The Chinese University of Hong Kong, Shenzhen
School of Data Science

MDS5122 / AIR5011
Assignment 2

Due by: 23:59, Dec 5th, 2025

Instructions:

1. You must submit your files on Blackboard. Please upload a PDF file along with your code (excluding large files such as the dataset).
2. Your submission must be clearly answered and well-presented to receive full credit. Please detail how you leverage your acquired knowledge to solve the problem and present your proposed solution step by step. Ensure that your solutions are legible and written in English.
3. The programming language is Python, and your code should include necessary comments so that others can easily follow its logic. Copying answers violates academic integrity.
4. Your report should reflect your individuality and originality. If your answers are inspired by other sources (*e.g.*, the Internet or AI), please maintain academic integrity by including a reference or acknowledgment section in your submitted paper and indicate how these sources inspired you.
5. Late submissions or instances of plagiarism will not be graded.

A. Implementing a Simple Multi-Modal Cross Attention (100 points)

In this assignment, you will implement a simplified version of the TASTE (Text-Aligned Speech Tokenization and Embedding) model proposed by Tseng et al. (2025). The goal is to understand how an attention mechanism can be used to align speech features (audio) with text tokens, and how to reconstruct speech from those aligned embeddings. You will focus on the tokenizer stage of TASTE and integrate it with a pretrained text-to-speech model (CosyVoice) to reconstruct speech audio.

Background

Traditional speech tokenizers (e.g., EnCodec, SpeechTokenizer) produce long speech token sequences independent of text, making it difficult to align speech and language. TASTE addresses this by using an attention-based aggregator to align the length of speech representations with that of the text tokens. In this assignment, you will reproduce this alignment stage and evaluate how it affects downstream speech reconstruction using the CosyVoice model.

1. Dataset Preparation (15 points total)

1.1 Download LibriSpeech (5 pts)

LibriSpeech is a public English speech dataset containing audiobook recordings and transcriptions. Download it from https://www.openslr.org/12

Use train-clean-100.tar.gz for training (100 h) and test-clean.tar.gz for testing (5 h). Each sample includes a WAV file and a text transcript. Write a short Python script to pair each wav file with its transcription line. *You can choose a small subset instead of using the whole dataset for training and evaluation.*

1.2 Preprocessing (10 pts)

- Resample all audio to 16 kHz using torchaudio.
- Store paired data as (audio, transcription) tuples into a .jsonl file.
- Optional: visualize a Mel-spectrogram.

2. Implementing the Text-Aligned Tokenizer and Integration (50 points total)

2.1 Extract Speech and Text Features

Whisper is an open-source speech recognition model developed by OpenAI that maps input audio to text or hidden features. You will use Whisper's encoder to extract speech representations and its tokenizer to obtain text tokens.

2.2 Implement the Attention Aggregator (20 pts)

Design an attention module to align the speech and text representations:

$$Q = v, K = h^{(L)}, V = h^{(l)}$$

and we have

$$z = \text{Attention}(Q, K, V)$$

The output z should have the same sequence length as v, producing text-aligned speech embeddings. This allows the speech embeddings to be compressed or stretched to match the length of text tokens, establishing one-to-one alignment between the two modalities. If you encounter a mismatch between the embedding dimensions of the text and speech features, try to insert a trainable adapter layer (e.g., a linear projection) before feeding them into the attention module to make their dimensions consistent. For example:

$$h' = Wh$$

where W is learnable linear adapters.

2.3 Understanding and Using S3 Units (10 pts)

S3 units are discrete intermediate speech representations used by the CosyVoice model to bridge between text and waveform. Each unit represents a short segment of speech that encodes both phonetic and prosodic information (such as tone and rhythm). Predicting these units accurately allows the model to reconstruct natural-sounding speech via a downstream vocoder. The CosyVoice repository provides tools to extract S3 units from raw audio. Please refer to the official GitHub page for instructions on generating S3 tokens from LibriSpeech samples: https://github.com/FunAudioLLM/CosyVoice and you can download the 300M version of CosyVoice from ModelScope: https://www.modelscope.cn/studios/iic/CosyVoice-300M

Then, use the ground-truth S3 tokens extracted from the dataset as the prediction target. The aggregator output z will be passed to the CosyVoice LLM, and you should minimize the cross-entropy loss:

$$L = \text{CrossEntropy}\left(S3_{predict}, S3_{groundtruth}\right)$$

2.4 Combine Text + Speech Embeddings for CosyVoice LLM (20 pts)

Integrate your aligned speech embeddings into a pretrained CosyVoice model. CosyVoice is an open-source multilingual text-to-speech model that predicts S3 units as intermediate representations. You can download the 300M version from ModelScope: https://www.modelscope.cn/studios/iic/CosyVoice-300M

Integration steps.

a) Obtain the text embedding v from CosyVoice's text encoder.
b) Combine it with your text-aligned speech embedding z: $e_{joint} = v + z$
c) Replace the original text-only embedding with e_{joint} as input to the CosyVoice LLM decoder.
d) Finetune the decoder to predict S3 units using the ground-truth S3 sequences.

Ensure that the input format to the CosyVoice LLM matches its original implementation. In addition to text embeddings, the model also requires several auxiliary prompts such as the speaker embedding and other conditioning features. Refer to the official CosyVoice GitHub documentation for details on how to obtain and prepare these prompts correctly.

3. Analysis and Report (35 points total)

3.1 Hyperparameter Summary & Training Curve (15 pts)

The report should have:

a) Attention heads, hidden size, number of layers (2 pts)
b) Learning rate, batch size, epochs, optimizer (2 pts)
c) Hardware and training time (2 pts)

During training, at each step, you should record both the loss and the predicted S3 token. Please report the following:

a) The training loss curve (4 pts)
b) Report the top-1 S3-token prediction accuracy on the test-clean split. (5 pts)

3.2 Questions (20 pts)

a) Why the Aggregator Aligns Speech to Text Length? (10 pts)
   In this assignment, you implemented a cross-attention-based aggregator where the query is the text embedding, and keys/values come from speech features. Explain why this architecture naturally forces the output length to match the number of text tokens. Additionally, discuss what would happen if the speech features were used as queries instead.
b) Effect of Shallow vs. Deep Speech Features in Aggregation? (10 pts)
   The TASTE paper shows that using the last-layer Whisper representation as keys and shallow-layer representations as values leads to better speech reconstruction. Based on your implementation and training results: Why might shallow-layer features be more suitable as values for reconstructing S3 units, while deep-layer features better capture alignment cues? Support your argument using observations from your loss curve or S3-token accuracy.

Tool Functions

You can use s3.sh to get s3 tokens (speech tokens) and use utt2text_and_feature.py to get text/speech feature.

Submission

- Source code (.py or .ipynb)
- Report (.pdf)
- The requirements.txt of your environment
- A README.md that clearly describes how to run the code of the whole pipeline and reproduce the results. (including data preprocessing, training, and evaluation)

Your report MUST be written using the CVPR 2025 Author Kit in a camera-ready format, with your names, student IDs, and affiliations clearly displayed, and must adhere strictly to CVPR's requirements as if you were submitting to this conference, *e.g.*, a maximum of eight pages, although you may not actually submit it. You must also release and upload your code as a supplement to the Blackboard system, along with a Markdown document that clearly instructs others on how to set up the environment and reproduce your results. If you are confident in your project, you may optionally submit it to any other conference you like. Apart from the requirements stated above, there are no concrete restrictions on how your method should be implemented. If you cannot come up with your own ideas, you can follow the suggestions below to complete this final project.

Suggestions (Not Mandatory):

1. Check TASTE: Text-Aligned Speech Tokenization and Embedding for Spoken Language Modeling for details of the background and model architecture.

Scoring Criteria:

Your submission must contain the following sections:

1. Dataset Preparation **(15 points)**
2. Implementing the Attention-Based Aggregator **(50 points)**
3. Analysis and Report **(35 points)**

We will review and score each part of your paper as if you had submitted it to the CVPR conference. Using your own ideas, conducting ablation studies, performing comparative experiments, and so on means that you have a good chance of earning higher credits.

Some Tips on Experiments and Computational Resources:

0. **Adjusting Hyperparameters:** Consider adjusting hyperparameters, *e.g.*, model size, batch size, to ensure your experiments can be conducted within the anticipated time budget. You may include various techniques to reduce memory usage, such as half-precision training and quantized training. It is advisable to train using a GPU, with the experiment typically requiring a minimum of 12GB of GPU memory.
1. **NVIDIA GPUs:** If you have an NVIDIA GPU on your local machine, I recommend installing the CUDA version of PyTorch. This will allow the training process to utilize the GPU for acceleration.
2. **Online Free GPUs:** If you only have access to a CPU, you can explore online computational resources. *Google Colab* is an excellent platform for running deep learning experiments online. It is completely free to use, just sign in with a Google account. Similarly, you might consider *Kaggle* and *Tianchi*, both of which offer free GPU resources.
3. **University Resources:**Additionally, our school's high-performance computing platform is available, although registration is required and it is not free.
4. **Third-Party GPU Providers:** There are also several third-party GPU cloud providers, such as GpuShare and AutoDL, which offer affordable options, though they are not free.

If you have any issues, please don't hesitate to contact our TAs.