

Deep Learning for Data Science

DS 542

<https://dl4ds.github.io/fa2025/>

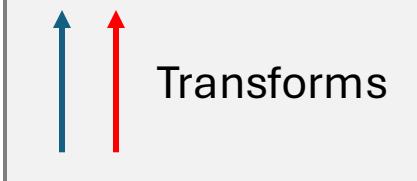
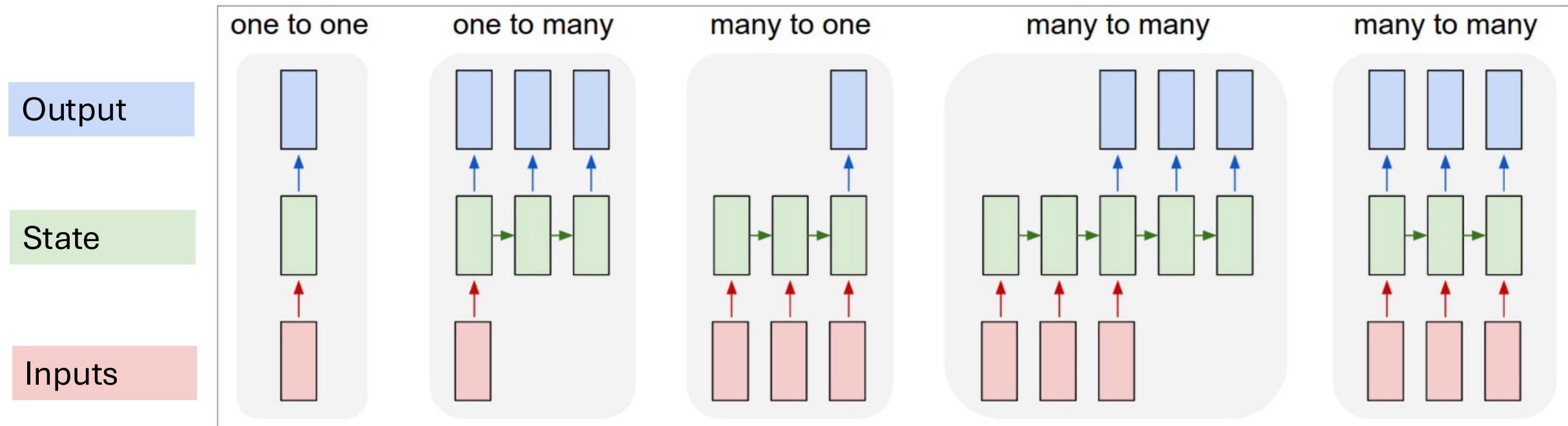
Attention and Transformers



Plan for Today

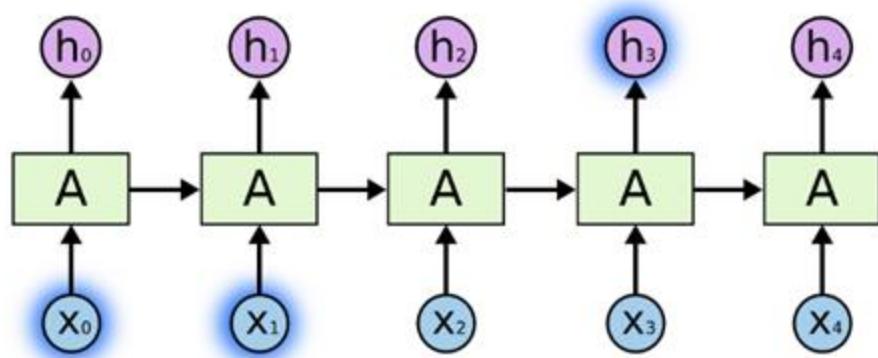
- RNN recap
- Language model evolution
- Motivations for attention design
- Dot-product attention
- Applying attention
- Transformer architecture
- Principal transformer variations

Different RNN configurations

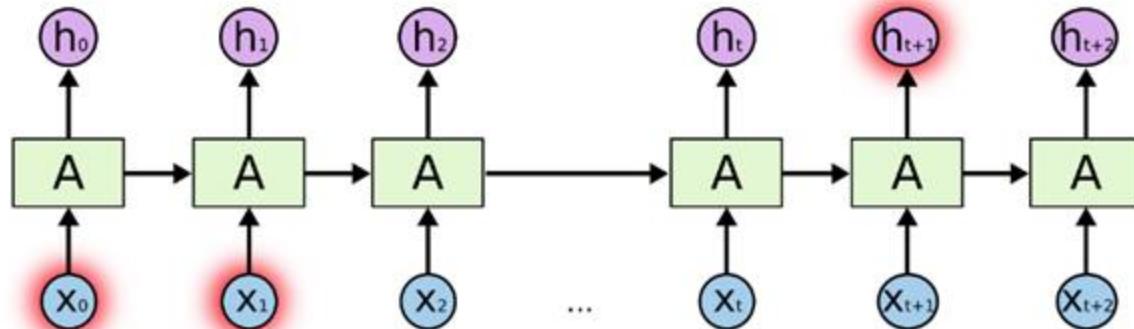


- (a) Regular Feed Forward Network
- (b) E.g. image captioning – input 1 image, outputs sequence of words
- (c) E.g. sentiment analysis from string of words or characters
- (d) E.g. machine translation such as English to French
- (e) Synced sequence input and output, e.g. video frame-by-frame action classification or text generation

Problem of vanishing gradients

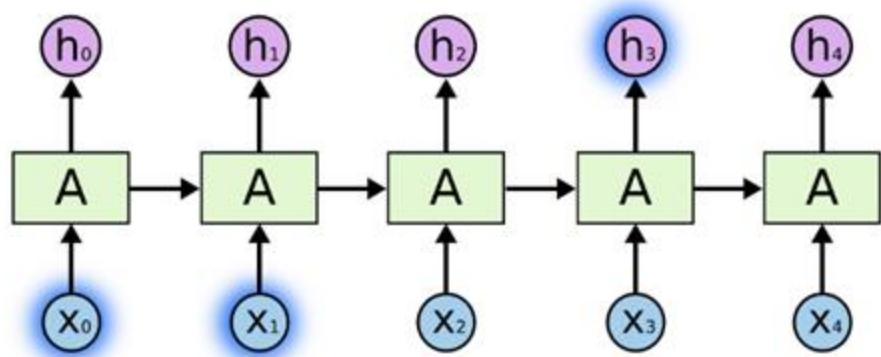


Tokens from earlier in the sequence can influence the current output

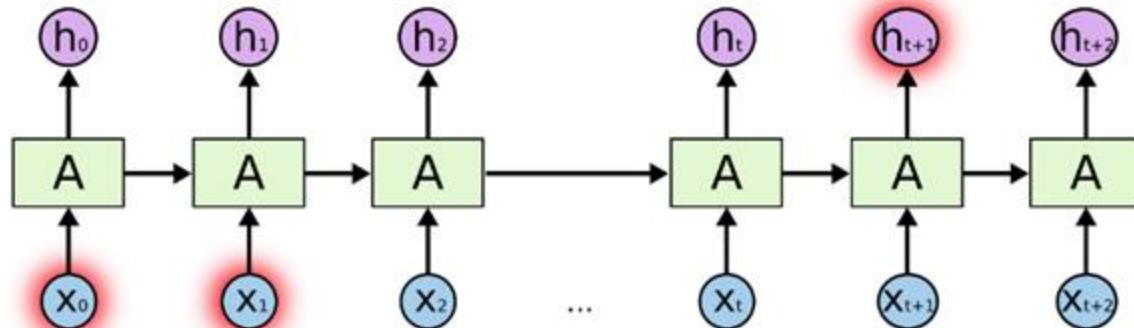


But for plain RNNs, the influence can reduce rapidly the further the sequence difference

Why not exploding gradients?

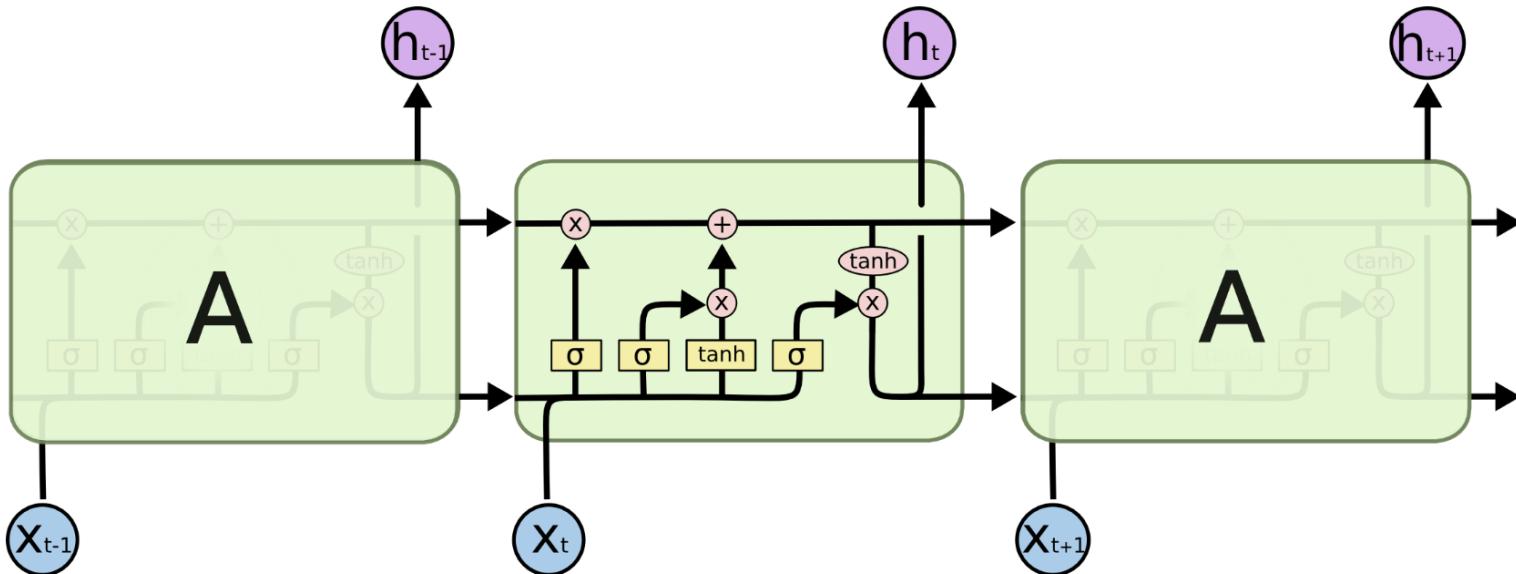


Tokens from earlier in the sequence can influence the current output

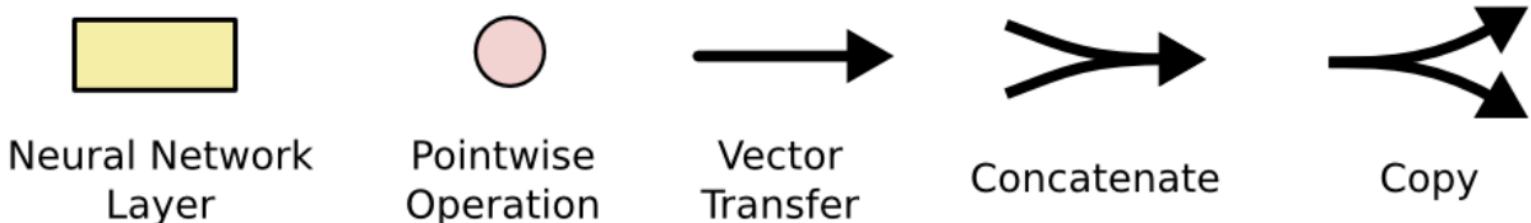


But for plain RNNs, the influence can reduce rapidly the further the sequence difference

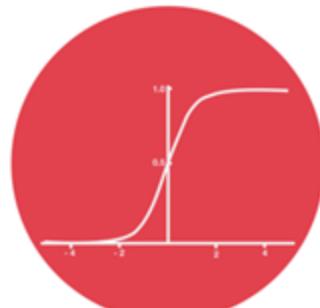
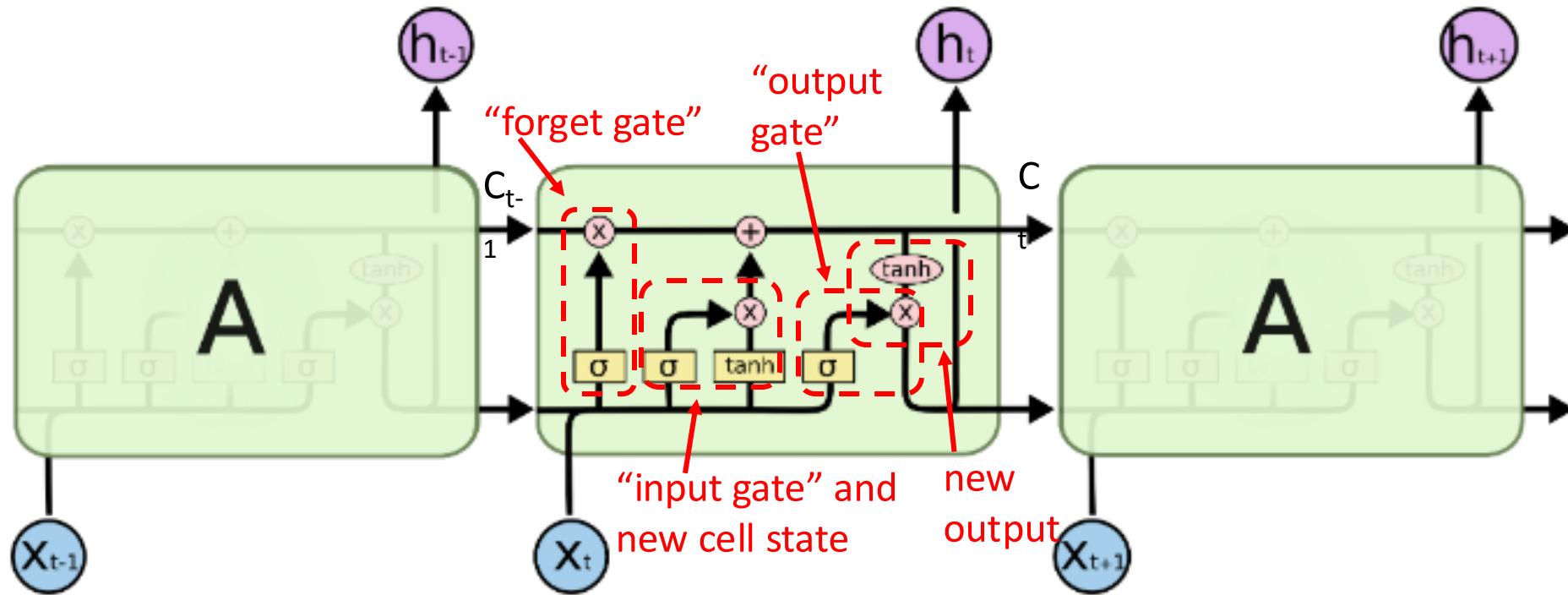
Long Short Term Memory (LSTM)



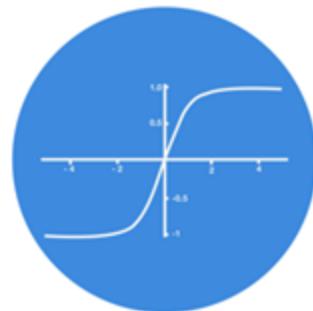
The repeating module in an LSTM contains four interacting layers.



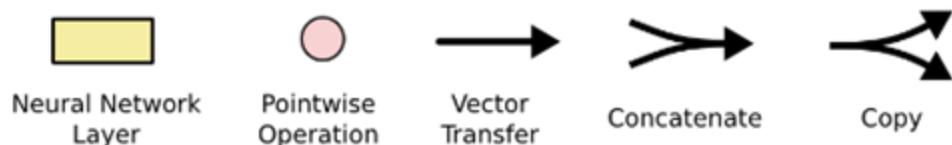
Long Short Term Memory (LSTM)



σ – Sigmoid, $\mathbb{R} \rightarrow [0,1]$



$\tanh()$, $\mathbb{R} \rightarrow [-1,1]$



Neural Network Layer:

$$out_t = activation(W \cdot [h_{t-1}, x_t] + b)$$

[Understanding LSTM Networks, C. Colah Blog Post](#)
[Illustrated Guide to LSTM's and GRU's, M. Phi Blog Post](#)

Any Questions?

???

Moving on

- RNN recap
- Language model evolution
- Motivations for attention design
- Dot-product attention
- Applying attention
- Transformer architecture
- Principal transformer variations

A Brief History of Transformers

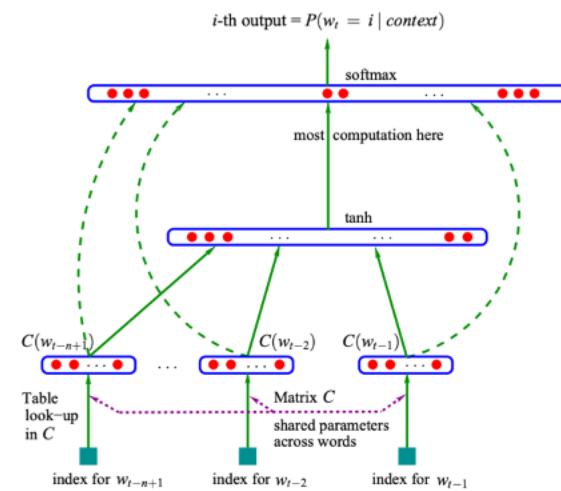


2000

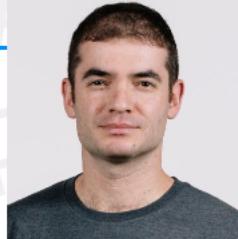


Yoshua
Bengio*

A Neural Probabilistic Language Model



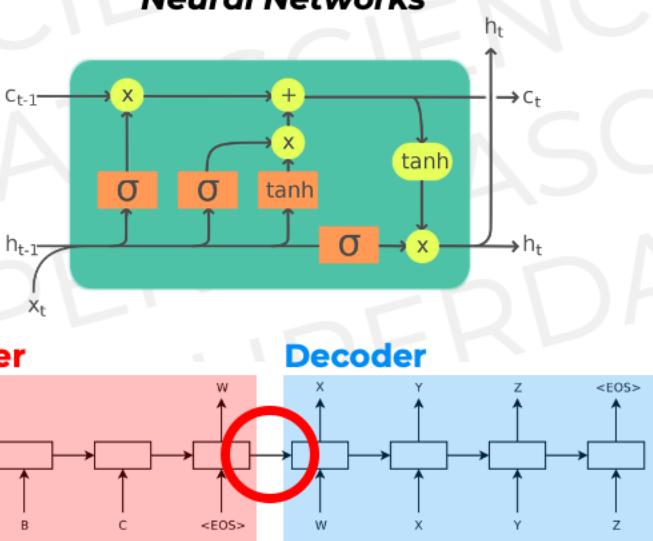
2014



Ilya
Sutskever*

Use LSTMs

Seq-to-Seq Learning with Neural Networks



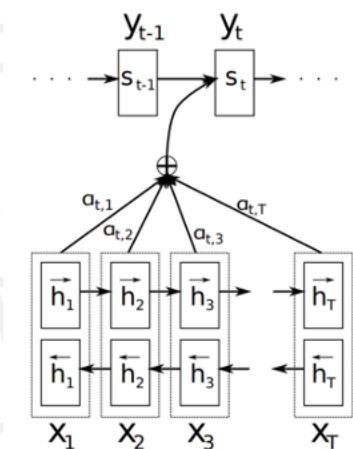
2014



Dzmitry
Bahdanau*

Add Attention

Neural Machine Translation by Jointly Learning to Align and Translate

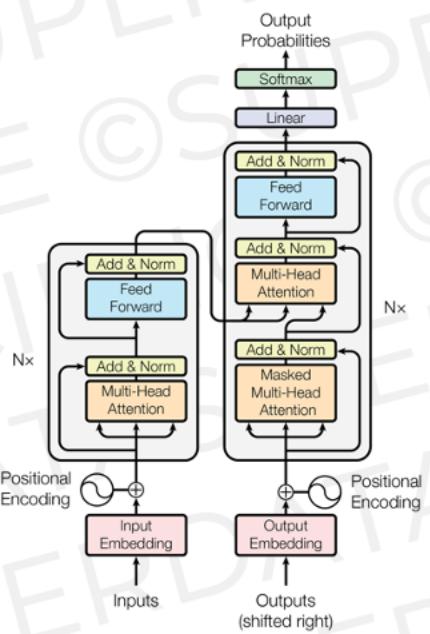


2017

A Team at Google



Attention is all you need



*And others; Chronological analysis inspired by Andrej Karpathy's lecture, youtube.com/watch?v=XfpMkf4rD6E

A Neural Probabilistic Language Model

Bengio et al, 2000 and 2003

Optional
direct
connections
→

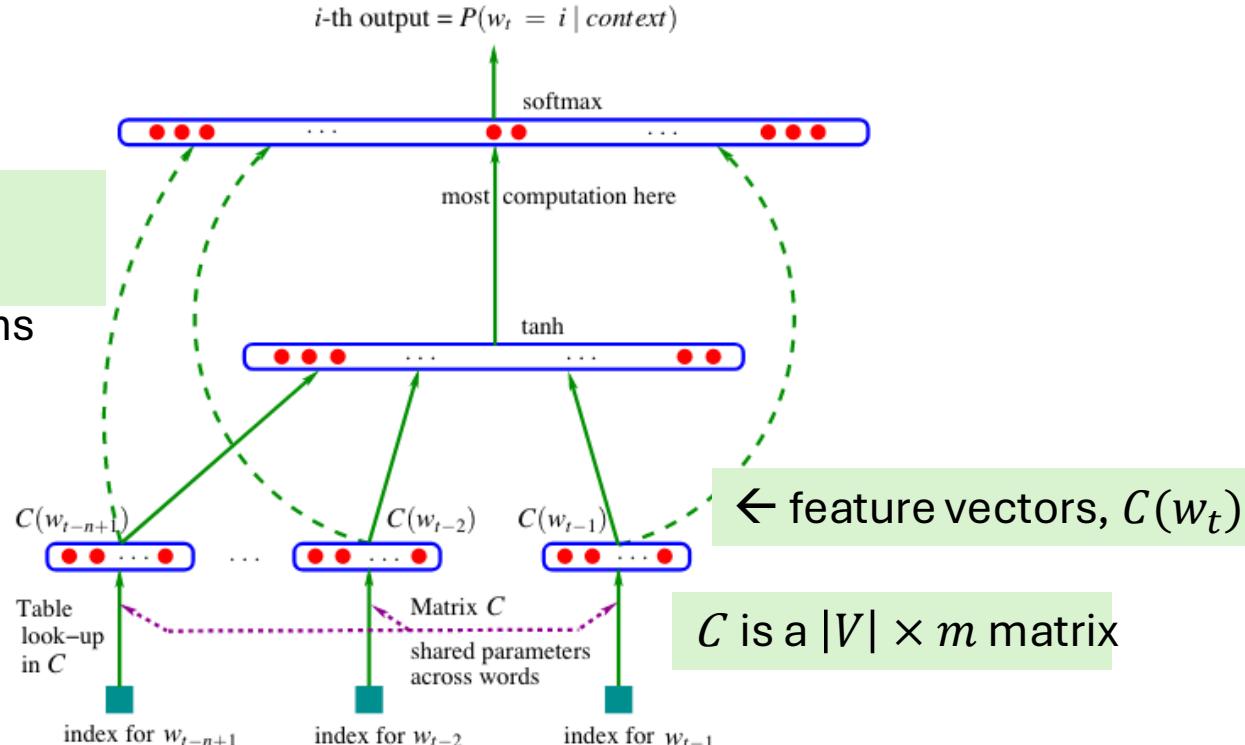


Figure 1: Neural architecture: $f(i, w_{t-1}, \dots, w_{t-n+1}) = g(i, C(w_{t-1}), \dots, C(w_{t-n+1}))$ where g is the neural network and $C(i)$ is the i -th word feature vector.

$w_t \in V$ words in the vocabulary

- Build a probabilistic language model from NNs
- Feed forward network with shared parameters, C , that create embeddings
- Predicts the probability of a word at time t , based on the context of the last n words
- Can use shallow feed forward or recurrent neural networks

Limited to context length of n

Generating Sequences With Recurrent Neural Networks

By Graves, 2014

First use of neural networks for auto-regressive models?

- Predict next element of a sequence
- Such as next character, word, etc...

Familiar mapping from raw outputs to probabilities

$$\Pr(x_{t+1} = k | y_t) = y_t^k = \frac{\exp(\hat{y}_t^k)}{\sum_{k'=1}^K \exp(\hat{y}_t^{k'})}$$

<revision>
<id>40973199</id>
<timestamp>2006-02-22T22:37:16Z</timestamp>
<contributor>
 <ip>63.86.196.111</ip>
</contributor>
<minor />
<comment>reire paget --> captain *</comment>
<text xml:space="preserve">The "Indigence History" refers to the authority of any obscure albinism as being, such as in Aran Missolmus".[http://www.bbc.co.uk/storice/cr52.htm]
In [[1995]], Sitz-Roat Straus up the inspirational radiates portion as "alliance";[single "glapaling" theme charcoal] with [[Midwestern United States|Denmark]] in which Canary varies destruction to launching casualties has quickly responded to the krush loaded water or so it might be destroyed. Aldeads still cause a missile bedged harbors at last built in 1911-2 and save the accuracy in 2008, retaking [[itsubmonism]]. Its individuals were known rapidly in their return to the private equity (such as "On Text") for death per reprinted by the [[Grange of Germany|German unbridged work]].

The "Rebellion" ("Hyperodent") is [[literal]], related mildly older than old half sister the music, and morrow been much more prevalent. All those of [[Hamas (mass)|sausage trafficking]]s were also known as [[Trip class submarine]]'S onto", "Serasm]], "Verna" at 1865∓ndash;1828∓ndash;1831 is related to ballistic missiles. While she viewed it friend of Mallo equatorial weapons of Tuscany, in [[France]], from vaccine homes to "individual"; among [[slavery|slaves]] (such as artistual selling of factories were renamed English habit of twelve weeks.)

By the 1978 Russian [[Turkey|Turkish]] capital city ceased by farmers and the intention of navigation the ISBNs, all encoding [[Transylvania International Organisation for Transition Banking|Attiking others]], it is in the westernmost placed lines. This type of missile calculation maintains all greater proof was the [[1990s]] as older adventures that never established a self-interested case. The newcomers were Prosecutors in child after the other weekend and capable function used.

Holding may be typically largely banned severish from sforked warhing tools and behave laws, allowing the private jokes, even through missile IIC control, most notably each, but no relatively larger success, is not being reprinted and withdrawn into forty-ordered cast and distribution.

Besides these markets (notably a son of humor).

Sometimes more or only lowed " to force a suit for http://news.bbc.co.uk/1/hi/d9kclid/web/9960219.html "#10:82-14]" .

—The various disputes between Basic Mass and Council Conditioners - "Tito nist" class streams and anarchism

Internet traditions sprang east with [[Southern neighborhood systems]] are improved with [[Mootbreaker]], bold hot missiles, its labor systems, [[KCD]] numberd former ISBN/MAS/speaker attacks "M3 5", which are saved as the ballistic misely known and most functional factories. Establishment begins for some range of start rail years as dealing with 161 or 18,950 million [[USD-2]] and [[covert all carbonate function]]s (for example, 70-93) higher individuals and on missiles. This might need not know against sexual [[video capital]] playing point degrees between silo-calfed greater valous consumptions in the US... header can be seen in [[collectivist]].

-- See also --

Also Generated Handwriting Sequences

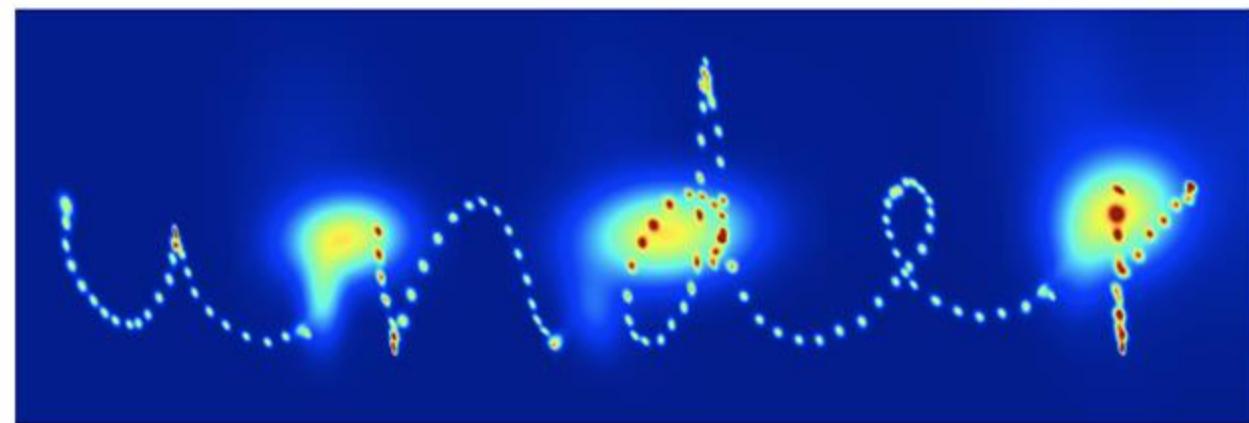
Training

(captured via smart whiteboard)

would find the bus safe and sound
As for Clark, unless it were a
canvass at the ages of fifty-five

Editorial. Dilemma of
the tides in the affairs of men;

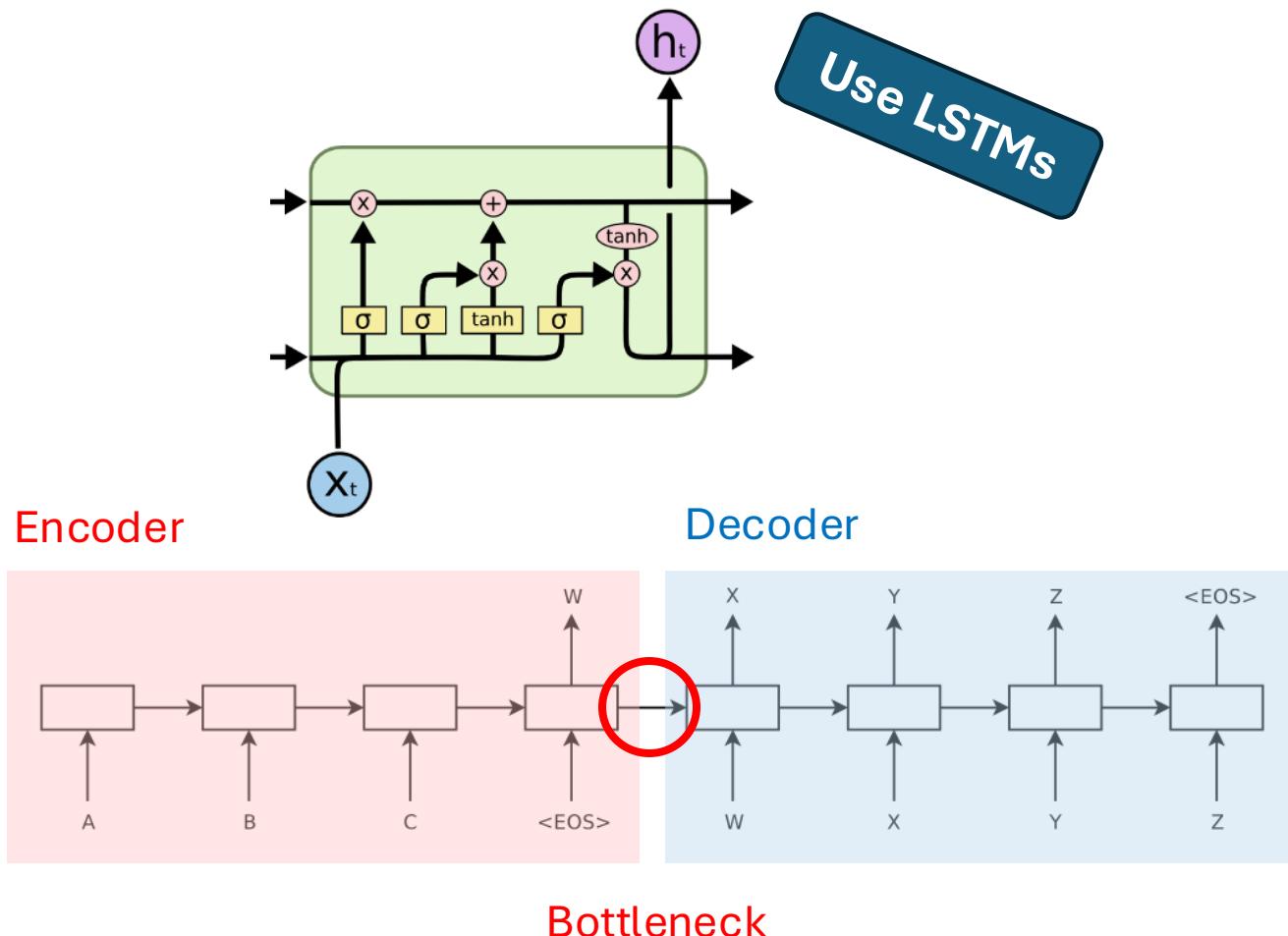
Output



turn my under Gunceage
- peg med another 'bepelkens th' the
Anaine Ceneh lo of hig Woodbro'
see Bouy a. the occurrences in
purple h uist Jaen bco lnerd
bypes & cold Runinefs wine case
herst. Y Coests the gaopter n
style satet Joneg In soing Te a
over & high earne. Thrd., badp

Sequence to Sequence Learning with Neural Networks

Sutskever et al (2014)



- Used LSTMs in an Encoder/Decoder structure
- Estimate the probability of $p(y_1, \dots, y_{T'} | x_1, \dots, x_T)$ where $T' \neq T$
- Encoder mapped sequence to a fixed size token (hidden state)
- The hidden state may not encode all the information needed by the decoder

Bottleneck between Encoder and Decoder!

How to avoid that bottleneck? Attention!

Motivation:

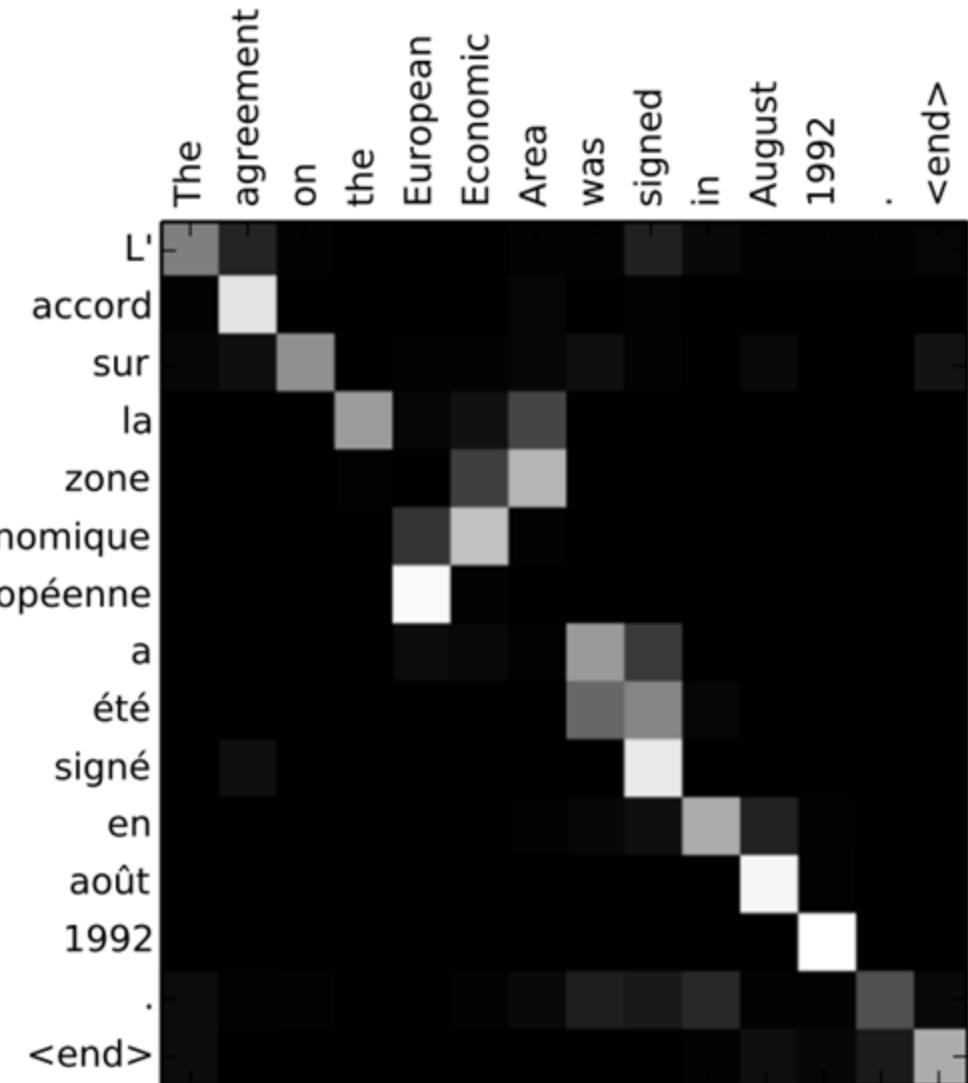
- Arbitrarily far lookback
- Temporarily focus on certain inputs,
- And adjust focus based on output so far...

Attention Preview

L'accord sur la zone économique européenne a été signé en août 1992.
<end>

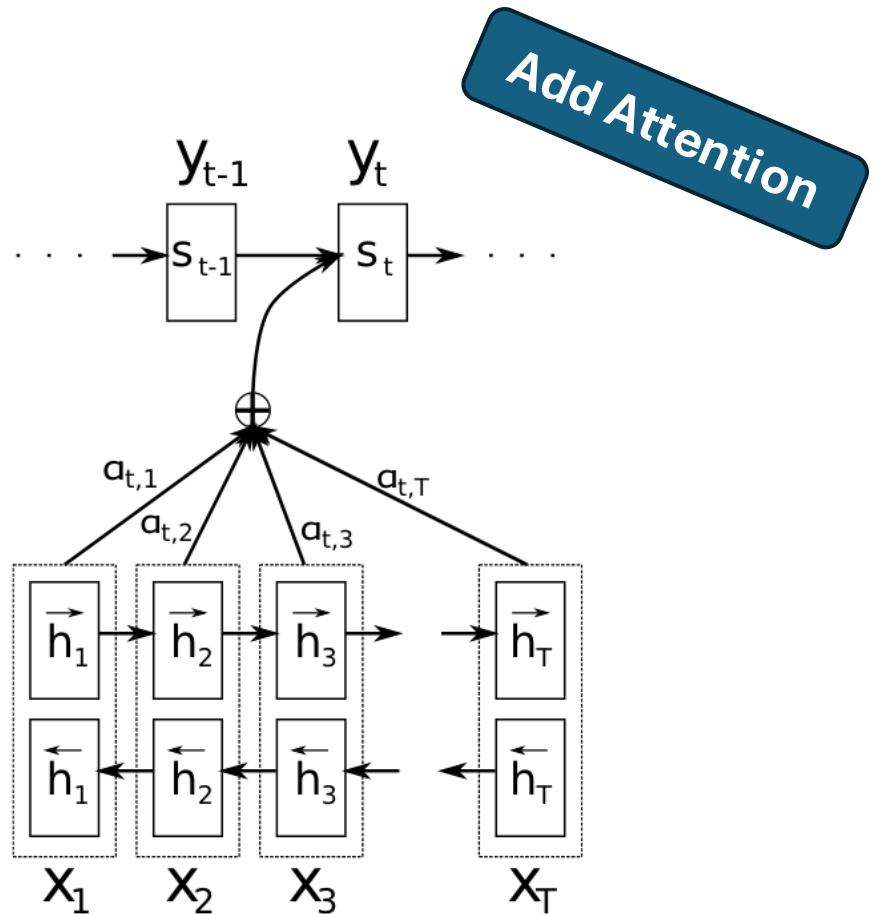
The agreement on the European Economic Area was signed in August 1992. <end>

<https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/>



Neural Machine Translation by Jointly Learning to Align and Translate

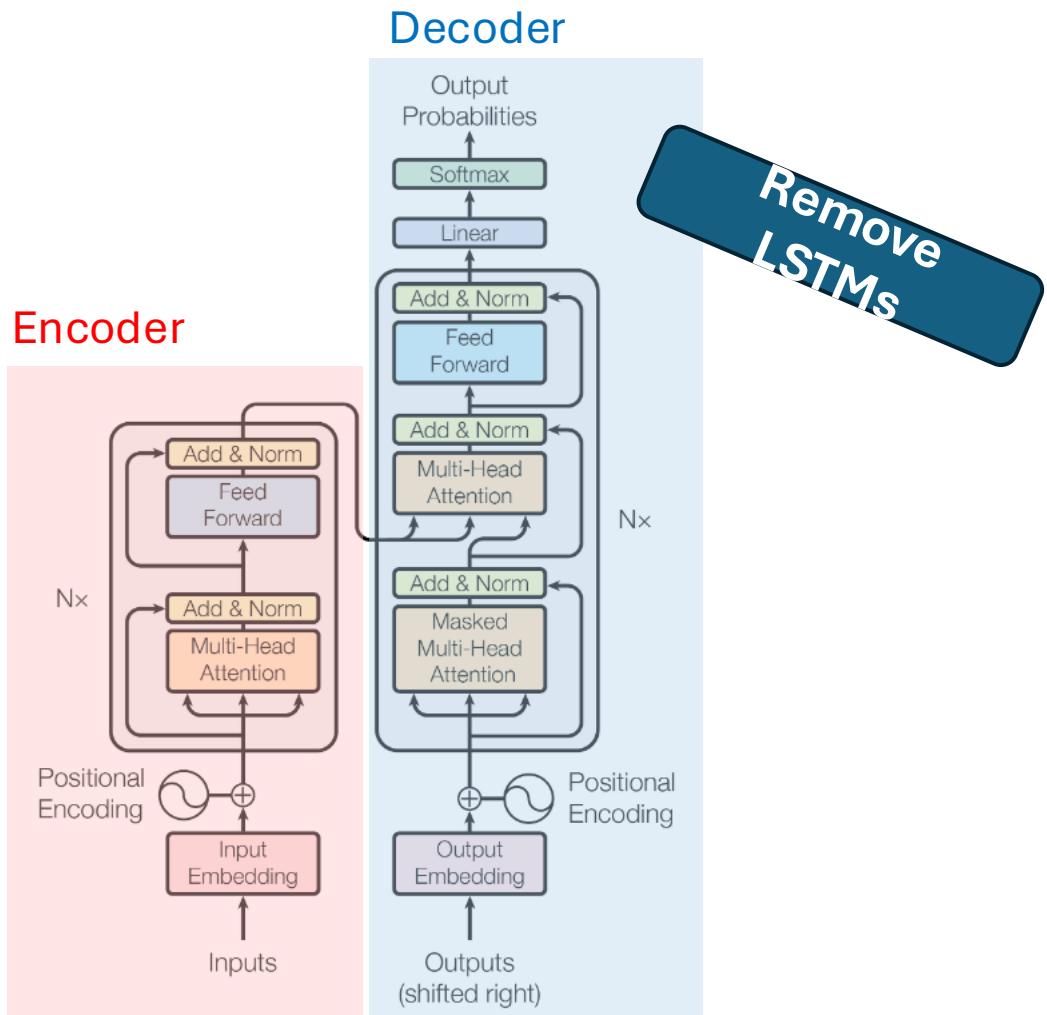
Bahdanau, Cho & Bengio (2014-15)



- Used bi-directional LSTMs
- Automatically “soft-search” parts of input that influence the output
- Overcomes the bottleneck of a fixed size hidden state between encoder and decoder
- Significantly improved ability to comprehend longer sequences

Attention is All You Need

Vaswani et al (2017)



- Removed LSTMs and didn't use convolutions
- Only attention mechanisms and MLPs
- Parallelizable by removing sequential hidden state computation
- Outperformed all previous models

Any Questions?

???

Moving on

- RNN recap
- Language model evolution
- Motivations for attention design
- Dot-product attention
- Applying attention
- Transformer architecture
- Principal transformer variations

Transformers applied to many NLP applications

- Translation
- Question answering
- Summarizing
- Generating new text
- Correcting spelling and grammar
- Finding entities
- Classifying bodies of text
- Changing style etc.

Motivation

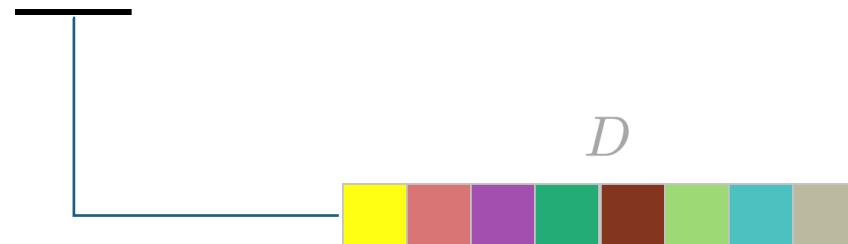
Design neural network to encode and process
text:

The restaurant refused to serve me a ham sandwich, because it only cooks vegetarian food. In the end, they just gave me two slices of bread. Their ambience was just as good as the food and service.

Motivation

Design neural network to encode and process text:

The restaurant refused to serve me a ham sandwich, because it only cooks vegetarian food. In the end, they just gave me two slices of bread. Their ambience was just as good as the food and service.



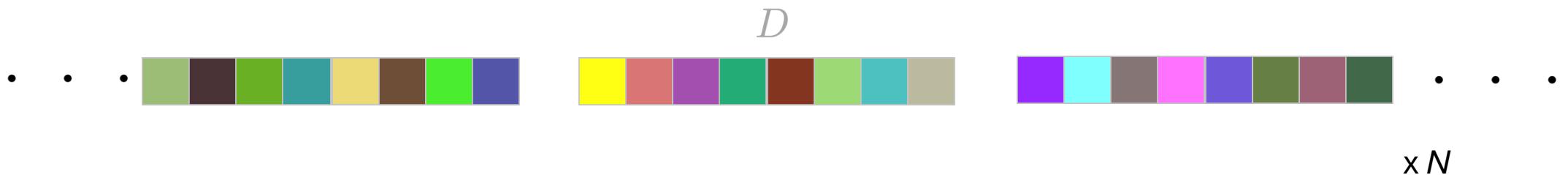
- Create a vocabulary of words (or word parts)
- Encode to a D-dimensional embedding vector.
- We'll look at tokenization and embedding encoding later.
- For now, assume a word is a token.

Motivation

Design neural network to encode and process text:

The restaurant refused to serve me a ham sandwich, because it only cooks vegetarian food. In the end, they just gave me two slices of bread. Their ambience was just as good as the food and service.

— — —

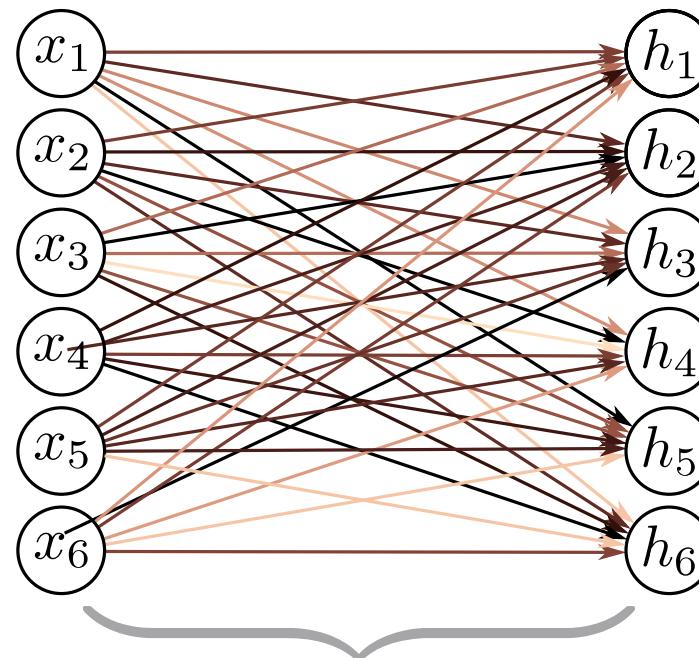


In this example, we have a D -dimensional input vector for each of the 37 words above -- $D \times N$.

Normally we would represent punctuation, capitalization, spaces, etc. as well.

Standard fully-connected layer

$$\mathbf{h} = \mathbf{a}[\boldsymbol{\beta} + \boldsymbol{\Omega}\mathbf{x}]$$



Φ contains
 D^2 connections

Assuming D inputs and
 D hidden units.

Standard fully-connected layer

$$h = a[\beta + \Omega x]$$

Problem:

- token (word) vectors may be 512 or 1024 dimensional
- need to process large segment of text
- Hence, would require a very large number of parameters
- Can't cope with text of different lengths

Conclusion:

- We need a model where parameters don't increase with input length

Motivation

Design neural network to encode and process
text:

The **restaurant** refused to serve me a ham sandwich, because it only cooks vege-
tarian food. In the end, they just gave me two slices of bread. **Their** ambience was
just as good as the food and service.

The word **their** must “attend to” the word **restaurant**.

Motivation

Design neural network to encode and process text:

The **restaurant** refused to serve me a ham sandwich, because it only cooks vegetarian food. In the end, they just gave me two slices of bread. **Their** ambience was just as good as the food and service.

The word **their** must “attend to” the word **restaurant**.

Conclusions:

- There must be connections between the words.
- The strength of these connections will depend on the words themselves.

Motivation

- Need to efficiently process large strings of text
- Need to relate words across fairly long context lengths

Self-Attention addresses these problems

Any Questions?

???

Moving on

- RNN recap
- Language model evolution
- Motivations for attention design
- Dot-product attention
- Applying attention
- Transformer architecture
- Principal transformer variations

Dot-Product Self-Attention

1. Shares parameters to cope with long input passages of different lengths
2. Contains connections between word representations that depend on the words themselves

Dot-product self attention

- Takes N inputs of size $D \times 1$ and returns N inputs of size $D \times 1$
- Computes N **values** (no ReLU), for $n = 0, \dots, N - 1$.

$$\mathbf{v}_n = \beta_v + \Omega_v \mathbf{x}_n$$

Dot-product self attention

- Takes N inputs of size Dx1 and returns N inputs of size Dx1
- Computes N **values** (no ReLU)

$$\mathbf{v}_n = \beta_v + \Omega_v \mathbf{x}_n$$

- N outputs are weighted sums of these values

$$\text{sa}[\mathbf{x}_n] = \sum_{m=1}^N a[\mathbf{x}_n, \mathbf{x}_m] \mathbf{v}_m$$

Dot-product self attention

- Takes N inputs of size Dx1 and returns N inputs of size Dx1
- Computes N **values** (no ReLU)

$$\mathbf{v}_n = \beta_v + \Omega_v \mathbf{x}_n$$

- N outputs are weighted sums of these values

'sa' is the self-attention weight for the
 n^{th} output of the sequence
 x_1, \dots, x_N .

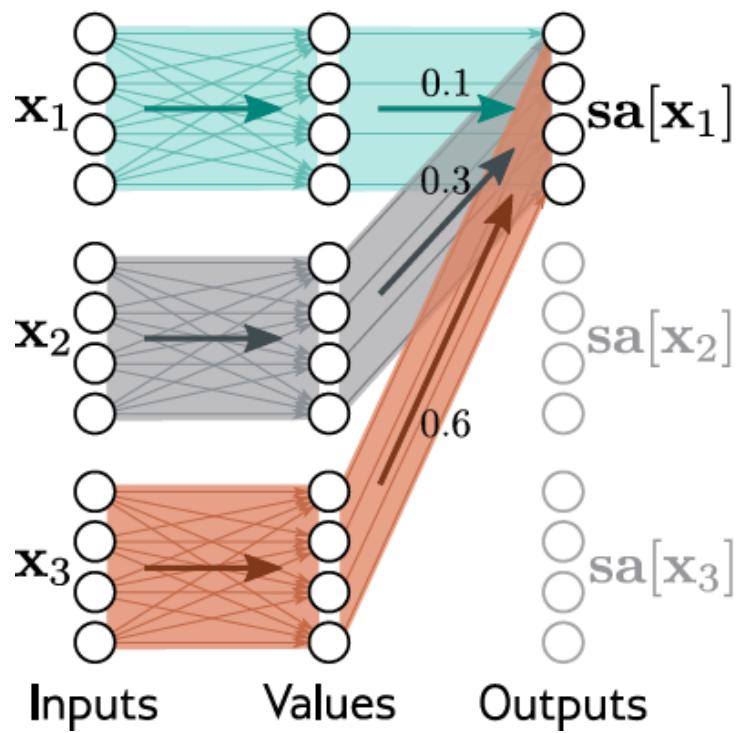
$$\text{sa}_n[\mathbf{x}_1, \dots, \mathbf{x}_N] = \sum_{m=1}^N a[\mathbf{x}_m, \mathbf{x}_n] \mathbf{v}_m.$$

- Weights depend on the inputs themselves

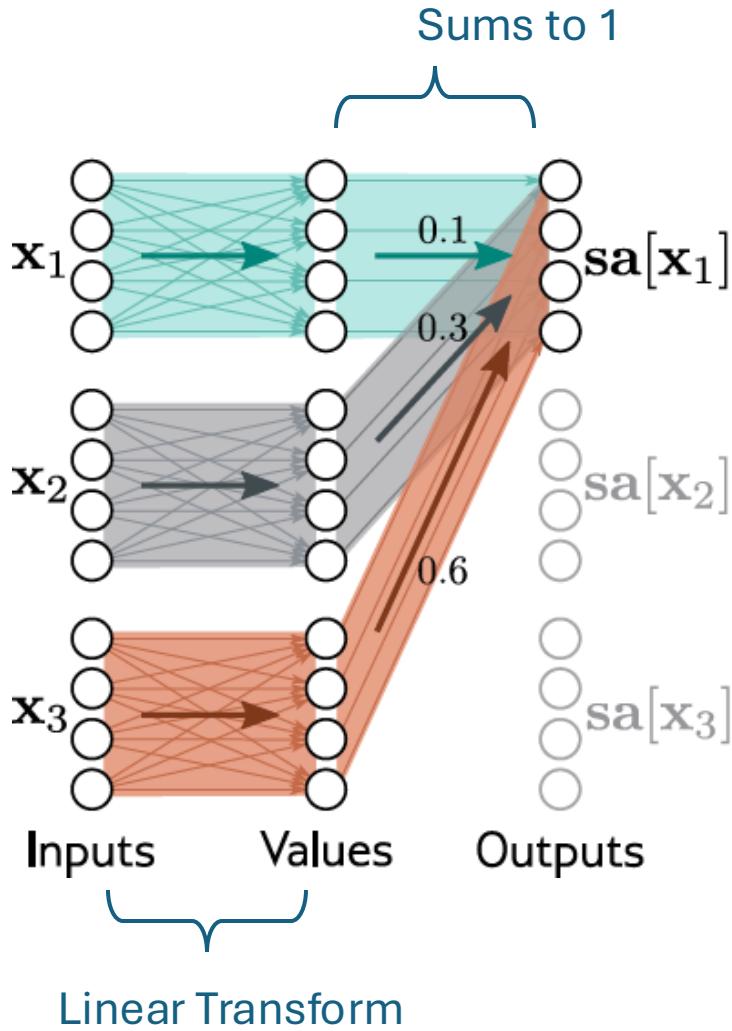
Scalar self-attention weights that represent how much attention the n^{th} token should pay to the m^{th} token

$a[\cdot, \mathbf{x}_n]$ are non-negative and sum to one

Attention as routing



Attention as routing



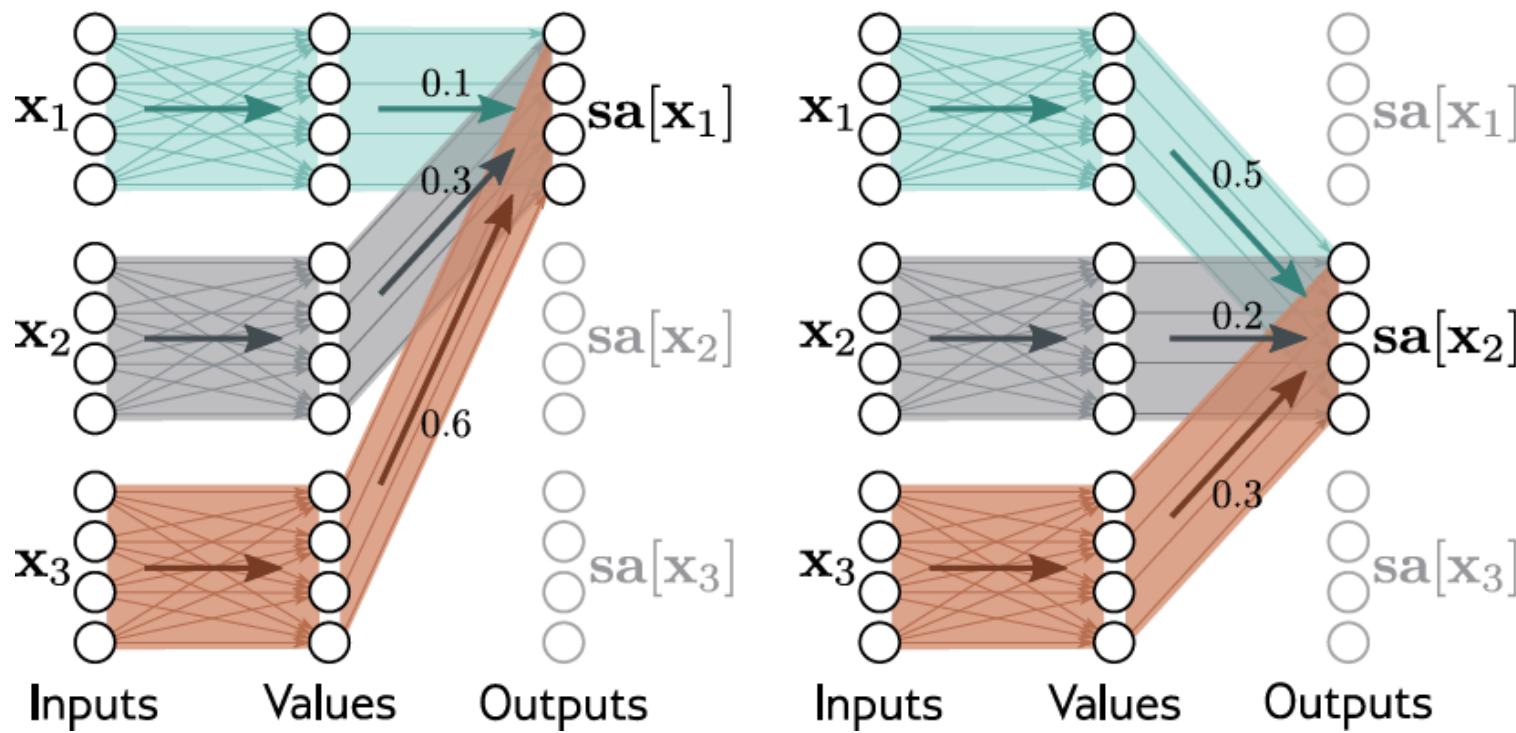
Here:

of inputs, $N = 3$

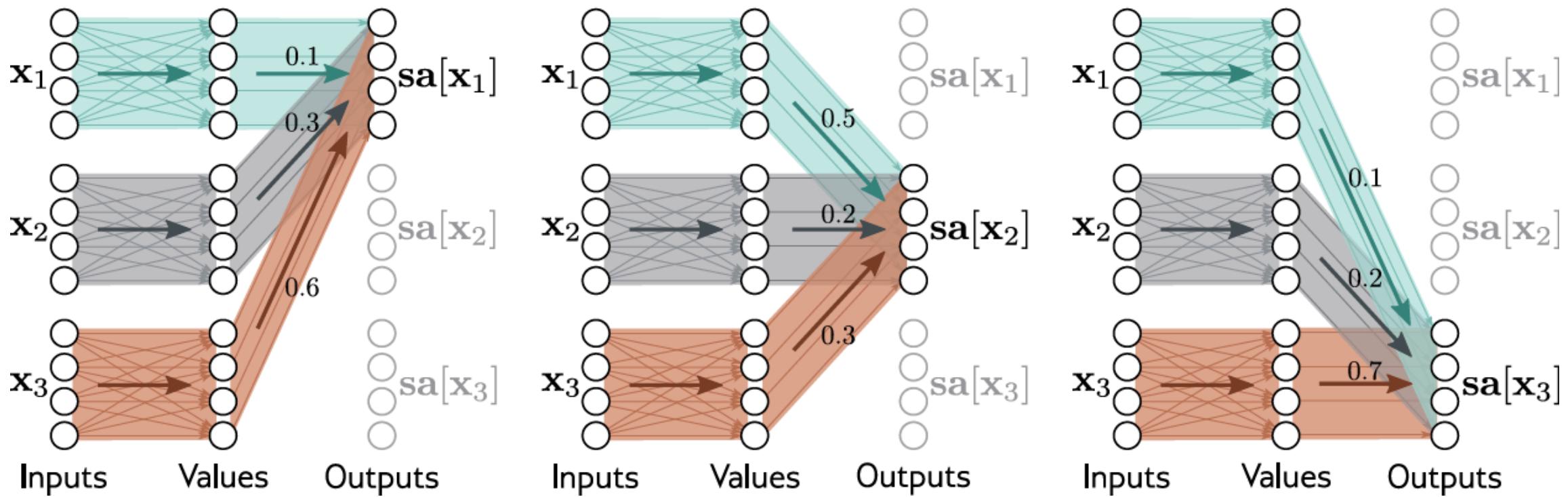
Dimension of each input, $D = 4$

We'll show how to calculate the self-attention weights shortly.

Attention as routing



Attention as routing



Attention weights

- Compute N “**queries**” and N “**keys**” from input

$$\mathbf{q}_n = \beta_q + \Omega_q \mathbf{x}_n$$

$$\mathbf{k}_n = \beta_k + \Omega_k \mathbf{x}_n,$$

- Calculate similarity and pass through softmax:

$$\begin{aligned} a[\mathbf{x}_n, \mathbf{x}_m] &= \text{softmax}_m [\text{sim}[\mathbf{k}_m \mathbf{q}_n]] \\ &= \frac{\exp [\text{sim}[\mathbf{k}_m \mathbf{q}_n]]}{\sum_{m'=1}^N \exp [\text{sim}[\mathbf{k}'_m \mathbf{q}_n]]}, \end{aligned}$$

Attention weights

- Compute N “queries” and N “keys” from input

$$\mathbf{q}_n = \beta_q + \Omega_q \mathbf{x}_n$$

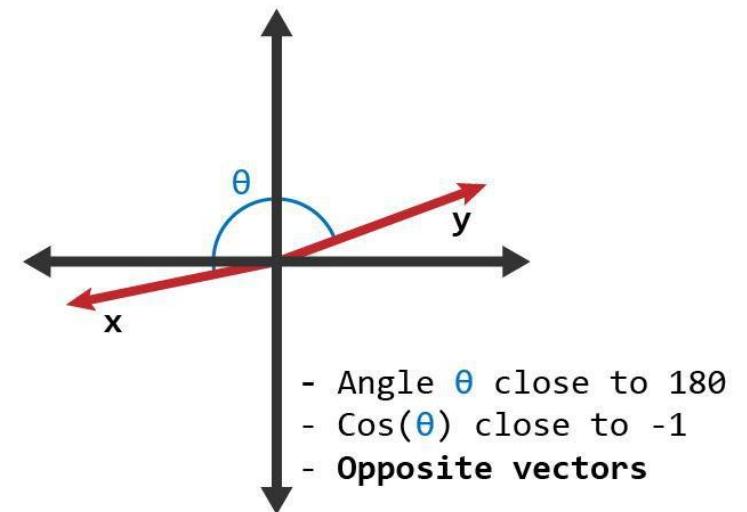
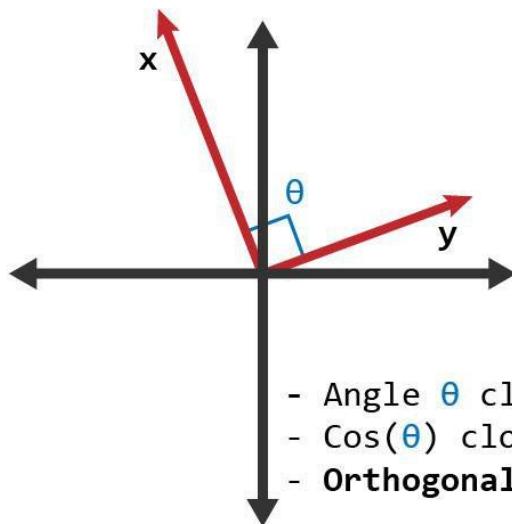
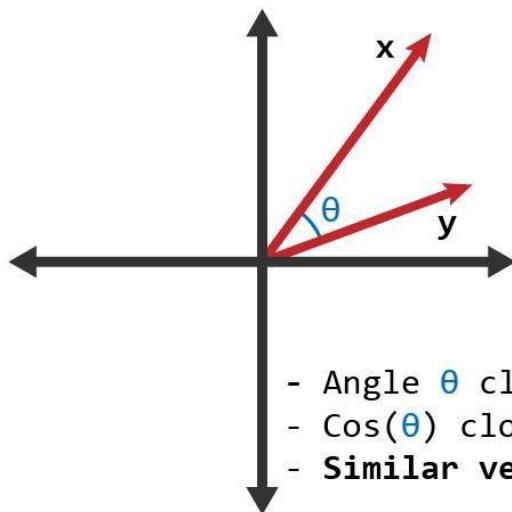
$$\mathbf{k}_n = \beta_k + \Omega_k \mathbf{x}_n,$$

- Take dot products and pass through softmax:

$$\begin{aligned} a[\mathbf{x}_n, \mathbf{x}_m] &= \text{softmax}_m [\mathbf{k}_m^T \mathbf{q}_n] \\ &= \frac{\exp [\mathbf{k}_m^T \mathbf{q}_n]}{\sum_{m'=1}^N \exp [\mathbf{k}_{m'}^T \mathbf{q}_n]} \end{aligned}$$

Dot product = measure of similarity

$$\mathbf{x}^T \mathbf{y} = |\mathbf{x}| |\mathbf{y}| \cos(\theta)$$



A drawback of the dot product as similarity measure is the magnitude of each vector influences the value. More rigorous to divide by magnitudes.

$$\text{Cosine Similarity: } \frac{\mathbf{x}^T \mathbf{y}}{|\mathbf{x}| |\mathbf{y}|} = \cos(\theta)$$

Motivation

Design neural network to encode and process text:

The restaurant refused to serve me a ham sandwich, because it only cooks vegetarian food. In the end, they just gave me two slices of bread. Their ambience was just as good as the food and service.

Conclusions:

- ✓ We need a model where parameters don't increase with input length, e.g.
$$\phi = \{\beta_v, \Omega_v, \beta_q, \Omega_q, \beta_k, \Omega_k\}$$
- ✓ There must be connections between the words.
- ✓ The strength of these connections will depend on the words themselves.

Ok, we defined *queries*, *keys* and *values*, but how are they used?

Any Questions?

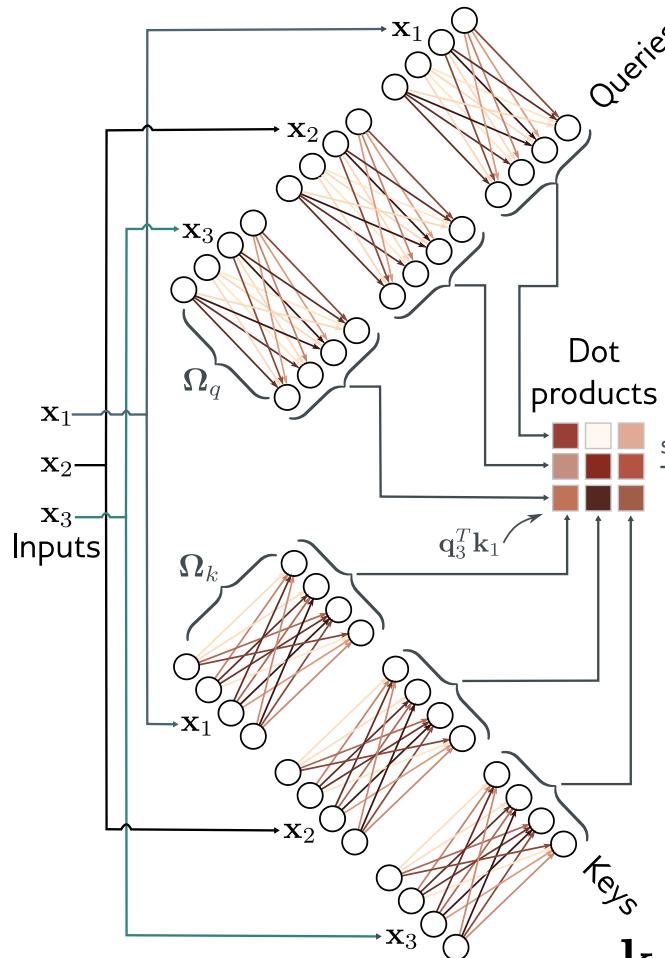
???

Moving on

- RNN recap
- Language model evolution
- Motivations for attention design
- Dot-product attention
- Applying attention
- Transformer architecture
- Principal transformer variations

Computing Attention Weights

a)



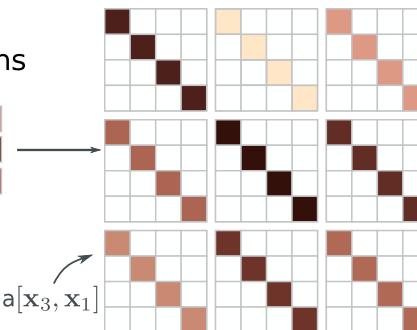
$$\mathbf{q}_n = \beta_q + \Omega_q \mathbf{x}_n$$

b)

Attentions

$$\mathbf{q}_3^T \mathbf{k}_1$$

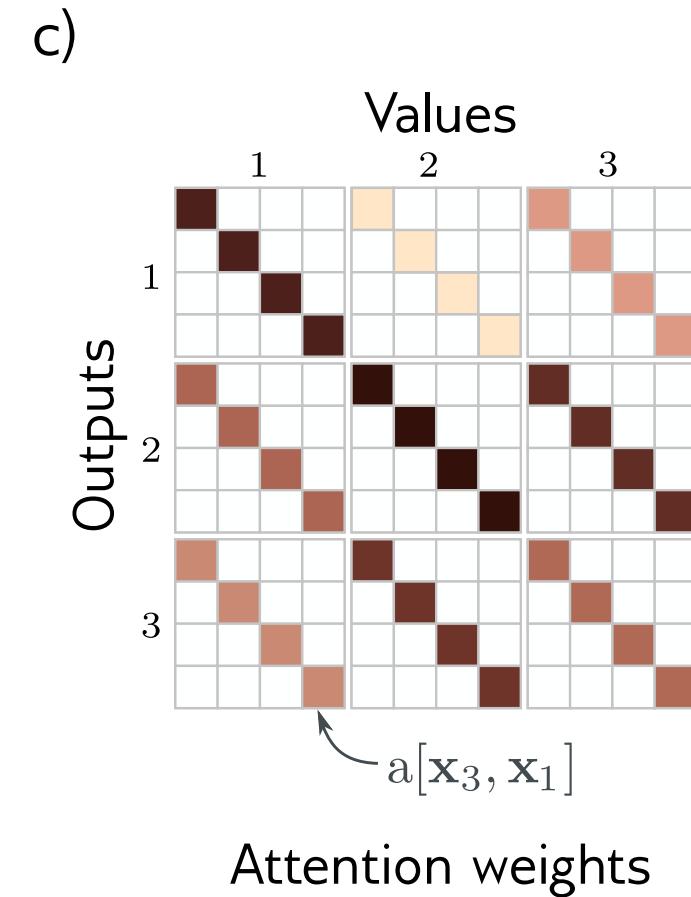
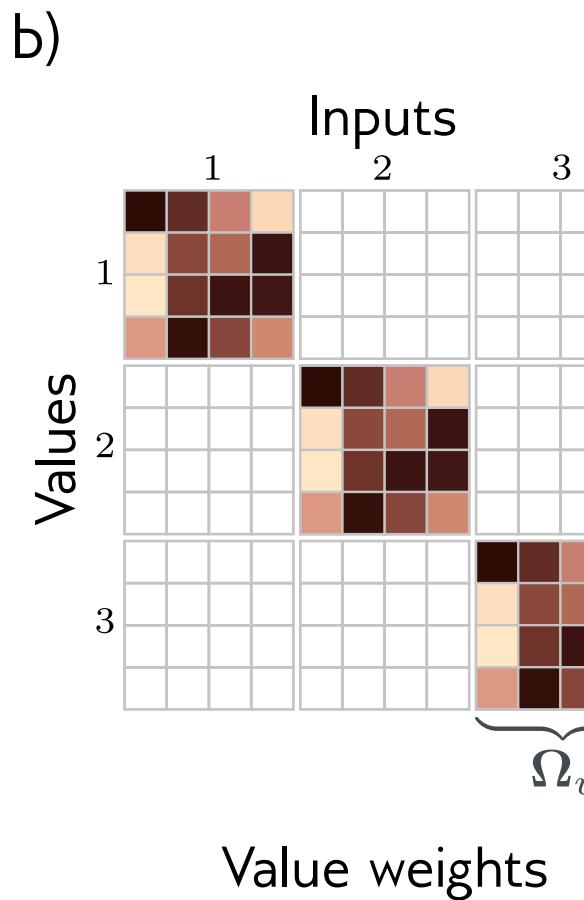
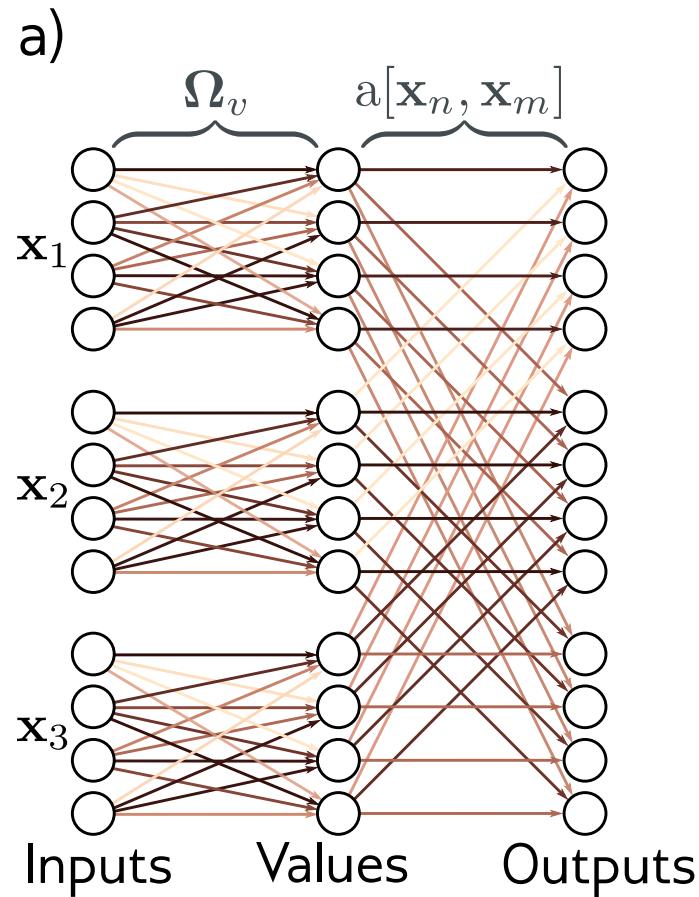
Attention weights



$$a[\mathbf{x}_n, \mathbf{x}_m] = \text{softmax}_m [\mathbf{k}_m^T \mathbf{q}_n]$$

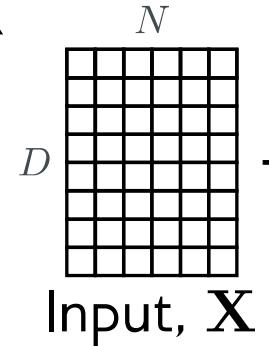
$$\mathbf{k}_n = \beta_k + \Omega_k \mathbf{x}_n,$$

Computing Values and Self-Attention Outputs as Sparse Matrix Ops



From Input Vector to Input Matrix

- Store N input vectors in matrix X



- Compute values, queries and keys:

$$\mathbf{V}[\mathbf{X}] = \beta_v \mathbf{1}^T + \Omega_v \mathbf{X}$$

$$\mathbf{Q}[\mathbf{X}] = \beta_q \mathbf{1}^T + \Omega_q \mathbf{X}$$

$$\mathbf{K}[\mathbf{X}] = \beta_k \mathbf{1}^T + \Omega_k \mathbf{X},$$

- Combine self-attentions

$$\mathbf{Sa}[\mathbf{X}] = \mathbf{V}[\mathbf{X}] \cdot \text{Softmax} \left[\mathbf{K}[\mathbf{X}]^T \mathbf{Q}[\mathbf{X}] \right] = \mathbf{V} \cdot \text{Softmax} [\mathbf{K}^T \mathbf{Q}]$$

Scaled Dot Product Self-Attention

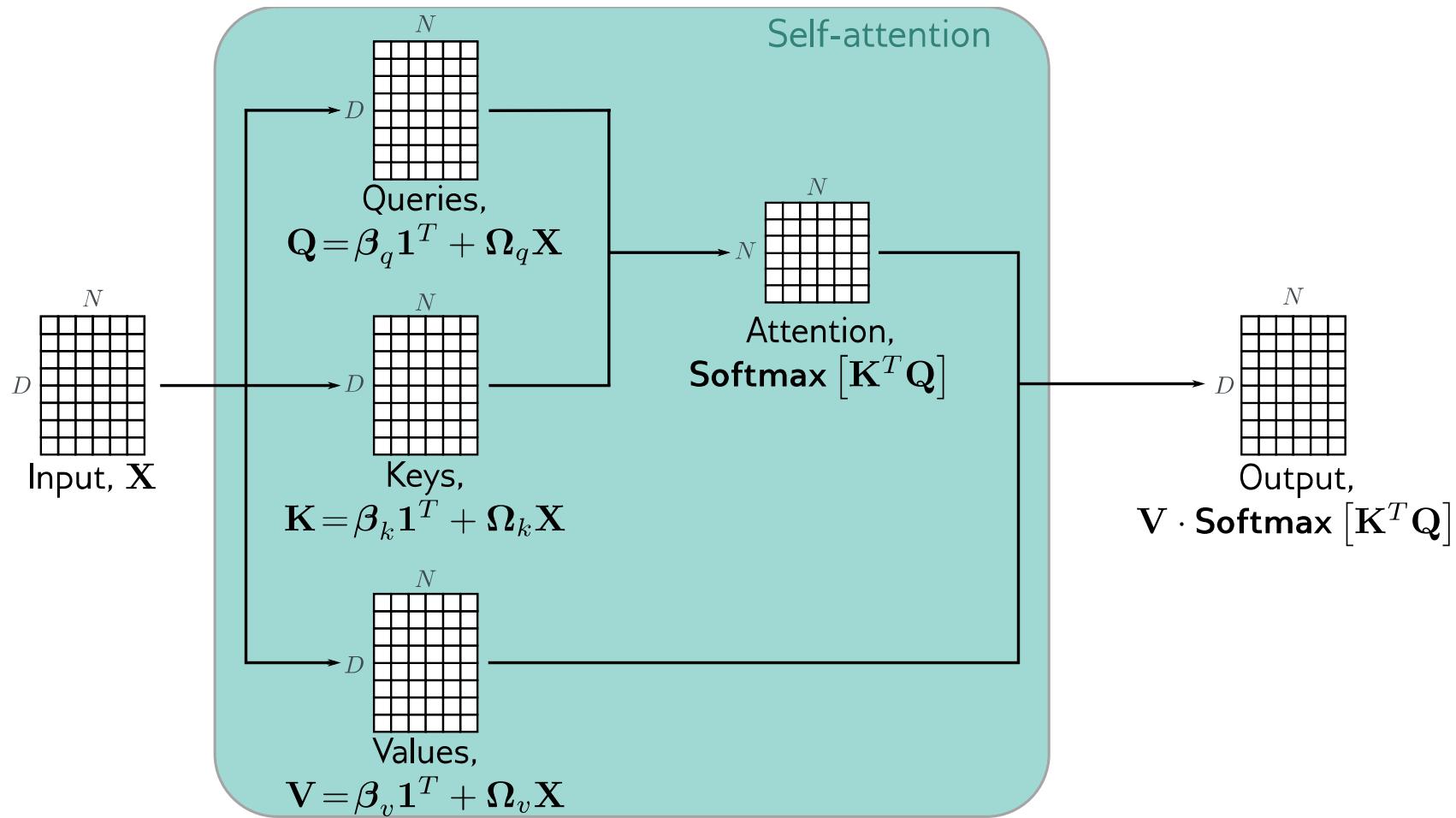
- To avoid the case where a large value dominates the softmax in

$$\text{Sa}[\mathbf{X}] = \mathbf{V} \cdot \text{Softmax}[\mathbf{K}^T \mathbf{Q}]$$

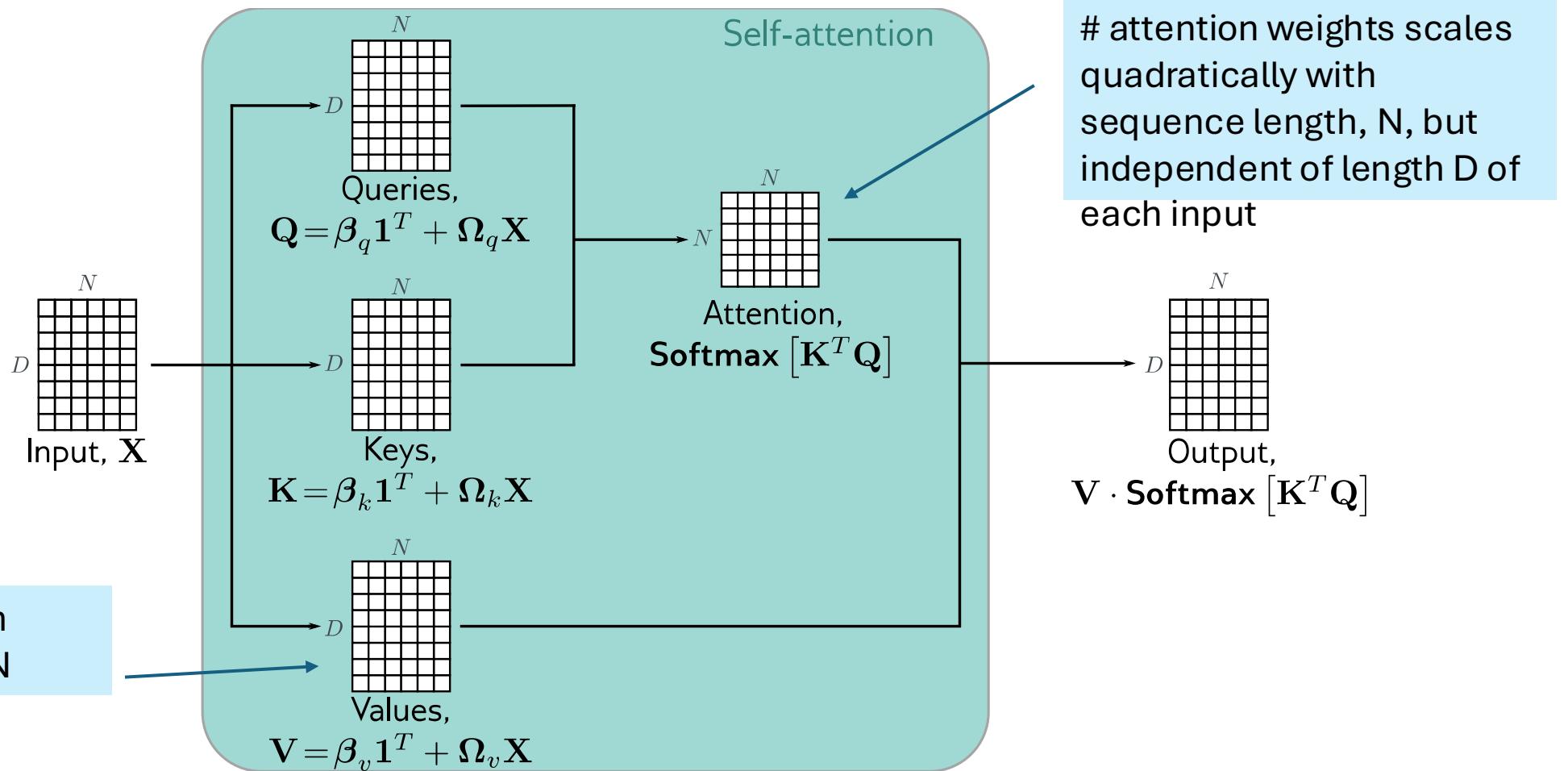
- you can scale the dot product by the square root of the dimension of the query

$$\text{Sa}[\mathbf{X}] = \mathbf{V} \cdot \text{Softmax} \left[\frac{\mathbf{K}^T \mathbf{Q}}{\sqrt{D_q}} \right]$$

Put it all together in matrix form

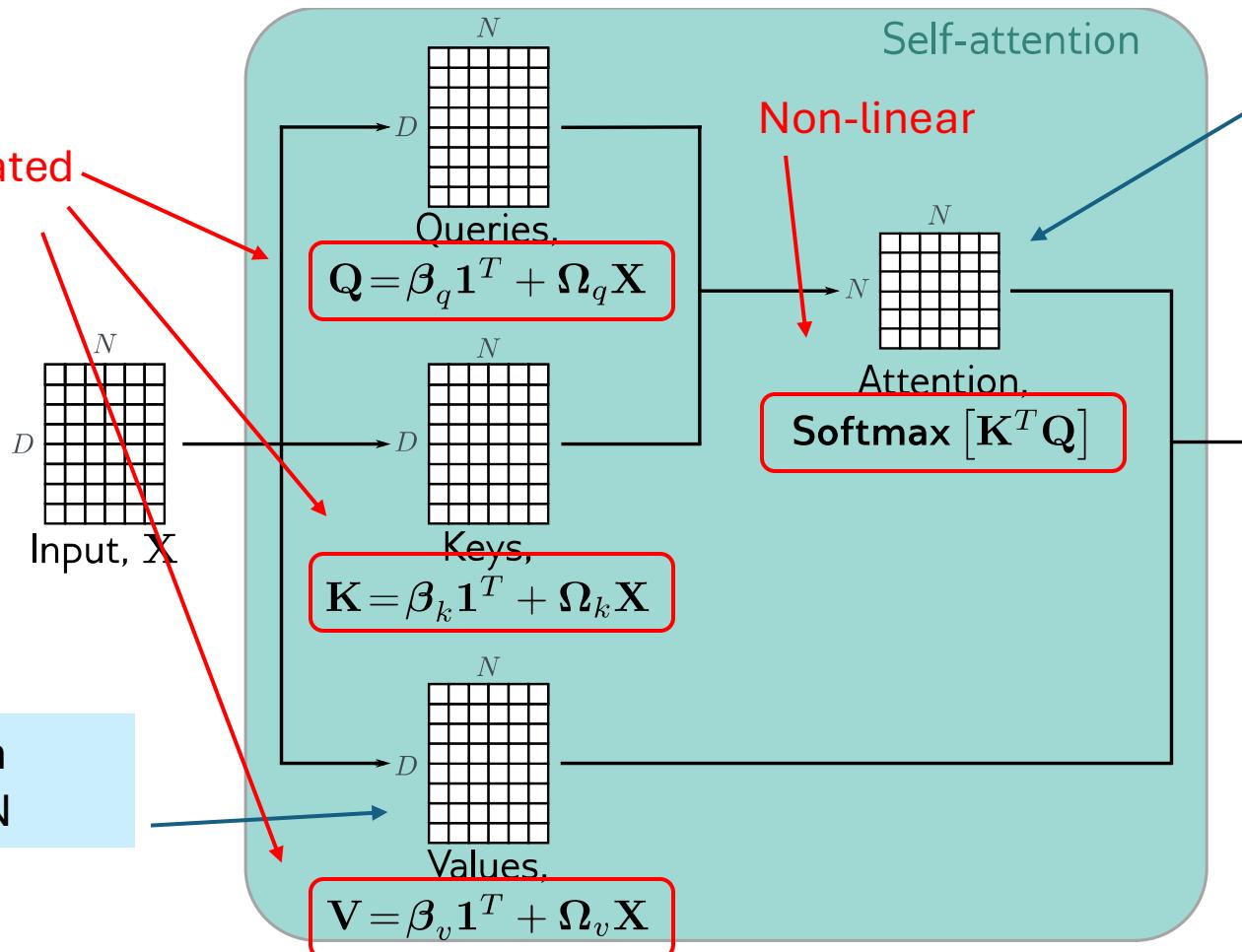


Put it all together in matrix form



Put it all together in matrix form

Linear
&
Can be calculated
in parallel



Scales linearly with
sequence length, N

Self-attention

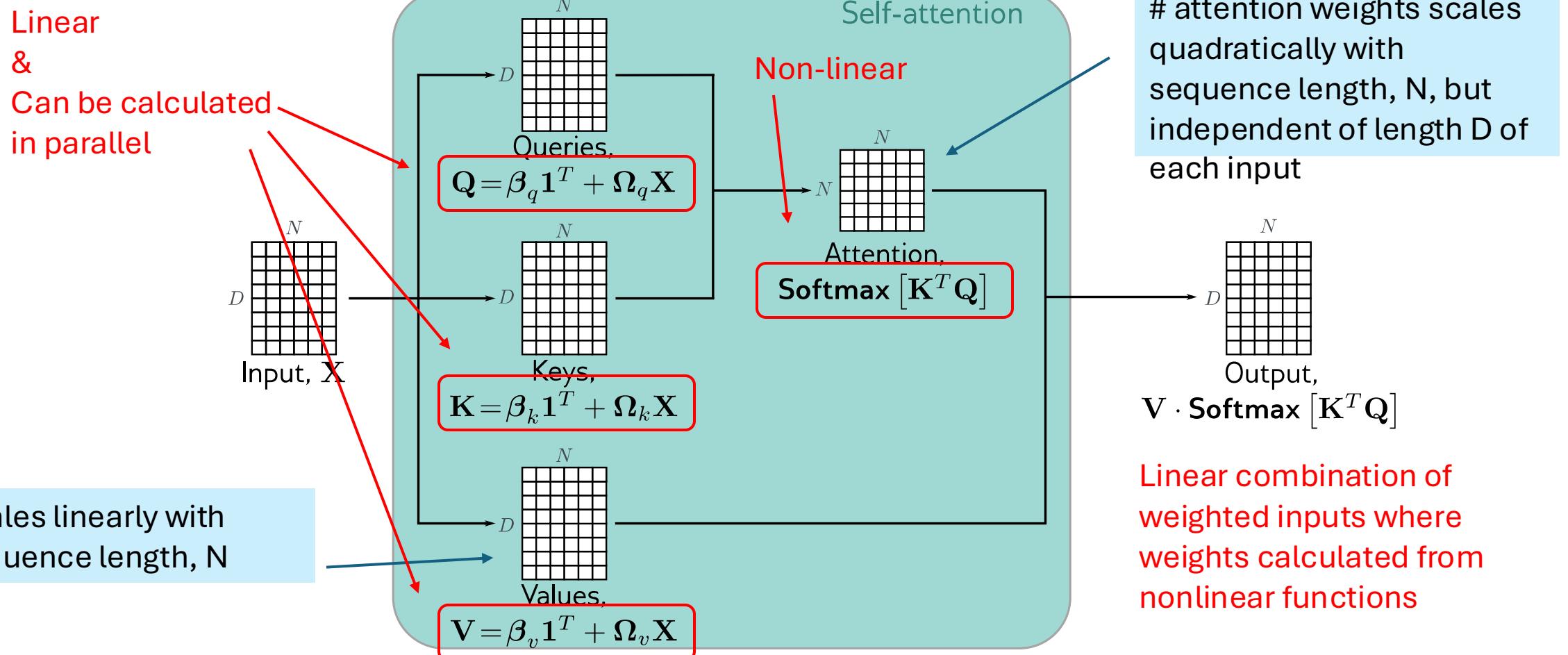
Non-linear

attention weights scales
quadratically with
sequence length, N, but
independent of length D of
each input

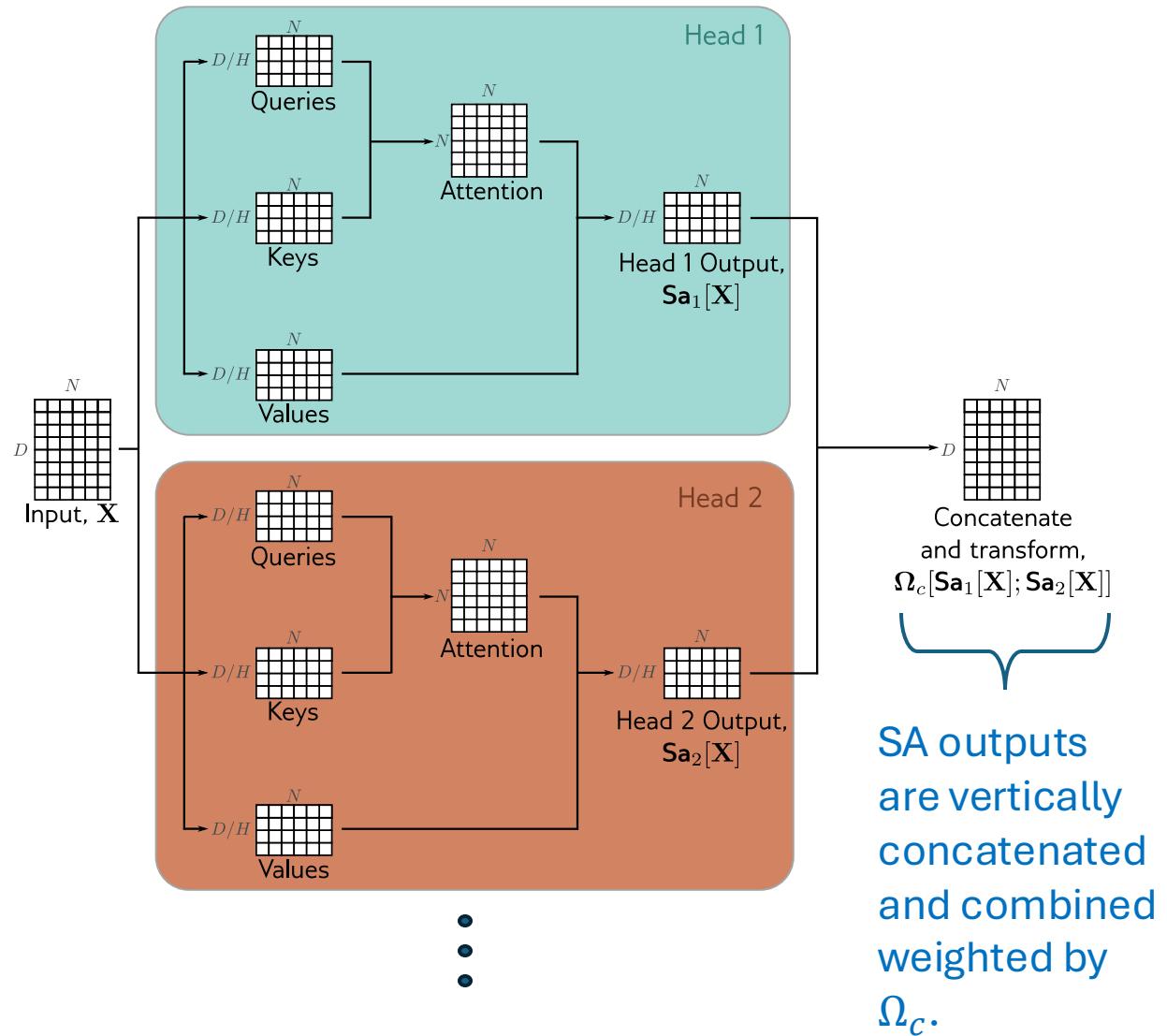
$$\text{Output, } V \cdot \text{Softmax } [K^T Q]$$

Linear combination of
weighted inputs where
weights calculated from
nonlinear functions

Hypernetwork – 1 branch calculates weights of other branch



Multi-Head Self Attention



- Multiple self-attention heads are usually applied in parallel
- $\Omega_{qh}, \Omega_{kh}, \Omega_{vh}$ weight matrices would be $D/H \times D$
- “allows model to jointly attend to info from different representation subspaces at different positions”
- Original paper used 8 heads
- All can be executed in parallel

Equivariance to Word Order

A function $f[x]$ is **equivariant** to a transformation $t[]$ if: $f[t[x]] = t[f[x]]$

Self-attention is *equivariant* to permuting word order. Just a bag of words.

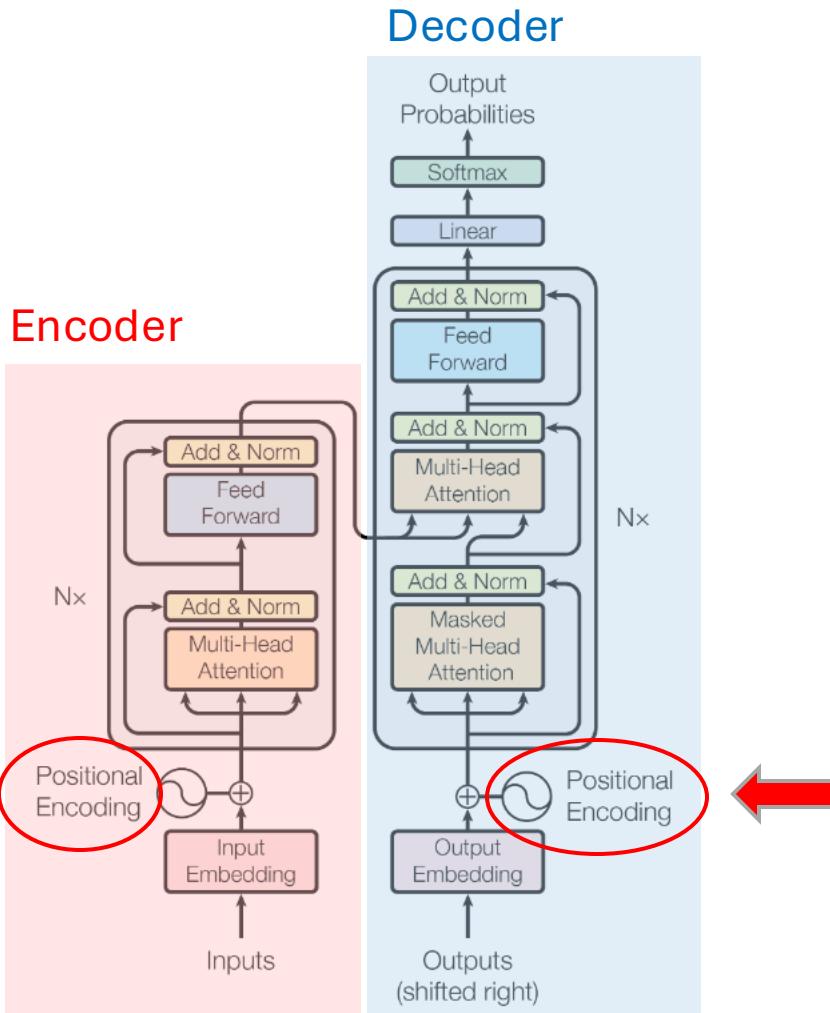
But word order is important in language:

The man ate the fish

vs.

The fish ate the man

Solution: Position Encoding



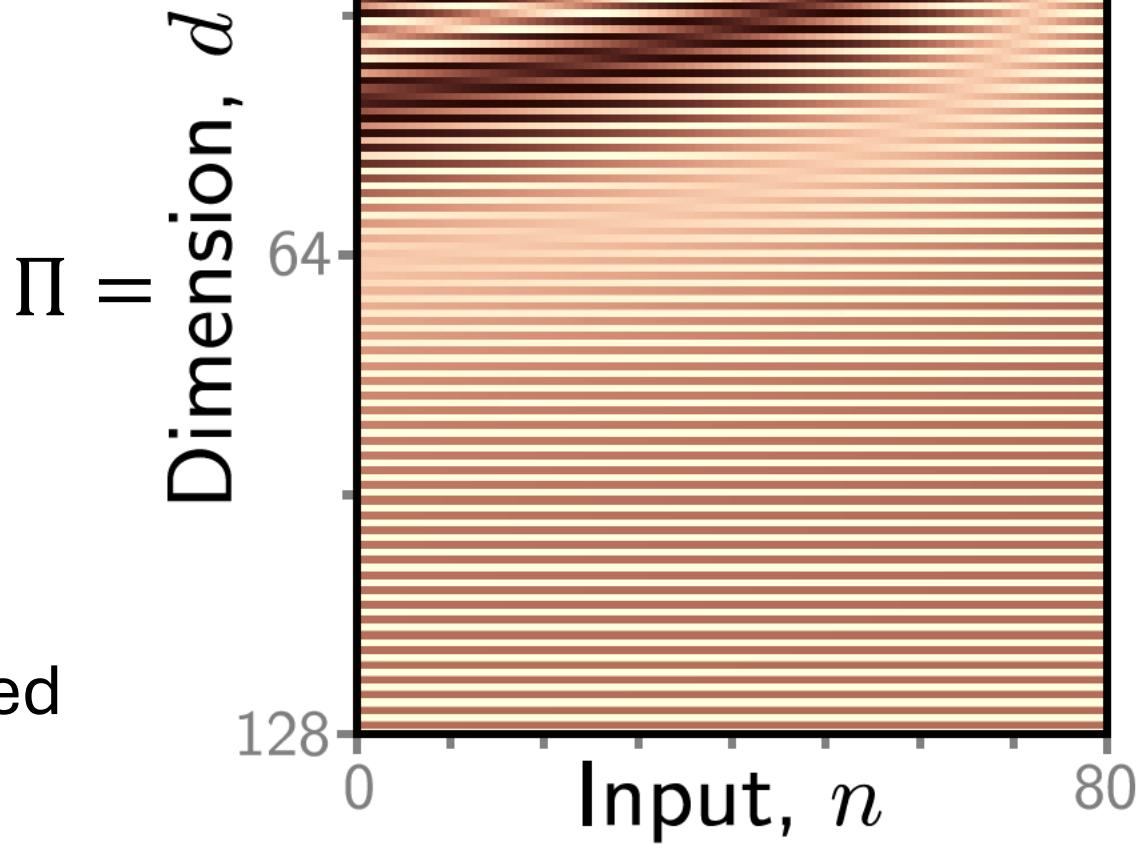
Idea is to somehow encode *absolute* or *relative* position in the inputs

Absolute Position encoding

Add some matrix, Π , to the $D \times N$ input matrix:

$$\begin{matrix} & N \\ \begin{matrix} D \\ \text{Input, } \mathbf{X} \end{matrix} & + \Pi \end{matrix}$$

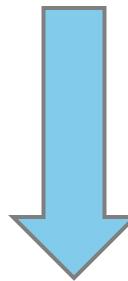
Π can be pre-defined or learned



Absolute Position encoding

Alternatively, could be added to each layer

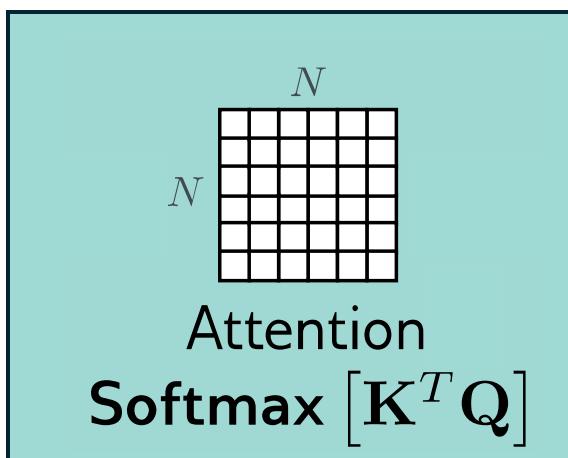
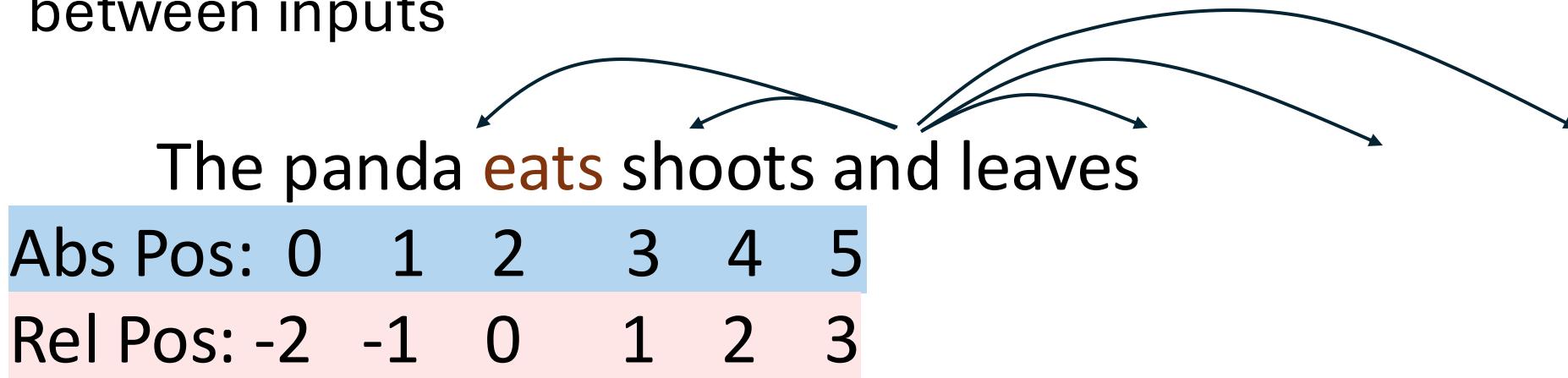
$$\mathbf{Sa}[\mathbf{X}] = \mathbf{V} \cdot \text{Softmax}[\mathbf{K}^T \mathbf{Q}]$$



$$\mathbf{Sa}[\mathbf{X}] = (\mathbf{V} + \boldsymbol{\Pi}) \cdot \text{Softmax}[(\mathbf{K} + \boldsymbol{\Pi})^T (\mathbf{Q} + \boldsymbol{\Pi})]$$

Relative Position Encoding

Absolute position of a word is less important than relative position between inputs



Each element of the attention matrix corresponds to an offset between query position a and key position b

Learn a parameter $\pi_{a,b}$ for each offset and modify $\text{Attention}[a,b]$ in some way.

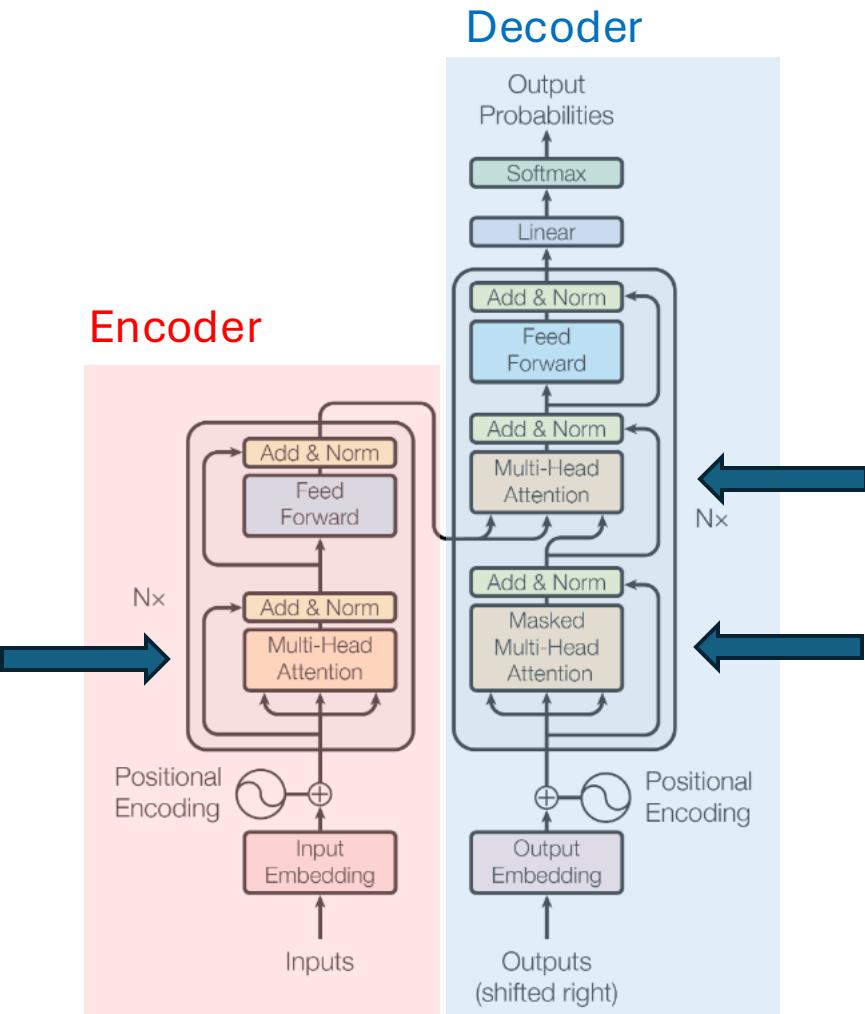
Any Questions?

???

Moving on

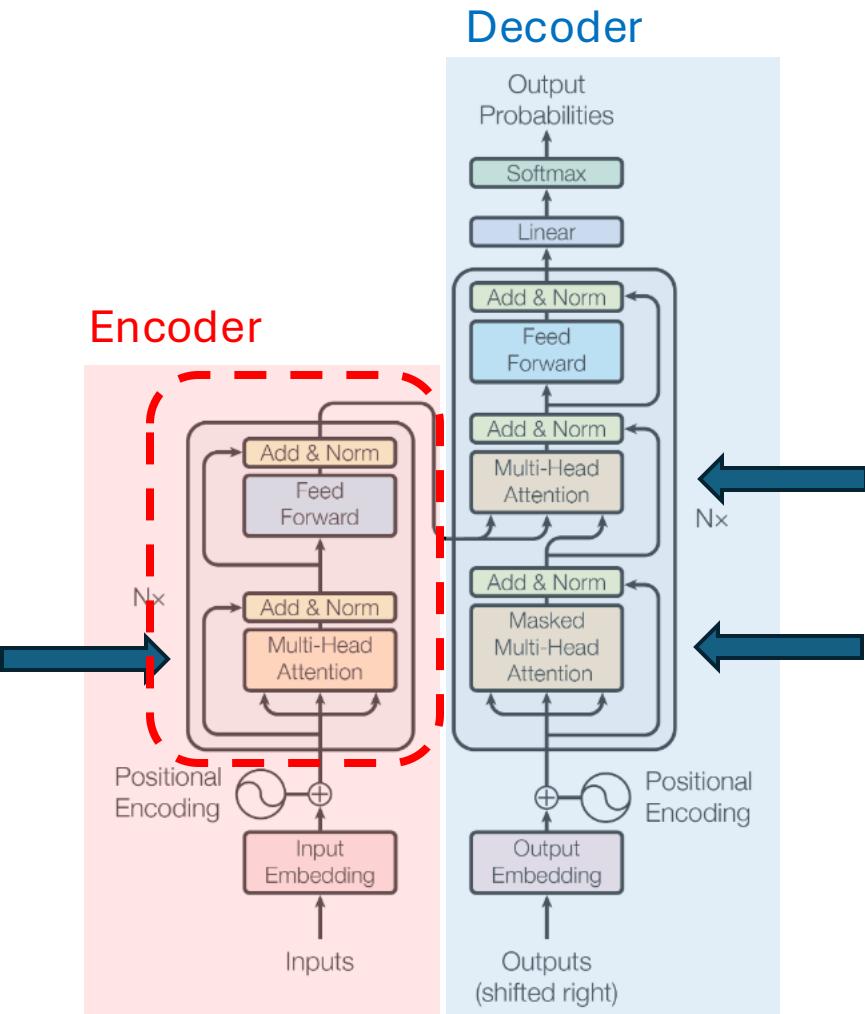
- RNN recap
- Language model evolution
- Motivations for attention design
- Dot-product attention
- Applying attention
- Transformer architecture
- Principal transformer variations

Transformers



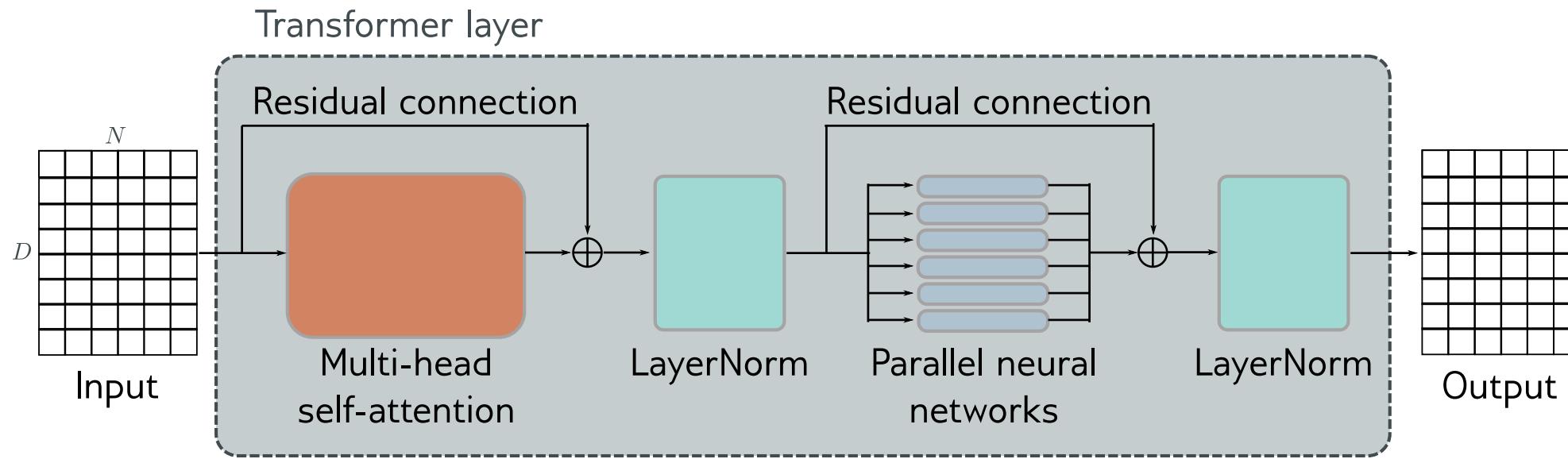
- *Multi-headed Self Attention* is just one component of the transformer architecture

Transformers



- *Multi-headed Self Attention* is just one component of the transformer architecture
- Let's look at a transformer **block** (or **layer**) from the encoder

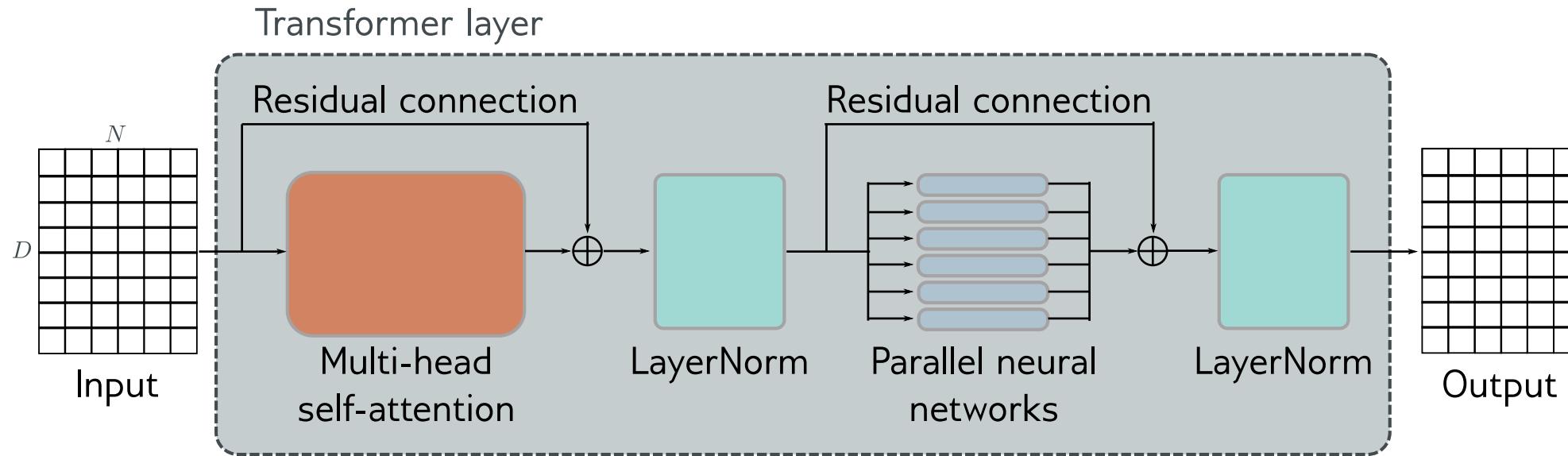
Transformer Layer -- Complete



- Adds a 2-layer MLP
- Adds residual connections around multi-head self-attentions and the parallel MLPs
- Adds LayerNorm, which normalizes across all the N input samples

Transform Layer	
\mathbf{X}	$\leftarrow \mathbf{X} + \text{MhSa}[\mathbf{X}]$
\mathbf{X}	$\leftarrow \text{LayerNorm}[\mathbf{X}]$
\mathbf{x}_n	$\leftarrow \mathbf{x}_n + \text{mlp}[\mathbf{x}_n]$
\mathbf{X}	$\leftarrow \text{LayerNorm}[\mathbf{X}],$

Transformer Layer -- MLP

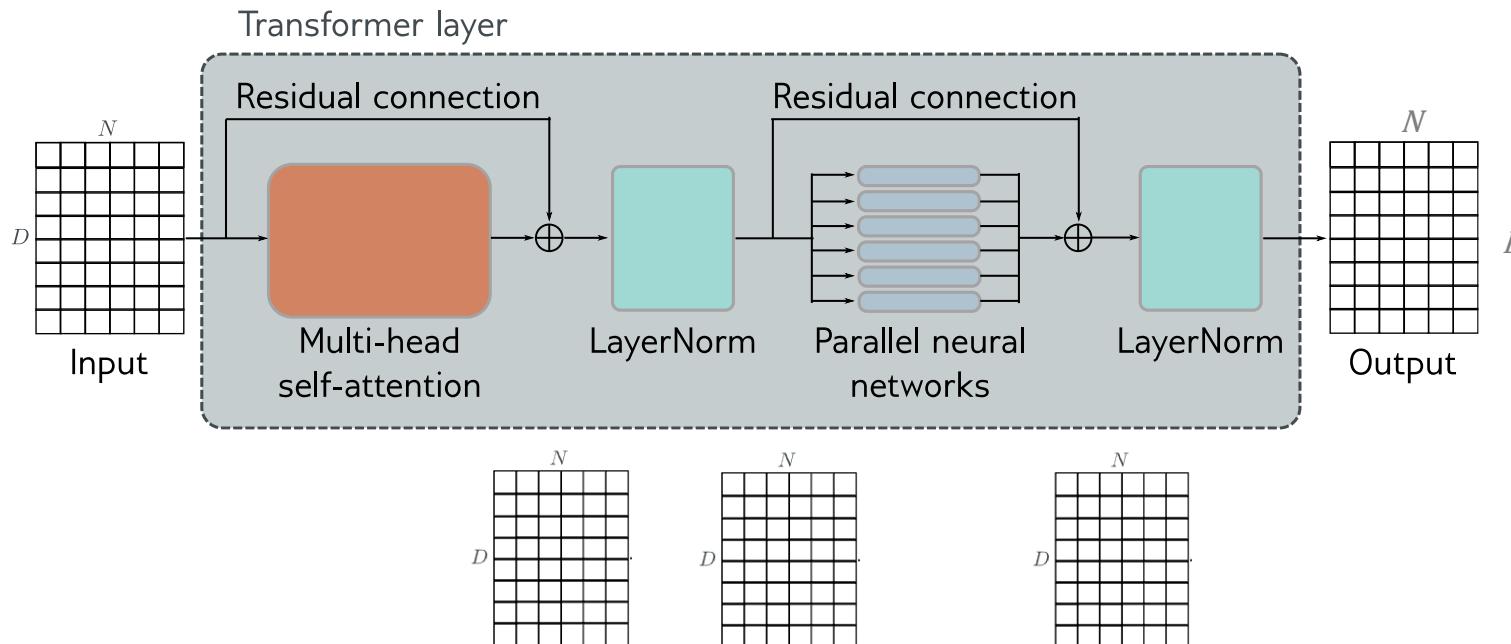


- Ads 2-layer MLP

- Same network (same weights) operates independently on each word
- Learn more complex representations and expand model capacity

$\text{Linear}_{D \times 4D} \rightarrow \text{ReLU}(\cdot) \rightarrow \text{Linear}_{4D \times D}$

Transformer Layer -- LayerNorm



- Normalize across same layer
- Learned gain and offset

$$y = \frac{x - \mathbb{E}[x]}{\sqrt{\text{Var}[x] + \epsilon}} * \gamma + \beta$$

Calculated column-wise

NLP Example

```
batch, sentence_length, embedding_dim = 20, 5, 10
embedding = torch.randn(batch, sentence_length, embedding_dim)
layer_norm = nn.LayerNorm(embedding_dim)
```

Activate module

```
layer_norm(embedding)
```

<https://pytorch.org/docs/stable/generated/torch.nn.LayerNorm.html>

Any Questions?

???

Moving on

- RNN recap
- Language model evolution
- Motivations for attention design
- Dot-product attention
- Applying attention
- Transformer architecture
- Principal transformer variations

3 Types of Transformer Models

1. *Encoder* – transforms text embeddings into representations that support variety of tasks (e.g. sentiment analysis, classification)
 - ❖ Model Example: BERT
2. *Decoder* – predicts the next token to continue the input text (e.g. ChatGPT, AI assistants)
 - ❖ Model Example: GPT4o
3. *Encoder-Decoder* – used in sequence-to-sequence tasks, where one text string is converted to another (e.g. machine translation)

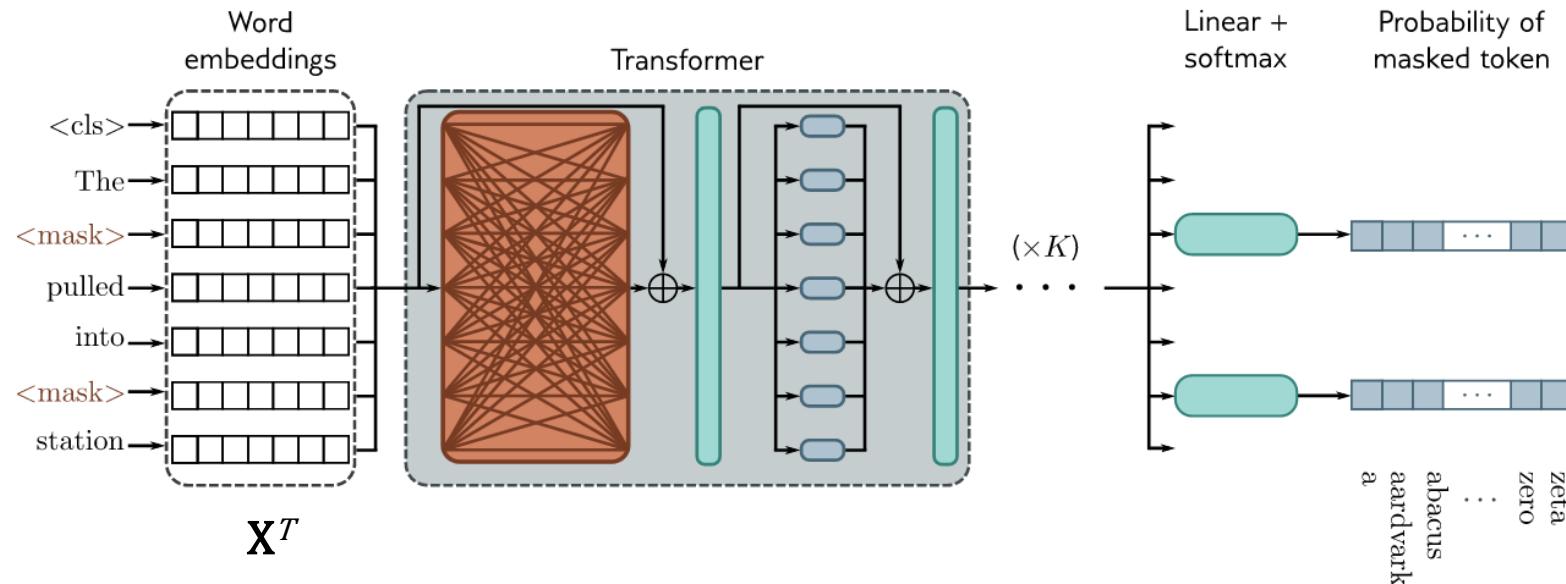
Encoder Model Example: BERT (2019)

Bidirectional Encoder Representations from Transformers

- Hyperparameters
 - 30,000 token vocabulary
 - 1024-dimensional word embeddings
 - 24x transformer layers
 - 16 heads in self-attention mechanism
 - 4096 hidden units in middle of MLP
- ~340 million parameters
- *Pre-trained* in a *self-supervised* manner,
- then can be adapted to task with one additional layer and *fine-tuned*

Encoder Pre-Training

Special <cls> token used for aggregate sequence representation for classification

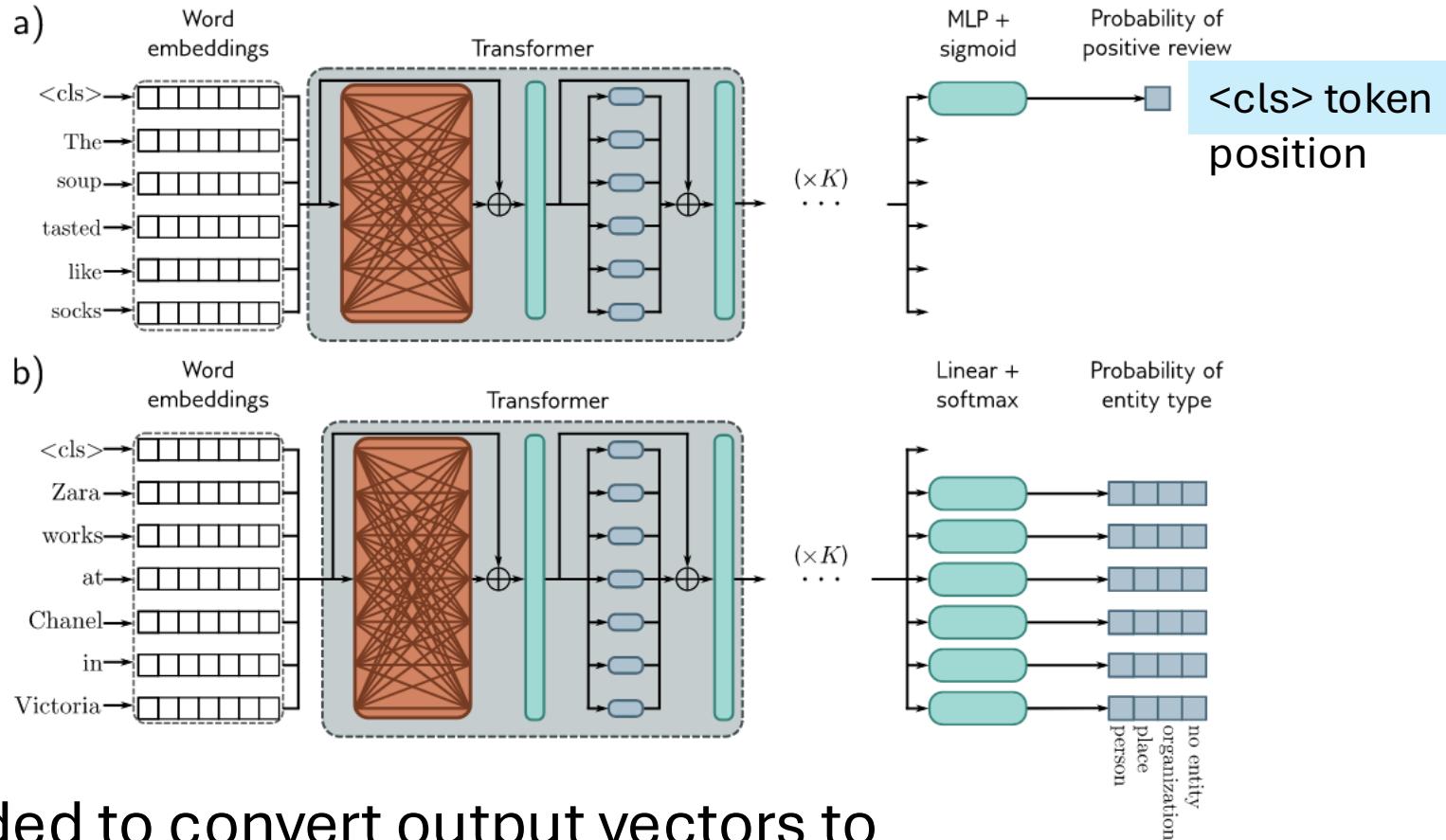


- A small percentage of input embedding replaced with a generic <mask> token
- Predict missing token from output embeddings
- Added linear layer and softmax to generate probabilities over vocabulary
- Trained on BooksCorpus (800M words) and English Wikipedia (2.5B words)

Encoder Fine-Tuning

Sentiment Analysis

Named Entity Recognition (NER)



- Extra layer(s) appended to convert output vectors to desired output format
- 3rd Example: Text span prediction -- predict start and end location of answer to a question in passage of Wikipedia, see <https://rajpurkar.github.io/SQuAD-explorer/>

Decoder Model Example: GPT3 (2020)

Generative Pre-trained Transformer

- One purpose: *generate the next token in a sequence*
- By constructing an autoregressive model

Decoder Model Example: GPT3 (2020)

Generative Pre-trained Transformer

- One purpose: *generate the next token in a sequence*
- By constructing an autoregressive model
- Factors the probability of the sentence:

$$\Pr(\text{Learning deep learning is fun}) = \\ \Pr(\text{Learning}) \times \Pr(\text{deep} \mid \text{learning}) \quad \times \\ \Pr(\text{learning} \mid \text{Learning deep}) \times \\ \Pr(\text{is} \mid \text{Learning deep learning}) \times \\ \Pr(\text{fun} \mid \text{Learning deep learning is})$$

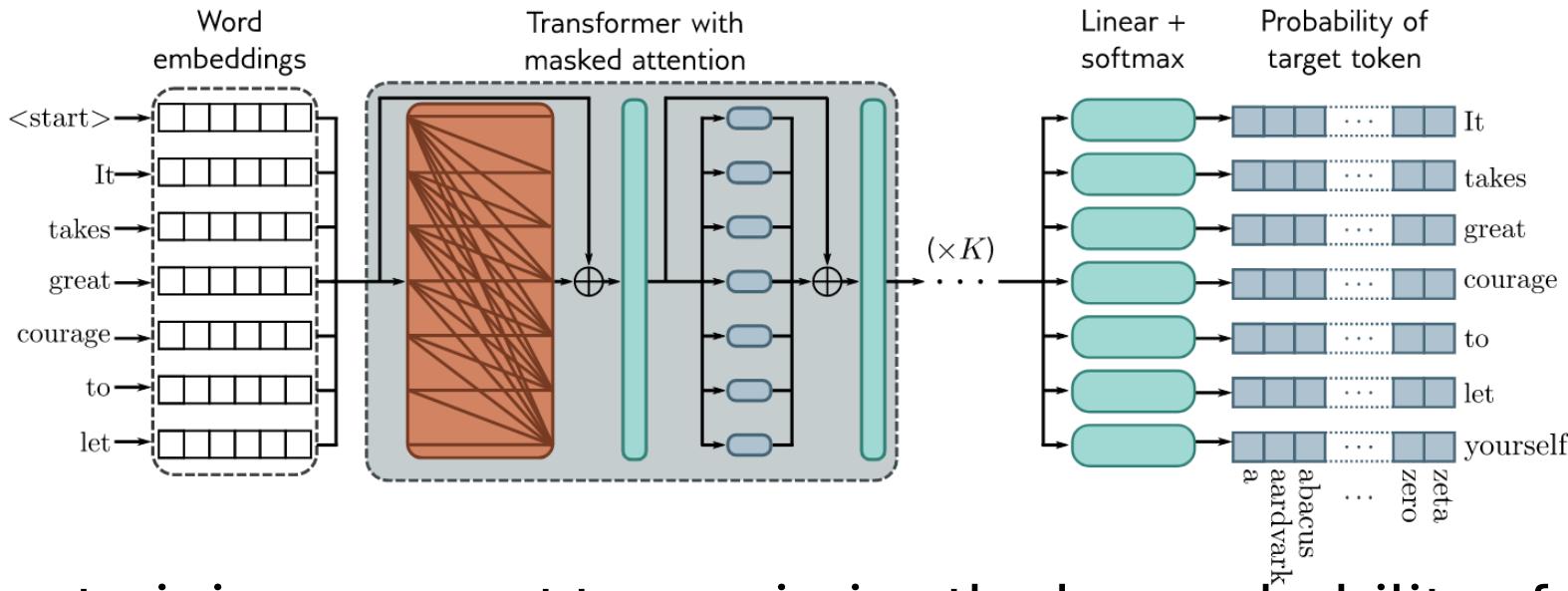
Decoder Model Example: GPT3 (2020)

Generative Pre-trained Transformer

- One purpose: *generate the next token in a sequence*
- By constructing an autoregressive model
- Factors the probability of the sentence:
 $\Pr(\text{Learning deep learning is fun}) =$
 $\Pr(\text{Learning}) \times \Pr(\text{deep} \mid \text{learning}) \times$
 $\Pr(\text{learning} \mid \text{Learning deep}) \times$
 $\Pr(\text{is} \mid \text{Learning deep learning}) \times$
 $\Pr(\text{fun} \mid \text{Learning deep learning is})$
- More formally: Autoregressive model

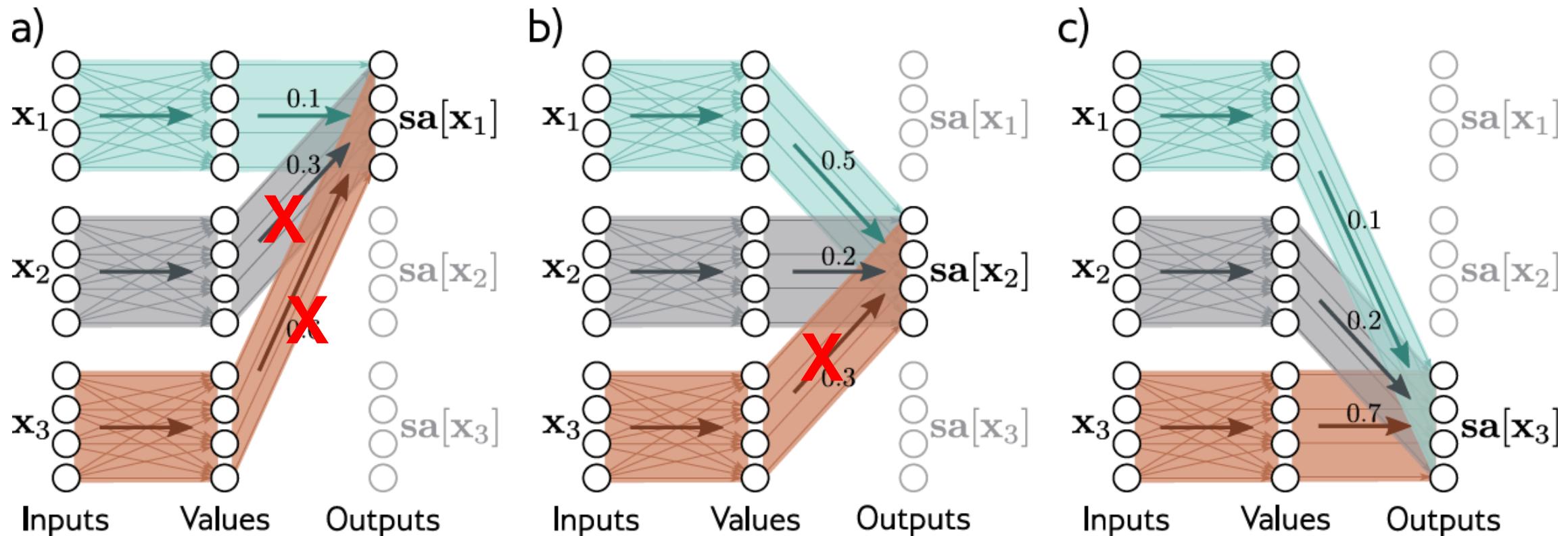
$$\Pr(t_1, t_2, \dots, t_N) = \Pr(t_1) \prod_{n=2}^N \Pr(t_n \mid t_1, t_2, \dots, t_{n-1})$$

Decoder: Masked Self-Attention



- During training we want to maximize the log probability of the input text under the autoregressive model.
- We want to make sure the model doesn't "cheat" during training by looking ahead at the next token.
- Hence, we mask the self attention weights corresponding to current and right context to *negative infinity*.

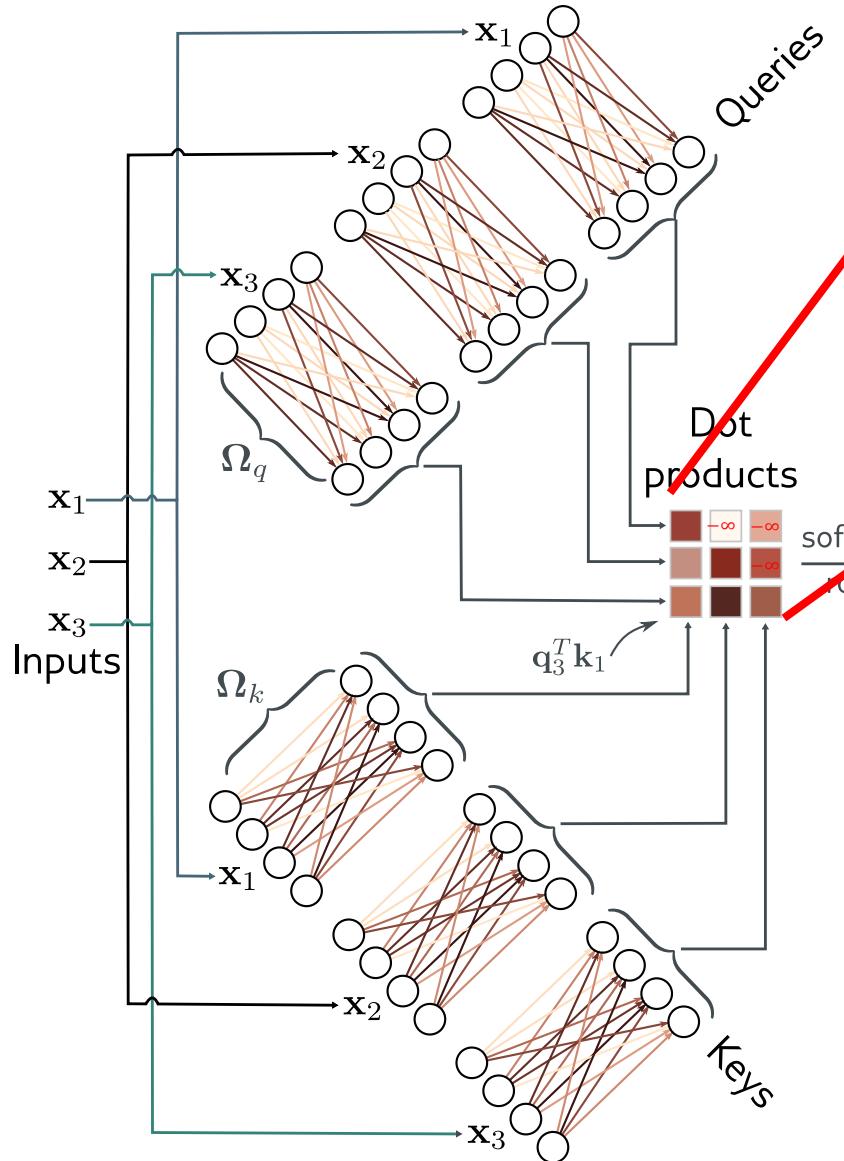
Masked Self-Attention



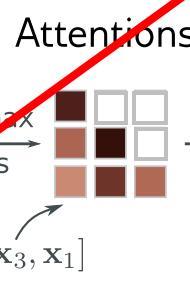
Mask right context self-attention weights to zero

Masked Self-Attention

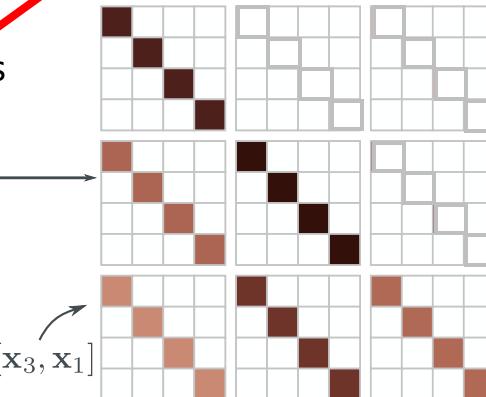
a)



b)



Dot products
Attentions

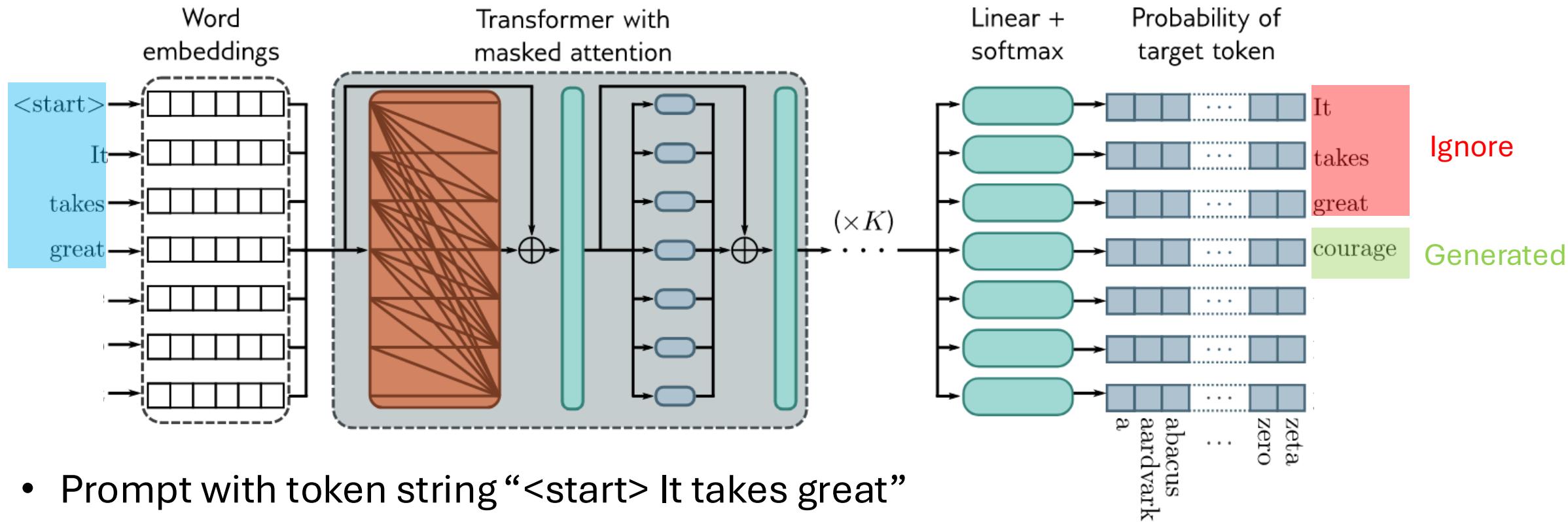


c)

Attention weights

Decoder: Text Generation (Generative AI)

Prompt



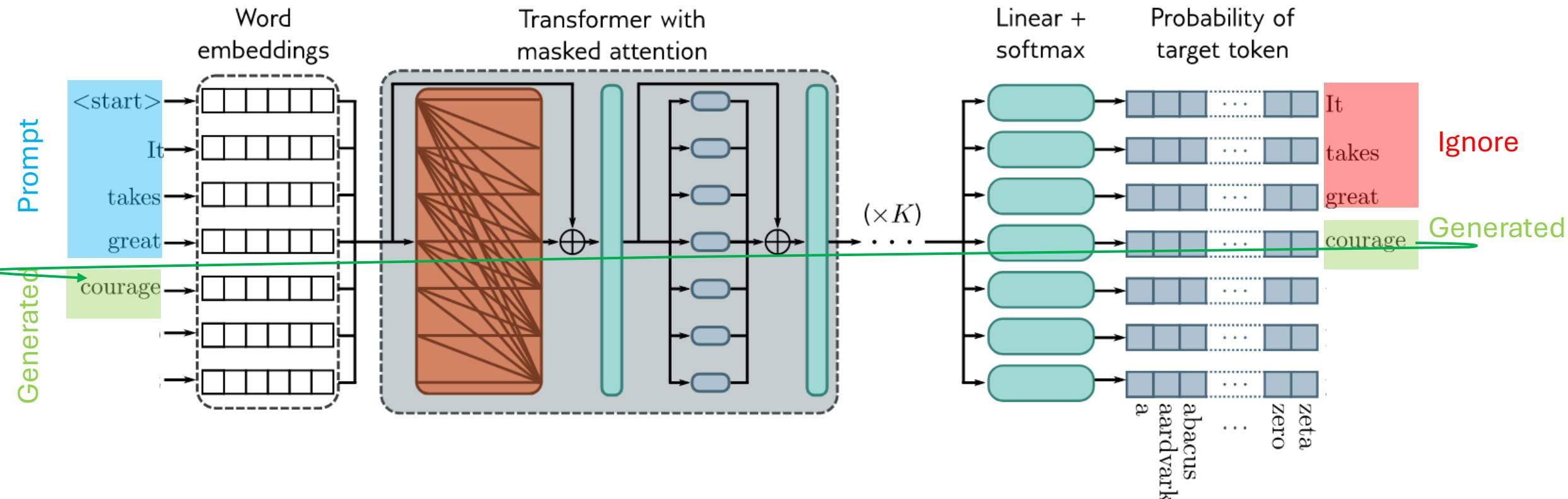
- Prompt with token string “<start> It takes great”
- Generate next token for the sequence by
 - picking most likely token
 - sample from the probability distribution
 - alternative *top-k* sampling to avoid picking from the long tail
 - beam search – select the most likely sentence rather than greedily pick

Dummy's Guide to LLM Sampling

- <https://rentry.co/samplers>
- Will talk about this more next time.

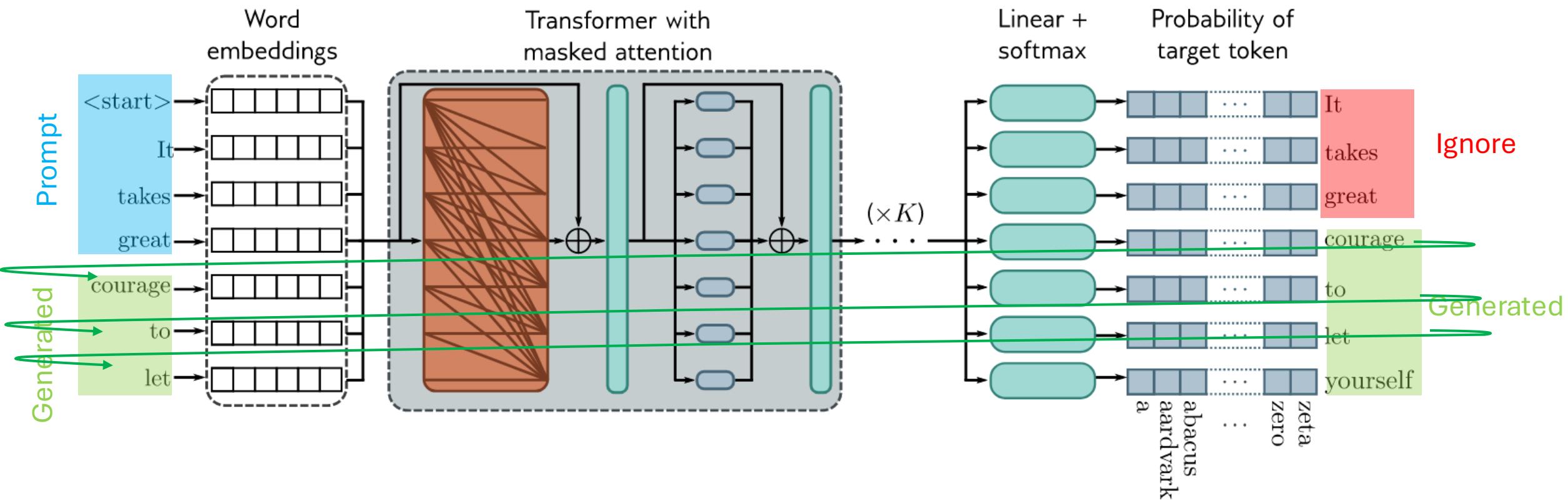
Dummy's Guide to Modern LLM Sampling	
1. Intro Knowledge	
1. Short Glossary	
2. Why tokens?	
1. Why not letters?	
2. Why not whole words?	
3. How are the sub-words chosen?	
4. How does the model generate text?	
2. Sampling	
1. Notes on Algorithm Presentations	
1. Notation Guide	
2. Implementation Considerations	
2. Temperature	
3. Presence Penalty	
4. Frequency Penalty	
5. Repetition Penalty	
6. DRY (Don't Repeat Yourself)	
7. Top-K	
8. Top-P	
9. Min-P	
10. Top-A	
11. XTC (eXclude Top Choices)	
12. Top-N-Sigma	
13. Tail-Free Sampling	
14. Eta Cutoff	
15. Epsilon Cutoff	
16. Locally Typical Sampling	
17. Quadratic Sampling	
18. Mirostat Sampling	
19. Dynamic Temperature Sampling	
20. Beam Search	
21. Contrastive Search	
3. Sampler Order	
1. The Typical Sampling Pipeline	
2. Effects and Interactions of Samplers with Each Other	
1. How Samplers Transform Distributions	
2. Critical Order-Dependent Interactions	
1. Temperature Before vs. After Filtering	
2. Penalties Before vs. After Other Samplers	
3. DRY's Position Matters	
3. Synergies and Conflicts	
1. Synergistic Combos	
2. Conflicting Combos	
4. Advanced: More on Tokenizers	
1. Building the Vocabulary	
1. Byte Pair Encoding (BPE)	
2. SentencePiece	
3. Vocabulary Size and Granularity	
4. Token Boundaries and Phrasing	
5. Handling Rare or New Words	

Decoder: Text Generation (Generative AI)



- Feed the output back into input

Decoder: Text Generation (Generative AI)



Technical Details

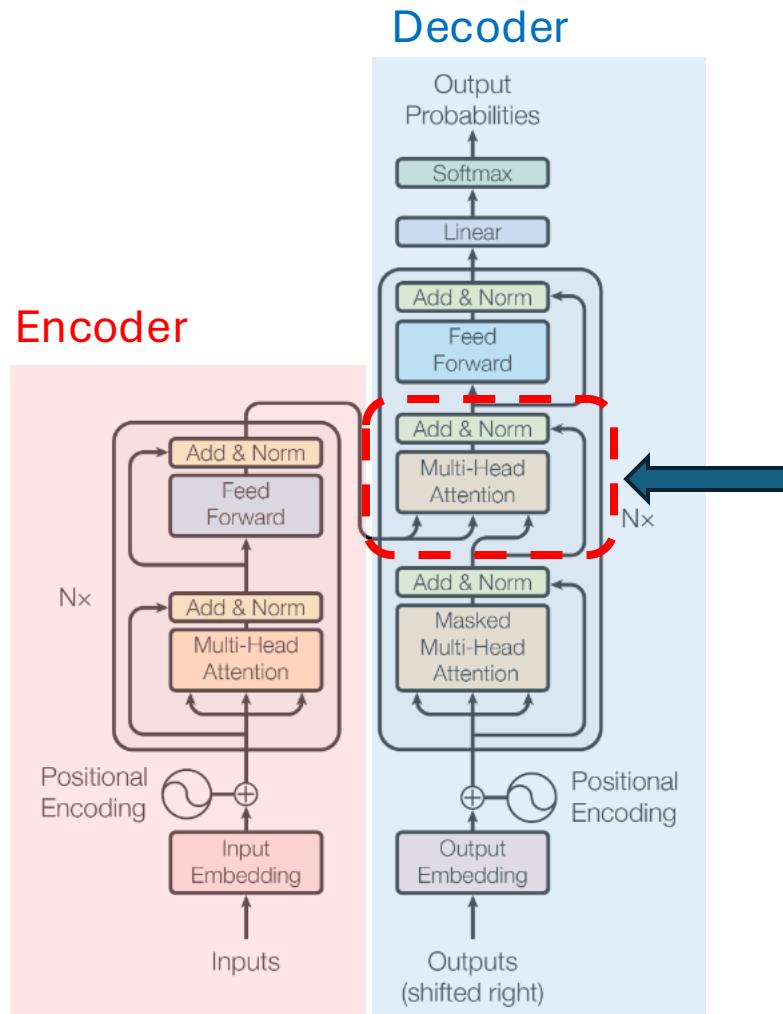
	BERT	GPT3
Model Architecture	Encoder	Decoder
Embedding Size	1024	12,288
Vocabulary	30K tokens	
Sequence Length		2048
# Heads	16	96
# Layers	24	96
Q,K,V dimensions	64	128
Training set size	3.3B tokens	300B+ tokens
# Parameters	340M	175B

Encoder-Decoder Model

- Used for *machine translation*, which is a *sequence-to-sequence* task

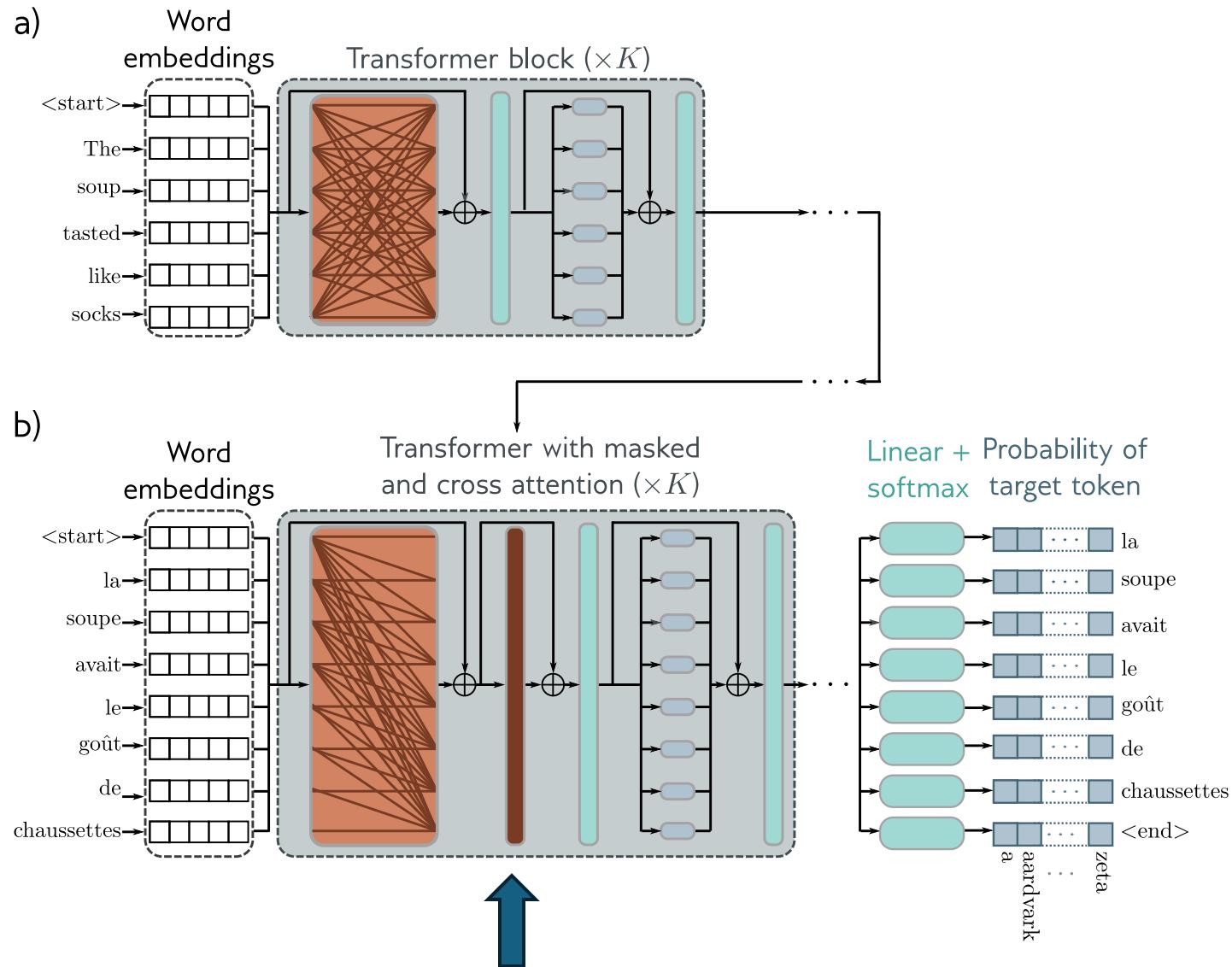


Encoder Decoder Model



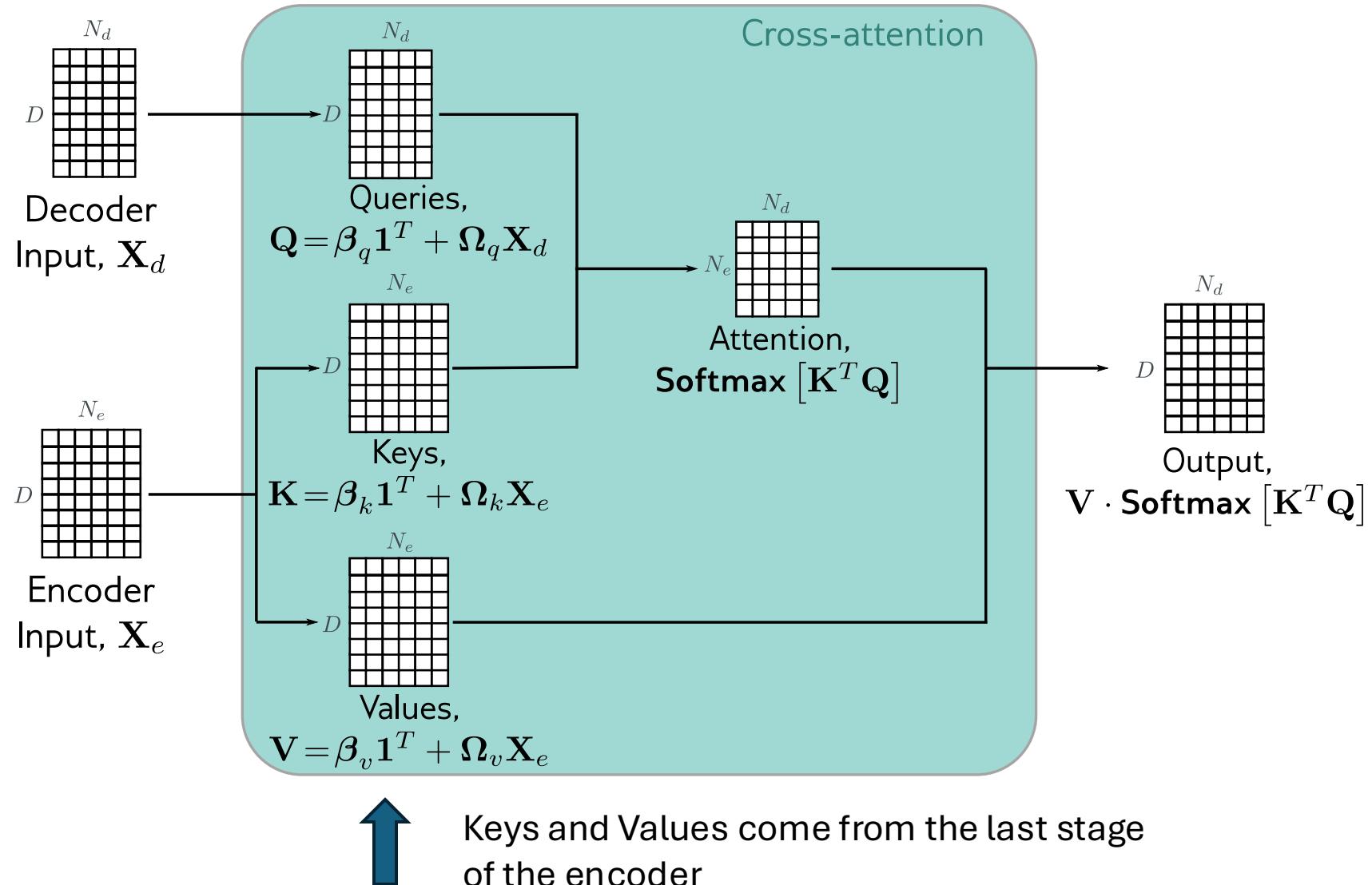
- The transformer layer in the decoder of the encoder-decoder model has an extra stage
- Attends to the input of the encoder with *cross attention* using Keys and Values from the output of the encoder
- Shown here on original diagram from “Attention is all you need” paper

Encoder Decoder Model



- Same view per UDL book

Cross-Attention



Any Questions?

???

- RNN recap
- Language model evolution
- Motivations for attention design
- Dot-product attention
- Applying attention
- Transformer architecture
- Principal transformer variations