Transformers (II)

CS 685, Fall 2025

Advanced Natural Language Processing

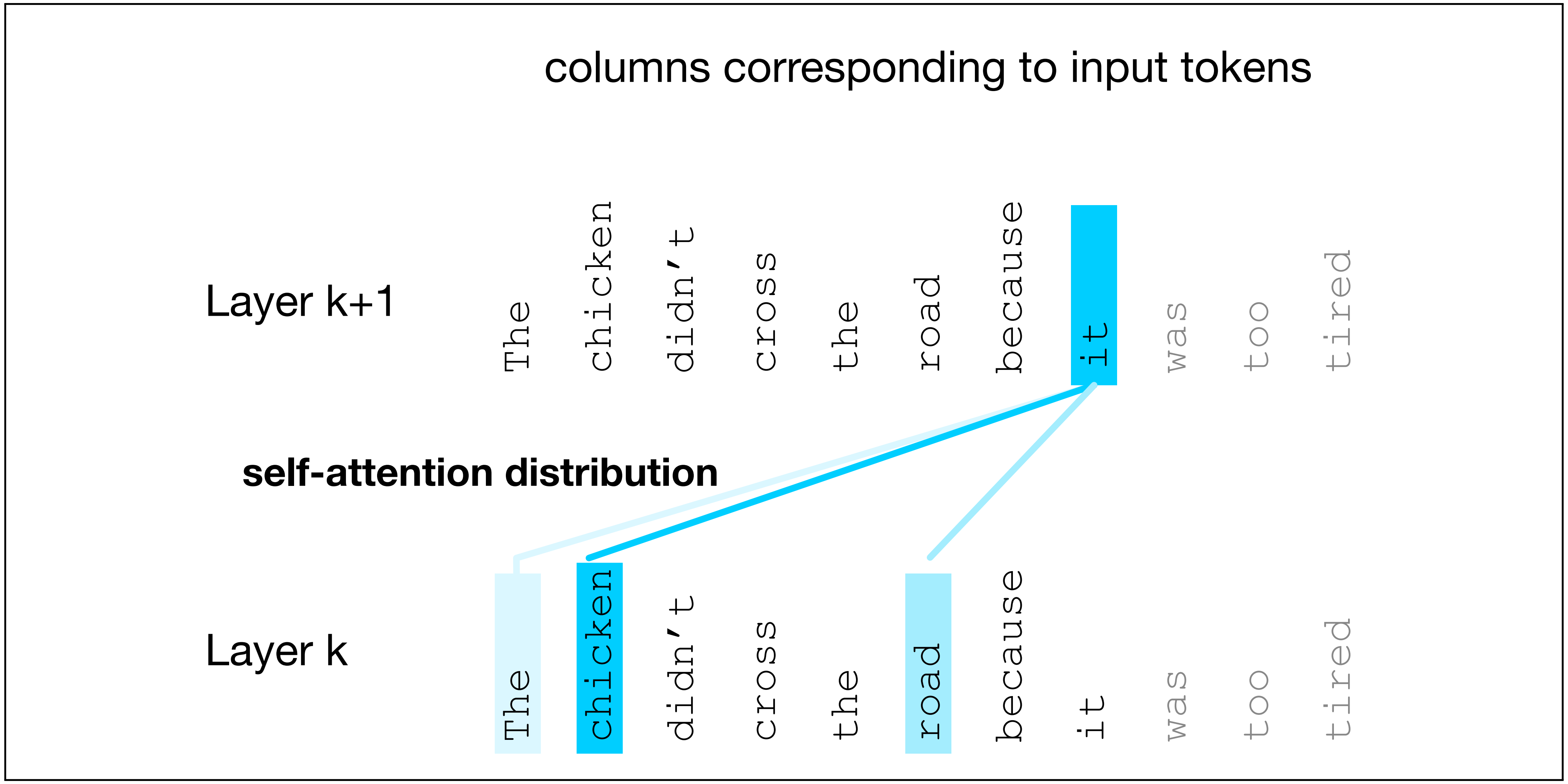
https://people.cs.umass.edu/~brenocon/cs685_f25/

Brendan O'Connor

College of Information and Computer Sciences

University of Massachusetts Amherst

- Practice midterm -- posted to Piazza



$$
\begin{aligned}
\mathbf{q}_i = \mathbf{x}_i\mathbf{W}^{\mathbf{Q}};\quad \mathbf{k}_j &= \mathbf{x}_j\mathbf{W}^{\mathbf{K}};\quad \mathbf{v}_j = \mathbf{x}_j\mathbf{W}^{\mathbf{V}} \\
\text{score}(\mathbf{x}_i,\mathbf{x}_j) &= \frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d_k}} \\
\alpha_{ij} &= \text{softmax}(\text{score}(\mathbf{x}_i,\mathbf{x}_j))\ \ \forall j \le i \\
\mathbf{a}_i &= \sum_{j \le i} \alpha_{ij}\mathbf{v}_j
\end{aligned}
$$

[SLP3, ch. 8]

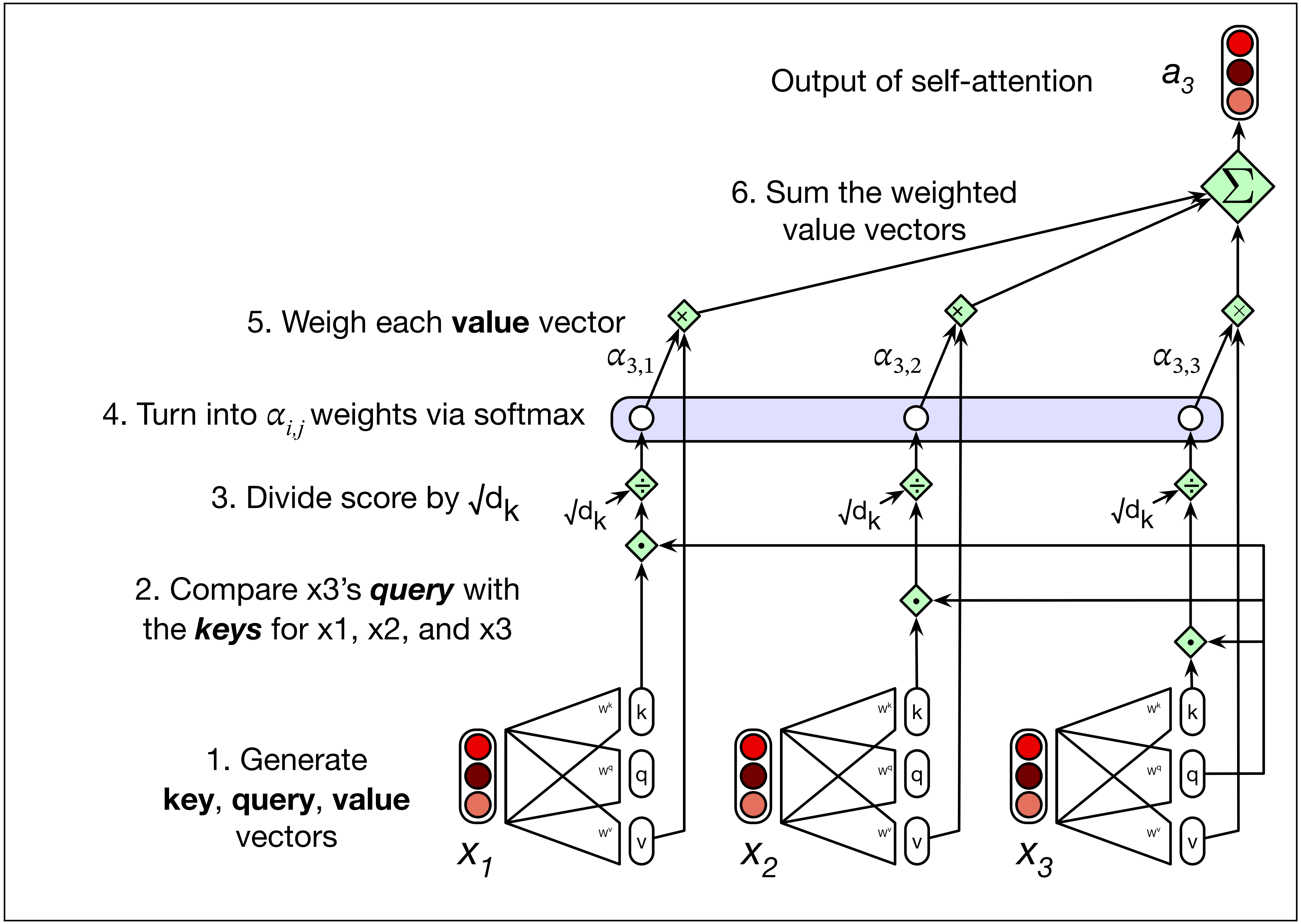


Figure 9.4 Calculating the value of $\mathbf{a}_3$, the third element of a sequence using causal (left-to-right) self-attention.

[SLP3, ch. 8]

- Mathematical comparisons
  - Associate array, but soft
  - Kernel functions, but learned
  - Sequence automaton (choosing, copying, etc.)

- <u>Parallelization</u> of query-key products
  - Comp. advantage vs. RNNs
  - Need to mask out future information

Full Transformer block

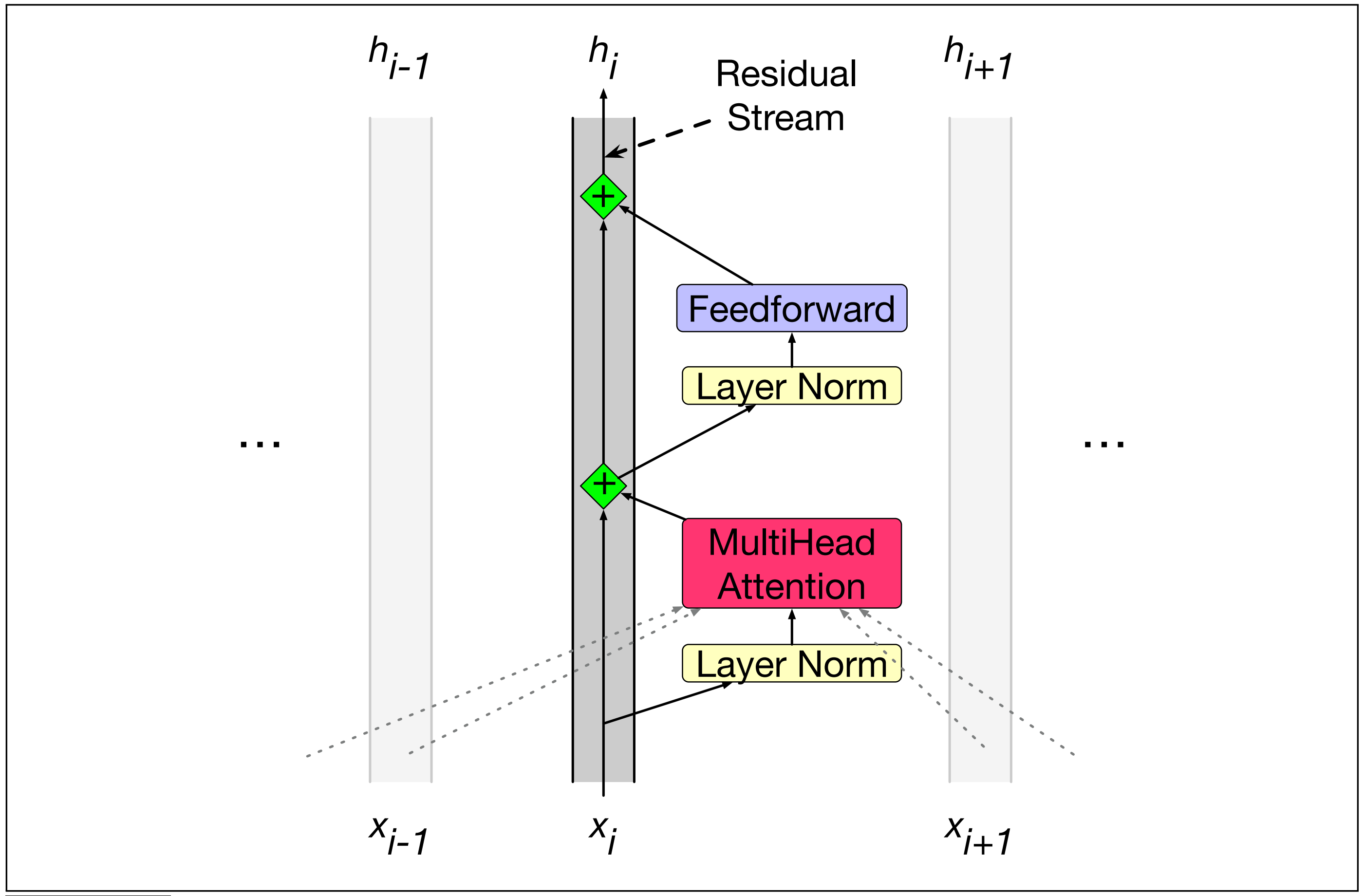


Figure 8.6 The architecture of a transformer block showing the **residual stream**. This figure shows the **prenorm** version of the architecture, in which the layer norms happen before the attention and feedforward layers rather than after.

$$
\begin{aligned}
\mathbf{t}_i^1 &= \mathrm{LayerNorm}(\mathbf{x}_i) \\
\mathbf{t}_i^2 &= \mathrm{MultiHeadAttention}(\mathbf{t}_i^1, [\mathbf{t}_1^1, \cdots, \mathbf{t}_N^1]) \\
\mathbf{t}_i^3 &= \mathbf{t}_i^2 + \mathbf{x}_i \\
\mathbf{t}_i^4 &= \mathrm{LayerNorm}(\mathbf{t}_i^3) \\
\mathbf{t}_i^5 &= \mathrm{FFN}(\mathbf{t}_i^4) \\
\mathbf{h}_i &= \mathbf{t}_i^5 + \mathbf{t}_i^3
\end{aligned}
$$

$$\mathrm{FFN}(\mathbf{x}_i) = \mathrm{ReLU}(\mathbf{x}_i \mathbf{W_1} + b_1)\mathbf{W_2} + b_2$$

$$\mathrm{LayerNorm}(\mathbf{x}) = \gamma \frac{(\mathbf{x} - \mu)}{\sigma} + \beta$$

- Multiple layers with MLPs (FF layers)
- Multiple heads

[SLP3, ch. 8]

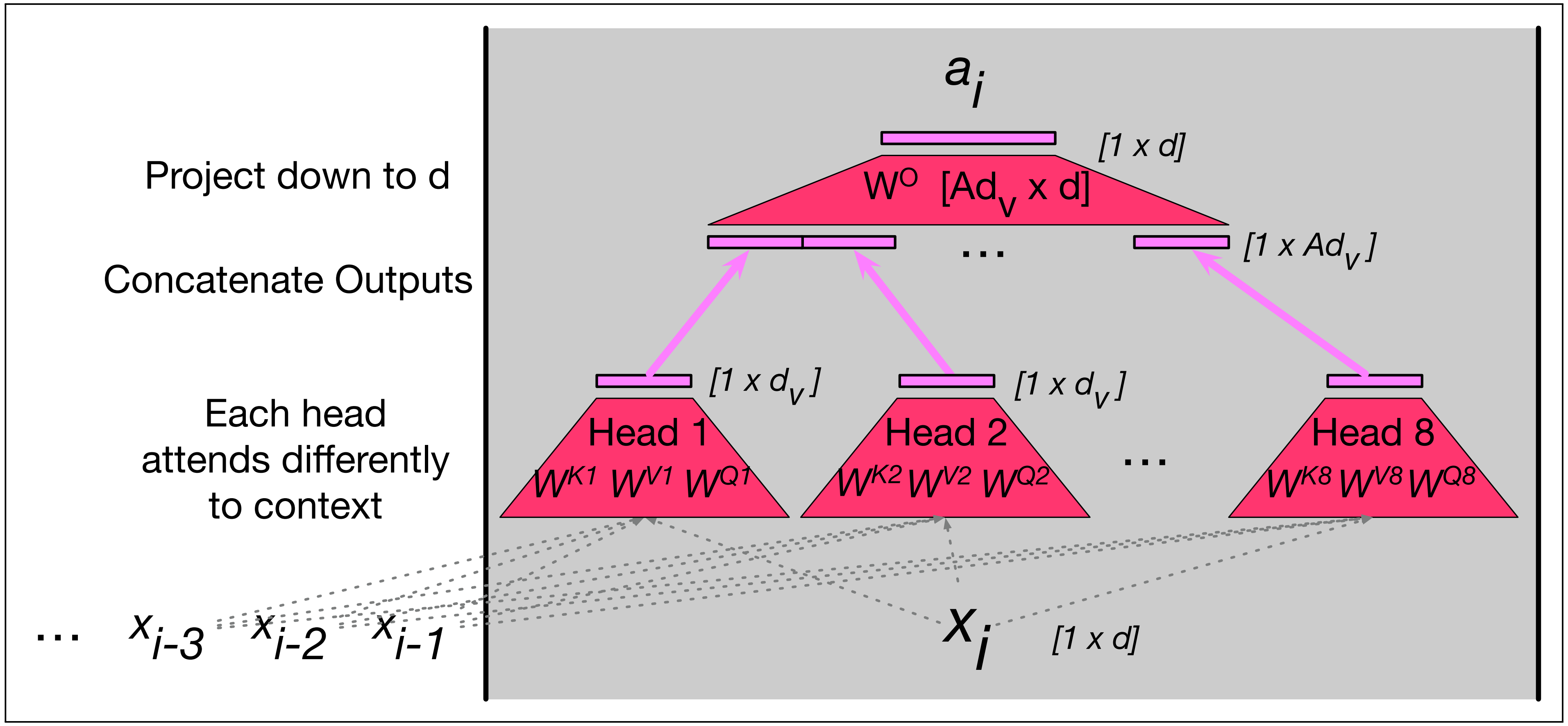


Figure 8.5 The multi-head attention computation for input $\mathbf{x}_i$, producing output $\mathbf{a}_i$. A multi-head attention layer has A heads, each with its own query, key, and value weight matrices. The outputs from each of the heads are concatenated and then projected down to d, thus producing an output of the same size as the input.

$$
\begin{aligned}
\mathbf{q}_i^c = \mathbf{x}_i\mathbf{W}^{\mathbf{Qc}};\quad \mathbf{k}_j^c = \mathbf{x}_j\mathbf{W}^{\mathbf{Kc}};\quad \mathbf{v}_j^c &= \mathbf{x}_j\mathbf{W}^{\mathbf{Vc}};\quad \forall\, c \quad 1 \le c \le A \\
\text{score}^c(\mathbf{x}_i, \mathbf{x}_j) &= \frac{\mathbf{q}_i^c \cdot \mathbf{k}_j^c}{\sqrt{d_k}} \\
\alpha_{ij}^c &= \text{softmax}(\text{score}^c(\mathbf{x}_i, \mathbf{x}_j)) \;\; \forall j \le i \\
\mathbf{head}_i^c &= \sum_{j \le i} \alpha_{ij}^c \mathbf{v}_j^c \\
\mathbf{a}_i &= (\mathbf{head}^1 \oplus \mathbf{head}^2 \ldots \oplus \mathbf{head}^A)\mathbf{W}^O \\
\text{MultiHeadAttention}(\mathbf{x}_i, [\mathbf{x}_1, \cdots, \mathbf{x}_{i-1}]) &= \mathbf{a}_i
\end{aligned}
$$

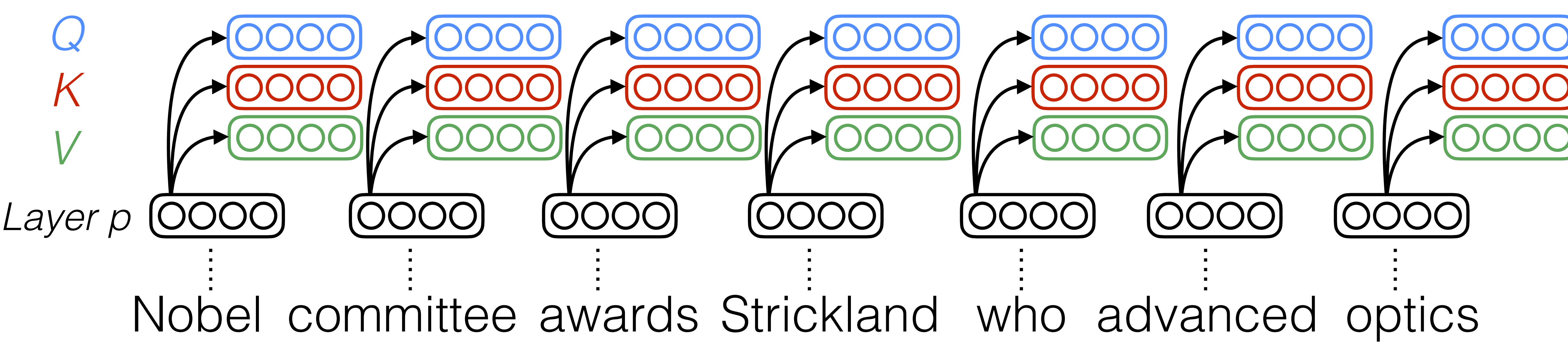
- Multiple layers with MLPs (FF layers)
- Multiple heads

[SLP3, ch. 8]

Self-attention

[Vaswani et al. 2017.
slide: Emma Strubell]

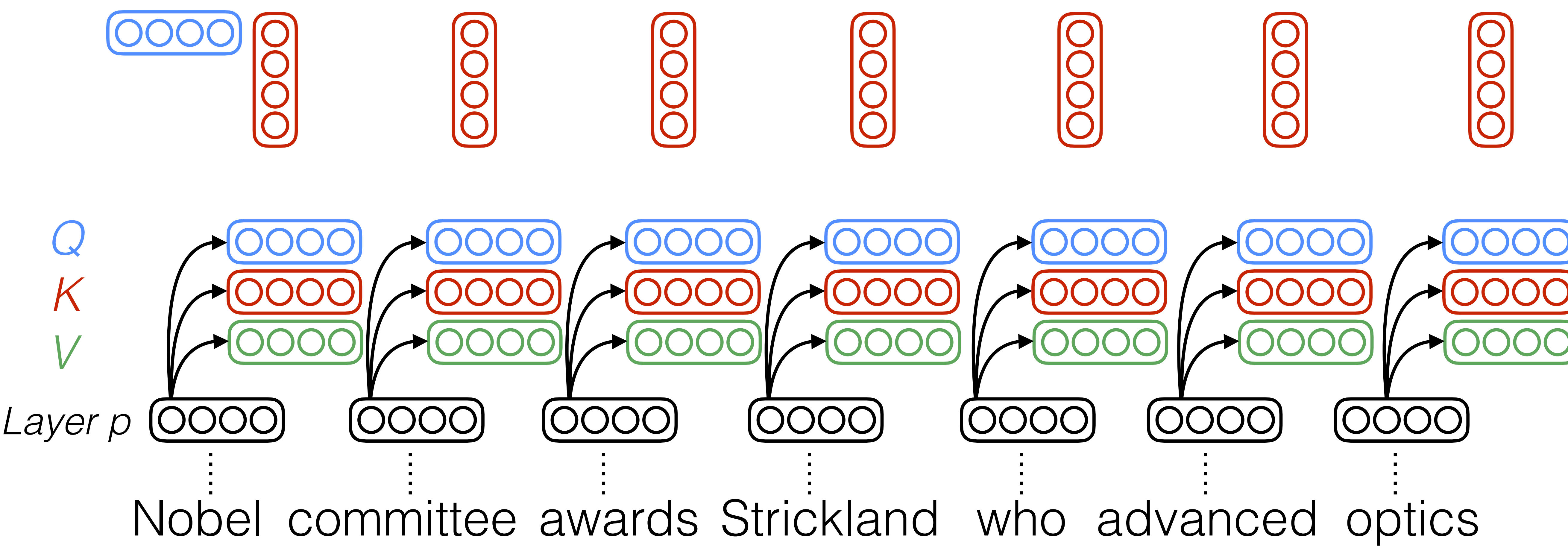
$$\mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Self-attention

[Vaswani et al. 2017.
slide: Emma Strubell]

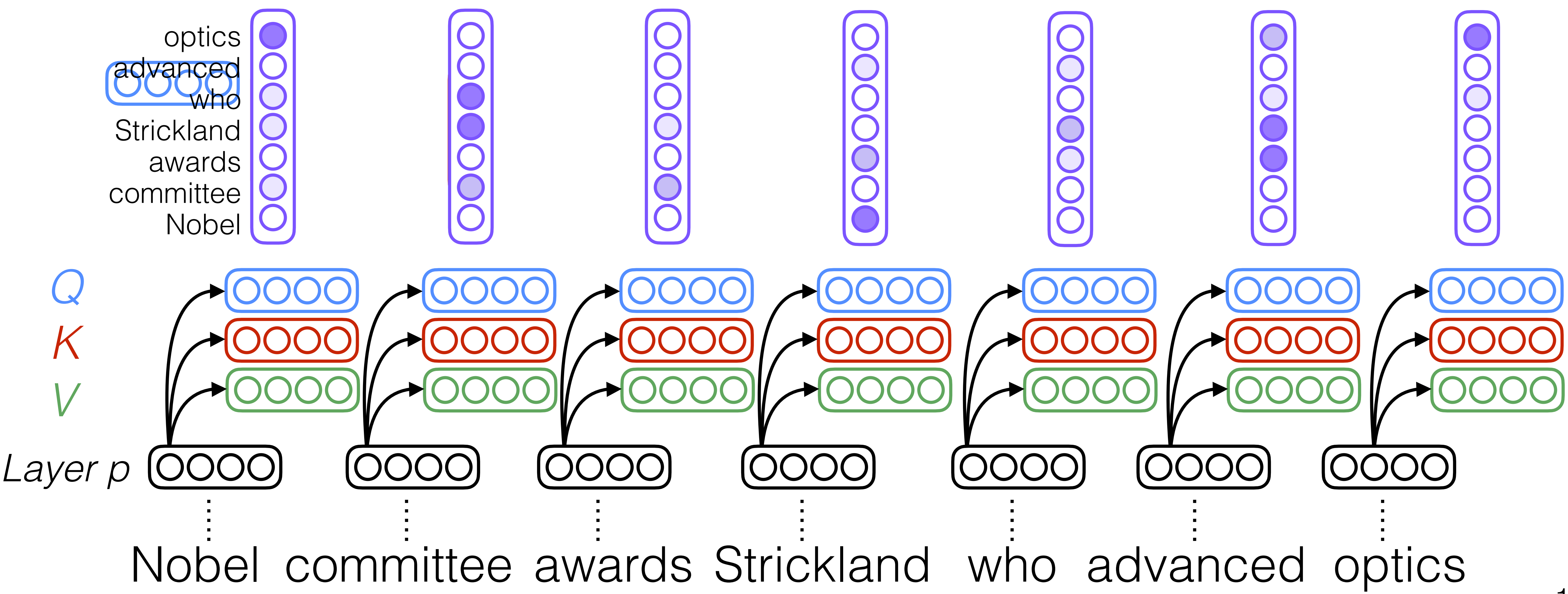
$$\mathrm{softmax}\left(\frac{\boxed{QK^T}}{\sqrt{d_k}}\right)V$$



Self-attention

[Vaswani et al. 2017.
slide: Emma Strubell]

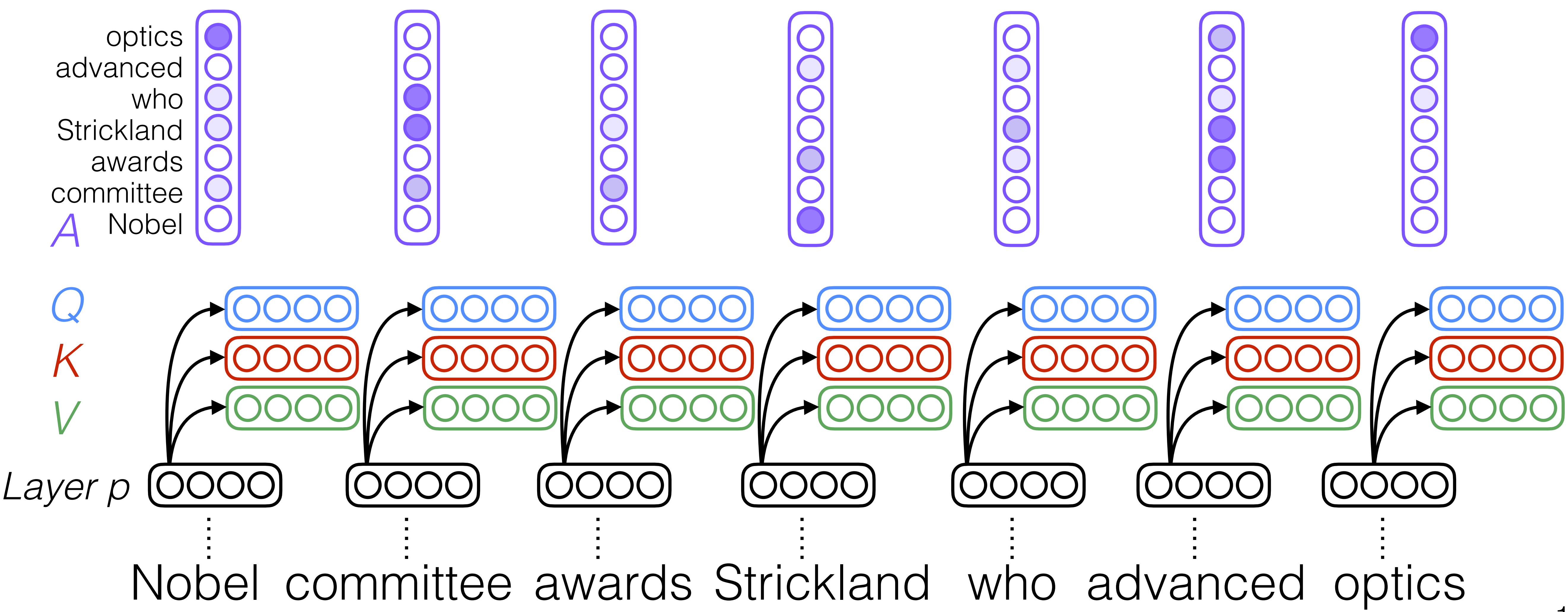
$$\boxed{\mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})}V$$



Self-attention

[Vaswani et al. 2017.
slide: Emma Strubell]

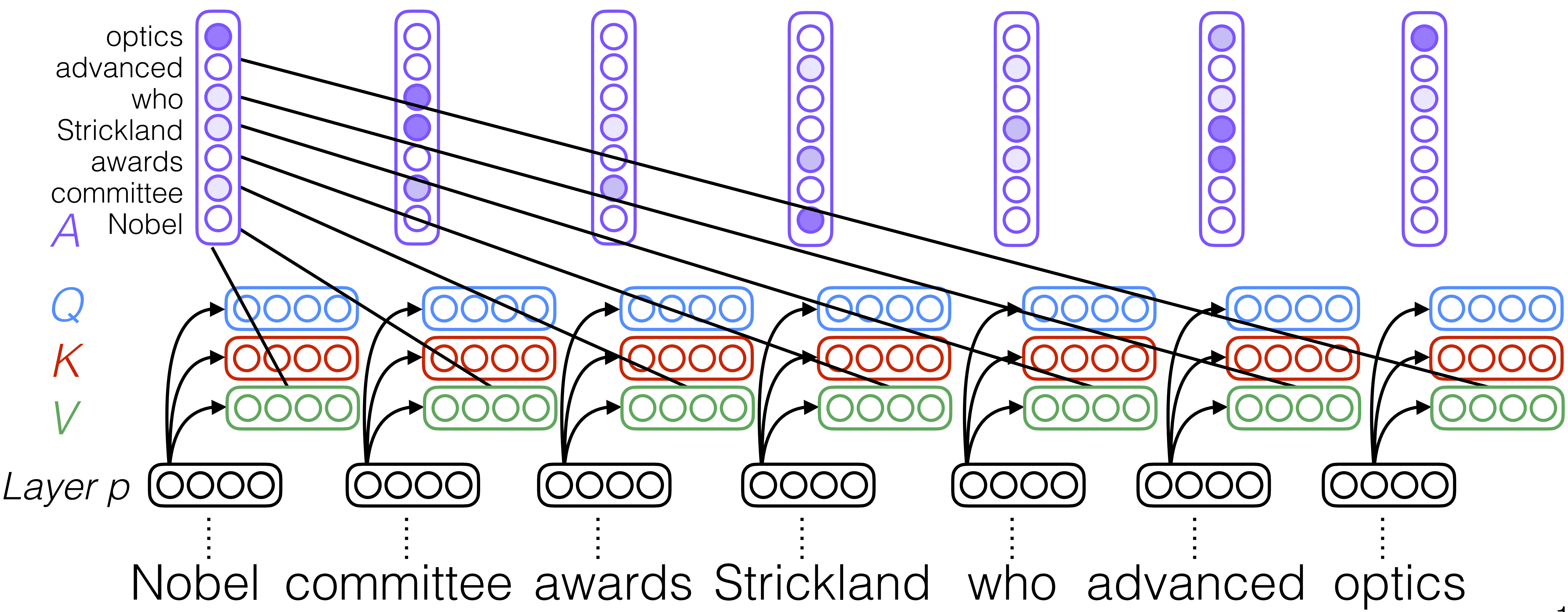
$$\boxed{\mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})}V$$



Self-attention

[Vaswani et al. 2017. slide: Emma Strubell]

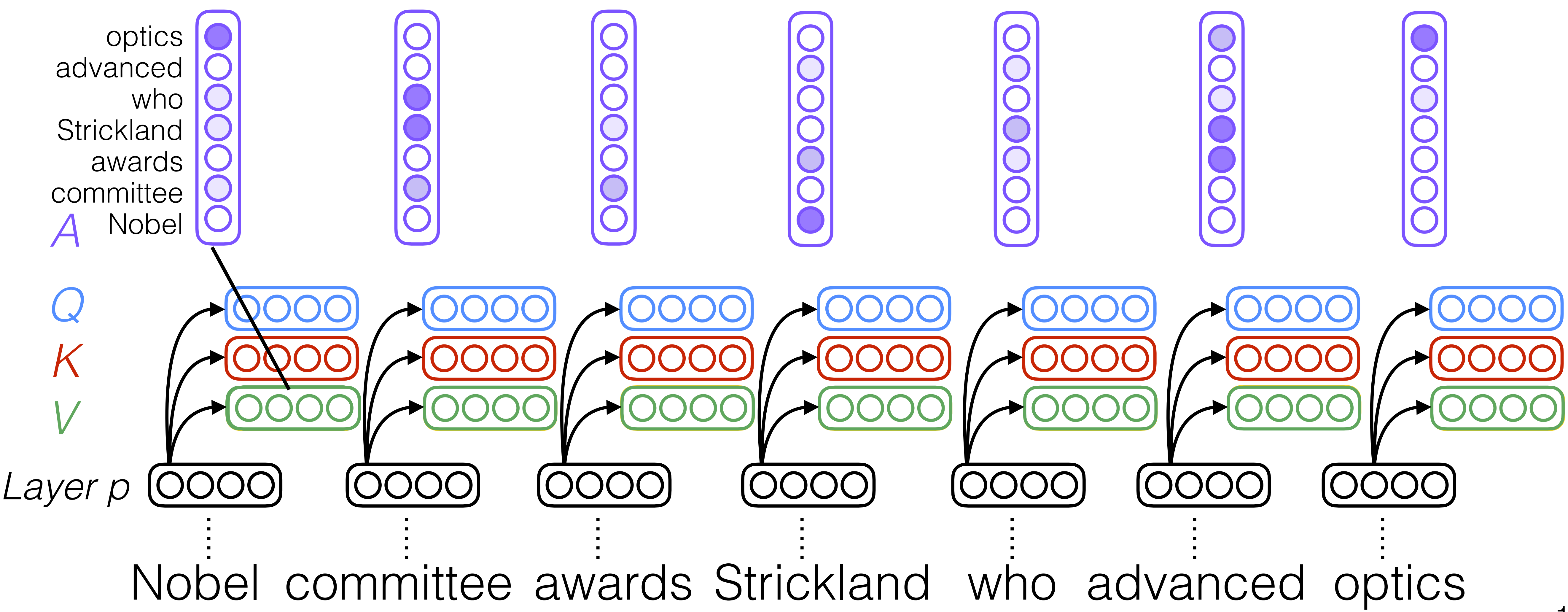
$$\boxed{\mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V}$$



Self-attention

[Vaswani et al. 2017.
slide: Emma Strubell]

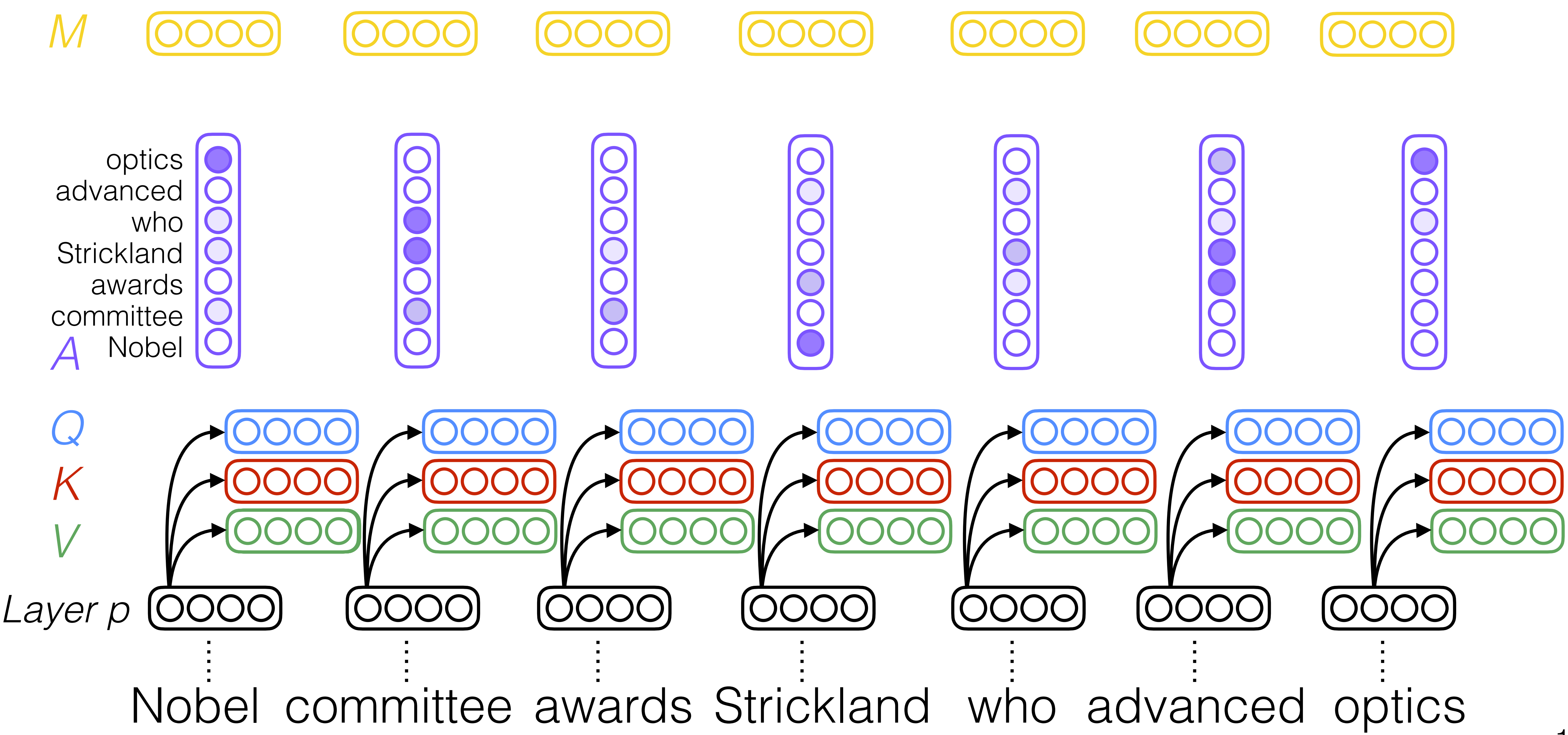
$$\mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Self-attention

[Vaswani et al. 2017.
slide: Emma Strubell]

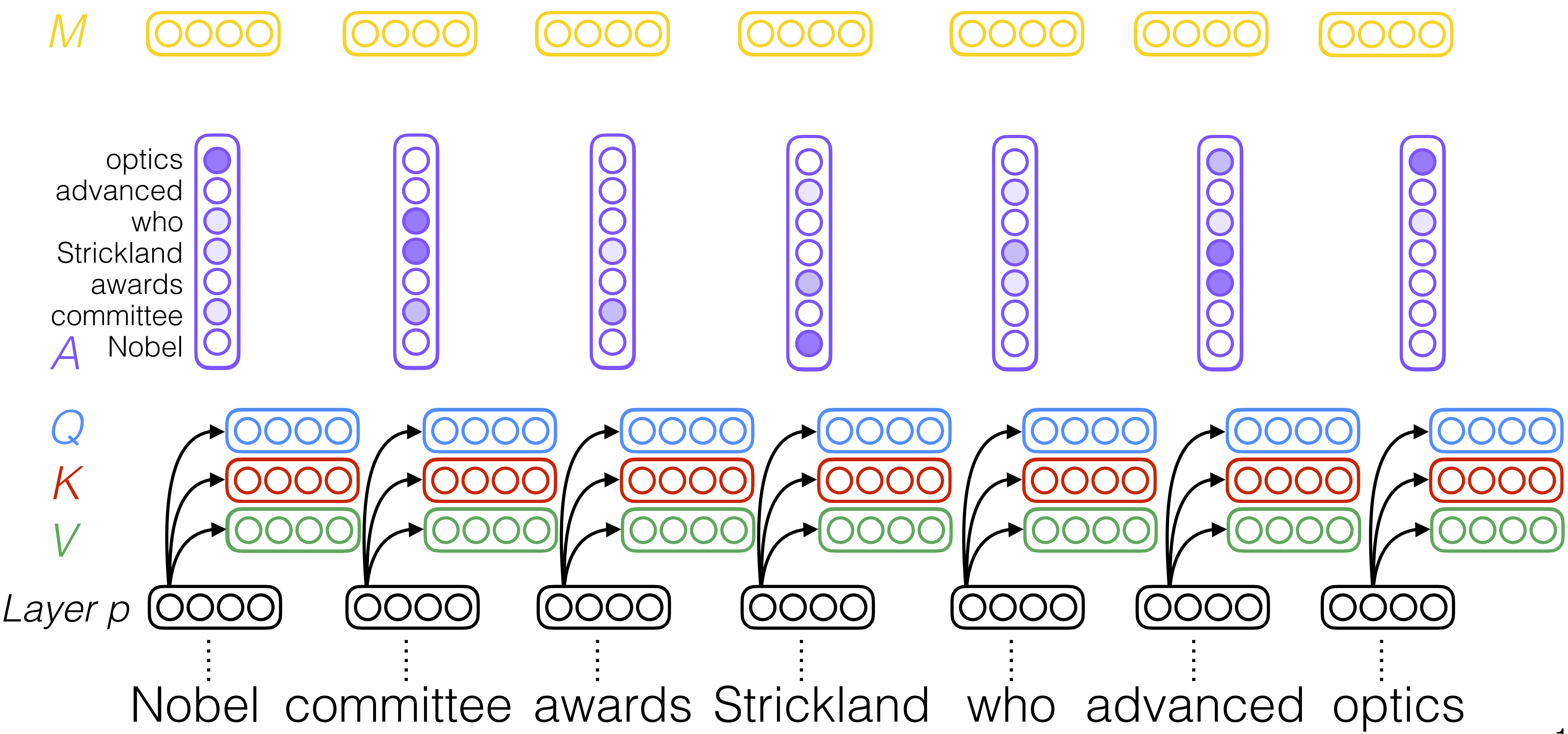
$$\boxed{\mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V}$$



Self-attention

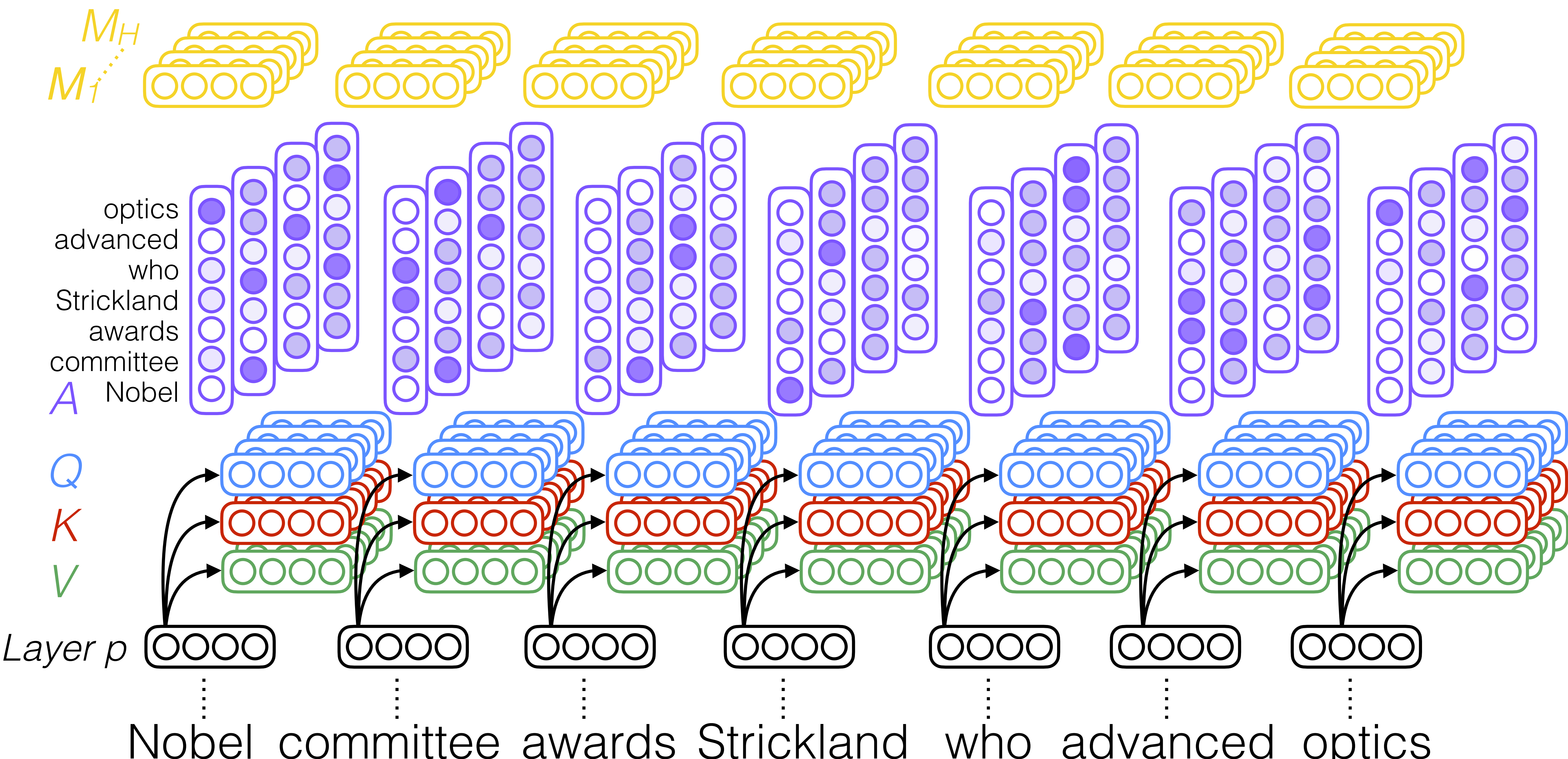
[Vaswani et al. 2017. slide: Emma Strubell]

$$\boxed{\mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V}$$



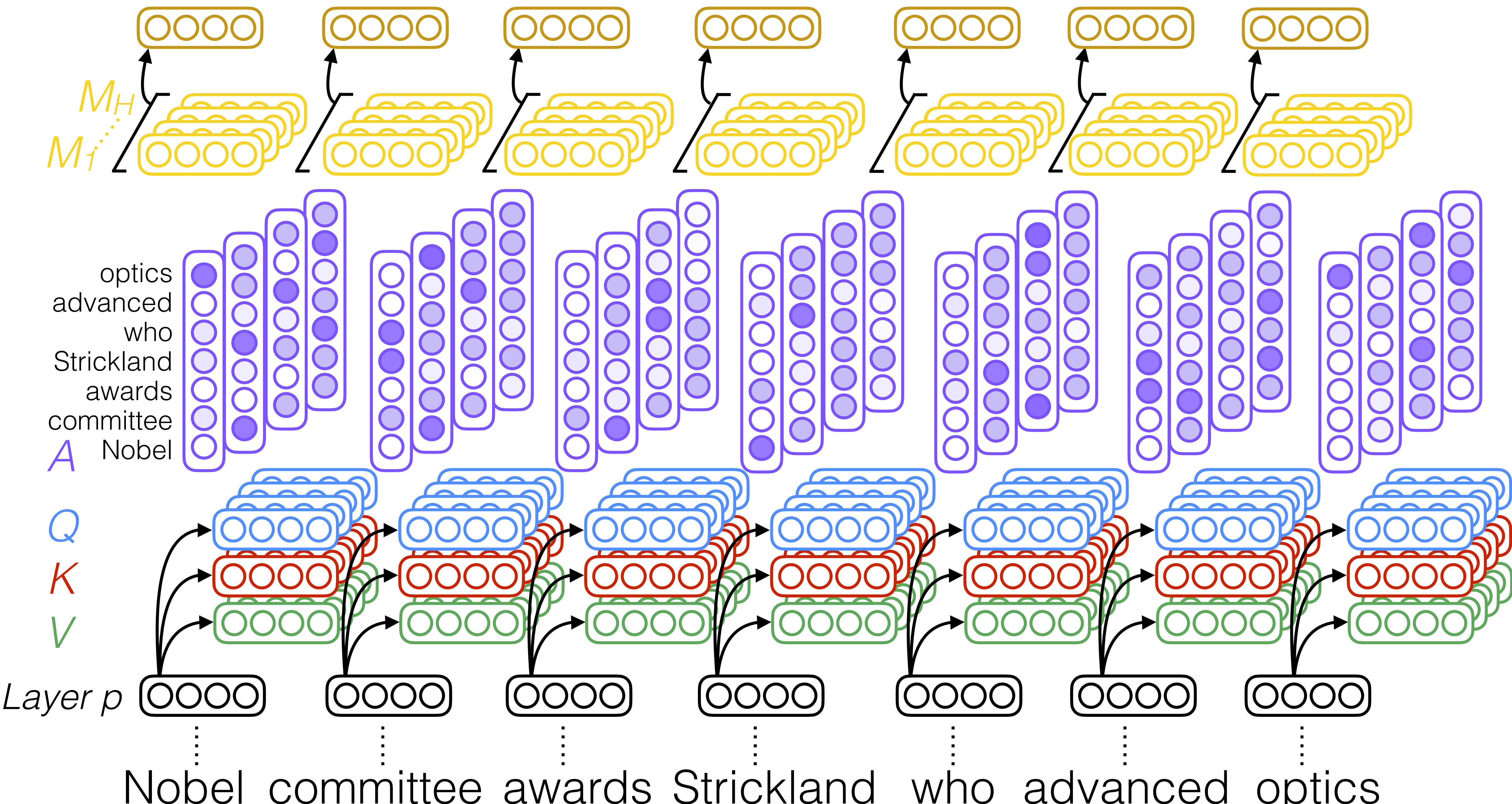
Multi-head self-attention

[Vaswani et al. 2017. slide: Emma Strubell]



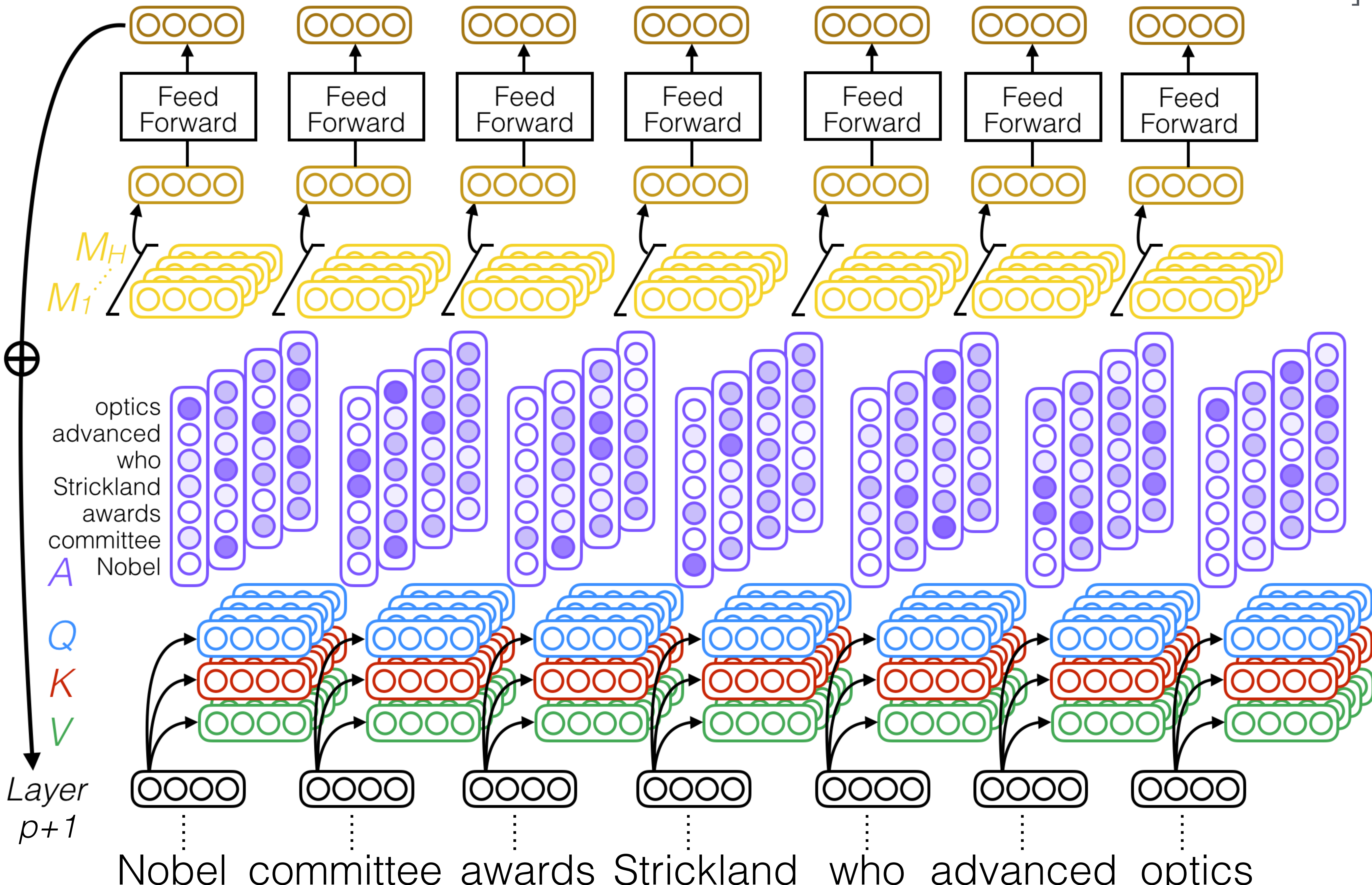
Multi-head self-attention

[Vaswani et al. 2017. slide: Emma Strubell]



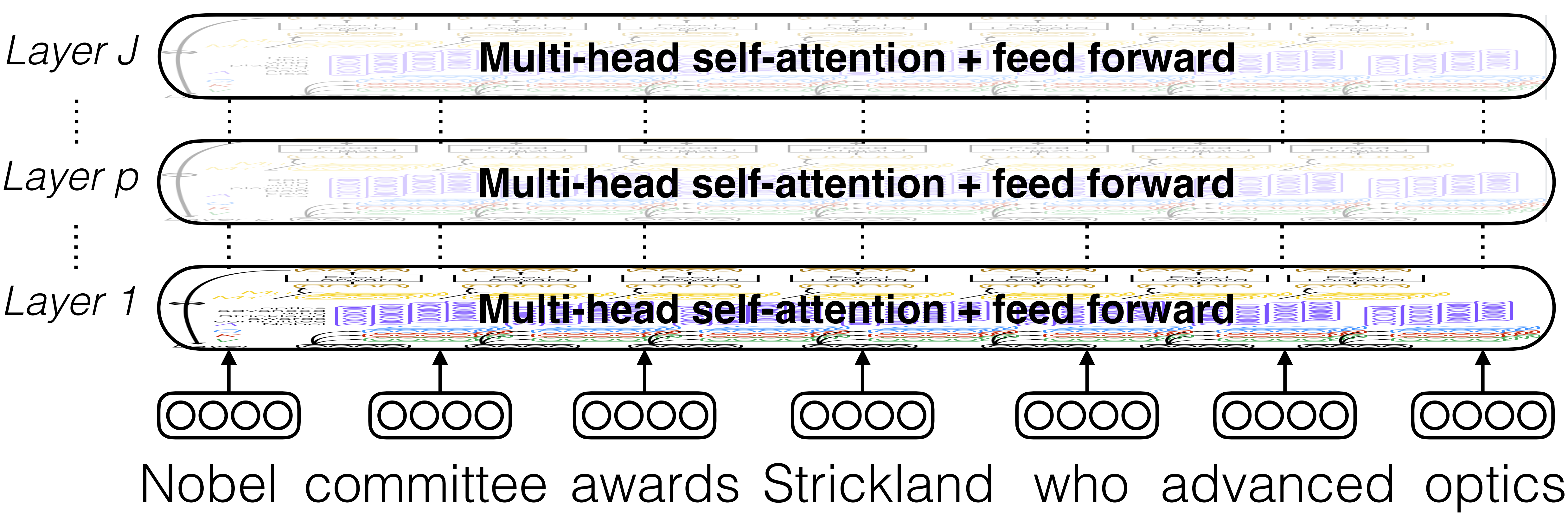
Multi-head self-attention

[Vaswani et al. 2017. slide: Emma Strubell]



Multi-head self-attention

[Vaswani et al. 2017. slide: Emma Strubell]



- Parallelized training (switch slides)

Transformers

Parallelizing Attention Computation

[slide: SLP3]

Parallelizing computation using X

For attention/transformer block we've been computing a **single** output at a **single** time step *i* in a **single** residual stream.

But we can pack the N tokens of the input sequence into a single matrix **X** of size $[N \times d]$.

Each row of X is the embedding of one token of the input.

X can have 1K-32K rows, each of the dimensionality of the embedding *d* (the **model dimension**)

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^{\mathbf{Q}};\ \ \mathbf{K} = \mathbf{X}\mathbf{W}^{\mathbf{K}};\ \ \mathbf{V} = \mathbf{X}\mathbf{W}^{\mathbf{V}}$$

[slide: SLP3]

QK^T

Now can do a single matrix multiply to combine Q and K^T

q1•k1	**q1•k2**	**q1•k3**	**q1•k4**
q2•k1	**q2•k2**	**q2•k3**	**q2•k4**
q3•k1	**q3•k2**	**q3•k3**	**q3•k4**
q4•k1	**q4•k2**	**q4•k3**	**q4•k4**

N

N

[slide: SLP3]

Parallelizing attention

- Scale the scores, take the softmax, and then multiply the result by V resulting in a matrix of shape $N \times d$
  - An attention vector for each input token

$$\mathbf{A} = \text{softmax}\left(\text{mask}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_k}}\right)\right)\mathbf{V}$$

[slide: SLP3]

Masking out the future

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

- What is this mask function?
  QK^T has a score for each query dot every key, *including those that follow the query*.
- Guessing the next word is pretty simple if you already know it!

[slide: SLP3]

Masking out the future

$$\mathbf{A} = \text{softmax}\left(\text{mask}\left(\frac{\mathbf{QK}^\intercal}{\sqrt{d_k}}\right)\right)\mathbf{V}$$

Add $-\infty$ to cells in upper triangle

The softmax will turn it to 0

N

q1•k1	$-\infty$	$-\infty$	$-\infty$
q2•k1	**q2•k2**	$-\infty$	$-\infty$
q3•k1	**q3•k2**	**q3•k3**	$-\infty$
q4•k1	**q4•k2**	**q4•k3**	**q4•k4**

N

[slide: SLP3]

Another point: Attention is quadratic in length

$$\mathbf{A} = \text{softmax}\left(\text{mask}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_k}}\right)\right)\mathbf{V}$$

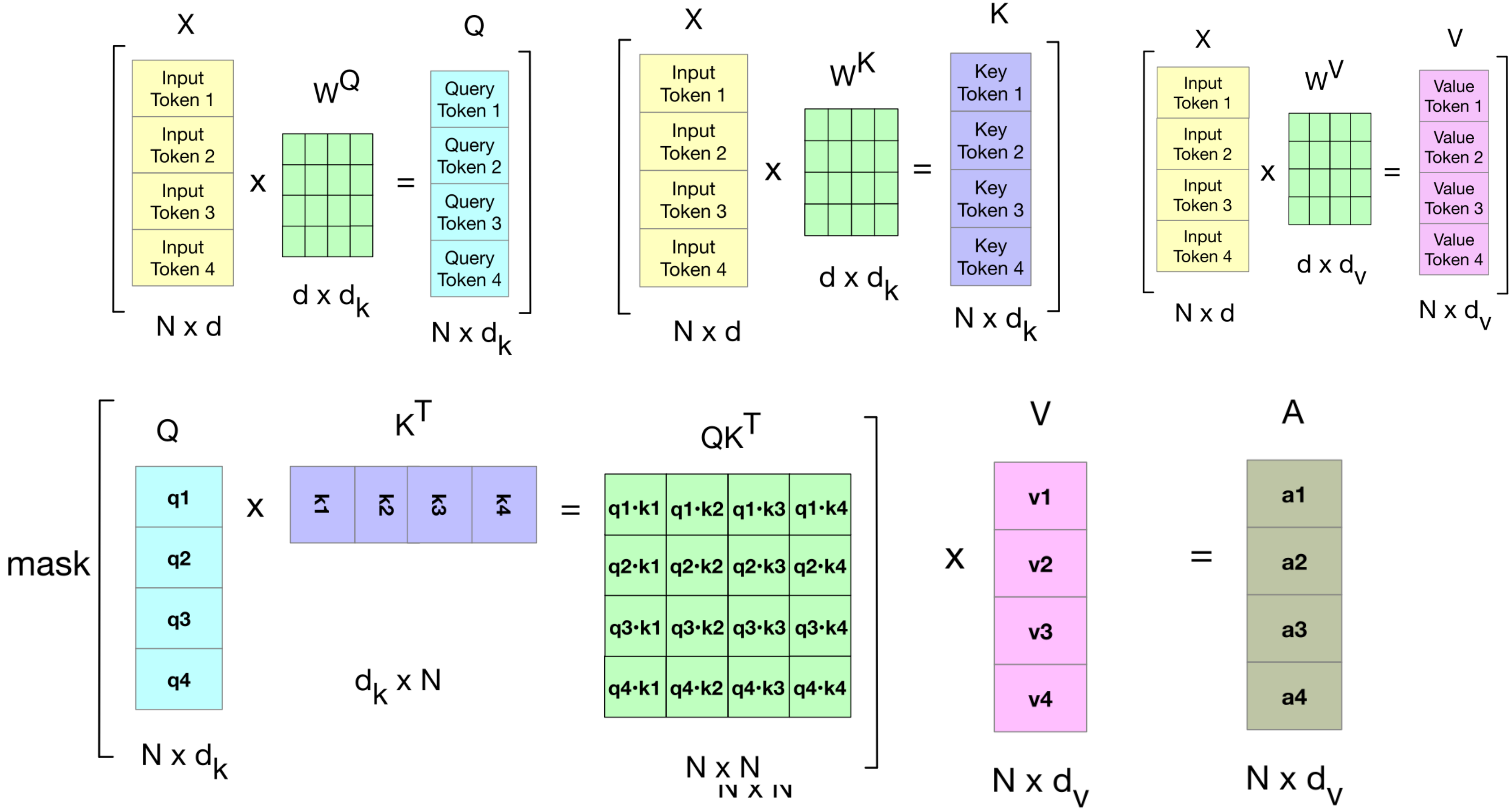
N

q1•k1	$-\infty$	$-\infty$	$-\infty$
q2•k1	q2•k2	$-\infty$	$-\infty$
q3•k1	q3•k2	q3•k3	$-\infty$
q4•k1	q4•k2	q4•k3	q4•k4

N

[slide: SLP3]

Attention again



[slide: SLP3]

Parallelizing Multi-head Attention

$$\mathbf{Q^i} = \mathbf{XW^{Qi}}\ ;\ \ \mathbf{K^i} = \mathbf{XW^{Ki}}\ ;\ \ \mathbf{V^i} = \mathbf{XW^{Vi}}$$

$$\mathbf{head}_i = \mathrm{SelfAttention}(\mathbf{Q^i}, \mathbf{K^i}, \mathbf{V^i}) = \mathrm{softmax}\left(\frac{\mathbf{Q^i}\mathbf{K^{i\top}}}{\sqrt{d_k}}\right)\mathbf{V^i}$$

$$\mathrm{MultiHeadAttention}(\mathbf{X}) = (\mathbf{head}_1 \oplus \mathbf{head}_2 \ldots \oplus \mathbf{head}_h)\mathbf{W^O}$$

[slide: SLP3]

Parallelizing Multi-head Attention

$$\begin{aligned} \mathbf{O} &= \text{LayerNorm}(\mathbf{X} + \text{MultiHeadAttention}(\mathbf{X})) \\ \mathbf{H} &= \text{LayerNorm}(\mathbf{O} + \text{FFN}(\mathbf{O})) \end{aligned}$$

or

$$\begin{aligned} \mathbf{T}^1 &= \text{MultiHeadAttention}(\mathbf{X}) \\ \mathbf{T}^2 &= \mathbf{X} + \mathbf{T}^1 \\ \mathbf{T}^3 &= \text{LayerNorm}(\mathbf{T}^2) \\ \mathbf{T}^4 &= \text{FFN}(\mathbf{T}^3) \\ \mathbf{T}^5 &= \mathbf{T}^4 + \mathbf{T}^3 \\ \mathbf{H} &= \text{LayerNorm}(\mathbf{T}^5) \end{aligned}$$

[slide: SLP3]

Transformers

Parallelizing Attention Computation

[slide: SLP3]

Transformers

Input and output: Position embeddings and the Language Model Head

[slide: SLP3]

Token and Position Embeddings

The matrix X (of shape [$N \times d$]) has an embedding for each word in the context.

This embedding is created by adding two distinct embedding for each input

- token embedding
- positional embedding

[slide: SLP3]

Token Embeddings

Embedding matrix E has shape $[|V| \times d]$.

- One row for each of the $|V|$ tokens in the vocabulary.
- Each word is a row vector of d dimensions

Given: string "*Thanks for all the*"

1. Tokenize with BPE and convert into vocab indices

w = [5,4000,10532,2224]

2. Select the corresponding rows from E, each row an embedding

- (row 5, row 4000, row 10532, row 2224).

[slide: SLP3]

Position Embeddings

There are many methods, but we'll just describe the simplest: absolute position.

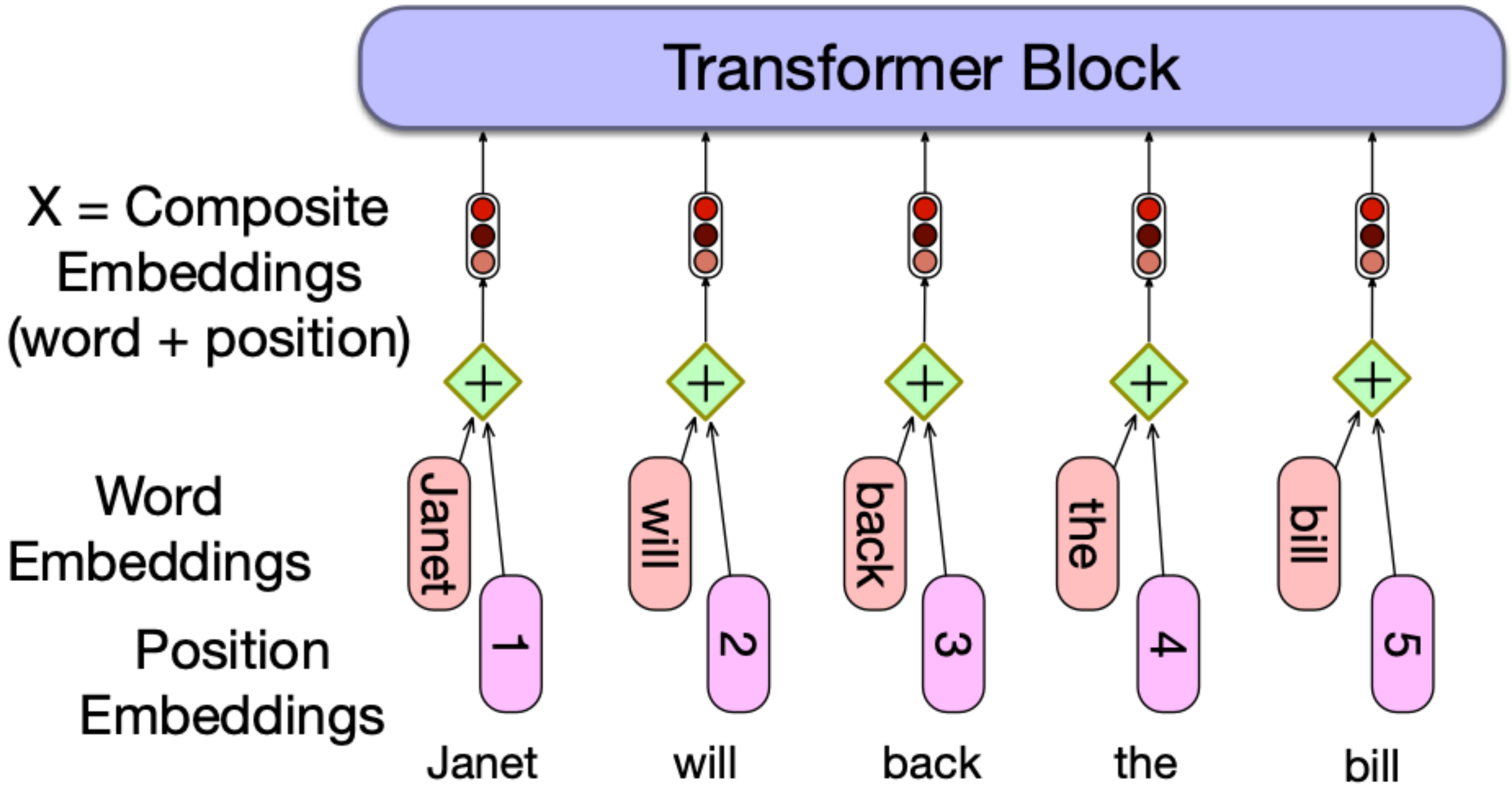
Goal: learn a position embedding matrix Epos of shape $[1 \times N]$.

Start with randomly initialized embeddings

- one for each integer up to some maximum length.
- i.e., just as we have an embedding for token *fish*, we'll have an embedding for position 3 and position 17.
- As with word embeddings, these position embeddings are learned along with other parameters during training.

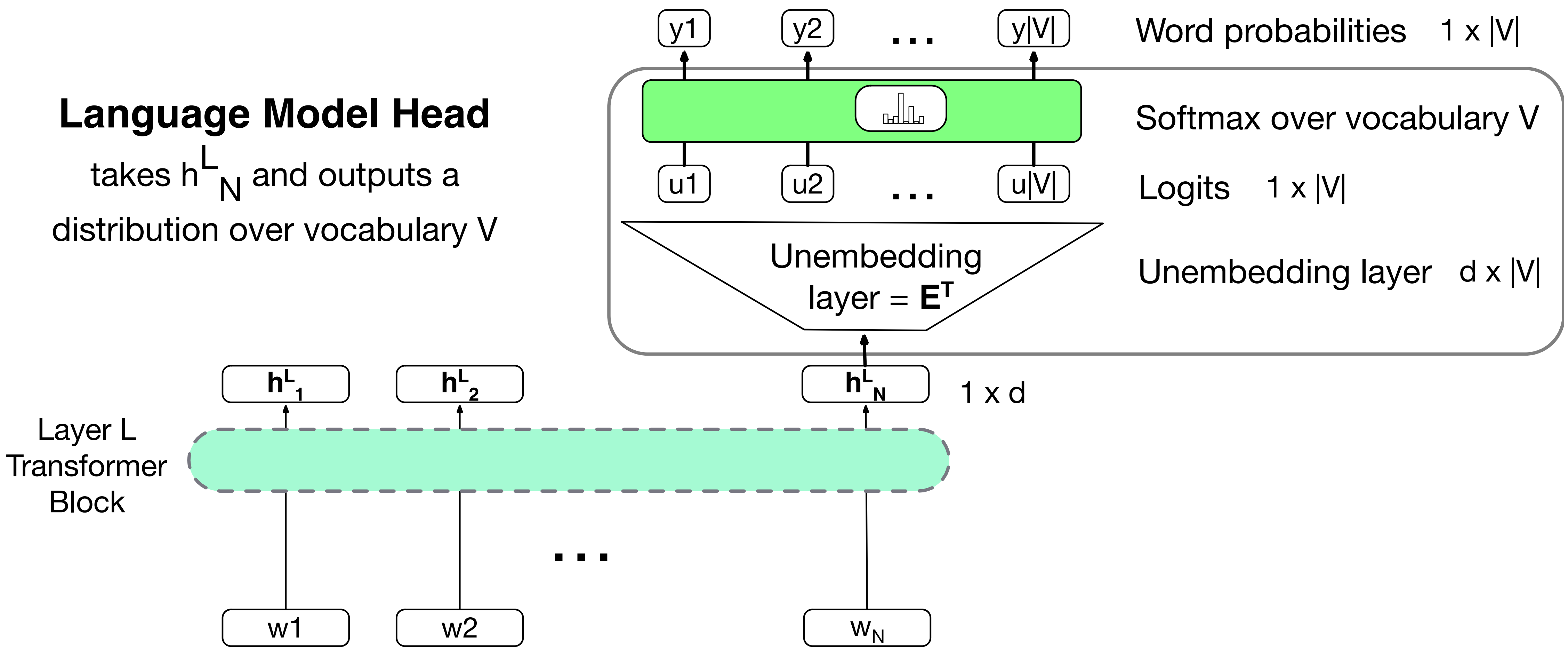
[slide: SLP3]

Each x is just the sum of word and position embeddings



[slide: SLP3]

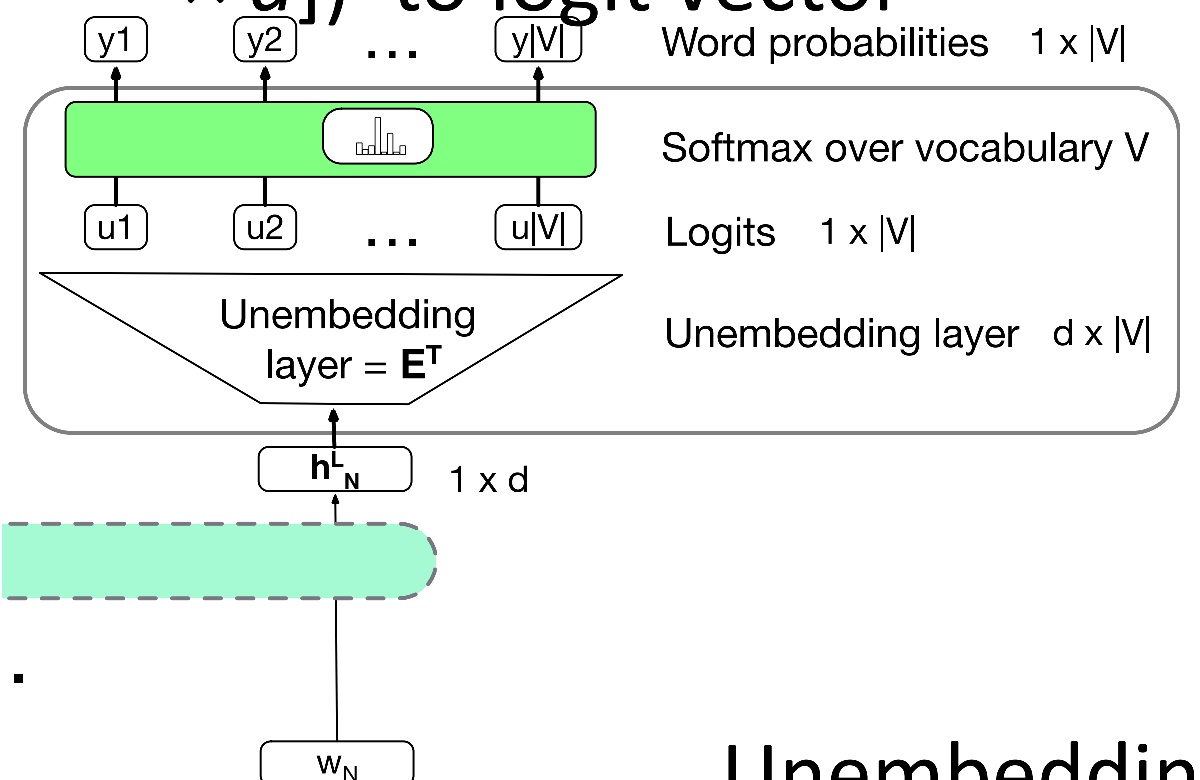
Language modeling head



[slide: SLP3]

Language modeling head

Unembedding layer: linear layer projects from h^L_N (shape [1 × d]) to logit vector



Why "unembedding"?
Tied to $\mathbf{E}^T$

Weight tying, we use the same weights for two different matrices

Unembedding layer maps from an embedding to a 1x|V| vector of logits

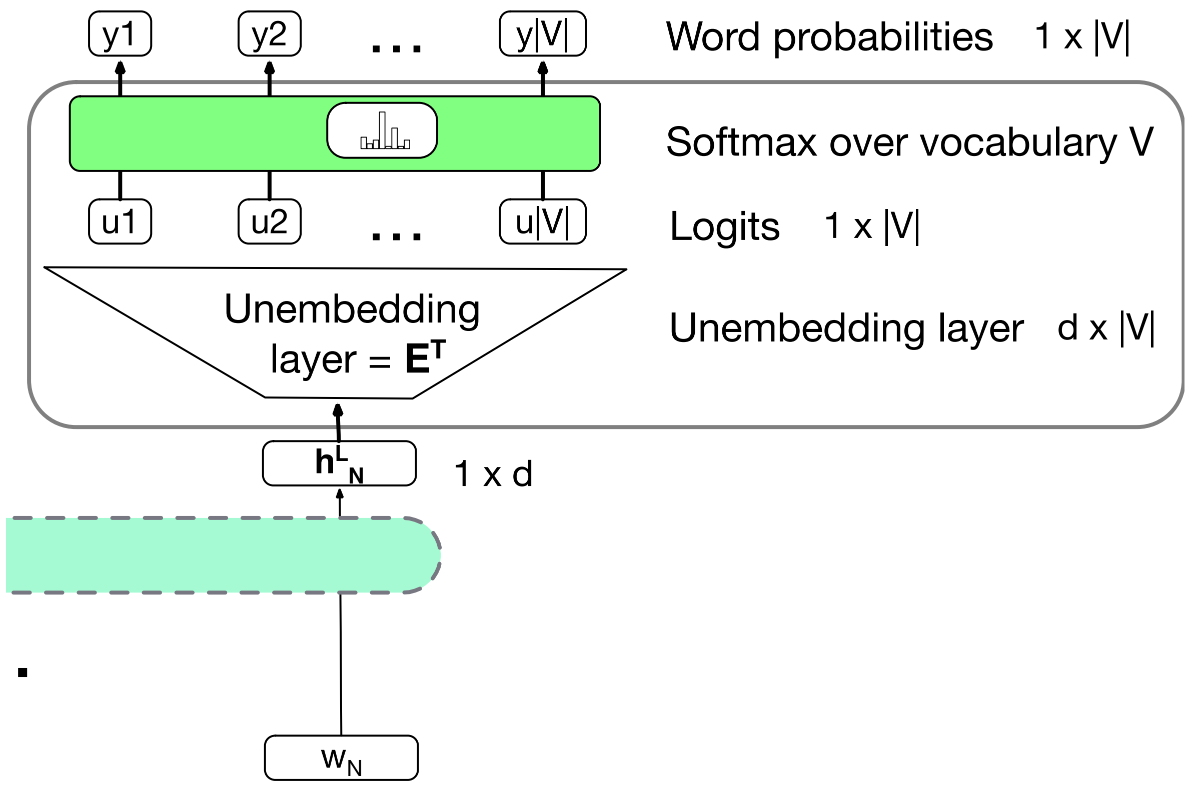
[slide: SLP3]

Language modeling head

Logits, the score vector u

One score for each of the $|V|$ possible words in the vocabulary V. Shape $1 \times |V|$.

Softmax turns the logits into probabilities over vocabulary. Shape $1 \times |V|$.



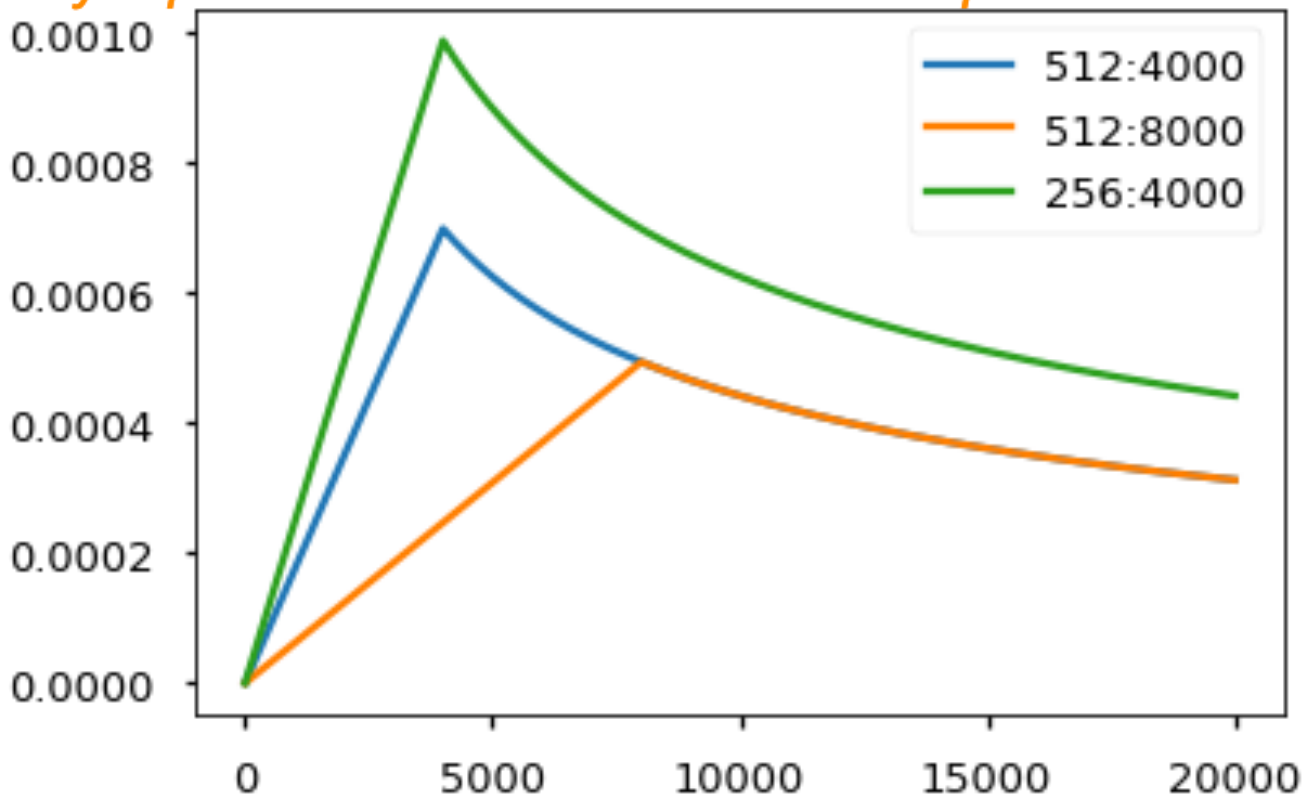
$$u = h_N^L \, E^T$$
$$y = \text{softmax}(u)$$

[slide: SLP3]

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.*



- Training instability is a notorious issue
  - Esp. with many layers, >10 or >20
- Yet something is going right. Not clear why!

Byte pair encoding (BPE)

- Deal with rare words / large vocabulary by instead using ***subword*** tokenization

system	sentence
source	health research institutes
reference	Gesundheitsforschungsinstitute
WDict	Forschungsinstitute
C2-50k	Fo|rs|ch|un|gs|in|st|it|ut|io|ne|n
BPE-60k	Gesundheits|forsch|ungsinstitu|ten
BPE-J90k	Gesundheits|forsch|ungsin|stitute
source	asinine situation
reference	dumme Situation
WDict	asinine situation $\rightarrow$ UNK $\rightarrow$ asinine
C2-50k	as|in|in|e situation $\rightarrow$ As|in|en|si|tu|at|io|n
BPE-60k	as|in|ine situation $\rightarrow$ A|in|line-|Situation
BPE-J90K	as|in|ine situation $\rightarrow$ As|in|in-|Situation

Sennrich et al., ACL 2016