SYLLABUS

Deep Learning - 414443

Credit Scheme :	Examination Scheme :
03 Credits	Mid_Semester : 30 Marks End_Semester : 70 Marks

Unit I Fundamentals of Deep Learning

What is Deep Learning?, Multilayer Perceptron ,Feed forward neural, Back propagation, Gradient descent, Vanishing gradient problem, **Activation Functions :** RELU, LRELU, ERELU, Optimization Algorithms, **Hyper parameters :** Layer size, Magnitude (momentum, learning rate),Regularization (dropout, drop connect, L1, L2) **(Chapter - 1)**

Unit II Convolutional Neural Network

Introduction to CNN, Convolution Operation, Parameter Sharing, Equivariant Representation, Pooling, Variants of the Basic Convolution Function, The basic Architecture of CNN, Popular CNN Architecture – AlexNet. **(Chapter - 2)**

Unit III Recurrent Neural Networks

Recurrent Neural Networks: Types of Recurrent Neural Networks, Feed-Forward Neural Networks vs Recurrent Neural Networks, Long Short-Term Memory Networks (LSTM), Encoder Decoder architectures, Recursive Neural Networks. **(Chapter - 3)**

Unit IV Autoencoders

Undercomplete Autoencoders, Regulraized Autoencoders-Sparse Autoencoders, Stochastic Encoders and Decoders, Denoising Autoencoders, Contractive Autoencoders, Applications of Autoencoders. **(Chapter - 4)**

Unit V Representation Learning

Greedy Layerwise Pre-training, Transfer Learning and Domain Adaption, Distributed Representation, Variants of CNN: DenseNet. **(Chapter - 5)**

Unit VI Applications of Deep Learning

Overview of Deep Learning Applications : Image Classification, Social N/w/ analysis, Speech Recognition, Recommender system, Natural Language Processing. **(Chapter - 6)**

TABLE OF CONTENTS

Unit I

Unit II

Unit III

Unit IV

Unit V

Unit VI

Unit I

1 Fundamentals of Deep Learning

Syllabus

What is Deep Learning ?, Multilayer Perceptron ,Feed forward neural, Back propagation, Gradient descent, Vanishing gradient problem, ***Activation Functions :*** *RELU, LRELU, ERELU, Optimization Algorithms,* ***Hyper parameters :*** *Layer size, Magnitude (momentum, learning rate),Regularization (dropout, drop connect, L1, L2)*

Contents

1.1 What is Deep Learning ?

- The term "deep" usually refers to the number of hidden layers in the neural network.
- Deep learning is a subset of machine learning, which is predicated on idea of learning from example. In machine learning, instead of teaching a computer a massive list of rules to solve the problem, we give it a model with which it can evaluate examples, and a small set of instructions to modify the model when it makes a mistake.
- The basic idea of deep learning is that repeated composition of functions can often reduce the requirements on the number of base functions (computational units) by a factor that is exponentially related to the number of layers in the network.
- Deep learning eliminates some of data pre-processing that is typically involved with machine learning.
- Fig. 1.1.1 shows relation between AI, ML and Deep learning.

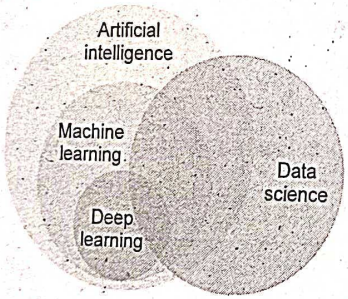


Fig. 1.1.1 Relation between AI, ML and Deep learning

- For example, let's say that we had a set of photos of different pets, and we wanted to categorize by "cat" and "dog". Deep learning algorithms can determine which features (e.g. ears) are most important to distinguish each animal from another. In machine learning, this hierarchy of features is established manually by a human expert.
- In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

- Deep learning classifies information through layers of neural networks, which have a set of inputs that receive raw data. For example, if a neural network is trained with images of birds, it can be used to recognize images of birds. More layers enable more precise results, such as distinguishing a crow from a raven as compared to distinguishing a crow from a chicken.
- Deep Learning consists of the following methods and their variations:
  a) Unsupervised learning systems such as Boltzman machines for preliminary training, Auto-encoders, Generative adversarial network.
  b) Supervised learning such as Convolution neural networks which brought technoogy of pattern recognition to a new level.
  c) Recurrent neural networks, allowing to train on processes in time.
  d) Recursive neural networks, allowing to include feedback between circuit elements and chains.

1.1.1 Reasons for Using Deep Learning

1. **Analyzing unstructured data :** Deep learning algorithms can be trained to look at text data by analyzing social media posts, news, and surveys to provide valuable business and customer insights.
2. **Data labelling :** Deep learning requires labeled data for training. Once trained, it can label new data and identify different types of data on its own.
3. **Feature engineering :** A deep learning algorithm can save time because it does not require humans to extract features manually from raw data.
4. **Efficiency :** When a deep learning algorithm is properly trained, it can perform thousands of tasks over and over again, faster than humans.
5. **Training :** The neural networks used in deep learning have the ability to be applied to many different data types and applications. Additionally, a deep learning model can adapt by retraining it with new data.

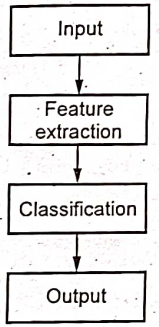
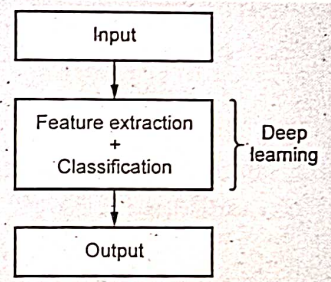
1.1.2 Application of Deep Learning

1. **Aerospace and defense :** Deep learning is utilized extensively to help satellites identify specific objects or areas of interest and classify them as safe or unsafe for soldiers.
2. **Financial services :** Financial institutions regularly use predictive analytics to drive algorithmic trading of stocks, assess business risks for loan approvals, detect fraud, and help manage credit and investment portfolios for clients.

3. **Medical research :** The medical research field uses deep learning extensively. For example, in ongoing cancer research, deep learning is used to detect the presence of cancer cells automatically.
4. **Industrial automation :** The heavy machinery sector is one that requires a large number of safety measures. Deep learning helps with the improvement of worker safety in such environments by detecting any person or objects that comes within the unsafe radius of a heavy machine.
5. **Facial recognition :** This feature utilizing deep learning is being used not just for a range of security purposes but will soon enable purchases at stores. Facial recognition is already being extensively used in airports to enable seamless, paperless check-ins.

1.1.3 Difference between Machine Learning and Deep Learning

Sr. No.	Machine Learning	Deep Learning
1.	Machine learning uses algorithms to parse data, learn from that data, and make informed decisions based on what it has learned.	Deep learning structures algorithms in layers to create an "artificial neural network" that can learn and make intelligent decisions on its own.
2.	Machine learning gives lesser accuracy.	Deep learning gives more accuracy.
3.	Machine learning requires less time for training.	Deep learning requires more time for training.
4.	Needs accurately identified features by human intervention.	It can create new features.
5.	Machine learning models mostly require data in a structured form.	Deep Learning models can work with structured and unstructured data both as they rely on the layers of the Artificial neural network.
6.	Algorithms are detected by data analysts to examine specific variables in data sets.	Algorithms are largely self-directed on data analysis once they are put into production.

7.	Machine learning can work on low-end machines.	Deep learning model needs a huge amount of data to work efficiently, so they need GPU's and hence the high-end machine
8.		

1.1.4 Difference between ML, AI and Data Science

Sr. No.	Machine Learning	Artificial Intelligence	Data Science
1.	Focuses on providing a means for algorithms and systems to learn from experience with data and use that experience to improve over time.	Focuses on giving machines cognitive and intellectual capabilities similar to those of humans.	Focuses on extracting information needles from data haystacks to aid in decision-making and planning.
2.	Machine Learning uses statistical models.	Artificial Intelligence uses logic and decision trees.	Data Science deals with structured data.
3.	A form of analytics in which software programs learn about data and find patterns.	Development of computerized applications that simulate human intelligence and interaction.	The process of using advanced analytics to extract relevant information from data.
4.	Objective is to maximize accuracy.	Objective is to maximize the chance of success.	Objective is to extract actionable insights from the data.

5.	ML can be done through supervised, unsupervised or reinforcement learning approaches.	AI encompasses a collection of intelligence concepts, including elements of perception, planning and prediction.	Uses statistics, data mathematics, wrangling, big data analytics, machine learning and various other methods to answer analytics questions.
6.	ML is concerned with knowledge accumulation.	AI is concerned with knowledge dissemination and conscious machine actions.	Data science is all about data engineering.

1.1.5 Difference between AI, ML and Deep Learning

Sr. No.	AI	ML	DL
1.	AI aims towards building machines that are capable to think like humans.	ML aims to learn through data to solve the problem.	DL aims to build neural networks that automatically discover patterns for feature detection.
2.	AI is subset of data science	ML is subset of AI and data science	DL is subset of AI, ML and data science
3.	All systems of artificial intelligence fall into three types: a) Artificial Narrow Intelligence b) Artificial General Intelligence c) Artificial Super Intelligence	ML algorithms can be broadly classified into three categories a) Supervised learning b) Unsupervised learning c) Reinforcement learning	Deep learning architectures are as follows: a) Convolutional Neural Networks b) Recurrent Neural Networks c) Recursive Neural Networks

4.	Making machines intelligent may or may not need high computational power.	These algorithms can work easily on normal low performance computers without GPUs.	Algorithms are dependent on high performance hardware components that include GPUs.

1.1.6 Advantages and Disadvantages of Deep Learning

Advantages of Deep Learning

- No need for feature engineering.
- DL solves the problem on the end-to-end basis.
- Deep learning gives more accuracy.

Disadvantages of Deep Learning

- DL needs high-performance hardware.
- DL needs much more time to train.
- It is very difficult to assess its performance in real world applications.
- It is very hard to understand.

1.2 Multilayer Perceptron

- The perceptron is very useful for classifying data sets that are linearly separable.
- The Multilayer Perceptron (MLP) model features multiple layers that are interconnected in such a way that they form a feed-forward neural network. Each neuron in one layer has directed connections to the neurons of a separate layer.
- It consists of three types of layers: the input layer, output layer and hidden layer.
- Fig. 1.2.1 shows multilayer perceptron model.

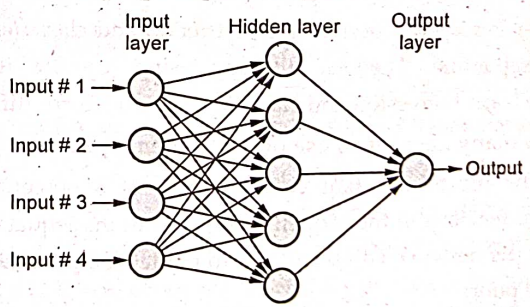


Fig. 1.2.1 Multilayer perceptron model

- The **input layer** receives the input signal to be processed. The input layer distributes the values to each of the neurons in the hidden layer. In addition to the predictor variables, there is a constant input of 1.0, called the bias that is fed to each of the hidden layers; the bias is multiplied by a weight and added to the sum going into the neuron.
- **Hidden layer :** Arriving at a neuron in the hidden layer, the value from each input neuron is multiplied by a weight and the resulting weighted values are added together producing a combined value. The weighted sum is fed into a transfer function, which outputs a value. The outputs from the hidden layer are distributed to the output layer.
- **Output layer :** Arriving at a neuron in the output layer, the value from each hidden layer neuron is multiplied by a weight, and the resulting weighted values are added together producing a combined value. The weighted sum is fed into a transfer function, which outputs a value. The output values are the outputs of the network.
- The required task such as prediction and classification is performed by the output layer. An arbitrary number of hidden layers that are placed in between the input and output layer are the true computational engine of the MLP.
- The neurons in the MLP are trained with the back propagation learning algorithm. MLPs are designed to approximate any continuous function and can solve problems which are not linearly separable. The major use cases of MLP are pattern classification, recognition, prediction and approximation.
- The perceptron is very useful for classifying data sets that are linearly separable. They encounter serious limitations with data sets that do not conform to this pattern as discovered with the XOR problem. The XOR problem shows that for any classification of four points that there exists a set that are not linearly separable.
- The MultiLayer Perceptron breaks this restriction and classifies datasets which are not linearly separable. They do this by using a more robust and complex architecture to learn regression and classification models for difficult datasets.
- **Deciding how many neurons to use in the hidden layers :**
  a) One of the most important characteristics of a perceptron network is the number of neurons in the hidden layer(s). If an inadequate number of neurons are used, the network will be unable to model complex data, and the resulting fit will be poor.
  b) If too many neurons are used, the training time may become excessively long, and, worse, the network may over fit the data. When overfitting occurs, the

network will begin to model random noise in the data. The result is that the model fits the training data extremely well, but it generalizes poorly to new, unseen data. Validation must be used to test for this.

1.3 Feed Forward Neural

- Feed Forward Neural Network is an artificial neural network in which the connections between nodes does not form a cycle. The feed forward model is the simplest form of neural network as information is only processed in one direction. While the data may pass through multiple hidden nodes, it always moves in one direction and never backwards.
- They are called feed forward because information only travels forward in the network (no loops), first through the input nodes, then through the hidden nodes (if present) and finally through the output nodes.
- Feed-forward networks tends to be simple networks that associates inputs with outputs. It can be used in pattern recognition. This type of organization is represented as bottom-up or top-down.
- Fig. 1.3.1 shows basic structure of a Feed Forward (FF) Neural Network

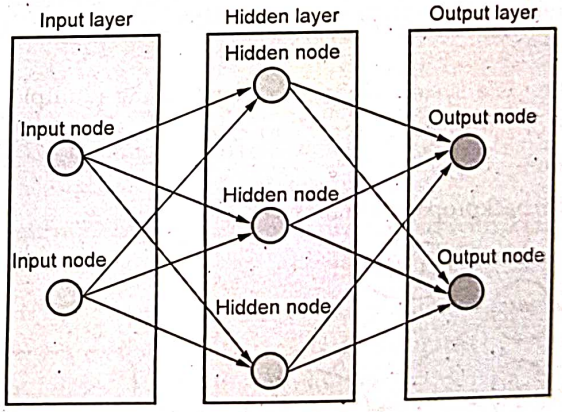


Fig. 1.3.1 Basic structure of a Feed Forward Neural Network

- Input layer contains one or more input nodes. For example, suppose we want to predict whether it will rain tomorrow and base our decision on two variables, humidity and wind speed. In that case, our first input would be the value for humidity, and the second input would be the value for wind speed.
- Hidden layer : This layer contains an activation function.
- Output layer contains one or more output nodes.

- Feed foward neural networks are primarily used for supervised learning in cases where the data to be learned is neither sequential nor time-dependent.
- Feed-forward networks have the following characteristics :
  1. Perceptron's are arranged in layers, with the first layer taking in inputs and the last layer producing outputs. The middle layers have no connection with the external world, and hence are called hidden layers.
  2. Each perceptron in one layer is connected to every perceptron on the next layer. Hence information is constantly "fed forward" from one layer to the next and this explains why these networks are called feed-forward networks.
  3. There is no connection among perceptron's in the same layer.

1.4 Back Propagation

- Backpropagation is a training method used for a multi-layer neural network. It is also called the generalized delta rule. It is a gradient descent method which minimizes the total squared error of the output computed by the net.
- The backpropagation algorithm looks for the minimum value of the error function in weight space using a technique called the delta rule or gradient descent. The weights that minimize the error function is then considered to be a solution to the learning problem.
- Backpropagation is a systematic method for training multiple layer ANN. It is a generalization of Widrow-Hoff error correction rule. 80 % of ANN applications uses backpropagation.
- Fig. 1.4.1 shows backpropagation network.

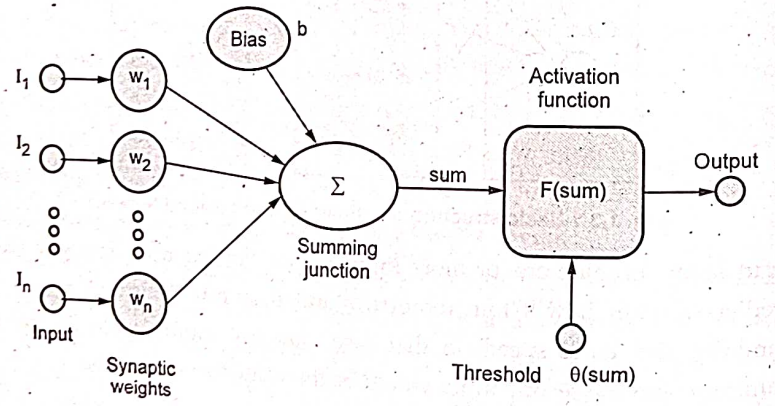


Fig. 1.4.1 Backpropagation network

- Consider a simple neuron :
  a. Neuron has a summing junction and activation function.
  b. Any non linear function which differentiable everywhere and increases everywhere with sum can be used as activation function.
  c. Examples : Logistic function, Arc tangent function, Hyperbolic tangent activation function.
- These activation function makes the multilayer network to have greater representational power than single layer network only when non-linearity is introduced.
- **Need of hidden layers :**
  1. A network with only two layers (input and output) can only represent the input with whatever representation already exists in the input data.
  2. If the data is discontinuous or non-linearly separable, the innate representation is inconsistent, and the mapping cannot be learned using two layers (Input and Output).
  3. Therefore, hidden layer(s) are used between input and output layers.
- **Weights** connects unit (neuron) in one layer only to those in the next higher layer. The output of the unit is scaled by the value of the connecting weight, and it is fed forward to provide a portion of the activation for the units in the next higher layer.
- Backpropagation can be applied to an artificial neural network with any number of hidden layers. The training objective is to adjust the weights so that the application of a set of inputs produces the desired outputs.
- **Training procedure :** The network is usually trained with a large number of input-output pairs.
  1. Generate weights randomly to small random values (both positive and negative) to ensure that the network is not saturated by large values of weights.
  2. Choose a training pair from the training set.
  3. Apply the input vector to network input.
  4. Calculate the network output.
  5. Calculate the error, the difference between the network output and the desired output.
  6. Adjust the weights of the network in a way that minimizes this error.
  7. Repeat steps 2 - 6 for each pair of input-output in the training set until the error for the entire system is acceptably low.

Forward pass and backward pass :

- Backpropagation neural network training involves two passes.
  1. In the forward pass, the input signals moves forward from the network input to the output.
  2. In the backward pass, the calculated error signals propagate backward through the network, where they are used to adjust the weights.
  3. In the forward pass, the calculation of the output is carried out, layer by layer, in the forward direction. The output of one layer is the input to the next layer.
- In the reverse pass,
  a. The weights of the output neuron layer are adjusted first since the target value of each output neuron is available to guide the adjustment of the associated weights, using the delta rule.
  b. Next, we adjust the weights of the middle layers. As the middle layer neurons have no target values, it makes the problem complex.
- **Selection of number of hidden units :** The number of hidden units depends on the number of input units.
  1. Never choose h to be more than twice the number of input units.
  2. You can load p patterns of I elements into $\log_2$ p hidden units.
  3. Ensure that we must have at least 1/e times as many training examples.
  4. Feature extraction requires fewer hidden units than inputs.
  5. Learning many examples of disjointed inputs requires more hidden units than inputs.
  6. The number of hidden units required for a classification task increases with the number of classes in the task. Large networks require longer training times.

Factors influencing backpropagation training

- The training time can be reduced by using :
  1. **Bias :** Networks with biases can represent relationships between inputs and outputs more easily than networks without biases. Adding a bias to each neuron is usually desirable to offset the origin of the activation function. The weight of the bias is trainable similar to weight except that the input is always +1.
  2. **Momentum :** The use of momentum enhances the stability of the training process. Momentum is used to keep the training process going in the same general direction analogous to the way that momentum of a moving object

behaves. In backpropagation with momentum, the weight change is a combination of the current gradient and the previous gradient.

1.4.1 Advantages and Disadvantages

Advantages of backpropagation :

1. It is simple, fast and easy to program.
2. Only numbers of the input are tuned and not any other parameter.
3. No need to have prior knowledge about the network.
4. It is flexible.
5. A standard approach and works efficiently.
6. It does not require the user to learn special functions.

Disadvantages of backpropagation :

1. Backpropagation possibly be sensitive to noisy data and irregularity.
2. The performance of this is highly reliant on the input data.
3. Needs excessive time for training.
4. The need for a matrix-based method for backpropagation instead of mini-batch.

1.5 Gradient Descent

- Much of machine learning can be written as an optimization problem.
- Example loss functions : Logistic regression, linear regression, principle component analysis, neural network loss.
- A very efficient way to train logistic models is with **Stochastic Gradient Descent (SGD).**
- One challenge with training on power law data (i.e. most data) is that the terms in the gradient can have very different strengths
- The idea behind stochastic gradient descent is iterating a weight update based on the gradient of loss function :

$$\bar{w}(k+1) = \bar{w}(k) - \gamma \nabla L(\bar{w})$$

- Logistic regression is designed as a **binary classifer** (output say {0, 1}) but actually **outputs the probability** that the input instance is in the "1" class.
- A logistic classifier has the form :

$$p(X) = \frac{1}{1 + \exp(-X\beta)}$$

where $X = (X_1, \ldots, X_n)$ is a vector of features.

- Stochastic gradient has some serious limitations however, especially if the gradients vary widely in magnitude. Some coefficients change very fast, others very slowly.
- This happens for text, user activity and social media data (and other power-law data), because gradient magnitudes scale with feature frequency, i.e. over several orders of magnitude.
- It is not possible to set a single learning rate that trains the frequent and infrequent features at the same time.
- An example of stochastic gradient descent with perceptron loss is shown as follows :

```
from sklearn.linear_model import SGDClassifier
```

1.5.1 Finding the Optimal Hyper-parameters through Grid Search

- In statistics, hyperparameter is a parameter from a prior distribution; it captures the prior belief before data is observed.
- In any machine learning algorithm, these parameters need to be initialized before training a model.
- **Model hyperparameters** *are the properties that govern the entire training process.*
- Hyperparameters are important because they directly control the behaviour of the training algorithm and have a significant impact on the performance of the model is being trained.
- Choosing appropriate hyperparameters plays a crucial role in the success of our neural network architecture. Since it makes a huge impact on the learned model.
- For example, if the learning rate is too low, the model will miss the important patterns in the data. If it is high, it may have collisions.
- Choosing good hyperparameters gives two benefits :
  1. Efficiently search the space of possible hyperparameters.
  2. Easy to manage a large set of experiments for hyperparameter tuning.
- **The process of finding most optimal hyperparameters in machine learning is called hyperparameter optimisation.**
- Grid search is a very traditional technique for implementing hyperparameters. It brute force all combinations. Grid search requires to create two set of hyperparameters.

  1. Learning rate 2. Number of layers

- Grid search trains the algorithm for all combinations by using the two set of hyperparameters and measures the performance using "Cross Validation" technique.
- This validation technique gives assurance that our trained model got most of the patterns from the dataset.
- One of the best methods to do validation by using "K-Fold Cross Validation" which helps to provide ample data for training the model and ample data for validations.
- With this technique, we simply build a model for each possible combination of all of the hyperparameter values provided, evaluating each model and selecting the architecture which produces the best results.
- For example, say you have two continuous parameters α and β, where manually selected values for the parameters are the following :

  $\alpha \in \{0, 1, 2\}$

  $\beta \in \{.25, .50, .75\}$
- Then the pairing of the selected hyperparametric values, H, can take on any of the following :

  $H \in \{(0, .25), (0, .50), (0, .75), (1, .25), (1, .50), (1, .75), (2, .25), (2, .50), (2, .75)\}$
- Grid search will examine each pairing of and to determine the best performing combination. The resulting pairs, *H*, are simply each output that results from taking the Cartesian product of α and β.
- While straightforward, this "brute force" approach for hyperparameter optimization has some drawbacks. Higher-dimensional hyperparametric spaces are far more time consuming to test than the simple two-dimensional problem presented here.
- Also, because there will always be a fixed number of training samples for any given model, the model's predictive power will decrease as the number of dimensions increases. This is known as Hughes phenomenon.

1.5.2 Vanishing Gradient Problem

- When back-propagation is used, the earlier layers will receive very small updates compared to the later layers. This problem is referred to as the vanishing gradient problem.
- The vanishing gradient problem is essentially a situation in which a deep multilayer feed-forward network or a Recurrent Neural Network (RNN) does not have the ability to propagate useful gradient information from the output end of the model back to the layers near the input end of the model.

- Weight initialization is one technique that can be used to solve the vanishing gradient problem. It involves artificially creating an initial value for weights in a neural network to prevent the backpropagation algorithm from assigning weights that are unrealistically small.
- The most important solution to the vanishing gradient problem is a specific type of neural network called Long Short-Term Memory Networks (LSTMs).
- Indication of vanishing gradient problem :
  a) The parameters of the higher layers change to a great extent, while the parameters of lower layers barely change.
  b) The model weights could become 0 during training.
  c) The model learns at a particularly slow pace and the training could stagnate at a very early phase after only a few iterations.
- Some methods that are proposed to overcome the vanishing gradient problem:
  a) Residual neural networks (ResNets)
  b) Multi-level hierarchy
  c) Long short term memory (LSTM)
  d) Faster hardware
  e) ReLU
  f) Batch normalization

1.6 Activation Functions

- Activation functions also known as transfer function is used to map input nodes to output nodes in certain fashion.
- The activation function is the most important factor in a neural network which decided whether or not a neuron will be activated or not and transferred to the next layer.
- Activation functions help in normalizing the output between 0 to 1 or –1 to 1. It helps in the process of backpropagation due to their differentiable property. During backpropagation, loss function gets updated, and activation function helps the gradient descent curves to achieve their local minima.
- Activation function basically decides in any neural network that given input or receiving information is relevant or it is irrelevant.
- These activation function makes the multilayer network to have greater representational power than single layer network only when non-linearity is introduced.

- The input to the activation function is sum which is defined by the following equation.

$$\text{sum} = I_1W_1 + I_2W_2 + \ldots + I_nW_n = \sum_{j=1}^{n} I_jW_j + b$$

- **Activation Function : Logistic Function**

$$f(\text{sum}) = \frac{1}{(1 + e^{-s \times \text{sum}})} = (1 + s^{-s \times \text{sum}})^{-1}$$

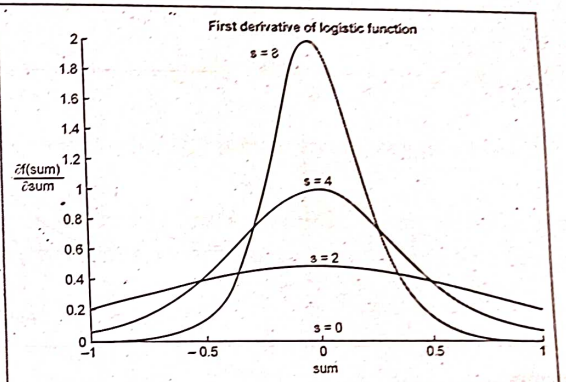


Fig. 1.6.1

- Logistic function monotonically increases from a lower limit (0 or – 1) to an upper limit (+1) as sum increases. In which values vary between 0 and 1, with a value of 0.5 when I is zero.
- **Activation Function : Arc Tangent**

$$f(\text{sum}) = \frac{2}{\pi}\tan^{-1}(s \times \text{sum})$$

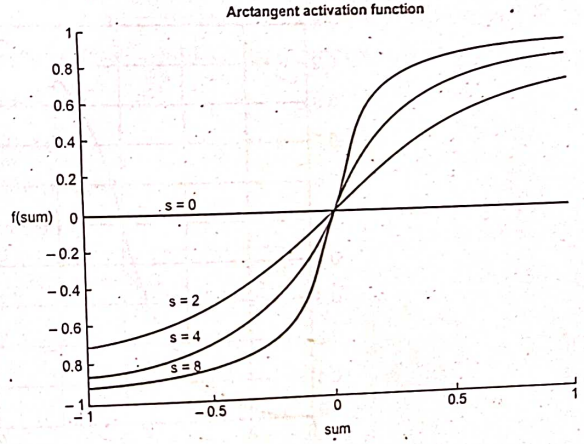


Fig. 1.6.2

- **Activation Function : Hyperbolic Tangent**

$$f(sum) = \tanh(s * I) = \frac{e^{s \times sum} - e^{-s \times sum}}{e^{s \times sum} + e^{-s \times sum}}$$

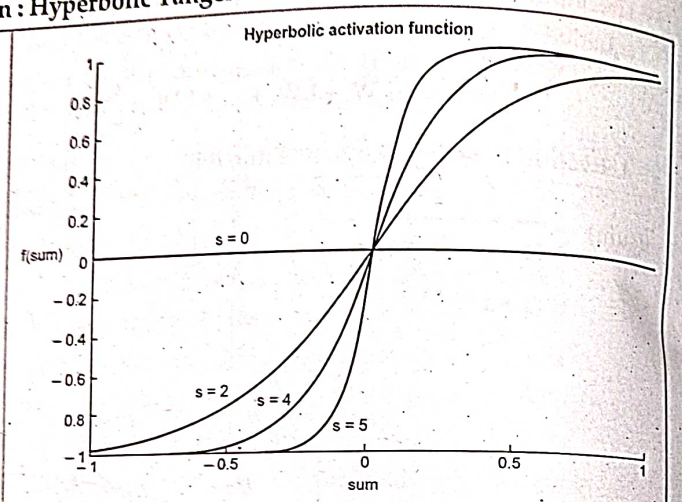


Fig. 1.6.3

1.6.1 Sigmoid

- A sigmoid function produces a curve with an "S" shape. The example sigmoid function shown on the left is a special case of the logistic function, which models the growth of some set.

$$sig(t) = \frac{1}{1 + e^{-t}}$$

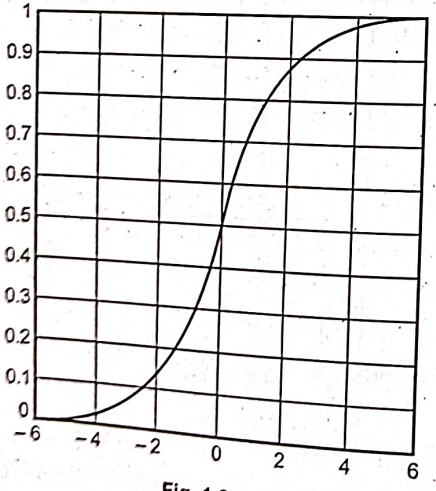


Fig. 1.6.4

- In general, a sigmoid function is real-valued and differentiable, having a non-negative or non-positive first derivative, one local minimum, and one local maximum.
- The logistic sigmoid function is related to the hyperbolic tangent as follows :

$$1 - 2\,sig(x) = 1 - 2\frac{1}{1 + e^{-x}} = -\tanh\frac{x}{2}$$

- Sigmoid functions are often used in artificial neural networks to introduce nonlinearity in the model.
- A neural network element computes a linear combination of its input signals, and applies a sigmoid function to the result.
- A reason for its popularity in neural networks is because the sigmoid function satisfies a property between the derivative and itself such that it is computationally easy to perform.

$$\frac{d}{dt} sig(t) = sig(t)\,(1 - sig(t))$$

- Derivatives of the sigmoid function are usually employed in learning algorithms.

1.6.2 ReLU

- Rectified Linear Unit (ReLU) solve the vanishing gradient problem. ReLU is a non-linear function or piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero.
- It is the most commonly used activation function in neural networks, especially in Convolutional Neural Networks (CNNs) and Multilayer perceptron's.
- Mathematically, it is expressed as

$$f(x) = \max(0, x)$$

where **x : input to neuron**

- **Fig. 1.6.5 shows ReLU function**

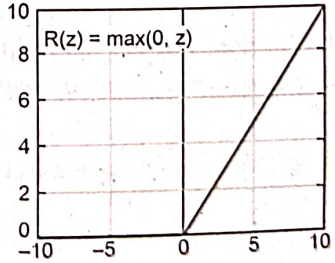


Fig. 1.6.5 ReLU function

- Most of the times, the "continuous function optimization" problem arises in majority of the machine learning algorithms, in which most of the input given are the real numbers, and so are the outputs. These kinds of problems that take only discrete values as input are generally referred to as "Combinatorial Optimization Problems".
- The optimization algorithms state that if more information about the target function can be made available, it becomes easier to optimize that function and that information can also be utilized effectively for further data processing.
- The major point that comes across the execution of an optimization algorithm is to decide whether we can differentiate an objective function at a given point or not.
- That is, can we calculate the first derivative of a function for a given solution or not ?
- Based on this point, the optimization algorithms are further classified into two categories : One that differentiates the function and other that does not.
- Hence, in this section, we shall discuss the "differentiable" and "non differentiable" objective functions that can be used to group multiple optimization algorithms.

1.7.1 Differentiable Objective Function

- If we can calculate the derivative of a function at any given point while input is given to the system such function can be referred to as "Differential Function".
- One can define the derivative of a function as the rate at which the function changes its value at a given point of time. This is often referred to as a slope too.
- One can apply the optimization techniques on these derivative functions using simple calculus.
- Optimization techniques can sound easier if the derivatives of these "continuous functions", as mentioned above, can be calculated. Some of the algorithms that uses these gradient values of the derivatives are as follows :

1. **Bracketing algorithms :**
   - ♦ This technique is helpful when there are problems having only one input variable and the optima exist in the pre - defined specific criteria or range.
   - ♦ These algorithms can easily navigate this range that is known already and locate the optima. The only drawback is that the algorithm assumes that there is only one optima present in the model.

   - ♦ The advantage of using this algorithm is that it can be utilized in a model even if sometimes there is no derivative information about the variables available.
   - ♦ Some of the examples that use bracketing algorithms are Fibonacci search, Golden Section search, Bisection method, etc.
2. **Local descent algorithms :**
   - ♦ These algorithms work for the models in which there are multiple input variables with one global optima.
   - ♦ The algorithm is widely used in the line search problem.
   - ♦ This problem includes the definition of the direction to move during a search space and then it performs the bracketing type search in a line in the direction chosen.
   - ♦ The algorithm executes until no other iteration of finding the improved directions is possible.
   - ♦ These iterations make the algorithm costly as it continues its execution till an effective direction is obtained.
3. **First order algorithms :**
   - ♦ These algorithms use the first order derivatives (gradient) to decide the direction to move in the search space.
   - ♦ This algorithm works by first calculating the first derivative of the function, and then following it in the opposite direction, for example going downhill to minimum value for minimization problems, with the help of step size, also known as "learning rate".
   - ♦ This step size, or learning rate, is a hyperparameter in the algorithm, that decides the distance to cover, or how far to cover in a search space, which is opposite to the normally used local descent algorithms, which do not have this hyperparameter and performs a full line search in every directions specified.
   - ♦ These algorithms are also known as "Gradient Descent" algorithms and following the introduction to some of the minor extensions, these are also known as Momentum, Adagrad, RMSProp, Adam, etc.
   - ♦ These gradient descent algorithms are also helpful in training the artificial neural networks and implementing deep learning models in it, by providing the template for Stochastic Gradient Descent, useful for artificial neural networks.

- Here, the gradient will be based on assumption, instead of direct calculation, using the prediction techniques on the trained data.

4. **Second order algorithms :**
   - These algorithms use the second order derivative of the input variables for choosing the direction of movement in the search space.
   - The algorithms work appropriately only for the objective functions in which the Hessian matrix needs to be calculated.
   - Some of the examples in which the second order algorithms are used.
     - Newton's method
     - Secant method
   - These algorithms are also known as Quasi Newton Methods.

1.7.2 Non - Differentiable Objective Function

- Although, optimization algorithms working on the derivatives of the objective functions are efficient and fast, there are certain objective functions whose derivatives cannot be calculated, the reason being the complexity of the function.
- Some of the reasons for the complexities in the function include.
  - Lack of analysis of the function
  - Multiple optima required.
  - Evaluation of stochastic functions
  - Objective functions are discontinuous.
- The optimization algorithms that do not make the compulsion for the first or second order derivatives for their objective functions are called as Black - box optimization algorithms. Some of these algorithms are :
  - Direct algorithms
  - Stochastic algorithms
  - Population algorithms

Let us have a brief of each of these :

1. **Direct algorithms :**
   - These algorithms are used when the calculation of the derivatives of the objective function is not possible.
   - The algorithms work with an assumption that the objective function contains single optima.

   - These methods are also referred to as "pattern search" algorithms, since they analyze the search space using the geometrical shapes and patterns.
   - The gradient information required to run the algorithm is calculated directly from the objective function by computing the difference between the scores obtained from the points in the search space.
   - This information estimated are then helpful in choosing a direction to commute in the search space and cover the region of the present optima.
   - Some of the examples that use these direct algorithms are Cyclic Coordinate search, Powell's method, Hooke-Jeeves method, etc.

2. **Stochastic algorithms :**
   - For the variables whose derivatives cannot be calculated, stochastic optimization algorithms use the randomness for those objective functions to commute in the search space.
   - Hence, due to the randomness involved, the stochastic algorithms involve many data sampling for the objective function.

3. **Population algorithms :**
   - These algorithms maintain a pool of solutions for a given input, often known as a population of candidate solutions, which are used to explore, sample the optima.
   - These algorithms are mostly used in the problems that are more challenging and also involves the evaluation of functions containing considerable noise in it, in addition to the presence of multiple global optima. The solutions of such algorithms are difficult to be found by other methods.
   - Genetic algorithms, differential evolution, particle swarm optimization, etc. are the examples of population algorithms.

1.8 Hyperparameters

- Hyperparameters are parameters whose values control the learning process and determine the values of model parameters that a learning algorithm ends up learning.
- While designing a machine learning model, one always has multiple choices for the architectural design for the model. This creates a confusion on which design to choose for the model based on its optimality. And due to this, there are always trials for defining a perfect machine learning model.

- The parameters that are used to define these machine learning models are known as the hyperparameters and the rigorous search for these parameters to build an optimized model is known as hyperparameter tuning.
- Hyperparameters are not model parameters, which can be directly trained from data. Model parameters usually specify the way to transform the input into the required output, whereas hyperparameters define the actual structure of the model that gives the required data.

1.8.1 Layer Size

- Layer size is defined by the number of neurons in a given layer. Input and output layers are relatively easy to figure out because they correspond directly to how our modeling problem handles input and ouput.
- For the input layer, this will match up to the number of features in the input vector. For the output layer, this will either be a single output neuron or a number of neurons matching the number of classes we are trying to predict.
- It is obvious that a neural network with 3 layers will give better performance than that of 2 layers. Increasing more than 3 doesn't help that much in neural networks. In the case of CNN, an increasing number of layers makes the model better.

1.8.2 Magnitude : Learning Rate

- The amount that the weights are updated during training is referred to as the step size or the learning rate. Specifically, the learning rate is a configurable hyper-parameter used in the training of neural networks that has a small positive value, often in the range between 0.0 and 1.0.
- For example, if learning rate is 0.1, then the weights in the network are updated 0.1 * (estimated weight error) or 10 % of the estimated weight error each time the weights are updated. The learning rate hyper-parameter controls the rate or speed at which the model learns.
- Learning rates are tricky because they end up being specific to the dataset and even to other hyper-parameters. This creates a lot of overhead for finding the right setting for hyper-parameters.
- Large learning rates () make the model learn faster but at the same time it may cause us to miss the minimum loss function and only reach the surrounding of it. In cases where the learning rate is too large, the optimizer overshoots the minimum and the loss updates will lead to divergent behaviours.

- On the other hand, choosing lower learning rate values gives a better chance of finding the local minima with the trade-off of needing larger number of epochs and more time.
- **Momentum** can accelerate learning on those problems where the high-dimensional weight space that is being navigated by the optimization process has structures that mislead the gradient descent algorithm, such as flat regions or steep curvature.

1.9 Regularization

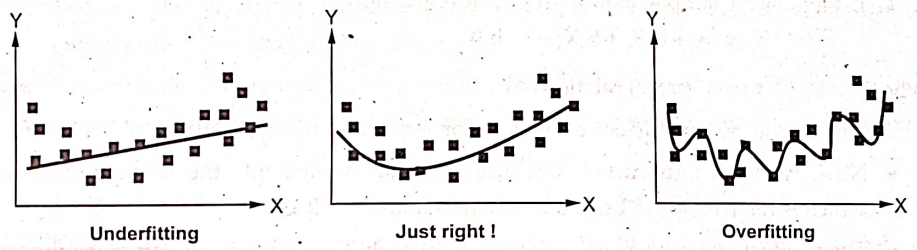


Fig. 1.9.1

- Just have a look at the above figure, and we can immediately predict that once we try to cover every minutest feature of the input data, there can be irregularities in the extracted features, which can introduce noise in the output. This is referred to as "Overfitting".
- This may also happen with the lesser number of features extracted as some of the important details might be missed out. This will leave an effect on the accuracy of the outputs produced. This is referred to as "Underfitting".
- This also shows that the complexity for processing the input elements increases with overfitting. Also, neural networks being a complex interconnection of nodes, the issue of overfitting may arise frequently.
- To eliminate this, regularization is used, in which we have to make the slightest modification in the design of the neural network, and we can get better outcomes.

1.9.1 Regularization in Machine Learning

- One of the most important factors that affect the machine learning model is overfitting.
- The machine learning model may perform poorly if it tries to capture even the noise present in the dataset applied for training the system, which ultimately results in overfitting. In this context, noise doesn't mean the ambiguous or false data, but

those inputs which do not acquire the required features to execute the machine learning model.

- Analyzing these data inputs may surely make the model flexible, but the risk of overfitting will also increase accordingly.
- One of the ways to avoid this is to cross validate the training dataset, and decide accordingly the parameters to include that can increase the efficiency and performance of the model.
- Let this be the simple relation for linear regression :

$$Y \approx b_0 + b_1X_1 + b_2X_2 + \ldots b_pX_p$$

Where Y = Learned relation

β = Co-efficient estimators for different variables and/or predictors (X)

- Now, we shall introduce a loss function, that implements the fitting procedure, which is referred to as "Residual Sum of Squares" or RSS.
- The co-efficient in the function is chosen in such a way that it can minimize the loss function easily.

Hence,

$$RSS = \sum_{i=1}^{n}\left(Y_i - \beta_0 - \sum_{j=1}^{p}\beta_jX_{ij}\right)^2$$

- Above equation will help in adjusting the co-efficient function depending on the training dataset.
- In case noise is present in the training dataset, then the adjusted co-efficient won't be generalized when the future datasets will be introduced. Hence, at this point, regularization comes into picture and makes this adjusted co-efficient shrink towards zero.
- One of the methods to implement this is the ridge regression, also known as L2 regularization. Lets have a quick overview on this.

1.9.2 Ridge Regression (L2 Regularization)

- Ridge regression, also known as L2 regularization, is a technique of regularization to avoid the overfitting in training data set, which introduces a small bias in the training model, through which one can get long term predictions for that input.
- In this method, a penalty term is added to the cost function. This amount of bias altered to the cost function in the model is also known as ridge regression penalty.

- Hence, the equation for the cost function, after introducing the ridge regression penalty is as follows :

$$\sum_{i=1}^{m}(y_i - y'_i)^2 = \sum_{i=1}^{m}\left(Y_i - \sum_{j=1}^{n}\beta_j \times X_{ij}\right)^2 + \lambda\sum_{j=0}^{n}\beta_j^2$$

Here, λ is multiplied by the square of the weight set for the individual feature of the input data. This term is ridge regression penalty.

- It regularizes the co-efficient set for the model and hence the ridge regression term deduces the values of the coefficient, which ultimately helps in deducing the complexity of the machine learning model.
- From the above equation, we can observe that if the value of tends to zero, the last term on the right - hand side will tend to zero, thus making the above equation a representation of a simple linear regression model.
- Hence, lower the value of , the model will tend to linear regression.
- This model is important to execute the neural networks for machine learning, as there would be risks of failure for generalized linear regression models, if there are dependencies found between its variables. Hence, ridge regression is used here.

1.9.3 Lasso Regression (L1 Regularization)

- One more technique to reduce the overfitting, and thus the complexity of the model is the lasso regression.
- Lasso regression stands for Least Absolute and Selection Operator and is also sometimes known as L1 regularization.
- The equation for the lasso regression is almost same as that of the ridge regression, except for a change that the value of the penalty term is taken as the absolute weights.
- The advantage of taking the absolute values is that its slope can shrink to 0, as compared to the ridge regression, where the slope will shrink it near to 0.
- The following equation gives the cost function defined in the Lasso regression :

$$\sum_{i=1}^{m}(y_i - y'_i)^2 = \sum_{i=1}^{m}\left(Y_i - \sum_{j=0}^{n}\beta_j \times X_{ij}\right)^2 + \lambda\sum_{j=0}^{n}|\beta_j|^2$$

- Due to the acceptance of absolute values for the cost function, some of the features of the input dataset can be ignored completely while evaluating the machine learning model, and hence the feature selection and overfitting can be reduced to much extent.

- On the other hand, ridge regression does not ignore any feature in the model and includes it all for model evaluation. The complexity of the model can be reduced using the shrinking of co-efficient in the ridge regression model.

1.9.4 Dropout

- Dropout was introduced by "Hinton et al"and this method is now very popular. It consists of setting to zero the output of each hidden neuron in chosen layer with some probability and is proven to be very effective in reducing overfitting.
- Fig. 1.9.2 shows dropout regulations.

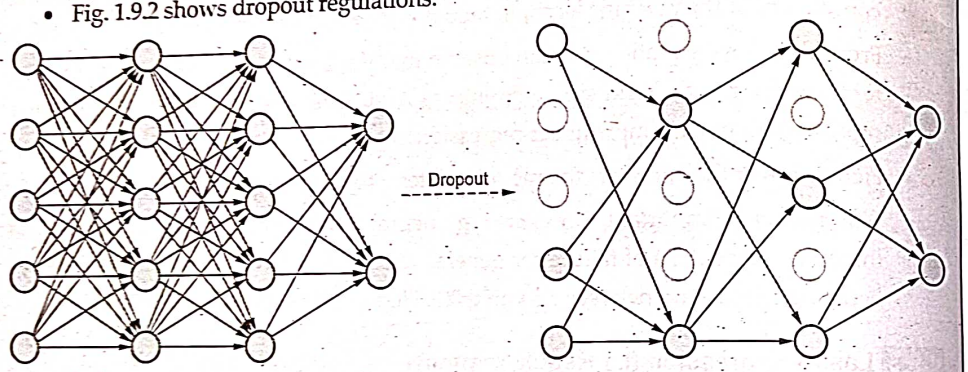


Fig. 1.9.2 Dropout regulation

- To achieve dropout regularization, some neurons in the artificial neural network are randomly disabled. That prevents them from being too dependent on one another as they learn the correlations. Thus, the neurons work more independently, and the artificial neural network learns multiple independent correlations in the data based on different configurations of the neurons.
- It is used to improve the training of neural networks by omitting a hidden unit. It also speeds training.
- Dropout is driven by randomly dropping a neuron so that it will not contribute to the forward pass and back-propagation.
- Dropout is an inexpensive but powerful method of regularizing a broad family of models.

1.9.5 DropConnect

- DropConnect, known as the generalized version of Dropout, is the method used for regularizing deep neural networks. Fig. 1.9.3 shows dropconnect.

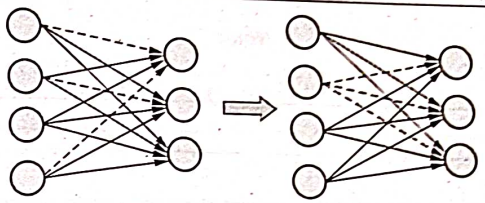


Fig. 1.9.3 Dropconnect

- DropConnect has been proposed to add more noise to the network. The primary difference is that instead of randomly dropping the output of the neurons, we randomly drop the connection between neurons.
- In other words, the fully connected layer with DropConnect becomes a sparsely connected layer in which the connections are chosen at random during the training stage.

❑❑❑

Unit II

2 Convolutional Neural Network

Syllabus

Introduction to CNN, Convolution Operation, Parameter Sharing, Equivariant Representation, Pooling, Variants of the Basic Convolution Function, The basic Architecture of CNN, Popular CNN Architecture – AlexNet.

Contents

2.1 Introduction to Convolutional Neural Network

- Convolutional neural network (CNN) is a deep learning neural network designed for processing structured arrays of data such as images. A CNN is a feed-forward neural network, often with up to 20 or 30 layers. The power of a convolutional neural network comes from a special kind of layer called the convolutional layer.
- Convolutional neural network is also called ConvNet.
- In CNN, 'convolution' is referred to as the mathematical function. It's a type of linear operation in which you can multiply two functions to create a third function that expresses how one function's shape can be changed by the other.
- In simple terms, two images that are represented in the form of two matrices, are multiplied to provide an output that is used to extract information from the image.
- CNN represents the input data in the form of multidimensional arrays. It works well for a large number of labeled data. CNN extract the each and every portion of input image, which is known as receptive field. It assigns weights for each neuron based on the significant role of the receptive field.
- Instead of preprocessing the data to derive features like textures and shapes, a CNN takes just the image's raw pixel data as input and "learns" how to extract these features, and ultimately infer what object they constitute.
- The goal of CNN is to reduce the images so that it would be easier to process without losing features that are valuable for accurate prediction.
- A convolutional neural network is made up of numerous layers, such as convolution layers, pooling layers, and fully connected layers, and it uses a back-propagation algorithm to learn spatial hierarchies of data automatically and adaptively.
- To understand the Concept of Convolutional Neural Networks (CNNs), let us take an example of the images our brain can interpret.
- As soon as we see an image, our brain starts categorizing it based on the color, shape and sometimes also the message that image is conveying. Similar thing can be done through machines even after a rigorous training. But the difficulty is there is a huge difference in what humans interpret and what machine does. For a machine, the image is merely an array of pixels. There is a unique pattern included, the image is merely an array of pixels. There is a unique pattern included in each object present in the image and the computer tries to find out these patterns to get the information about the image.

- Machines can be trained giving tons of images to increase its ability to recognize the objects included in a given input image.
- Most of the digital companies have opted for CNNs for image recognition, some of these include Google, Amazon, Instagram, Interest, Facebook, etc.
- Hence, we define a convolutional neural network as : "A neural network consisting of multiple convolutional layers which are used mainly for image processing, classification, segmentation and other correlated data".

2.1.1 Advantages and Disadvantages of CNN

1. Advantages :

- CNN automatically detects the important features without any human supervision.
- CNN is also computationally efficient
- Higher accuracy
- Weight sharing is another major advantage of CNNs.
- Convolutional neural networks also minimize computation in comparison with a regular neural network.
- CNNs make use of the same knowledge across all image locations.

2. Disadvantages :

- Adversarial attacks are cases of feeding the network 'bad' examples to cause misclassification.
- CNN requires lot of training data.
- CNNs tend to be much slower because of operations like maxpool.

2.1.2 Application of CNN

- CNN is mostly used for image classification, for example to determine the satellite images containing mountains and valleys or recognition of handwriting, etc. image segmentation, signal processing, etc. are the areas where CNN are used.
- **Object detection :** Self-driving cars, AI-powered surveillance systems, and smart homes often use CNN to be able to identify and mark objects. CNN can identify objects on the photos and in real-time, classify, and label them.
- **Voice synthesis :** Google Assistant's voice synthesizer uses Deepmind's WaveNet ConvNet model.
- **Astrophysics :** They are used to make sense of radio telescope data and predict the probable visual image to represent that data.

2.2 Convolution Operation

- Convolution operation focuses on extracting/preserving important features from the input. Convolution operation allows the network to detect horizontal and vertical edges of an image and then based on those edges build high-level features in the following layers of neural network.
- In general form, convolution is an operation on two functions of a real valued argument. To motivate the definition of convolution, we start with examples of two functions we might use.
- Suppose we are tracking the location of a spaceship with a laser sensor. Laser sensor provides a single output x(t), the position of the spaceship at time t. Both "x" and "t" are real-valued, i.e., we can get a different reading from the laser sensor at any instant in time.
- Now suppose that our laser sensor is somewhat noisy. To obtain a less noisy estimate of the spaceship's position, we would like to average together several measurements. Of course, more recent measurements are more relevant, so we will want this to be a weighted average that gives more weight to recent measurements.
- We can do this with a weighting function w(a), where "a" is the age of a measurement. If we apply such a weighted average operation at every moment, we obtain a new function providing a smoothed estimate of the position "s" of the spaceship.
- Convolution operation uses three parameters : Input image, Feature detector and Feature map.
- Convolution operation involves an input matrix and a filter, also known as the kernel. Input matrix can be pixel values of a grayscale image whereas a filter is a relatively small matrix that detects edges by darkening areas of input image where there are transitions from brighter to darker areas. There can be different types of filters depending upon what type of features we want to detect, e.g. vertical, horizontal, or diagonal, etc.
- Input image is converted into binary 1 and 0. The convolution operation, shown in Fig. 2.2.1 is known as the feature detector of a CNN. The input to a convolution can be raw data or a feature map output from another convolution. It is often interpreted as a filter in which the kernel filters input data for certain kinds of information.

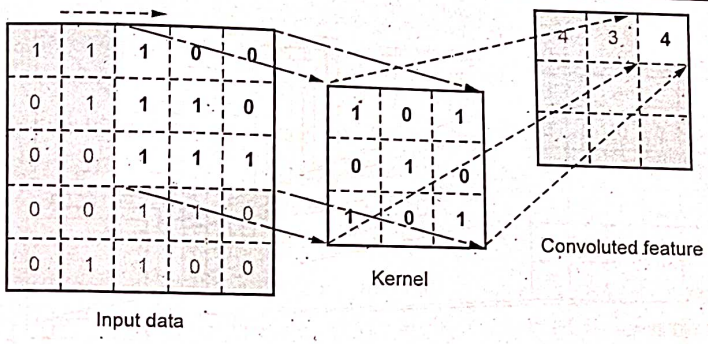


Fig. 2.2.1 Convolution operation

- Sometimes a 5×5 or a 7×7 matrix is used as a feature detector. The feature detector is often referred to as a "kernel" or a "filter,". At each step, the kernel is multiplied by the input data values within its bounds, creating a single entry in the output feature map.
- Generally, an image can be considered as a matrix whose elements are numbers between 0 and 255. The size of image matrix is: *image height*image width*number of image channels*
- A grayscale image has 1 channel, where a colour image has 3 channels.
- Kernel : A kernel is a small matrix of numbers that is used in image convolutions. Differently sized kernels containing different patterns of numbers produce different results under convolution. The size of a kernel is arbitrary but 3x3 is often used. Fig. 2.2.2 shows example of kernel.

0	1	0
1	1	1
0	1	0

Fig. 2.2.2 Example of kernel

- Convolutional layers perform transformations on the input data volume that are a function of the activations in the input volume and the parameters.
- In reality, convolutional neural networks develop multiple feature detectors and use them to develop several feature maps which are referred to as convolutional layers and it is shown in Fig. 2.2.3.

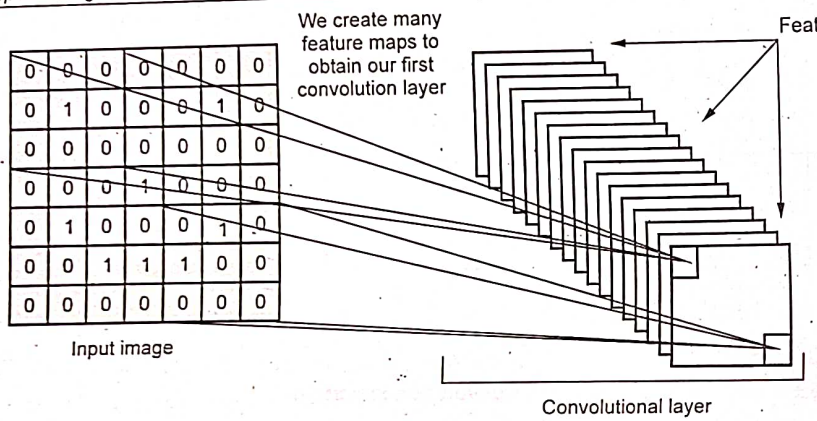


Fig. 2.2.3 Feature detectors

- Through training, the network determines what features it finds important in order for it to be able to scan images and categorize them more accurately.
- Convolutional layers have parameters for the layer and additional hyper-parameters. Gradient descent is used to train the parameters in this layer such that the class scores are consistent with the labels in the training set.
- Components of convolutional layers are as follows :
  a) Filters
  b) Activation maps
  c) Parameter sharing
  d) Layer-specific hyper-parameters
- Filters are a function that has a width and height smaller than the width and height of the input volume. The filters are tensors, and they are used to convolve the input tensor when the tensor is passed to the layer instance. The random values inside the filter tensors are the weights of the convolutional layer.
- Sliding each filter across the spatial dimensions (width, height) of the input volume during the forward pass of information through the CNN. This produces a two dimensional output called an activation map for that specific filter.

2.2.1 Parameter Sharing

- Parameter sharing is used in CNN to control the total parameter count. Convolutional layers reduce the parameter count further by using a technique called parameter sharing.

- The user can reduce the number of parameters by making an assumption that if one feature can compute at some spatial position $(x,)$, then it is useful to compute a different place (x_2,y_2).
- In other words, denoting a single 2D slice of depth as a depth slice. For example, during back-propagation, every neuron in the network will compute the gradient for its weights, but these gradients will be added up across each depth slice and only update a single set of weights per slice.
- If all neurons in a single depth slice are using the same weight vector, then the forward pass of the convolutional layer can be computed in each depth slice as a convolution of the neuron's weights with the input volume. This is the reason why it is common to refer to the sets of weights as a filter (or a kernel), that is convolved with the input.
- Fig. 2.2.4 shows convolution shares the same parameters across all spatial locations.

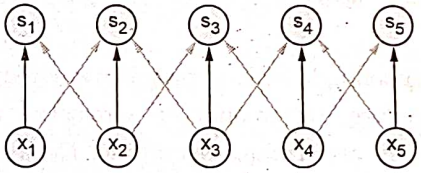


Fig. 2.2.4 Convolution shares the same parameters across all spatial locations

2.2.2 Equivariant Representation

- Convolution function is equivariant to translation. This means that shifting the input and applying convolution is equivalent to applying convolution to the input and shifting it.
- If we move the object in the input, its representation will move the same amount in the output.
- **General definition :** If representation(transform(x)) = transform(representation(x)) then representation is equivariant to the transform
- Convolution is equivariant to translation. This is a direct consequence of parameter sharing.
- It is useful when detecting structures that are common in the input. For example, edges in an image.
- Equivariance in early layers is good. We are able to achieve translation-invariance (via max-pooling) due to this property.

- Convolution is not equivariant to other operations such as change in scale or rotation.
- **Example of equivariance :** With 2D images convolution creates a map where certain features appear in the input. If we move the object in the input, the representation will move the same amount in the output. It is useful to detect edges in first layer of convolutional network. Same edges appear everywhere in image, so it is practical to share parameters across entire image.

2.3 Pooling

- Pooling helps the representation become slightly invariant to small translations of the input. A pooling function takes the output of the previous layer at a certain location L and computes a "summary" of the neighborhood around L.
- The pooling layer reduces the height and width of the input. It helps reduce computation, as well as helps make feature detectors more invariant to its position in the input
- The function of the pooling layer is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. No learning takes place on the pooling layers.
- The addition of a pooling layer after the convolutional layer is a common pattern used for ordering layers within a convolutional neural network that may be repeated one or more times in a given model.
- The pooling layer operates upon each feature map separately to create a new set of the same number of pooled feature maps. Pooling involves selecting a pooling operation, much like a filter to be applied to feature maps.
- The size of the pooling operation or filter is smaller than the size of the feature map. This means that the pooling layer will always reduce the size of each feature map by a factor of 2, e.g. each dimension is halved, reducing the number of pixels or values in each feature map to one quarter the size.
- For example, a pooling layer applied to a feature map of 6×6 (36 pixels) will result in an output pooled feature map of 3×3 (9 pixels). The pooling operation is specified, rather than learned.
- The pooling operation, also called subsampling, is used to reduce the dimensionality of feature maps from the convolution operation. Max pooling and average pooling are the most common pooling operations used in the CNN.

- Pooling units are obtained using functions like max-pooling, average pooling and even L2-norm pooling. At the pooling layer, forward propagation results in an N×N pooling block being reduced to a single value - value of the "winning unit". Back-propagation of the pooling layer then computes the error which is acquired by this single value "winning unit".
- Pooling layers, also known as down sampling, conducts dimensionality reduction, reducing the number of parameters in the input. Similar to the convolutional layer, the pooling operation sweeps a filter across the entire input, but the difference is that this filter does not have any weights. Instead, the kernel applies an aggregation function to the values within the receptive field, populating the output array. There are two main types of pooling :
- Max pooling : As the filter moves across the input, it selects the pixel with the maximum value to send to the output array. As an aside, this approach tends to be used more often compared to average pooling.
- Average pooling : As the filter moves across the input, it calculates the average value within the receptive field to send to the output array.
- Invariance to local translation can be useful if we care more about whether a certain feature is present rather than exactly where it is.

2.4 Variants of the Basic Convolution Function

- Various types of convolution functions are as follows :
  1. Stride
  2. Zero padding
  3. Unshared convolution
  4. Partial connectivity between channels
  5. Tiled
- Convolution functions used in practice differ slightly compared to convolution operation as it is usually understood in the mathematical literature.
- In general a convolution layer consists of application of several different kernels to the input. This allows the extraction of several different features at all locations in the input. This means that in each layer, a single kernel is not applied. Multiple kernels, are used as different feature detectors.
- The input is generally not real-valued but instead vector valued. Multi-channel convolutions are commutative only if number of output and input channels is the same.

- In order to allow for calculation of features at a coarser level strided convolutions can be used. The effect of strided convolution is the same as that of a convolution followed by a down sampling stage. This can be used to reduce the representation size.
- The **stride** indicates the pace by which the filter moves horizontally and vertically over the pixels of the input image during convolution. Fig. 2.4.1 shows stride during convolution.

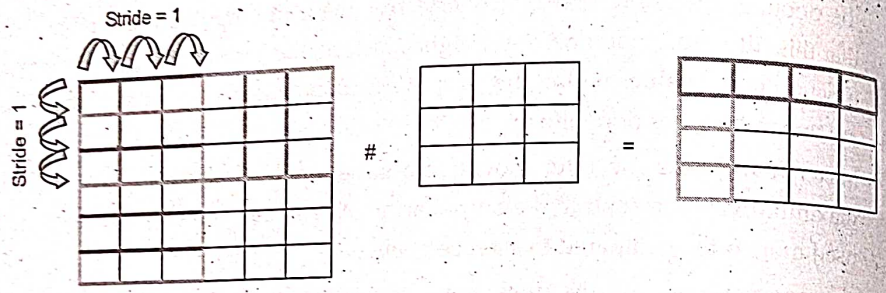


Fig. 2.4.1 Stride during convolution

- Stride is a parameter of the neural network's filter that modifies the amount of movement over the image or video. Stride is a component for the compression of images and video data. For example, if a neural network's stride is set to 1, the filter will move one pixel, or unit, at a time. If stride = 1, the filter will move one pixel.
- Stride depends on what we expect in our output image. We prefer a smaller stride size if we expect several fine-grained features to reflect in our output. On the other hand, if we are only interested in the macro-level of features, we choose a larger stride size.
- **Padding** is the process of adding one or more pixels of zeros all around the boundaries of an image, in order to increase its effective size. Zero padding helps to make output dimensions and kernel size independent. Three common zero padding strategies are :
  a) **Valid convolution :** Extreme case in which no zero-padding is used whatsoever, and the convolution kernel is only allowed to visit positions where the entire kernel is contained entirely within the input. For a kernel of size k in any dimension, the input shape of m in the direction will become $m - k + 1$ in the output. This shrinkage restricts architecture depth.
  b) **Same convolution :** Just enough zero-padding is added to keep the size of the output equal to the size of the input. Essentially, for a dimension where kernel

size is k, the input is padded by $k - 1$ zeros in that dimension

  c) **Full convolution :** Other extreme case where enough zeroes are added for every pixel to be visited k times in each direction, resulting an output image of width $m + k - 1$.
  d) The 1D block is composed by a configurable number of filters, where the filter has a set size; a convolution operation is performed between the vector and the filter, producing as output a new vector with as many channels as the number of filters. Every value in the tensor is then fed through an activation function to introduce nonlinearity.
- Fig. 2.4.2 shows 1D convolution with a kernel sized 3 and stride 1.

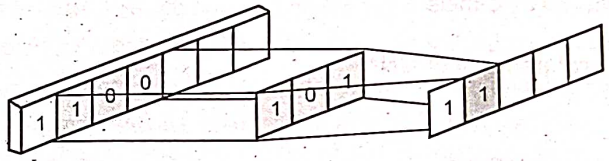


Fig. 2.4.2 1D convolution with a kernel sized 3 and stride 1.

- The default is to move filters of a set Width by 1 element at a time when performing convolutions; this is called Horizontal stride and it can be altered by the user. The bigger the stride, the smaller the output vector will be. This can be used to reduce the number of parameters and memory used but leads to a loss of information.
- In terms of test set accuracy, the optimal padding is somewhere between same and valid.

Unshared Convolution

- In some case when we do not want to use convolution but want to use locally connected layer. We use Unshared convolution. The Indices into weight W are respectively :
  i : Output channel
  j : Output row;
  k : Output column
  l : Input channel
  m : Row offset within input
  n : Column offset within input
- The linear part of a locally connected layer is then given by

$$Z_{i,j,k} = \sum_{l,m,n} V_{i,j+m-1,k+n-1} W_{i,j,k,l,m,n}$$

- Merits : Able to learn location sensitive filters.
- Demerit : Memory requirement increase only by a factor of the size of the entire output feature map.

Tiled Convolution

- Tiled convolution learn a set of kernels that is rotated through as we move through space, rather than learning a separate set of weights at every spatial location as in locally connected layer.
- It offers a compromise between a convolutional layer and a locally connected layer.
- Memory requirements for storing the parameters will increase only by a factor of the size of this set of kernels.

2.5 The Basic Architecture of CNN

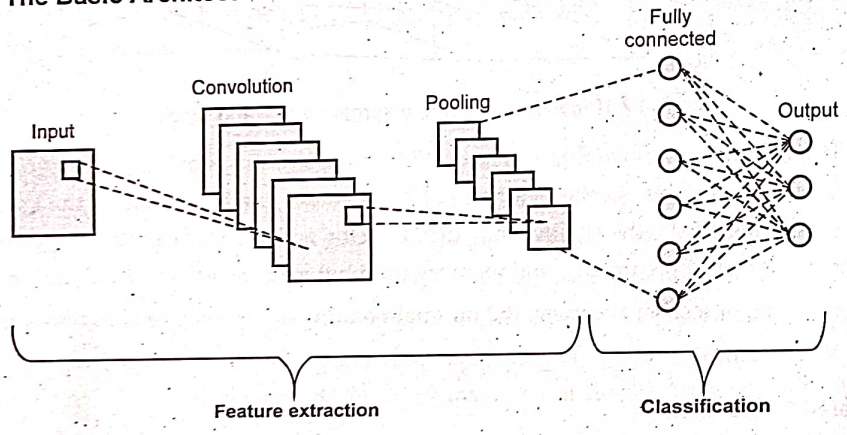


Fig. 2.5.1 Basic architecture of CNN

- A convolutional neural network, as discussed above, has the following layers that are useful for various deep learning algorithms. Let us see the working of these layers taking an example of the image having dimension of $12 \times 12 \times 4$. These are :
  1. **Input layer :** This layer will accept the image of width 12, height 12 and depth 4.
  2. **Convolution layer :** It computes the volume of the image by getting the dot product between the image filters possible and the image patch. For example, there are 10 filters possible, then the volume will be computed as $12 \times 12 \times 10$.
  3. **Activation function layer :** This layer applies activation function to each element in the output of the convolutional layer. Some of the well accepted activation functions are ReLu, Sigmoid, Tanh, Leaky ReLu, etc. These functions

will not change the volume obtained at the convolutional layer and hence it will remain equal to $12 \times 12 \times 10$.

4. **Pool layer :** This function mainly reduces the volume of the intermediate outputs, which enables fast computation of the network model, thus preventing it from overfitting.

2.6 Popular CNN Architecture : Alexnet

- Alexnet is a Deep Convolutional Neural Network for image classification.
- AlexNet is composed of 5 convolutional layers with a combination of max-pooling layers, 3 fully connected layers, and 2 dropout layers. The activation function used in all layers is Relu. The activation function used in the output layer is Softmax. The total number of parameters in this architecture is around 60 million.
- The AlexNet architecture was created with large-scale image datasets in mind, and it produced state-of-the-art results when it was first released. It has 60 million characteristics in all.
- Fig. 2.6.1 shows AlexNet architecture.

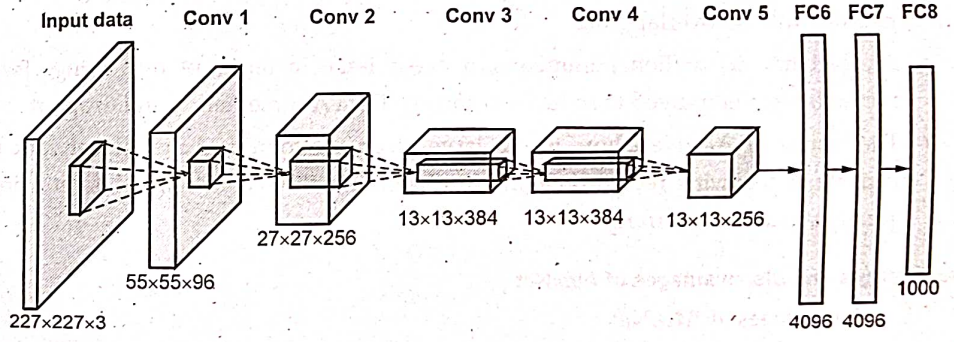


Fig. 2.6.1 AlexNet architecture

- For the AlexNet architecture, the convolutional kernels are extracted during the back-propagation optimization procedure by optimizing the whole cost function with the stochastic gradient descent algorithm.
- Generally, the convolutional layers act upon the input feature maps with the sliding convolutional kernels to generate the convolved feature maps, and the pooling layers operate on the convolved feature maps to aggregate the information within the given neighborhood window with a max pooling operation or average pooling operation.

- Max pooling is applied between every two conv layers. After the tensors are flattened, two fully-connected (dense) layers are used. The output layer is a softmax layer to compute the softmax loss function for learning
- AlexNet uses Rectified Linear Units (ReLU) instead of the tanh function, which was standard at the time. ReLU's advantage is in training time; a CNN using ReLU was able to reach a 25 % error on the CIFAR-10 dataset six times faster than a CNN using tanh.
- Use of multiple GPUs and parallel computing is highlighted in the training of AlexNet. The use of ReLU as the activation function for image classification by CNN.
- Multiple GPUs : GPUs were still rolling around with 3 gigabytes of memory. This was especially bad because the training set had 1.2 million images. AlexNet allows for multi-GPU training by putting half of the model's neurons on one GPU and the other half on another GPU. Not only does this mean that a bigger model can be trained, but it also cuts down on the training time.
- Overlapping Pooling.: CNNs traditionally "pool" outputs of neighboring groups of neurons with no overlapping.
- AlexNet had 60 million parameters, a major issue in terms of overfitting. Two methods were employed to reduce overfitting: Data Augmentation and dropout.
- The results of AlexNet show that a large, deep convolutional neural network is capable of achieving record-breaking results on a highly challenging dataset using purely supervised learning.

Advantages and Disadvantages of AlexNet

1. **Advantages of AlexNet**
   - ♦ AlexNet is considered as the milestone of CNN for image classification
   - ♦ Many methods such as the conv and pooling design, dropout, GPU, parallel computing, ReLU is still the industrial standard for computer vision.
   - ♦ The unique advantage of AlexNet is the directly image input to the classification model.
   - ♦ The convolution layers can automatically extract the edges of the images, and fully connected layers learning these features.
   - ♦ Theoretically the complexity of visual patterns can be effective extracted by adding more conv layers

2. **Disadvantages of AlexNet**
   - ♦ AlexNet is NOT deep enough compared to the later model such as VGG Net, GoogLENet, and ResNet.
   - ♦ The use of large convolution filters (5 * 5) is not encouraged.
   - ♦ Use normal distribution to initiate the weights in the neural networks cannot effective solve the problem of gradient vanishing, replaced by the Xavier method later:
   - ♦ The performance is surpassed by more complex models such as GoogLENet and ResNet.

❑❑❑