Pgm: SOLVING XOR PROBLEM USING DNN

Aim :

To solve the XOR problem using Deep Neural Network (DNN) and demonstrate how a multi-layer perceptron can be used to learn and predict the XOR logic gate output.

Procedure :

1. Define XOR problem
2. Prepare dataset
3. Build DNN model
4. Compile the model using suitable optimizer and loss function
5. Train the model using the XOR dataset
6. Evaluate the model for custom inputs
7. Display the predictions

Program : (DNN using 2-input)

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

# XOR input and output
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y = np.array([[0], [1], [1], [0]])

# Build the model
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(4, input_dim=2, activation="tanh"),  # Use tanh instead of relu
    tf.keras.layers.Dense(1, activation="sigmoid")
])
```

```
# Summary of the model
model.summary()

# Compile the model
model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy"])

# Train the model for more epochs
history = model.fit(X, y, epochs=1000, verbose=0)

# Make predictions
predictions = model.predict(X)

# Print predictions
for i, pred in enumerate(predictions):
    print(f"Input: {X[i]}, Prediction: {pred[0]:.4f}, Class: {int(pred[0] > 0.5)}")

# Plot the training loss
plt.plot(history.history['loss'])
plt.title('Model Loss During Training')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.show()
```

Model: "sequential_11"

Layer (type)	Output Shape	Param #
dense_22 (Dense)	(None, 4)	12
dense_23 (Dense)	(None, 1)	5

Total params: 17 (68.00 B)
Trainable params: 17 (68.00 B)
Non-trainable params: 0 (0.00 B)

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
Input: [0 0], Prediction: 0.1659, Class: 0
Input: [0 1], Prediction: 0.8140, Class: 1
Input: [1 0], Prediction: 0.5816, Class: 1
Input: [1 1], Prediction: 0.4709, Class: 0
```