Ex. No:	1	SOLVING XOR PROBLEM USING MULTILAYER PERCEPTRON
Date:	28/11/24	

AIM

To demonstrate the ability of a multilayer perceptron to solve the XOR problem, highlighting the power of deep architectures in handling non-linear relationships.

ALGORITHM

Step 1: Prepare a dataset with input features (0 or 1) and corresponding XOR outputs.
Step 2: Design a multilayer perceptron with input, hidden, and output layers.
Step 3: Randomly initialize the network's weights and biases.
Step 4: Choose a loss function (e.g., mean squared error) and an optimization algorithm (e.g., stochastic gradient descent).
Step 5: Train the network using the dataset, adjusting weights and biases iteratively.
Step 6: Evaluate the trained network's performance on XOR inputs and compare with expected outputs.

PROGRAM

```
Import numpy as np
from keras. models import sequential
from keras. layers import dense
x- train = np. array ([0,0],[0,1], [1,0],[1,1])
y- train = np. array ( [0,0], [1],[1],[0])
model = sequential ()
model. add ( dence (8, input - dim =2,
                activation ='relu'))
```

```
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy',
        optimizer='adam', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=1500, batch_size=4)
loss, accuracy = model.evaluate(x_train, y_train)
print(f"Loss: {loss}, Accuracy: {accuracy}")
x_test = np.array([[0,0],[0,1],[1,0],[1,1]])
predictions = model.predict(x_test)
print(predictions)
for i in range(len(x_test)):
    print(f"Input: {x_test[i]}, predicted
        output: {predictions[i]}")
```

OUTPUT:

```
Epoch 1/1500
1/1 [==============================] - 0s 360 ms/step -
    loss: 0.6985 - accuracy: 0.7500
Epoch 2/1500
1/1 [==============================] - 0s 10ms/step -
    loss: 0.6977 - accuracy: 0.5000
Epoch 3/1500
1/1 [==============================] - 0s 0s/step -
    loss: 0.6969 - accuracy: 0.5000
[[[0.20657766]
  [0.951823]
  [0.88884634]
  [0.02532875]]]
```

Input : [0,0] , predicted output : [0.20657766]
Input : [0,1] , predicted output : [0.951823]
Input : [1,0] , predicted output : [0.88884634]
Input : [1,1] , predicted output : [0.02532875]

INFERENCE

From the above experiment. I have learnt about solving XOR problem using DNN.

VIVA VOICE

1) What Is a Multi-layer Perceptron(MLP)?

A multilayer perceptron is a type of artificial neural network consisting of multiple layers of neurons. These neurons typically use non linear activation function.

2) What is Deep Learning?

It is a subset of machine learning that uses multi layered neural network called deep neural network.

3) What is a Neural Network?

It is a type of ml process called deep learning that uses interconnected nodes or neurons in layered structure

4) What Do You Understand by Backpropagation?

Backpropagation or Backward propagation of errors is an algorithm that is designed to lsl of errors.

5) What Is the Role of Activation Functions in a Neural Network?

It decide whether a neuron should be activated or not by calculating weighted sum and further adding bias to it.

V.Pal 20/11/21

RESULT

Thus, the program for solving XOR problem using Multilayer perceptron has been executed and verified successfully.

Ex.No:	2	IMPLEMENTATION OF IMAGE CLASSIFICATION USING CNN.
Date:	30/1/24	

AIM

To execute Face Recognition using CNN in Python program.

ALGORITHM

Step 1: Import necessary libraries.
Step 2: Load the LFW dataset.
Step 3: Extract images and labels.
Step 4: Create the CNN model:
- Initialize a Sequential model.
- Add a 2D convolutional layer (`Conv2D`) with 32 filters, a 3x3 kernel, and ReLU activation.
- Add a max-pooling layer (`MaxPooling2D`) with a 2x2 pool size.
- Add another 2D convolutional layer with 64 filters and ReLU activation.
- Add another max-pooling layer.
- Flatten the output.
- Add a dense layer (`Dense`) with 128 units and ReLU activation.

Step 5: Compile and train the model.
Step 6: End of the algorithm.

PROGRAM

```
from sklearn.datasets import fetch_lfw_people
from sklearn.model_selection import train_test_split
from keras.models import sequential
from keras.layers import conv2D maxpooling 2D,
flatten, Dense from keras.utils
```

```
import to categorical
import numpy as np
import matplotlib.pyplot as plt.
lfw = datasets = fetch_lfw_people (min_faces_per son = 70,
  images = lfw_datasets_images.
  labels = lfw_dataset .target
  target_names = lfw = datasets . target_names
   images = images. astype ('float 32')/255
   labels = to_categorical (labels)
train_images , test_images , train_labels , test_labels =
 train_test_split (images . labels, test_size =0.2)
model = sequential ()
model . add (conv2D/32/ 3,3) activation = 'relu')
model . compile ( optimizer = 'rmsprop', loss ='categorical-
crossentropy ; metrics [accuracy]
model . fit (train_images , train label . batch_size 3.2
  epochs = 3)
 sample_index = 0
 sample_image = test_image [ sample_index]
 sample_label = np. argmax ( test_labels [ sample_index])
 sample_images = np. expand_dims ( sample_images,
  axis =0)
```

```
model.predict(sample_image)
plt.imshow(sample_image[0], cmap='gray')
plt.title(f'predicted {predicted_name}\n True: {sample_name}')
plt.axis('off')
plt.show()
```

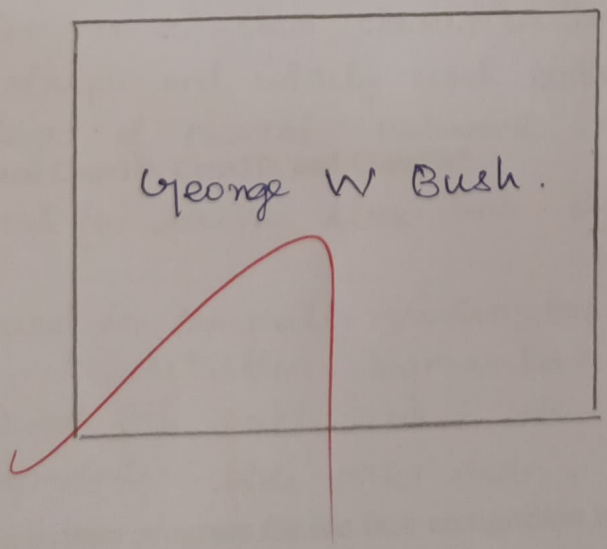
OUTPUT

Epoch 1/10
33/33 [= = = = =] - 3s 74 ms/step
loss : 1.7440 - accuracy 0.4019.

Epoch 2/10
33/33 [= = = = =] - 2s 69 ms/s step
loss : 1.6888 accuracy : 0.4136.

Epoch 3/10
33/33 [= = = = =] - 2s 68 ms/s step
loss : 1.6953 accuracy : 0.4136

Predicted : George W Bush
True : George W Bush.



INFERENCE

From the above program, I learnt about the implementation of image classification using CNN.

VIVA VOICE:

1) What Are the Different Layers on CNN?

* Input layer
* pooling layer
* convolutional layer
* fully connected layer
* Activation layer
* Dense layer.

2) What is Pooling on CNN, and How Does It Work?

It is technique used in convolutional neural network to reduce the spatial dimensions of the feature maps generated by convolutional layer.

3) What is Overfitting and Underfitting, and How to Combat Them?

overfitting: occurs when a ml model learns the training data too well capturing noises or problem fluctation

underfitting: occurs when a model is too simple to capture the underlying structure of the data.

4) What Are the Softmax and ReLU Functions?

Softmax: often used in the output layer of neural networks for multi-class classification task.

ReLU: simple and widely used activation function is hidden layer of neural network.

5) Difference between Conv1D, Conv2D, and Conv3D?

conv1D: used for tasks like text classification, audio processing:

Conv2D: used in computer vision tasks for tasks like object detection, segmentation.

Conv3D: used for tasks when the data has a temporal components like MRI scan.

RESULT

Thus, the python program for the face recognition is implemented and executed successfully and the output is verified.

Ex.No:	3	BUILDING A DEEP LEARNING MODEL
Date:	6/2/24.	

AIM

To write a program to build a simple feedforward neural network (FNN) for binary classification using TensorFlow and Keras.

ALGORITHM

Step 1: Load and preprocess the dataset.
Step 2: Specify the number of input features, hidden layers, neurons in each hidden layer, activation functions, and output layer.
Step 3: Compile the model by specifying the optimizer, loss function, and evaluation metrics.
Step 4: Specify the number of epochs and batch size.
Step 5: Compute metrics such as accuracy, precision, recall, and F1-score.

PROGRAM

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import sequential
from tensorflow.keras.layers import Dense
np.random.seed(0)
x = np.random.randn(1000,10)
y = np.random.randint(2, size=1000)
x_train, x_test = x[:800], x[800:]
y_train, y_test = y[:800], y[800:]
input_domain = x_train.shape[1]
```

```
hidden_layers = 2
reasons_per_layer = 'relu'
output_dim = 1
model = Sequential()
model.add(Dense(neurons_per_layer, input_domain =
        input_dim, activation = activation_function)
for in range(hidden_layer - 1):
    model.add(Dense/neurons_per_layer, activation =
            activation_function))
model.add(Dense(output_dim, activation = 'sigmoid'))
model.compile(optimizer='adam', loss = "binary_
    crossentropy". metrics ['accuracy'])
epochs = 3
batch_size = 32
model.fit(X_train, y_train, epochs = epochs, batch_
    size = batch_size, verbon = 1) loss, accuracy =
            model.evaluate(x_test, y_test)
print("Test loss: ", loss)
print("Test Accuracy: ", accuracy)
```

OUTPUT

Epoch 1/3

25/25 [==========] -0s 1ms/step - loss: 0.7059

accuracy 0.5085

Epoch 2/3

25/25 [==========] -0s 730ms/step loss: 0.6891

accuracy : 0.5337.

Epoch 3/3

27/25 [==========] -0s 1ms/step loss: 0.6799

accuracy : 0.5863

7/7 [==========] -0s 0s/step loss : 0.7111

accuracy : 0.5050

Test loss: 0.7110827651931 76

Test accuracy: 0.50499999 52316284.

INFERENCE:

From the above program, I learnt to build a deep learning model with FNN for binary classification.

VIVA VOCE:

1) What are the disadvantages of deep learning?

Disadvantage is the ability to execute feature engineering by itself. In this approach, an algorithm scans the data.

2) What is a deep learning model, and how does it differ from traditional machine learning models?

It is the subset of ML that uses artificial neural network to mimic learning process of human brain.

3) Can you outline the general workflow for building a deep learning model?

We can train our model on the training dataset and we evaluate it on validation dataset.

4) What are some common architectures used in deep learning models, such as feedforward neural networks, convolutional neural networks (CNNs), and recurrent neural networks (RNNs)?

5) What Are the Applications of a Recurrent Neural Network (RNN)?

RNNs are used in application like Natural Language processing speech Recognition, Time Series prediction and remenu generator.

N.R. 13/24

RESULT

Thus the program for Building a deep learning network was executed successfully and verified.

Ex. No: 4 Date: 13/2/24.	IMPLEMENT THE ANALYSIS OF X-RAY IMAGE USING AUTOENCODERS.

AIM:

To apply autoencoders to X-ray images for feature extraction and anomaly detection, illustrating the potential of unsupervised learning in medical image analysis.

ALGORITHM:

Step 1: Gather a dataset of X-ray images for analysis.
Step 2: Design an autoencoder architecture with an encoder and a decoder.
Step 3: Define a suitable loss function, often using mean squared error..
Step 4: Train the autoencoder using X-ray images to learn compact representations.
Step 5: Evaluate the reconstruction quality and encoded representations
Step 6: Use encoded representations for tasks like anomaly detection or image demising.

PROGRAM

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Input, Dense.
from tensorflow.keras.models import model
import matplotlib.pyplot as plt
X-ray-image = np.random.rand (256,256)
noisy-X-ray = x ray-image+ np.random.normal(0.0.1, x-ray-
                                        image-shape)
x-ray-image = x.ray-image (255.0, noisy-x.ray = noisy-x-ray
                                        (255.0)
input-layer = Input (shape.(256,256))
encoded = Dense (128, activation='relu') (input-layer)
```

```
decoded = Dense(256, activation='sigmoid')(encoded)
autoencoder = model(input_layer, decoded)
autoencoder.compile(optimizer='adam', loss='mean_squared
                                        _error')
denoised_x_ray = autoencoder.predict(np.expand_dims
                    (noisy_x_ray, axis=0))
plt.figure(figsize=(10,5))
plt.subplot(1,3,1)
plt.imshow(x_ray_image, cmap='gray')
plt.title('original x_ray')
plt.subplot(1,3,2)
plt.imshow(noisy_x_ray, cmap='gray')
plt.title("Noisy x_ray")
plt.subplot(1,3,3)
plt.imshow(denoised_x_ray[0], cmap='gray')
plt.title("Denoised x_ray")
plt.tight_layout()
plt.show()
```

OUTPUT

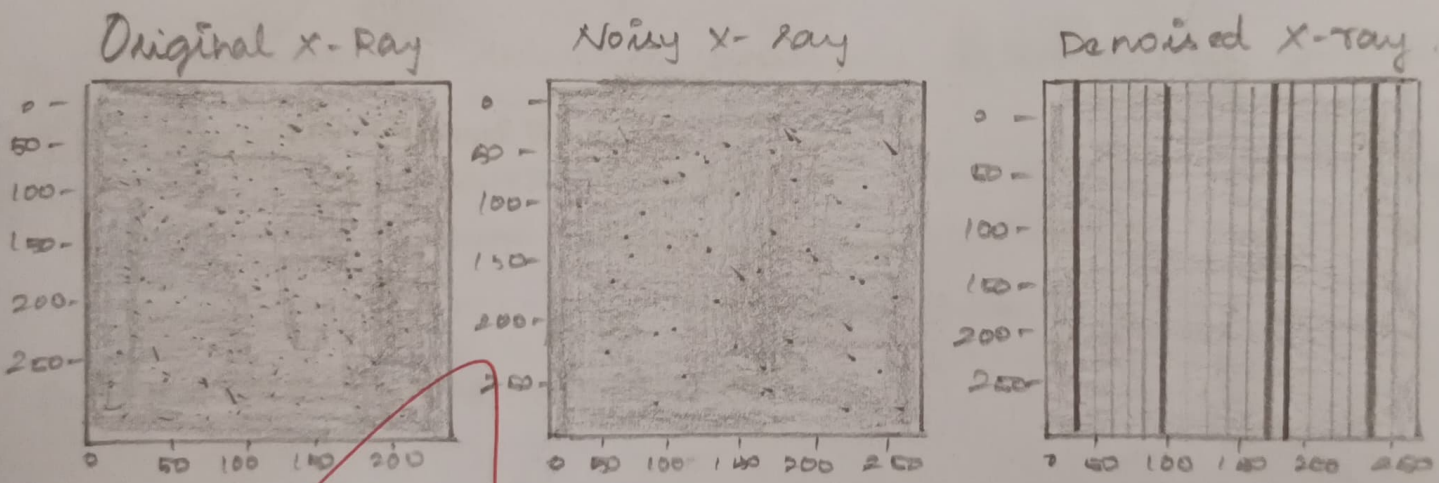
Epoch 1/2.

8/8 [==========] - 0s 0s/step - loss: 0.2468

Epoch 2/2

8/8 [==========] 0s 0s/step - loss - 0.2427

1/1 [==========] - 0s 69 ms/step



INFERENCE:

From the above program. I have learnt about analysis of x-ray image using autoencoders.

VIVA VOCE:

1) What Is an Auto-encoder?

An autoencoder is a type of neural network used for unsupervised learning that learns to encode input data into a lower dimensional representation.

1) What are some of the uses of Autoencoders in Deep Learning?

Used in DL for tasks like data denoising, dimensionality reduction, anamoly detection and generate modelling.

2) How Does an LSTM Network Work?

LSTM networks are type of recurrent neural network designed to overcome the vanishing gradiant problem by maintaining a memory over long sequence

3) What are the deep learning frameworks or tools?

pytorch,
Keras.
caffe, mornet

4) How does an autoencoder differ from other types of neural networks?

Unlike other neural network autoencoders are trained to reconstruct input data rather than predicting a target output.

RESULT

Thus the program to implement the analysis of X-ray image using autoencoders was executed successfully & verified.

Ex. No: 5 Date:	IMPLEMENT SPEECH RECOGNITION USING NLP.

AIM

To create a speech recognition system using natural language processing (NLP) techniques, demonstrating the fusion of audio data and language understanding in enabling voice-controlled applications.

ALGORITHM

Step 1: Collect a dataset of spoken audio samples and their corresponding transcripts.

Step 2: Preprocess the audio data by converting it into spectrograms or other suitable representations.

Step 3: Design a deep learning model, such as a recurrent neural network (RNN) or a transformer, for speech recognition.

Step 4: Implement a suitable loss function, like connectionist temporal classification (CTC) loss.

Step 5: Train the model on the audio-transcript pairs.

Step 6: Evaluate the model's performance by measuring word error rate or other relevant metrics.

PROGRAM

```
import numpy as np
import tensorflow as tf
from keras. layer in port input dense
from tensorflow. Keras models import model
import matplotlib as plt.
Sample_audio_features = np. random. rand (100, 100, 20)
input_layers = Input ( shape = (100, 20))
embedding = Embedding (input_dim = 10 000, output_dim = 128)
                                    ( Input_layer)
```

```
embedding_reshaped = tf.keras.layers.Reshape((1,-1,128))
                                    (embedding)

lstm = LSTM(128)(embedding_reshaped)

output_layers = Dense(10, activation='softmax')(lstm)

speech_recognition_model = model(inputs=input_layer,
                                 outputs=output_layer)

speech_recognition_model.compile(optimizer='adam',
                                 loss='categorical_crossentropy', metrics=['accuracy'])

sample_labels = np.random.randint(10, size=(100,10))

speech_recognition_model.fit(sample_audio_features,
                             sample_labels, epochs=10, batch_size=32)

sample_input_audio = np.random.rand(1,100,20)

predicted_probs = speech_recognition_model.predict
                                    (sample_input_audio)

predicted_word_index = np.argmax(predicted_probs)

print("Predicted Word Index:", predicted_word_index)
```

OUTPUT

Predicted Word Index : 4.

INFERENCE

From the above program, I learnt the implementation of speech Recognition using NLP.

VIVA VOICE:

1) What is speech recognition, and how does it differ from traditional NLP tasks?

It is the process of converting spoken words into text. It differs from the traditional NLP tasks in that it deals with audio input rather than text.

2) What are the main challenges in speech recognition, and how can NLP help address them?

Challenges: Noise, accents, variation in speech. NLP provides language understanding, context analysis and error correction.

3) What are acoustic models and language models in the context of speech recognition?

Acoustic models analyze audio signals to recognize speech sounds with language models assign probabilities to sequence of words.

4) How do language models handle out-of-vocabulary words in speech recognition?

language models handle out of vocabulary words by using techniques like subword units or incoorpating external knowledge.

5) What are some real-world applications of speech recognition using NLP

Virtual assitant
Voice controlled device
speech-to-text
Automated customer service.

VP 27/11/24

RESULT

Thus the program for implement of speech recognition using NLP has been executed & verified successfully.

Ex. No : 6 Date: 27/2/24.	DEVELOP A CODE TO DESIGN OBJECT DETECTION AND CLASSIFICATION FOR TRAFFIC ANALYSIS USING CNN

AIM

To construct a convolutional neural network (CNN) that can simultaneously detect and classify objects in traffic images, highlighting the role of CNNs in complex tasks like traffic analysis and surveillance.

ALGOGITHM

Step 1: Assemble a dataset of traffic images with object annotations and labels.
Step 2: Design a convolutional neural network (CNN) architecture for object detection and classification.
Step 3: Implement non-maximum suppression for eliminating duplicate object detections.
Step 4: Choose appropriate loss functions for object detection and classification, such as region proposal network (RPN) loss and categorical cross-entropy.
Step 5: Train the CNN on the dataset and fine-tune the model.
Step 6: Evaluate the model's performance in terms of object detection accuracy and classification accuracy.

PROGRAM

```
import numpy as np
import tensorflow as tf
from tensorflow.Keras.layers import Conv2D, Maxpooling 2D,
        Flatten, Dense.
from tensorflow.Keras.layers import sequential
sample_images = np.random.rand (100, 64, 64, 3)
sample_labels = np.random.randint (2, size = 100)
model = sequential ([ Conv2D (32, (3,3), activation = 'relu', input_shape=
        (64,64,3)), Maxpooling ((2,2)), Conv2D (64, (3,3)),
        activation = 'relu'], maxpooling 2D((2,2)), Conv2D(128, (3,3))
        activation = 'relu'], maxpooling 2D((2,2)), Flatten(), Dense
```

```
(128, activation='relu'), Dense, (1, activation='sigmoid')])
model.compile(optimizer='adam', loss='binary_crossentropy',
    metrics=['accuracy'])
model.fit(sample_images, sample_labels, epochs=10,
    batch_size=32)
sample_traffic_image = np.random.rand(1,64,64,3)
Prediction = model.predict(sample_traffic_image)
print("Predicted Probability:", prediction).
```

OUTPUT

Predicted probability : [[0. 49626318]]

INFERENCE

From the above program I have learnt object detection and classification for traffic analysis.

VIVA VOCE:

1) **What is traffic analysis, and why is it important in the context of transportation systems?**

Traffic analysis involves studying patterns, behaviours and trends in transportation system. crucial for optimizing traffic flow impaining safety.

2) **How can CNNs be used for traffic analysis and classification tasks?**

CNN can analyze traffic by processing images or video of traffic scenes to identify and classify vechile pedestrains and other object.

3) **What are the main steps involved in training a CNN for traffic classification?**

The main steps in training a CNN for traffic classification involve data collection, preprocessing model architecture design, training with labelled data validation and fine tuning.

4) **What are the advantages of using CNNs over traditional machine learning methods for traffic analysis?**

CNNs offers advantages over traditional ML methods for traffic analysis due to their ability to automatically extract hierrarchical features from raw data such as image without manual feature engineering.

5) **How do pooling layers contribute to feature extraction in CNNs for traffic analysis?**

Pooling layers in CNNs contribute to feature extraction by reducing the spatial dimensions of feature maps.

RESULT

Thus the program for Develop Object Detection and Classification for Traffic Analysis using CNN was executed successfully and verified.