“End-to-End Multilingual Speech Processing: STT, Mental Health Detection, and TTS”.

OpenAI Whisper: Speech-to-Text (STT) System

1. Introduction

OpenAI Whisper is a state-of-the-art speech recognition model capable of **multilingual transcription** and **translation**.
It uses a transformer-based **encoder–decoder architecture** and is trained on a large, diverse dataset of audio and text pairs.
Whisper not only performs **speech-to-text** in multiple languages but also handles **language identification**, **translation**, and **robust transcription** even in noisy environments.

2. Whisper Architecture

2.1 Log-Mel Spectrogram

- The raw audio waveform is converted into a **log-Mel spectrogram**.
- A **spectrogram** is a visual representation of sound, showing how frequency content evolves over time.
- The **Mel scale** mimics how humans perceive pitch — compressing high frequencies more than low frequencies.
- Log scaling further emphasizes perceptually important differences.

Key Points of Log-Mel Spectrogram:

- Captures both **time** and **frequency domain** information.
- Makes speech features **language-independent**.
- Provides a compact input for neural networks.

2.2 Encoder

- Input: Log-Mel spectrogram.
- The encoder contains multiple **Transformer Encoder Blocks**.
- Each block has:
  - **Multi-Head Self-Attention** – captures relationships across time steps in the spectrogram.
  - **Feedforward MLP layers** – learn higher-level abstractions.
  - **Sinusoidal Positional Encoding** – injects order information (time).

2.3 Decoder

- Input: Previously generated tokens (text).
- The decoder uses:
  - **Self-Attention** – relates words within the text sequence.
  - **Cross-Attention** – links encoder outputs (audio features) with the text being generated.
  - **MLPs** for transformation and prediction.
  - **Learned Positional Encoding** for token order.

Role of Decoder:

Generates the transcription token-by-token, conditioned on audio features and previously generated text.

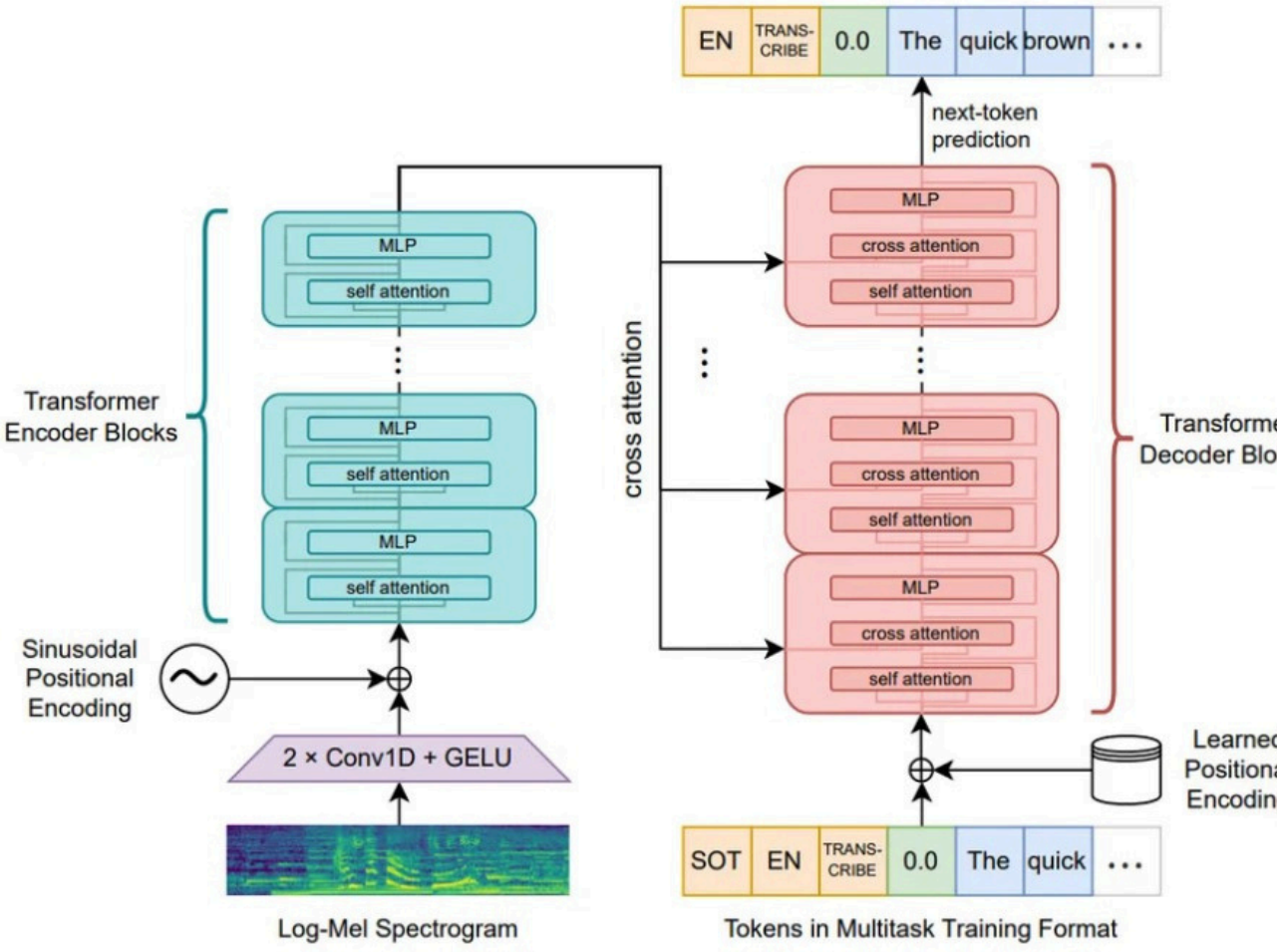
2.4 Multilingual & Language Detection

- Whisper is trained in a **multitask setup**:
  - **SOT (Start of Transcript) tokens** indicate beginning of decoding.
  - Language-specific tokens (<|en|>, <|hi|>, etc.) guide transcription in the correct language.
- Whisper learns to **detect language automatically** from spectrogram features.
- It associates unique spectro-temporal patterns of speech (intonation, phoneme frequencies) with language identifiers.

This allows Whisper to seamlessly switch between **English, Hindi, and other languages**, making it robust in multilingual settings.

3. Workflow:(open-Ai)

1. Audio waveform → **Log-Mel Spectrogram**.
2. Spectrogram → **Encoder** → embeddings.
3. **Decoder** generates tokens → transcribed/translated text.
4. Final text can be **monolingual transcription** or **cross-lingual translation**.



Whisper Architecture

Grammar Correction in Multilingual Speech Pipeline:

1. Introduction

Grammar correction is an essential step in ensuring that transcribed text (from Whisper STT) is **fluent, syntactically correct, and meaningful**.
In this project, two approaches are used for grammar correction and translation depending on the language:

- **English Grammar Correction → T5 Transformer model**
- **Hindi Grammar Correction/Translation → mBART and MarianMT models**

2. English Grammar Correction

Model Used: T5 (Text-to-Text Transfer Transformer)

- We use **vennify/t5-base-grammar-correction**, a fine-tuned version of Google's **T5-base**.
- T5 is a **sequence-to-sequence transformer model** where every NLP task is cast as text-to-text.
- Input: noisy/ungrammatical English text.
- Output: corrected English text.

Working:

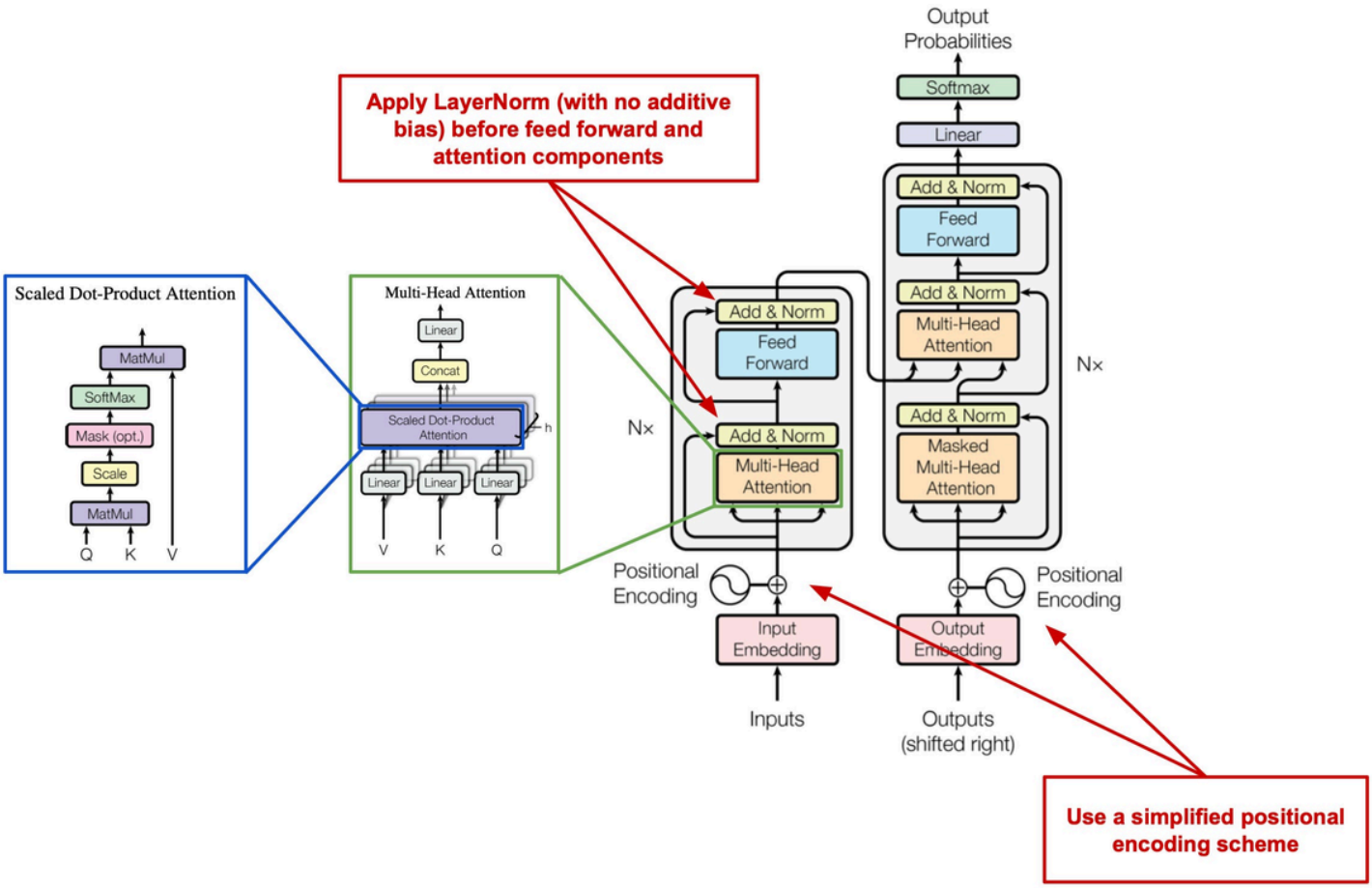
1. The input transcription text is tokenized.
2. Encoder processes input → produces embeddings.
3. Decoder generates grammatically corrected sequence, token by token.

Example:

Input (Uncorrected):"I am very hapy today becuse the wether is good."

Output (Corrected by T5 Model):"I am very happy today because the weather is good."

workflow:



T5 (Text-to-Text Transfer Transformer):

3. Hindi Grammar Correction & Translation

Hindi grammar correction is more complex since **direct grammar correction datasets/models** are scarce. Instead, we use **translation-based normalization**:

3.2 mBART-50 (Multilingual BART)

- Model: "facebook/mbart-large-50-many-to-many-mmt"
- Supports **50+ languages**, including Hindi and English.
- Used for **direct Hindi → English translation** (better than MarianMT for some sentences).
- Translation improves **fluency and grammar** because the decoder reconstructs proper syntax.

Working:

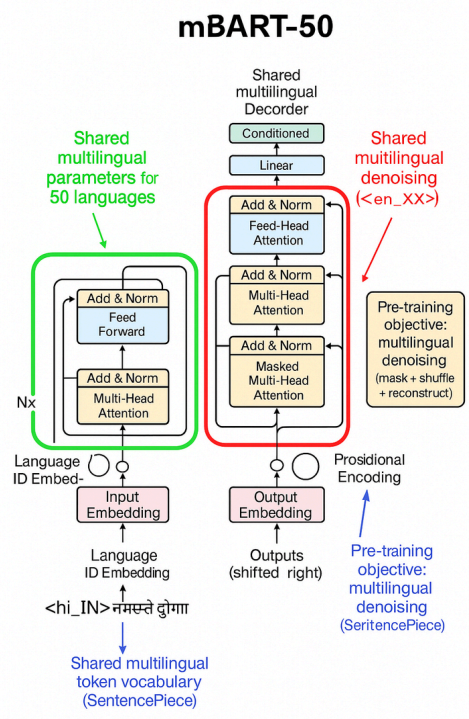
1. Input Hindi text is tokenized with language tags (hi_IN).
2. Encoder processes the tokens into embeddings.
3. Decoder generates English text tokens (en_XX).
4. Sentiment analysis is performed on English text (since the classifier was trained on English).
5. If needed, text is translated back into Hindi for TTS output.

4. Integration into Pipeline:

- **For English audio/text**:
  - Whisper → T5 Grammar Correction → Sentiment Analysis → TTS
- **For Hindi audio/text**:
  - Whisper → Hindi transcription → **mBART (Hindi → English)** → Sentiment Analysis → MarianMT (English → Hindi) → TTS

4. Integration into Pipeline

- **For English audio/text**:
  - Whisper → T5 Grammar Correction → Sentiment Analysis → TTS
- **For Hindi audio/text**:
  - Whisper → Hindi transcription → **mBART (Hindi → English)** → Sentiment Analysis → MarianMT (English → Hindi) → TTS



4. Sentiment / Mental Health Detection using DistilBERT

4.1 Introduction

After grammar correction and translation, the next step in the multilingual speech pipeline is **mental health detection** based on the emotional and linguistic tone of the text.
This task is implemented using a **fine-tuned DistilBERT model** trained on a balanced dataset containing multiple mental health categories such as *Depression, Stress, Anxiety, Bipolar, Suicidal, Personality Disorder,* and *Normal*.

The goal is to automatically detect and classify the **mental health condition** reflected in a speaker's transcribed or written text, enabling early awareness and digital mental health assistance.

4.3 Dataset and Preprocessing

A **balanced text dataset** containing user posts and statements related to mental health was used.
The preprocessing pipeline included:

- Lowercasing and punctuation removal
- Contraction expansion (e.g., *I'm → I am*)
- Tokenization using DistilBertTokenizerFast
- One-hot encoding of class labels using LabelEncoder

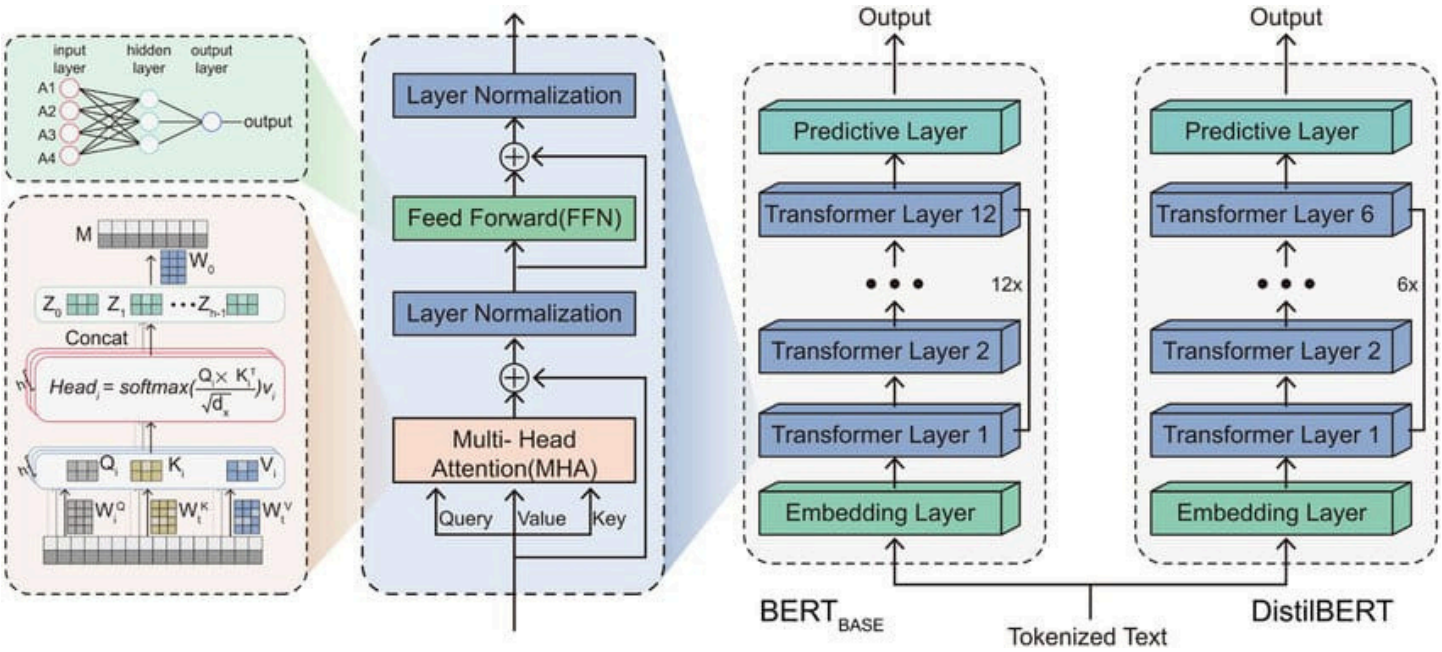
The dataset was divided into **training (82%)** and **validation (18%)** splits for performance evaluation.

4.2 Model Architecture

DistilBERT is a lighter and faster version of BERT that retains 97% of its performance while using only 40% of the parameters.
It is based on the **Transformer encoder** architecture and trained using **knowledge distillation**, which transfers knowledge from a larger model (BERT-base) into a smaller one.

Key Components

- **Tokenizer:** Converts input text into token IDs with attention masks.
- **Embedding Layer:** Maps tokens to high-dimensional feature vectors.
- **Encoder Layers:** Stack of self-attention and feedforward sublayers that extract contextual representations.
- **Classification Head:** A dense output layer maps the [CLS] token representation to one of the target mental health classes.



4.2 (DistilBERT)

4.4 Training Process

- **Model Used:** TFDistilBertForSequenceClassification (TensorFlow implementation)
- **Batch Size:** 16
- **Learning Rate:** 3e-5 with linear warmup and weight decay
- **Epochs:** 6 (achieved 92%+ accuracy)
- **Loss Function:** Categorical Cross-Entropy
- **Optimizer:** Adam with a custom learning rate scheduler
- **Evaluation Metrics:** Accuracy, Precision, Recall, F1-score, and Confusion Matrix visualization

The model predicts a mental health category for a given text and provides a confidence score (softmax probability).

Example output:

Input: "I feel hopeless and empty every day."

Predicted Status: Depression

Confidence: 65.6%

This model integrates seamlessly into the main pipeline for both English and Hindi inputs.

5. Text-to-Speech (TTS) Generation

5.1 Overview

The final stage of the proposed multilingual speech processing pipeline is **Text-to-Speech (TTS)** conversion, which transforms the processed and sentiment-classified text back into audible speech.
This step ensures a complete **end-to-end human–machine communication loop**, where the system not only understands and interprets speech but also responds naturally in the same or another language.

In this project, the **Google Text-to-Speech (gTTS)** API is utilized for synthesizing speech from text in both **English and Hindi**.

5.2 Working Principle

The **gTTS** library is a lightweight and efficient interface for Google's TTS engine, which uses **deep neural network–based speech synthesis** to generate natural-sounding speech.
It supports multiple languages, accents, and voices, making it ideal for multilingual applications like this project.

The working process involves the following steps:

1. **Input Text:**
   The grammatically corrected or translated text (output from the previous modules) is passed to the TTS engine.
2. **Language Detection:**
   The system determines whether the text is in **English** or **Hindi**, based on the input pipeline.
3. **Speech Synthesis:**
   - For **English text**, gTTS uses "lang='en'".
   - For **Hindi text**, it uses "lang='hi'".
     The model converts the text to an audio waveform in .mp3 format using Google's neural vocoder.
4. **Audio Playback:**
   The synthesized speech is saved temporarily (e.g., /tmp/audio.mp3) and played back using Gradio's audio component.

5.5 Example Output

Input Text (English):

“I am feeling happy and calm today.”
Predicted Sentiment: (Normal**)**
TTS Output (Audio):
A clear English voice saying *“I am feeling happy and calm today.”*

Input Text (Hindi):

“मुझे आज बहुत अच्छा लग रहा है।”
Predicted Sentiment: (Normal**)**
TTS Output (Audio):
A natural Hindi voice saying *“मुझे आज बहुत अच्छा लग रहा है।”*

5.6 Summary

The TTS module completes the multilingual NLP pipeline by generating natural, expressive, and language-consistent audio responses.
By combining Whisper for speech-to-text, T5 for grammar correction, mBART for translation, DistilBERT for sentiment analysis, and gTTS for speech synthesis, the system achieves a **fully functional end-to-end multilingual communication framework**.

5.7 Reference

- Google Text-to-Speech API (gTTS): https://pypi.org/project/gTTS/
- van den Oord et al., “WaveNet: A Generative Model for Raw Audio,” *Google DeepMind*, 2016.