2025 edition

Deep Learning for Music Analysis and Generation

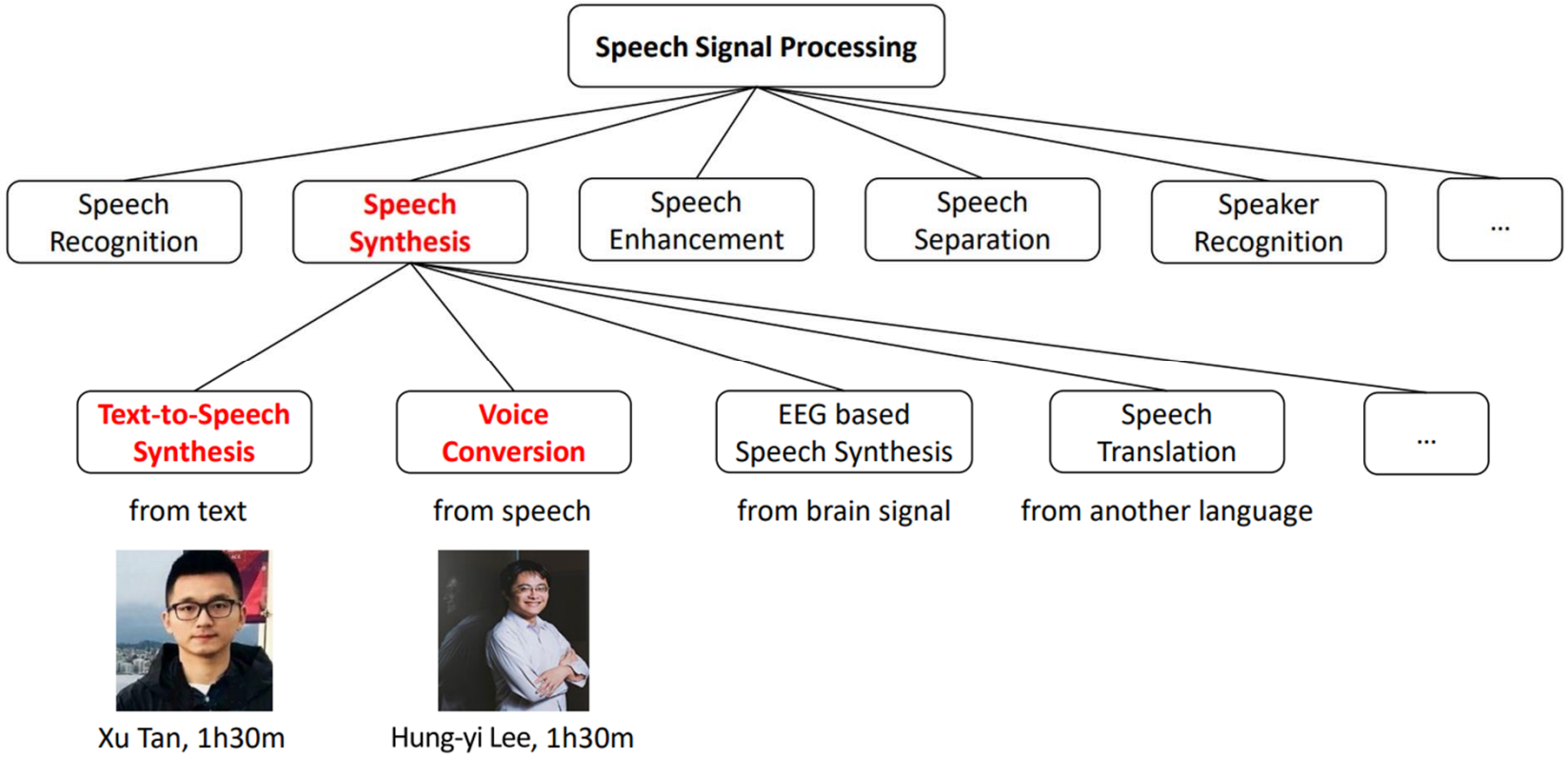
Singing Voice Generation & Song Generation

(text → 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

Reference 3: ISMIR 2024 Tutorial

https://ismir2024.ismir.net/tutorials

T6: “Lyrics and Singing Voice Processing in Music Information Retrieval: Analysis, Alignment, Transcription and Applications"

Presenters: Daniel Stoller, Emir Demirel, Kento Watanabe, and Brendan O’Connor

Reference 4: ISMIR 2015 Tutorial

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

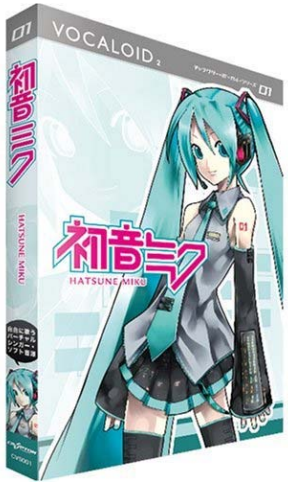
- All popular music cultures around the world use singing
- The singing voice is the most expressive of all musical instruments
- Singing voice produces both sound and words
- Various commercial applications

Outline

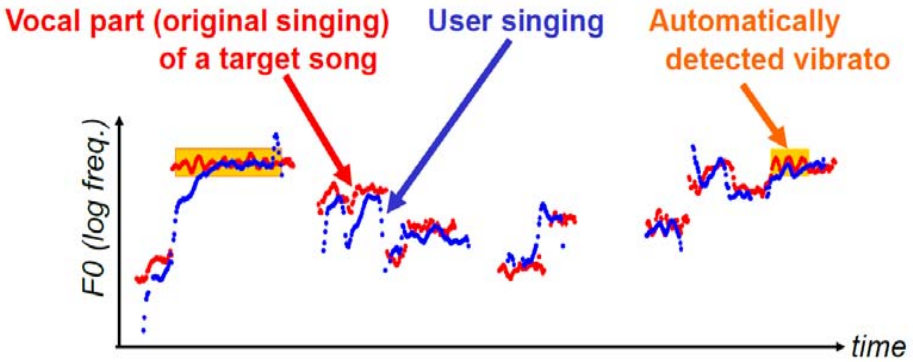
- **Singing voice processing: Historical review**
- Neural text-to-speech synthesis
- Neural singing voice synthesis
- Build your SVS model
- Song generation

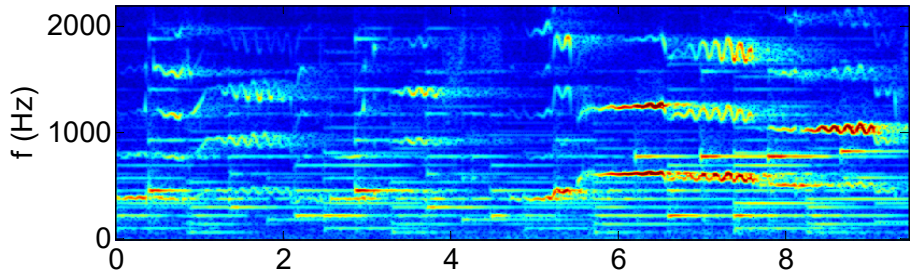
Commercial Applications

- 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/

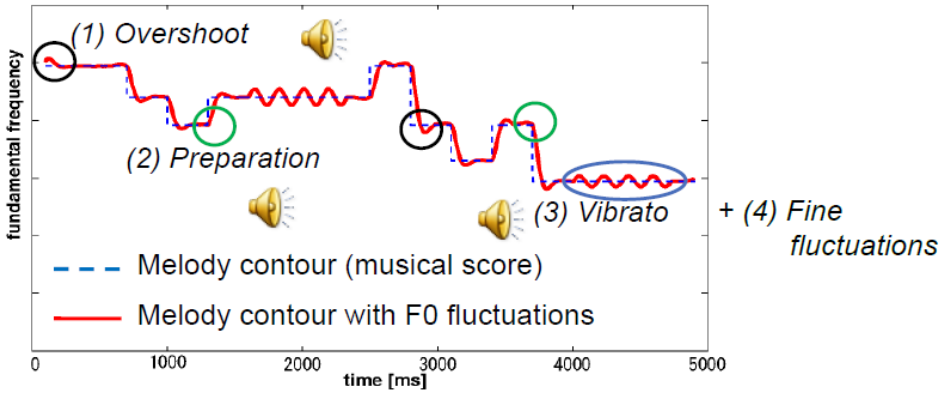
Ref: Kajita et al, “Songrium: Browsing and listening environment for music content creation community,” SMC 2015

Singing Voice Generation

- **Singing voice synthesis** (SVS)

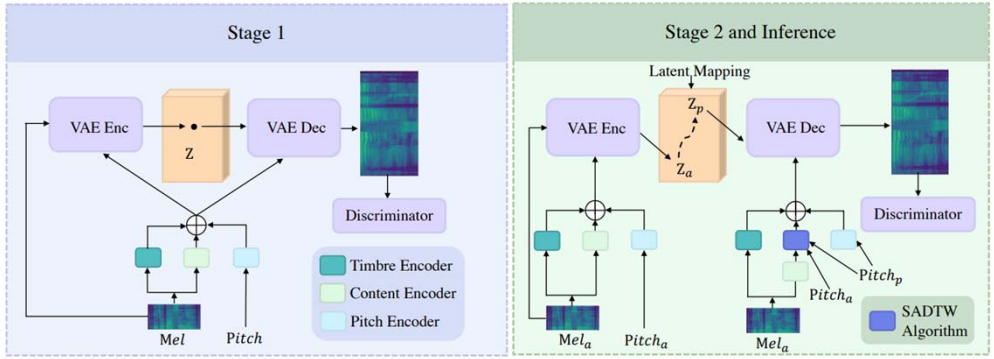
- **Singing voice conversion** (SVC)

- **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

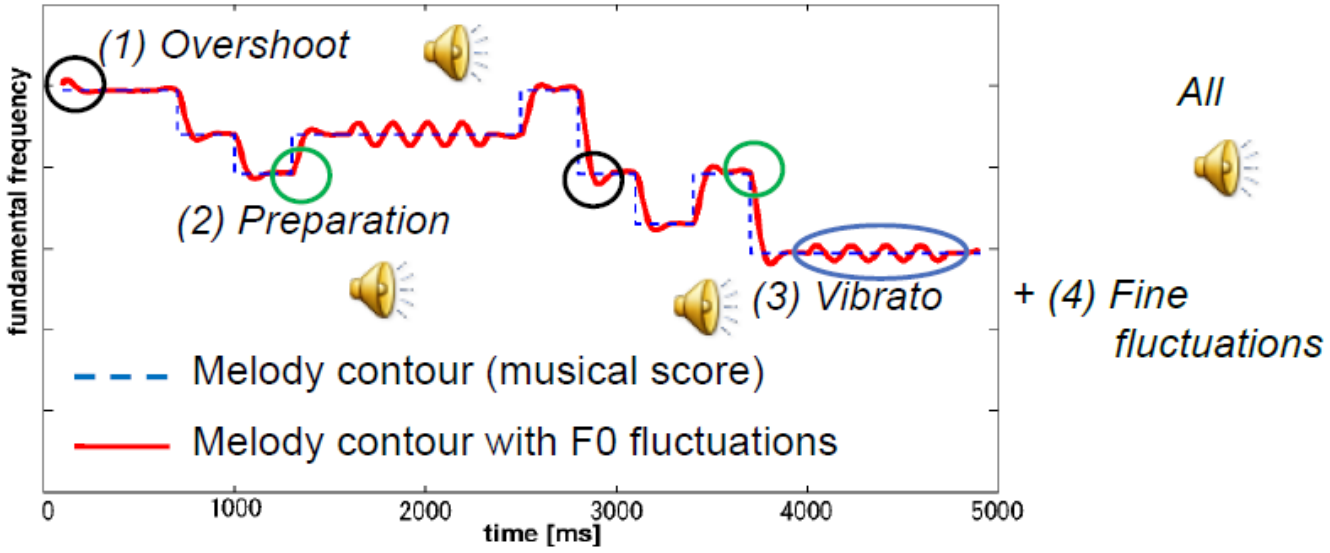
- **Singing voice beautification**



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

Singing Generation Tasks: 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 Generation Tasks: Singing Voice Beautifier

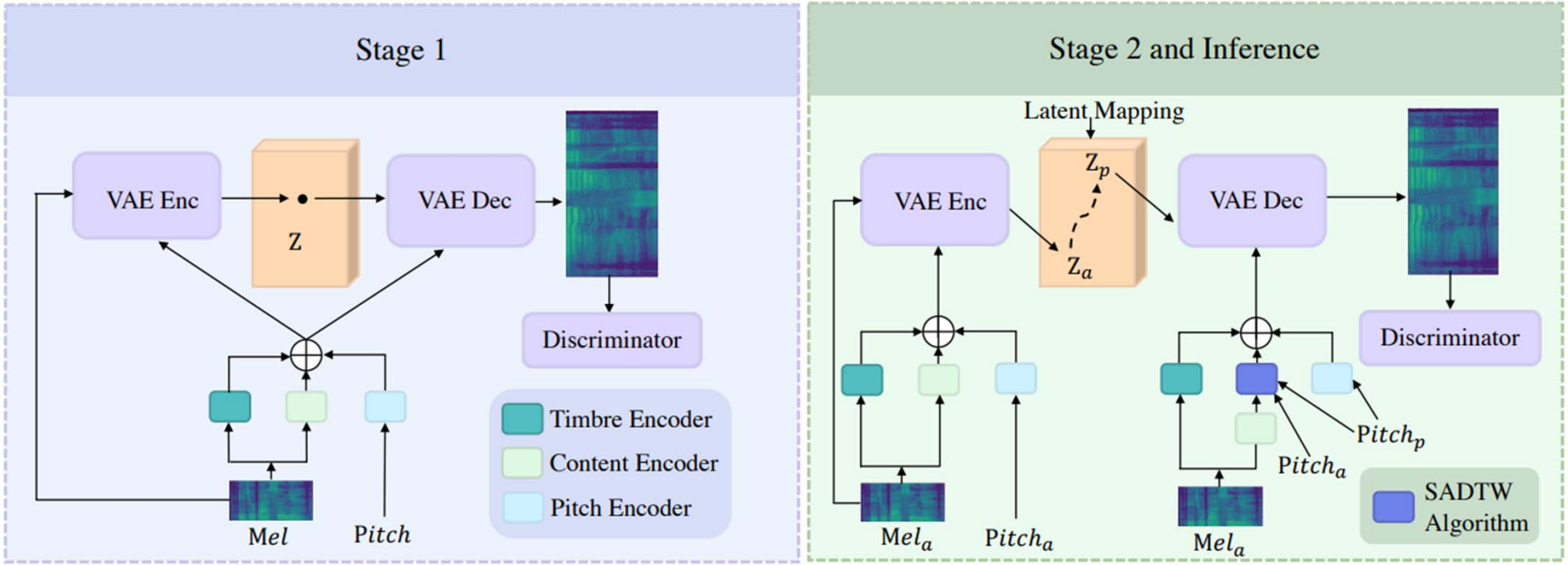


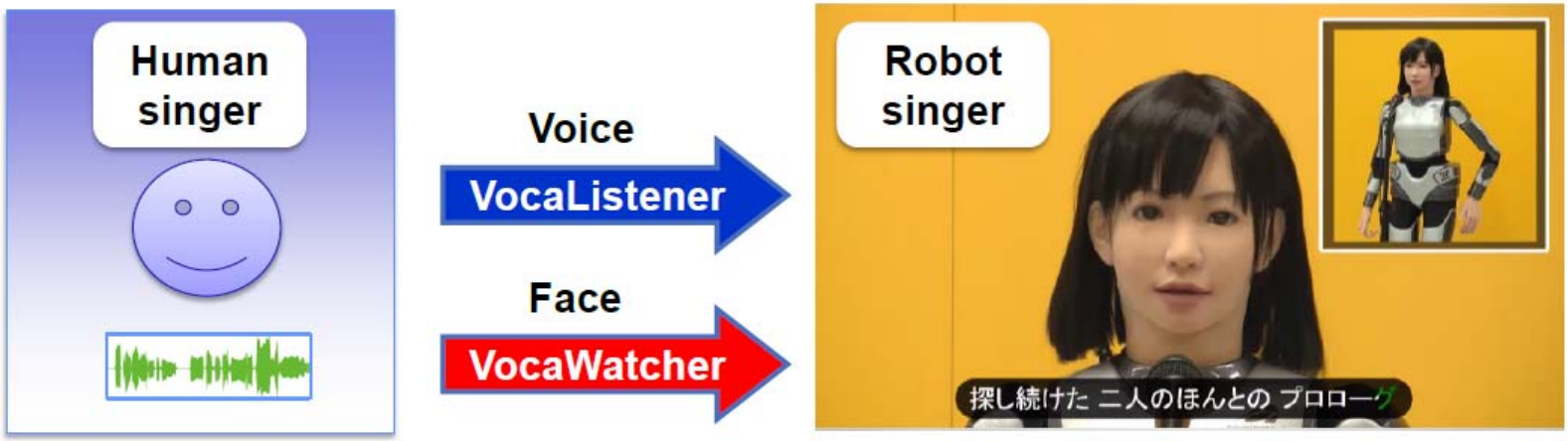
Figure 1: The overview of NVSB. The training process consists of 2 stages, and the second stage shares the same pipeline with the inference stage. "VAE Enc" means the encoder of CVAE; "VAE Dec" means the decoder of CVAE; "Mel" means the mel-spectrogram; "**z**" means the latent variable of the vocal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

Singing Generation Tasks: Humanoid Robot Singer

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

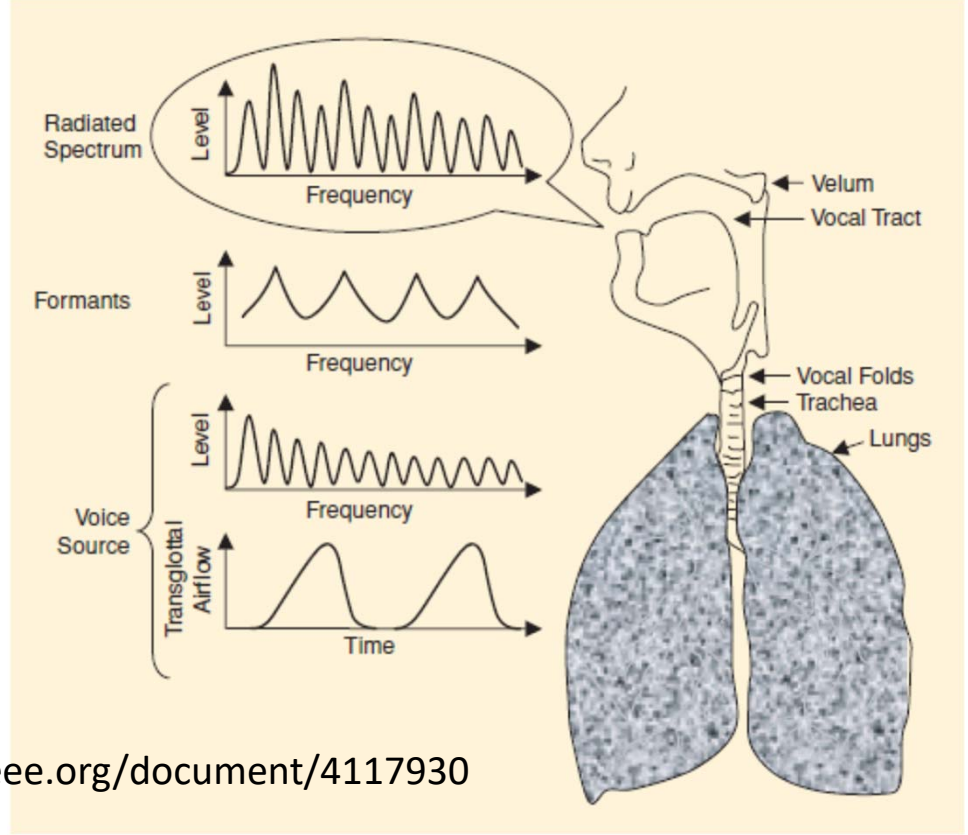
- Imitating a human singer
  - imitate vocal expressions to synthesize singing voices
  - imitate facial expressions to generate robot motions



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

Singing Generation Tasks: Singing Voice Synthesis

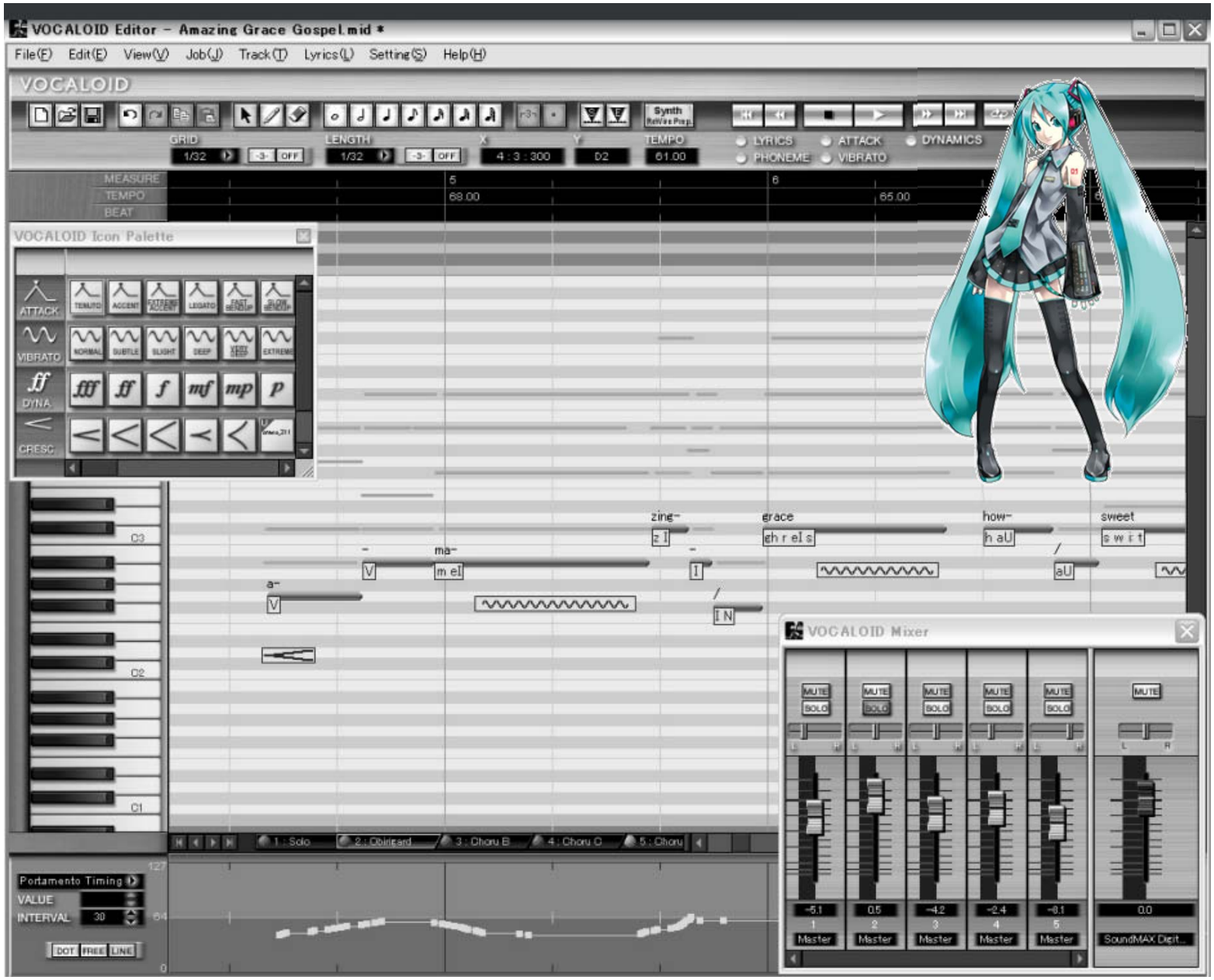
- **text+MIDI → singing audio**
- Cook, “Singing voice synthesis: History, current work, and future directions,” *CMJ 1996*
  https://www.jstor.org/stable/3680822?seq=6



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

- **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)

Hatsune Miku

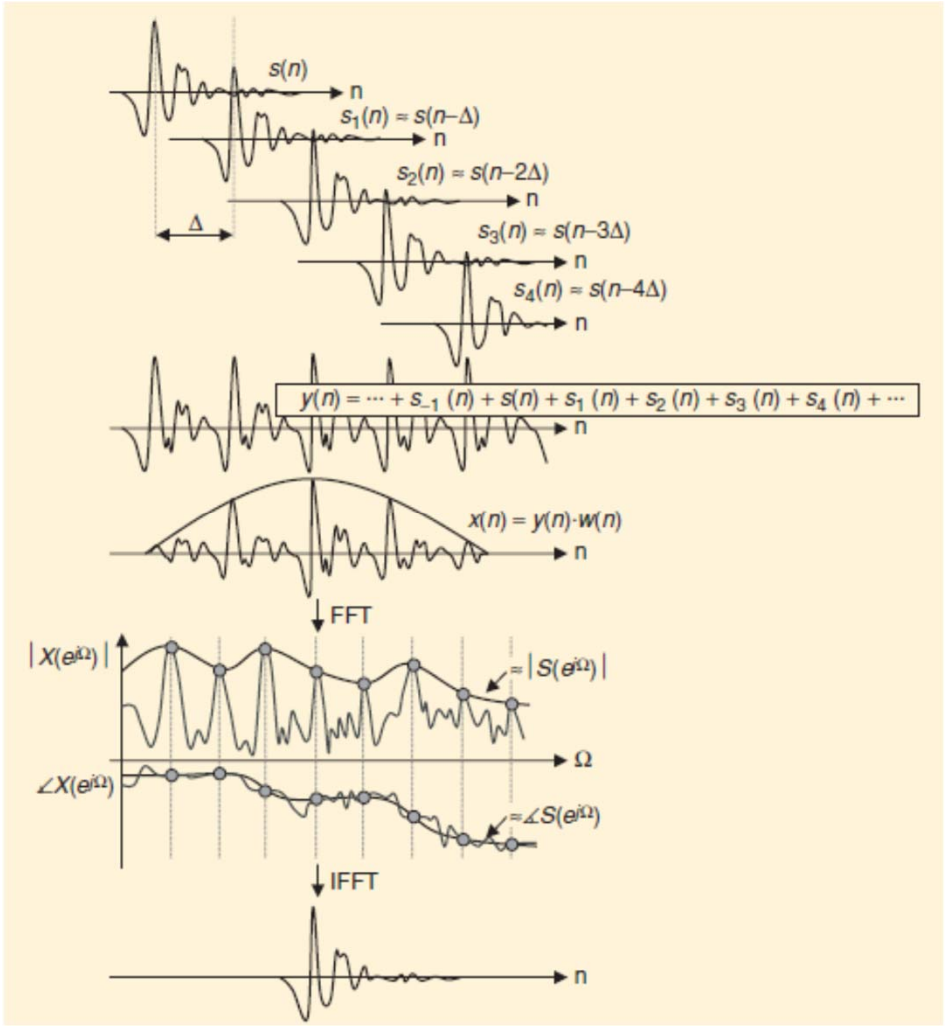
https://www.youtube.com/watch?v=jhl5afLEKdo

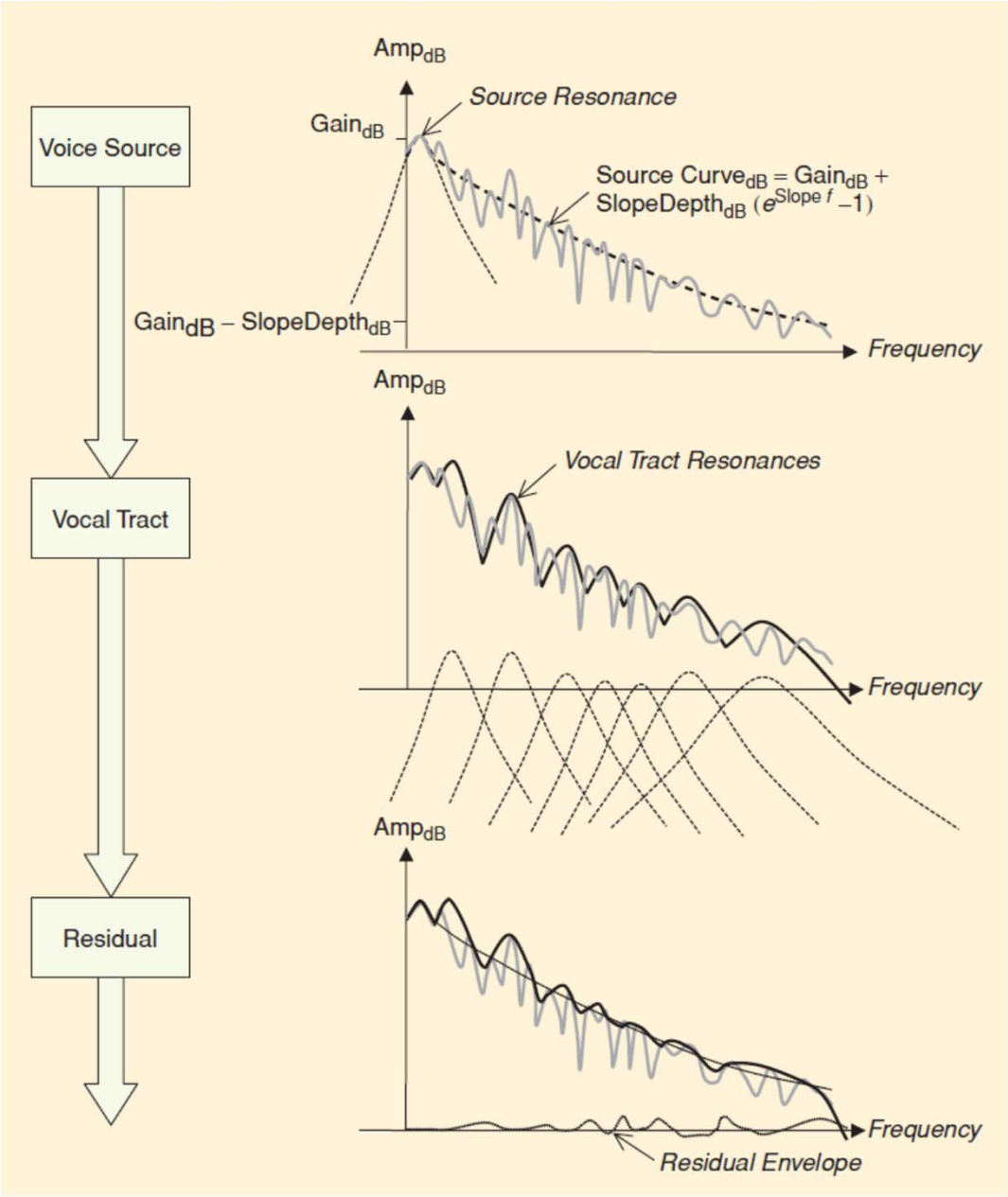
【Hatsune Miku】 World is Mine / ryo（supercell）【初音ミク】

(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

The Uncanny Valley

- **Samples** are “safe”, but the music may sound flat and rigid
- In pursuit of naturalness, we need other approaches that do not use samples

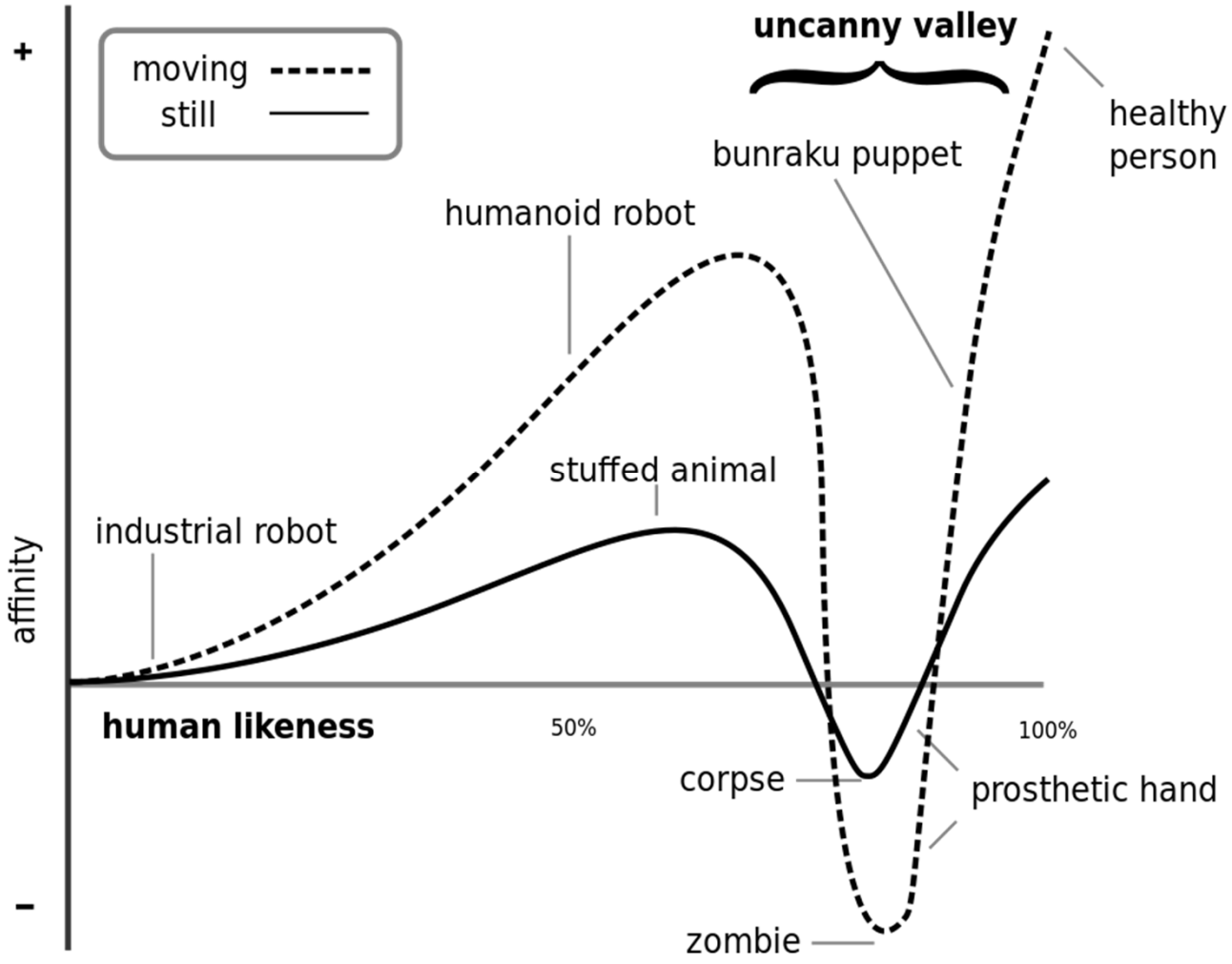
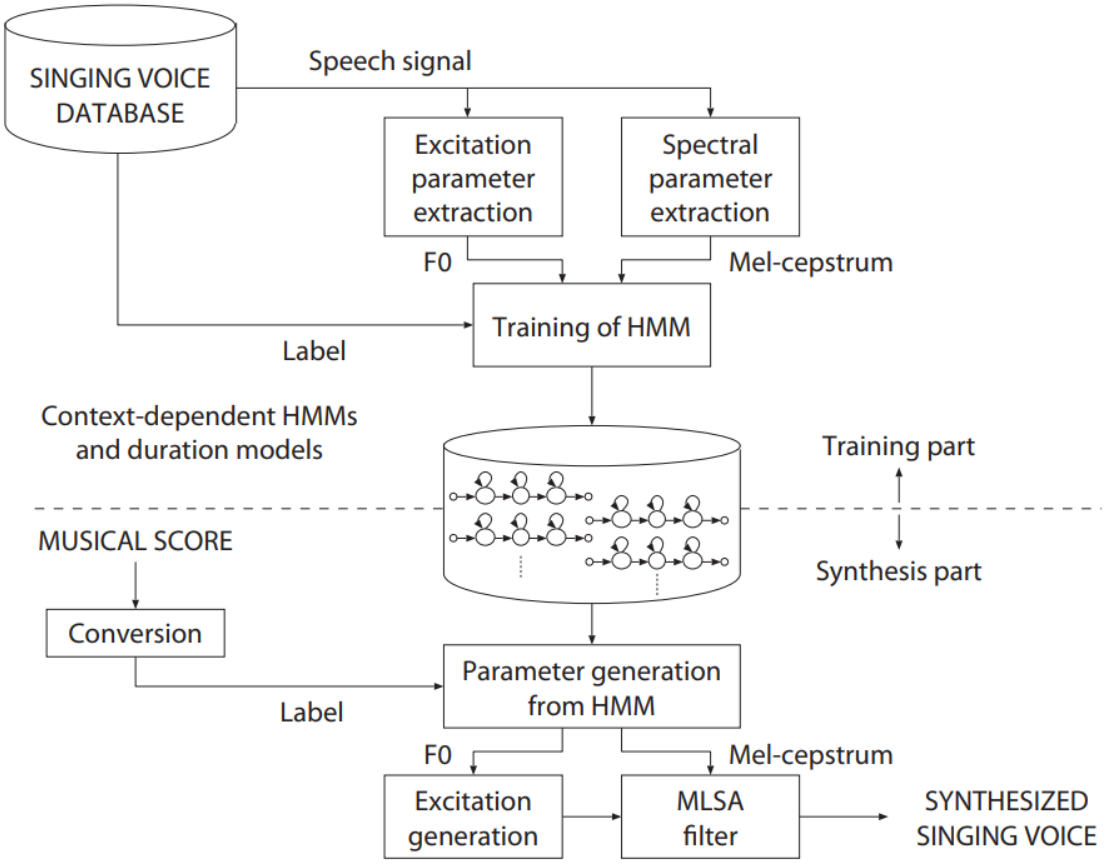


Image source: https://en.wikipedia.org/wiki/Uncanny_valley

Singing Voice Synthesis in 2010

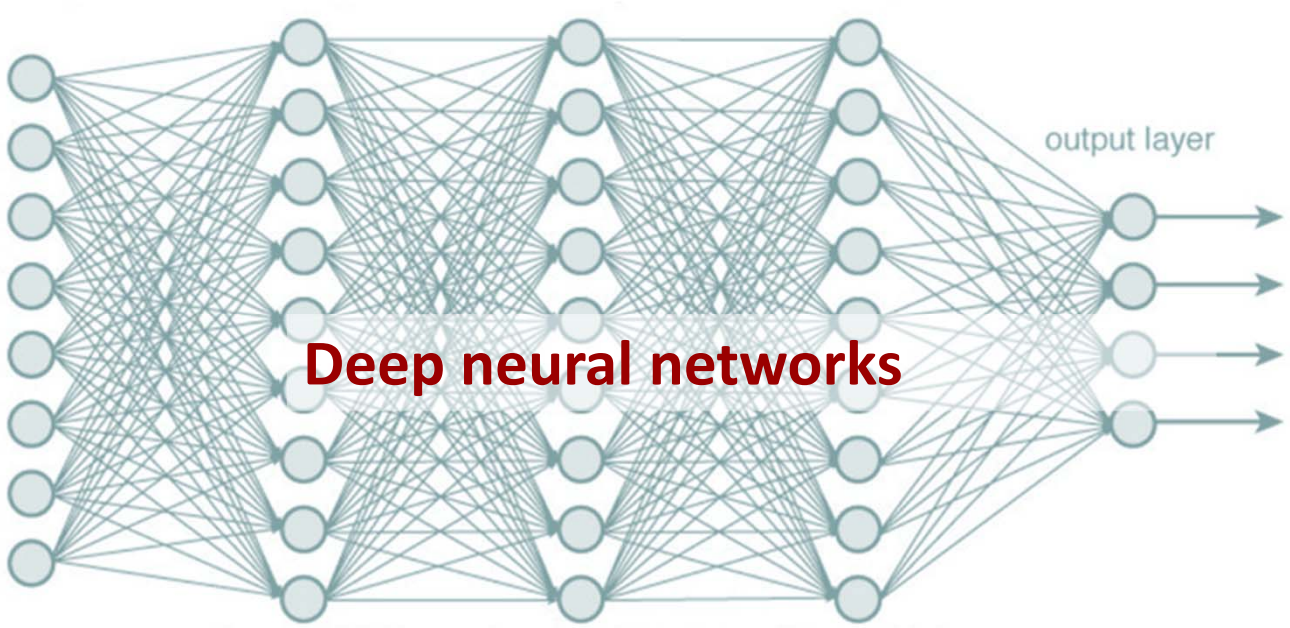
1. **Human knowledge**
2. **Lot's of signal processing**
3. **Few parameters**

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



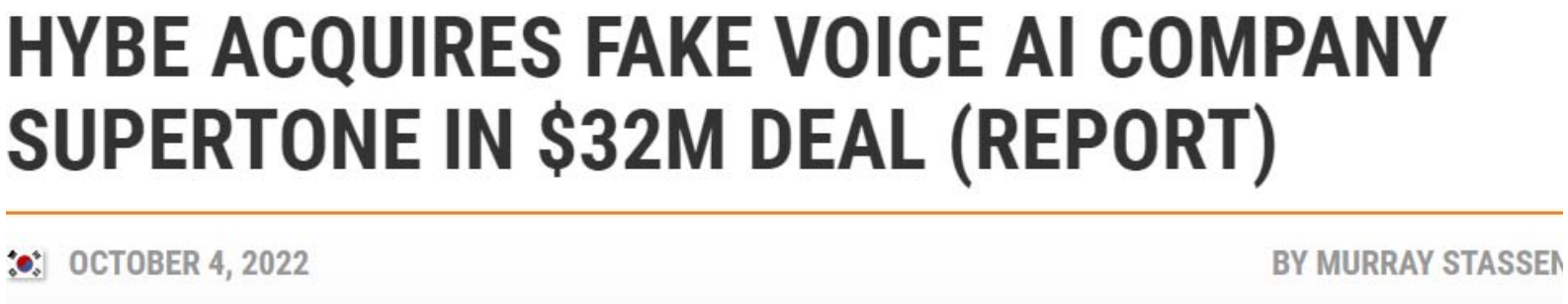
DL: It's about Learning Input/Output Relationships

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



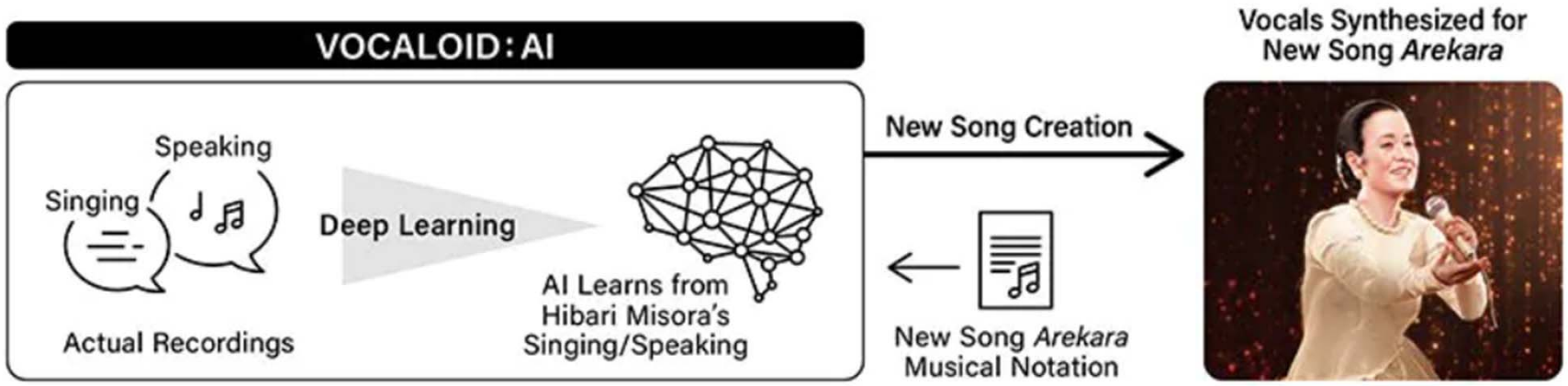
Research + Data

- **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

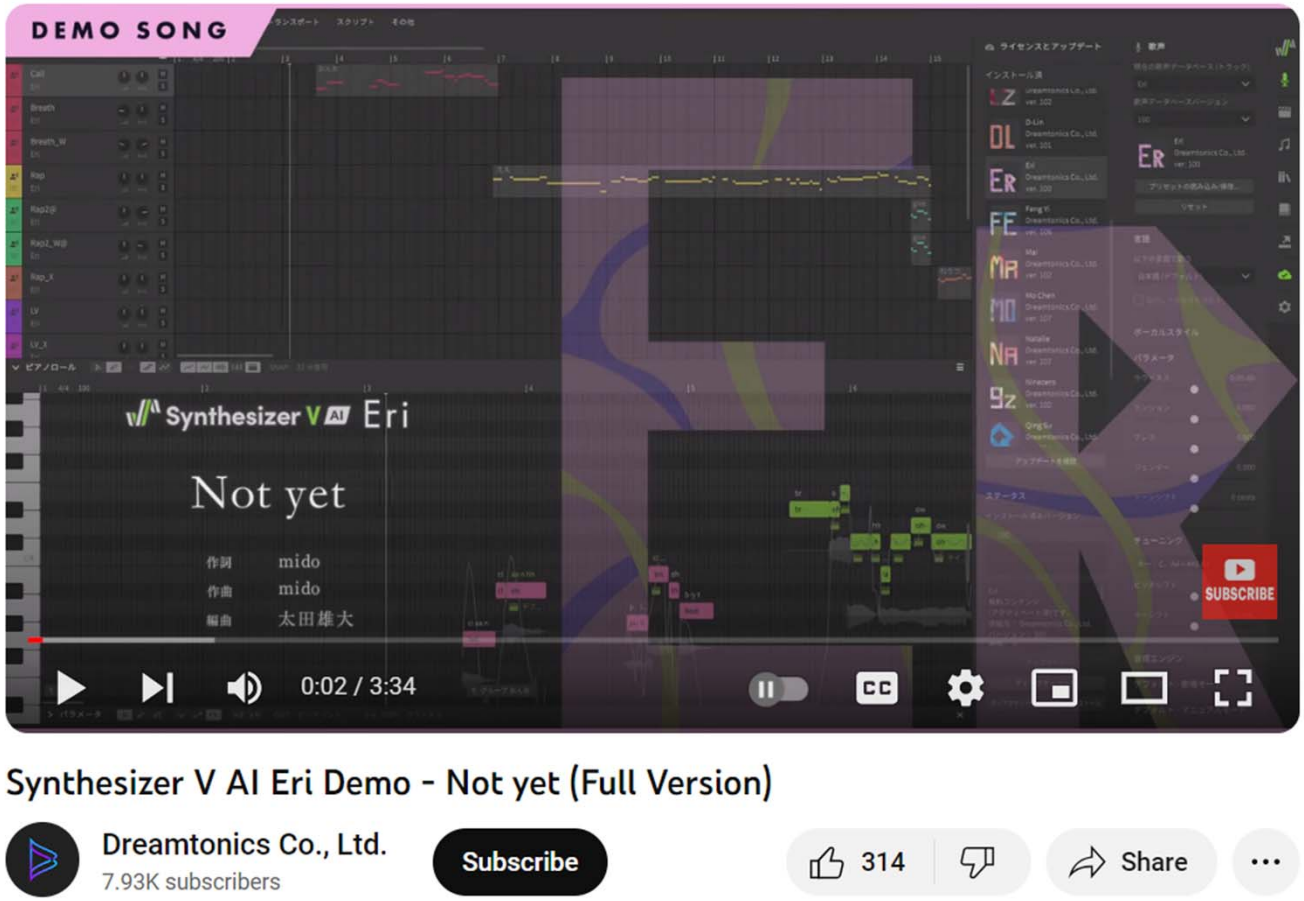


VOCALOID6 uses VOCALOID:AI, an AI-based technology that makes it possible to generate even more natural-sounding and highly expressive singing voices.

(Source: https://tw.yamaha.com/zh/news_events/2019/20191008_vocaloid_ai.html)

SynthV

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



Singing Synthesis at the Taiwan AI Labs

https://www.youtube.com/watch?v=nWTuZIRU80A

「音樂製作人的工作是無法被取代的」。AI vocal 要怎麼唱，能唱得多好，終究需要專業音樂製作人，以人類的美學和經驗去引導 AI，要如何將 AI 昇華到情感面，終究還是需要製作人的能力，以及對音樂的想像力。

作為一個仍在線上的歌手與製作人，由我親自處理自己的AI vocal，讓這首歌傳達出「創作者、歌者不怕 AI 的挑戰」、「我們擁有自己聲音的控制權」等訊息，同時也是「人類的思考和意志，才是人之所以為人」的巨大宣示。

透過聆聽《教我如何做你的愛人》，試著探討：「若 AI 已經能模擬原唱的一切，那麼原唱歌手的價值會是什麼？」

當 AI 真正學會唱歌之後，就是創作人與歌手，重新理解自身價值的時候了。....by公主

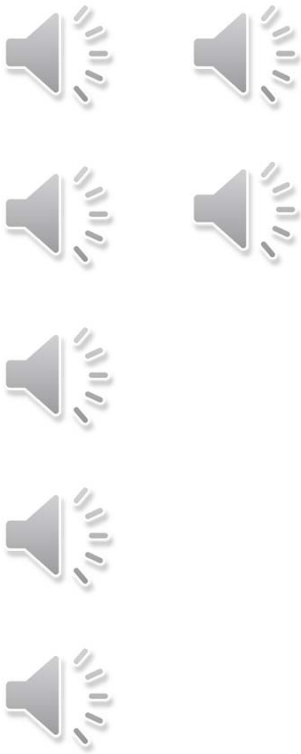
Our Own Experience

2019/08
AI phantom

2020/02
bad articulation

2021/06
intelligible

2021/10
AI Sandee v0



2022/06
better quality

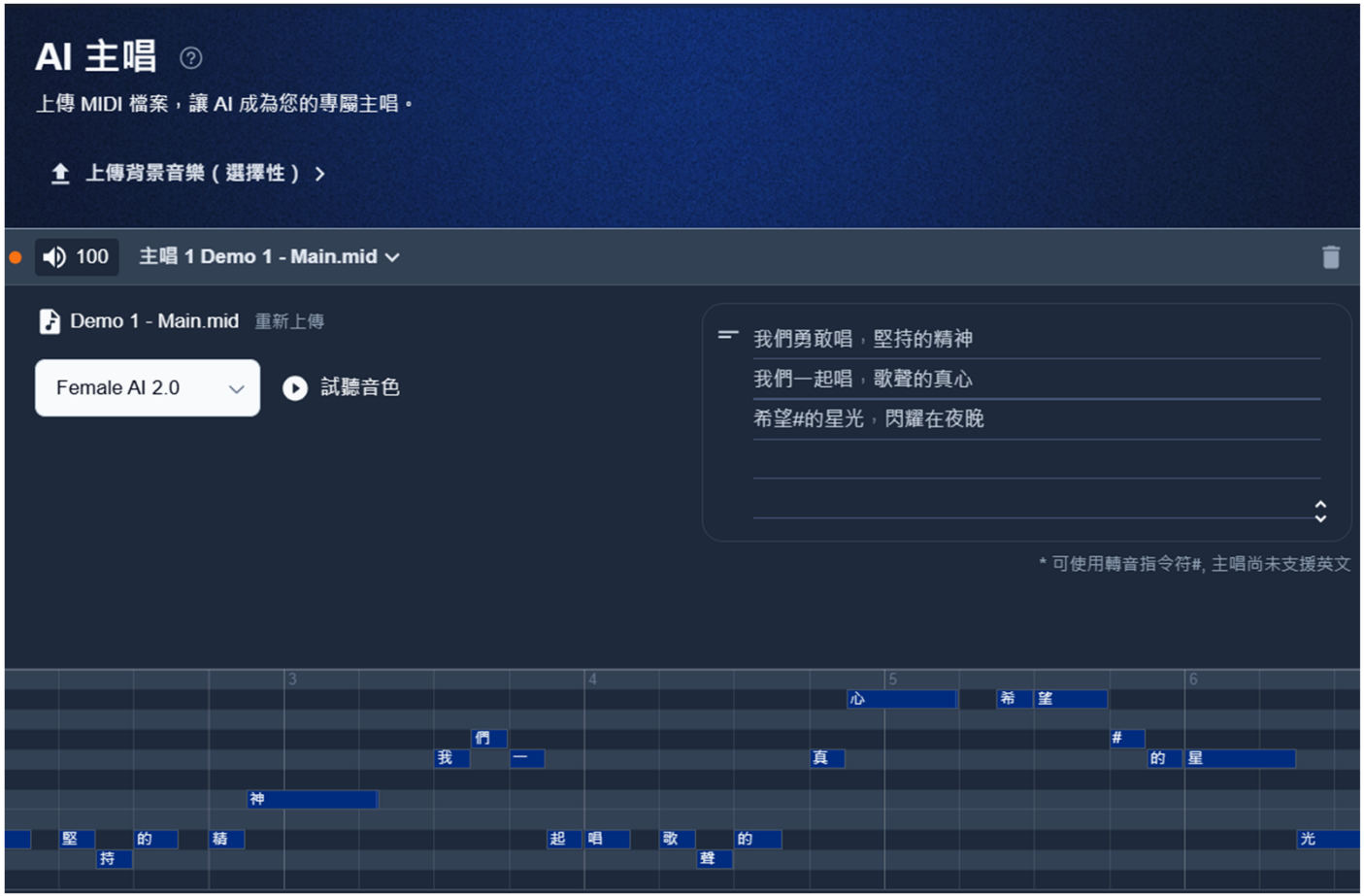
2022/12
close to ready

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



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

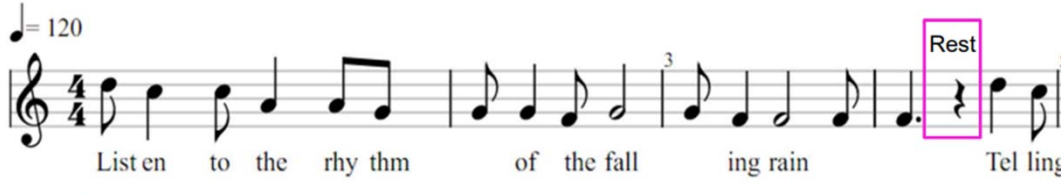
SVS vs SVC

- **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)

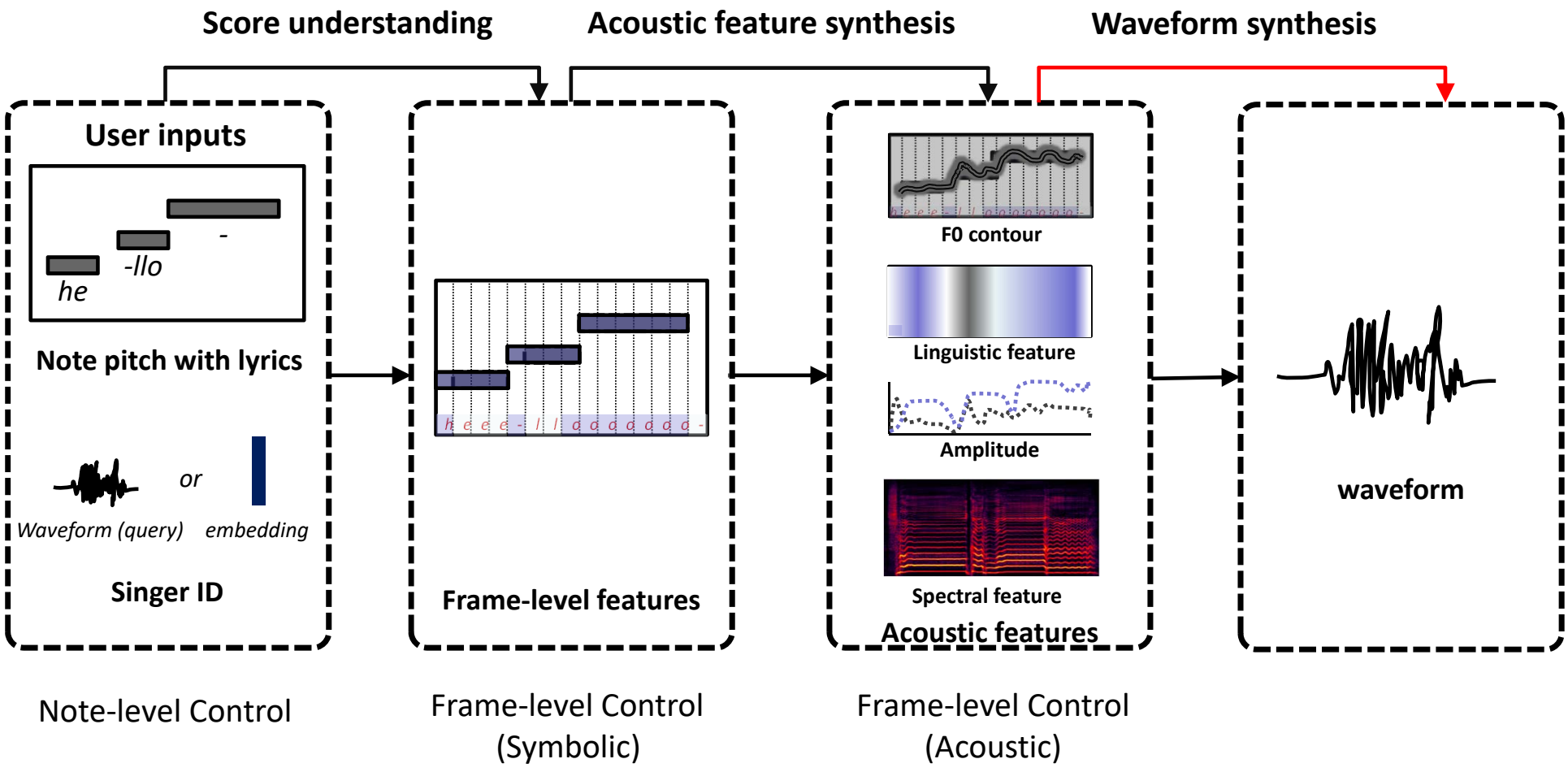


Lyrics	List	en	to	the	rhy	thm	of	the	fall	ing	rain	Tel	ing
Note	D5	C5	C5	A4	A4	G4	G4	F4	G4	F4	F4	D5	C5
Duration	0.5	1	0.5	1	0.5	1	1	0.5	2.5	1	4	1	0.5
Rest	0	0	0	0	0	0	0	0	0	0	0	1	0

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



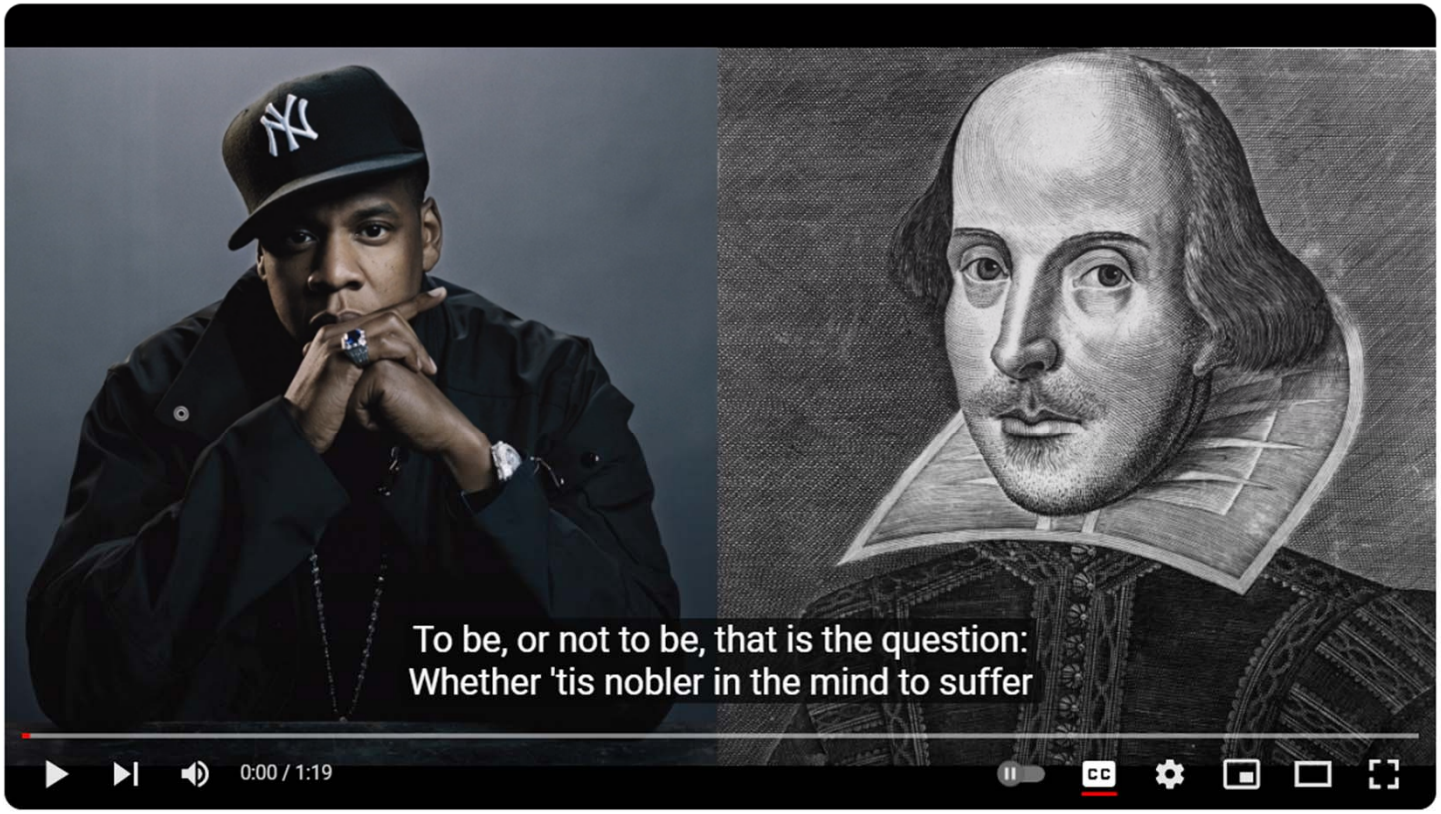
(Adapted from the slides of the ISMIR 2022 tutorial presented by Hyeong-Seok Choi)

Outline

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

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

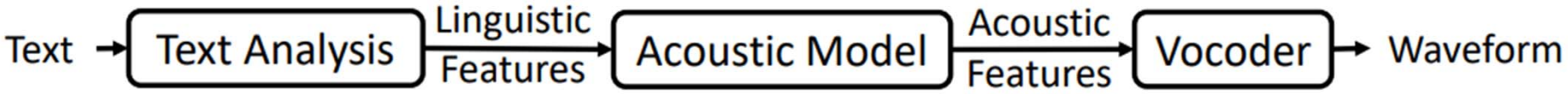


Figure 1: The three key components in neural TTS.

- 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)

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

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)** → "tʊ" "ə" "pleɪs"

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

- Question:
  - For **melody-to-lyrics generation**, how should we represent the lyrics? Character- or phoneme-based?
  - For **lyrics-to-melody generation**, how should we represent the lyrics? Character- or phoneme-based?
  - For **SVS generation**, character- or 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

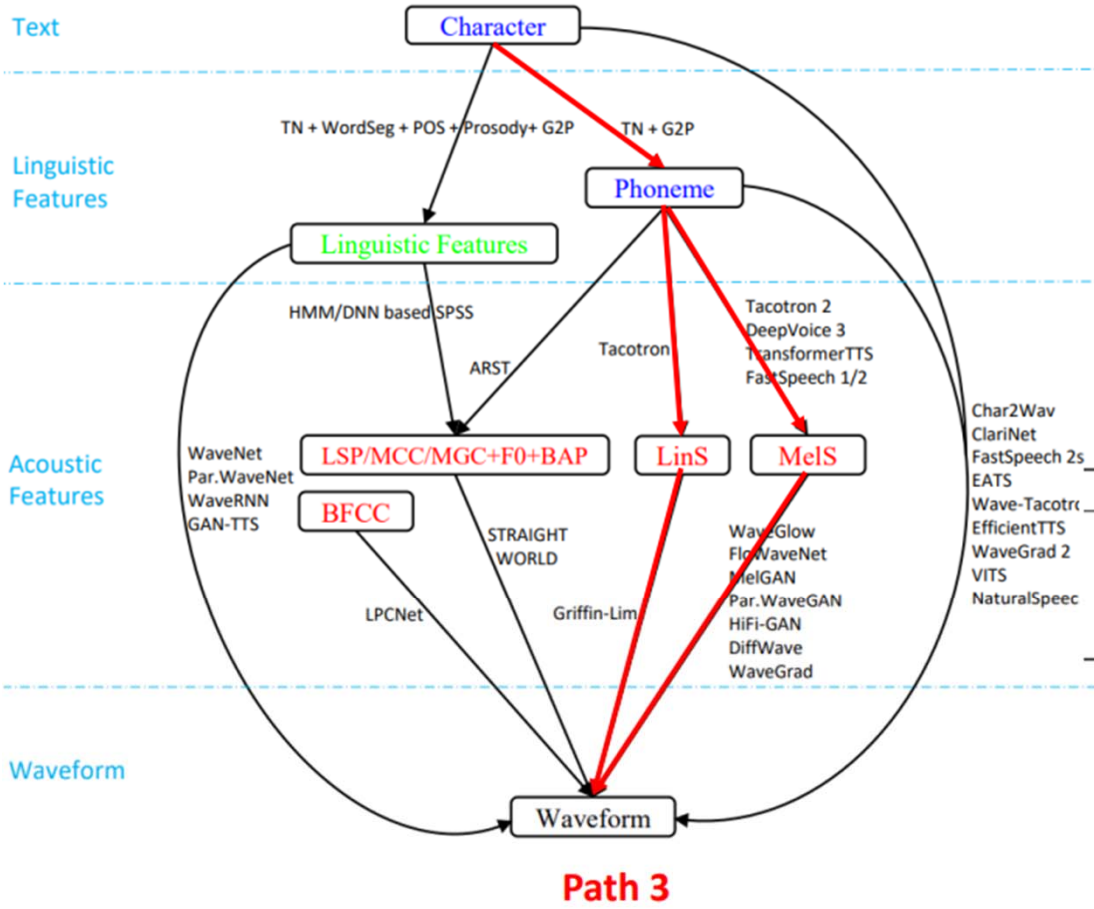
Phonemeizer for melody-to-lyrics generation

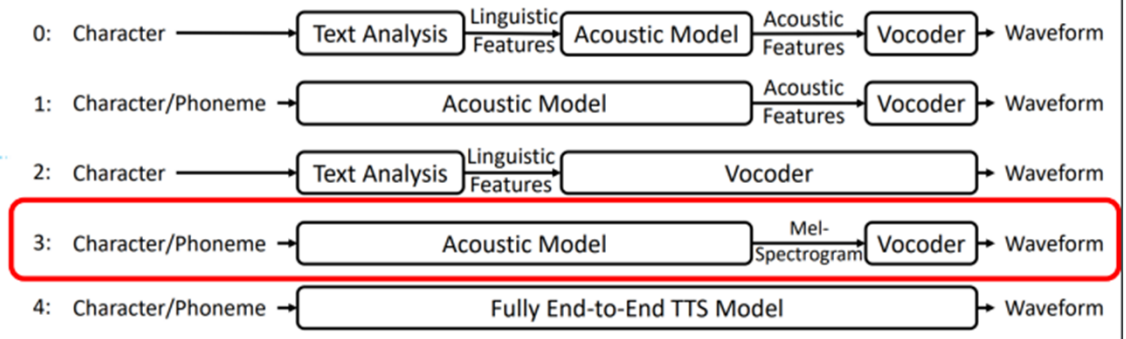
- **Strong-beat notes and weak-beat notes** refer to melody notes beginning at strong and weak beats, respectively. In 4/4 time music, the first and third beats are stronger than the second and fourth beats.
- **Long notes and short notes:** long notes are notes with relatively longer duration. Based 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 (['], [ˌ]) 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 [ː] 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

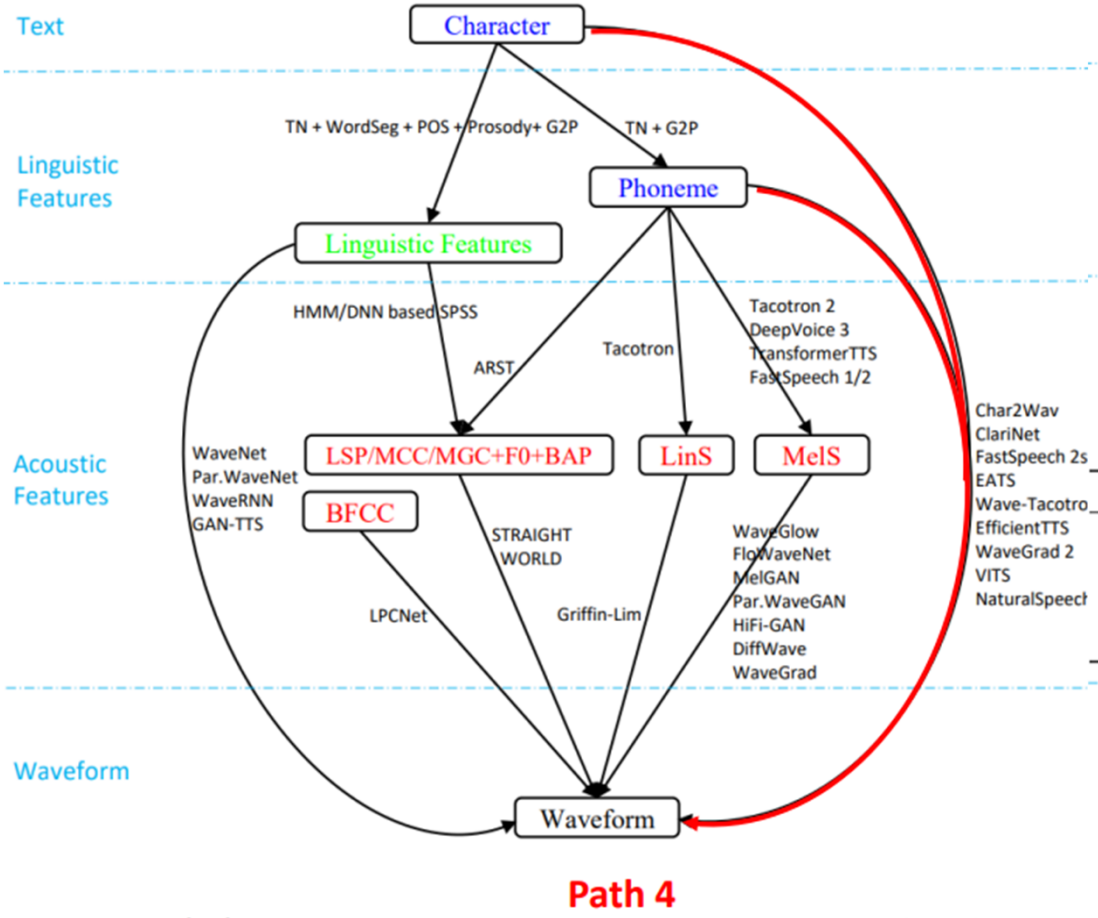


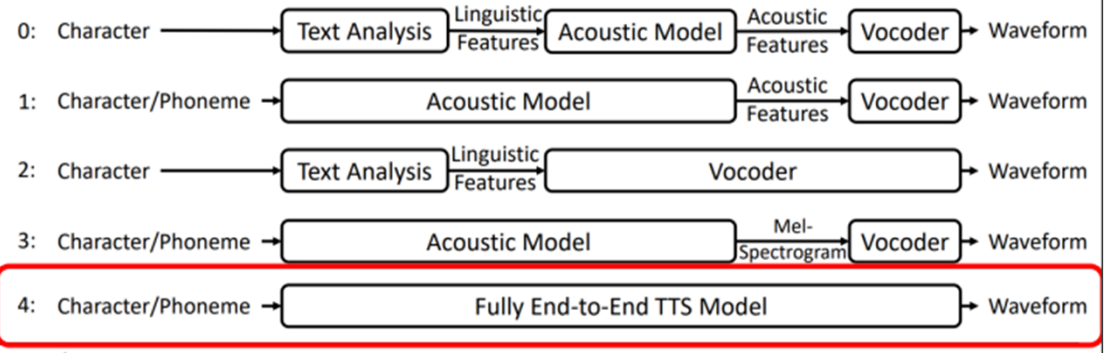


Stage	Models
0	SPSS [418, 358, 417, 427, 359]
1	ARST [377]
2	WaveNet [255], DeepVoice 1/2 [8, 88], Par. WaveNet [256], WaveRNN [151], HiFi-GAN [23]
3	DeepVoice 3 [271], Tacotron 2 [304], FastSpeech 1/2 [291, 293], WaveGlow [280], FloWaveNet [164]
4	Char2Wav [316], ClariNet [270], FastSpeech 2s [293],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

TTS

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

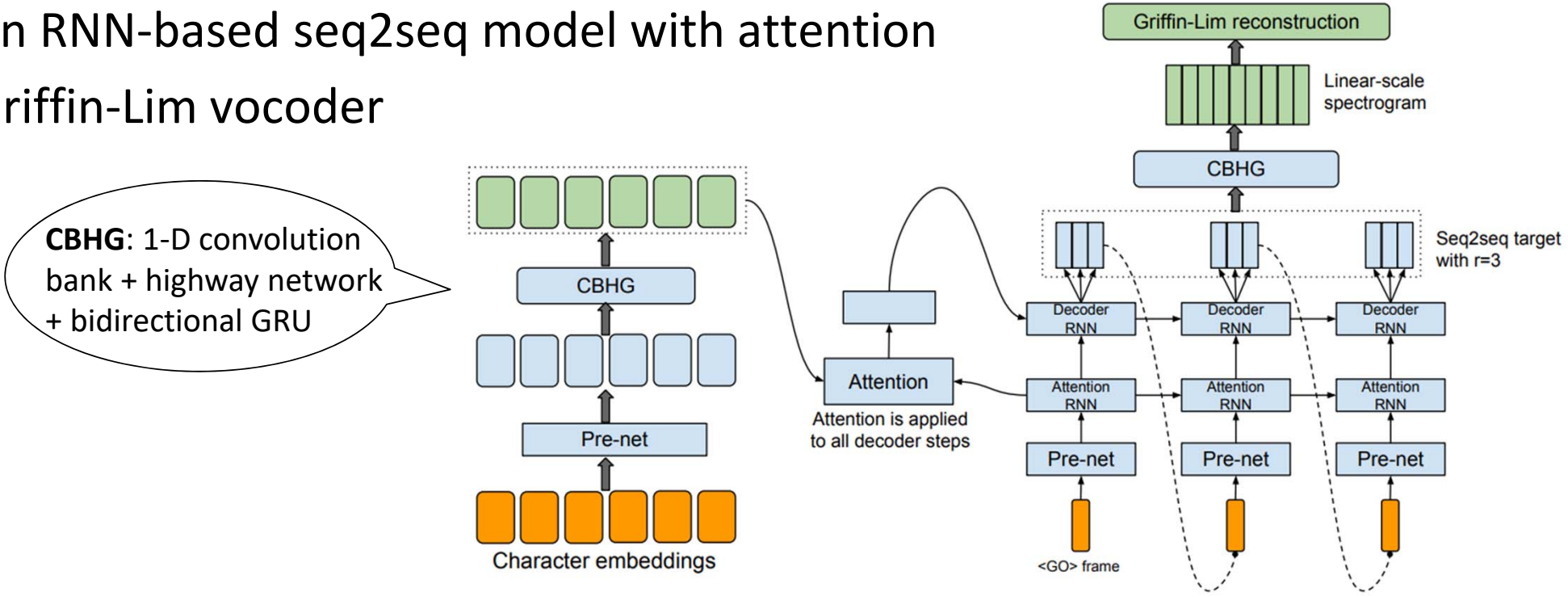




Stage	Models
0	SPSS [418, 358, 417, 427, 359]
1	ARST [377]
2	WaveNet [255], DeepVoice 1/2 [8, 88], Par. WaveNet [256], WaveRNN [151], HiFi-GAN [23]
3	DeepVoice 3 [271], Tacotron 2 [304], FastSpeech 1/2 [291, 293], WaveGlow [280], FloWaveNet [164]
4	Char2Wav [316], ClariNet [270], FastSpeech 2s [293],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

TacoTron

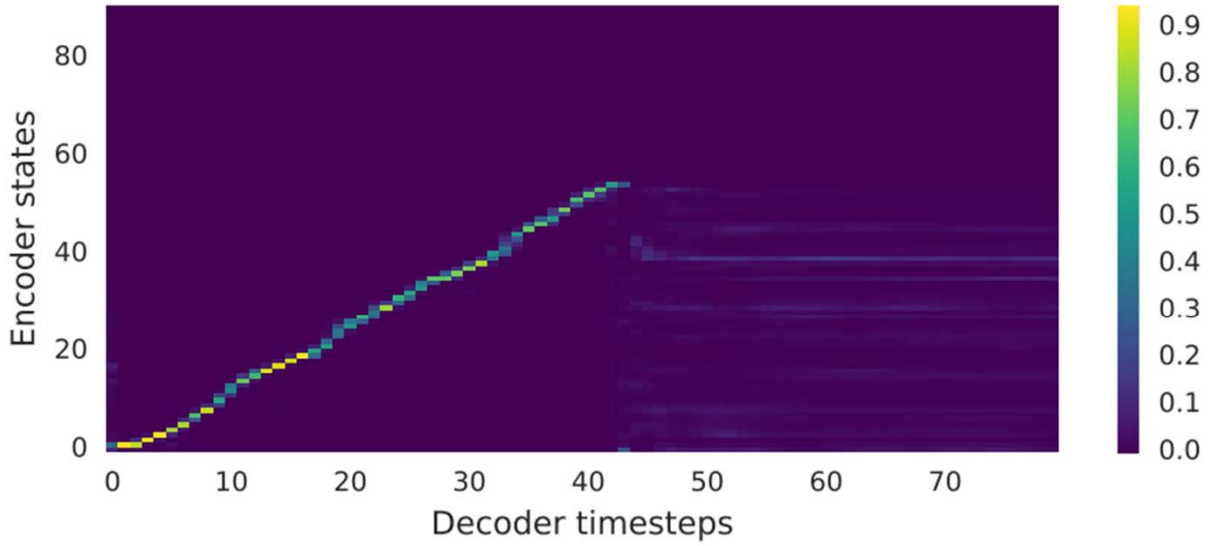
- Takes **characters** as input and **predict frames of (linear) spectrogram**
- **Autoregressive** (AR)
- An RNN-based seq2seq model with attention
- Griffin-Lim vocoder



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

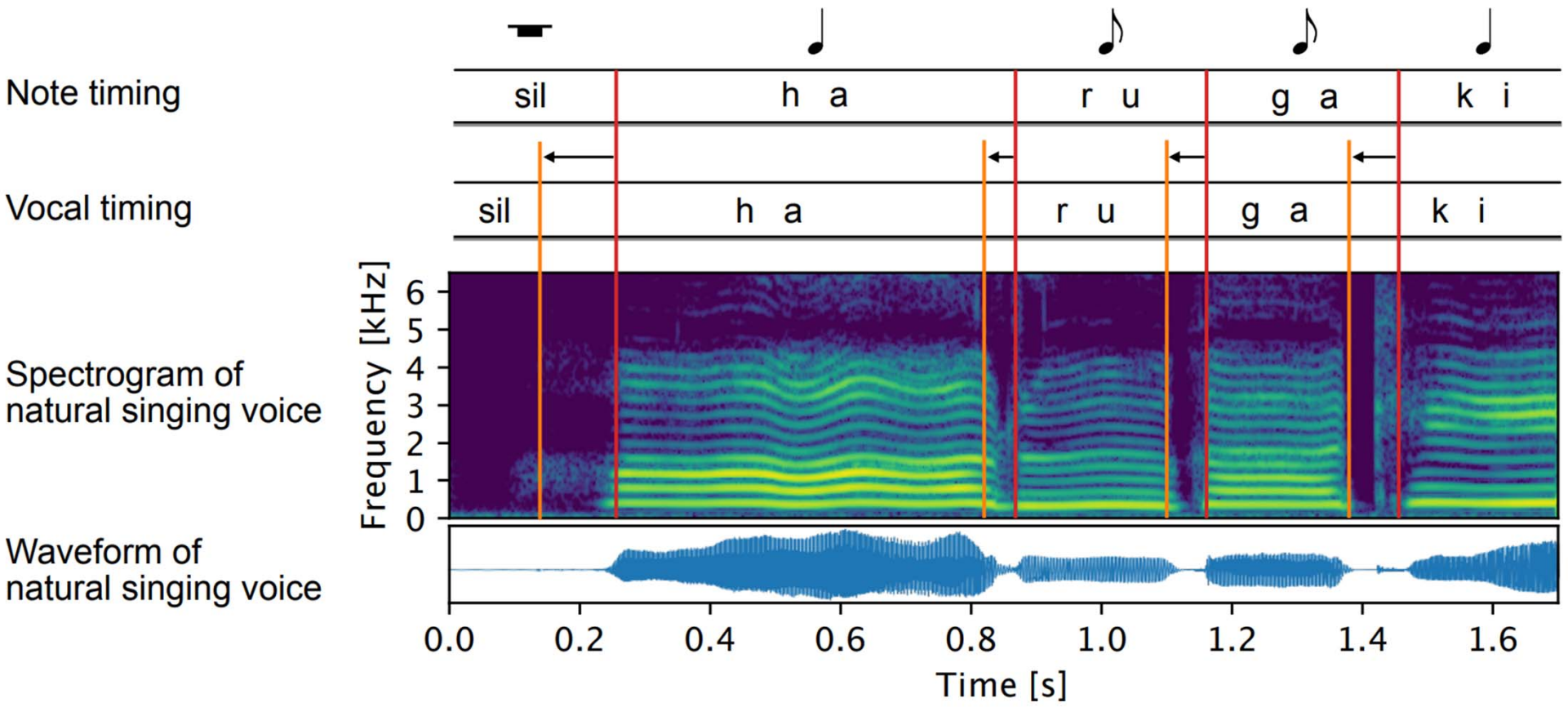
Why Attention?

- 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 have **different lengths**



(c) Tacotron (proposed)

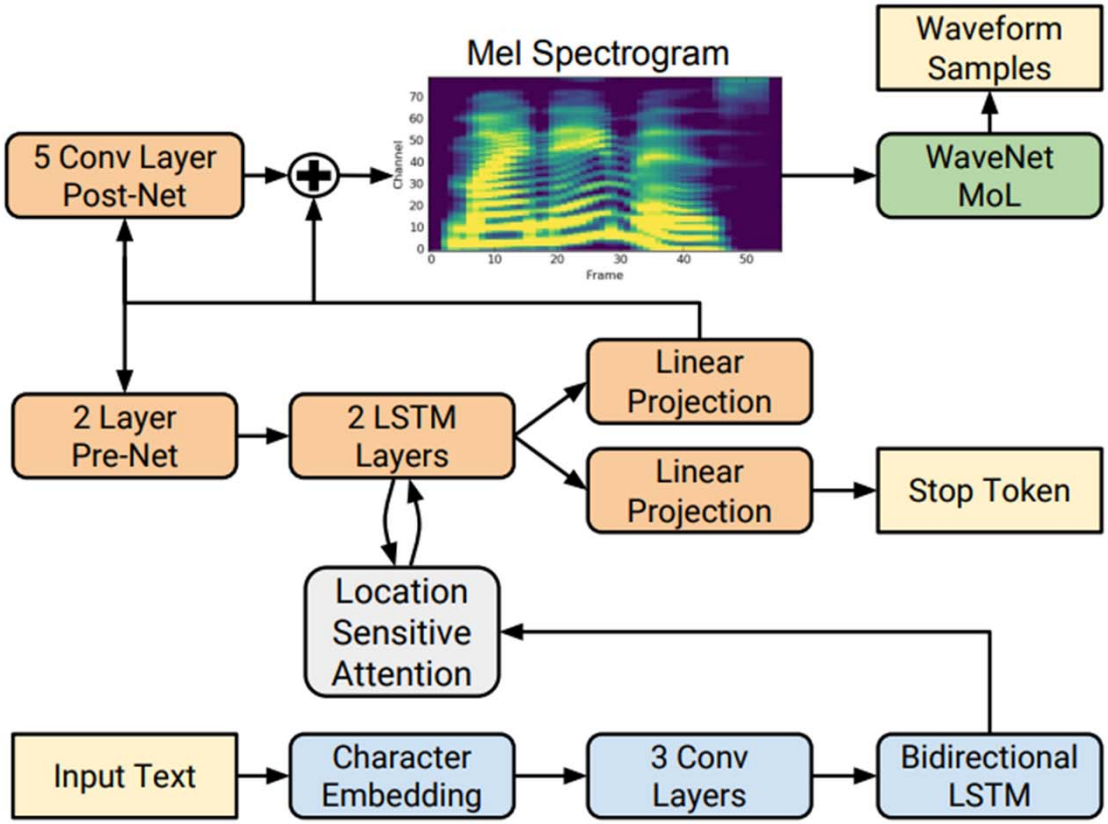
Text-Audio Alignment



TacoTron 2

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

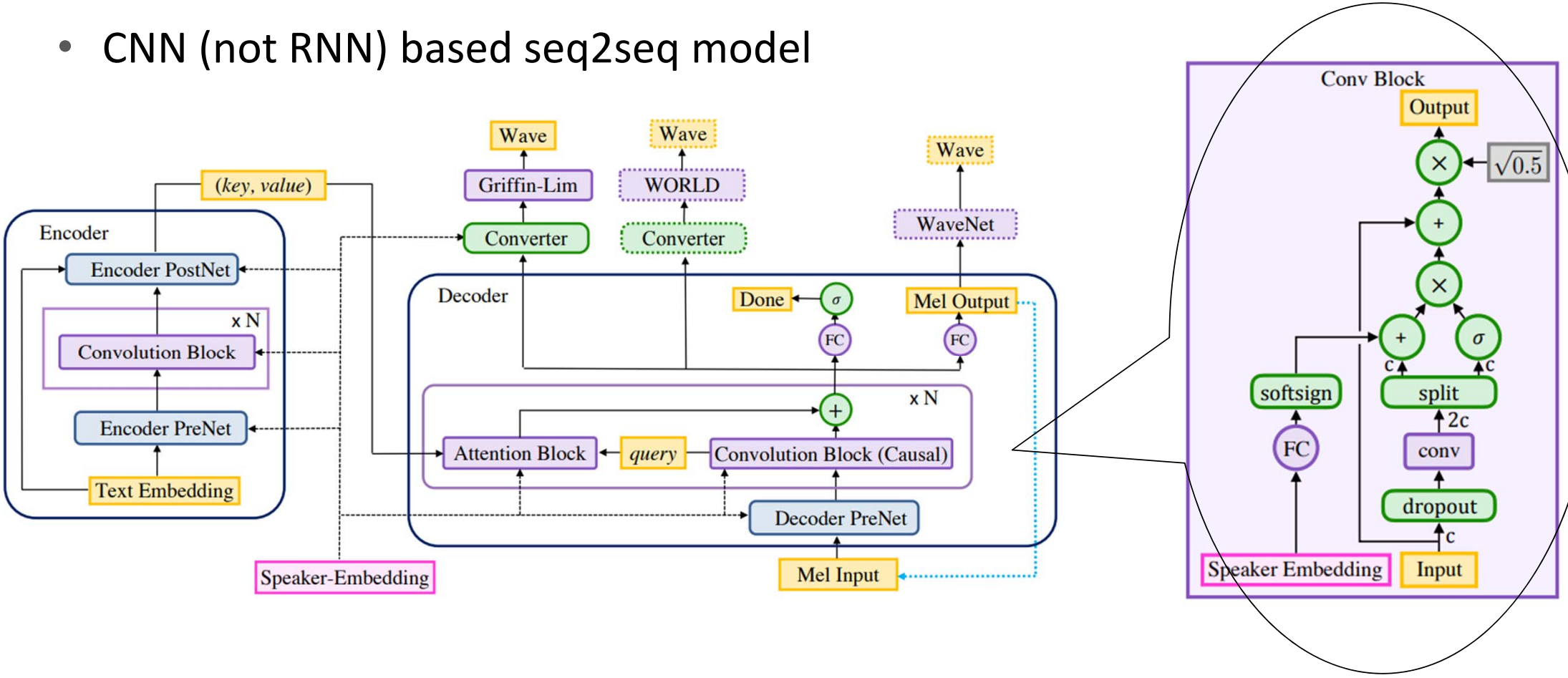
System	MOS
Parametric	3.492 ± 0.096
Tacotron (Griffin-Lim)	4.001 ± 0.087
Concatenative	4.166 ± 0.091
WaveNet (Linguistic)	4.341 ± 0.051
Ground truth	4.582 ± 0.053
Tacotron 2 (this paper)	$\mathbf{4.526 \pm 0.066}$



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

Deep Voice 3

- CNN (not RNN) based seq2seq model

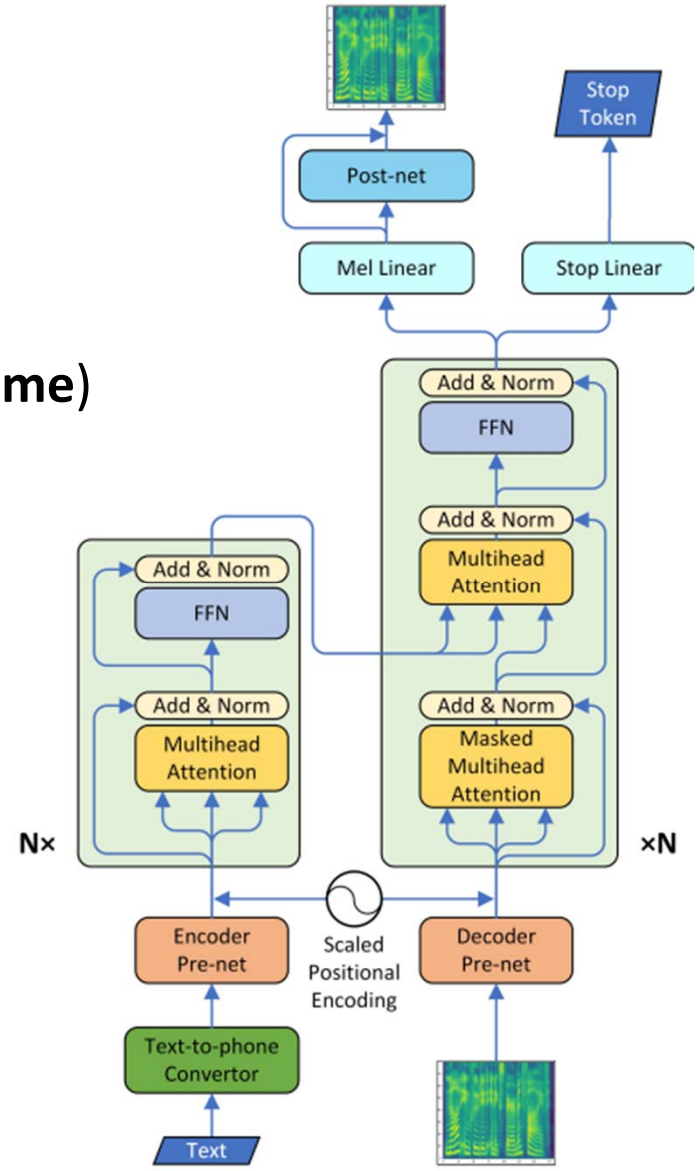


Ref: Ping et al, "Deep Voice 3: Scaling text-to-speech with convolutional sequence learning," ICLR 2018

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 AR-based Seq2seq models

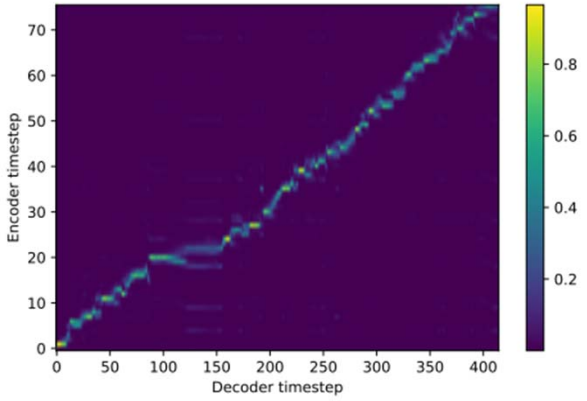
- **Inference is slow**
- **Synthesized speech is usually not robust**
  - Due to *error propagation* and the *wrong attention alignments* between text and speech in the autoregressive generation, the generated mel-spectrogram is usually deficient with the problem of **words skipping** and **repeating**
- **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

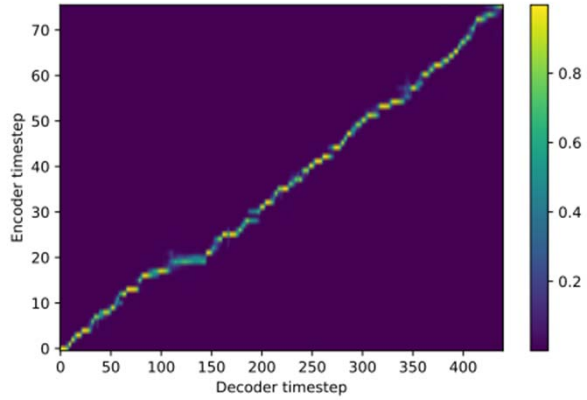
Robustness Problem About Alignment

- 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

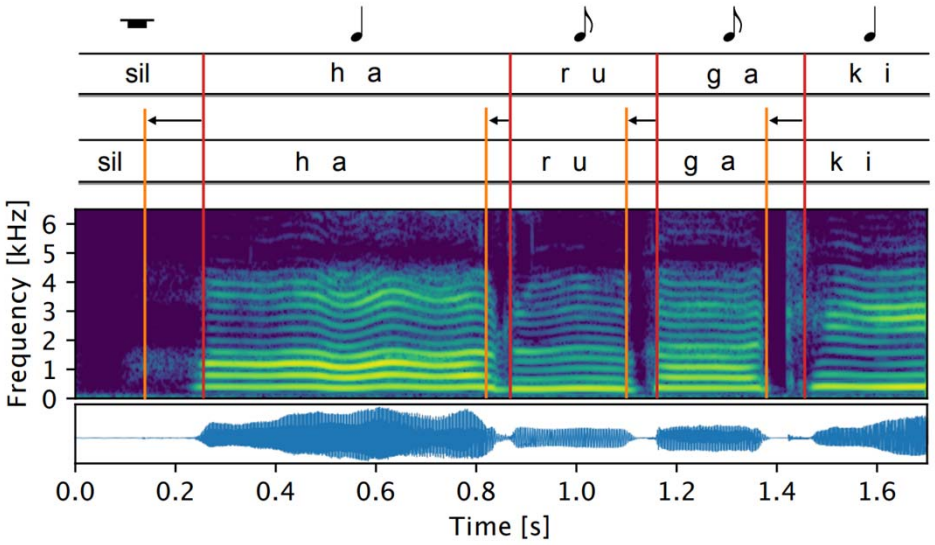
- **Inference is slow → Use non-autogressive models!**
- **Synthesized speech is usually not robust → Predict phoneme duration!** (i.e. predict onset and offset times)
  - Due to *error propagation* and the *wrong attention alignments* between text and speech in the autoregressive generation, the generated 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

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

Predict Phoneme Duration?

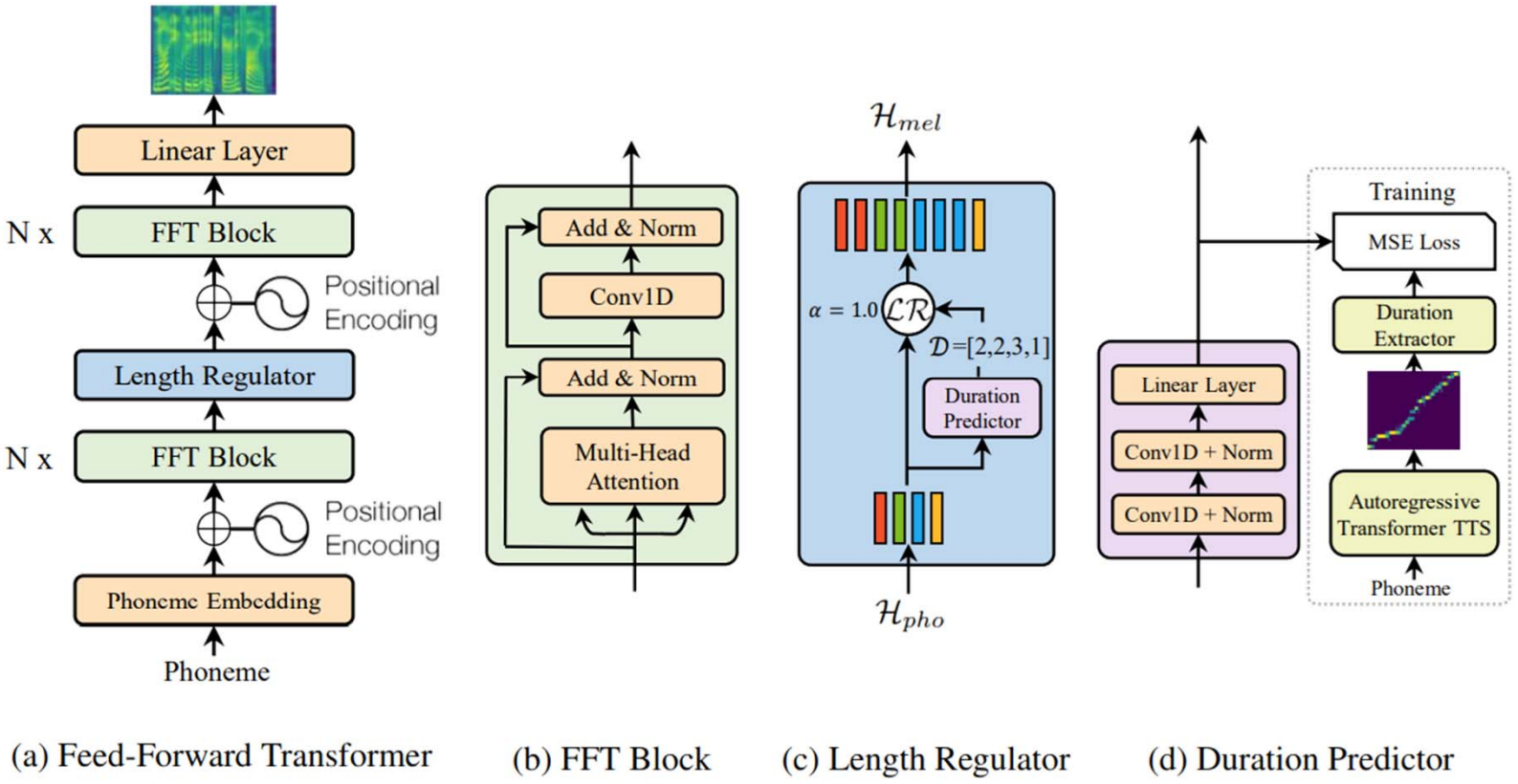
- Cross-attention alignment can be an overkill, because the alignment here for TTS between the input and output sequences must be monotonic
  - For other tasks such as machine translation (MT), the alignment may *NOT* be monotonic, making cross-attention a good choice though

- 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

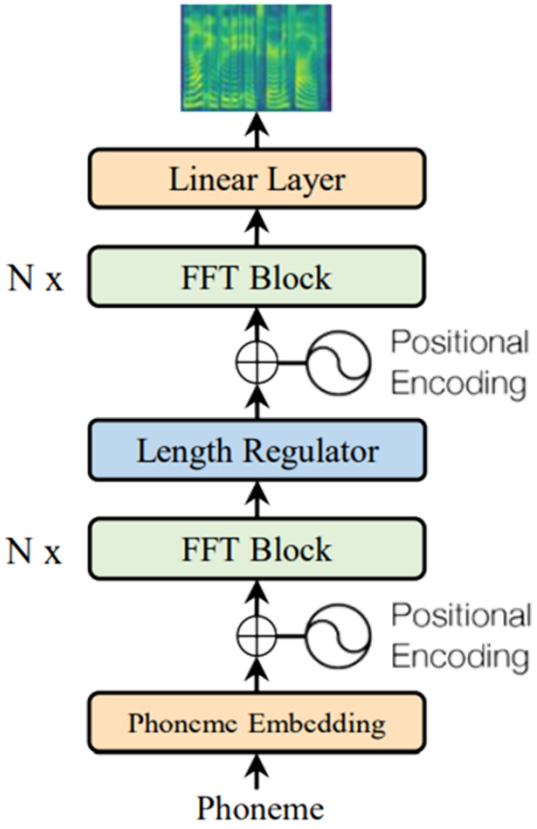
https://speechresearch.github.io/fastspeech/



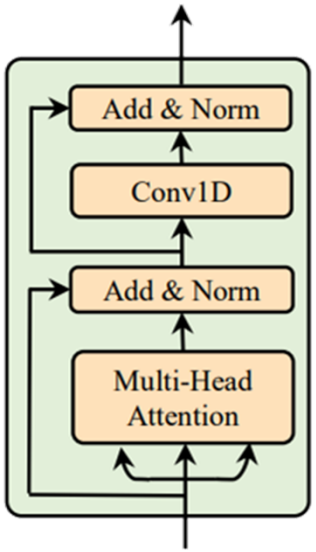
(a) Feed-Forward Transformer (b) FFT Block (c) Length Regulator (d) Duration Predictor

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

FastSpeech



(a) Feed-Forward Transformer



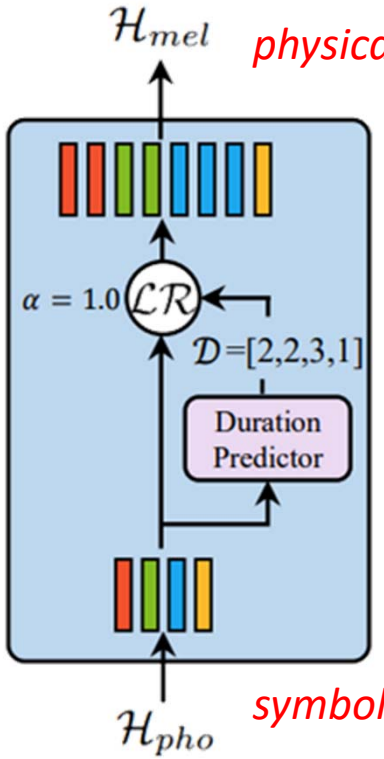
(b) FFT Block

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

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*

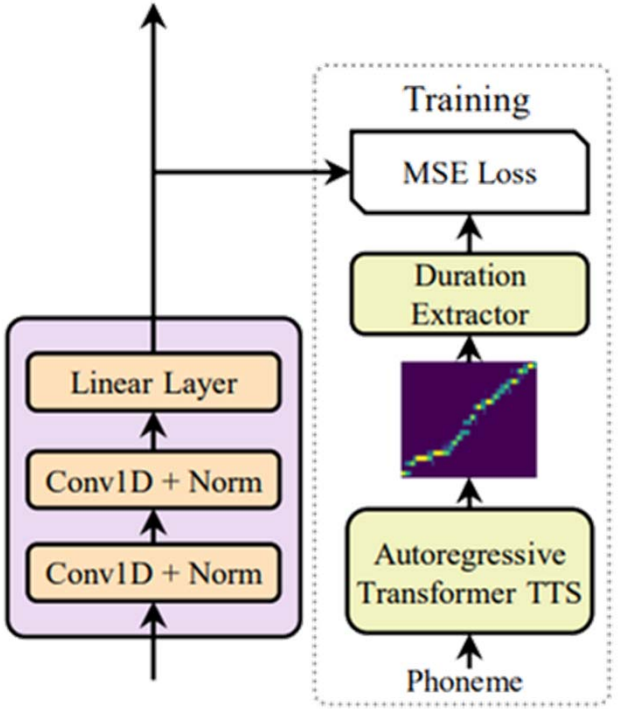


(c) Length Regulator

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

FastSpeech

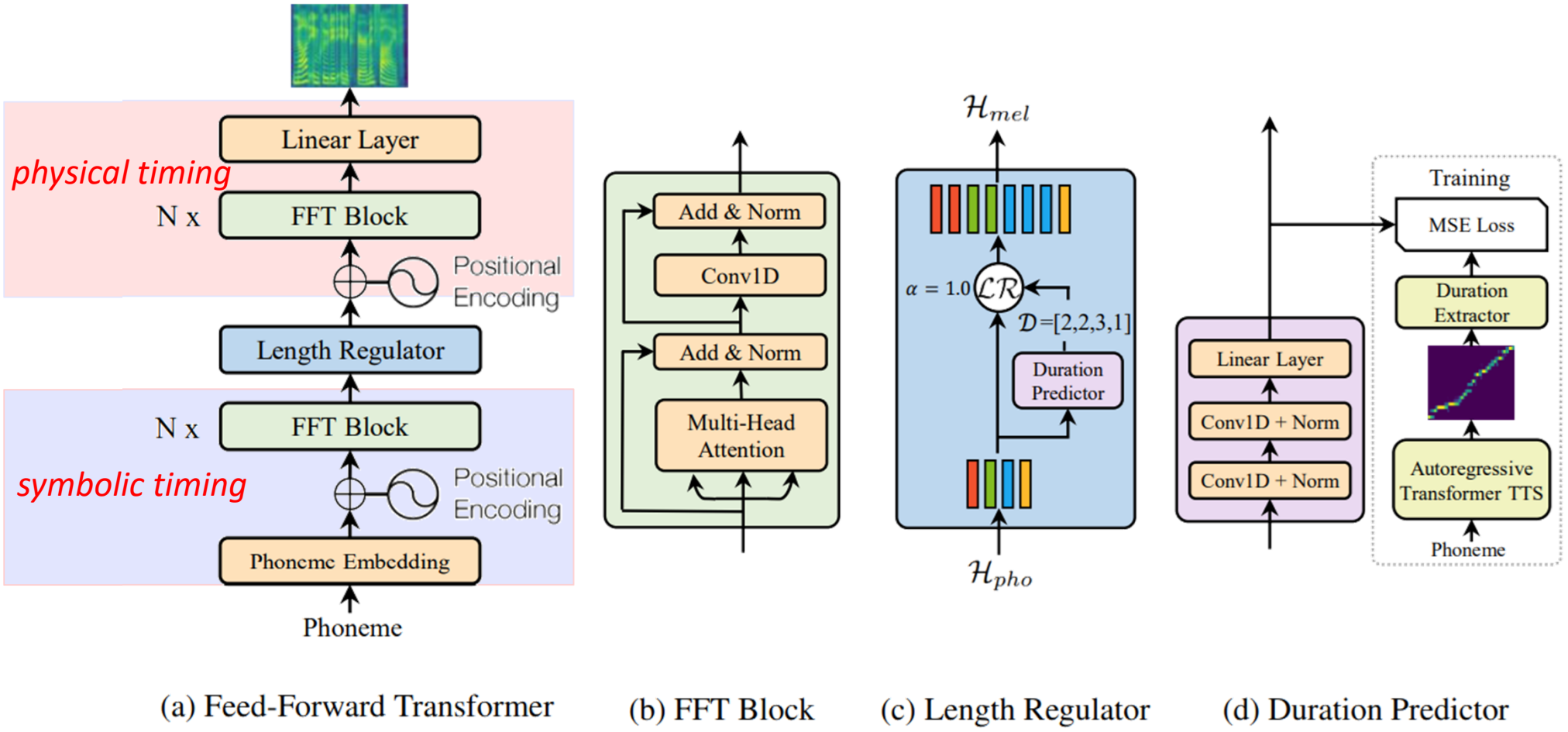
- **Phoneme duration predictor**
  - Produces *hard alignments*
    - while AR models use soft and automatic attention alignments
  - Predict the length in the **logarithmic** domain and use **MSE loss**
  - **Jointly trained** with FFT layers
- Ground truth phoneme duration for training?
  - From an **autoregressive *teacher*** model
  - Via **Montreal forced alignment** (MFA) https://github.com/MontrealCorpusTools/Montreal-Forced-Aligner
  - Or by **manually labeling**



(d) Duration Predictor

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

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: 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.

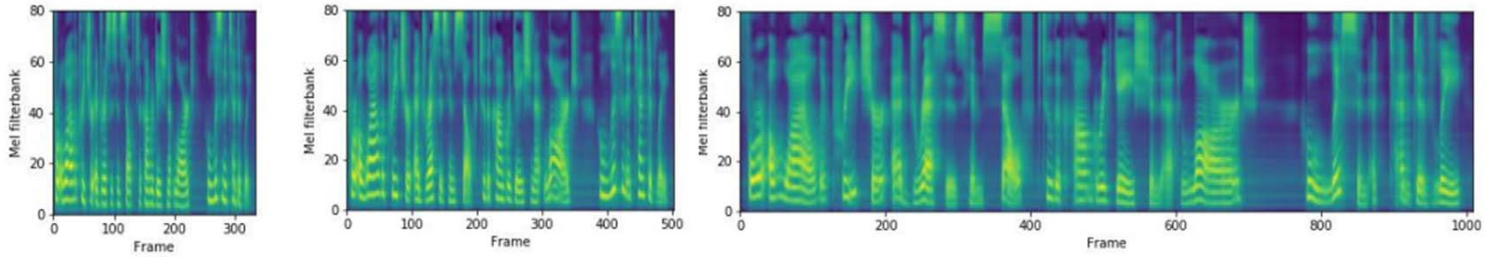
Method	Latency (s)	Speedup
Transformer TTS [14] (Mel)	6.735 ± 3.969	/
FastSpeech (Mel)	0.025 ± 0.005	269.40×
Transformer TTS [14] (Mel + WaveGlow)	6.895 ± 3.969	/
FastSpeech (Mel + WaveGlow)	0.180 ± 0.078	38.30×

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%

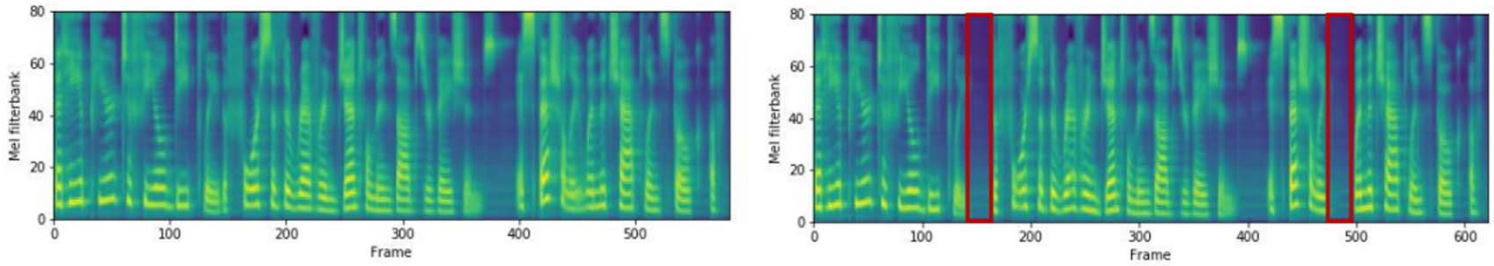
Table 3: The comparison of robustness between FastSpeech and other systems on the 50 particularly hard sentences. Each kind of word error is counted at most once per sentence.

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*".

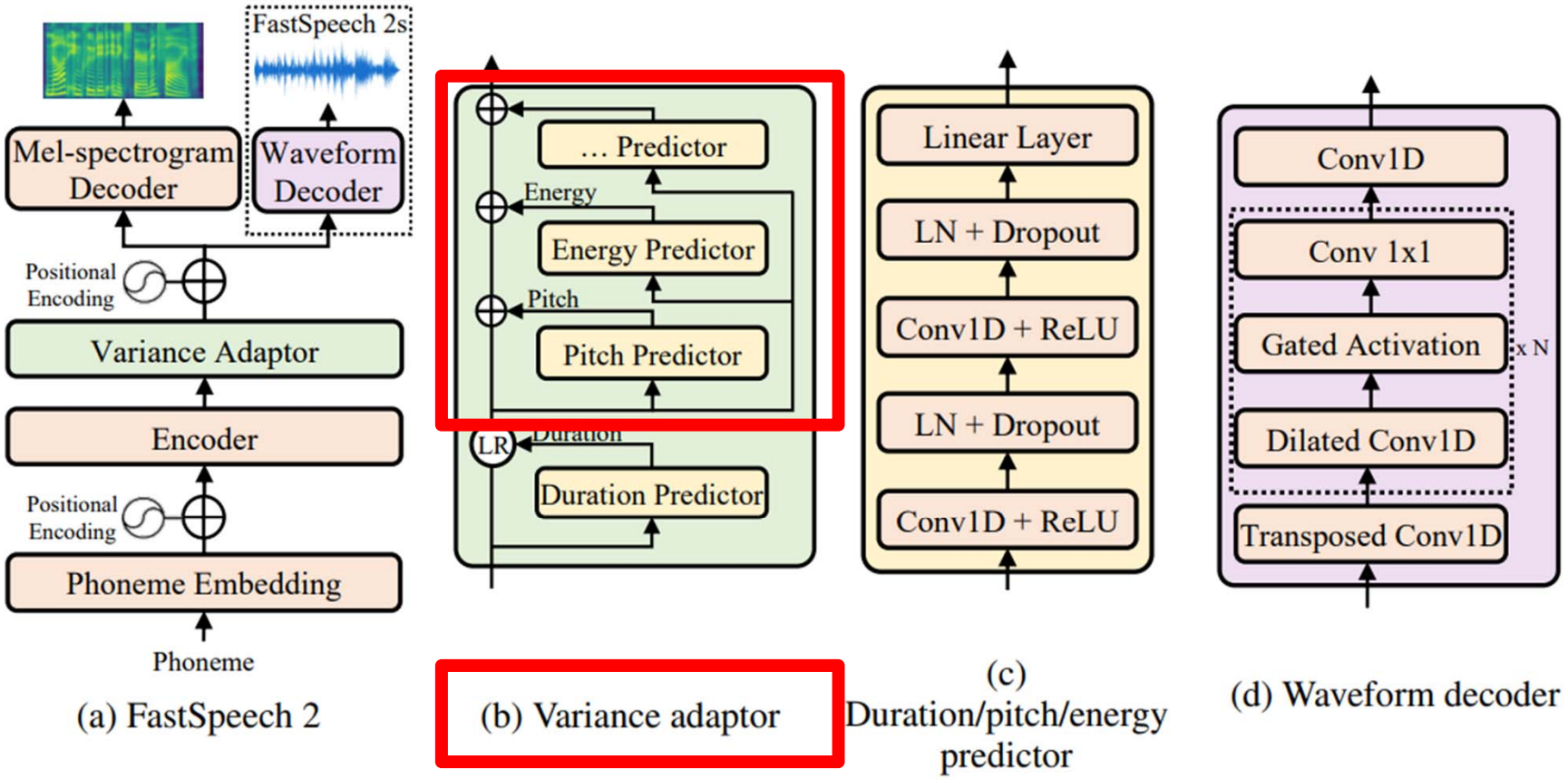


(a) Original Mel-spectrograms (b) Mel-spectrograms after adding breaks

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

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



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

FastSpeech 2 vs FastSpeech

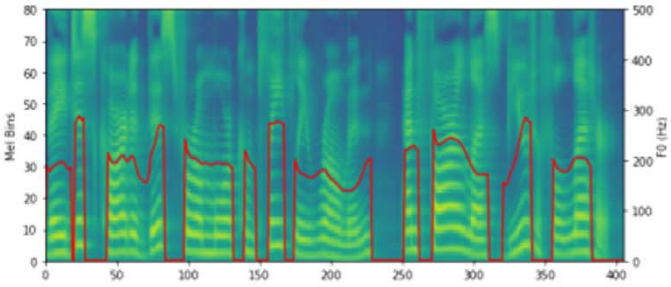
- 3x training speed-up
  - Simplifies to a single-stage end-to-end pipeline without teacher dependency
- More controllable
  - Pitch & energy

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

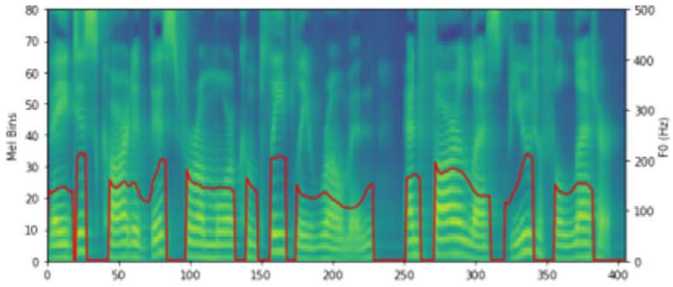
Method	Training Time (h)	Inference Speed (RTF)	Inference Speedup
Transformer TTS (Li et al., 2019)	38.64	9.32×10^{-1}	/
FastSpeech (Ren et al., 2019)	53.12	1.92×10^{-2}	48.5×
FastSpeech 2	**17.02**	1.95×10^{-2}	47.8×
FastSpeech 2s	92.18	$\mathbf{1.80 \times 10^{-2}}$	**51.8**×

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

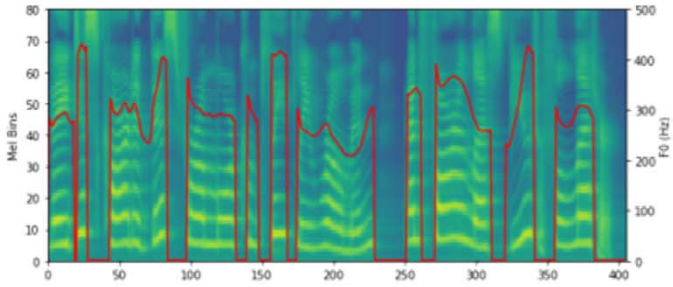
FastSpeech 2: Controllability



(a) $\hat{F}_0 = F_0$



(b) $\hat{F}_0 = 0.75F_0$

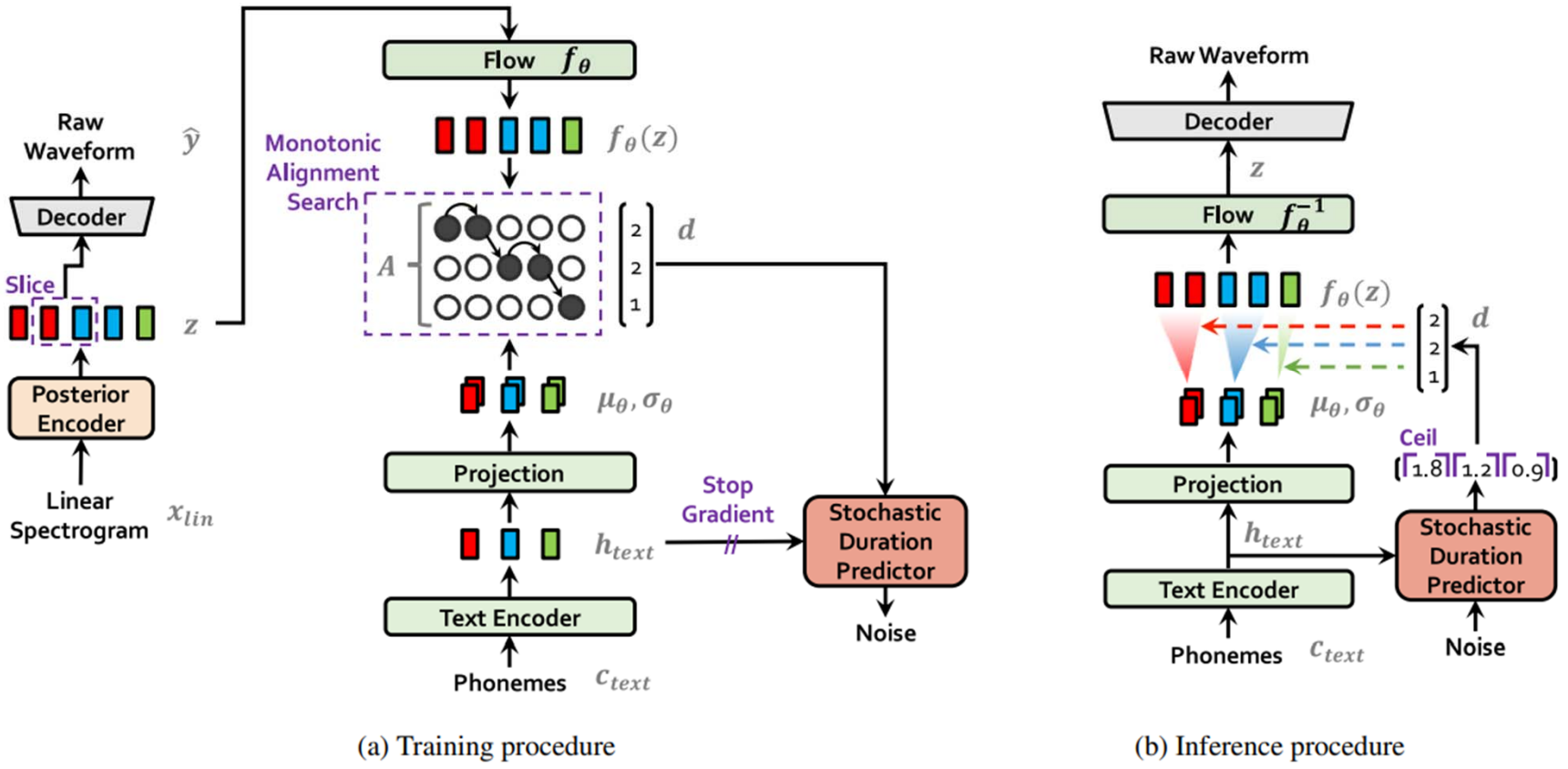


(c) $\hat{F}_0 = 1.50F_0$

Ref: Ren et al, “FastSpeech 2: Fast and high-quality end-to-end text to speech,” ICLR 2021

VITS: Flow-based TTS

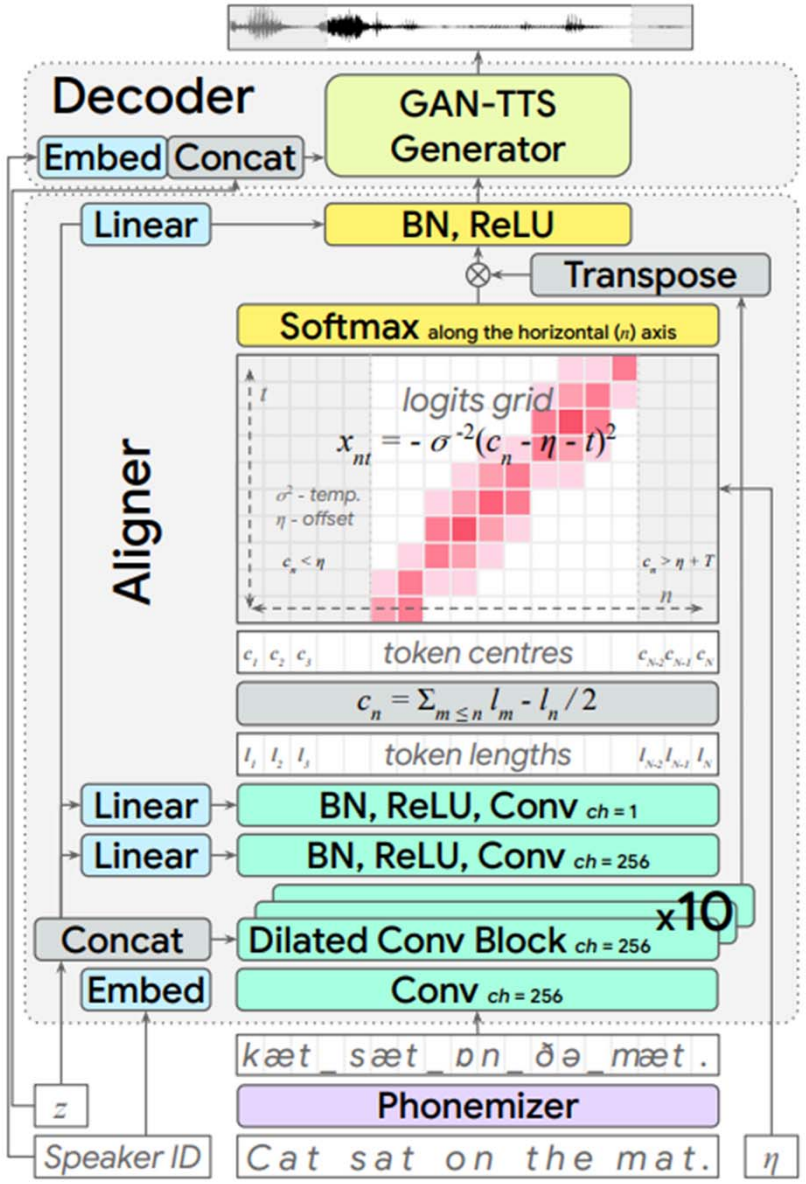
https://jaywalnut310.github.io/vits-demo/



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 These Non-autoregressive Models

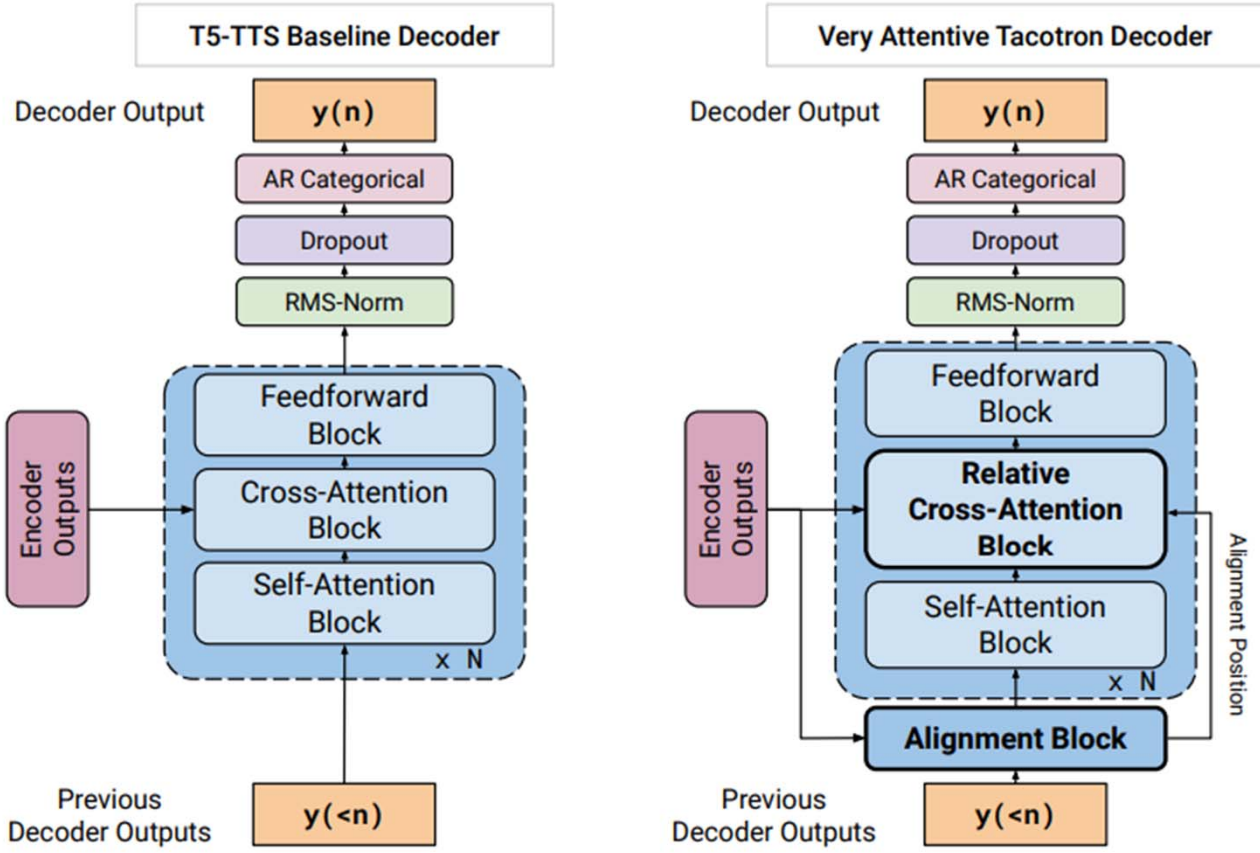
- It's kind of deterministic; **lack randomness**
  - In contrast, in autoregressive models, we do *sampling*, which introduces randomness
- Result of autoregressive models can sometimes sound **more natural**
  - Possibly due to the autoregressive prediction

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

Very Attentive Tacotron

Robust and unbounded length generalization in autoregressive Transformer-based TTS

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

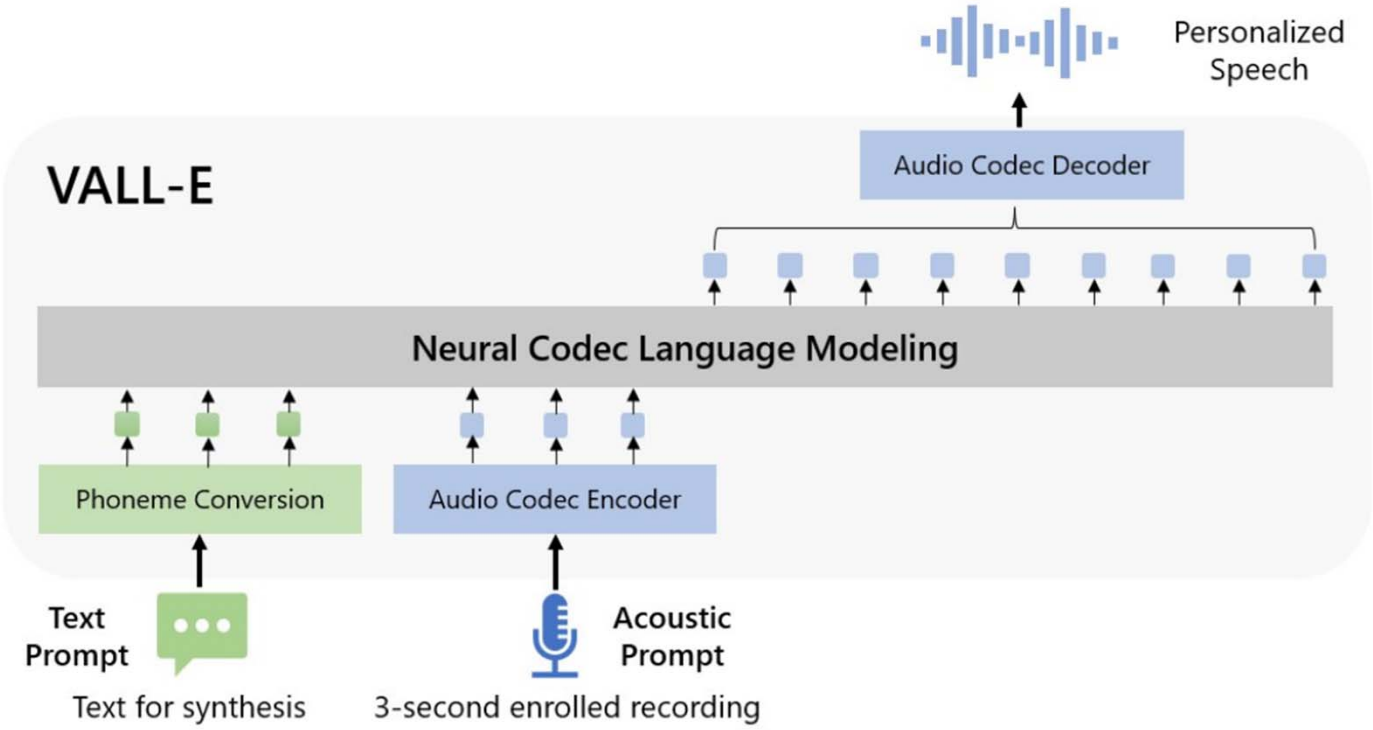


Ref: Battenberg et al, "Very Attentive Tacotron: Robust and unbounded length generalization in autoregressive Transformer-based text-to-speech," arXiv 2024

VALL-E

https://www.microsoft.com/en-us/research/project/vall-e-x/

- Use **audio codec models** to turn audio into **tokens**



Ref: Wang et al, "Neural codec language models are zero-shot text to speech synthesizers," arXiv 2023

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

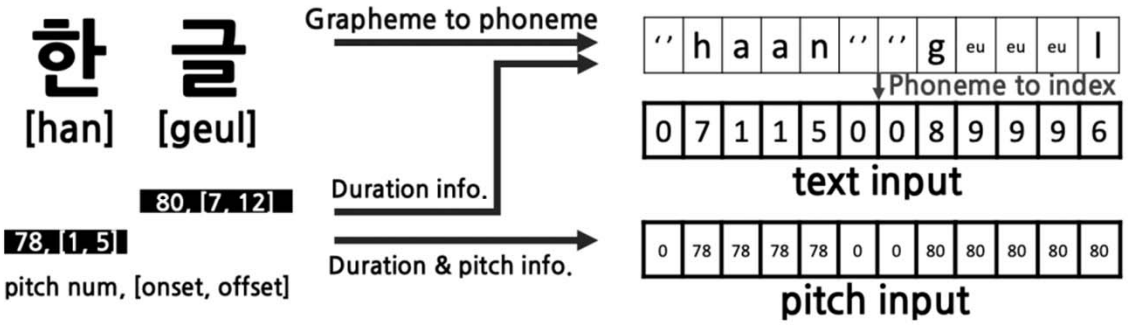
- **FastSpeech 2** (ICLR'21)
  - Use a feed-forward Transformer architecture
  - Non-autoregressive
  - Use duration predictor (among many other variance adaptors)

Outline

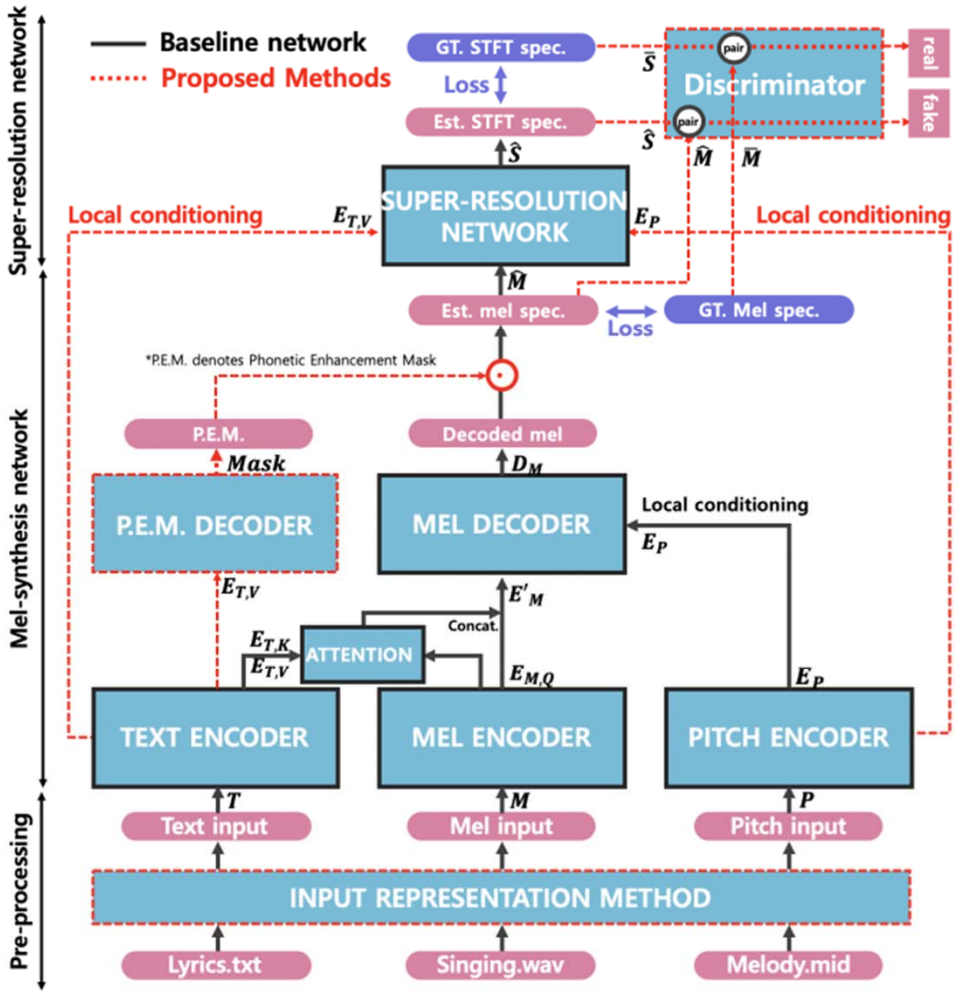
- Singing voice processing: Historical review
- Neural text-to-speech synthesis
- **Neural singing voice synthesis**
- Build your SVS model
- Song generation

SNU'2019 Model

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



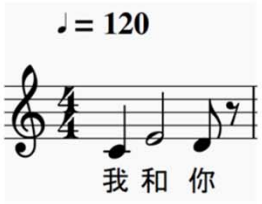
- **DB**: 60 songs from 1 female singer (~2hrs), manual annotation

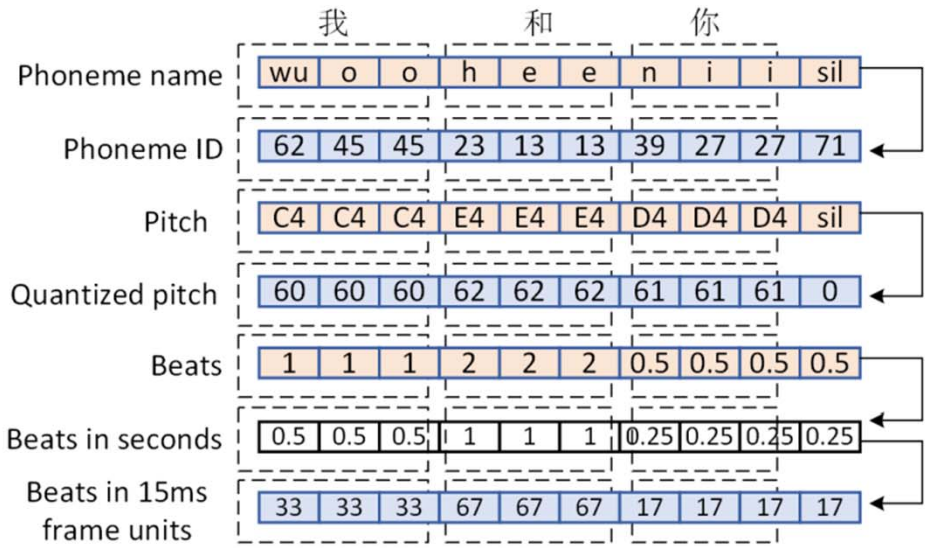


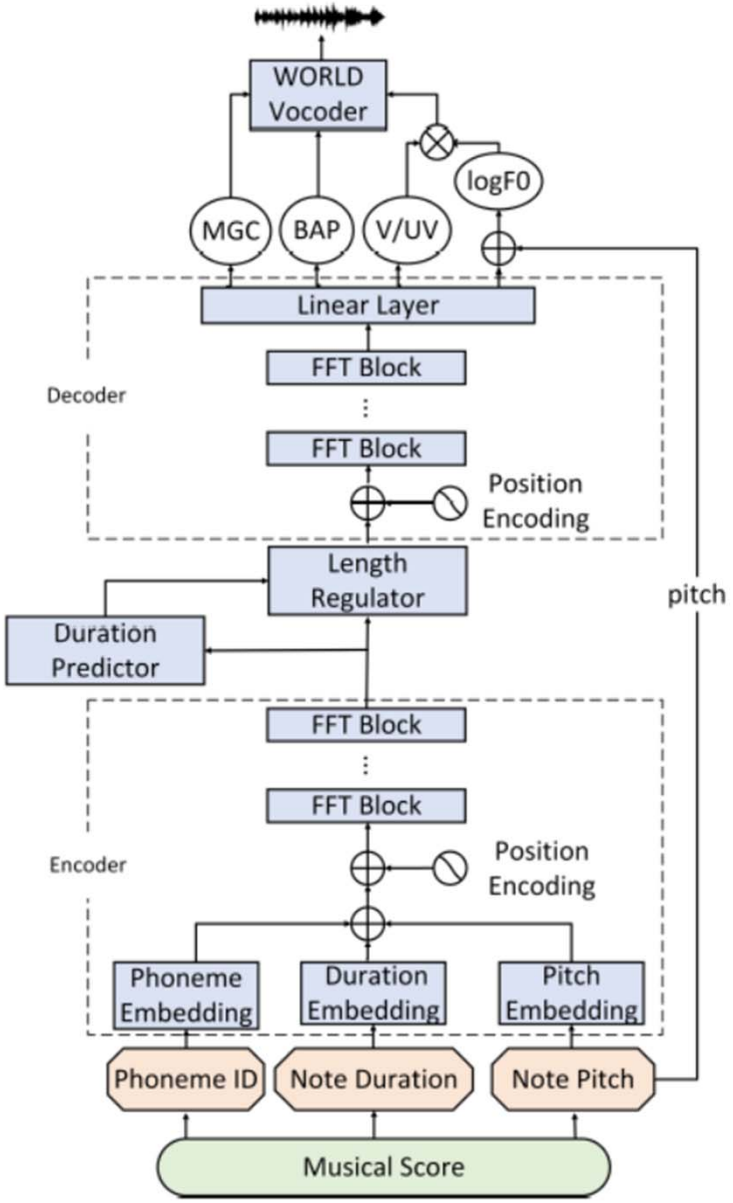
Ref: Lee et al, "Adversarially trained end-to-end Korean singing voice synthesis system," INTERSPEECH 2019

XiaoiceSing

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





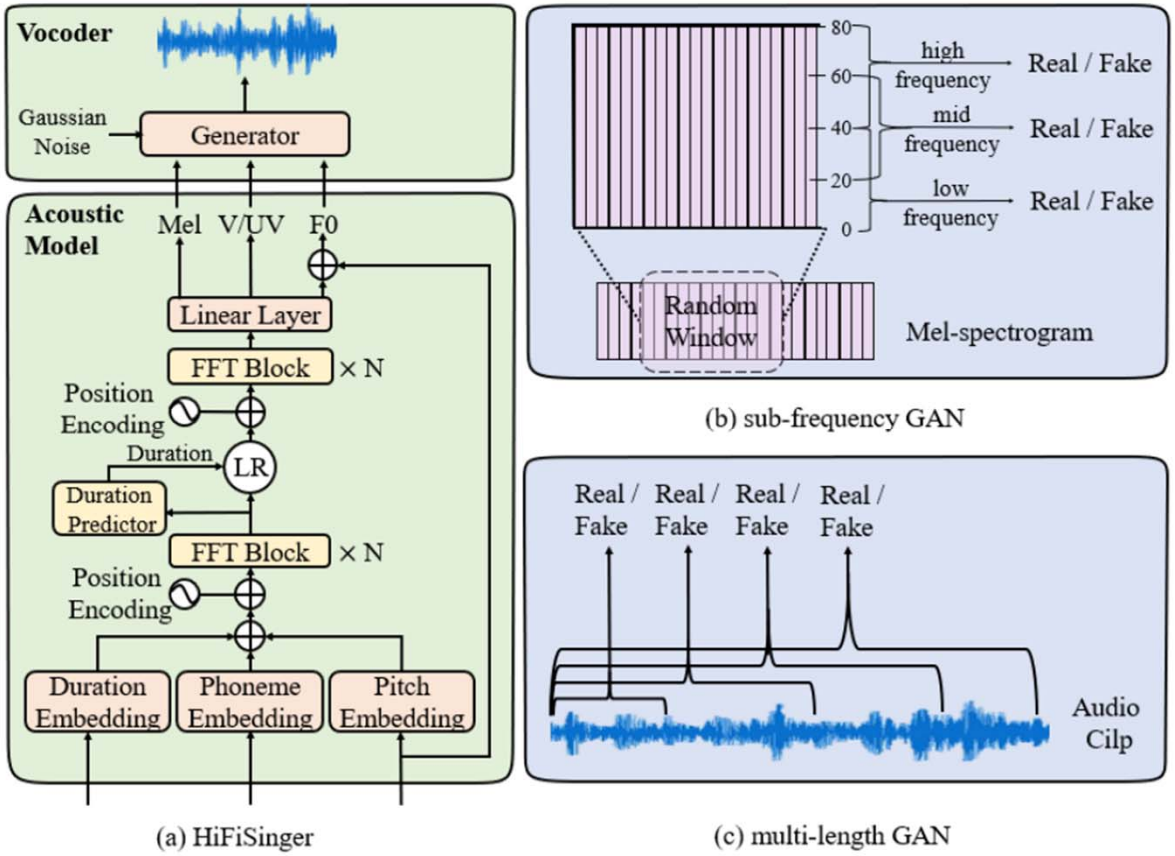


Ref: Lu et al, "XiaoiceSing: A high-quality and integrated singing voice synthesis system," INTERSPEECH 2020

HiFiSinger

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

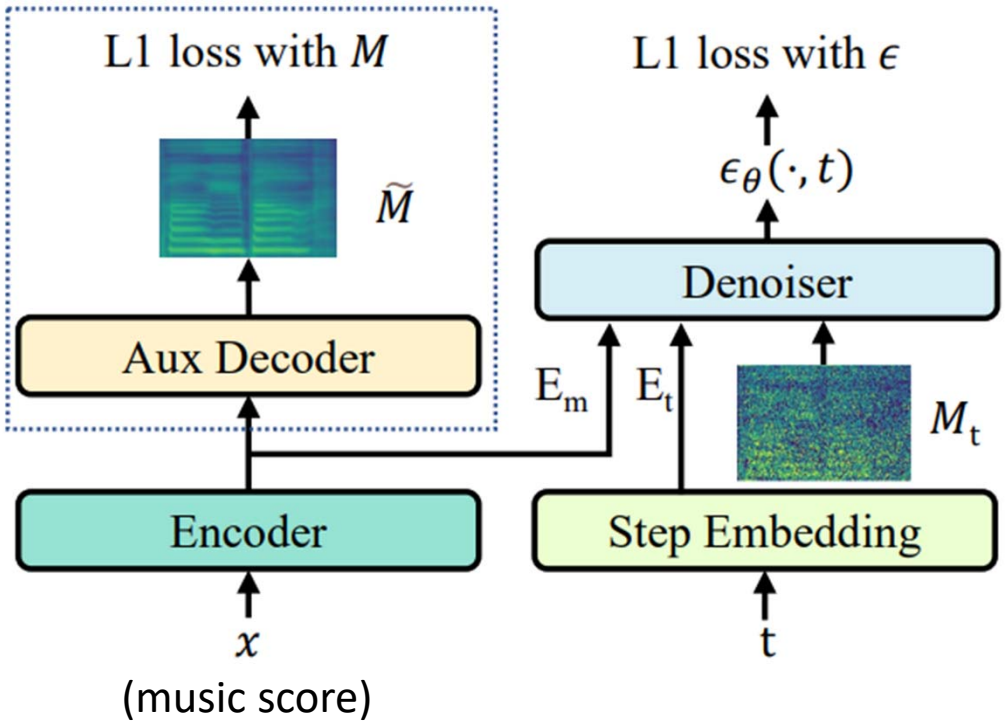
- Use Parallel WaveGAN based neural vocoder
- Use sub-frequency GAN to improve Mel-spectrogram generation
- Use multi-length GAN to improve waveform generation
- **DB**: 11 hrs songs from a female singer, manual annotation

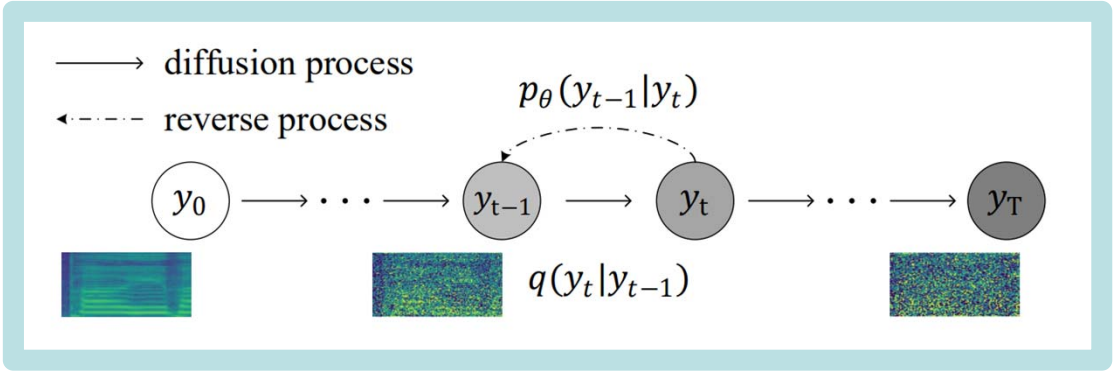


Ref: Chen et al, "HiFiSinger: Towards high-fidelity neural singing voice synthesis," arXiv 2020

DiffSinger

https://jisang93.github.io/hidden singer-demo/





- **Non-causal WaveNet-based denoiser** over Mel-spectrograms
- **FastSpeech-like encoder** to provide a "noisy starting point" (low-resolution Mel-spec) to speed up inference
  - pre-trained via knowledge distillation from a teacher like FastSpeech 2

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

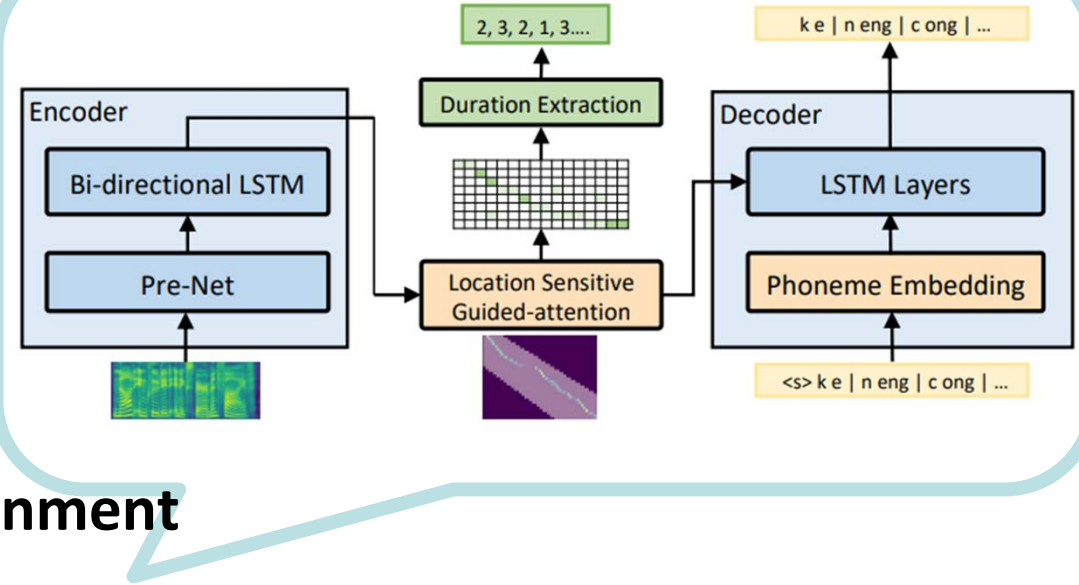
Progress of SVS Benefits from that of TTS

- 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

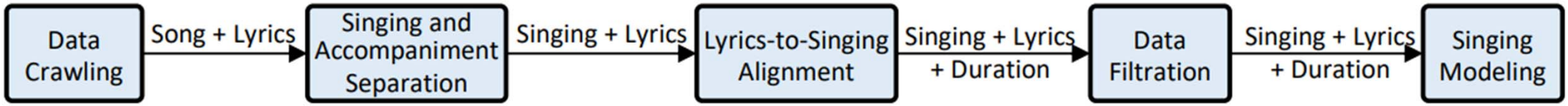
DeepSinger

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

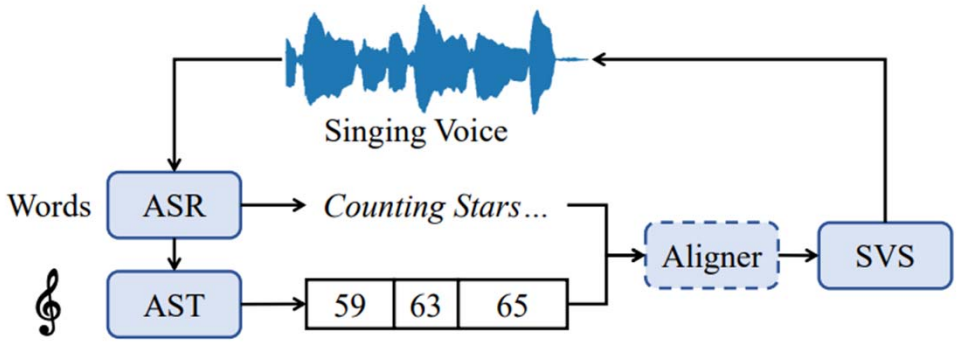


- Data **from the web** + **automatic alignment**



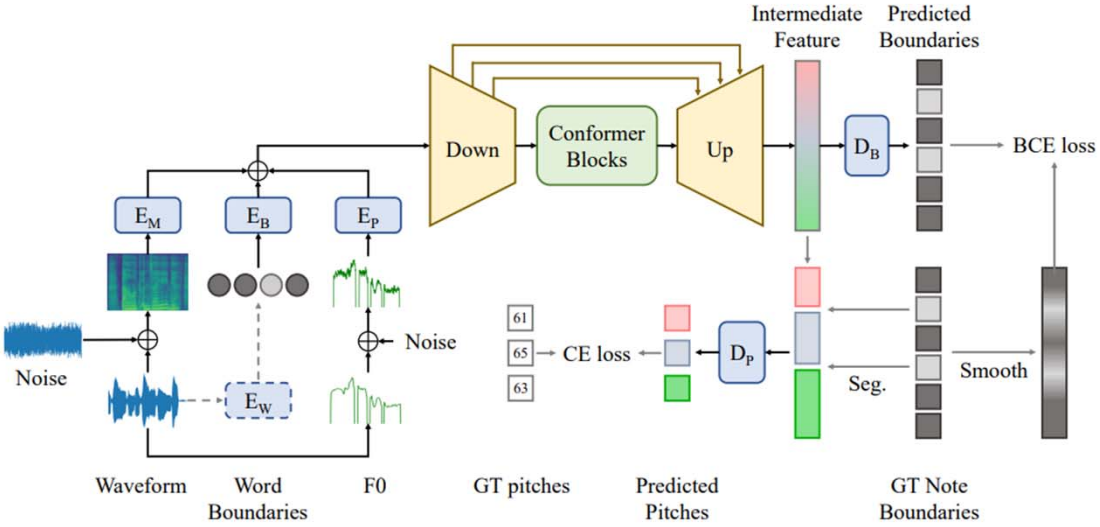
Ref: Ren et al, "DeepSinger: Singing voice synthesis with data mined from the web," KDD 2020

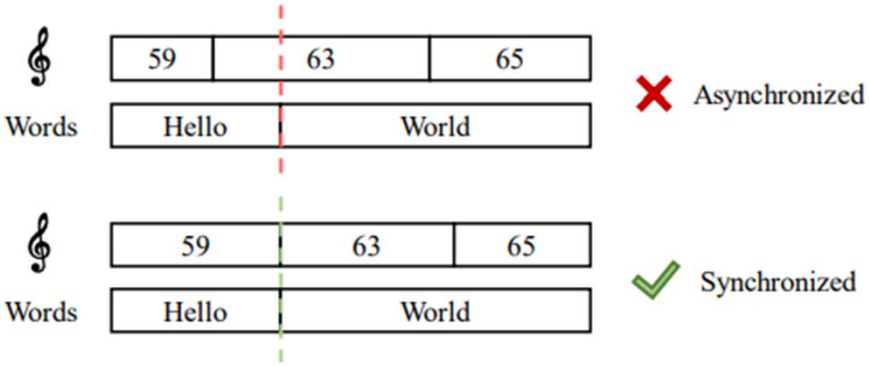
Automatic Alignment



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(large)	45.2	3.36 ± 0.12	3.45 ± 0.09

-P: pitch correction; -Q: overall quality

Ref: Li et al, "Robust singing voice transcription serves synthesis," ACL 2024

Singing Voice Conversion

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

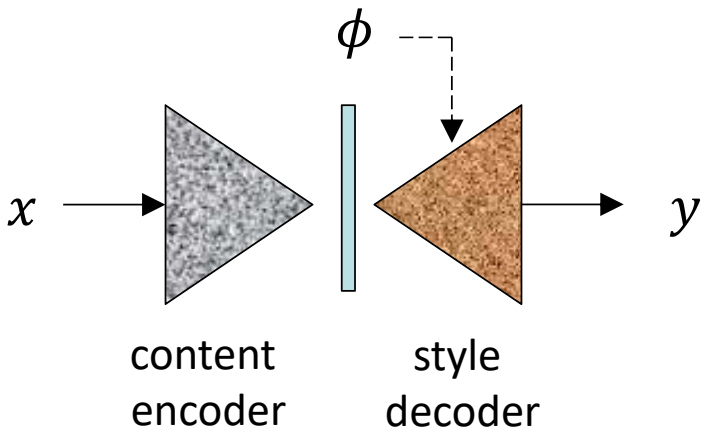


Table 3: Details of participating systems in SVCC 2023.

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

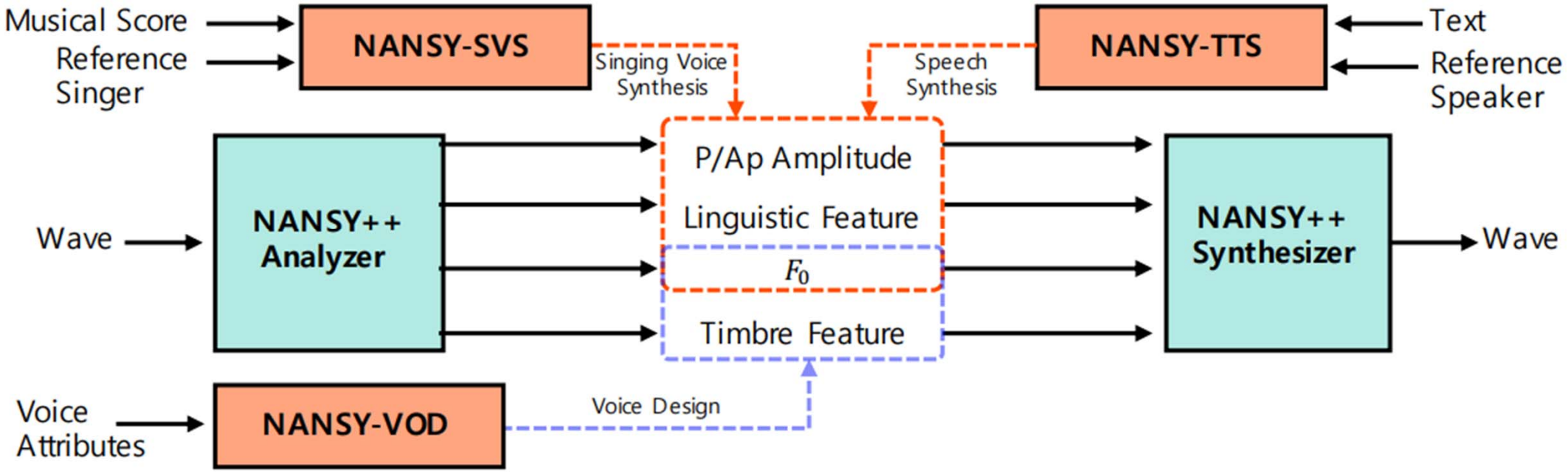
(**PPG**: phonetic posteriorgrams)

Ref: Huang et al, "The singing voice conversion challenge 2023," ASRU 2023

Nansy++

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

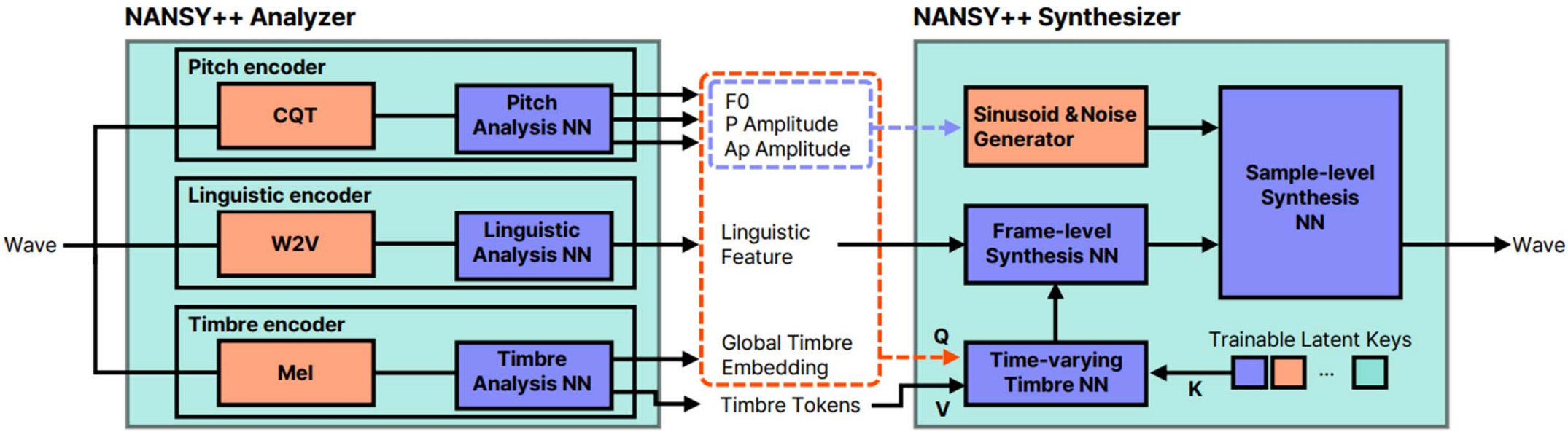
- A single model that do 1) VC/SVC, 2) TTS, 3) SVS, and 4) voice designing



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

Nansy++

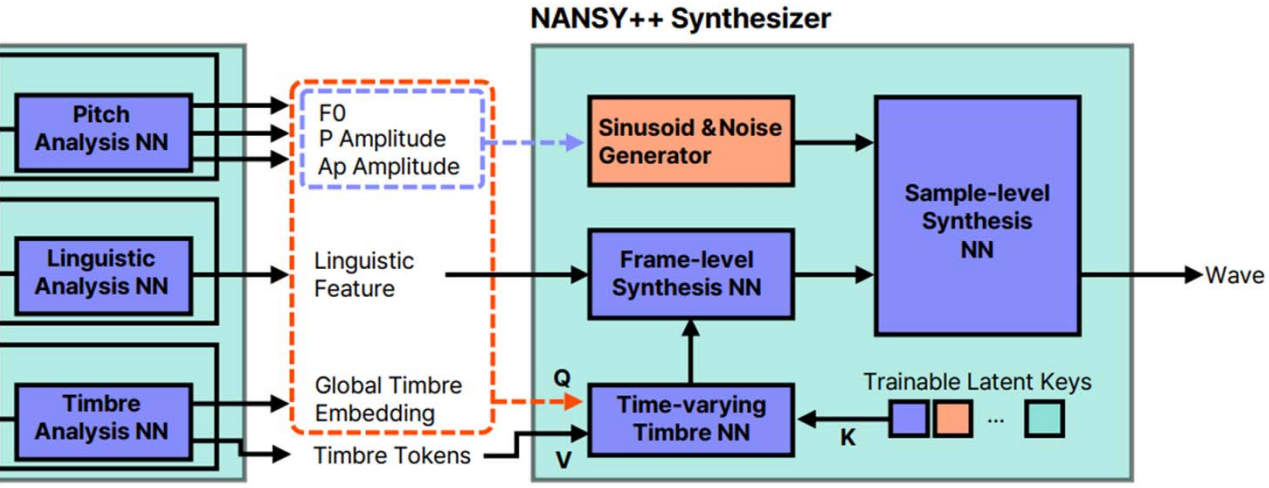
- “We trained the backbone model on 10,571 hours (speech: 10,092 hours, singing: 479 hours) of proprietary 44.1 kHz audio recordings composed of 6,176 speakers and 624 singers”

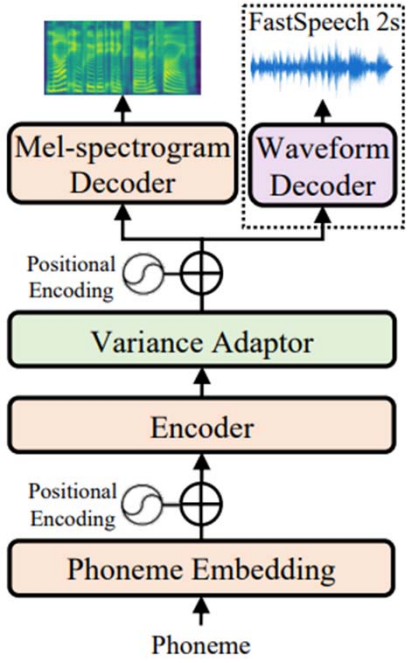


Ref: Choi et al, “NANSY++: Unified voice synthesis with neural analysis and synthesis,” ICLR 2023

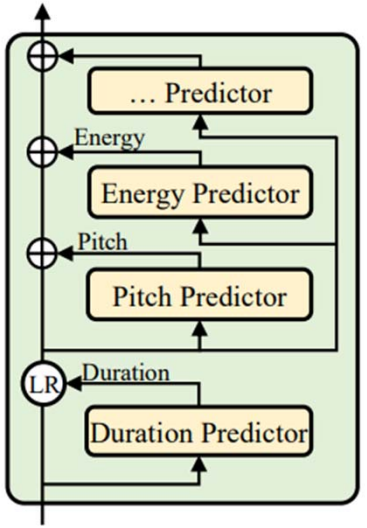
Nansy++ vs FastSpeech 2

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





(a) FastSpeech 2



(b) Variance adaptor

Recap: SVS

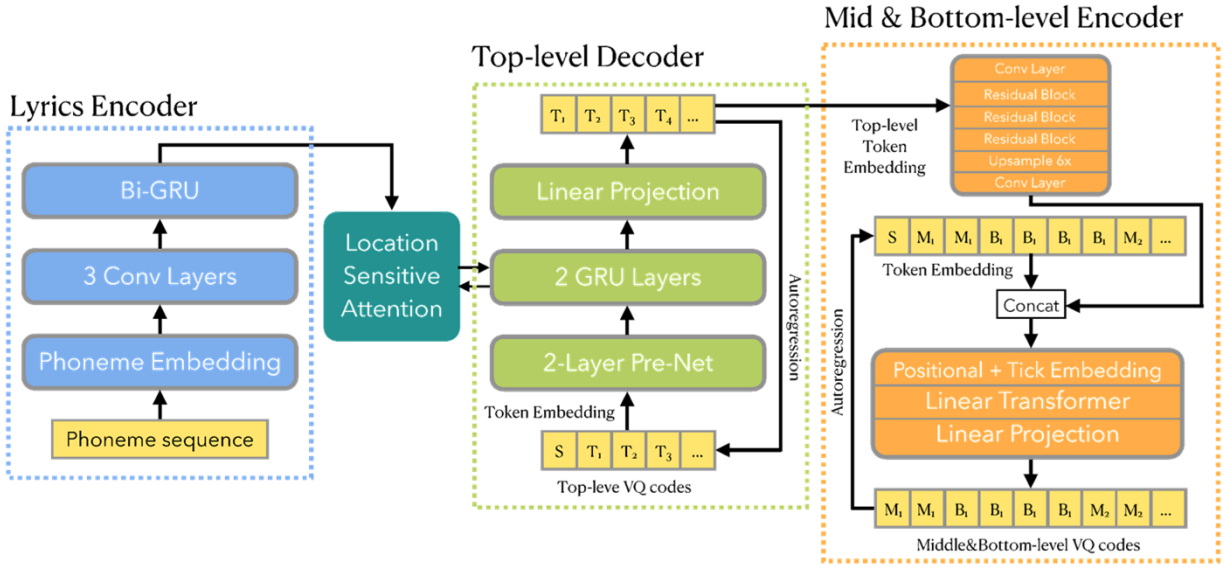
- Many SVS papers do not open source code…
  - For example: HiFiSinger, Nansy++

- One 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

KaraSinger (from our lab)

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



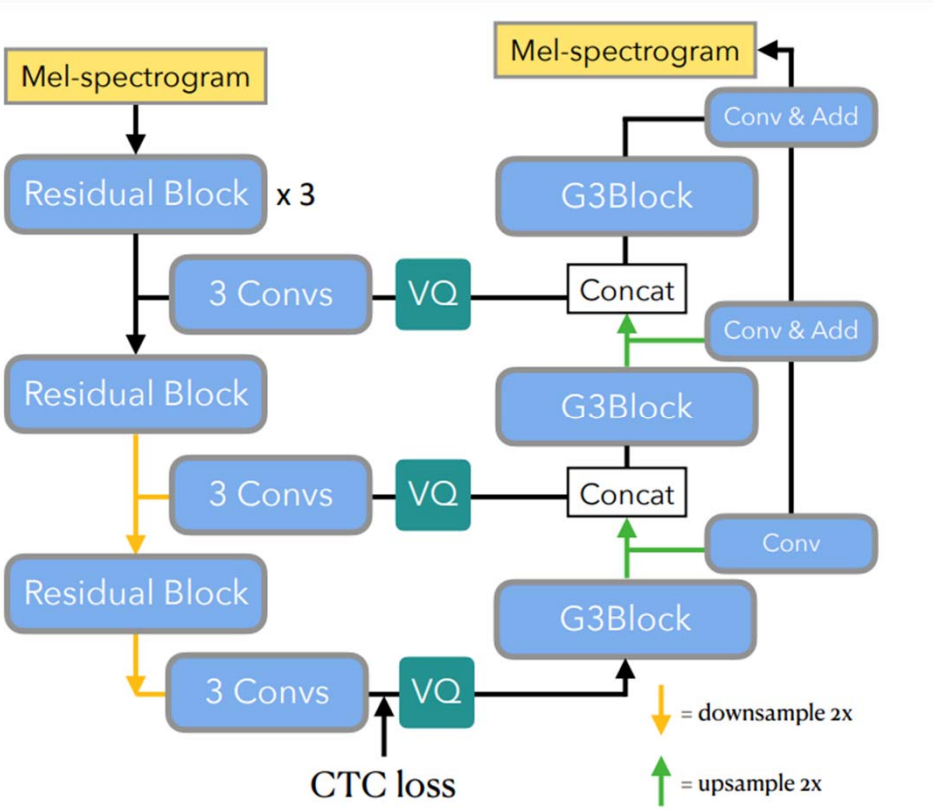
Ref 1: Liao et al, "KaraSinger: Score-free singing voice synthesis with VQ-VAE using Mel-spectrograms," ICASSP 2022

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

KaraSinger: VQVAE on Mel-spectrograms

https://jerrygood0703.github.io/KaraSinger/

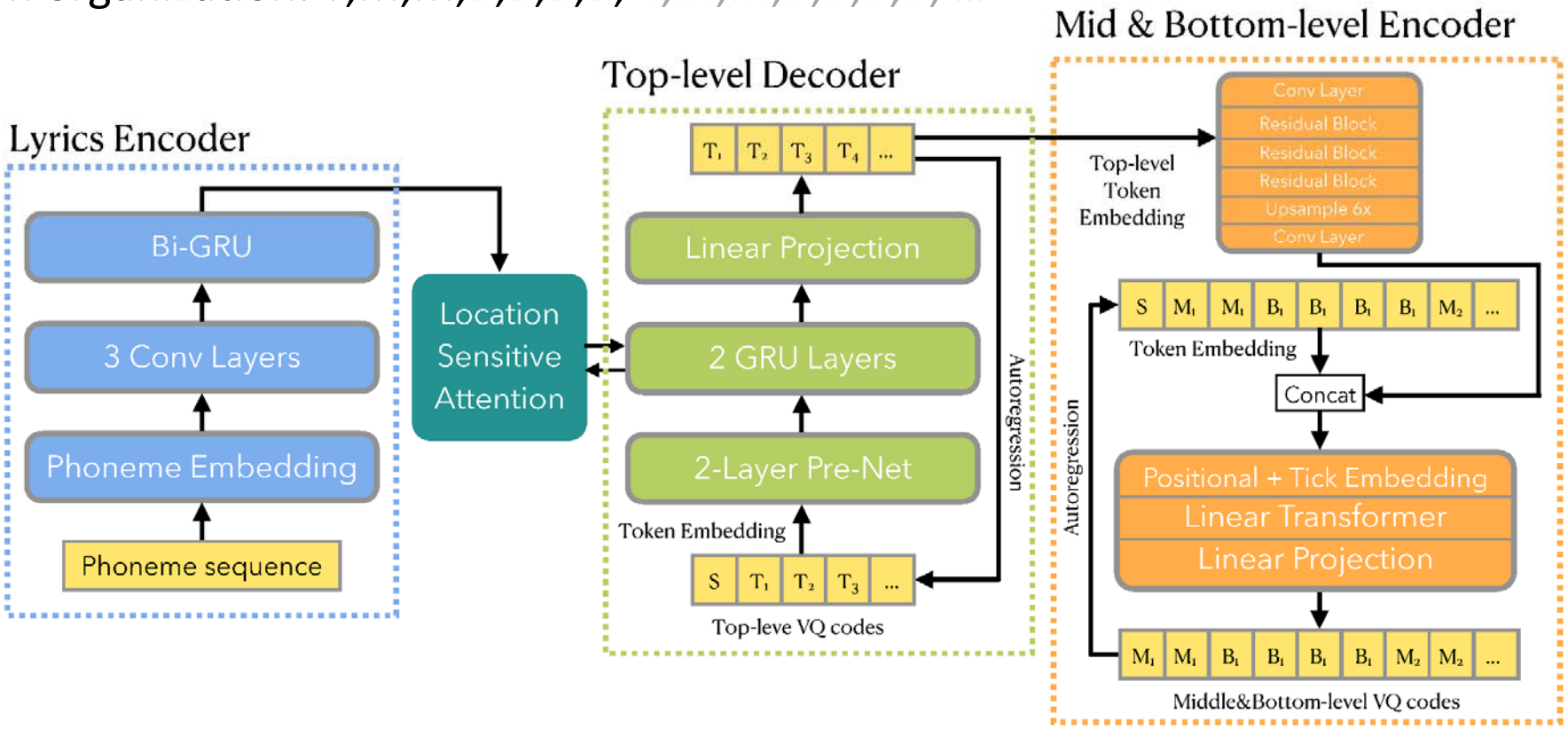
- Turn audio into **tokens**
  - Multi-scale hierarchical **VQ-VAE** with **top, mid, bottom** layers
  - Encoder: five residual blocks
  - Decoder: G3blocks (GRU + dilated convolution + group normalization)
  - MelGAN vocoder
- 4 days training on a single NVIDIA RTX 2080Ti, 1:1 RTF for inference



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

KaraSinger: VQVAE on Mel-spectrograms

- **LM over the tokens**
  - GRU for top codes (T), Transformer for **joint prediction** of mid (M) & bottom (B)
  - Token organization: T,M,M,B,B,B,B, T,M,M,B,B,B,B, …



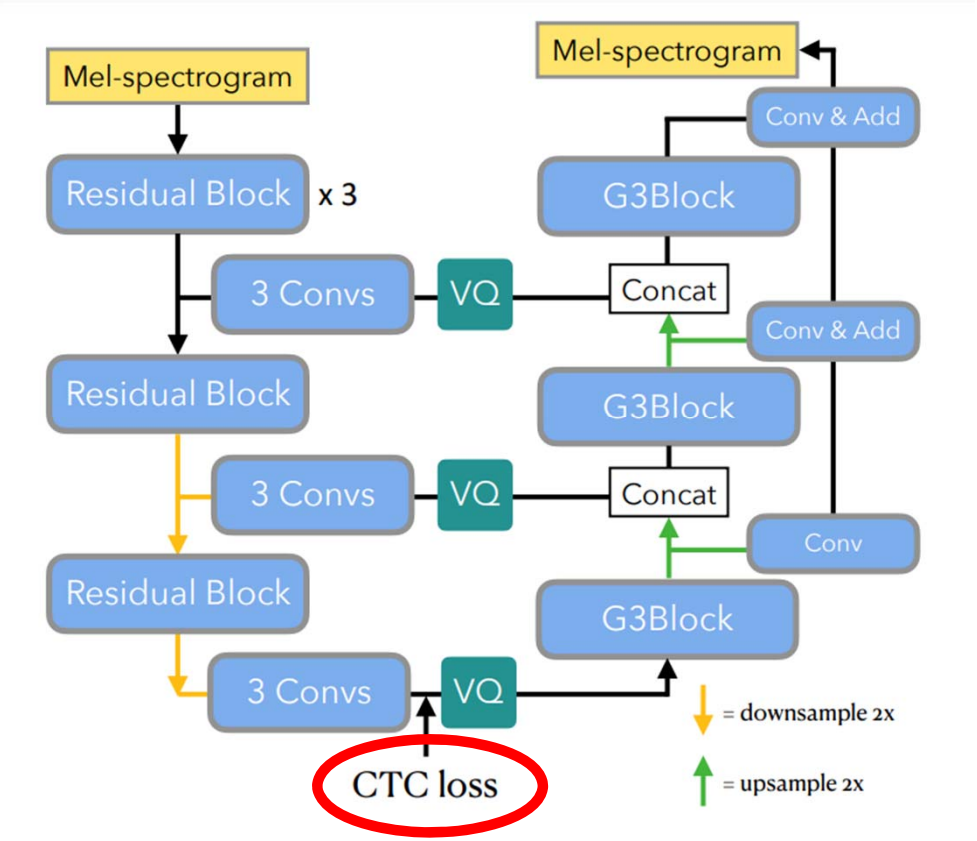
KaraSinger: VQVAE on Mel-spectrograms

https://jerrygood0703.github.io/KaraSinger/

- Connectionist temporal classification (**CTC**) loss on the top-level code to carry phoneme-related information
  - Without it the seq2seq model struggles to learn meaningful alignments in cross-attention to attend to the input lyrics

Model	Intelligibility	Musicality	Overall
noCTC	1.30 ± 0.14	2.00 ± 0.16	1.64 ± 0.14
3-level	3.30 ± 0.18	3.43 ± 0.16	3.19 ± 0.17
Proposed	4.23 ± 0.14	3.85 ± 0.14	3.67 ± 0.15

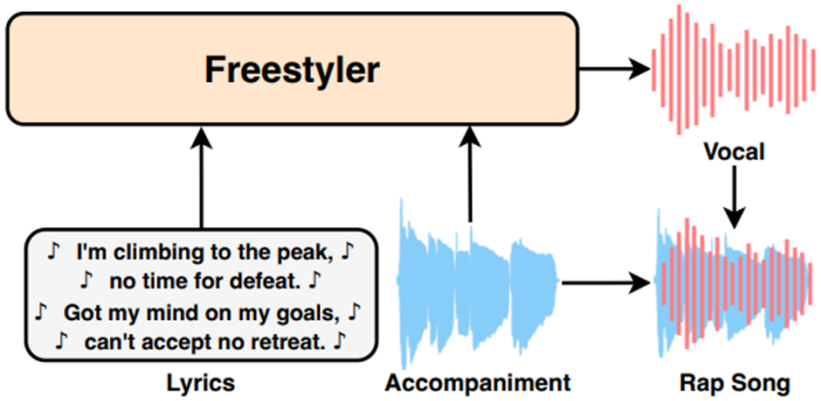
*3-level: no joint prediction of M and B



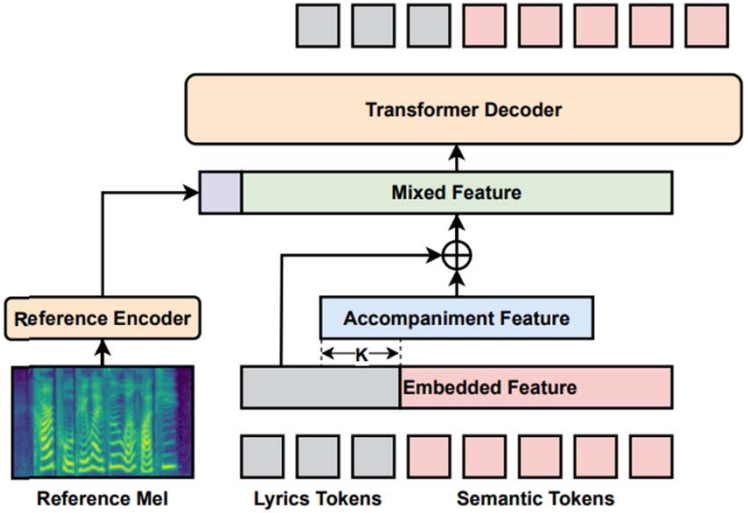
Ref: Liao et al, "KaraSinger: Score-free singing voice synthesis with VQ-VAE using Mel-spectrograms," ICASSP 2022

Freestyler

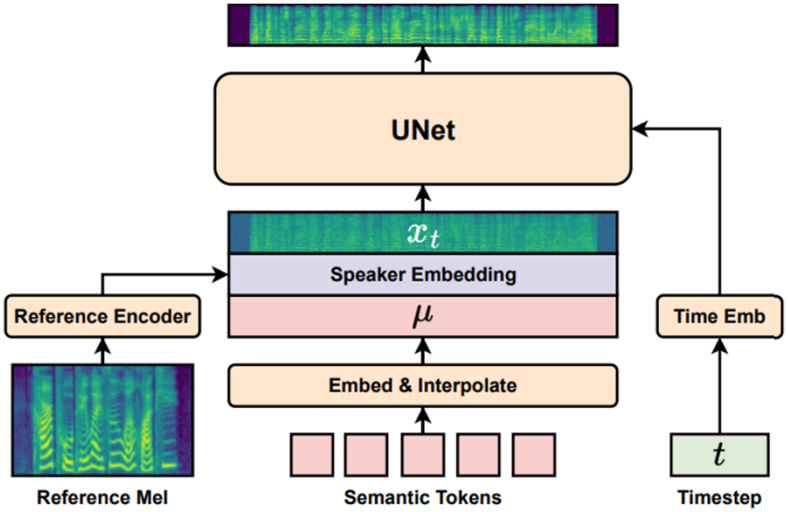
https://nzqian.github.io/Freestyler/



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



(a) Lyrics-to-Semantic



(b) Semantic-to-Spectrogram

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**
- Song generation

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/

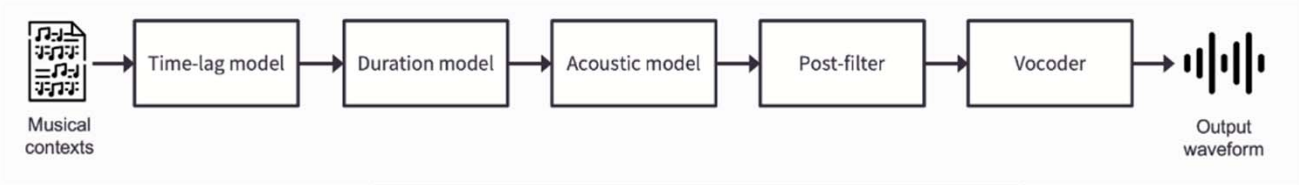
Dataset	Language	Style	#Hour	#Singer	Quality	Musicality	Score	Align-ment	Style Transfer
Opencpop [41]	Chinese	Pop	5.25	1	Studio	Ama.	Perform. MIDI	✓	✗
M4Singer [42]	Chinese	Pop	29.77	20	Studio	50% Ama. 50% Prof.	Perform. MIDI	✓	✗
Children Song [43]	Korean English	Children	4.86	1	Studio	Prof. but plain	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

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

Library for SVC: so-vits-svc

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

SoftVC VITS Singing Voice Conversion

AI

Adobe Photoshop Generative Expand and Generative Fill	OpenAI GPT-4	Runway Gen-2
Alitheon FeaturePrint	Dedrone City-Wide Drone Detection	Meta SeamlessM4T
So-VITS-SVC	AlertCalifornia and Cal Fire AI Wildfire Detector	Stable Audio
TrailGuard AI	OpenAI Dall-E 3	Project Gutenberg Open Audiobook Collection
AudioShake	Humane Ai Pin	

"The Best Inventions of 2023," Time Magazine
https://time.com/collection/best-inventions-2023/

Outline

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

Song Generation: Intro

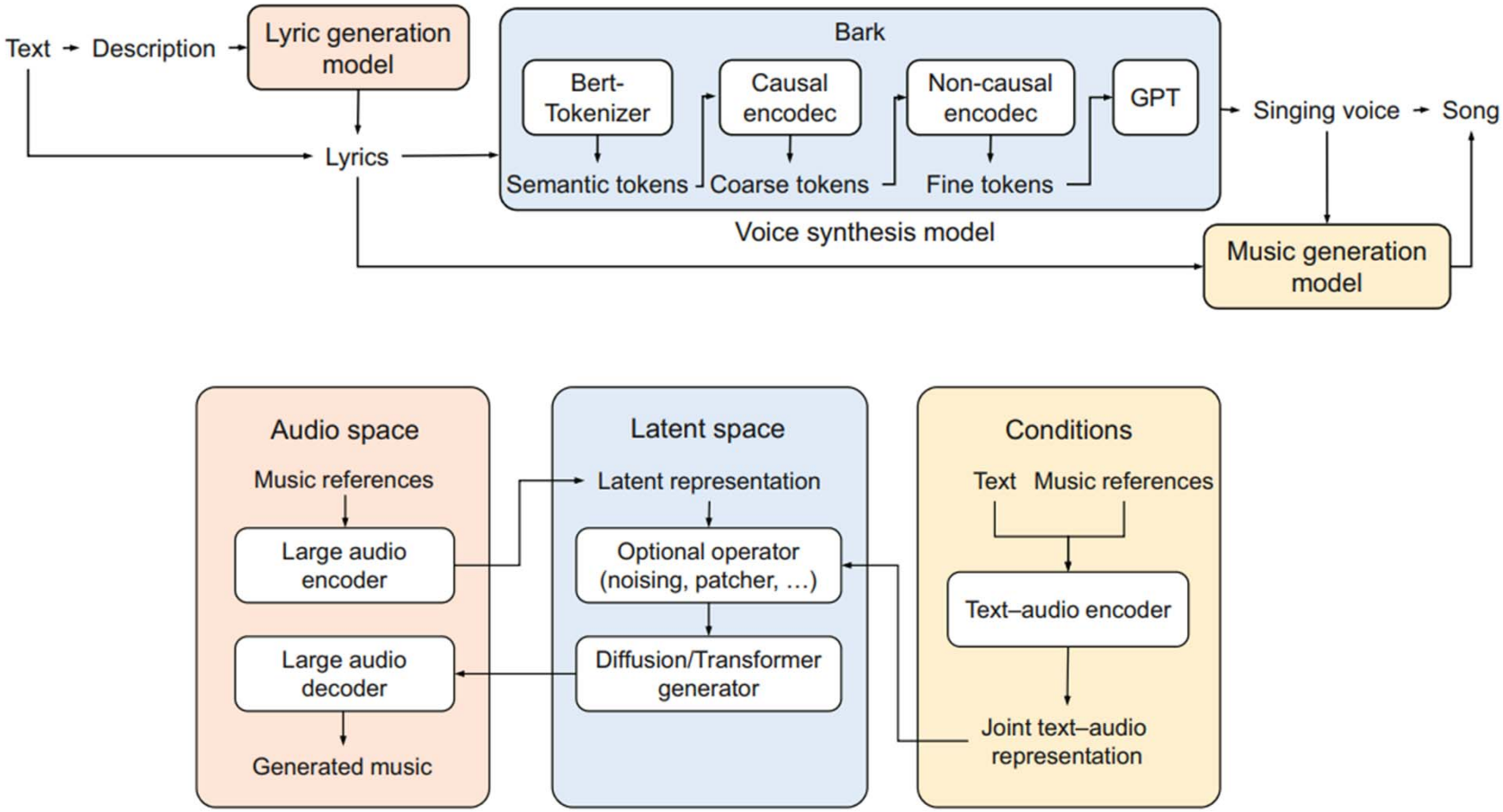
- Generate both **vocal** and the **instrumental** accompaniment
- Usually requires only **lyrics** and a style prompt (e.g., text) during inference
- Usually aim to generate music with **full-song length**
  - In contrast, SVS models usually generate one sentence at a time

Suno

https://github.com/suno-ai/bark

- Suno operates as a proprietary system
- But, it released **bark**, an open-source **transformer**-based text-to-audio model released in 2023/04
  - Bark can generate highly realistic, multilingual speech as well as other audio - including music, background noise and simple sound effects
- It follows a GPT style architecture similar to **AudioLM** and **Vall-E** and a quantized audio representation from **EnCodec**
  - The output limited to ~13-14 seconds
  - The full version of Bark requires around 12Gb of memory to run
  - Bark supports 100+ speaker presets across supported languages

Suno



Ref: Yu et al, “Suno: Potential, prospects, and trends,” Frontiers of Information Technology & Electronic Engineering, 2024

Open Source Models for Song Generation

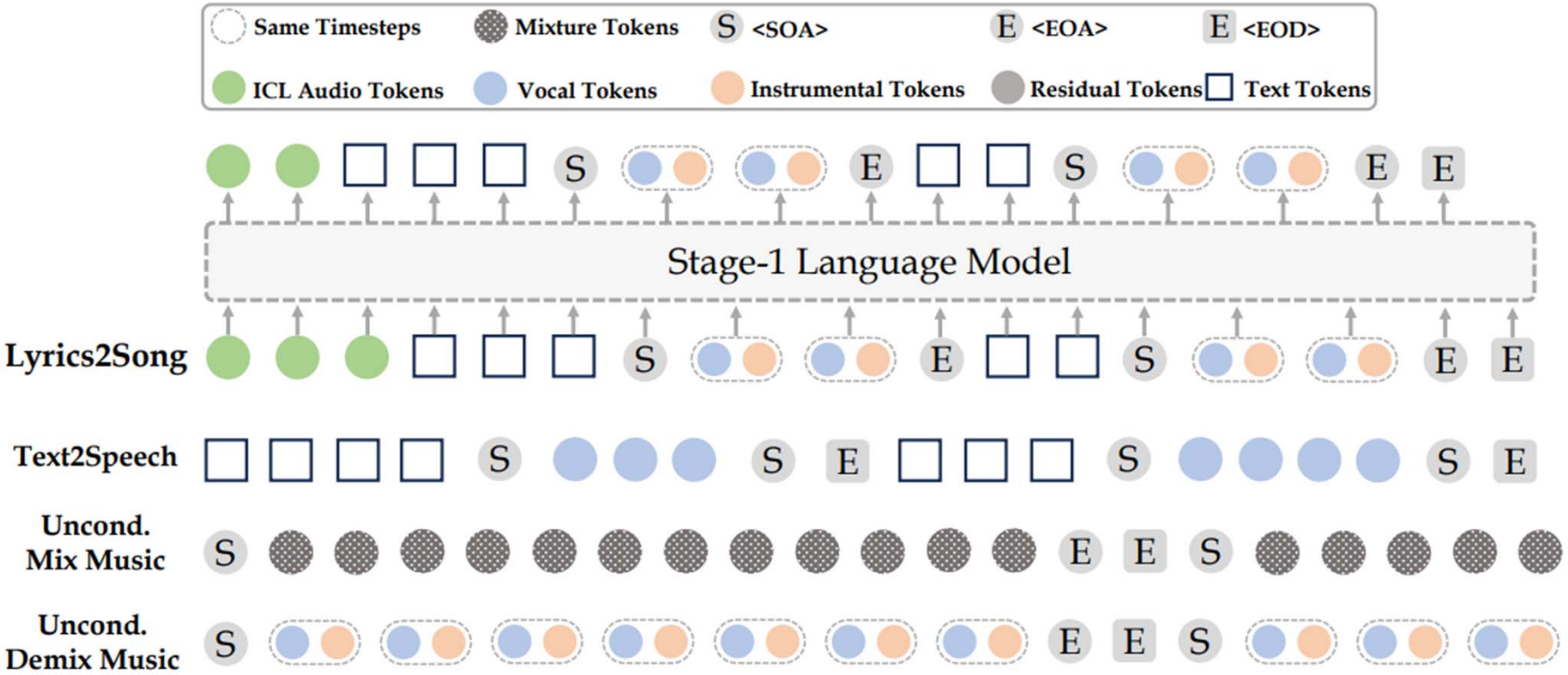
- Quite a few since 2025

System	Training data (hrs)	Compute resources
Yue (Yuan et al., 2025)	650K	16× H800
DiffRhythm (Ning et al., 2025)	60K	8× Huawei Ascend 910B
Acestep (Gong et al., 2025)	100K	120× A100
SongBloom (Yang et al., 2025)	100K	16× A100
JAM (Liu et al., 2025a)	54K	8× H100
Levo (Lei et al., 2025)	110K	8× A100
ComposerFlow	**6K**	**1 × 3090**

Ref: “ComposerFlow: Step-by-step compositional song generation,” 2025

YuE

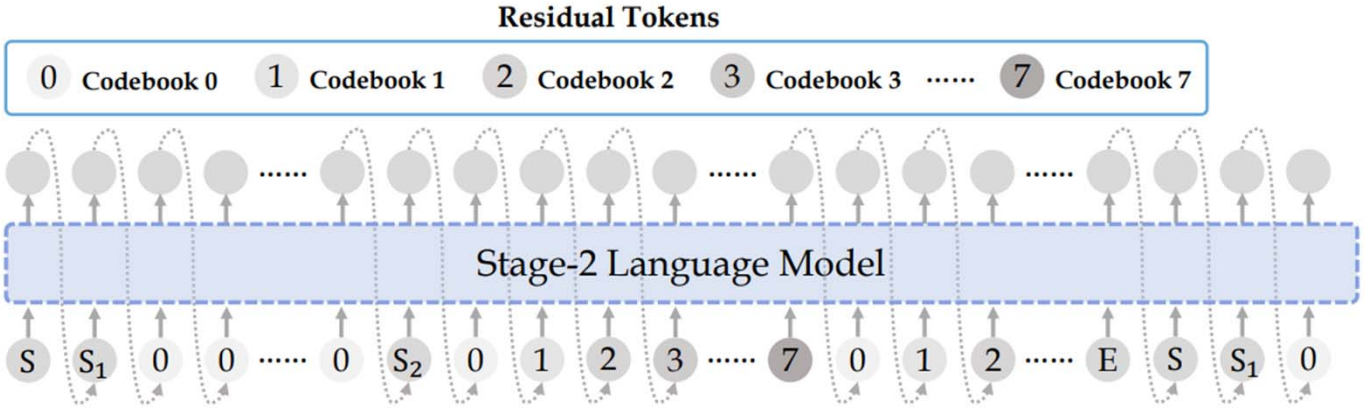
- **Dual next token prediction**: vocal & instrumental tokens

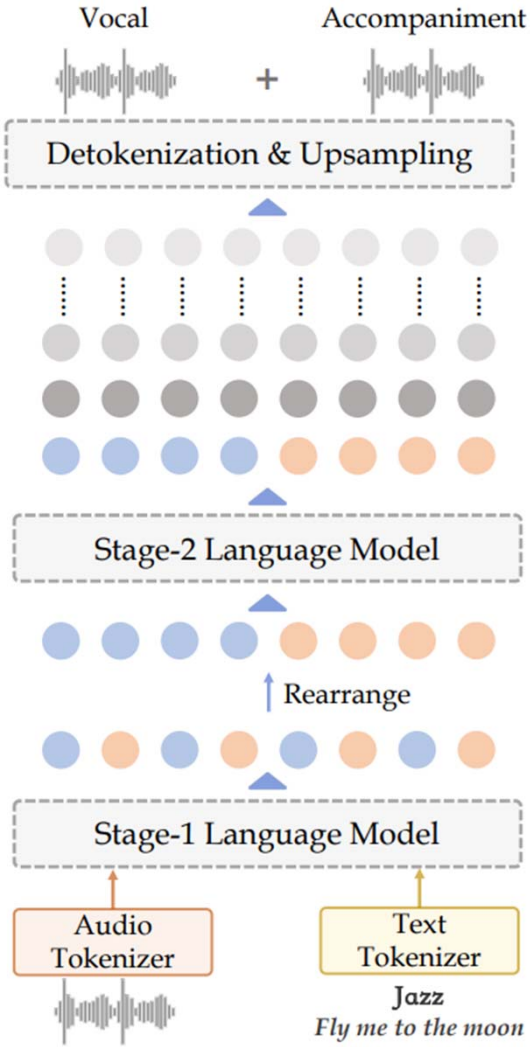


Ref: Yuan et al, "YuE: Scaling open foundation models for long-form music generation," 2025

YuE

- Two-stage LMs: coarse tokens first, then others
  - Both stages adopt the **LLaMA2** architecture
  - Use **X-codec** to facilitate lyrics conditioning
  - The training does not converge when using pure acoustic tokenizers such Encodec32k and HiFiCodec





Ref: Yuan et al, "YuE: Scaling open foundation models for long-form music generation," 2025

YuE

https://map-yue.github.io/

- Structural progressive conditioning for long-context lyrical alignment
- A multitask, multiphase pre-training recipe to converge and generalize
  - TTS
  - TTM
    - The majority of our dataset consists of unconditional music
    - We annotate all tracks using Qwen2-Audio
    - 40% tracks are separated into vocal-instrumental dual-track format using UVR12
  - Lyrics-to-song
    - we implement heuristic filtering to remove irrelevant content and exclude overly short lyrics (less than 10 sentences), retaining only approximately 10% of matched tracks

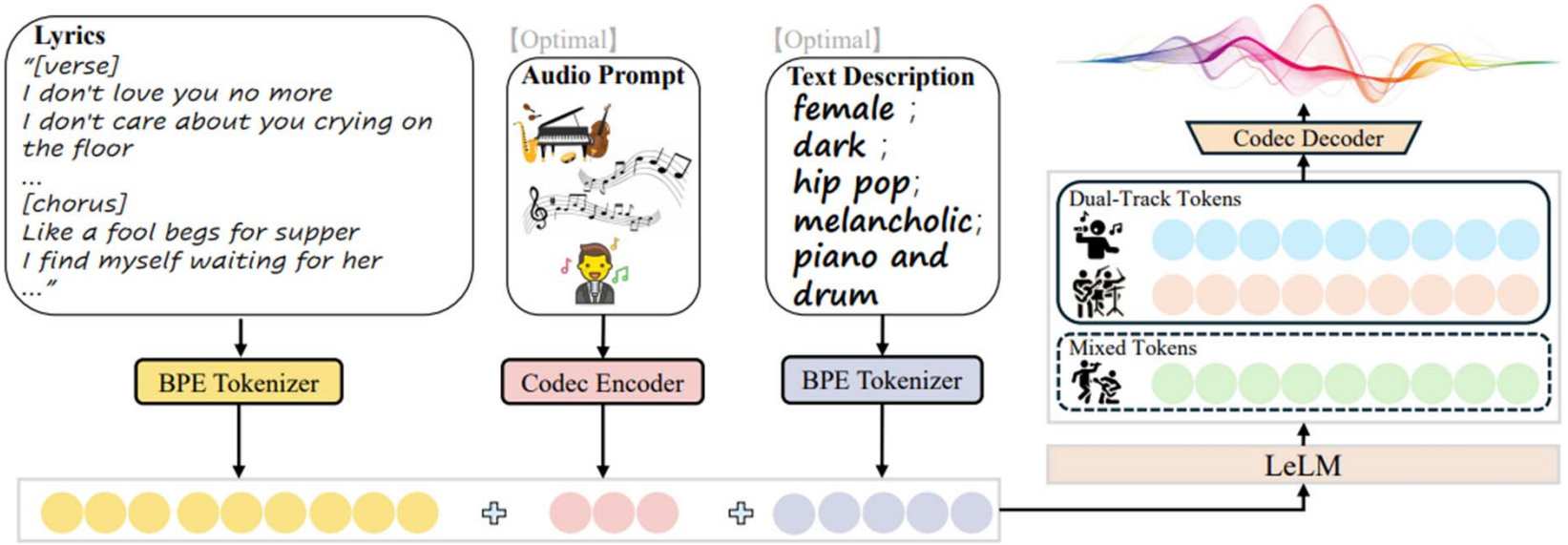
Table 2. Decomposition of Lyrics-to-Song Generation Capabilities

Essential Capabilities
1) Modeling of Human Vocal
2) Modeling of Instrumental
3) Joint Modeling of Vocal and Instrumental
4) Aligning Cross-Modal/Same-Modal Controls (lyrics, style, structure, in-context learning)

Levo

https://levo-demo.github.io/

- Mixed tokens and dual-track tokens
- DPO (direct preference optimization)



Ref: Lei et al, "LeVo: High-quality song generation with multi-preference alignment," arXiv 2025

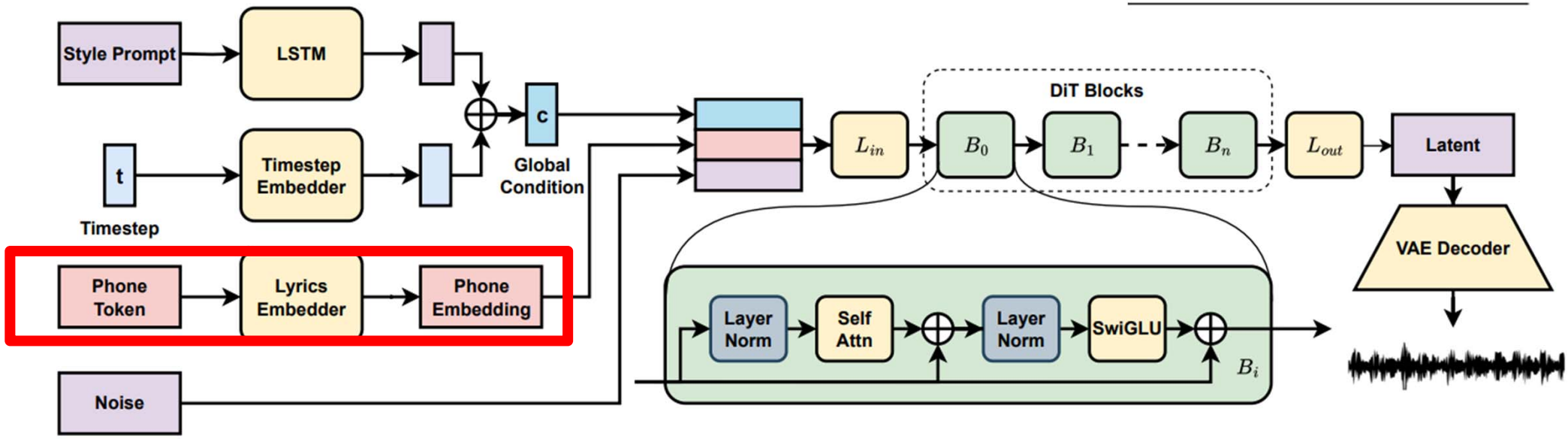
DiffRhythm

https://nzqian.github.io/DiffRhythm/

Method	RTF	Relative Speed
Yue	0.083x	1.00x
DiffRhythm	10.03x	120.84x

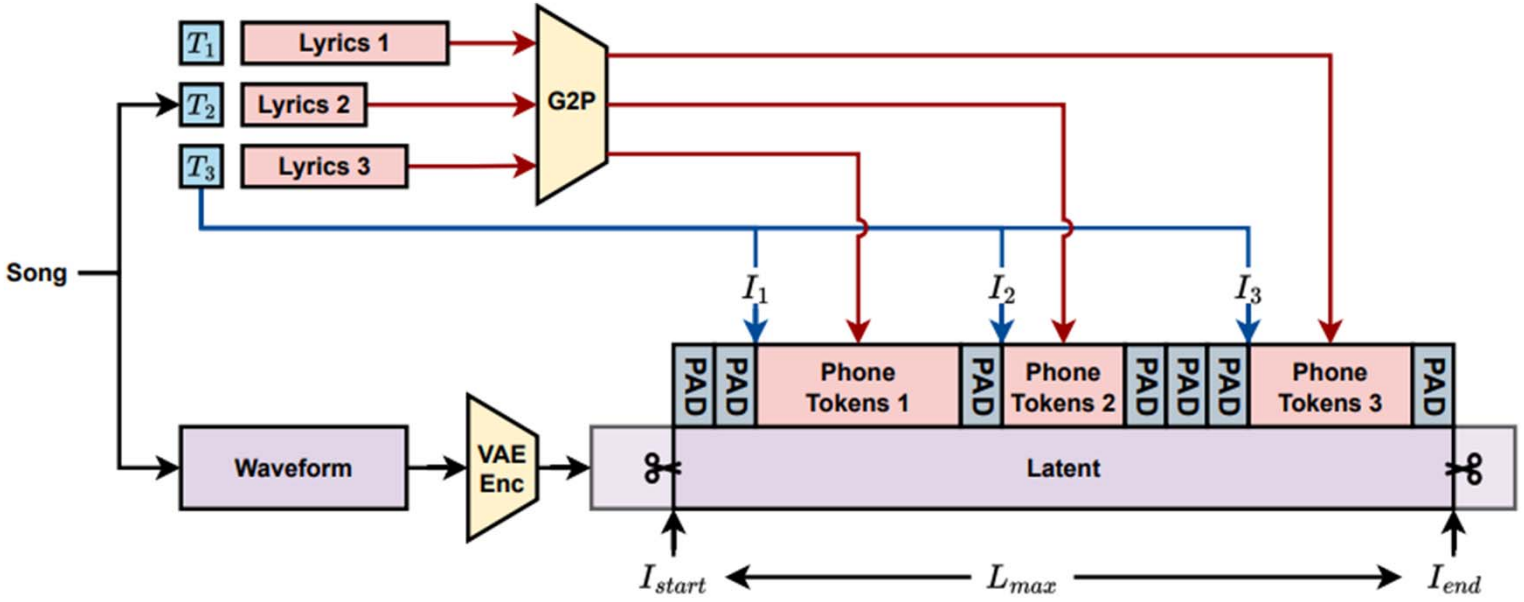
- Generate the mixture directly
- Faster: <0.04 RTF

Generation Length	RTF↓
285 s	0.034
95 s	0.037



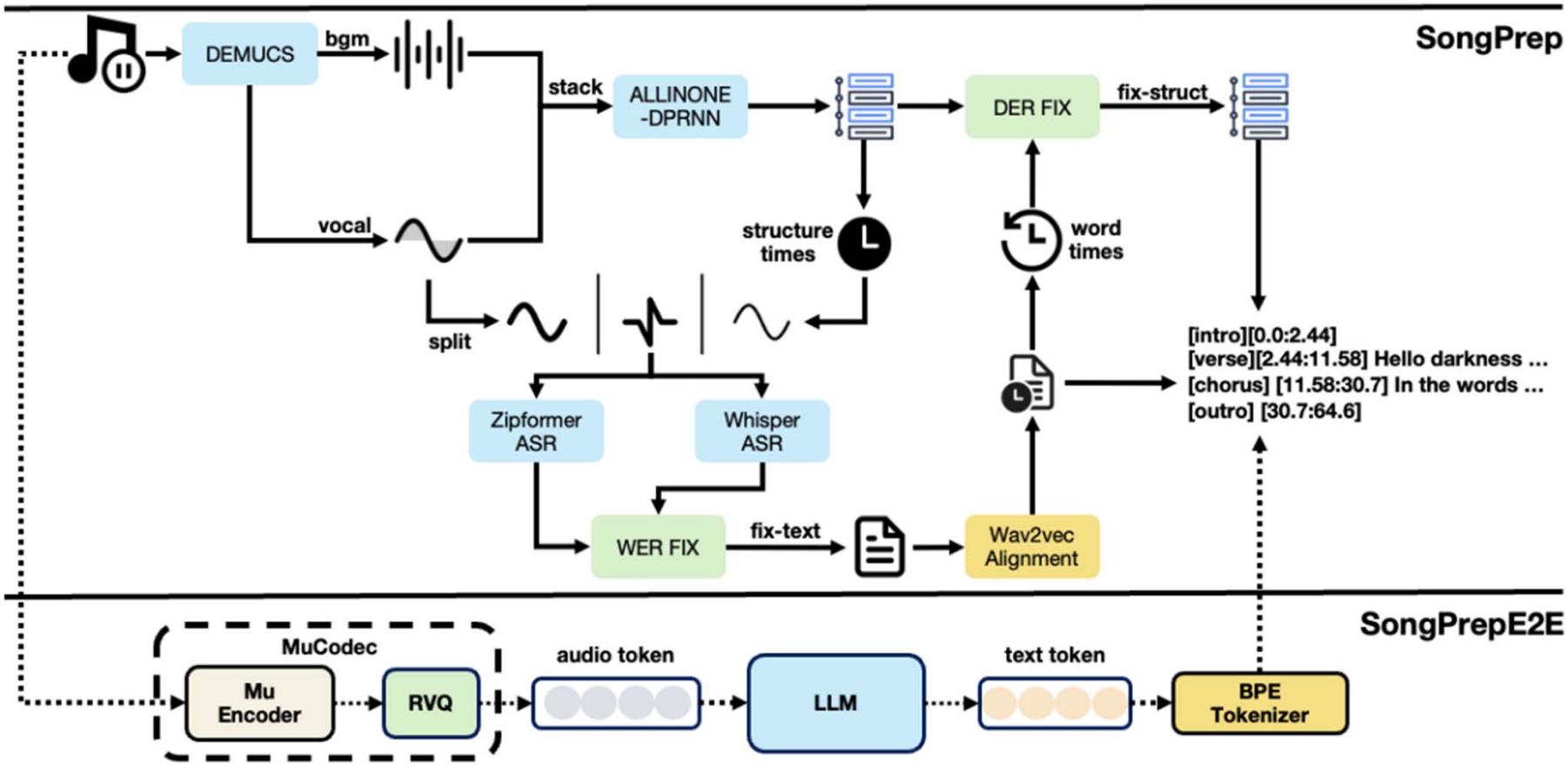
Ref: Ning et al, "DiffRhythm: Blazingly fast and embarrassingly simple end-to-end full-length song generation with latent diffusion," 2025

DiffRhythm



- "With conventional text conditioning approaches in diffusion-based TTS models like cross-attention mechanisms or direct feature concatenation, we failed to achieve intelligibility in song generation."
- Propose to reduce the difficulty of alignment with a sentence-level alignment paradigm that requires only sentence-start annotations

SongPrep

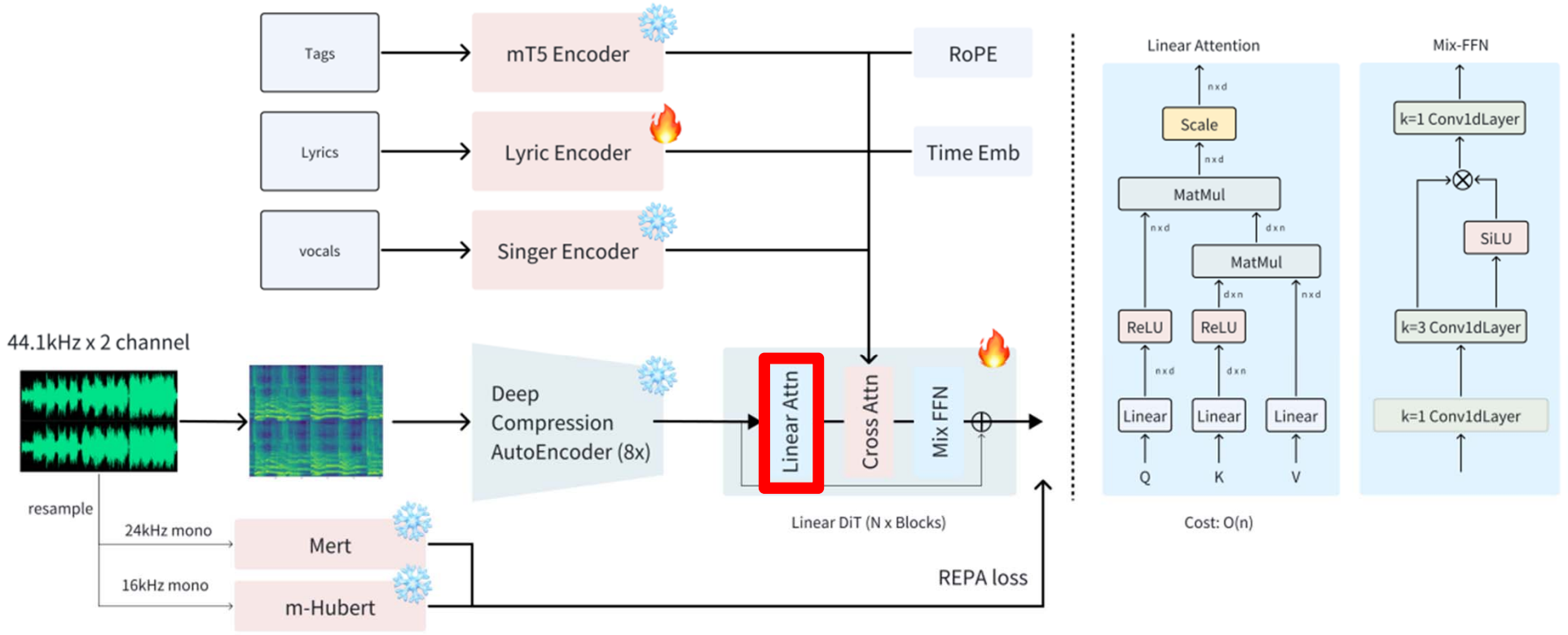


Ref: Tan et al, “SongPrep: A preprocessing framework and end-to-end model for full-song structure parsing and lyrics transcription,” arXiv 2025

ACE-Step

https://ace-step.github.io/

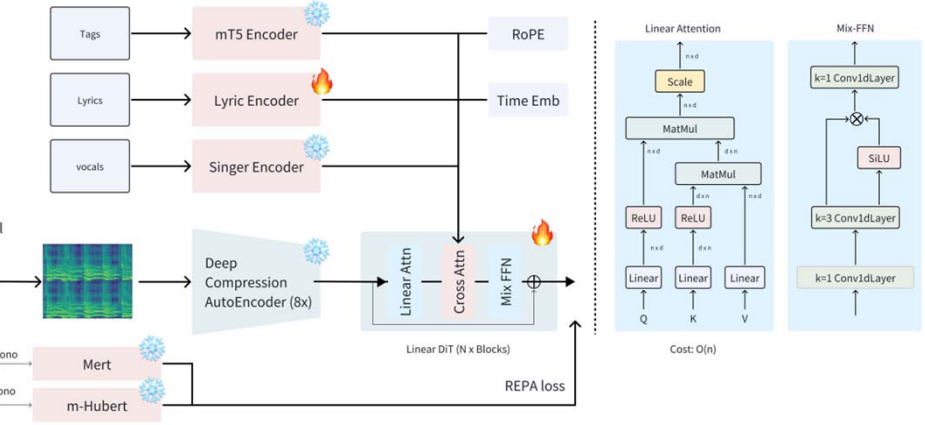
Method	RTF	Relative Speed
Yue	0.083x	1.00x
DiffRhythm	10.03x	120.84x
ACE-Step	15.63x	188.31x

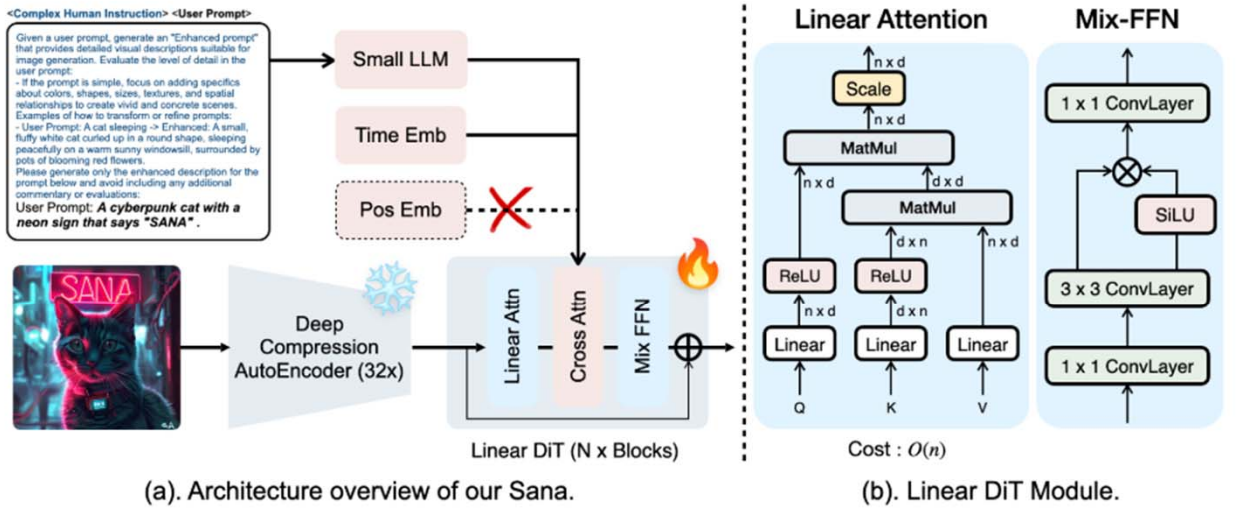


Ref: Gong et al, "ACE-Step: A step towards music generation foundation model," arXiv 2025

ACE-Step

- Adapt the **Linear DiT** structure from NV's text-to-image model SANA with two modifications
  - a single AdaLN layer's parameters are shared across all DiT blocks
  - Conv-1D instead of Conv-2D





Overview of Sana: Fig. (a) describes the high-level training pipeline, containing our 32× deep compression Autoencoder, Linear DiT, and complex human instruction. Note that Positional embedding is not required in our framework. Fig. (b) describes the detailed design of the Linear Attention and Mix-FFN in Linear DiT.

SANA: Efficient High-resolution image synthesis with linear diffusion transformers, arXiv 2024
https://github.com/NVlabs/Sana

Ref: Gong et al, "ACE-Step: A step towards music generation foundation model," arXiv 2025

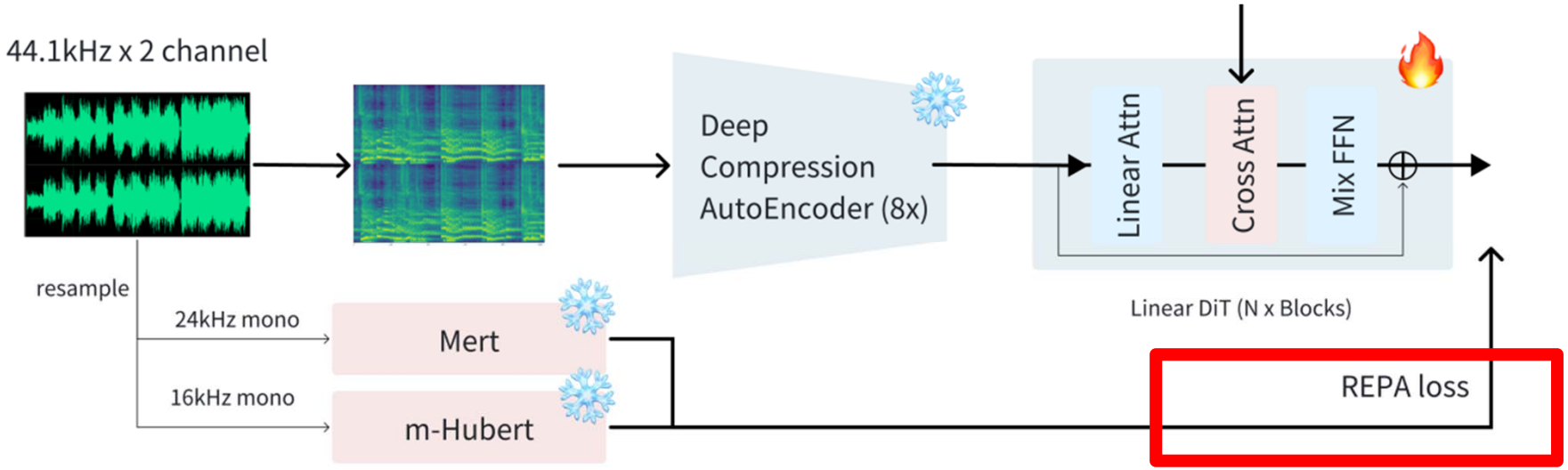
ACE-Step

- Data
  - 1.8 million unique musical pieces (roughly 100,000 hours)
  - Meta's Audiobox aesthetics toolkit for quality control
  - Qwen-omini for audio captioning
  - Whisper 3.0 for lyrics transcription
  - MIR tools for structural segmentation, key detection, tempo estimation
- GPU

Parameter	Specification
Computational Resources	15 nodes, each with 8 NVIDIA A100 GPUs (120 GPUs total).
Batch Size	Per-GPU: 1, Global: 120.
Steps	460,000 steps for pretrain; 240,000 steps for finetune.
Dataset	Full 100,000-hour music dataset for pretrain; Curated high-quality subset of approx. 20,000 hours for fintune
Total Training Time	Approx. 264 hours.
Optimizer Type	AdamW with 1e-2 weight decay and (0.8, 0.9) betas.
Gradient Clipping	Max norm of 0.5.
Learning Rate	1e-4, with linear warm-up over the first 4,000 steps.
Timestep Sampling	Logit-normal scheme: $u \sim \mathcal{N}(\text{mean} = 0.0, \text{std} = 1.0)$; shift=3.0.
Sequence Lengths (Max Context)	4096 tokens for lyrics, 256 tokens for text prompts, 2584 tokens for mel-spectropgram latents

ACE-Step

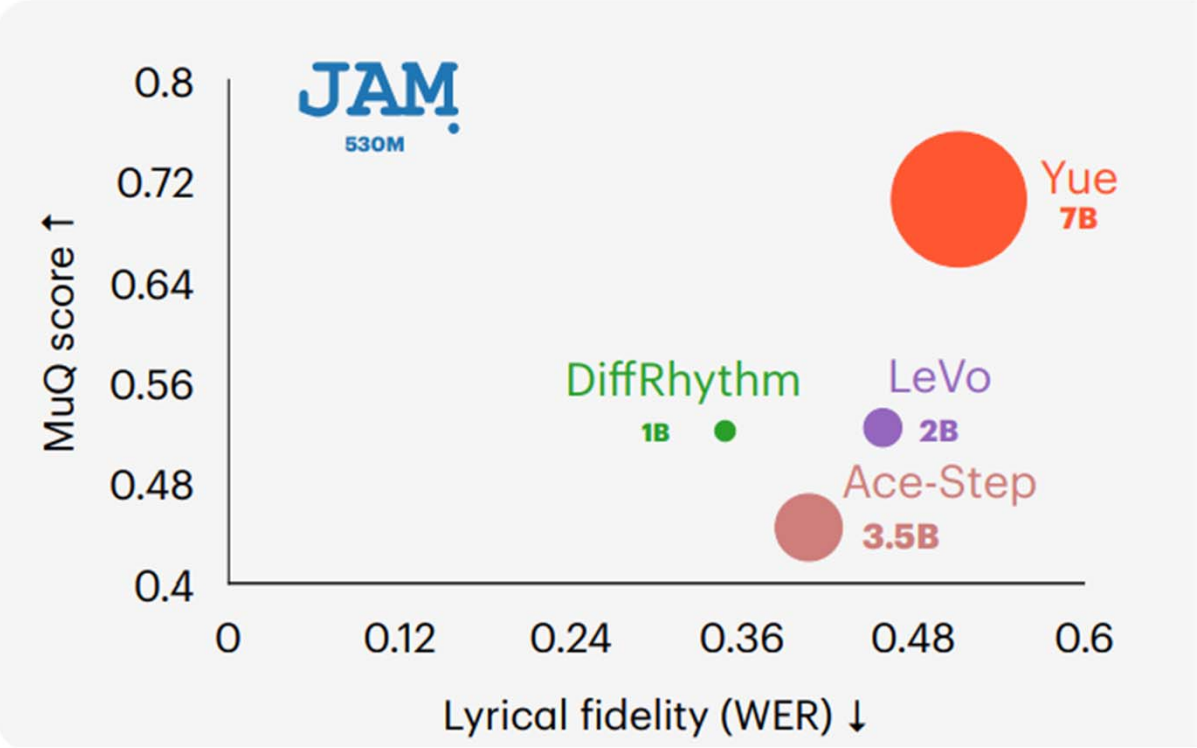
- Addressing lyric-audio alignment challenges
  - using phoneme embeddings alone is not enough
  - the Representation Alignment (**REPA**) framework is found critical



JAM

https://declare-lab.github.io/jamify

- Conditional flow-matching model (530M parameters)
- Use both **word-** and **phoneme-level timing** inputs (not just sentence-level) for reduced word error rates (WER) and phoneme error rates (PER)
- Use SongEval as the reward signal for **DPO**



Ref: Liu et al, "JAM: A tiny flow-based song generator with fine-grained controllability and aesthetic alignment," arXiv 2025