Towards Efficient Machine Learning for Speech and Music Applications

Xu Tan/谭旭
Senior Researcher
Microsoft Research Asia

xuta@microsoft.com

Background

- Machine learning and deep learning have achieved great success on CV, NLP, speech, and many other areas
- Key for deep learning
  - Data
  - Model
  - Computation
- For practical usage
  - Lack of training data
  - Limited storage and computation
  - Fast inference
  - Especially for language, speech, and music domains

Data/Memory/Computation/Time-Efficient machine learning is important

Techniques for efficient machine learning

- Autoregressive → Non-Autoregressive (Time-efficient)
  - NAR, Flow, GAN, VAE, Diffusion, etc
- Lightweight model (Time/Memory/Computation-efficient)
  - Pruning, quantization, knowledge distillation, neural architecture search
- Domain knowledge (Time/Memory/Computation-efficient)
  - Subscaling, subband, LPC, multi-frame prediction, speech/music knowledge
- Self-supervised/Transfer learning (Data-efficient)
  - Pre-training/fine-tuning, cross-lingual/speaker/style/domain transfer, etc
- Semi-supervised learning (Data-efficient)
  - Knowledge distillation, back transformation, etc

Outline

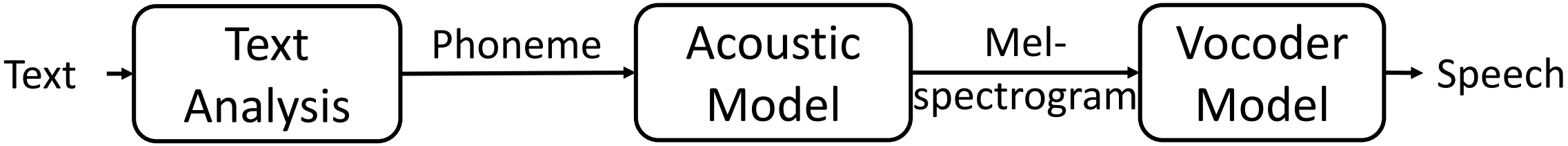
- Time-efficient
  - **FastSpeech 1/2** (TTS, NeurIPS 2019/ICLR 2021)
  - **FastCorrect 1/2** (ASR, paper submission)
  - **PriorGrad** (TTS, paper submission)
- Memory/Computation-efficient
  - **LightSpeech** (TTS, ICASSP 2021)
  - **AdaSpeech** (TTS, ICLR 2021)
- Data-efficient
  - **AdaSpeech 2/3** (TTS, ICASSP 2021/INTERSPECH 2021)
  - **LRSpeech** (TTS/ASR, ICML 2019/KDD 2020)
  - **MixSpeech** (ASR, ICASSP 2021)
  - **SongMASS** (Music, AAAI 2021)
  - **MusicBERT** (Music, ACL 2021)
  - **DeepRapper** (Music, ACL 2021)

Outline

• Time-efficient	
• **FastSpeech 1/2** (TTS, NeurIPS 2019/ICLR 2021)	AR→NAR
• **FastCorrect 1/2** (ASR, paper submission)	AR→NAR
• **PriorGrad** (TTS, paper submission)	Diffusion
• Memory/Computation-efficient	
• **LightSpeech** (TTS, ICASSP 2021)	NAS
• **AdaSpeech** (TTS, ICLR 2021)	Cross-speaker/domain-knowledge
• Data-efficient	
• **AdaSpeech 2/3** (TTS, ICASSP 2021/INTERSPECH 2021)	Cross-speaker/domain-knowledge
• **LRSpeech** (TTS/ASR, ICML 2019/KDD 2020)	Cross-lingual/KD/BT
• **MixSpeech** (ASR, ICASSP 2021)	Domain-knowledge
• **SongMASS** (Music, AAAI 2021)	Pre-training/domain-knowledge
• **MusicBERT** (Music, ACL 2021)	Pre-training
• **DeepRapper** (Music, ACL 2021)	Pre-training

Text to speech synthesis

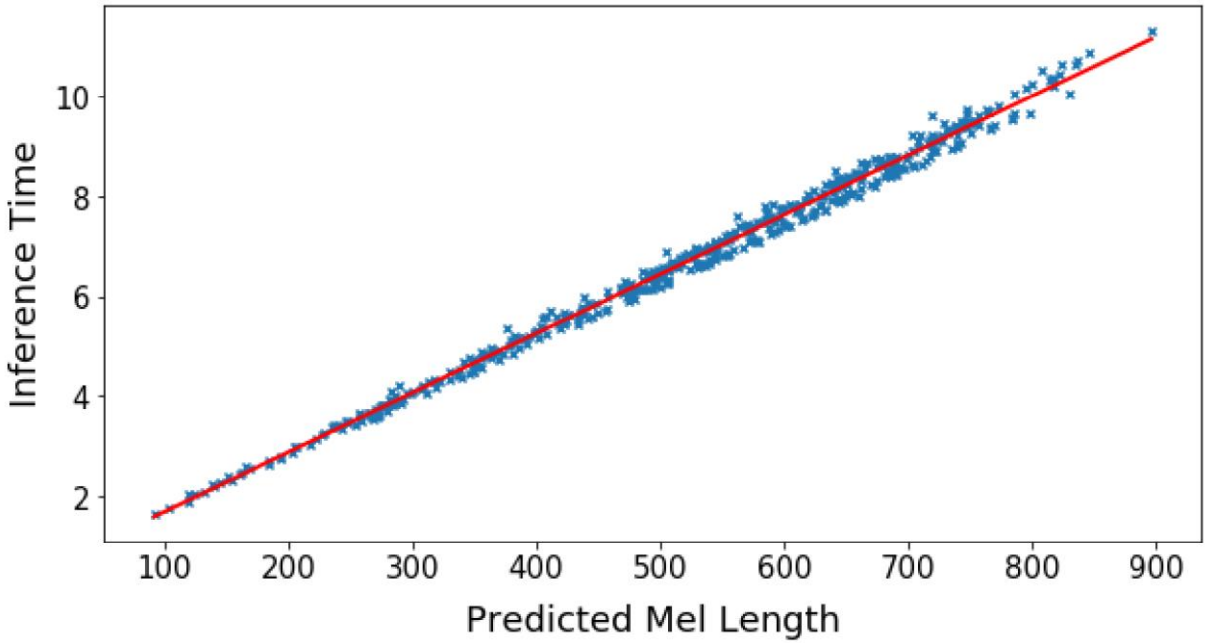
- Neural based end-to-end speech synthesis
  - Text Analysis: text → phoneme (e.g., *Jan.* → *January* → *dʒænjueri)*
    - Text normalization, grapheme-to-phoneme conversion, polyphone disambiguation
  - Acoustic Model: phoneme → mel-spectrogram
    - Tacotron 2, DeepVoice 3, TransformerTTS, FastSpeech 1/2
  - Vocoder: Mel-spectrogram → waveform (neural model)
    - WaveNet, WaveRNN, LPCNet, WaveGlow, MelGAN



Tan, Xu, et al. "A survey on neural speech synthesis." arXiv preprint arXiv:2106.15561 (2021).

Time-efficient ML for TTS

- End-to-end neural TTS model usually adopts autoregressive mel-spectrogram and waveform generation
  - Sequence is very long, e.g., 1s speech, 500 mel, 24000 waveform points
  - Slow inference speed

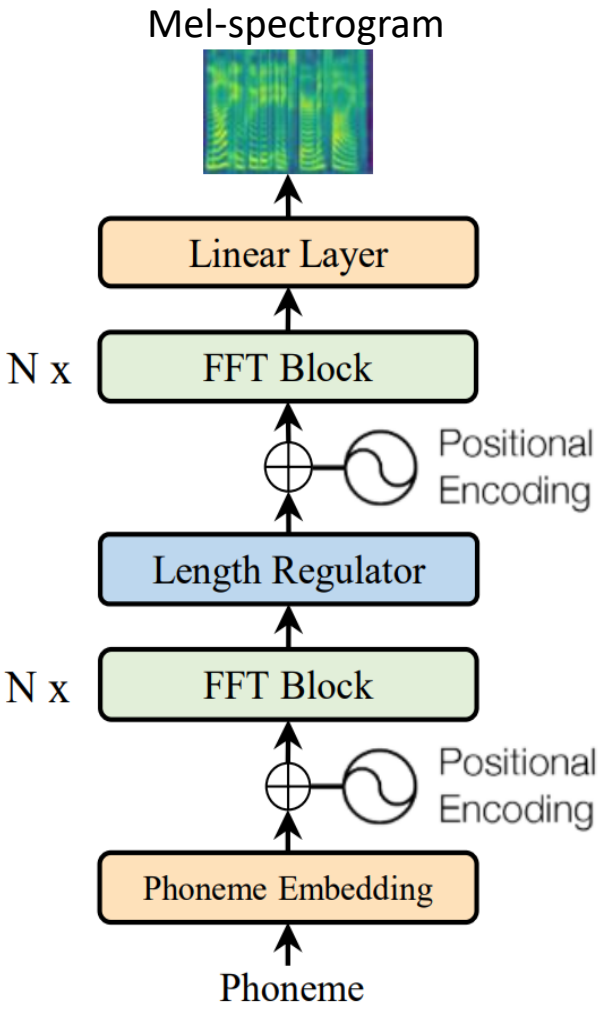


FastSpeech

- Problems: Previous autoregressive TTS models (Tacotron 2, DeepVoice 3, Transformer TTS) suffer from
  - Slow inference speed: autoregressive mel-spectrogram generation is slow for long sequence;
  - Not robust: words skipping and repeating;
  - Lack of controllability: hard to control the voice speed/prosody in the autoregressive generation

  You can call me directly at 4257037344 or my cell 4254447474 or send me a meeting request with all the appropriate information.

- Key designs in FastSpeech
  - Generate mel-spectrogram in parallel (for speedup)
  - Remove the text-speech attention mechanism (for robustness)
  - Feed-forward transformer with length regulator (for controllability)



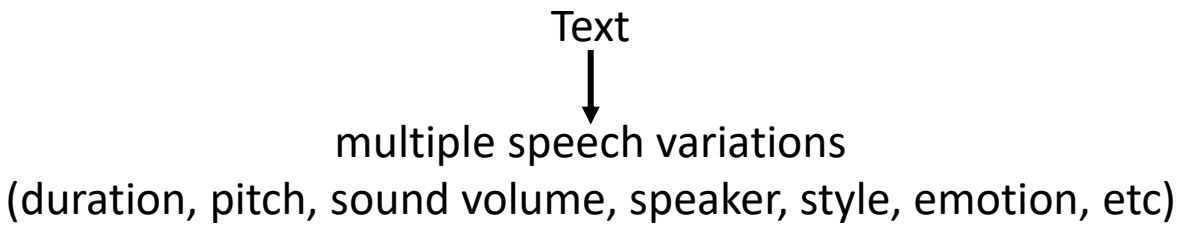
FastSpeech

- FastSpeech has the following advantages
  - **Extremely fast**: **270x** inference speedup on mel-spectrogram generation, **38x** speedup on final waveform generation!
  - **Robust**: no bad case of words skipping and repeating.
  - **Controllable**: can control voice speed and prosody.
  - **Voice quality**: on par or better than previous SOTA model.

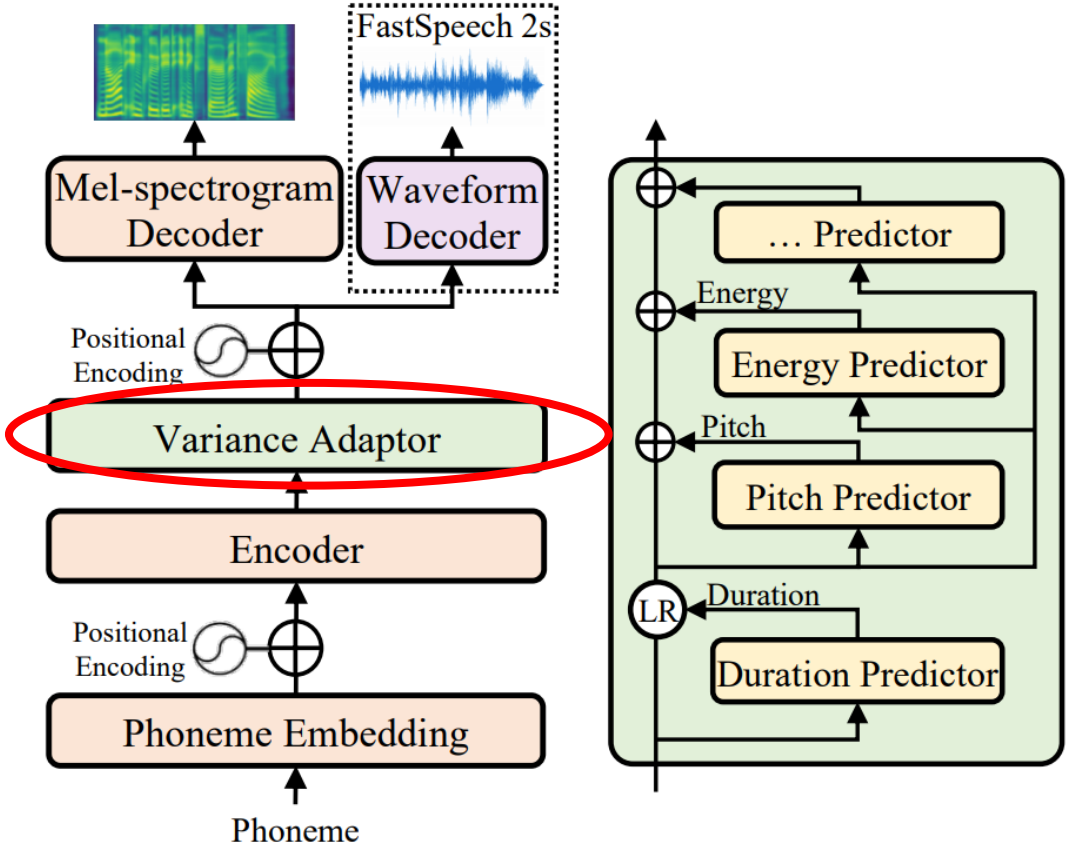
https://speechresearch.github.io/fastspeech/

FastSpeech 2

- The improvement space for FastSpeech
  - **Training pipeline complicated**: two-stage teacher-student distillation
  - **Target is not good**: the target mels distilled from teacher suffer from information loss
  - **Duration is not accurate**: the duration extracted from teacher is not accurate enough
- Improvements in FastSpeech 2
  - **Simplify training pipeline**: remove teacher-student distillation
  - **Use ground-truth speech as target**: avoid information loss
  - **Improve duration** & **Introduce more variance information**: ease the **one-to-many mapping** problem



FastSpeech 2



(a) FastSpeech 2 (b) Variance adaptor

- Variance adaptor: use variance predictor to predict duration, pitch, energy, etc.
- FastSpeech 2 improves FastSpeech with
  - more simplified training pipeline
  - higher voice quality
  - maintain the advantages of **fast, robust and even more controllable** synthesis in FastSpeech

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

FastSpeech 1/2

- Product Transfer: FastSpeech 1/2 are deployed on Microsoft **Azure Speech Service (TTS)** for **70+ languages/locales**

Languages	Locales	Languages	Locales	Languages	Locales	Languages	Locales
Arabic	ar-EG, ar-SA	Finnish	fi-FI	Japanese	ja-JP	Slovenian	sl-SI
Bulgarian	bg-BG	French	fr-FR, fr-CA, fr-CH	Korean	ko-KR	Spanish	es-ES, es-MX
Catalan	ca-ES	German	de-DE, de-AT, de-CH	Malay	ms-MY	Swedish	sv-SE
Chinese	zh-CN, zh-HK, zh-TW	Greek	el-GR	Norwegian	nb-NO	Tamil	ta-IN
Croatian	hr-HR	Hebrew	he-IL	Polish	pl-PL	Telugu	te-IN
Czech	cs-CZ	Hindi	hi-IN	Portuguese	pt-BR, pt-PT	Thai	th-TH
Danish	da-DK	Hungarian	hu-HU	Romanian	ro-RO	Turkish	tr-TR
Dutch	nl-NL	Indonesian	id-ID	Russia	ru-RU	Vietnamese	vi-VN
English	en-US, en-UK, en-AU, en-CA, en-IN, en-IE	Italian	it-IT	Slovak	sk-SK	Irish	ga-IE
Estonian	et-EE	Maltese	mt-MT	Lithuanian	lt-LT	Latvian	lv-LV

https://azure.microsoft.com/en-us/services/cognitive-services/text-to-speech

Outline

- Time-efficient
  - **FastSpeech 1/2** (TTS, NeurIPS 2019/ICLR 2021)
  - **FastCorrect 1/2** (ASR, paper submission)
  - **PriorGrad** (TTS, paper submission)
- Memory/Computation-efficient
  - **LightSpeech** (TTS, ICASSP 2021)
  - **AdaSpeech** (TTS, ICLR 2021)
- Data-efficient
  - **AdaSpeech 2/3** (TTS, ICASSP 2021/INTERSPECH 2021)
  - **LRSpeech** (TTS/ASR, ICML 2019/KDD 2020)
  - **MixSpeech** (ASR, ICASSP 2021)
  - **SongMASS** (Music, AAAI 2021)
  - **MusicBERT** (Music, ACL 2021)
  - **DeepRapper** (Music, ACL 2021)

ASR error correction

- ASR post-processing
  - Reranking: Select the best one from N-best candidates
  - Correction: Correct the error in 1-best/N-best candidates
  - Why correction? Reranking cannot create new/better candidates, but just select existing candidates
- ASR error correction
  - Given training corpus *(S, T),* an ASR model *M*, *M(S)* is the text generated from speech *S* by *M,* a correction model *C* is trained on *(M(S), T).*
- Problem
  - Previous works simply use encoder-decoder autoregressive (AR) model for correction
  - The latency is too high, and cannot be used for online deployment
    - e.g., ASR latency is 500ms on CPU, AR correction model latency is 660ms! Slow down by >2x
  - Goal: reduce latency while keeping similar accuracy

Naïve NAR solution fails

- Naïve solution
  - Non-autoregressive (NAR) sequence generation model from NMT
  - Even increase WER
- Task analysis
  - Different from non-autoregressive NMT (all tokens need to be modified)
  - Few modifications in ASR error correction (e.g., 10% WER)
  - How to detect errors? How to modify errors? How to avoid modifying correct ones?
- Leverage prior knowledge in correction?
  - Monotonic alignment
  - Error pattern: insert, delete, substitute
  - Provide detailed error patterns for error detections and corrections

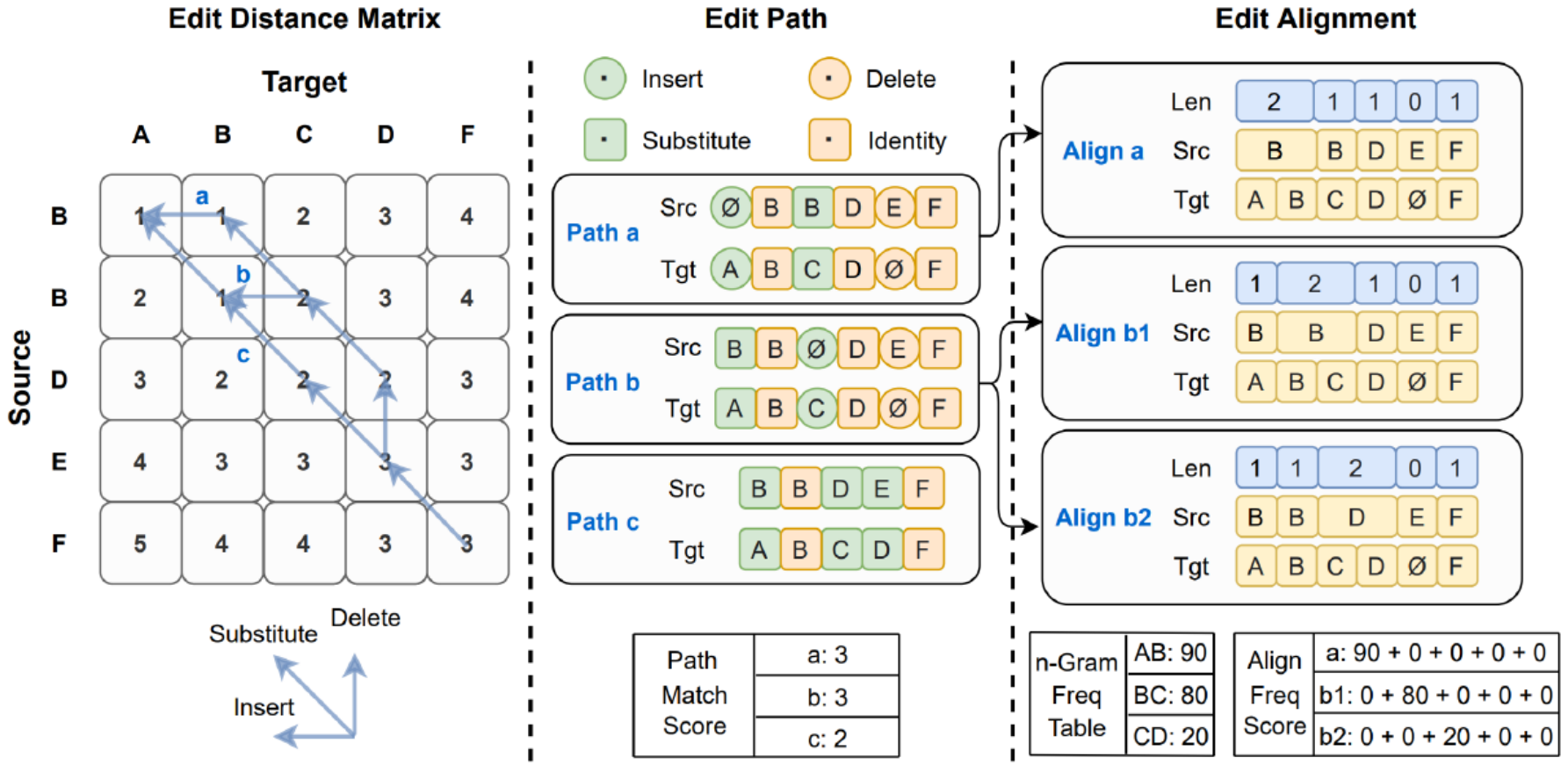
Our solution: FastCorrect

- FastCorrect: Fast Error Correction with Edit Alignment for Automatic Speech Recognition
  - 1 best correction
  - Offline experiment: **7-9x** inference speedup, **8%** WER reduction
  - Online deployment ongoing
- FastCorrect 2: Fast Error Correction on Multiple Candidates for Automatic Speech Recognition
  - N best correction
  - Offline experiment: **6x** inference speedup, **11%** WER reduction
  - Online deployment ongoing

FastCorrect

- Edit alignment
  - Calculate edit path from edit distance
  - Choose edit alignment
  - Get duration of each source token (0 deletion, 1 unchanged/substitution, >1 insertion)
- Non-autoregressive model
  - Encoder, duration predictor, decoder

FastCorrect: Edit alignment



Source: B B D E F

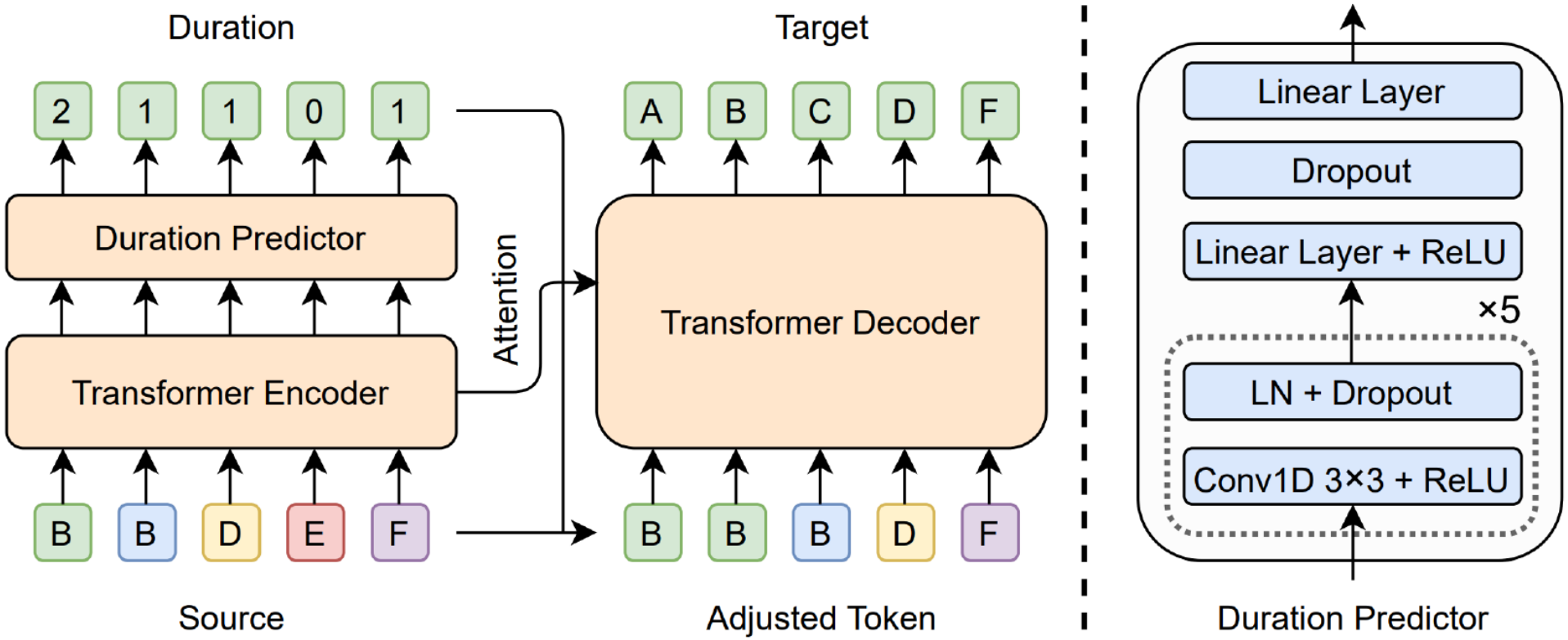
Target: A B C D F

Source: |B|,|B|,|B|,|D|,|E|,|F|

Target: |A|,|B|,|C|,|D|,| |,|F|

2 1 1 0 1

FastCorrect: NAR model



FastCorrect: Pre-training

- WER is usually low, effective training cases for correction models are limited
- Construct large-scale pseudo paired data for pre-training
- Randomly deleting, inserting and substituting in text
  - Substitution with homophone dictionary
  - The probability of modifying a word is set to the WER of the ASR model

FastCorrect: Experiments

- Dataset
  - AISHELL-1: 178 hours Mandarin speech
  - Internal dataset: 92k hours Mandarin speech
  - Pseudo data for pre-training: 400M sentences
- Model config
  - 6-6 layer encoder-decoder, hidden=512
  - Length predictor, 5-layer CNN, kernel=3

FastCorrect: Experiments

- Accuracy and latency

AISHELL-1	Test Set		Dev Set		Latency (ms/sent) on Test Set		
	WER	WERR	WER	WERR	GPU	CPU*4	CPU
No correction	4.83	-	4.46	-	-	-	-
AR model	4.08	15.53	3.80	14.80	149.5 (1×)	248.9 (1×)	531.3 (1×)
LevT (MIter=1) [9]	4.73	2.07	4.37	2.02	54.0 (2.8×)	82.7 (3.0×)	158.1 (3.4×)
LevT (MIter=3) [9]	4.74	1.86	4.38	1.79	60.5 (2.5×)	83.9 (3.0×)	161.6 (3.3×)
FELIX [21]	4.63	4.14	4.26	4.48	23.8 (6.3×)	41.7 (6.0×)	85.7 (6.2×)
FastCorrect	**4.16**	**13.87**	**3.89**	**13.3**	**21.2** (7.1×)	**40.8** (6.1×)	**82.3** (6.5×)

Internal Dataset	Test Set		Dev Set		Latency (ms/sent) on Test Set		
	WER	WERR	WER	WERR	GPU	CPU*4	CPU
No correction	11.17	-	11.24	-	-	-	-
AR model	10.22	8.50	10.31	8.27	191.5 (1×)	336 (1×)	657.7 (1×)
LevT (MIter=1) [9]	11.26	-0.80	11.35	-0.98	60.5 (3.2×)	102.6 (3.3×)	196.5 (3.3×)
LevT (MIter=3) [9]	11.45	-2.50	11.56	-2.85	75.6 (2.5×)	118.9 (2.8×)	248.0 (2.7×)
FELIX [21]	11.14	0.27	11.21	0.27	25.9 (7.4×)	43.0 (7.8×)	90.9 (7.2×)
FastCorrect	**10.27**	**8.06**	**10.35**	**7.92**	**21.5** (8.9×)	**42.4** (7.9×)	**88.6** (7.4×)

FastCorrect: Experiments

- Ablation study

Model	Internal Dataset	AISHELL-1 Dataset
No correction	11.17	4.83
AR model	10.22	4.08
- Pre-training	10.26	16.01
- Fine-tuning	11.70	5.28
FastCorrect	10.27	4.16
- Pre-training	10.33	4.83
- Fine-tuning	11.74	5.19
- Edit Alignment	12.27	4.67

FastCorrect: Experiments

- Comparison to AR model with shallow decoder

Model	AISHELL-1			Internal Dataset		
	WER	Latency (ms/sent)		WER	Latency (ms/sent)	
	%	GPU	CPU	%	GPU	CPU
No Correction	4.83	-	-	11.17	-	-
AR 6-6	4.08	149.5 (1×)	531.3 (1×)	10.26	190.6 (1×)	648.3 (1×)
AR 8-4	4.14	120.5 (1.2×)	427.6 (1.2×)	10.28	144.1 (1.3×)	542.0 (1.2×)
AR 10-2	4.23	84.0 (1.8×)	317.6 (1.5×)	10.33	100.8 (1.9×)	431.2 (1.5×)
AR 11-1	4.30	66.5 (2.2×)	281.0 (1.7×)	10.44	79.1 (2.4×)	372.3 (1.7×)
FastCorrect	4.16	**21.2** (7.1×)	**82.3** (6.5×)	10.33	**21.4** (8.9×)	**86.8** (7.5×)

FastCorrect: Experiments

- Analysis of FastCorrect, LevT and FELIX

Model	Internal Dataset				AISHELL-1			
	P_{edit}	R_{edit}	P_{right}	WERR	P_{edit}	R_{edit}	P_{right}	WERR
AR model	94.3	31.0	18.9	8.50	97.2	47.4	35.1	15.53
LevT	74.0	**41.3**	11.4	-0.80	91.6	26.1	20.3	2.07
FELIX	93.6	19.9	10.1	0.27	96.5	33.8	22.8	4.14
FastCorrect	**95.0**	27.6	**16.2**	**8.06**	**96.8**	**48.1**	**26.4**	**13.87**

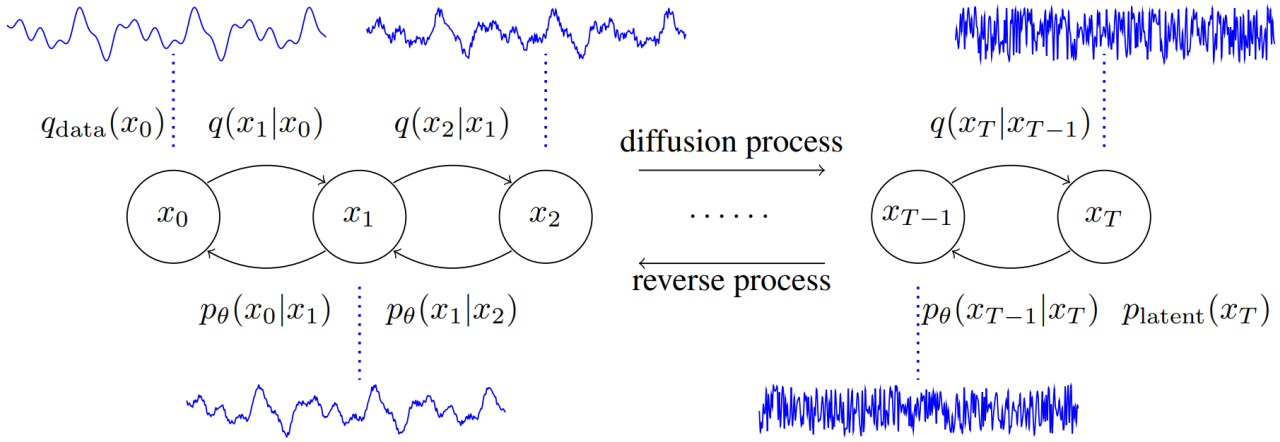
Better error-dectection and error-correction ability

Outline

• Time-efficient ML	
• **FastSpeech 1/2** (TTS, NeurIPS 2019/ICLR 2021)	AR→NAR
• **FastCorrect 1/2** (ASR, paper submission)	AR→NAR
• **PriorGrad** (TTS, paper submission)	Diffusion
• Memory/Computation-efficient ML	
• **LightSpeech** (TTS, ICASSP 2021)	NAS
• **AdaSpeech** (TTS, ICLR 2021)	Cross-speaker/domain-knowledge
• Data-efficient ML	
• **AdaSpeech 2/3** (TTS, ICASSP 2021/INTERSPECH 2021)	Cross-speaker/domain-knowledge
• **LRSpeech** (TTS/ASR, ICML 2019/KDD 2020)	Cross-lingual/KD/BT
• **MixSpeech** (ASR, ICASSP 2021)	Domain-knowledge
• **SongMASS** (Music, AAAI 2021)	Pre-training/domain-knowledge
• **MusicBERT** (Music, ACL 2021)	Pre-training
• **DeepRapper** (Music, ACL 2021)	Pre-training

Diffusion model for TTS

- Diffusion model



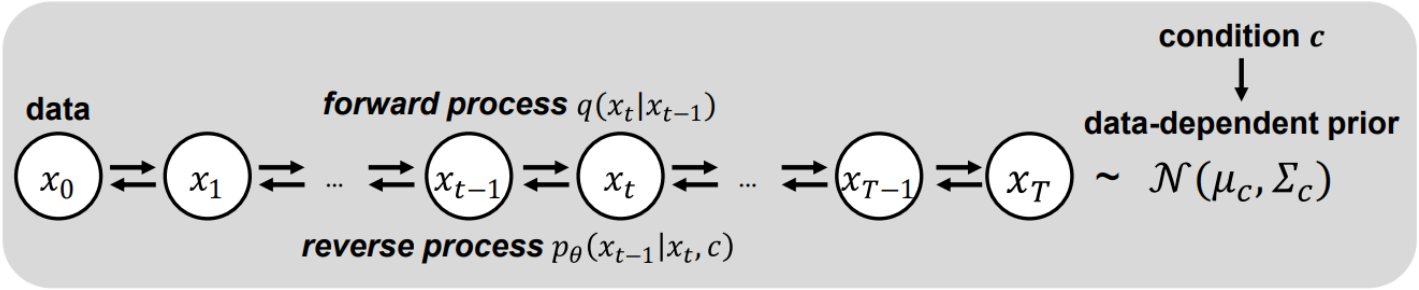
$$q(x_t|x_{t-1}) := \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I})$$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}\left(x_{t-1}; \mu_\theta(x_t,t), \Sigma_\theta(x_t,t)\right)$$

$$\mu_\theta(x_t,t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t,t)\right) \qquad \sigma_\theta(x_t,t) = \tilde{\beta}_t^{\frac{1}{2}}$$

Time-efficient diffusion model for TTS

- Our method: PriorGrad



Algorithm 1 Training of PriorGrad

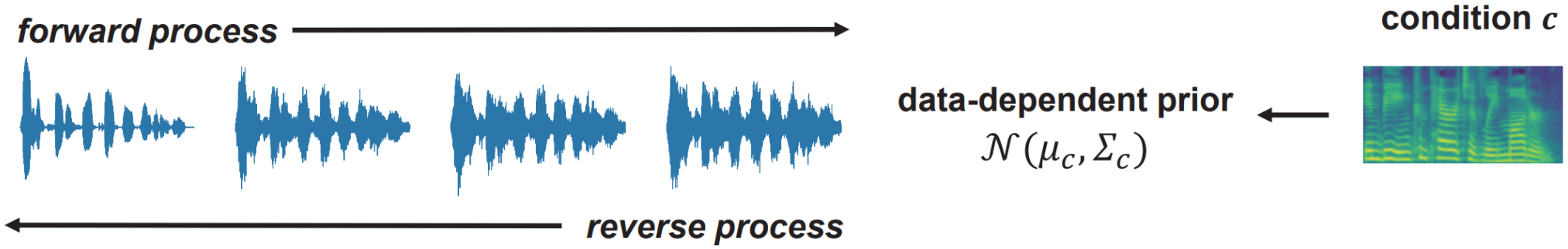
repeat
  (μ, Σ) = data-dependent prior
  Sample $x_0 \sim q_{data}, \epsilon \sim \mathcal{N}(0, \Sigma)$
  Sample $t \sim \mathcal{U}(\{1, \cdots, T\})$
  $x_t = \sqrt{\bar{\alpha}_t}(x_0 - \mu) + \sqrt{1 - \bar{\alpha}_t}\epsilon$
  $\mathcal{L} = \|\epsilon - \epsilon_\theta(x_t, c, t)\|^2_{\Sigma^{-1}}$
  Update the model parameter θ with $\nabla_\theta \mathcal{L}$
until converged

Algorithm 2 Sampling of PriorGrad

(μ, Σ) = data-dependent prior
Sample $x_T \sim \mathcal{N}(0, \Sigma)$
for $t = T, T-1, \cdots, 1$ **do**
  $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, c, t)$
  if $t > 1$ **then**
    $x_{t-1} = x_{t-1} + \sigma_t \Sigma^{\frac{1}{2}}$
  else
    $x_{t-1} = x_{t-1} + \mu$
  end if
end for
return x_0

PriorGrad

- PriorGrad for vocoder

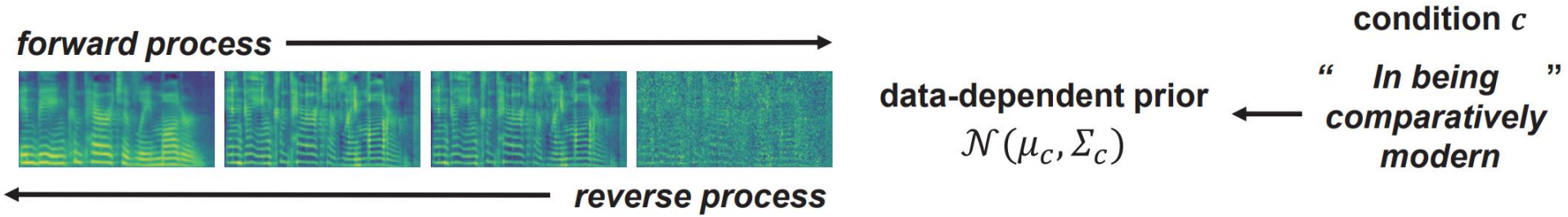


Type	Method	MOS
	GT	4.31 ± 0.11
Vocoder	GT + WaveGrad [2] (1M)	4.01 ± 0.11
	GT + PriorGrad (300K)	$\mathbf{4.06 \pm 0.11}$
Text-to-speech	FastSpeech 2 [28] + WaveGrad [2] (1M)	4.01 ± 0.14
	FastSpeech 2 [28] + PriorGrad (300K)	3.97 ± 0.12

Method	100K	500K	1M
WaveGrad	0	0	0
PriorGrad	**0.297**	**0.224**	**0.333**

PriorGrad

- PriorGrad for acoustic model



Method	Small	Large
GT (PWG [40])	4.12 ± 0.17	
Baseline (300K)	3.69 ± 0.15	3.86 ± 0.12
PriorGrad (60K)	3.73 ± 0.14	$\mathbf{3.98 \pm 0.12}$

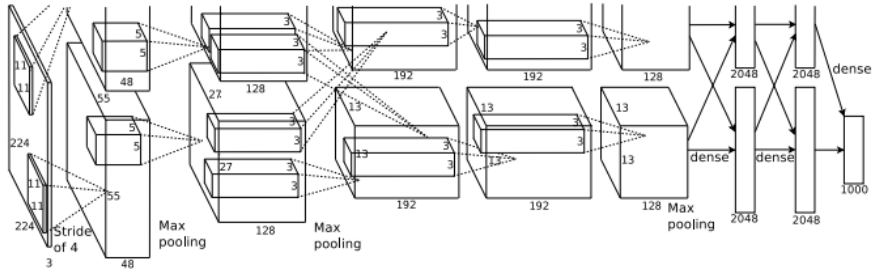
Model	Small	Large
Baseline (300K)	0	0
PriorGrad (60K)	**0.145**	**0.408**

Outline

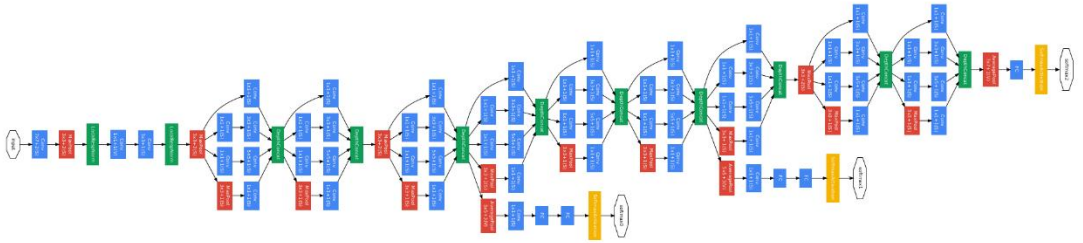
- Time-efficient ML
  - **FastSpeech 1/2** (TTS, NeurIPS 2019/ICLR 2021)
  - **FastCorrect 1/2** (ASR, paper submission)
  - **PriorGrad** (TTS, paper submission)
- Memory/Computation-efficient ML
  - **LightSpeech** (TTS, ICASSP 2021)
  - **AdaSpeech** (TTS, ICLR 2021)
- Data-efficient ML
  - **AdaSpeech 2/3** (TTS, ICASSP 2021/INTERSPECH 2021)
  - **LRSpeech** (TTS/ASR, ICML 2019/KDD 2020)
  - **MixSpeech** (ASR, ICASSP 2021)
  - **SongMASS** (Music, AAAI 2021)
  - **MusicBERT** (Music, ACL 2021)
  - **DeepRapper** (Music, ACL 2021)

Architecture of an NN is crucial to its performance

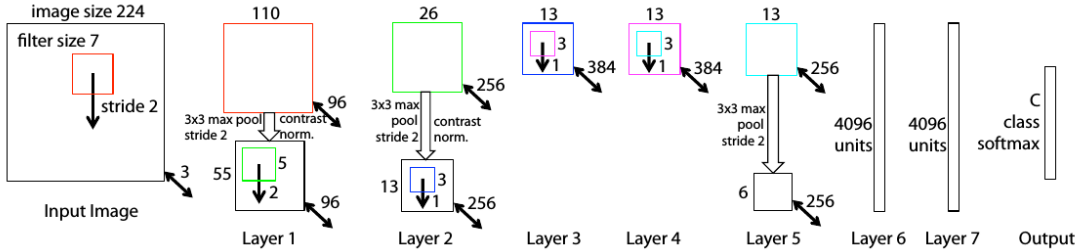
ImageNet winning neural architectures



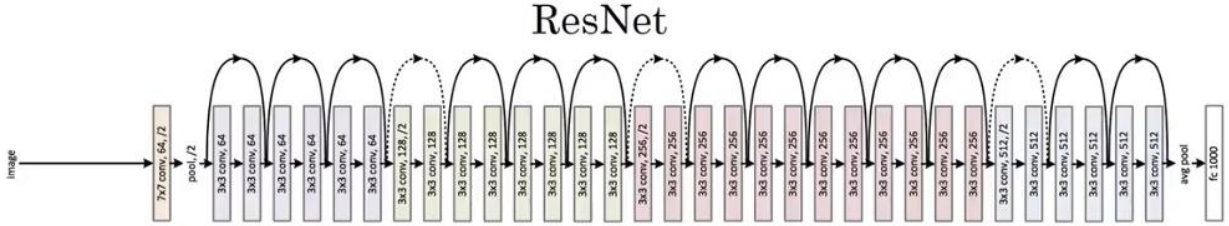
AlexNet 2012



Inception 2014

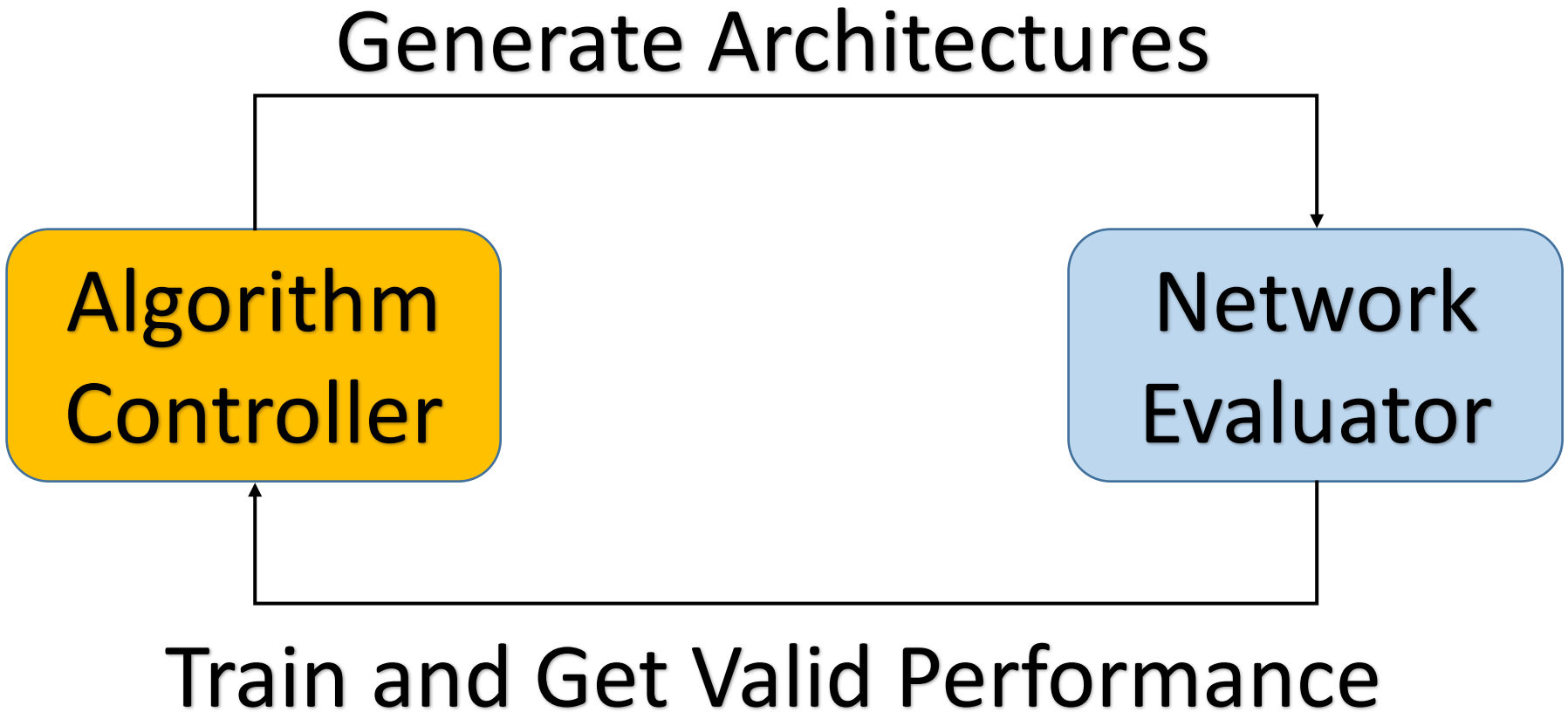


ZFNet 2013



ResNet 2015

General framework of NAS



Our work on NAS

- NAS algorithms
  - Semi-Supervised Neural Architecture Search, NeurIPS 2020
  - Neural Architecture Search with GBDT, arXiv 2020
  - Analyzing and Mitigating Interference in Neural Architecture Search, In submission
- NAS applications
  - **LightSpeech**: Lightweight and Fast Text to Speech with Neural Architecture Search, ICASSP 2021
  - NAS-BERT: Task-Agnostic and Adaptive-Size BERT Compression with Neural Architecture Search, KDD 2021

LightSpeech

- Background
  - Deploying TTS in mobile phones or embedded devices requires extremely small memory usage and inference latency.
  - Although FastSpeech series can synthesize speech very fast, it is not enough for this scenarios.
  - Further speedup FastSpeech and reduce the model size are necessary.
- Thus, we propose LightSpeech, using neural architecture search (GBDT-NAS) for lightweight architectures

Model	#Params	Compression Ratio	MACs	Ratio	Inference Speed (RTF)	Inference Speedup
FastSpeech 2	27.0M	/	12.50G	/	6.1×10^{-2}	/
LightSpeech	**1.8M**	**15x**	**0.76G**	**16x**	9.3×10^{-3}	**6.5x**

Model	#Params	CMOS
FastSpeech 2	27.0M	0
FastSpeech 2*	1.8M	-0.230
LightSpeech	1.8M	+0.04

https://speechresearch.github.io/lightspeech/

Outline

- Time-efficient ML
  - **FastSpeech 1/2** (TTS, NeurIPS 2019/ICLR 2021)
  - **FastCorrect 1/2** (ASR, paper submission)
  - **PriorGrad** (TTS, paper submission)
- Memory/Computation-efficient ML
  - **LightSpeech** (TTS, ICASSP 2021)
  - **AdaSpeech** (TTS, ICLR 2021)
- Data-efficient ML
  - **AdaSpeech 2/3** (TTS, ICASSP 2021/INTERSPECH 2021)
  - **LRSpeech** (TTS/ASR, ICML 2019/KDD 2020)
  - **MixSpeech** (ASR, ICASSP 2021)
  - **SongMASS** (Music, AAAI 2021)
  - **MusicBERT** (Music, ACL 2021)
  - **DeepRapper** (Music, ACL 2021)

Background

- Custom voice is an important service in text to speech (TTS)
  - Microsoft Azure: https://speech.microsoft.com/customvoice
- The scenario is to support TTS for the voice of any user/customer
  - User need to record their voice with few sentences using their own devices
  - Upload to speech service for voice adaption
  - Speech service provide a custom model and serve for this voice

Challenges and solutions

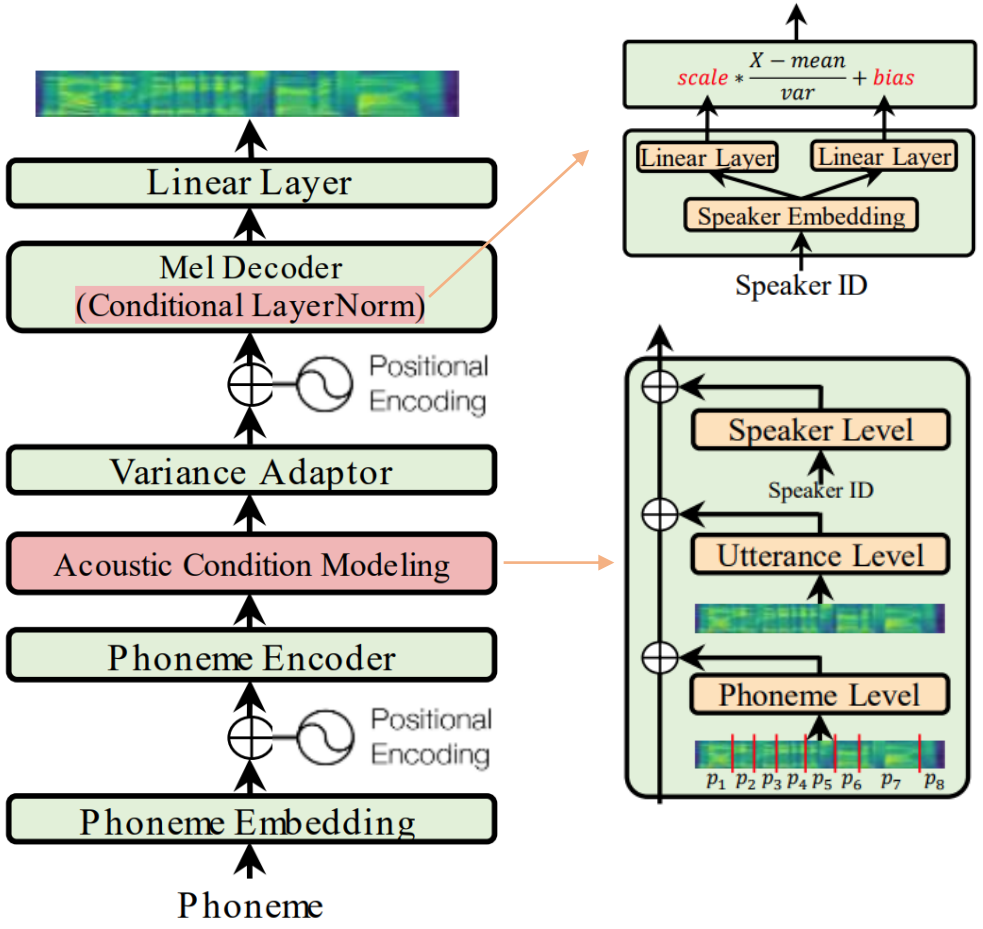
- Challenge 1: Few adaptation data and parameters, different acoustics
  - *AdaSpeech: Adaptive Text to Speech for Custom Voice*, ICLR 2021
- Challenge 2: How to leverage plenty of untranscribed speech?
  - *AdaSpeech 2: Adaptive Text to Speech with Untranscribed Data*, ICASSP 2021
- Challenge 3: How to adapt to spontaneous speech?
  - *AdaSpeech 3: Adaptive Text to Speech for Spontaneous Styles*, submitted to INTERSPEECH 2021

AdaSpeech: Adaptive Text to Speech for Custom Voice

- Pipeline: pre-training, fine-tuning, inference
- Challenges
  - To support diverse customers, the adaptation model needs to handle diverse acoustic conditions which are very different from source speech data
  - To support many customers, the adaptation parameters need to be small enough for each target speaker to reduce memory usage while maintaining high voice quality
    - e.g., each user/voice with 100MB, 1M users, total memory storage = 100PB!
- However, related works
  - Too many adaptation parameters
  - Poor adaptation quality with few parameters
  - Only consider source and adaptation data are in the same domain

AdaSpeech ——Key designs

- Acoustic condition modeling
  - Model diverse acoustic conditions at speaker/utterance/phoneme level
- Conditional layer normalization
  - To fine-tune as small parameters as possible while ensuring voice quality
- Consider adaptation data is different from source data
  - More challenging but close to product scenario



AdaSpeech——Experiments

- Train on LibriTTS, adapt to LJSpeech/VCTK/LibriTTS with 20 sentences
  - AdaSpeech achieves better quality with similar parameters
  - AdaSpeech achieves on par quality with much smaller adaptation parameters

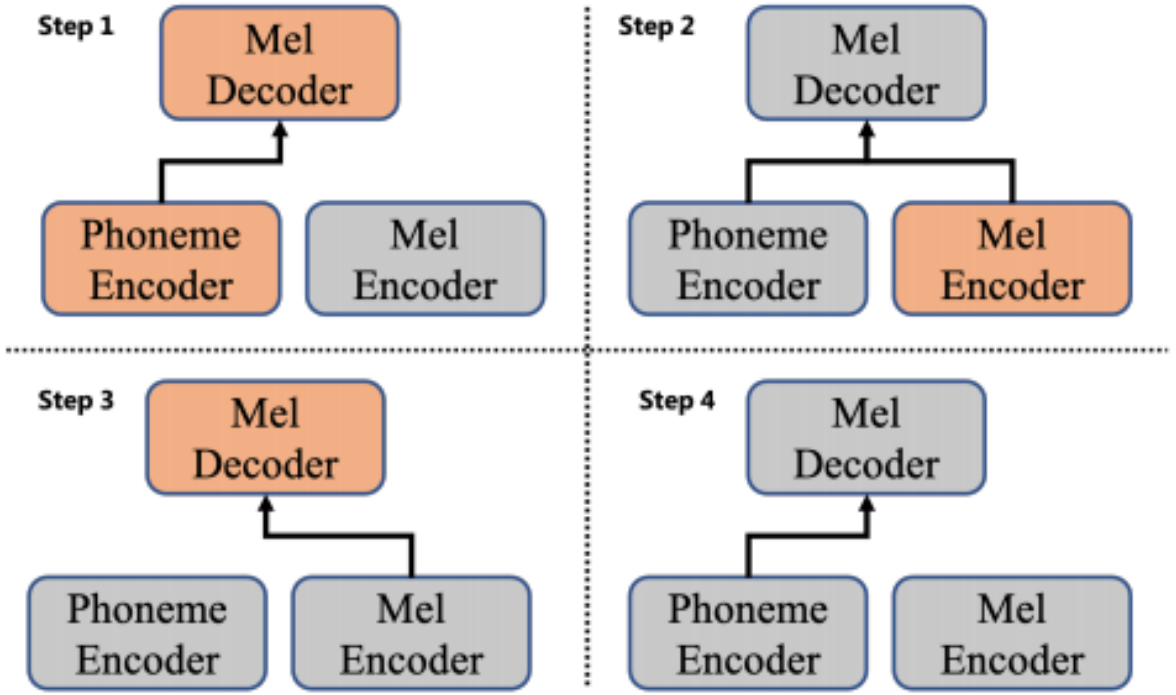
Metric	Setting	# Params/Speaker	LJSpeech	VCTK	LibriTTS
MOS	*GT*	/	3.98 ± 0.12	3.87 ± 0.11	3.72 ± 0.12
	GT mel + Vocoder	/	3.75 ± 0.10	3.74 ± 0.11	3.65 ± 0.12
	Baseline (spk emb)	256 (256)	2.37 ± 0.14	2.36 ± 0.10	3.02 ± 0.13
	Baseline (decoder)	14.1M (14.1M)	3.44 ± 0.13	3.35 ± 0.12	3.51 ± 0.11
	AdaSpeech	1.2M (4.9K)	3.45 ± 0.11	3.39 ± 0.10	3.55 ± 0.12
SMOS	*GT*	/	4.36 ± 0.11	4.44 ± 0.10	4.31 ± 0.07
	GT mel + Vocoder	/	4.29 ± 0.11	4.36 ± 0.11	4.31 ± 0.07
	Baseline (spk emb)	256 (256)	2.79 ± 0.19	3.34 ± 0.19	4.00 ± 0.12
	Baseline (decoder)	14.1M (14.1M)	3.57 ± 0.12	3.90 ± 0.12	4.10 ± 0.10
	AdaSpeech	1.2M (4.9K)	3.59 ± 0.15	3.96 ± 0.15	4.13 ± 0.09

https://speechresearch.github.io/adaspeech/

- Product Transfer: AdaSpeech is deployed in **Microsoft Azure Speech (TTS)** for custom voice service https://speech.microsoft.com/customvoice

AdaSpeech 2: Adaptive Text to Speech with Untranscribed Data

- Only untranscribed data, how to adapt?
  - In online meeting, only speech can be collected, without corresponding transcripts
- AdaSpeech 2, speech reconstruction with latent alignment
  - Step 1: source TTS model training
  - Step 2: speech reconstruction
  - Step 3: speaker adapatation
  - Step 4: inference



AdaSpeech 2——Experiments

- Train on LibriTTS, adapt to LJSpeech/VCTK with 50 untranscribed sentences
  - AdaSpeech 2 achieves better quality than previous methods
  - AdaSpeech 2 achieves on par quality with transcribed adaptation (AdaSpeech)

Metric	Setting	VCTK	LJSpeech
MOS	*GT*	3.58 ± 0.12	3.63 ± 0.11
	GT mel+Vocoder	3.42 ± 0.12	3.49 ± 0.11
	Joint-training	2.91 ± 0.09	2.89 ± 0.12
	PPG-based	3.39 ± 0.11	3.44 ± 0.12
	AdaSpeech	3.39 ± 0.10	3.45 ± 0.11
	AdaSpeech 2	3.38 ± 0.12	3.42 ± 0.12
SMOS	*GT*	4.20 ± 0.12	4.24 ± 0.09
	GT mel+Vocoder	4.06 ± 0.08	4.02 ± 0.11
	Joint-training	3.71 ± 0.13	3.19 ± 0.16
	PPG-based	3.82 ± 0.11	3.51 ± 0.15
	AdaSpeech	3.94 ± 0.12	3.59 ± 0.12
	AdaSpeech 2	3.84 ± 0.08	3.51 ± 0.12

https://speechresearch.github.io/adaspeech2/

AdaSpeech 3: Adaptive Text to Speech for Spontaneous Style

- Current TTS voices mostly focus on reading style. Spontaneous-style voice is useful for scenarios like podcast, conversation, etc.
- Challenges
  - Lack of spontaneous data
  - Difficulty in modeling filled pauses (FP, um and uh) and diverse rhythms

Cecily package in all of that um yeah so …

AdaSpeech 3: Adaptive Text to Speech for Spontaneous Style

- Train on LibriTTS, adapt with reading/spontaneous style data
  - AdaSpeech 3 achieve better voice quality and similarity than baseline AdaSpeech

Setting	Naturalness	Pause	Speaking Rate
GT	4.14 ± 0.06	4.01 ± 0.06	3.04 ± 0.06
GT mel+Voc	3.84 ± 0.06	3.78 ± 0.06	3.06 ± 0.08
AdaSpeech	3.21 ± 0.06	3.36 ± 0.06	2.66 ± 0.08
AdaSpeech 3	3.45 ± 0.06	3.53 ± 0.06	2.79 ± 0.06

Setting	**SMOS**
GT	4.33 ± 0.14
GT mel+Vocoder	4.07 ± 0.14
AdaSpeech	3.45 ± 0.18
AdaSpeech 3	3.75 ± 0.16

Cecily package in all of that um yeah so …

GT AdaSpeech AdaSpeech 3

*Six spoons of fresh snow peas, **um**, five thick slabs of blue cheese, and maybe a snack for her brother Bob.*

Before FP insertion After FP insertion

Outline

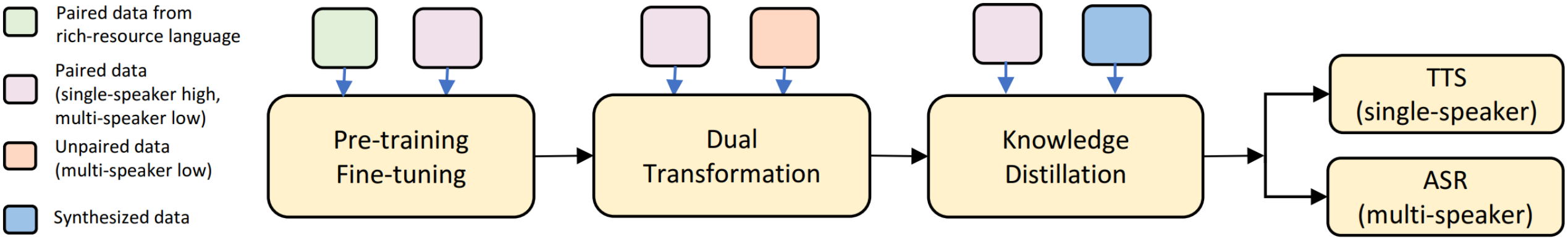
- Time-efficient ML
  - **FastSpeech 1/2** (TTS, NeurIPS 2019/ICLR 2021)
  - **FastCorrect 1/2** (ASR, paper submission)
  - **PriorGrad** (TTS, paper submission)
- Memory/Computation-efficient ML
  - **LightSpeech** (TTS, ICASSP 2021)
  - **AdaSpeech** (TTS, ICLR 2021)
- Data-efficient ML
  - **AdaSpeech 2/3** (TTS, ICASSP 2021/INTERSPECH 2021)
  - **LRSpeech** (TTS/ASR, ICML 2019/KDD 2020)
  - **MixSpeech** (ASR, ICASSP 2021)
  - **SongMASS** (Music, AAAI 2021)
  - **MusicBERT** (Music, ACL 2021)
  - **DeepRapper** (Music, ACL 2021)

Low-resource TTS

- There are **7,000+** languages in the world, but popular commercialized speech services only support **dozens of** languages

- There is strong business demand to support more languages in TTS. However, the data collection cost is high.
  - For TTS, the minimum data labeling cost for one language: ¥ 1 million

Low-resource TTS——LRSpeech



- **Step 1**: Language transfer
  - Human languages share similar pronunciations; Rich-resource language data is "free''
- **Step 2**: TTS and ASR help with each other
  - Leverage the task duality with unpaired speech and text data
- **Step 3**: Customization for product deployment with knowledge distillation
  - Better accuracy by data knowledge distillation
  - Customize multi-speaker TTS to a target-speaker TTS, and to small model

Low-resource TTS——LRSpeech

- Results

Language	Intelligibility Rate (IR)	Mean Opinion Score (MOS)
English	98.08	3.57
Lithuanian	98.60	3.65

LRSpeech achieves **high IR score (>98%)** and **MOS score (>3.5)**

https://speechresearch.github.io/lrspeech/

- Data cost

Data Resource	Full-Resource	Speech Chain [36]	Almost Unsup [29]	SeqRQ-AE [20]	Our Method
Text normalization rule	✓	?	✓	✓	✓
Pronunciation lexicon	✓	×	✓	✓	×
Paired data (single-speaker, high)	dozens of hours	20 hours	200 sentences	200 sentences	50 sentences
Paired data (multi-speaker, low)	hundreds of hours	×	×	×	1000 sentences
Unpaired speech (single-speaker, high)	×	80 hours	13000 sentences	13000 sentences	×
Unpaired speech (multi-speaker, low)	×	×	×	×	13000 sentences
Unpaired text	×	✓	✓	✓	✓
Total Data Cost	312000	120000	74000	74000	833

LRSpeech

100x data cost reduction compared to previous works

Low-resource TTS——LRSpeech

- Product deployment
  - LRSpeech has been deployed in Microsoft Azure Text to Speech service
  - Extend 5 new low-resource languages for TTS: Maltese, Lithuanian, Estonian, Irish, Latvian

Locale	Language (Region)	Average MOS	Intelligibility
mt-MT	Maltese (Malta)	3.59	98.40%
lt-LT	Lithuanian (Lithuania)	4.35	99.25%
et-EE	Estonian (Estonia)	4.52	98.73%
ga-IE	Irish (Ireland)	4.62	99.43%
lv-LV	Latvian (Latvia)	4.51	99.13%

https://techcommunity.microsoft.com/t5/azure-ai/neural-text-to-speech-previews-five-new-languages-with/ba-p/1907604

Outline

- Time-efficient ML
  - **FastSpeech 1/2** (TTS, NeurIPS 2019/ICLR 2021)
  - **FastCorrect 1/2** (ASR, paper submission)
  - **PriorGrad** (TTS, paper submission)
- Memory/Computation-efficient ML
  - **LightSpeech** (TTS, ICASSP 2021)
  - **AdaSpeech** (TTS, ICLR 2021)
- Data-efficient ML
  - **AdaSpeech 2/3** (TTS, ICASSP 2021/INTERSPECH 2021)
  - **LRSpeech** (TTS/ASR, ICML 2019/KDD 2020)
  - **MixSpeech** (ASR, ICASSP 2021)
  - **SongMASS** (Music, AAAI 2021)
  - **MusicBERT** (Music, ACL 2021)
  - **DeepRapper** (Music, ACL 2021)

MusicBERT: Symbolic Music Understanding with Large-Scale Pre-Training, ACL 2021

- Understanding music is important for generation
  - Emotion recognition
  - Genre classification
  - Melody/accompaniment extraction
  - Structure analysis
- Previous works on music understanding
  - PiRhDy (ACM MM 2020), contextual word embedding
  - Shallow model, too much complicated design with music knowledge

MusicBERT

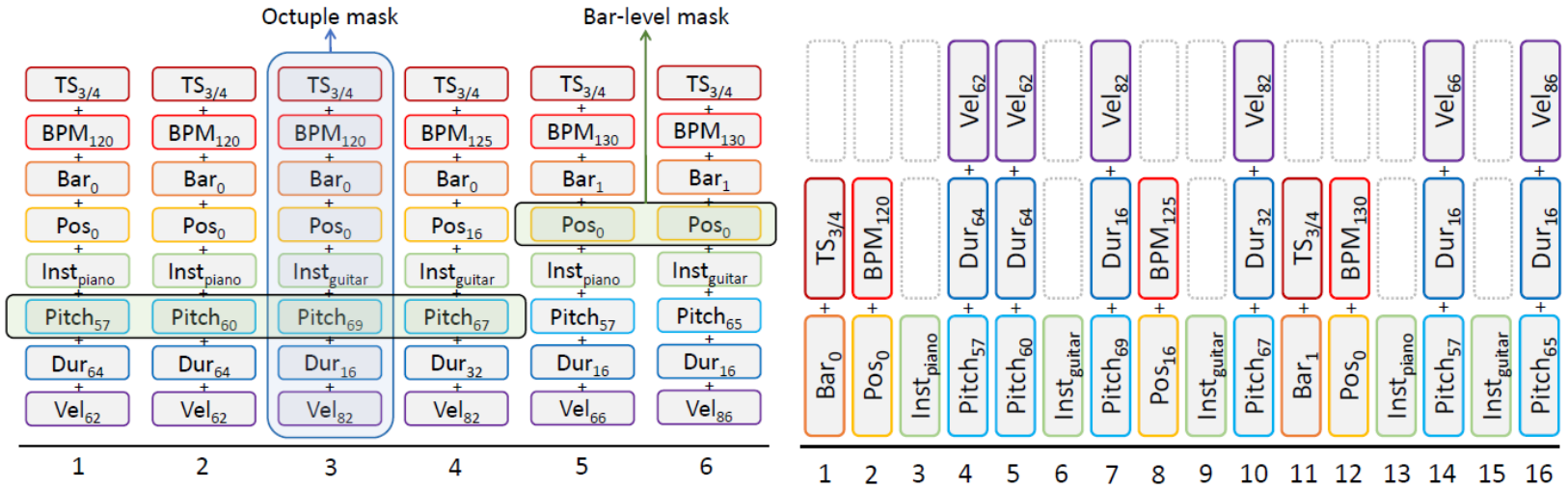
- Dataset construction: Million MIDI Dataset (MMD)
  - Crawled from various MIDI and sheet music websites
  - 1.5 million songs after deduplication and cleaning (10x larger than LMD)

Dataset	Songs	Notes (Millions)
MAESTRO	1,184	6
GiantMIDI-Piano	10,854	39
LMD	148,403	535
MMD	**1,524,557**	**2,075**

- Data representation: OctupleMIDI
  - Compound token: (TimeSig_4/4, Tempo_120, Bar_1, Pos_35, Piano, Pitch_64, Dur_12, Vel_38)
  - Supports changing tempo and time signature
  - Shorter length compared to previous encoding

MusicBERT

- OctupleMIDI representation



(a) OctupleMIDI encoding.

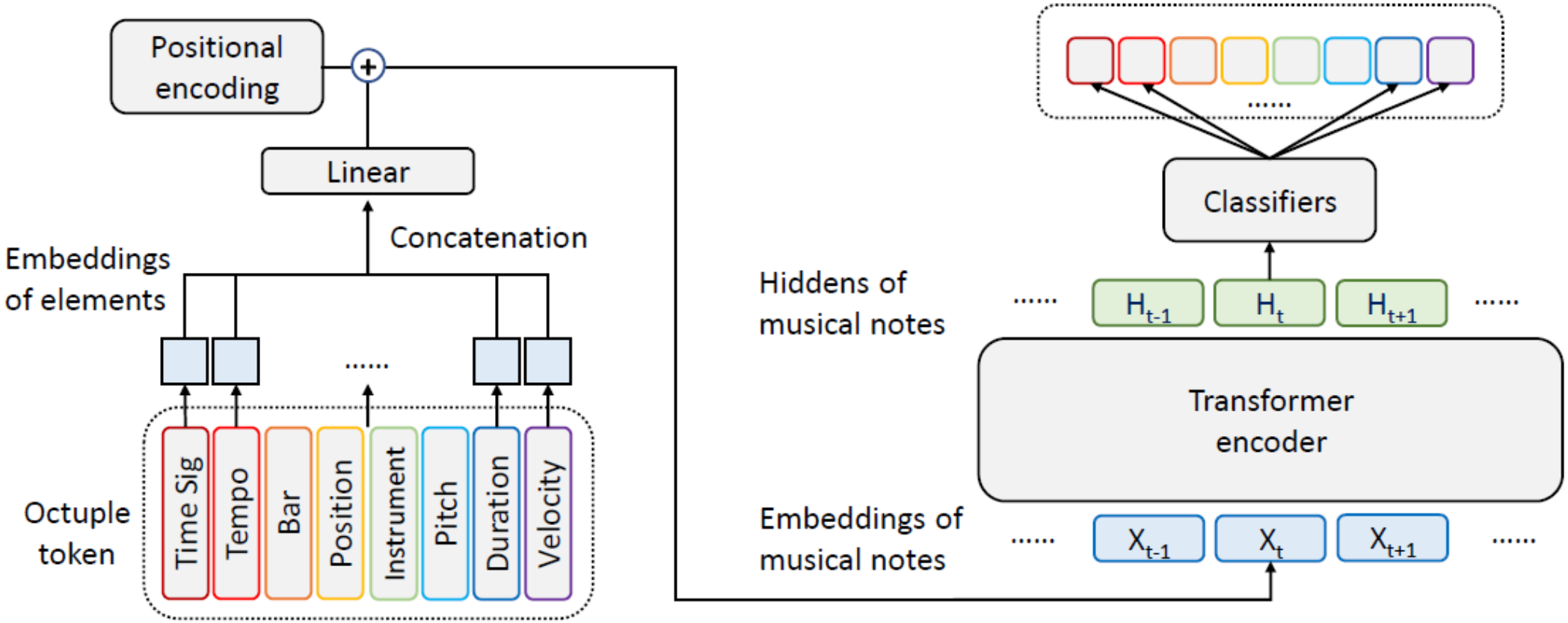
(b) CP-Like encoding.

(c) REMI-Like encoding.

Encoding	OctupleMIDI	CP-like	REMI-like
Tokens	**3607**	6906	15679

MusicBERT

- Model structure



MusicBERT

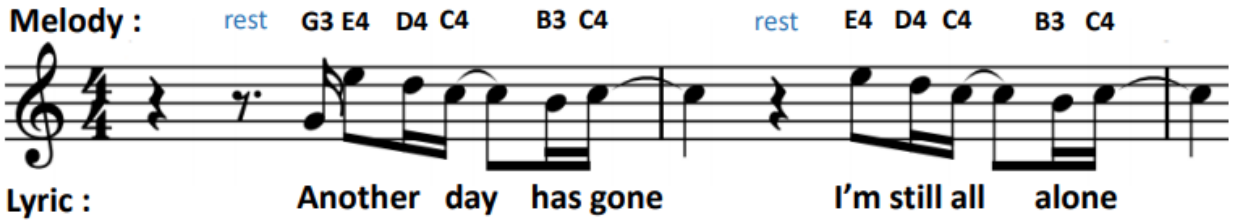
- Performance on music understanding tasks
  - Melody completion: Two sequences classification
  - Accompaniment completion: Melody and accompaniment sequences classification
  - Genre classification: Single sentence classification

Model	Melody Completion					Accompaniment Suggestion					Classification	
	MAP	HITS @1	HITS @5	HITS @10	HITS @25	MAP	HITS @1	HITS @5	HITS @20	HITS @25	Genre F1	Style F1
melody2vec$_F$	0.646	0.578	0.717	0.774	0.867	-	-	-	-	-	0.649	0.299
melody2vec$_B$	0.641	0.571	0.712	0.772	0.866	-	-	-	-	-	0.647	0.293
tonnetz	0.683	0.545	0.865	0.946	0.993	0.423	0.101	0.407	0.628	0.897	0.627	0.253
pianoroll	0.762	0.645	0.916	0.967	0.995	0.567	0.166	0.541	0.720	0.921	0.640	0.365
PiRhDy$_{GH}$	0.858	0.775	0.966	0.988	0.999	0.651	0.211	0.625	0.812	0.965	0.663	0.448
PiRhDy$_{GM}$	0.971	0.950	0.995	0.998	0.999	0.567	0.184	0.540	0.718	0.919	0.668	0.471
MusicBERT$_{small}$	0.979	0.966	0.995	0.998	**1.000**	0.920	0.325	0.834	0.991	0.996	0.762	0.604
MusicBERT$_{base}$	**0.984**	**0.973**	**0.997**	**0.999**	**1.000**	**0.945**	**0.333**	**0.856**	**0.995**	**0.998**	**0.784**	**0.651**

SOTA accuracy on various music understanding tasks

SongMASS: Automatic Song Writing with Masked Sequence to Sequence Pre-training, AAAI 2021

- Background
  - Lyric-to-melody and melody-to-lyric generation are two important tasks for song writing
  - Lyric and melody are weakly coupled, lack of data → **pre-training**
  - Lyric and melody are strictly aligned → **attention alignment**

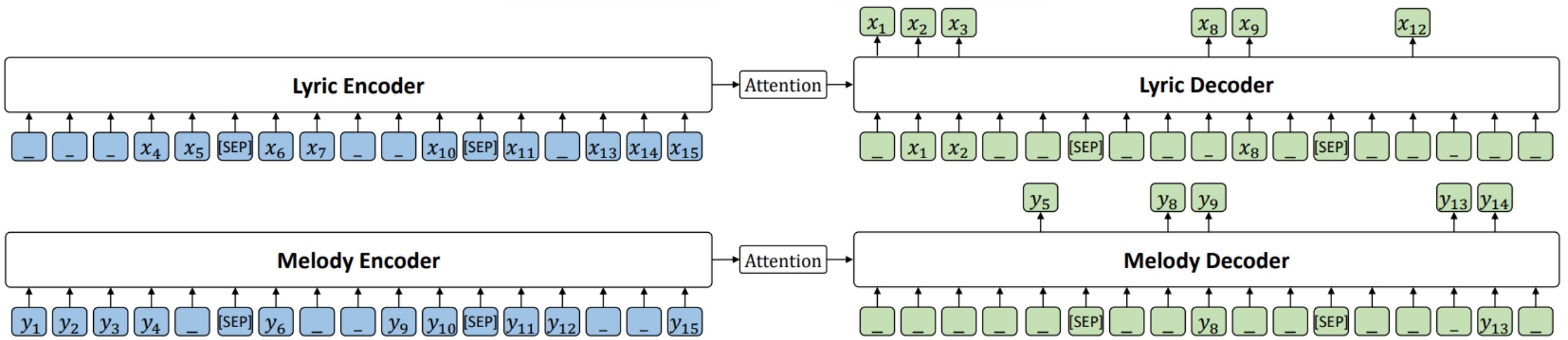


Paired Aligned Data :

Lyric	**Another**				**day**	**has**	**gone**	**I'm**		**still**	**alone**	
Pitch	R	G3	E4	D4	C4	B3	C4	R	E4	C4	B3	C4
Duration	$\frac{7}{16}$	$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{3}{16}$	$\frac{1}{16}$	$\frac{5}{16}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{3}{16}$	$\frac{1}{16}$	$\frac{5}{16}$

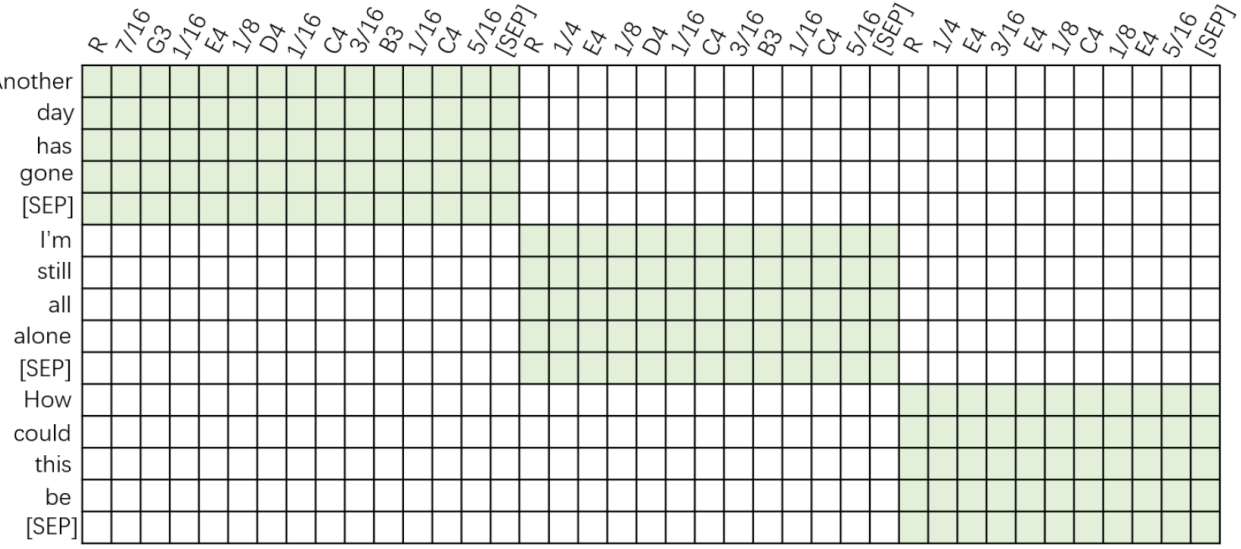
SongMASS

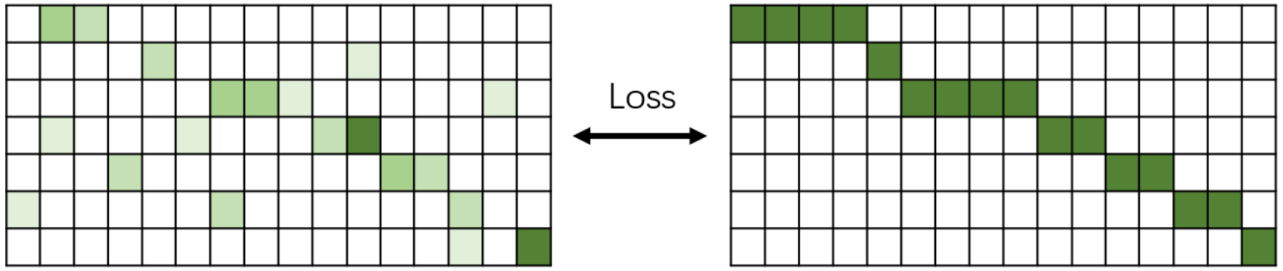
- Masked sequence to sequence (MASS) pre-training
  - Document-level MASS, mask each a segment in each sentence and predict all segments in the target
  - Separate encoder and decoder, add supervised loss to guide the pre-training



SongMASS

- Lyric and melody alignment
  - In training, sentence-level and token-level alignment with attention constraint
  - In inference, sentence-level with SEP token, token-level with dynamic programming

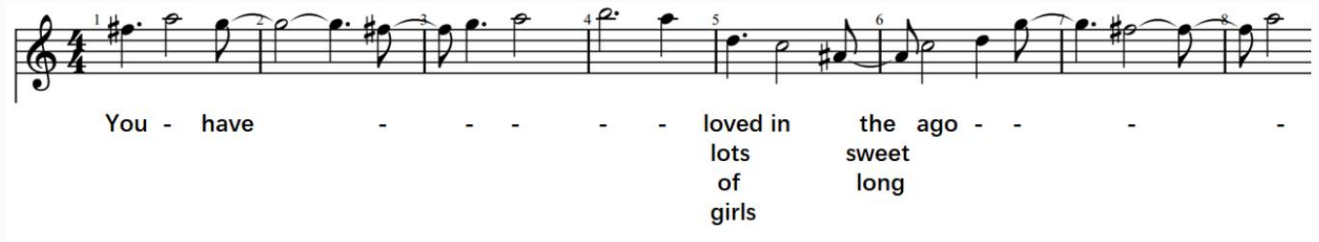




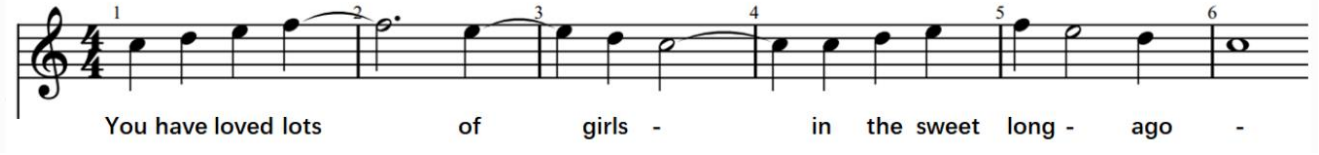
SongMASS

- Demo

Baseline



SongMASS



https://speechresearch.github.io/songmass/

```
1 3 5 3  2     1    6  1
you have loved lots of girls
1  1   7     6    5 3 6
in the sweet long ago
1   -    1   7   6     5      3  6
and each one has meant heaven to you
3   5 5 3 2 1     6    1
you have    vowed your affection
1  1    7   6  5 3
to each one in turn
3   3    5     3  2    1 6 1
and have sworn to them be  true
6 6 6 5    5 3    2   1
you   have kissed the moon
1     1   7     7      6 5 3
while the world seemed in  tune
6    3    3   5  3    2 1   2
then left her to hunt a new game
1    3 5 3    2     1  6   1
does it  ever occur to you later
1  2 1 3
my boy
1 2 1 3 2 1 3   2
that      doing the
6 6      5       5 3 2 1
i wonder kissing her   now
6 1    1        2 1 3
wonder teaching her
1      2       1    3   -
wonder looking into her eyes
1         6     -       1
breathing sighs telling lies
1 1      7      6   5 3 6
i wonder buying the wine
1   1    7    6 5    3  -    6
```

DeepRapper: Neural Rap Generation with Rhyme and Rhythm Modeling, ACL 2021

- Explore a new lyric-melody relationship: Rap
- Rap is a musical form of vocal delivery that incorporates “rhyme, rhythmic speech, and street vernacular”
  - Originated in America in the 1970s
  - Popular in the world especially in young people
- Hip-Hop
  - 1970s originated from New York, young people in African-American and Latino
  - Street culture
  - Four elements in Hip-Hop
    - DJ (Disc Jockey)
    - Rap (MC)
    - Street Dance (B-Boy)
    - Graffiti

DeepRapper

- Lyric with Rhyme and Rhythm, and sing out
  - Rhyme and Rhythm (beat) is important
  - Rap cares more about beat/duration, rather than pitch (melody)
- However, previous works on rap generation only consider rhyme, but ignores rhythm
  - How they control rhyme? Use Rhyme list. Complicated and not learned end-to-end
  - No rhythm/beat information, cannot be directly used!

DeepRapper

- Model

Dataset	#Songs	#Sentences
D-RAP	16,246	832,646
D-SONG	52,737	2,083,143
D-LYRIC	272,839	9,659,503

DeepRapper

- Generated results
  - N-Rhyme: 1-rhyme, 2-rhyme, multi-rhyme
    - 下苦功 练武功 变武松
  - Diversity of rhyme

- Demo
  - https://deeprapper.github.io/

Summary

• Time-efficient ML	
• **FastSpeech 1/2** (TTS, NeurIPS 2019/ICLR 2021)	AR→NAR
• **FastCorrect 1/2** (ASR, paper submission)	AR→NAR
• **PriorGrad** (TTS, paper submission)	Diffusion
• Memory/Computation-efficient ML	
• **LightSpeech** (TTS, ICASSP 2021)	NAS
• **AdaSpeech** (TTS, ICLR 2021)	Cross-speaker/domain-knowledge
• Data-efficient ML	
• **AdaSpeech 2/3** (TTS, ICASSP 2021/INTERSPECH 2021)	Cross-speaker/domain-knowledge
• **LRSpeech** (TTS/ASR, ICML 2019/KDD 2020)	Cross-lingual/KD/BT
• **MixSpeech** (ASR, ICASSP 2021)	Domain-knowledge
• **SongMASS** (Music, AAAI 2021)	Pre-training/domain-knowledge
• **MusicBERT** (Music, ACL 2021)	Pre-training
• **DeepRapper** (Music, ACL 2021)	Pre-training

Summary

- Language, speech, and music related tasks are widely found in product service
- For practical applications, efficient machine learning is critical

- Accuracy vs Data, Model, Computation

Summary

- Hammer ：Machine Learning Tools
- Nail: Application Tasks

- If you have a hammer, everything looks like a nail
- If you need to knock in a nail, everything looks like a hammer

- A hammer in your hand, but no hammer in your mind
- Connect hammers to nails

Thank You!

Xu Tan/谭旭
Senior Researcher @ Microsoft Research Asia
xuta@microsoft.com

https://www.microsoft.com/en-us/research/people/xuta/
https://speechresearch.github.io/

Homepage

Speech Research Demo