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

Peter Bell

Automatic Speech Recognition – ASR Lecture 17
18 March 2024

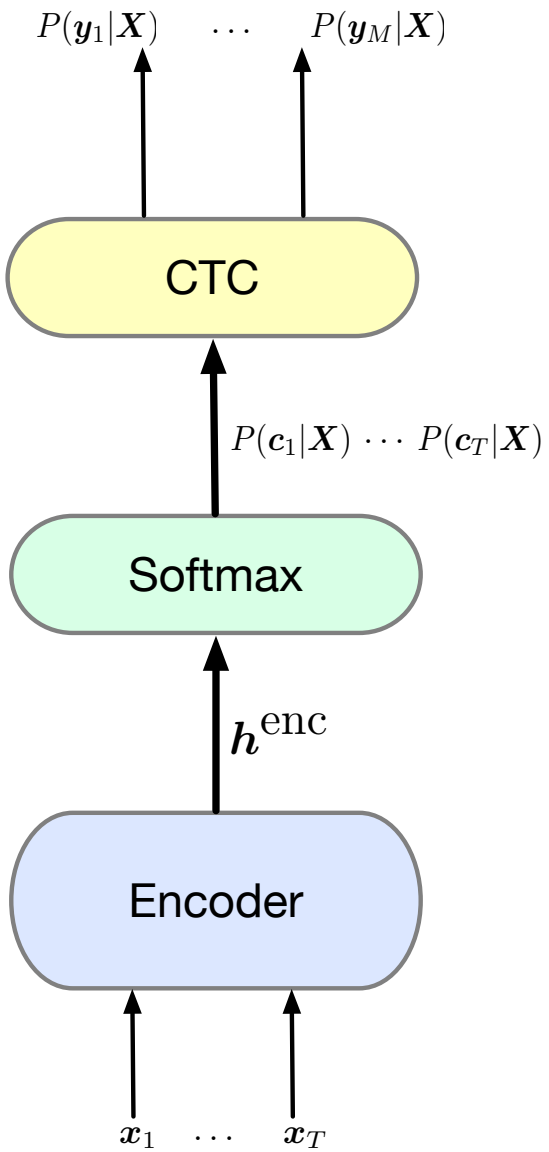
Recap – CTC

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

CTC Model

View CTC as having three components:

- **Encoder**: Deep (bidirectional) LSTM recurrent network which maps acoustic features $X = x_1, \ldots, x_T$ to a sequence of hidden vectors $h^{\text{enc}} = h_1^{\text{enc}}, \ldots, h_T^{\text{enc}}$.
- **Softmax**: Computes the label probabilities $P(c_1|X), \ldots, P(c_T|X)$
- **CTC**: Computes the subword sequence $P(y_1|X), \ldots, P(y_M|X)$



Limitations of CTC

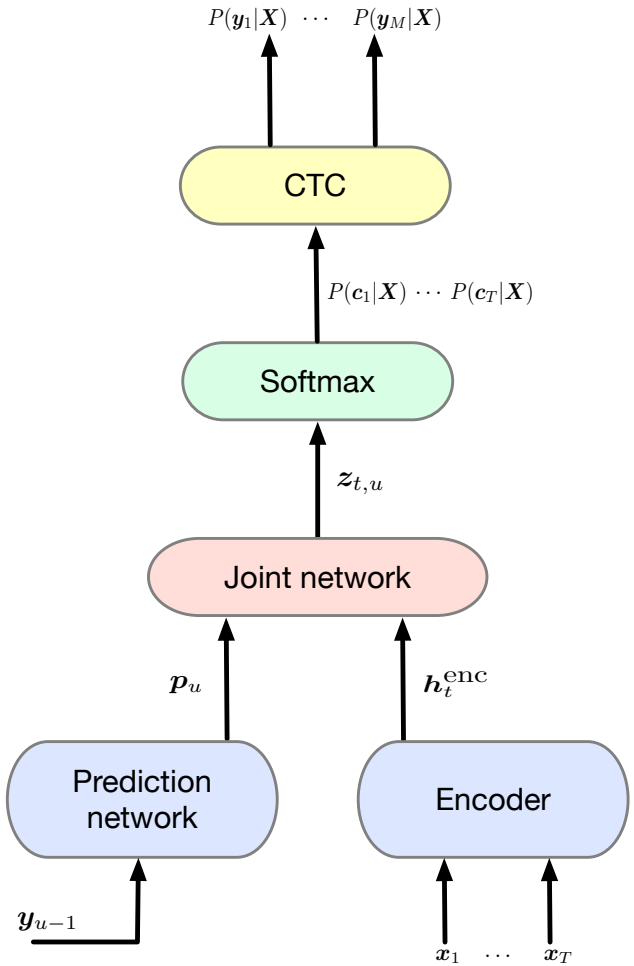
- CTC – pros
  - Can train end-to-end without requiring framewise alignments
  - Sums over all possible alignments (using forward-backward)
  - Preserves monotonic relationship between acoustic frames and output labels

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 $X = x_1, \ldots, x_T$ to hidden vectors $h^{\text{enc}} = h_1^{\text{enc}}, \ldots, h_T^{\text{enc}}$.
- **Prediction network**: Recurrent network which takes the previous output subword label y_{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 $Z = z_1, \ldots, z_T$ by a applying a shallow feed-forward net to h^{enc} and p_u
- Followed by **softmax** and **CTC** components as before



RNN Transducer Model

- RNN transducer can operate left-to-right is 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)
- Google "all-neural" on-device speech recognition uses unidirectional RNN transducers

  https://ai.googleblog.com/2019/03/an-all-neural-on-device-speech.html

RNN-T: forward recursion

Consider an expanded label sequence with an empty symbol

$$\bar{Y} = (\epsilon, y_1, \epsilon, y_2, \ldots, y_U, \epsilon)$$

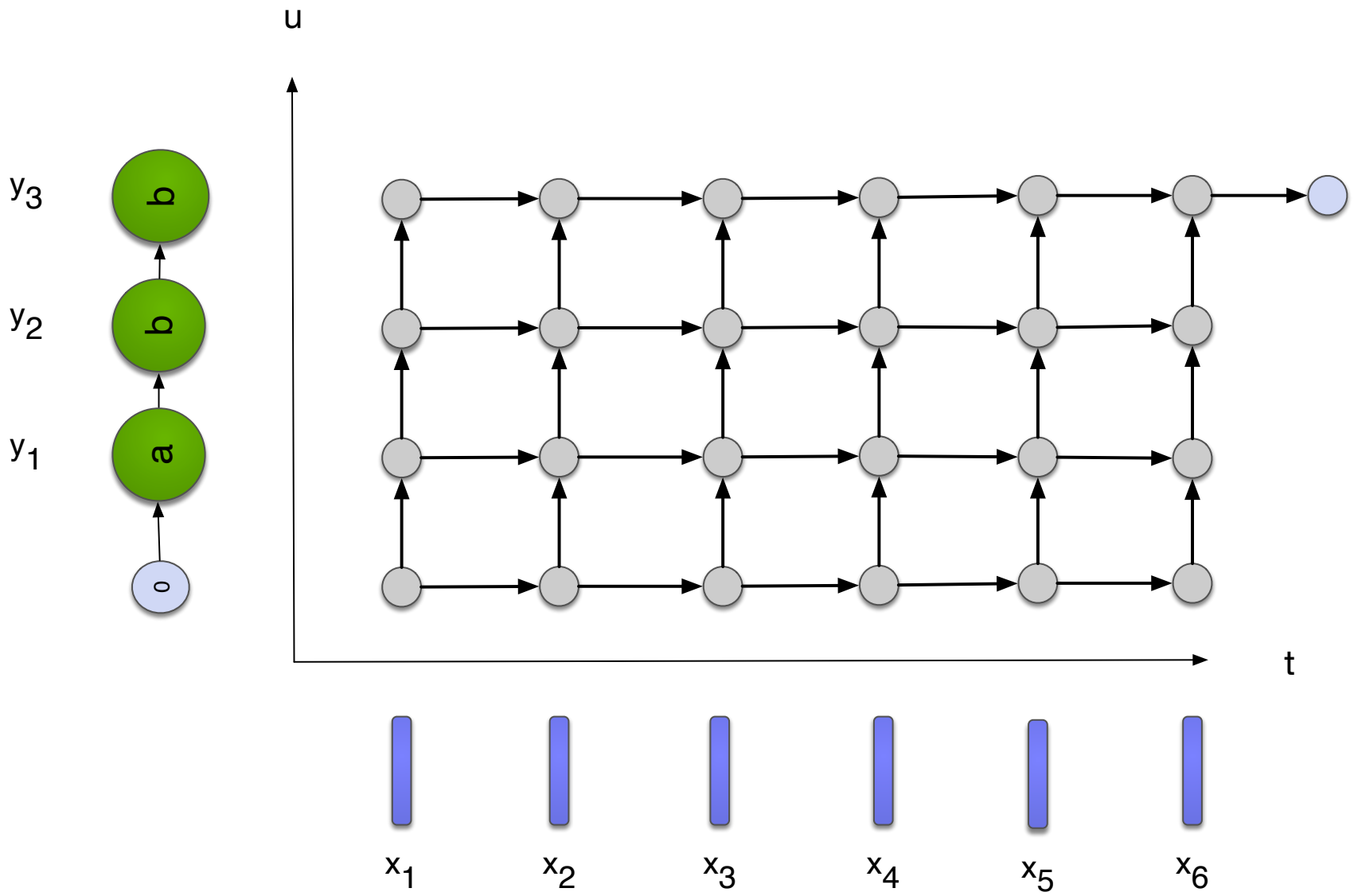
The forward probability is:

$$\alpha_u(t) = P(y_1, \ldots y_u | h_1^{\text{enc}}, \ldots, h_t^{\text{enc}})$$

Define:

$$y(t,u) = P(y_{u+1}|h_t, p_u)$$
$$\varnothing(t,u) = P(\epsilon|h_t, p_u)$$

RNN-T: forward recursion



RNN-T: Forward recursion

- **Initialisation:**

$$\begin{aligned} \alpha_u(0) &= 1 \qquad u = 1 \\ &= 0 \qquad \textit{otherwise} \end{aligned}$$

- **Recursion:**

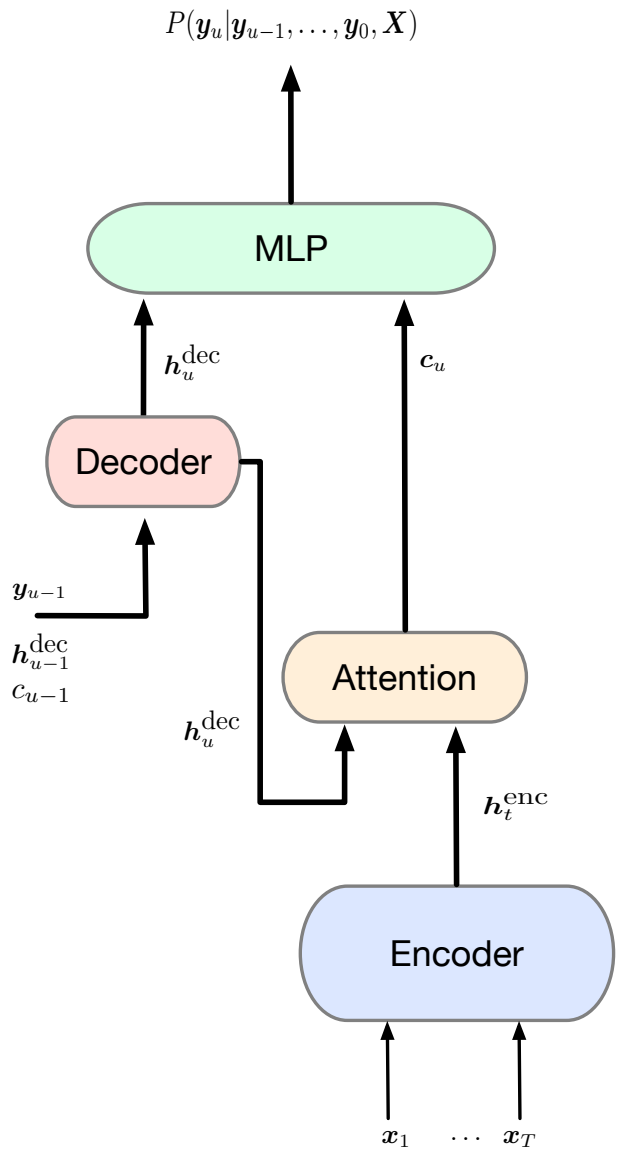
$$\begin{aligned} \alpha_u(t) =& \alpha_u(t-1)\varnothing(t-1,u) \\ &+ \alpha_{u-1}(t)y(t,u-1) \end{aligned}$$

- **Termination:**

$$P(Y|X) = \alpha_U(T)\varnothing(T,U)$$

Attention-based Encoder-Decoder Model

- **Encoder:** Acoustic model using a recurrent network to map acoustic features $X = x_1, \ldots, x_T$ to hidden vectors $h^{\text{enc}} = h_1^{\text{enc}}, \ldots, h_T^{\text{enc}}$.
- **Decoder**: Computes distribution over labels conditioned on previously predicted labels and the acoustics, $P(y_u|y_{u-1}, \ldots, y_0, X)$
- **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)



The Decoder

- The decoder directly generates the output subword sequence Y
- At each decoding step u, the decoder RNN uses the previous output y_{u-1}, the previous decoder RNN hidden state h^{dec}_{u-1}, and the previous context vector c_{u-1} to generate the current decoder hidden state h^{dec}_u

$$h^{\text{dec}}_u = \text{RNN}(h^{\text{dec}}_{u-1}, y_{u-1}, c_{u-1})$$

- The context vector is computed by the attention mechanism

The Attention Mechanism

- The attention mechanism uses the current decoder RNN hidden state h_u^{dec}, and the sequence of encoder hidden states h_t^{enc} to compute an alignment matrix α_{ut}:

$$\alpha_{ut} = \text{Attention}(h_u^{\text{dec}}, h_t^{\text{enc}})$$

- The alignment vector is used as weights in a weighted sum of the encoder hidden states to compute the context vector c_u:

$$c_u = \sum_{t=1}^{T} \alpha_{ut} h_t^{\text{enc}}$$

- The decoder uses the context vector c_u and the current decoder hidden state h_u^{dec} to estimate the subword distribution:

$$y_u \sim \text{LabelDistribution}(c_u, h_u^{\text{dec}})$$

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

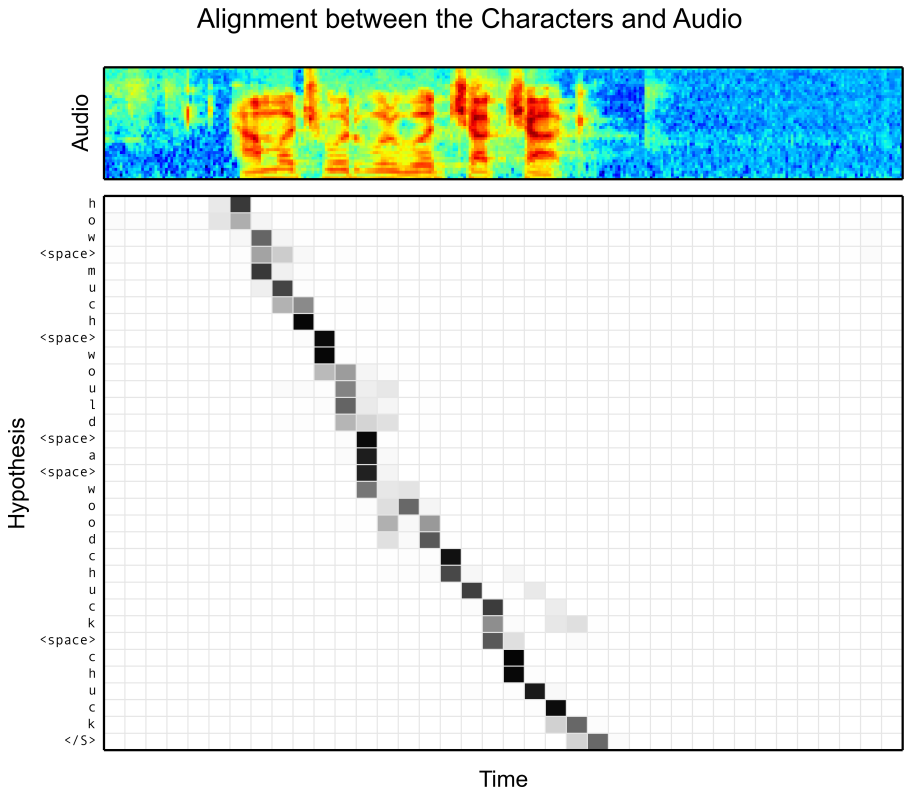
Alignment Vector

- Attention models the alignment between the current output y_u and the input sequence 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

$$e_{ut} = v^T \tanh(W h_u^{\text{dec}} + V h_t^{\text{enc}} + b)$$

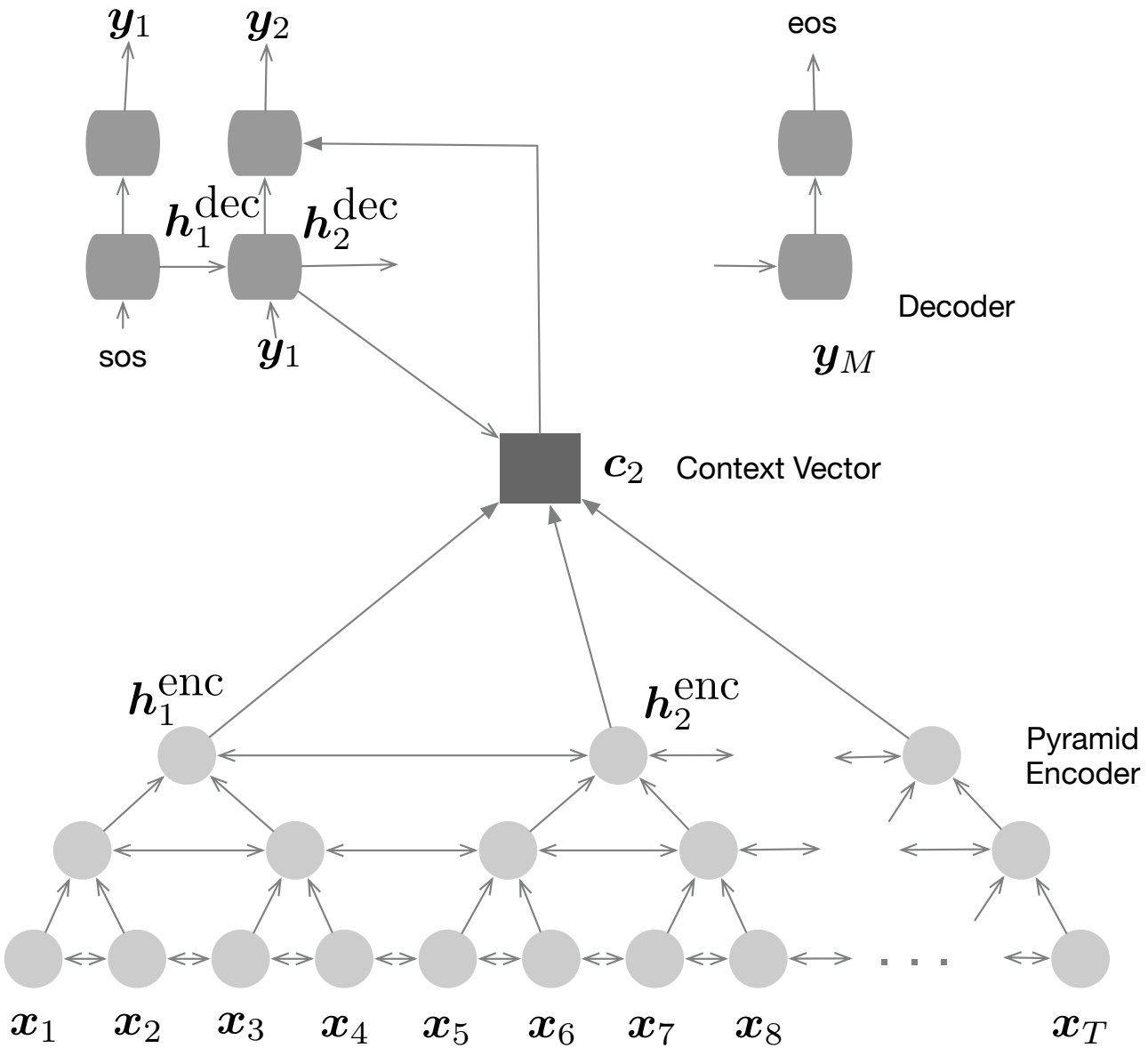
$$\alpha_{ut} = \frac{\exp(e_{ut})}{\sum_k \exp(e_{uk})}$$

Alignment between labels and acoustics



"How much would a woodchuck chuck"

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):

$$h_t^j = RNN(h_t^{j-1}, h_{t-1}^j)$$

  - Pyramid model concatenates consecutive hidden states:

$$h_t^j = pyrRNN([h_{2t-1}^{j-1}, h_{2t}^{j-1}], h_{t-1}^j)$$

  - 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

Learning

- Model trained to maximise the log probability of correct sequences

$$\sum_u \log P(y_u | X, y_{<u})$$

  where $y_{<u}$ is the sequence $y_1, \ldots, y_{u-1}$
- An interesting subtlety: what value should be used for $y_{<u}$?
  - 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[3]	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.	**6.3**	**11.2**	**8.1**	**19.7**	**8.7**

Prabhavalkar et al (2017)

Other Refinements

- Wordpiece models – rather than using single graphemes as labels use multi-grapheme units (up to a word in length) - similar to bye 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

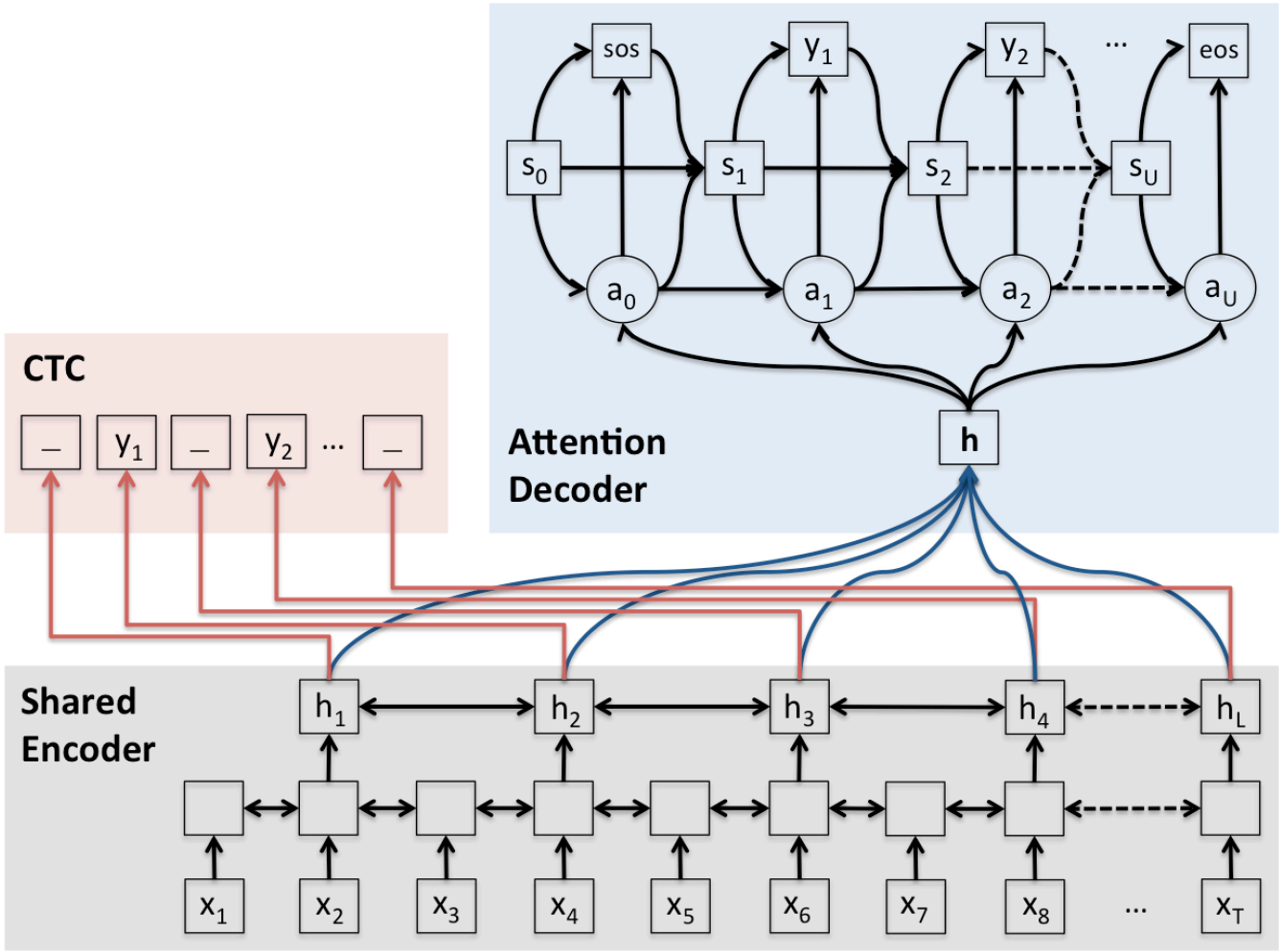
Reduced WER on Voice Search from 9.2% to 5.6% – their hybrid HMM-LSTM system has WER of 6.7% on this task

Chiu et al (2018)

Hybrid CTC/Attention

- 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)

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

Reading

- 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
- Chan et al (2016), "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition", IEEE ICASSP, pp. 4960-4964 https://ieeexplore.ieee.org/abstract/document/7472621
- Chiu et al (2018), "State-of-the-art sequence recognition with sequence-to-sequence models", IEEE ICASSP. https://arxiv.org/abs/1712.01769
- 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