AIM PROCEDURE FOR GEN AI

1. Applications of Generative Models using Text Generation

✓ AIM

To generate text automatically using a generative model (GPT-2) based on a given prompt.

✓ PROCEDURE

Install the **transformers** library.

Import the **text-generation pipeline**.

Load the **GPT-2 model**.

Give a starting prompt (example: *"Once upon a time in a futuristic city,"*).

Run the model to generate text.

Display the generated output.

✓ RESULT

The GPT-2 model successfully generated a continuation of the given prompt, showing how generative models can create new text automatically.

2. Integrating ChatGPT with Voice Interface (TTS)

✓ AIM

To convert text into speech using the gTTS (Google Text-to-Speech) library in Google Colab.

✓ PROCEDURE

Install the **gTTS** library using
!pip install gTTS

Import **gTTS** and **Audio** from IPython.

Write the text you want to convert into speech.

Use gTTS(text) to convert the text to audio.

Save the audio file as "speech.mp3".

Play the audio in Google Colab using Audio("speech.mp3", autoplay=True).

✓ OUTPUT

A speech audio file (speech.mp3) is generated and played in Google Colab, allowing the text to be heard as spoken voice.

3.Generate Responsive HTML Page from Description

✓ AIM

To create a responsive Contact Form webpage using HTML and CSS.

✓ PROCEDURE

Create an **HTML file** and add structure using <form>, <input>, <textarea>, and <button>.

Add headings and labels for Name, Email, and Message.

Link an external **CSS file** using
<link rel="stylesheet" href="style.css">

In **style.css**, apply styling for:

page background

contact form container

labels, input fields, textarea

submit button

Use **flexbox** in the body to center the form.

Save both files and open the HTML file in a browser to view the responsive contact form.

✓ RESULT

A clean and responsive **Contact Us webpage** is successfully created.
The form is centered, styled with CSS, and adjusts well on different screen sizes.

4.Generate SQL Queries from English Instructions

Aim

To generate SQL queries automatically from English (natural language) instructions using a Generative Language Model.

Procedure

Take a simple English instruction as input (e.g., "Show all students with marks above 80").

Use a language model (like ChatGPT or GPT API) to convert the instruction into an SQL query.

Display the generated SQL query as output.

Test with different natural language inputs.

Result

The model successfully converts natural English sentences into correct SQL queries.

5. Installing chatgpt

Aim

To install and use ChatGPT through the OpenAI Python library and generate a response from the chatbot.

Procedure

Install the OpenAI package using pip install openai.

Import the OpenAI client in Python.

Initialize the client using your API key.

Send a user message to the ChatGPT model.

Receive and display the response generated by ChatGPT.

Result

The program successfully connected to ChatGPT and displayed a friendly reply to the user’s input.

6. Create Chatbot using GPT Model

Aim

To create an interactive chatbot using the GPT model that responds to user inputs in real time.

Procedure

Install the OpenAI Python package.

Import and initialize the OpenAI client with an API key.

Create a loop to accept user input continuously.

Send each input to the GPT model to generate a response.

Display the chatbot's reply and stop when the user types **"exit"**.

Result

The program successfully created a working chatbot. It accepted user messages, generated intelligent replies using the GPT model, and ended the conversation when the user typed **"exit"**.

7.Solving XOR Problem using DNN

Aim

To solve the XOR logic problem using a Deep Neural Network (DNN) with Keras.

Procedure

Prepare the XOR input data (X) and output labels (y).

Build a neural network with one hidden layer and ReLU activation.

Compile the model using binary cross-entropy and Adam optimizer.

Train the model on the XOR data for several epochs.

Predict the output for all XOR inputs.

Result

The neural network successfully learned the XOR function, producing outputs close to the expected values [0, 1, 1, 0].

7. <u>Face recognition using CNN</u>

Aim

To build a Convolutional Neural Network (CNN) that recognizes faces using the LFW (Labeled Faces in the Wild) dataset.

Procedure

Install required libraries and load the LFW face dataset.

Preprocess the images by normalizing and reshaping them.

One-hot encode the labels and split data into training and testing sets.

Build a CNN model with convolution, pooling, flatten, and dense layers.

Train the model and evaluate accuracy on test data.

Plot training/validation accuracy and loss.

Predict labels for test images and compare with actual labels.

Result

The CNN model successfully learned to classify faces from the LFW dataset and produced a test accuracy (around 70–85%, depending on system). The model also correctly predicted several test face images, demonstrating effective face recognition.

8. <u>Image Augmentation using GANs</u>

AIM

To generate new augmented images using a Generative Adversarial Network (GAN) by training a Generator and Discriminator on the MNIST dataset.

PROCEDURE

Load and normalize the MNIST dataset.

Build the **Generator** to create fake images from random noise.

Build the **Discriminator** to classify real and fake images.

Train both models using adversarial learning for a few iterations.

Generate new augmented images from the trained Generator.

Display the generated images using matplotlib.

RESULT

The GAN successfully generated synthetic MNIST-like handwritten digit images. The output displays multiple generated 28×28 grayscale digit images showing that image augmentation using GANs is achieved.

10.Machine Translation using Encoder-Decoder Model

AIM

To build a simple Encoder–Decoder LSTM model for machine translation that converts an English input sentence into its French translation.

PROCEDURE

Prepare a small example dataset with one English sentence and its French translation.

Tokenize both input (English) and target (French) sentences.

Build the **Encoder** using an Embedding layer and an LSTM that outputs hidden states.

Build the **Decoder** using an Embedding layer, LSTM, and Dense layer with softmax to generate translated words.

Compile the Encoder–Decoder model.

Display the input and expected translated output.

RESULT

A basic Encoder–Decoder model was successfully created for machine translation.
The program displays the English input **"how are you?"** and its French output **"comment ça va ?"**, proving the translation workflow.
