Pimpri-Chinchwad Educational Trusts

Pimpri-Chinchwad College of Engineering And Research,

Ravet, Pune

DEPARTMENT OF COMPUTER ENGINEERING

LAB MANUAL

LAB PRACTICE V

	Pimpri Chinchwad Education Trust's **Pimpri Chinchwad College of Engineering & Research Ravet, Pune** **IQAC PCCOER**	
		Term: II

LABORATORY MANUAL

Subject: Laboratory Practice-V

[Subject Code: 410254]

Class: BE Computer Engineering

Semester: II

Prepared by:

Dr.Yogeshwari V. Mahajan
Mrs. Shrinika Mahajan
Subject Teachers

Approved by

Dr. Archana A. Chaugule
H.O.D

Vision – Mission of the Institute

Vision

To be a Premier institute of technical education & research to serve the need of society and all the stakeholders.

Mission

To establish state-of-the-art facilities to create an environment resulting in individuals who are technically sound having professionalism, research and innovative aptitude with high moral and ethical values.

Vision – Mission of the Computer Department

Vision

To strive for excellence in the field of Computer Engineering and Research through Creative Problem Solving related to societal needs.

Mission:

M1: Establish strong fundamentals, domain knowledge and skills among the students with analytical thinking, conceptual knowledge, social awareness and expertise in the latest tools & technologies to serve industrial demands.

M2: Establish leadership skills, team spirit and high ethical values among the students to serve industrial demands and societal needs.

M3: Guide students towards Research and Development, and a willingness to learn by connecting themselves to the global society.

Program Educational Objectives (PEO)

PEO1: To prepare graduates who have strong mathematical, scientific, and Computer enginee fundamentals, to meet technological challenges and be globally compe

PEO2: To prepare committed and motivated graduates with strong communication, ethical va leadership skills, which augment their professional competency and make them productive t players.

PEO3: To prepare graduates with technical proficiency, research outlook and problem-sol abilities to produce innovative solutions in the field of Computer Engineering.

Program Specific Outcomes (PSO)

PSO1: Problem-Solving Skills- The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality project.

PSO2: Professional Skills-The ability to understand, analyze and develop computer programs in the areas related to algorithms, software testing, application software, web design, data analytics, IOT and networking for efficient design of computer-based

PSO3: Successful Career and Entrepreneurship- The ability to employ modern computer languages, environments, and platforms in creating innovative career paths to be an entrepreneur, and a zest for higher studies, and to generate IPR & Deliver a quality project..

Course Objectives:

- To understand and implement searching and sorting algorithms.
- To learn the fundamentals of GPU Computing in the CUDA environment.
- To illustrate the concepts of Artificial Intelligence/Machine Learning(AI/ML).
- To understand Hardware acceleration.
- To implement different deep learning models.

Course Outcomes:

CO	Statements	Cognitive level of learning
C414.1	**Analyze and measure** performance of sequential and parallel algorithms.	**(Analyze)**
C414.2	**Design and Implement** solutions for multicourse/Distributed/parallel environment.	**(Design)**
C414.3	**Identify and apply** the suitable algorithms to solve AI/ML problems.	**(Apply)**
C414.4	**Apply** the technique of Deep Neural network for implementing Linear regression and classification.	**(Apply)**
C414.5	**Apply** the technique of Convolution (CNN) for implementing Deep Learning models.	**(Apply)**
C414.6	**Design and develop** Recurrent Neural Network (RNN) for prediction.	**(Design)**

Index

Sr. No.	Title of the Experiment	CO	Date of Performance	Page No.		Sign. of teacher	Remarks*
				From	To		
1	Design and implement Parallel Breadth First Search and Depth First Search based on existing algorithms using OpenMP. Use a Tree or an undirected graph for BFS and DFS .	C414.1					
2	.Write a program to implement Parallel Bubble Sort and Merge sort using OpenMP. Use existing algorithms and measure the performance of sequential and parallel algorithms.	C414.1					
3	Implement Min, Max, Sum and Average operations using Parallel Reduction.	C414.1					
4	Write a CUDA Program for : 1. Addition of two large vectors 2. Matrix Multiplication using CUDA C	C414.1					
5	**Mini Project :((HPC)** Students are Implement any one mini project.	C414.1					
6	Linear regression by using Deep Neural network: Implement Boston housing price predictionproblem by Linear regression using Deep Neural network. Use Boston House price predictiondataset	C414.2					
7	Classification using Deep neural network (Any One from the following) 1. Multiclass classification using Deep Neural Networks: Example: Use the OCR letter recognition datasethttps://archive.ics.uci.edu/ml/datasets/letter+recognition 2. Binary classification using Deep Neural Networks Example: Classify movie reviews into positive" reviews and "negative" reviews, just based on the text content of the reviews. Use IMDB dataset	C414.2					

8	Convolutional neural network (CNN) (Any One from the following) • Use any dataset of plant disease and design a plant disease detection system using CNN. • Use MNIST Fashion Dataset and create a classifier to classify fashion clothing into categories.	C414.3					
9	Recurrent neural network (RNN) Use the Google stock prices dataset and design a time seriesanalysis and prediction system using RNN.	C414.3					
10	**Mini Project :(Deep Learning)** Implement any one mini project.	C414.3					

CERTIFICATE

This is to certify that Mr./Miss/Mrs._____

Roll No.:_____Exam. Seat No.:_____of SE/TE/BE Computer has carried out above practical /term work within PCCOER as prescribed by SavitribaiPhule Pune University, Pune during the academic year 20 -20 . His/Her performance is satisfactory and attendance is_____%.

Date: **Faculty I/C** **HOD** **Principal**

410250 : High Performance Computing

Group A

Assignment No. 1

Problem Statement 1: Design and implement Parallel Breadth First Search and Depth First Search based on existing algorithms using OpenMP. Use a Tree or an undirected graph for BFS and DFS .

Objective of the Assignment: Students should be able to perform Parallel Breadth First Search based on existing algorithms using OpenMP.

Prerequisite: 1. Basic of programming language

2. Concept of BFS

3. Concept of Parallelism

What is BFS?

BFS stands for Breadth-First Search. It is a graph traversal algorithm used to explore all the nodes of a graph or tree systematically, starting from the root node or a specified starting point, and visiting all the neighboring nodes at the current depth level before moving on to the next depth level.

The algorithm uses a queue data structure to keep track of the nodes that need to be visited, and marks each visited node to avoid processing it again. The basic idea of the BFS algorithm is to visit all the nodes at a given level before moving on to the next level, which ensures that all the nodes are visited in breadth-first order.

BFS is commonly used in many applications, such as finding the shortest path between two nodes, solving puzzles, and searching through a tree or graph.

Example of BFS

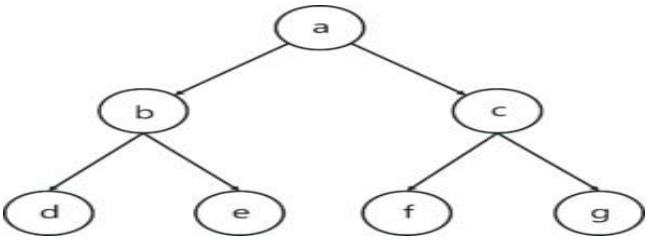
Now let's take a look at the steps involved in traversing a graph by using Breadth-First Search:

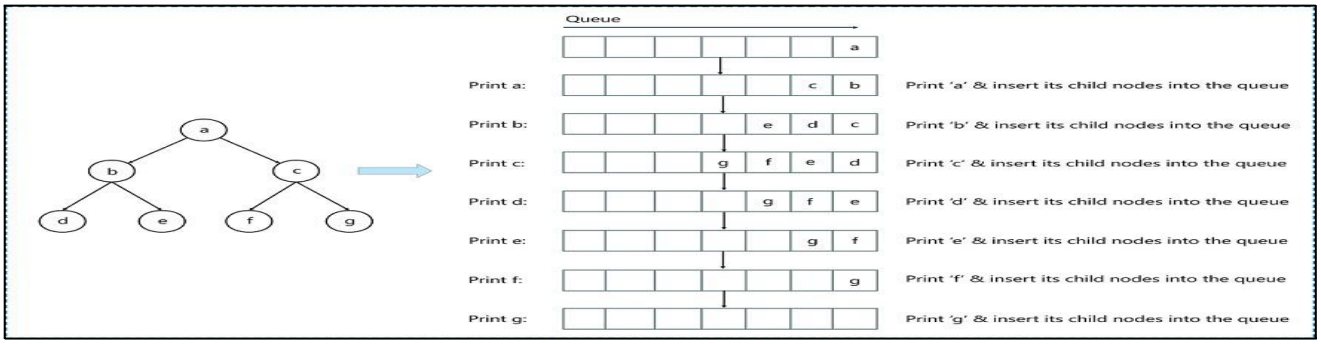
Step1: Take an Empty Queue.

Step 2: Select a starting node (visiting a node) and insert it into the Queue.

Step3: Provided that the Queue is not empty, extract the node from the Queue and insert its child nodes(exploring a node) into the Queue.

Step 4: Print the extracted node.





Concept of OpenMP

- OpenMP (Open Multi-Processing) is an application programming interface (API) that supports shared-memory parallel programming in C, C++, and Fortran. It is used to write parallel programs that can run on multicore processors, multiprocessor systems, and parallel computing clusters.
- OpenMP provides a set of directives and functions that can be inserted into the source code of a program to parallelize its execution. These directives are simple and easy to use, and they can be applied to loops, sections, functions, and other program constructs. The compiler then generates parallel code that can run on multiple processors concurrently.
- OpenMP programs are designed to take advantage of the shared-memory architecture of modern processors, where multiple processor cores can access the same memory. OpenMP uses a fork-join model of parallel execution, where a master thread forks multiple worker threads to execute a parallel region of the code, and then waits for all threads to complete before continuing with the sequential part of the code.
- OpenMPiswidelyusedinscientificcomputing,engineering,andotherfieldsthatrequirehigh-performance computing. It is supported by most modern compilers and is available on a

widerange of platforms, including desktops, servers, and supercomputers.

HowParallelBFSWork

- Parallel BFS(Breadth-First Search)is an algorithm used to explore all the nodes of a graph or tree the metrically in parallel. It is a popular parallel algorithm used for graph traversal in distributed computing, shared-memory systems, and parallel clusters.
- The parallel BFS algorithm starts by selecting a root node or a specified starting point, and then assigning it to a thread or processor in the system. Each thread maintains a local queue of nodes to be visited and marks each visited node to avoid processing it again.
- The algorithm then proceeds in levels, where each level represents a set of nodes that are at a certain distance from the root node. Each thread processes the nodes in its local queue at the current level, and then exchanges the nodes that are adjacent to the current level with other threads or processors. This is done to ensure that the nodes at the next level are visited by the next iteration of the algorithm.
- The parallel BFS algorithm uses two phases: the computation phase and the communication phase. Inthecomputationphase,eachthreadprocessesthenodesinitslocalqueue,whileinthecommunication phase, the threads exchange the nodes that are adjacent to the current level with other threads or processors.
- The parallel BFS algorithm terminates when all nodes have been visited or when a specified node has-been found. The result of the algorithm is the set of visited nodes or the shortest path from the root node to the target node.
- Parallel BFS can be implemented using different parallel programming models, such as OpenMP, MPI, CUDA, and others. The performance of the algorithm depends on the number of threads or processors used, the size of the graph, and the communication overhead between the threads or processors.

Conclusion-In this way we can achieve parallelism while implementing BFS

Program:

```
#include<iostream>
#include<stdlib.h>
#include<queue>
using namespace std;

class node
{
public:
```

```
node *left, *right;
int data;
};

class Breadthfs
{
public:

node *insert(node *, int);
void bfs(node *);

};

node *insert(node *root, int data)
// inserts a node in tree
{

if(!root)
   {

        root=new node;
        root->left=NULL;
        root->right=NULL;
        root->data=data;
        return root;
   }

queue<node *> q;
q.push(root);

while(!q.empty())
   {

        node *temp=q.front();
        q.pop();

        if(temp->left==NULL)
         {

                temp->left=new node;
                temp->left->left=NULL;
                temp->left->right=NULL;
                temp->left->data=data;
                return root;
         }
        else
         {
```

```
        q.push(temp->left);

         }

        if(temp->right==NULL)
         {

                temp->right=new node;
                temp->right->left=NULL;
                temp->right->right=NULL;
                temp->right->data=data;
                return root;
         }
        else
         {

        q.push(temp->right);

         }

   }

}

void bfs(node *head)
{
        queue<node*> q;
        q.push(head);

        int qSize;

        while (!q.empty())
         {
                qSize = q.size();
                 #pragma omp parallel for
        //creates parallel threads
                for (int i = 0; i < qSize; i++)
                 {
                        node* currNode;
                         #pragma omp critical
                         {
                        currNode = q.front();
                        q.pop();
                        cout<<"\t"<<currNode->data;

                         }// prints parent node
                         #pragma omp critical
                         {
```

```
                        if(currNode->left)// push parent's left node in queue
                                q.push(currNode->left);
                        if(currNode->right)
                                q.push(currNode->right);
                         }// push parent's right node in queue

                }
        }

}
int main(){

node *root=NULL;
int data;
char ans;

do
   {
        cout<<"\n enter data=>";
        cin>>data;

        root=insert(root,data);

        cout<<"do you want insert one more node?";
        cin>>ans;

}while(ans=='y'||ans=='Y');

bfs(root);

return 0;
}
```

Run Commands:

```
g++ -fopenmp bfs.cpp -o bfs
./bfs
```

Output:

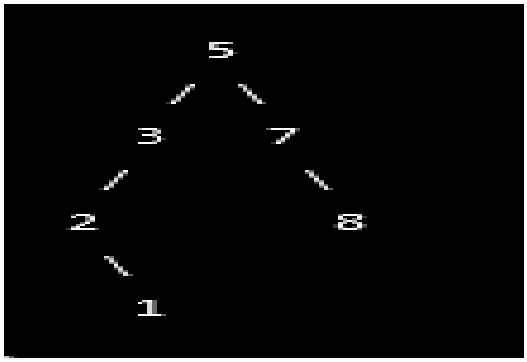
This code represents a breadth-first search (BFS) algorithm on a binary tree using OpenMP for parallelization. The program asks for user input to insert nodes into the binary tree and then performs the BFS algorithm using multiple threads. Here's an example output for a binary tree with nodes 5, 3, 2, 1, 7, and 8:

```
Enter data => 5
Do you want to insert one more node? (y/n) y

Enter data => 3
Do you want to insert one more node? (y/n) y

Enter data => 2
Do you want to insert one more node? (y/n) y

Enter data => 1
Do you want to insert one more node? (y/n) y

Enter data => 7
Do you want to insert one more node? (y/n) y

Enter data => 8
Do you want to insert one more node? (y/n) n

        5       3       7       2       1       8
```

The nodes are printed in breadth-first order. The #pragma omp parallel for statement is used to parallelize the for loop that processes each level of the binary tree. The #pragma omp critical statement is used to synchronize access to shared data structures, such as the queue that stores the nodes of the binary tree.

Here is an example of the breadth-first traversal for a binary tree with the values 5, 3, 2, 1, 7, and 8:

Starting with the root node containing value 5:



The traversal would be:

```
5, 3, 7, 2, 8, 1
```

Design and implement Parallel Depth First Search based on existing algorithms using OpenMP. Use a Tree or an undirected graph for DFS

Objective of the Assignment: Students should be able to perform Parallel Depth First Search based

onexisting algorithms using OpenMP

Prerequisite:

1. Basic of programming language
2. Concept of DFS
3. Concept of Parallelism

Title of the Assignment: Design and implement Parallel Depth First Search based on existing algorithms using OpenMP. Use aTree or an undirected graph for DFS

Objective of the Assignment: Students should be able to perform Parallel Depth First Search based onexisting algorithms using OpenMP

Prerequisite:

4. Basic of programming language
5. Concept ofDFS
6. Concept of Parallelism

Contents for Theory:

1. **What is DFS?**
2. **Example of DFS**
3. **Concept of OpenMP**
4. **How Parallel DFS Work**

Program:2 Parallel Depth First Search based on existing algorithms using OpenMP

```
#include <iostream>
#include <vector>
#include <stack>
#include <omp.h>
```

```
using namespace std;

const int MAX = 100000;
vector<int> graph[MAX];
bool visited[MAX];

void dfs(int node) {
stack<int> s;
s.push(node);

while (!s.empty()) {
int curr_node = s.top();
s.pop();

if (!visited[curr_node]) {
visited[curr_node] = true;

if (visited[curr_node]) {
cout<< curr_node << " ";
}

#pragma omp parallel for
for (int i = 0; i < graph[curr_node].size(); i++) {
int adj_node = graph[curr_node][i];
if (!visited[adj_node]) {
s.push(adj_node);
}
}
}
}
}

int main() {
int n, m, start_node;
cout<< "Enter No of Node,Edges,and start node:" ;
cin>> n >> m >> start_node;
//n: node,m:edges

cout<< "Enter Pair of edges:" ;
for (int i = 0; i < m; i++) {
int u, v;

cin>> u >> v;
//u and v: Pair of edges
graph[u].push_back(v);
graph[v].push_back(u);
}
```

```
#pragma omp parallel for
for (int i = 0; i < n; i++) {
visited[i] = false;
}

dfs(start_node);

/*      for (int i = 0; i < n; i++) {
if (visited[i]) {
cout<< i << " ";
}
}*/

return 0;
}
```

Explanation:

Let's go through the code step by step:

1. We start by including the necessary headers and declaring some global variables, such as the graph adjacency list, an array to keep track of visited nodes, and a maximum limit for the number of nodes in the graph.
2. Next, we define a function called dfs() which takes a starting node as input and performs the depth-first search algorithm. We use a stack to keep track of the nodes to be visited. The algorithm works as follows:

- We push the starting node onto the stack.
- While the stack is not empty, we pop the top node from the stack.
- If the current node has not been visited, we mark it as visited and explore all its neighbors that have not been visited yet by adding them to the stack.

3. In the main() function, we read the input values: the number of nodes in the graph, the number of edges, and the starting node.
4. We then read the edge information and build the adjacency list of the graph.
5. We initialize the visited array to false for all nodes in the graph using OpenMP's parallel for construct to allow multiple threads to set the array values in parallel.
6. We call the dfs() function with the starting node, which performs the depth-first search algorithm in parallel.
7. Finally, we print out the list of visited nodes in the order they were visited.

The parallelization of the DFS algorithm is achieved by using the OpenMP parallel for construct inside the dfs() function to explore the neighbours of each node in parallel. This allows multiple threads to work on different parts of the graph simultaneously, which can lead to significant performance gains on multi-core CPUs

step-by-step explanation of how this program works:

dfs function

This function performs a parallel depth-first search starting from a given node. It takes the node number as a parameter and modifies the global visited array to keep track of which nodes have been visited.

1. It starts by creating an empty stack and pushing the starting node onto it.

```c++
void dfs(int node) {
    stack<int> s;
    s.push(node);
```

2. It enters a loop that continues until the stack is empty.

```c++
    while (!s.empty()) {
        int curr_node = s.top();
        s.pop();
```

3. For each node that is popped off the stack, it checks if that node has already been visited. If not, it sets the visited flag for that node to true and adds its unvisited neighbors to the stack in parallel.

```c++
        if (!visited[curr_node]) {
            visited[curr_node] = true;

            #pragma omp parallel for
            for (int i = 0; i < graph[curr_node].size(); i++) {
                int adj_node = graph[curr_node][i];
                if (!visited[adj_node]) {
                    s.push(adj_node);
                }
            }
        }
    }
}
```

Note that the #pragma omp parallel for directive is used to parallelize the loop that adds neighboring nodes to the stack. This can speed up the execution of the program on multi-core machines.

main function

This function reads in the input, initializes the visited array, calls the dfs function, and prints out the list of visited nodes.

1. It reads in the number of nodes, the number of edges, and the starting node from standard input.

```c++
int main() {
    int n, m, start_node;
    cin >> n >> m >> start_node;
```

2. It reads in the edge list and builds the adjacency list representation of the graph. Note that each edge is added to both the u node's and v node's adjacency list to represent an undirected graph.

```c++
    for (int i = 0; i < m; i++) {
        int u, v;
        cin >> u >> v;
        graph[u].push_back(v);
        graph[v].push_back(u);
    }
```

3. It initializes the visited array to false in parallel.

```c++
    #pragma omp parallel for
    for (int i = 0; i < n; i++) {
        visited[i] = false;
    }
```

4. It calls the dfs function with the starting node as the argument.

```c++
    dfs(start_node);
```

5.It prints out the list of visited nodes in ascending order.

```c++
    for (int i = 0; i < n; i++) {
        if (visited[i]) {
            cout << i << " ";
        }
    }

    return 0;
}
```

Output:

here's an example input and output for a small graph with 6 nodes and 5 edges:

Input:

```
6 7 0
0 1
0 2
1 3
2 4
2 5
4 5
5 3
```

Output:

```
0 1 2 4 5 3
```

Assignment 1 questions
1. What if BFS?
2. What is OpenMP?What is its significance in parallel programming?
3. Write down applications of Parallel BFS
4. How can BFS be parallelized using OpenMP? Describe the parallel BFS algorithm using OpenMP.
5. Write Down Commands used in OpenMP?
6. What if DFS?
7. Write a parallel Depth First Search (DFS) algorithm using OpenMP
8. What is the advantage of using parallel programming in DFS?
9. How can you parallelize a DFS algorithm using OpenMP?
10. What is a race condition in parallel programming, and how can it be avoided in OpenMP?

Assignment No: 2

2. **Write a program to implement Parallel Bubble Sort and Merge sort using OpenMP. Use existing algorithms and measure the performance of sequential and parallel algorithms.**

Objective of the Assignment: Students should be able to Write a program to implement Parallel Bubble Sort and can measure the performance of sequential and parallel algorithms.

Prerequisite:

1. Basic of programming language
2. Concept of Bubble Sort

Concept of Parallelism

What is Bubble Sort?

Bubble Sort is a simple sorting algorithm that works by repeatedly swapping adjacent elements if they arein the wrong order. It is called "bubble" sort because the algorithm moves the larger elements towards theend of the array in a manner that resembles the rising of bubbles in a liquid.

The basic algorithm of Bubble Sort is as follows:

1. Start at the beginning of the array.
2. Compare the first two elements. If the first element is greater than the second element, swap them.
3. Move to the next pair of elements and repeat step 2.
4. Continue the process until the end of the array is reached.
5. If any swaps were made in step 2-4, repeat the process from step 1.

The time complexity of Bubble Sort is O(n^2), which makes it inefficient for large lists. However, it has the advantage of being easy to understand and implement, and it is useful for educational purposes and for sorting small datasets.

BubbleSorthaslimitedpractical use in modern software development due to its inefficient time complexity of O(n^2) which makes it unsuitable for sorting large datasets. However, Bubble Sort has some advantages and use cases that make it a valuable algorithm to understand, such as:

1. Simplicity: Bubble Sort is one of the simplest sorting algorithms, and it is easy to understand and implement. It can be used to introduce the concept of sorting to beginners and as a basis for more complex sorting algorithms.
2. Educational purposes: Bubble Sort is often used in academic settings to teach the principles of sorting algorithms and to help students understand how algorithms work.
3. Small datasets: For very small datasets, Bubble Sort can be an efficient sorting algorithm, as itsoverheadis relatively low.
4. Partially sorted datasets: If a dataset is already partially sorted, Bubble Sort can be very efficient. Since Bubble Sort only swaps adjacent elements that are in the wrong order, it has a low numberof operations for a partially sorted dataset.
5. Performance optimization: Although Bubble Sort itself is not suitable for sorting large datasets, some of its techniques can be used in combination with other sorting algorithms to optimize their performance. For example, Bubble Sort can be used to optimize the performance of InsertionSort by reducing the number of comparisons needed
6. Example of Bubble sort

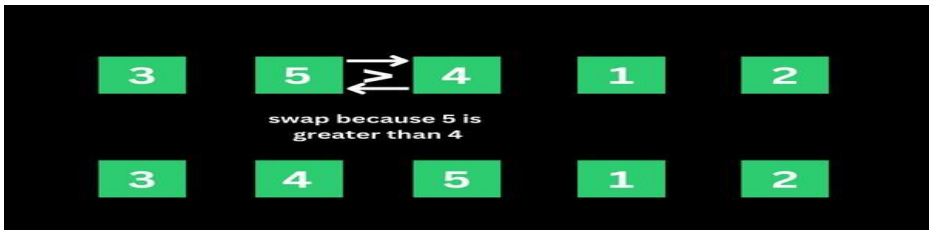
Let's say we want to sort a series of numbers 5, 3, 4, 1, and 2 so that they are arranged in ascendingorder…

The sorting begins the first iteration by comparing the first two values. If the first value is greater than thesecond, the algorithm pushes the first value to the index of the second value.

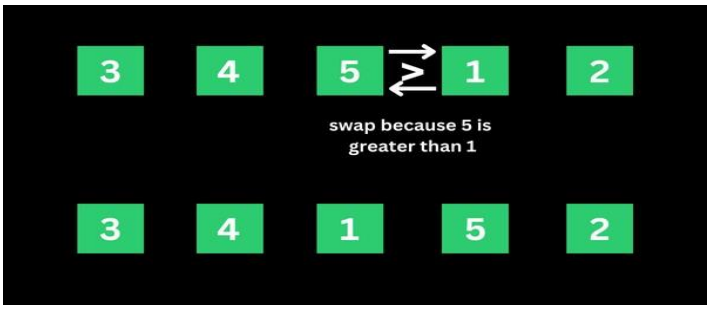
First Iteration of the Sorting

Step 1: In the case of 5, 3, 4, 1, and 2, 5 is greater than 3. So 5 takes the position of 3 and the numbers become 3, 5, 4, 1, and 2.

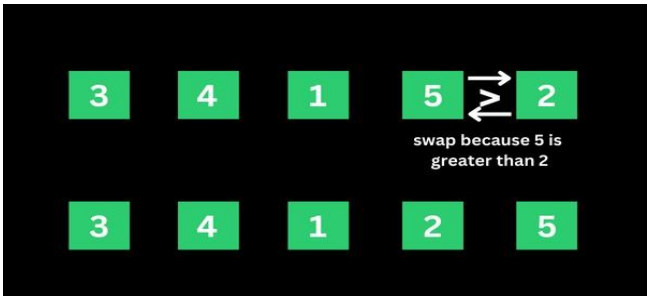
Step 2: The algorithm now has 3, 5, 4, 1, and 2 to compare, this time around, it compares the next twovalues,whichare5 and4.5 isgreaterthan4, so5takes theindexof 4andthevalues nowbecome3, 4,5,
1, and 2.



Step 3:The algorithm now has 3, 4, 5, 1, and 2 tocompare. It compares the next two values, which aand 1. 5 is greater than 1, so 5 takes the index of 1 and the numbers become 3, 4, 1, 5, and 2.



Step 4:The algorithm now has 3, 4, 1, 5, and 2 tocompare. It compares the next two values, which are 5 and 2. 5 is greater than 2, so 5 takes the index of 2 and the numbers become 3, 4, 1, 2, and 5.



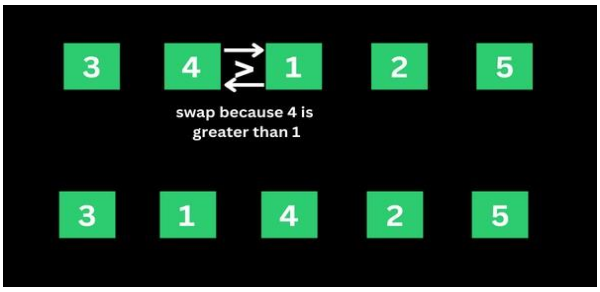
That’sthefirstiteration.Andthenumbersarenowarrangedas3,4,1,2,and5–fromtheinitial5,3,4,1,and2.Asyou

mightrealize,5 should be the last number if the numbers are sorted in ascending order.

This means the first iteration is really completed.

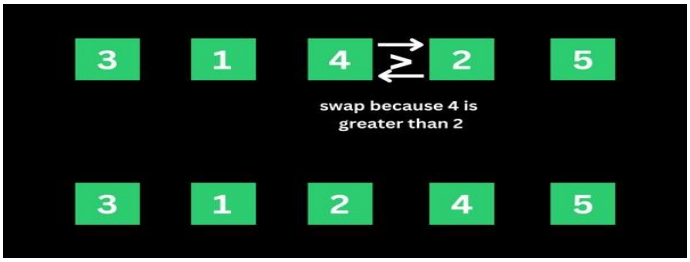
SecondIterationoftheSortingandtheRest

Thealgorithmstartsthesecond iterationwiththelastresult of3,4,1,2, and5.Thistime around,3issmaller than 4, so no swapping happens.This mans the numbers will remain the same.The algorithm proceeds to compare 4 and

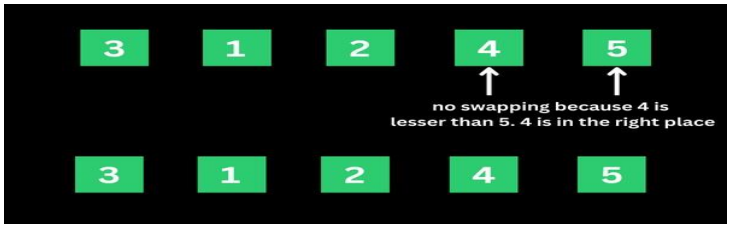


1. 4 is greater than 1, so 4 is swapped for 1 and the numbersbecome 3, 1, 4, 2, and 5.

The algorithm now proceeds to compare 4 and 2. 4 is greater than 2, so 4 is swapped for 2 and



thenumbers become



3, 1, 2, 4, and 5now in the right place, so no swapping occurs between 4 and 5 because 4 is smaller than 5.

That’showthealgorithmcontinuestocomparethenumbersuntiltheyarearrangedinascendingorderof1, 2, 3, 4,

and 5.

Concept of OpenMP

- OpenMP (Open Multi-Processing) is an application programming interface (API) that supportsshared-memoryparallelprogramminginC,C++,andFortran.Itisusedtowriteparallelprograms that can run on multicore processors, multiprocessor systems, and parallel computingclusters.
- OpenMP provides a set of directives and functions that can be inserted into the source code of aprogram to parallelize its execution. These directives are simple and easy to use, and they can beapplied to loops, sections, functions, and other program constructs. The compiler then generatesparallelcode that can runon multiple processors concurrently.
- OpenMPprogramsaredesignedtotakeadvantage of the shared-memory architecture of moder

Program: Write a program to implement Parallel Bubble Sort using OpenMP

```
#include<iostream>
#include<stdlib.h>
#include<omp.h>
using namespace std;

void bubble(int *, int);
void swap(int &, int &);

void bubble(int *a, int n)
{
for(  int i = 0;  i < n;  i++ )
{
int first = i % 2;
#pragma omp parallel for shared(a,first)
for(  int j = first;  j < n-1;  j += 2  )
{
```

```
if(  a[ j ]  >  a[ j+1 ]  )
{
swap(  a[ j ],  a[ j+1 ]  );
}
}
}
}

void swap(int &a, int &b)
{
int test;
test=a;
a=b;
b=test;

}
int main()
{
int *a,n;
cout<<"\n enter total no of elements=>";
cin>>n;
a=new int[n];
cout<<"\n enter elements=>";
for(int i=0;i<n;i++)
{
cin>>a[i];
}

bubble(a,n);

cout<<"\n sorted array is=>";
for(int i=0;i<n;i++)
```

```
{
cout<<a[i]<<endl;
}

return 0;
}
```

Explanation:

Here, we're using the #pragma omp parallel directive to parallelize the code using OpenMP. Within the parallel block, we have a while loop that runs until the array is sorted. Inside the while loop, we have a #pragma omp for directive that parallelizes the for loop that performs the bubble sort.

Note that the flag variable is used to keep track of whether any swaps were made during an iteration of the for loop. If no swaps were made, then the array is already sorted and we can exit the loop.

explanation of each line of the code:

```c++
#include <iostream>
#include <omp.h>

using namespace std;
```

These lines are the standard C++ header files used for input/output operations and OpenMP, respectively.

```c++
void parallel_bubble_sort(int arr[], int n) {
    #pragma omp parallel
    {
        int i, temp, flag = 1;
        while (flag) {
            flag = 0;
            #pragma omp for
            for (i = 0; i < n-1; i++) {
                if (arr[i] > arr[i+1]) {
                    temp = arr[i];
                    arr[i] = arr[i+1];
                    arr[i+1] = temp;
                    flag = 1;
                }
            }
        }
    }
}
```

This is the function parallel_bubble_sort() that implements the parallel bubble sort algorithm using OpenMP.

- #pragma omp parallel is a directive that creates a team of threads to execute the parallel code inside the block. In this case, the block contains the code for bubble sort algorithm.
- int i, temp, flag = 1; declares the variables i, temp, and flag that will be used inside the while loop.
- while (flag) is a loop that runs until the flag variable is 0.
- flag = 0; sets the flag variable to 0 before starting each iteration of the for loop.
- #pragma omp for is a directive that parallelizes the for loop, by dividing the loop iterations among the threads in the team. Each thread performs the sorting operation on a subset of the array, thereby making the sorting process faster.
- for (i = 0; i < n-1; i++) is a for loop that iterates over the array, from 0 to n-1.
- if (arr[i] > arr[i+1]) checks if the current element is greater than the next element.
- temp = arr[i]; arr[i] = arr[i+1]; arr[i+1] = temp; swaps the current element with the next element, using a temporary variable.
- flag = 1; sets the flag variable to 1, indicating that a swap has been made.
- Finally, the sorted array is printed using a for loop.

```c++
int main() {
    int arr[] = {5, 3, 1, 9, 8, 2, 4, 7, 6};
    int n = sizeof(arr)/sizeof(arr[0]);

    parallel_bubble_sort(arr, n);

    cout << "Sorted array: ";
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

This is the main() function, which initializes an array arr and its size n. The function parallel_bubble_sort() is called with these arguments to sort the array. The sorted array is then printed to the console using a for loop.

How to Run code in Ubuntu:

1. **Open a terminal window.**

Compile the code using the following command:

```
g++ -fopenmp filename.cpp -o executable
```

Run the program by executing the following command:

```bash
./executable
```

Output

```c
Sorted array: 1 2 3 4 5 6 7 8 9
```

This is because the input array `{5, 3, 1, 9, 8, 2, 4, 7, 6}` is sorted in ascending order using the parallel bubble sort algorithm implemented in the `parallel_bubble_sort()` function. The sorted array is then printed to the console in the `main()` function using a for loop.

To measure the performance of sequential Bubble sort and parallel Bubble sort algorithms, you can follow these steps:

1. Implement both the sequential and parallel Bubble sort algorithms.
2. Choose a range of test cases, such as arrays of different sizes and different degrees of sortedness, to test the performance of both algorithms.
3. Use a reliable timer to measure the execution time of each algorithm on each test case.
4. Record the execution times and analyze the results.

When measuring the performance of the parallel Bubble sort algorithm, you will need to specify the number of threads to use. You can experiment with different numbers of threads to find the optimal value for your system.

Here are some additional tips for measuring performance:

- Run each algorithm multiple times on each test case and take the average execution time to reduce the impact of variations in system load and other factors.

- Monitor system resource usage during execution, such as CPU utilization and memory consumption, to detect any performance bottlenecks.
- Visualize the results using charts or graphs to make it easier to compare the performance of the two algorithms.

How to check CPU utilisation and memory consumption in ubuntu

In Ubuntu, you can use a variety of tools to check CPU utilization and memory consumption. Here are some common tools:

1. top: The top command provides a real-time view of system resource usage, including CPU utilization and memory consumption. To use it, open a terminal window and type top. The output will display a list of processes sorted by resource usage, with the most resource-intensive processes at the top.
2. htop: htop is a more advanced version of top that provides additional features, such as interactive process filtering and a color-coded display. To use it, open a terminal window and type htop.
3. ps: The ps command provides a snapshot of system resource usage at a particular moment in time. To use it, open a terminal window and type ps aux. This will display a list of all running processes and their resource usage.
4. free: The free command provides information about system memory usage, including total, used, and free memory. To use it, open a terminal window and type free -h.
5. vmstat: The vmstat command provides a variety of system statistics, including CPU utilization, memory usage, and disk activity. To use it, open a terminal window and type vmstat.

processors,wheremultipleprocessorcorescanaccessthesamememory.OpenMPusesafork-join model of parallel execution, where a master thread forks multiple worker threads to execute a parallel region of the code, and then waits for all threads to complete before continuing with the sequential part of the code.

How Parallel Bubble Sort Work

- Parallel Bubble Sort is a modification of the classic Bubble Sort algorithm that takes advantage ofparallel processing to speed up the sorting process.

- InparallelBubbleSort,thelistofelementsisdividedintomultiplesublists that are sortedconcurrentlybymultiplethreads.Eachthreadsortsitssublistusing the regular Bubble Sort

algorithm. When all sublists have been sorted, they are merged together to form the final sortedlist.

- The parallelization of the algorithm is achieved using OpenMP, a programming API that supportsparallel processing in C++, Fortran, and other programming languages. OpenMP provides a set ofcompiler directives that allow developers to specify which parts of the code can be executed inparallel.

- Inthe parallel Bubble Sort algorithm, the main loop that iterates over the list of elements isdivided into multiple iterations that are executed concurrently by multiple threads. Each threadsorts a subset of the list, and the threads synchronize their work at the end of each iteration toensure that the elements are properly ordered.

- Parallel Bubble Sort can provide a significant speedup over the regular Bubble Sort algorithm, especially when sorting large datasets on multi-core processors. However, the speedup is limited by the overhead of thread creation and synchronization, and it may not be worth the effort for small datasets or when using a single-core processor.

How to measure the performance of sequential and parallel algorithms?

TomeasuretheperformanceofsequentialBubblesortandparallelBubblesortalgorithms, you can follow these steps:

1. Implement both the sequential and parallel Bubble sort algorithms.
2. Choosearangeoftestcases,suchasarraysofdifferentsizesanddifferentdegreesofsortedness,to test the performance of both algorithms. Use a reliable timer to measure the execution time of each algorithm on each test case.
3. Record the execution times and analyze the results.

When measuring the performance of the parallel Bubble sort algorithm, you will need to specify the number of threads to use. You can experiment with different numbers of threads to find the optimal value for your system.

Here are some additional tips for measuring performance:

- Run each algorithm multiple times on each test case and take the average execution time to reduce the impact of variations in system load and other factors.
- Monitor system resource usage during execution, such as CPU utilization and memory consumption, to detect any performance bottlenecks.
- Visualizetheresultsusingchartsorgraphstomakeiteasiertocomparetheperformanceofthetwo algorithms.

How to check CPU utilization and memory consumption in ubuntu

In Ubuntu, you can use a variety of tools to check CPU utilization and memory consumption. Here are some common tools:

1. **top:**Thetopcommandprovidesareal-timeviewofsystemresource usage, including CPU utilization and memory consumption. To use it, open a terminal window and type top. The outputwilldisplayalistofprocessessortedbyresourceusage,withthemostresource-intensiveprocesses at the top.
2. **htop**: htop is a more advanced version of top that provides additional features, such as interactiveprocessfilteringandacolor-codeddisplay.Touseit,openaterminalwindowandtypehtop.
3. **ps**: The ps command provides a snapshot of system resource usage at a particular moment in time.To use it, open a terminal window and type ps aux. This will display a list of all running processesand their resource usage.
4. **free:** The free command provides information about system memory usage, including total, used,andfreememory.Touse it,open aterminal windowand typefree-h.
5. **vmstat:** The vmstat command provides a variety of system statistics, including CPU utilization, memory usage ,and disk activity. To use it,open a terminal window and type vmstat.
6. **Conclusion**- InthiswaywecanimplementBubbleSortinparallelwayusingOpenMPalsocome to know how to how to measure performance of serial and parallel algorithm

Title of the Assignment: Write a program to implement Parallel Merge Sort. Use existing algorithms and measure the performance of sequential and parallel algorithms.

Objective of the Assignment: Students should be able to Write a program to implement Parallel Merge Sort and can measure the performance of sequential and parallel algorithms.

Prerequisite:

1. Basic of programming language
2. Concept of Merge Sort
3. Concept of Parallelism

What is Merge Sort?

Merge sort is a sorting algorithm that uses a divide-and-conquer approach to sort an array or a list of elements. The algorithm works by recursively dividing the input array into two halves, sorting each half,andthen merging the sorted halves to produce a sorted output.

The merge sort algorithm can be broken down into the following steps:

1. Divide the input array into two halves.
2. Recursivelysortthelefthalfofthearray.
3. Recursivelysorttherighthalfofthearray.
4. Mergethetwosortedhalvesintoasinglesortedoutputarray.

- Themergingstepiswhere the bulk of the work happens in merge sort. The algorithm comparesthe first elements of each sorted half, selects the smaller element, and appends it to the outputarray. This process continues until all elements from both halves have been appended to the outputarray.
- The time complexity of merge sort is O(n log n), which makes it an efficient sorting algorithm for large input arrays. However, merge sort also requires additional memory to store the output array, which can make it less suitable for use with limited memory resources.
- In simple terms, we can say that the process of merge sort is to divide the array into two halves, sort each half, and then merge the sorted halves back together. This process is repeated until the entire array is sorted.
- One thing that you might wonder is what is the specialty of this algorithm. We already have number of sorting algorithms then why do we need this algorithm? One of the main advantages of merge sort is that it has a time complexity of O(n log n), which means it can sort large arraysrelatively quickly. It is also a stable sort, which means that the order of elements with equal

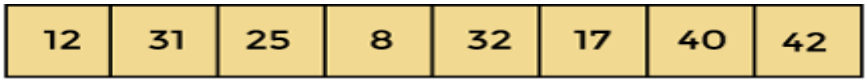
valuesis preserved during the sort.

- Mergesortisapopularchoiceforsortinglargedatasetsbecauseitisrelativelyefficientand easyto implement. It is often used in conjunction with other algorithms, such as quicksort, to improvethe overall performance of a sorting routine.

Example of Merge sort

Now,let's see the workingofmergesortAlgorithm.Tounderstandtheworkingofthemergesort

algorithm,let'stakeanunsortedarray.Itwillbeeasiertounderstandthemergesortviaanexample.Let the elements of array are -

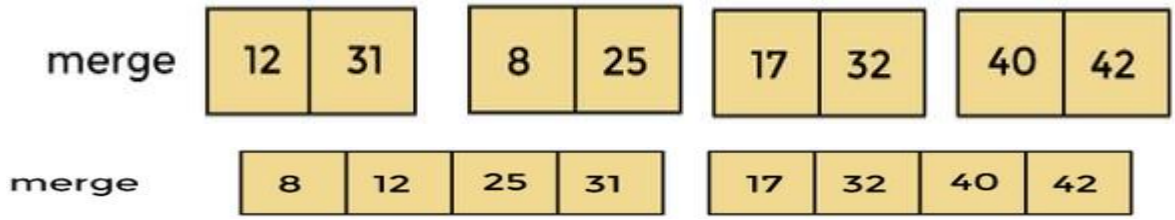


- Accordingtothemergesort,firstdividethegivenarrayintotwoequalhalves.Mergesortkeepsdividing the list into equal parts until it cannot be further divided.
- Astherareeightelementsin thegivenarray,soitis dividedintotwoarraysofsize 4.

- Now,again dividethesetwoarraysintohalves.Astheyareofsize4,dividethemintonewarraysof size 2.
- Now,againdividethesearraystogettheatomicvaluethatcannotbefurtherdivided.

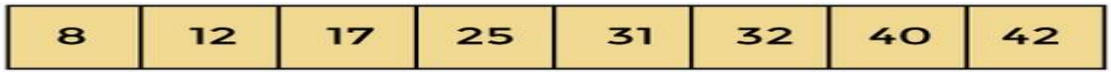
- Now, combine them in the same manner they were broken.
- In combining, first compare the element of each array and then combine them into another array in sorted order.

- So,firstcompare12and31,both are in sorted positions. Then compare 25and8,and in the list of two values, put 8 first followed by 25. Then compare 32 and 17, sort them and put 17 first followedby 32.After that, compare40 and 42,and place them sequentially.

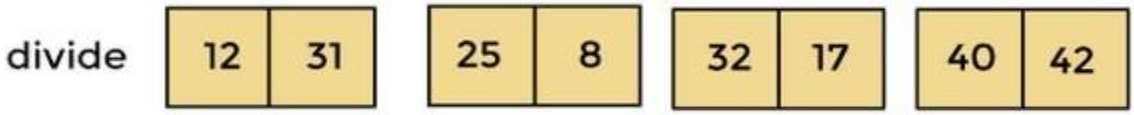


Then extiterationofcombining,nowcomparethearrayswithtwodatavaluesandmergethemintoan array of found values insorted order.

- Now, there is a final merging of the arrays. After the final merging of above arrays, the array will look like -



Concept of OpenMP



- OpenMP (Open Multi-Processing) is an application programming interface (API) that supportsshared-memoryparallelprogramminginC,C++,andFortran.Itisusedtowriteparallelprograms that can run on multicore processors, multiprocessor systems, and parallel computingclusters.
- OpenMP provides a set of directives and functions that can be inserted into the source code of aprogram to parallelize its execution. These directives are simple and easy to use, and they can beapplied to loops, sections, functions, and other program constructs. The compiler then generatesparallelcode that can runon multiple processors concurrently.
- OpenMP programs are designed to take advantage of the shared-memory architecture of

modernprocessors,wheremultipleprocessorcorescanaccessthesamememory.OpenMPusesafork-join model of parallel execution, where a master thread forks multiple worker threads toexecute a parallel region of the code, and then waits for all threads to complete before continuingwith the sequential part of the code.

HowParallelMergeSortWork

- Parallel merge sort is a parallelized version of the merge sort algorithm that takes advantage ofmultiple processors or cores to improve its performance. In parallel merge sort, the input array isdivided into smaller subarrays, which are sorted in parallel using multiple processors or cores.Thesorted subarrays are thenmerged together in parallel toproduce the final sorted output.
- The parallel mergesort algorithm canbe broken downinto thefollowing steps:

the input array into smaller subarrays.

- Assign each subarray to a separate processor or core for sorting.
- Sorteach subarrayin parallelusing themerge sortalgorithm.
- Mergethe sortedsubarrays together inparallel to producethe finalsorted output.
- The merging step in parallel merge sort is performed in a similar way to the merging step in thesequentialmergesortalgorithm.However,becausethesubarraysaresorted in parallel, themerging step can also be performed in parallel using multiple processors or cores. This cansignificantlyreduce the timerequired to mergethe sorted subarraysand produce thefinal output.
- Parallel merge sort can provide significant performance benefits for large input arrays with manyelements, especially when running on hardware with multiple processors or cores. However, italso requires additional overhead to manage the parallelization, and may not always provideperformance improvements for smaller input sizes or when run on hardware with limited parallelprocessing capabilities.

How to measure the performance of sequential and parallel algorithms?

Thereareseveralmetricsthatcanbeusedtomeasuretheperformanceofsequentialandparallelmergesort algorithms:

1. **Execution time:** Execution time is the amount of time it takes for the algorithm to complete itssortingoperation.Thismetriccanbeusedtocomparethespeedofsequentialandparallelmergesort algorithms.
2. **Speedup**: Speedup is the ratio of the execution time of the sequential merge sort algorithm to theexecutiontimeoftheparallelmergesortalgorithm.Aspeedupofgreaterthan1indicatesthattheparallel algorithm is faster than the sequential algorithm.
3. **Efficiency:** Efficiency is the ratio of the speedup to the numberofprocessorsorcoresusedintheparallel algorithm. This metric can be used to determine how well the parallel algorithm is utilizing the available resources.
4. **Scalability**: Scalability is the ability of the algorithm to maintain its performance as the input size and number of processors or cores increase. A scalable algorithm will maintain a consistent Speedup and efficiency as more resources are added.

To measure the performance of sequential and parallel merge sort algorithms, you can perform

experiments on different input sizes and numbers of processors or cores. By measuring the execution time, speedup, efficiency, and scalabilityofthealgorithmsunderdifferentconditions,youcandeterminehich algorithm is more efficient fordifferentinputsizesandhardwareconfigurations.Additionally,youcan use profiling tools to analyze the performance of the algorithms and identify areas for optimization

Conclusion-In this way we can implement Merge Sort in parallel way using OpenMP also come to know how to how to measure performance of serial and parallel algorithm

Program: Merge sort using OpenMP. Use existing algorithms and measure the performance of sequential and parallel algorithms.

Program:

```
#include<iostream>
#include<stdlib.h>
#include<omp.h>
using namespace std;


void mergesort(int a[],int i,int j);
void merge(int a[],int i1,int j1,int i2,int j2);

void mergesort(int a[],int i,int j)
{
        int mid;
        if(i<j)
        {
        mid=(i+j)/2;

        #pragma omp parallel sections
        {

        #pragma omp section
        {
                mergesort(a,i,mid);
        }

        #pragma omp section
        {
                mergesort(a,mid+1,j);
        }
        }

        merge(a,i,mid,mid+1,j);
        }
```

```
}

void merge(int a[],int i1,int j1,int i2,int j2)
{
        int temp[1000];
        int i,j,k;
        i=i1;
        j=i2;
        k=0;

        while(i<=j1 && j<=j2)
        {
        if(a[i]<a[j])
        {
        temp[k++]=a[i++];
        }
        else
        {
        temp[k++]=a[j++];
   }
        }

        while(i<=j1)
        {
        temp[k++]=a[i++];
        }

        while(j<=j2)
        {
        temp[k++]=a[j++];
        }

        for(i=i1,j=0;i<=j2;i++,j++)
        {
        a[i]=temp[j];
        }
}


int main()
{
        int *a,n,i;
        cout<<"\n enter total no of elements=>";
        cin>>n;
        a= new int[n];

        cout<<"\n enter elements=>";
        for(i=0;i<n;i++)
        {
```

```
        cin>>a[i];
        }
  //      start=.....
//#pragma omp…..
        mergesort(a, 0, n-1);
//          stop…….
        cout<<"\n sorted array is=>";
        for(i=0;i<n;i++)
        {
        cout<<"\n"<<a[i];
        }
        // Cout<<Stop-Start
        return 0;
}
```

Second Code:

```
#include <iostream>
#include <omp.h>

void merge(int* arr, int l, int m, int r) {
        int i, j, k;
        int n1 = m - l + 1;
        int n2 = r - m;

        int L[n1], R[n2];

        for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
        for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];

        i = 0;
        j = 0;
        k = l;
        while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
        arr[k] = L[i];
        i++;
        }
        else {
        arr[k] = R[j];
        j++;
        }
        k++;
        }

        while (i < n1) {
        arr[k] = L[i];
        i++;
```

```
        k++;
        }

        while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
        }
}

void mergeSort(int* arr, int l, int r) {
        if (l < r) {
        int m = l + (r - l) / 2;
        #pragma omp parallel sections
        {
        #pragma omp section
        {
                mergeSort(arr, l, m);
        }
        #pragma omp section
        {
                mergeSort(arr, m + 1, r);
        }
        }

        merge(arr, l, m, r);
        }
}

int main() {
        int arr[] = { 12, 11, 13, 5, 6, 7 };
        int n = sizeof(arr) / sizeof(arr[0]);
  double start, stop;
        std::cout << "Given array is: ";
        for (int i = 0; i < n; i++)
        std::cout << arr[i] << " ";
        std::cout << std::endl;
 start = omp_get_wtime();
  #pragma omp parallel
  {
        mergeSort(arr, 0, n - 1);
}
  stop = omp_get_wtime();
        std::cout << "Sorted array is: ";
        for (int i = 0; i < n; i++)
        std::cout << arr[i] << " ";
        std::cout << std::endl;

std::cout<<stop-start;
        return 0;
```

```
}
```

Explanation

```c++
#include<iostream>
#include<stdlib.h>
#include<omp.h>
using namespace std;
```

This block of code is a set of header files and a namespace declaration. iostream and stdlib.h are header files, and omp.h is the header file for OpenMP (Open Multi-Processing) library that is used for parallel programming.

```c++
void mergesort(int a[],int i,int j);
void merge(int a[],int i1,int j1,int i2,int j2);
```

These are function declarations for mergesort and merge functions.

```c++
void mergesort(int a[],int i,int j)
{
    int mid;
    if(i<j)
    {
        mid=(i+j)/2;

        #pragma omp parallel sections
        {
            #pragma omp section
            {
                mergesort(a,i,mid);
            }

            #pragma omp section
            {
                mergesort(a,mid+1,j);
            }
        }

        merge(a,i,mid,mid+1,j);
    }
}
```

This is the merge sort function, which implements the merge sort algorithm recursively. a[] is the input array that is to be sorted, i and j are the starting and ending indices of the subarray being sorted.

First, it checks whether i is less than j because if i is greater than or equal to j, it means that there is only one element in the sub array, which is already sorted. Then it calculates the midpoint of the subarray using (i+j)/2.

The #pragma omp parallel sections directive creates a parallel region that specifies two parallel sections that can be executed concurrently. The merge sort function is then recursively called on the left and right halves of the sub array in two separate parallel sections.

Finally, the merge function is called to merge the sorted halves of the subarray.

```c
void merge(int a[],int i1,int j1,int i2,int j2)
```

This is the function signature which takes an integer array a and four integer variables i1, j1, i2, and j2. i1 and j1 define the start and end indices of the first sorted subarray, and i2 and j2 define the start and end indices of the second sorted subarray.

```c
int temp[1000];
int i,j,k;
i=i1;
j=i2;
k=0;
```

Here, a temporary array temp is created with a size of 1000. Three integer variables i, j, and k are initialized. i and j are set to the start indices of the two subarrays, while k is set to 0.

```c
while(i<=j1 && j<=j2)
{
    if(a[i]<a[j])
    {
        temp[k++]=a[i++];
    }
    else
    {
        temp[k++]=a[j++];
    }
}
```

This is a while loop that runs as long as i is less than or equal to j1 and j is less than or equal to j2. Inside the loop, if the element at index i of the first subarray is less than the element at index j of the second subarray, then the element at index i is copied to the temp array at index k, and i and k are incremented. Otherwise, the element at index j is copied to the temp array at index k, and j and k are incremented.

```
while(i<=j1)
{
    temp[k++]=a[i++];
}

while(j<=j2)
{
    temp[k++]=a[j++];
}
```

After the above loop terminates, there may be some elements left in one of the subarrays. These loops copy the remaining elements into the temp array.

```
for(i=i1,j=0;i<=j2;i++,j++)
{
    a[i]=temp[j];
}
```

Finally, the sorted temp array is copied back to the original a array. The loop runs from i1 to j2 and copies the elements of temp array to the corresponding indices in the a array. The loop variable j starts from 0 and increments alongside.

Questions

1. What is Bubble sort?
2. What are advantages to use bubble sort
3. How parallel Bubble sort work explain it with example
4. What is merge sort?
5. How parallel merge sort work explain with example
6. What are different metrics to measure the performance of merge sort?

Assignment No: 3

3.Implement Min, Max, Sum and Average operations using Parallel Reduction.

Void Min_reduction()

- void min_reduction(vector<int>& arr) declares a void function that takes a reference to an integer vector as its argument.
- int min_value = INT_MAX; initializes an integer variable min_value to the largest possible integer value using the INT_MAX constant from the <climits> header file. This is done to ensure that min_value is initially greater than any element in arr.
- #pragma omp parallel for reduction(min: min_value) is an OpenMP directive that specifies that the following loop should be executed in parallel using multiple threads. The reduction(min: min_value) clause indicates that each thread should maintain a private copy of min_value and update it with the minimum value it finds in its portion of the loop. Once the loop is complete, OpenMP will combine all the private copies of min_value into a single shared value that represents the minimum value in arr.
- for (int i = 0; i < arr.size(); i++) { is a loop that iterates over each element of arr.
- if (arr[i] < min_value) { min_value = arr[i]; } checks if the current element of arr is less than min_value. If so, it updates min_value to be the current element.
- cout<< "Minimum value: " << min_value << endl; prints out the minimum value found in arr.

void max_reduction()

- void max_reduction(vector<int>& arr) declares a void function that takes a reference to an integer vector as its argument.
- int max_value = INT_MIN; initializes an integer variable max_value to the smallest possible integer value using the INT_MIN constant from the <climits> header file. This is done to ensure that max_value is initially smaller than any element in arr.
- #pragma omp parallel for reduction(max: max_value) is an OpenMP directive that specifies that the following loop should be executed in parallel using multiple threads. The reduction(max: max_value) clause indicates that each thread should maintain a private copy of max_value and update it with the

maximum value it finds in its portion of the loop. Once the loop is complete, OpenMP will combine all the private copies of max_value into a single shared value that represents the maximum value in arr.

- for (int i = 0; i < arr.size(); i++) { is a loop that iterates over each element of arr.
- if (arr[i] > max_value) { max_value = arr[i]; } checks if the current element of arr is greater than max_value. If so, it updates max_value to be the current element.
- cout<< "Maximum value: " << max_value << endl; prints out the maximum value found in arr.

#include <climits>

<climits> is a header file in C++ that contains constants related to integer types. This header file provides implementation-defined constants for minimum and maximum values of integral types, such as INT_MAX (maximum value of int) and INT_MIN (minimum value of int).

Using these constants instead of hard coding the values of the minimum and maximum integer values is a good practice because it makes the code more readable and avoids the possibility of introducing errors in the code. The use of these constants also ensures that the code will work correctly across different platforms and compilers.

INT_MIN :

Minimum value for an object of type int

Value of INT_MIN is -32767 ($-2^{15}+1$) or less*

INT_MAX :

Maximum value for an object of type int

Value of INT_MAX is 2147483647 (-2^{31} to $2^{31}-1$)

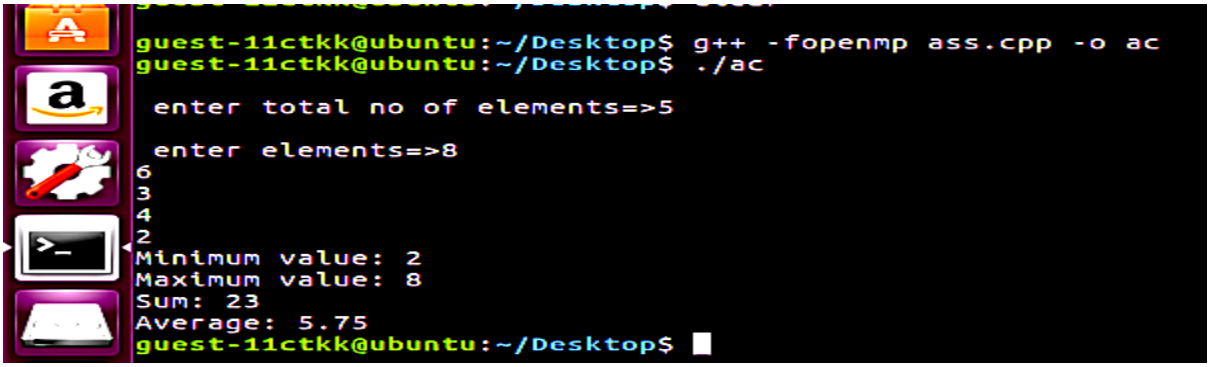
Program: Implement Min, Max, Sum and Average operations using Parallel Reduction.

```
#include <iostream>
//#include <vector>
#include <omp.h>
#include <climits>
using namespace std;
void min_reduction(int arr[], int n) {
int min_value = INT_MAX;
```

```
  #pragma omp parallel for reduction(min: min_value)
for (int i = 0; i < n; i++) {
        if (arr[i] < min_value) {
        min_value = arr[i];
        }
  }
cout<< "Minimum value: " << min_value << endl;
}

void max_reduction(int arr[], int n) {
int max_value = INT_MIN;
  #pragma omp parallel for reduction(max: max_value)
for (int i = 0; i < n; i++) {
        if (arr[i] > max_value) {
        max_value = arr[i];
        }
  }
cout<< "Maximum value: " << max_value << endl;
}

void sum_reduction(int arr[], int n) {
int sum = 0;
   #pragma omp parallel for reduction(+: sum)
for (int i = 0; i < n; i++) {
        sum += arr[i];
  }
cout<< "Sum: " << sum << endl;
}

void average_reduction(int arr[], int n) {
int sum = 0;
  #pragma omp parallel for reduction(+: sum)
for (int i = 0; i < n; i++) {
        sum += arr[i];
  }
cout<< "Average: " << (double)sum / (n-1) << endl;
}

int main() {
int *arr,n;
cout<<"\n enter total no of elements=>";
cin>>n;
arr=new int[n];
cout<<"\n enter elements=>";
for(int i=0;i<n;i++)
    {
        cin>>arr[i];
    }

//   int arr[] = {5, 2, 9, 1, 7, 6, 8, 3, 4};
```

```
//   int n = size(arr);

  min_reduction(arr, n);
  max_reduction(arr, n);
  sum_reduction(arr, n);
  average_reduction(arr, n);
}
```

Outcome:



Questions

1. What are the benefits of using parallel reduction for basic operations on large arrays?
2. How does OpenMP's "reduction" clause work in parallel reduction?
3. How do you set up a C++ program for parallel computation with OpenMP?
4. What are the performance characteristics of parallel reduction, and how do they vary based on input size?
5. How can you modify the provided code example for more complex operations using parallel reduction?

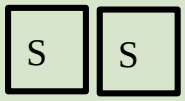
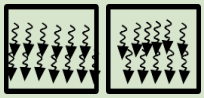
Assignment No: 4

Title: CUDA Program for : Addition of two large vectors and Matrix Multiplication using CUDA

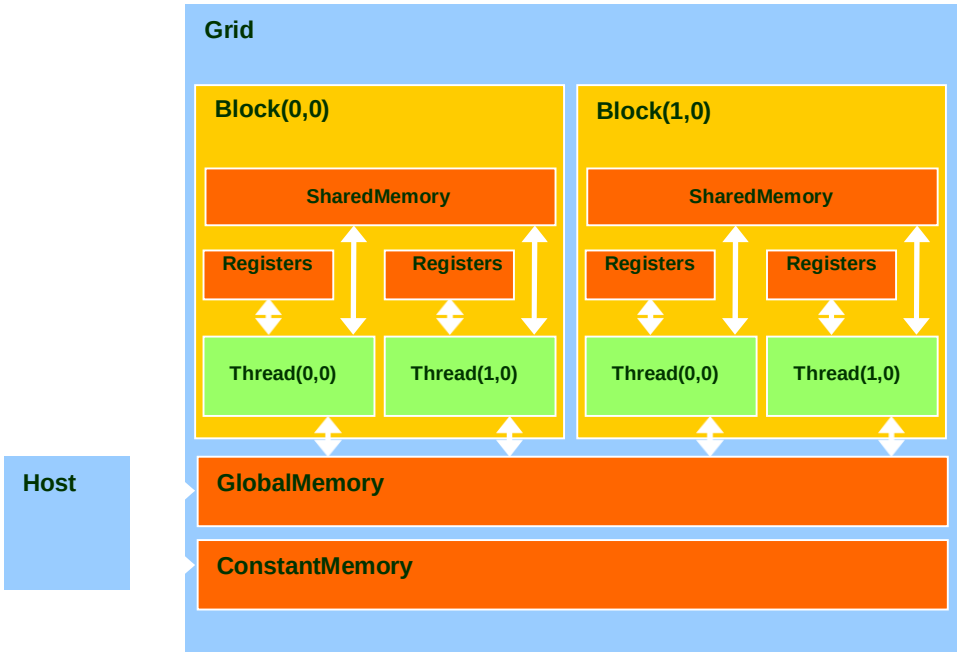
Objective: 1. Addition of two large vectors

2. Matrix Multiplication using CUDA

Concept:

	Scal	Vect	Cor	Car
Hardwa				
	ALUU	U	S	GP
Threa				
	Threa	War	ThreadBlo	BlockGri
Memo	RegisterFi		ck	d
Addre ssSpa	Localperthre		Share dMem	Glob

Programmer View of CUDA Memories:



DeclaringCUDAVariables:

Variabledeclaration	Memory	Scope	Lifetime
intLocalVar;	register	thread	thread
device_____shared_____intSharedVar;	shared	block	block
device_____ intGlobalVar;	global	grid	application
device_____constant_____intConstantVar;	constant	grid	application

- **device**_____isoptionalwhenusedwith_____**shared**____,or____**constant**_____
- Automaticvariablesresideinaregister
  - –Except per-threadarraysthatresidein global memory

Shared Memory in CUDA:

- A special type of memory whose contents are explicitly defined andusedinthekernelsourcecode
  - OneineachSM
  - Accessedatmuchhigherspeed(inbothlatencyandthroughput)thanglobalmemory
  - Scopeofaccessandsharing-threadblocks
  - Lifetime– threadblock,contentswilldisappearafterthecorrespondingthreadfinishesterminate sexecution
  - Accessedbymemoryload/storeinstructions
  - Aformofscratchpadmemoryincomputerarchitecture

Program: 1) **Addition of two large vectors**

```
#include<iostream>
#include<cstdlib>
usingnamespacestd;
//VectorAdd parallel function
__global__voidvectorAdd(int *a, int *b, int *result, int n)
{
int tid=threadIdx.x+blockIdx.x*blockDim.x;
if(tid<n)
   {
      result[tid]=a[tid]+b[tid];
   }
}
intmain()
{
int *a,*b,*c;
int *a_dev,*b_dev,*c_dev;
int n=1<<24;
   a=newint[n];
   b=newint[n];
   c=newint[n];
int *d=newint[n];
int size=n*sizeof(int);
cudaMalloc(&a_dev,size);
cudaMalloc(&b_dev,size);
cudaMalloc(&c_dev,size);
//Array initialization..You can use Randon function to assign values
for(int i=0;i<n;i++)
   {
      a[i]=1;
      b[i]=2;
      d[i]=a[i]+b[i]; //calculating serial addition
   }
   cudaEvent_t start,end;
cudaEventCreate(&start);
cudaEventCreate(&end);
cudaMemcpy(a_dev,a,size,cudaMemcpyHostToDevice);
cudaMemcpy(b_dev,b,size,cudaMemcpyHostToDevice);
int threads=1024;
int blocks=(n+threads-1)/threads;
cudaEventRecord(start);
//Parallel addition program
   vectorAdd<<<blocks,threads>>>(a_dev,b_dev,c_dev,n);
cudaEventRecord(end);
cudaEventSynchronize(end);
floattime=0.0;
```

```
cudaEventElapsedTime(&time,start,end);
cudaMemcpy(c,c_dev,size,cudaMemcpyDeviceToHost);
//Calculate the error term.
int error=0;
for(int i=0;i<n;i++){
      error+=d[i]-c[i];
//cout<<" gpu "<<c[i]<<" CPU "<<d[i];
   }
   cout<<"Error : "<<error;
   cout<<"\nTime Elapsed:  "<<time;
return0;
}
```

Program:

2) Matrix Multiplication using CUDA C

```
#include<iostram.h>
#include<cstdlib>
#include<cmath>
usingnamespacestd;
//Matrix multiplication Cuda
__global__voidmatrixMultiplication(int *a, int *b, int *c, int n)
{
int row=threadIdx.y+blockDim.y*blockIdx.y;
int col=threadIdx.x+blockDim.x*blockIdx.x;
int sum=0;
if(row<n && col<n)
for(int j=0;j<n;j++)
   {
      sum=sum+a[row*n+j]*b[j*n+col];
   }
   c[n*row+col]=sum;
}
intmain()
{
```

```
int *a,*b,*c;
int *a_dev,*b_dev,*c_dev;
int n=3;
   a=newint[n*n];
   b=newint[n*n];
   c=newint[n*n];
int *d=newint[n*n];
int size=n*n*sizeof(int);
cudaMalloc(&a_dev,size);
cudaMalloc(&b_dev,size);
cudaMalloc(&c_dev,size);
//Array initialization
for(int i=0;i<n*n;i++)
   {
        a[i]=2;  //rand()%n;
      b[i]=1;//rand()%n;
// d[i]=a[i]+b[i];
   }
   cudaEvent_t start,end;
cudaEventCreate(&start);
cudaEventCreate(&end);
cudaMemcpy(a_dev,a,size,cudaMemcpyHostToDevice);
cudaMemcpy(b_dev,b,size,cudaMemcpyHostToDevice);
dim3threadsPerBlock(n, n);
dim3blocksPerGrid(1, 1);
if(n*n>512){
      threadsPerBlock.x=512;
      threadsPerBlock.y=512;
      blocksPerGrid.x=ceil((double)n/(double)threadsPerBlock.x);
      blocksPerGrid.y=ceil((double)n/(double)threadsPerBlock.y);
   }
//GPU Multiplication
cudaEventRecord(start);
```

```
    matrixMultiplication<<<blocksPerGrid,threadsPerBlock>>>(a_dev,b_dev,c_dev,n);
cudaEventRecord(end);
cudaEventSynchronize(end);
floattime=0.0;
cudaEventElapsedTime(&time,start,end);
cudaMemcpy(c,c_dev,size,cudaMemcpyDeviceToHost);
//CPU matrix multiplication
int sum=0;
for(int row=0;row<n;row++)
   {
for(int col=0;col<n;col++)
      {
        sum=0;
for(int k=0;k<n;k++)
          sum=sum+a[row*n+k]*b[k*n+col];
         d[row*n+col]=sum;
      }
   }
int error=0;
for(int i=0;i<n*n;i++){
      error+=d[i]-c[i];
//cout<<" gpu "<<c[i]<<" CPU "<<d[i]<<endl;
   }
   cout<<"Error : "<<error;
   cout<<"\nTime Elapsed:  "<<time;
return0;
}
```

Questions:

1. What is the purpose of using CUDA to perform addition of two large vectors?
2. How do you allocate memory for the vectors on the device using CUDA?
3. How do you launch the CUDA kernel to perform the addition of two large vectors?
4. How can you optimize the performance of the CUDA program for adding two large vectors?

Group B: 410251: Deep Learning

Assignment No:1

Title of the Assignment: Linear regression by using Deep Neural network: Implement Boston housing price. Prediction problem by Linear regression using Deep Neural network. Use Boston House price prediction dataset.

Objective of the Assignment: Students should be able to perform Linear regression by using Deep Neural network on Boston House Dataset.

Prerequisite:

1. Basic of programming language
2. Concept of Linear Regression

3. Concept of Deep Neural Network

Contents for Theory:

1. What is Linear Regression
2. Example of Linear Regression
3. Concept of Deep Neural Network
4. How Deep Neural Network Work
5. Code Explanation with Output

What is Linear Regression?

Linear regression is a statistical approach that is commonly used to model the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship between the variables and uses mathematical methods to estimate the coefficients that best fit the data.

Deep neural networks are a type of machine learning algorithm that are modeled after the structure and function of the human brain. They consist of multiple layers of interconnected neurons that process data and learn from it to make predictions or classifications.

Linear regression using deep neural networks combines the principles of linear regression with the powerof deep learning algorithms. In this approach, the input features are passed through one or more layers ofneurons to extract features and then a linear regression model is applied to the output of the last layer tomake predictions. The weights and biases of the neural network are adjusted during training to optimizethe performance of the model.

This approach can be used for a variety of tasks, including predicting numerical values, such as stockprices or housing prices, and classifying data into categories, such as detecting whether an image containsa particular object or not. It is often used in fields such as finance, healthcare, and image recognition.

ExampleOfLinearRegression

Asuitableexampleoflinearregressionusingdeep neural network would be predicting the price of ahouse based on various features such as the size of the house, the number of bedrooms, the location, andthe age of the house.

Inthisexample,theinputfeatureswouldbefedintoadeepneuralnetwork,consisting ofmultiplelayersof interconnected neurons. The first few layers of the network would learn to extract features from theinput data, such as identifying patterns and correlations between the input features.

The output of the last layer would then be passed through a linear regression model, which would use thelearned features to predict the price of the house.

Duringtraining,theweightsandbiasesoftheneuralnetworkwouldbeadjustedtominimizethedifference between the predicted price and the actual price of the house. This process is known as gradientdescent, and it involves iteratively adjusting the model's parameters until the optimal values are reached.

Once the model is trained, it can be used to predict the price of a new house based on its features. Thisapproach can be used in the real estate industry to provide accurate and reliable estimates of house prices,which can help both buyers and sellers make informed decisions.

Concept of Deep Neural Network

deep neural network is a type of machine learning algorithm that is modeled after the structure andfunction of the human brain. It consists of multiple layers of interconnected nodes, or artificial neurons,that process data and learn from it to make predictions or classifications.

Each layer of the network performs a specific type of processing on the data, such as identifying patternsor correlations between features, and passes the results to the next layer. The layers closest to the input areknown as the "input layer", while the layers closest to the output are known as the "output layer".

Theintermediatelayersbetweentheinputandoutputlayersareknownas"hiddenlayers".Theselayersare responsible for extracting increasingly complex features from the input data, and can be deep (i.e.,containing many hidden layers) or shallow (i.e., containing only a few hidden layers).

Deep neural networks are trained using a process known as backpropagation, which involves adjusting theweightsandbiasesofthenodesbasedontheerrorbetweenthepredictedoutputandtheactualoutput.Thisprocess is repeated for multiple iterations until themodelreaches an optimal level of accuracy.

Deep neural networks are used in a variety of applications, such as image and speech recognition, naturallanguage processing, and recommendation systems. They are capable of learning from vast amounts ofdataandcanautomaticallyextract features from raw data, making them a powerful tool for solvingcomplex problems in a wide range of domains.

HowDeepNeuralNetworkWork-

Boston House Price Prediction is a common example used to illustrate how a deep neural network canworkforregressiontasks.Thegoalofthistaskis to predict the price of a house in Boston based on various features such as the number of rooms, crime rate, and accessibility to public transportation.

Here's how a deep neural network can work for Boston House Price Prediction:

1. **Data preprocessing:** The first step is to preprocess the data. This involves normalizing the input features to have a mean of 0 and a standard deviation of 1, which helps the network learn more efficiently. The dataset is then split into training and testing sets.
2. **Model architecture:** A deep neural network is then defined with multiple layers. The first layer isthe input layer, which takes in the normalized features. This is followed by several hidden layers,whichcanbedeeporshallow.Thelastlayeristheoutputlayer,whichpredictsthehouseprice.
3. **Model training:** The model is then trained using the training set. During training, the weights and biases of the nodes are adjusted based on the error between the predicted output and the actual output. This is done using an optimization algorithm such as stochastic gradient descent.
4. **Modelevaluation:**Oncethemodelistrained,itisevaluatedusingthetestingset.Theperformance of the model is measured using metrics such as mean squared error or mean absolute error.
5. **Modelprediction:**Finally, the trained model can be used to make predictions on new data, such as predicting the price of a new house in Boston based on its features.
6. By using a deep neural network for Boston House Price Prediction, we can obtain accurate predictions based on a large set of input features. This approach is scalable and can be used for other regression tasks as well.

Boston House Price Prediction Dataset-

BostonHousePricePredictionisawell-knowndatasetinmachinelearningandisoftenusedtodemonstrate regression analysis techniques. The dataset contains information about 506 houses in Boston,Massachusetts, USA. The goal is to predict the median value of owner-occupied homes in thousands ofdollars.

Thedatasetincludes13inputfeatures,whichare:

CRIM: per capita crime rate by town

ZN: proportion of residential land zoned for lots over 25,000 sq.ft.

INDUS: proportion of non-retail business acres per town

CHAS: Charles River dummy variable (1 if tract bounds river; 0 otherwise)

NOX: nitric oxides concentration (parts per 10 million)

RM: average number of rooms per dwelling

AGE: proportion of owner-occupiedunitsbuiltpriorto1940**DIS:** weighted distances to five Boston employment centers**RAD:** index of accessibility to radial highways

TAX:full-valueproperty-taxrateper$10,000

PTRATIO:pupil-teacherratiobytown

B: 1000(Bk - 0.63)^2 where Bk is the proportion of black people by town

LSTAT:%lowerstatusofthepopulation

Theoutput variable isthe median valueof owner-occupied homesin thousands ofdollars (MEDV).

Topredictthemedianvalueofowner-occupiedhomes,aregressionmodelistrainedonthedataset.Themodel can be a simple linear regression model or a more complex model, such as a deep neural network. Afterthemodelistrained,itcanbeusedtopredictthemedianvalueofowner-occupiedhomes basedontheinputfeatures.Themodel'saccuracycanbeevaluatedusingmetricssuchasmeansquarederrorormeanabsoluteerror.

Boston House Price Prediction is a example of regression analysis and is often used to teach machinelearning concepts. The dataset is also used in research to compare the performance of different regressionmodels.

SourceCodewithExplanation-

```
#Importingthepandasfordataprocessingandnumpyfornumericalcomputing
    importnumpy as
    npimportpandasaspd
# Importing the Boston Housing dataset from the sklearnfrom
sklearn.datasets import load_boston
    boston = load_boston()
#Converting the data into pandas dataframedata =
pd.DataFrame(boston.data)
```

#First look at the datadata.head()

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

#Adding the feature names to the dataframedata.columns=boston.feature_names#Adding the target variable to the datasetdata['PRICE'] = boston.target

#Looking at the data with names and target variabledata.head(n=10)

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2
5	0.02985	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	3.0	222.0	18.7	394.12	5.21	28.7
6	0.08829	12.5	7.87	0.0	0.524	6.012	66.6	5.5605	5.0	311.0	15.2	395.60	12.43	22.9
7	0.14455	12.5	7.87	0.0	0.524	6.172	96.1	5.9505	5.0	311.0	15.2	396.90	19.15	27.1
8	0.21124	12.5	7.87	0.0	0.524	5.631	100.0	6.0821	5.0	311.0	15.2	386.63	29.93	16.5
9	0.17004	12.5	7.87	0.0	0.524	6.004	85.9	6.5921	5.0	311.0	15.2	386.71	17.10	18.9

#Shape of the dataprint(data.shape)

#Checking the null values in the datasetdata.isnull().sum()

CRIM 0
ZN 0
INDUS 0
CHAS 0
NOX 0
RM 0
AGE 0
DIS 0
RAD 0
TAX 0
PTRATIO 0
B 0
LSTAT 0
PRICE 0

dtype: int64

#Checking the statistics of the datadata.describe()

#Thisissometimesveryuseful,forexampleifyoulookattheCRIMthemaxis 88.97 and 75% of the value is below 3.677083 and

#meanis3.613524 so it means the max values is actually an outlier or there are

outliers present in the column

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000

data.info()

<class 'pandas.core.frame.DataFrame'>RangeIndex: 506 entries,0 to505Data columns (total 14 columns):

#	Column	Non-Null Count	Dtype
0	CRIM	506 non-null	float64
1	ZN	506 non-null	float64
2	INDUS	506 non-null	float64
3	CHAS	506 non-null	float64
4	NOX	506 non-null	float64
5	RM	506 non-null	float64
6	AGE	506 non-null	float64
7	DIS	506 non-null	float64

8	RAD	506	non-null	float64
9	TAX	506	non-null	float64
10	PTRATIO	506	non-null	float64
11	B	506	non-null	float64
12	LSTAT	506	non-null	float64
13	PRICE	506	non-null	float64

dtypes:

float64(14)memoryusage:5

5.5ceckingthedistributionof

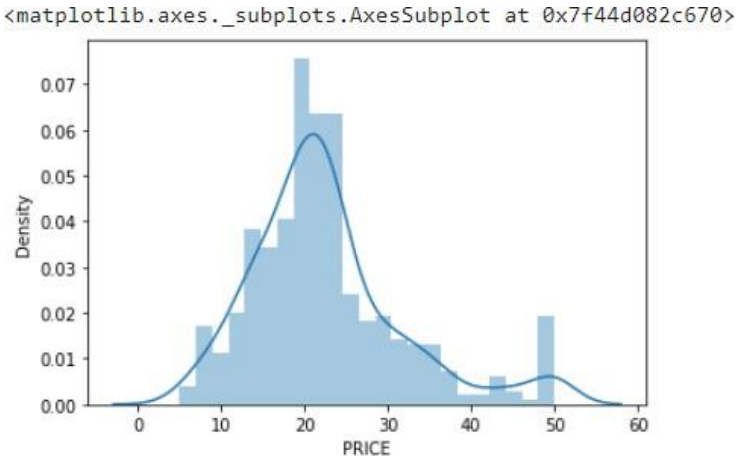
thetargetvariableimport

seaborn as sns

sns.distplot(data.PRICE)

#The distribution seems normal, has not be the data normal we would have performlog transformation or took to square root of the data to make the data normal.

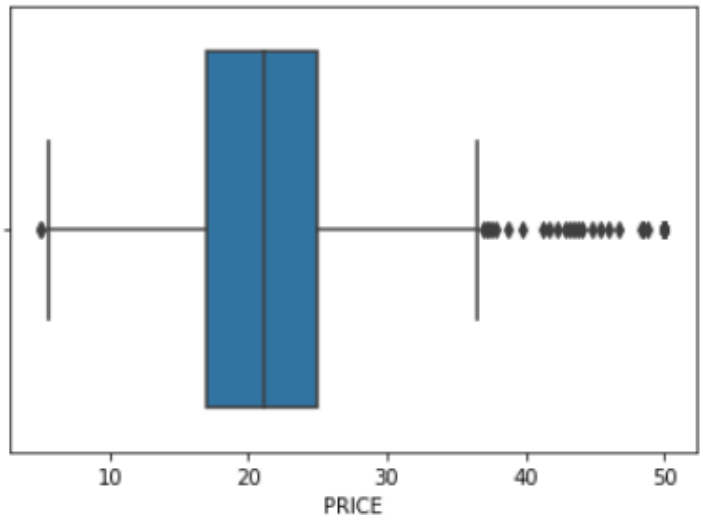
Normal distribution is need for the machine learning for better predictiblityof the model



#Distribution using box

plotsns.boxplot(data.PRICE)

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f44d077ed60>
```



#Checking the correlation of theindependent feature with thedependent feature# Correlation is a statistical technique that can show whether and how stronglypairs of variables are related.An intelligent correlation analysis can lead to agreater understanding of your data

#checking Correlation of the datacorrelation = data.corr()correlation.loc['PRICE']

CRIM -0.388305

Name: PRICE, dtype: float64# plotting the heatmap

importmatplotlib.pyplot as plt

fig,axes = plt.subplots(figsize=(15,12))sns.heatmap(correlation,square=True, annot=True)

By looking at the correlation plot LSAT is negatively correlated with -0.75 andRM is positively correlated tothe price and PTRATIO iscorrelated negativelywith -0.51# Checking the scatter plot with the

most correlated featuresplt.figure(figsize = (20,5))

features = ['LSTAT','RM','PTRATIO']

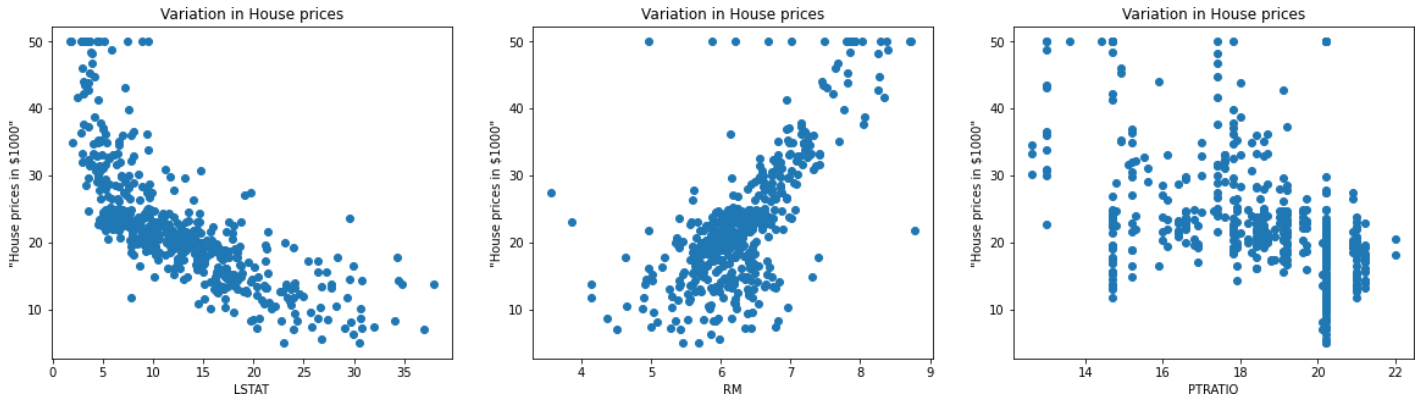
fori, col in enumerate(features):plt.subplot(1,

len(features) , i+1)x = data[col]

y = data.PRICE

plt.scatter(x, y, marker='o')plt.title("Variation in House prices")plt.xlabel(col)

plt.ylabel("'House prices in $1000"')



Splitting the dependent feature and independent feature#X = data[['LSTAT','RM','PTRATIO']]

X = data.iloc[:,:-1]y= data.PRICE

In order to provide a standardized input to our neural network, we need theperform the normalization of our dataset.

This can be seen as an step to reduce the differences in scale that may arisefrom the existent features.

We perform this normalization by subtracting the mean from our data anddividing it by the standard deviation.

One more time, this normalization should only be performed by using the meanand standard deviation from the training set,

in order to avoid any information leak from the test set.

mean = X_train.mean(axis=0)std = X_train.std(axis=0)

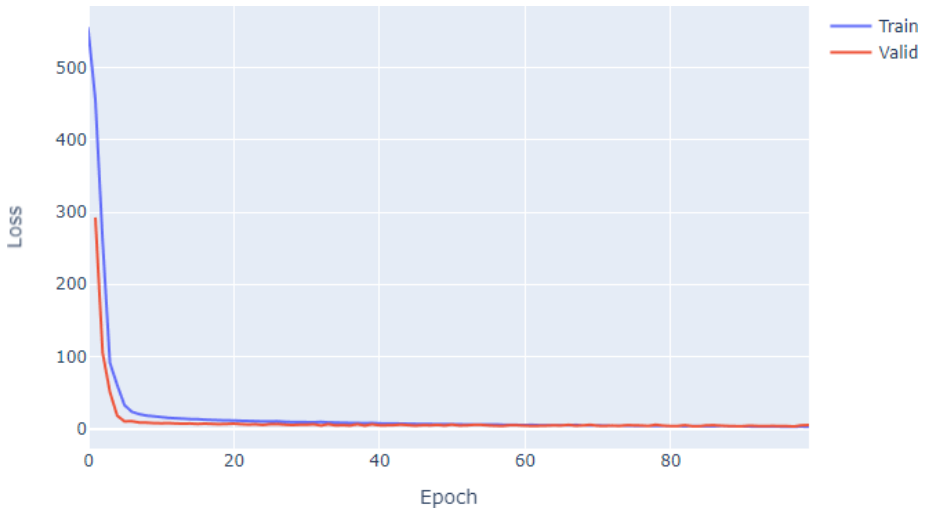
X_train = (X_train - mean) / stdX_test = (X_test - mean) / std#Linear Regression

fromsklearn.linear_model import LinearRegressionr regressor = LinearRegression()#Fitting the modelregressor.fit(X_train,y_train)# Model Evaluation

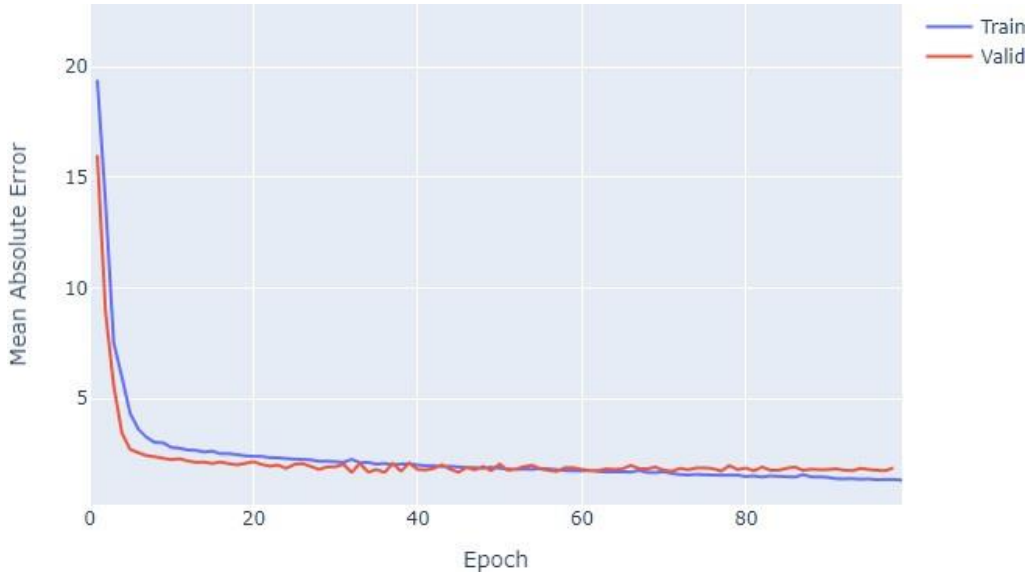
#Prediction on the test datasety_pred=regressor.predict(X_test)

```
# Predicting RMSE the Test set results
fromsklearn.metrics import mean_squared_error
rmse = (np.sqrt(mean_squared_error(y_test, y_pred)))print(rmse)

fromsklearn.metrics import r2_scorer2 =
r2_score(y_test, y_pred)print(r2)
# Neural
Networks#Scalingthedata
set
fromsklearn.preprocessing import StandardScalersc =
StandardScaler()
X_train = sc.fit_transform(X_train)X_test =
sc.transform(X_test)
# Due to the small amount of presented data in this dataset, we must be carefulto not create an overly complex model,
# which could lead to overfitting our data. For this, we are going to adopt anarchitecture based on two Dense layers,
# the first with 128 and the second with 64 neurons, both using a ReLU activationfunction.
# A dense layer with a linear activation will be used as output layer.
# In order to allow us to know if our model is properly learning, we will use amean squared error loss function and to report the performance of it we willadopt the mean average error metric.
# By using the summary method from Keras, we can see that we have a total of10,113 parameters, which is acceptable for us.

#Creating the neural network modelimport
keras
fromkeras.layers import Dense, Activation,Dropoutfrom
keras.models import Sequential

model = Sequential()
model.add(Dense(128,activation = 'relu',input_dim
=13))model.add(Dense(64,activation =
'relu')model.add(Dense(32,activation =
```

```
'relu'))model.add(Dense(16,activation = 'relu'))model.add(Dense(1))
#model.compile(optimizer='adam', loss='mse',
metrics=['mae'])model.compile(optimizer='adam',loss='mean_squared_error',metrics=['mae'])
!pip install ann_visualizer
!pip install graphviz
fromann_visualizer.visualize import ann_viz;#Build your
model here
ann_viz(model, title="DEMO ANN");
history = model.fit(X_train, y_train, epochs=100, validation_split=0.05)
# By plotting both loss and mean average error, we can see that our model wascapable of learning
patterns in our data without overfitting taking place (asshown by the validation set curves)
fromplotly.subplots import make_subplotsimport
plotly.graph_objects as go

fig = go.Figure()fig.add_trace(go.Scattergl(y=history.history['loss'],
                                name='Train'))

fig.add_trace(go.Scattergl(y=history.history['val_loss'],
                                name='Valid'))



fig.update_layout(height=500, width=700,
                                xaxis_title='Epoch',yaxis_
                                title='Loss')
fig.show()
```




```
fig = go.Figure()ig.add_trace(go.Scattergl(y=history.history['mae'],
                                name='Train'))

fig.add_trace(go.Scattergl(y=history.history['val_mae'],
                                name='Valid'))



fig.update_layout(height=500, width=700,
                                xaxis_title='Epoch',yaxis_title='MeanAbsol
                                uteError')
fig.show()
```



#Evaluation

ofthemodely_pred=model.predict(X_te

st)

mse_nn, mae_nn = model.evaluate(X_test, y_test)

print('Mean squared error on test data: ', mse_nn)print('Meanabsoluteerrorontestdata:',mae_nn)

4/4[==============================]-0s4ms/step-loss:10.5717-mae:2.2670

Mean squared error on test data: 10.571733474731445Mean absolute error on test data: 2.2669904232025146#Comparison with traditional approaches

#First let's try with a simple algorithm, the Linear Regression:

fromsklearn.metrics import mean_absolute_error

lr_model = LinearRegression()lr_model.fi(X_train ,y_train

y_pred_lr = lr_model.predict(X_test)

mse_lr = mean_squared_error(y_test, y_pred_lr)mae_lr=mean_absolute_error(y_test,y_pred_lr)

print('Mean squared error on test data: ', mse_lr)print('Mean absolute error on test data: ', mae_lr)from sklearn.metrics import r2_score

r2 = r2_score(y_test, y_pred)print(r2)

0.8812832788381159

Predicting RMSE the Test set results

fromsklearn.metrics import mean_squared_error

rmse = (np.sqrt(mean_squared_error(y_test, y_pred)))print(rmse)

3.320768607496587

Make predictions on new dataimport sklearn

new_data = sklearn.preprocessing.StandardScaler().fit_transform(([[0.1, 10.0,5.0, 0, 0.4, 6.0, 50, 6.0, 1, 400, 20, 300, 10]]))

prediction = model.predict(new_data)print("Predictedhouseprice:",prediction)

1/1 [==============================] - 0s 70ms/step

Predicted house price: [[11.104753]]

#new_data = sklearn.preprocessing.StandardScaler().fit_transform(([[0.1, 10.0, 5.0,0,0.4,6.0,50,6.0,1,400,20,300, 10]])) is a line of code that standardizes the input features of a new data point.

Inthisspecificcase,wehaveanewdatapointrepresentedasalistof13numericvalues([0.1,10.0,5.0,0,0.4,6.0,50,6.0,1, 400, 20, 300, 10]) that represents the values for the 13 features ofthe Boston House Price dataset. The StandardScaler() function from the sklearn.preprocessing moduleis used to standardize the data.

Standardization scales each featureto have zero mean and unit variance, which is a common preprocessingstep in machine learning to ensure that all features contributeequally to the model.

The fit_transform() method is used to fit the scaler to the data andapplythestandardizationtransformation.Theresultisanewdata

point with standardized feature values.

Conclusion- In this way we can predict the Boston House Price using Deep Neural Network.

Assignment Question

1. What is Linear Regression?
2. What is a Deep Neural Network?
3. What is the concept of standardization?
4. Why split data into train and test?
5. Write Down Application of Deep Neural Network?

Deep Learning

Assignment No: 2A

Title of the Assignment: Binary classification using Deep Neural Networks Example: Classify movie reviews into positive" reviews and "negative" reviews, just based on the text content of the reviews. Use IMDB dataset

Objective of the Assignment: Students should be able to Classify movie reviews into positive reviews and "negative reviews on IMDB Dataset.

Prerequisite:

1. Basic of programming language
2. Concept of Classification
3. Concept of Deep Neural Network

Contents for Theory:

1. What is Classification
2. Example of Classification
3. How Deep Neural Network Work on Classification
4. Code Explanation with Output

---

what is Classification?

Classification is a type of supervised learning in machine learning that involves categorizing data intopredefined classes or categories based on a set of features or characteristics. It is used to predict the classofnew, unseen databased on thepatterns learned fromthe labeled training data.

In classification, a model is trained on a labeled dataset, where each data point has a known class label.The model learns to associate the input features with the corresponding class labels and can then be usedtoclassify new, unseen data.

For example, we can use classification to identify whether an email is spam or not based on its contentand

metadata, to predict whether a patient has a disease based on their medical records and symptoms, ortoclassify images into different categories based on their visual features.

Classification algorithms can vary in complexity, ranging from simple models such as decision trees andk-nearest neighbors to more complex models such as support vector machines and neural networks. Thechoice of algorithm depends on the nature of the data, the size of the dataset, and the desired level ofaccuracyand interpretability.

Classificationisacommontaskindeepneuralnetworks,wherethegoalistopredicttheclassofaninput based on its features. Here's an example of how classification can be performed in a deep neuralnetwork using the popular MNISTdataset of handwritten digits.

The MNIST dataset contains 60,000 training images and 10,000 testing images of handwritten digitsfrom 0 to 9. Each image is a grayscale 28x28 pixel image, and the task is to classify each image into oneof the 10 classes corresponding to the 10 digits.

We can use a convolutional neural network (CNN) to classify the MNIST dataset. A CNN is a type ofdeep neural network that is commonly used for image classification tasks.

How Deep Neural Network Work on Classification-

Deep neural networks are commonly used for classification tasks because they can automatically learn to extract relevant features from raw input data and map them to the correct output class.

The basic architecture of a deep neural network for classification consists of three main parts: an input layer, one or more hidden layers, and an output layer. The input layer receives the raw input data, which is usually preprocessed to a fixed size and format. The hidden layers are composed of neurons that apply linear transformations and nonlinear activations to the input features to extract relevant patterns and representations. Finally, the output layer produces the predicted class labels, usually as a probability distribution over the possible classes.

During training, the deep neural network learns to adjust its weights and biases in each layer to minimize the difference between the predicted output and the true labels. This is typically done by optimizing loss function that measures the discrepancy between the predicted and true labels, using techniques such as gradient descent or stochastic gradient descent.

One of the key advantages of deep neural networks for classification is their ability to learn hierarchical representations of the input data. In a deep neural network with multiple hidden layers, each layer learns to capture more complex and abstract features than the previous layer, by building on the representations learned by the earlier layers. This hierarchical structure allows deep neural networks to learn highly discriminative features that can separate different classes of input data, even when the data is highly complex or noisy.

Overall, the effectiveness of deepneuralnetworksforclassificationdependsonthechoiceofarchitecture, hyper parameters, and training procedure, as well as the quality and quantity of the training data. When trained properly, deep neural networks can achieve state-of-the-art performance on a wide range of classification tasks, from image recognition to natural language processing.

IMDB Dataset-The IMDB dataset is a large collection of movie reviews collected from the IMDBwebsite, which is a popular source of user-generated movie ratings and reviews. The dataset consists of50,000 movie reviews, split into 25,000 reviews for training and 25,000 reviews for testing.

Each review is represented as a sequence of words, where each word is represented by an integer indexbased on its frequency in the dataset. The labels for each review are binary, with 0 indicating a negativereviewand 1 indicating a positive review.

The IMDB dataset is commonly used as a benchmark for sentiment analysis and text classification tasks,where the goal is to classify the movie reviews as either positive or negative based on their text content.The dataset is challenging because the reviews are often highly subjective and can contain complexlanguage and nuances of meaning, making it difficult for traditional machine learning approaches toaccurately classify them.

p learning approaches, such as deep neural networks, have achieved state-of-the-art performance on the IMDB dataset by automatically learning to extract relevant features from the raw text data and map them to the correct output class. The IMDB dataset is widely used in research and education for natural language processing and machine learning, as it provides a rich source of labeled text data for training and testing deep learning models.

Source Code and Output-

The IMDB sentiment classification dataset consists of 50,000 movie reviews from IMDB users that arelabeled as either positive (1) or negative (0).

#Thereviewsarepreprocessedandeachoneisencodedasasequenceofwordindexesintheformofintegers.

#Thewordswithinthereviewsareindexedbytheiroverallfrequencywithinthedataset. Forexample,the integer "2" encodes the second most frequent word in the data.

#The 50,000 reviews are split into 25,000 for training and 25,000 for testing.

#TextProcesswordbywordatdiffrenttimestamp(YoumayuseRNNLSTMGRU)# convert input text to vector reprint input text

DOMAIN: Digital content and entertainment industry

#CONTEXT: The objective of this project is to build a text classification model that analyses the customer's sentiments based on their reviews in the IMDB database. The model uses a complex deeplearningmodeltobuildanembeddinglayerfollowedbyaclassificationalgorithmto analyze the sentiment of the customers.

DATA DESCRIPTION: The Dataset of 50,000 movie reviews from IMDB, labeled by sentiment(positive/negative).

#Reviewshavebeenpreprocessed,andeachreviewisencodedasasequenceofwordindexes(integers).

For convenience, the words are indexed by their frequency in the dataset, meaning the for that hasindex 1 is the most frequent word.

Use the first 20 words from each review to speed up training, using a max vocabulary size of 10,000.

#Asaconvention,"0"doesnot stand for a specific word, but instead is used to encode any unknownword.

#PROJECTOBJECTIVE:BuildasequentialNLPclassifierwhichcanuseinputtextparameterstodetermine the customer sentiments.

```
import numpy as
npimportpandasaspd
from sklearn.model_selection import
train_test_split#loadingimdbdatawithmostfrequent
```

```
from keras.datasets import imdb
(X_train,y_train),(X_test,y_test)=imdb.load_data(num_words=10000)#youmaytaketop10,000word frequently used review of movies other are discarded
#consolidatingdataforEDAExploratorydataanalysis(EDA)isusedbydatascientiststoanalyzeandinvestigate data sets and summarize their main characteristics
data=np.concatenate((X_train,X_test),axis=0)#axis0isfirstrunningverticallydownwardsacrossrows (axis 0), axis 1 is second running horizontally across columns (axis 1),
label = np.concatenate((y_train, y_test), axis=0)
X_train.shape(
25000,)
X_test.shape(
25000,)
y_train.shape(
25000,)
y_test.shape(
25000,)
print("Review is ",X_train[0]) # series of no converted word to vocabulory associated with indexprint("Review is ",y_train[0])
Reviewis[1,194,1153,194,8255,78,228,5,6,1463,4369,5012,134,26,4,715, 8,118,1634, 14,
394,20,13,119,954,189,102,5,207,110,3103,21,14,69,188,8,30,23,7,4,249,126,93,4, 114,
9,2300,1523,5,647,4,116,9,35,8163,4,229,9,340,1322,4,118,9,4,130,4901,19,4,1002, 5,
89,29,952,46,37,4,455,9,45,43,38,1543,1905,398,4,1649,26,6853,5,163,11,3215,2,4,
1153,9,194,775,7,8255,2,349,2637,148,605,2,8003,15,123,125,68,2,6853,15,349,165,
4362,98,5,4,228,9,43,2,1157,15,299,120,5,120,174,11,220, 175,136,50, 9,4373,228, 8255,
5,2,656,245,2350,5,4,9837,131,152,491,18,2,32,7464,1212,14,9,6,371,78,22,625,64,
1382, 9, 8, 168, 145, 23, 4, 1690, 15, 16, 4, 1355, 5, 28, 6, 52, 154, 462, 33, 89, 78, 285, 16, 145, 95]
Review is0
vocab=imdb.get_word_index() # Retrieve the word index file mapping words to indicesprint(vocab)
{'fawn':34701,'tsukino':52006,'nunnery':52007,'sonja':16816,'vani':63951, 'woods': 1408, 'spiders':
16115,
```

y_train

array([1, 0, 0, ..., 0, 1, 0])

y_test

array([0, 1, 1, ..., 0, 0, 0])

Function to perform relevant sequence adding on the data

#Nowitistimetoprepareourdata.Wewillvectorizeeveryreviewandfillitwithzerossothatit

contains exactly 10000 numbers.

#That means we fill every review that is shorter than 500 with zeros.

#Wedothisbecausethebiggestreviewisnearlythatlongandeveryinputforourneuralnetworkneedsto have the same size.

#Wealsotransformthetargetsintofloats.

sequences is name of method the review less than 10000 we perform padding overthere#

binary vectorization code:

#VECTORIZE asone cannotfeed integersinto aNN

Encoding the integer sequences into a binary matrix - one hot encoder basically

#Fromintegersrepresentingwords,atvariouslengths-toanormalizedonehotencodedtensor(matrix)of 10k columns

defvectorize(sequences,dimension=10000): #Wewillvectorizeeveryreviewandfillitwithzerosso that it contains exactly 10,000 numbers.

```
  # Create an all-zero matrix of shape (len(sequences),
    dimension)results = np.zeros((len(sequences), dimension))
    for i, sequence in
       enumerate(sequences):results[i,
       sequence] = 1
    return results
```

#Nowwesplitourdataintoatrainingandatestingset.#Thetraini

ngsetwillcontainreviewsandthetestingset##SetaVALIDAT

IONset

```
test_x =
data[:10000]test_y =
label[:10000]train_x =
data[10000:]train_y =
label[10000:]test_x.sha
pe
(10000,)
```

test_y.shape(

```
10000,)
train_x.shape(
40000,)
train_y.shape(
40000,)
print("Categories:", np. unique(label))
print("Number of unique words:", len(np.unique(np.hstack(data))))
```

#Thehstack()functionisusedtostackarraysinsequencehorizontally(columnwise).Categories: [0 1]
Number of unique words:
9998length = [len(i) for i in
data]
print("Average Review length:",
np.mean(length))print("StandardDeviation:",round(
np.std(length)))

#Thewholedatasetcontains9998uniquewordsandtheaveragereviewlengthis234words,withastandard deviation of 173 words.
AverageReviewlength:234.75892St
andard Deviation: 173
If you look at the data you will realize it has been already pre-processed.
All words have been mapped to integers and the integers represent the words sorted by their frequency.#This is very common in text analysis to represent a dataset like this.
So 4 represents the 4th most used word,#
5 the 5th most used word and so on...
#Theinteger1isreservedforthestartmarker,
the integer 2 for an unknown word and 0 for padding.#
Let's look at a single training example:
print("Label:", label[0])
Label: 1
print("Label:", label[1])
Label:
0print(data[0])
# Retrieves a dict mapping words to their index in the IMDB
dataset.index = imdb.get_word_index()# word to index
#Createinvertedindexfromadictionarywithdocumentidsaskeysandalistoftermsasvaluesforeach document
reverse_index=dict([(value,key)for(key,value)inindex.items()])#idtoword

decoded = " ".join([reverse_index.get(i - 3, "#") for i in data[0]])

```
#Theindicesareoffsetby3because0,1and2arereservedindicesfor"padding","startofsequence"and
"unknown".
print(decoded)

#this film was just brilliant casting location scenery story direction everyone's really suited the part they
```

played and you could just imagine being there robert # is an amazing actor and now the same beingdirector # father came from the same scottish island as myself so i loved the fact there was a realconnection with this film the witty remarks throughout the film

#Adding sequence to data

Vectorization is the process of converting textual data into numerical vectors and is a process that isusually applied once the text is cleaned.

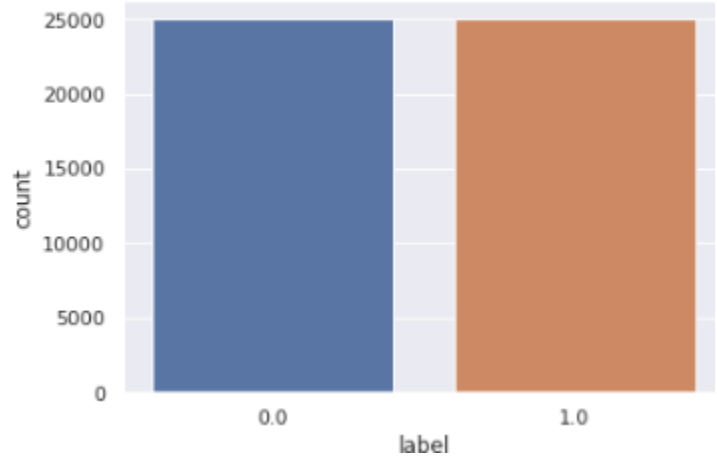
data = vectorize(data)

label =

np.array(label).astype("float32")labelDF

=pd.DataFrame({'label':label})sns.count

```
<AxesSubplot:xlabel='label', ylabel='count'>
```



plot(x='label', data=labelDF)

Creating train and test data set

fromsklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(data,label, test_size=0.20, random_state=1)

X_train.shape(40000

,10000)

X_test.shape(10000,

10000)

Let's create sequential model

fromkeras.utils import

to_categoricalfrom keras import

models

```
fromkeras import layersmodel
= models.Sequential()# Input
- Layer
#Notethatwesettheinput-shapeto10,000attheinput-layerbecauseourreviewsare10,000integerslong.
#Theinput-
layertakes10,000asinputandoutputsitwithashapeof50.model.add(layers.De
nse(50, activation = "relu", input_shape=(10000, )))
```

Hidden - Layers

#Pleasenoteyoushouldalwaysuseadropoutrate between20%and50%.# hereinourcase0.3 means30% dropout we are using dropout to prevent overfitting.

#Bytheway,ifyouwantyoucanbuildasentimentanalysiswithoutLSTMs,thenyousimplyneedtoreplace it by a flatten layer:

model.add(layers.Dropout(0.3, noise_shape=None, seed=None))model.add(layers.Dense(50, activation = "relu"))model.add(layers.Dropout(0.2, noise_shape=None, seed=None))model.add(layers.Dense(50, activation = "relu"))

Output- Layer

model.add(layers.Dense(1, activation = "sigmoid"))model.summary()

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 50)	500050
dropout (Dropout)	(None, 50)	0
dense_1 (Dense)	(None, 50)	2550
dropout_1 (Dropout)	(None, 50)	0
dense_2 (Dense)	(None, 50)	2550
dense_3 (Dense)	(None, 1)	51

Total params: 505,201

Trainable params: 505,201

Non-trainable params: 0#For

```
early stopping
# Stop training when a monitored metric has stopped improving.# monitor:
Quantity to be monitored.
#patience:Numberofepochswithnoimprovementafterwhichtrainingwillbestopped.
importtensorflow as tf
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3)
#Weusethe“adam”optimizer,analgorithmthatchangestheweightsandbiasesduring training.
#Wealsochoosebinary-crossentropyasloss(becausewedealwithbinaryclassification) and accuracy as our
evaluation metric.


model.compile(optimizer
  ="adam",
  loss  =  "binary_crossentropy",metrics
  = ["accuracy"]
)
fromsklearn.model_selection import train_test_split


results                    =
  model.fit(X_train,
  y_train,epochs= 2,
  batch_size = 500,
  validation_data          =          (X_test,
  y_test),callbacks=[callback]
)
#      Let's      check      mean      accuracy      of      our
modelprint(np.mean(results.history["val_accuracy"]))#
Evaluate the model
score  =  model.evaluate(X_test,  y_test,  batch_size=500)print('Test
loss:', score[0])
print('Test accuracy:', score[1])
20/20 [==============================] - 1s 24ms/step - loss: 0.2511 - accuracy:
```

0.8986

Test loss: 0.25108325481414795

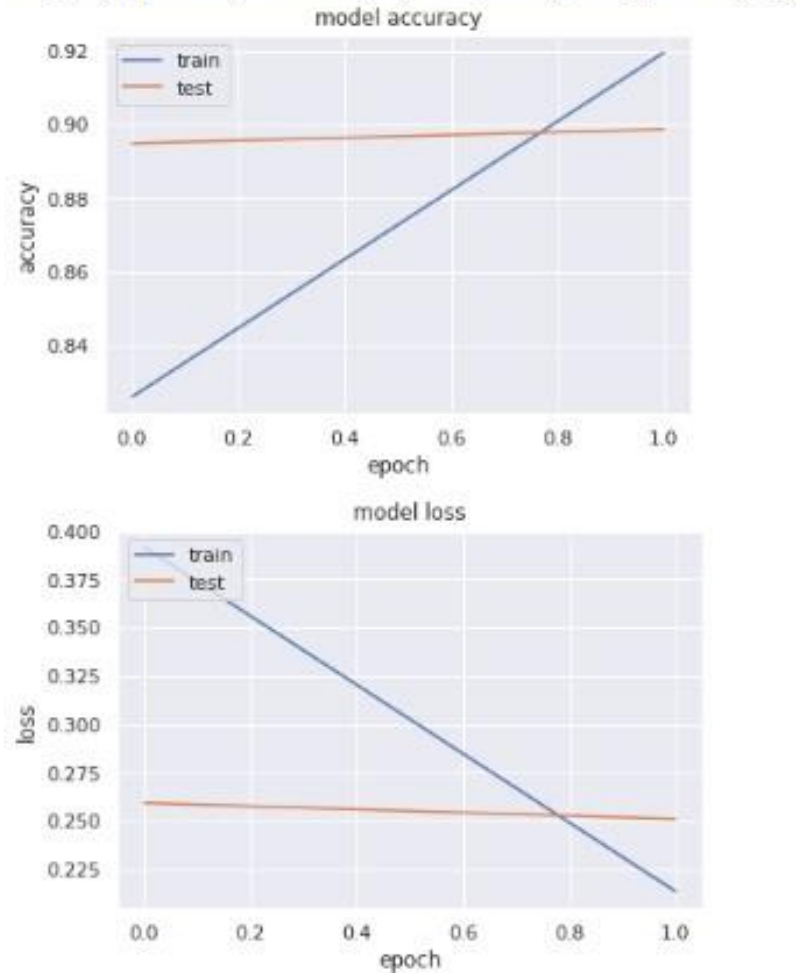
Test accuracy: 0.8985999822616577

#Let's plot training history of our model.

```
# list all data in historyprint(results.history.keys())
# summarize history for accuracyplt.plot(results.history['accuracy'])plt.plot(results.history['val_accuracy'])plt.title('model accuracy')plt.ylabel('accuracy')plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')plt.show()
# summarize history for lossplt.plot(results.history['loss'])plt.plot(results.history['val_loss'])plt.title('model loss')plt.ylabel('loss')
```

```
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')plt.show()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```



Conclusion- In this way we can Classify the Movie Reviews by using DNN.

Assignment Question

1. What is Binary Classification?
2. What is binary Cross Entropy?
3. What is Validation Split?
4. What is the Epoch Cycle?
5. What is Adam Optimizer?

GroupB Deep Learning

Assignment No :3B

TitleoftheAssignment: UseMNISTFashionDatasetandcreateaclassifiertoclassifyfashionclothinginto categories.

ObjectiveoftheAssignment: StudentsshouldbeabletoClassifymoviereviewsintopositivereviewsand "negative reviews on IMDB Dataset.

Prerequisite:

1. Basic of programming language
2. Concept of Classification
3. Concept of Deep Neural Network

Contents forTheory:

1. What is Classification
2. Example of Classification
3. What is CNN?
4. How Deep Neural Network Work on Classification
5. Code Explanation with Output

What is Classification?

Classification is a type of supervised learning in machine learning that involves categorizing data intopredefined classes or categories based on a set of features or characteristics. It is used to predict the classofnew, unseen databased on thepatterns learned fromthe labeled training data.

In classification, a model is trained on a labeled dataset, where each data point has a known class label.The model learns to associate the input features with the corresponding class labels and can then be usedtoclassify new, unseen data.

For example, we can use classification to identify whether an email is spam or not based on its content and metadata, to predict whether a patient has a disease based on their medical records and symptoms, ortoclassify images into different categories based on their visual features.

Classification algorithms can vary in complexity, ranging from simple models such as decision trees andk-nearest neighbors to more complex models such as support vector machines and neural networks. Thechoice of algorithm depends on the nature of the data, the size of the dataset, and the desired level ofaccuracyand interpretability.

Example- Classification is a common task in deep neural networks, where the goal is to predict the classof an input based on its features. Here's an example of how classification can be performed in a deepneural network using the popular MNISTdataset of handwritten digits.

The MNIST dataset contains 60,000 training images and 10,000 testing images of handwritten digitsfrom 0 to 9. Each image is a grayscale 28x28 pixel image, and the task is to classify each image into oneof the 10 classes corresponding to the 10 digits.

We can use a convolutional neural network (CNN) to classify the MNIST dataset. A CNN is a type ofdeep neural network that is commonly used for image classification tasks.

What us CNN-

Convolutional Neural Networks (CNNs) are commonly used for image classification tasks, and they aredesigned to automatically learn and extract features from input images. Let's consider an example ofusing a CNN to classify images of handwritten digits.a typical CNN architecture for image classification, there are several layers, including convolutionallayers, pooling layers, and fully connected layers. Here's a diagram of a simple CNN architecture for the digit classification task:

The input to the network is an image of size 28x28 pixels, and the output is a probability distribution over the 10 possible digits (0 to 9).

The convolutional layers in the CNN apply filters to the input image, looking for specific patterns andfeatures. Each filter produces a feature map that highlights areas of the image that match the filter. The filters are learned during training, so the network can automatically learn which features are mostrelevant for the classification task.

The pooling layers in the CNN downsample the feature maps, reducing the spatial dimensions of thedata. This helps to reduce the number of parameters in the network, while also making the features morerobust to small variations in the input image.

ThefullyconnectedlayersintheCNNtaketheflattenedoutputfromthelastpoolinglayerand performa classification task by outputting a probability distribution over the 10 possible digits.

During training, the network learns the optimal values of the filters and parameters by minimizing a loss

function. This is typically done using stochastic gradient descent or a similar optimization algorithm.
Once trained, the network can be used to classify new images by passing them through the network and computing the output probability distribution.
Overall, CNNs are powerful tools for image recognition tasks and have been used successfully in many applications, including object detection, face recognition, and medical image analysis.
CNNs have a wide range of applications in various fields, some of which are:

Image classification: CNNs are commonly used for image classification tasks, such as identifying objects in images and recognizing faces.

Object detection: CNN scan be used for object detection in images and videos, which involves identifying the location of objects in an image and drawing bounding boxes around them.

Semanticsegmentation:CNNs can be used for semantic segmentation, whichinvolvespartitioninganimage into segments and assigning each segment a semantic label (e.g., "road", "sky", "building").

Natural language processing: CNN scan be used for natural language processing tasks, such as sentiment analysis and text classification.

Medical imaging: CNNs are used in medical imaging for tasks such as diagnosing diseases from X-rays and identifying tumors from MRI scans.

Autonomousvehicles:CNNsareusedinautonomousvehiclesfortaskssuchasobjectdetectionandlane detection.

Videoanalysis:CNNscanbeusedfortaskssuchasvideoclassification,actionrecognition,andvideocaptioning. Overall,CNNsareapowerfultoolforawiderangeofapplications,andtheyhavebeenusedsuccessfullyin many areas of research and industry.

How Deep Neural Network Work on Classification using CNN-

DeepneuralnetworksusingCNNsworkonclassificationtasksbylearningtoautomaticallyextractfeatures from input images and using those features to make predictions. Here's how it works:

Input layer:The input layer of the network takes in the image data as input.

Convolutionallayers:Theconvolutionallayersapplyfilterstotheinputimagestoextractrelevantfeatures.Eachfilter producesa featuremap thathighlights areasof theimage thatmatch thefilter.

Activationfunctions:Anactivationfunctionisappliedtotheoutputofeachconvolutionallayertointroduce non-linearity into the network.

Poolinglayers:Thepoolinglayersdownsamplethefeaturemapstoreducethespatialdimensionsofthedata.

Dropoutlayer:Dropoutisusedtopreventoverfittingbyrandomlydroppingoutapercentageoftheneurons in the network during training.

Fullyconnectedlayers:Thefullyconnectedlayerstaketheflattenedoutputfromthelastpoolinglayerand perform a classification task by outputting a probability distribution over the possible classes.

Softmaxactivationfunction:Thesoftmaxactivationfunctionisappliedtotheoutputofthelastfullyconnected layer to produce a probability distribution over the possible classes.

Lossfunction:Alossfunctionisusedtocomputethedifferencebetweenthepredictedprobabilitiesandthe actual labels.

Optimization:Anoptimizationalgorithm,suchasstochasticgradientdescent,isusedtominimizetheloss function by adjusting the values of the network parameters.

Training:Thenetworkistrainedonalargedatasetoflabeledimages,adjustingthevaluesoftheparameters to minimize the loss function.

Prediction:Oncetrained,thenetworkcanbeusedtoclassifynewimagesbypassingthemthroughthenetwork and computing the output probability distribution.

MNISTDataset-

The MNIST Fashion dataset is a collection of 70,000 grayscale images of 28x28 pixels, representing 10different categories of clothing and accessories. The categories include T-shirts/tops, trousers, pullovers, dresses, coats, sandals, shirts, sneakers, bags, and ankle boots.

The dataset is often used as a benchmark for testing image classification algorithms, and it is consideredamorechallengingversionoftheoriginalMNISTdatasetwhichcontainshandwrittendigits.The

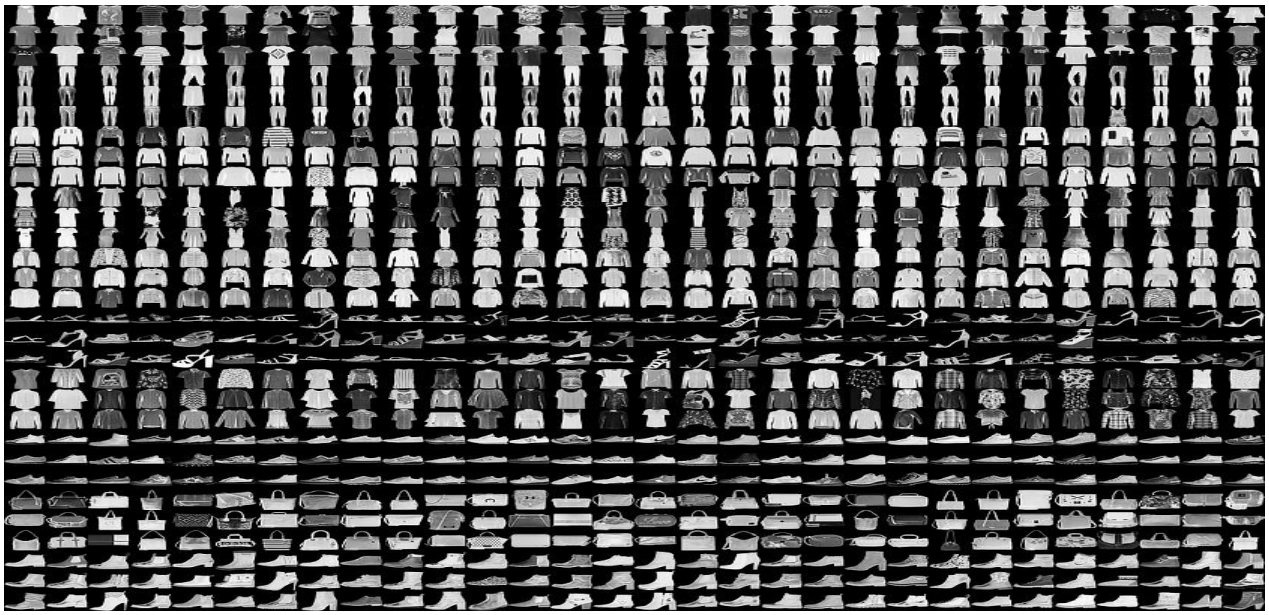
MNIST Fashion dataset was released by Zalando Research in 2017 and has since become a populardatasetin the machine learning community.

heMNIST Fashion dataset is a collection of 70,000 grayscale images of 28x28 pixels each. Theseimages represent 10 different categories of clothing and accessories, with each category containing 7,000images.The categories are as follows:

T-
shirt/topsTro
usersPullov
ersDressesC
oatsSandals
ShirtsSneak
ersBags
Ankle boots

TheimageswereobtainedfromZalando'sonline store and are preprocessed to be normalized andcentered. The training set contains 60,000 images, while the test set contains 10,000 images. The goal ofthe dataset is to accurately classify the images into their respective categories.

TheMNISTFashiondatasetisoftenusedasabenchmarkfortestingimageclassificationalgorithms,anditisconsidered amorechallengingversionoftheoriginalMNISTdatasetwhichcontainshandwritten digits. The dataset is widely used in the machine learning community for research andeducational purposes.HerearethegeneralstepstoperformConvolutionalNeuralNetwork(CNN)ontheMNISTFashiondatas



et:

- Importthenecessarylibraries,includingTensorFlow,Keras,NumPy,andMatplotlib.

- LoadthedatasetusingKeras'built-infunction,keras.datasets.fashion_mnist.load_data().Thiswill provide the training and testing sets, which will be used to train and evaluate the CNN.
- Preprocess the data by normalizing the pixel values between 0 and 1, and reshaping the images tobe of size (28, 28, 1) for compatibility with the CNN.
- DefinetheCNNarchitecture,includingthenumberandsizeoffilters,activationfunctions,andpooling layers.This can vary based on the specific problem being addressed.
- Compilethemodelbyspecifyingthelossfunction,optimizer,andevaluationmetrics.Commonchoicesi ncludecategoricalcross-entropy,Adamoptimizer,andaccuracy metric.
- TraintheCNNonthetrainingsetusingthefit()function,specifyingthenumberofepochsandbatch size.
- Evaluatetheperformanceofthemodelonthetestingsetusingtheevaluate()function.Thiswillprovide metrics such as accuracy and loss on the test set.
- Usethetrainedmodeltomakepredictionsonnewimages,ifdesired,usingthepredict()function.

SourceCodewithOutput-

```
importtensorflow as tf
importmatplotlib.pyplot as
pltfrom tensorflow import
kerasimport numpy as np

(x_train, y_train), (x_test, y_test) = keras.datasets.fashion_mnist.load_data()

#Thereare10imageclasses inthisdatasetandeachclasshasamappingcorrespondingtothe followinglabels:

#0 T-
shirt/top#1Tro
user
#2 pullover
#3 Dress
#4 Coat
#5 sandals
```

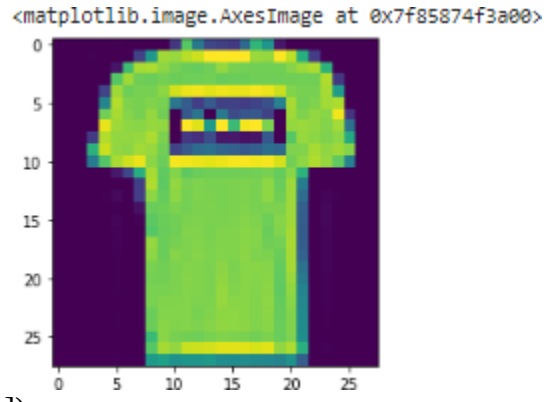
#6 shirt

#7 sneaker

#8 bag

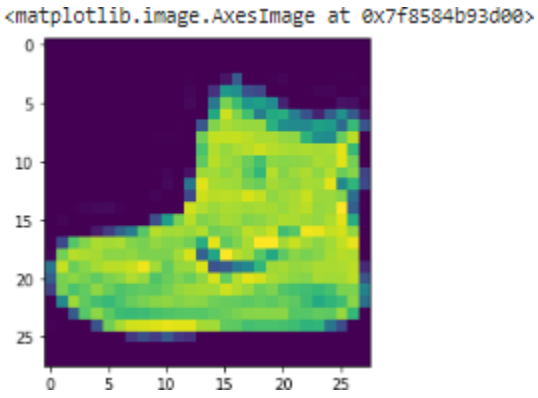
#9 ankle

bootplt.imshow(x_train



[1])

plt.imshow(x_train[0])



Next, we will preprocess the data by scaling the pixel values to be between 0 and 1, and then reshapingthe images to be 28x28 pixels.

x_train = x_train.astype('float32') /
255.0x_test = x_test.astype('float32') /
255.0

x_train = x_train.reshape(-1, 28, 28, 1)

```
x_test = x_test.reshape(-1, 28, 28, 1)


# 28, 28 comes from width, height, 1 comes from the number of channels#
-1 means that the length in that dimension is inferred.
#Thisisdonebasedontheconstraintthatthenumberofelementsinanndarrayor Tensorwhenreshaped must remain the same
#eachimageisarow vector(784elements)and therearelots ofsuchrows(let itben,so thereare784nelements).SoTensorFlow can infer that -1is n.
#convertingthetraining_imagesarrayto4dimensionalarraywithsizes60000,28,28,1for 0thto3rddimension.
x_train.shape(6
0000,28,28)
x_test.shape(1000
0,28,28,1)
y_train.shape(
60000,)
y_test.shape(
10000,)
# We will use a convolutional neural network (CNN) to classify the fashion items.#TheCNNwillconsistofmultipleconvolutionallayersfollowedbymaxpooling,#
dropout, and dense layers. Here is the code for the model:
model = keras.Sequential([
    keras.layers.Conv2D(32, (3,3), activation='relu',
    input_shape=(28,28,1)),# 32 filters (default), randomly initialized
    # 3*3 is Size of Filter
    # 28,28,1 size of Input Image
    # No zero-padding: every output 2 pixels less in every dimension
    # in Paramter shwon 320 is value of weights: (3x3 filter weights + 32 bias) * 32 filters#32*3*3=288(Total)+32(bias)= 320
    keras.layers.MaxPooling2D((2,2)),
    # It shown 13 * 13 size image with 32 channel or filter or depth.keras.layers.Dropout(0.25),
```

```
    #ReduceOverfittingofTrainingsampledropout25%Neuronkeras
    .layers.Conv2D(64, (3,3), activation='relu'),
    # Deeper layers use 64
    filters# 3*3 is Size of Filter
    # Observe how the input image on 28x28x1 is transformed to a 3x3x64 feature map
    #13(Size)-3(FilterSize)+1(bias)=11SizeforWidthandHeightwith64Depthorfiltterorchannel# in
    Paramter shwon 18496 is value of weights: (3x3 filter weights + 64 bias) * 64 filters
    # 64*3*3=576+1=577*32 + 32(bias)=18496
keras.layers.MaxPooling2D((2,2)),
  # It shown 5 * 5 size image with 64 channel or filter or
depth.keras.layers.Dropout(0.25),


    keras.layers.Conv2D(128, (3,3),
    activation='relu'),# Deeper layers use 128 filters
    # 3*3 is Size of Filter
    # Observe how the input image on 28x28x1 is transformed to a 3x3x128 feature map
    #Itshow5(Size)-3(FilterSize)+1(bias)=3SizeforWidthandHeightwith64Depthorfiltterorchannel
    #128*3*3=1152+1=1153*64+64(bias)=73856
    #Toclassifytheimages,westillneedaDenseandSoftmaxlayer.
    #Weneedtoflattenthe3x3x128featuremaptoavectorofsize1152keras.la
    yers.Flatten(),
    keras.layers.Dense(128,
    activation='relu'),# 128 Size of Node in
    Dense Layer
    #1152*128=147584
    keras.layers.Dropout(0.25),keras.layers.Den
    se(10, activation='softmax')# 10 Size of
    Node another Dense Layer
    # 128*10+10 bias= 1290
])
```

model.summary()M

odel:"sequential"

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D(None, 13, 13, 32)         0
 )

 dropout (Dropout)           (None, 13, 13, 32)        0

 conv2d_1 (Conv2D)           (None,11,11,64)           18496

 max_pooling2d_1 (MaxPooling(None, 5, 5, 64)           0
 2D)

 dropout_1 (Dropout)         (None, 5, 5, 64)


conv2d_2 (Conv2D)            (None,3,3,128)            73856
flatten (Flatten)            (None,1152)               0

dense (Dense)                (None, 128)               147584

dropout_2 (Dropout)          (None, 128)               0

dense_1 (Dense)              (None, 10)                1290

=================================================================
```

Totalparams:241,546

Trainableparams:241,546

Non-trainable params: 0

#CompileandTraintheModel

After defining the model, we will compile it and train it on the training data.model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])history = model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test))

1875 is a number of batches. By default batches contain 32 samles.60000 / 32 = 1875#Finally, wewill evaluatethe performanceof the modelon thetest data.

test_loss, test_acc = model.evaluate(x_test, y_test)print('Testaccuracy:',test_acc)

313/313 [==============================] - 3s 10ms/step - loss: 0.2606 - accuracy: 0.9031

Testaccuracy:0.9031000137329102

Conclusion- In this way we can classify fashion clothing into categories using CNN.

Assignment Question

1. What is Binary Classification?
2. What is binary Cross Entropy?
3. What is Validation Split?
4. What is the Epoch Cycle
5. What is Adam Optimizer?