

курс «Глубокое обучение»

Синтез речи: вокодеры

Александр Дьяконов

28 ноября 2022 года

План

Модели:

DeepVoice: Real-time Neural TTS

Deep Voice 2: Multi-Speaker Neural Text-to-Speech

Tacotron (Google)

DeepVoice3

Tacotron 2 = Tacotron-style model + modified

Melotron

Glow-TTS

FastSpeech2

FastSpeech2

Вокодеры:

Griffin-Lim

WaveNet

HiFiGAN

Синтез: задача

Постановка задачи TTS:

текст → **МОДЕЛЬ** → звук

Реальность:

текст → **НОРМАЛИЗАТОР / выделение фонем** → **фонемы** → **МОДЕЛЬ** → мел-спектрограмма
→ **ВОКОДЕР** → звук

подчёркнутая часть может отсутствовать

Этапы

генерация спектрограммы (predictor)
генерация звука (vocoder)

Синтез: проблемы

- **Нужен очень хороший датасет**
Чистый звук, ~ 10x часов
- **Желательно при генерации выбирать голос, тон (удивление, рассерженность и т.п.)**
- **Сложно оценивать модели (нет правильного ответа, субъективность, много возможных ошибок)**
 - **Нужна качественная натуральная речь (а что это?!)**
 - **Синтез должен быть быстрый (для формирования ответа для бота)**

Размер воспроизводимых данных зависит от их качества (ограничения на качество)
– частоты дискретизации и битовой глубины

**1 секунда 44.1 kHz 16 битного звука
= 44100·65536 значений**

Синтез: зачем нужен вокодер

Почему синтез двухэтапный?

- **промежуточное пространство более бедное**
нужно меньше признаков восстанавливать по тексту
- **оно может учитывать особенности человеческого восприятия**
поэтому, например, мел-спектrogramмы
 - **вокодеры можно обучать отдельно**
на неразмеченных голосовых данных

Синтез: применение

Широкая область применения:

- **голосовые ассистенты (Apple Siri, Google Now, Amazon Alexa, Yandex Алиса)**
- **помощь для людей с проблемами зрения**
- **озвучка персонажей в компьютерных играх и кино**

Почему нужен свой синтез?

- **есть платные решения – далеки от идеала (на русском)**
- **открытые решения – нужно «доделывать»**
- **дополнительные требования (клонирование голоса, тест Тьюринга)**

Синтез: нормализация данных (Text Normalization)

4 March 2014	the fourth of march twenty fourteen
C2	c two
.266	point two six six
Games.com	games dot com

+ в один регистр, удаление спец-символов

- числа (везде, особенно – время, суммы, адреса, телефоны)
 - аббревиатуры
 - url

нужна, если есть неправильная разметка

можно собирать датасет уже с правильной разметкой

зависит от контекста

4 → четыре, четверо, четвёртый, ...

решение: seq2seq / WER

<https://www.kaggle.com/c/text-normalization-challenge-english-language/>

Синтез: сегментация (Word segmentation)

применяется для некоторых языков, ex: китайского

Синтез: разметка частей речи (Part-of-speech tagging)

в некоторых языках (англ.) от части речи зависит произношение

Синтез: предсказание ударения (Prosody prediction)

Prosody (rhythm + stress + intonation) → syllable duration, loudness, pitch

иногда прогноз подобных вещей встроен в текстовый кодировщик
ToBI (tones and break indices)

Kim Silverman, et al. «Tobi: A standard for labeling english prosody»? 1992

Andrew Rosenberg «Autobi-a tool for automatic tobi annotation»., 2010

Синтез: разметка данных

в идеале – разметка ударений

в русском – простановка ё

ещё бывает первое, второе ударение и т.п.

назовите ват+ше и+мя, пож+луйста

решение: написание простановщика ударений seq2seq, transformer(!)

есть словари, но есть проблема контекста:

замотк – затмок

в идеале – разметка интонации

ВАМ пришло подтверждение?

вам ПРИШЛО подтверждение?

вам пришло ПОДТВЕРЖДЕНИЕ?

в идеале – разметка пауз

извините, Р1 вы меня слышите? Р3 скажите да, если слышите.

решение: forced alignment (определение продолжительности звуков + пауз)

Синтез: перевод в фонемную запись – Grapheme-to-phoneme (G2P) conversion

TTS может работать с буквами и фонемами

тендер	[т'эн'д'эр]
регрессия	[р'эгр'эс' : ийа]

произнесение отличается от написания

Синтез: оценка качества

MOS (mean opinion score) – опрашивается группа людей, каждому человеку требуется оценить аудиозапись по пятибалльной шкале, результаты усредняются

субъективно, долго, дорого

MOS в разных статьях не согласованы

CMOS – сравнение двух аудио

Датасеты

LJSpeech

LibriTTS

CommonVoice

OpenTTS

Традиционные подходы

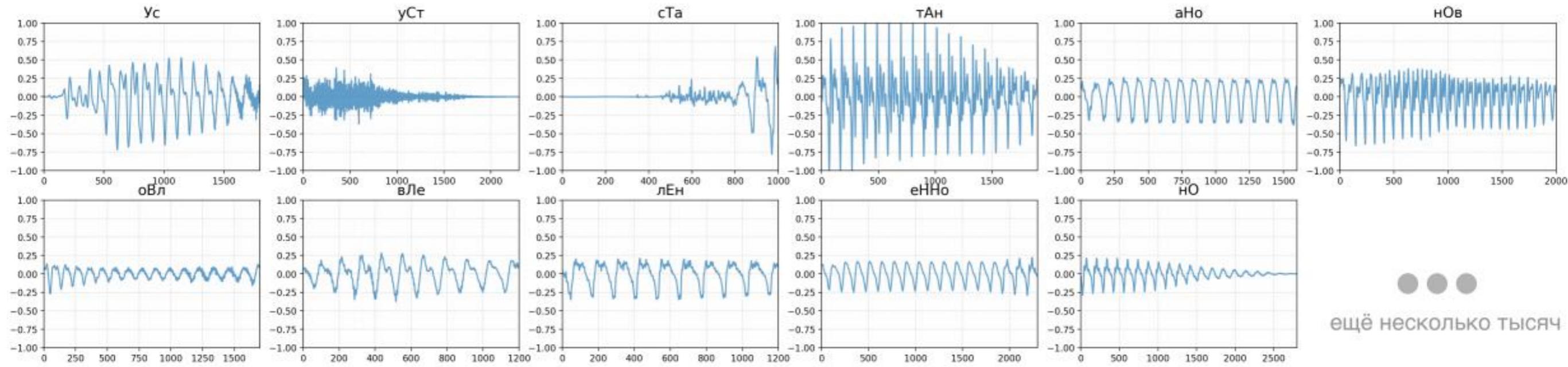
Конкатенативный подход (Concatenative)
натурально, но не гибко, большие модели
соединяя предзаписанные кусочки речи

Statistical Parametric
более гибкие, небольшие модели
сначала прогнозируем некоторые акустические параметры, потом по ним звук

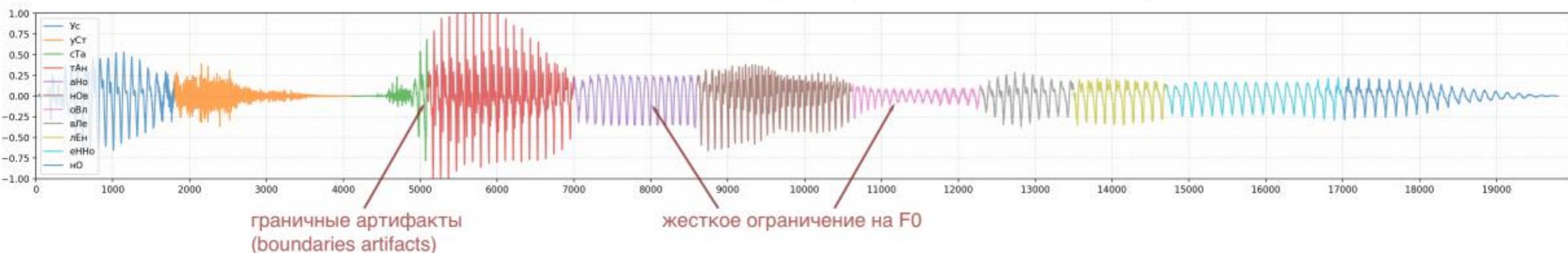
есть и другие
Articulatory Synthesis – симуляция движения губ, языка и т.п.
Formant Synthesis – на наборе правил

Конкатенативный подход

Коллекция звуков (units database)



Конкатенация звуков (units concatenation)



Нейросетевой подход

Естественная архитектура «кодировщик-декодировщик»

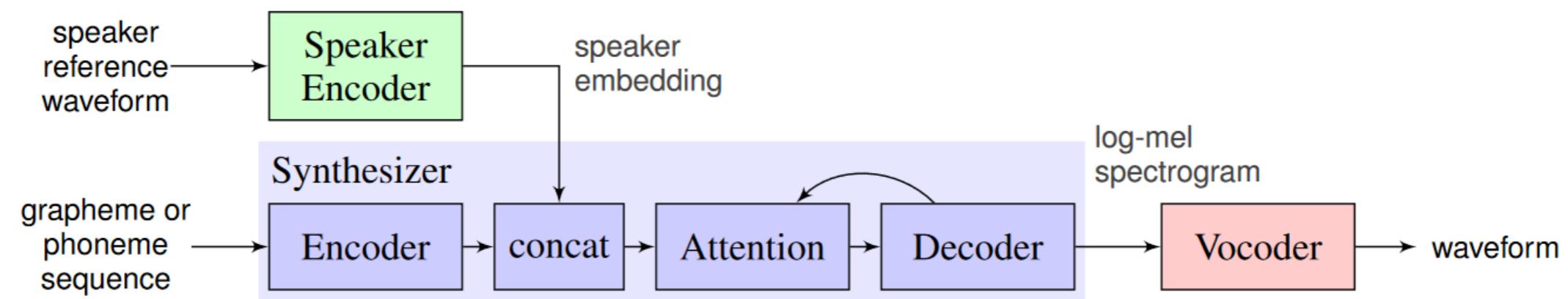


Figure 1: Model overview. Each of the three components are trained independently.

<https://arxiv.org/pdf/1806.04558.pdf>

TTS: как генерируется спектрограмма

Авторегрессионно

Tacotron

Melotron

Transformer TTS

FlowTron

нужно много запусков для генерации

главная проблема, что длина выхода не определяется длиной входа и они различны

Параллельно

Fast speech

Fast pitch

Align-tts

Flow-tts

Glow-tts

нужно предсказывать длины произнесения токенов / фонем

Архитектура

CNN, RNN, transformer, Flow (иногда смесь)

**Проблема – по тексту надо сигнал,
причём его длина неизвестна!**

TTS: размытое понятие end2end

Естественно, end2end: text → waveform (FastSpeech 2s, EATs, ClariNet)

Но в статьях можно встретить такие якобы end2end:

WaveNet: linguistic features → waveform

Tacotron: character / phoneme → spectrogram

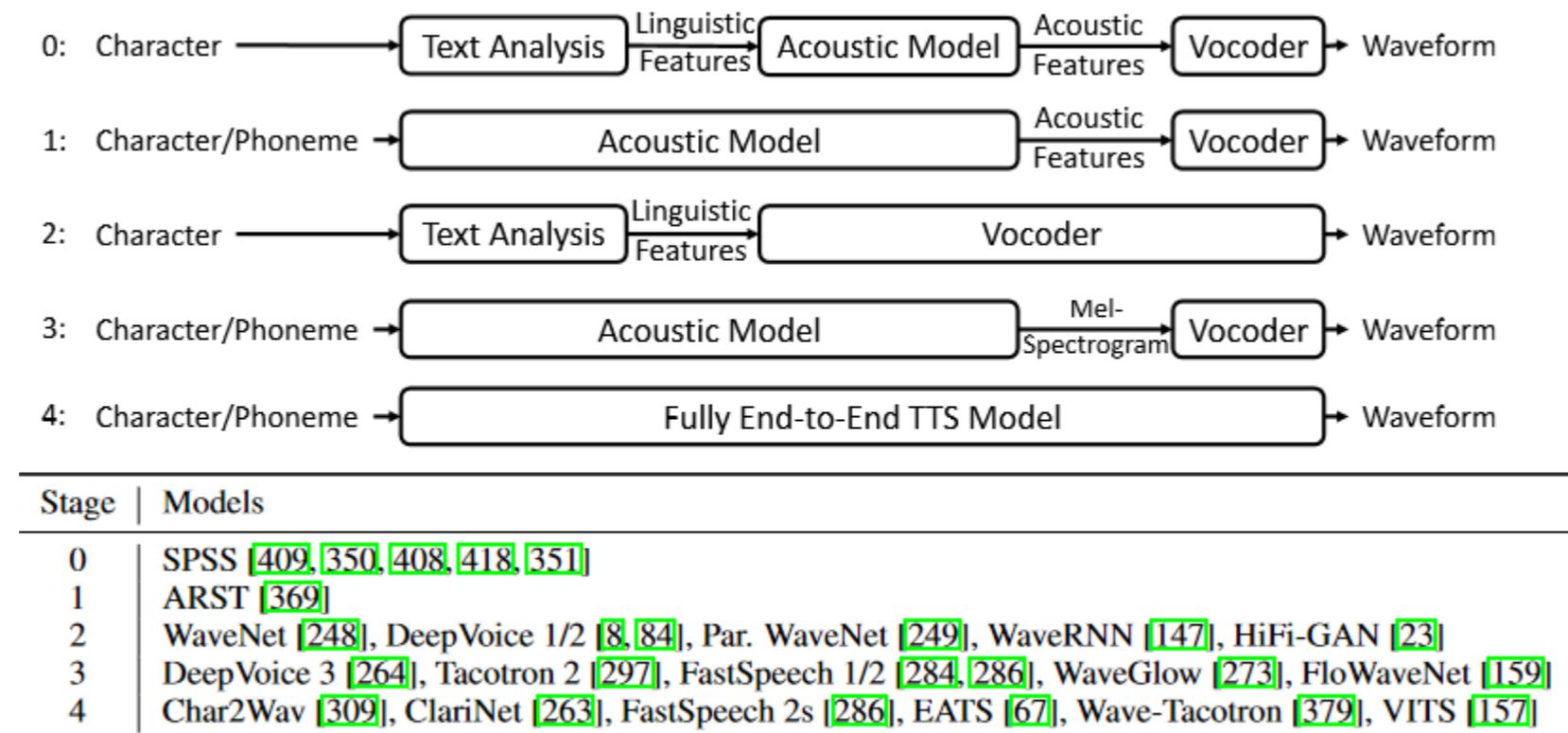
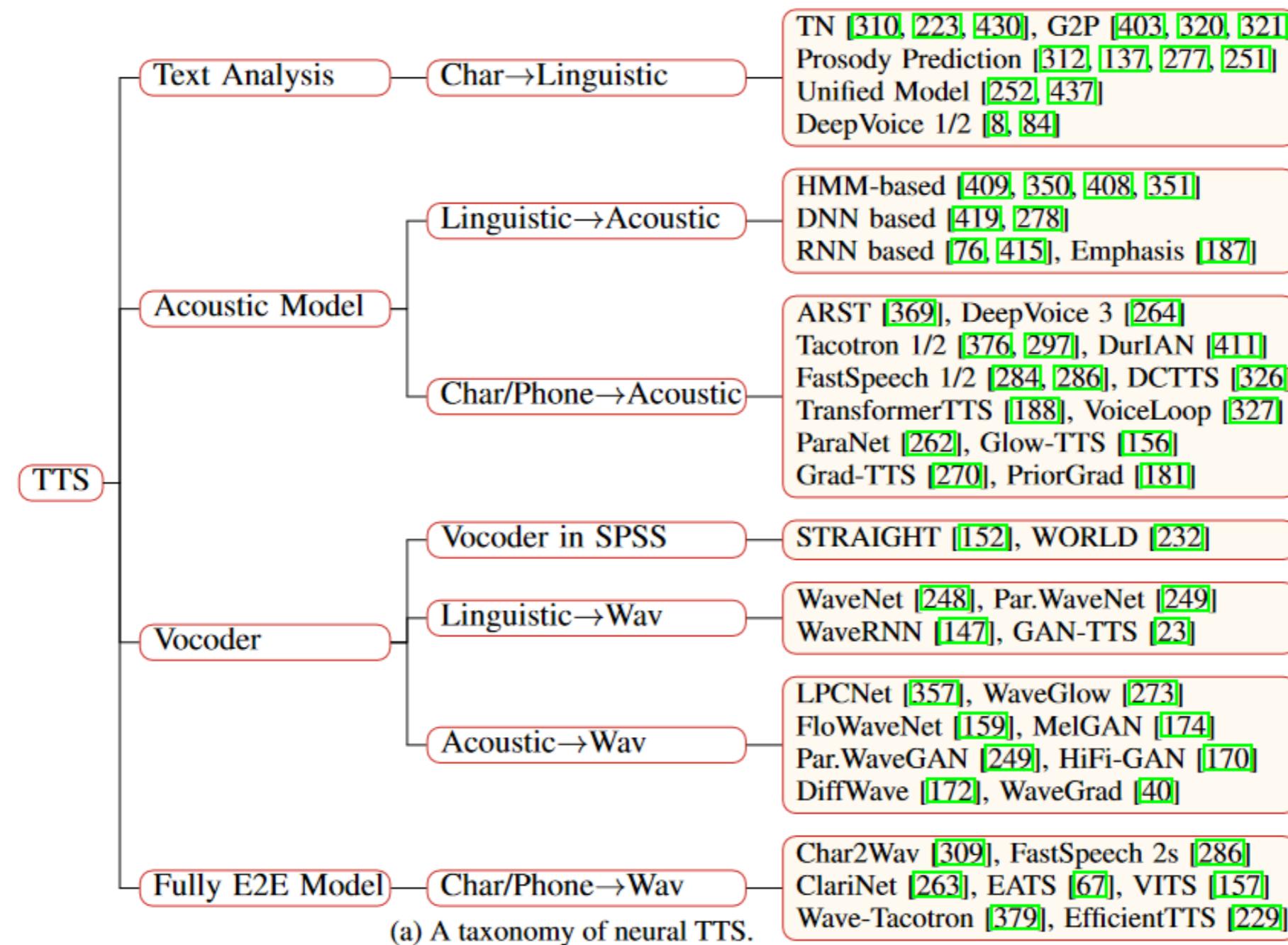
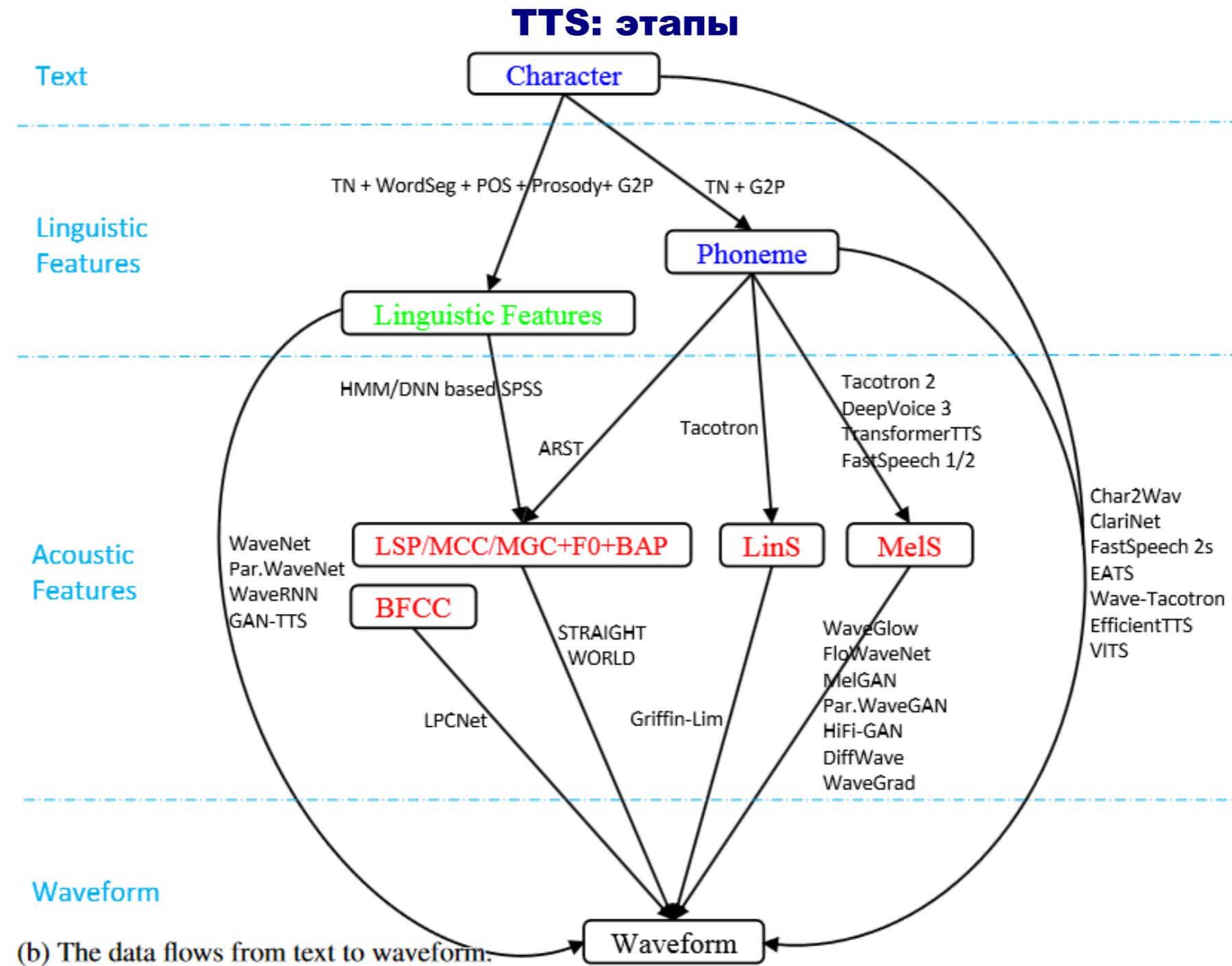


Figure 4: The progressively end-to-end process for TTS models.

TTS: этапы





Вокодеры: спектrogramма → аудио

- **параметрические вокодеры генерируют звук по F0, spectral envelope и т.д. WORLD и др.**
- **алгоритм Griffin-Lim оценки сигнала по спектrogramме**
- **нейросетевые SampleRNN, WaveNet и модификации, WaveRNN**

Table 3: A list of vocoders and their corresponding characteristics.

Vocoder	Input	AR/NAR	Modeling	Architecture
WaveNet [248]	Linguistic Feature	AR	/	CNN
SampleRNN [227]	/	AR	/	RNN
WaveRNN [147]	Linguistic Feature	AR	/	RNN
LPCNet [357]	BFCC	AR	/	RNN
Univ. WaveRNN [209]	Mel-Spectrogram	AR	/	RNN
SC-WaveRNN [259]	Mel-Spectrogram	AR	/	RNN
MB WaveRNN [411]	Mel-Spectrogram	AR	/	RNN
FFTNet [142]	Cepstrum	AR	/	CNN
Par. WaveNet [249]	Linguistic Feature	NAR	Flow	CNN
WaveGlow [273]	Mel-Spectrogram	NAR	Flow	Hybrid/CNN
FloWaveNet [159]	Mel-Spectrogram	NAR	Flow	Hybrid/CNN
WaveFlow [265]	Mel-Spectrogram	AR	Flow	Hybrid/CNN
SqueezeWave [426]	Mel-Spectrogram	NAR	Flow	CNN
WaveGAN [66]	/	NAR	GAN	CNN
GELP [146]	Mel-Spectrogram	NAR	GAN	CNN
GAN-TTS [23]	Linguistic Feature	NAR	GAN	CNN
MelGAN [174]	Mel-Spectrogram	NAR	GAN	CNN
Par. WaveGAN [396]	Mel-Spectrogram	NAR	GAN	CNN
HiFi-GAN [170]	Mel-Spectrogram	NAR	GAN	Hybrid/CNN
VocGAN [401]	Mel-Spectrogram	NAR	GAN	CNN
GED [93]	Linguistic Feature	NAR	GAN	CNN
Wave-VAE [262]	Mel-Spectrogram	NAR	VAE	CNN
WaveGrad [40]	Mel-Spectrogram	NAR	Diffusion	Hybrid/CNN
DiffWave [172]	Mel-Spectrogram	NAR	Diffusion	Hybrid/CNN
PriorGrad [181]	Mel-Spectrogram	NAR	Diffusion	Hybrid/CNN

Вокодеры: Griffin-Lim

вход: $a = \|\text{STFT}(y)\|$

известна магнитуда, но не известна фаза

инициализация: $\varphi \sim U[0, 2\pi)$

приближение: $f = \text{ISTFT}(ae^{i\varphi})$

в цикле: $\varphi = \arg \text{STFT}(f), \quad f = \text{ISTFT}(ye^{i\varphi})$

здесь нет обучения, можно сделать параллельно и быстро

есть «металличность» в результате

STFT = short-time Fourier transform (оконное преобразование Фурье)

Вокодеры: WaveNet

первый нейросетевой вокодер

dilated convolutions

end-to-end-обучение без учёта какой-то специфики речи

авторегрессия $p(x_t | x_{t-1}, \dots, x_1; \lambda)$

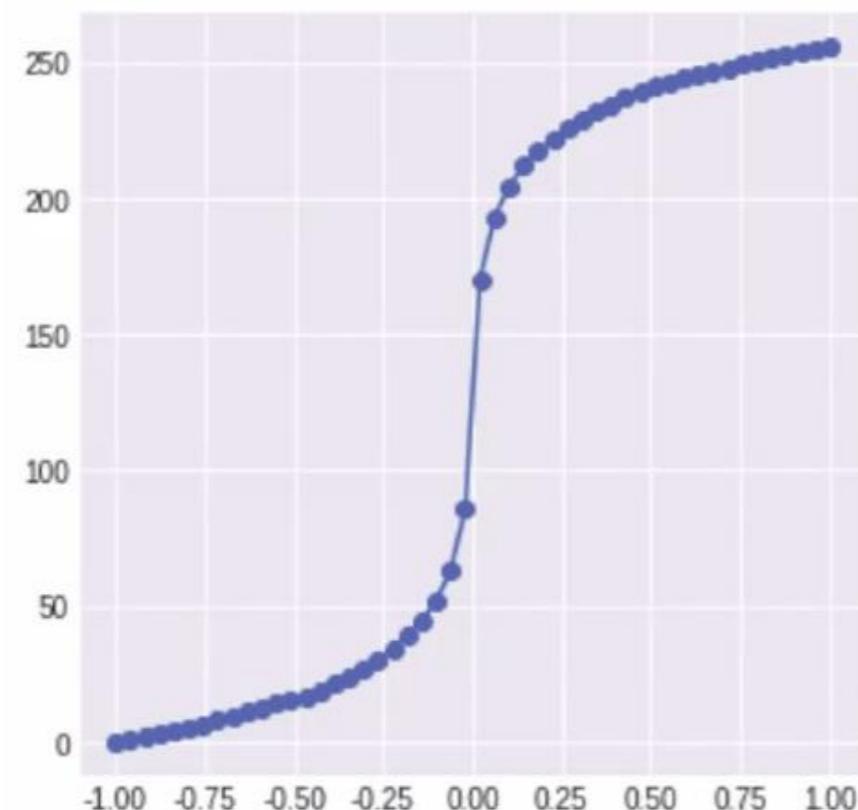
$$-\sum_t \log p(x_t | x_{t-1}, \dots, x_1; \lambda) \rightarrow \min$$

лучше ответ сэмплировать из распределения $p(x_t | x_{t-1}, \dots, x_1; \lambda)$
(более естественный)

он более ресурсоёмкий, не очень быстрый

<https://deepmind.com/blog/article/wavenet-generative-model-raw-audio>

<https://arxiv.org/pdf/1609.03499.pdf>

WaveNet: μ-Law

$$f(x_t) = \text{sign}(x_t) \frac{\ln(1 + \mu |x_t|)}{\ln(1 + \mu)}$$

$$-1 < x_t < 1 \text{ and } \mu = 255$$

Обычно звук 16-битовый, поэтому надо предсказывать одно значение из 65536 в каждый момент времени

Выход: 8-битный сигнал (256 значений – через softmax предсказываем) + специальное преобразование μ-Law, которое соответствует «человеческой восприимчивости»

Causal Convolution («причинная» свёртка)

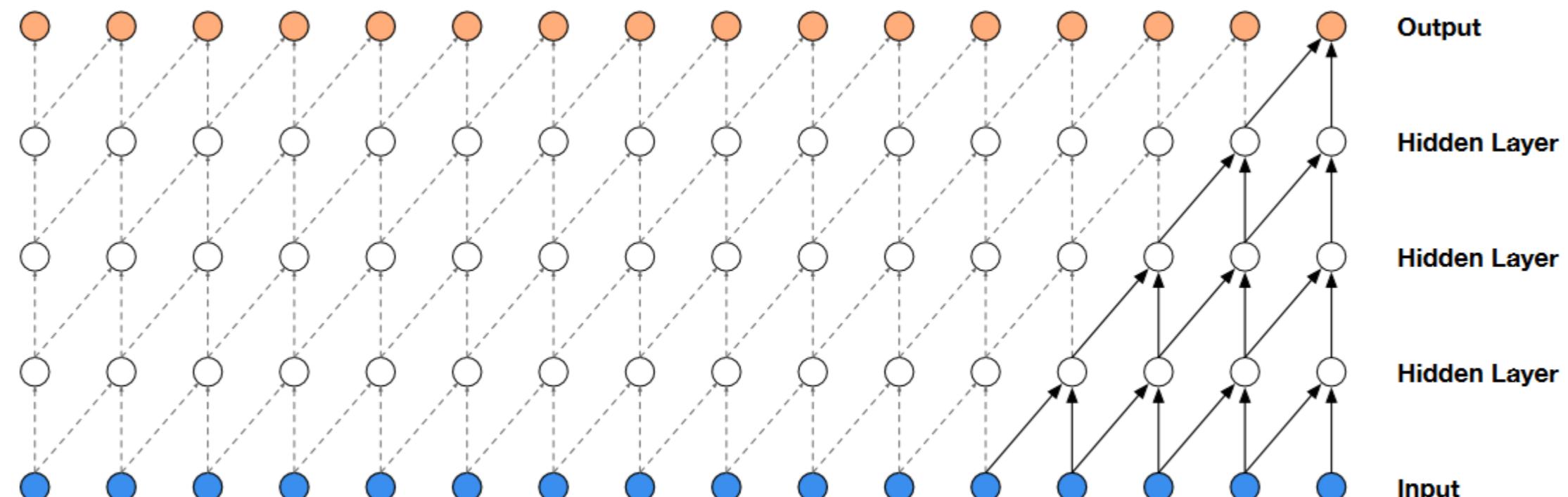
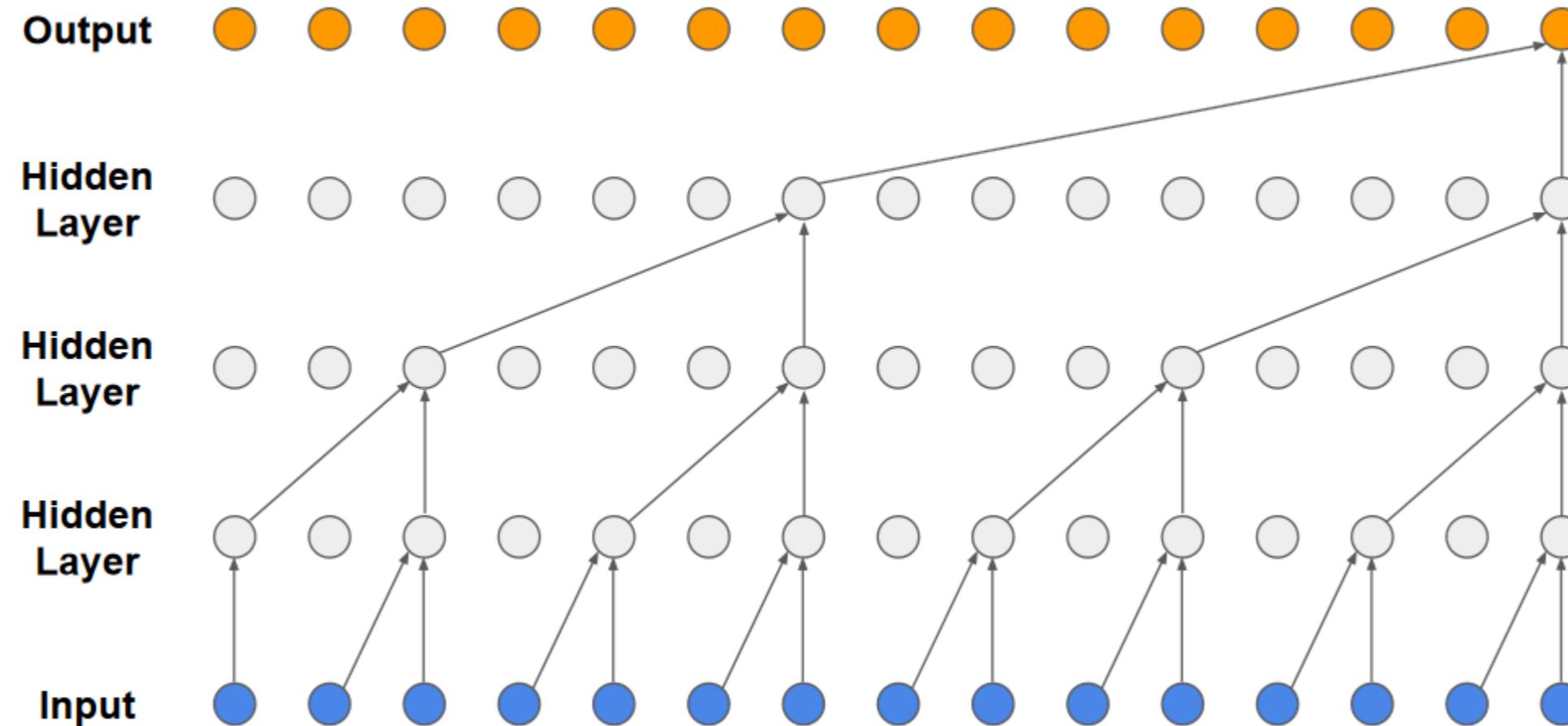


Figure 2: Visualization of a stack of causal convolutional layers.

**нет зависимости от будущего
можно распараллелить – в разные моменты времени считать параллельно**

Causal Dilated Convolution («причинная» разреженная свёртка)



Causal Dilated Convolution («причинная» разреженная свёртка)

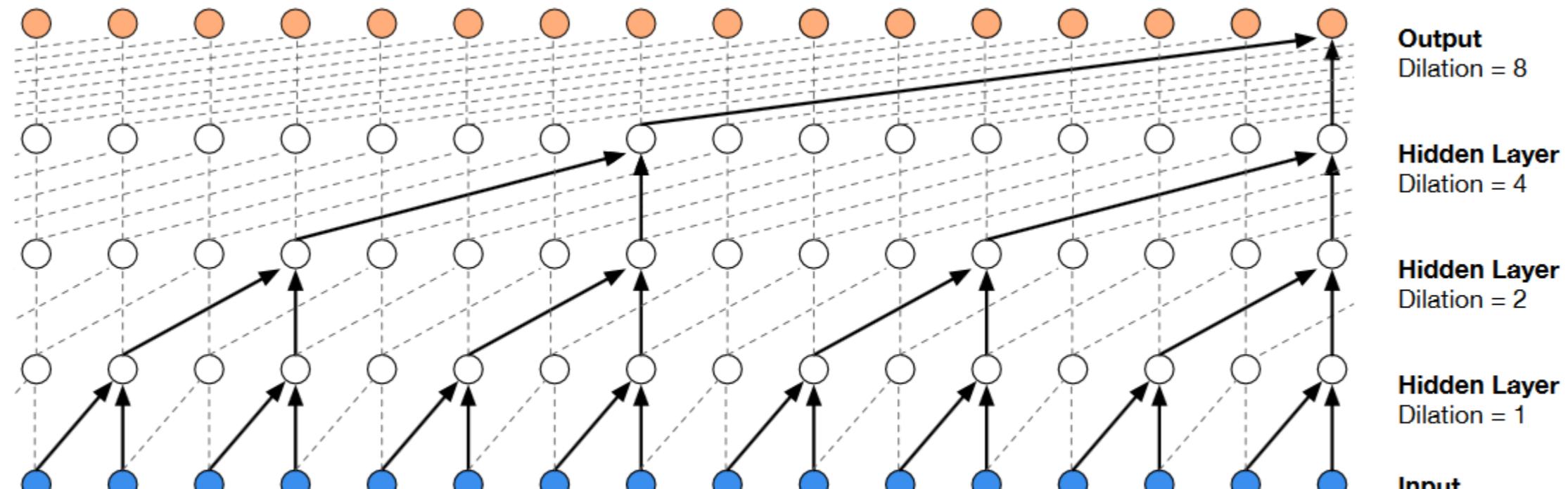


Figure 3: Visualization of a stack of *dilated* causal convolutional layers.

**даже у короткого звука много точек – нужна большая зона охвата
на каждом слое один набор весов (мало параметров)
каждый слой – блок след. слайд – его архитектура**

WaveNet: Gated activation units

Как в Pixel-CNN:

$$\mathbf{z} = \tanh(W_{f,k} * \mathbf{x}) \odot \sigma(W_{g,k} * \mathbf{x})$$

при условной генерации (ex: для учёта информации о спикере):

$$\mathbf{z} = \tanh(W_{f,k} * \mathbf{x} + V_{f,k}^T \mathbf{h}) \odot \sigma(W_{g,k} * \mathbf{x} + V_{g,k}^T \mathbf{h})$$

*** – свёртка, (.) – поэлементное умножение**

k – номер слоя

f – filter

g – gate

WaveNet: Residual and skip connections

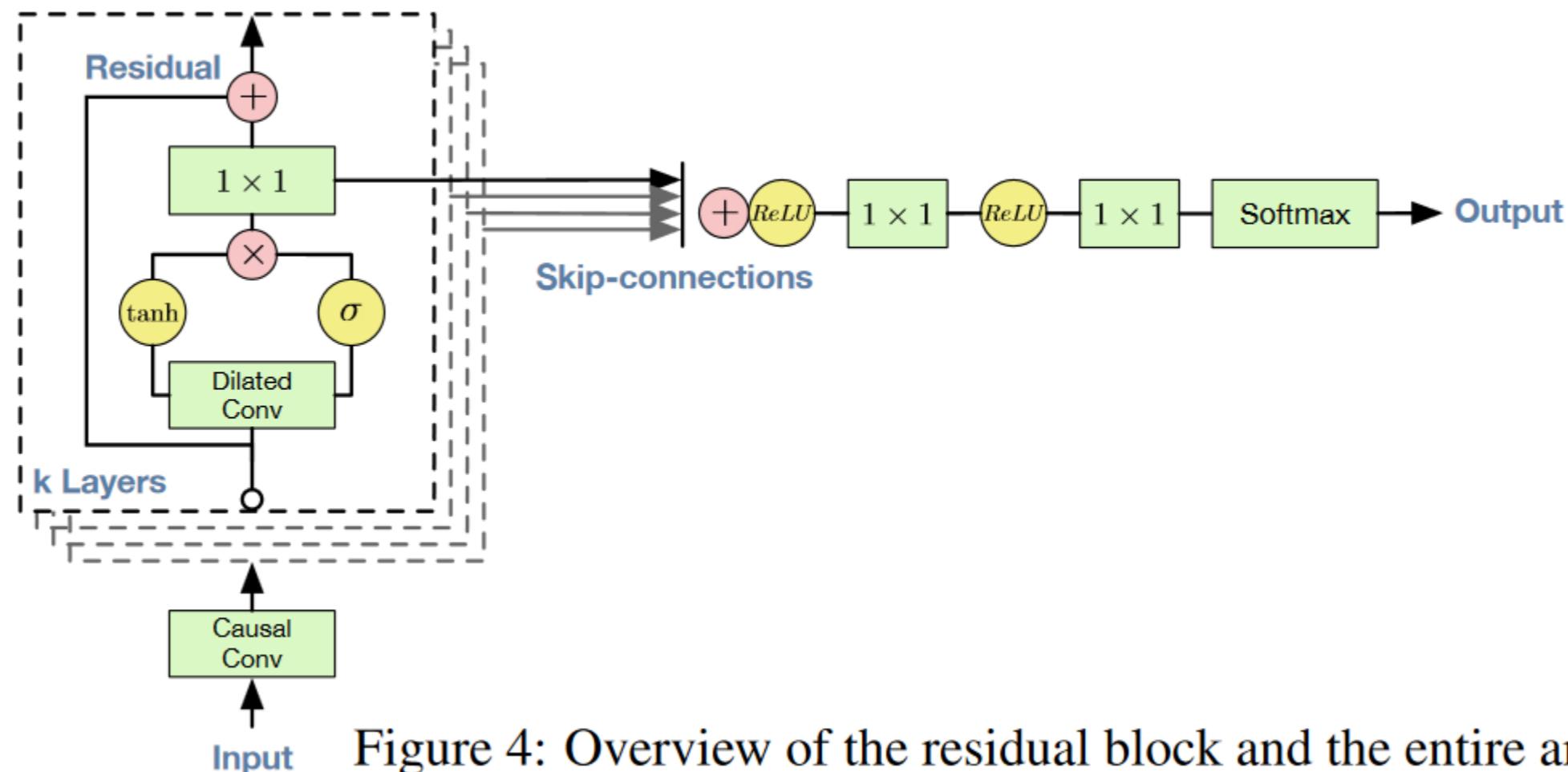


Figure 4: Overview of the residual block and the entire architecture.

+ residual
+ skip-connection
нет пулинга

однотипные слои, суммирование со всех слоёв

Вокодеры: Parallel WaveNet

Inverse autoregressive flows (IAFs) + WaveNet

Aaron van den Oord, et al. «Parallel WaveNet: Fast High-Fidelity Speech Synthesis» //

<https://arxiv.org/abs/1711.10433>

WaveNet – «teacher»

parallel WaveNet – «student»

Power loss — энергия в разных частотных каналах такая же, как в речи человека

Perceptual loss — reconstruction loss (the Euclidean distance between feature maps in the classifier) and style loss (the Euclidean distance between the Gram matrices)

Contrastive loss – penalizes waveforms that have high likelihood regardless of the conditioning vector

Parallel WaveNet

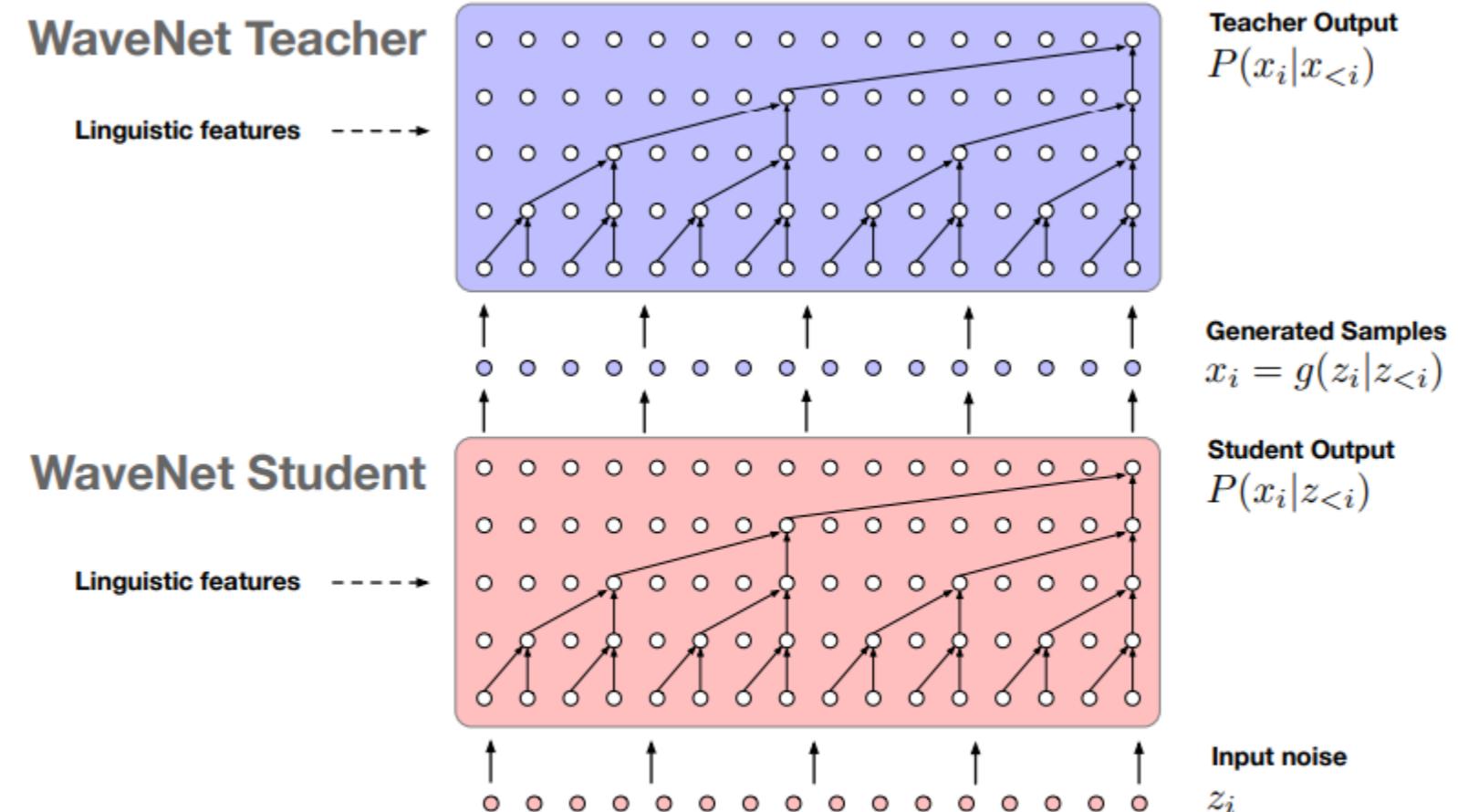


Figure 2: **Overview of Probability Density Distillation.** A pre-trained WaveNet teacher is used to score the samples x output by the student. The student is trained to minimise the KL-divergence between its distribution and that of the teacher by maximising the log-likelihood of its samples under the teacher and maximising its own entropy at the same time.

Вокодеры: HiFiGAN

основан на GAN

один генератор

мелспектrogramma → транспонированные свёртки пока длина не станет = длине сырого
сигнала

после каждой свёртки multi-receptive field fusion (MRF) module

два дискриминатора: multi-scale and multi-period

возможно усложнение и увеличение числа параметров
используется weight_norm
на вход не подаётся шум, как и в MelGAN

Jungil Kong et al. «HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity
Speech Synthesis» // <https://arxiv.org/abs/2010.05646> <https://github.com/jik876/hifi-gan>

Вокодеры: HiFiGAN

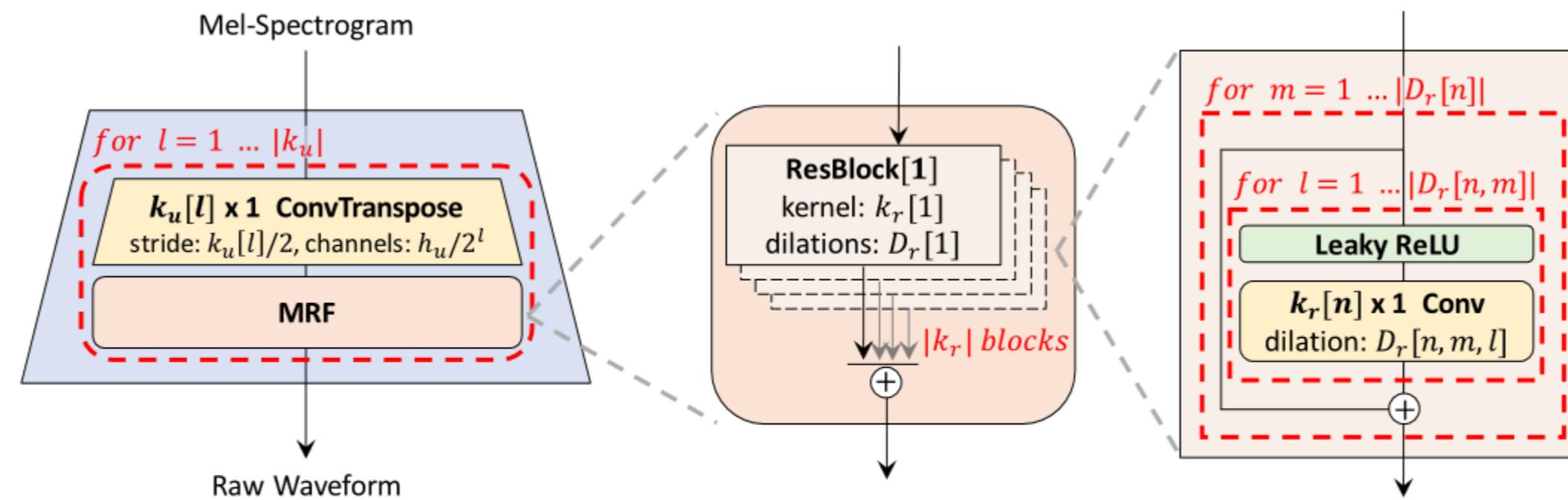


Figure 1: The generator upsamples mel-spectrograms up to $|k_u|$ times to match the temporal resolution of raw waveforms. A MRF module adds features from $|k_r|$ residual blocks of different kernel sizes and dilation rates. Lastly, the n -th residual block with kernel size $k_r[n]$ and dilation rates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

MRF – сумма Res-блоков (каждый с разным kernel sizes / dilation rates)

HiFiGAN: Multi-Period Discriminator (MPD)

**Предварительный решейп сигнала
Ширину берут 2, 3, 5, 7, 11
– несколько дискриминаторов**

**Сделали эксперимент на искусственных данных, в котором показали полезность такого
дискриминатора**

Multi-Scale Discriminator (MSD)

Сделали по аналогии с MelGAN

Смесь трёх дискриминаторов с разным входом:

raw audio, ×2 average-pooled audio, and ×4 average-pooled audio,

HiFiGAN – несколько видов дискриминаторов

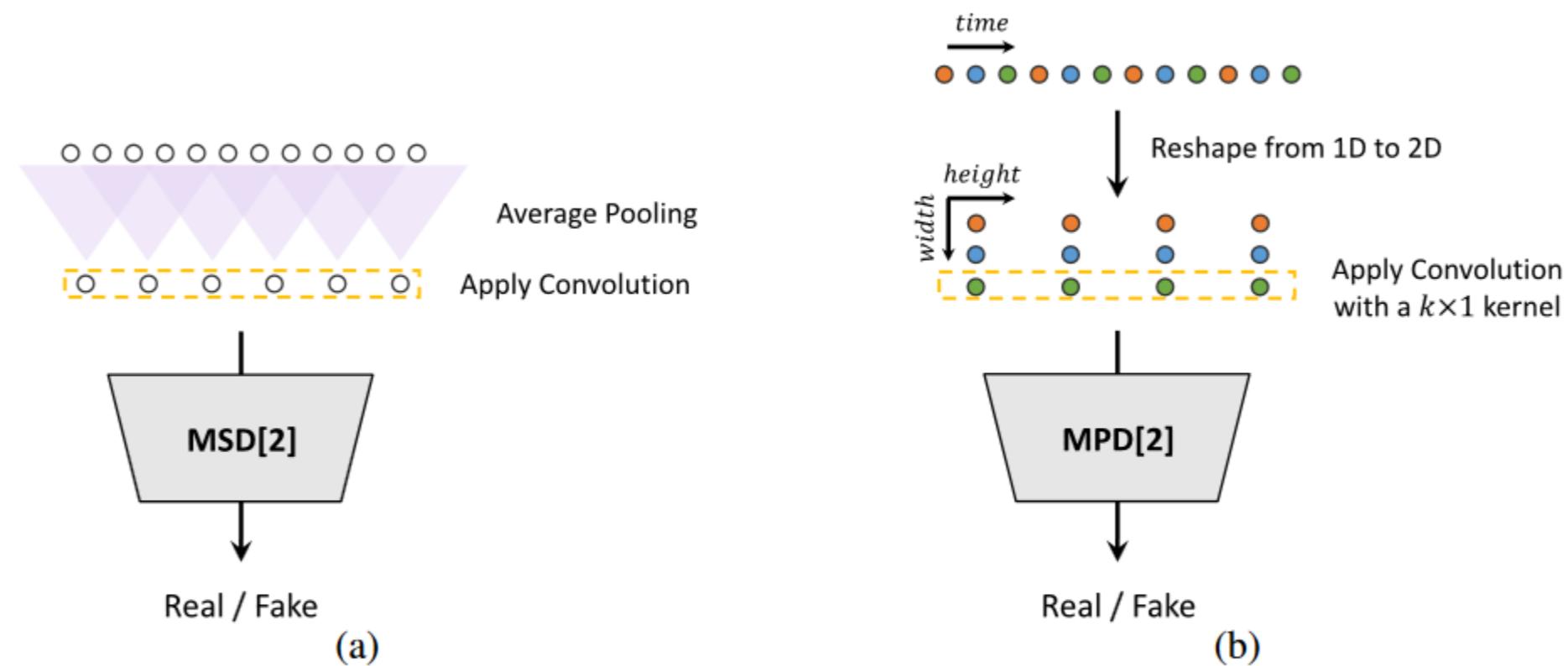
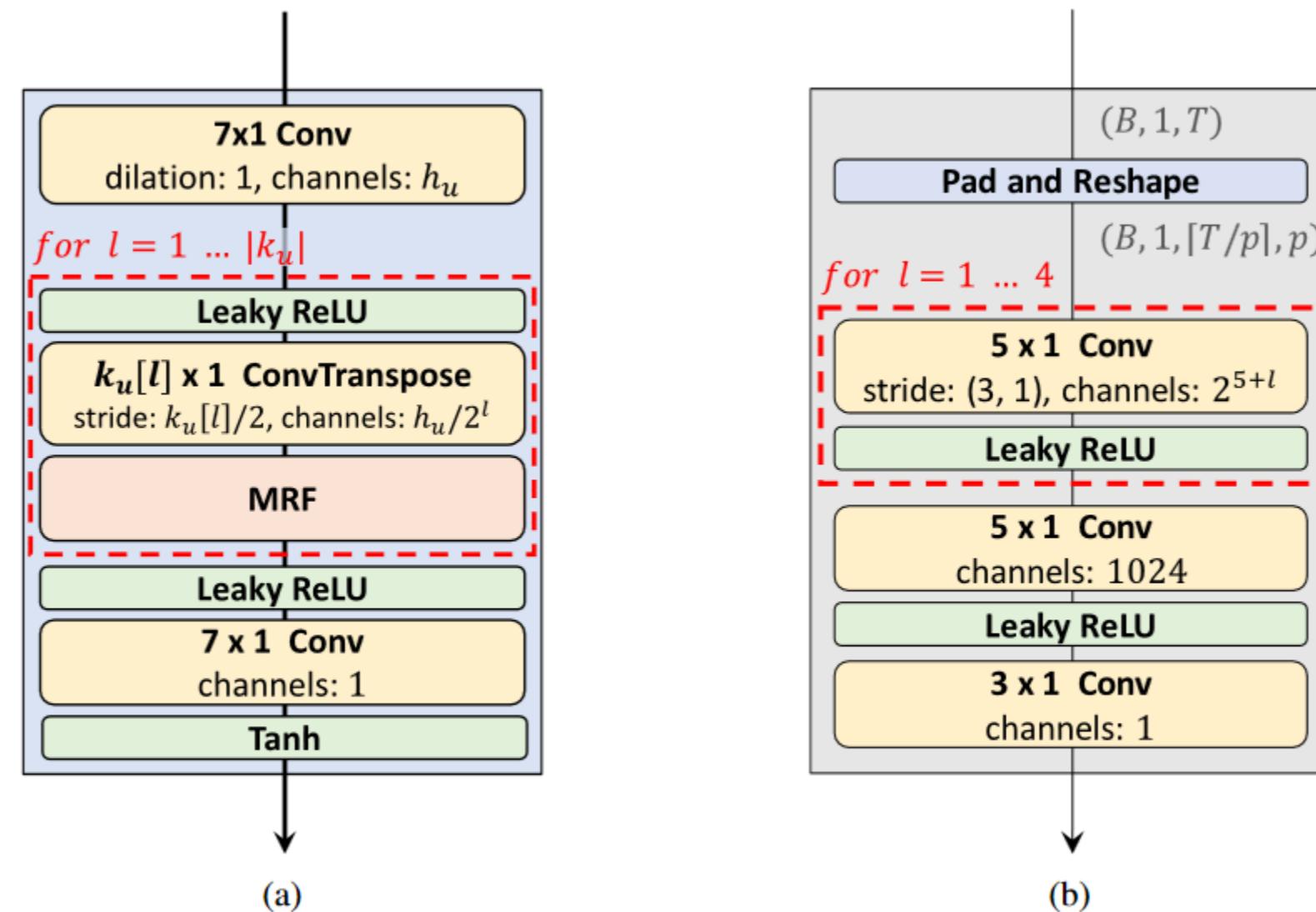


Figure 2: (a) The second sub-discriminator of MSD. (b) The second sub-discriminator of MPD with period 3.

HiFiGAN

Figure 4: (a) The generator. (b) The sub-discriminator of MPD with period p .

Multi-Period Discriminator (MPD) / Multi-Scale Discriminator (MSD)

```

class DiscriminatorP(torch.nn.Module):
    def __init__(self, period, kernel_size=5, stride=3,
use_spectral_norm=False):
        super(DiscriminatorP, self).__init__()
        self.period = period
        norm_f = weight_norm if use_spectral_norm == False else spectral_norm
        self.convs = nn.ModuleList([
            norm_f(Conv2d(1, 32, (kernel_size, 1), (stride, 1),
padding=(get_padding(5, 1), 0))),
            norm_f(Conv2d(32, 128, (kernel_size, 1), (stride, 1),
padding=(get_padding(5, 1), 0))),
            norm_f(Conv2d(128, 512, (kernel_size, 1), (stride, 1),
padding=(get_padding(5, 1), 0))),
            norm_f(Conv2d(512, 1024, (kernel_size, 1), (stride, 1),
padding=(get_padding(5, 1), 0))),
            norm_f(Conv2d(1024, 1024, (kernel_size, 1), 1, padding=(2, 0))),
        ])
        self.conv_post = norm_f(Conv2d(1024, 1, (3, 1), 1, padding=(1, 0)))

    def forward(self, x):
        fmap = []

        # 1d to 2d
        b, c, t = x.shape
        if t % self.period != 0: # pad first
            n_pad = self.period - (t % self.period)
            x = F.pad(x, (0, n_pad), "reflect")
            t = t + n_pad
        x = x.view(b, c, t // self.period, self.period)

        for l in self.convs:
            x = l(x)
            x = F.leaky_relu(x, LRELU_SLOPE)
            fmap.append(x)
        x = self.conv_post(x)

        fmap.append(x)
        x = torch.flatten(x, 1, -1)

        return x, fmap

```

```

class DiscriminatorS(torch.nn.Module):
    def __init__(self, use_spectral_norm=False):
        super(DiscriminatorS, self).__init__()
        norm_f = weight_norm if use_spectral_norm == False else spectral_norm
        self.convs = nn.ModuleList([
            norm_f(Conv1d(1, 128, 15, 1, padding=7)),
            norm_f(Conv1d(128, 128, 41, 2, groups=4, padding=20)),
            norm_f(Conv1d(128, 256, 41, 2, groups=16, padding=20)),
            norm_f(Conv1d(256, 512, 41, 4, groups=16, padding=20)),
            norm_f(Conv1d(512, 1024, 41, 4, groups=16, padding=20)),
            norm_f(Conv1d(1024, 1024, 41, 1, groups=16, padding=20)),
            norm_f(Conv1d(1024, 1024, 5, 1, padding=2))
        ])
        self.conv_post = norm_f(Conv1d(1024, 1, 3, 1, padding=1))

    def forward(self, x):
        fmap = []
        for l in self.convs:
            x = l(x)
            x = F.leaky_relu(x, LRELU_SLOPE)
            fmap.append(x)
        x = self.conv_post(x)
        fmap.append(x)
        x = torch.flatten(x, 1, -1)

        return x, fmap

```

HiFiGAN – функции потерь

$$\mathcal{L}_{FM}(G; D) = \mathbb{E}_{(x,s)} \left[\sum_{i=1}^T \frac{1}{N_i} \|D^i(x) - D^i(G(s))\|_1 \right] \quad \mathcal{L}_{Mel}(G) = \mathbb{E}_{(x,s)} \left[\|\phi(x) - \phi(G(s))\|_1 \right]$$

$$\mathcal{L}_{Adv}(D; G) = \mathbb{E}_{(x,s)} \left[(D(x) - 1)^2 + (D(G(s)))^2 \right]$$

$$\mathcal{L}_{Adv}(G; D) = \mathbb{E}_s \left[(D(G(s)) - 1)^2 \right]$$

$$\mathcal{L}_G = \sum_{k=1}^K \left[\mathcal{L}_{Adv}(G; D_k) + \lambda_{fm} \mathcal{L}_{FM}(G; D_k) \right] + \lambda_{mel} \mathcal{L}_{Mel}(G)$$

$$\mathcal{L}_D = \sum_{k=1}^K \mathcal{L}_{Adv}(D_k; G)$$

HiFiGAN – качество

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 ($\times 0.003$)	24.73
WaveGlow	3.81 (± 0.08)	4.72 ($\times 0.21$)	501 ($\times 22.75$)	87.73
MelGAN	3.79 (± 0.09)	145.52 ($\times 6.59$)	14,238 ($\times 645.73$)	4.26
HiFi-GAN V1	4.36 (± 0.07)	31.74 ($\times 1.43$)	3,701 ($\times 167.86$)	13.92
HiFi-GAN V2	4.23 (± 0.07)	214.97 ($\times 9.74$)	16,863 ($\times 764.80$)	0.92
HiFi-GAN V3	4.05 (± 0.08)	296.38 ($\times 13.44$)	26,169 ($\times 1,186.80$)	1.46

**MeIGAN не имел MPD и мел-спектрограммного лосса,
hinge loss вместо MSE, но генератор имел гораздо больше параметров**

V1-V3 – разные конфигурации

HiFiGAN – Ablation study

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)

При отказе от MRF оставляется блок с наибольшим dilation и kernel size, остальные удаляются.

Дискриминатор содержит в 20 раз больше параметров, чем генератор.

HiFiGAN – изучение

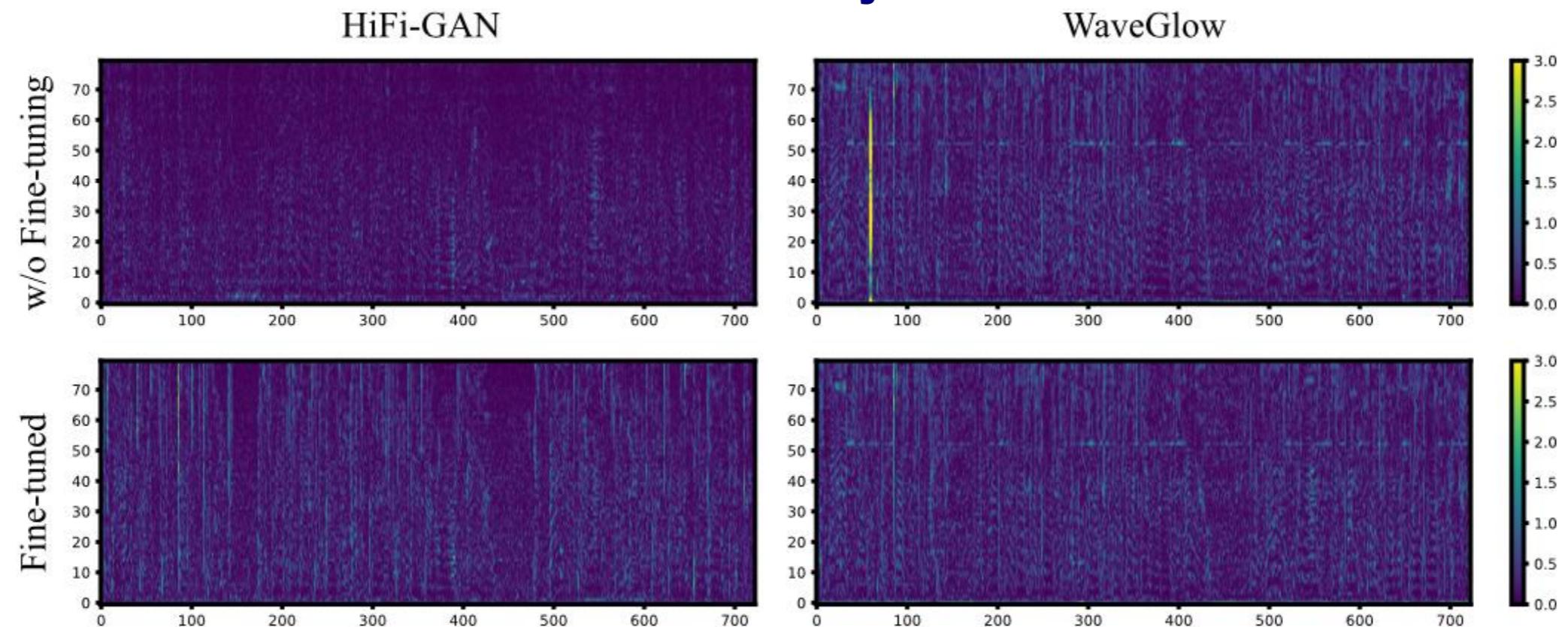


Figure 3: Pixel-wise difference in the mel-spectrogram domain between generated waveforms and a mel-spectrogram from Tacotron2. Before fine-tuning, HiFi-GAN generates waveforms corresponding to input conditions accurately. After fine-tuning, the error of the mel-spectrogram level increased, but the perceptual quality increased.

попробовали дотюнить HiFiGAN и WaveGlow

первый становится ещё лучше по MOS,

но интересно, что попиксельная разница у него становится хуже...

Какие ещё GAN-вокодеры есть

Table 5: Several representative GAN based vocoders and their characteristics.

GAN	Generator	Discriminator	Loss
WaveGAN [66]	DCGAN [281]	/	WGAN-GP [94]
GAN-TTS [23]	/	Random Window D	Hinge-Loss GAN [194]
MelGAN [174]	/	Multi-Scale D	LS-GAN [225] Feature Matching Loss [178]
Par.WaveGAN [396]	WaveNet [248]	/	LS-GAN, Multi-STFT Loss
HiFi-GAN [170]	Multi-Receptive Field Fusion	Multi-Period D, Multi-Scale D	LS-GAN, STFT Loss, Feature Matching Loss
VocGAN [401]	Multi-Scale G	Hierarchical D	LS-GAN, Multi-STFT Loss, Feature Matching Loss
GED [93]	/	Random Window D	Hinge-Loss GAN, Repulsive loss

часто **dilated convolution**, работа на разных масштабах

дискриминаторы: на случайному окне, на разных масштабах (MelGAN), Multi-period (HiFi-GAN) – [T] → [p, T/p] + 2D-conv

есть специфические ошибки: STFT loss [10,395], feature matching loss [178]

Какие ещё GAN-вокодеры есть

Table 6: Some characteristics of several representative generative models used in vocoders.

Generative Model	AR	VAE	Flow/AR	Flow/Bipartite	Diffusion	GAN
Vocoder (e.g.)	WaveNet	WaveVAE	Par.WaveNet	FloWaveNet	DiffWave	MelGAN
Simple	Y	N	N	N	N	N
Parallel	N	Y	Y	Y	Y	Y
Latent Manipulate	N	Y	Y	Y	Y	Y*
Likelihood Estimate	Y	Y	Y	Y	Y	N

**Дальше будем про акустические модели:
получают по языковым признакам → аудио (чаще мел-спектrogramму)**

сейчас обзор подходов...

проблема: разная длина входа и выхода

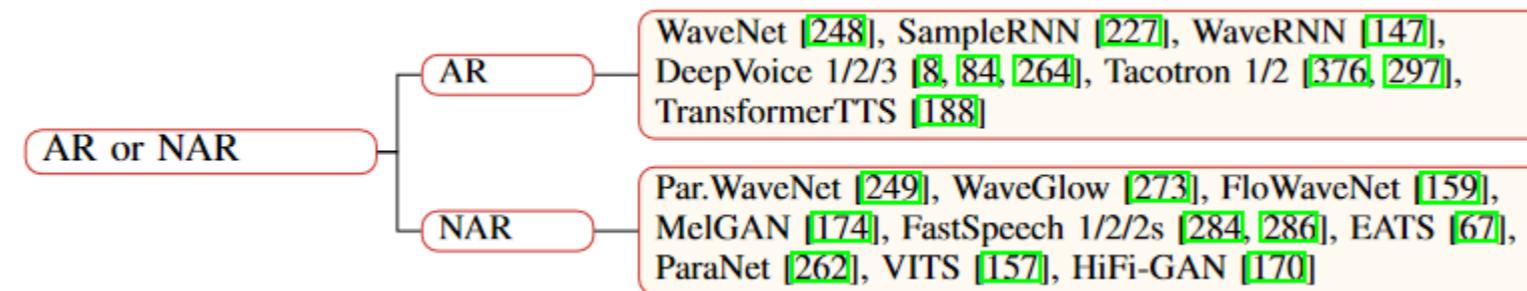
нужно соответствие (alignment)

- авторегрессионно порождать ответ**
- предсказывать продолжительности фонем (alignment)**

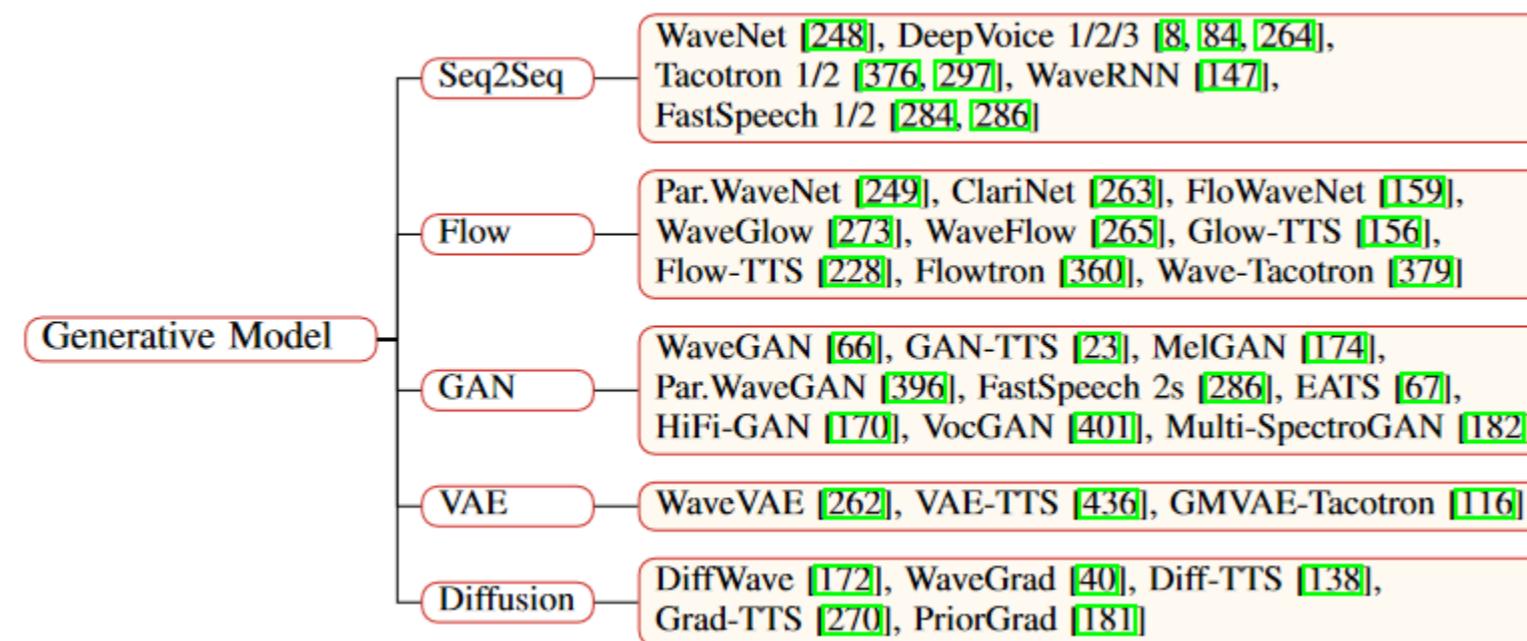
Table 2: A list of acoustic models and their corresponding characteristics. “Ling” stands for linguistic features, “Ch” stands for character, “Ph” stands for phoneme, “MCC” stands for mel-cepstral coefficients [79], “MGC” stands for mel-generalized coefficients [349], “BAP” stands for band aperiodicities [153, 154], “LSP” stands for line spectral pairs [132], “LinS” stands for linear-spectrograms, and “MelS” stands for mel-spectrograms. “NAR*” means the model uses autoregressive structures upon non-autoregressive structures and is not fully parallel.

Acoustic Model	Input→Output	AR/NAR	Modeling	Structure
HMM-based [409, 350]	Ling→MCC+F0	/	/	HMM
DNN-based [419]	Ling→MCC+BAP+F0	NAR	/	DNN
LSTM-based [76]	Ling→LSP+F0	AR	/	RNN
EMPHASIS [187]	Ling→LinS+CAP+F0	AR	/	Hybrid
ARST [369]	Ph→LSP+BAP+F0	AR	Seq2Seq	RNN
VoiceLoop [327]	Ph→MGC+BAP+F0	AR	/	hybrid
Tacotron [376]	Ch→LinS	AR	Seq2Seq	Hybrid/RNN
Tacotron 2 [297]	Ch→MelS	AR	Seq2Seq	RNN
DurIAN [411]	Ph→MelS	AR	Seq2Seq	RNN
Non-Att Tacotron [298]	Ph→MelS	AR	/	Hybrid/CNN/RNN
Para. Tacotron 1/2 [72, 73]	Ph→MelS	NAR	/	Hybrid/Self-Att/CNN
MelNet [361]	Ch→MelS	AR	/	RNN
DeepVoice [8]	Ch/Ph→MelS	AR	/	CNN
DeepVoice 2 [84]	Ch/Ph→MelS	AR	/	CNN
DeepVoice 3 [264]	Ch/Ph→MelS	AR	Seq2Seq	CNN
ParaNet [262]	Ph→MelS	NAR	Seq2Seq	CNN
DCTTS [326]	Ch→MelS	AR	Seq2Seq	CNN
SpeedySpeech [355]	Ph→MelS	NAR	/	CNN
TalkNet 1/2 [19, 18]	Ch→MelS	NAR	/	CNN
Transformer/TTS [188]	Ph→MelS	AR	Seq2Seq	Self-Att
MultiSpeech [38]	Ph→MelS	AR	Seq2Seq	Self-Att
FastSpeech 1/2 [284, 286]	Ph→MelS	NAR	Seq2Seq	Self-Att
AlignTTS [422]	Ch/Ph→MelS	NAR	Seq2Seq	Self-Att
JDIT [193]	Ph→MelS	NAR	Seq2Seq	Self-Att
FastPitch [177]	Ph→MelS	NAR	Seq2Seq	Self-Att
AdaSpeech 1/2/3 [39, 397, 398]	Ph→MelS	NAR	Seq2Seq	Self-Att
Denoispeech [427]	Ph→MelS	NAR	Seq2Seq	Self-Att
DeviceTTS [123]	Ph→MelS	NAR	/	Hybrid/DNN/RNN
LightSpeech [214]	Ph→MelS	NAR	/	Hybrid/Self-Att/CNN
Flow-TTS [228]	Ch/Ph→MelS	NAR*	Flow	Hybrid/CNN/RNN
Glow-TTS [156]	Ph→MelS	NAR	Flow	Hybrid/Self-Att/CNN
Flowtron [360]	Ph→MelS	AR	Flow	Hybrid/RNN
EfficientTTS [229]	Ch→MelS	NAR	Flow	Hybrid/CNN
GMVAE-Tacotron [116]	Ph→MelS	AR	VAE	Hybrid/RNN
VAE-TTS [436]	Ph→MelS	AR	VAE	Hybrid/RNN
BVAE-TTS [183]	Ph→MelS	NAR	VAE	CNN
GAN exposure [96]	Ph→MelS	AR	GAN	Hybrid/RNN
TTS-Stylization [218]	Ch→MelS	AR	GAN	Hybrid/RNN
Multi-SpectroGAN [182]	Ph→MelS	NAR	GAN	Hybrid/Self-Att/CNN
Diff-TTS [138]	Ph→MelS	NAR*	Diffusion	Hybrid/CNN
Grad-TTS [270]	Ph→MelS	NAR	Diffusion	Hybrid/Self-Att/CNN
PriorGrad [181]	Ph→MelS	NAR	Diffusion	Hybrid/Self-Att/CNN

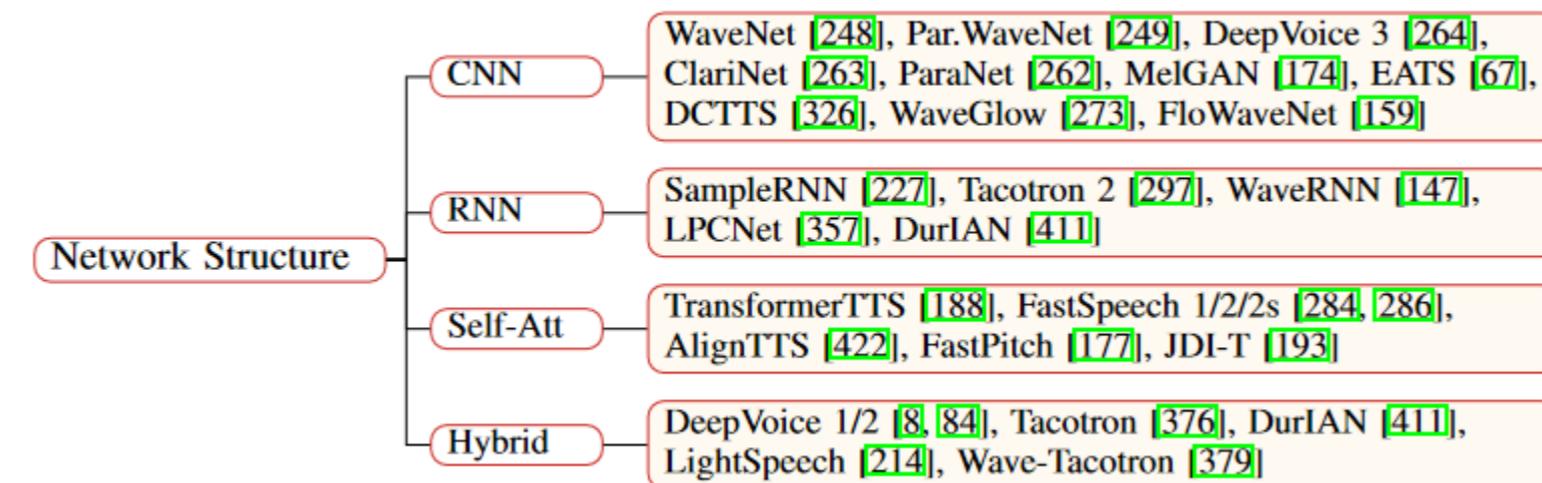
Немного про термины: авторегрессионные и неавторегрессионные



Немного про термины: какая генеративная модель в основе



Немного про термины: какая структура сети



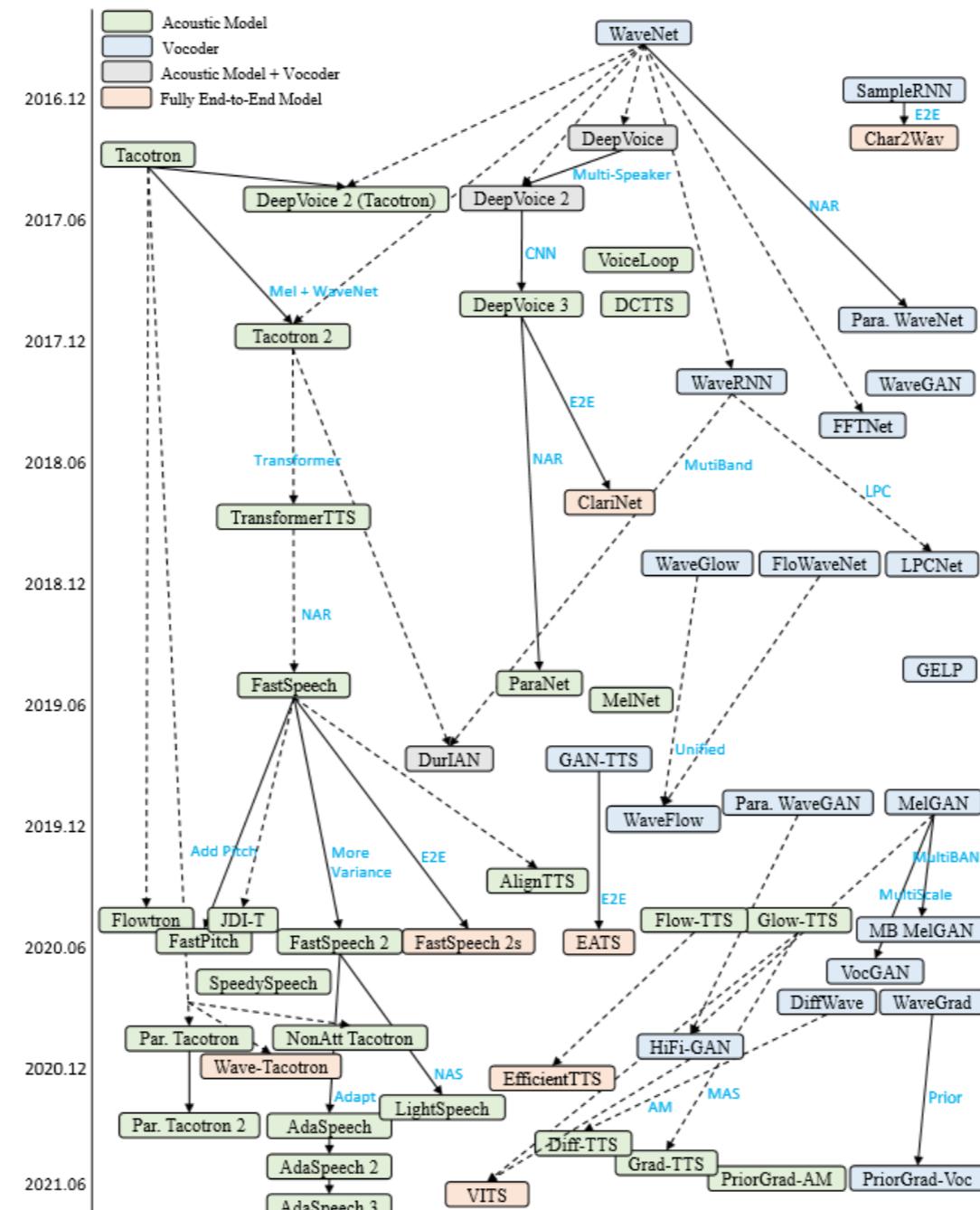


Figure 6: The evolution of neural TTS models.

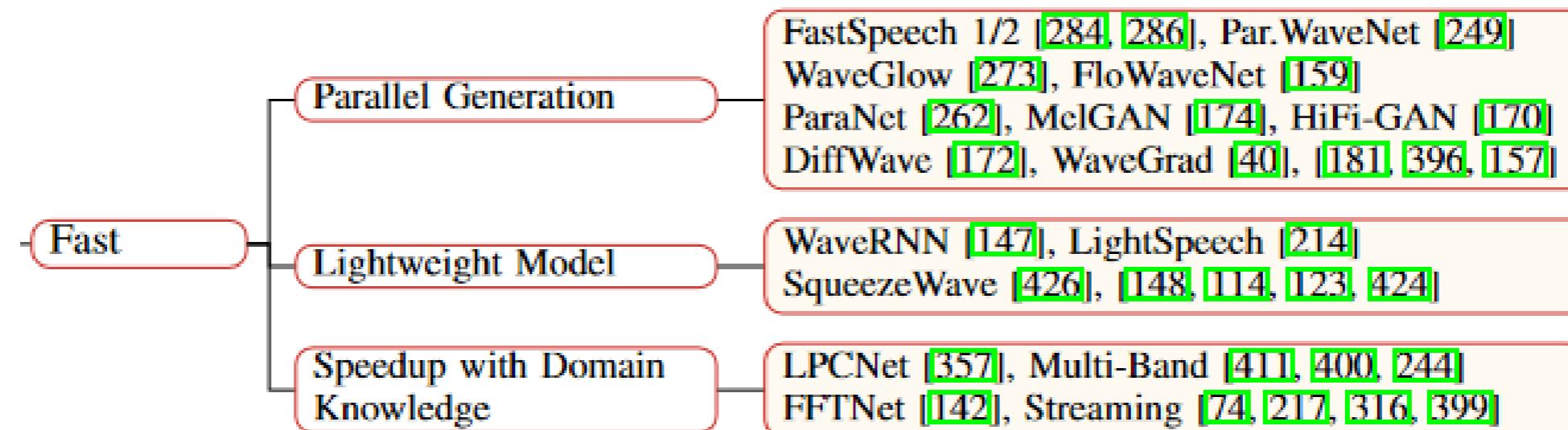
Методы увеличения скорости синтеза (Fast TTS)

- неавторегрессионный синтез
- более «лёгкая» и эффективная сеть

pruning, quantization, knowledge distillation, neural architecture search, ...

- учёт предметной области (DomainKnowledge) ???

linear prediction, multiband modeling, subscale prediction, multi-frame prediction, streaming synthesis, etc.



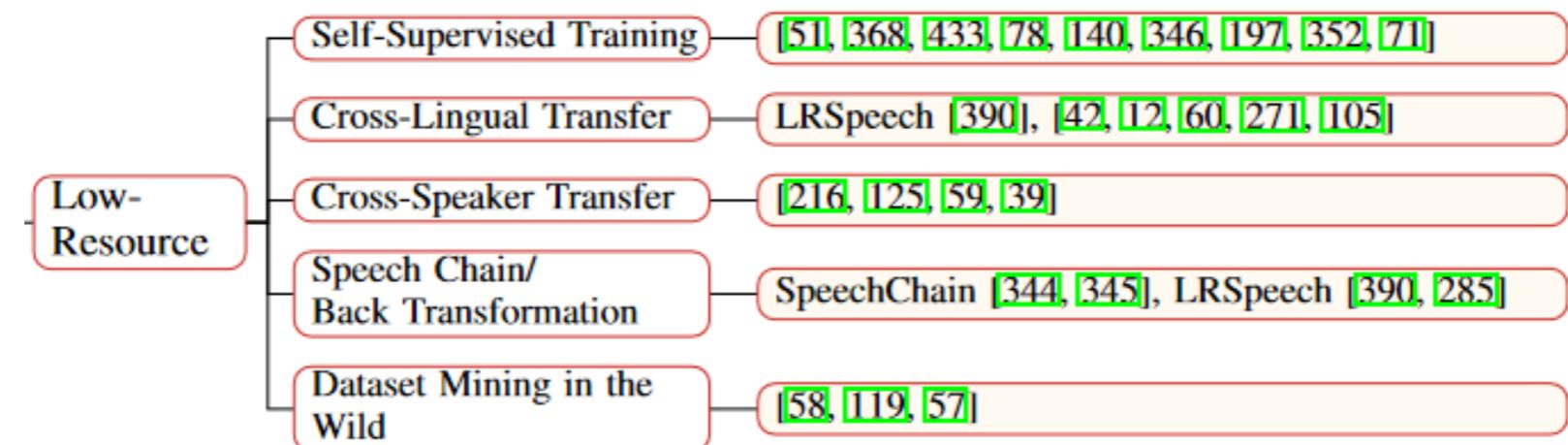
Методы увеличения скорости синтеза (Fast TTS)

к вопросу о неавторегрессионности

Table 8: The time complexity of different TTS models in training and inference with regard to sequence length N . T is the number of steps/iterations in flow/diffusion based models.

Modeling Paradigm	TTS Model	Training	Inference
AR (RNN)	Tacotron 1/2, SampleRNN, LPCNet	$\mathcal{O}(N)$	$\mathcal{O}(N)$
AR (CNN/Self-Att)	DeepVoice 3, TransformerTTS, WaveNet	$\mathcal{O}(1)$	$\mathcal{O}(N)$
NAR (CNN/Self-Att)	FastSpeech 1/2, ParaNet	$\mathcal{O}(1)$	$\mathcal{O}(1)$
NAR (GAN/VAE)	MelGAN, HiFi-GAN, FastSpeech 2s, EATS	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Flow (AR)	Par. WaveNet, ClariNet, Flowtron	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Flow (Bipartite)	WaveGlow, FloWaveNet, Glow-TTS	$\mathcal{O}(T)$	$\mathcal{O}(T)$
Diffusion	DiffWave, WaveGrad, Grad-TTS, PriorGrad	$\mathcal{O}(T)$	$\mathcal{O}(T)$

Low-Resource TTS



когда мало данных (на этом языке, для этого спикера и т.п.)

Self-supervised training

- используем BERT в text encoder
- speech decoder преодобучаем по аналогии с LM предсказывать мел-спектрограмму
 - speech decoder обучаем вместе с задачей конверсии голоса **[433]**
- звук квантуется в дискретную последовательность (некий универсальный алфавит)
[197, 352, 429]

Cross-lingual transfer

Если делать TTS: фонемы → речь,
то можно предобучить на «богатом языке»

Иногда строят словари соответствия фонем на разных языках

Для построения систем на многих языках:

- International phonetic alphabet (IPA) [106]
 - byte representation [105]

Cross-speaker transfer

учим синтезировать конкретный один голос + конверсия в другой [125]
voice adaptation / voice cloning [43, 39]

Speech chain/Back transformation

пользуемся тем, что TTS и STT обратные задачи

Speech chain [344,345] and back transformation [285,390]

Dataset mining in the wild

используем доступные данные + технику для их улучшения

speech enhancement [356], denoising [427], disentangling [377,117]

DeepVoice: Real-time Neural TTS

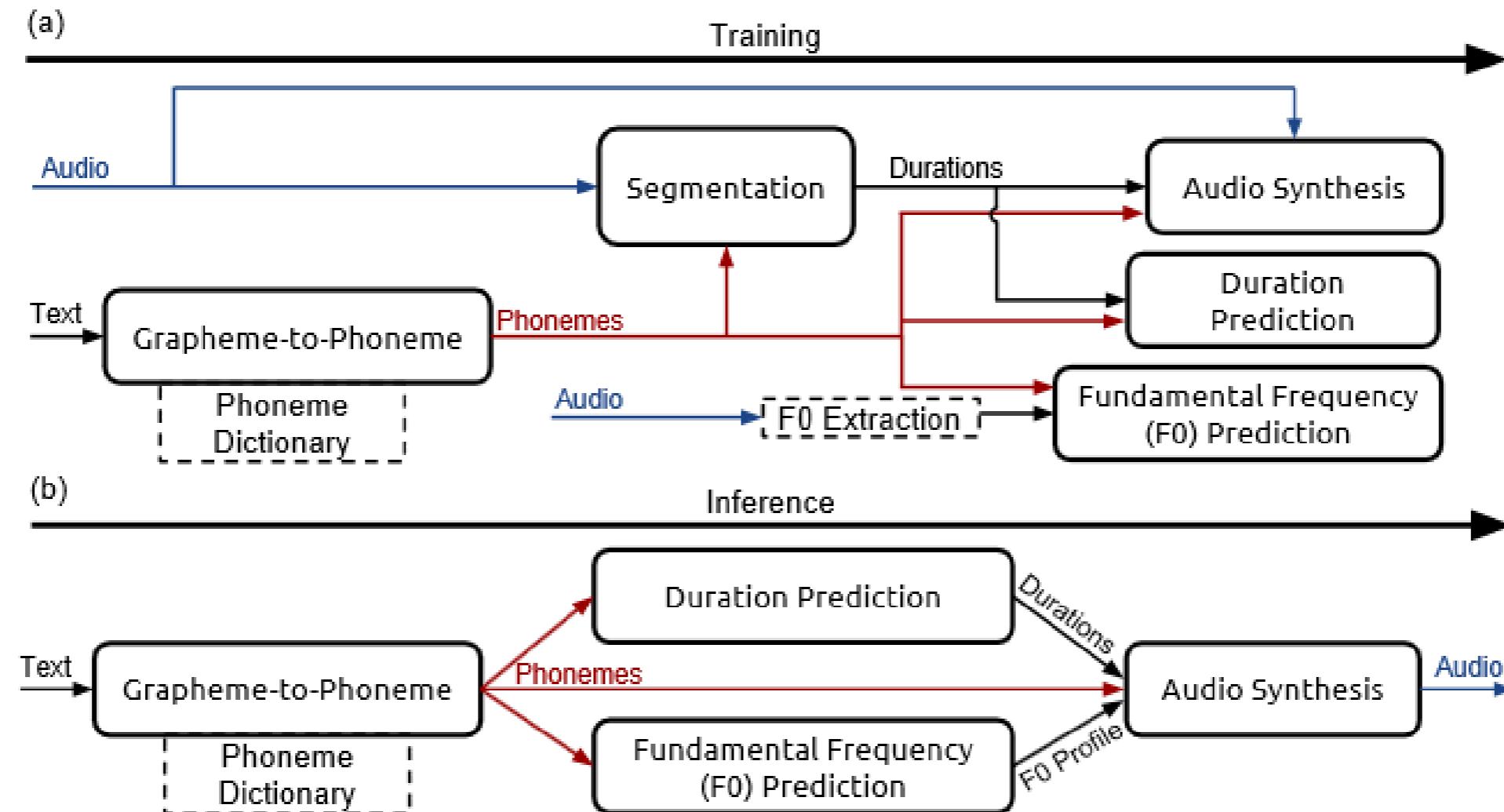
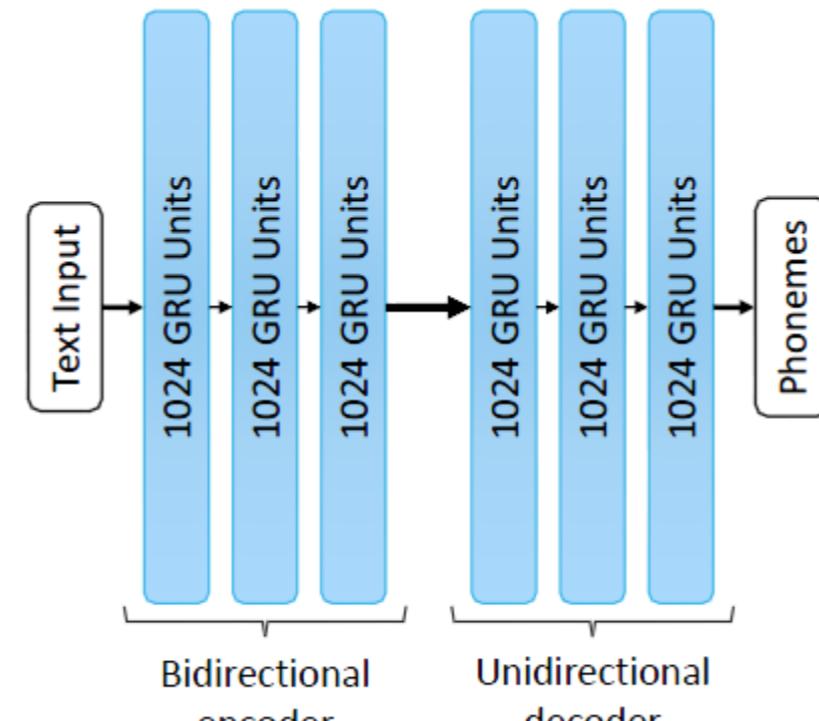


Figure 1. System diagram depicting (a) training procedure and (b) inference procedure, with inputs on the left and outputs on the right. In our system, the duration prediction model and the F0 prediction model are performed by a single neural network trained with a joint loss. The grapheme-to-phoneme model is used as a fallback for words that are not present in a phoneme dictionary, such as CMUDict. Dotted lines denote non-learned components.

DeepVoice: grapheme-to-phoneme model

- переводит текст в фонемы (например, используя фонетический словарь ARPABET),
построен по архитектуре encoder-decoder (3+3, 1024hu, многослойные bi-GRU)



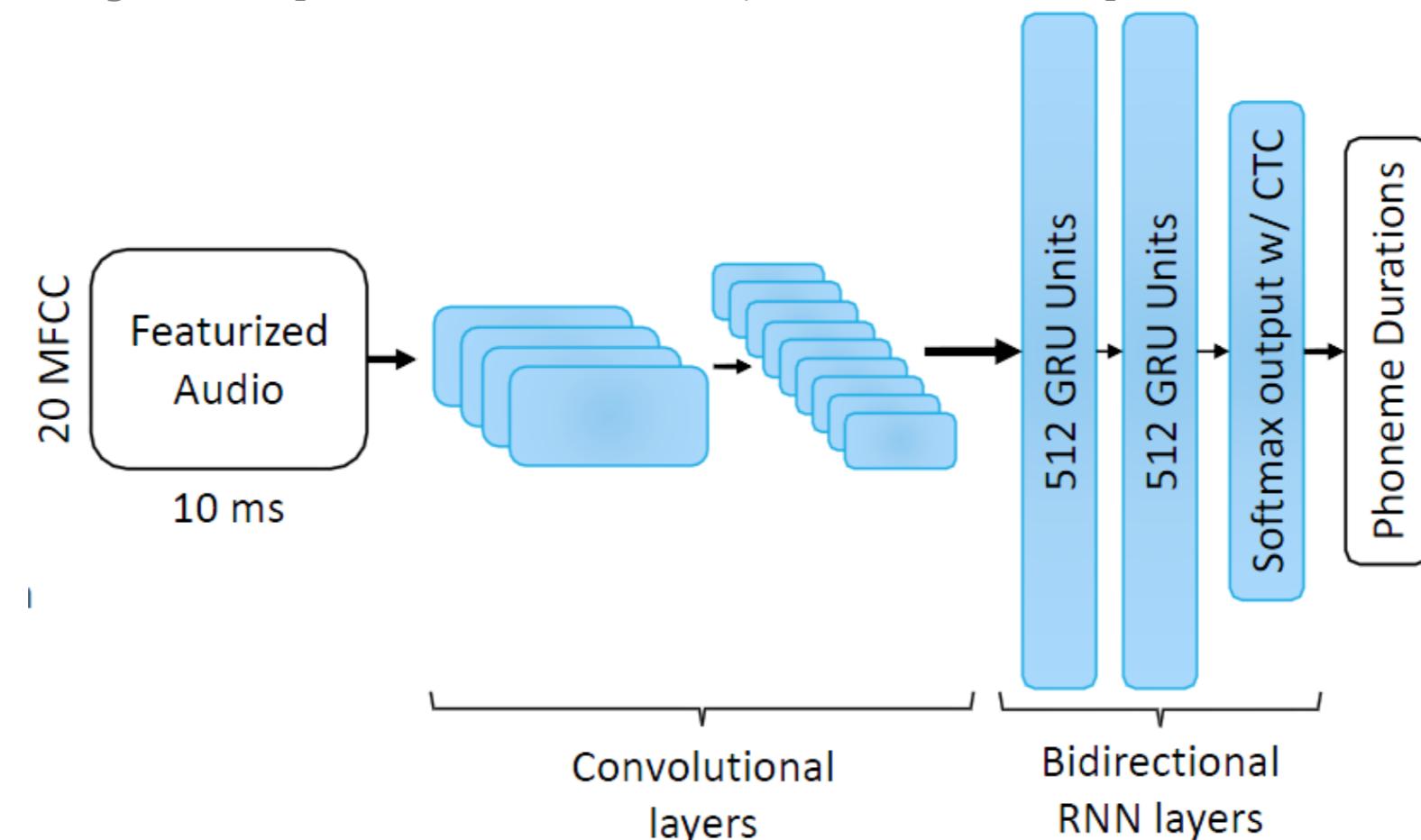
**teacher forcing, drop out
beam search width=5**

**Вместо hand-engineered features предлагается использовать
лишь фонемы и F0, а всю остальную работу переложить на NN**

DeepVoice: segmentation model

– находить начало и конец фонемы на аудио (нужна для обучения),

свёрточная рекуррентная сеть, обученная (с использованием CTC-loss) на выравнивание аудио (представленное 20-MFCC) с последовательностью фонем (на самом деле, пар фонем – так лучше предсказывается, когда одна фонема меняет другую)

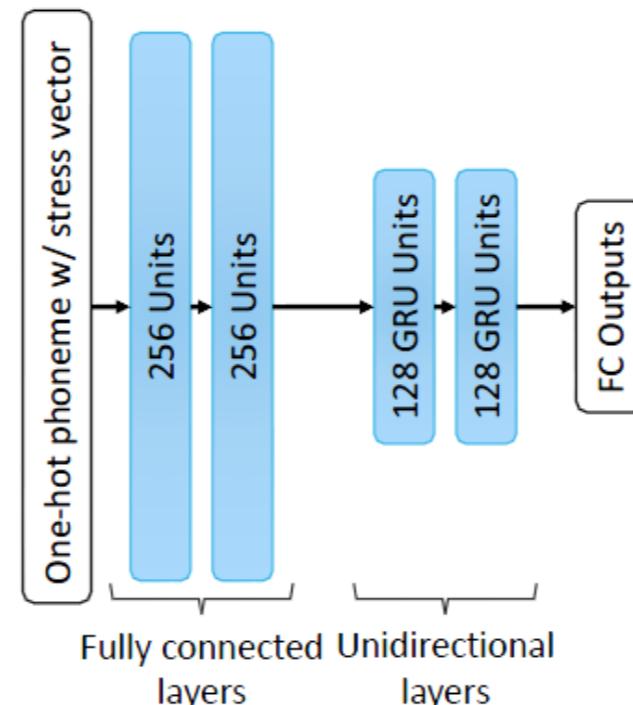


2D свёртки во времени и частоте

DeepVoice: phoneme duration model / fundamental frequency model

– предсказывать длительность фонемы / F0 для фонемы

совмещены в одну: 2 FC + 2 uni-GRU



вход – последовательность фонем (с ударениями)

выход –

- 1. длительность фонемы,**
 - 2. вероятность, что фонема произносится,**
 - 3. 20 F0-частот распределенных по всей длительности произношения фонемы.**
- ошибка = л/к 3х ошибок**

Deep Voice 2: Multi-Speaker Neural Text-to-Speech = Deep Voice + Multi-Speaker (speaker embedding)

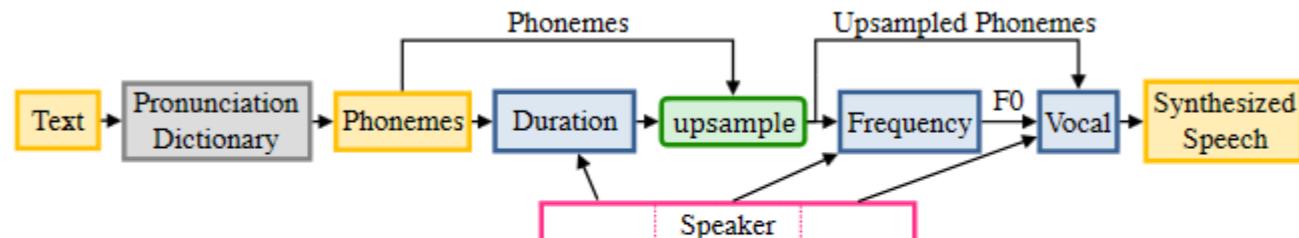


Figure 1: Inference system diagram: first text-phonemes dictionary conversion, second predict phoneme durations, third upsample and generate F_0 , finally feed F_0 and phonemes to vocal model.

Отличия от DeepVoice:
Разделили phoneme duration и frequency модели
+ batch normalization, residual connections в CNN

Sercan Arik, Gregory Diamos, Andrew Gibiansky, John Miller, Kainan Peng, Wei Ping, Jonathan Raiman, Yanqi Zhou
«Deep Voice 2: Multi-Speaker Neural Text-to-Speech» <https://arxiv.org/abs/1705.08947>

Deep Voice 2: Multi-Speaker Neural Text-to-Speech

добавление спикера: при обучении выучаются представления спикеров

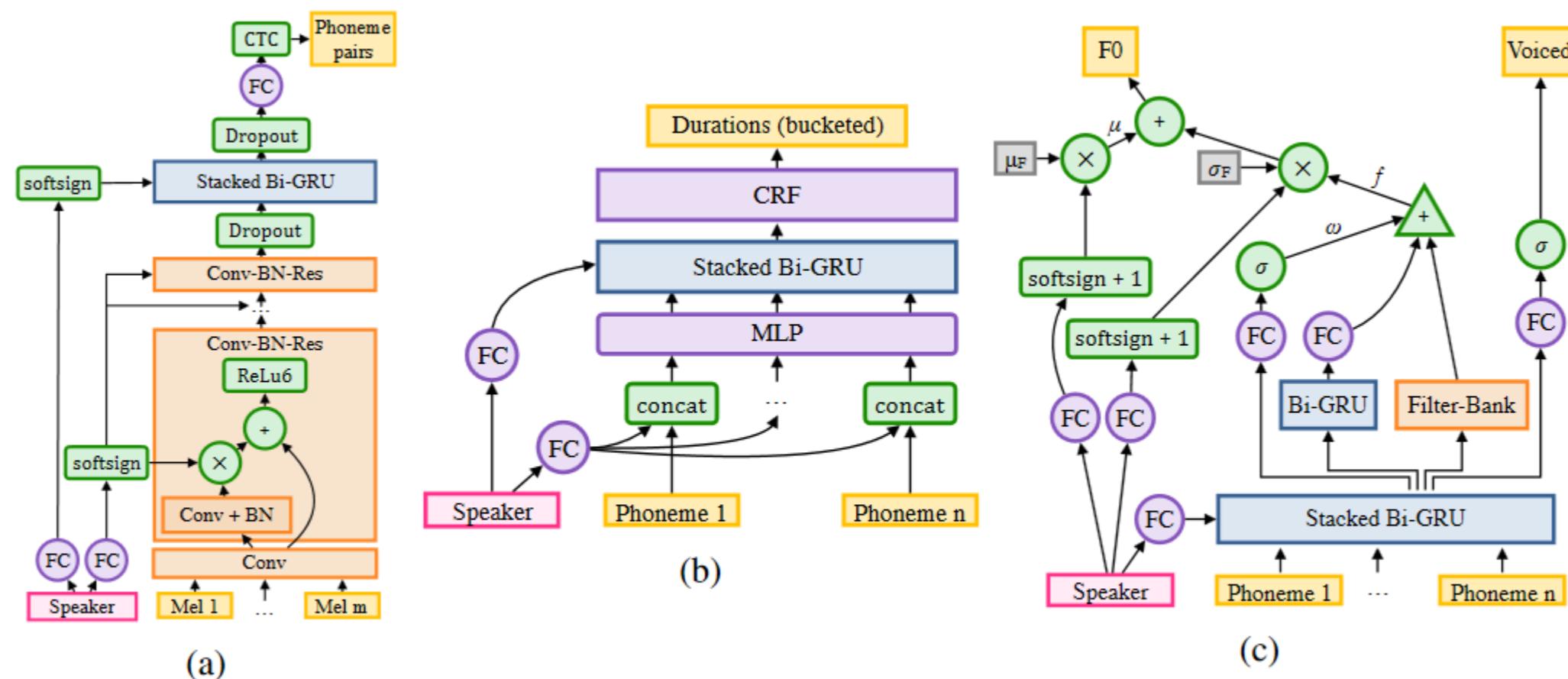


Figure 2: Architecture for the multi-speaker (a) segmentation, (b) duration, and (c) frequency model.

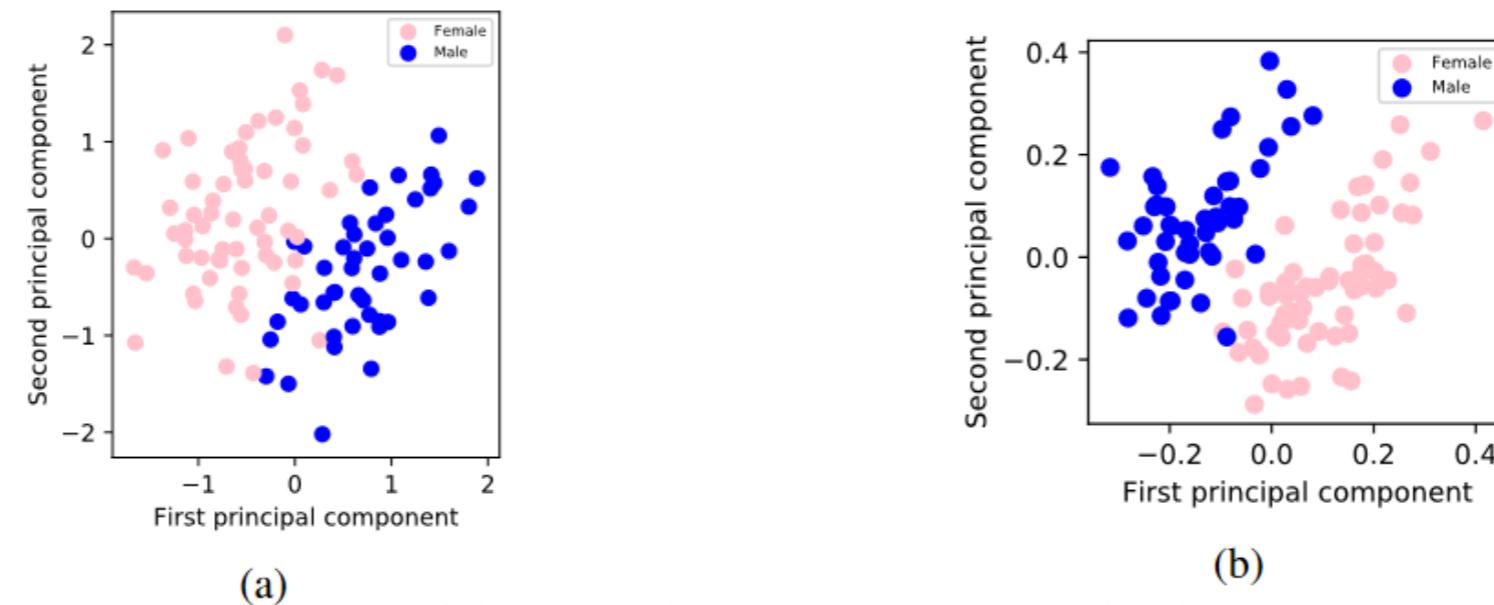


Figure 4: Principal components of the learned speaker embeddings for the (a) 80-layer vocal model and (b) character-to-spectrogram model for VCTK dataset. See Appendix D.3 for details.

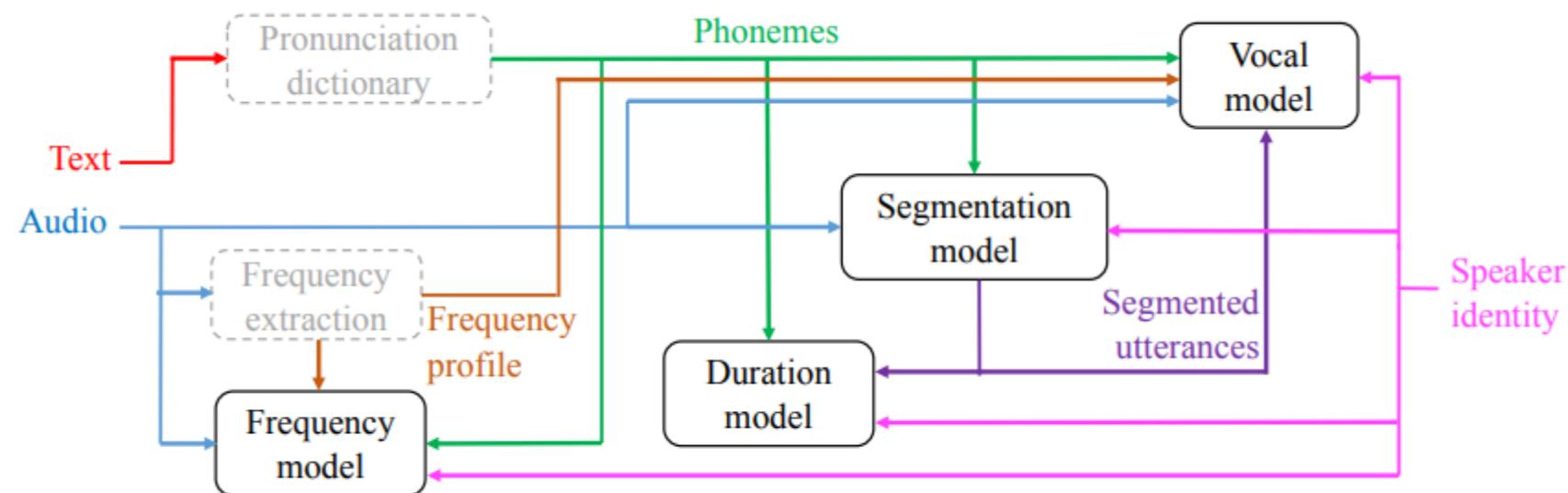


Figure 5: System diagram for training procedure for Deep Voice 2.

TTS: Tacotron (Google)

end2end (в некотором смысле)

RNN-based

seq2seq + attention

encoder + decoder + post-processing net

end-to-end text-to-speech (в DeepVoice было нечестно – разные модели)

не требует какого-то конкретного вокодера (на выходе – спектrogramма)

**применяется attention механизм для генерации mel-спектrogramмы,
которая затем подается на вход postprocessing network для получения линейной
спектrogramмы**

Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio и др., «Tacotron: Towards End-to-End Speech Synthesis» //
<https://arxiv.org/pdf/1703.10135.pdf>

Tacotron

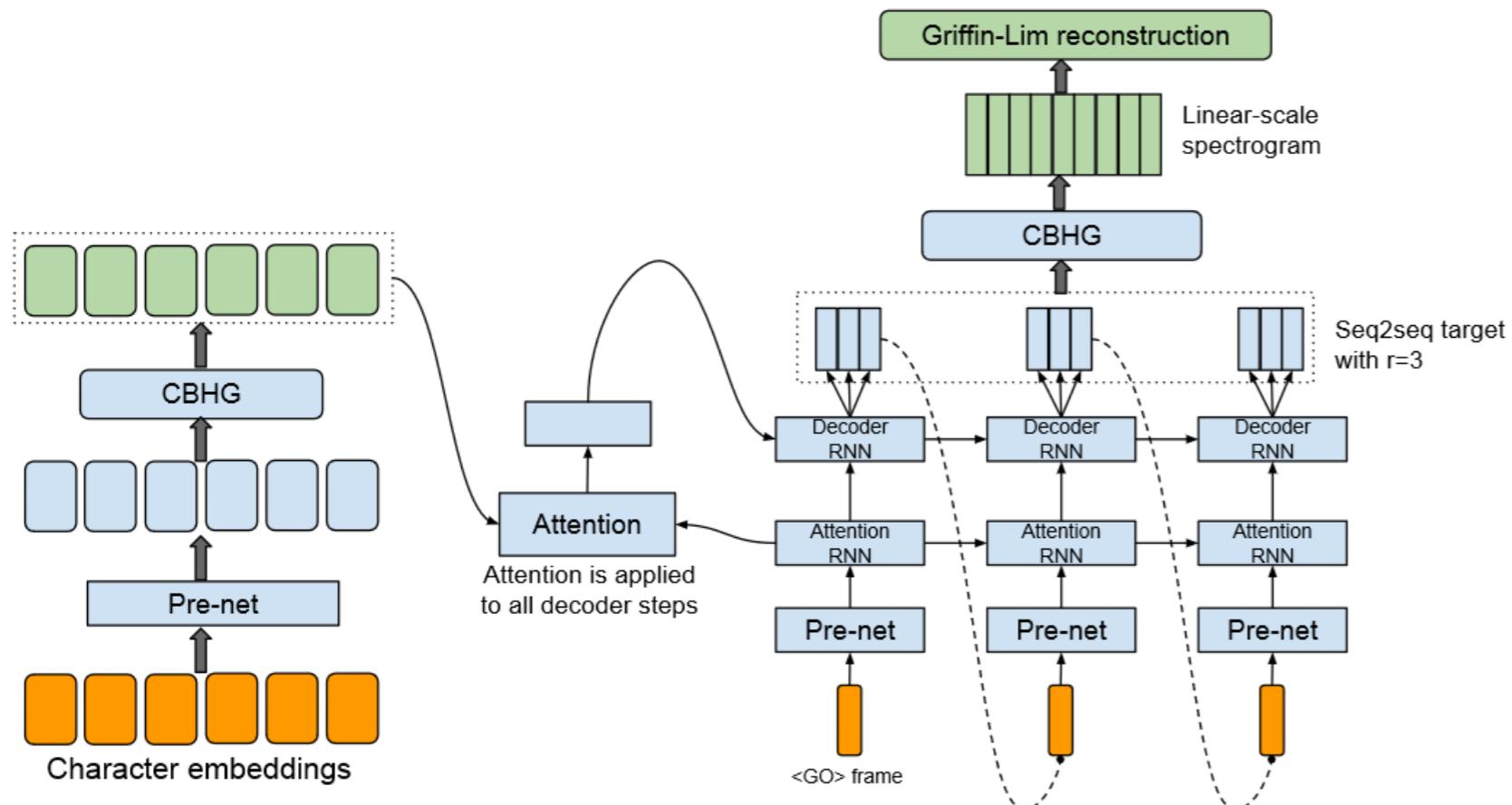


Figure 1: Model architecture. The model takes characters as input and outputs the corresponding raw spectrogram, which is then fed to the Griffin-Lim reconstruction algorithm to synthesize speech.

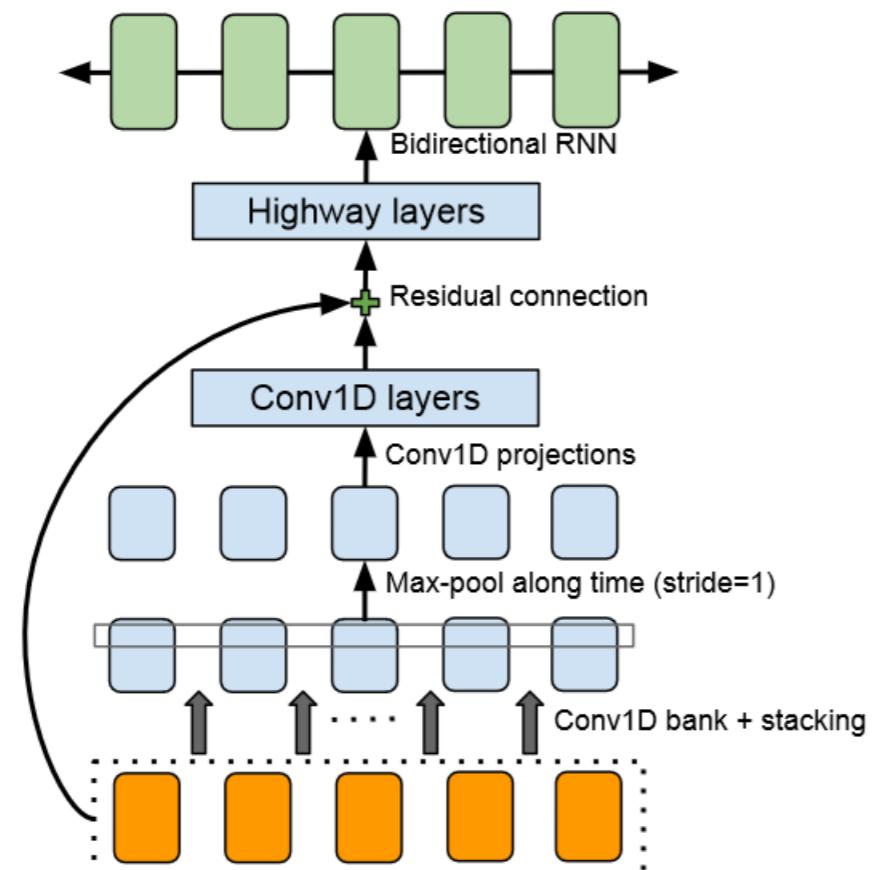
Pre-Net

символы в ОНЕ

2 Dense-ReLU layers с 50%
dropout
bottleneck layer

ВХОД: СИМВОЛЫ, ВЫХОД: linear-spectrogram

Tacotron: CBHG (Convolutional Bank + Highway network + GRU)



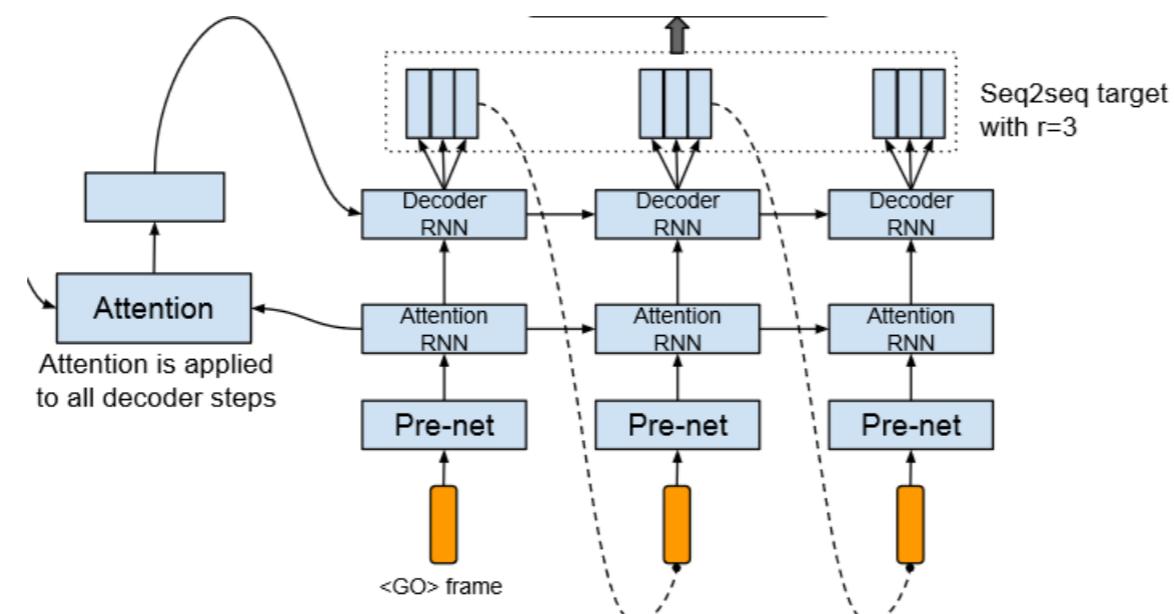
**16 свёрточных слоёв
max-pool (ширина задаётся)
highway network
Bidirectional GRU**

**Для получения высокоуровневых
признаков текста и
последующего внимания**

**Вначале банк свёрток ширины
1,2,...,K (каждой ширины
несколько свёрток)**

Figure 2: The CBHG (1-D convolution bank + highway network + bidirectional GRU) module adapted from Lee et al. (2016).

Tacotron: Decoder



выход – 80-band mel-scale spectrogram

**Attention RNN (GRU cells) + 2 residual GRU cells + FC output layer
vertical residual connections**

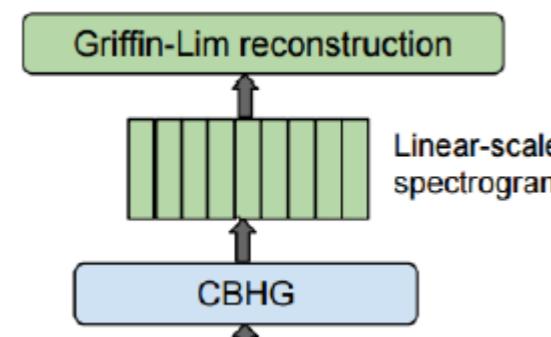
**предсказываем несколько неперекрывающихся фреймов
(сокращает время генерации, соседние фреймы коррелируют)**

Tacotron

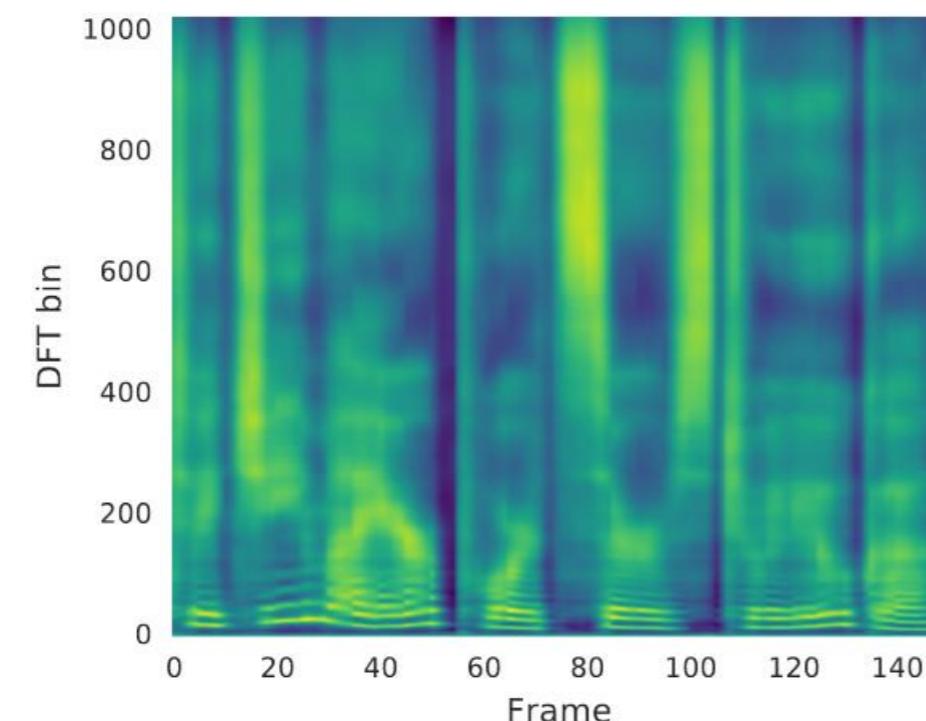
Table 1: Hyper-parameters and network architectures. “conv- k - c -ReLU” denotes 1-D convolution with width k and c output channels with ReLU activation. FC stands for fully-connected.

Spectral analysis	<i>pre-emphasis</i> : 0.97; <i>frame length</i> : 50 ms; <i>frame shift</i> : 12.5 ms; <i>window type</i> : Hann
Character embedding	256-D
Encoder CBHG	<i>Conv1D bank</i> : $K=16$, conv- k -128-ReLU <i>Max pooling</i> : stride=1, width=2 <i>Conv1D projections</i> : conv-3-128-ReLU → conv-3-128-Linear <i>Highway net</i> : 4 layers of FC-128-ReLU <i>Bidirectional GRU</i> : 128 cells
Encoder pre-net	FC-256-ReLU → Dropout(0.5) → FC-128-ReLU → Dropout(0.5)
Decoder pre-net	FC-256-ReLU → Dropout(0.5) → FC-128-ReLU → Dropout(0.5)
Decoder RNN	2-layer residual GRU (256 cells)
Attention RNN	1-layer GRU (256 cells)
Post-processing net CBHG	<i>Conv1D bank</i> : $K=8$, conv- k -128-ReLU <i>Max pooling</i> : stride=1, width=2 <i>Conv1D projections</i> : conv-3-256-ReLU → conv-3-80-Linear <i>Highway net</i> : 4 layers of FC-128-ReLU <i>Bidirectional GRU</i> : 128 cells
Reduction factor (r)	2

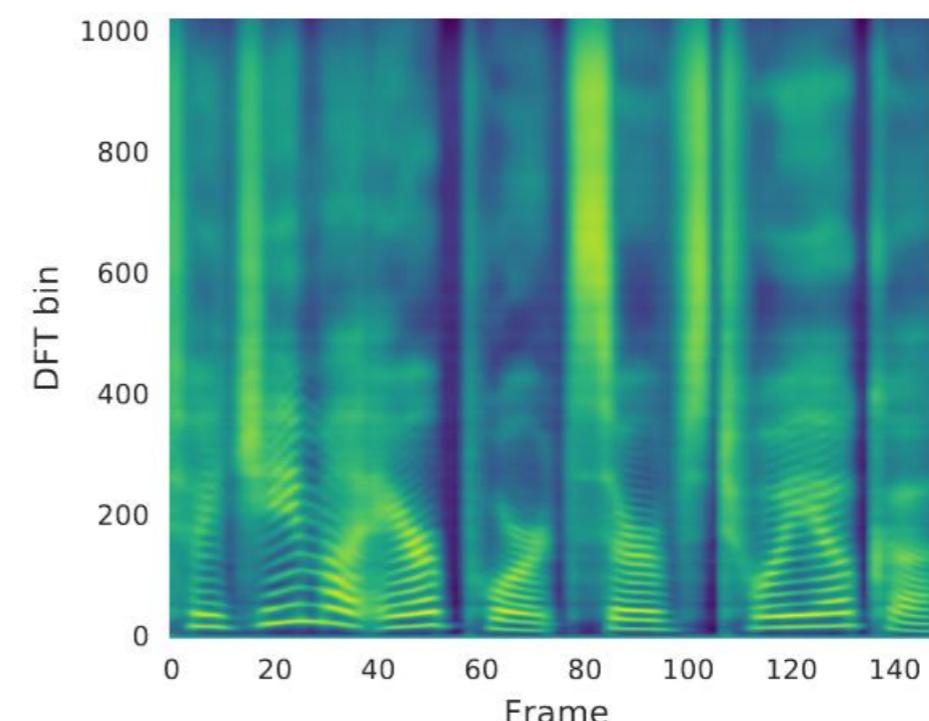
Tacotron: Post-processing net



**seq2seq target → waveform
CBHG + Griffin-Lim algorithm**



(a) Without post-processing net



(b) With post-processing net

Figure 4: Predicted spectrograms with and without using the post-processing net.

DeepVoice3

fully-convolutional network structure

нарямую предсказываем мел-спектрограмму

По качеству сопоставима Tacotron, но сходится в 10 раз быстрее (свертки вместо RNN).

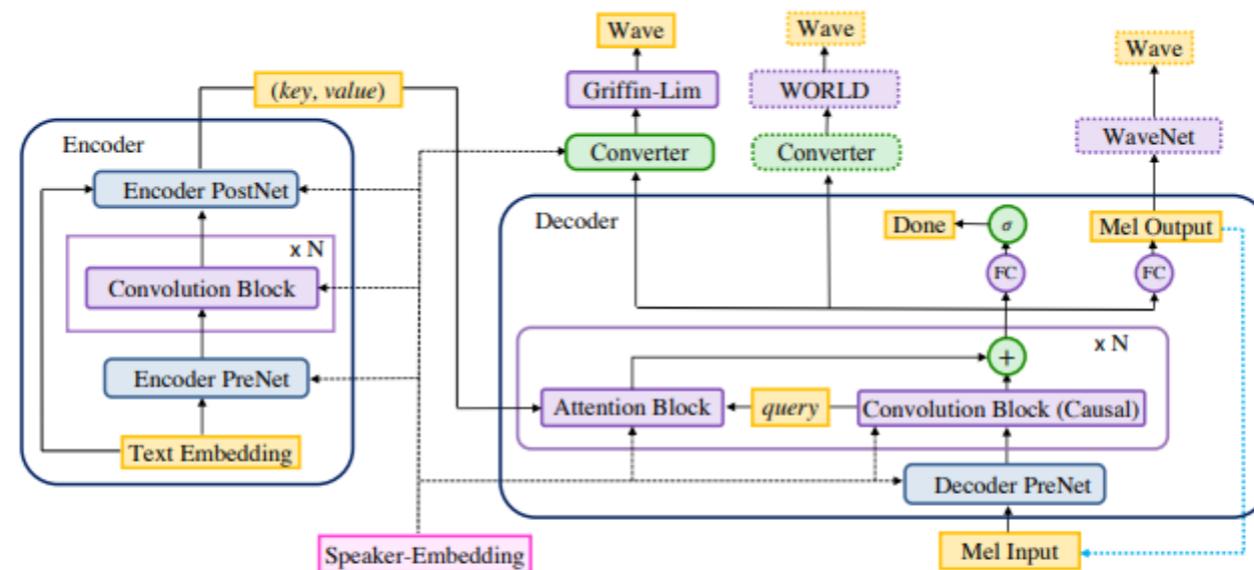


Figure 1: Deep Voice 3 uses residual convolutional layers to encode text into per-timestep *key* and *value* vectors for an attention-based decoder. The decoder uses these to predict the mel-scale log magnitude spectrograms that correspond to the output audio. (Light blue dotted arrows depict the autoregressive process during inference.) The hidden states of the decoder are then fed to a converter network to predict the vocoder parameters for waveform synthesis. See Appendix A for more details.

W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, J. Raiman и J. Miller «Deep voice 3: 2000-speaker neural text-to-speech» // arXiv:1710.07654, 2017

DeepVoice3

Encoder — свёрточный, текстовые признаки → внутренне представление

Decoder — свёрточный, условный, авторегрессионно декодирует внутреннее представление

Converter — постпроцессинг

Residual gated convolutions (вместо RNN)

Tacotron 2 = Tacotron-style model + modified WaveNet vocode

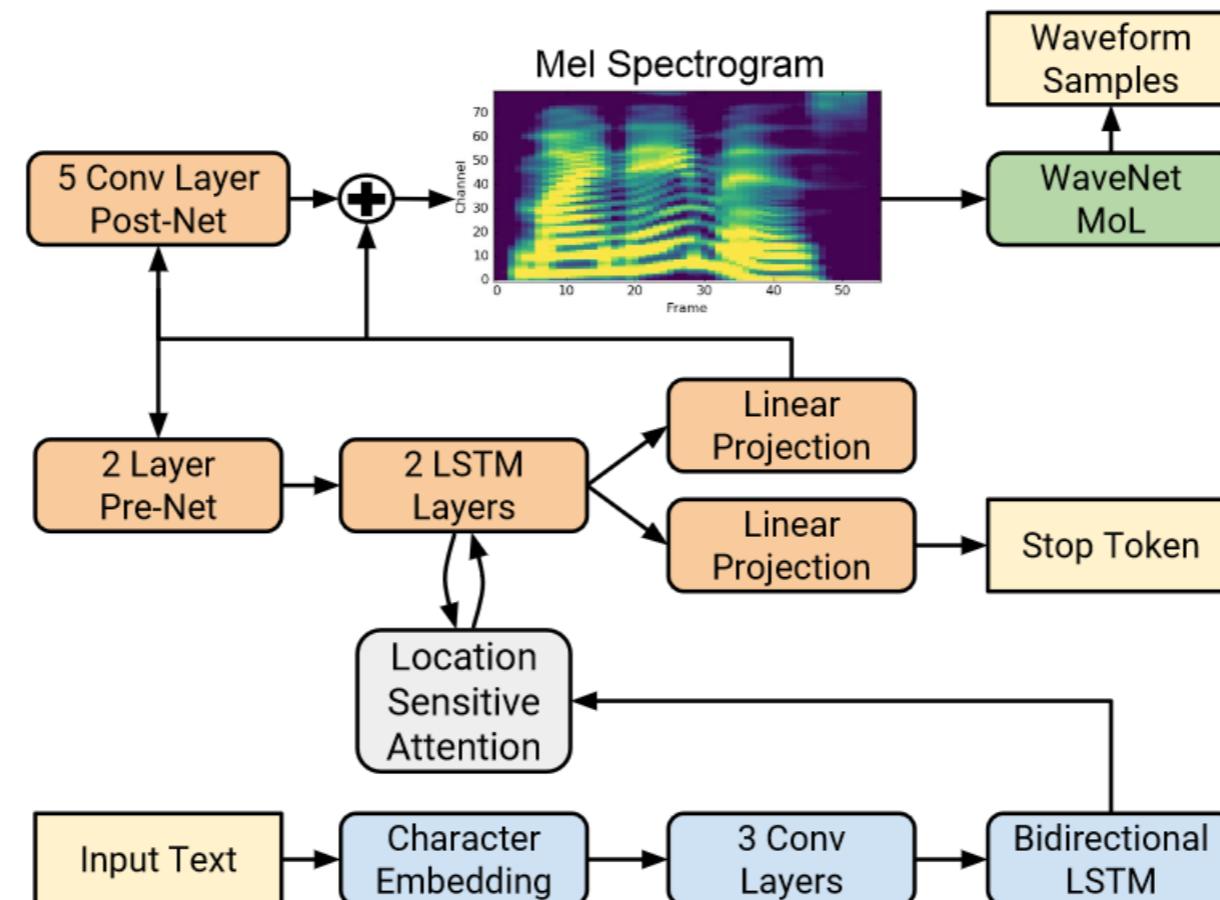


Fig. 1. Block diagram of the Tacotron 2 system architecture.

J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerry-Ryan и др. «Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions» // <https://arxiv.org/pdf/1712.05884.pdf>

Tacotron 2

**рекуррентная seq2seq сеть с attention механизмом,
post-processing net для mel-спектrogramмы,
wavenet вокодер (обучается отдельно) вместо Griffin-Lim
выход: mel-spectrogram вместо linear-spectrogram**

заменили CBHG+GRU → LSTM

Location Sensitive Attention [11]

**Декодер на каждом шаге авторегрессионно предсказывает мел спектrogramму
и остановку генерации**

ошибка =

- MSE между целевыми мелспектrogramмами и входами Post-net
- MSE между целевыми мелспектrogramмами и выходами Post-net
 - логлосс для предсказания стоп-токена

Tacotron 2

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)	4.526 ± 0.066

Table 1. Mean Opinion Score (MOS) evaluations with 95% confidence intervals computed from the t-distribution for various systems.

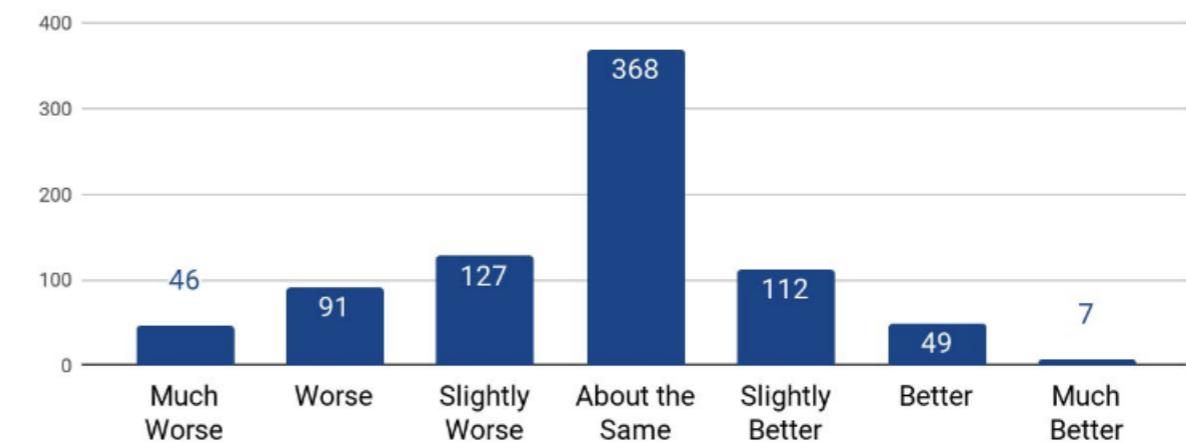
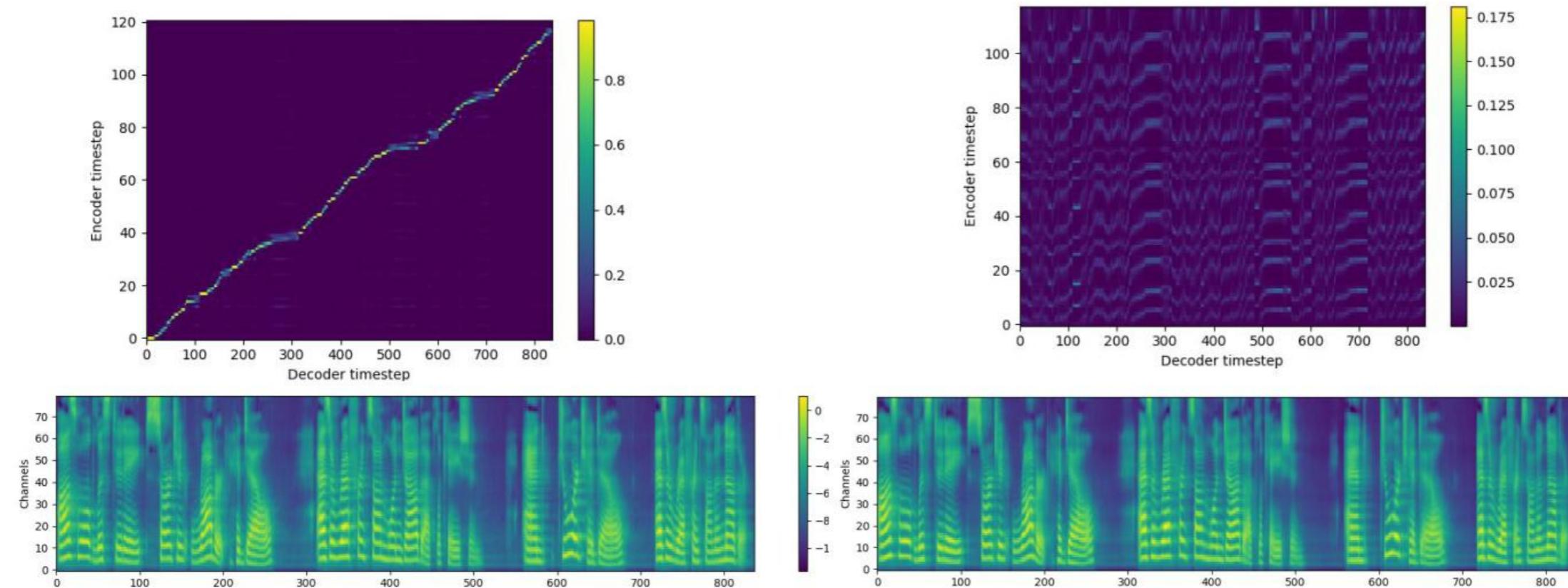


Fig. 2. Synthesized vs. ground truth: 800 ratings on 100 items.

«Неправильное» внимание



Mellotron

Расширение Tacotron для более эмоциональной и выразительной речи

**добавлены обучаемые эмбеддинги спикеров
добавили в модель Global Style Token (GST)**

– сеть, которая отображает мелспектrogramму аудио произвольной длины в вектор фиксированного размера, называемый вектором стиля

контроль над стилем через примеры

Rafael Valle, Jason Li, Ryan Prenger, Bryan Catanzaro «Mellotron: Multispeaker expressive voice synthesis by conditioning on rhythm, pitch and global style tokens»

<https://arxiv.org/abs/1910.11997>

к описанию Mellotron: Что такое GST (Global Style Token)

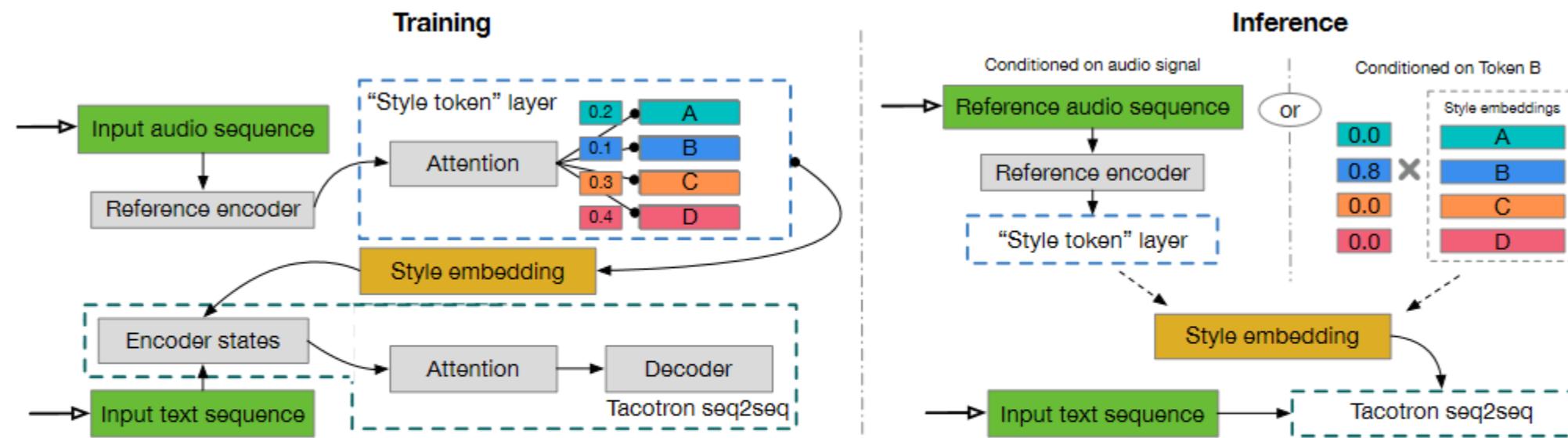


Figure 1. Model diagram. During **training**, the log-mel spectrogram of the training target is fed to the reference encoder followed by a style token layer. The resulting style embedding is used to condition the Tacotron text encoder states. During **inference**, we can feed an arbitrary reference signal to synthesize text with its speaking style. Alternatively, we can remove the reference encoder and directly control synthesis using the learned interpretable tokens.

Reference encoder – аудио произвольной длины в вектор фиксированной длины (CNN + RNN)

**взвешенная сумма GST конкатенируется с text emb. (на каждом тике времени t)
учим совместно**

Yuxuan Wang et al. «Style Tokens: Unsupervised Style Modeling, Control and Transfer in End-to-End Speech Synthesis» // <https://arxiv.org/pdf/1803.09017.pdf>

Melotron: GST

Кодировщик: CNN + GRU + слой style token

**Последний содержит эмбединги стиля,
к которым по полученному вектору через МНА осуществляется запрос**

При обучении вход – целевая мелспектрограмма

При работе – та, стиль которой должен быть

Mellotron

**Дополнительный вход – контур основной частоты F0,
Вычисленный алгоритмом YIN**

http://audition.ens.fr/adc/pdf/2002_JASA_YIN.pdf

**При обучении – вычисленное для целевого
При работе – авторы предлагают использовать нули**

Mellotron

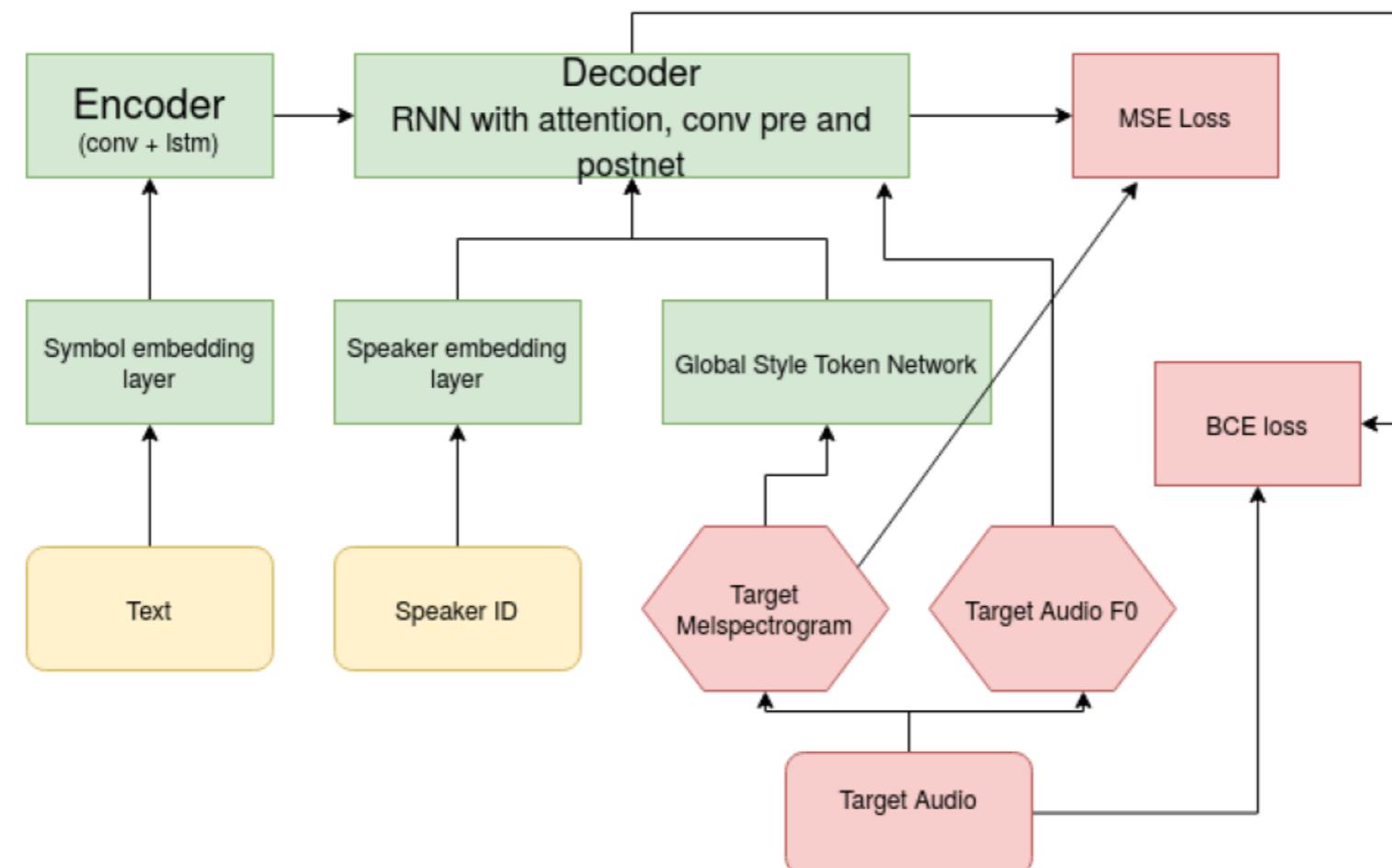


Рис. 4: Модель Mellotron при обучении (красное - то что доступно/вычисляется только на обучении, зеленое - модель, желтое - данные доступные всегда)

Рис. [Владислав Филимонов]

Glow-TTS

Glow-TTS – потоковая (flow) генеративная модель для параллельного (не авторегрессионного) TTS, не требующая внешнего выравнивания

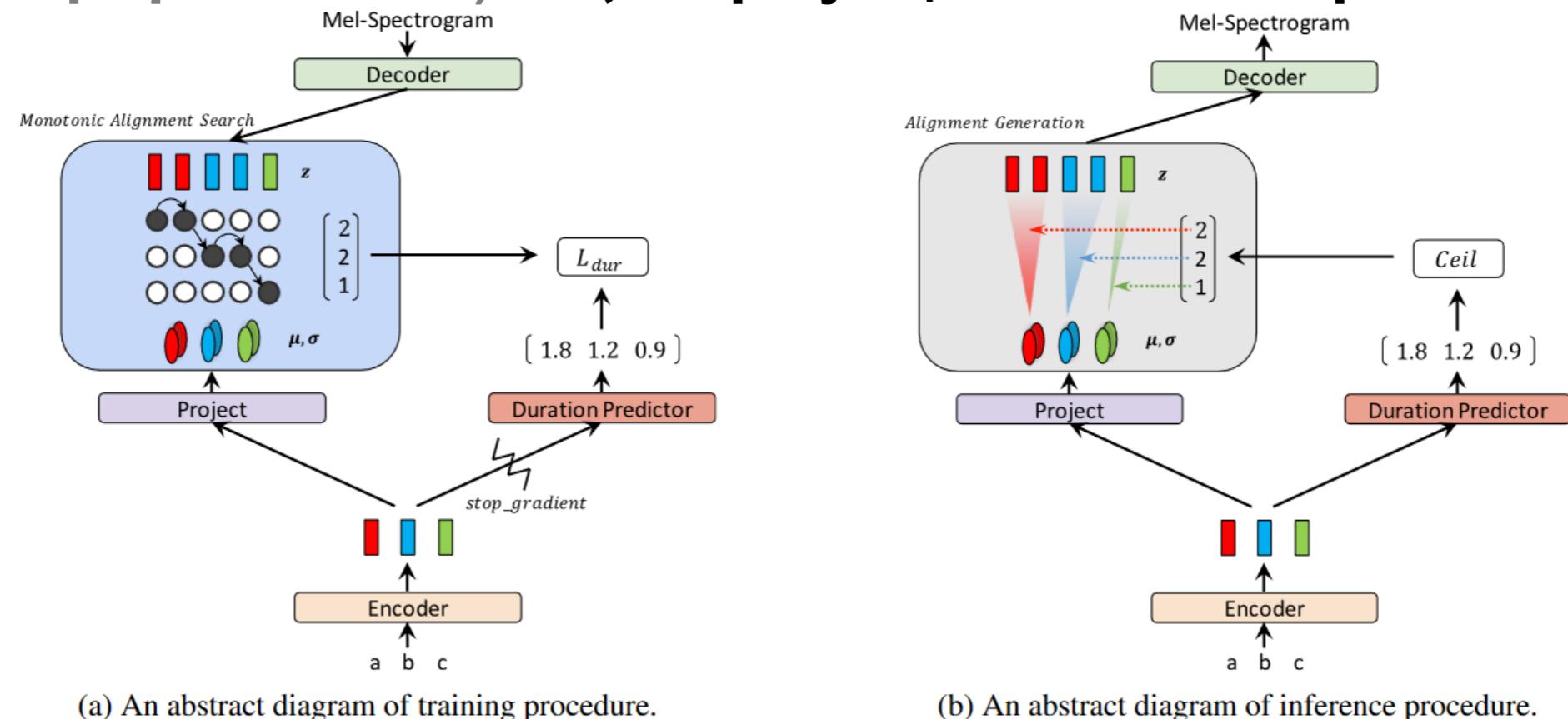


Figure 1. Training and inference procedures of Glow-TTS.

<https://deeppai.org/publication/glow-tts-a-generative-flow-for-text-to-speech-via-monotonic-alignment-search>

Glow-TTS

**Обуславливание не в каждый поток,
а влияет на параметры распределения (μ, σ)
для каждого токена**

$$\log P_X(x|c; \theta, A) = \log P_Z(z; c, \theta, A) + \log \det \frac{\partial f_{dec}(x)}{\partial x} \quad (1)$$

$$\log P_Z(z; c, \theta, A) = \sum_{j=1}^{T_{mel}} \log \mathcal{N}(z_j; \mu_{A(j)}, \sigma_{A(j)}) \quad (2)$$

**Тут возникает понятие (оптимального) выравнивания:
соответствия $A(j)$ -го токена и z_j**

**итеративное решение двух задач:
поиск оптимального выравнивания
поиск параметров распределения**

Glow-TTS

Algorithm 1 Monotonic Alignment Search

Input: latent representation z , the statistics of prior distribution μ, σ , the mel-spectrogram length T_{mel} , the text length T_{text}

Output: monotonic alignment A^*

Initialize $Q_{\cdot, \cdot} \leftarrow -\infty$, a cache to store the maximum log-likelihood calculations

Compute the first raw $Q_{1,j} \leftarrow \sum_{k=1}^j \log \mathcal{N}(z_k; \mu_1, \sigma_1)$

for $j = 2$ **to** T_{mel} **do**

for $i = 2$ **to** $\min(j, T_{text})$ **do**

$Q_{i,j} \leftarrow \max(Q_{i-1,j-1}, Q_{i,j-1}) + \log \mathcal{N}(z_j; \mu_i, \sigma_i)$

end for

end for

Initialize $A^*(T_{mel}) \leftarrow T_{text}$

for $j = T_{mel} - 1$ **to** 1 **do**

$A^*(j) \leftarrow \arg \max_{i \in \{A^*(j+1)-1, A^*(j+1)\}} Q_{i,j}$

end for

Glow-TTS

Table 1. The Mean Opinion Score (MOS) of single speaker TTS models with 95% confidence intervals.

METHOD	9-SCALE MOS
GT	4.54 ± 0.06
GT (MEL + WAVEGLOW)	4.19 ± 0.07
TACOTRON2 (MEL + WAVEGLOW)	3.88 ± 0.08
GLOW-TTS ($T = 0.333$, MEL + WAVEGLOW)	4.01 ± 0.08
GLOW-TTS ($T = 0.500$, MEL + WAVEGLOW)	3.96 ± 0.08
GLOW-TTS ($T = 0.667$, MEL + WAVEGLOW)	3.97 ± 0.08

Table 2. The Mean Opinion Score (MOS) of a multi-speaker TTS with 95% confidence intervals.

METHOD	9-SCALE MOS
GT	4.29 ± 0.06
GT (MEL + WAVEGLOW)	4.06 ± 0.07
GLOW-TTS ($T = 0.333$, MEL + WAVEGLOW)	3.40 ± 0.09
GLOW-TTS ($T = 0.500$, MEL + WAVEGLOW)	3.54 ± 0.09
GLOW-TTS ($T = 0.667$, MEL + WAVEGLOW)	3.52 ± 0.09

ground truth = GT

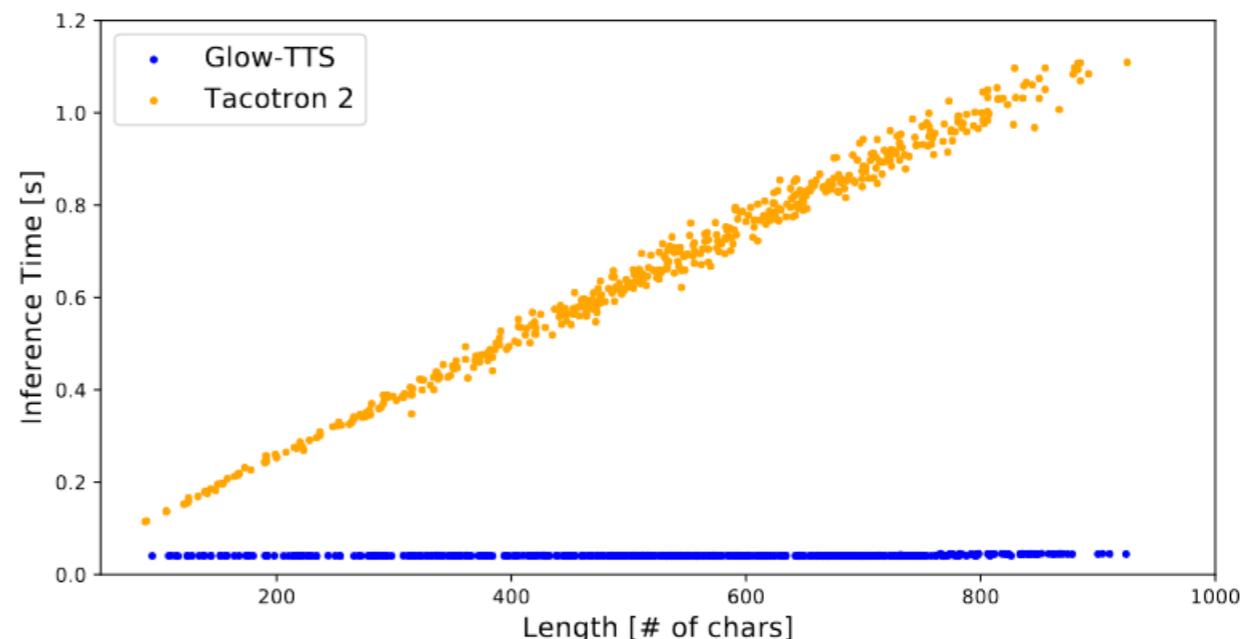
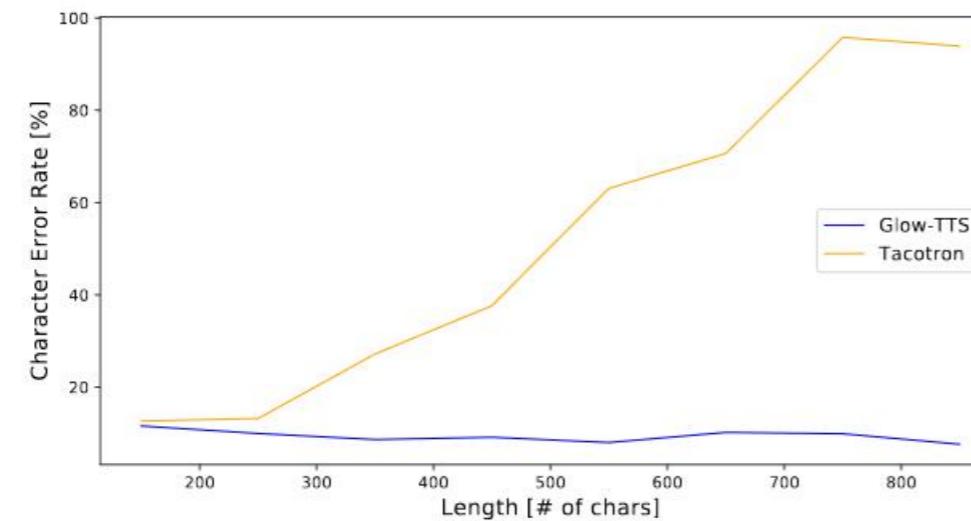


Figure 3. The inference time comparison for Tacotron 2 and Glow-TTS (yellow: Tacotron2, blue: Glow-TTS).

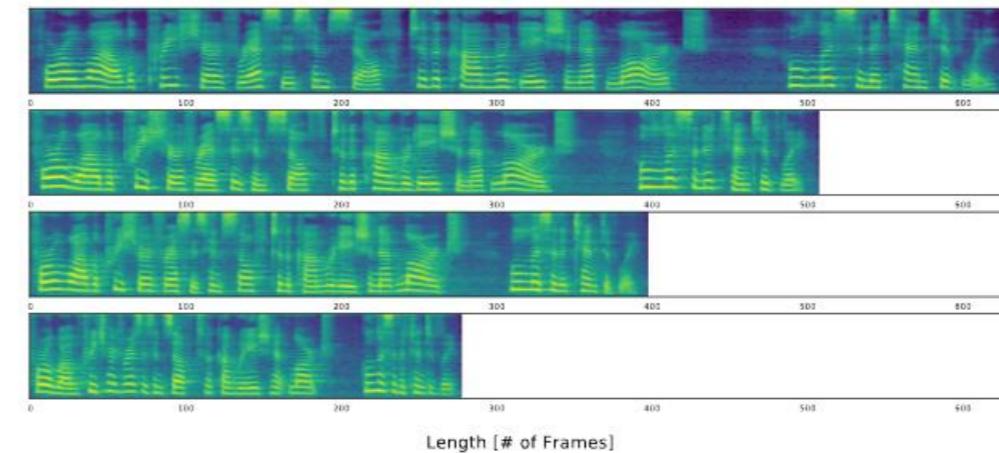
После выхода статьи стало лучше:

- 1) **vocoder = HiFi-GAN**
- 2) **если пустые фонемы между всеми проставить**

ABC → A_B_C



(a) Robustness to the length of input utterance (yellow: Tacotron2, blue: Glow-TTS).



(b) Mel-spectrograms with different speaking duration (x1.25, x1.0, x0.75, and x0.5 from top to bottom, respectively).

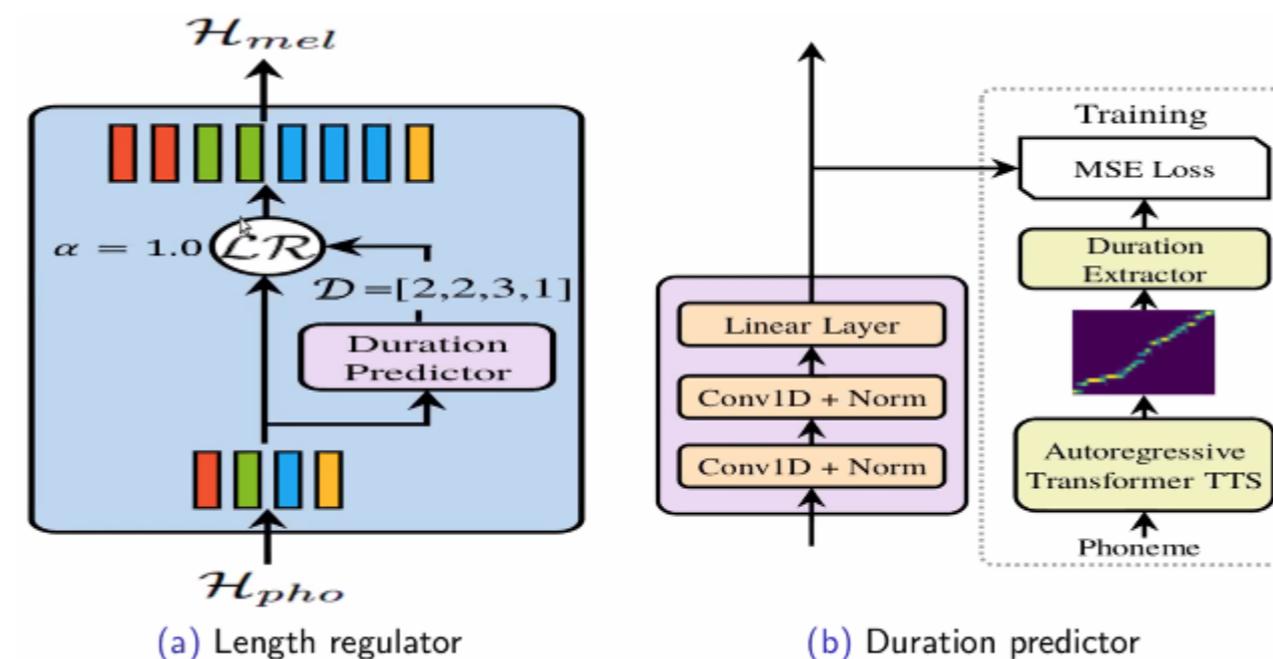
CER – у последующего Google ASR

FastSpeech

быстрое трансформерная параллельная генерация мел-спектрограмм

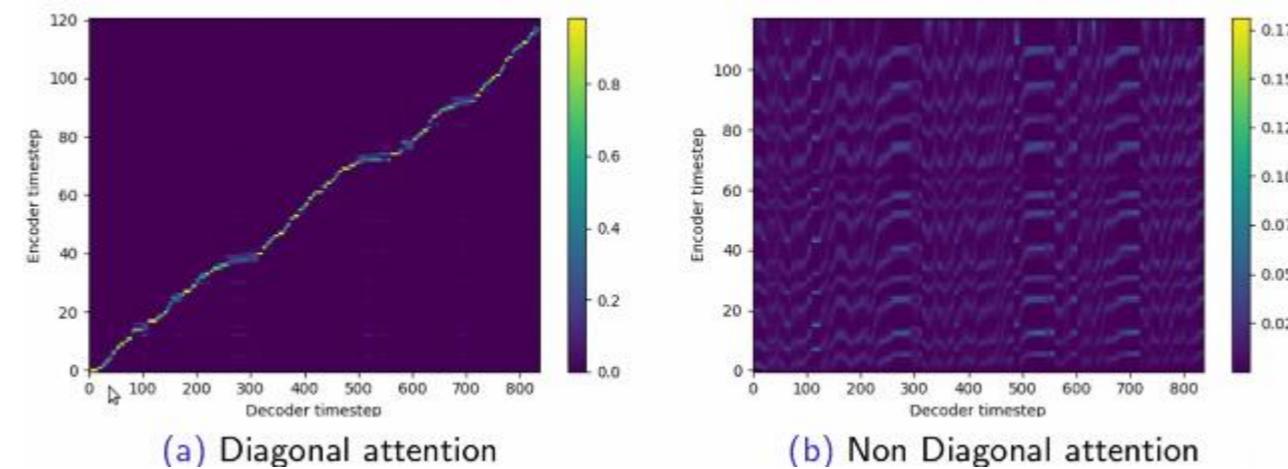
нет внимания между текстом и речью (нет пропусков и повторений – что бывало раньше)
чтобы избавиться от авторегрессионности,
надо прогнозировать продолжительность

Length Regulator – сколько будет фреймов и привести представления фонем в нужную размерность



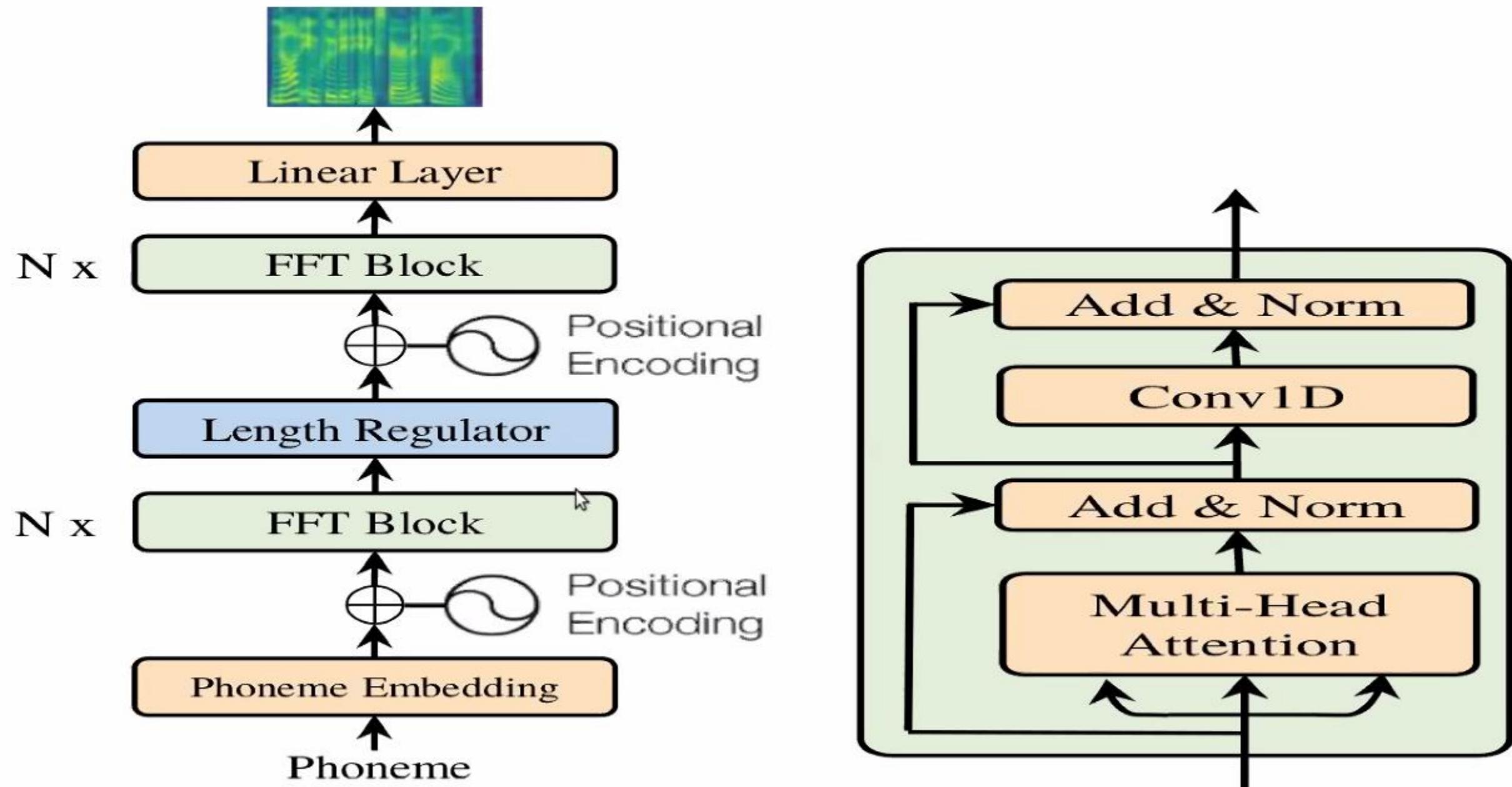
FastSpeech

**как предсказывают продолжительность
использовали предыдущую модель Transformer TTS,
там по матрице внимания оценивают продолжительность
– на этом можно обучить модель**



**есть тонкость – несколько вниманий,
выбираем максимально диагональную матрицу внимания**

FastSpeech



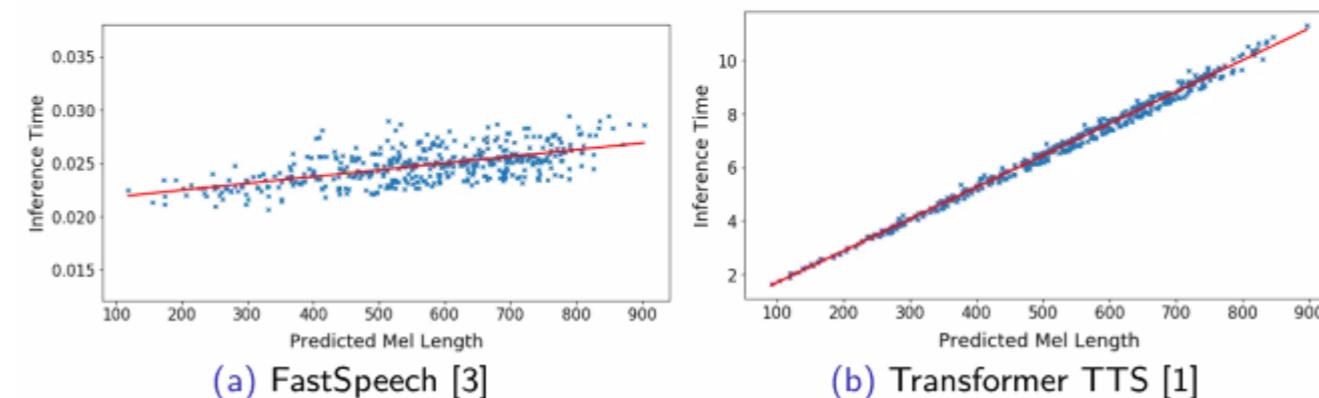
FastSpeech

Method	MOS
<i>GT</i>	4.41 ± 0.08
<i>GT (Mel + WaveGlow[5])</i>	4.00 ± 0.09
<i>Tacotron 2 [6] (Mel + WaveGlow [5])</i>	3.86 ± 0.09
<i>Transformer TTS [1] (Mel + WaveGlow[5])</i>	3.88 ± 0.09
<i>FastSpeech [3] (Mel + WaveGlow[5])</i>	3.84 ± 0.08

Table: The MOS with 95% confidence intervals.

неавторегрессионная – быстрее, слабо зависит от длины генерируемой фразы:

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

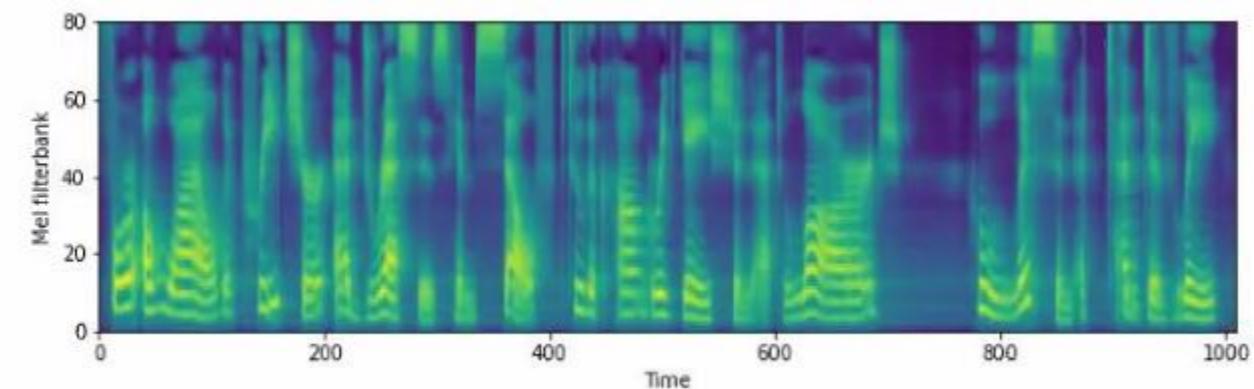
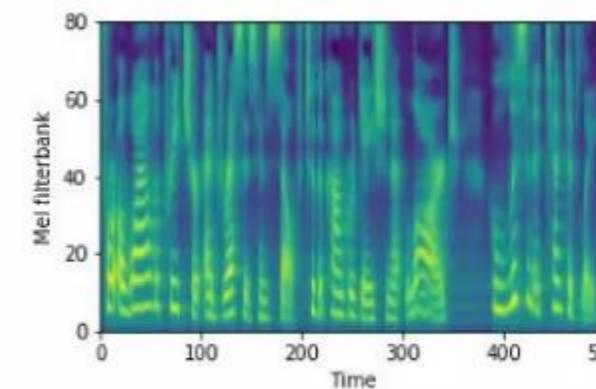
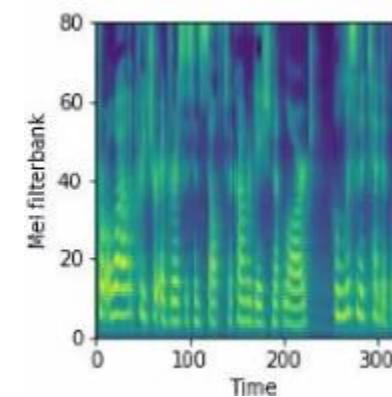


FastSpeech

Артефакты на 50 сложных предложениях:

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

Можно увеличивать / уменьшать скорость:



FastSpeech2

FastSpeech

неавторегрессионная модель

autoregressive teacher model for duration prediction (2)
generated mel-spectrograms for knowledge distillation

FastSpeech2

directly training the model with ground-truth target (вместо 2)
introducing more variation information of speech
(e.g., pitch, energy and more accurate duration) as conditional inputs

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 // <https://arxiv.org/pdf/2006.04558.pdf>

FastSpeech2

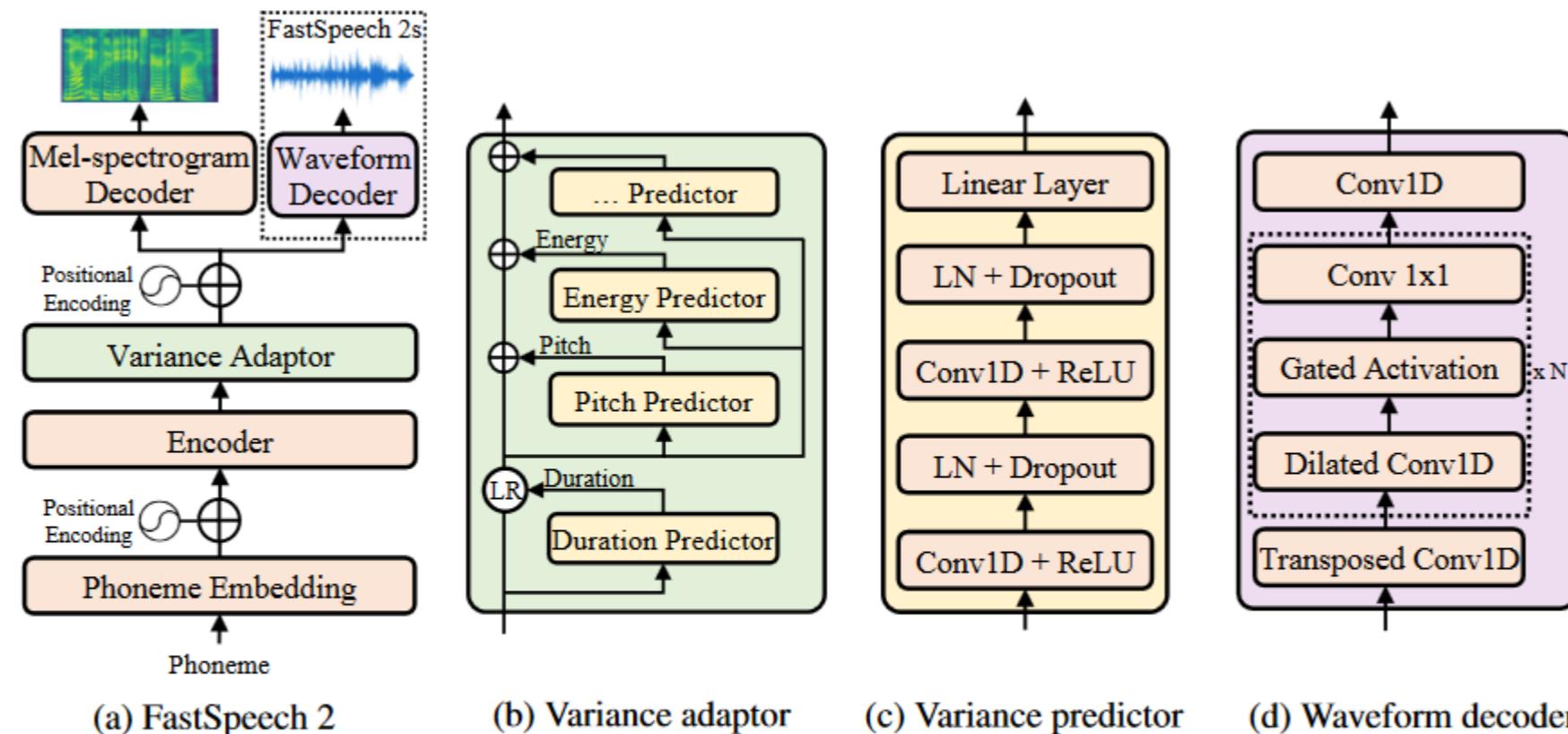


Figure 1: The overall architecture for FastSpeech 2 and 2s. LR in subfigure (b) denotes the length regulator operation proposed in FastSpeech. LN in subfigure (c) denotes layer normalization. Variance predictor represents duration/pitch/energy predictor.

FastSpeech 2s – минутя мел-спектограмму сразу в waveform

FastSpeech2

1) представления фонем

2) variance adaptor – добавление различной информации:
duration, pitch, energy
предсказывается с MSE-loss

Variance predictor = 2-layer 1D-convolutional network with ReLU activation, each followed by the layer normalization and the dropout layer + extra linear layer

Duration predictor – в логарифмической шкале
истинные извлекаем с помощью MFA (Montreal forced aligner)
pitch predictor – F0 (по фрейму)
energy predictor – sequence of the energy

3) mel-spectrogram decoder – получение мел-спектограммы

feed-forward Transformer block = stack of self-attention layer + 1D-convolution

FastSpeech2

как и в fastpitch предсказываем какое-то значение (например, энергию),
область её значений разделена на бины,
каждому номеру бина соответствует некоторая кодировка

FastSpeech2

encoder

объём словаря фонем = 76

4 feed-forward Transformer (FFT) blocks

размерность представления = 256

число головок = 2

mel-spectrogram decoder

4 feed-forward Transformer (FFT) blocks

размерность представления = 256

число головок = 2

размерность выхода = 80

MAE

подробнее ниже

FastSpeech2

Table 6: Hyperparameters of Transformer TTS, FastSpeech and FastSpeech 2. Decoder is for Transformer TTS, mel-spectrogram decoder is for FastSpeech and FastSpeech 2 and waveform decoder is for FastSpeech 2s.

Hyperparameter	Transformer TTS	FastSpeech/FastSpeech 2/2s
Phoneme Embedding Dimension	256	256
Pre-net Layers	3	/
Pre-net Hidden	256	/
Encoder Layers	4	4
Encoder Hidden	256	256
Encoder Conv1D Kernel	9	9
Encoder Conv1D Filter Size	1024	1024
Encoder Attention Heads	2	2
Decoder/Mel-Spectrogram Decoder Layers	4	4
Decoder/Mel-Spectrogram Decoder Hidden	384	384
Decoder/Mel-Spectrogram Decoder Conv1D Kernel	9	9
Decoder/Mel-Spectrogram Decoder Conv1D Filter Size	1024	1024
Decoder/Mel-Spectrogram Decoder Attention Headers	2	2
Encoder/Decoder Dropout	0.1	0.2
Variance Predictor Conv1D Kernel	/	3
Variance Predictor Conv1D Filter Size	/	256
Variance Predictor Dropout	/	0.5
Waveform Decoder Convolution Blocks	/	30
Waveform Decoder Dilated Conv1D Kernel size	/	3
Waveform Decoder Transposed Conv1D Filter Size	/	64
Waveform Decoder Skip Channle Size	/	64
Batch Size	48	48/48/12
Total Number of Parameters	24M	23M/27M/28M

FastSpeech2

Method	MOS
<i>GT</i>	4.27 ± 0.07
<i>GT (Mel + PWG)</i>	3.92 ± 0.08
<i>Tacotron 2 [21] (Mel + PWG)</i>	3.74 ± 0.07
<i>Transformer TTS [10] (Mel + PWG)</i>	3.79 ± 0.08
<i>FastSpeech [20] (Mel + PWG)</i>	3.67 ± 0.08
<i>FastSpeech 2 (Mel + PWG)</i>	3.77 ± 0.08
<i>FastSpeech 2s</i>	3.79 ± 0.08

Table 1: The MOS with 95% confidence intervals.

PWG = Parallel WaveGAN (vocoder)

Method	Training Time (h)	Inference Speed (RTF)	Inference Speedup
<i>Transformer TTS [10]</i>	38.64	8.26×10^{-1}	/
<i>FastSpeech [20]</i>	53.12	5.41×10^{-3}	152×
<i>FastSpeech 2</i>	16.46	5.51×10^{-3}	149×
<i>FastSpeech 2s</i>	/	4.87×10^{-3}	170×

Table 2: The comparison of training time and inference latency in waveform synthesis. The training time of *FastSpeech* includes teacher and student training. RTF denotes the real-time factor, that is the time (in seconds) required for the system to synthesize one second waveform. The training and inference latency 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. Besides, we do not include the time of GPU memory garbage collection and transferring input and output data between the CPU and the GPU. The speedup in waveform synthesis for *FastSpeech* is larger than that reported in Ren et al. [20] since we use Parallel WaveGAN as the vocoder which is much faster than WaveGlow.

End2end модели

Table 7: A list of fully end-to-end TTS models.

Model	One-Stage Training	AR/NAR	Modeling	Architecture
Char2Wav [309]	N	AR	Seq2Seq	RNN
ClariNet [263]	N	AR	Flow	CNN
FastSpeech 2s [286]	Y	NAR	GAN	Self-Att/CNN
EATS [67]	Y	NAR	GAN	CNN
Wave-Tacotron [379]	Y	AR	Flow	CNN/RNN/Hybrid
EfficientTTS-Wav [229]	Y	NAR	GAN	CNN
VITS [157]	Y	NAR	VAE+Flow	CNN/Self-Att/Hybrid

Ссылки

хороший обзор – из него взяты картинки

Xu Tan et al. «A Survey on Neural Speech Synthesis» // <https://arxiv.org/pdf/2106.15561.pdf>