**PANIMALAR INSTITUTE OF TECHNOLOGY**

## JAISAKTHI EDUCATIONAL TRUST

**(Affiliated to Anna University, Chennai) Bangalore Trunk Road, Varadharajapuram, Poonamallee, Chennai – 600 123**



**DEPARTMENT OF**

**ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**AD8711 – DEEP LEARNING LABORATORY VII SEMESTER – IV YEAR**

**LAB MANUAL ACADEMIC YEAR 2023-24**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Ex. No.** | **CONTENTS** | **PAGE No.** |
| **-** | VISION AND MISSION OF THE INSTITUTE | iii |
| **-** | VISION AND MISSION OF THE DEPARMENT | iv |
| **-** | PROGRAM EDUCATIONAL OBJECTIVES | v |
| **-** | PROGRAM OUTCOMES | vi |
| **-** | PROGRAM SPECIFIC OUTCOMES | vii |
| **-** | SYLLABUS | viii |

**VISION AND MISSION OF THE INSTITUTE**

**VISION**

An Institution of Excellence by imparting quality education and serve as a perennial source of technical manpower with dynamic professionalism and entrepreneurship having social responsibility for the progress of the society and nation

# MISSION

Panimalar Institute of Technology will strive to emerge as an Institution of Excellence inthe country by

* Providing state-of-the-art infrastructure facilities for designing and developingsolutions for engineering problems.
* Imparting quality education and training through qualified, experienced and committed members of the faculty**.**
* Inculcating high moral values in the minds of the Students and transforming them into a well-rounded personality**.**
* Establishing Industry Institute interaction to make students ready for the industrial environment.
* Promoting research based projects/activities in the emerging areas of Engineering & Technology.

# VISION AND MISSION OF THE DEPARTMENT

**VISION**

To establish a unique standard of quality education by enriching the problem solving skillsthat adapt swiftly to the challenges of the society and industry. Producing professionals who shall be the leaders in technology employing Artificial Intelligence and Data Science along with core Computer Science.

# MISSION

* To create an academic environment for higher learning, academic practices and research endeavours.
* To educate the students with latest technologies to update their knowledge in the field ofAI and Data science.
* To empower students with knowledge through state-of-art infrastructure and curriculum.
* To produce successful professionals to serve the needs of Industry and society.
* To produce entrepreneurs in Artificial Intelligence and Data Science through excellence in education and research.

# PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

|  |  |
| --- | --- |
| **PEO 1 :** | To provide graduates with the proficiency to utilize the fundamental knowledge of  basic sciences, mathematics, Artificial Intelligence, data science and statistics to build systems that require management and analysis of large volume of data. |
| **PEO 2 :** | To enrich graduates with necessary technical skills to pursue pioneering research in the field of AI and Data Science and create disruptive and sustainable solutions for  the welfare of ecosystems. |
| **PEO 3 :** | To enable graduates to think logically, pursue lifelong learning and collaborate with  an ethical attitude in a multidisciplinary team. |
| **PEO 4 :** | To enable the graduates to design and model AI based solutions to critical problem  domains in the real world. |
| **PEO 5 :** | To enrich the innovative thoughts and creative ideas of the graduates for effective  contribution towards economy building. |

**PROGRAM OUTCOMES OF THE DEPARTMENT**

Engineering Graduates will be able to:

|  |  |
| --- | --- |
| **PO 1 :** | **Engineering Knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of  complex engineering problems |
| **PO 2 :** | **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. |
| **PO 3 :** | **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. |
| **PO 4 :** | **Conduct Investigations of Complex Problems:** Use research-based knowledge  andresearch methods, including design of experiments, analysis and interpretation of data, andsynthesis of the information to provide valid conclusions. |
| **PO 5 :** | **Modern Tool Usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling of complex  engineering activities with an understanding of the limitations. |
| **PO 6 :** | **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 practice. |
| **PO 7 :** | **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. |
| **PO 8 :** | **Ethics:** Apply ethical principles and commit to professional ethics and  responsibilities and norms of the engineering practice. |
| **PO 9 :** | **Individual and Team Work:** Function effectively as an individual, and as a  member or leader in diverse teams, and in multidisciplinary settings. |
| **PO 10 :** | **Communication:** 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. |
| **PO 11 :** | **Project Management and Finance:** Demonstrate knowledge and understanding of the 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. |
| **PO 12 :** | **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 OF THE DEPARTMENT

|  |  |
| --- | --- |
| **PSO 1 :** | Graduates should be able to evolve AI based efficient domain specific processes  for effective decision making in several domains such as business and governance domains. |
| **PSO 2 :** | Graduates should be able to arrive at actionable Fore sight, Insight , hind sight  from data for solving business and engineering problems. |
| **PSO 3 :** | Graduates should be able to create, select and apply the theoretical knowledge of AI and Data Analytics along with practical industrial tools and techniques to manage  and solve wicked societal problems. |
| **PSO 4 :** | Graduates should be capable of developing data analytics and data visualization  skills, skills pertaining to knowledge acquisition, knowledge representation and knowledge engineering, and hence capable of coordinating complex projects |
| **PSO 5 :** | Graduates should be able to carry out fundamental research to cater the critical  needs of the society through cutting edge technologies of AI. |

## SYLLABUS

**OBJECTIVES:**

## AD8711 – DEEP LEARNING LABORATORY L T P C

**0 0 4 2**

* To learn deep neural networks and apply for simple problems
* To Learn and apply Convolution Neural Network for image processing
* To Learn and apply Recurrent Neural Network and its variants for text analysis
* To augment data using generative models
* To explore real world applications with deep neural network

**LIST OF THE EXPERIMENTS**

|  |
| --- |
| 1. Solving XOR problem using Multilayer perceptron 2. Implement character and Digit Recognition using ANN. 3. Implement the analysis of X-ray image using autoencoders 4. Implement Speech Recognition using NLP 5. Develop a code to design object detection and classification for traffic analysis using CNN 6. Implement online fraud detection of share market data using any one of the data analytics tools. 7. Implement image augmentation using deep RBM. 8. Implement Sentiment Analysis using LSTM. 9. Mini Project: Number plate recognition of traffic video analysis.   **OUTCOMES:**  CO1: Apply deep neural network for simple problems  CO2: Apply Convolution Neural Network for image processing  CO3: Apply Recurrent Neural Network and its variants for text analysis CO4: Apply generative models for data augmentation  CO5: Develop a real world application using suitable deep neural networks |
| **SOFTWARE REQUIRED:**   * Understanding on Working of Colab and Transfer Learning Networks * High end GPU Systems ( Huge Computation) |

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **S.NO.** | **TITLE OF THE EXPERIMENTS** | **PAGE NO.** |
| 1. | Solving XOR problem using Multilayer perceptron | 1 |
| 2. | Implement character and Digit Recognition using ANN | 3 |
| 3 A. | Implement the analysis of Handwritten Images using Autoencoders | 6 |
| 3 B. | Implement the analysis of Medical X-Ray image classification using CNN | 9 |
| 4. | Implement speech Recognition using NLP | 12 |
| 5. | Develop a code to design Object Detection and Classification for Traffic  analysis using CNN | 17 |
| 6. | Implement online Fraud Detection of share market data using Data  Analytics Tools. | 21 |
| 7 A. | Implement Image Augmentation using Tensor Flow | 25 |
| 7 B. | Implement RBM Modeling to understand Hand Written Digits | 28 |
| 8. | Implement Sentiment Analysis Using LSTM | 31 |

## EX.NO: 1 SOLVING XOR PROBLEM USING MULTILAYER PERCEPTRON

**AIM:**

model.

To implement python program for solving XOR problem using Multilayer perceptron

## ALGORITHM:

1. Import the necessary libraries including tensorflow as tf and numpy as np.
2. Define the XOR input data X and the corresponding labels y.
3. Create the MLP model using the Sequential API from TensorFlow.
4. Add two Dense layers to the model.
   1. The first layer has 4 units and uses the sigmoid activation function.
   2. The second layer has 1 unit (output) and also uses the sigmoid activation function.
5. Compile it using the binary cross-entropy loss function and the stochastic gradient descent (SGD) optimizer with a learning rate of 0.1.
6. Use the trained model to make predictions on the input data X and print the rounded predictions.

## PROGRAM:

from keras.models import Sequential from keras.layers import Dense import numpy as np

import tensorflow as tf tf.random.set\_seed(69)

X\_train = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])

y\_train = np.array([[0], [1], [1], [0]]) model = Sequential([

Dense(4, input\_dim=2, activation='sigmoid'), Dense(1, activation='sigmoid')])

model.compile(loss='binary\_crossentropy',

optimizer='adam', metrics=['accuracy'])

model.fit(X\_train, y\_train, epochs=2000, verbose=0)

loss, accuracy = model.evaluate(X\_train, y\_train, verbose=0) print(f"Loss: {loss:.2f}, Accuracy: {accuracy:.2f}")

X\_test = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])

y\_pred = (model.predict(X\_test, verbose=0) > 0.5).astype("int32") print("Input\tOutput")

for i in range(len(X\_test)): print(f"{X\_test[i]}\t{y\_pred[i][0]}")

|  |  |
| --- | --- |
| **OUTPUT:** |  |
| Input | Output |
| [0 0] | 0 |
| [0 1] | 1 |
| [1 0] | 1 |
| [1 1] | 0 |

## RESULT:

Thus the implementation of python program to solve XOR problem using CNN was successfully implemented and executed.

# EX.NO: 2 IMPLEMENT CHARACTER AND DIGIT RECOGNITION USING ANN

## AIM:

model.

To implement the python program for character and digit recognition using ANN

## ALGORITHM:

1. Import the necessary libraries, including Tensor Flow and its components.
2. Load the dataset, which contains images of handwritten digits along with their corresponding labels.
3. After loading, preprocess the data by scaling the pixel values to a range of 0 to 1.
4. Create ANN model using sequential API from Keras.
5. Compile the model with Adam optimizer, sparse categorical cross-entropy loss function, and accuracy as the metric to monitor.
6. Train the model using the training data.
7. Evaluate the model’s performance on the test data and print the test accuracy.
8. Finally perform predictions on the first five test images and compare the predicted labels with the actual labels.

## PROGRAM:

**# Import Package**

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras.datasets import mnist from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Flatten

## # Load the MNIST dataset

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data() # Preprocess the data

x\_train = x\_train / 255.0 x\_test = x\_test / 255.0

## # Create the ANN model

model = Sequential() model.add(Flatten(input\_shape=(28, 28))) model.add(Dense(128, activation='relu')) model.add(Dense(10, activation='softmax'))

**# Compile the model** model.compile(optimizer='adam',loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

## # Train the model

model.fit(x\_train, y\_train, epochs=5, batch\_size=32, verbose=1)

## # Evaluate the model

test\_loss, test\_acc = model.evaluate(x\_test, y\_test, verbose=0) print(f'Test accuracy: {test\_acc}')

## # Perform predictions

predictions = model.predict(x\_test[:5])

predicted\_labels = [tf.argmax(prediction).numpy() for prediction in predictions] print(f'Predicted labels: {predicted\_labels}')

print(f'Actual labels: {y\_test[:5]}')

## OUTPUT:

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz> 11490434/11490434 [================] - 0s 0us/step

Epoch 1/5

1875/1875 [========================] - 9s 3ms/step - loss: 1.1438 - accuracy: 0.7335

Epoch 2/5

1875/1875 [========================] - 5s 3ms/step - loss: 0.4745 - accuracy: 0.8759

Epoch 3/5

1875/1875 [========================] - 5s 3ms/step - loss: 0.3777 - accuracy: 0.8958

Epoch 4/5

1875/1875 [========================] - 5s 3ms/step - loss: 0.3396 - accuracy: 0.9043

Epoch 5/5

1875/1875 [========================] - 5s 3ms/step - loss: 0.3180 - accuracy: 0.9092 [8]:

<keras.callbacks.History at 0x79c5e07dc790> Test accuracy: 0.9135000109672546

1/1 [==============================] - 0s 20ms/step

Predicted labels: [7, 2, 1, 0, 4]

Actual labels: [7 2 1 0 4]

## RESULT:

Thus the python program for character and digit recognition using ANN model was implemented and executed successfully.

## EX.NO: 3.A) IMPLEMENT THE ANALYSIS OF HANDWRITTEN IMAGES USING AUTOENCODERS

**AIM:**

To implement python program to analysis images using auto encoders.

## ALGORITHM:

* 1. Import the necessary libraries, including Tensor Flow and its components.
  2. Load the dataset, which contains images of handwritten along with their corresponding labels.
  3. After loading, normalize the pixel values between 0 and 1.
  4. Define an auto encoder model using Tensor flow’s Keras API.
  5. Compile the auto encoder model with the Adam optimizer and binary cross-entropy loss function.
  6. Train the model using training data set.
  7. After training, use the trained auto encoder to reconstruct the images from the test set.
  8. Finally plot original and reconstructed images for visual comparison.

## PROGRAM:

**#Import Packages** import tensorflow as tf import numpy as np

import matplotlib.pyplot as plt

## # Load the MNIST dataset

(x\_train, \_), (x\_test, \_) = tf.keras.datasets.mnist.load\_data()

## # Normalize the pixel values between 0 and 1

x\_train = x\_train.astype('float32') / 255. x\_test = x\_test.astype('float32') / 255.

## # Reshape the input images

x\_train = x\_train.reshape((len(x\_train), np.prod(x\_train.shape[1:]))) x\_test = x\_test.reshape((len(x\_test), np.prod(x\_test.shape[1:])))

**# Define the autoencoder model** input\_dim = x\_train.shape[1] encoding\_dim = 32

input\_img = tf.keras.Input(shape=(input\_dim,))

encoded = tf.keras.layers.Dense(encoding\_dim, activation='relu')(input\_img) decoded = tf.keras.layers.Dense(input\_dim, activation='sigmoid')(encoded) autoencoder = tf.keras.Model(input\_img, decoded)

## # Compile the autoencoder model

autoencoder.compile(optimizer='adam', loss='binary\_crossentropy')

## # Train the autoencoder

autoencoder.fit(x\_train, x\_train, epochs=50, batch\_size=256, shuffle=True, validation\_data=(x\_test, x\_test))

## # Use the autoencoder to reconstruct the X-ray image

reconstructed\_images = autoencoder.predict(x\_test)

**# Plot the original and reconstructed images** n = 10 # Number of images to display plt.figure(figsize=(20, 4))

for i in range(n):

## # Display original images

ax = plt.subplot(2, n, i + 1) plt.imshow(x\_test[i].reshape(28, 28)) plt.gray() ax.get\_xaxis().set\_visible(False) ax.get\_yaxis().set\_visible(False)

## # Display reconstructed images

ax = plt.subplot(2, n, i + 1 + n) plt.imshow(reconstructed\_images[i].reshape(28, 28)) plt.gray()

ax.get\_xaxis().set\_visible(False) ax.get\_yaxis().set\_visible(False) plt.show()

## OUTPUT:

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz> 11490434/11490434 [============================] - 0s 0us/step

Epoch 1/10

235/235 [==========================] - 2s 6ms/step - loss: 0.0040 - val\_loss: 0.0041

Epoch 2/10

235/235 [==========================] - 1s 5ms/step - loss: 0.0040 - val\_loss: 0.0041

Epoch 3/10

235/235 [==========================] - 1s 5ms/step - loss: 0.0040 - val\_loss: 0.0041

Epoch 4/10

235/235 [==========================] - 1s 5ms/step - loss: 0.0040 - val\_loss: 0.0040

Epoch 5/10

235/235 [==========================] - 1s 6ms/step - loss: 0.0040 - val\_loss: 0.0040

Epoch 6/10

235/235 [==========================] - 1s 5ms/step - loss: 0.0040 - val\_loss: 0.0040

Epoch 7/10

235/235 [==========================] - 1s 5ms/step - loss: 0.0040 - val\_loss: 0.0040

Epoch 8/10

235/235 [==========================] - 1s 5ms/step - loss: 0.0040 - val\_loss: 0.0040

Epoch 9/10

235/235 [==========================] - 1s 5ms/step - loss: 0.0040 - val\_loss: 0.0040

Epoch 10/10

235/235 [==========================] - 1s 5ms/step - loss: 0.0040 - val\_loss: 0.0040

[10]:

<keras.callbacks.History at 0x798a8145f9d0>





## RESULT:

Thus the implementation of python program to analyse images using auto encoders was successfully implemented andexecuted.

## EX. NO: 3)B IMPLEMENT THE ANALYSIS OF MEDICAL X-RAY IMAGE CLASSIFICATION USING CNN

**AIM:**

Model.

To implement python program for classifying medical X-RAY images using CNN

## ALGORITHM:

1. Import all necessary packages including tensorflow as tf and and keras.
2. Rescale the scales array of the original image pixel values to be between [0,1].
3. Transformation techniques are applied randomly to the images, except for the rescale.
4. Load the X-Ray images.
5. Set the path of test, train and valid folder.
6. Set the target size of the image, Each image will be resized to this size.
7. Number of images to be generated by batch from the generator. Define the batch size as 16
8. Set “binary” if only two classes to predict, if not set to “categorical,” both input and output are likely to be the same image, set to “input” in this case.

## PROGRAM:

import matplotlib.pyplot as plt **#For Visualization**

import numpy as np **#For handling arrays**

import pandas as pd **# For handling data**

## #Define Directories for train, test & Validation Set

train\_path ='D://Babisha//train' test\_path ='D://Babisha//test' valid\_path ='D://Babisha//val' batch\_size = 16

## #The dimension of the images we are going to define is 500x500 img\_height = 500

img\_width = 500

from tensorflow.keras.preprocessing.image import ImageDataGenerator

## # Create Image Data Generator for Train Set

image\_gen = ImageDataGenerator(

rescale = 1./255, shear\_range = 0.2,

zoom\_range = 0.2, horizontal\_flip = True, )

test\_data\_gen = ImageDataGenerator(rescale = 1./255) train = image\_gen.flow\_from\_directory(

train\_path, target\_size=(99, 128), color\_mode='grayscale', class\_mode='binary', batch\_size=batch\_size)

test = test\_data\_gen.flow\_from\_directory( test\_path,

target\_size=(99, 128), color\_mode='grayscale', shuffle=False,

## #setting shuffle as False just so we can later compare it with predicted values without having indexing problem

class\_mode='binary', batch\_size=batch\_size)

valid = test\_data\_gen.flow\_from\_directory( valid\_path,

target\_size=(99,128), color\_mode='grayscale', class\_mode='binary', batch\_size=batch\_size)

plt.figure(figsize=(12, 12)) for i in range(0, 10):

plt.subplot(2, 5, i+1)

for X\_batch, Y\_batch in train: image = X\_batch[0]

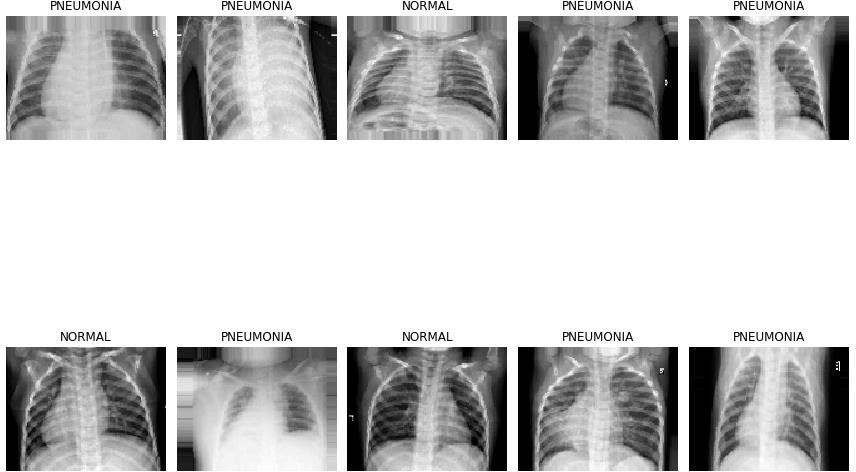
dic = {0:'NORMAL', 1:'PNEUMONIA'}

plt.title(dic.get(Y\_batch[0]))

plt.axis('off') plt.imshow(np.squeeze(image),cmap='gray',interpolation='nearest') break

plt.tight\_layout() plt.show()

## OUTPUT:

Found 4192 images belonging to 2 classes. Found 624 images belonging to 2 classes. Found 1040 images belonging to 2 classes.

## RESULT:

Thus the classification of medical X-RAY images using CNN model was implemented and verified successfully.

## EX.NO: 4 IMPLEMENT SPEECH RECOGNITION USING NLP

**AIM:**

To write a python program to implement speech recognition using NLP.

## ALGORITHM:

* 1. Import the necessary packages for speech recognition.
  2. Load the audio file to perform speech recognition.
  3. Install the needed packages and functions for speech recognizer.
  4. Read three different audio file for converting audio to text.
  5. Using recognize\_google(audio) function convers audio file to text
  6. Print the text file.

## PROGRAM:

#**Install the packages**

!pip install librosa

!pip install SpeechRecognition

!pip install googletrans

!pip install gTTs

!pip install PyAudio

!pip install googletrans==3.1.0a0

!pip install googletrans==4.0.0-rc1

import numpy as np # **linear algebra**

import pandas as pd # **data processing, CSV file I/O (e.g. pd.read\_csv)**

from tqdm import tqdm pd.options.display.max\_colwidth = 200 import warnings warnings.filterwarnings('ignore') import os

for dirname, \_, filenames in os.walk('/kaggle/input'): for filename in filenames[:2]:

print(os.path.join(dirname, filename))

import librosa

import IPython.display as ipd import speech\_recognition as sr from googletrans import Translator from gtts import gTTS

ipd.Audio('/kaggle/input/nlp-specialization-data/harvard.wav') ipd.Audio('/kaggle/input/nlp-specialization-data/testSpeech.wav') ipd.Audio('/kaggle/input/nlp-specialization-data/singleEnglishWord.wav') r = sr.Recognizer()

harvard = sr.AudioFile('/kaggle/input/nlp-specialization-data/testSpeech.wav') with harvard as source:

audio = r.record(source) type(audio) speech\_recognition.AudioData text=r.recognize\_google(audio) text

## Output:

/kaggle/input/nlp-specialization-data/Wikipedia\_Toxicity\_Dataset.csv

/kaggle/input/nlp-specialization-data/Medical\_Notes/Medical\_Notes/1893.txt

/kaggle/input/nlp-specialization-data/Medical\_Notes/Medical\_Notes/1711.txt

/kaggle/input/nlp-specialization-data/pubmed2018\_w2v\_200D/pubmed2018\_w2v\_200D/RE ADME.txt

/kaggle/input/nlp-specialization-data/pubmed2018\_w2v\_200D/pubmed2018\_w2v\_200D/pub med2018\_w2v\_200D.bin

Requirement already satisfied: librosa in /opt/conda/lib/python3.7/site-packages (0.8.1) Requirement already satisfied: numpy>=1.15.0 in /opt/conda/lib/python3.7/site-packages (fro m librosa) (1.19.5)

Requirement already satisfied: scikit-learn!=0.19.0,>=0.14.0 in /opt/conda/lib/python3.7/site- packages (from librosa) (0.23.2)

Requirement already satisfied: pooch>=1.0 in /opt/conda/lib/python3.7/site-packages (from li brosa) (1.4.0)

Requirement already satisfied: scipy>=1.0.0 in /opt/conda/lib/python3.7/site-packages (from l ibrosa) (1.6.3)

Requirement already satisfied: resampy>=0.2.2 in /opt/conda/lib/python3.7/site-packages (fro m librosa) (0.2.2)

Requirement already satisfied: decorator>=3.0.0 in /opt/conda/lib/python3.7/site-packages (fr om librosa) (5.0.9)

Requirement already satisfied: soundfile>=0.10.2 in /opt/conda/lib/python3.7/site-packages (f rom librosa) (0.10.3.post1)

Requirement already satisfied: audioread>=2.0.0 in /opt/conda/lib/python3.7/site-packages (fr om librosa) (2.1.9)

Requirement already satisfied: numba>=0.43.0 in /opt/conda/lib/python3.7/site-packages (fro m librosa) (0.53.1)





speech\_recognition.AudioData result2:

{ 'alternative': [ { 'confidence': 0.78671092,

'transcript': 'Birch canoe slid on the smooth plank ' 'glue the sea to a dark blue '

'background it is easy to tell the ' 'depth of the well these days a ' 'chicken leg of a variegated price is ' "often served in Randall's the juice " 'of lemons makes find the boxes on ' 'the side the pump truck the ha ' 'grimstead top corn and garbage for '

'hours of City Works in a large size ' 'and stockings and hard to sell'},

{ 'transcript': 'Birch canoe slid on the smooth plank ' 'glue the sea to a dark blue '

'background it is easy to tell the ' 'depth of the well these day the ' 'chicken leg of a variegated price is ' "often served in Randall's the juice "

'of lemons makes find the boxes down ' 'beside the pump truck the ha ' 'grimstead top corn and garbage for ' 'hours of City Works in a large size ' 'and stockings and hard to sell'},

{ 'transcript': 'Birch canoe slid on the smooth plank ' 'glue the sea to a dark blue '

'background it is easy to tell the ' 'depth of the well these days a ' 'chicken leg of a variegated price is ' 'often served in roundels the juice '

'of lemons makes find the boxes down ' 'beside the pump truck the ha ' 'grimstead top corn and garbage for ' 'hours of City Works in a large size ' 'and stockings and hard to sell'},

{ 'transcript': 'Birch canoe slid on the smooth plank ' 'glue the sea to a dark blue '

'background it is easy to tell the '

'depth of the well these days a ' 'chicken leg of a variegated price is ' "often served in Randall's the juice " 'of lemons makes find the boxes down ' 'beside the pump truck the ha ' 'grimstead top corn and garbage for ' 'hours of City Works in a large size ' 'and stockings and hard to sell'},

{ 'transcript': 'Birch canoe slid on the smooth plank ' 'glue the sea to a dark blue '

'background it is easy to tell the ' 'depth of the well these days a ' 'chicken leg of a variegated price is ' "often served in Randall's the juice "

'of lemons makes find the boxes down ' 'beside the pump truck the ha ' 'grimstead topcon and garbage for ' 'hours of City Works in a large size ' 'and stockings and hard to sell'}],

'final': True}

"Birch canoe slid on the smooth plank glue the sea to a dark blue background it is easy to tell the depth of the well these days a chicken leg of a variegated price is often served in Randall's the juice of lemons makes find the boxes on the side the pump truck the ha grimstead top corn and garbage for hours of City Works in a large size and stockings and hard to sell"

## RESULT:

Thus the implementation of speech recognition using NLP was successfully implemented and executed.

## EX. NO: 5 DEVELOP A CODE TO DESIGN OBJECT DETECTION AND CLASSIFICATION FOR TRAFFIC ANALYSIS USING CNN

**AIM:**

To develop a code to design object detection and classification for traffic analysis

using CNN model.

## ALGORITHM:

* + 1. Gather a large dataset of labeled images that contain various traffic objects you want to detect and classify.
    2. Preprocess the collected data to ensure consistency and compatibility for training.
    3. Choose a CNN architecture suitable for object detection and classification tasks.
    4. Initialize the chosen CNN model with random weights and train it on your labeled dataset.
    5. During training, the model learns to detect objects and classify them correctly based on the provided annotations.
    6. Split your dataset into training and validation sets. After training, evaluate the performance of the model on the validation set.
    7. Fine-tune the trained model to improve its performance. This can involve adjusting hyperparameters, exploring different network architectures.
    8. Once the model is trained and optimized, it can be used for object detection and classification in new, unseen images.
    9. To refine the detected bounding boxes, post-processing techniques can be applied.
    10. Utilize the detected and classified objects for traffic analysis purposes. This can involve counting the number of vehicles, estimating traffic flow, analyzing vehicle types, tracking vehicle trajectories, or detecting traffic violations.

## PROGRAM:

import os

import matplotlib.pyplot as plt import numpy as np

from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense from keras.models import Sequential

from keras.utils import load\_img, img\_to\_array, to\_categorical from sklearn.model\_selection import train\_test\_split

dataset\_path = 'datasets/5-TRAFFIC-ANALYSIS-CNN' image\_width, image\_height = 64, 64

num\_classes = len(os.listdir(dataset\_path)) X, y = [], []

class\_names = {}

for class\_name in os.listdir(dataset\_path): if class\_name not in class\_names:

class\_names[class\_name] = len(class\_names) class\_path = os.path.join(dataset\_path, class\_name) for image\_name in os.listdir(class\_path):

image\_path = os.path.join(class\_path, image\_name)

image = load\_img(image\_path, target\_size=(image\_width, image\_height)) image = img\_to\_array(image)

X.append(image) y.append(class\_names[class\_name])

X = np.array(X) / 255.0

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, to\_categorical(y, num\_classes=num\_classes), test\_size=0.2)

model = Sequential([

Conv2D(32, (3,3), activation='relu', input\_shape=(image\_width, image\_height, 3)), MaxPooling2D(2,2),

Conv2D(64, (3,3), activation='relu'), MaxPooling2D(2,2),

Conv2D(128, (3,3), activation='relu'), MaxPooling2D(2,2),

Flatten(),

Dense(512, activation='relu'), Dense(num\_classes, activation='sigmoid')])

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy']) model.summary()

model.fit(X\_train, y\_train, epochs=10, batch\_size=32,verbose=0) score, acc = model.evaluate(X\_test, y\_test, verbose=0) print(f"Score: {(score\*100):.2f}% | Accuracy: {(acc\*100):.2f} %") random\_indexes = np.random.choice(len(X\_test), size=9)

reverse\_map = {v:k for k,v in class\_names.items()}

y\_pred = model.predict(X\_test[random\_indexes], verbose=0) y\_pred\_class = [reverse\_map[np.argmax(y)] for y in y\_pred] y\_test = np.argmax(y\_test[random\_indexes], axis=1) y\_test\_class = [reverse\_map[y] for y in y\_test]

fig, ax = plt.subplots(3,3, figsize=(10,10)) for i, index in enumerate(random\_indexes):

ax[i // 3, i % 3].imshow(X\_test[index])

ax[i // 3, i % 3].set\_title(f"Predicted: {y\_pred\_class[i]}\nActual: {y\_test\_class[i]}") ax[i // 3, i % 3].axis('off')

plt.tight\_layout() plt.show()

## OUTPUT:

Model: "sequential"

Layer (type) Output Shape Param #

=================================================================

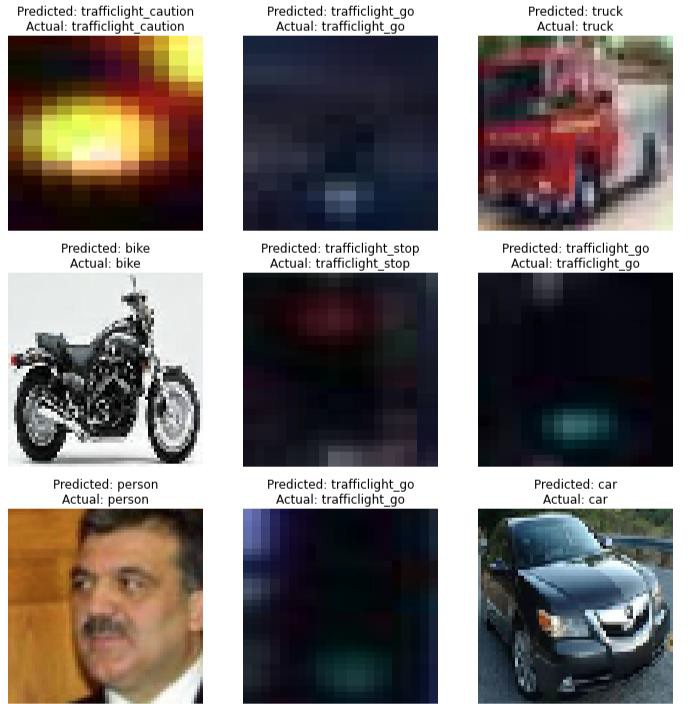
|  |  |  |
| --- | --- | --- |
| conv2d (Conv2D) | (None, 62, 62, 32) | 896 |
| max\_pooling2d (MaxPooling2D) | (None, 31, 31, 32) | 0 |
| conv2d\_1 (Conv2D) | (None, 29, 29, 64) | 18496 |
| max\_pooling2d\_1 (MaxPooling | (None, 14, 14, 64) | 0 |
| 2D) |  |  |
| conv2d\_2 (Conv2D) | (None, 12, 12, 128) | 73856 |
| max\_pooling2d\_2 (MaxPooling | (None, 6, 6, 128) | 0 |
| 2D) |  |  |
| flatten (Flatten) | (None, 4608) | 0 |
| dense (Dense) | (None, 512) | 2359808 |
| dense\_1 (Dense) | (None, 7) | 3591 |

=================================================================

Total params: 2,456,647

Trainable params: 2,456,647

Non-trainable params: 0



## RESULT:

Thus the implementation of python program design object detection and classification for traffic analysis using CNN model was executed and verified successfully.

## EX.NO:6 IMPLEMENT ONLINE FRAUD DETECTION OF SHARE MARKET DATA USING DATA ANALYTICS TOOLS.

**AIM:**

To implement online fraud detection of share market data using data analytics tool.

## ALGORITHM:

1. Import the necessary libraries, including tensorflow as tf, pandas as pd, and numpy as np.
2. Load the stock market data using pd.read\_csv() and extract the relevant features and labels from the data.
3. Normalize the features by subtracting the mean and dividing by the standard deviation using (features - features.mean()) / features.std().
4. Define the MLP model using the Sequential API from TensorFlow.
5. Add three dense layers to the model,
   1. First two layers have ReLU activation.
   2. Last layer uses sigmoid activation for binary classification.
6. Train the model using model.fit() on the normalized features and labels.
7. After training, we predict the fraud probabilities using model.predict() on the normalized features.
8. Add the predictions as a new column named 'FraudProbability' to the original data using data['FraudProbability'] = predictions.
9. Finally, we print the fraudulent activities by filtering the data based on the predicted fraud probabilities using data[data['FraudProbability'] > 0.5].

## PROGRAM

import tensorflow as tf import pandas as pd import numpy as np

## # Load the stock market data

data = pd.read\_csv('D:\Babisha\stock1.csv')

## # Extract the features (e.g., price, volume, etc.) from the data

features = data[['Open','Close']].values

## # Define the labels (fraud or not fraud)

labels = np.array(data['Adj Close'])

## # Normalize the features

normalized\_features = (features - features.mean()) / features.std()

## # Define the MLP model

model = tf.keras.models.Sequential([

tf.keras.layers.Dense(64, activation='relu', input\_shape=(2,)), tf.keras.layers.Dense(32, activation='relu'), tf.keras.layers.Dense(1, activation='sigmoid')

])

## # Compile the model

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

## # Train the model

model.fit(normalized\_features, labels, epochs=10, batch\_size=32, validation\_split=0.2)

## # Predict the fraud probabilities

predictions = model.predict(normalized\_features)

## # Add the predictions as a new column in the data

data['FraudProbability'] = predictions

## # Print the fraudulent activities

fraudulent\_activities = data[data['FraudProbability'] > 0.5] print("Fraudulent Activities:")

print(fraudulent\_activities)

## OUTPUT

Epoch 1/10

43/43 [==============================] - 2s 5ms/step - loss: -86.1644 - accuracy: 0

.0000e+00 - val\_loss: -429.2317 - val\_accuracy: 0.0000e+00 Epoch 2/10

43/43 [==============================] - 0s 1ms/step - loss: -405.5972 - accuracy:

0.0000e+00 - val\_loss: -1608.5574 - val\_accuracy: 0.0000e+00

Epoch 3/10

43/43 [==============================] - 0s 1ms/step - loss: -1365.9751 - accuracy

: 0.0000e+00 - val\_loss: -4702.0205 - val\_accuracy: 0.0000e+00 Epoch 4/10

43/43 [==============================] - 0s 1ms/step - loss: -3519.1863 - accuracy

: 0.0000e+00 - val\_loss: -10974.2188 - val\_accuracy: 0.0000e+00 Epoch 5/10

43/43 [==============================] - 0s 1ms/step - loss: -7582.8442 - accuracy

: 0.0000e+00 - val\_loss: -21873.9551 - val\_accuracy: 0.0000e+00 Epoch 6/10

43/43 [=============================] - 0s 1ms/step - loss: -14148.5811 - accuracy

: 0.0000e+00 - val\_loss: -39032.0312 - val\_accuracy: 0.0000e+00 Epoch 7/10

43/43 [==============================] - 0s 1ms/step - loss: -23938.2207 - accurac

y: 0.0000e+00 - val\_loss: -63014.4492 - val\_accuracy: 0.0000e+00 Epoch 8/10

43/43 [==============================] - 0s 1ms/step - loss: -37390.9531 - accurac

y: 0.0000e+00 - val\_loss: -95703.1016 - val\_accuracy: 0.0000e+00 Epoch 9/10

43/43 [==============================] - 0s 822us/step - loss: -55393.1602 - accur

acy: 0.0000e+00 - val\_loss: -137511.5938 - val\_accuracy: 0.0000e+00 Epoch 10/10

43/43 [==============================] - 0s 1ms/step - loss: -78131.5859 - accurac

y: 0.0000e+00 - val\_loss: -191255.2500 - val\_accuracy: 0.0000e+00 53/53 [==============================] - 0s 538us/step

Fraudulent Activities:

Date Open High Low Close Volume 0 6/29/2010 19.000000 25.000000 17.540001 23.889999 18766300

1 6/30/2010 25.790001 30.420000 23.299999 23.830000 17187100

2 7/1/2010 25.000000 25.920000 20.270000 21.959999 8218800

3 7/2/2010 23.000000 23.100000 18.709999 19.200001 5139800

4 7/6/2010 20.000000 20.000000 15.830000 16.110001 6866900

... ... ... ... ... ... ...

1687 3/13/2017 244.820007 246.850006 242.779999 246.169998 3010700

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1688 | | 3/14/2017 | 246.110001 | 258.119995 | 246.020004 | 258.000000 | 7575500 |
| 1689 | | 3/15/2017 | 257.000000 | 261.000000 | 254.270004 | 255.729996 | 4816600 |
| 1690 | | 3/16/2017 | 262.399994 | 265.750000 | 259.059998 | 262.049988 | 7100400 |
| 1691 | | 3/17/2017 | 264.000000 | 265.329987 | 261.200012 | 261.500000 | 6475900 |
|  | Adj Close | | FraudProbability | | | | |
| 0 | 23.889999 | | 1.0 | | | | |
| 1 | 23.830000 | | 1.0 | | | | |
| 2 | 21.959999 | | 1.0 | | | | |
| 3 | 19.200001 | | 1.0 | | | | |
| 4 | 16.110001 | | 1.0 | | | | |
| ... | ... | | ... | | | | |
| 1687 | | 246.169998 | 1.0 | | | | |
| 1688 | | 258.000000 | 1.0 | | | | |
| 1689 | | 255.729996 | 1.0 | | | | |
| 1690 | | 262.049988 | 1.0 | | | | |
| 1691 | | 261.500000 | 1.0 | | | | |

## RESULT:

Thus the implementation of online fraud detection of share market data using data analytics tool was implemented and verified successfully.

## EX.NO: 7A IMPLEMENT IMAGE AUGMENTATION USING TENSOR FLOW AIM:

To write a python program to implement image augmentation using Tensor flow.

## ALGORITHM:

1. Import the packages necessary for image augmentation using tensor flow.
2. Load the dataset for training the data.
3. Reshape the input data to 4D (samples, height, width, channels).
4. Convert the pixel values to float and normalize to the range [0, 1].
5. Define the image augmentation parameters.
6. Generate augmented images from the training dataset.
7. Convert the augmented images and labels to NumPy arrays.
8. Display some examples of original and augmented images.

## PROGRAM:

import tensorflow as tf

from tensorflow.keras.datasets import mnist

from tensorflow.keras.preprocessing.image import ImageDataGenerator import matplotlib.pyplot as plt

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data() x\_train = x\_train.reshape(x\_train.shape[0], 28, 28, 1)

x\_test = x\_test.reshape(x\_test.shape[0], 28, 28, 1) x\_train = x\_train.astype('float32') / 255.0

x\_test = x\_test.astype('float32') / 255.0 datagen = ImageDataGenerator(

rotation\_range=10, width\_shift\_range=0.1, height\_shift\_range=0.1, shear\_range=0.2, zoom\_range=0.2, fill\_mode='nearest')

augmented\_images = []

augmented\_labels = []

for x, y in zip(x\_train, y\_train): x = tf.expand\_dims(x, 0)

augmented\_iterator = datagen.flow(x, batch\_size=1) augmented\_image = next(augmented\_iterator)[0] augmented\_images.append(augmented\_image) augmented\_labels.append(y)

augmented\_images = np.array(augmented\_images) augmented\_labels = np.array(augmented\_labels) fig, axes = plt.subplots(2, 5, figsize=(12, 6))

for i, ax in enumerate(axes.flat): if i < 5:

ax.imshow(x\_train[i].reshape(28, 28), cmap='gray') ax.set\_title('Original')

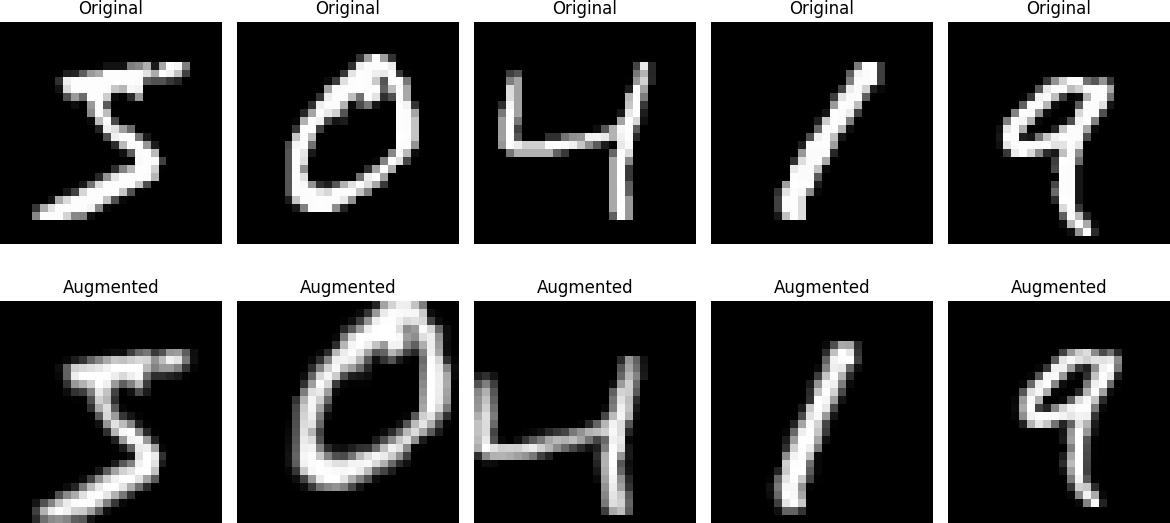
else:

ax.imshow(augmented\_images[i - 5].reshape(28, 28), cmap='gray') ax.set\_title('Augmented')

ax.axis('off') plt.tight\_layout() plt.show()

## OUTPUT:

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz> 11490434/11490434 [==============================] - 0s 0us/step



## RESULT:

Thus the program to implement image augmentation using Tensor flow was successfully implanted and executed.

## EX.NO:7B IMPLEMENT RBM MODELING TO UNDERSTAND HAND WRITTEN DIGITS

**AIM:**

digits.

To write a python program to implement RBM modeling to understand hand written

## ALGORITHM:

1. Import the necessary packages needed for RBM modeling.
2. Load the input file for processing.
3. Get the pixel value of the image.
4. Read the handwritten digits for min-max scaling to convert the image pixel from 0 to 255 range 0 to 1.
5. Import BernoulliRBM package and execute the RBM modeling for 10 iteration over the training data set.
6. Perform Gibbs Sampling over the dataset,
7. Print the handwritten images produced by RBM model.

## PROGRAM:

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv) import os

for dirname, \_, filenames in os.walk('/kaggle/input'): for filename in filenames:

print(os.path.join(dirname, filename)) import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline plt.rcParams['image.cmap'] = 'gray' import pandas as pd

def gen\_mnist\_image(X):

return np.rollaxis(np.rollaxis(X[0:200].reshape(20, -1, 28, 28), 0, 2), 1, 3).reshape(-1, 20 \*

28)

X\_train = pd.read\_csv('../input/mnist-digit-recognizer/train.csv').values[:,1:]

X\_train = (X\_train - np.min(X\_train, 0)) / (np.max(X\_train, 0) + 0.0001) # 0-1 scaling

plt.figure(figsize=(10,20)) plt.imshow(gen\_mnist\_image(X\_train));

from sklearn.neural\_network import BernoulliRBM

rbm = BernoulliRBM(n\_components=100, learning\_rate=0.01, random\_state=0, verbose=True)

rbm.fit(X\_train)

xx = X\_train[:40].copy() for ii in range(1000):

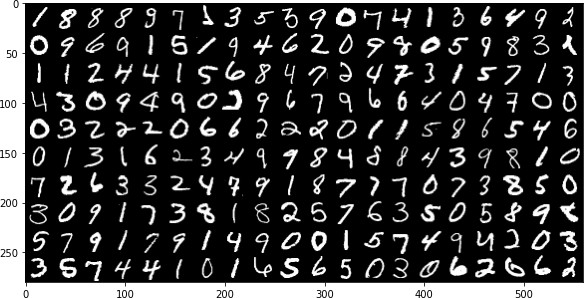
for n in range(40):

xx[n] = rbm.gibbs(xx[n]) plt.figure(figsize=(10,20)) plt.imshow(gen\_mnist\_image(xx)) xx = X\_train[:40].copy()

for ii in range(10000): for n in range(40):

xx[n] = rbm.gibbs(xx[n]) plt.figure(figsize=(10,20)) plt.imshow(gen\_mnist\_image(xx))

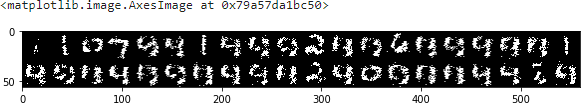
## OUTPUT:

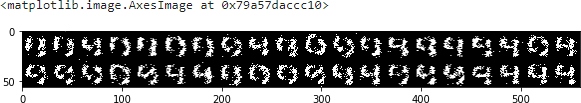
/kaggle/input/mnist-digit-recognizer/train.csv

[BernoulliRBM] Iteration 1, pseudo-likelihood = -119.88, time = 9.48s [BernoulliRBM] Iteration 2, pseudo-likelihood = -104.69, time = 11.60s

[BernoulliRBM] Iteration 3, pseudo-likelihood = -97.89, time = 11.29s [BernoulliRBM] Iteration 4, pseudo-likelihood = -93.03, time = 10.58s [BernoulliRBM] Iteration 5, pseudo-likelihood = -90.10, time = 10.71s [BernoulliRBM] Iteration 6, pseudo-likelihood = -88.20, time = 11.41s [BernoulliRBM] Iteration 7, pseudo-likelihood = -86.16, time = 10.47s [BernoulliRBM] Iteration 8, pseudo-likelihood = -85.37, time = 10.69s [BernoulliRBM] Iteration 9, pseudo-likelihood = -83.71, time = 11.40s [BernoulliRBM] Iteration 10, pseudo-likelihood = -82.54, time = 10.53s

BernoulliRBM(learning\_rate=0.01, n\_components=100, random\_state=0, verbose=True)





## RESULT:

Thus the implementation of RBM modeling to understand hand written digits was successfully implemented and executed.

## EX.NO: 8 IMPLEMENT SENTIMENT ANALYSIS USING LSTM

**AIM:**

To write a python program to implement Sentiment analysis using LSTM

## ALGORITHM:

1. Import the necessary packages needed for sentiment analysis.
2. Load the input file for processing.
3. Convert each word in your text data into a dense vector representation called word embeddings.
4. Split the dataset into training and testing sets. The training set will be used to train the LSTM model,
5. Pad the input sequences with zeros to make them of equal length. This is necessary because LSTM models require fixed-length input sequences.
6. Construct an LSTM model using a deep learning framework like TensorFlow or PyTorch.
7. Evaluate the performance of your trained LSTM model using the testing set.
8. Calculate metrics like accuracy, precision, recall, and F1-score to measure how well the model predicts sentiment.

## PROGRAM:

from keras.datasets import imdb

(x\_train, y\_train), (x\_test, y\_test) = imdb.load\_data(num\_words=10000) from keras.utils import pad\_sequences

max\_len = 100

x\_train = pad\_sequences(x\_train, maxlen=max\_len) x\_test = pad\_sequences(x\_test, maxlen=max\_len) from keras.models import Sequential

from keras.layers import Embedding, LSTM, Dense model = Sequential([

Embedding(10000, 192),

LSTM(128, dropout=0.2, recurrent\_dropout=0.2), Dense(1, activation='sigmoid')])

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy']) model.fit(x\_train, y\_train, batch\_size=32, epochs=5, verbose=0)

score, acc = model.evaluate(x\_test, y\_test, batch\_size=32, verbose=0) print(f"Test score: {score:.2f} - Test accuracy: {acc:.2f}") word\_index = imdb.get\_word\_index()

reverse\_word\_index = {value: key for (key, value) in word\_index.items()} def decode\_review(review):

return ' '.join([reverse\_word\_index.get(i - 3, '?') for i in review]) import numpy as np

random\_indices = np.random.randint(0, len(x\_test), 5) x\_sample = x\_test[random\_indices]

y\_sample = y\_test[random\_indices]

y\_pred = model.predict(x\_sample, batch\_size=5, verbose=0) for i in range(len(x\_sample)):

print(f"Review: {decode\_review(x\_sample[i])[0:30]}...") print(f"Sentiment: {'Positive' if y\_sample[i] == 1 else 'Negative'}") print(f"Predicted sentiment: {'Positive' if y\_pred[i] > 0.5 else 'Negative'}") confidence = y\_pred[i] if y\_pred[i] > 0.5 else 1 - y\_pred[i] print(f"Confidence: {confidence[0]:.2f}")

print('-'\*50)

## OUTPUT:

Review: was terrible and the last 25 3... Sentiment: Negative

Predicted sentiment: Negative Confidence: 1.00

Review: his daughters through a pun in... Sentiment: Negative

Predicted sentiment: Negative Confidence: 1.00

Review: the book too and i am being un... Sentiment: Negative

Predicted sentiment: Negative Confidence: 1.00

Review: huge ? and disregard the envir... Sentiment: Positive

Predicted sentiment: Positive Confidence: 0.79

Review: the japanese in korea in medie... Sentiment: Positive

Predicted sentiment: Positive Confidence: 0.97

## RESULT:

Thus the implement of python program for Sentiment analysis using LSTM was executed and verified successfully.