

Deep Learning for Music Analysis and Generation

HW2

GAN-based Mel-Vocoder



Yi-Hsuan Yang Ph.D.

yhyangtw@ntu.edu.tw

The Homework

- **GAN-based Mel-Vocoder**
- The same as HW1
 - Submit your **codes**, **report**, and **prediction result** for the test set
 - Done individually
 - No cheating
 - Evaluation
 - Report (50%), accuracy (50%)
 - Choose the **best three** for a quick oral presentation (5 mins each) during the class (bonus points offered)

Timeline

- **W6 – 10/12** (Thursday): Announcement of HW2
- **W9 – 11/01** (Wednesday **23:59pm**): **Deadline**
 - Late submission: 1 day (-20%), 2 days (-40%), after that (-60%)
- **W9 – 11/02** (Thursday): Announcement of HW3
- **W10 – 11/09** (Thursday): Oral presentation of HW2
 - I will be in touch with the three presenters two days ahead (i.e., on 11/07)
 - No need to prepare fancy slides

Training/Validation Dataset: M4Singer

<https://github.com/M4Singer/M4Singer>

- 700 Chinese pop songs recorded by 20 professional Chinese singers
- Meant for research on singing voice synthesis

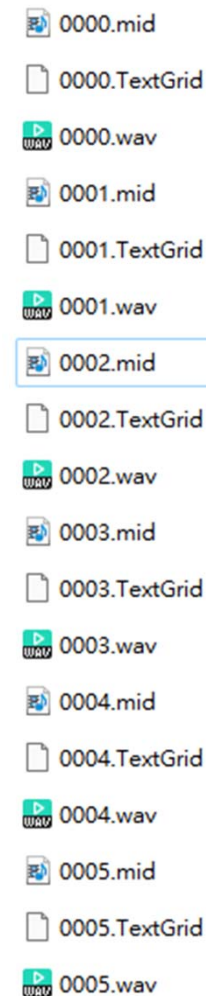
Corpus	Language	#Hours	#Singers	Alignment	Score
NUS-48E [12]	English	1.91	12	✓	✗
NHSS [39]	English	7	10	✓	✗
JVS-MuSiC [44]	Japanese	2.28	100	✗	✗
Tohoku Kiritan [30]	Japanese	1	1	✓	✓
PopCS [26]	Chinese	5.89	1	✗	✗
OpenSinger [17]	Chinese	50	66	✗	✗
Opencpop [48]	Chinese	5.25	1	✓	✓
M4Singer (Our)	Chinese	29.77	20	✓	✓

Training/Validation Dataset: M4Singer

<https://github.com/M4Singer/M4Singer>

- No accompaniment
- Short utterances (a few words) between 3–10 seconds

Data Segmentation: After the annotation, we segment the audio into smaller fragments to facilitate training, while the alignment and score are correspondingly segmented into sentence-level segments. By leveraging the manual alignment results, we set a threshold for the unvoiced region as the primary condition for performing the segmentation process. And the voiced region is also constrained with the maximum length as the secondary condition. By doing so, almost all sentences are between 3 and 10 seconds. Finally, 20,942 utterances with a total duration of around 29.77 hours are obtained.



0000.mid
0000.TextGrid
0000.wav
0001.mid
0001.TextGrid
0001.wav
0002.mid
0002.TextGrid
0002.wav
0003.mid
0003.TextGrid
0003.wav
0004.mid
0004.TextGrid
0004.wav
0005.mid
0005.TextGrid
0005.wav

Test Dataset: ???

- Will give you the Mel-spectrograms and ask you to generate the waveforms

Train/Validation split

- Predefined by the TAs
 - Singer-level split
 - Training & validation sets: released on Day 1
 - Test set: released on Day 1
 - You can get to know how your vocoder performs by simply listening to the result

Mel-Spectrogram Configuration

- Sampling rate, window size, hop size
- Code will be provided by the TA

Evaluation

- Report the following objective metrics (used in [1]) on the **validation** set
 - **MSSTFT**: multi-resolution STFT loss (reconstruction loss in the magnitude STFT domain)
 - **MAE-f0**: mean absolute errors in F0, in log scale (cent)
 - **Fréchet Audio Distance** (FAD)
 - https://github.com/google-research/google-research/tree/master/frechet_audio_distance
 - <https://arxiv.org/abs/1812.08466>
- Code will be provided by the TA
- We will choose **one** of the metrics for ranking your results on the **test** set

[1] Wu et al, “DDSP-based singing vocoders: A new subtractive-based synthesizer and a comprehensive evaluation,” ISMIR 2022

Objective Evaluation Metric: MSSTFT

- Reconstruction loss in the magnitude STFT domain
- Need a *reference* audio (i.e., the groundtruth waveform)

For monophonic instrumental sounds, Engel *et al.* [23] showed it effective to use the multi-resolution STFT (MSSTFT) loss as the reconstruction loss for training. This loss considers the difference between the magnitude spectrograms of the target and synthesized audio, denoted as \mathbf{S}_j and $\widehat{\mathbf{S}}_j$ below, for J different resolutions.

$$l_{\text{MSSTFT}} = \sum_{j=1}^J \|\mathbf{S}_j - \widehat{\mathbf{S}}_j\|_1 + \|\log(\mathbf{S}_j) - \log(\widehat{\mathbf{S}}_j)\|_1. \quad (6)$$

Ref 1: Wu et al, “DDSP-based singing vocoders: A new subtractive-based synthesizer and a comprehensive evaluation,” ISMIR 2022

Ref 2: Engel et al, “DDSP: Differentiable digital signal processing,” ICLR 2021

Required Runs

- Try the **Griffin & Lim** algorithm and **report** its result on the ***validation set***
- **Train any GAN-based Mel-vocoder(s)** on your training set, and **report** its result on the ***validation set***
 - It's fine to use open source code
 - Better to retrain the model on the training set instead of use a pretrained model shared on the Internet as is, for such a pretrained model probably won't work well on your validation set (many Mel-vocoders are not universal)
- Pick **only one** Mel-vocoder you trained that you believe to perform the best, and **submit its result on the *test set***

Additional Data

- You can **only use the training set** to train the Mel-Vocoder
 - Not the validation set
 - **NO** other datasets (for otherwise we won't know whether the performance comes with advantages in the GAN architecture design or the additional data)

GAN Training Can be Slow

- NOTE: GAN training can **take days** to converge; better to start earlier
- TA's experience:
 - **1-2** days using 1080Ti (VRAM 12GB) not yet converge but good enough
 - Can let it train longer

Bonus

- Try models that use **F0** as an input
 - Tools to compute the F0: YIN, WORLD, or CREPE
 - (However, models for F0 estimation may take magnitude STFT as input; you need to find one that takes Mel-spectrogram as input)
- **Loop generation**
 - Train a generator using loops that generates the Mel-spectrograms
 - Train a Mel-vocoder using the same set of loops
 - Reference
 - Hung et al, “A benchmarking initiative for audio-domain music generation using the FreeSound loop dataset,” ISMIR 2021
 - Yeh et al, “Exploiting pre-trained feature networks for generative adversarial networks in audio-domain loop generation,” ISMIR 2022
 - Ask the **TA** for a copy of the *preprocessed* FreeSound loop dataset

The Report

- The same as HW1
 - Cover page: your name, student ID etc
 - **Novelty highlight** (one page; *optional*): what's special about your work?
 - **Methodology highlight** (one page): how did you make it?
 - Or, list the *attempts* you have made
 - **Result highlight** (one page): result on your validation set
 - **Findings highlight** (one page): main takeaways of your study
 - **Details of your approach** (multi-pages)
 - If you use open source code, you may want to read some of the associated paper(s) and summarize your understanding of the paper(s) (e.g., why it works)
 - **Result analysis & discussion** (multi-pages)

HW2 scoring and submission

HW2 TA: 葉軒瑜(Fischer Yeh)

Office hour: Thu.6 13:20 ~ 14:05 @BL505

When you encounter problem:

1. Check out all course materials and announcement documents
2. Use the power of the internet and AI
3. Use “Discussions” on NTU COOL
4. Email me fish90510@gmail.com or come to office hour

Rules

- Don't cheat
- Use Gan-base vocoder methods
- Can use public codes with cite in report
- Can use pretrained model
- Cannot add extra data

M4singer (data release on NTU Cool assignments)

Singer#Song

---- m4singer ----

- Alto-1#newboy
- Alto-1#小幸运
- Alto-1#云烟成雨
- Alto-1#匆匆那年
- Alto-1#左边
- Alto-1#打错了
- Alto-1#怀念
- Alto-1#我喜欢上你时的内心活动
- Alto-1#走马
- Alto-1#闷
- Alto-1#奇妙能力歌
- Alto-1#抱歉抱歉
- Alto-1#空空
- Alto-1#美错

ClipNum.wav

- 0000.mid
- 0000.TextGrid
- 0000.wav
- 0001.mid
- 0001.TextGrid
- 0001.wav
- 0002.mid
- 0002.TextGrid
- 0002.wav

M4singer

Provided

- Sample rate: 48000Hz
- Ext: wav
- Channel: Mono
- Length: one line

Testing

- Mel spectrogram
- Sample rate: 22050Hz
- Channel: Mono
- Length: under 30s

Data Split

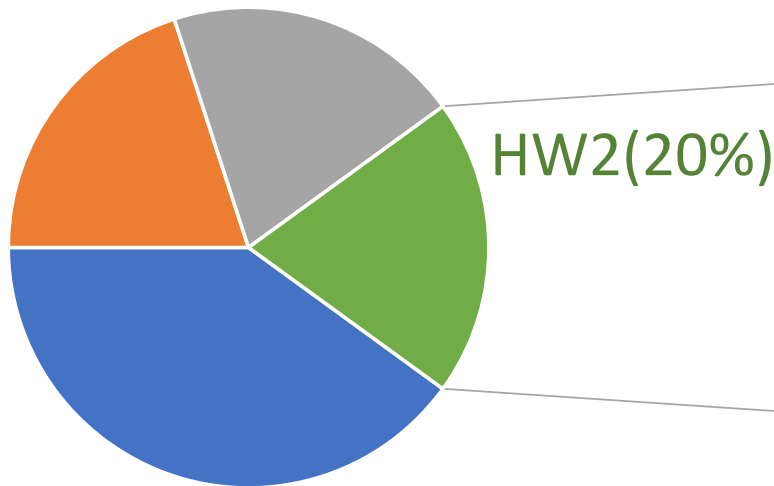
- m4singer:
Train your vocoder using only this data.
- m4singer_valid:
Validate your vocoder using this data in your experiment and report
- testing_mel:
The Mel spectrogram to be reconstructed into audio files

Scoring

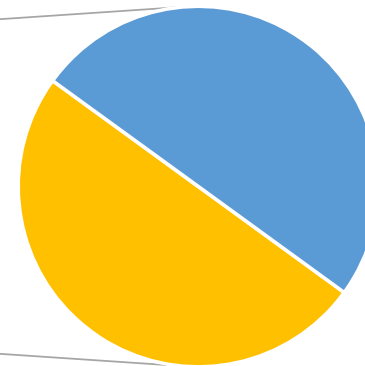
- HW2 accounts for 20% of the total grade.

- In HW2, Report (50%), recover audio quality(50%)

Semester Grades



Quality(50%)



Report(50%)

Report (same as HW1)

- Write with PPT or PPT-like format (16:9)
- Upload studentID_report.pdf (ex: r12345678_report.pdf)
- Please create a report that is clear and can be understood without the need for oral explanations.
There is **no specific length requirement**, but it should clearly communicate the experiments conducted and their results. Approximately **10 pages** is a suggested standard, but not a strict limitation.

Code (same as HW1)

- Upload all your source code to a cloud drive, open access permissions, and then upload the link to the NTU Cool assignment HW2_report in comments, as well as include it on the first page of the report.
- We will randomly select several classmates' code to run inference on your model, so please ensure that the files you upload include trained model and can successfully execute the entire inference process.
- Don't upload: training data, testing data, preprocessed data, others model, cache file

Code (same as HW1)

- You will need to **upload requirements.txt**

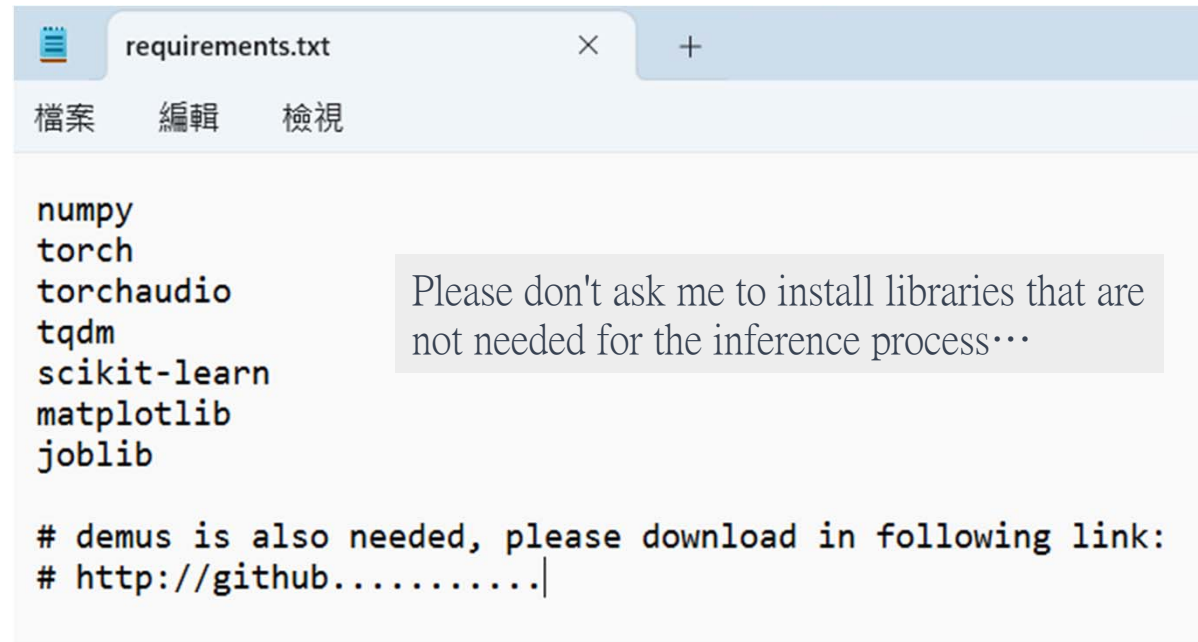
I will run:

```
pip install -r requirements.txt
```

to install all library you need to run inference.

If you have used third-party programs that cannot be installed directly via 'pip install,' please use '#' to write the installation URL and method in a text file.

Example

A screenshot of a text editor window titled 'requirements.txt'. The window has a light blue header bar with a file icon, the title, and window controls (close, maximize, and a plus sign for new file). Below the header is a menu bar with three items: '檔案' (File), '編輯' (Edit), and '檢視' (View). The main text area is white and contains a list of Python packages: 'numpy', 'torch', 'torchaudio', 'tqdm', 'scikit-learn', 'matplotlib', and 'joblib'. Below this list, there is a comment in Chinese: '# demus is also needed, please download in following link:' followed by a partially visible URL '# http://github.....'. A semi-transparent grey box with rounded corners is overlaid on the right side of the text area, containing the text 'Please don't ask me to install libraries that are not needed for the inference process...'.

```
requirements.txt  
× +  
檔案 編輯 檢視  
  
numpy  
torch  
torchaudio  
tqdm  
scikit-learn  
matplotlib  
joblib  
  
# demus is also needed, please download in following link:  
# http://github.....|  
  
Please don't ask me to install libraries that are  
not needed for the inference process...
```


Code (same as HW1)

- You will also need to **upload README.txt or README.pdf** to guide me on how to perform inference on your model. I will use data in the same format as the test data, and the results must match exactly with the predictions you uploaded.
- Your inference process should allow me to choose the test data folder path as well as specify the filename and path for the output prediction.csv.

Mel spectrogram

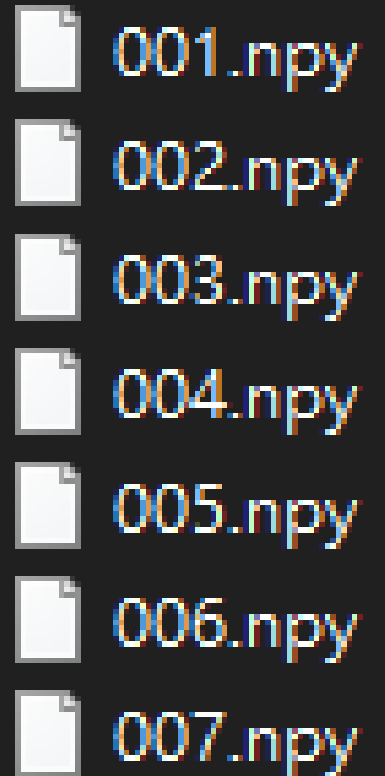
- Sample code on NTU COOL named [Melspectrogram.py](#)
(I used this program to obtain the testing Mel spectrogram.)

- Parameter setting:

num_mels=80	sampling_rate=22050
n_fft=1024	fmin=0
hop_size=256	fmax=8000
win_size=1024	

Testing

- There are 30 testing Mel spectrogram.
- File name: ID.npy
- `mel = np.load('ID.npy')`



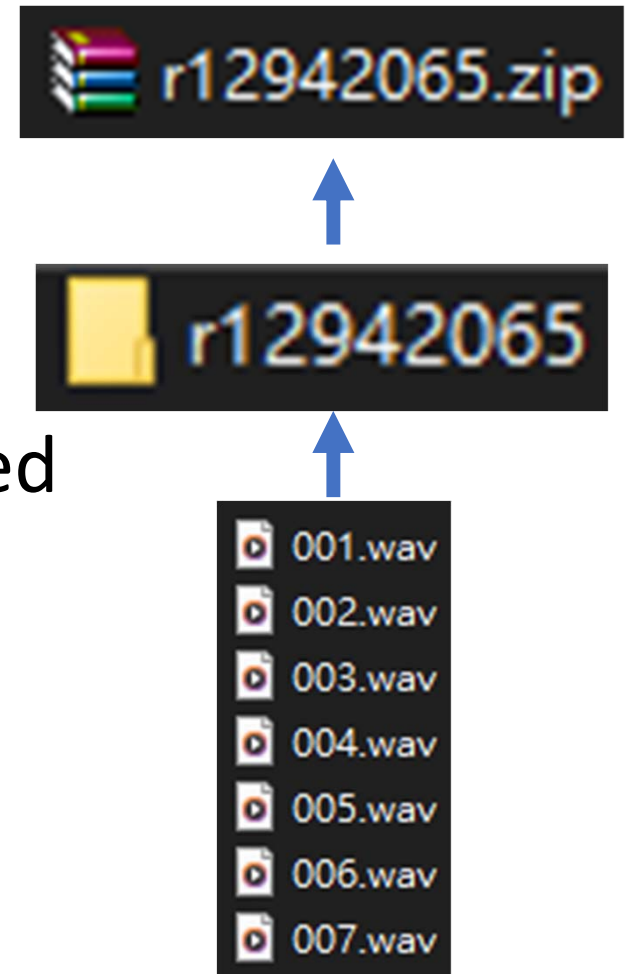
Testing submission

(1) Audio Format:

001.wav, 002.wav, ..., 030.wav

(2) Place all audio files into a folder named 'studentID'.

(3) Compress the folder into a file named 'studentID.zip'.



Evaluation

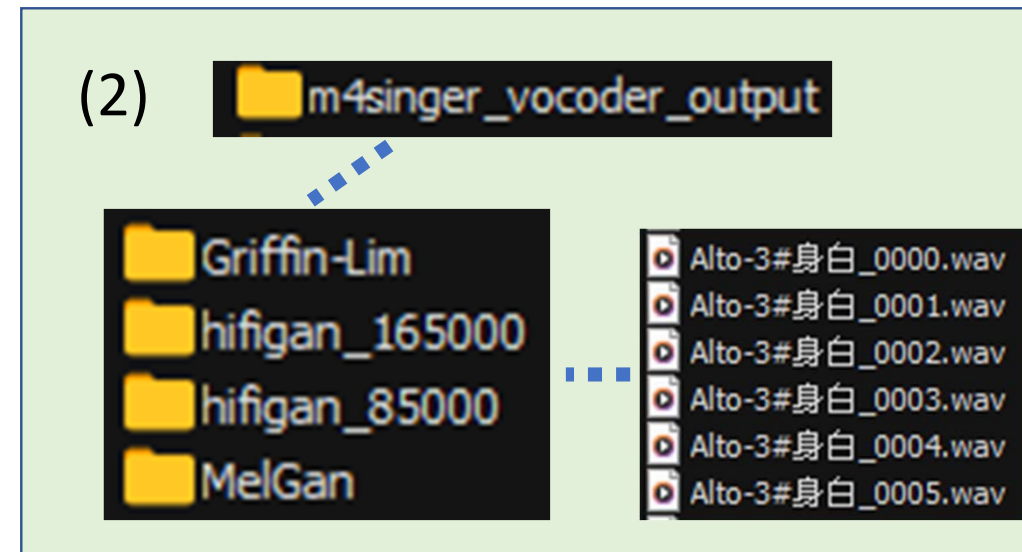
- Use the [Vocoder_eval](#) on NTU COOL
- Usage:

(1) `pip install -r requirements.txt`

(2) Place the output from all different model vocoders, with each model's results in a separate folder, into the same directory.

(3) Set answer audio path and output path in `evaluate.py`

(4) `python evaluate.py --model MODEL`



(3)

```
129     gt_dir = '/path/to/m4singer_valid'
130     synth_dir = '/path/to/your/vocoder_output_dir'+a.model

80     y_path = os.path.join(gt_dir, wav_path.split('_')[0]+'/' + wav_path.split('_')[1])
```

Evaluation Metrics

- M-STFT

Auroloss: <https://github.com/csteinmetz1/auraloss>

- Log2f0_mean

Crepe: <https://github.com/marl/crepe>

```
"M-STFT": 1.1419245269563463,  
"log2f0_mean": 0.16598119455312405,  
"FAD": 0.5688181666613907
```

- FAD

ref:

https://github.com/haoheliu/audioldm_eval/tree/main/audioldm_eval

Coding advise

- Training a vocoder can be **time-consuming**, so it's a good idea to start early and let it train in the background.
- Make sure your program can **save checkpoints or files** during both training and validation to prevent the loss of progress in case of unexpected program interruptions, avoiding the need to start from scratch.
- Pay attention to the sample rate handling and the details of the Mel spectrogram.
- During training, it's important to have a method to check the progress of the training and preferably listen to some generated audio files. (Initially, these files may contain noise, but after approximately 24 hours, you should be able to hear human voices.)

ALL things you need to do before 11/1 23:59

- HW2_report
 - StudentID_report.pdf
 - Cloud drive link
 - README.txt or README.pdf
 - Requirements.txt
 - Codes and model to run inference
 - Others codes
- HW2_prediction
 - StudentID.zip
 - StudentID
 - 0xx.wav

