STAT 453: Introduction to Deep Learning and Generative Models

Ben Lengerich

Lecture 17: Attention & Transformers

November 5, 2026

Reading: See course homepage

Last time: Recurrent Neural Networks (RNNs)

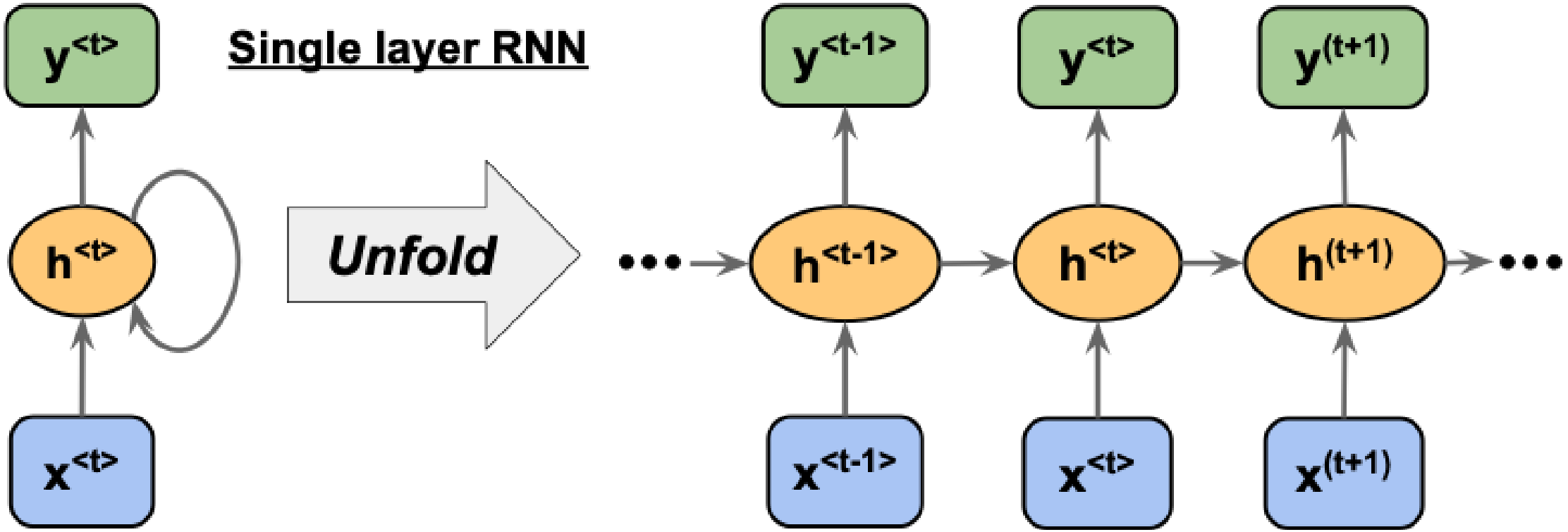


Image source: Sebastian Raschka, Vahid Mirjalili. Python Machine Learning. 3rd Edition. Packt, 2019

Ben Lengerich © University of Wisconsin-Madison 2026

Where we're going: From RNN…

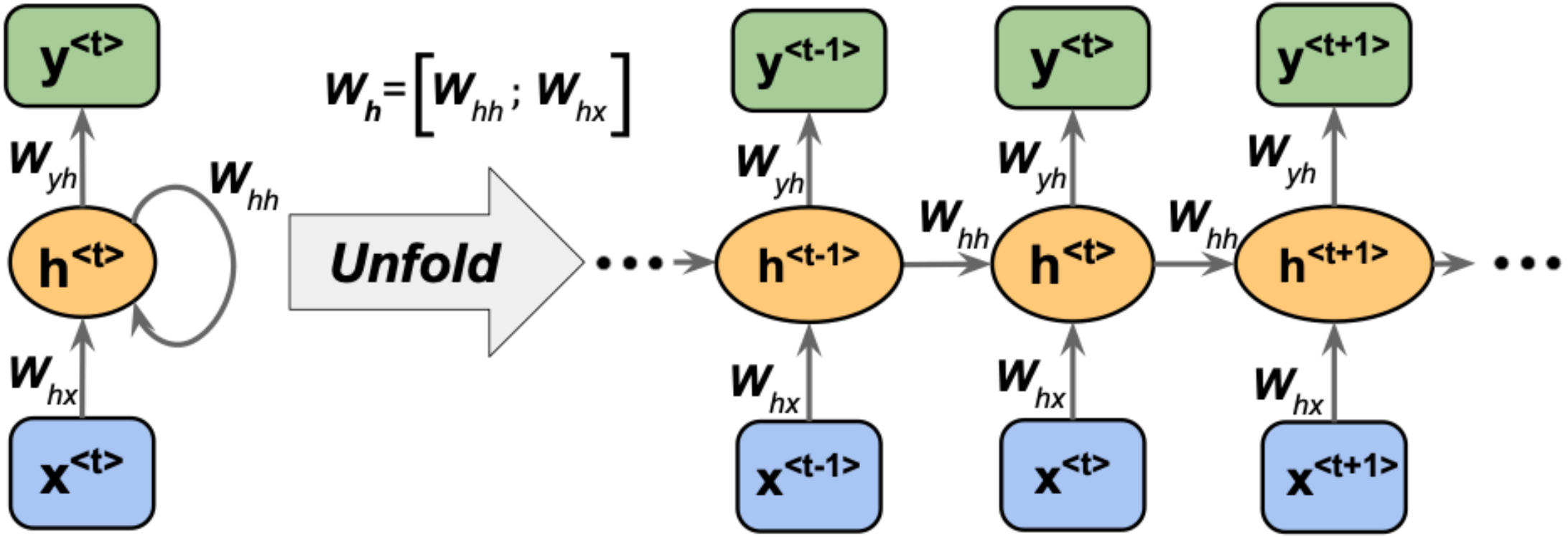


Image source: Sebastian Raschka, Vahid Mirjalili. Python Machine Learning. 3rd Edition. Packt, 2019

Ben Lengerich © University of Wisconsin-Madison 2026

Where we're going: From RNN…to GPT

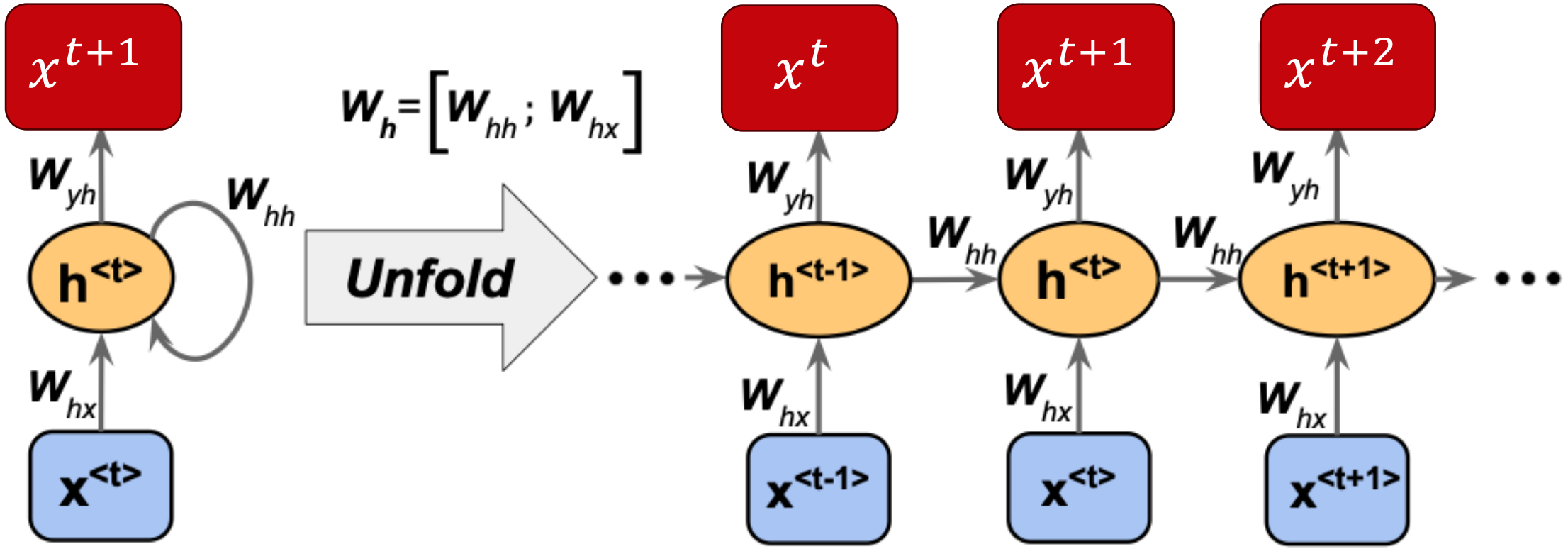
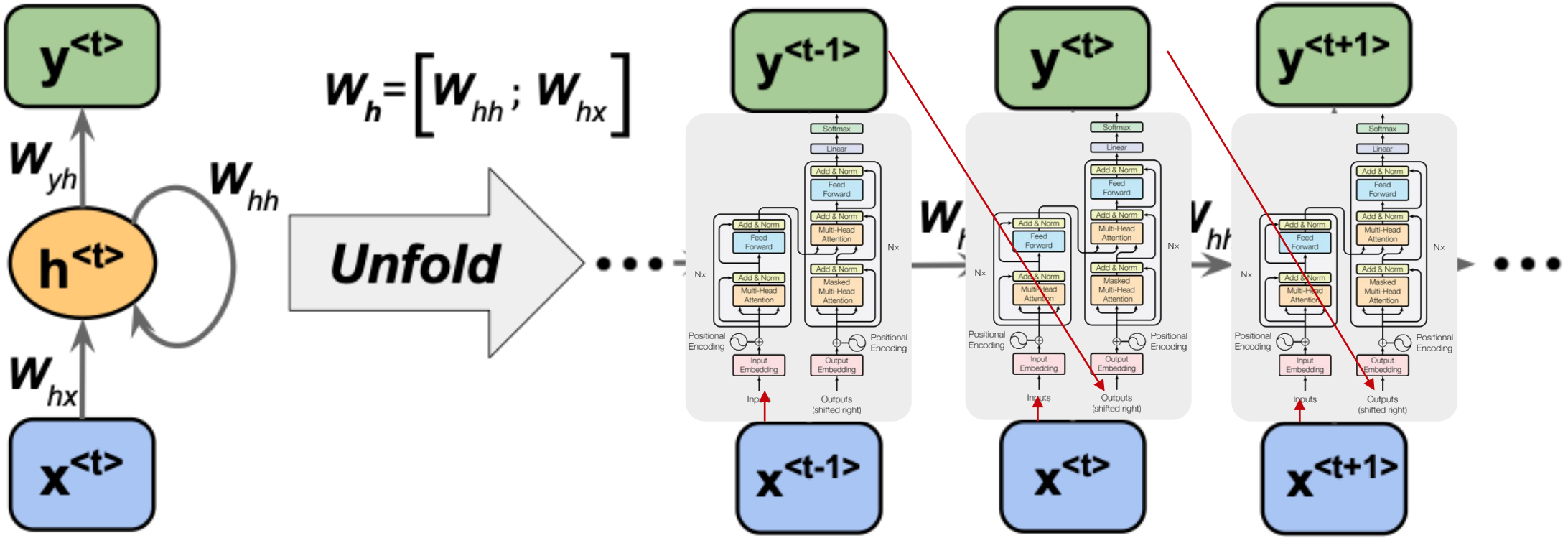


Image source: Sebastian Raschka, Vahid Mirjalili. Python Machine Learning. 3rd Edition. Packt, 2019

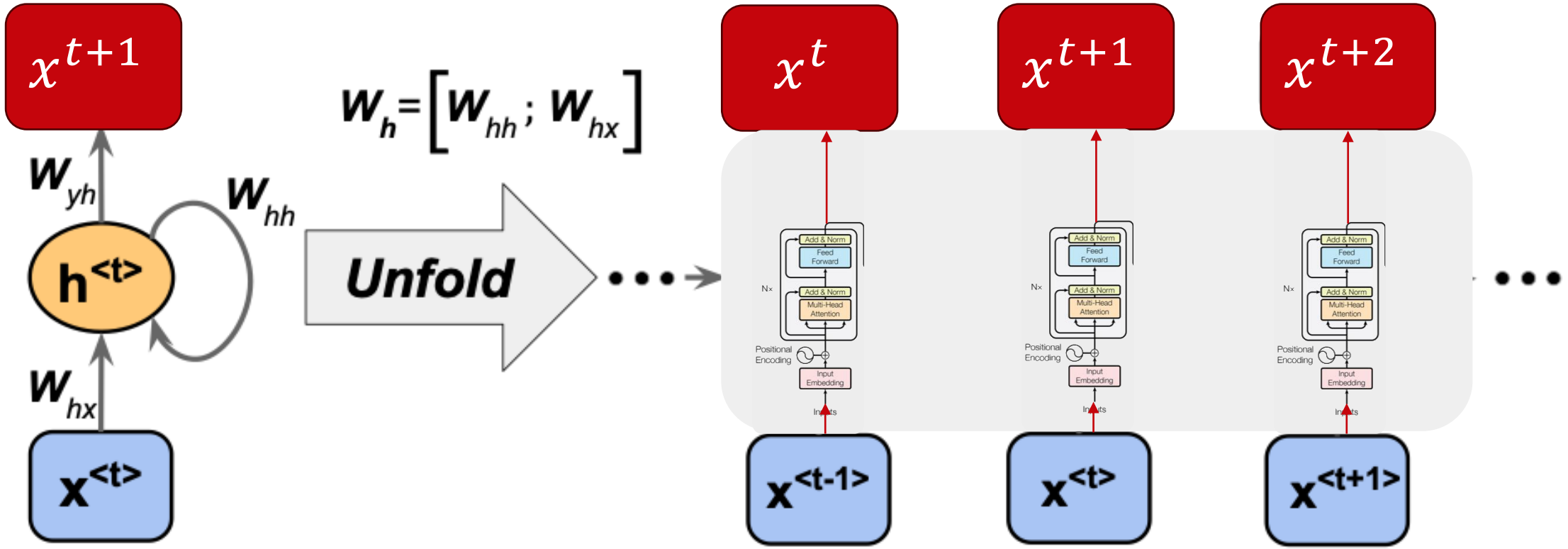
Where we're going: From RNN…to GPT…by Transformers



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

Where we're going: From RNN…to GPT…by Transformers



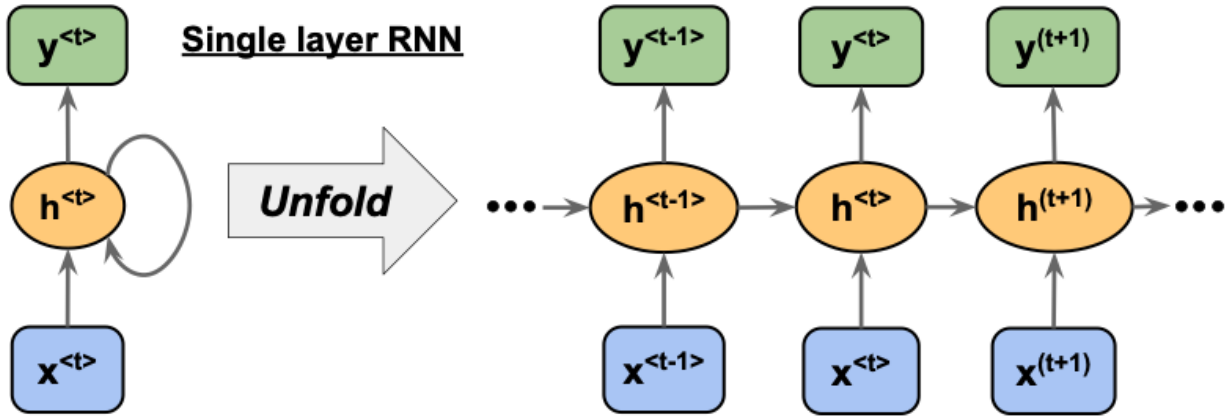
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I., 2017. Attention Is All You Need.

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

The Attention Mechanism

Why Attention?

- Consider machine translation:
  - What if we need the whole sequence to translate a 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.



Ben Lengerich © University of Wisconsin-Madison 2026

“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

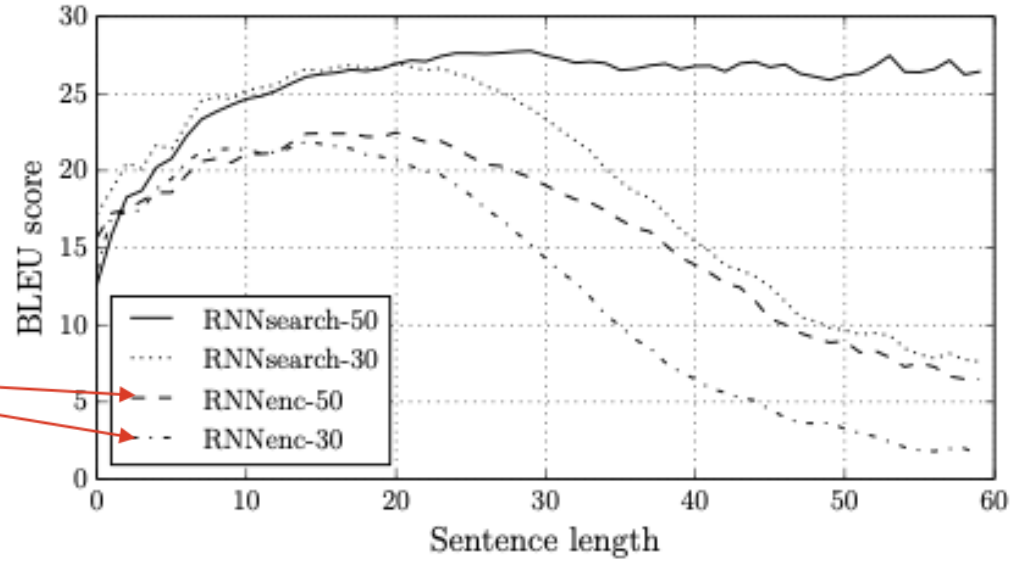


Figure 2: The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models.

“Attention”

Original idea:

Update forward-RNN hidden state with “attention” state from bi-directional RNN that looks at entire input sequence.

Ben Lengerich © University of Wisconsin-Madison 2026

“Attention”

Original idea:

Update forward-RNN hidden state with “attention” state from bi-directional RNN that looks at entire input sequence.

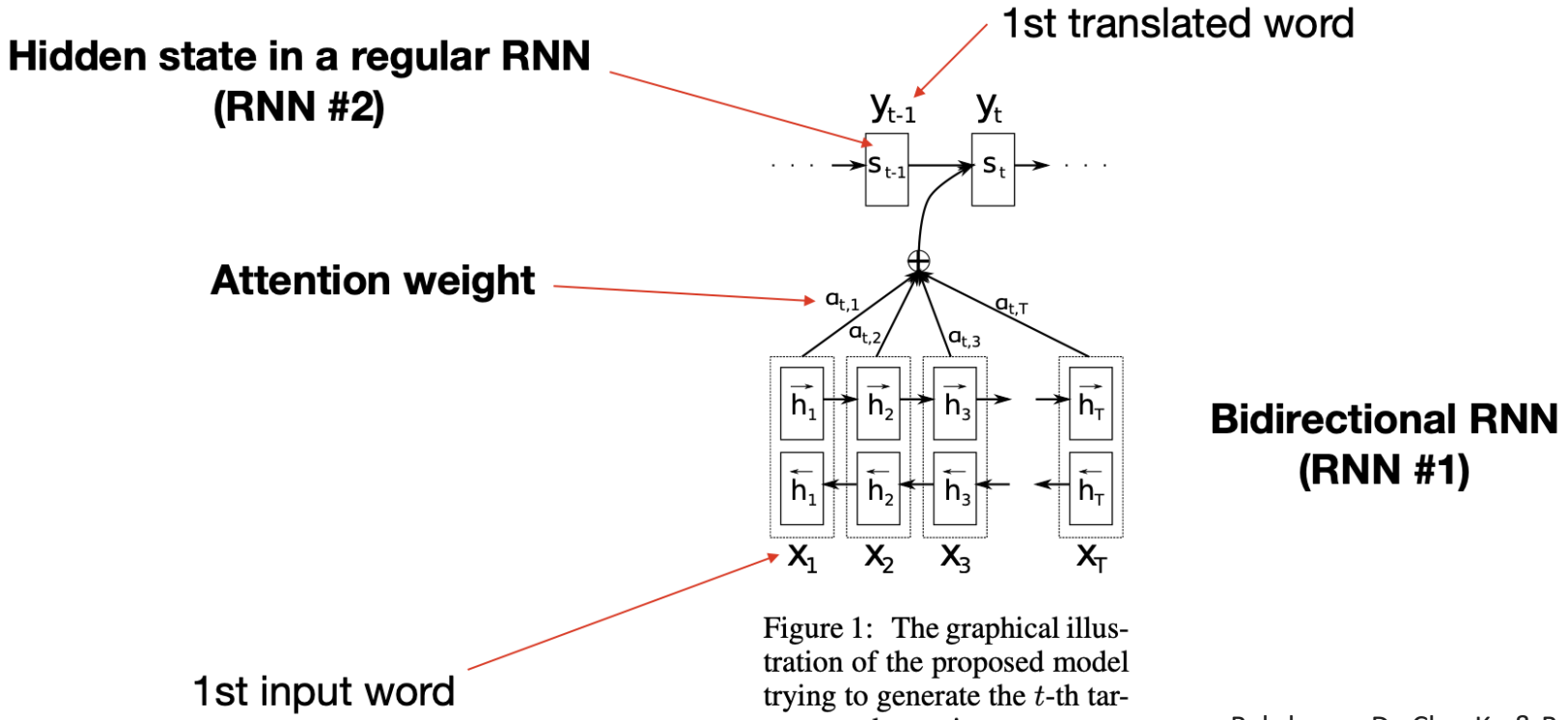


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

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

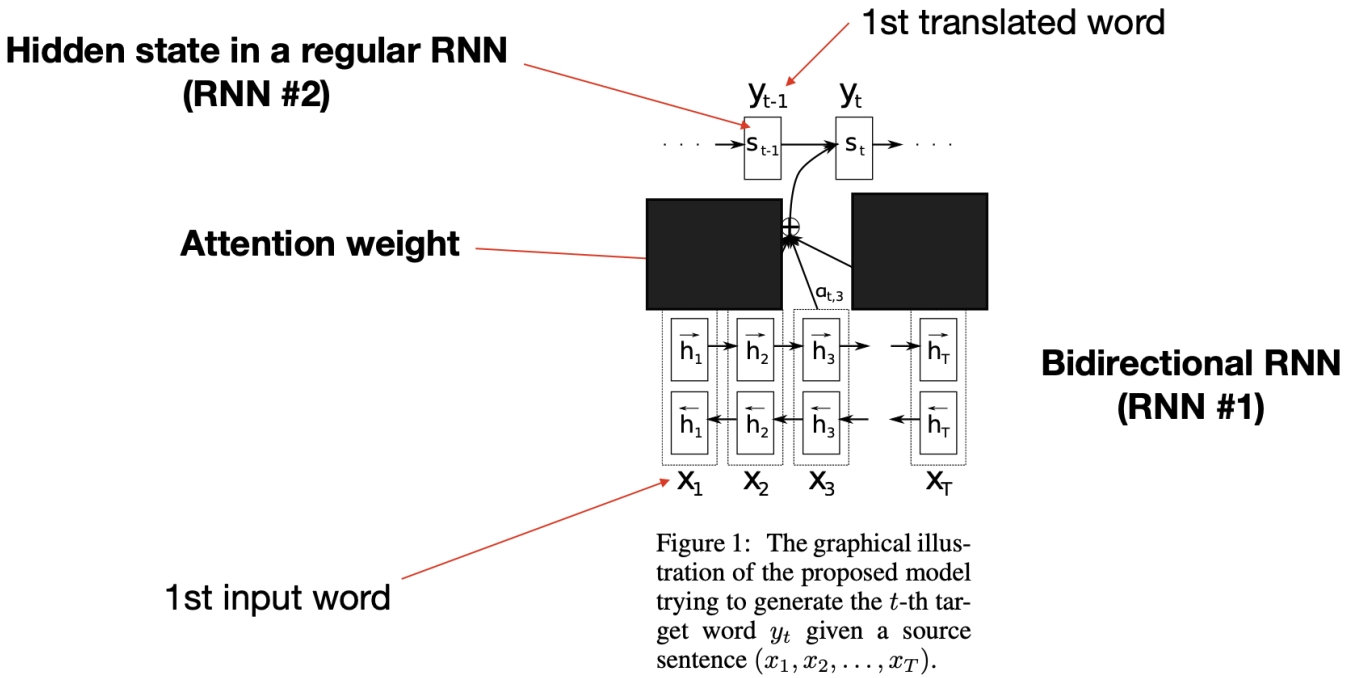
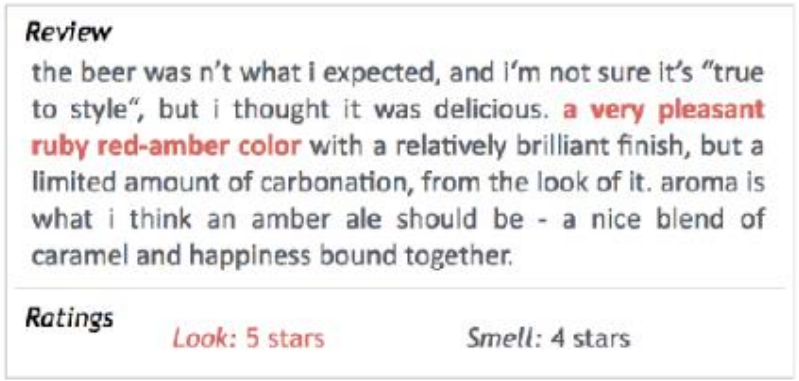


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

Benefit: Interpretable?



Lei et al 2016

Soft attention

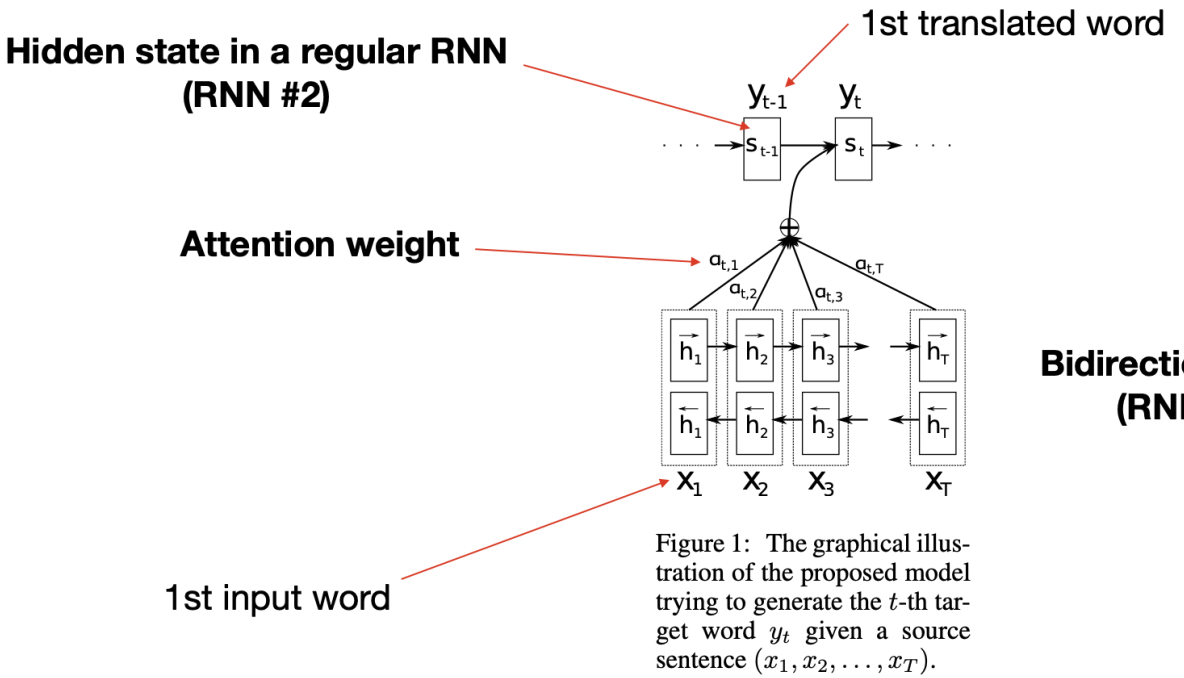
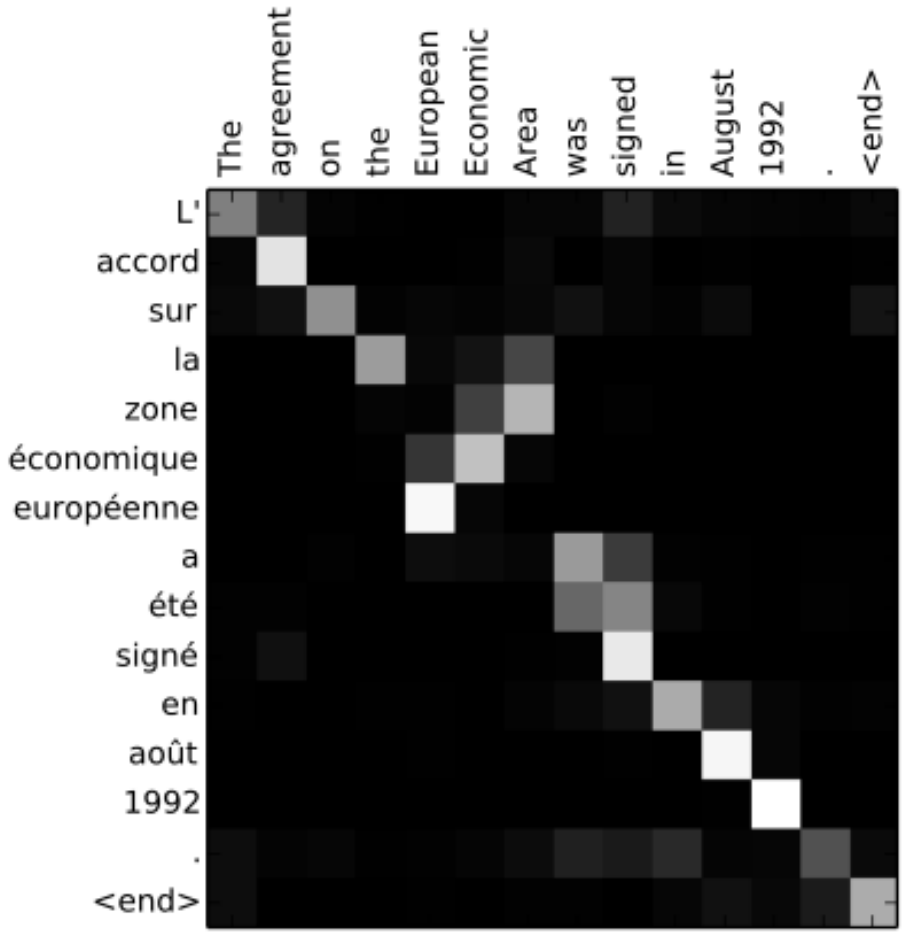
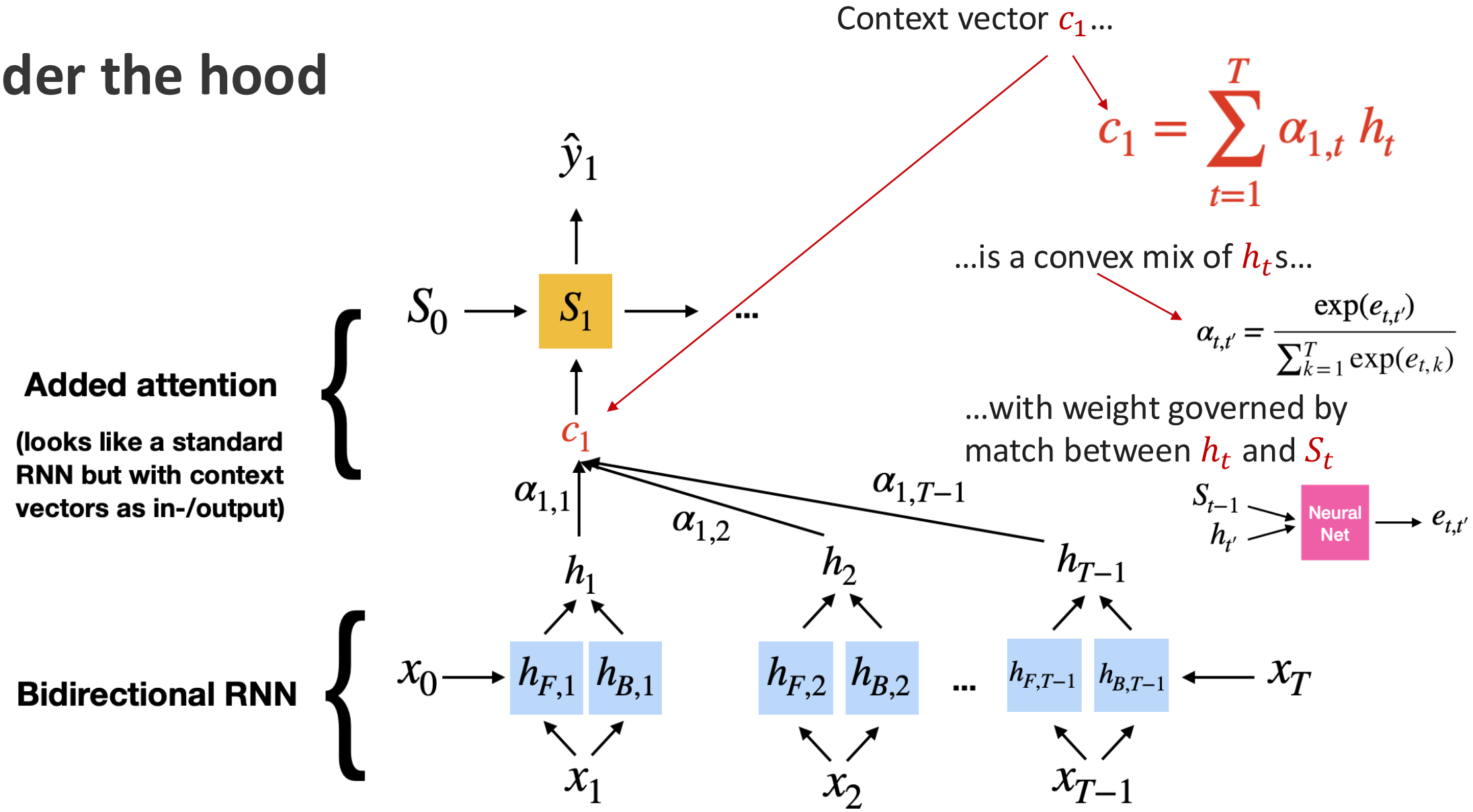


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.



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

Under the hood



Example: Soft Attention for Image Captioning

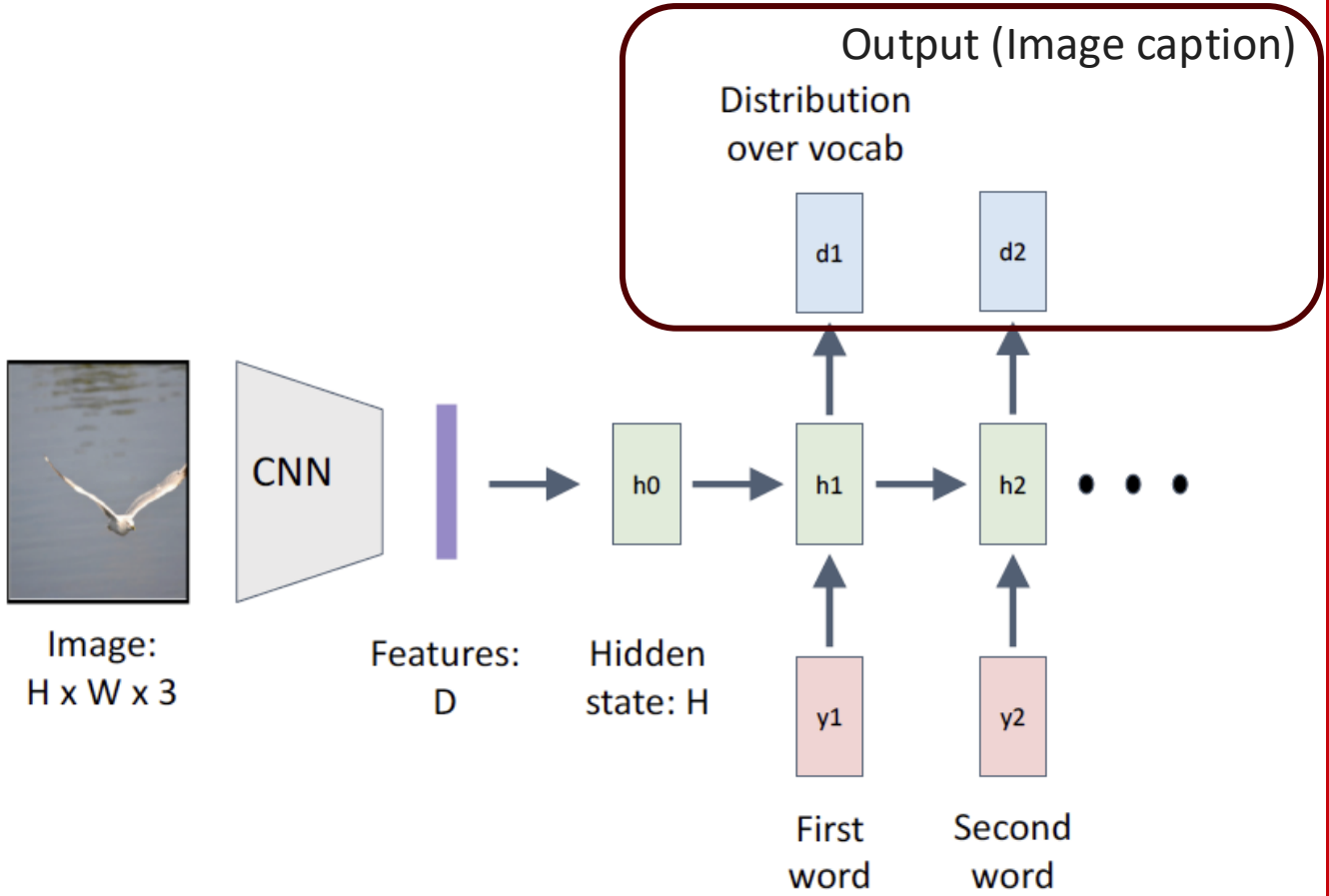
Suppose we want to build a model that takes in images and outputs a text caption.

We have some good tools in our toolkit:

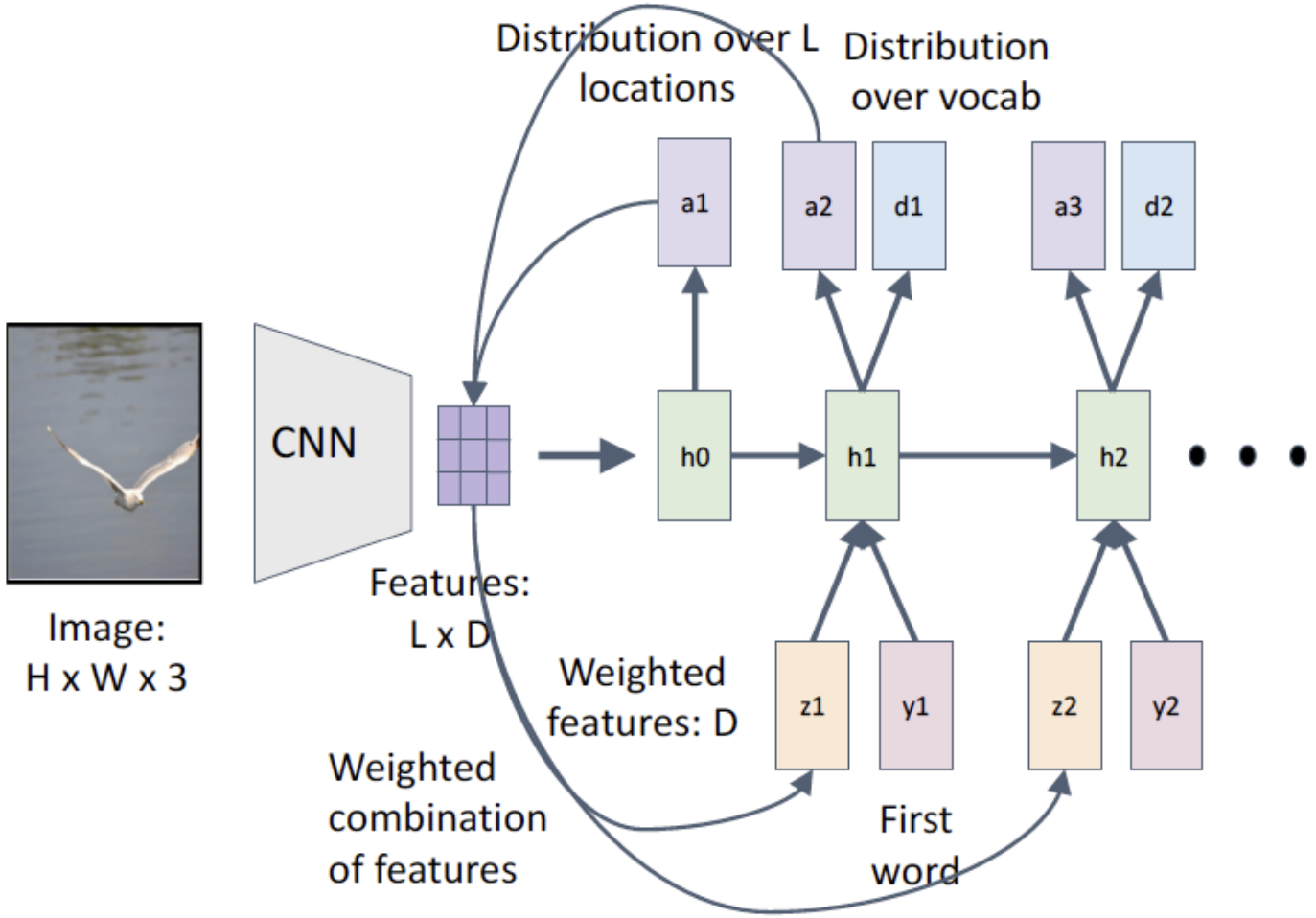
- **CNNs** turn images into features, and we can backprop through them.
- **RNNs** can operate over sequences of outputs to generate variable-length outputs.
- We can turn numbers into words with **embedding tables**.

But! RNNs look at whole image just once, but different parts of the image are important for different parts of the caption.

Can we solve this with attention?

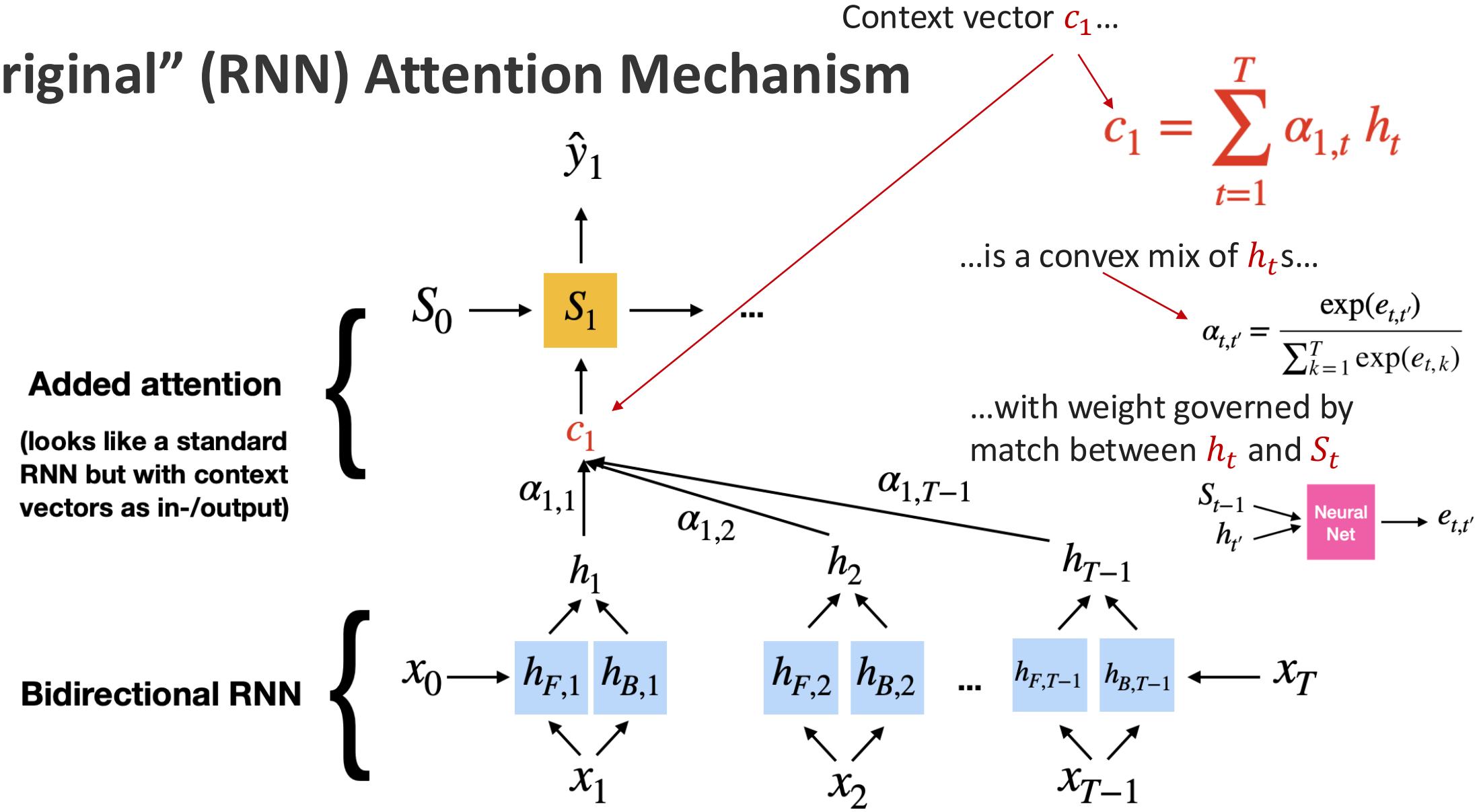


Example: Soft Attention for Image Captioning

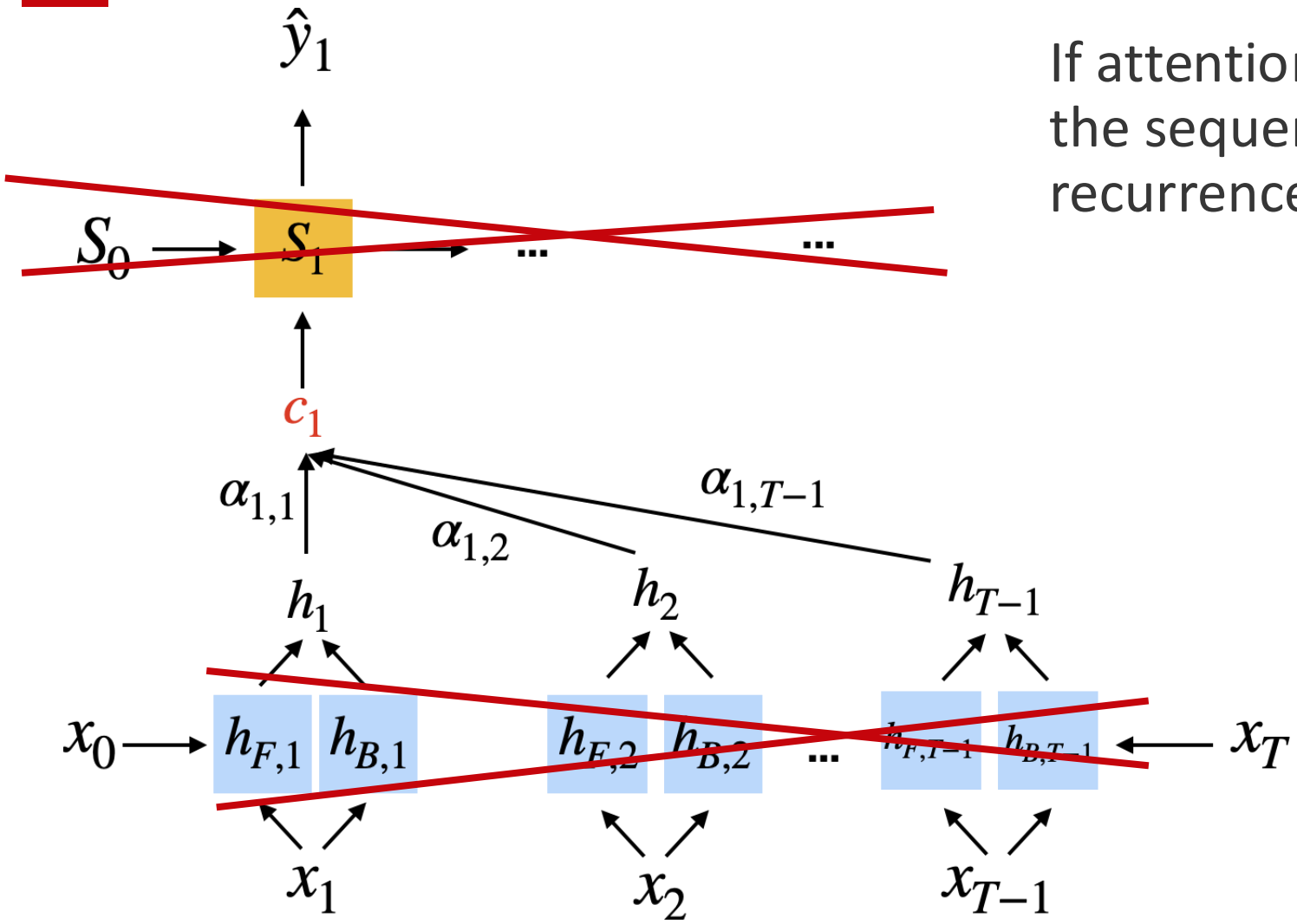


Self-Attention

“Original” (RNN) Attention Mechanism



Can we get rid of the sequential parts?



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

Yes!

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 »

https://arxiv.org/abs/1706.03762

Self-attention (very basic form)

No learnable parameters?

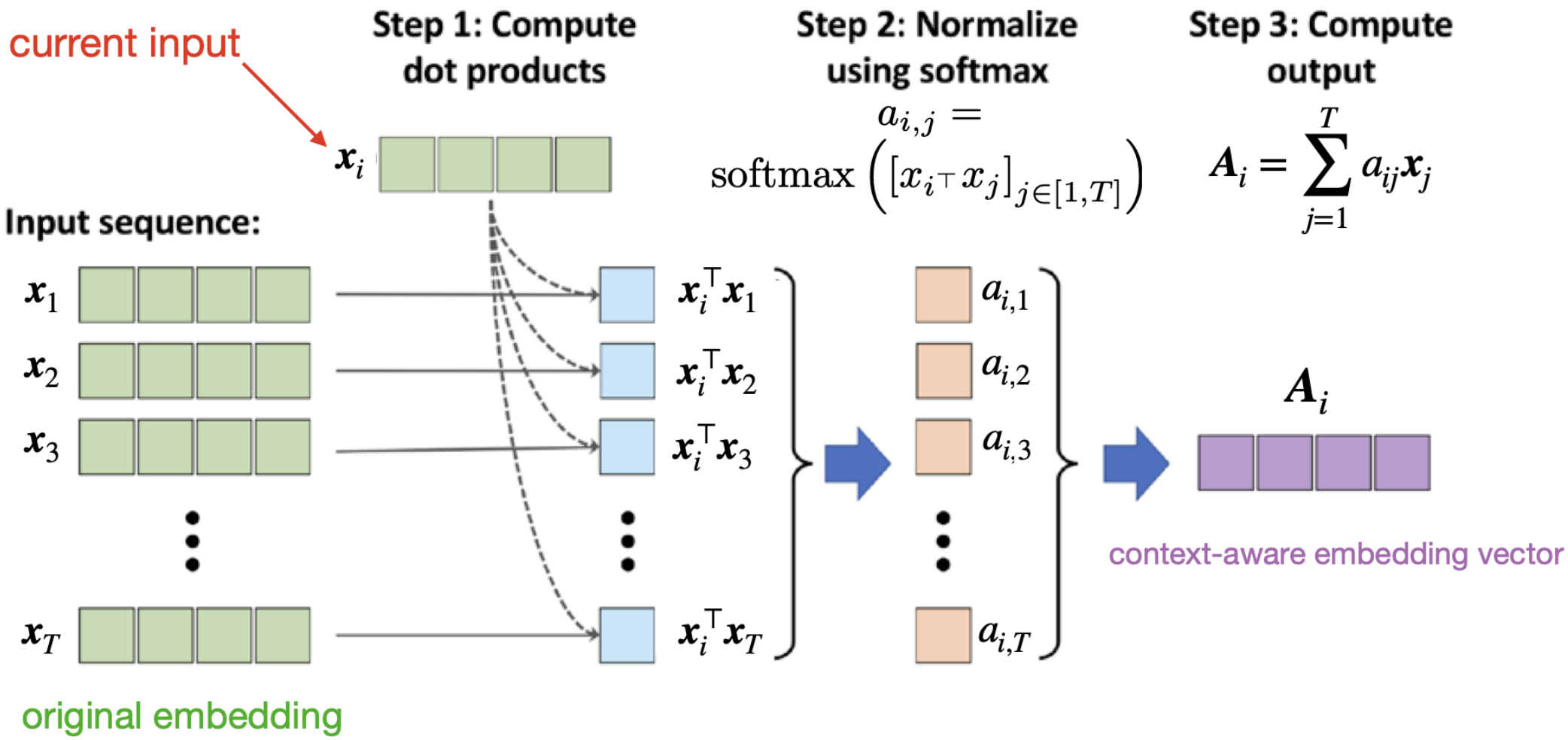


Image source: Raschka & Mirjalili 2019. Python Machine Learning, 3rd edition

Learnable Self-attention

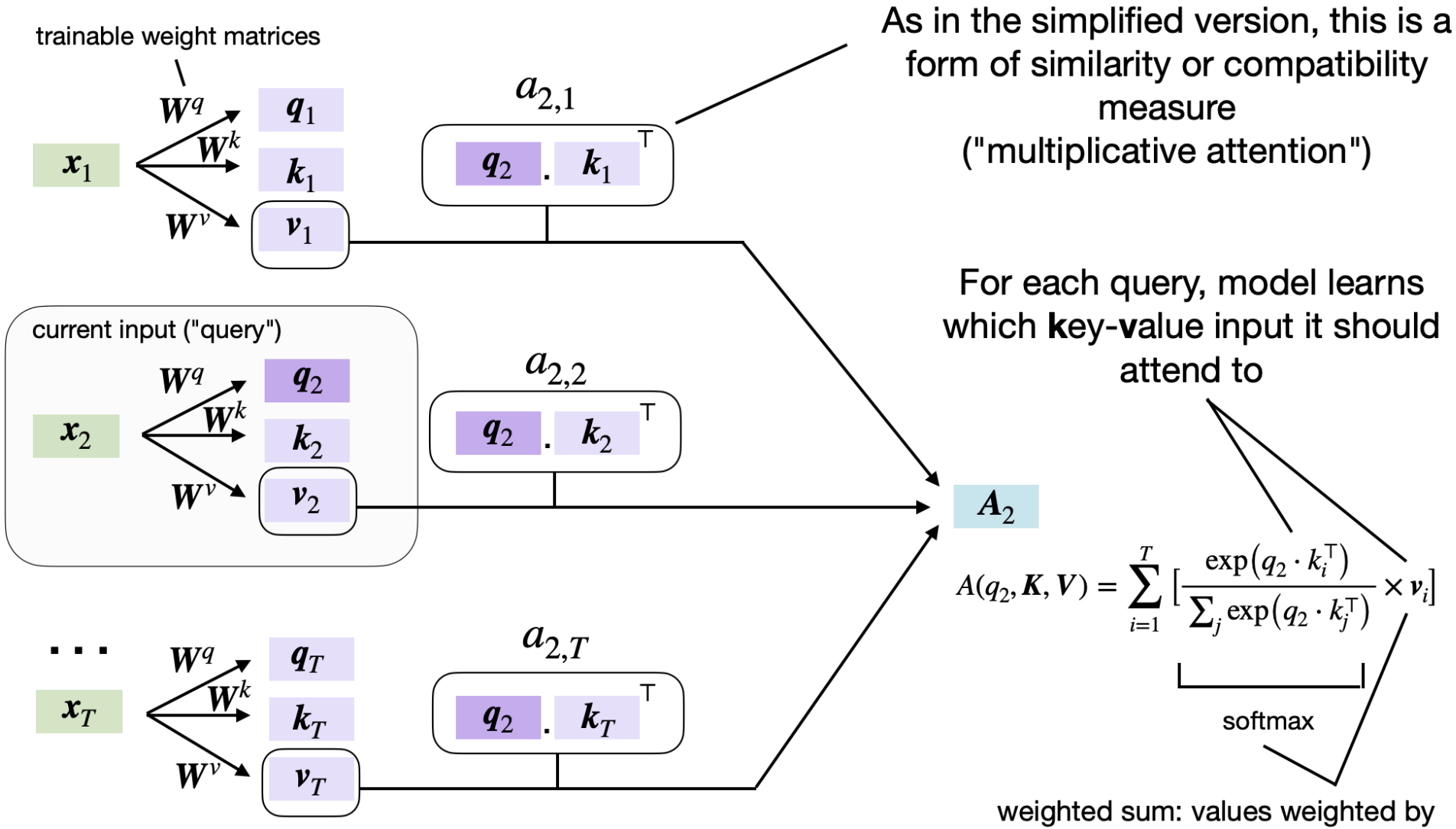
- Previous basic version had no learnable parameters, so not very useful.
- Let's add **3 weight matrices** that will project the input embedding into 3 useful spaces for operations:

$$\text{query} \quad \boldsymbol{q_i} = \boldsymbol{W^q x_i}$$
$$\text{key} \quad \mathrm{k_i} = \boldsymbol{W^k x_i}$$
$$\text{value} \quad \mathrm{v_i} = \boldsymbol{W^v x_i}$$

"Queries select keys, and key unlock values"

Ben Lengerich © University of Wisconsin-Madison 2026

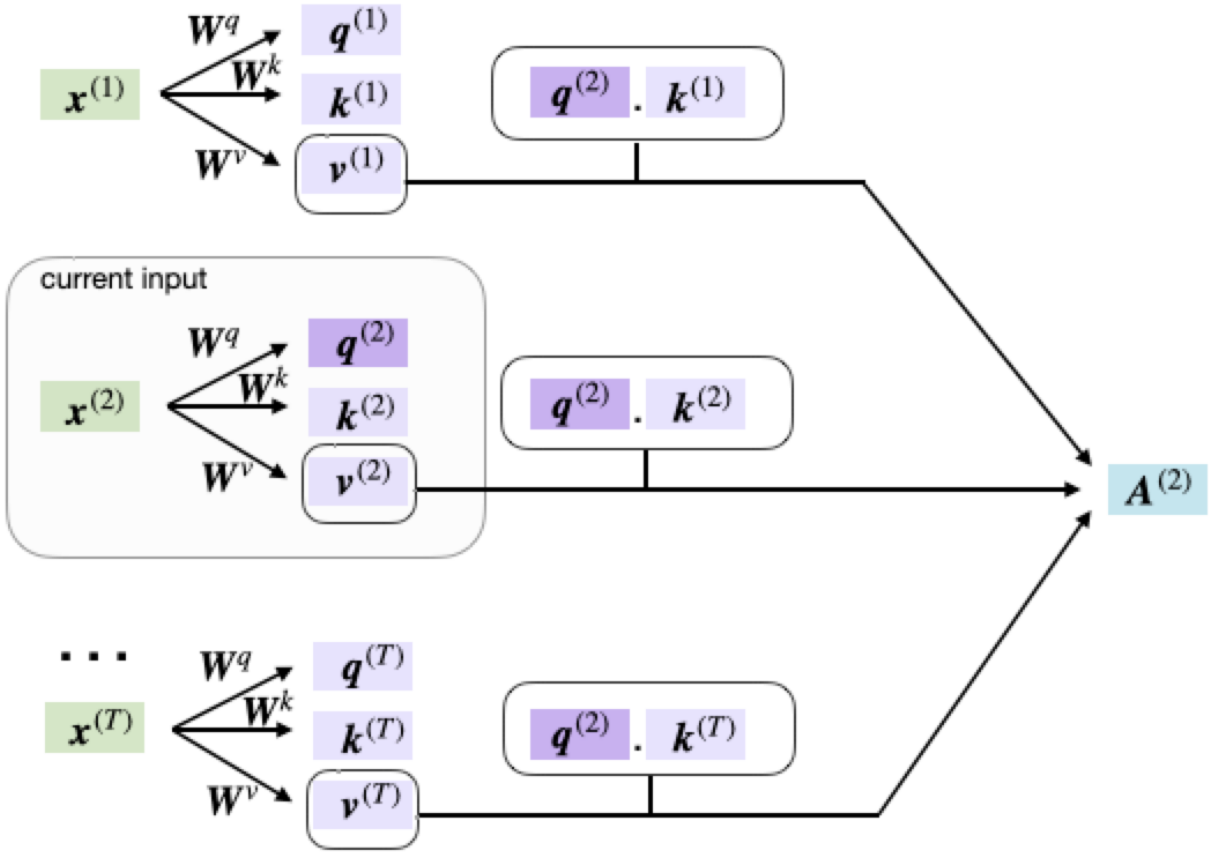
Learnable Self-attention



Ben Lengerich © University of Wisconsin-Madison 2026

Learnable Self-attention

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

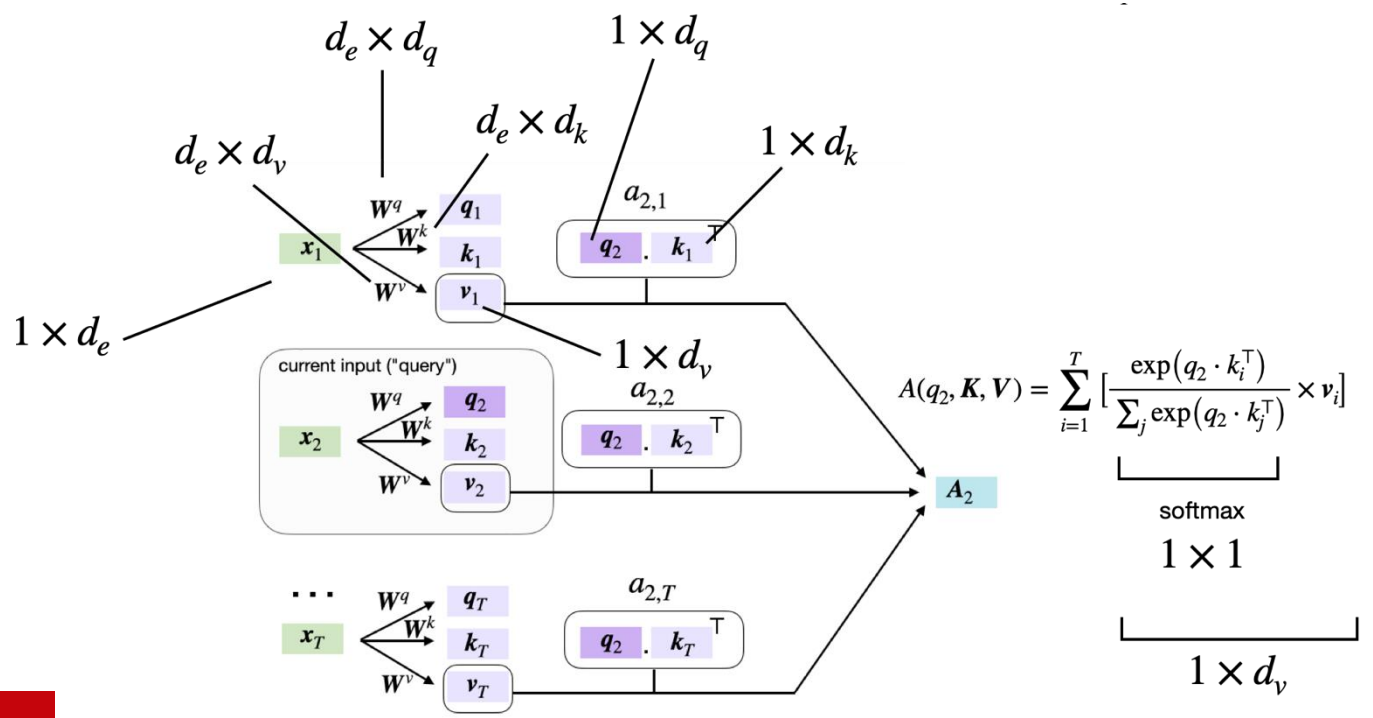


Learnable Self-attention

- Dimensional analysis:
  - Each **x** get embedded into continuous-valued space. Let's call that embedding size d_e.
  - Each **x** embedding gets projected into a query by $\boldsymbol{W^q}$. We get to pick that output size. Let's call that d_q.
  - Each **x** embedding gets projected into a key by $\boldsymbol{W^k}$. We get to pick that output size. Let's call that d_k.
  - Each **x** embedding gets projected into a value by $\boldsymbol{W^v}$. We get to pick that output size. Let's call that d_v.
  - We're going to do dot-products between **keys** and **queries**, so we need to pick $d_k = d_v$.
- In original Transformer paper:
  - $d_e = 512$
  - $d_q = d_k = 64$
  - $d_v = 64$

One more wrinkle of original Transformer paper:

- They chose to use 8 transformer **heads** ("channels")
- 8*64 = 512, so everything ended up the same dim.



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 $\boldsymbol{W^q}$, $\boldsymbol{W^k}$, $\boldsymbol{W^v}$, then concatenate the results
- Allows attending to different parts in the sequence differently

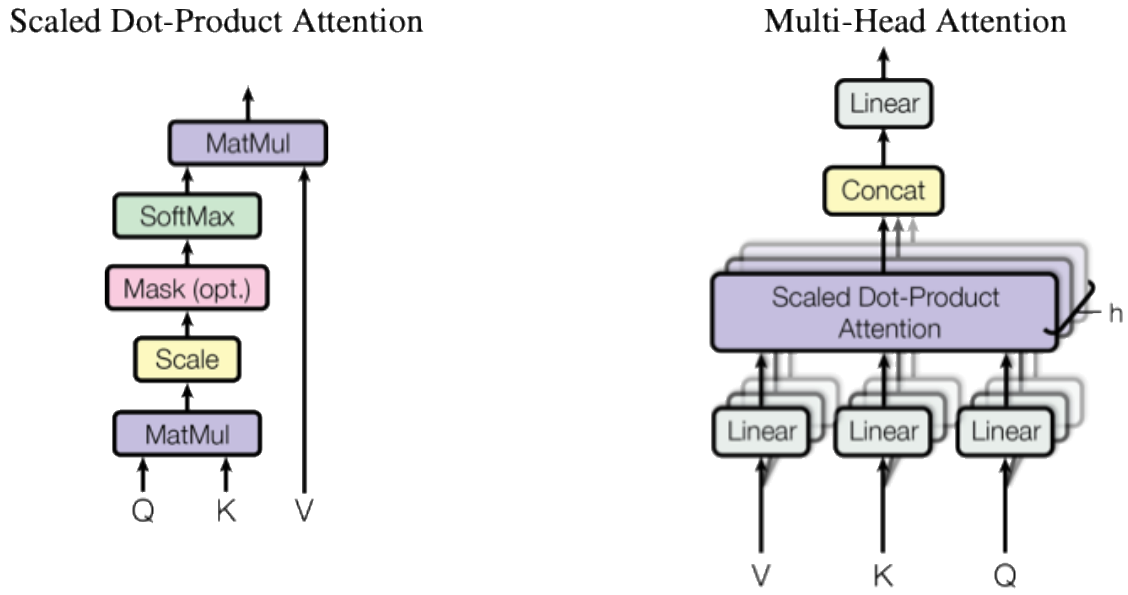
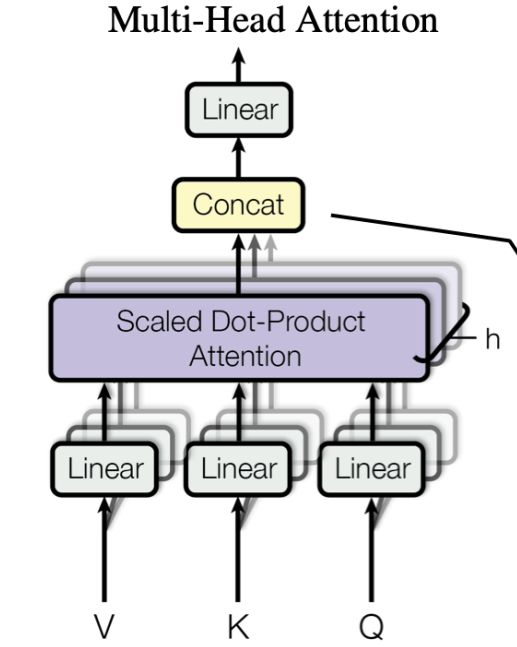


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

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



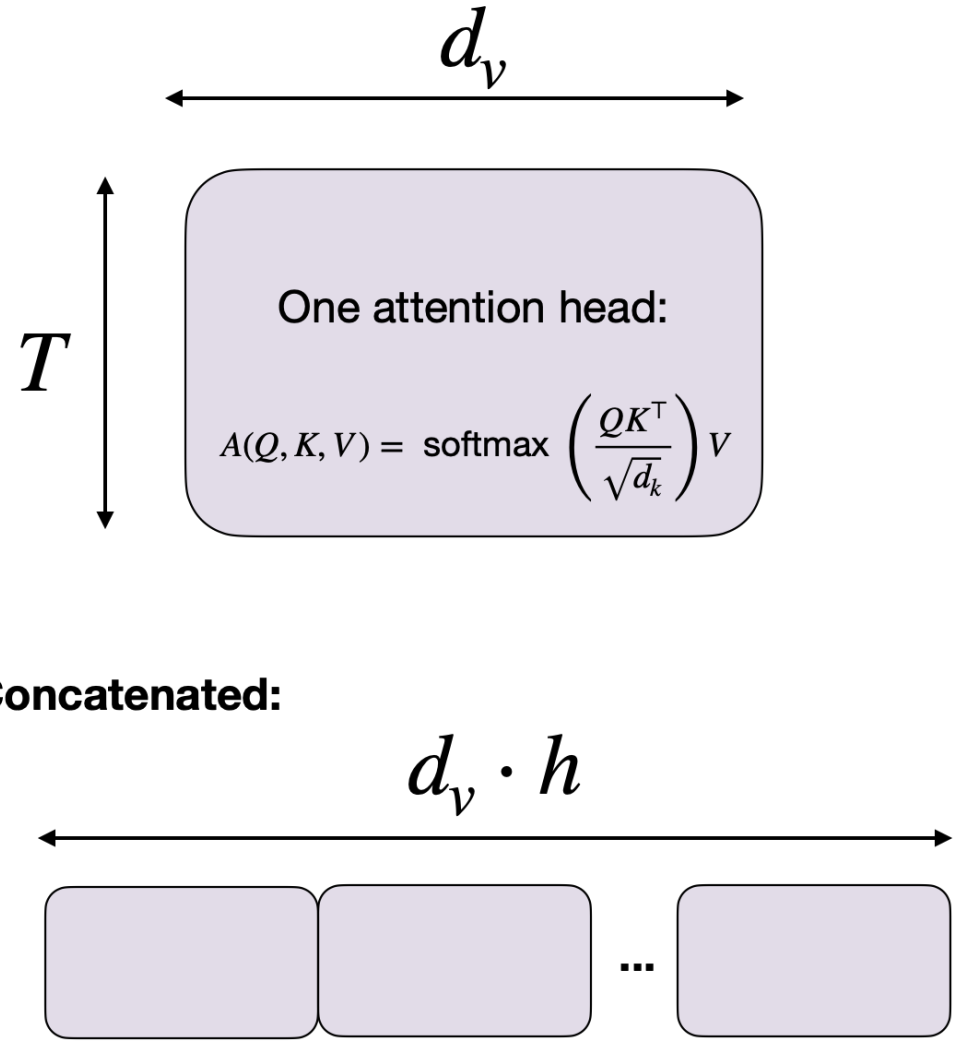
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I., 2017. Attention Is All You Need.

Input sequence dim.
in original transformer:

$T \times d_e = T \times 512$

and

$d_v = 512/h = 64$



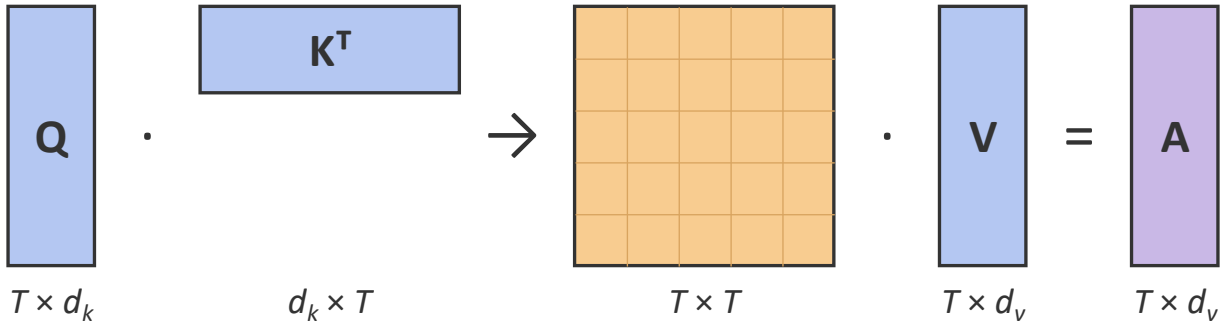
Self-attention in matrix form

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}^k, \quad \mathbf{V} = \mathbf{X}\mathbf{W}^v$$

X: the T input embeddings stacked as rows ($T \times d_e$)

$$\mathbf{A} = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}$$

softmax is applied to each row separately



scores: every token vs. every token

row i of **A** = context-aware embedding of token i

- Same computation as the previous slides, for **all *T* tokens at once**, as a few matrix multiplies (fast on GPUs)
- The $T \times T$ score matrix is why attention costs $O(T^2)$ memory: double the context, 4× the work
- Dividing by $\sqrt{d_k}$ keeps the dot products from growing with dimension, so the softmax doesn't saturate

Ben Lengerich © University of Wisconsin-Madison 2026

At the end of the day…

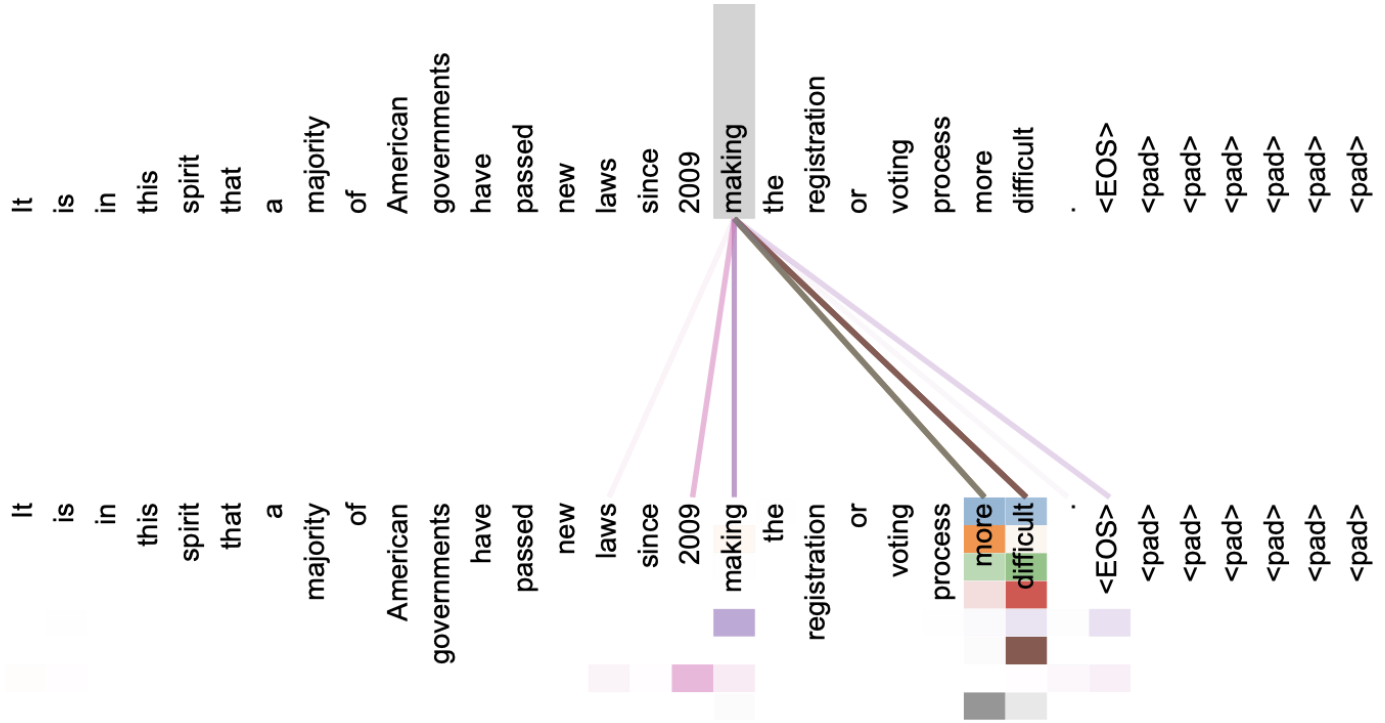


Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb 'making', completing the phrase 'making...more difficult'. Attentions here shown only for the word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

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 »

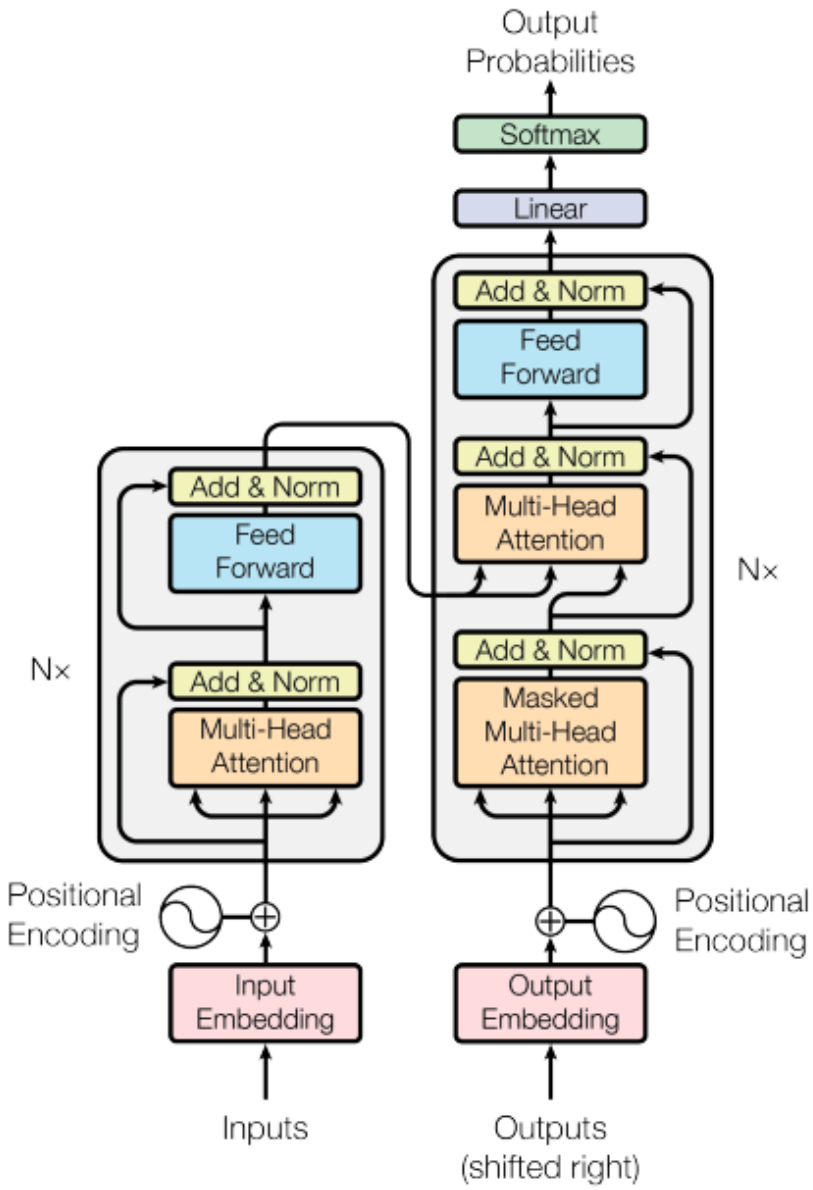
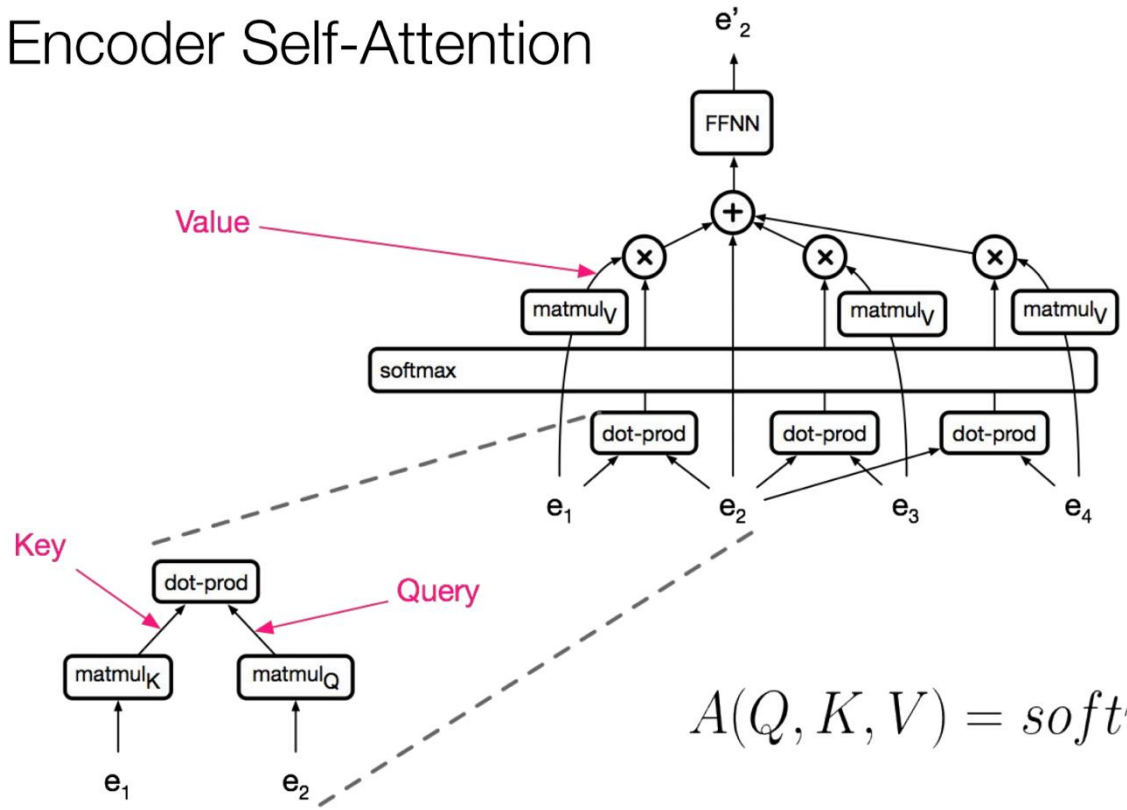


Figure 1: The Transformer - model architecture.

Ben Lengerich © University of Wisconsin-Madison 2026

The ”Transformer”: Encoder



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

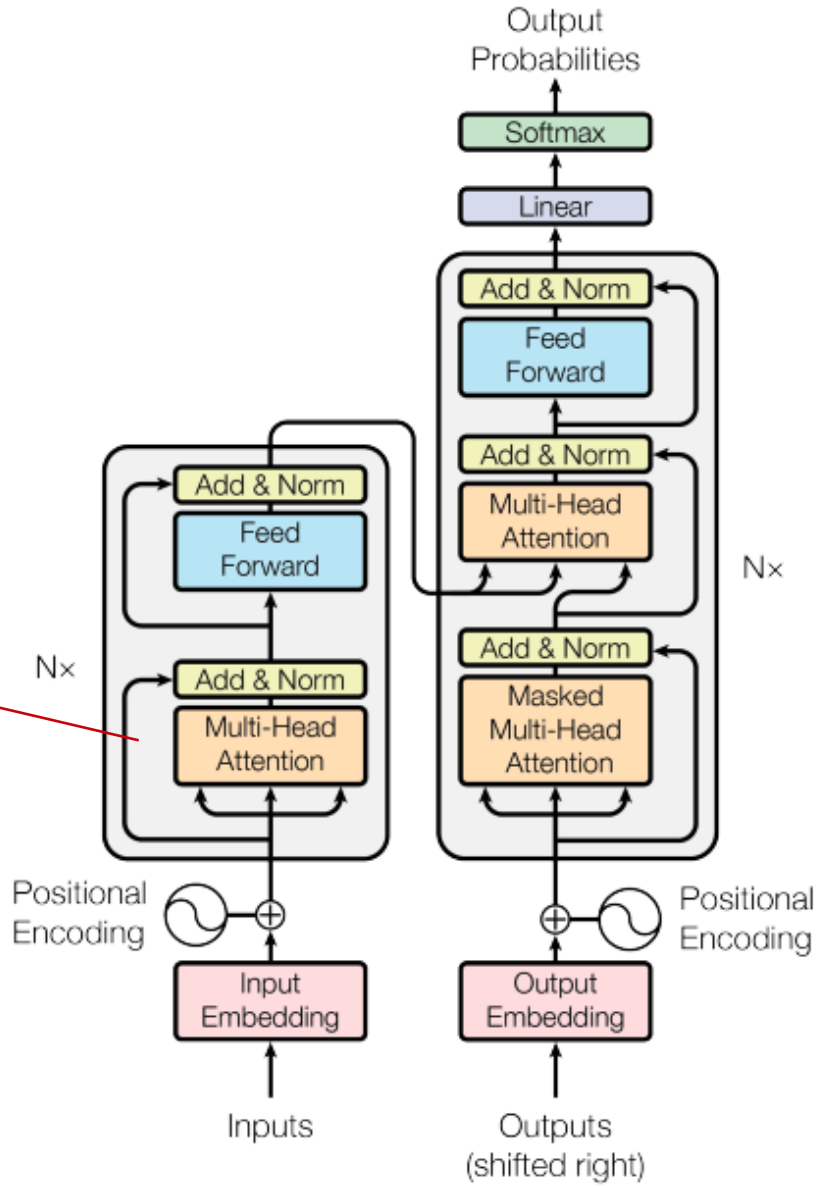
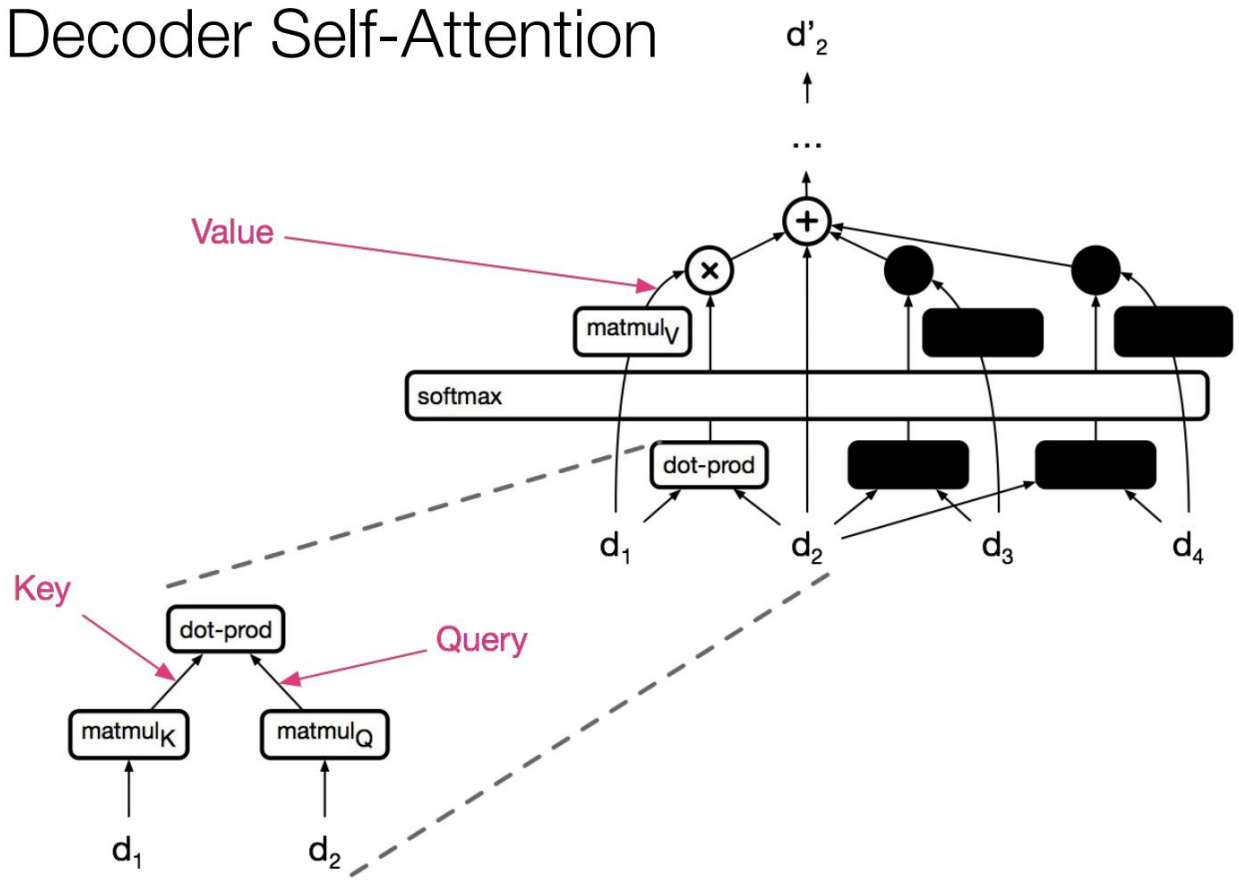


Figure 1: The Transformer - model architecture.

Vaswani [“Self-Attention for Generative Models”]

The ”Transformer”: Decoder



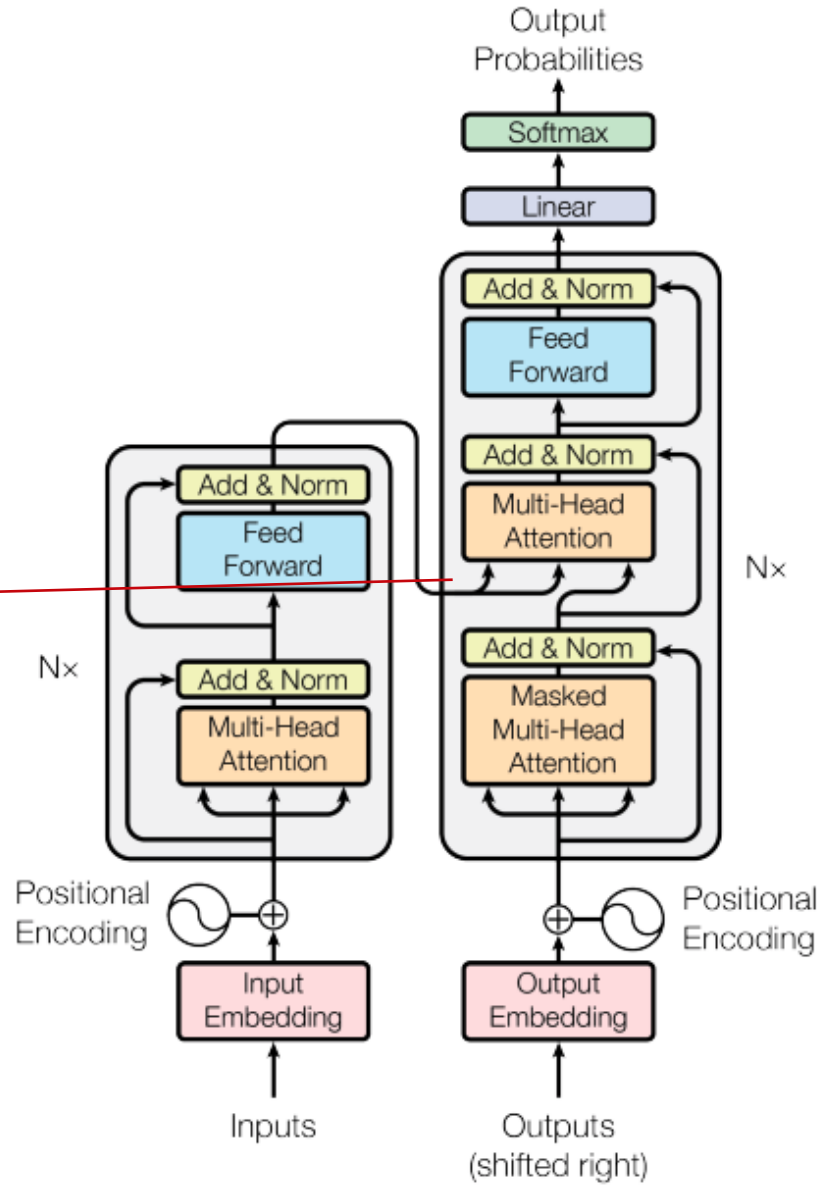
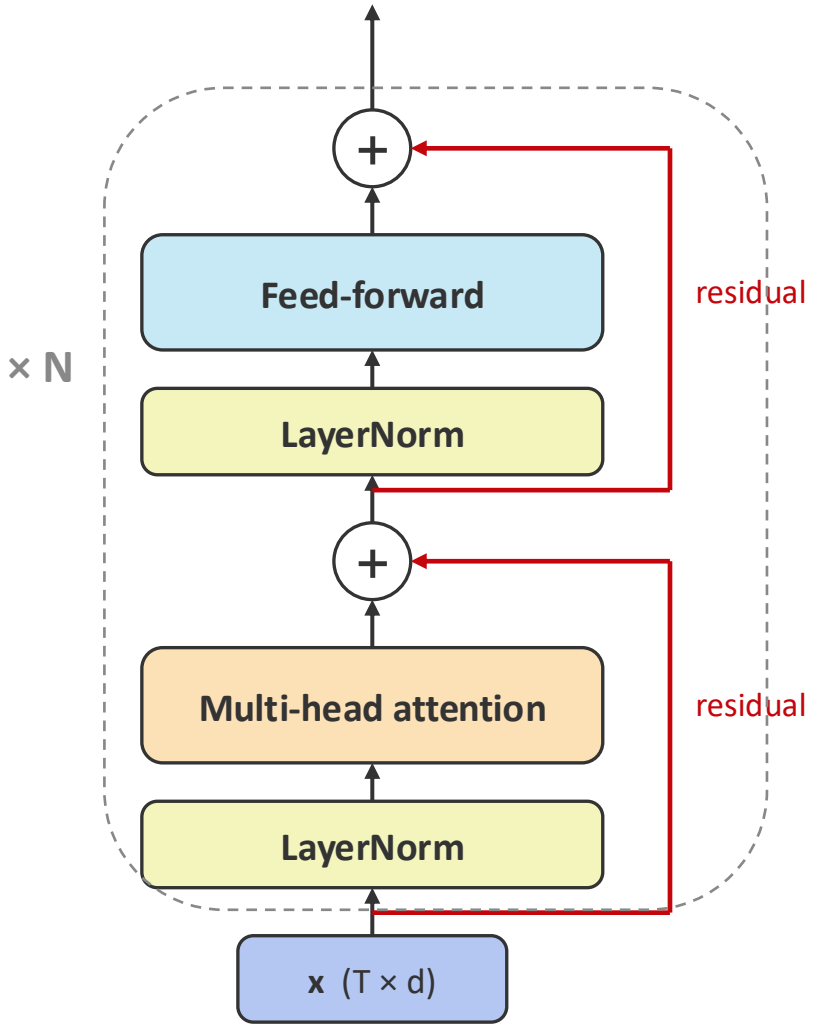


Figure 1: The Transformer - model architecture.

Vaswani [“Self-Attention for Generative Models”]

Inside a transformer block



$$\mathbf{h} = \mathbf{x} + \mathrm{MultiHead}(\mathrm{LN}(\mathbf{x}))$$

$$\mathrm{out} = \mathbf{h} + \mathrm{FFN}(\mathrm{LN}(\mathbf{h}))$$

$$\mathrm{FFN}(\mathbf{h}) = \mathbf{W}_2\,\phi(\mathbf{W}_1\mathbf{h} + \mathbf{b}_1) + \mathbf{b}_2$$

ϕ = ReLU in the original; GELU (GPT, BERT) or SwiGLU (Llama and most open models) later

- Attention mixes information **across** tokens; the feed-forward net processes **each token on its own** (same MLP at every position, hidden width 4×: 2048 for d = 512)
- **Residual connections** ("Add"): a direct path for gradients through many layers, like the LSTM's cell state
- **LayerNorm** ("Norm") normalizes each token's vector. The original paper normalized *after* each add; most models today normalize *before* each sublayer ("pre-norm," shown here), which trains more stably
- Stack N identical blocks (N = 6 in the original transformer)

Ben Lengerich © University of Wisconsin-Madison 2026

Causal masking (decoder self-attention)



$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}} + \mathbf{M}\right)\mathbf{V}$$

$$M_{ij} = 0 \;\text{ if } j \le i, \qquad M_{ij} = -\infty \;\text{ if } j > i$$

- Token *i* can only attend to tokens 1…*i*: exp(−∞) = 0, so **future tokens get zero weight**
- One forward pass trains next-token prediction at **all *T* positions in parallel**, with no peeking at the answers. This is the training advantage over RNNs.
- Generation still goes one token at a time, like an RNN
- This masked decoder is the building block of **GPT** (next lecture)

Ben Lengerich © University of Wisconsin-Madison 2026

Transformer Tricks (4?)

- **Self Attention:** Each layer combines words with others
- **Multi-headed Attention:** 8 attention heads learned independently
- **Normalized Dot-product Attention:** Scale by $1/\sqrt{d_k}$ so softmax doesn't saturate when d_k is large
- **Positional Encodings:** Make sure that even if we don't have RNN, can still distinguish positions

Ben Lengerich © University of Wisconsin-Madison 2026

Transformer Tricks – Positional Encodings

- Scaled dot-product and fully-connected layer are permutation equivariant (reordering the inputs just reorders the outputs)
- 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

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

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

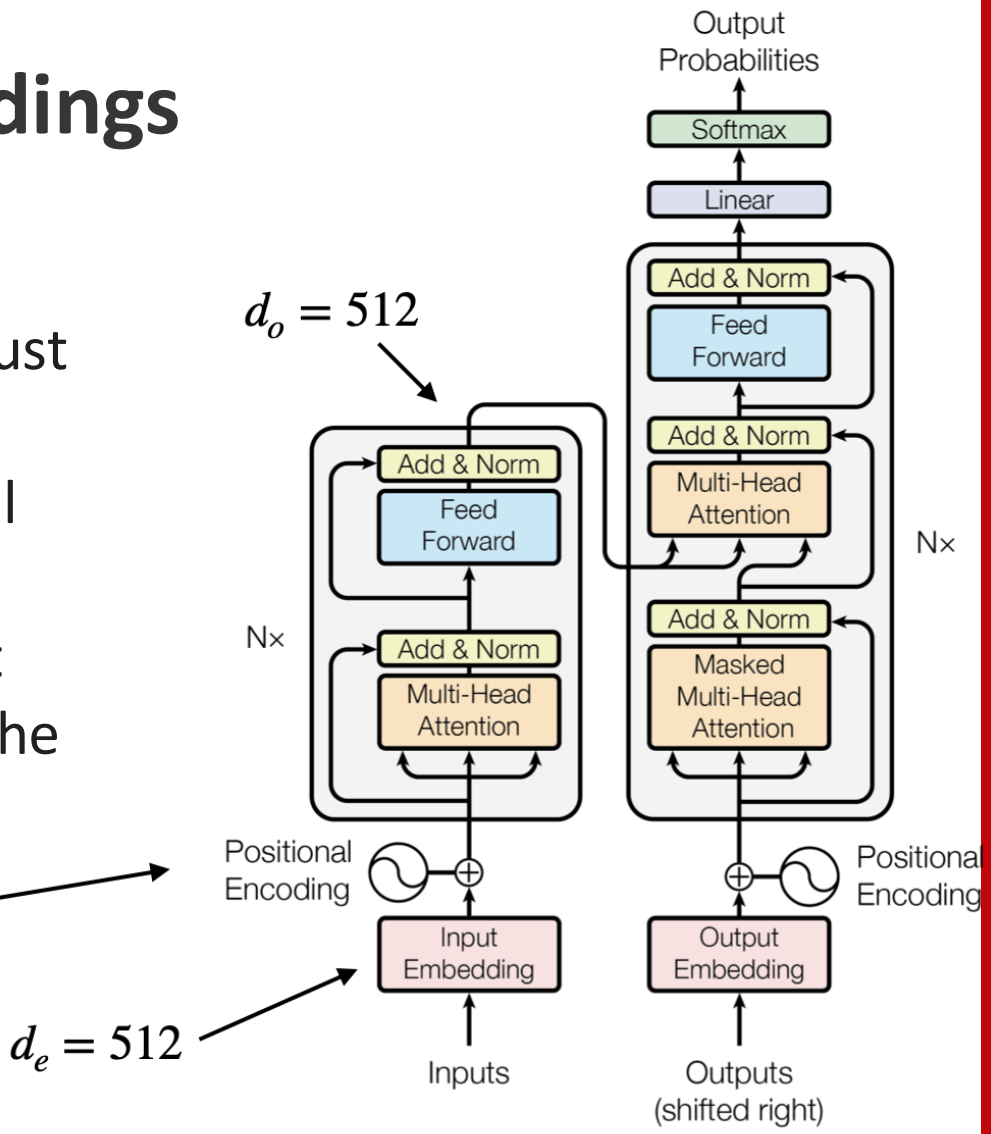


Figure 1: The Transformer - model architecture.

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

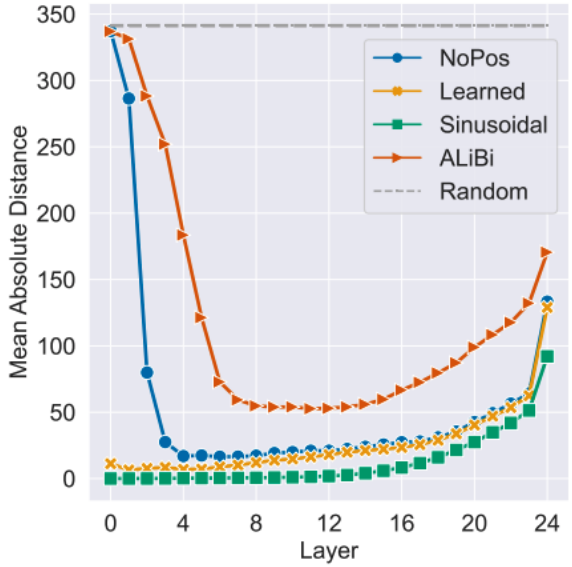
Do we even need positional encodings?

Transformer Language Models without Positional Encodings Still Learn Positional Information

Adi Haviv[τ] **Ori Ram**[τ] **Ofir Press**[ω] **Peter Izsak**[ι] **Omer Levy**[τμ]

[τ]Tel Aviv University [ω]University of Washington [ι]Intel Labs [μ]Meta AI

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



Caveat: this holds for **causal** (decoder-only) models. The mask itself leaks position: token *i* attends to exactly *i* tokens.

Without a mask (encoders), dropping positional encodings makes the model permutation-equivariant again.

Positional encodings today

- **Learned absolute embeddings:** one trained vector per position, added to the token embedding (BERT; GPT-1, -2, -3). Simple, but nothing to use beyond the longest training length.
- **Rotary embeddings (RoPE; Su et al. 2021):** add nothing to the input. Instead, rotate each query and key by an angle proportional to its position:

$$q_m \to R_{m\theta}\, q_m, \quad k_n \to R_{n\theta}\, k_n \qquad \Longrightarrow \qquad (R_{m\theta}\, q_m)^\top (R_{n\theta}\, k_n) = q_m^\top R_{(n-m)\theta}\, k_n$$

The score depends only on the **offset *n* – *m***: relative position for free. Used by Llama, Qwen, Mistral and most open LLMs today.

- **ALiBi (Press et al. 2022):** no position vectors at all. Subtract a penalty proportional to distance from each attention score.

Three families of transformers

Encoder-only

Examples
BERT (2018)

Attention
bidirectional: every token sees the whole input

Trained to
fill in masked-out words

Good at
classification, search, embeddings

Encoder–decoder

Examples
original Transformer (2017), T5 (2019)

Attention
bidirectional encoder + causal decoder that also attends to the encoder

Trained to
map an input sequence to an output sequence

Good at
translation, summarization

Decoder-only

Examples
GPT series (2018–), Llama, Qwen, DeepSeek

Attention
causal: each token sees only the past

Trained to
predict the next token

Good at
open-ended generation and chat; the dominant design today

Decoder-only won out for generation. Next time: GPT.

Next Time: GPT

Generative Pre-Trained Transformer

Questions?

