



STAT 453: Introduction to Deep Learning and Generative Models

Ben Lengerich

Lecture 20: Attention & Transformers

November 12, 2025

Reading: See course homepage

Last time: Recurrent Neural Networks (RNNs)

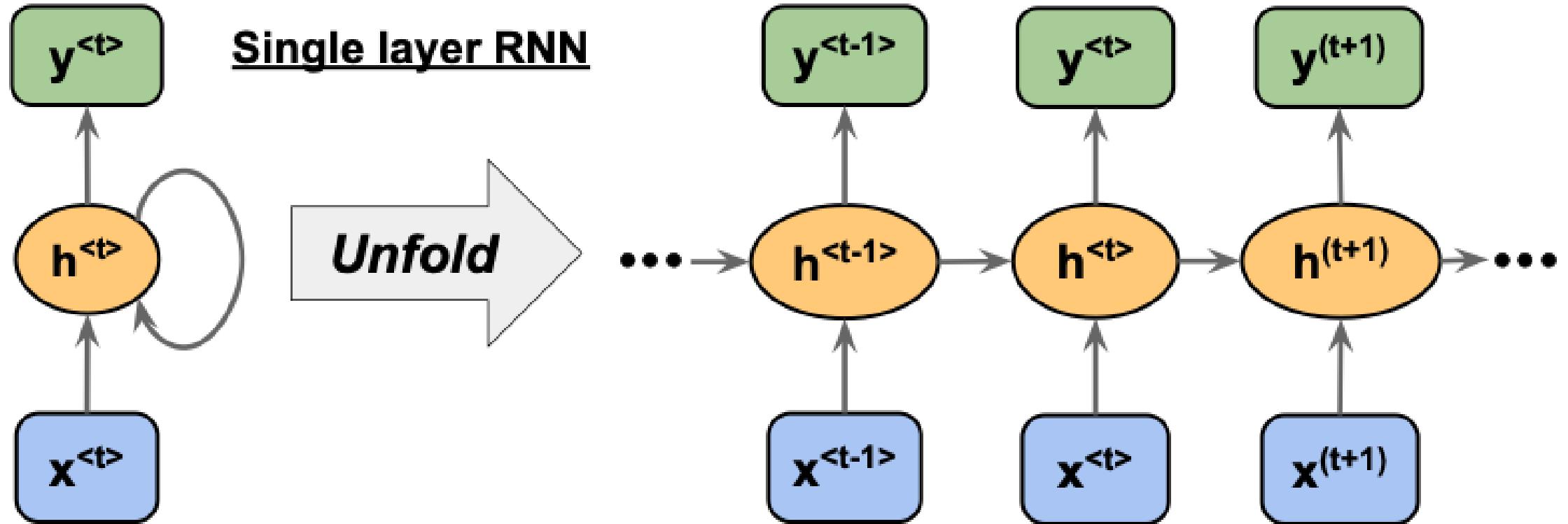


Image source: Sebastian Raschka, Vahid Mirjalili. Python Machine Learning. 3rd Edition. Packt, 2019

From RNN...

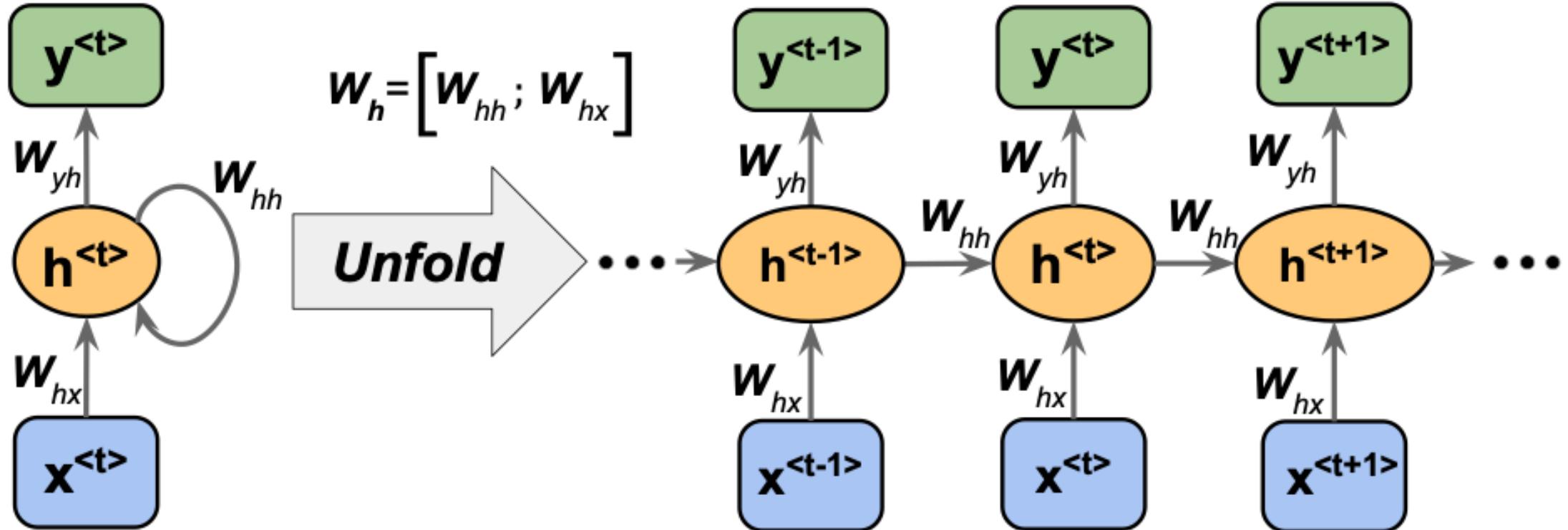


Image source: Sebastian Raschka, Vahid Mirjalili. Python Machine Learning. 3rd Edition. Packt, 2019

From RNN...to GPT

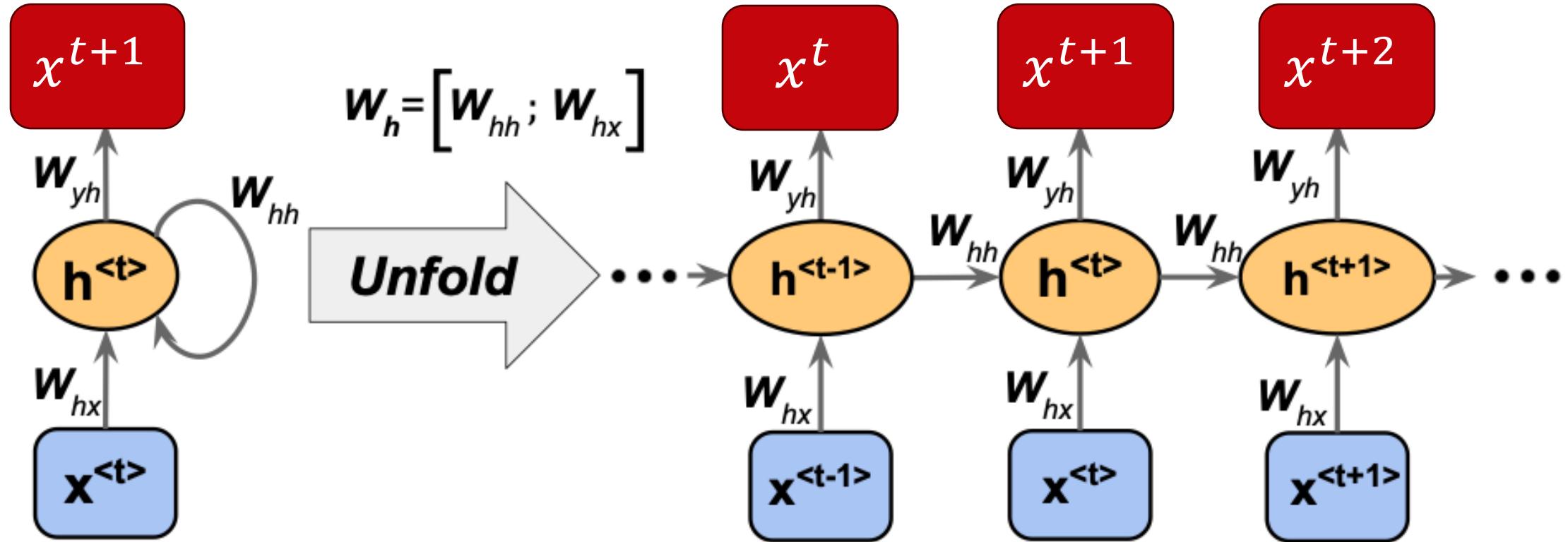
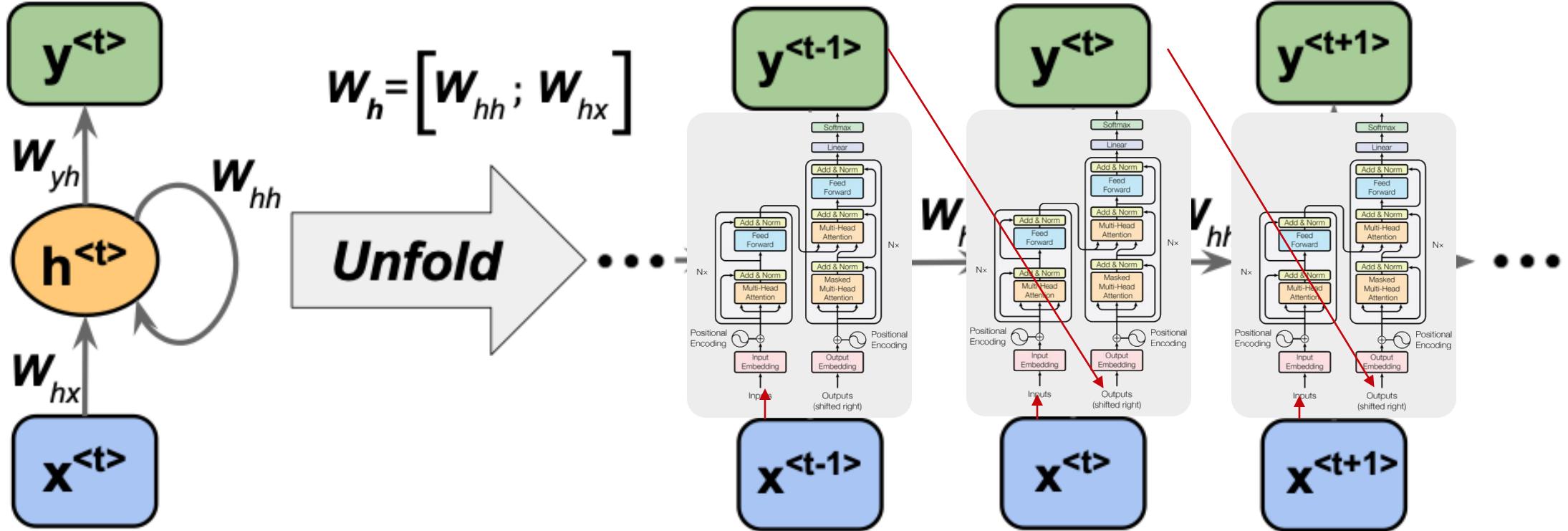


Image source: Sebastian Raschka, Vahid Mirjalili. Python Machine Learning. 3rd Edition. Packt, 2019



From RNN...to GPT...by Transformers

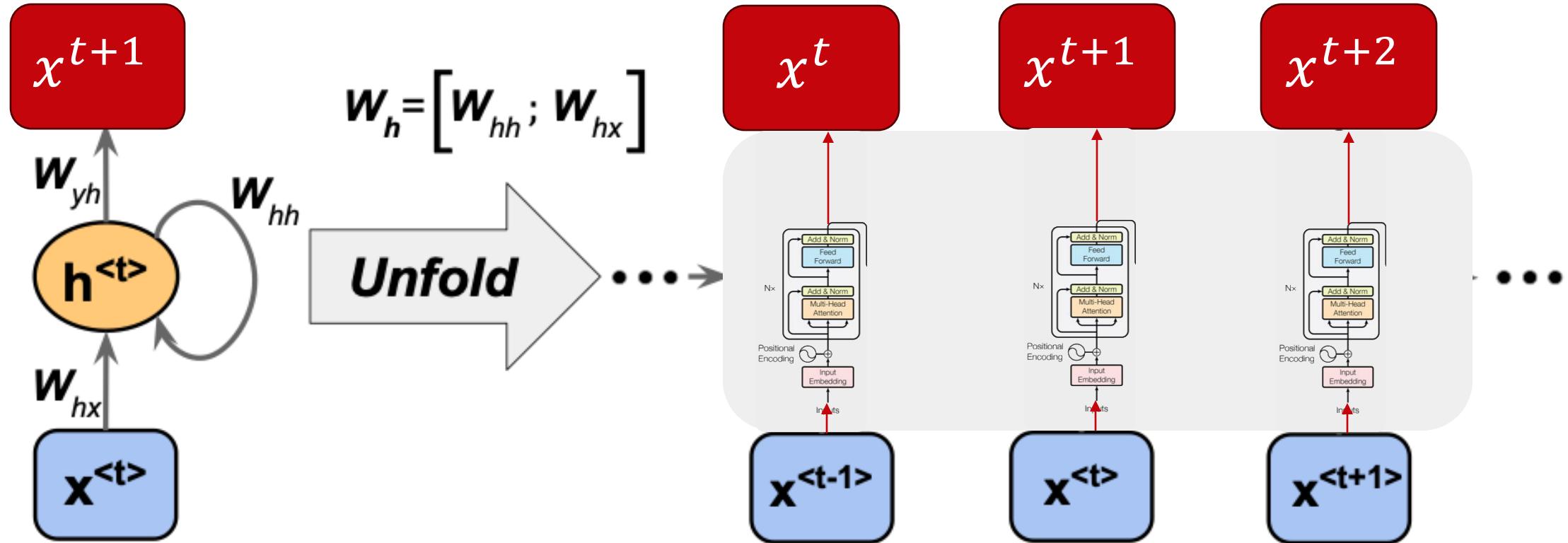


Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I., 2017. Attention Is All You Need.

Image source: Sebastian Raschka, Vahid Mirjalili. Python Machine Learning. 3rd Edition. Packt, 2019



From RNN...to GPT...by Transformers



Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I., 2017. Attention Is All You Need.

Image source: Sebastian Raschka, Vahid Mirjalili. Python Machine Learning. 3rd Edition. Packt, 2019



Today

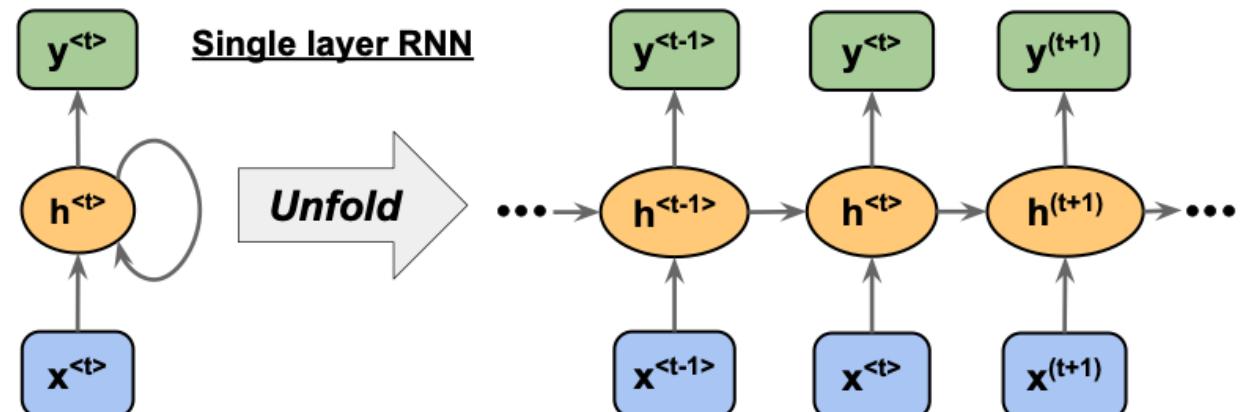
- **Different Ways to Model Text**
- Sequence Modeling with RNNs
- Different Types of Sequence Modeling Tasks
- Backpropagation Through Time
- Long-Short Term Memory (LSTM)
- Many-to-one Word RNNs



The Attention Mechanism

Why Attention?

- Consider machine translation:
 - Do we really need the whole sequence to translate each word?
 - Where is **the library**? →
 - Donde esta **la** biblioteca?
 - Where is **the huge public library**? →
 - Donde esta **la** enorme biblioteca publica?
- Problem: RNNs compress all information into a fixed-length vector.
Long-range dependencies are tricky.



“Attention”

- Originally developed for language translation:
Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. <https://arxiv.org/abs/1409.0473>
- *“... allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word ...”*

“traditional”
encoder+decoder
RNN

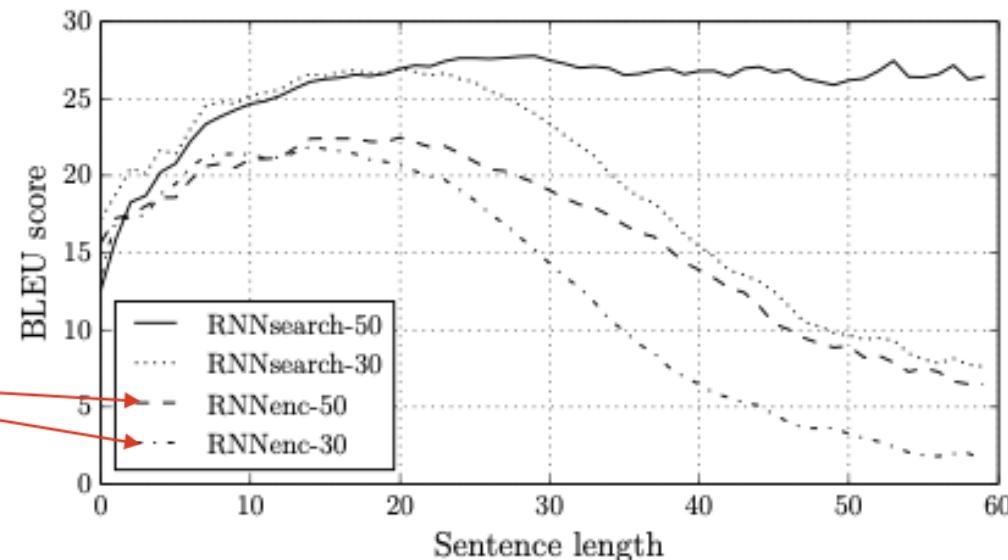


Figure 2: The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models.



“Attention”

Main idea:

Assign attention weight to each word, to know how much "attention" the model should pay to each word

“Attention”

Main idea:

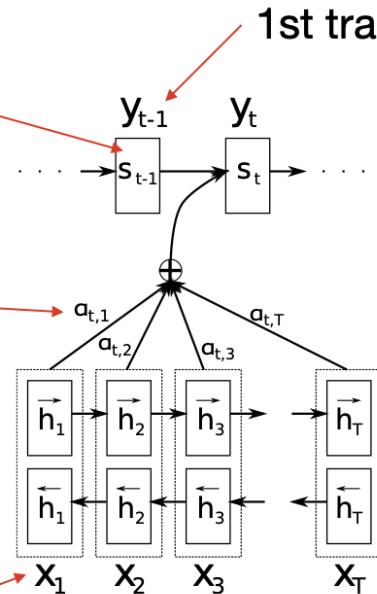
Assign attention weight to each word, to know how much "attention" the model should pay to each word

**Hidden state in a regular RNN
(RNN #2)**

Attention weight

1st input word

1st translated word



**Bidirectional RNN
(RNN #1)**

Figure 1: The graphical illustration of the proposed model trying to generate the t -th target word y_t given a source sentence (x_1, x_2, \dots, x_T) .

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. <https://arxiv.org/abs/1409.0473>



Hard attention?

- Make a zero-one decision about where to attend.
- Problem: Hard to train. Requires methods such as reinforcement learning

Benefit: Interpretable?

Review

the beer was n't what i expected, and i'm not sure it's "true to style", but i thought it was delicious. a **very pleasant ruby red-amber color** with a relatively brilliant finish, but a limited amount of carbonation, from the look of it. aroma is what i think an amber ale should be - a nice blend of caramel and happiness bound together.

Ratings

Look: 5 stars

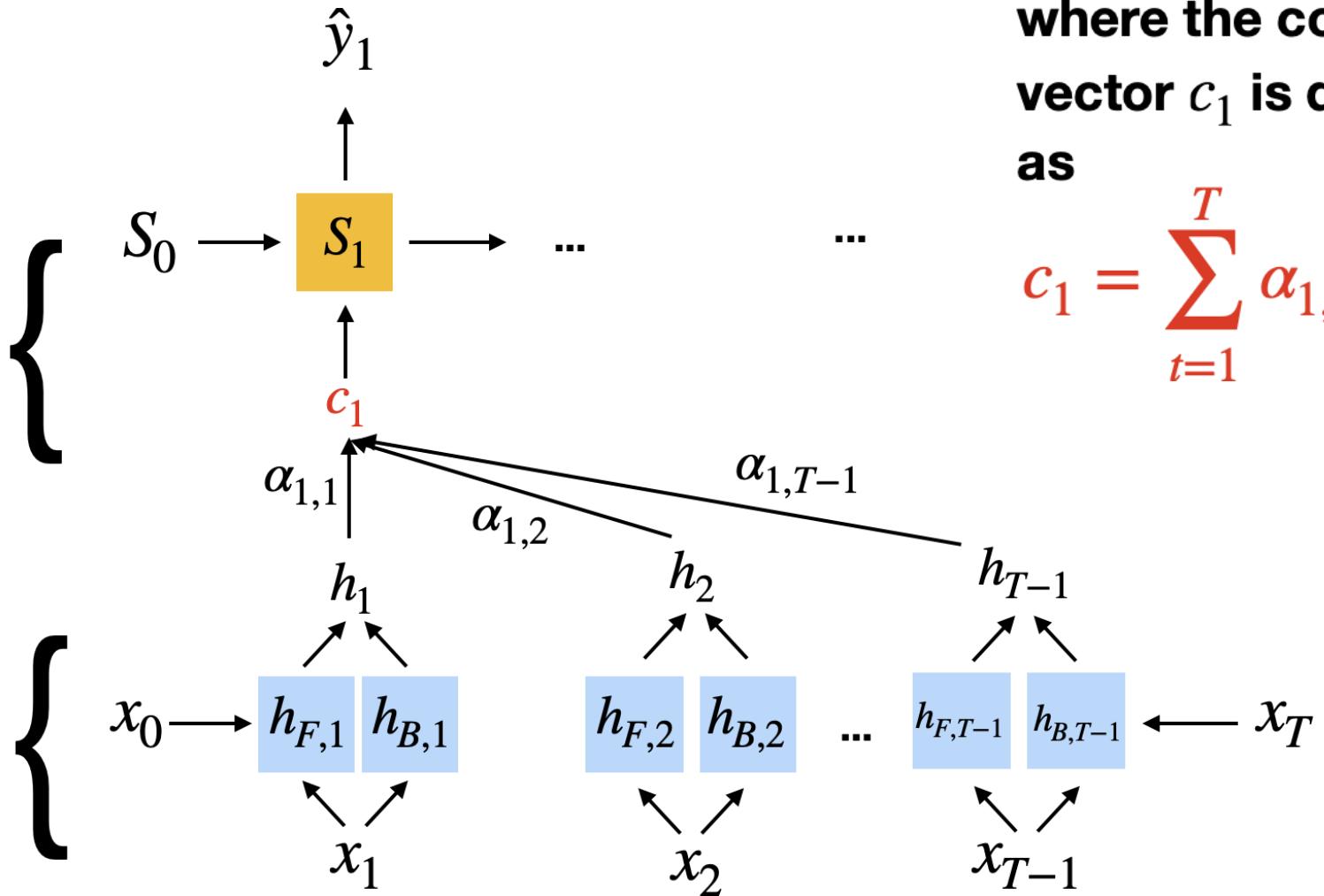
Smell: 4 stars

Lei et al 2016

Soft attention

Added attention
 (looks like a standard RNN but with context vectors as in-/output)

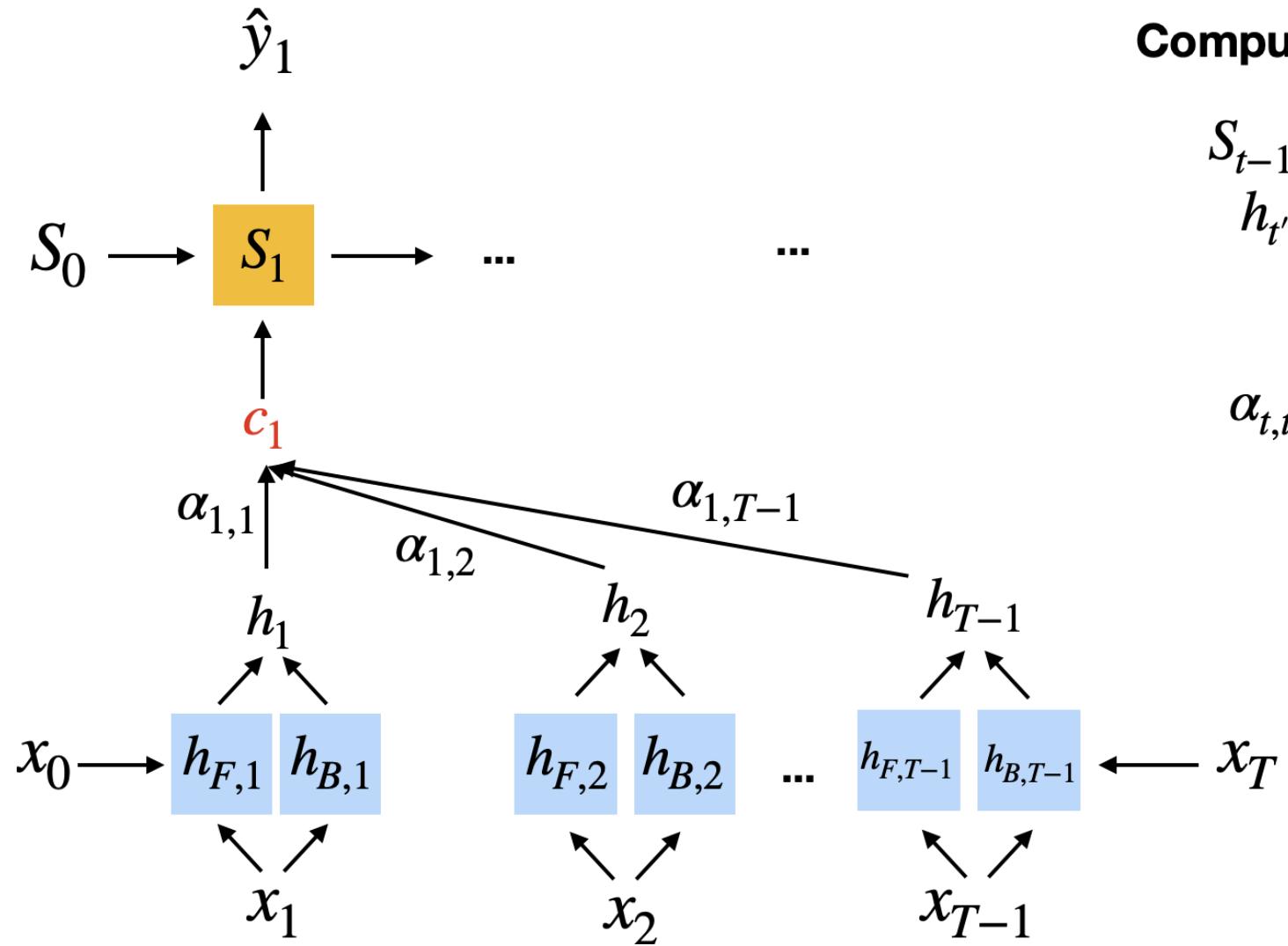
Bidirectional RNN



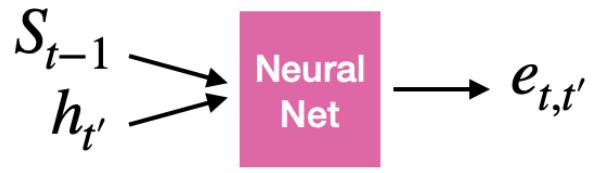
where the context vector c_1 is defined as

$$c_1 = \sum_{t=1}^T \alpha_{1,t} h_t$$

Soft attention

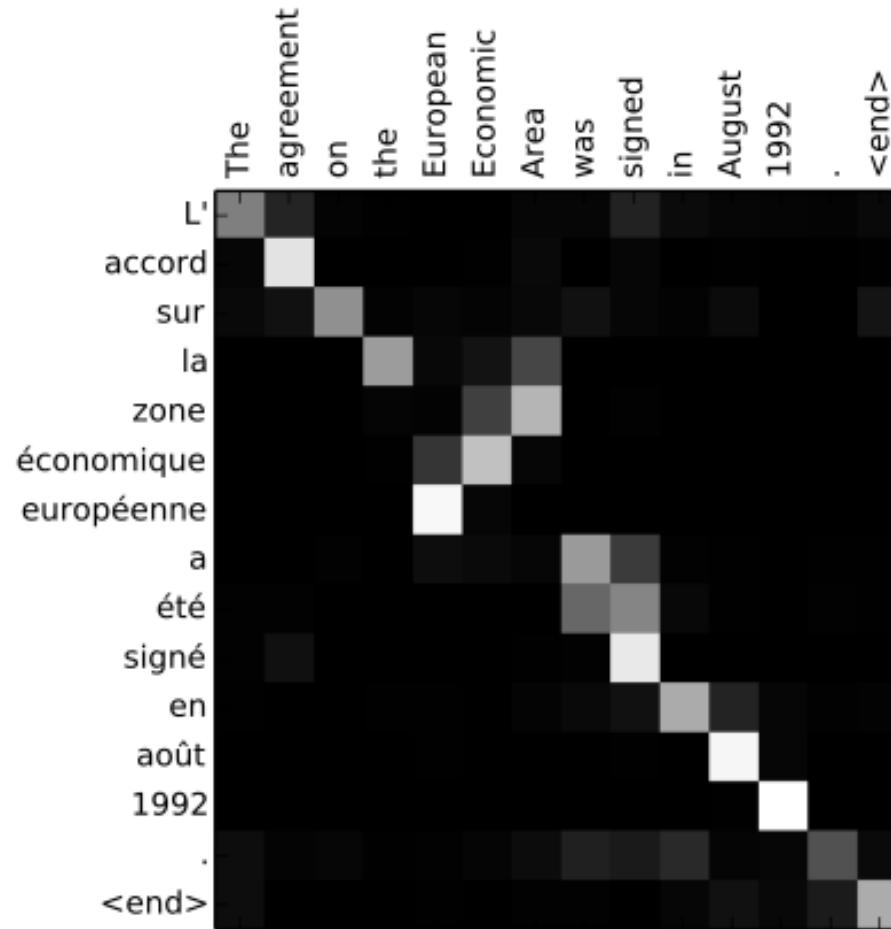


Computing attention weights

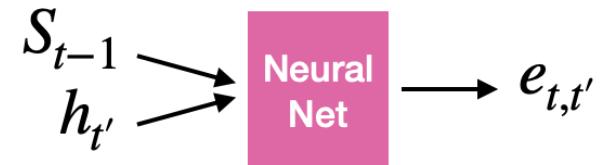


$$\alpha_{t,t'} = \frac{\exp(e_{t,t'})}{\sum_{t'=1}^T \exp(e_{t,t'})}$$

Soft attention



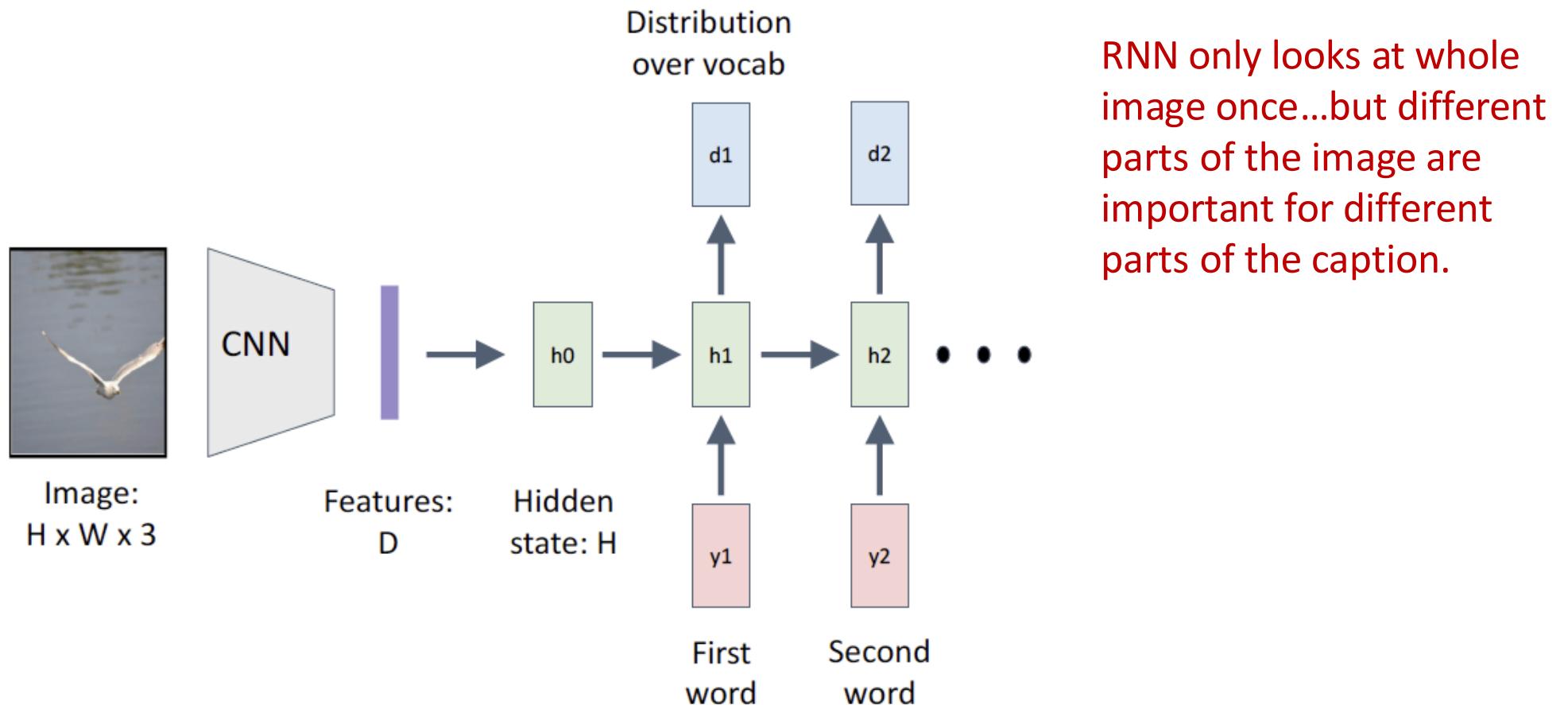
Computing attention weights



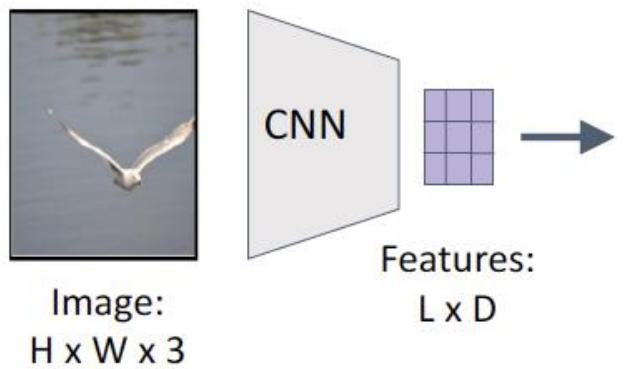
$$\alpha_{t,t'} = \frac{\exp(e_{t,t'})}{\sum_{t'=1}^T \exp(e_{t,t'})}$$

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. <https://arxiv.org/abs/1409.0473>

Example: Soft Attention for Image Captioning

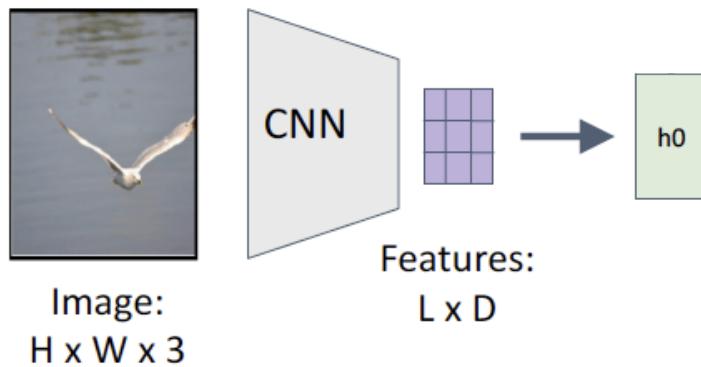


Example: Soft Attention for Image Captioning



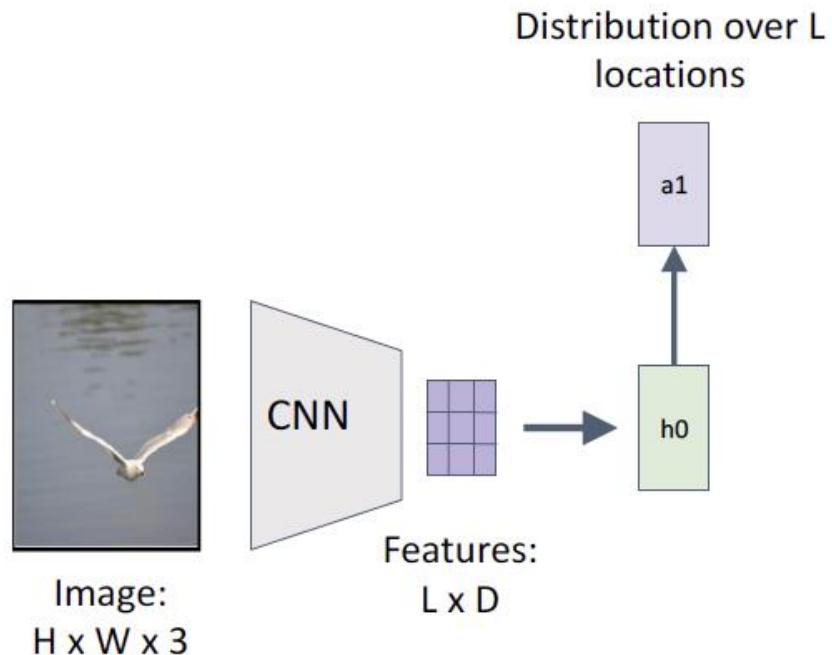
Xu et al, "Show, Attend and Tell:
Neural Image Caption Generation
with Visual Attention", ICML 2015

Example: Soft Attention for Image Captioning



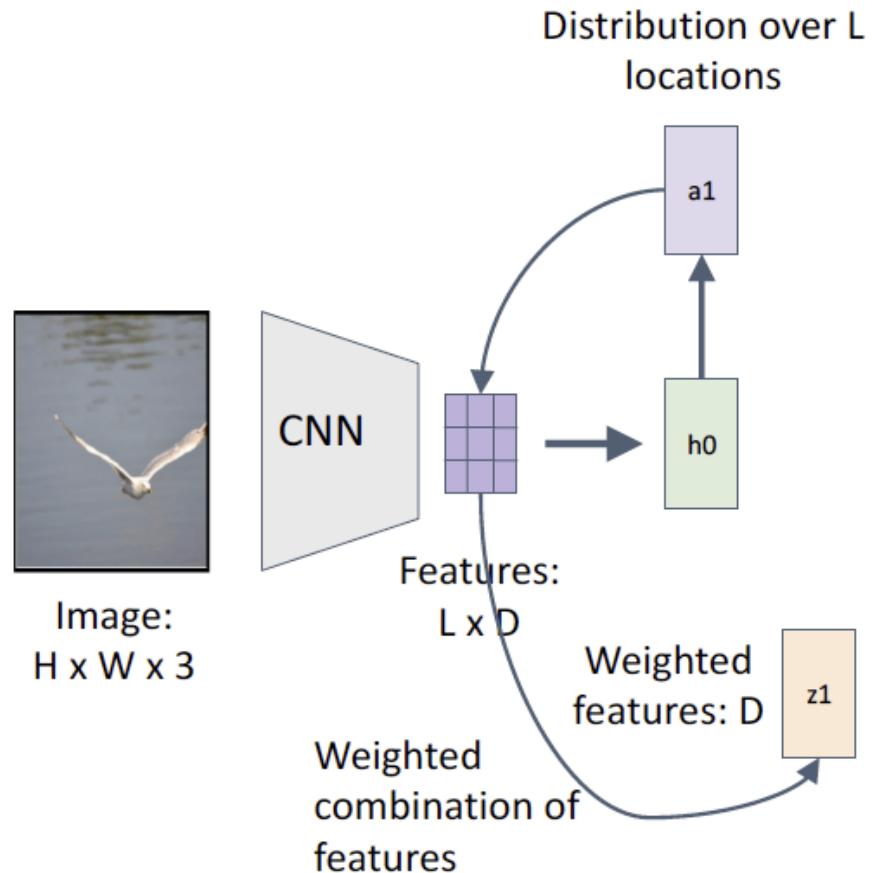
Xu et al, "Show, Attend and Tell:
Neural Image Caption Generation
with Visual Attention", ICML 2015

Example: Soft Attention for Image Captioning

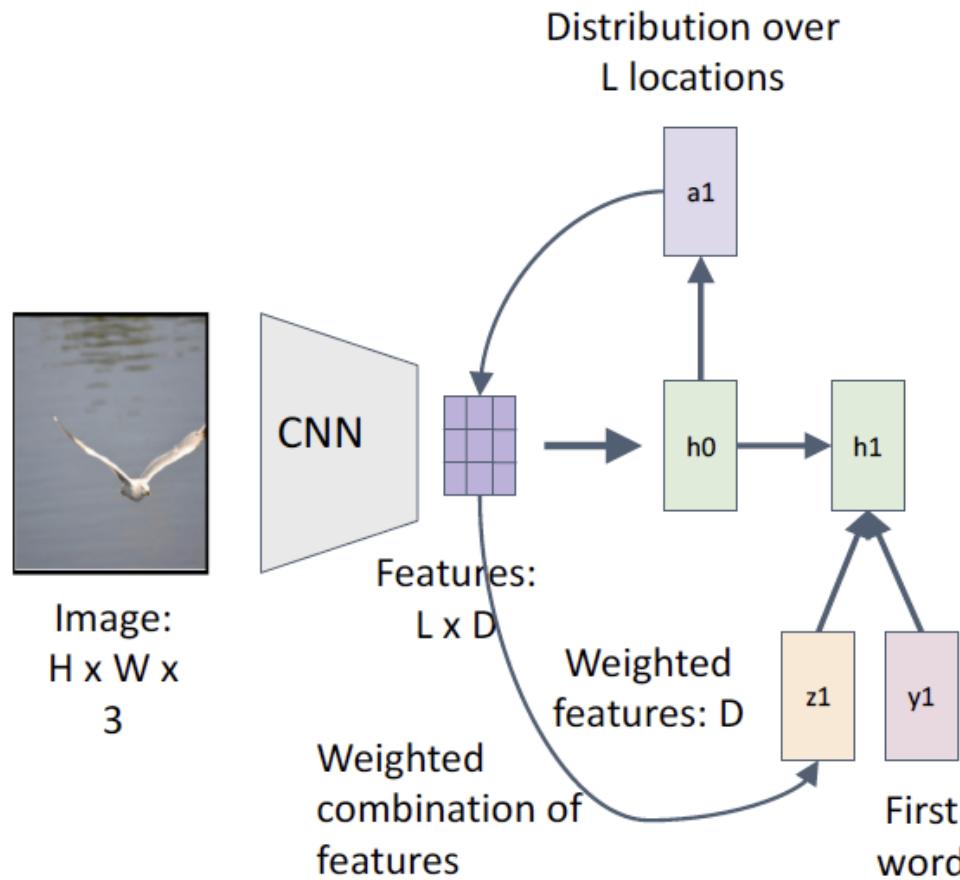


Xu et al, "Show, Attend and Tell:
Neural Image Caption Generation
with Visual Attention", ICML 2015

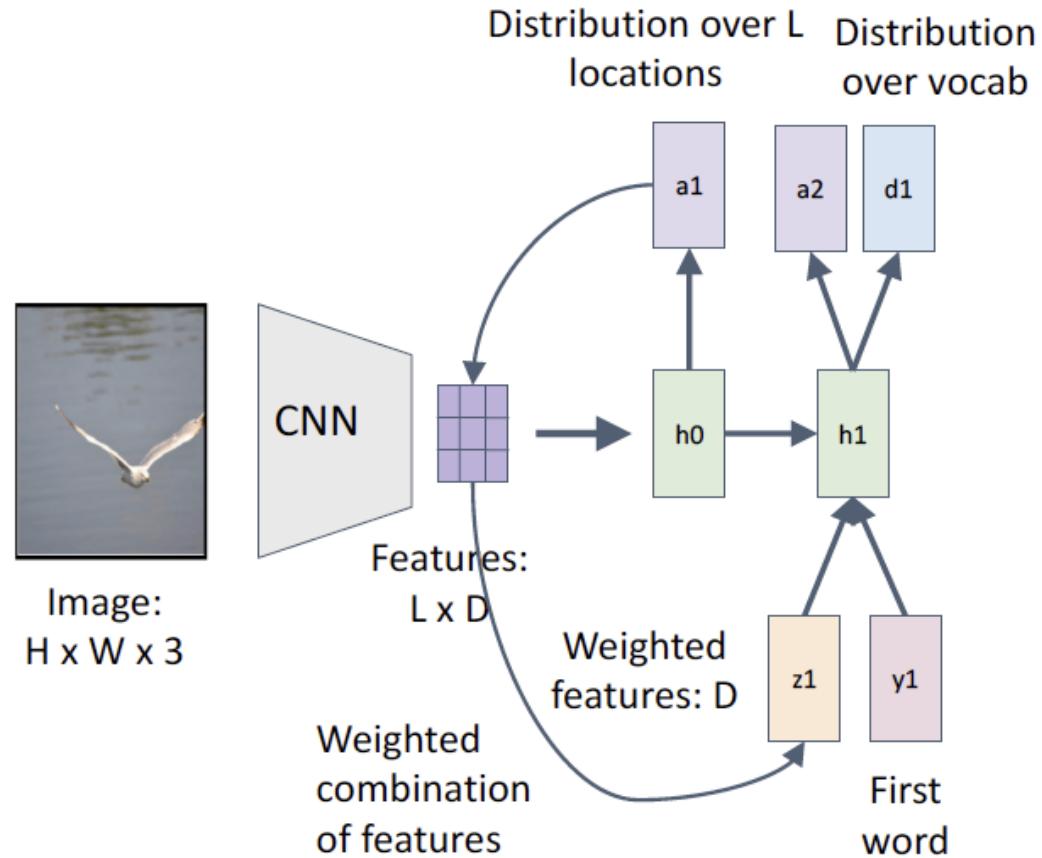
Example: Soft Attention for Image Captioning



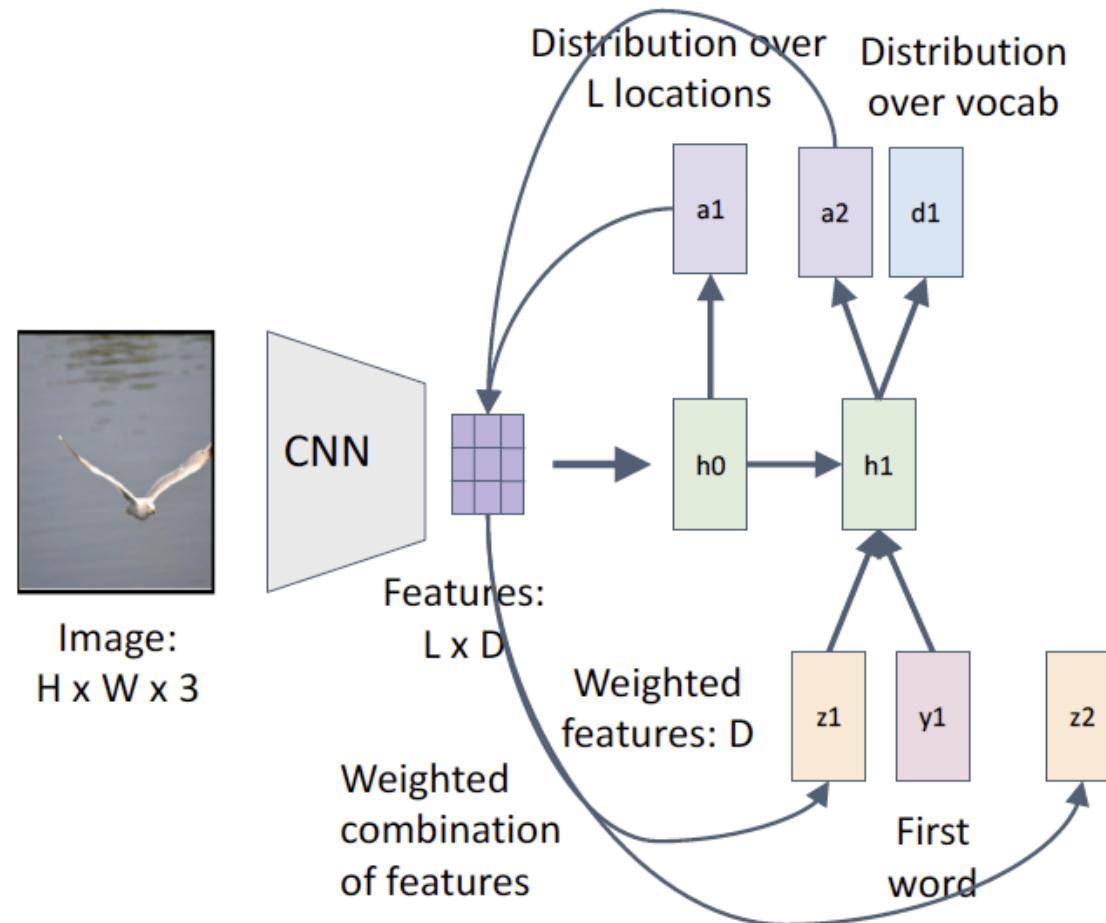
Example: Soft Attention for Image Captioning



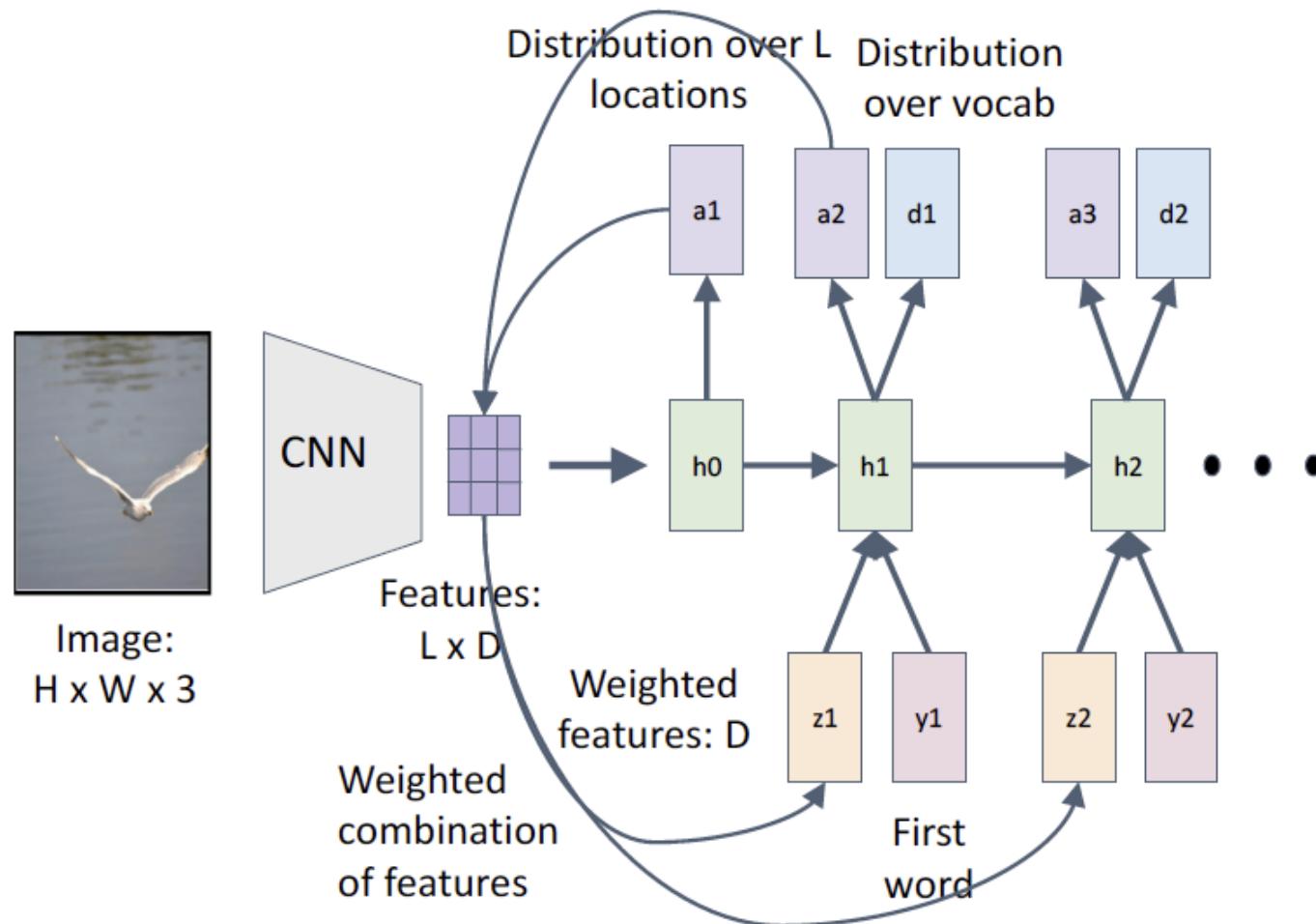
Example: Soft Attention for Image Captioning



Example: Soft Attention for Image Captioning

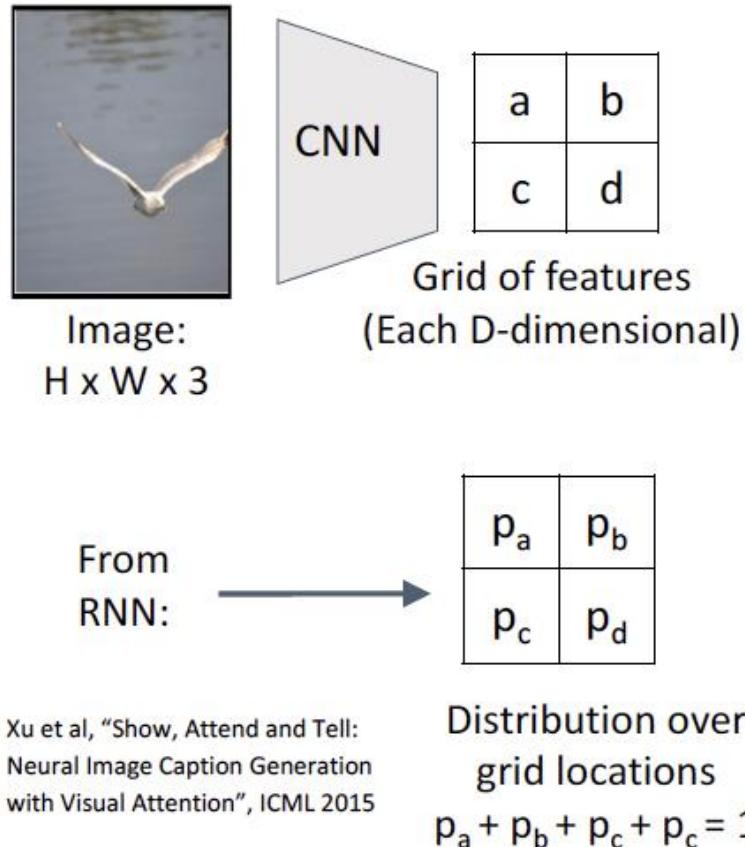


Example: Soft Attention for Image Captioning

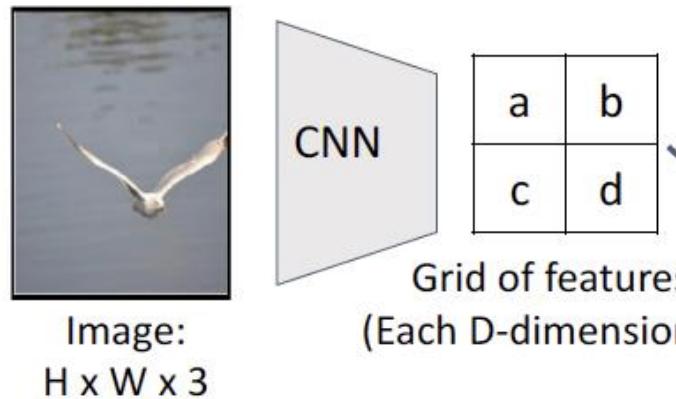




Aside: CNNs were an example of hard attention



Aside: CNNs were an example of hard attention



From
RNN:

p_a	p_b
p_c	p_d

Xu et al, "Show, Attend and Tell:
Neural Image Caption Generation
with Visual Attention", ICML 2015

Distribution over
grid locations
 $p_a + p_b + p_c + p_d = 1$

Hard attention:
Sample ONE location
according to p , $z =$ that vector

With argmax, dz/dp is zero
almost everywhere ...
Can't use gradient descent;
need reinforcement learning

Soft attention:
Summarize ALL locations

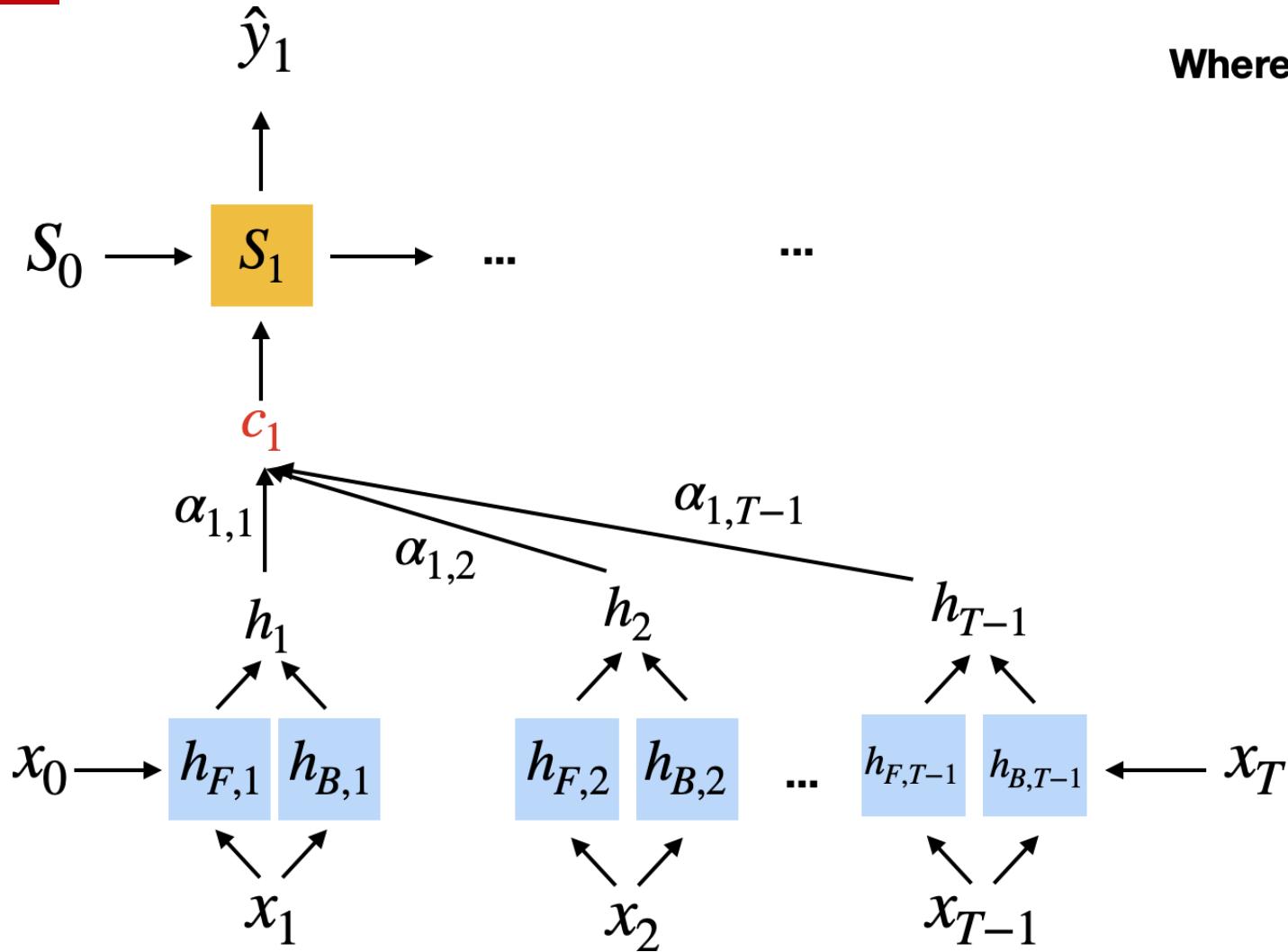
$$z = p_a a + p_b b + p_c c + p_d d$$

Derivative dz/dp is nice!
Train with gradient descent



Self-Attention

"Original" (RNN) Attention Mechanism

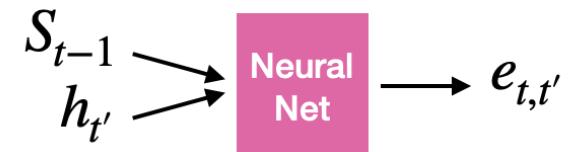


Where the context vector c_1 is defined as

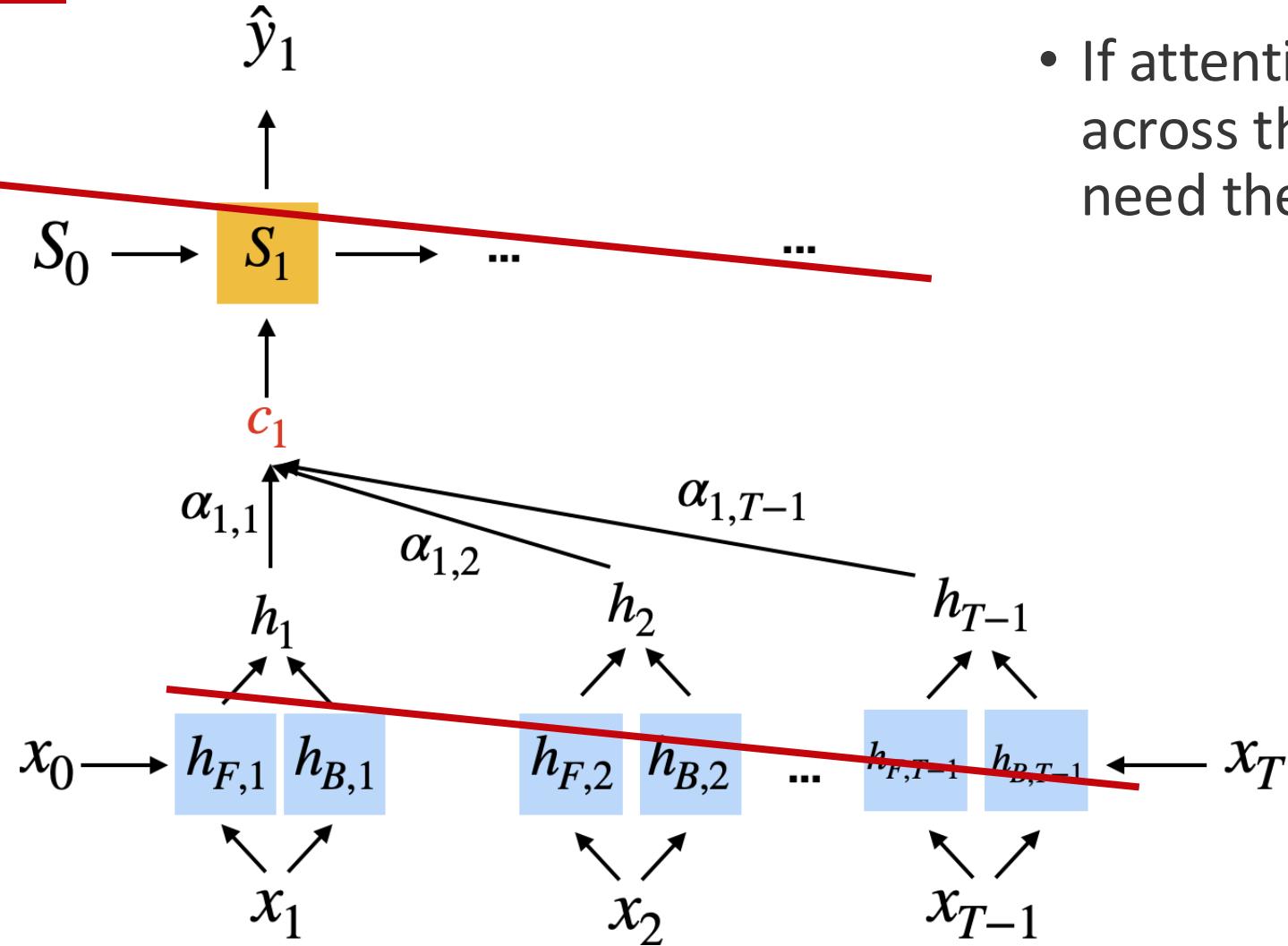
$$c_1 = \sum_{t=1}^T \alpha_{1,t} h_t$$

And the attention weights are

$$\alpha_{t,t'} = \frac{\exp(e_{t,t'})}{\sum_{t'=1}^T \exp(e_{t,t'})}$$



Can we get rid of the sequential parts?



- If attention already ties inputs across the sequence, do we really need the recurrence?



Self-attention (very basic form)

Main procedure:

- 1) Derive attention weights: similarity between current input and all other inputs (next slide)
- 2) Normalize weights via softmax (next slide)
- 3) Compute attention value from normalized weights and corresponding inputs (below)

Self-attention as weighted sum:

output corresponding to the i -th input

$$A_i = \sum_{j=1}^T a_{ij} x_j$$

weight based on similarity between current input x_i and all other inputs



Self-attention (very basic form)

Self-attention as weighted sum:

output corresponding to the i -th input

$$A_i = \sum_{j=0}^T a_{ij} x_j$$

weight based on similarity between current input x_i and all other inputs

How to compute the attention weights?

here as simple dot product:

$$e_{ij} = x_i^\top x_j$$

repeat this for all inputs $j \in \{1 \dots T\}$, then normalize

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{j=1}^T \exp(e_{ij})} = \text{softmax}\left(\left[e_{ij}\right]_{j=1 \dots T}\right)$$

Self-attention (very basic form)

No learnable parameters?

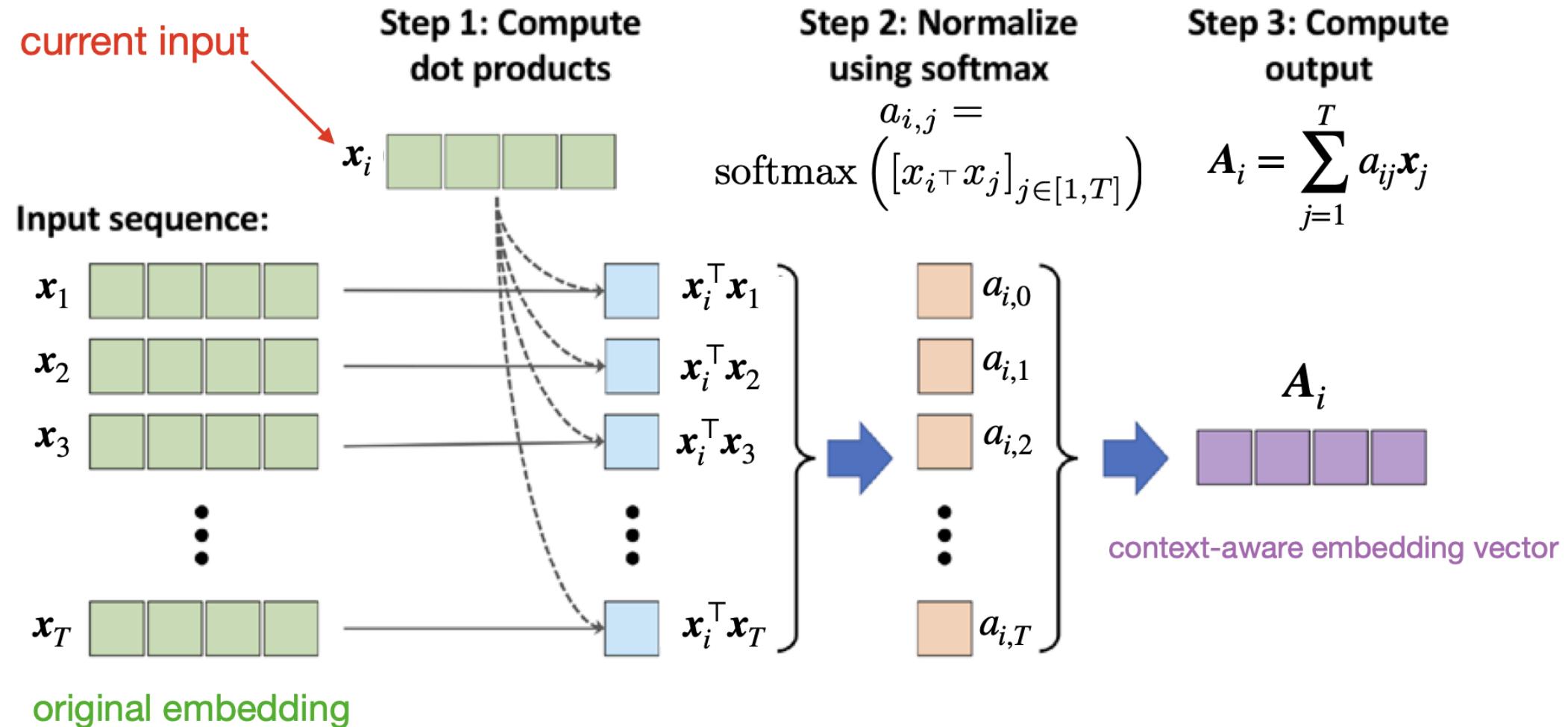


Image source: Raschka & Mirjalili 2019. Python Machine Learning, 3rd edition



Learnable Self-attention

- Previous basic version did not involve any learnable parameters, so not very useful for learning a language model
- We are now adding 3 trainable weight matrices that are multiplied with the input sequence embeddings

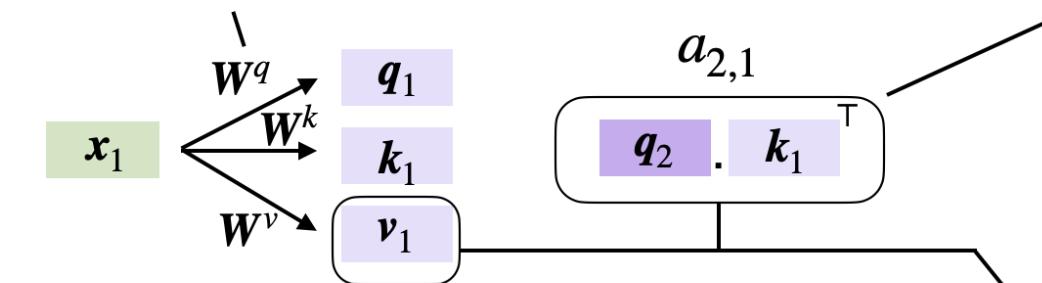
$$\text{query} = W^q \mathbf{x}_i$$

$$\text{key} = W^k \mathbf{x}_i$$

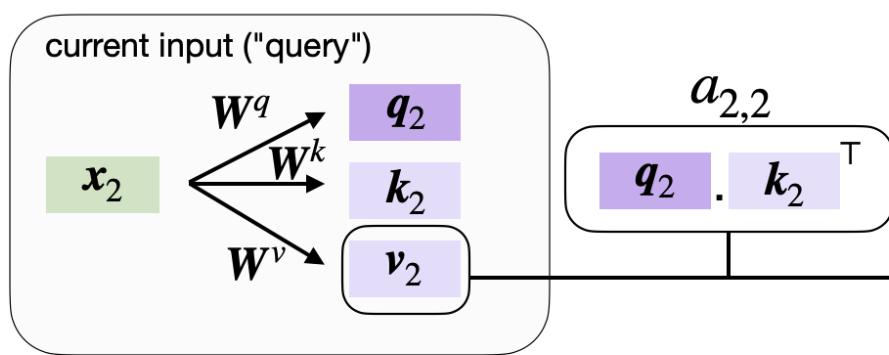
$$\text{value} = W^v \mathbf{x}_i$$

Learnable Self-attention

trainable weight matrices



As in the simplified version, this is a form of similarity or compatibility measure ("multiplicative attention")

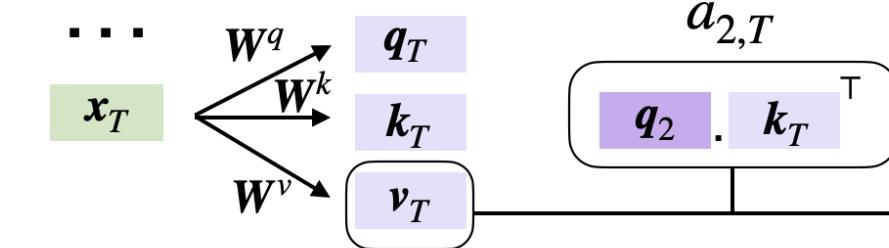


For each query, model learns which **key-value** input it should attend to

$$A_2$$

$$A(q_2, \mathbf{K}, \mathbf{V}) = \sum_{i=1}^T \left[\frac{\exp(q_2 \cdot k_i^\top)}{\sum_j \exp(q_2 \cdot k_j^\top)} \times v_i \right]$$

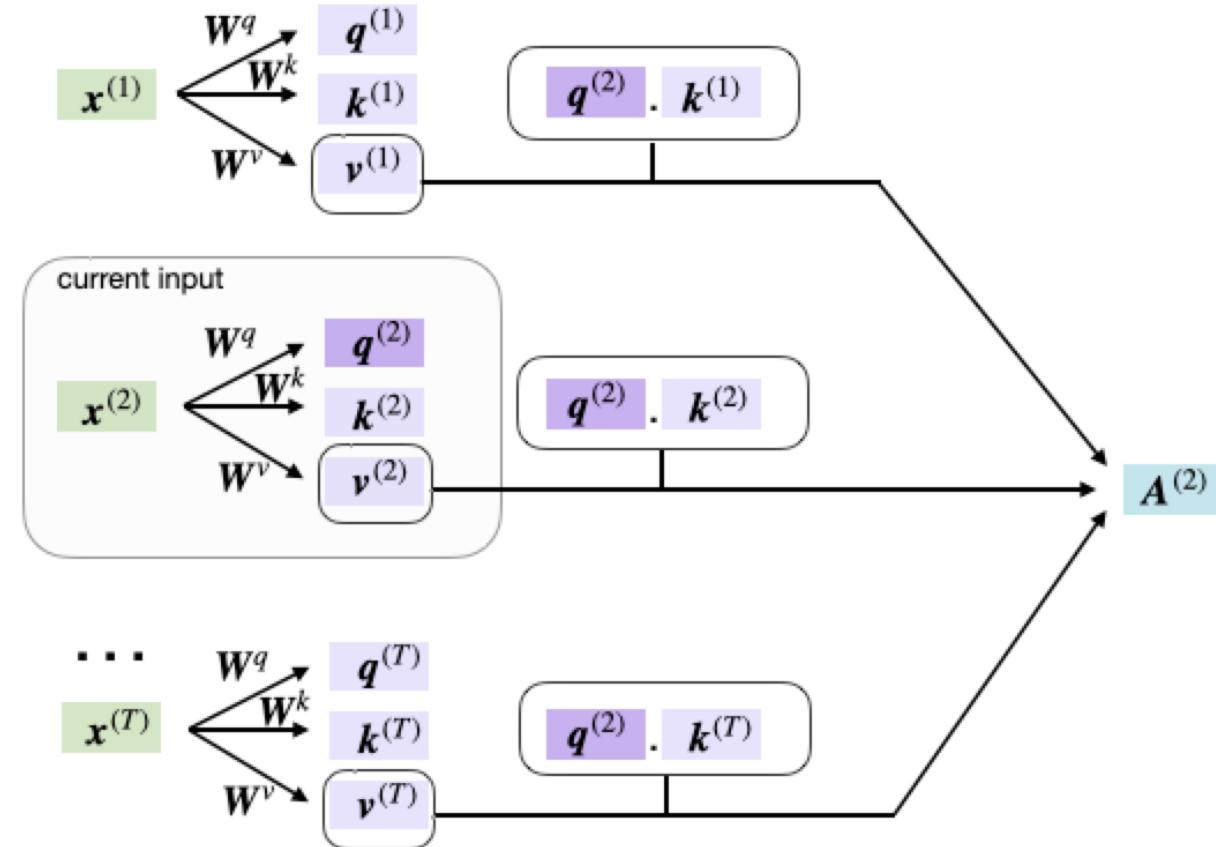
softmax



weighted sum: values weighted by attention weight (softmax score)

Learnable Self-attention

"self"-attention because
input to query and key-value pair
are the same

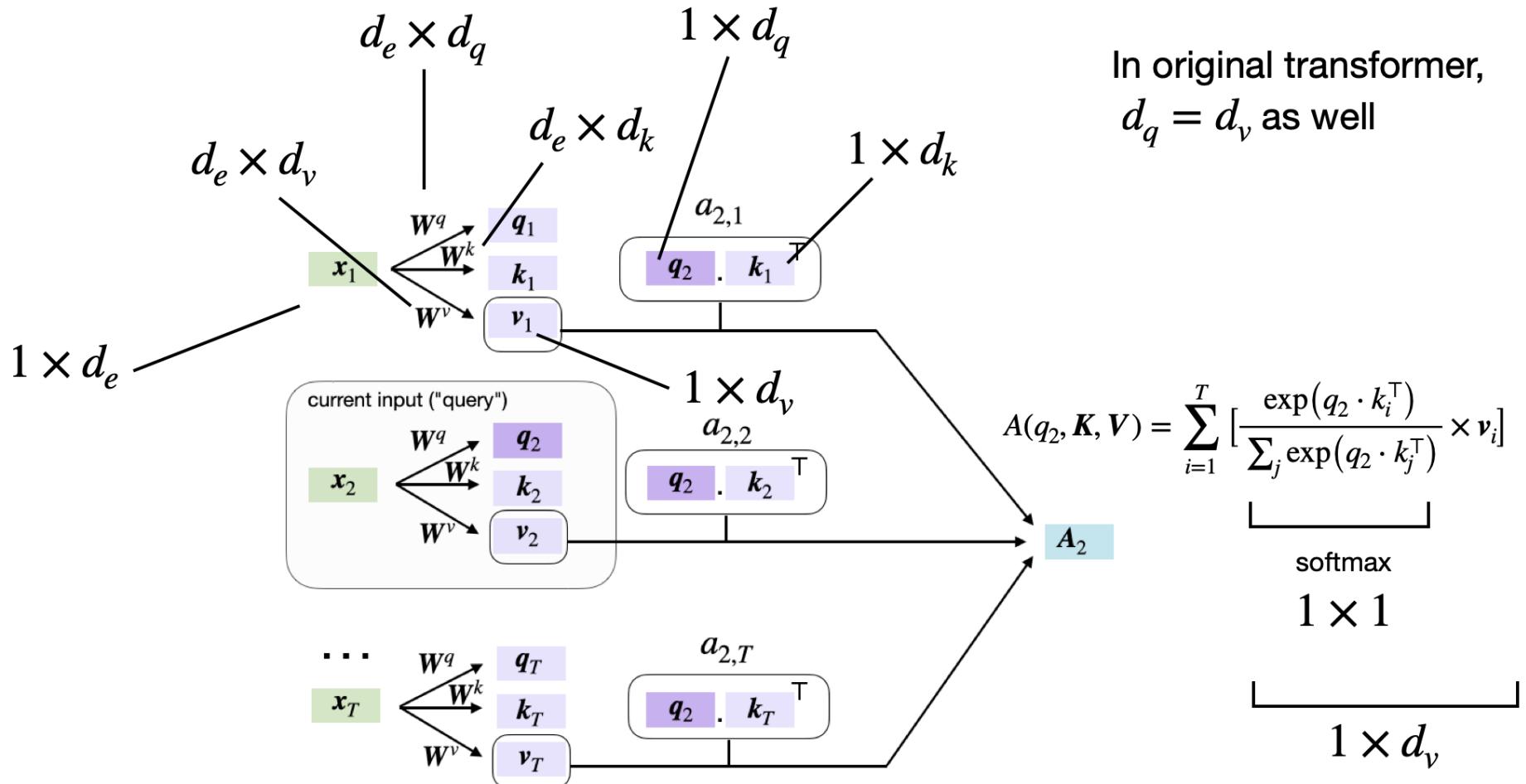


Learnable Self-attention

d_e = embedding size (original transformer = 512)

where $d_q = d_k$

In original transformer,
 $d_q = d_v$ as well



At the end of the day...

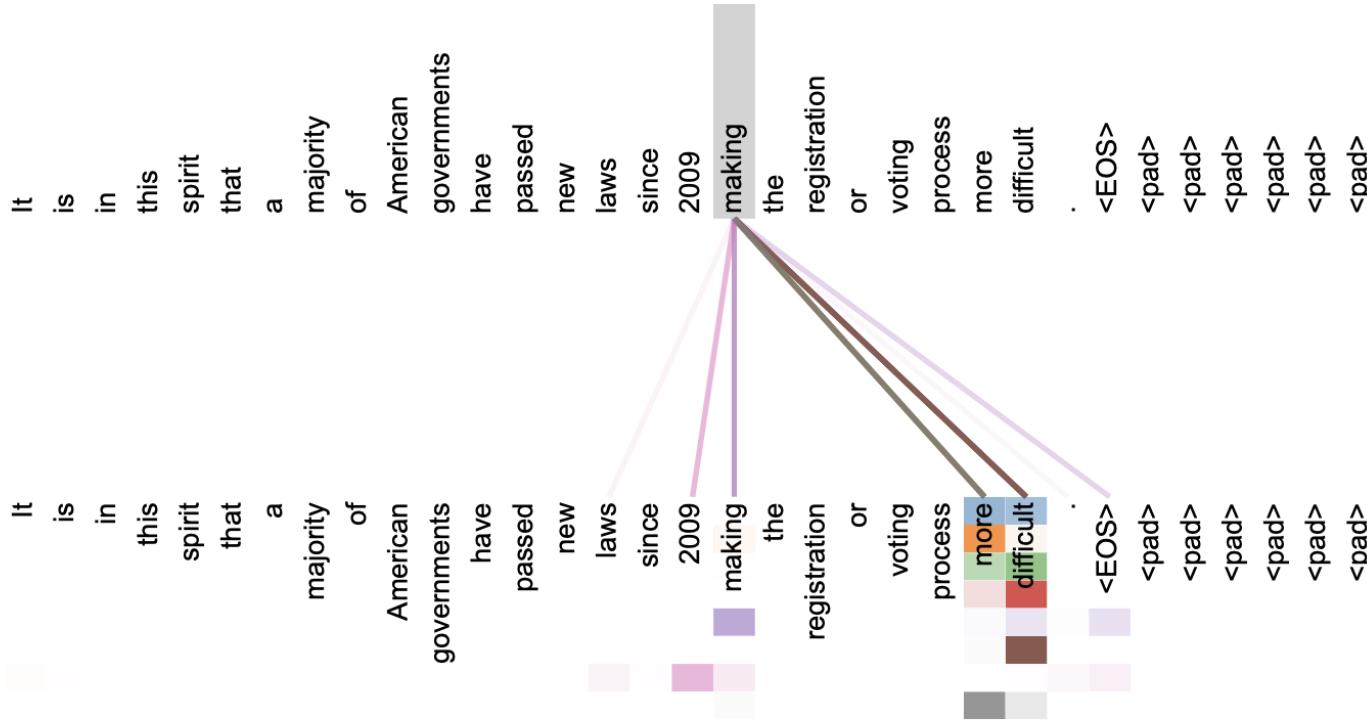


Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb ‘making’, completing the phrase ‘making...more difficult’. Attentions here shown only for the word ‘making’. Different colors represent different heads. Best viewed in color.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).



The "Transformer"

Attention Is All You Need

Ashish Vaswani*

Google Brain

avaswani@google.com

Noam Shazeer*

Google Brain

noam@google.com

Niki Parmar*

Google Research

nikip@google.com

Jakob Uszkoreit*

Google Research

usz@google.com

Llion Jones*

Google Research

llion@google.com

Aidan N. Gomez* †

University of Toronto

aidan@cs.toronto.edu

Lukasz Kaiser*

Google Brain

lukaszkaiser@google.com

Illia Polosukhin* ‡

illia.polosukhin@gmail.com

Attention is all you need

[A Vaswani, N Shazeer, N Parmar... - Advances in neural ...](#), 2017 - proceedings.neurips.cc

... to attend to **all** positions in the decoder up to and including that position. **We need** to prevent

... **We** implement this inside of scaled dot-product **attention** by masking out (setting to $-\infty$) ...

[☆ Save](#) [99 Cite](#) [Cited by 174852](#) [Related articles](#) [All 73 versions](#) [»](#)

<https://arxiv.org/abs/1706.03762>



The Transformer



The "Transformer"

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

Attention is all you need

[A Vaswani, N Shazeer, N Parmar... - Advances in neural ...](#), 2017 - proceedings.neurips.cc

... to attend to **all** positions in the decoder up to and including that position. **We need** to prevent
... **We implement** this inside of scaled dot-product **attention** by masking out (setting to $-\infty$) ...

☆ Save 99 Cite Cited by 174852 Related articles All 73 versions ➔

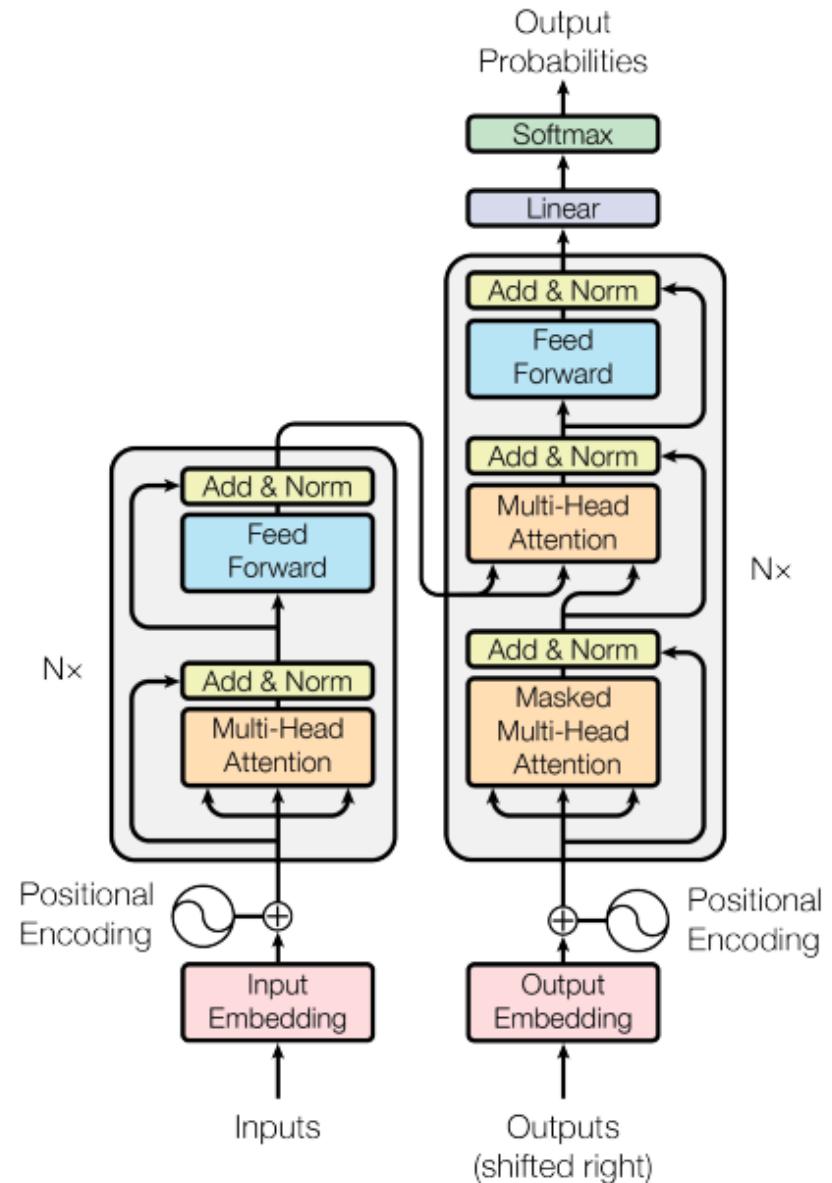
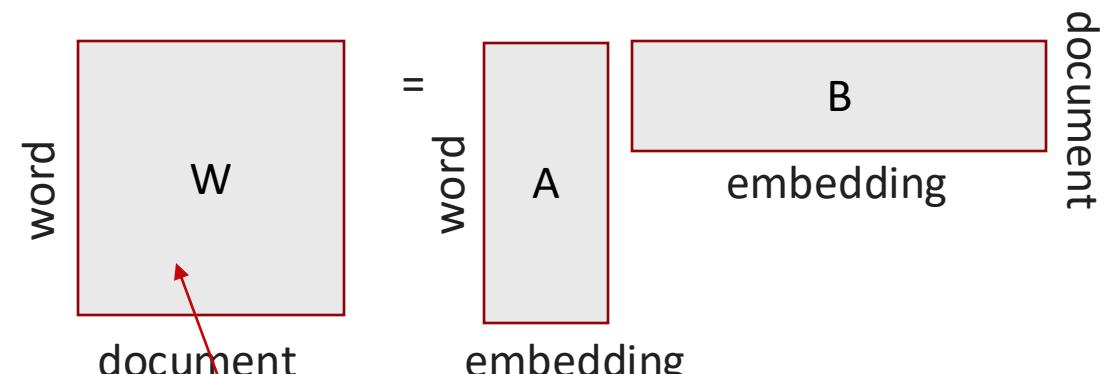


Figure 1: The Transformer - model architecture.

Start with word embeddings...

- Lookup table that translates words (or more formally “tokens”) into continuous-valued “embeddings”
- Simplest form: random embeddings
- Slightly better: TF-IDF embeddings
- Many ways to improve pre-trained embeddings



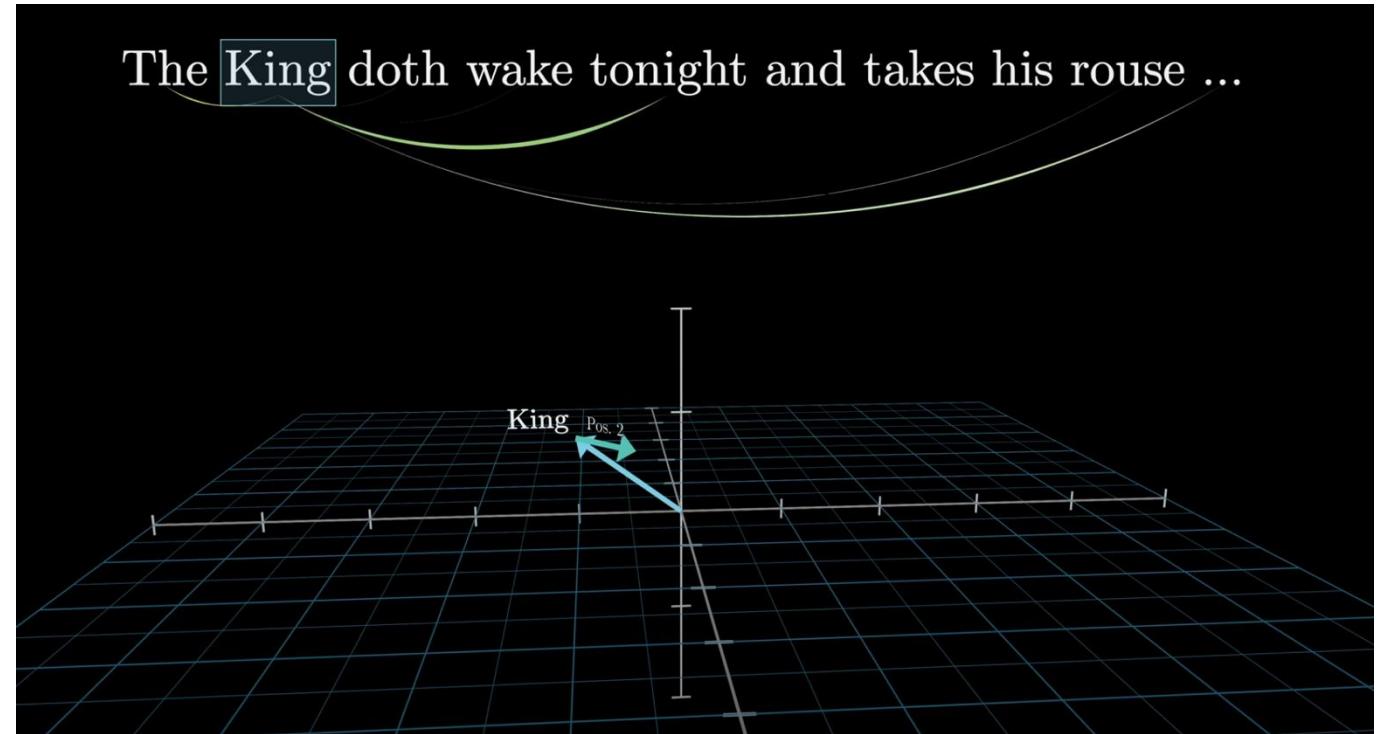
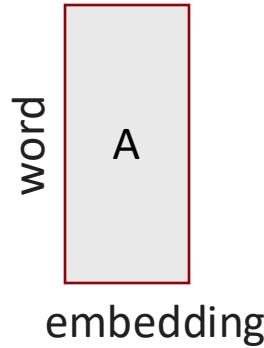
$$w_{x,y} = \text{tf}_{x,y} \times \log \left(\frac{N}{df_x} \right)$$

TF-IDF
Term x within document y

$\text{tf}_{x,y}$ = frequency of x in y
 df_x = number of documents containing x
 N = total number of documents

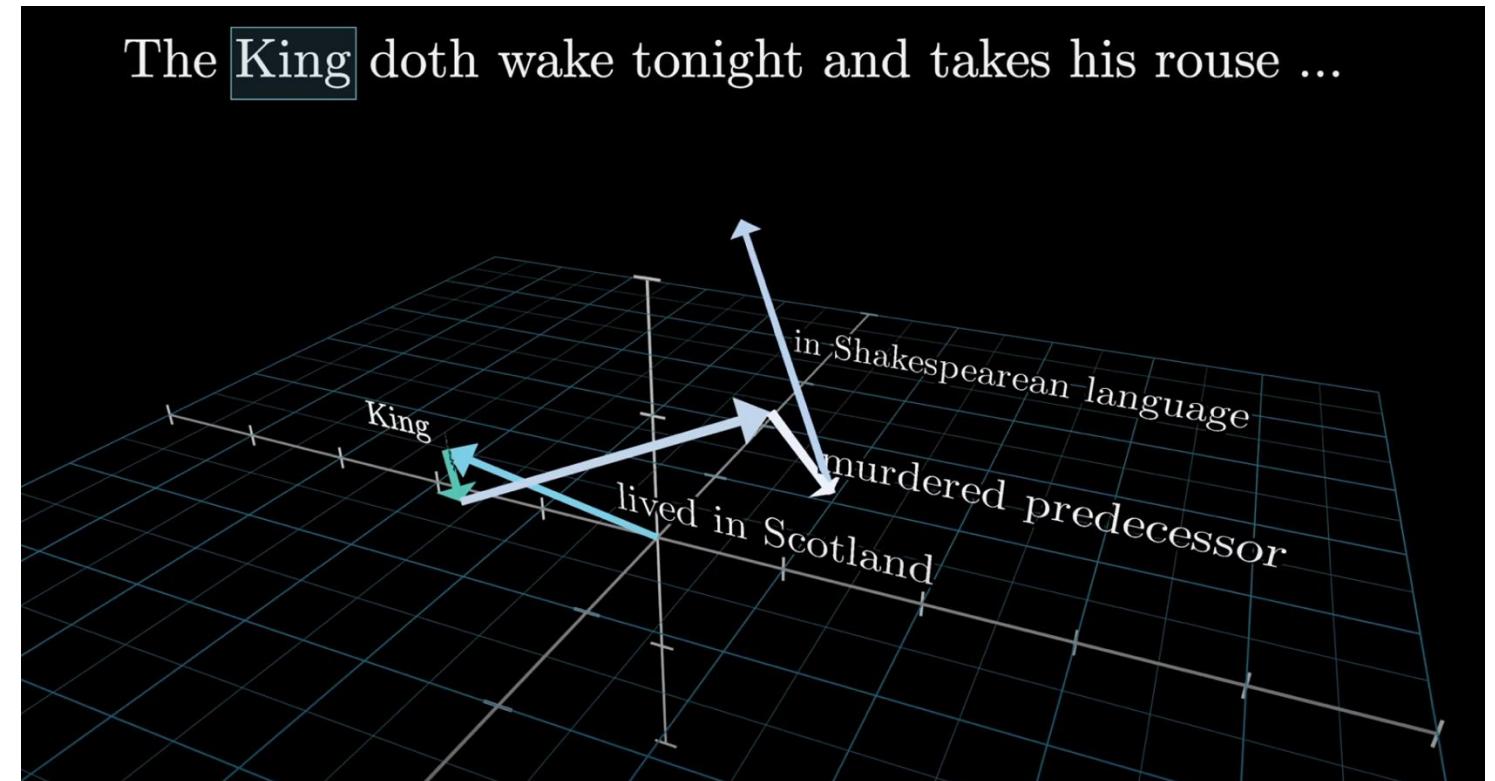
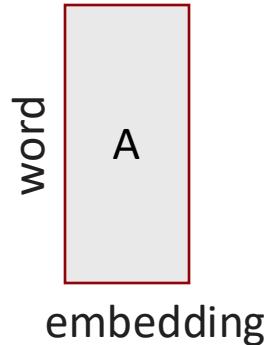


Start with word embeddings...



3Blue1Brown ["Attention in Transformers"](#)

Update embeddings by context



3Blue1Brown "[Attention in Transformers](#)"

Multi-headed Attention

- Apply self-attention multiple times in parallel (similar to multiple kernels for channels in CNNs)
- For each head (self-attention layer), use different , then concatenate the results,
- 8 attention heads in the original transformer
- Allows attending to different parts in the sequence differently

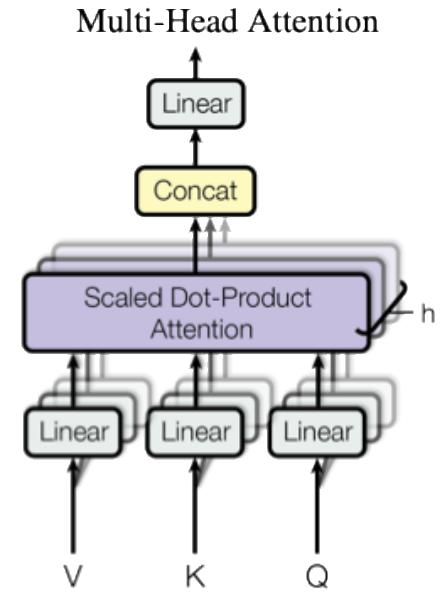
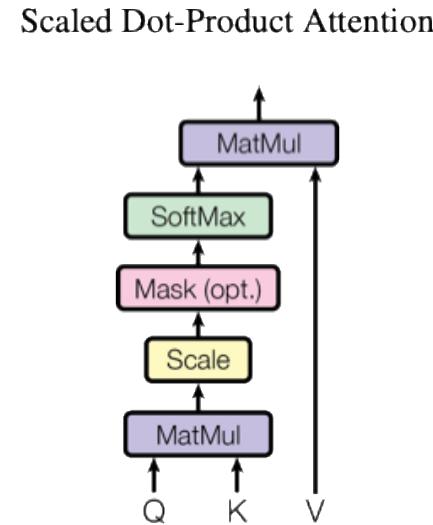
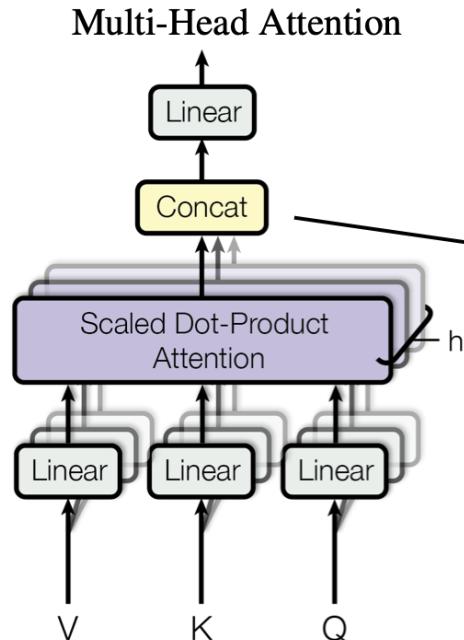


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., and Polosukhin, I., 2017. Attention Is All You Need.

Multi-headed Attention



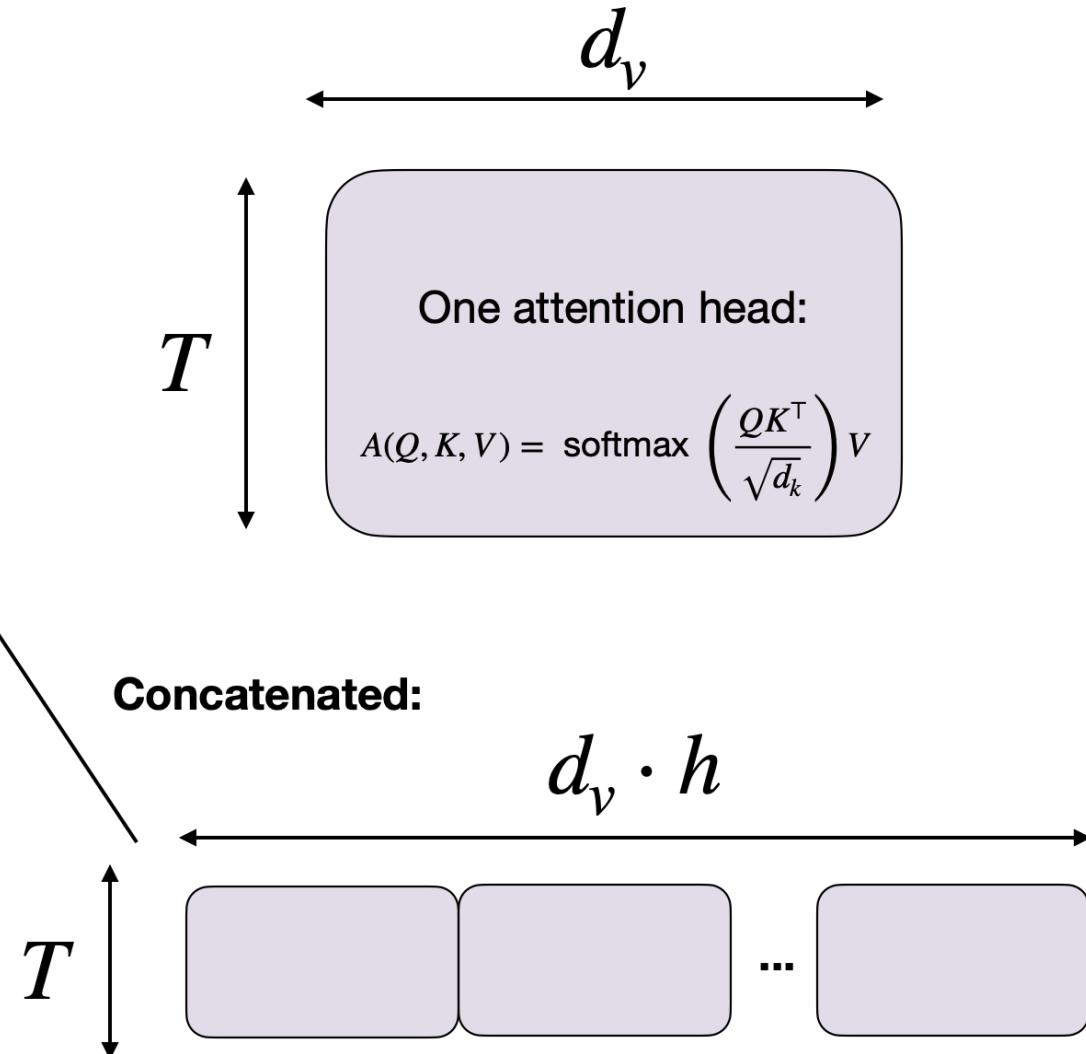
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I., 2017. Attention Is All You Need.

Input sequence dim.
in original transformer:

$$T \times d_e = T \times 512$$

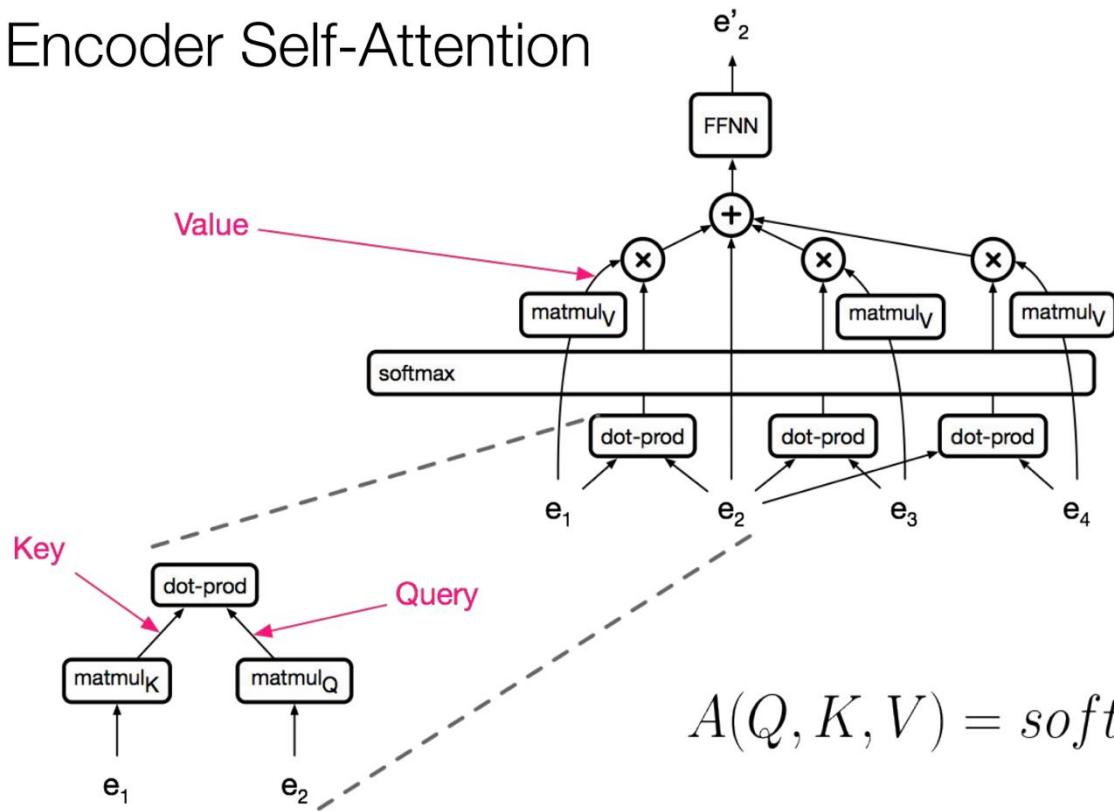
and

$$d_v = 512/h = 64$$



The "Transformer": Encoder

Encoder Self-Attention



Vaswani ["Self-Attention for Generative Models"](#)

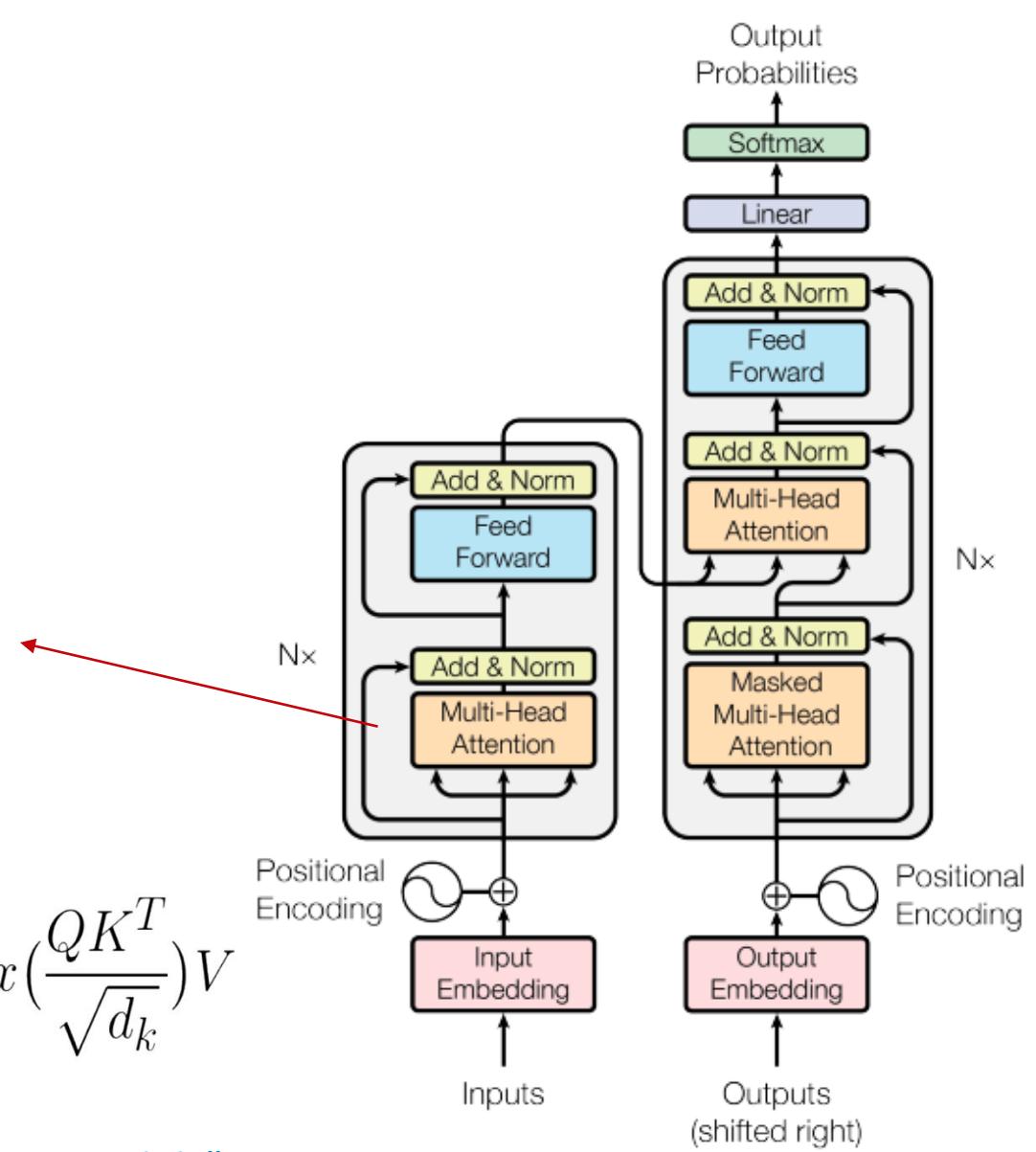
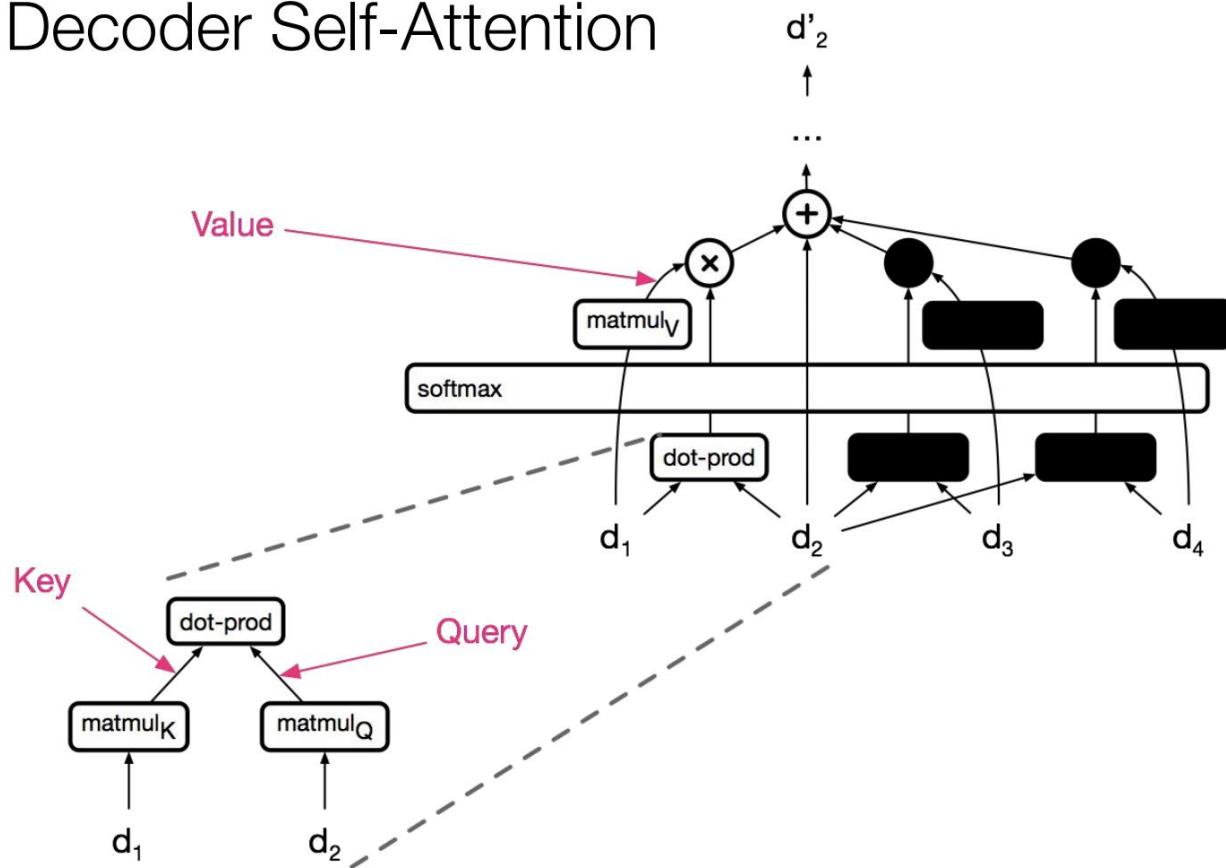


Figure 1: The Transformer - model architecture.

The "Transformer": Decoder

Decoder Self-Attention



Vaswani ["Self-Attention for Generative Models"](#)

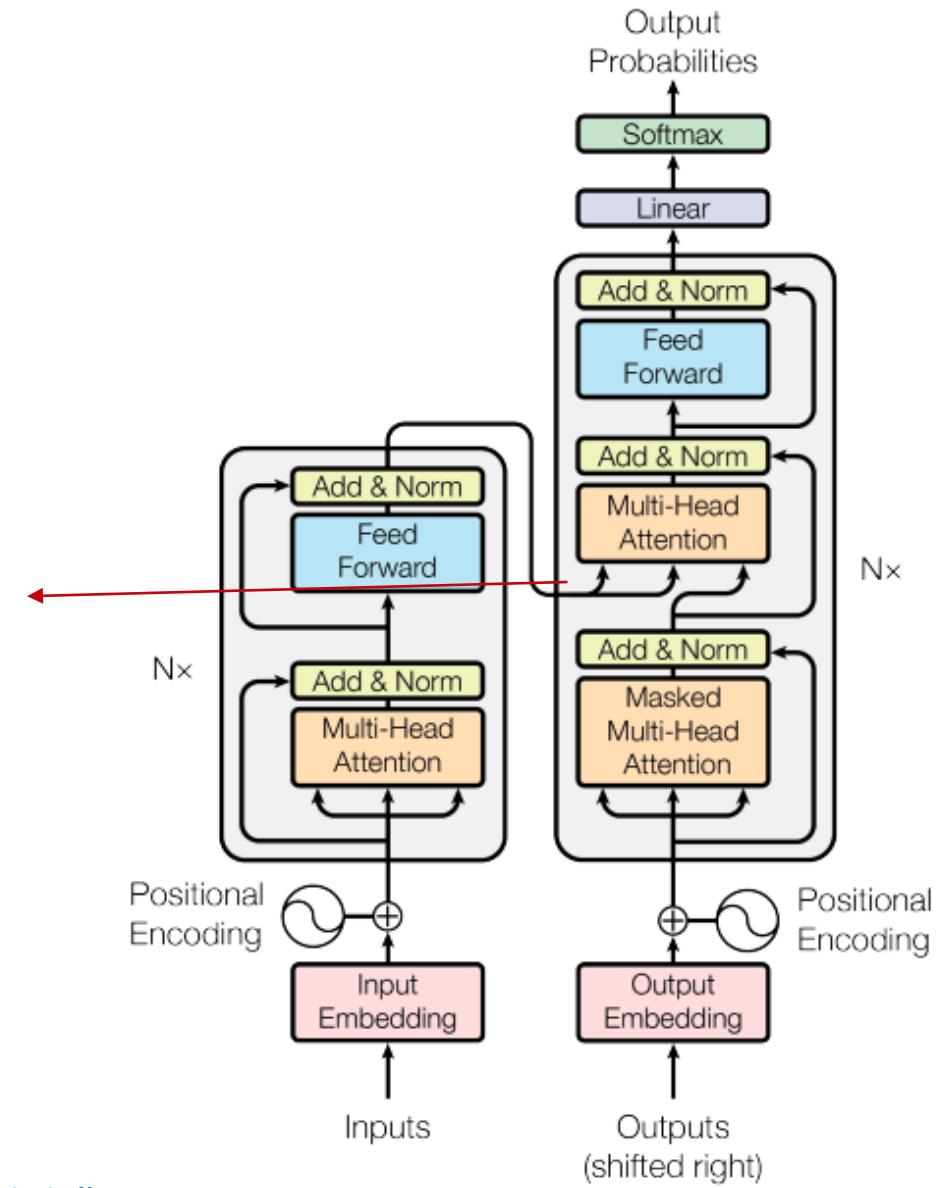


Figure 1: The Transformer - model architecture.



Transformer Tricks (4?)

- **Self Attention:** Each layer combines words with others
- **Multi-headed Attention:** 8 attention heads learned independently
- **Normalized Dot-product Attention:** Remove bias in dot product when using large networks
- **Positional Encodings:** Make sure that even if we don't have RNN, can still distinguish positions

Transformer Tricks – Positional Encodings

- Scaled dot-product and fully-connected layer are permutation invariant
- Sinusoidal positional encoding is a vector of small values (constants) added to the embeddings
- As a result, same word will have slightly different embeddings depending on where they occur in the sentence

$$PE_{pos,2i} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

$$PE_{pos,2i+1} = \cos\left(\frac{pos}{10000^{\frac{(2i+1)}{d_{model}}}}\right)$$

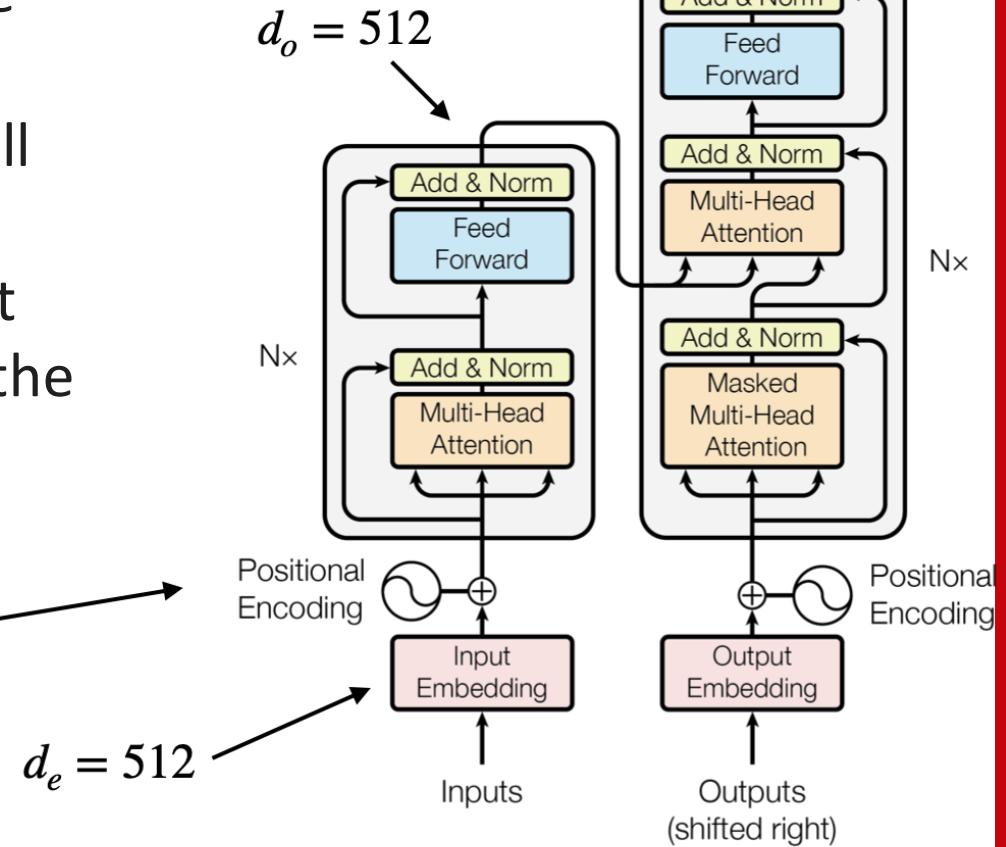


Figure 1: The Transformer - model architecture.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I., 2017. Attention Is All You Need.



Transformer Tricks (3?)

Transformer Language Models without Positional Encodings Still Learn Positional Information

Adi Haviv^T Ori Ram^T Ofir Press^ω Peter Izsak^I Omer Levy^{T,μ}

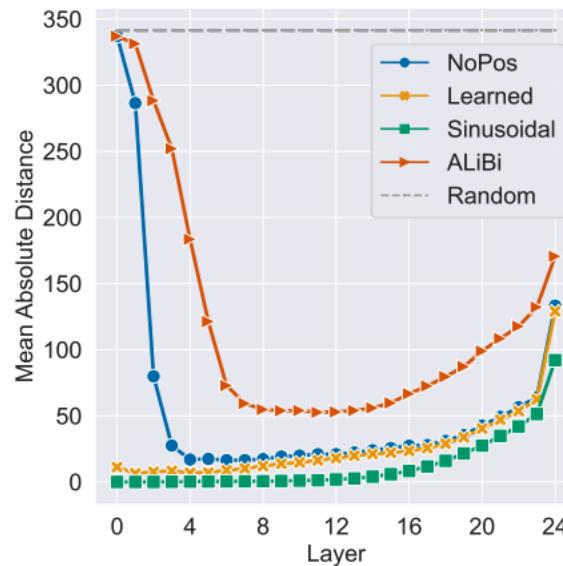
^TTel Aviv University

^ωUniversity of Washington

^IIntel Labs

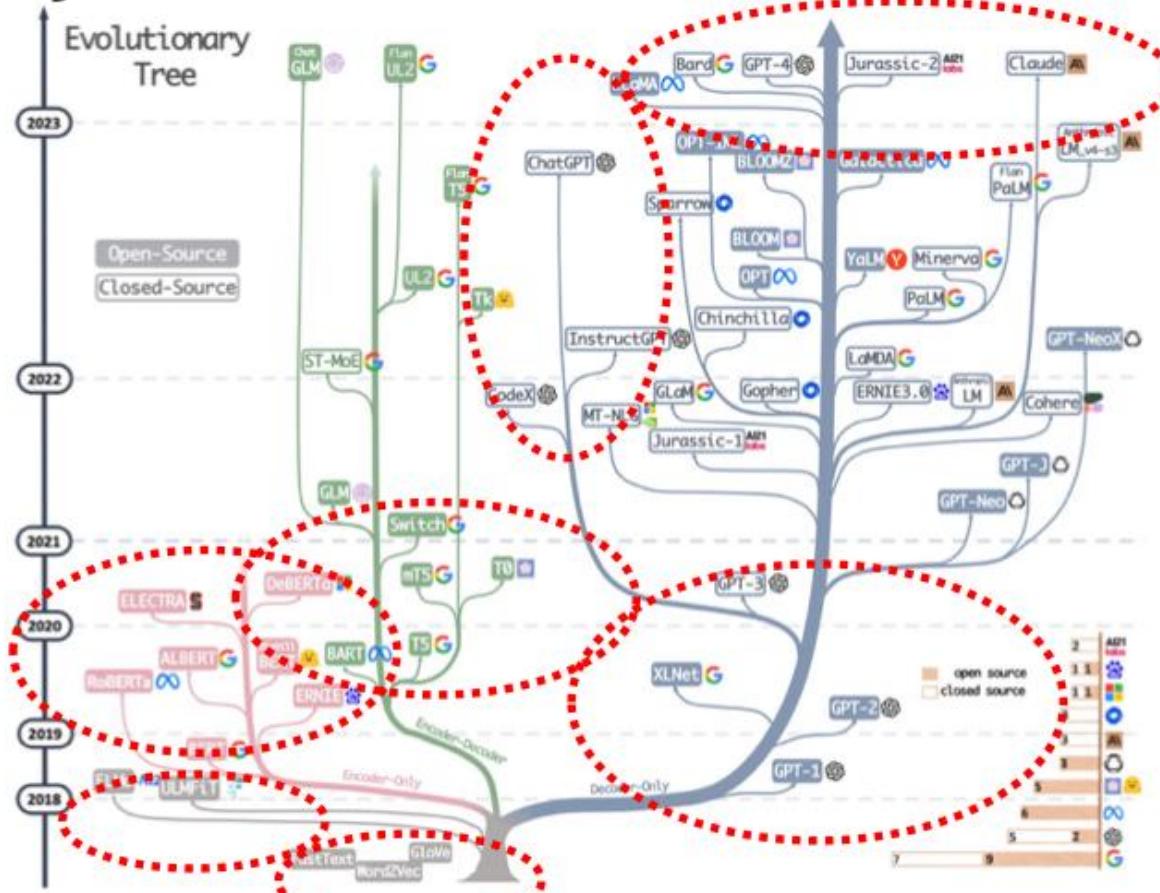
^μMeta AI

(adi.haviv, ori.ram, levyomer)@cs.tau.ac.il, ofirp@cs.washington.edu, peter.izsak@intel.com



Many variants of transformer architectures

Many variants: encoder, decoder, or both



Yang et al., 2023, arxiv: 2304.13712



Next Time: GPT

Generative Pre-Trained Transformer

Questions?

