NN Based Speech to Speech Translation

Temporally, all the papers claimed to achieve speech-to-speech translation are based on “Voice recognition + Translate + TTS”

We found that maybe it can be done in two steps: “Voice to foreign Text + TTS” based on NN.

Speech to Text Translation

- Sequence-to-Sequence Models Can Directly Transcribe Foreign Speech (From Google)
  - Seq2seq(or RNN Encoder–Decoder) model (Cho et al., 2014) based. Basic seq2seq consists of two recurrent neural networks (RNNs): an encoder that processes the input and a decoder that generates the output.
  - In this article, 3 NN are used: **Encoder** -> **Decoder**, **Attention Network**.
  - **Encoder**: Recurrent NN. Turn input feature frames $x_{1..T}$ to a sequence of hidden activation, $h_{1..L}$.
  - **Decoder**: A stacked RNN with D layers, input: h and y_{k-1} , output: y_k. Dependence on the input c_k , which is mediated through an attention network, a fixed dimensional context vector.

$$o_k^1,\ s_k^1 = d^1(y_{k-1}, s_{k-1}^1, c_{k-1}) \quad (3)$$

$$o_k^n,\ s_k^n = d^n(o_k^{n-1}, s_{k-1}^n, c_k) \quad (4)$$

where d^n is a long short-term memory (LSTM) cell [23].

  - **Attention Network**: summarizes the entire input sequence as c_k
    - c_k , computed from the first decoder layer output at each output step k. a_e and a_d are small fully connected networks.

$$c_k = \sum_l \alpha_{kl} h_l \quad (5)$$

$$\alpha_{kl} = \text{softmax}(a_e(h_l)^T a_d(o_k^1)) \quad (6)$$

Sequence-to-Sequence Models Can Directly Transcribe Foreign Speech (From Google)

- Speech Model
  - For both end-to-end speech translation, and speech recognition baseline model to compare.
  - Use 80 channel log mel filterbank features extracted from 25ms windows with a hop size of 10ms, stacked with delta and delta-delta features. The output softmax of all models predicts one of 90 symbols, that includes English and Spanish lowercase letters.
  - **Encoder(8** layers): Raw feature, delta, delta-deltas formed as a T × 80 × 3 tensor -> two conv layers with ReLU and Batch normalization -> bidirectional convolutional LSTM(1×3 filter) -> three bidirectional LSTM layers of size 256 in each direction, interleaved with a 512-dimensional linear projection, followed by batch normalization and a ReLU activation -> get 512-dimensional encoder representation, h_l.
  - **Attention Network:** a_e and a_d , each contain a single hidden layer with 128 units -> c_k
  - **Decoder**: Input is created by concatenating a 64- dimensional embedding for y_{k-1}, the symbol emitted at the previous time step, and c_k. Input -> a stack of four unidirectional LSTM layers with 256 units. The concatenation of the attention context and LSTM output is passed into a softmax layer which predicts the probability of emitting each symbol in the output vocabulary.

Sequence-to-Sequence Models Can Directly Transcribe Foreign Speech (From Google)

- Neural Machine Translation Model
  - Text to text, as baseline, to compare with end-to-end model.
  - Encoder: The encoder network consists of four encoder layers (5 LSTM layers in total). As in the base architecture, the bottom layer is a bidirectional LSTM and the remaining layers are all unidirectional.
  - Decoder: The decoder network consists of 4 stacked LSTM layers. All encoder and decoder LSTM layers contain 512 units.
  - Attention Network: The attention network uses a single hidden layer with 512 units. Use the same character-level vocabulary for input and output as the speech model.
- Multitask Training
  - Using supervision from source language transcripts by jointly training speech recognition and translation models in a multi-task configuration.
  - Using 16 asynchronous workers, weight noise is introduced after 30k overall steps, and the learning rate is decayed after 1.5m overall steps.

Sequence-to-Sequence Models Can Directly Transcribe Foreign Speech (From Google)

- Implementation: TensorFlow
  - Speech Model
    - Adam optimizer, $\beta 1 = 0.9$, $\beta 2 = 0.999$, and $\varepsilon = 10^{-6}$
    - Learning rate: start from 10^{-3}, decayed by a factor of 10 after 1m steps
    - L2 weight decay is used with a weight of 1e-6, and, beginning from step 20k.
    - Gaussian weight noise with std of 0.125 is added to weights for all LSTM layers and decoder embeddings.
    - Optimized and tuned for Fisher dataset.
  - NMT Model
    - To reduce overfitting on the small training corpus, used reduced sized model.
    - Applied dropout with probability 0.2 during training to reduce overfitting.
    - Train using SGD with a single replica. Training converges after about 100k steps using mini-batches of 128 sentence pairs.

Sequence-to-Sequence Models Can Directly Transcribe Foreign Speech (From Google)

- Experiments:
  - On Spanish-English, Spanish from Fisher and Callhome dataset. English from translation of Spanish dataset.
  - Tuning depth of decoder: automatic speech recognition (ASR) normally uses shallow decoder, NMT models often use much deeper decoders
  - Tuning multitask training:
    - One-to-many in which an encoder is shared between speech translation and recognition tasks (Perform Better)
    - Many-to-one in which a decoder is shared between speech and text translation tasks
  - Report speech recognition results as word error rates (WER) and translation results using BLEU (bilingual evaluation understudy)

Sequence-to-Sequence Models Can Directly Transcribe Foreign Speech (From Google)

- Experiments results:

Table 2: *Varying the number of shared encoder LSTM layers in the multi-task setting. BLEU score on the Fisher/dev set.*

Num shared encoder LSTM layers			
3 (all)	2	1	0
46.2	45.1	45.3	44.2

Table 3: *Speech recognition model performance in WER.*

		Fisher		Callhome	
	dev	dev2	test	devtest	evltest
Ours[3]	25.7	25.1	23.2	44.5	45.3
Post et al. [17]	41.3	40.0	36.5	64.7	65.3
Kumar et al. [19]	29.8	29.8	25.3	–	–

Table 4: *Translation performance on ground truth transcripts.*

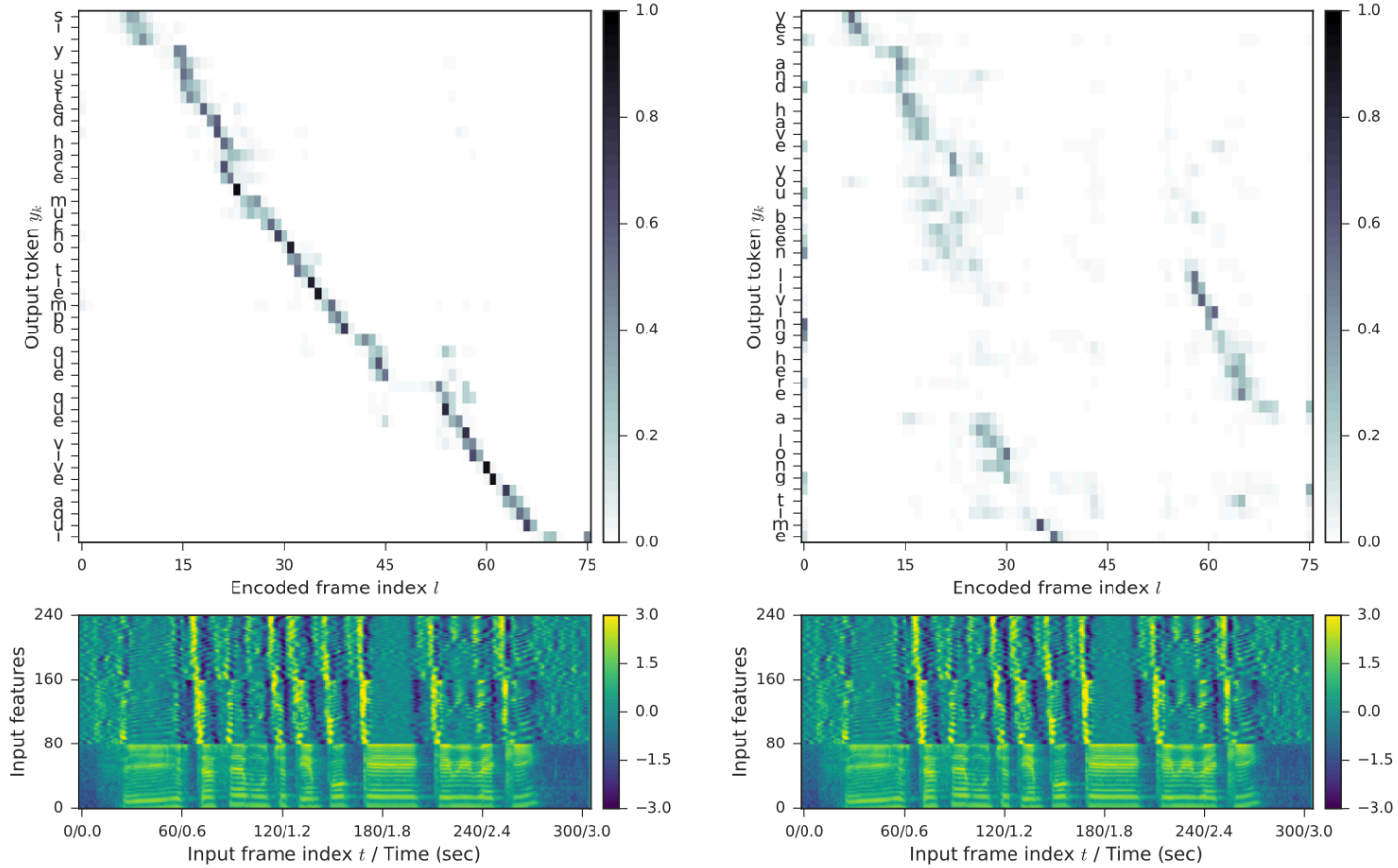
		Fisher		Callhome	
	dev	dev2	test	devtest	evltest
Ours	58.7	59.9	57.9	28.2	27.9
Post et al. [17]	–	–	58.7	–	27.8
Kumar et al. [19]	–	65.4	62.9	–	–

Table 5: *Speech translation model performance in BLEU score.*

		Fisher		Callhome	
Model	dev	dev2	test	devtest	evltest
End-to-end ST[3]	46.5	47.3	47.3	16.4	16.6
Multi-task ST / ASR[3]	48.3	49.1	48.7	16.8	17.4
ASR→NMT cascade[3]	45.1	46.1	45.5	16.2	16.6
Post et al. [17]	–	35.4	–	–	11.7
Kumar et al. [19]	–	40.1	40.4	–	–

Sequence-to-Sequence Models Can Directly Transcribe Foreign Speech (From Google)

- Experiments results:



(a) *Spanish speech recognition decoder attention.*

(b) *Spanish-to-English speech translation decoder attention.*

Figure 1: *Example attention probabilities α_{kl} from a multi-task model with two decoders. The ASR attention is roughly monotonic, whereas the translation attention contains an example of word reordering typical of seq2seq MT models, attending primarily to frames $l = 58 - 70$ while emitting "living here". The recognition decoder attends to these frames while emitting the corresponding Spanish phrase "vive aqui". The ASR decoder is also more confident than the translation attention, which tends to be smoothed out across many input frames for each output token. This is a consequence of the ambiguous mapping between Spanish sounds and English translation.*

Deep Voice: Real-time Neural TTS

Overview:

1) Use Deep Neural Network throughout the whole TTS process

2) Fewer parameters training for better data adaptation

3) Production-ready quality:

   i. Real-time responsive

   ii. Tunable trade-off between accuracy and speed

   iii. Completely standalone, can be trained from scratch using a dataset of short audio clips and corresponding textual transcripts.

   iv. Go here for experiencing the synthesized voice that sounds almost as natural as human beings

Models

1) **Grapheme-to-phoneme conversion model**

   Phonemes encoded using a phonemic alphabet such as ARPABET

2) **Segmentation model**

   Identifies where in the audio each phoneme begins and ends.

3) **Phoneme duration prediction model**

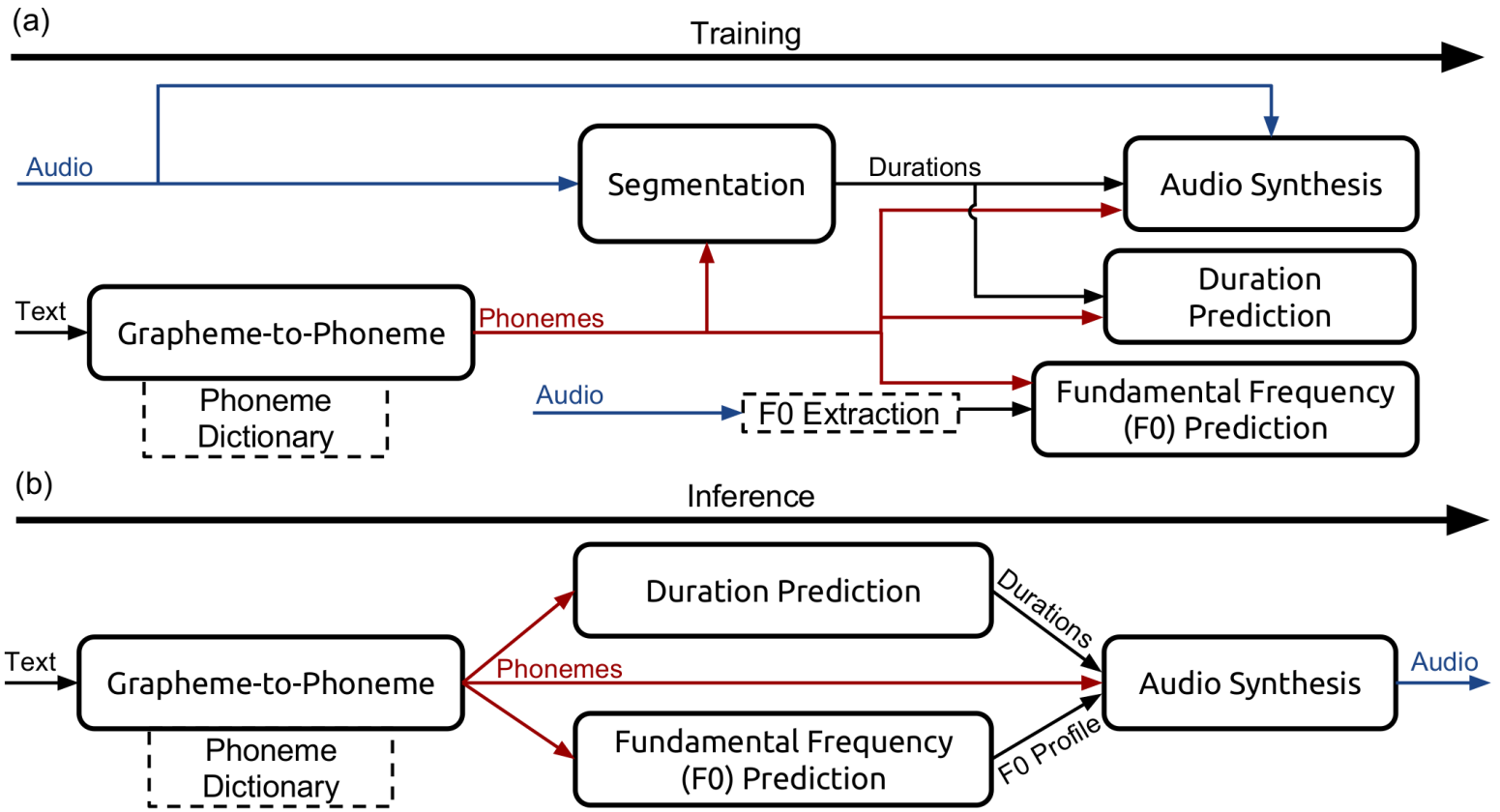
4) **Fundamental frequency prediction model**

   Predicts whether a phoneme is voiced and its fundamental frequency (F0) if it is

5) **Audio synthesis model**

   Combines the outputs of the aforementioned models at a high sampling rate

Models in a nutshell



Models are

- implemented using the TensorFlow framework
- trained on two datasets spoken by a professional female speaker
    1. an internal English speech containing approximately 20 hours of speech data segmented into 13,079 utterances
    2. a subset of the Blizzard 2013 data to demonstrate flexibility

Grapheme-to-Phoneme Model

- A multi-layer bidirectional encoder with a gated recurrent unit (GRU) nonlinearity and an equally deep unidirectional GRU decoder
  - The initial state of every decoder layer is initialized to the final hidden state of the corresponding encoder forward layer.
  - The architecture is trained with teacher forcing and decoding is performed using beam search.
  - 3 bidirectional layers with 1024 units each in the encoder
  - 3 unidirectional layers of the same size in the decoder
  - A beam search with a width of 5 candidates
  - Dropout with probability 0.95 after each recurrent layer
- For training, use the *Adam optimization algorithm* with

β_1	β_2	ε	batch size	learning rate	annealing rate applied every 1000 iterations
0.9	0.999	10^{-8}	64	10^{-3}	0.85

Segmentation Model

- Annotate the training voice data with phoneme boundaries
- Trained to output the alignment between a given utterance and a sequence of target phonemes
- **Connectionist temporal classification (CTC) loss function** for a mapping between sound and text
- Adapt the convolutional recurrent neural network architecture from *Deep speech 2* (End-to-end speech recognition in English and Mandarin) for phoneme boundary detection
- A highlight point: Train to predict sequences of phoneme **pairs rather than single** phonemes for precise boundary detection and brief peaks elimination
  - The network will then tend to output phoneme pairs at timesteps close to the boundary between two phonemes in a pair
- *Adam optimization algorithm*:

β_1	β_2	ε	batch size	learning rate	annealing rate applied every 500 iterations
0.9	0.999	10^{-8}	128	10^{-4}	0.95

Duration and Frequency Model

- Jointly prediction
  - Input a sequence of phonemes with stresses, with each phoneme and stress being encoded as a one-hot vector
  - First two fully connected layers with 256 units each (dropout probability 0.8)
  - Then two unidirectional recurrent layers with 128 GRU cells each
  - Finally one fully-connected output layer (dropout probability 0.8) produces three estimations for every input phoneme
    1. The phoneme duration
    2. The probability that the phoneme is voiced (i.e. has a fundamental frequency)
    3. 20 time-dependent F0 values, sampled uniformly over the predicted duration
- The model is optimized by minimizing a joint loss that combines
  - Phoneme duration error
  - Fundamental frequency error
  - The negative log likelihood of the probability that the phoneme is voiced, and
  - A penalty term proportional to the absolute change of F0 with respect to time to impose smoothness

β_1	β_2	ε	batch size	learning rate	annealing rate applied every 400 iterations
0.9	0.999	10^{-8}	128	3×10^{-4}	0.9886

Audio Synthesis Model

- WaveNet has (refer to Appendix A for network details)
  - a conditioning network, which upsamples linguistic features to the desired frequency, followed by ℓ 2×1 convolution layers with r residual output channels and gated tanh nonlinearities
  - an autoregressive network that generates a probability distribution P(y) over discretized audio samples $y \in \{0, 1, ..., 255\}$
  - used transposed convolutions for upsampling and conditioning
- Deep Voice is essentially a variant of WaveNet and our variations are
  - The number of layers ℓ
  - The number of residual channels r (dimension of the hidden state of every layer)
  - The number of skip channels s (the dimension to which layer outputs are projected prior to the output layer)
  - Break the convolution into two matrix multiplies per timestep with W_{prev} and W_{cur}, connected with residual connections
  - Concatenate the hidden state of every layer to an ℓr vector and projected to s skip channels with W_{skip}, followed by two layers of 1×1 convolutions (with weights W_{relu} and W_{out}) with relu nonlinearities
  - First encode the inputs with a stack of bidirectional quasi-RNN (QRNN) layers and then perform upsampling by repetition to the desired frequency

ℓ	r	s	β_1	β_2	ε	batch size	learning rate	annealing rate applied every 1000 iterations
40	64	256	0.9	0.999	10^{-8}	8	10^{-3}	0.9886

Performance and Results

Model	Segmentation	Grapheme-to-Phoneme	Phoneme Duration and Fundamental Frequency	Audio Synthesis
TitanX Maxwell	8	1	1	8
No. Iterations	~14,000	~20,000	~20,000	~300,000
Time per iteration	1300 ms	150 ms	120 ms	450 ms
Error Rate	7%	5.8% (phoneme) 28.7% (word)	mean absolute error of 38 ms (duration) 29.4 Hz (frequency)	
Miscellaneous	• Randomly shifting phoneme boundaries by 10-30 ms makes no difference in the audio quality • Suspect that audio quality is insensitive to the phoneme pair error rate past a certain point	• Data obtained from CMUDict • Strip out all words that do not start with a letter, contain numbers, or have multiple pronunciations, leaving **124,978 out of the original 133,854** grapheme-phoneme sequence pairs • Unlike prior work, no language model was used during decoding		More details on next page

Audio synthesis performance

- 1st dataset:
  - Divide the utterances into one second chunks with a quarter second of context for each chunk, padding each utterance with a quarter second of silence at the beginning.
  - Filter out chunks that are predominantly silence and end up with 74,348 total chunks.
  - Trained models with varying depth, including 10, 20, 30, and 40 layers in the residual layer stack.
    - Models below 20 layers result in poor quality audio
    - The 20, 30, and 40 layer models all produce high quality recognizable speech
    - The 40 layer models have less noise than the 20 layer models, which can be detected with high-quality over-ear headphones
  - A single 1.25s chunk is sufficient to saturate the compute on the GPU and that batching does not increase training efficiency
- 2nd dataset:
  - Retrain all of the models with identical hyper-parameters on the Blizzard 2013 dataset. Select a 20.5 hour subset segmented into 9,741 utterances
  - Evaluated the model using the mean opinion score (MOS) ratings (ratings between one and five, higher values being better) from Mechanical Turk using the CrowdMOS toolkit and methodology
  - On the held out set, 16 kHz companded and expanded audio receives a MOS score of 4.65±0.13, while our synthesized audio received a MOS score of 2.67±0.37.

Optimizing Inference

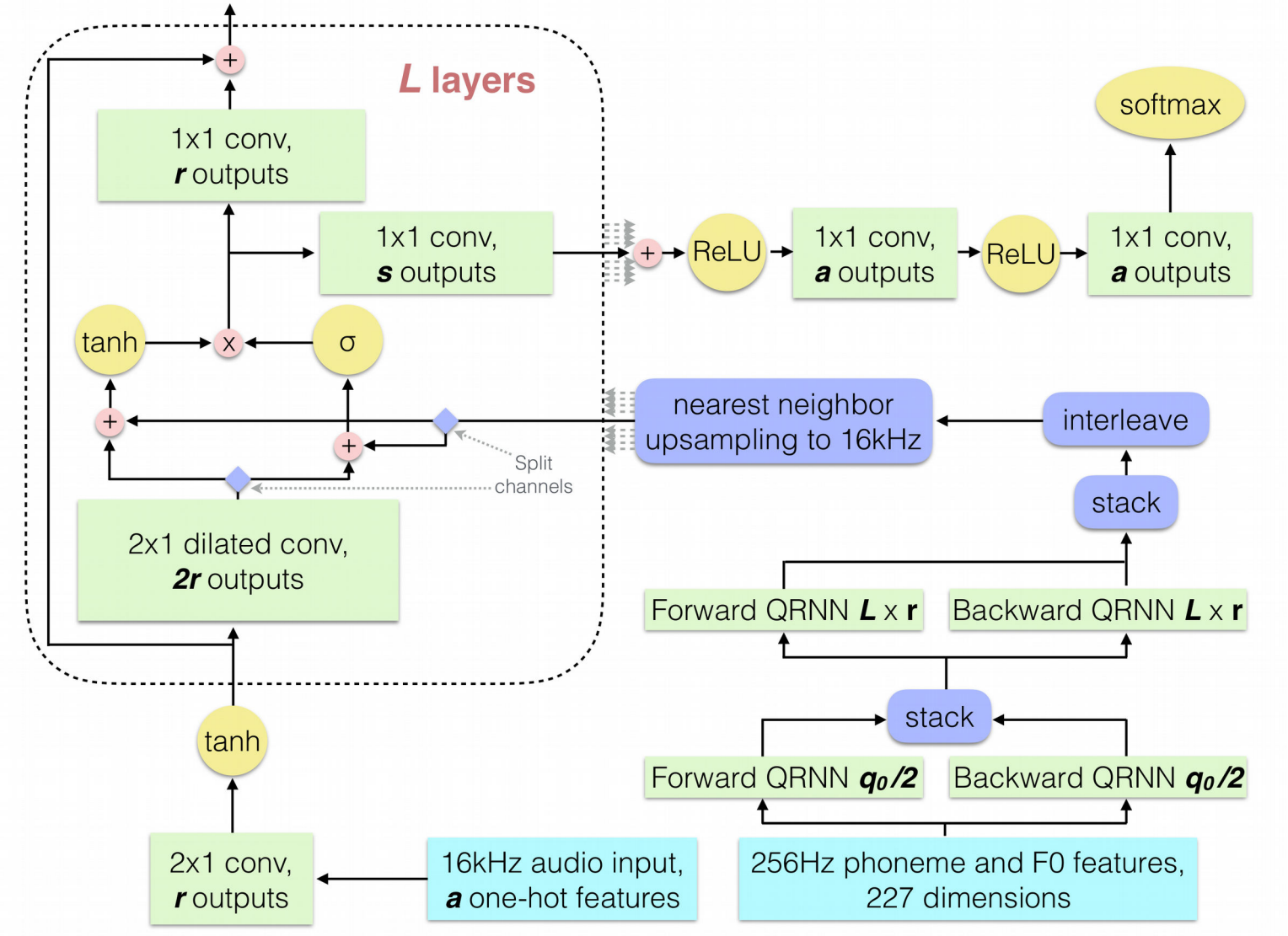
- CPU Implementation
  - We achieve real-time CPU inference by avoiding any recomputation, doing cache-friendly memory accesses, parallelizing work via multithreading with efficient synchronization, minimizing nonlinearity FLOPs, avoiding cache thrashing and thread contention via thread pinning, and using custom hardware-optimized routines for matrix multiplication and convolution.
  - For the CPU implementation, we split the computation into the following steps:
    - Sample Embedding – Layer Inference – Output
- GPU Implementation
  - Challenge: Usually, code is run on the GPU in a sequence of kernel invocations, with every matrix multiply or vector operation being its own kernel. But the latency for a CUDA kernel launch (which may be up to 50 μs) combined with the time needed to load the entire model from GPU memory are prohibitively large for an approach like this. An inference kernel in this style ends up being approximately 1000X slower than real-time.
  - Build a kernel using the techniques of persistent RNNs which generates all samples in the output audio in a single kernel launch. The weights for the model are loaded to registers once and then used without unloading them for the entire duration of inference. Due to the mismatch between the CUDA programming model and such persistent kernels, the resulting kernels are specialized to particular model sizes and are incredibly labor-intensive to write.
  - We believe that with these techniques and a better implementation we can achieve real-time WaveNet inference on GPUs as well as CPUs.

Conclusion

- As prevailing as it is right now, speech-to-speech translation (S2ST) is still largely done with a 3-step process: automate speech recognition (ASR), machine translation (MT), and text-to-speech synthesis (TTS).
- Seq2seq model can directly translate speech into text in a different language, its architecture is essentially the same as that of an attention-based ASR neural system. Besides, the ASR and end-to-end Speech Translation models have the same number of parameters, and utilize the same decoding algorithm – narrow beam search. The end-to-end trained model outperforms an ASR-MT cascade even though it never explicitly searches over transcriptions in the source language during decoding.
- We believe that combining these two models together can yield a better voice-to-voice translation service that not only improves the overall efficiency, but also provides better synthesized voice that sounds more human than machine for better user experience.

Appendix A

The modified WaveNet architecture



Components are colored according to function: teal inputs, green convolutions and QRNNs, yellow unary operations and softmax, pink binary operations, and indigo reshapes, transposes, and slices.