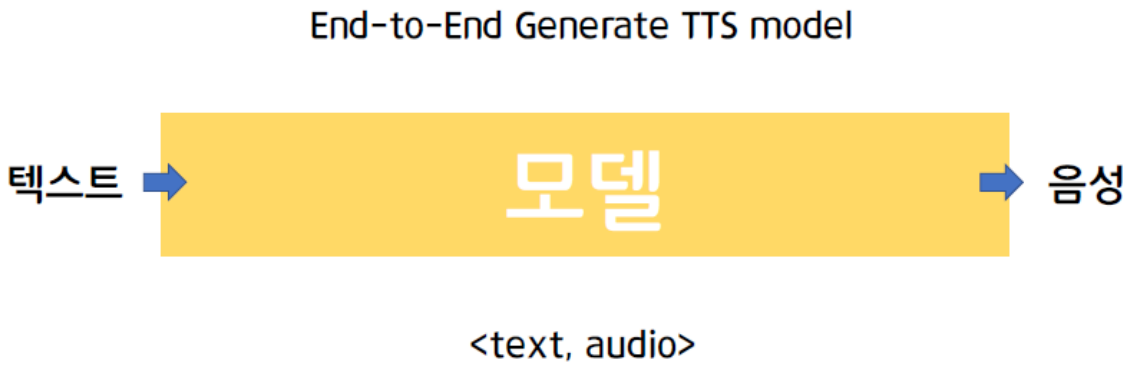
Tacotron1 : 논문 리뷰

0 Abstract

- TTS : text to speech
- 원래 text 를 speech 로 합성하는 시스템은 여러 단계로 구성이 됩니다.
    - 텍스트 분석
    - 음향 모델
    - 오디오 합성모듈 등등
- 이러한 구성요소를 구축하려면 광범위한 domain 지식이 필요합니다.
- 그래서 이 논문에서는 문자로 부터 직접 음성을 합성하는 end-to-end generate text-to- speech 모델인 타코트론을 제시합니다.
- <text , audio> pair 를 이용해서 end to end로 학습이 가능하다.
- tacotron 은 주관적인 5등급 평균 의견점수 3.82점을 획득하여, 자연성 측면에서 생상 파라메트릭 시스템을 능가한다.
- 또한 타코트론은 프레임 레벨에서 speech 를 생성하므로 샘플레벨 자동 회귀 방식보다 훨씬 빠르다.



1 Introduction

- 현대의 TTS 파이프 라인은 복잡하다. (2009년 꺼)
- TTS 는 a text frontend extracting various linguistic features, a duration model, an acoustic feature prediction model and a complex signal-processing-based vocoder 이것들을 가지고 있어야 했다.
- 그리고 이런 구성요소들은 도메인 전문지식이 필요하고, 디자인하는데 힘들다.
- 또한 독립적으로 훈련되므로 각 구성요소의 오류가 더 심각해질수 있다. (error 의 누적)

- 따라서 최소한의 인간의 annotation(주석) 으로 <text, audio> 쌍을 훈련할수 있는 통합된 end to end 시스템은 많은 장점이 있다.
    - feature engineering이 간단하다. (디자인이 간단하다.)
    - 발화자, 언어, 감정 등의 feature 를 쉽게 조절할수 있다.

        - 이게 가능한 이유는, 특정 구성요소에 국한 되지 않고, 모델의 맨 처음에서 일어날수 있기 때문이다.
    - 새로운 데이터에 더 잘 적응한다.
    - 단일 모델은 각 오류가 복합될수 있는 multi stage model 보다 더 견고하다. (오류누적 X)
- 이런 장점들은 end to end 모델이 현실 세계에서 발견되는 noisy data 를 잘 훈련시킬수 있다는것을 의미한다.

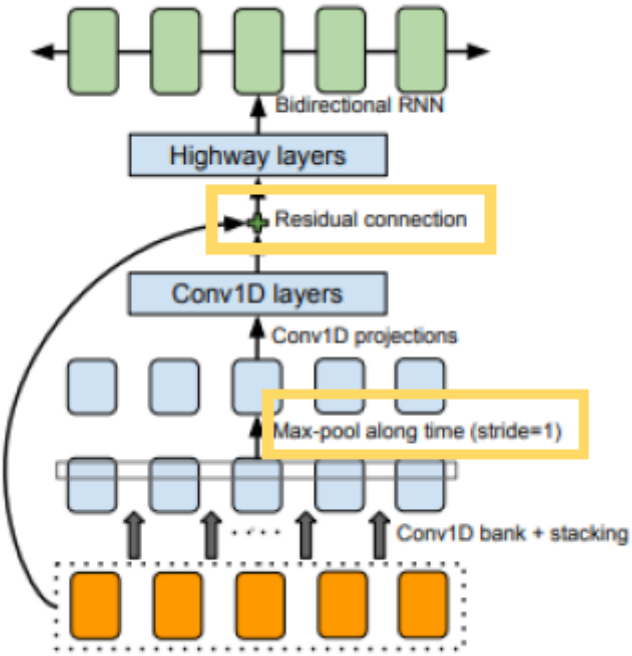
- TTS 는 large-scale inverse problem 이 있다.
    - 고도로 압축된 소스를 오디오로 압축을 푼다.
- 동일한 텍스트가 다른 발음이나 말하기 스타일에 대응될수 있기 때문에, 이것은 end to end 모델에서 특히 어려운 학습이다. 주어진 입력에 대한 신호 수준에서 큰 변동에 대처해야한다.
- 또한, end to end 음성인식이나 기계번역과는 달리, TTS는 출력이 연속적이며 일반적으로 입력보다 출력이 더 길다.
    - 이런 속성은 예측 오류를 빠르게 누적시킨다.
- 그래서 이 논문에서는 attention과 seq2seq 에 기초한 end to end 모델 Tacotron을 제안한다.
- tacotron 모델은 문자를 입력으로 사용하고, spectrogram 을 출력하며, seq2seq 모델의 성능을 향상시키기 위해 몇가지 기법을 사용한다.
- <text , audio> pair로 쌍이 주어진다면 타코트론은 random initialize로 처음부터 완전히 학습될 수있다.
- 음소 수준의 정렬(phoneme-level alignment) 이 필요 없어 대화록이 있는 대량의 음향 데이터를 사용하는것으로 쉽게 확장할수 있다.

2 related work

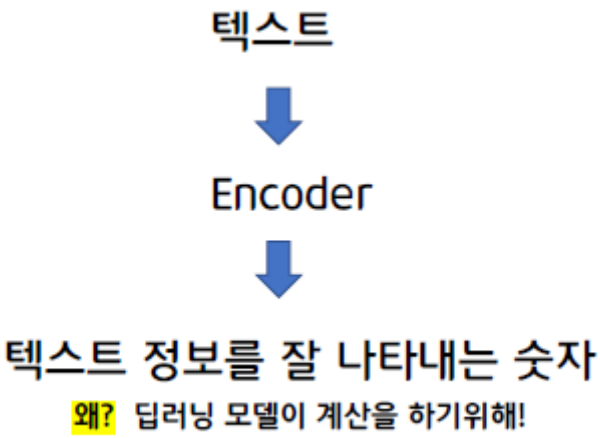
1. WaveNet
    - 느리다. (due to its sample-level autoregressive nature)
    - TTS로 바로 활용할 수 없으며 TTS-Frontend로부터 Linguistic Feature를 입력으로 넣어줘야 하는 단점이 있다.
    - not end to end
    - vocoder 로 사용될수 있다.
2. deep voice
    - TTS 파이프라인에 있는 모든 구성요소를 해당 신경망으로 대체하는 딥보이스
    - 각 구성요소는 독립적으로 훈련되며, end to end 방식으로 훈련되지 않는다.
3. Char2Wav
    - end to end 모델이지만
    - 여전히 , SampleRNN neural vocoder 를 사용하기 전에 vocoder parameter 를 예측한다. 출력이 Audio 자체가 아닌 Vocoder 파라미터를 예측하는 방식이다. 따라서 Vocoder의 성능에 따라 결과가 달라지게 된다.
    - Tacotron 은 이걸 개선시켰다.

3 Model Architecture

- Tacotron 의 backbone은 seq2seq 과 attention 이다.
- 인코더 부분과, attention 기반인 decoder 로 이루어져있다.

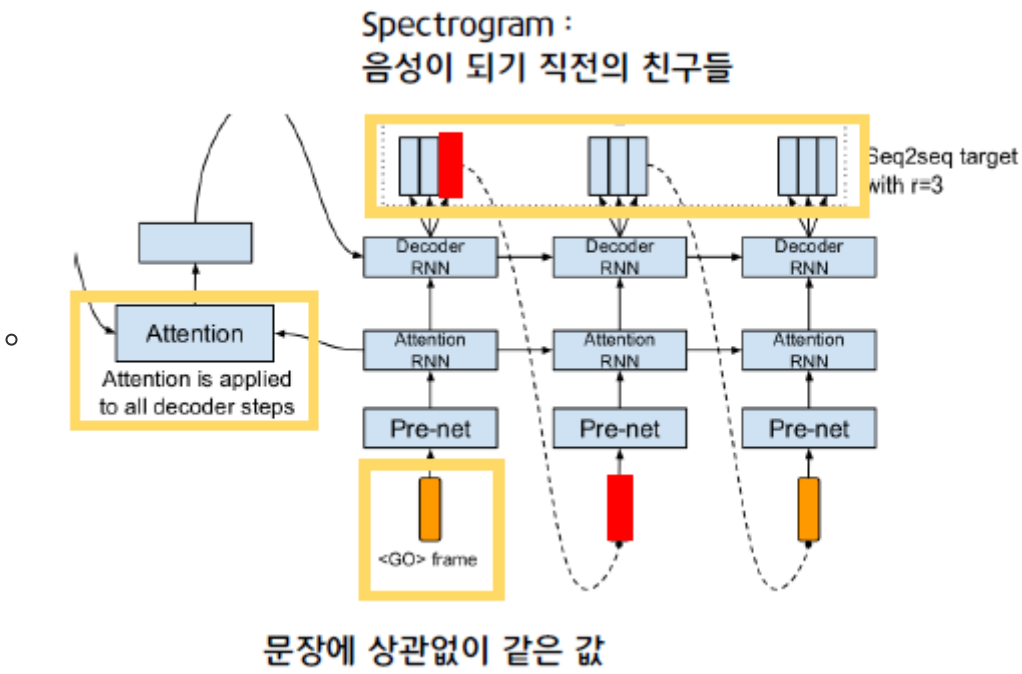
1. CBHG model
   - 

   - 부분적인 특징을 잡아내는 CNN과 연속적인 DATA 를 생성하기좋은 RNN의 특징을 모두 활용!
   - 1D convolution bank highway network GRU
   - reduces overfitting, makes fewer mispronunciation

2. Encoding
   - 

   - 인코딩 이라는것은 , 한글 또는 영어로 구성된 문장의 정보를 가장 잘 나타내는 숫자의 배열로 나타내주는것
   - 인코더는 입력 문자 임베딩 열을 받아 annotation vector를 출력하는 부분.
   - 문자 임베딩 열은 각 문자를 one hot 벡터로 변환한뒤, 연속 벡터로 변환한 결과를 나열한것이다.
   - 인코더의 역할은, 텍스트 정보를 잘 나타내는 숫자로 바꿔주는 역할!
   - 왜? 딥러닝 모델이 계산을 하기 위해!
   -

3. decoding



- mel spectrogram 생성(음성이 되기 직전의 숫자들)
- 이전에 만든 스팩트로그램을 다시 입력으로 넣어서 다음 스팩트로그램을 생성하는데 사용
- 한번에 N 개의 spectrogram을 예측한다. (overlap 아님)
  - reduction facter 라고 함. (r)
  - 이를 통해 모델의 사이즈, training time을 줄일수 있다.
- 문장이 끝날때까지 반복된다!
-

4. attention
   - 학습되지않은 문장에 대해 얼마나 좋은 성능을 보여줄수 있는가?
   - 지금 내가 말하고자 하는 text 어디에 집중하면 될까?
   -
5. Vocoder
   - Griffin Lim : 스팩트로그램을 음성으로 만들어주는 알고리즘
   -

4 Model Detail

- 24KHz 샘플링 속도를 사용했다.
- r 이 클수록 효과가 좋다.
- Adam optimizer 를 사용
  - learning rate decay : 0.001 에서 0.0005, 0.0003 , 0.0001이렇게..
- loss : mel spectrogram 과 linear spectrogram을 사용해서 구함
-

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

Hann windowing
50ms frame length
12.5ms frame shift
2048 point Fourier transform
24kHz sampling rate

Adam optimizer
Loss = mel_loss + linear_loss
Mel_loss = tf.reduce_mean(th.abs(mel_out − mel_target))
linear_loss = tf.reduce_mean(th.abs(mel_out − mel_target))