Wave-Tacotron: Spectrogram-free end-to-end text-to-speech synthesis

Ron J. Weiss, RJ Skerry-Ryan, Eric Battenberg, Soroosh Mariooryad, and Diederik P. Kingma
{ronw, rjryan, ebattenberg, soroosh, durk}@google.com

**Overview**

**Summary**
- TTS* in one sequence-to-sequence model
  - block-autoregressive normalization flow, no vocoder
  - "normalized-text" or phoneme-to-speech
- Directly predict 48ms waveform blocks at each decoder step
  - no overlap, no spectrograms
- End-to-end training, maximizing likelihood
- High fidelity output
  - trails Tacotron+WaveRNN baseline
  - higher sample variation, captures modes of training data?
  - ~10x faster than real-time synthesis on TPU

**Background**
- Tacotron [1]: phoneme input, mel spectrogram frame output
  - autoregressive decoder, each step generates new frame
  - e.g., WaveRNN [3], sample-by-sample autoregressive
- Wave-Tacotron: generate sequence of non-overlapping waveform blocks
  - K = 960 samples (40 ms at 24 kHz)

**Architecture**
- Replace decoder pre-net and vocoder with conditional normalization flow
  
  \[ P(y_t | x_t) = P(y_t | x_{t-1}, y_{t-1}) \]

  where \( J = K / L \)

  - Tacotron encoder/dencoder predicts flow conditioning input
  - Train end-to-end, maximize likelihood of training data
  - Block-autoregressive generation
    - waveform samples in each block generated in parallel

**Multiscale network [7]**
- Squeeze waveform block into frames, length L = 10 samples
  - M = 6 stages, each processes signal at different scale
  - N = 12 steps per stage
  - deep convnet: \( M \times N = 60 \) total steps
  - sinusoidal position embeddings encode position in each frame

**Data**
- US English, single female speaker, sampled at 24 kHz
  - 39 hours training, 601 utterances held out
  - Tacotron-PN (postnet) + Griffin-Lim (similar to [1])

**Generation speed**
- Seconds to generate 5 seconds of speech
  - 90 input tokens, batch size 1
  - Wave-Tacotron ~10x faster than real-time on TPU (2x on CPU)
  - slower as frame size K decreases (more autoregressive steps)
  - ~10x faster than Tacotron+WaveRNN on TPU (2x on CPU)
  - ~2.5x slower than fully parallel vocoder on CPU

**Ablations**
  - 2 layer decoder LSTM
    - 256 channels in coupling layers
  - Optimal sampling temperature \( T = 0.7 \)
  - Deep multiscale flow is critical
  - Varying block size K
    - quality starts degrading for K = 40 ms

**Results**
- Tacotron + WaveRNN best
  - char/phoneme roughly on par
  - Tacotron+WaveRNN by ~0.2 points
  - phoneme/char
  - network uses capacity to model detailed waveform structure instead of pronunciation?
- Large gap to Tacotron+PN and Tacotron + Flowcoder

**Normalizing flow**
- Model joint distribution of K samples: \( P(y_T, y_{T-1}, \ldots, y_1) \)
  - similar to FloWaveNet [4], WaveGlow [5] neural vocoders
  - Invertible network
    - training: transform waveform block into noise
    - sampling: transform noise sample into waveform block using inverse
  - Change of variables
    - Maximize likelihood \( P(y_T | x_T) = P(z_T | y_T, x_T) \) (det(dz/dy))
  - Coupling layers [6] \( \rightarrow \) fast inverse, Jacobian determinant

**Training**
- Teacher forcing conditioning
  - At each step: transform waveform block \( y_t \) into noise \( z_t \)
  - Flow loss:
    - \( - \log P(y_t | z_t) + \log P(z_t | y_t) \)
    - \( \log \) spherical Gaussian
  - EOS stop token classifier loss: \( P(t \text{ is last frame}) \)

**Sampling**
- Based on flow network
  - take inverse of each layer, reverse order
  - At each step:
    - sample noise vector
    - generate waveform block with flow
  - autoregressive conditioning on previous output \( y^* \)
  - concatenate blocks \( y^* \) to form final signal

**Experiments**

**Sample variation**
- Generate 12 samples from the same input text
  - Baselines generate very consistent samples, across vocoders
    - same prosody every time
  - Wave-Tacotron has high variance
    - captures multimodal training distribution?
    - Tacotron regression loss collapses to single prosody mode?
  - similar pattern in Flowtron [8]
    - useful for ASR data augmentation?

**References**

---

Sound examples: https://google.github.io/tacotron/publications/wave-tacotron