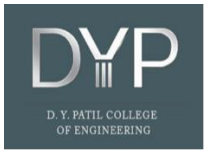


D. Y. PATIL COLLEGE OF ENGINEERING, AKURDI, PUNE

Department of Artificial Intelligence and Data Science

2023-24

LAB MANUAL
Computer Laboratory-IV
(BE)
Semester II

D Y Patil College of Engineering, Akurdi, Pune

Department of Artificial Intelligence & Data Science

Vision of Institute

"Empowerment through knowledge"

Mission of Institute

1. To educate the students to transform them as professionally competent and quality conscious engineers.
2. To Provide Conducive Environment for Teaching Learning and overall personality development.
3. To culminate the Institute into an International seat of excellence

VISION of Department

Developing highly skilled and competent IT professional for sustainable growth in the field of Artificial Intelligence and Data science.

MISSION of Department

1. To empower students for developing intelligent systems and innovative products for societal problems.
2. To build strong foundation in Data computation, Intelligent Systems that enables self-development entrepreneurship and Intellectual property.
3. To develop competent and skilled IT professional by imparting global skills and technologies for serving society.

Program Outcomes (PO's)

PO1	Engineering knowledge	Apply the knowledge of mathematics, science, Engineering fundamentals, and an Engineering specialization to the solution of complex Engineering problems.
PO2	Problem analysis	Identify, formulate, review research literature and analyze complex Engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and Engineering sciences.
PO3	Design / Development of Solutions	Design solutions for complex Engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and Environmental considerations.
PO4	Conduct Investigations of Complex Problems	Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	Modern Tool Usage	Create, select, and apply appropriate techniques, resources, and modern Engineering and IT tools including prediction and modeling to complex Engineering activities with an understanding of the limitations.
PO6	The Engineer and Society	Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practices.
PO7	Environment and Sustainability	Understand the impact of the professional Engineering solutions in societal and Environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	Ethics	Apply ethical principles and commit to professional ethics and responsibilities and norms of Engineering practice.
PO9	Individual and Team Work	Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	Communication Skills	Communicate effectively on complex Engineering activities with the Engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	Project Management and Finance	Demonstrate knowledge and understanding of Engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary Environments.
PO12	Life-long Learning	Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSO's)

PSO1	Professional Skills	The ability to understand, analyze and develop computer programs in the areas related to algorithms, system software, multimedia, web design, networking, artificial intelligence and data science for efficient design of computer-based systems of varying complexities.
PSO2	Problem-Solving Skills	The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product for business success.
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 to have a zest for higher studies.

Program Educational Objective (PEO's)

PEO1	To prepare globally competent graduates having strong fundamentals and domain knowledge to provide effective solutions for engineering problems.
PEO2	To prepare the graduates to work as a committed professional with strong professional ethics and values, sense of responsibilities, understanding of legal, safety, health, societal, cultural and environmental issues.
PEO3	To prepare committed and motivated graduates with research attitude, lifelong learning, investigative approach, and multidisciplinary thinking.
PEO4	To prepare the graduates with strong managerial and communication skills to work effectively as individual as well as in teams.

Computer Laboratory -IV

Course Code	Course Name	Teaching Scheme (Hrs./ Week)	Credits
417532 D	Computer Laboratory-IV: Deep Learning	2	2
417533 B	Computer Laboratory-IV: Business Intelligence	2	2
417532B	Computer Laboratory-IV: Big Data Analysis	2	2
417533D	Computer Laboratory-IV: Reinforcement Learning	2	2

Course Objectives:

- To understand the fundamental concepts and techniques of Virtual reality
- To understand Big Data Analytics Concepts
- To learn the fundamentals of software development for portable devices
- To understand fundamental concepts of Deep Learning
- To be familiar with the various application areas of augmented realities
- To introduce the concepts and components of Business Intelligence (BI)
- To understand the concepts of Information Systems

Course Outcomes:

On completion of the course, learner will be able to–

CO1: Apply basic principles of elective subjects to problem solving and modeling
CO2: Use tools and techniques in area of software development to build mini projects
CO3: Design and develop applications on subjects of their choice
CO4: Implement and manage deployment, administration & security

Operating System recommended: Practical can be performed on suitable development platform

Table of Contents

	Big Data Analytics		
11	Develop a MapReduce program to calculate the frequency of a given word in a given file.		
12	Develop a MapReduce program to find the grades of student's**.**		
13	Implement Matrix Multiplication using Map Reduce		
14	Develop a MapReduce program to analyze Titanic ship data and to find the average age of the people (both male and female) who died in the tragedy. How many persons are survived in each class.		
15	Hive: Introduction Creation of Database and Table, Hive Partition, Hive Built in Function and Operators, Hive View and Index.		
	Reinforcement Learning		
16	Study of the TensorFlow and OpenAI Gym Library		
17	Implement a Deep Q-Network (DQN) using a deep neural network library (e.g., TensorFlow or PyTorch) and train it on a simple environment like CartPole or Mountain Car.		
18	Implement a policy gradient algorithms like REINFORCE or Proximal Policy Optimization (PPO) to solve a continuous control task		
19	Build a multi-agent environment, such as a cooperative or competitive game, and implement algorithms like Independent Q-Learning or Multi-Agent Deep Deterministic Policy Gradients (MADDPG)		
20	Develop an actor-critic model using neural networks and train it on a classic RL benchmark, such as the Acrobot or Inverted Pendulum.		

Lab Assignment No.	1
Title	Problem Statement – Real estate agents want help to predict the house price for regions in the USA. He gave you the dataset to work on and you decided to use the Linear Regression Model. Create a model that will help him to estimate what the house would sell for. URL for a dataset: https://github.com/huzaifsayed/Linear-Regression-Model-for-House-PricePrediction/blob/master/USA_Housing.csv
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-III-Computational Intelligence
Assessment Marks	
Assessor's Sign	

ASSIGNMENT No: 01

Title:- Problem Statement – Real estate agents want help to predict the house price for regions in the USA. He gave you the dataset to work on and you decided to use the Linear Regression Model. Create a model that will help him to estimate what the house would sell for. URL for a dataset:

https://github.com/huzaifsayed/Linear-Regression-Model-for-House-Price Prediction/blob/master/USA_Housing.csv

Objective:

Develop a model to estimate house prices based on relevant features using Linear Regression.

Outcome-

- Apply Linear Regression in a real-world scenario.
- Understand the implementation of a regression model for house price prediction.

Software Requirements -

- Python (3.x recommended)
- Jupyter Notebook or any Python IDE

Hardware Requirement

A machine with sufficient RAM and processing power for model training (8GB RAMrecommended)

Prerequisites -

- Basic understanding of Python programming
- Familiarity with the concepts of supervised learning

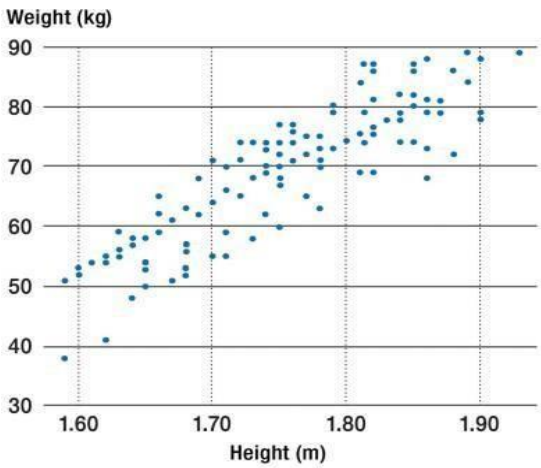
What is Linear Regressoin?

When we see a relationship in a scatterplot, we can use a line to summarize the relationship in the data. We can also use that line to make predictions in the data. This process is called linear regression.Linear Regression is a supervised learning algorithm used for predicting a continuous outcome, typically represented by the target variable. In the context of this assignment, we aim to predict house prices based on various features such as area income, house age, number of rooms, and others.

Linear regression is used to study the linear relationship between a dependent variable Y (blood pressure) and one or more independent variables X (age, weight, sex).

The dependent variable Y must be continuous, while the independent variables may be either continuous (age), binary (sex), or categorical (social status). The initial judgment of a possible relationship between two continuous variables should always be made on the basis of a scatter

plot (scatter graph). This type of plot will show whether the relationship is linear (figure 1) or nonlinear (figure 2).



Com

Figure 1 A scatter plot showing a linear relationship

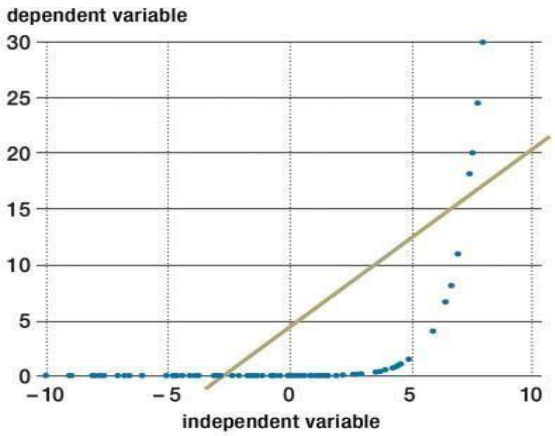


Figure 2 A scatter plot showing an exponential relationship. In this case, it would not be appropriate to compute a coefficient of

Simple linear regression formula–

The formula for a simple linear regression is:

$$y = \beta_0 + \beta_1 X + \epsilon$$

• y is the predicted value of the dependent variable (y) for any given value of the independent variable (x).

- B_0 is the intercept , the predicted value of y when the x is 0.
- B_1 is the regression coefficient – how much we expect y to change as x increases.
- x is the independent variable (the variable we expect is influencing y).
- e is the error of the estimate, or how much variation there is in our estimate of the regression coefficient.

Linear regression finds the line of best fit line through your data by searching for the regression coefficient (B_1) that minimizes the total error (e) of the model.

While you can perform a linear regression by hand, this is a tedious process, so most people use statistical programs to help them quickly analyze the data.

Model Training –

Training a regression model involves teaching the model to predict continuous values based on input features. Here's a brief explanation of the process, along with some images to illustrate key concepts.
Regression Model Training:

1. Data Collection: Gather a dataset with input features (independent variables) and corresponding target values (dependent variable).
2. Data Splitting: Split the dataset into training and testing sets. The training set is used to train the model, and the testing set is used to evaluate its performance.
3. Model Selection: Choose a regression model architecture. Common choices include linear regression, decision trees, or more complex models like neural networks.
4. Feature Scaling: Normalize or standardize the input features to ensure that they are on a similar scale. This helps the model converge faster during training.
5. Model Training: Feed the training data into the chosen model and adjust the model's parameters to minimize the difference between predicted and actual target values.
6. Loss Function: Use a loss function to measure the difference between predicted and actual values. The goal is to minimize this loss during training.
7. Gradient Descent: Use optimization algorithms like gradient descent to iteratively update the model parameters and reduce the loss.
8. Model Evaluation: Evaluate the trained model on the testing set to assess its performance on unseen data.
9. Prediction: Once satisfied with the model's performance, use it to make predictions on new, unseen data.

10 Model Deployment: If the model performs well, deploy it to production for making real-world predictions.

Evaluation Metrics –

The performance of the model will be assessed using various evaluation metrics, such as Mean Squared Error (MSE) and R-squared. These metrics provide insights into how well the model

generalizes to unseen data.

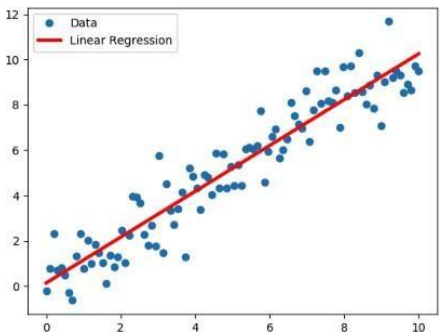


Figure: Linear Regression Model

Algorithm –

Simple Linear Regression Algorithm –

You ll start with the simplest case, which is simple linear regression. There are five basic steps when you re implementing linear regression:

1. Import the packages and classes that you need.

2. Provide data to work with, and eventually do appropriate transformations.
3. Create a regression model and fit it with existing data.
4. Check the results of model fitting to know whether the model is satisfactory.
5. Apply the model for predictions.

These steps are more or less general for most of the regression approaches and implementations. Throughout the rest of the tutorial, you'll learn how to do these steps for several different scenarios.

Step 1: Import packages and classes

The first step is to import the package numpy and the class LinearRegression from sklearn.linear_model:

```
Python
>>> import numpy as np
>>> from sklearn.linear_model import LinearRegression
```

Now, you have all the functionalities that you need to implement linear regression.
The fundamental data type of NumPy is the array type called numpy.ndarray. The rest of this tutorial uses the term **array** to refer to instances of the type numpy.ndarray.

You'll use the class sklearn.linear_model.LinearRegression to perform linear and polynomial regression and make predictions accordingly.

Step 2: Provide data

The second step is defining data to work with. The inputs (regressors, x) and output (response,) should be arrays or similar objects. This is the simplest way of providing data for regression:

```
Python
>>> x = np.array([5, 15, 25, 35, 45, 55]).reshape((-1, 1))
>>> y = np.array([5, 20, 14, 32, 22, 38])
```

Now, you have two arrays: the input, x, and the output, y. You should call .reshape() on x because this array must be **two-dimensional**, or more precisely, it must have **one column** and **as many rows as necessary**. That's exactly what the argument (-1, 1) of .reshape() specifies.

This is how x and y look now:

```
Python
>>> x

array([[ 5],

[15],

[25],

[35],


>>> y

array([5, 20, 14, 32, 22, 38])
```

As you can see, x has two dimensions, and x.shape is (6, 1), while y has a single dimension, and y.shape is (6,).

Step 3: Create a model and fit it

The next step is to create a linear regression model and fit it using the existing data.

Create an instance of the class LinearRegression, which will represent the regression model and output, x and y, as the arguments. In other words, .fit() fits the model. It returns self, which is the variable model itself. That's why you can replace the last two statements with this one:

```
Python
>>> model = LinearRegression().fit(x, y)
```

This statement does the same thing as the previous two. It's just shorter.

Step 4: Get results

Once you have your model fitted, you can get the results to check whether the model works satisfactorily and to interpret it. You can obtain the coefficient of determination, R^2, with .score() called on model:

```
Python
>>> r_sq = model.score(x, y)

>>> print(f"coefficient of determination: {r_sq}")
```

When you're applying .score(), the arguments are also the predictor x and response y, and the return value is R^2.

The attributes of model are .intercept_, which represents the coefficient b_0, and .coef_, which represents b_1:

```
Python
>>> print(f"intercept: {model.intercept_}")

intercept: 5.633333333333329

>>> print(f"slope: {model.coef_}")
```

slope: [0.54]

The code above illustrates how to get b_0 and b_1. You can notice that .intercept_ is a scalar, while .coef_ is an array.

The value of b_0 is approximately 5.63. This illustrates that your model predicts the response 5.63 when x is zero. The value $b_1 = 0.54$ means that the predicted response rises by 0.54 when x is increased by one.

You'll notice that you can provide y as a two-dimensional array as well. In this case, you'll get a similar result. This is how it might look:

```
predicted response:
[ 8.33333333 13.73333333 19.13333333 24.53333333 29.93333333 35.33333333]
```

When applying .predict(), you pass the regressor as the argument and get the corresponding predicted response. This is a nearly identical way to predict the response:

```
Python
>>> y_pred = model.intercept_ + model.coef_ * x

>>> print(f"predicted response:  \n{y_pred}") predicted
response:

[[ 8.33333333]

[13.73333333]

[19.13333333]
```

In this case, you multiply each element of x with model.coef_ and add model.intercept_ to the product.

The output here differs from the previous example only in dimensions. The predicted response is now a twodimensional array, while in the previous case, it had one dimension.

If you reduce the number of dimensions of x to one, then these two approaches will yield the same result. You can do this by replacing x with x.reshape(-1), x.flatten(), or x.ravel() when multiplying it with model.coef_.

Application -

estate pricing strategies

1. Assisting clients in making informed decisions

2. Market analysis for regions in the USA
3. Predictive tool for estimating house prices

Reference : -

process involves exploring the data, data collection, model fitting, and interpretation of regression results. Practical skills in using statistical software for regression analysis are likely to be developed, enhancing students ability to analyze relationships between variables and make predictions based on linear model.

References – https://en.wikipedia.org/wiki/Linear_regression

https://www.khanacademy.org/math/statistics-probability/describing-relationships-quantitativedata/introduction-to-trend-lines/a/linear-regression-review

https://github.com/huzaifsayed/Linear-Regression-Model-for-House-Price- Prediction/blob/master/linear-regression-model.jpg

Conclusion: Thus Implemented a Linear Regression Model to predict house prices for regions in the USA using the provided dataset.

Lab Assignment No.	02
Title	Build a Multiclass classifier using the CNN model. Use MNIST or any other suitable dataset. a. Perform Data Pre-processing b. Define Model and perform training c. Evaluate Results using confusion matrix.
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV: Deep Learning
Assessment Marks	
Assessor's Sign	

ASSIGNMENT No: 02

ASSIGNMENT No: 02

Title:- Build a Multiclass classifier using the CNN model. Use MNIST or any other suitable dataset. a. Perform Data Pre-processing
d. Define Model and perform training
e. Evaluate Results using confusion matrix.

Mapping with Syllabus –

Unit 3

Objective –

Building a multiclass classifier using a Convolutional Neural Network (CNN) using MNIST or any other suitable dataset. It involves several steps, including data pre-processing, defining the model architecture, training the model, and evaluating its performance using a confusion matrix.

Outcome –

Implement the technique of Convolution neural network (CNN)

Software Requirements –

- Python (3.x recommended)
- TensorFlow (Deep learning framework for building CNNs)
- Jupyter Notebook, any Python IDE, or Google Colab (for running Python code)

Hardware Requirements –

- A machine with at least 8GB of RAM is recommended for model training.
- A multi-core CPU is suitable, and for faster training, a GPU (Graphics Processing Unit) is highly recommended.

Prerequisites –

- Basic understanding of Python programming
- Familiarity with the concepts of Neural Networks, especially Convolutional Neural Networks (CNNs)

Dataset –

https://github.com/AmritK10/MNIST-CNN

Libraries or Modules Used –

- Numpy - for linear algebra.
- Pandas - for data analysis.
- Matplotlib - for data visualization. • Tensorflow - for neural networks.

Theory –

ANN or Artificial Neural Network is a multi-layer fully-connected neural net that consists of many layers, including an input layer, multiple hidden layers, and an output layer. This is a very popular deep learning algorithm used in various classification tasks like audio and words Similarly, we have Convolutional Neural Networks(CNNs) for image classification.

CNNis basically a model known to be Convolutional Neural Network and in recent times it has gained a lot of popularity because of its usefulness. CNN uses multilayer perceptrons to do computational works. CNN uses relatively little pre-processing compared to other image classification algorithms. This means the network learns through filters that in traditional algorithms were hand-engineered. So, for the image processing tasks CNNs are the best- suited option.
Applying a Convolutional Neural Network (CNN) on the MNIST dataset is a popular way to learn about and demonstrate the capabilities of CNNs for image classification tasks. The

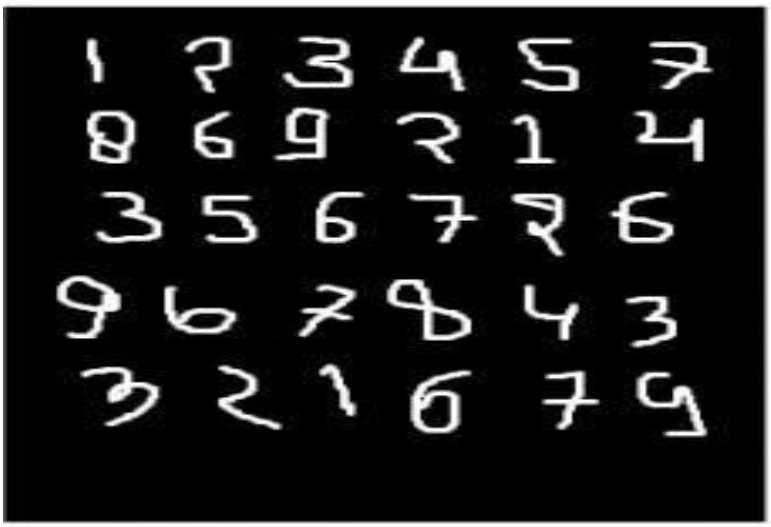
MNIST dataset consists of 28×28 grayscale images of hand-written digits (0-9), with a training set of 60,000 examples and a test set of 10,000 examples.

Here is a basic approach to applying a CNN on the MNIST dataset using the Python programming language and the Keras library:

1. Load and preprocess the data: The MNIST dataset can be loaded using the Keras library, and the images can be normalized to have pixel values between 0 and 1.
2. Define the model architecture: The CNN can be constructed using the Keras Sequential API, which allows for easy building of sequential models layer-by-layer. The architecture should typically include convolutional layers, pooling layers, and fully-connected layers.
3. Compile the model: The model needs to be compiled with a loss function, an optimizer, and a metric for evaluation.
4. Train the model: The model can be trained on the training set using the Keras fit() function. It is important to monitor the training accuracy and loss to ensure the model is converging properly.
5. Evaluate the model: The trained model can be evaluated on the test set using the Keras evaluate() function. The evaluation metric typically used for classification tasks is accuracy.

MNIST dataset:

mnist dataset is a dataset of handwritten images as shown below in the image



We can get 99.06% accuracy by using CNN(Convolutional Neural Network) with a functional model. The reason for using a functional model is to maintain easiness while connecting the layers.

Algorithm –

1. Import Libraries:
   - Import necessary libraries, including TensorFlow and Keras.

2. Load and Pre-process the MNIST Dataset:
   - Load the MNIST dataset, which consists of 28x28 grayscale images of handwritten digits (0 through 9).
   - Pre-process the data by normalizing pixel values (between 0 and 1), reshaping images, and one-hot encoding labels.

3. Define CNN Model Architecture:
   - Design the CNN architecture with convolutional layers, pooling layers, and fully connected layers.
   - Use activation functions like ReLU to introduce non-linearity.
   - The final layer has 10 units with softmax activation for multiclass classification.

4. Compile the Model:
   - Specify the optimizer (e.g., 'adam'), loss function (e.g., 'categorical_crossentropy' for multiclass classification), and evaluation metric (e.g., 'accuracy').

5. Train the Model:
   - Train the CNN using the training dataset.
   - Specify the number of epochs (passes through the entire dataset) and batch size.

6. Evaluate the Model:
   - Evaluate the trained CNN on the test dataset to assess its performance.

- Measure metrics such as accuracy to understand how well the model generalizes to unseen data.

These steps provide a comprehensive overview of the process involved in building and training a CNN for image classification using the MNIST dataset. Adjustments to these steps can be made based on specific model requirements and task objectives.

Application –

1. **Handwritten Digit Recognition:**

Recognizes handwritten digits (0-9) with applications in automated systems.

2. **Automatic Check Processing:**

Processes checks by recognizing handwritten amounts and account numbers.

3. **Postal Code Recognition:**

Recognizes postal codes on envelopes for automated mail sorting

4. **Document Classification:**

- Classifies documents based on handwritten patterns or characters.

5. **Medical Imaging:**

- Analyzes medical images for detecting anomalies or identifying patterns.

6. **Character Recognition in Forms:**

Recognizes handwritten characters in forms for efficient data entry

7. **Gesture Recognition:**

Recognizes hand gestures for sign language translation or device control

8. **Product Label Recognition:**

- Reads and interprets product labels for inventory or quality control.

9. **Traffic Sign Recognition:**

- Identifies handwritten or printed characters on traffic signs for intelligent transportation.

10. **Historical Document Analysis:**

- Analyzes historical handwritten documents for digitization and preservation.

Inference –

CNN excels in diverse applications: finance (check processing), healthcare (image analysis), retail (label recognition), and transportation (sign recognition). It aids digitization efforts and ensures cultural heritage preservation. Overall, CNN offers valuable automation and efficiency across industries.

References:

https://www.geeksforgeeks.org/applying-convolutional-neural-network-on-mnist-dataset/

https://www.youtube.com/watch?v=9cPMFTwBdM4

Conclusion: Thus I Build a Multiclass classifier using the CNN model using MNIST or any other suitable dataset by Performing Data Pre-processing, Defining Model and perform training and Evaluating Results using confusion matrix.

Lab Assignment No.	03
Title	Design RNN or its variant including LSTM or GRU a) Select a suitable time series dataset. E.g - Predict sentiments based on product reviews. b) Apply for prediction
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV- Deep learning
Assessment Marks	
Assessor's Sign	

ASSIGNMENT No: 03

Title:-

Design RNN or its variant including LSTM or GRU
a) Select a suitable time series dataset. E.g - Predict sentiments based on product reviews. b) Apply for prediction

Mapping with Syllabus –

Unit 4

Objective –

Implement a Recurrent Neural Network (RNN) or its variant (LSTM or GRU) on a selected time series dataset, such as predicting sentiments based on product reviews, to develop a predictive model for sentiment analysis.

Outcome –

Solve the language translation problem by Recurrent neural network(RNN)

Software Requirements –

- Python (3.x recommended)
- Jupyter Notebook or any Python IDE or Google Colab

Hardware Requirements –

A machine with sufficient RAM and processing power for model training (8GB RAM recommended)

Prerequisites –

- Basic understanding of Python programming
- Familiarity with the concepts of Neural Networks

Dataset –

Inbuilt tensorflow-keras-imdb dataset

Libraries or Modules Used

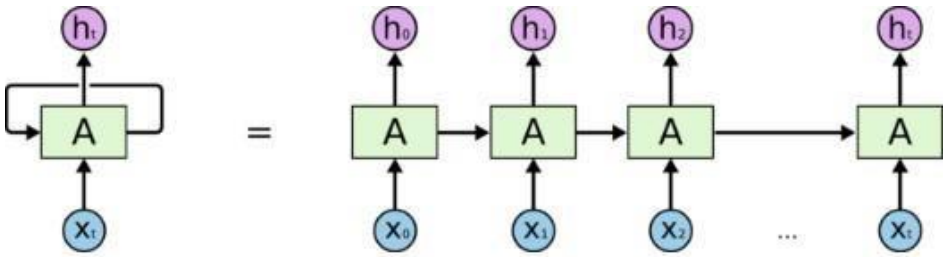
- Keras
- Tensorflow

Theory –

Recurrent Neural Network (RNN)
Recurrent Neural Network is a generalization of feedforward neural network that has an internal memory. RNN is

recurrent in nature as it performs the same function for every input of data while the output of the current input depends on the past one computation. After producing the output, it is copied and sent back into the recurrent network. For making a decision, it considers the current input and the output that it has learned from the previous input.

Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. In other neural networks, all the inputs are independent of each other. But in RNN, all the inputs are related to each other.



An unrolled recurrent neural network.

First, it takes the X(0) from the sequence of input and then it outputs h(0) which together with X(1) is the input for the next step. So, the h(0) and X(1) is the input for the next step. Similarly, h(1) from the next is the input with X(2) for the next step and so on. This way, it keeps remembering the context while training The formula for the current state is

.

$$h_t = f(h_{t-1}, x_t)$$

Applying Activation Function:

$$h_t = \tanh (W_{hh}h_{t-1} + W_{xh}x_t)$$

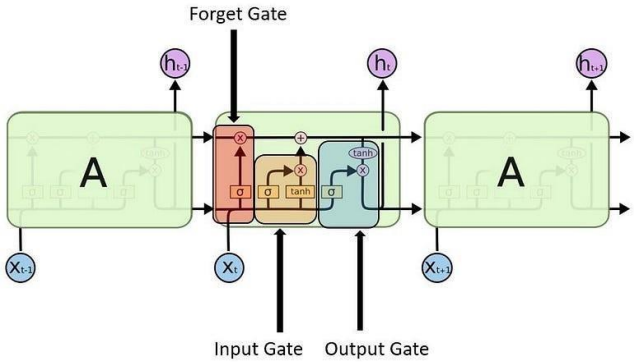
W is weight, h is the single hidden vector, Whh is the weight at previous hidden state, Whx is the weight at current input state, tanh is the Activation funtion, that implements a Non-linearity that squashes the activations to the range[-1.1]

$$y_t = W_{hy}h_t$$

Yt is the output state. Why is the weight at the output state.

Long Short Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are a modified version of recurrent neural networks, which makes it easier to remember past data in memory. The vanishing gradient problem of RNN is resolved here. LSTM is well-suited to classify, process and predict time series given time lags of unknown duration. It trains the model by using back-propagation. In an LSTM network, three gates are present:



1) Input gate - discover which value from input should be used to modify the memory. Sigmoid function decides which values to let through 0,1. and tanh function gives weightage to the values which are passed deciding their level of importance ranging from-1 to 1.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

2) Forget gate - discover what details to be discarded from the block. It is decided by the sigmoid function. it looks at the previous state(ht-1) and the content input(Xt) and outputs a number between 0(omit this)and 1(keep this)for each number in the cell state Ct−1.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

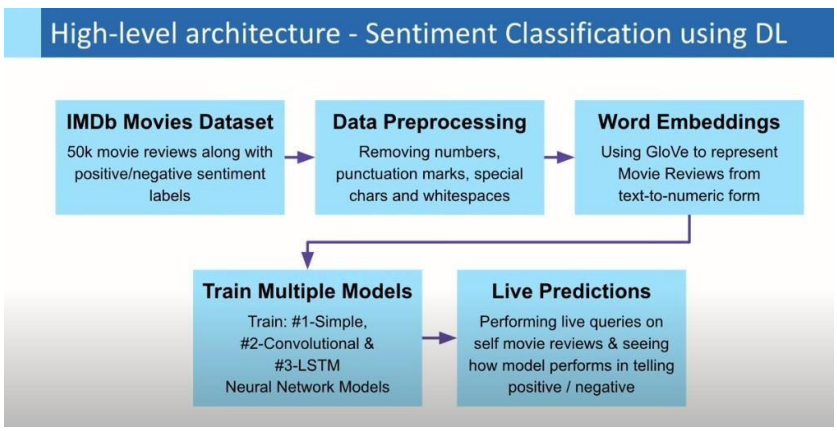
3) Output gate — the input and the memory of the block is used to decide the output. Sigmoid function decides which values to let through 0,1. and tanh function gives weightage to the values which are passed deciding their level of importance ranging from-1 to 1 and multiplied with output of Sigmoid.

$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

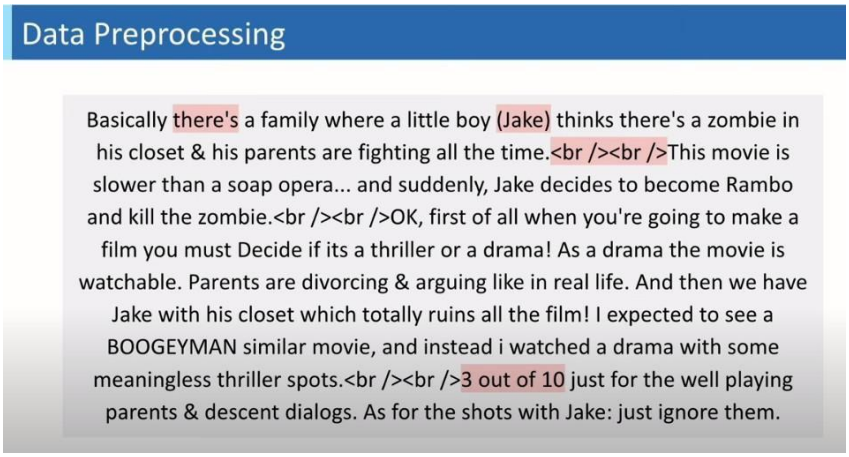
$$h_t = o_t * \tanh\left(C_t\right)$$

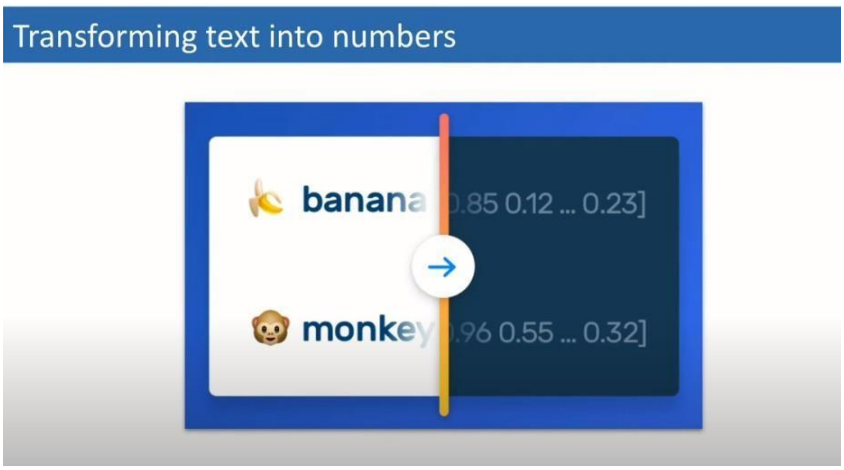
Algorithm -

1) Load IMDb Movie Reviews dataset (50,000 reviews)

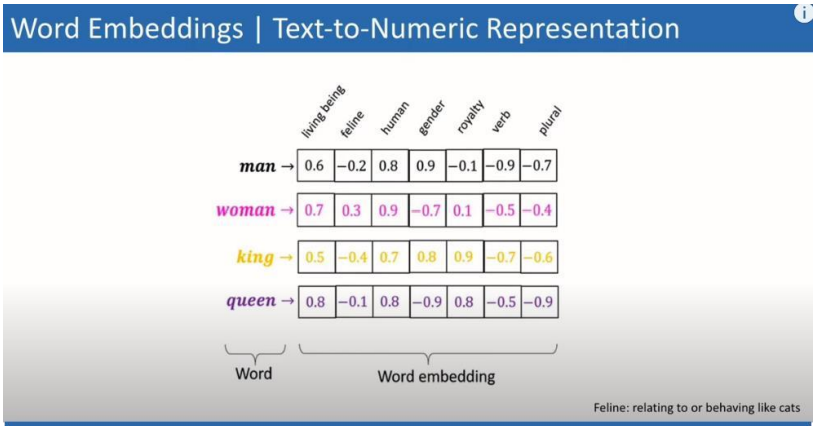


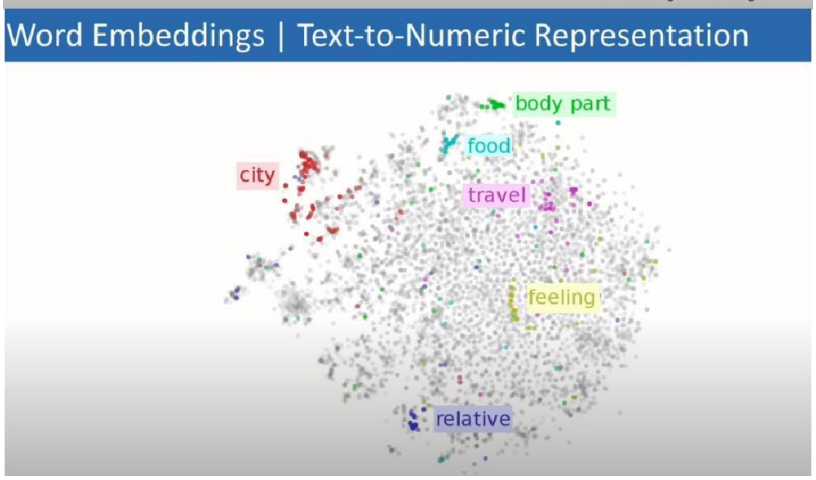
2) Pre-process dataset by removing special characters, numbers, etc. from user reviews + convert sentiment labels positive & negative to numbers 1 & 0, respectively





3) Import GloVe Word Embedding to build Embedding Dictionary + Use this to build Embedding Matrix for our Corpus





4) Model Training using Deep Learning in Keras for separate: Simple Neural Net, CNN and LSTM Models

Application –

1) Product Review Sentiment Analysis:
Predict sentiment (positive, negative, neutral) from user reviews for product improvement insights. 2) Customer Feedback Analysis:
Analyze sentiments in customer feedback to understand overall satisfaction and identify areas for improvement. 3) Brand Monitoring:
Monitor social media for product mentions and analyze sentiments to assess brand perception. 4) Market Research:
Analyze sentiments in market surveys to gauge consumer opinions about specific products or features. 5) Quality Assurance in E-commerce:
Automatically categorize and flag reviews with negative sentiments to improve product quality.

Inference –

The process involves data preparation, embedding, model design, training, and evaluation.
The use of different architectures such as Simple Neural Net, CNN, and LSTM allows for comparison and analysis of their performance on sentiment prediction for IMDb movie reviews. The GloVe Word Embedding enhances the models' understanding of the textual data.
Finally, predictions are made on real IMDb movie reviews to assess the models' applicability and accuracy.

References:

https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e https://colah.github.io/posts/2015-08-Understanding-LSTMs/ https://youtu.be/oWo9SNcyxlI?si=0OzO6SUYZ_FxbTgY

Conclusion: Thus Designed RNN or its variant including LSTM or GRU

Lab Assignment No.	04
Title	Design and implement a CNN for Image Classification a) Select a suitable image classification dataset (medical imaging, agricultural, etc.). b) Optimized with different hyper-parameters including learningrate, filter size, no. of layers, optimizers, dropouts, etc

Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV: Deep Learning
Assessment Marks	
Assessor's Sign	

ASSIGNMENT No: 04

Title:-

Design and implement Union, Intersection, Complement and Difference operations on fuzzy sets. Also create fuzzy relations by Cartesian product of any two fuzzy sets and perform max-min composition on any two fuzzy relations.

Problem Statement: Design and implement Union, Intersection, Complement, and Difference operations on fuzzy sets. Create fuzzy relations by the Cartesian product of any two fuzzy sets and perform max-min composition on any two fuzzy relations.

Objective:

1. To implement Union, Intersection, Complement, and Difference operations on fuzzy sets.
2. To create fuzzy relations using the Cartesian product of fuzzy sets and perform max-min compositionon fuzzy relations.
3. To demonstrate the application of these operations in fuzzy logic.

Outcome-

1. Successfully implement Union, Intersection, Complement, and Difference operations on fuzzy sets.
2. Create fuzzy relations using the Cartesian product of fuzzy sets and perform max-min composition onfuzzy relations.

Software Requirement

- Python (3.x recommended)
- Jupyter Notebook or any Python IDE

Hardware Requirement

A machine with sufficient RAM and processing power for model training (8GB RAM recommended)

Prerequisities-

What is Fuzzy Set ?

Fuzzy refers to something that is unclear or vague . Hence, Fuzzy Set is a Set where every key is associated with value, which is between 0 to 1 based on the certainty .This value is often called as degree of membership. Fuzzy Set is denoted with a Tilde Sign on top of the normal Set notation.

Operations on Fuzzy Set with Code :

1. Union :

Operations on Fuzzy Set with Code :

1. Union : Consider 2 Fuzzy Sets denoted by A and B, then let's consider Y be the Union of them, then for every member of A and B, Y will be:

degree_of_membership(Y)= max(degree_of_membership(A), degree_of_membership(B))

EXAMPLE :

```
# Example to Demonstrate the

# Union of Two Fuzzy Sets

A = dict()

B = dict()

Y = dict()

A = {"a": 0.2, "b": 0.3, "c": 0.6, "d": 0.6}

B = {"a": 0.9, "b": 0.9, "c": 0.4, "d": 0.5}

print('The First Fuzzy Set is :', A)

print('The Second Fuzzy Set is :', B)

for A_key, B_key in zip(A, B):

        A_value = A[A_key]

        B_value = B[B_key]


        if A_value > B_value:

                Y[A_key] = A_value

        else:

                Y[B_key] = B_value

print('Fuzzy Set Union is :', Y)
```

Output

The First Fuzzy Set is : {'a': 0.2, 'b': 0.3, 'c': 0.6, 'd': 0.6}

The Second Fuzzy Set is : {'a': 0.9, 'b': 0.9, 'c': 0.4, 'd': 0.5}

Fuzzy Set Union is : {'a': 0.9, 'b': 0.9, 'c': 0.6, 'd': 0.6}

2. Intersection :

Consider 2 Fuzzy Sets denoted by A and B, then let's consider Y be the Intersection of them, then for every member of A and B, Y will be:

degree_of_membership(Y)= min(degree_of_membership(A), degree_of_membership(B))

EXAMPLE :

```
# Example to Demonstrate

# Intersection of Two Fuzzy Sets

A = dict()

B = dict()

Y = dict()

A = {"a": 0.2, "b": 0.3, "c": 0.6, "d": 0.6}

B = {"a": 0.9, "b": 0.9, "c": 0.4, "d": 0.5}

print('The First Fuzzy Set is :', A)

print('The Second Fuzzy Set is :', B)

for A_key, B_key in zip(A, B):

        A_value = A[A_key]

        B_value = B[B_key]

        if A_value < B_value:

                Y[A_key] = A_value

        else:

                Y[B_key] = B_value
```

```
print('Fuzzy Set Intersection is :', Y)
```

Output

The First Fuzzy Set is : {'a': 0.2, 'b': 0.3, 'c': 0.6, 'd': 0.6}

The Second Fuzzy Set is : {'a': 0.9, 'b': 0.9, 'c': 0.4, 'd': 0.5}

Fuzzy Set Intersection is : {'a': 0.2, 'b': 0.3, 'c': 0.4, 'd': 0.5}

3. Complement :

Consider a Fuzzy Sets denoted by A , then let's consider Y be the Complement of it, then for every member ofA , Y will be:

degree_of_membership(Y)= 1 - degree_of_membership(A)

EXAMPLE :

```
# Example to Demonstrate the

# Difference Between Two Fuzzy

SetsA = dict()

Y = dict()

A = {"a": 0.2, "b": 0.3, "c": 0.6, "d": 0.6}

print('The Fuzzy Set is :',

A)for A_key in A:

Y[A_key]= 1-A[A_key]

print('Fuzzy Set Complement is :', Y)
```

Output

The Fuzzy Set is : {'a': 0.2, 'b': 0.3, 'c': 0.6, 'd': 0.6}

Fuzzy Set Complement is : {'a': 0.8, 'b': 0.7, 'c': 0.4, 'd': 0.4}

4.Difference :

Consider 2 Fuzzy Sets denoted by A and B, then let's consider Y be the Intersection of them, then for

everymember of A and B, Y will be:

degree_of_membership(Y)= min(degree_of_membership(A), 1- degree_of_membership(B))

EXAMPLE :

```
# Example to Demonstrate the

# Difference Between Two Fuzzy

SetsA = dict()

B =

dict()Y =

dict()

A = {"a": 0.2, "b": 0.3, "c": 0.6, "d": 0.6}

B = {"a": 0.9, "b": 0.9, "c": 0.4, "d": 0.5}

print('The First Fuzzy Set is :', A)

print('The Second Fuzzy Set is :',

B)for A_key, B_key in zip(A, B):

        A_value = A[A_key]

        B_value = B[B_key]

        B_value = 1 - B_value

        if A_value < B_value:

                Y[A_key] = A_value

        else:

                Y[B_key] = B_value

print('Fuzzy Set Difference is :', Y)
```

Conclusion: Fuzzy logic provides a powerful framework for dealing with uncertainty and imprecision in data. By implementing operations on fuzzy sets and relations, we can model complex relationships and make more flexible decisions in various applications such as control systems, artificial intelligence, and pattern recognition. The Union, Intersection, Complement, and Difference operations allow us to manipulate fuzzy sets, while fuzzy relations and max-min composition enable us to represent and combine fuzzy relationships between sets. These operations and techniques expand the capabilities of traditional logic and set theory,

offering a more nuanced approach to handling vague and ambiguous infor

Lab Assignment No.	06
Title	Import Data from different Sources such as (Excel, Sql Server, Oracle etc.) and load in targeted system.
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV: Business Intelligence
Assessment Marks	
Assessor's Sign	

ASSIGNMENT No: 06

Problem Statement: Import Data from different Sources such as (Excel, Sql Server, Oracle etc.) and load in targeted system.

Title: Import Data from different Sources such as (Excel, Sql Server, Oracle etc.) and load in targeted system.

Problem Statement: Import Data from different Sources such as (Excel, Sql Server, Oracle etc.) and load in targeted system.

Prerequisite:

Basics of Python

Software Requirements: Power BI Tool

Hardware Requirements: PIV, 2GB

RAM, 500 GB HDD

Learning Objectives:

Learn to import data from different sources such as(Excel, Sql Server, Oracle etc.) and load in targeted system.

Outcomes:

After completion of this assignment students are able to understand how to import data from different sources such as(Excel, Sql Server, Oracle etc.) and load in targeted system.

Theory:

What is Legacy Data?

Legacy data, according to Business Dictionary, is "information maintained in an old or outof-date format or computer system that is consequently challenging to access or handle."

Sources of Legacy Data

Where does legacy data come from? Virtually everywhere. Figure 1 indicates that there are many sources from which you may obtain legacy data. This includes existing databases, often relational, although non-RDBs such as hierarchical, network, object, XML, object/relational databases, and NoSQL databases. Files, such as XML documents or "flat files"ù such as configuration files and comma-delimited text files, are also common sources of legacy data. Software, including legacy applications that have been wrapped (perhaps via CORBA) and

access to existing information. The point to be made is that there is often far more to gaining access to legacy data than simply writing an SQL query against an existing relational database.

legacy services such as web services or CICS transactions, can also provide access to existing information.

The point to be made is that there is often far more to gaining

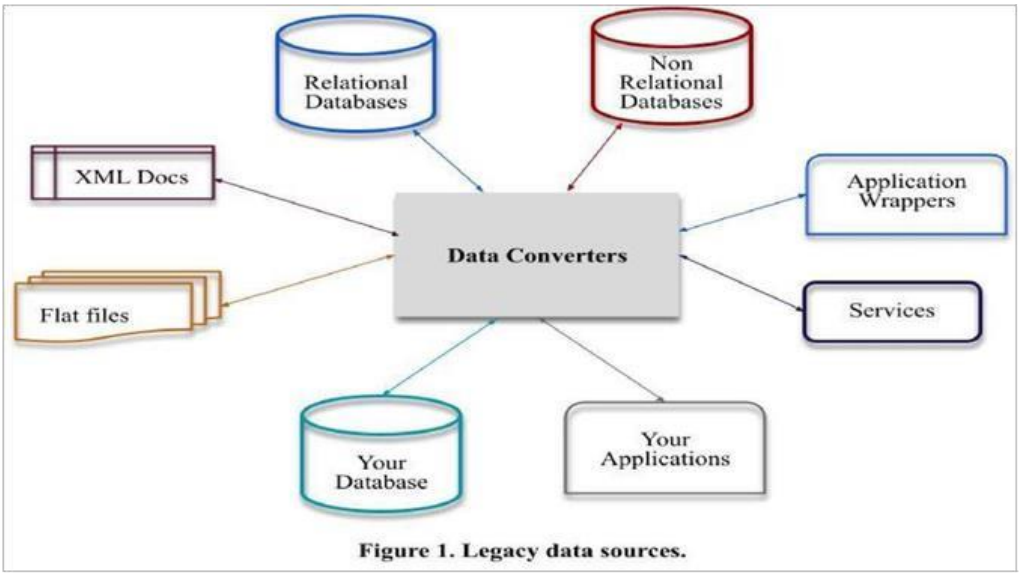
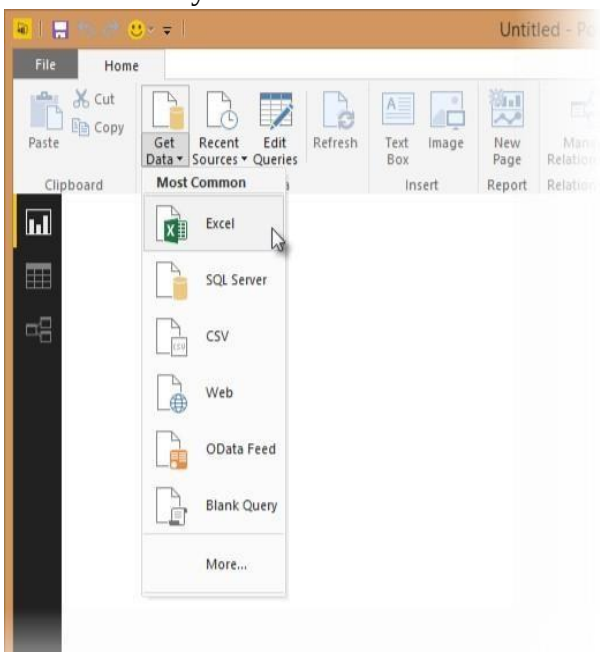


Figure 1. Legacy data sources.

Importing Excel Data

1) Launch Power BI Desktop.
2) From the Home ribbon, select Get Data. Excel is one of the Most Common data connections, so you can select it directly from the Get Data menu



1) If you select the Get Data button directly, you can also select FIle > Excel and select Connect.

2) In the Open File dialog box, select the Products.xlsx file.

3) In the Navigator pane, select the Products table and then select Edit.

Importing Data from OData Feed

In this task, you'll bring in order data. This step represents connecting to a sales system. You import data into Power BI Desktop from the sample Northwind OData feed at the following URL, which you can copy (and then paste) in the steps below: http://services.odata.org/V3/Northwind/Northwind.svc/ Connect to an

OData feed:

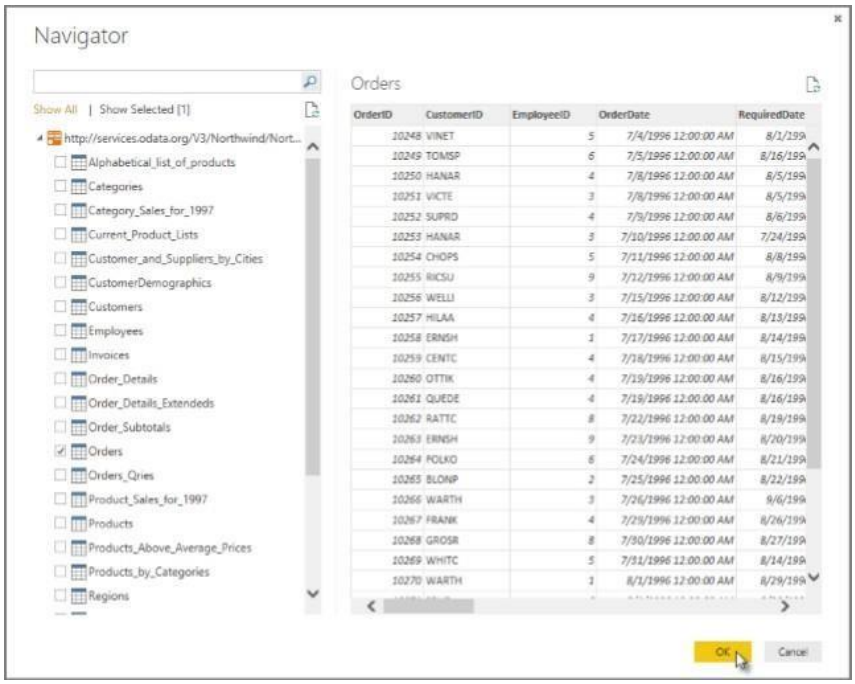
rom the Home ribbon tab in Query Editor, select Get Data.

Browse to the OData Feed data source.

In the OData Feed dialog box, paste the URL for the Northwind OData feed.

Select OK.

In the Navigator pane, select the Orders table, and then select Edit.



Conclusion: - This way, Implemented a program for inverted files.

Lab Assignment No.	07
Title	Data Visualization from Extraction Transformation and Loading (ETL) Process.
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV-Business Intelligence
Assessment Marks	
Assessor's Sign	

ASSIGNMENT No: 07

Title: Data Visualization from Extraction Transformation and Loading (ETL) Process.

Problem Statement: Data Visualization from Extraction Transformation and Loading (ETL) Process.

Prerequisite:

Basics of Python

Software Requirements: Jupyter **Hardware Requirements:**

PIV, 2GB RAM, 500 GB HDD

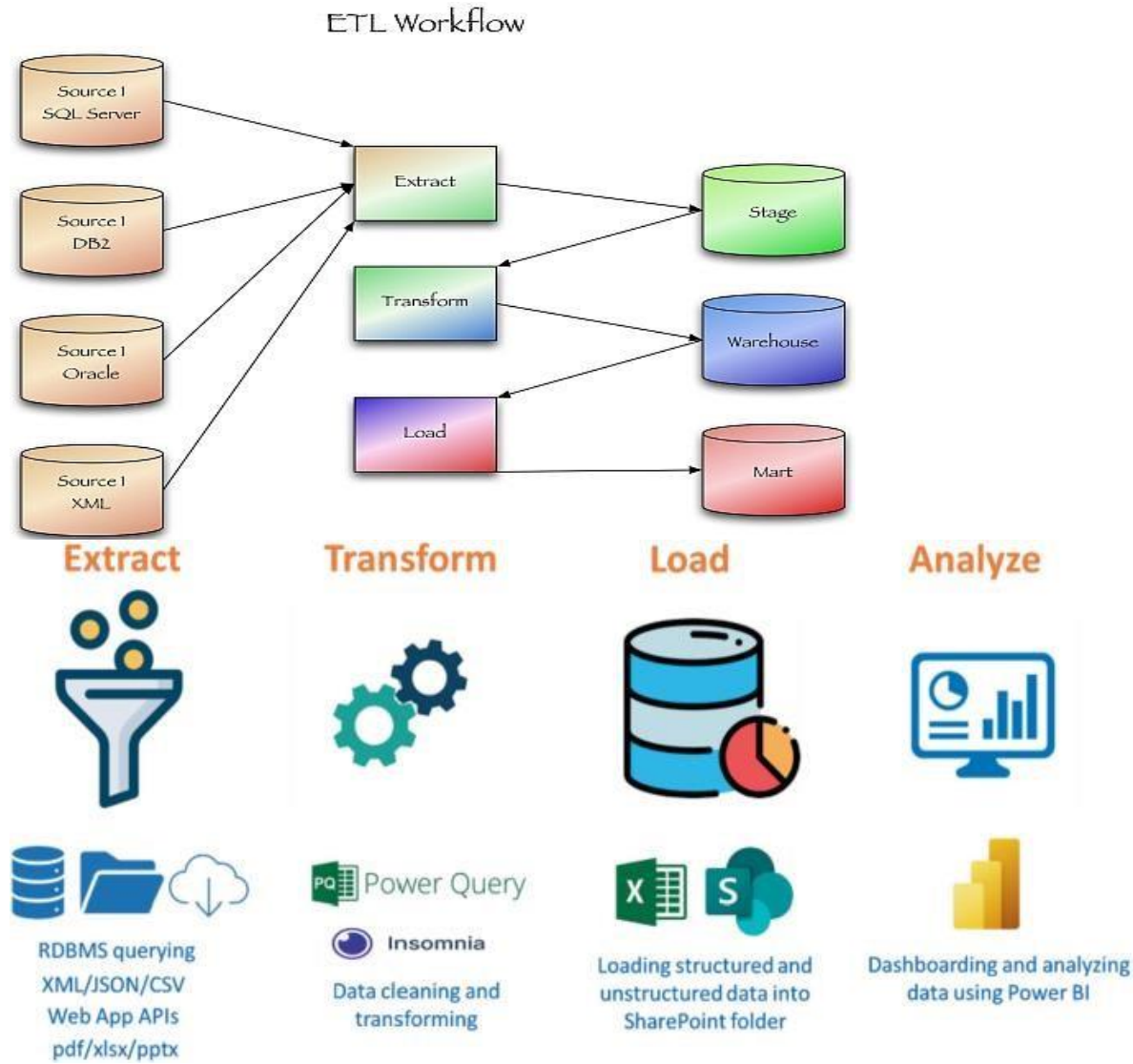
Outcomes:

After completion of this assignment students are able to understand how Data Visualization is done through Extraction Transformation and Loading (ETL) Process **Theory:**

Extract, transform, and load (ETL) are 3 data processes, followed after data collection. Extraction takes data, collected in data sources like flat files, databases (relational, hierarchical etc.), transactional datastores, semi-structured repositories (e.g. email systems or document libraries) with different structure and format, pre-validating extracted data and parsing valid data to destination (e.g. staging database)

Transformation takes extracted data and applies predefined rules and functions to it, including selection (e.g. ignore or remove NULLs), data cleansing, encoding (e.g. mapping “Male” to “M”), deriving (e.g. calculating designated value as a product of extracted value and predefined constant) , sorting, joining data from multiple sources (e.g. lookup or merge), aggregation (e.g. summary for each month), transposing (columns to rows or vice versa), splitting, disaggregation, lookups (e.g. validation through dictionaries), predefined validation etc. which may lead to rejection of some data. Transformed data can be stored into Data Warehouse (DW).

Load takes transformed data and places it into end target, in most cases called Data Mart (sometimes they called Data Warehouse too). Load can append, refresh or/and overwrite preexisting data, apply constraints and execute appropriate triggers (to enforce data integrity, uniqueness, mandatory fields, provide log etc.) and may start additional processes, like data backup or replication.



Conclusion:-

This way Data Visualization from Extraction Transformation and Loading (ETL) Process is done.

Lab Assignment No.	8
Title	Perform the Extraction Transformation and Loading (ETL) process to construct the database in the Sql server / Power BI.
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV-Business Intelligence
Assessment Marks	
Assessor's Sign	

ASSIGNMENT No: 08

Title: Perform the Extraction Transformation and Loading (ETL) process to construct the database in the Sql server / Power BI.

Problem Statement: Perform the Extraction Transformation and Loading (ETL) process to construct the database in the Sql server / Power BI.

Prerequisite:

Basics of Python

Software Requirements: Jupyter **Hardware Requirements:**

PIV, 2GB RAM, 500 GB HDD

Outcomes:

After completion of this assignment students are able to understand how to Perform the Extraction Transformation and Loading (ETL) process to construct the database in the Sql server / Power BI.

Theory:

Step 1 : Data Extraction :

The data extraction is first step of ETL. There are 2 Types of Data Extraction

Full Extraction : All the data from source systems or operational systems gets extracted to staging area. (Initial Load)

Partial Extraction : Sometimes we get notification from the source system to update specific date. It is called as Delta load.

Source System Performance: The Extraction strategies should not affect source system performance.

Step 2 : Data Transformation :

The data transformation is second step.After extracting the data there is big need to do the transformation as per the target system.I would like to give you some bullet points of Data Transformation.

- Data Extracted from source system is in to Raw format. We need to transform it before loading in to target server.
- Data has to be cleaned, mapped and transformed.
- There are following important steps of Data Transformation :

2. **Selection :** Select data to load in target

3. **Matching :** Match the data with target system

Data Transforming : We need to change data as per target table structures

- Character set conversion : Need to transform the character sets as per the target systems. (Firstname and last name example)
- Calculated and derived values: In source system there is first val and second val and in target we need the calculation of first val and second val.
- Data Conversion in different formats : If in source system date in in DDMMYY format and in target the date is in DDMONYYYY format then this transformation needs to be done at transformation phase.

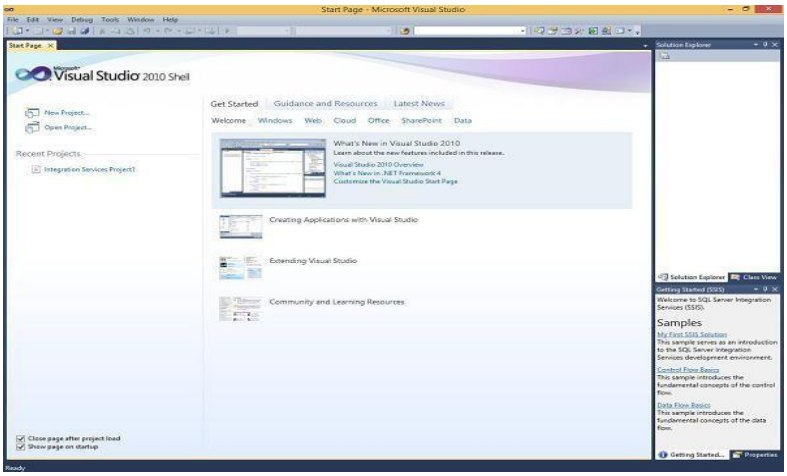
Step 3 : Data Loading

- Data loading phase loads the prepared data from staging tables to main tables.

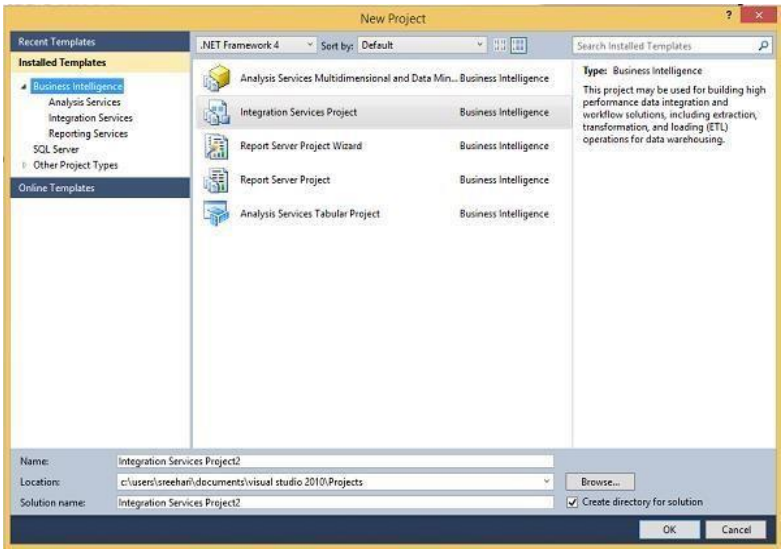
ETL process in SQL Server:

Following are the steps to open BIDS\SSDT.

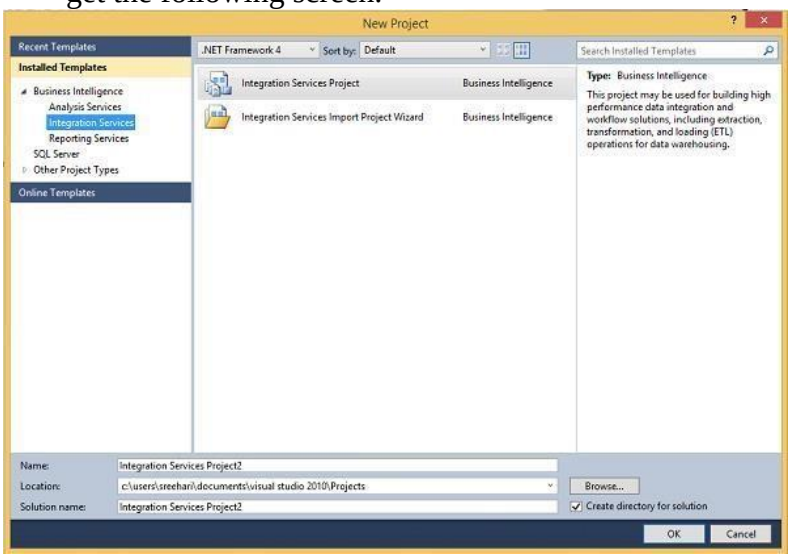
Step 1 − Open either BIDS\SSDT based on the version from the Microsoft SQL Server programs group. The following screen appears



Step 2 − The above screen shows SSDT has opened. Go to file at the top left corner in the above image and click New. Select project and the following screen opens.



Step 3 − Select Integration Services under Business Intelligence on the top left corner in the above screen to get the following screen.



Step 4 − In the above screen, select either Integration Services Project or Integration Services Import Project Wizard based on your requirement to develop\create the package.

There are two modes − Native Mode (SQL Server Mode) and Share Point Mode.
There are two models − Tabular Model (For Team and Personal Analysis) and Multi Dimensions Model (For Corporate Analysis).

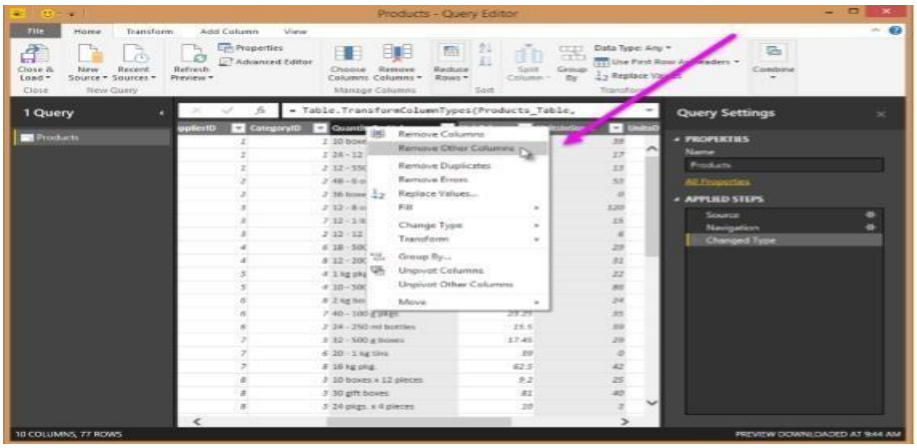
The BIDS (Business Intelligence Studio till 2008 R2) and SSDT (SQL Server Data Tools from 2012) are environments to work with SSAS.

ETL Process in Power BI

1) Remove other columns to only display columns of interest

Power BI Desktop includes Query Editor, which is where you shape and transform your data connections. Query Editor opens automatically when you select **Edit** from Navigator. You can also open the Query Editor by selecting Edit Queries from the Home ribbon in Power BI Desktop. The following steps are performed in Query Editor.

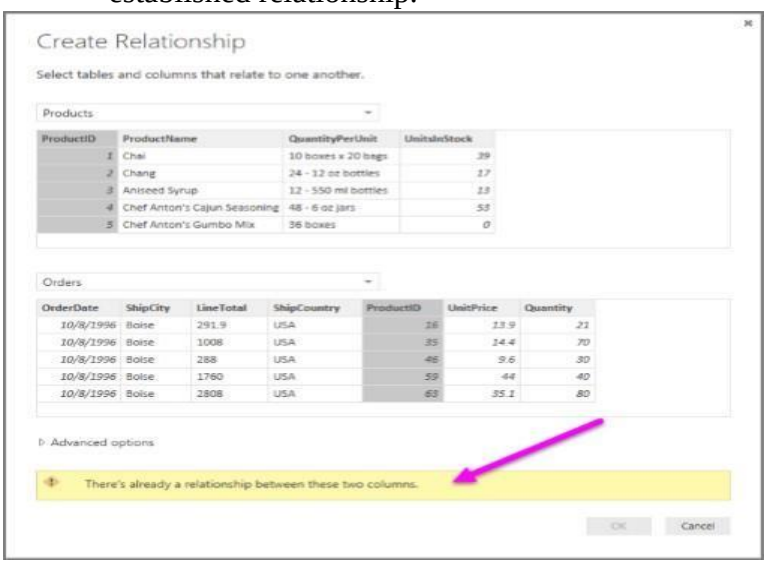
1. In **Query Editor**, select the **ProductID, ProductName**, **QuantityPerUnit,** and **UnitsInStock** columns (use **Ctrl+Click** to select more than one column, or **Shift+Click** to select columns that are beside each other).
2. Select **Remove Columns > Remove** Other Columns from the ribbon, or right-click on a column header and click Remove Other Columns.



3. **Change the data type of the UnitsInStock column**

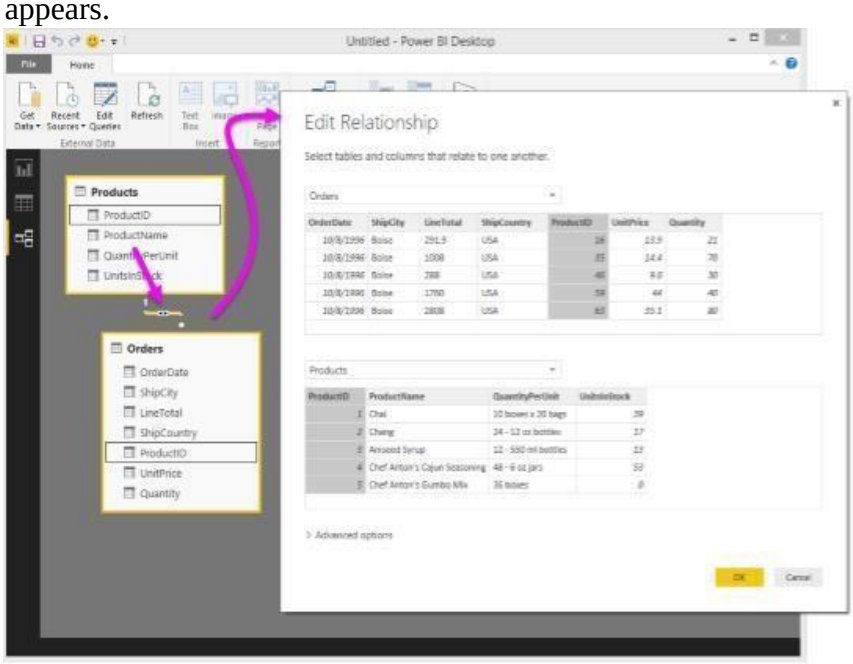
When Query Editor connects to data, it reviews each field and to determine the best data type. For the Excel workbook, products in stock will always be a whole number, so in this step you confirm the **UnitsInStock** column's datatype is Whole Number.

1. Select the **UnitsInStock** column.
2. Select the **Data Type drop-down button** in the **Home ribbon**.
3. If not already a Whole Number, select **Whole Number** for data type from the drop down (the Data Type: button also displays the data type for the current selection).
4. When we attempt to create the relationship, we see that one already exists! As shown in the Create Relationship dialog (by the shaded columns), the ProductsID fields in each query already have an established relationship.



5. Select Cancel, and then select Relationship view in Power BI Desktop.

6. When you double-click the arrow on the line that connects the to queries, an Edit Relationship dialog appears.



Conclusion: Thus Performed Extraction Transformation and Loading (ETL) process to construct the database in the Sql server / Power BI.

Assignment :11

AIM: To Develop a MapReduce program to calculate the frequency of a given word in agiven file**Map Function** – It takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (Key-Value pair).

Example – (Map function in Word Count)

Input

Set of data

Bus, Car, bus, car, train, car, bus, car, train, bus, TRAIN,BUS, buS, caR, CAR, car, BUS, TRAIN

Output

Convert into another set of data

(Key,Value)

(Bus,1), (Car,1), (bus,1), (car,1), (train,1), (car,1), (bus,1), (car,1), (train,1), (bus,1),

(TRAIN,1),(BUS,1), (buS,1), (caR,1), (CAR,1), (car,1), (BUS,1), (TRAIN,1)

Reduce Function – Takes the output from Map as an input and combines those data tuplesinto a smaller set of tuples.

Example – (Reduce function in Word Count)

Input Set of Tuples (output of Map function)

(Bus,1), (Car,1), (bus,1), (car,1), (train,1), (car,1), (bus,1), (car,1), (train,1), (bus,1),

(TRAIN,1),(BUS,1),

(buS,1),(caR,1),(CAR,1), (car,1), (BUS,1), (TRAIN,1)

Output **Converts into smaller set of tuples**

(BUS,7), (CAR,7), (TRAIN,4)

Work Flow of Program

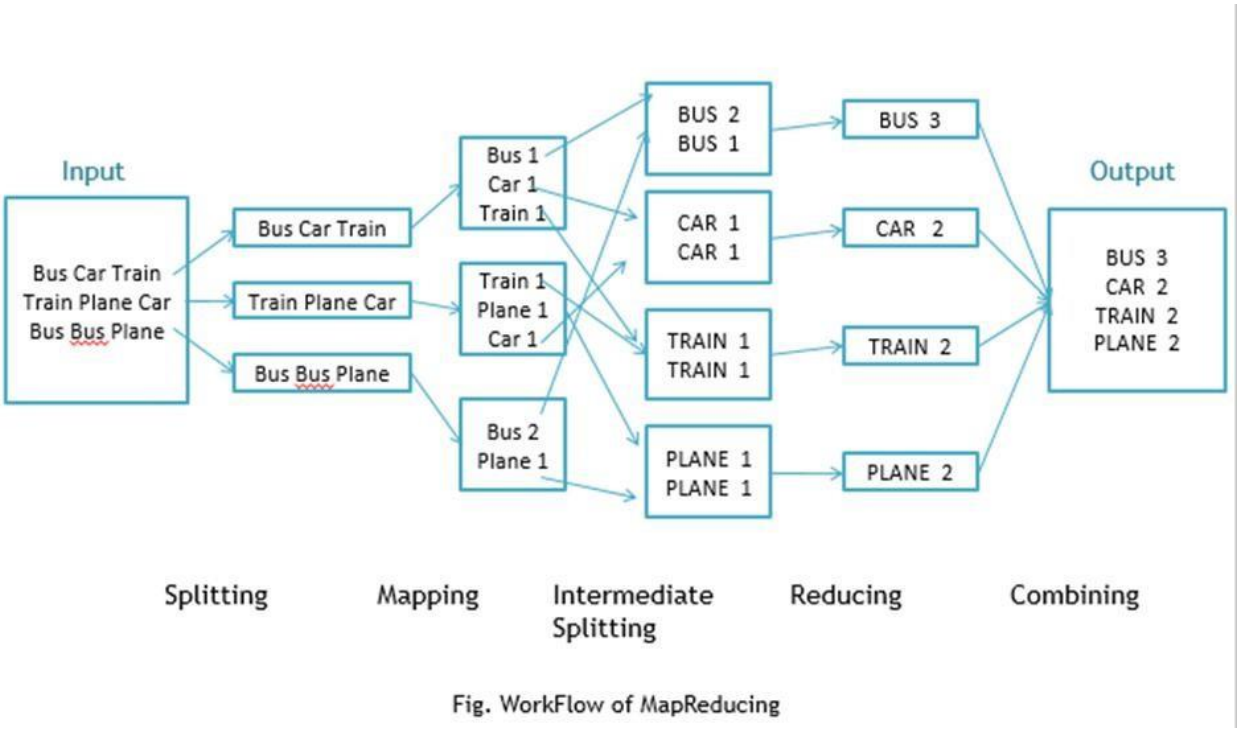


Fig. WorkFlow of MapReducing

Workflow of MapReduce consists of 5 steps

1. **Splitting –** The splitting parameter can be anything, e.g. splitting by space,comma, semicolon, or even by a new line ('\n').
2. **Mapping –** as explained above
3. Intermediate splitting – the entire process in parallel on different clusters. In orderto group them in "Reduce Phase" the similar KEY data should be on same cluster.
4. **Reduce –** it is nothing but mostly group by phase
5. **Combining –** The last phase where all the data (individual result set from eachcluster) is combine together to form a Result

Now Let's See the Word Count Program in Java

Make sure that Hadoop is installed on your system with java idk

Steps to follow

Step 1. Open Eclipse> File > New > Java Project > (Name it – MRProgramsDemo) > Finish

Step 2. Right Click > New > Package (Name it - PackageDemo) > Finish

Step 3. Right Click on Package > New > Class (Name it - WordCount)
Step 4. Add Following Reference Libraries –

Right Click on Project > Build Path> Add External Archivals

- /usr/lib/hadoop-0.20/hadoop-core.jar
- Usr/lib/hadoop-0.20/lib/Commons-cli-1.2.jar

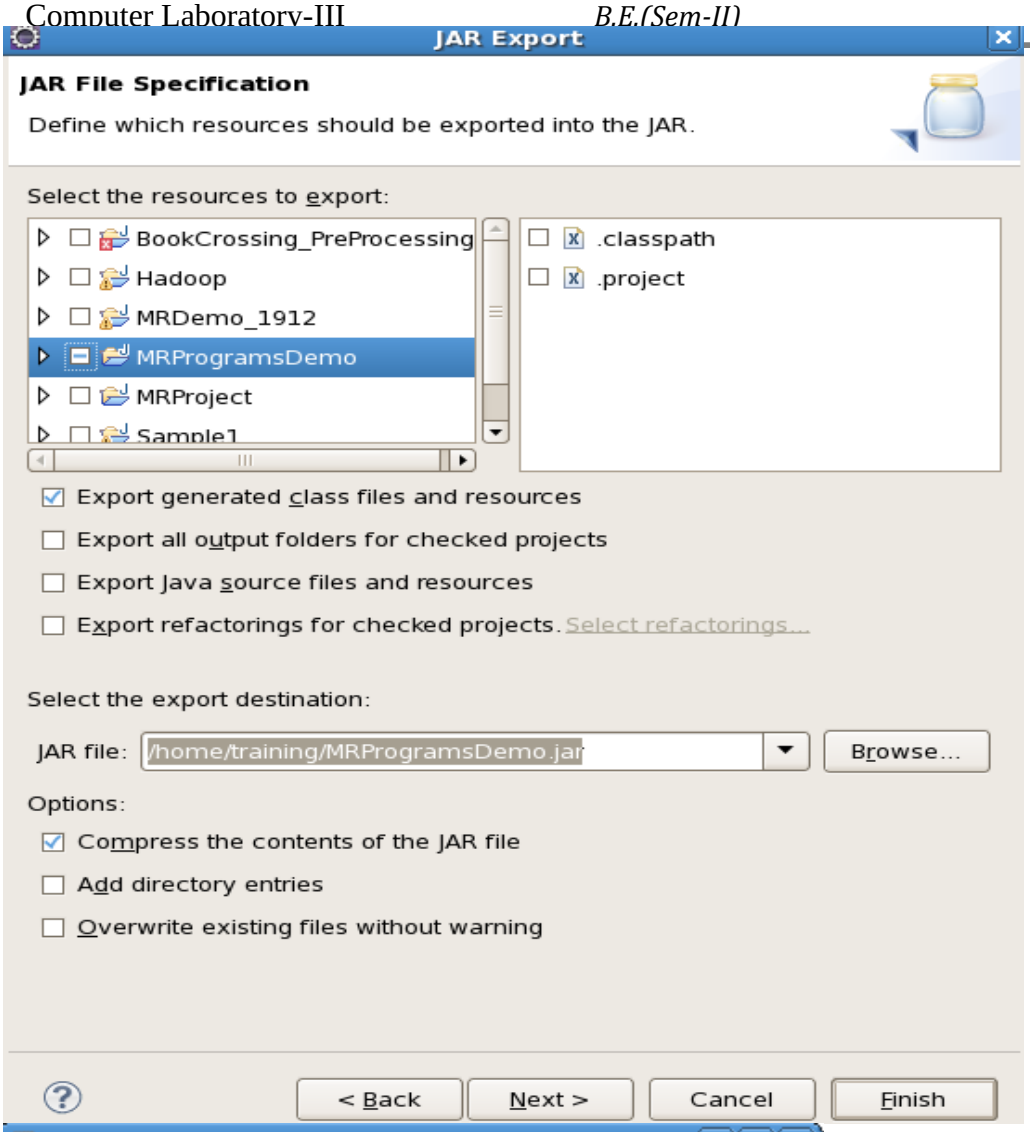
Program: Step 5. Type following Program :

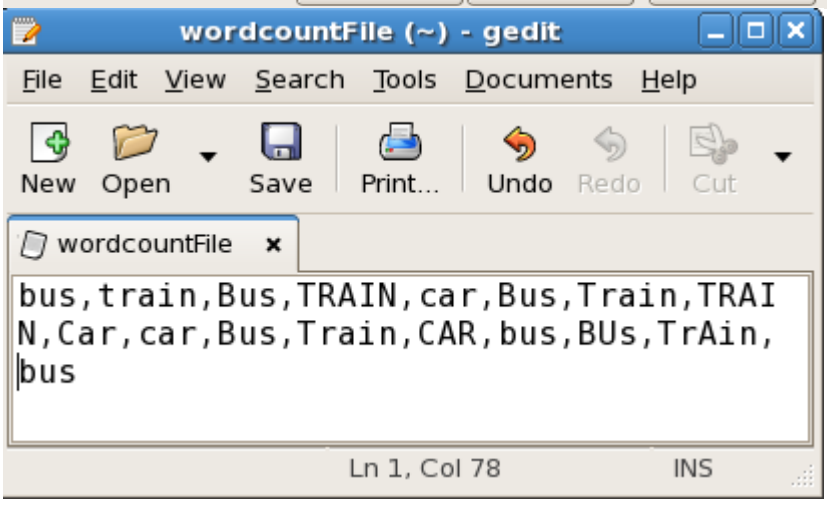
```
package
PackageDemo; import
java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import
org.apache.hadoop.mapreduce.Mapper;
import
org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
public class WordCount {
public static void main(String [] args) throws Exception
{
Configuration c=new Configuration();
String[] files=new GenericOptionsParser(c,args).getRemainingArgs();
Path input=new Path(files[0]);
Path output=new
Path(files[1]); Job j=new
Job(c,"wordcount");
j.setJarByClass(WordCount.cl
ass);
j.setMapperClass(MapForWordCount.class);
j.setReducerClass(ReduceForWordCount.class);
j.setOutputKeyClass(Text.class);
j.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(j, input);
FileOutputFormat.setOutputPath(j, output);
System.exit(j.waitForCompletion(true)?0:1);
}
```

```
public static class MapForWordCount extends Mapper<LongWritable, Text, Text,
IntWritable>{
public void map(LongWritable  key, Text  value, Context  con) throws IOException,
InterruptedException
{
String line = value.toString();

String[]
words=line.split(",");
for(String word: words )
{
   Text outputKey = new
   Text(word.toUpperCase().trim());IntWritable
   outputValue = new IntWritable(1);
   con.write(outputKey, outputValue);
}
}
}
public static class ReduceForWordCount extends Reducer<Text, IntWritable, Text,
IntWritable>
{
public void reduce(Text word, Iterable<IntWritable> values, Context con) throws
IOException,
InterruptedException
{
int sum = 0;
for(IntWritable value : values)
{
sum += value.get();
}
con.write(word, new IntWritable(sum));
}
}
}
```

Make Jar File

Right Click on Project> Export> Select export destination as Jar File > next> Finish





To Move this into Hadoop directly, open the terminal and enter the following commands:

[training@localhost ~]$ hadoop fs -put wordcountFile wordCountFile

Run Jar file

(Hadoop jar jarfilename.jar packageName.ClassName PathToInputTextFile PathToOutputDirectry)

[training@localhost ~]$ Hadoop jar MRProgramsDemo.jarPackageDemo.WordCount wordCountFile MRDir1

Result: Open Result

```
[training@localhost ~]$ hadoop fs -ls MRDir1
Found 3 items
-rw-r--r-- 1 training supergroup
0 2016-02-23 03:36
/user/training/MRDir1/_SUCCESSdrwxr-xr-x -
training supergroup
0 2016-02-23 03:36 /user/training/MRDir1/_logs
-rw-r--r-- 1 training supergroup
20 2016-02-23 03:36 /user/training/MRDir1/part-r-00000
[training@localhost ~]$ hadoop fs -cat MRDir1/part-r-
00000BUS 7
CAR     4
TRAIN 6
```

Assignment 12

Develop a MapReduce program to find the grades of student's**.**

```
import
java.util.Scanner;
public class
JavaExample
{
    public static void main(String args[])
    {
            /* This program assumes that the student has 6 subjects,
             * thats why I have created the array of size 6. You can
             * change this as per the requirement.
             */
        int marks[] = new
        int[6];int i;
        float total=0, avg;
        Scanner scanner = new
        Scanner(System.in);for(i=0; i<6; i++) {
           System.out.print("Enter Marks of Subject"+(i+1)+":");marks[i]
           = scanner.nextInt();
           total = total + marks[i];
        }
        scanner.close();
        //Calculating
        averagehere avg
        = total/6;
        System.out.print("The student Grade is: ");
        if(avg>=80)
        {
            System.out.print("A");
        }
        else if(avg>=60 && avg<80)
        {
```

```
            System.out.print("B");
        }
        else if(avg>=40 && avg<60)
        {

            System.out.print("C");
        }
             else
        {

            System.out.print("D");

        }
    }
}
```

Expected Output:

Enter Marks of Subject1:40Enter Marks of Subject2:80 Enter Marks of Subject3:80
Enter Marks of Subject4:40
Enter Marks of Subject5:60Enter Marks of Subject6:60 The student Grade is: B

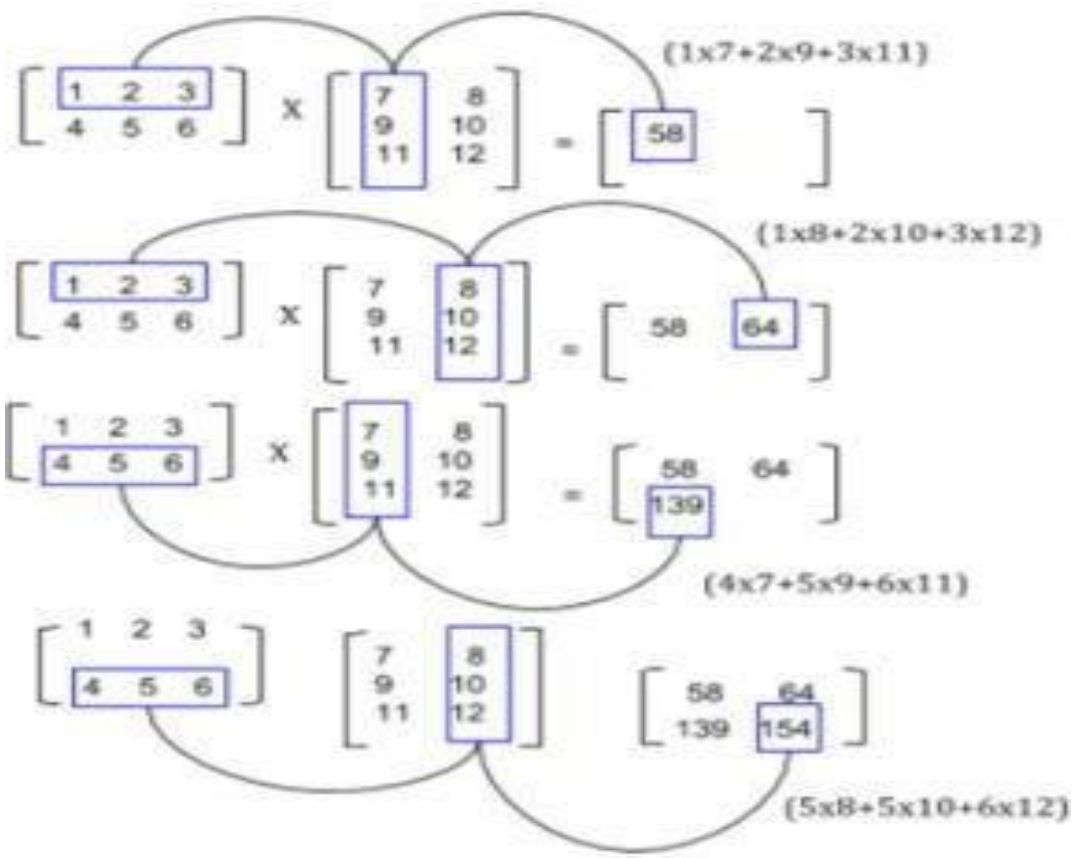
Actual Output

Result:

Assignment 13:

Develop a Map Reduce program to implement Matrix Multiplication.

In **mathematics**, **matrix multiplication** or the **matrix product** is a binary operationthat produces a matrix from two matrices. The definition is motivated by linear equations and linear transformations on vectors, which have numerous applications in applied mathematics, physics, and engineering. In more detail, if **A** is an $n \times m$ matrix and **B** is an $m \times p$ matrix, their matrix product **AB** is an $n \times p$ matrix, in which the m entries across a row of **A** are multiplied with the m entries down a column of **B** and summed to produce an entry of **AB**. When two linear transformations are represented by matrices, then the matrix product represents the composition of the two transformations.



Algorithm for Map Function.

a. for each element m_{ij} of M do
produce (key,value) pairs as ((i,k), (M,j,m_{ij}), for k=1,2,3,.. upto the number of columns of N

b. for each element njk of N do
produce (key,value) pairs as ((i,k),(N,j,N_{jk}), for i = 1,2,3,.. Upto the number ofrows of M.

c. return Set of (key,value) pairs that each key (i,k), has list with values (M,j,m_{ij}) and (N, j,n_{jk}) for all possible values of j.

Algorithm for Reduce Function.

d. for each key (i,k) do
e. sort values begin with M by j in listM sort values begin with N by j in listN multiply mij and njk for jth value of each list
f. sum up mij x njk return (i,k), Σj=1 mij x njk

Step 1. Download the hadoop jar files with these links.

Download Hadoop Common Jar files: https://goo.gl/G4MyHp
$ wget https://goo.gl/G4MyHp -O hadoop-common-2.2.0.jar Download
Hadoop Mapreduce Jar File: https://goo.gl/KT8yfB
$ wget https://goo.gl/KT8yfB -O hadoop-mapreduce-client-core-2.7.1.jar

Step 2. Creating Mapper file for Matrix Multiplication.

```
import
java.io.DataInput;
import
java.io.DataOutput;
import
java.io.IOException
;import
java.util.ArrayList;

import
org.apache.hadoop.conf.Configuration
;import org.apache.hadoop.fs.Path;
import
org.apache.hadoop.io.DoubleWritable
;import
org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.Writable;
import
org.apache.hadoop.io.WritableComparable
;import org.apache.hadoop.mapreduce.Job;
import
org.apache.hadoop.mapreduce.Mapper;
import
org.apache.hadoop.mapreduce.Reducer;
import
org.apache.hadoop.mapreduce.lib.input.*;
import
org.apache.hadoop.mapreduce.lib.output.*
;import
```

```
org.apache.hadoop.util.ReflectionUtils;

class Element implements
        Writable {int tag;
        int
        index
        ;
        doubl
        e
        value
        ;
        Elem
        ent()
        {
                tag = 0;
                index = 0;
                value = 0.0;
        }
        Element(int tag, int index, double
                value) {this.tag = tag;
                this.index
                = index;
                this.value
                = value;
        }
        @Override

        public void readFields(DataInput input) throws
                IOException {tag = input.readInt();
                index =
                input.readInt();
                value =
                input.readDouble();
        }
        @Override
        public void write(DataOutput output) throws
                IOException {output.writeInt(tag);
                output.writeInt(index)
                ;
                output.writeDouble(va
```

```
            lue);
        }
    }
    class Pair implements
        WritableComparable<Pair> {int i;
        int j;

        Pair() {
            i = 0;


            j = 0;
        }
        Pair(int i, int j) {
            t
            h
            i
            s
            .i
            =
            i;
            t
            h
            i
            s
            .j
            =
            j;
        }
        @Override
        public void readFields(DataInput input) throws
            IOException {i = input.readInt();
            j = input.readInt();
        }
        @Override
        public void write(DataOutput output) throws
            IOException {output.writeInt(i);
            output.writeInt(j);
        }
```

```
        @Override
        public int compareTo(Pair
                compare) {if (i >
                compare.i) {
                        return 1;

                } else if ( i <
                        compare.i) {
                        return -1;
                } else {
                        if(j > compare.j) {
                                return 1;
                        } else if (j <
                                compare.j
                                ) {return -
                                1;
                        }
                }
                return 0;
        }
        public String
                toString() {
                return i + " "
                + j + " ";
        }
}
public class Multiply {
        public static class MatriceMapperM extends Mapper<Object,Text,IntWritable,Element>
                @Override
                public void map(Object key, Text value, Context
                                context) throws IOException,
                                InterruptedException {
                        String readLine = value.toString();
                        String[] stringTokens =
                        readLine.split(",");

                        int index = Integer.parseInt(stringTokens[0]);
                        double elementValue =
                        Double.parseDouble(stringTokens[2]);Element e =
```

```
                    new Element(0, index, elementValue); IntWritable
                    keyValue = new
IntWritable(Integer.parseInt(stringTokens[1]));
                    context.write(keyValue, e);
                }
        }
        public static class MatriceMapperN extends
                Mapper<Object,Text,IntWritable,Element> {@Override
                public void map(Object key, Text value, Context
                                context) throws IOException,
                                InterruptedException {
                    String readLine = value.toString();
                    String[] stringTokens =
                    readLine.split(",");
                    int index = Integer.parseInt(stringTokens[1]);
                    double elementValue =
                    Double.parseDouble(stringTokens[2]);Element e =
                    new Element(1,index, elementValue);

                    IntWritable
keyValue = new
IntWritable(Integer.parseInt(stringTokens
[0]));
                    context.write(keyValue, e);
                }
        }
        public static class ReducerMxN extends
Reducer<IntWritable,Element, Pair,DoubleWritable> {
        @Override
        public void reduce(IntWritable key, Iterable<Element> values, Context
context) throwsIOException, InterruptedException {
                ArrayList<Element> M = new
                ArrayList<Element>();ArrayList<Element> N
                = new ArrayList<Element>();Configuration
                conf = context.getConfiguration();
                for(Element element : values) {
                    Element tempElement = ReflectionUtils.newInstance(Element.class,
conf);

                    ReflectionUtils.copy(conf, element, tempElement);
```

```
                    if (tempElement.tag
                            == 0) {
                            M.add(tempEl
                            ement);
                    } else if(tempElement.tag
                            == 1) {
                            N.add(tempEleme
                            nt);
                    }
            }
            for(int i=0;i<M.size();i++) {
                    for(int j=0;j<N.size();j++) {

                            Pair p = new Pair(M.get(i).index,N.get(j).index);
                            double multiplyOutput = M.get(i).value * N.get(j).value;

                            context.write(p, new DoubleWritable(multiplyOutput));
                            }
                    }
            }
    }
    public static class MapMxN extends Mapper<Object, Text, Pair,
            DoubleWritable> {@Override
            public void map(Object key, Text value, Context
                            context) throws IOException,
                            InterruptedException {

                    String readLine =
                    value.toString(); String[]
                    pairValue = readLine.split(" ");
                    Pair p = new
Pair(Integer.parseInt(pairValue[0]),Integer.parseInt(pairValue[1]));
                    DoubleWritable val = new
DoubleWritable(Double.parseDouble(pairValue[2]));
                    context.write(p, val);
            }
    }
    public static class ReduceMxN extends Reducer<Pair,
DoubleWritable, Pair,DoubleWritable> {
```

```
        @Override
        public void reduce(Pair key, Iterable<DoubleWritable> values, Context
context)throws IOException, InterruptedException {
            double sum = 0.0;
            for(DoubleWritable value :
            values) {

                sum += value.get();
            }
            context.write(key, new DoubleWritable(sum));
        }
    }
    public static void main(String[] args) throws
        Exception {Job job = Job.getInstance();
        job.setJobName("MapIntermediate");
        job.setJarByClass(Project1.class);
        MultipleInputs.addInputPath(job, new Path(args[0]),
TextInputFormat.class,MatriceMapperM.class);
        MultipleInputs.addInputPath(job, new Path(args[1]),
TextInputFormat.class,MatriceMapperN.class);
        job.setReducerClass(ReducerMxN.class);
        job.setMapOutputKeyClass(IntWritable.class);
        job.setMapOutputValueClass(Element.class);
        job.setOutputKeyClass(Pair.class);
        job.setOutputValueClass(DoubleWritable.class);
        job.setOutputFormatClass(TextOutputFormat.cla
        ss); FileOutputFormat.setOutputPath(job, new
        Path(args[2]));job.waitForCompletion(true);
        Job job2 = Job.getInstance();
        job2.setJobName("MapFinalOutput");

        job2.setJarByClass(Project1.class);

        job2.setMapperClass(MapMxN.class);
        job2.setReducerClass(ReduceMxN.class);

        job2.setMapOutputKeyClass(Pair.class);
        job2.setMapOutputValueClass(DoubleWritable.class);

        job2.setOutputKeyClass(Pair.class);
```

```
        job2.setOutputValueClass(DoubleWritable.class);

        job2.setInputFormatClass(TextInputFormat.class);
        job2.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.setInputPaths(job2, new
        Path(args[2]));
        FileOutputFormat.setOutputPath(job2, new
        Path(args[3]));
        job2.waitForCompletion(true);
    }
}
```

Step 5. Compiling the program in particular folder named as operation

```
#!/bin/bash

rm -rf multiply.jar classes

module load

hadoop/2.6.0

mkdir -p classes
javac -d classes -cp classes:`$HADOOP_HOME/bin/hadoop classpath`
Multiply.javajar cf multiply.jar -C classes .

echo "end"
```

Step 6. Running the program in particular folder named as operation

```
export
HADOOP_CONF_DIR=/home/$USER/cometcluste
rmodule load hadoop/2.6.0
myhadoop-
configure.sh
start-dfs.sh

start-yarn.sh
```

```
hdfs dfs -mkdir -p /user/$USER
hdfs dfs -put M-matrix-large.txt /user/$USER/M-matrix-
large.txthdfs dfs -put N-matrix-large.txt /user/$USER/N-
matrix-large.txt
hadoop jar multiply.jar edu.uta.cse6331.Multiply /user/$USER/M-matrix-large.txt
/user/$USER/N-matrix-large.txt /user/$USER/intermediate
/user/$USER/outputrm -rf output-distr
mkdir output-distr
hdfs dfs -get /user/$USER/output/part* output-distr

stop
-
yarn
.sh
stop
-
dfs.
sh
myhadoop-cleanup.sh
```

Expected Output:

module load

hadoop/2.6.0rm -

rf output

intermediate

hadoop --config $HOME jar multiply.jar edu.uta.cse6331.Multiply M-matrix-small.txt N-matrix-small.txt intermediate output

Actual Output:

Result:

Assignment 14

AIM: Develop a MapReduce program to analyze Titanic ship data and to find the average age of the people (both male and female) who died in the tragedy. How many persons are survived in each class.

The titanic data will be..

Column 1 :PassengerI d Column 2 : Survived (survived=0 &died=1)Column 3 :Pclass Column 4 : Name
Column 5 : Sex Column 6 : Age
Column 7 :SibSp Column 8 :Parch
Column 9 : Ticket Column 10 : Fare

Column 11 :Cabin Column 12 : Embarked

Description:

There have been huge disasters in the history of Map reduce, but the magnitude of the Titanic'sdisaster ranks as high as the depth it sank too. So much so that subsequent disasters have alwaysbeen described as "titanic in proportion" – implying huge losses.

Anyone who has read about the Titanic, know that a perfect combination of natural events and human errors led to the sinking of the Titanic on its fateful maiden journey from Southampton toNew York on April 14, 1912.

There have been several questions put forward to understand the cause/s of the tragedy – foremostamong them is: What made it sink and even more intriguing How can a 46,000 ton ship sink to the depth of 13,000 feet in a matter of 3 hours? This is a mind boggling

question indeed!

There have been as many inquisitions as there have been questions raised and equally that manytypes of analysis methods applied to arrive at conclusions. But this blog is not about analyzing why or what made the Titanic sink – it is about analyzing the data that is present about the Titanic publicly. It actually uses Hadoop MapReduce to analyze and arrive at:

- The average age of the people (both male and female) who died in the tragedyusing Hadoop MapReduce.
- How many persons survived – traveling class wise.

This blog is about analyzing the data of Titanic. This total analysis is performed in HadoopMapReduce.

This Titanic data is publically available and the Titanic data set is described below under the heading Data Set Description.

Using that dataset we will perform some Analysis and will draw out some insights like finding the average age of male and females died in Titanic, Number of males and females died in each compartment.

DATA SET DESCRIPTION

Column 1 : PassengerI
Column 2 : Survived (survived=0 & died=1) Column 3 :
PclassColumn 4 : Name
Column 5 : Sex
Column 6 :
Age Column 7
: SibSp
Column 8 :
Parch Column
9

: Ticket
Column 10 :
Fare Column
11 : Cabin
Column 12 :
Embarked

Mapper code:

```
public class Average_age {
public static class Map extends Mapper<LongWritable, Text, Text,

IntWritable> {private Text gender = new Text();

private IntWritable age = new IntWritable();

public void map(LongWritable key, Text value, Context context
) throwsIOException, InterruptedException {

   String line = value.toString(); String str[]=line.split(","); if(str.length>6){
   gender.set(str[4]); if((str[1].equals("0")) ){
if(str[5].matches("\\d

+")){ int

i=Integer.parseInt(str

[5]);

age.set(i);
}
}
}
context.write(gender, age)
}
}
```

Reducer Code:

```
public static class Reduce extends Reducer<Text,IntWritable, Text,
IntWritable> {public void reduce(Text key, Iterable<IntWritable> values,
Context context) throws IOException, InterruptedException {
int
sum
= 0;
int
l=0;
for (IntWritable val :
values) {l+=1;
```

```
sum += val.get();
}
sum=sum/l;
context.write(key, new IntWritable(sum));
}
}
```

Configuration Code: job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(IntWritable.class);

https://github.com/kiran0541/Map-Reduce/blob/master/Average%20age%20of%20male%20and%20female%20people%20died%20in%20titanic

Way to to execute the Jar file to get the result of the first problem statement:

hadoop jar average.jar /TitanicData.txt /avg_out

Here 'hadoop' specifies we are running a Hadoop command and jar specifies which type of application we are running and average.jar is the jar file which wehave created which consists the above source code and the path of the Input filename in our case it is TitanicData.txt and the output file where to store the outputthere we have given it as avg out.

Way to view the output:

hadoop dfs –cat /avg_out/part-r-00000

Here 'hadoop' specifies that we are running a Hadoop commandand 'dfs' specifies that we are performing an operation related to HadoopDistributed File System and '- cat' is used to view the contents of a file and'avg_out/part-r-00000' is the file where the output is stored. Part file is created by default by the TextInputFormat class of Hadoop.

Output:

Result:

Assignment: 15

Hive: Introduction Creation of Database and Table, Hive Partition, Hive Built in Function and Operators, Hive View and Index.

Description:

Hive is an open-source data warehousing solution built on top of Hadoop. It supports an SQL-likequery language called HiveQL. These queries are compiled into MapReduce jobs that are executedon Hadoop. While Hive uses Hadoop for execution of queries, it reduces the effort that goes into writing and maintaining MapReduce jobs.

Hive supports database concepts like tables, columns, rows and partitions. Both primitive (integer,float, string) and complex data-types(map, list, struct) are supported. Moreover, these types can becomposed to support structures of arbitrary complexity. The tables are serialized/deserialized usingdefault serializers/deserializer. Any new data format and type can be supported by implementing SerDe and ObjectInspector java interface.

RESOURCES:

VMWare, XAMPP Server, Web Browser, 1GB RAM, Hard Disk 80 GB.

PROGRAM LOGIC:

SYNTAX for HIVE Database
OperationsDATABASE Creation
CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database name>
Drop Database Statement
DROP DATABASE StatementDROP (DATABASE|SCHEMA) [IF EXISTS]
database_name
[RESTRICT|CASCADE];
Creating and Dropping Table in
HIVE
CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]
table_name
[(col_name data_type [COMMENT col_comment], ...)]
[COMMENT table_comment] [ROW FORMAT row_format]
[STORED ASfile_format]

Loading Data into table
log_dataSyntax:
LOAD DATA LOCAL INPATH '<path>/u.data' OVERWRITE INTO TABLE
u_data;
Alter Table
in HIVE
Syntax
ALTER TABLE name RENAME TO new_name
ALTER TABLE name ADD COLUMNS (col_spec[,
col_spec ...])ALTER TABLE name DROP [COLUMN]
column_name
ALTER TABLE name CHANGE column_name new_name
new_type ALTER TABLE name REPLACE COLUMNS
(col_spec[, col_spec ...])Creating and Dropping View
CREATE VIEW [IF NOT EXISTS] view_name [(column_name
[COMMENT

column_comment], ...)] [COMMENT table_comment] AS SELECT ...
Droppi
ng
View
Syntax:
DROP VIEW
view_name
Functions in
HIVE
String Functions:- round(), ceil(), substr(), upper(), reg_exp() etc
Date and Time Functions:- year(), month(), day(),
to_date() etcAggregate Functions :- sum(), min(),
max(), count(), avg() etc INDEXES
CREATE INDEX index_name ON TABLE base_table_name
(col_name, ...)AS 'index.handler.class.name'
[WITH DEFERRED REBUILD]
[IDXPROPERTIES
(property_name=property_value, ...)][IN TABLE
index_table_name]
[PARTITIONED BY
(col_name, ...)][
[ROW FORMAT ...] STORED AS ...
| STORED BY ...

]
[LOCATION
hdfs_path]
[TBLPROPERTI
ES (...)]
Creating Index
CREATE INDEX index_ip ON TABLE log_data(ip_address) AS
'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler' WITH
DEFERREDREBUILD;
Altering and Inserting Index
ALTER INDEX index_ip_address ON log_data
REBUILD;Storing Index Data in Metastore
SET
hive.index.compact.file=/home/administrator/Desktop/big/metastore_db/tmp/index_ipad
d ress_result;
SET
hive.input.format=org.apache.hadoop.hive.ql.index.compact.HiveCompactIndexI
nputFor mat;
Dropping Index
DROP INDEX INDEX_NAME on TABLE_NAME;

OUTPUT:

Assignment: 16

Aim: Study of the TensorFlow and OpenAI Gym Library

Objective: The objective of this lab exercise is to introduce students to the TensorFlow and OpenAI Gym libraries, equipping them with the essential tools for implementing and experimenting with reinforcement learning algorithms.

Outcome: By the end of this lab exercise, students will have gained a thorough understanding of the TensorFlow and OpenAI Gym libraries. They will be able to develop and experiment with reinforcement learning algorithms using these libraries, paving the way for further exploration and research in the field of reinforcement learning.

Theory:

TensorFlow: TensorFlow is an open-source machine learning framework developed by Google. It provides tools for building and training various machine learning models, including deep neural networks. TensorFlow offers flexibility and scalability, making it suitable for both research and production environments.

Key Features of TensorFlow:

- **Automatic Differentiation**: TensorFlow's computational graph abstraction enables automatic differentiation, allowing users to compute gradients efficiently for training deep neural networks.
- **TensorBoard**: TensorFlow includes TensorBoard, a visualization toolkit, to visualize computational graphs, monitor training metrics, and debug models.
- **High-level APIs**: TensorFlow provides high-level APIs like Keras, tf.data, and tf.estimator to simplify the process of building and training machine learning models.
- **Deployment Flexibility**: TensorFlow models can be deployed across various platforms, including desktops, servers, mobile devices, and the cloud, using TensorFlow Serving, TensorFlow Lite, or TensorFlow.js.

Applications:

- **Image Recognition**: TensorFlow is widely used for image recognition tasks, such as object detection, image classification, and semantic segmentation. Applications include medical imaging, self-driving cars, and security surveillance.
- **Natural Language Processing (NLP)**: TensorFlow powers state-of-the-art NLP models for tasks like text classification, sentiment analysis, machine translation, and named entity recognition. Applications range from chatbots and virtual assistants to language translation services.
- **Reinforcement Learning**: TensorFlow is a popular choice for implementing reinforcement

learning algorithms due to its flexibility and efficiency. Applications include game playing (e.g., AlphaGo), robotics, and autonomous systems.

- **Recommendation Systems**: TensorFlow enables the development of recommendation systems that personalize content and product recommendations based on user preferences and behavior. Applications include e-commerce platforms, streaming services, and social media platforms.
- **Time Series Analysis**: TensorFlow supports the creation of models for time series forecasting, anomaly detection, and pattern recognition. Applications include financial forecasting, weather prediction, and industrial process monitoring.

OpenAI Gym: OpenAI Gym is a toolkit for developing and comparing reinforcement learning algorithms. It provides a wide range of environments, from simple toy problems to complex simulations, allowing researchers to test their algorithms in diverse settings. OpenAI Gym simplifies the process of interfacing with environments, making it easier to focus on algorithm development.

Key Features of OpenAI Gym:

- **Environment Abstraction**: OpenAI Gym provides a simple and consistent interface for interacting with reinforcement learning environments, allowing users to focus on algorithm development.
- **Standardized Benchmarks**: OpenAI Gym includes a variety of environments with standardized benchmarks, enabling fair comparison and evaluation of reinforcement learning algorithms.
- **Easy Integration**: OpenAI Gym environments can be easily integrated with popular reinforcement learning libraries like TensorFlow, PyTorch, and Keras, making it a preferred choice for researchers and practitioners.

Applications:

- **Game Playing**: OpenAI Gym provides environments for classic arcade games (e.g., Atari games), board games (e.g., Chess, Go), and simulated environments (e.g., MuJoCo) for training agents to play games autonomously.
- **Robotics**: OpenAI Gym environments simulate robotic tasks such as manipulation, navigation, and control. Researchers use these environments to develop and test algorithms for robot learning and control.
- **Autonomous Vehicles**: OpenAI Gym environments model driving scenarios, allowing researchers to train agents to navigate roads, follow traffic rules, and interact with other vehicles in simulated environments.
- **Finance**: OpenAI Gym provides environments for modeling and optimizing financial processes, including portfolio management, algorithmic trading, and risk assessment.
- **Healthcare**: OpenAI Gym environments can be adapted to simulate healthcare scenarios, such as patient diagnosis, treatment planning, and drug discovery, facilitating research in medical AI.

Steps for Implementation:

1. Install TensorFlow and OpenAI Gym libraries using pip or conda.
2. Explore the documentation and resources available for both libraries to understand

their functionalities and usage.

3. Experiment with simple examples provided in the documentation to get hands-on experience with TensorFlow and OpenAI Gym.
4. Develop custom environments using OpenAI Gym to understand how environments are defined and interacted with.
5. Implement basic neural networks using TensorFlow to understand its syntax and operations.

Conclusion:

Assignment: 17

AIM: Implement a Deep Q-Network (DQN) using a deep neural network library (e.g., TensorFlow or PyTorch) and train it on a simple environment like CartPole or Mountain Car.

Q Learning

Let‟s say we know the expected reward of each action at every step. This would essentially be like a cheat sheet for the agent! Our agent will know exactly which action to perform.

It will perform the sequence of actions that will eventually generate the maximum total reward. This total reward is also called the Q-value and we will formalise our strategy as:

$$Q(s,a) = r(s,a) + \gamma \max_a Q(s',a)$$

The above equation states that the Q-value yielded from being at state *s* and performing action *a* isthe immediate reward r(s,a) plus the highest Q-value possible from the next state *s'*. Gamma here is the discount factor which controls the contribution of rewards further in the future.

Q(s‟,a) again depends on Q(s”,a) which will then have a coefficient of gamma squared. So, the Q- value depends on Q-values of future states as shown here:

$$Q(s,a) \rightarrow \gamma Q(s',a) + \gamma^2 Q(s'',a) \dots\dots\dots \gamma^n Q(s''^{\dots n},a)$$

Adjusting the value of gamma will diminish or increase the contribution of future rewards.

Since this is a recursive equation, we can start with making arbitrary assumptions for all q-values. With experience, it will converge to the optimal policy. In practical situations, this is implemented as an update:

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha\left[R_{t+1} + \gamma \max_a Q(S_{t+1},a) - Q(S_t,A_t)\right]$$

where alpha is the learning rate or step size. This simply determines to what extent newly acquired information overrides old information.

Why 'Deep' Q-Learning?

Q-learning is a simple yet quite powerful algorithm to create a cheat sheet for our agent. This helps the agent figure out exactly which action to perform.

But what if this cheatsheet is too long? Imagine an environment with 10,000 states and 1,000 actions per state. This would create a table of 10 million cells. Things will quickly get out of control!

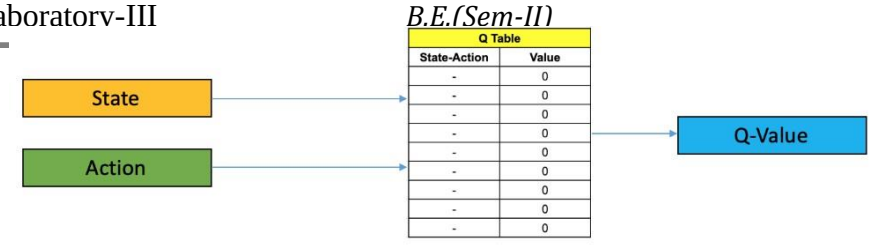
It is pretty clear that we can"t infer the Q-value of new states from already explored states. This presents two problems:

- First, the amount of memory required to save and update that table would increase as thenumber of states increases
- Second, the amount of time required to explore each state to create the required Q-tablewould be unrealistic

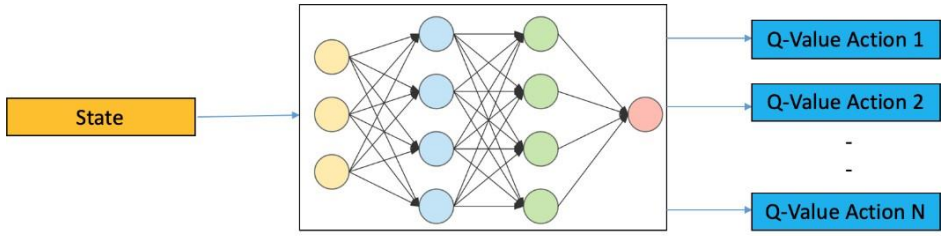
Here"s a thought – what if we approximate these Q-values with machine learning models such as a neural network? Well, this was the idea behind DeepMind"s algorithm that led to its acquisition by Google for 500 million dollars!

Deep Q-Networks

In deep Q-learning, we use a neural network to approximate the Q-value function. The state is given as the input and the Q-value of all possible actions is generated as the output. The comparison between Q-learning & deep Q-learning is wonderfully illustrated below:



Q Learning



Deep Q Learning

So, what are the steps involved in reinforcement learning using deep Q-learning networks (DQNs)?

1. All the past experience is stored by the user in memory
2. The next action is determined by the maximum output of the Q-network
3. The loss function here is mean squared error of the predicted Q-value and the target Q-value
   – Q*. This is basically a regression problem. However, we do not know the target or actual value here as we are dealing with a reinforcement learning problem. Going back to the Q- value update equation derived fromthe Bellman equation. we have:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right]$$

The section in green represents the target. We can argue that it is predicting its own value, but sinceR is the unbiased true reward, the network is going to update its gradient using backpropagation to finally converge.

Challenges in Deep RL as Compared to Deep Learning

So far, this all looks great. We understood how neural networks can help the agent learn the best actions. However, there is a challenge when we compare deep RL to deep learning (DL):

- **Non-stationary or unstable target:** Let us go back to the pseudocode for deep Q-learning:
-

Start with $Q_0(s, a)$ for all s, a.

Get initial state s

For k = 1, 2, ... till convergence

Sample action a, get next state s'

If s' is terminal:

$$\text{target} = R(s, a, s')$$

Sample new initial state s'

else:

$$\text{target} = R(s, a, s') + \gamma \max_{a'} Q_k(s', a')$$

$$\theta_{k+1} \leftarrow \theta_k - \alpha \nabla_\theta \mathbb{E}_{s' \sim P(s'|s,a)} \left[(Q_\theta(s, a) - \text{target}(s'))^2 \right] \Big|_{\theta=\theta_k}$$

$$s \leftarrow s'$$

Chasing a nonstationary target!

Updates are correlated within a trajectory!

As you can see in the above code, the target is continuously changing with each iteration. In deep learning, the target variable does not change and hence the training is stable, which is just not true forRL.

To summarise, we often depend on the policy or value functions in reinforcement learning to sample actions. However, this is frequently changing as we continuously learn what to explore. As we play out the game, we get to know more about the ground truth values of states and actions and hence, the output is also changing.

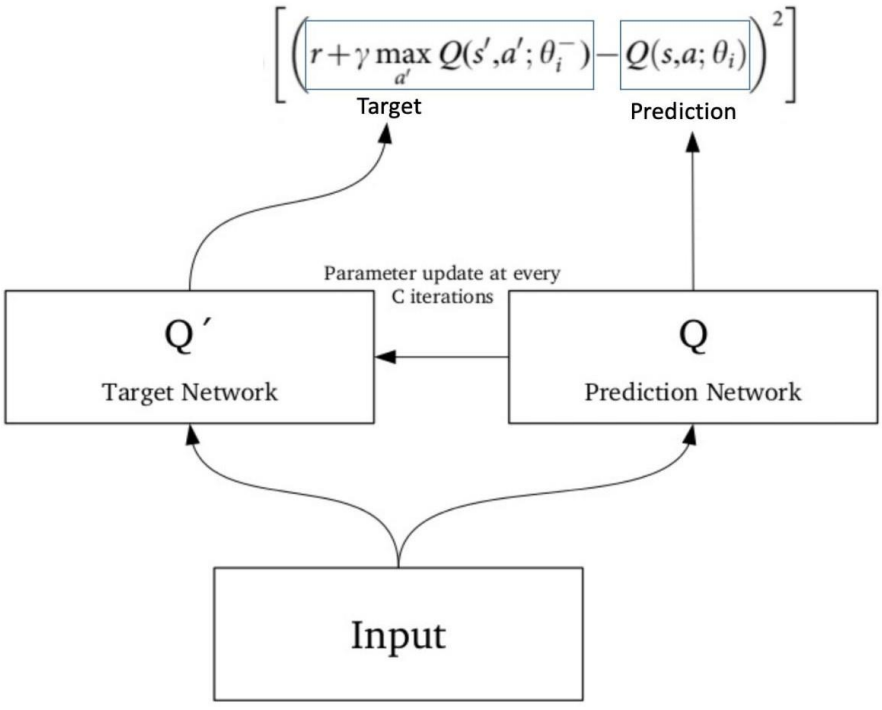
So, we try to learn to map for a constantly changing input and output. But then what is the solution?

1. Target Network

Since the same network is calculating the predicted value and the target value, there could be a lot of divergence between these two. So, instead of using 1one neural network for learning, we can use two.

We could use a separate network to estimate the target. This target network has the same architectureas the function approximator but with frozen parameters. For every C iterations (a hyperparameter), the parameters from the prediction network are

copied to the target network. This leads to morestable training because it keeps the target function fixed (for a while):



1. Experience Replay

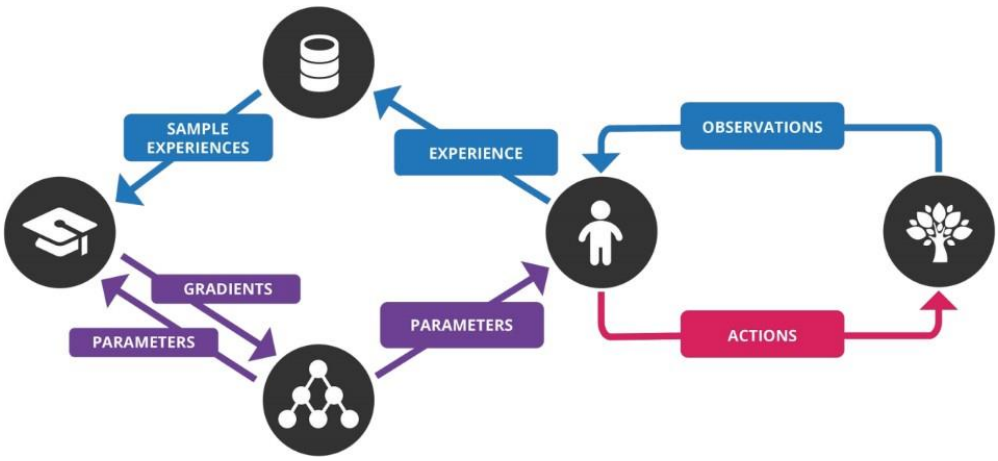
To perform experience replay, we store the agent's experiences – et=(st,at,rt,st+1)
What does the above statement mean? Instead of running Q-learning on state/action pairs as they occur during simulation or the actual experience, the system stores the data discovered for [state, action, reward, next_state] – in a large table.

Let‟s understand this using an example.

Suppose we are trying to build a video game bot where each frame of the game represents a different state. During training, we could sample a random batch of 64 frames from the last 100,000 frames to train our network. This would get us a subset within which the correlation amongst the samples is low and will also provide better sampling efficiency.

Putting it all Together

The concepts we have learned so far? They all combine to make the deep Q-learning algorithm that was used to achive human-level level performance in Atari games (using just the video frames of the game).



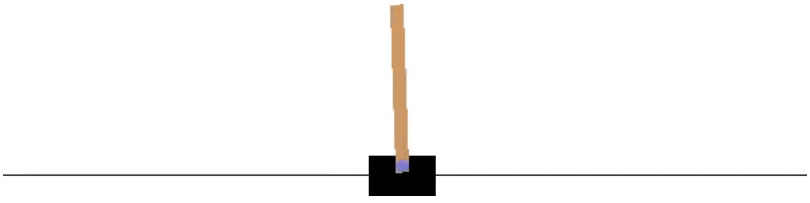
I have listed the steps involved in a deep Q-network (DQN) below:

1. Preprocess and feed the game screen (state s) to our DQN, which will return the Q-values of all possible actions in the state
2. Select an action using the epsilon-greedy policy. With the probability epsilon, we select a random action *a*and with probability 1-epsilon, we select an action that has a maximum Q- value, such as a = argmax(Q(s,a,w))
3. Perform this action in a state *s* and move to a new state *s'* to receive a reward. This state s‟ is the preprocessed image of the next game screen. We store this transition in our replay buffer as <s,a,r,s‟>
4. Next, sample some random batches of transitions from the replay buffer and calculate the loss
5. It is known that: $Loss = (r + \gamma max_{a'} Q(s', a'; \theta') - Q(s, a; \theta))^2$ which is just the squared
difference between target Q and predicted Q
6. Perform gradient descent with respect to our actual network parameters in order to minimizethis loss
7. After every C iterations, copy our actual network weights to the target network weights
8. Repeat these steps for *M* number of episodes

Implementing Deep Q-Learning in Python using Keras & OpenAI Gym

Alright, so we have a solid grasp on the theoretical aspects of deep Q-learning. How about seeing itin action now? That‟s right – let‟s fire up our Python notebooks!

We will make an agent that can play a game called CartPole. We can also use an Atari game but training an agent to play that takes a while (from a few hours to a day). The idea behind our approachwill remain the same so you can try this on an Atari game on your machine.



CartPole is one of the simplest environments in the OpenAI gym (a game simulator).

As you can see in the above animation, the goal of CartPole is to balance a pole that"s connected with one joint on top of a moving cart.

Instead of pixel information, there are four kinds of information given by the state (such as the angle of the pole and position of the cart). An agent can move the cart by performing a series of actions of0 or 1, pushing the cart left or right.

We will use the *keras-rl* library here which lets us implement deep Q-learning out of the box.

Step 1: Install keras-rl library

From the terminal, run the following code block:

```
git clone https://github.com/matthiasplappert/keras-rl.git

cd keras-rl

python setup.py install
```

Step 2: Install dependencies for the CartPole environment

Assuming you have pip installed, you need to install the following libraries:

```
pip install h5py

pip install gym
```

Step 3: Let's get started!

First, we have to import the necessary modules:

```
import numpy as np

import gym



from keras.models import Sequential

from keras.layers import Dense, Activation, Flatten

from keras.optimizers import Adam



from rl.agents.dqn import DQNAgent

from rl.policy import EpsGreedyQPolicy
```

```
from rl.memory import SequentialMemory
```

Then, set the relevant variables:

```
ENV_NAME = 'CartPole-v0'



# Get the environment and extract the number of actions available in the Cartpole problem

env = gym.make(ENV_NAME)

np.random.seed(123)

env.seed(123)

nb_actions = env.action_space.n
```

Next, we will build a very simple single hidden layer neural network model:

```
model = Sequential()

model.add(Flatten(input_shape=(1,) + env.observation_space.shape))

model.add(Dense(16))

model.add(Activation('relu'))

model.add(Dense(nb_actions))
```

```
model.add(Activation('linear'))

print(model.summary())
```

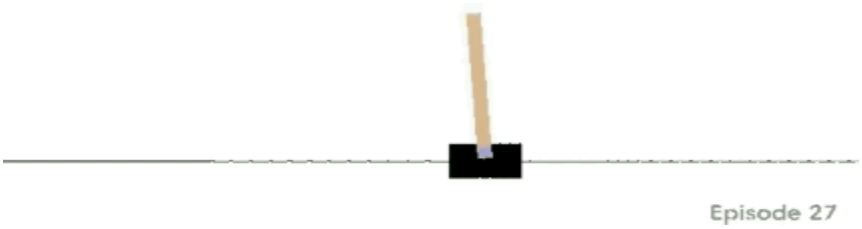
Now, configure and compile our agent. We will set our policy as Epsilon Greedy and our memory as Sequential Memory because we want to store the result of actions we performed and the rewards we get for each action.

```
policy = EpsGreedyQPolicy()

memory = SequentialMemory(limit=50000, window_length=1)

dqn = DQNAgent(model=model, nb_actions=nb_actions, memory=memory, nb_steps_warmup=10,

target_model_update=1e-2, policy=policy)

dqn.compile(Adam(lr=1e-3), metrics=['mae'])



# Okay, now it's time to learn something! We visualize the training here for show, but this slows down training quite a lot.

dqn.fit(env, nb_steps=5000, visualize=True, verbose=2)
```

Test our reinforcement learning model:

```
dqn.test(env, nb_episodes=5, visualize=True)
```

This will be the output of our model:



Not bad! Congratulations on building your very first deep Q-learning model.

Assignment 18

Aim: Implement a policy gradient algorithm like REINFORCE or Proximal Policy Optimization (PPO) to solve a continuous control task

Objective: The objective of this practical assignment is to implement a policy gradient algorithm, specifically REINFORCE (vanilla policy gradient) or Proximal Policy Optimization (PPO), to address a continuous control task.

Theory:

- **Policy Gradient Methods**: Policy gradient methods are a class of reinforcement learning algorithms that directly parameterize the policy function and update it based on the gradient of expected rewards. These methods offer advantages in handling continuous action spaces and are particularly suitable for tasks where explicit exploration is required.
- **REINFORCE (Vanilla Policy Gradient)**: REINFORCE is a fundamental policy gradient algorithm that estimates the policy gradient using samples collected during interaction with the environment. It updates the policy parameters in the direction of the gradient of expected rewards, scaled by the advantage function or returns. REINFORCE is simple to implement and serves as the basis for more advanced policy gradient algorithms.
- **Proximal Policy Optimization (PPO)**: PPO is a state-of-the-art policy gradient algorithm designed to address issues like stability, sample efficiency, and policy updates in reinforcement learning. It employs a surrogate objective function and a clipping mechanism to ensure smooth policy updates and prevents large policy changes. PPO strikes a balance between exploration and exploitation and has demonstrated robust performance across various domains.
- **Applications**:
    - **Robotics**: Policy gradient methods are widely used in robotics for controlling robot manipulators, quadcopters, and other autonomous systems. They enable precise and adaptive control of continuous actions in dynamic and uncertain environments.
    - **Game Playing**: Policy gradient algorithms have been applied to play video games, board games, and strategic games, achieving human-level or superhuman performance in many cases. They learn policies directly from raw observations, making them suitable for a wide range of gaming scenarios.
    - **Finance**: Policy gradient methods can optimize trading strategies, portfolio management, and risk assessment in financial markets. They adapt to changing market conditions and learn optimal policies for maximizing returns while minimizing risks.
    - **Healthcare**: Policy gradient algorithms are utilized in healthcare for personalized treatment planning, medical imaging analysis, and drug discovery. They assist healthcare professionals in making informed decisions based on patient data and clinical guidelines.

Algorithm: REINFORCE (Vanilla Policy Gradient) or Proximal Policy Optimization (PPO)

Steps for Implementation:

1. Select a continuous control task environment from OpenAI Gym or design a custom environment if necessary.
2. Implement the chosen policy gradient algorithm (REINFORCE or PPO) using a deep learning framework like TensorFlow or PyTorch.
3. Define the policy network architecture to map states to actions in the chosen environment.
4. Collect trajectories by interacting with the environment and compute the policy gradient using rewards obtained.
5. Update the policy parameters using gradient ascent or other optimization techniques, ensuring convergence and stability.
6. Implement exploration strategies (e.g., Gaussian noise) to encourage exploration and prevent premature convergence.
7. Monitor training progress, including rewards obtained, policy updates, and convergence metrics, to evaluate algorithm performance.
8. Experiment with different hyperparameters, network architectures, and training configurations to optimize performance and convergence speed.

Example Code:

```
import gym
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam

class PolicyNetwork(tf.keras.Model):
    def __init__(self, num_actions):
        super(PolicyNetwork, self).__init__()
        self.dense1 = Dense(64, activation='relu')
        self.dense2 = Dense(64, activation='relu')
        self.dense3_mean = Dense(num_actions, activation='tanh')
        self.dense3_std = Dense(num_actions, activation='softplus')

    def call(self, state):
        x = self.dense1(state)
        x = self.dense2(x)
        mean = self.dense3_mean(x)
        std = self.dense3_std(x)
        return mean, std

class PPOAgent:
    def __init__(self, num_actions, state_shape, lr_actor=0.0003, lr_critic=0.001, gamma=0.99, clip_ratio=0.2, value_coef=0.5, entropy_coef=0.01):
```

```
        self.policy_network = PolicyNetwork(num_actions)
        self.optimizer = Adam(learning_rate=lr_actor)
        self.gamma = gamma
        self.clip_ratio = clip_ratio
        self.value_coef = value_coef
        self.entropy_coef = entropy_coef

    def choose_action(self, state):
        mean, std = self.policy_network(state)
        action = tf.random.normal(tf.shape(mean), mean, std)
        log_prob = self.log_probability(action, mean, std)
        return action, log_prob

    def log_probability(self, action, mean, std):
        dist = tfp.distributions.Normal(mean, std)
        log_prob = dist.log_prob(action)
        return tf.reduce_sum(log_prob, axis=1)

    def compute_loss(self, states, actions, advantages, old_log_probs):
        mean, std = self.policy_network(states)
        log_probs = self.log_probability(actions, mean, std)
        ratio = tf.exp(log_probs - old_log_probs)
        clipped_ratio = tf.clip_by_value(ratio, 1 - self.clip_ratio, 1 + self.clip_ratio)
        actor_loss = -tf.reduce_mean(tf.minimum(ratio * advantages, clipped_ratio *
advantages))

        values = self.policy_network(states)[0]
        critic_loss = tf.reduce_mean(tf.square(values - advantages))

        entropy = tf.reduce_mean(-log_probs)
        total_loss = actor_loss + self.value_coef * critic_loss - self.entropy_coef *
entropy
        return total_loss

    def train_step(self, states, actions, advantages, old_log_probs):
        with tf.GradientTape() as tape:
            loss = self.compute_loss(states, actions, advantages, old_log_probs)
        gradients = tape.gradient(loss, self.policy_network.trainable_variables)
        self.optimizer.apply_gradients(zip(gradients,
self.policy_network.trainable_variables))
```

Conclusion:

Assignment 19

Aim: Build a multi-agent environment, such as a cooperative or competitive game, and implement algorithms like Independent Q-Learning or Multi-Agent Deep Deterministic Policy Gradients (MADDPG)

Objective: The objective of this practical assignment is to build a multi-agent environment, such as a cooperative or competitive game, and implement the Multi-Agent Deep Deterministic Policy Gradients (MADDPG) algorithm to train agents to collaborate effectively in solving tasks within the environment.

Theory:

- **Multi-Agent Reinforcement Learning**: Multi-agent reinforcement learning (MARL) deals with scenarios where multiple agents interact with each other and the environment to achieve individual or collective goals. MARL introduces challenges such as coordination, competition, communication, and the emergence of complex behaviors through interaction.
- **Multi-Agent Deep Deterministic Policy Gradients (MADDPG)**: MADDPG is an extension of the Deep Deterministic Policy Gradients (DDPG) algorithm for multi-agent settings. It combines centralized training with decentralized execution, where each agent learns its policy based on local observations but considers the actions of other agents during training. MADDPG uses a centralized critic network to estimate the value of joint actions and decentralized actor networks to select individual actions.

Algorithm: Multi-Agent Deep Deterministic Policy Gradients (MADDPG)

Steps for Implementation:

1. Define a multi-agent environment that captures the interactions and objectives of the task (e.g., cooperative or competitive game).
2. Implement actor-critic neural networks for each agent, including actor networks for action selection and critic networks for value estimation.
3. Initialize target networks for stability and implement mechanisms for periodically updating target network parameters.
4. Set up a replay buffer to store experiences for training and implement methods for sampling batches of experiences.
5. Implement the MADDPG training loop, including action selection, experience collection, training steps, and target network updates.
6. Tune hyperparameters such as learning rates, discount factors, batch sizes, and target update intervals to optimize algorithm performance.
7. Monitor training progress, including episode rewards, convergence metrics, and agent behaviors, to evaluate algorithm effectiveness and stability.

Pseudo Code:

```
Initialize replay buffer R
Initialize actor-critic networks for each agent
Initialize target actor-critic networks for each agent with the same weights
Set hyperparameters: discount factor gamma, actor learning rate lr_actor, critic learning rate lr_critic
Set target update interval N
Set maximum number of episodes M
Set batch size B

for episode = 1 to M:
    Initialize episode reward to 0
    Reset environment to initial state
    for each agent:
        Reset agent state
    while not done:
        for each agent:
            Choose action using the current actor network and state
            Execute action in the environment
            Store transition (state, action, reward, next_state, done) in replay buffer
            Accumulate reward for the episode
        if size of replay buffer > B:
            for each agent:
                Sample batch of experiences from replay buffer
                Update actor-critic networks using MADDPG algorithm:
                    - Compute target Q-value using target critic network and target actor network
                    - Compute critic loss using predicted Q-value from critic network and target Q-value
                    - Compute actor loss using gradient of critic with respect to actions
                    - Update actor and critic networks using gradients
        if episode % N == 0:
            for each agent:
                Update target actor-critic network with weights from current actor-critic network
        if done:
            Break loop

    Print episode reward
```

Conclusion:

Assignment 20

Aim: Develop an actor-critic model using neural networks and train it on a classic RL benchmark, such as the Acrobot or Inverted Pendulum

Objective: The objective of this practical assignment is to develop an actor-critic model using neural networks and train it on a classic reinforcement learning (RL) benchmark, such as the Acrobot or Inverted Pendulum. This assignment aims to provide hands-on experience in implementing the actor-critic algorithm, which combines aspects of both value-based and policy-based methods, and apply it to solve challenging control tasks.

Theory:

- **Actor-Critic Algorithm:** The actor-critic algorithm is a hybrid reinforcement learning (RL) approach that combines elements of both policy-based and value-based methods. It maintains two separate networks: an actor network that learns a policy to select actions, and a critic network that estimates the value of state-action pairs. The actor network is updated based on policy gradients to improve the policy, while the critic network is updated based on temporal difference (TD) errors to estimate the value function. Actor-critic algorithms offer advantages such as stability, efficiency, and the ability to handle continuous action spaces.
- **Neural Networks:** Neural networks are a fundamental component of actor-critic models, used to represent both the actor and critic networks. The actor network typically outputs a probability distribution over actions given a state, while the critic network estimates the value function or action-value function. Deep neural networks with multiple layers are often employed to capture complex mappings between states and actions. Training neural networks involves optimizing their parameters using gradient-based optimization techniques such as stochastic gradient descent (SGD) or variants like Adam.
- **Classic RL Benchmarks:** Classic RL benchmarks, such as the Acrobot or Inverted Pendulum, are standard control tasks used to evaluate and compare reinforcement learning algorithms. These benchmarks serve as standard testbeds for benchmarking the performance of RL algorithms and assessing their ability to solve complex control problems. For example, the Inverted Pendulum task involves balancing an inverted pendulum on a cart by applying appropriate control actions, while the Acrobot task requires swinging up a double pendulum to a certain height using torque control.
- **Applications:** Actor-critic algorithms have been successfully applied to various real-world problems in robotics, autonomous systems, finance, gaming, and more. For example, in robotics, actor-critic methods are used for robot control tasks such as manipulation, locomotion, and navigation. In finance, actor-critic algorithms can be employed for automated trading strategies and portfolio management. In gaming, actor-critic models can learn to play complex video games by interacting with the game environment and optimizing their policies to maximize rewards. Overall, actor-critic algorithms offer a versatile and powerful framework for addressing a wide range of sequential decision-making problems.

Algorithm: Actor-Critic Algorithm using Neural Networks

Steps for Implementation:

1. Choose a classic RL benchmark environment, such as the Acrobot or Inverted Pendulum, from OpenAI Gym or a similar library.
2. Design and implement actor and critic neural networks using a deep learning framework like TensorFlow or PyTorch.
3. Initialize the networks' parameters and set hyperparameters such as learning rates, discount factor, and exploration strategy.
4. Implement the actor-critic training loop, which consists of interacting with the environment, collecting experiences, updating the networks' weights, and repeating until convergence.
5. Monitor training progress by evaluating the model's performance on the benchmark task and visualizing learning curves (e.g., episode rewards or value estimates) over time.
6. Fine-tune hyperparameters and network architectures to improve performance and convergence speed.

Pseudo Code:

```
Initialize actor and critic neural networks with random weights
Initialize target networks with the same weights as actor and critic networks
Initialize replay buffer to store experiences
Initialize environment for the chosen RL benchmark (e.g., Acrobot or Inverted Pendulum)
Set hyperparameters: learning rate, discount factor, target update interval, batch size, etc.

for episode = 1 to max_episodes:
    Initialize episode-specific variables: total_reward = 0, state = initial_state
    for step = 1 to max_steps_per_episode:
        Sample action from actor network with exploration strategy (e.g., ε-greedy)
        Execute action in environment, observe next_state, reward, and done
        Store transition (state, action, reward, next_state, done) in replay buffer
        total_reward += reward
        state = next_state

        if replay buffer size >= batch_size:
            Sample batch of experiences from replay buffer
            Update critic network:
                Compute target value using target critic network and next_state
                Compute critic loss as mean squared error between predicted and target values
                Backpropagate gradients and update critic network parameters

            Update actor network:
                Compute advantage using TD error (target value - predicted value)
                Compute actor loss as negative log probability of selected action multiplied by advantage
                Backpropagate gradients and update actor network parameters

        if step % target_update_interval == 0:
            Update target networks with parameters from actor and critic networks
```

if done:
Break inner loop

Print episode number and total_reward

Conclusion: