Speech Synthesis

Milan Straka

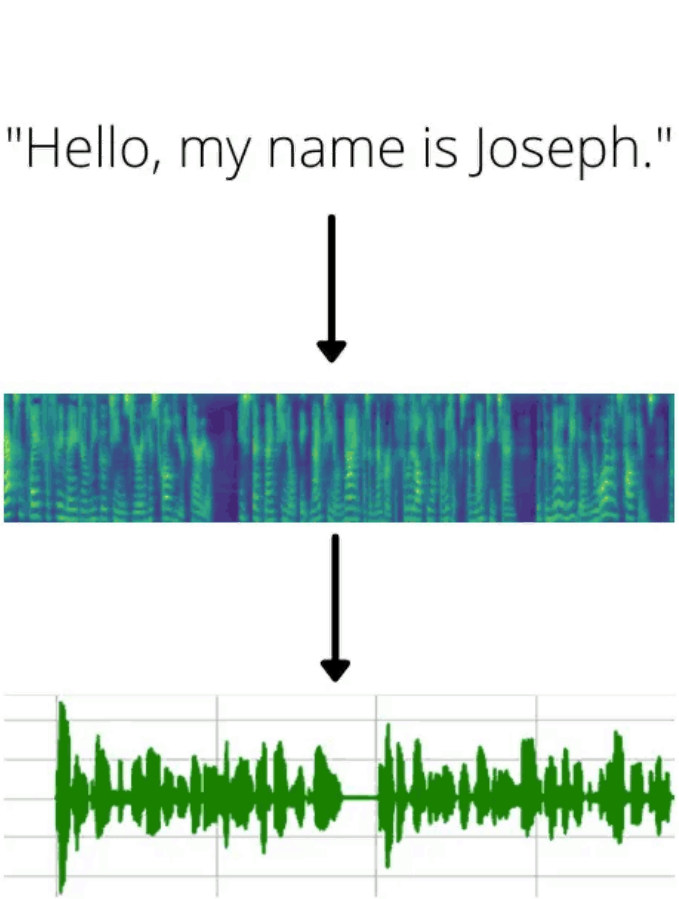
May 20, 2025

Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

EUROPEAN UNION
European Structural and Investment Fund
Operational Programme Research,
Development and Education

Mel Spectrograms

Speech Generation Pipeline



Text Input

Mel Spectrogram

Wave Form

*https://miro.medium.com/v2/1*QxNK0_jgds7tKIDaHGRz7A.png*

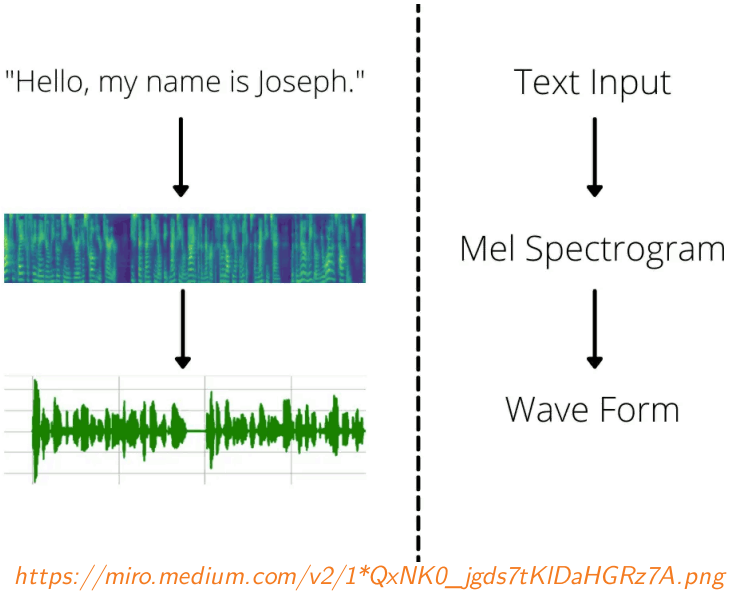
Speech Generation Pipeline

Traditionally, the input text was represented by **phonemes**.

	monophthongs				diphthongs			
VOWELS	iː sheep	ɪ ship	ʊ good	uː shoot	ɪə here	eɪ wait		
	e bed	ə teacher	ɜː bird	ɔː door	ʊə tourist	ɔɪ boy	əʊ show	
	æ cat	ʌ up	ɑː far	ɒ on	eə hair	aɪ my	aʊ cow	
CONSONANTS	p pea	b boat	t tea	d dog	ʧ cheese	ʤ June	k car	g go
	f fly	v video	θ think	ð this	s see	z zoo	ʃ shall	ʒ television
	m man	n now	ŋ sing	h hat	l love	r red	w wet	j yes

Phonemic Chart
voiced
unvoiced

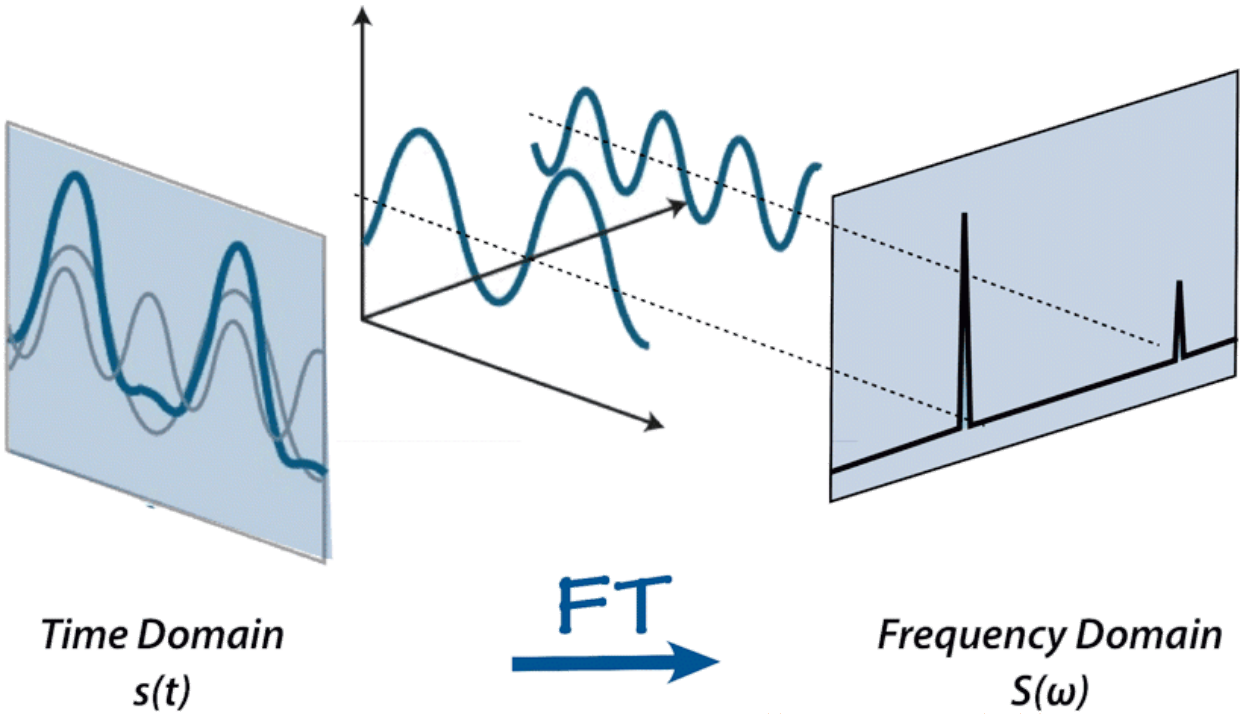
https://www.englishclub.com/images/pronunciation/Phonemic-Chart.jpg



https://miro.medium.com/v2/1*QxNK0_jgds7tKlDaHGRz7A.png

However, nowadays the input is most commonly represented using **characters** (some systems still allow specifying some input words using phonemes to allow more control over the output).

Time vs Frequency Domain



https://miro.medium.com/v2/0*LrglrtyqX7iK_UIN

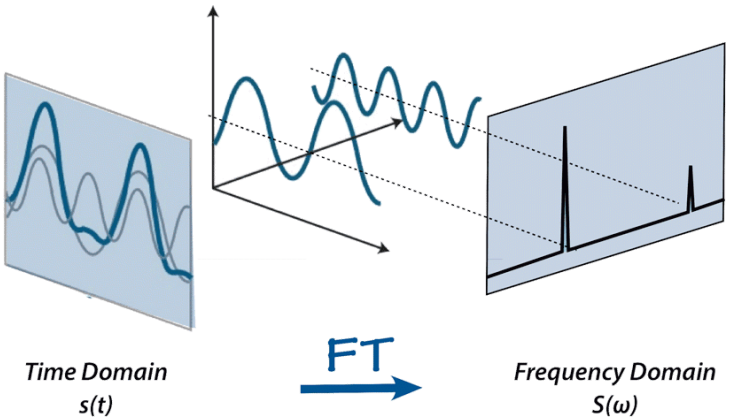
A sound can be represented in the **time domain**, as a sequence of amplitudes, i.e., how the signal changes over time.

Alternatively, it can be represented in a **frequency domain**, where a signal is represented by a *spectrum*: the magnitudes and phases of sinusoids with a collection of frequencies.

Time vs Frequency Domain

To convert a signal between time and frequency domains, Fourier transform can be used. In case of discrete-time signal, **discrete Fourier transform (DFT)** converts

- N complex-valued equally-spaced samples in the time domain to
- N complex-valued coefficients (magnitude and phase) of sinusoids with frequencies 1 Hz, 2 Hz, 3 Hz, ..., N Hz.



https://miro.medium.com/v2/0*LrglrtyqX7iK_UIN

Fast Fourier transform (FFT) is an algorithm that can perform discrete Fourier transform in in $\mathcal{O}(N \log N)$ time for $N = 2^k$ a power of two.

When the input signal consists of only real values, the upper half of the frequency spectrum is determined by the lower half ($x_{N-i} = \overline{x_i}$, where $\overline{x}$ denotes complex conjugate); therefore, a discrete signal of N samples can be represented by sinusoids with frequencies of 1, 2, 3, ..., $\frac{N}{2}$.

Spectrogram

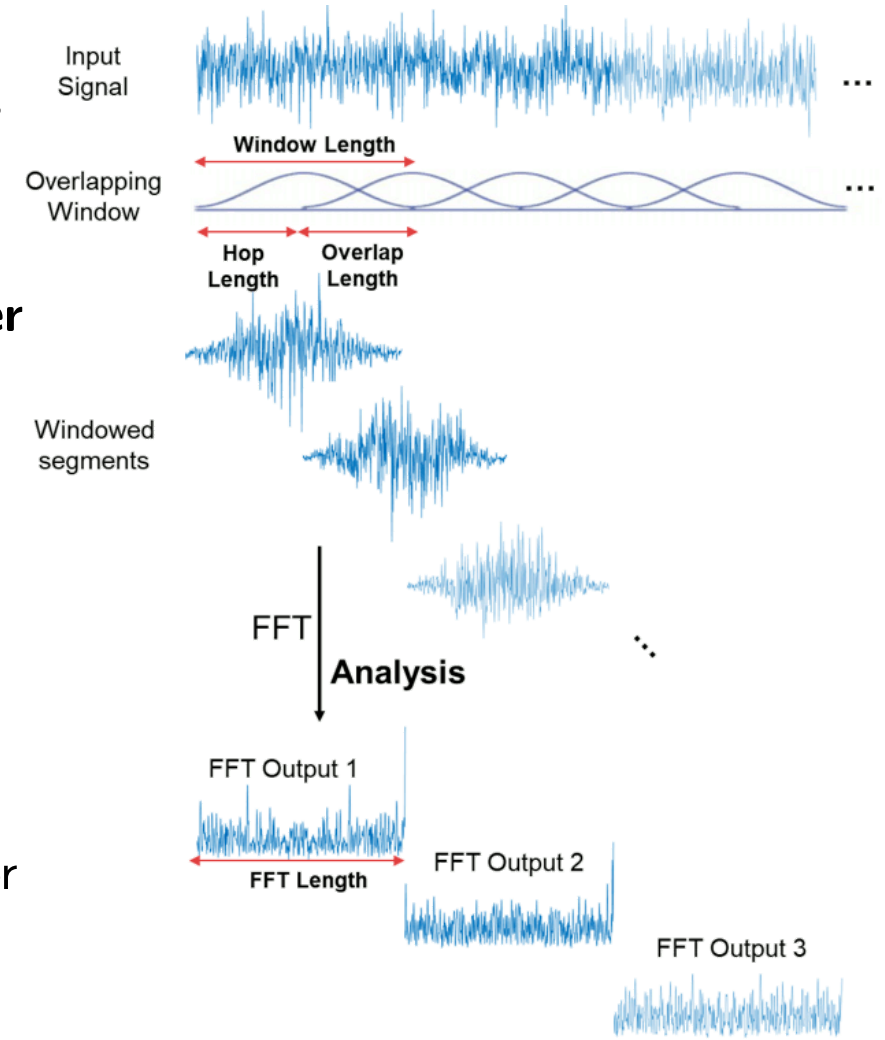
We can represent an arbitrary-length signal by a **spectrogram**, a series of fixed-length spectral densities (i.e., for every sinusoid, we compute some real-valued density, usually *power density* x^2).

The spectrogram is computed using **short-time Fourier transform (STFT)**, where

- DFT is performed on windows of fixed size,
- neighboring windows are shifted by a fixed hop length.

For every window, the signal is composed of sinusoids with frequencies up to half of the window length.

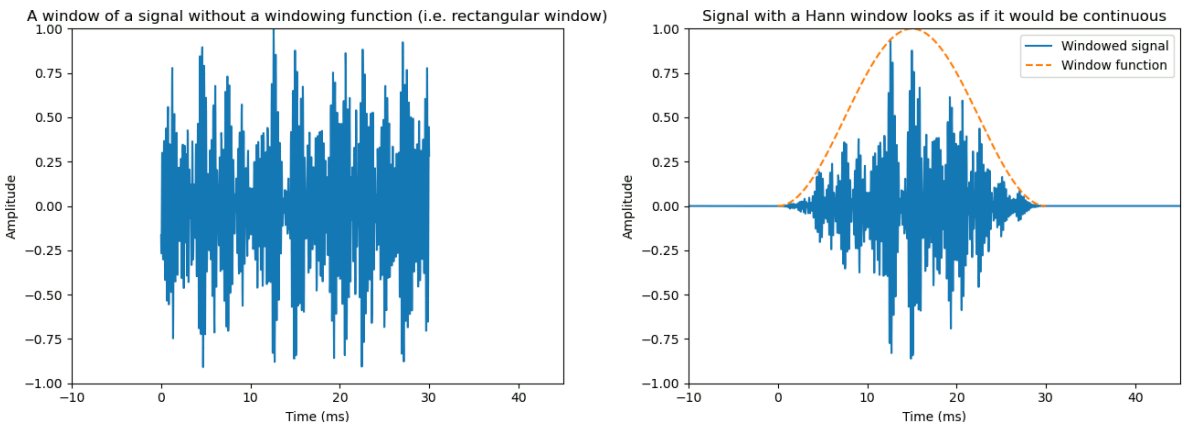
In speech synthesis, assuming a sampling frequency of 22.05kHz or 24kHz, commonly used values are 1024 for window length (denoted commonly as *n_fft*; ~45ms) and hop length of 256 (circa 11ms).



https://www.mathworks.com/help/dsp/ref/stft_output.png

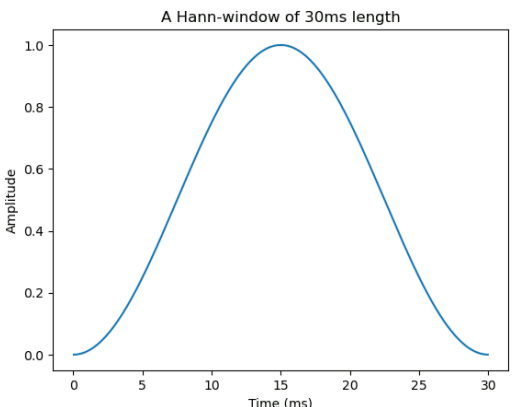
Spectrogram

To avoid discontinuities at the borders of the windows, *windowing* is used.

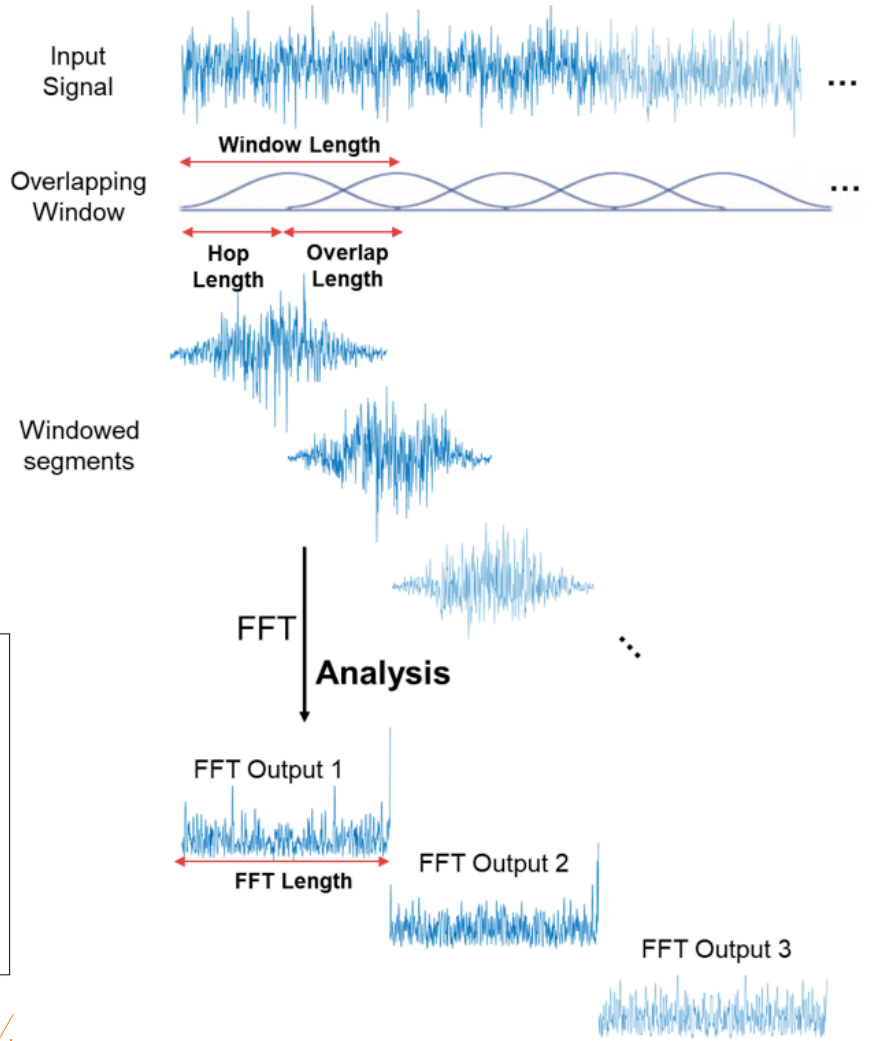


https://speechprocessingbook.aalto.fi/Representations/Windowing.html

Many possible window function exist; we will use the **Hann** window, which is as suitably scaled cosine function.



https://speechprocessingbook.aalto.fi/Representations/Windowing.html



https://www.mathworks.com/help/dsp/ref/stft_output.png

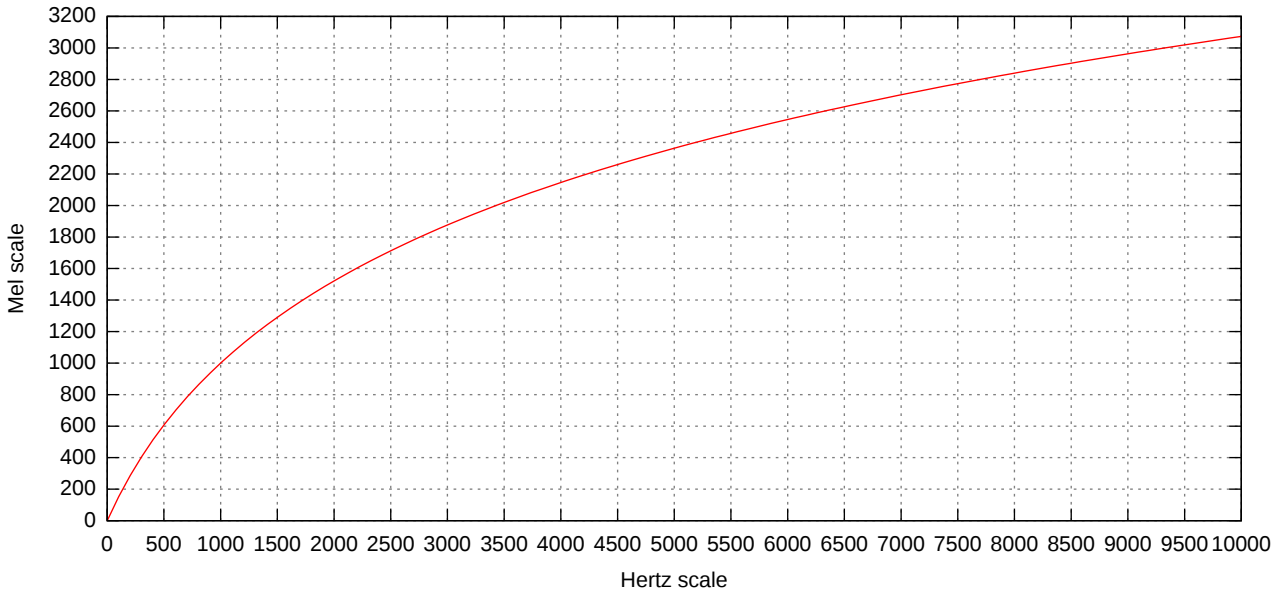
Mel Spectrogram

Linearly increasing frequencies are not perceived to be equal in distance from one another by humans.

Therefore, the **mel scale** (from the word melody) was proposed, so that the pitches were judged by listeners to be in equal distance.

There are in fact several mel scales, the most commonly used is

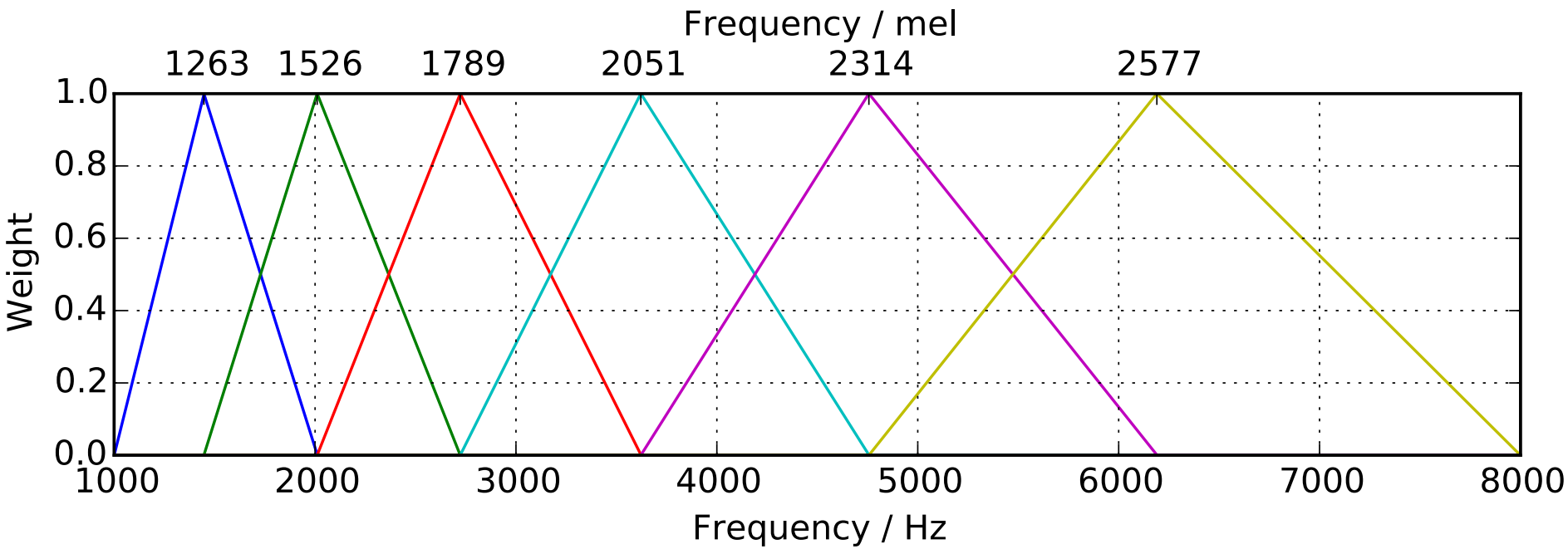
$$m = 2595 \log_{10} \left(1 + \frac{f}{700}\right).$$



https://upload.wikimedia.org/wikipedia/commons/a/aa/Mel-Hz_plot.svg

Mel Spectrogram

To convert spectrogram to mel spectrogram, a collection of triangular filters equally-distanced in the mel scale is used.



https://siggigue.github.io/pyfilterbank//melbank-1_00.pdf

Finally, we usually take the logarithm of the power spectrum, given that humans percieve loudness on logarithmic scale (i.e., dBs are also a logarithmic scale).

Mel Spectrogram

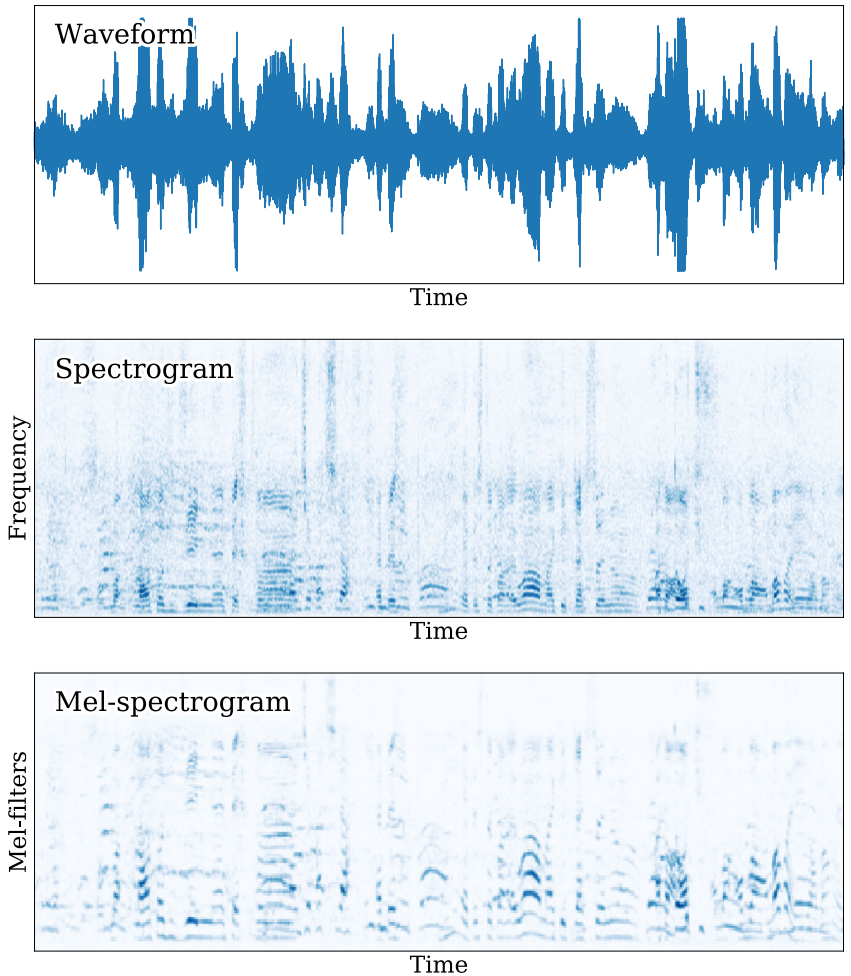
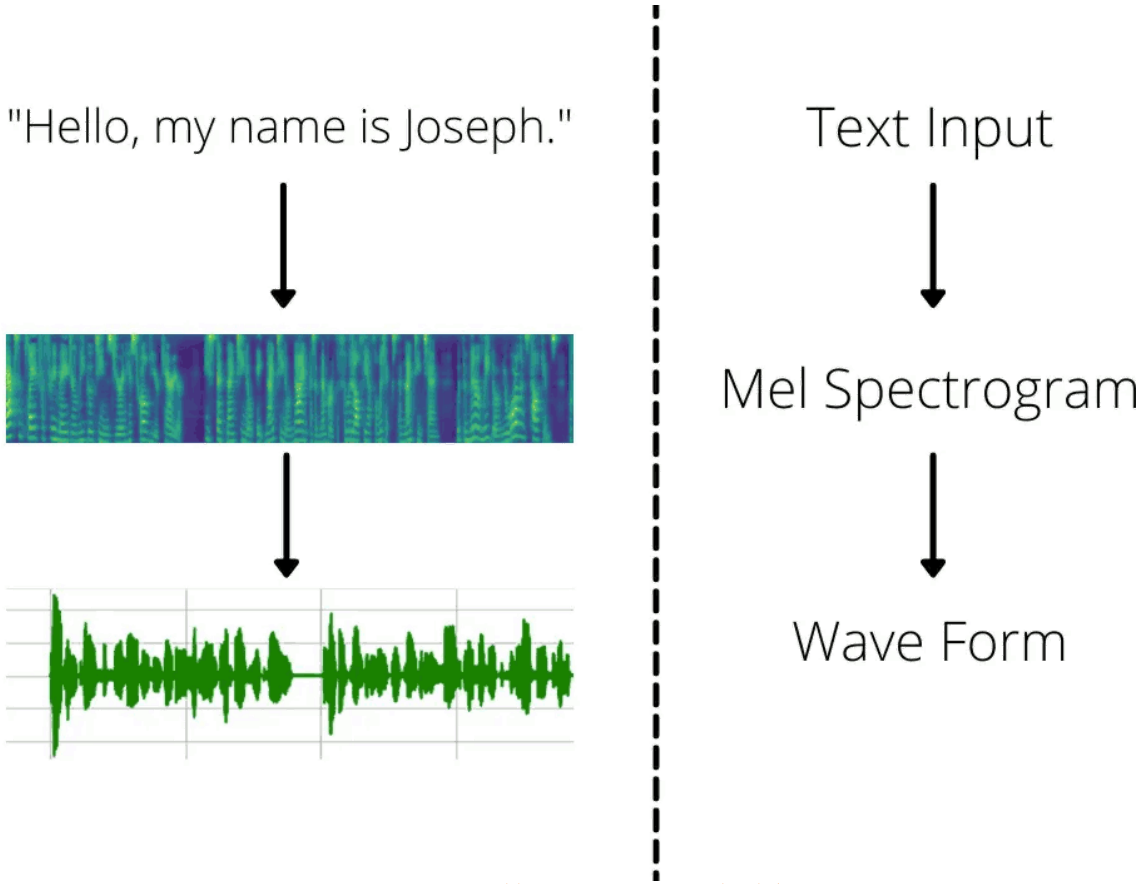


Fig. 1 Waveform, spectrogram, and mel-spectrogram of a 10-s speech segment obtained from Google AudioSet. The mel-spectrogram, based on the auditory-based mel-frequency scale, provides better resolution for lower frequencies than the spectrogram

Figure 1 of "Exploring convolutional, recurrent, and hybrid deep neural networks for speech and music detection in a large audio dataset", https://doi.org/10.1186/s13636-019-0152-1

Speech Generation Pipeline Revisited



*https://miro.medium.com/v2/1*QxNK0_jgds7tKIDaHGRz7A.png*

In the context of speech synthesis, a **vocoder** (combination of words voice and encoder) is the second part of the pipeline, converting a (usually mel) spectrogram to a waveform.

WaveNet

WaveNet

WaveNet, proposed in 2016, was one of the first neural architectures that produced high quality human speech.

Not many details were published in the original paper. From today's perspective, WaveNet is a vocoder capable of converting mel spectrograms to waveforms.

In the following, we start the description without the mel spectrogram conditioning, but we add it later.

Our goal is to model speech, using a convolutional auto-regressive model

$$P(\boldsymbol{x}) = \prod_t P(x_t | x_{t-1}, \ldots, x_1).$$

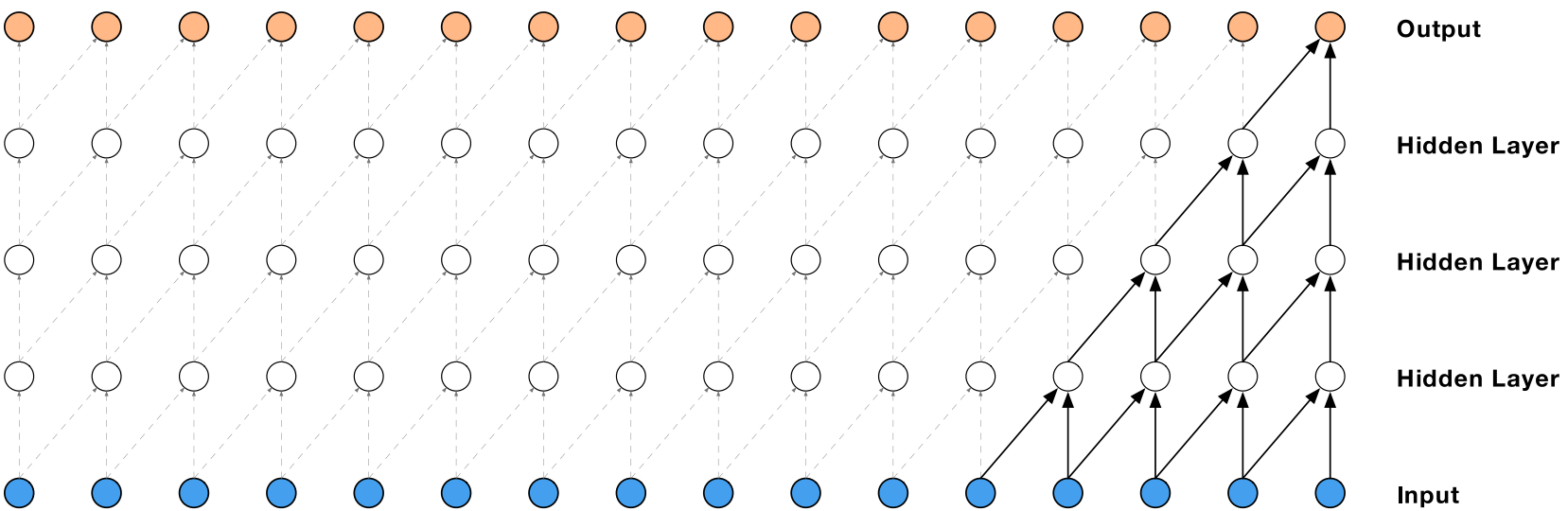


Figure 2: Visualization of a stack of causal convolutional layers.

Figure 2 of "WaveNet: A Generative Model for Raw Audio", https://arxiv.org/abs/1609.03499

However, to achieve larger receptive field, we utilize **dilated** (or **atrous**) convolutions:

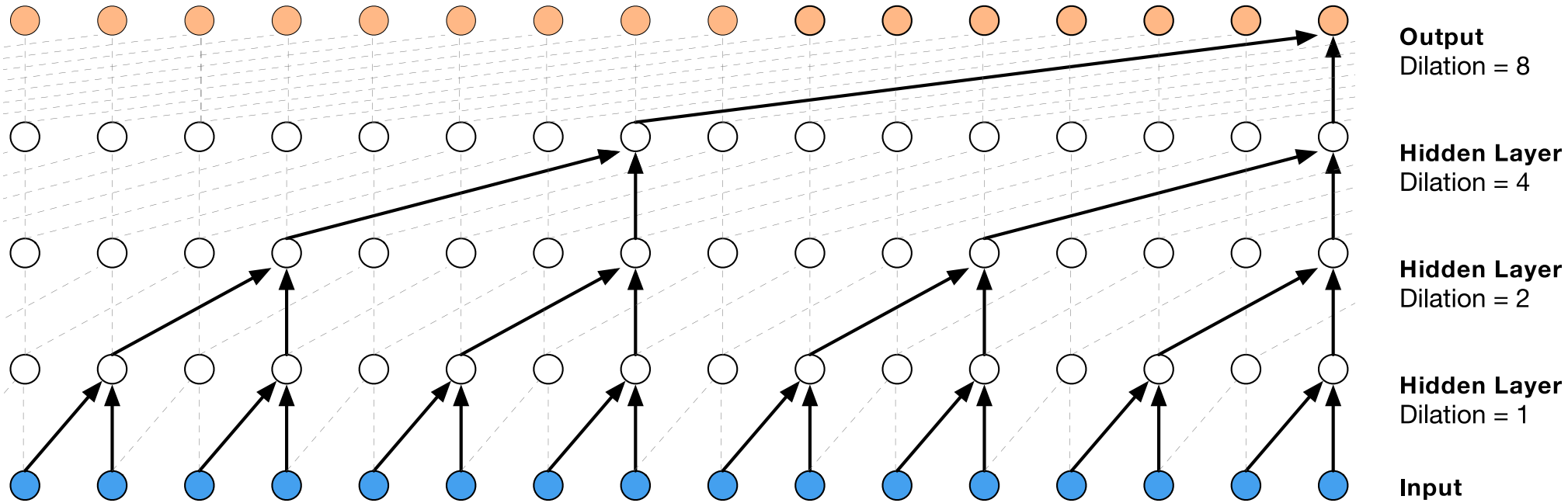
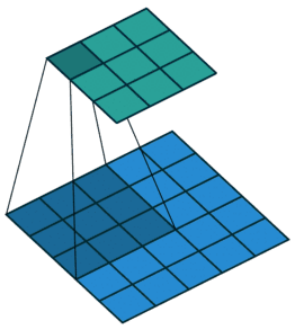


Figure 3: Visualization of a stack of *dilated* causal convolutional layers.

Figure 3 of "WaveNet: A Generative Model for Raw Audio", https://arxiv.org/abs/1609.03499

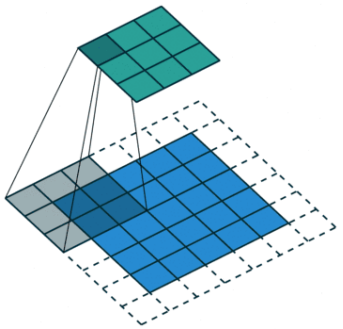
Dilated Versus Regular Versus Strided Convolutions

Regular Convolution



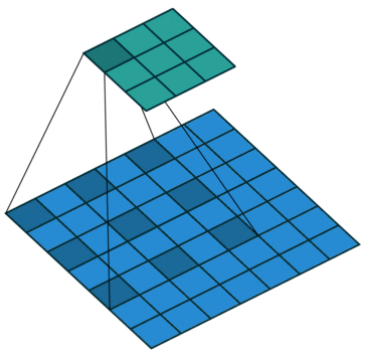
https://github.com/vdumoulin/conv_arithmetic

Strided Convolution



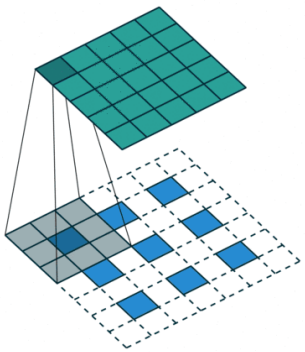
https://github.com/vdumoulin/conv_arithmetic

Dilated Convolution



https://github.com/vdumoulin/conv_arithmetic

Transposed Strided Convolution



https://github.com/vdumoulin/conv_arithmetic

Output Distribution

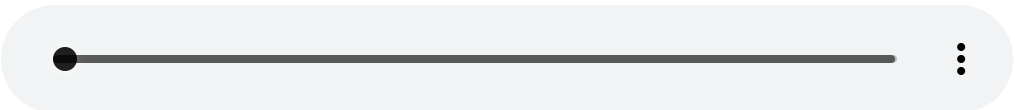
WaveNet generates audio with 16kHz frequency and 16-bit samples.

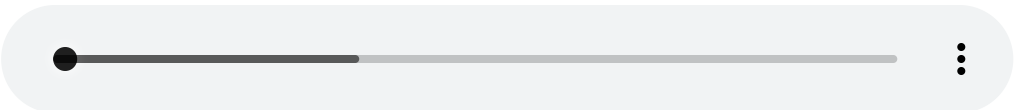
However, classification into $65\,536$ classes would not be efficient. Instead, WaveNet adopts the μ-law transformation, which passes the input samples in $[-1, 1]$ range through the μ-law encoding

$$\text{sign}(x)\frac{\log(1+255|x|)}{\log(1+255)},$$

and the resulting $[-1, 1]$ range is linearly quantized into 256 buckets.

The model therefore predicts each samples using classification into 256 classes, and then uses the inverse of the above transformation on the model predictions.





WaveNet – Architecture

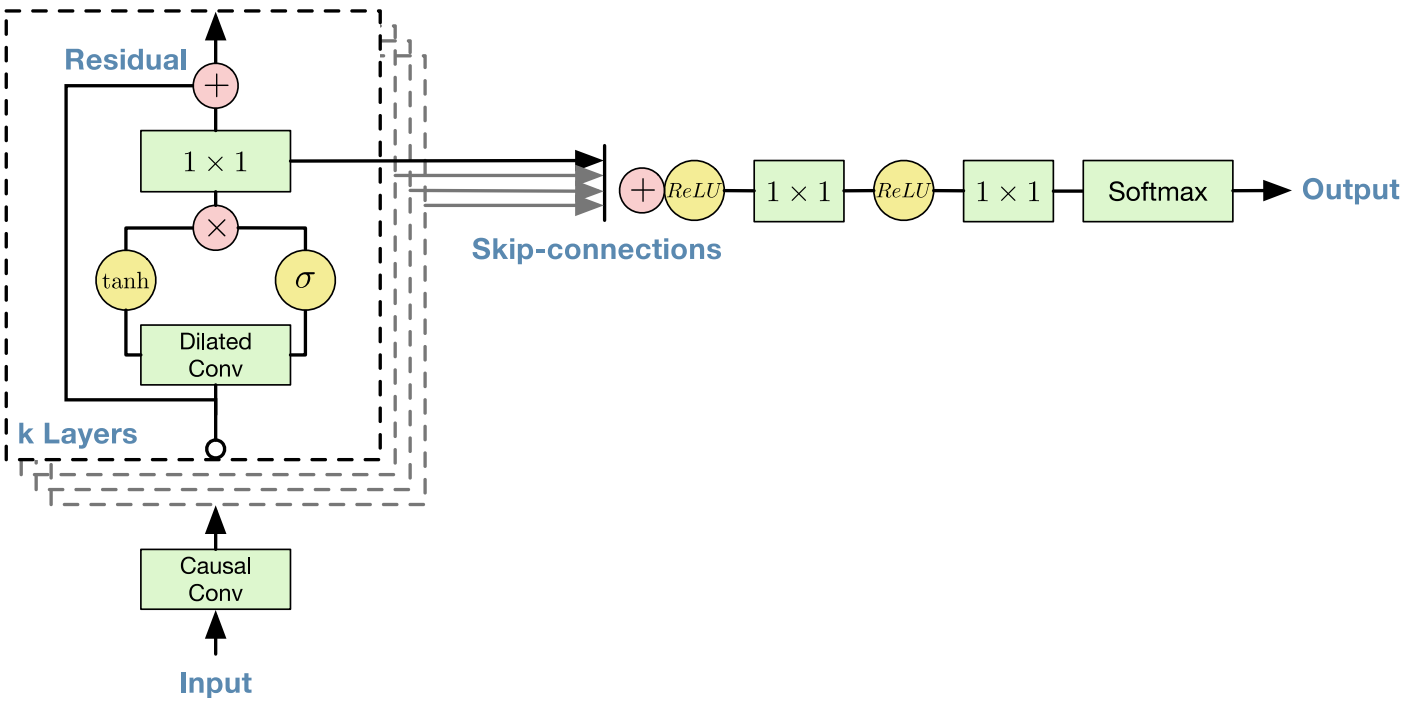


Figure 4: Overview of the residual block and the entire architecture.

Figure 4 of "WaveNet: A Generative Model for Raw Audio", https://arxiv.org/abs/1609.03499

The outputs of the dilated convolutions are passed through the *gated activation unit*:

$$\boldsymbol{z} = \tanh(\boldsymbol{W}_f * \boldsymbol{x}) \odot \sigma(\boldsymbol{W}_g * \boldsymbol{x}).$$

Global Conditioning

Global conditioning is performed by a single latent representation $\boldsymbol{h}$, changing the gated activation function to

$$\boldsymbol{z} = \tanh(\boldsymbol{W}_f * \boldsymbol{x} + \boldsymbol{V}_f \boldsymbol{h}) \odot \sigma(\boldsymbol{W}_g * \boldsymbol{x} + \boldsymbol{V}_g \boldsymbol{h}).$$

Local Conditioning

For local conditioning, we are given a time series $\boldsymbol{h}$, possibly with a lower sampling frequency (for example the mel spectrogram). We first use transposed convolutions $\boldsymbol{y} = f(\boldsymbol{h})$ to match resolution and then compute analogously to global conditioning

$$\boldsymbol{z} = \tanh(\boldsymbol{W}_f * \boldsymbol{x} + \boldsymbol{V}_f * \boldsymbol{y}) \odot \sigma(\boldsymbol{W}_g * \boldsymbol{x} + \boldsymbol{V}_g * \boldsymbol{y}).$$

The authors mention that using repetition instead of transposed convolution worked slightly worse.

WaveNet

The original paper did not mention hyperparameters, but later it was revealed that:

- 30 layers were used
  - grouped into 3 dilation stacks with 10 layers each
  - in a dilation stack, dilation rate increases by a factor of 2, starting with rate 1 and reaching maximum dilation of 512
- kernel size of a dilated convolution is 2 (and increased to 3 in Parallel WaveNet)
- residual connection has dimension 512
- gating layer uses 256+256 hidden units
- the 1×1 convolutions in the output step produce 256 filters
- trained for $1\,000\,000$ steps using Adam with a fixed learning rate of 2e-4

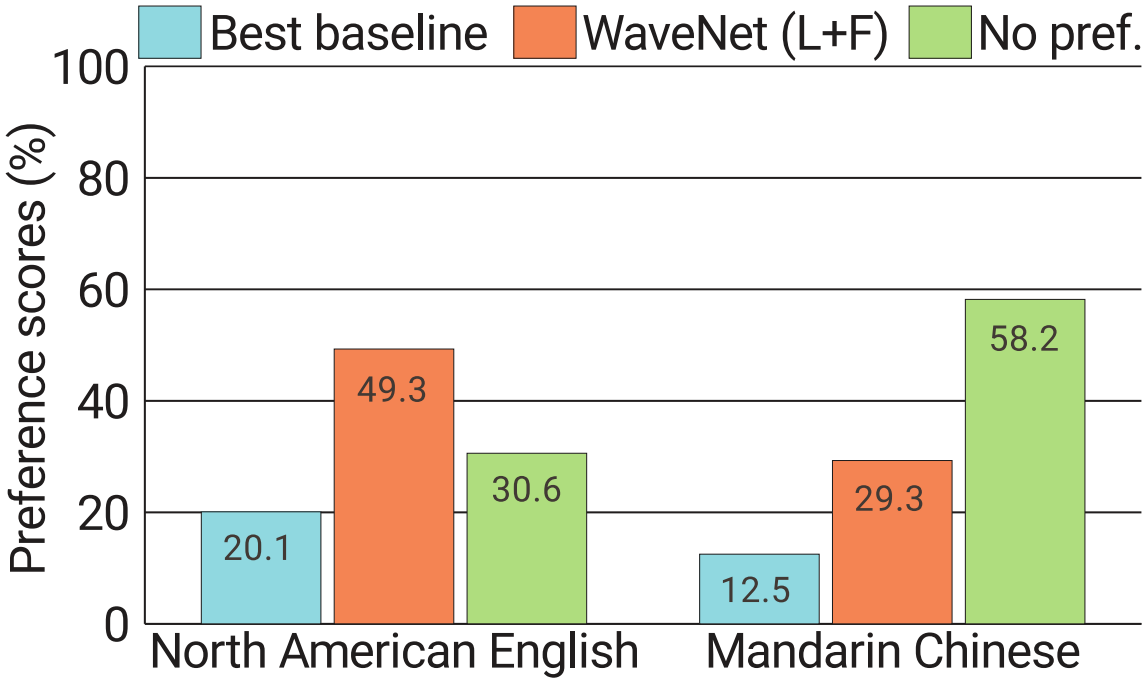


Figure 5: Subjective preference scores (%) of speech samples between (top) two baselines, (middle) two WaveNets, and (bottom) the best baseline and WaveNet. Note that LSTM and Concat correspond to LSTM-RNN-based statistical parametric and HMM-driven unit selection concatenative baseline synthesizers, and WaveNet (L) and WaveNet (L+F) correspond to the WaveNet conditioned on linguistic features only and that conditioned on both linguistic features and log F_0 values.

Figure 5 of "WaveNet: A Generative Model for Raw Audio", https://arxiv.org/abs/1609.03499

Gated Activations in Transformers

Gated Activations in Transformers

Similar gated activations seem to work the best in Transformers, in the FFN module.

Activation Name	Formula	$\mathrm{FFN}(x; \boldsymbol{W}_1, \boldsymbol{W}_2)$
ReLU	$\max(0, x)$	$\max(0, \boldsymbol{x}\boldsymbol{W}_1)\boldsymbol{W}_2$
GELU	$x\Phi(x)$	$\mathrm{GELU}(\boldsymbol{x}\boldsymbol{W}_1)\boldsymbol{W}_2$
Swish	$x\sigma(x)$	$\mathrm{Swish}(\boldsymbol{x}\boldsymbol{W}_1)\boldsymbol{W}_2$

There are several variants of the new gated activations:

Activation Name	Formula	$\mathrm{FFN}(x; \boldsymbol{W}, \boldsymbol{V}, \boldsymbol{W}_2)$
GLU (Gated Linear Unit)	$\sigma(\boldsymbol{x}\boldsymbol{W} + \boldsymbol{b}) \odot (\boldsymbol{x}\boldsymbol{V} + \boldsymbol{c})$	$(\sigma(\boldsymbol{x}\boldsymbol{W}) \odot \boldsymbol{x}\boldsymbol{V})\boldsymbol{W}_2$
ReGLU	$\max(0, \boldsymbol{x}\boldsymbol{W} + \boldsymbol{b}) \odot (\boldsymbol{x}\boldsymbol{V} + \boldsymbol{c})$	$(\max(0, \boldsymbol{x}\boldsymbol{W}) \odot \boldsymbol{x}\boldsymbol{V})\boldsymbol{W}_2$
GEGLU	$\mathrm{GELU}(\boldsymbol{x}\boldsymbol{W} + \boldsymbol{b}) \odot (\boldsymbol{x}\boldsymbol{V} + \boldsymbol{c})$	$(\mathrm{GELU}(\boldsymbol{x}\boldsymbol{W}) \odot \boldsymbol{x}\boldsymbol{V})\boldsymbol{W}_2$
SwiGLU	$\mathrm{Swish}(\boldsymbol{x}\boldsymbol{W} + \boldsymbol{b}) \odot (\boldsymbol{x}\boldsymbol{V} + \boldsymbol{c})$	$(\mathrm{Swish}(\boldsymbol{x}\boldsymbol{W}) \odot \boldsymbol{x}\boldsymbol{V})\boldsymbol{W}_2$

Gated Activations in Transformers

	Score Average	CoLA MCC	SST-2 Acc	MRPC F1	MRPC Acc	STSB PCC	STSB SCC	QQP F1	QQP Acc	MNLIm Acc	MNLImm Acc	QNLI Acc	RTE Acc
FFN_{ReLU}	83.80	51.32	94.04	**93.08**	**90.20**	89.64	89.42	89.01	91.75	85.83	86.42	92.81	80.14
FFN_{GELU}	83.86	53.48	94.04	92.81	**90.20**	89.69	89.49	88.63	91.62	85.89	86.13	92.39	80.51
FFN_{Swish}	83.60	49.79	93.69	92.31	89.46	89.20	88.98	88.84	91.67	85.22	85.02	92.33	81.23
FFN_{GLU}	84.20	49.16	94.27	92.39	89.46	89.46	89.35	88.79	91.62	86.36	86.18	92.92	**84.12**
FFN_{GEGLU}	84.12	53.65	93.92	92.68	89.71	90.26	90.13	89.11	91.85	86.15	86.17	92.81	79.42
$FFN_{Bilinear}$	83.79	51.02	**94.38**	92.28	89.46	90.06	89.84	88.95	91.69	**86.90**	**87.08**	92.92	81.95
FFN_{SwiGLU}	84.36	51.59	93.92	92.23	88.97	**90.32**	**90.13**	**89.14**	**91.87**	86.45	86.47	**92.93**	83.39
FFN_{ReGLU}	**84.67**	**56.16**	**94.38**	92.06	89.22	89.97	89.85	88.86	91.72	86.20	86.40	92.68	81.59

Table 2 of "GLU Variants Improve Transformer", https://arxiv.org/abs/2002.05202

	EM	F1
FFN_{ReLU}	83.18	90.87
FFN_{GELU}	83.09	90.79
FFN_{Swish}	83.25	90.76
FFN_{GLU}	82.88	90.69
FFN_{GEGLU}	83.55	91.12
$FFN_{Bilinear}$	**83.82**	91.06
FFN_{SwiGLU}	83.42	91.03
FFN_{ReGLU}	83.53	**91.18**

Table 4 of "GLU Variants Improve Transformer", https://arxiv.org/abs/2002.05202

Model	Params	Ops	Step/s	Early loss	Final loss	SGLUE	XSum	WebQ	WMT EnDe
Vanilla Transformer	$223M$	$11.1T$	3.50	2.182 ± 0.005	1.838	71.66	17.78	23.02	26.62
GeLU	$223M$	$11.1T$	3.58	2.179 ± 0.003	1.838	**75.79**	**17.86**	**25.13**	26.47
Swish	$223M$	$11.1T$	3.62	2.186 ± 0.003	1.847	**73.77**	17.74	**24.34**	**26.75**
ELU	$223M$	$11.1T$	3.56	2.270 ± 0.007	1.932	67.83	16.73	23.02	26.08
GLU	$223M$	$11.1T$	3.59	2.174 ± 0.003	**1.814**	**74.20**	**17.42**	24.34	**27.12**
GeGLU	$223M$	$11.1T$	3.55	2.130 ± 0.006	**1.792**	**75.96**	**18.27**	**24.87**	**26.87**
ReGLU	$223M$	$11.1T$	3.57	2.145 ± 0.004	**1.803**	**76.17**	**18.36**	**24.87**	**27.02**
SeLU	$223M$	$11.1T$	3.55	2.315 ± 0.004	1.948	68.76	16.76	22.75	25.99
SwiGLU	$223M$	$11.1T$	3.53	2.127 ± 0.003	**1.789**	**76.00**	**18.20**	**24.34**	**27.02**
LiGLU	$223M$	$11.1T$	3.59	2.149 ± 0.005	**1.798**	**75.34**	**17.97**	**24.34**	26.53
Sigmoid	$223M$	$11.1T$	3.63	2.291 ± 0.019	1.867	**74.31**	17.51	23.02	26.30
Softplus	$223M$	$11.1T$	3.47	2.207 ± 0.011	1.850	**72.45**	17.65	**24.34**	**26.89**

Table 1 of "Do Transformer Modifications Transfer Across Implementations and Applications?", https://arxiv.org/abs/2102.11972

Parallel WaveNet

Parallel WaveNet

Parallel WaveNet is an improvement of the original WaveNet by the same authors.

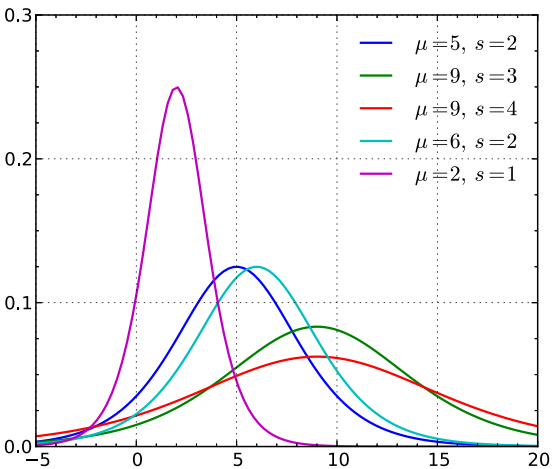
First, the output distribution was changed from 256 μ-law values to a Mixture of Logistic (suggested in another paper – PixelCNN++, but reused in other architectures since):

$$x \sim \sum_i \pi_i \operatorname{Logistic}(\mu_i, s_i).$$

The logistic distribution is a distribution with a σ as cumulative density function (where the mean and scale is parametrized by μ and s). Therefore, we can write

$$P(x|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{s}) = \sum_i \pi_i \left[\sigma\left(\frac{x + 0.5 - \mu_i}{\boldsymbol{s}_i}\right) - \sigma\left(\frac{x - 0.5 - \mu_i}{\boldsymbol{s}_i}\right)\right],$$

where we replace -0.5 and 0.5 in the edge cases by $-\infty$ and ∞.



https://commons.wikimedia.org/wiki/File:Logisticpdfunction.svg

In Parallel WaveNet teacher, 10 mixture components are used.

Parallel WaveNet

Auto-regressive (sequential) inference is extremely slow in WaveNet.

Instead, we model $P(x_t)$ as $P(x_t|\boldsymbol{z}_{<t}) = \text{Logistic}\left(x_t; \mu^1(\boldsymbol{z}_{<t}), s^1(\boldsymbol{z}_{<t})\right)$ for a *random* $\boldsymbol{z}$ drawn from a logistic distribution $\text{Logistic}(\mathbf{0}, \mathbf{1})$. Therefore, using the reparametrization trick,

$$x_t^1 = \mu^1(\boldsymbol{z}_{<t}) + z_t \cdot s^1(\boldsymbol{z}_{<t}).$$

Usually, one iteration of the algorithm does not produce good enough results – consequently, 4 iterations were used by the authors. In further iterations,

$$x_t^i = \mu^i(\boldsymbol{x}_{<t}^{i-1}) + x_t^{i-1} \cdot s^i(\boldsymbol{x}_{<t}^{i-1}).$$

After N iterations, $P(\boldsymbol{x}_t^N|\boldsymbol{z}_{<t})$ is a logistic distribution with location $\boldsymbol{\mu}^{\text{tot}}$ and scale $\boldsymbol{s}^{\text{tot}}$:

$$\mu_t^{\text{tot}} = \sum_{i=1}^{N} \mu^i(\boldsymbol{x}_{<t}^{i-1}) \cdot \left(\prod_{j>i}^{N} s^j(\boldsymbol{x}_{<t}^{j-1})\right) \text{ and } s_t^{\text{tot}} = \prod_{i=1}^{N} s^i(\boldsymbol{x}_{<t}^{i-1}),$$

where we have denoted $\boldsymbol{z}$ as $\boldsymbol{x}^0$ for convenience.

Parallel WaveNet

The consequences of changing the model from $x_t = f(x_{t-1}, \ldots, x_1)$ to

$$x_t^1 = \mu^1(\boldsymbol{z}_{<t}) + z_t \cdot s^1(\boldsymbol{z}_{<t})$$
$$x_t^i = \mu^i(\boldsymbol{x}_{<t}^{i-1}) + x_t^{i-1} \cdot s^i(\boldsymbol{x}_{<t}^{i-1})$$

are:

- During inference, the prediction can be computed in parallel, because x_t^i depends only on $\boldsymbol{x}_{<t}^{i-1}$, not on $x_{<t}^i$.
- However, we cannot perform training in parallel. If we try maximizing the log-likelihood of an input sequence $\boldsymbol{x}^1$, we need to find out which $\boldsymbol{z}$ sequence generates it.
  - The z_1 can be computed using x_1^1.
  - However, z_2 depends not only on x_1^1 and x_2^1, but also on z_1; generally, z_t depends on $\boldsymbol{x}^1$ and also on all $\boldsymbol{z}_{<t}$, and can be computed only sequentially.

Therefore, WaveNet can perform parallel training and sequential inference, while the proposed model can perform parallel inference but sequential training.

Probability Density Distillation

The authors propose to train the network by a **probability density distillation** using a teacher WaveNet (producing a mixture of logistic with 10 components) with KL-divergence as a loss.

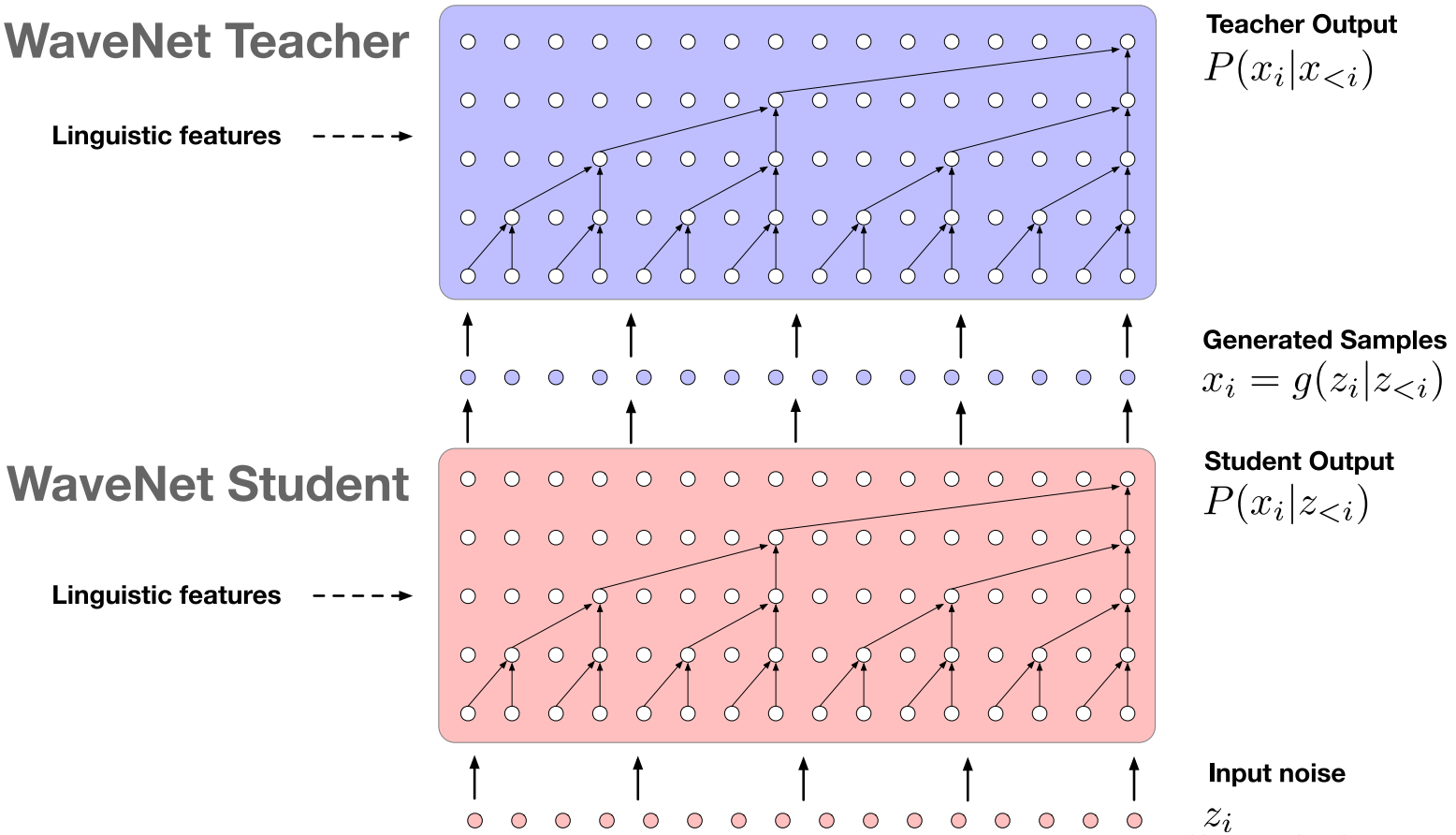


Figure 2 of "Parallel WaveNet: Fast High-Fidelity Speech Synthesis", https://arxiv.org/abs/1711.10433

Probability Density Distillation

Therefore, instead of computing $\boldsymbol{z}$ from some gold $\boldsymbol{x}_g$, we

- sample a random $\boldsymbol{z}$;
- generate the output $\boldsymbol{x}$;
- use the teacher WaveNet model to estimate the log-likelihood of $\boldsymbol{x}$;
- update the student to match the log-likelihood of the teacher.

Denoting the teacher distribution as P_T and the student distribution as P_S, the loss is

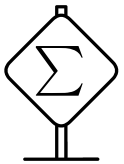
$$D_{\mathrm{KL}}(P_S||P_T) = H(P_S, P_T) - H(P_S).$$

Therefore, we do not only minimize cross-entropy, but we also try to keep the entropy of the student as high as possible – it is indeed crucial not to match just the mode of the teacher.

- Consider a teacher generating white noise, where every sample comes from $\mathcal{N}(0, 1)$ – in this case, the cross-entropy loss of a constant $\mathbf{0}$, complete silence, would be maximal.

In a sense, probability density distillation is similar to GANs. However, the teacher is kept fixed, and the student does not attempt to fool it but to match its distribution instead.

Probability Density Distillation Details



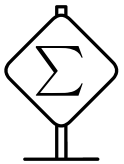
Because the entropy of a logistic distribution $\text{Logistic}(\mu, s)$ is $\log s + 2$, the entropy term $H(P_S)$ can be rewritten as follows:

$$H(P_S) = \mathbb{E}_{z \sim \text{Logistic}(0,1)} \left[\sum_{t=1}^{T} -\log p_S(x_t | \boldsymbol{z}_{<t}) \right]$$

$$= \mathbb{E}_{z \sim \text{Logistic}(0,1)} \left[\sum_{t=1}^{T} \log s(\boldsymbol{z}_{<t}, \boldsymbol{\theta}) \right] + 2T.$$

Therefore, this term can be computed without having to generate $\boldsymbol{x}$.

Probability Density Distillation Details



However, the cross-entropy term $H(P_S, P_T)$ requires sampling from P_S to estimate:

$$
\begin{aligned}
H(P_S, P_T) &= \int_{\boldsymbol{x}} -P_S(\boldsymbol{x}) \log P_T(\boldsymbol{x}) \\
&= \sum_{t=1}^{T} \int_{\boldsymbol{x}} -P_S(\boldsymbol{x}) \log P_T(x_t|\boldsymbol{x}_{<t}) \\
&= \sum_{t=1}^{T} \int_{\boldsymbol{x}} -P_S(\boldsymbol{x}_{<t}) P_S(x_t|\boldsymbol{x}_{<t}) P_S(\boldsymbol{x}_{>t}|\boldsymbol{x}_{\leq t}) \log P_T(x_t|\boldsymbol{x}_{<t}) \\
&= \sum_{t=1}^{T} \mathbb{E}_{P_S(\boldsymbol{x}_{<t})} \left[\int_{x_t} -P_S(x_t|\boldsymbol{x}_{<t}) \log P_T(x_t|\boldsymbol{x}_{<t}) \underbrace{\int_{\boldsymbol{x}_{>t}} P_S(\boldsymbol{x}_{>t}|\boldsymbol{x}_{\leq t})}_{1} \right] \\
&= \sum_{t=1}^{T} \mathbb{E}_{P_S(\boldsymbol{x}_{<t})} H\Big(P_S(x_t|\boldsymbol{x}_{<t}), P_T(x_t|\boldsymbol{x}_{<t}) \Big).
\end{aligned}
$$

Probability Density Distillation Details

$$H(P_S, P_T) = \sum_{t=1}^{T} \mathbb{E}_{P_S(\boldsymbol{x}_{<t})} H\Big(P_S(x_t|\boldsymbol{x}_{<t}), P_T(x_t|\boldsymbol{x}_{<t})\Big)$$

We can therefore estimate $H(P_S, P_T)$ by:

- drawing a single sample $\boldsymbol{x}$ from the student P_S *[a Logistic($\boldsymbol{\mu}^{\text{tot}}, \boldsymbol{s}^{\text{tot}}$)]*,
- compute all $P_T(x_t|\boldsymbol{x}_{<t})$ from the teacher in parallel *[mixture of logistic distributions]*,
- and finally evaluate $H(P_S(x_t|\boldsymbol{x}_{<t}), P_T(x_t|\boldsymbol{x}_{<t}))$ by sampling multiple different x_t from the $P_S(x_t|\boldsymbol{x}_{<t})$.

The authors state that this unbiased estimator has a much lower variance than naively evaluating a single sequence sample under the teacher using the original formulation.

Parallel WaveNet

The Parallel WaveNet model consists of 4 iterations with 10, 10, 10, 30 layers, respectively. The dimension of the residuals and the gating units is 64 (compared to 512 in WaveNet).

The Parallel WaveNet generates over 500k samples per second, compared to ~170 samples per second of a regular WaveNet – more than a 1000 times speedup.

Method	**Subjective 5-scale MOS**
16kHz, 8-bit μ-law, 25h data:	
LSTM-RNN parametric [27]	3.67 ± 0.098
HMM-driven concatenative [27]	3.86 ± 0.137
WaveNet [27]	4.21 ± 0.081
24kHz, 16-bit linear PCM, 65h data:	
HMM-driven concatenative	4.19 ± 0.097
Autoregressive WaveNet	4.41 ± 0.069
Distilled WaveNet	4.41 ± 0.078

Table 1 of "Parallel WaveNet: Fast High-Fidelity Speech Synthesis", https://arxiv.org/abs/1711.10433

For comparison, using a single iteration with 30 layers achieve MOS of 4.21.

Parallel WaveNet

The Parallel WaveNet can be trained to generate speech of multiple speakers (using the global conditioning). Because such a model needs larger capacity, it used 30 layers in every iteration (instead of 10, 10, 10, 30).

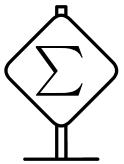
	Parametric	Concatenative	Distilled WaveNet
English speaker 1 (female - 65h data)	3.88	4.19	4.41
English speaker 2 (male - 21h data)	3.96	4.09	4.34
English speaker 3 (male - 10h data)	3.77	3.65	4.47
English speaker 4 (female - 9h data)	3.42	3.40	3.97
Japanese speaker (female - 28h data)	4.07	3.47	4.23

Table 2: Comparison of MOS scores on English and Japanese with multi-speaker distilled WaveNets. Note that some speakers sounded less appealing to people and always get lower MOS, however distilled parallel WaveNet always achieved significantly better results.

Table 2 of "Parallel WaveNet: Fast High-Fidelity Speech Synthesis", https://arxiv.org/abs/1711.10433

Parallel WaveNet – Additional Losses

To generate high-quality audio, the probability density distillation is not entirely sufficient. The authors therefore introduce additional losses:



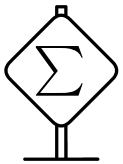
- **power loss**: ensures the power in different frequency bands is on average similar between the generated speech and human speech. For a conditioned training data $(\boldsymbol{x}, \boldsymbol{c})$ and WaveNet student g, the loss is

$$\left\| \mathrm{STFT}(g(\boldsymbol{z}, \boldsymbol{c})) - \mathrm{STFT}(\boldsymbol{x}) \right\|^2.$$

- **perceptual loss**: apart from the power in frequency bands, we can use a pre-trained classifier to extract features from generated and human speech and add a loss measuring their difference. The authors propose the loss as squared Frobenius norm of differences between Gram matrices (uncentered covariance matrices) of features of a WaveNet-like classifier predicting phones from raw audio.
- **contrastive loss**: to make the model respect the conditioning instead of generating outputs with high likelihood independent on the conditioning, the authors propose a contrastive distillation loss ($\gamma = 0.3$ is used in the paper):

$$D_{\mathrm{KL}}\left(P_S(\boldsymbol{c}_1) || P_T(\boldsymbol{c}_1)\right) - \gamma D_{\mathrm{KL}}\left(P_S(\boldsymbol{c}_1) || P_T(\boldsymbol{c}_2)\right).$$

Parallel WaveNet – Additional Losses



Method	Preference Scores versus baseline concatenative system Win - Lose - Neutral
Losses used	
KL + Power	60% - 15% - 25%
KL + Power + Perceptual	66% - 10% - 24%
KL + Power + Perceptual + Contrastive (= default)	65% - 9% - 26%

Table 3: Performance with respect to different combinations of loss terms. We report preference comparison scores since their mean opinion scores tend to be very close and inconclusive.

Table 3 of "Parallel WaveNet: Fast High-Fidelity Speech Synthesis", https://arxiv.org/abs/1711.10433

Tacotron 2

Tacotron 2

Tacotron 2 model presents end-to-end speech synthesis directly from text. It consists of two components trained separately:

- a seq2seq model processing input characters and generating mel spectrograms;
- a Parallel WaveNet generating the speech from Mel spectrograms.

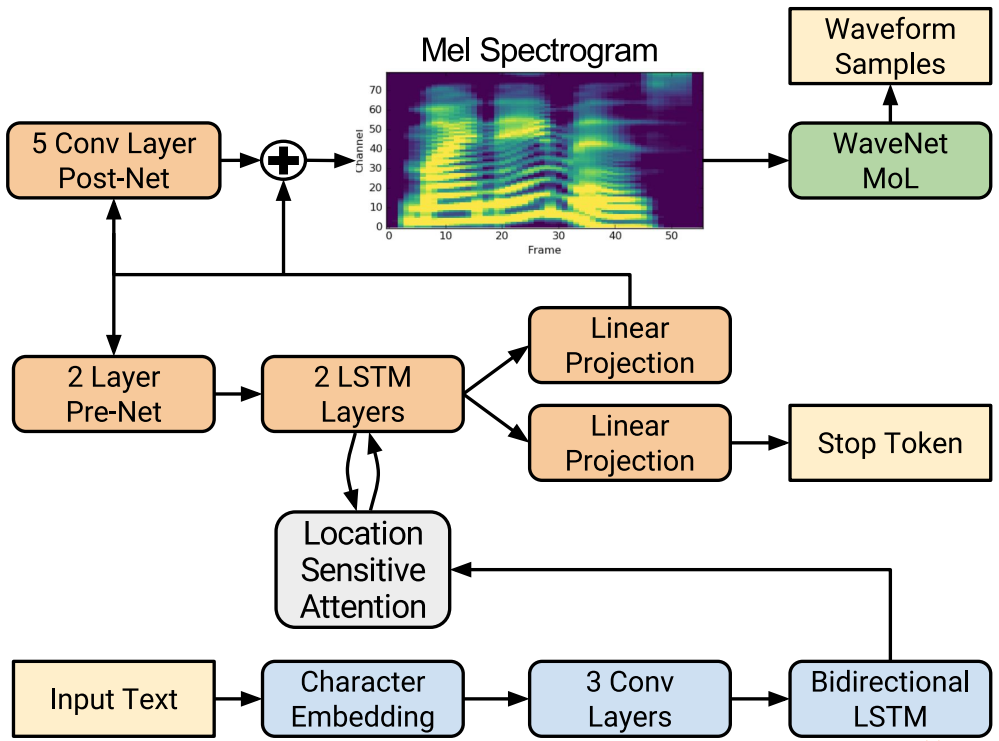


Figure 1 of "Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions", https://arxiv.org/abs/1712.05884

The Mel spectrograms used in Tacotron 2 are fairly standard:

- The authors propose a frame size of 50ms, 12.5ms frame hop, and a Hann window.
- STFT magnitudes are transformed into 80-channel Mel scale spanning 175Hz to 7.6kHz, followed by a log dynamic range compression (clipping input values to at least 0.01).

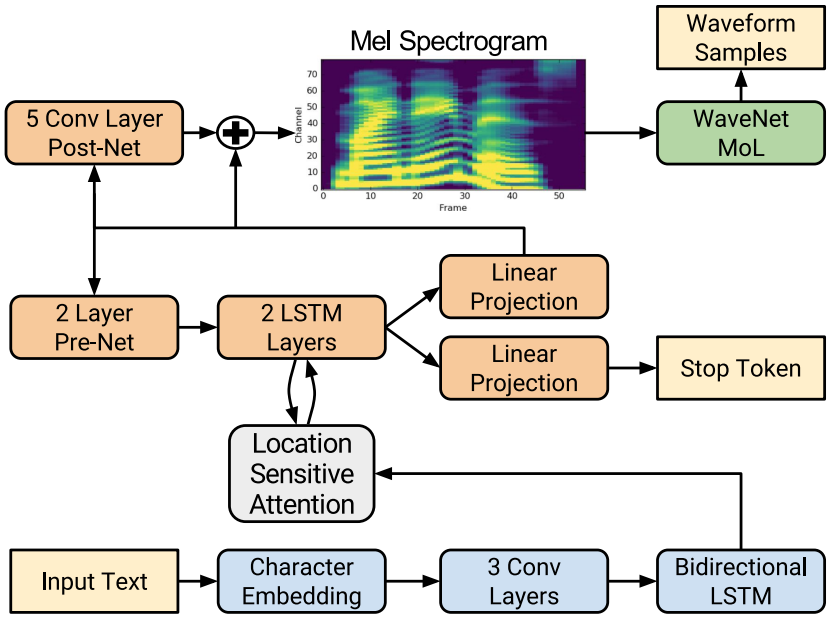


Figure 1 of "Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions", https://arxiv.org/abs/1712.05884

Architecture

The characters are represented using 512-dimensional embeddings, processed by 3 Conv+BN+ReLU with kernel size 5 and 512 channels, following by a single bi-directional LSTM with 512 units.

Tacotron 2

To make sequential processing of input characters easier, Tacotron 2 utilizes *location-sensitive attention*, which is an extension of the additive attention. While the additive (Bahdanau) attention computes

$$\boldsymbol{\alpha}_i = \mathrm{Attend}(\boldsymbol{s}_{i-1}, \boldsymbol{h}),$$

$$\alpha_{ij} = \mathrm{softmax}\left(\boldsymbol{v}^\top \tanh(\boldsymbol{V}\boldsymbol{h}_j + \boldsymbol{W}\boldsymbol{s}_{i-1} + \boldsymbol{b})\right),$$

the location-sensitive attention also inputs the previous cummulative attention weights $\hat{\boldsymbol{\alpha}}_i$ into the current attention computation:

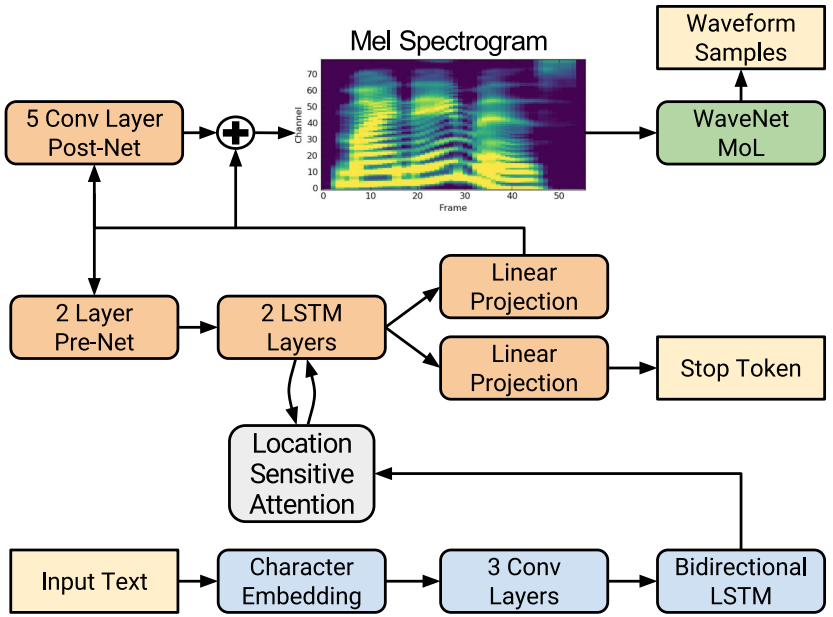


Figure 1 of "Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions", https://arxiv.org/abs/1712.05884

$$\boldsymbol{\alpha}_i = \mathrm{Attend}(\boldsymbol{s}_{i-1}, \boldsymbol{h}, \hat{\boldsymbol{\alpha}}_{i-1}).$$

In detail, the previous attention weights are processed by a 1-D convolution with kernel $\boldsymbol{F}$:

$$\alpha_{ij} = \mathrm{softmax}\left(\boldsymbol{v}^\top \tanh(\boldsymbol{V}\boldsymbol{h}_j + \boldsymbol{W}\boldsymbol{s}_{i-1} + (\boldsymbol{F} * \hat{\boldsymbol{\alpha}}_{i-1})_j + \boldsymbol{b})\right),$$

using 32-dimensional 1D convolution with kernel size 31.

Tacotron 2

Usually, the current state in the attention $\boldsymbol{s}_{i-1}$ is represented using the decoder cell state. However, in this model, it is beneficial for the attention to has access to the generated spectrogram.

Therefore, a separate *attention RNN* is used, which has to goal to represent the "mel spectrogram generated so far": its inputs are the decoder input (the result of the pre-net) concatenated with the attention context vector $\boldsymbol{\alpha}_i^T \boldsymbol{h}$, and the output of this attention RNN is used as $\boldsymbol{s}_{i-1}$ in the attention computation.

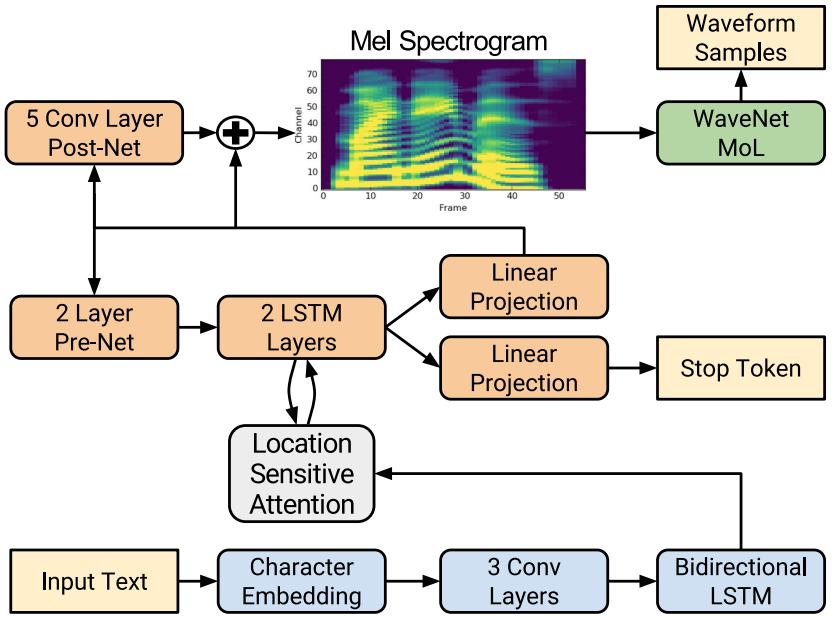


Figure 1 of "Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions", https://arxiv.org/abs/1712.05884

Tacotron 2

The decoder predicts the spectrogram one frame at a time. The predictions from the previous step are first passed through a *pre-net* composed of 2 fully-connected ReLU layers with 256 units and concatenated with attention context vector.

The decoder consists of 2 1024-dimensional LSTM cells and its output is linearly projected to the predicted frame.

The stop-token is predicted from a concatenation of decoder output and attention context vector, followed by a sigmoid activation.

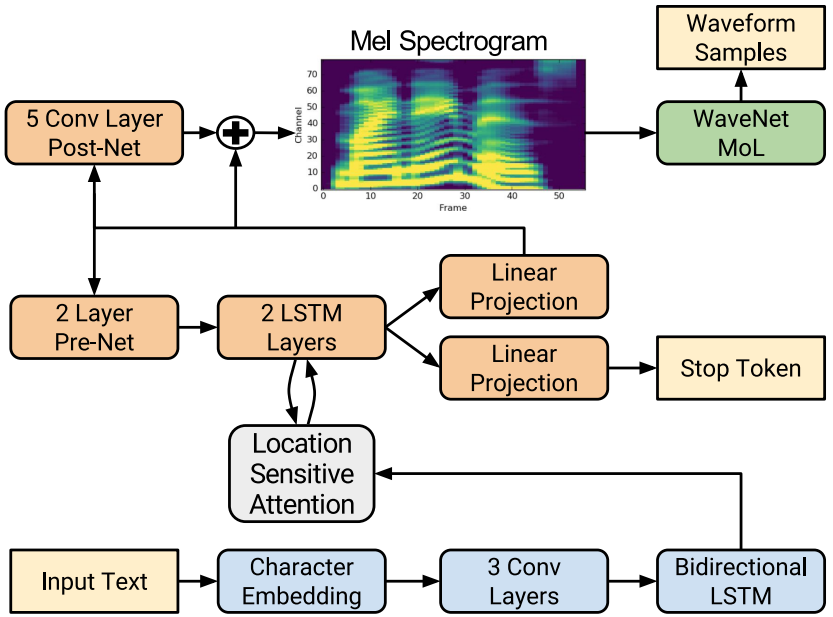


Figure 1 of "Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions", https://arxiv.org/abs/1712.05884

The predicted spectrogram frame is post-processed by a *post-net* composed of 5 convolutional layers followed by batch normalization and tanh activation (on all but the last layers), with 512 channels and kernel size 5.

The target objective is the MSE error of the spectrogram and the decoder output both before and after the post-net.

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

Table 1 of "Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions", https://arxiv.org/abs/1712.05884

Tacotron 2

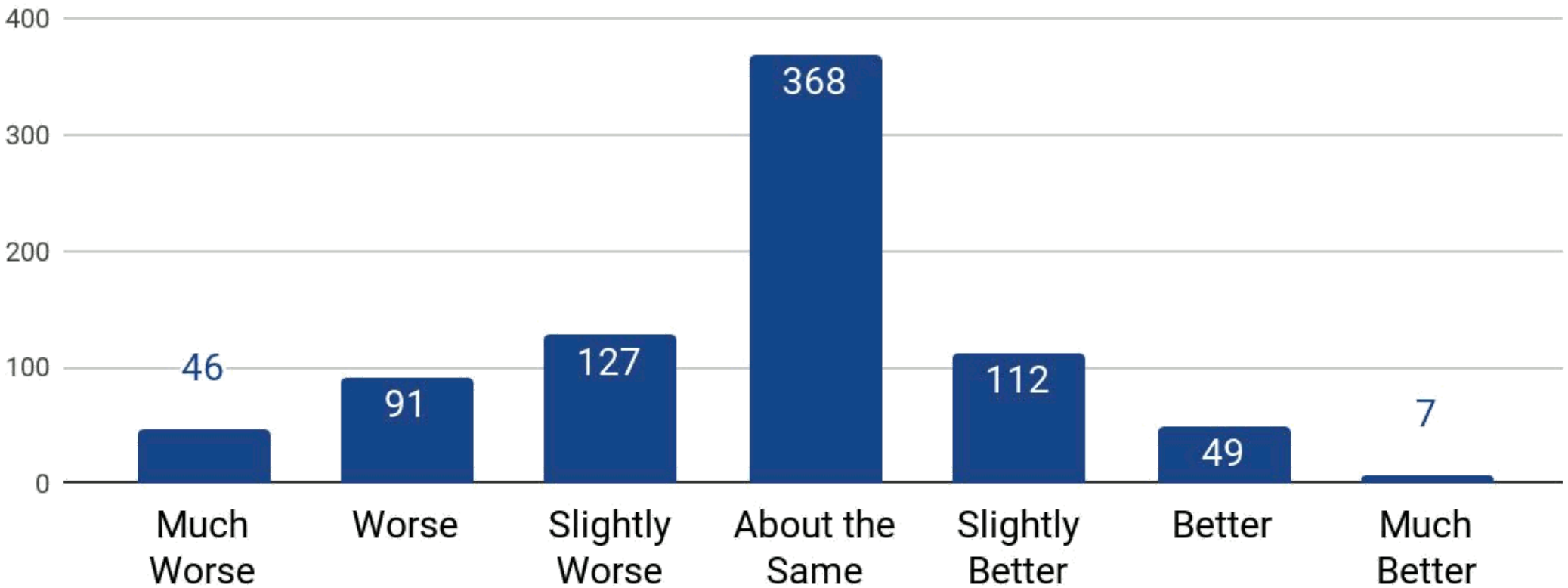


Figure 2 of "Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions", https://arxiv.org/abs/1712.05884

You can listen to samples at https://google.github.io/tacotron/publications/tacotron2/

Training	Synthesis: Predicted	Synthesis: Ground truth
Predicted	4.526 ± 0.066	4.449 ± 0.060
Ground truth	4.362 ± 0.066	4.522 ± 0.055

Table 2. Comparison of evaluated MOS for our system when WaveNet trained on predicted/ground truth mel spectrograms are made to synthesize from predicted/ground truth mel spectrograms.

Table 2 of "Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions", https://arxiv.org/abs/1712.05884

System	MOS
Tacotron 2 (Linear + G-L)	3.944 ± 0.091
Tacotron 2 (Linear + WaveNet)	4.510 ± 0.054
Tacotron 2 (Mel + WaveNet)	$\mathbf{4.526 \pm 0.066}$

Table 3. Comparison of evaluated MOS for Griffin-Lim vs. WaveNet as a vocoder, and using 1,025-dimensional linear spectrograms vs. 80-dimensional mel spectrograms as conditioning inputs to WaveNet.

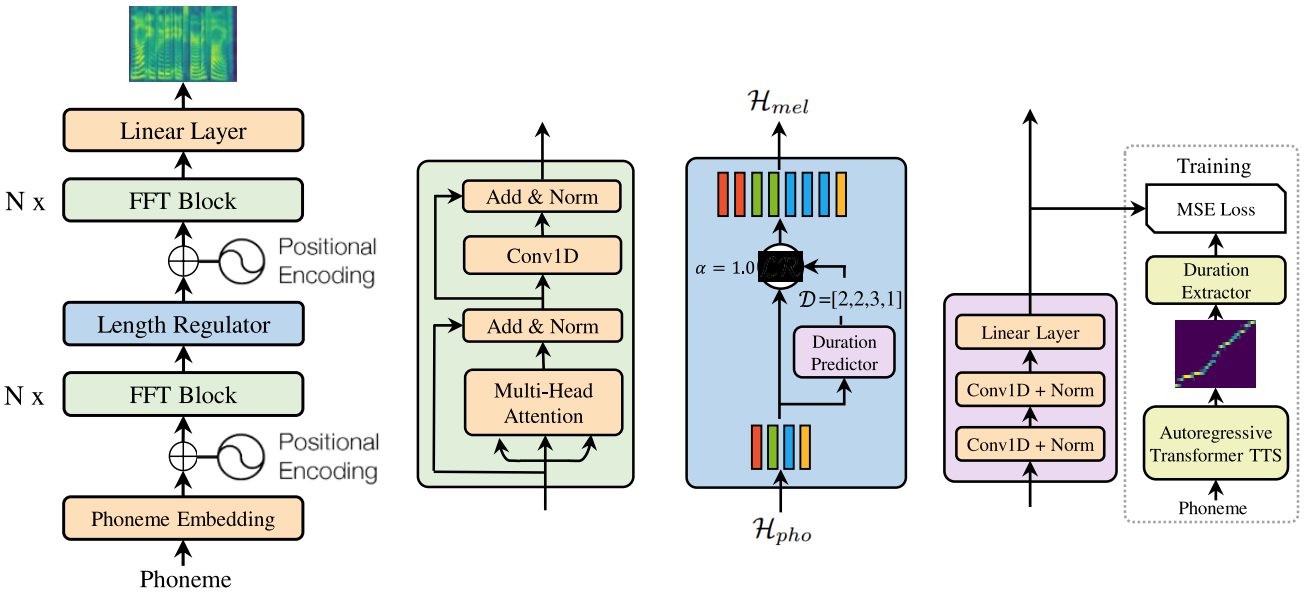
Table 3 of "Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions", https://arxiv.org/abs/1712.05884

Total layers	Num cycles	Dilation cycle size	Receptive field (samples / ms)	MOS
30	3	10	6,139 / 255.8	4.526 ± 0.066
24	4	6	505 / 21.0	4.547 ± 0.056
12	2	6	253 / 10.5	4.481 ± 0.059
30	30	1	61 / 2.5	3.930 ± 0.076

Table 4. WaveNet with various layer and receptive field sizes.

Table 4 of "Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions", https://arxiv.org/abs/1712.05884

FastSpeech



(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.

Figure 1 of "FastSpeech: Fast, Robust and Controllable Text to Speech", https://arxiv.org/abs/1905.09263

FastSpeech performs non-autoregressive decoding of mel spectrogram from the input text.

A hard character–spectrogram frame alignment is computed using some existing system, and the representation of input character is then repeated according to the alignment.

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

Table 1: The MOS with 95% confidence intervals.

Table 1 of "FastSpeech: Fast, Robust and Controllable Text to Speech", https://arxiv.org/abs/1905.09263

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.

Table 3 of "FastSpeech: Fast, Robust and Controllable Text to Speech", https://arxiv.org/abs/1905.09263

Later improved in FastPitch and FastSpeech 2 systems.

One TTS Alignment To Rule Them All

One TTS Alignment To Rule Them All

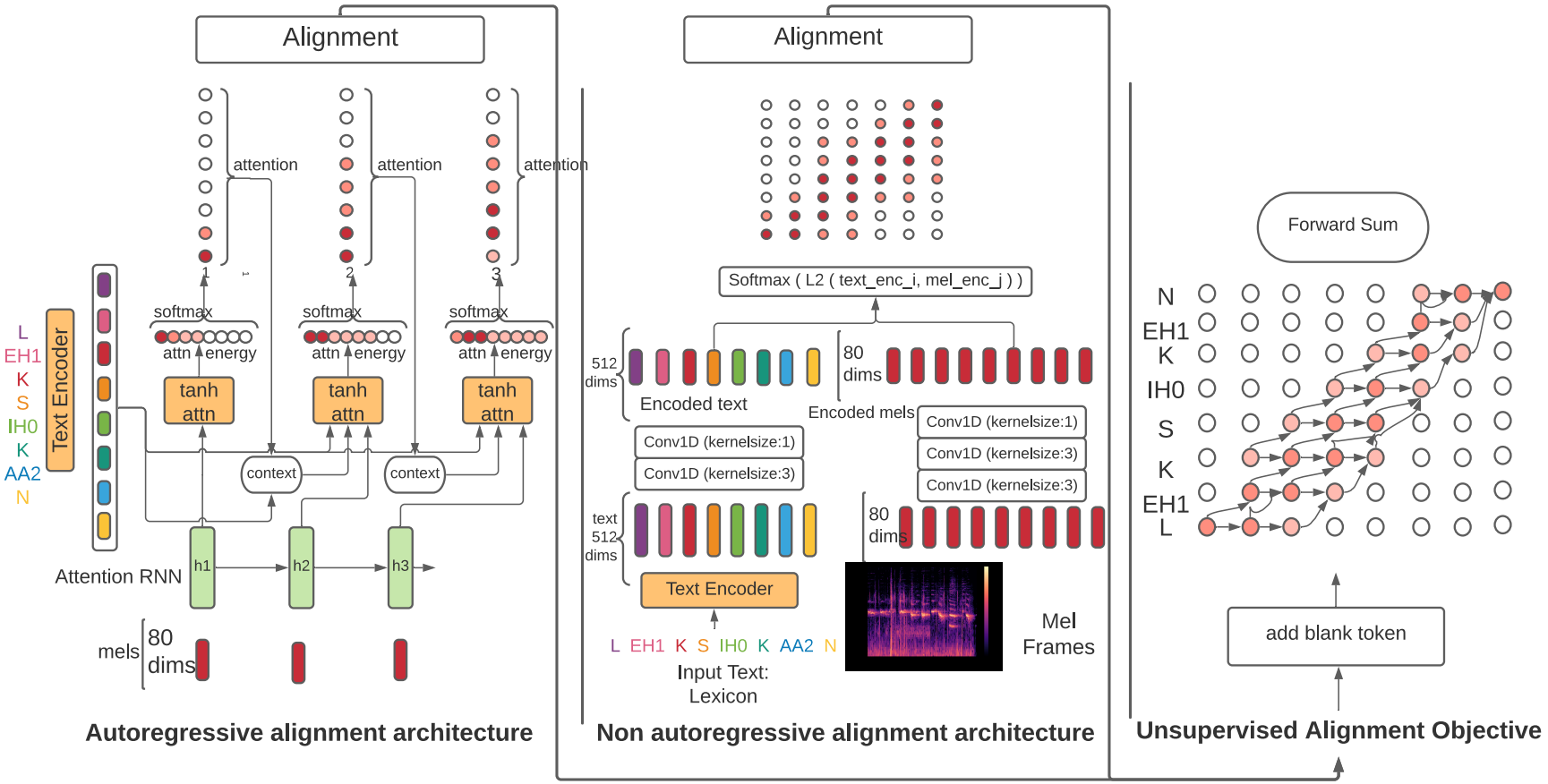


Figure 1: *Overview of our Alignment Learning Framework: autoregressive models use a sequential attention mechanism to generate alignments between text and mels. Non-autoregressive models encode text and mels using simple 1D convolutions and use pairwise L_2 distance to compute the alignments. The alignments represent the distribution $P(s_t|x_t)$ and the alignment objective (Equation 1).*

Figure 1 of "One TTS Alignment To Rule Them All", https://arxiv.org/abs/2108.10447

One TTS Alignment To Rule Them All

Let s_i denotes the i-th input text character, and $\boldsymbol{x}_t$ be the t-th mel spectrogram frame.

Assuming we have access to some alignment

$$P(s_i|\boldsymbol{x}_t;\boldsymbol{\theta}).$$

We can train the alignment to maximize the probability of all monotonic alignments between the mel spectrogram and the input text:

$$\mathcal{L}(\boldsymbol{\theta};\boldsymbol{X},\boldsymbol{s}) \stackrel{\text{def}}{=} \sum_{\substack{\boldsymbol{c}\text{ any valid}\\ \text{monotonic alignment}}} \prod_{t=1}^{T} P(c_t|\boldsymbol{x}_t;\boldsymbol{\theta}).$$

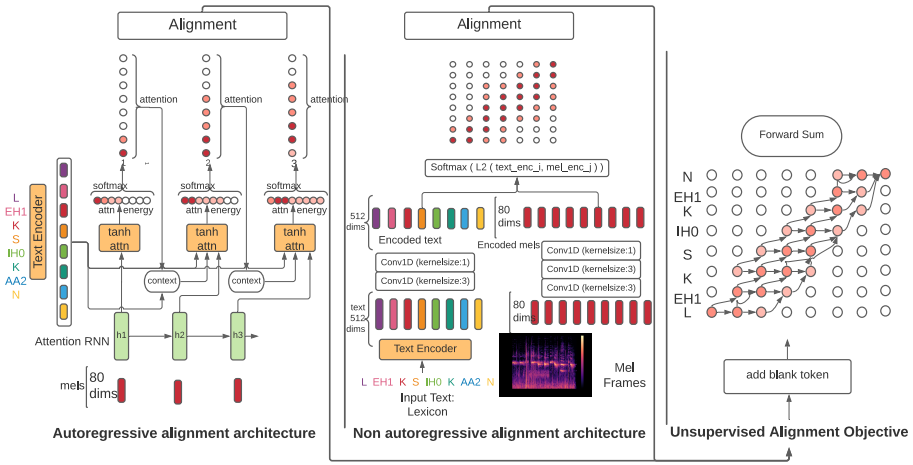
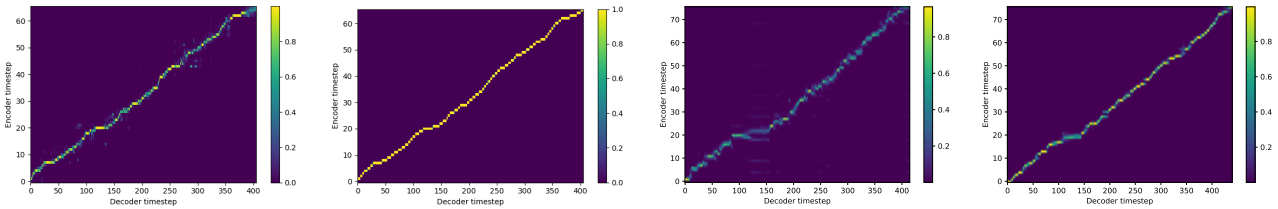


Figure 1: *Overview of our Alignment Learning Framework: autoregressive models use a sequential attention mechanism to generate alignments between text and mels. Non-autoregressive models encode text and mels using simple 1D convolutions and use pairwise L_2 distance to compute the alignments. The alignments represent the distribution $P(s_t|x_t)$ and the alignment objective (Equation 1).*

Figure 1 of "One TTS Alignment To Rule Them All", https://arxiv.org/abs/2108.10447

Such a loss can be computed by exploiting the CTC loss (given that efficient implementations for it already exist).

One TTS Alignment To Rule Them All



(a) *Flowtron* (b) *Flowtron with* $\mathcal{L}_{align}$ (c) *Tacotron 2* (d) *Tacotron 2 with* $\mathcal{L}_{align}$

Figure 3: *Converged soft alignments for Flowtron, Tacotron2. Alignment framework provides sharper and more connected alignments.*

fident and continuous alignments, and by extension, continuous speech without repeating or missing words.

Table 1: *Pairwise preference scores judged by human raters, shown with 95% confidence intervals. Scores above 0.5 indicate models trained with* $\mathcal{L}_{align}$ *were preferred by majority of raters.*

Figure 3 of "One TTS Alignment To Rule Them All", https://arxiv.org/abs/2108.10447

Table 1: *Pairwise preference scores judged by human raters, shown with 95% confidence intervals. Scores above 0.5 indicate models trained with* $\mathcal{L}_{align}$ *were preferred by majority of raters.*

Model	Alignment Framework vs Baseline
Tacotron 2	0.556 ± 0.068
Flowtron ($\sigma = .5$)	0.635 ± 0.065
RAD-TTS ($\sigma = .5$)	0.639 ± 0.066
FastPitch	0.565 ± 0.068
FastSpeech2	0.521 ± 0.067

Table 1 of "One TTS Alignment To Rule Them All", https://arxiv.org/abs/2108.10447

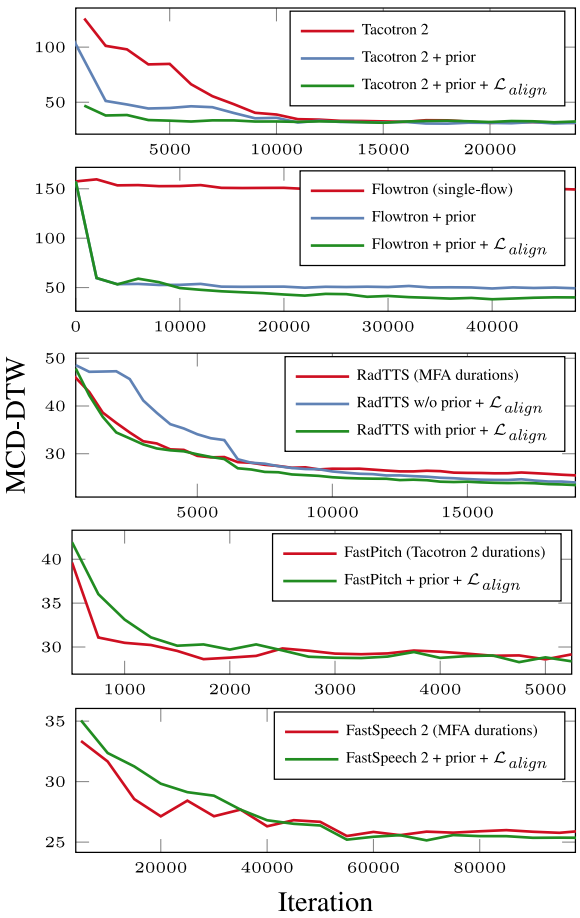


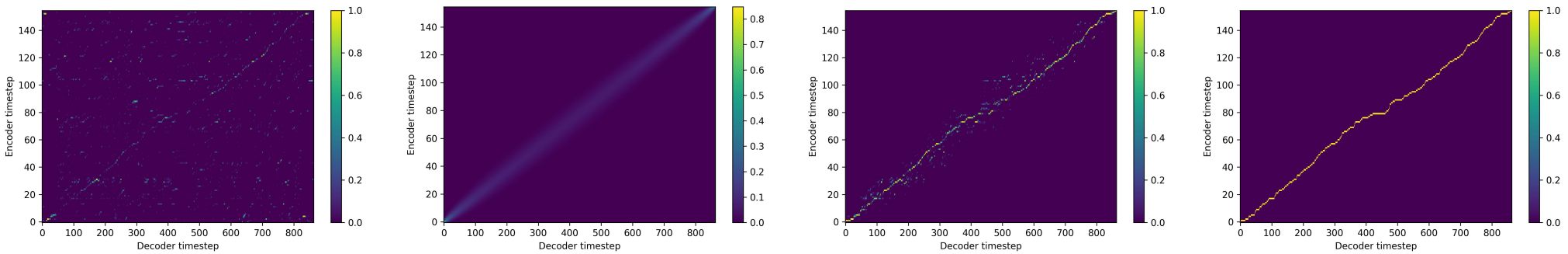
Figure 2: *Convergence rate improvements in TTS models with the alignment learning framework*

Figure 2 of "One TTS Alignment To Rule Them All", https://arxiv.org/abs/2108.10447

Alignment Prior

For faster alignment convergence (which makes the full model training faster), during training we might use a static 2D prior that is wider near the center and narrower near the corners.

The complete alignment is then the normalized product of the computed alignment and the alignment prior.



(a) Soft alignment $\mathcal{A}_{soft}$ (b) Beta-binomial Prior (c) $\mathcal{A}_{soft}$ w/ Prior (d) Hard alignment $\mathcal{A}_{hard}$

Figure 2. Visualizations of the alignment attention matrices $\mathcal{A}$. The vertical axis represents text tokens from bottom to top. The horizontal axis represents mel-frames from left to right. Fig. 2a shows the baseline soft attention map. Fig. 2c combines Fig. 2a with Fig. 2b to penalize alignments straying too far from the diagonal. Fig. 2d is the most likely monotonic alignment extracted from Fig. 2c with Viterbi.

Figure 2 of "RAD-TTS: Parallel Flow-Based TTS with Robust Alignment Learning and Diverse Synthesis", https://openreview.net/forum?id=0NQwnnwAORi