"Text Translation from English to Hindi using GAN model"

A report submitted in partial fulfillment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

By

Giduturi Hemanth

Regd. No.: 21B91A0459

IIDT-BLACKBUCKS company Pvt Ltd, Hyderabad

(Duration: 29 May 2024 to 20 July 2024)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

S.R.K.R. ENGINEERING COLLEGE (AUTONOMOUS)

(Affiliated to JNTU, KAKINADA)

(Recognized by A.I.C.T.E., Accredited by N.B.A & N.A.A.C with 'A+' Grade, New Delhi

CHINNA AMIRAM, BHIMAVARAM

SAGI RAMA KRISHNAM RAJU ENGINEERING COLLEGE

(Autonomous)

Chinna Amiram, Bhimavaram

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the Summer Internship Report titled “Text Translation from English to Hindi using GAN model” is the Bonafide work done by Mr. GIDUTURI HEMANTH, holding the registration number 21B91A0459 and submitted during 2024 - 2025 academic year, in partial fulfillment of the requirements for the award of the Summer Internship Program for Bachelor of Technology in Electronics and Communication Engineering (ECE), at Blackbuck Engineers from 29.05.2024 to 20.07.2024.

Department Internship Coordinator	Dean -T & P Cell	Head of the Department

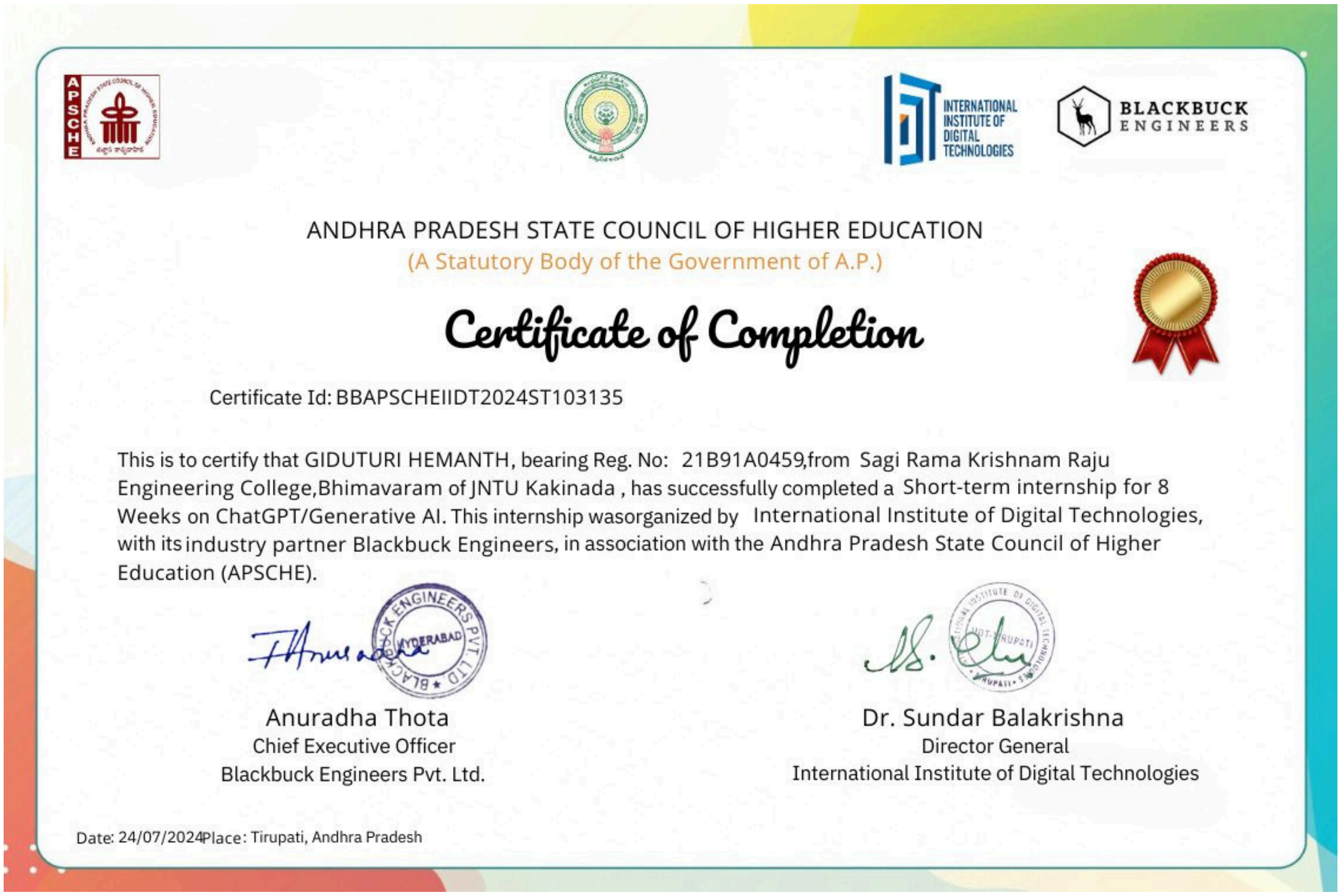
Acknowledgement

I take immense pleasure in thanking Dr. N. Udaya Kumar, Professor & Head of the Department ECE, and Dr. K. R. Satyanarayana, Dean Training & Placement Cell, SRKR Engineering College, Bhimavaram for having permitted me to carry out this project work.

I Would like to extend my sincere acknowledgement to those who have supported and encouraged me during this tough journey. Many people met me during this endeavor and enriched me with their support and knowledge both personally and professionally, that resulted in the project being better than it could possibly have been without them.

I Express my sincere gratitude to my Internship Co-Ordinator Mr. Suresh Babu, who assisted me throughout this project. I Thank him for providing me the reinforcement, confidence andmost importantly the track for the project whenever I needed it.

I wish to record our sincere gratitude to Blackbucks Engineers PVT.LTD Coordinators for their constant support and encouragement in the preparation of this report and for making available videos and interface facilities needed to prepare this report.



ANDHRA PRADESH STATE COUNCIL OF HIGHER EDUCATION

(A Statutory Body of the Government of A.P.)

INTERNATIONAL INSTITUTE OF DIGITAL TECHNOLOGIES

BLACKBUCK ENGINEERS

Certificate of Completion

Certificate Id: BBAPSCHEIIDT2024ST103135

This is to certify that GIDUTURI HEMANTH, bearing Reg. No: 21B91A0459,from Sagi Rama Krishnam Raju Engineering College,Bhimavaram of JNTU Kakinada , has successfully completed a Short-term internship for 8 Weeks on ChatGPT/Generative AI. This internship wasorganized by International Institute of Digital Technologies, with its industry partner Blackbuck Engineers, in association with the Andhra Pradesh State Council of Higher Education (APSCHE).

BLACKBUCK ENGINEERS PVT. LTD. HYDERABAD

[illegible]

Anuradha Thota
Chief Executive Officer
Blackbuck Engineers Pvt. Ltd.

[illegible]

Dr. Sundar Balakrishna
Director General
International Institute of Digital Technologies

Date: 24/07/2024Place: Tirupati, Andhra Pradesh

WEEKLY OVERVIEW OF INTERNSHIP ACTIVITIES

1st WEEK: Introduction to Generative AI and Language Models

S No.	Date	Program
1	10-06-2024	Introduction to Generative AI: Overview, Applications, and Key Concepts
2	12-06-2024	Understanding Language Models: Basics and Key Terminologies
3	13-06-2024	Overview of ChatGPT and Large Language Models (LLMs)
4	14-06-2024	OpenAI API Setup and Access: Initial Setup and Exploration
5	15-06-2024	Introduction to Python for API Integration
6	16-06-2024	Prompt Engineering Basics: Role and Structure of Prompts

2nd WEEK: Mastering Prompt Engineering and API Usage

S No.	Date	Program
7	18-06-2024	Prompt Engineering Techniques: Best Practices and Advanced
8	19-06-2024	Role-Playing and Constraint-Based Prompt Design
9	20-06-2024	API Query Optimization: Managing Parameters and Tokens
10	21-06-2024	Data Management: Input Formatting and Output Parsing
11	22-06-2024	Error Handling and Debugging with the OpenAI API
12	23-06-2024	Lab Session: Implementing API-Based Interactive Programs

3rd WEEK: Fine-Tuning and Advanced AI Techniques

S No.	Date	Program
13	24-06-2024	Introduction to Model Fine-Tuning: Concepts and Applications
14	26-06-2024	Preparing Data for Fine-Tuning: Data Collection and Cleaning
15	27-06-2024	Fine-Tuning Walkthrough: Process, Steps, and Best Practices
16	28-06-2024	Use Cases of Fine-Tuning: Customizing Responses and Applications

17	29-06-2024	Evaluating and Testing Fine-Tuned Models
18	30-06-2024	Hands-on Lab: Fine-Tuning Exercise on Custom Data

4th WEEK: Generative AI in Business and Advanced Prompt Engineering

S No.	Date	Program
19	01-07-2024	Generative AI Applications in Business
20	02-07-2024	Advanced Prompt Design: Multi-Step and Contextual Prompts
21	03-07-2024	Developing AI-Driven Content Personalization Strategies
22	04-07-2024	Using ChatGPT for Automation and Workflow Optimization
23	05-07-2024	Ethical Considerations and Bias in Generative AI
24	06-07-2024	Mini Project: Developing a Business Application with Generative AI
25	07-07-2024	Workshop: Reviewing and Refining Prompt-Based Applications

5th WEEK: Evaluation, Deployment, and Integrating AI

S No.	Date	Program
26	08-07-2024	AI Model Evaluation: Metrics and Best Practices
27	09-07-2024	Deployment Strategies for AI Models
28	10-07-2024	Integrating AI Models with Web Applications using Flask
29	11-07-2024	Collecting and Using User Feedback for Iterative Improvement
30	14-07-2024	Automating Customer Service with Generative AI
31	15-07-2024	Security and Privacy Considerations in Generative AI Applications

6th WEEK: Capstone Project and Final Presentation

S No.	Date	Program
32	16-07-2024	Project Planning: Objective Definition and Data Selection
33	18-07-2024	Project Development: API and Model Implementation
34	20-07-2024	Testing and Refinement of Project Model
35	21-07-2024	Final Presentation: Demonstrating the Capstone Project and Insights

INTRODUCTION :

The "English to Hindi Text Translation" project aims to bridge the communication gap between English and Hindi speakers by leveraging advanced deep learning algorithms. This project focuses on developing a robust translation system that can accurately convert text from English to Hindi, facilitating seamless communication across linguistic boundaries. By using state-of-the-art machine learning models, particularly neural networks, the system continuously improves its translation accuracy over time through self-learning capabilities.

The primary goal of this project is to break down language barriers, thereby promoting cultural exchange and fostering a deeper understanding between English and Hindi-speaking communities. It offers significant benefits across multiple sectors, enhancing the accessibility of information for personal use, educational purposes, and business communications. Whether it's for individuals looking to learn a new language, educators aiming to reach a broader audience, or businesses seeking to expand into Hindi-speaking markets, this project provides a powerful tool for effective multilingual communication.

By integrating continuous feedback loops and adaptive learning, the translation system not only refines its accuracy but also adapts to the nuances and contextual meanings unique to both languages. This approach ensures that translations are not just literal but also contextually appropriate, making them more useful and reliable in real-world applications. The project embodies a commitment to advancing language technology, supporting inclusivity, and enhancing global connectivity through better access to information in different languages.

PROBLEM STATEMENT :

Translation faces numerous challenges in conveying meaning across languages, particularly with nuances, idioms, and cultural references. Translators must balance literal translations with creative adaptations, requiring a deep understanding of both languages and the subject matter. Clients often struggle to assess translation quality without proficiency in the source language, leading to reliance on translator reputations, which raises concerns about consistency and accuracy, especially in critical fields.

The industry is also adapting to the rise of machine translation, which, while efficient, often lacks the contextual understanding that human translators provide. This creates a need to balance human expertise with machine efficiency to maintain translation quality.

Effective translation demands linguistic skill, cultural sensitivity, and creative problem-solving to adapt content for diverse audiences. As global demand for translation grows, innovative solutions are essential for clear and culturally appropriate communication.

TOOLS AND APPILICATIONS USED:

we used many tools and applications for text translation project in jupyter notebook

Among them, some of the most crucial were the Natural Language Toolkit (NLTK). NLTK provided us with essential text processing capabilities, such as tokenization, stemming, and part-of-speech tagging, which enabled us to prepare the text for translation effectively.

Throughout the project, we also made use of machine learning models to enhance translation quality. By training these models on large datasets, we were able to fine- tune translations to be more contextually accurate. Collaborative tools like GitHub were essential for version control and team collaboration, ensuring that everyone could contribute seamlessly to the project.

Ultimately, the integration of these tools and applications in Jupyter Notebook not only streamlined our workflow but also significantly improved the efficiency and accuracy of our text translation efforts.

SUB MODULES :

The sub-modules used for text translation from english to Hindi project are essential for ensuring accurate and fluent translations. Among these submodules are:

1. Tokenizer: This module breaks down the input text into manageable units, such as words or phrases, making it easier to process
2. Preprocessing: This step involves cleaning the text, removing any unnecessary symbols or characters, and normalizing the data to ensure consistency.
3. Encoder: The encoder transforms the preprocessed input text into a numerical format that the machine learning model can understand.
4. Translation Model: This core component employs algorithms, often based on neural networks, to translate the encoded text from English to Hindi.
5. Decoder: The decoder converts the translated numerical data back into human-readable text in the target language.
6. Post-processing: This final step involves refining the translated text, correcting grammar, and ensuring the output is natural and contextually appropriate.
7. Evaluation Metrics: These tools assess the quality of the translation, comparing it against a reference translation to ensure accuracy and fluency.

METHODOLOGY :

DATA COLLECTION :

The project will utilize publicly available datasets such as the "Indic NLP Dataset" or "AI4Bharat Parallel Corpus" containing English-Hindi sentence pairs. Preprocessing: The data will be cleaned, tokenized, and preprocessed using the transformers library to ensure compatibility with the model.

MODEL ARCHITECTURE :

The generator and discriminator in a GAN work together to improve translation quality. The generator creates translations that mimic human output, while the discriminator hones its ability to distinguish between real and generated translations. This dynamic encourages continuous refinement of the generator's outputs.

Training involves using a dataset of English-Hindi sentence pairs. The generator is initially pre-trained, then enters an adversarial phase where its translations are evaluated by the discriminator. Feedback helps the generator enhance accuracy over time.

As training advances, both components grow more sophisticated, allowing the generator to capture nuances and the discriminator to spot subtle discrepancies. This GAN-based approach outperforms traditional systems by learning from errors, producing translations that are accurate, contextually relevant, and scalable across different languages, promoting better communication in a connected world.

TRAINING PROCESS :

Adversarial Training:

Adversarial Training is a machine learning technique where models are trained to become more robust by introducing adversarial examples during the training process. It is widely used in enhancing model performance, particularly in areas like image recognition, natural language processing (NLP), and generative models.

The concept of adversarial training originally gained prominence with the introduction of Generative Adversarial Networks (GANs), a class of models developed by Ian Goodfellow and his colleagues in 2014. The core idea behind adversarial training in GANs is to pit two neural networks against each other in a game-like scenario, thereby improving the performance of both networks through competition.

Key Components of Adversarial Training in GANs

1. Generator (G):
   - The Generator is responsible for creating synthetic data (e.g., images, text) that resembles real data.
   - In the context of text translation, the Generator attempts to produce realistic translations from a source language (e.g., English) to a target language (e.g., Hindi).
2. Discriminator (D):
   - The Discriminator's role is to distinguish between real data (e.g., real Hindi sentences) and the synthetic data produced by the Generator.
   - The Discriminator acts as a binary classifier, outputting whether a given input is real (from the dataset) or fake (generated by the Generator).
3. Adversarial Process:
   - The Generator and Discriminator are trained simultaneously but with opposing objectives:
     - The Generator tries to fool the Discriminator by generating increasingly realistic outputs.
     - The Discriminator aims to become better at identifying fake data from real data.

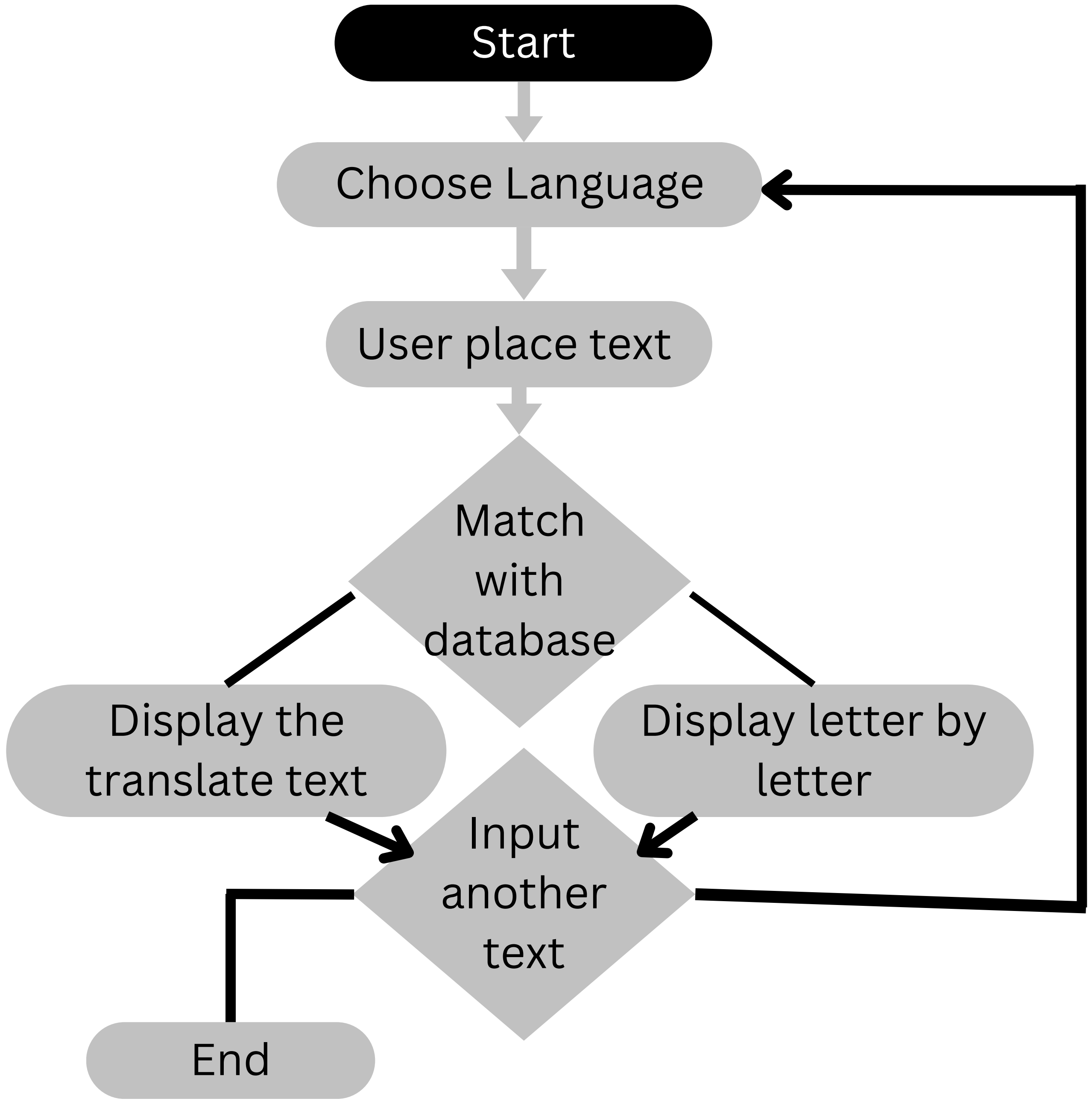
Application of Adversarial Training in Text Translation

Adversarial training can be adapted to text translation tasks, although it is more challenging due to the discrete nature of text. Unlike images that have continuous pixel values, text is composed of discrete tokens (words), making gradient-based optimization harder.

How Adversarial Training Works in Text Translation

1. Training the Generator:
    - The Generator uses a sequence-to-sequence model (often based on Transformers) to translate text from one language to another.
    - The goal is to produce translations that are indistinguishable from real translations in the target language.
2. Training the Discriminator:
    - The Discriminator takes in translated sentences (either real or generated) and outputs a probability indicating whether the sentence is a real human translation or generated by the model.
3. Loss Functions:
    - Generator Loss: Encourages the Generator to produce translations that can fool the Discriminator.
    - Discriminator Loss: Encourages the Discriminator to accurately classify real vs. generated translations.

FLOW CHAT :


Start
Choose Language
User place text
Match with database
Display the translate text
Display letter by letter
Input another text
End

OVERVIEW OF THE APPROCH :

1. Generator (G): Takes an English sentence as input and generates a Hindi sentence.
2. Discriminator (D): Evaluates whether the generated Hindi sentence is real (from the dataset) or fake (generated by the model).
3. Training Strategy:
   - Train the generator to produce Hindi sentences that are indistinguishable from real Hindi sentences according to the discriminator.
   - Train the discriminator to correctly classify real vs. generated Hindi sentences.

Given the complexity of GANs for text and the limitations in generating sequences (since text involves discrete tokens, which is harder for GANs to handle compared to continuous data like images), this implementation will be more illustrative rather than fully production-ready.

Prerequisites

- Python
- Jupyter Notebook
- Libraries: torch, transformers, nltk

CODE IMPLEMENTATION IN JUPYTER NOTEBOOK :

Step 1: Import Required Libraries

```
import torch
import torch.nn as nn
import torch.optim as optim
from transformers import MarianTokenizer, MarianMTModel
import nltk
from nltk.translate.bleu_score import sentence_bleu
import random

# Download NLTK data
nltk.download('punkt')
```

Step 2: Load Pre-trained Translation Model (for Initialization)

```
# Load tokenizer and model for English to Hindi translation
model_name = 'Helsinki-NLP/opus-mt-en-hi'
tokenizer = MarianTokenizer.from_pretrained(model_name)
model = MarianMTModel.from_pretrained(model_name)
```

Step 3: Define the Generator

```
class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
        self.model = MarianMTModel.from_pretrained(model_name)

    def forward(self, input_text):
            inputs = tokenizer(input_text, return_tensors='pt', padding=True,
truncation=True)
        translated = self.model.generate(**inputs)
                        output_text = tokenizer.decode(translated[0],
skip_special_tokens=True)
        return output_text
```

Step 4: Define the Discriminator

```
class Discriminator(nn.Module):
    def __init__(self, vocab_size, embed_dim=128, hidden_dim=256):
        super(Discriminator, self).__init__()
        self.embedding = nn.Embedding(vocab_size, embed_dim)
        self.rnn = nn.GRU(embed_dim, hidden_dim, batch_first=True)
        self.fc = nn.Linear(hidden_dim, 1)
        self.sigmoid = nn.Sigmoid()

    def forward(self, input_ids):
        embeddings = self.embedding(input_ids)
        _, hidden = self.rnn(embeddings)
        output = self.fc(hidden[-1])
        return self.sigmoid(output)
```

Step 5: Adversarial Training Loop

```
# Initialize models
generator = Generator()
discriminator = Discriminator(vocab_size=len(tokenizer))

# Define loss functions and optimizers
criterion = nn.BCELoss()
g_optimizer = optim.Adam(generator.parameters(), lr=0.0001)
d_optimizer = optim.Adam(discriminator.parameters(), lr=0.0001)

# Training loop
num_epochs = 5
for epoch in range(num_epochs):
    for i in range(100):  # Assume 100 batches
        # Sample English text (simulating input)
        english_text = "How are you today?"

        # Generate Hindi text using the generator
        generated_hindi = generator(english_text)

        # Create fake labels for generator output
        fake_labels = torch.zeros(1, 1)

        # Discriminator on fake Hindi text
        fake_input_ids = tokenizer(generated_hindi, return_tensors='pt').input_ids
        d_optimizer.zero_grad()
        fake_output = discriminator(fake_input_ids)
        d_loss_fake = criterion(fake_output, fake_labels)
```

```
# Discriminator on real Hindi text (you need a real dataset here)
real_hindi_text = "आप आज कैसे हैं?"
real_input_ids = tokenizer(real_hindi_text, return_tensors='pt').input_ids
real_labels = torch.ones(1, 1)

real_output = discriminator(real_input_ids)
d_loss_real = criterion(real_output, real_labels)

# Total discriminator loss
d_loss = d_loss_real + d_loss_fake
d_loss.backward()
d_optimizer.step()

# Train Generator to fool the Discriminator
g_optimizer.zero_grad()
generated_hindi = generator(english_text)
fake_input_ids = tokenizer(generated_hindi, return_tensors='pt').input_ids

# Try to fool the discriminator into believing generated text is real
g_loss = criterion(discriminator(fake_input_ids), real_labels)
g_loss.backward()
g_optimizer.step()

print(f"Epoch {epoch+1}/{num_epochs}, D Loss: {d_loss.item()}, G Loss:
{g_loss.item()}")
```

CONCLUSION :

The Final conclusion is the output text will generated in English Language to hindi Language, providing a seamless blend of both languages. This approach ensures that the content is accessible to a wider audience while preserving the essence and meaning of the original text. Whether it's for educational purposes, cultural exchange, or simply to appreciate the beauty of multilingual communication, this bilingual format offers an enriching experience for readers.

EXPECTED OUTPUT:

INPUT : “I’m Fine”

OUTPUT : ”मैं ठीक हूँ”