

2024 edition

Deep Learning for Music Analysis and Generation

Singing Voice Generation

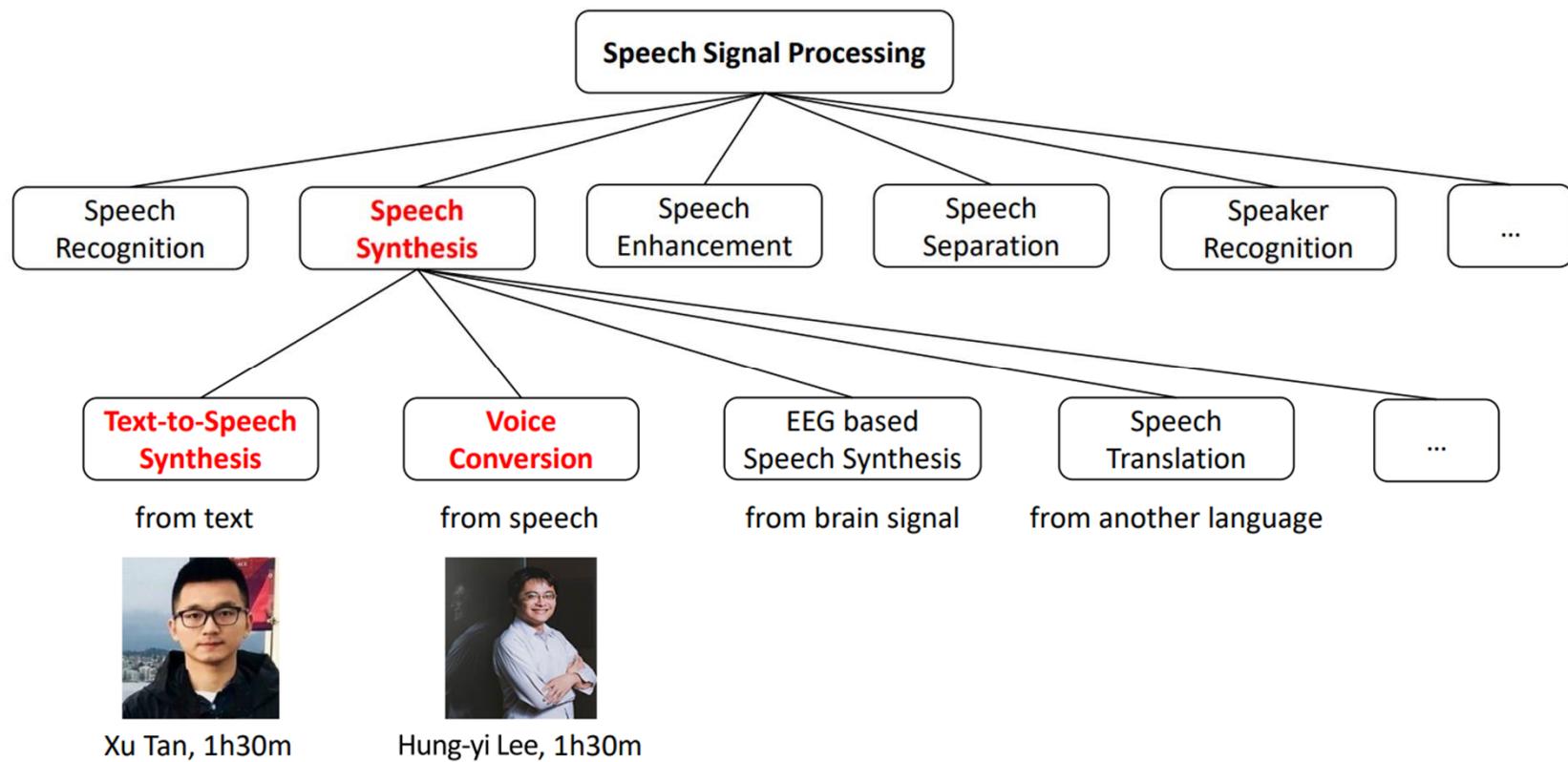
(text+MIDI → singing audio)



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

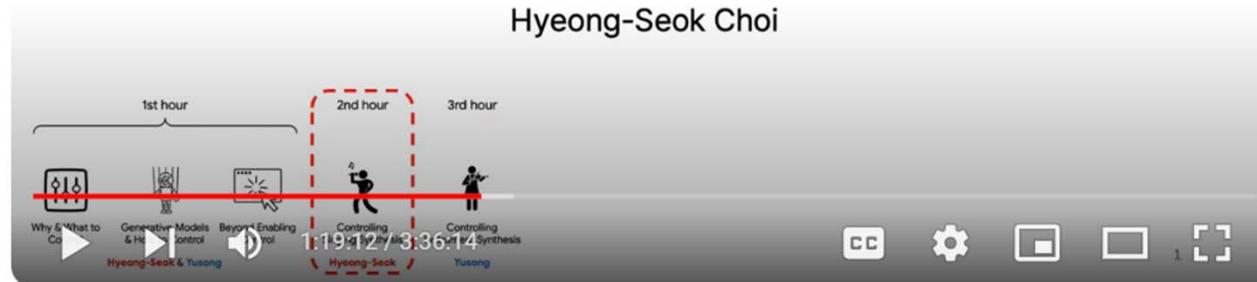
<https://github.com/lukewys/ISMIR2022-tutorial>

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



Designing Controllable Singing Voice Synthesis System

Hyeong-Seok Choi



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



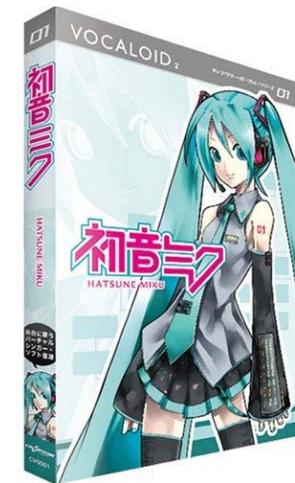
Tutorial 1. Why singing is interesting

(Simon Dixon, Masataka Goto, Matthias Mauch)

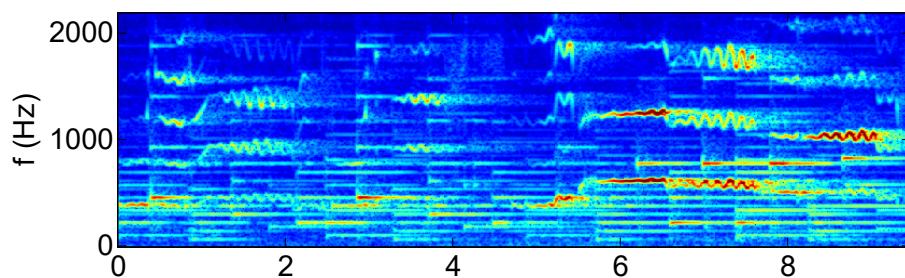
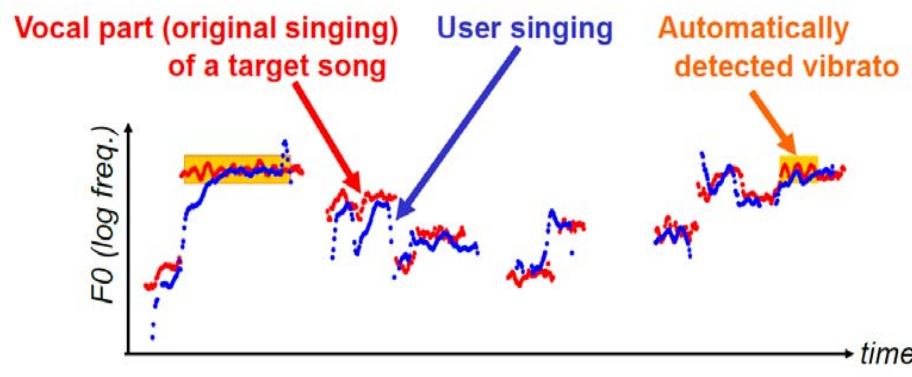
- All popular music cultures around the world use singing
- The singing voice is the most expressive of all musical instruments
- Singing voice produces both sound and words
- Various commercial applications

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)



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

<https://songrium.jp/>



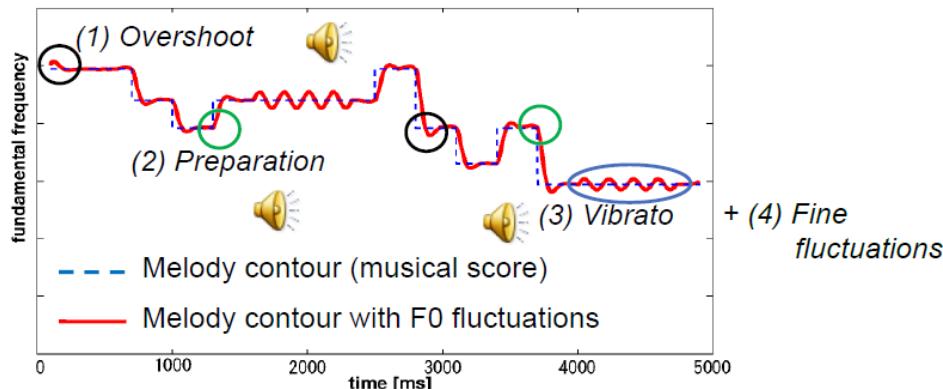
Ref: Kajita et al, "Songrium: Browsing and listening environment for music content creation community," SMC 2015

Singing Voice Generation

- Singing voice synthesis (SVS)



- Speech-to-singing synthesis

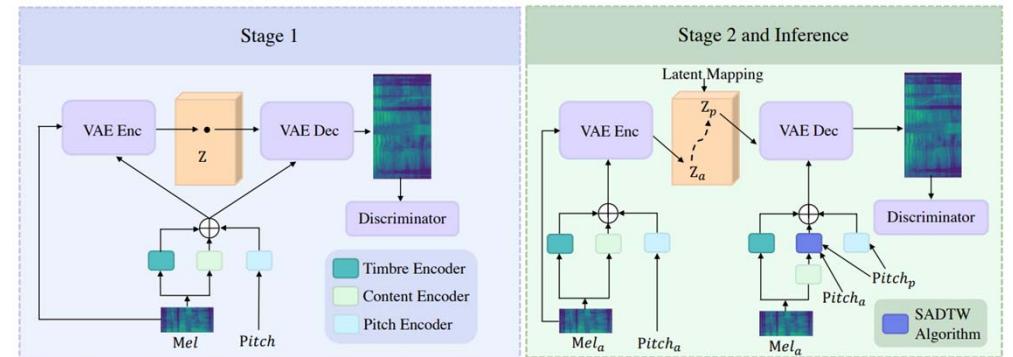


Ref: Saitou et al, "Speech-to-singing synthesis: converting speaking voices to singing voices by controlling acoustic features unique to singing voices," WASPAA 2007

- Singing voice conversion (SVC)



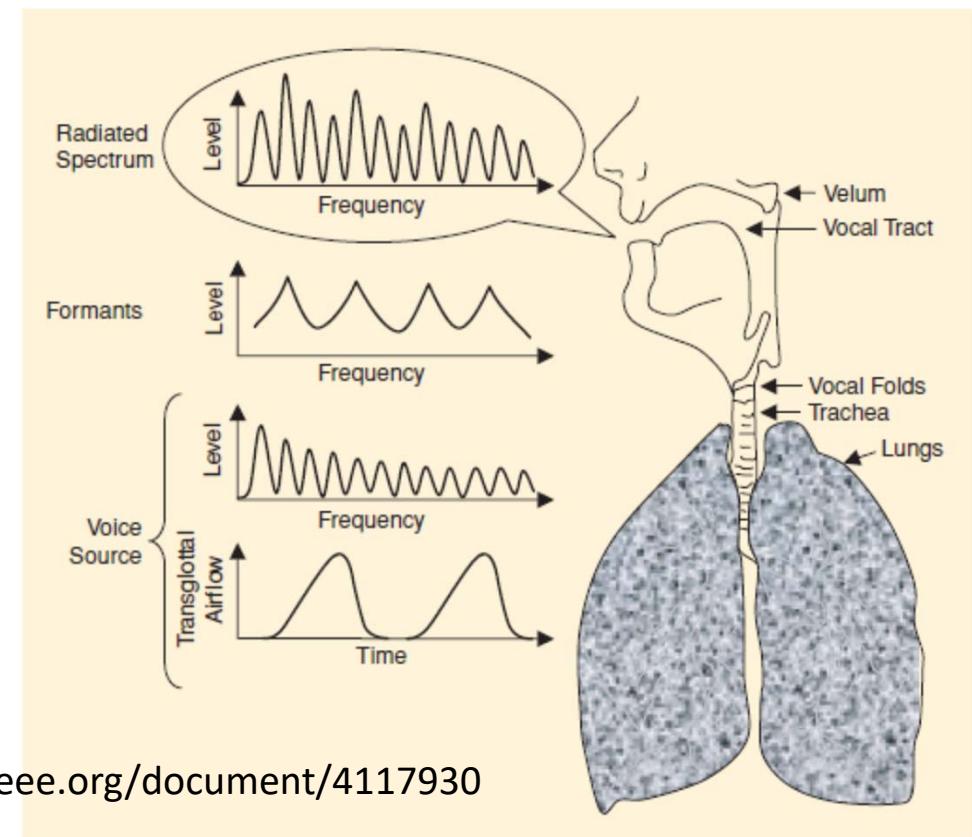
- Singing voice beautification



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

Singing Generation Tasks: Singing Voice Synthesis

- **text+MIDI → singing audio**
- Cook, “Singing voice synthesis: History, current work, and future directions,”
CMJ 1996
<https://www.jstor.org/stable/3680822?seq=6>



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

Vocaloid

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



Singing Voice Synthesis in the Past

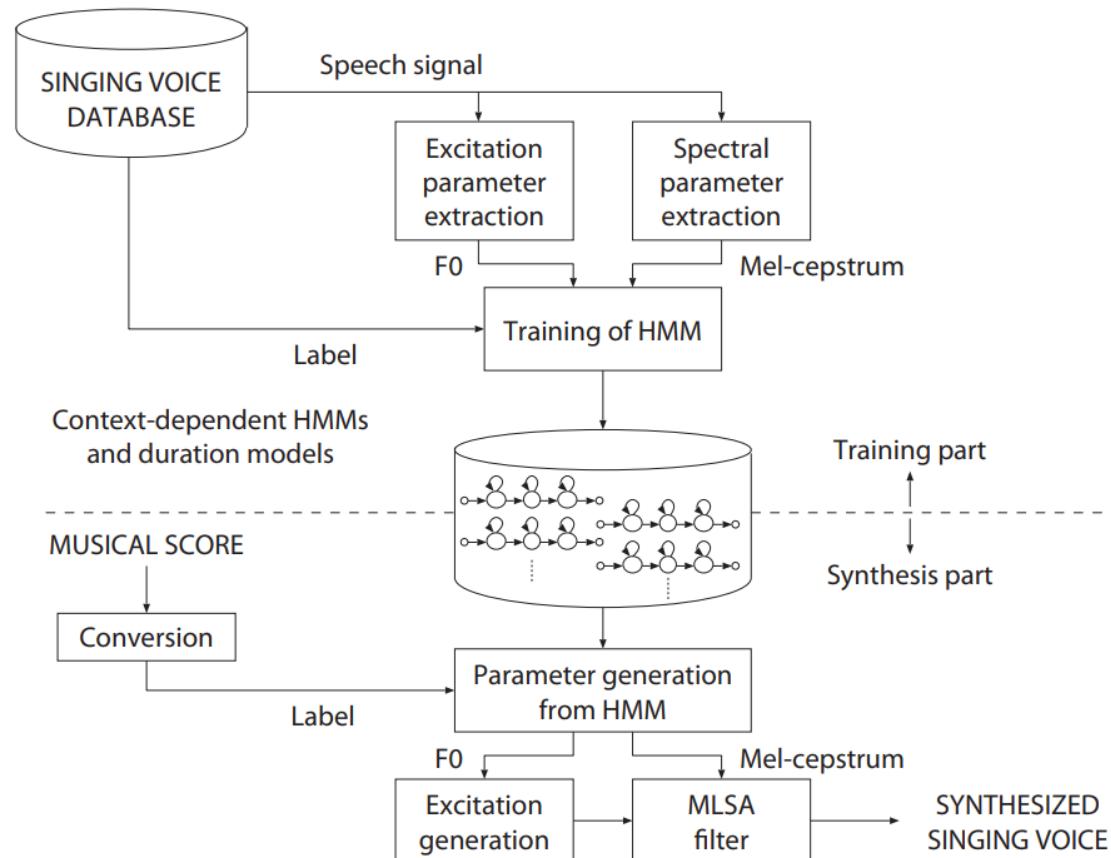
1. Human knowledge
 2. Lot's of signal processing
 3. Few parameters
-
- Great success, but characteristic ***robotic*** sound



Singing Voice Synthesis in 2010

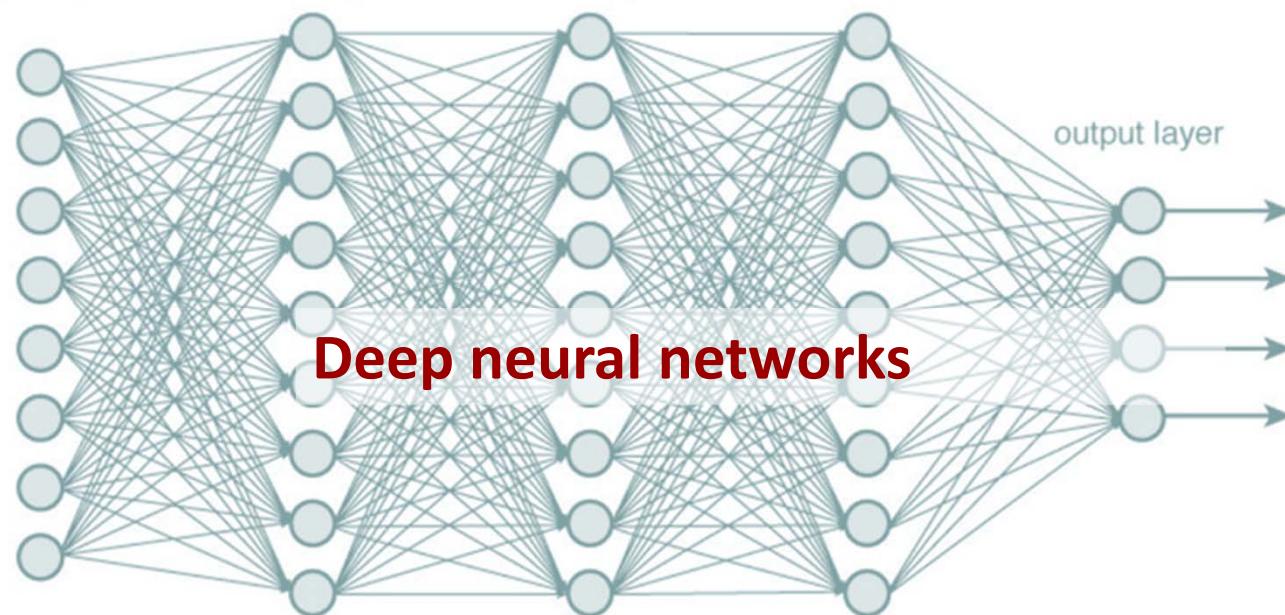
1. Human knowledge
2. Lot's of signal processing
3. Few parameters

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



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
 - 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

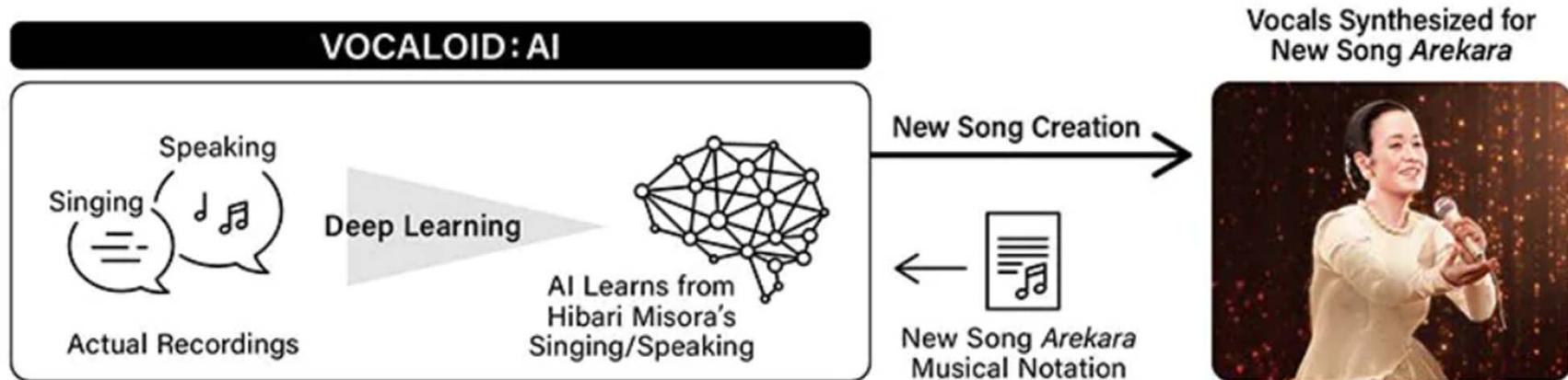
HYBE ACQUIRES FAKE VOICE AI COMPANY SUPERTONE IN \$32M DEAL (REPORT)



OCTOBER 4, 2022

BY MURRAY STASSEN

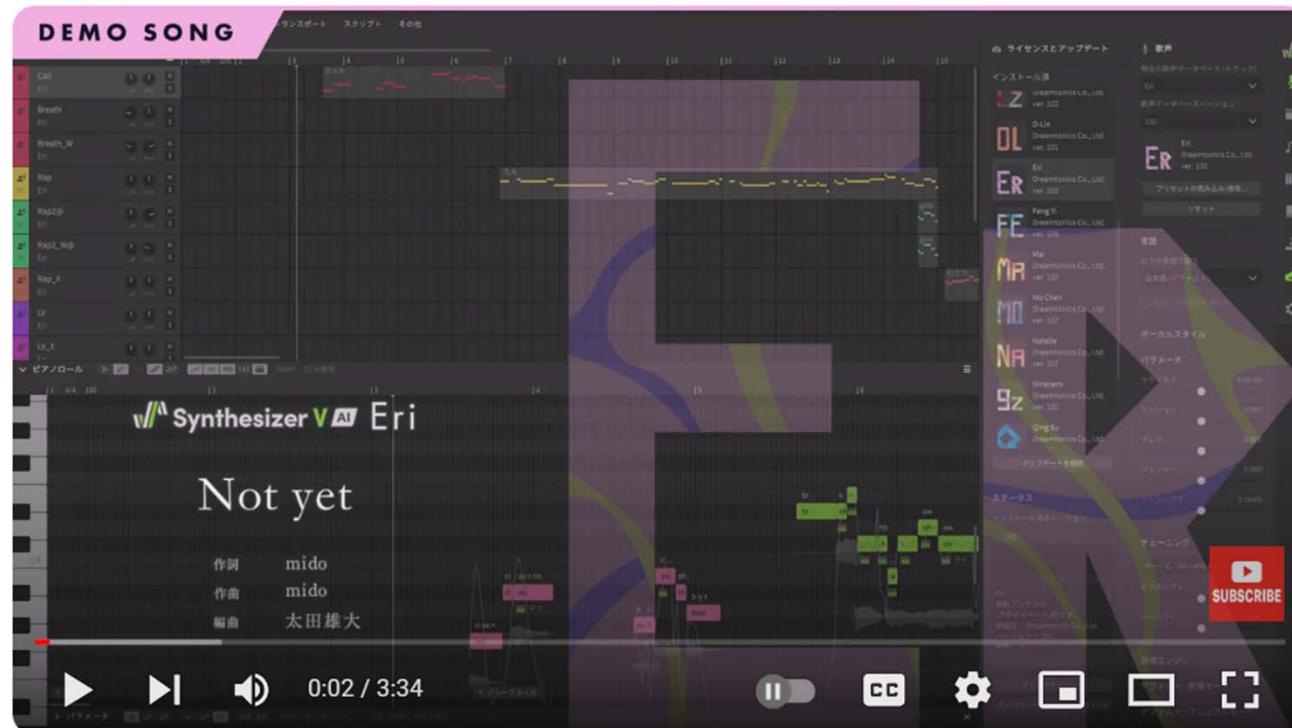
VOCALOID 6



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

SynthV

https://www.youtube.com/watch?v=Yrs4rF_V92s



Synthesizer V AI Eri Demo - Not yet (Full Version)



Dreamtonics Co., Ltd.
7.93K subscribers

Subscribe

314



Share

...

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://studio.yating.tw/music/ai-vocal>

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

The screenshot shows the 'AI 主唱' (AI Vocal) interface. At the top, it says 'AI 主唱' with a help icon, and '上傳 MIDI 檔案，讓 AI 成為您的專屬主唱。' Below that is a button to '上傳背景音樂 (選擇性)' (Upload background music). A file list shows 'Demo 1 - Main.mid' with a '重新上傳' (Re-upload) button. A dropdown menu shows 'Female AI 2.0' and a '試聽音色' (Listen to sound color) button. To the right, lyrics are displayed in a box:

= 我們勇敢唱，堅持的精神
我們一起唱，歌聲的真心
希望#的星光，閃耀在夜晚

* 可使用轉音指令符#, 主唱尚未支援英文

At the bottom, a grid shows a musical score with lyrics overlaid. The lyrics are: 壓持的精 神 我們一起唱 真的希望#的星光。 The score has measures 3, 4, 5, and 6.

Singing Synthesis at the Taiwan AI Labs

<https://studio.yating.tw/music/text-to-song>

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



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

SVS

- **text+MIDI → singing audio
(score → performance + sounds)**

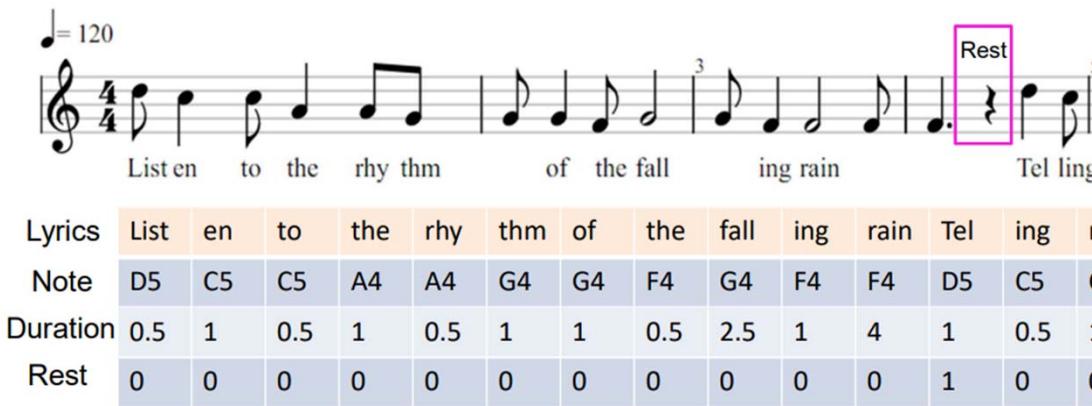
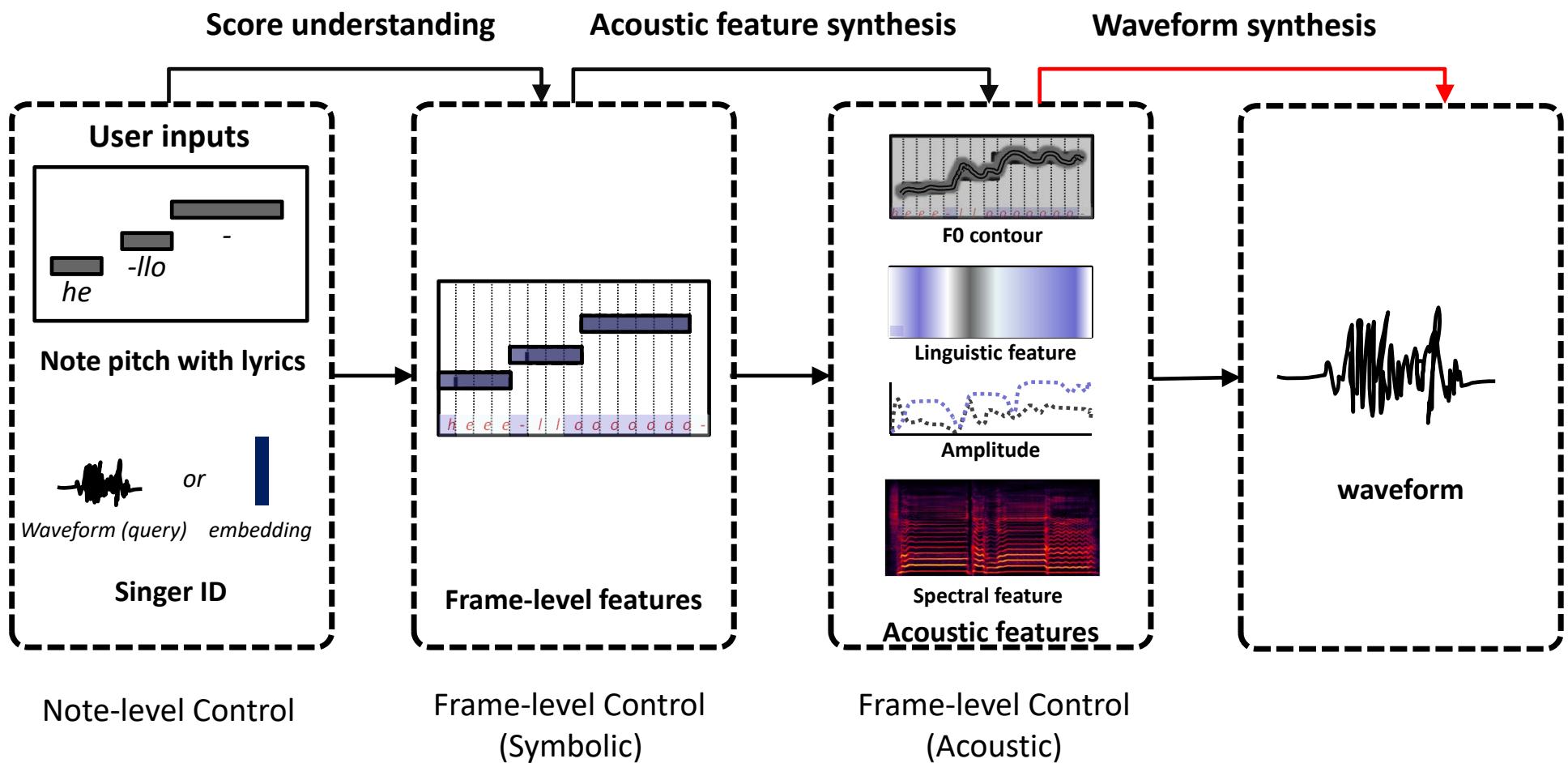


Fig. 1: An example of alignment between lyrics and melody

- “Performance” attributes to predict from the score
 - **F0** [MIDI pitch → hz]
 - **onset time** and **offset time** (in ms) [**musical time → physical time**]
 - singing techniques (optional)

SVS



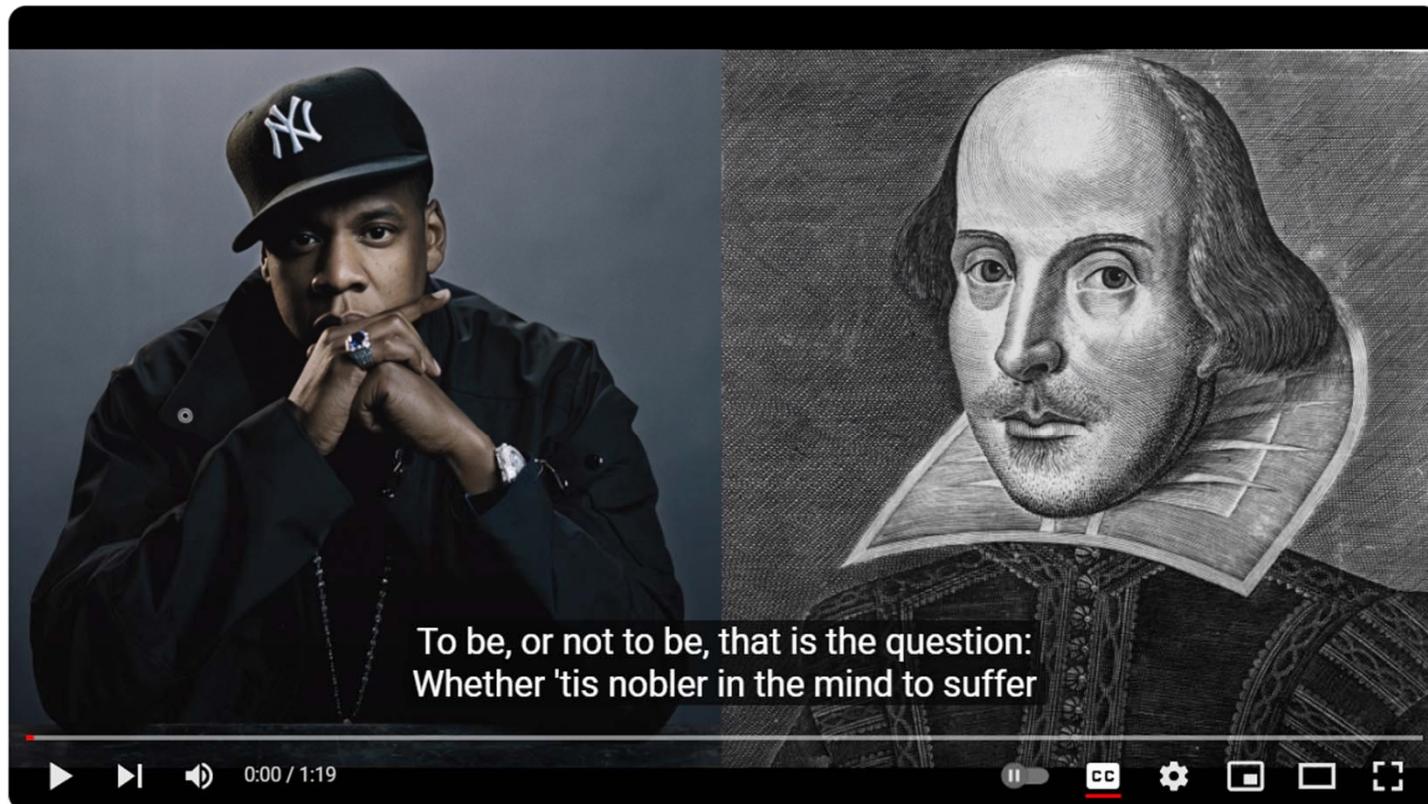
Adapted from the slides of the ISMIR 2022 tutorial presented by Hyeong-Seok Choi

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 predict the **onset** and **offset** of each word or syllable
- Differences
 - Musical expressivity (**f0**, dynamics, singing techniques)
 - Cost to collect data

TTS vs. SVS

- **TTS:**
 - text → speech audio
 - need to predict **onset time** and **offset time** of each word or syllable
- **SVS:**
 - text+MIDI → singing audio
 - need to predict **onset time**, **offset time**, and **f0** of each word or syllable

TTS

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

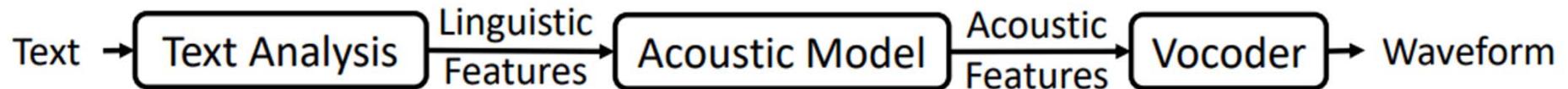


Figure 1: The three key components in neural TTS.

- 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** (or directly generate waveform)

Character vs Phoneme

- **Character**/spelling (grapheme)
 - Used in LLM, ASR ...
- Pronunciation (**phoneme**)
 - Used in TTS, SVS

User: Lyrics → Syllable ↔ Note

Example:

“to a place” → **User (syllabify)** → “to” “a” “place”
→ **Grapheme2Phoneme (G2P)** → “tu” “ə” “pleɪs”

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

- Question: For 1) melody-to-lyrics generation, and 2) lyrics-to-melody generation, do we prefer the character- or the phoneme-based representation for the lyrics?

Grapheme-to-phoneme Conversion Tools

- Grapheme-to-phoneme conversion
 - **phonemizer:** <https://github.com/bootphon/phonemizer>

	espeak	espeak-mbrola	festival	segments
phone set	IPA	SAMPA	custom	user defined
supported languages	100+	35	US English	user defined
processing speed	fast	slow	very slow	fast
phone tokens	✓	✓	✓	✓
syllable tokens	✗	✗	✓	✗
word tokens	✓	✗	✓	✓
punctuation preservation	✓	✗	✓	✓
stressed phones	✓	✗	✗	✗
tie	✓	✗	✗	✗

- **g2p:** <https://github.com/Kyubyong/g2p>

Phonemeizer for melody-to-lyrics generation

- **Strong-beat notes and weak-beat notes** refer to melody notes beginning at strong and weak beats, respectively. In 4/4 time music, the first and third beats are stronger than the second and fourth beats.
- **Long notes and short notes:** long notes are notes with relatively longer duration. Based on [Nichols *et al.*, 2009], we define long notes as those whose duration is greater than the average duration of all notes in a melody phrase, with all non-long notes being short notes.
- **Stressed syllables and weak syllables:** stressed syllables are accented when pronounced and are marked by special symbols ([˘], [˘]) in the international phonetic alphabet (IPA)¹; Weak syllables (a.k.a. unstressed syllables) do not have special markers in their IPA notations.
- **Long syllables and short syllables:** Long syllables have long vowels (marked by [:] in IPA) or diphthongs. A diphthong is a syllable with two vowel sounds, for example, sad ['sæd], hey [heɪ], etc. Syllables without long vowels or diphthongs are short syllables.

Ref: Liang et al, “XAI-Lyricist,” IJCAI 2024

TTS

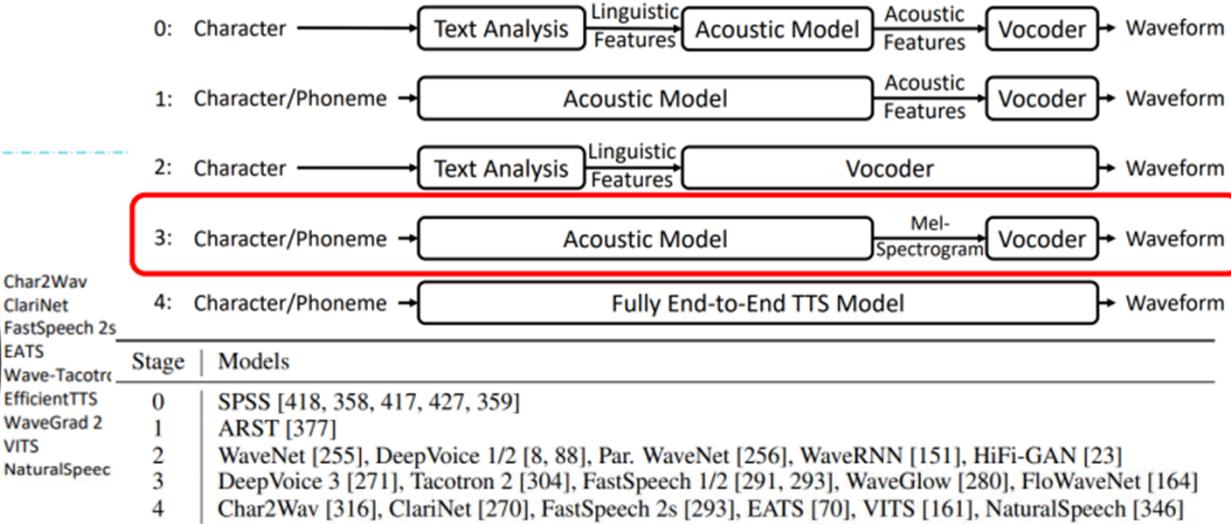
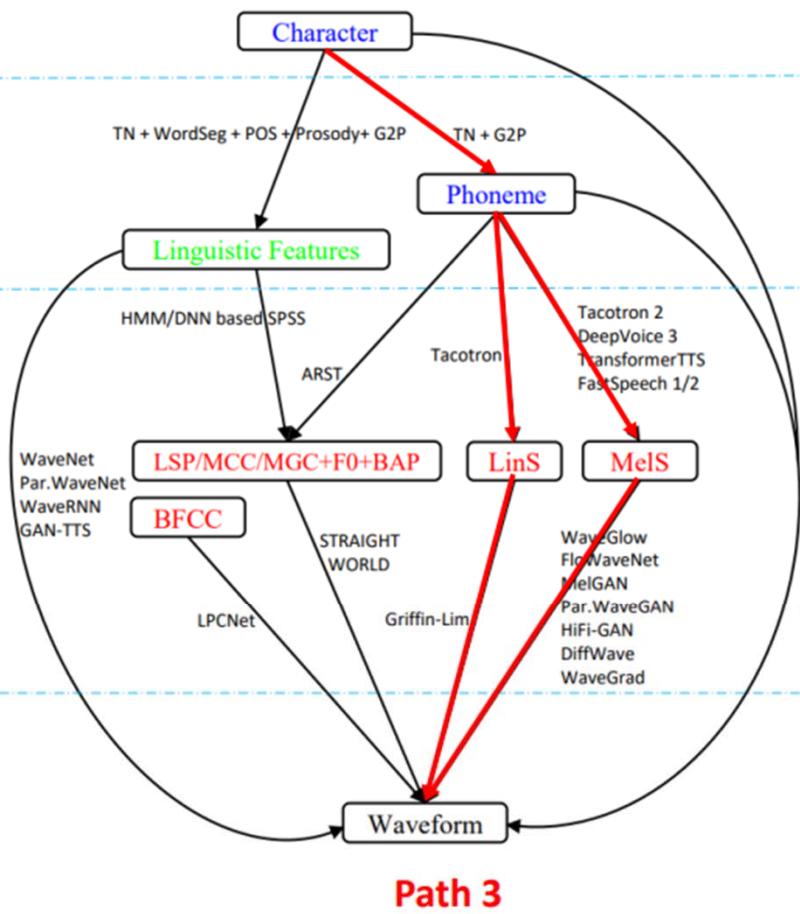
<https://github.com/tts-tutorial/interspeech2022>

Text

Linguistic Features

Acoustic Features

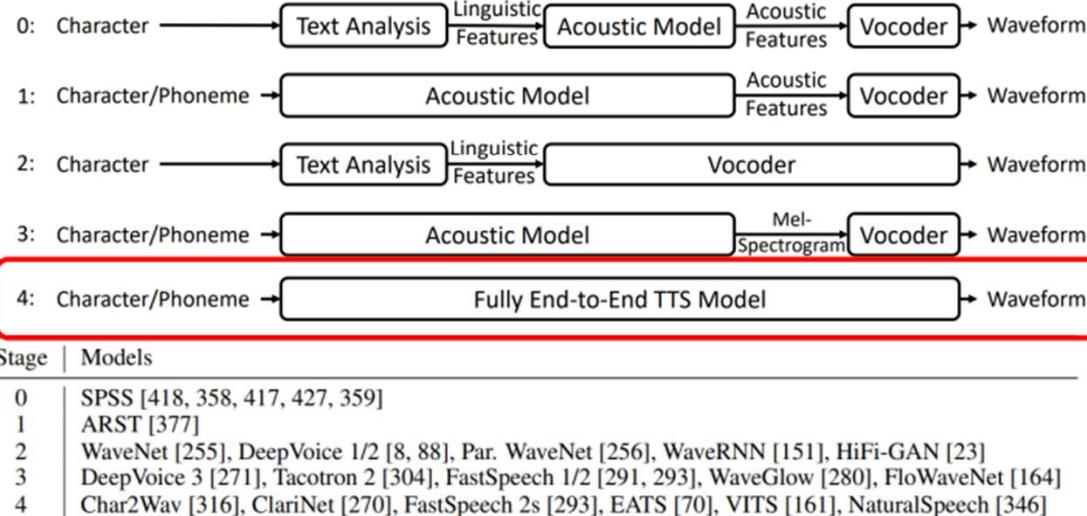
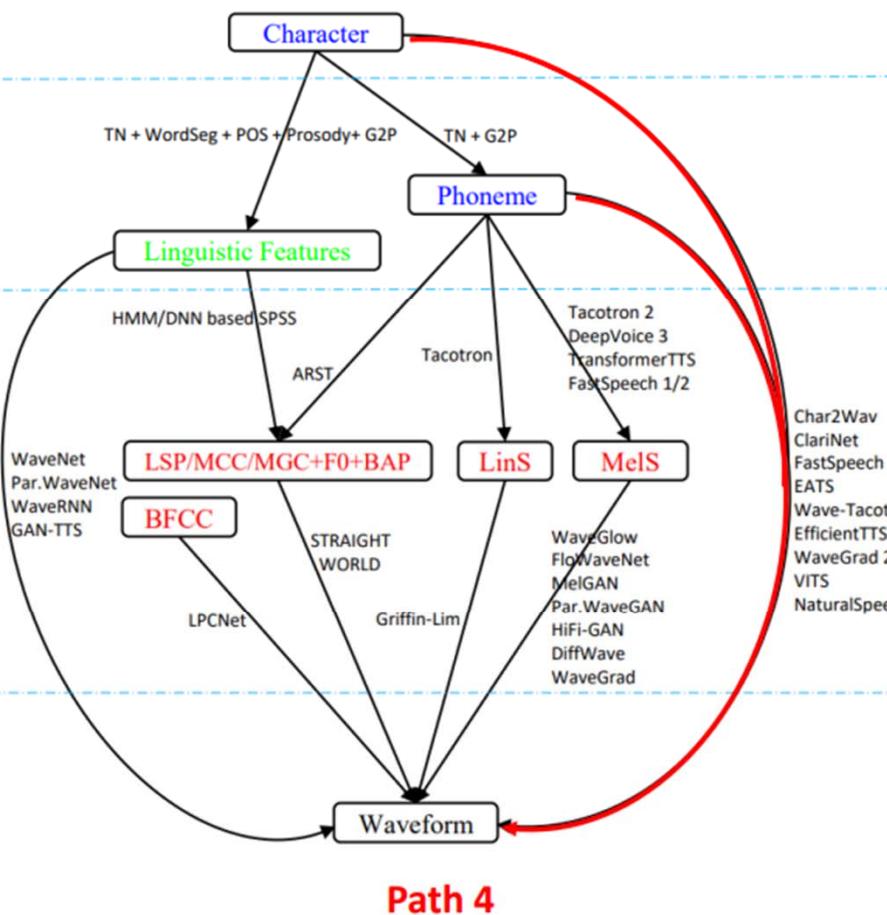
Waveform



TTS

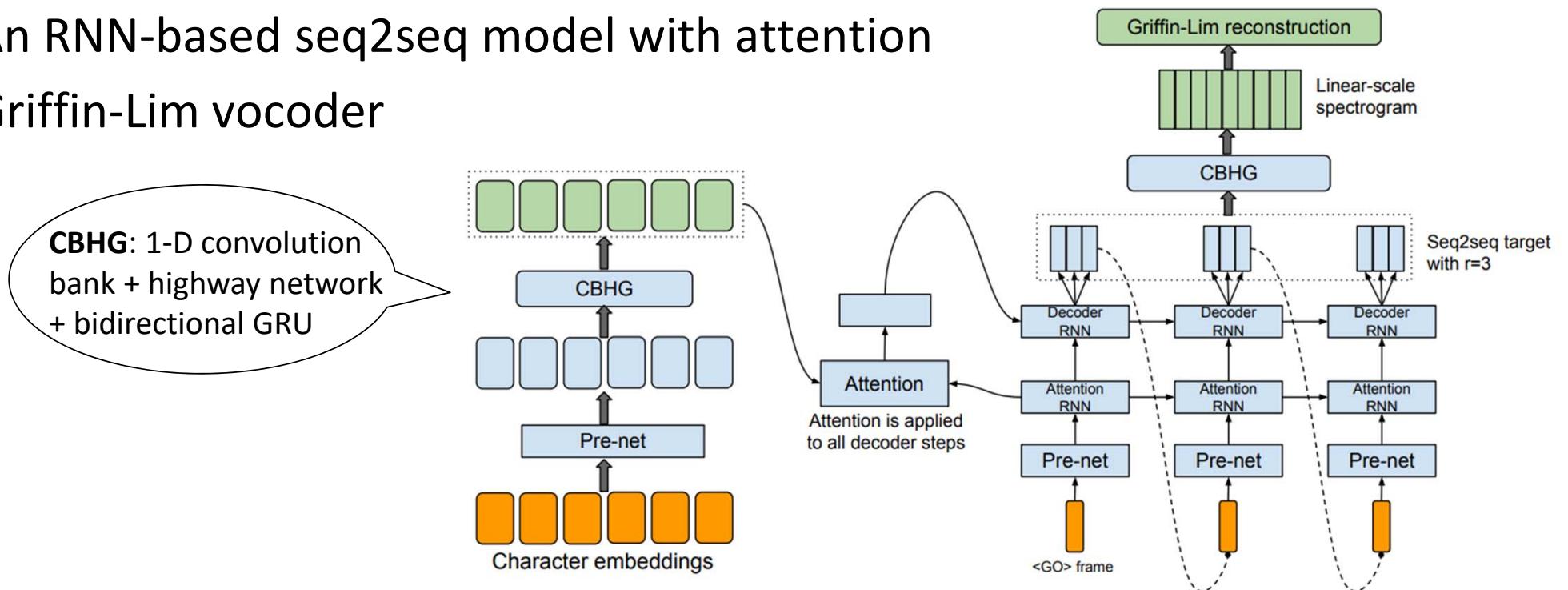
<https://github.com/tts-tutorial/interspeech2022>

Text



TacoTron

- Takes **characters** as input and **predict frames of (linear) spectrogram**
- **Autoregressive (AR)**
- An RNN-based seq2seq model with attention
- Griffin-Lim vocoder

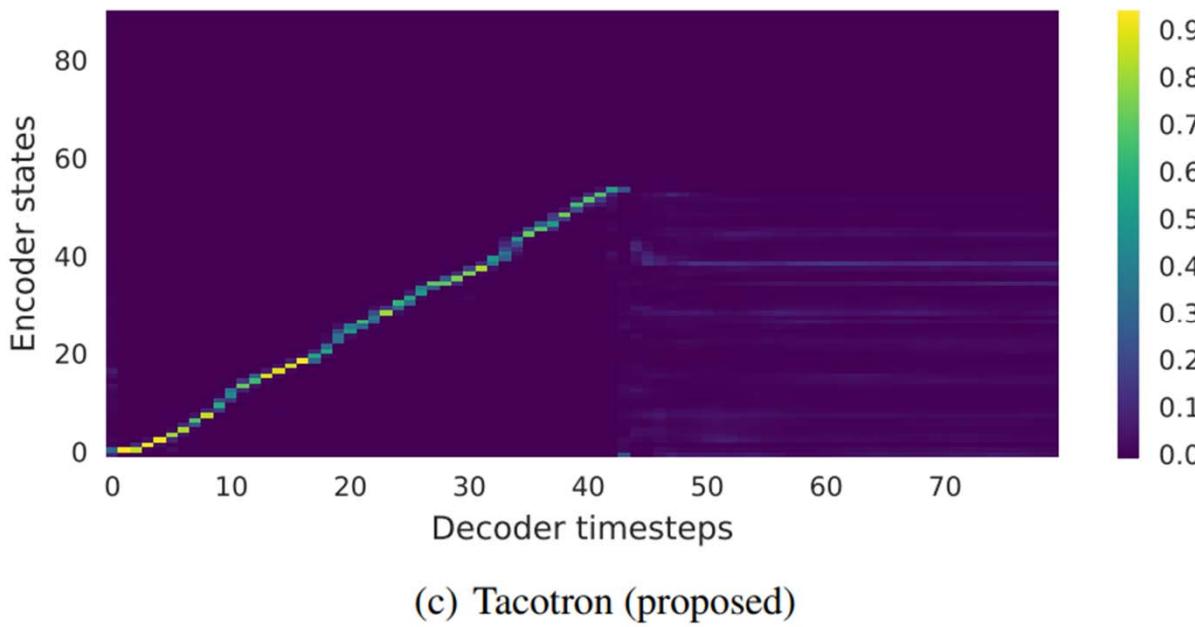


Ref: Wang et al, "Tacotron: Towards end-to-end speech synthesis," INTERSPEECH 2017

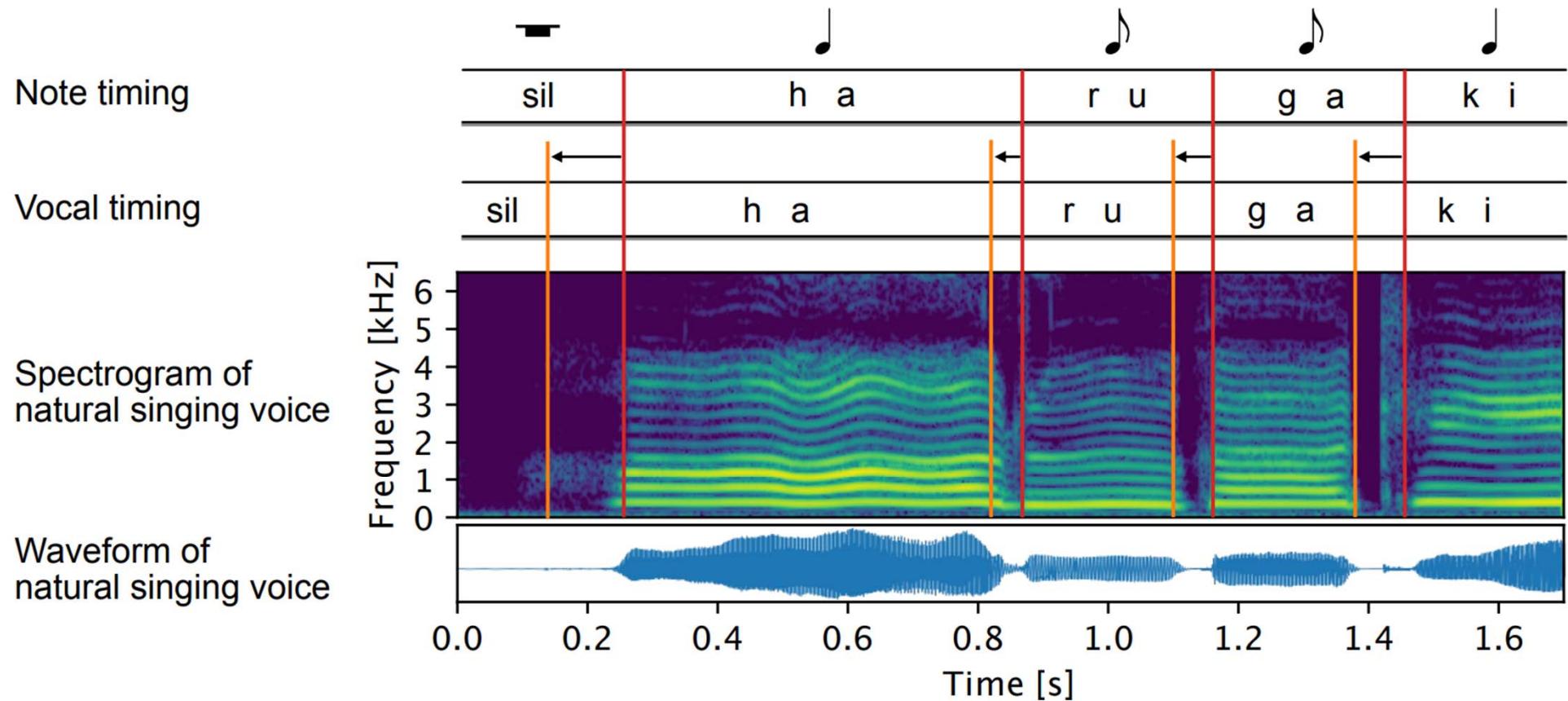
32

Why Attention?

- To learn the **alignment** between input text and output audio (i.e. to predict the onset/offset of each word or syllable)
- Unlike source separation and vocoder, for TTS & SVS, the input and output are of **different lengths**



Text-Audio Alignment

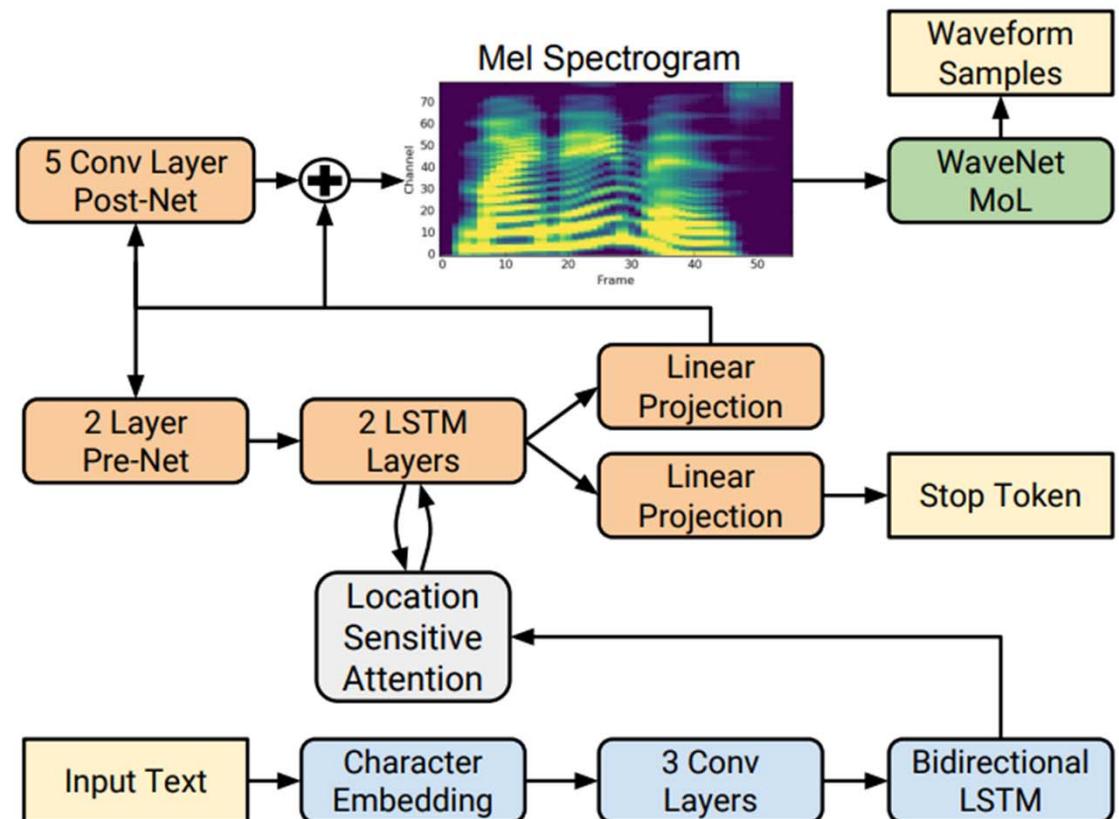


TacoTron 2

- Modifications

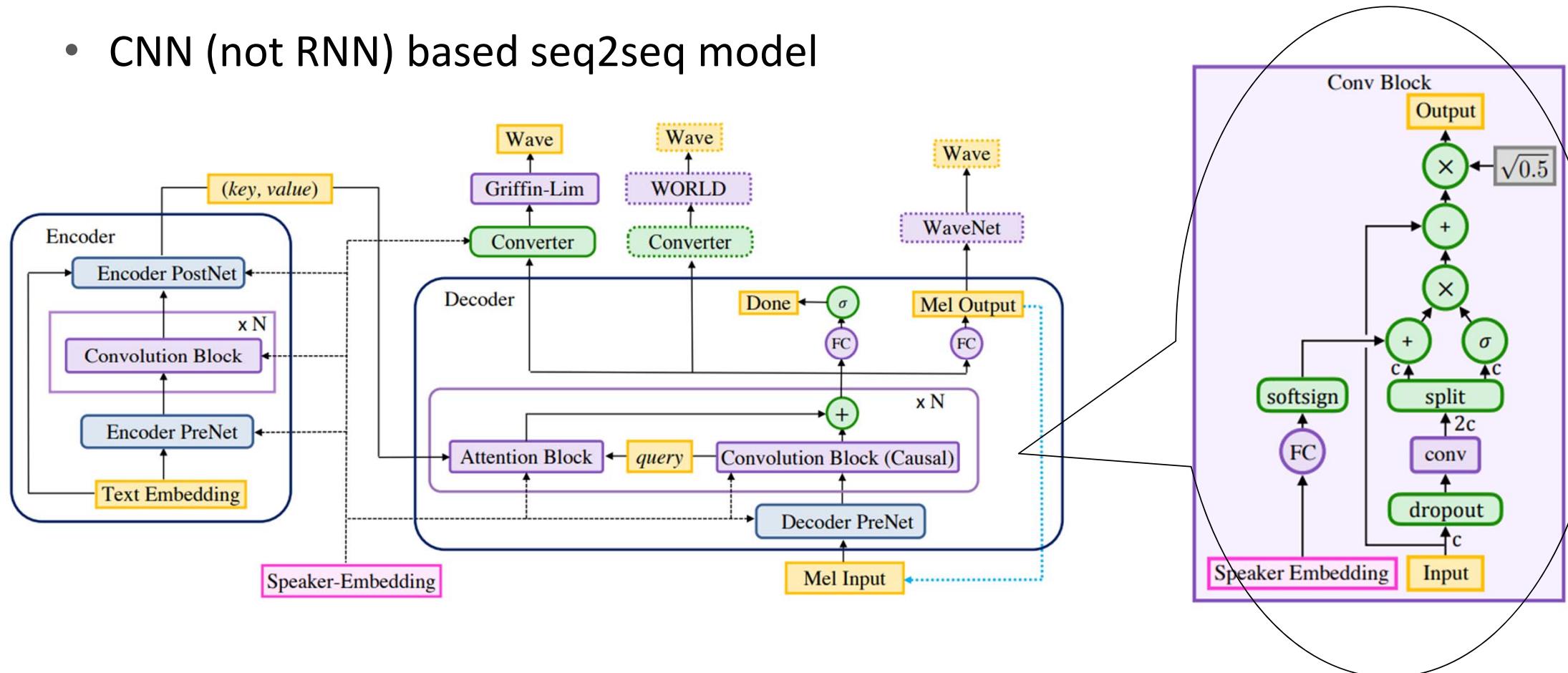
- Mel-spectrograms instead of linear spectrograms
- WaveNet-like Mel-vocoder
- Simpler architecture (no CBHG)
- Location sensitive attention
- “Stop token”

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)	4.526 ± 0.066



Deep Voice 3

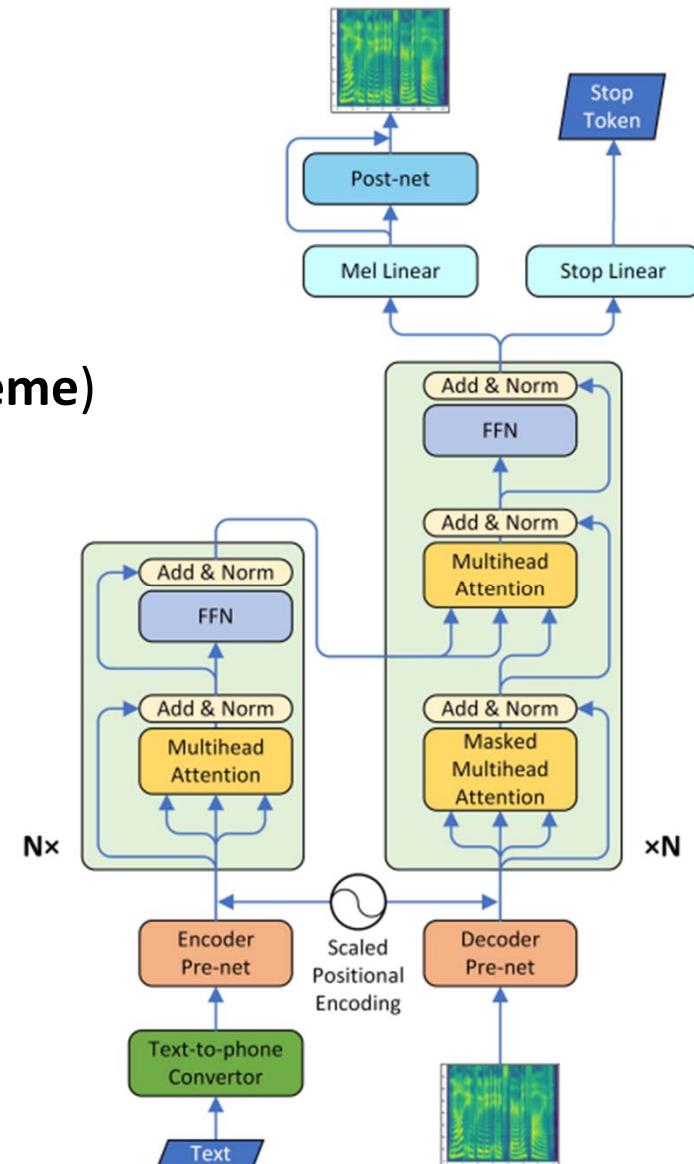
- CNN (not RNN) based seq2seq model



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	-



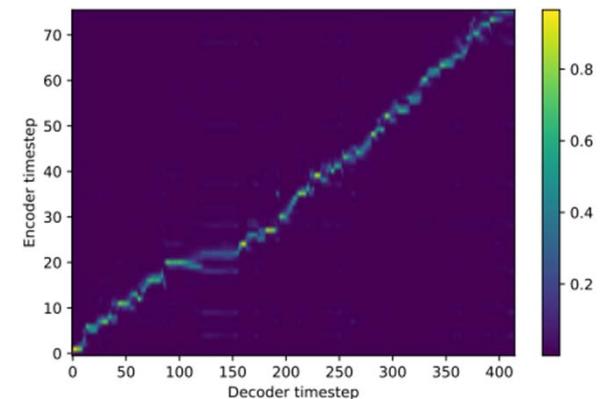
Ref: Li et al, "Neural speech synthesis with Transformer network," AAAI 2019

Drawbacks of AR-based Seq2seq 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

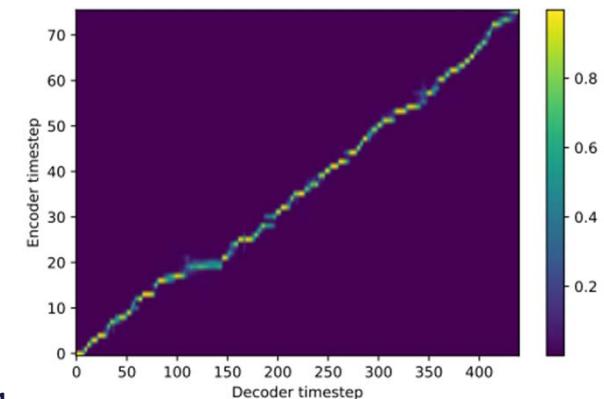
Robustness Problem About Alignment

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



(c) *Tacotron 2*

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



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

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

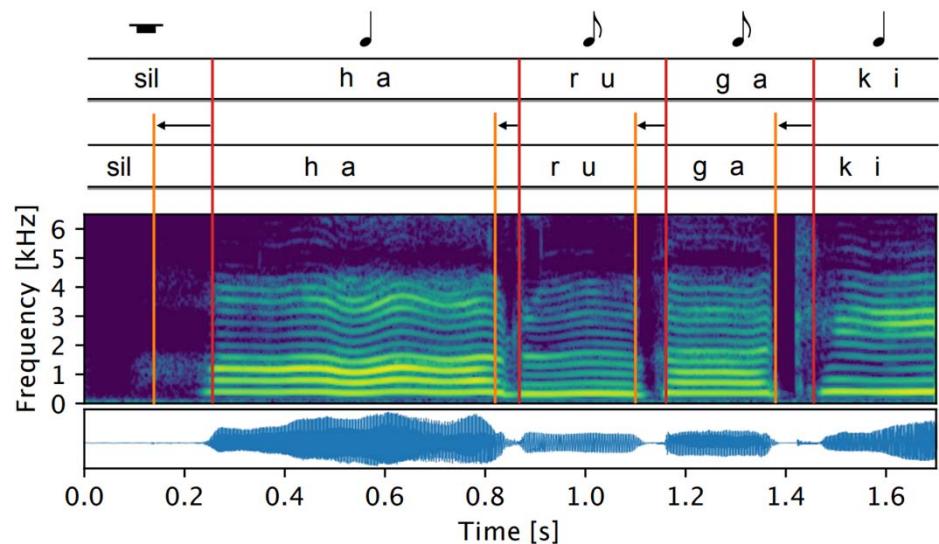
Ref 2: “One TTS alignment to rule them all,” arXiv 2021

Drawbacks of Autoregressive models

- Slow inference speed → Use non-autoregressive models!
- Synthesized speech is usually not robust → Predict phoneme duration!
(i.e. predict onset and offset times)
 - 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

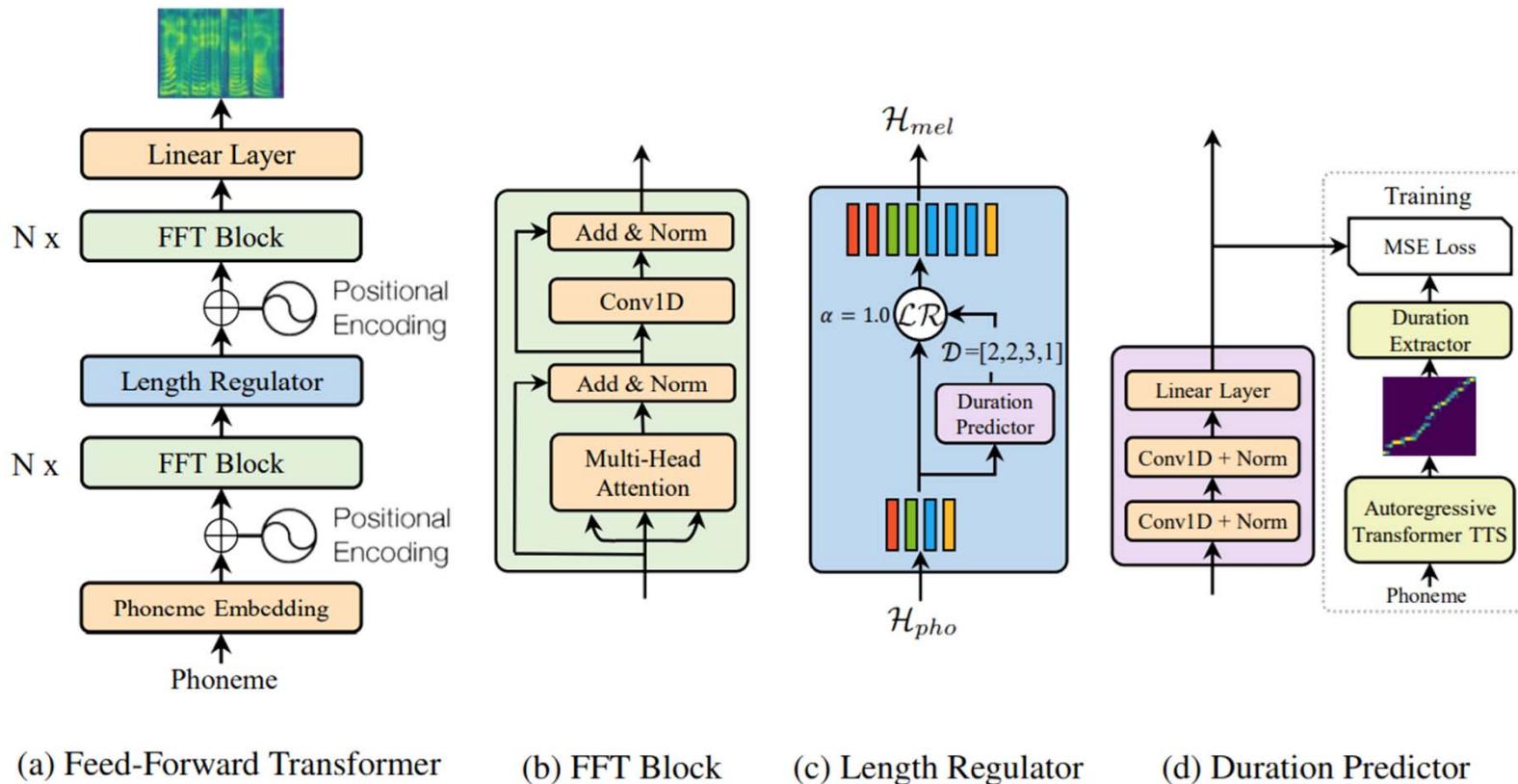
Predict Phoneme Duration?

- Cross-attention based alignment is too much because for TTS the alignment between input sequence and output sequence must be monotonic
 - For other tasks such as machine translation (MT), the alignment may not be monotonic, making cross-attention a good choice; but this is not the case here
- It's actually good enough to know the **duration** of each character or phoneme (e.g., how many frames, or how many time samples)



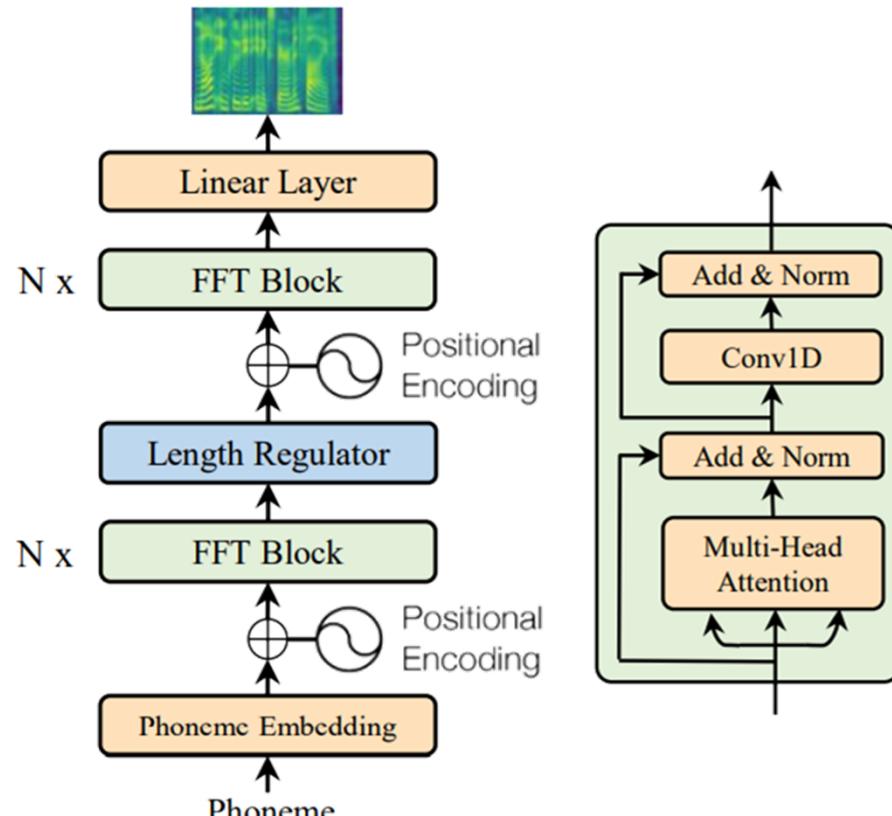
FastSpeech

<https://speechresearch.github.io/fastspeech/>



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

FastSpeech



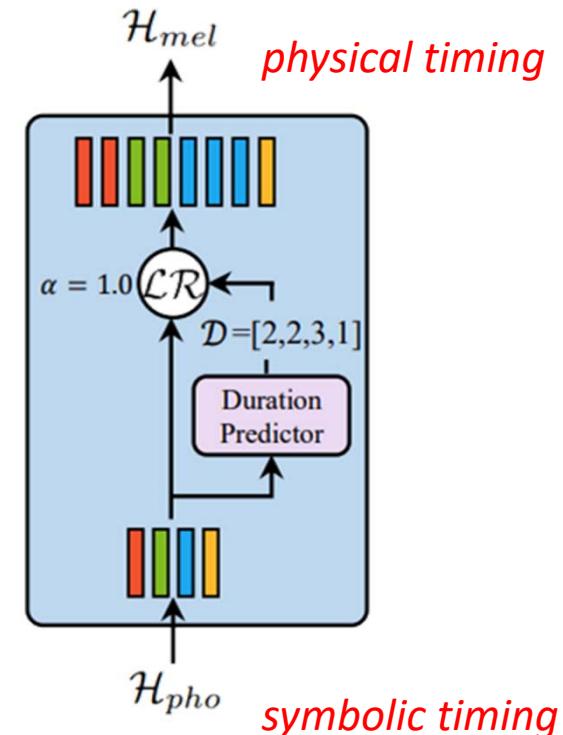
(a) Feed-Forward Transformer

(b) FFT Block

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

FastSpeech

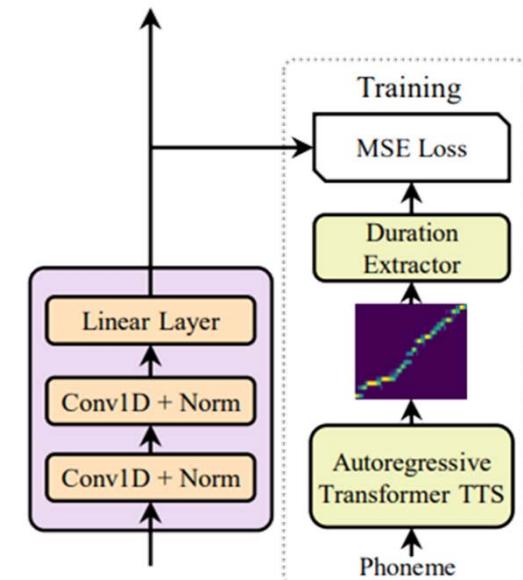
- **Length regulator** → *Robustness*
 - Up-samples the phoneme sequence according to the **phoneme duration** (i.e., the number of frames that each phoneme corresponds to) to match the length of the mel-spectrogram sequence
- The phoneme duration is from a **duration predictor**
 - Explicitly determine 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

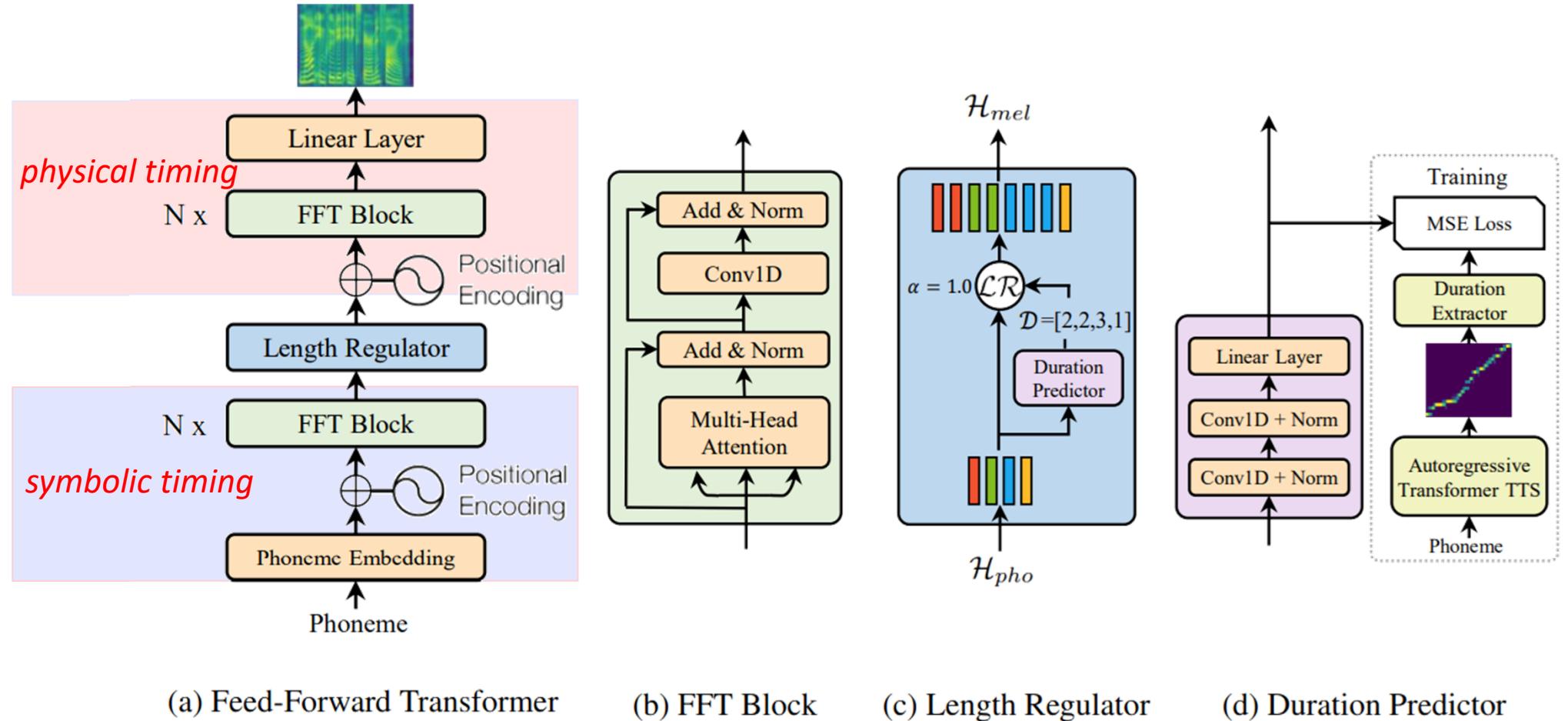
FastSpeech

- **Phoneme duration predictor**
 - Produces *hard alignments*
 - while AR models use soft and automatic attention alignments
 - 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)**
<https://github.com/MontrealCorpusTools/Montreal-Forced-Aligner>
 - Or by **manually labeling**



(d) Duration Predictor

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

46

FastSpeech: Evaluation Result

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

Table 1: The MOS with 95% confidence intervals.

Method	Latency (s)	Speedup
<i>Transformer TTS [14] (Mel)</i>	6.735 ± 3.969	/
<i>FastSpeech (Mel)</i>	0.025 ± 0.005	$269.40 \times$
<i>Transformer TTS [14] (Mel + WaveGlow)</i>	6.895 ± 3.969	/
<i>FastSpeech (Mel + WaveGlow)</i>	0.180 ± 0.078	$38.30 \times$

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
<i>Tacotron 2</i>	4	11	12	24%
<i>Transformer TTS</i>	7	15	17	34%
<i>FastSpeech</i>	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

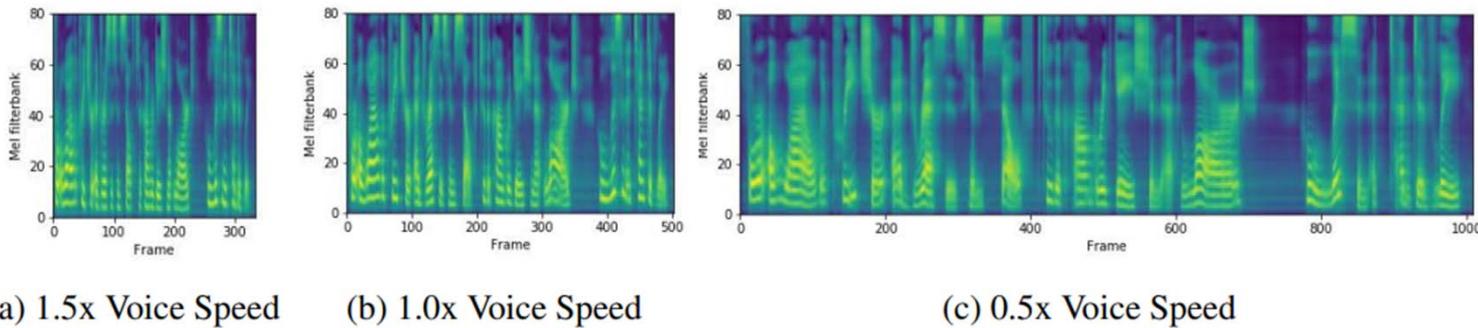


Figure 3: The mel-spectrograms of the voice with 1.5x, 1.0x and 0.5x speed respectively. The input text is "*For a while the preacher addresses himself to the congregation at large, who listen attentively*".

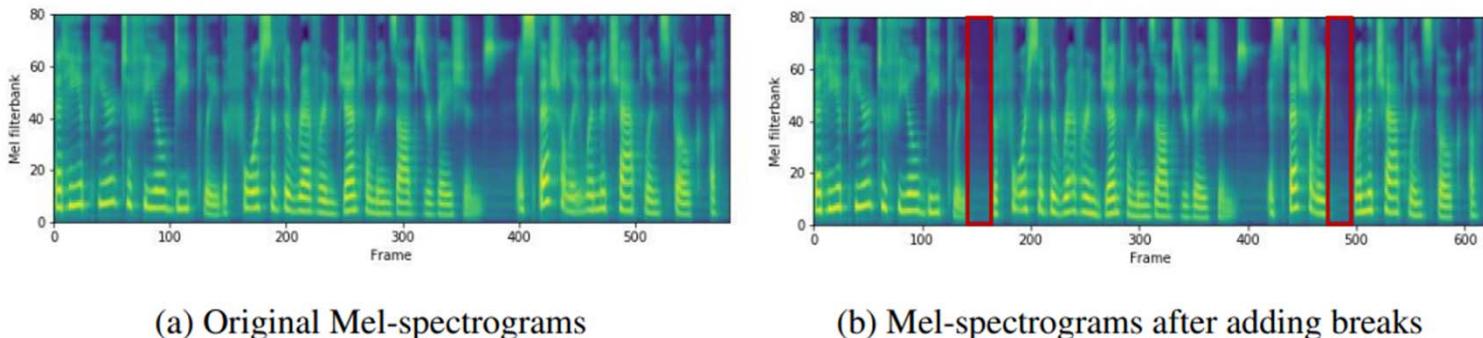
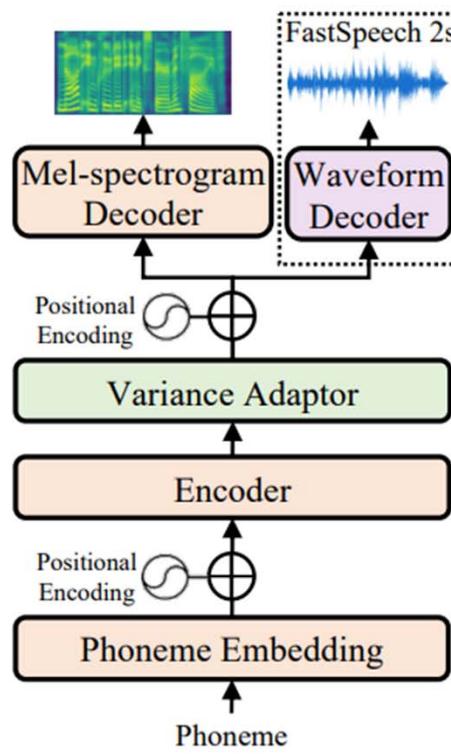


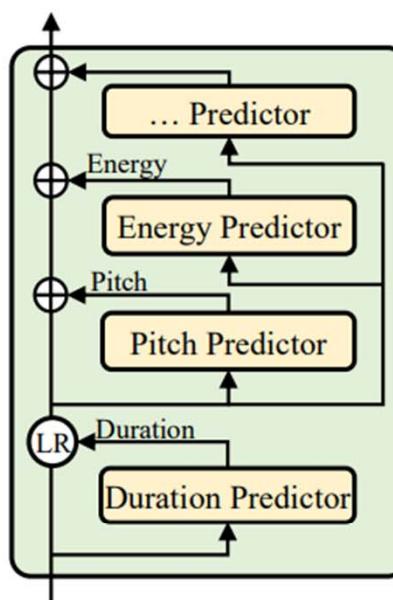
Figure 4: The mel-spectrograms before and after adding breaks between words. The corresponding text is "*that he appeared to feel **deeply** the force of the reverend gentleman's observations, especially when the chaplain spoke of*". We add breaks after the words "*deeply*" and "*especially*" to improve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

FastSpeech 2 & FastSpeech 2s

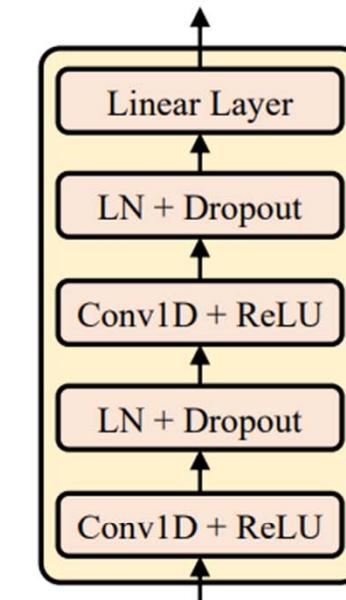
<https://speechresearch.github.io/fastspeech2/>



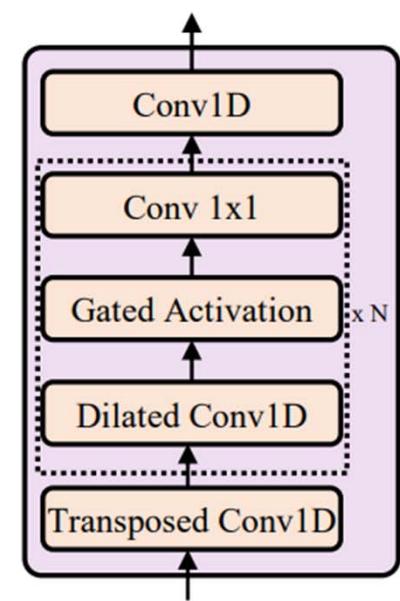
(a) FastSpeech 2



(b) Variance adaptor



(c)
Duration/pitch/energy
predictor



(d) Waveform decoder

FastSpeech 2 vs FastSpeech

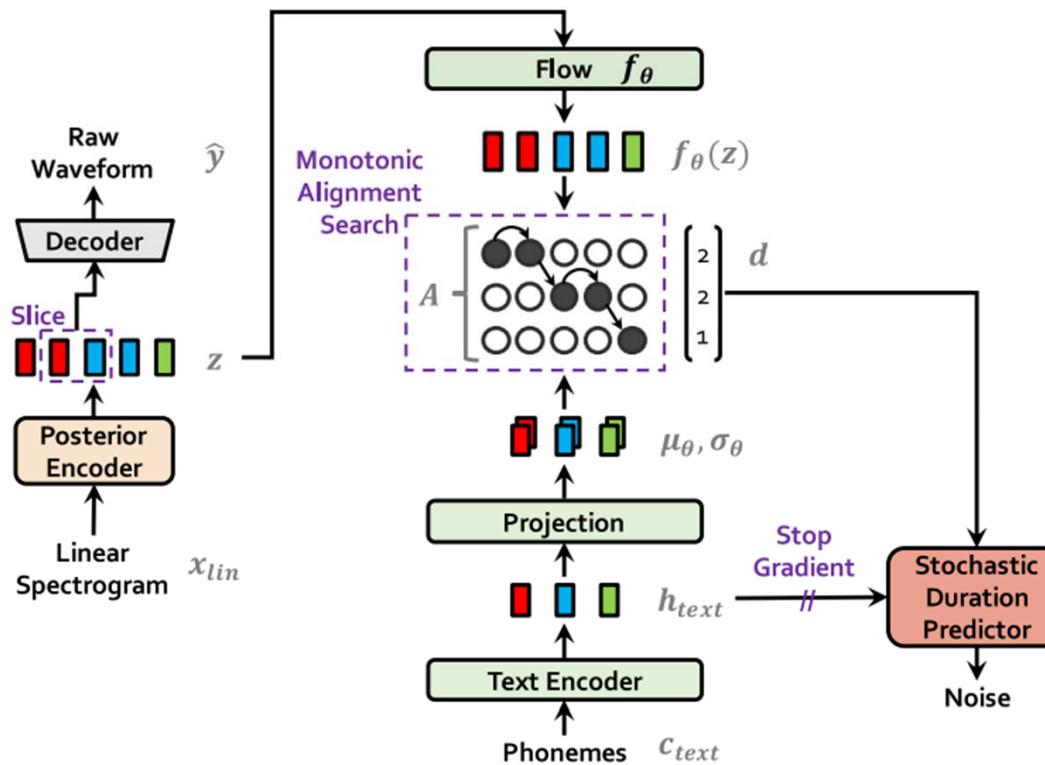
- 3x training speed-up
- More controllable

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

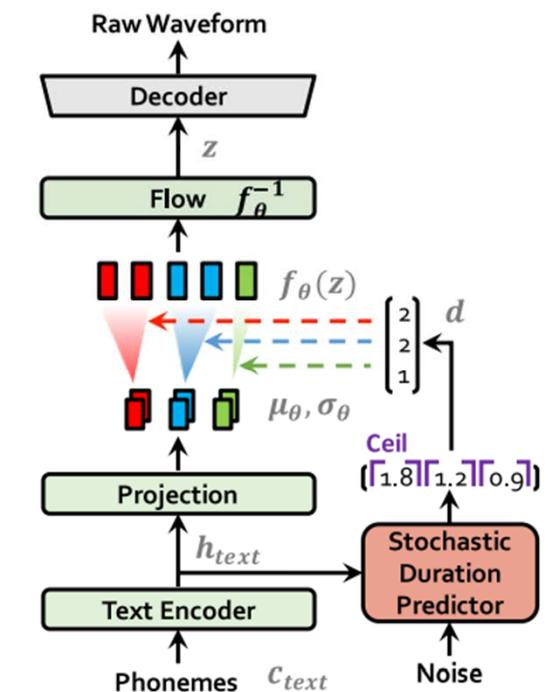
Method	Training Time (h)	Inference Speed (RTF)	Inference Speedup
<i>Transformer TTS (Li et al., 2019)</i>	38.64	9.32×10^{-1}	/
<i>FastSpeech (Ren et al., 2019)</i>	53.12	1.92×10^{-2}	$48.5 \times$
<i>FastSpeech 2</i>	17.02	1.95×10^{-2}	$47.8 \times$
<i>FastSpeech 2s</i>	92.18	1.80×10^{-2}	51.8 ×

VITS

<https://jaywalnut310.github.io/vits-demo/>



(a) Training procedure

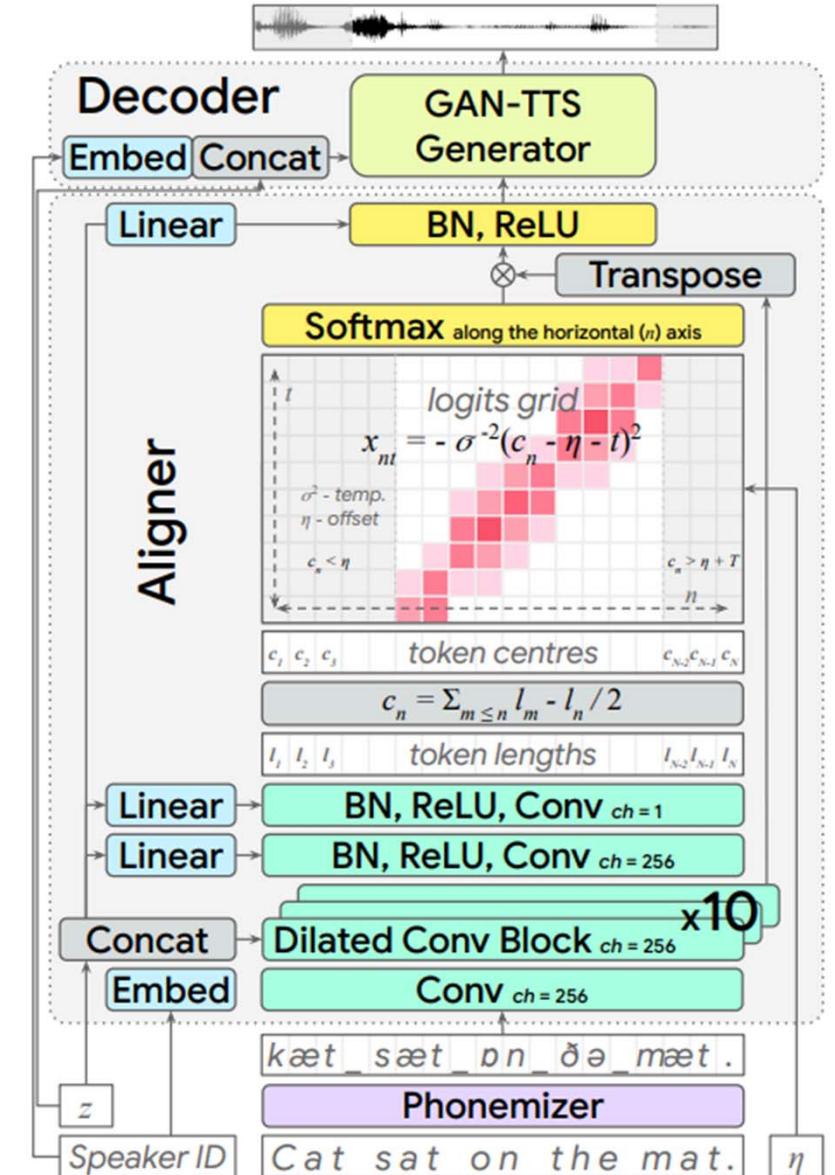


(b) Inference procedure

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



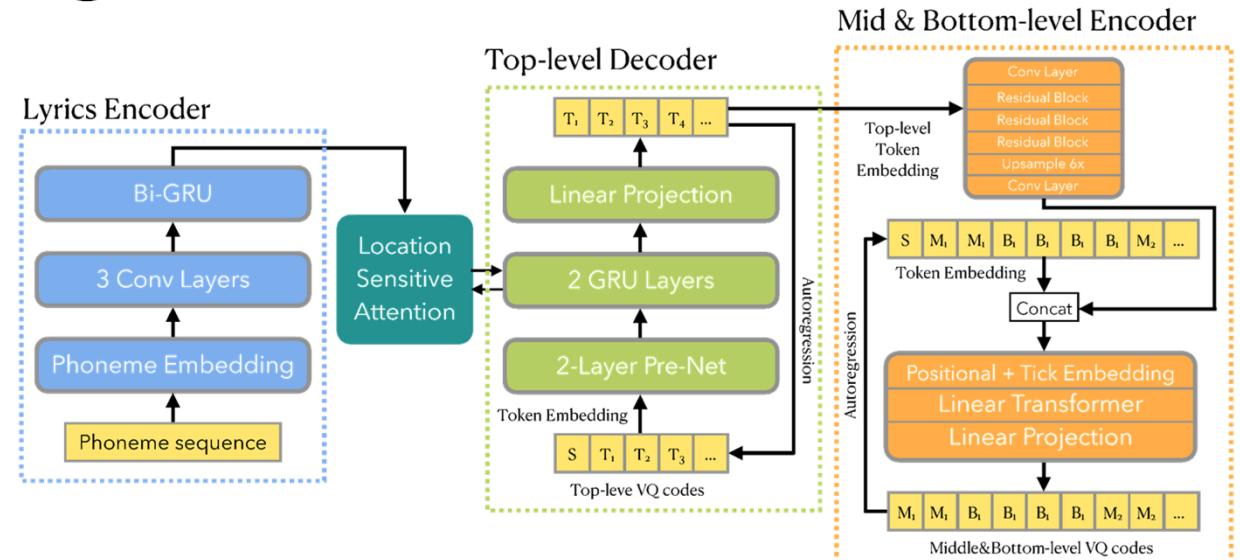
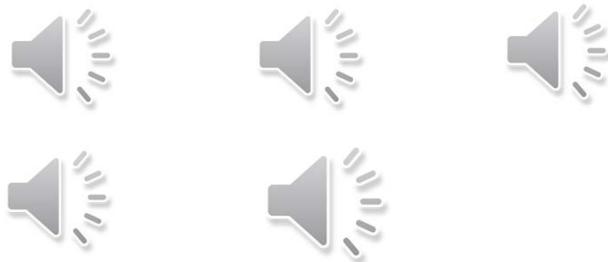
Ref: Donahue et al, “End-to-end adversarial text-to-speech,” ICLR 2021

Drawbacks of Non-autoregressive models

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

KaraSinger (from our lab)

- 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

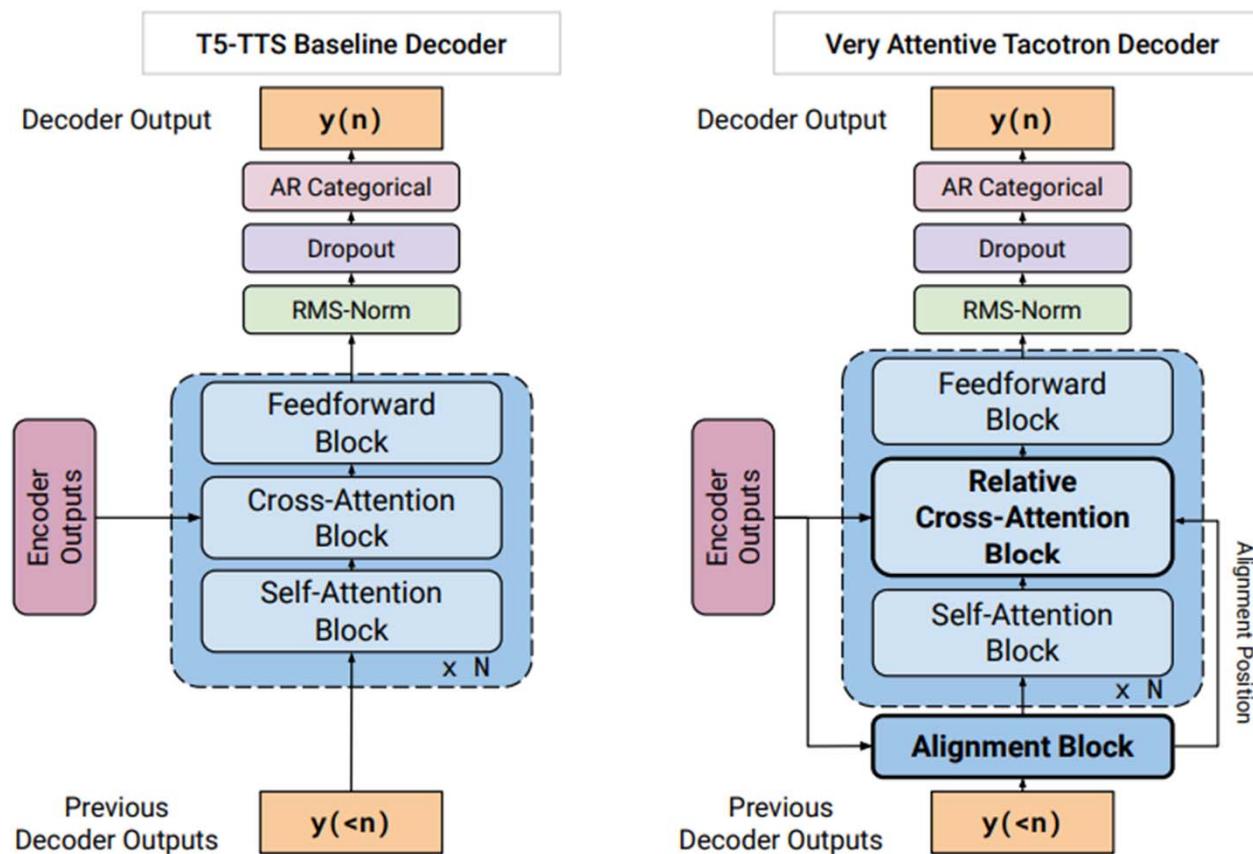
Ref 2: Liu et al, "Score and lyrics-free singing voice generation," ICCC 2020

Very Attentive Tacotron

Robust and unbounded length generalization in autoregressive Transformer-based TTS

https://google.github.io/tacotron/publications/very_attentive_tacotron/index.html

Ref: Battenberg et al,
“Very Attentive Tacotron:
Robust and unbounded
length generalization in
autoregressive
Transformer-based text-
to-speech,” arXiv 2024



Recap: TTS

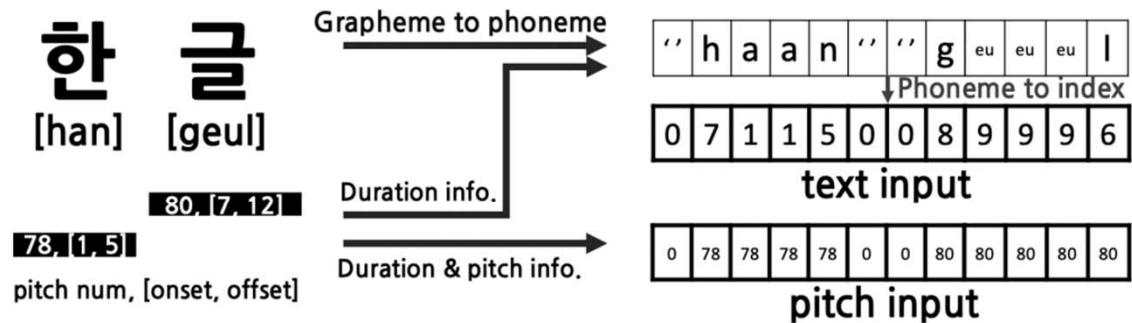
- **TacoTron 2** (ICASSP'18)
 - Use a recurrent seq2seq architecture
 - Autoregressive
 - Let attention learns the alignment between the input phoneme sequence and the output Mel-spectrogram sequence
- **FastSpeech 2** (ICLR'21)
 - Use a feed-forward Transformer architecture
 - Non-autoregressive
 - Use duration predictor (among many other variance adaptors)

Outline

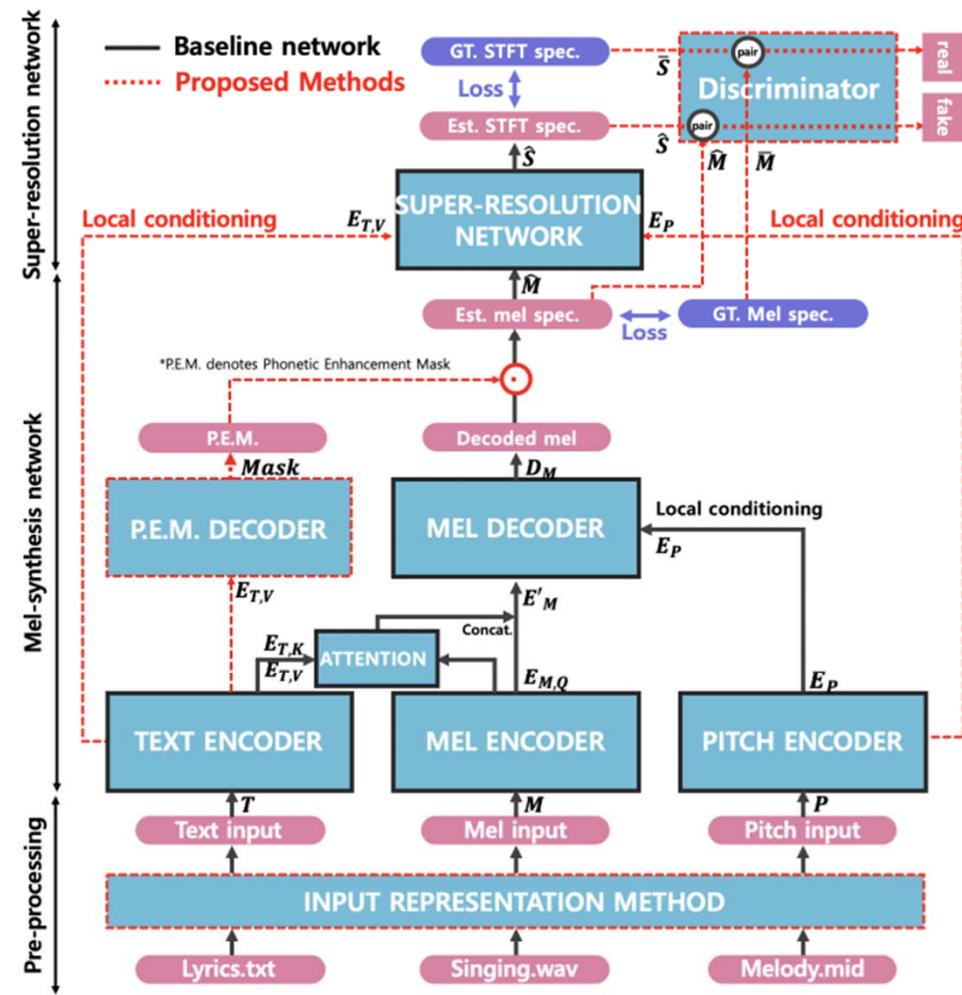
- Singing voice processing: Historical review
- Neural text-to-speech synthesis
- **Neural singing voice synthesis**
- Build your SVS model

SNU'2019 Model

- Autoregressive
 - Use GAN
 - **Pitch embedding**

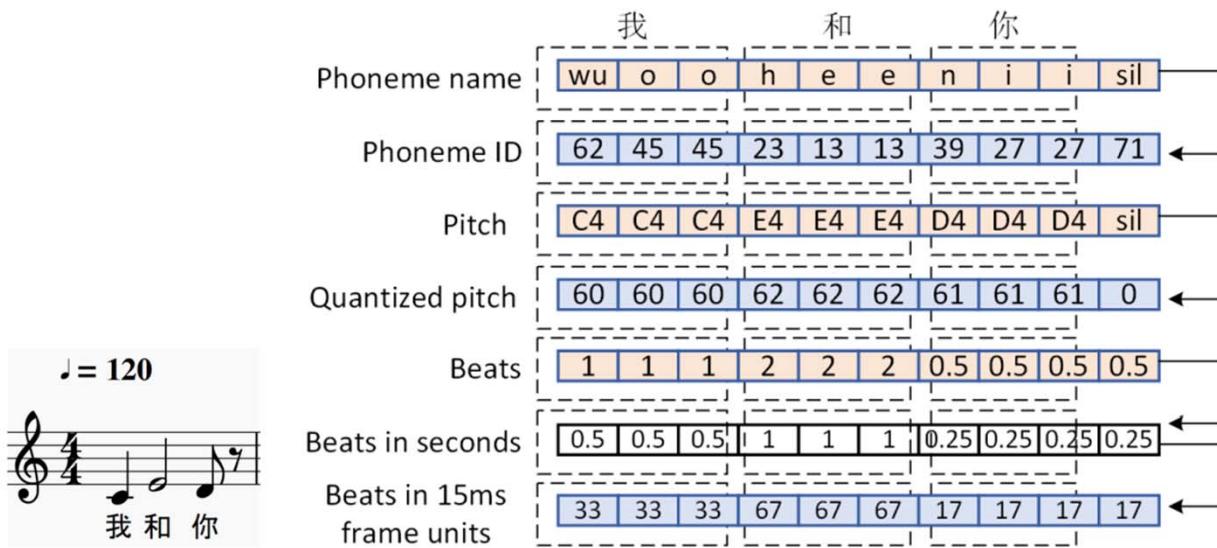


- **DB:** 60 songs from 1 female singer (~2hrs), manual annotation

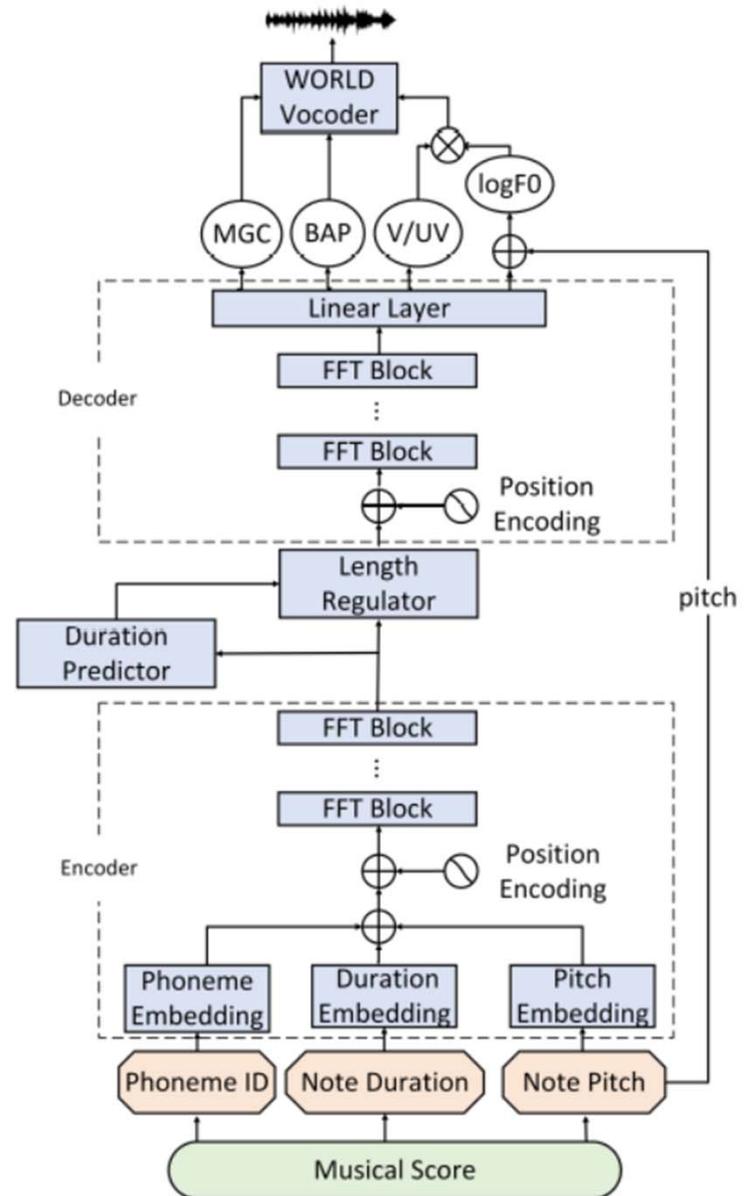


XiaoiceSing

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



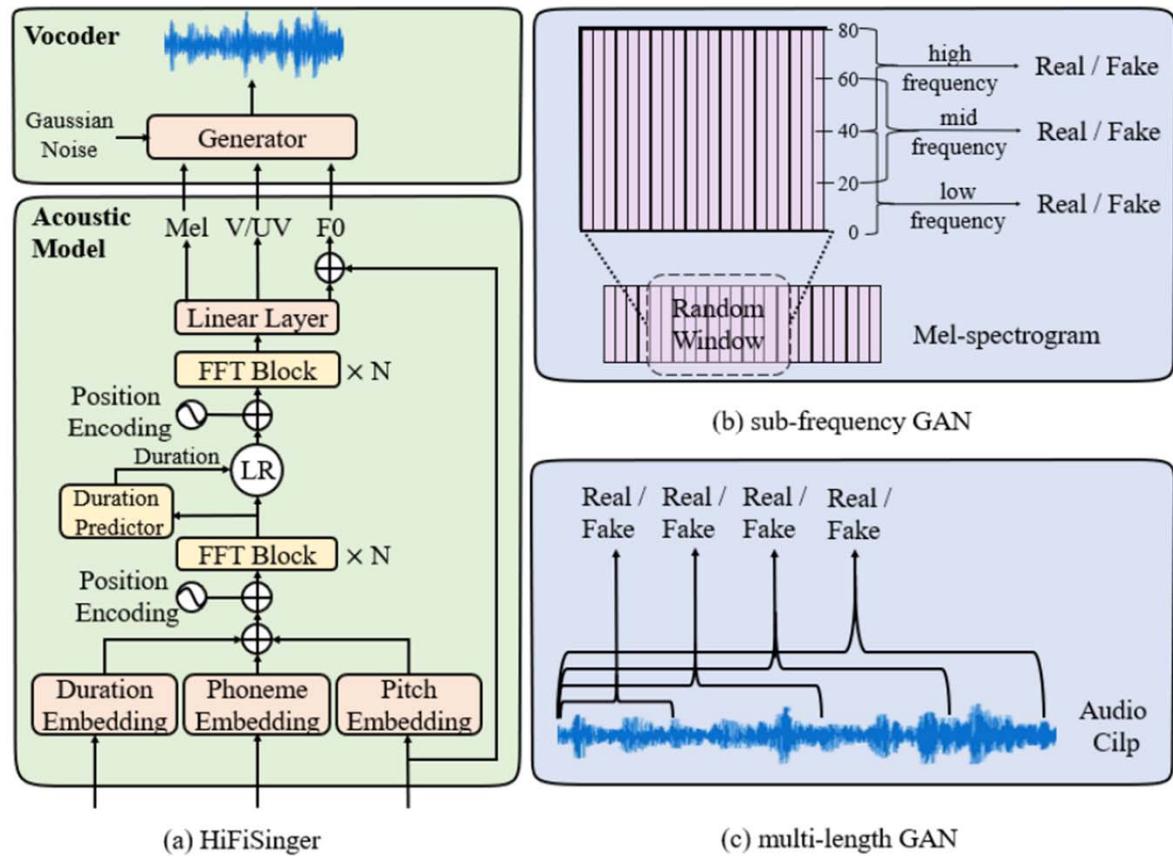
Ref: Lu et al, "XiaoiceSing: A high-quality and integrated singing voice synthesis system," INTERSPEECH 2020



HiFiSinger

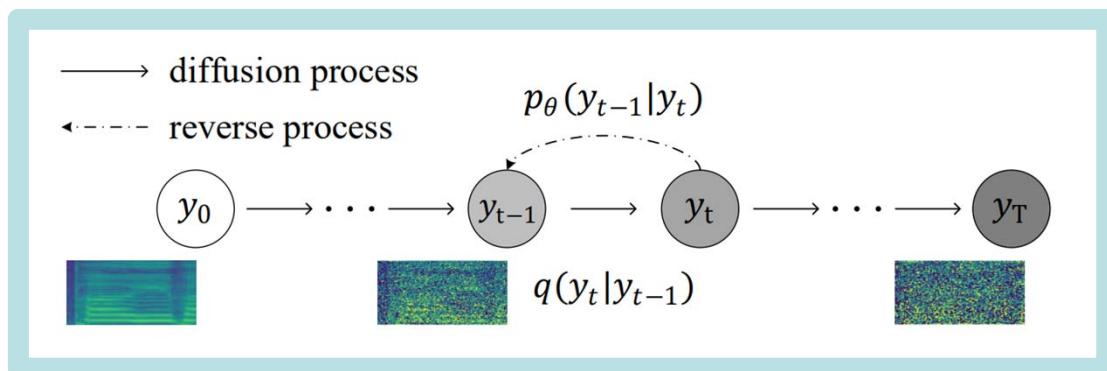
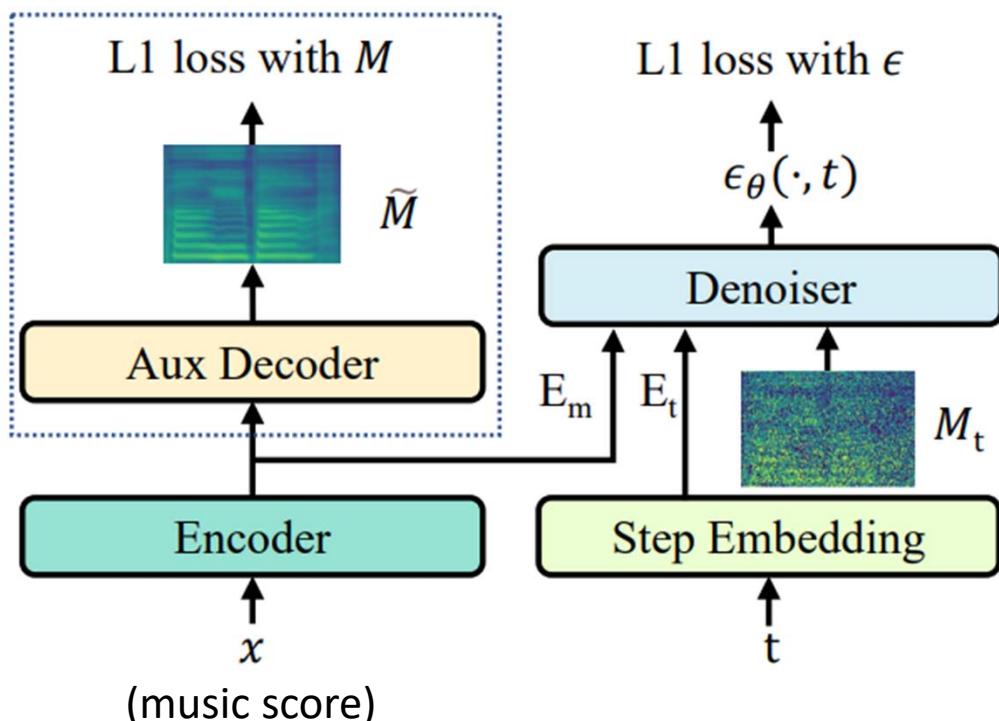
<https://speechresearch.github.io/hifisinger/>

- 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



DiffSinger

https://jisang93.github.io/hidden_singer-demo/



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

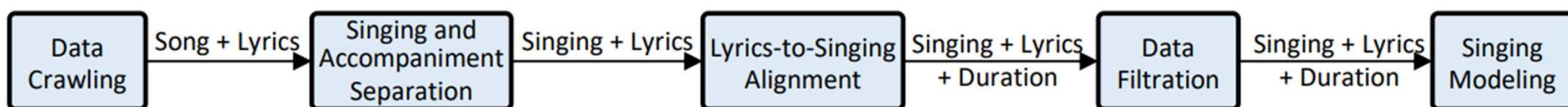
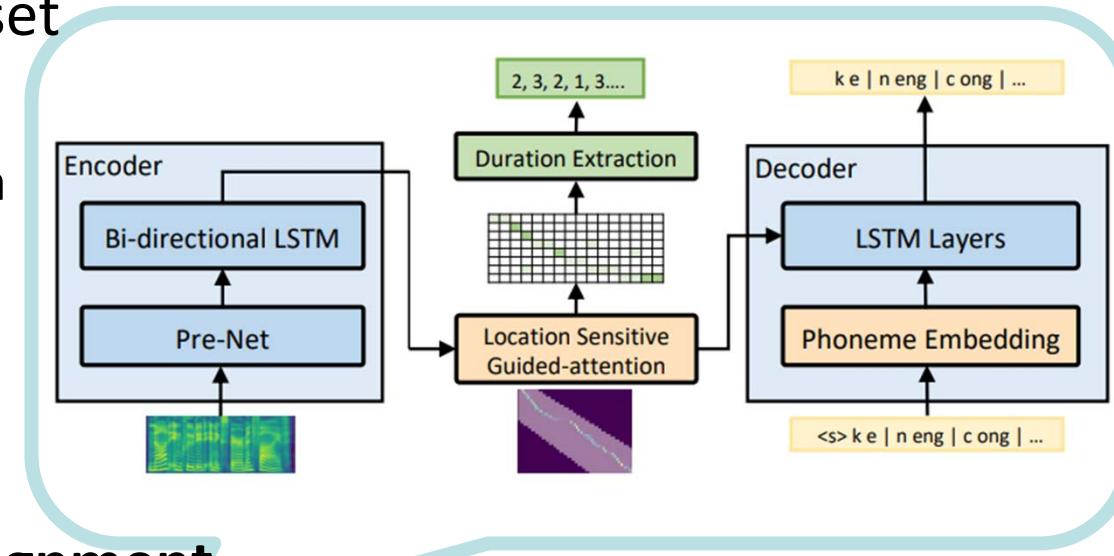
Progress of SVS Benefits from that of TTS

- 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 predict the **onset** and **offset** of each word or syllable
- Differences
 - Musical expressivity (**f0**, dynamics, singing techniques)
 - Cost to collect data

DeepSinger

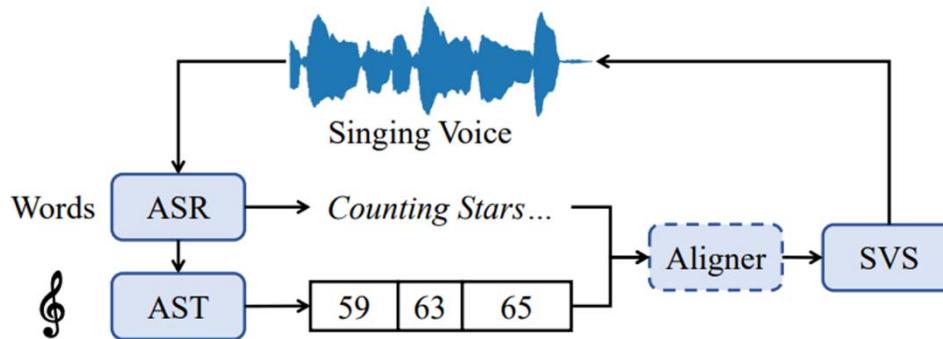
<https://speechresearch.github.io/deepsinger/>

- Huge effort in building an SVS dataset
 - 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
- Data from the web + automatic alignment



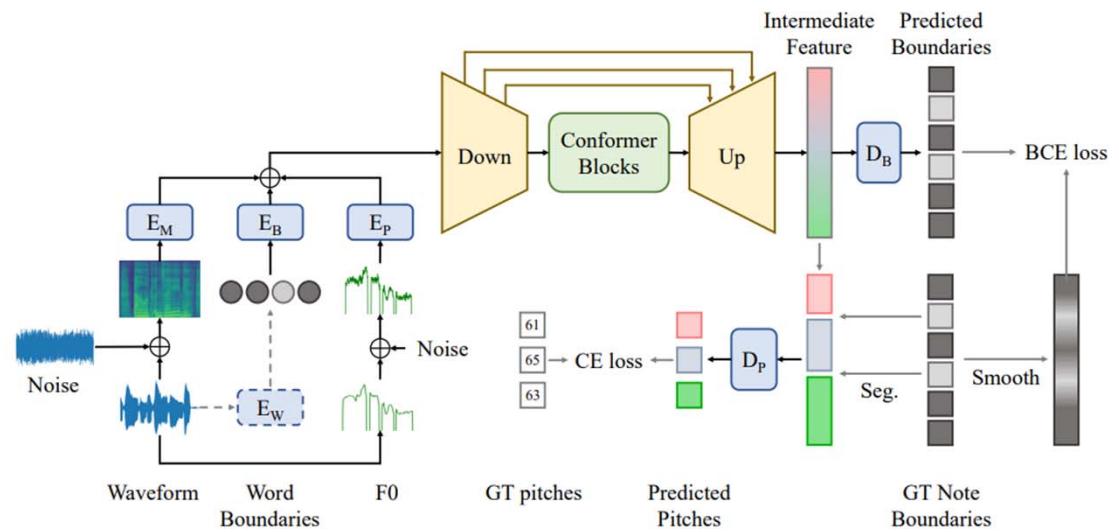
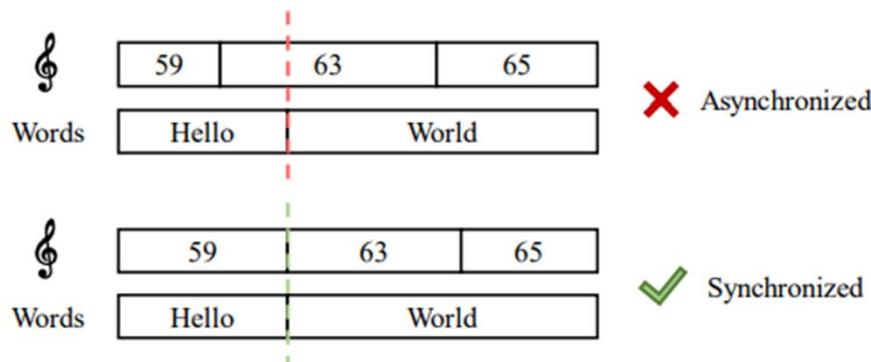
Ref: Ren et al, "DeepSinger: Singing voice synthesis with data mined from the web," KDD 2020

Automatic Alignment



AST: automatic singing voice transcription

ASR: automatic speech recognition



Model	RPA	MOS-P	MOS-Q
GT	96.1	4.02 ± 0.05	4.04 ± 0.06
$S(\text{large})$	45.2	3.36 ± 0.12	3.45 ± 0.09

-P: pitch correction; -Q: overall quality

Singing Voice Conversion

- Many-to-one (only a target singer) vs many-to-many
- In-domain vs cross-domain

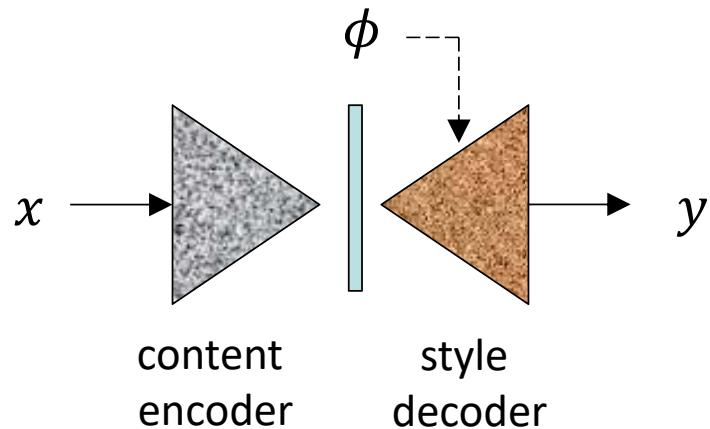


Table 3: Details of participating systems in SVCC 2023.

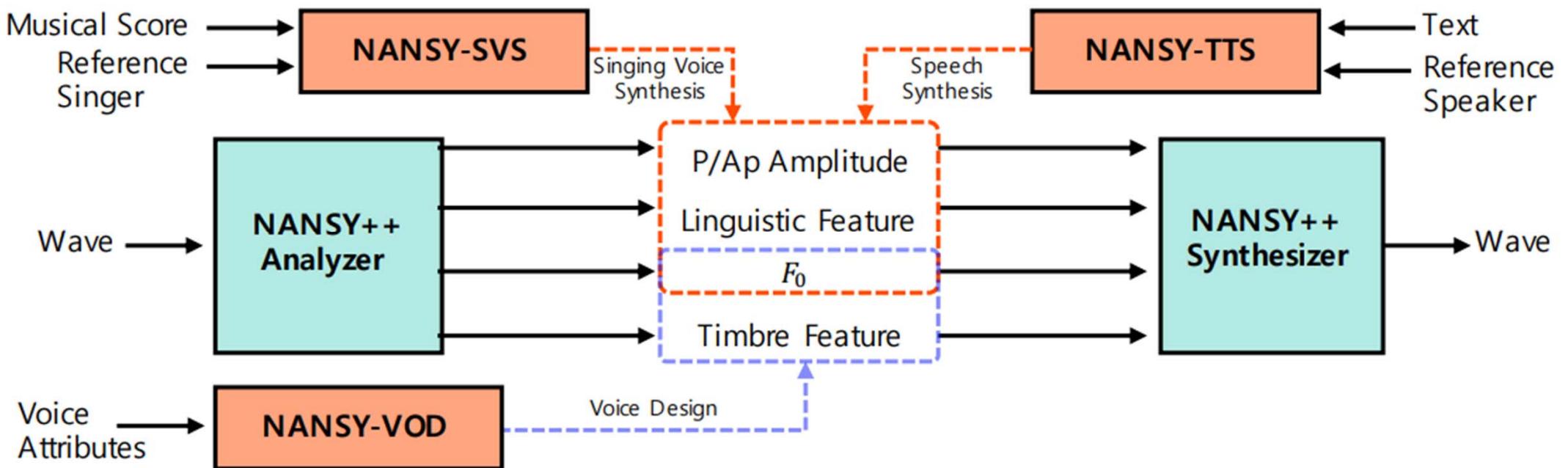
ID	Content Feature	VAE	Vocoder
B01	PPG	N	HiFi-GAN
B02	HuBERT	N	HN-uSFGAN
T01	PPG	N	HiFi-GAN + BigVGAN*
T02	HuBERT	Y	DSPGAN
T03	HuBERT	Y	HiFi-GAN
T05	PPG	N	HiFi-GAN
T06	ContentVec	Y	N/A (nsf-HiFi-GAN)‡
T07	HuBERT	Y	N/A (HiFi-GAN)‡
T09	Uncertain	Y	nsf-HiFi-GAN
T10	WavLM	N	BigVGAN
T11	PPG	N	HiFi-GAN
T12	HuBERT	N	HiFi-GAN
T13	ContentVec	N	SiFi-GAN
T15	None (Melspec)†	N	nsf-HiFi-GAN
T16	PPG	N	BigVGAN
T17	HuBERT	N	nsf-HiFi-GAN
T19	None (Melspec)†	N	HiFi-GAN
T20	HuBERT	Y	nsf-HiFi-GAN
T21	ContentVec	Y	nsf-HiFi-GAN
T22	PPG+ContentVec	N	BigVGAN
T23	PPG	Y	DSPGAN

(PPG: phonetic posteriograms)

Nansy++

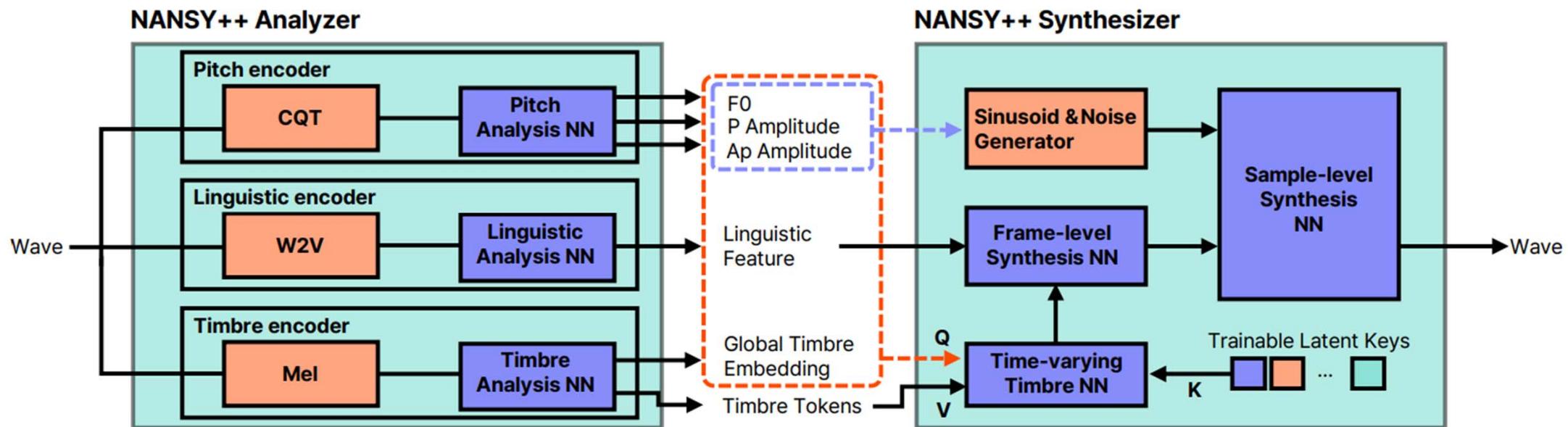
<https://bald-lifeboat-9af.notion.site/Demo-Page-For-NANSY-67d92406f62b4630906282117c7f0c39>

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



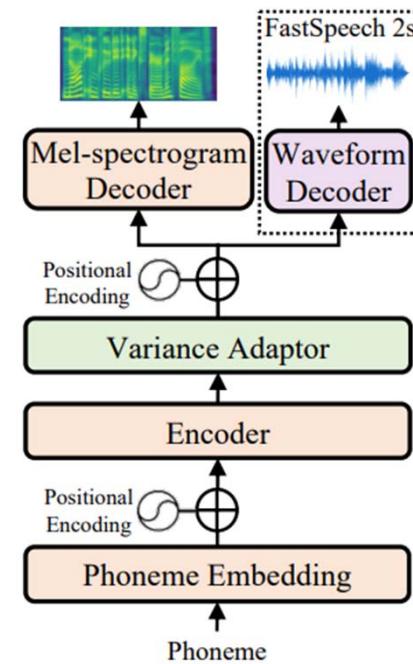
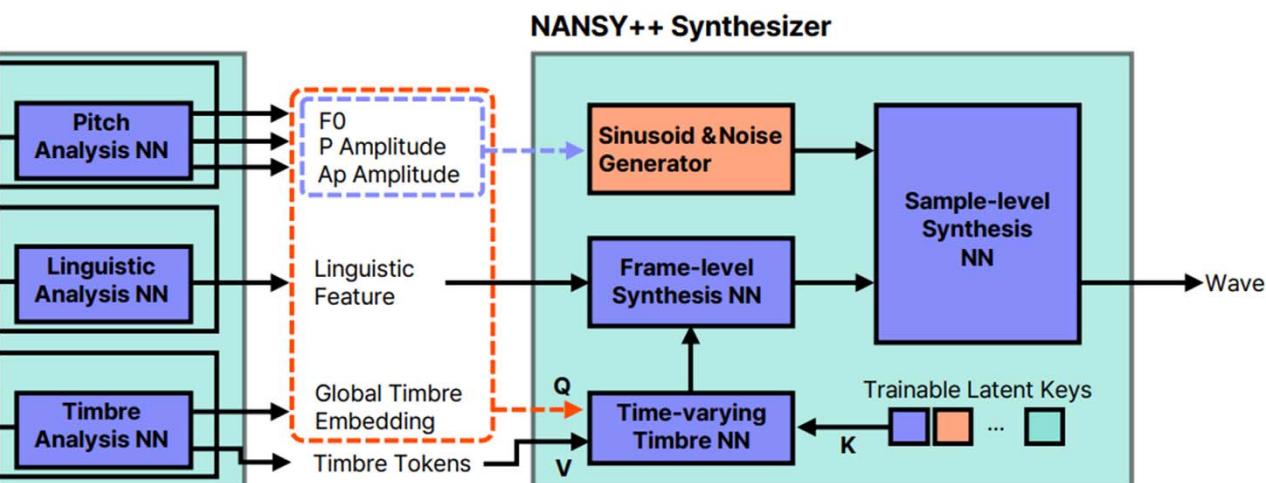
Nansy++

- “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”

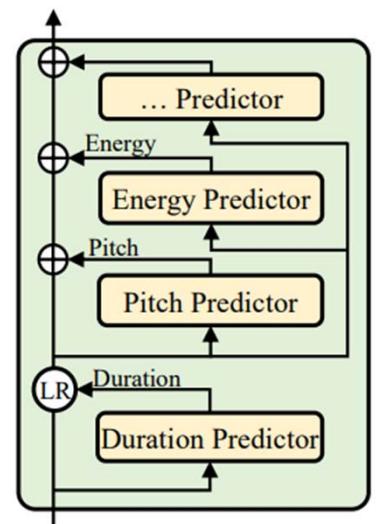


Nansy++ vs FastSpeech 2

- 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++

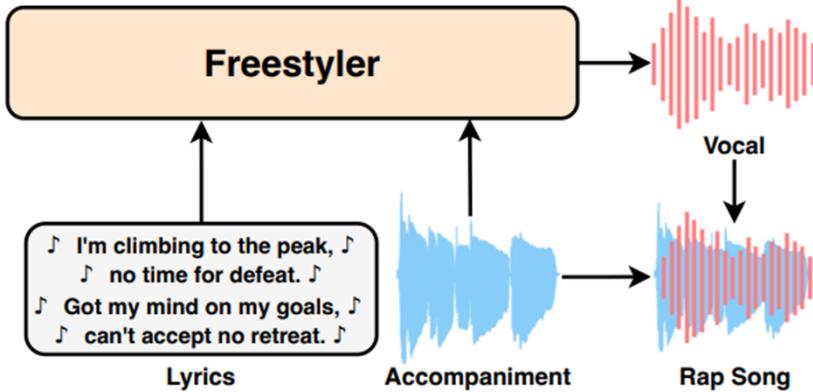
- One exception:
DiffSinger

– [https://github.com/Moon
InTheRiver/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

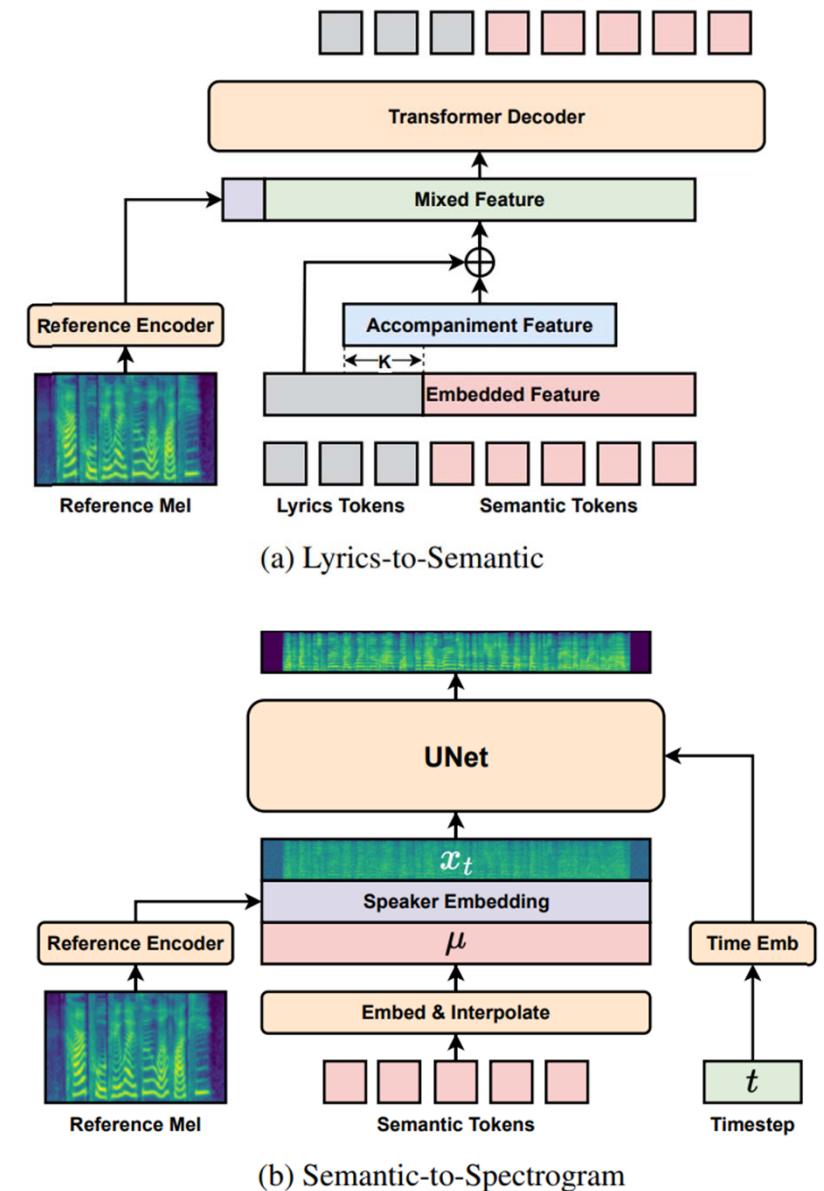
Freestyler

<https://nzqian.github.io/Freestyler/>



First off, I want to thank the pioneers for making this possible. Fake, humbleness, snake, dummy shit is intolerable. Living by principle. Staring out over women in swimming suits by the pool. I'm a slide with a few. Every chance was spoiled. No enhancement pill. No dear antler oil. And they panties be soiled. I know some niggas that'll damage your squad. You

Ref: Ning et al, "Drop the beat! Freestyler for accompaniment conditioned rapping voice generation, arXiv 2024



Outline

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

Datasets

<https://gtsinger.github.io/>

Corpus	Language	Singer	Hours		Manual Annotations				Controlled Comparison
			Singing	Speech	Align	RMS	Tech	Style	
VocalSet [25]	1	20	10.1	0	✗	✗	✗	✗	✓
CSD [4]	2	1	4.86	0	✗	✗	✗	✗	✗
KVT [10]	1	114	18.85	0	✗	✗	✗	✓	✗
PopBuTFy [16]	2	34	50.8	0	✗	✗	✗	✗	✗
OpenSinger [8]	1	66	50	0	✗	✗	✗	✗	✗
NHSS [20]	1	10	4.75	2.25	✗	✗	✗	✗	✗
Tohoku Kiritan [18]	1	1	1	0	✓	✗	✗	✗	✗
OpenCpop [23]	1	1	5.25	0	✓	✗	✗	✗	✗
M4Singer [26]	1	20	29.77	0	✓	✗	✗	✗	✗
GTSinger (Ours)	9	20	80.59	16.16	✓	✓	✓	✓	✓

Ref: Zhang et al, “GTSinger: A global multi-technique singing corpus with realistic music scores for all singing tasks, 2024

Datasets: SingStyle111

<https://dsqvival.github.io/singstyle111/>

Dataset	Language	Style	#Hour	#Singer	Quality	Musicality	Score	Alignment	Style Transfer
Opencpop [41]	Chinese	Pop	5.25	1	Studio	Ama.	Perform. MIDI	✓	✗
M4Singer [42]	Chinese	Pop	29.77	20	Studio	50% Ama. 50% Prof.	Perform. MIDI	✓	✗
Children Song [43]	Korean English	Children	4.86	1	Studio	Prof. but plain	Perform. MIDI	word	✗
Tohoku Kiritan [44]	Japanese	Pop	0.95	1	Studio	Prof.	Score	✓	✗
PopCS [28]	Chinese	Pop	5.89	6	Not Clean	Ama.	✗	✗	✗
Open-Singer [35]	Chinese	Pop	50	66	Studio	Ama.	✗	✗	✗
VocalSet [45] Annotated [46]	Five Vowels	Opera	10.1	20	Studio	Prof.	Score	✓	technique transfer
NHSS [47]	English	Pop	3.5	10	Studio	Ama.	✗	✓	✗
NUS-48E [48]	English	Pop Children	1.41	12	Studio	Ama.	✗	✓	✗
RWC [49]	Japanese English	Pop	4	27	Not Solo	Prof.	Both	✗	✗
TONAS [50]	Spanish	Flamenco	0.34	> 40	Not Clean	Prof.	✗	✗	✗
Vocadito [51]	Seven Languages	Pop Children	0.23	29	Not Clean	Ama.	✗	✗	✗
MIR-1K [52]	Chinese	Pop	2.22	19	Not Solo	Ama.	✗	✗	✗
StyleSing111 (Ours)	English Chinese Italian	Opera Pop Folk Jazz etc.	12.8	8	Studio	Prof.	Both	✓	✓

- Children

▶ 0:03 / 0:15
⋮
- Chinese Folk

▶ 0:02 / 0:16
⋮
- Jazz

▶ 0:03 / 0:12
⋮
- Musical

▶ 0:02 / 0:13
⋮
- Opera

▶ 0:08 / 0:14
⋮
- Pop

▶ 0:00 / 0:00
⋮
- Rock

▶ 0:00 / 0:00
⋮

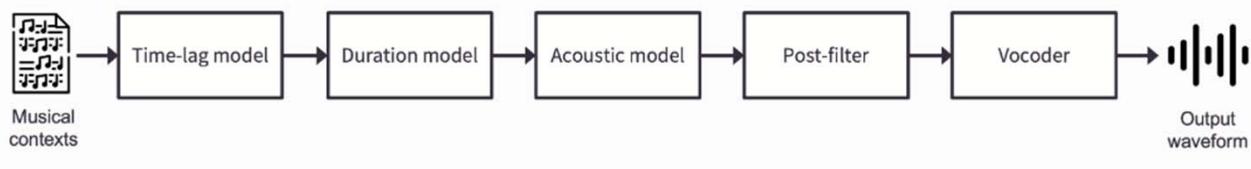
Ref: Dai et al, “SingStyle111: A multilingual singing dataset with style transfer, ISMIR 2023

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

About

Neural network-based singing voice synthesis library for research



r9y9 Ryuichi Yamamoto

Library for SVS: Muskits

<https://github.com/SJTMusicTeam/Muskits>

- Various network architectures 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)

About

An open source music processing toolkit

- Multi-speaker & Multilingual extension

- Speaker ID embedding
- Language ID embedding
- Global style token (GST) embedding

- Various language support

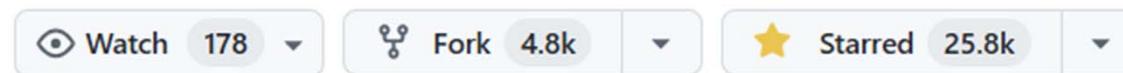
- Jp / En / Kr / Zh

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

Library for SVC: so-vits-svc

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



About

SoftVC VITS Singing Voice Conversion

flow ai deep-learning voice
speech pytorch audio-analysis
generative-adversarial-network
variational-inference voice-conversion vc
voice-changer vits
singing-voice-conversion voiceconversion
sovits so-vits-svc