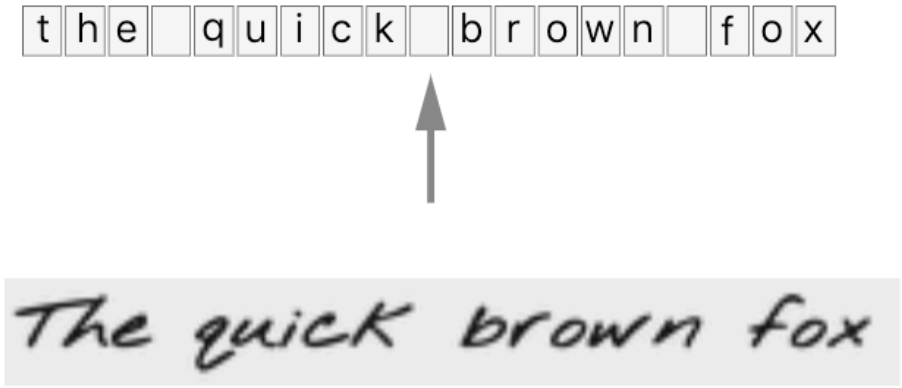
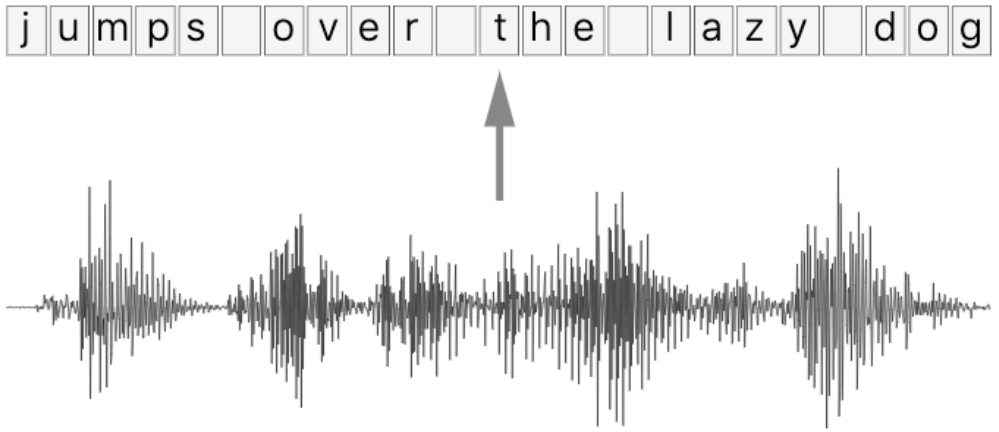
Sequence to Sequence

27.5.2019

Sequence-to-Sequence



Handwriting recognition: The input can be (x, y) coordinates of a pen stroke or pixels in an image.



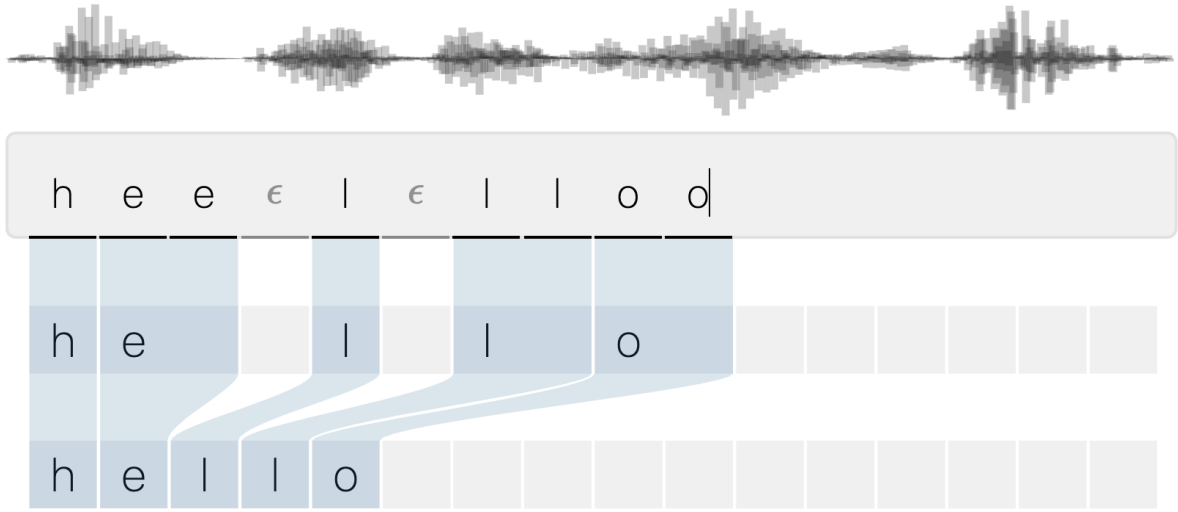
Speech recognition: The input can be a spectrogram or some other frequency based feature extractor.

https://distill.pub/2017/ctc/

Sequence-to-Sequence

- Speech recognition
- Handwriting recognition
- Machine translation
- Speech synthesis (text to speech)
- Summarization
- Text generation

Speech Recognition: Hybrid HMM-DNN

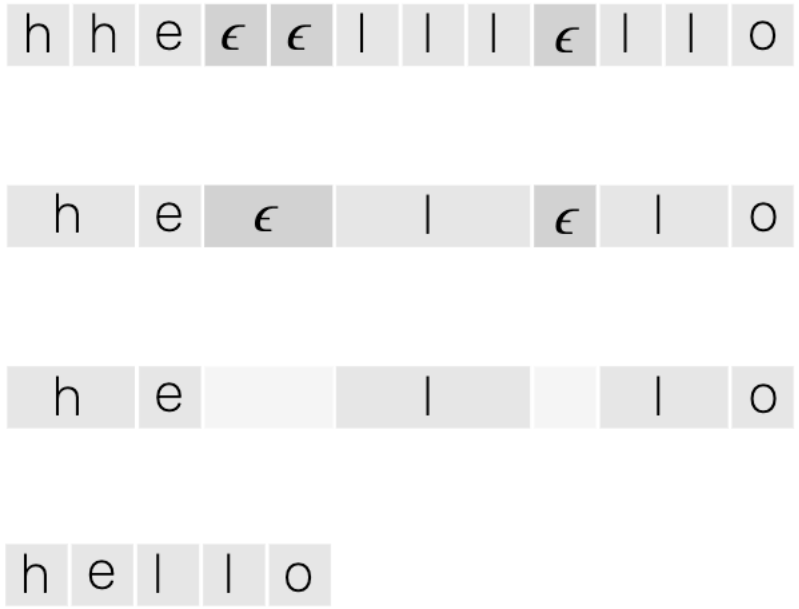


- Use feed-forward net to predict phones
- Use existing HMM/graph structure to form words and sentences
- **Problem: requires alignment for frame-wise training!**
- *(Did not work for translation and generation)*

Connectionist Temporal Classification (CTC)

- Map from $X = [x_1, x_2, \ldots, x_T]$ to $Y = [y_1, y_2, \ldots, y_U]$, where
  - Both X and Y can vary in length.
  - The ratio of the lengths of X and Y can vary.
  - We don’t have an accurate alignment of X and Y
- Key ideas:
  - Use epsilon padding to ensure equal lengths
  - Marginalize over all possible alignments

CTC – Blank (eps) Symbol

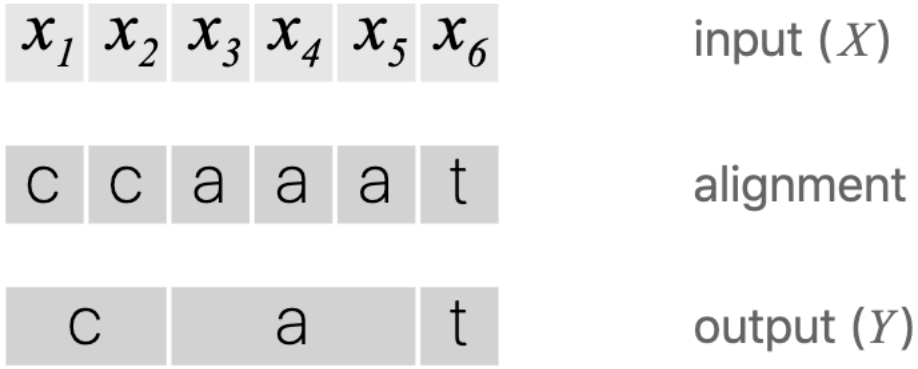


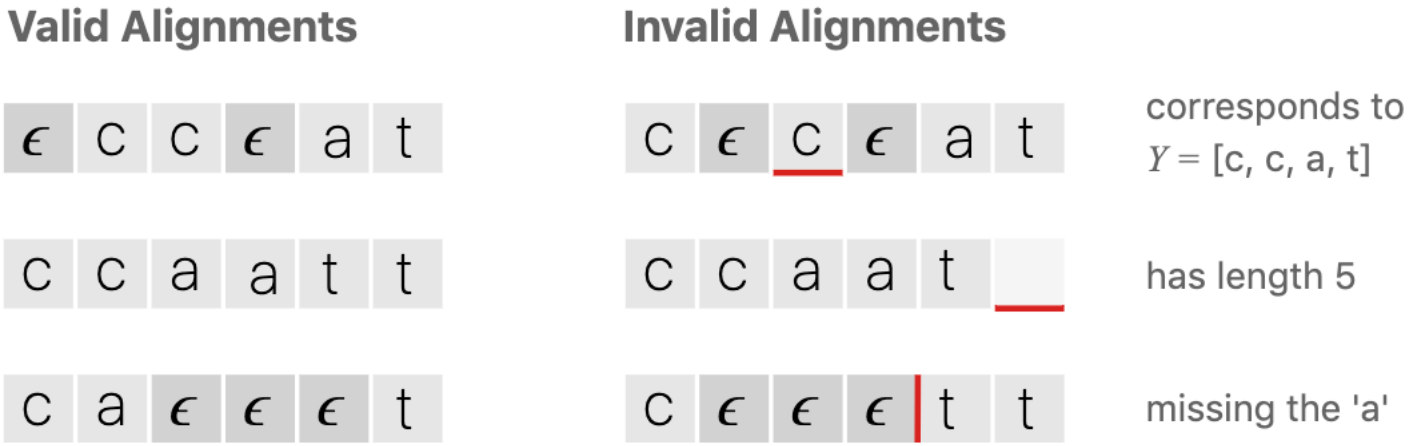
First, merge repeat characters.

Then, remove any ϵ tokens.

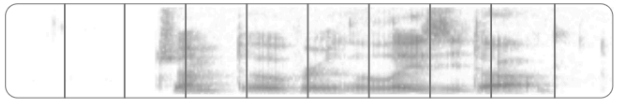
The remaining characters are the output.

Example

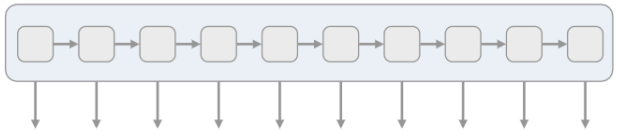

x_1 x_2 x_3 x_4 x_5 x_6
input (X)
c c a a a t
alignment
c a t
output (Y)


Valid Alignments
Invalid Alignments
ϵ c c ϵ a t
c ϵ c ϵ a t
corresponds to
Y = [c, c, a, t]
c c a a t t
c c a a t
has length 5
c a ϵ ϵ ϵ t
c ϵ ϵ ϵ t t
missing the 'a'

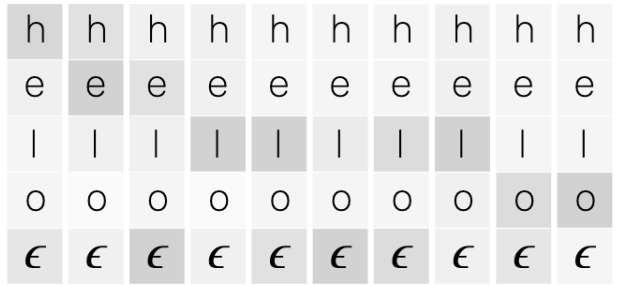
CTC Loss Function – illustrated



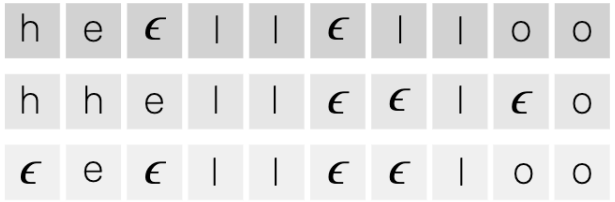
We start with an input sequence, like a spectrogram of audio.



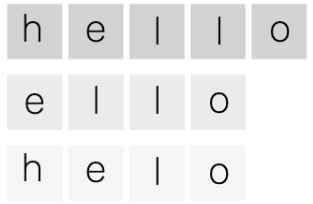
The input is fed into an RNN, for example.



The network gives $p_t(a \mid X)$, a distribution over the outputs {h, e, l, o, ϵ} for each input step.



With the per time-step output distribution, we compute the probability of different sequences



By marginalizing over alignments, we get a distribution over outputs.

CTC Loss Function

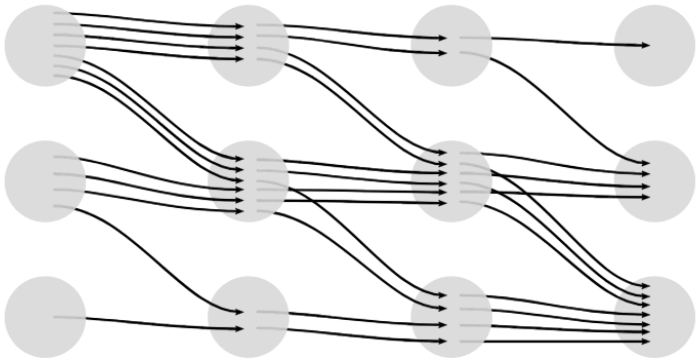
To be precise, the CTC objective for a single (X, Y) pair is:

$$p(Y \mid X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p_t(a_t \mid X)$$

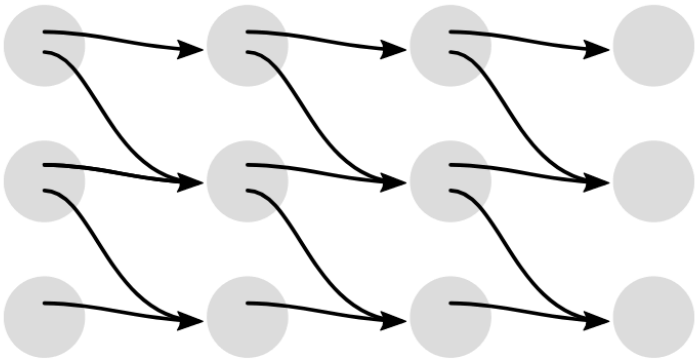
The CTC conditional **probability** **marginalizes** over the set of valid alignments computing the **probability** for a single alignment step-by-step.

Marginalize over all Alignments?

- Valid alignments
  - Retain the original sequence of tokens
  - Optionally insert epsilons between tokens
- Enumerating all alignments is way too expensive → DP!

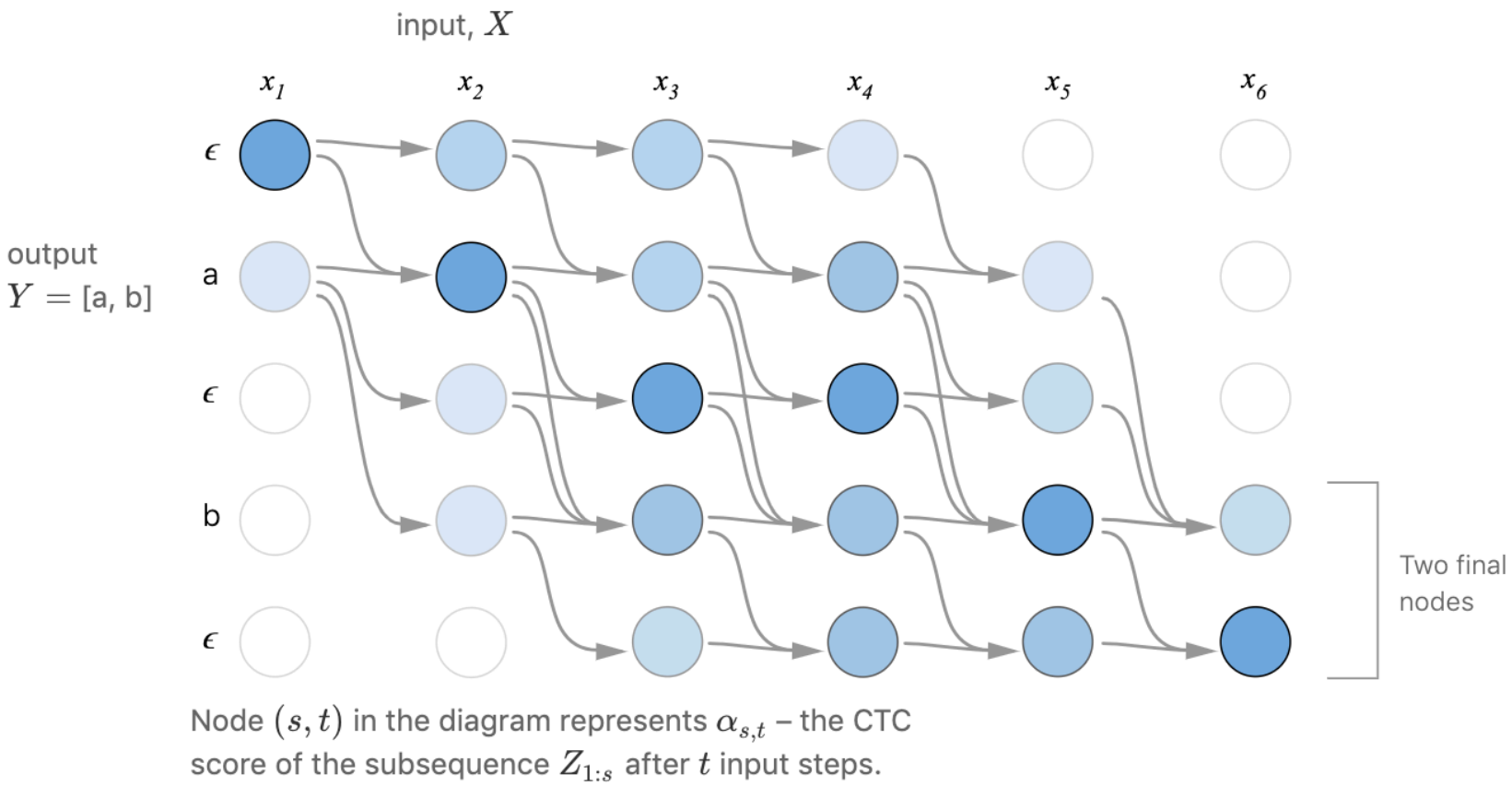


Summing over all alignments can be very expensive.



Dynamic programming merges alignments, so it's much faster.

CTC Alignment Graph



Node (s, t) in the diagram represents $\alpha_{s,t}$ – the CTC score of the subsequence $Z_{1:s}$ after t input steps.

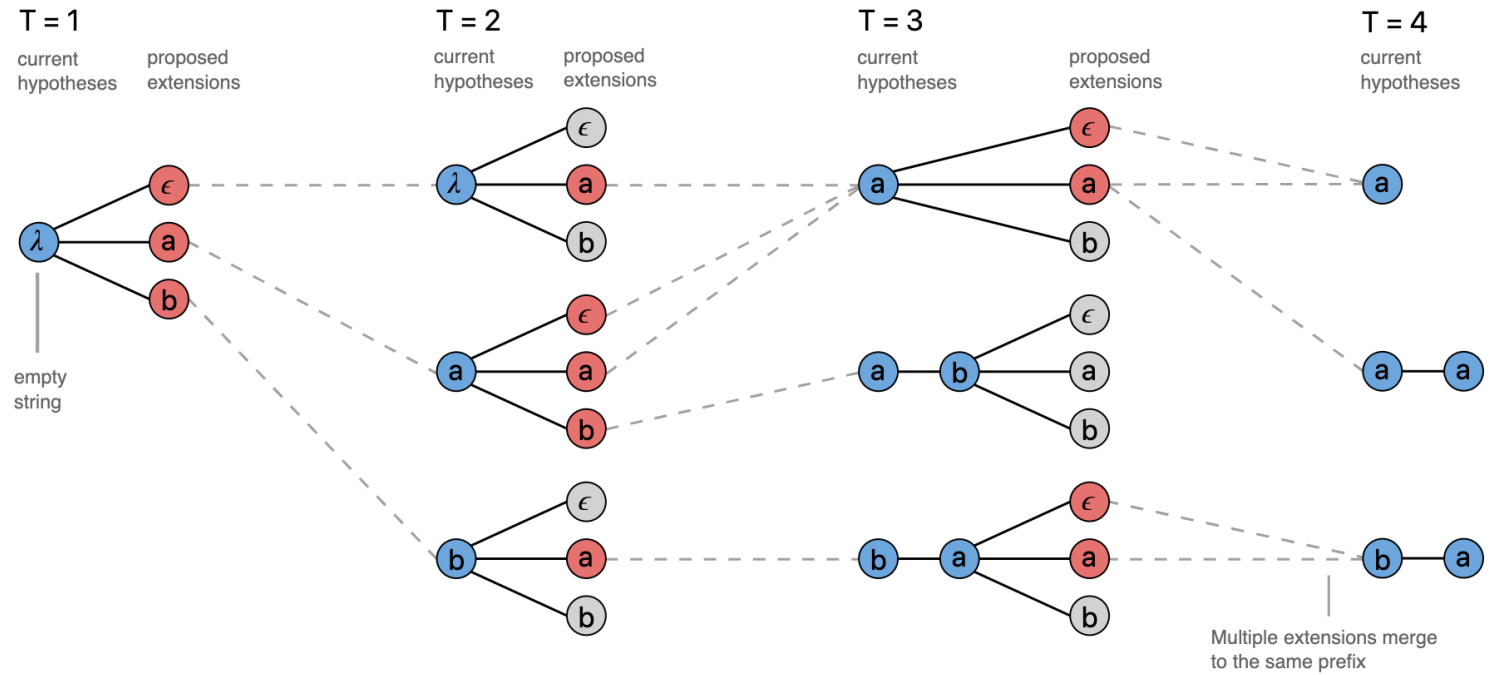
CTC Training

- Loss can be analytically computed → Gradient!
- Maximum Likelihood estimate, including sequence info

$$\sum_{(X,Y)\in\mathcal{D}} -\log\ p(Y \mid X) \quad \rightarrow \text{min.}$$

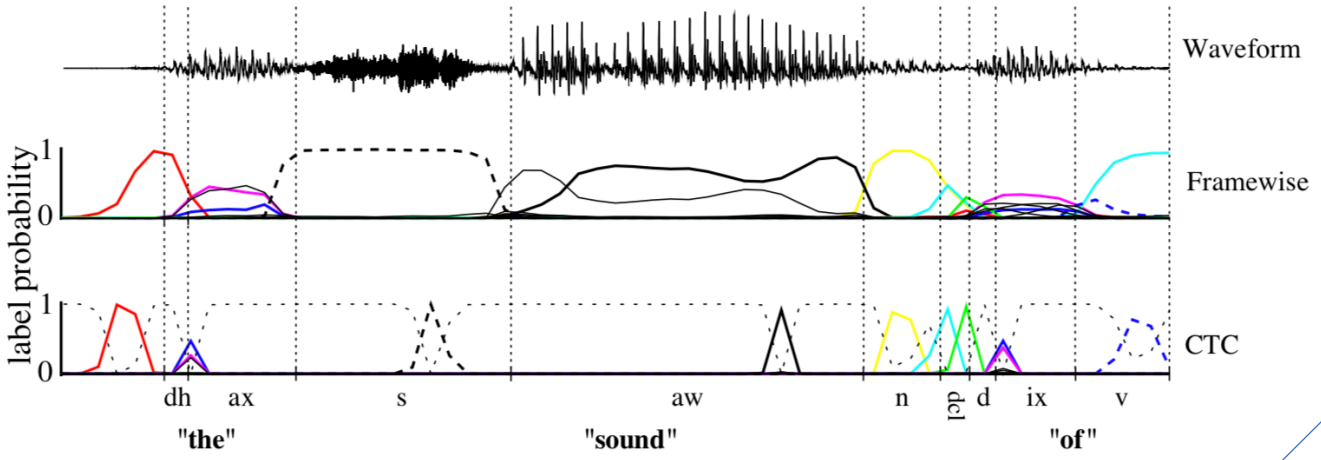
CTC Inference

- How to handle epsilons and repeats?
  - “naive“ way (regexp)
  - Beam search



The CTC beam search algorithm with an output alphabet $\{\epsilon, a, b\}$ and a beam size of three.

CTC for Speech Recognition



„spikey“ activations for non-blanks

Table 1. **Label Error Rate (LER) on TIMIT**. CTC and hybrid results are means over 5 runs, ± standard error. All differences were significant ($p < 0.01$), except between weighted error BLSTM/HMM and CTC (best path).

System	LER
Context-independent HMM	38.85 %
Context-dependent HMM	35.21 %
BLSTM/HMM	33.84 ± 0.06 %
Weighted error BLSTM/HMM	31.57 ± 0.06 %
CTC (best path)	31.47 ± 0.21 %
CTC (prefix search)	30.51 ± 0.19 %

„traditional“ HMM-GMM

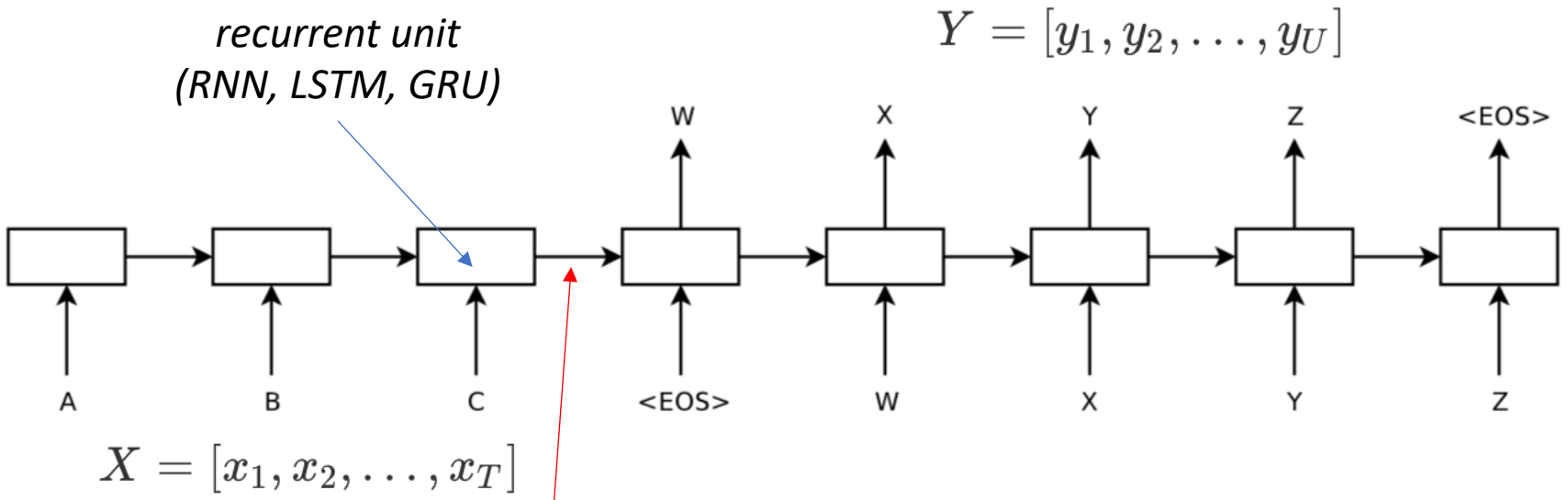
„hybrid“ HMM-DNN

First „end-to-end“ in 2006!

CTC References

- „Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks.“ A. Graves et al., ICML2006
- https://distill.pub/2017/ctc/
- TensorFlow
  - https://www.tensorflow.org/api_docs/python/tf/nn/ctc_loss
  - https://www.tensorflow.org/api_docs/python/tf/nn/ctc_beam_search_decoder
- cuDNN
  - https://docs.nvidia.com/deeplearning/sdk/cudnn-developer-guide/index.html#cudnnCTCLossAlgo_t

Encoder-Decoder Networks



- Encoder builds up history vector by consuming input
- Decoder produces output based on history and previous outputs
- Hard to train (gradients…), successful for MT: I. Sitskever et al.

Encoder-Decoder for Machine Translation

- „Sequence to Sequence Learning with Neural Networks.“ I. Sutskever, O. Vinyals and Quoc V. Le, NIPS2014
  - LSTM as recurrent unit
  - Reverse input sentences
  - Training for „about 10 days“ on a 8-GPU machine (K40?)

Attention based!
(more in a second)

Method	test BLEU score (ntst14)
Bahdanau et al. [2]	28.45
Baseline System [29]	33.30
Single forward LSTM, beam size 12	26.17
Single reversed LSTM, beam size 12	30.59
Ensemble of 5 reversed LSTMs, beam size 1	33.00
Ensemble of 2 reversed LSTMs, beam size 12	33.27
Ensemble of 5 reversed LSTMs, beam size 2	34.50
Ensemble of 5 reversed LSTMs, beam size 12	**34.81**

Table 1: The performance of the LSTM on WMT'14 English to French test set (ntst14). Note that an ensemble of 5 LSTMs with a beam of size 2 is cheaper than of a single LSTM with a beam of size 12.

Encoder-Decoder Networks

images: positional!

- (more or less) obvious shortcoming:
  How should a single vector encode temporal order or collocation?

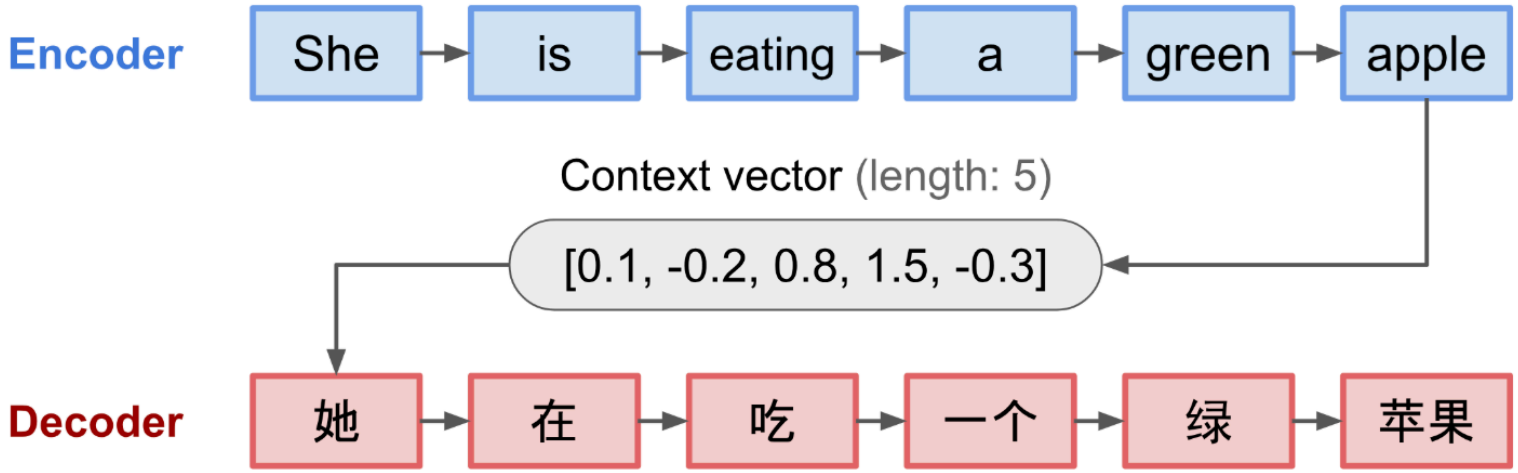
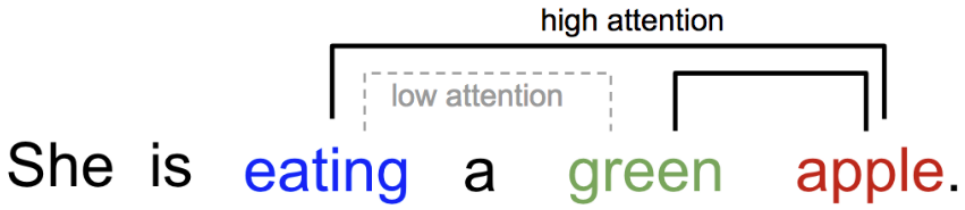


Fig. 3. The encoder-decoder model, translating the sentence "she is eating a green apple" to Chinese. The visualization of both encoder and decoder is unrolled in time.

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

Attention!



Words "attend" to each other to varying degree.

Fig. 7. "A woman is throwing a frisbee in a park." (Image source: Fig. 6(b) in Xu et al. 2015)

Additive Attention by Bahdanau

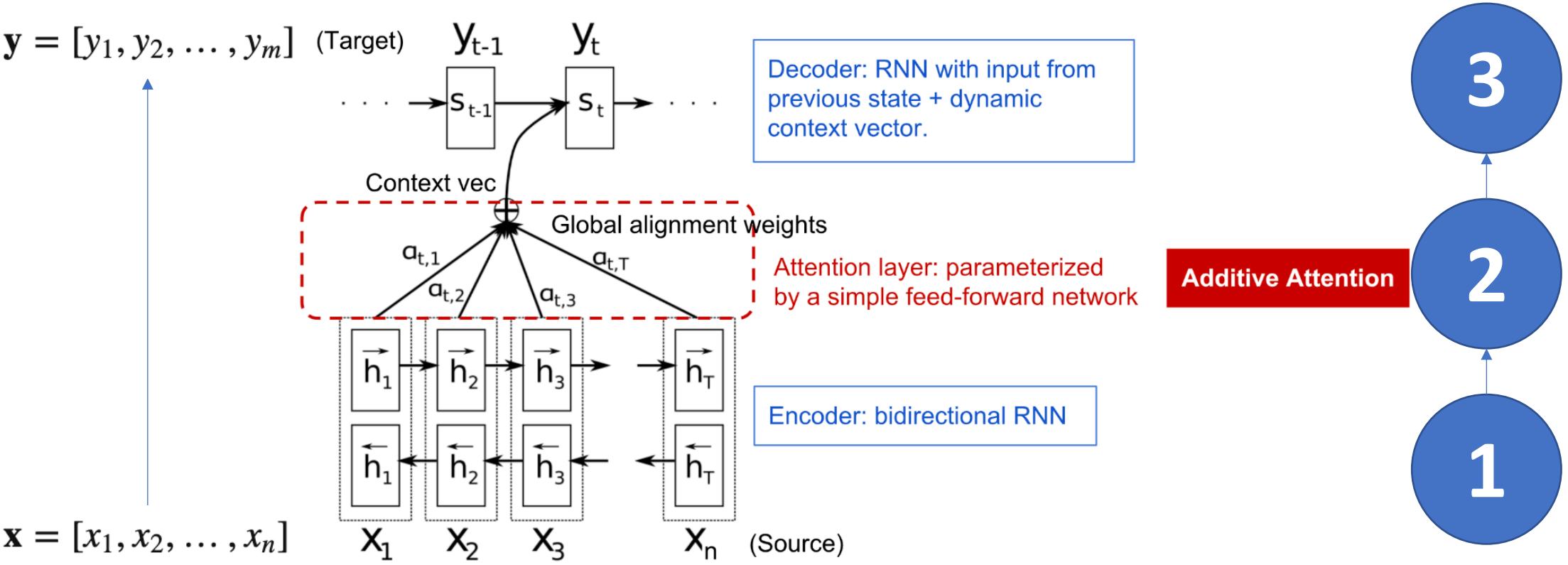


Fig. 4. The encoder-decoder model with additive attention mechanism in Bahdanau et al., 2015.

Attention formalized

$$\boldsymbol{h}_i = [\overrightarrow{\boldsymbol{h}}_i^\top ; \overleftarrow{\boldsymbol{h}}_i^\top]^\top, i = 1, \ldots, n$$

combined History (B-LSTM)

$$\mathbf{c}_t = \sum_{i=1}^{n} \alpha_{t,i} \boldsymbol{h}_i$$

context vector for output y_t

feed-forward net!

$$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$$

$$\alpha_{t,i} = \text{align}(y_t, x_i)$$

alignment score: how well do y_t and x_i align?

$$= \frac{\exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i))}{\sum_{i'=1}^{n} \exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_{i'}))}$$

softmax: how does x_i contribute to Y at time t-1?

Alignment scores visualized

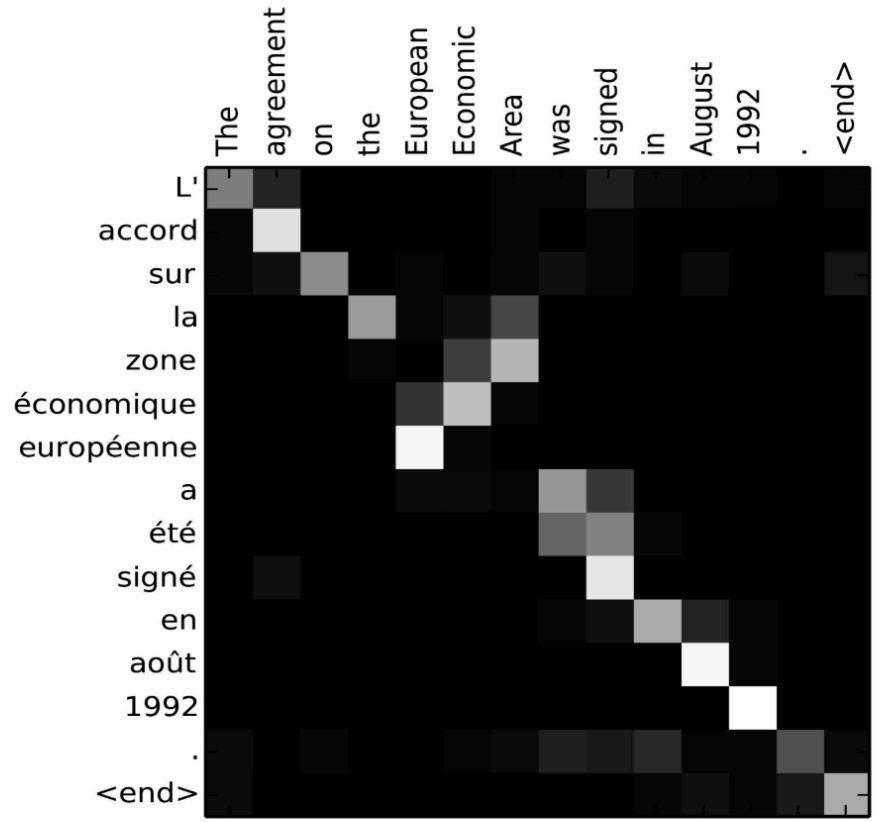


Fig. 5. Alignment matrix of "L'accord sur l'Espace économique européen a été signé en août 1992" (French) and its English translation "The agreement on the European Economic Area was signed in August 1992". (Image source: Fig 3 in Bahdanau et al., 2015)

Attention for MT

- „Neural Machine Translation by Jointly Learning to Align and Translate.“, D. Bahdanau, KH Cho and Y Bengio, ICLR2015

Model	All	No UNK°
RNNencdec-30	13.93	24.19
RNNsearch-30	21.50	31.44
RNNencdec-50	17.82	26.71
RNNsearch-50	26.75	34.16
RNNsearch-50⋆	28.45	36.15
Moses	33.30	35.63

Table 1: BLEU scores of the trained models computed on the test set. The second and third columns show respectively the scores on all the sentences and, on the sentences without any unknown word in themselves and in the reference translations. Note that RNNsearch-50⋆ was trained much longer until the performance on the development set stopped improving. (○) We disallowed the models to generate [UNK] tokens when only the sentences having no unknown words were evaluated (last column).

Attention Visualized

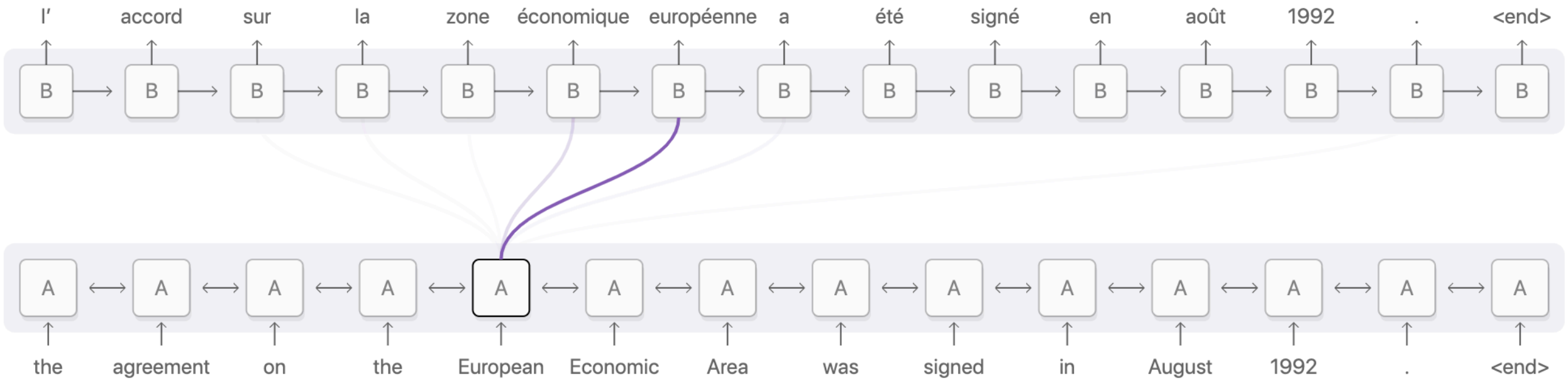


Diagram derived from Fig. 3 of Bahdanau, *et al.* 2014

https://distill.pub/2016/augmented-rnns/#attentional-interfaces

Attention Visualized

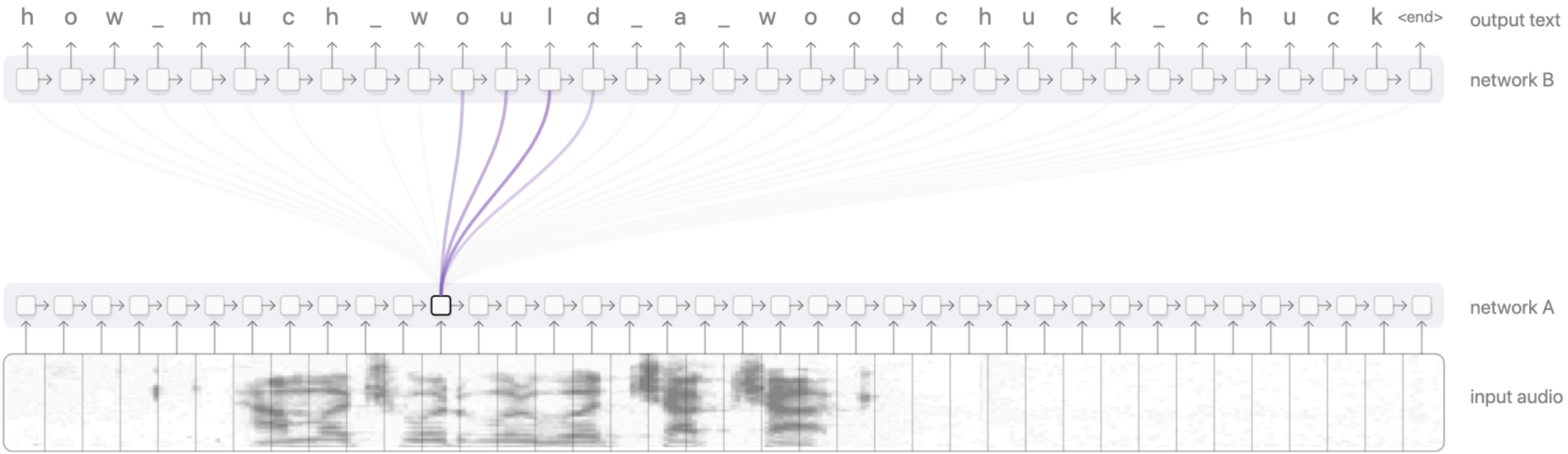


Figure derived from Chan, *et al.* 2015

https://distill.pub/2016/augmented-rnns/#attentional-interfaces

Attention Mechanisms

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

(*) Referred to as "concat" in Luong, et al., 2015 and as "additive attention" in Vaswani, et al., 2017.

(^) It adds a scaling factor $1/\sqrt{n}$, motivated by the concern when the input is large, the softmax function may have an extremely small gradient, hard for efficient learning.

Attention Categories

Name	Definition	Citation
Self-Attention(&)	Relating different positions of the same input sequence. Theoretically the self-attention can adopt any score functions above, but just replace the target sequence with the same input sequence.	Cheng2016
Global/Soft	Attending to the entire input state space.	Xu2015
Local/Hard	Attending to the part of input state space; i.e. a patch of the input image.	Xu2015; Luong2015

(&) Also, referred to as “intra-attention” in Cheng et al., 2016 and some other papers.

Self-Attention (Inter-Attention)

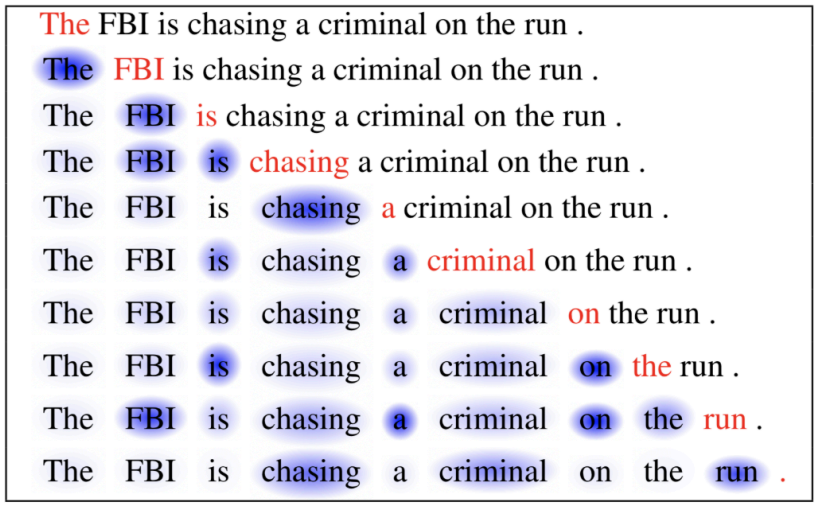


Fig. 6. The current word is in red and the size of the blue shade indicates the activation level. (Image source: Cheng et al., 2016)

Fig. 7. "A woman is throwing a frisbee in a park." (Image source: Fig. 6(b) in Xu et al. 2015)

Global vs. Local Attention

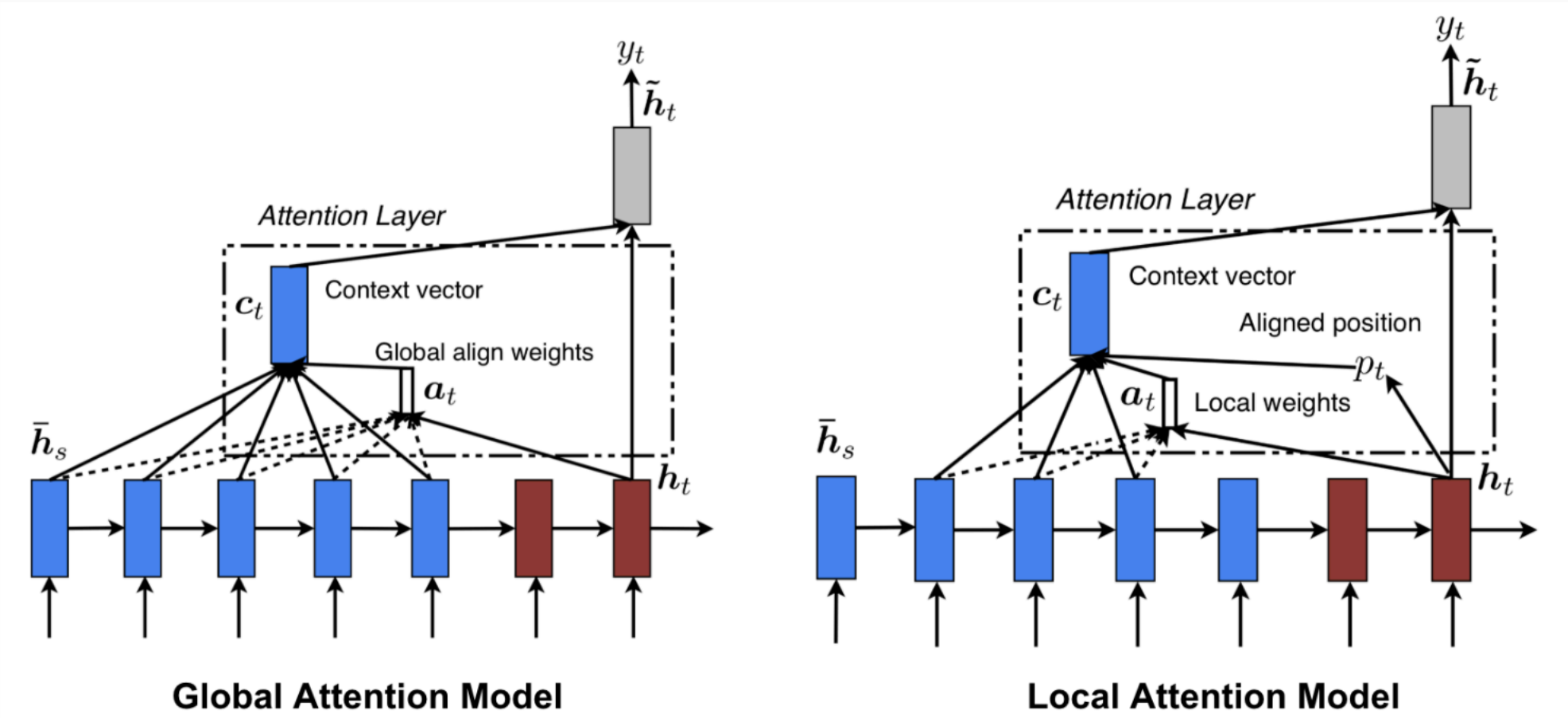
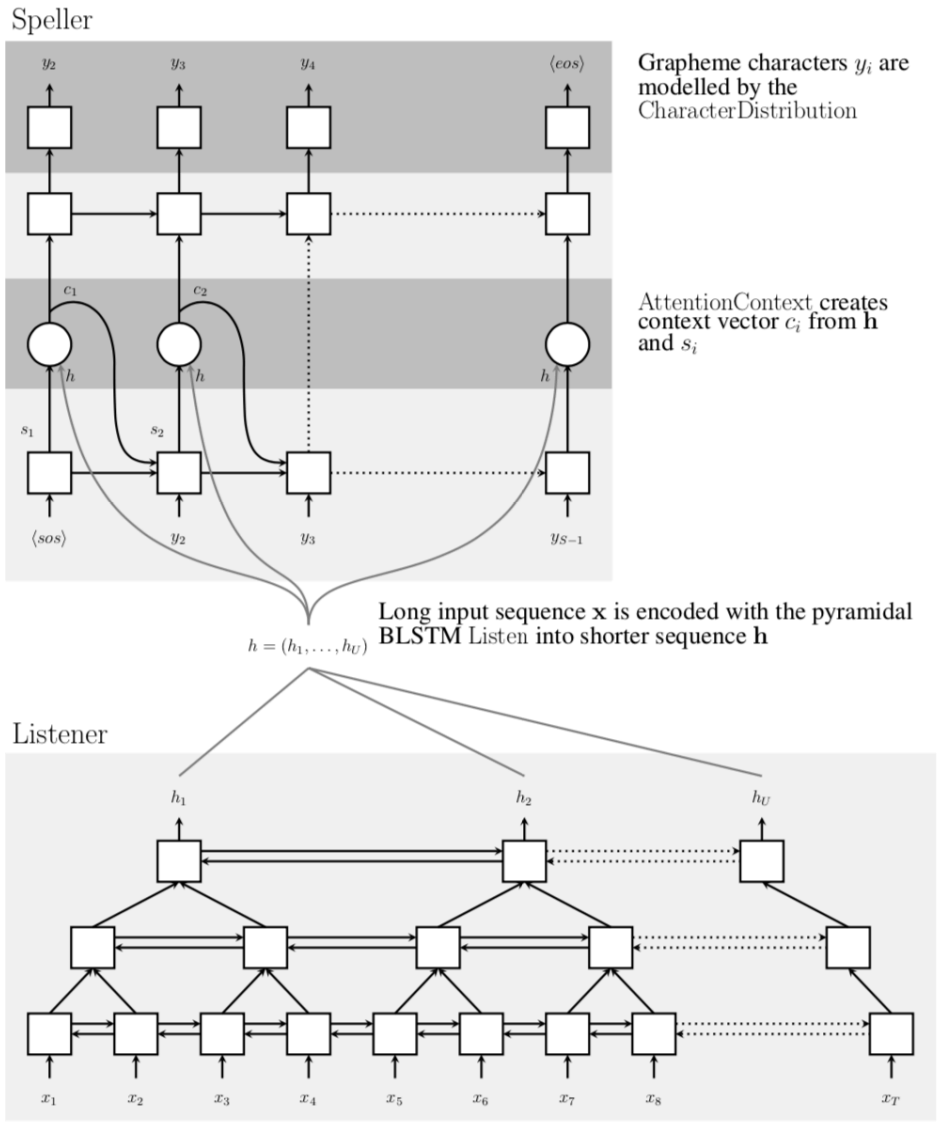
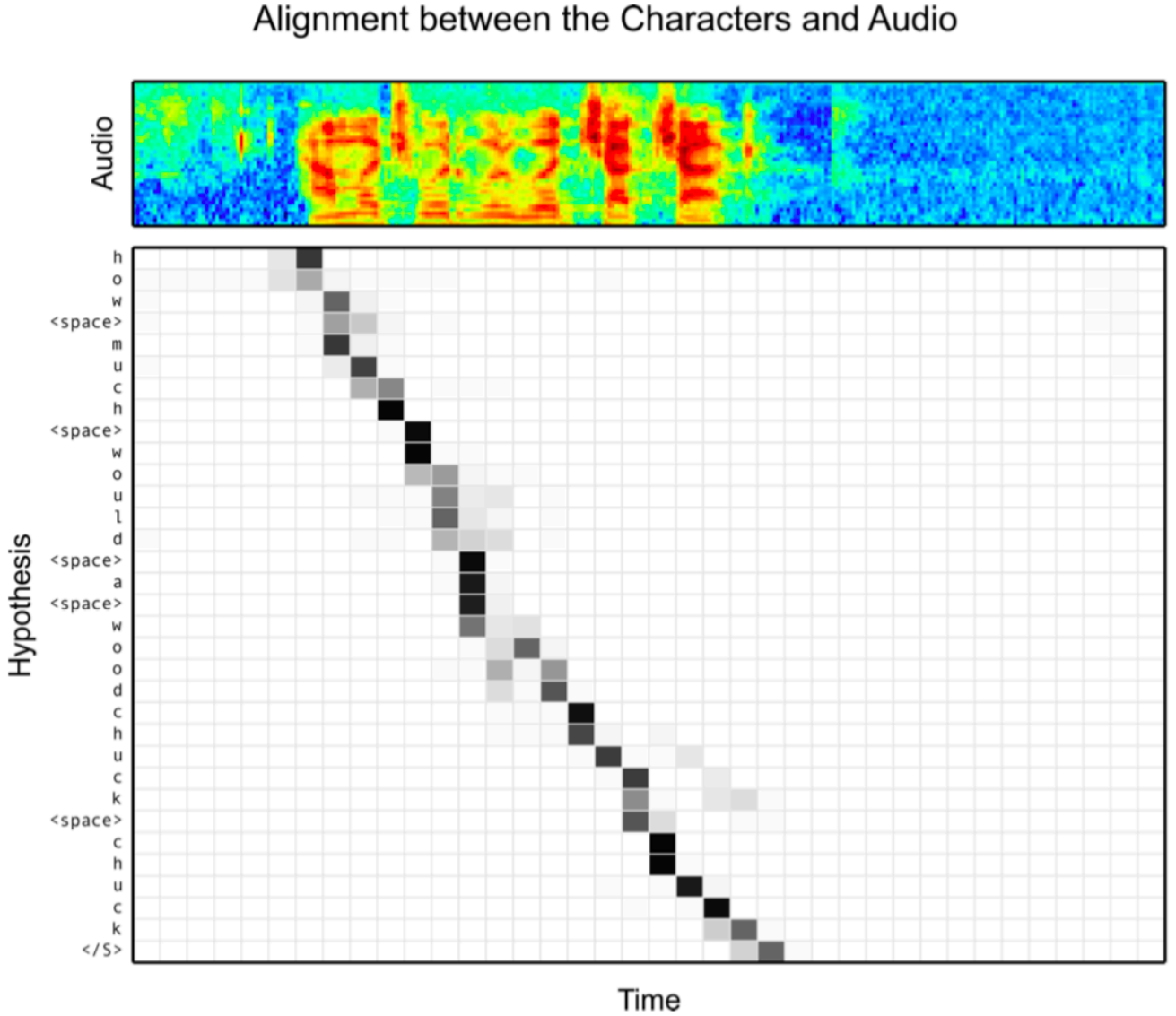


Fig. 8. Global vs local attention (Image source: Fig 2 & 3 in Luong, et al., 2015)

Listen, Attend and Spell (Chan et al. ICASSP16)





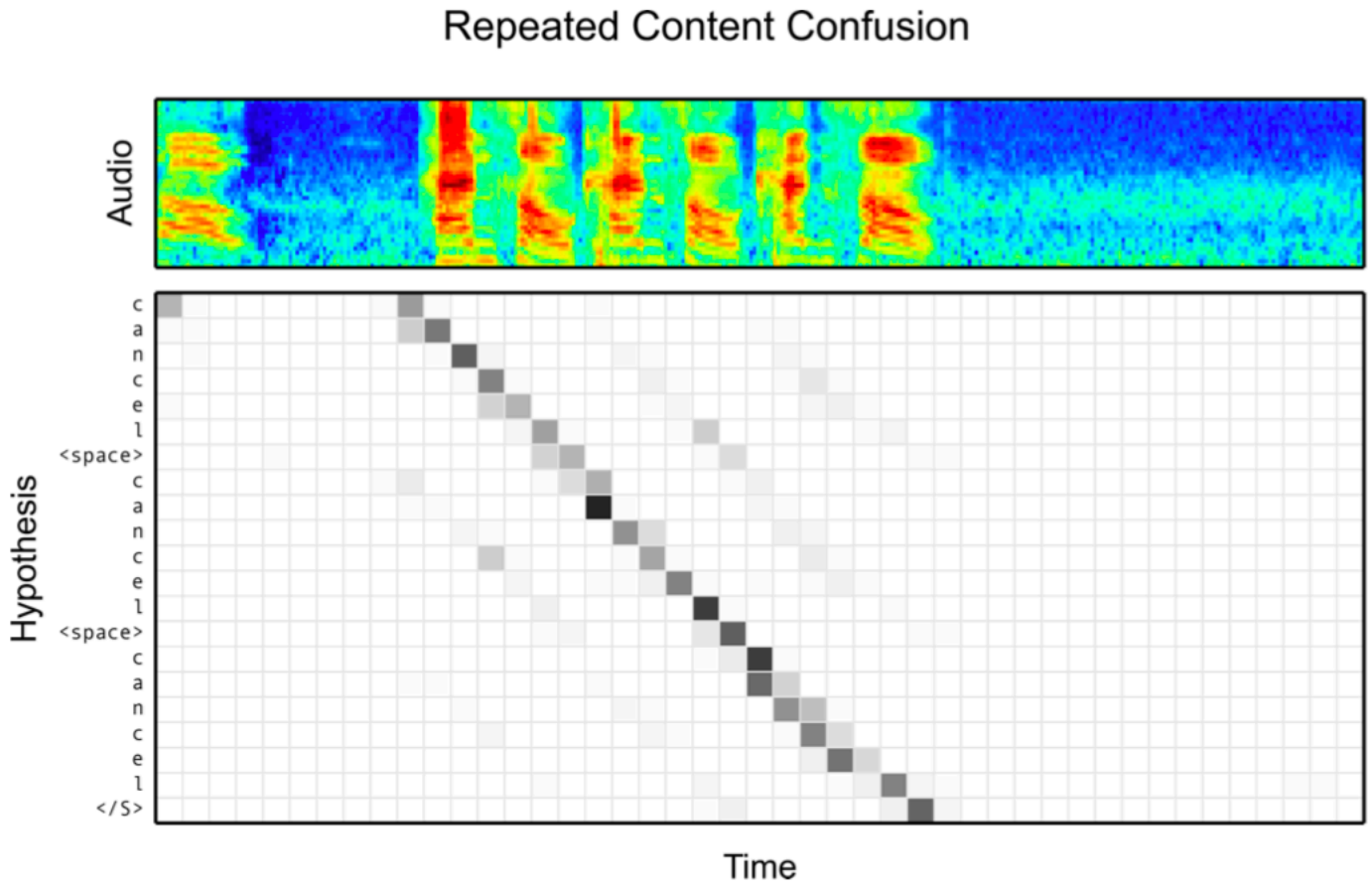
Listen, Attend and Spell (cont‘d)

Table 1: WER comparison on the clean and noisy Google voice search task. The CLDNN-HMM system is the state-of-the-art system, the Listen, Attend and Spell (LAS) models are decoded with a beam size of 32. Language Model (LM) rescoring was applied to our beams, and a sampling trick was applied to bridge the gap between training and inference.

„traditional“ ASR

Model	Clean WER	Noisy WER
CLDNN-HMM [20]	8.0	8.9
LAS	16.2	19.0
LAS + LM Rescoring	12.6	14.7
LAS + Sampling	14.1	16.5
LAS + Sampling + LM Rescoring	10.3	12.0

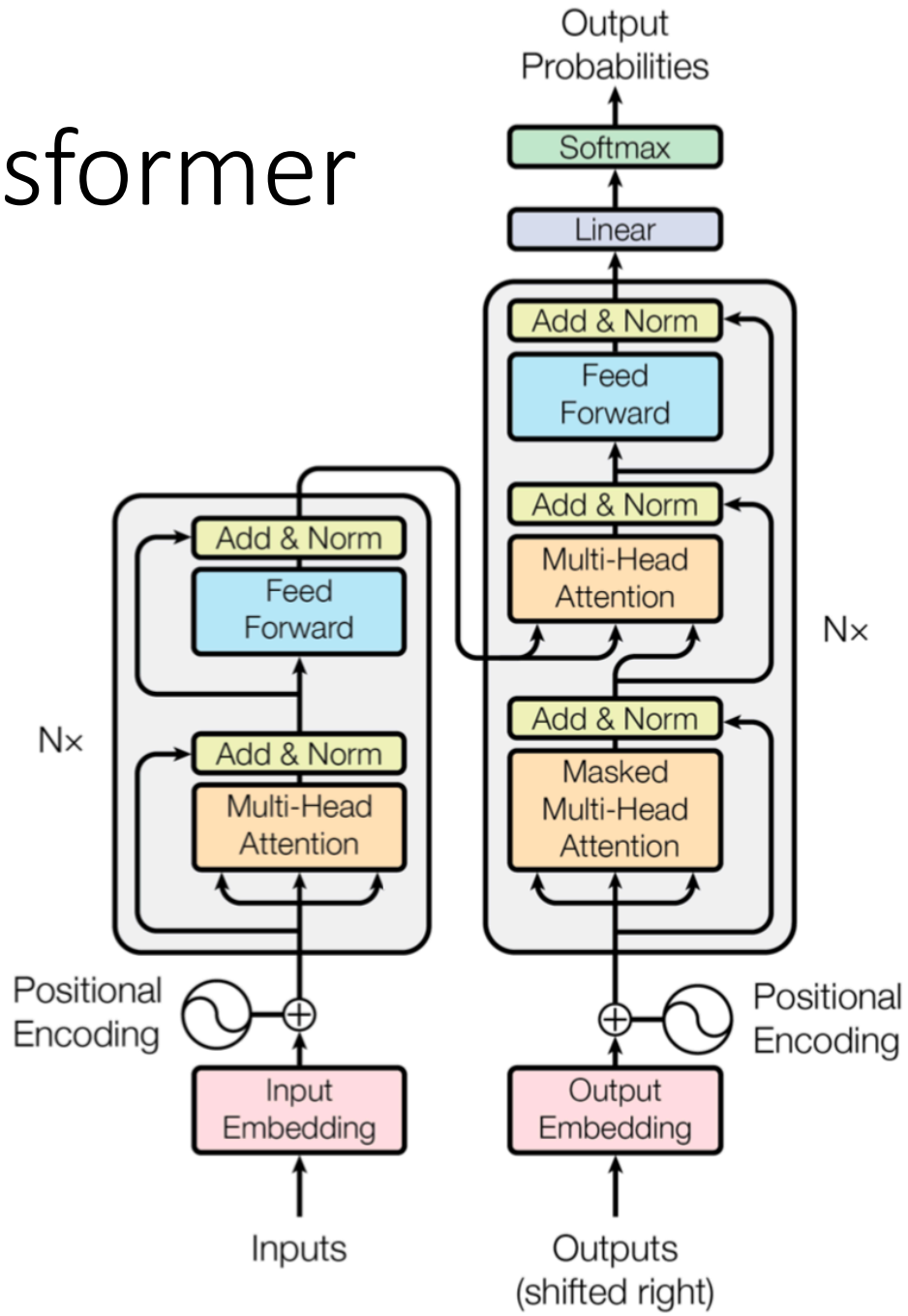
Listen, Attend and Spell (cont‘d)

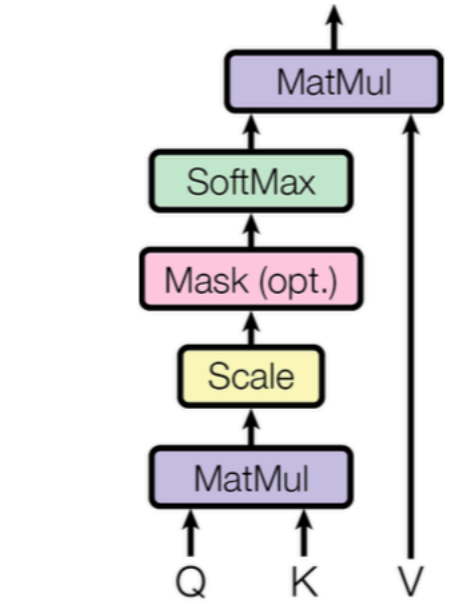


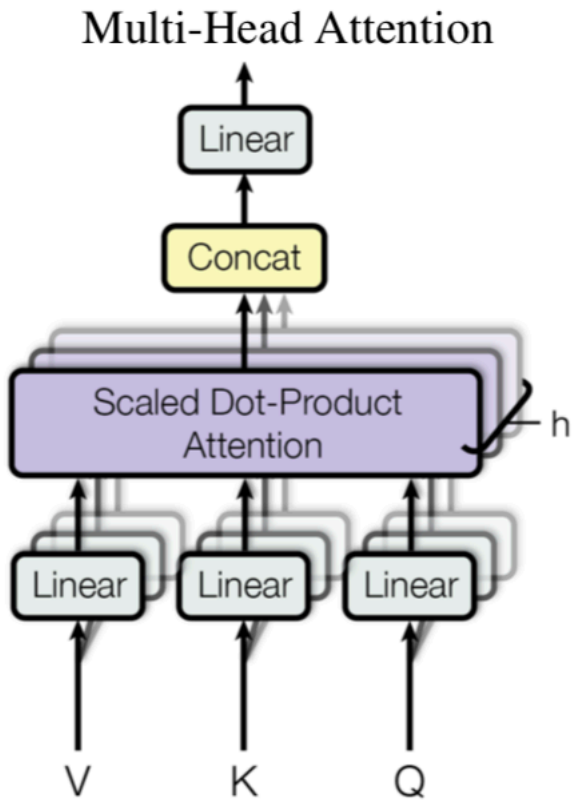
Attention is all you need (Vaswani et al. NIPS17)

- LSTM and (particularly) GRU established as state-of-the-art
- Attention mechanism great way to incorporate context
- …recurrent units require lots of data and are hard to tune.
- Proposed solution:
  - Massive use of attention…
  - …with basic nets/layers?

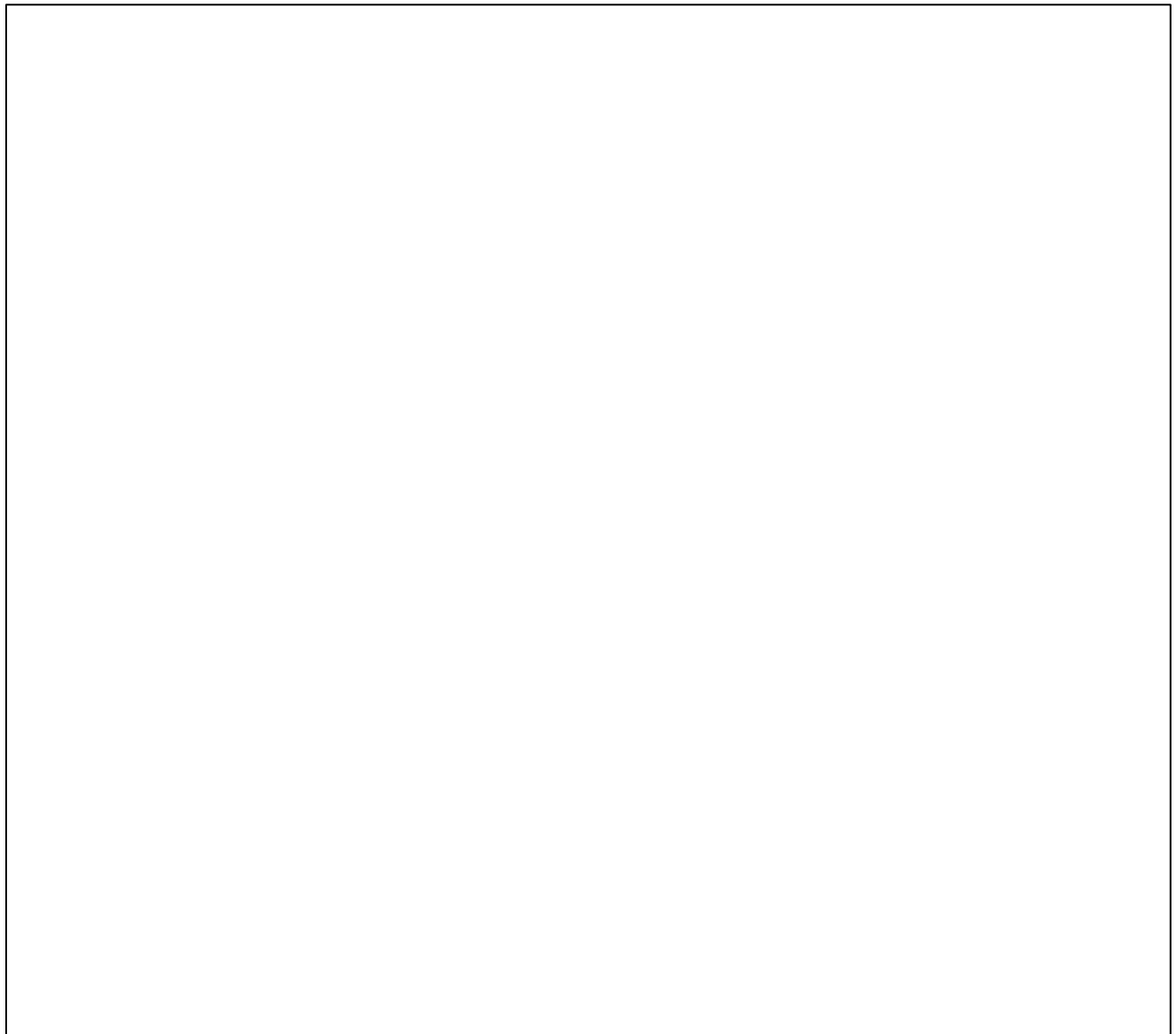
Transformer





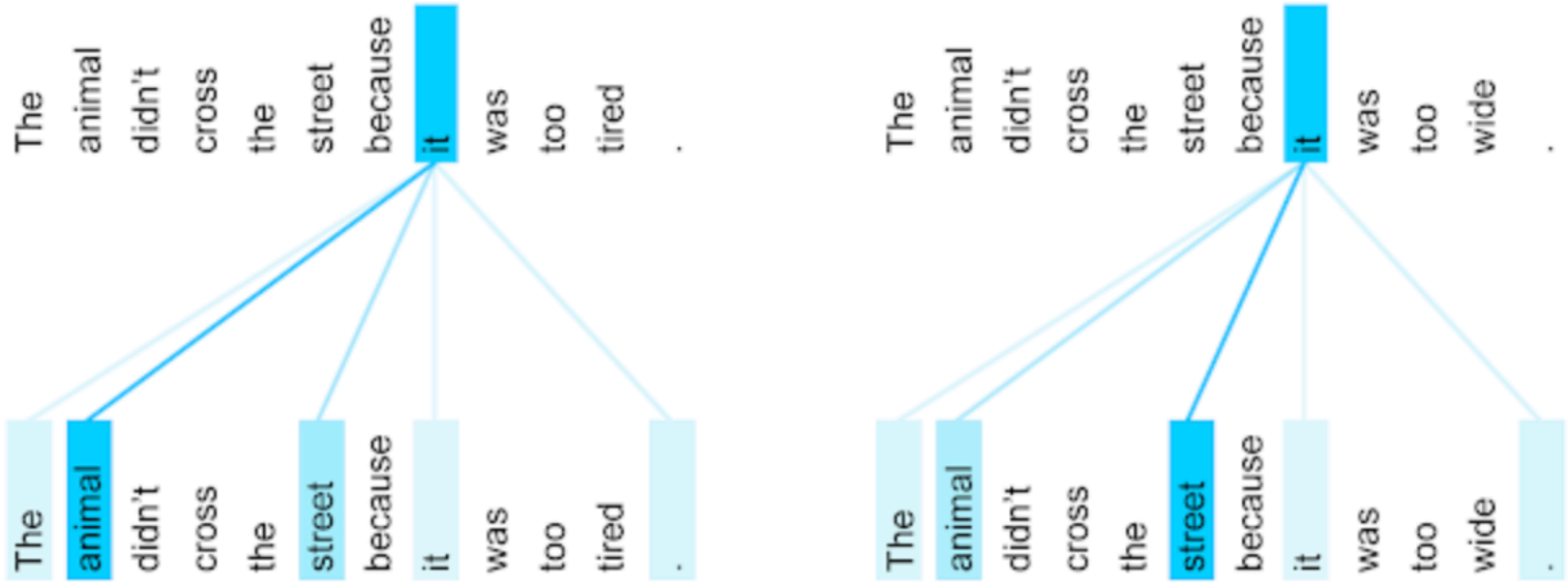


Transformer Visualized



https://ai.googleblog.com/2017/08/

Transformer Visualized



The encoder self-attention distribution for the word “it” from the 5th to the 6th layer of a Transformer trained on English to French translation (one of eight attention heads).

Transformer

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Transformer Generalizes!

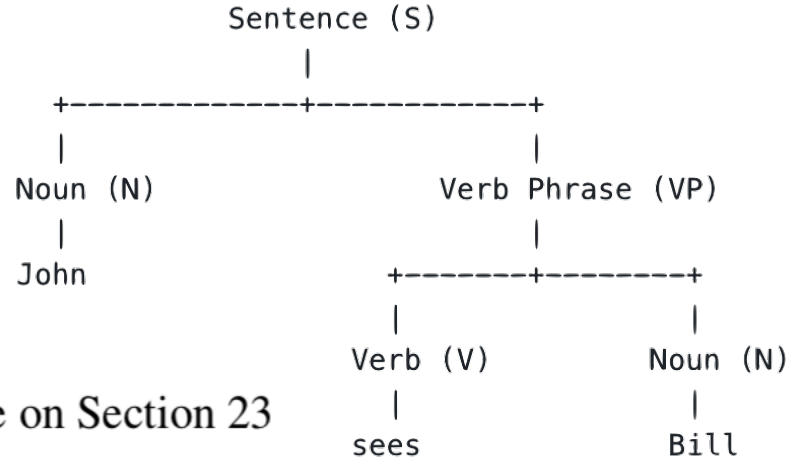


Table 4: The Transformer generalizes well to English constituency parsing (Results are on Section 23 of WSJ)

Parser	Training	WSJ 23 F1
Vinyals & Kaiser el al. (2014) [37]	WSJ only, discriminative	88.3
Petrov et al. (2006) [29]	WSJ only, discriminative	90.4
Zhu et al. (2013) [40]	WSJ only, discriminative	90.4
Dyer et al. (2016) [8]	WSJ only, discriminative	91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [40]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [26]	semi-supervised	92.1
Vinyals & Kaiser el al. (2014) [37]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Luong et al. (2015) [23]	multi-task	93.0
Dyer et al. (2016) [8]	generative	93.3

A Simple Neural Attentive Meta-Learner (Mishra et al., NIPS MetaLearn 2017)

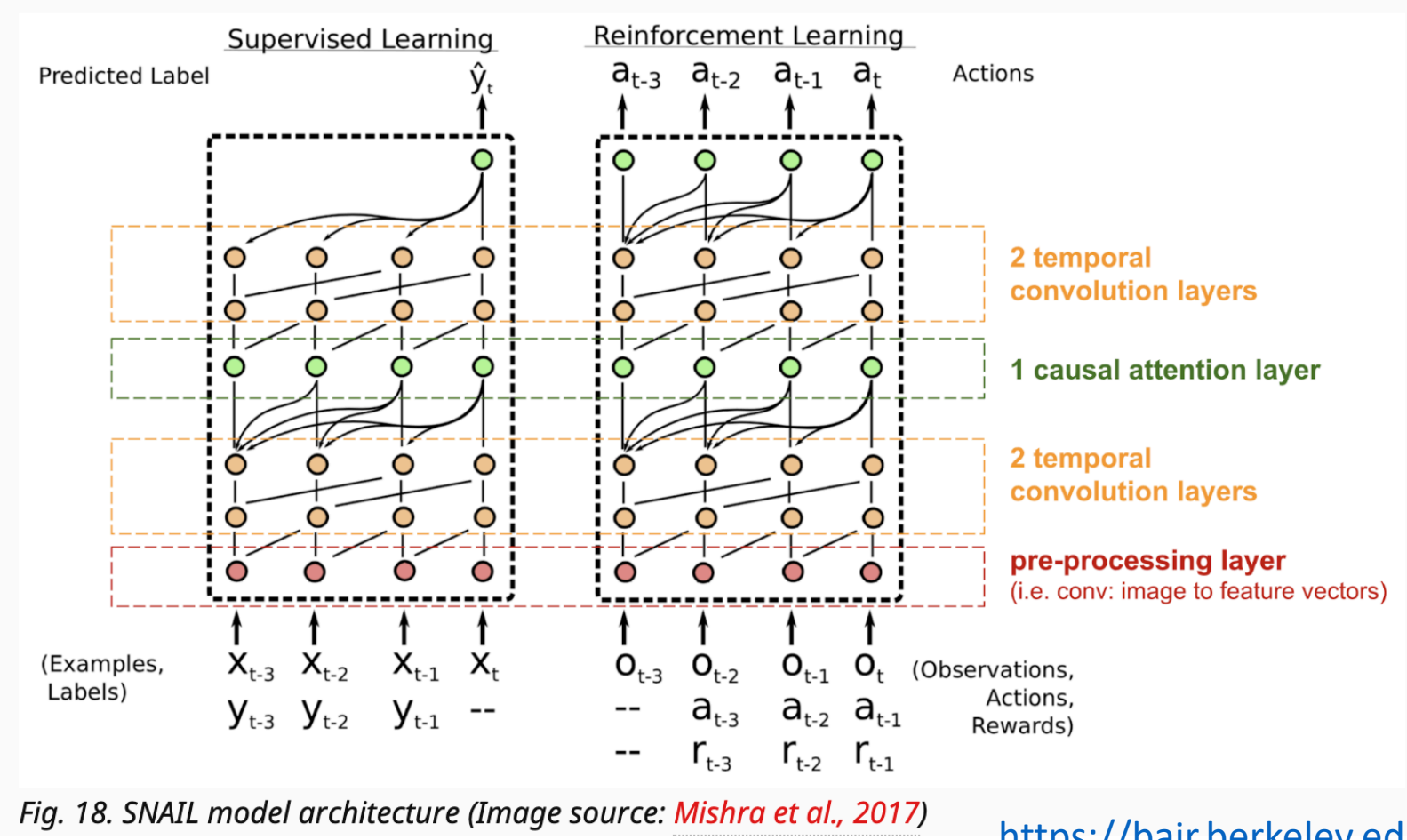


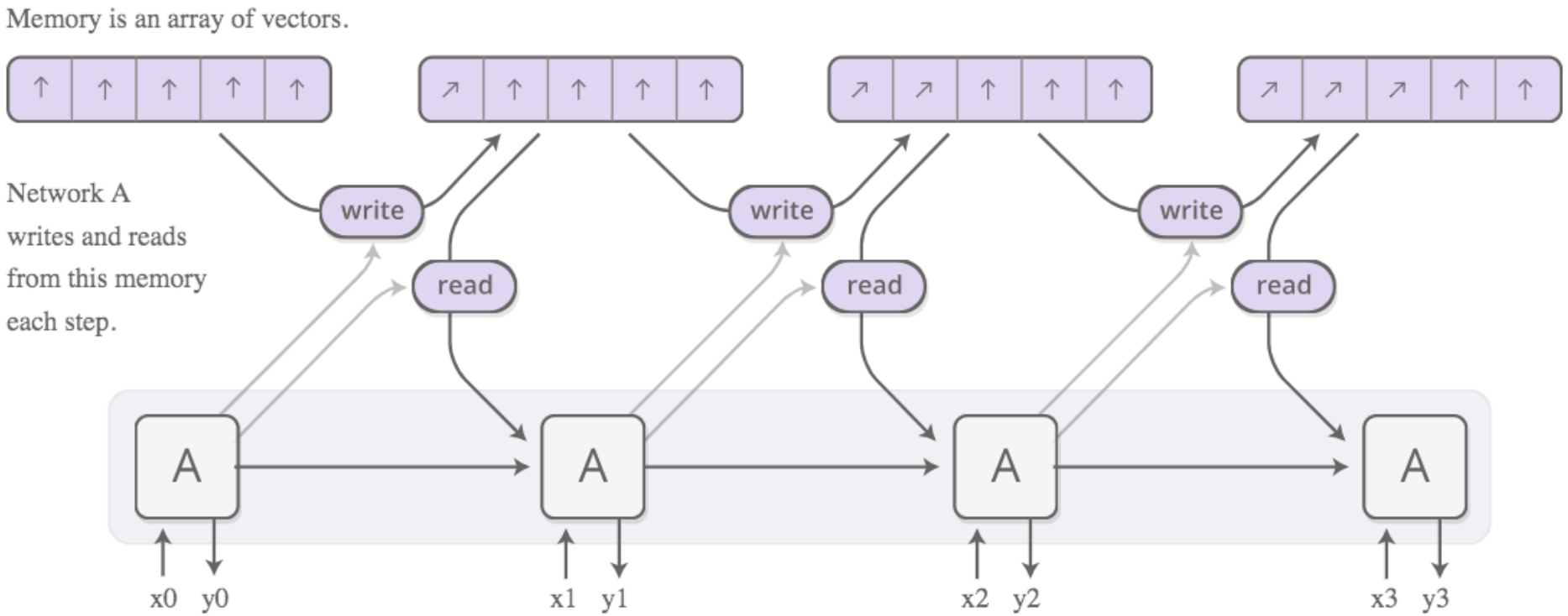
Fig. 18. SNAIL model architecture (Image source: Mishra et al., 2017)

https://bair.berkeley.edu/blog/2017/07/18/learning-to-learn/

Outlook: Other Crazy Things…

https://distill.pub/2016/augmented-rnns/

Neural Turing Machines



https://distill.pub/2016/augmented-rnns/#neural-turing-machines

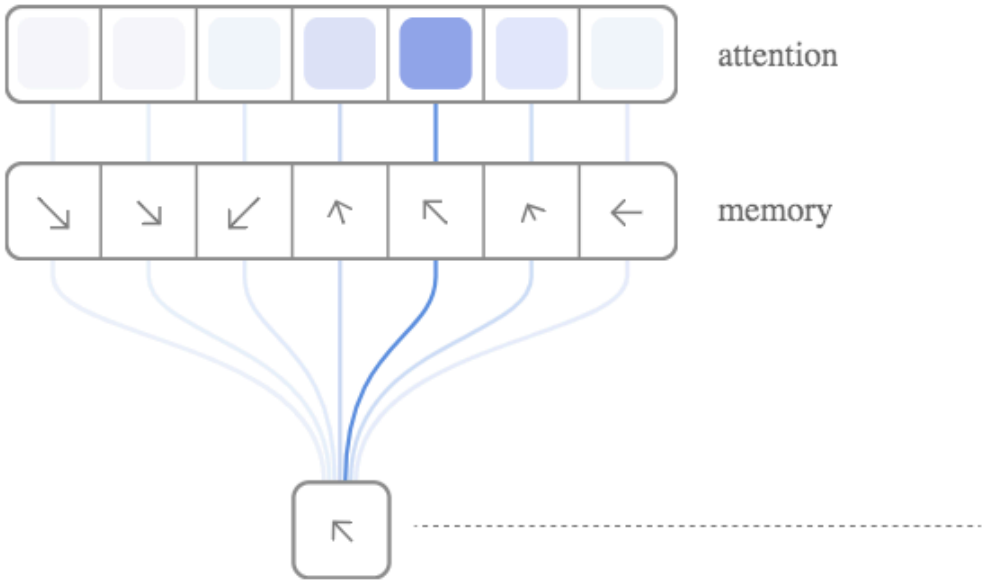
Graves et al. „Neural Turing Machines“, https://arxiv.org/abs/1410.5401

Neural Turing Machines

- Vectors are “natural language” of neural nets
- How to read/write from memory?
- How to differentiate?

In every step, read and write **everywhere!**

NTM: Read

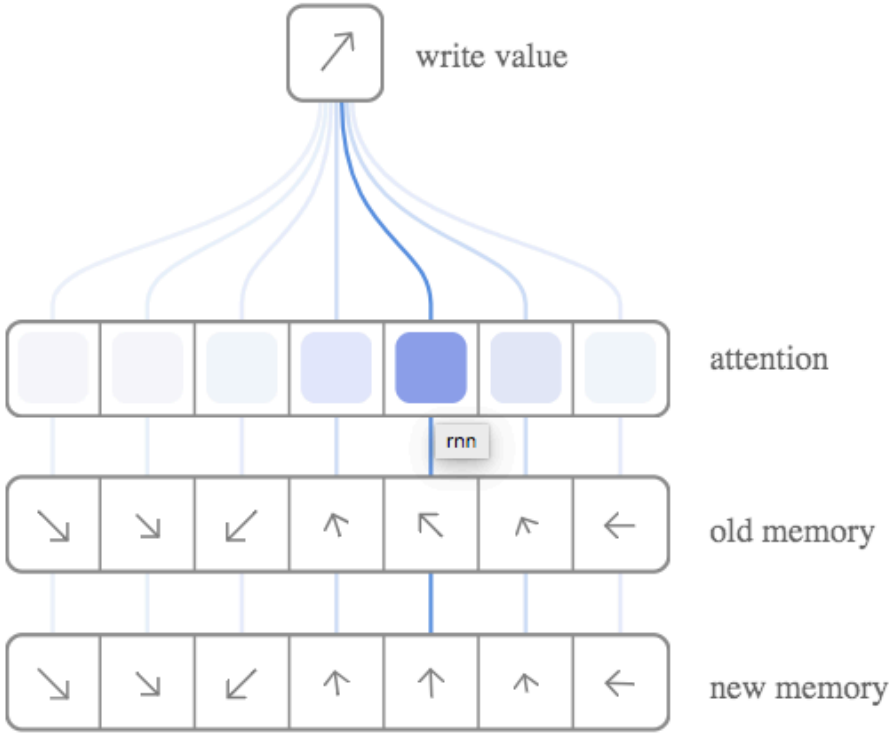


The RNN gives an attention distribution which describe how we spread out the amount we care about different memory positions.

The read result is a weighted sum.

$$r \leftarrow \sum_i a_i M_i$$

NTM: Write

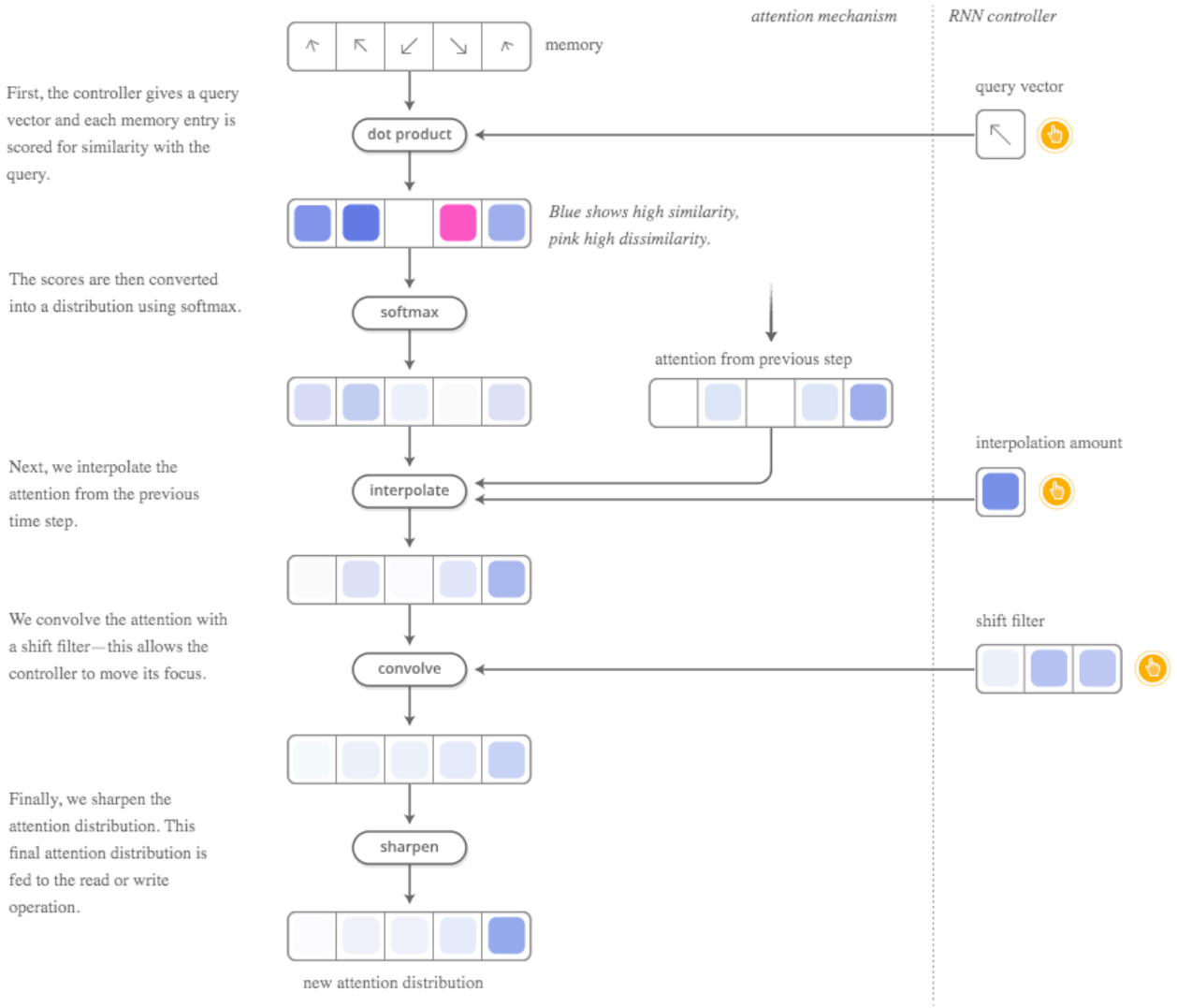


Instead of writing to one location, we write everywhere, just to different extents.

The RNN gives an attention distribution, describing how much we should change each memory position towards the write value.

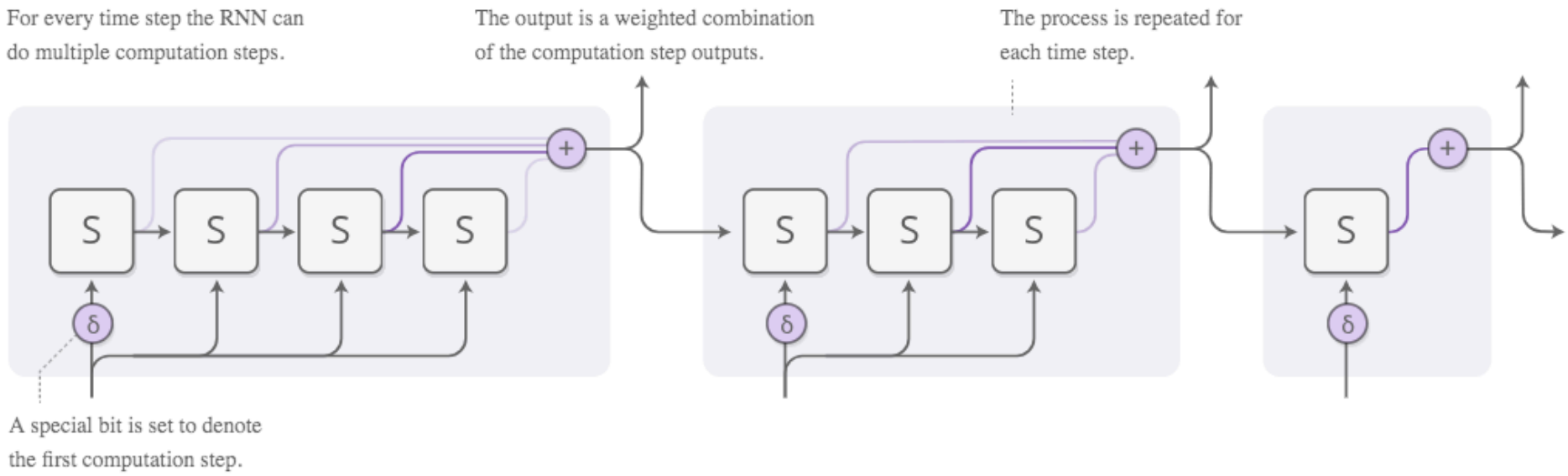
$$M_i \leftarrow a_i w + (1 - a_i) M_i$$

NTM: Inference

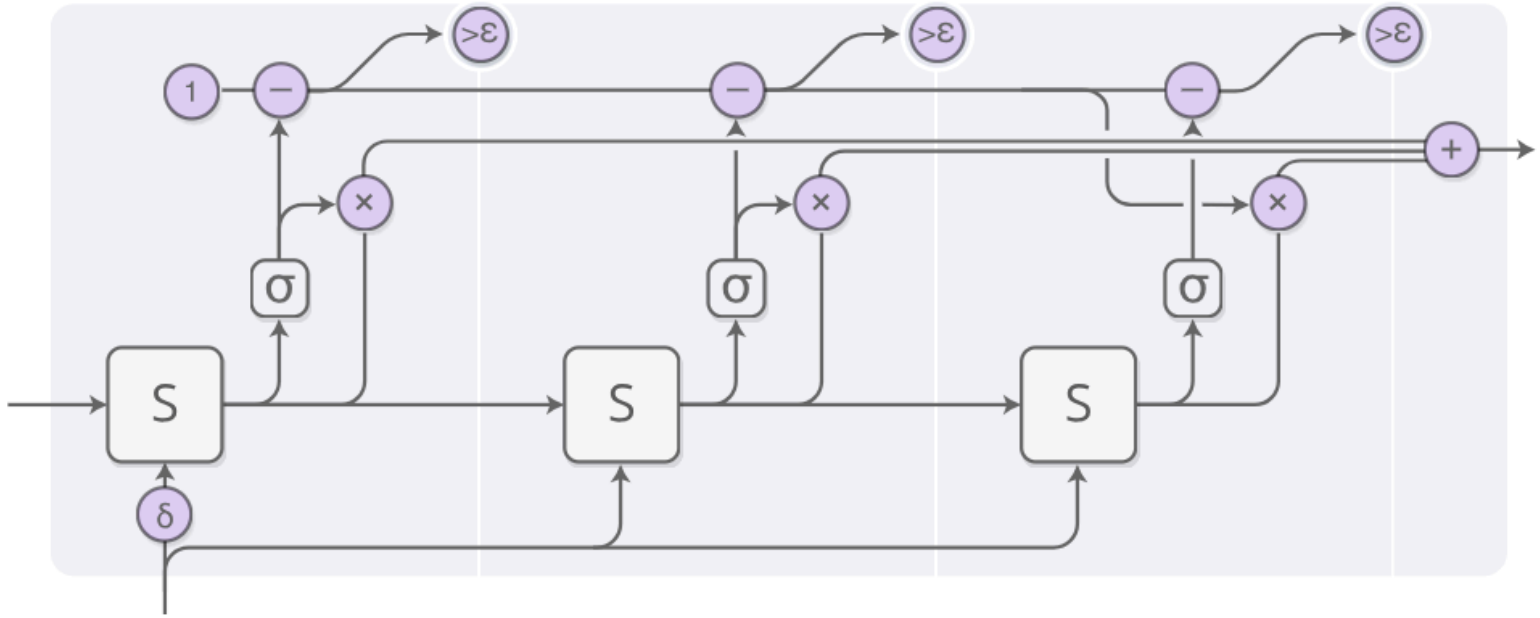


Adaptive Compute Time

- Allow RNN to execute variable amounts of computation for each timestep?
- How many timesteps? …attention!

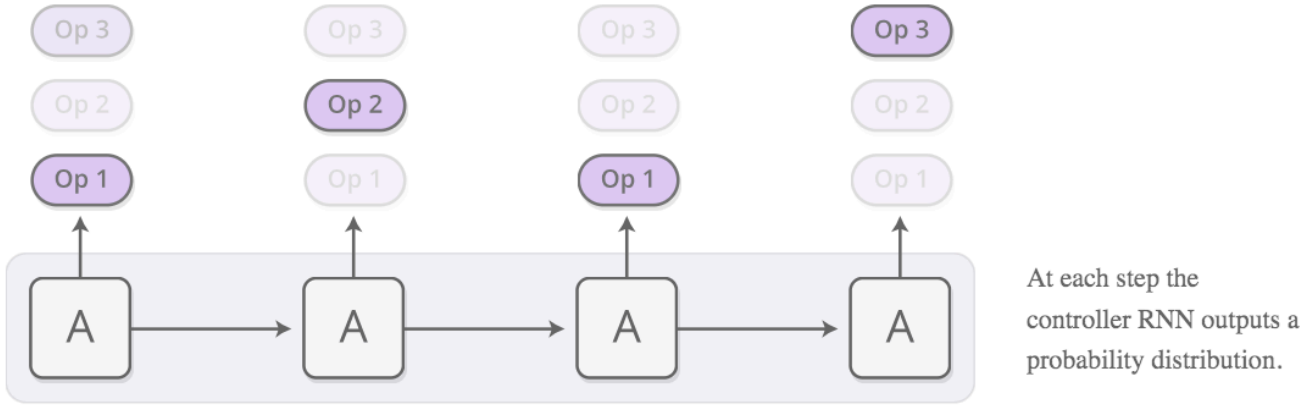


Adaptive Compute Time (cont‘d)



Neural Programmer: Inducing Latent Programs with Gradient Descent (Neelakantan et al. ICLR2015)

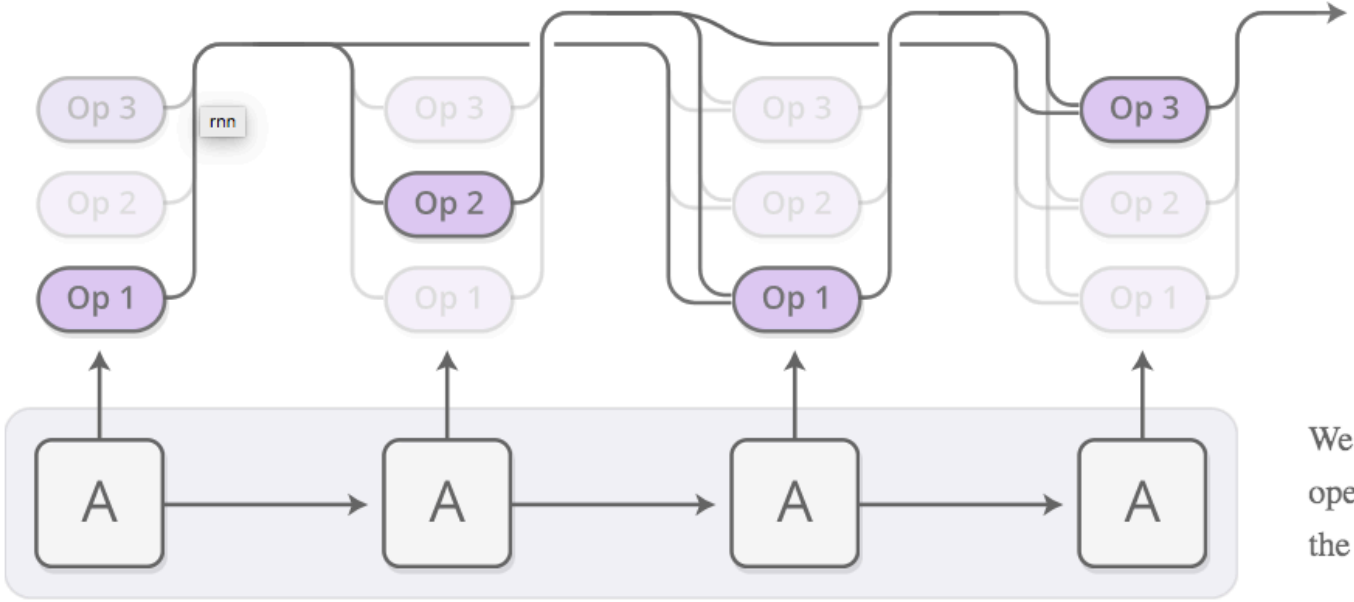
- How about modeling actions/operations?
- Like arithmetic, loops, etc.?



At each step the controller RNN outputs a probability distribution.

https://arxiv.org/abs/1511.04834

Neural Programmer (cont‘d)



…and use attention to make it differentiable!

Summary

- Connectionist Temporal Classification
  - allows to directly learn sequence-to-sequence mappings by introducing a blank (eps) symbol and marginalizing over all alignments
  - time-synchronous
- Attention allows a network to „look at the big picture“
  - Different attention mechanisms (content-based, self, multi-head, …)
  - Typically includes feed-forward layer as transformation
  - Attention can often be nicely visualized (→ „explainable AI“)
  - Transformer works without recurrency and thus requires less data!
- Concept of attention opens up new directions of research (NTM, NP)