Lab 11 : LSTM, Deepspeech, Wav2vec, Wav2letter use models for speech recognition

Objective:

The goal of **Speech Recognition Models** is to explore and implement different deep learning-based speech recognition models, including **Wav2Letter, DeepSpeech, and Wav2Vec 2.0**.

Input:

An audio file containing spoken words, such as:

Output:

The corresponding **text transcription** of the spoken words.

Wav2letter

In [ ]:
```
import torch
import torchaudio
from transformers import Wav2LetterProcessor, Wav2LetterForCTC
import numpy as np
import wave

def load_audio(filepath):
    waveform, sample_rate = torchaudio.load(filepath)
    target_sample_rate = 16000  # Wav2Vec expects 16kHz
    if sample_rate != target_sample_rate:
        transform = torchaudio.transforms.Resample(orig_freq=sample_rate, new_freq=target_sample_rate)
        waveform = transform(waveform)
    return waveform, target_sample_rate

# LSTM Model (Simplified)
class LSTMSpeechRecognizer(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim, num_layers=2):
        super(LSTMSpeechRecognizer, self).__init__()
        self.lstm = torch.nn.LSTM(input_dim, hidden_dim, num_layers, batch_first=True)
        self.fc = torch.nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        x, _ = self.lstm(x)
        x = self.fc(x[:, -1, :])
        return x

# Load Wav2Letter 2.0 Model
processor = Wav2LetterProcessor.from_pretrained("facebook/wav2Letter-base-960h")
wav2vec_model = Wav2LetterForCTC.from_pretrained("facebook/wav2Letter-base-960h")

def wav2vec_recognize(filepath):
    waveform, sample_rate = load_audio(filepath)
    input_values = processor(waveform.squeeze().numpy(), return_tensors="pt", sampling_rate=sample_rate).input_val
    logits = wav2vec_model(input_values).logits
    predicted_ids = torch.argmax(logits, dim=-1)
    transcription = processor.batch_decode(predicted_ids)[0]
    return transcription

# Test function
def test_models(filepath):
    print("\n\n\nWav2Letter Recognition:", wav2vec_recognize(filepath))

# Example usage
audio_file = "../dataset/Recordings/Recording-1.wav"
test_models(audio_file)
```

```
Some weights of Wav2Vec2ForCTC were not initialized from the model checkpoint at facebook/wav2vec2-base-960h and ar
e newly initialized: ['wav2vec2.masked_spec_embed']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```
Wav2Letter Recognition: LIVER IS NORMAL IN SIZE AND WITH NODMAL PAREN CHAIMAL ECOGENOCITY WITH NO SIGN OF SPACE OCC
UPYING LEISION OR BIDUCT'S DILATION
```

DeepSpeech

In [5]:

```python
import deepspeech
import numpy as np
import wave

# Load DeepSpeech Model
model_file = "../deepspeech-0.9.3-models.pbmm"
scorer_file = "../deepspeech-0.9.3-models.scorer"
model = deepspeech.Model(model_file)
model.enableExternalScorer(scorer_file)

# Function to read audio
def read_audio(filename):
    with wave.open(filename, 'rb') as wf:
        audio = np.frombuffer(wf.readframes(wf.getnframes()), dtype=np.int16)
        return audio, wf.getframerate()

# Transcribe speech
audio, rate = read_audio("../dataset/numbers/one/00176480_nohash_0.wav")
text = model.stt(audio)
print("Transcription:", text)
```

```
TensorFlow: v2.3.0-6-g23ad988
DeepSpeech: v0.9.3-0-gf2e9c85
Transcription: one
```

Wav2Vec 2.0-Based Speech Recognition (PyTorch)

In [6]:

```python
import torch
import torchaudio
from torchaudio.transforms import Resample
from transformers import Wav2Vec2ForCTC, Wav2Vec2Processor

# Load pretrained Wav2Vec2 model
model_name = "facebook/wav2vec2-base-960h"
processor = Wav2Vec2Processor.from_pretrained(model_name)
model = Wav2Vec2ForCTC.from_pretrained(model_name)

# Load audio
waveform, sample_rate = torchaudio.load("../dataset/Recordings/Recording-1.wav")

# Check if resampling is needed
target_sample_rate = 16000
if sample_rate != target_sample_rate:
    resampler = Resample(orig_freq=sample_rate, new_freq=target_sample_rate)
    waveform = resampler(waveform)

# Process input
input_values = processor(waveform.squeeze().numpy(), sampling_rate=target_sample_rate, return_tensors="pt").input_

# Perform inference
with torch.no_grad():
    logits = model(input_values).logits

# Decode prediction
predicted_ids = torch.argmax(logits, dim=-1)
transcription = processor.batch_decode(predicted_ids)[0]
print("\n\nTranscription:", transcription)
```

```
Some weights of Wav2Vec2ForCTC were not initialized from the model checkpoint at facebook/wav2vec2-base-960h and ar
e newly initialized: ['wav2vec2.masked_spec_embed']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```
Transcription: LIVER IS NORMAL IN SIZE AND WITH NODMAL PAREN CHAIMAL ECOGENOCITY WITH NO SIGN OF SPACE OCCUPYING LE
ISION OR BIDUCT'S DILATION
```

Inference (Understanding the Process):

1. Wav2Letter (Character-Based Speech Recognition)

- Uses **Wav2LetterProcessor** and **Wav2LetterForCTC** from **Hugging Face**.
- Converts speech into a **sequence of characters** using **CTC (Connectionist Temporal Classification)**.
- **Steps:**
  1. Load the **audio file** and **resample** it to 16kHz.
  2. Convert it into a format compatible with the model.
  3. Perform inference to obtain **logits**.
  4. Decode the predicted characters into words.

2. DeepSpeech (End-to-End Speech Recognition)

- Uses the **Mozilla DeepSpeech** model.
- Converts **audio waveforms** into **text** using **a pre-trained deep RNN**.
- **Steps:**
  1. Load the **DeepSpeech model** and external **scorer** for language modeling.
  2. Read the **audio file** as an array.
  3. Perform inference to **generate a text transcription**.

3. Wav2Vec 2.0 (Transformer-Based Speech Recognition)

- Uses **Wav2Vec2ForCTC** from **Hugging Face Transformers**.
- Learns speech representations in a **self-supervised manner**.
- **Steps:**
  1. Load the **pre-trained Wav2Vec2 model**.
  2. Preprocess the **audio waveform** (resampling if needed).
  3. Perform inference to obtain **logits**.
  4. Use a **decoder** to extract words.

Comparison of Models:

Model	Architecture	Training Style	Strengths
Wav2Letter	CNN-based	Supervised	Fast and lightweight
DeepSpeech	RNN-based (LSTM/GRU)	Supervised	Efficient for **low-resource devices**
Wav2Vec 2.0	Transformer-based	**Self-Supervised**	High accuracy, **best for real-world tasks**

Conclusion:

- **Wav2Vec 2.0** is the most advanced and accurate for **modern speech recognition tasks**.
- **DeepSpeech** is a great balance between **efficiency and performance**.
- **Wav2Letter** is simple but **outdated compared to Wav2Vec 2.0**.