2023 November 2

Deep Learning for Music Analysis and Generation

Singing Voice Generation

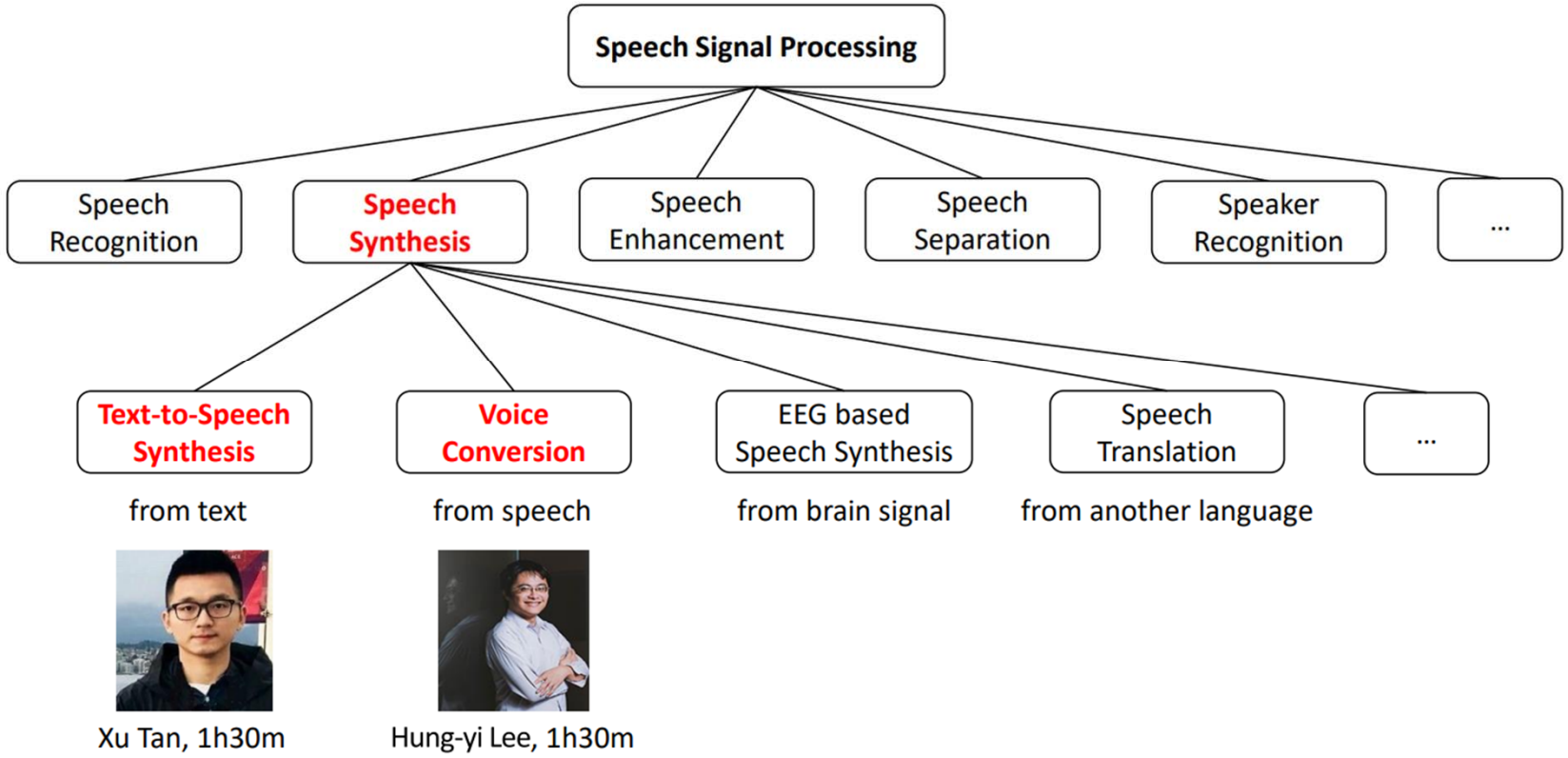
(text+MIDI → singing audio)

Yi-Hsuan Yang Ph.D.

yhyangtw@ntu.edu.tw

Reference 1: INTERSPEECH 2022 Tutorial

https://github.com/tts-tutorial/interspeech2022



Reference 2: ISMIR 2022 Tutorial

https://github.com/lukewys/ISMIR2022-tutorial

https://youtu.be/7U-zDL5con8?si=aKKAx6fB6ij44cko&t=4732

T3(M): Designing Controllable Synthesis System for Musical Signals

Outline

- **Singing voice processing: Historical review**
- Neural text-to-speech synthesis
- Neural singing voice synthesis
- Build your SVS model

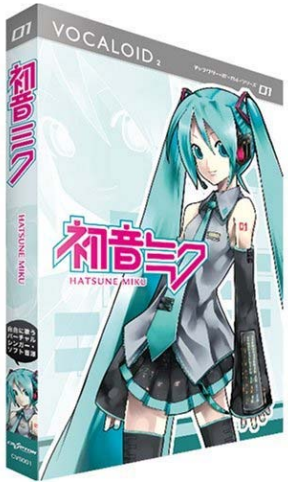
Reference: ISMIR 2015 Tutorial

http://ismir2015.uma.es/docs/ISMIR2015tutorial_Singing.pdf

- All popular music cultures around the world use singing
- The singing voice is the most expressive of all musical instruments
- Singing voice produces both sound and words
- Various commercial applications

Commercial Applications

- Automatic pitch correction (e.g. auto-tune)
- Singing skill evaluation for Karaoke
- Query-by-humming (e.g. soundhound)
- Singing synthesis (e.g. vocaloid)

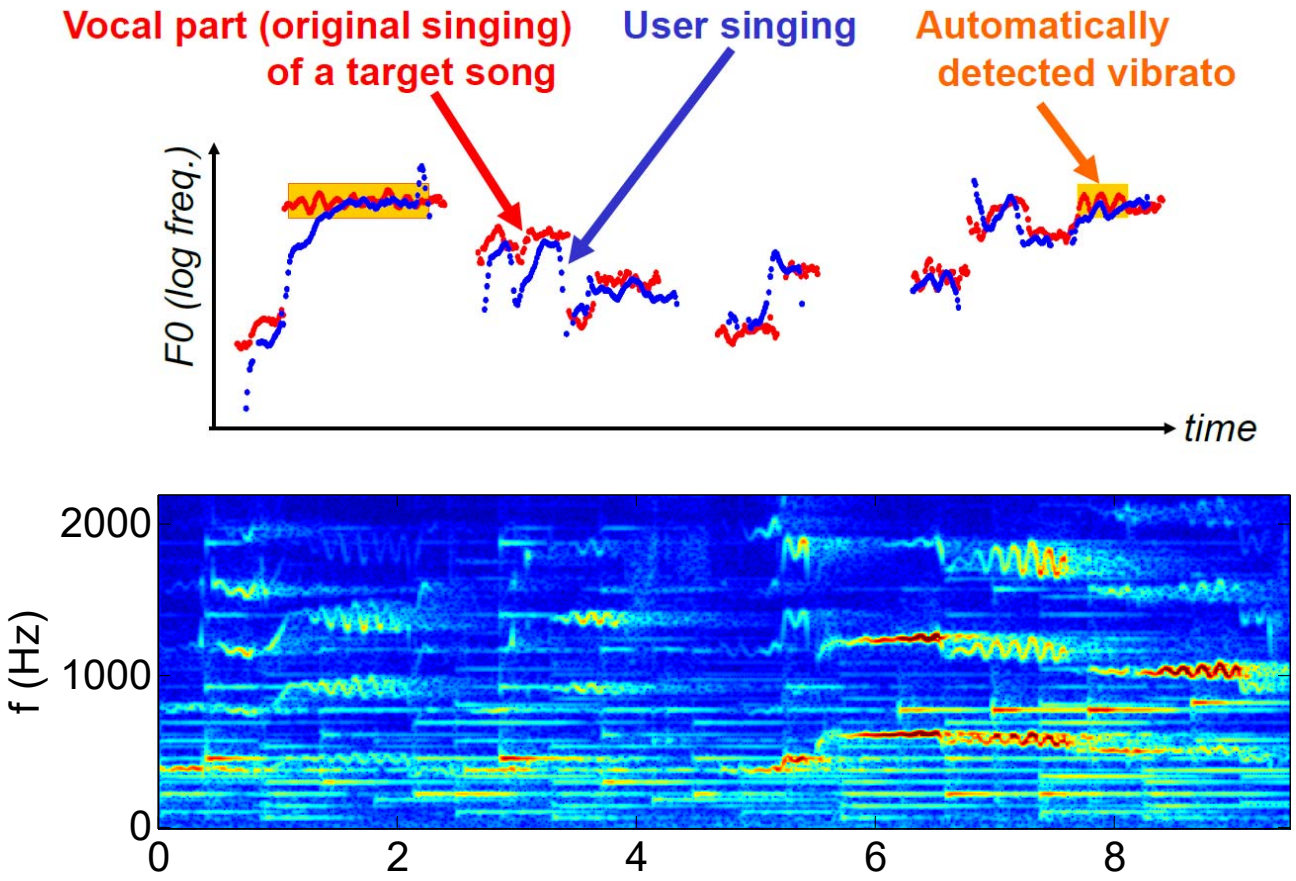


Songrium

https://songrium.jp/

Ref: Kajita et al, “Songrium: Browsing and listening environment for music content creation community,” SMC 2015

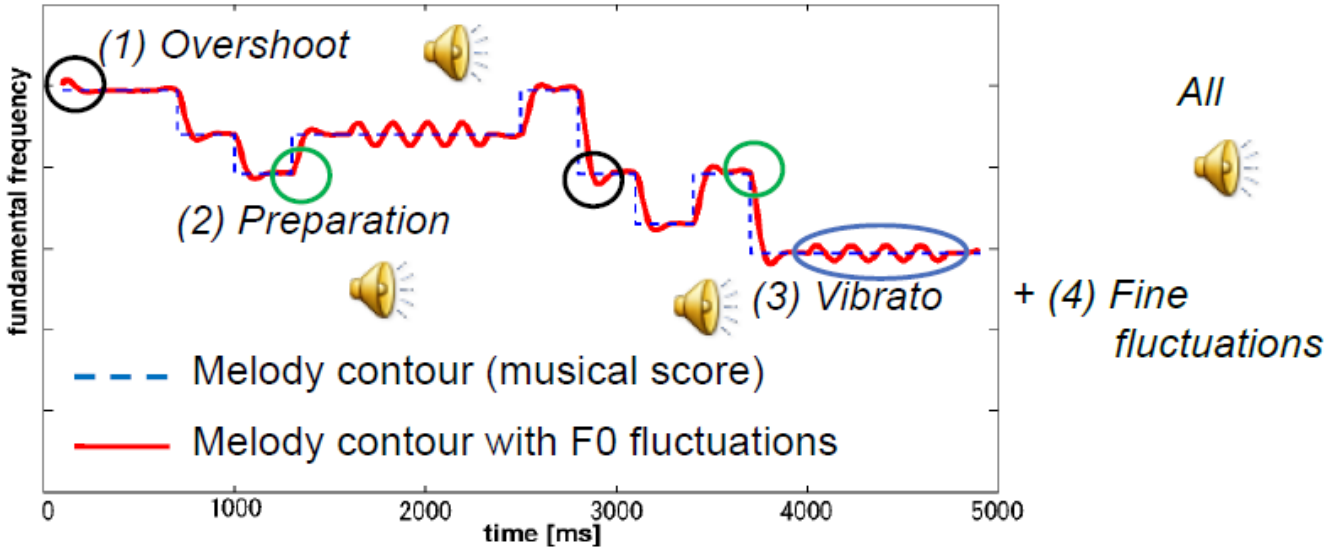
Singing Voice Analysis



Ref: Nakano et al, “MiruSinger: a singing skill visualization interface using real-time feedback and music CD recordings as referential data,” ISM 2007

Speech to Singing Synthesis

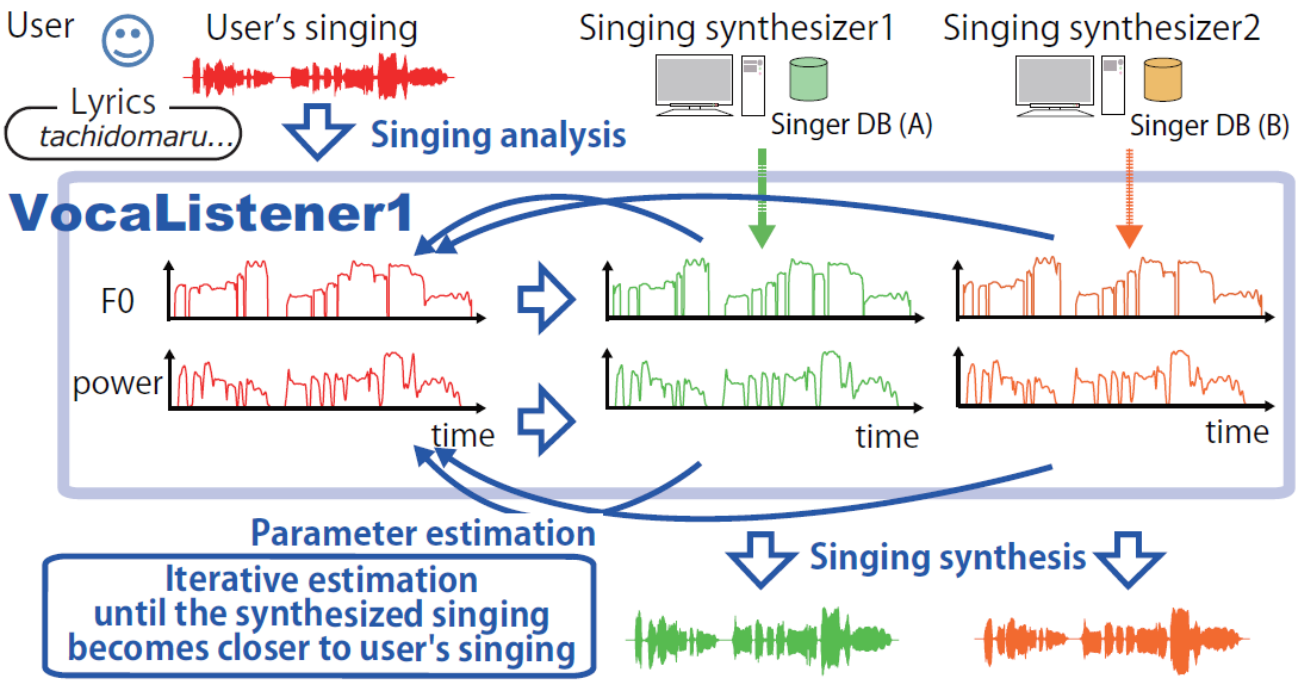
- Convert a speaking voice to a singing voice by changing 1) F0, 2) phoneme duration, and 3) singing formant
- Add four types of F0 fluctuations on musical notes



Ref: Saitou et al, "Speech-to-singing synthesis: converting speaking voices to singing voices by controlling acoustic features unique to singing voices," WASPAA 2007

Singing Voice Conversion

https://staff.aist.go.jp/t.nakano/VocaListener/



Ref: Nakano et al, "VocaListener2: A singing synthesis system able to mimic a user's singing in terms of voice timbre changes as well as pitch and dynamics," ICASSP 2011

Singing Voice Beautifier

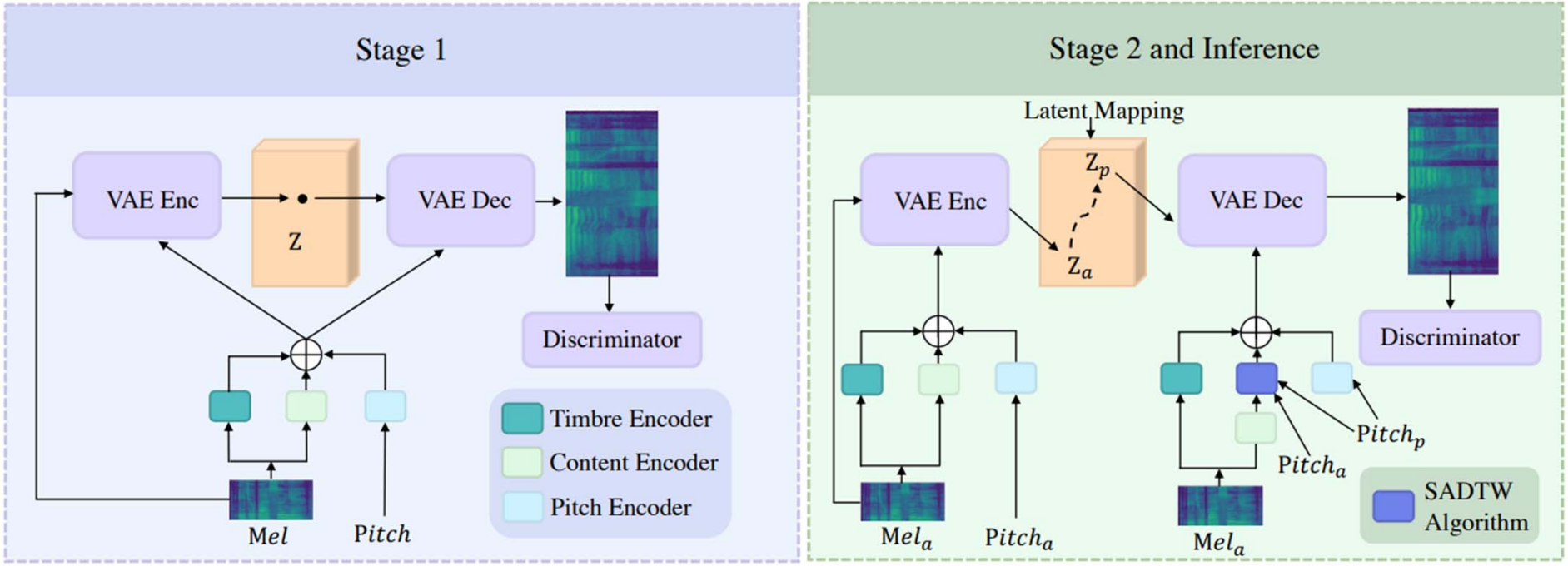


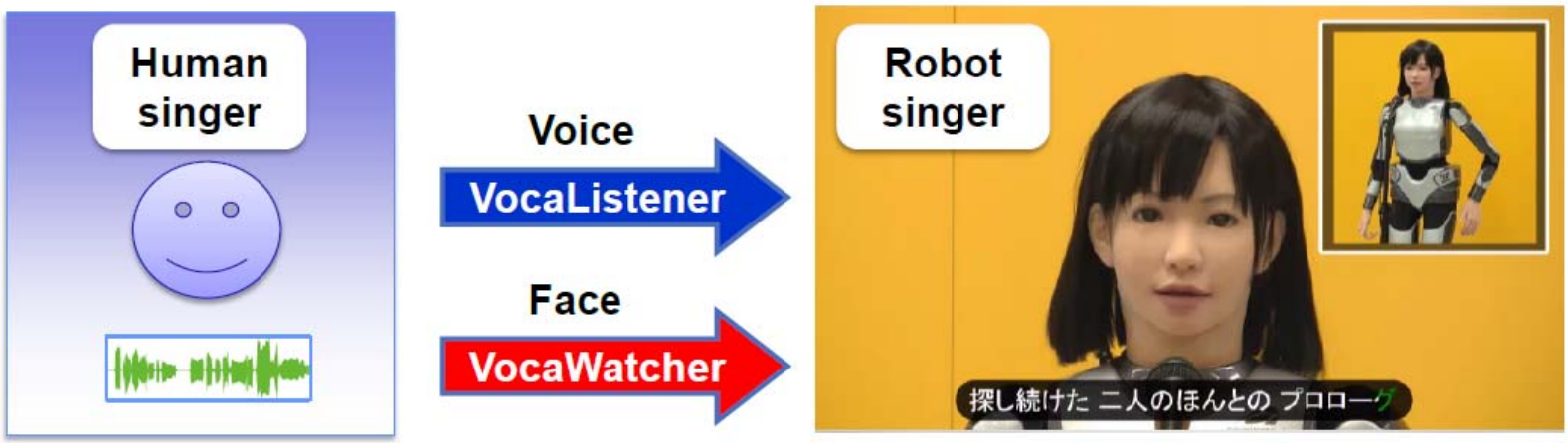
Figure 1: The overview of NVSB. The training process consists of 2 stages, and the second stage shares the same pipeline with the inference stage. "VAE Enc" means the encoder of CVAE; "VAE Dec" means the decoder of CVAE; "Mel" means the mel-spectrogram; "**z**" means the latent variable of the vocal tone; the "a"/"p" subscript means the amateur/professional version.

Ref: Liu et al, "Learning the beauty in songs: Neural singing voice beautifier," ACL 2022

Humanoid Robot Singer

https://staff.aist.go.jp/t.nakano/VocaWatcher/

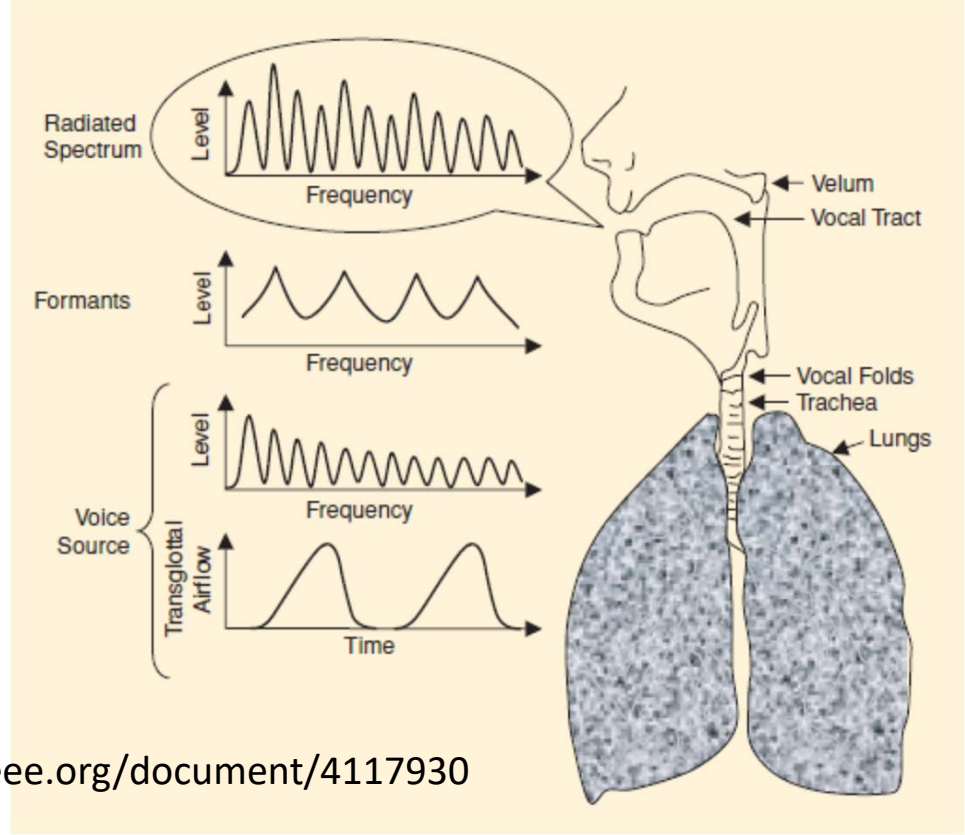
- Imitating a human singer
  - imitate vocal expressions to synthesize singing voices
  - imitate facial expressions to generate robot motions



Ref: Kajita et al, "VocaWatcher: Natural singing motion generator for a humanoid robot," IROS 2011

Singing Voice Synthesis in the Past

- Perry R. Cook, "**Singing Voice Synthesis: History, Current Work, and Future Directions**," *Computer Music Journal*, Vol. 20, No. 3 (Autumn, **1996**), pp. 38-46 (9 pages)
  https://www.jstor.org/stable/3680822?seq=6



https://ieeexplore.ieee.org/document/4117930

Singing Voice Synthesis in the Past

- **Vocaloid** by Xavier Serra
  https://en.wikipedia.org/wiki/Xavier_Serra

Daisy was a collaboration project with Yamaha. The aim of the project was to develop a singing voice synthesizer in which the user would input the lyrics and the notes of a vocal melody and obtain a synthetic performance of a virtual singer. To synthesize such performance the system concatenates a chain of elemental synthesis units. These units are obtained by transposing and time-scaling samples from singers databases. These databases are created out of recording, analyzing, labeling and storing singers performing in as many different musical and phonetic contexts as possible.

Based on Daisy's research, Yamaha released a product named Vocaloid. Vocaloid was presented at the 114th Audio Engineering Society (AES) Convention in Amsterdam in March 2003 and had a big media impact leaded by The New York Times article "Could I Get That Song in Elvis, Please?". The Vocaloid synthesizer was Nominee for the European IST (Information Society Technologies) Prize 2005 and is the choice of leading artists including Mike Oldfield.

The research carried out for Daisy is mainly described in Bonada and Loscos (2003), Bonada et al. (2003), and Bonada et al (2001).

Source:
Alex Loscos, *Spectral Processing of the Singing Voice*, PhD Thesis, UPF, Spain (supervised by Xavier Serra)

Vocaloid

- Vocal+android
  （歌唱）（人形機器人）
  - A singing voice synthesizer related product of Yamaha, first released in 2004
  - User keys in lyrics and MIDI through an editor to create singing voice

Hatsune Miku

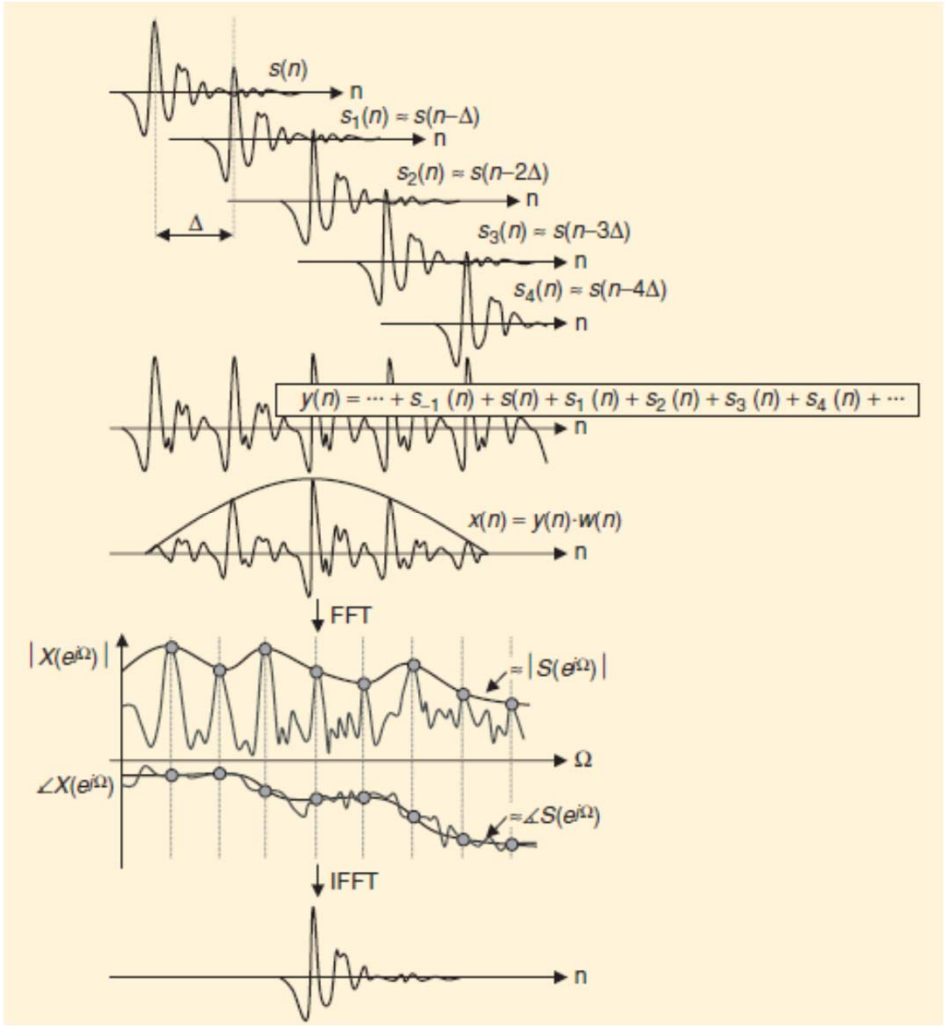
https://www.youtube.com/watch?v=jhl5afLEKdo

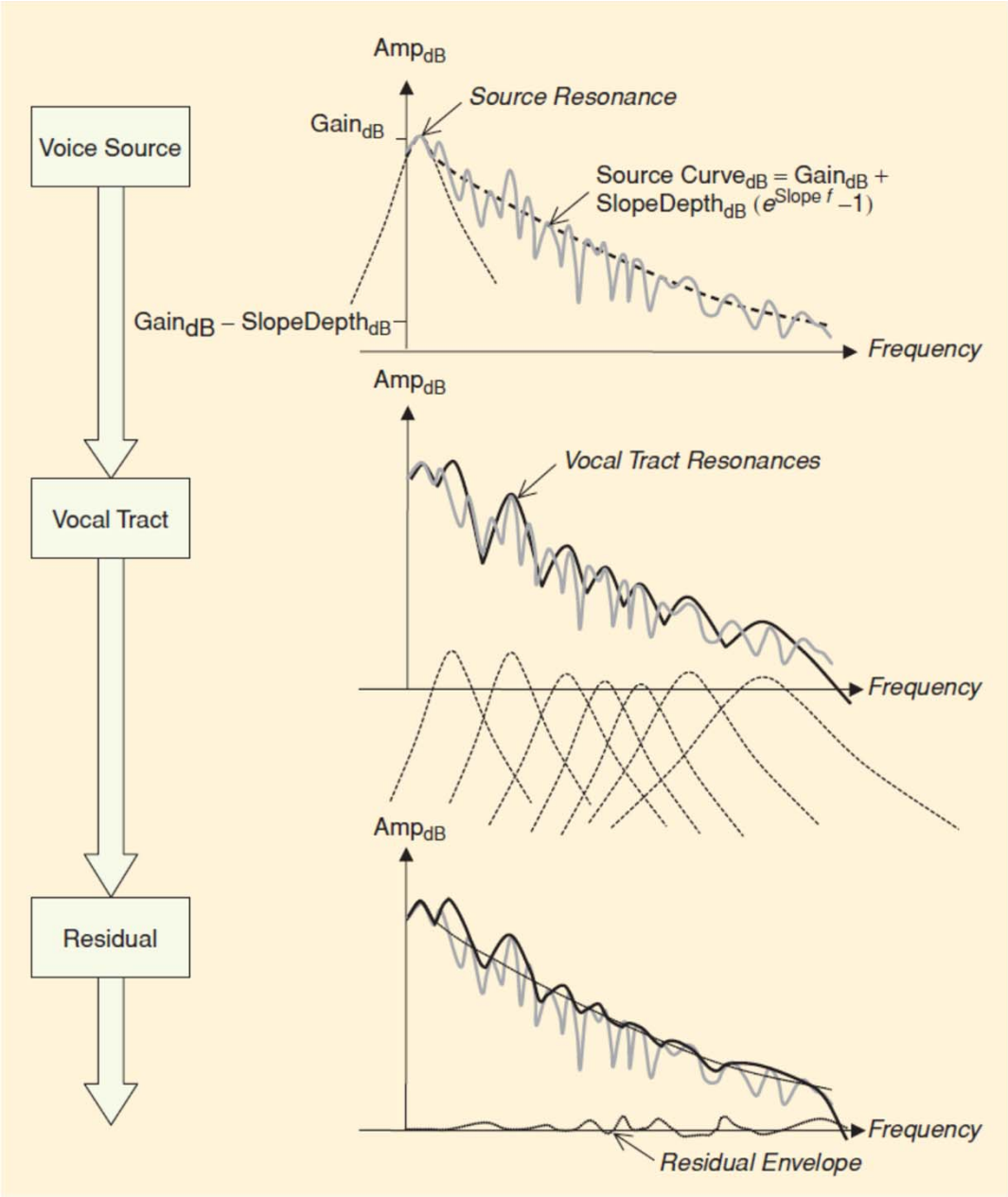
【Hatsune Miku】 World is Mine / ryo（supercell）【初音ミク】

(Bonada and Loscos, 2003) “Sample-based singing voice synthesizer by spectral concatenation”

```The singing synthesis system we present generates a performance of an artificial singer out of the musical score and the phonetic transcription of a song using a frame-based frequency domain technique. This performance mimics the real singing of a singer that has been previously recorded, analyzed and stored in a database, in which we store his voice characteristics (phonetics) and his low-level expressivity (attacks, releases, note transitions and vibratos). **To synthesize such performance the systems concatenates a set of elemental synthesis units (phonetic articulations and stationeries). These units are obtained by transposing and time-scaling the database samples**. The concatenation of these transformed samples is performed by spreading out the spectral shape and phase discontinuities of the boundaries along a set of transition frames that surround the joint frames. […]```

Source: http://mtg.upf.edu/node/322
```





https://ieeexplore.ieee.org/document/4117930

Singing Voice Synthesis in the Past

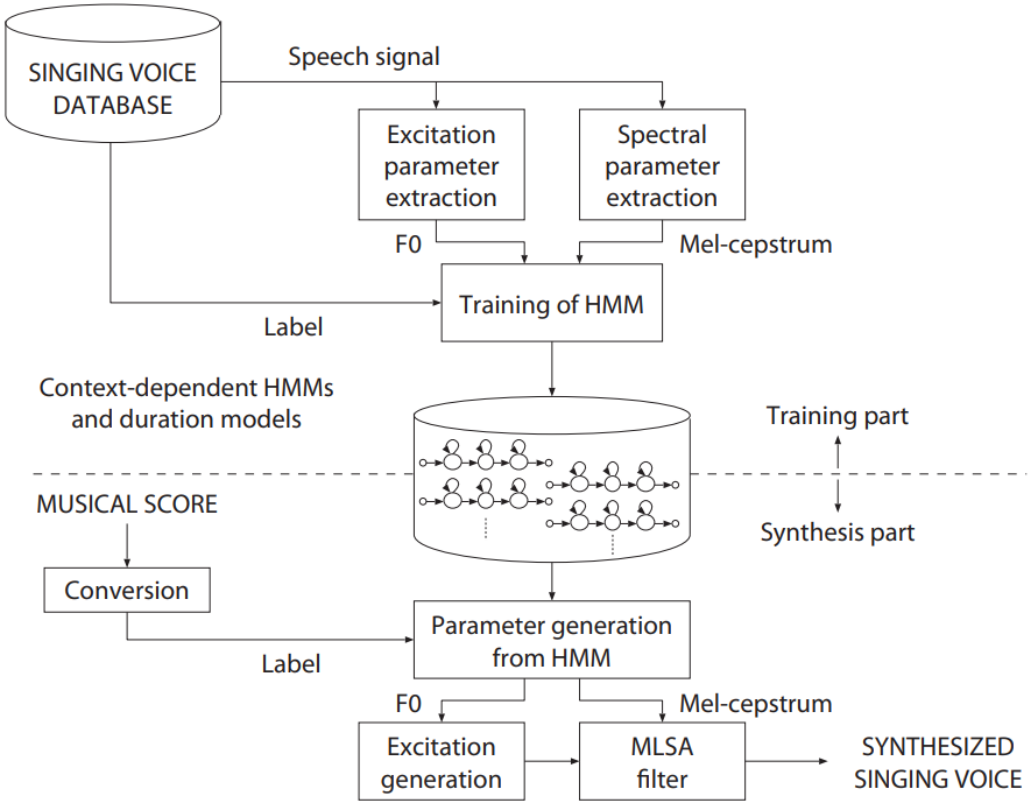
1. **Human knowledge**
2. **Lot's of signal processing**
3. **Few parameters**

- Great success,
  but characteristic
  robotic sound

Singing Voice Synthesis in 2010

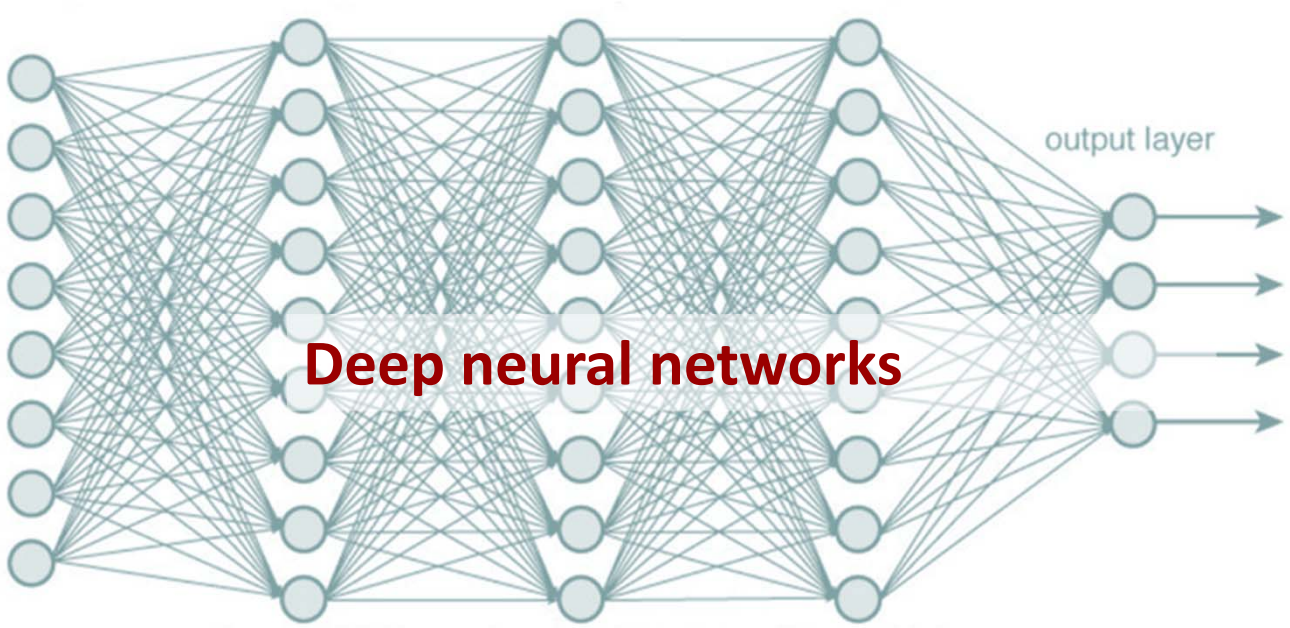
- K. Oura et al., “Recent development of the **HMM-based** singing voice synthesis system—Sinsy,” *Proc. ISCA Workshop on Speech Synthesis,* 2010

1. **Human knowledge**
2. **Lot’s of signal processing**
3. **Few parameters**



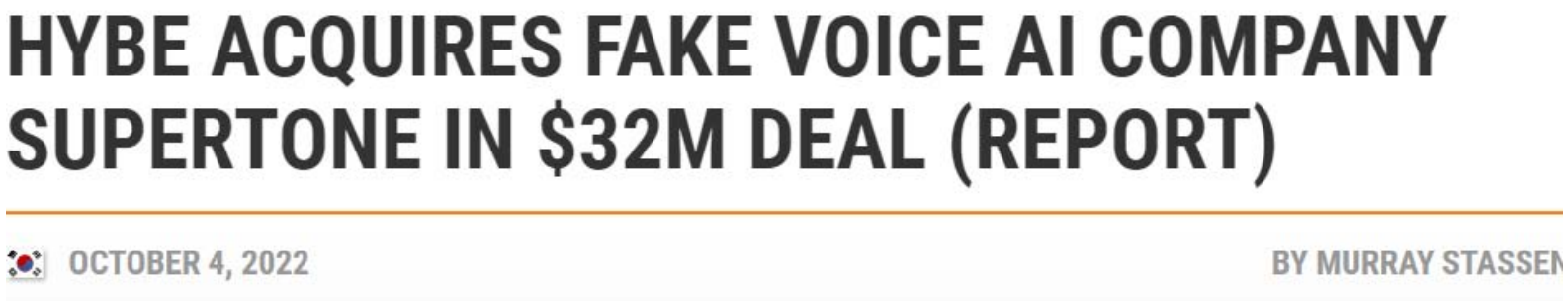
DL: It's about Learning Input/Output Relationships

- Input: **lyrics + melody notes** (per sentence)
- Output: corresponding **singing audio**



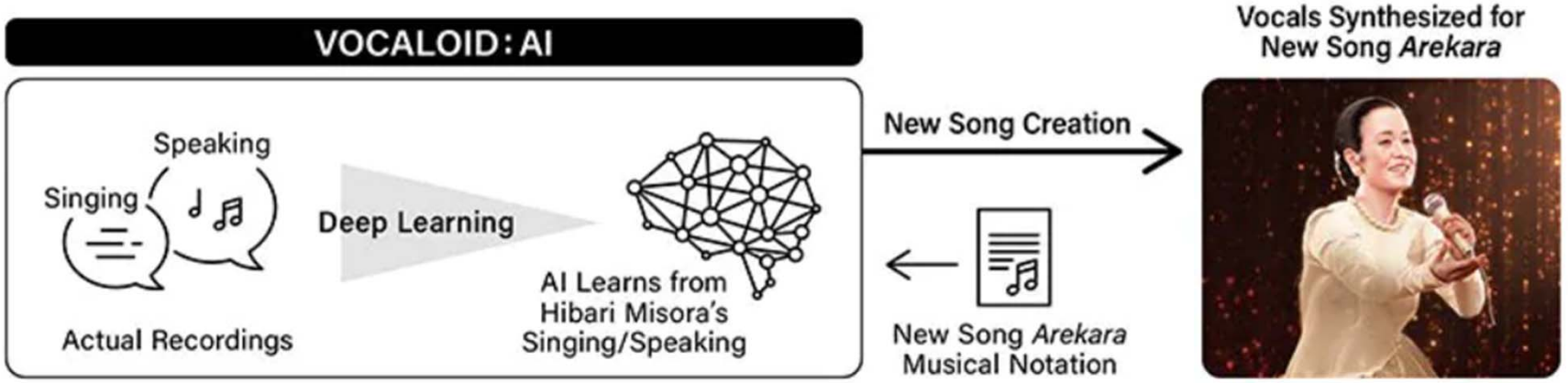
Research + Data

- **Supertone:** https://www.youtube.com/watch?v=MPG4_scT798
  - A startup co-founded by Prof. Kyogu Lee at Seoul National University
  - Around **30 PhD students** in his lab (as of 2023)
  - Tones of data ("We trained the backbone model on 10,571 hours (speech: 10,092 hours, singing: **479 hours**) of proprietary 44.1 kHz audio recordings composed of 6,176 speakers and **624 singers**")
  - Company acquired by HYBE (home to BTS) in Oct. 2022


HYBE ACQUIRES FAKE VOICE AI COMPANY SUPERTONE IN $32M DEAL (REPORT)

OCTOBER 4, 2022 BY MURRAY STASSEN

VOCALOID 6 & Synthesizer V



VOCALOID6 uses VOCALOID:AI, an AI-based technology that makes it possible to generate even more natural-sounding and highly expressive singing voices.

Synthesizer V AI is the next generation of the Synthesizer V vocal synthesis engine, and update to the Synthesizer V Studio software developed by Dreamtonics. The software was unveiled October 30, 2020 in a press release alongside voice database Saki AI and the announcement of Koharu Rikka.[1][2]

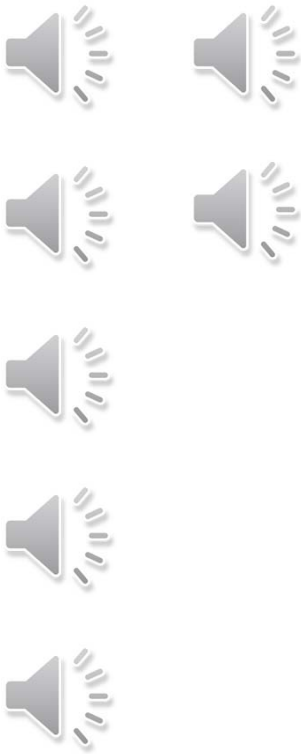
Our Own Experience

2019/08
AI phantom

2020/02
bad articulation

2021/06
intelligible

2021/10
AI Sandee v0



2022/06
better quality

2022/12
close to ready

2023/03

Singing Synthesis at the Taiwan AI Labs

https://music.yating.tw/ai-music/beta/voxai

- For *pro* users
  - Given lyrics and melody notes (MIDI), generate singing audio

Singing Synthesis at the Taiwan AI Labs

https://music.yating.tw/ai-music/beta/vocal

- For *common* users
  - Lyrics expansion
  - Given lyrics, generate melody
  - Singing synthesis
  - Backing track generation
  - Automatic mixing

Singing Voice Conversion

Original singing

Conversion result

SVS vs SVC

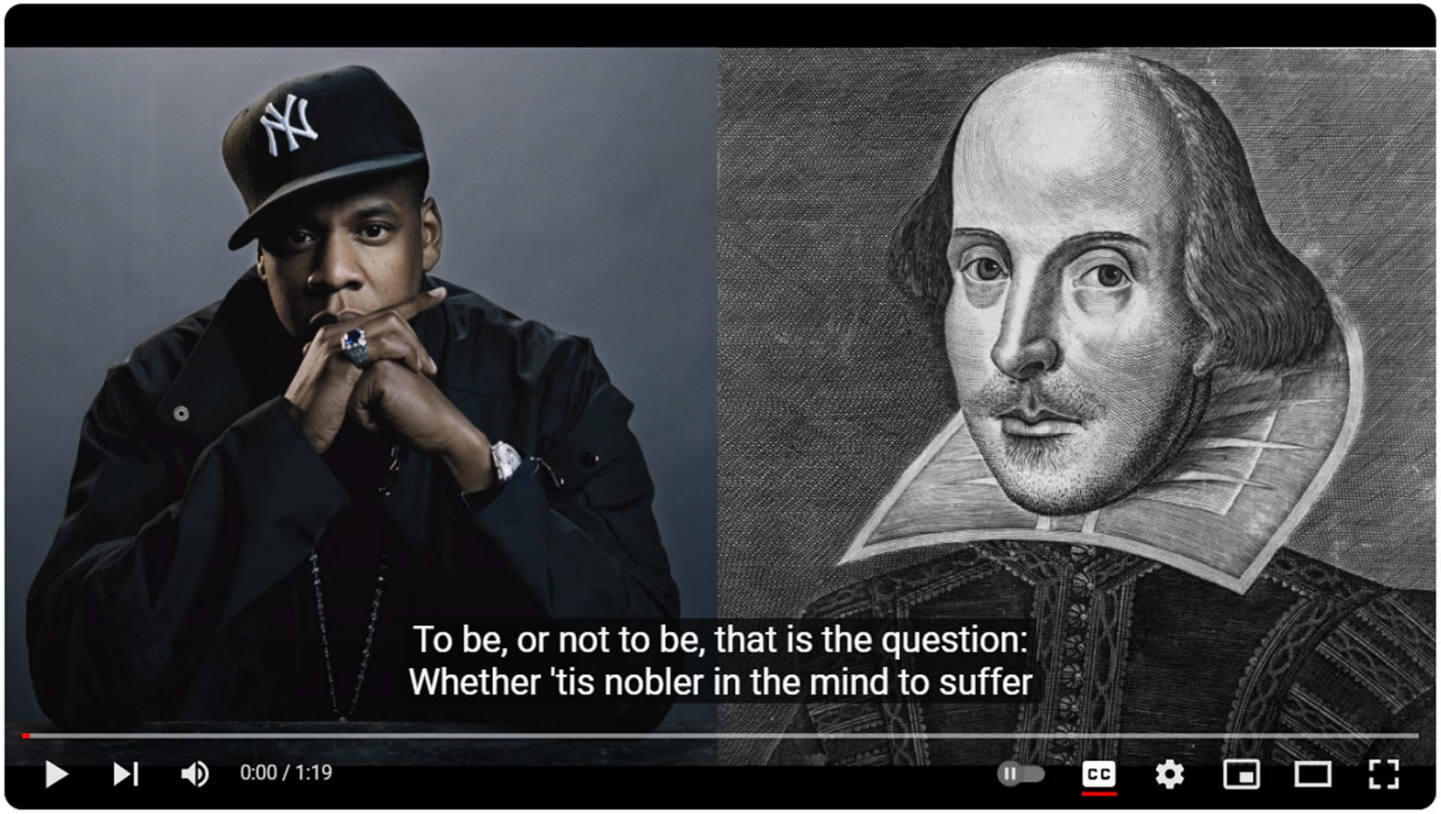
- **Singing voice synthesis (SVS):**
  lyrics + MIDI notes → audio
  - input/output *different domains*
    (**score** → **performance** + **sounds**)
  - lots of details to be determined
  - input/output *different lengths*
  - need to consider temporal dynamics
- **Singing voice conversion (SVC):**
  ref audio → audio
  - *same domains*
  - mainly about modeling timbre
  - can be of *same length*
  - "local" conversion might be sufficient

Outline

- Singing voice processing: Historical review
- **Neural text-to-speech synthesis**
- Neural singing voice synthesis
- Build your SVS model

TTS

https://www.youtube.com/watch?v=m7u-y9oqUSw



Jay-Z raps the "To Be, Or Not To Be" soliloquy from Hamlet (Speech Synthesis)

Progress of SVS Benefits from that of TTS

- SVS and TTS are similar
  - **Text-to-speech** (TTS): given text (and speaker ID), generate audio
  - **Singing voice synthesis** (SVS): given text and **MIDI** (and singer ID), generate audio
  - Both can be viewed as conditional audio generation problem
  - Both need to deal with the alignment problem between the input and the output
- Differences
  - Musical expressivity (timing, f0, dynamics, singing techniques)
  - Cost to collect data

TTS

https://github.com/tts-tutorial/interspeech2022

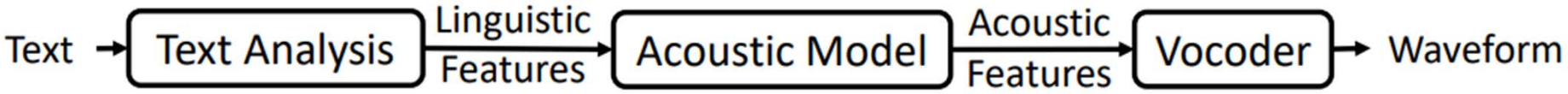


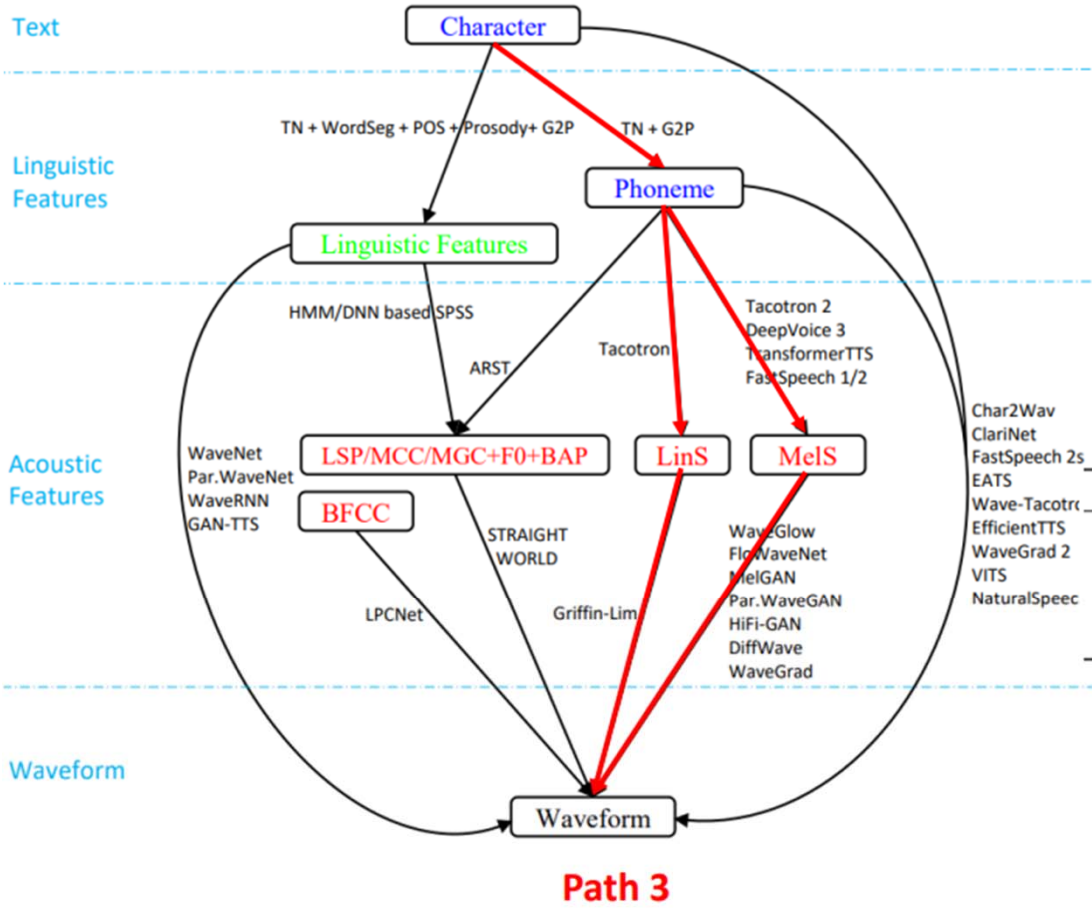
Figure 1: The three key components in neural TTS.

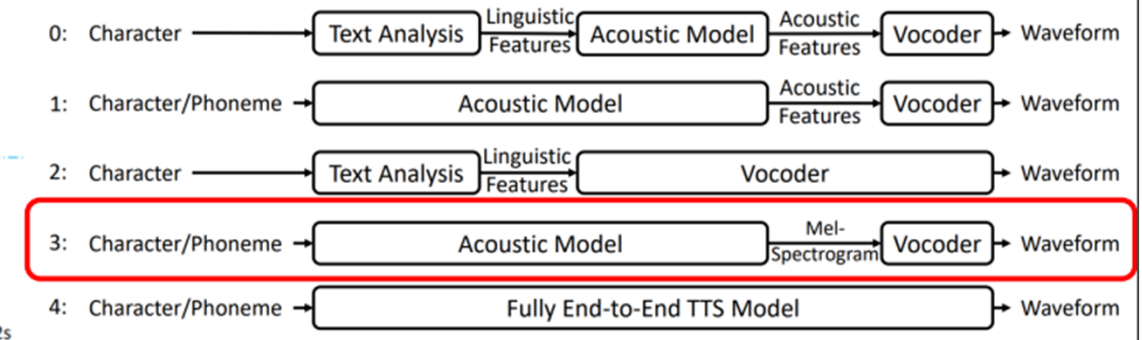
- Different linguistic features and acoustic features have been proposed
- SOTA neural TTS models
  - Simplify text analysis and directly take **character/phoneme sequences** as input
  - Simplify acoustic features with **Mel-spectrograms**

Ref: Tan et al, "A survey on neural speech synthesis," arXiv 2021

TTS

https://github.com/tts-tutorial/interspeech2022

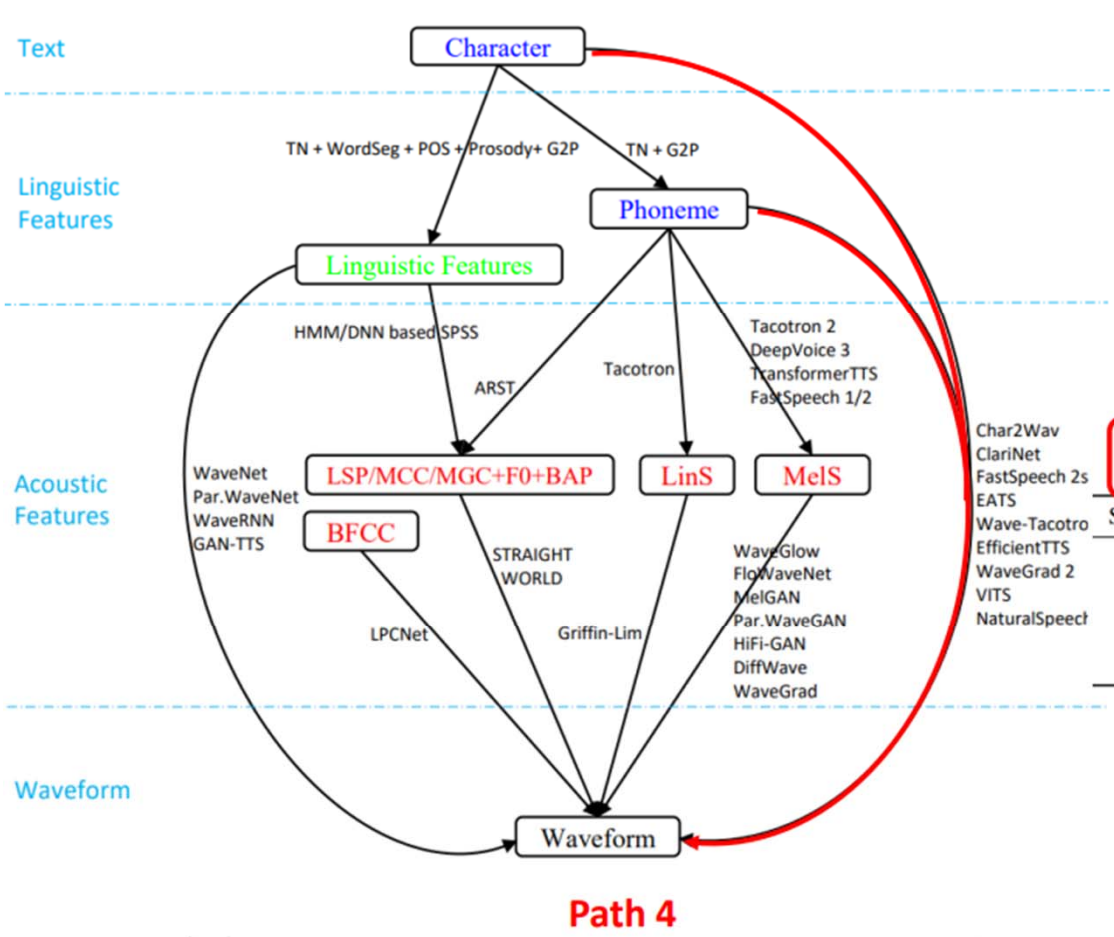


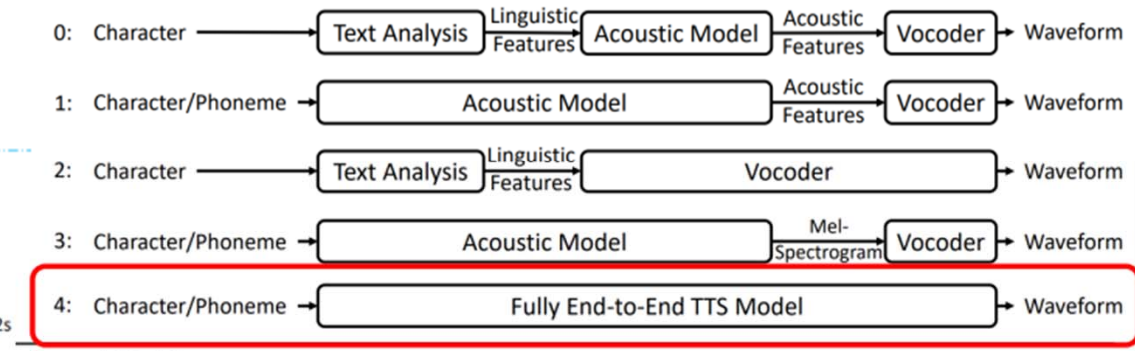


Stage	Models
0	SPSS [418, 358, 417, 427, 359]
1	ARST [377]
2	WaveNet [255], DeepVoice 1/2 [8, 88], Par. WaveNet [256], WaveRNN [151], HiFi-GAN [23]
3	DeepVoice 3 [271], Tacotron 2 [304], FastSpeech 1/2 [291, 293], WaveGlow [280], FloWaveNet [164]
4	Char2Wav [316], ClariNet [270], FastSpeech 2s [293], EATS [70], VITS [161], NaturalSpeech [346]

TTS

https://github.com/tts-tutorial/interspeech2022

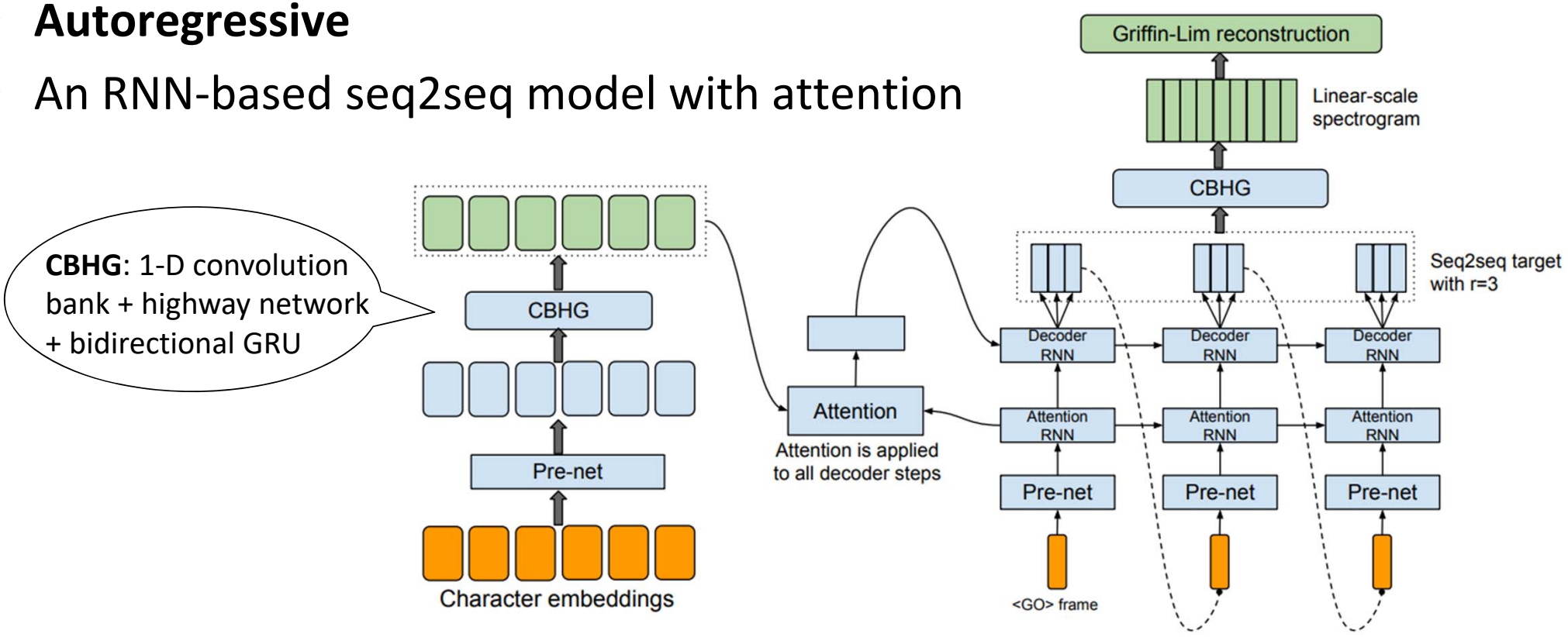




Stage	Models
0	SPSS [418, 358, 417, 427, 359]
1	ARST [377]
2	WaveNet [255], DeepVoice 1/2 [8, 88], Par. WaveNet [256], WaveRNN [151], HiFi-GAN [23]
3	DeepVoice 3 [271], Tacotron 2 [304], FastSpeech 1/2 [291, 293], WaveGlow [280], FloWaveNet [164]
4	Char2Wav [316], ClariNet [270], FastSpeech 2s [293], EATS [70], VITS [161], NaturalSpeech [346]

TacoTron

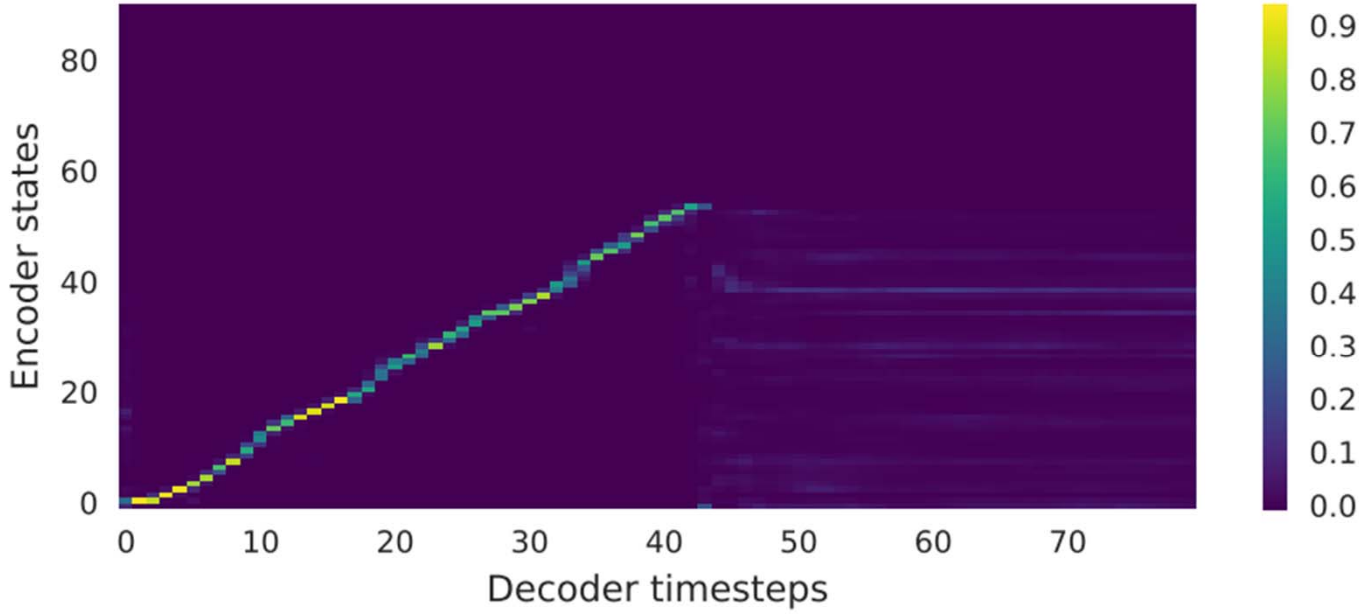
- Takes characters as input and **predict spectrogram frames**
- **Autoregressive**
- An RNN-based seq2seq model with attention



Ref: Wang et al, "Tacotron: Towards end-to-end speech synthesis," INTERSPEECH 2017

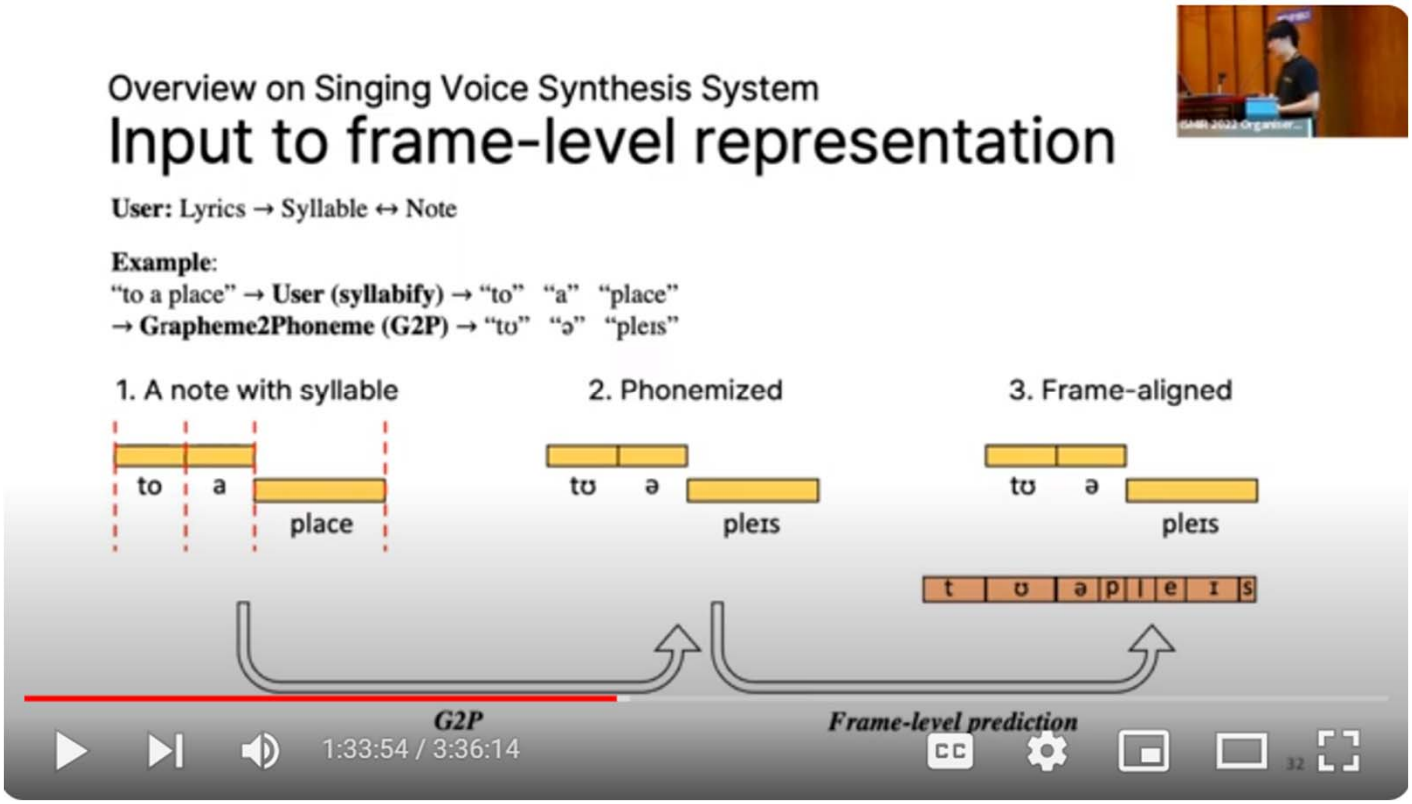
Why Attention?

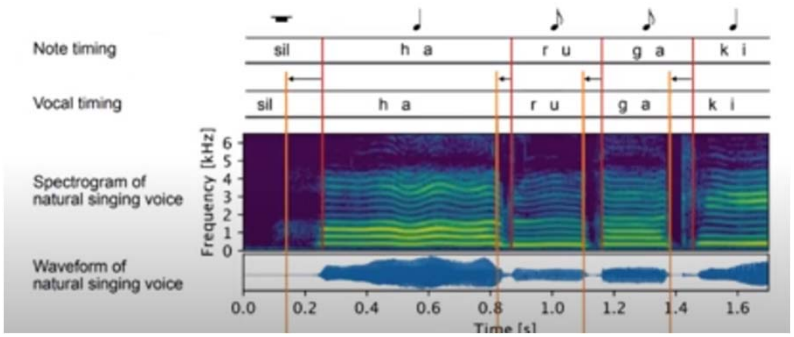
- To learn the **alignment** between input text and output audio
- Unlike source separation and vocoder, for TTS/SVS, the input and output are of **different lengths**, so some sorts of alignment is needed



(c) Tacotron (proposed)

Text-Audio Alignment



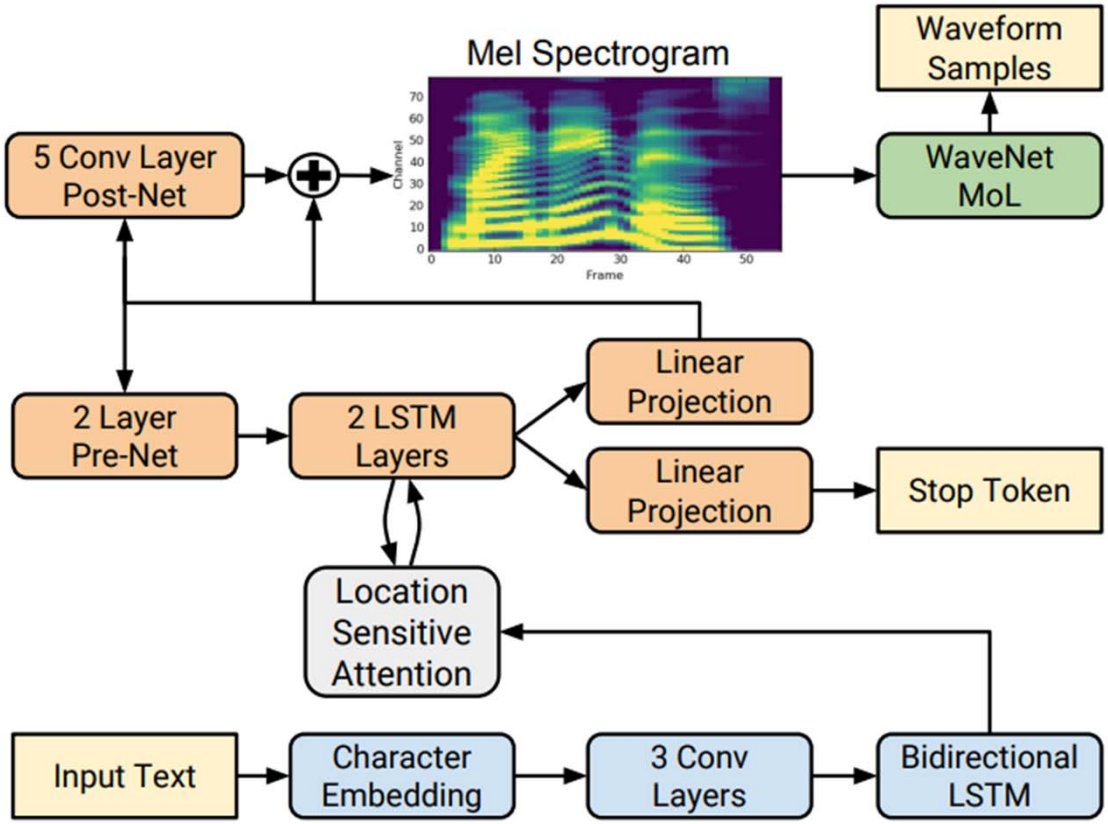


Snapshot of the ISMIR 2022 tutorial presented by Hyeong-Seok Choi

TacoTron 2

- Modifications
  - Mel-spectrograms instead of linear spectrograms
  - WaveNet-like Mel-vocoder
  - Simpler architecture (no CBHG)
  - Location sensitive attention
  - “Stop token”

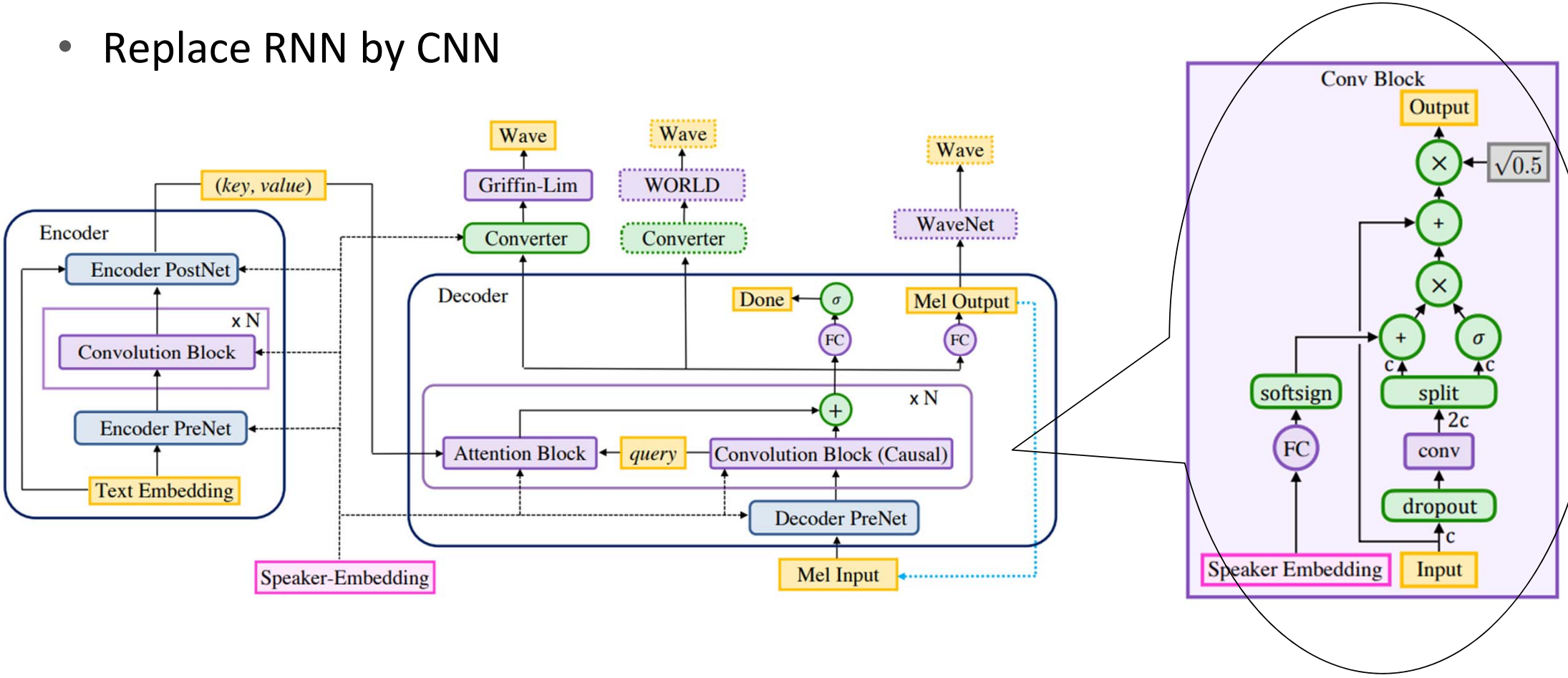
System	MOS
Parametric	3.492 ± 0.096
Tacotron (Griffin-Lim)	4.001 ± 0.087
Concatenative	4.166 ± 0.091
WaveNet (Linguistic)	4.341 ± 0.051
Ground truth	4.582 ± 0.053
Tacotron 2 (this paper)	$\mathbf{4.526 \pm 0.066}$



Ref: Shen et al, “Natural TTS synthesis by conditioning WaveNet on Mel spectrogram predictions,” ICASSP 2018

Deep Voice 3

- Replace RNN by CNN

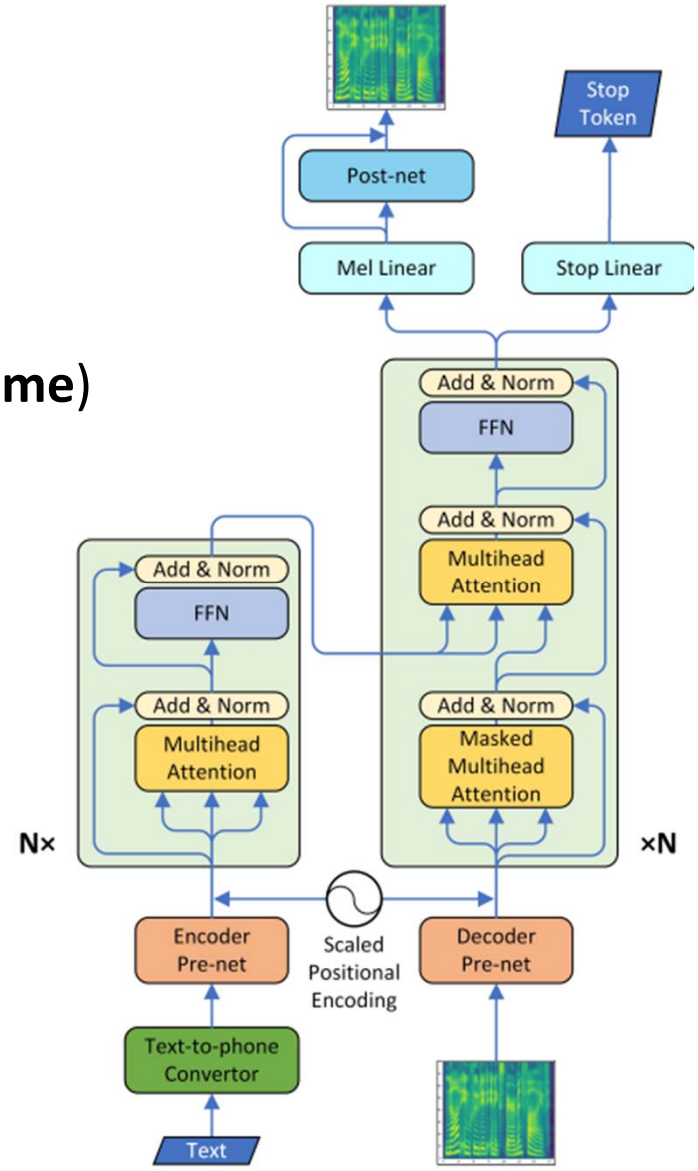


Ref: Ping et al, "Deep Voice 3: Scaling text-to-speech with convolutional sequence learning," ICLR 2018

Transformer-TTS

- Replace RNN by multi-head attention
- Using **phoneme** sequences as input
  - *Character/spelling* (**grapheme**) vs *pronunciation* (**phoneme**)

System	MOS	CMOS
Tacotron2	4.39 ± 0.05	0
Our Model	4.39 ± 0.05	**0.048**
Ground Truth	4.44 ± 0.05	-



Ref: Li et al, “Neural speech synthesis with Transformer network,” AAAI 2019

Drawbacks of Autoregressive models

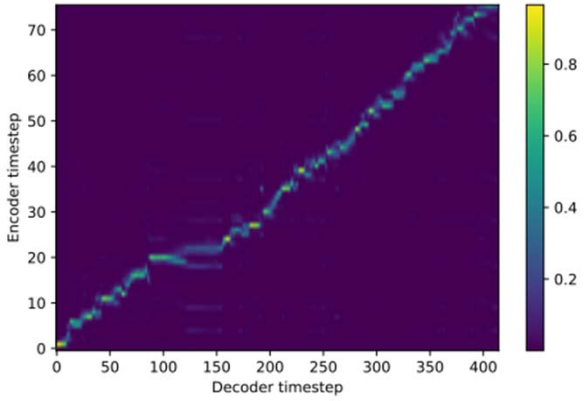
- **Slow inference speed**
- **Synthesized speech is usually not robust**
  - Due to *error propagation* and the *wrong attention alignments* between text and speech in the autoregressive generation, the generated mel-spectrogram is usually deficient with the problem of **words skipping** and **repeating**
- **Synthesized speech is lack of controllability**
  - Hard to directly control the *voice speed* and *prosody* in autoregressive generation

Ref: Ren et al, "Fastspeech: Fast, robust and controllable text to speech," NeurIPS 2019

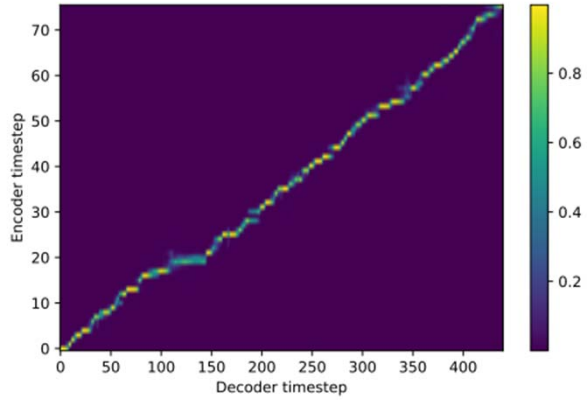
Robustness Problem About Alignment

- Properties of good alignment [1]
  - **Monotonicity**
  - **Continuity**
  - **Completeness**: cover all input tokens

- However, unless specifically enforcing related constraints (e.g., [2]), the alignment learned by attention modules may not have the aforementioned properties



(c) *Tacotron 2*



(d) *Tacotron 2 with* $\mathcal{L}_{align}$

Ref 1: “EfficientTTS: An efficient and high-quality text-to-speech architecture,” ICML 2021

Ref 2: “One TTS alignment to rule them all,” arXiv 2021

Drawbacks of Autoregressive models

- **Slow inference speed → Use non-autogressive models!**
- **Synthesized speech is usually not robust → Predict phoneme duration!**
  - Due to *error propagation* and the *wrong attention alignments* between text and speech in the autoregressive generation, the generated mel-spectrogram is usually deficient with the problem of words skipping and repeating”
- **Synthesized speech is lack of controllability → Add conditioning**
  - Hard to directly control the voice speed and prosody in autoregressive generation

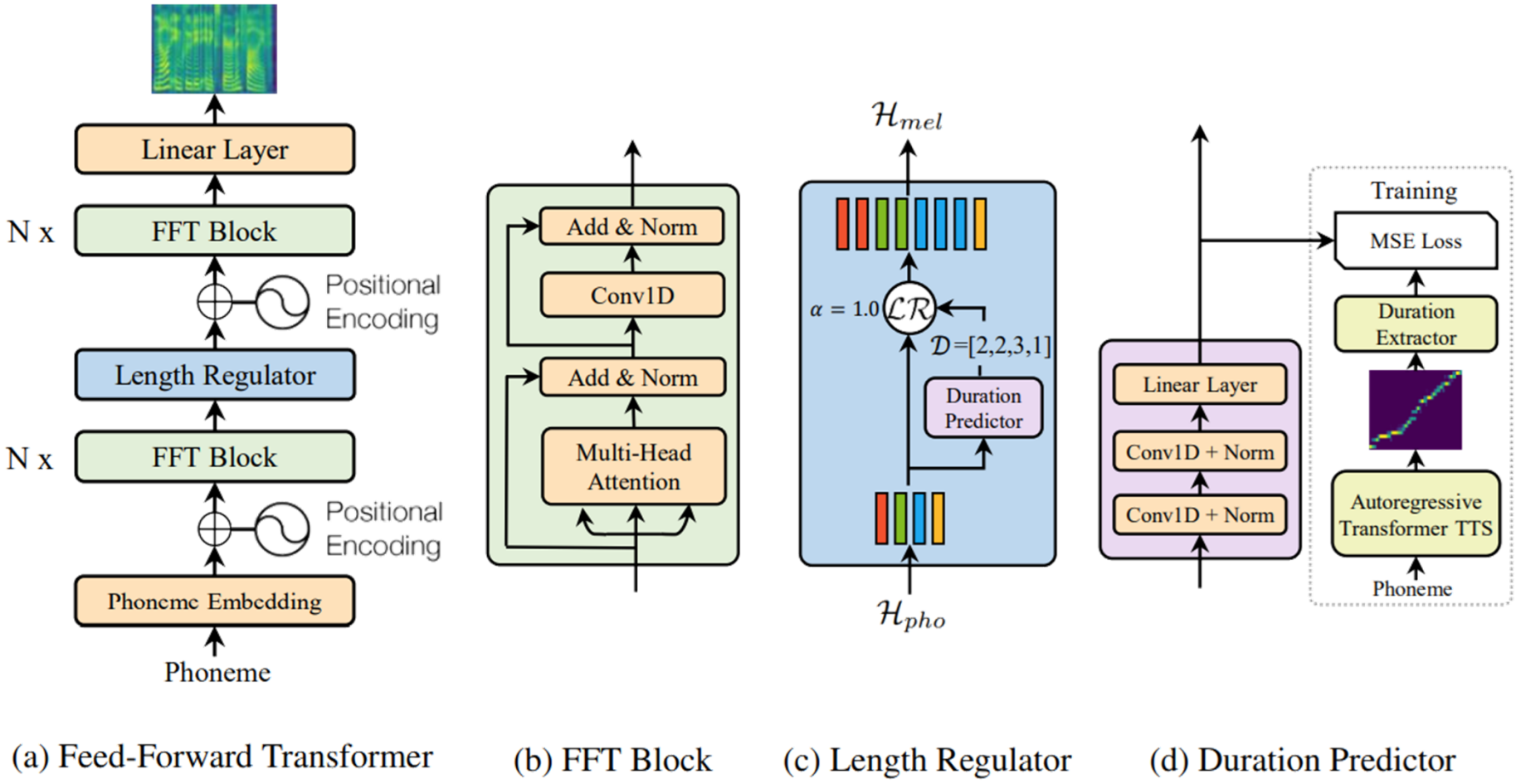
Ref: Ren et al, “Fastspeech: Fast, robust and controllable text to speech,” NeurIPS 2019

Predict Phoneme Duration?

- Cross-attention based alignment is too much because for TTS the alignment between input sequence and output sequence must be monotonic

- Actually, it's good enough to know the **duration** of each character or phoneme (e.g., how many frames, or how many time samples)

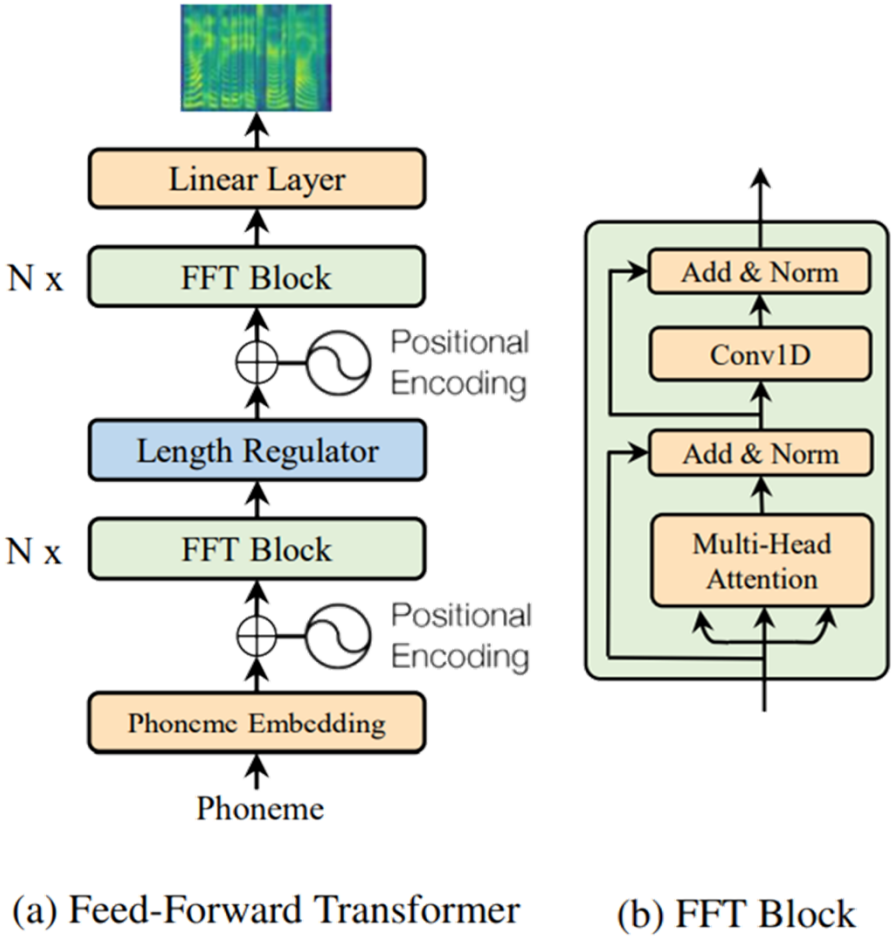
FastSpeech



(a) Feed-Forward Transformer (b) FFT Block (c) Length Regulator (d) Duration Predictor

Ref: Ren et al, "Fastspeech: Fast, robust and controllable text to speech," NeurIPS 2019

FastSpeech



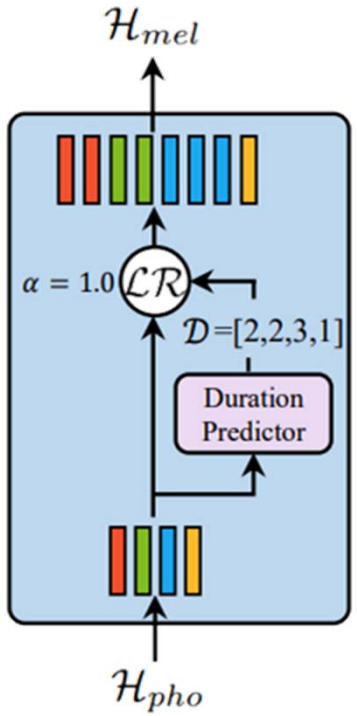
(a) Feed-Forward Transformer (b) FFT Block

- **Feed-forward Transformer** (FFT)
  → Non-autogressive & *fast inference*
  - No causal masking
  - Parallel generation
  - Conv1D instead of dense layers inside an FFT block

Ref: Ren et al, "Fastspeech: Fast, robust and controllable text to speech," NeurIPS 2019

FastSpeech

- **Length regulator** → *Robustness*
  - Up-samples the phoneme sequence according to the phoneme duration (i.e., the number of frames that each phoneme corresponds to) to match the length of the mel-spectrogram sequence
- The phoneme duration is from a **duration predictor**
  - Explicitly determine the correspondence between the phoneme sequence and the mel-spectrogram sequence
  - **Hard alignment**
- We can also *manually* adjust the phoneme duration → *Controllability*

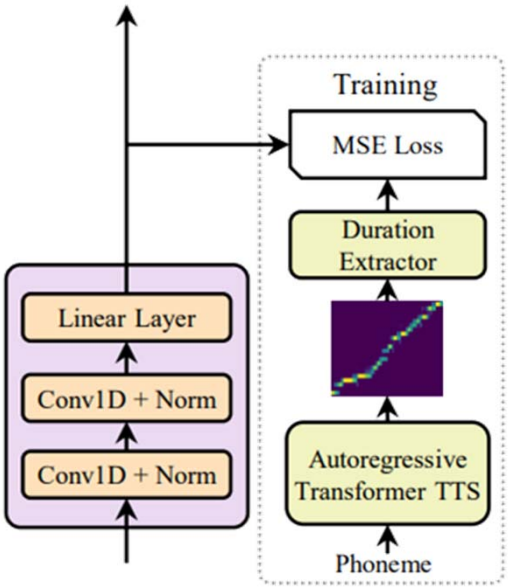


(c) Length Regulator

Ref: Ren et al, "Fastspeech: Fast, robust and controllable text to speech," NeurIPS 2019

FastSpeech

- **Phoneme duration predictor**
  - Produces *hard alignments*, which is very different from soft and automatic attention alignments in the autoregressive models
  - Predict the length in the logarithmic domain and use **MSE loss**
  - **Jointly trained** with FFT layers
- Ground truth phoneme duration for training?
  - From an autoregressive *teacher* model
  - Via Montreal forced alignment (MFA)
  - Or by manually labeling



(d) Duration Predictor

Ref: Ren et al, "Fastspeech: Fast, robust and controllable text to speech," NeurIPS 2019

FastSpeech: Evaluation Result

Method	MOS
GT	4.41 ± 0.08
GT (Mel + WaveGlow)	4.00 ± 0.09
Tacotron 2 [22] (Mel + WaveGlow)	3.86 ± 0.09
Merlin [28] (WORLD)	2.40 ± 0.13
Transformer TTS [14] (Mel + WaveGlow)	3.88 ± 0.09
FastSpeech (Mel + WaveGlow)	3.84 ± 0.08

Table 1: The MOS with 95% confidence intervals.

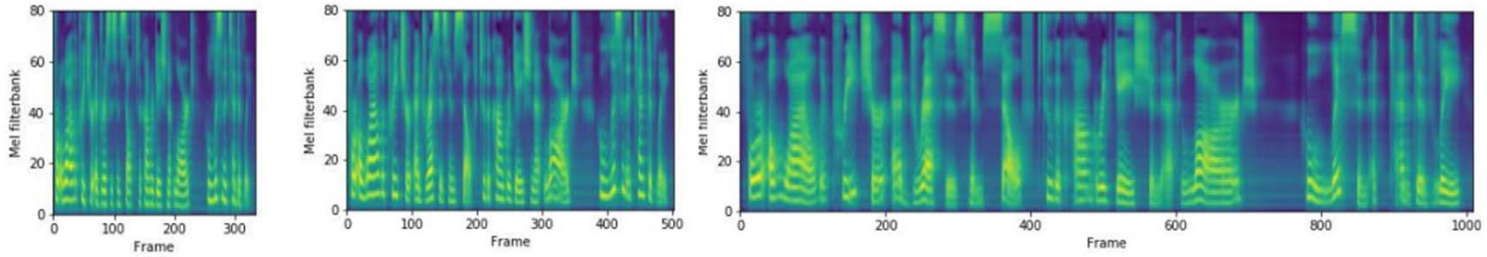
Method	Latency (s)	Speedup
Transformer TTS [14] (Mel)	6.735 ± 3.969	/
FastSpeech (Mel)	0.025 ± 0.005	269.40×
Transformer TTS [14] (Mel + WaveGlow)	6.895 ± 3.969	/
FastSpeech (Mel + WaveGlow)	0.180 ± 0.078	38.30×

Table 2: The comparison of inference latency with 95% confidence intervals. The evaluation is conducted on a server with 12 Intel Xeon CPU, 256GB memory, 1 NVIDIA V100 GPU and batch size of 1. The average length of the generated mel-spectrograms for the two systems are both about 560.

Method	Repeats	Skips	Error Sentences	Error Rate
Tacotron 2	4	11	12	24%
Transformer TTS	7	15	17	34%
FastSpeech	0	0	0	0%

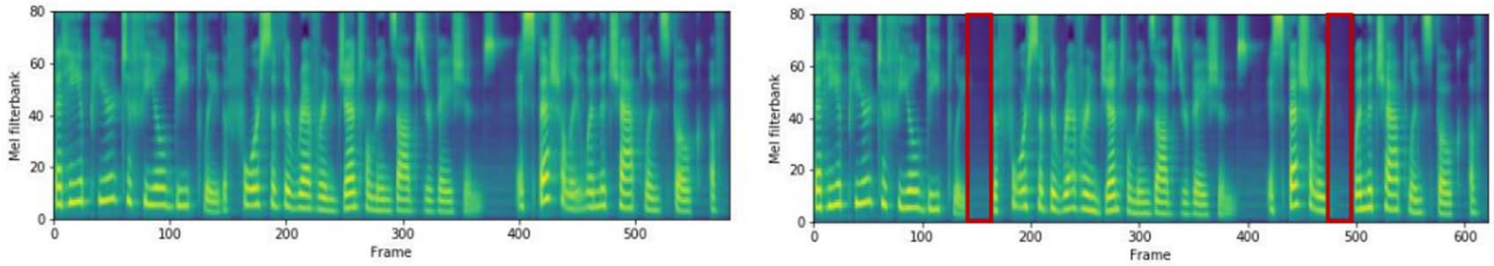
Table 3: The comparison of robustness between FastSpeech and other systems on the 50 particularly hard sentences. Each kind of word error is counted at most once per sentence.

FastSpeech: Controllability



(a) 1.5x Voice Speed (b) 1.0x Voice Speed (c) 0.5x Voice Speed

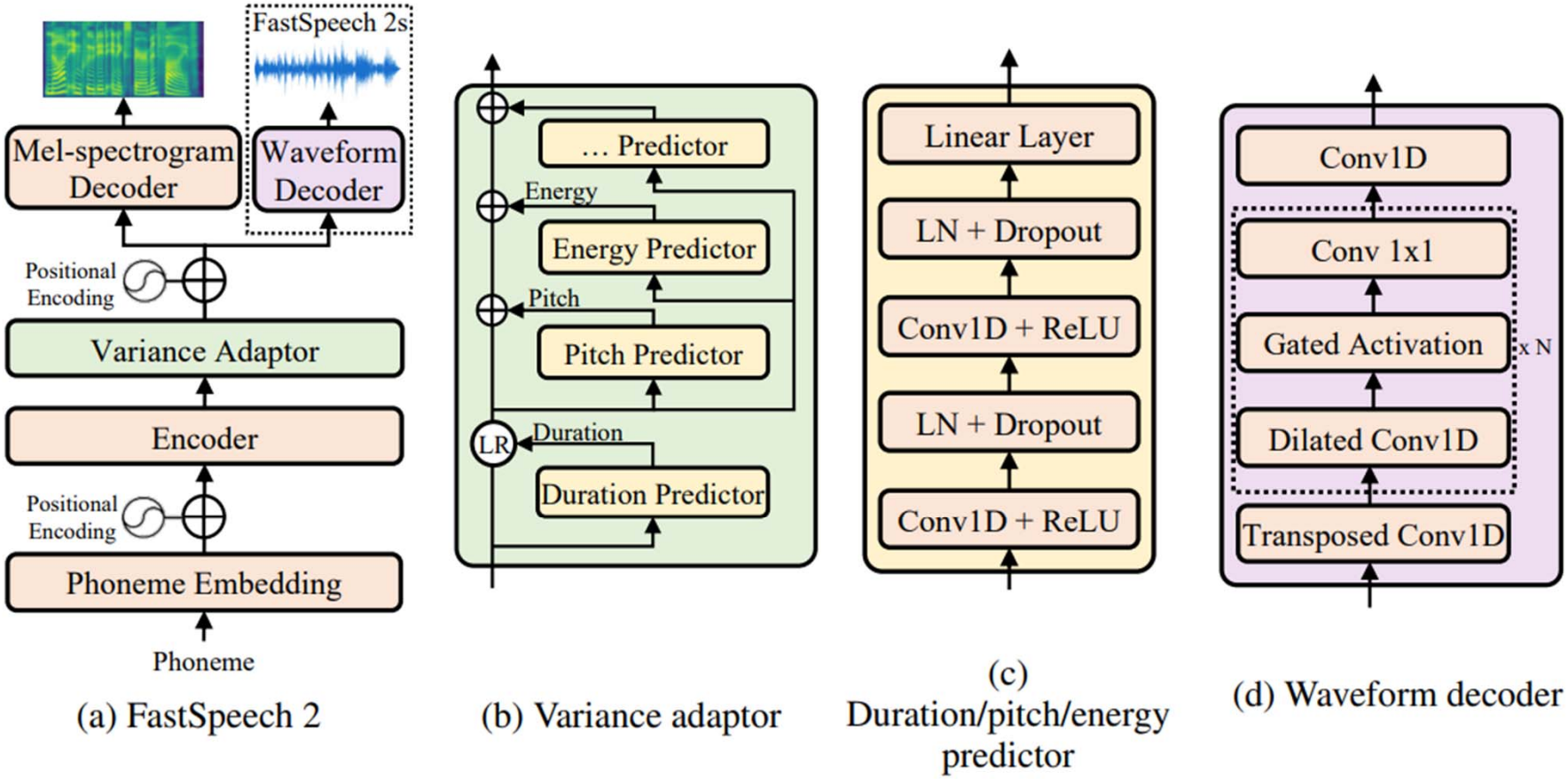
Figure 3: The mel-spectrograms of the voice with 1.5x, 1.0x and 0.5x speed respectively. The input text is "*For a while the preacher addresses himself to the congregation at large, who listen attentively*".



(a) Original Mel-spectrograms (b) Mel-spectrograms after adding breaks

Figure 4: The mel-spectrograms before and after adding breaks between words. The corresponding text is "*that he appeared to feel **deeply** the force of the reverend gentleman's observations, **especially** when the chaplain spoke of*". We add breaks after the words "*deeply*" and "*especially*" to improve the prosody. The red boxes in Figure 4b correspond to the added breaks.

FastSpeech 2 & FastSpeech 2s



Ref: Ren et al, "Fastspeech 2: Fast and high-quality end-to-end text to speech," ICLR 2021

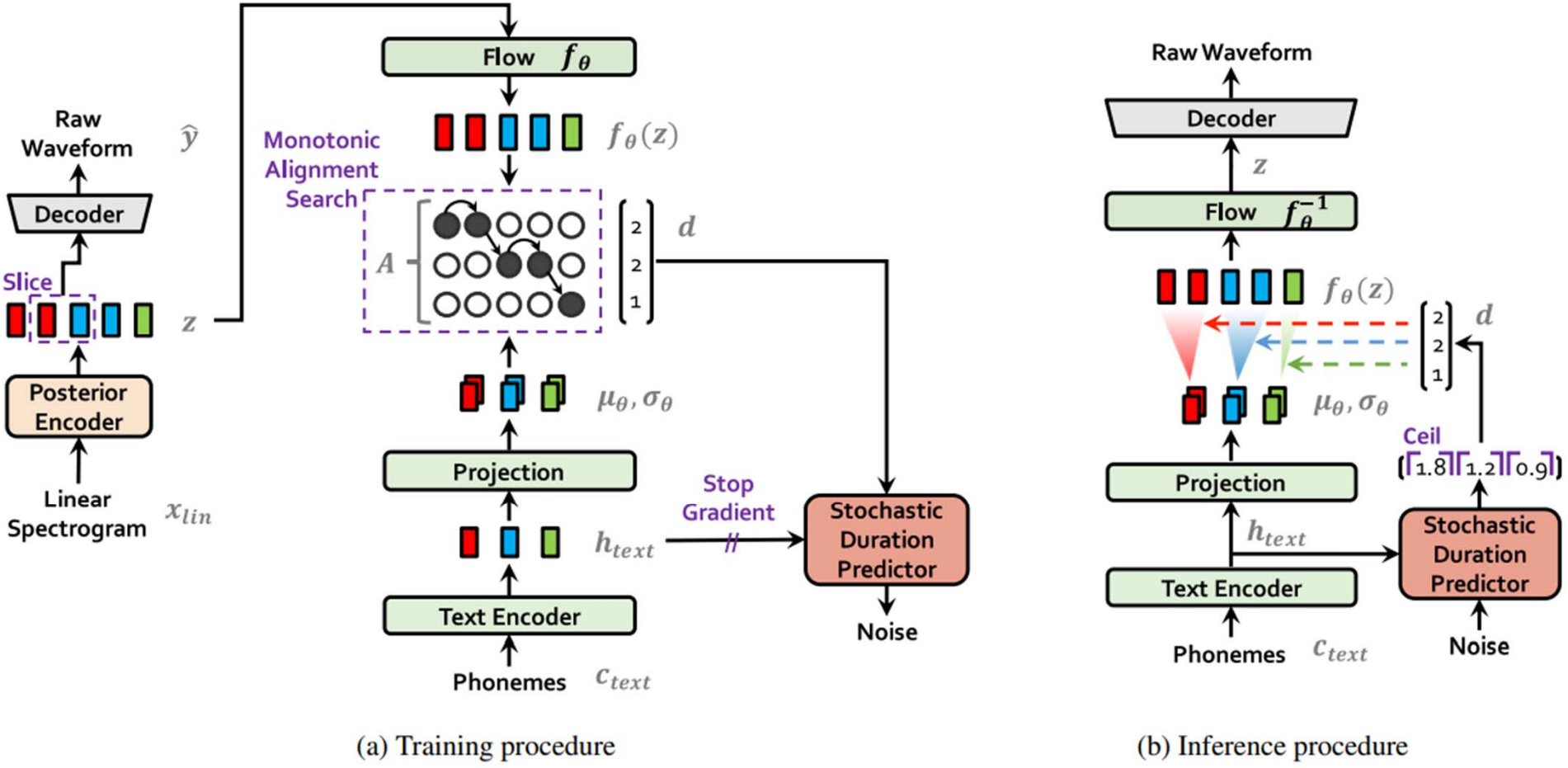
FastSpeech 2 vs FastSpeech

- 3x training speed-up
- More controllability

Method	MOS
GT	4.30 ± 0.07
GT (Mel + PWG)	3.92 ± 0.08
Tacotron 2 (Shen et al., 2018) (Mel + PWG)	3.70 ± 0.08
Transformer TTS (Li et al., 2019) (Mel + PWG)	3.72 ± 0.07
FastSpeech (Ren et al., 2019) (Mel + PWG)	3.68 ± 0.09
FastSpeech 2 (Mel + PWG)	3.83 ± 0.08
FastSpeech 2s	3.71 ± 0.09

Method	Training Time (h)	Inference Speed (RTF)	Inference Speedup
Transformer TTS (Li et al., 2019)	38.64	9.32×10^{-1}	/
FastSpeech (Ren et al., 2019)	53.12	1.92×10^{-2}	48.5×
FastSpeech 2	**17.02**	1.95×10^{-2}	47.8×
FastSpeech 2s	92.18	$\mathbf{1.80 \times 10^{-2}}$	**51.8**×

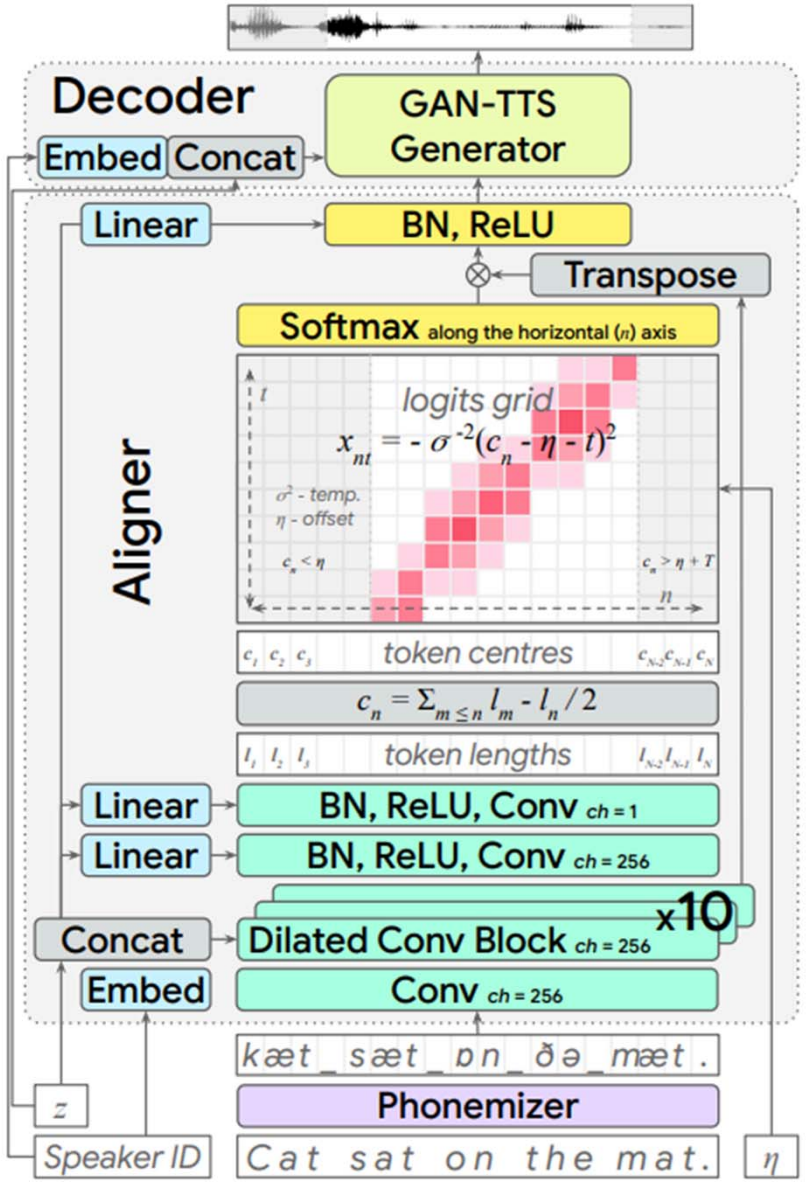
VITS



Ref: Kim et al, “Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech,” ICML 2021

EATS

- Feedforward, not autoregressive
- Directly generate waveforms; no vocoders
- Soft DTW for **differentiable alignment**



Ref: Donahue et al, “End-to-end adversarial text-to-speech,” ICLR 2021

Drawbacks of Non-autoregressive models

- It's like deterministic; **lack randomness**
- Result of autoregressive models can sometimes sound more natural possibly due to the autoregressive prediction

Ref: Ren et al, "Fastspeech: Fast, robust and controllable text to speech," NeurIPS 2019

Recap: TTS

- **TacoTron 2** (ICASSP'18)
  - Use a recurrent seq2seq architecture
  - Autoregressive
  - Let attention learns the alignment between the input phoneme sequence and the output Mel-spectrogram sequence

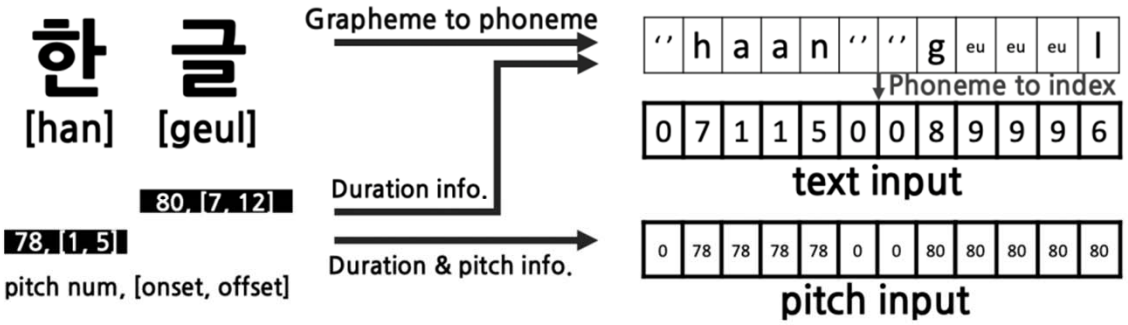
- **FastSpeech 2** (ICLR'21)
  - Use a feed-forward Transformer architecture
  - Non-autoregressive
  - Use duration predictor (among many other variance adaptors)

Outline

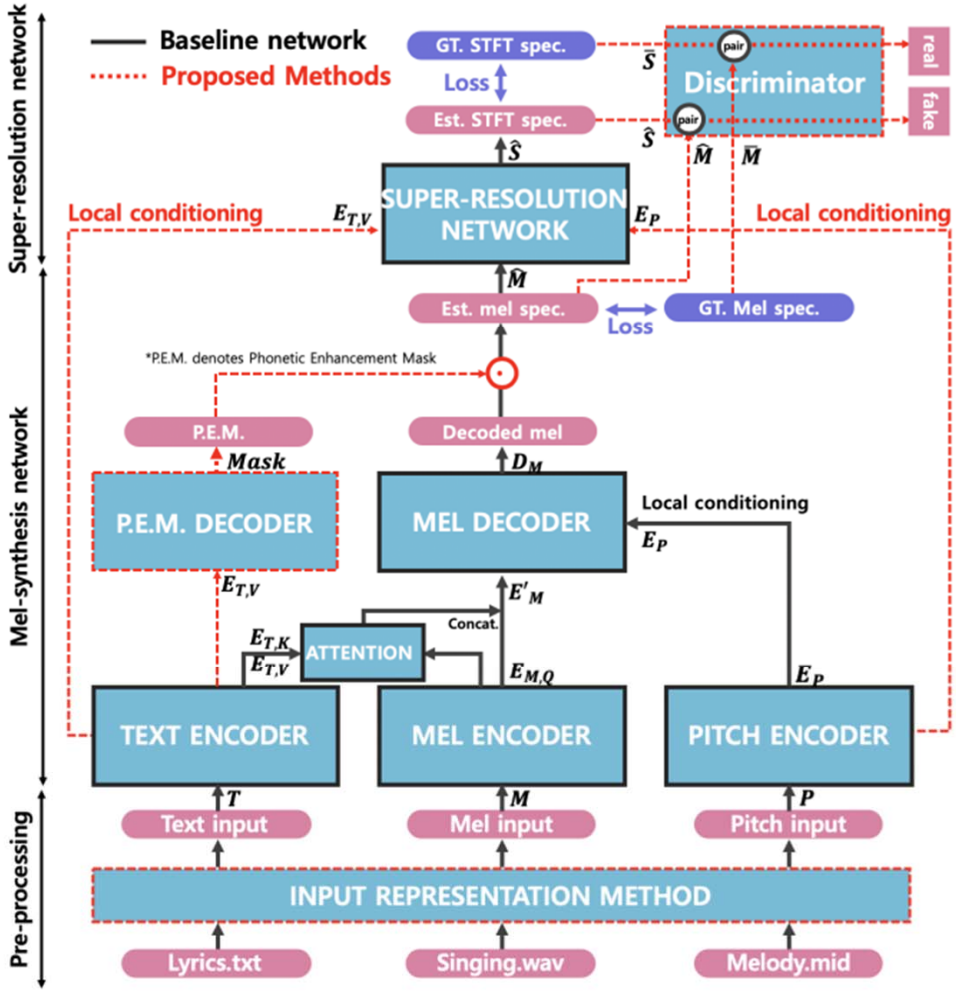
- Singing voice processing: Historical review
- Neural text-to-speech synthesis
- **Neural singing voice synthesis**
- Build your SVS model

SNU'2019 Model

- Autoregressive
- Use GAN
- Pitch embedding



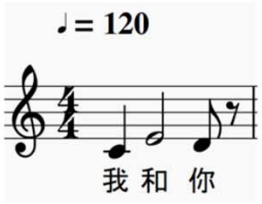
- **DB**: 60 songs from 1 female singer (~2hrs), manual annotation

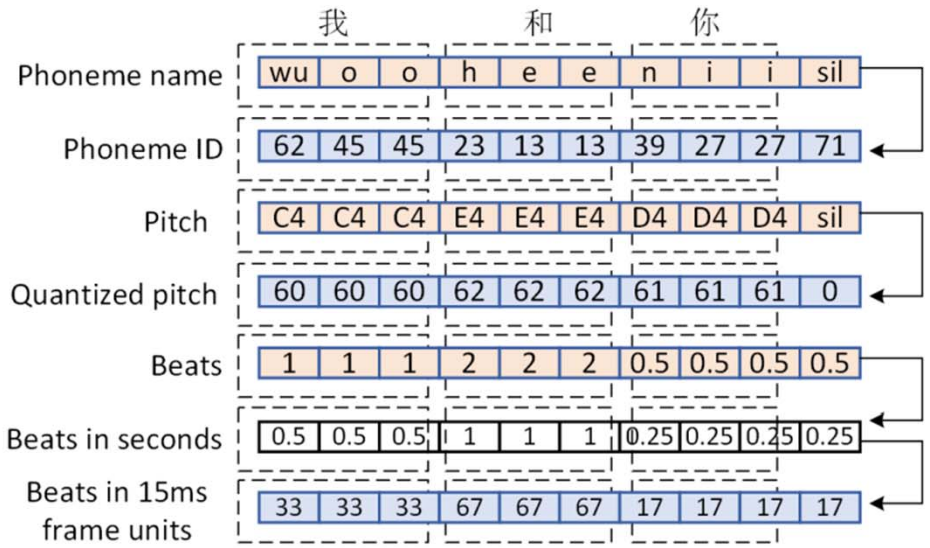


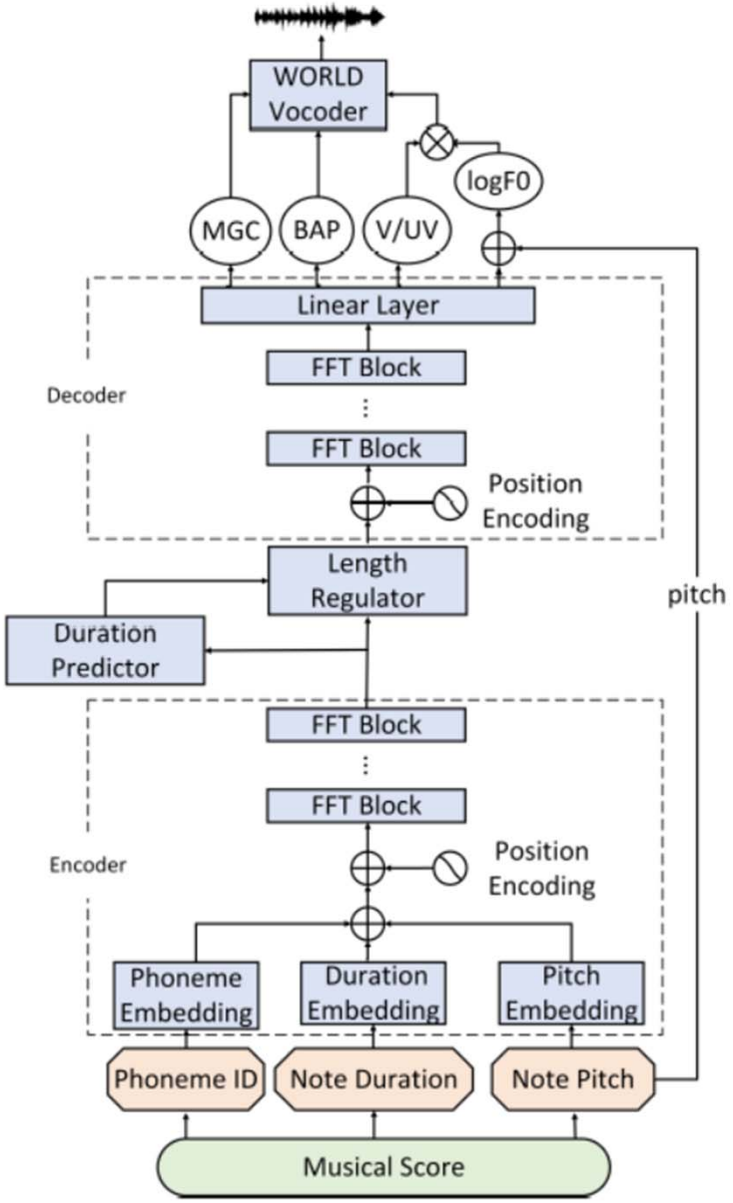
Ref: Lee et al, "Adversarially trained end-to-end Korean singing voice synthesis system," INTERSPEECH 2019

XiaoiceSing

- FastSpeech + WORLD vocoder
- **DB**: 2297 Mandarin pop songs from a female singer (~74hrs), manual annotation





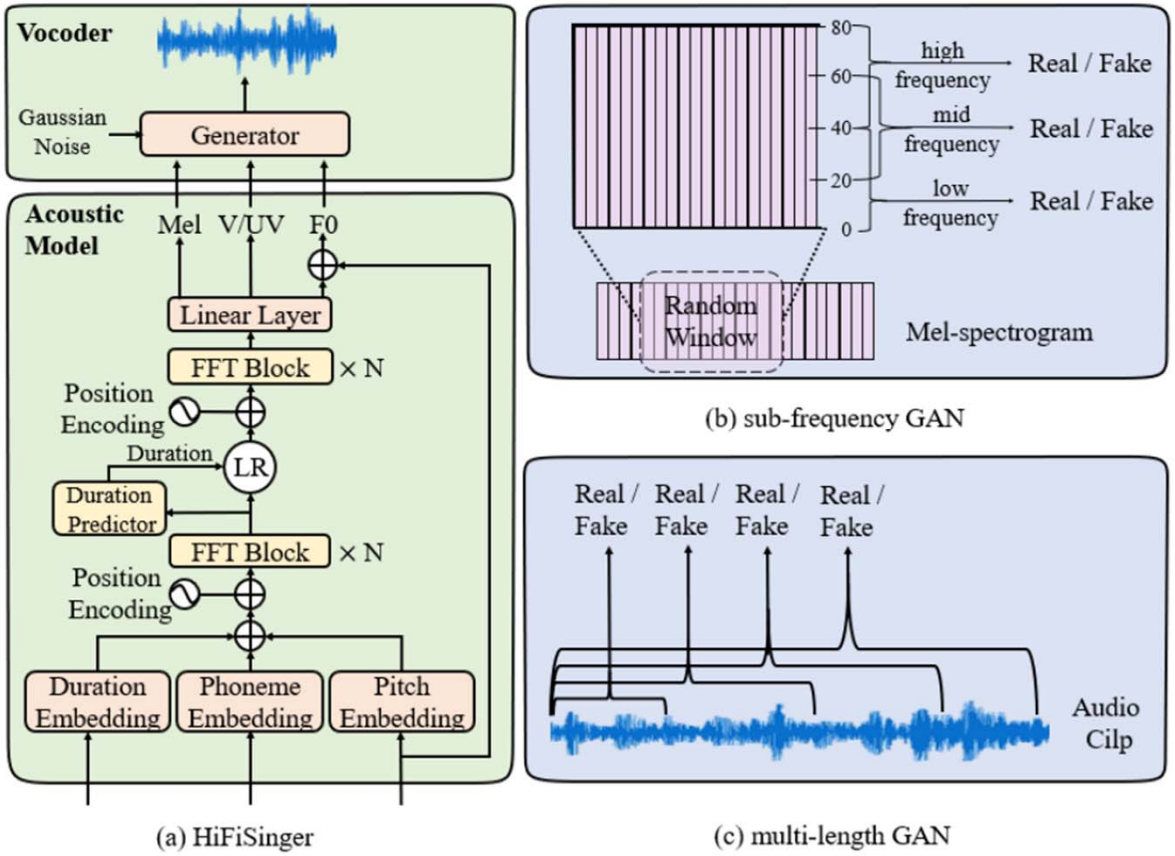


Ref: Lu et al, "XiaoiceSing: A high-quality and integrated singing voice synthesis system," INTERSPEECH 2020

HiFiSinger

https://speechresearch.github.io/hifisinger/

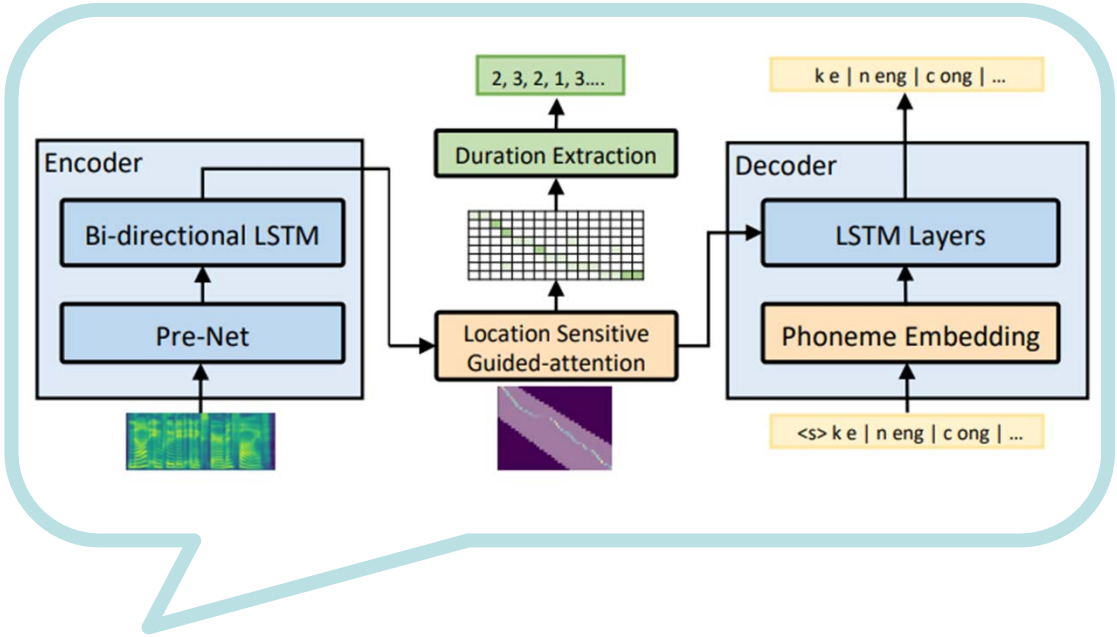
- Use Parallel WaveGAN based neural vocoder
- Use sub-frequency GAN to improve Mel-spectrogram generation
- Use multi-length GAN to improve waveform generation
- **DB**: 11 hrs songs from a female singer, manual annotation

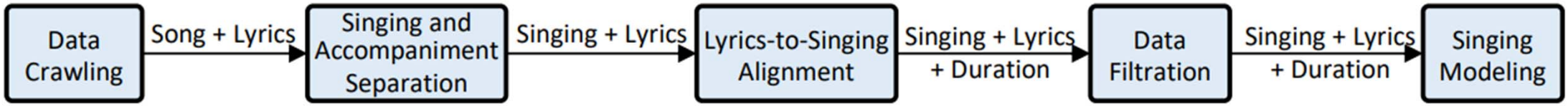


Ref: Chen et al, "HiFiSinger: Towards high-fidelity neural singing voice synthesis," arXiv 2020

DeepSinger

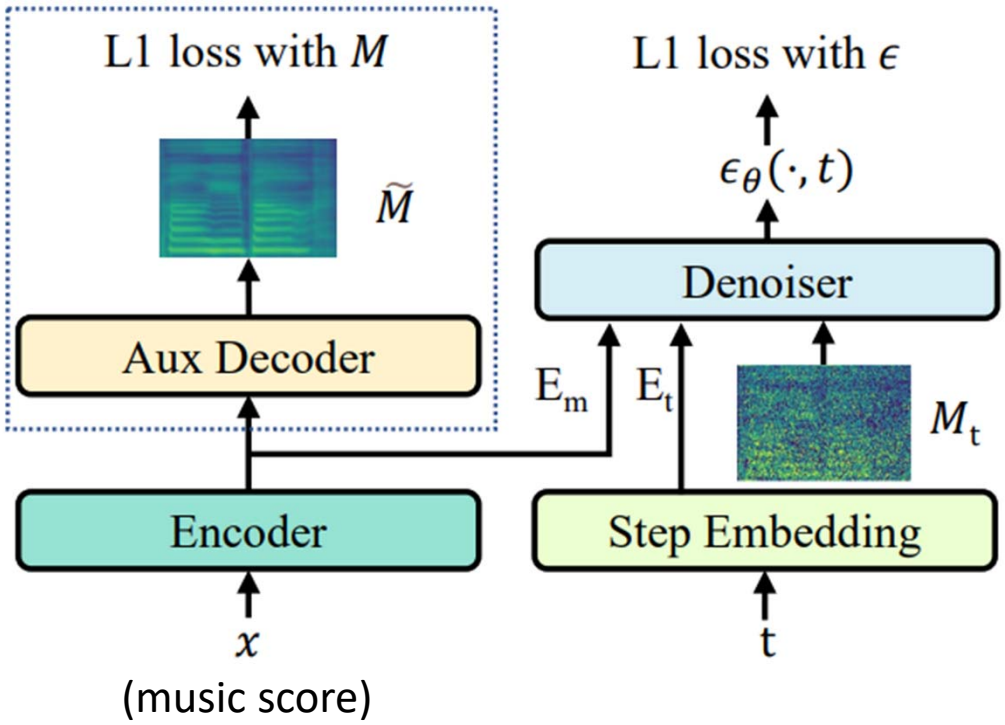
- Efforts in building an SVS dataset **is huge!**
  - Record singing in professional studio
  - Manual annotation to get paired data
    - Usually first manually split a whole song into aligned lyrics and audio in sentence level, and then extract the duration of each phoneme either by manual alignment or a phonetic timing model
- DeepSinger uses data **from the web** + **automatic alignment**

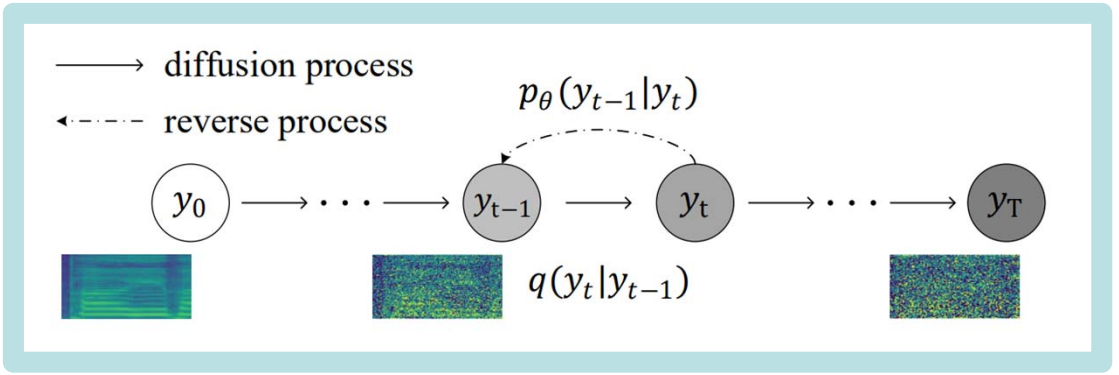




Ref: Ren et al, "DeepSinger: Singing voice synthesis with data mined from the web," KDD 2020

DiffSinger



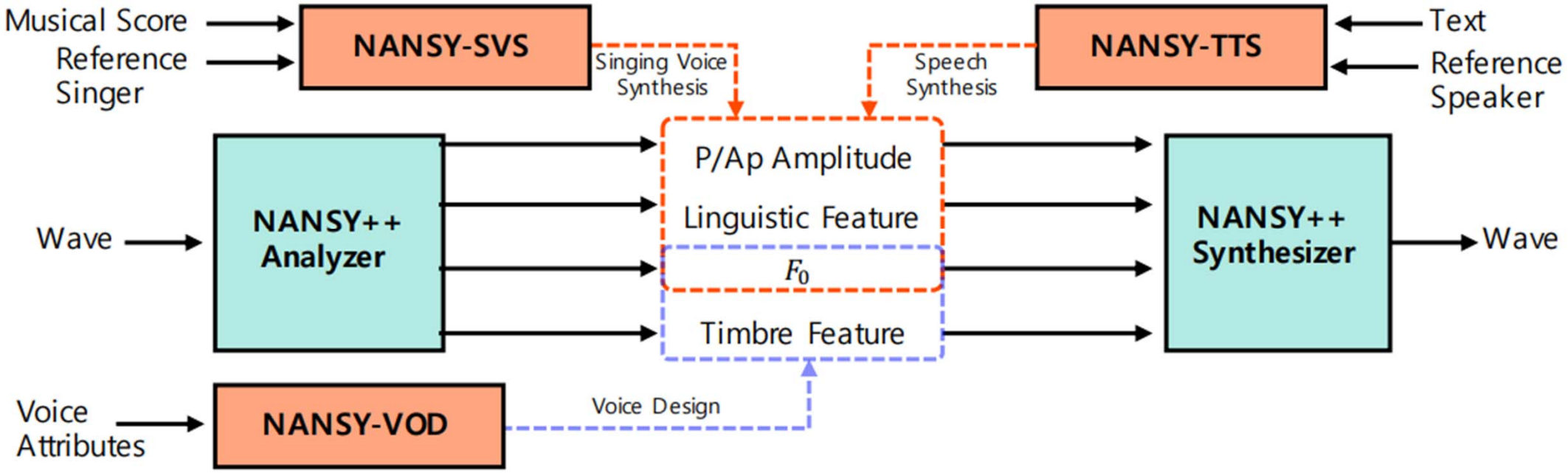


- **FastSpeech-like encoder** to provide condition for the reverse diffusion process
- **Non-causal WaveNet-based denoiser** to convert noise into Mel-specrogram

Ref: Ren et al, "DiffSinger: Singing voice synthesis via shallow diffusion mechanism," AAAI 2022

Nansy++

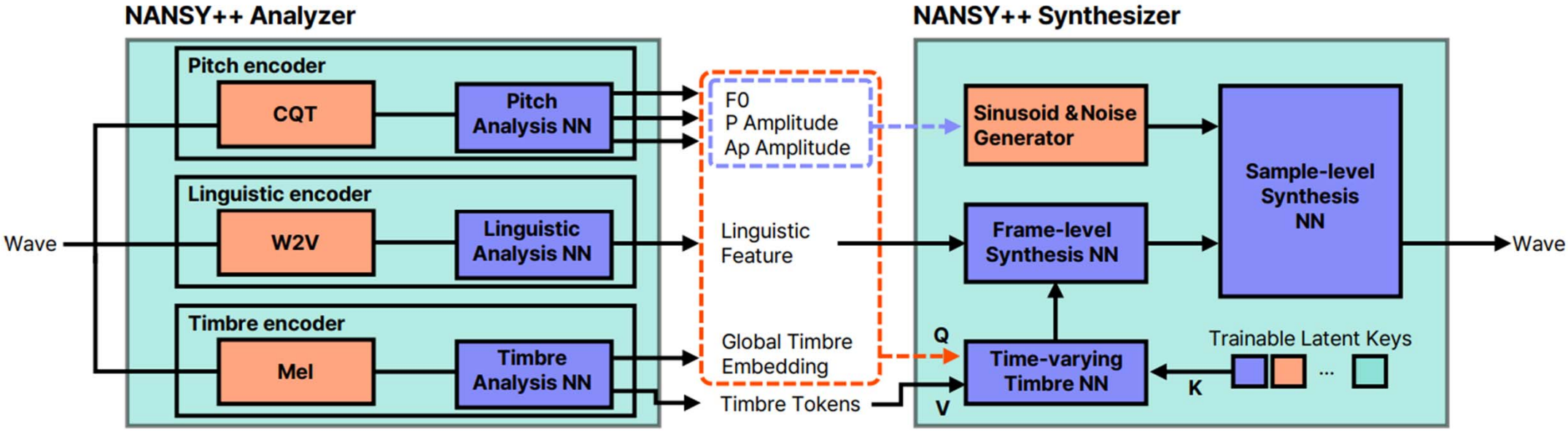
- A single model that do 1) VC/SVC, 2) TTS, 3) SVS, and 4) voice designing



Ref: Choi et al, "NANSY++: Unified voice synthesis with neural analysis and synthesis," ICLR 2023

Nansy++

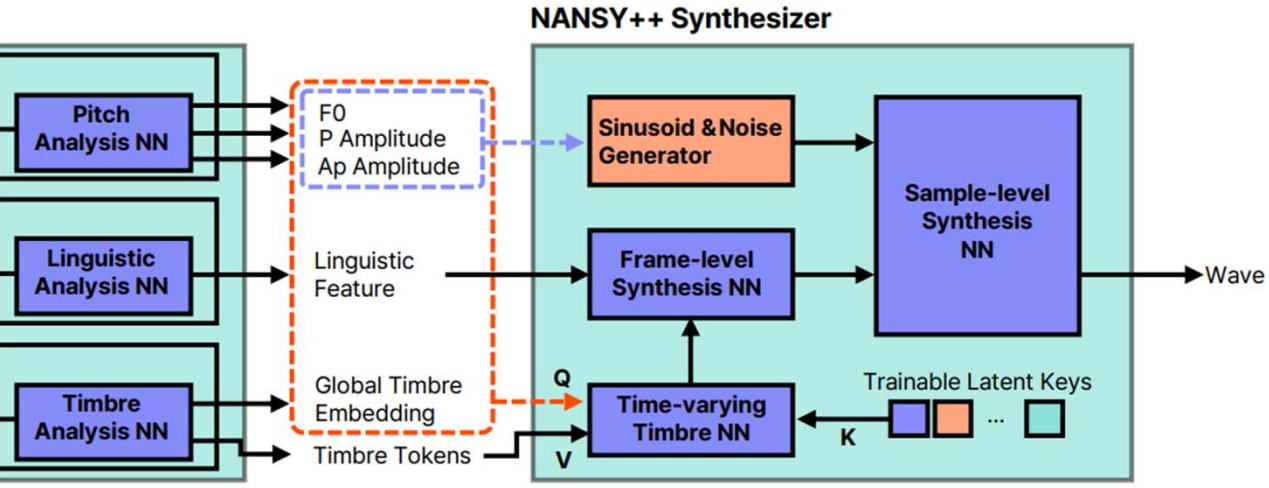
- “We trained the backbone model on 10,571 hours (speech: 10,092 hours, singing: 479 hours) of proprietary 44.1 kHz audio recordings composed of 6,176 speakers and 624 singers”

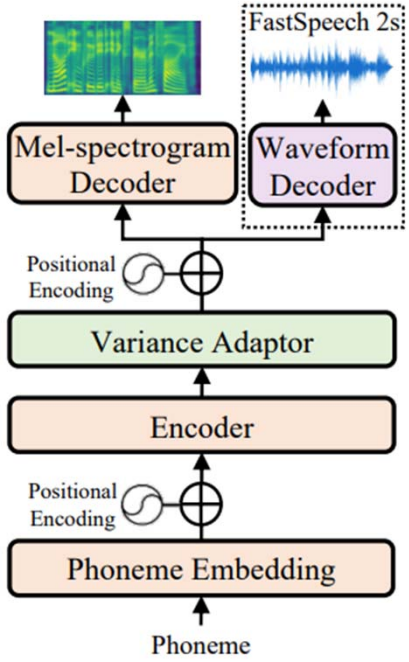


Ref: Choi et al, “NANSY++: Unified voice synthesis with neural analysis and synthesis,” ICLR 2023

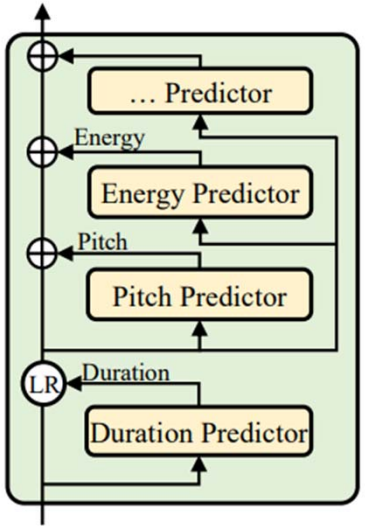
Nansy++ vs FastSpeech 2

- Nansy++ may provide more explicit disentanglement of various attributes and therefore allows for more flexible control





(a) FastSpeech 2



(b) Variance adaptor

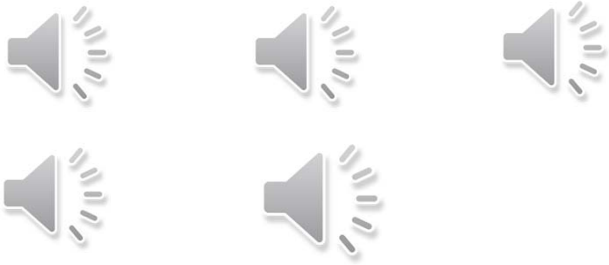
Recap: SVS

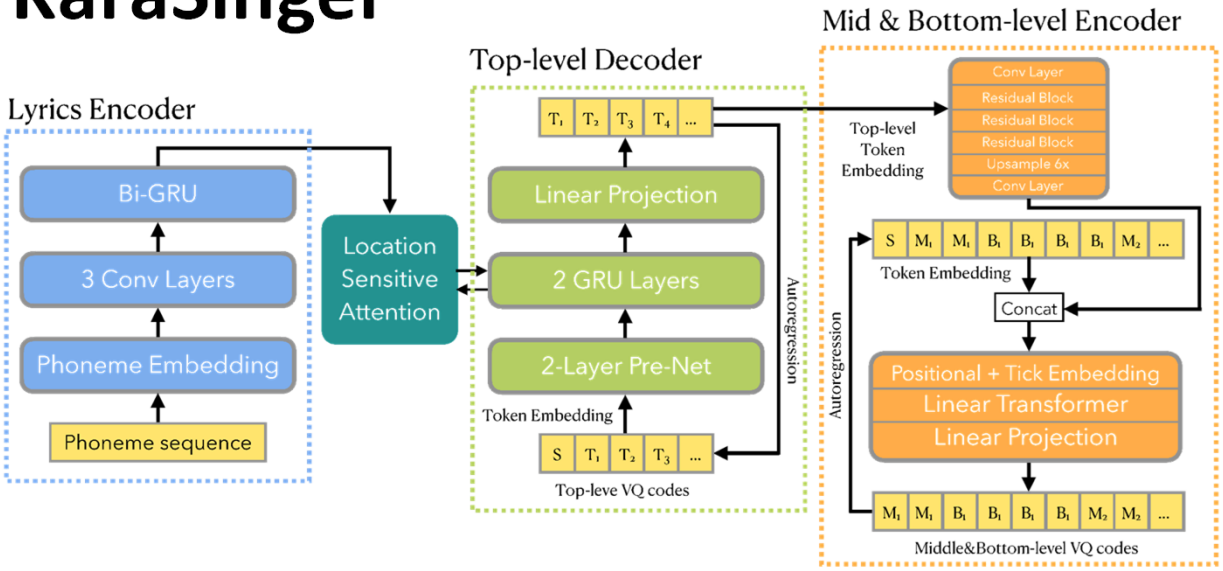
- Many SVS papers do not open source code…
  - For example: HiFiSinger, Nansy++

- One notable exception: **DiffSinger**
  - https://github.com/MoonInTheRiver/DiffSinger

Mel Pipeline	Dataset	Pitch Input	F0 Prediction	Acceleration Method	Vocoder
DiffSpeech (Text->F0, Text+F0->Mel, Mel->Wav)	Ljspeech	None	Explicit	Shallow Diffusion	HiFiGAN
DiffSinger (Lyric+F0->Mel, Mel->Wav)	PopCS	Ground-Truth F0	None	Shallow Diffusion	NSF-HiFiGAN
DiffSinger (Lyric+MIDI->F0, Lyric+F0->Mel, Mel->Wav)	OpenCpop	MIDI	Explicit	Shallow Diffusion	NSF-HiFiGAN
FFT-Singer (Lyric+MIDI->F0, Lyric+F0->Mel, Mel->Wav)	OpenCpop	MIDI	Explicit	Invalid	NSF-HiFiGAN
DiffSinger (Lyric+MIDI->Mel, Mel->Wav)	OpenCpop	MIDI	Implicit	None	Pitch-Extractor + NSF-HiFiGAN
DiffSinger+PNDM (Lyric+MIDI->Mel, Mel->Wav)	OpenCpop	MIDI	Implicit	PLMS	Pitch-Extractor + NSF-HiFiGAN
DiffSpeech+PNDM (Text->Mel, Mel->Wav)	Ljspeech	None	Implicit	PLMS	HiFiGAN

- Autoregressive
  - Based on TacoTron 2
- Two modes
  - **MIDI-free** (https://jerrygood0703.github.io/KaraSinger/)
  - **MIDI-free & lyrics-free** (unpublished)





Ref 1: Liao et al, “KaraSinger: Score-free singing voice synthesis with VQ-VAE using Mel-spectrograms,” ICASSP 2022

Ref 2: Liu et al, “Score and lyrics-free singing voice generation,” ICCC 2020

Outline

- Singing voice processing: Historical review
- Neural text-to-speech synthesis
- Neural singing voice synthesis
- **Build your SVS model**

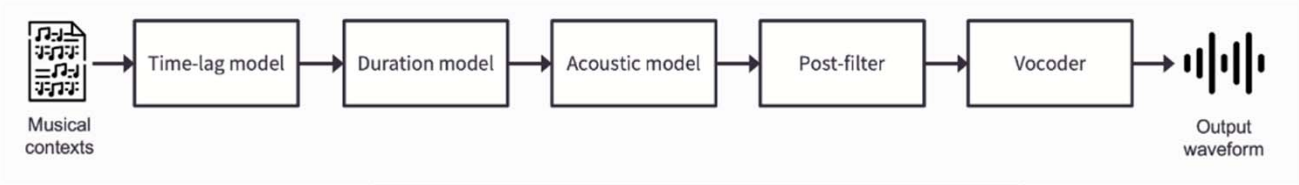
Datasets

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	✓	✓

Ref: Zhang et al, "M4Singer: A multi-style, multi-singer and musical score provided Mandarin singing corpus," NeurIPS dataset track paper, 2022

Library for SVS: NNSVS

https://github.com/nnsvs/nnsvs
https://nnsvs.github.io/
https://r9y9.github.io/projects/nnsvs/



Stage 0: Data preparation

Stage 1: Feature generation

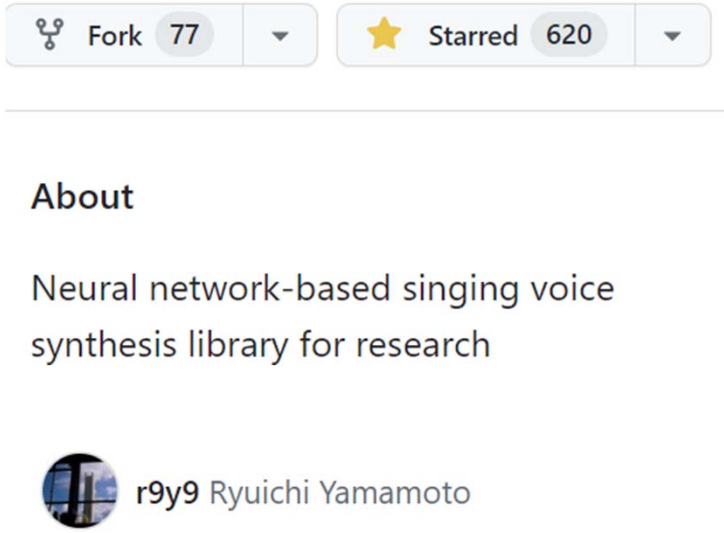
Stage 2: Train time-lag model

Stage 3: Train duration model

Stage 4: Train acoustic model

Stage 5: Generate features

Stage 6: Synthesis waveforms



Ref: Yamamoto et al, "NNSVS: A neural network-based singing voice synthesis toolkit," ICASSP 2023

Library for SVS: Muskits

https://github.com/SJTMusicTeam/Muskits

- **Various network architecutres** for end-to-end SVS
  - RNN-based non-autoregressive model
  - Xiaoice
  - Sequence-to-sequence Transformer (with GLU-based encoder)
  - MLP singer
  - Tacotron-singing (in progress)
  - DiffSinger (to be published)
- Multi-speaker & Multilingual extention
  - Speaker ID embedding
  - Language ID embedding
  - Global sytle token (GST) embedding
- Various language support
  - Jp / En / Kr / Zh

About

An opensource music processing toolkit

Table 1: *Details of experimental data: N_{src}, N_{lang}, N_{singer} stands for number of databases, languages, and singers, respectively.*

Task	N_{src}, N_{lang}, N_{singer}	Duration(h)		
		Train	Dev	Test
Single-singer	1,1,1	0.63	0.08	0.07
Multi-singer	4,1,4	3.01	0.34	0.38
Multilingual	6,4,6	11.72	0.49	1.01
Transfer	1,1,1	0.71	0.02	0.05

Ref: Shi et al, “Muskits: an end-to-end music processing toolkit for singing voice synthesis,” INTERSPEECH 2022

Useful Tools

- Grapheme-to-phoneme conversion
  - **phonemizer**: https://github.com/bootphon/phonemizer
  - **g2p**: https://github.com/Kyubyong/g2p

- Forced alignment
  - **Montreal Forced Aligner**: https://github.com/MontrealCorpusTools/Montreal-Forced-Aligner

Library for SVC: so-vits-svc

https://github.com/svc-develop-team/so-vits-svc