Speech Understanding
Programming Assignment 2
Report

[Link to Github Repository]

[Question 1. Speaker Verification]
Goal: In speaker verification, the training dataset consists of audio clips paired with speaker IDs, denoted as (D = (x_i, y_i)). Given an audio clip (x) and a reference clip (x_0), the objective is to ascertain whether (x_0) and (x) belong to the same speaker.

(Task i) Choose three pre-trained models from the list: 'ecapa_tdnn', 'hubert_large', 'wav2vec2_xlsr', 'unispeech_sat', 'wavlm_base_plus', 'wavlm_large' trained on the VoxCeleb1 dataset. You can find the pre-trained models on this link.

Now, as per the requirements of Task i, I choose three pre-trained models as following: 'hubert_large', 'wavlm_base_plus', and 'wavlm_large'. The justification and reasoning for the same can be found as follows:

hubert_large:

- Justification: The Hubert model is a highly efficient and effective audio model based on self-supervised learning. It is designed specifically for processing audio data efficiently.
- Reasoning: In the context of speaker verification, the Hubert model's architecture allows it to capture essential features of audio signals relevant to speaker identity. Its large size implies a more complex representation capability, potentially capturing nuanced speaker characteristics.

wavlm_base_plus:

- Justification: The Wav2Vec2 model architecture, upon which wavlm_base_plus is based, has shown impressive performance on various audio-related tasks, including speech recognition and speaker verification.
- Reasoning: The Wav2Vec2 model employs self-supervised learning to learn representations from raw audio, making it suitable for tasks such as speaker verification where capturing intricate audio features is crucial. Additionally, choosing the base plus variant balances model size and performance, making it computationally efficient while still capable of capturing relevant speaker information.

wavlm_large:

- Justification: The 'wavlm_large' model represents a more complex and deeper architecture compared to the 'wavlm_base_plus'. In complex tasks like speaker verification, larger models often have a better capacity to capture subtle speaker characteristics.
- Reasoning: By selecting the 'wavlm_large' model, we aim to leverage its increased model capacity to potentially capture more nuanced aspects of

speaker identity, thereby enhancing the accuracy and robustness of the speaker verification system.

In summary, the choice of 'hubert_large', 'wavlm_base_plus', and 'wavlm_large' models for speaker verification is justified based on their architecture, efficiency, performance on related tasks, and the balance between model size and capacity to capture relevant audio features for speaker identification. These models are expected to provide a solid foundation for achieving accurate speaker verification results.

(Task ii) Calculate the EER (%) on the VoxCeleb1-H dataset using the above selected models. You can get the dataset from here.

Now, EER(%) on the VoxCeleb1-H dataset using the above selected models is as follows:

Models	EER (%)
hubert_large	1.8334
wavlm_base_plus	2.0015
wavlm_large	1.4530

Based on the obtained Equal Error Rate (EER) values for the selected pre-trained models on the VoxCeleb1-H dataset, the results demonstrate promising performance across the board. The 'hubert_large' model achieves an EER of 1.8334%, showcasing its robust capability in speaker verification tasks. Similarly, the 'wavlm_base_plus' model achieves a slightly higher but still commendable EER of 2.0015%, indicating its effectiveness in capturing relevant speaker features despite its relatively smaller size compared to 'wavlm_large'. Impressively, the 'wavlm_large' model achieves the lowest EER of 1.4530%, underscoring its superior capacity to discern subtle nuances in speaker characteristics owing to its larger architecture.

In comparison, while all three models perform well, 'wavlm_large' outperforms both 'hubert_large' and 'wavlm_base_plus' models, boasting the lowest EER. However, it's worth noting that the differences in EER among the models are relatively small, emphasizing the overall effectiveness of all selected models in the speaker verification task. Despite the marginal variation, the 'wavlm_large' model demonstrates a slight edge over the others, suggesting its potential for achieving higher accuracy and reliability in real-world speaker verification applications.

(Task iii) Compare your result with Table II of the WavLM paper.
Now, upon drawing a comparison of my results with Table II of the WavLM paper, it can be noted that:

Models	EER (%) calculated	EER (%) reported
hubert_large	1.8334	1.678
wavlm_base_plus	2.0015	1.758
wavlm_large	1.4530	1.318

The calculated Equal Error Rate (EER) values for the 'hubert_large', 'wavlm_base_plus', and 'wavlm_large' models closely align with the reported values from Table II of the WavLM paper, demonstrating strong consistency in performance trends. For the 'hubert_large' model, the calculated EER of 1.8334% is slightly higher than the reported value of 1.678%, differing by approximately 0.155%. Similarly, the 'wavlm_base_plus' model exhibits a calculated EER of 2.0015%, marginally exceeding the reported value of 1.758% by approximately 0.243%. Likewise, the calculated EER for the 'wavlm_large' model, at 1.4530%, closely resembles the reported value of 1.318%, differing by only approximately 0.135%. These minor differences in EER values could stem from variations in dataset preprocessing, model fine-tuning, or evaluation methodologies. However, despite these discrepancies, the alignment of performance trends between the calculated and reported values underscores the reliability and effectiveness of the selected models in speaker verification tasks, affirming their consistency and robustness across different experimental setups.

(Task iv) Evaluate the selected models on the test set of any one Indian language of the Kathbath Dataset. Report the EER (%)
Now, I have chosen the ‘hindi’ language of the Kathbath Dataset to evaluate upon the selected models. The calculated EER (%) is as follows:

Models	EER (%)
hubert_large	3.9823
wavlm_base_plus	4.0023
wavlm_large	3.5067

The evaluation results on the test set of the Indian language from the Kathbath Dataset reveal promising performance for the selected pre-trained models. The 'hubert_large' model achieved an EER of 3.9823%, showcasing its effectiveness in speaker verification tasks even in the context of Indian languages, which may present unique linguistic challenges. Similarly, the 'wavlm_base_plus' model attained an EER of 4.0023%, demonstrating its competitive performance despite potential variations in linguistic characteristics and data quality within the Indian language test set. Impressively, the 'wavlm_large' model outperformed the others with an EER of 3.5067%, reaffirming its superior capability to capture intricate speaker characteristics,

which may be particularly advantageous when dealing with the linguistic diversity and data complexity inherent in Indian languages. Overall, the obtained EER values align with the expected performance trends, indicating that the selected models are robust and versatile for speaker verification tasks across different linguistic contexts, including Indian languages from the Kathbath Dataset.

(Task v) Fine-tune, the best model on the validation set of the selected language of Kathbath Dataset. Report the EER(%).
Now, as per the results of Task iv, wavlm_large is the best performing model. Accordingly, I fine-tuned it on the hindi language of the Kathbath dataset and the results are as follows:

Model	EER(%) without fine tune	EER(%) with fine tune
wavlm_large	3.5067	2.8432

(Task vi) Provide an analysis of the results along with plausible reasons for the observed outcomes.
The comparison between the EER values of the 'wavlm_large' model without fine-tuning and after fine-tuning on the Hindi language of the Kathbath Dataset reveals a notable improvement in performance following the fine-tuning process. Initially, the 'wavlm_large' model achieved an EER of 3.5067% without fine-tuning, demonstrating solid performance in speaker verification tasks. However, after fine-tuning on the Hindi language dataset, the EER significantly decreased to 2.8432%, indicating a substantial enhancement in performance.

The reduction in Equal Error Rate (EER) from 3.5067% without fine-tuning to 2.8432% with fine-tuning for the 'wavlm_large' model can be attributed to several technical enhancements facilitated by the fine-tuning process. Fine-tuning allows the model to adapt its parameters to the specifics of the target dataset, in this case, likely the Kathbath Dataset in Hindi, leading to improved performance in speaker verification tasks on Hindi speech. Through domain adaptation, the model refines its representations to better capture the nuances of Hindi speech, resulting in more accurate speaker discrimination. Additionally, fine-tuning facilitates continuous feature learning from the target dataset, enabling the model to extract more discriminative features and better distinguish between speakers. By reducing overfitting and enhancing model capacity utilization, fine-tuning ensures that the 'wavlm_large' model effectively generalizes to unseen data and utilizes its complex architecture to capture intricate speaker characteristics. Collectively, these technical improvements contribute to the observed reduction in EER, highlighting the efficacy of fine-tuning in enhancing the performance of the 'wavlm_large' model for speaker verification tasks in Hindi speech.

[Question 2. Source Separation]

Goal: The goal of speech separation is to estimate individual speaker signals from their mixture, where the source signals may be overlapped with each other entirely or partially.

(Task i) Generate the LibriMix dataset by combining two speakers from the LibriSpeech dataset, focusing solely on the LibriSpeech_test_clean partition. Take help from this Github repo.
Now, to generate the LibriMix dataset by combining two speakers from the LibriSpeech dataset the following steps are follows:

- First, I have cloned the provided github repository.
- Next, I have downloaded the required dataset: Librispeech and whamnoise.
- Next, I have deleted the unnecessary files from the metadata folder which comprises of the train and validation files.
- Lastly, with a few modifications in the provided scripts, I used the 'generate_librimix.sh' file to generate the required data.

(Task ii) Partition the resulting LibriMix dataset into a 70-30 split for training and testing purposes. Evaluate the performance of the pre-trained Sepformer on the testing sset, employing scale-invariant sigal-to-noise ratio improvement (SISNRi) and signal-to-distortion ratio improvement (SDRi) as metrics. For metric computation, consult the provided paper and utilize the code from torchmetrics.
Now, after partitioning the resulting LibriMix dataset into a 70-30 split for training and testing purposes, I have evaluated the performance of the pre-trained Sepformer on the testing set. The observed values are as follows:

Metric	Value
SISNRi	14.67
SDRi	11.65

The reported results of Sepformer on the LibriMix test partition, with a Signal-to-Interference-plus-Noise Ratio improvement (SISNRi) of 14.67 and a Source-to-Distortion Ratio improvement (SDRi) of 11.65, signify the model's strong performance in speech separation tasks. The SISNRi metric measures the improvement in the quality of the separated speech signal compared to the original mixture, focusing on the speech signal's clarity relative to the interference and noise. A SISNRi value of 14.67 indicates a significant enhancement in speech intelligibility and clarity after separation, suggesting that Sepformer effectively isolates speech from background noise and interference. Similarly, the SDRi metric evaluates the reduction in distortion or artifacts introduced during the separation process, with a higher SDRi indicating better preservation of the original speech signal's quality. With an SDRi of 11.65, Sepformer demonstrates its ability to minimize distortion and faithfully recover the original speech signal from the mixture, leading to high-quality separated audio. These results underscore Sepformer's effectiveness in tackling the challenging task of speech separation,

making it a promising candidate for real-world applications such as speech enhancement and speaker diarization.

(Task iii) Fine-tune the SepFormer model using the training set and report its performance on the test split of the LibriMix dataset.
Now, I have fine-tuned the SepFormer model using the training set and the resulting values are as follows:

Metric	Without fine-tune	With fine-tune
SISNRi	14.67	19.55
SDRi	11.65	20.01

(Task iv) Provide observations on the changes in performance throughout the experiment.
The comparison between the results of the SepFormer model without fine-tuning and after fine-tuning on the training set demonstrates a substantial improvement in performance across both metrics, namely the Signal-to-Interference-plus-Noise Ratio improvement (SISNRi) and the Source-to-Distortion Ratio improvement (SDRi). Initially, without fine-tuning, the SepFormer model achieved a commendable SISNRi of 14.67 and an SDRi of 11.65, indicating effective speech separation with significant enhancement in speech quality and minimal distortion. However, after fine-tuning on the training set, the model's performance notably improved, with the SISNRi increasing to 19.55 and the SDRi rising to 20.01. This substantial enhancement in both metrics signifies the effectiveness of fine-tuning in adapting the SepFormer model more closely to the characteristics of the training data, resulting in improved separation of speech from background noise and interference. The higher SISNRi and SDRi values after fine-tuning indicate clearer and more intelligible separated speech with reduced distortion, demonstrating the superior quality of the fine-tuned SepFormer model in speech separation tasks. Overall, these results underscore the significance of fine-tuning in enhancing the performance of the SepFormer model and its potential for achieving state-of-the-art results in speech separation applications.

NOTE: Please note that all fine-tuning procedures are implemented using the github repositories of the said model.