Generative Models for Text-to-Speech and Singing Voice Synthesis

Ren Yi

Research Scientist @ TikTok

ByteDance

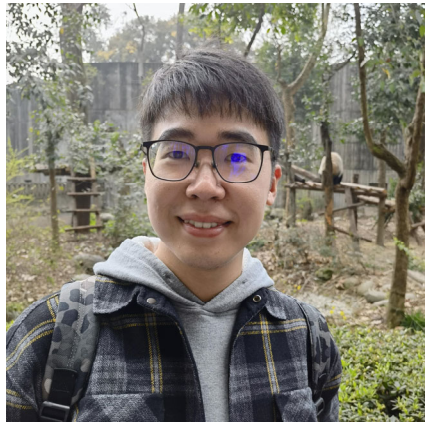
Outline

- About Me
- Foundation of Text-to-Speech (TTS)
- Non-Generative and Generative Models
- Generative TTS Models
- Foundation of Singing Voice Synthesis (SVS)
- Generative SVS Models

About Me

ByteDance 字节跳动

About Me



- My name is Ren Yi.
- I graduated from
  - Bachelor: Chu Kochen Honors College, Zhejiang University;
  - Master: Zhejiang University, advised by Zhou Zhao.
- Now I am a research scientist @ Tiktok Singapore, leading a fundamental audio research team in AILab.
- My research interests include **speech synthesis, singing voice synthesis, neural machine translation and talking face generation.**
- I have published more than 60 papers with 3000+ citations at the top international AI conferences.
- My personal homepage: https://rayeren.github.io/.
- Email: ren.yi@bytedance.com

ByteDance

Foundation of Text-to-Speech (TTS)

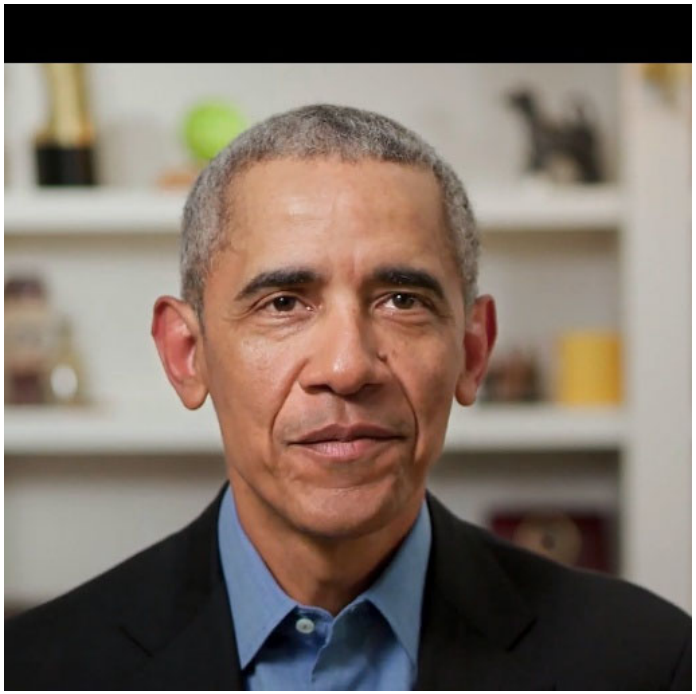
ByteDance 字节跳动

Obama Demo

“Over the past few years, speech synthesis systems have seen rapid advances thanks to deep learning. As anyone who owns a voice assistant knows, artificial voices are becoming more and more natural and convincing. The good news is you can recreate this impressive technology yourself, using high quality open-source tools.”

ByteDance

TTS + Talking head



+

TTS Demo (English)

Text	Happy	Sad
Hey, how's it going? I'd love to have someone to chat with.		
Do you want to grab a coffee or maybe go for a walk sometime?		

https://www.volcengine.com/product/tts

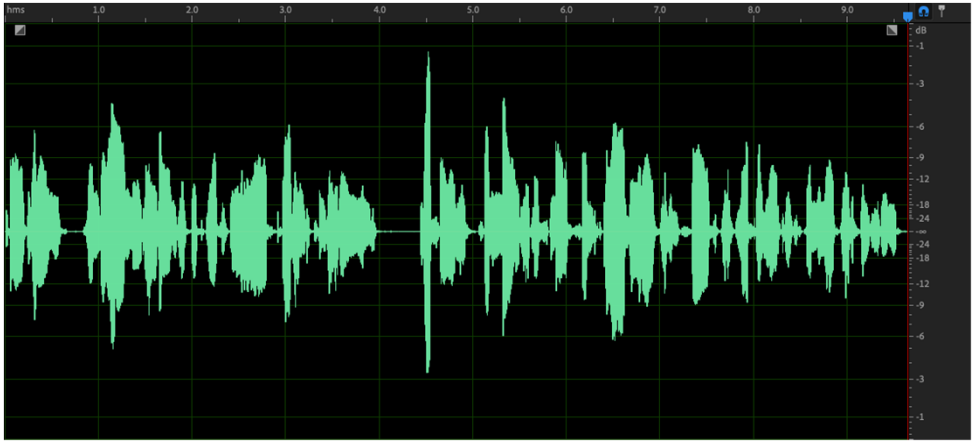
TTS Demo (Chinese)

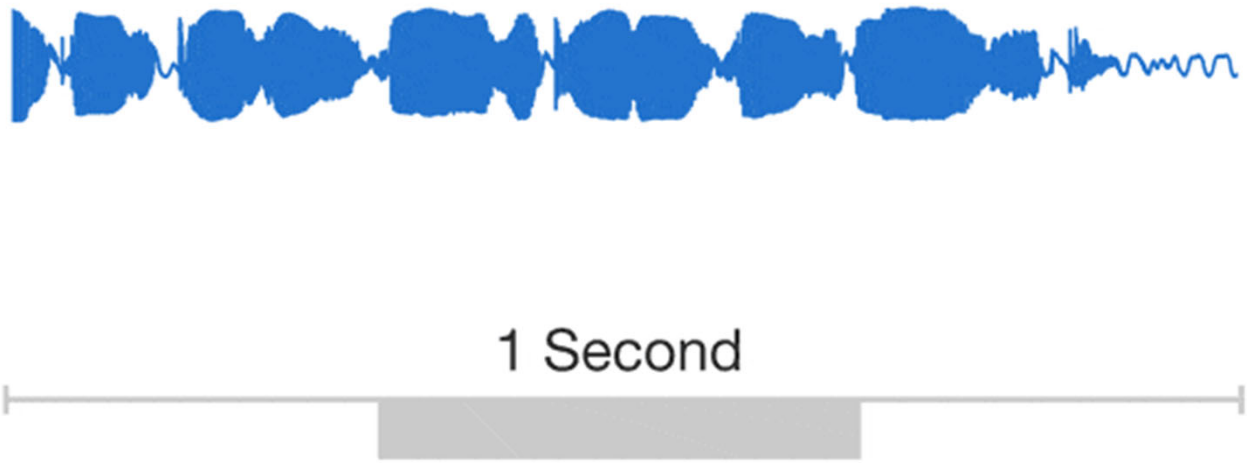
Text	Man	Child	Woman
远远观望着也挺好的			
今天去打疫苗咯			
[scare]朱佩佩同学啊，你可别诬陷我啊，我一周也就迟到五天。			
[surprise]秧秧，怎么是你？			

https://www.volcengine.com/product/tts

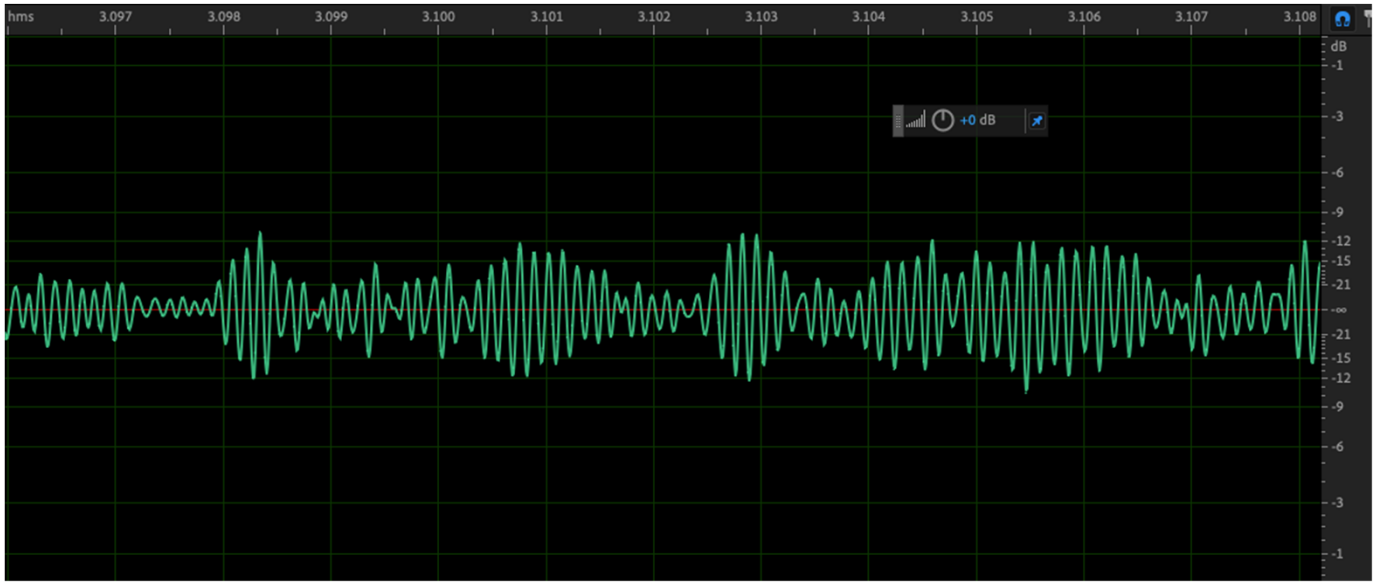
How Do Human Produce Speech?

Waveforms

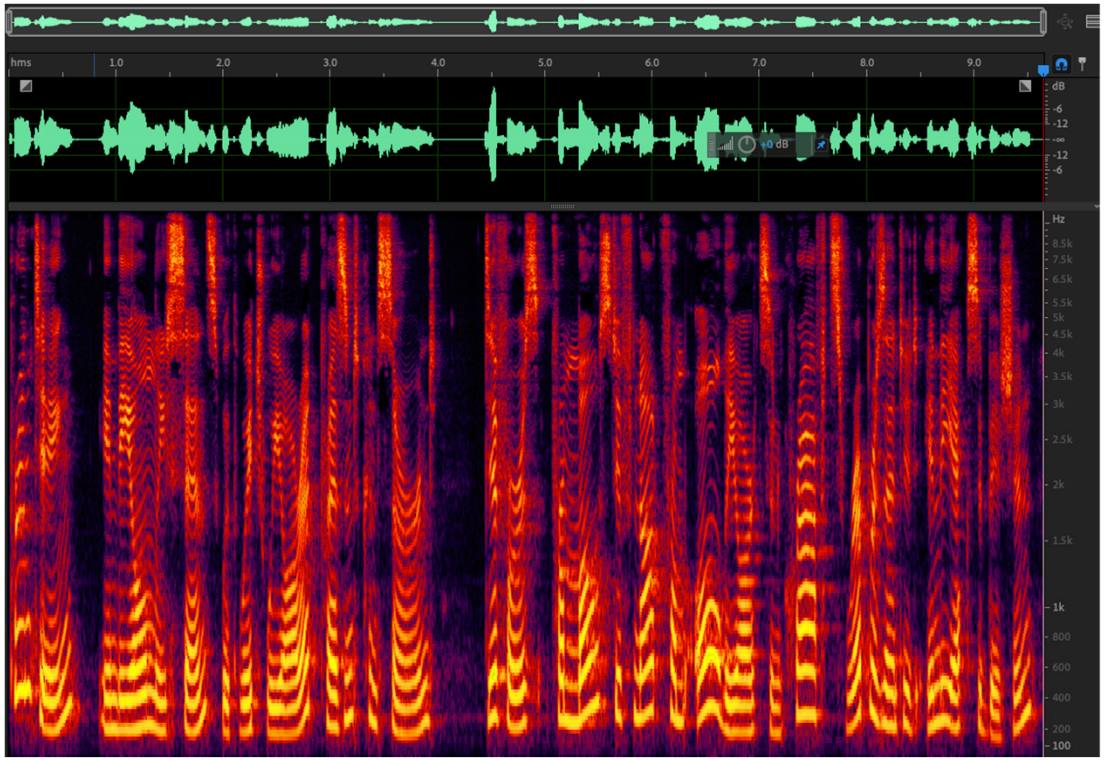




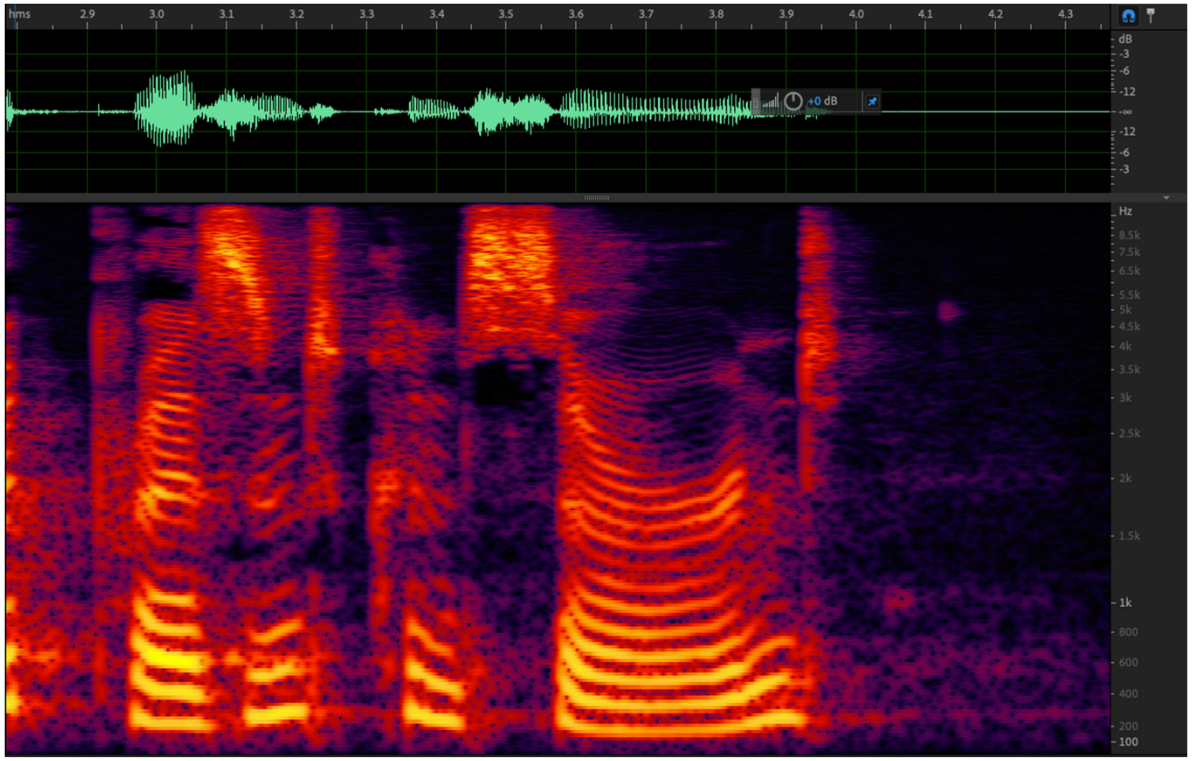
Waveforms



Spectrograms

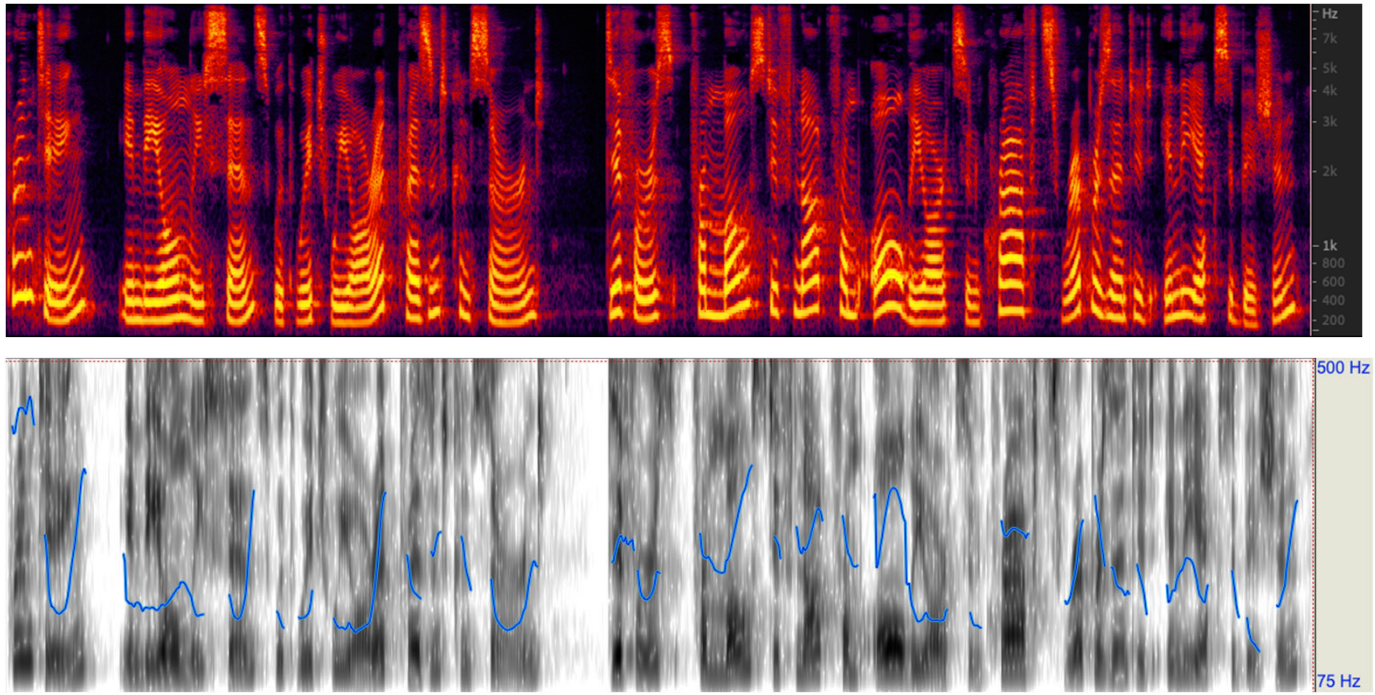


Spectrograms



Pitch

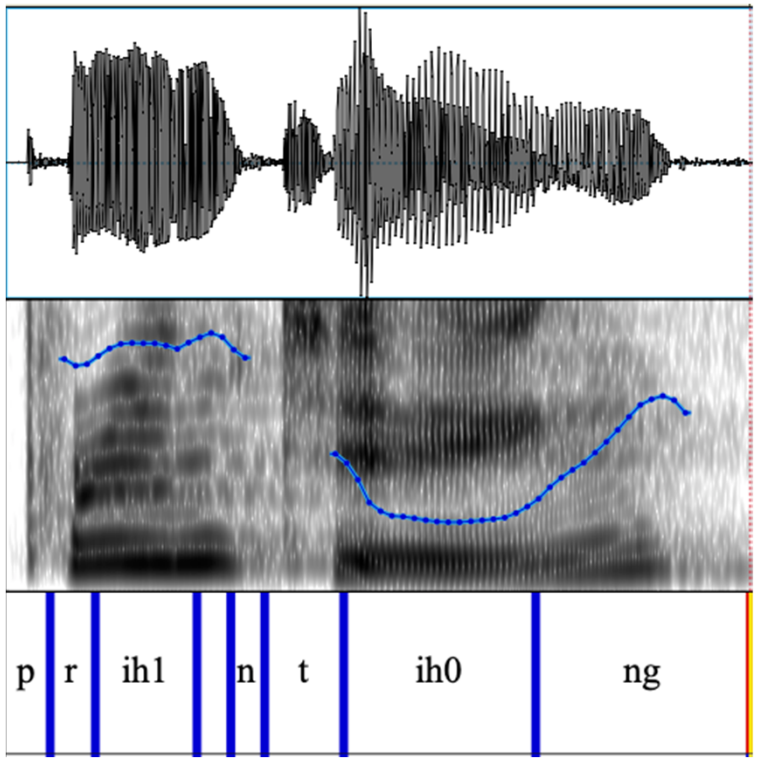
- Fundamental/harmonic
- unvoiced/voiced



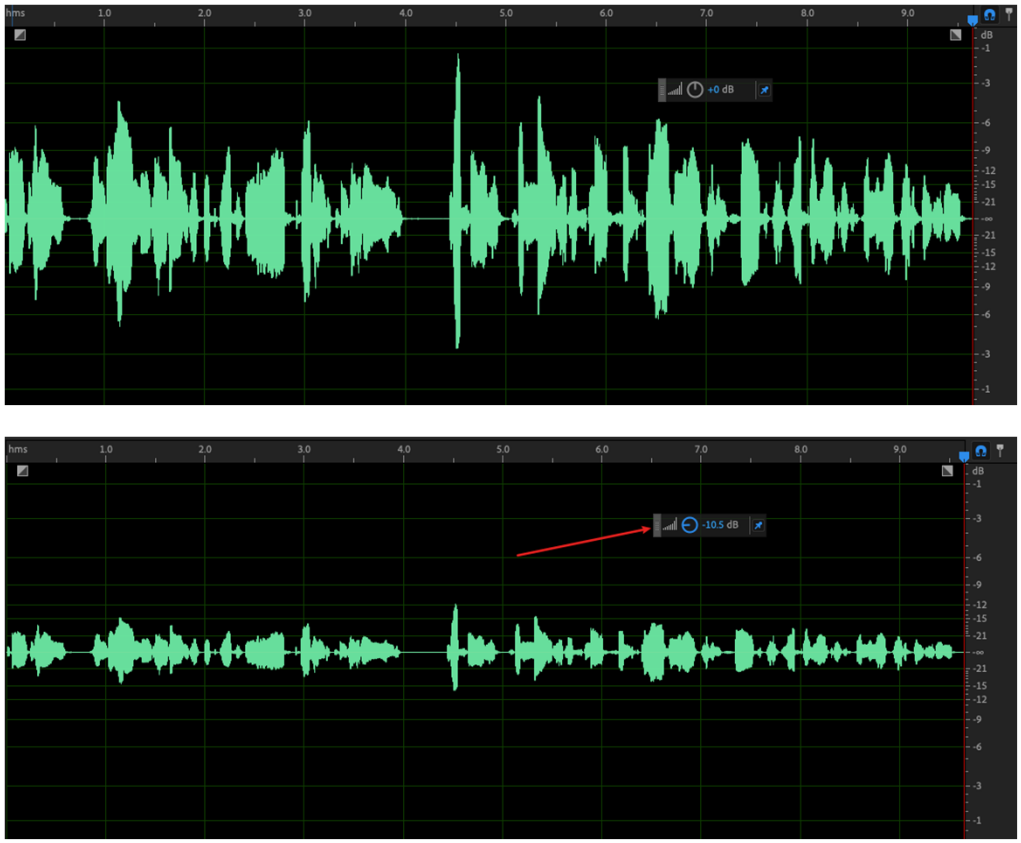
Content

- Text
  - Printing, in the only sense with which we are at present concerned, differs from most if not from all the arts and crafts represented in the Exhibition
- Phoneme
  - P R IH1 N T IH0 NG | , | IH0 N | DH AH0 | OW1 N L IY0 | S EH1 N S | W IH1 DH | W IH1 CH | W IY1 | AA1 R | AE1 T | P R EH1 Z AH0 N T | K AH0 N S ER1 N D | , | D IH1 F ER0 Z | F R AH1 M | M OW1 S T | IH1 F | N AA1 T | F R AH1 M | AO1 L | DH AH0 | AA1 R T S | AH0 N D | K R AE1 F T S | R EH2 P R AH0 Z EH1 N T AH0 D | IH0 N | DH AH0 | EH2 K S AH0 B IH1 SH AH0 N

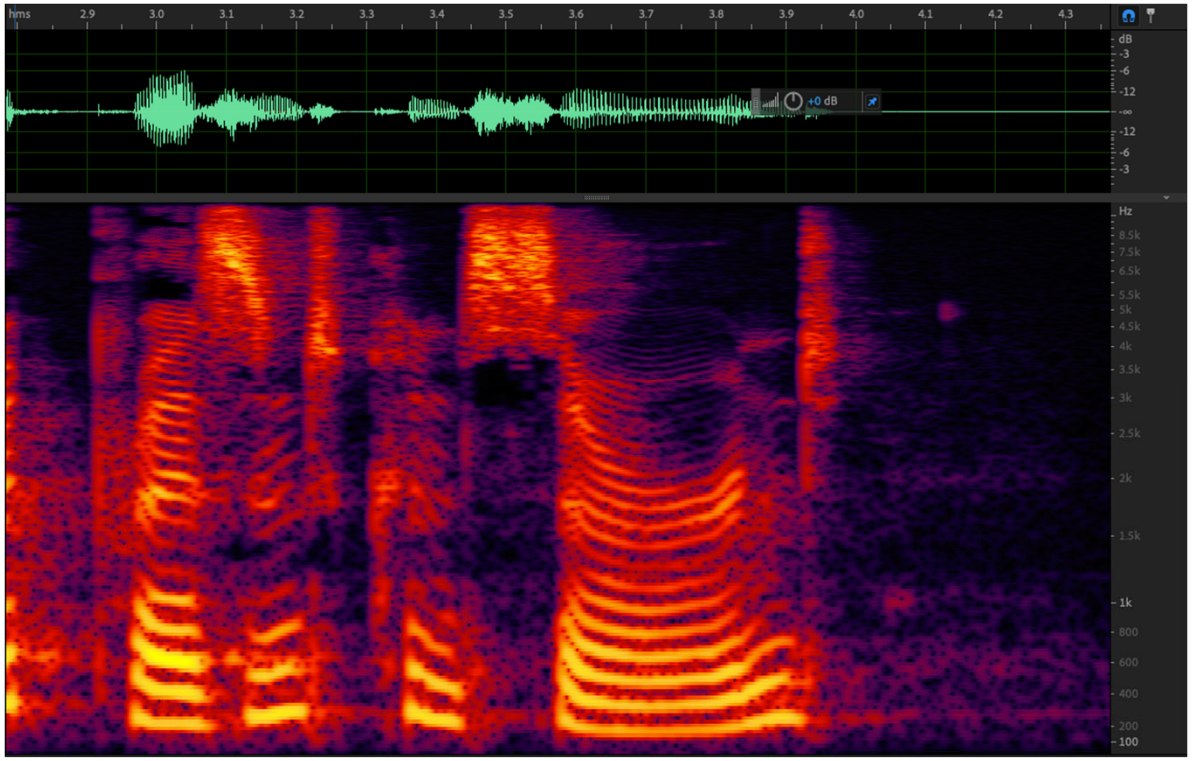
Alignment



Energy/Volume



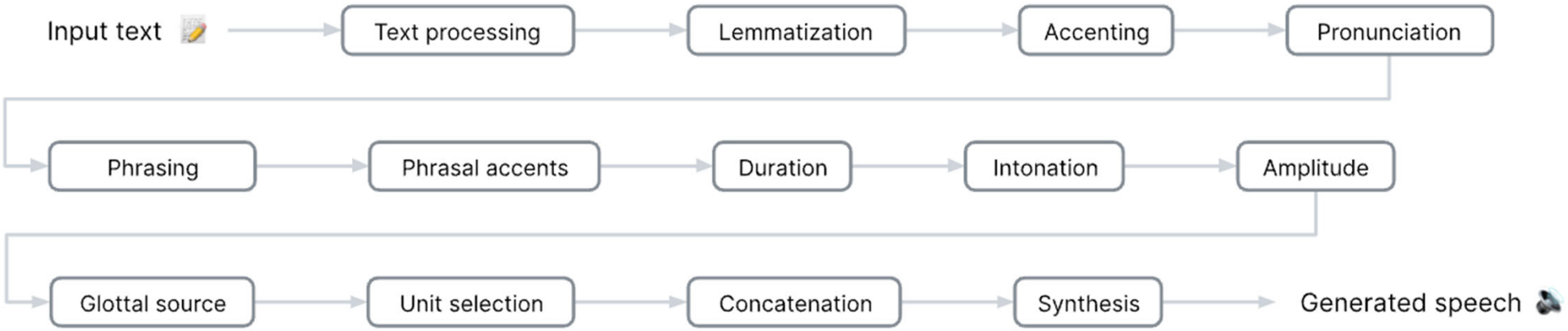
Timbre



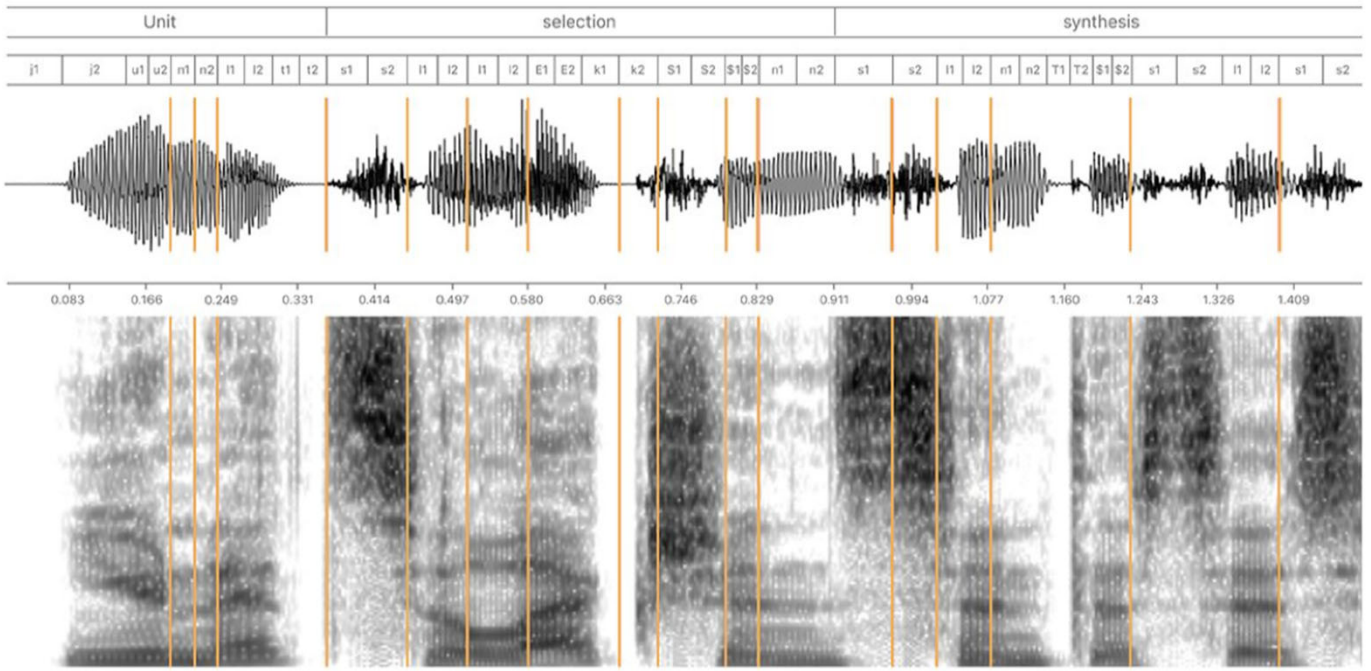
Pink Trombone - Bare-handed Speech Synthesis

- https://github.com/zakaton/Pink-Trombone#-common-phonemes
- https://zakaton.github.io/Pink-Trombone/

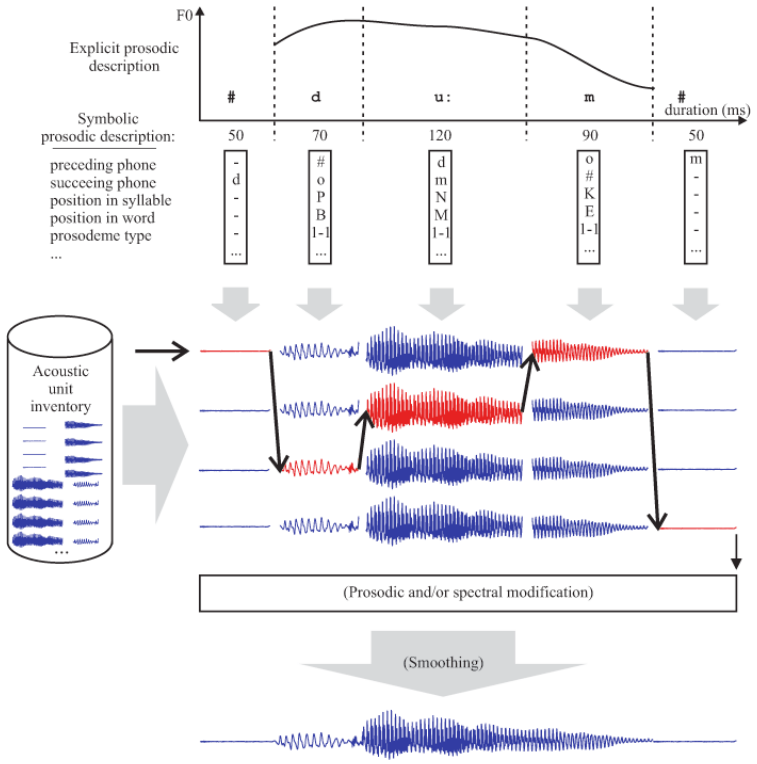
A 13 stage TTS system from Bell Labs



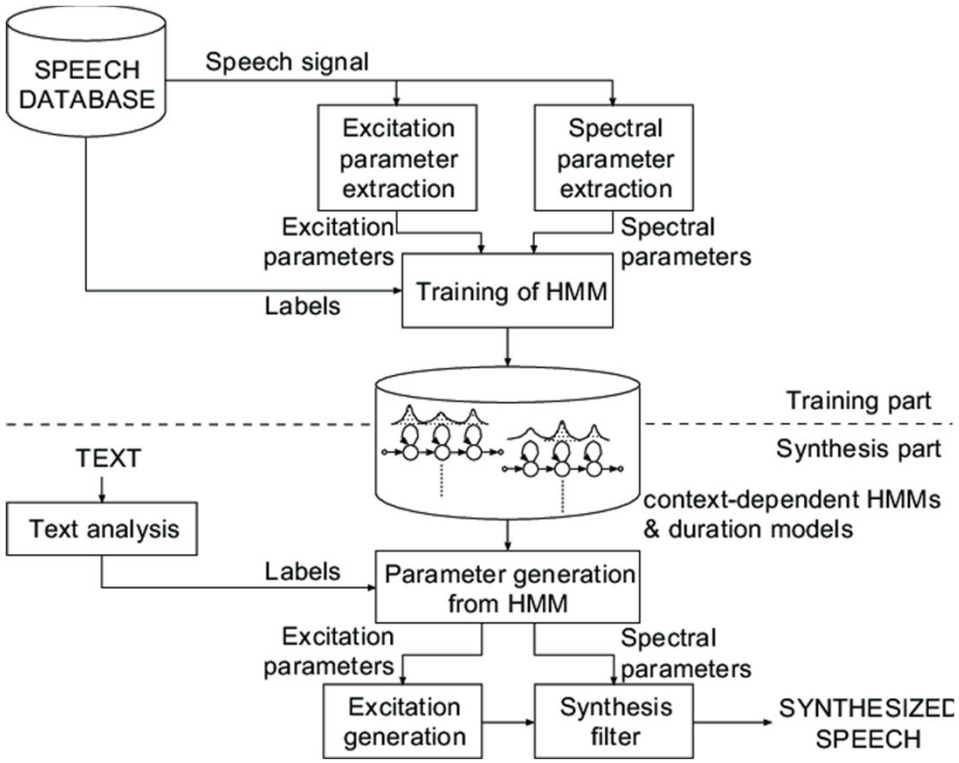
Concatenative Speech Synthesis



Concatenative Speech Synthesis

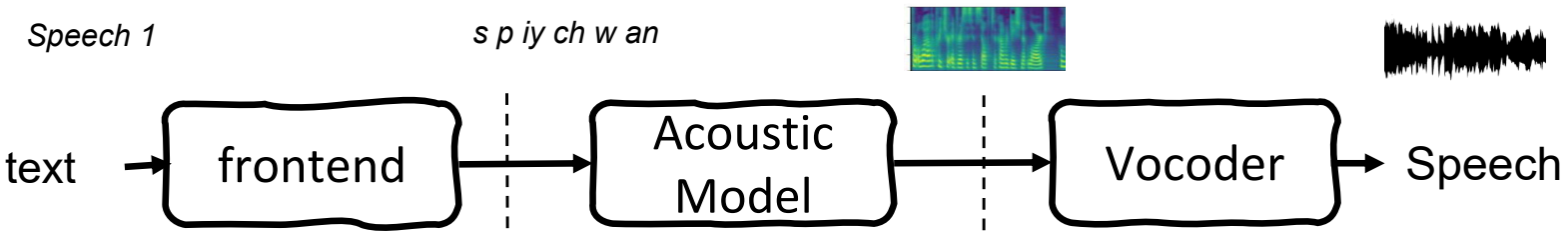


Statistical Parametric Speech Synthesis



A typical modern TTS pipeline

Modern End2end TTS Framework



Text -> Linguistic feature

- Text Normalization
- Grapheme to Phoneme
- Prosody Flag Prediction

Linguistic feature -> Acoustic

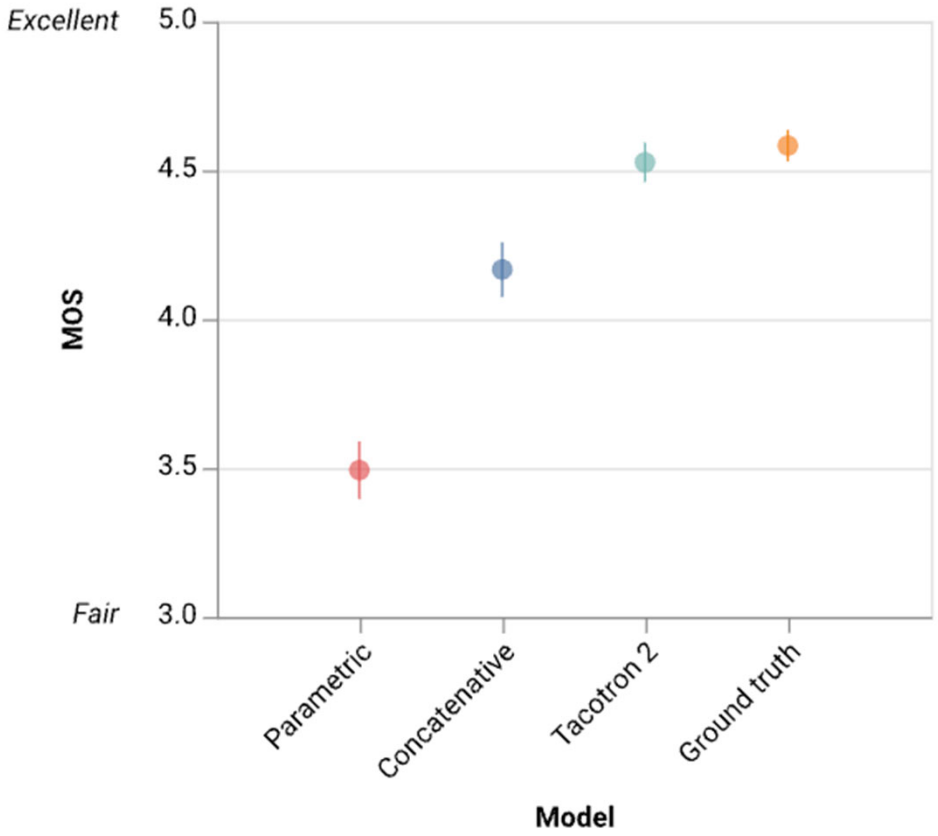
- Tacotron 1/2
- Deep Voice 1/2/3
- Transformer TTS
- FastSpeech/FastSpeech 2
- Glow-TTS
- BVAE-TTS
- PortaSpeech
- VITS

Acoustic features→ Speech

- Griffin-Lim
- WORLD/STRAIGHT
- WaveNet/Parallel WaveNet
- WaveGlow/FloWaveNet
- WaveRNN/LPCNet
- HIFIGAN

Synthesized Speech Quality Evaluation

- Ultimately, we have to rely on human judgement
- We want to do that in a structured way
- Industry standard is **Mean Opinion Score (MOS)**
  - Ask a pool of human reviewers to score the naturalness of the speech on a five point scale (1 = Bad, 2 = Poor, 3 = Fair, 4 = Good, 5 = Excellent)
  - Take the average of these scores



Text-to-speech as sequence-to-sequence mapping

Automatic speech recognition (ASR)

 → “Hello my name is Heiga Zen”

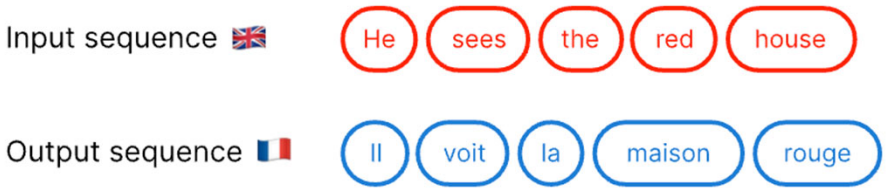
Machine translation (MT)

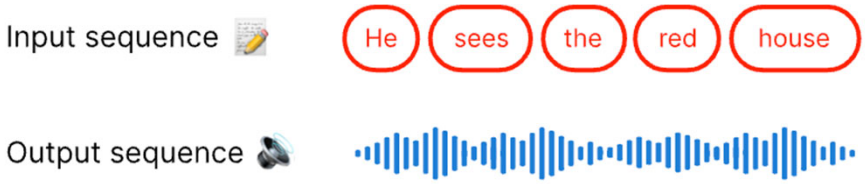
“Hello my name is Heiga Zen” → “Ich heiße Heiga Zen”

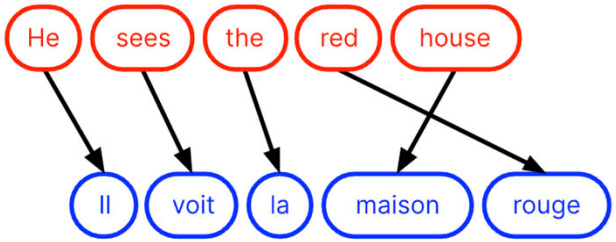
Text-to-speech synthesis (TTS)

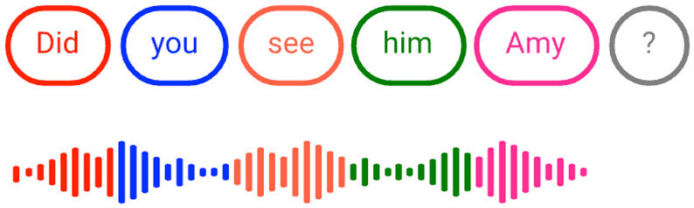
“Hello my name is Heiga Zen” → 

Text-to-speech as sequence-to-sequence mapping







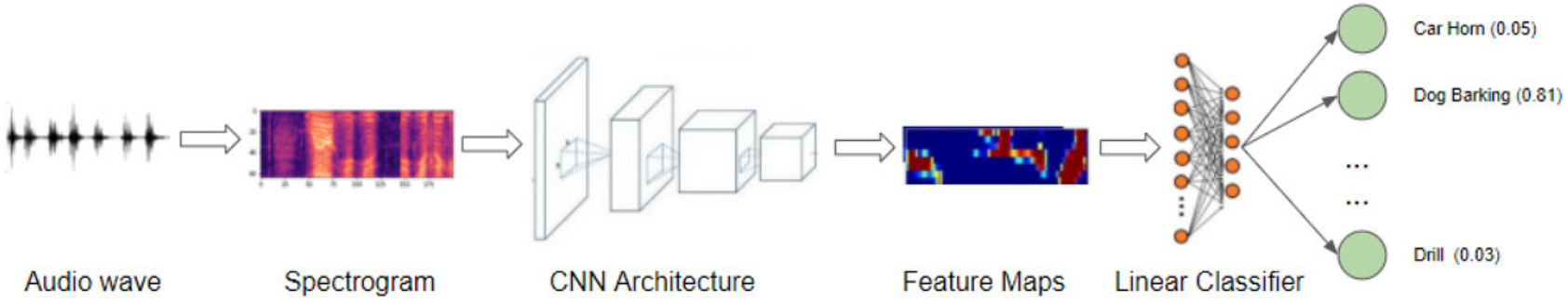


Characteristics of seq2seq problem

- No one-to-one (or one-to-N) mapping between input items and output items
- An output item could depend on a weighted combination of input items (attention)
- We may need to look back at the output sequence generated so far to ensure fluency (autoregressive)

Non-Generative and Generative Models

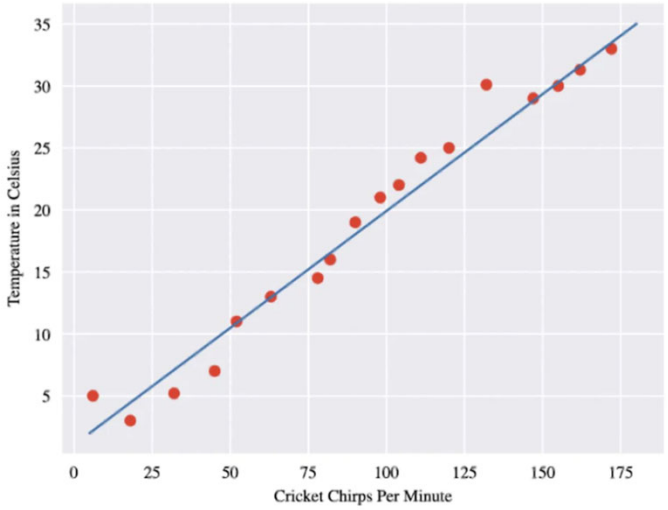
Classification

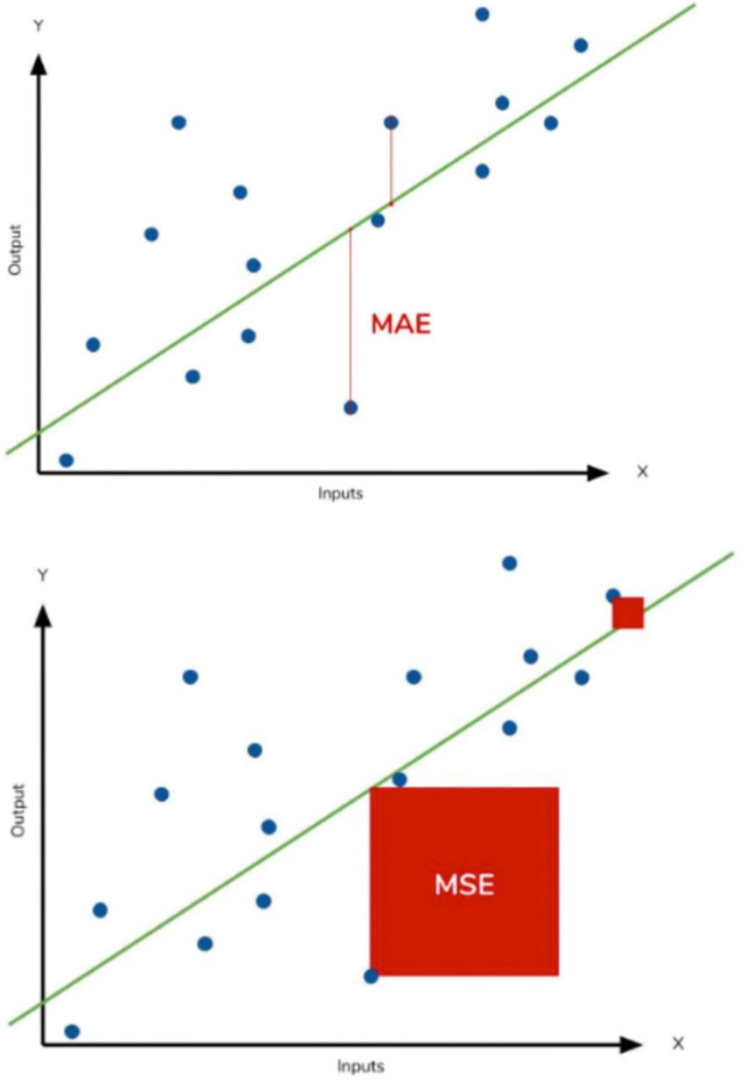


Classification

- Model P(Y|X), where X can be image, text or audio; Y is a label.
- Info(X) >> Info(Y) and I(X, Y) ≈ Info(Y)
- Usually use cross entropy as training loss, which is a part of the KL divergence.

Regression





Dance

Regression

- Model P(Y|X), where X can be image, text or audio; Y is continuous value.
- Usually use L1 or L2 (MSE) losses
- Understand L1/L2 loss from probability perspective

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} p_{\mathrm{model}}(\mathbb{X}; \boldsymbol{\theta}),$$

$$= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{m} p_{\mathrm{model}}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

$$p(y \mid \boldsymbol{x}) = \mathcal{N}(y; \hat{y}(\boldsymbol{x}; \boldsymbol{w}), \sigma^2)$$

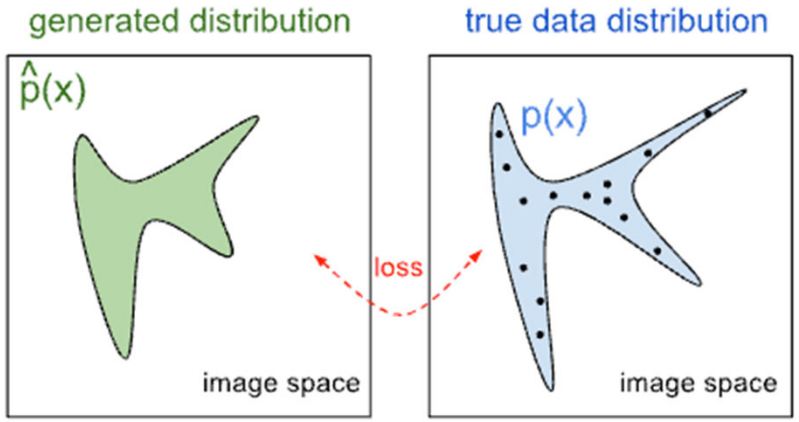
$$p(x^{(i)}; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(x^{(i)} - \mu)^2}{\sigma^2}\right)$$

L2 loss is under the Gaussian assumption about the distribution.
Similarly, L1 is under the Laplace distribution assumption.

https://towardsdatascience.com/mse-is-cross-entropy-at-heart-maximum-likelihood-estimation-explained-181a29450a0b

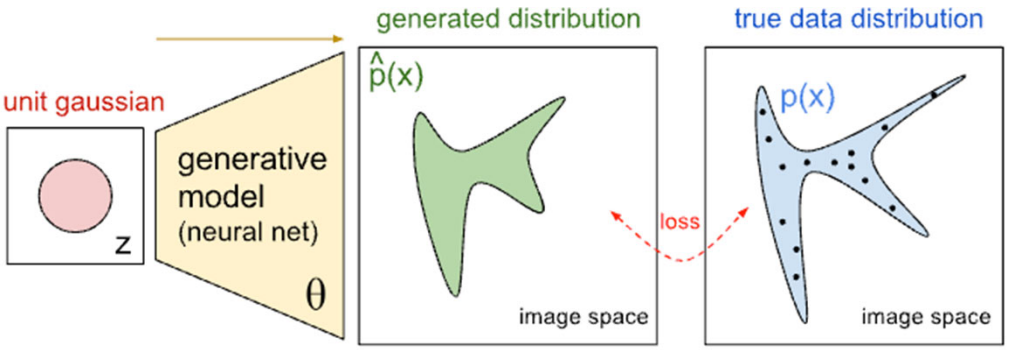
Are Regression/Classification Models Generative?

- Answer: No.
- Reason:
  - The distribution assumptions in these models are very simple, which cannot model the complex high-dimensional data distribution (like a data point in 256*256 image space, which has 65536 dims)



Definition of Generative Model

- A kind of model that can fit the **real distribution of data points**.
- To train a generative model we first collect a large amount of data in some domain (e.g., think millions of images, sentences, or sounds, etc.) and then train a model to **generate data like it**. (from OpenAI's blog)



- What I cannot create, I do not understand." —*Richard Feynman*

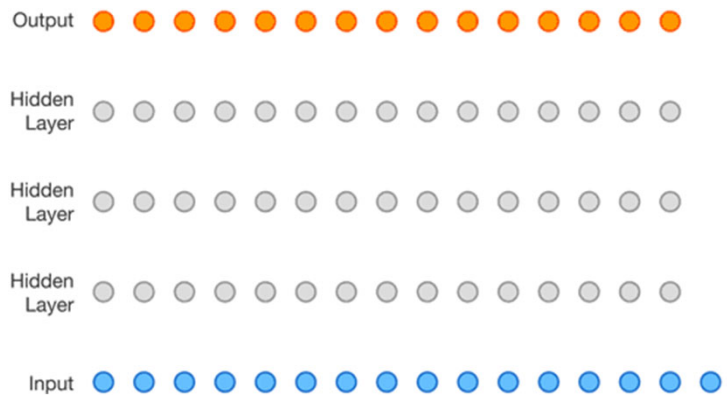
https://openai.com/blog/generative-models/

Autoregressive

- Tackle the joint distribution issue using probability chain rule

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i | x_1, x_2, \ldots, x_{i-1}) = \prod_{i=1}^{n} p(x_i | \mathbf{x}_{<i}) \qquad X_{x+1} = softmax[\Pi_{t=1}^{T} P(X_{t+1} \mid X_t, X_{t-1}, ..., X_1, X_0)]$$

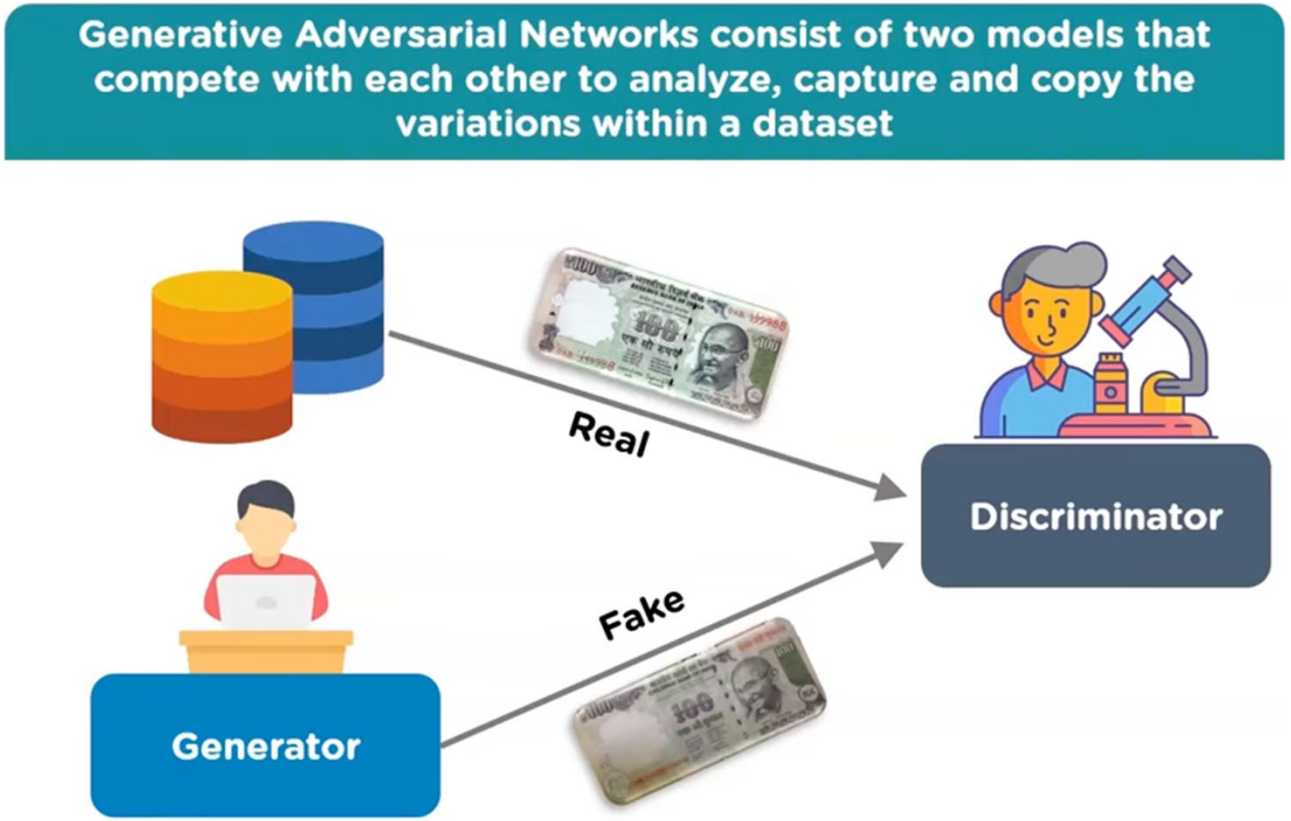
- Training: Teacher forcing, use ground-truth $X_{<i}$
- Inference: use previously generated $X_{<i}$



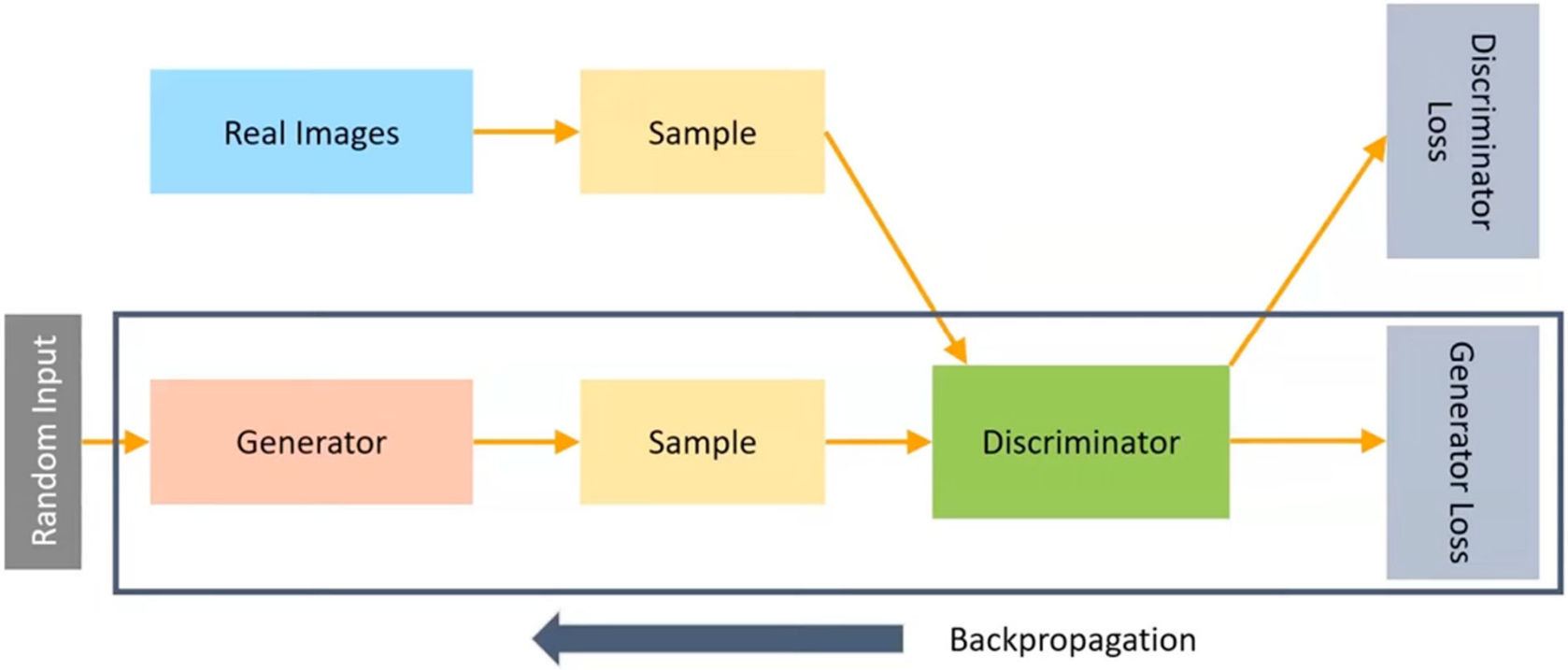
Autoregressive

- Strengths:
  - Mathematically complete
  - Simple to understand
- Weakness:
  - Gap between training and inference
  - Conditional distribution $p(x_i|x_{<i})$ data samples are sparse. We can hardly find enough data points with same prefix $x_{<i}$ to model the complete uncertainty and distribution shape.

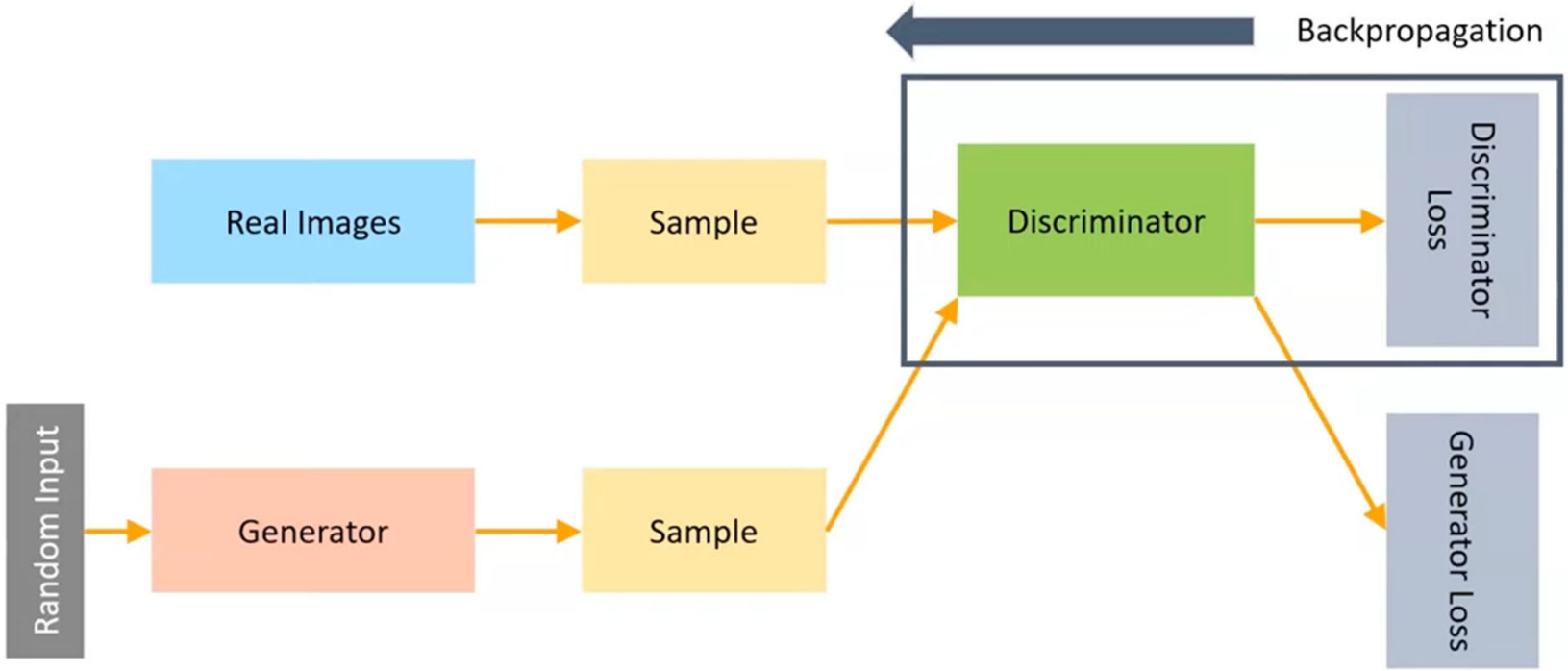
GAN (Generative Adversarial Network)



GAN – Generator Training



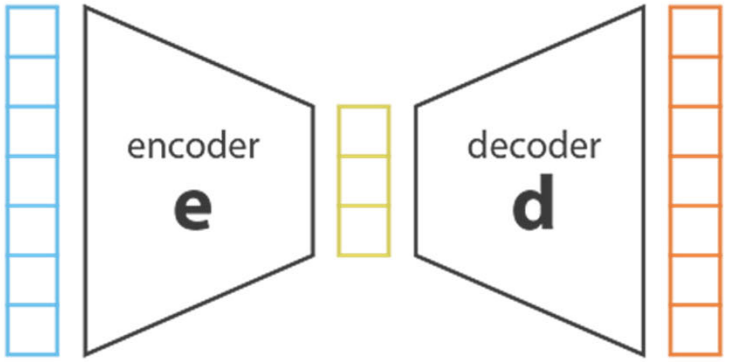
GAN – Discriminator Training



GAN

- Strengths
  - Simple: inference is totally same as training process for generator.
- Weaknesses
  - Mode collapse. Mode collapse prevents you from maintaining all uncertainty (modes) in target space, especially in conditional generation.
  - GAN loss should be adjusted carefully.

VAE (Variational AutoEncoder)



L1/L2, SSIM, adversarial losses……

$x = d(e(x))$ ➡ **lossless encoding**
no information is lost when reducing the number of dimensions

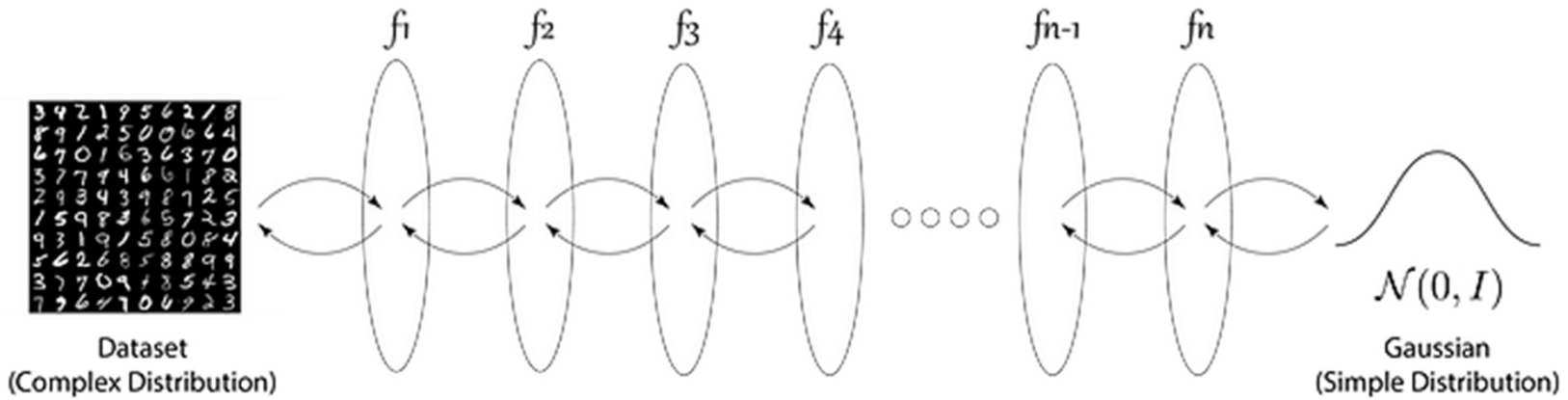
$x \neq d(e(x))$ ➡ **lossy encoding**
some information is lost when reducing the number of dimensions and can't be recovered later

VAE

- Strengths
  - Like a signal filter, can choose what information need to store in the latent space by using different losses
  - Simple and flexible
- Weaknesses
  - Posterior collapse
  - Reconstruction heavily relies on the loss function. Usually need to use with other generative models

Normalizing Flow

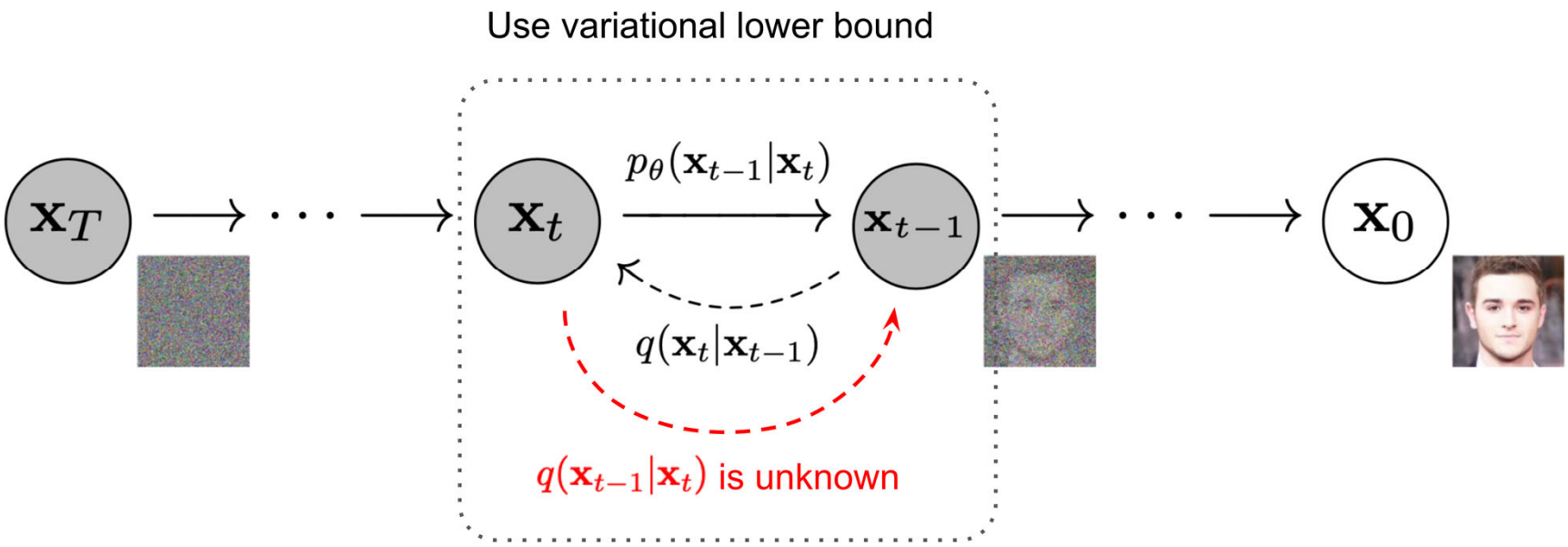
- Suppose we have data points $\{x_1, x_2, x_3\}$ sampled from q(x)
- We train a NN (denoted as f) to map these points to $f(x_1)$, $f(x_2)$ and $f(x_3)$
- If NN is invertible and the distribution of $f(x_i)$ is simple and analytical.
- Then we can sample any data points $\{x_k\}$ by sampling $x'_k \sim P(f(x))$ and $x_k = f^{-1}(x'_k)$
- So, the training target is to force P(f(x)) to be a simple distribution



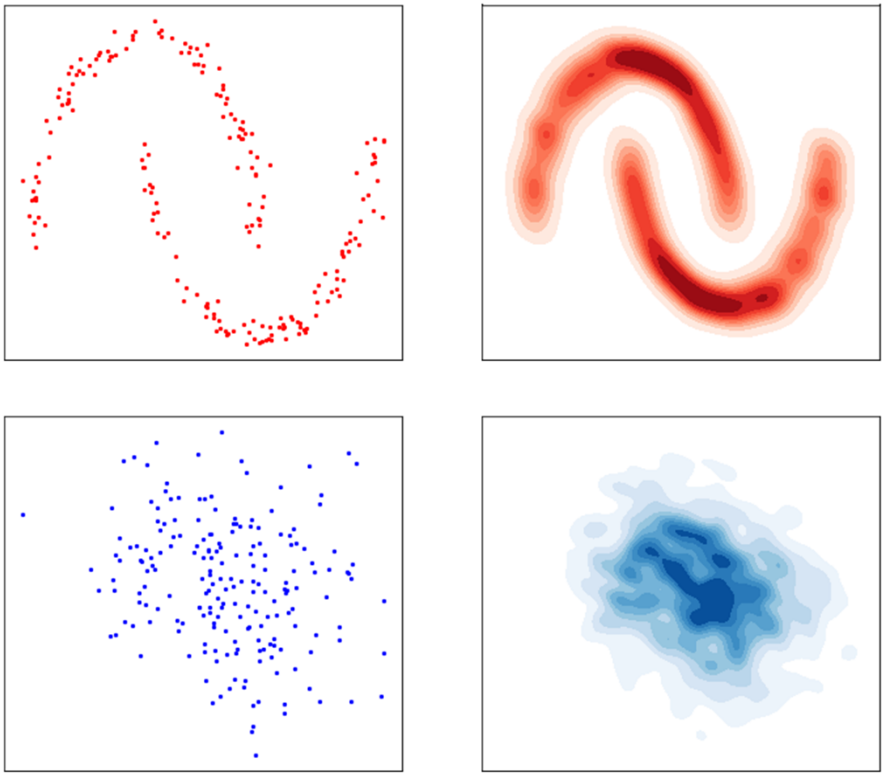
Normalizing Flow

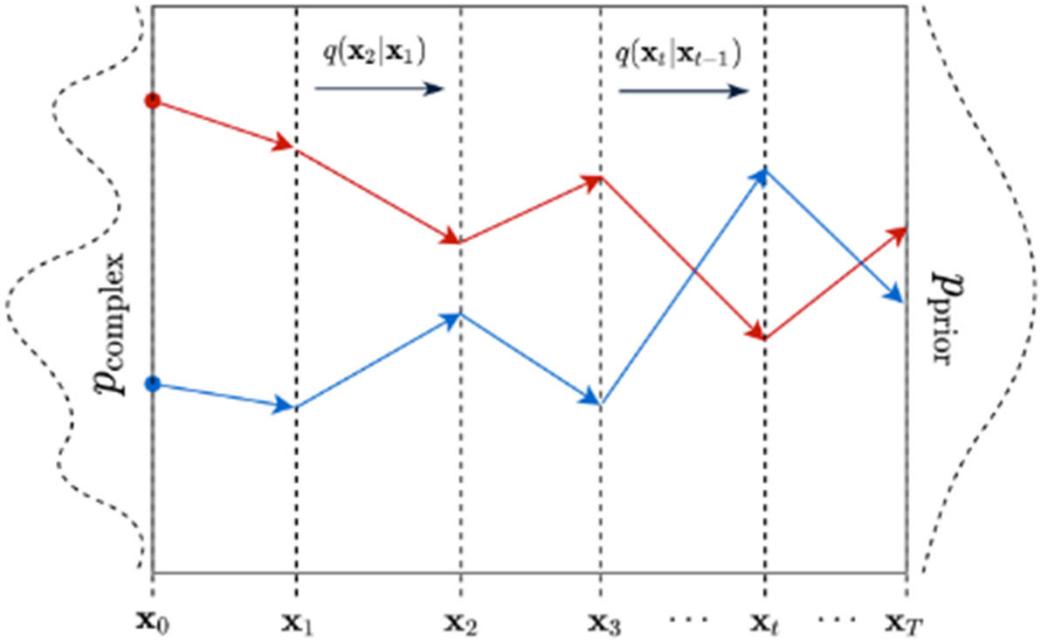
- Strengths
  - Mathematically complete
  - Strictly bijection, no information lost in conversion
- Weaknesses
  - The only model that cannot filter input noise
  - Require huge model size to map all information into prior distribution
  - Training is unstable. NaN issue occurs often due to reversibility requirement

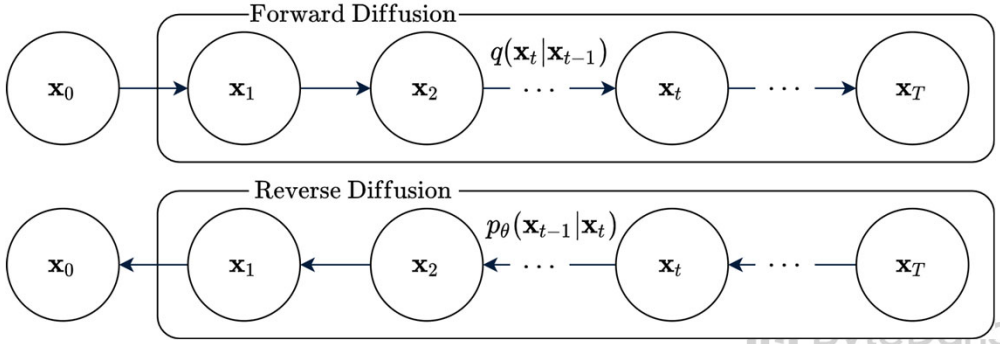
Diffusion Models



Diffusion Models







https://ayandas.me/blog-tut/2021/12/04/diffusion-prob-models.html

Diffusion Models

- Strengths
  - Mathematically complete
  - Nearly bijection, nearly no information lost in conversion (DDIM)
  - Very simple training loss and network architecture
  - Training is stable. Loss is relevant to generation quality
- Weaknesses
  - Sensitive to target data value range and noise schedule.
  - Slow inference.
  - Large model footprint. But not as large as normalizing flow since the bijection is not so strict.

Generative Models for Text-to-Speech

ByteDance 字节跳动

Generative TTS/SVS Model Family

- Tacotron 1/2
- FastSpeech / FastSpeech 2 (non-generative)
- Glow-TTS
- VITS
- PortaSpeech
- MegaTTS 1/2

Tacotron 2 🌮 - A bit of history

- Tacotron 1 was introduced in a March 2017 paper by Google researchers
- Quickly followed up by Tacotron 2 (December 2017) which improved on + simplified the original
- Google's implementation is closed source, but high quality open-source implementations exist (as we'll see)
- Why is it called TACOTRON?

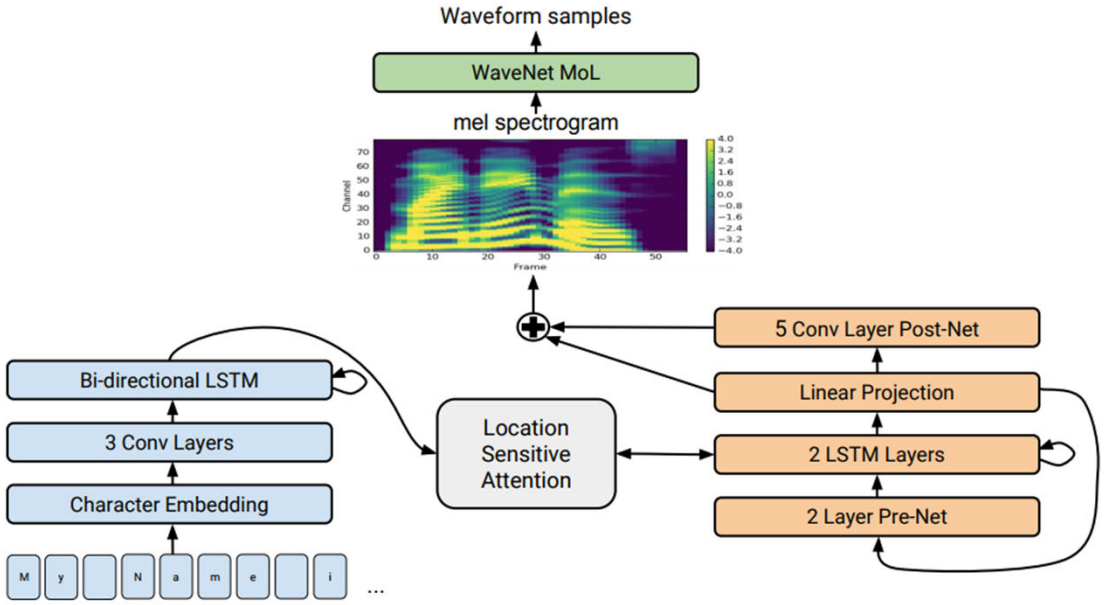
TACOTRON: TOWARDS END-TO-END SPEECH SYNTHESIS

Yuxuan Wang*, RJ Skerry-Ryan*, Daisy Stanton, Yonghui Wu, Ron J. Weiss†, Navdeep Jaitly, Zongheng Yang, Ying Xiao*, Zhifeng Chen, Samy Bengio†, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, Rif A. Saurous*

*These authors really like tacos.
†These authors would prefer sushi.

Tacotron 2

- Encoder: BLSTM
- Decoder: BLSTM
- Attention: Location sensitive attention
- Input: Char/Phoneme
- Output: Mel-spectrograms



Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions.

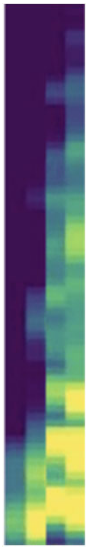
Tacotron 2 - loss & learning process

- Text: “Hello world”
- For each example in our training set we try to predict the spectrogram as accurately as possible, given the transcript. We make our prediction r frames at a time. (We call r the “reduction factor”. Typically r = 2)

Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions.

Tacotron 2 - loss & learning process

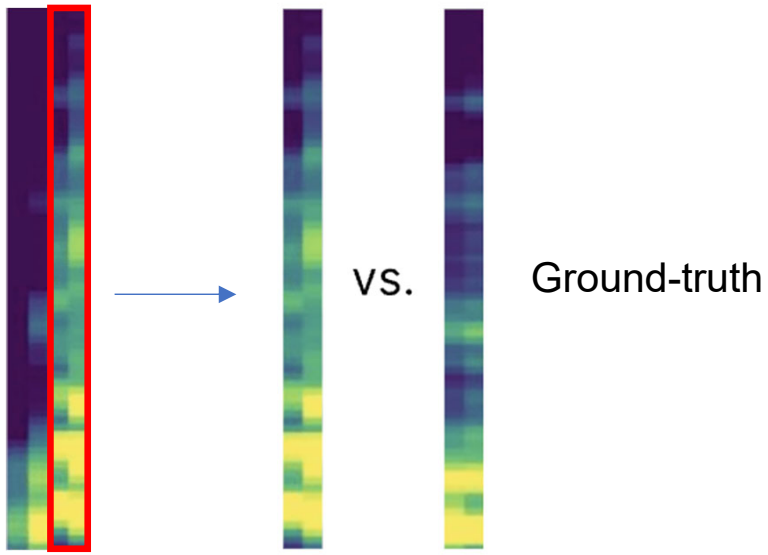
- Text: “Hello world”
- We predict the next r frames.
- But for the purposes of our prediction, our previously predicted frames are replaced by the frames from the spectrogram in the training data (teacher forcing)



Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions.

Tacotron 2 - loss & learning process

- At each step we calculate the difference between our predicted spectrogram frames and our ground truth frames (L2 frame reconstruction loss).
- This is the key loss we'll be seeking to minimise during training.

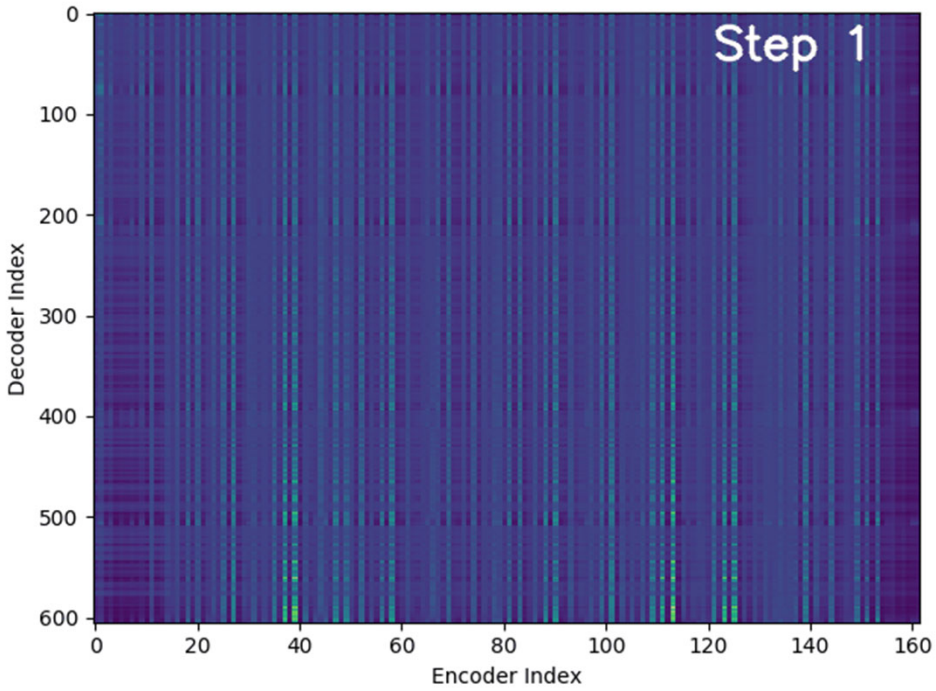


ByteDance

Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions.

Tacotron 2 - attention

- Recall that attention will determine the correspondence between our input and output sequences. This means that for TTS attention will control speech pace, rhythm, stress etc.



Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions.

Defects of Autoregressive TTS

- Traditional end-to-end autoregressive speech synthesis methods have some fatal flaws:
  - The synthesis speed of traditional end-to-end methods is very slow, making it difficult to apply in situations where speed and real-time requirements are high.
  - Limited by the synthesis speed, the expansion cost of this method is very high, and it is difficult to provide stable services in scenarios with high traffic and concurrency.
  - Traditional end-to-end speech synthesis can result in repeated or omitted words, which is extremely fatal and intolerable for commercial speech synthesis systems.
  - Traditional end-to-end methods cannot control the speed, rhythm, and pauses of speech at a fine-grained level.

FastSpeech: Fast, Robust and Controllable Text to Speech

NeurIPS 2019

Yi Ren[*]
Zhejiang University
rayeren@zju.edu.cn

Yangjun Ruan[*]
Zhejiang University
ruanyj3107@zju.edu.cn

Xu Tan
Microsoft Research
xuta@microsoft.com

Tao Qin
Microsoft Research
taoqin@microsoft.com

Sheng Zhao
Microsoft STC Asia
Sheng.Zhao@microsoft.com

Zhou Zhao[†]
Zhejiang University
zhaozhou@zju.edu.cn

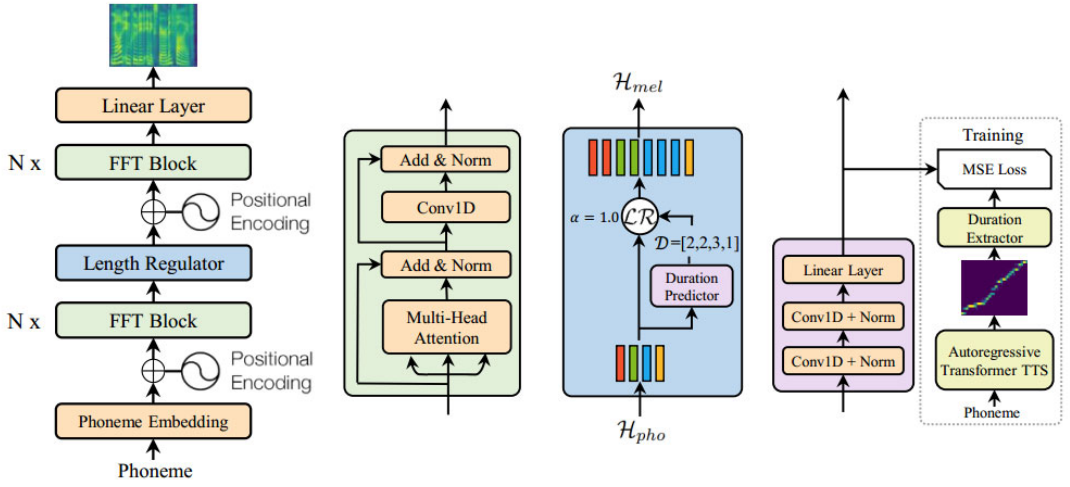
Tie-Yan Liu
Microsoft Research
tyliu@microsoft.com

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

ByteDance

FastSpeech

- Uses a fully parallel non-autoregressive architecture to solve the problem of slow generation speed
- Introduces knowledge distillation to make the performance of audio generation close to that of the autoregressive model.
- Introduces a duration predictor to predict strong alignment between text and spectrum, eliminating word skipping and missing words in generated speech.
- Introduces length regulator to solve the controllability problem of autoregressive models.



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

Figure 1: The overall architecture for FastSpeech. (a). The feed-forward Transformer. (b). The feed-forward Transformer block. (c). The length regulator. (d). The duration predictor. MSE loss denotes the loss between predicted and extracted duration, which only exists in the training process.

FastSpeech

- Voice Quality

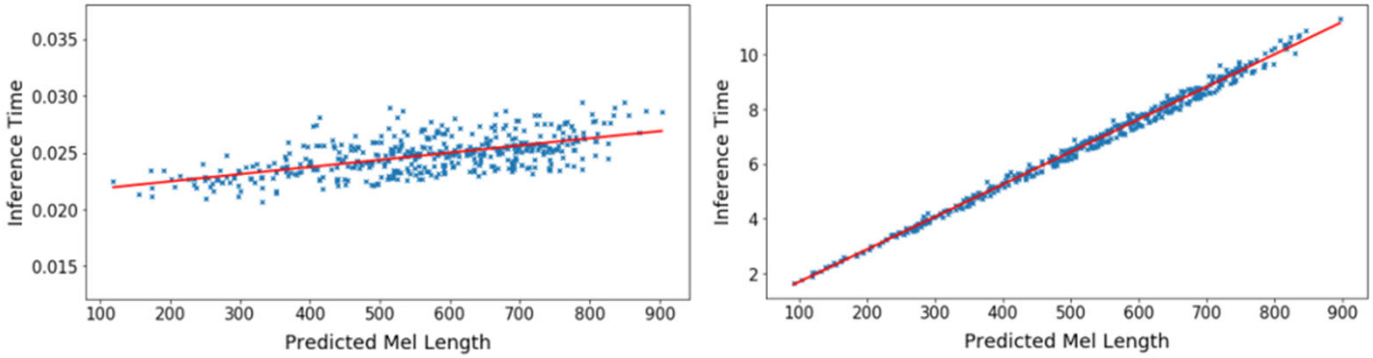
Method	MOS
GT	4.41 ± 0.08
GT (Mel + WaveGlow)	4.00 ± 0.09
Tacotron 2 [20] (Mel + WaveGlow)	3.86 ± 0.09
Merlin [25] (WORLD)	2.40 ± 0.13
Transformer TTS [13] (Mel + WaveGlow)	3.88 ± 0.09
FastSpeech (Mel + WaveGlow)	3.84 ± 0.08

Table 1: The MOS with 95% confidence intervals.

FastSpeech

- Inference speedup

Method	Latency (ms)	Speedup
Transformer TTS [13] (Mel)	6.735 ± 3.969	/
FastSpeech (Mel)	0.025 ± 0.005	$269.40\times$
Transformer TTS [13] (Mel + WaveGlow)	6.895 ± 3.969	/
FastSpeech (Mel + WaveGlow)	0.180 ± 0.078	$38.30\times$



(a) FastSpeech (b) Transformer TTS

FastSpeech

- Robustness

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.

zero zero zero zero zero zero zero zero two seven nine eight F three forty zero zero zero zero zero six four two eight zero one eight

c five eight zero three three nine a zero bf eight FALSE zero zero zero bba3add2 - c229 - 4cdb -

Calendaring agent failed with error code 0x80070005 while saving appointment .

Exit process - break ld - Load module - output ud - Unload module - ignore ser - System error - ignore ibp - Initial breakpoint -

Common DB connectors include the DB - nine , DB - fifteen , DB - nineteen , DB - twenty five , DB - thirty seven , and DB - fifty connectors .

To deliver interfaces that are significantly better suited to create and process RFC eight twenty one , RFC eight twenty two , RFC nine seventy seven , and MIME content .

int1 , int2 , int3 , int4 , int5 , int6 , int7 , int8 , int9 ,

seven _ ctl00 ctl04 ctl01 ctl00 ctl00

Http0XX , Http1XX , Http2XX , Http3XX ,

config file must contain A , B , C , D , E , F , and G .

mondo - debug mondo - ship motif - debug motif - ship sts - debug sts - ship Comparing local files to checkpoint files ...

Rusbvts . dll Dsaccessbvts . dll Exchmembvt . dll Draino . dll Im trying to deploy a new topology , and I keep getting this error .

You can call me directly at four two five seven zero three seven three four four or my cell four two five four four four seven four seven four or send me a meeting request with all the appropriate information .

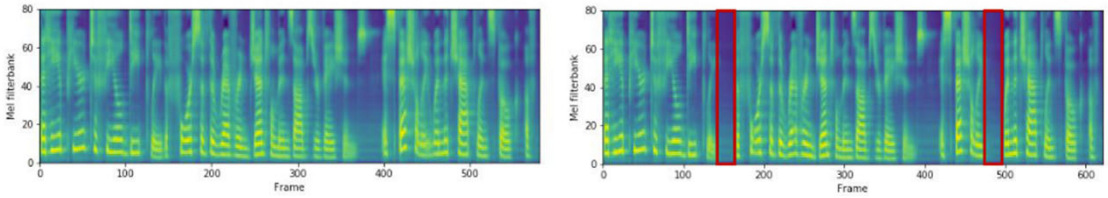
Failed zero point zero zero percent < one zero zero one zero zero zero zero Internal . Exchange . ContentFilter . BVT ContentFilter . BVT_log . xml Error ! Filename not specified .

C colon backslash o one two f c p a r t y backslash d e v one two backslash oasys backslash legacy backslash web backslash HELP

src backslash mapi backslash t n e f d e c dot c dot o l d backslash backslash m o z a r t f one backslash e x five

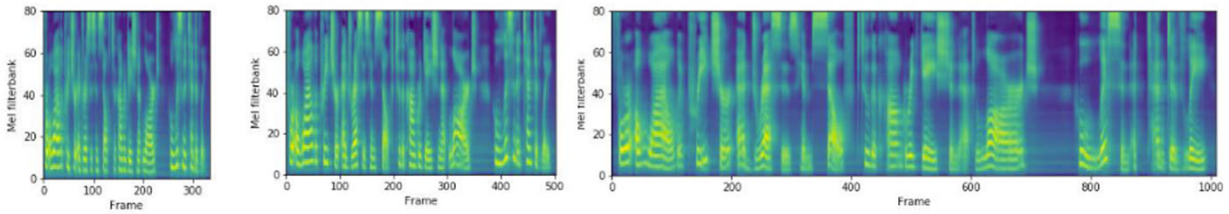
FastSpeech

- Controllability



(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.

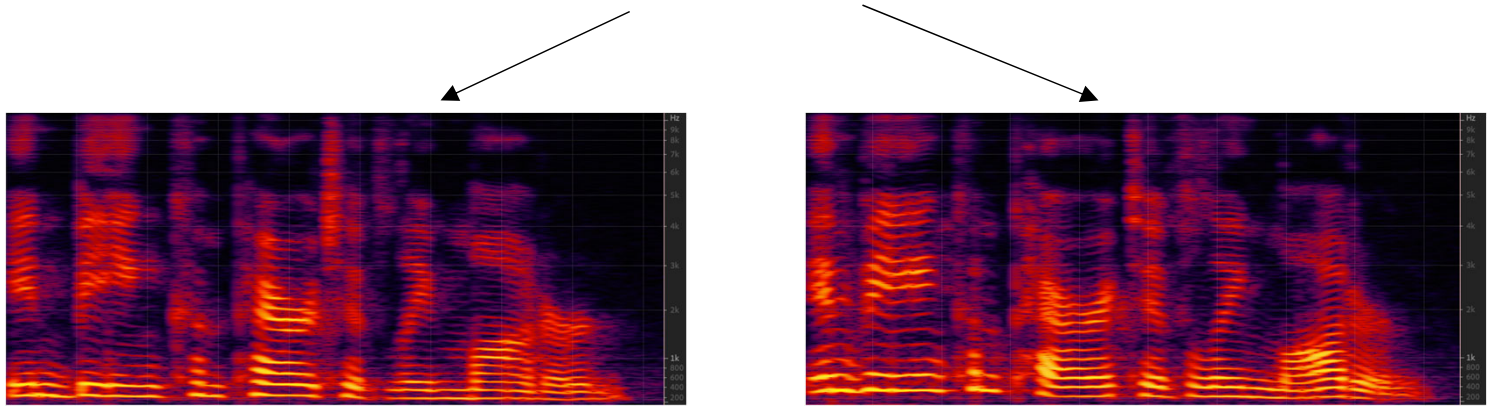


(a) 1.5x Voice Speed (b) 1.0x Voice Speed (c) 0.5x Voice Speed

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*".

One-to-Many Mapping Problem In TTS

- In being comparatively modern.



- Speech variance: pitch, energy, duration, timbre…

How to Solve One-to-Many Mapping?

- In FastSpeech:
  - Provide more information as input
    - Duration from teacher model
  - Simplify the data distribution in output
    - Knowledge distillation

Disadvantages of FastSpeech

- Duration extracted from the teacher model
  - is not accurate enough
- Teacher-student distillation
  - is complicated and time-consuming
  - leads to information loss due to data simplification

FastSpeech 2: Fast and High-Quality End-to-End Text to Speech

ICLR 2021

Yi Ren[1]*, **Chenxu Hu**[1]*, **Xu Tan**[2], **Tao Qin**[2], **Sheng Zhao**[3], **Zhou Zhao**[1]†, **Tie-Yan Liu**[2]

[1]Zhejiang University
{rayeren,chenxuhu,zhaozhou}@zju.edu.cn

[2]Microsoft Research Asia
{xuta,taoqin,tyliu}@microsoft.com

[3]Microsoft Azure Speech
Sheng.Zhao@microsoft.com

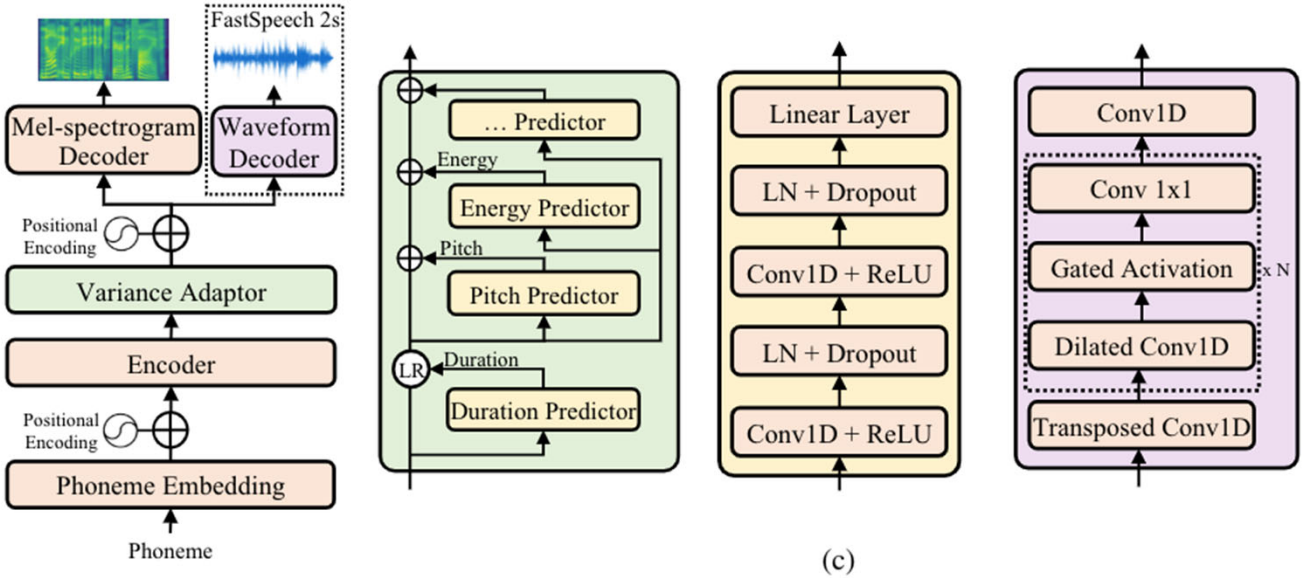
https://speechresearch.github.io/fastspeech2/

ByteDance

FastSpeech 2/2s

- FastSpeech 2: Solve One-to-Many Mapping
  - Introducing more variation information of speech
    - Pitch (CWT), energy and more accurate duration
  - For inaccurate duration in FastSpeech
    - introducing more accurate duration
  - For information loss in FastSpeech
    - Directly training the model with ground-truth target instead of the simplified output
- FastSpeech 2s
  - the first attempt to directly generate speech waveform from text in parallel

Model Architecture



(a) FastSpeech 2 (b) Variance adaptor (c) Duration/pitch/energy predictor (d) Waveform decoder

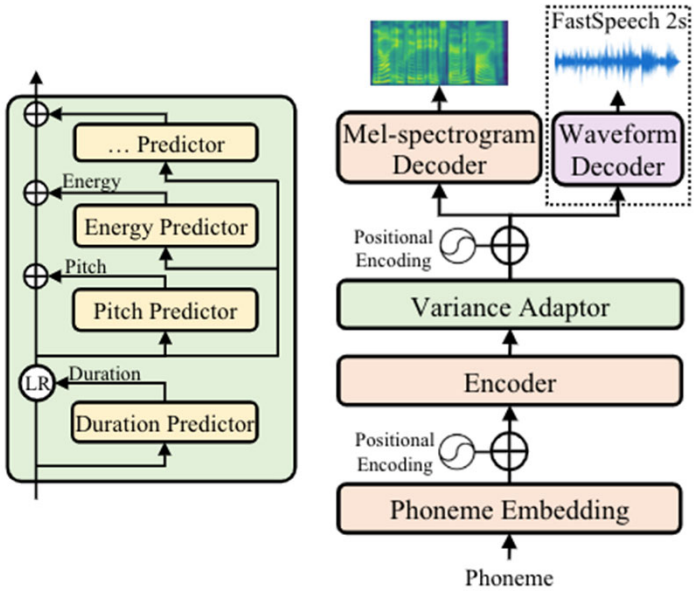
Figure 1: The overall architecture for FastSpeech 2 and 2s. LR in subfigure (b) denotes the length regulator proposed in FastSpeech. LN in subfigure (c) denotes layer normalization.

Variance Adaptor

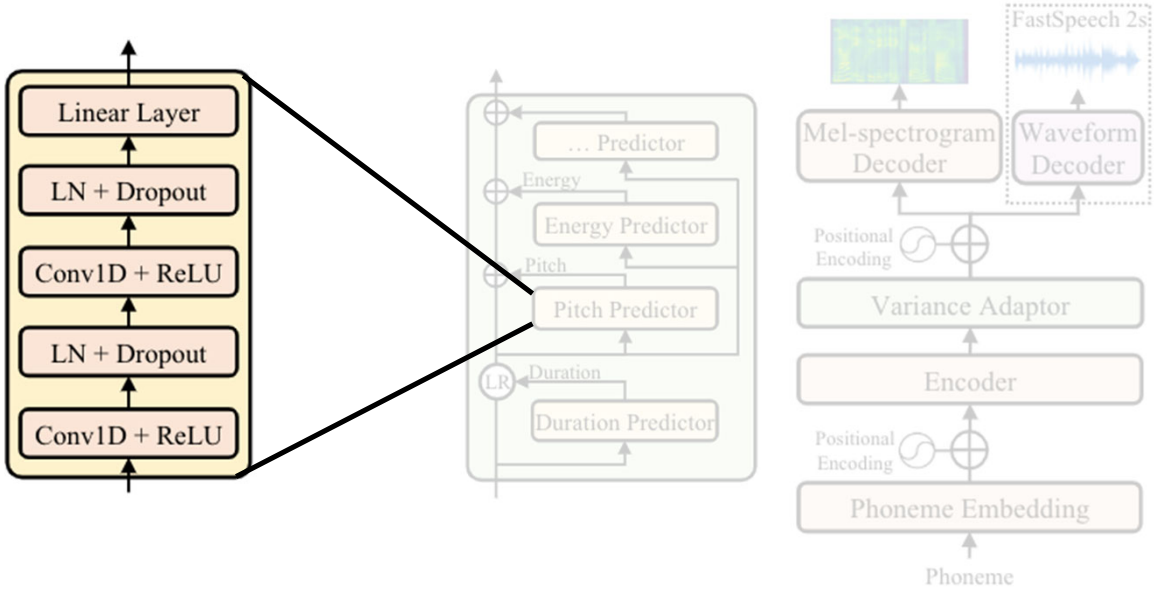
- **Phoneme duration:** how long the speech voice sounds

- **Pitch:** a key feature to convey emotions and greatly affects the speech prosody

- **Energy:** frame-level magnitude of mel-spectrograms and directly affects the volume and prosody of speech.



Variance Predictor



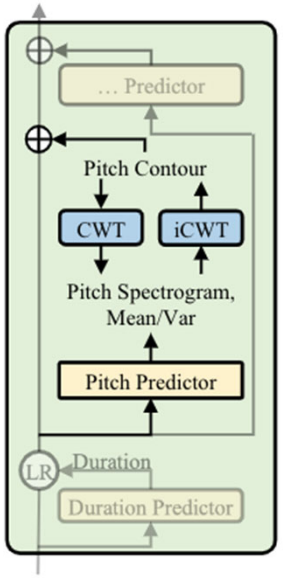
CWT Pitch Prediction

- **Motivation**: To better predict the variations in pitch contour
- **Idea**: Use continuous wavelet transform (CWT) to decompose the continuous pitch contour to pitch spectrogram.
- In training:

text input --fit--> **pitch spec** <--CWT-- **pitch contour**

- In Inference

text input --predict--> **pitch spec** --iCWT-> **pitch contour**



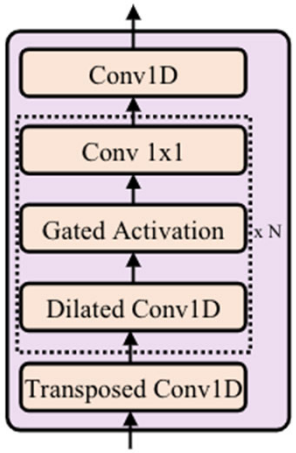
FastSpeech 2s
(Waveform Decoder)

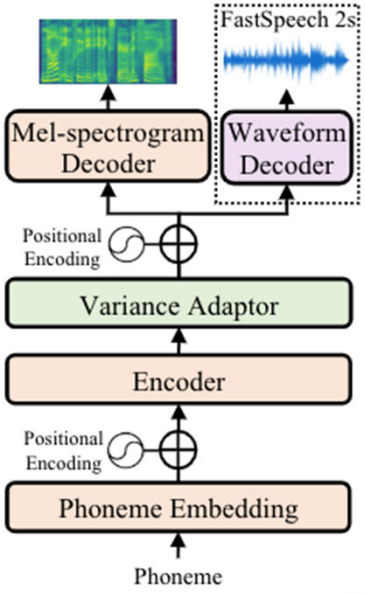
Challenges

- The information gap between the text and waveform is larger than that in text-to-spectrogram generation.
- Difficult to train on the audio clip that corresponds to the full text sequence due to the extremely long waveform samples and limited GPU memory.

FastSpeech 2s

- We adversarial training in the waveform decoder to force it to implicitly recover the phase information by itself.
- We leverage the mel-spectrogram decoder of FastSpeech 2, which is trained on the full text sequence to help on the text feature extraction.





Audio Quality Comparison

- FastSpeech 2 can surpass and FastSpeech 2s can match the voice quality of autoregressive models
- FastSpeech 2 outperforms FastSpeech
  - effectiveness of providing variance information

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

(a) The MOS with 95% confidence intervals.

Method	CMOS
FastSpeech 2	0.000
FastSpeech	-0.885
Transformer TTS	-0.235

(b) CMOS comparison.

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

Training Time and Inference Latency

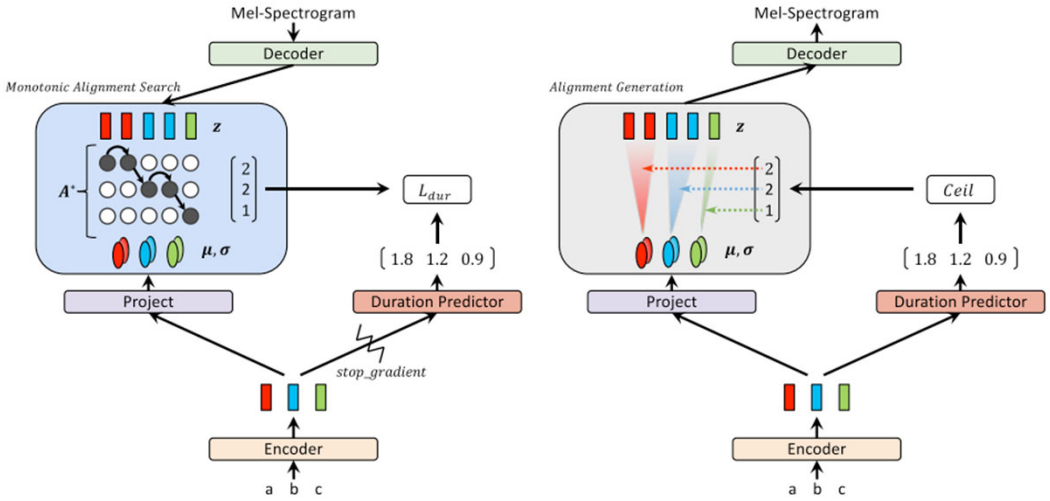
- FastSpeech 2 reduces the total training time compared with FastSpeech
- Inference Speed: FastSpeech 2s > FastSpeech 2 >> Transformer TTS

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**×

Table 2: The comparison of training time and inference latency in waveform synthesis. The training time of *FastSpeech* includes teacher and student training. RTF denotes the real-time factor, that is the time (in seconds) required for the system to synthesize one second waveform. The training and inference latency tests are conducted on a server with 36 Intel Xeon CPUs, 256GB memory, 1 NVIDIA V100 GPU and batch size of 48 for training and 1 for inference. Besides, we do not include the time of GPU memory garbage collection and transferring input and output data between the CPU and the GPU. The speedup in waveform synthesis for FastSpeech is larger than that reported in Ren et al. (2019) since we use Parallel WaveGAN as the vocoder which is much faster than WaveGlow.

Glow-TTS

- Generative
- Flow
- Strengths
  - Automatically alignment
  - High-quality outputs
- Weakness
  - Slow training
  - Large model size
  - Unstable training in multi-speaker datasets



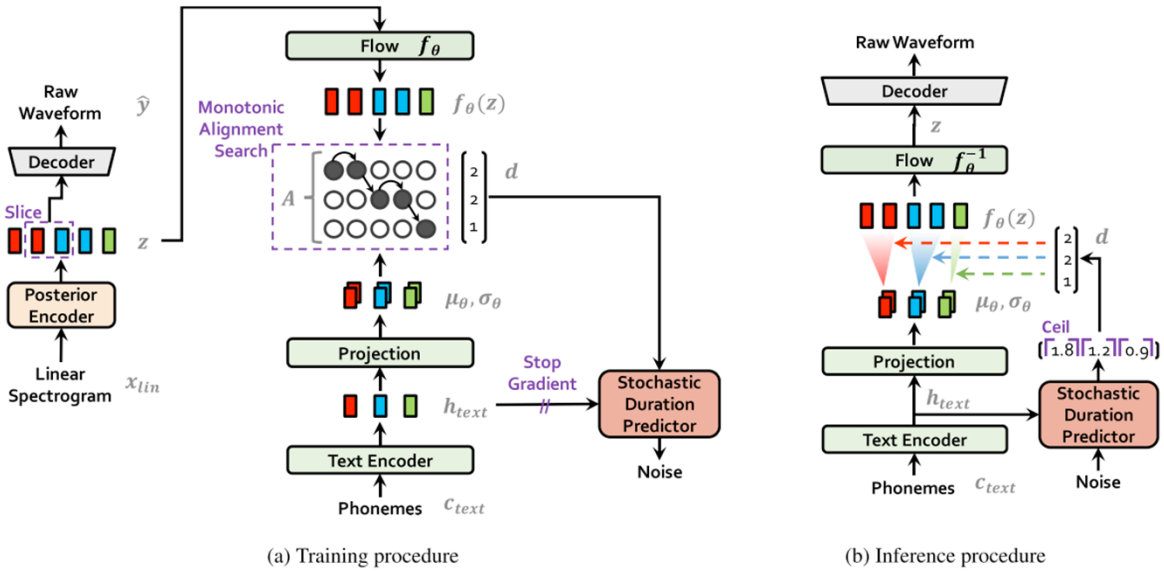
(a) An abstract diagram of the training procedure. (b) An abstract diagram of the inference procedure.

ByteDance

Glow-TTS: A generative flow for text-to-speech via monotonic alignment search

VITS

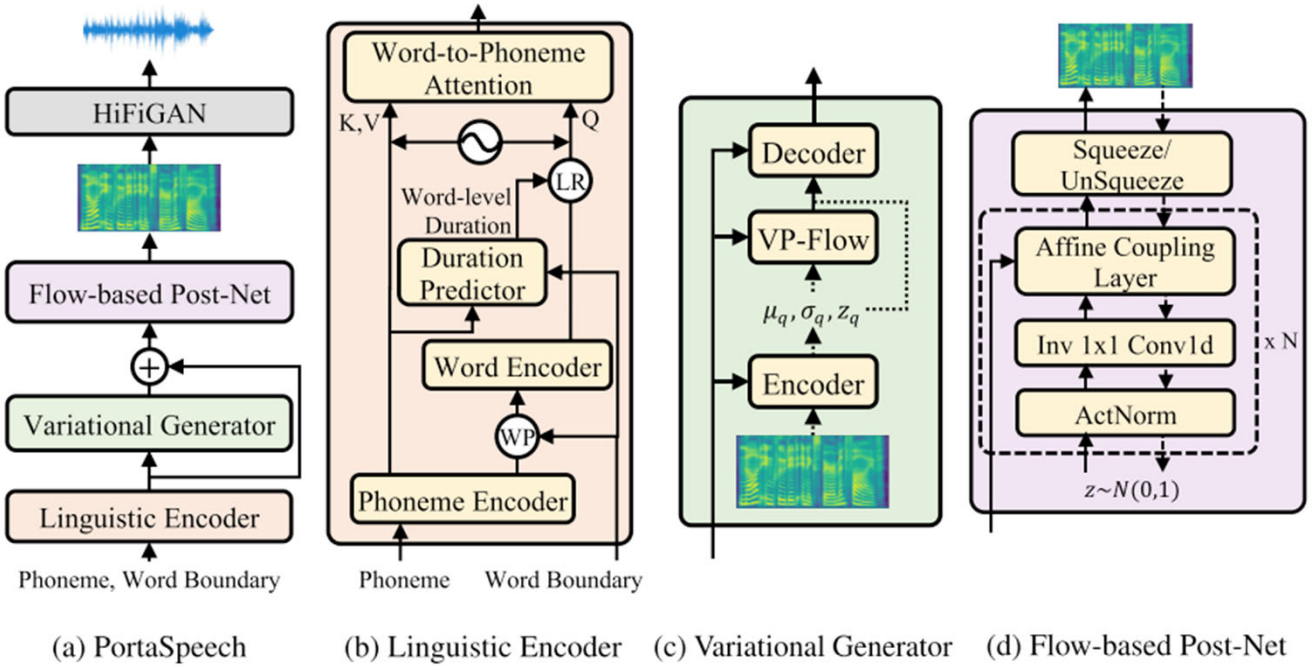
- Generative
- VAE+Flow+GAN
- Strengths
  - Automatically alignment
  - High-quality outputs
  - End2end model
- Weakness
  - Slow training



Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech

PortaSpeech

- Generative
- VAE+Flow
- Strengths
  - Small model size
  - High-quality outputs
  - Good prosody
- Weakness
  - Complicated model design



(a) PortaSpeech (b) Linguistic Encoder (c) Variational Generator (d) Flow-based Post-Net

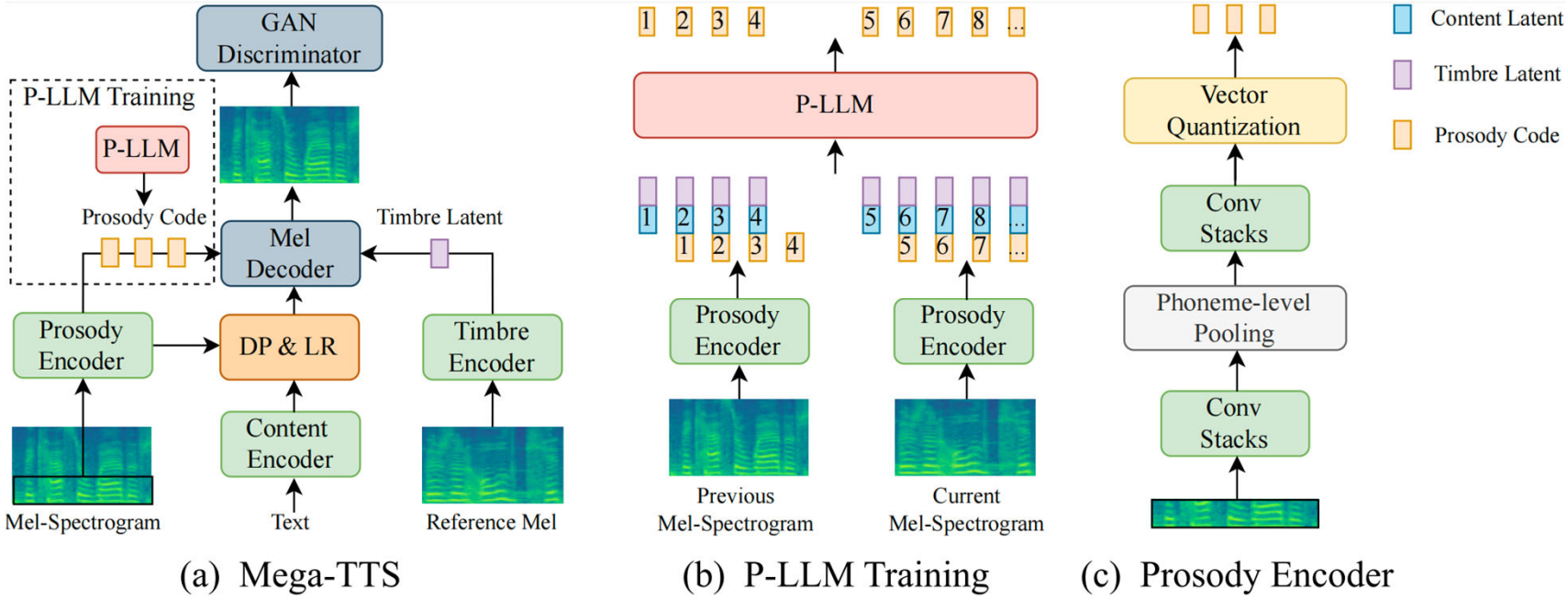
ByteDance

PortaSpeech : Portable and High-Quality Generative Text-to-Speech

MegaTTS Demo

- https://mega-tts.github.io/mega2_demo/

MegaTTS

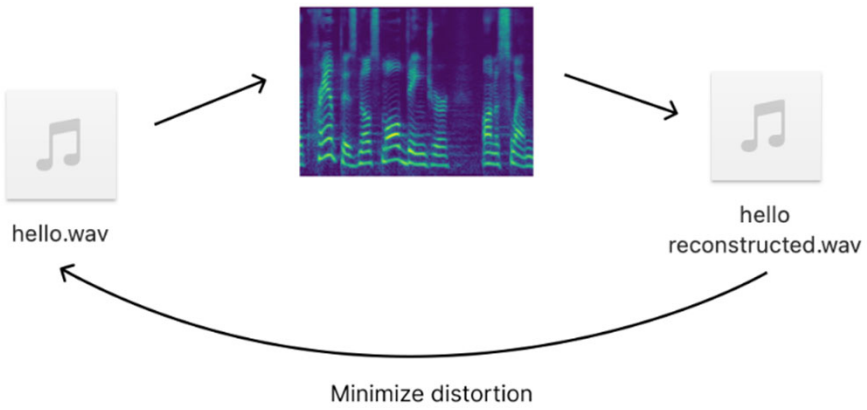


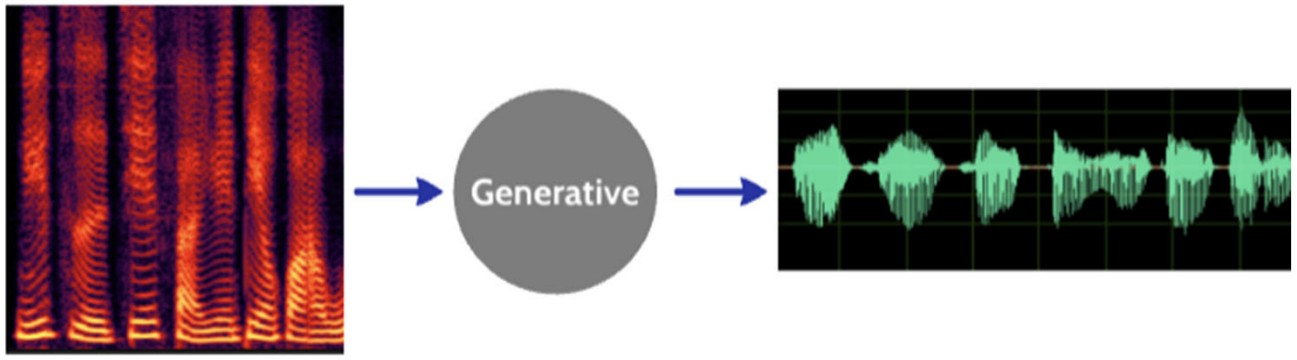
Mega-TTS Zero-Shot Text-to-Speech at Scale with Intrinsic Inductive Bias
https://mega-tts.github.io/demo-page/ https://mega-tts.github.io/mega2_demo/

ByteDance

Vocoders

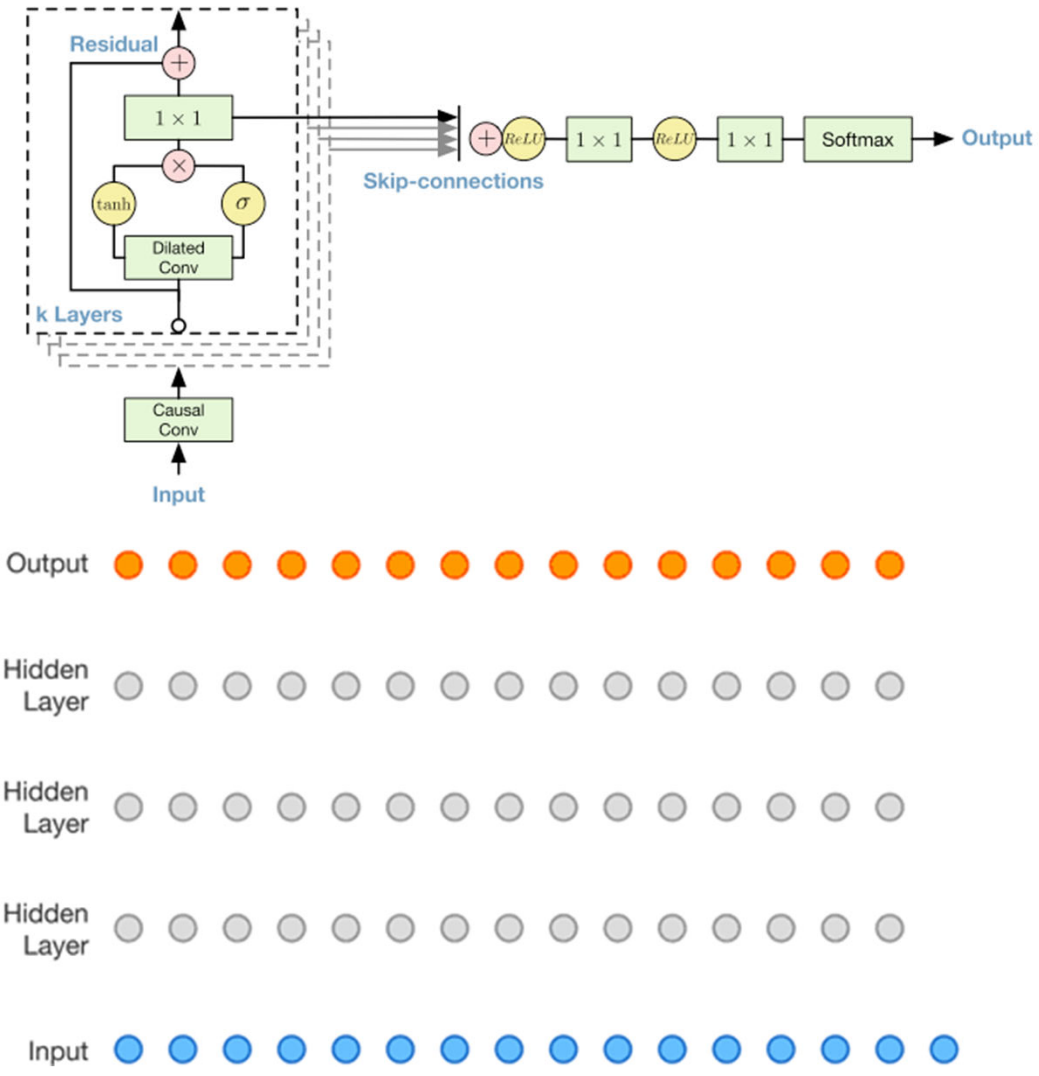
- WaveNet (AR)
- WaveRNN (AR)
- WaveGlow (Flow)
- Parallel WaveGAN (GAN)
- Hifi-GAN (GAN)





WaveNet

- Generative
- Autoregressive
- Strengths
  - High-quality outputs
  - Simple and easy to implement
  - Generalizable to other domains
- Weakness
  - Extremely slow



WaveGlow

- Generative
- Autoregressive
- Strengths
  - High-quality outputs
  - Simple and easy to implement
  - Generalizable to other domains
- Weakness
  - Extremely slow

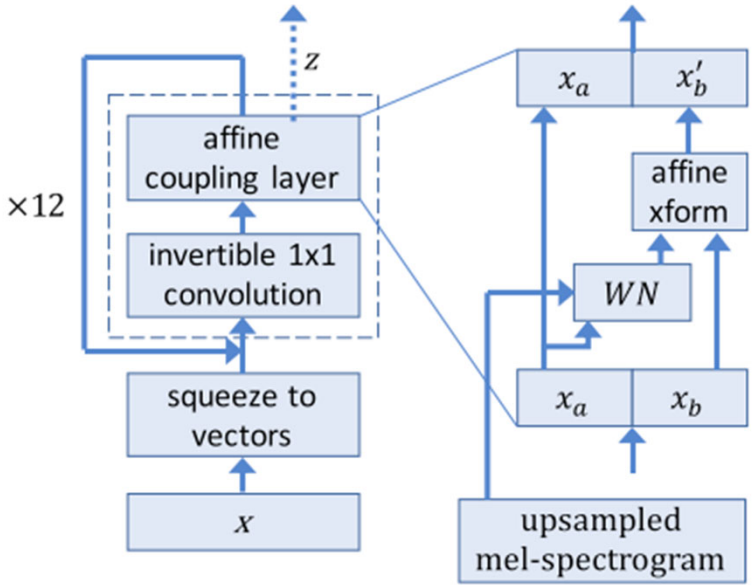
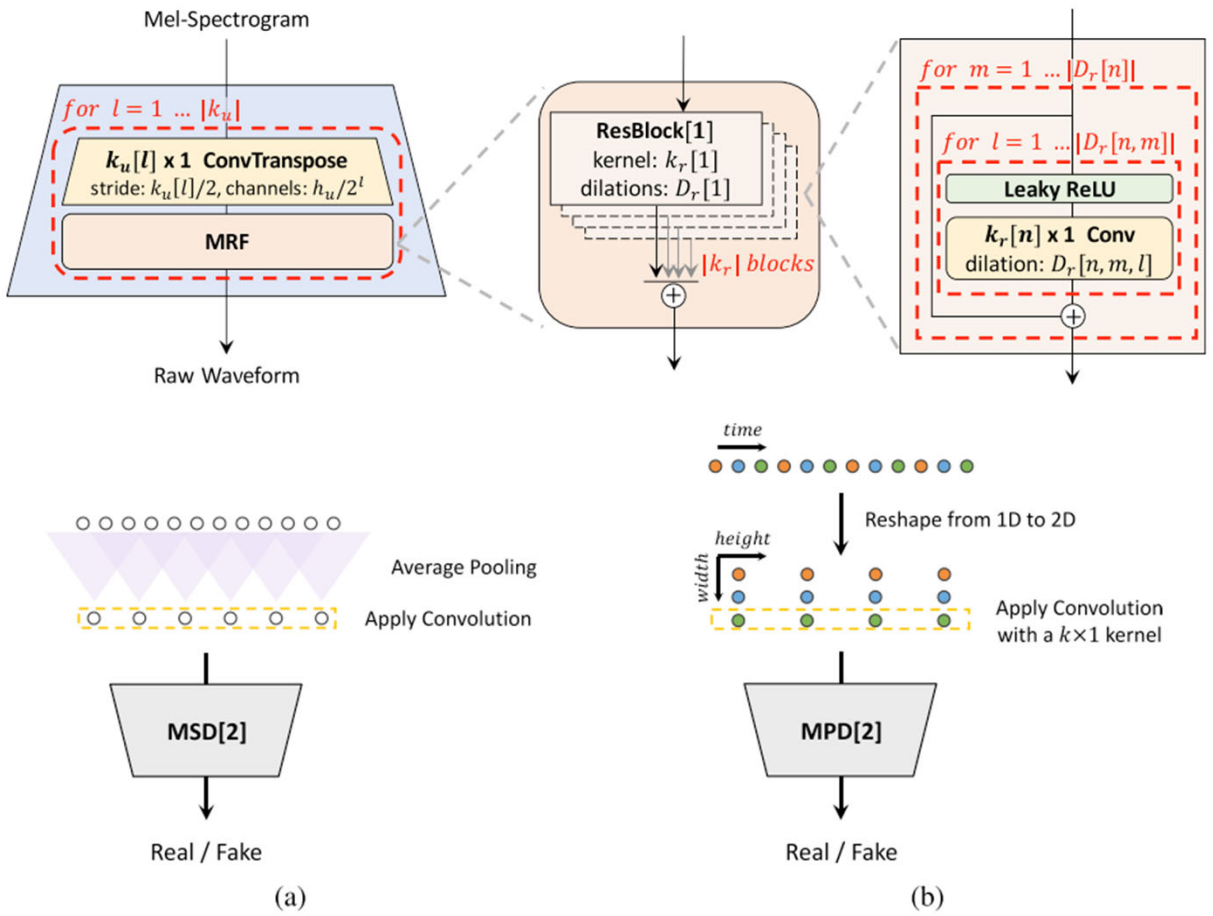


Fig. 1: WaveGlow network

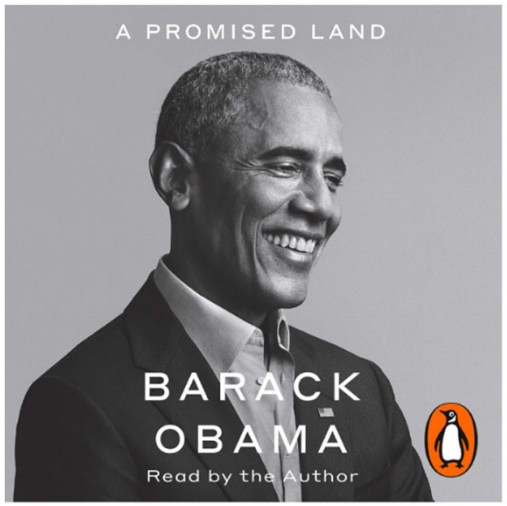
Hifi-GAN

- Generative
- GAN
- Most popular vocoder currently
- Strengths
  - High-quality outputs
  - Small model size
  - Very fast inference speed



ByteDance

Training your TTS model



- Step 0: Choose an open-source TTS implementation
  - There are many good implementations out there.
    - https://github.com/NATSpeech/NATSpeech
    - https://github.com/ming024/FastSpeech2
    - https://github.com/jaywalnut310/vits
    - https://github.com/NVIDIA/tacotron2
    - https://github.com/coqui-ai/TTS
- Step 1: Prepare our data
  - You can utilize public datasets, such as LJSpeech, or other data sourced from the internet.
- Step 2: Format the dataset
  - Each repository follows its unique protocol and file structure for the training dataset. Please follow their documents/README.

Foundation of Singing Voice Synthesis (SVS) and Generative SVS

ByteDance 字节跳动

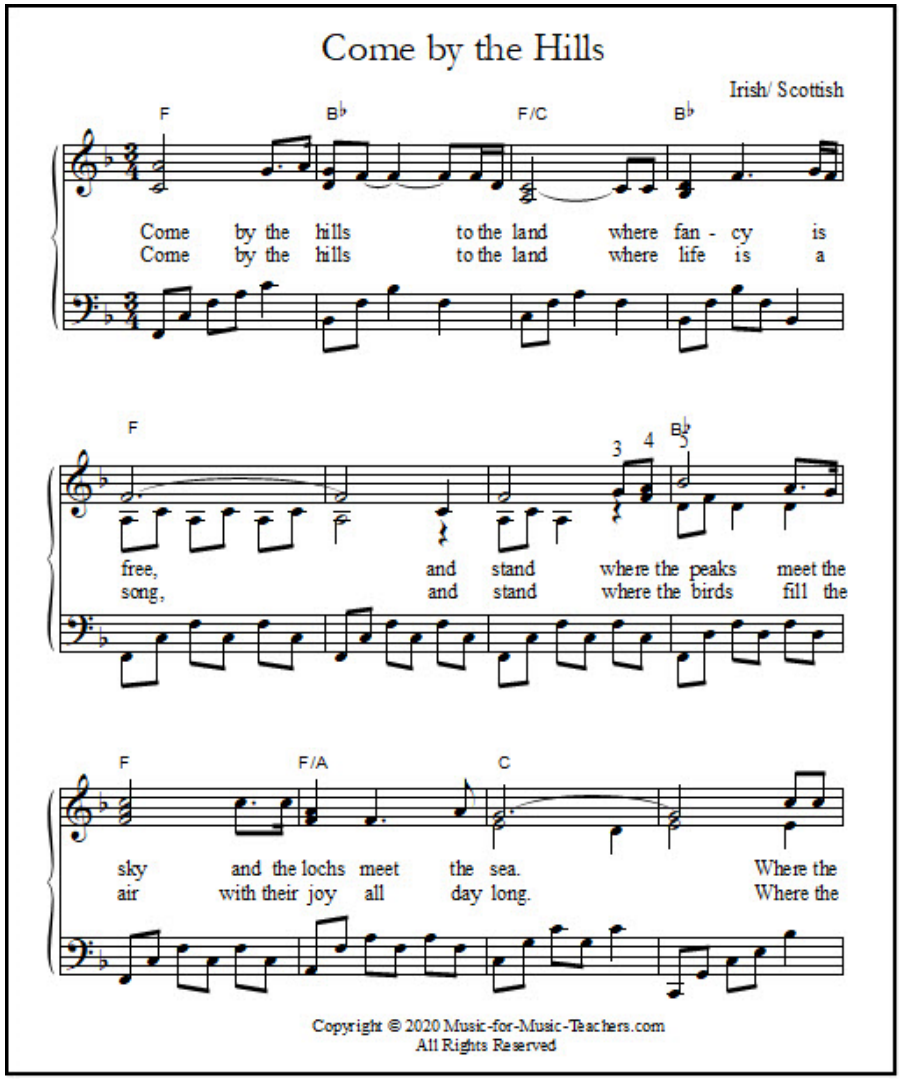
DiffSinger Demo (Mandarin)

- https://www.bilibili.com/video/BV1be411N7JA

DiffSinger Demo (Polish/Japanese)

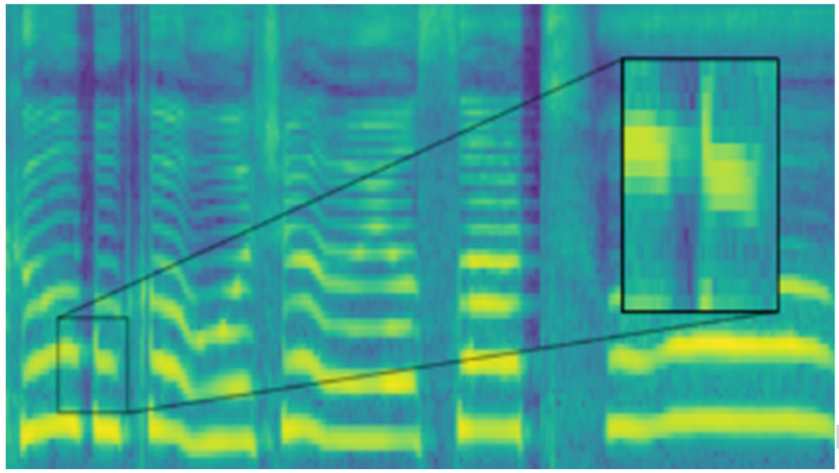
Music Score

- Information in a music score:
  - Note
    - Note pitch
    - Note duration
  - Lyrics
    - Lyrics duration
  - Alignment
    - Note – lyrics alignment
  - Tempo
  - Pause/rest/silence
  - Strength/Dynamics

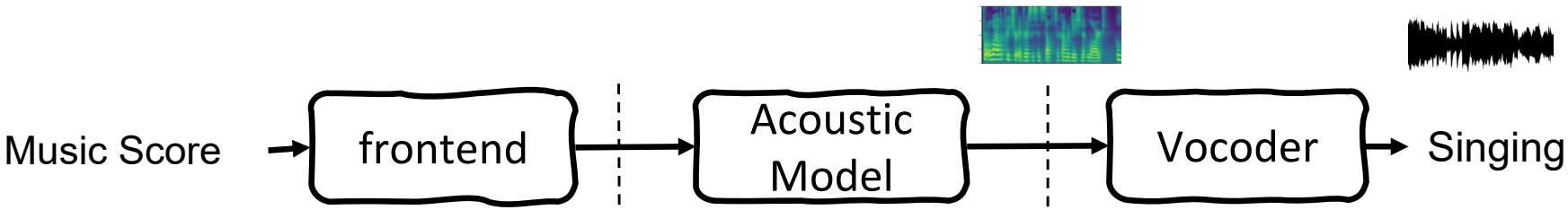


Challenges in Singing Voice Synthesis

- High sampling rate (>44.1kHz)
  - High spectrogram resolution
- Singing skills
  - vibrato/portamento/falsetto
- Artistic with very high standards for aesthetics
- Limited open-sourced dataset; more expensive to record than speech
  - Professional singer
  - Professional studio
  - Professional data annotation



Modern End2end SVS Framework



Score (note+lyrics)-> Linguistic

- Text Normalization
- Grapheme to Phoneme
- Note-phoneme alignment

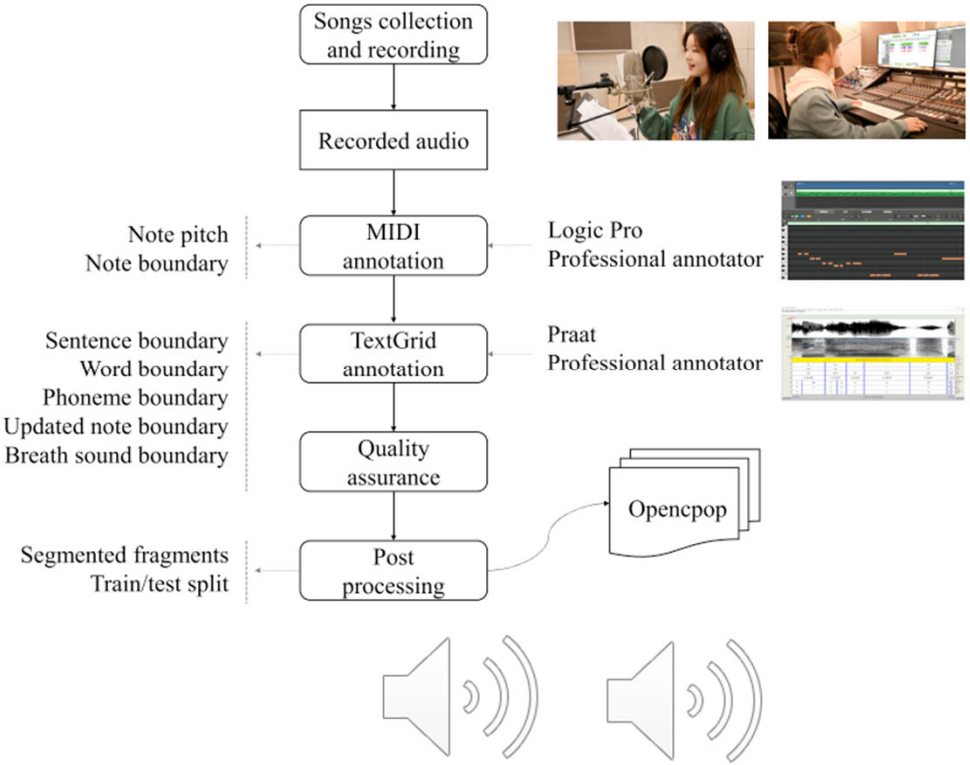
Linguistic feature -> Acoustic

- Diffsinger
- ……

Acoustic features→ Speech

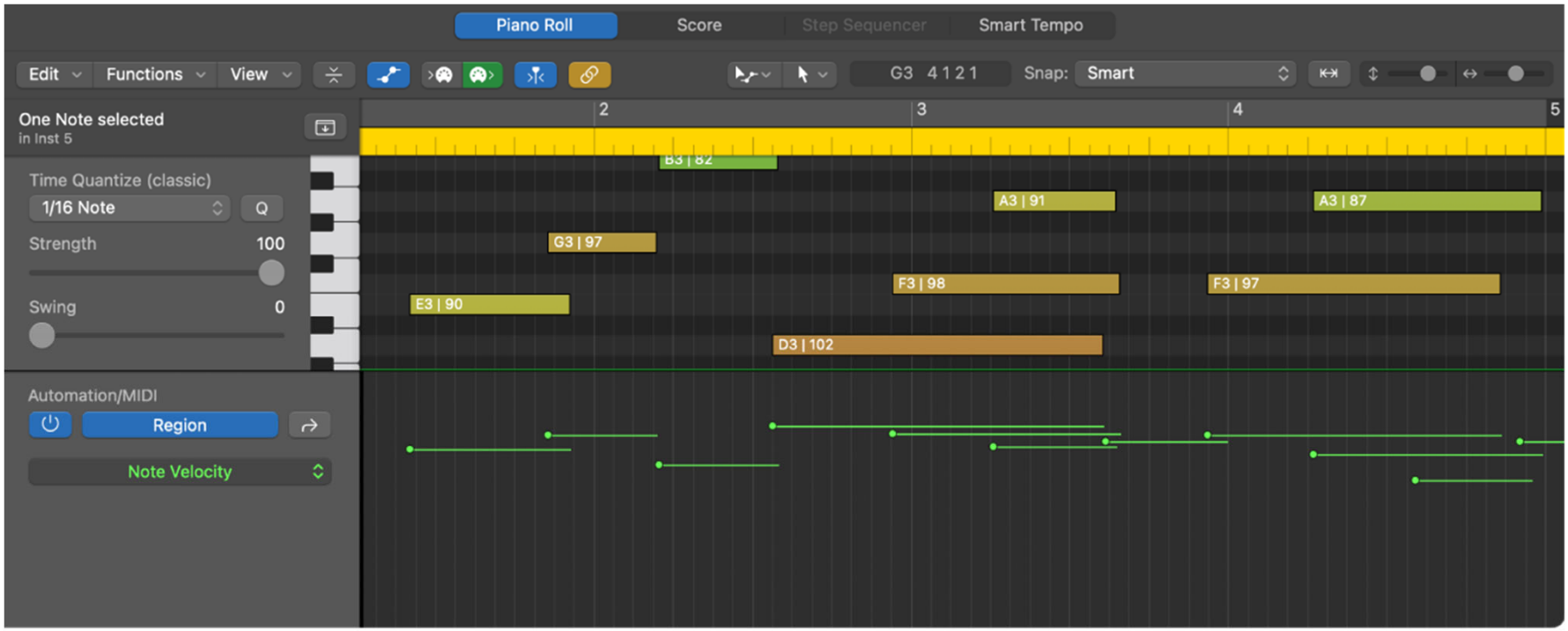
- WaveNet/Parallel WaveNet
- WaveGlow/FloWaveNet
- WaveRNN/LPCNet
- HIFIGAN

Dataset: Opencpop



Opencpop: A High-Quality Open Source Chinese Popular Song Corpus for Singing Voice Synthesis,
https://wenet.org.cn/opencpop/

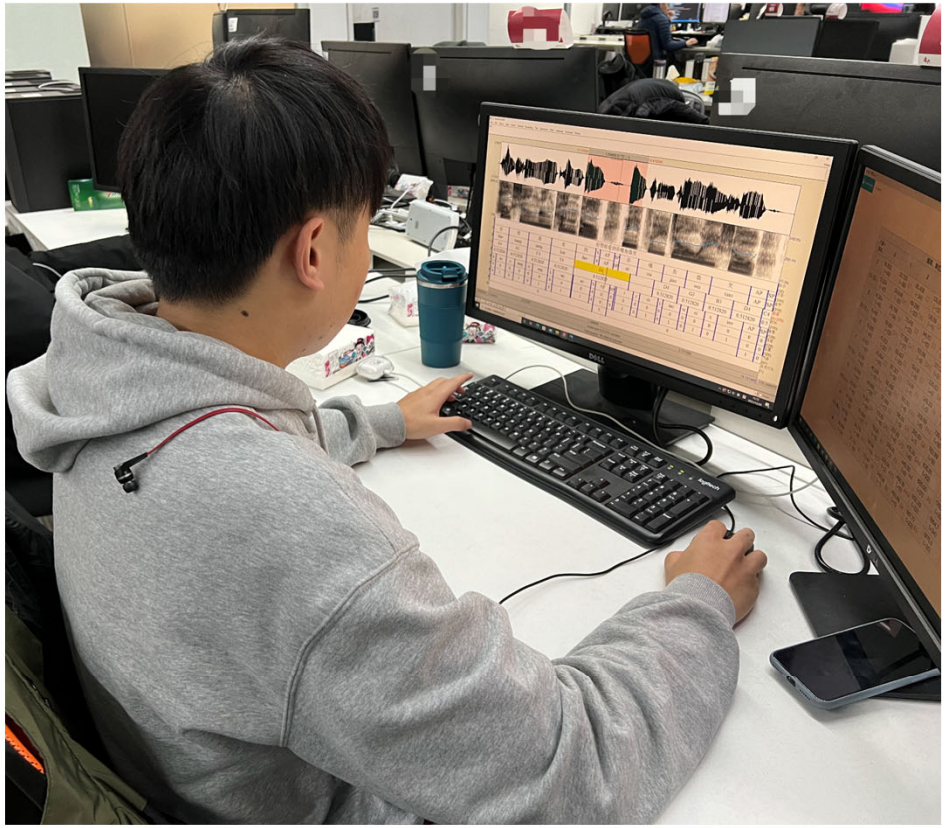
MIDI / Notes



Dataset: Opencpop

	点燃所有希望										
(a)	点	燃		AP	所	有	希	望	_	SP	AP
(b)	dian	ran		AP	suo	you	xi	wang	_	SP	AP
(c)	B3	G#4		rest	B3	B3	B3	E4	F#4	rest	rest
(d)	0.605	0.942		0.343	0.421	.261	0.831	.227	0.841	0.832	0.402
(e)	ian	r	an	AP	s	ou	y ou	x	i	ang	ang	SP	AP
(f)	0	0	0	0	0	0	0 0	0	0	0	1	0	0

(a) Chinese character; (b) syllable; (c) note; (d) duration; (e) phoneme; (f) is slur (1 means yes)



Opencpop: A High-Quality Open Source Chinese Popular Song Corpus for Singing Voice Synthesis,
https://wenet.org.cn/opencpop/

ByteDance

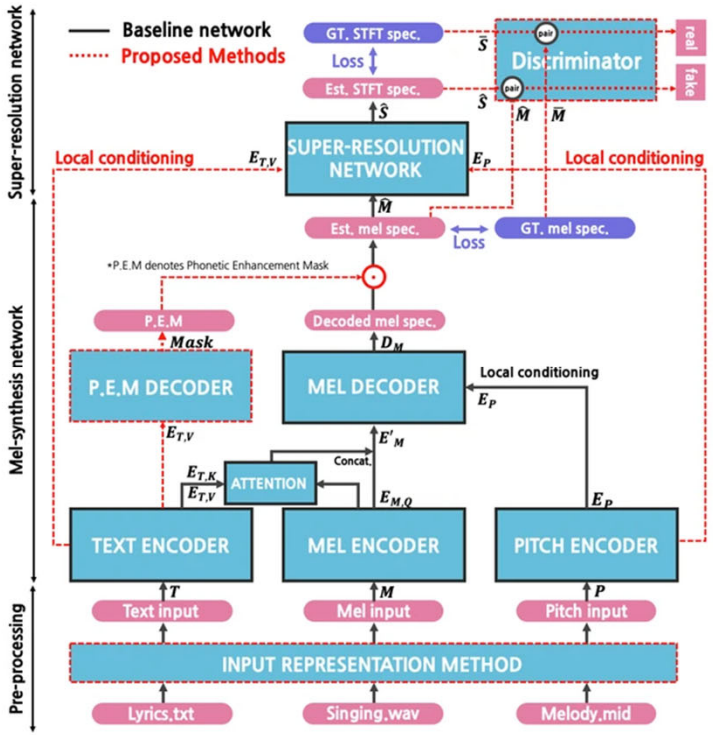
Dataset: Opencpop

- **100 unique Mandarin songs**, which were recorded by a **professional female singer**.
- Recorded with studio-quality at a sampling rate of **44,100 Hz** in a **professional recording studio environment**.
- All singing recordings have been phonetically annotated with utterance/note/phoneme boundaries and pitch types.
- Contains **3,756 utterances**, with a total of about **5.2 hours**.
- The testing set consists of 5 randomly chosen songs, and baseline synthesized results are provided.

Opencpop: A High-Quality Open Source Chinese Popular Song Corpus for Singing Voice Synthesis,
https://wenet.org.cn/opencpop/

Adversarially Trained End-to-end Korean Singing Voice Synthesis System

- Generative
- GAN
- Strengths
  - High-quality outputs
- Weakness
  - Slow inference



Variable description	Operation description
T : Input Text $E_{T,K}$: Encoded Text key P : Input Pitch $E_{T,V}$: Encoded Text value M : Input Mel spec. E_P : Encoded Pitch D_M : Decoded Mel E_M : Encoded Mel Att : Attention between Text and Mel $Mask$: phonetic enhancement mask $\hat{M}$: $D_M \times Mask$, Estimated Mel $\hat{S}$: Estimated linear spec. M : Ground truth Mel S : Ground truth linear spec.	$relu/\sigma/smax$: rectified linear, sigmoid, softmax activation unit el : Embedding Lookup table do : dropout with rate 0.95 $C^{o}_{k,d}$: 1d convolution with kernel size k, dilation d, and output dim o. $C2^{o}_{w,h}$: 2d convolution with kernel w,h, output dim o. $DC^{o}_{k,s}$: 1d de-convolution with kernel size k, strides s, output dim o. $FC(o)$: fully connected layer with output dim o. $mm(X)$: matrix multiplication with matrix X GP : global average pooling $AP_{w,h}$: average pooling with kernel w,h $ip(X)$: inner product with X $\|^{n}$: repeat same operation n times $[:n]$: slice tensor to n_{th} channel $[n:]$: slice tensor from n_{th} channel

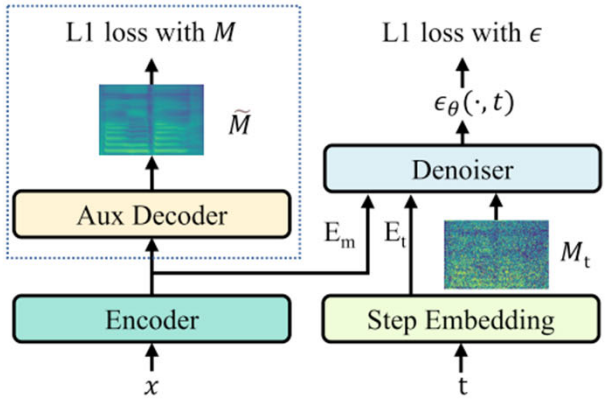
Basic units	
Highway Gated Conv. unit	$X\|hw^{o}_{n}\| := \{X\|mm(1-H_1\|\sigma)\} + \{H_2\|mm(H_1\|\sigma)\}$ (where $H_1 = X\|C^{2*o}_{3,n}\|[:o], H_2 = X\|C^{2*o}_{3,n}\|[o:]$) $(X\|hw_{a-b}\| := X\|hw_a\|hw_b\|)$
Residual Conv. unit	$X\|RC^{o}\| := [\{X\|C2^{o}_{3,3}\|C2^{o}_{3,3}\|\} + \{X\|C2^{o}_{3,3}\|C2^{o}_{3,3}\|C2^{o}_{1,1}\|\}]\|AP_{2,2}\|$
Mel-synthesis network : $\hat{M} = MS(M,T,P)$	
Pitch Enc	$P\|el\|C^{512}_{1,1}\|relu\|do\|C^{512}_{1,1}\|do\|hw^{512}_{1-3-9-27}\|^2\|(hw^{512}_{1}\|do)\|^2 = E_P$
Text Enc	$T\|el\|C^{512}_{1,1}\|relu\|do\|C^{512}_{1,1}\|do\|hw^{512}_{1-3-9-27}\|^2\|(hw^{512}_{1}\|do)\|^2 = [E_{T,K}, E_{T,V}]$
Mel Enc	$M\|(C^{256}_{1,1}\|relu\|do)^2\|C^{256}_{1,1}\|do\|hw^{256}_{1-3-9-27}\|^2\| = E_{M,Q}$
Attention	$E_{M,Q}\|mm\,(E_{T,K})/\sqrt{256}\|smax\| = Att,\ \ Att\|mm(E_{T,V})\|concat(E_M)\| = E'_M$
Mel Dec	$E'_M\|C^{256}_{1,1}\|do\|$ ┬ $C^{256}_{1,1}\| + \{E_P[:256]\}\|\sigma$ / $C^{256}_{1,1}\| + \{E_P[256:]\}\|relu$ ┴ $\|hw^{256}_{1-3-9-27-1-1}\|(C^{256}_{1,1}\|relu\|do)^3\|\sigma\| = D_M$
P.E.M Dec	$E_{T,V}\|C^{512}_{1,1}\|relu\|do\|C^{80}_{1,1}\|do\|hw^{80}_{1-3-9-27}\|^2\|C^{80}_{1,1}\|\sigma\| = Mask$
Super-resolution network : $\hat{S} = SR(\hat{M}, E_{T,V}, E_P)$	
Super Resolution Network	$\hat{M}$ ┬ $C^{512}_{1,1}\|do\| + \{E_P\|C^{512}_{1,1}\|do\|\} + \{E_{T,V}\|C^{512}_{1,1}\|do\|\}\|\sigma$ / $C^{512}_{1,1}\|do\| + \{E_P\|C^{512}_{1,1}\|do\|\} + \{E_{T,V}\|C^{512}_{1,1}\|do\|\}\|relu$ ┴ $\|hw^{512}_{1-3}$ $\|(DC^{512}_{2,2}\|hw^{512}_{1-3})\|^2\|C^{1024}_{1,1}\|hw^{1024}_{1-1}\|C^{513}_{1,1}\|(C^{513}_{1,1}\|relu)\|^2\|C^{513}_{1,1}\|\sigma\| = \hat{S}$
Discriminator	$\hat{S}\|C2^{64}_{7,3}\|RC_{64}\|RC_{128}$ ┬ $\|RC_{256}\|RC_{512}\|RC_{1024}\|RC_{128}\|relu\|GP\|FC(1)$ / $C2^{1}_{3,3}\|ip(\hat{M}\|C^{128}_{1,1}\|)$ ┴ $= Fake$
	$S\|C2^{64}_{7,3}\|RC_{64}\|RC_{128}$ ┬ $\|RC_{256}\|RC_{512}\|RC_{1024}\|RC_{128}\|relu\|GP\|FC(1)$ / $C2^{1}_{3,3}\|ip(M\|C^{128}_{1,1}\|)$ ┴ $= Real$

- [Demo page: https://ksinging.mystrikingly.com/](https://ksinging.mystrikingly.com/)

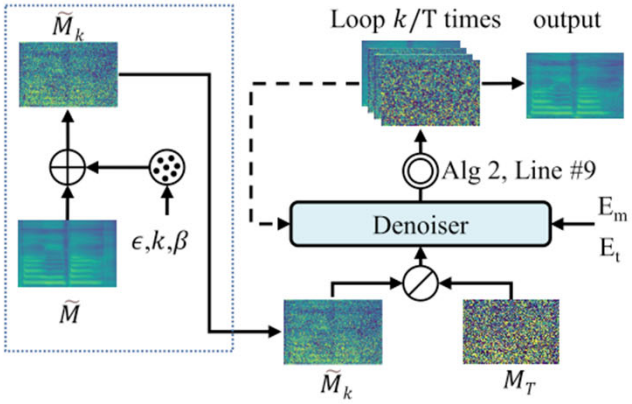
ByteDance

DiffSinger

- Generative
- Diffusion
- Strengths
  - High-quality outputs
- Weakness
  - Slow inference



(a) The training procedure of DiffSinger.



(b) The inference procedure of DiffSinger.

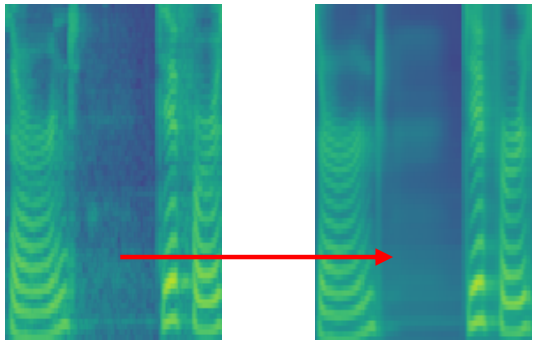
- [Demo page: https://huggingface.co/spaces/Silentlin/DiffSinger](https://huggingface.co/spaces/Silentlin/DiffSinger)

ByteDance

DiffSinger: Singing Voice Synthesis via Shallow Diffusion Mechanism

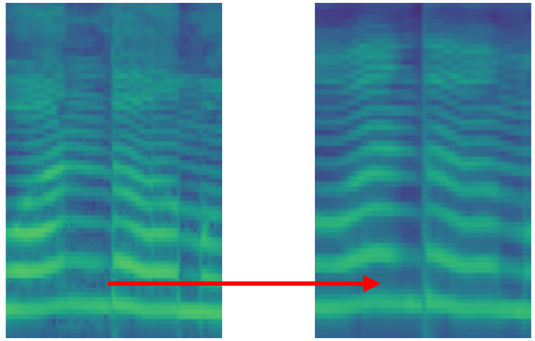
DiffSinger

- Blurry non-speech part



Real	Previous	Ours

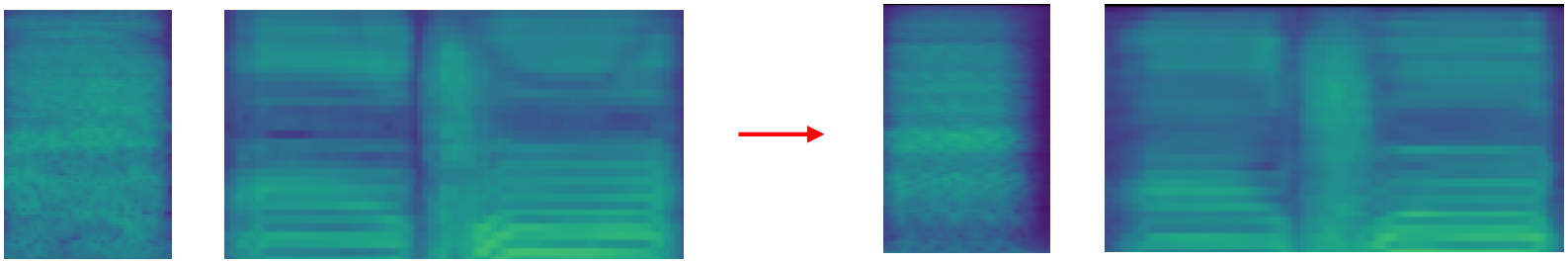
- Details between harmonics vanishing



DiffSinger

- GAN's problem

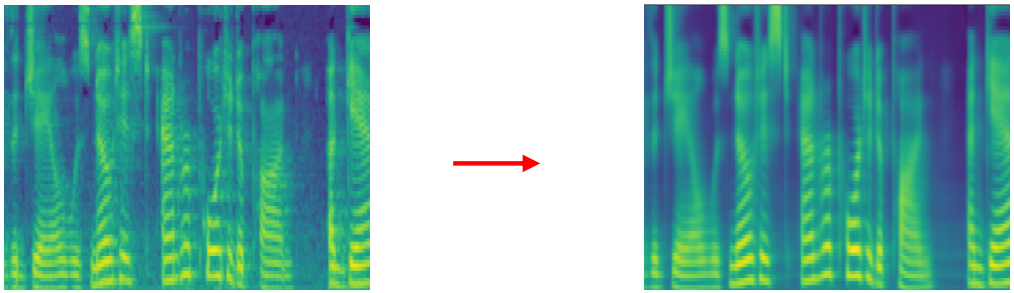
The latent variable doesn't work in conditional GAN [1], causing repeated patterns and a lack of diversity.



Real Predicted

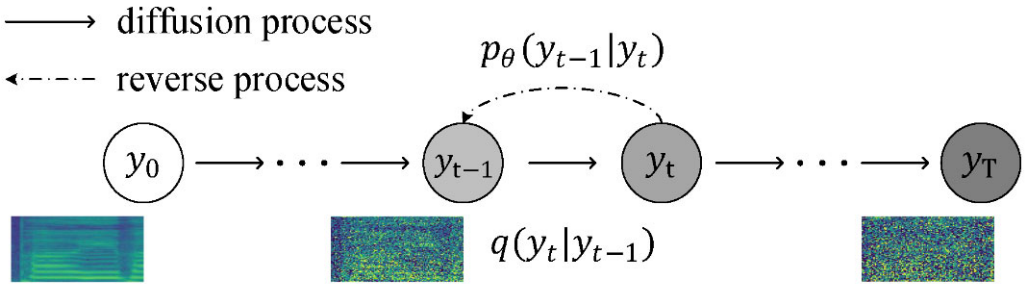
- VAE's problem

The blurry prediction problem still exists.



Real Predicted

DiffSinger



- Forward

$$q(\mathbf{y}_t|\mathbf{y}_{t-1}) := \mathcal{N}(\mathbf{y}_t; \sqrt{1-\beta_t}\mathbf{y}_{t-1}, \beta_t\mathbf{I}).$$

$$q(\mathbf{y}_t|\mathbf{y}_0) = \mathcal{N}(\mathbf{y}_t; \sqrt{\bar{\alpha}_t}\mathbf{y}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

- Reverse (network with parameter θ)

$$p_\theta(\mathbf{y}_{t-1}|\mathbf{y}_t) := \mathcal{N}(\mathbf{y}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{y}_t, t), \sigma_t^2\mathbf{I}).$$

- Inference.

Run the reverse procedure iteratively

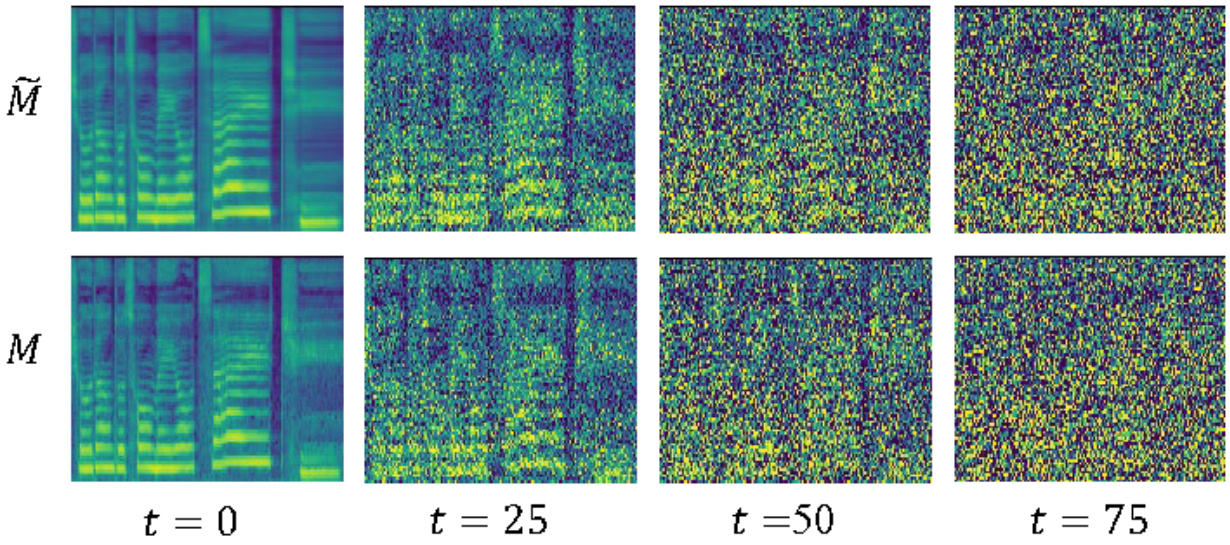
for $t = k, k-1, ..., 1$ **do**
  if $t = 1$ **then** $\mathbf{z} = \mathbf{0}$;
  else Sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
  $M_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(M_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(M_t, x, t)\right) + \sigma_t\mathbf{z}$
end

How to prevent too many iterations?

DiffSinger

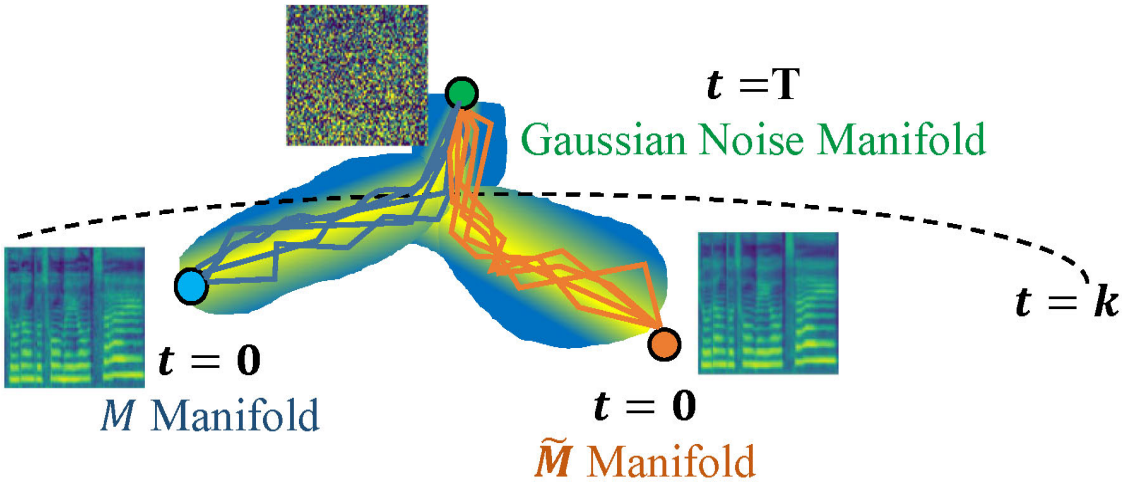
- An interesting phenomenon

$\tilde{M}$: Fastspeech2 optimized by L1/L2 loss M : Ground-Truth



DiffSinger

- Shallow Diffusion Mechanism

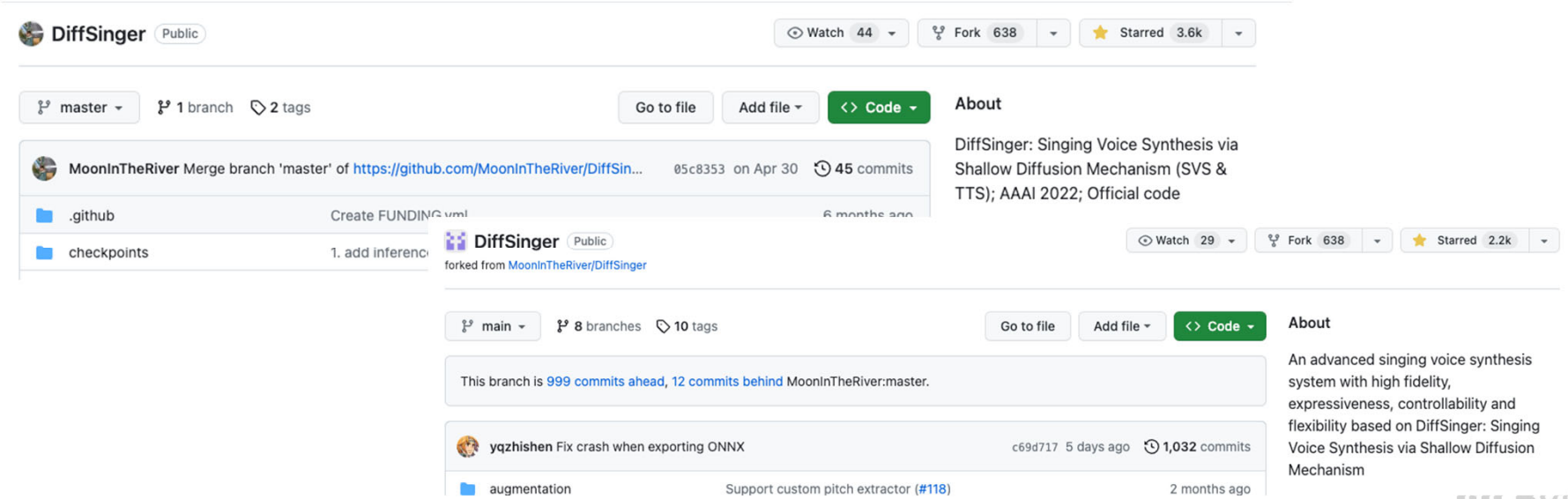


$$D_{KL}(\mathcal{N}(M_t)||\mathcal{N}(\widetilde{M}_t)) = \frac{\bar{\alpha}_t}{2(1-\bar{\alpha}_t)}\|\widetilde{M}_0 - M_0\|_2^2$$

$$\begin{aligned}
&\mathbb{E}_{M\in\mathcal{Y}'}\left[D_{\mathrm{KL}}\left(\mathcal{N}(M_t)\|\mathcal{N}(\widetilde{M}_t)\right)\right] \\
&= \mathbb{E}_{M\in\mathcal{Y}'}\left[\frac{\bar{\alpha}_t}{2(1-\bar{\alpha}_t)}\|\widetilde{M}_0 - M_0\|_2^2\right] \\
&\leq \mathbb{E}_{M\in\mathcal{Y}'}[D_{KL}(\mathcal{N}(M_T)||\mathcal{N}(\mathbf{0},\mathbf{I}))]
\end{aligned}$$

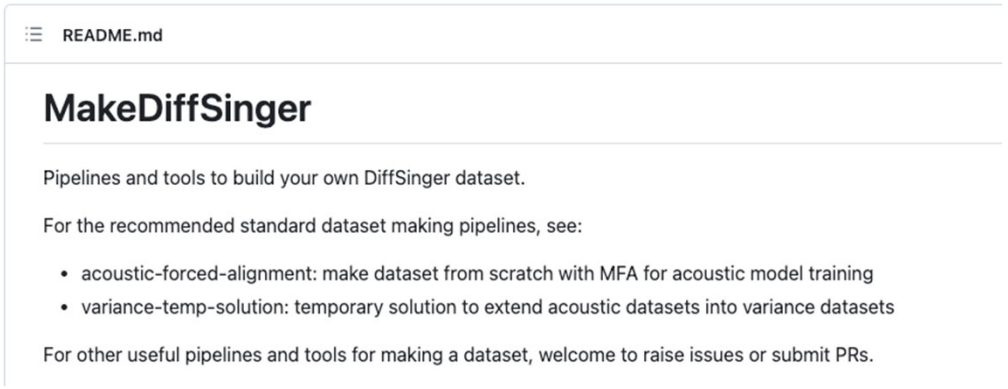
Training your SVS model

- Step 0: Choose an open-source TTS implementation. I recommend:
  - https://github.com/MoonInTheRiver/DiffSinger (original Diffsinger version)
  - https://github.com/openvpi/DiffSinger (the SVS community has made some improvements based on the original Diffsinger version)



Training your SVS model

- Step 1: Prepare our data
  - You can utilize public datasets, such as Opencpop,
    or other data sourced from the internet.
    https://wenet.org.cn/opencpop/
- Step 2: Format the dataset
  - Each repository follows its unique protocol and file structure for the training dataset. Please follow their documents/README.
    - https://github.com/openvpi/MakeDiffSinger


README.md

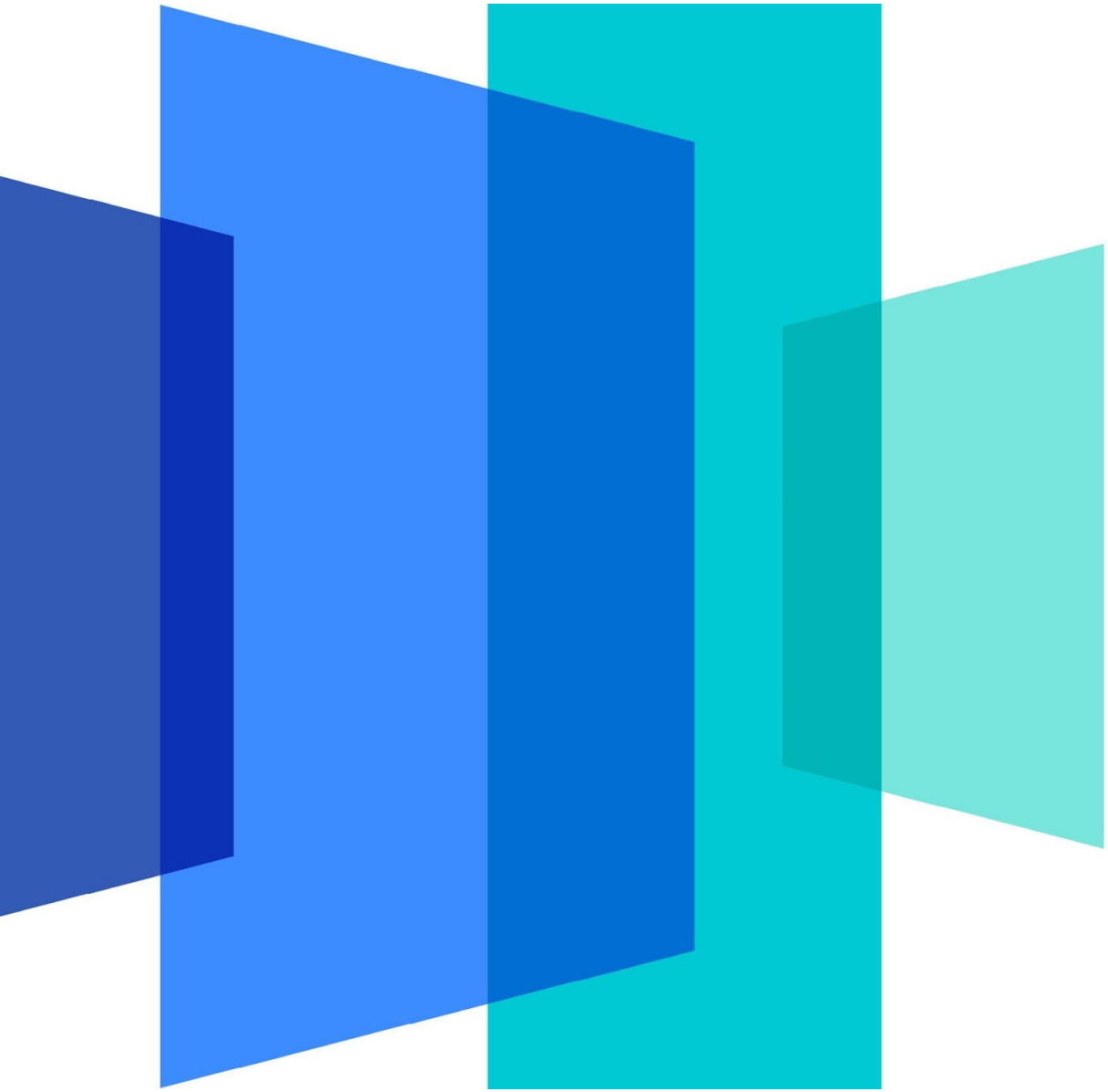
MakeDiffSinger

Pipelines and tools to build your own DiffSinger dataset.

For the recommended standard dataset making pipelines, see:

- acoustic-forced-alignment: make dataset from scratch with MFA for acoustic model training
- variance-temp-solution: temporary solution to extend acoustic datasets into variance datasets

For other useful pipelines and tools for making a dataset, welcome to raise issues or submit PRs.



THANKS

ByteDance