

# End-to-end systems 2: Encoder-Decoder models

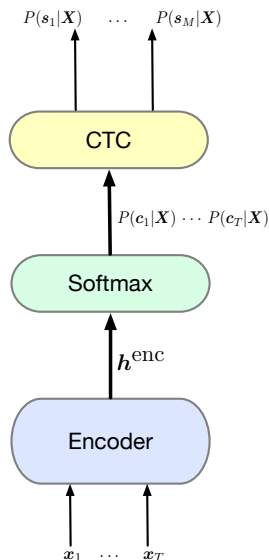
Peter Bell

Automatic Speech Recognition – ASR Lecture 16  
12 March 2020

- Adds a blank ( $\epsilon$ ) symbol to the output labels
- A deep LSTM (for example) maps input sequence  $\mathbf{X}$  (length  $T$ ) to a label sequence  $\mathbf{C}$  (length  $T$ )
- Use CTC compression rule (merge adjacent repeated symbols, then remove blanks) to produce subword sequence  $\mathbf{S}$  (length  $M \leq T$ )
- CTC loss function computes the probability  $P(\mathbf{S}|\mathbf{X})$  by summing over all possible valid alignments  $P(\mathbf{C}|\mathbf{X})$

View CTC as having three components:

- **Encoder:** Deep (bidirectional) LSTM recurrent network which maps acoustic features  $\mathbf{X} = \mathbf{x}_1, \dots, \mathbf{x}_T$  to a sequence of hidden vectors  $\mathbf{h}^{\text{enc}} = \mathbf{h}_1^{\text{enc}}, \dots, \mathbf{h}_T^{\text{enc}}$ .
- **Softmax:** Computes the label probabilities  $P(\mathbf{c}_1|\mathbf{X}), \dots, P(\mathbf{c}_T|\mathbf{X})$
- **CTC:** Computes the subword sequence  $P(\mathbf{s}_1|\mathbf{X}), \dots, P(\mathbf{s}_M|\mathbf{X})$



# Limitations of CTC

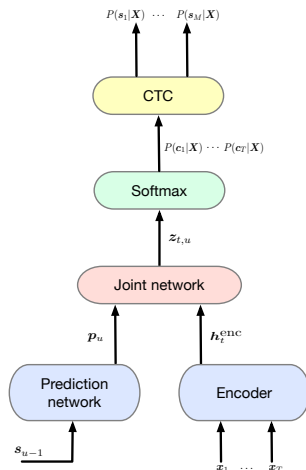
- CTC – pros
  - Can train end-to-end without requiring frame-wise alignments
  - Sums over all possible alignments (using forward-backward)
  - Preserves monotonic relationship between acoustic frames and output labels

# Limitations of CTC

- CTC – pros
  - Can train end-to-end without requiring framewise alignments
  - Sums over all possible alignments (using forward-backward)
  - Preserves monotonic relationship between acoustic frames and output labels
- CTC – cons
  - Assumes output predictions at different times are independent
  - Requires additional language and pronunciation models to introduce dependencies between output labels
  - Incorporation of language models is typically ad-hoc
  - End-to-end training of CTC models (also of LF-MMI models) updates the acoustic model parameters using a sequence level criterion, but does not update the pronunciations or language models

# RNN Transducer Model

- **Encoder:** Acoustic model network mapping acoustic features  $\mathbf{X} = \mathbf{x}_1, \dots, \mathbf{x}_T$  to hidden vectors  $\mathbf{h}^{\text{enc}} = \mathbf{h}_1^{\text{enc}}, \dots, \mathbf{h}_T^{\text{enc}}$ .
- **Prediction network:** Recurrent network which takes the previous output subword label  $s_{u-1}$  as input and predicts the next subword label  $p_u$  – acts as a language model (over subwords)
- **Joint network:** Computes a joint hidden vector  $\mathbf{Z} = \mathbf{z}_1, \dots, \mathbf{z}_T$  by applying a shallow feed-forward net to  $\mathbf{h}^{\text{enc}}$  and  $p_u$
- Followed by **softmax** and **CTC** components as before

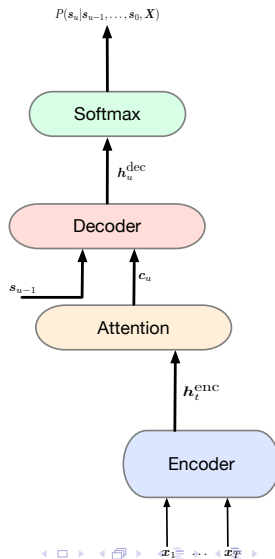


# RNN Transducer Model

- RNN transducer can operate left-to-right in a frame-synchronous manner (if the encoder is a unidirectional LSTM)
- Acoustic model (encoder) and language model (prediction network) parts are modelled independently and combined in the joint network. However everything is optimised to a common sequence-level objective (using the CTC loss function).
- With sufficient training data, additional language and pronunciation models are not necessary ( [Google experiments](#) )
- The recently announced Google “all-neural” on-device speech recognition uses unidirectional RNN transducers  
<https://ai.googleblog.com/2019/03/an-all-neural-on-device-speech.html>

# Attention-based Encoder-Decoder Model

- **Encoder:** Acoustic model using a recurrent network to map acoustic features  $\mathbf{X} = \mathbf{x}_1, \dots, \mathbf{x}_T$  to hidden vectors  $\mathbf{h}^{\text{enc}} = \mathbf{h}_1^{\text{enc}}, \dots, \mathbf{h}_T^{\text{enc}}$ .
- **Decoder:** Computes distribution over labels conditioned on previously predicted labels and the acoustics,  $P(s_u | s_{u-1}, \dots, s_0, \mathbf{x})$
- **Attention:** Constructs a *context vector* for the decoder network based on attention weights computed over all frames in the encoder output
- Google's "Listen, Attend, and Spell" model: Chan et al (2016), ICASSP.





- The decoder directly generates the output subword sequence  $\mathbf{S}$
- At each decoding time step  $u$ , the decoder RNN uses the previous output  $\mathbf{s}_{u-1}$ , the previous decoder RNN hidden state  $\mathbf{h}_{u-1}^{\text{dec}}$ , and the previous context vector  $\mathbf{c}_{u-1}$  to generate the current decoder hidden state  $\mathbf{h}_u^{\text{dec}}$

$$\mathbf{h}_u^{\text{dec}} = \text{RNN}(\mathbf{h}_{u-1}^{\text{dec}}, \mathbf{s}_{u-1}, \mathbf{c}_{u-1})$$

- The context vector is computed by the attention mechanism

# The Attention Mechanism

- The attention mechanism uses the current decoder RNN hidden state  $\mathbf{h}_{u-1}^{\text{dec}}$ , and the sequence of encoder hidden states  $\mathbf{h}^{\text{enc}}$  to compute an alignment matrix  $\alpha_{ut}$ :

$$\alpha_{ut} = \text{Attention}(\mathbf{h}_u^{\text{dec}}, \mathbf{h}_t^{\text{enc}})$$

- The alignment vector is used as weights in a weighted sum of the encoder hidden states to compute the context vector  $\mathbf{c}_u$ :

$$\mathbf{c}_u = \sum_{t=1}^T \alpha_{ut} \mathbf{h}_t^{\text{enc}}$$

- The decoder uses the context vector  $\mathbf{c}_u$  and the current decoder hidden state  $\mathbf{h}_u^{\text{dec}}$  to estimate the subword distribution:

$$\mathbf{s}_u \sim \text{LabelDistribution}(\mathbf{c}_u, \mathbf{h}_u^{\text{dec}})$$

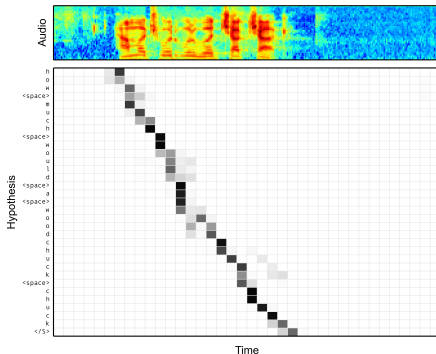
where LabelDistribution is a single layer neural network with a softmax output over the labels.

- Attention models the alignment between the current output  $\mathbf{s}_u$  and the input sequence  $\mathbf{x}$  – it matches the “input clock” with the “output clock”
- Various ways to compute the attention - content-based attention commonly used. Single hidden layer followed by a softmax

$$e_{ut} = \mathbf{v}^T \tanh(\mathbf{W}\mathbf{h}_u^{\text{dec}} + \mathbf{V}\mathbf{h}_t^{\text{enc}} + \mathbf{b})$$
$$\alpha_{ut} = \frac{\exp(e_{ut})}{\sum_k \exp(e_{uk})}$$

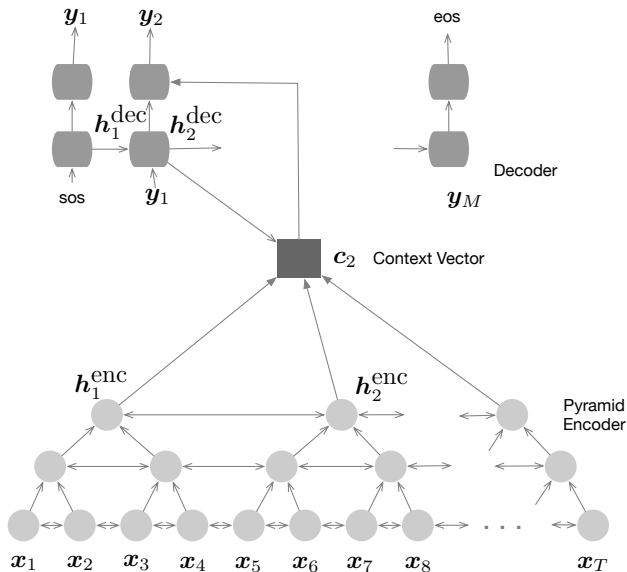
# Alignment between labels and acoustics

Alignment between the Characters and Audio

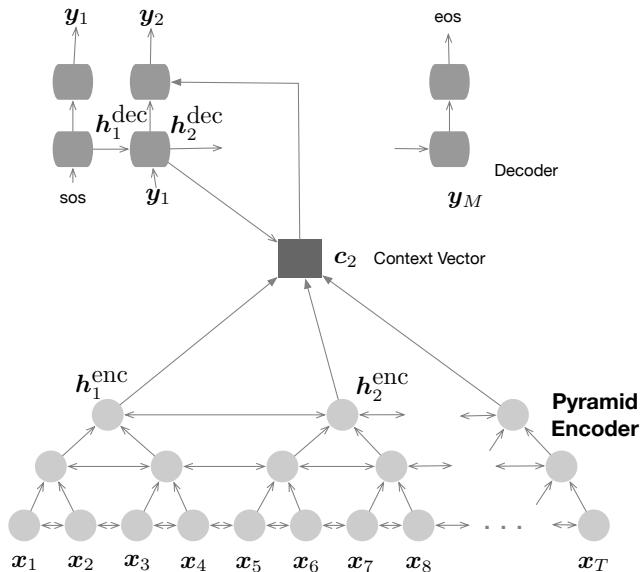


“How much would a woodchuck chuck”

# Attention Mechanism



# Attention Mechanism



# Pyramid Encoder

- A significant problem with a naive end-to-end model is the length of the input sequences... A direct BLSTM encoder can be difficult and slow to train – hard to extract the relevant information from many time steps
- Use a pyramid architecture – each successive layer reduces the resolution by a factor of 2.
  - Typical deep BLSTM hidden state (layer  $j$ , time  $t$ ):

$$\mathbf{h}_t^j = \text{RNN}(h_t^{j-1}, h_{t-1}^j)$$

- Pyramid model concatenates consecutive hidden states:

$$\mathbf{h}_t^j = \text{pyrRNN}([h_{2t-1}^{j-1}, h_{2t}^{j-1}], h_{t-1}^j)$$

- 3 layers in a pyramid architecture reduces the time resolution (shortens the sequence) by a factor of 8
- The attention mechanism thus has an easier job, weighting over 8x fewer encoder hidden states

- Model trained to maximise the log probability of correct sequences

$$\sum_u \log P(\mathbf{s}_u | \mathbf{x}, \mathbf{s}_{<u})$$

where  $\mathbf{s}_{<u}$  is the sequence  $s_1, \dots, s_{u-1}$

- An interesting subtlety: what value should be used for  $\mathbf{s}_{<u}$ ?
  - The previous predictions? this is consistent between training and test, but adds noise at training time



- Model trained to maximise the log probability of correct sequences

$$\sum_u \log P(\mathbf{s}_u | \mathbf{x}, \mathbf{s}_{<u})$$

where  $\mathbf{s}_{<u}$  is the sequence  $s_1, \dots, s_{u-1}$

- An interesting subtlety: what value should be used for  $\mathbf{s}_{<u}$ ?
  - The previous predictions? this is consistent between training and test, but adds noise at training time
  - The ground truth labels (*teacher forcing*)? This speeds up learning, especially early on, but there is a mismatch between training and testing

- Model trained to maximise the log probability of correct sequences

$$\sum_u \log P(\mathbf{s}_u | \mathbf{x}, \mathbf{s}_{<u})$$

where  $\mathbf{s}_{<u}$  is the sequence  $s_1, \dots, s_{u-1}$

- An interesting subtlety: what value should be used for  $\mathbf{s}_{<u}$ ?
  - The previous predictions? this is consistent between training and test, but adds noise at training time
  - The ground truth labels (*teacher forcing*)? This speeds up learning, especially early on, but there is a mismatch between training and testing
  - **Scheduled sampling**? Sample a label from the estimated distribution. This reduces the noise in training, but doesn't create a big gap between training and test

# Decoding and Rescoring

- Decode without a separate pronunciation model or an external language model!
- Simply decode the grapheme sequence! (It is possible to rescore with a language model if desired)
- Decoding use a beam search in which 15-best hypotheses are retained at each decoding step

# Results (2017)

Google Voice Search data, 12,500h training data, 15M hand-transcribed utterances

Model	Clean		Noisy		numeric
	dict	vs	dict	vs	
Baseline Uni. CDP	6.4	9.9	8.7	14.6	11.4
Baseline BiDi. CDP	5.4	8.6	6.9	-	11.4
End-to-end systems					
CTC-grapheme <sup>3</sup>	39.4	53.4	-	-	-
RNN Transducer	6.6	12.8	8.5	22.0	9.9
RNN Trans. with att.	6.5	12.5	8.4	21.5	9.7
Att. 1-layer dec.	6.6	11.7	8.7	20.6	9.0
Att. 2-layer dec.	<b>6.3</b>	<b>11.2</b>	<b>8.1</b>	<b>19.7</b>	<b>8.7</b>

Prabhavalkar et al (2017), "A Comparison of Sequence-to-Sequence Models for Speech Recognition", Interspeech. [https://www.isca-speech.org/archive/Interspeech\\_2017/abstracts/0233.html](https://www.isca-speech.org/archive/Interspeech_2017/abstracts/0233.html)

## Other Refinements

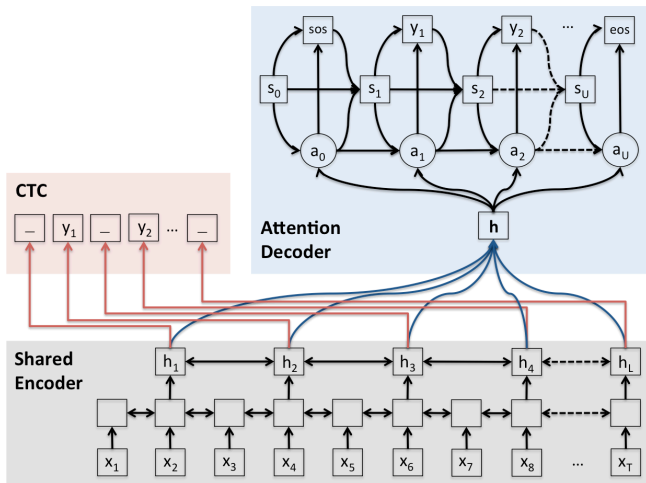
- Wordpiece models – rather than using single graphemes as labels use multi-grapheme units (up to a word in length) - similar to byte pair encoding in machine translation
- Multiheaded attention – use multiple attention distributions
- Minimum WER training – modify the loss function to interpolate a word error rate term
- Label smoothing – smooth the ground truth distribution by interpolating with a uniform distribution
- LM rescoring – use an external language model (5-gram) trained on large amount of text

Reduced WER on Voice Search from 9.2% to 5.6% – their hybrid HMM-LSTM system has WER of 6.7% on this task

Chiu et al, “State-of-the-art sequence recognition with sequence-to-sequence models”, ICASSP 2018. <https://arxiv.org/abs/1712.01769>

- Attention is very flexible – does not constrain relationship between acoustics and labels to be monotonic
- This can be a problem, especially when huge amounts of training data not available
- Possible solutions:
  - Windowed attention, in which the attention is restricted a set of encoder hidden states
  - Hybrid CTC/Attention model - use CTC and attention jointly during training and recognition – regularises the system to favour monotonic alignments

# Hybrid CTC/Attention



Watanabe et al (2017), "Hybrid CTC/Attention Architecture for End-to-End Speech Recognition", IEEE STSP, 11:1240–1252.

<https://ieeexplore.ieee.org/document/8068205>

# Summary

- End-to-end models for speech recognition: CTC, RNN Transducer, Attention Encoder-Decoder
- RNN Transducer and Attention-based model integrate acoustic model, pronunciation model, and language model into a single neural network
- With large amounts of hand-transcribed training data, attention-based model can be more accurate than context-dependent NN/HMM
- RNN transducer can operate in online (left-to-right) mode
- Attention based model operates over an utterance at a time (since attention is over the complete encoded utterance)
- Very active research area! Eg. recent use of self-attention (Transformer) in place of recurrent architectures