Page No. :
Date. : / /

Practical Assignment 1

Name: Tanmay Bhagwat
ROLL NO: 20C0011
Sub: HPC
Class: BE CompA

Aim: Design and implement parallel Depth first Search based on existing algorithm using OpenMP. Use a Tree or an undirected graph for DFS

Objective: Students should be able to perform Parallel Depth first Search based on existing algorithms using OpenMP.

Pre-requisite:
1. Basic of programming language
2. Concept of DFS
3. Concept of Parallelism

Theory:
DFS:

DFS stands for Depth-first-Search. It is a popular graph traversal algorithm that explores as far as possible along each branch before backtracking. This algorithm can be used to find the shortest path between two vertices or to traverse a graph in a systematic way. The algorithm starts at the root node and explores as far as possible along each branch before backtracking.

DFS can be implemented using either a recursive or an iterative approach. The recursive approach is simpler to implement but can lead to a stack overflow error for very large graphs. The iterative approach uses a stack to keep track of nodes to be explored and is preferred for large graphs.

DFS can also be used to detect cycles in a graph. If a cycle exists in a graph, the DFS algorithm will eventually reach a node

Page No. :
Date. : / /

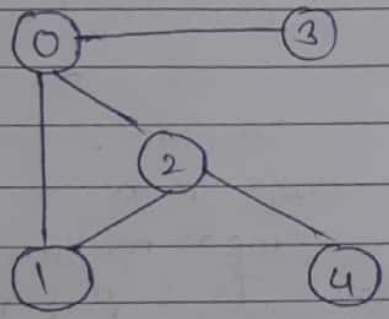
that has already been visited, indicating that a cycle exists.
A standard DFS implementation puts each vertex of graph into one of two categories

1. Visited
2. Not visited

The DFS algorithm works as follows:

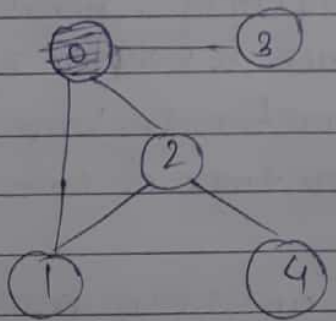
1. Start by putting any one of the graphs vertices on top of a stack.
2. Take the ~~step~~ top item of stack and add it to the visited list.
3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the top of the stack
4. Keep repeating steps 2 and 3 until the stack is empty.

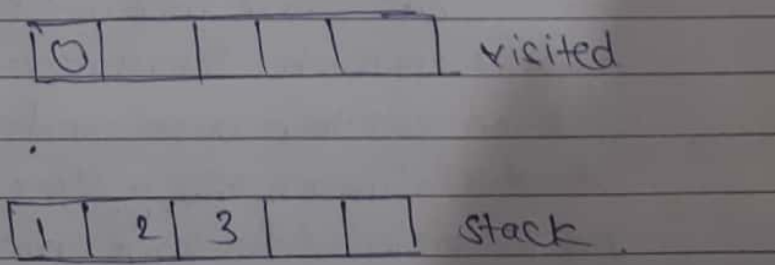
ex



undirected graph with 5 vertices

We start from vertex 0, the DFS algorithm starts by putting it in the visited list & putting all its adjacent vertices in the stack.

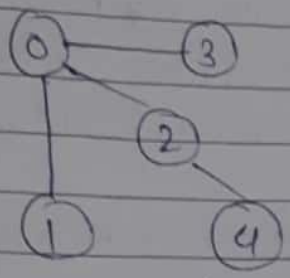


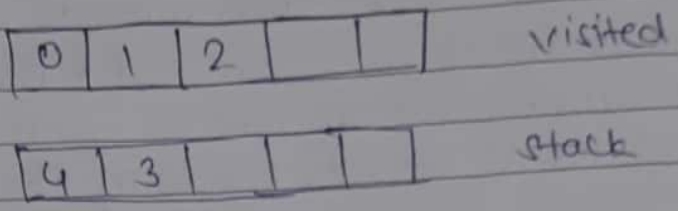


visit element & put it in visited list

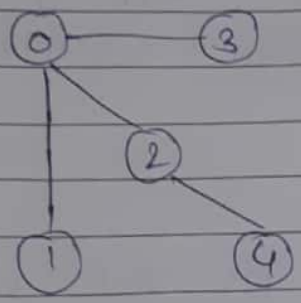
Page No. :
Date. : / /

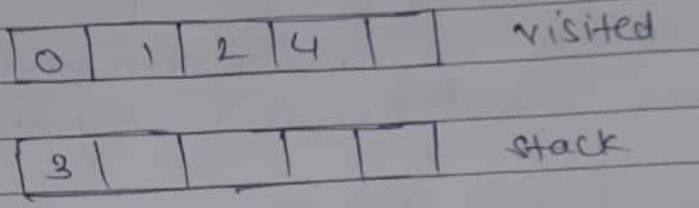
vertex 2 has an unvisited adjacent vertex in 4, so we add that to the top of the stack and visit it





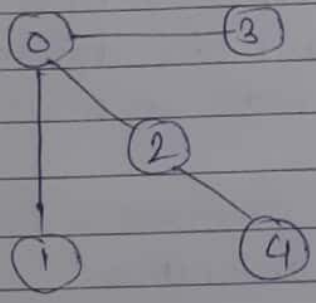
vertex 2 has an unvisited adjacent vertex in 4, so we add that to top of stack & visit it

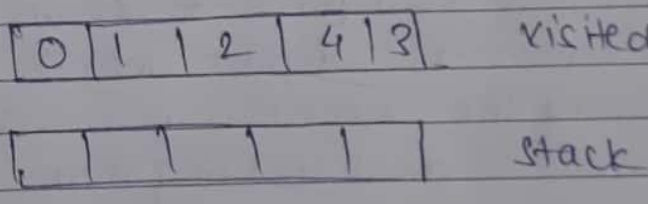




vertex 2 has an unvisited adjacent vertex in 4, so we add that to the top of the stack and visit it.

After we visit the last element 3, it doesn't have any unvisited algorithm adjacent nodes, so we have completed the Depth first Traversal of the graph.





After we visit the last element 3, it doesn't have any unvisited adjacent nodes, so we have completed the Depth first Traversal of the graph.

Page No.:
Date.: / /

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 & parallel computing clusters
- OpenMP provides a set of directives & functions that can be inserted into the source code of the program to parallelize its execution. These directives are simple & easy to use, & they can be applied to loops, section, functions, and other program constructs. The compiler then generates parallel code that can run on multiple processor concurrently

How Parallel DFS work:

- Parallel Depth first search (DFS) is an algorithm that explores the depth of a graph structures to search for nodes. In contrast to serial DFS algorithm that explores nodes in a serial manner, parallel DFS algorithm explores nodes in a parallel manner, providing a significant speedup in large graphs.
- Parallel DFS works by dividing the graph into smaller subgraphs that are explored simultaneously. Each processor or thread is assigned a subgraph to explore, and they work independently to explore the subgraph using the standard DFS algorithm. During the exploration processes, the nodes are marked as visited to avoid revisiting them.
- Parallel the subgraph, the processors maintains a stack data structure stores the nodes in order of exploration

Conclusion:- In this way we can achieve parallelism while implementing DFS.

Page No. :
Date. : / /

Practical Assignment 2

Name : Tanmay Bhagwat
Roll No : 20C0011
Sub : HPC
Class : BE Comp A.

Aim : Write a program to implement parallel Bubble sort and parallel merge sort using openMP using existing algorithms and measure the performance of sequential and parallel algorithms.

Objectives : Study of parallel sorting algorithms like bubble sort and merge sort.

Prerequisities : Students should know basic concepts of bubble sort and merge sort.

Theory : ① What is sorting?

Sorting is a process of arranging elements in a group in particular order i.e. ascending order, descending order, alphabetical order, etc.

② What is parallel sorting?

A sequential sorting algorithm may not be efficient enough when we have to sort a huge volume of data. Therefore parallel algorithms are used in sorting.

Bubble Sort : The bubble sort is to compare the two adjacent elements. If they are not in the right order switch them.

Page No. :
Date. : / /

Parallel Bubble Sort :-

- Implement as a pipeline.
- Let local size = n/no-prac. We divide the array into no-prac parts and each process executes the bubble sort on its part include comparing the last element with the first one belonging to the next thread.
- Implement with the loop (instead of if)

```
for (j=0; j<n-1; j++)
```

- for every iteration of i, each thread needs to wait until the previous thread has finished that iteration before starting.
- We'll co-ordinate using barrier.

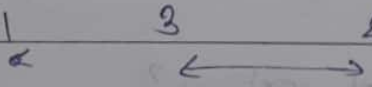
Example :- 4, 3, 1, 2

Step 1 :- 4 3 1 2

Step 2 :- 3 4 1 2

Step 3 :- 3 1 4 2

Step 4 :- 1 3 2 4



Step 5 :- 1 2 3 4

Merge sort :-

- Collects sorted list onto one processor.
- Merge elements as they come together
- Simple three structure
- Parallelism is limited when near the ~~root~~ rest.

Page No. :
Date. : / /

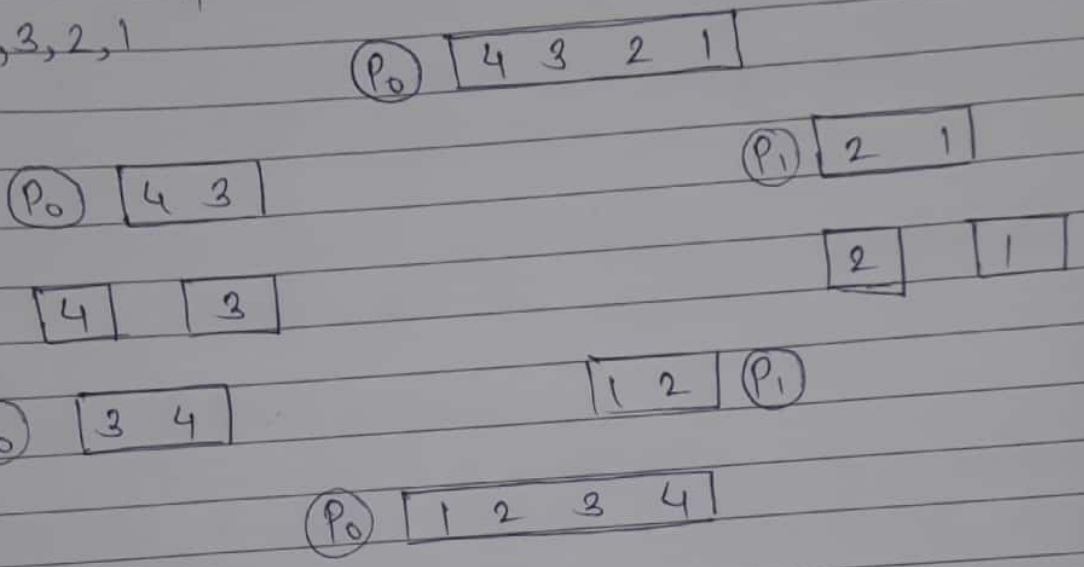
Parallel Merge Sort :-

- Parallelize processing of sub problems.
- Max parallelization achieved with one processor per node (at each layer/height).

Example :-

Perform merge sort on the following list of elements given 2 processor, P_0 & P_1, which processor is responsible for which comparison

4, 3, 2, 1



Conclusion :- Thus we have studied parallel bubble sort and parallel merge sort implementation

For Educational Use Only

Page No. :
Date. : / /

Practical Assignment 3

Name : Tanmay Bhagwat
Roll No : 20C0011
Sub : HPC
Class : BE Comp A

Aim : To implement min, max, sum and average operations using parallel reduction.

Objectives : To study and implementation of directive based parallel programming model.

Prerequistie : 64 bit open source linux or its derivation programming languages : C/C++

Theory :- Open MP :-

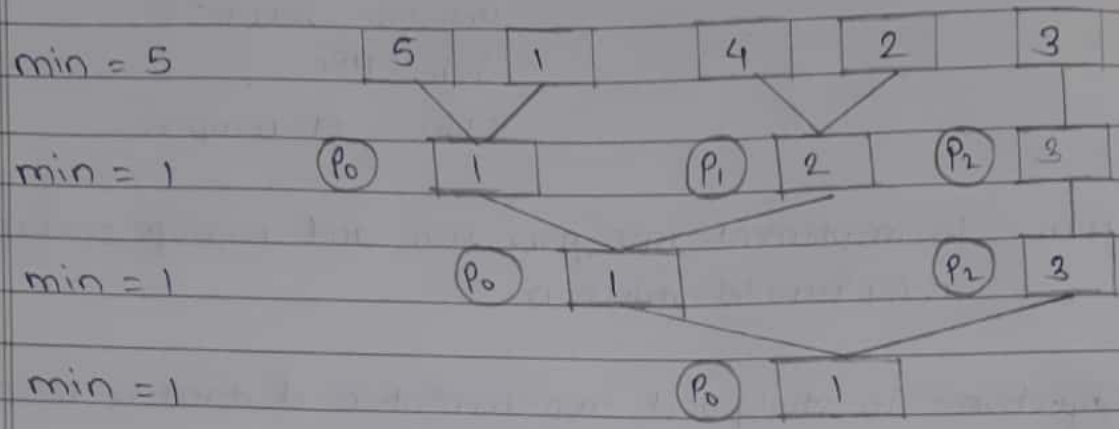
Open MP is a set of C/C++ programs (fortran equivalents) which provide the programmer a high level frontend which get translated as calls to threads (or other similar entities). The key phrase here is : "higher-level" ; the goal is to better enable the programmer to "think parallel" allevating him/her of the burden and distraction of dealing with setting up and co-ordinating threads.

Operation Minimum (Min) :-

The minimum value is the smallest value is the array. The implementation of this operation, the minimum value is initialised to the first element in array, an then compared to each sub sequent element in loop. If smaller element is found the minimum value is updated accordingly.

Page No.:
Date.: / /

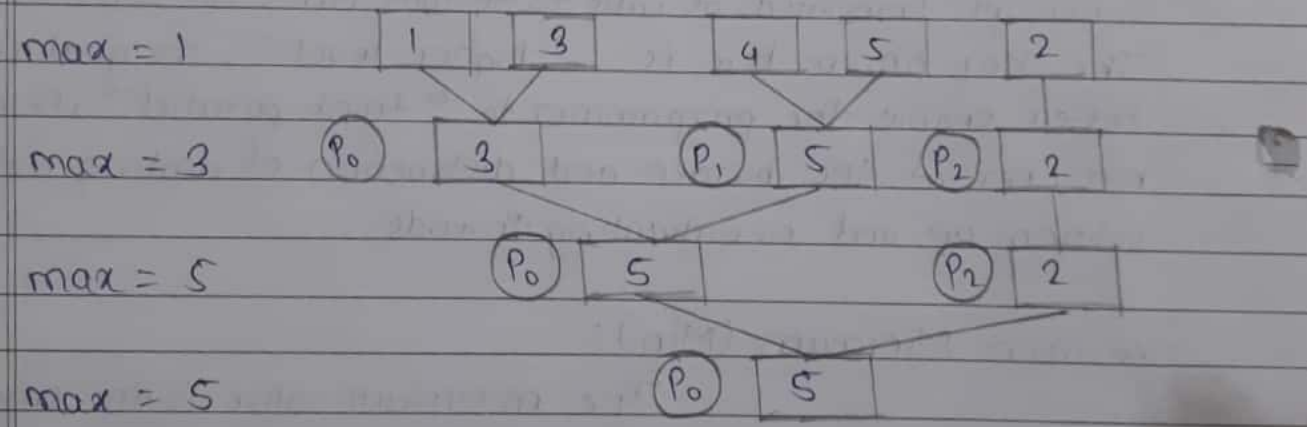
Example : [5, 1, 4, 2, 3]



Operation Maximum :

The maximum value is the largest value in array. In implementation of the operation, the maximum value is initialized to first element in the array and then compared to each subsequent element in the loop.

Example : [1, 3, 4, 5, 2]



Operation sum of values (sum) :

The sum of values is the total of the values in the array. In implementation of this operation the sum is initialized to grow and then by using parallel threads we do summation of two array elements in one thread.

Page No. :
Date. : / /

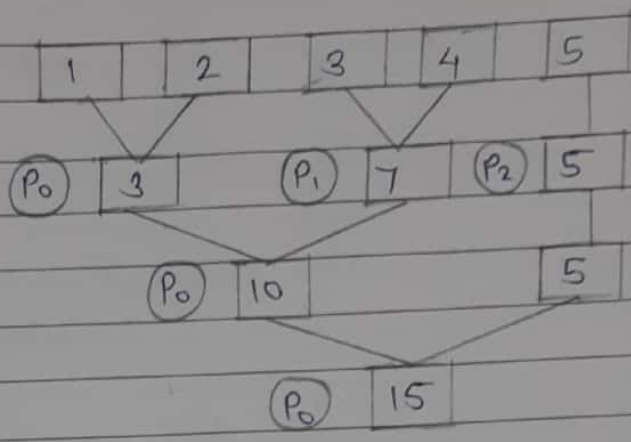
Example : [1, 2, 3, 4, 5]

Sum [$P_0 = 0$]

Sum [$P_0 = 3, P_1 = 7, P_2 = 5$]

Sum [$P_0 = 10, P_1 = 5$]

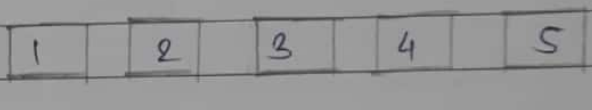
Sum [$P_0 = 15$]



Average operations (Avg) :-

The average operation value is the mean value of the all elements in the array. In implementation of this operations. The sum of all mean of two elements in an array is calculated and sorted in multiple threads.

Example : [1, 2, 3, 4, 5]

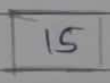


sum = 0

sum [$P_0 = 3, P_1 = 7, P_2 = 5$]

Sum [$P_0 = 10, P_1 = 5$]

sum [15]



Now avg = sum / n

∴ average avg = sum / n

= 15 / 5

= 3

∴ average = 3

Conclusion :- Hence, we successfully studied and implemented min, max, sum & average operations using parallel reduction using open MP.

Page No. :
Date. : / /

Practical Assignment - 4

Name: Tanmay Bhagwat
Roll No: 20C0011.
Class: BE Comp A.
Sub: HPC

Aim: Implement HPC application for AI/ML domain.

Pre-requisites: 64 bit open source linux or its derivative
C/C++ programming, OpenMP

Theory:

Here we will develop an application using OpenMP libraries and developing an application which will perform tokenization in C language.

Tokenization:

Tokenization is the process of breaking down a text into smaller components called tokens. Tokens can be words or phrases, sentences or any other meaningful unit of text.

Tokenization is a crucial step in any Natural Language Processing applications such as text classification, named entity recognition and machine translation. OpenMP can be used because OpenMP is a set of computer directives and library outlines for parallel computing and programming in C and other languages. OpenMP can be used parallelize the tokenization process which can improve the speed of process when dealing with larger text. The basic idea behind parallelized tokenization using OpenMP is to split the text into smaller chunks and process each chunk to parallel using multiple threads.

Page No. :
Date. : / /

To tokenize a text using openMP we can split the text into smaller chunks and assign each other chunk to a different thread each thread can then process its chunk of text independently and generate a set of tokens. To ensure that each thread processes an equal amount of text we can "omp.get thread num" function to get the ID of each thread.

OpenMP can be a powerful tool for parallelizing tokenization, its important to note that the performance gain will depend upon the specific use case and the size of text being tokenized in some cases the overhead of dividing the text into chunks and combining the results may outweight the benefits of tokenization.

The code reads the line of text from a file called "example.txt" and uses open MP to tokenize each line in parallel. Here's a breakdown of how it works.

1. The 'tokenize' function takes a single line of text as input and tokenize it using 'strtok' function from the 'string.h' library. The function uses the "amp.get thread num" function to get the ID of the current thread.
2. The "main" function initializes the number of threads to use and open the file for reading.
3. The #pragma amp parallel directive creates a parallel region.
4. The while (fgets(line, MAXLINE_LENGTH, file) = NULL) loop reads the each line of text from the file.
5. Once all threads have finished processing their chunks of text, the program clases the file and returns 0.

Page No. :
Date. : / /

Note that this code uses a critical section to ensure that each thread points its output in a synchronized manner. This is because the 'printf' function is not Thread Safe and can result in garbled output.

Conclusion : Hence we successfully developed an application using HPC & using OpenMP for AI/ML domain.

For Educational Use Only

Assignment 1

Aim: Linear regression by using Deep Neural Network. Implement Boston housing price prediction problem by Linear regression regression using Deep network. Use Boston Housing price prediction dataset

Objectives: i) To implement different deep learning models
ii) To understand Hardware acceleration
iii) To illustrate concepts of AI/ML

Requirements: 64 bit windows, Python, Jupyter Notebook.

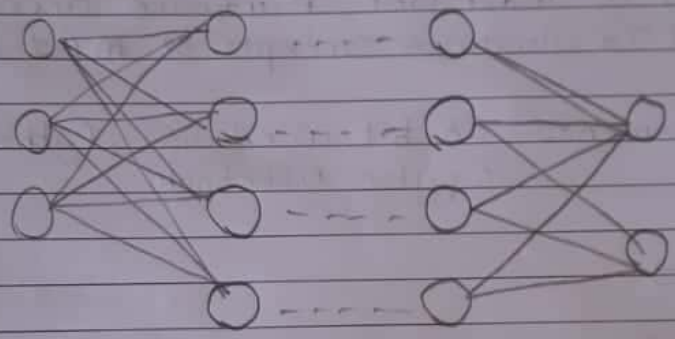
Theory

Linear Regression:
It is a simple but powerful statistical method that aims to model the relationship between a dependent variable and one or more independent variables.
In deep learning, linear regression is used as a basic building block for more complex model. In neural networks, linear regression can be used as a way to combine inputs (features) to generate a single output.

Deep Neural Network-
A deep neural network is an ANN with multiple

hidden layers between the input and output layers. Similar to shallow ANN's, Deep neural networks can model complex non-linear relationship.

The main purpose of neural network is to receive a set of inputs, perform progressively complex calculations on them and give output to solve real world problems like classification.



Input layers | Hidden layer 1 | Hidden layer N | Output layers

A linear regression neural network takes in a vector of input features multiplies each feature by a weight, adds up the weighted inputs and then passes the results through linear activation function to obtain predicted value of dependent variable mathematically as, $y = w_1 x_1 + w_2 x_2 + \ldots w_n x_n + b$
The goal is to find values of weights and bias term that minimize the difference between predicted values and actual values of dependent.

Rajdhani
DATE / /

variables. This is achieved by using a loss function such as mean squared error (MSE) which measures the average squared difference between them.

Algorithm:

i) Import all python libraries required such as tensorflow, numpy, pandas, matplotlib, seaborn, etc.

ii) Load the dataset and split it into training dataset and testing dataset.

iii) Conduct exploratory analysis on both training and testing such as:
   a) Check data shape and type
   b) Converting data to dataframe using pandas library.
   c) View the datasets
   d) Perform preprocessing on datasets.

iv) Create Deep Neural Network model.
Train and Test the created model.

v) Model Evaluation:
Preview the mean value of training and validation data.
Evaluate model on the test data.
Plot the loss curves.

Rajdhani
DATE / /

vi) View the model predictions

Conclusion:
Hence in this assignment we learned how to implement linear regression and deep neural network models to predict the price of house in Boston using Boston housing price prediction dataset.

Assignment 2

Aim: Classification using Deep Neural Network
Binary classification using Deep Neural Network classify movie reviews into 'positive' reviews and 'negative' reviews, just based on text content of the reviews use IMDB dataset

Dataset Description:
We will use IMDB dataset which contains 50000 movie reviews that are labelled as 'positive' or 'negative'. The dataset is split into 25000 reviews for training and 25000 reviews for testing

Objectives: i) To implement different deep learning models.
ii) To understand Hardware acceleration
iii) To illustrate concepts of AI / ML

Requirements: 64 bit windows, Python, Jupyter Notebook

Theory

Binary Classification:
Binary classification is a type of machine learning problem where the task is to classify data into two categories. In this practical assignment we will use Deep Neural Networks to perform binary classification of movie review based on their text content.

Rajdhani
DATE / /

Deep Neural Networks are a type of machine learning model that are capable of learning complex patterns of data.
In machine learning, binary classification is a supervised learning algorithm that categorizes new observations into one of two classes.

Algorithm:

i) Load the dataset using built in function in keras

ii) Pre-process the dataset by converting the integer sequence into a binary matrix using one-hot encoding.

iii) Split the training dataset into training and validation sets.

iv) Implement a deep neural network with following architecture
- An embedding layer to convert the integer sequence into desire vectors of fixed size
- two dense layers with ReLU activation function.
- A final state dense layer with a sigmoid activation function to output probability for 'positive' or 'negative'.

Rajdhani

Rajdhani
DATE / /

v) Train the model using adam optimizer and binary cross-entropy loss function.

vi) Evaluate the model on the test dataset and report the accuracy and loss.

vii) Experiment with different hyperparameters such as number of hidden layers and learning rate and evaluate the model performance.

viii) Solve the trained model for future use.

ix) END

Conclusion: Hence we have successfully implemented binary classification for IMDB dataset.

Assignment 3

Title: Convolutional neural network : Use MNIST fashion dataset and create a classifier to classify fashioning clothing into categories

Objectives: Apply the technique of Deep Learning models with convolutional method.

Prerequisites:
1) The concept of Convolutional Neural Networks
2) Basics of deep neural network

Requirements:
1) Jupyter Notebook
2) Python and its libraries.

Theory:

Convolutional Neural Network (CNN)

It is a type of a neural network commonly used in Deep learning for image recognition classification and segmentation tasks.
The architecture of a CNN is designed to hierarchical of features from input images.

In a CNN, the input image is first passed through a series of convolutional layers each

Rajdhani
DATE / /

of which applies a set of filters to input image to extract features at different spatial scales.
These features are then passed through a pooling layer, which reduces the dimensionality of features while preserving their important spatial information.

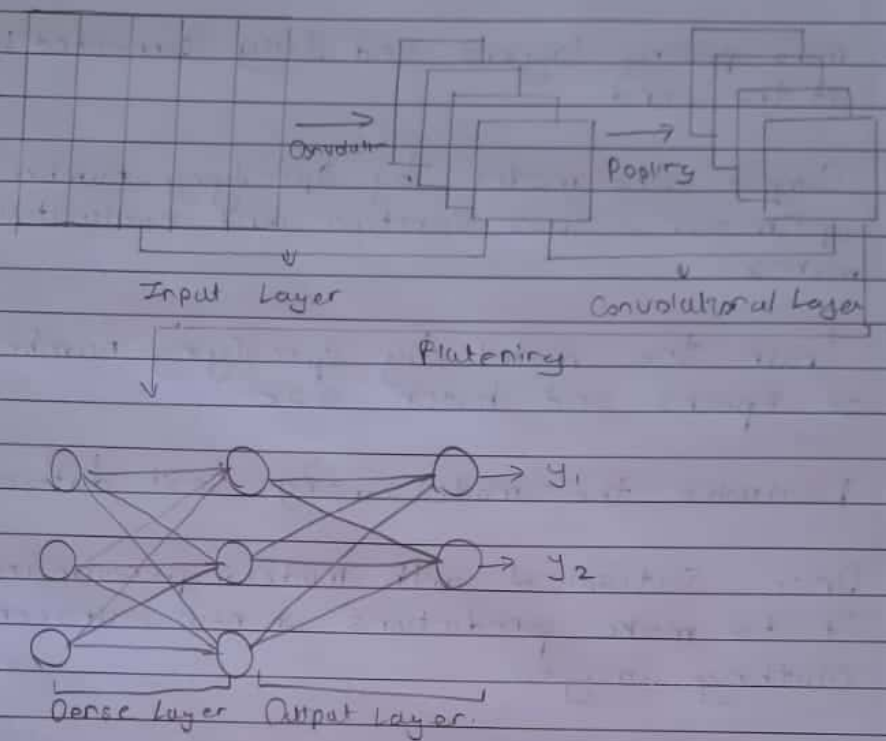
Finally, the output of the last pooling layer is passed through one or more fully connected layers, which perform classification or regression based on learnt features.

CNN's are typically trained using the back propagation algorithm along with optimization techniques. During training, CNN learns to automatically extract relevant features from input data. The network adjust its weight and biased to minimize a defined a loss function.

Applications of CNN include:

1) Image Classification
2) Object Detection
3) Image Segmentation
4) Natural Language Processing

Rajdhani
DATE / /



Algorithm:

1) Input all the necessary libraries like: tensorflow, pandas, numpy, etc.

2) Load dataset and split it into training and testing dataset and normalize data to values between 0 and 1.

3) Define the CNN architecture that consists of multiple convolutional layers followed by

Rajdhani
DATE / /

Rajdhani
DATE / /

max-pooling layers and fully connected layers at the end.

4) Compile the model by specifying number of optimizer, loss function and evaluation matrix.

5) Train the model by specifying number of epochs and batch size.

6) Evaluate the model using test dataset

7) Once satisfied with model's performance, use it to make predictions on new, unseen fashion clothing images.

Conclusion:
Hence, a model to create CNN classifier using MNIST fashion dataset to classify fashion clothing into categories was implemented.

Page No. :
Date. : / /

Practical Assignment 4

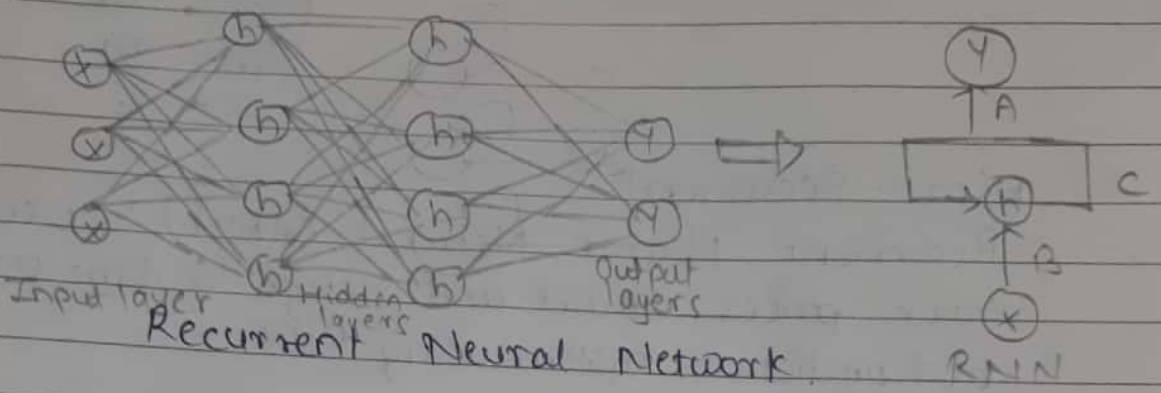
Name : Tanmay Bhagwat
Roll No : 20C011
Class : BE comp A
Sub : DL

Aim : Recurrent Neural Network (RNN) : Use the Google stock prices dataset and design a time series analysis and prediction system using RNN

Requirement :
64 bit operating systems, Python installation, Jupyter Notebook, Python Libraries, Tensorflow, Numpy and Pandas.

Theory :
Recurrent Neural Networks :
Recurrent Neural Networks (RNN) is a type of Neural Network where the output from the previous step is fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other. Still in cases when it is required to predict the next word of a sentence, the previous words are required and hence there is need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a hidden layer. The main and most important features of RNN is its Hidden State, which remembers some information about a sequence. The state is also referred to as Memory State since it remembers the previous input to the network. It uses the same parameters for each input as it perf performs the same task on all the inputs or

hidden layers to produce the output. This reduces the complex of parameters, unlike other neural networks.



Steps involved in RNN

1) Data Preprocessing

2) Initialize Parameters.

3) Define the RNN Architecture: Choose the type of RNN at architecture for you want to use (e.g. Vanilla RNN, LSMT, GRU)

4] Forward Propagation

5] Compute Loss.

6) Backpropogation Through Time

7) Update Parameters

8] Repeat Training.

Page No. :
Date. : / /

9] Evaluation

10] Prediction

11] Post-processing:- Depending on the specific task, decide post-processing steps such as decoding, converting probabilities to class labels, etc.

Conclusion :-

Hence we prepared a model for the Google Stock prices dataset. We also designed a time series analysis and prediction system using RNN.