Attention-based models for ASR

Desh Raj

March 11, 2019

Outline

- Listen, Attend, and Spell architecture: attention-based decoding
- Self-attention for encoder in LAS
- Self-attention layer in TDNN model (Kaldi)

Listen, Attend and Spell

“Listen, Attend, and Spell: A neural network for large vocabulary conversational speech recognition.” William Chan, Navdeep Jaitly, Quoc Le, Oriol Vinyals. (ICASSP 2016)

Motivation

- Conventional ASR models are complicated.
- Involve several components: Lexicon, acoustic model, language model, etc.
- Make several assumptions:
  - Conditional independence between frames
  - Markov assumptions

What is required?

- Basic problem in ASR: Given **x**, find **y**.
- No assumptions, just chain rule.

$$p(\mathbf{y} \mid \mathbf{x}) = \prod_i p(y_i \mid \mathbf{x}, y_{<i})$$

$$\mathbf{x} = (x_1, \ldots, x_T), \quad \mathbf{y} = (y_1, \ldots, y_S)$$

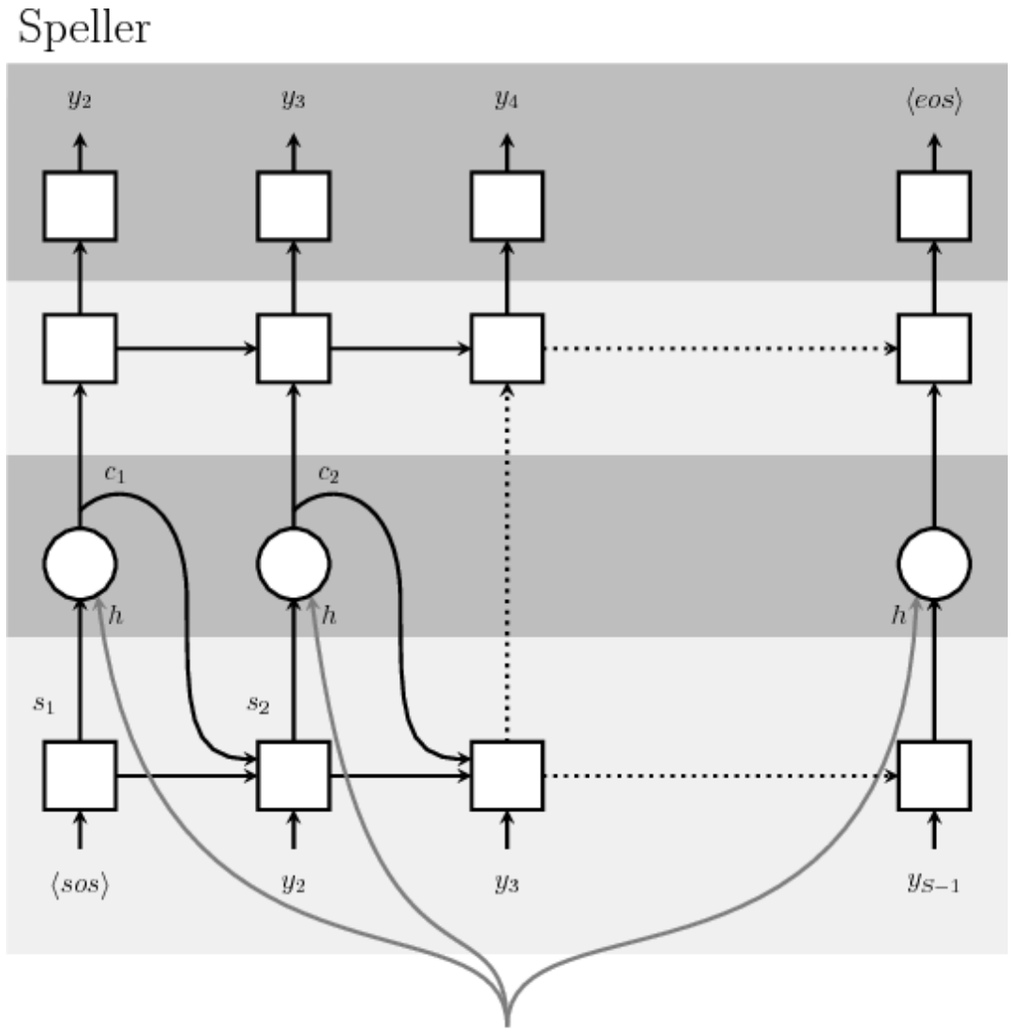
$$p(\mathbf{y} \mid \mathbf{x}) = \prod_i p(y_i \mid \mathbf{x}, y_{<i})$$

- This makes the model:
  - Discriminative
  - End-to-end

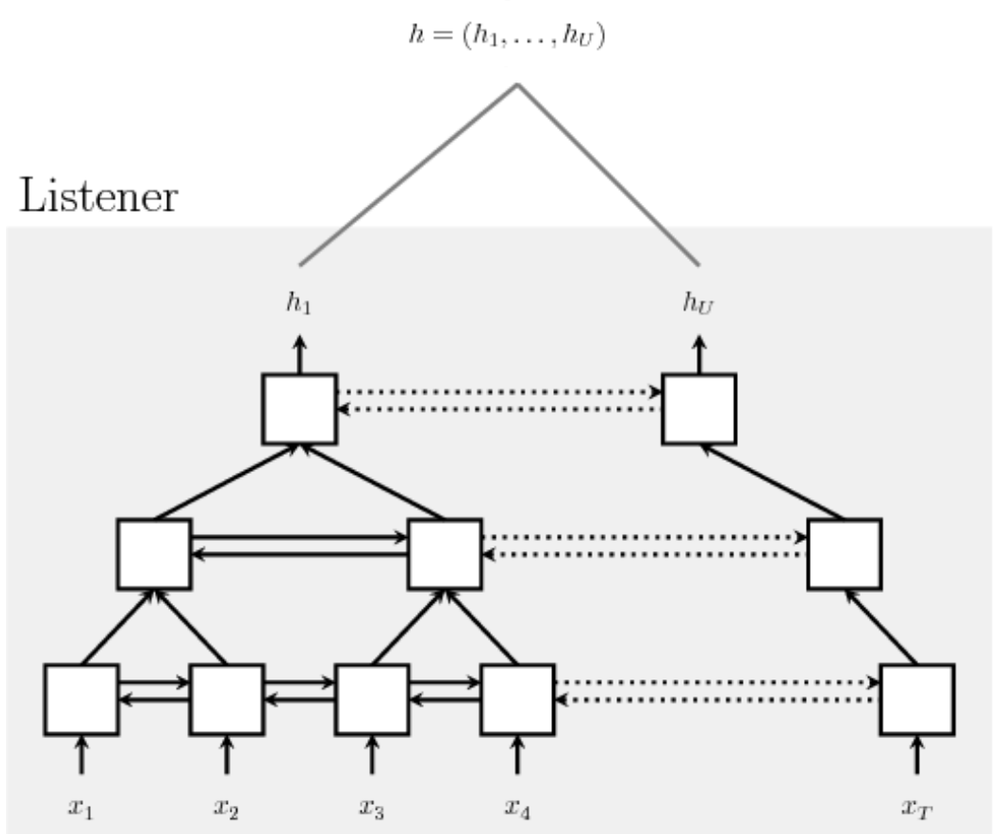
A note on “end-to-end”

- **End-to-end model**: Encompasses all components of the ASR pipeline into the trainable parameters.

- **End-to-end training**: Adjust/train acoustic model parameters to work well with fixed components like lexicon and language model.



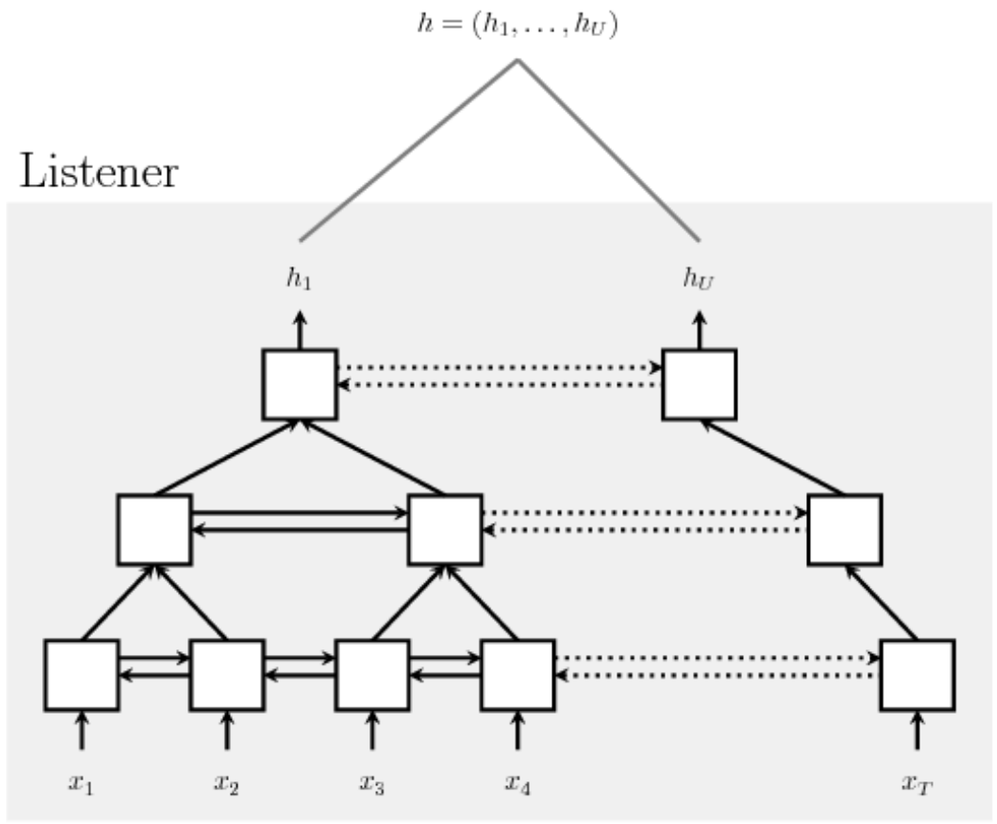
AttendAndSpell



Listen

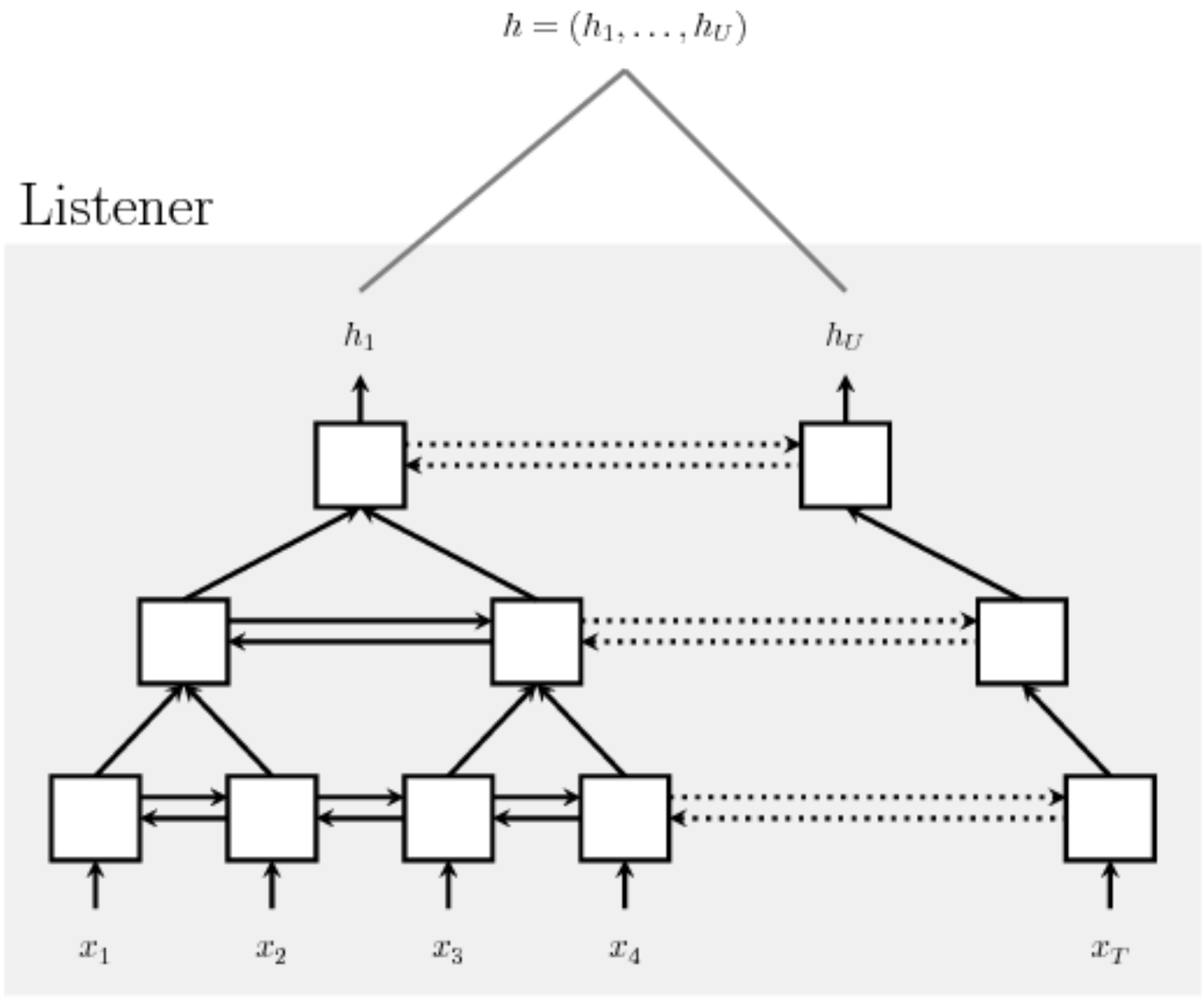
The Listener

- Transform input **x** into a higher-level representation **h**.
- $U < T$



Pyramidal BLSTM

***T* can be thousands of frames long!**



1. Easier for decoder to extract relevant information

2. Learn nonlinear feature representation of acoustic signals

3. Reduces computational complexity of decoder

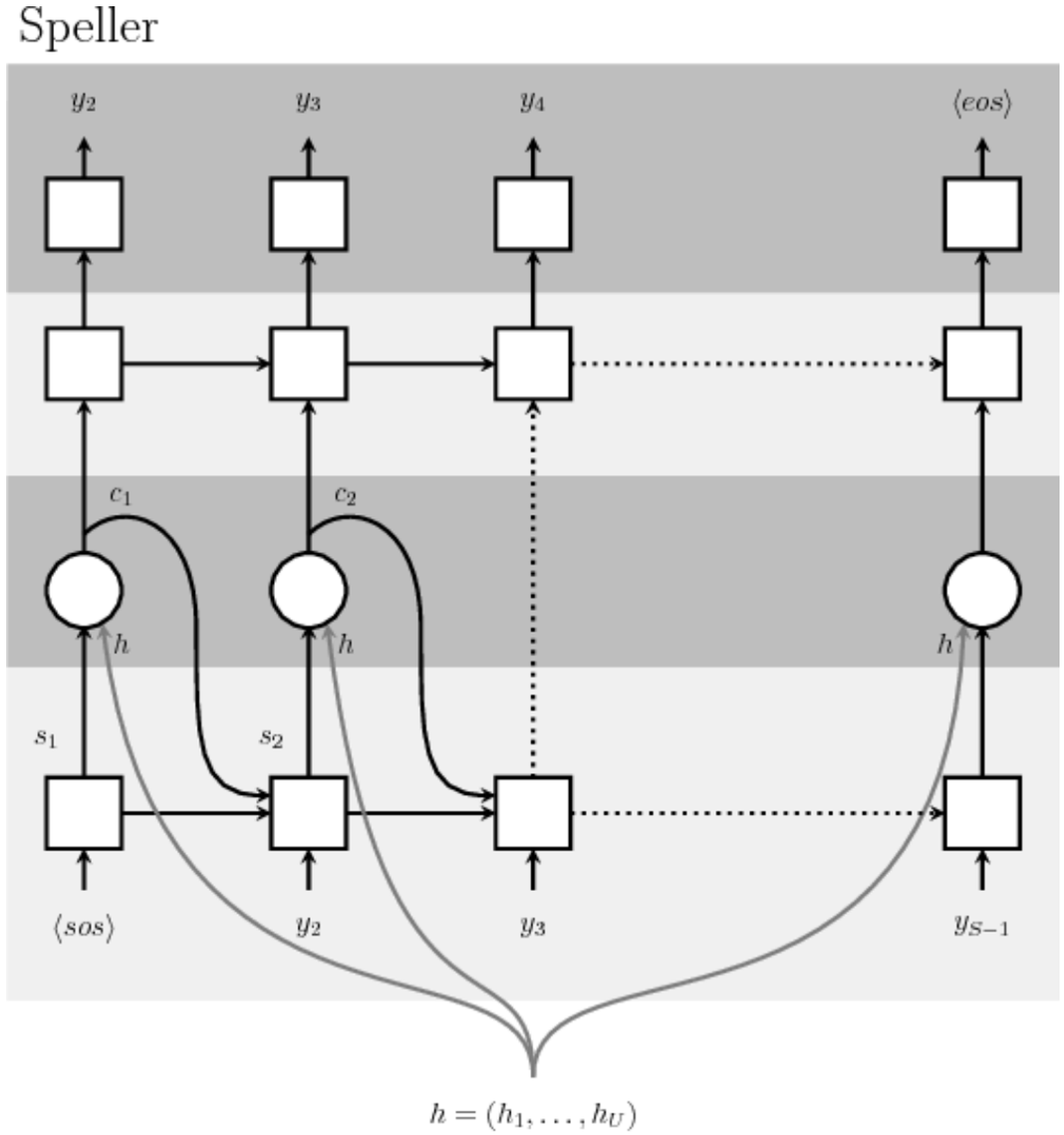
The Speller

- Attention-based LSTM transducer
- Content-based MLP attention is used here

$$e_{i,u} = \langle \phi(s_i), \psi(h_u) \rangle$$

$$\alpha_{i,u} = \frac{\exp(e_{i,u})}{\sum_{u'} \exp(e_{i,u'})}$$

$$c_i = \sum_u \alpha_{i,u} h_u$$



Training in LAS

$$\tilde{\theta} = \max_{\theta} \sum_{i} \log P(y_i | \mathbf{x}, \tilde{y}_{<i}; \theta)$$

- Character-based training can be done using ground truth.
- But, no ground truth at test time! Errors may propagate?
- Sample character from model with 10% probability.

Decoding

- Left-to-right beam search
- Beams are rescored using an independently trained LM
- Normalize with number of characters to mitigate bias for shorter utterances

$$s(\mathbf{y}|\mathbf{x}) = \frac{\log P(\mathbf{y}|\mathbf{x})}{|\mathbf{y}|_c} + \lambda \log P_{\mathrm{LM}}(\mathbf{y})$$

Results

- 2000 hours (3 million utterances) of Google Voice Search (so not really “conversational”)
- 20 times data augmentation

Model	Clean WER	Noisy WER
CLDNN-HMM [22]	8.0	8.9
LAS	14.1	16.5
LAS + LM Rescoring	10.3	12.0

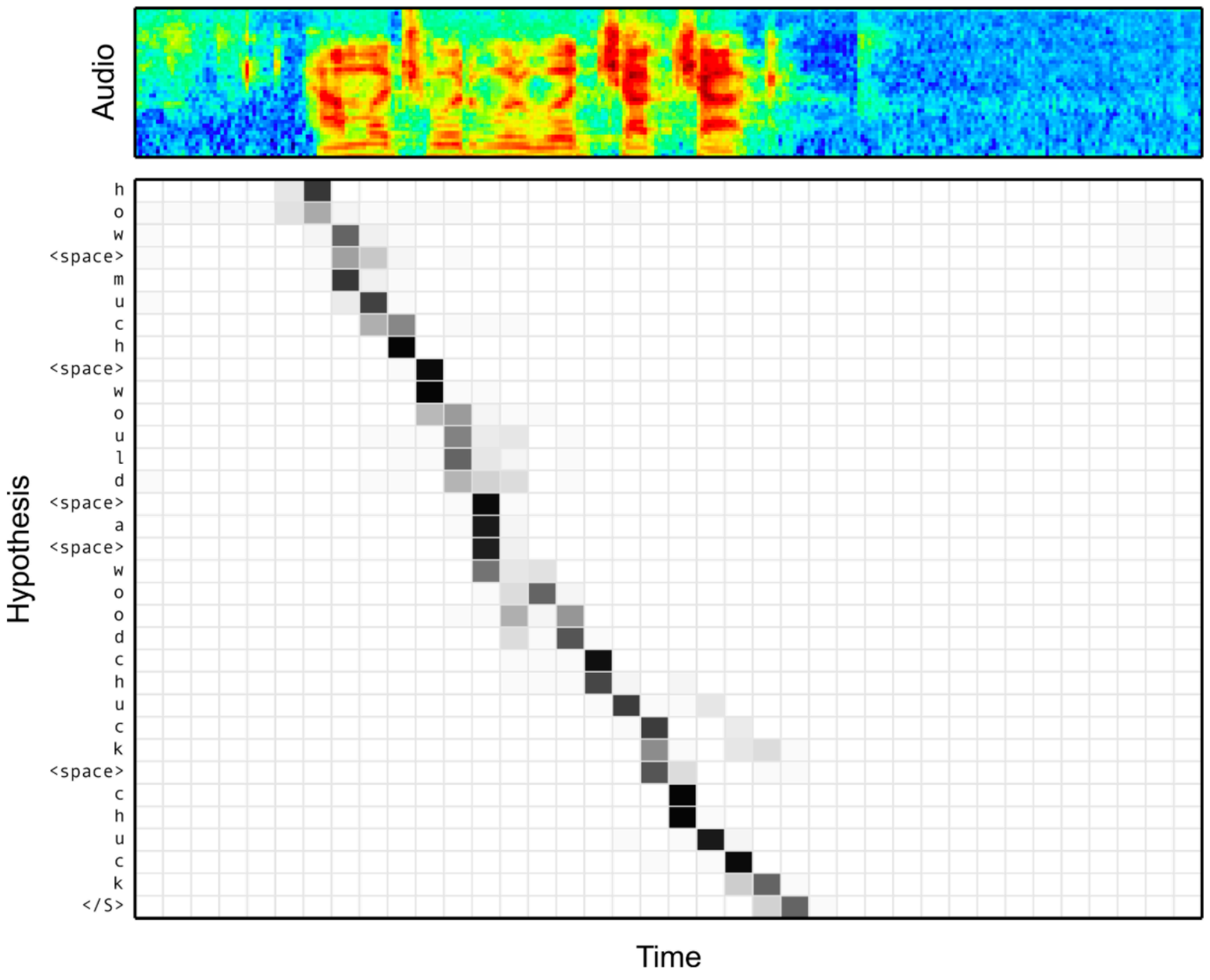


Fig. 2: Alignments between character outputs and audio signal produced by the Listen, Attend and Spell (LAS) model for the utterance "how much would a woodchuck chuck". The content based attention mechanism was able to identify the start position in the audio sequence for the first character correctly. The alignment produced is generally monotonic without a need for any location based priors.

From attention-based decoder to self-attention-based encoder

“Self attentional acoustic models.” Matthias Sperber, Jan Niehues, Graham Neubig, Sebastian Stüker, Alex Waibel. (ICASSP 2018)

Preliminaries

- Attention and its types

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

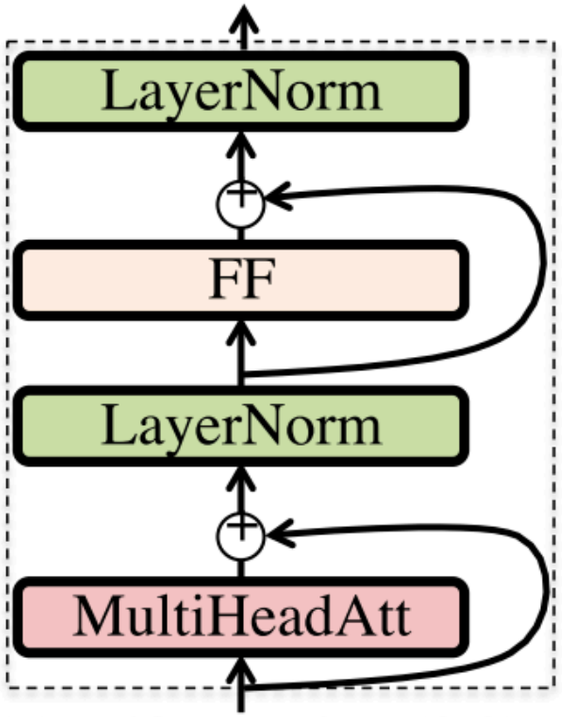
Source: https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

Preliminaries

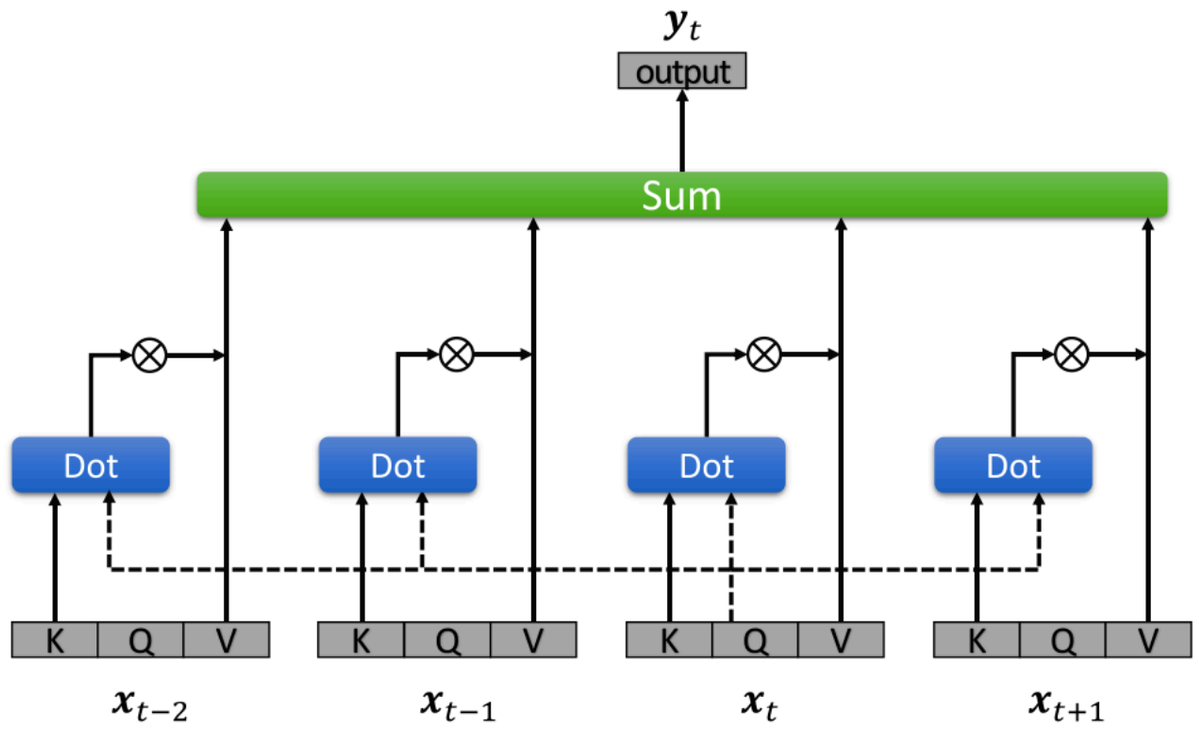
- **Self-attention**
  - Compute a representation of the sequence, i.e., output length = input length
  - Uses pairwise similarity scores

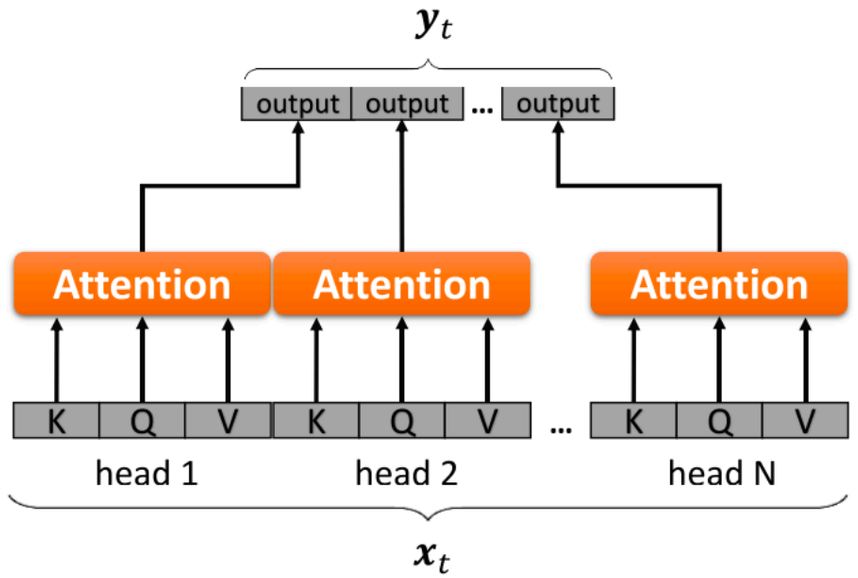
Self-attention in LAS

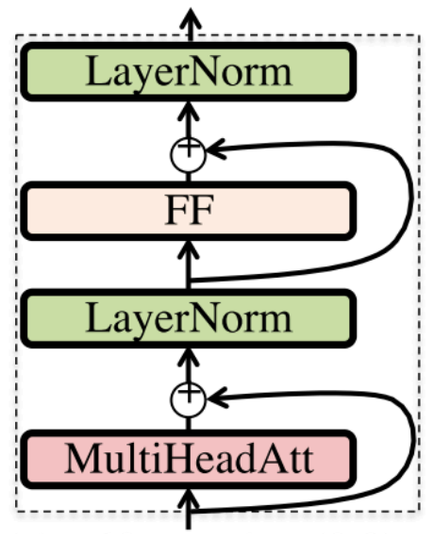
- Encoder in original LAS is pyramidal BLSTM.
- Replace this with a self-attention based encoder. Why?
- Computationally more efficient.
- Direct conditioning on short and long term text, without the need to pass information through RNN states.



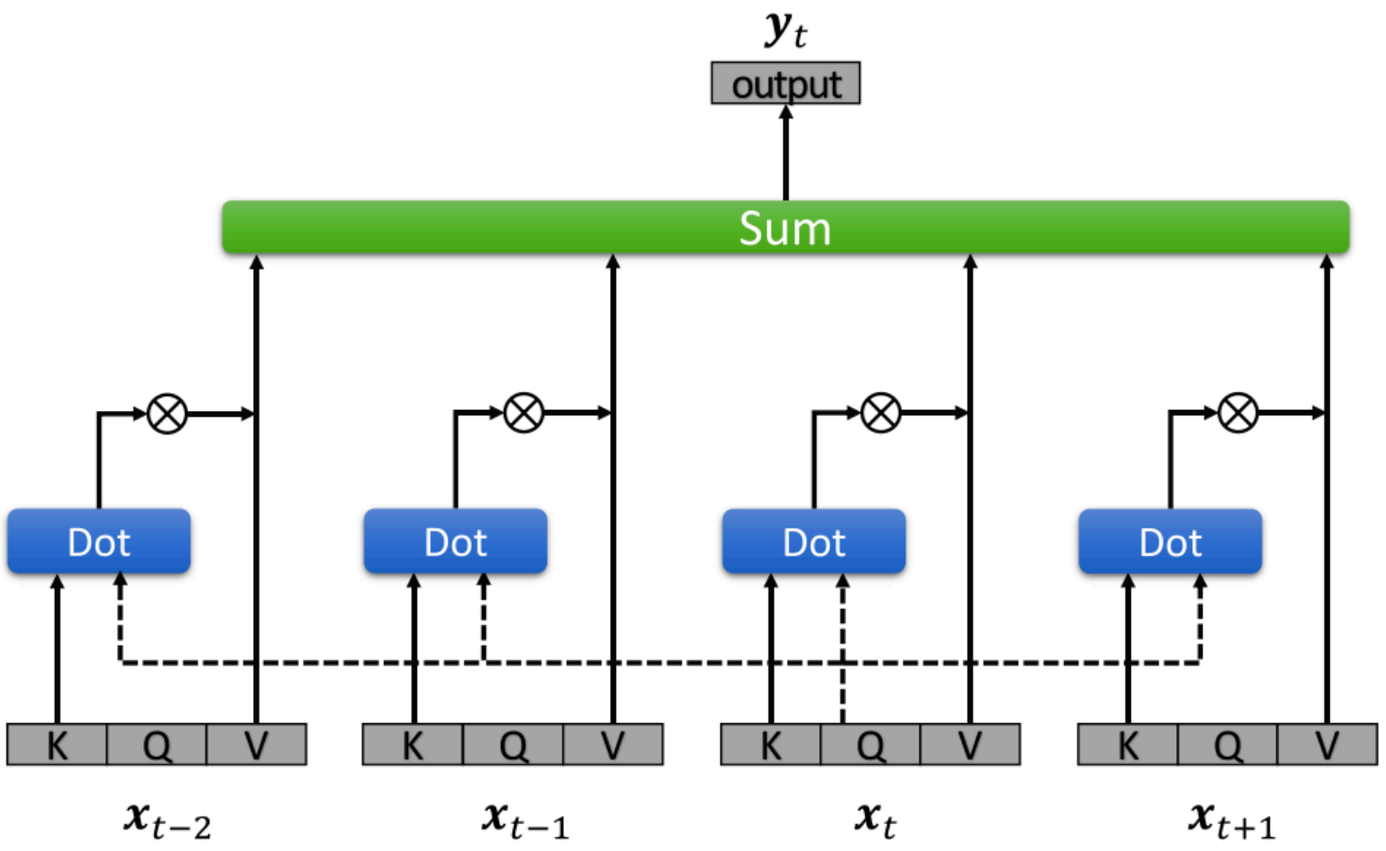
(c) self-attention (§ 3)

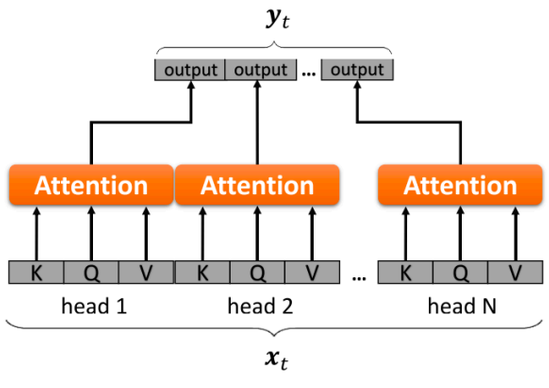






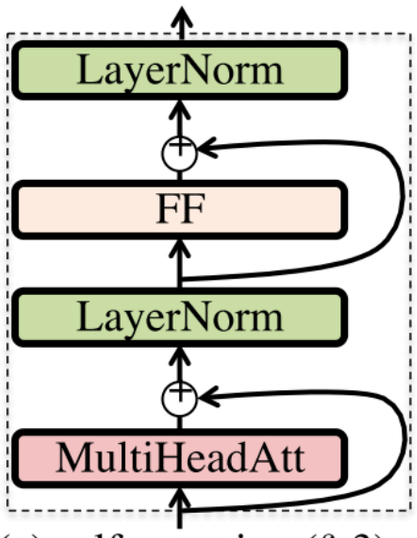
(c) self-attention (§ 3)



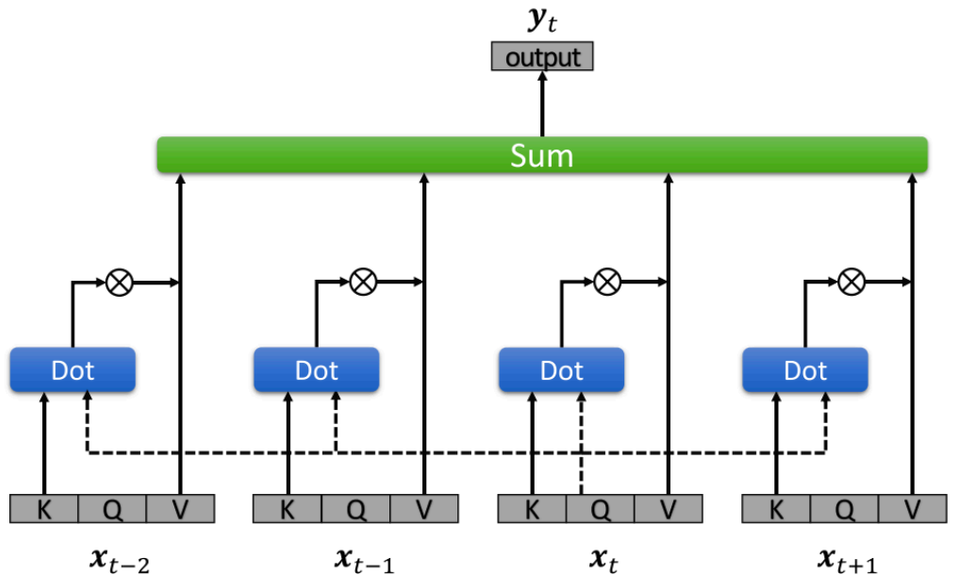


$$Q_i = XW_i^{\mathrm{Q}}, K_i = XW_i^{\mathrm{K}}, V_i = XW_i^{\mathrm{V}} \quad (1)$$

$$\mathrm{head}_i = \mathrm{softmax}(\frac{Q_i K_i^{\mathrm{T}}}{\sqrt{d}})V_i \quad \forall i \quad (2)$$



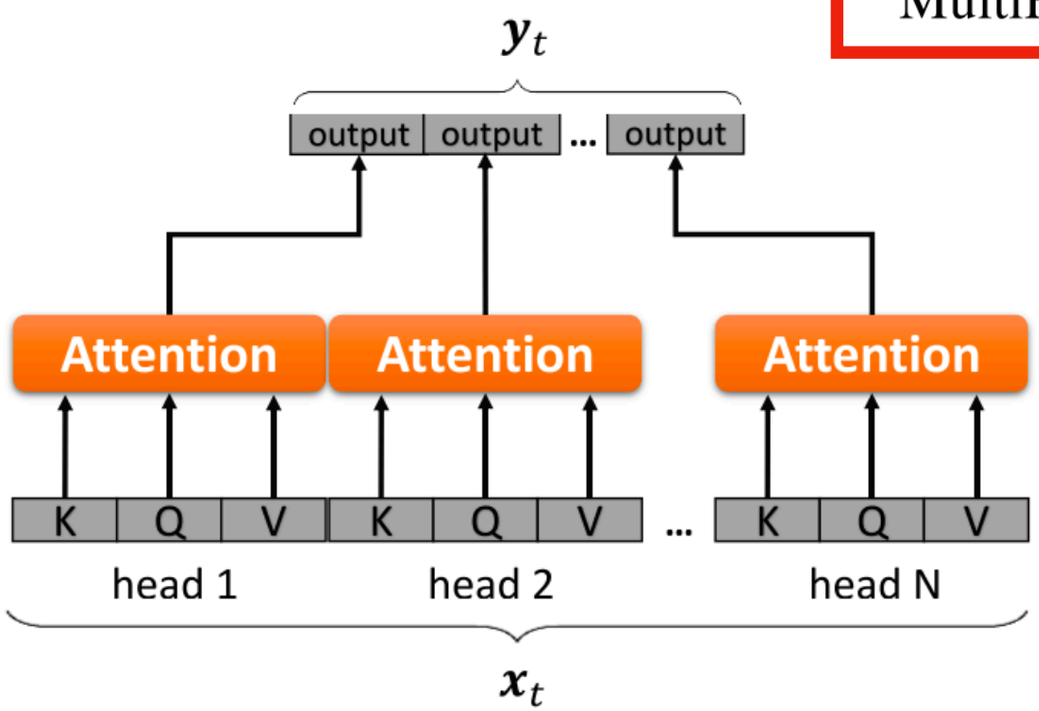
(c) self-attention (§ 3)

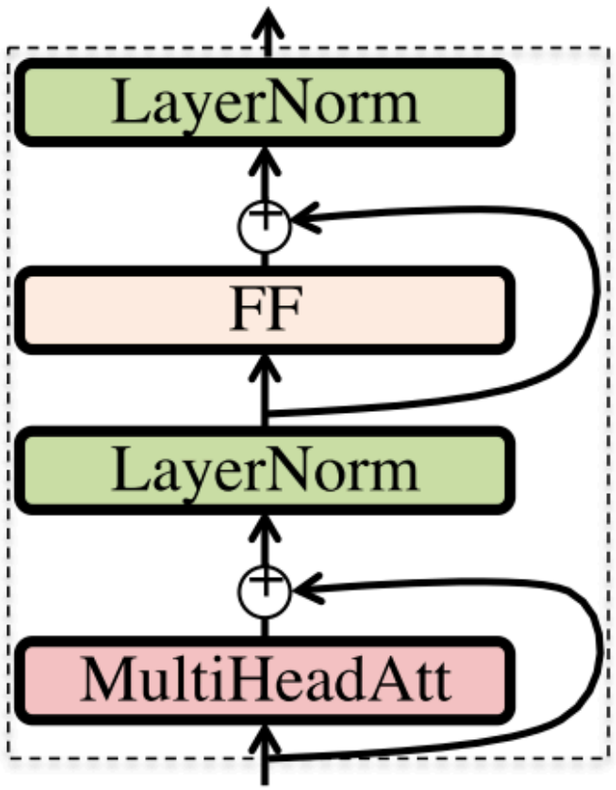


$$Q_i = XW_i^{\mathrm{Q}}, K_i = XW_i^{\mathrm{K}}, V_i = XW_i^{\mathrm{V}} \quad (1)$$

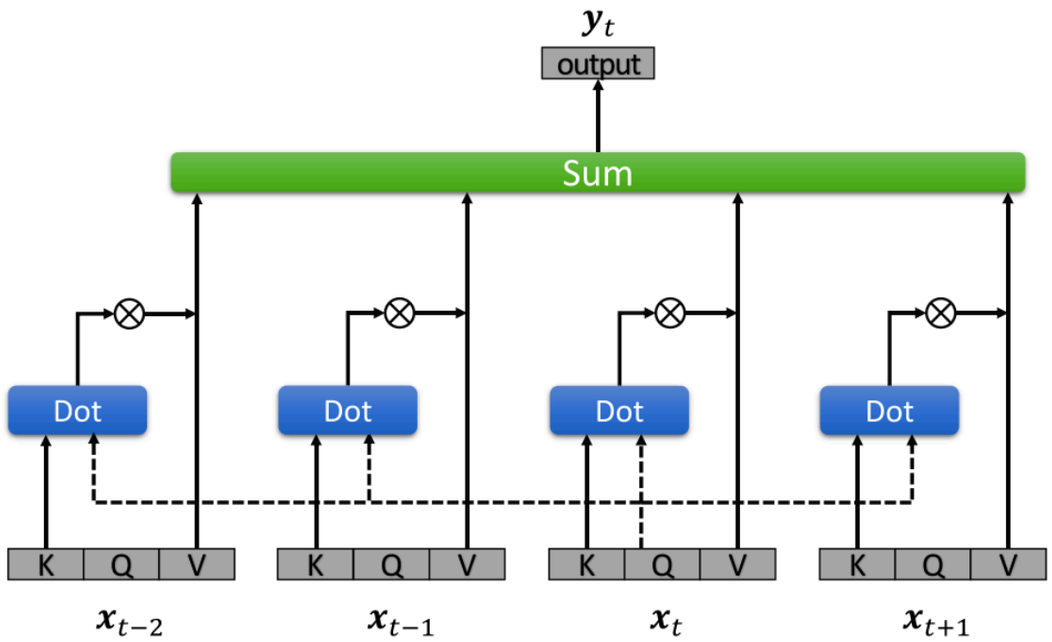
$$\mathrm{head}_i = \mathrm{softmax}(\frac{Q_i K_i^{\mathrm{T}}}{\sqrt{d}})V_i \quad \forall i \quad (2)$$

$$\mathrm{MultiHeadAtt} = \mathrm{concat}(\mathrm{head}_1, \mathrm{head}_2, \ldots) \quad (3)$$





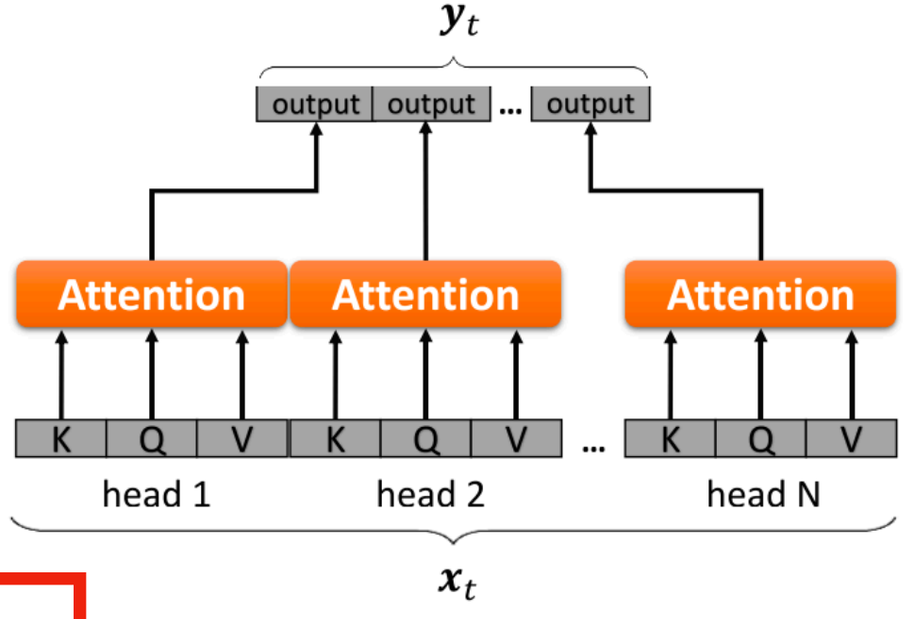
(c) self-attention (§ 3)



$$Q_i = XW_i^{\mathrm{Q}}, K_i = XW_i^{\mathrm{K}}, V_i = XW_i^{\mathrm{V}} \quad (1)$$

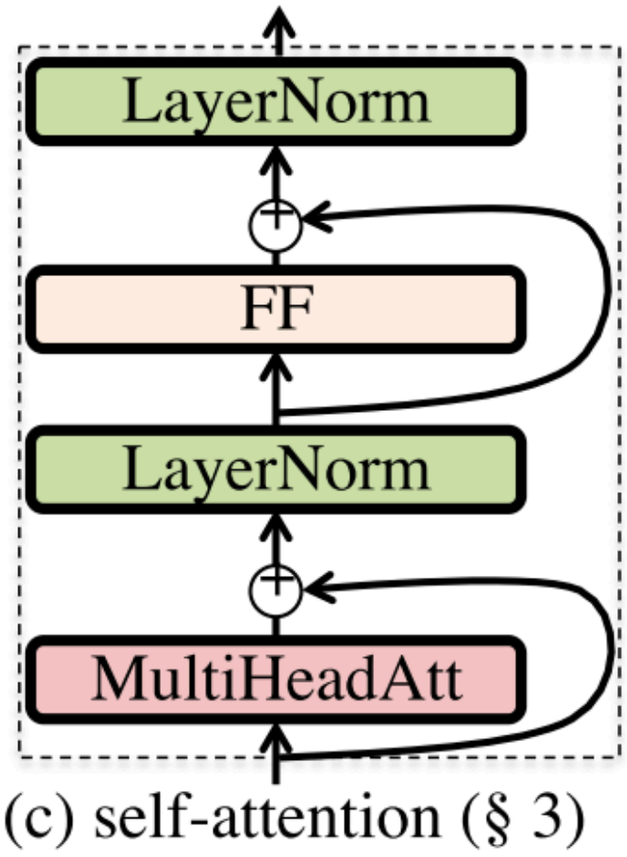
$$\mathrm{head}_i = \mathrm{softmax}(\frac{Q_i K_i^{\mathrm{T}}}{\sqrt{d}})V_i \quad \forall i \quad (2)$$

$$\mathrm{MultiHeadAtt} = \mathrm{concat}(\mathrm{head}_1, \mathrm{head}_2, \ldots) \quad (3)$$



$$\mathrm{MidLayer} = \mathrm{LayerNorm}\,[\mathrm{MultiHeadAtt} + X] \quad (4)$$

$$\mathrm{SAL} = \mathrm{LayerNorm}\,[\mathrm{FF}\,(\mathrm{MidLayer}) + \mathrm{MidLayer}] \quad (5)$$



(c) self-attention (§ 3)

$$\text{MidLayer} = \text{LayerNorm}\left[\text{MultiHeadAtt} + X\right] \quad (4)$$

$$\text{SAL} = \text{LayerNorm}\left[\text{FF}\left(\text{MidLayer}\right) + \text{MidLayer}\right] \quad (5)$$

- **LayerNorm** is like BatchNorm, but instead of normalizing over mini batch, it normalizes all features of a single input.
- Why?
  - BatchNorm is difficult for RNNs.
  - Need large minibatch size to estimate correct mean and variance.

Problems in acoustic modeling with self-attention encoders

1. Very long frame sequences -> quadratic memory requirement

2. Encoding positional information in the model

3. Effective modeling of context relevance -> frame vectors have much less information than word vectors

1. Downsampling

- Reshape before every self-attention block
- Similar to pyramidal LSTM concept (but makes less sense?)

$$X \in \mathbb{R}^{l \times d} \rightarrow_{\text{reshape}} \hat{X} \in \mathbb{R}^{\frac{l}{a} \times ad}$$

2. Position Modeling

- RNNs encode position naturally in the model.
- But self-attention is position agnostic. How to solve?

1. Concatenate positional embeddings to the input features
2. Hybrid models: Use stacked or interleaved LSTM layers with self-attention

3. Attention biasing

- Self-attention head equation -> No explicit way of controlling the context range

$$\textbf{head}_i = \textbf{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d}}\right) V_i$$

3. Attention biasing

- Self-attention head equation -> No explicit way of controlling the context range
- Add a bias matrix *M*

$$\textbf{head}_i = \textbf{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d}} + M\right) V_i$$

- **Hard masking**: all attention weights outside a context window are set to 0
- **Soft masking**: Gaussian mask is used

$$M_{jk} = \frac{-(j-k)^2}{2\sigma^2}.$$

Some Results on Tedlium

- Although WER is almost same as LAS model, training speed is much faster.

Table 1: *Comparison to baselines. Training speed (char/sec) was measured on a GTX 1080 Ti GPU.*

model	dev WER	test WER	char/sec
pyramidal	15.83	16.16	1.1k
LSTM/NiN	14.57	14.70	1.1k
stacked hybrid	16.38	17.48	2.4k
interleaved hybrid	15.29	16.71	1.5k

Some Results on Tedlium

- Poor results without RNNs in the encoder.

Table 2: *WER results on position modeling.*

model	dev	test
add (trig.)	diverged	
concat (trig.)	30.27	38.60
concat (emb.)	29.81	31.74
stacked hybrid	16.38	17.48
interleaved hybrid	15.29	16.71

Some Results on Tedlium

- Gaussian masking with large variance gives best results.

Table 3: *WER results on attention biasing.*

model	dev	test
stacked hybrid	16.38	17.48
+ local masking	15.42	16.17
+ Gauss mask (init. small)	16.05	16.96
+ Gauss mask (init. large)	14.90	15.89
interleaved hybrid	15.29	16.71
+ local masking	15.44	16.19
+ Gauss mask (init. small)	16.43	16.89
+ Gauss mask (init. large)	15.00	15.82

Interpretability of attention heads

- Train with phoneme sequences rather than characters.
- Use soft alignment from decoder attention scores (recall LAS)
- Certain attention heads respond to certain types of acoustic events.

Table 4: *Analysis of function of attention heads. Note that we conducted a small amount of cherry picking by removing 4 outliers that did not seem to fit categories (OY from head 1, ZH from head 3, EH and ER from head 7). Entropy is computed over the correlation scores, truncated below 0.*

i	top phonemes	entropy	comments
1	S, TH, Z	3.7	sibilants
2	</s>	1.9	silence
3	UW, Y, IY, IX	3.6	”you” diphthong
	B, G, D		voiced plosives
	M, NG, N		nasals
4	XM, AW, AA, AY, L, AO, AH	3.2	A, schwa
5	ZH, AXR, R	3.5	R, ZH
6	ZH, Z, S	3.2	sibilants
	IY, IH, Y, UW		”you” diphthong
7	S, </s>, TH CH, SH, F	3.4	fricative, noise
8	mixed	3.7	unfocused

Self attention in Kaldi

“A time-restricted self attention layer for ASR.” Daniel Povey, Hossein Hadian, Pegah Ghahremani, Ke Li, Sanjeev Khudanpur. (ICASSP 2018)

The model

- Add a self-attention layer to TDNN or TDNN-LSTM models.

- Training with lattice-free MMI (covered last week)

- **Note:** This is an end-to-end training, not end-to-end model, i.e., we still use external LMs, lexicon, etc.

Positional encodings

- Similar to the hard masking technique in the last paper.
- One-hot relative position encoding concatenated with feature vectors

$$\boldsymbol{y_t} = \sum_{\tau=t-L}^{t+R} c_t(\tau)\mathbf{extend}(\boldsymbol{v_t}, \tau, t)$$

$$c_t(\tau) = \frac{\exp(\boldsymbol{q_t} \cdot \mathbf{extend}(\boldsymbol{k_t}, \tau, t))}{Z_t}$$

Experiments

- Datasets: WSJ, TED-LIUM, Switchboard, AMI
- Extensive investigation into:
  - Number of self-attention heads
  - Key/value dimensions
  - Context size

Some Results

- Self-attention layer is more effective when used towards the end of the network.

Database	Test set	Baseline	L2	L4	L6	L7
Switchboard	eval/fullset	15.0	15.2	14.9	14.8	**14.6**
	eval/callhm	19.9	20.2	19.8	19.7	**19.5**
TED-LIUM	dev	8.6	8.4	8.4	**8.3**	8.4
	test	8.9	8.9	8.9	8.7	**8.5**

Table 2. Effect of location of the attention layer in the network. Li means layer i is attention.

Some Results

- Mid-size contexts are most effective. Ideal context size was determined as [-15,6].

Database	Test set	Baseline	Total context				
			13	19	25	31	37
Switchboard	eval/fullset	15.0	14.8	14.6	**14.5**	14.6	14.7
	eval/callhm	19.9	19.7	19.4	**19.3**	**19.3**	**19.3**
TED-LIUM	dev	8.6	8.4	**8.3**	8.4	8.6	8.4
	test	8.9	8.7	8.7	**8.6**	8.7	8.7

Table 3. Effect of symmetric context size.

Some More Results

- 60 self-attention heads gave best WER results.
- A key/value dimension ratio of about 0.5 is ideal.
- In TDNN-LSTMs, it sped up decoding by 20%.

Conclusions

- (Self) Attention is used in place of RNNs in encoders to speed up the encoding.

- But we need to think about very long frame sequences and position modeling.

- In decoders, attention is used to avoid using conditional independence assumptions and to avoid forgetting long contexts.

“Attention is all you need.” Well, maybe not!

– Some ASR person