

# COMP547

## DEEP UNSUPERVISED LEARNING

Lecture #4 – Attention and Transformers

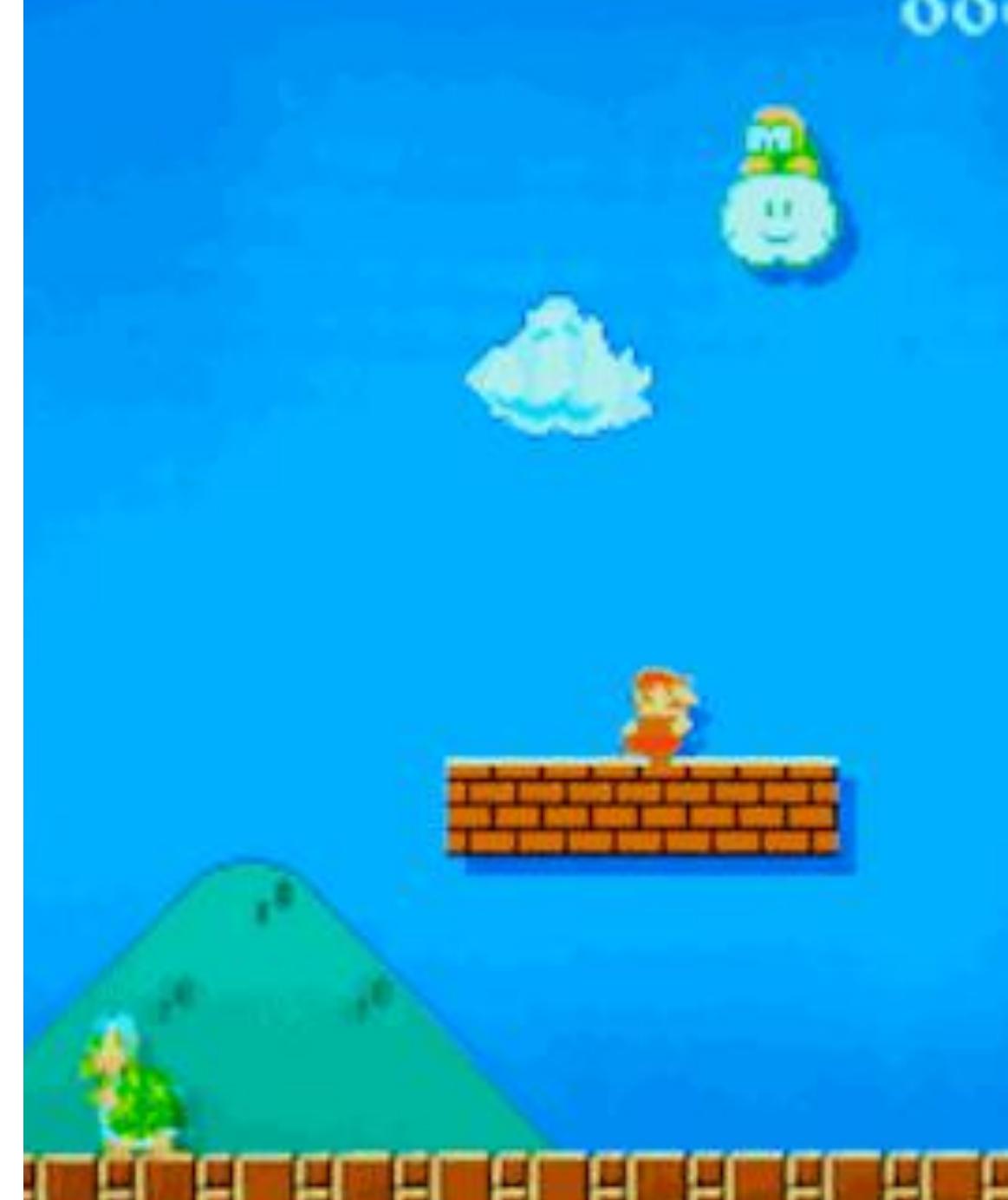


Aykut Erdem // Koç University // Spring 2022

# Previously on COMP547

- sequence modeling
- recurrent neural networks (RNNs)
- language modeling with RNNs
- how to train RNNs
- long short-term memory (LSTM)
- gated recurrent unit (GRU)

Using RNNs to generate Super Mario Maker levels, Adam Geitgey



# Lecture overview

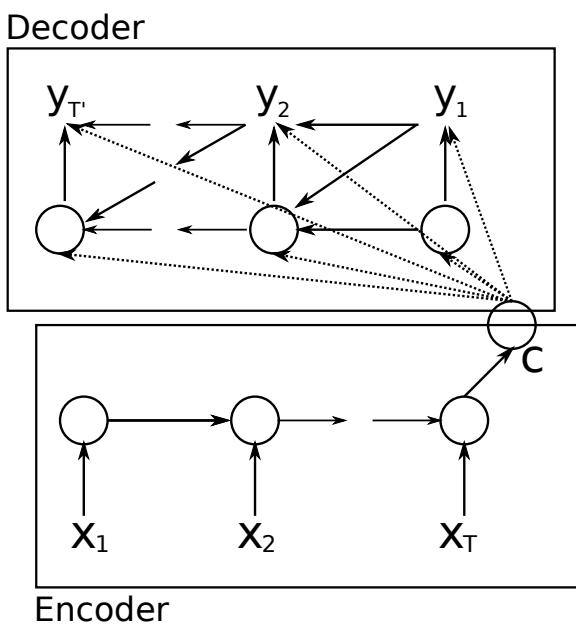
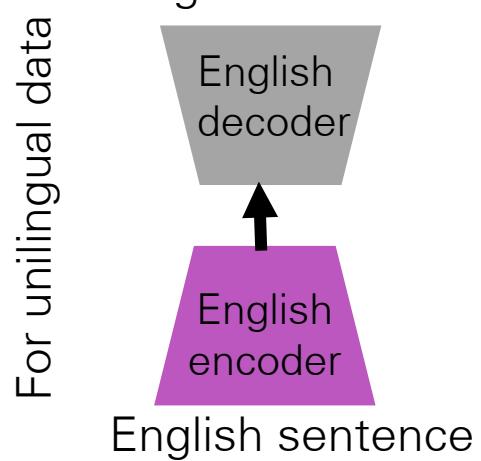
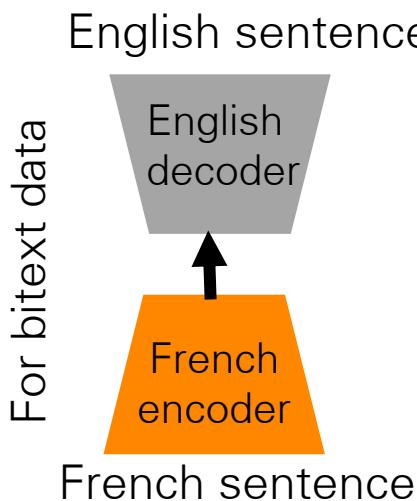
- content-based attention
- location-based attention
- soft vs. hard attention
- case study: Show, Attend and Tell
- self-attention
- case study: Transformer networks

**Disclaimer:** Much of the material and slides for this lecture were borrowed from

- Dzmitry Bahdanau's IFT 6266 slides
- Graham Neubig's CMU CS11-747 Neural Networks for NLP class
- Mateusz Malinowski's lecture on Attention-based Networks
- Yoshua Bengio's talk on From Attention to Memory and towards Longer-Term Dependencies
- Kyunghyun Cho's slides on neural sequence modeling
- Arian Hosseini's IFT 6135 slides
- Phillip Isola's MIT 6.S898 slides

# Encoder-Decoder Framework

- Intermediate representation of meaning  
= 'universal representation'
- Encoder: from word sequence to sentence representation
- Decoder: from representation to word sequence distribution



# Sequence Representations

- But what if we could use multiple vectors, based on the length of the sequence



# Attention Models in Deep Learning

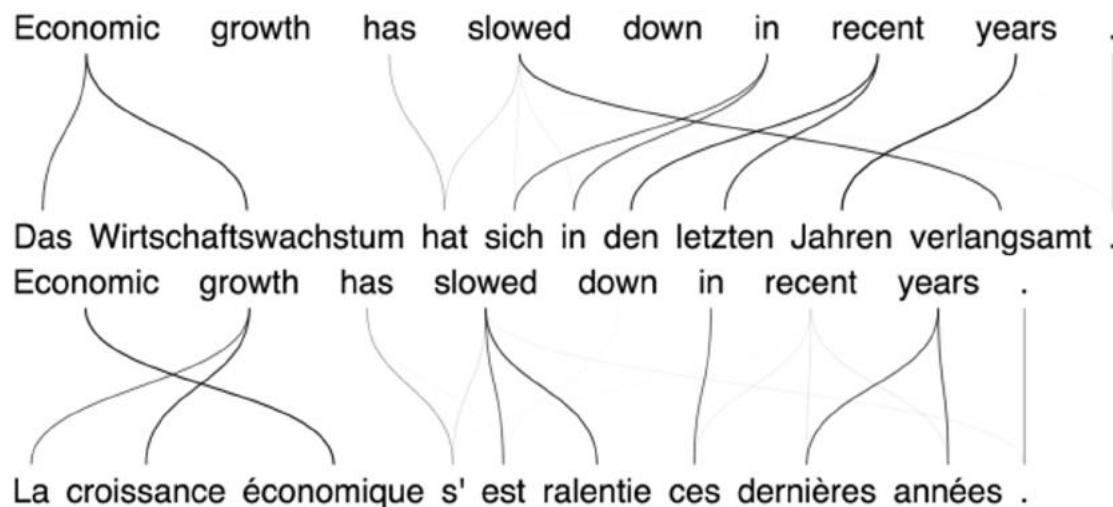
# A lot of things are called “attention” these days...

1. Attention (alignment) models used in applications of deep 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

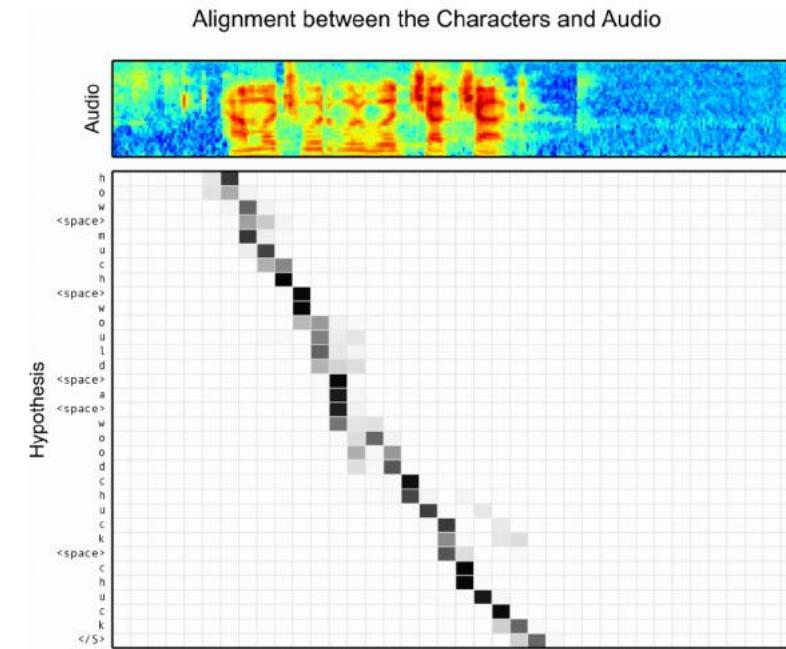
The shared idea: **focus on the relevant parts of the input (output)**.

# Attention in Deep Learning Applications [to Language Processing]

machine translation



speech recognition



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

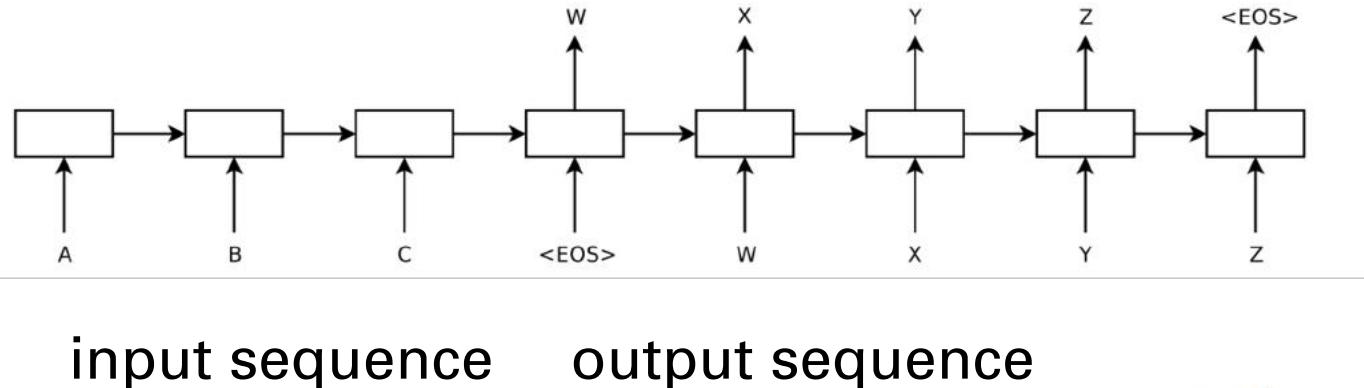
# Example: Machine Translation

[“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



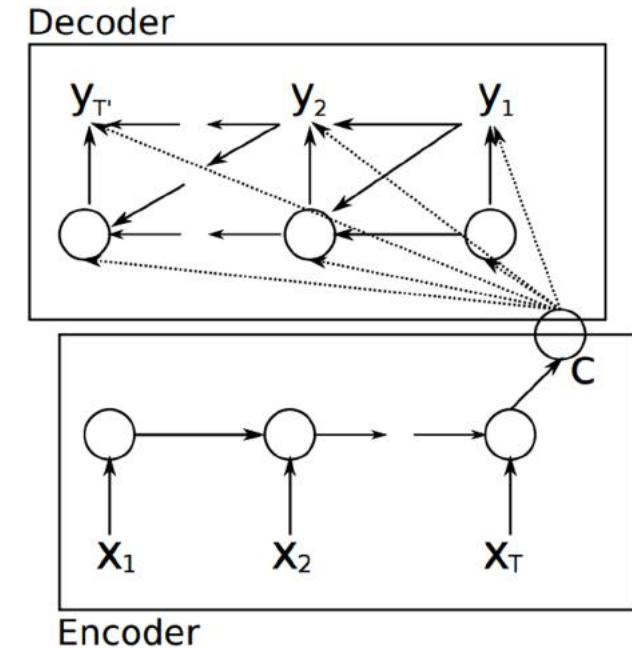
$$p(y_1, \dots, y_{T'} | x_1, \dots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \dots, y_{t-1})$$

fixed size representation

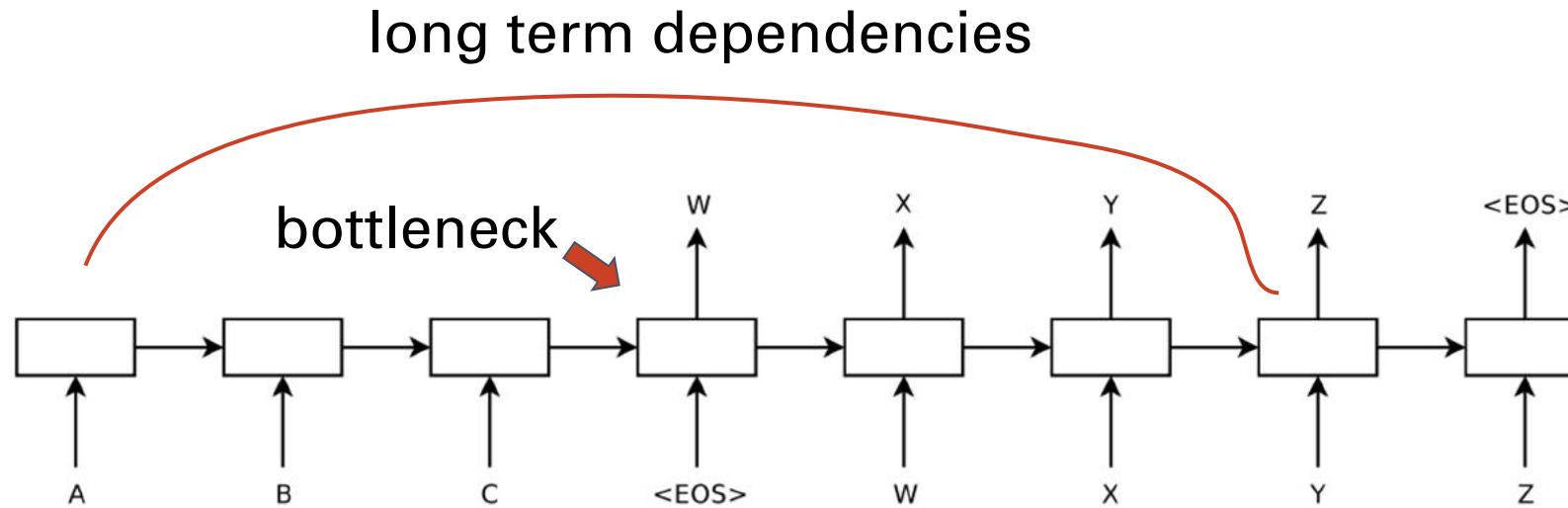
Recurrent Continuous Translation Models, Kalchbrenner et al, EMNLP 2013

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



# Problems with vanilla seq2seq



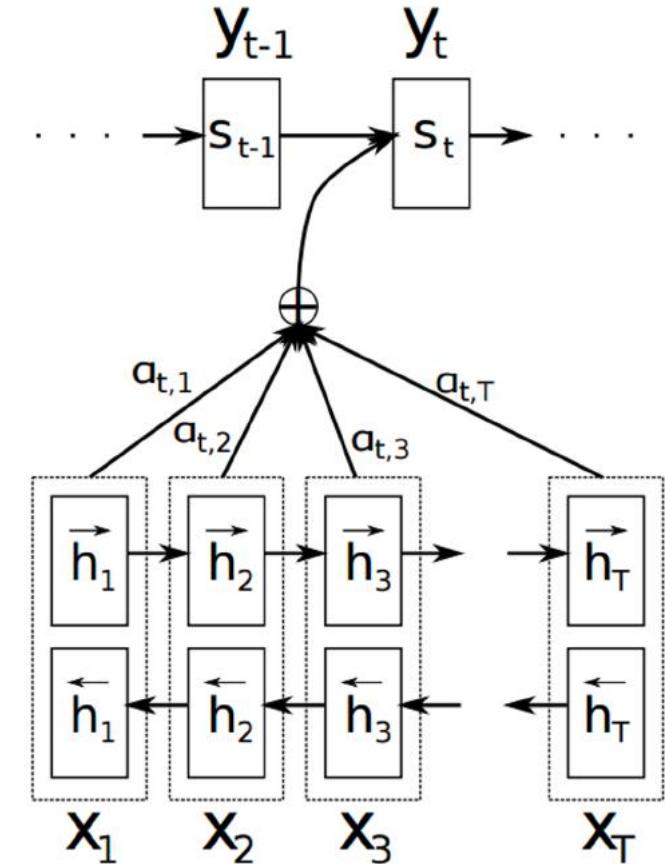
- training the network to encode 50 words in a vector is hard  $\Rightarrow$  very big models are needed
- gradients has to flow for 50 steps back without vanishing  $\Rightarrow$  training can be slow and require lots of data

# Soft attention

lets decoder focus on the relevant hidden states of the encoder, avoids squeezing everything into the last hidden state  $\Rightarrow$  no bottleneck!

dynamically creates shortcuts in the computation graph that allow the gradient to flow freely  
 $\Rightarrow$  shorter dependencies!

best with a bidirectional encoder

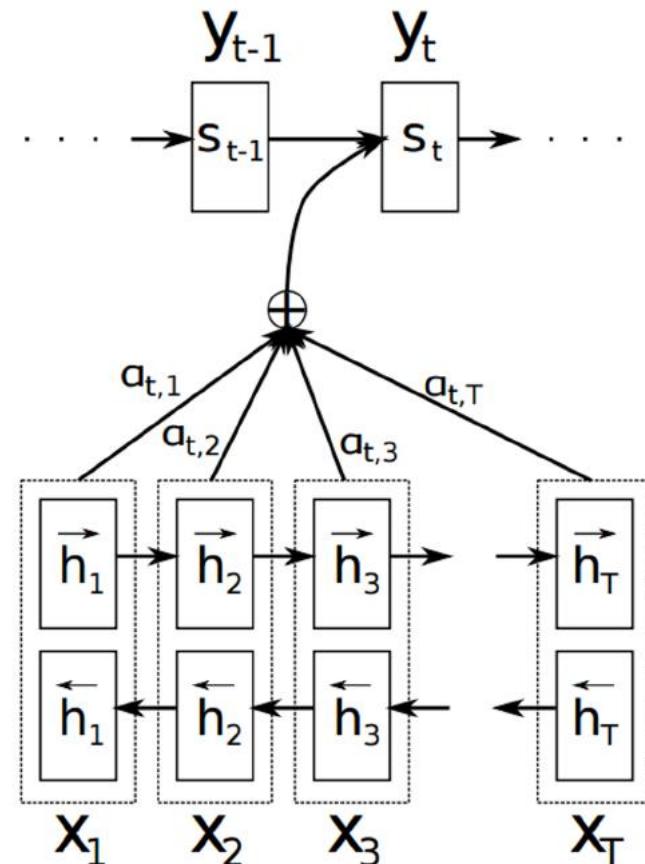


# Soft attention - math 1

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

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

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



# Soft attention - math 2

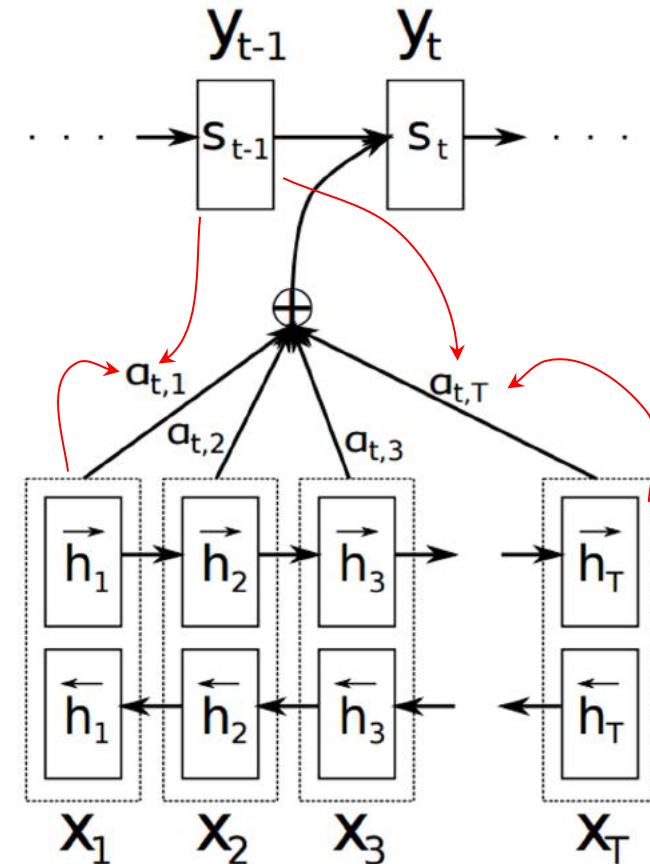
But where do the weights come from?  
They are computed by another network!

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

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

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!**

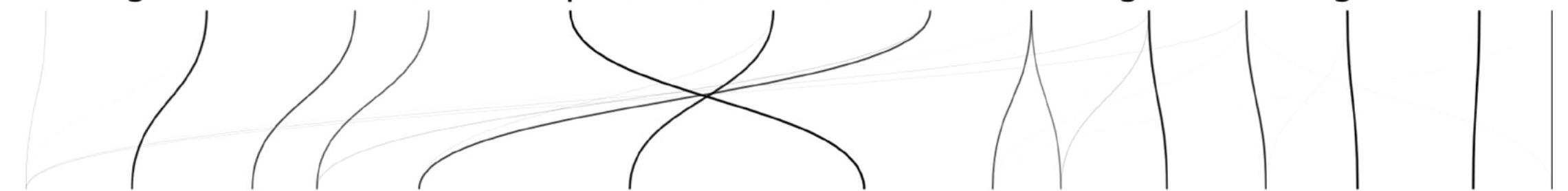
$$e_{ij} = v_a^\top \tanh(W_a s_{i-1} + U_a h_j)$$

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

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

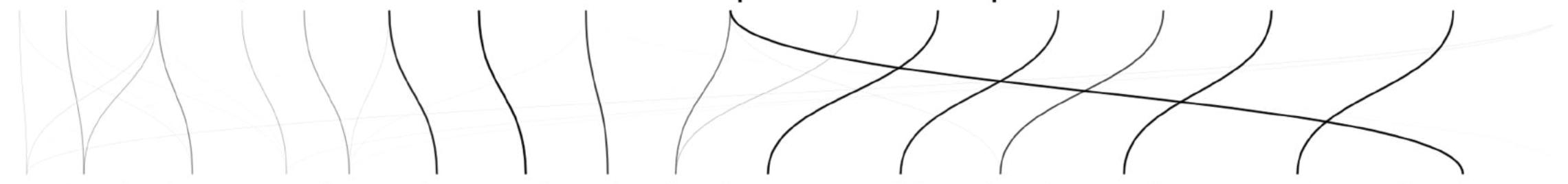
# Soft attention - visualization

The agreement on the European Economic Area was signed in August 1992 .



L' accord sur l' Espace économique européen a été signé en août 1992 .

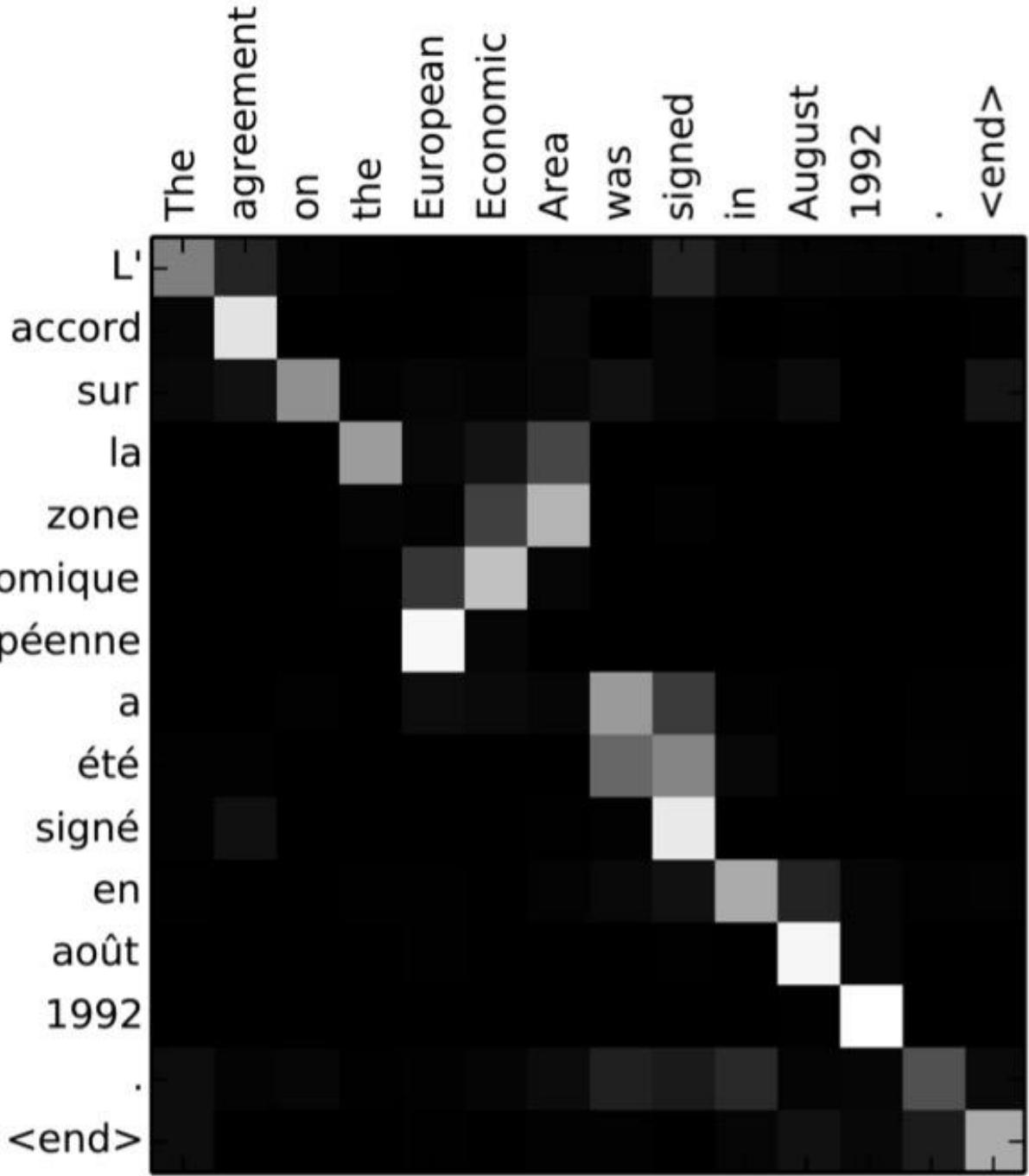
It is known , that the verb often occupies the last position in German sentences



Es ist bekannt , dass das Verb oft die letzte Position in deutschen Strafen einnimmt

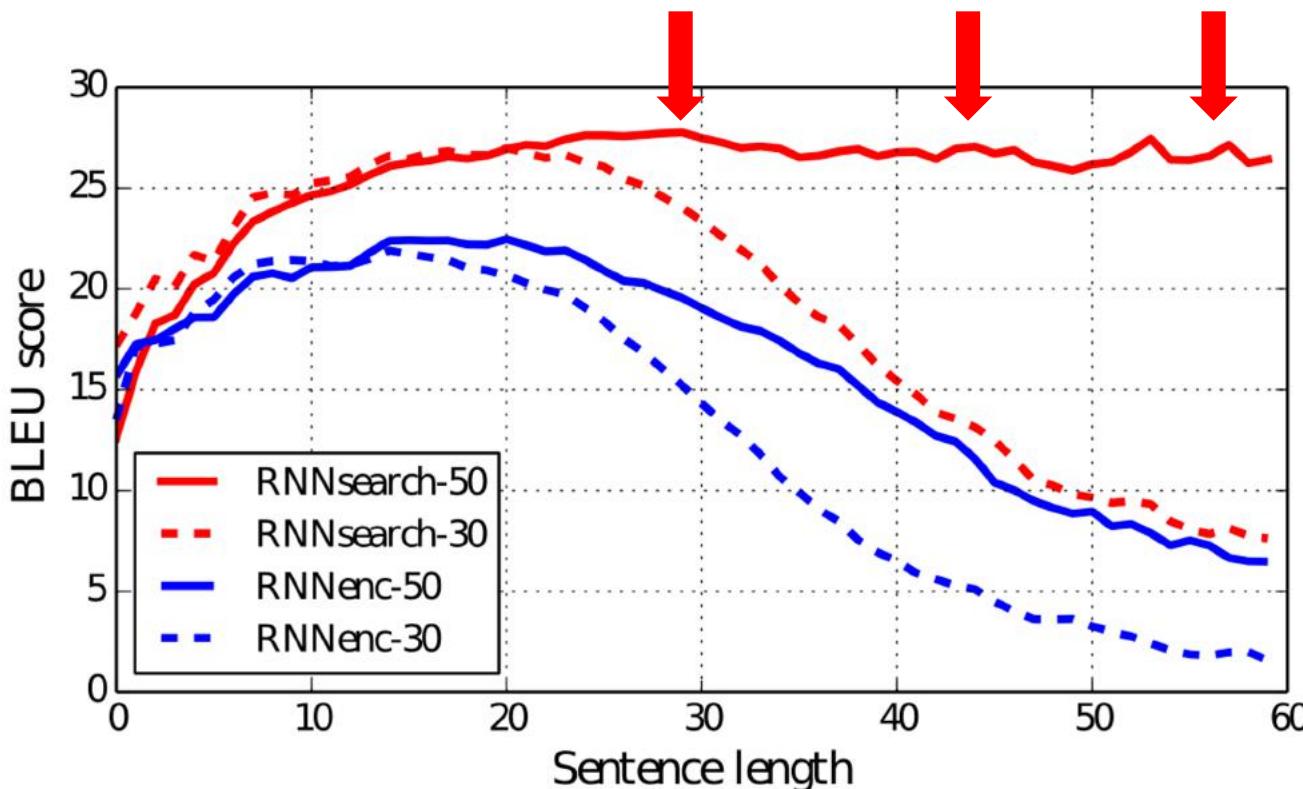
# Soft attention - visualization

(Bahdanau et al 2014, Jean et al 2014,  
Gulcehre et al 2015, Jean et al 2015)



# 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

# End-to-End Machine Translation with Recurrent Nets and Attention Mechanism

(Bahdanau et al 2014, Jean et al 2014, Gulcehre et al 2015, Jean et al 2015)

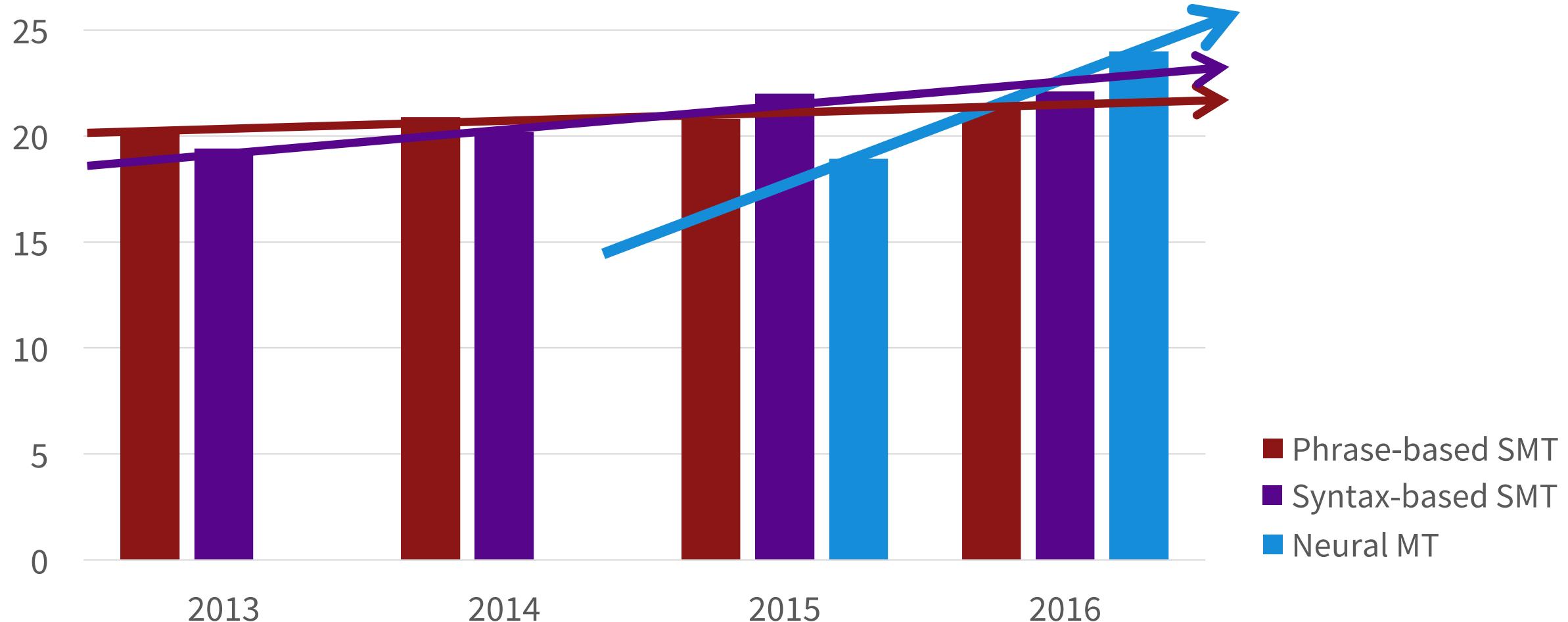


Figure credit: Rico Sennrich

# 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  $> 1000$

# 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

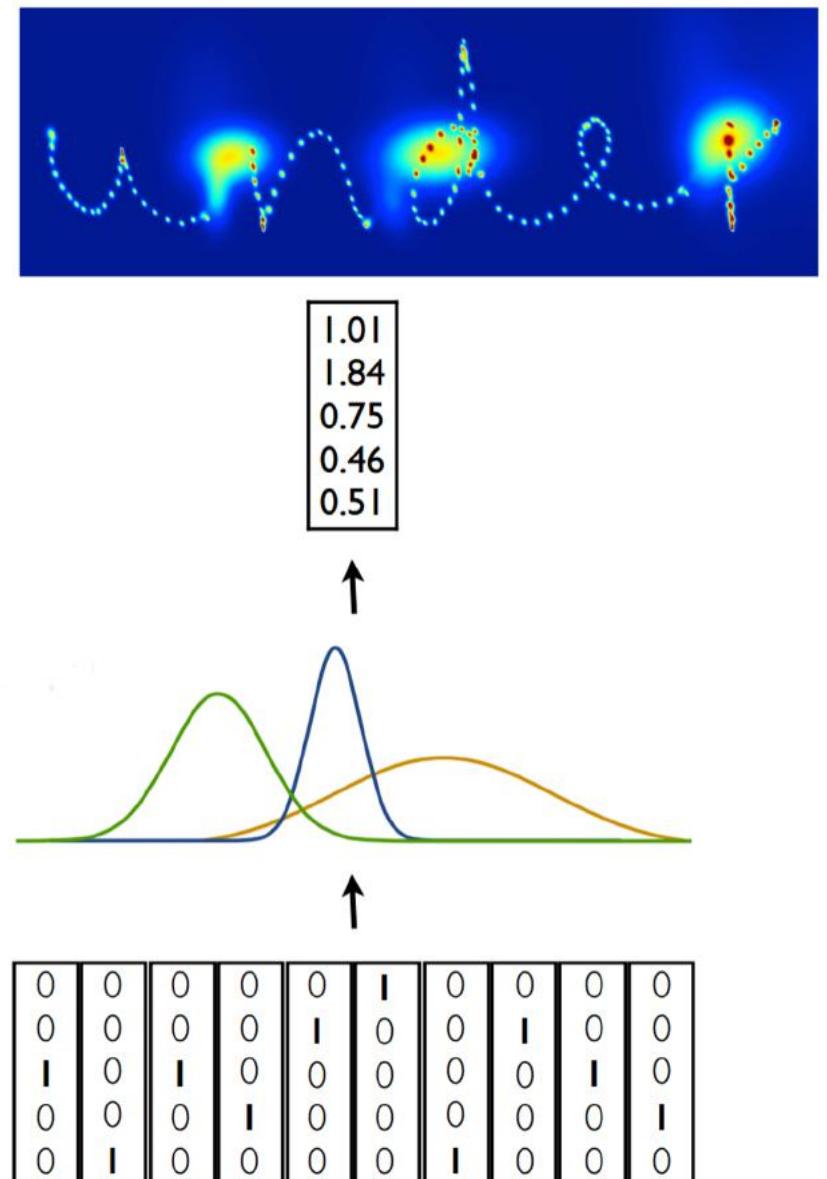
# Gaussian mixture location-based attention

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

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

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



# Gaussian mixture location-based attention

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

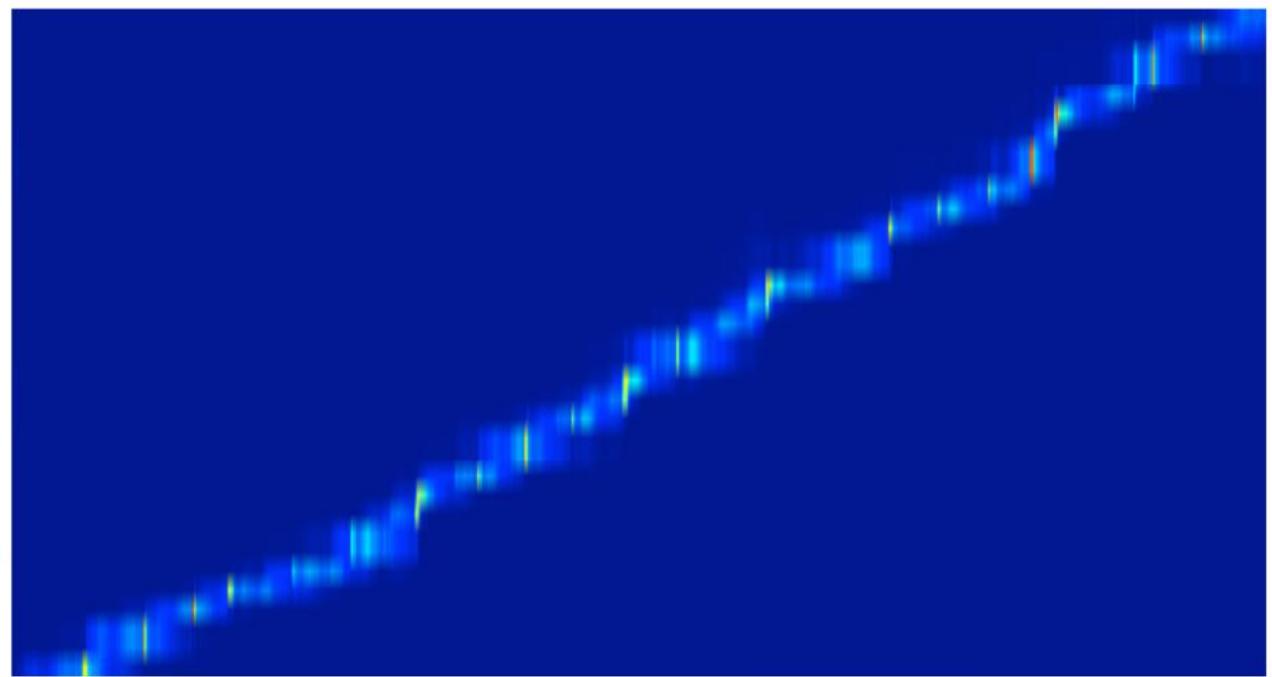
$$(\hat{\alpha}_t, \hat{\beta}_t, \hat{\kappa}_t) = W_{h^1 p} h_t^1 + b_p$$

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

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

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

Thought that the muster from



I thought that the muster from

# 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 instead 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.

# Various Attention Score Functions

- $\mathbf{q}$  is the query and  $\mathbf{k}$  is the key

- Multi-layer Perceptron

(Bahdanau et al. 2015)

$$a(\mathbf{q}, \mathbf{k}) = \mathbf{w}_2^\top \tanh(W_1[\mathbf{q}; \mathbf{k}])$$

- Flexible, often very good with large data

- Bilinear (Luong et al. 2015)

$$a(\mathbf{q}, \mathbf{k}) = \mathbf{q}^\top W \mathbf{k}$$

- Dot Product (Luong et al. 2015)

$$a(\mathbf{q}, \mathbf{k}) = \mathbf{q}^\top \mathbf{k}$$

- No parameters! But requires sizes to be the same.

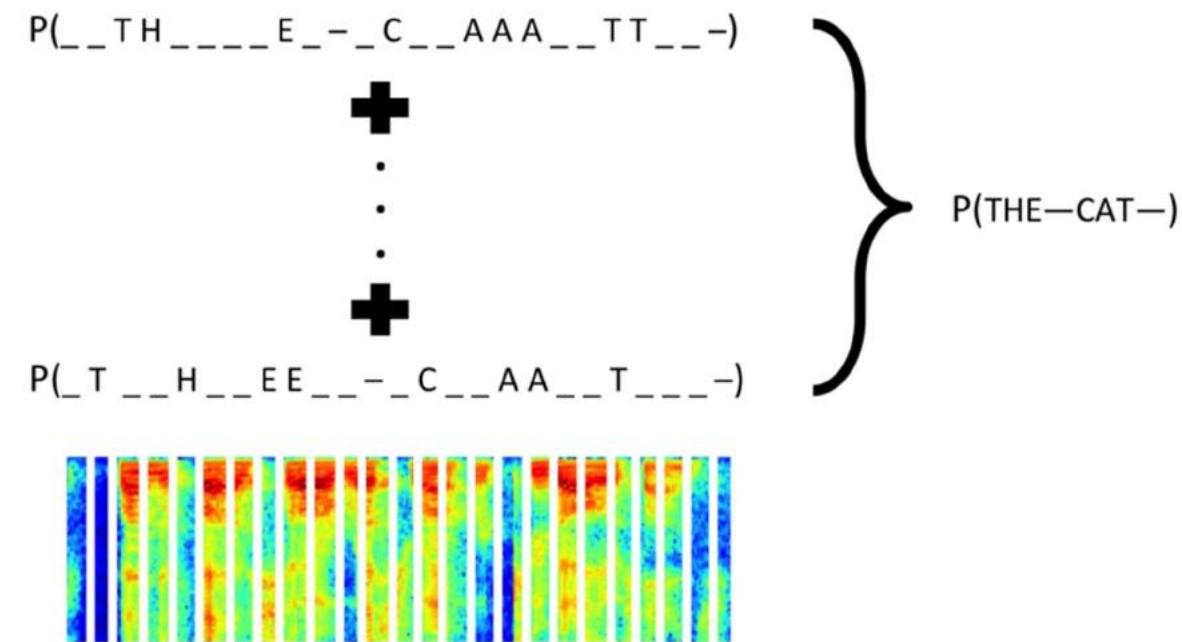
- Scaled Dot Product (Vaswani et al. 2017)

- Problem: scale of dot product increases as dimensions get larger
- Fix: scale by size of the vector

$$a(\mathbf{q}, \mathbf{k}) = \frac{\mathbf{q}^\top \mathbf{k}}{\sqrt{|\mathbf{k}|}}$$

# Going back in time: Connection Temporal Classification (CTC)

- 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

$$p(l|x)$$

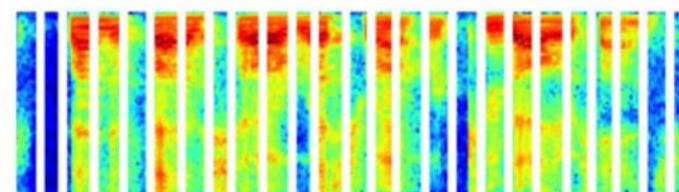
conditional  
probability of a  
labeling with blanks

sum over all labelling  
with blanks

probability of  
outputting  $\pi_t$   
at the step t

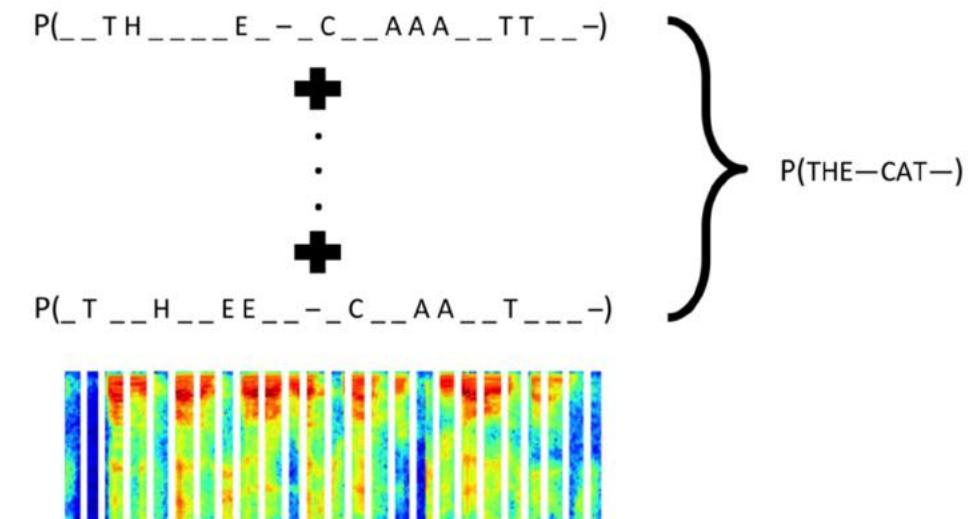
$$P(\text{---TH---EE---C---AA---TT---}) \quad P(\text{THE---CAT---})$$

+  
 .  
 .  
 .  
 +



# 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



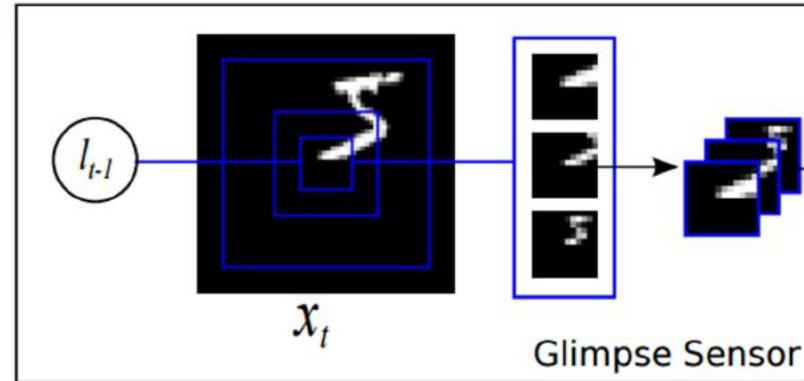
A dog is standing on a hardwood floor.

# Models of Visual Attention

- 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”* (Mnih 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

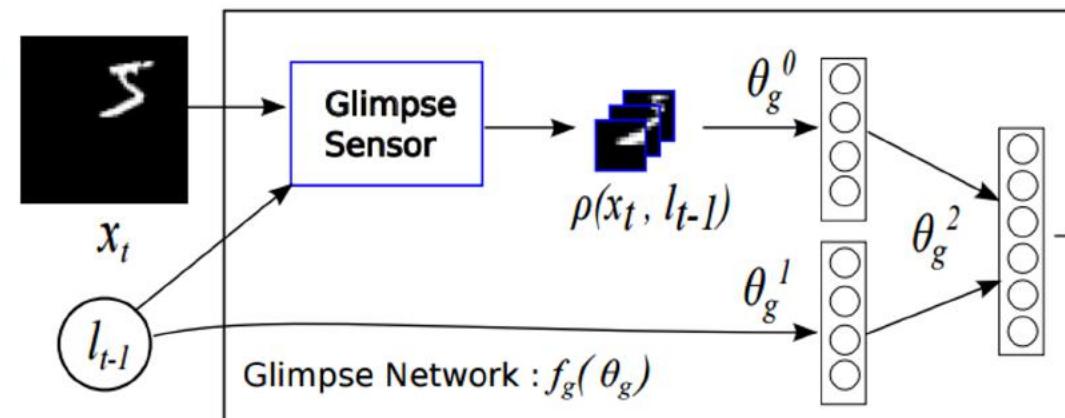
# A Recurrent Model of Visual Attention

A)



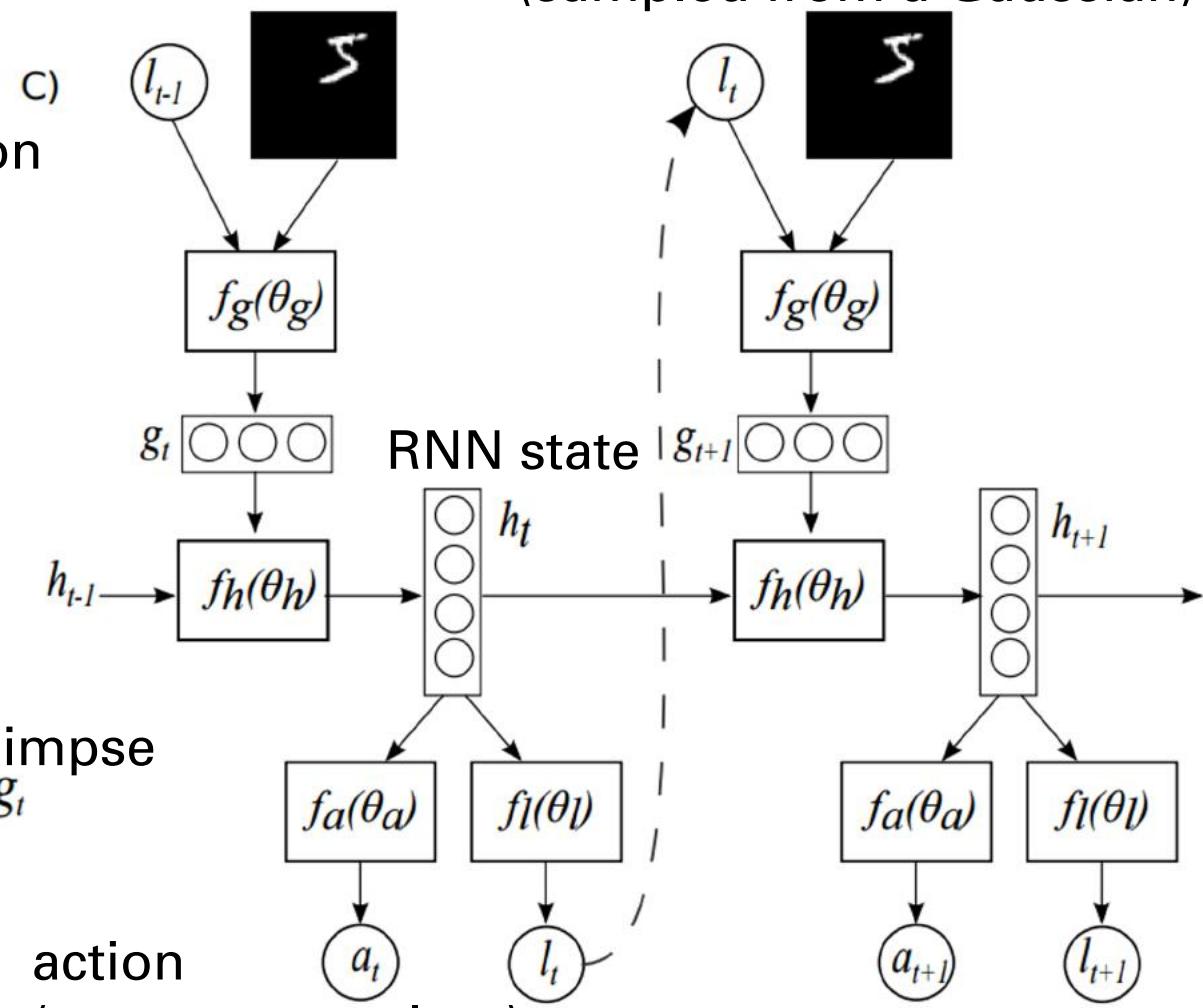
"retina-like" representation  
 $\rightarrow \rho(x_t, l_{t-1})$

B)



glimpse

action  
(e.g. output a class)



location  
(sampled from a Gaussian)

# 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

The gradient of  $J$  has to be approximated (REINFORCE)

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


**Baseline** is used to lower the variance of the estimator:

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

# A Recurrent Visual Attention Model - visualization

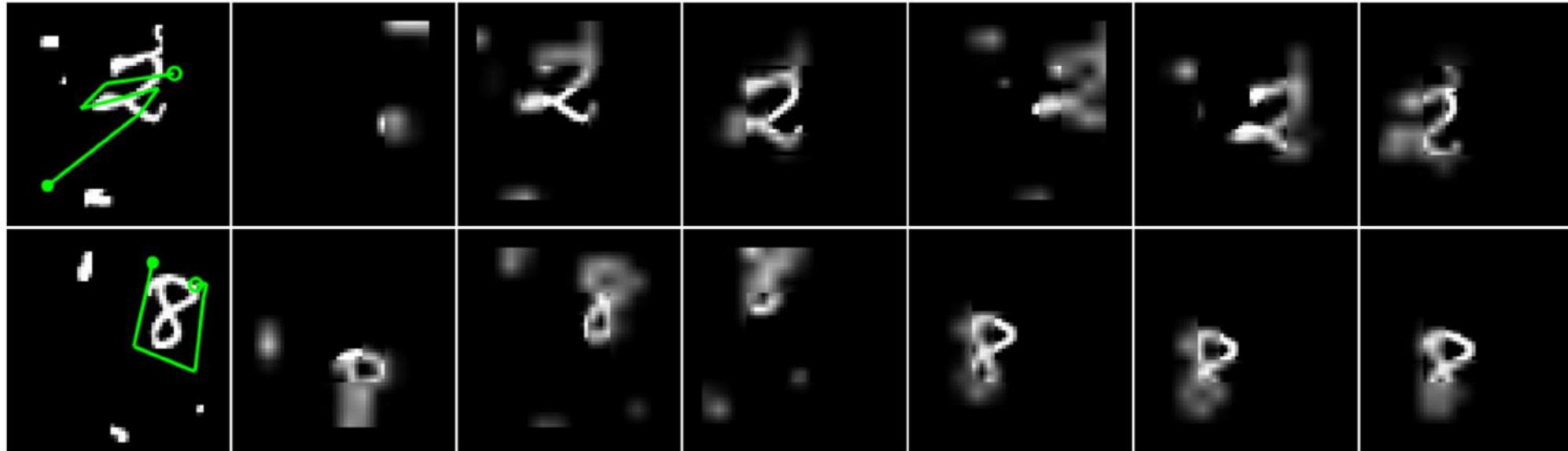


Figure 3: Examples of the learned policy on  $60 \times 60$  cluttered-translated MNIST task. Column 1: The input image with glimpse path overlaid in green. Columns 2-7: The six glimpses the network chooses. The center of each image shows the full resolution glimpse, the outer low resolution areas are 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

Recurrent Attention Model (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
- $O(\text{input size})$
- 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

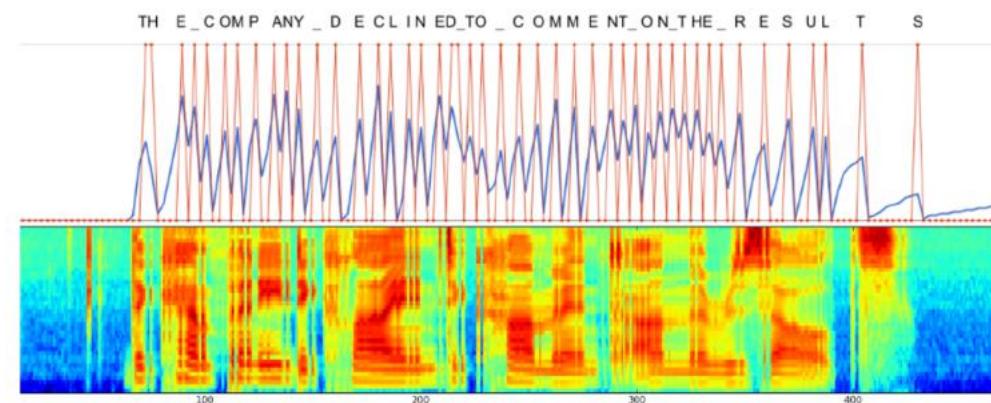


A dog is standing on a hardwood floor.

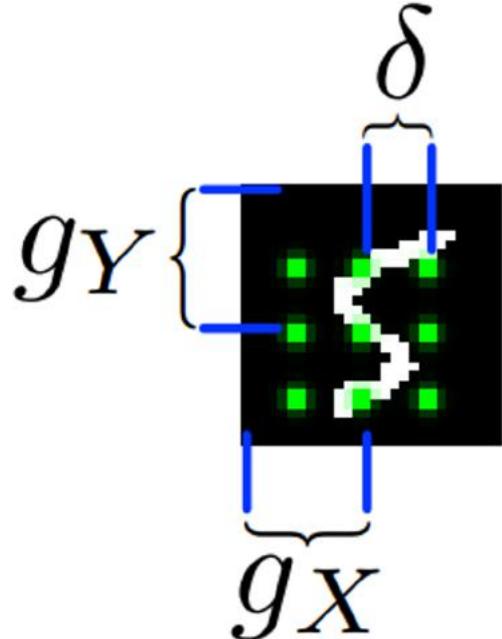
Can hard attention be used for seq2seq? Yes.

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  $3 \times 3$  grid of filters superimposed on an image. The stride ( $\delta$ ) 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 ( $\sigma$ ) of the patches, while the patches themselves are shown to the right. The top patch has a small  $\delta$  and high  $\sigma$ , giving a zoomed-in but blurry view of the centre of the digit; the middle patch has large  $\delta$  and low  $\sigma$ , effectively downampling the whole image; and the bottom patch has high  $\delta$  and  $\sigma$ .

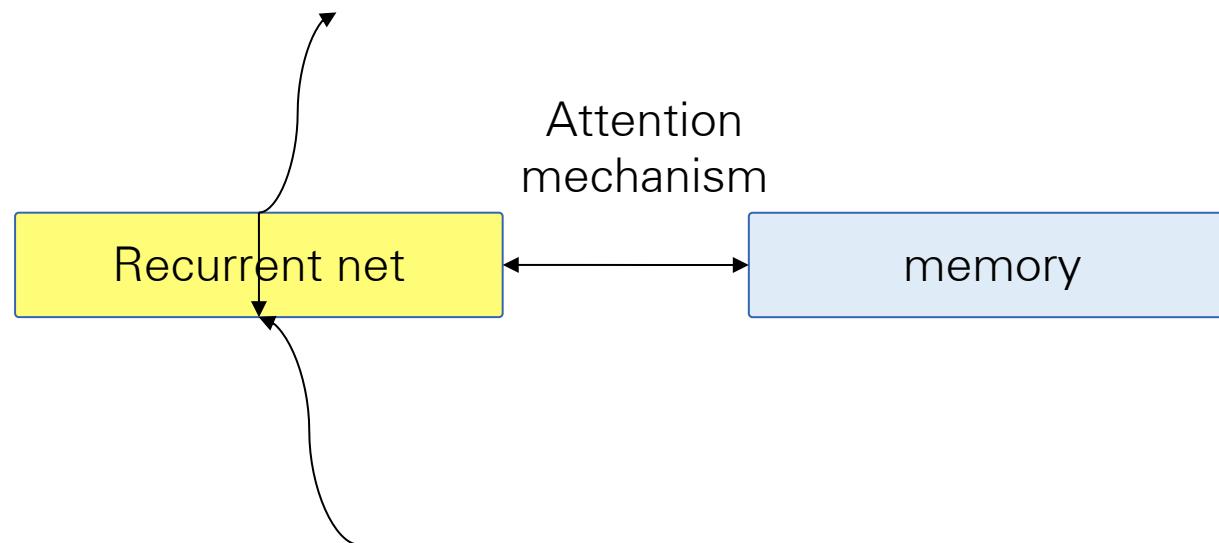


# Why attention?

- Long term memories - attending to memories
  - Dealing with gradient vanishing problem
- Exceeding limitations of a global representation
  - Attending/focusing to smaller parts of data
    - patches in images
    - words or phrases in sentences
- Decoupling representation from a problem
  - Different problems required different sizes of representations
    - LSTM with longer sentences requires larger vectors
- Overcoming computational limits for visual data
  - Focusing only on the parts of images
  - Scalability independent of the size of images
- Adds some interpretability to the models (error inspection)

# Attention on Memory Elements

- Recurrent networks cannot remember things for very long
  - The cortex only remember things for 20 seconds
- We need a “hippocampus” (a separate memory module)
  - LSTM [Hochreiter 1997], registers
  - Memory networks [Weston et 2014] (FAIR), associative memory
  - NTM [Graves et al. 2014], “tape”.



# Recall: Long-Term Dependencies



- The RNN gradient is a product of Jacobian matrices, each associated with a step in the forward computation. To store information robustly in a finite-dimensional state, the dynamics must be contractive [Bengio et al 1994].

$$L = L(s_T(s_{T-1}(\dots s_{t+1}(s_t, \dots))))$$

$$\frac{\partial L}{\partial s_t} = \frac{\partial L}{\partial s_T} \frac{\partial s_T}{\partial s_{T-1}} \dots \frac{\partial s_{t+1}}{\partial s_t}$$

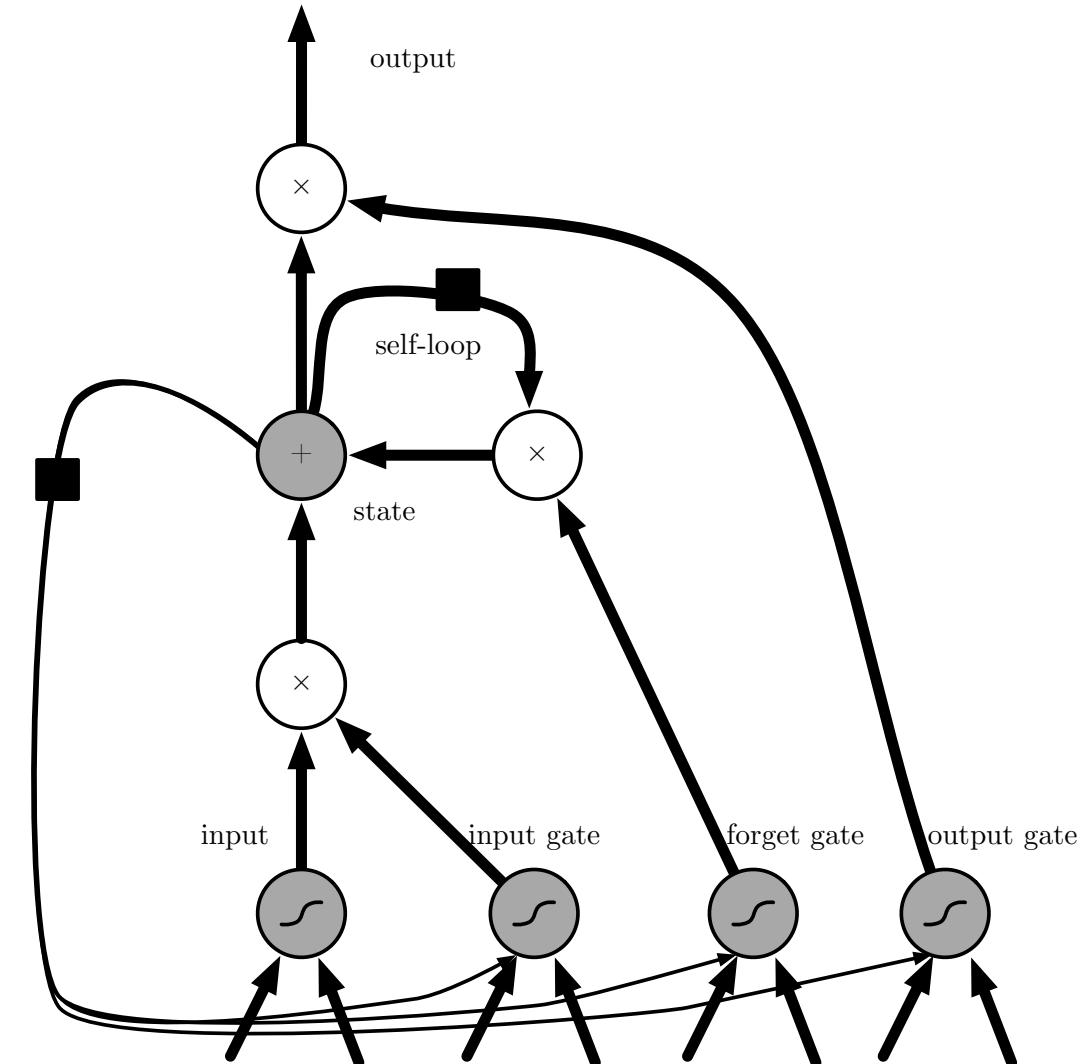
Storing bits  
robustly requires  
sing. values < 1

- Problems:
  - sing. values of Jacobians > 1 → gradients explode
  - or sing. values < 1 → gradients shrink & vanish (Hochreiter 1991)
  - or random → variance grows exponentially

Gradient  
clipping

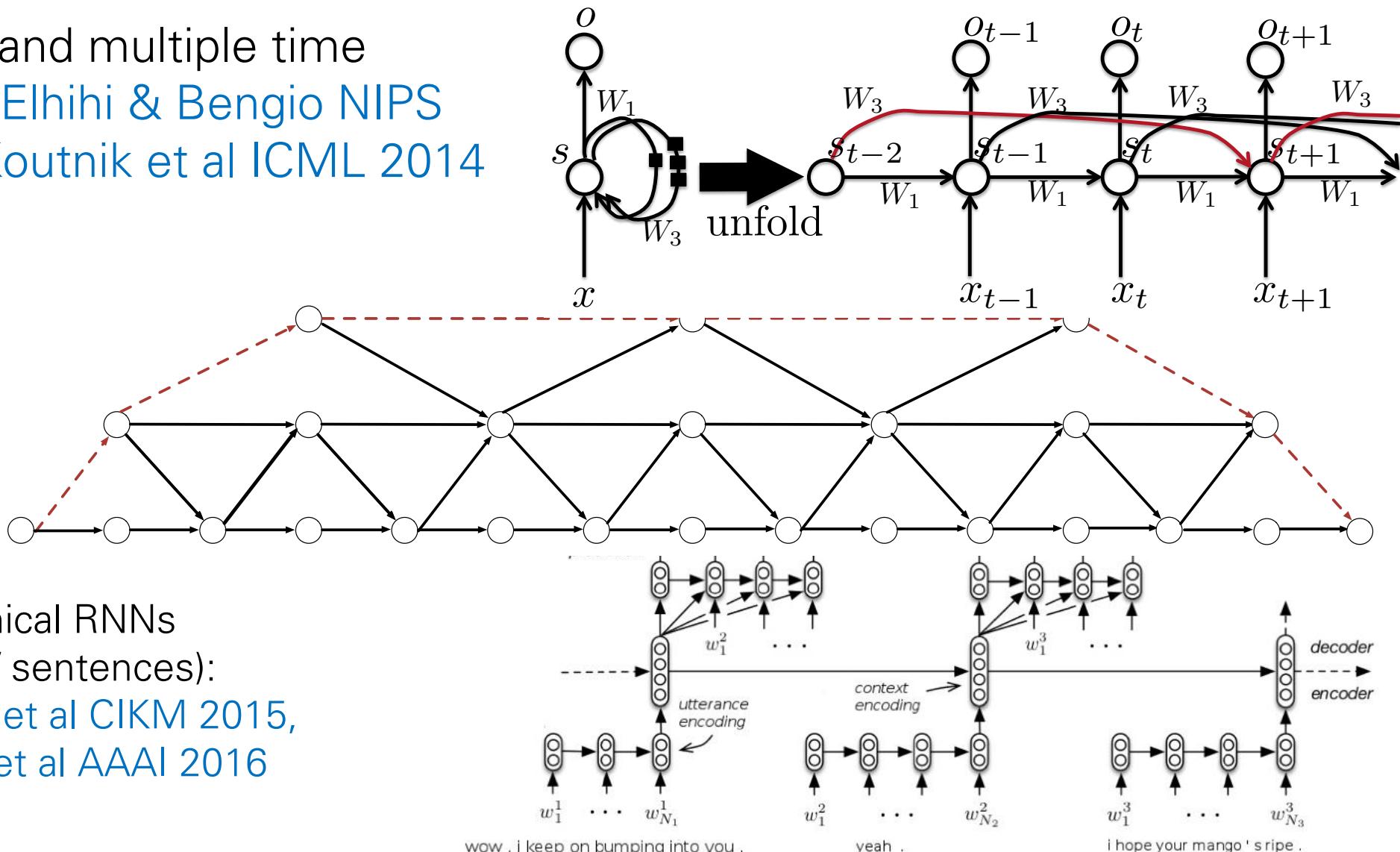
# Gated Recurrent Units & LSTM

- Create a path where gradients can flow for longer with self-loop
- Corresponds to an eigenvalue of Jacobian slightly less than 1
- LSTM is **heavily used**  
(Hochreiter & Schmidhuber 1997)
- GRU light-weight version  
(Cho et al 2014)



# Delays & Hierarchies to Reach Farther

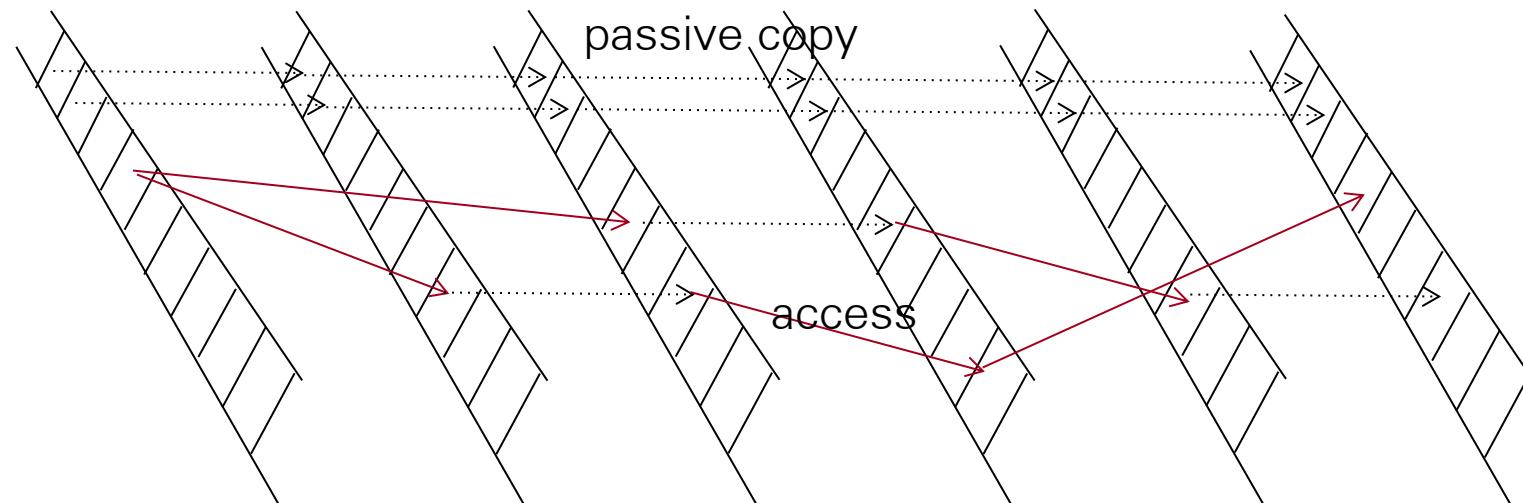
- Delays and multiple time scales, [Elhihi & Bengio NIPS 1995](#), [Koutnik et al ICML 2014](#)



Hierarchical RNNs  
(words / sentences):  
[Sordoni et al CIKM 2015](#),  
[Serban et al AAAI 2016](#)

# Large Memory Networks: Sparse Access Memory for Long-Term Dependencies

- A mental state stored in an external memory can stay for arbitrarily long durations, until evoked for read or write
- Forgetting = vanishing gradient.
- Memory = larger state, avoiding the need for forgetting/vanishing



# Memory Networks

- Class of models that combine large memory with **learning component that can read and write to it.**
- Incorporates **reasoning** with **attention** over **memory** (RAM).
- **Most ML has limited memory** which is more-or-less all that's needed for “low level” tasks e.g. object detection.

Jason Weston, Sumit Chopra, Antoine Bordes. **Memory Networks**. ICLR 2016

S. Sukhbaatar, A. Szlam, J. Weston, R. Fergus. **End-to-end Memory Networks**. NIPS 2015

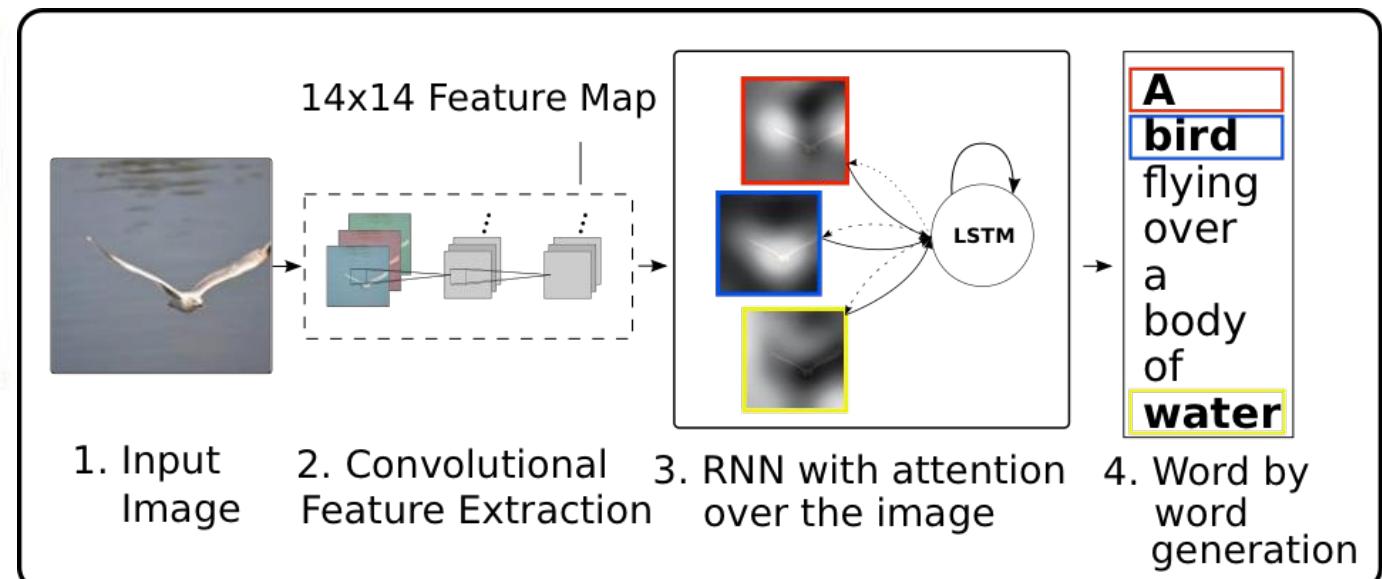
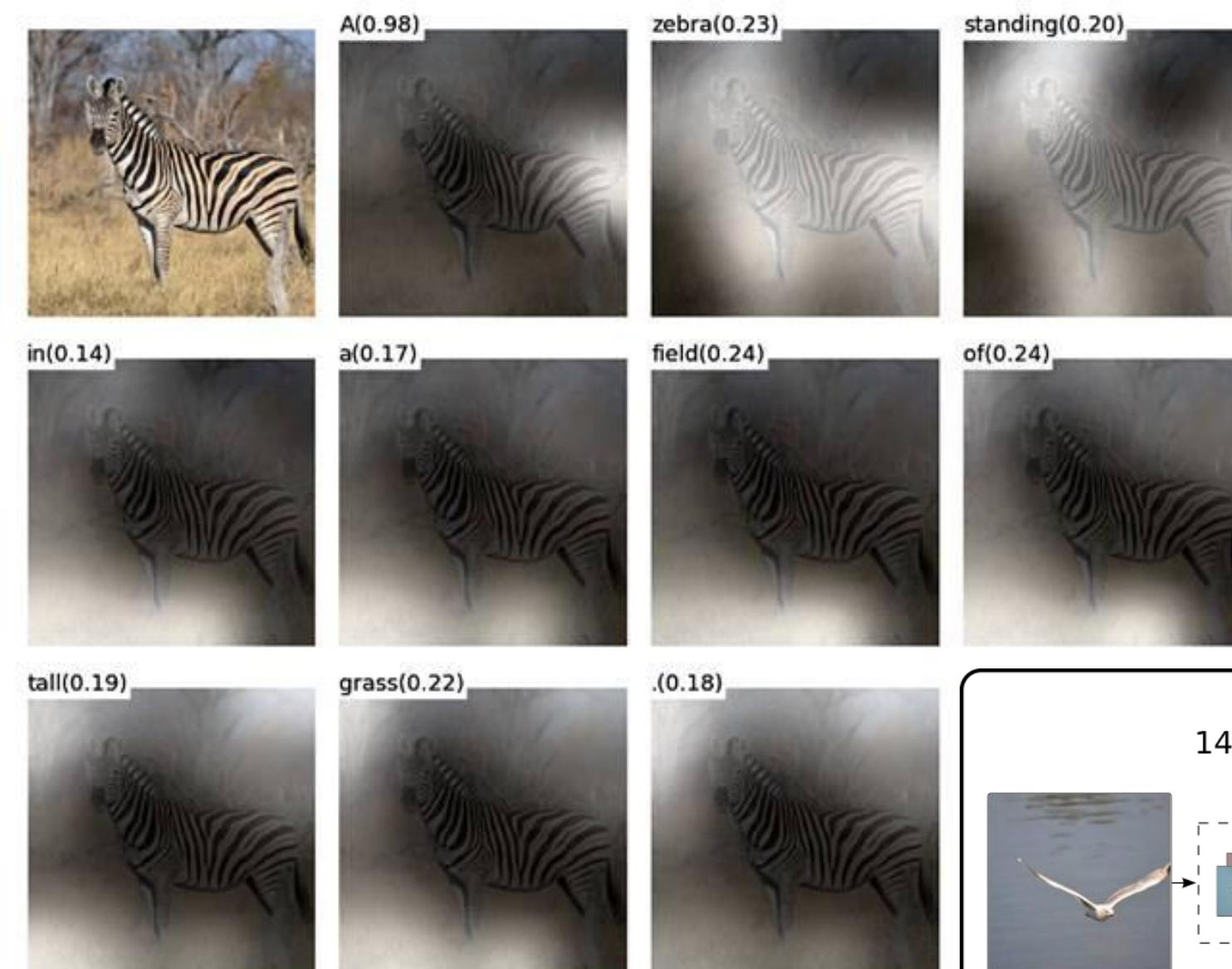
Ankit Kumar et al. **Ask Me Anything: Dynamic Memory Networks for Natural Language Processing**. ICML 2016

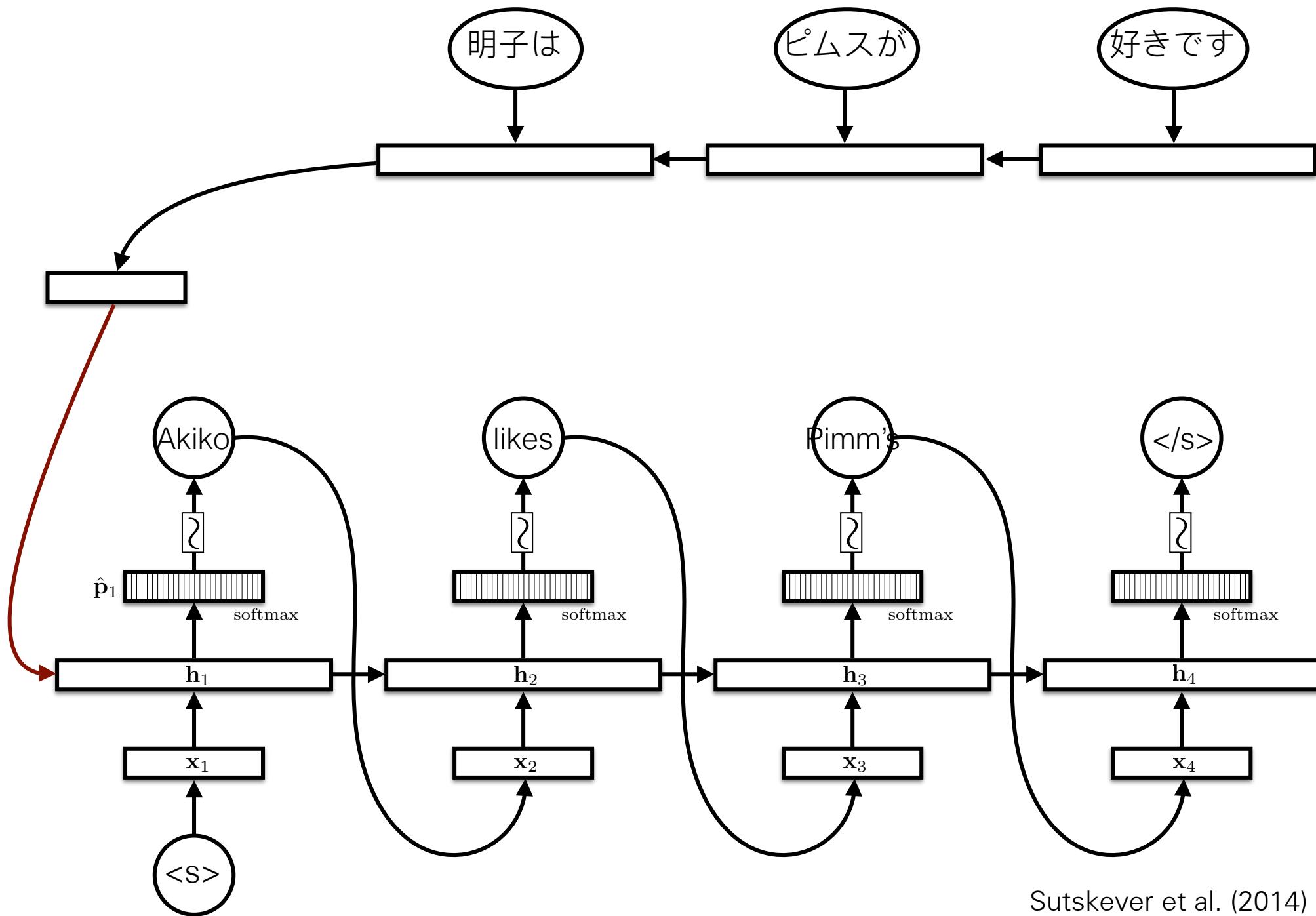
Alex Graves et al. **Hybrid computing using a neural network with dynamic external memory**. Nature, 538(7626): 471–476, 2016.

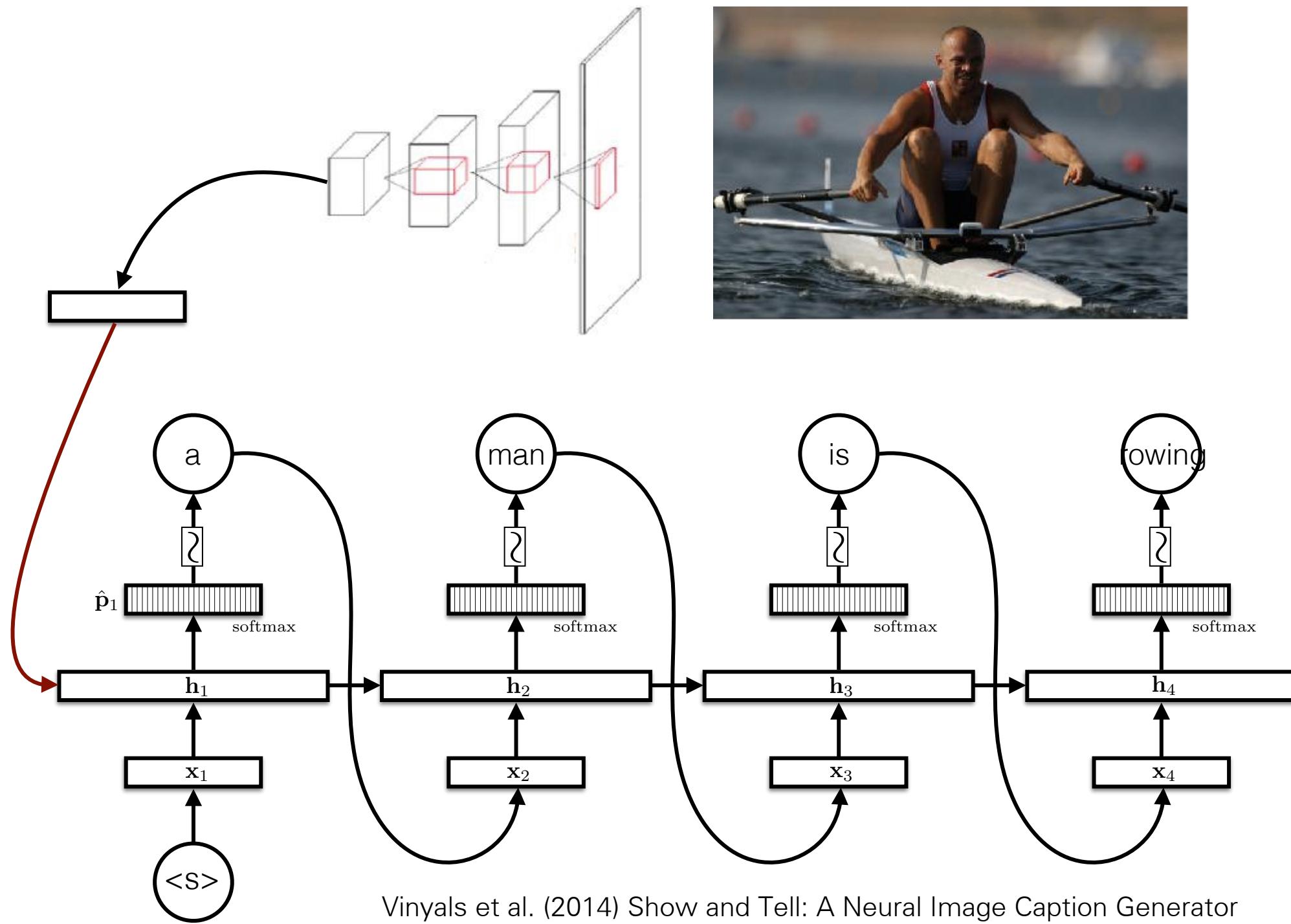
# Case Study: Show, Attend and Tell

Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, Y. Bengio. ICML 2015

# Paying Attention to Selected Parts of the Image While Uttering Words

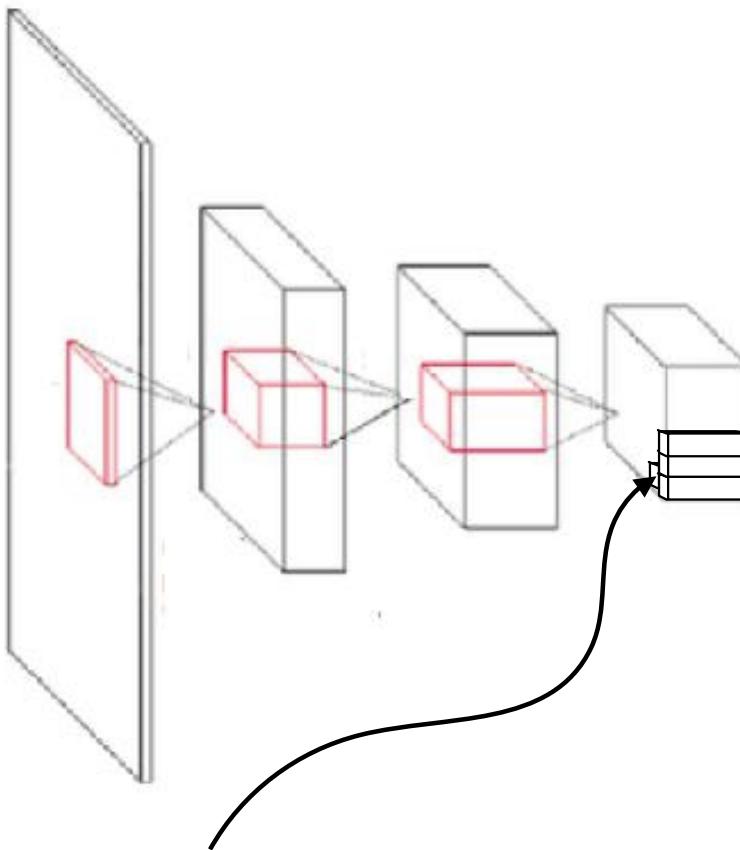






Vinyals et al. (2014) Show and Tell: A Neural Image Caption Generator

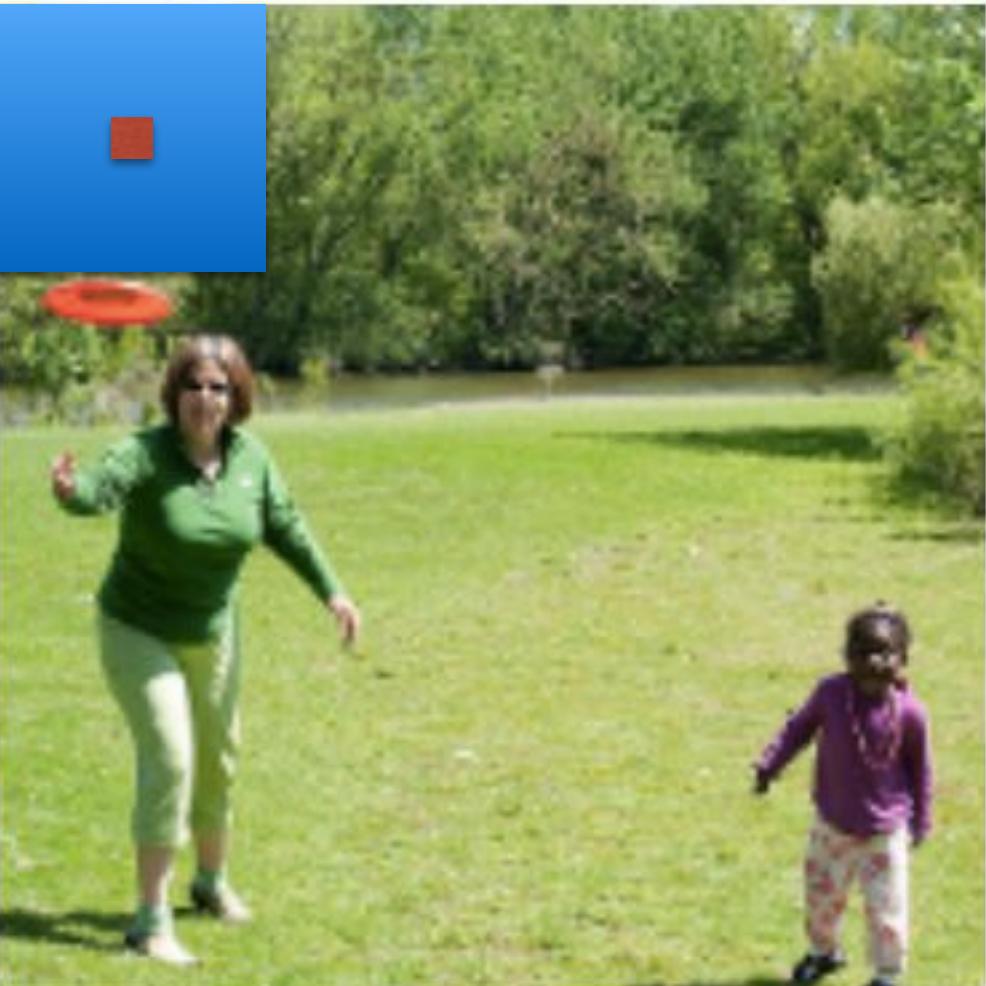
# Regions in ConvNets



- Each point in a “higher” level of a convnet defines spatially localized feature vectors/matrices.
- Xu et al. calls these “annotation vectors”,  $\mathbf{a}_i$ ,  $i \in \{1, \dots, L\}$

# Regions in ConvNets

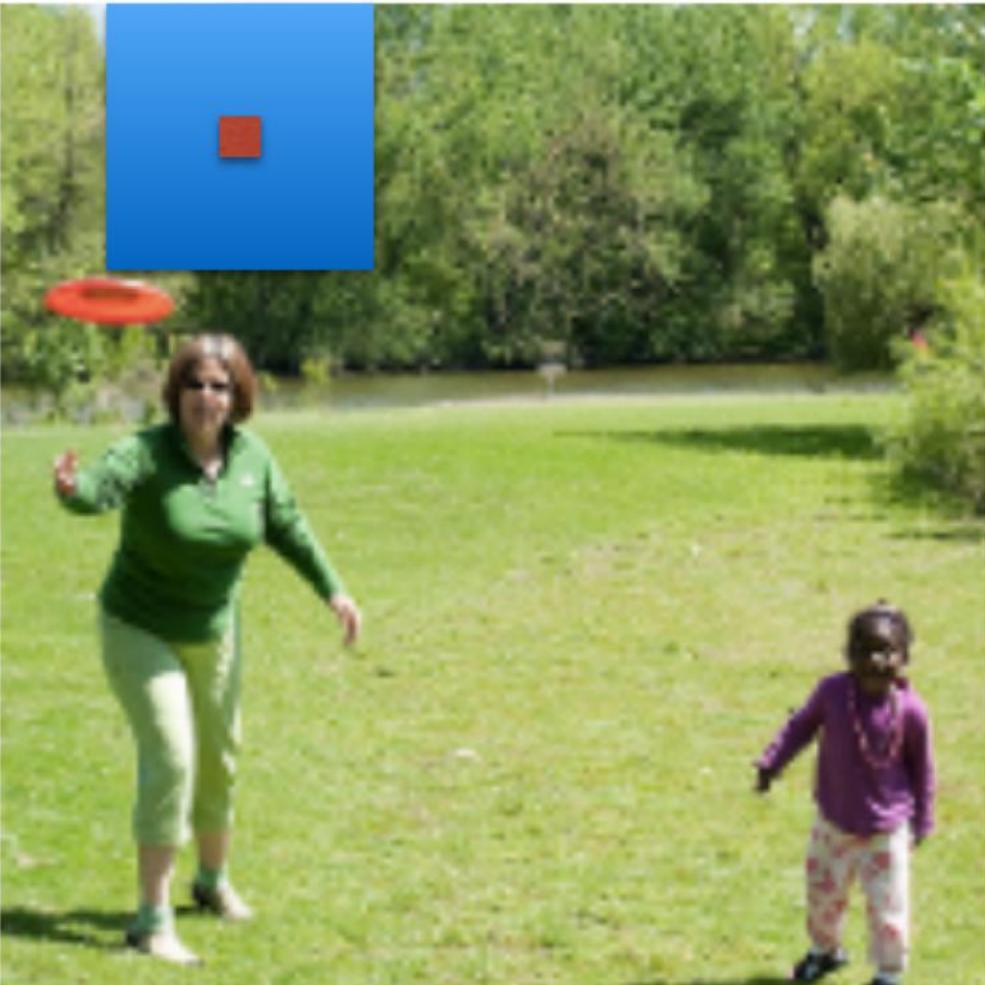
$a_1$



$$F = \begin{bmatrix} | \\ a_1 \\ | \\ \end{bmatrix}$$

# Regions in ConvNets

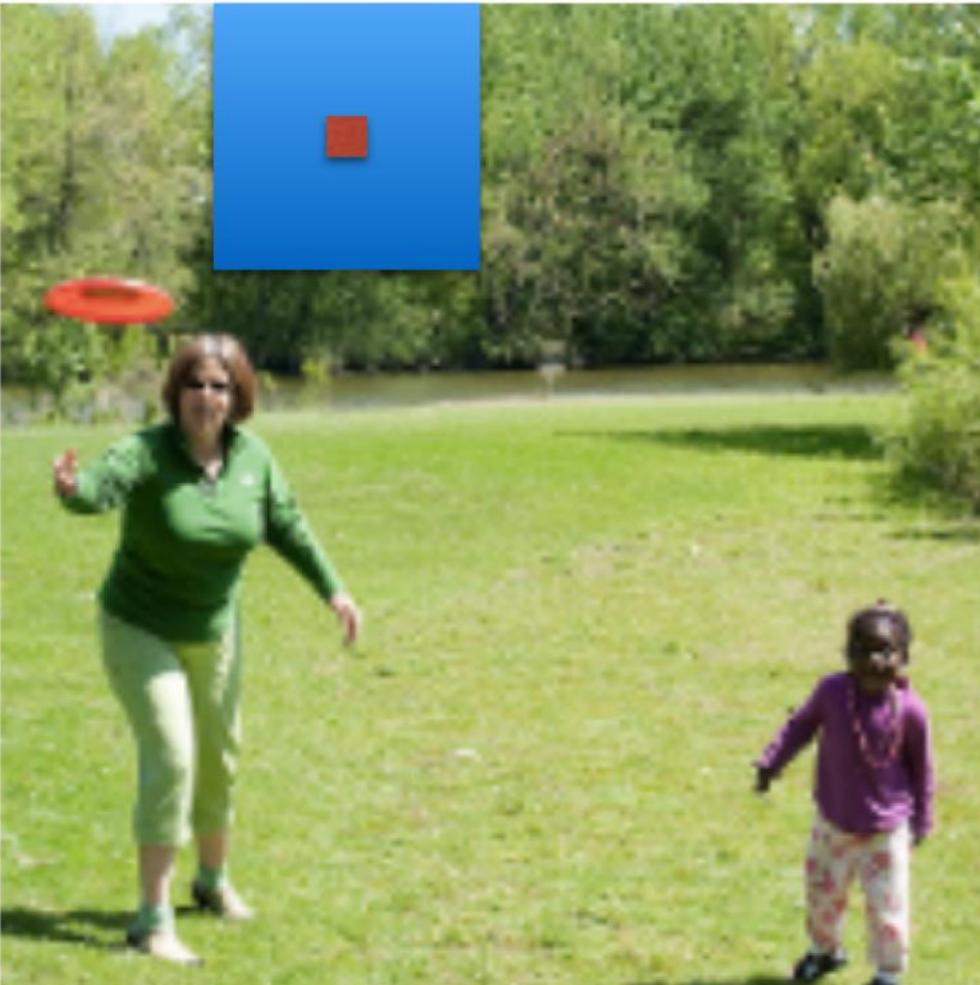
$\mathbf{a}_2$



$$\mathbf{F} = \begin{bmatrix} | & | \\ \mathbf{a}_1 & \mathbf{a}_2 \\ | & | \end{bmatrix}$$

# Regions in ConvNets

$a_3$



$$\mathbf{F} = \begin{bmatrix} | & | & | & \\ \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 & \dots \\ | & | & | & \end{bmatrix}$$

# Extension of LSTM via the context vector

- Extract L D-dimensional annotations
  - Lower convolutional layer to have the correspondence between the feature vectors and portions of the 2-D image

$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \mathbf{g}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} T_{D+m+n, n} \begin{pmatrix} \mathbf{E} \mathbf{y}_{t-1} \\ \mathbf{h}_{t-1} \\ \hat{\mathbf{z}}_t \end{pmatrix} \quad (1)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \quad (2)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t). \quad (3)$$

**E**: embedding matrix

**y**: captions

**h**: previous hidden state

**z**: context vector, a dynamic representation of the relevant part of the image input at time t

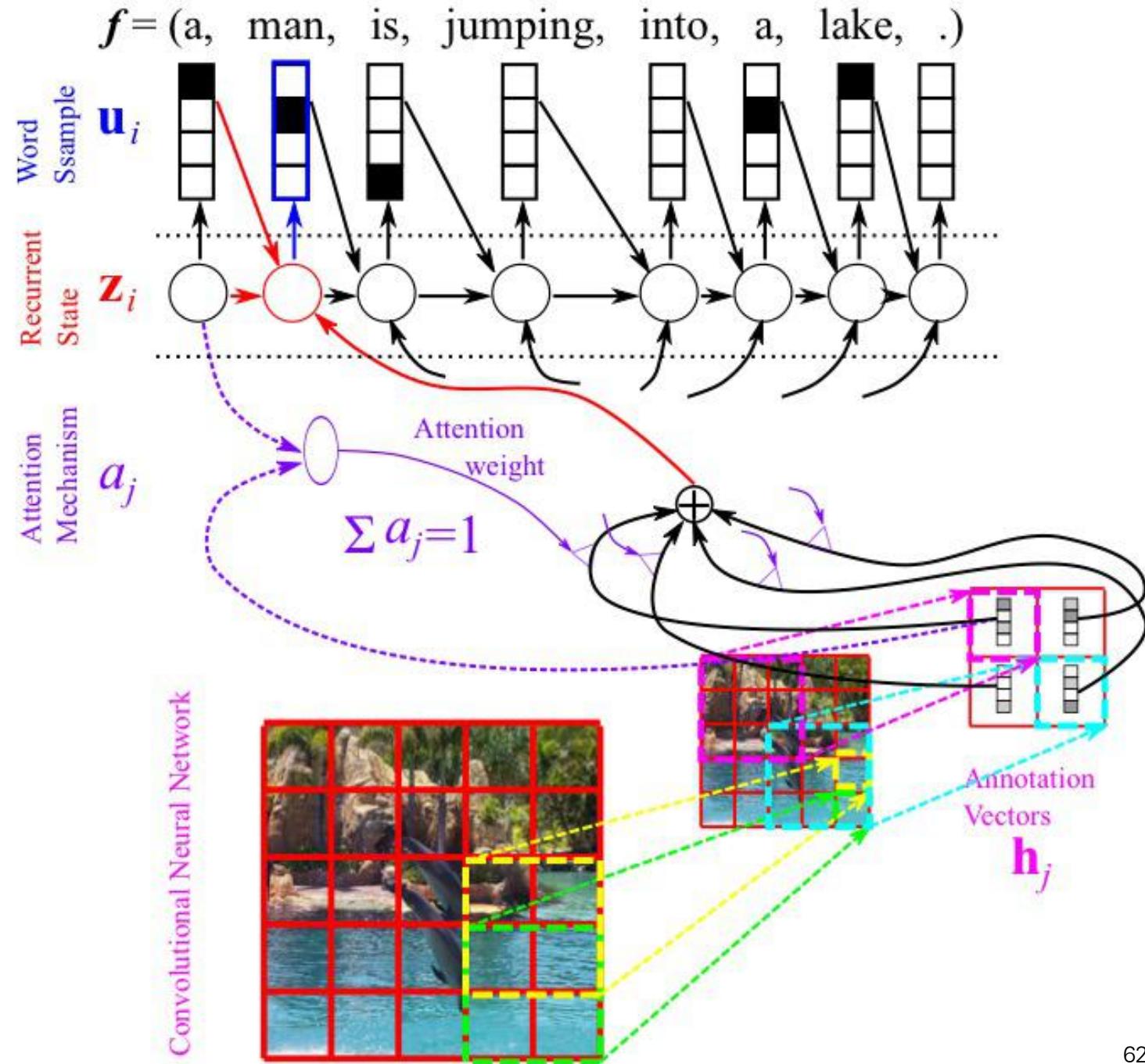
$$e_{ti} = f_{\text{att}}(\mathbf{a}_i, \mathbf{h}_{t-1})$$
$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{k=1}^L \exp(e_{tk})}.$$

A MLP conditioned on the previous hidden state

$$\hat{\mathbf{z}}_t = \phi(\{\mathbf{a}_i\}, \{\alpha_i\}) \quad \phi \text{ is the 'attention' ('focus') function – 'soft' / 'hard'$$

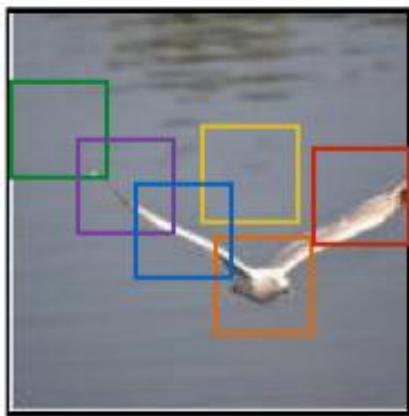
$$p(\mathbf{y}_t | \mathbf{a}, \mathbf{y}_1^{t-1}) \propto \exp(\mathbf{L}_o(\mathbf{E} \mathbf{y}_{t-1} + \mathbf{L}_h \mathbf{h}_t + \mathbf{L}_z \hat{\mathbf{z}}_t))$$

# How soft/hard attention works

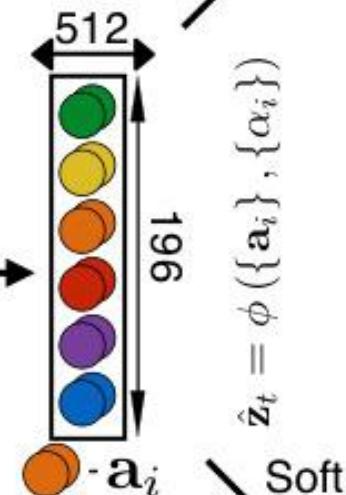


# How soft/hard attention works

A bird flying over a body of water.



conv-512  
conv-512  
maxpool  
 $14 \times 14 \times 512 = 196 \times 512$  (L x D)  
annotations



$$L_s = \sum_s p(s | \mathbf{a}) \log p(\mathbf{y} | s, \mathbf{a})$$

A variational lower bound of maximum likelihood

Sample regions of attention

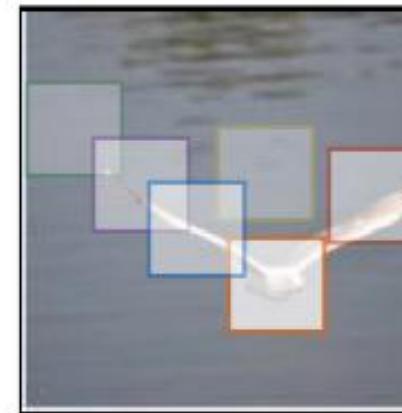
$$\hat{\mathbf{z}}_t = \langle \text{orange}, \text{orange}, \text{red}, \text{blue} \rangle$$

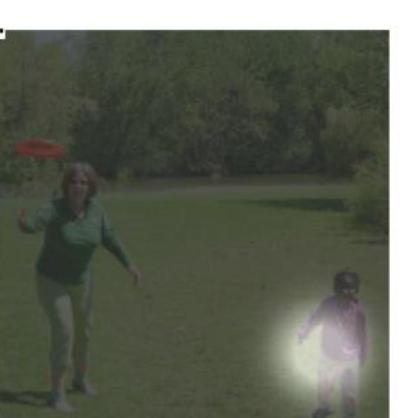
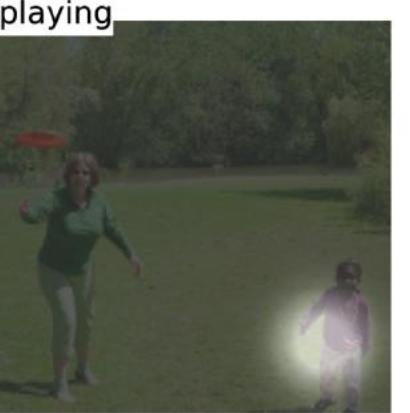
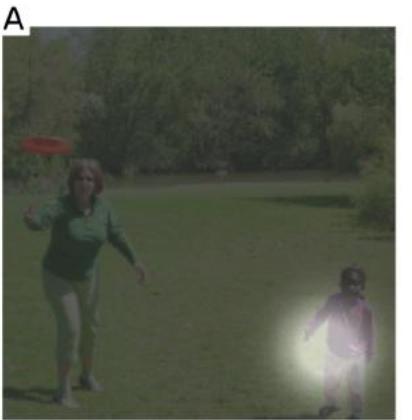


$$L_z = \sum_{z \in \{\text{orange}, \text{orange}, \text{red}, \text{blue}\}} \log p(\mathbf{y} | z)$$

$$\hat{\mathbf{z}}_t = \langle p_1 \ p_2 \ p_3 \ p_4 \ p_5 \ p_6 \rangle, \langle \text{orange}, \text{yellow}, \text{red}, \text{purple}, \text{blue} \rangle \rangle$$

Computes the expected attention





Hard  
Attention



Soft Attention

# The Good



A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



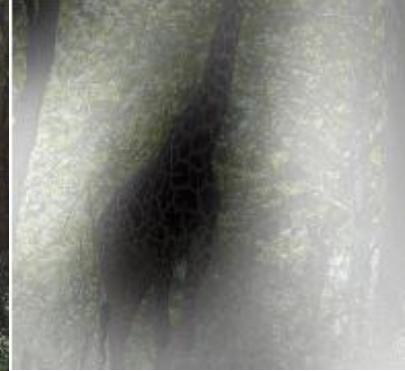
A stop sign is on a road with a mountain in the background.



A little girl sitting on a bed with a teddy bear.



A group of people sitting on a boat in the water.



A giraffe standing in a forest with trees in the background.

# And the Bad



A large white bird standing in a forest.



A woman holding a clock in her hand.



A man wearing a hat and a hat on a skateboard.



A person is standing on a beach with a surfboard.



A woman is sitting at a table with a large pizza.



A man is talking on his cell phone while another man watches.

# Quantitative results

Model	Human		Automatic	
	M1	M2	BLEU	CIDEr
Human	0.638	0.675	0.471	0.91
Google*	0.273	0.317	0.587	0.946
MSR•	0.268	0.322	0.567	0.925
Attention-based*	0.262	0.272	0.523	0.878
Captivator°	0.250	0.301	0.601	0.937
Berkeley LRCN◊	0.246	0.268	0.534	0.891

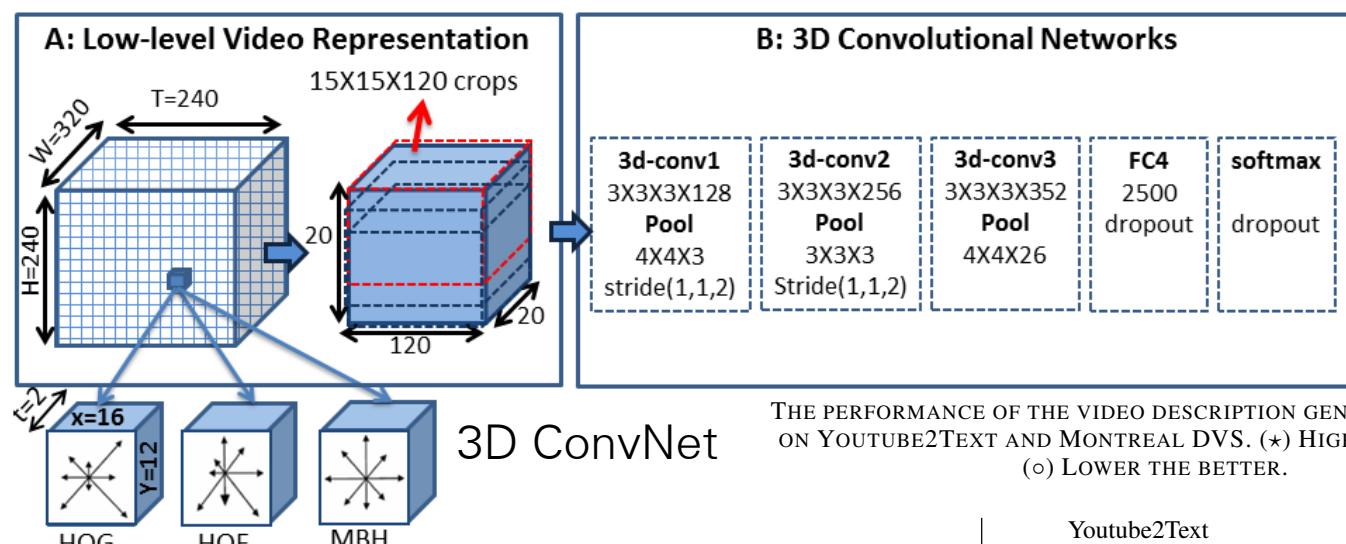
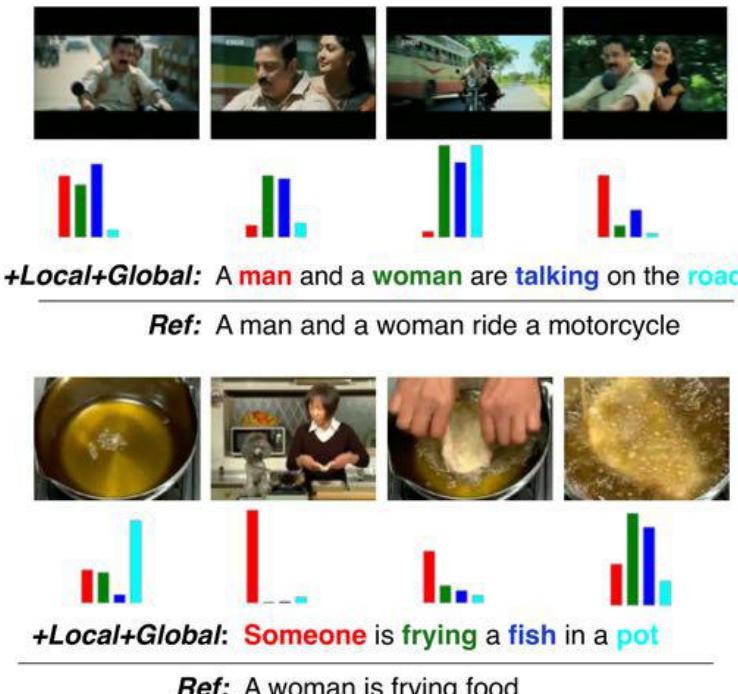
M1: human preferred (or equal) the method over human annotation

M2: turing test

- Add soft attention to image captioning: **+2 BLEU**
- Add hard attention to image captioning: **+4 BLEU**

# Video Description Generation

- Two encoders
  - Context set consists of per-frame context vectors, and attention mechanism that selects one of those vectors for each output symbol being decoded – capturing the global temporal structure across frames
  - 3-D conv-net that applies local filters across spatio-temporal dimensions working on motion statistics
- Both encoders are complementary



THE PERFORMANCE OF THE VIDEO DESCRIPTION GENERATION MODELS ON YOUTUBE2TEXT AND MONTREAL DVS. (\*) HIGHER THE BETTER.  
(○) LOWER THE BETTER.

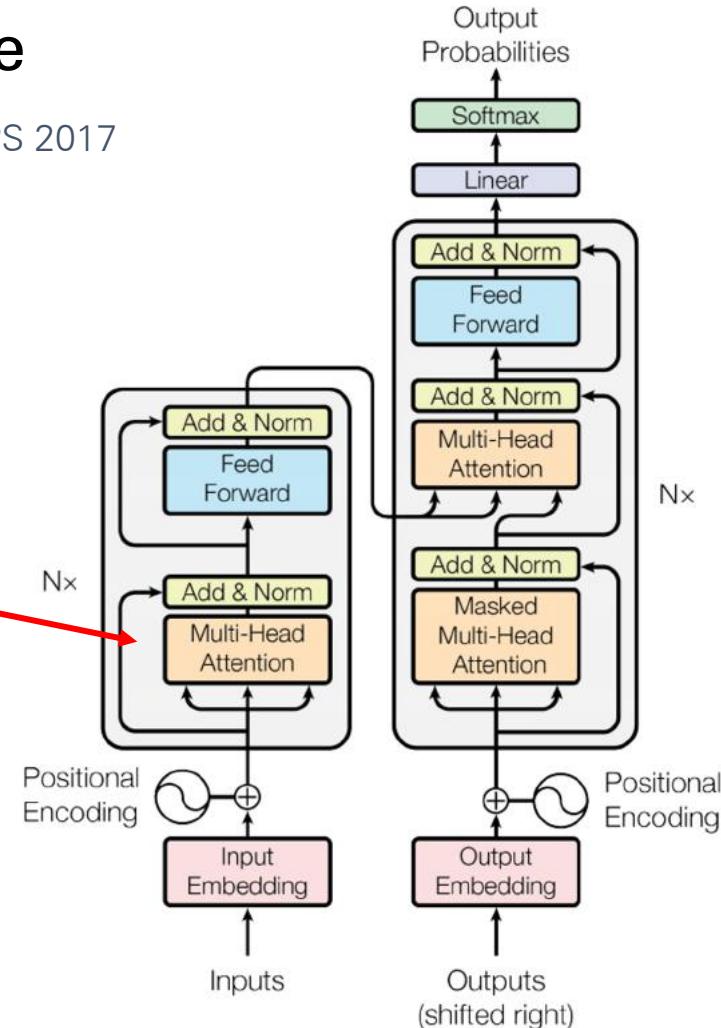
Model	Youtube2Text		Montreal DVS	
	METEOR*	Perplexity°	METEOR	Perplexity
Enc-Dec	0.2868	33.09	0.044	88.28
+ 3-D CNN	0.2832	33.42	0.051	84.41
+ Per-frame CNN	0.2900	27.89	.040	66.63
+ Both	0.2960	27.55	0.057	65.44

# 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!

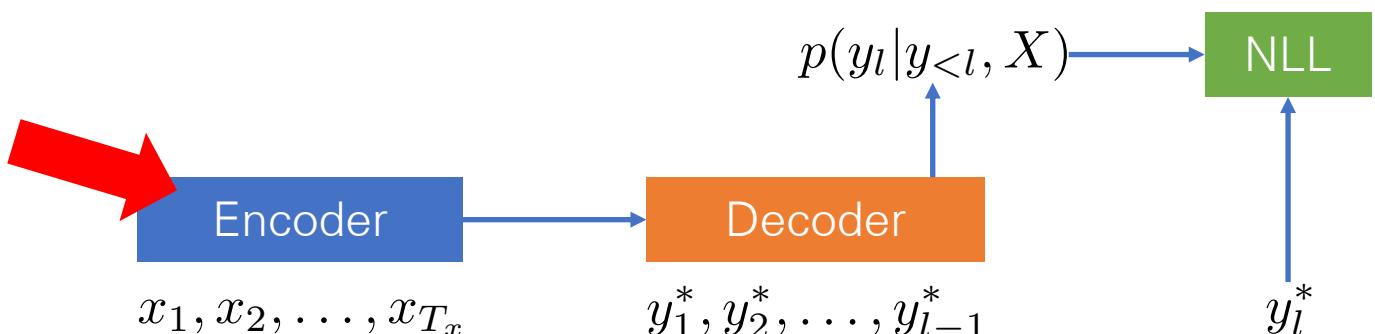


# Parametrization – Recurrent Neural Nets

- Following Bahdanau et al. [2015]
- The encoder turns a sequence of tokens into a sequence of contextualized vectors.

$$h_t = [\vec{h}_t; \overleftarrow{h}_t], \text{ where } \vec{h}_t = \text{RNN}(x_t, \vec{h}_{t-1}), \overleftarrow{h}_t = \text{RNN}(x_t, \overleftarrow{h}_{t+1})$$

- The underlying principle behind recently successful contextualized embeddings
  - ELMo [Peters et al., 2018], BERT [Devlin et al., 2019] and all the other muppets



# Parametrization – Recurrent Neural Nets

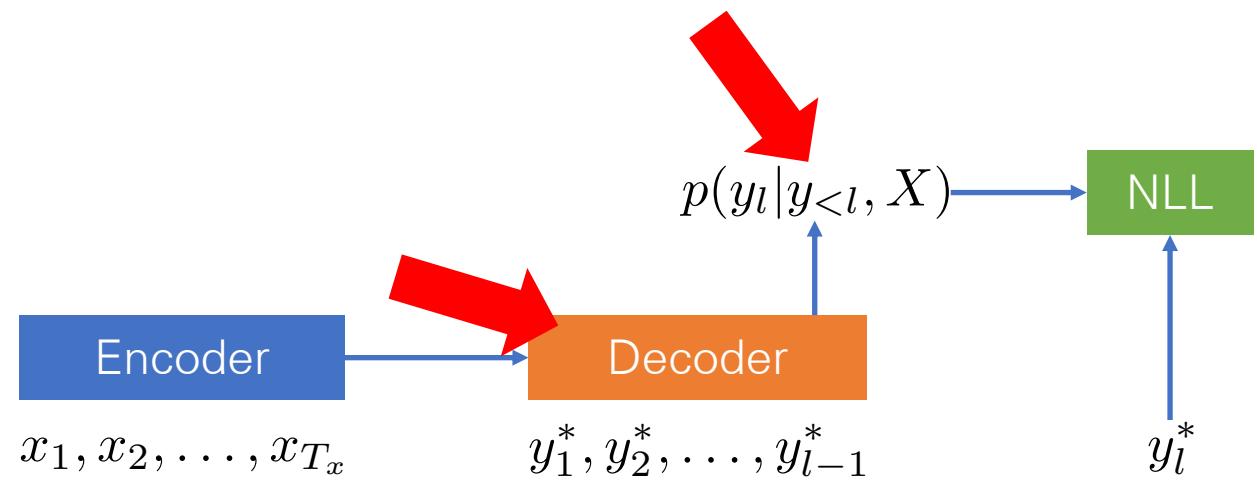
- Following Bahdanau et al. [2015]
- The decoder consists of three stages
  1. Attention: attend to a small subset of source vectors
  2. Update: update its internal state
  3. Predict: predict the next token
- Attention has become the core component in many recent advances
  - Transformers [Vaswani et al., 2017], ...

$$\alpha_{t'} \propto \exp(\text{ATT}(h_{t'}, z_{t-1}, y_{t-1}))$$

$$c_t = \sum_{t'=1}^{T_x} \alpha_{t'} h_{t'}$$

$$z_t = \text{RNN}([y_{t-1}; c_t], z_{t-1})$$

$$p(y_t = v | y_{<t}, X) \propto \exp(\text{OUT}(z_t, v))$$

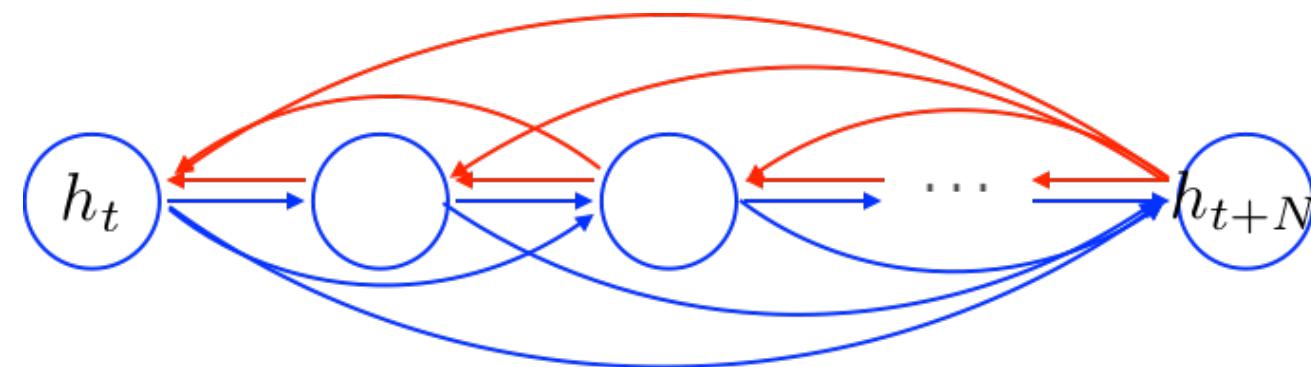


# Side-note: gated recurrent units to attention

- A key idea behind LSTM and GRU is the additive update

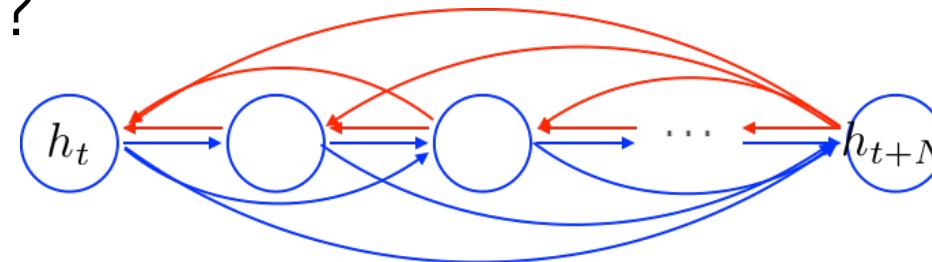
$$h_t = u_t \odot h_{t-1} + (1 - u_t) \odot \tilde{h}_t, \text{ where } \tilde{h}_t = f(x_t, h_{t-1})$$

- This additive update creates linear short-cut connections



# Side-note: gated recurrent units to attention

- What are these shortcuts?



- If we unroll it, we see it's a weighted combination of all previous hidden vectors:

$$\begin{aligned} h_t &= u_t \odot h_{t-1} + (1 - u_t) \odot \tilde{h}_t, \\ &= u_t \odot (u_{t-1} \odot h_{t-2} + (1 - u_{t-1}) \odot \tilde{h}_{t-1}) + (1 - u_t) \odot \tilde{h}_t, \\ &= u_t \odot (u_{t-1} \odot (u_{t-2} \odot h_{t-3} + (1 - u_{t-2}) \odot \tilde{h}_{t-2}) + (1 - u_{t-1}) \odot \tilde{h}_{t-1}) + (1 - u_t) \odot \tilde{h}_t, \\ &\quad \vdots \\ &= \sum_{i=1}^t \left( \prod_{j=i}^{t-i+1} u_j \right) \left( \prod_{k=1}^{i-1} (1 - u_k) \right) \tilde{h}_i \end{aligned}$$

# Side-note: gated recurrent units to attention

1. Can we “free” these dependent weights?

$$h_t = \sum_{i=1}^t \left( \prod_{j=i}^{t-i+1} u_j \right) \left( \prod_{k=1}^{i-1} (1 - u_k) \right) \tilde{h}_i \quad 0$$

2. Can we “free” candidate vectors?

3. Can we separate keys and values?

$$h_t = \sum_{i=1}^t \alpha_i \tilde{h}_i, \text{ where } \alpha_i \propto \exp(\text{ATT}(\tilde{h}_i, x_t)) \quad 1$$

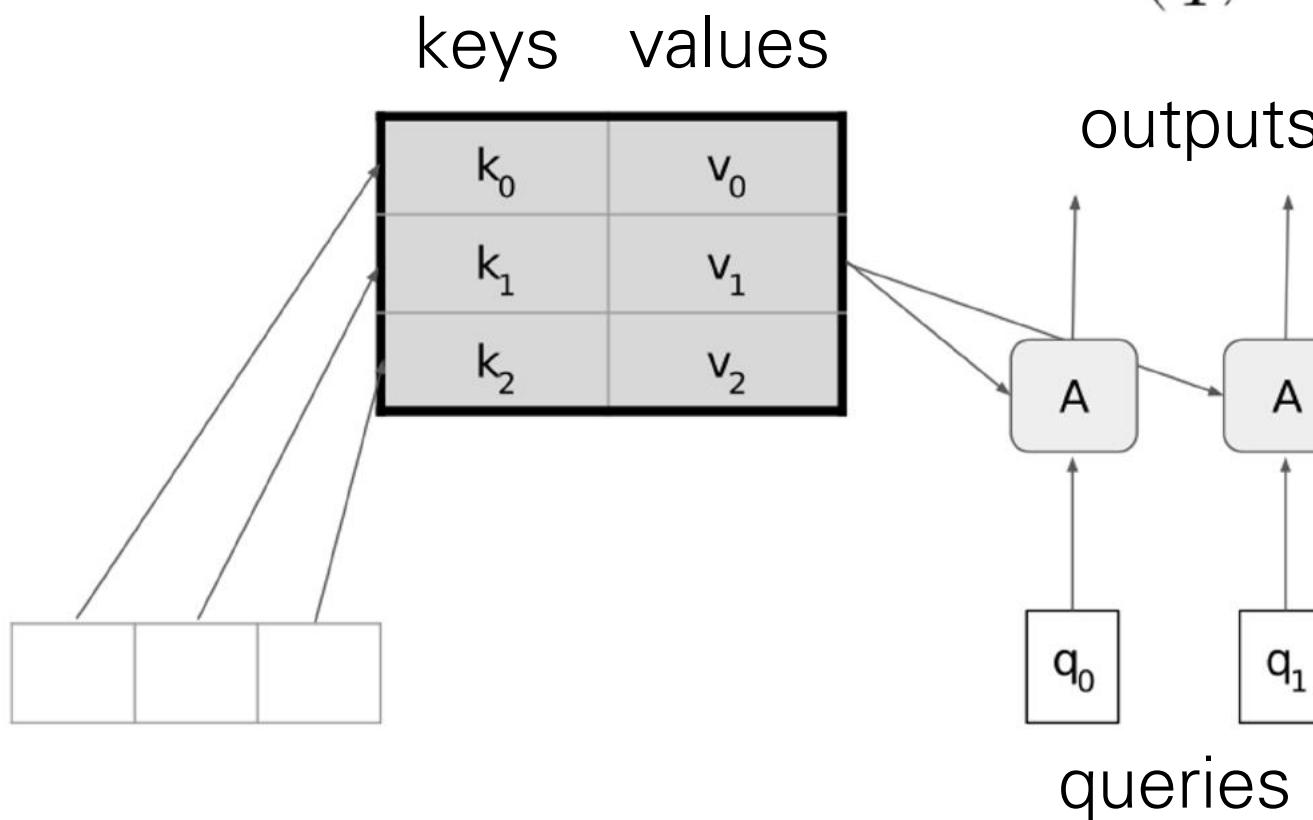
4. Can we have multiple attention heads?

$$h_t = \sum_{i=1}^t \alpha_i f(x_i), \text{ where } \alpha_i \propto \exp(\text{ATT}(f(x_i), x_t)) \quad 2$$

$$h_t = \sum_{i=1}^t \alpha_i V(f(x_i)), \text{ where } \alpha_i \propto \exp(\text{ATT}(K(f(x_i)), Q(x_t))) \quad 3$$

$$h_t = [h_t^1; \dots; h_t^K], \text{ where } h_t^k = \sum_{i=1}^t \alpha_i^k V^k(f(x_i)), \text{ where } \alpha_i^k \propto \exp(\text{ATT}(K^k(f(x_i)), Q^k(x_t))) \quad 4$$

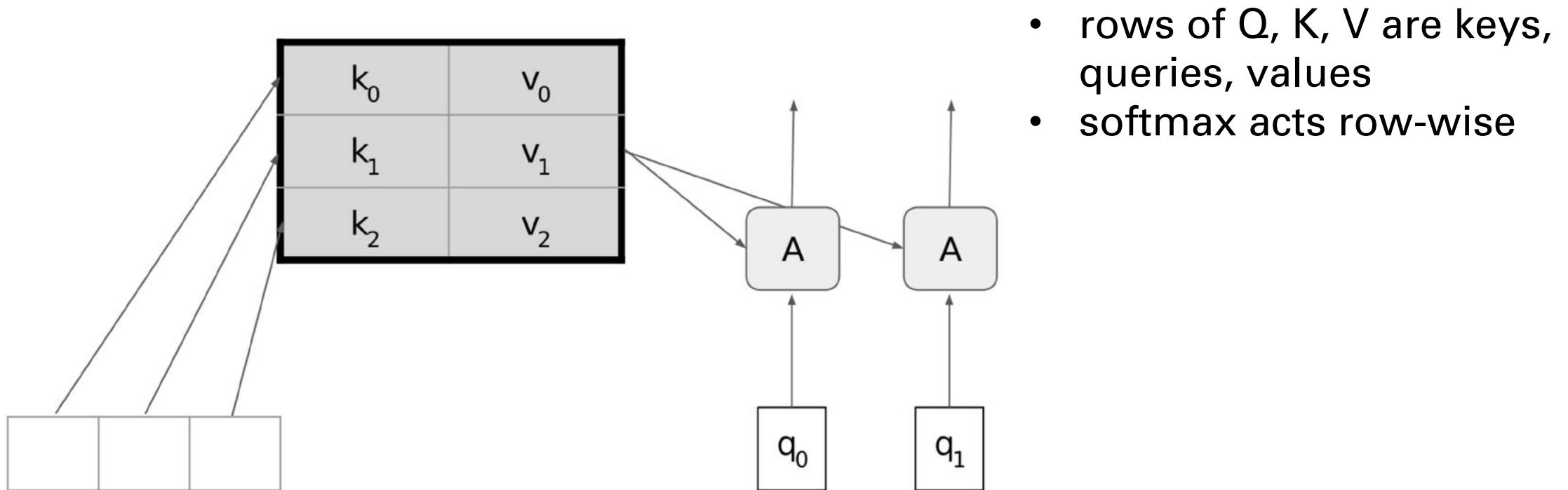
# Generalized dot-product attention - vector form



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

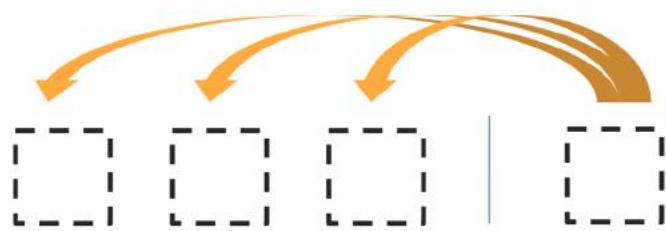
# Generalized dot-product attention - matrix form

$$A(Q, K, V) = \text{softmax}(QK^T)V$$

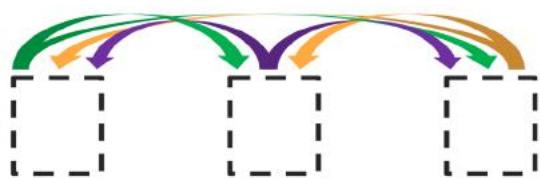


# Three types of attention in Transformer

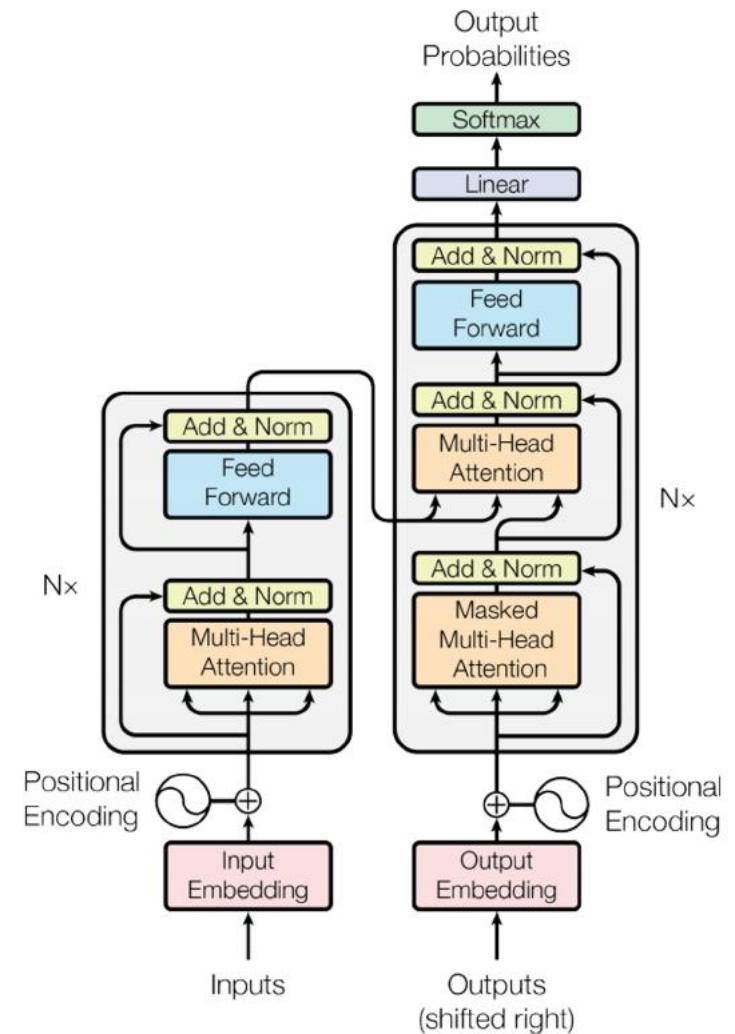
- usual attention between encoder and decoder:  
 $Q=[\text{current state}] \ K=V=[\text{BiRNN states}]$



- self-attention in the encoder (encoder attends to itself!)  
 $Q=K=V=[\text{encoder states}]$



- masked self-attention in the decoder (attends to itself, but a state can only attend previous states)  
 $Q=K=V=[\text{decoder states}]$



# Other tricks in Transformer

- allows different processing of information coming from different locations

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

$$\text{where } \text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

- positional embeddings are required to preserve the order information:

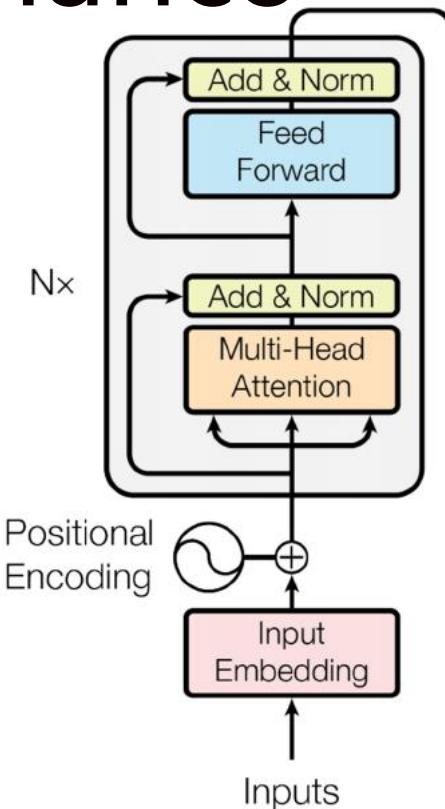
$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$

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

(trainable parameter embeddings also work)

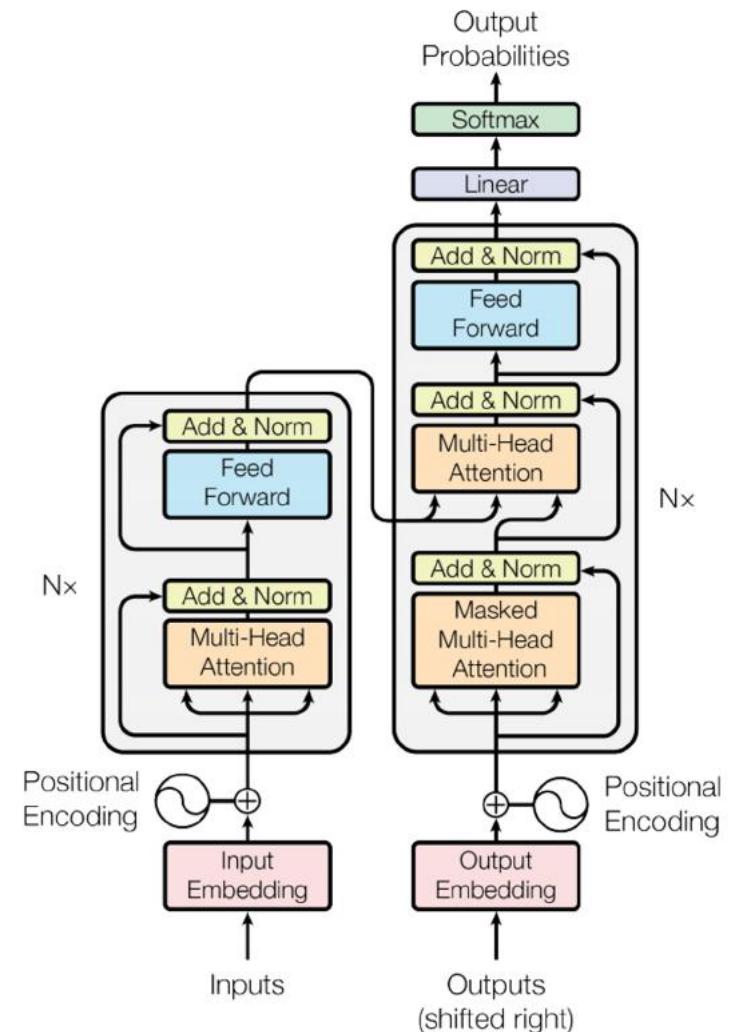
# Transformer Full Model and Performance

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	<b>41.29</b>	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1		<b><math>3.3 \cdot 10^{18}</math></b>
Transformer (big)	<b>28.4</b>	<b>41.8</b>		$2.3 \cdot 10^{19}$



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

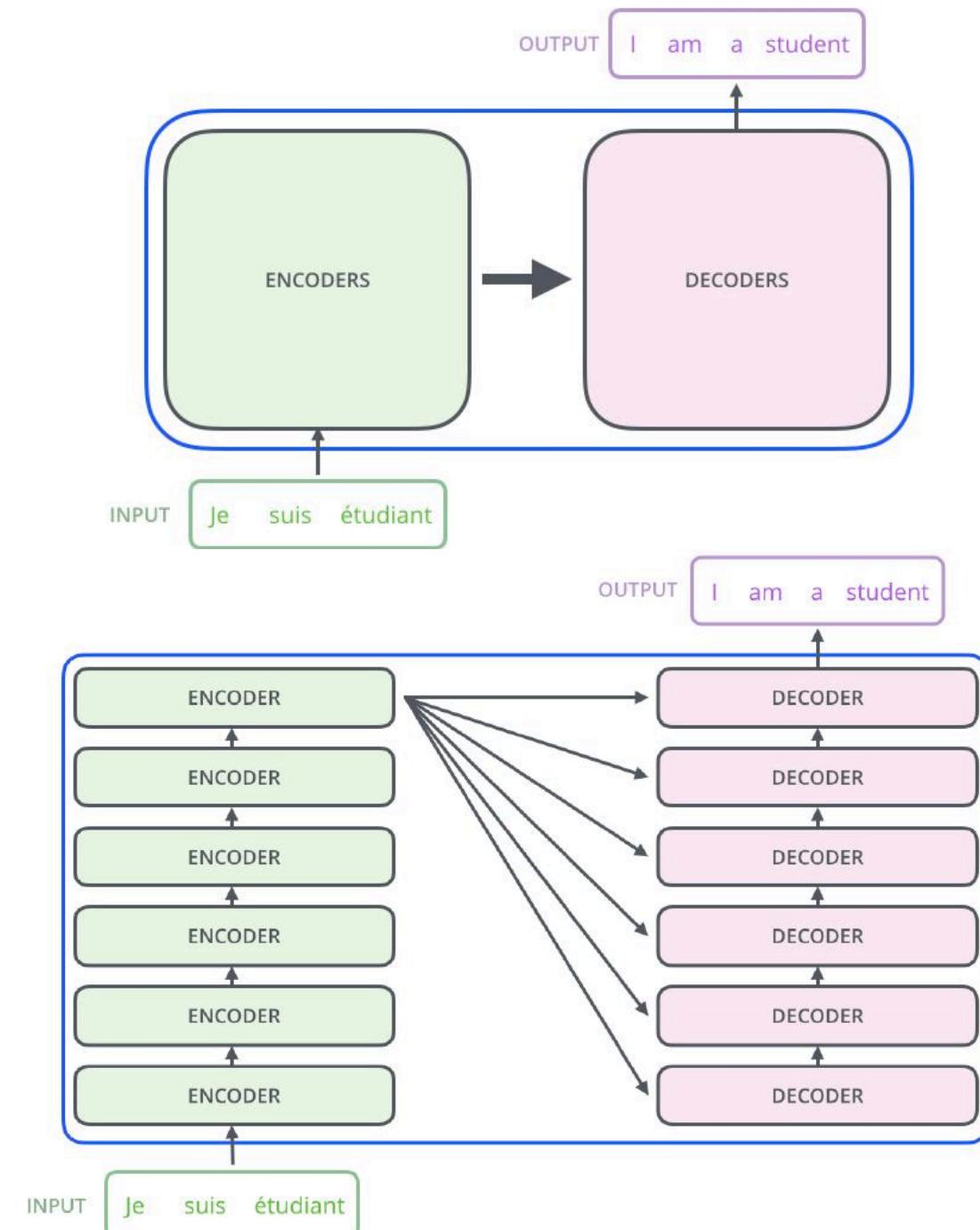
# Transformers for Text



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

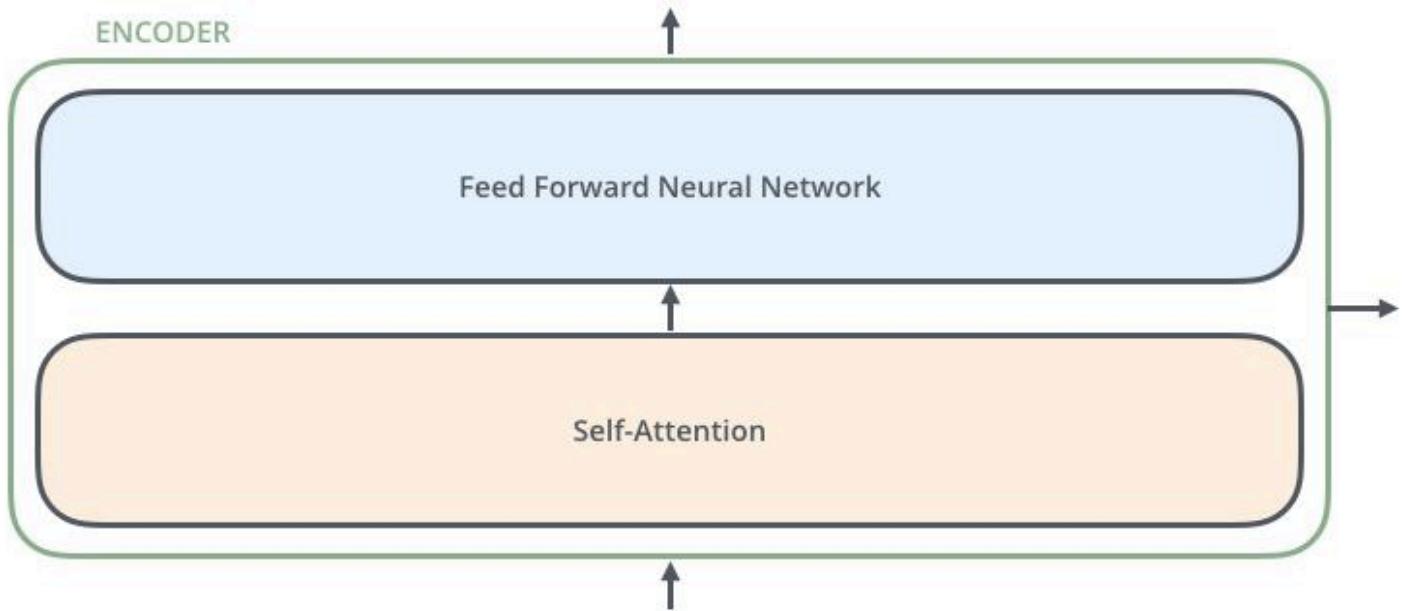
# Transformer Model

- It is a sequence to sequence model (from the original paper)
- the encoder component is a stack of encoders (6 in this paper)
- the decoding component is also a stack of decoders of the same number

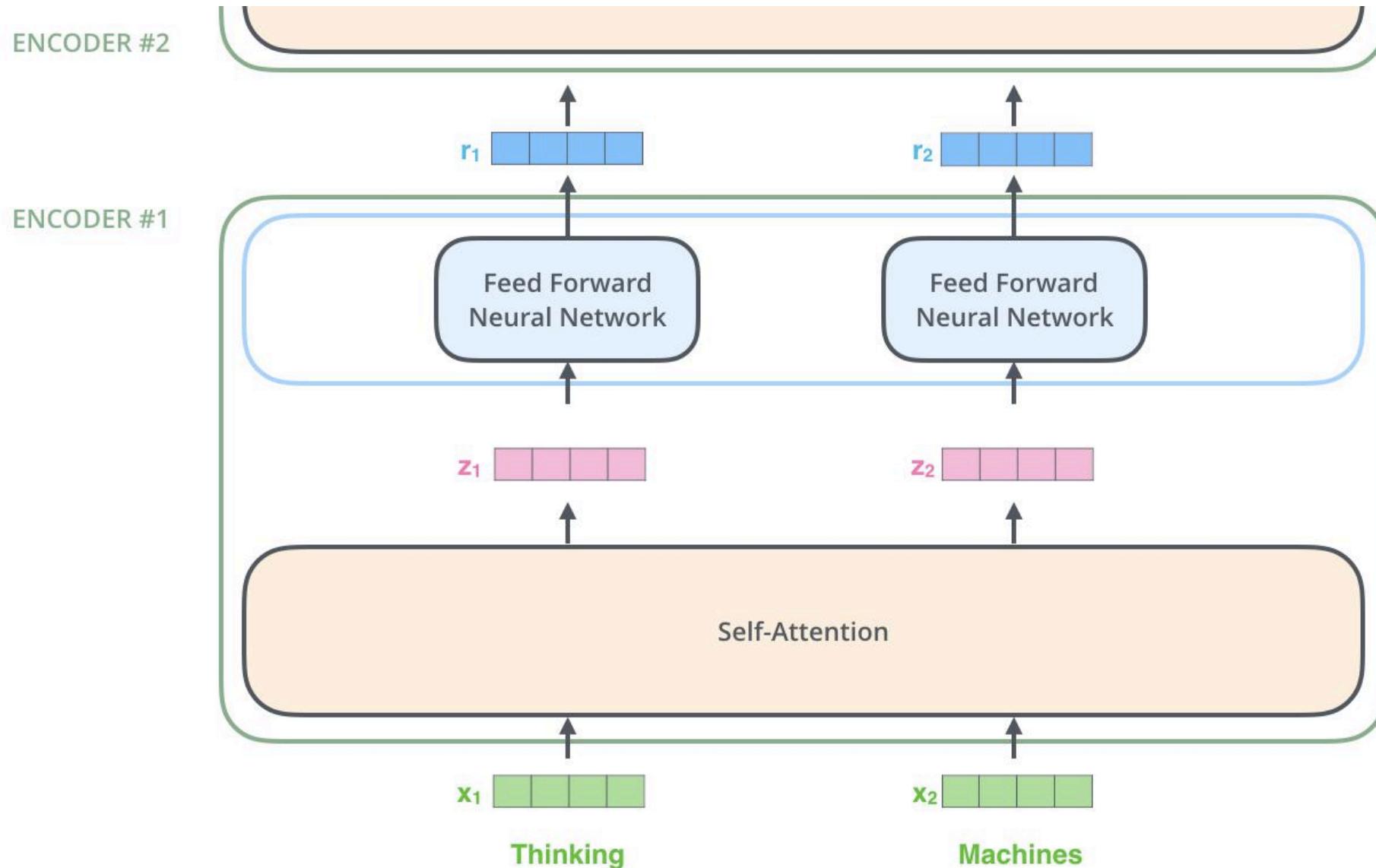


# Transformer Model: Encoder

- The encoder can be broken down into 2 parts

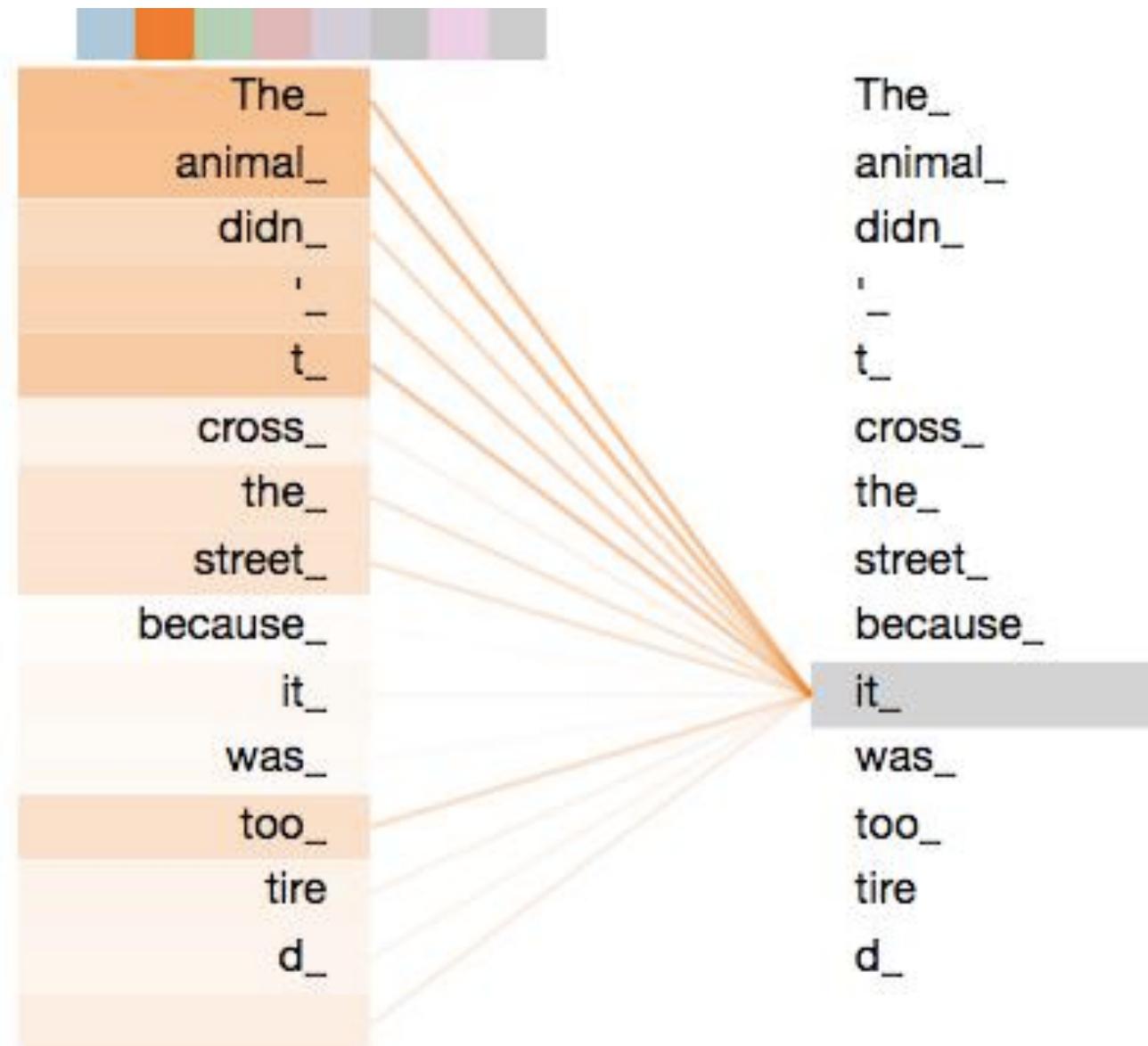


# Transformer Model: Encoder



# Transformer Model: Encoder

- Example: “The animal didn't cross the street because it was too tired”
  - Associate “it” with “animal”
  - look for clues when encoding



# Self-Attention: Step 1 (Create Vectors)

Input

Thinking

Machines

Embedding

$X_1$

$X_2$

Queries

$q_1$

$q_2$

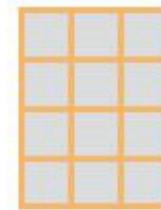


$W^Q$

Keys

$k_1$

$k_2$

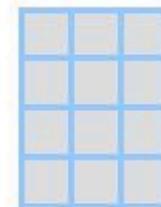


$W^K$

Values

$v_1$

$v_2$



$W^V$

- Abstractions useful for calculating and thinking about attention

# Self-Attention: Step 2 (Calculate score), 3 and 4

Input	<b>Thinking</b>	
Embedding	$x_1$	
Queries	$q_1$	
Keys	$k_1$	
Values	$v_1$	
Score	$q_1 \cdot k_1 = 112$	
Divide by 8 ( $\sqrt{d_k}$ )	14	
Softmax	0.88	
	<b>Machines</b>	
	$x_2$	
	$q_2$	
	$k_2$	
	$v_2$	
	$q_1 \cdot k_2 = 96$	
	12	
	0.12	

# Self-Attention: Step 5

- multiply each value vector by the softmax score
- sum up the weighted value vectors
- produces the output

Input

Embedding

Queries

Keys

Values

Score

Divide by 8 ( $\sqrt{d_k}$ )

Softmax

Softmax

X

Value

Sum

Thinking

$x_1$  

$q_1$  

$k_1$  

$v_1$  

$$q_1 \cdot k_1 = 112$$

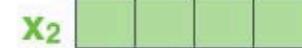
14

0.88

$v_1$  

$z_1$  

Machines

$x_2$  

$q_2$  

$k_2$  

$v_2$  

$$q_1 \cdot k_2 = 96$$

12

0.12

$v_2$  

$z_2$  

# Self-Attention: Matrix Form

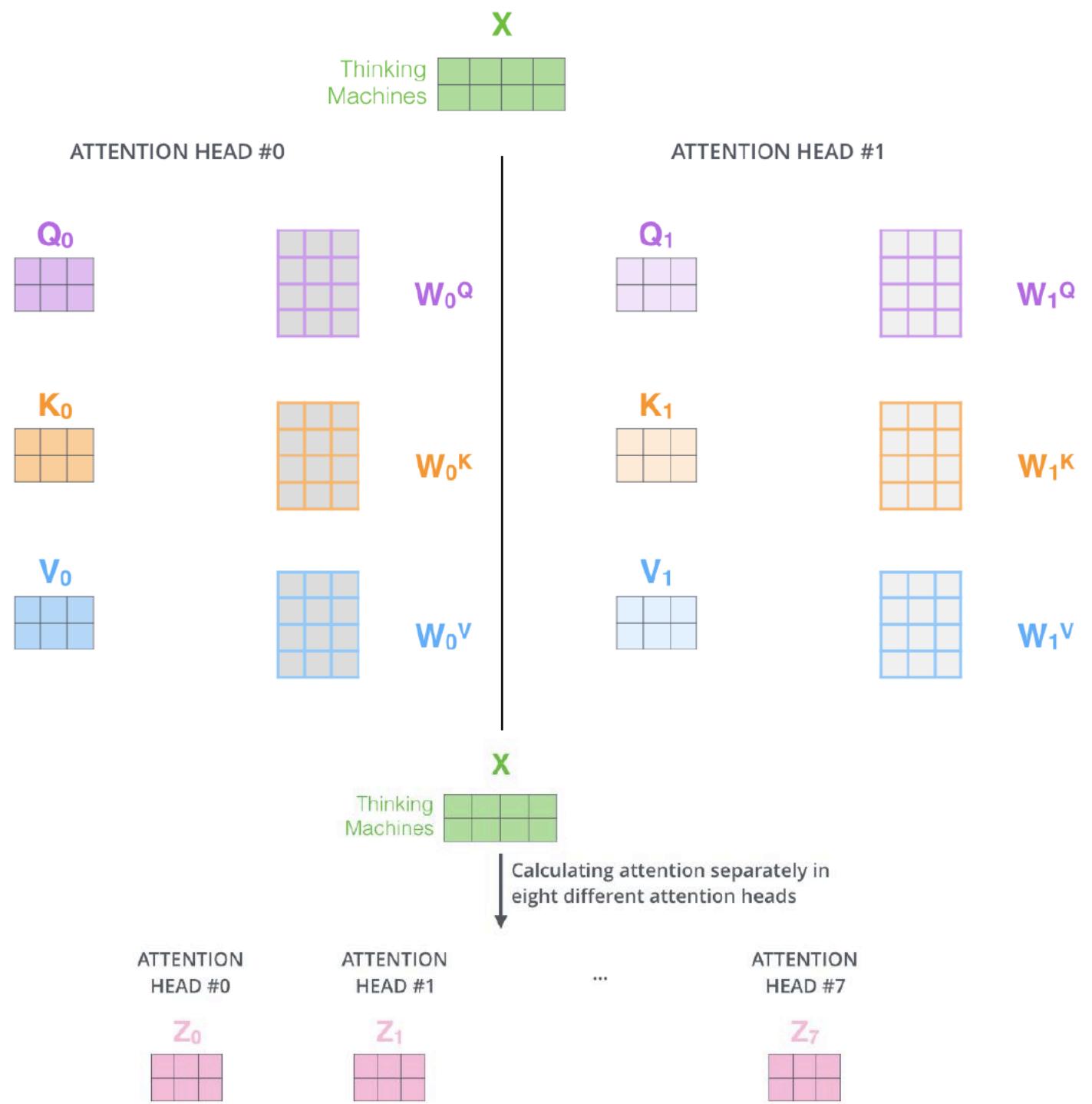
$$\text{softmax}\left(\frac{\mathbf{Q} \times \mathbf{K}^T}{\sqrt{d_k}}\right) \mathbf{V} = \mathbf{Z}$$

The diagram illustrates the matrix form of self-attention. It shows the computation of attention weights from query  $\mathbf{Q}$  and key  $\mathbf{K}^T$ , and their application to value  $\mathbf{V}$  to produce output  $\mathbf{Z}$ . The matrices are represented as 2x3 grids:

- $\mathbf{Q}$  (Query) is a 2x3 grid of purple squares.
- $\mathbf{K}^T$  (Key Transpose) is a 3x2 grid of orange squares.
- $\mathbf{V}$  (Value) is a 2x3 grid of blue squares.
- $\mathbf{Z}$  (Output) is a 2x3 grid of pink squares.

The multiplication operation  $\mathbf{Q} \times \mathbf{K}^T$  is shown as a horizontal line connecting the two matrices, indicating that each row of  $\mathbf{Q}$  is multiplied by each column of  $\mathbf{K}^T$ .

# Self-Attention: Multiple Heads



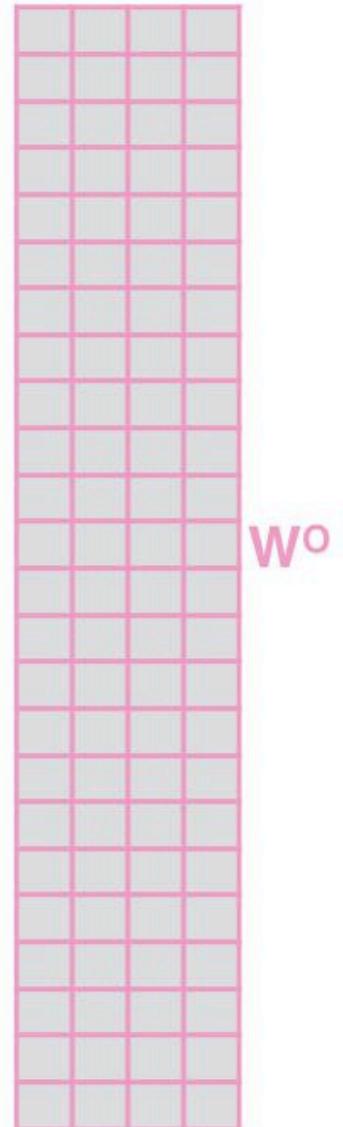
# Self-Attention: Multiple Heads

1) Concatenate all the attention heads



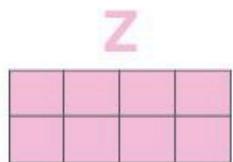
2) Multiply with a weight matrix  $W^o$  that was trained jointly with the model

$x$

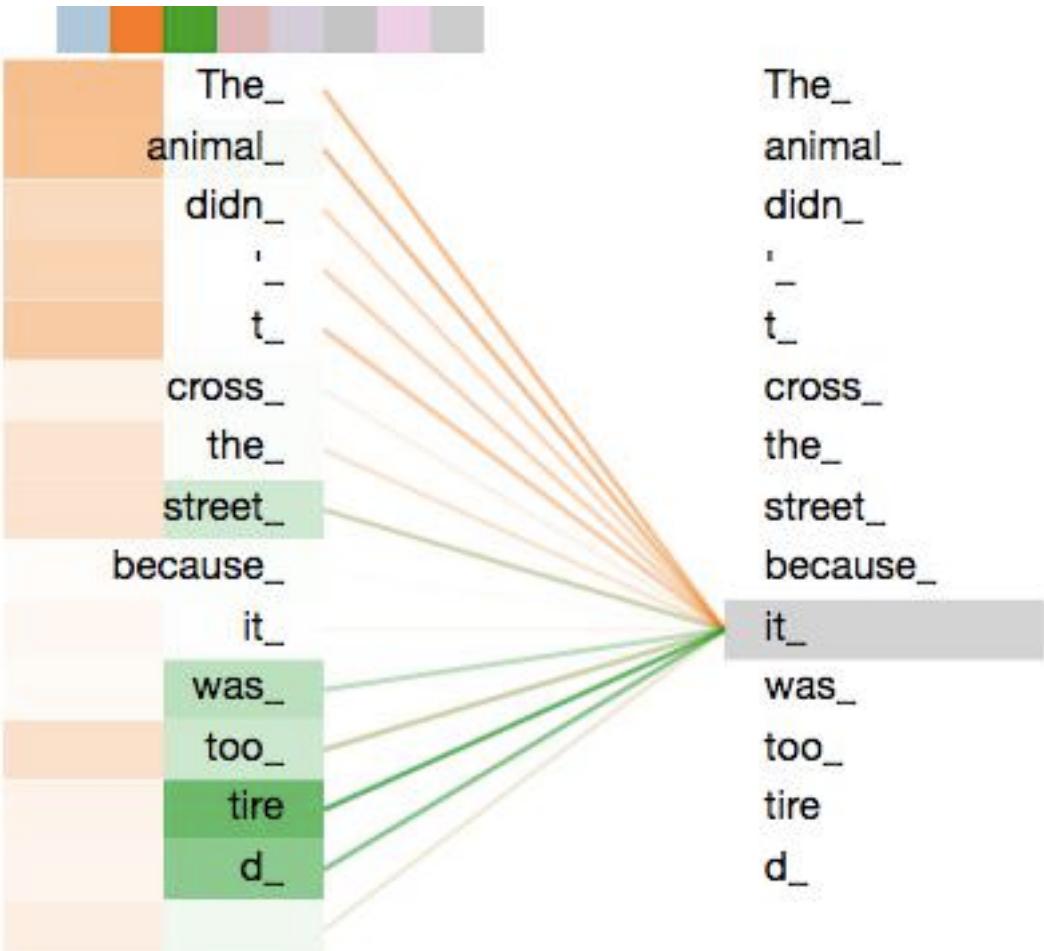


3) The result would be the  $Z$  matrix that captures information from all the attention heads. We can send this forward to the FFNN

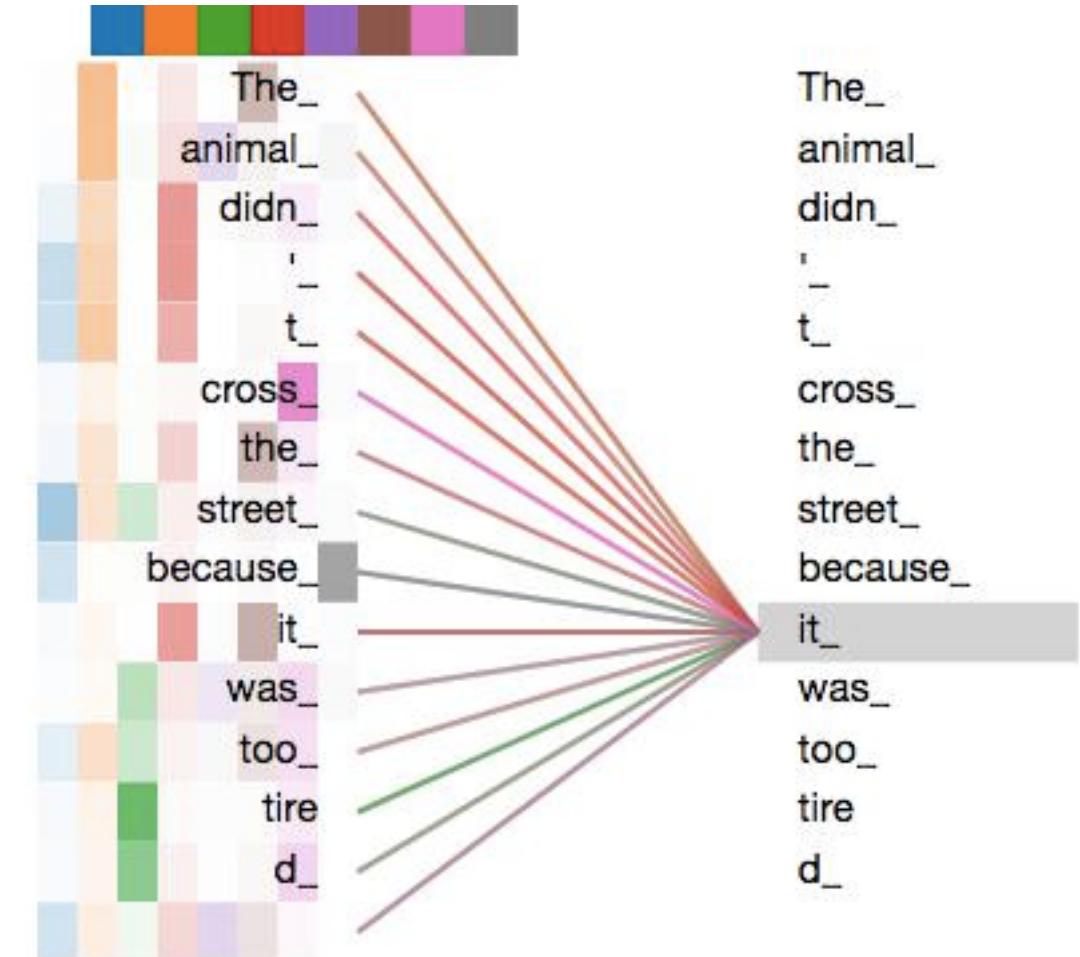
=



# Self-Attention: Multiple Heads



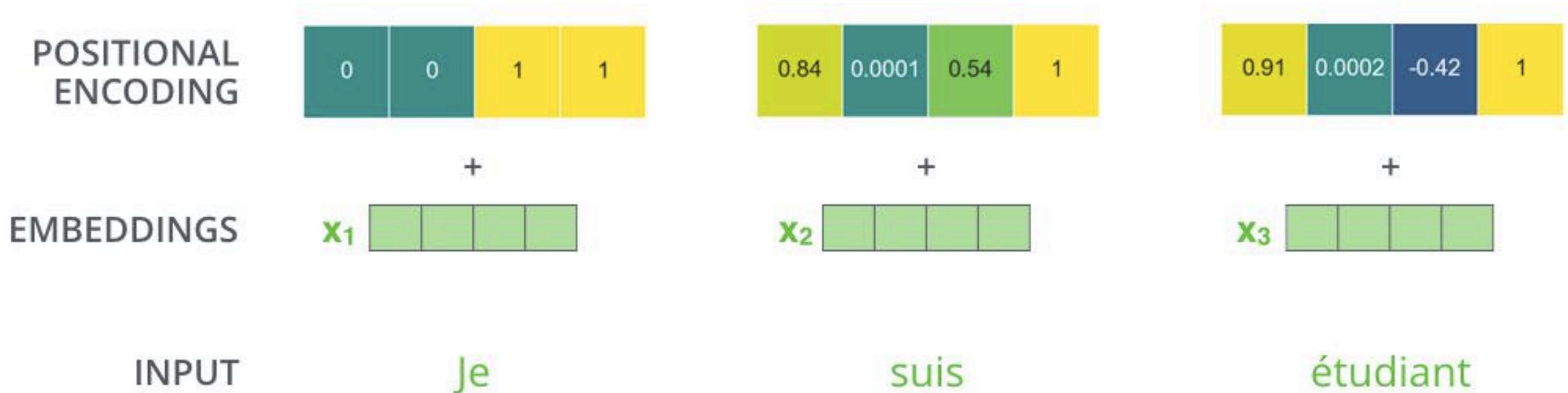
- Where different attention heads are focusing (the model's repr of "it" has some of "animal" and "tired")



- With all heads in the picture, things are harder to interpret

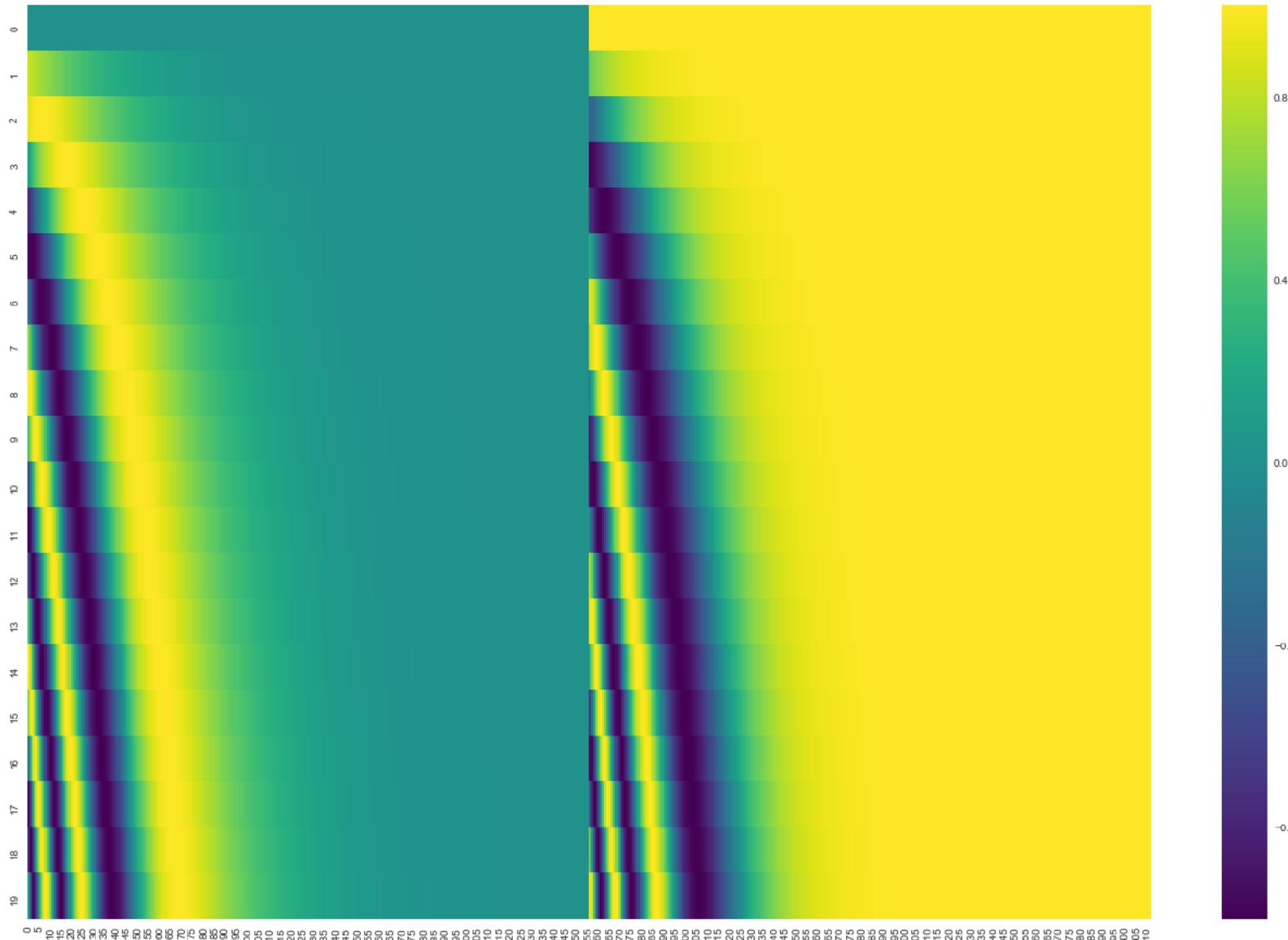
# Positional Embeddings

- To give the model a sense of order
- Learned or predefined



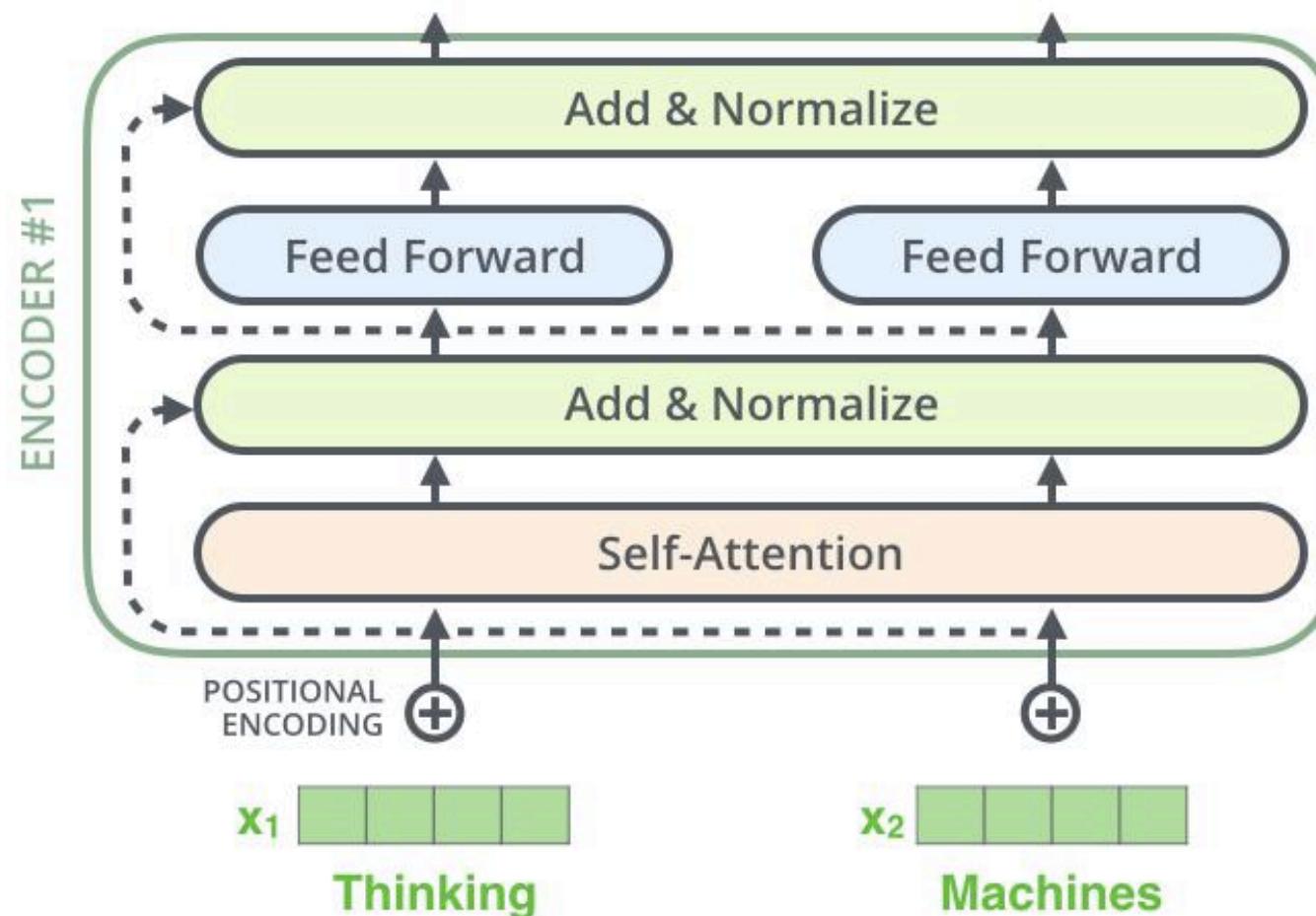
# Positional Embeddings

- What does it look like?



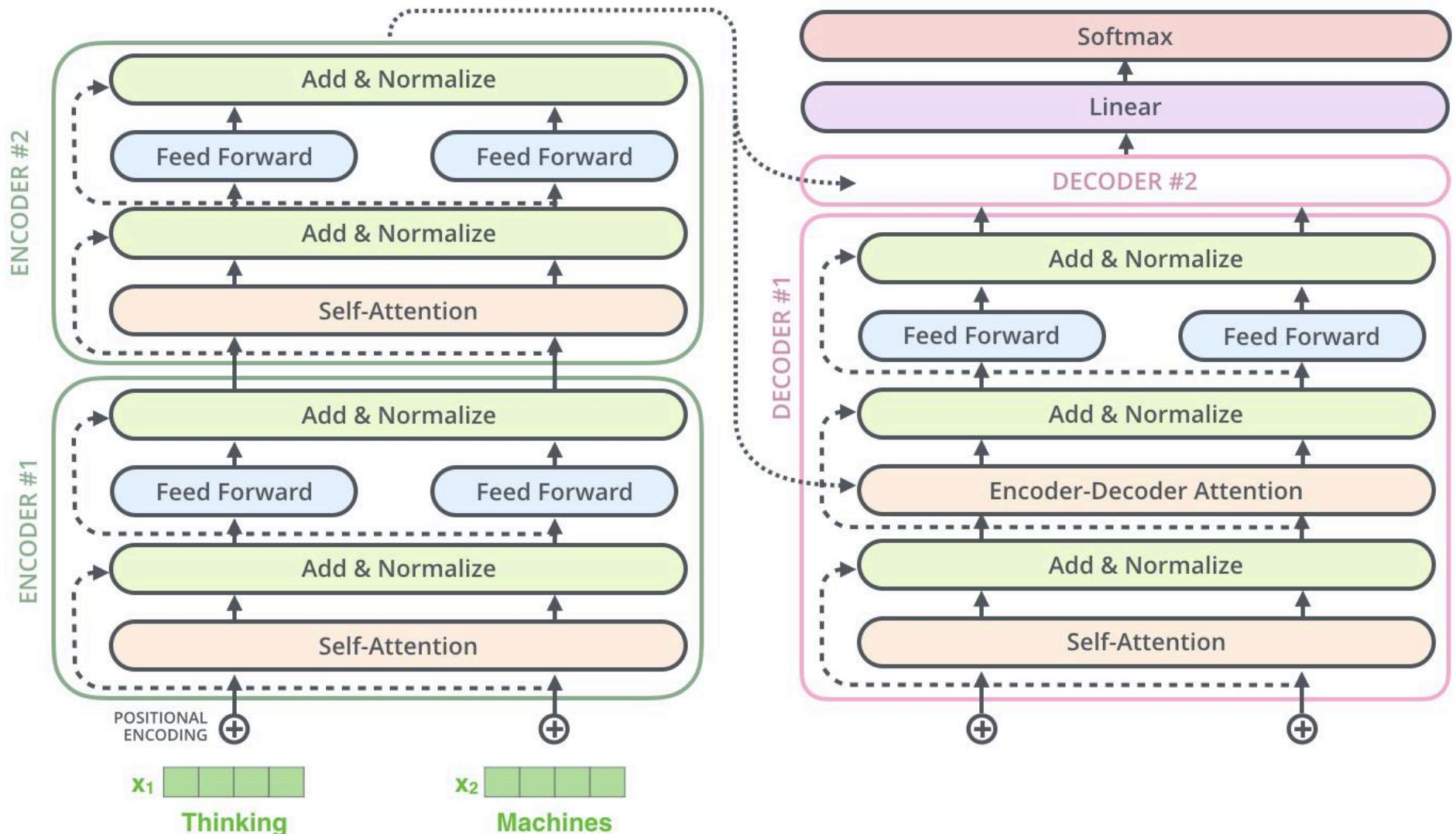
# The Residuals

- Each sub-layer in each encoder has a residual connection around it followed by a layer normalization



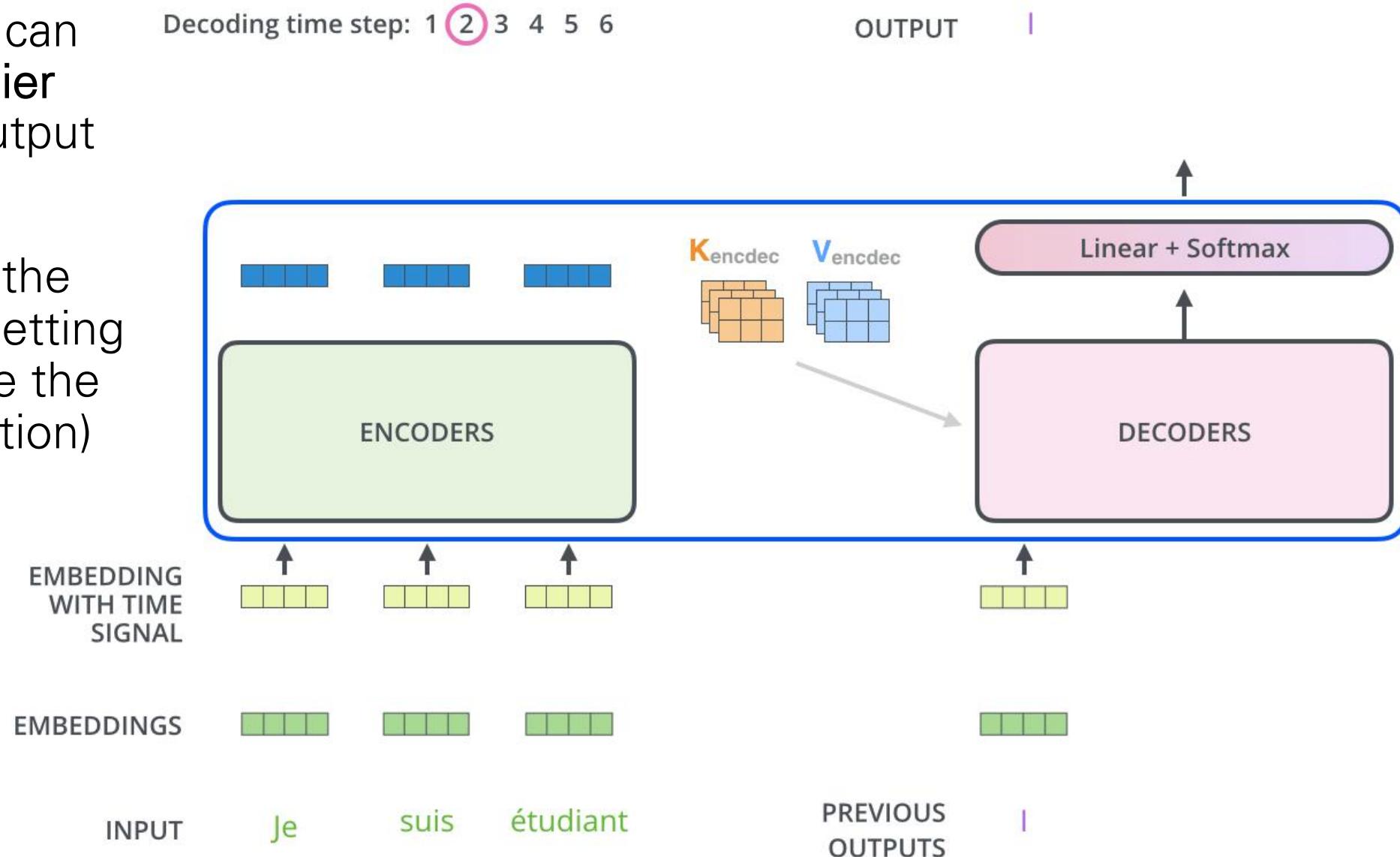
# The Residuals

- This goes for sub-layers in decoder as well



# The Decoder

- The self-attention can only attend to earlier positions in the output sequence.
- Done by masking the future positions (setting them to  $-\infty$  before the softmax in calculation)



# Final Layer

- The self-attention can only attend to earlier positions in the output sequence.
- Done by masking the future positions (setting them to `-inf` before the softmax in calculation)

Which word in our vocabulary is associated with this index?

Get the index of the cell with the highest value (`argmax`)

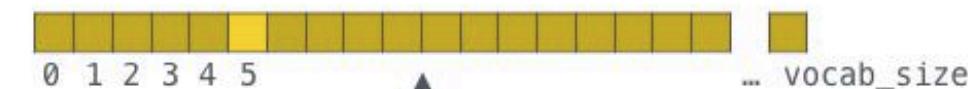
`log_probs`

`logits`

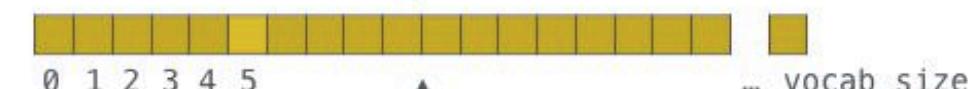
Decoder stack output

am

5



Softmax



Linear



# Results

- Machine Translation: WMT-2014 BLEU

	EN-DE	EN-FR
GNMT (orig)	24.6	39.9
ConvSeq2Seq	25.2	40.5
Transformer*	<b>28.4</b>	<b>41.8</b>

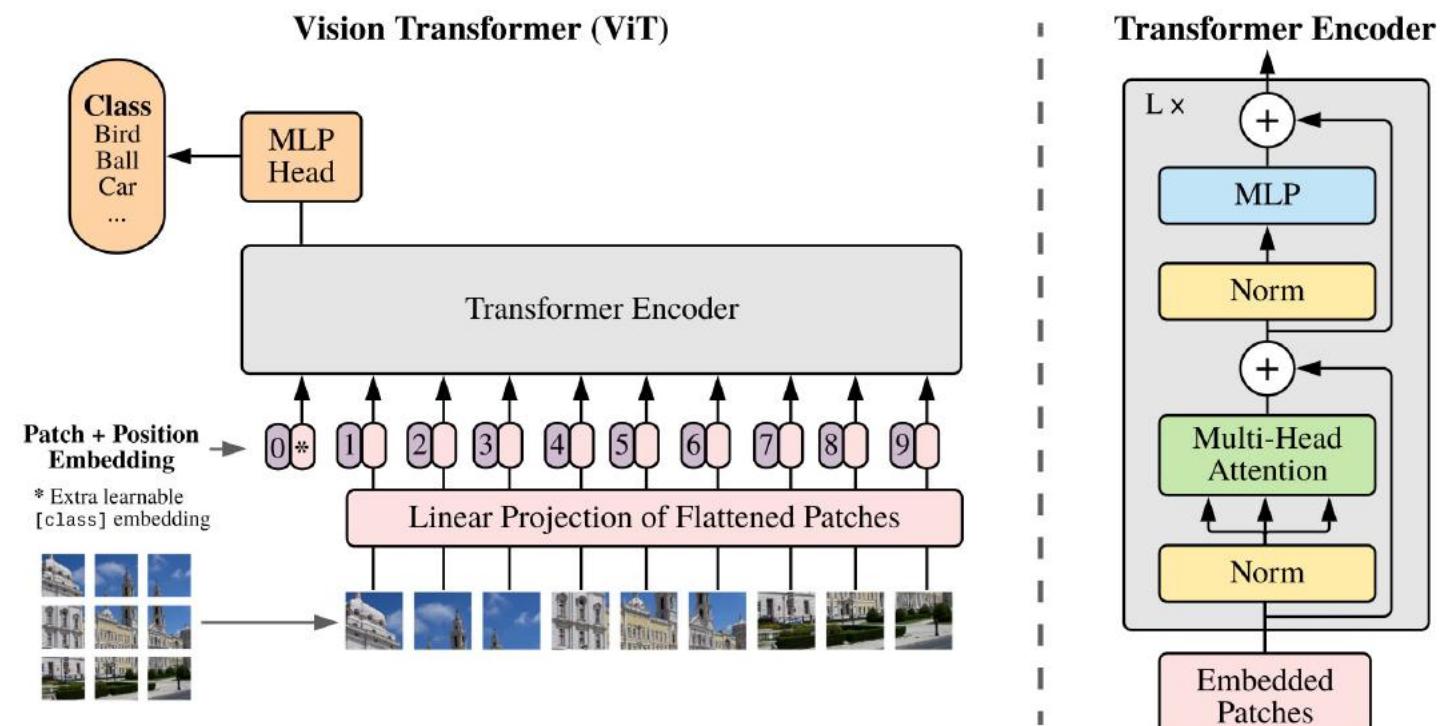
- Transformer models trained >3x faster than the others

# What Matters

- row B: reducing attention key size hurts the model
- row C: bigger model is better
- row D: dropout is helpful
- sinusoidal with learned positional emb have same results

	$N$	$d_{\text{model}}$	$d_{\text{ff}}$	$h$	$d_k$	$d_v$	$P_{\text{drop}}$	$\epsilon_{ls}$	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$		
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65		
(A)				1	512	512				5.29	24.9			
				4	128	128				5.00	25.5			
				16	32	32				4.91	25.8			
				32	16	16				5.01	25.4			
(B)						16				5.16	25.1	58		
						32				5.01	25.4	60		
(C)				2						6.11	23.7	36		
				4						5.19	25.3	50		
				8						4.88	25.5	80		
				256		32	32			5.75	24.5	28		
				1024		128	128			4.66	26.0	168		
				1024		32	32			5.12	25.4	53		
				4096						4.75	26.2	90		
(D)							0.0			5.77	24.6			
							0.2			4.95	25.5			
							0.0			4.67	25.3			
										5.47	25.7			
(E)	positional embedding instead of sinusoids									4.92	25.7			
big	6	1024	4096	16				0.3	300K	<b>4.33</b>	<b>26.4</b>	213		

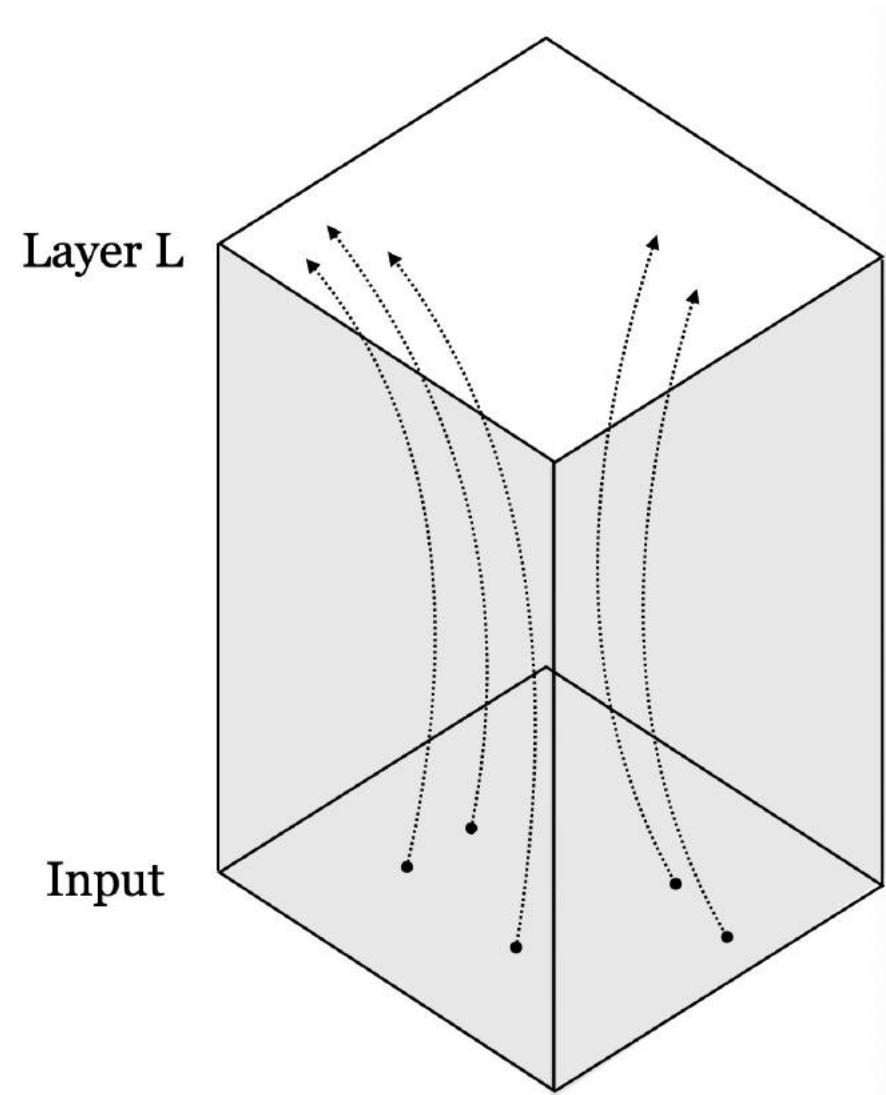
# Transformers for Vision



An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, Dosovitskiy et al, ICLR 2021

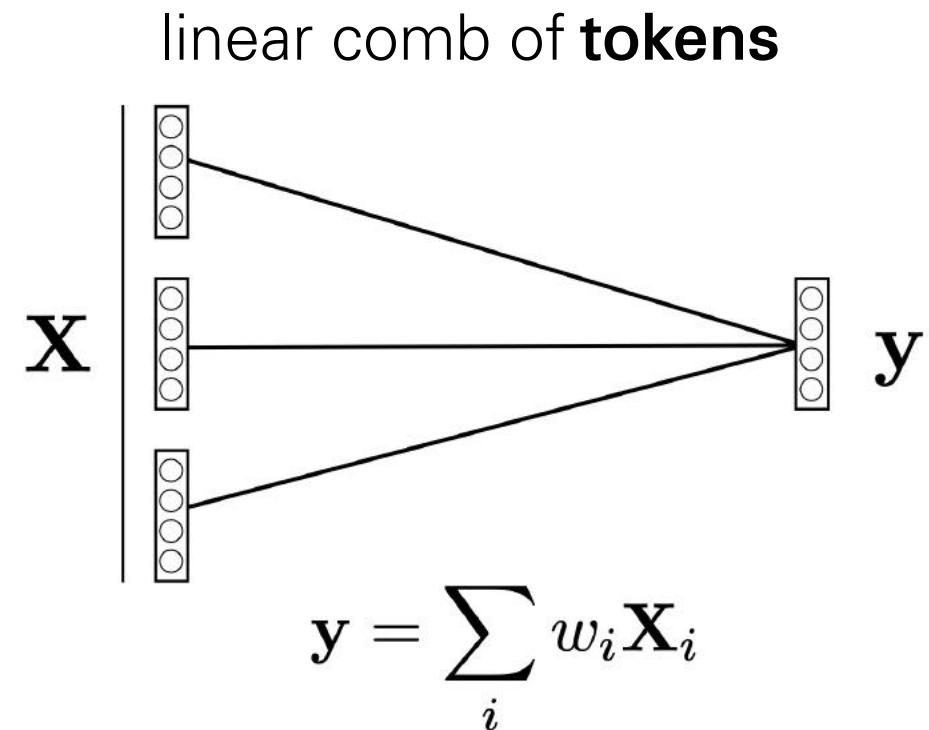
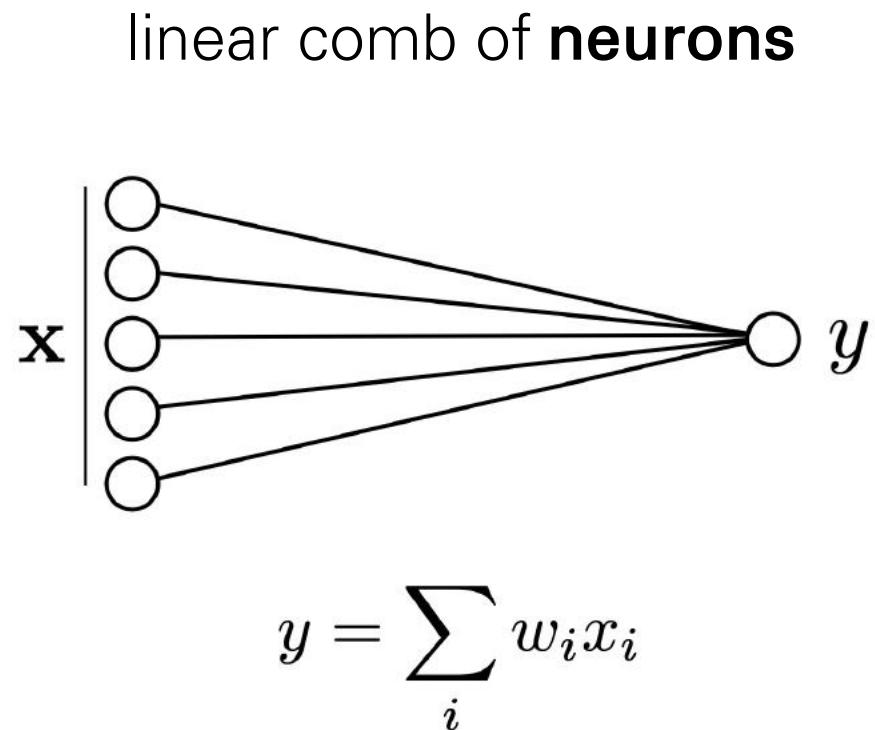
# Deep Nets are Data Transformers

- Deep nets transform datapoints, layer by layer
- Each layer is a different representation of the data
- We call these representations embeddings



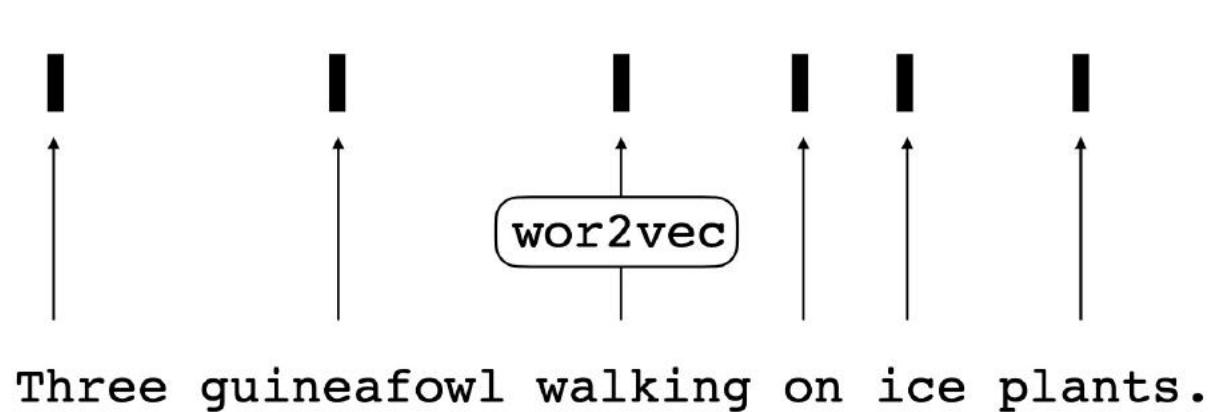
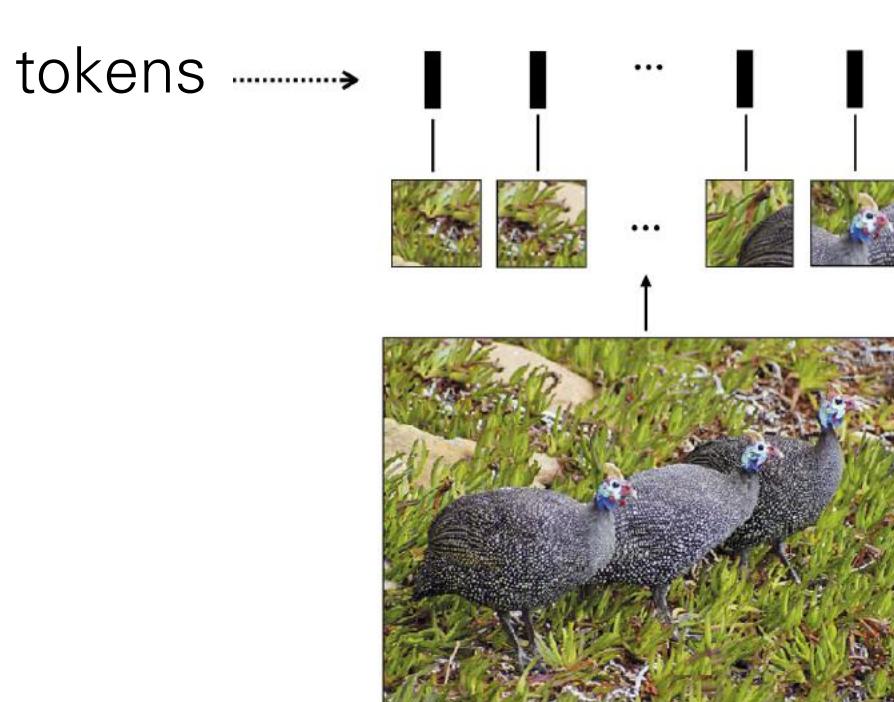
# A New Data Structure: Tokens

- A **token** is just transformer lingo for a vector of neurons (a.k.a. an embedding)
- But the connotation is that a token is an encapsulated bundle of information; with transformers we will operate over tokens rather than over neurons



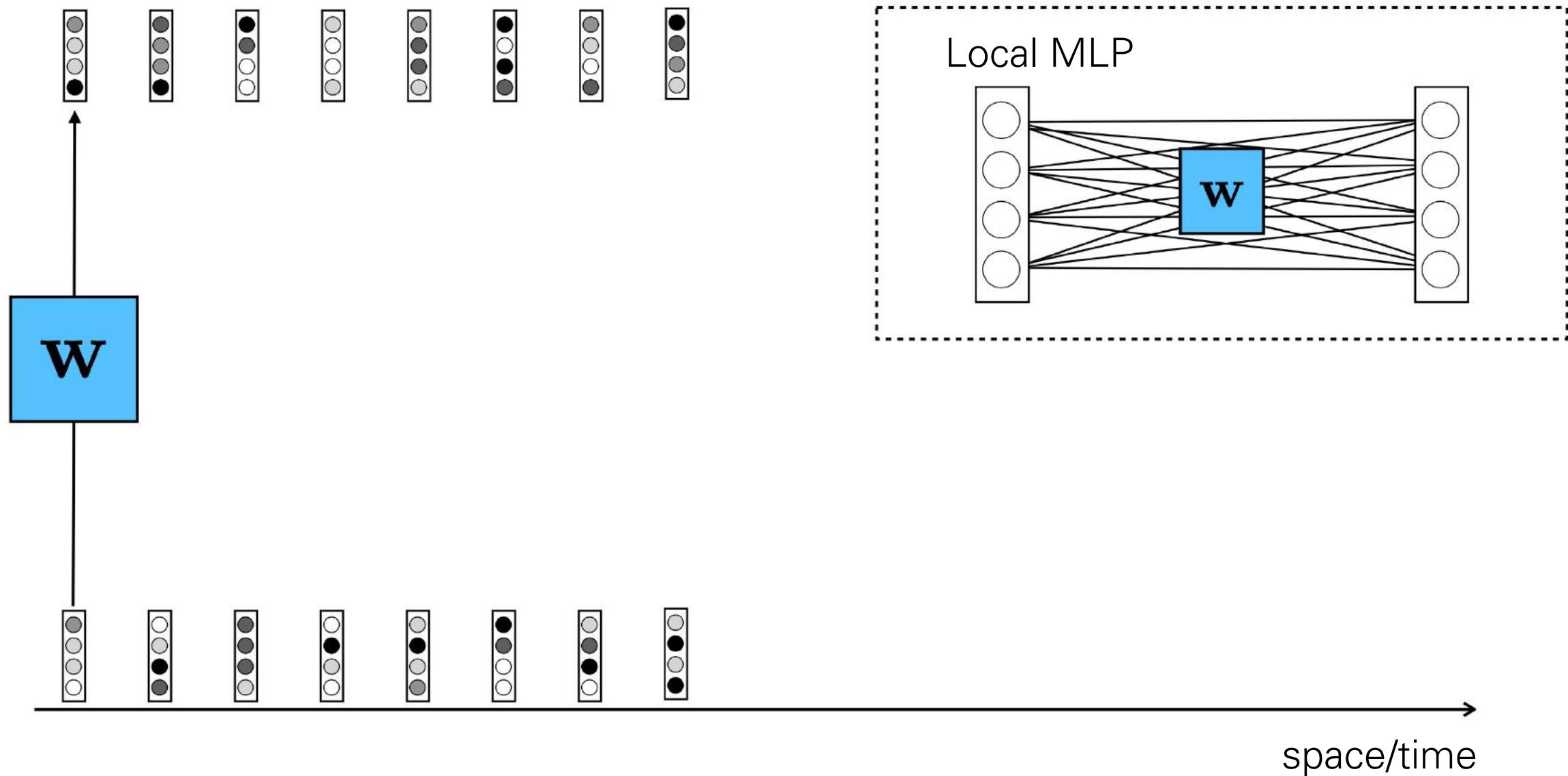
# Tokenizing The Input Data

- When operating over neurons, we represent the input as an array of scalar-valued measurements (e.g., pixels)
- When operating over tokens, we represent the input as an array of vector-valued measurements

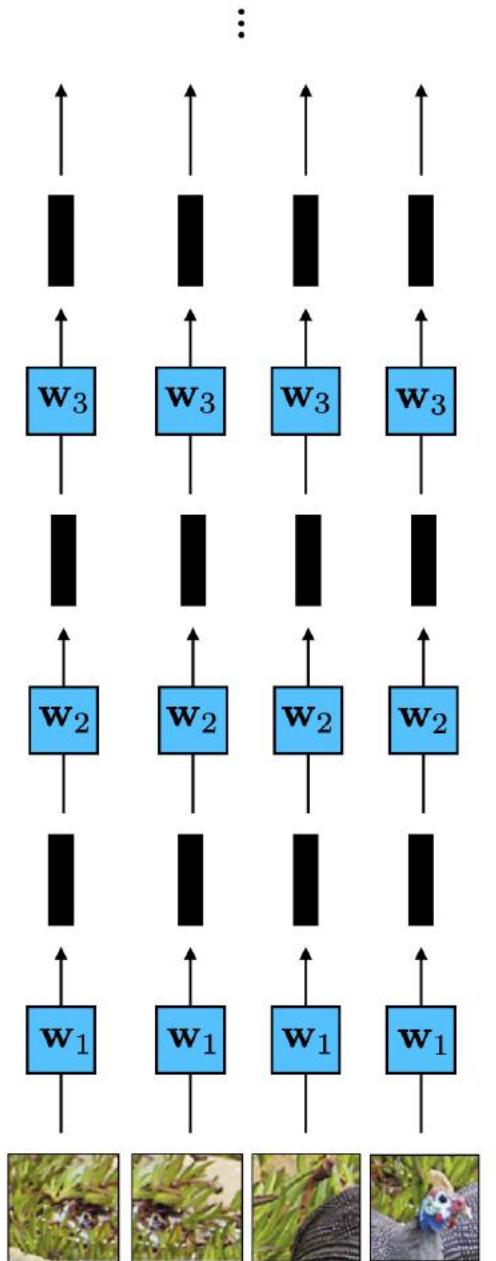


Three guineafowl walking on ice plants.

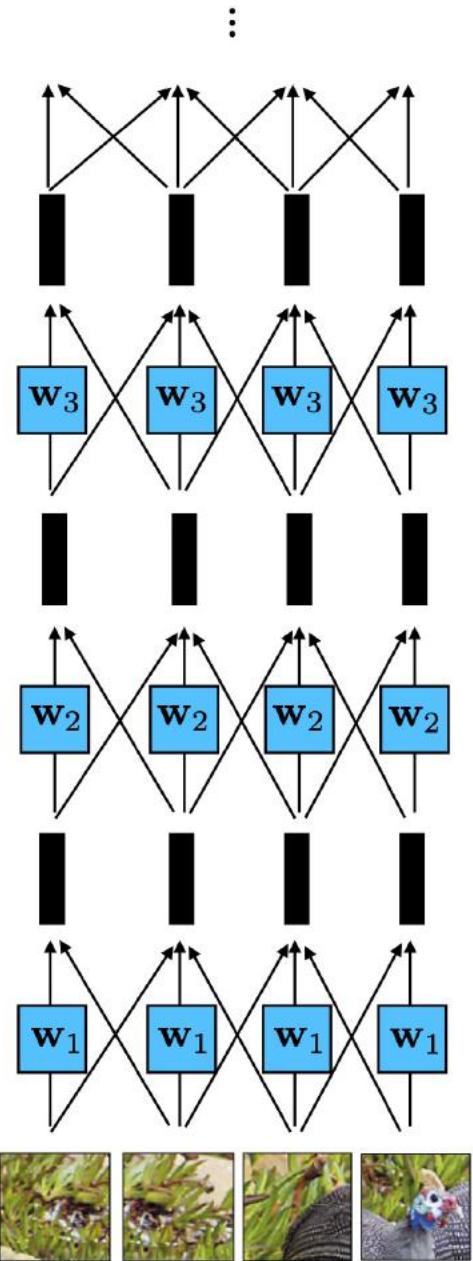
# Convolution over Tokens



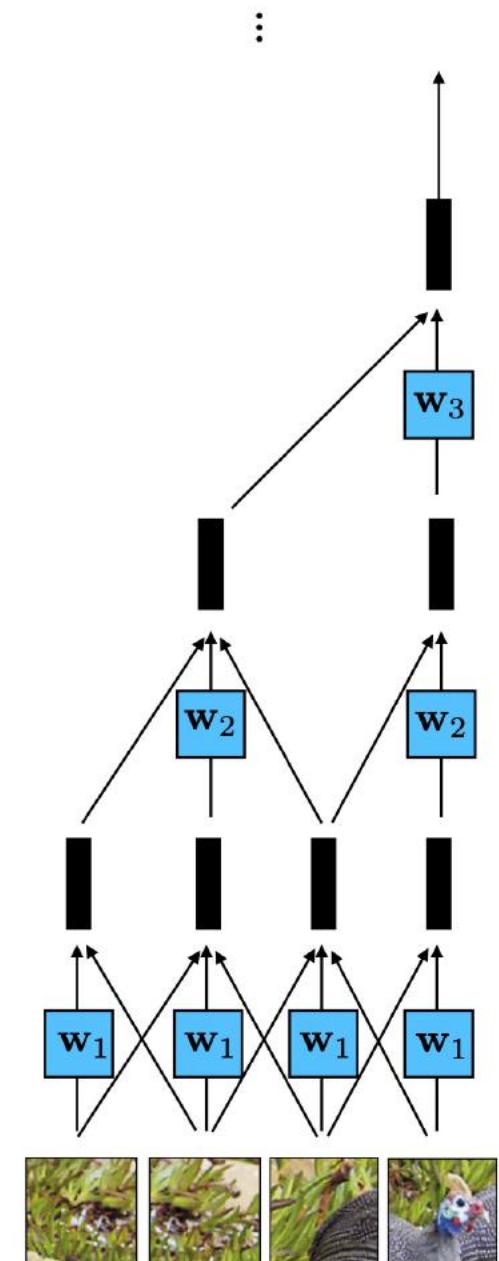
conv w/o overlap



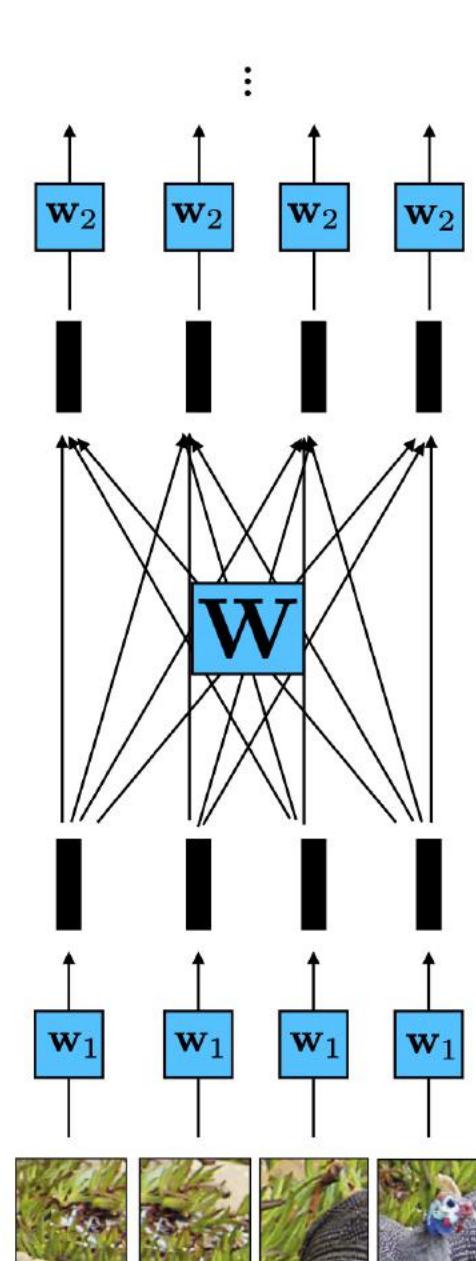
conv w overlap



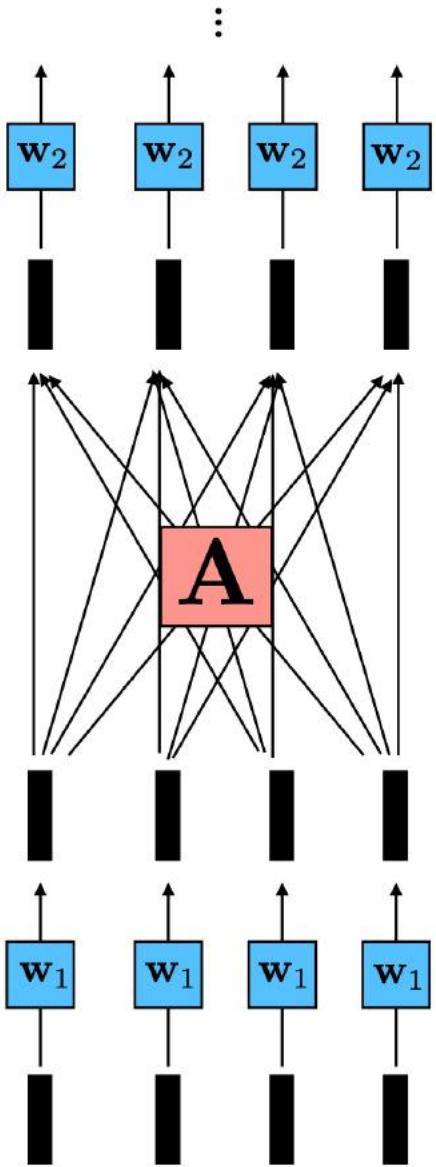
conv pyramid



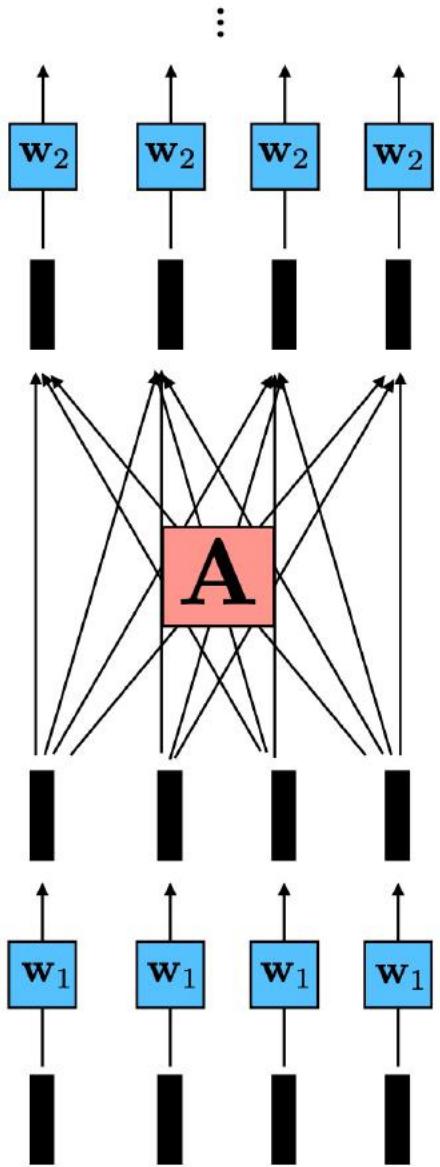
fc layer



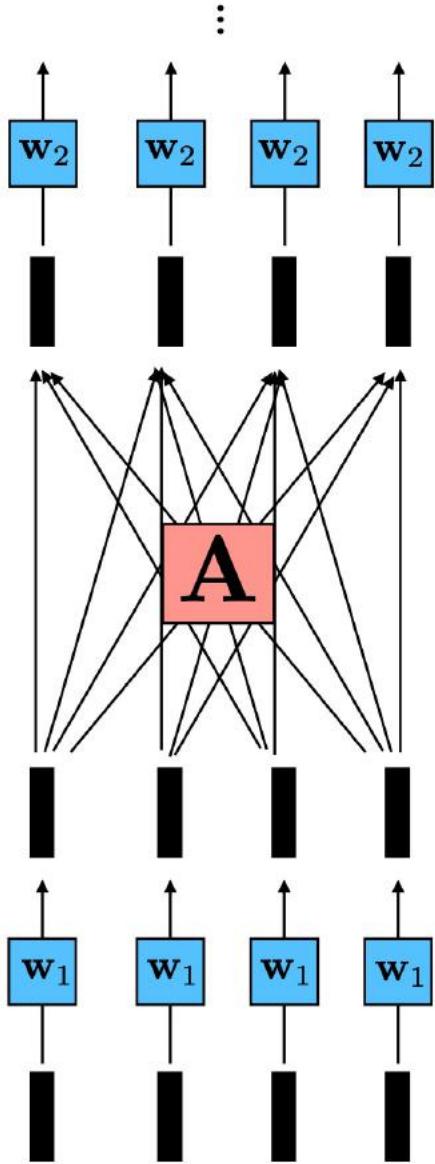
# Attention Layer



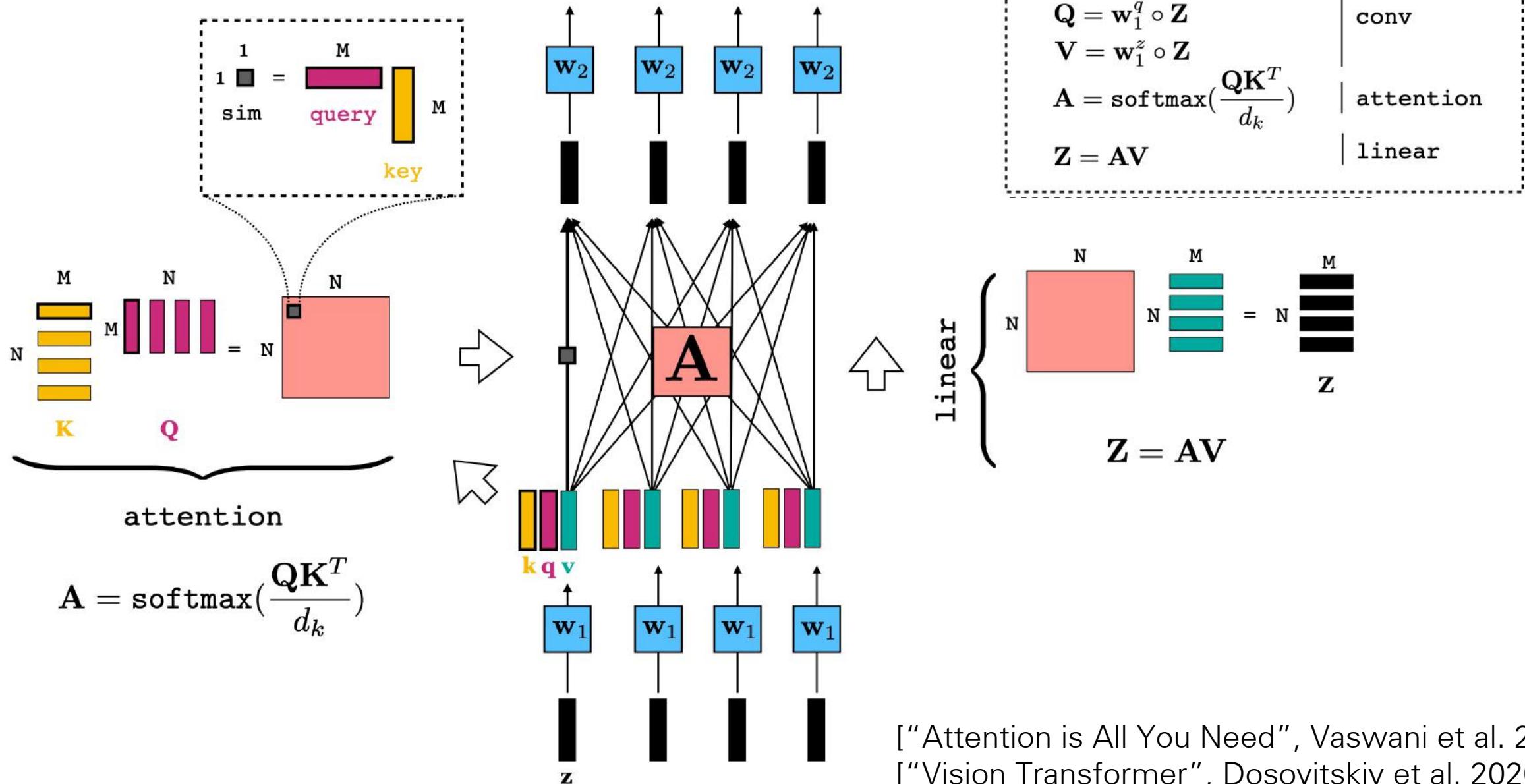
# Attention Layer



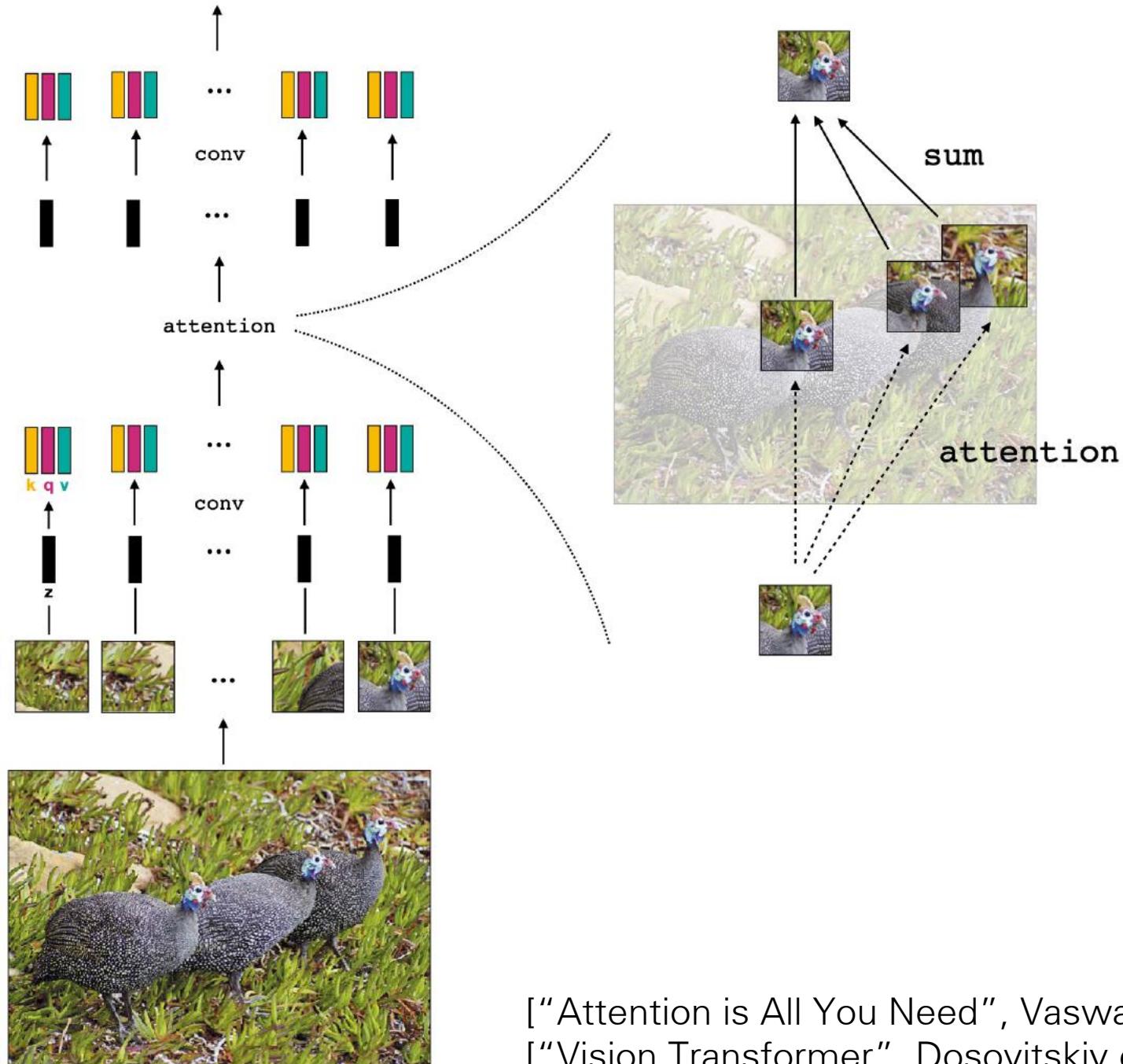
# Attention Layer



# Attention Layer



# Transformer (simplified)



[“Attention is All You Need”, Vaswani et al. 2017]  
[“Vision Transformer”, Dosovitskiy et al. 2020]

# Attention Maps In A Trained Transformer



["DINO", Caron et all. 2021]

# Summary

- attention is used to focus on parts of inputs/outputs
- it can be content/location based and hard/soft
- its 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]

**Next lecture:**  
**Autoregressive Models**