Генерация речи

Виктор Китов
victorkitov.github.io

Содержание

Введение

- Задача - генерация речи по тексту (speech synthesis, text-to-speech, TTS)
- Применения:
  - объявления в аэропортах, на вокзалах
  - интеллектуальные колонки (Яндекс.Станция)
  - виртуальные ассистенты в машинах, играх
  - чтение книг
  - обучение иностранному языку
  - помощь слепым, больным дислексией

Требования к выходу

- Выходная речь должна
  - быть понятной
  - не содержать шумов и артефактов
  - быть выразительной, эмоционально насыщенной
- Доп. возможности:
  - изменения типа голоса
    - спокойный, восторженный, шепот (есть в Алисе), ...
  - изменение спикера

Качество генерации

- Распознаваемость информации в речи:

$$WER(SpeechRecognizer(Generator(text)), text)$$

- Оценка качества речи респондентами
  - Mean Opinion Score[1] - средняя оценка качества
    - большая дисперсия из-за особенностей респондентов
  - MUSHRA (MUltiple Stimuli with Hidden Reference and Anchor)
    - респондентам дают прослушать эталонную запись (реальным голосом) и эталон
    - в число оцениваемых озвучиваний также вставляется скрытно эталон (можно проводить парный t-test)
  - SBS (side-by-side comparison) - респонденты сопоставляют 2 синтеза бок о бок (относит. шкала)
    - достаточно небольшого #данных для сопоставления

MOS, MUSHRA - абсолют. шкалы (можно сравнивать)

[1] CrowdMOS - оценка доверит. интервалов, верификация респондентов.

Скорость генерации

- Частота CD, mp3 - 44гГц: нужно выдавать 44100 значений в секунду.
- Метрики производительности TTS:
  - latency - задержка между получением данных и началом генерации
  - real-time factor (RTF) - сколько секунд занимает генерация 1 сек речи
    - для онлайн-приложений д. быть ≤ 1.

Датасеты

- Популярные датасеты:
  - LJ Speech, VCTK, M-AILABS, CommonVoice, OpenTTS
- В коммерческих применениях - записывают много часов целевого спикера
  - текст должен широко покрывать фонетические конструкции, разные интонации
  - хорошая студия, без шумов (важно для сужения неоднозначностей генерации)
    - т.к. генерация речи - one-to-many, и так много неоднозначности

Лингвистическое представление[2]

- Перед озвучиванием текст нормализуется в лингвистическое представление:
  - текст разбивается на предложения
  - цифры - в текст (5->"пять", "пятерых", "пятый")
  - раскрытие аббревиатур (кв.м.-> квадратный метр, квадратных метров)
  - раскрытие неоднозначности (зАмок - замОк)
  - разметка частей речи (для ударений: IMpact imPACT)
  - ударения (по словарю, но иногда зависят от контекста - зАмок/замОК)
  - е -> е/ё
- Решается seq2seq или трансформером.

[2] Neural Models for Text Normalization for Speech Applications.

Лингвистическое представление

- Преобразование графем в фонемы (яблоня->[й][а][б][л][о][нь][а])
  - не всегда нужна, можно интегрировать в акустическую модель
  - ищем слово в словаре, если нет - то модель предсказывает
    - модель можно тоже на словаре обучить
- Возможна расстановка пауз, длительностей, интонаций.
- Стандарт разметки - Speech Synthesis Markup Language.
  - нормализация текста, расстановка ударений, специйфикация спикеров, интонаций.

Содержание

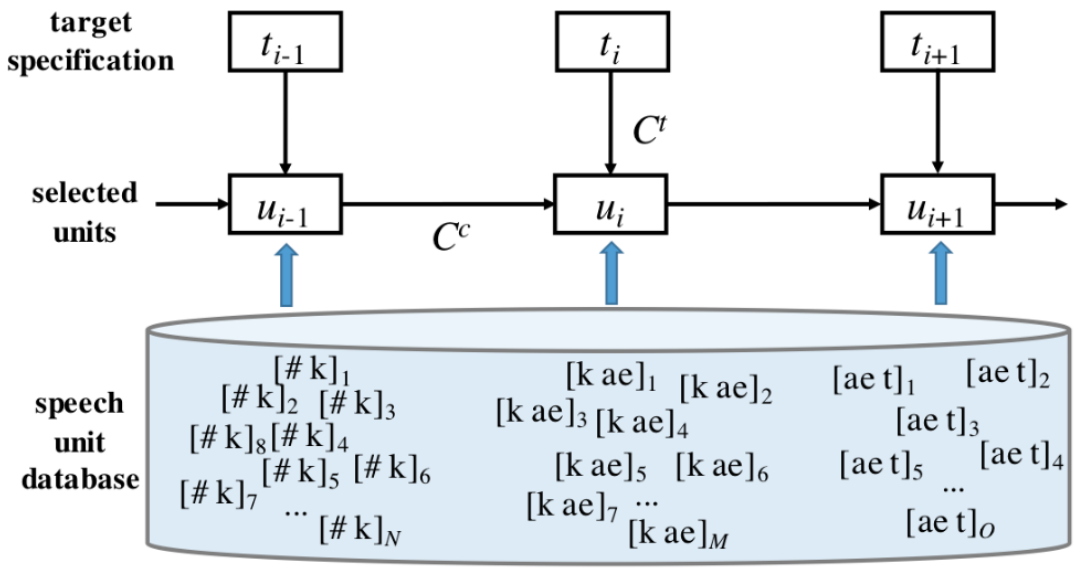
Объединяющий синтез[3,4]

- Объединяющий синтез (concatenative synthesis) - генерация речи предзаписанными блоками.
  - блоки-слова: бОльшая естественность, применимо в ограниченных доменах
    - например, объявления в аэропорту
  - блоки-фонемы (42-в русском, 44 в английском)
    - возможность произнести любое слово, но артефакты на стыках
  - блоки-дифоны (дифон-участок речи между серединами соседних фонем)
    - компромисс между универсальностью и естественностью
    - можно использовать трифоны

[3] Hunt, Black (1996).
[4] Deep Learning for Siri's Voice.

Схема работы

- Шаги объединяющего синтеза:
  - перевести текст в фонемы с разметкой
    - разметка: длительность, высота, громкость, интонация
  - сопоставление дифонам звуков (unit selection)
  - постпроцессинг: сглаживание на стыках



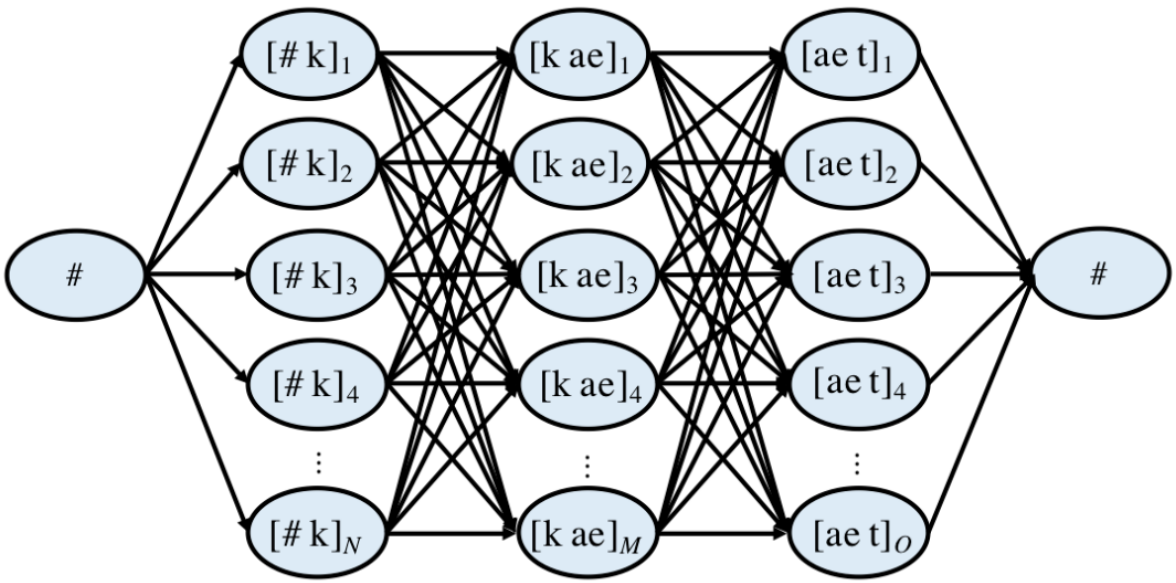
Стоимости сопоставления

- Обозначим:
  - t_i - фонема в момент i, $i = 1, 2...N$
  - u_i - выбираемый звук в момент i (юнит), всего S юнитов.
  - $C(u|t)$ - цена за выбор звука u для известной фонемы t
    - соответствие длительности, смысла, высоты, громкости, интонации
  - $C(u, u')$ - цена за выбор u' следующим звуком после u

Генерация синтеза - критерий

- Выбор наиболее естественного озвучивания $t_1, ...t_N$:

$$u_1^*, ...u_N^* = \arg\min_{u_1,...u_N} \left\{ \underbrace{\sum_{i=1}^{N} C(u_i|t_i)}_{\text{смысл}} + \underbrace{\sum_{i=1}^{N-1} C(u_{i-1}, u_i)}_{\text{сочетаемость}} \right\}$$



Генерация синтеза - алгоритмы

- Полный перебор путей - непрактично.
- можно решить за $O(S^2N)$ алгоритмом Витерби
  - можно $\downarrow S$ оставляя $u : C(u|t_i) \leq threshold$
- Приближенное решение: лучевой поиск.
  - обеспечивает real-time TTS

Алгоритм Витерби - обозначения

- $v_t(j) = \min_{u_1,...u_{t-1}} C(u_1, ...u_{t-1}u_t = j|x_1, ...x_{t-1})$ - мин. возможная цена, если последний звук j
- $p_t(j)$ - индекс оптимального звука в момент $t-1$, если в t был звук j.
- k_t - индекс оптимального звука в момент t

$$u_1^* = k_1,\ u_2^* = k_2,\ ...\ u_N^* = k_N$$

Алгоритм Витерби - шаги

Инициализация: $v_1(j) = C(u_1 = j|t_1)$

Алгоритм Витерби - шаги

Инициализация: $v_1(j) = C(u_1 = j|t_1)$
для $t = 2, 3, ...N$:

$$\begin{aligned} v_t(j) &= \min_{u_1,...u_{t-2}u_{t-1}} C(u_1, ...u_{t-2}, u_{t-1}, u_t = j|x_1, ...x_t) \\ &= \min_i \min_{u_1,...u_{t-2}} \{C(u_1, ...u_{t-2}, u_{t-1} = i) \\ &\qquad + C(u_{t-1} = i, u_t = j) + C(u_t = j|x_t)\} \\ &= \min_i \{v_{t-1}(i) + C(u_{t-1} = i, u_t = j) + C(u_t = j|x_t)\} \end{aligned}$$

Алгоритм Витерби - шаги

Инициализация: $v_1(j) = C(u_1 = j|t_1)$
для $t = 2, 3, ...N$:

$$\begin{aligned} v_t(j) &= \min_{u_1,...u_{t-2}u_{t-1}} C(u_1, ...u_{t-2}, u_{t-1}, u_t = j|x_1, ...x_t) \\ &= \min_i \min_{u_1,...u_{t-2}} \{C(u_1, ...u_{t-2}, u_{t-1} = i) \\ &\qquad + C(u_{t-1} = i, u_t = j) + C(u_t = j|x_t)\} \\ &= \min_i \{v_{t-1}(i) + C(u_{t-1} = i, u_t = j) + C(u_t = j|x_t)\} \end{aligned}$$

$$p_t(j) = \arg\min_i \{v_{t-1}(i) + C(u_{t-1} = i, u_t = j) + C(u_t = j|x_t)\}$$

Алгоритм Витерби - шаги

Инициализация: $v_1(j) = C(u_1 = j|t_1)$
для $t = 2, 3, ...N$:

$$\begin{aligned} v_t(j) &= \min_{u_1,...u_{t-2}u_{t-1}} C(u_1, ...u_{t-2}, u_{t-1}, u_t = j|x_1, ...x_t) \\ &= \min_i \min_{u_1,...u_{t-2}} \{C(u_1, ...u_{t-2}, u_{t-1} = i) \\ &\qquad + C(u_{t-1} = i, u_t = j) + C(u_t = j|x_t)\} \\ &= \min_i \{v_{t-1}(i) + C(u_{t-1} = i, u_t = j) + C(u_t = j|x_t)\} \end{aligned}$$

$$p_t(j) = \arg\min_i \{v_{t-1}(i) + C(u_{t-1} = i, u_t = j) + C(u_t = j|x_t)\}$$

$\min_{u_1,...u_N} C(u_1, ...u_N|x_1, ...x_N) = \min_j v_N(j)$

$k_N = \arg\min_j v_N(j)$; для $t = N, N-1, ...2$:

$$k_{t-1} = p_t(k_t)$$

Обсуждение

+: простота, интерпретируемость

+: расширяемость

- новые спикеры и манеры говорить

-: стыки фонем нельзя полностью удалить

-: высокие требования по памяти

- хранение звуков в разных интонациях/контекстах

-: учитывает лишь попарные, а не долгосрочные зависимости в интонации

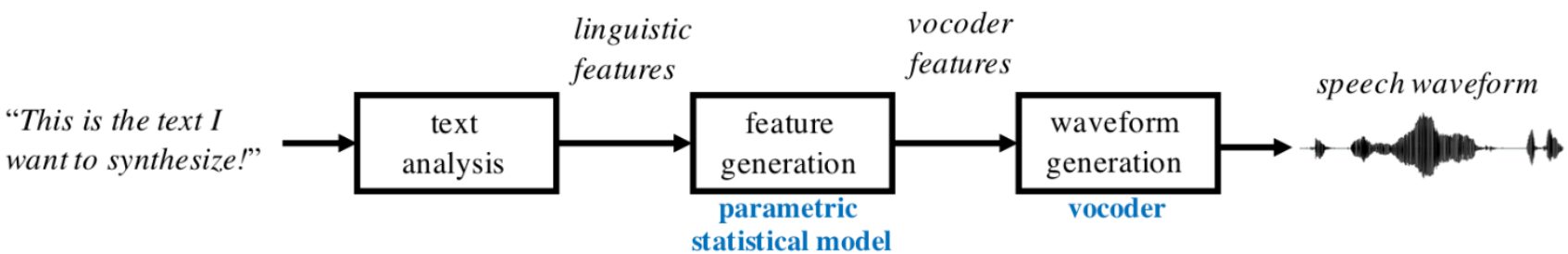
- монотонная речь

-: для изменения спикера, стиля, интонаций нужно всё перезаписывать

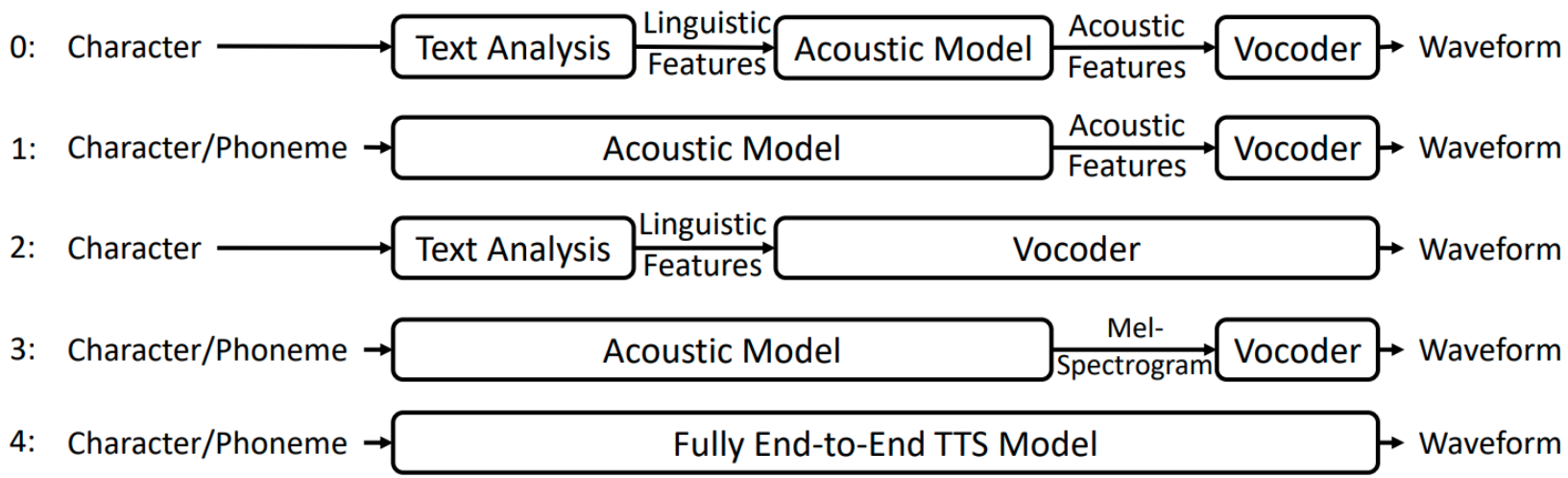
Содержание

Параметрическая синтез речи

- Параметрическая генерация речи (statistical parametric speech synthesis, SPSS)
- Использует DL модель для превращения текста в голос.
  - нет проблем со стыками
  - гибкость варьировать спикера, стиль, интонацию
- Линг. признаки - нормализованный текст из фонем
- Акустическая модель: фонемы->звуковые признаки для кажд. фрейма 10мс
  - спектрограмма, мел-спектрограмма, MFCC
  - ↑ размерности
- Вокодер это переводит в итоговый звук (↑ размерности)



Объединение шагов в моделях



Stage	Models
0	SPSS [416, 356, 415, 425, 357]
1	ARST [375]
2	WaveNet [254], DeepVoice 1/2 [8, 87], Par. WaveNet [255], WaveRNN [150], HiFi-GAN [23]
3	DeepVoice 3 [270], Tacotron 2 [303], FastSpeech 1/2 [290, 292], WaveGlow [279], FloWaveNet [163]
4	Char2Wav [315], ClariNet [269], FastSpeech 2s [292], EATS [69], Wave-Tacotron [385], VITS [160]

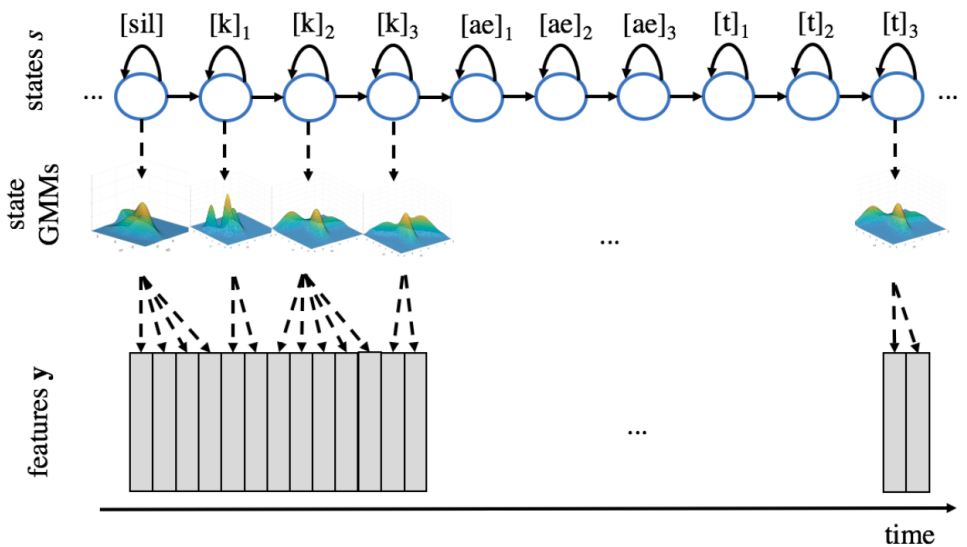
Обучение

Отдельно обучаются:

- Нормализация текста
  - например по распознанной речи спикера, прочитывающего текст
- Акустическая модель обучается на (нормализованный текст, мел-спектрограмма)
  - задает спикера, стиль, интонации, темп
- Вокодер обучается на (мел-спектрограмма, звук)
  - можно использовать неразмеченную речь

Скрытая модель Маркова

- Одна из первых акустических моделей - скрытая модель Маркова смеси Гауссиан (HMM-GMM)
  - s - фонемы, $P(s_t|s_{t-1})$ оцениваем по живой речи
  - выходн. речевые признаки $p(y|s) \sim \sum_k \phi_{k,s}\mathcal{N}(x|\mu_{k,s},\Sigma_{k,s})$
    - $\widehat{y}$ сэмплируем (лучевой поиск)
    - или $\widehat{Y} = \arg\max_Y P(Y|S)$
  - длительность фонемы - сколько пробыли в состоянии
    - или отдельная модель длительности (duration model)



Особенности генерации речи

- Генерация текст->звук напрямую не используют:
  - слишком сильное $\uparrow$ размерности.
- Даже на 2x шагах:
  - Акустическая модель может по-разному произнести текст
  - вокодер по разному восстановить фазы
- Вокодеру нужно "додумать":
  - информацию о фазах (теряется при переходе к спектру)
  - полный спектр по усредненным спектрам mel-спектрограммы

Алгоритм Гриффина-Лима

Алгоритм Гриффина-Лима[5] - вокодер без параметров.

- не содержит параметров
- итеративная схема восстановления спектра (компоненты, фазы) по его модулям
  - стартуя со случайных инициализаций неизвестных пар-ров
- плохо работает на мел-спектрограмме ("металлический" голос)

[5] Griffin, Lim (1984).

WaveNet[6]

- WaveNet - генеративная модель для звука:

$$p(x_t x_{t-1}, ...x_1) = p(x_t|x_{1:t-1})p(x_{t-1}|x_{1:t-2})...p(x_1)$$

- Упрощение - смотрим только на K шагов назад:

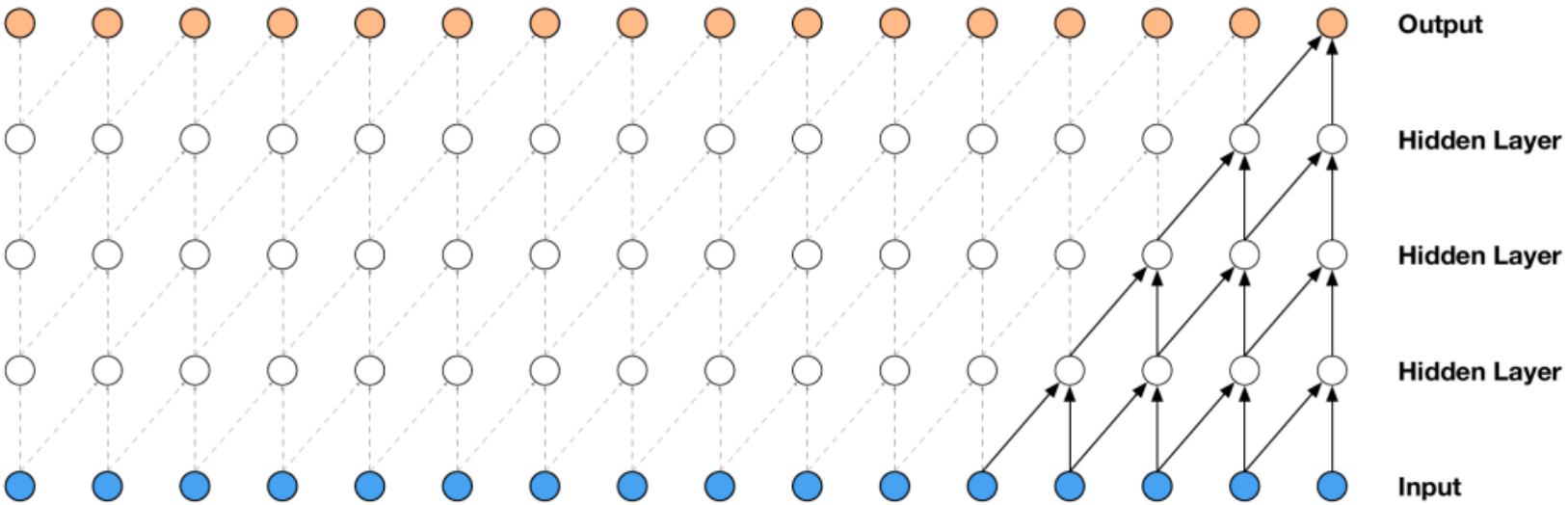
$$p(x_t x_{t-1}, ...x_1) = p(x_t|x_{t-1-K:t-1})p(x_{t-1}|x_{t-2-K:t-2})...p(x_1)$$

- Вход: эмбеддинг x_t
- Выход - SoftMax-вероятности 256 классов.
  - 8-битная интенсивность сигнала, квантизованного по μ-закону.

[6] https://arxiv.org/pdf/1609.03499.pdf

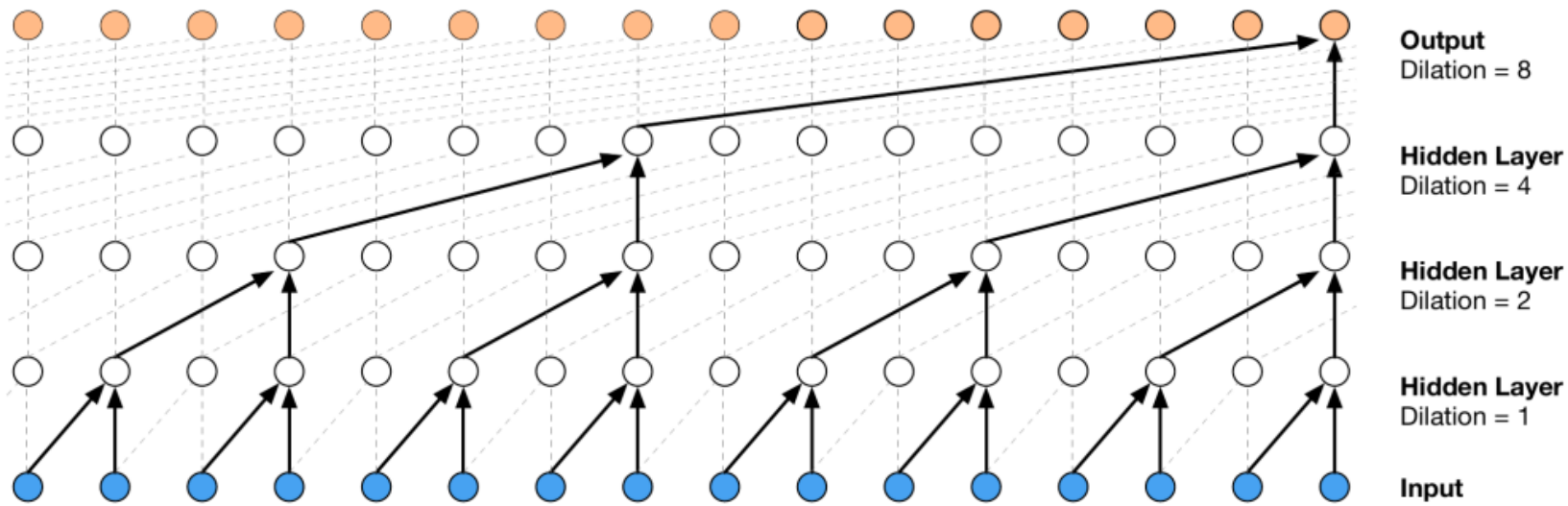
Dilated convolution

- Для моделирования $p(x_t|x_{t-1}, ...x_{t-K})$ используется свёртка по истории (causal conv).
- У обычной свёртки - малая область видимости:



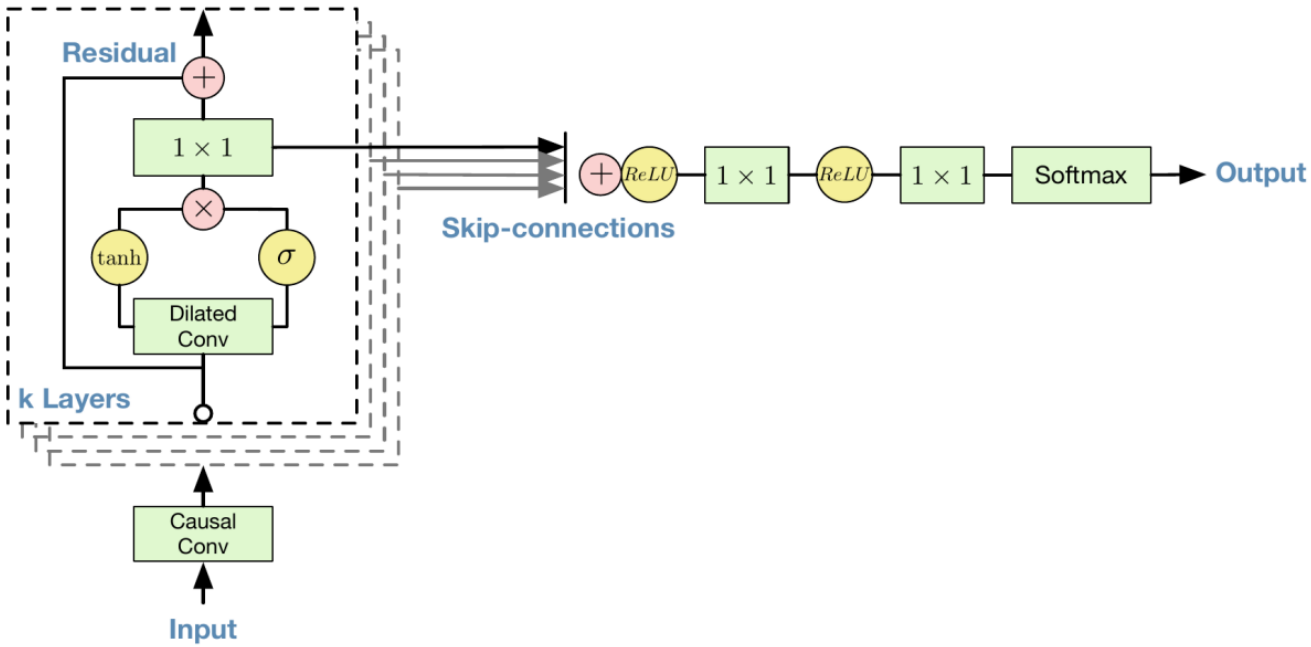
Dilated convolution

- Для моделирования $p(x_t|x_{t-1}, ...x_{t-K})$ используется свёртка по истории (causal conv).
- Поэтому используется свёртка с прореживанием:



- Обучение - параллельное по t (знаем таргеты)
- Генерация - последовательная
  - зато можно переиспользовать ранее посчитанные активации.

Архитектура



- Блоки из прореженных свёрткой наслаиваются k раз, образуя области видимости блока:

$$1, 2, 4, ..., 512, 1, 2, 4, ..., 512, 1, 2, 4, ..., 512$$

- Прогноз - по сумме выходов из нескольких веток.

Условная генерация

- В свёрточном блоке безусловной генерации вместо ReLU используется ($*$ - свёртка, $\odot$ - поэлем. умножение)

$$\mathbf{z} = \text{tangh}\left(W * \mathbf{x}\right) \odot \sigma\left(\tilde{W} * \mathbf{x}\right)$$

- В глобально условной генерации (h - эмбеддинг глоб. условия (напр. спикера))

$$\mathbf{z} = \text{tangh}\left(W * \mathbf{x} + V\mathbf{h}\right) \odot \sigma\left(\tilde{W} * \mathbf{x} + \tilde{V}\mathbf{h}\right)$$

- В локально условной генерации (y - эмбеддинг лок. условия (напр. произносимой сейчас фонемы))
  - Upsample(y) - повторенная нужно #раз фонема
  - $B*$ и $\tilde{B}*$ - 1x1 свёртки

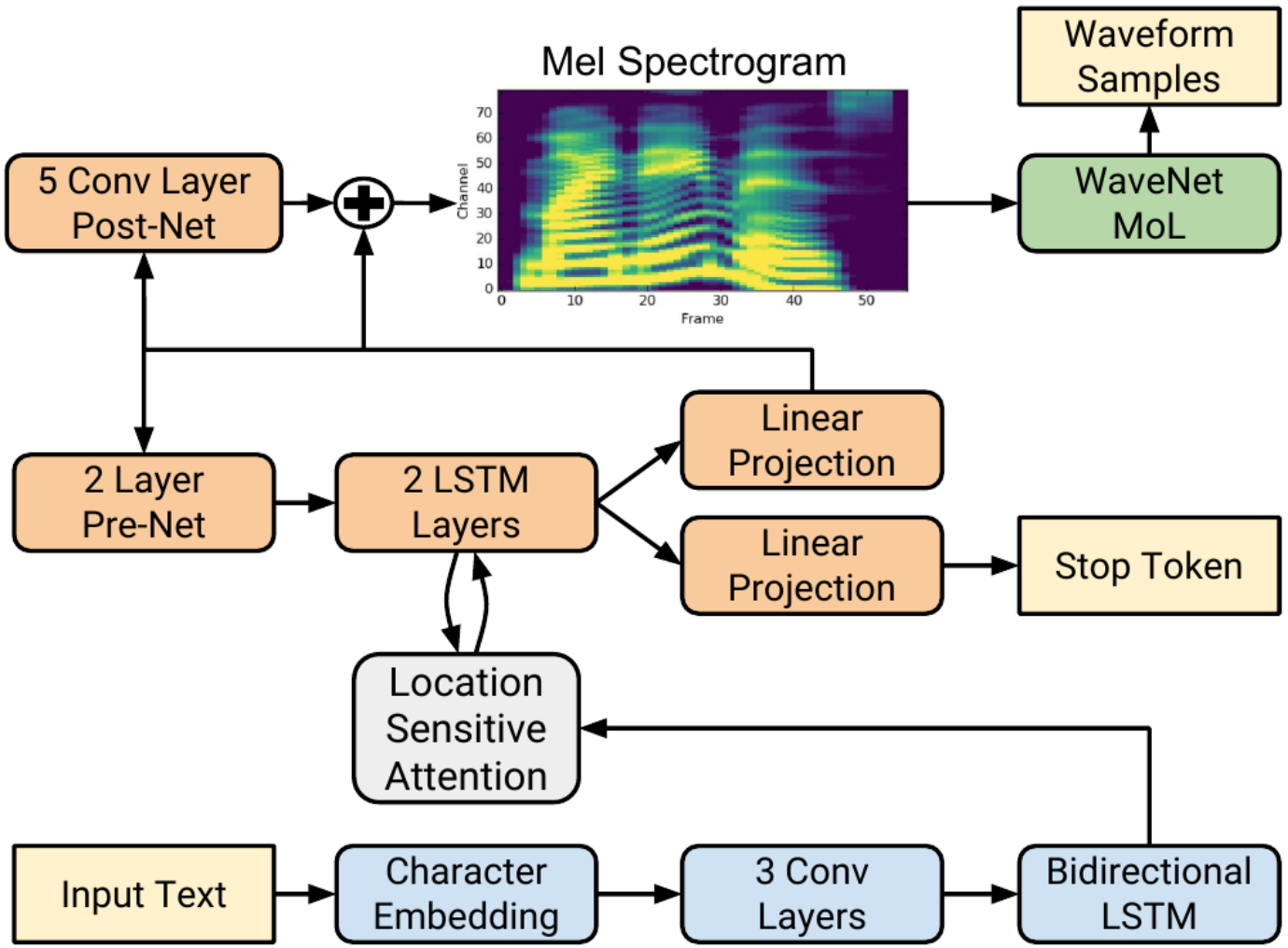
$$\mathbf{z} = \text{tangh}\left(W * \mathbf{x} + B * \text{Upsample}(y)\right) \odot \sigma\left(\tilde{W} * \mathbf{x} + \tilde{B} * \text{Upsample}(\mathbf{y})\right)$$

Tacotron 2[7,8]

- WaveNet требует лингвистические признаки, F_0, длительности фонем.
- Tacotron 2 генерирует по тексту сразу мел-спетрограмму, озвучиваемую условным WaveNet
  - качество звука выше, чем у WaveNet напрямую
- Архитектура: кодировщик - рекуррентный декодировщик со вниманием.
- Генерация продолжнается авторегрессионно, пока выход сети StopToken не станет $\geq threshold$.

[7] https://arxiv.org/pdf/1712.05884.pdf
[8] https://habr.com/ru/company/nix/blog/436312/

Архитектура



Архитектура

Кодировщик:

1. входные символы -> эмбеддинги $\in \mathbb{R}^{512}$
2. 3 свёртки 5×1 (каждая смотрит на 5 соседних символов)
   - после каждой: батч-нормализация, затем ReLU
3. Двунаправленная LSTM -> выходы кодировщика

Архитектура

Декодировщик:

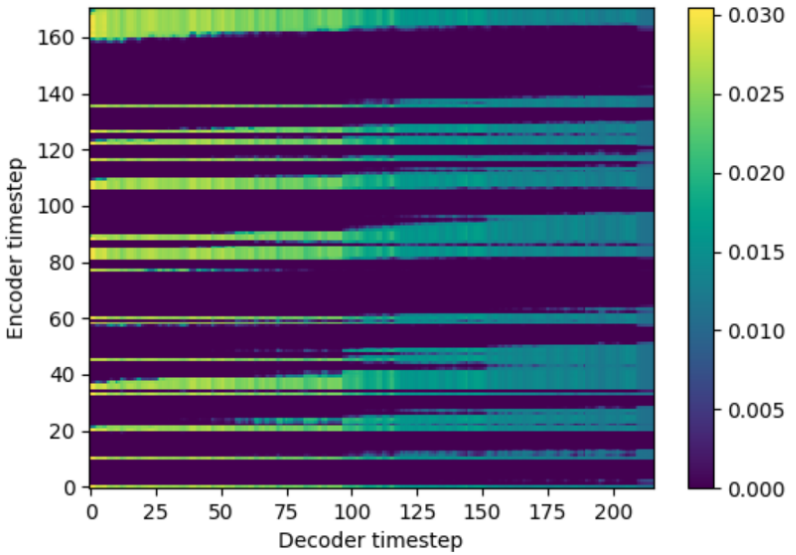
1. Две однонаправленные LSTM+внимание, учитывающее локацию
2. Выход LSTM подаётся 2м полносвязным слоям с малым #нейронов (information bottleneck)

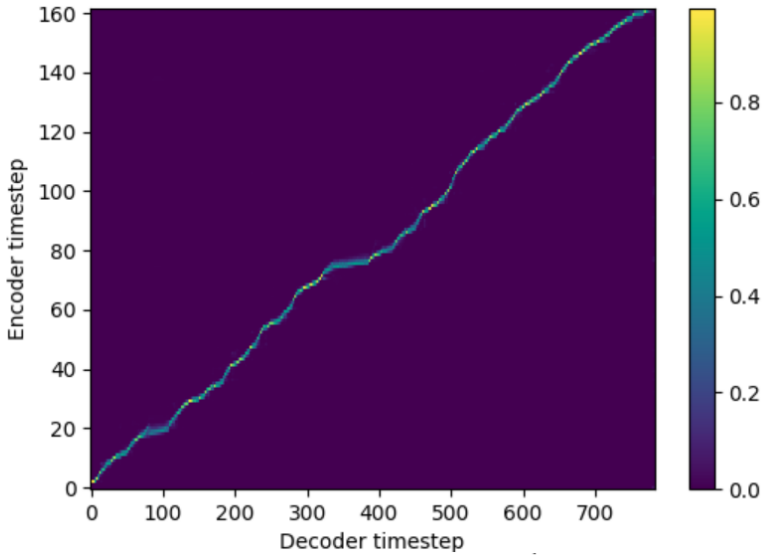
Регуляризации декодировщика:

- dropout свёрточных слоёв (при inference тоже - для вариабельности выходов)
- пересчёт состояния LSTM - zoneout (с опр. вероятностью передаём состояние 2 шага назад)

Location sensitive attention

Правильно и неправильно обученное внимание:





Внимание, учитывающее локацию

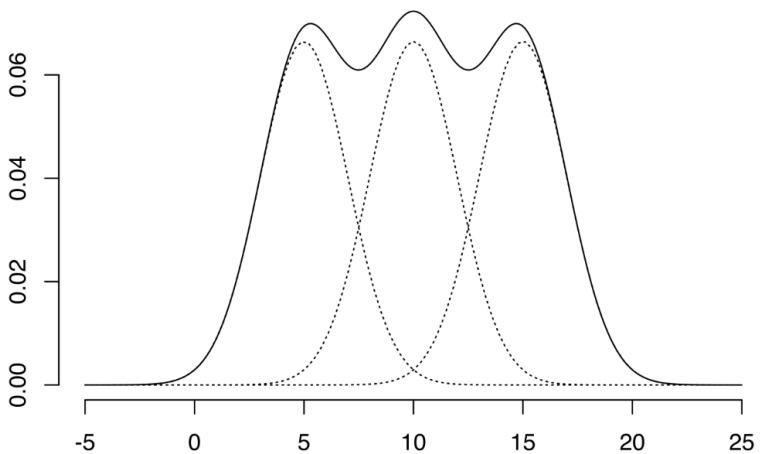
- В декодере Tacotron 2 для содействия обучению используется внимание, учитывающее локацию (location-sensitive attention)[9].
  - учитывает, куда смотрели на предыдущем шаге
- По входам $x_1, ...x_N$ извлечем признаки $f_1, ...f_N$ $(F \in \mathbb{R}^{D\times N})$

Внимание, учитывающее локацию:

$$\begin{aligned} e_{tj} &= \text{score}\left(s_{t-1}, h_j, \right) \\ &= w^T \tanh\left(W s_{t-1} + V h_j + UF\alpha_{t-1} + b\right) \\ \alpha_{ti} &= \exp(e_{ti}) / \sum_j \exp(e_{tj}) \\ c_t &= \sum_j \alpha_{tj} h_j \\ s_t &= f\left(s_{t-1}, y_{t-1}, c_t\right) \end{aligned}$$

Вокодер

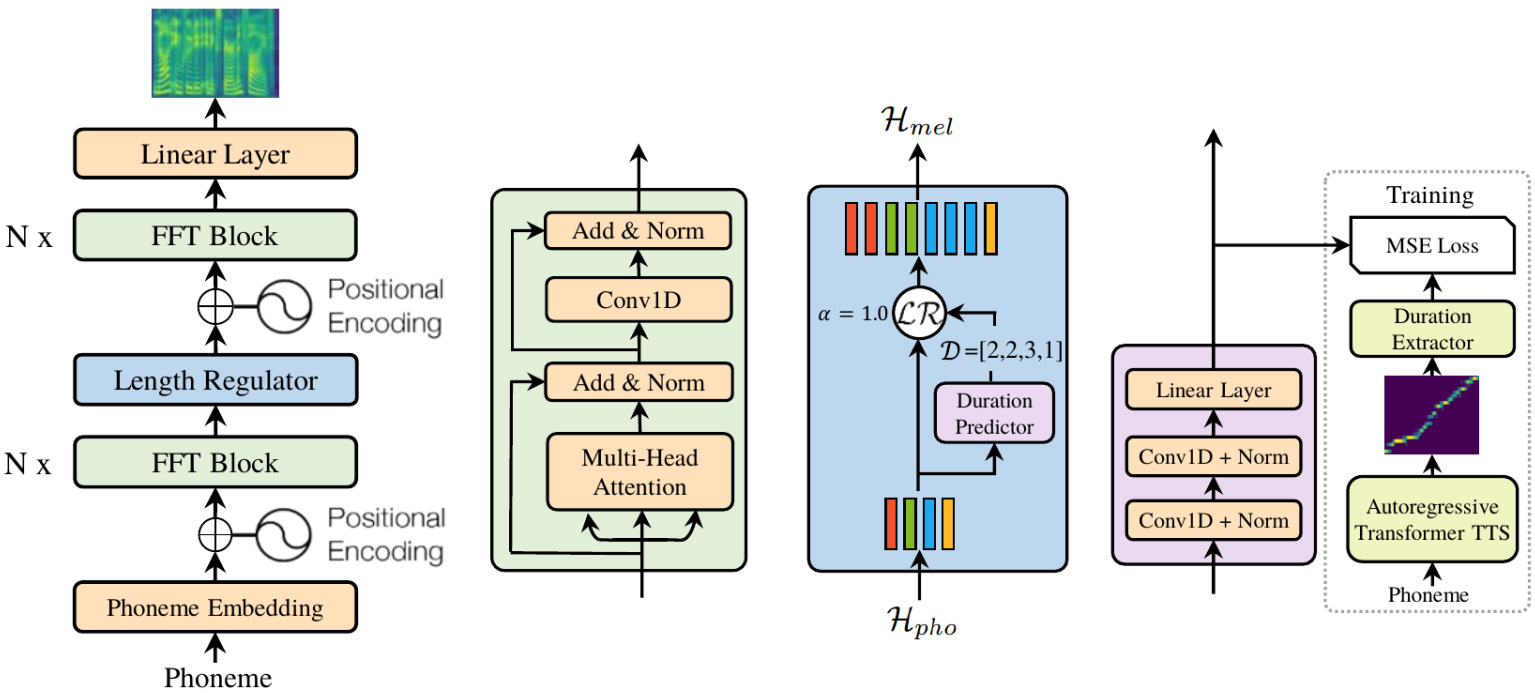
- В качестве вокодера использовался WaveNet
- Но WaveNet предсказывал интенсивность не через SoftMax, а через смесь 10 логистических распределений (ф-ция распределения - $\sigma(\cdot)$).
- Tacotron 2+WaveNet показал лучше качество по MOS, чем WaveNet на лингвистических признаках.



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

FastSpeech[10]

- FastSpeech - акустическая модель параллельной генерации мел-спектрограммы.
  - вокодер WaveGlow



(a) Feed-Forward Transformer (b) FFT Block (c) Length Regulator (d) Duration Predictor

[10] https://arxiv.org/pdf/1905.09263.pdf

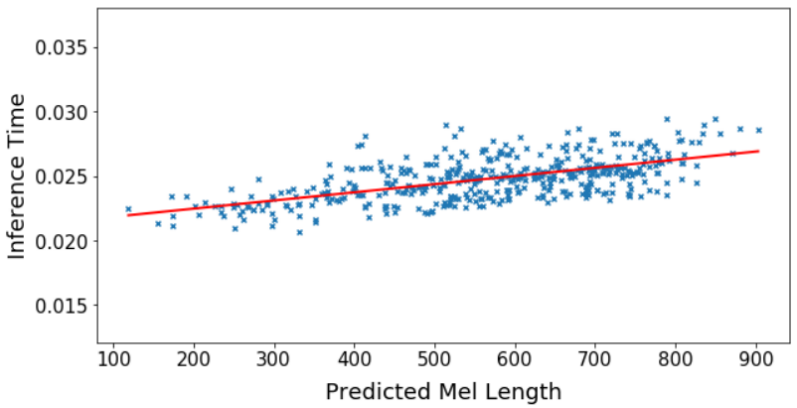
FastSpeech

- Параллельная генерация мел-спектрограммы.
- Каждая фонема дублируется через duration predictor
  - нельзя пропустить фонемы и немонотонно по ним пройти (проще настроить, чем модели с последоват. вниманием)
  - можно вручную контролировать:
    - общий темп речи
    - длительность пауз между словами и предложениями
- Duration предиктор обучается через внимание др. авторегрессионной TTS модели с вниманием.

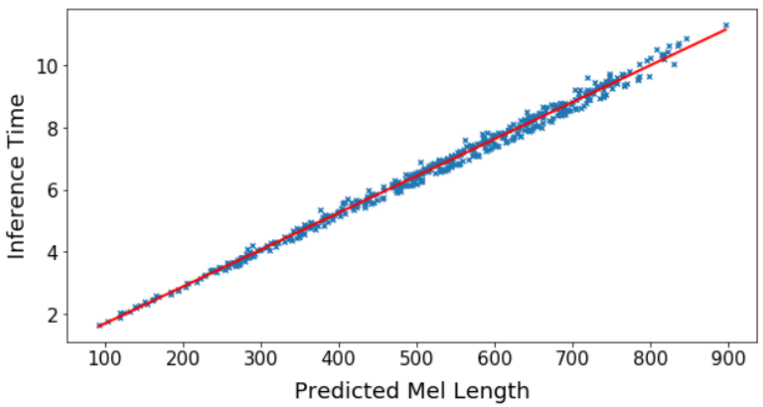
FastSpeech

Ускорение за счёт параллельной генерации мел-спектрограммы:

Method	Latency (s)	Speedup
Transformer TTS [14] (Mel)	6.735 ± 3.969	/
FastSpeech (Mel)	0.025 ± 0.005	$269.40\times$
Transformer TTS [14] (Mel + WaveGlow)	6.895 ± 3.969	/
FastSpeech (Mel + WaveGlow)	0.180 ± 0.078	$38.30\times$



(a) FastSpeech

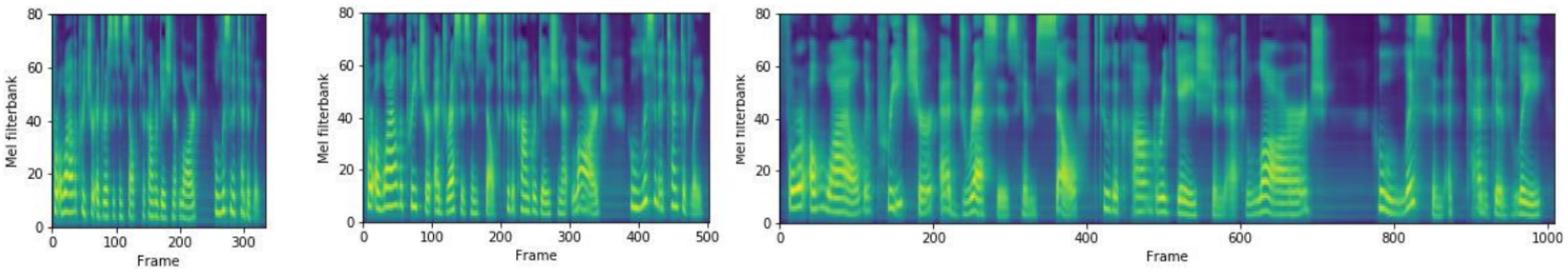


(b) Transformer TTS

Inference time (second) vs. mel-spectrogram length for FastSpeech and Transformer TTS.

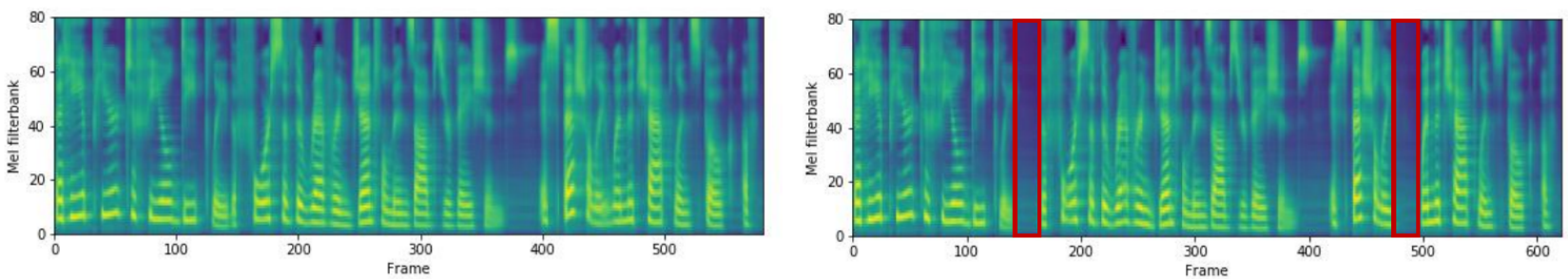
Возможность управления синтезом

- Масштабируя выход duration predictor, можно управлять скоростью речи:



(a) 1.5x Voice Speed (b) 1.0x Voice Speed (c) 0.5x Voice Speed

- Можно изменять только длительность пауз:



(a) Original Mel-spectrograms (b) Mel-spectrograms after adding breaks

Заключение

- Подходы к синтезу речи:
  - объединяющий синтез речи (concatenative)
    - простота, расширяемость
    - проблемы на стыках, монотонная речь, сложности при изменении темпа, стиля голоса
  - параметрический синтез речи (statistical parametric)
    - непрерывная генерация без стыков
    - возможность варьировать стиль речи
    - сложная настройка
    - включает акустическую модель и вокодер
- WaveNet - используется как вокодер.
- Акустические модели: Tacotron 2 (качественнее), FastSpeech (быстрее, проще настройка).