Актуальные подходы синтеза речи

Обзор и практическое применение

25 Апреля 2018

Введение

Задача: перевод последовательности символов в речевой аудио-сигнал с максимально реалистичным звучанием.

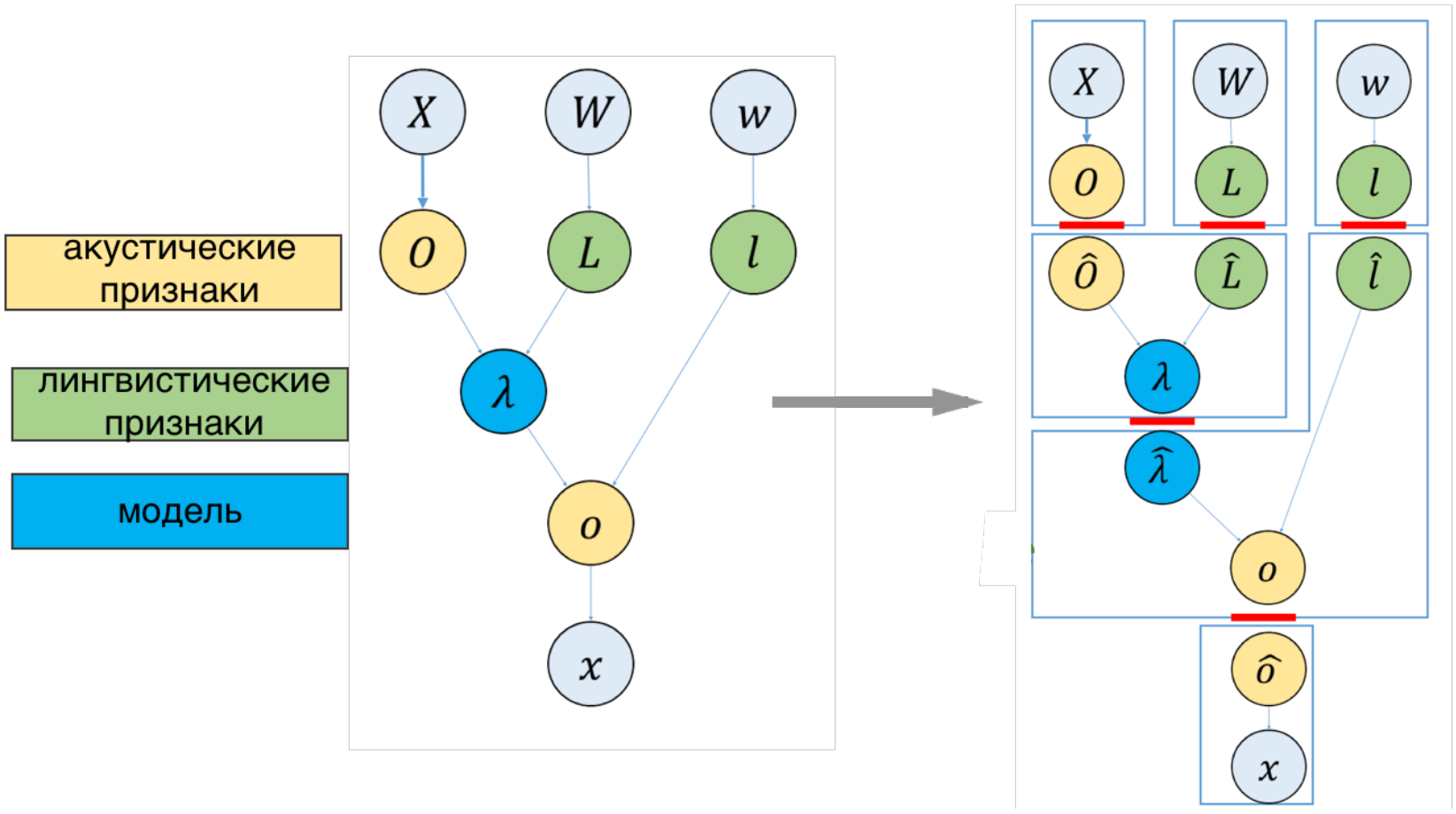
Широкая область применения:

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

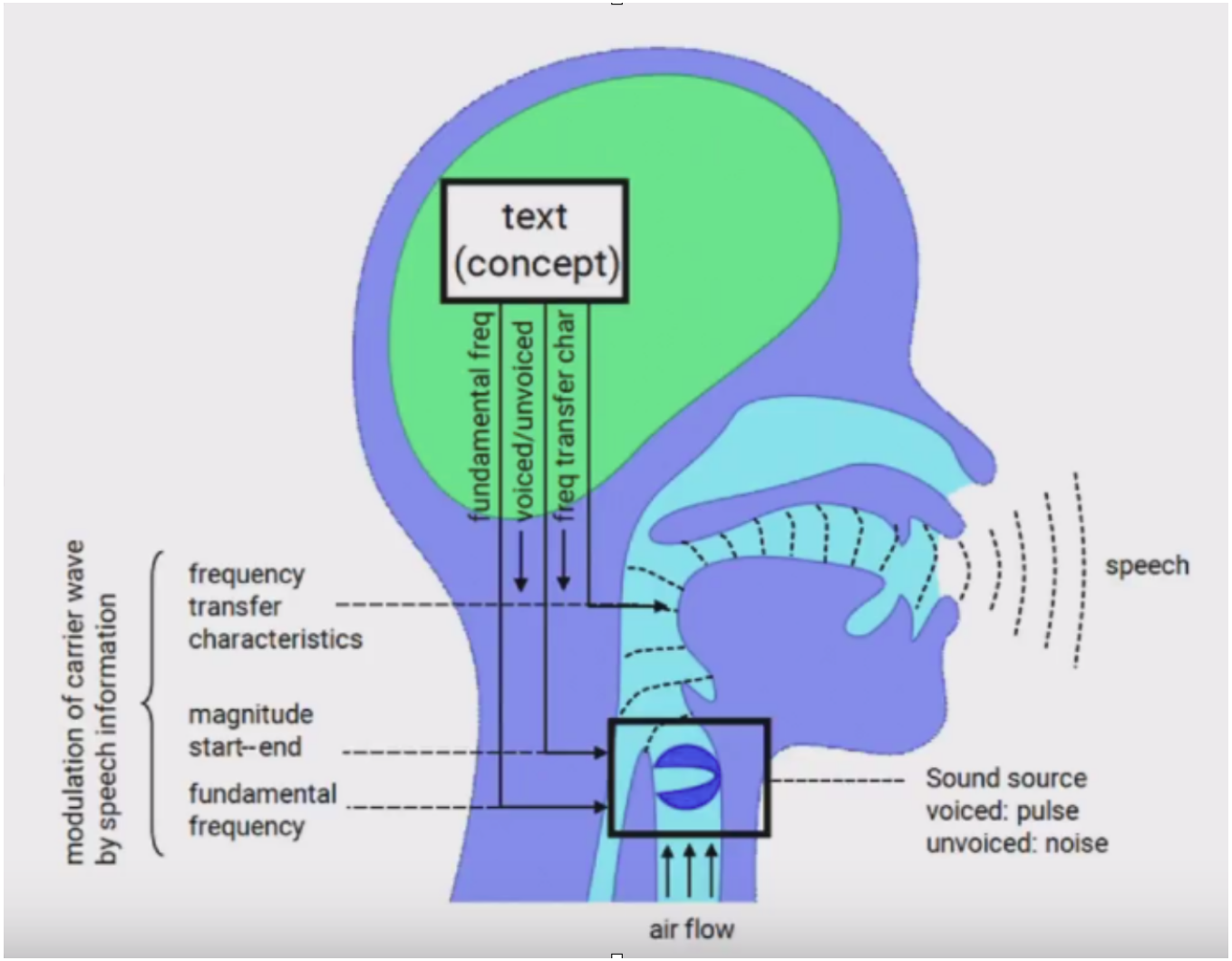
Почему нужен свой синтез?

- Amazon Polly, ЦРТ VoiceFabric – не бесплатны, далеки от идеала (на русском),
- открытые решения – нужно «допиливать»,
- и предпочтительна уникальность голоса.

Общий подход

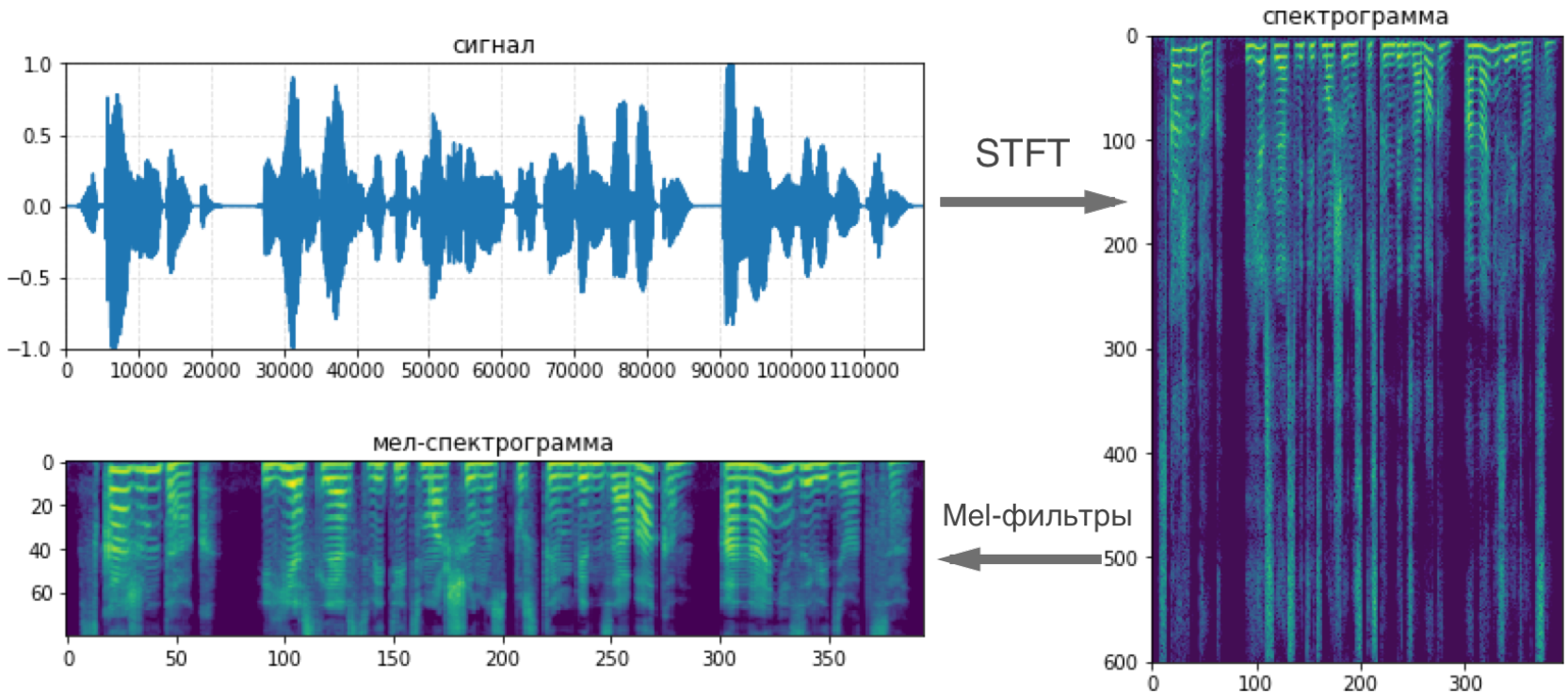


Физиологическая модель



Звук

Производные звукового сигнала[1]

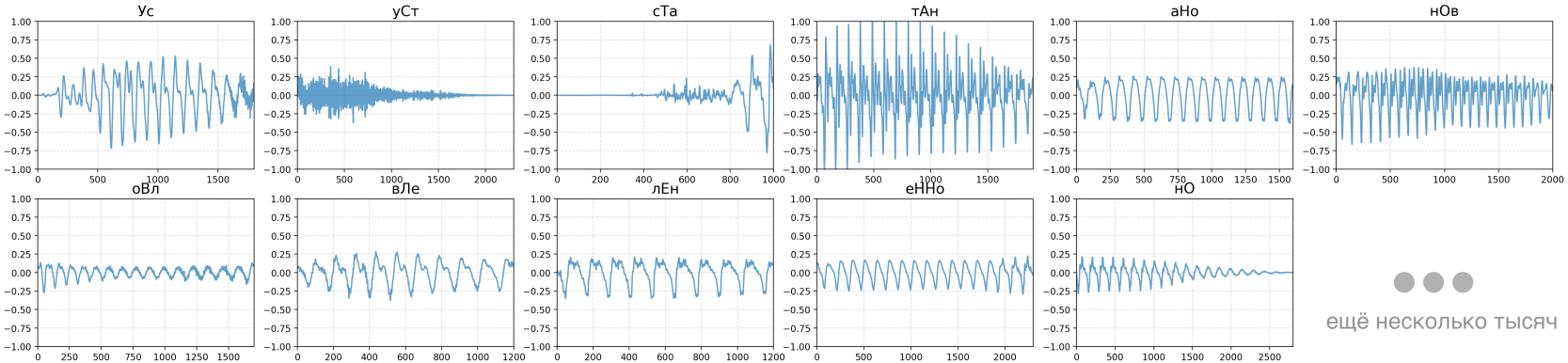


Sampling Rate: 24 KHz
STFT windows size: 1200
STFT hop size: 300

[1] Неплохое введение тут: http://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html

Конкатенативный подход

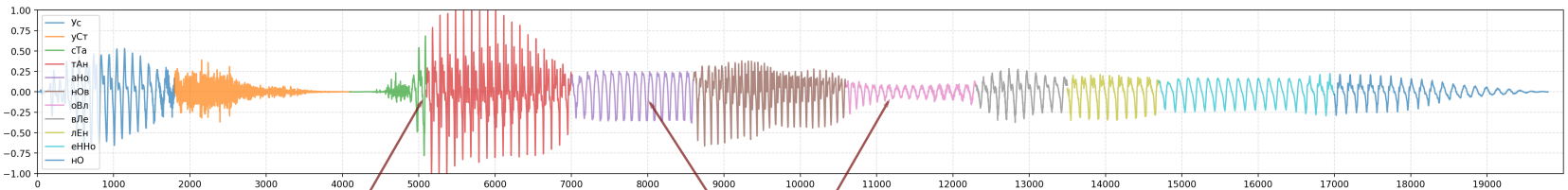
Коллекция звуков (units database)



●●●

ещё несколько тысяч

Конкатенация звуков (units concatenation)



граничные артифакты
(boundaries artifacts)

жесткое ограничение на F0

Deep Voice: Real-time Neural TTS[2]

Вместо hand-engineered features предлагается использовать лишь фонемы и F0, а всю остальную работу переложить на NN. Модель состоит из следующих компонентов:

- **grapheme-to-phoneme model** – переводить текст в фонемы (например, ARPABET),
- **segmentation model** – находить начало и конец фонемы на аудио (нужна для обучения),
- **phoneme duration model** – предсказывать длительность фонемы,
- **fundamental frequency model** – предсказывать F0 для фонемы,
- **audio synthesis model** – используя выходы предыдущих моделей, сгенерировать аудио-сигнал (рассмотрим чуть позже).

[2] S. O. Arik, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, J. Raiman, S. Sengupta и др., "Deep voice: Real-time neural text-to-speech", *arXiv preprint arXiv:1702.07825*, 2017.

DeepVoice (детально)

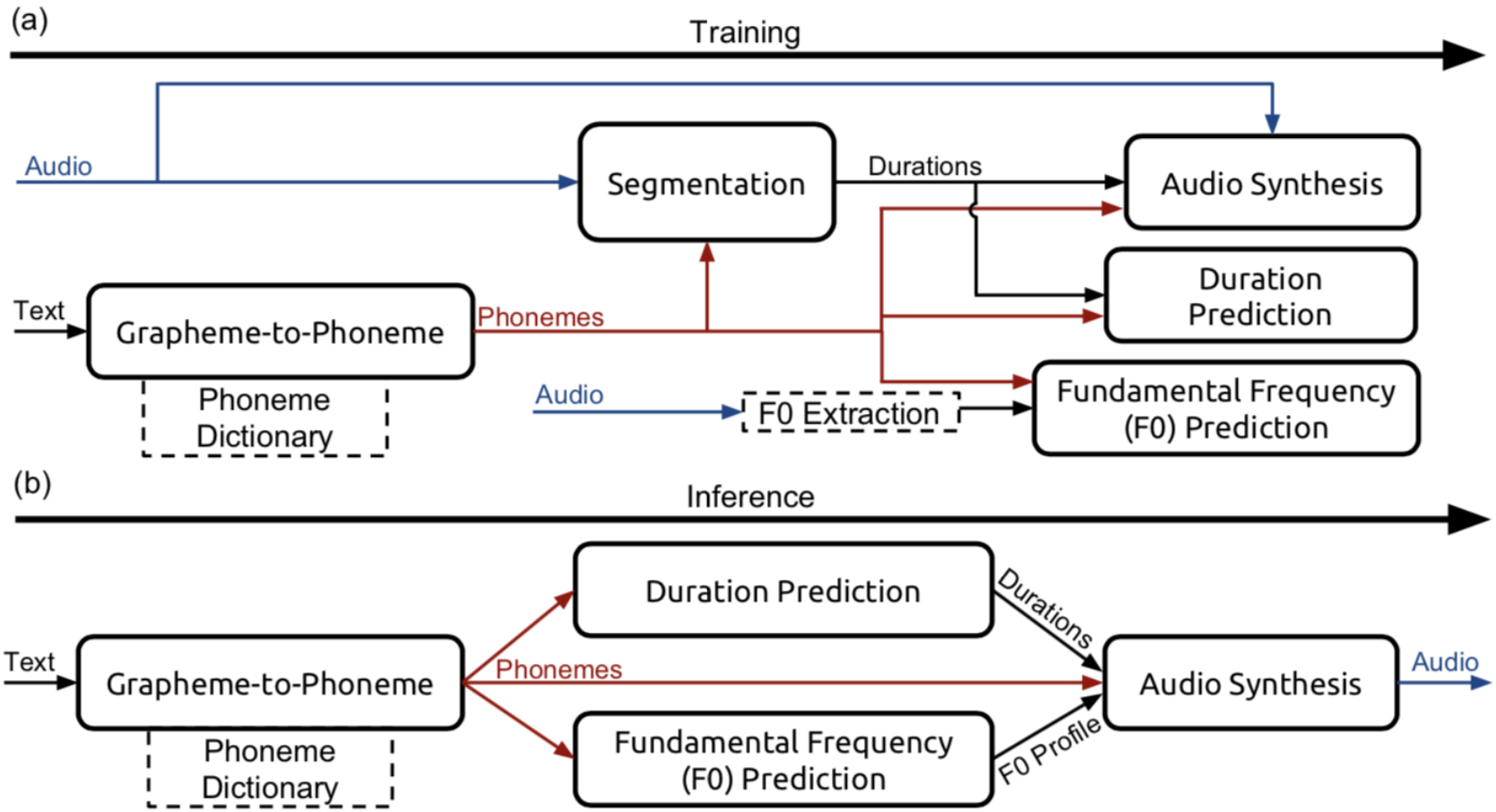
Grapheme-to-phoneme model – построен по архитектуре encoder-decoder (многослойные bi-GRU).

Segmentation model – свёрточная рекуррентная сеть, обученная (с использованием connectionist temporal classification loss) на выравнивание аудио (представленное 20-MFCC) с последовательностью фонем (на самом деле, пар фонем).

Fundamental frequency model и **phoneme duration model** совмещены в одну. На входе – последовательность фонем (с ударениями), затем два полносвязных слоя, затем uni-GRU, затем плоносвязный слой, который предсказывает:

1. длительность фонемы,
2. вероятность. что фонема произносится,
3. а так же 20 F0-частот распределенных по всей длительности произношения фонемы.

DeepVoice (схематично)



Вокодеры

Грубо говоря, их можно разбить на три группы:

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

Таким образом, мы

1. либо генерируем спектрограмму и переводим её в сигнал с Griffin-Lim[3],
2. либо генерируем некоторые параметры[4] и подаем на вход нейросетевому вокодеру (как это делается в DeepVoice).

[3] работает на практике не очень, далее будет пример

[4] mel-спектрограмма сама по себе может выступать в качестве параметризации

Griffin-Lim[5] (алгоритм)

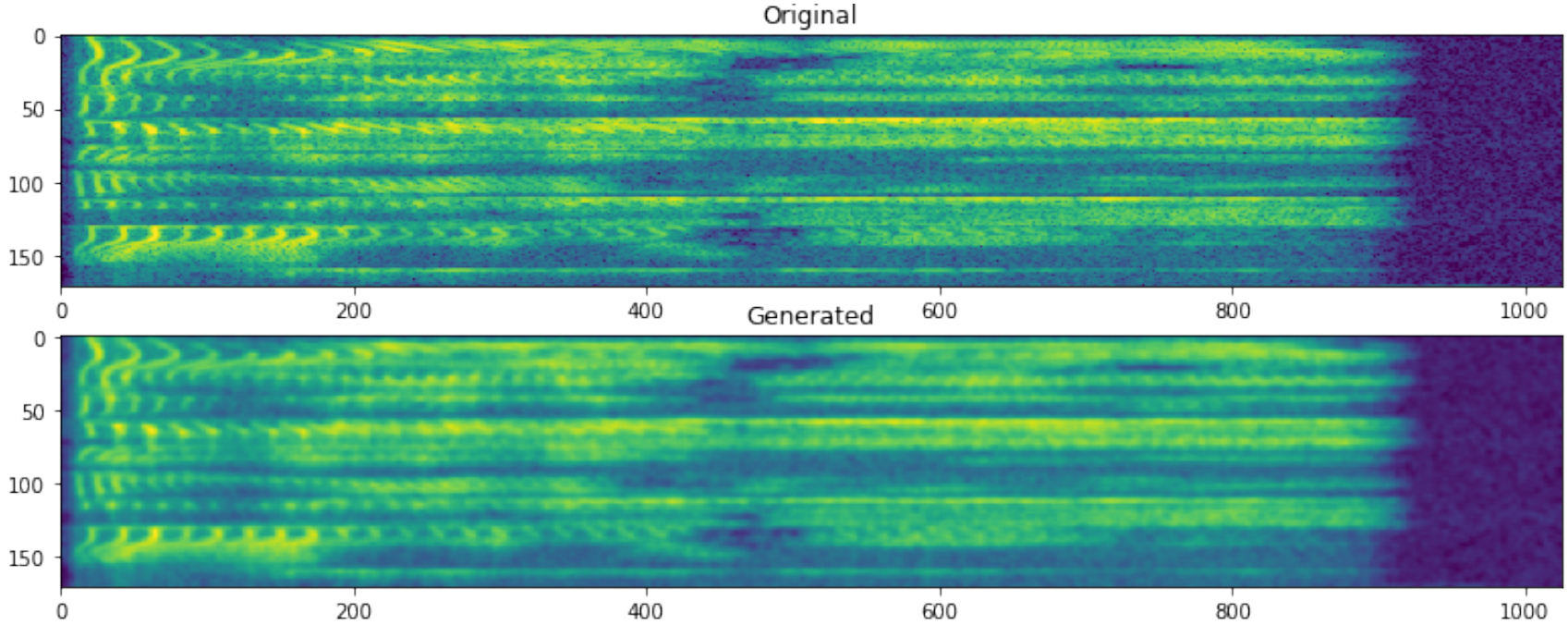
Data: $\rho = \|STFT(y)\|$ – амплитуда STFT сигнала y
Result: $\tilde{y} \approx y$ – востановленный сигнал y
$\phi \sim \mathbb{U}(0, 2\pi)$ – инициализация случайным образом;
$\tilde{y} = ISTFT(\rho * e^{i\phi})$ – первое приближение;
for *нужное число итераций* **do**
  $\phi = \arg STFT(\tilde{y})$ – угол преобразования;
  $\tilde{y} = ISTFT(\rho * e^{i\phi})$ – новое приближение;
end

Algorithm 1: Griffin-Lim

- Непараметрический метод (не требует обучения).
- Эффективно вычислим (легко параллелится);

[5] D. Griffin и J. Lim, "Signal estimation from modified short-time Fourier transform", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, т. 32, № 2, с. 236—243, 1984.

Griffin-Lim (недостаток)



MSE loss «размывает» спетрограммы, что добавляет «железность» к аудио, полученному от Griffin-Lim.

На оригинальных спектрограммах Griffin-Lim производит сигнал, почти неотличимый от оригинала.

Нейросетевой вокодер лишён данной проблемы, так как основная информация о голосе хранится в его параметрах сети.

WaveNet[6]

Математика

Вероятность сигнала $\boldsymbol{x} = x_1, \ldots, x_T$ может быть представлена в виде произведения условных вероятностей:

$$p(\boldsymbol{x}|\lambda) = \prod_{t=1}^{T} p(x_t|x_{t-1}, \ldots, x_1; \lambda),$$

где каждый сэмпл x_t зависит от всех предыдущих сэмплов, а λ – вектор или матрица, задающие условия синтеза (local или global conditioning).

Для задачи синтеза речи мы можем оценить каждый множитель как $p(x_t|x_{t-1}, \ldots, x_{t-m}; \lambda)$, где гиперпараметр m (receptive field) имеет приемлемое значение.

---

[6] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior и K. Kavukcuoglu, "Wavenet: A generative model for raw audio", *arXiv preprint arXiv:1609.03499*, 2016.

WaveNet

Оптимизация

Параметризуем распределение $p(x_t|x_{t-1}, \ldots, x_{t-m}; \lambda)$ (например, представим нейросетью) и будем оптимизировать функционал:

$$\mathcal{NLL}(\theta) = -\sum_{t=1}^{T} \log p(x_t|x_{t-1}, \ldots, x_{t-m}; \theta; \lambda)$$

Сэмплировать сигналы $\boldsymbol{x}$ мы можем построением последовательностей значений $x_1, x_2, \ldots, x_T$ авторегрессионным способом.

WaveNet

Метрика качества

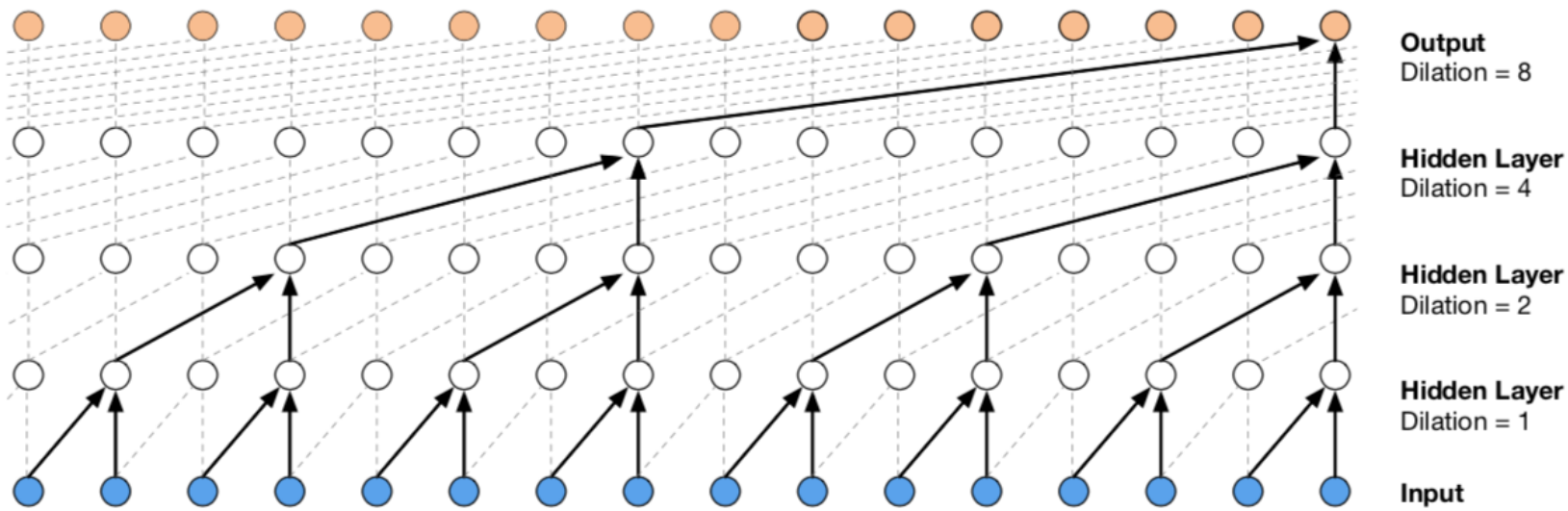
В качестве метрики можно использовать значения логарифма правдоподобия, но восприятие речи имеет качественный характер, поэтому лучше использовать MOS (mean opinion score).

Опрашивается группа людей, каждому человеку требуется оценить аудио-запись по пятибальной шкале, результаты усредняются. Эта метрика относительная и ресурсозатратная.

WaveNet

Архитектура

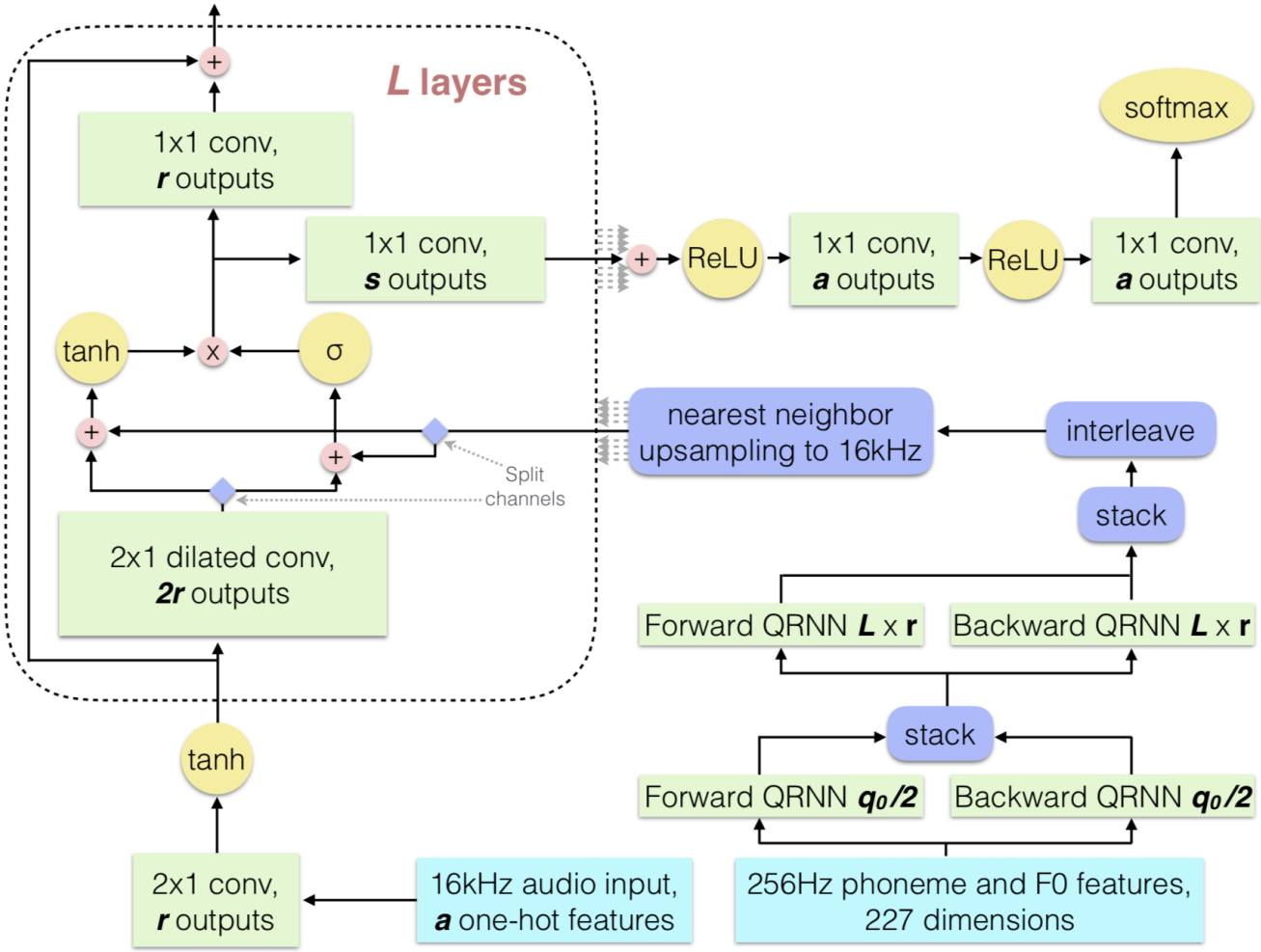
Граф потоков данных нейросети можно представить в следующем виде (пока без conditioning):



Каждый слой (hidden layer) представлен в виде блока.

WaveNet

Архитектура блока

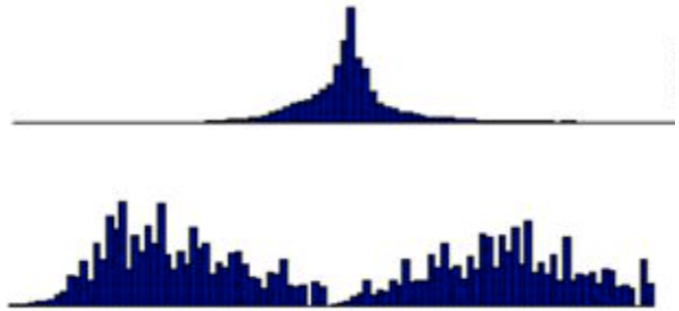


WaveNet

Представление аудио

Представим аудио в виде дискретного 8-битного сигнала.

Для более равномерного покрытия закодированным по формуле (μ-закону):



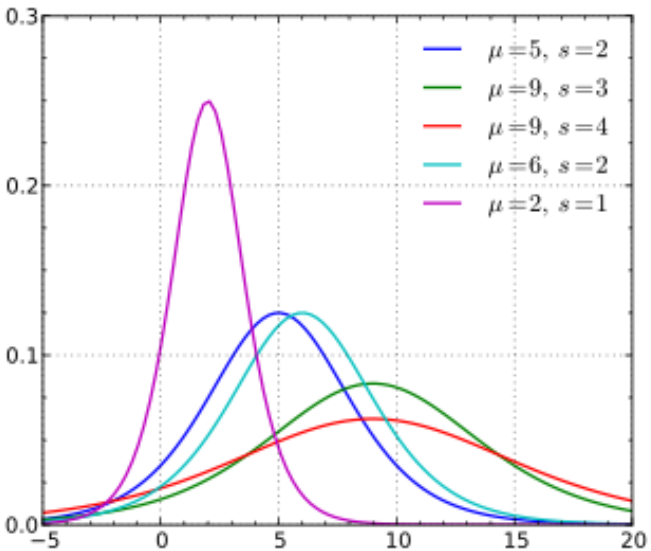
$$F(x) = sgn(x)\frac{\ln(1+\mu|x|)}{\ln(1+\mu)}$$

- Обучение проводим при помощи SGD (например, Adam).
- При обучение WaveNet работает как feed-forward сеть.
- Целевой сигнал, с которым считаем loss, сдвинут на receptive field сэмплов.
- Обучившись без conditioning, получим речь на непонятном языке.

WaveNet

Повышение разрядности аудио

Существует ощутимая разница между 8-битным и 16-битным аудио (шипение). Решение: предсказывать параметры некотрого распределения, из которого потом сэмплировать.



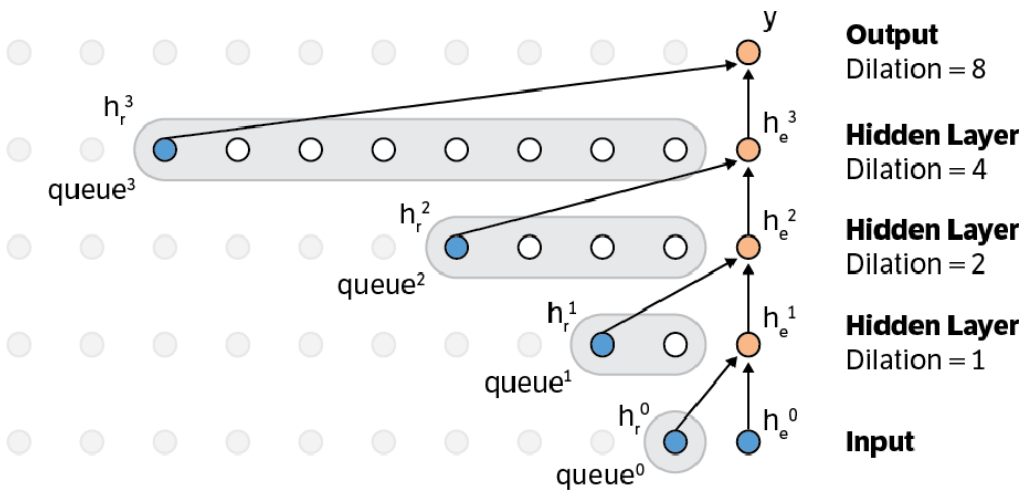
Вместо $p(x_t)$ предсказываем $\pi_t^{(i)}, \mu_t^{(i)}, s_t^{(i)}$, $i \in 1, \ldots, K$ – параметры смеси логистических распределений.

$$x_t \sim \sum_{i=1}^{K} \pi_t^{(i)} \mathit{logistic}(\mu_t^{(i)}, s_t^{(i)})$$

WaveNet

Быстрый инференс

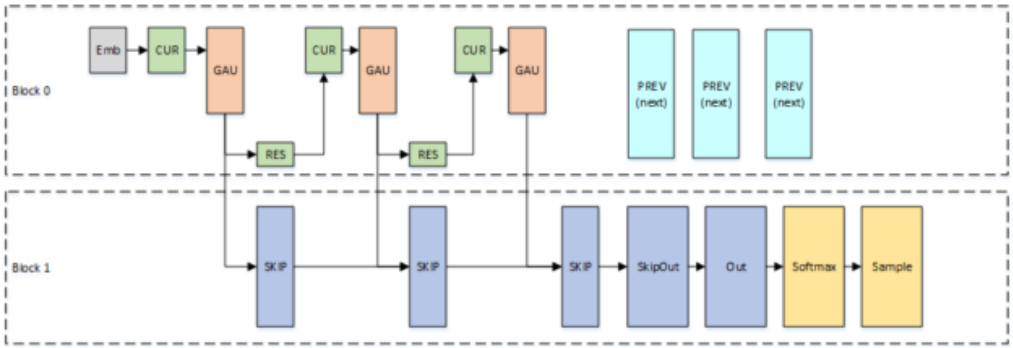
Наивная генерация на современном процессоре в 10.000 раз медленнее realtime.
Хранения промежуточных результатов[7] позволяет добиться линейной скорости от количества слоёв, но и этого недостаточно (50x-100x), поэтому приходится делать низкоуровневую реализацию с (BLAS + многопоточность).

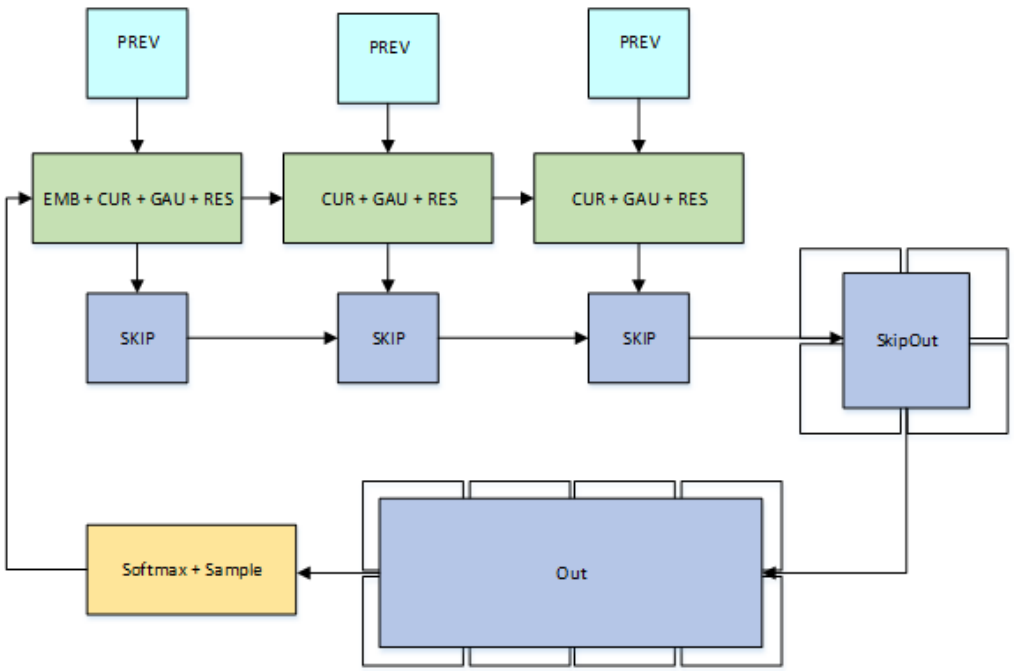


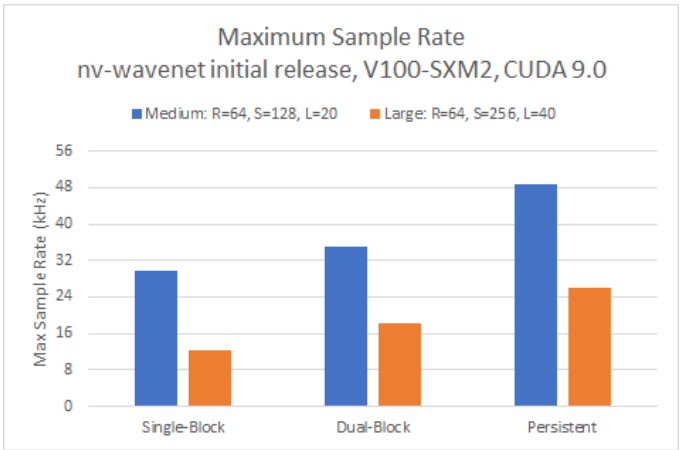
[7] T. L. Paine, P. Khorrami, S. Chang, Y. Zhang, P. Ramachandran, M. A. Hasegawa-Johnson и T. S. Huang, "Fast wavenet generation algorithm", *arXiv preprint arXiv:1611.09482*, 2016.

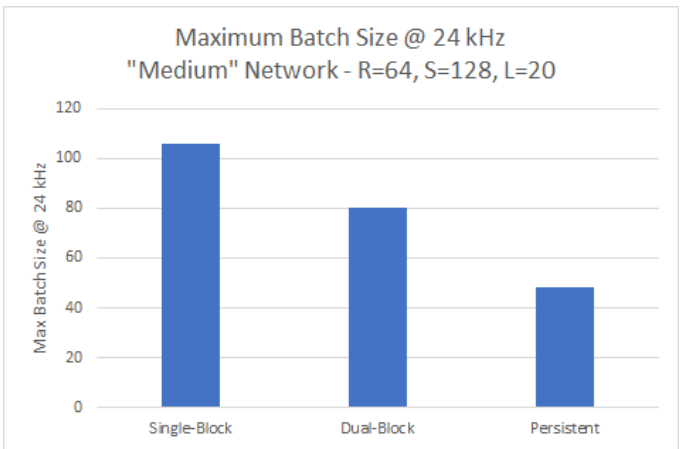
WaveNet

Быстрый инференс от NVIDIA[8]



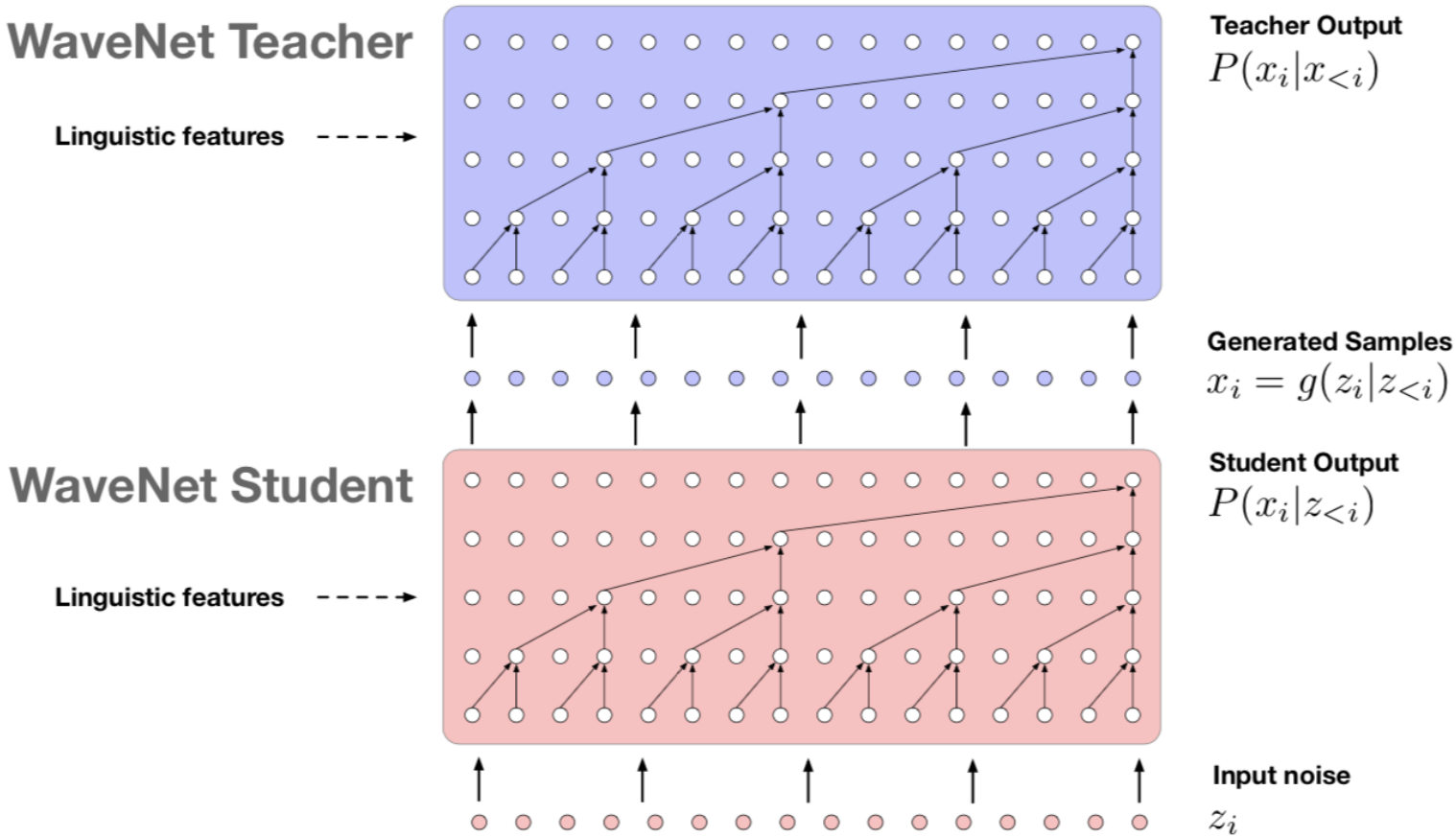






[8] https://github.com/NVIDIA/nv-wavenet

Parallel WaveNet[9]



[9] A. v. d. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. v. d. Driessche, E. Lockhart, L. C. Cobo, F. Stimberg и др., "Parallel WaveNet: Fast High-Fidelity Speech Synthesis", *arXiv preprint arXiv:1711.10433*, 2017.

SampleRNN[10]

Общая идея

- Построим **иерархию** рекуррентных сетей, каждая из которых имеет своё временное разрешение (temporal resolution).
- На вход каждой сети будем подавать **выход предыдущей сети**, увеличенный в разрешении, и **последовательность предыдущих сэмплов**.
- Увеличение разрешения будем производить обучаемым способом (strided transposed convolution).
- Одновременно со входом можно подавать conditioning.

[10] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville и Y. Bengio, "SampleRNN: An unconditional end-to-end neural audio generation model", *arXiv preprint arXiv:1612.07837*, 2016.

SampleRNN

Детально

$$inp_t = \begin{cases} W_x f_t^{(k)} + c_t^{(k+1)}; & 1 < k < K \\ f_t^{(k=K)}; & k = K \end{cases}$$

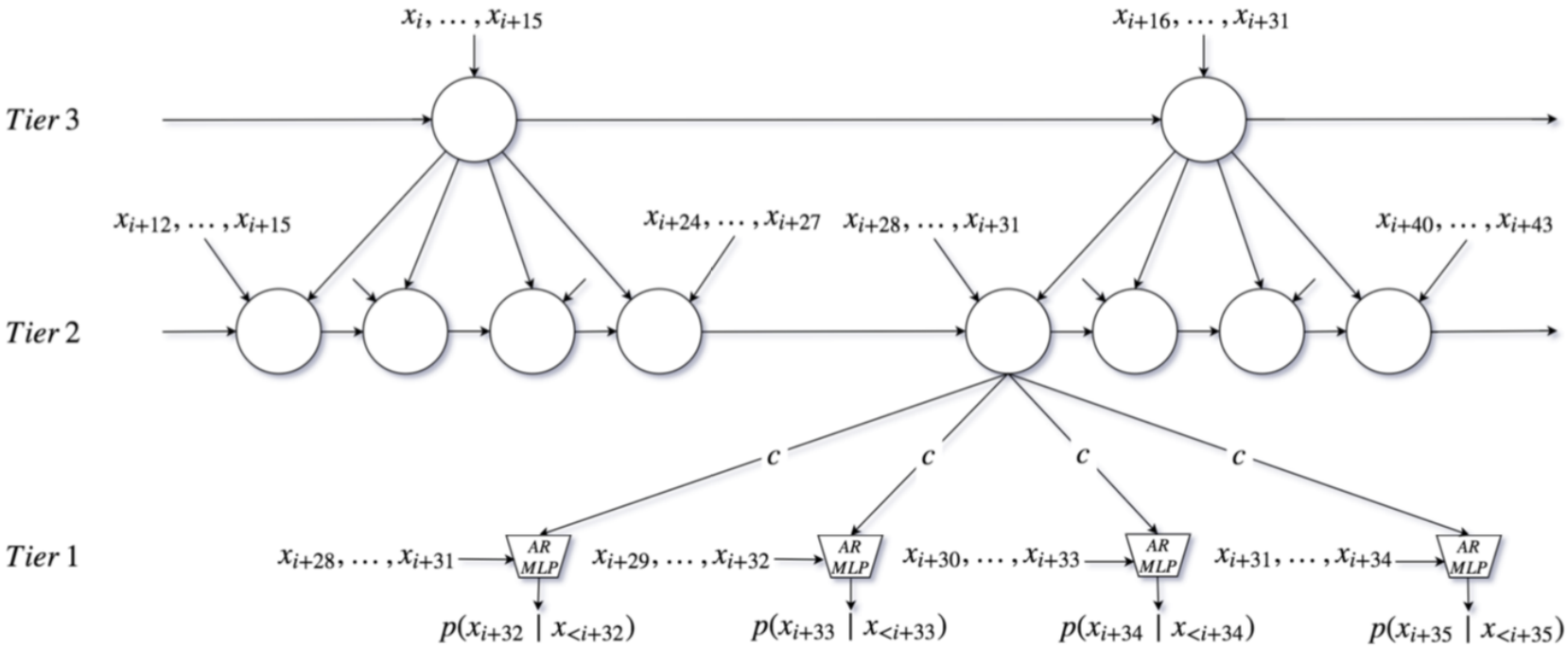
$$h_t = \mathcal{H}(h_{t-1}, inp_t)$$

$$c^{(k)}_{(t-1)*r+j} = W_j h_t; \qquad 1 \le j \le r$$

$$f_i^{(1)} = flatten([e_{i-FS^{(1)}+1}, \dots, e_i])$$

$$inp_i^{(1)} = W_x^{(1)} f_i^{(1)} + c_i^{(2)}$$

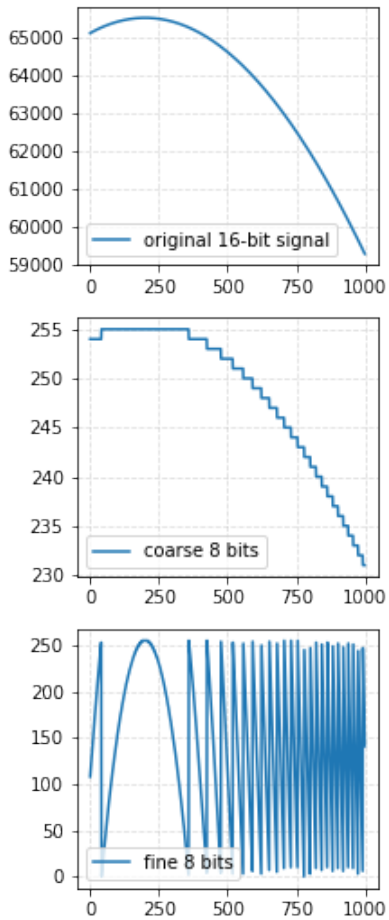
$$p(x_{i+1}|x_1, \dots, x_i) = Softmax(MLP(inp_i^{(1)}))$$



WaveRNN[11]

Общий идея

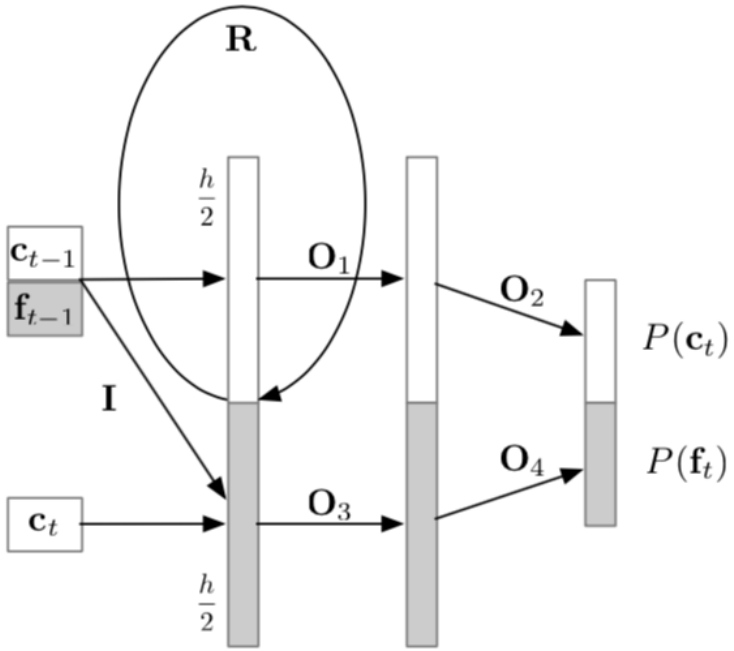
- В качестве модели используется модификация GRU.
- Для каждого предсказывается его coarse (старшие биты) и fine (младшие биты) части по отдельности.



[11] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. v. d. Oord, S. Dieleman и K. Kavukcuoglu, "Efficient Neural Audio Synthesis", *arXiv preprint arXiv:1802.08435*, 2018.

WaveRNN

Детально



$$\mathbf{x}_t = [\mathbf{c}_{t-1}, \mathbf{f}_{t-1}, \mathbf{c}_t]$$
$$\mathbf{u}_t = \sigma(\mathbf{R}_u \mathbf{h}_{t-1} + \mathbf{I}_u^\star \mathbf{x}_t)$$
$$\mathbf{r}_t = \sigma(\mathbf{R}_r \mathbf{h}_{t-1} + \mathbf{I}_r^\star \mathbf{x}_t)$$
$$\mathbf{e}_t = \tau(\mathbf{r}_t \circ (\mathbf{R}_e \mathbf{h}_{t-1}) + \mathbf{I}_e^\star \mathbf{x}_t)$$
$$\mathbf{h}_t = \mathbf{u}_t \circ \mathbf{h}_{t-1} + (1 - \mathbf{u}_t) \circ \mathbf{e}_t$$
$$\mathbf{y}_c, \mathbf{y}_f = \text{split}(\mathbf{h}_t)$$
$$P(\mathbf{c}_t) = \text{softmax}(\mathbf{O}_2 \, \text{relu}(\mathbf{O}_1 \mathbf{y}_c))$$
$$P(\mathbf{f}_t) = \text{softmax}(\mathbf{O}_4 \, \text{relu}(\mathbf{O}_3 \mathbf{y}_f))$$

- удивительно, но она работает (ну почти),
- можно эффективно использовать cuDNN при обучении,
- очень быстрый инференс сравнительно WaveNet

WaveRNN
Модификации

Качество модели сильно зависит от размерности вектора внутреннего представления.

Сделав матрицу весов **блочно-разряженной**, можно достичь оптимального соотношения скорость/качество.

Model	NLL	MOS
WaveNet	5.29	4.51 ± 0.08
WaveRNN 224	5.67	3.73 ± 0.09
WaveRNN 384	5.56	4.23 ± 0.09
WaveRNN 512	5.51	–
WaveRNN 896	5.42	4.37 ± 0.07
WaveRNN 1024	5.40	–
WaveRNN 1536	5.36	–
WaveRNN 2048	5.33	4.46 ± 0.07
Sparse WR Mobile	5.52	4.33 ± 0.08
Sparse WR 224 / 1536@97.8%	5.48	4.39 ± 0.07
Sparse WR 384 / 2048@96.4%	5.42	4.48 ± 0.07

Так же описывается способ сделать инференс ещё быстрее.

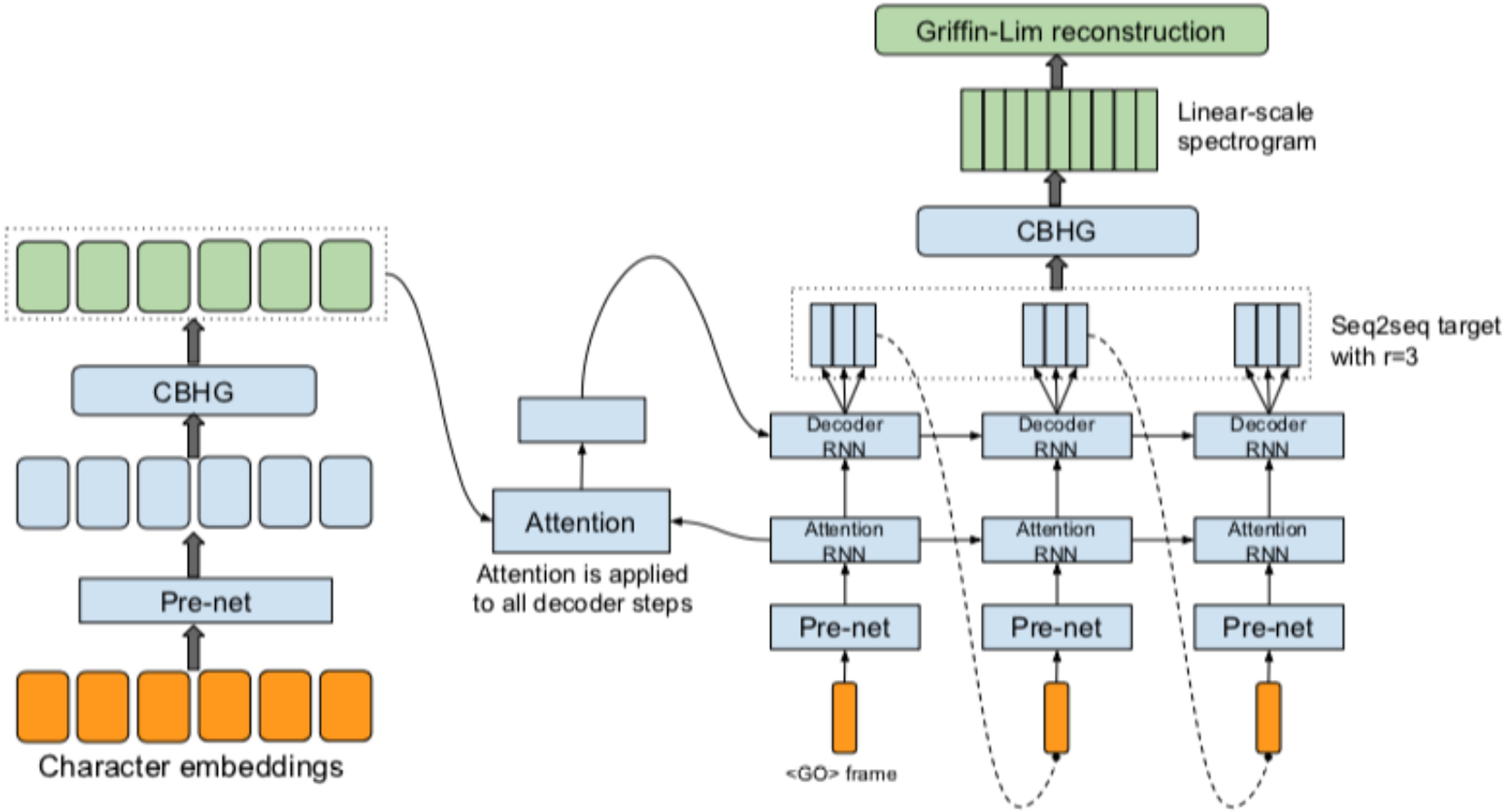
Tacotron[12]

- Действительно end-to-end text-to-speech подход, в то время как DeepVoice обучает раздельно нескольких моделей.
- Не требует какого-то конкретного вокодера (на выходе – спектрограмма).
- Применяется **attention** механизм для генерации mel-спектрограммы, которая затем подается на вход **postprocessing network** для получения линейной спектрограммы.

[12] 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 syn", *arXiv preprint arXiv:1703.10135*, 2017.

Tacotron

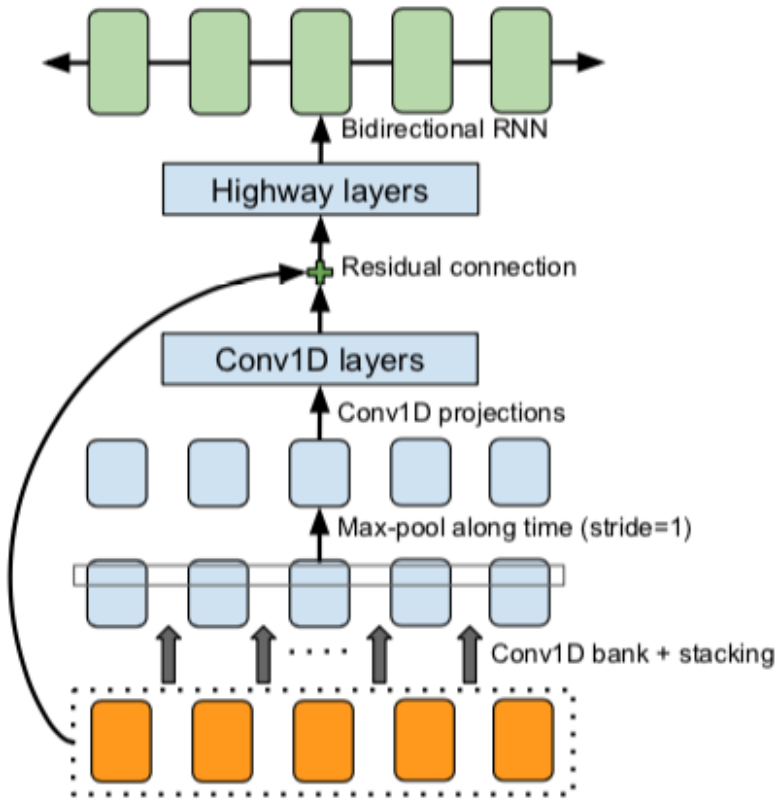
Общая архитектура



Tacotron
CBHG-модуль

1. 1-D свёрточная сеть
2. highway network
3. bidirectional GRU

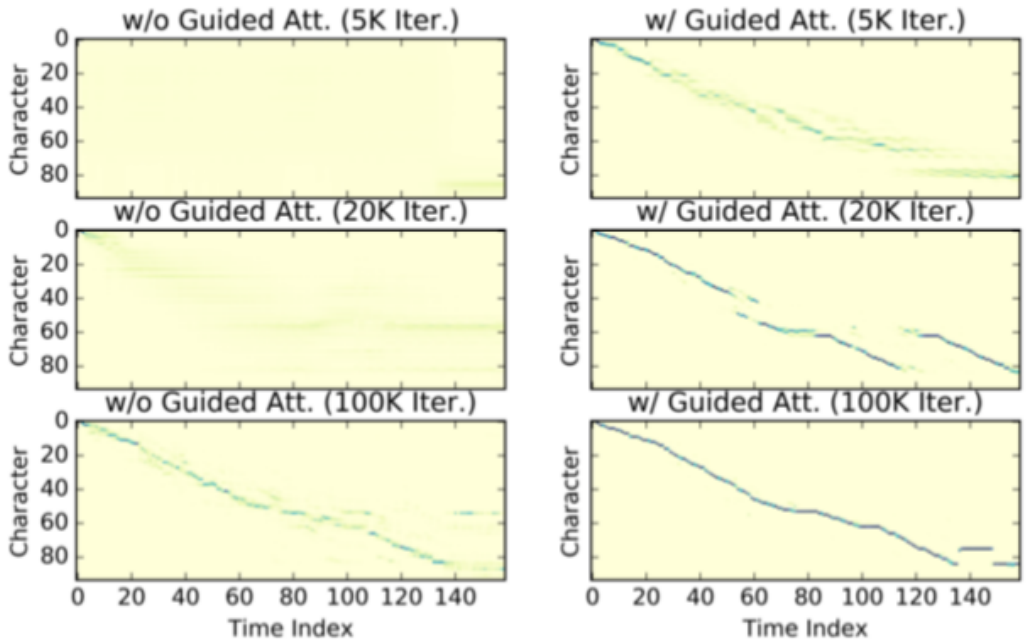
Утверждается, что **CBHG** – хороший экстрактор представлений для последовательностей.



Guided Attention

Использование априорных знаний может помочь при обучении **attention модуля**. Авторы статьи[13] (помимо собственной архитектуры, схожей с Tacotron) предлагают штрафовать матрицу внимания при отклонении её от диагональной формы.

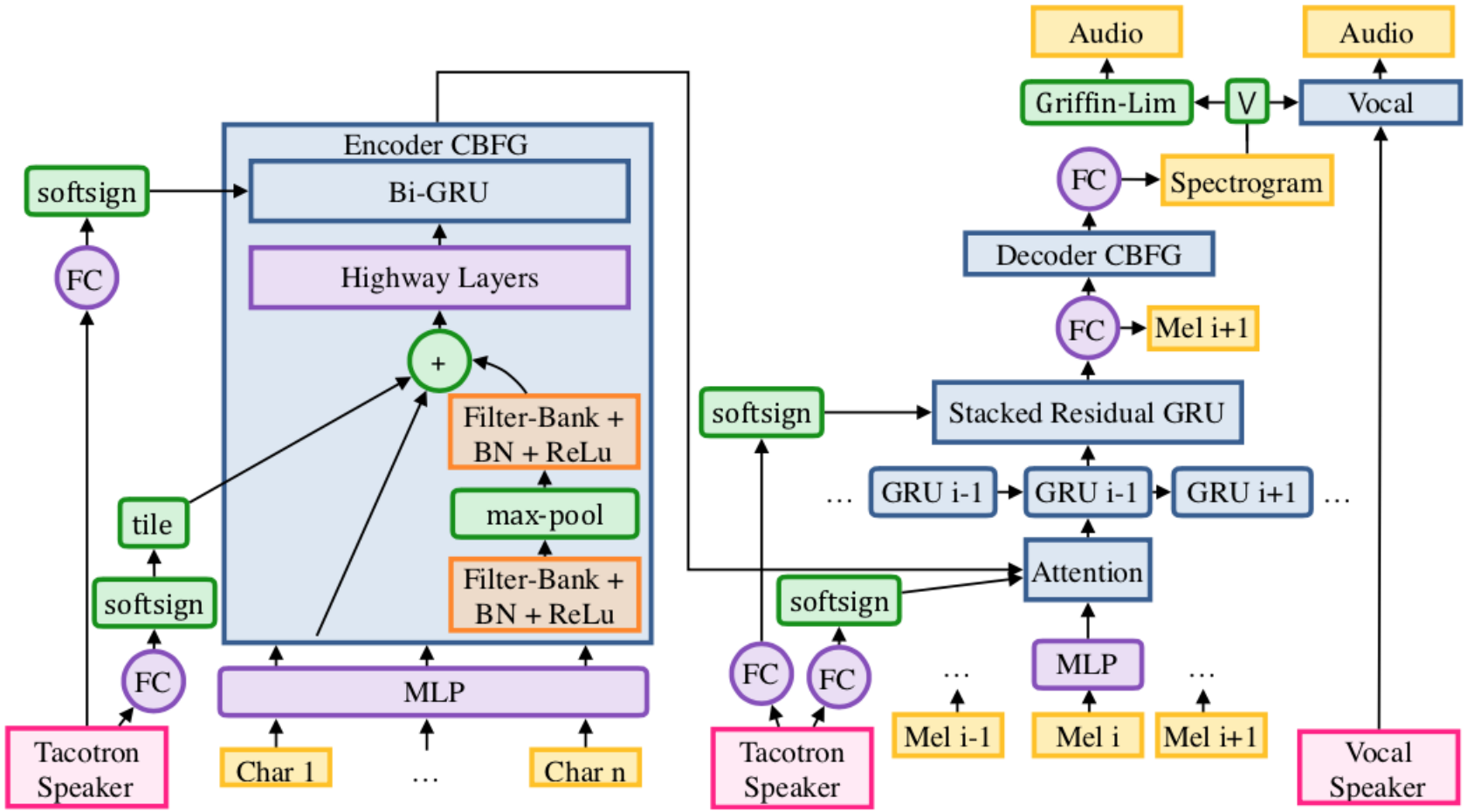
Тем самым удается достичь ускорения сходимости на порядок.



[13] H. Tachibana, K. Uenoyama и S. Aihara, "Efficiently Trainable Text-to-Speech System Based on Deep Convolutional Networks with Guided Attention", *arXiv preprint arXiv:1710.08969*, 2017.

Multi-Speaker TTS[14]

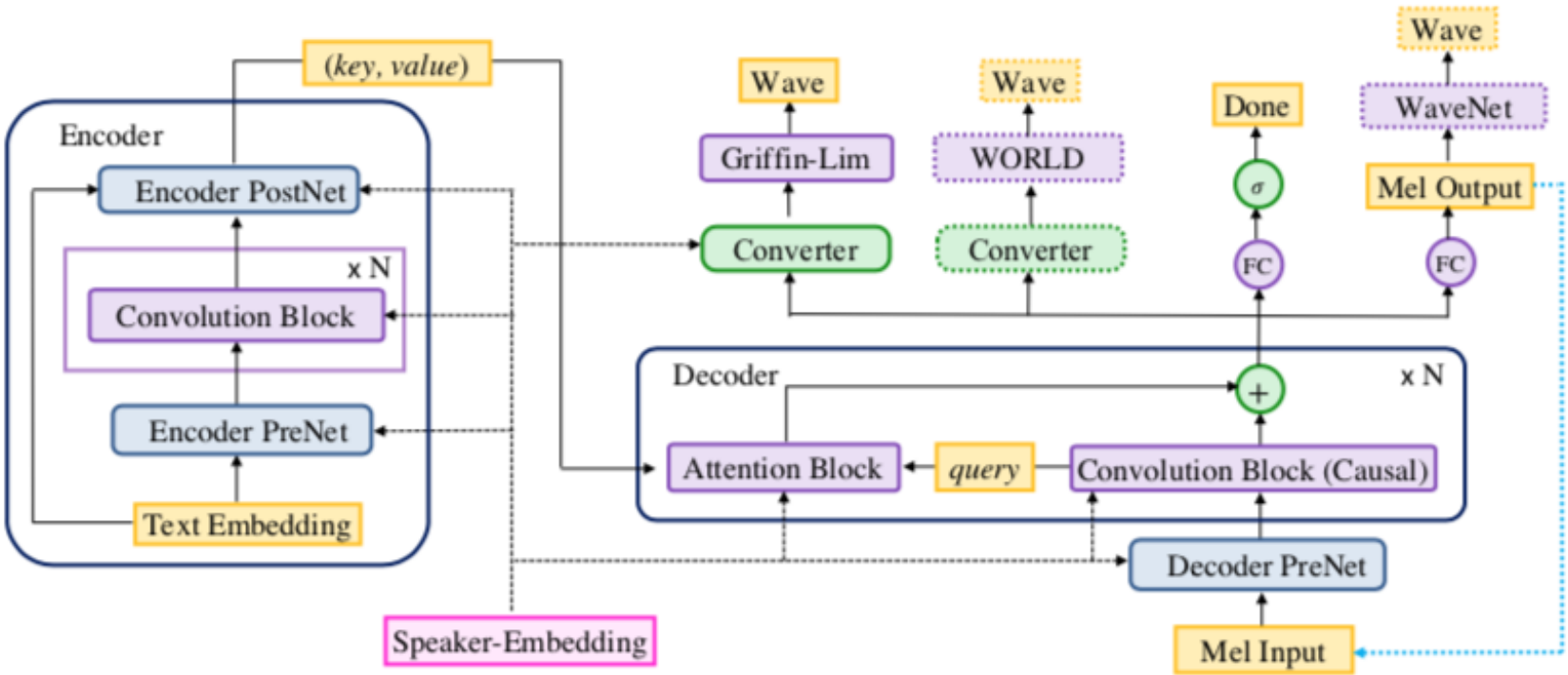
Авторы DeepVoice показывают, как сделать Tacotron Multi-Speaker, добавив Speaker Embedding.



[14] S. O. Arık, G. Diamos, A. Gibiansky, J. Miller, K. Peng, W. Ping, J. Raiman и Y. Zhou, "Deep voice 2: Multi-speaker neural text-to-speech", *arXiv preprint arXiv:1705.08947*, 2017.

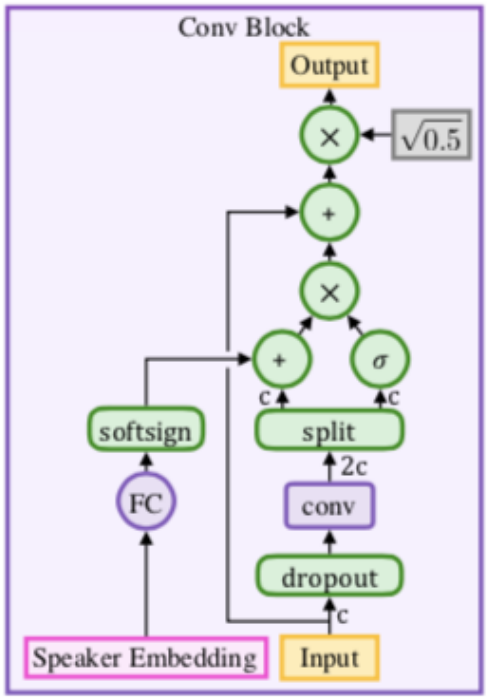
DeepVoice3[15]

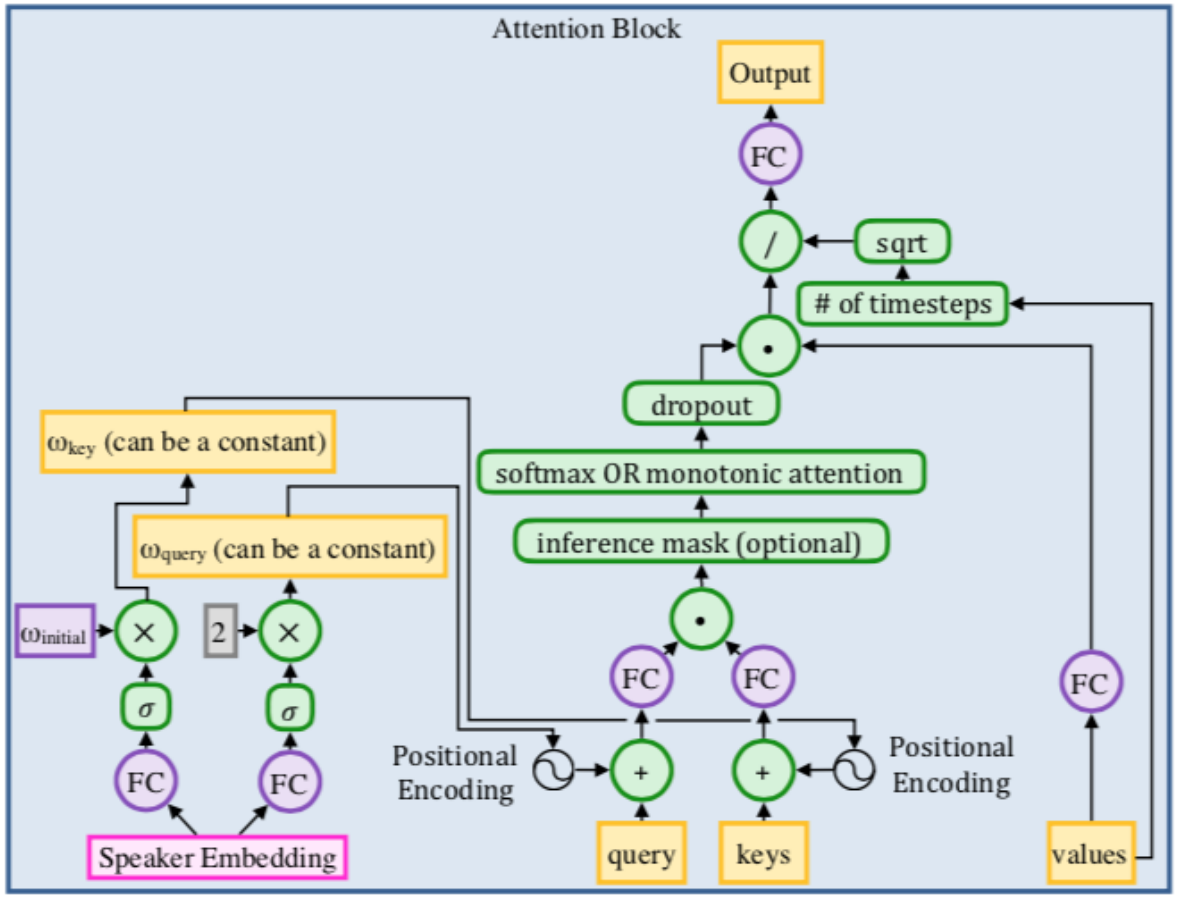
По качеству сопоставима Tacotron, но сходится в 10 раз быстрее (свертки вместо RNN).



[15] 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 preprint arXiv:1710.07654*, 2017.

DeepVoice3

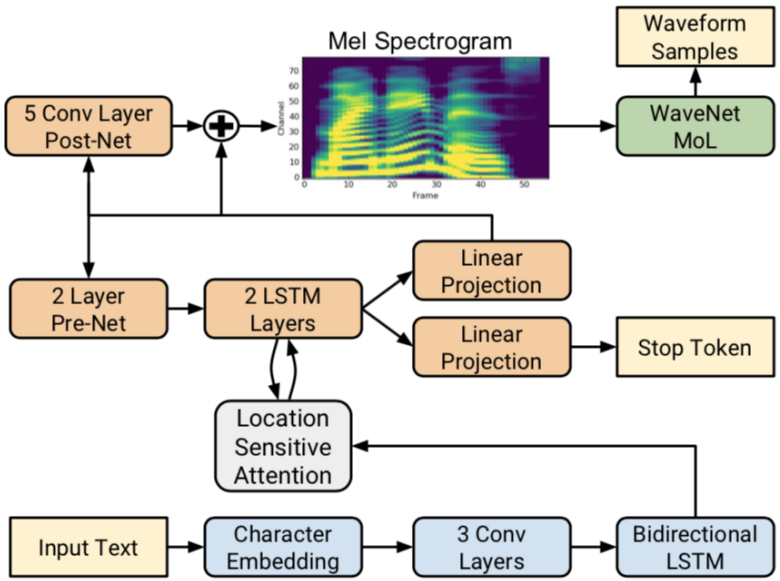




https://github.com/r9y9/deepvoice3_pytorch

https://github.com/r9y9/wavenet_vocoder

Tacotron2[16]



- рекуррентная seq2seq сеть с attention механизмом,
- post-processing net для mel-спектрограммы,
- wavenet вокодер (обучается отдельно)

[16] 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", *arXiv preprint arXiv:1712.05884*, 2017.