Streaming Parrotron for on-device speech-to-speech conversion

Oleg Rybakov, Fadi Biadsy, Xia Zhang, Liyang Jiang, Phoenix Meadowlark, Shivani Agrawal

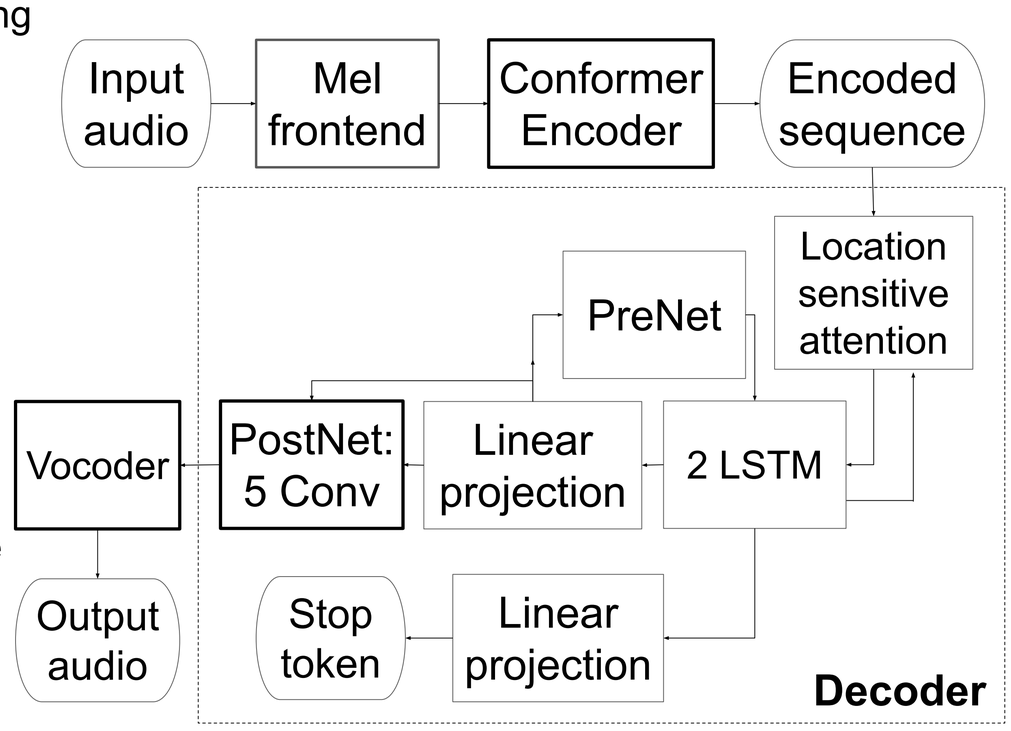
Introduction:

- Objective: run speech to speech model on device in real time with minimum delay and accuracy impact.
- We design semi-streaming approach: as soon as the speaker stops speaking, we run the spectrogram decoder in streaming mode along the side of a streaming vocoder to generate output speech.
- How to stream the model with minimal accuracy loss: we propose a hybrid approach for look-ahead in the encoder which combines a look-ahead feature stacker with a look-ahead self-attention.

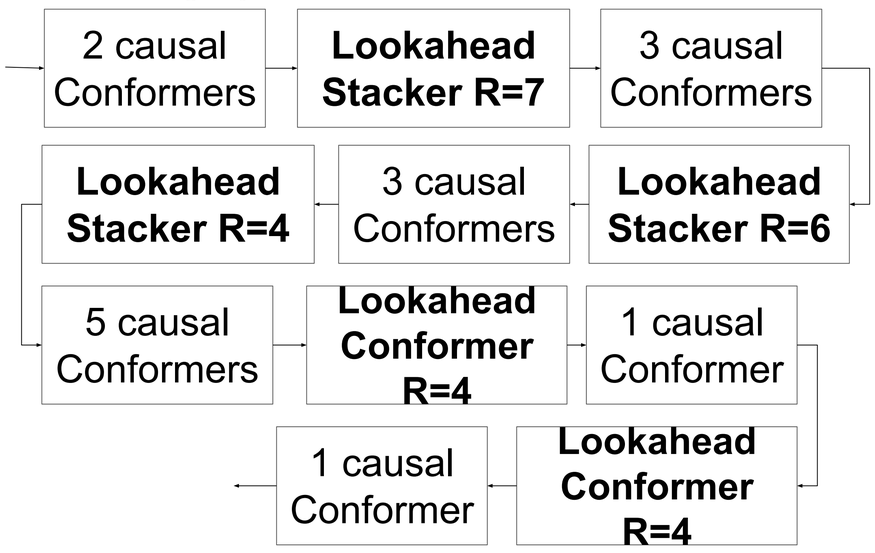
Streaming Parrotron:

While user is talking run Frontend and Encoder in streaming mode

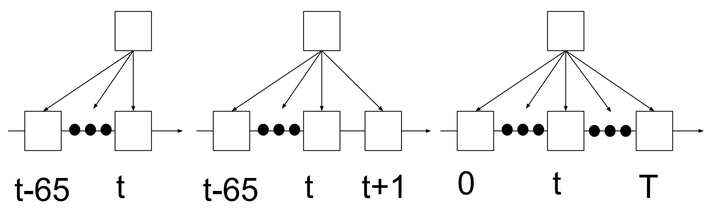
As soon as user stop talking run decoder with vocoder in streaming mode and play output audio at the same time.



Streaming hybrid lookahead Conformer encoder:



Self attention options:



Experimental results:

10s benchmark of non-streaming Parrotron on Pixel4

Model part	Latency [sec]
Encoder	2.8
Decoder	2.7
Vocoder sGL	2.4
Vocoder nGL	7

25ms benchmark of streaming decoder sDec with vocoders (GL, MG) on Pixel4

	sDec+GL	sDec+MG
Latency [ms]	16.0	13.4
RTF	1.6x	1.9x
Size [MB]	122	147
Memory [MB]	139	166

Base vs streaming models accuracy and total delay of processing 10 seconds audio

Model	WER [%]	delay [ms]
Base[7]*	14.7	7900
Streaming LSA_LS	15.3	400
Streaming LSA	16.4	440
Streaming Causal [7]	19.2	0

80ms benchmark of streaming encoder on Pixel4

	float32	int8 (int4*)
Latency [ms]	40	32
RTF2x	1.6x	2.5x
Size [MB]	436	111 (70*)
Memory [MB]	905	186

Conclusion:

- Our best configuration runs with a Real-Time-Factor of 2x on Pixel4, with a delay of 320ms, to start emitting audio as soon as the speaker stops speaking, whereas the non-streaming model requires a 7-second delay.
- With hybrid streaming Conformer we obtain a significant error reduction in comparison to a fully casual model, while only 0.7% error degradation when compared to a full context nonstreaming Base model.

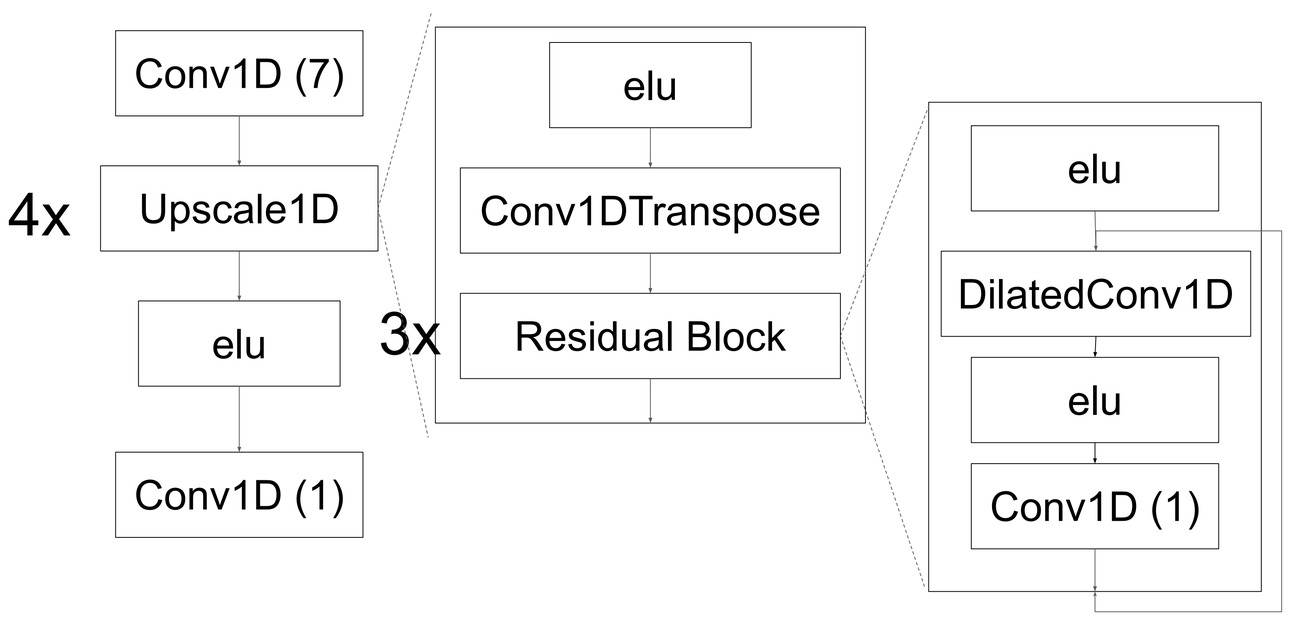
Real time spectrogram inversion on mobile phone

Oleg Rybakov, Marco Tagliasacchi, Yunpeng Li, Liyang Jiang, Xia Zhang, Fadi Biadsy

Introduction:

- Objective: run vocoder in real time on mobile phone with minimum delay, latency, memory footprint and accuracy impact.
- We design streaming aware MelGAN and our redesign of RTISI-LA in Tensorflow for end to end benchmarking on Pixel4.
- We present objective and subjective evaluation of these approaches on production Parrotron application.

Streaming MelGAN and GL:



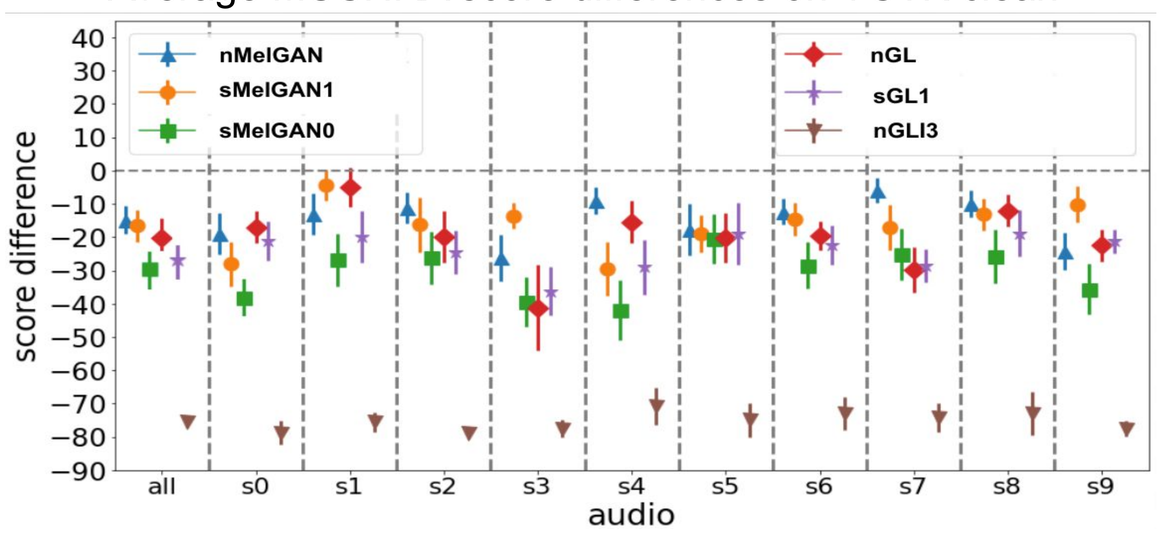

```
# Initialize magnitude and STFT sliding window
mag_w = tf.zeros((1,w_size,1025),tf.float32)
stft_w = tf.zeros((1,w_size,1025),tf.complex64)
def streaming_griffin_lim(mag_f,mag_w, stft_w):
  # Inverse log
  mag_f = tf.exp(mag_f) - delta
  # Magnitude frame to complex frame
  stft_f = tf.complex(mag_f, 0.0) * tf.exp(tf.complex(0.0,
                                          tf.zeros_like(mag_f)))
  # Magnitude sliding window
  mag_w = tf.concat([mag_w, mag_f], 1)
  mag_w = mag_w[:, -w_size:, :]
  # STFT sliding window
  stft_w = tf.concat([stft_w, stft_f], 1)
  stft_w = stft_w[:, -w_size:, :]
  commit_phase = tf.math.angle(stft_w[:, 0:ind, :])
  for _ in range(n_iters):  # GL iterations
    audio_w = tf.signal.inverse_stft(stft_w, frame_size, frame_step,
                                     fft_size, window_fn=None)
    stft_w = tf.signal.stft(audio_w, frame_size, frame_step,
                            fft_size, window_fn=hann_window)
    uncommit_phase = tf.math.angle(stft_w[:, ind:w_size, :])
    phase_w = tf.concat([commit_phase, uncommit_phase], 1)
    stft_w = tf.complex(mag_w, 0.0) * tf.exp(tf.complex(0.0,
                                          phase_w))
  stft_o = stft_w[:,ind:ind+1,:] # Output frame
  return stft_o, mag_w, stft_w
```

Experimental results:

Comparing WER from different vocoders after running Parrotron on atypical speech

Models	Deaf	ALS	MD
nGL	21.2	22.5	10.4
sGL1	22.2	22.5	10.4
sMelGAN1	20.6	23.0	10.9

Average MUSHRA score differences on VCTK clean



Delay with streaming latency of processing 12.5ms of audio and memory footprint

Models	Delay (look ahead) [ms]	Stream latency [ms]	File size [MB]	Memory [MB]
nGL	2000	N/A	0.1	
nMelGAN	187	N/A	25	
sGL1	12	5.2	0.1	7.6
sMelGAN1	12	6.7	25	34
sMelGAN0	0	6.7	25	34

Conclusion:

- We show that streaming GL and streaming MelGAN have RTF > 2x on Pixel4.
- With only one hop delay/lookahead (12.5ms) streaming MelGAN significantly outperforms GL approaches and a strictly causal MelGAN (with MUSHRA subjective evaluation).
- Streaming GL uses 4.5x smaller memory in comparison to streaming MelGAN.
- Both streaming GL and MelGAN showed comparable WER results on atypical speech.

2-bit Conformer quantization for automatic speech recognition

Oleg Rybakov, Phoenix Meadowlark, Shaojin Ding, David Qiu, Jian Li, David Rim, Yanzhang He

Introduction:

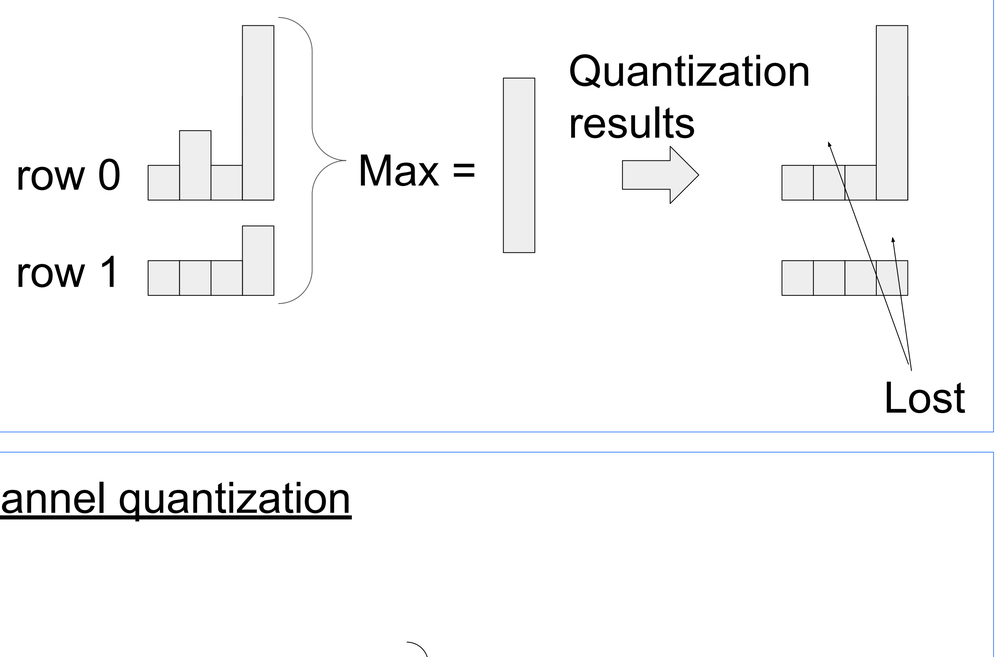
- Large models can give higher accuracy but it is hard to fit them in memory.
- Can we reduce model size (using 2bit quantization) with minimal accuracy loss?

Quantization:

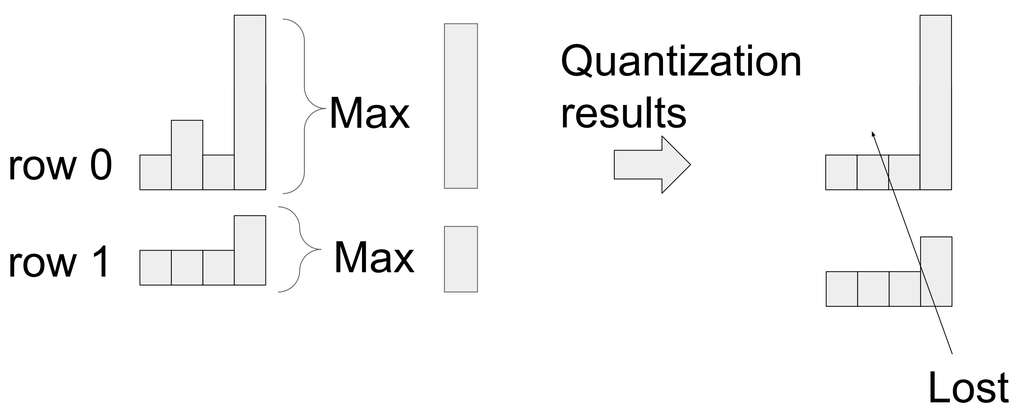
- Weight clipping - greedy search for clipping values to optimize MSE
- Propagate gradient through scale
- Asymmetric quantization
- Split channels into sub-channels

$$\text{round}\left(\frac{\text{greedy\_clipping}\left(\text{sub\_channel\_split}(\mathbf{W}) - \text{offset}\right)}{\text{scale}}\right)$$

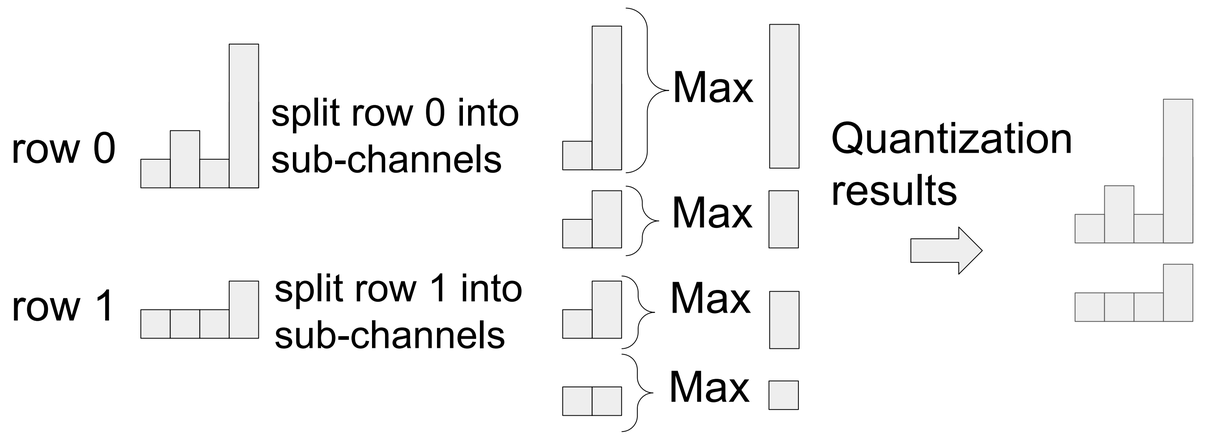
Per-tensor quantization
has one global max value



Per-channel quantization
has per-channel max value



Sub-channel quantization



Experimental results:

Librispeech

Model	test-clean WER [%]	test-other WER [%]	Model size [MB]
Float ConformerL	2.0	4.4	474.5
2bit Sym	3.6	8.1	53.8
2bit Asym	2.2	5.0	54.0
2bit Asym+Scale	2.2	4.6	54.0
2bit Asym+Scale+Sub+Clip	2.0	4.5	55.3

Prod large scale data

Model	WER [%]	Model size [MB]
Float model	6.0	474.5
4bit	6.3	65
2bit	7.4	37.5

Conclusion:

- Reduced model size down to 55MB (SOTA) with minimal or no accuracy loss on Librispeech.
- Quantization quality depends on both model and data size:
  - The larger the model (> 100M parameters), the easier it is to quantize its weights.
  - When training the model with large-scale datasets, we illustrated the inevitable WER regression.