Attention Models in Deep Learning

D. Bahdanau for IFT 6266

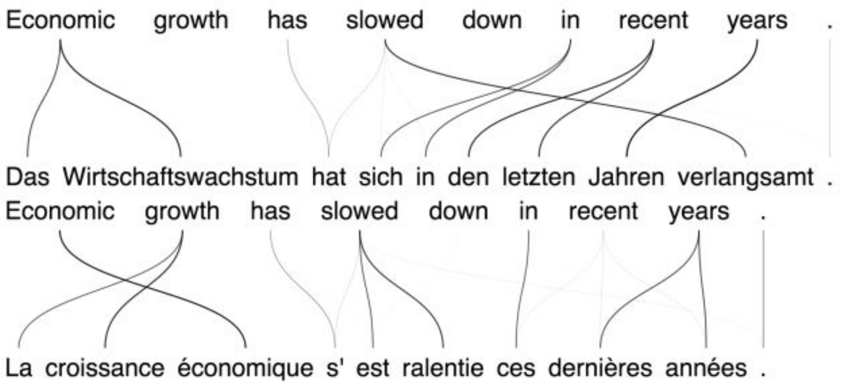
A lot of things are called “attention” these days...

1. Attention (alignment) models used in applications of deep supervised learning with **variable-length** inputs and outputs (typical sequential).
2. Models of visual attention that process a region of an image at high resolution or the whole image at low resolution.
3. Internal self-attention mechanisms can be used to replace recurrent and convolutional networks for sequential data.
4. Addressing schemes of memory-augmented neural networks (next lecture by Aaron)

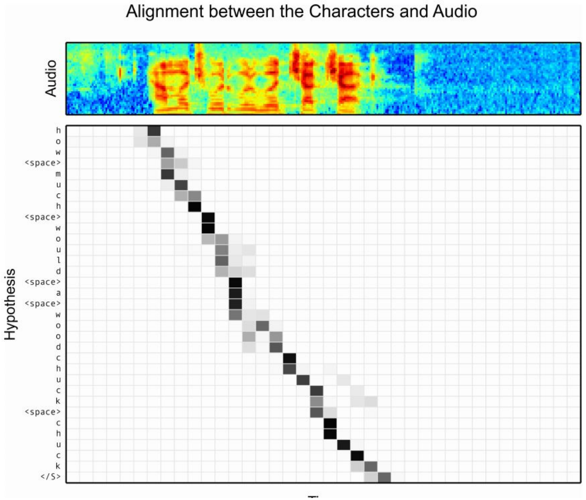
The shared idea: **focus on the relevant parts of the input (output).**

Attention in Deep Learning Applications [to Language Processing]

machine translation



speech recognition



speech synthesis, summarization, … any sequence-to-sequence (seq2seq) task

Traditional deep learning approach

input -> d-dimensional feature vector -> layer_1 -> .... -> layer_k -> output

Good for: image classification, phoneme recognition, decision-making in reflex agents (ATARI)
Less good for: text classification
Not really good for: … everything else?!

Example: Machine Translation

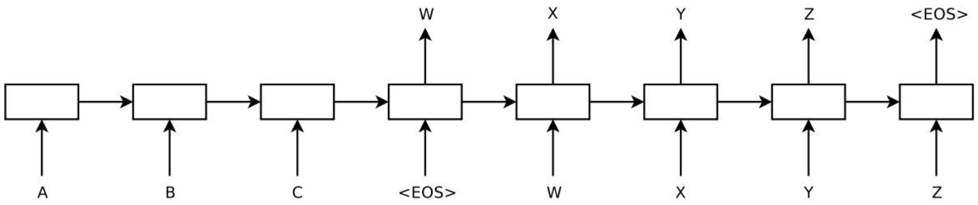
[“An”, “RNN”, “example”, “.”] -> [“Un”, “example”, “de”, “RNN”, “.”]

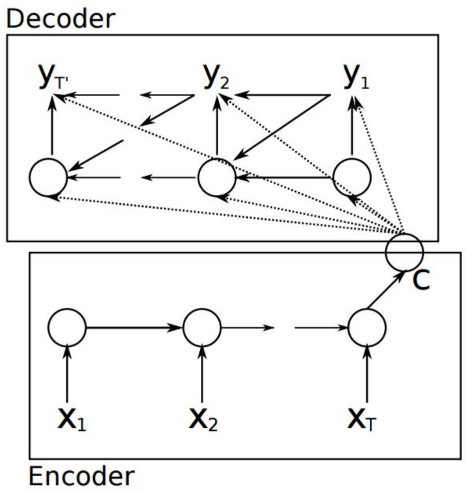
Machine translation presented a challenge to vanilla deep learning

- input and output are sequences
- the lengths vary
- input and output may have different lengths
- no obvious correspondence between positions in the input and in the output

Vanilla seq2seq learning for machine translation

Recurrent Continuous Translation Models, Kalchbrenner et al, EMNLP 2013
Sequence to Sequence Learning with Recurrent Neural Networks, Sutskever et al., NIPS 2014
Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation, Cho et al., EMNLP 2014



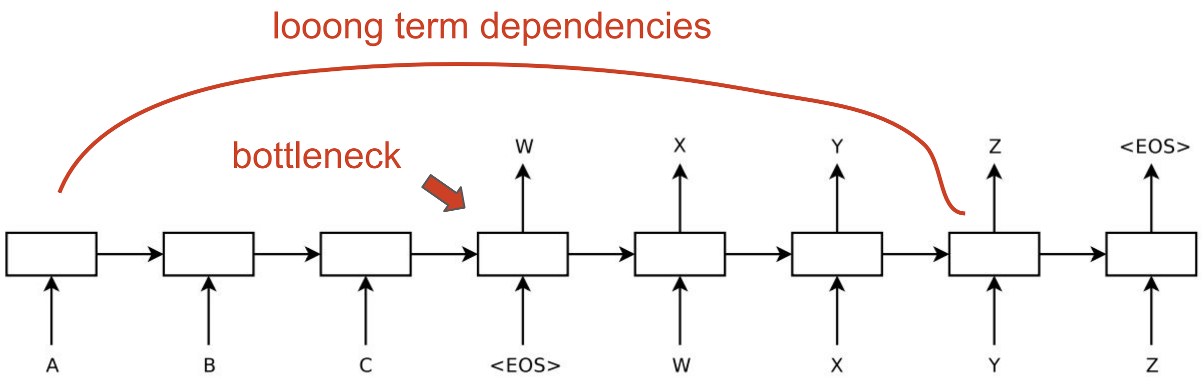


input sequence output sequence

$$p(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \ldots, y_{t-1})$$

fixed size representation

Problems with vanilla seq2seq

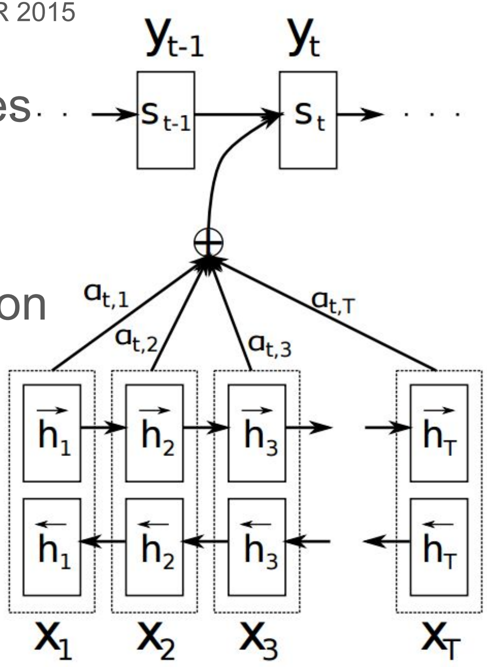


- training the network to encode 50 words in a vector is hard => very big models are needed
- gradients has to flow for 50 steps back without vanishing => training can be slow and require lots of data

Soft attention

Neural Machine Translation by Jointly Learning to Align and Translate, Bahdanau et al, ICLR 2015

- lets decoder focus on the relevant hidden states of the encoder, avoids squeezing everything into the last hidden state => no bottleneck!
- dynamically creates shortcuts in the computation graph that allow the gradient to flow freely => shorter dependencies!
- best with a bidirectional encoder

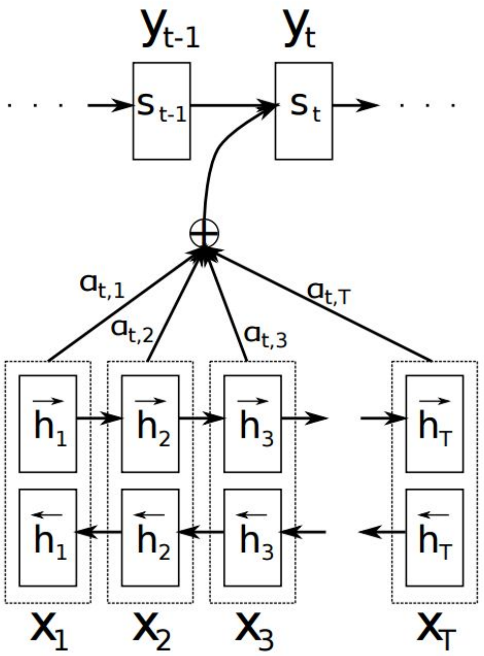


Soft attention - math 1

At each step the decoder consumes a different weighted combination of the encoder states, called context vector or glimpse.

$$p(y_i | y_1, \ldots, y_{i-1}, \mathbf{x}) = g(\cancel{y_{i-1}}, s_i, c_i)$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$



Soft attention - math 2

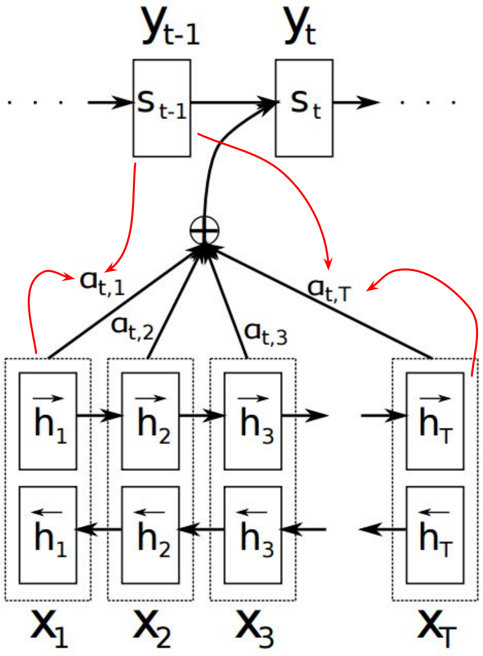
But where do the weights come from? They are computed by another network!

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})},$$

$$e_{ij} = a(s_{i-1}, h_j)$$

The choice from the original paper is 1-layer MLP:

$$a(s_{i-1}, h_j) = v_a^\top \tanh(W_a s_{i-1} + U_a h_j)$$



Soft attention - computational aspects

The computational complexity of using soft attention is quadratic. But it's not slow:

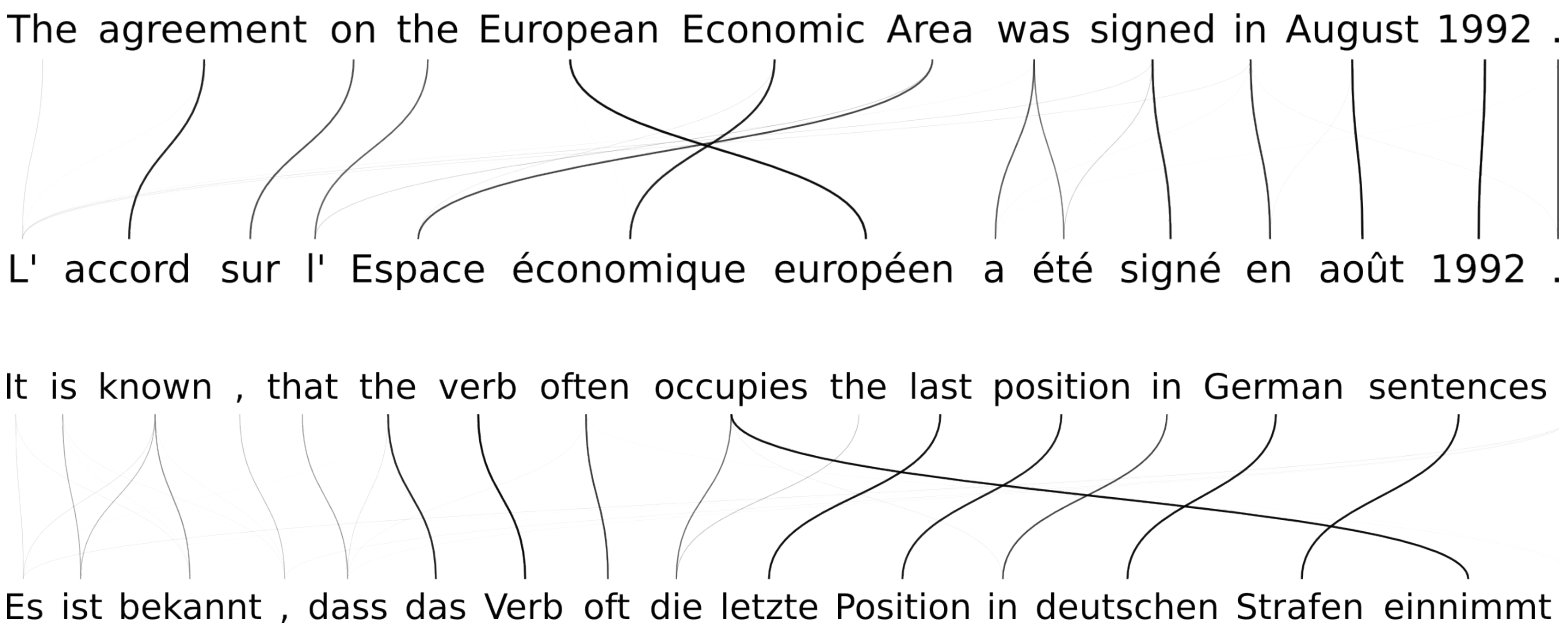
- for each pair of i and j
  - sum two vectors
  - apply tanh
  - compute dot product
- can be done in parallel for all j, i.e.
  - add a vector to a matrix
  - apply tanh
  - compute vector-matrix product
- softmax is cheap
- weighted combination is another vector-matrix product
- in summary: just vector-matrix products = fast!

$$e_{ij} = v_a^\top \tanh\left(W_a s_{i-1} + U_a h_j\right)$$

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}\right)}$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j,$$

Soft attention - visualization

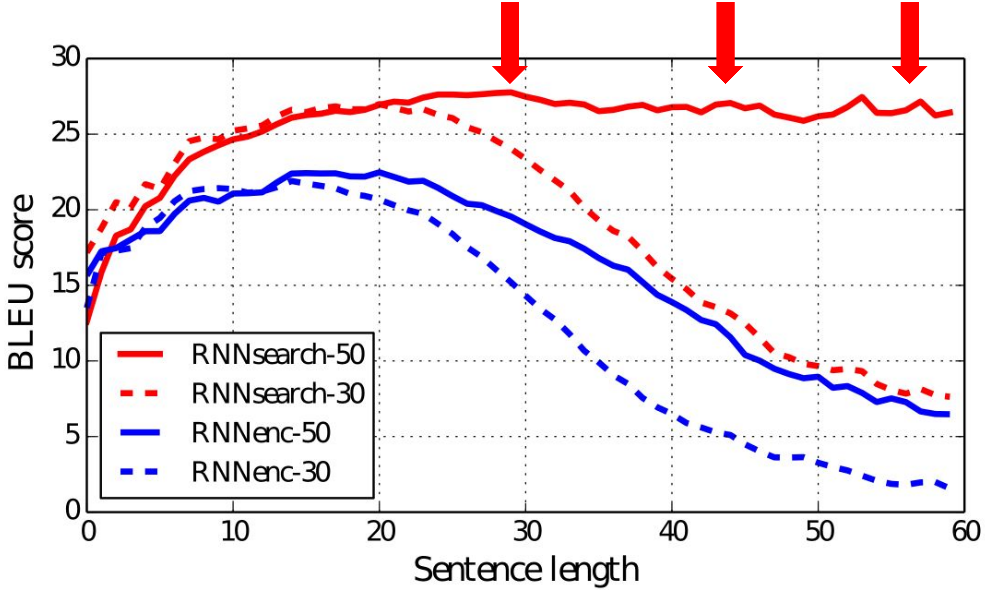


Great visualizations at http://distill.pub/2016/augmented-rnns/#attentional-interfaces

[penalty???]

Soft attention - improvements

no performance drop on long sentences



much better than RNN Encoder-Decoder

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

without unknown words comparable with the SMT system

Soft content-based attention pros and cons

Pros

- faster training, better performance
- good inductive bias for many tasks => lowers sample complexity

Cons

- not good enough inductive bias for tasks with monotonic alignment (handwriting recognition, speech recognition)
- chokes on sequences of length >1000

Exercise: what would happen if we remove biRNN?

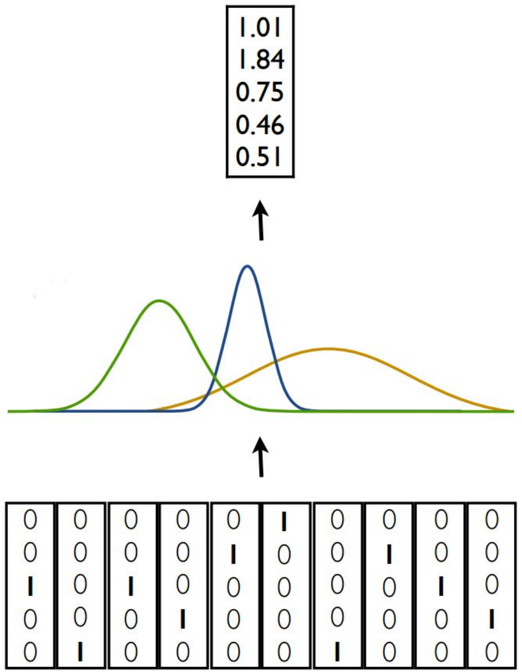
Location-based attention

- in **content-based** attention the attention weights depend on the content at different positions of the input (hence BiRNN)
- in **location-based** attention the current attention weights are computed relative to the previous attention weights

Gaussian mixture location-based attention

Section 5, Generating Sequence with Recurrent Neural Networks, A. Graves 2014

Originally proposed for handwriting synthesis.
The (unnormalized) weight of the input position u at the time step t is parametrized as a mixture of K Gaussians



$$\phi(t,u) = \sum_{k=1}^{K} \alpha_t^k \exp\left(-\beta_t^k \left(\kappa_t^k - u\right)^2\right)$$

$$w_t = \sum_{u=1}^{U} \phi(t,u) c_u$$

Gaussian mixture location-based attention

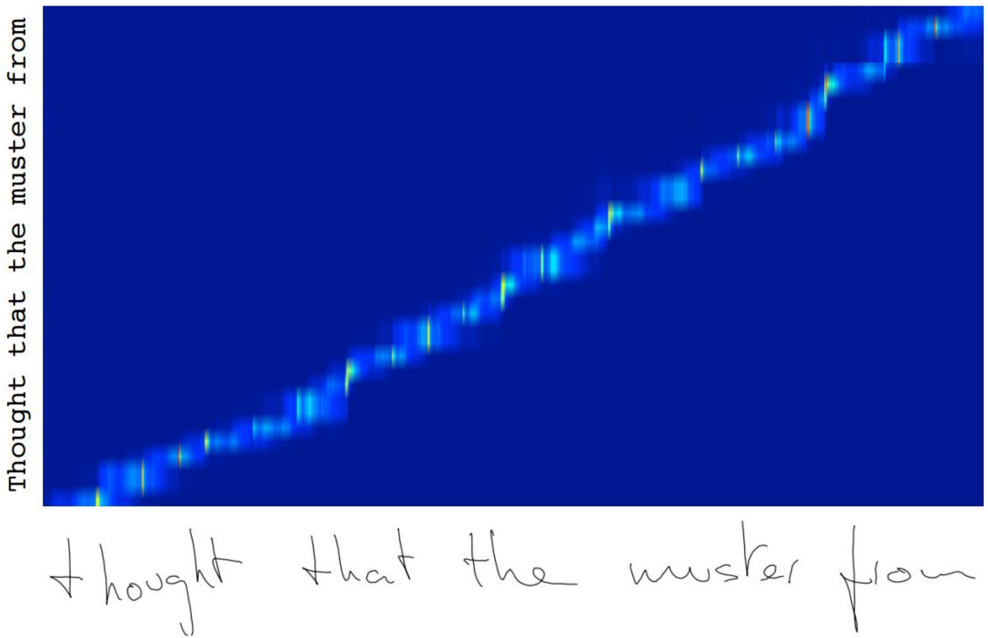
The new locations of Gaussians are computed as a sum of the previous ones and the predicted offsets

$$(\hat{\alpha}_t, \hat{\beta}_t, \hat{\kappa}_t) = W_{h^1 p} h_t^1 + b_p$$

$$\alpha_t = \exp\left(\hat{\alpha}_t\right)$$

$$\beta_t = \exp\left(\hat{\beta}_t\right)$$

$$\kappa_t = \kappa_{t-1} + \exp\left(\hat{\kappa}_t\right)$$



Gaussian mixture location-based attention

- the first soft attention mechanism ever!

Pros

- good for problems with monotonic alignment

Cons:

- predicting the offset can be challenging
- only monotonic alignment (although *exp* in theory could be removed)

Various soft-attentions

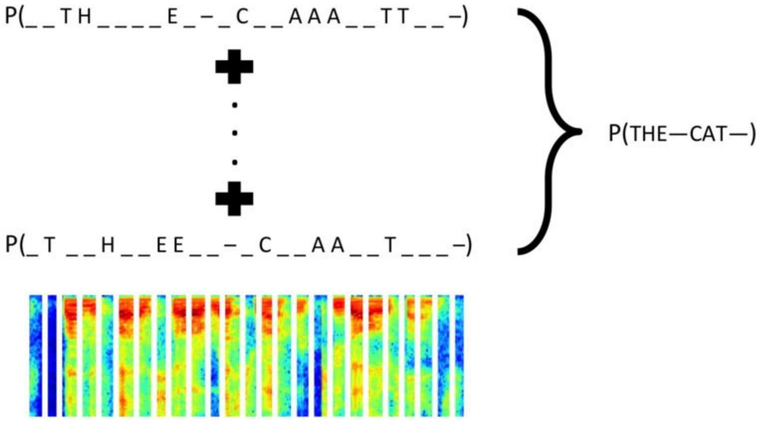
- use dot-product or non-linearity of choice instead of tanh in content-based attention
- use unidirectional RNN insteaf of Bi- (but not pure word embeddings!)
- explicitly remember past alignments with an RNN
- use a separate embedding for each of the positions of the input (heavily used in Memory Networks)
- mix content-based and location-based attentions

See “Attention-Based Models for Speech Recognition” by Chorowski et al (2015) for a scalability analysis of various attention mechanisms on speech recognition.

Going back in time: Connection Temporal Classification (CTC)

Connectionist Temporal Classification: Labellling Unsegmented Sequence Data with Recurrent Neural Networks, Graves et al, ICML 2006

- CTC is a predecessor of soft attention that is still widely used
- has very successful inductive bias for monotonous seq2seq transduction
- core idea: sum over all possible ways of inserting blank tokens in the output so that it aligns with the input



CTC

labeling

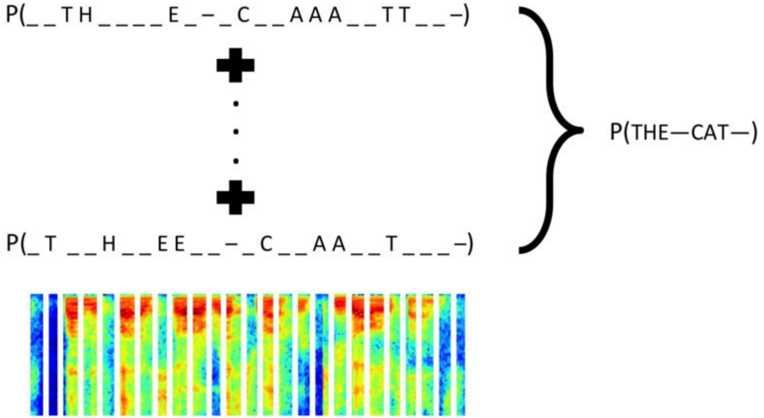
conditional probability of a labeling with blanks

probability of outputting \pi_t at the step t

$$p(l|x) = \sum_{\pi \in \mathcal{B}^{-1}(l)} p(\pi|x) = \sum_{\pi \in \mathcal{B}^{-1}(l)} \prod_t y^t_{\pi_t}$$

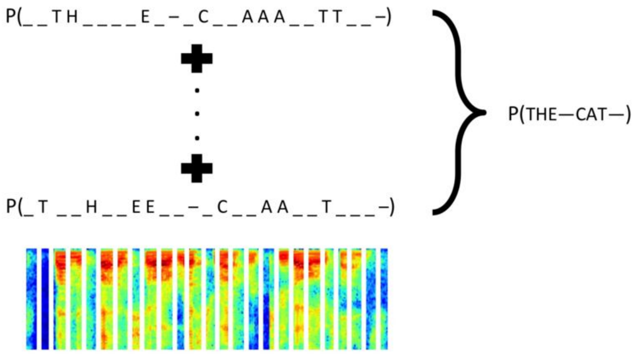
input

sum over all labelling with blanks



CTC

- can be viewed as modelling p(y|x) as sum of all p(y|a,x), where a is a monotonic alignment
- thanks to the monotonicity assumption the marginalization of a can be carried out with forward-backward algorithm (a.k.a. dynamic programming)
- *hard stochastic monotonic attention*
- popular in speech and handwriting recognition
- y_i are conditionally independent given a and x but this can be fixed



Soft Attention and CTC for seq2seq: summary

- the most flexible and general is content-based soft attention and it is very widely used, especially in natural language processing
- location-based soft attention is appropriate for when the input and the output can be monotonously aligned; location-based and content-based approaches can be mixed
- CTC is less generic but can be hard to beat on tasks with monotonous alignments

Visual and Hard Attention

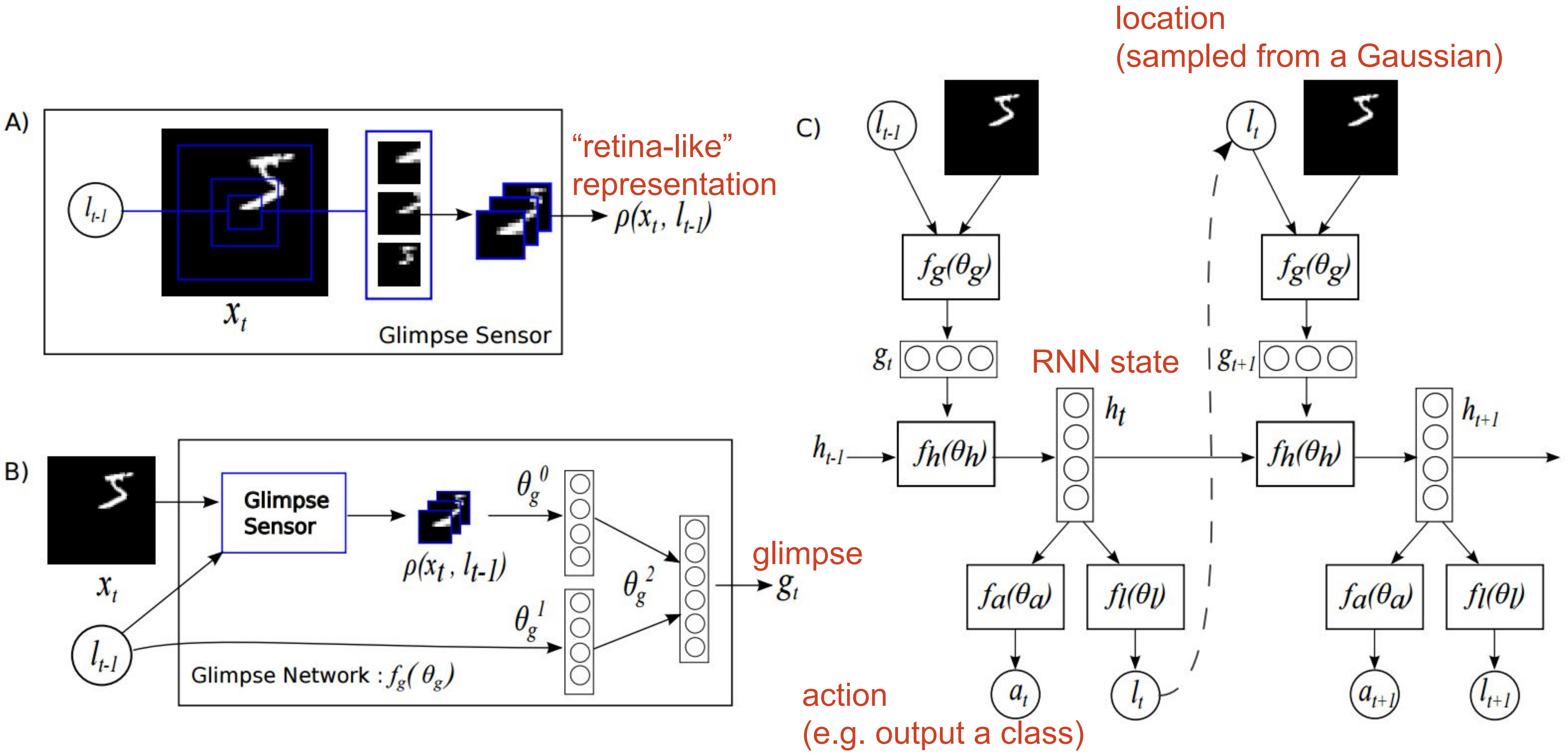
A dog is standing on a hardwood floor.

Models of Visual Attention

Recurrent Models of Visual Attention, V. Mnih et al, NIPS 2014

- Convnets are great! But they process the whole image at a high resolution.
- “*Instead humans focus attention selectively on parts of the visual space to acquire information when and where it is needed, and combine information from different fixations over time to build up an internal representation of the scene*” (Mhin et al, 2014)
- hence the idea: build a recurrent network that focus on a patch of an input image at each step and combines information from multiple steps

A Recurrent Model of Visual Attention



A Recurrent Model of Visual Attention - math 1

Objective:

interaction sequence

$$J(\theta) = \mathbb{E}_{p(s_{1:T};\theta)}\left[\sum_{t=1}^{T} r_t\right] = \mathbb{E}_{p(s_{1:T};\theta)}[R],$$

sum of rewards

When used for classification the correct class is known. Instead of sampling the actions the following expression is used as a reward:

$$\log \pi(a_T^* | s_{1:T}; \theta)$$

=> optimizes Jensen lower bound on the log-probability $p(a^*|x)$!

A Recurrent Model of Visual Attention

next action

The gradient of J has to be approximated (REINFORCE)

$$\nabla_\theta J = \sum_{t=1}^{T} \mathbb{E}_{p(s_{1:T};\theta)} \left[\nabla_\theta \log \pi(u_t|s_{1:t};\theta) R\right] \approx \frac{1}{M} \sum_{i=1}^{M} \sum_{t=1}^{T} \nabla_\theta \log \pi(u_t^i|s_{1:t}^i;\theta) R^i$$

Baseline is used to lower the variance of the estimator:

$$\frac{1}{M} \sum_{i=1}^{M} \sum_{t=1}^{T} \nabla_\theta \log \pi(u_t^i|s_{1:t}^i;\theta) \left(R_t^i - b_t\right)$$

A Recurrent Visual Attention Model - visualization

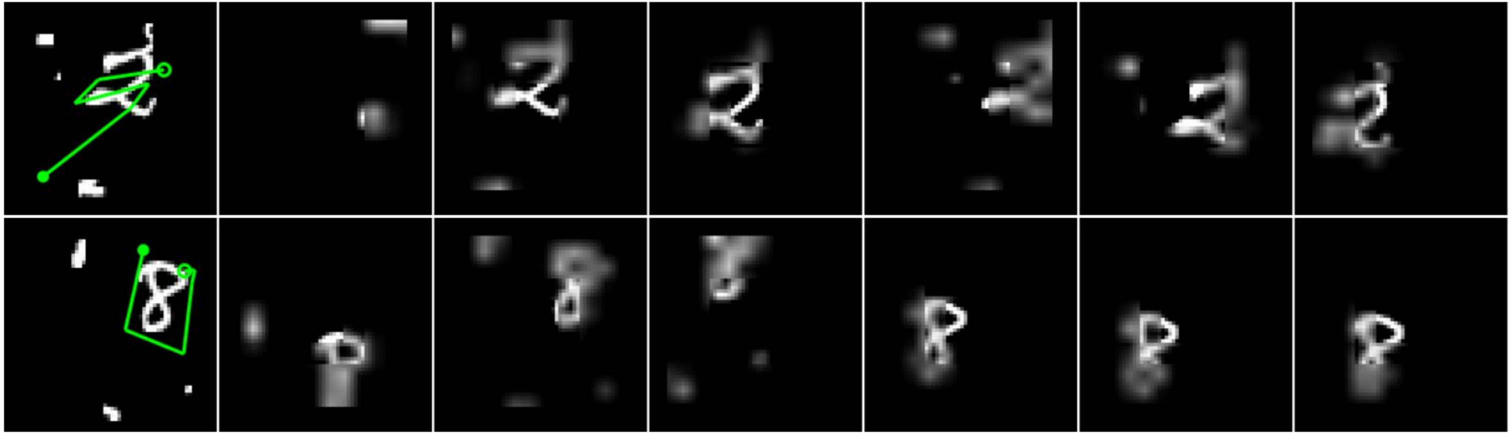


Figure 3: Examples of the learned policy on 60×60 cluttered-translated MNIST task. Column 1: The input image with glimpse path overlaid in green. Columns 2-7: The six glimpses the network chooses. The center of each image shows the full resolution glimpse, the outer low resolution areas are obtained by upscaling the low resolution glimpses back to full image size. The glimpse paths clearly show that the learned policy avoids computation in empty or noisy parts of the input space and directly explores the area around the object of interest.

Soft and Hard Attention

RAM attention mechanism is *hard* - it outputs a precise location where to look.

Content-based attention from neural MT is soft - it assigns weights to all input locations.

CTC can be interpreted as a hard attention mechanism with tractable gradient.

Soft and Hard Attention

Soft

- deterministic
- exact gradient
- O(input size)
- typically easy to train

Hard

- stochastic*
- gradient approximation**
- O(1)
- harder to train

* deterministic hard attention would not have gradients

** exact gradient can be computed for models with tractable marginalization (e.g. CTC)

Soft and Hard Attention

Can soft content-based attention be used for vision? Yes.

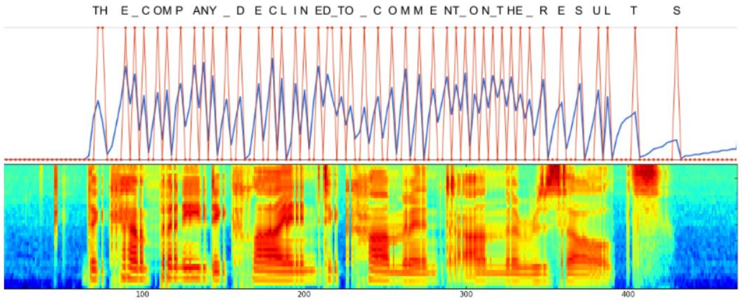
Show Attend and Tell, Xu et al, ICML 2015

A dog is standing on a hardwood floor.

Can hard attention be used for seq2seq? Yes.

Learning Online Alignments with
Continuous Rewards Policy Gradient,
Luo et al, NIPS 2016

(but the learning curves are a nightmare…)



DRAW: soft location-based attention for vision

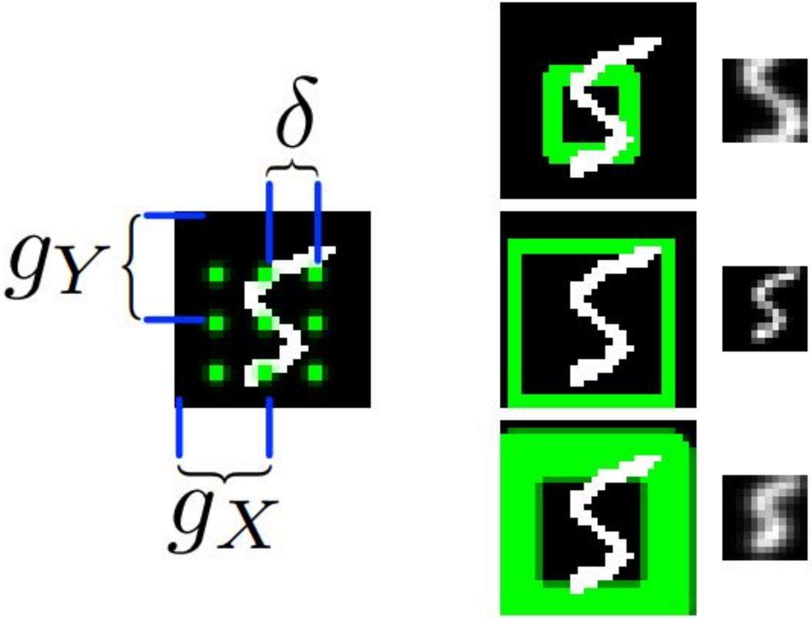


Figure 3. **Left:** A 3×3 grid of filters superimposed on an image. The stride (δ) and centre location (g_X, g_Y) are indicated. **Right:** Three $N \times N$ patches extracted from the image ($N = 12$). The green rectangles on the left indicate the boundary and precision (σ) of the patches, while the patches themselves are shown to the right. The top patch has a small δ and high σ, giving a zoomed-in but blurry view of the centre of the digit; the middle patch has large δ and low σ, effectively downsampling the whole image; and the bottom patch has high δ and σ.

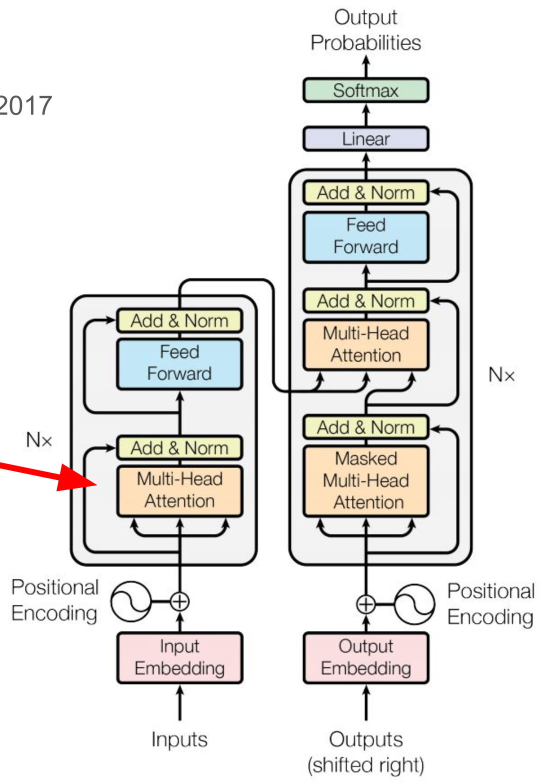
Exercise: can we use DRAW-like attention for speech recognition?

Internal self-attention in deep learning models

Transformer from Google

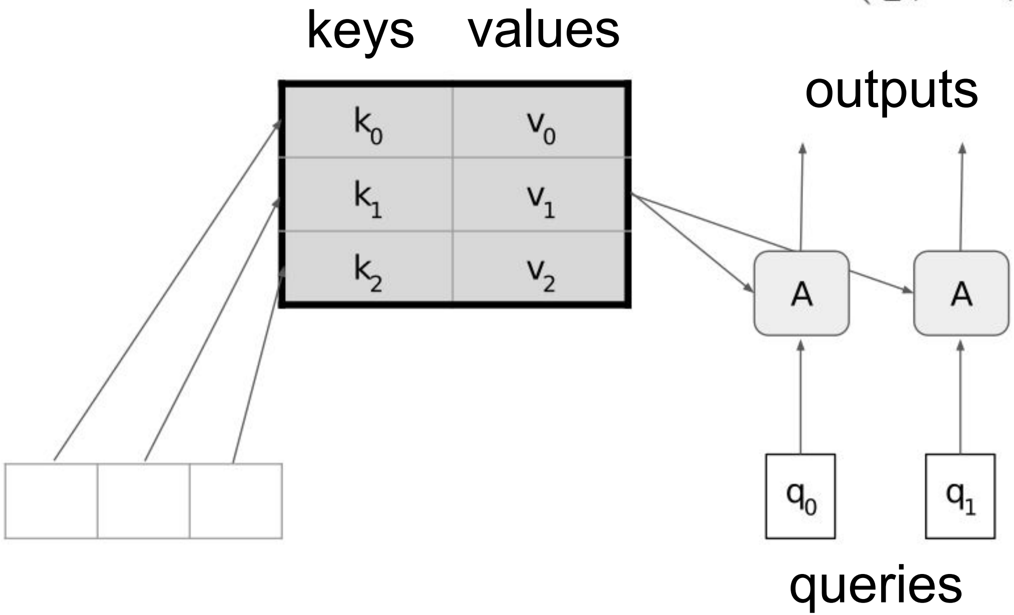
Attention Is All You Need, Vaswani et al, NIPS 2017

In addition to connecting the decoder with the encoder, attention can be used inside the model, replacing RNN and CNN!



Generalized dot-product attention - vector form

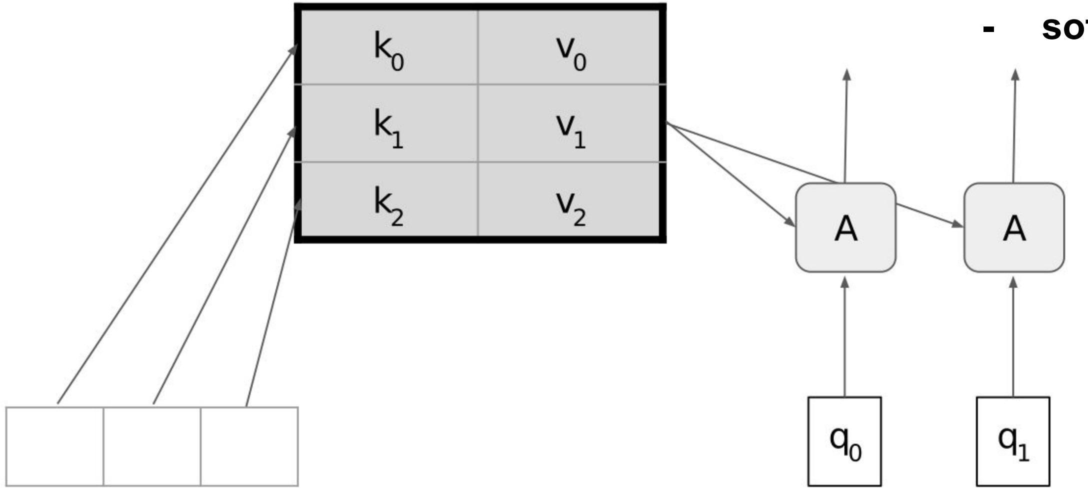
$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i$$



Generalized dot-product attention - matrix form

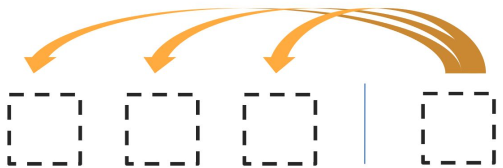
$$A(Q, K, V) = softmax(QK^T)V$$

- rows of Q, K, V are keys, queries, values
- softmax acts row-wise

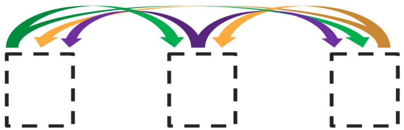


Three types of attention in Transformer

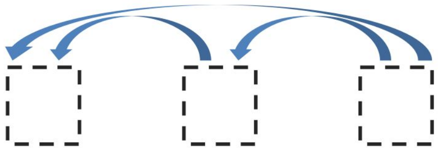
- usual attention between encoder and decoder:
  Q=[current state] K=V=[BiRNN states]



- self-attention in the encoder,
  Q=K=V=[encoder states] - encoder attends to itself!



- masked self-attention in the decoder,
  Q=K=V=[decoder states] - decoder attends to itself,
  but a states can only attend previous states



Other tricks in Transformer

- allows different processing of information coming from different locations

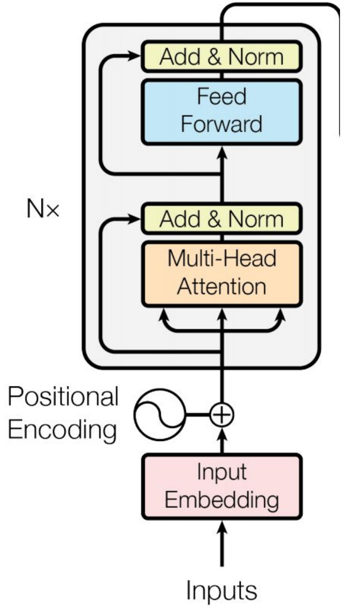
$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW^Q_i, KW^K_i, VW^V_i)$$

- positional embeddings are required to preserve the order information:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\mathrm{model}}}$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\mathrm{model}}}$$

(trainable parameter embeddings also work)

Transformer Full Model and Performance



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	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

- 6 layers like that in encoder
- 6 layers with masking in the decoder
- usual soft-attention between the encoder and the decoder

Summary

- attention is used to focus on parts of inputs/outputs
- it can be content/location based and hard/soft
- it’s three main distinct uses are
    - connecting encoder and decoder in sequence-to-sequence task
    - achieving scale-invariance and focus in image processing
    - self-attention can be a basic building block for neural nets, often replacing RNNs and CNNs [recent research, take it with a grain of salt]