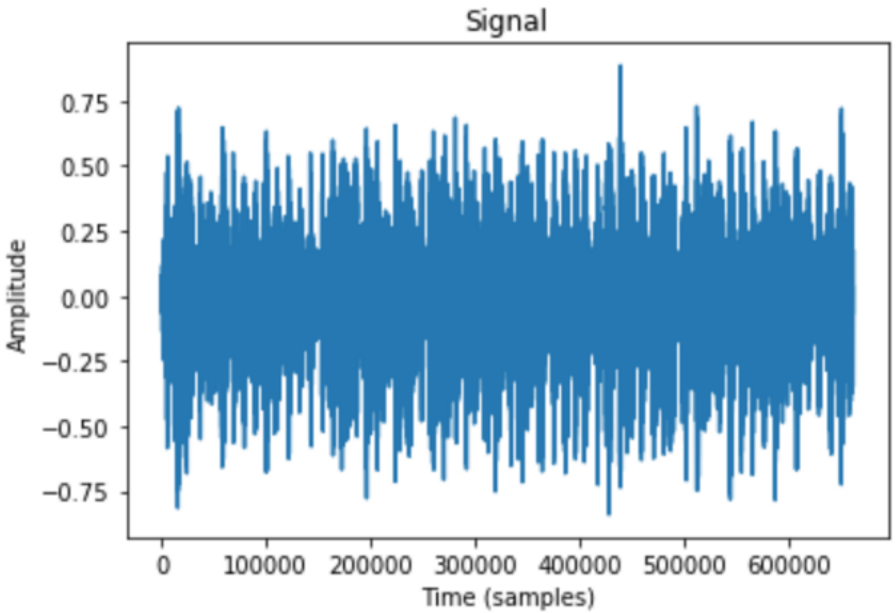
FastSpeech 2: Fast and High-Quality End-to-EndText to Speech

Vladislav Filimonov

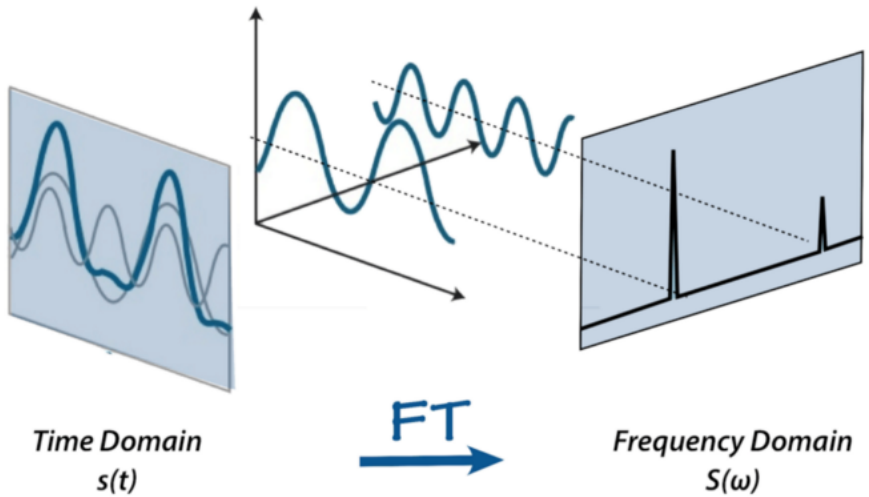
CMC MSU

September 29, 2020

Waveforms and Spectrum[1]



(a) Waveform of $\sim$ 14.5 seconds



(b) Concept of Fourier Transform

[1] Quick intro to spectrogram

Short Time Fourier Transform

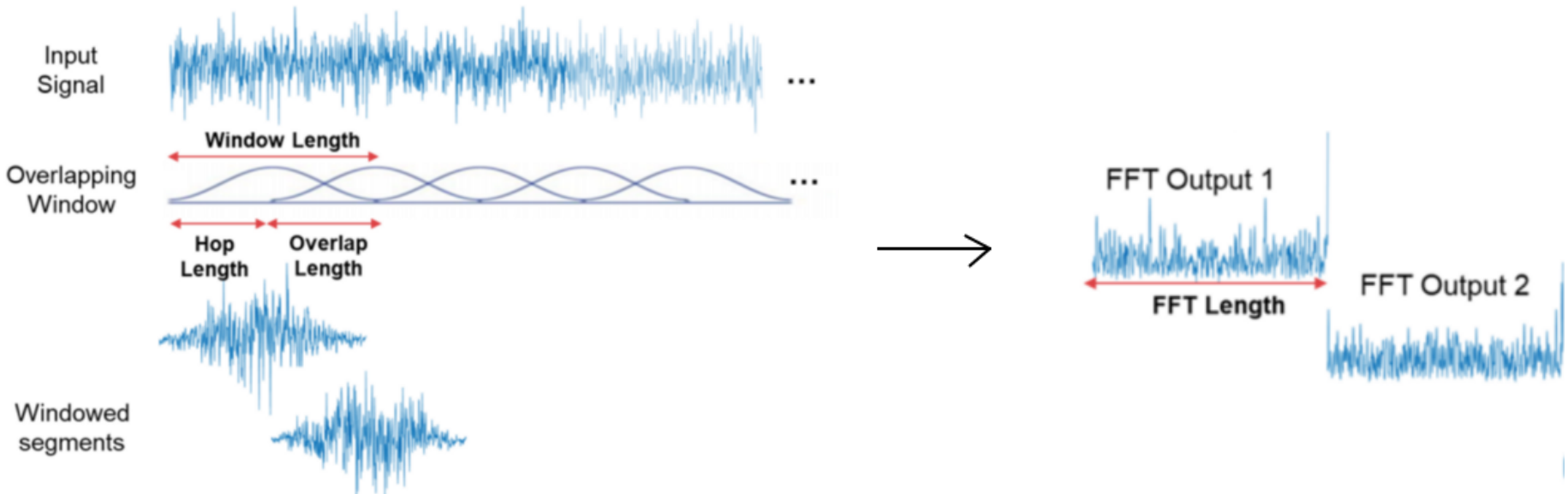
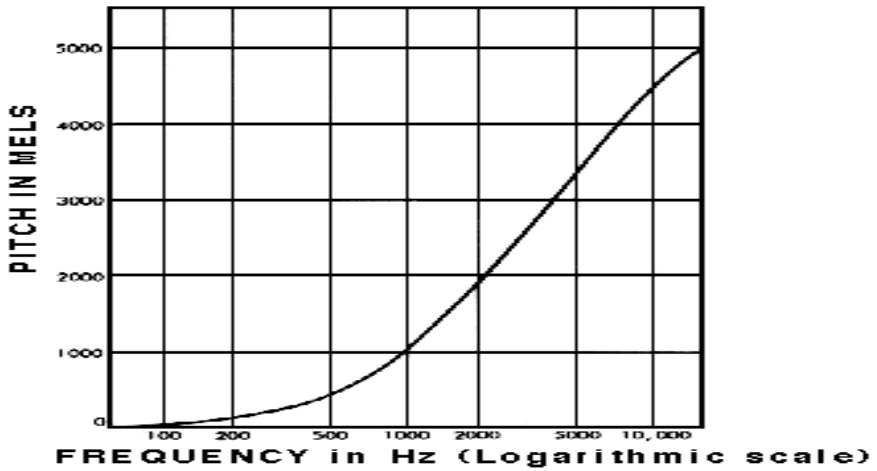
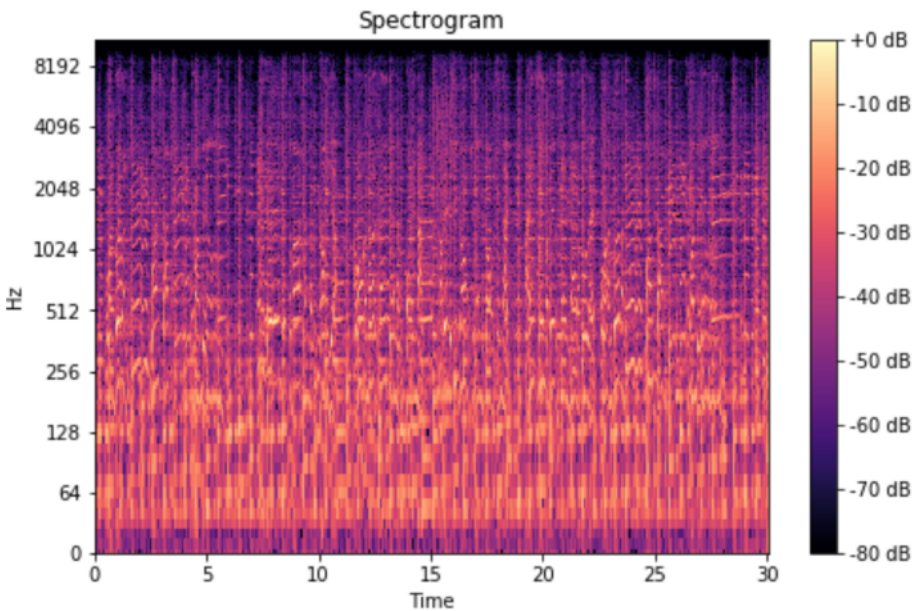


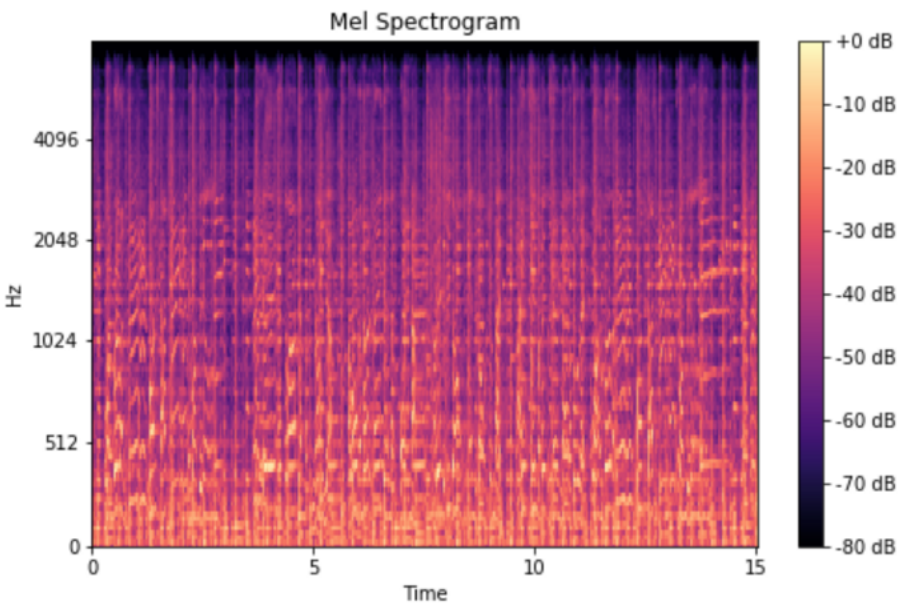
Figure: Concept of STFT[2]

[2] STFT

Spectrogram and Mel scale[3]







[3] Mel scale

Common Text to Speech System

Common TTS systems consists of 2 parts:

1. Synthesizer / acoustic model: given text and audio attributes (e.g. speaker, duration, pitch, etc.) generates mel spectrogram;
2. Vocoder: given mel spectrogram generates audio signal.

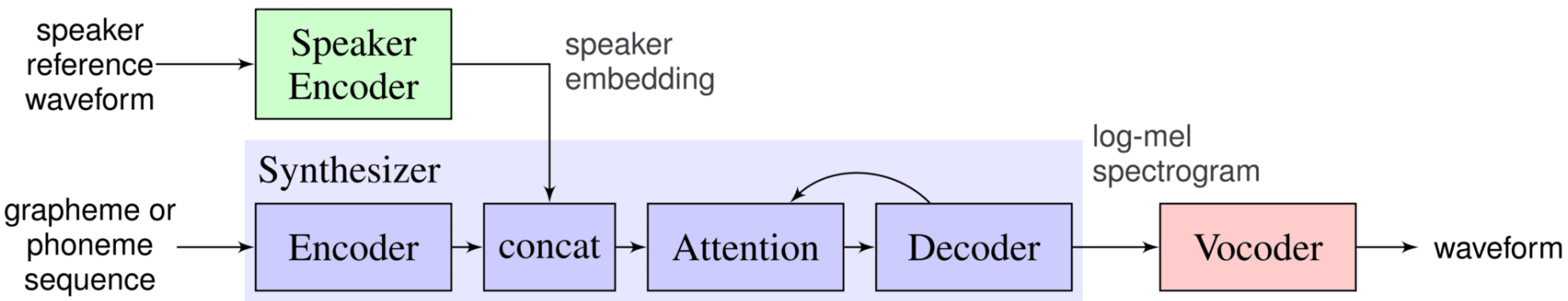
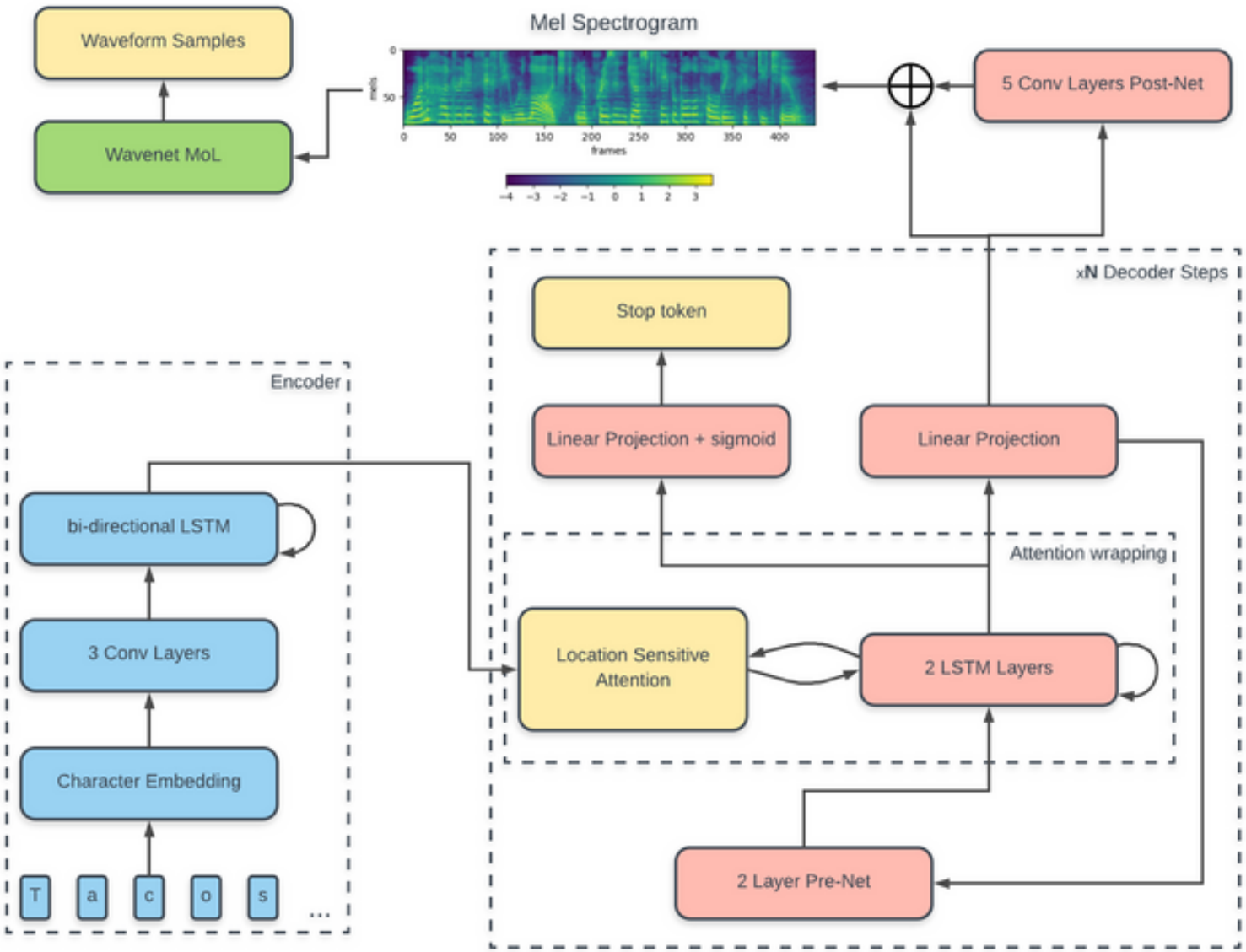


Figure: Common autoregressive TTS system [4]

[4] Image Source

Tacotron 2 [1]


Waveform Samples
Wavenet MoL
Mel Spectrogram
mels
frames
5 Conv Layers Post-Net
xN Decoder Steps
Stop token
Linear Projection + sigmoid
Linear Projection
Encoder
bi-directional LSTM
3 Conv Layers
Character Embedding
T
a
c
o
s
...
Attention wrapping
Location Sensitive Attention
2 LSTM Layers
2 Layer Pre-Net

Multi-Head Attention

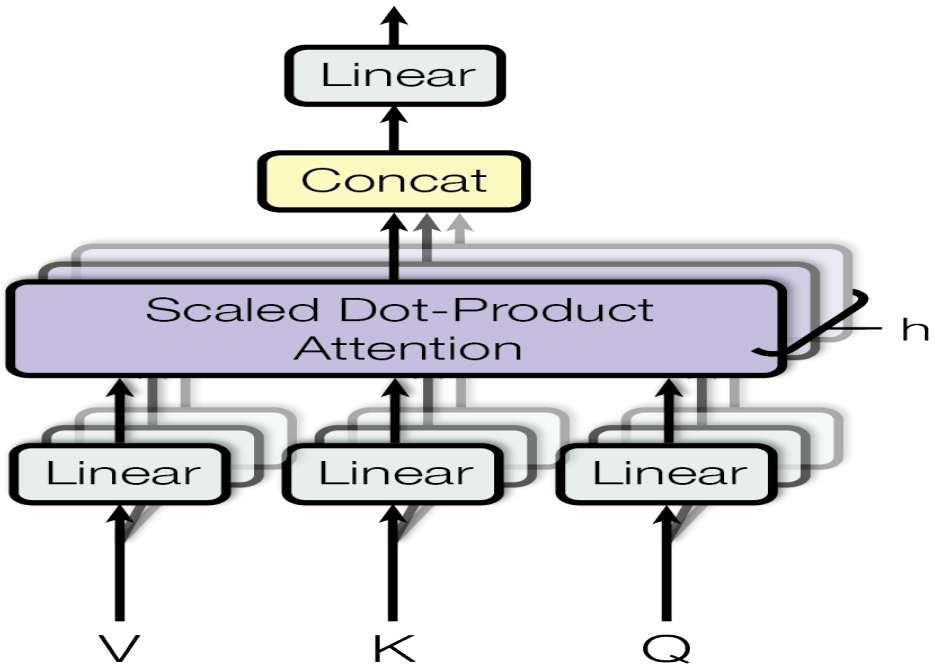


Figure: Multi-head Attention [2]

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

Multi-Head Attention

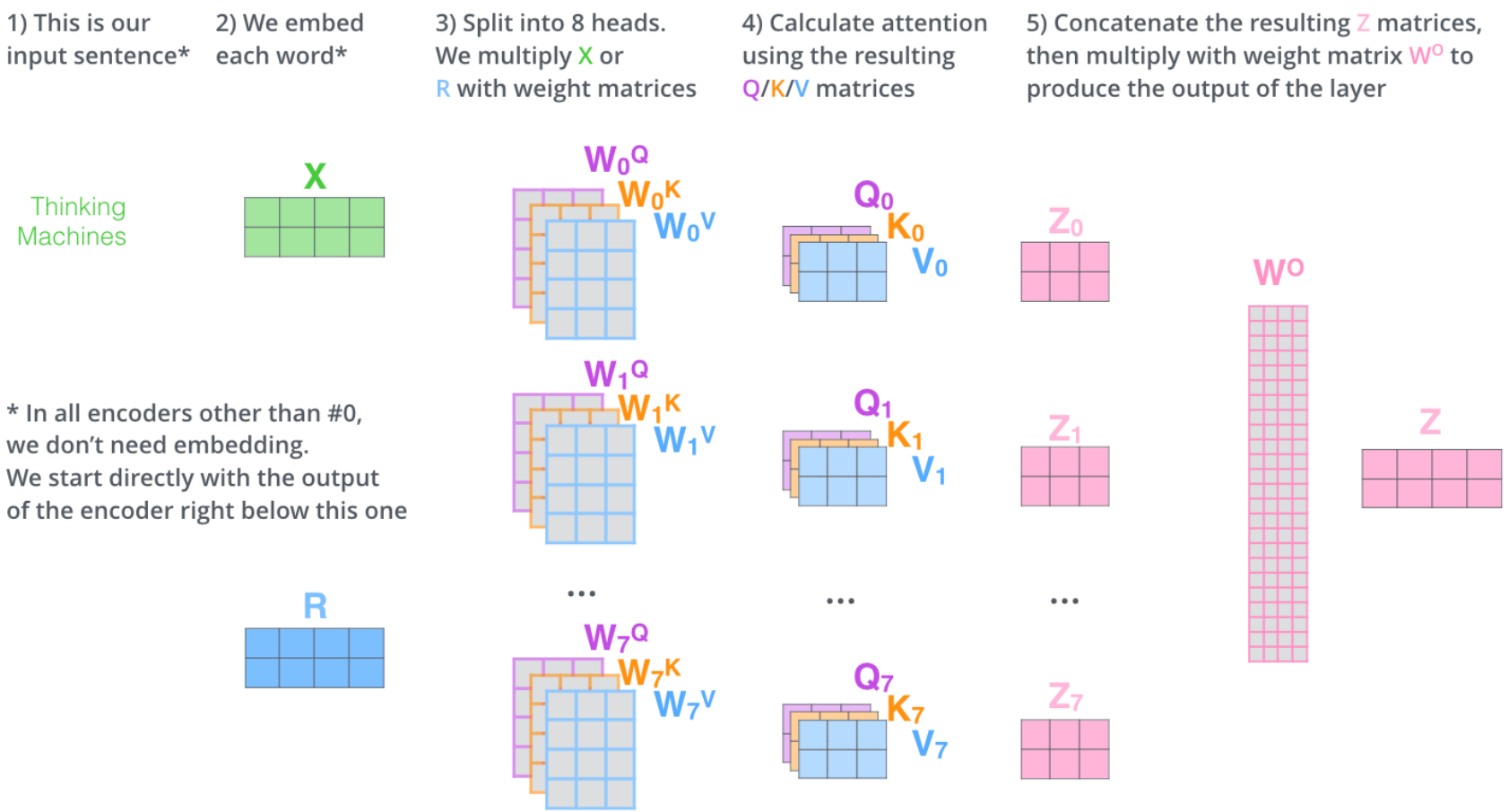


Figure: Multi-head attention [5]

[5] Image source and good visualization of transformer

Transformer TTS[2]: motivation and problems

Motivation: Lets use Transformer as encoder and decoder for TTS System, since transformers are good and fast! Problems:

- Still autoregressive model $\rightarrow$ slow inference speed.
- Bigger model than Tacotron based models $\rightarrow$ training speed up is not so great (~ 1.5).
- Unexplored ways to condition on some attributes (e.g. speaker, pitch, etc).

Transformer TTS[3]: relation with Transformer for NMT

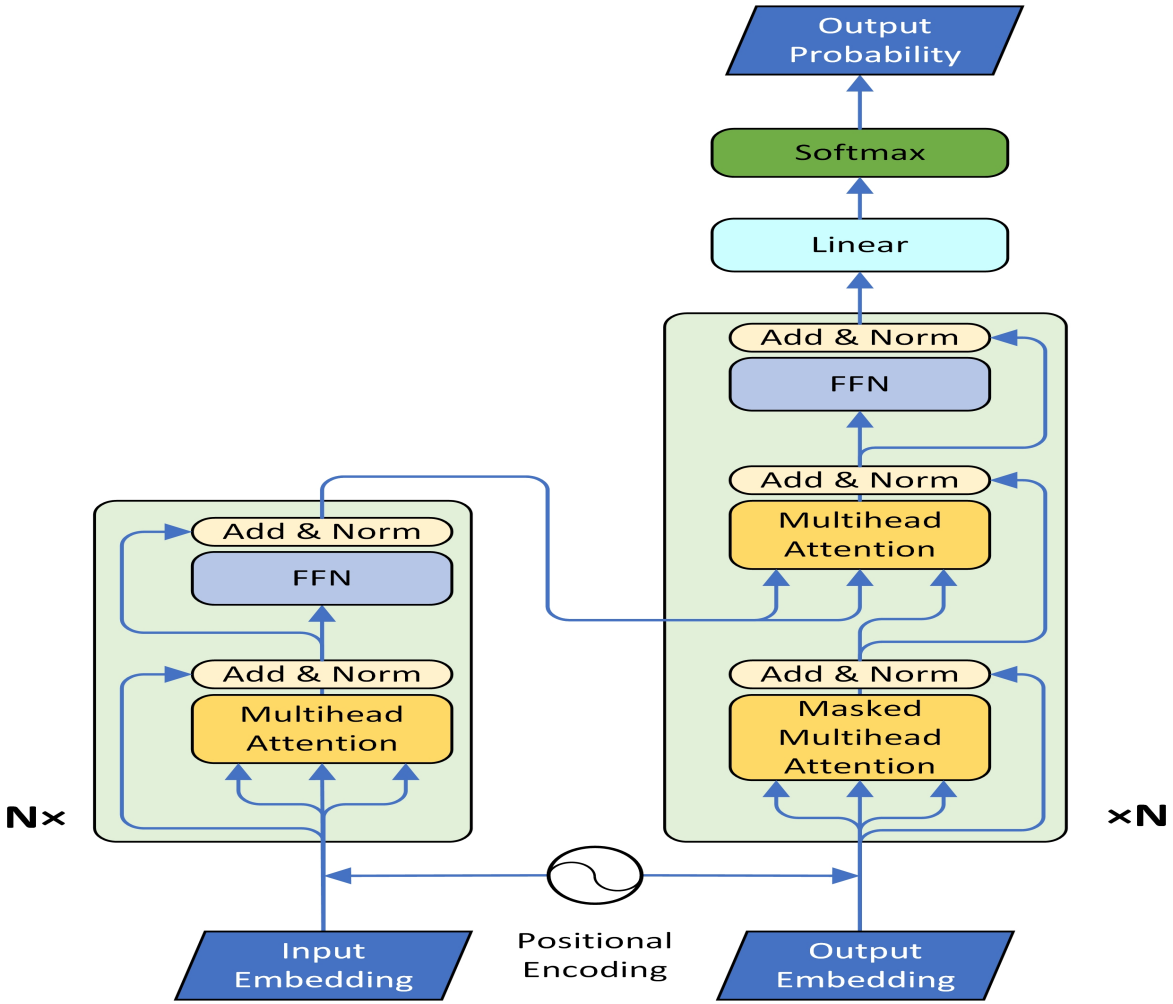


Figure: Transformer for NMT [6]

[6] Image source

Transformer TTS[2]: relation with Transformer for NMT

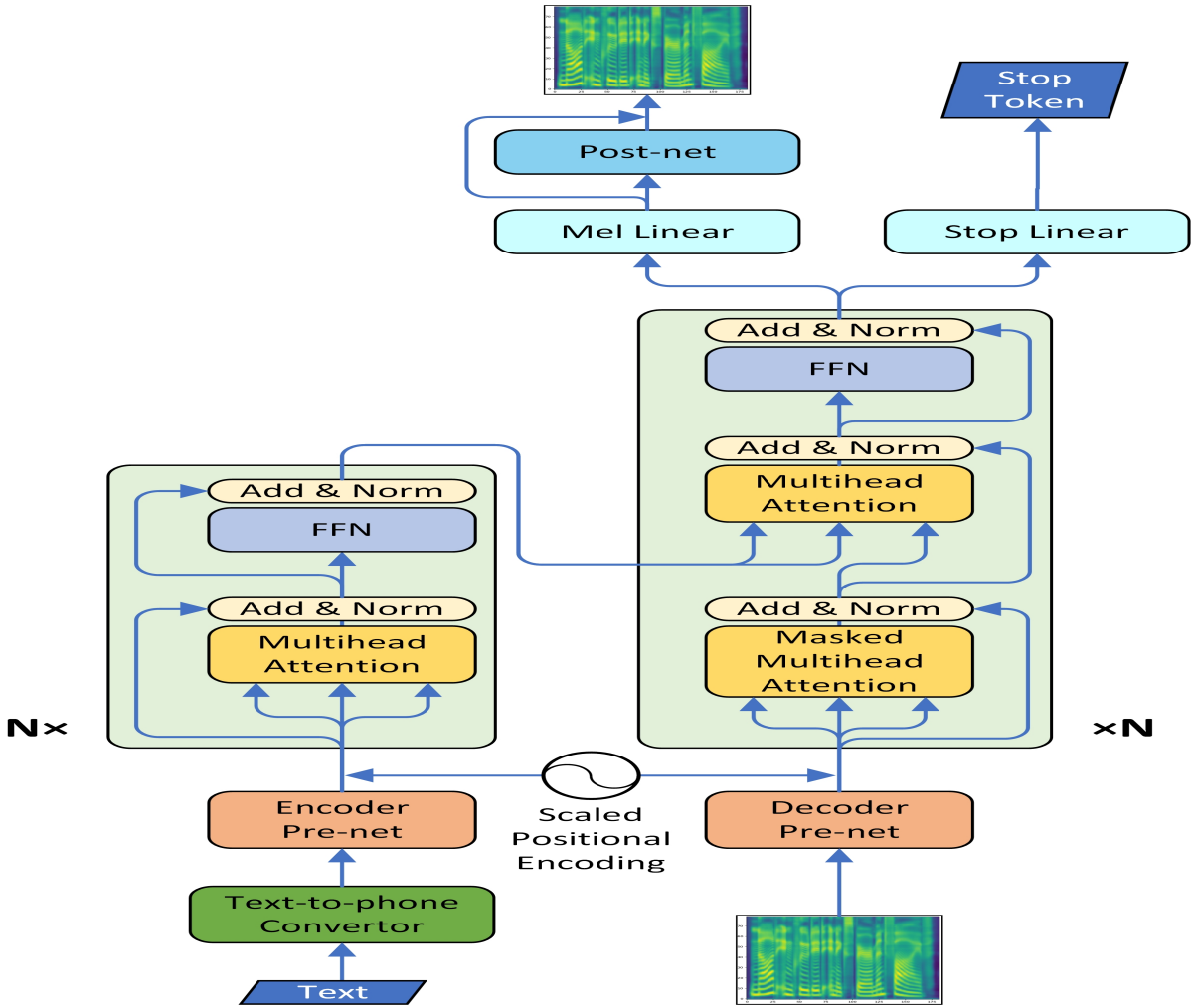


Figure: Transformer TTS [7]

[7] Image source

Transformer TTS[2]: architecture

3.1 Text-to-Phoneme Converter

English pronunciation has certain regularities, for example, there are two kinds of syllables in English: open and closed. The letter "a" is often pronounced as /eɪ/ when it's in an open syllable, while it is pronounced as /æ/ or /aː/ in closed syllables. We can rely on the neural network to learn such a regularity in the training process. However, it is difficult to learn all the regularities when, which is often the case, the training data is not sufficient enough, and some exceptions have too few occurrences for neural networks to learn. So we make a rule system and implement it as a text-to-phoneme converter, which can cover the vast majority of cases.

(a) Citation from [2]

(b) Citation from [2]

Scaled positional encoding:

$$PE(pos, 2i) = \sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{2}$$

$$PE(pos, 2i+1) = \cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{3}$$

$$x_i = \text{prenet}(\text{phoneme}_i) + \alpha PE(i) \tag{4}$$

Transformer TTS[2]: architecture

Encoder Pre-net: 3 layer CNN with ReLU and BN (share information about neighbours phoneme).
Decoder Pre-net: 2 layer Feed Forward NN with ReLU.
Decoder Post-net: 5 layer CNN with ReLU.

Transformer TTS[2]: results

Training Tacotron2 and Transformer TTS on 25 hour single speaker internal dataset.
WaveNet vocoder separately trained on same dataset.
38 fixed samples were evaluated by 20 listeners for MOS, and pairs of these samples for CMOS.

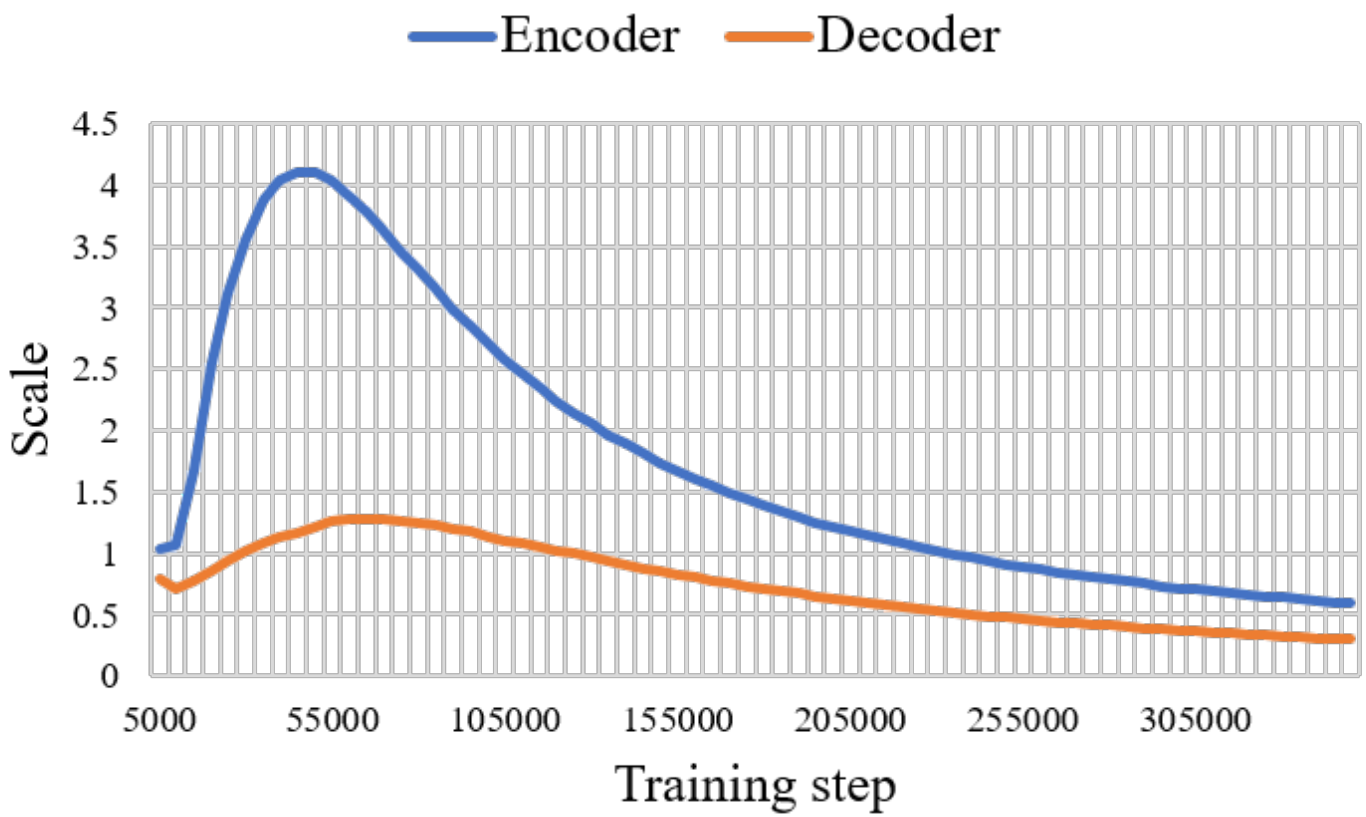
System	MOS	CMOS
Tacotron2	4.39 ± 0.05	0
Our Model	4.39 ± 0.05	**0.048**
Ground Truth	4.44 ± 0.05	-

Table: MOS comparison among Transoformer TTS, our Tacotron2 and recordings.

Time consume in a **single training step for Transformer TTS is $\sim$0.4s, which is 4.25 times faster than that of Tacotron2 ($\sim$1.7s)** with equal batch size (16 samples per batch). Since the parameter quantity of Transformer TTS is almost twice than Tacotron2, **it still takes $\sim$3 days to converge comparing to $\sim$4.5 days of that for Tacotron2**.

Transformer TTS[2]: results

PE Type	MOS
Original	4.37 ± 0.05
Scaled	**4.40** ± 0.05
Ground Truth	4.41 ± 0.04

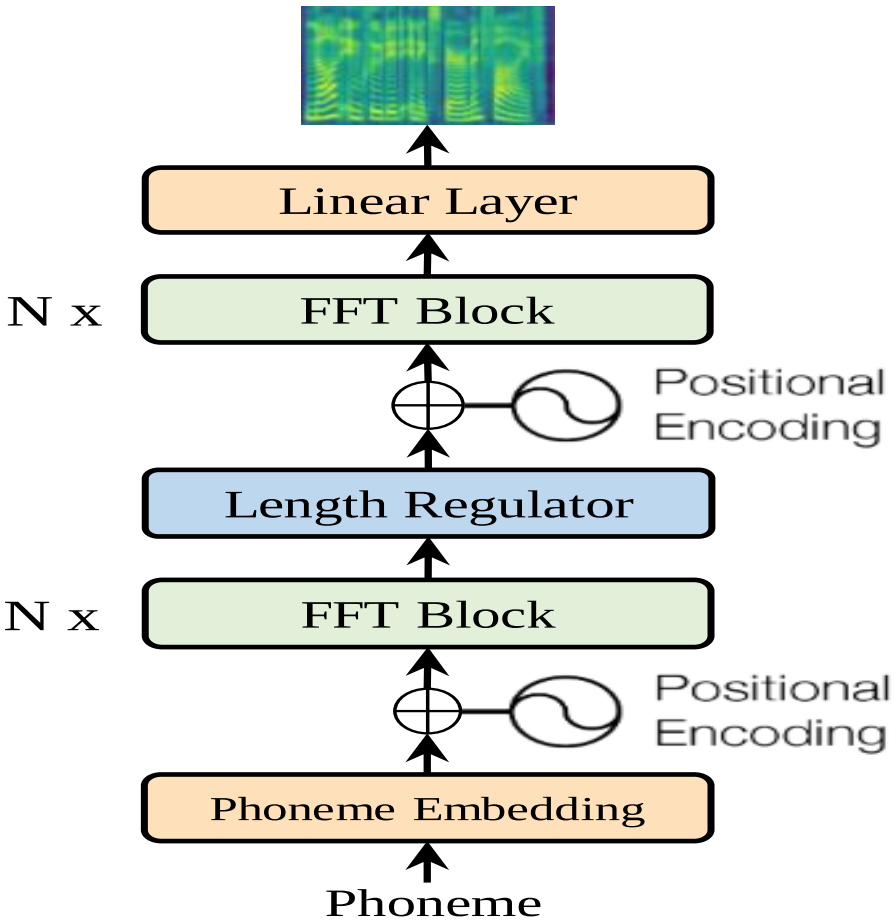


FastSpeech [4]: motivation and problems

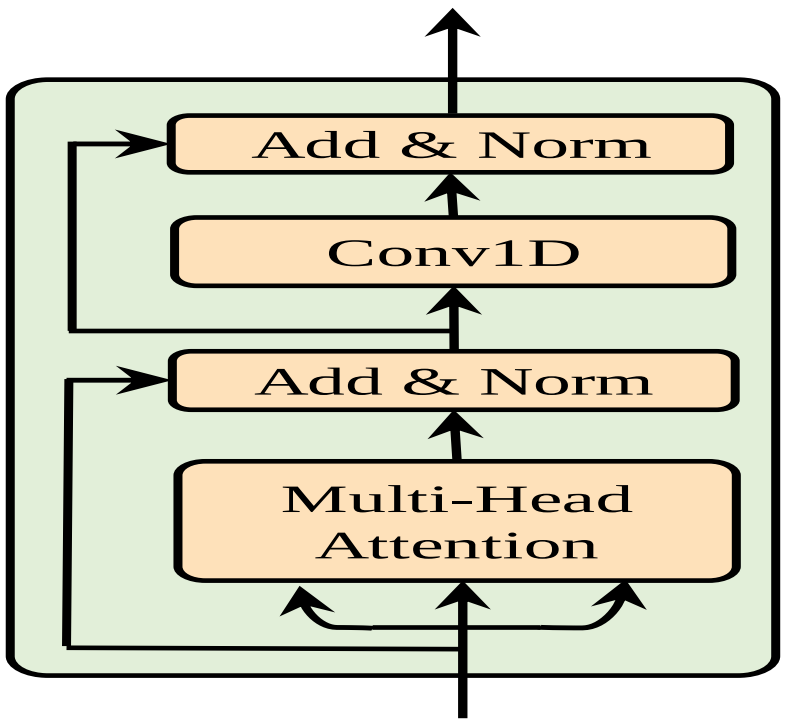
Motivation: Transformers [3] are good and fast, but autoregressive models are slow at inference, need to create non autoregressive model. Problems:

- Transformer block doesn't change the spatial dimension of inputs (dimension of text $<$ dimension of mel spectrograms). Need to find a way to correctly upsample input inside model.
- Unexplored ways to condition on some attributes (e.g. speaker, pitch, etc).

FastSpeech [4]: architecture

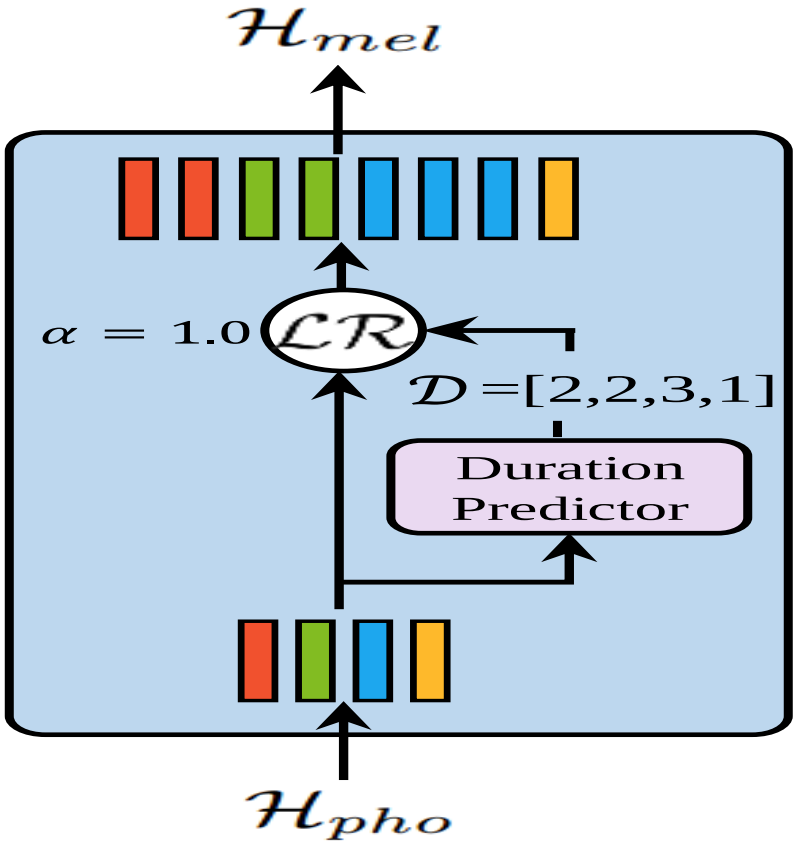


(a) FastSpeech

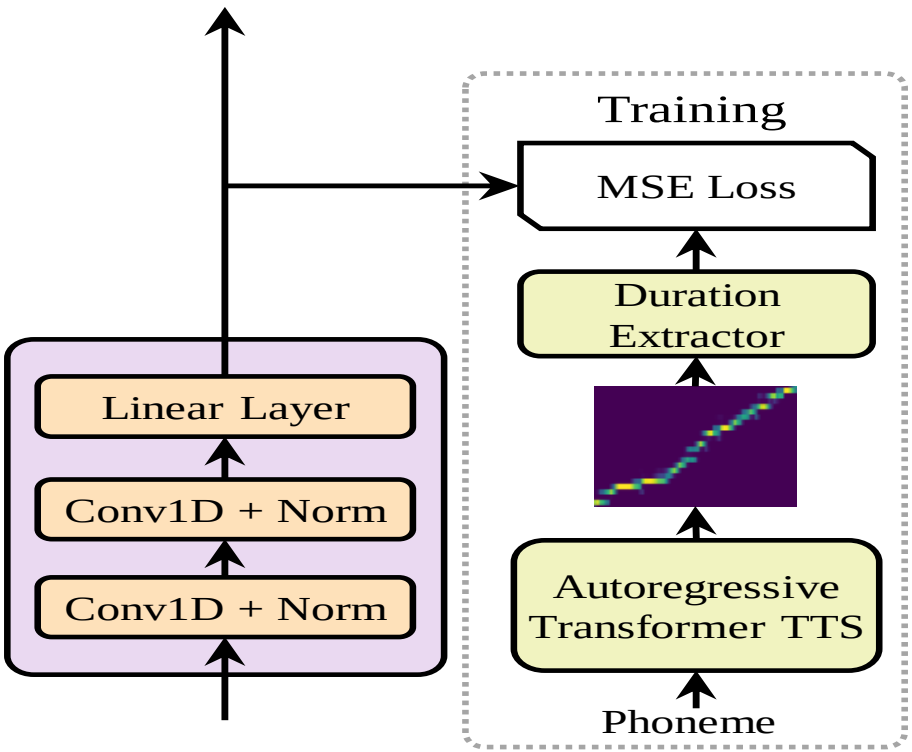


(b) FFT block

FastSpeech [4]: architecture

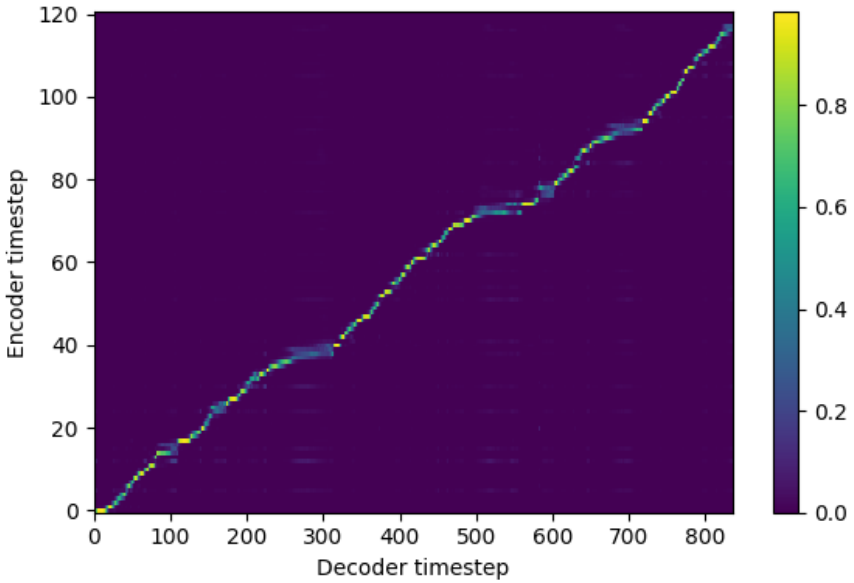


(a) Length regulator

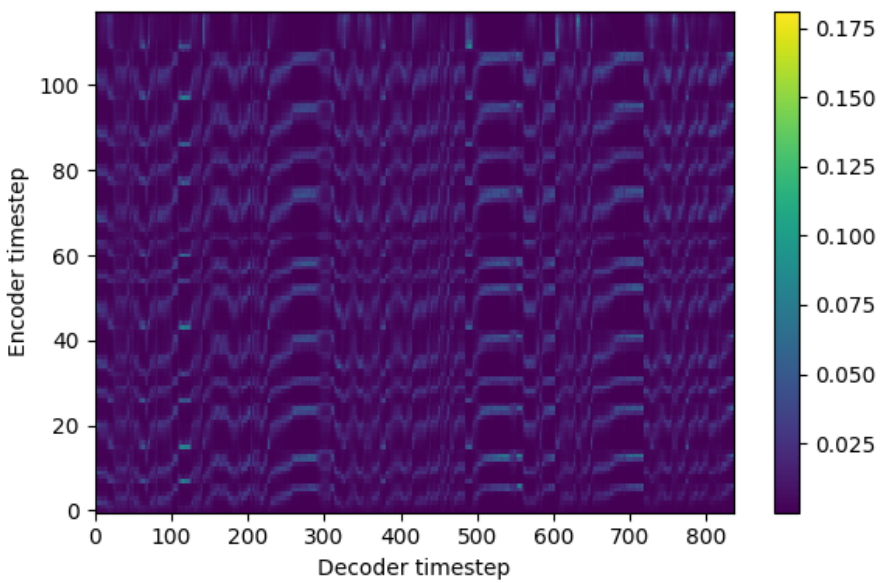


(b) Duration predictor

FastSpeech [4]: GT duration extraction



(a) Diagonal attention



(b) Non Diagonal attention

Calculate focus rate $F = \frac{1}{S}\sum_{s=1}^{S} \max_{1\leq t\leq T} a_{s,t}$, to find the most diagonal attention $A \in R^{n\times S}$.
Extract durations from it as $\mathcal{D} = [d_1, d_2, ..., d_n]$ according to the duration extractor $d_i = \sum_{s=1}^{S}[\arg\max_t a_{s,t} = i]$.

FastSpeech [4]: results

Training all models on LJSpeech [5] dataset (24 hours of single speaker). 300 samples from test for each model were listened by 20 listeners for MOS.

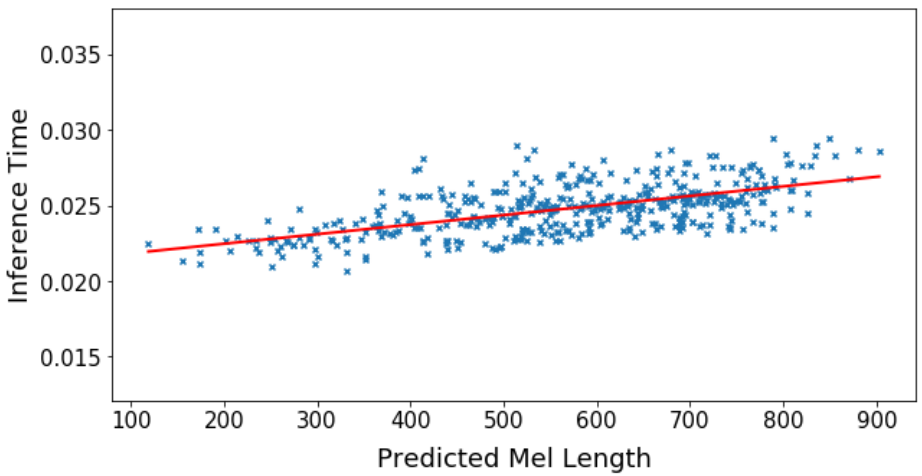
Method	MOS
GT	4.41 ± 0.08
GT (Mel + WaveGlow[6])	4.00 ± 0.09
Tacotron 2 [1] (Mel + WaveGlow [6])	3.86 ± 0.09
Transformer TTS [2] (Mel + WaveGlow[6])	3.88 ± 0.09
FastSpeech [4] (Mel + WaveGlow[6])	3.84 ± 0.08

Table: The MOS with 95% confidence intervals.

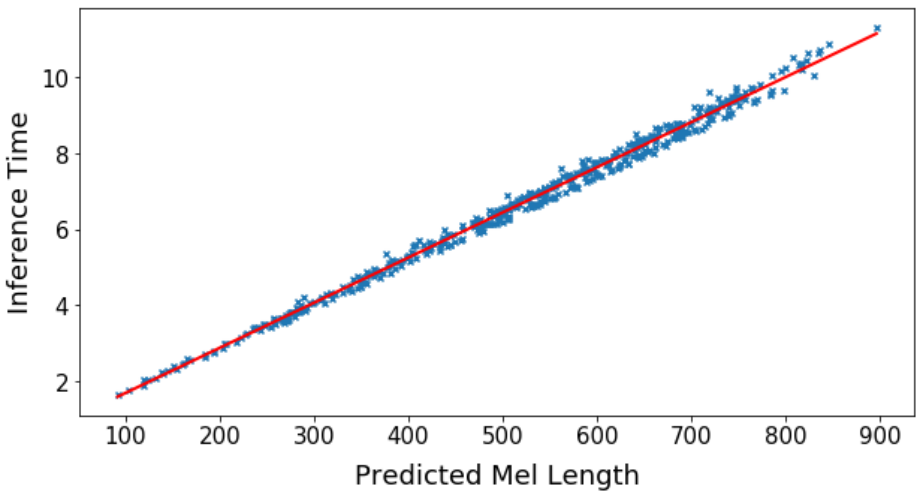
FastSpeech [4]: results

The comparison of inference latency with 95% confidence intervals. The evaluation is conducted on a server with 12 Intel Xeon CPU, 256GB memory, 1 NVIDIA V100 GPU and batch size of 1.

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



(a) FastSpeech [4]

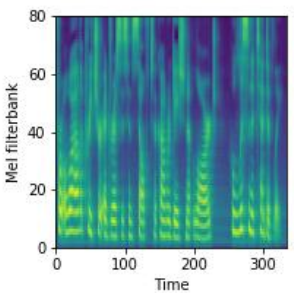


(b) Transformer TTS [2]

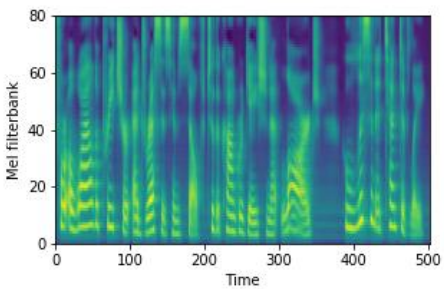
FastSpeech [4]: results

The comparison of robustness between FastSpeech and others on the 50 hard sentences.

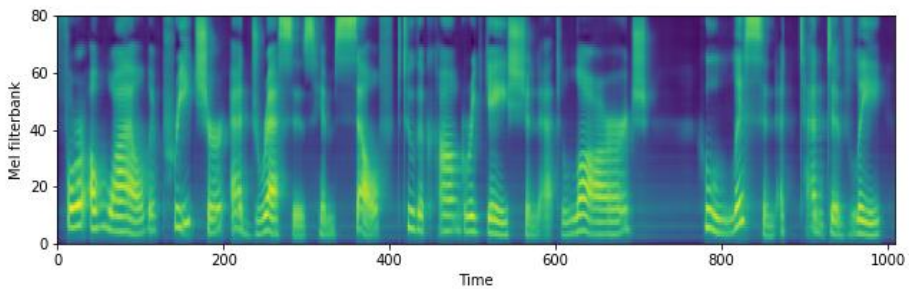
Method	Repeats	Skips	Error Sentences	Error Rate
Tacotron 2	4	11	12	24%
Transformer TTS	7	15	17	34%
FastSpeech	0	0	0	0%



(a) 1.5x voice speed



(b) 1.0x voice speed



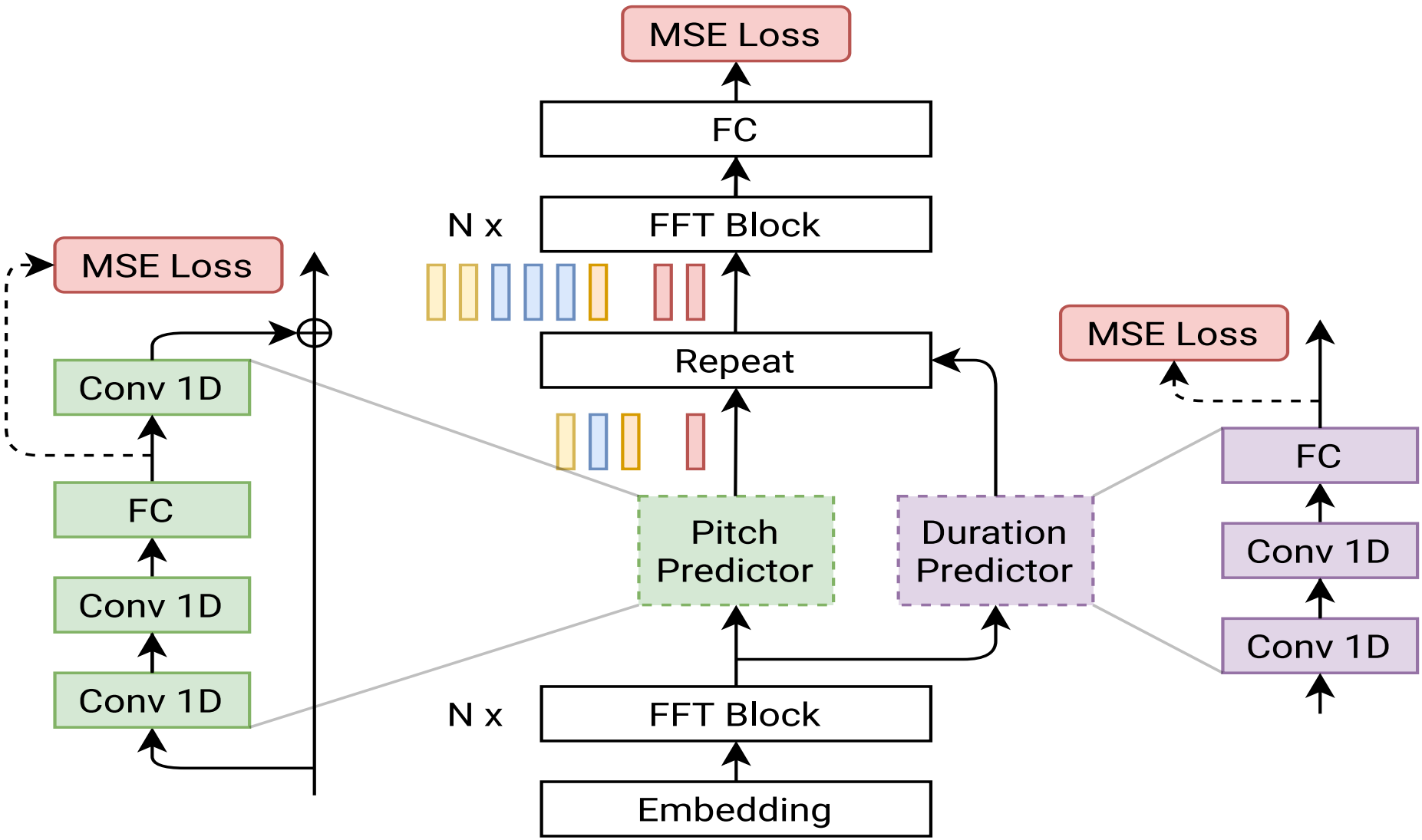
(c) 0.5x voice speed

FastPitch [7]: motivation and problems

Motivation: Non autoregressive transformer models [4] are good and fast at both training inference, lets improve them via better durations and pitch conditioning. Problems:

- Extracting durations from autoregressive TTS requires training ($\rightarrow$ big single speaker dataset) and speaker dependent.
- Transformer TTS [2] has a lot of attention matrix and using the focus rate heuristics could be not the best choice.

FastPitch [7]: architecture



FastPitch [7]: architecture

$x = (x_1, \ldots, x_t)$ - input lexical units, and $y = (y_1, \ldots, y_T)$ - target mel spectrogram.

$$h = \text{FFTransformer}(x). \tag{5}$$

$$\hat{d} = \text{DurationPredictor}(h); \;\; \hat{p} = \text{PitchPredictor}(h) \tag{6}$$

$$\begin{aligned} g &= h + \text{PitchEmbedding}(p) \\ \hat{y} &= \text{FFTransformer}([\underbrace{g_1, \ldots, g_1}_{d_1}, \underbrace{g_2, \ldots, g_2}_{d_2}, \ldots \underbrace{g_t, \ldots, g_t}_{d_t}]). \end{aligned} \tag{7}$$

The model optimizes mean-squared error (MSE) between the predicted and ground-truth modalities

$$\mathcal{L} = \|\hat{y} - y\|_2^2 + \alpha\|\hat{p} - p\|_2^2 + \gamma\|\hat{d} - d\|_2^2 \tag{8}$$

FastPitch [7]: GT durations

Durations of input symbols are estimated with a Tacotron2[1] model trained on LJSpeech-1.1 [5].
Let $A \in R^{t \times T}$ be the final Tacotron2 attention matrix.
The duration of the ith input symbol is:

$$d_i = \sum_{c=1}^{T} [\text{argmax}_r A_{r,c} = i]. \quad (9)$$

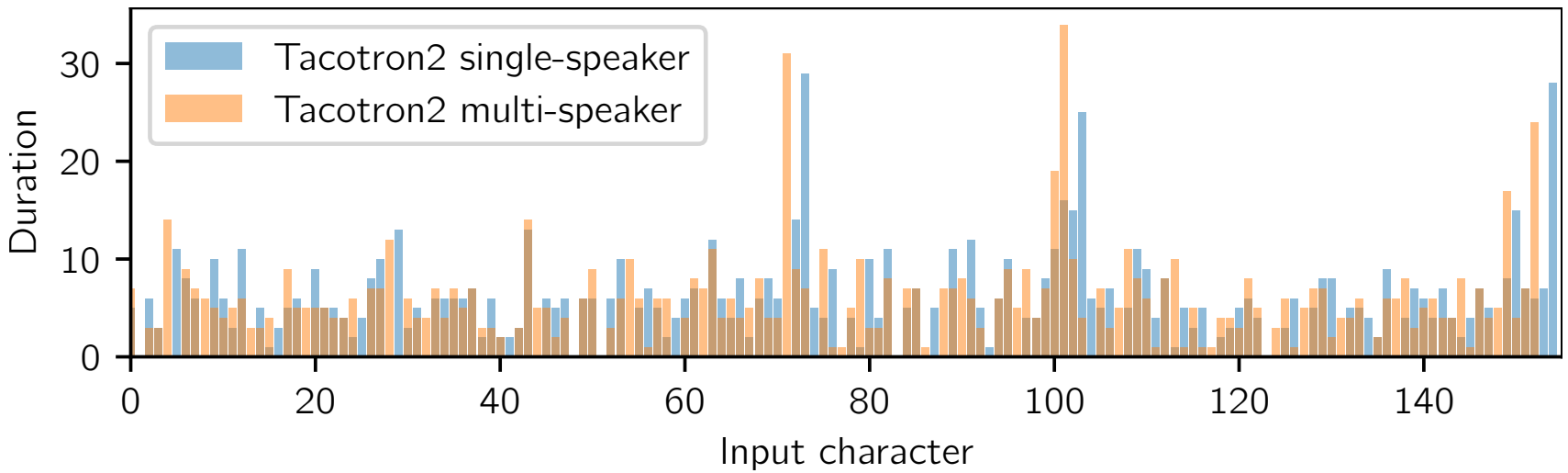


Figure: **Varying character durations** extracted with different Tacotron2 models allow to train FastPitch checkpoint of comparable quality

FastPitch [7]: GT pitch

Let a be the windowed signal, calculated using Hann windows. The algorithm finds an array of maxima of the normalized autocorrelation function r_x

$$r_a(\tau) = \frac{\int_0^{T-\tau} a_t \; a_{t+\tau} \; dt}{\int_0^T a_t^2 \; dt}$$
$$r_x(\tau) = \frac{r_a(\tau)}{r_w(\tau)}, \tag{10}$$

where r_a denotes autocorrelation of the windowed signal, and r_w autocorrelation of the Hann window. These maxima become the candidate frequencies. The unvoiced candidate is present at every time step.

FastPitch [7]: GT pitch

Then, the lowest-cost path through the array of candidates is calculated with the Viterbi algorithm. The path minimizes the transitions between the candidate frequencies.

$$\mathit{TransitionCost}(F_1, F_2) = \begin{cases} 0 & \text{if } F_1 = 0 \text{ and } F_2 = 0 \\ \mathit{VoicedUnvoicedCost} & \text{if } F_1 = 0 \text{ xor } F_2 = 0 \\ \mathit{OctaveJumpCost} & \text{if } F_1 \neq 0 \text{ and } F_2 \neq 0. \end{cases} \tag{11}$$

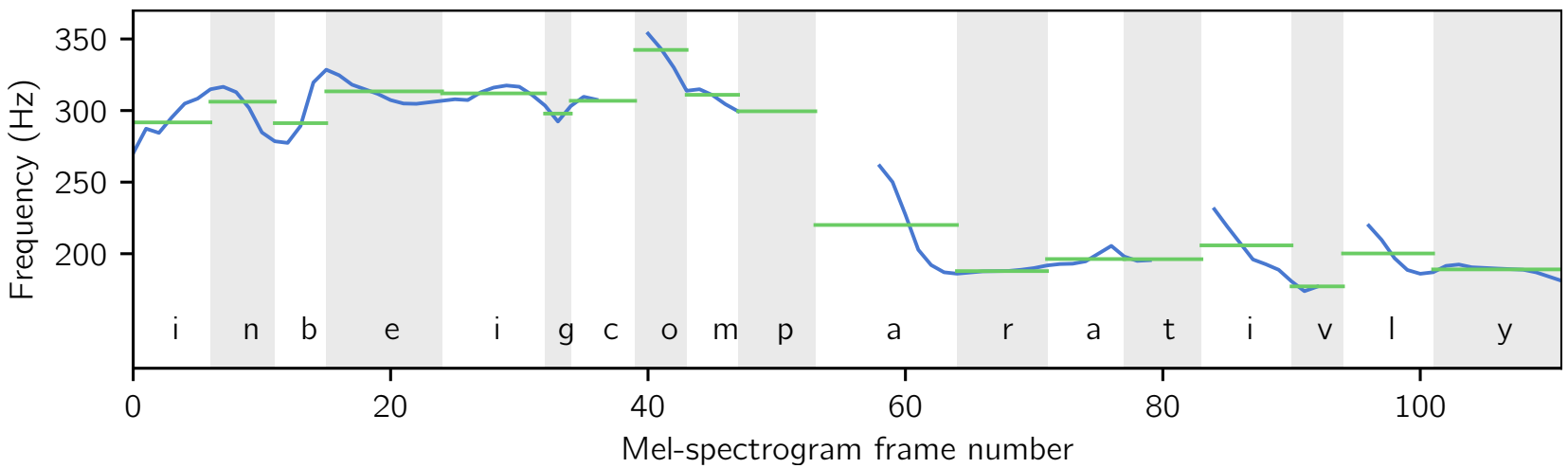


Figure: Raw values are shown in blue, values averaged over input characters in green. Silent characters have zero duration and no fundamental frequency.

FastPitch [7]: results

Training all models on LJSpeech [5] dataset (24 hours of single speaker). 30 samples from test for each model were listened by 60 listeners for MOS (each model got at least 250 scores).

Model	MOS
Tacotron2 (Mel + WaveGlow)	3.946 ± 0.134
FastPitch (Mel + WaveGlow)	4.080 ± 0.133

FastPitch [7]: results

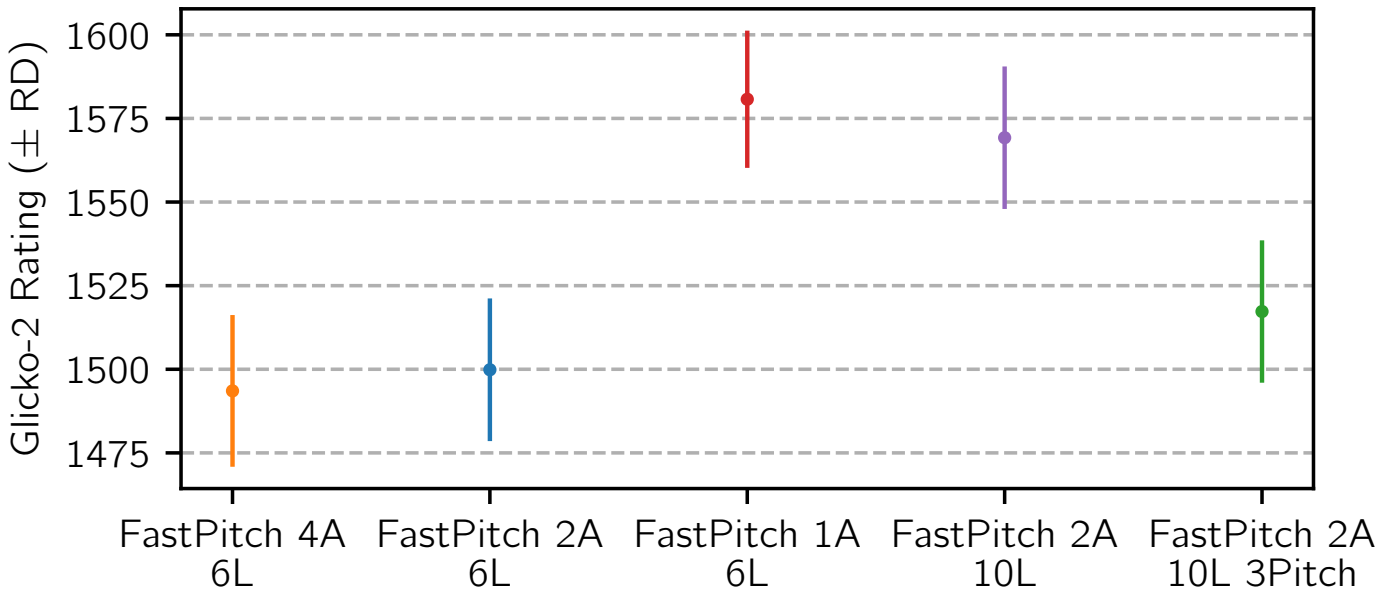


Figure: **Glicko-2 ranking** of different configurations of FastPitch. The compared models have different numbers of attention heads (A), layers (L), and pitch averaged to three values per input symbol instead of one (3Pitch).

FastSpeech 2 [8]: motivation and problems

Motivation:

- Non autoregressive transformer models [4] could be improved with additional conditioning, lets add more conditioning.
- Extracting durations from autoregressive TTS requires training and speaker dependent. Lets use ASR models for extracting durations.
- Common TTS system consists of 2 independently trained part, lets attempt to make it truly end-to-end.

Problems:

- Training end-to-end TTS system is complex due to bigger variance of waveform than mel spectrogram.
- It is difficult to train on the audio clip that corresponds to the full text sequence due to the extremely long waveform samples and limited memory. As a result, we can only train on a short audio clip that corresponds to a partial text sequence which hardens training.

FastSpeech 2 [8]: architecture

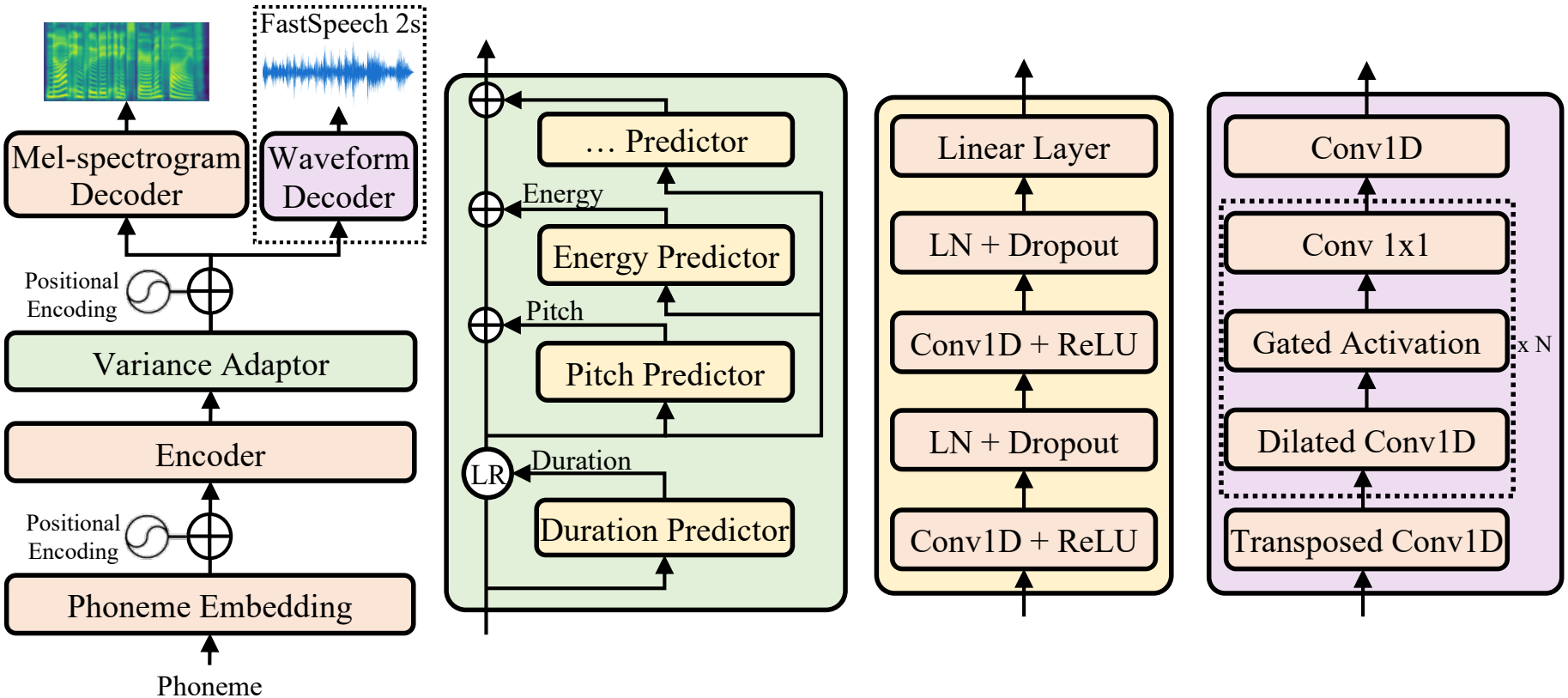


Figure: The overall architecture for FastSpeech 2 and 2s. LR in subfigure (b) denotes the length regulator operation proposed in FastSpeech. LN in subfigure (c) denotes layer normalization. Variance predictor represents duration/pitch/energy predictor.

FastSpeech 2 [8]: architecture

Duration extracted via ASR model MFA [9].
Fundamental frequency F_0 extracted from the raw waveform [8] with the same hop size of target mel-spectrograms to obtain the pitch of each frame, and **compute L2-norm of the amplitude of each STFT frame as the energy**.
F_0 **and energy are quantized for each frame to 256 possible values and encoded them into a sequence of one-hot vectors**. The pitch and energy embeddings are added to the hidden sequence.
The pitch and energy predictors directly predict the values of F_0 and energy instead of the one-hot vector and are optimized with mean square error.

[8] $F0$ extracted using PyWorldVocoder from `https://github.com/JeremyCCHsu/Python-Wrapper-for-World-Vocoder`

FastSpeech 2 [8]: results

Training all models on LJSpeech [5] dataset (24 hours of single speaker). 523 samples from test for each model were listened by 12 listeners for MOS.

Method	MOS
GT	4.27 ± 0.07
GT (Mel + PWG)	3.92 ± 0.08
Tacotron 2(Mel + PWG)	3.74 ± 0.07
Transformer TTS (Mel + PWG)	$\mathbf{3.79 \pm 0.08}$
FastSpeech (Mel + PWG)	3.67 ± 0.08
FastSpeech 2 (Mel + PWG)	3.77 ± 0.08
FastSpeech 2s	$\mathbf{3.79 \pm 0.08}$

Table: The MOS with 95% confidence intervals.

FastSpeech 2 [8]: results

The comparison of training time and inference latency in waveform synthesis. RTF denotes the real-time factor, that is the time (in seconds) required for the system to synthesize one second waveform. The training and inference latency test is conducted on a server with 36 Intel Xeon CPU, 256GB memory, 1 NVIDIA V100 GPU and batch size of 48 for training and 1 for inference.

Method	Training Time (h)	RTF	Inference Speedup
Transformer TTS	38.64	8.26×10^{-1}	/
FastSpeech	53.12	5.41×10^{-3}	152×
FastSpeech 2	**16.46**	5.51×10^{-3}	149×
FastSpeech 2s	/	$\mathbf{4.87 \times 10^{-3}}$	**170**×

FastSpeech 2 [8]: results

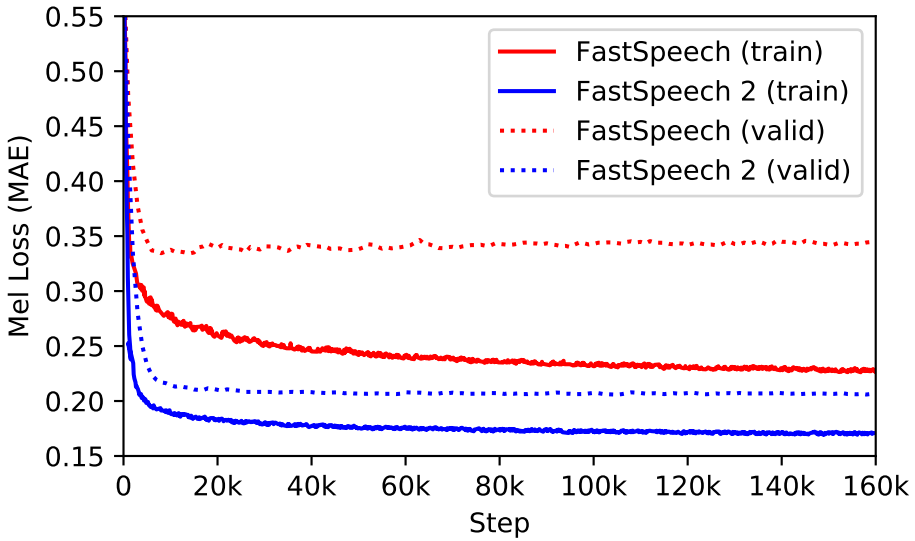


Figure: The training and validation loss curves of FastSpeech and FastSpeech 2.

Method	Pitch	Energy
FastSpeech	21.67	0.142
FastSpeech 2	20.30	**0.131**
FastSpeech 2s	**20.28**	0.133

Table: The mean absolute error of the pitch and energy in synthesized speech audio.

FastSpeech 2 [8]: results

50 audio were manually aligned with the corresponding text in phoneme level to get the ground-truth phoneme-level duration.

Method	Δ (ms)
Duration from teacher model	19.68
Duration from MFA	12.47

Table: Alignment accuracy comparison.

Setting	CMOS
FastSpeech + Duration from teacher	0
FastSpeech + Duration from MFA	+0.195

Table: CMOS comparison.

FastSpeech 2 [8] results

Setting	CMOS
FastSpeech 2	0
FastSpeech 2 - energy	-0.045
FastSpeech 2 - pitch	-0.230
FastSpeech 2 - pitch - energy	-0.385

Table: CMOS comparison for FastSpeech 2.

Setting	CMOS
FastSpeech 2s	0
FastSpeech 2s - energy	-0.150
FastSpeech 2s - pitch	-1.045
FastSpeech 2s - pitch - energy	-1.070

Table: CMOS comparison for FastSpeech 2s.

[1] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, R. J. Skerry-Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and Yonghui Wu.
Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions.
CoRR, abs/1712.05884, 2017.

[2] Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, Ming Liu, and Ming Zhou.
Close to human quality TTS with transformer.
CoRR, abs/1809.08895, 2018.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin.
Attention is all you need.
CoRR, abs/1706.03762, 2017.

[4] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu.
Fastspeech: Fast, robust and controllable text to speech.
CoRR, abs/1905.09263, 2019.

[5] Keith Ito.
The lj speech dataset.
`https://keithito.com/LJ-Speech-Dataset/`, 2017.

[6] Ryan Prenger, Rafael Valle, and Bryan Catanzaro.
Waveglow: A flow-based generative network for speech synthesis.
CoRR, abs/1811.00002, 2018.

[7] Adrian Łańcucki.
Fastpitch: Parallel text-to-speech with pitch prediction, 2020.

[8] Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu.
Fastspeech 2: Fast and high-quality end-to-end text to speech, 2020.

[9] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger.
Montreal forced aligner: Trainable text-speech alignment using kaldi. pages 498–502, 08 2017.