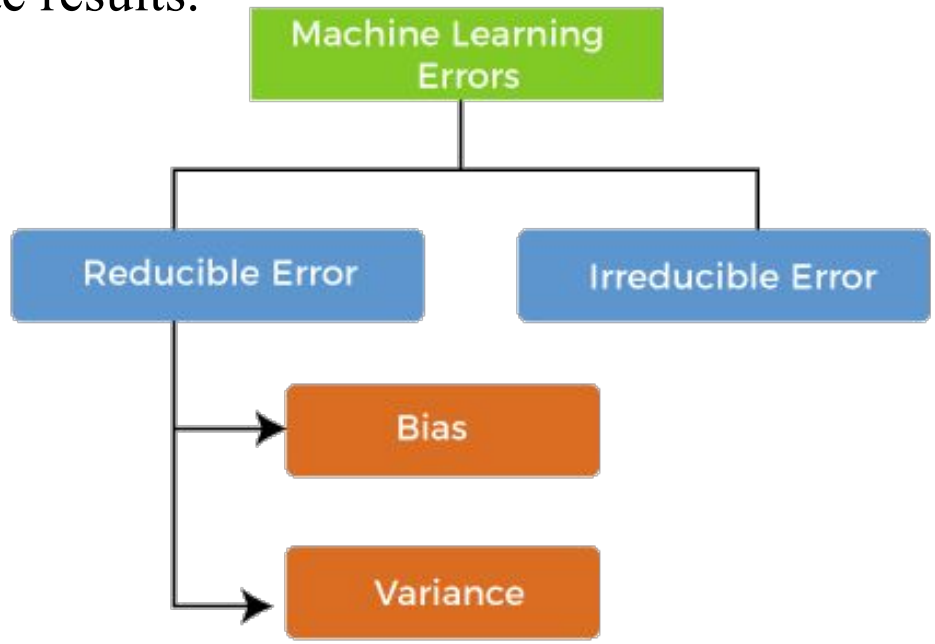
- It can be used to solve complex ML problems as it works with unlabelled data sets.
- It is used to solve multiple tasks in comparison to supervised learning.

Disadvantages of unsupervised learning

- Using unlabeled data sets may predict inaccurate outputs.
- It is a relatively complex algorithm as it deals with unlabelled datasets and also does not map with output.

Bias and Variance

- If the machine learning model is not accurate, it can make predictions errors, and these prediction errors are usually known as Bias and Variance.
- In machine learning, these errors will always be present as there is always a slight difference between the model predictions and actual predictions.
- The main aim of ML/data science analysts is to reduce these errors in order to get more accurate results.



What is Bias?

- In general, a machine learning model analyses the data, find patterns in it and make predictions. While training, the model learns these patterns in the dataset and applies them to test data for prediction.
- ***While making predictions, a difference occurs between prediction values made by the model and actual values/expected values**, **and this difference is known as bias errors or Errors due to bias***.
- It can be defined as an inability of machine learning algorithms such as Linear Regression to capture the true relationship between the data points.
- A model has either:
- **Low Bias:** A low bias model will make fewer assumptions about the form of the target function.
- **High Bias:** A model with a high bias makes more assumptions, and the model becomes unable to capture the important features of our dataset. **A high bias model also cannot perform well on new data.**

Bias

- Some examples of machine learning algorithms with low bias **are Decision Trees, k-Nearest Neighbours and Support Vector Machines**. At the same time, an algorithm with high bias is **Linear Regression and Logistic Regression.**

High bias mainly occurs due to a much simple model. Below are some ways to reduce the high bias:

- Increase the input features as the model is underfitted.
- Use more complex models, such as including some polynomial features.

What is a Variance Error?

- The variance would specify the amount of variation in the prediction if the different training data was used. In simple words, ***variance tells that how much a random variable is different from its expected value.***
- Ideally, a model should not vary too much from one training dataset to another, which means the algorithm should be good in understanding the hidden mapping between inputs and output variables. Variance errors are either of **low variance or high variance.**
- **Low variance** means there is a small variation in the prediction of the target function with changes in the training data set.
- **High variance** shows a large variation in the prediction of the target function with changes in the training dataset.

Since, with high variance, the model learns too much from the dataset, it leads to overfitting of the model. A model with high variance has the below problems:

- A high variance model leads to overfitting.
- Increase model complexities.
- Usually, nonlinear algorithms have a lot of flexibility to fit the model, have high variance.
- Some examples of machine learning algorithms with low variance are, **Linear Regression, Logistic Regression, and Linear discriminant analysis**. At the same time, algorithms with high variance are **decision tree, Support Vector Machine, and K-nearest neighbours.**

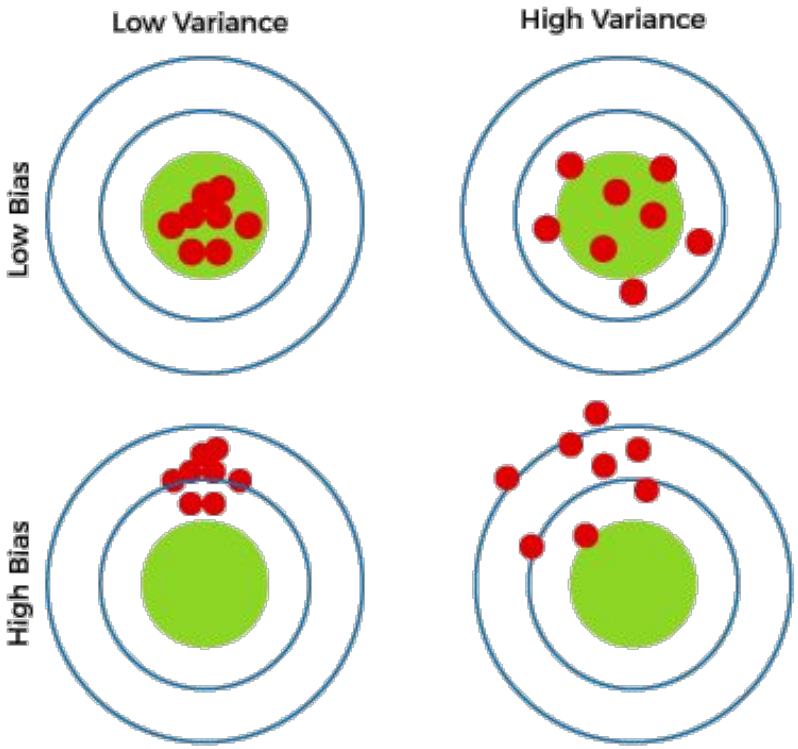
Ways to Reduce High Variance:

Reduce the input features or number of parameters as a modelis overfitted.

- Do not use a much complex model.
- Increase the training data.

Different Combinations of Bias-Variance

There are four possible combinations of bias and variances, which are represented by the below diagram:



1. **Low-Bias, Low-Variance:**
   The combination of low bias and low variance shows an ideal machine learning model. However, it is not possible practically.

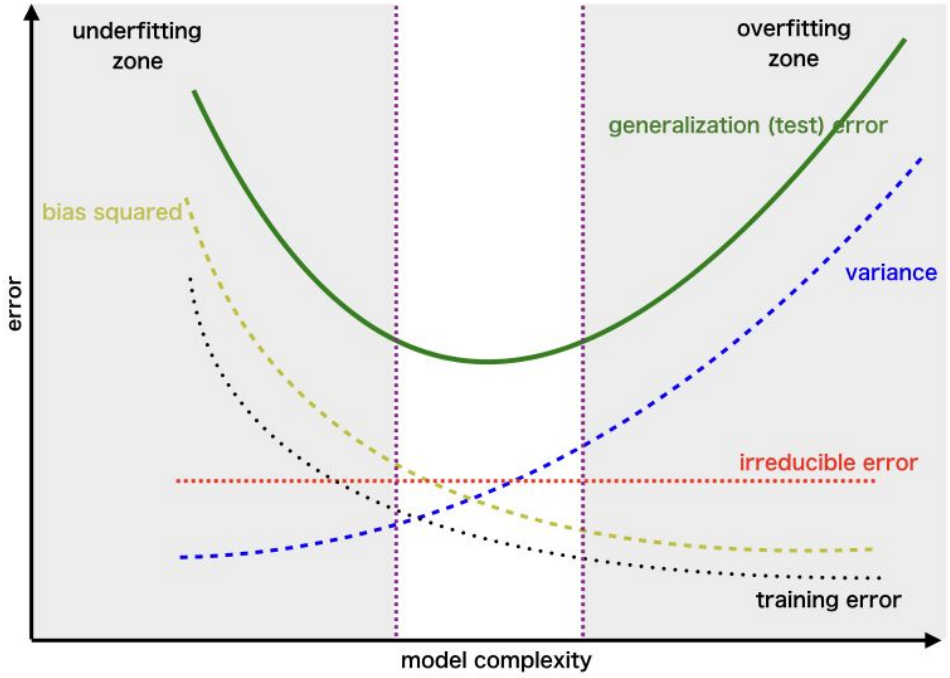
2. **Low-Bias, High-Variance:** With low bias and high variance, model predictions are inconsistent and accurate on average. This case occurs when the model learns with a large number of parameters and hence leads to an **overfitting**

3. **High-Bias, Low-Variance:** With High bias and low variance, predictions are consistent but inaccurate on average. This case occurs when a model does not learn well with the training dataset or uses few numbers of the parameter. It leads to **underfitting** problems in the model.

4. **High-Bias, High-Variance:**
   With high bias and high variance, predictions are inconsistent and also inaccurate on average.

Bias-Variance Trade-Off

- While building the machine learning model, it is really important to take care of bias and variance in order to avoid overfitting and underfitting in the model.
- If the model is very simple with fewer parameters, it may have low variance and high bias. Whereas, if the model has a large number of parameters, it will have high variance and low bias. So, it is required to make a balance between bias and variance errors, and this balance between the bias error and variance error is known as **the Bias-Variance trade-off.**



Hyper Parameters

- **Hyperparameters are those parameters that are explicitly defined by the user to control the learning process.**
- These hyperparameters are used to improve the learning of the model, and their values are set before starting the learning process of the model.
- The value of the Hyperparameter is selected and set by the engineer before the learning algorithm begins training the model.
- **Hence, these are external to the model, and their values cannot be changed during the training process**.

Some examples of Hyperparameters

- The k in kNN or K-Nearest Neighbour algorithm
- Learning rate for training a neural network
- Train-test split ratio
- Number of Epochs
- Branches in Decision Tree
- Number of clusters in Clustering Algorithm

Model Parameters:

- Model parameters are configuration variables that are internal to the model, and a model learns them on its own. For example**, W Weights or Coefficients of independent variables in the Linear regression model, and biases of a neural network, cluster centroid in clustering.**

Some key points for model parameters are as follows:

- They are used by the model for making predictions.
- They are learned by the model from the data itself
- These are usually not set manually.
- These are the part of the model and key to a machine learning Algorithm.

Model Hyperparameters:

- Hyperparameters are those parameters that are explicitly defined by the user to control the learning process. Some key points for model parameters are as follows:
- These are usually defined manually by the machine learning engineer.
- One cannot know the exact best value for hyperparameters for the given problem. The best value can be determined either by the rule of thumb or by trial and error.
- Some examples of Hyperparameters are **the learning rate for training a neural network.**

Categories of Hyperparameters

Hyperparameter for Optimization

The process of selecting the best hyperparameters to use is known as hyperparameter tuning, and the tuning process is also known as hyperparameter optimization.

- **Learning Rate:** The learning rate is the hyperparameter in optimization algorithms that controls how much the model needs to change in response to the estimated error for each time when the model's weights are updated.
- It is one of the crucial parameters while building a neural network, and also it determines the frequency of cross-checking with model parameters. Selecting the optimized learning rate is a challenging task because if the learning rate is very less, then it may slow down the training process. On the other hand, if the learning rate is too large, then it may not optimize the model properly.

Batch Size: To enhance the speed of the learning process, the training set is divided into different subsets, which are known as a batch.

Number of Epochs: An epoch can be defined as the complete cycle for training the machine learning model.

- Epoch represents an iterative learning process.
- The number of epochs varies from model to model, and various models are created with more than one epoch.
- To determine the right number of epochs, a validation error is taken into account.
- The number of epochs is increased until there is a reduction in a validation error. If there is no improvement in reduction error for the consecutive epochs, then it indicates to stop increasing the number of epochs.

Hyperparameter for Specific Models

A number of Hidden Units: Hidden units are part of neural networks, which refer to the components comprising the layers of processors between input and output units in a neural network.

Number of Layers: A neural network is made up of vertically arranged components, which are called layers.

- There are mainly **input layers, hidden layers, and output layers**. A 3-layered neural network gives a better performance than a 2-layered network.
- For a Convolutional Neural network, a greater number of layers make a better model.

Overfitting vs Underfitting

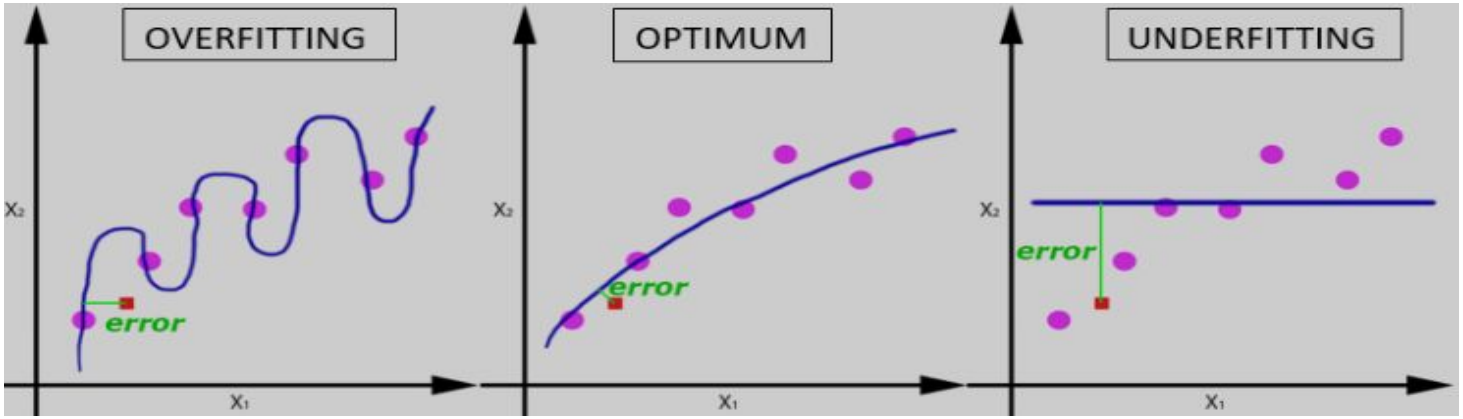
These terms describe two opposing extremes which both result in poor performance.

Overfitting refers to a model that was trained too much on the particulars of the training data (when the model learns the noise in the dataset). A model that is overfit will not perform well on new, unseen data.

Underfitting typically refers to a model that has not been trained sufficiently. This could be due to insufficient training time or a model that was simply not trained properly. A model that is underfit will perform poorly on the training data as well as new, unseen data alike.

As the number of training iterations increases, the parameters of the model are updated and the curve goes from underfitting to optimal to overfitting.

The optimal state is is referred to as **generalization**. This is where the model performs well on both training data and new data not seen during the training process.



Limitations of Machine Learning

1. Inadequate Training Data

- The major issue that comes while using machine learning algorithms is the lack of data. Although data plays a vital role in the processing of machine learning algorithms, many data scientists claim that inadequate data, noisy data, and unclean data are extremely exhausting the machine learning algorithms.
- For example, a simple task requires thousands of sample data, and an advanced task such as speech or image recognition needs millions of sample data examples.

2.Poor quality of data

As we have discussed above, data plays a significant role in machine learning, and it must be of good quality as well. Noisy data, incomplete data, inaccurate data, and unclean data lead to less accuracy in classification and low-quality results. Hence, data quality can also be considered as a major common problem while processing machine learning algorithms.

3.Non-representative training data

- To make sure our training model is generalized well or not, we have to ensure that sample training data must be representative of new cases that we need to generalize. The training data must cover all cases that are already occurred as well as occurring.
- Further, if we are using non-representative training data in the model, it results in less accurate predictions. A machine learning model is said to be ideal if it predicts well for generalized cases and provides accurate decisions. If there is less training data, then there will be a sampling noise in the model, called the non-representative training set. It won't be accurate in predictions. To overcome this, it will be biased against one class or a group.
- Hence, we should use representative data in training to protect against being biased and make accurate predictions without any drift.

4. Overfitting and Underfitting

- Overfitting is one of the most common issues faced by Machine Learning engineers and data scientists. Whenever a machine learning model is trained with a huge amount of data, it starts capturing noise and inaccurate data into the training data set. It negatively affects the performance of the model.
- Underfitting is just the opposite of overfitting. Whenever a machine learning model is trained with fewer amounts of data, and as a result, it provides incomplete and inaccurate data and destroys the accuracy of the machine learning model.

5.Monitoring and maintenance

- As we know that generalized output data is mandatory for any machine learning model; hence, regular monitoring and maintenance become compulsory for the same. Different results for different actions require data change; hence editing of codes as well as resources for monitoring them also become necessary.

6.Getting bad recommendations

- A machine learning model operates under a specific context which results in bad recommendations and concept drift in the model.
- Let's understand with an example where at a specific time customer is looking for some gadgets, but now customer requirement changed over time but still machine learning model showing same recommendations to the customer while customer expectation has been changed.
- This incident is called a Data Drift. It generally occurs when new data is introduced or interpretation of data changes. However, we can overcome this by regularly updating and monitoring data according to the expectations.

7. Lack of skilled resources

Although Machine Learning and Artificial Intelligence are continuously growing in the market, still these industries are fresher in comparison to others. The absence of skilled resources in the form of manpower is also an issue. Hence, we need manpower having in-depth knowledge of mathematics, science, and technologies for developing and managing scientific substances for machine learning.

8. Customer Segmentation

Customer segmentation is also an important issue while developing a machine learning algorithm. To identify the customers who paid for the recommendations shown by the model and who don't even check them. Hence, an algorithm is necessary to recognize the customer behavior and trigger a relevant recommendation for the user based on past experience.

9. Process Complexity of Machine Learning

The machine learning process is very complex, which is also another major issue faced by machine learning engineers and data scientists. However, Machine Learning and Artificial Intelligence are very new technologies but are still in an experimental phase and continuously being changing over time. There is the majority of hits and trial experiments; hence the probability of error is higher than expected. Further, it also includes analyzing the data, removing data bias, training data, applying complex mathematical calculations, etc., making the procedure more complicated and quite tedious.

10. Data Bias

Data Biasing is also found a big challenge in Machine Learning. These errors exist when certain elements of the dataset are heavily weighted or need more importance than others. Biased data leads to inaccurate results, skewed outcomes, and other analytical errors. However, we can resolve this error by determining where data is actually biased in the dataset. Further, take necessary steps to reduce it.

11. Lack of Explainability

- This basically means the outputs cannot be easily comprehended as it is programmed in specific ways to deliver for certain conditions. Hence, a lack of explainability is also found in machine learning algorithms which reduce the credibility of the algorithms.

12. Slow implementations and results

- This issue is also very commonly seen in machine learning models. However, machine learning models are highly efficient in producing accurate results but are time-consuming. Slow programming, excessive requirements' and overloaded data take more time to provide accurate results than expected. This needs continuous maintenance and monitoring of the model for delivering accurate results.

13. Irrelevant features

- Although machine learning models are intended to give the best possible outcome, if we feed garbage data as input, then the result will also be garbage. Hence, we should use relevant features in our training sample. A machine learning model is said to be good if training data has a good set of features or less to no irrelevant features.

History of Deep Learning

The 1943

- The history of deep learning can be traced back to 1943, when Walter Pitts and Warren McCulloch created a computer model based on the neural networks of the human brain.

The 1960s

- Henry J. Kelley is given credit for developing the basics of a continuous Back Propagation Model in 1960.
- In 1962, a simpler version based only on the chain rule was developed by Stuart Dreyfus. The earliest efforts in developing deep learning algorithms came from Alexey Grigoryevich Ivakhnenko (developed the Group Method of Data Handling) and Valentin Grigor'evich Lapa (author of Cybernetics and Forecasting Techniques) in 1965.

- They used models with polynomial (complicated equations) activation functions, that were then analyzed statistically. From each layer, the best statistically chosen features were then forwarded on to the next layer (a slow, manual process).

The 1970s

- During the 1970's the first AI winter kicked in, the result of promises that couldn't be kept.
- The first "convolutional neural networks" were used by Kunihiko Fukushima. Fukushima designed neural networks with multiple pooling and convolutional layers.
- In 1979, he developed an artificial neural network, called Neocognitron, which used a hierarchical, multilayered design. This design allowed the computer the "learn" to recognize visual patterns.

- The networks resembled modern versions, but were trained with a reinforcement strategy of recurring activation in multiple layers, which gained strength over time.
- Additionally, Fukushima’s design allowed important features to be adjusted manually by increasing the “weight” of certain connections.

The 1980s and 90s

- In 1989, Yann LeCun provided the first practical demonstration of backpropagation at Bell Labs.
- He combined convolutional neural networks with back propagation onto read “handwritten” digits.
- This system was eventually used to read the numbers of handwritten checks.
- In 1995, Dana Cortes and Vladimir Vapnik developed the support vector machine (a system for mapping and recognizing similar data).
- LSTM (long short-term memory) for recurrent neural networks was developed in 1997, by Sepp Hochreiter and Juergen Schmidhuber.

2000-2010

- Around the year 2000, The Vanishing Gradient Problem appeared. It was discovered “features” (lessons) formed in lower layers were not being learned by the upper layers, because no learning signal reached these layers.
- Two solutions used to solve this problem were layer-by-layer pre-training and the development of long short-term memory.
- In 2001, a research report by META Group (now called Gartner) described he challenges and opportunities of data growth as three-dimensional.
- The report described the increasing volume of data and the increasing speed of data as increasing the range of data sources and types. This was a call to prepare for the onslaught of Big Data, which was just starting.

- In 2009, Fei-Fei Li, an AI professor at Stanford launched ImageNet, assembled a free database of more than 14 million labeled images. The Internet is, and was, full of unlabeled images. Labeled images were needed to “train” neural nets.
- Professor Li said, “Our vision was that big data would change the way machine learning works. Data drives learning.”

2011-Present

- By 2011, the speed of GPUs had increased significantly, making it possible to train convolutional neural networks “without” the layer-by-layer pre-training. With the increased computing speed, it became obvious deep learning had significant advantages in terms of efficiency and speed.
- One example is AlexNet, a convolutional neural network whose architecture won several international competitions during 2011 and 2012. Rectified linear units were used to enhance the speed and dropout.
- Also in 2012, Google Brain released the results of an unusual project known as The Cat Experiment. The free-spirited project explored the difficulties of “unsupervised learning.”
- Deep learning uses “supervised learning,” meaning the convolutional neural net is trained using labeled data (think images from ImageNet). Using unsupervised learning, a convolutional neural net is given unlabeled data, and is then asked to seek out recurring patterns.

- The Cat Experiment used a neural net spread over 1,000 computers. Ten million “unlabeled” images were taken randomly from YouTube, shown to the system, and then the training software was allowed to run. At the end of the training, one neuron in the highest layer was found to respond strongly to the images of cats. Andrew Ng, the project’s founder said, “We also found a neuron that responded very strongly to human faces.” Unsupervised learning remains a significant goal in the field of deep learning.
- The Generative Adversarial Neural Network (GAN) was introduced in 2014. GAN was created by Ian Goodfellow. With GAN, two neural networks play against each other in a game. The goal of the game is for one network to imitate a photo, and trick its opponent into believing it is real. The opponent is, of course, looking for flaws. The game is played until the near perfect photo tricks the opponent. GAN provides a way to perfect a product (and has also begun being used by scammers).

Advantages of Deep Learning:

1. **Automatic feature learning:** Deep learning algorithms can automatically learn features from the data, which means that they don't require the features to be hand-engineered. This is particularly useful for tasks where the features are difficult to define, such as image recognition.
2. **Handling large and complex data:** Deep learning algorithms can handle large and complex datasets that would be difficult for traditional machine learning algorithms to process. This makes it a useful tool for extracting insights from big data.
3. **Improved performance:** Deep learning algorithms have been shown to achieve state-of-the-art performance on a wide range of problems, including image and speech recognition, natural language processing, and computer vision.
4. **Handling non-linear relationships:** Deep learning can uncover non-linear relationships in data that would be difficult to detect through traditional methods.

5. Handling structured and unstructured data: Deep learning algorithms can handle both structured and unstructured data such as images, text, and audio.

6.Predictive modeling: Deep learning can be used to make predictions about future events or trends, which can help organizations plan for the future and make strategic decisions.

7. Handling missing data: Deep learning algorithms can handle missing data and still make predictions, which is useful in real-world applications where data is often incomplete.

8.Handling sequential data: Deep learning algorithms such as Recurrent Neural Networks (RNNs) and Long Short-term Memory (LSTM) networks are particularly suited to handle sequential data such as time series, speech, and text. These algorithms have the ability to maintain context and memory over time, which allows them to make predictions or decisions based on past inputs.

9. Scalability: Deep learning models can be easily scaled to handle an increasing amount of data and can be deployed on cloud platforms and edge devices.

10. Generalization: Deep learning models can generalize well to new situations or contexts, as they are able to learn abstract and hierarchical representations of the data.

Challenges of Deep Learning:

While deep learning has many advantages, there are also some Challenges to consider:

1. **High computational cost:** Training deep learning models requires significant computational resources, including powerful GPUs and large amounts of memory. This can be costly and time-consuming.
2. **Overfitting:** Overfitting occurs when a model is trained too well on the training data and performs poorly on new, unseen data. This is a common problem in deep learning, especially with large neural networks, and can be caused by a lack of data, a complex model, or a lack of regularization.
3. **Lack of interpretability:** Deep learning models, especially those with many layers, can be complex and difficult to interpret. This can make it difficult to understand how the model is making predictions and to identify any errors or biases in the model.
4. **Dependence on data quality:** Deep learning algorithms rely on the quality of the data they are trained on. If the data is noisy, incomplete, or biased, the model's performance will be negatively affected.
5. **Data privacy and security concerns:** As deep learning models often rely on large amounts of data, there are concerns about data privacy and security. Misuse of data by malicious actors can lead to serious consequences like identity theft, financial loss and invasion of privacy.

1. **Lack of domain expertise:** Deep learning requires a good understanding of the domain and the problem you are trying to solve. If the domain expertise is lacking, it can be difficult to formulate the problem and select the appropriate algorithm.
2. **Unforeseen consequences:** Deep learning models can lead to unintended consequences, for example, a biased model can discriminate against certain groups of people, leading to ethical concerns.
3. **Limited to the data its trained on:** Deep learning models can only make predictions based on the data it has been trained on. They may not be able to generalize to new situations or contexts that were not represented in the training data.
4. **Black box models:** some deep learning models are considered as "black-box" models, as it is difficult to understand how the model is making predictions and identifying the factors that influence the predictions.

Learning representations from data

- Representation learning is a class of machine learning approaches that allow a system to discover the representations required for feature detection or classification from raw data.
- The requirement for manual feature engineering is reduced by allowing a machine to learn the features and apply them to a given activity.
- Basically, Machine learning tasks such as classification frequently demand input that is mathematically and computationally convenient to process, which motivates representation learning.
- Real-world data, such as photos, video, and sensor data, has resisted attempts to define certain qualities algorithmically.
- An approach is to examine the data for such traits or representations rather than depending on explicit techniques.

Methods of Representation Learning

Supervised Learning

- This is referred to as supervised learning when the ML or DL model maps the input X to the output Y. The computer tries to correct itself by comparing model output to ground truth, and the learning process optimizes the mapping from input to output. This process is repeated until the optimization function reaches global minima.
- Even when the optimization function reaches the global minima, new data does not always perform well, resulting in overfitting. While supervised learning does not necessitate a significant amount of data to learn the mapping from input to output, it does necessitate the learned features.
- The prediction accuracy can improve by up to 17 percent when the learned attributes are incorporated into the supervised learning algorithm.
- Using labelled input data, features are learned in supervised feature learning.
- Supervised neural networks, multilayer perceptrons, and (supervised) dictionary learning are some examples.

Unsupervised Learning

- Unsupervised learning is a sort of machine learning in which the labels are ignored in favour of the observation itself. Unsupervised learning isn't used for classification or regression; instead, it's used to uncover underlying patterns, cluster data, denoise it, detect outliers, and decompose data, among other things.
- When working with data x, we must be very careful about whatever features z we use to ensure that the patterns produced are accurate. It has been observed that having more data does not always imply having better representations. We must be careful to develop a model that is both flexible and expressive so that the extracted features can convey critical information.
- Unsupervised feature learning learns features from unlabeled input data by following the methods such as Dictionary learning, independent component analysis, autoencoders, matrix factorization, and various forms of clustering are among examples.

Deep learning works in three figures

- ML about mapping inputs (e.g., images) to targets/outputs (e.g., label "cat").. through observing many examples of inputs and targets/outputs
- Neural networks: do this input-to-target mapping via a deep sequence of simple data transformations (layers) [data transformations are learned by exposure to examples]
- But how does this happen concretely? See Figure 1.

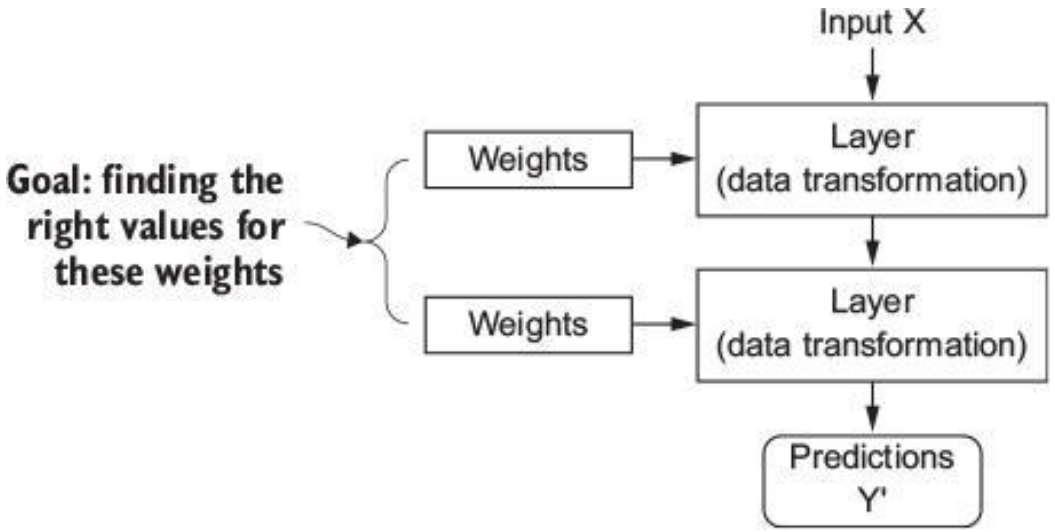


Figure 1 A neural network is parameterized by its weights (Chollet & Allaire, 2018, Fig. 1.7)

- Specification of what a layer does to its input data is stored in the layer's weights (= a bunch of numbers) (see See Figure 1)
  - Weights are also sometimes called the parameters of a layer
- Learning (in this context): Find set of values for the weights of all layers in a network, such that the network will correctly map example inputs to their associated targets
  - Deep neural network can contain tens of millions of parameters
- Finding correct values for all of them may seem like a daunting task…

- Loss function (also called objective function) To control output of neural network, you need to be able to measure how far this output is from what you expected
- Loss function takes predictions of the network and the true target (what you wanted the network to output) and computes a distance score, capturing how well the network has done on this specific example (see Figure 2).

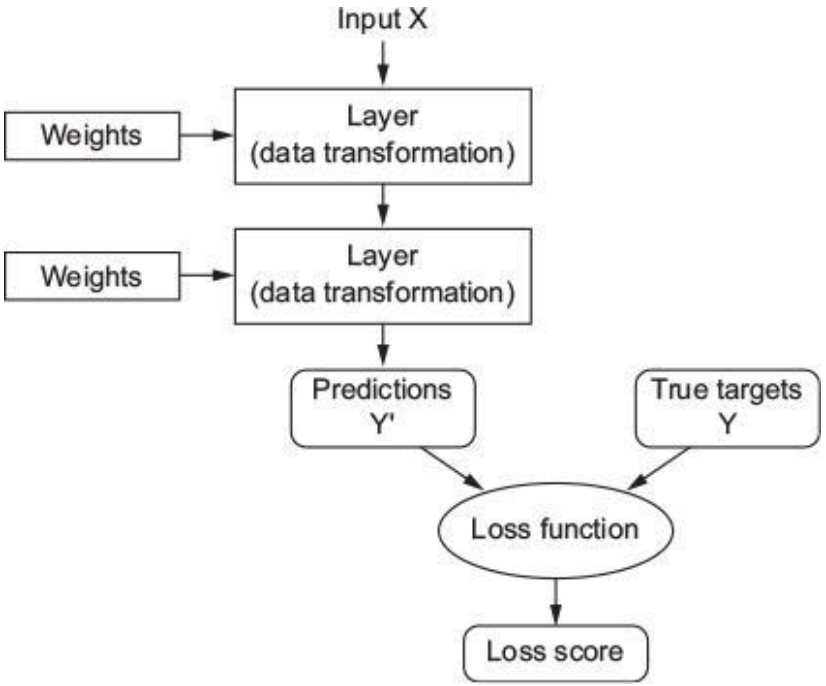


Figure 2: A loss function measures the quality of the network's output (Chollet & Allaire, 2018, Fig. 2)

- Loss score: Is used as feedback signal to adjust the value of the weights, in a direction that will lower loss score for the current example (see Figure 3)
  - Adjustment = job of optimizer
- Optimizer: Implements Backpropagation algorithm (central algorithm in deep learning)
  - Starting point: weights of the network are assigned random values and network implements series of random transformations
    - Output far from targets and loss score is high
  - Network processes examples and adjusts weights a little in correct direction → with every example loss score decreases (= training loop)
    - Training loop: repeated a sufficient number of times, yields weight values that minimize the loss function
      - Typically tens of iterations over thousands of examples
- Trained network: Network with minimal loss for which the outputs are as close as they can be to the targets

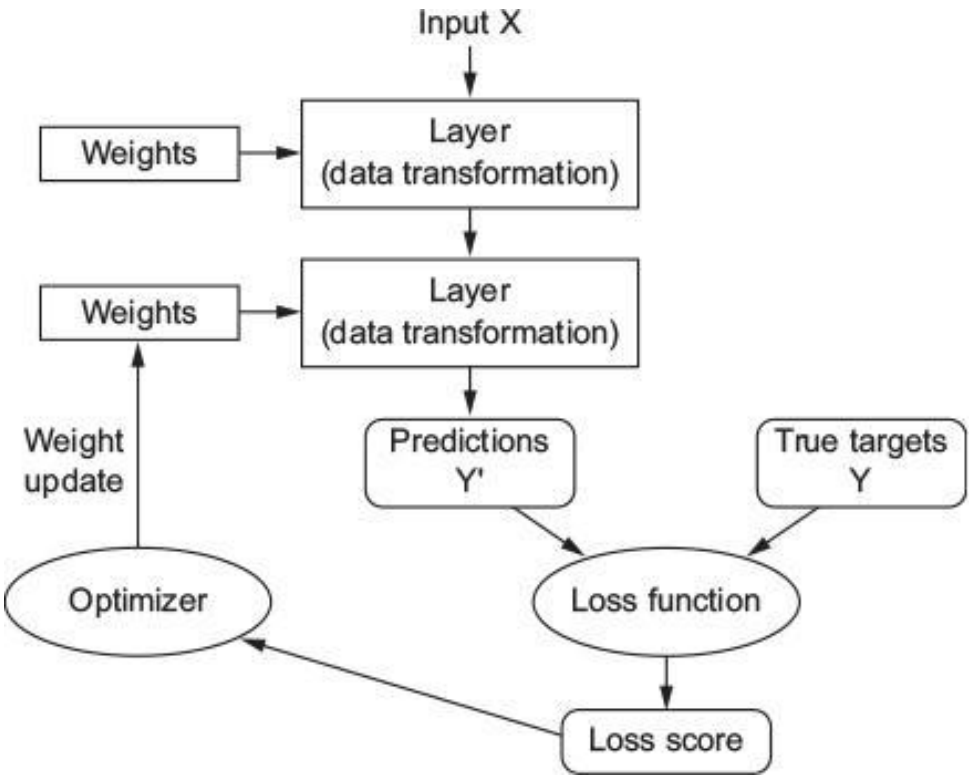
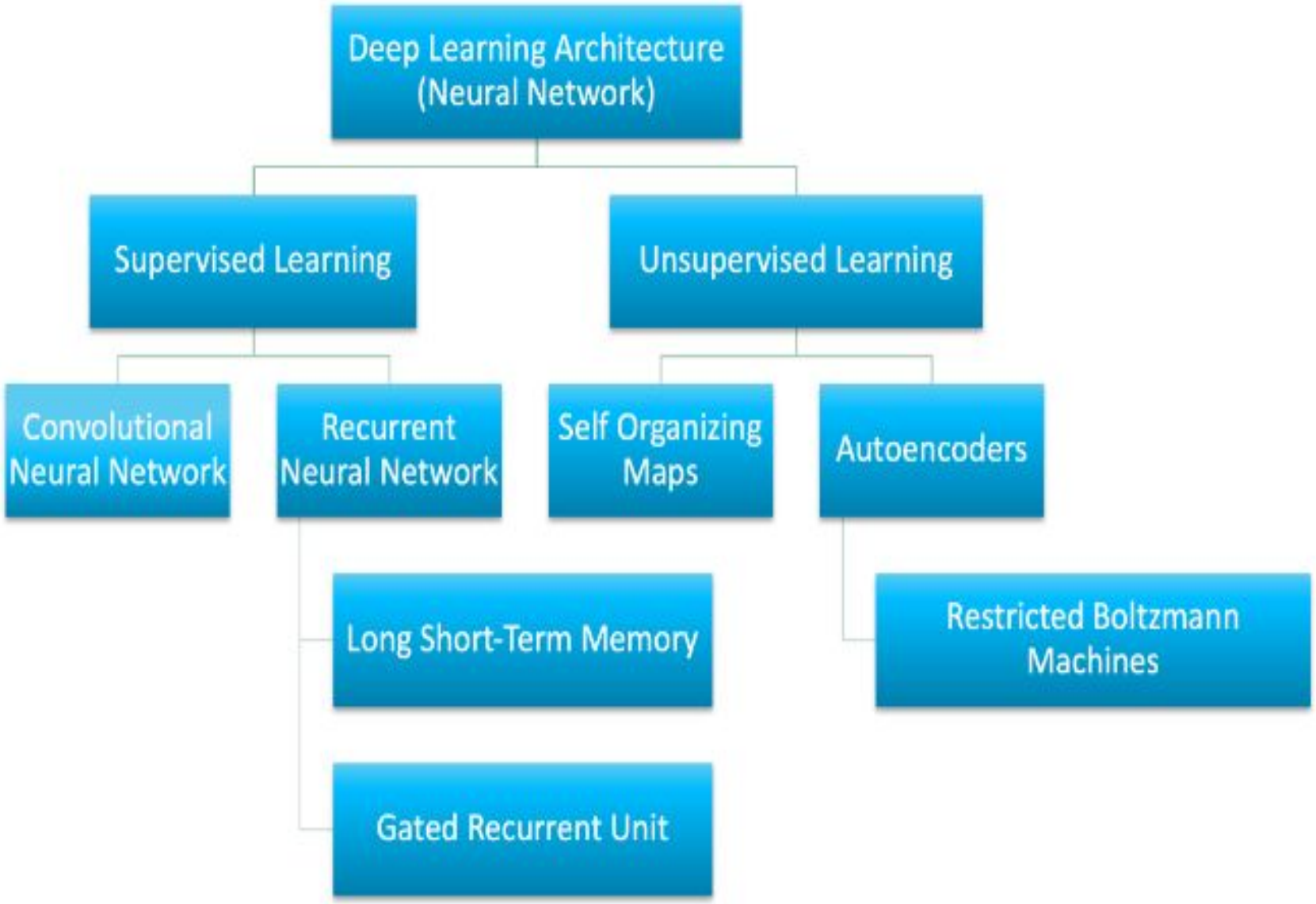


Figure 3: The loss score is used as a feedback signal to adjust the weights (Chollet & Allaire, 2018, Fig. 1.9)

Deep Learning Architecture

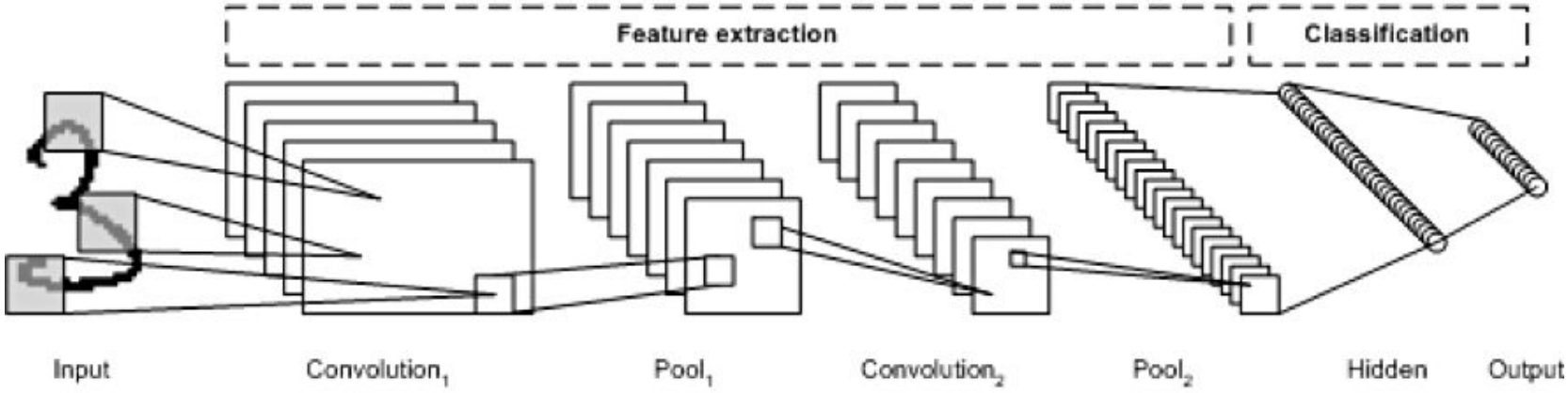
- The number of architectures and algorithms that are used in deep learning is wide and varied.
- Notably, long short-term memory (LSTM) and convolutional neural networks (CNNs) are two of the oldest approaches in this list but also two of the most used in various applications.
- Artificial neural network (ANN) is the underlying architecture behind deep learning. Based on ANN, several variations of the algorithms have been invented.

Deep Learning Architecture



Convolutional neural networks

- A CNN is a multilayer neural network that was biologically inspired by the animal visual cortex. The architecture is particularly useful in image-processing applications. The first CNN was created by Yann LeCun; at the time, the architecture focused on handwritten character recognition, such as postal code interpretation. As a deep network, early layers recognize features (such as edges), and later layers recombine these features into higher-level attributes of the input.
- The LeNet CNN architecture is made up of several layers that implement feature extraction and then classification (see the following image). The image is divided into receptive fields that feed into a convolutional layer, which then extracts features from the input image.
- The next step is pooling, which reduces the dimensionality of the extracted features (through down-sampling) while retaining the most important information (typically, through max pooling).
- Another convolution and pooling step is then performed that feeds into a fully connected multilayer perceptron. The final output layer of this network is a set of nodes that identify features of the image (in this case, a node per identified number). You train the network by using back-propagation.



The use of deep layers of processing, convolutions, pooling, and a fully connected classification layer opened the door to various new applications of deep learning neural networks. In addition to image processing, the CNN has been successfully applied to video recognition and various tasks within natural language processing.

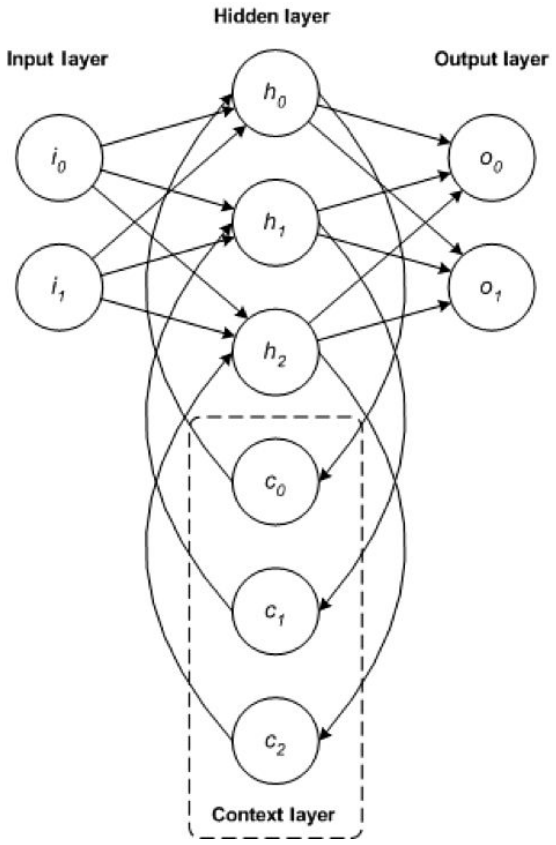
Example applications: Image recognition, video analysis, and natural language processing

Recurrent neural networks

- The RNN is one of the foundational network architectures from which other deep learning architectures are built.
- The primary difference between a typical multilayer network and a recurrent network is that rather than completely feed-forward connections, a recurrent network might have connections that feed back into prior layers (or into the same layer).
- This feedback allows RNNs to maintain memory of past inputs and model problems in time.
- RNNs consist of a rich set of architectures (we'll look at one popular topology called LSTM next).
- The key differentiator is feedback within the network, which could manifest itself from a hidden layer, the output layer, or some combination thereof.

RNN Model

- RNNs can be unfolded in time and trained with standard back-propagation or by using a variant of back-propagation that is called back-propagation in time (BPTT).
- *Example applications: Speech recognition and handwriting recognition*



Advantages:

- RNN can process inputs of any length.
- An RNN model is modeled to remember each information throughout the time which is very helpful in any time series predictor.
- Even if the input size is larger, the model size does not increase.
- The weights can be shared across the time steps.
- RNN can use their internal memory for processing the arbitrary series of inputs which is not the case with feedforward neural networks.

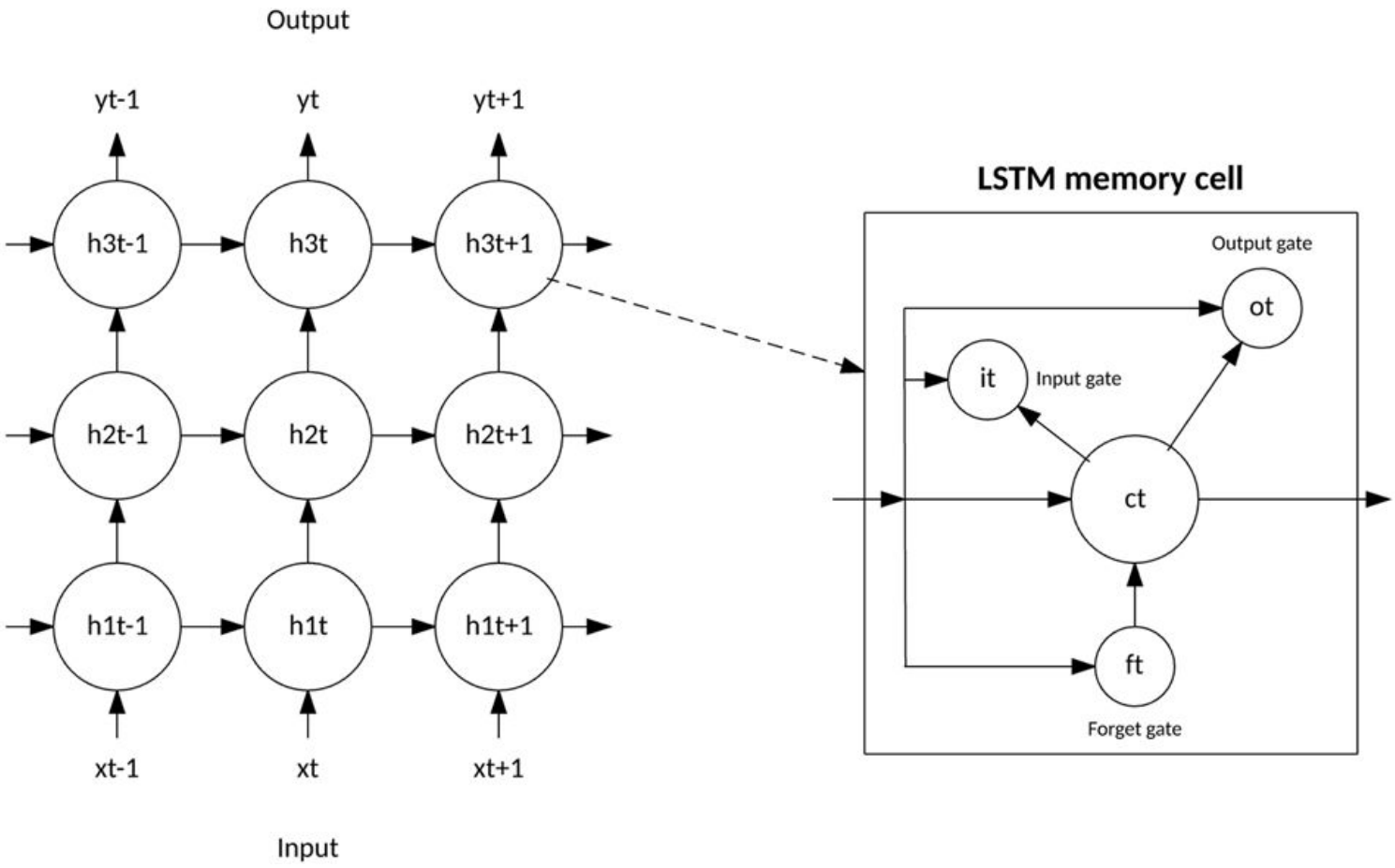
Disadvantages:

- Due to its recurrent nature, the computation is slow.
- Training of RNN models can be difficult.
- If we are using relu or tanh as activation functions, it becomes very difficult to process sequences that are very long.
- Prone to problems such as exploding and gradient vanishing.

LSTM networks

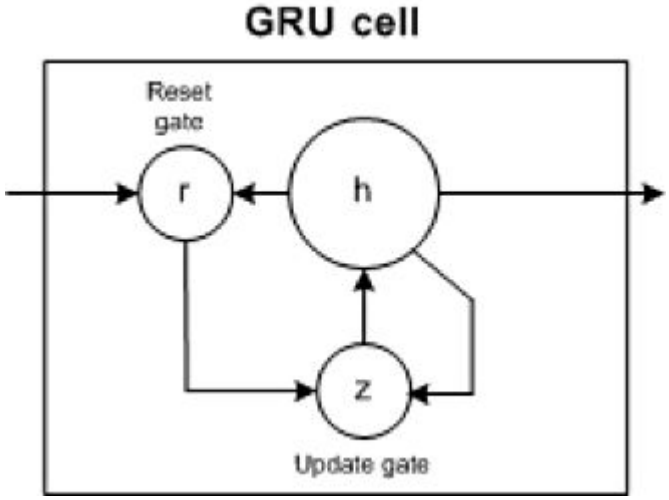
- The LSTM was created in 1997 by Hochreiter and Schimdhuber, but it has grown in popularity in recent years as an RNN architecture for various applications.
- You'll find LSTMs in products that you use every day, such as smartphones. IBM applied LSTMs in IBM Watson® for milestone-setting conversational speech recognition.
- The LSTM departed from typical neuron-based neural network architectures and instead introduced the concept of a memory cell.
- The memory cell can retain its value for a short or long time as a function of its inputs, which allows the cell to remember what's important and not just its last computed value.

- The LSTM memory cell contains three gates that control how information flows into or out of the cell. The input gate controls when new information can flow into the memory.
- The forget gate controls when an existing piece of information is forgotten, allowing the cell to remember new data.
- Finally, the output gate controls when the information that is contained in the cell is used in the output from the cell. The cell also contains weights, which control each gate.
- The training algorithm, commonly BPTT, optimizes these weights based on the resulting network output error.
- Recent applications of CNNs and LSTMs produced image and video captioning systems in which an image or video is captioned in natural language.
- The CNN implements the image or video processing, and the LSTM is trained to convert the CNN output into natural language.
- *Example applications: Image and video captioning systems*


Output
yt-1
yt
yt+1
h3t-1
h3t
h3t+1
h2t-1
h2t
h2t+1
h1t-1
h1t
h1t+1
xt-1
xt
xt+1
Input
LSTM memory cell
Output gate
ot
it
Input gate
ct
ft
Forget gate

GRU networks

- In 2014, a simplification of the LSTM was introduced called the gated recurrent unit. This model has two gates, getting rid of the output gate present in the LSTM model.
- These gates are an update gate and a reset gate. The update gate indicates how much of the previous cell contents to maintain.
- The reset gate defines how to incorporate the new input with the previous cell contents. A GRU can model a standard RNN simply by setting the reset gate to 1 and the update gate to 0.
- The GRU is simpler than the LSTM, can be trained more quickly, and can be more efficient in its execution. However, the LSTM can be more expressive and with more data can lead to better results.
- *Example applications: Natural language text compression, handwriting recognition, speech recognition, gesture recognition, image captioning*

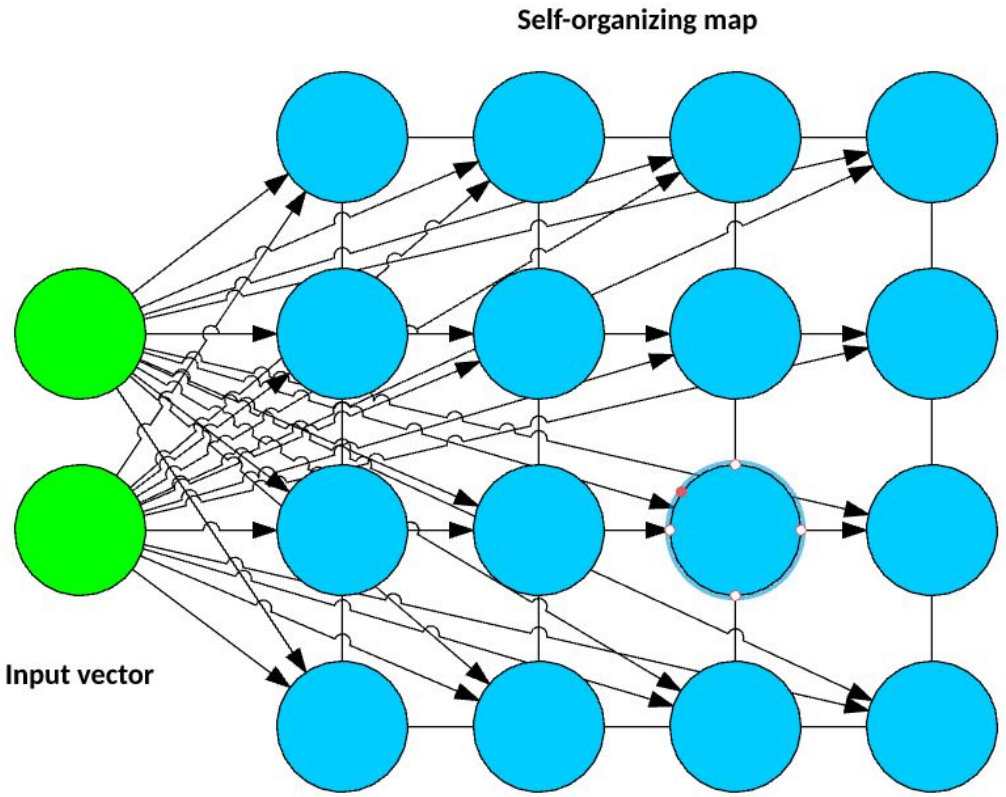

GRU cell
Reset
gate
r
h
z
Update gate

Unsupervised deep learning

Unsupervised learning refers to the problem space wherein there is no target label within the data that is used for training.

Self-organized maps

- Self-organized map (SOM) was invented by Dr. Teuvo Kohonen in 1982 and was popularly known as the Kohonen map.
- SOM is an unsupervised neural network that creates clusters of the input data set by reducing the dimensionality of the input.
- SOMs vary from the traditional artificial neural network in quite a few ways.

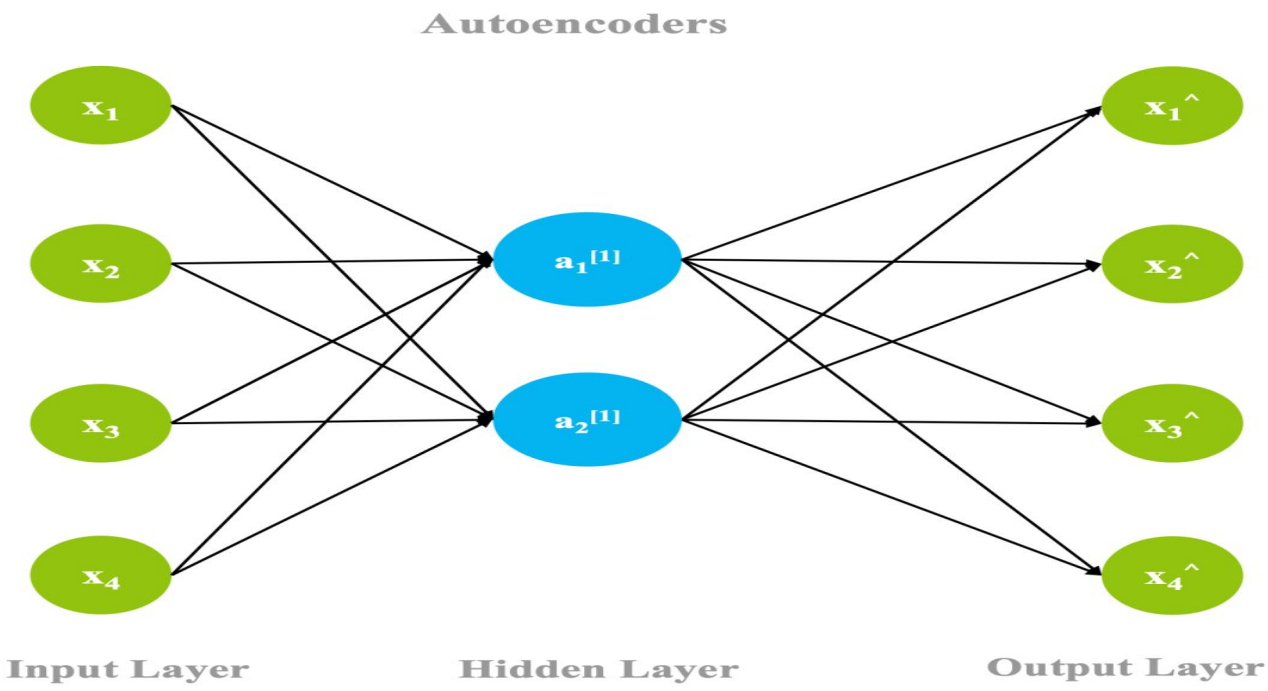


- The first significant variation is that weights serve as a characteristic of the node. After the inputs are normalized, a random input is first chosen. Random weights close to zero are initialized to each feature of the input record. These weights now represent the input node.
- Several combinations of these random weights represent variations of the input node. The euclidean distance between each of these output nodes with the input node is calculated.
- The node with the least distance is declared as the most accurate representation of the input and is marked as the *best matching unit* or *BMU*.
- With these BMUs as center points, other units are similarly calculated and assigned to the cluster that it is the distance from. Radius of points around BMU weights are updated based on proximity. Radius is shrunk.
- Next, in an SOM, no activation function is applied, and because there are no target labels to compare against there is no concept of calculating error and back propogation.
- *Example applications: Dimensionality reduction, clustering high-dimensional inputs to 2-dimensional output, radiant grade result, and cluster visualization*

Autoencoders

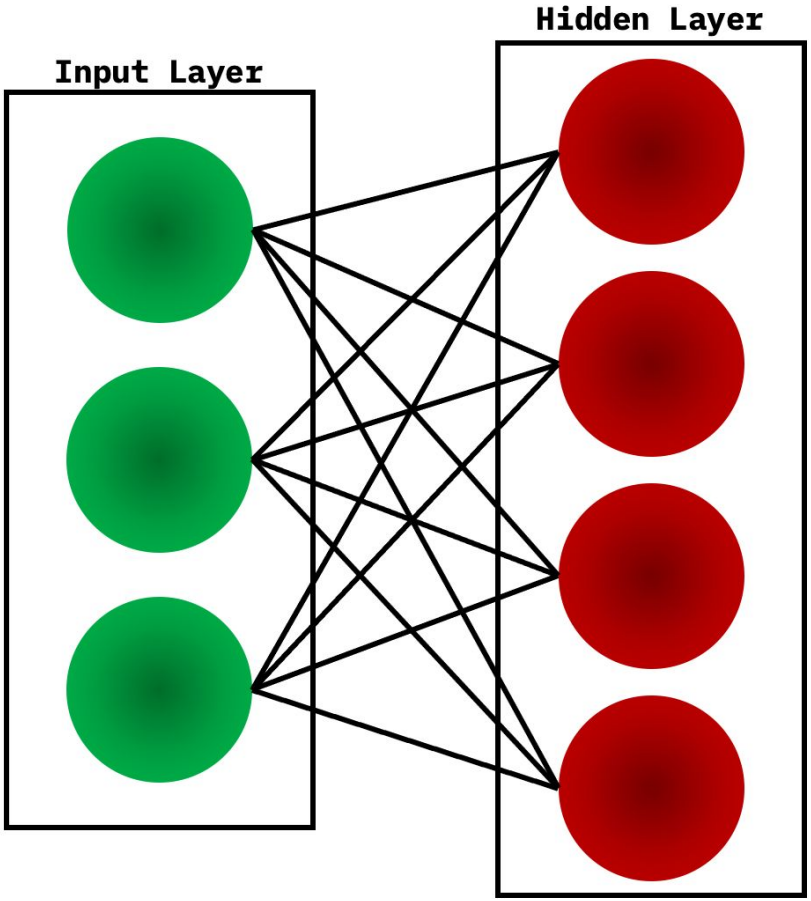
- Though the history of when autoencoders were invented is hazy, the first known usage of autoencoders was found to be by LeCun in 1987.
- This variant of an *ANN* is composed of 3 layers: input, hidden, and output layers.
- First, the input layer is encoded into the hidden layer using an appropriate encoding function.
- The number of nodes in the hidden layer is much less than the number of nodes in the input layer.
- This hidden layer contains the compressed representation of the original input. The output layer aims to reconstruct the input layer by using a decoder function.

- During the training phase, the difference between the input and the output layer is calculated using an error function, and the weights are adjusted to minimize the error.
- Unlike traditional unsupervised learning techniques, where there is no data to compare the outputs against, autoencoders learn continuosly using backward propagation. For this reason, autoencoders are classified as *self supervised* algorithms.
- *Example applications: Dimensionality reduction, data interpolation, and data compression/decompression*



Restricted Boltzmann Machines

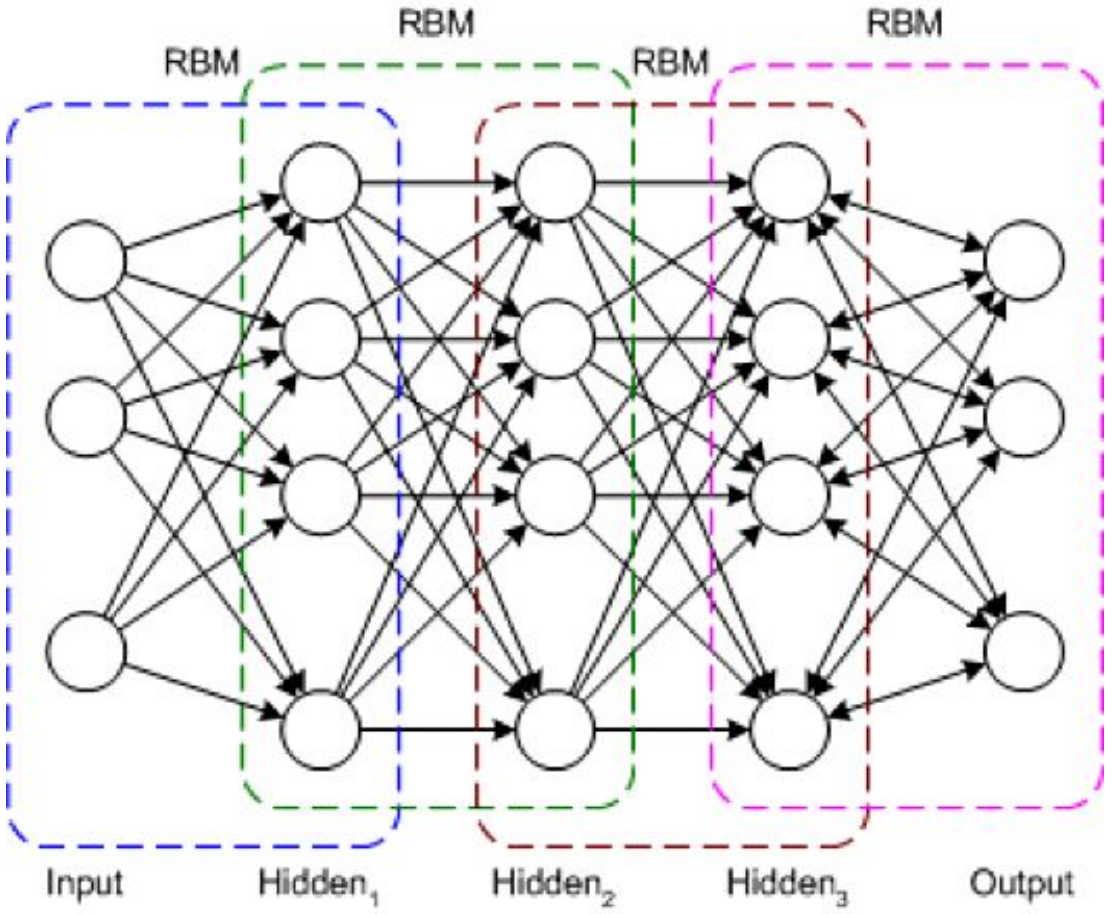
- Though RBMs became popular much later, they were originally invented by Paul Smolensky in 1986 and was known as a *Harmonium*.
- An RBM is a 2-layered neural network. The layers are input and hidden layers. As shown in the following figure, in RBMs every node in a hidden layer is connected to every node in a visible layer.
- In a traditional Boltzmann Machine, nodes within the input and hidden layer are also connected.
- Due to computational complexity, nodes within a layer are not connected in a *Restricted* Boltzmann Machine


Hidden Layer
Input Layer

- During the training phase, RBMs calculate the probabilty distribution of the training set using a stochastic approach. When the training begins, each neuron gets activated at random.
- Also, the model contains respective hidden and visible bias. While the hidden bias is used in the forward pass to build the activation, the visible bias helps in reconstructing the input.
- Because in an RBM the reconstructed input is always different from the original input, they are also known as *generative models*.
- Also, because of the built-in randomness, the same predictions result in different outputs. In fact, this is the most significant difference from an autoencoder, which is a deterministic model.
- *Example applications: Dimensionality reduction and collaborative filtering*

Deep belief networks

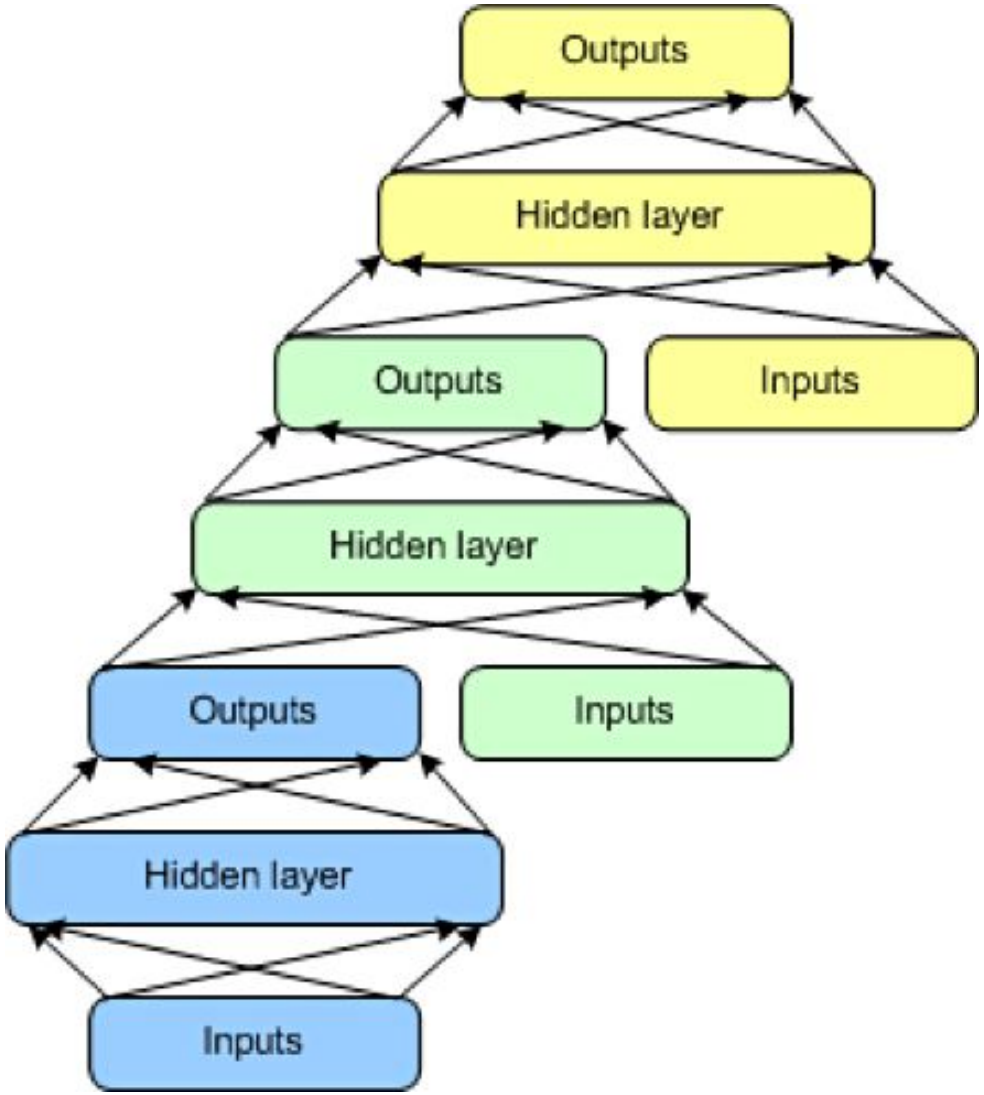
- The DBN is a typical network architecture, but includes a novel training algorithm.
- The DBN is a multilayer network (typically deep and including many hidden layers) in which each pair of connected layers is an RBM. In this way, a DBN is represented as a stack of RBMs.
- In the DBN, the input layer represents the raw sensory inputs, and each hidden layer learns abstract representations of this input.
- The output layer, which is treated somewhat differently than the other layers, implements the network classification.
- Training occurs in two steps: unsupervised pretraining and supervised fine-tuning.


RBM
RBM
RBM
RBM
Input
Hidden1
Hidden2
Hidden3
Output

- In unsupervised pretraining, each RBM is trained to reconstruct its input (for example, the first RBM reconstructs the input layer to the first hidden layer).
- The next RBM is trained similarly, but the first hidden layer is treated as the input (or visible) layer, and the RBM is trained by using the outputs of the first hidden layer as the inputs.
- This process continues until each layer is pretrained. When the pretraining is complete, fine-tuning begins. In this phase, the output nodes are applied labels to give them meaning (what they represent in the context of the network). Full network training is then applied by using either gradient descent learning or back-propagation to complete the training process.
- *Example applications: Image recognition, information retrieval, natural language understanding, and failure prediction*

Deep stacking networks

- The final architecture is the DSN, also called a deep convex network. A DSN is different from traditional deep learning frameworks in that although it consists of a deep network, it's actually a deep set of individual networks, each with its own hidden layers.
- This architecture is a response to one of the problems with deep learning, the complexity of training. Each layer in a deep learning architecture exponentially increases the complexity of training, so the DSN views training not as a single problem but as a set of individual training problems.
- The DSN consists of a set of modules, each of which is a subnetwork in the overall hierarchy of the DSN. In one instance of this architecture, three modules are created for the DSN. Each module consists of an input layer, a single hidden layer, and an output layer. Modules are stacked one on top of another, where the inputs of a module consist of the prior layer outputs and the original input vector.
- This layering allows the overall network to learn more complex classification than would be possible given a single module.


Outputs
Hidden layer
Outputs
Inputs
Hidden layer
Outputs
Inputs
Hidden layer
Inputs

- The DSN permits training of individual modules in isolation, making it efficient given the ability to train in parallel. Supervised training is implemented as back-propagation for each module rather than back-propagation over the entire network.
- For many problems, DSNs can perform better than typical DBNs, making them a popular and efficient network architecture.
- *Example applications: Information retrieval and continuous speech recognition*

Application of Deep Learning:

- Deep learning has a wide range of applications across multiple industries and fields. Some of the most common applications include:

1. **Computer vision:** Deep learning is used in image and video recognition, object detection, semantic segmentation, and other computer vision tasks. Applications include self-driving cars, security cameras, and image recognition for mobile devices.
2. **Natural language processing:** Deep learning is used in natural language understanding, machine translation, sentiment analysis, and other natural language processing tasks. Applications include chatbots, virtual assistants, and language-based search engines.
3. **Speech recognition:** Deep learning is used in speech recognition, voice identification, and voice synthesis. Applications include voice-controlled assistants, voice-enabled devices and voice-controlled robots.
4. **Predictive analytics:** Deep learning is used to analyze historical data and make predictions about future events. Applications include fraud detection, customer churn prediction, and demand forecasting.
5. **Recommender systems:** Deep learning is used to analyze patterns in data to recommend items to users. Applications include movie and music recommendations, news recommendations, and product recommendations.

6. Healthcare: Deep learning is used to analyze medical images and patient data, to improve diagnosis and treatment, and to identify potential health risks. Applications include cancer diagnosis, drug discovery, and personalized medicine.
7. Finance: Deep learning is used to detect fraudulent transactions, to identify potential risks, and to make predictions about stock prices. Applications include credit fraud detection, algorithmic trading, and risk management.
8. Marketing: Deep learning is used to analyze customer data, to predict customer behavior and to personalize marketing campaigns. Applications include customer segmentation, customer lifetime value prediction, and personalization
9. Gaming: Deep learning is used to train agents to play games, and to develop intelligent game-playing algorithms. Applications include game bots, game-playing AI, and adaptive game design.
10. Robotics: Deep learning is used to enable robots to learn from experience and adapt to their environment. Applications include autonomous vehicles, drones, and industrial robots.

Deep learning frameworks

Deep learning has gained immense popularity recently, and the various deep learning architectures make the field all the more widespread. To support the implementation of each of these architectures for different use cases, several frameworks are available.

While each of these frameworks comes with its pros and cons, picking the right deep learning framework based on your individual workload is an essential first step every developer, deep learning practitioner, or data scientist must take.

TensorFlow

- In the context of machine learning, *tensor* refers to the multidimensional array used in the mathematical models that describe neural networks. In other words, a tensor is usually a higher-dimension generalization of a matrix or a vector.

- Through a simple notation that uses a rank to show the number of dimensions, tensors allow the representation of complex `_n_-dimensional` vectors and hyper-shapes as `_n_-dimensional` arrays. Tensors have two properties: a datatype and a shape.
- TensorFlow is an open source deep learning framework that was released in late 2015 under the Apache 2.0 license. Since then, it has become one of the most widely adopted deep learning frameworks in the world (going by the number of GitHub projects based on it).
- TensorFlow traces its origins from Google DistBelief, a proprietary production deep learning system developed by the Google Brain project. Google designed TensorFlow from the ground up for distributed processing and to run optimally on Google's custom application-specific integrated circuit (ASIC) called the Tensor Processing Unit (TPU) in its production data centers. This design makes TensorFlow efficient for deep learning applications.

- Deep learning neural networks typically consist of many layers. They transfer data or perform operations between layers using multidimensional arrays. A tensor flows between the layers of a neural network, thus, the name TensorFlow.
- The main programming language for TensorFlow is Python. C++, the Java® language, and the Go application programming interface (API) are also available without stability promises, as are many third-party bindings for C#, Haskell, Julia, Rust, Ruby, Scala, R, and even PHP.
- Google has a mobile-optimized TensorFlow-Lite library to run TensorFlow applications on Android.

Benefits of TensorFlow

- **Eager execution.** TensorFlow 2 supports eager execution with which operations are evaluated immediately and concrete values are returned, without building graphs. This helps with kick-starting model building and debugging models.
- **Computational graph model.** TensorFlow uses data flow graphs called directed graphs to express computational models. This makes it intuitive for developers who can easily visualize what's going on within the neural network layers by using built-in tools and to perfect their neural network models by adjusting parameters and configurations interactively.
- **Simple-to-use API.** Python developers can use either the TensorFlow raw, low-level API, or core API to develop their own models, or use the higher-level API libraries for built-in models. TensorFlow has many built-in and contributed libraries, and it's possible to overlay a higher-level deep learning framework such as Keras to act as a high-level API. Many of the previous APIs have either been removed or updated to TensorFlow 2.0.
- **Flexible architecture.** A major advantage of using TensorFlow is that it has a modular, extensible, and flexible design. Developers can easily move models across CPU, GPU, or TPU processors with few code changes. Although originally designed for large-scale distributed training and inference, developers also can use TensorFlow to experiment with other machine learning models and system optimization of existing models.

- **Distributed processing.** Google Brain designed TensorFlow from the ground up for distributed processing on its custom ASIC TPU. In addition, TensorFlow can run on multiple NVIDIA GPU cores. Developers can take advantage of the Intel Xeon and Xeon Phi-based x64 CPU architectures or ARM64 CPU architectures. TensorFlow can run on multi-architecture and multicore systems as well as a distributed process that farms out compute-intensive processing as worker tasks. Developers can create clusters of TensorFlow servers and distribute the computational graph across those clusters for training. TensorFlow can perform distributed training either synchronously or asynchronously, both within the graph and between graphs and can share the common data in memory or across networked compute nodes.
- **Performance.** Performance is often a contentious topic, but most developers understand that any deep learning framework depends on the underlying hardware to run optimally to achieve high performance with a low-energy cost. Typically, the native development platform of any framework would achieve the best optimization. TensorFlow performs best on Google TPUs, but it manages to achieve high performance on various platforms-not just servers and desktops but also embedded systems and mobile devices. The framework supports a surprising number of programming languages, as well. Although another framework running natively, such as IBM Watson on the IBM platform, might sometimes outperform TensorFlow, it's still a favorite with developers because artificial intelligence (AI) projects can span platforms and programming languages targeting multiple end applications, all of which need to produce consistent results.

TensorFlow applications

- Google was using its proprietary version of TensorFlow for text and voice search, language translation, and image search applications, the major strengths of TensorFlow are in classification and inference. For example, Google implemented RankBrain, the engine that ranks Google search results, in TensorFlow.
- TensorFlow can be used to improve speech recognition and speech synthesis by differentiating multiple voices or filtering speech in high-ambient-noise environments, mimicking voice patterns for more natural-sounding text to speech.
- Additionally, it handles sentence structure in different languages to produce better translations. It can be used for image and video recognition as well as classification of objects, landmarks, people, sentiments, or activities. This has resulted in major improvements in image and video search.
- Because of its flexible, extensible, and modular design, TensorFlow doesn't limit developers to specific models or applications. Developers have used TensorFlow to implement not only machine learning and deep learning algorithms but also statistical and general computational models.

Keras

- Keras is a Python-based deep learning library that is different from other deep learning frameworks. Keras functions as a high-level API specification for neural networks. It can serve both as a user interface and to extend the capabilities of other deep learning framework back ends that it runs on.
- Keras started as a simplified front end for the academically popular Theano framework. Since then, the Keras API has become a part of Google TensorFlow. Keras officially supports Microsoft Cognitive Toolkit (CNTK), Deeplearning4J, and Apache MXNet.

- Keras is an open source Python package released under the Massachusetts Institute of Technology (MIT) license, with François Chollet, Google, Microsoft, and other contributors holding some of the software's copyrights.

- A Keras front end enables rapid prototyping of neural network models in research. The API is easy to learn and use and has the added advantage of easily porting models between frameworks.
- Because Keras is self-contained, you can use it without having to interact with the back-end framework on which it's running. Keras has its own graph data structures for defining computational graphs: It doesn't rely on the graph data structures of the underlying back-end framework. This approach frees you from having to learn to program the back-end framework, which is why Google chose to add the Keras API to its TensorFlow core.
- However, with Keras 2.4.0, Keras has discontinued multi-backend Keras and focuses exclusively on TensorFlow.

Benefits of Keras

- **Better user experience (UX) for deep learning applications.** The Keras API is user friendly. The API is well-designed, object oriented, and flexible, which makes for a better user experience. Researchers can define new deep learning models without needing to work with potentially complex back ends, resulting in simpler and leaner code.
- **Seamless Python integration.** Keras is a native Python package, which allows easy access to the entire Python data science ecosystem. For example, the Python scikit-learn API can also use Keras models. Developers familiar with back ends such as TensorFlow can use Python to extend Keras, as well.
- **Large, portable body of work and strong knowledge base.** Researchers have used Keras with the Theano back end for some time now. The result is a large body of work and a strong community knowledge base, which deep learning developers can easily port from Theano back ends to TensorFlow back ends. Even the weights are portable between back ends, which means that pretrained models can swap back ends easily with just a few tweaks. Keras and Theano research stays relevant for TensorFlow and other back ends. In addition, Keras makes many learning resources, documentation, and code samples freely available.

PyTorch

- PyTorch is an open source Python package released under the modified Berkeley Software Distribution license. Facebook, the Idiap Research Institute, New York University (NYU), and NEC Labs America hold the copyrights for PyTorch. Although Python is the language of choice for data science, PyTorch is a relative newcomer to the deep learning arena.
- Neural network algorithms typically compute peaks or troughs of a loss function, with most using a gradient descent function to do so. In Torch, PyTorch's predecessor, the Torch Autograd package, contributed by Twitter, computes the gradient functions. Torch Autograd is based on Python Autograd.
- The primary use case for PyTorch is research. Facebook uses PyTorch for innovative research and switches to Caffe2 for production. A format called Open Neural Network Exchange allows users to convert models between PyTorch and Caffe2 and reduces the lag time between research and production.

- Python packages such as Autograd and Chainer both use a technique known as tape-based auto-differentiation to calculate gradients. As such, those packages heavily influenced and inspired the design of PyTorch.
- Tape-based auto-differentiation works just like a tape recorder in that it records operations performed, and then replays them to compute gradients—a method also known as reverse-mode auto-differentiation. PyTorch Autograd has one of the fastest implementations of this function.
- Using this feature, PyTorch users can tweak their neural networks in an arbitrary manner without overhead or lag penalties. As a result, unlike in most well-known frameworks, PyTorch users can dynamically build graphs, with the framework's speed and flexibility facilitating the research and development of new deep learning algorithms.

Benefits of PyTorch

- **Dynamic computational graphs.** Most deep learning frameworks that use computational graphs generate and analyze graphs before runtime. In contrast, PyTorch builds graphs at runtime by using reverse-mode auto-differentiation. Therefore, arbitrary changes to a model do not add runtime lag or overhead to rebuild the model. PyTorch has one of the fastest implementations of reverse-mode auto-differentiation. Apart from being easier to debug, dynamic graphs allow PyTorch to handle variable-length inputs and outputs, which is especially useful in natural language processing for text and speech.
- **Lean back end.** Rather than using a single back end, PyTorch uses separate back ends for CPU and GPU and for distinct functional features. For example, the tensor back end for CPU is TH, while the tensor back end for GPU is THC. Similarly, the neural network back ends are THNN and THCUNN for CPU and GPU, respectively. Individual back ends result in lean code that is tightly focused for a specific task running on a specific class of processor with high memory efficiency. Use of separate back ends makes it easier to deploy PyTorch on constrained systems, such as those used in embedded applications.

- **Python-first approach.** Although it is a derivative of Torch, PyTorch is a native Python package by design. It does not function as a Python language binding but rather as an integral part of Python. PyTorch builds all its functions as Python classes. Hence, PyTorch code can seamlessly integrate with Python functions and other Python packages.
- **Imperative programming style.** Because a direct change to the program state triggers computation, code execution isn't deferred and produces simple code, avoiding many asynchronous executions that could cloud how the code executes. When manipulating data structures, this style is intuitive and easy to debug.
- **Highly extensible.** Users can program using C/C++ by using an extension API that is based on cFFI for Python and compiled for CPU or with CUDA for GPU operation. This feature allows the extension of PyTorch for new and experimental uses and makes it attractive for research use. For example, the PyTorch audio extension allows the loading of audio files.

PyTorch projects, including ones related to:

- Chatbots
- Machine translation
- Text search
- Text to speech
- Image and video classification

Caffe

- Another popular deep learning framework is Caffe. Caffe was originally developed as part of a Ph.D. dissertation but is now released under the Berkeley Software Distribution license. Caffe supports a wide range of deep learning architectures, including CNN and LSTM, but notably does not support RBMs or DBMs (although, Caffe2 will include such support).
- Caffe has been used for image classification and other vision applications, and it supports GPU-based acceleration with the NVIDIA CUDA Deep Neural Network library. Caffe supports Open Multi-Processing (OpenMP) for parallelizing deep learning algorithms over a cluster of systems. Caffe and Caffe2 are written in C++ for performance and offer a Python and MATLAB interface for deep learning training and execution.

Shogun

- Shogun is an open source machine learning software library built in C++. It offers a wide range of efficient and unified machine learning algorithms. The heart of Shogun lies in kernel machines such as support vector machines for regression and classification problems. Read on to understand more about this library.
- Shogun offers a full implementation of Hidden Markov models. Its core is written in C++ and it offers interfaces for MATLAB, Octave, Python, R, Java, Lua, Ruby and C#. Today, there is a large and active user community all over the world that uses Shogun as a base for education and research, and also contributes to the core package.
- Shogun was developed in 1999 by Soeren Sonnenburg and Gunnar Raetsch. The inventive focus was on large-scale kernel methods and bioinformatics. Since then, it has continuously been used for scientific research and has expanded massively. In early 2017, Shogun joined NumFocus.

Features

- Accessible: Shogun is not limited to any single language; you can use the toolkit through a unified interface (SWIG) from C++, Python, Octave, R, Java, Lua, C#, Ruby, etc. Thus Shogun is independent of trends in computing languages and importantly, this allows us to use it as a vehicle to expose one's algorithms to multiple communities. Shogun runs natively under Linux/UNIX, MacOS and Windows.
- Anybody can attempt and learn Shogun with minimal effort, simply by connecting to a Jupyter Notebook (an open source Web application that allows us to create and share documents that contain live code, equations or other things). Users can run Shogun in the cloud. It also offers this service for free through external donations.
- Shogun does not reinvent anything, but provides bindings to other sophisticated libraries like Tapkee, SLEP, GPML, LibSVM/LibLinear, SVMLight, LibOCAS, libqp, VowpalWabbit, and more.
- Free (open source): Shogun is community-based and non-commercial. It is currently released under the GPLv3. It is also moving towards BSD compatibility with optional GPL (General Public License) parts. This means companies can use the code without having to turn their code-base to GPL.

Benefits of using Shogun

- Shogun offers well-organised implementation of all standard ML algorithms. Its executions are competitive as measured by the MLPack benchmarking framework.
- Shogun offers complete modern/advance algorithms (like efficient SVM implementations, multiple kernel learning, etc). All of these are assisted by a collection of general-purpose methods for pre-processing, serialisation, I/O, evaluation and parameter tuning.
- Shogun has a substantial testing infrastructure, making it dependable in dozens of OS setups.