Rerelease AITA data

CS 333: Natural Language Processing

Fall 2025

Prof. Carolyn Anderson
Wellesley College

Reminders

- Tuesday -> Monday cycle for final two assignments:
  - HW 7 is due on Monday, 11/17
  - HW 8 will be released on Tuesday, 11/18 and due on Monday, 11/24
- My next help hours: Monday 4-5:30

WELLESLEY CS COLLOQUIUM

Professor Lindsey D. Cameron

Wharton School, University of Pennsylvania

Resocializing the Platform: Patchwork Embeddedness and How Workers ReConstitute Digital Platforms

14 NOV 2025 | 3:30 PM | SCI-H105

Snacks will be provided!

WELLESLEY
COMPUTER SCIENCE

?sb129@wellesley.edu
Accessibility and Disability:
accessibility@wellesley.edu

Two ML talks for the price of one:

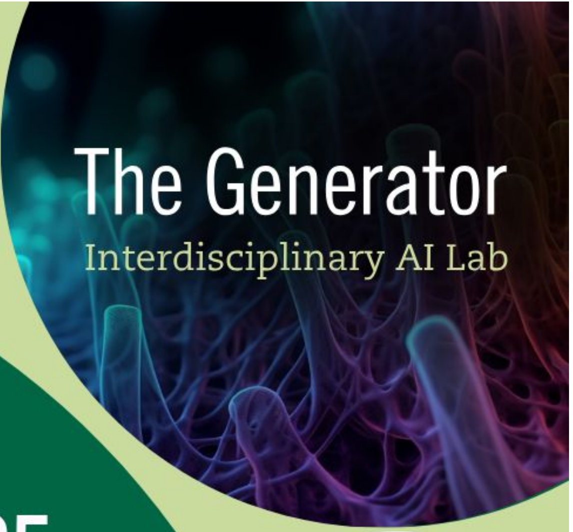
Co-designing Tools to Measure Student Learning with Machine Learning and Science Education Research

Dr. Kaitlin Gili

Subgroup Validity in Machine Learning for Echocardiogram Data

Cynthia Feeney

Thursday Nov. 20 at 12:45-1pm in H-105



Babson Biotech Club

The Generator Buildathon Fall 2025

Build Real-World Solutions with AI

Over $5,000 in prizes! Food & fun!

No Experience Necessary*

Sat. Nov. 15th | 9am - 8pm

Weissman Foundry

Microsoft

Lovable

Northeastern University

Harvard

Biotech Innovation Discovery

Entrepreneurial Applications for Biotechnologies

Global Access & Equity Solutions

Learn More & Register Today!

BABSON COLLEGE

Recap

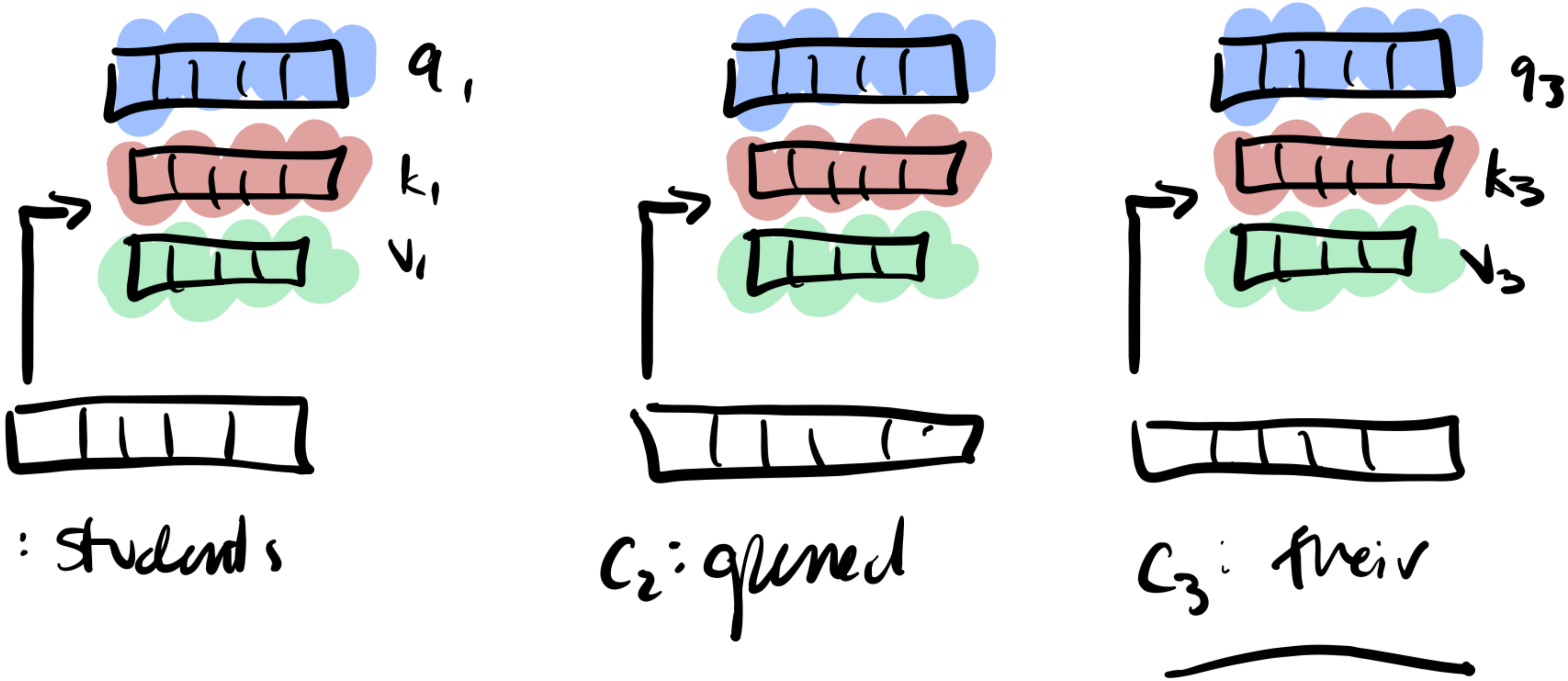
Self-Attention

Motivation: efficiency by parallelization

$q_1 = f(W_q c_1)$ $k_1 = f(W_k c_1)$ $v_1 = f(W_v c_1)$

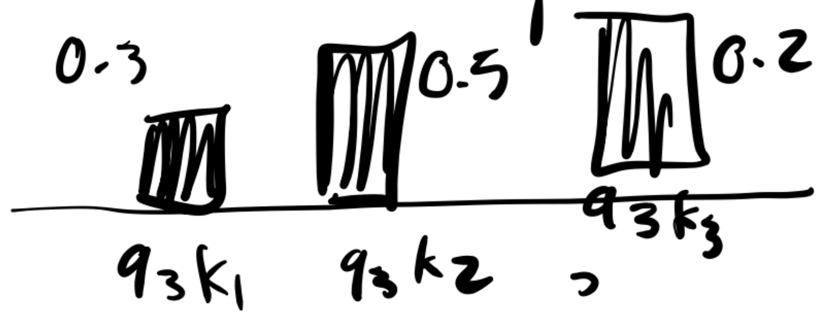
1. Take the dot product between q_3 & every k

$$\langle q_3 k_1 \quad q_3 k_2 \quad \underline{q_3 k_3} \rangle$$

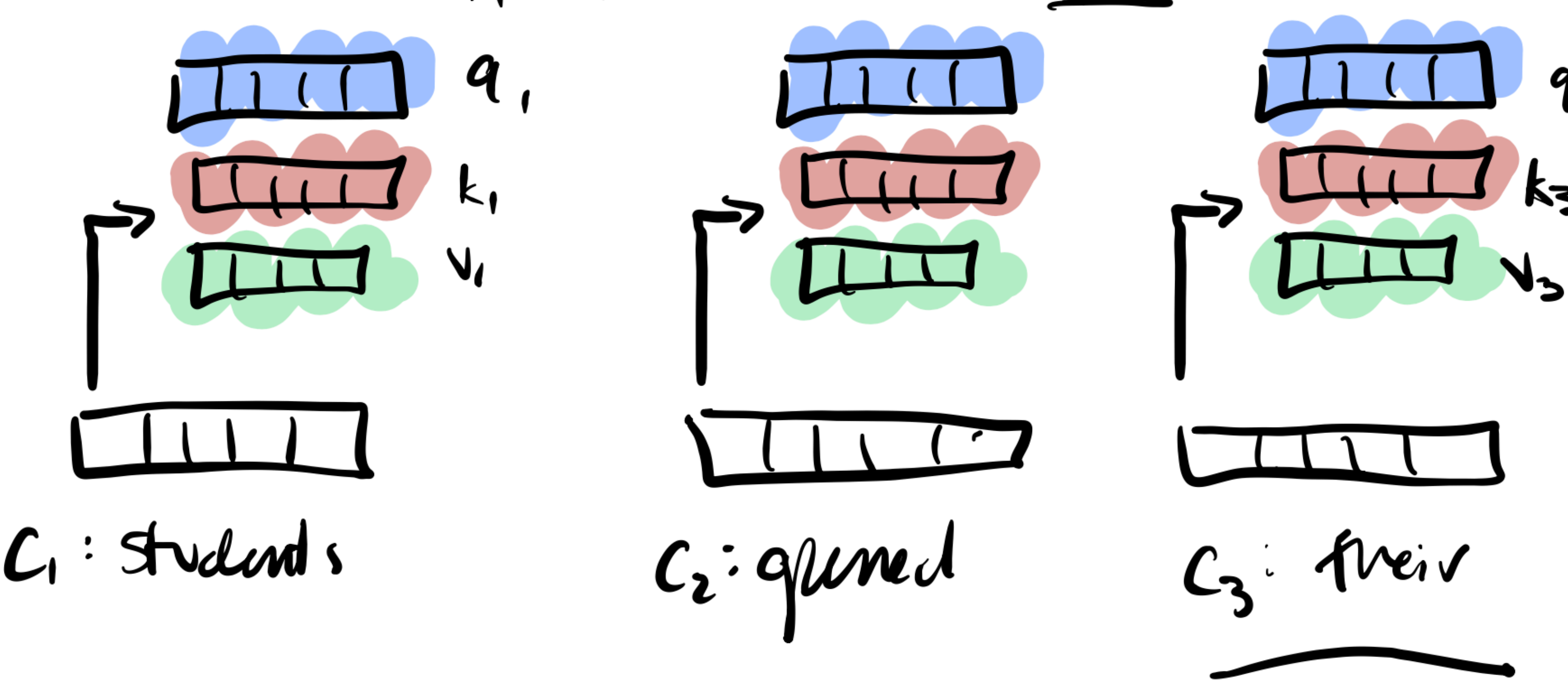


Self-Attention Motivation: efficiency & parallelization

Step 2): Softmax to get a distribution



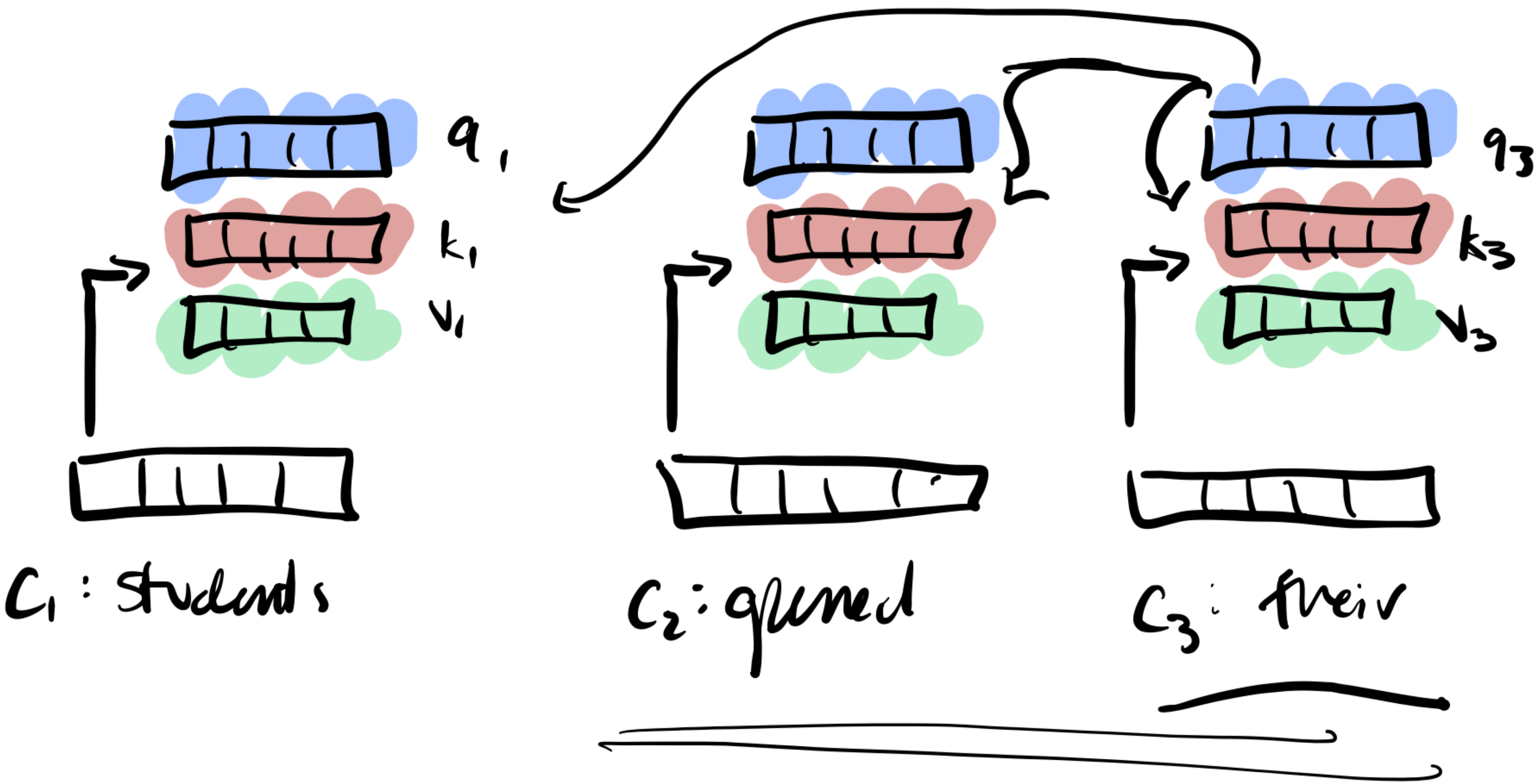
$\langle 0.3\ 0.5\ 0.2 \rangle$



Self-Attention Motivation: efficiency & parallelization

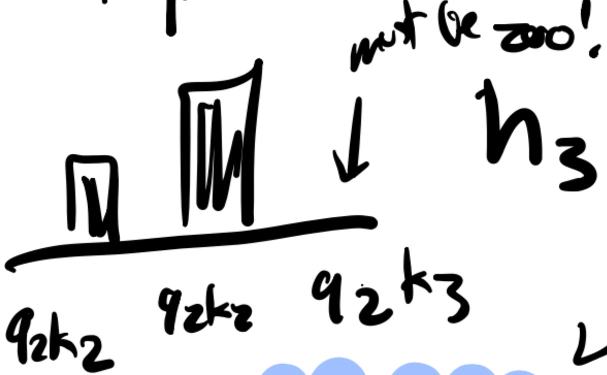
step 3): Calculate weighted average on values.

$$h_3 = 0.3v_1 + 0.5v_2 + 0.2v_3$$

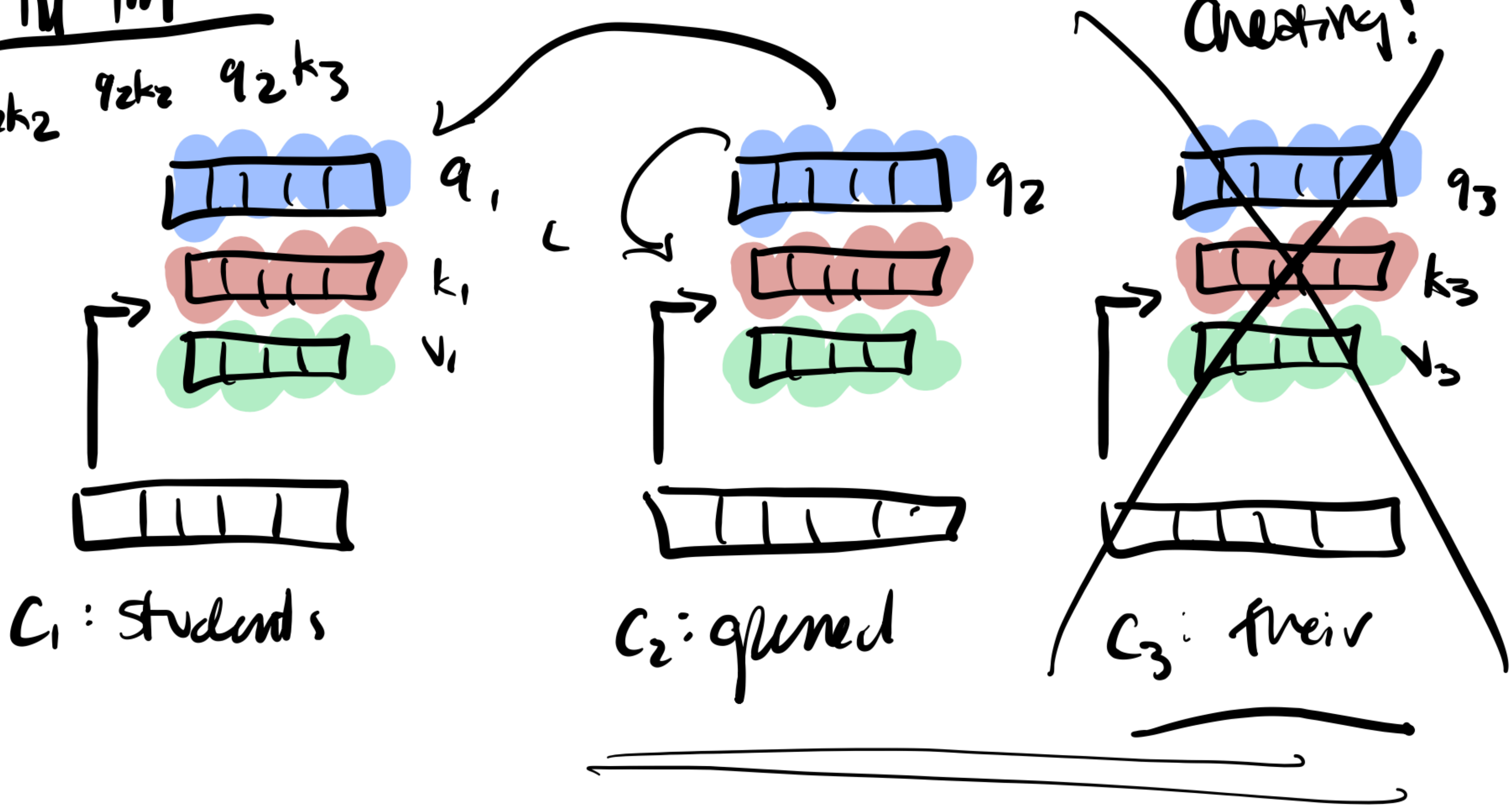


Self-Attention Motivation: efficiency & parallelization

step 3): Calculate weighted average on values.



$$h_3 = 0.3v_1 + 0.5v_2 + 0.2v_3$$



Parallelizing Self-Attention

Where is the word position information?

$a_1 = \langle q_1 k_1 \rangle$

$a_2 = \langle q_2 k_1, q_2 k_2 \rangle$

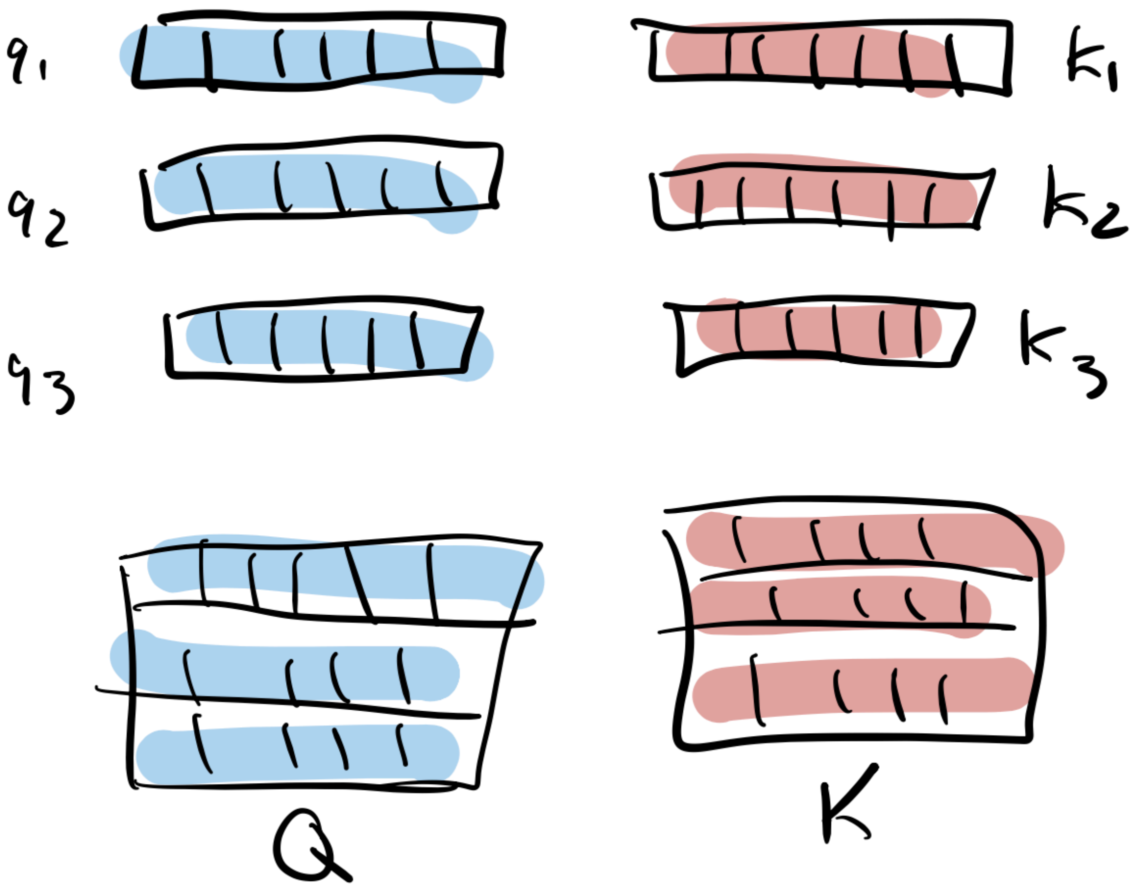
$a_3 = \langle q_3 k_1, q_3 k_2, q_3 k_3 \rangle$

$h_1 : v_1$

$h_2 : v_1, v_2$

$h_3 : v_1, v_2, v_3$

Solution: a MASK matrix that masks some attention scores by multiplying w/ $-\infty$



Transpose

$$\begin{array}{c|ccc} & k_1 & k_2 & k_3 \\ \hline q_1 & & \times & \times \\ q_2 & & & \times \\ q_3 & & & \end{array}$$

MASK $\quad$ Post-softmax

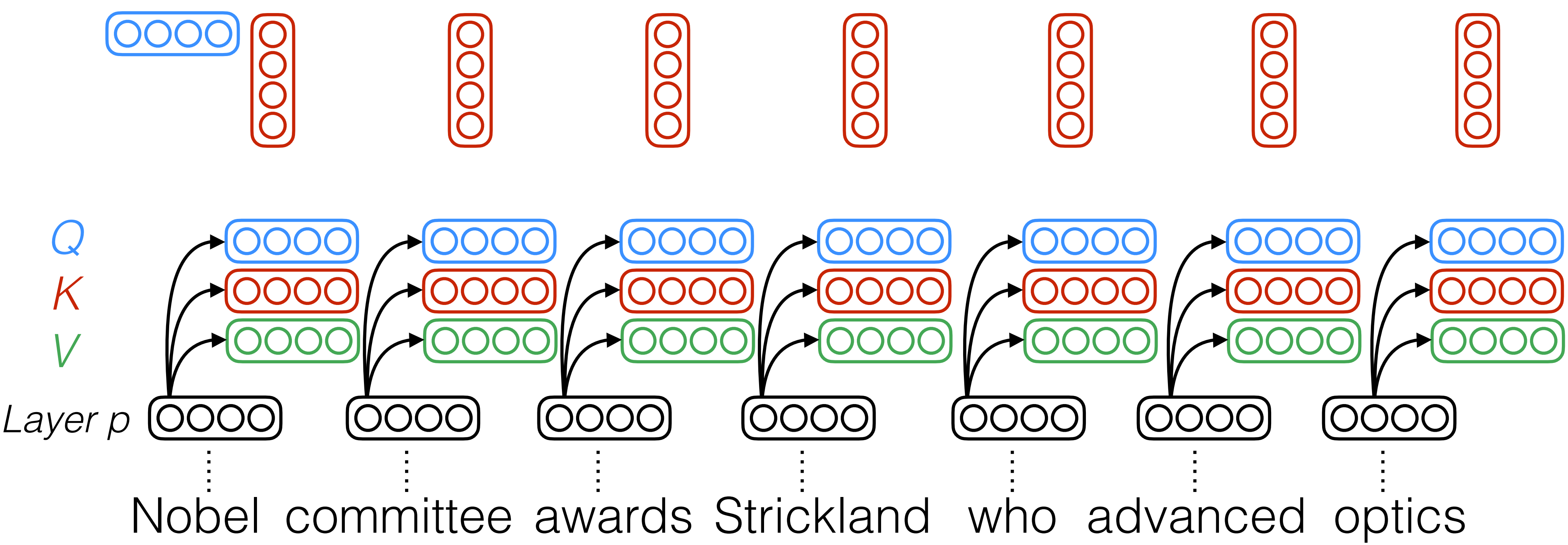
$$\begin{bmatrix} 1 & -\infty & -\infty \\ 1 & 1 & -\infty \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \cdot & 0 & 0 \\ \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot \end{bmatrix}$$

Scaling Up Self-Attention

Slides by Emma Strubell!

[Vaswani et al. 2017]

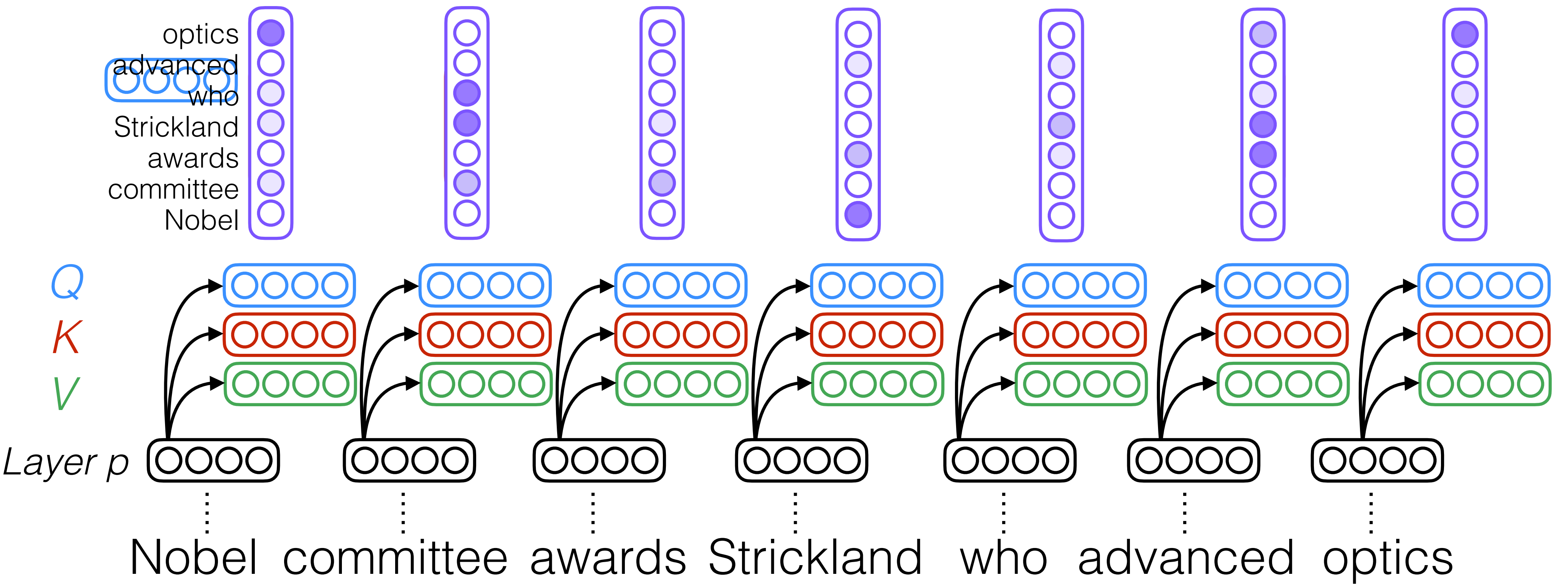
Self-attention



Slides by Emma Strubell!

[Vaswani et al. 2017]

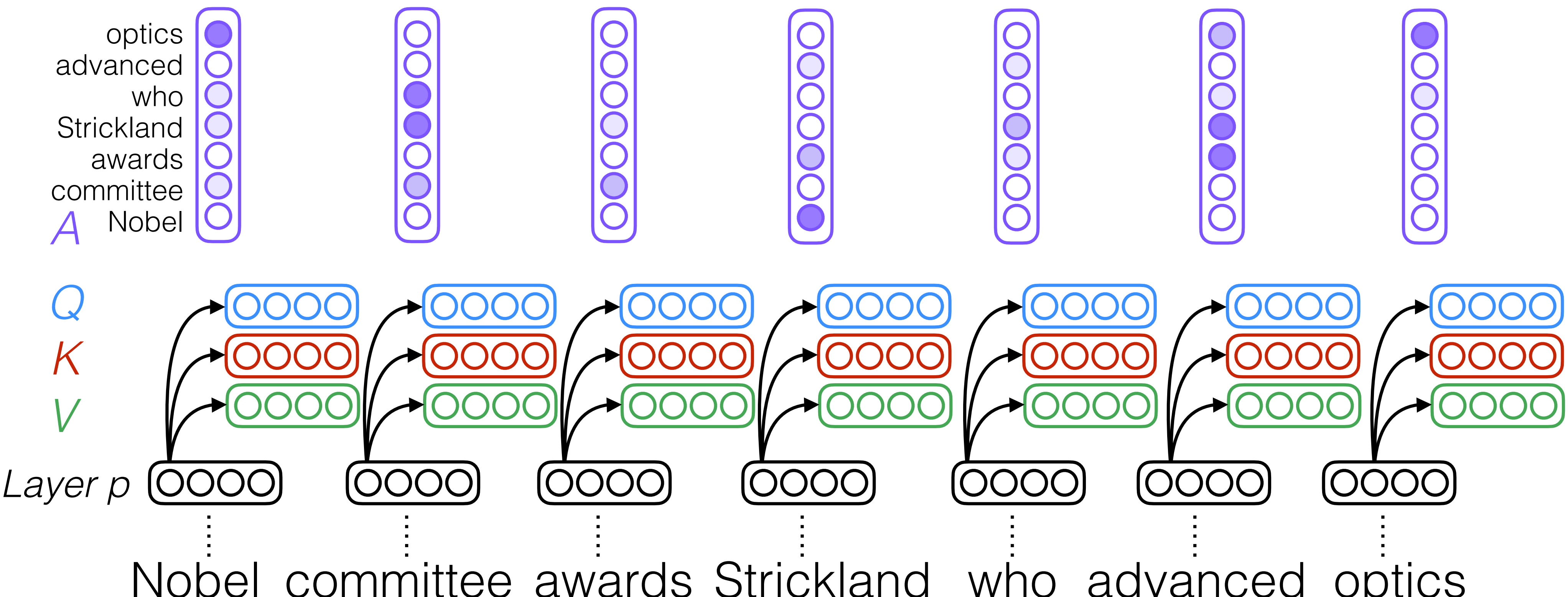
Self-attention



Slides by Emma Strubell!

[Vaswani et al. 2017]

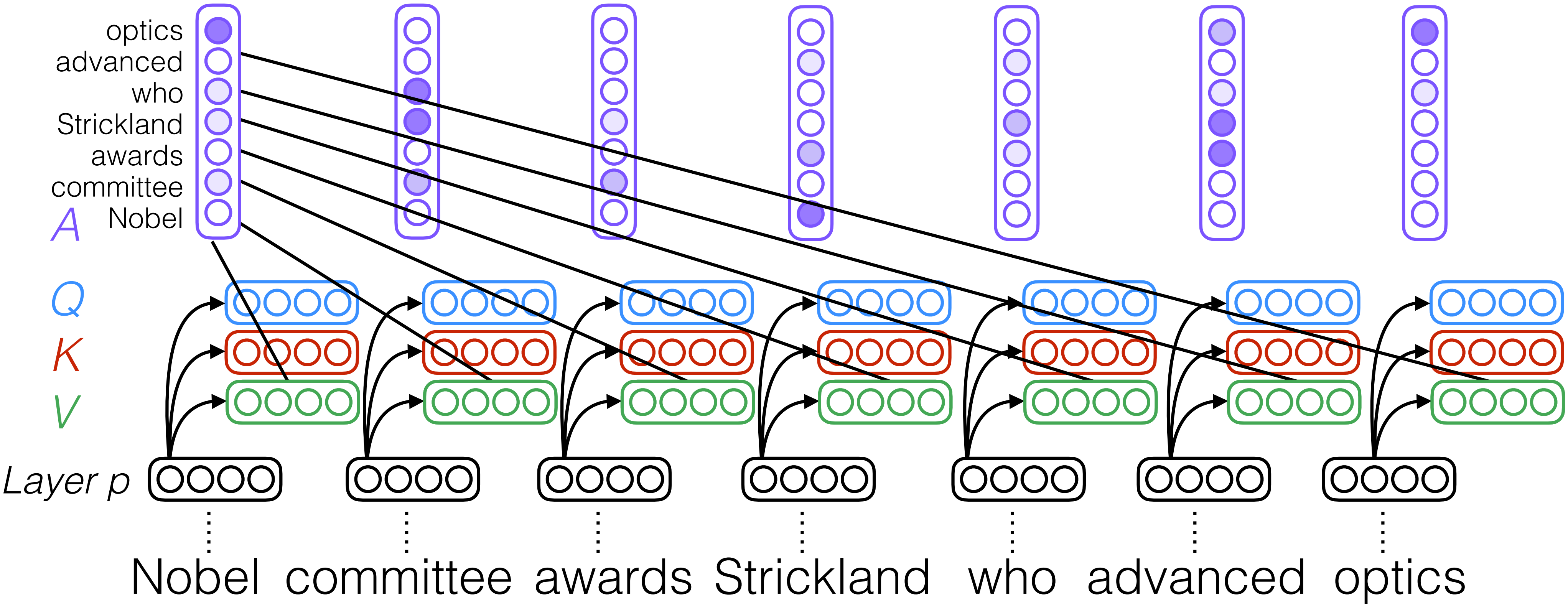
Self-attention



Slides by Emma Strubell!

[Vaswani et al. 2017]

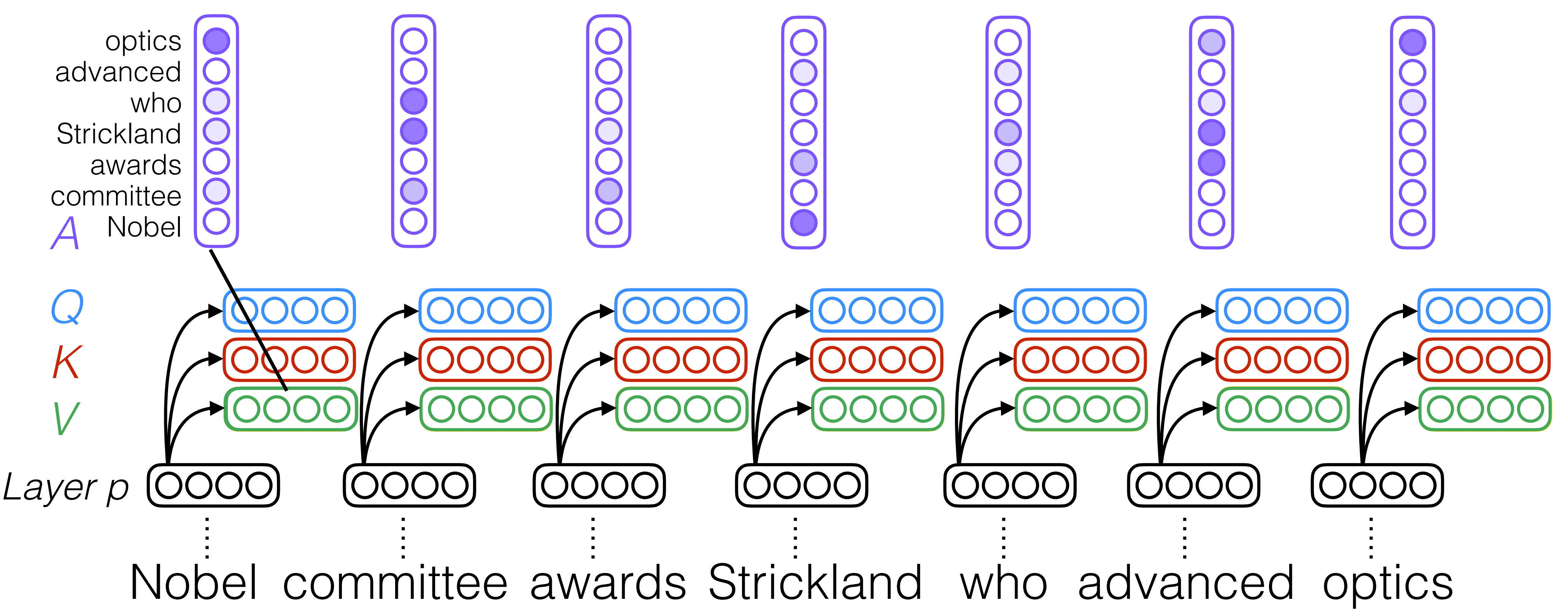
Self-attention



Slides by Emma Strubell!

[Vaswani et al. 2017]

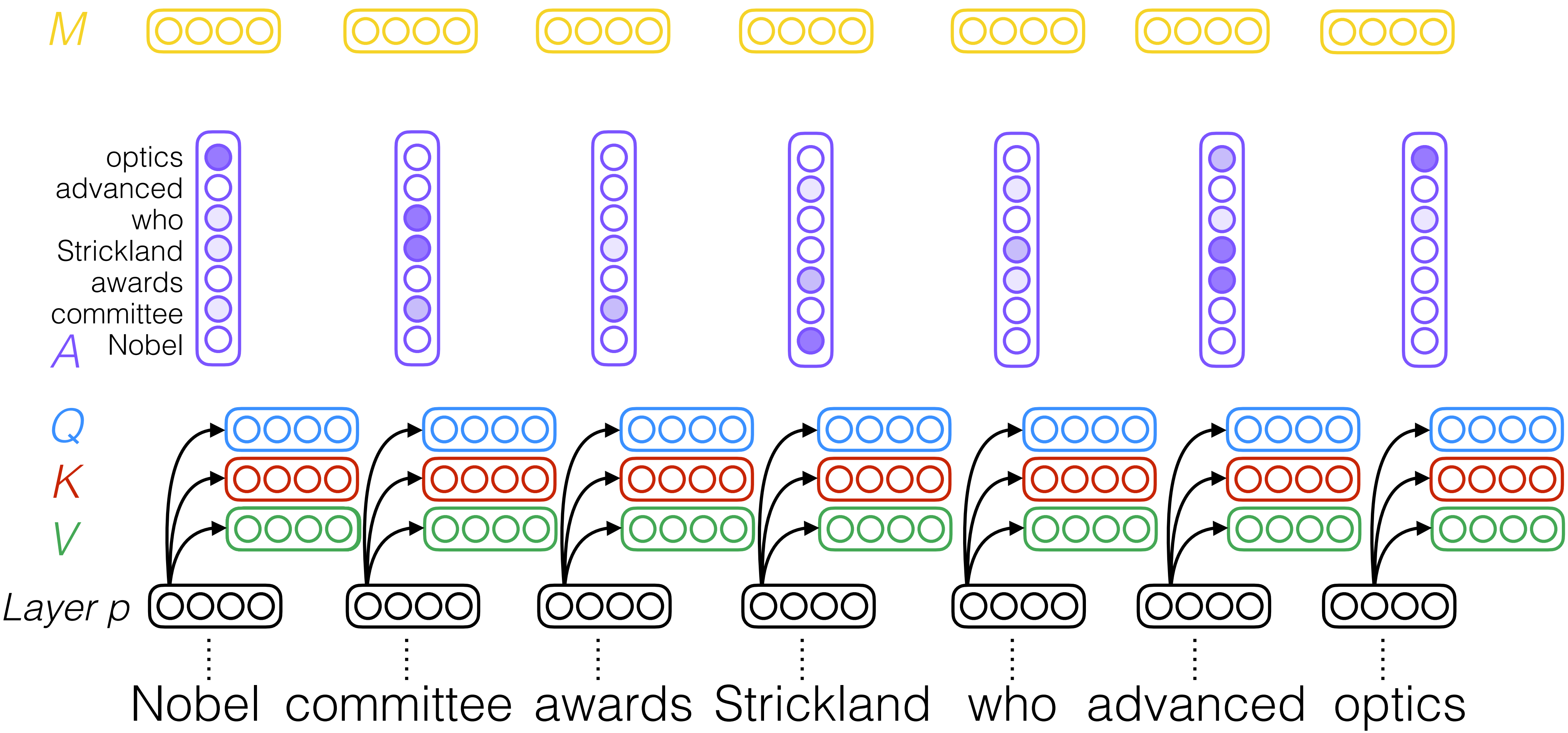
Self-attention



Slides by Emma Strubell!

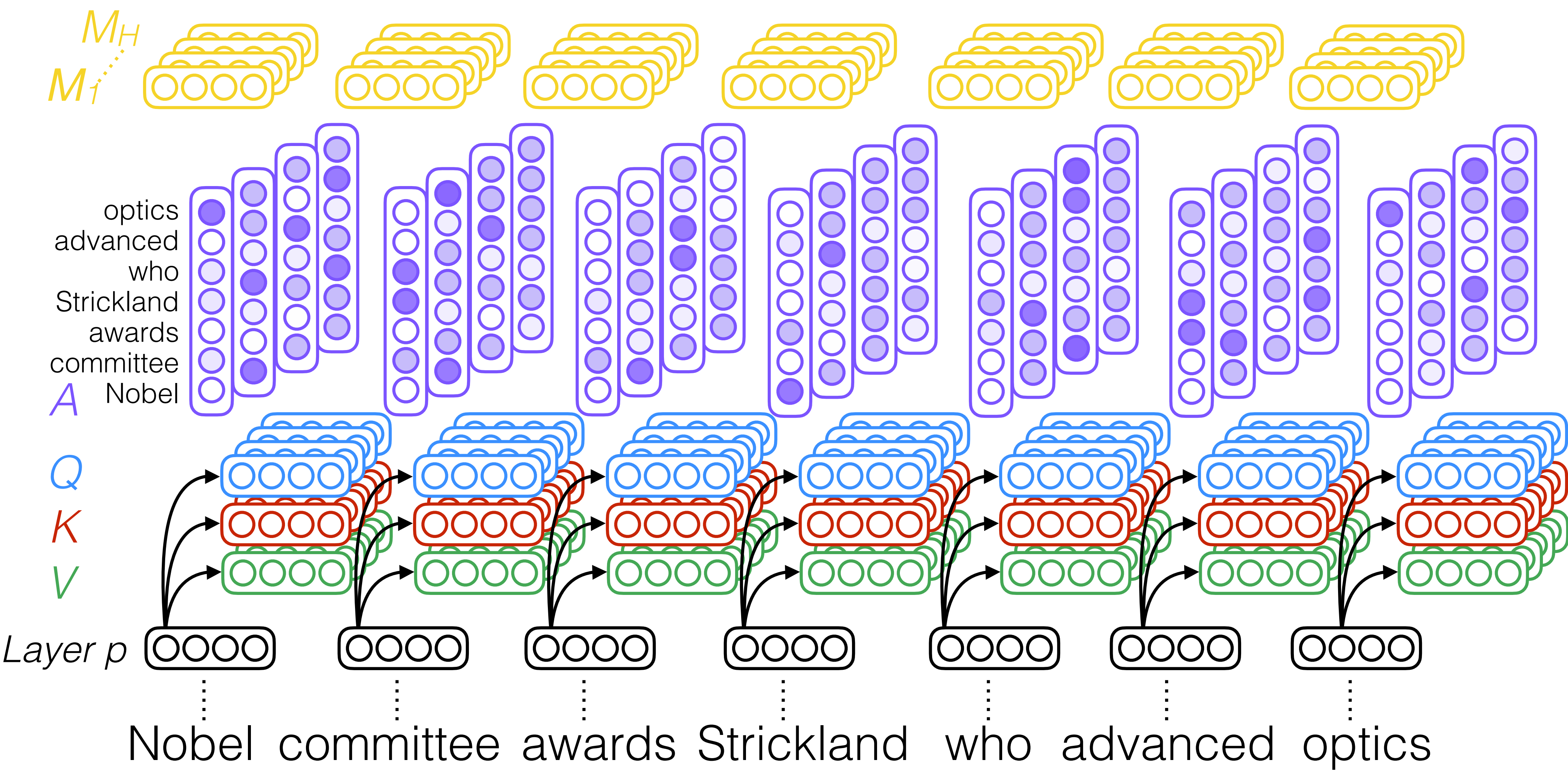
[Vaswani et al. 2017]

Self-attention



Slides by Emma Strubell! [Vaswani et al. 2017]

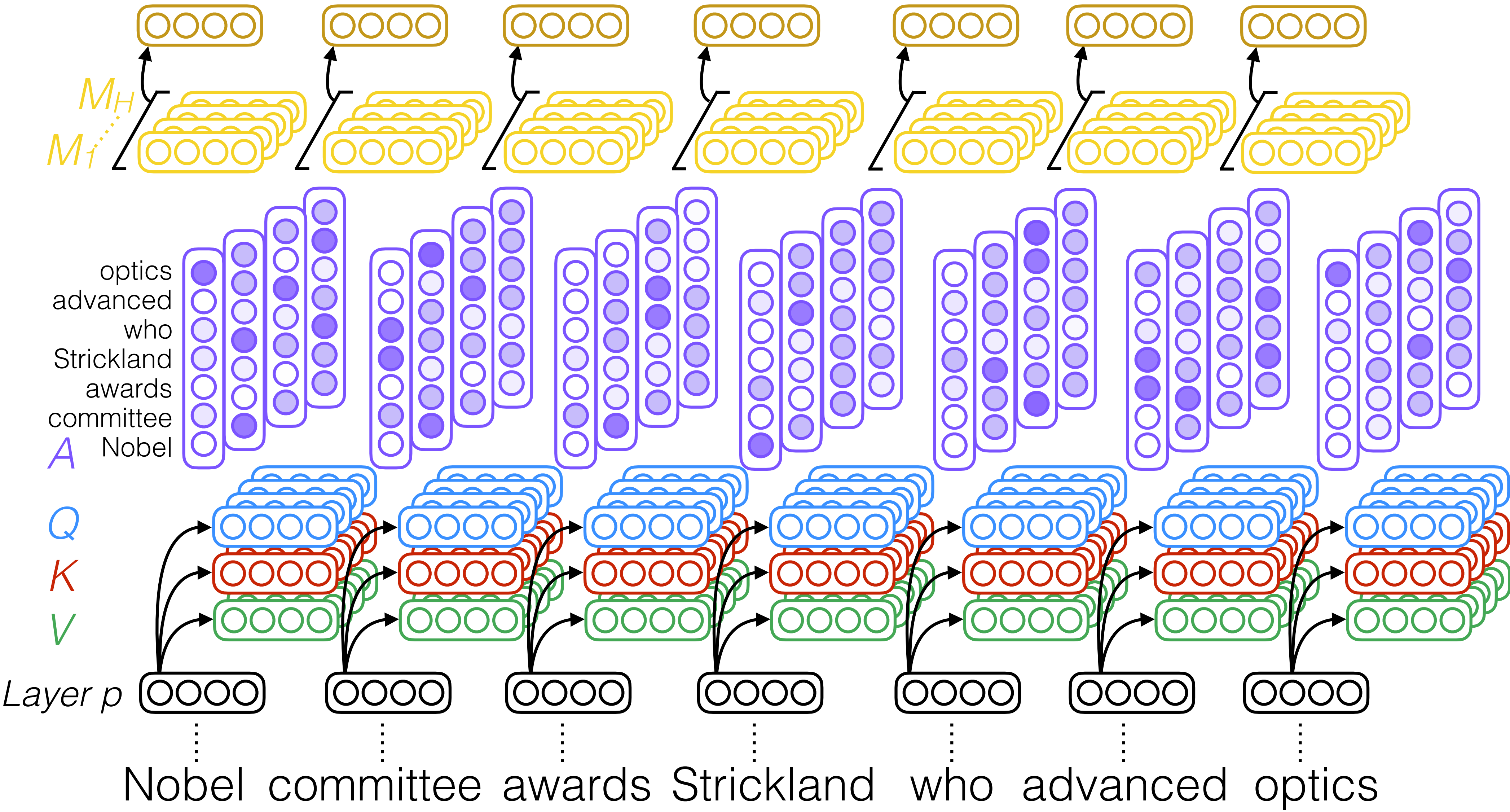
Multi-head self-attention



Slides by Emma Strubell!

[Vaswani et al. 2017]

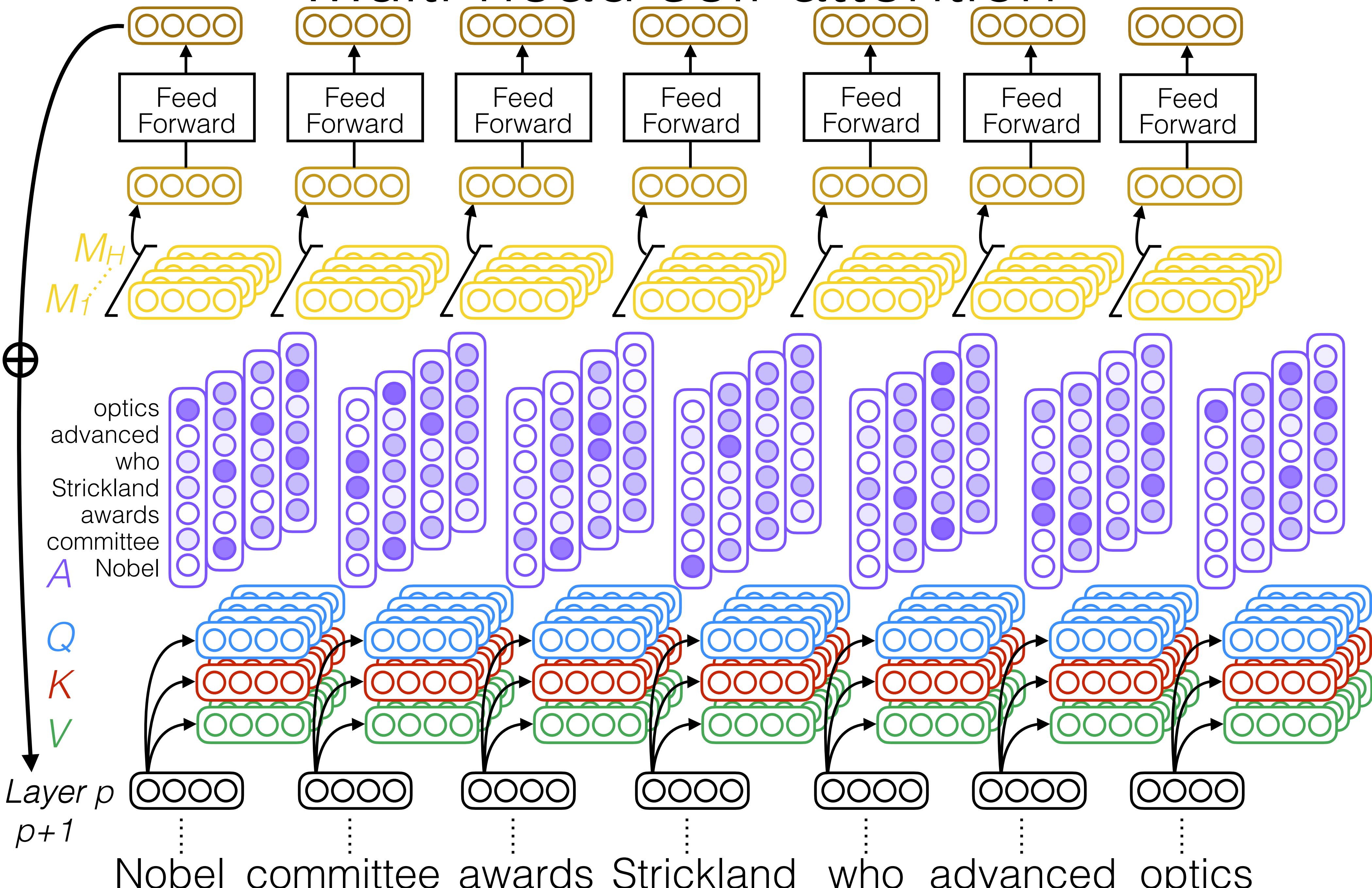
Multi-head self-attention



Slides by Emma Strubell!

[Vaswani et al. 2017]

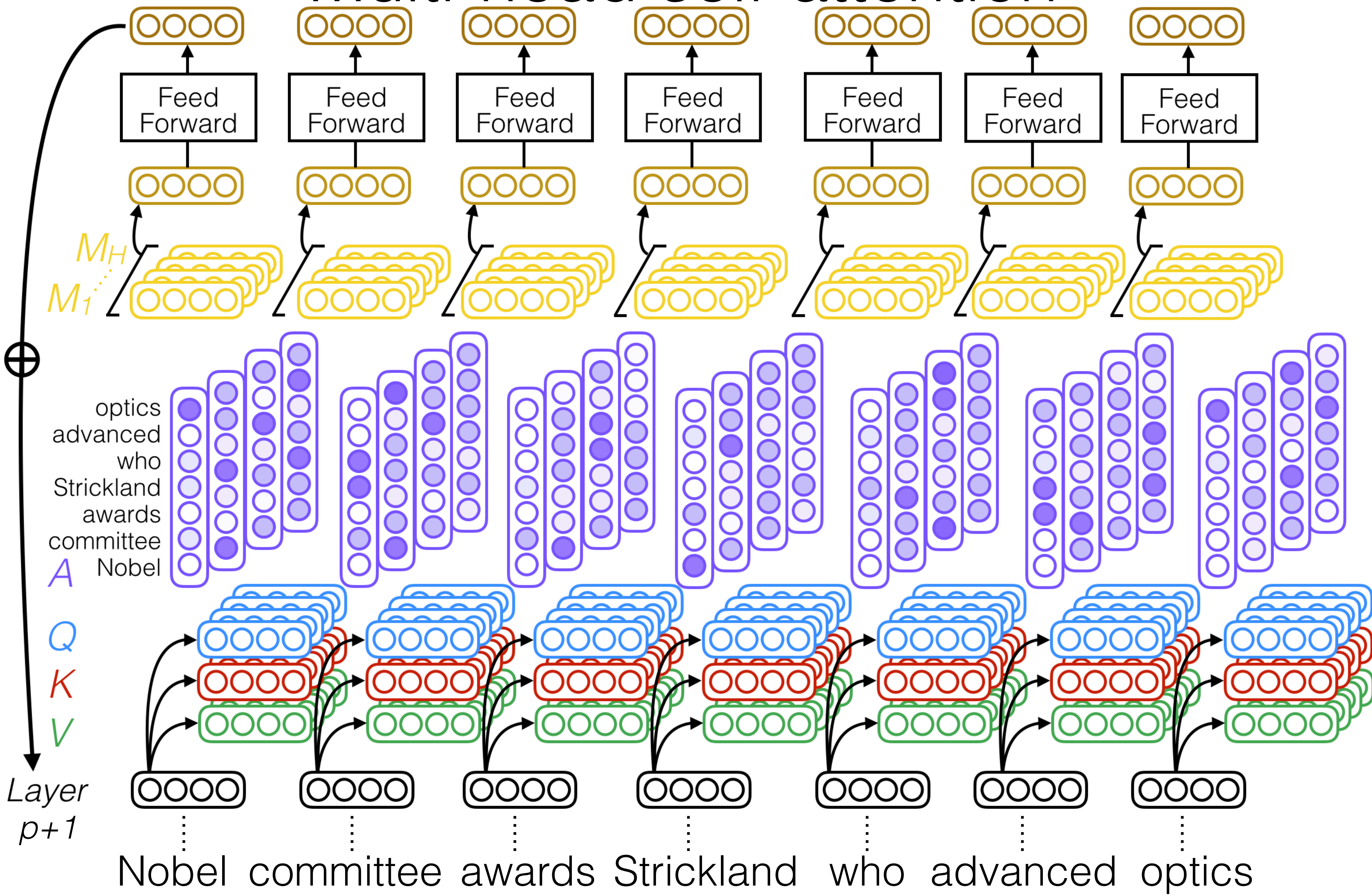
Multi-head self-attention



Slides by Emma Strubell!

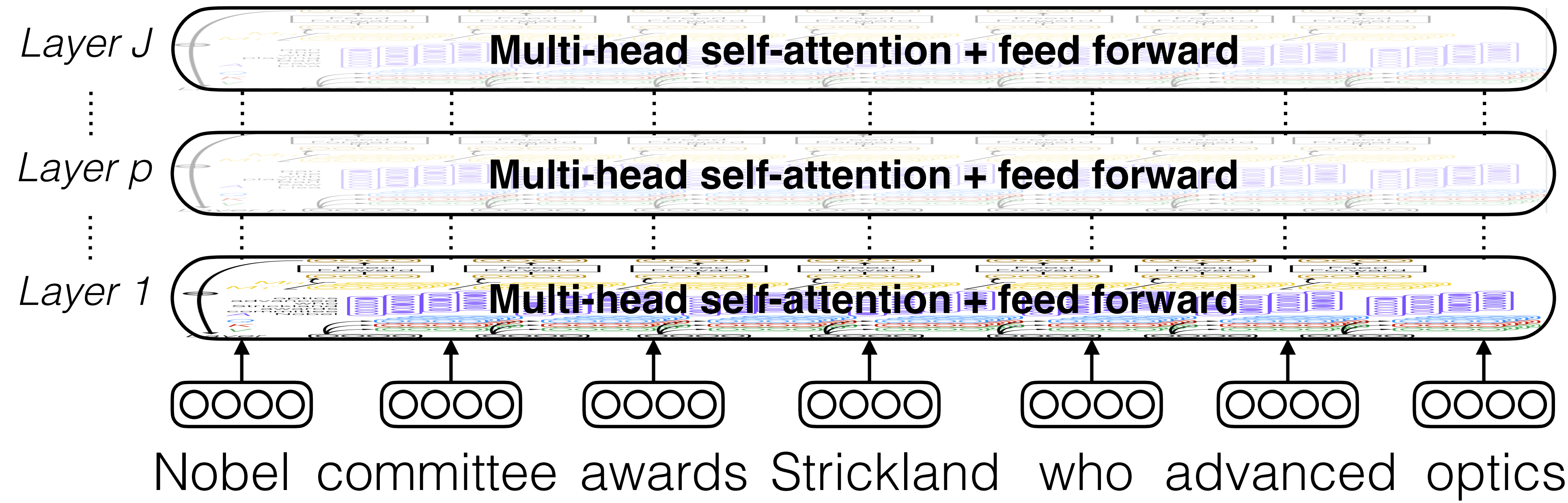
[Vaswani et al. 2017]

Multi-head self-attention



[Vaswani et al. 2017]

Multi-head self-attention



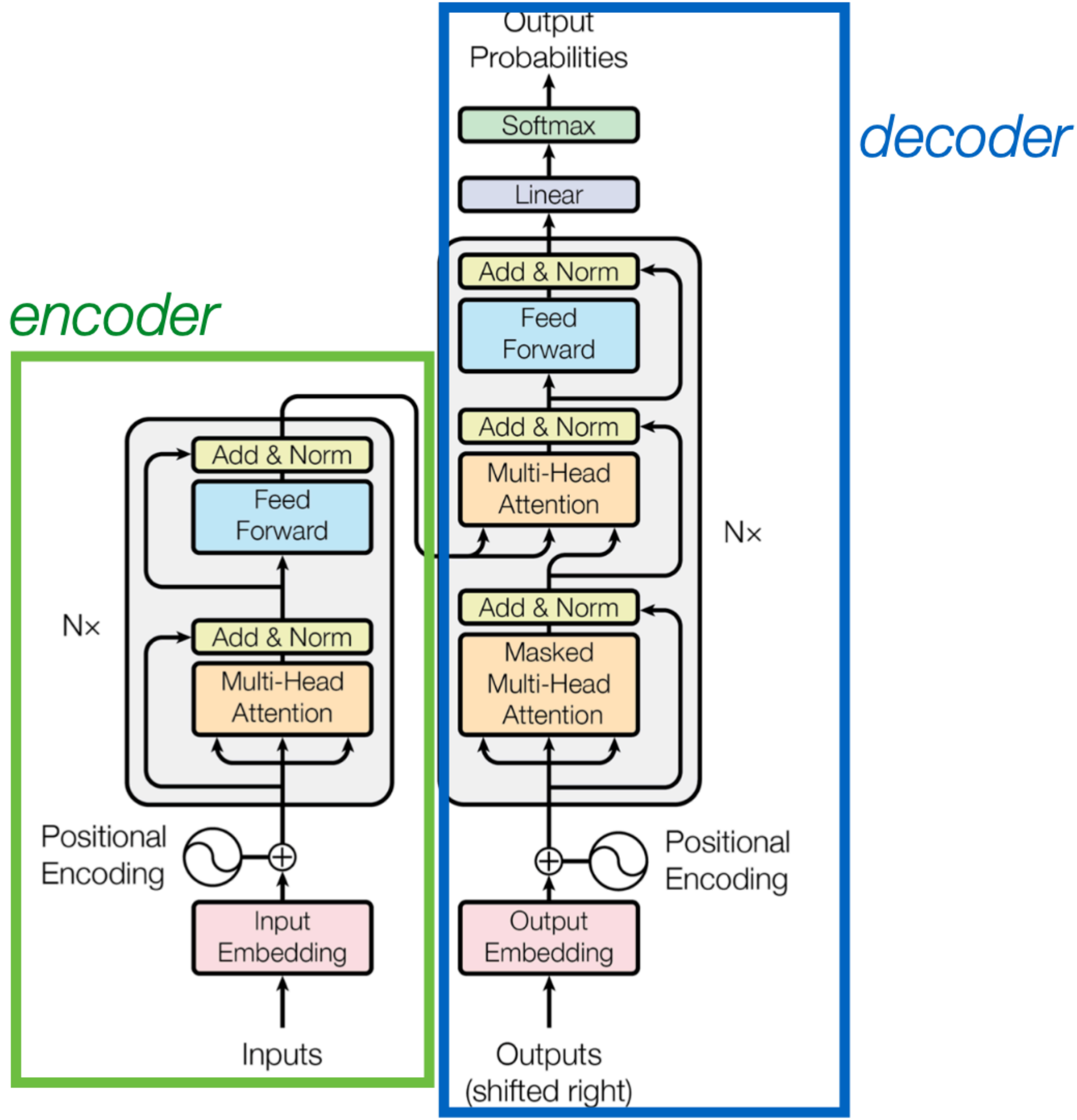
Transformers

QUESTION: We can parallelize in training (compute z_1, z_2, z_3 in parallel).

Can we also parallelize at test time?

The cat is ...

the dog is ...



Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Add & Norm
Masked
Multi-Head
Attention
Positional
Encoding
Output
Embedding
Outputs
(shifted right)
decoder
encoder
Add & Norm
Feed
Forward
Nx
Add & Norm
Multi-Head
Attention
Positional
Encoding
Input
Embedding
Inputs

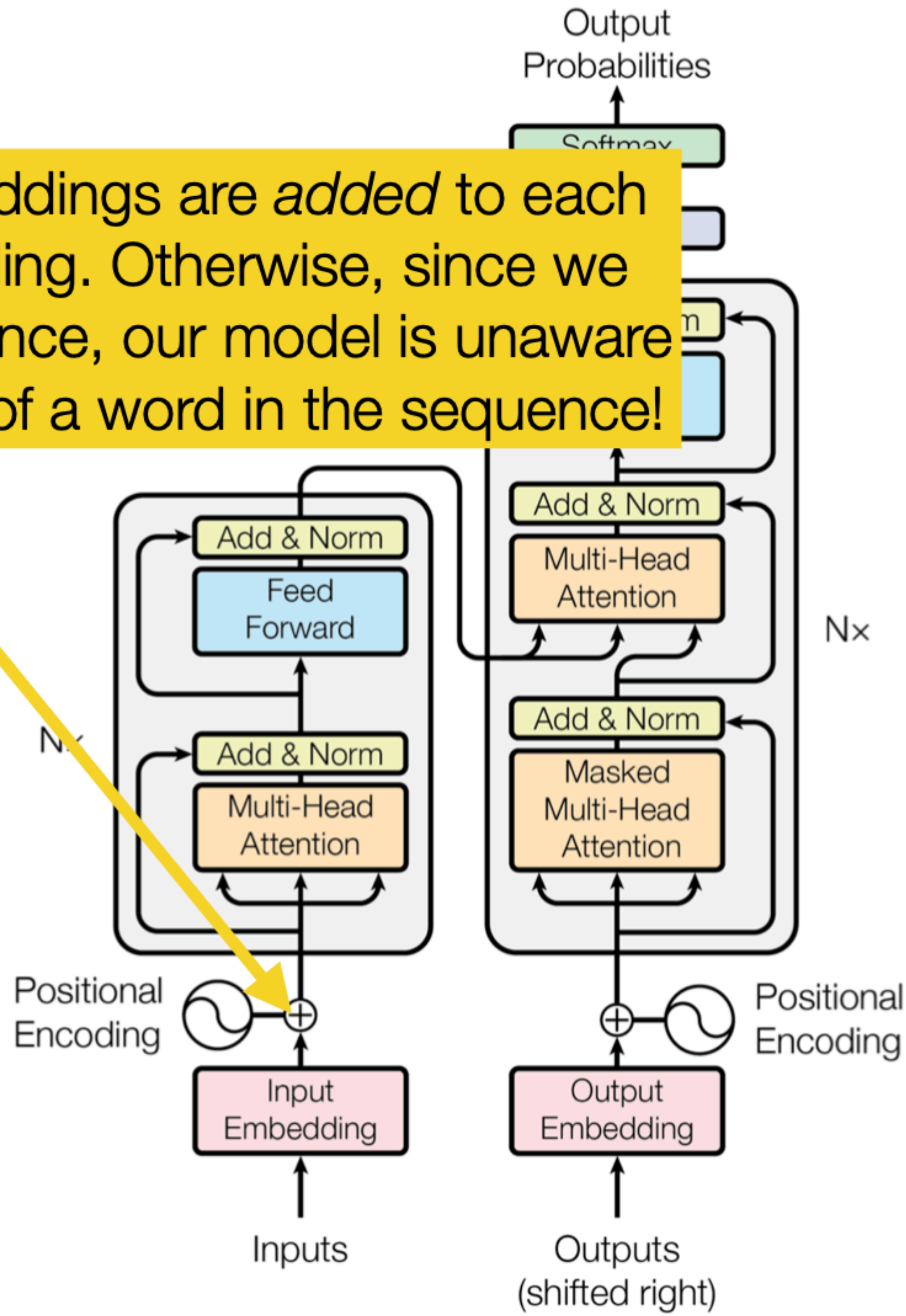
Blocks in a Seq2Seq Transformer:

Encoder:
- Unmasked self-attention
- Concat
- Feed forward

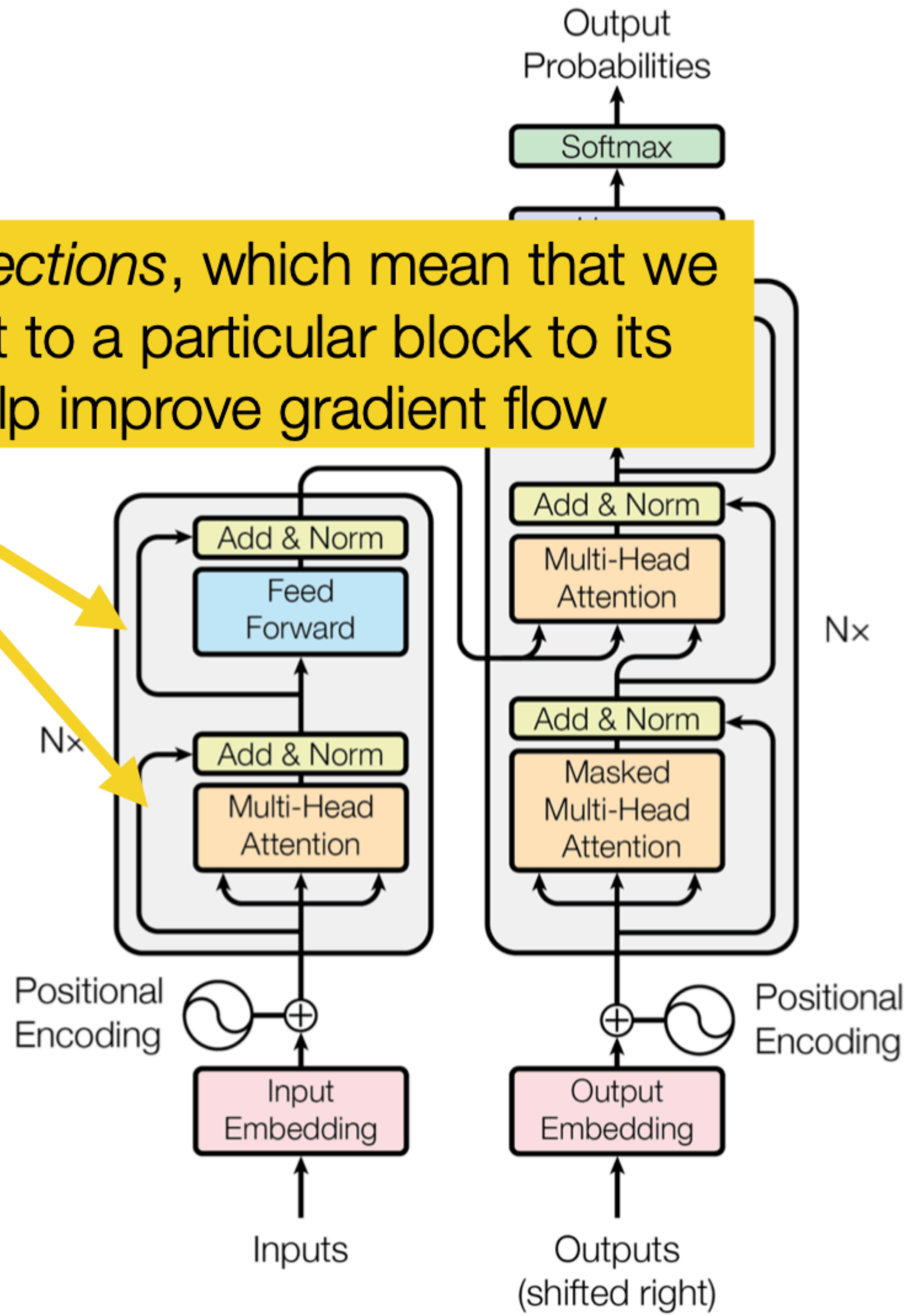
Decoder:
- Masked self-attention
- Concat
- Feed forward
- Cross attention
- Concat
- Feed forward

Both
Masked
self-
attention
block &
unmasked
cross-attention
block

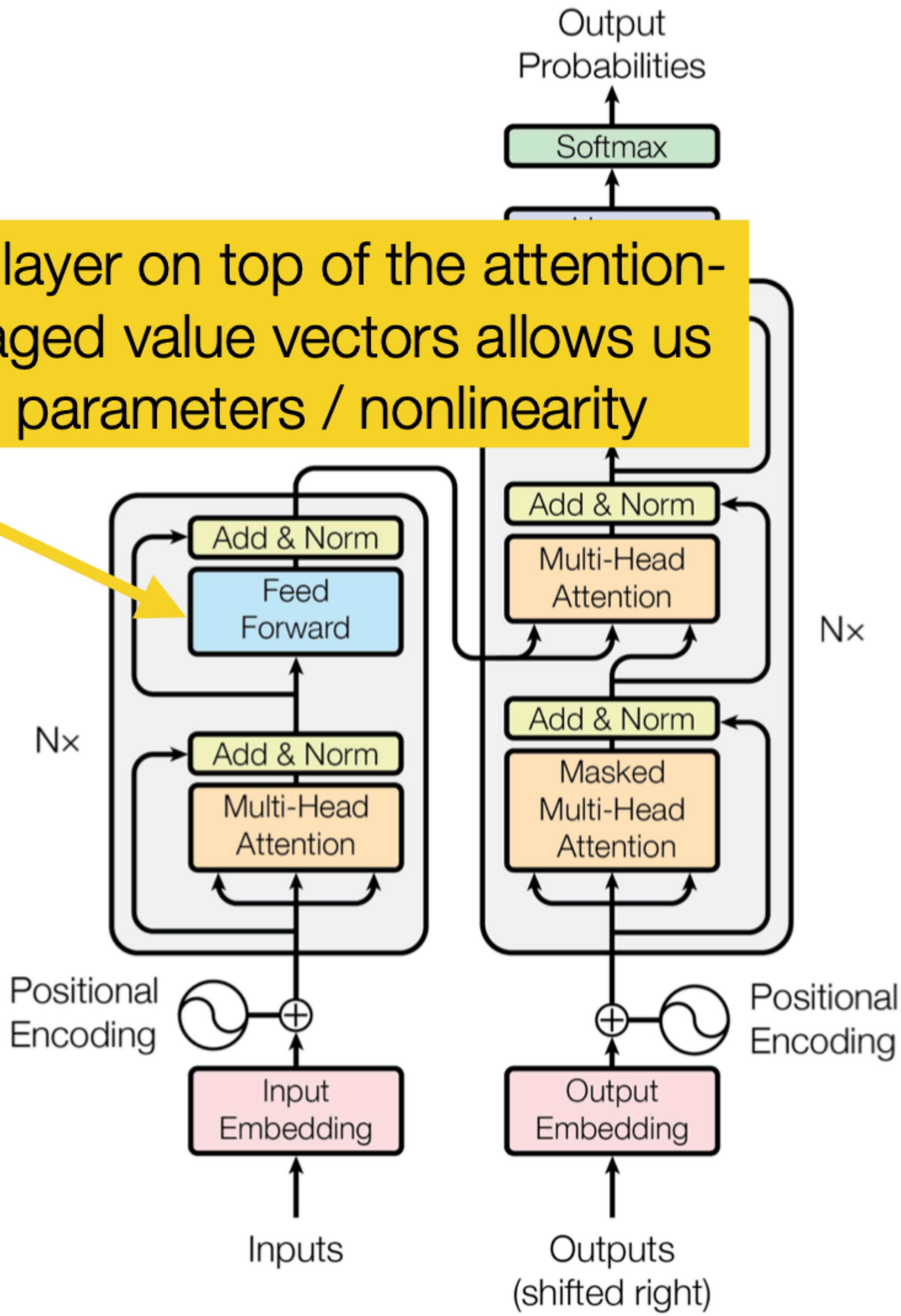
Position embeddings are *added* to each word embedding. Otherwise, since we have no recurrence, our model is unaware of the position of a word in the sequence!



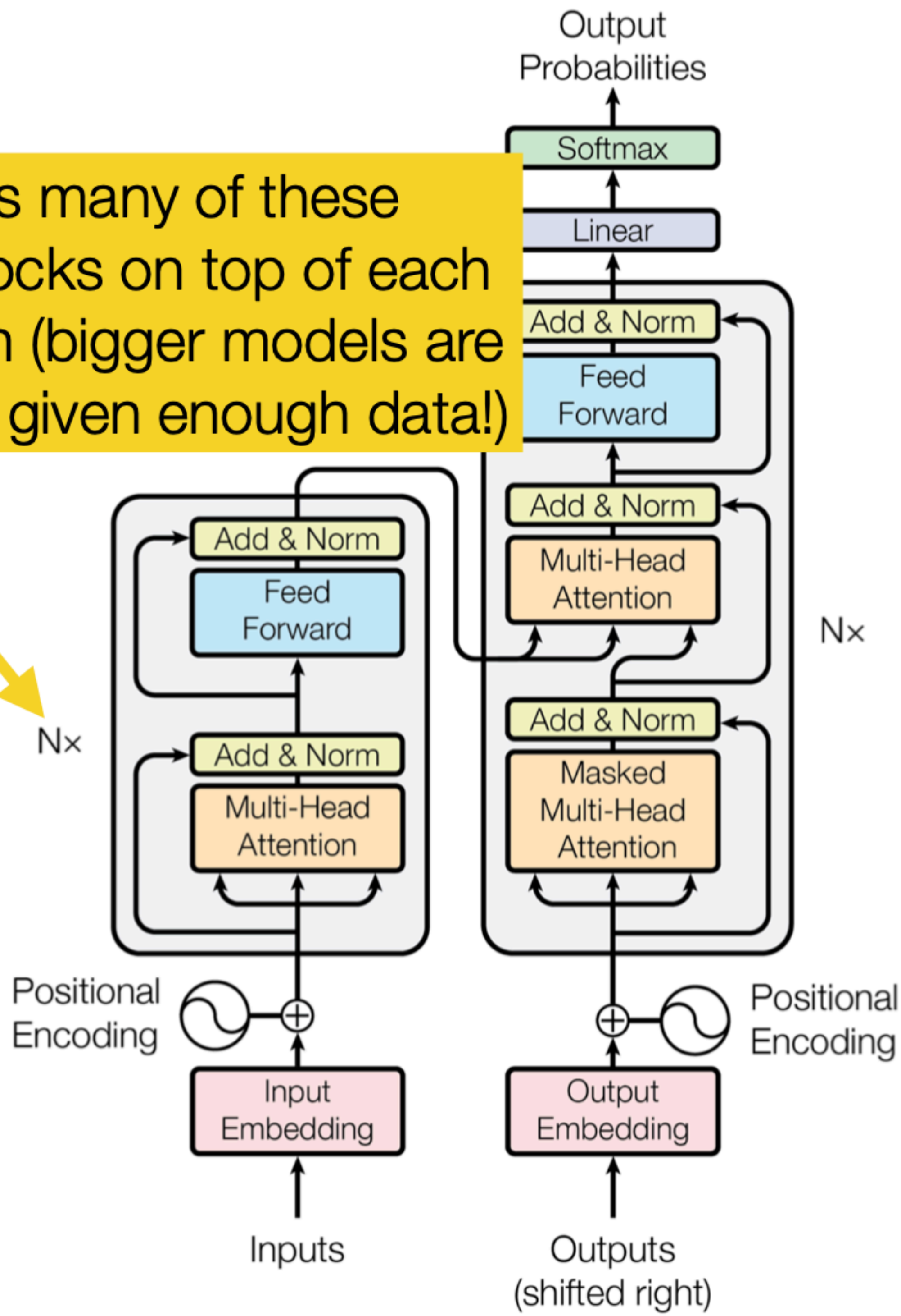
Residual connections, which mean that we add the input to a particular block to its output, help improve gradient flow



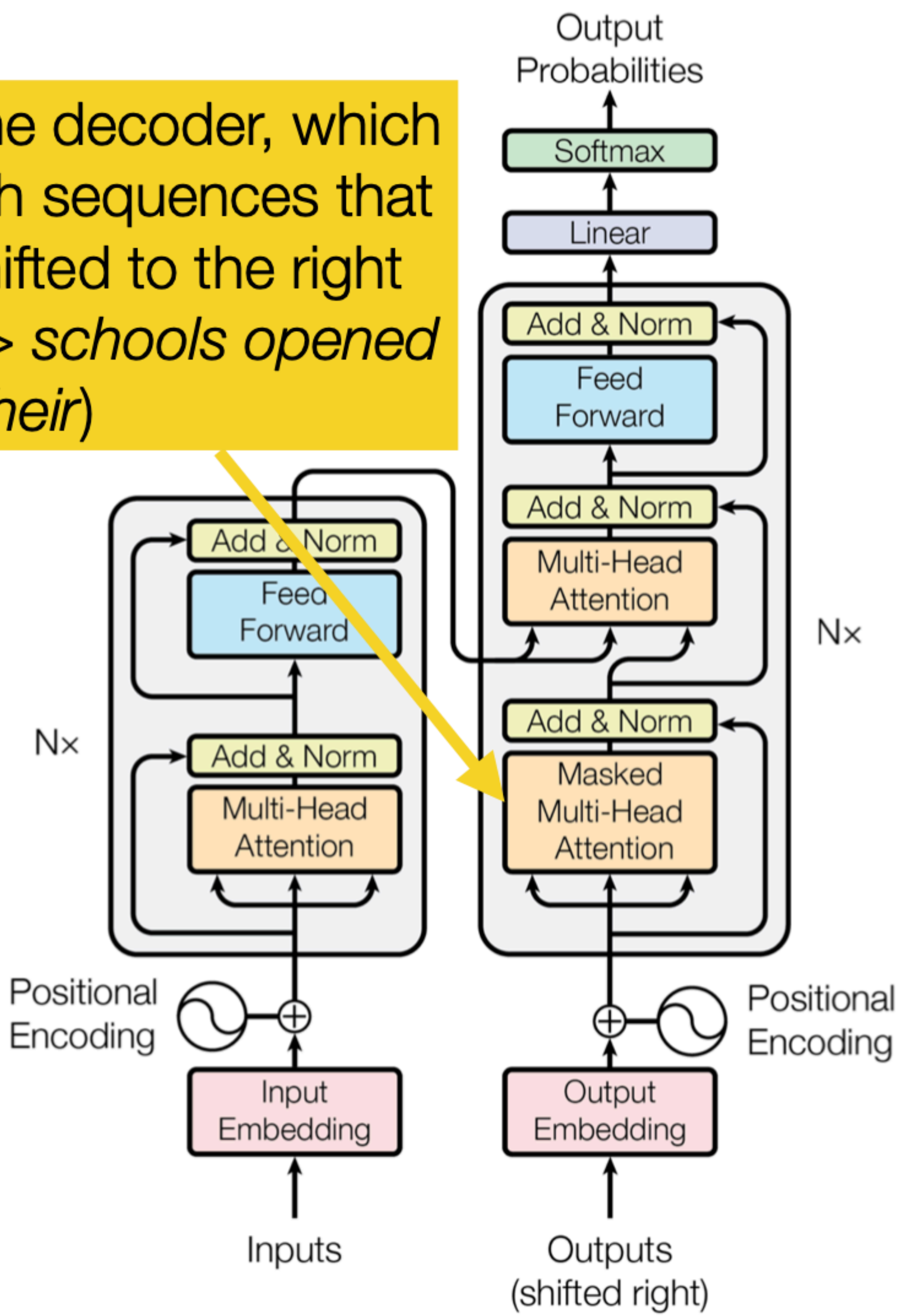
A feed-forward layer on top of the attention-weighted averaged value vectors allows us to add more parameters / nonlinearity



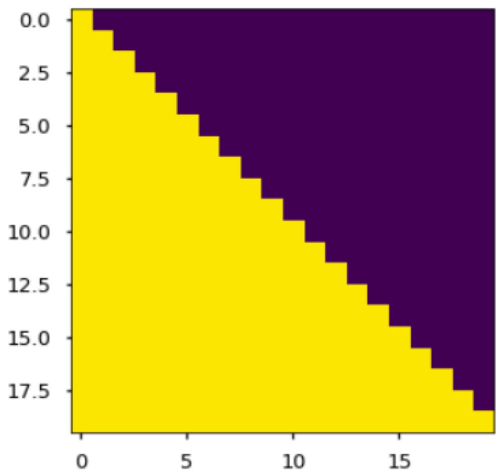
We stack as many of these *Transformer* blocks on top of each other as we can (bigger models are generally better given enough data!)

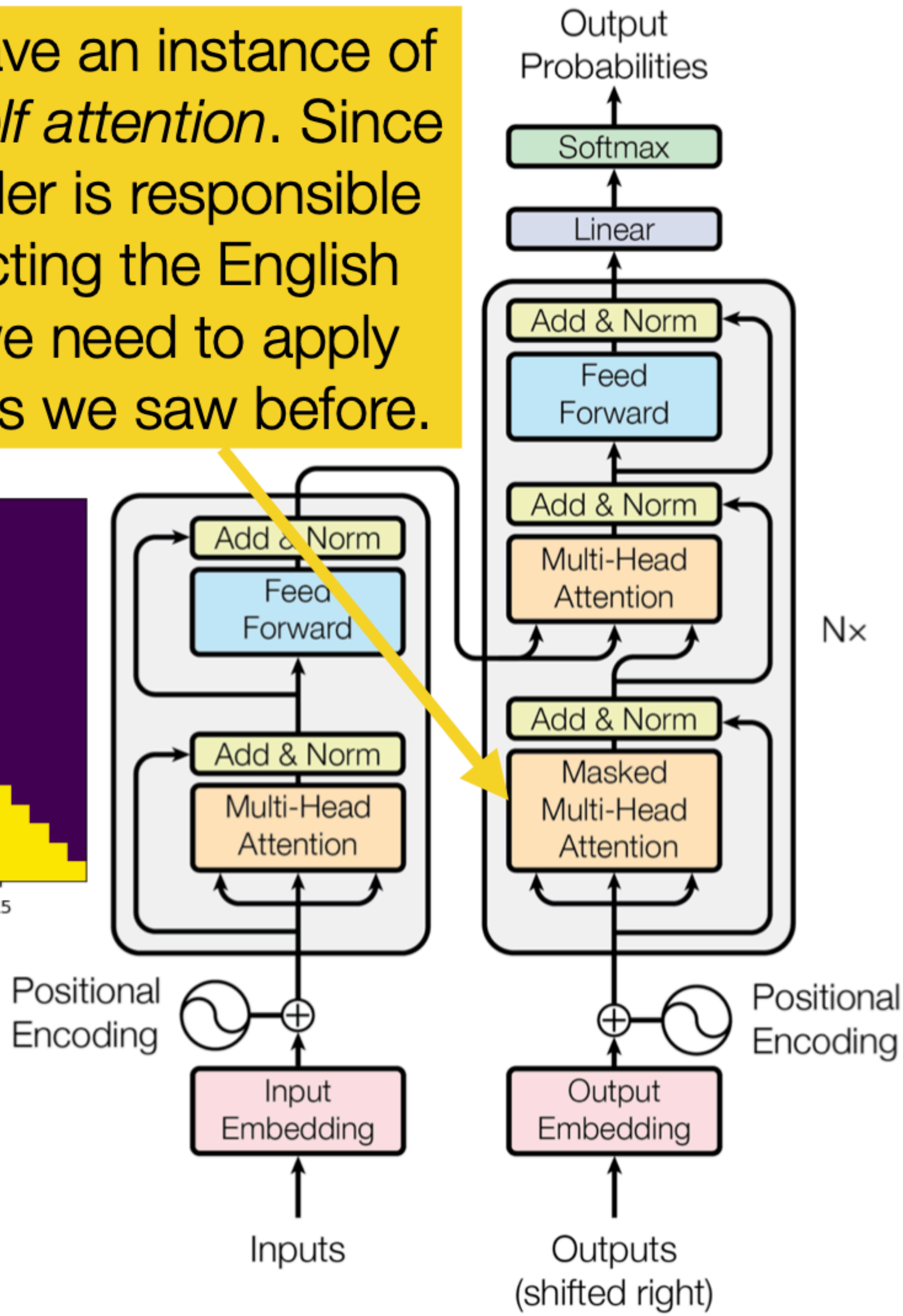


Moving onto the decoder, which takes in English sequences that have been shifted to the right (e.g., *<START> schools opened their*)

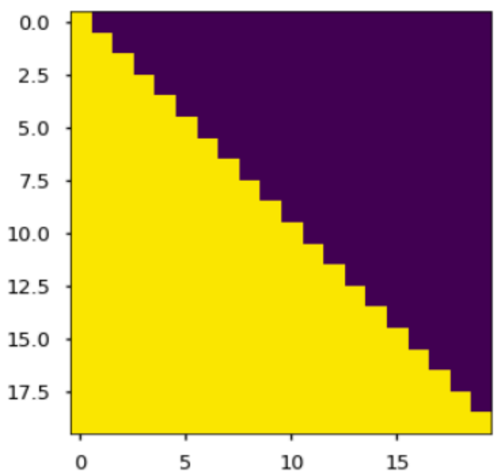


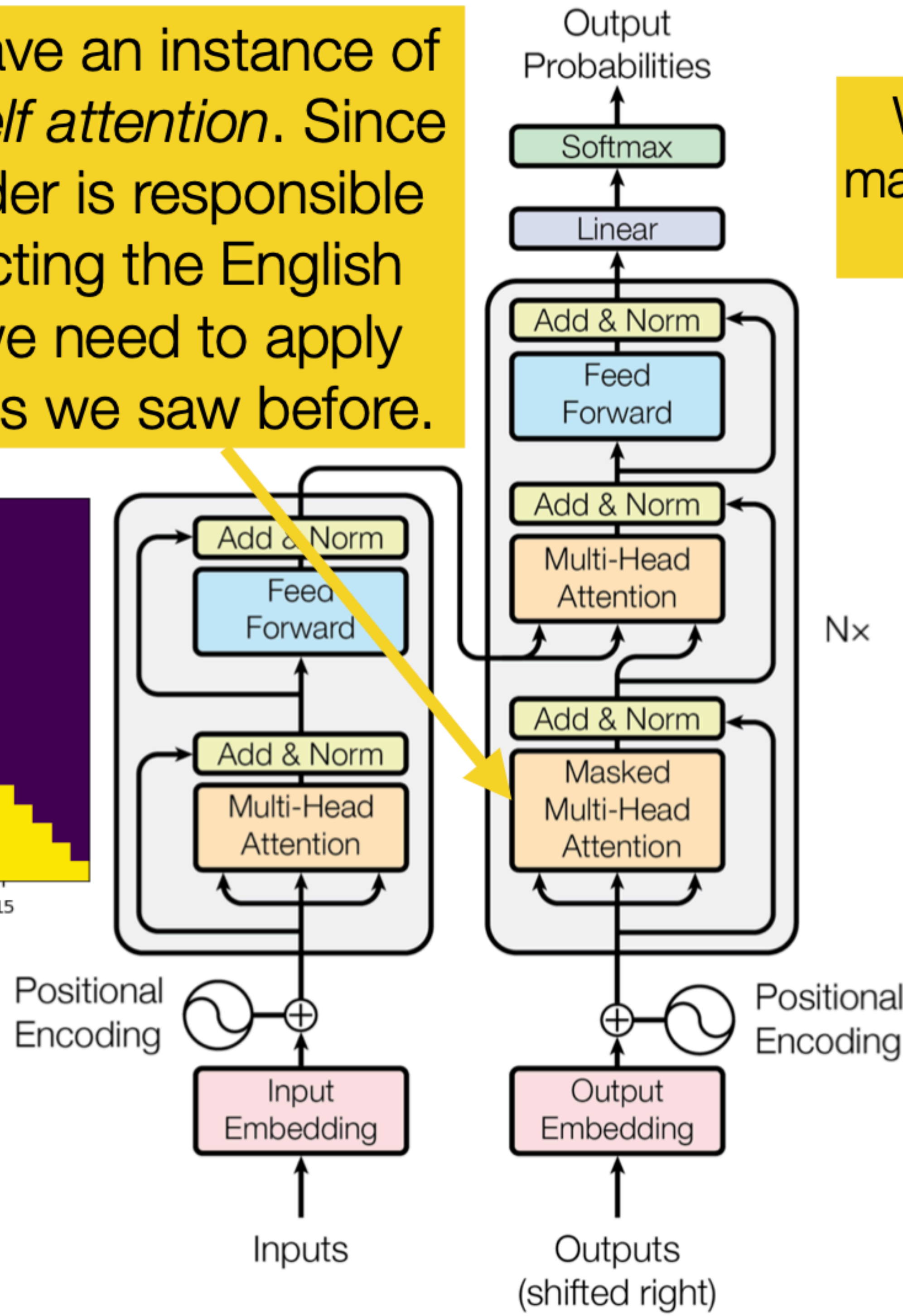
We first have an instance of *masked self attention*. Since the decoder is responsible for predicting the English words, we need to apply masking as we saw before.





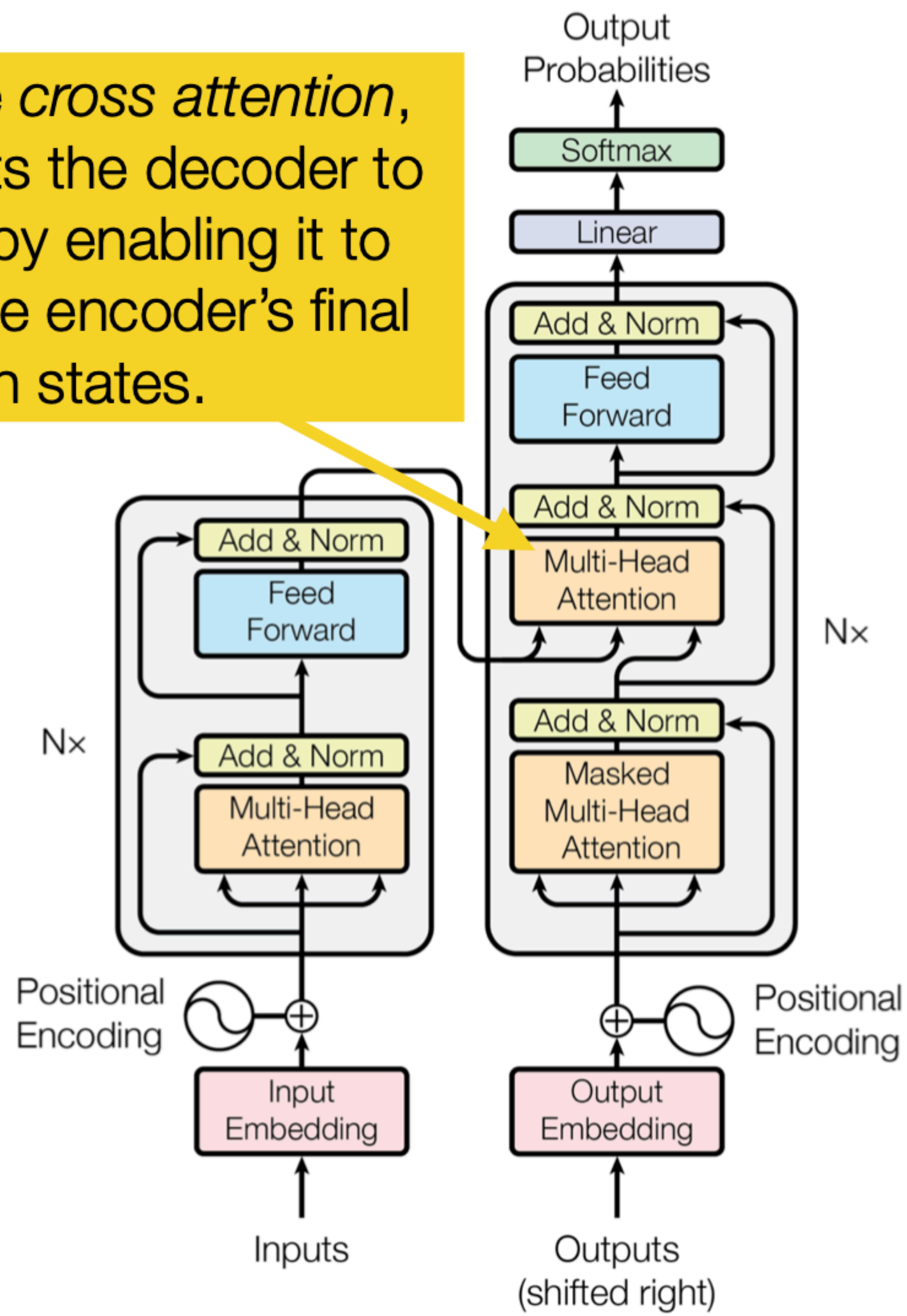
We first have an instance of *masked self attention*. Since the decoder is responsible for predicting the English words, we need to apply masking as we saw before.



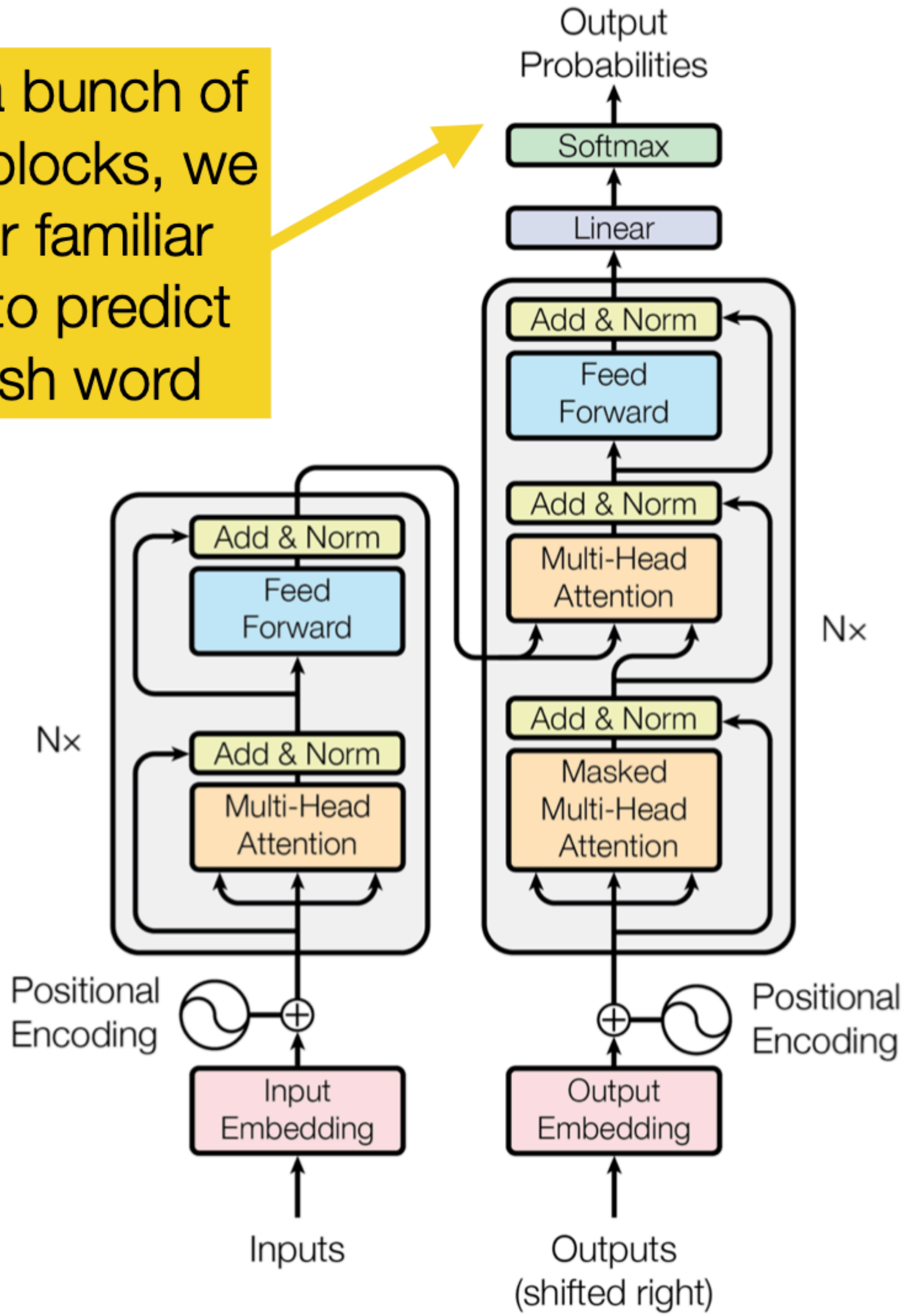


Why don't we do masked self-attention in the encoder?

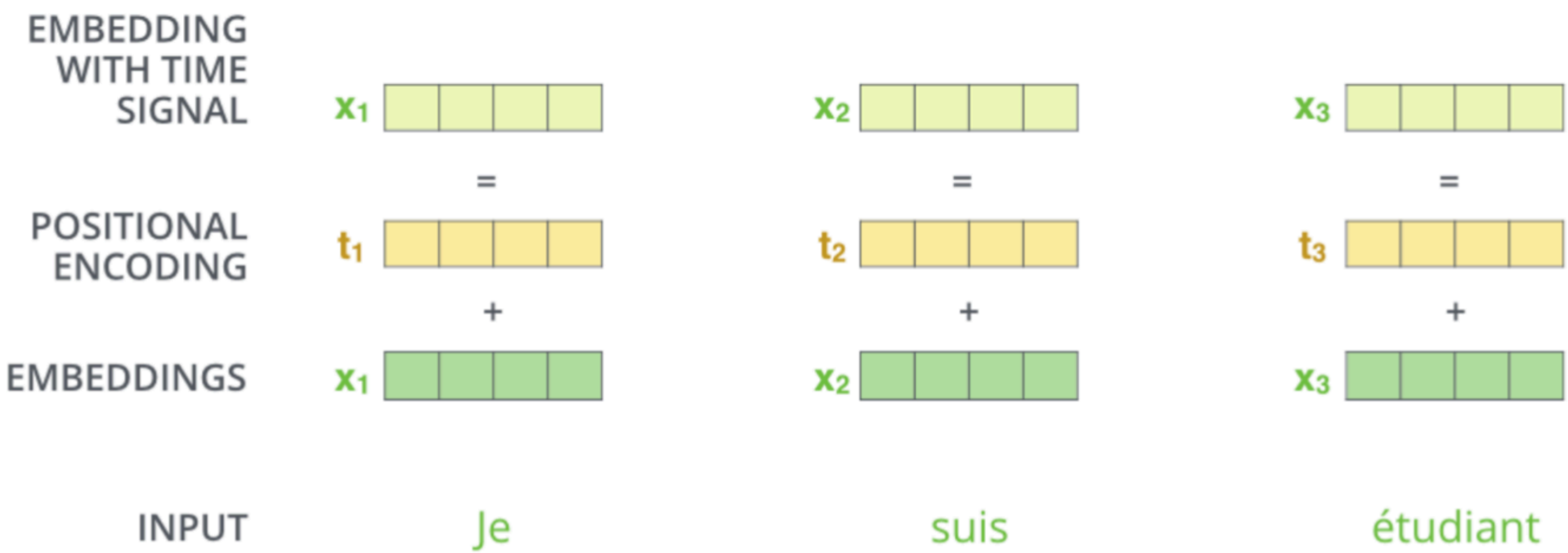
Now, we have *cross attention*, which connects the decoder to the encoder by enabling it to attend over the encoder's final hidden states.



After stacking a bunch of these decoder blocks, we finally have our familiar Softmax layer to predict the next English word



Positional encoding



$w_1 : w_3$ $w_4 : w_6$

Creating positional encodings?

- We could just concatenate a fixed value to each time step (e.g., 1, 2, 3, … 1000) that corresponds to its position, but then what happens if we get a sequence with 5000 words at test time?

- We want something that can generalize to arbitrary sequence lengths. We also may want to make attending to *relative positions* (e.g., tokens in a local window to the current token) easier.

- Distance between two positions should be consistent with variable-length inputs

Intuitive example

$0:$	0	0	0	0	$8:$	1	0	0	0
$1:$	0	0	0	1	$9:$	1	0	0	1
$2:$	0	0	1	0	$10:$	1	0	1	0
$3:$	0	0	1	1	$11:$	1	0	1	1
$4:$	0	1	0	0	$12:$	1	1	0	0
$5:$	0	1	0	1	$13:$	1	1	0	1
$6:$	0	1	1	0	$14:$	1	1	1	0
$7:$	0	1	1	1	$15:$	1	1	1	1

https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Newer Solution: Rotary Positional Embeddings (RoPE)

Su et al. (2023)

Key Idea: instead of adding a positional embedding, **rotate** the word embedding.

The angle of rotation (θ) is proportional to the word's position in the sentence.

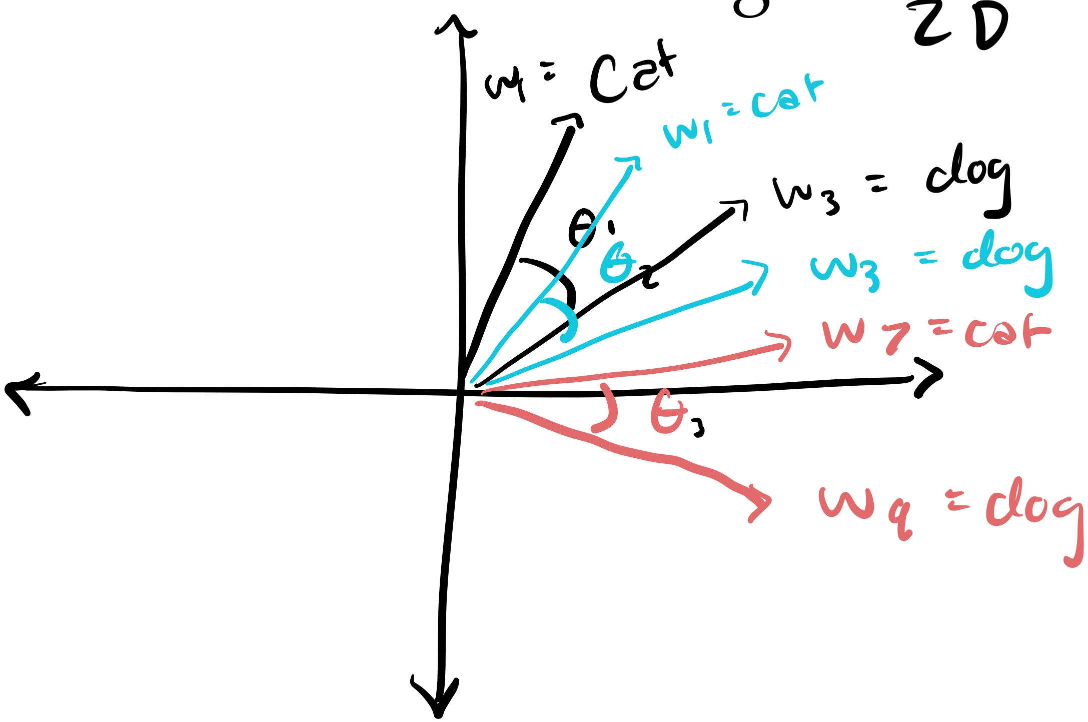
Two advantages: 1) **efficient caching** and 2) **preserves cosine similarity** between rotated embeddings at the same relative distance.

Newer Solution: Rotary Positional Embeddings (RoPE)

Su et al. (2023)

Key Idea: instead of adding a positional embedding, **rotate** the word embedding.

2D visualization



$\theta_1 = \theta_2 = \theta_3$

Do Positional Embeddings Actually Matter?

5034 words

5017 words

The Impact of Positional Encoding on Length Generalization in Transformers

Amirhossein Kazemnejad[1], **Inkit Padhi**[2]
Karthikeyan Natesan Ramamurthy[2], **Payel Das**[2], **Siva Reddy**[1,3,4]
[1]Mila, McGill University; [2]IBM Research;
[3]Facebook CIFAR AI Chair; [4]ServiceNow Research
{amirhossein.kazemnejad,siva.reddy}@mila.quebec
inkpad@ibm.com, {knatesa,daspa}@us.ibm.com

Length generalization,
to larger ones, is a cri
language models. Posi
influencing length gen
on extrapolation in dov
a systematic empirical
of decoder-only Trans
including Absolute Po
Rotary, in addition to
evaluation encompasses a battery of reasoning and mathematical tasks. Our findings

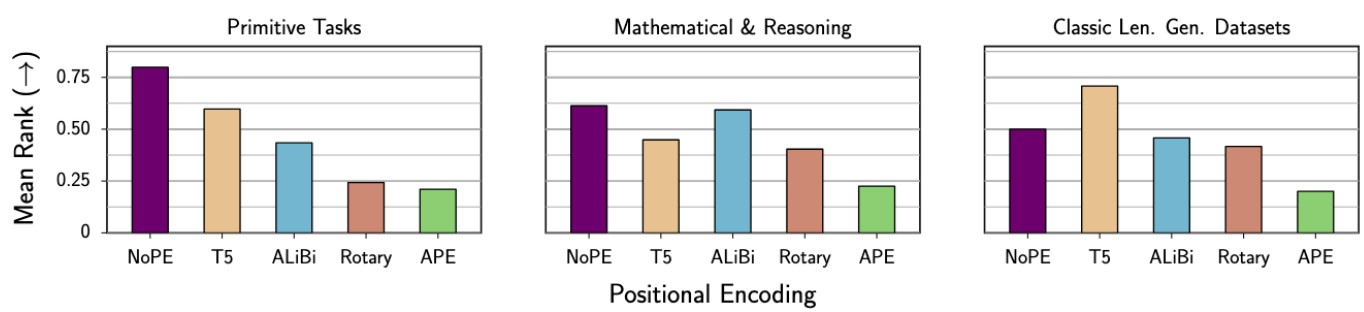


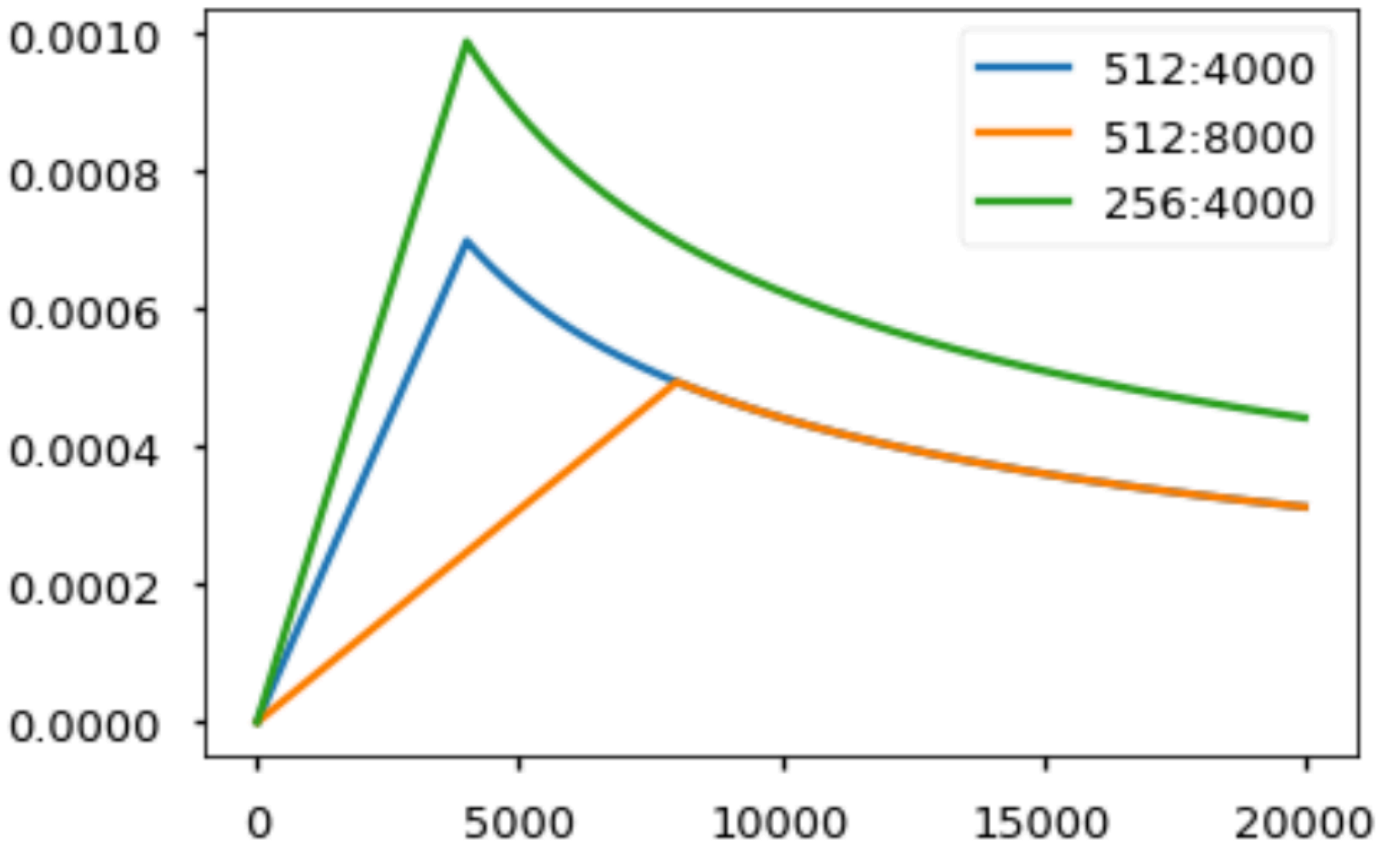
Figure 2: Aggregate ranking of positional encoding methods on length extrapolation across three different groups of tasks. No PE and T5's Relative Bias outperform other encoding methods in these categories.

Hacks to make Transformers work

Optimizer

We used the Adam optimizer (cite) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. We varied the learning rate over the course of training, according to the formula: $lrate = d_{\text{model}}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5})$ This corresponds to increasing the learning rate linearly for the first $warmup_steps$ training steps, and decreasing it thereafter proportionally to the inverse square root of the step number. We used $warmup_steps = 4000$.

> *Note: This part is very important. Need to train with this setup of the model.*



Label Smoothing

During training, we employed label smoothing of value $\epsilon_{ls} = 0.1$ (cite). This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

> *We implement label smoothing using the KL div loss. Instead of using a one-hot target distribution, we create a distribution that has `confidence` of the correct word and the rest of the `smoothing` mass distributed throughout the vocabulary.*

I went to class and took ___

cats	TV	notes	took	sofa
0	0	1	0	0

Label Smoothing

During training, we employed label smoothing of value $\epsilon_{ls} = 0.1$ (cite). This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

> *We implement label smoothing using the KL div loss. Instead of using a one-hot target distribution, we create a distribution that has* `confidence` *of the correct word and the rest of the* `smoothing` *mass distributed throughout the vocabulary.*

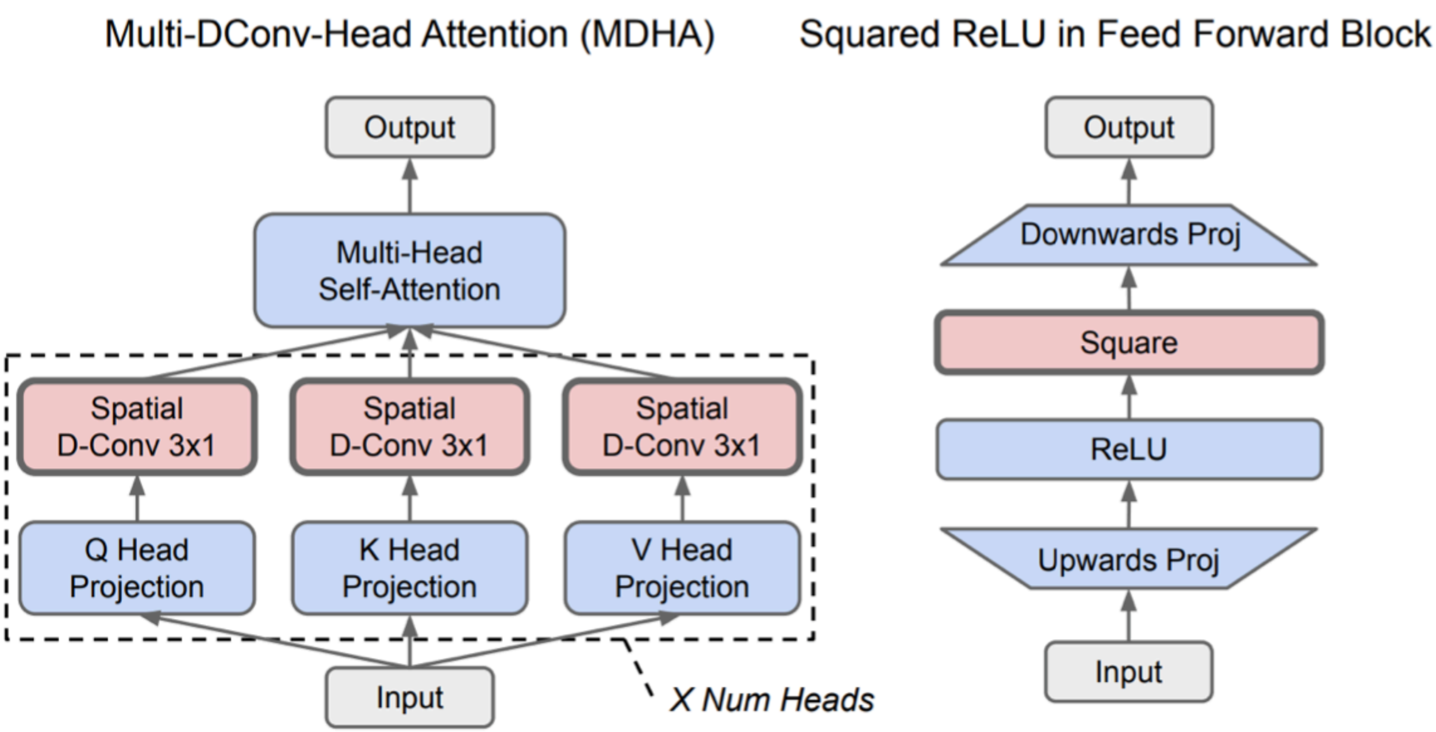
I went to class and took ___

cats	TV	notes	took	sofa
0	0	1	0	0
0.025	0.025	0.9	0.025	0.025

with label smoothing

Why these decisions?

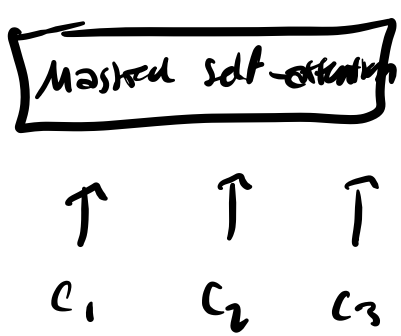
Unsatisfying answer: they empirically worked well.
Neural architecture search finds even better Transformer variants:



Primer: Searching for efficient Transformer architectures… So et al., Sep. 2021

Types of Transformers

Decoder only (GPT models, Llama models)



Useful for generating text