Speech Technologies

From basics to trends

HUAWEI TECHNOLOGIES CO. LTD (Russia)

Mikhail Kudinov

www.huawei.com

HUAWEI

Contents

Introduction

SPEECH TECHNOLOGIES

Speech Technologies in a Nutshell

- **Speech technology** is a type of computing technology that enables an electronic device to recognize, analyze and understand spoken word or audio containing human speech
- Subfields:
  - ⇨ Speech synthesis
  - ⇨ Speech recognition
  - ⇨ Speaker recognition/verification
  - ⇨ Speech coding
  - ⇨ Voice conversion

Speaker recognition

- **Speaker identification:** identify speaker from a known set by voice in the record
- **Speaker verification:** identify if the voice in the record belongs to the target speaker

  ⇨ Voice biometry/access control for Barclays
- **Speaker diarization:** segment a long speech record into parts according to which person is speaking now

Speech synthesis / voice conversion

- **Speech synthesis** in narrow sense is a generation of a speech signal based on text input
- **Voice conversion** is a transformation of a speech signal preserving linguistic information
- **Accessibility** for visual impaired users
  - ⇨ Microsoft Windows accessibility screen reader
- **Text readers**
  - ⇨ Xinhua news presenter
- **Accessibility** for speech-impaired users
  - ⇨ Deepmind’s initiative for speech-impaired users
- **Voice assistants**:
  - ⇨ Alexa, Siri, Cortana, Bixby…

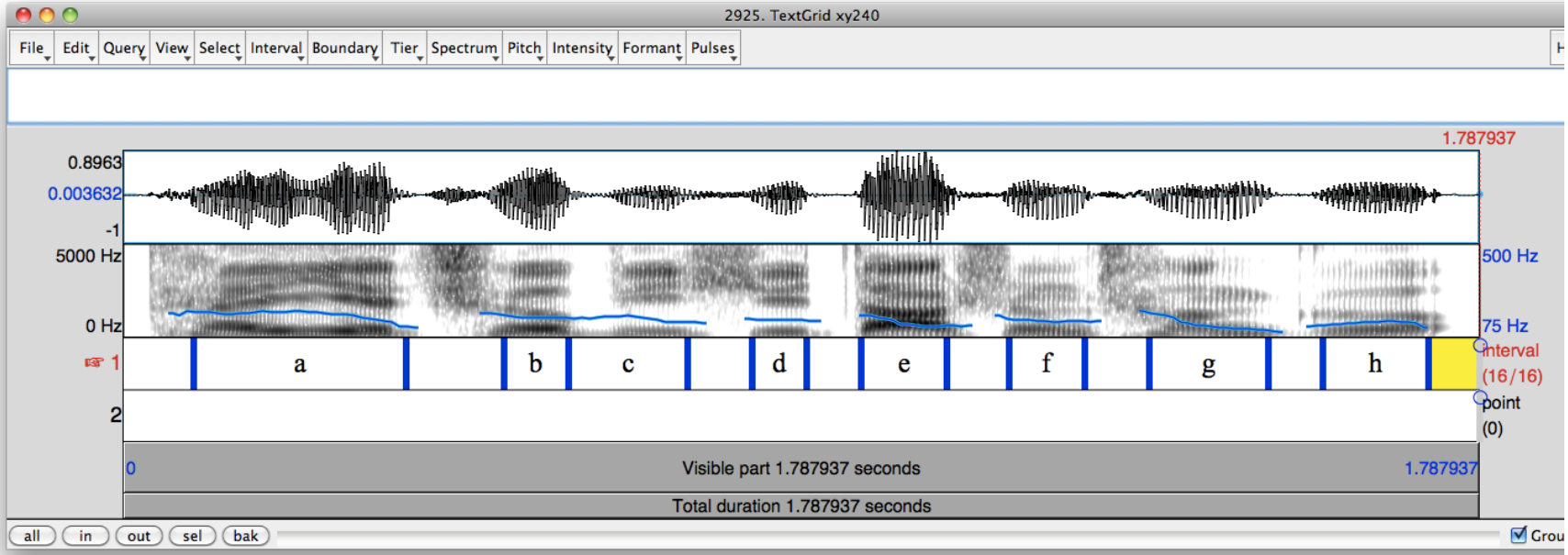
Speech recognition

- **Speech recognition** is a translation of a spoken utterance into text
- **Key-word spotting**:
  - ⇨ Wakeup phrases: Okay, Google!
- **Small vocabulary SR**:
  - ⇨ Voice dial
- **Large-vocabulary** SR:
  - ⇨ Speaker dependent: call center solution for Barclays
  - ⇨ Speaker independent: voice commands, voice search, voice assistants, dictation task

Speech representation and perception

WHAT IS SPEECH?

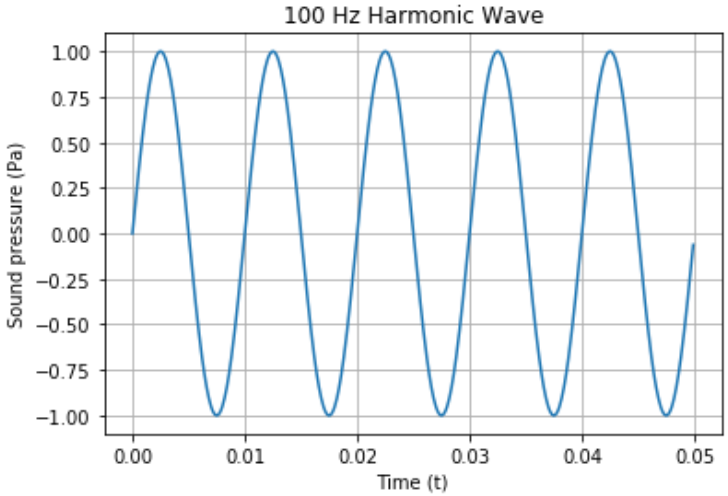
Waveform – Spectrogram – Text



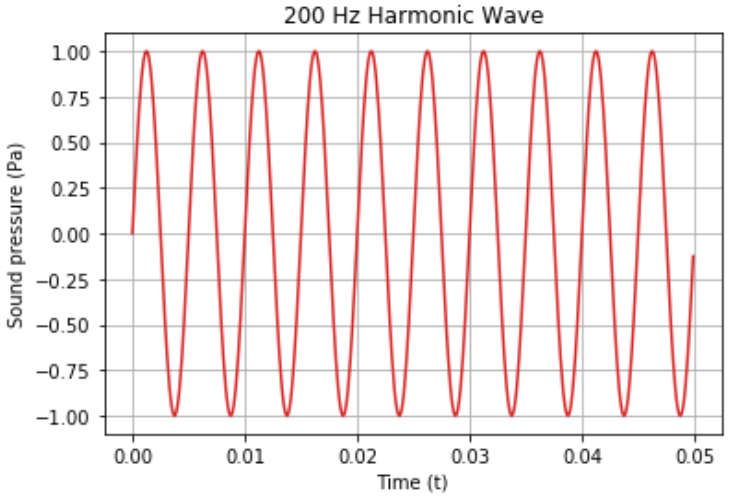
- Waveform: sound intensity as a function of time
- Spectrogram: short-time spectrum as a ‘function’ of a time step
- Text: linguistic meaning corresponding to a speech fragment

From sound wave to spectrogram:

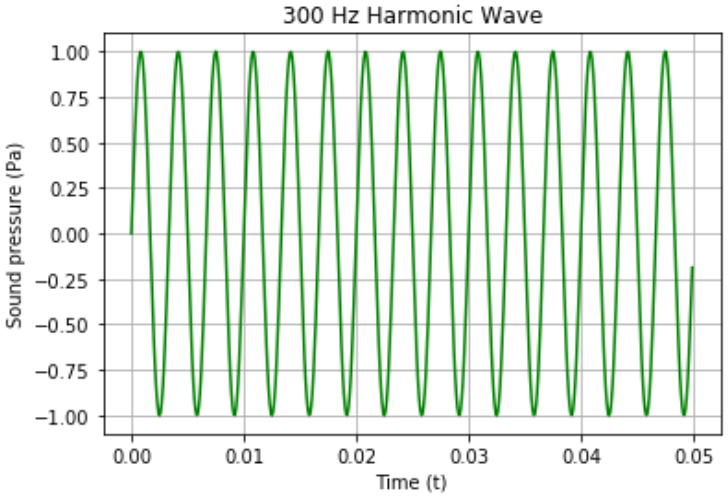
harmonic oscillations



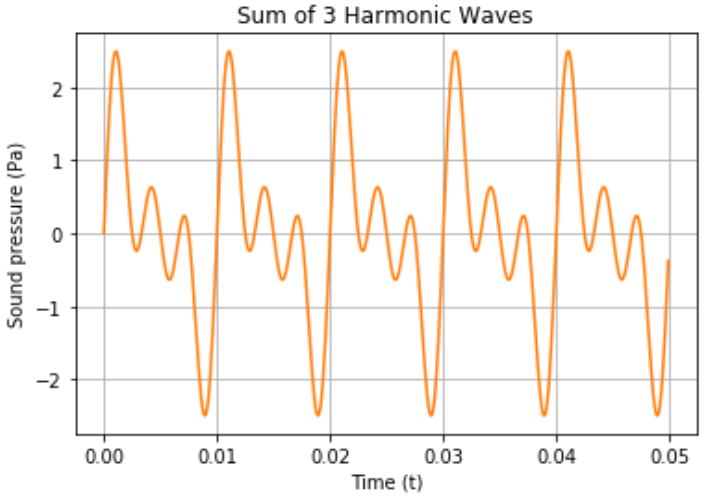
+



+

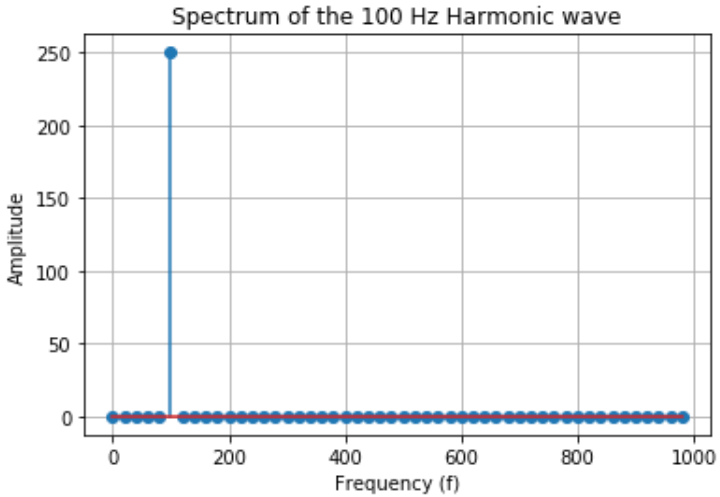


=

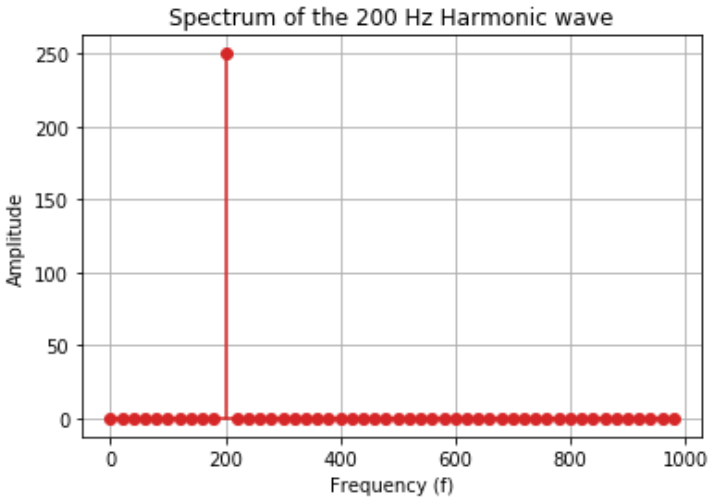


From sound wave to spectrogram:

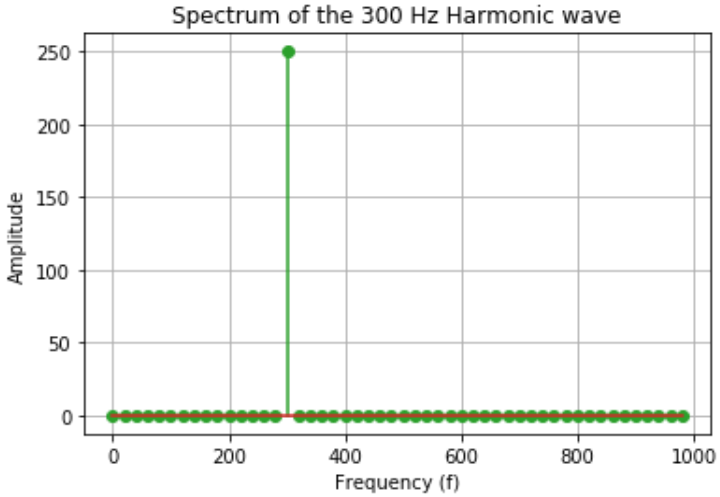
sum of spectra



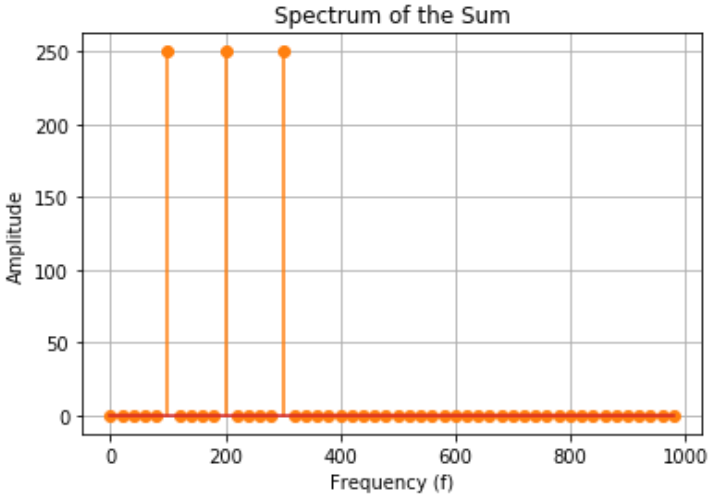
+



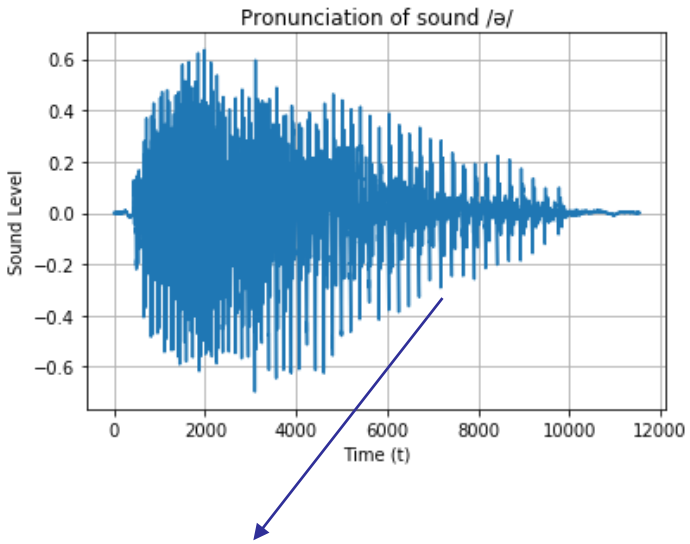
+



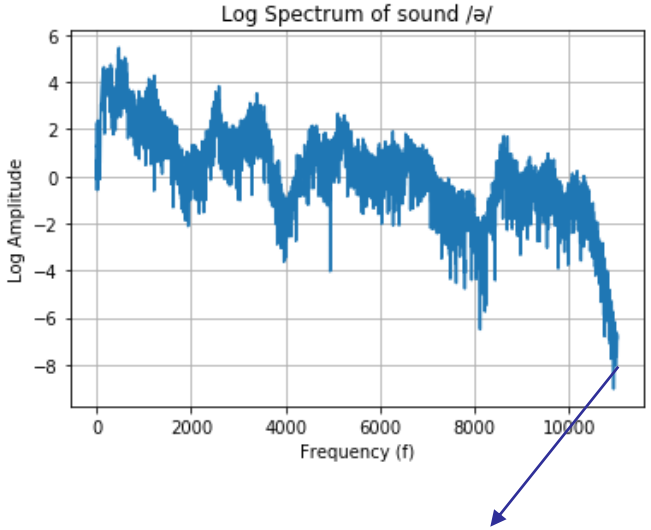
=



From sound wave to spectrogram: phoneme /ə/



Discrete Fourier Transform (DFT)*



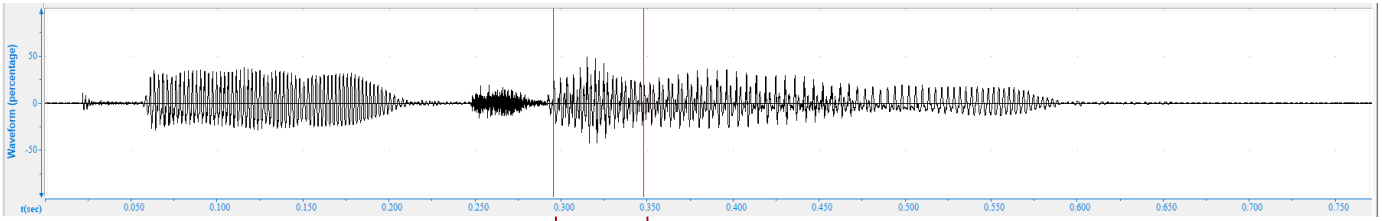
Digital signal is time-discretized so that each of the signal is represented by N samples. Number of samples per second is called *sampling rate*

11 kHz is the highest frequency (*Nyquist frequency*) which can be present in the signal at 22 kHz sampling rate

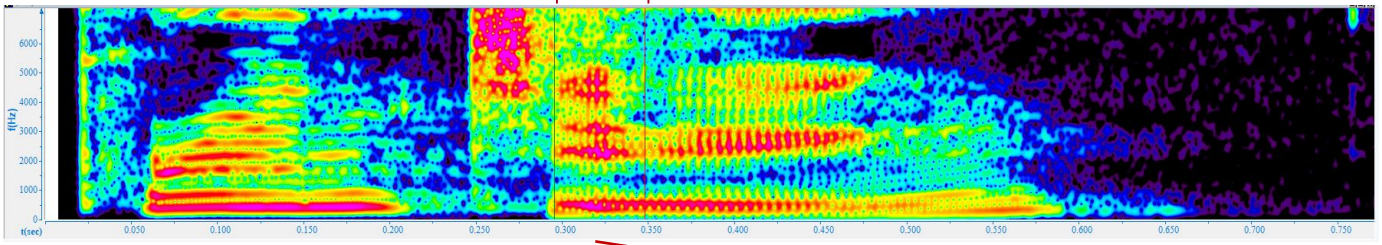
*actually this is half of log |FFT(x)|

From sound wave to spectrogram:

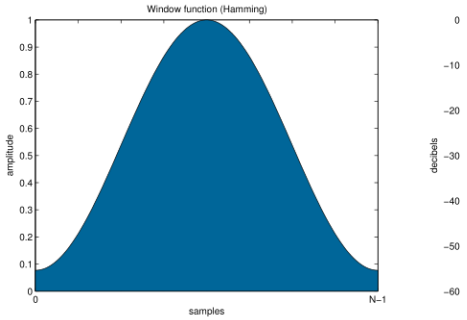
Short-time Fourier Transform



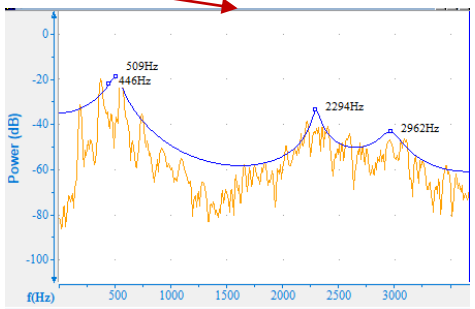
Waveform



Spectrogram



Windowing function is used to generate ‘smooth’ STFT

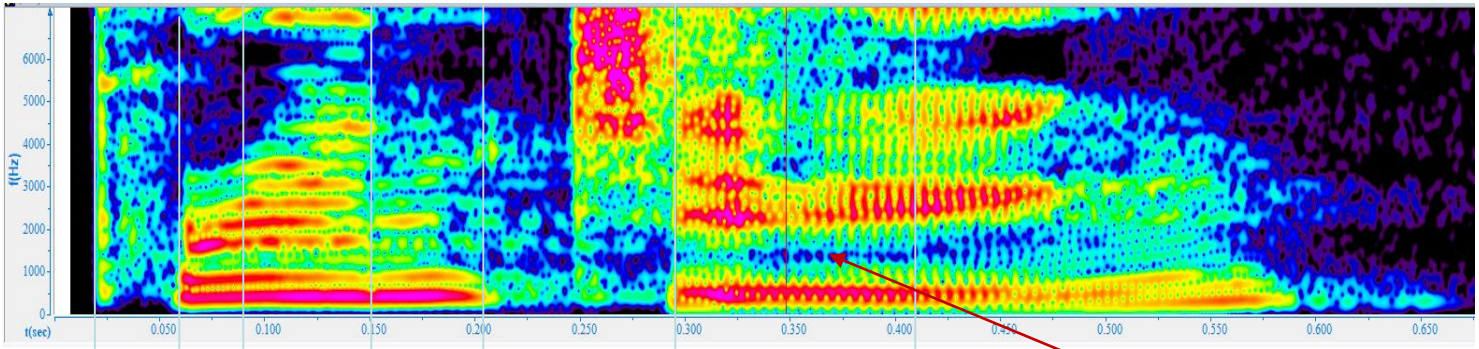


Fast Fourier Transform
(10-30 msec segment)

Spectrum of a segment

Spectrum (Fast Fourier Transform)

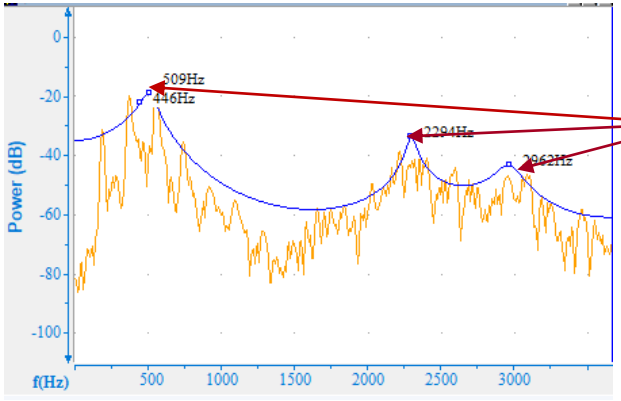
Closer look at the spectrogram



Stop sounds /p/, /t/ have a short pause and a noise segment

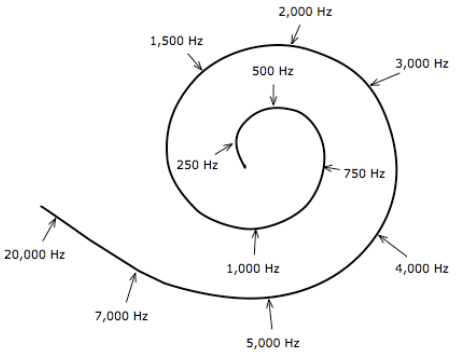
There is no clear border between sounds
But the neighboring sounds affect spectra of each other
Voiced sounds have high peaks in a low frequency range

English phoneme /i:/



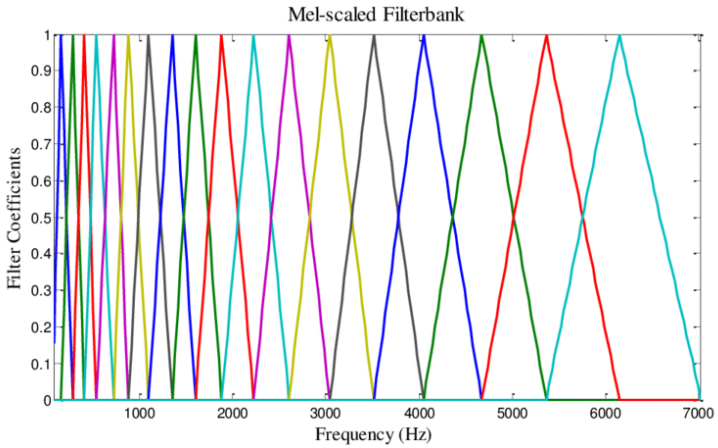
Formant frequencies

Cochlea carries out Fourier Transform in a human inner ear

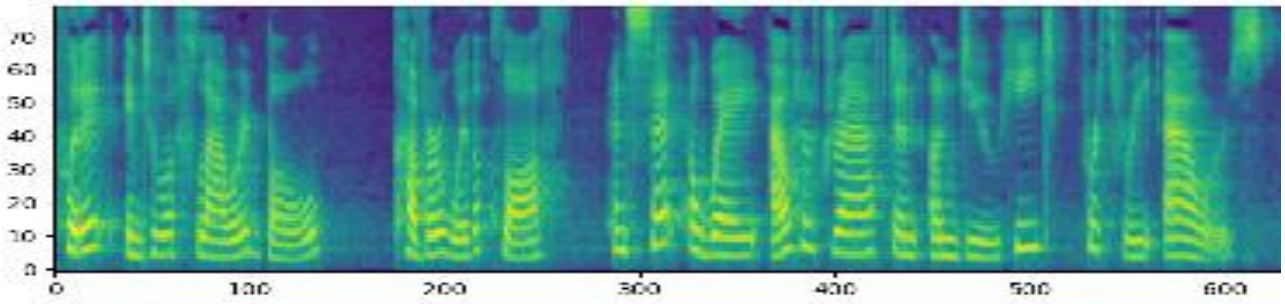


Perceptually motivated features: Mel-scale

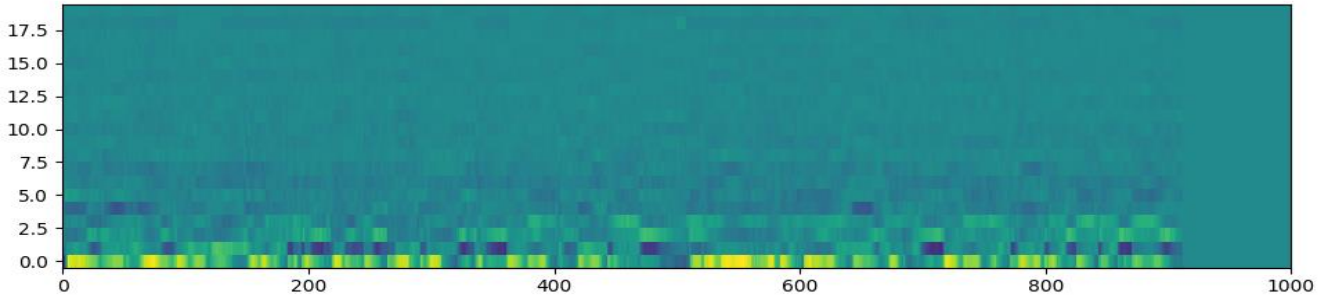
- FFT coefficients are redundant
- Sound perception by the human auditory system is highly non-linear
- Idea #1: compress spectrum into a small number of energies in critical subbands
  - ⇨ Mel-scale spectral coefficients (MFSC)
  - ⇨ Bark-scale spectral coefficients
- Idea #2: decorrelate log(MFSC) or log(BFSC) in PCA-like manner (use Discrete Cosine Transform)
  - ⇨ Mel-frequency cepstral coefficients: MFCC
  - ⇨ Bark-frequency cepstral coefficients: BFCC



Acoustic representations: MFSC vs. BFCC

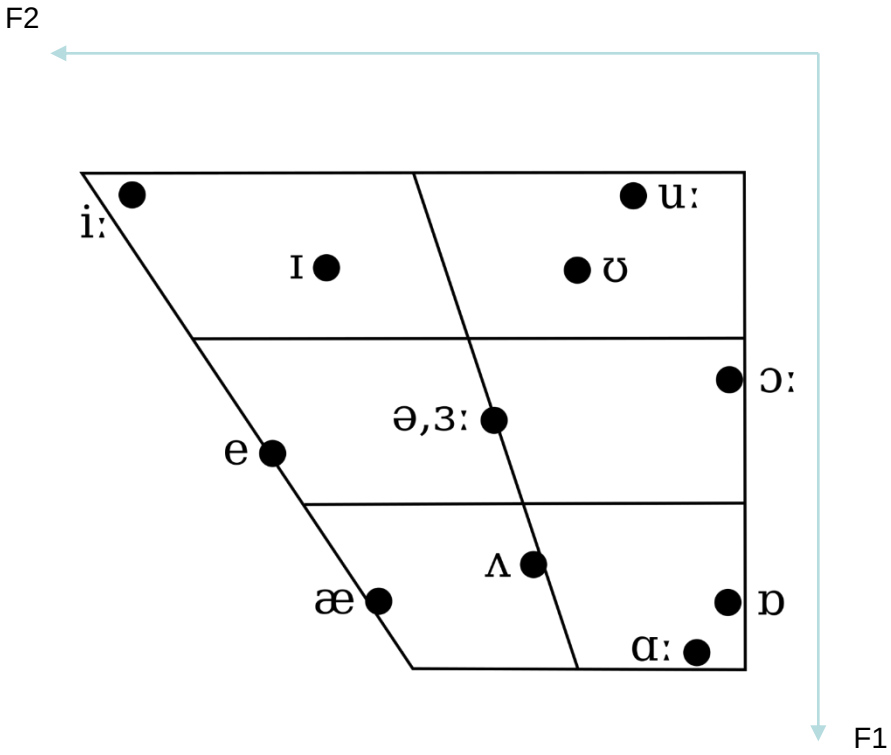


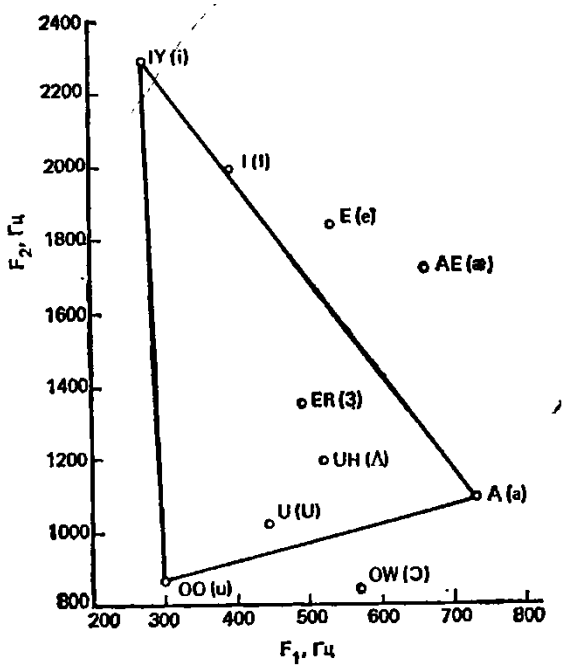
80-band MFSC



18-band Bark-scale cepstral coefficients

From spectrogram to text: phonetics (vowels)





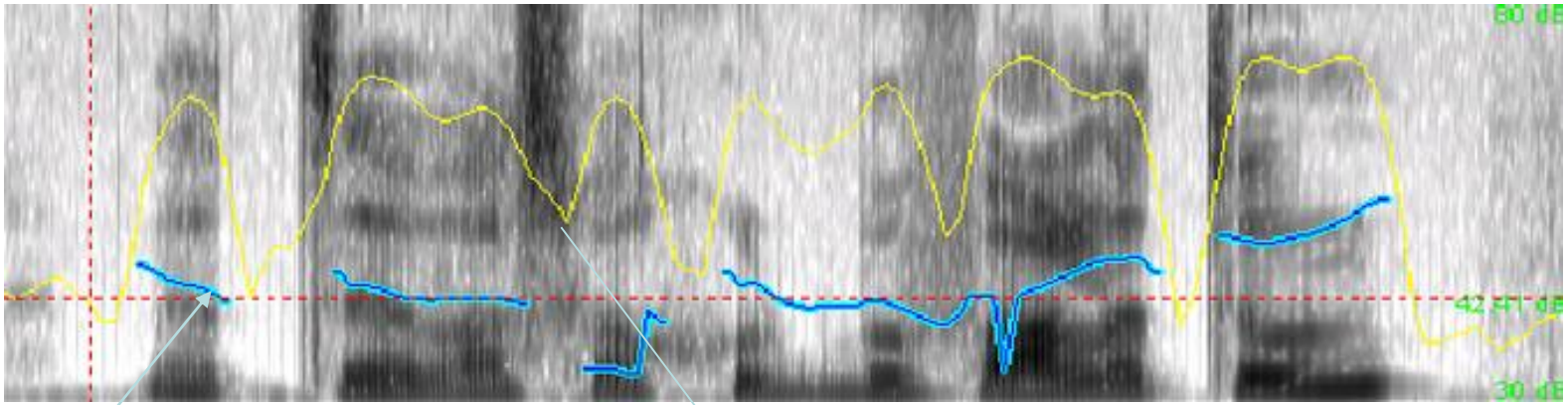
From spectrogram to text: phonetics (consonants)

the international phonetic alphabet (2005)

consonants (pulmonic)	LABIAL		CORONAL				DORSAL				RADICAL		LARYNGEAL
	Bilabial	Labio-dental	Dental	Alveolar	Palato-alveolar	Retroflex	Alveolo-palatal	Palatal	Velar	Uvular	Pharyngeal	Epi-glottal	Glottal
Nasal	m	ɱ		n		ɳ		ɲ	ŋ	ɴ			
Plosive	p b			t d		ʈ ɖ		c ɟ	k g	q ɢ		ʡ	ʔ
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ɕ ʑ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	ʜ ʢ	h ɦ
Approximant		ʋ		ɹ		ɻ		j	ɰ				
Tap, flap		ⱱ		ɾ		ɽ							
Trill	ʙ			r						ʀ		ʀ	
Lateral fricative				ɬ ɮ				ʎ̝	ʟ̝				
Lateral approximant				l		ɭ		ʎ	ʟ				
Lateral flap				ɺ		ɭ̆							

Where symbols appear in pairs, the one to the right represents a modally voiced consonant, except for murmured ɦ. Shaded areas denote articulations judged to be impossible. Light grey letters are unofficial extensions of the IPA.

Beyond text: speech prosody



- Speech is not just a sequence of phonemes. The *supersegmental* features of the utterance are extremely important
  - Pitch (fundamental frequency)
  - Loudness (energy)
  - Rhythm (distribution of lengths of the spoken sounds)

Speech synthesis

TEXT-TO-SPEECH

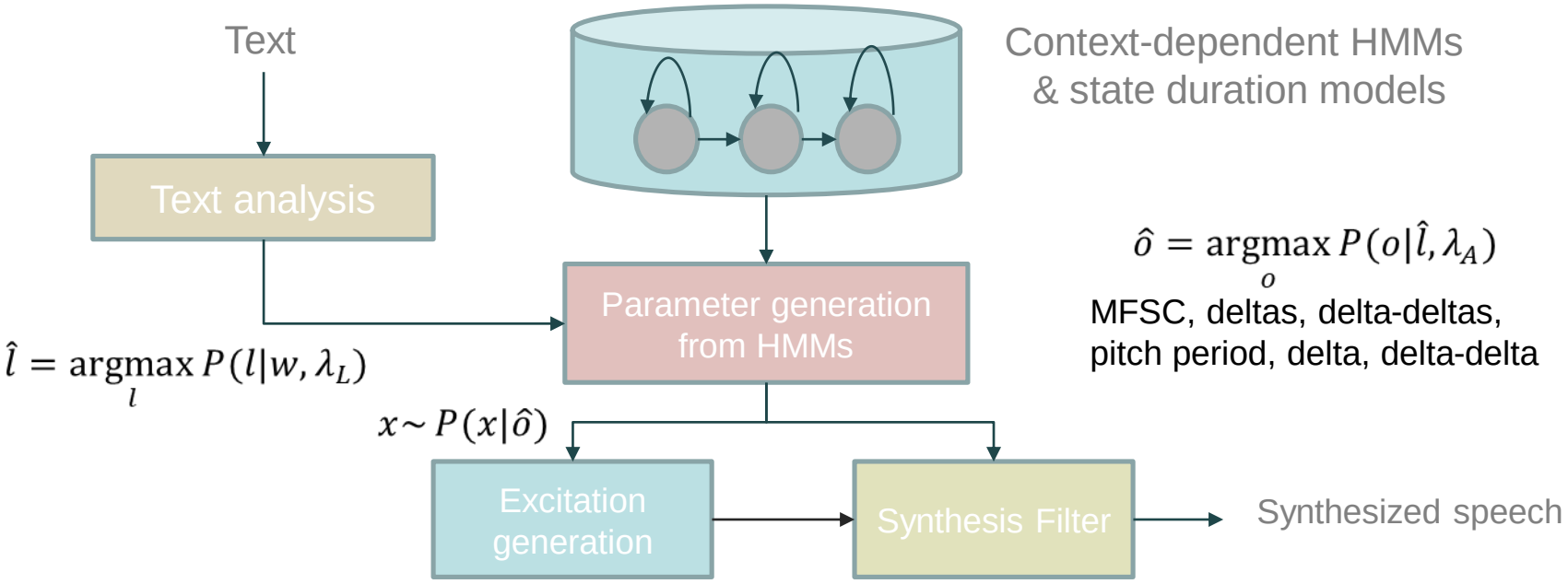
Speech synthesis approaches

- Rule-based, formant synthesis
  - ⇨ Phonetic units are generated according to hand-crafted rules
- Corpus-based, concatenative synthesis
  - ⇨ Concatenate speech units from a database
    - − Diphone synthesis
    - − Unit selection synthesis
- Corpus-based statistical synthesis
  - ⇨ Feature-generation+vocoder
    - − HMM
    - − DNN
  - ⇨ E2E systems

Statistical speech synthesis

- Given a training database consisting of pairs of speech waveforms ***X*** and texts ***W*** estimate a statistical model parameters λ for generating speech waveform **x** for text $\boldsymbol{w} \notin \boldsymbol{W}$: $x \sim P(\boldsymbol{x}|\boldsymbol{w}, \lambda)$
- Usually $P(\boldsymbol{x}|\boldsymbol{w}, \lambda)$ is decomposed into submodules
  - ⇨ $P(\boldsymbol{x}|\boldsymbol{w}, \lambda) = p(\boldsymbol{x}|\boldsymbol{f})\ p(\boldsymbol{f}|\boldsymbol{l}, \lambda_A)\ p(\boldsymbol{l}|\boldsymbol{w}, \lambda_L)$
  - ⇨ **f**: parameteric representation of speech waveform x
  - ⇨ **l**: linguistic feature
  - ⇨ $\lambda = \{\lambda_A, \lambda_L\}$: generative model parameter
    - λ_A : acoustic model parameter
    - λ_L: linguistic model parameter

Statistical speech synthesis before Tacotron era

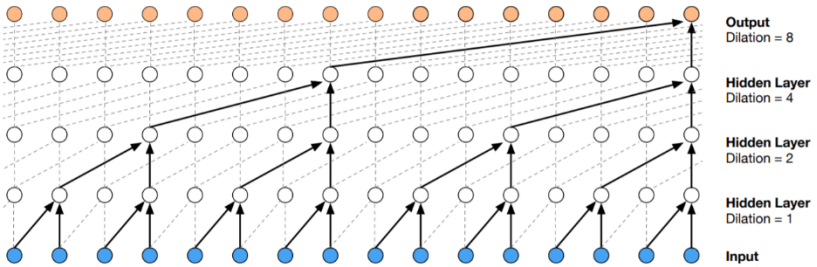


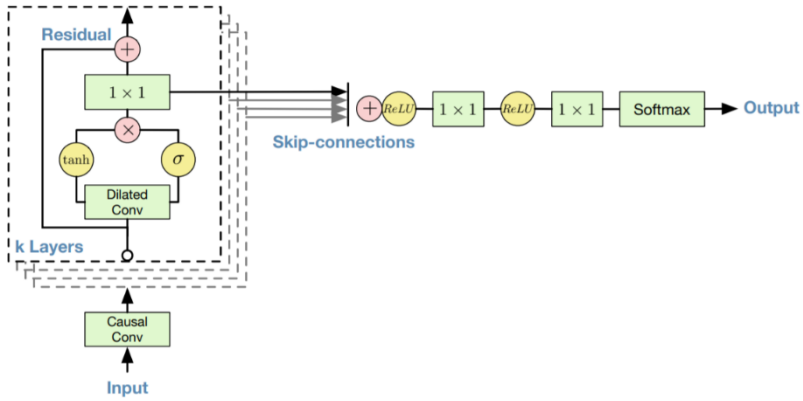
Neural vocoder: WaveNet [Oord17]

- Autoregressive model based
- Main building element: residual block with dilated convolutions
- 30 residual blocks
- Predicts mu-law companded outputs

$$\Rightarrow \frac{\ln(1+\mu|x|)}{1+\ln(1+\mu)}$$

- 16kHz sound
- Best RTF is ~3.0 (Nvidia P100) [Kal18]



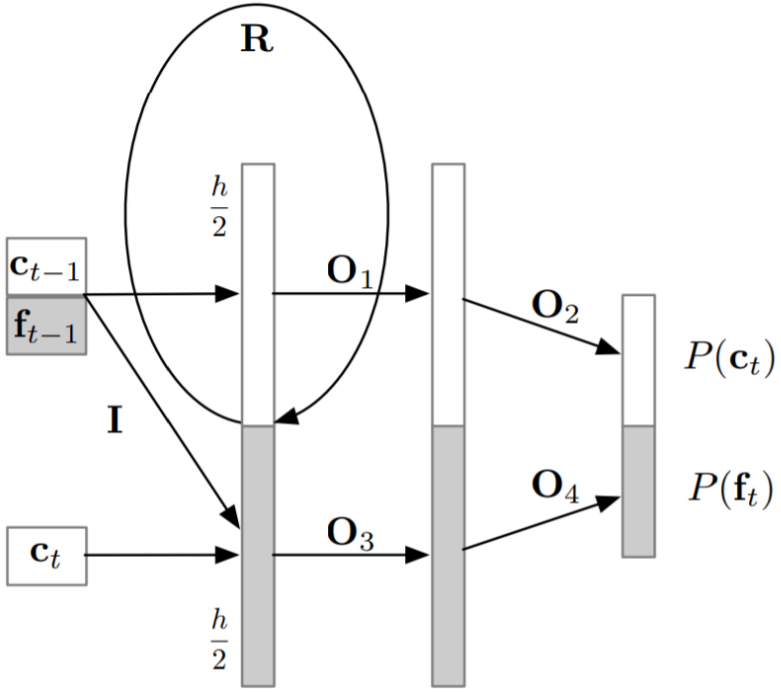


Parallel WaveNet, ClariNet (and WaveGlow)

- Idea #1: use trained autoregressive model as a reference distribution (teacher) to trained a non-autoregressive vocoder (based on IAF) via KL-divergence minimization
- Parallel WaveNet [Oord&Li17]
  - ⇨ MoL
  - ⇨ Power loss, Style loss, Contrastive loss
- ClariNet [Ping19]
  - ⇨ Normal distribution
  - ⇨ STFT loss (almost like power loss)
  - ⇨ Tricks with KL divergence regularization
- WaveGlow [Pren18] uses Glow network and is trained from scratch

Recurrent Vocoders: WaveRNN

- Recurrent neural network conditioned on linguistic features (so, like original WaveNet it's not actually a vocoder) [Kal18]
- Quality comparable to WaveNet but RTF=0.25 on GPU at 24kHz
- 16-bit prediction via 2-step prediction: coarse (first 8-bit) and fine (second 8-bit) parts
- Block sparsification



LPCNet: speech coding meets deep learning

- We can estimate a_k by minimizing energy of the residual

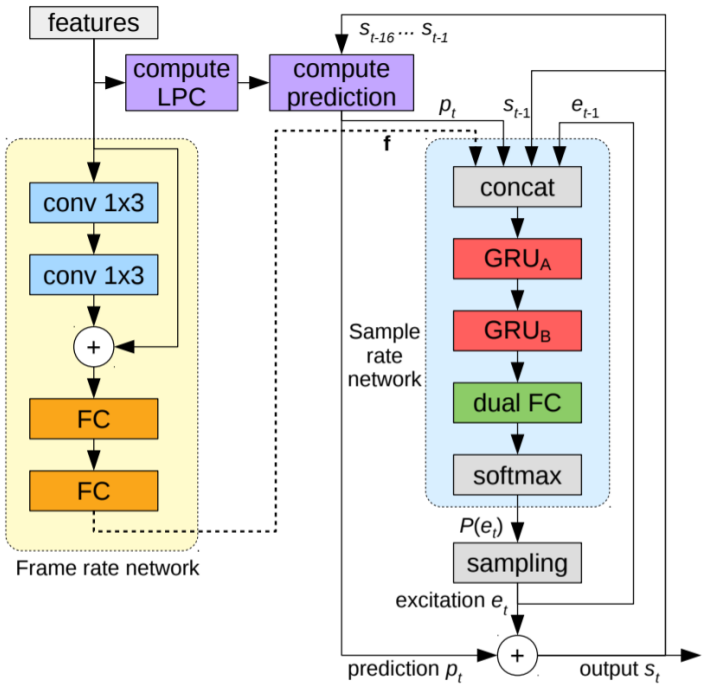
$$E = \sum_n e^2(n) = \sum_n (s(n) - \tilde{s}(n))^2 = \sum_n (s(n) - \sum_{k=1}^{p} a_k s(n-k))^2$$

  ⇨ which is equivalent to MSE minimization in the analysis window

- LPCs can be found in both time and frequency domain i.e. from spectrum
- If we know e(n) and a_k for the speech frame we can restore s(n)
- Small module of e(n) is a resource of compression in speech codecs

LPCNet

- Compute LPCs from Bark-scale cepstral coefficients
- Use frame-rate network to compute feature vector **f**
- Feed **f** to a sample rate recurrent network predicting excitation signal e(n)
- Output signal is computed as if it was restored from true LPCs and excitation signal
- Operates on 8kHz, 8-bit mu-law
- RTF: ~0.2 on CPU



Val[19]

HiFi-GAN [Kong20]

- Issue:
  - ⇨ GAN’s improve the sampling efficiency/memory usage, but the quality is still not acceptable
- Problem statement:
  - ⇨ Propose GAN, that will generate high-fidelity waveforms comparable with AR and Flow-based models
- Solution:
  - ⇨ Modelling periodic patterns of audio for enhancing quality

HiFi-GAN: Architecture (generator)

- Generator:
  - fully convolutional
  - upsamples mel to match the length of audio
  - ConvTranspose + Multi-Receptive Field Fusion module (MRF)
  - MRF: resblock with diverse receptive field patterns

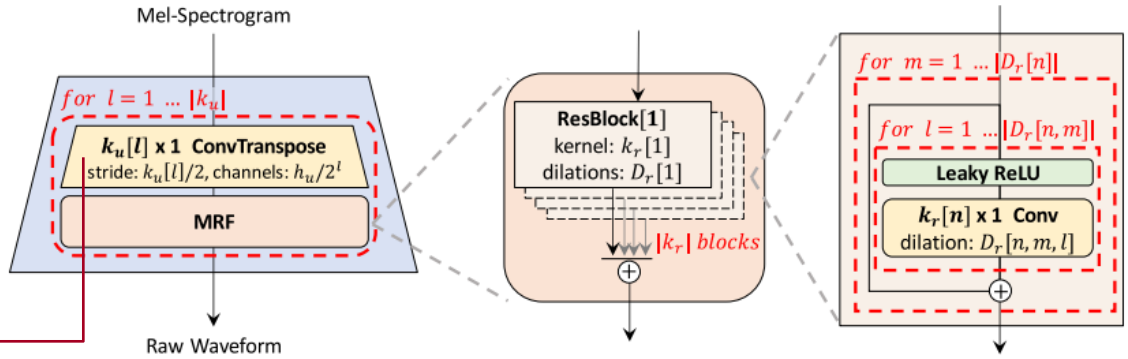


Figure 1: The generator upsamples mel-spectrograms up to $|k_u|$ times to match the temporal resolution of raw waveforms. A MRF module adds features from $|k_r|$ residual blocks of different kernel sizes and dilation rates. Lastly, the n-th residual block with kernel size $k_r[n]$ and dilation rates $D_r[n]$ in a MRF module is depicted.

$L_{out}=(L_{in}-1)\times stride-2\times padding+dilation\times(kernel\_size-1)+output\_padding+1$

HiFi-GAN: Architecture (discriminator)

- Multi-Period Discriminator:
  - mixture of sub-discriminators
  - each one works with given period *p*
  - used periods: [2, 3, 5, 7, 11]
- Multi-Scale Discriminator:
  - drawn from MelGAN paper
  - mixture of sub-discriminators
  - each one operates on different scales
  - scales: [raw, ×2, ×4]

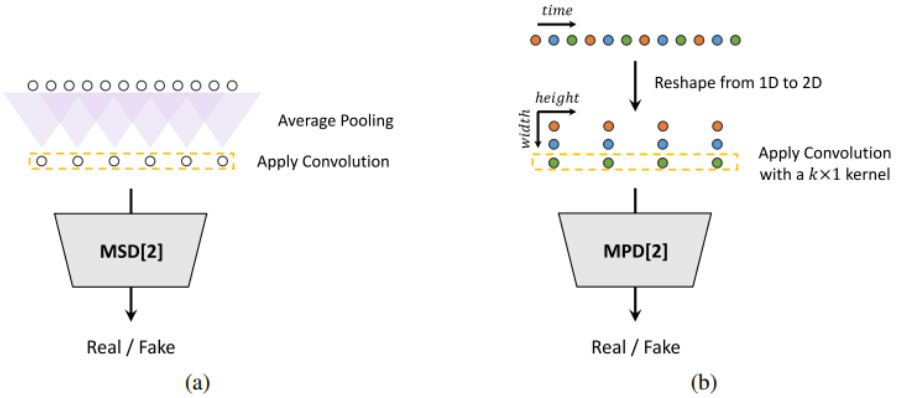


Figure 2: (a) The second sub-discriminator of MSD. (b) The second sub-discriminator of MPD with period 3.

HiFi-GAN: Results

Table 1: Comparison of the MOS and the synthesis speed. Speed of n kHz means that the model can generate $n \times 1000$ raw audio samples per second. The numbers in () mean the speed compared to real-time.

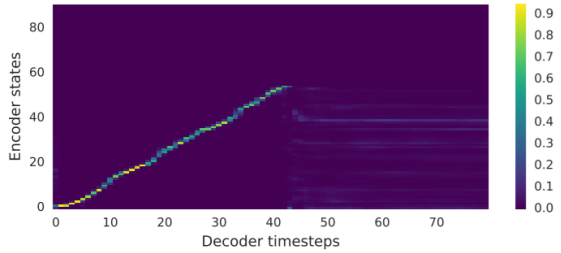
Model	MOS (CI)	Speed on CPU (kHz)	Speed on GPU (kHz)	# Param (M)
Ground Truth	4.45 (±0.06)	–	–	–
WaveNet (MoL)	4.02 (±0.08)	–	0.07 (×0.003)	24.73
WaveGlow	3.81 (±0.08)	4.72 (×0.21)	501 (×22.75)	87.73
MelGAN	3.79 (±0.09)	145.52 (×6.59)	14,238 (×645.73)	4.26
HiFi-GAN $V1$	**4.36** (±0.07)	31.74 (×1.43)	3,701 (×167.86)	13.92
HiFi-GAN $V2$	4.23 (±0.07)	214.97 (×9.74)	16,863 (×764.80)	**0.92**
HiFi-GAN $V3$	4.05 (±0.08)	**296.38 (×13.44)**	**26,169 (×1,186.80)**	1.46

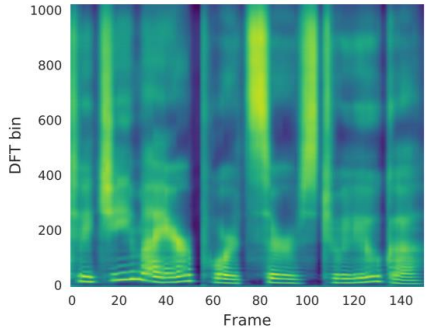
Table 2: Ablation study results. Comparison of the effect of each component on the synthesis quality.

Model	MOS (CI)
Ground Truth	4.57 (±0.04)
Baseline (HiFi-GAN $V3$)	4.10 (±0.05)
w/o MPD	2.28 (±0.09)
w/o MSD	3.74 (±0.05)
w/o MRF	3.92 (±0.05)
w/o Mel-Spectrogram Loss	3.25 (±0.05)
MPD p=[2,4,8,16,32]	3.90 (±0.05)
MelGAN	2.88 (±0.08)
MelGAN with MPD	3.35 (±0.07)

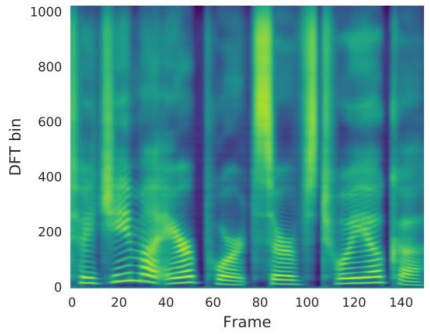
Neural feature-generation: Tacotron/Tacotron2

- Seq2Seq model with attention
  - ⇨ Content-based in Tacotron [Wang17]
  - ⇨ Location-sensitive attention in Tacoton2 [Shen18]
- Predicts 80-band MFSC
  - ⇨ Griffin-Lim vocoder in Tacotron
  - ⇨ WaveNet in Tacotron2
- Predicting N frames per inference step
  - ⇨ N = 2-3 in Tacotron
  - ⇨ N = 1 in Tacoton2 (as reported in [Shen18])
- Makes use of so-called PostNet increasing frequency resolution
- Tacotron2: separate stop-token prediction

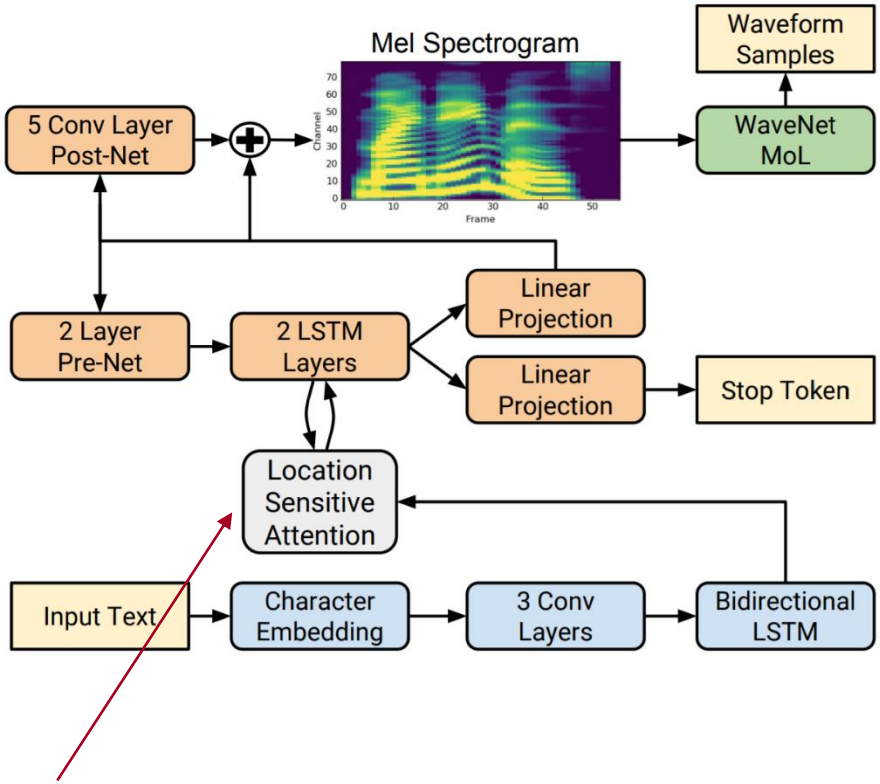




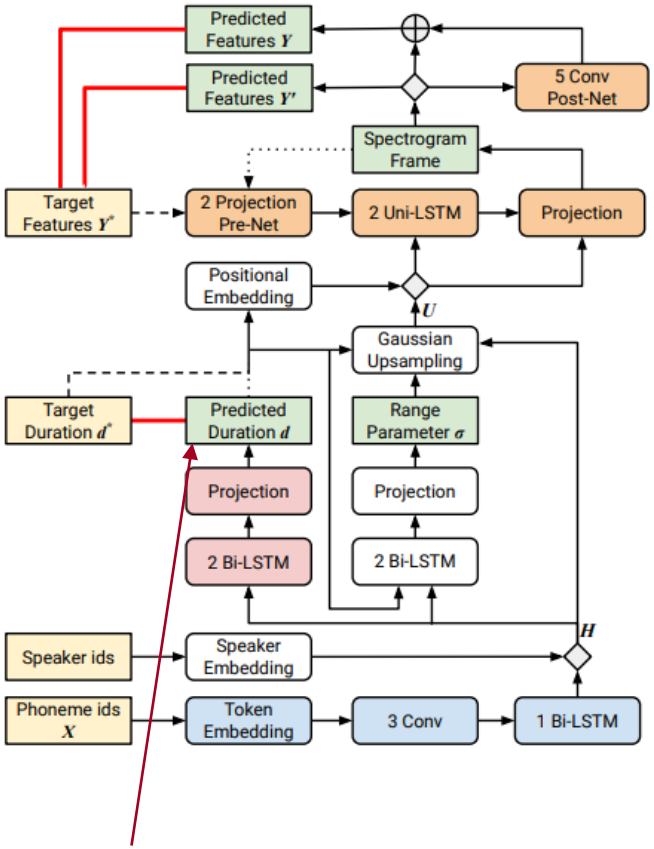
(a) Without post-processing net



Tacotron2 and Non-attentive Tacotron

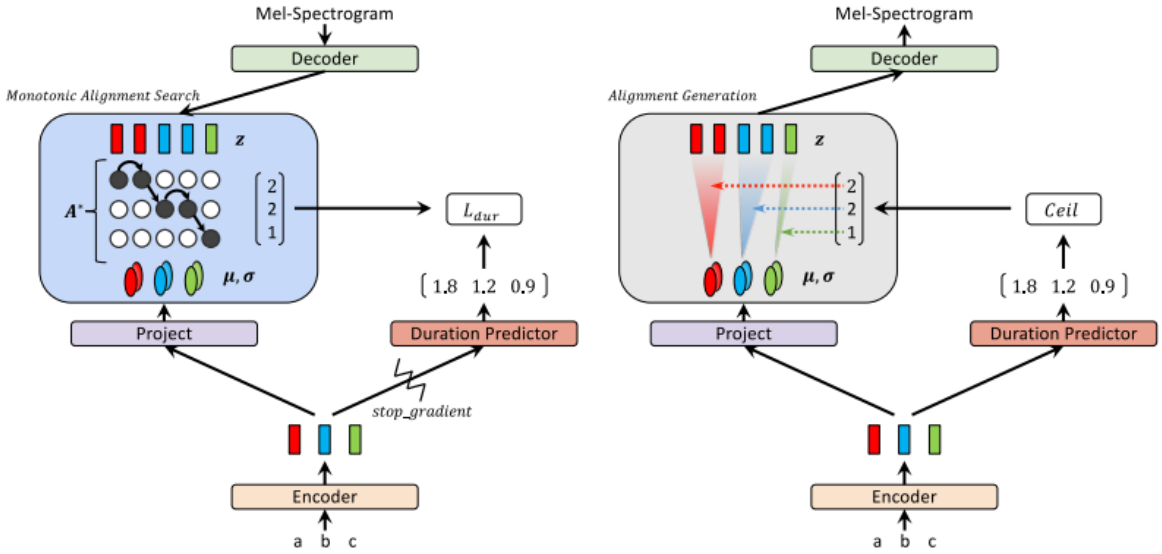


Attention mechanism [Shen18]



Duration prediction [Shen21]

Glow-TTS [Kim20]

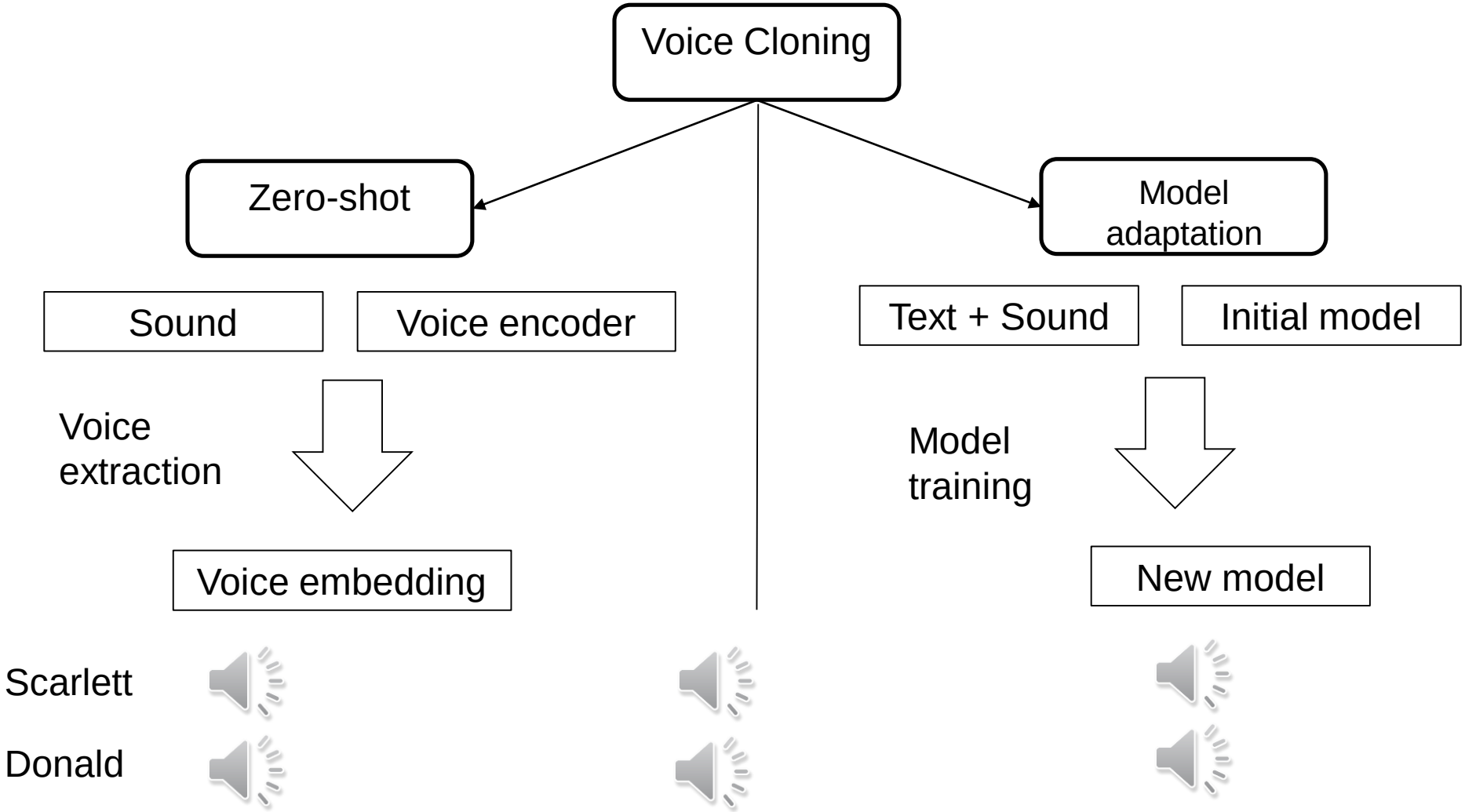


(a) An abstract diagram of the training procedure. (b) An abstract diagram of the inference procedure.

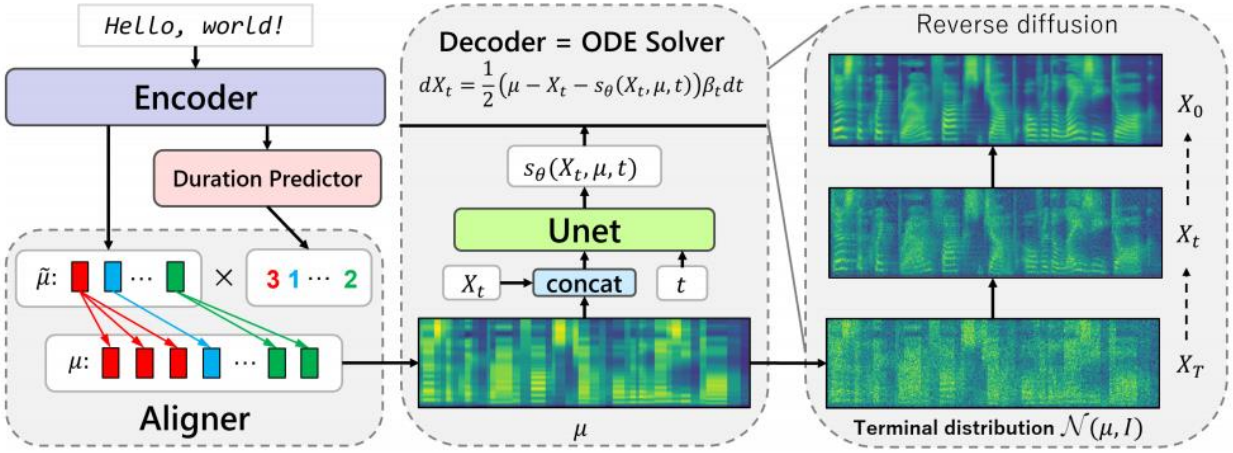
Figure 1: Training and inference procedures of Glow-TTS.

- Text encoder transforms text phoneme embeddings into a sequence of parameters of a Normal distribution N(mu, sigma)
- Then duration predictor upsamples the sequence to match the length of the target spectrogram
- Decoder is a Normalizing Flow (Glow) transforming samples from N(mu, sigma) to spectra
- Duration predictor is trained iteratively to match the optimal alignment path between decoder and encoder outputs

Voice cloning



GRAD-TTS [Popov21]



- Encoder predicts “noisy spectra”, i.e. means of Normal distribution N_0(mu, I)
- Decoder is trained to mimic a trajectory of a backward diffusion process restoring sample X_0 from X_T drawn from N_0(mu, I)
- Forward diffusion trajectory (X_0…X_T) can be calculated in closed form for any given X_0 so it can be used for “generating” target values for the backward diffusion
- Duration predictor is trained by means of Monotonic Alignment Search (MAS) from Glow-TTS paper

Accepted for ICML 2021

Speech recognition

SPEECH-TO-TEXT

Speech recognition as sequence labelling

- **Input sequence**: sequence of features (MFCC, MFSC or even raw wave samples)
- **Output sequence**: sequence of phonemes characters
- **Acoustic model**: feature sequence -> phoneme posteriors
- **Language model**: phoneme posteriors -> word sequence posteriors
- **Difference from the classical NLP sequence-labelling: lengths of the input and output sequences are different**
- Previous state-of-art: GMM-HMM models
- GMM acoustic models were replaced with DNN-based acoustic models: TDNN (actually CNN), LSTM, Bi-LSTM, Sequence-to-sequence, Transformer etc.
- Latest architectures can be trained to work without language models

ASR in 2010s

- DNN-HMM models
  - MFCC are replaced with MFSC
  - GMM-HMM system is trained for generating train set for DNN-based system
  - GMM is replaced with DNN as the acoustic model
  - DNN is usually 1d-convolutional (TDNN) or recurrent (GRU, LSTM or Bi-LSTM)
  - NN-based LM is used for hypotheses rescoring
- DNN-CTC models
  - Sequence-to-sequence neural models are trained from scratch with Connectionist Temporal Classification Loss (CTC)
- End2End models
  - No feature extractor is used. Input is raw waveform. No separate language model

Connectionist Temporal Classification (CTC) Intuition

- All operations in Forward-Backward procedure are differentiable
- So we can train a sequence model in a semi-supervised manner End-to-End without HMMs
- Criterion: $\underset{\theta}{\operatorname{argmax}} P(\boldsymbol{l}|X,\theta)$ where, X is a sequence of features, $\boldsymbol{l}$ is the corresponding text
- Let y_t^k is the k-th softmax output at time step *k* of the RNN
- Let π be a sequence of RNN outputs
- Define $\boldsymbol{\mathcal{B}}(\pi_{1..t})$ as a result of a mapping from the sequence of RNN outputs onto the subsequence of the orthographic form $\boldsymbol{l}$
- We assume that all y_t^k are conditionally independent

Connectionist Temporal Classification (CTC)

Forward-backward recursion

Actual CTC aligns feature sequence and ‘extended’ sequence of characters with special blank symbol allowing to map the input sequence directly onto orthographic form: e.g. /aple/ -> “apple”. So we also consider extended orthographic form $\boldsymbol{l}'$

Forward procedure

$$\alpha_t(s) \stackrel{\text{def}}{=} \sum_{\substack{\pi \in N^T: \\ \mathcal{B}(\pi_{1:t})=\mathbf{l}_{1:s}}} \prod_{t'=1}^{t} y^{t'}_{\pi_{t'}}$$

Initialization

$$\alpha_1(1) = y^1_b$$
$$\alpha_1(2) = y^1_{\mathbf{l}_1}$$
$$\alpha_1(s) = 0, \ \forall s > 2$$

Recursion

$$\alpha_t(s) = \begin{cases} \bar{\alpha}_t(s) y^t_{\mathbf{l}'_s} & \text{if } \mathbf{l}'_s = b \text{ or } \mathbf{l}'_{s-2} = \mathbf{l}'_s \\ \left(\bar{\alpha}_t(s) + \alpha_{t-1}(s-2)\right) y^t_{\mathbf{l}'_s} & \text{otherwise} \end{cases}$$

where

$$\bar{\alpha}_t(s) \stackrel{\text{def}}{=} \alpha_{t-1}(s) + \alpha_{t-1}(s-1).$$

Stop

$$p(\mathbf{l}|\mathbf{x}) = \alpha_T(|\mathbf{l}'|) + \alpha_T(|\mathbf{l}'|-1)$$

Backward procedure

$$\beta_t(s) \stackrel{\text{def}}{=} \sum_{\substack{\pi \in N^T: \\ \mathcal{B}(\pi_{t:T})=\mathbf{l}_{s:|\mathbf{l}|}}} \prod_{t'=t}^{T} y^{t'}_{\pi_{t'}}$$

Initialization

$$\beta_T(|\mathbf{l}'|) = y^T_b$$
$$\beta_T(|\mathbf{l}'|-1) = y^T_{\mathbf{l}_{|\mathbf{l}|}}$$
$$\beta_T(s) = 0, \ \forall s < |\mathbf{l}'|-1$$

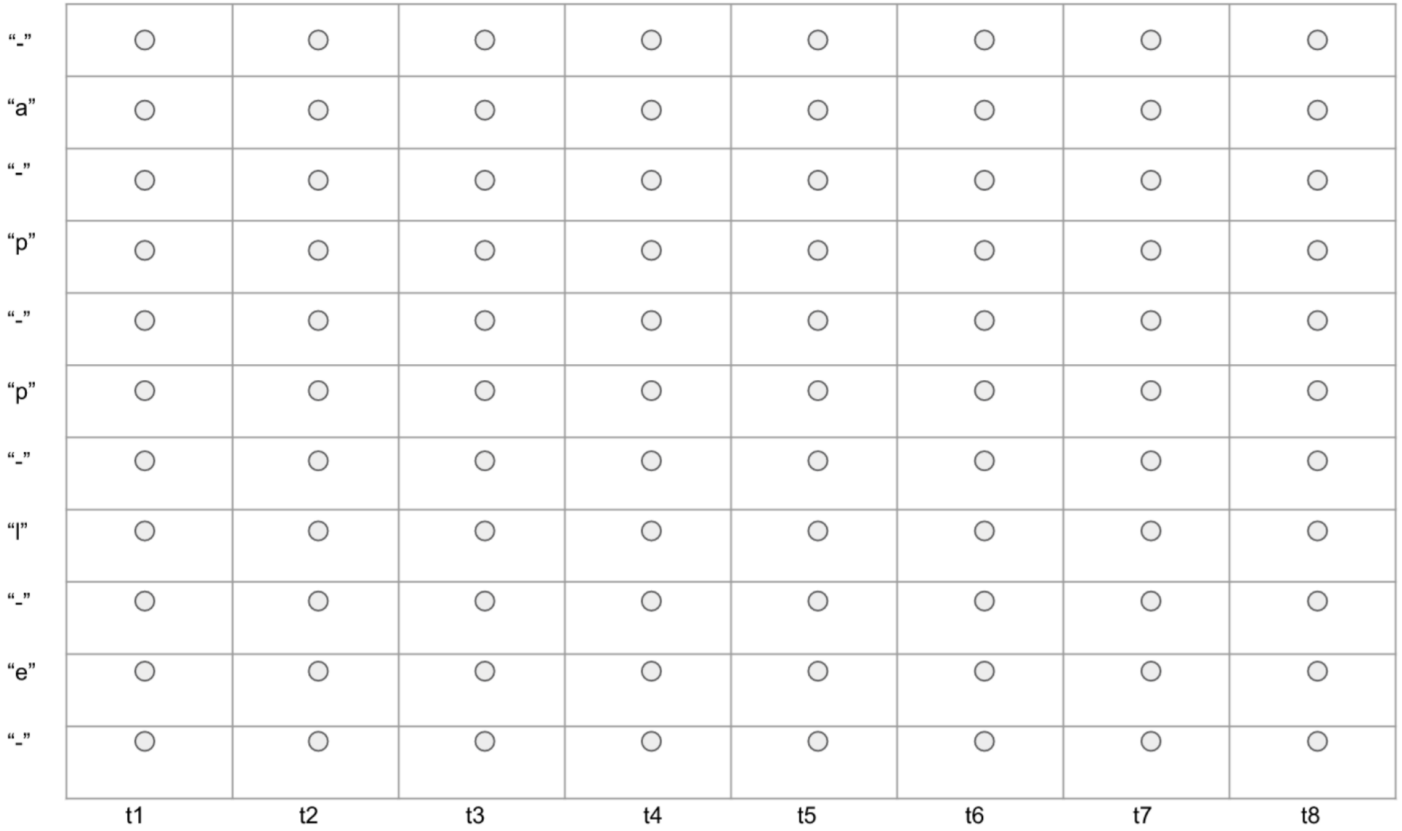
Recursion

$$\beta_t(s) = \begin{cases} \bar{\beta}_t(s) y^t_{\mathbf{l}'_s} & \text{if } \mathbf{l}'_s = b \text{ or } \mathbf{l}'_{s+2} = \mathbf{l}'_s \\ \left(\bar{\beta}_t(s) + \beta_{t+1}(s+2)\right) y^t_{\mathbf{l}'_s} & \text{otherwise} \end{cases}$$

where

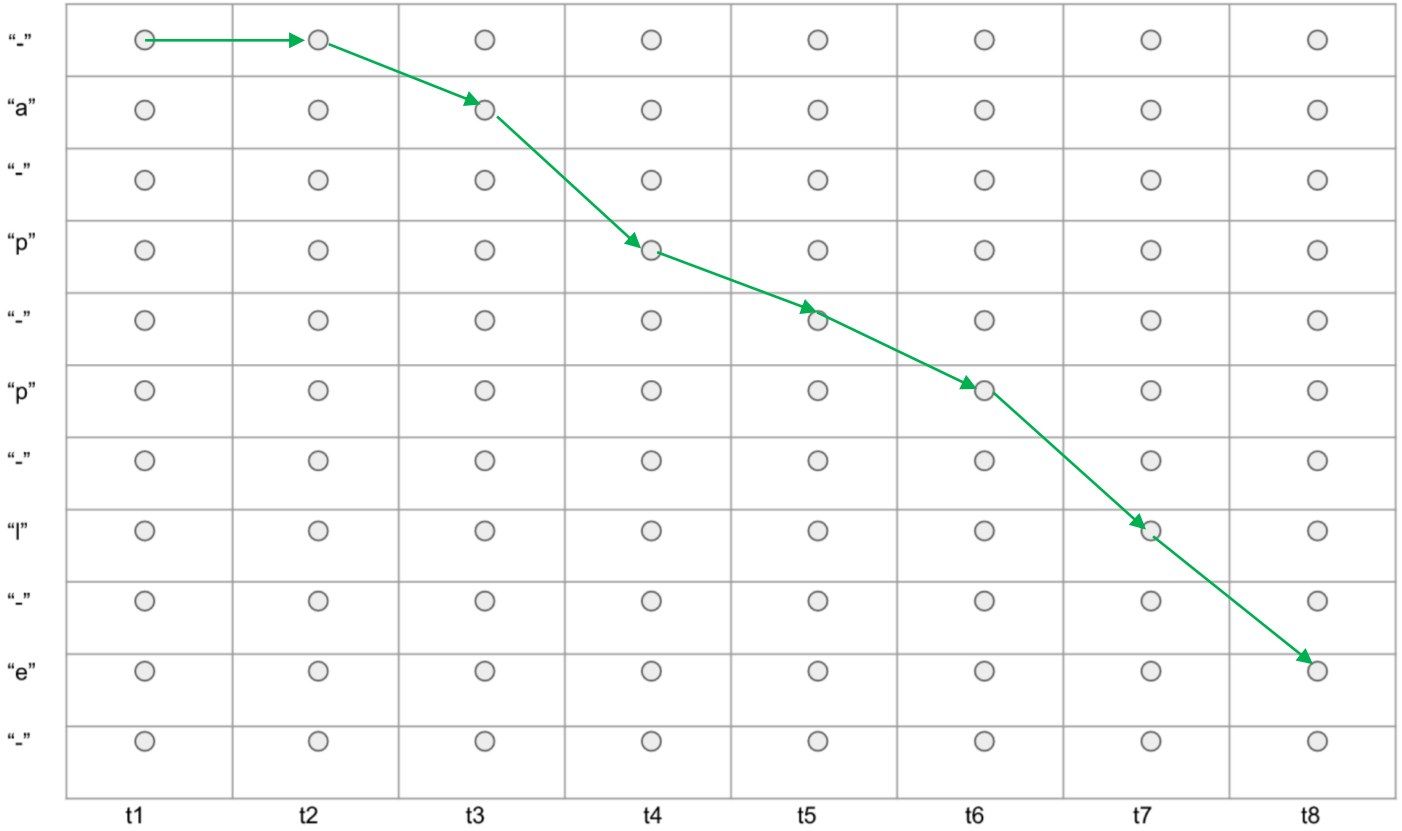
$$\bar{\beta}_t(s) \stackrel{\text{def}}{=} \beta_{t+1}(s) + \beta_{t+1}(s+1).$$

CTC example



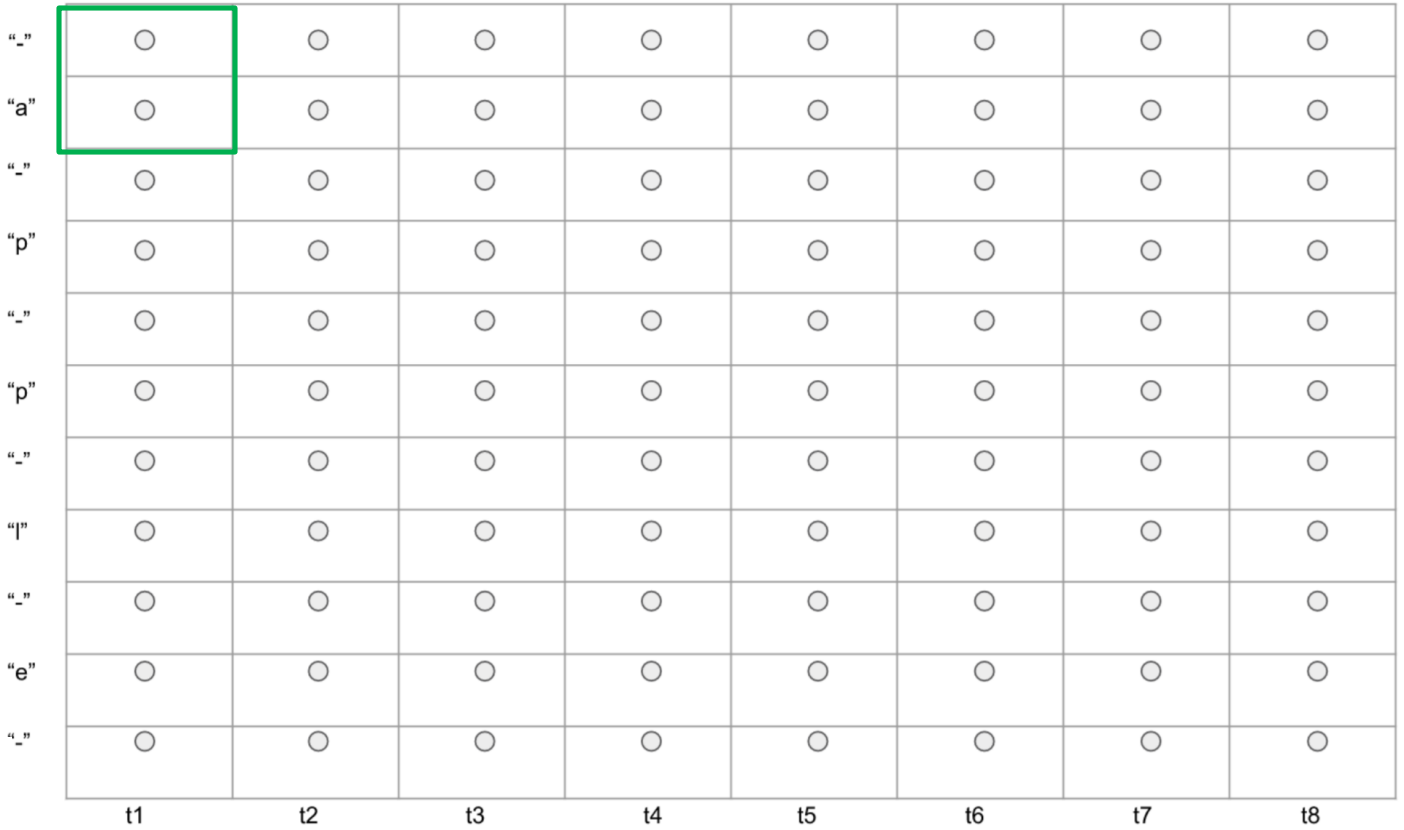
CTC example

$\mathcal{B}$(‘--ap-ple’)=‘apple’



CTC example

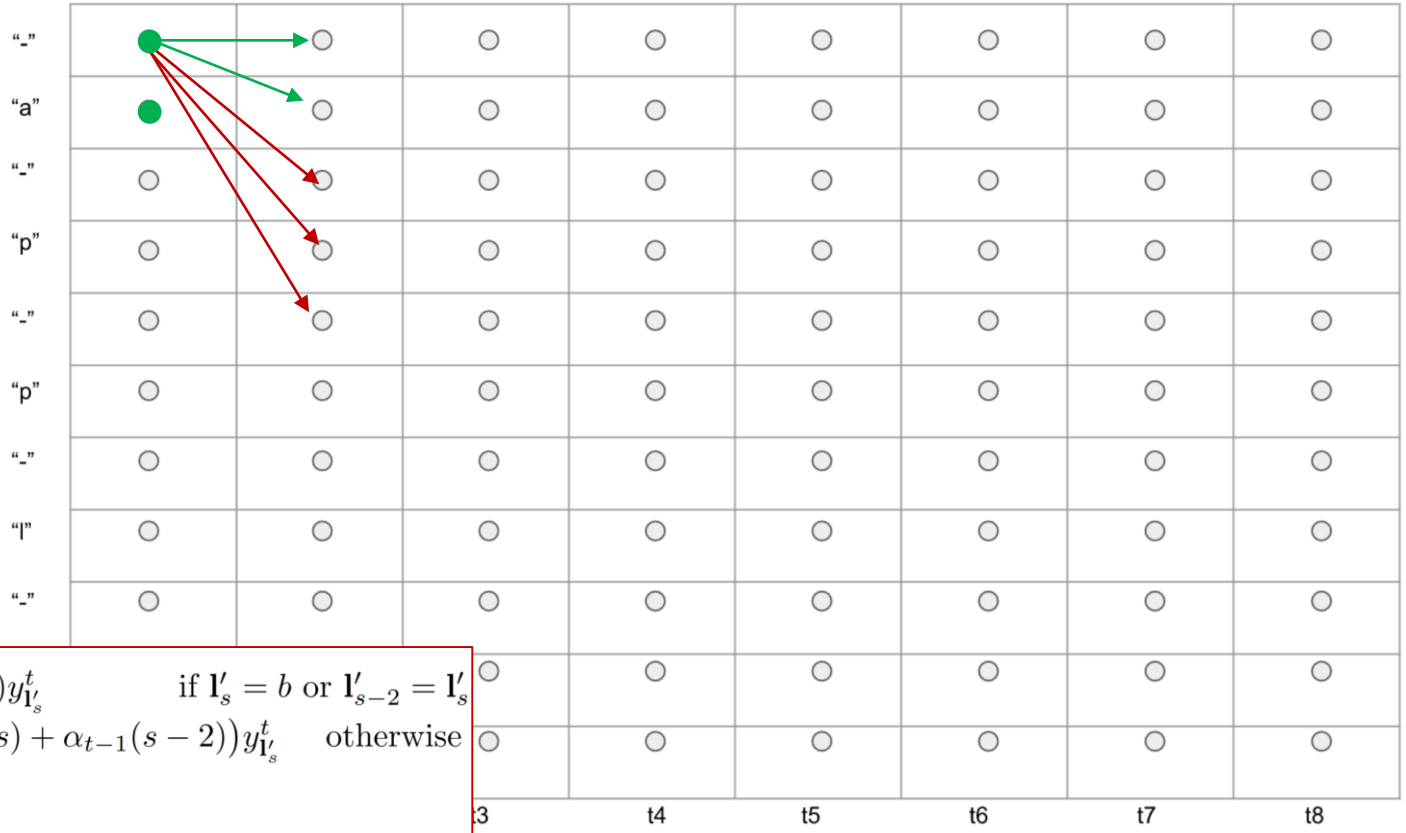
Initilization



$$\alpha_1(1) = y_b^1$$
$$\alpha_1(2) = y_{\mathrm{l}_1}^1$$
$$\alpha_1(s) = 0, \ \forall s > 2$$

CTC example

Recursion



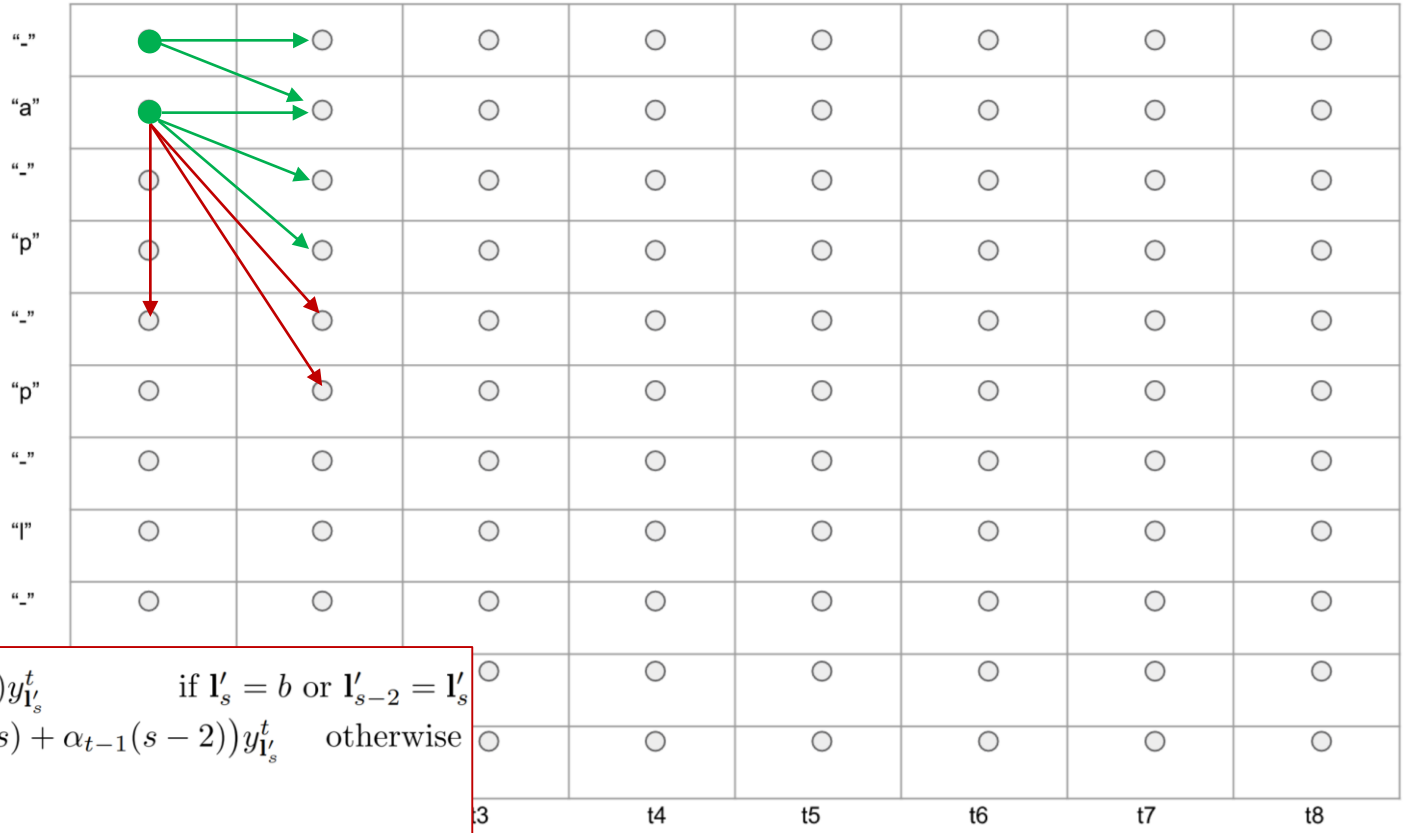
$$\alpha_t(s) = \begin{cases} \bar{\alpha}_t(s) y^t_{\mathbf{l}'_s} & \text{if } \mathbf{l}'_s = b \text{ or } \mathbf{l}'_{s-2} = \mathbf{l}'_s \\ \big(\bar{\alpha}_t(s) + \alpha_{t-1}(s-2)\big) y^t_{\mathbf{l}'_s} & \text{otherwise} \end{cases}$$

where

$$\bar{\alpha}_t(s) \overset{\text{def}}{=} \alpha_{t-1}(s) + \alpha_{t-1}(s-1).$$

CTC example

Recursion



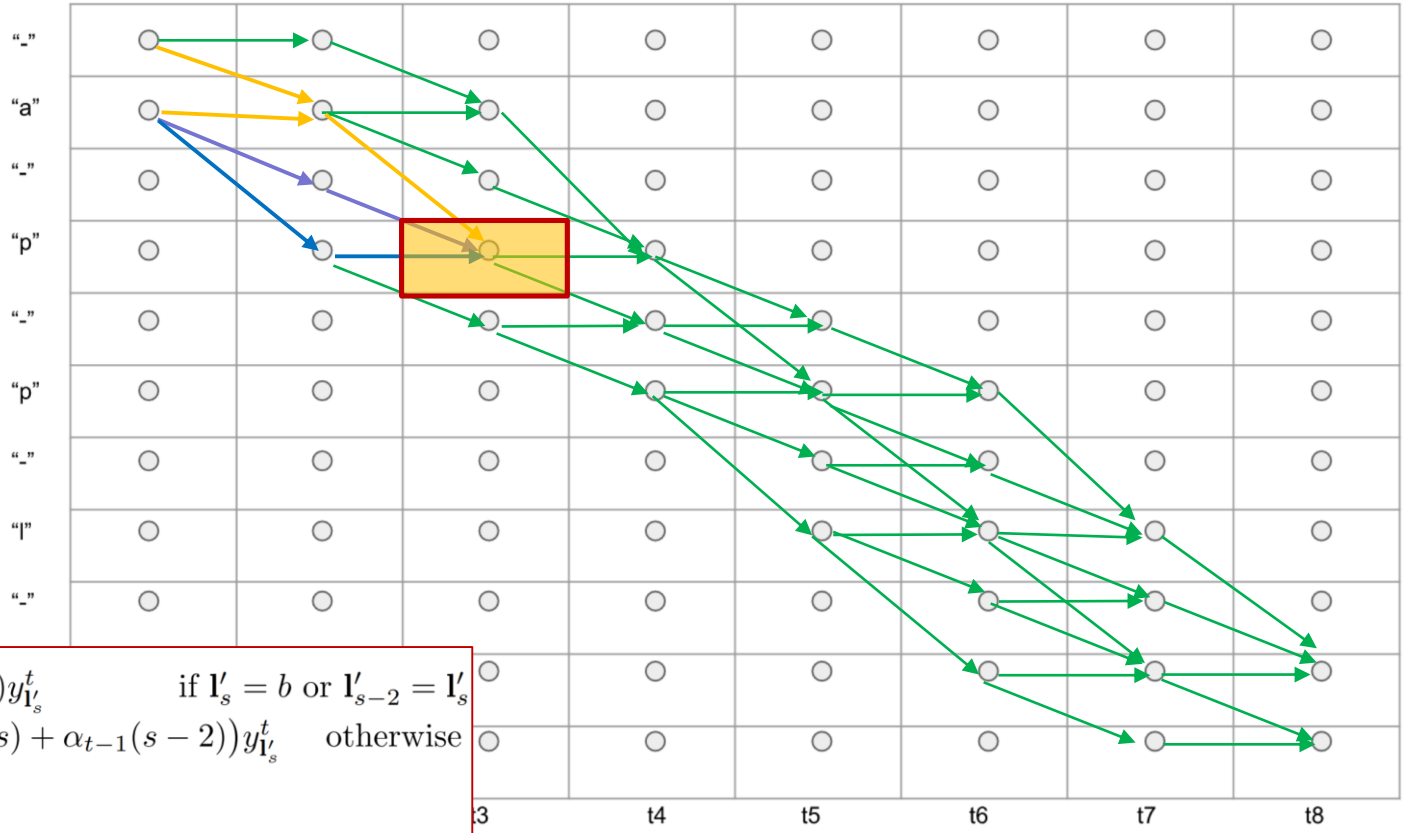
$$\alpha_t(s) = \begin{cases} \bar{\alpha}_t(s) y^t_{\mathbf{l}'_s} & \text{if } \mathbf{l}'_s = b \text{ or } \mathbf{l}'_{s-2} = \mathbf{l}'_s \\ \big(\bar{\alpha}_t(s) + \alpha_{t-1}(s-2)\big) y^t_{\mathbf{l}'_s} & \text{otherwise} \end{cases}$$

where

$$\bar{\alpha}_t(s) \overset{\text{def}}{=} \alpha_{t-1}(s) + \alpha_{t-1}(s-1).$$

CTC example

$$\alpha_3(4) = p(\text{"\_ap"}) + p(\text{"aap"}) + p(\text{"a\_p"}) + p(\text{"app"})$$



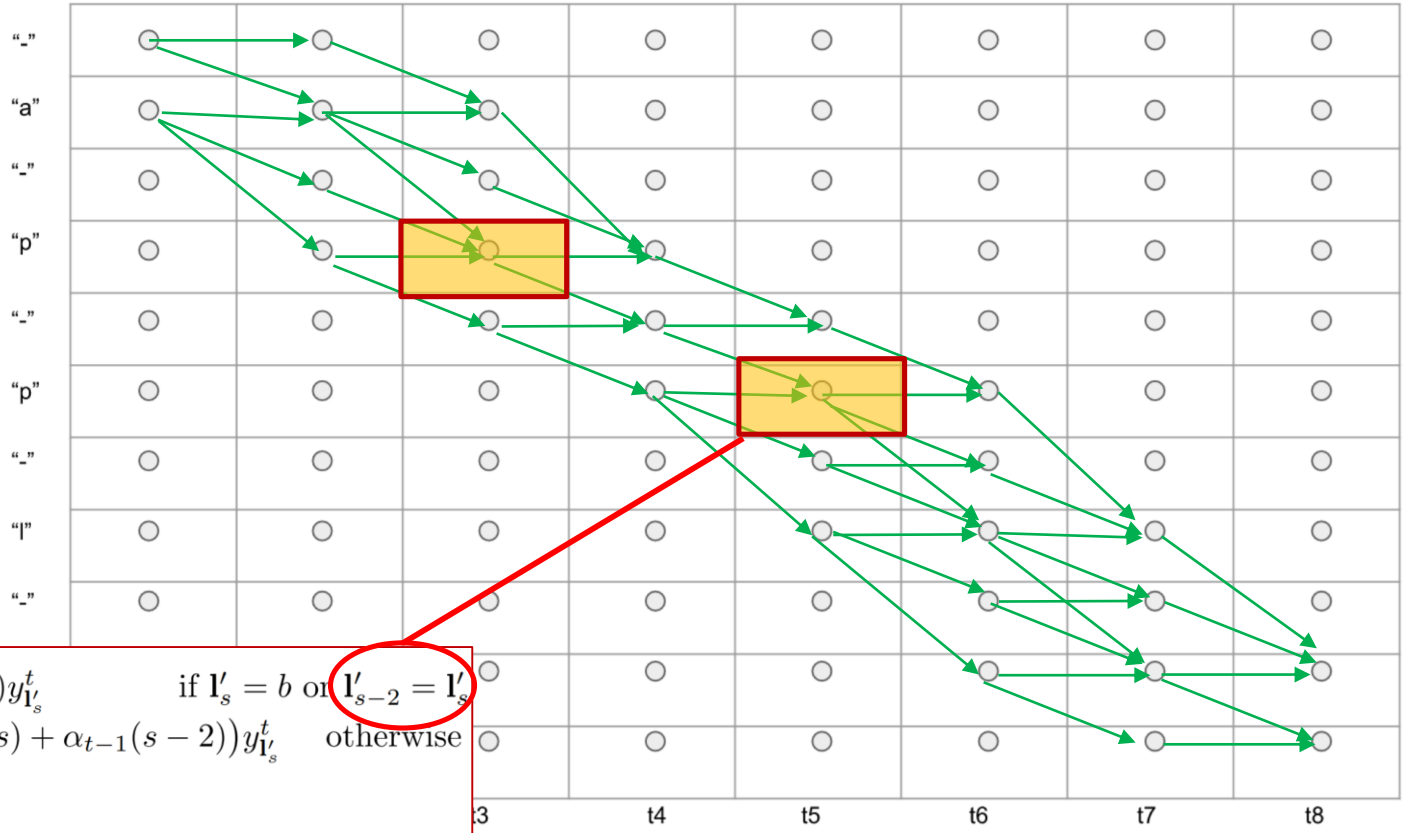
$$\alpha_t(s) = \begin{cases} \bar{\alpha}_t(s) y^t_{\mathbf{l}'_s} & \text{if } \mathbf{l}'_s = b \text{ or } \mathbf{l}'_{s-2} = \mathbf{l}'_s \\ \big(\bar{\alpha}_t(s) + \alpha_{t-1}(s-2)\big) y^t_{\mathbf{l}'_s} & \text{otherwise} \end{cases}$$

where

$$\bar{\alpha}_t(s) \stackrel{\text{def}}{=} \alpha_{t-1}(s) + \alpha_{t-1}(s-1).$$

CTC example

Handling double characters



$$\alpha_t(s) = \begin{cases} \bar{\alpha}_t(s) y^t_{\mathrm{l}'_s} & \text{if } \mathrm{l}'_s = b \text{ or } \mathrm{l}'_{s-2} = \mathrm{l}'_s \\ \big(\bar{\alpha}_t(s) + \alpha_{t-1}(s-2)\big) y^t_{\mathrm{l}'_s} & \text{otherwise} \end{cases}$$

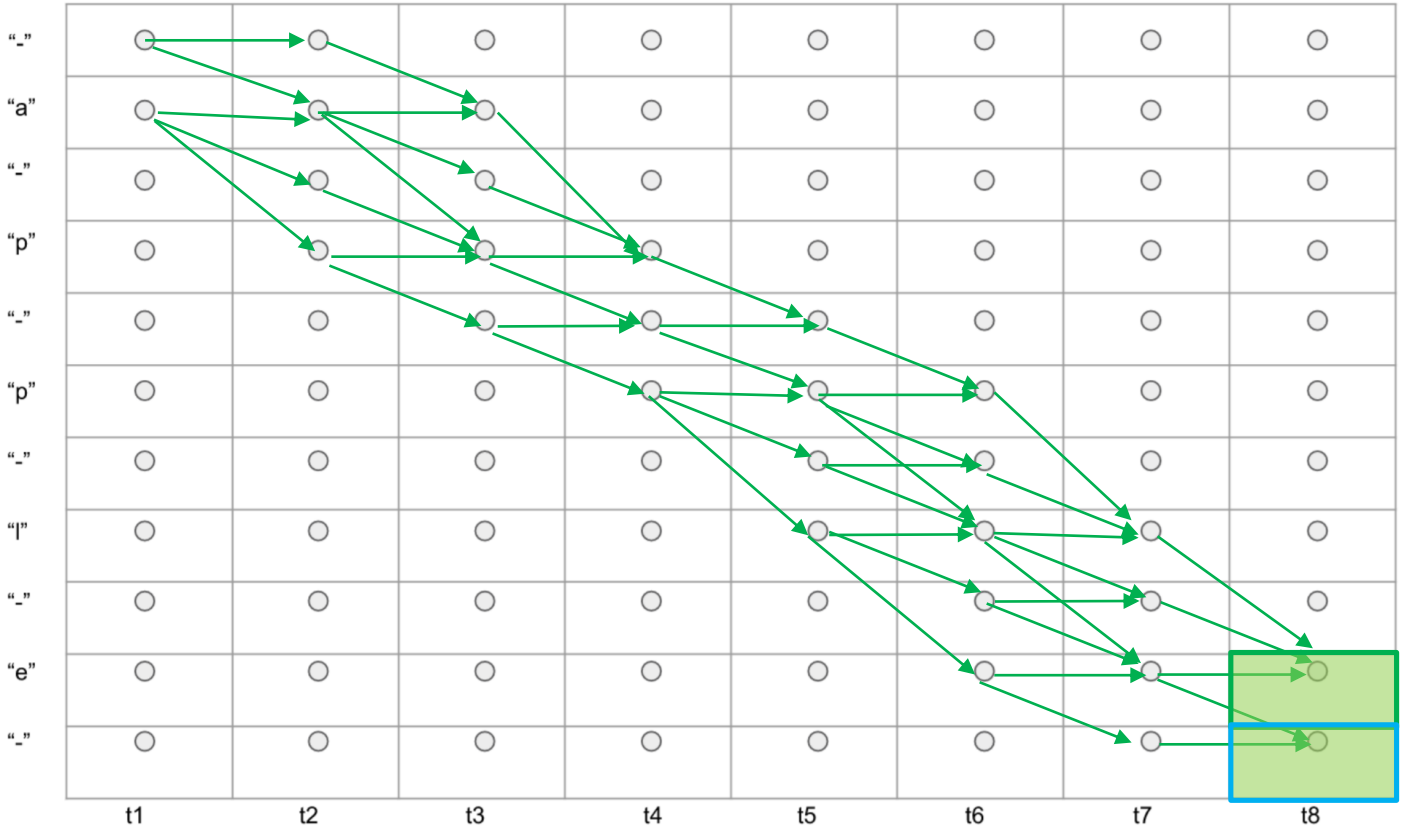
where

$$\bar{\alpha}_t(s) \stackrel{\text{def}}{=} \alpha_{t-1}(s) + \alpha_{t-1}(s-1).$$

CTC example

Stop

$$L = -\ln\big(\alpha_8(10) + \alpha_8(11)\big) = -\ln(\boldsymbol{l} =' apple' | X)$$



CTC example

$\alpha_3(2)$ = p("--a") + p("-aa") + p("aaa") =

$$= y_-^1 \cdot y_-^2 \cdot y_a^3 + y_-^1 \cdot y_a^2 \cdot y_a^3 + y_a^1 \cdot y_a^2 \cdot y_a^3$$

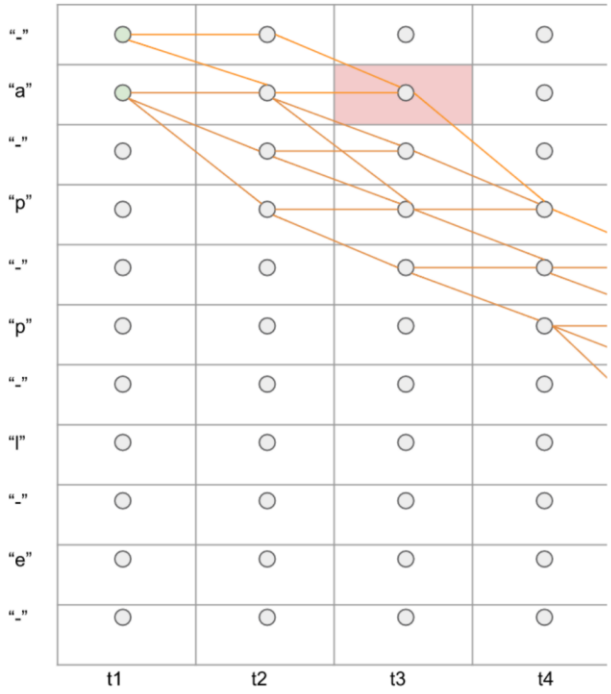
$\beta_3(2)$= p("ap-ple")= $y_a^3 \cdot y_p^4 \cdot y_-^5 \cdot y_p^6 \cdot y_l^7 \cdot y_e^8$

=> $\alpha_3(2) \cdot \beta_3(2) = y_-^1 \cdot y_-^2 \cdot y_a^3 \cdot y_a^3 \cdot y_p^4 \cdot y_-^5 \cdot y_p^6 \cdot y_l^7 \cdot y_e^8$ +

\+ $y_-^1 \cdot y_a^2 \cdot y_a^3 \cdot y_a^3 \cdot y_p^4 \cdot y_-^5 \cdot y_p^6 \cdot y_l^7 \cdot y_e^8$

\+ $y_a^1 \cdot y_a^2 \cdot y_a^3 \cdot y_a^3 \cdot y_p^4 \cdot y_-^5 \cdot y_p^6 \cdot y_l^7 \cdot y_e^8$

CTC example



$$\alpha_3(2) \cdot \beta_3(2) = y_-^1 \cdot y_-^2 \cdot y_a^3 \cdot \boxed{y_a^3} \cdot y_p^4 \cdot y_-^5 \cdot y_p^6 \cdot y_l^7 \cdot y_e^8 \; +$$

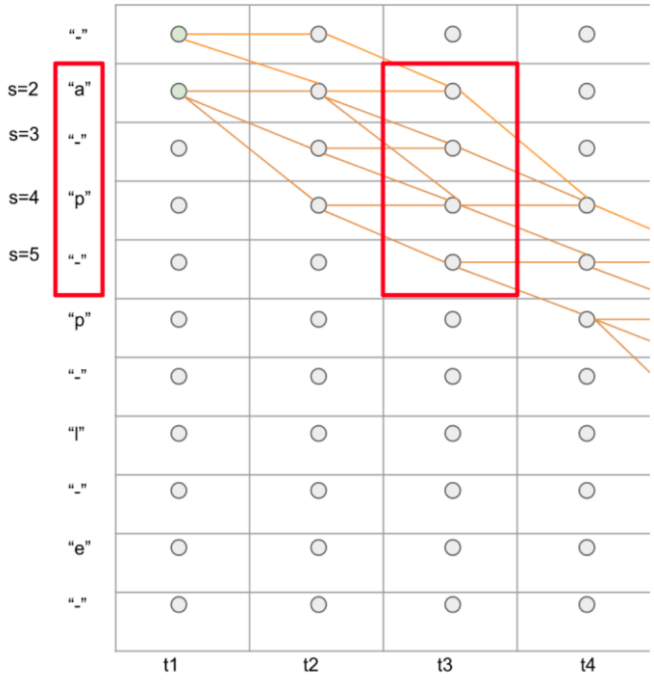
$$+ \quad y_-^1 \cdot y_a^2 \cdot y_a^3 \cdot \boxed{y_a^3} \cdot y_p^4 \cdot y_-^5 \cdot y_p^6 \cdot y_l^7 \cdot y_e^8$$

$$+ \quad y_a^1 \cdot y_a^2 \cdot y_a^3 \cdot \boxed{y_a^3} \cdot y_p^4 \cdot y_-^5 \cdot y_p^6 \cdot y_l^7 \cdot y_e^8 \; =$$

$$= (\; p(\text{“--aap-ple”}) + p(\text{“-aap-ple”}) + p(\text{“aaap-ple”}) \;) * y_a^3$$

$$\frac{\alpha_3(2) \cdot \beta_3(2)}{y_a^3} = (\; p(\text{“--aap-ple”}) + p(\text{“-aap-ple”}) + p(\text{“aaap-ple”}) \;)$$

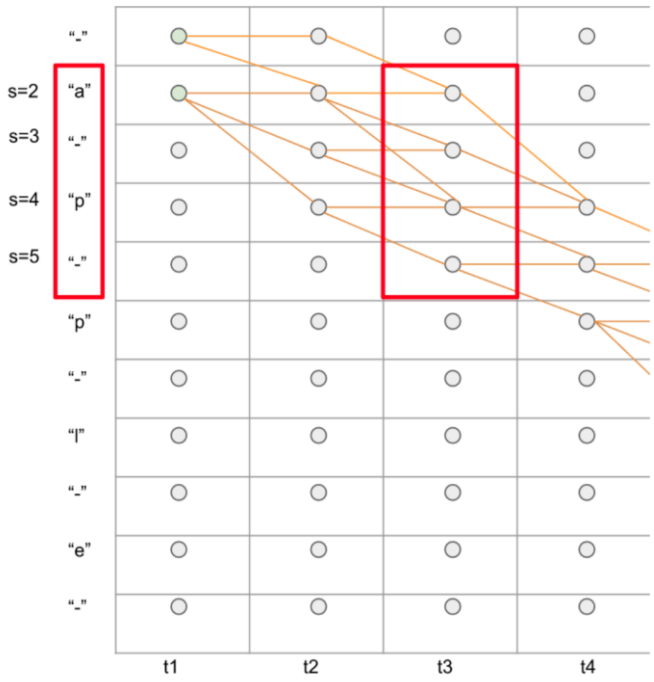
CTC example



$$p_3(\text{"apple"}) = \sum_{s=2}^{5} \frac{\alpha_3(s)\beta_3(s)}{y_{l_s}^3}$$

$$p_t(\text{"apple"}) = \sum_{s=1}^{|l|} \frac{\alpha_t(s)\beta_t(s)}{y_{l_s}^t}$$

CTC example: gradient



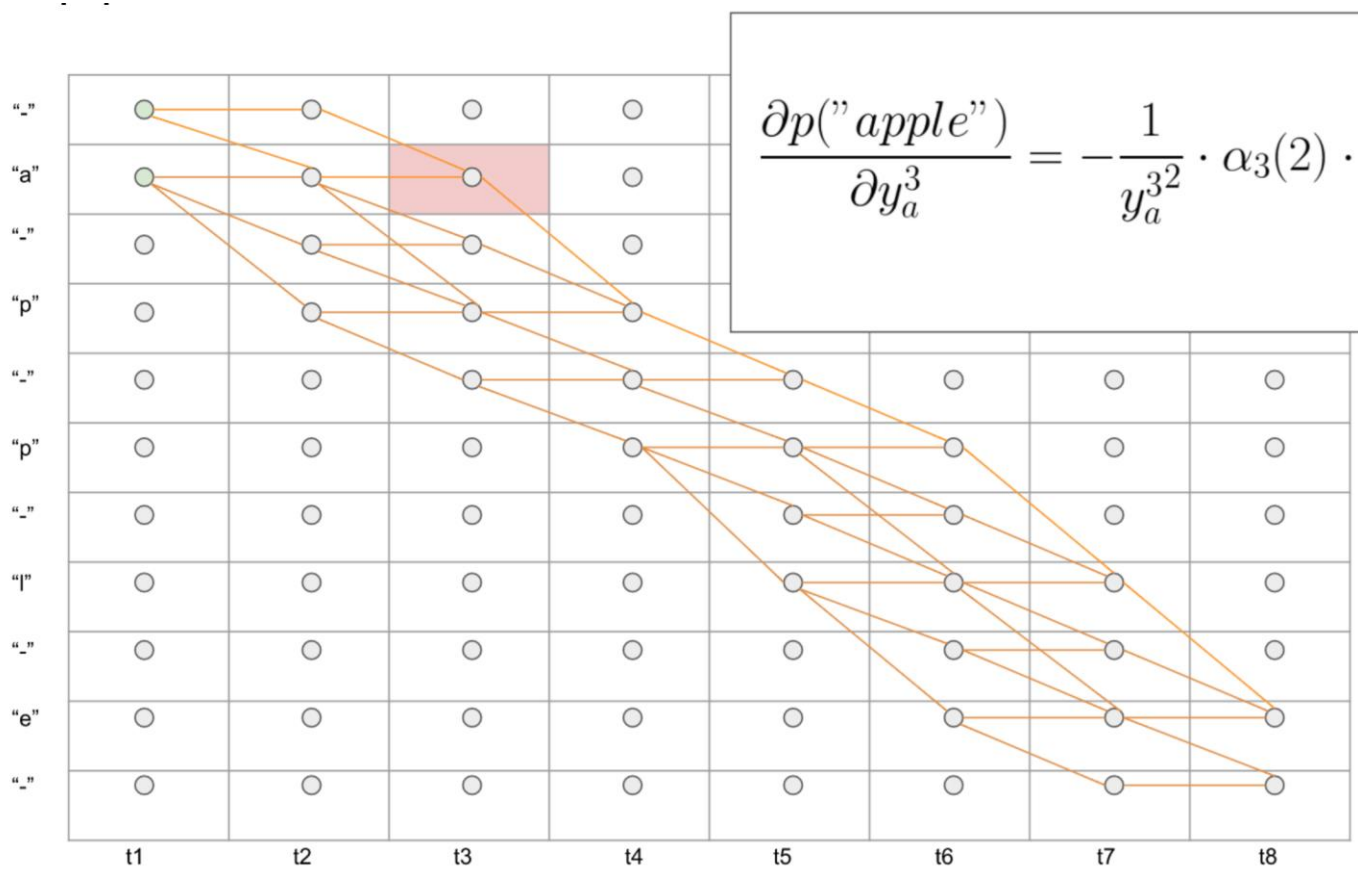
$$p(\text{"apple"}) = \sum_{s=1}^{|seq|} \frac{\alpha_t(s) \cdot \beta_t(s)}{y^t_{seq(s)}}$$

$$\frac{\partial p(\text{"apple"})}{\partial y^t_k} = -\frac{1}{{y^t_k}^2} \cdot \sum_{s:seq(s)=k} \alpha_t(s) \cdot \beta_t(s)$$

$$\frac{\partial(-ln(p(\text{"apple"})))}{\partial y^t_k} = -\frac{1}{p(\text{"apple"})} \frac{\partial p(\text{"apple"})}{\partial y^t_k}$$

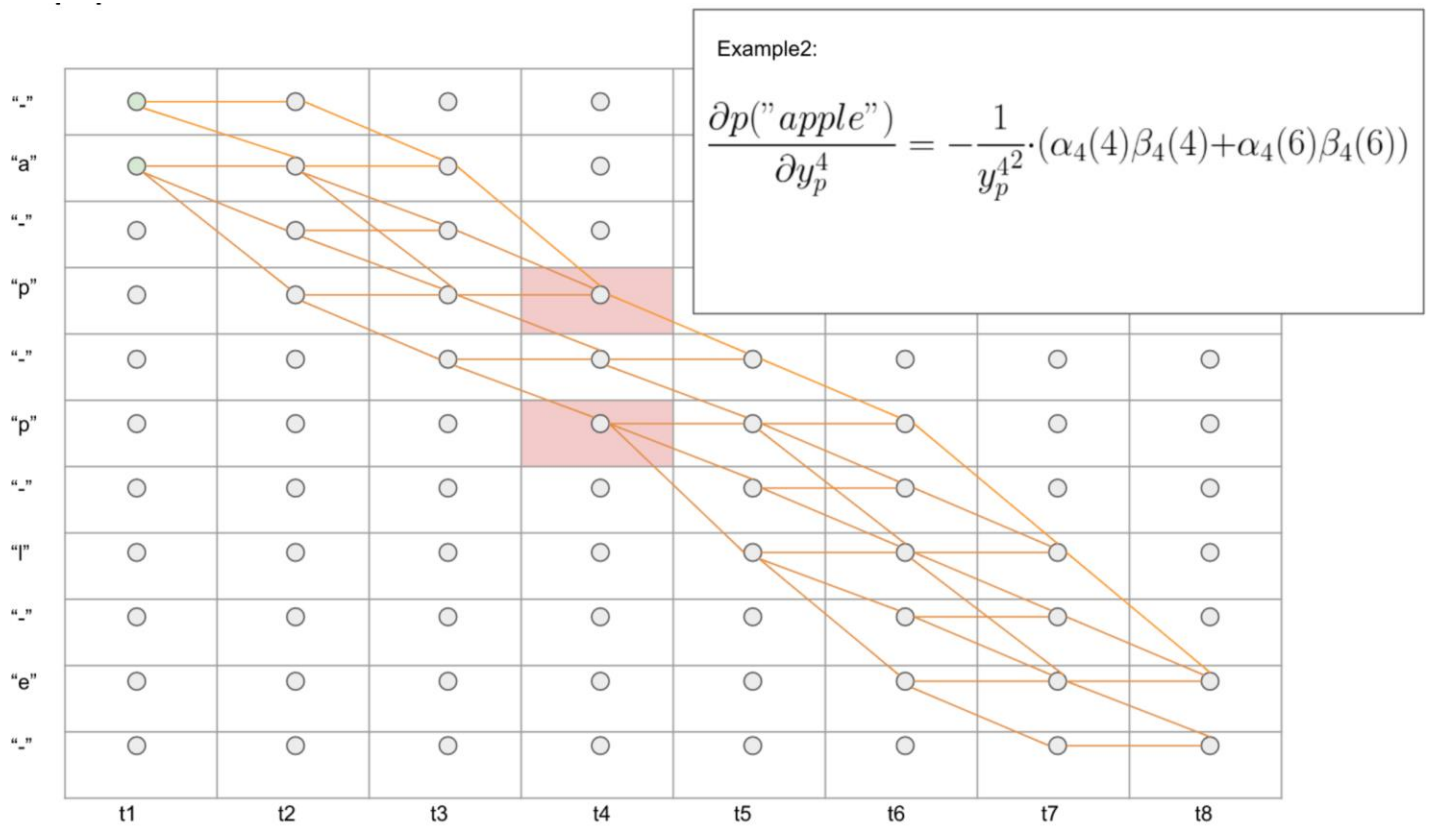
$$\frac{\partial p(\text{"apple"})}{\partial y^t_k} = -\frac{1}{{y^t_k}^2} \cdot \sum_{s:seq(s)=k} \alpha_t(s) \cdot \beta_t(s)$$

CTC example: gradient



$$\frac{\partial p(\text{"apple"})}{\partial y_a^3} = -\frac{1}{y_a^{3^2}} \cdot \alpha_3(2) \cdot \beta_3(2)$$

CTC example: gradient

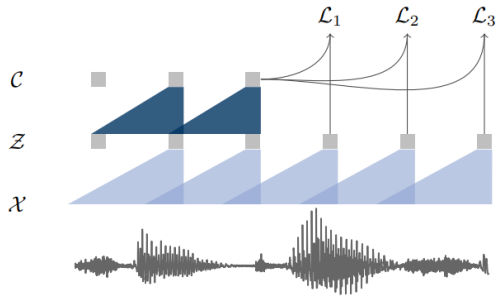


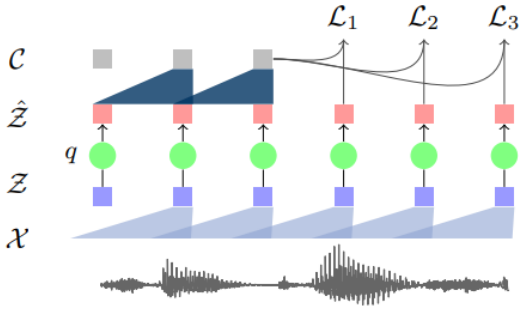
Wav2Vec: evolution

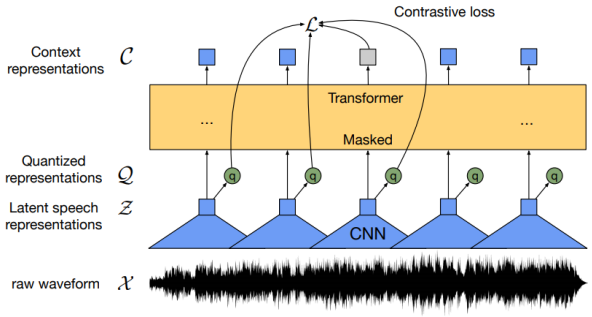
Common part:
- X is raw audio signal
- encoder network f: X -> Z
- context network g: Z -> C
- Objective: contrastive loss
- ~1000h base model pretraining

vq-wav2vec:
- quantizer q: Z -> Z^
- discretized pipeline with BERT

Wav2Vec 2.0:
- transformer g: Z->C
- layer normalization
- masking
- + diversity loss
- Large model : 60k h pretraining





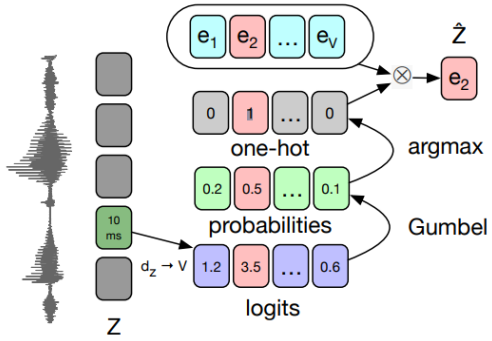


vq-wav2vec

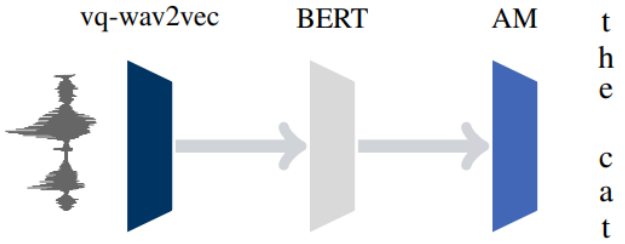
- Codebook contains V representations of size d

  $\hat{z} = e_i, \qquad e \in \mathbb{R}^{V \times d}$

- Gumbel-Softmax - differentiable way to select discrete codebook entity



- Dividing feature vector into several groups G



Pretraining BERT with discrete inputs of audio data q by masking spans

Wav2Vec Results

WER
LibriSpeech

	dev		test	
	clean	other	clean	other
vq-wav2vec Gumbel + Transformer Big	5.6	15.5	6.2	18.2
wav2vec2 LV-60 + Transformer	1.6	3.0	1.8	3.3

PER
CommonVoice

Model	D	#pt	#ft	es	fr	it	ky	nl	ru	sv	tr	tt	zh	Avg
Number of pretraining hours per language				168h	353h	90h	17h	29h	55h	3h	11h	17h	50h	793h
Number of fine-tuning hours per language				1h	1h	1h	1h	1h	1h	1h	1h	1h	1h	10h
XLSR-53	D_{53}	53	1	**2.9**	**5.0**	**5.7**	**6.1**	**5.8**	**8.1**	**12.2**	**7.1**	**5.1**	**18.3**	**7.6**

WER CommonVoice

Model	#pt	#ft	en	de	nl	fr	es	it	pt	pl
Number of training hours			44.7k	2k	1.6k	1.1k	918	247	161	104
XLSR-53	53	1-100h	13.2	7.4	10.9	9.8	7.9	12.0	15.7	18.9

LibriSpeech test clean:

1	Conformer + Wav2vec 2.0 + SpecAugment-based Noisy Student Training with Libri-Light	1.4
2	Conv + Transformer + wav2vec2.0 + pseudo labeling	1.5
3	ContextNet + SpecAugment-based Noisy Student Training with Libri-Light	1.7
4	Multistream CNN with Self-Attentive SRU	1.75
5	wav2vec 2.0 with Libri-Light	1.8

Wav2Vec Results

Table 1: WER on the Librispeech dev and test sets for the Libri-light low-resource labeled data setups of 10 min, 1 hour and 10 hours. As unlabeled data we use the audio of Librispeech (LS-960) or the larger LibriVox (LV-60k). ST (s2s scratch) trains a sequence to sequence model with a word-piece vocabulary on the pseudo-labeled data from random initialization while as ST (ctc ft) fine-tunes wav2vec 2.0 with the pseudo-labels using CTC and a letter-based vocabulary. All results are with language models at inference time.

Model	Unlbld data	dev		test	
		clean	other	clean	other
10 min labeled					
Discr. BERT [27]	LS-960	15.7	24.1	16.3	25.2
wav2vec 2.0 [24]	LS-960	6.6	10.6	6.8	10.8
+ ST (s2s scratch)	LS-960	4.1	7.0	5.0	8.1
+ ST (ctc ft)	LS-960	3.6	6.6	4.0	7.2
wav2vec 2.0 [24]	LV-60k	5.0	8.4	5.2	8.6
+ ST (s2s scratch)	LV-60k	2.6	4.7	3.1	5.4
+ ST (ctc ft)	LV-60k	2.8	4.6	3.0	5.2
1h labeled					
Discr. BERT [27]	LS-960	8.5	16.4	9.0	17.6
wav2vec 2.0 [24]	LS-960	3.8	7.1	3.9	7.6
+ ST (s2s scratch)	LS-960	2.9	5.6	3.4	6.6
+ ST (ctc ft)	LS-960	2.8	5.5	3.1	6.3
10h labeled					
Discr. BERT [27]	LS-960	5.3	13.2	5.9	14.1
IPL [14]	LS-960	23.5	25.5	24.4	26.0
wav2vec 2.0 [24]	LS-960	2.9	5.7	3.2	6.1
+ ST (s2s scratch)	LS-960	2.5	5.1	3.5	5.9
+ ST (ctc ft)	LS-960	2.6	5.2	2.9	5.7

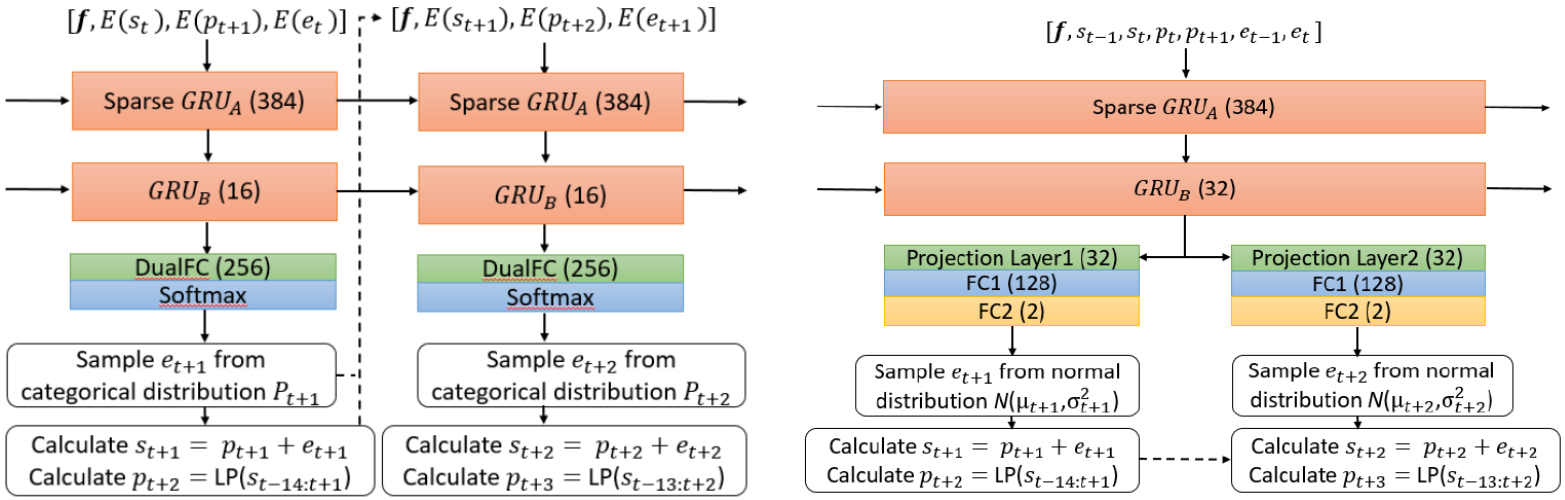
References

- [Oord17] A. van den Oord et al. WAVENET: A GENERATIVE MODEL FOR RAW AUDIO
- [Kal18] N. Kalchbrenner et al. EFFICIENT NEURAL AUDIO SYNTHESIS
- [Wang17] Y. Wang et al. TACOTRON: TOWARDS END-TO-END SPEECH SYNTHESIS
- [Shen18] J. Shen et al. NATURAL TTS SYNTHESIS BY CONDITIONING WAVENET ON MEL SPECTROGRAM PREDICTIONS
- [Shen21] J. Shen et al. Non-Attentive Tacotron: Robust and Controllable Neural TTS Synthesis Including Unsupervised Duration Modeling
- [Oord&Li17] A. van den Oord et al. PARALLEL WAVENET: FAST HIGH-FIDELITY SPEECH SYNTHESIS
- [Ping18] W.Ping et al. CLARINET: PARALLEL WAVE GENERATION IN END-TO-END TEXT-TO-SPEECH
- [Pren19] R.Prenger et al. WAVEGLOW: A FLOW-BASED GENERATIVE NETWORK FOR SPEECH SYNTHESIS
- [Val19] J.-M. Valin, J. Skoglund LPCNET: IMPROVING NEURAL SYNTHESIS THROUGH LINEAR PREDICTION
- [Kim20] Glow-TTS: A Generative Flow for Text-to-Speech via Monotonic Alignment Search
- [Kong20] HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis
- [Popov21] Grad-TTS: A Diffusion Probabilistic Model for Text-to-Speech

Thank You

www.huawei.com

Multisample Gaussian LPCNet



- Idea #1: predict n>1 samples per step
  - ⇨ Speedup up to 50%
- Idea #2: predict parameters of Normal distribution
  - ⇨ 16-bit sound
  - ⇨ ~50% reduction in number of parameters

Connectionist Temporal Classification (CTC)

Forward-backward recursion (simple case)

- $P\{D(X) = Y\} =$

$$\sum_k P\left\{D(X_1^t) = S_1^k, D(x_t) = s_k\right\} \cdot P\left\{D(X_{t+1}^T) = S_k^L \cup D(X_{t+1}^T) = S_{k+1}^L | D(x_t) = s_k\right\} =$$

$$= \sum_k P\left(D(X_1^t) = S_1^k, D(x_t) = s_k\right) \frac{\left[P\left(D(X_{t+1}^T) = S_k^L, D(x_t) = s_k\right) + P\left(D(X_{t+1}^T) = S_{k+1}^L, D(x_t) = s_k\right)\right]}{p(D(x_t) = s_k)}$$

$$= \sum_k \alpha_t(s_k)\beta_t(s_k)/y_t^k$$

Forward procedure (no blanks)

Init: $\alpha_1(s_k) = y_1^k$
Recursion: $\alpha_t(s_k) = [\alpha_{t-1}(s_{k-1}) + \alpha_{t-1}(s_k)]y_t^k$
Stop: $\alpha_T(s_L) = P(D(X) = Y)$

Backward procedure (no blanks)

Init: $\beta_T(s_k) = y_T^k$
Recursion: $\beta_t(s_k) = [\beta_{t+1}(s_{k+1}) + \beta_{t+1}(s_k)]y_t^k$
Stop: $\beta_1(s_1) = P(D(X) = Y)$