COMPUTING LAB-I

(CSC508)

LAB MANUAL

II Semester M. Tech (CSE)

Session - 2022-2023

Department of Computer Science and Engineering
IIT(ISM)

INDEX

1. i) CRYPTOSYSTEMS: Symmetric and Asymmetric

Aim/Objective:

Implementation the encryption and decryption using symmetric-key encryption algorithms.

Brief Description:

A cryptosystem is an implementation of cryptographic techniques and their accompanying infrastructure to provide information security services. A cryptosystem is also referred to as a cipher system.

Components of a Cryptosystem
The various components of a basic cryptosystem are as follows –

- **Plaintext.** It is the data to be protected during transmission.
- **Encryption Algorithm.** It is a mathematical process that produces a ciphertext for any given plaintext and encryption key. It is a cryptographic algorithm that takes plaintext and an encryption key as input and produces a ciphertext.
- **Ciphertext.** It is the scrambled version of the plaintext produced by the encryption algorithm using a specific the encryption key. The ciphertext is not guarded. It flows on public channel. It can be intercepted or compromised by anyone who has access to the communication channel.
- **Decryption Algorithm.** It is a mathematical process, that produces a unique plaintext for any given ciphertext and decryption key. It is a cryptographic algorithm that takes a ciphertext and a decryption key as input, and outputs a plaintext. The decryption algorithm essentially reverses the encryption algorithm and is thus closely related to it.
- **Symmetric Key Cryptosystem:** It is an encryption system where the sender and receiver of message use a single common key to encrypt and decrypt messages. Symmetric Key Systems are faster and simpler but the problem is that sender and receiver have to somehow exchange key in a secure manner. Eg- Caesar Cryptosystem
- **Asymmetric Key Cryptosystem:** Under this system a pair of keys is used to encrypt and decrypt information. A public key is used for encryption and a private key is used for decryption. Public key and Private Key are different. Even if the public key is known by everyone the intended receiver can only decode it because he alone knows the private key. Eg- RSA

Caesar Cryptosystem

In this one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a left shift of 3, D would be replaced by A, E would become B, and so on.

Apparatus and components required:

Compute with C/C++ compiler.

Experimental/numerical procedure:

Coding, compilation, editing, run and debug.

Questions:

i) Encrypt and Decrypt using Diagraph, C=(aP+b)mod N, a=3,b=5.

ii) Decipher the message YITJP GWJOW FAQTQ XCSMA ETSQU SQAPU SQGKCPQTYJ with the help of Hill cipher with the inverse key $\begin{pmatrix} 5 & 1 \\ 2 & 7 \end{pmatrix}$.

Rivest Shamir Adleman (RSA):

Aim/Objective:

Implementation of RSA algorithm for encryption and decryption.

Brief Description:

RSA Algorithm is used to encrypt and decrypt data in modern computer systems and other electronic devices. RSA algorithm is an asymmetric cryptographic algorithm as it creates 2 different keys for the purpose of encryption and decryption. It is public key cryptography as one of the keys involved is made public. RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman who first publicly described it in 1978.

Working of RSA algorithm

RSA involves use of public and private key for its operation. The keys are generated using the following steps:-

1. Two prime numbers are selected as **p** and **q**.
2. **n = p.q** which is the modulus of both the keys.
3. Calculate **totient= (p-1).(q-1)**
4. Choose **e** such that **e > 1** and co-prime to **totient** which means **gcd (e, totient)** must be equal to **1**, **e** is the public key
5. Choose **d** such that it satisfies the equation **d.e = 1 + k (totient)**, **d** is the private key not known to everyone.
6. Cipher text is calculated using the equation **c = m^e mod n** where **m** is the message.
7. With the help of **c** and **d** we decrypt message using equation **m = c^d mod n** where **d** is the private key.

Apparatus and components required:

Compute with C/C++/Java/python compiler.

Experimental/numerical procedure:

Coding, compilation, editing, run and debug.

Questions:

1) Encrypting and decrypting small numeral values, say p=29, q=37, m=211.
2) Encrypting and decrypting plain text messages containing alphabets using their ASCII value, m="This is Computing Lab-1".

ii) Diffie–Hellman Key Exchange

Aim/Objective:

Implementation Key Exchange algorithm: Diffie-Hellman.

Brief Description:

Key exchange (also key establishment) is a method in cryptography by which cryptographic keys are exchanged between two parties, allowing use of a cryptographic algorithm. If the sender and receiver wish to exchange encrypted messages, each must be equipped to encrypt messages to be sent and decrypt messages received. The nature of the equipping they require depends on the encryption technique they might use. If they use a code, both will require a copy of the same codebook. If they use a cipher, they will need appropriate keys. If the cipher is a symmetric key cipher, both will need a copy of the same key. If it is an asymmetric key cipher with the public/private key property, both will need the other's public key.

Diffie–Hellman Key Exchange establishes a shared secret between two parties that can be used for secret communication for exchanging data over a public network. It is primarily used as a method of exchanging cryptography keys for use in symmetric encryption algorithms like AES. The algorithm in itself is very simple. The process begins by having the two parties, Alice and Bob. Let's assume that Alice wants to establish a shared secret with Bob.

Algorithm

STEP-1: Both Alice and Bob shares the same public keys g and p.
STEP-2: Alice selects a random public key a.
STEP-3: Alice computes his secret key A as g^a (mod p).
STEP-4: Then Alice sends A to Bob.
STEP-5: Similarly Bob also selects a public key b and computes his secret key B as g^b (modp).
And sends the same back to Alice.
STEP-6: Now both of them compute their common secret key as ((g^a)^b) modp or ((g^b)^a)modp.

Apparatus and components required:

Compute with C/C++ compiler.

Experimental/numerical procedure:

Coding, compilation, editing, run and debug.

Questions:

1) Find the private and public keys for both Alice and Bob when p=23, g=9.
2) Do the same when p=53, g=11.

2. Primality Testing Methods: i) Miller-Rabin ii) Pollard's Rho Method

Aim/Objective:

Implementation of the Primality Testing.

Brief Description:

Given a positive integer, check if the number is prime or not. A prime is a natural number greater than 1 that has no positive divisors other than 1 and itself. Examples of the first few prime numbers are {2, 3, 5, …}.

Miller Rabin Method:

millerTest(int n, int d)
1) Pick a random number 'a' in range [2, n-2]
2) Compute: x = pow(a, d) % n
3) If x == 1 or x == n-1, return true.
4) Do following while d doesn't become n-1.
   a) x = (x*x) % n.
   b) If (x == 1) return false.
   c) If (x == n-1) return true.

Pollard's Rho Method:

1. Start with random x and c. Take y equal to x and f(x) = x2 + c.
2. While a divisor isn't obtained
3. Update x to f(x) (modulo n) [Tortoise Move]
4. Update y to f(f(y)) (modulo n) [Hare Move]
5. Calculate GCD of |x-y| and n
6. If GCD is not unity
   If GCD is n, repeat from step 2 with another set of x, y and c
   Else GCD is our answer

Apparatus and components required:

Compute with C/C++/Java/Python compiler.

Experimental/numerical procedure:

Coding, compilation, editing, run and debug.

Questions:

1) Find all primes less than a number **N**using Miller-Rabin Method, take k=4.
2) For a given number **N** check if it is prime or not using Pollard's Rho Algorithm.

3. Optimization Techniques: Particle Swarm Optimization(PSO)

Aim/Objective:

Implementation of Optimization Techniques: Particle swarm optimization.

Brief Description:

Particle swarm optimization (PSO) is an evolutionary computation approach to solve non linear global optimization problems. The PSO idea was made based on simulation of a simplified social system, the graceful but unpredictable choreography of birds flock. This system is initialized with a population of random solutions that are updated during iterations.

Parameters- f: Objective function, Vi: Velocity of the particle or agent, A: Population of agents, W: Inertia weight, C1: cognitive constant, U1, U2: random numbers, C2: social constant, Xi: Position of the particle or agent, P_b: Personal Best, g_b: global Best

Algorithm:

1. Create a 'population' of agents (particles) which is uniformly distributed over X.
2. Evaluate each particle's position considering the objective function(say the below function).
   z=f(x, y)=sinx^2+siny^2+sinx*siny
3. If a particle's present position is better than its previous best position, update it.
4. Find the best particle (according to the particle's last best places).
5. Update particles' velocities.
   $$V_i^{t+1} = W.V_i^t + c_1 U_1^t \left(P_{b_1}^{t+1} - P_i^t\right) + c_2 U_2^t \left(g_b^t - P_i^t\right)$$
6. Move particles to their new positions.
   $$P_i^{t+1} = P_i^t + v_i^{t+1}$$
7. Go to step 2 until the stopping criteria are satisfied.

Rastrigin function is a non-convex function and is often used as a performance test problem for optimization algorithms. **Equation** ->$f(x_1 \dots \dots x_n) = 10\text{n} + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$, minimum at f(0,…,0)=0.

Apparatus and components required:

Compute with C/C++/Java/Python compiler.

Experimental/numerical procedure:

Coding, compilation, editing, run and debug.

Questions:

Implement particle swarm optimization on rastrigin function.

4. Machine Learning Algorithms: i) Decision Tree ii) Naïve Bayes iii) Random Forest iv) KNN v) SVM vi) K-Means

i) Decision Tree:

Aim/Objective:

Implementation of Decision Tree with the help of given data.

Brief Description:

Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed.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.

Classification of Machine Learning

At a broad level, machine learning can be classified into three types:

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Supervised Learning

Supervised learning is a type of machine learning method in which we provide sample labeled data to the machine learning system in order to train it, and on that basis, it predicts the output.

The system creates a model using labeled data to understand the datasets and learn about each data, once the training and processing are done then we test the model by providing a sample data to check whether it is predicting the exact output or not.

Supervised learning can be grouped further in two categories of algorithms:

- Classification
- Regression

Unsupervised Learning

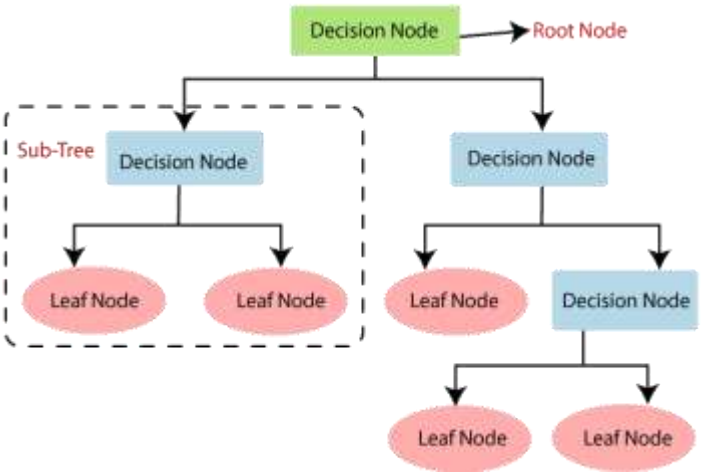
Unsupervised learning is a learning method in which a machine learns without any supervision.

The training is provided to the machine with the set of data that has not been labeled, classified, or categorized, and the algorithm needs to act on that data without any supervision. The goal of unsupervised learning is to restructure the input data into new features or a group of objects with similar patterns.

In unsupervised learning, we don't have a predetermined result. The machine tries to find useful insights from the huge amount of data. It can be further classifieds into two categories of algorithms:

- Clustering
- Association

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.



Algorithm:

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.
Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).
Step-3: Divide the S into subsets that contains possible values for the best attributes.
Step-4: Generate the decision tree node, which contains the best attribute.
Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Attribute Selection Measures
While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as Attribute selection measure or ASM. By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

1.**Information Gain:**

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.

Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

Entropy:
Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:
Entropy(s)= -P(yes)log_2 P(yes)- P(no))log_2 P(no)
Where,
S= Total number of samples
P(yes)= probability of yes
P(no)= probability of no

2.**Gini Index:**

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.

Gini Index= 1- $\sum$jPj^2

Apparatus and components required:

Compute with C/C++/Java/Python compiler.

Experimental/numerical procedure:

Coding, compilation, editing, run and debug.

ii) Navie Bayes:

Aim/Objective:

Implementation of Navie Bayes Algorithm using given data set.

Brief Description:

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.It is mainly used in text classification that includes a high-dimensional training dataset.It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

Naïve: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.

Bayes: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem:
Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
The formula for Bayes' theorem is given as:

$$P(A|B) = \frac{P(B|A).P(A)}{P(B)}$$

Where,
P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.
P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.
P(A) is Prior Probability: Probability of hypothesis before observing the evidence.
P(B) is Marginal Probability: Probability of Evidence.

Step:

- Step 1: Handling Data
  Data is loaded from the CSV File and spread into training and tested assets.
- Step 2: Summarizing the Data
  Summarise the properties in the training data set to calculate the probabilities and make predictions.
- Step 3: Making a Prediction
  A particular prediction is made using a summarise of the data set to make a single prediction.
- Step 4: Making all the Predictions
  Generate prediction given a test data set and a summarise data set.
- Step 5: Evaluate Accuracy

Accuracy of the prediction model for the test data set as a percentage correct out of them all the predictions made.

- Step 6: Tying all Together
  Finally, we tie to all steps together and form our own model of Naive Bayes Classifier.

Apparatus and components required:

Compute with C/C++/Java/Python compiler.

Experimental/numerical procedure:

Coding, compilation, editing, run and debug.

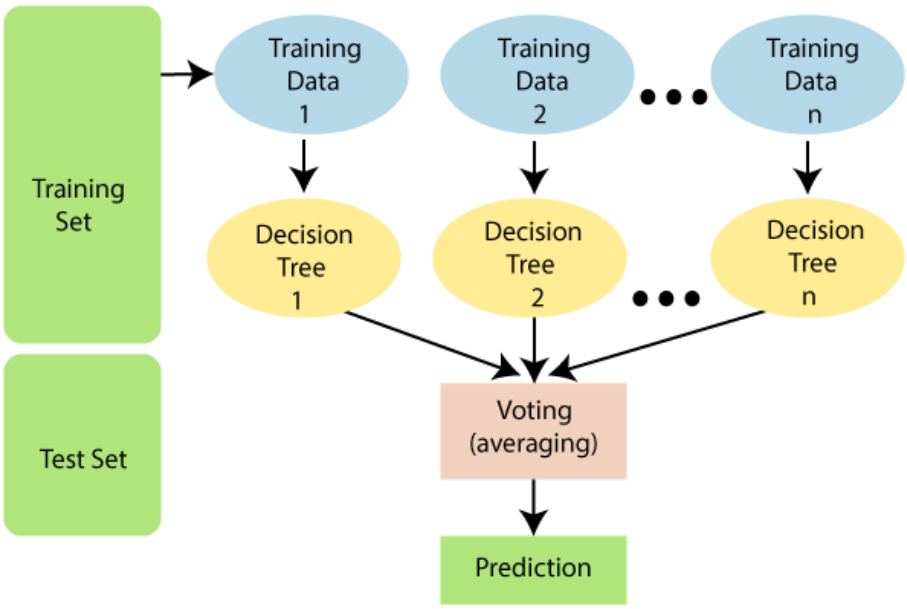
iii) Random Forest:

Aim/Objective:

Implementation of Random Forest using given data set.

Brief Description:

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.
As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."
The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.



Step to implement Random Forest Tree:
The Working process can be explained in the below steps and diagram:

- **Step-1:** Select random K data points from the training set.
- **Step-2:** Build the decision trees associated with the selected data points (Subsets).
- **Step-3:** Choose the number N for decision trees that you want to build.
- **Step-4:** Repeat Step 1 & 2.
- **Step-5:** For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

Apparatus and components required:

Compute with C/C++/Java/Python compiler.

Experimental/numerical procedure:

Coding, compilation, editing, run and debug.

iv) K-Nearest Neighbor(K-NN):

Aim/Objective:

Implementation of KNN(k-Nearest Neighbors) algorithm on the given data-set.

Brief Description:

K-Nearest Neighbors is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection. The K-Nearest Neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) with some mathematics we might have learned in our childhood— calculating the distance between points on a graph. There are other ways of calculating distance, and one way might be preferable depending on the problem we are solving. However, the straight-line distance (also called the Euclidean distance) is a popular and familiar choice. It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data).

Algorithm:

The K-NN working can be explained on the basis of the below algorithm:
Step-1: Select the number K of the neighbors.
Step-2: Calculate the Euclidean distance of K number of neighbors.
Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.
Step-4: Among these k neighbors, count the number of the data points in each category.
Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.
Step-6: Our model is ready.

Apparatus and components required:

Compute with C/C++ compiler.

Experimental/numerical procedure:

Coding, compilation, editing, run and debug.

v) Support Vector Machine(SVM):

Aim/Objective:

Classifying data using Support Vector Machines(SVMs) in a given data set.

Brief Description:

An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification, implicitly mapping their inputs into high-dimensional feature spaces.

Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

Algorithm:

1. Load the important libraries
2. Import dataset and extract the X variables and Y separately.
3. Divide the dataset into train and test
4. Initializing the SVM classifier model
5. Fitting the SVM classifier model
6. Coming up with predictions
7. Evaluating model's performance

Apparatus and components required:

Compute with Python compiler.

Experimental/numerical procedure:

Coding, compilation, editing, run and debug.

vi) K-Means Clustering:

Aim/Objective:

Implementation of K-means clustering with the help of given data set.

Brief Description:

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties. It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

Algorithim:

The working of the K-Means algorithm is explained in the below steps:
Step-1: Select the number K to decide the number of clusters.
Step-2: Select random K points or centroids. (It can be other from the input dataset).
Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.
Step-4: Calculate the variance and place a new centroid of each cluster.
Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.
Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.
Step-7: The model is ready.

Apparatus and components required:

Compute with C/C++/Python/Java compiler.

Experimental/numerical procedure:

Coding, compilation, editing, run and debug.

Questions:

1) Implement the K means on the given data set mall_customers_data.csv | Kaggle
2) Implement the K means on the given data set.

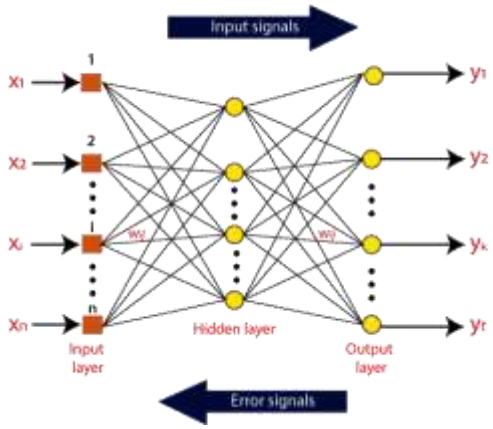
5. Neural Network : Convolution Neural Network(CNN)

Aim/Objective:

Implementation of Convolution Neural Network(ConvNet/CNN) using given image data set.

Brief Description:

Neural networks, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.Artificial neural networks (ANNs) are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.Neural networks rely on training data to learn and improve their accuracy over time.



A Convolution Neural Network(ConvNet/CNN) it is a Deep Learning Algorithm the purpose is that it takes image as input and from that gain importance of various part of image so that it can be differentiable from other part of the same image.The preprocessing of image done in ConvNet which in compared to other algorithm it is lower,and it have the ability of learning the characteristic and filter of image. CNN classified into three layers such as:

• Convolution Layer

• Pooling Layer

• Fully-connected Layer(FC)

Initially each layer work on small part of the image and focus on simple feature of image and then increase with layer the complexity of the image increases and feature from it also increase so that it extract feature from image at last.

Convolution Layer

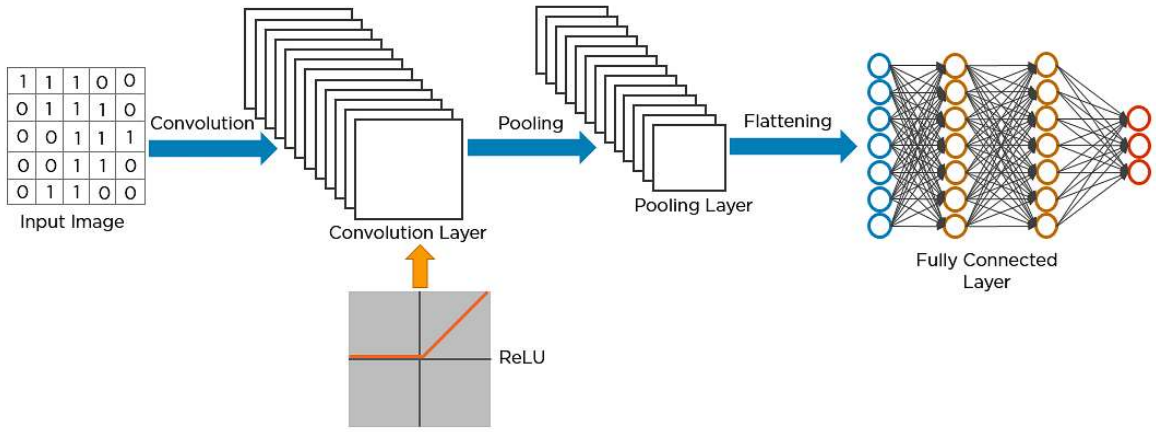
This is known as the main building block of CNN. In this layer most of computation occurs. The required component of the Convolution layer is that input image,filter(it help in feature detector) and also feature map which help in mapping the feature of each section of the metrics.In image it contain a 3D space which is RGB image .Filter work on the each field of image in order to feature and this process is called Convolution. Here the Filter is 2D ,which store only 2D feature in order do that it apply dot product between the input pixels and the filter and store in respective output array,and this process happen across the entire image and gain a output in form of dot product between input pixel and the filter and this whole known as a feature map or convolution feature. This repeat process help in converting them into a hierarchical layer and which is a reduced form of previous layer of feature extraction.And at last this layer generate the numerical values from the images and which help for neural network to extract the related pattern from it and interpret from the numerical value.

Pooling Layer

This layer termed in many way such as down sampling, conduct dimensionality reduction but the working principle is same as the Convolution Layer, but only difference is that it does not contain any weight corresponding to the filter process.This layer cause loss of information loss but come with more of the benefits that it help in reducing the complexity, help in improving the efficiency and also help in limit the risk of overfitting.There are two types of it, named as Max pooling and Average pooling.

Fully Connected Layer

This layer describe that each layer of pervious connected to next layer output. It perform the task of classification which is based on feature extraction from the previous layer and also their different among the filters. Fully Connected layer produce the value in term of probability which is ranges from 0 to 1.



Use case implementation using CNN

- Download the data set
- Import the data set
- Read the label names
- Display the images using matplotlib
- Use the helper function to handle data
- Create the model
- Apply the helper functions

- Create the layers for convolution and pooling
- Create the flattened layer by reshaping the pooling layer
- Create a fully connected layer
- Set the output to y_pred variable
- Apply the loss function
- Create the optimizer
- Create a variable to initialize all the global variables
- Run the model by creating a graph session

Apparatus and components required:

Compute with C/C++/Java/Python compiler.

Experimental/numerical procedure:

Coding, compilation, editing, run and debug.