Introduction to Transformer (Part I)

Large Language Models: Introduction and Recent Advances

ELL881 · AIL821

Tanmoy Chakraborty
Associate Professor, IIT Delhi
https://tanmoychak.com/

Slides are adopted from Sergey Levine and Emma Strubell

GitHub launches **GitHub Models** !!

Developers can now access and experiment with most LLMs directly on GitHub

Released on
August 1, 2024

GitHub Blog

From **Llama 3.1**, to **GPT-4o** and **GPT-4o mini**, to **Phi 3** or **Mistral Large 2**, **GitHub Models allows us to access each model via a built-in playground** that lets you test different prompts and model parameters, **for free, right in GitHub**

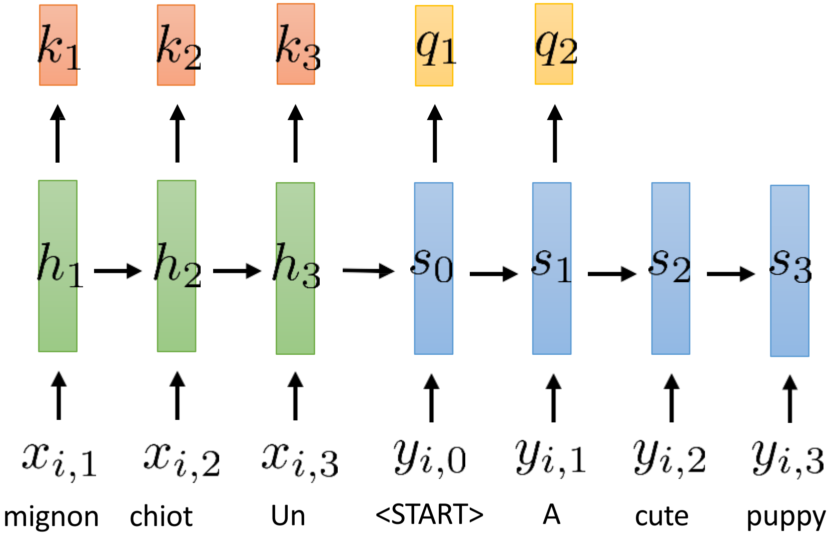
They also have a **glide path to bring the models to the developer environment** in **Codespaces** and **VS Code**

Privacy: **No prompts or outputs** in GitHub Models **will be shared with model providers**, nor used to train or improve the models.

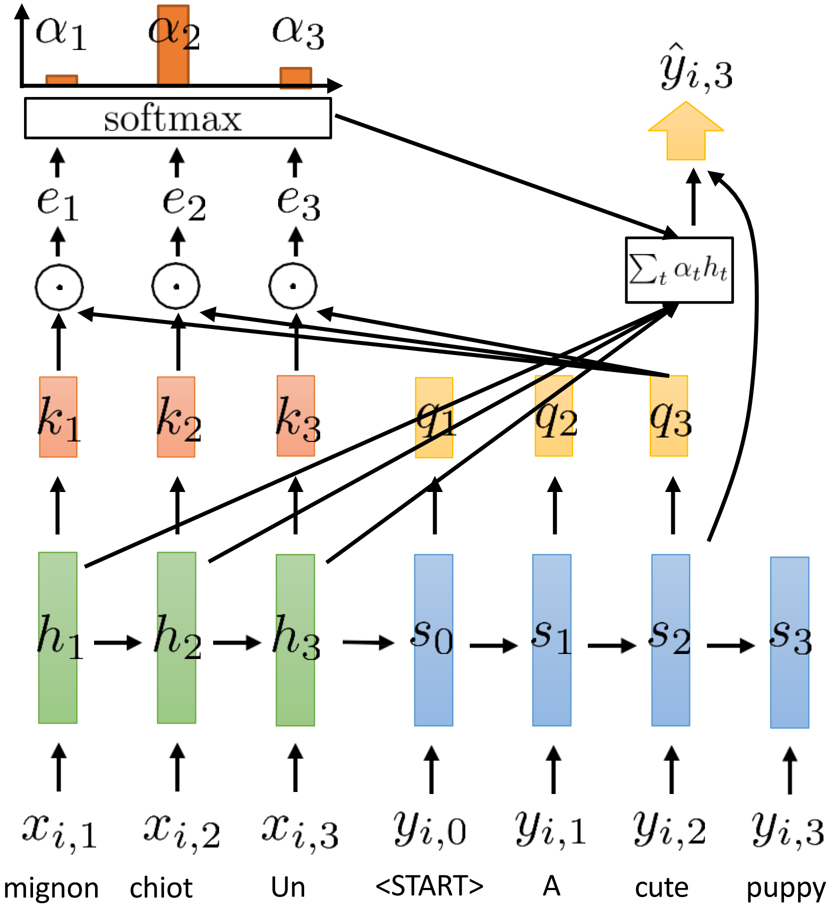
Is Attention All We Need?

Slides by Sergey Levine

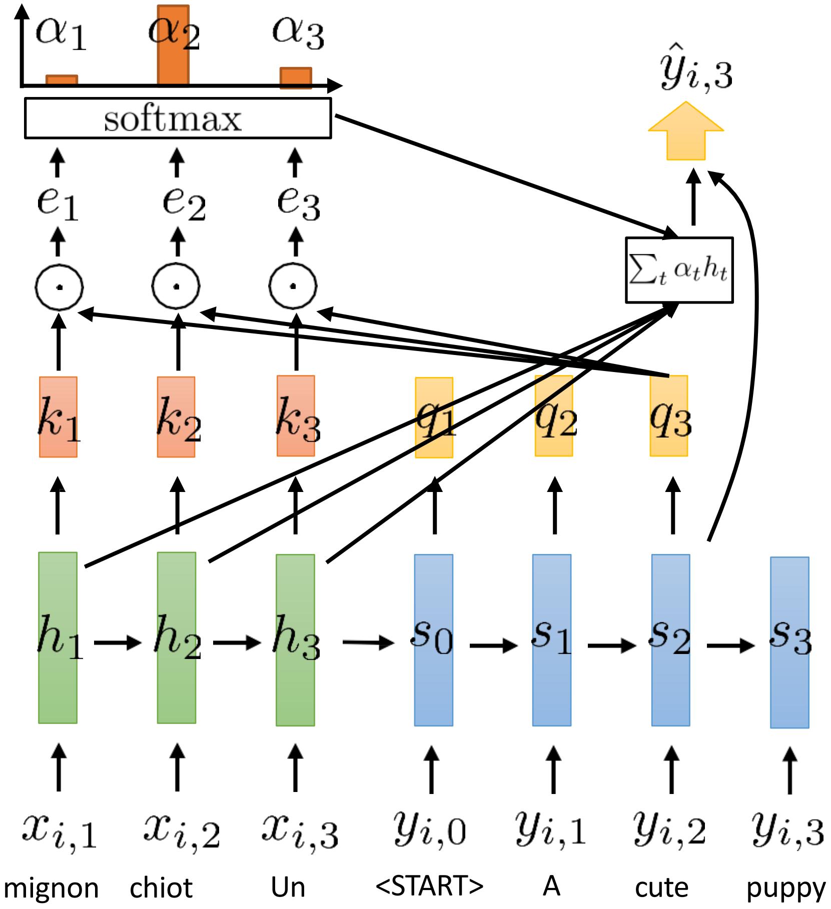
Recap: Attention



Recap: Attention



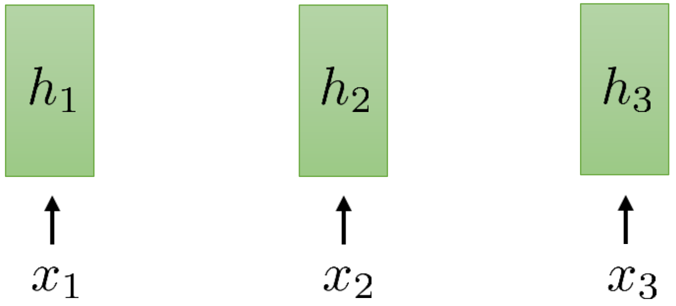
Recap: Attention



- If we have **attention**, do we even need recurrent connections?
- Can we transform our RNN into a **purely attention-based model**?
- Attention can access all time steps simultaneously, potentially doing everything that recurrence can, and even more. However, this approach presents some challenges:

The encoder lacks temporal dependencies at all!

Self-Attention



this is *not* a recurrent model!

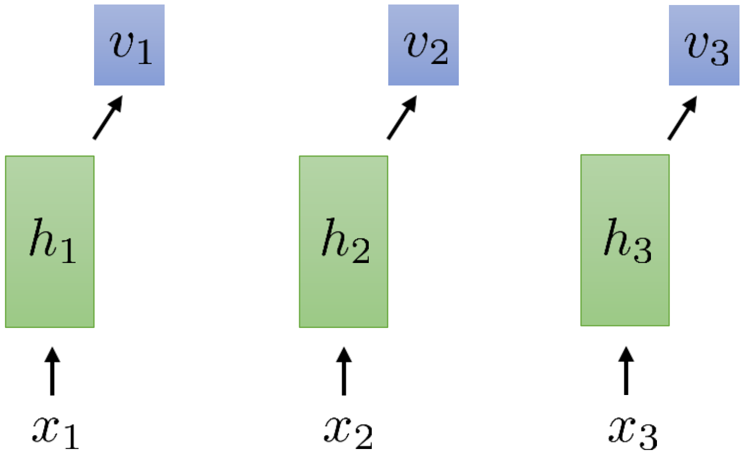
but still weight sharing:

$$h_t = \sigma(Wx_t + b)$$

shared weights at all time steps

(or any other nonlinear function)

Self-Attention



this is *not* a recurrent model!

but still weight sharing:

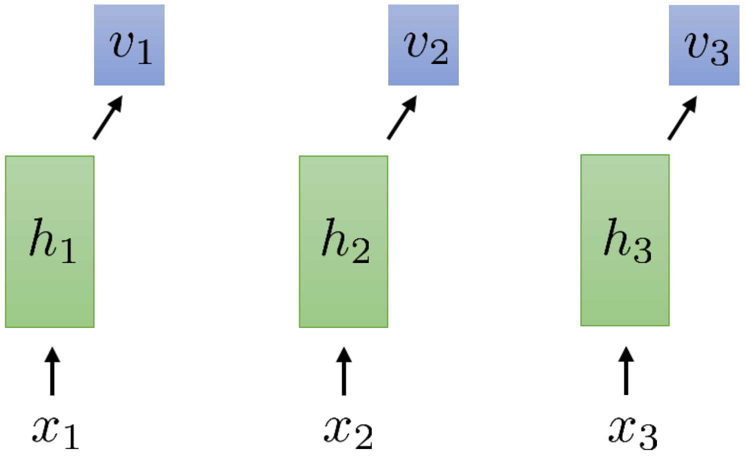
$$h_t = \sigma(Wx_t + b)$$

shared weights at all time steps

(or any other nonlinear function)

Self-Attention

$v_t = v(h_t)$ before just had $v(h_t) = h_t$, now e.g. $v(h_t) = W_v h_t$



this is *not* a recurrent model!
but still weight sharing:

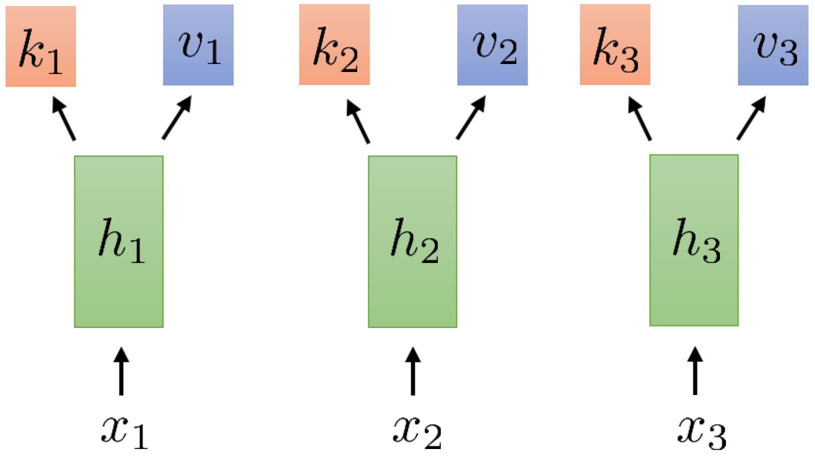
$$h_t = \sigma(W x_t + b)$$

shared weights at all time steps

(or any other nonlinear function)

Self-Attention

$v_t = v(h_t)$ before just had $v(h_t) = h_t$, now e.g. $v(h_t) = W_v h_t$



this is *not* a recurrent model!

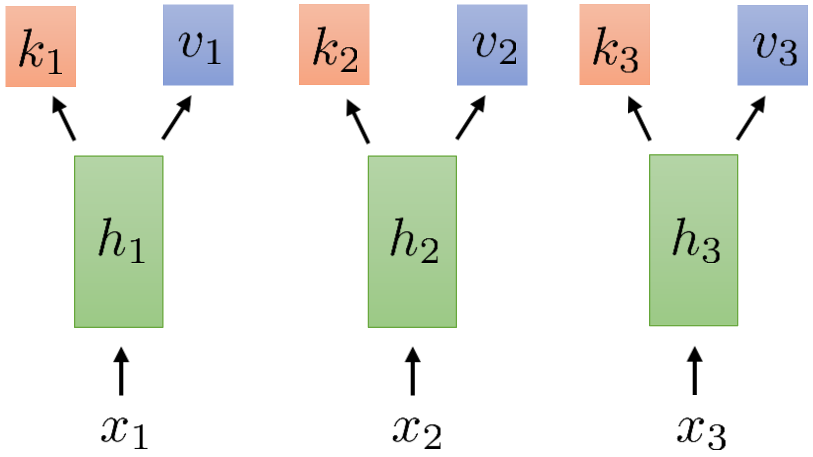
but still weight sharing:

$$h_t = \sigma(W x_t + b)$$

shared weights at all time steps

(or any other nonlinear function)

Self-Attention



$v_t = v(h_t)$ before just had $v(h_t) = h_t$, now e.g. $v(h_t) = W_v h_t$

$k_t = k(h_t)$ (just like before) e.g., $k_t = W_k h_t$

this is *not* a recurrent model!

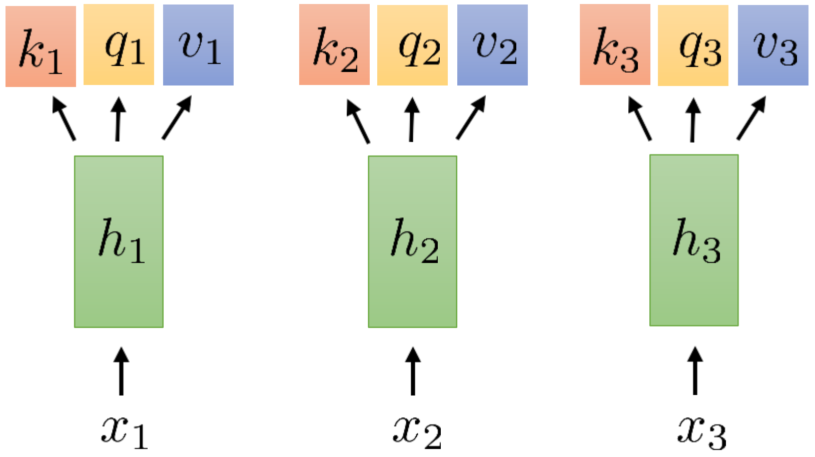
but still weight sharing:

$$h_t = \sigma(Wx_t + b)$$

shared weights at all time steps

(or any other nonlinear function)

Self-Attention



$v_t = v(h_t)$ before just had $v(h_t) = h_t$, now e.g. $v(h_t) = W_v h_t$

$k_t = k(h_t)$ (just like before) e.g., $k_t = W_k h_t$

this is *not* a recurrent model!

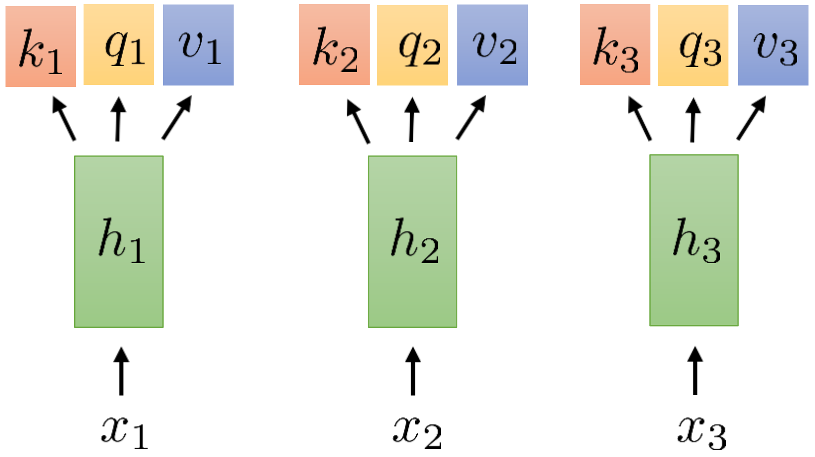
but still weight sharing:

$$h_t = \sigma(W x_t + b)$$

shared weights at all time steps

(or any other nonlinear function)

Self-Attention



$v_t = v(h_t)$ before just had $v(h_t) = h_t$, now e.g. $v(h_t) = W_v h_t$

$k_t = k(h_t)$ (just like before) e.g., $k_t = W_k h_t$

$q_t = q(h_t)$ e.g., $q_t = W_q h_t$

this is *not* a recurrent model!

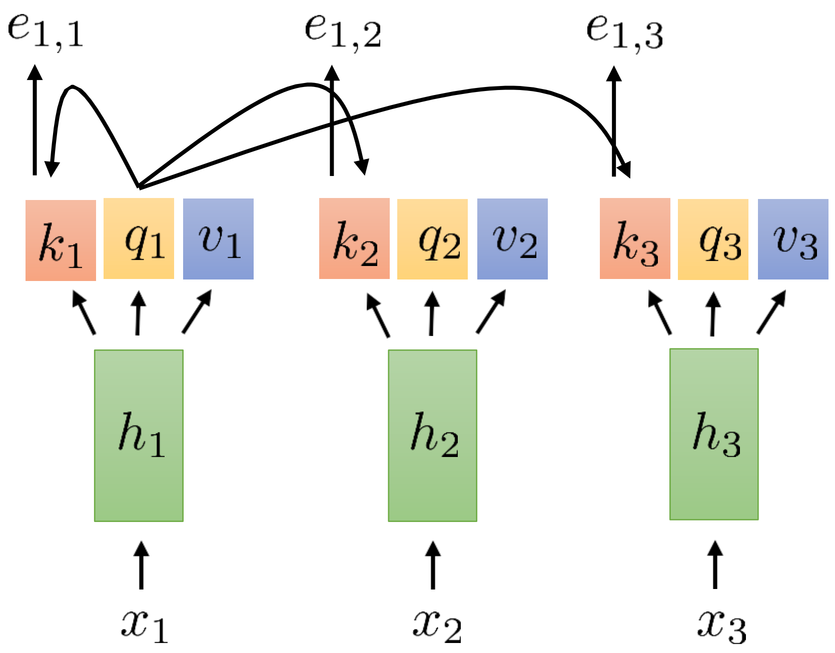
but still weight sharing:

$$h_t = \sigma(Wx_t + b)$$

shared weights at all time steps

(or any other nonlinear function)

Self-Attention



$v_t = v(h_t)$ before just had $v(h_t) = h_t$, now e.g. $v(h_t) = W_v h_t$

$k_t = k(h_t)$ (just like before) e.g., $k_t = W_k h_t$

$q_t = q(h_t)$ e.g., $q_t = W_q h_t$

this is *not* a recurrent model!

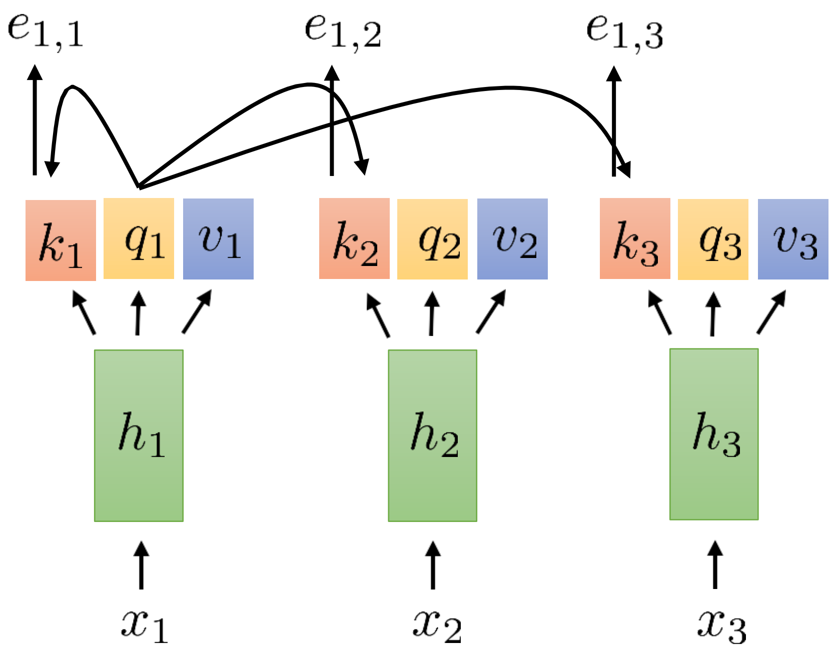
but still weight sharing:

$$h_t = \sigma(Wx_t + b)$$

shared weights at all time steps

(or any other nonlinear function)

Self-Attention



$e_{l,t} = q_l \cdot k_t$

$v_t = v(h_t)$ before just had $v(h_t) = h_t$, now e.g. $v(h_t) = W_v h_t$

$k_t = k(h_t)$ (just like before) e.g., $k_t = W_k h_t$

$q_t = q(h_t)$ e.g., $q_t = W_q h_t$

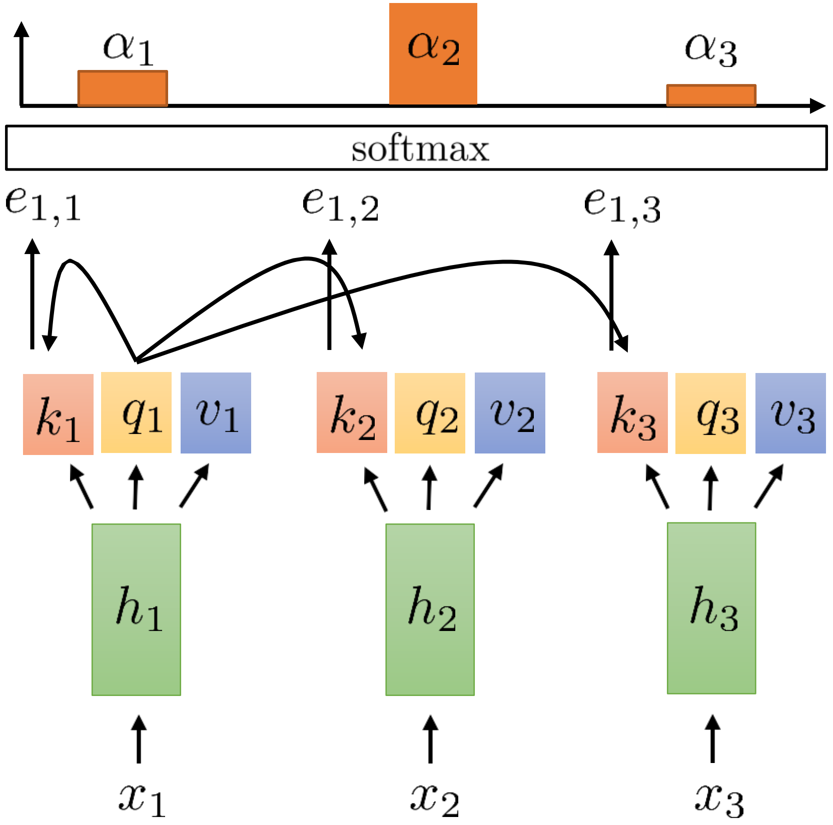
this is *not* a recurrent model!

but still weight sharing:

$$h_t = \sigma(\underset{\nwarrow \text{shared weights at all time steps}}{W} x_t + b)$$

(or any other nonlinear function)

Self-Attention



$e_{l,t} = q_l \cdot k_t$

$v_t = v(h_t)$ before just had $v(h_t) = h_t$, now e.g. $v(h_t) = W_v h_t$

$k_t = k(h_t)$ (just like before) e.g., $k_t = W_k h_t$

$q_t = q(h_t)$ e.g., $q_t = W_q h_t$

this is *not* a recurrent model!

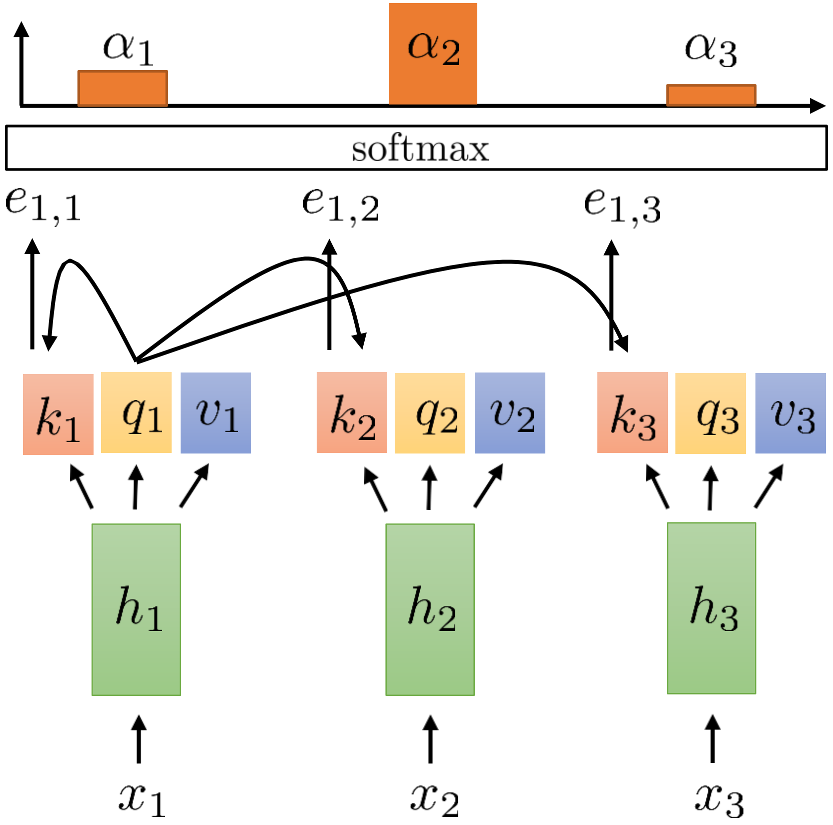
but still weight sharing:

$$h_t = \sigma(W x_t + b)$$

shared weights at all time steps

(or any other nonlinear function)

Self-Attention



$$\alpha_{l,t} = \exp(e_{l,t}) / \sum_{t'} \exp(e_{l,t'})$$

$$e_{l,t} = q_l \cdot k_t$$

$v_t = v(h_t)$ before just had $v(h_t) = h_t$, now e.g. $v(h_t) = W_v h_t$

$k_t = k(h_t)$ (just like before) e.g., $k_t = W_k h_t$

$q_t = q(h_t)$ e.g., $q_t = W_q h_t$

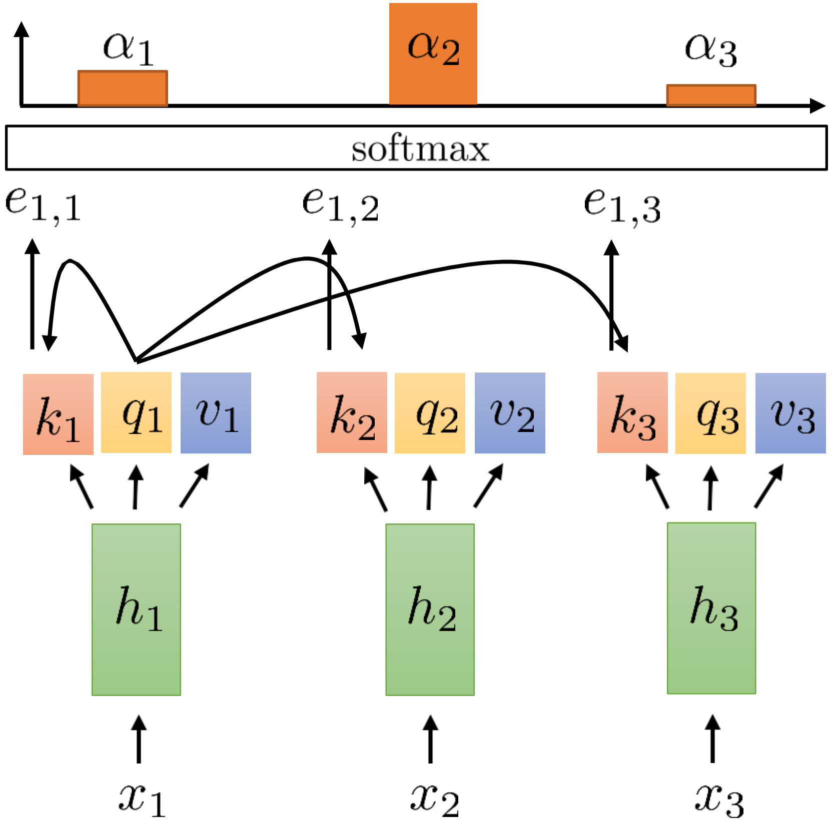
this is *not* a recurrent model!

but still weight sharing:

$$h_t = \sigma(\underset{\nwarrow \text{shared weights at all time steps}}{W} x_t + b)$$

(or any other nonlinear function)

Self-Attention



$\alpha_{l,t} = \exp(e_{l,t}) / \sum_{t'} \exp(e_{l,t'})$

$e_{l,t} = q_l \cdot k_t$

$v_t = v(h_t)$ before just had $v(h_t) = h_t$, now e.g. $v(h_t) = W_v h_t$

$k_t = k(h_t)$ (just like before) e.g., $k_t = W_k h_t$

$q_t = q(h_t)$ e.g., $q_t = W_q h_t$

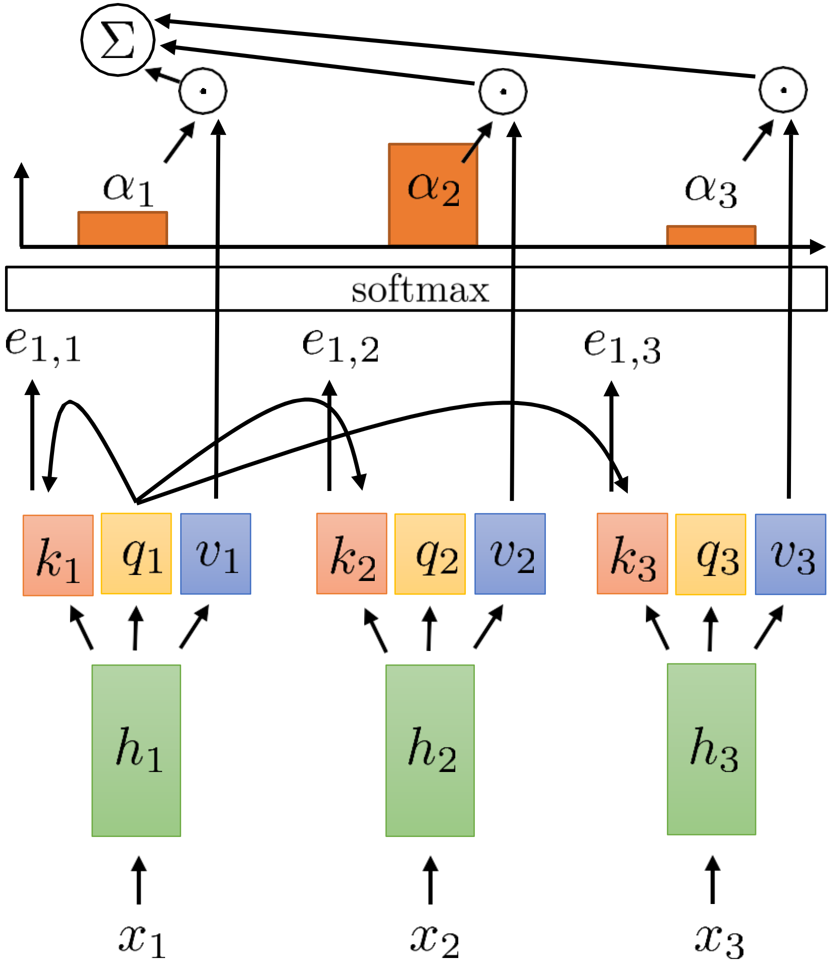
this is *not* a recurrent model!

but still weight sharing:

$$h_t = \sigma(\underset{\nwarrow \text{shared weights at all time steps}}{W} x_t + b)$$

(or any other nonlinear function)

Self-Attention



$$\alpha_{l,t} = \exp(e_{l,t}) / \sum_{t'} \exp(e_{l,t'})$$

$$e_{l,t} = q_l \cdot k_t$$

$v_t = v(h_t)$ before just had $v(h_t) = h_t$, now e.g. $v(h_t) = W_v h_t$

$k_t = k(h_t)$ (just like before) e.g., $k_t = W_k h_t$

$q_t = q(h_t)$ e.g., $q_t = W_q h_t$

this is *not* a recurrent model!

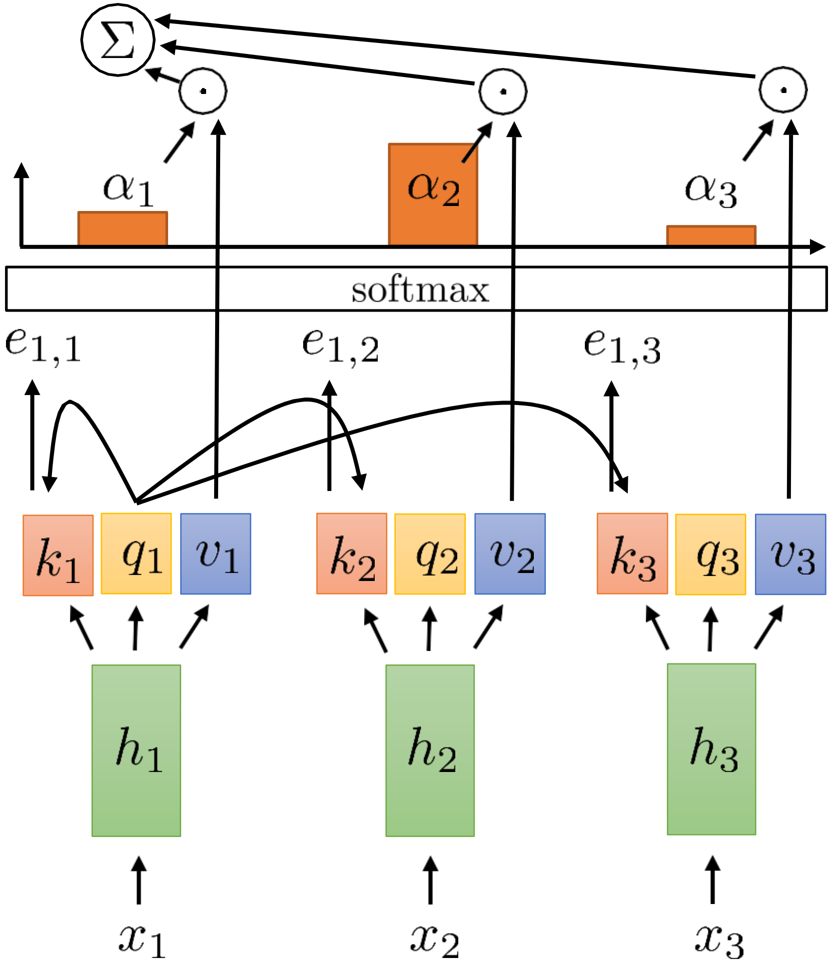
but still weight sharing:

$$h_t = \sigma(Wx_t + b)$$

shared weights at all time steps

(or any other nonlinear function)

Self-Attention



$$a_l = \sum_t \alpha_{l,t} v_t$$

$$\alpha_{l,t} = \exp(e_{l,t}) / \sum_{t'} \exp(e_{l,t'})$$

$$e_{l,t} = q_l \cdot k_t$$

$v_t = v(h_t)$ before just had $v(h_t) = h_t$, now e.g. $v(h_t) = W_v h_t$

$k_t = k(h_t)$ (just like before) e.g., $k_t = W_k h_t$

$q_t = q(h_t)$ e.g., $q_t = W_q h_t$

this is *not* a recurrent model!

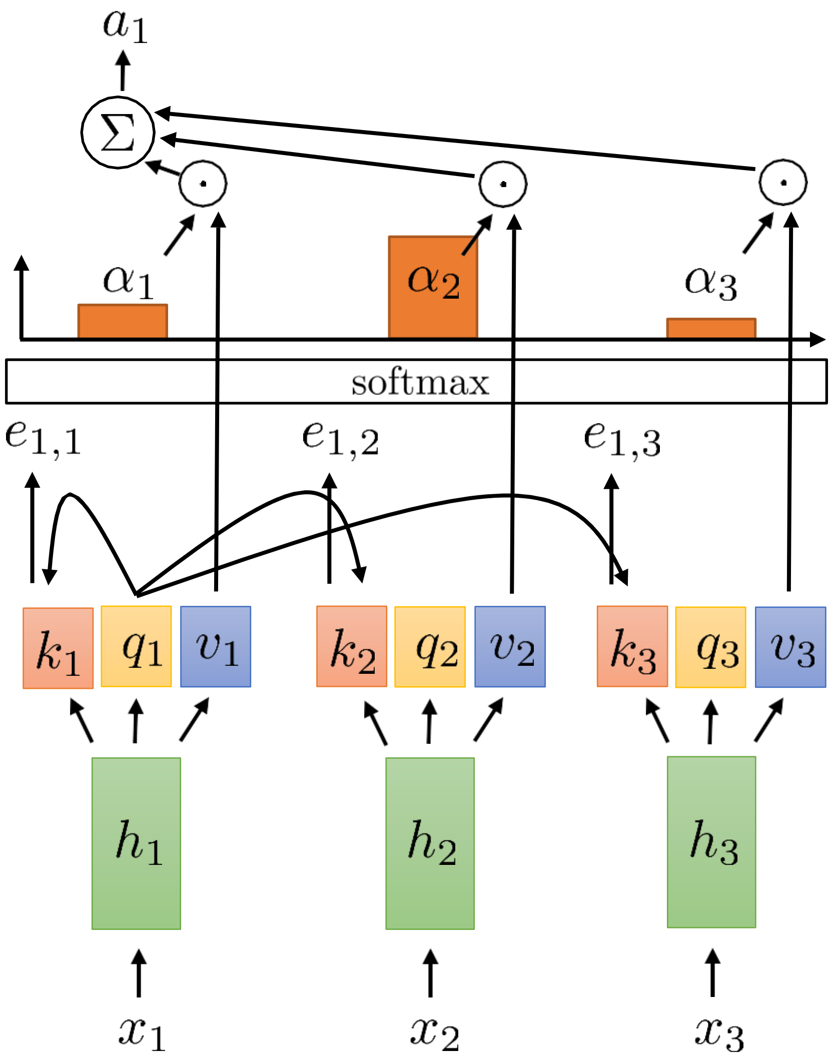
but still weight sharing:

$$h_t = \sigma(W x_t + b)$$

shared weights at all time steps

(or any other nonlinear function)

Self-Attention



$$a_l = \sum_t \alpha_{l,t} v_t$$

$$\alpha_{l,t} = \exp(e_{l,t}) / \sum_{t'} \exp(e_{l,t'})$$

$$e_{l,t} = q_l \cdot k_t$$

$v_t = v(h_t)$ before just had $v(h_t) = h_t$, now e.g. $v(h_t) = W_v h_t$

$k_t = k(h_t)$ (just like before) e.g., $k_t = W_k h_t$

$q_t = q(h_t)$ e.g., $q_t = W_q h_t$

this is *not* a recurrent model!

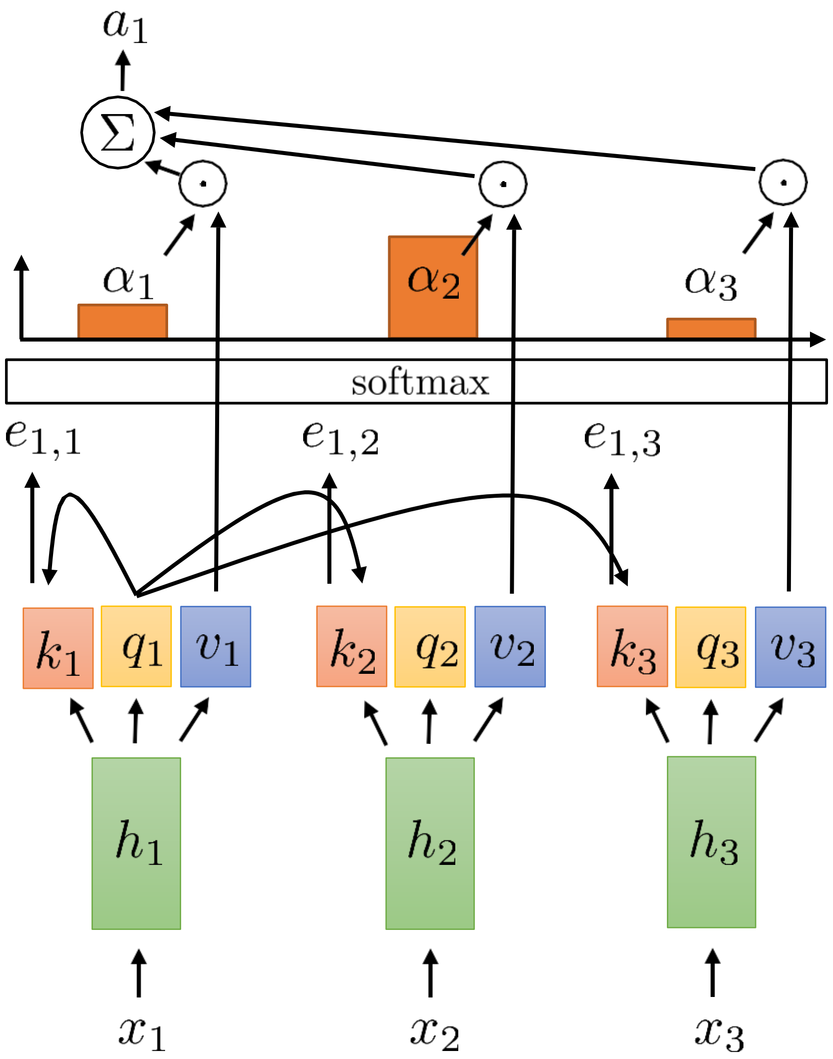
but still weight sharing:

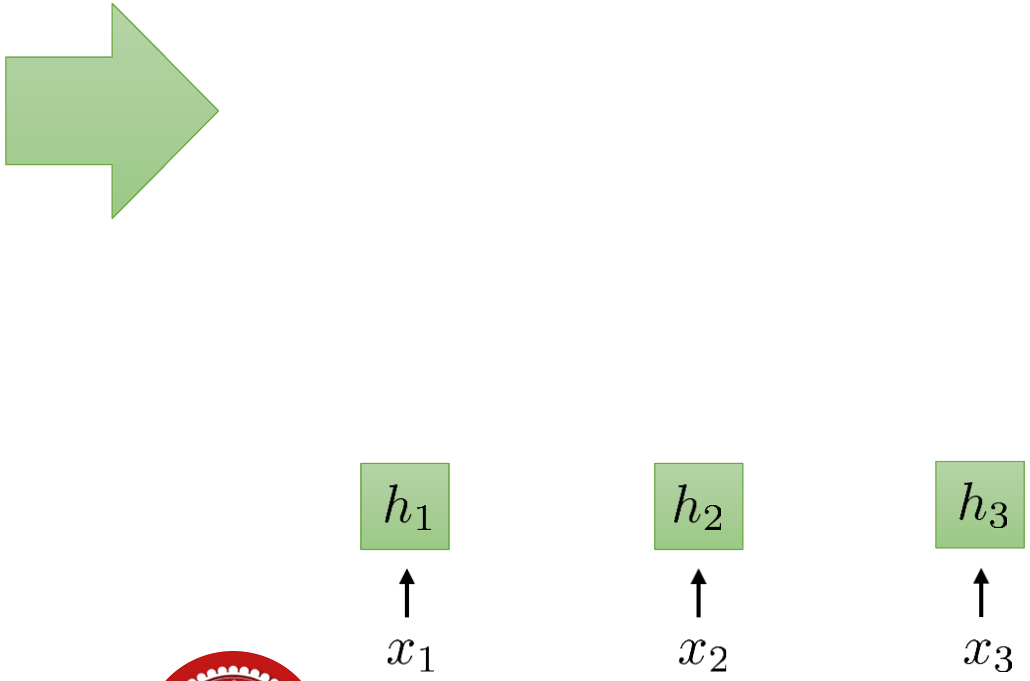
$$h_t = \sigma(Wx_t + b)$$

shared weights at all time steps

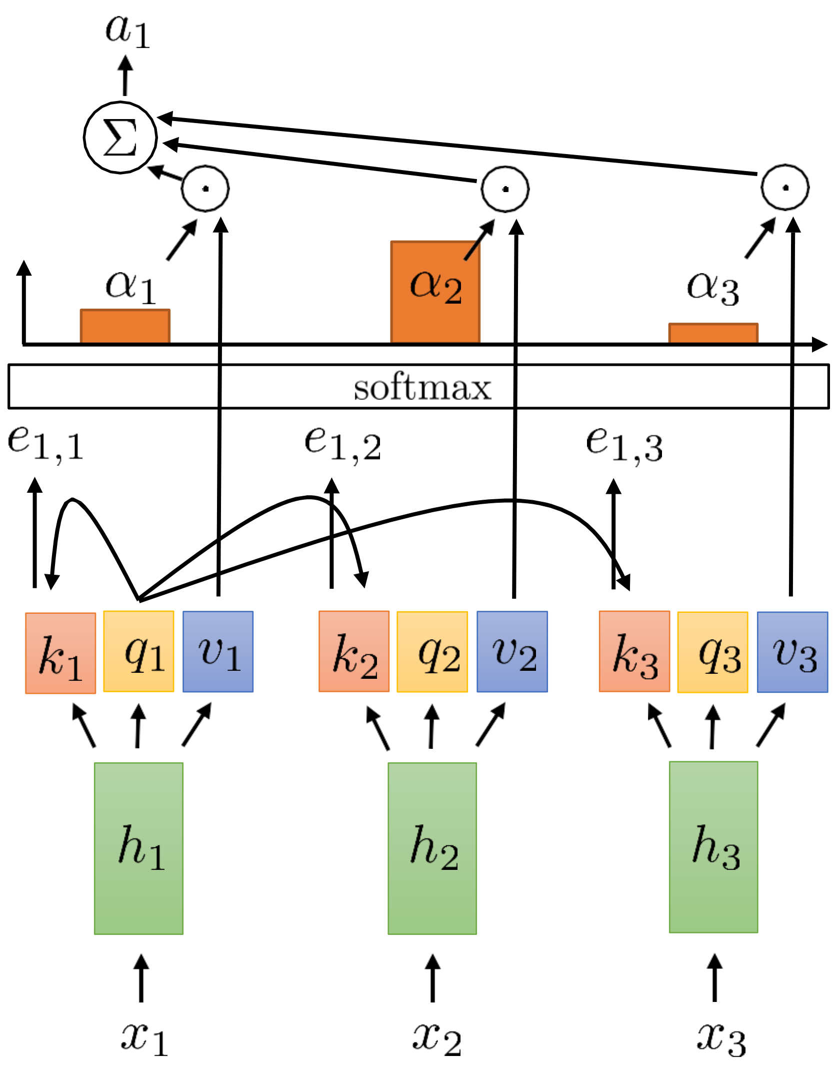
(or any other nonlinear function)

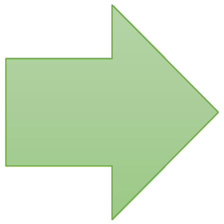
Self-Attention

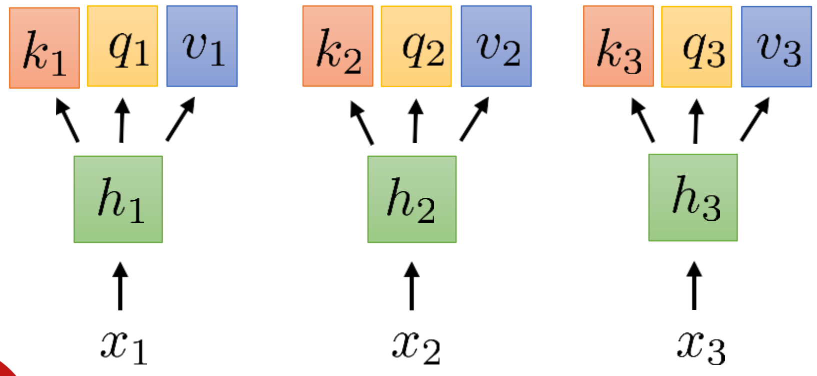




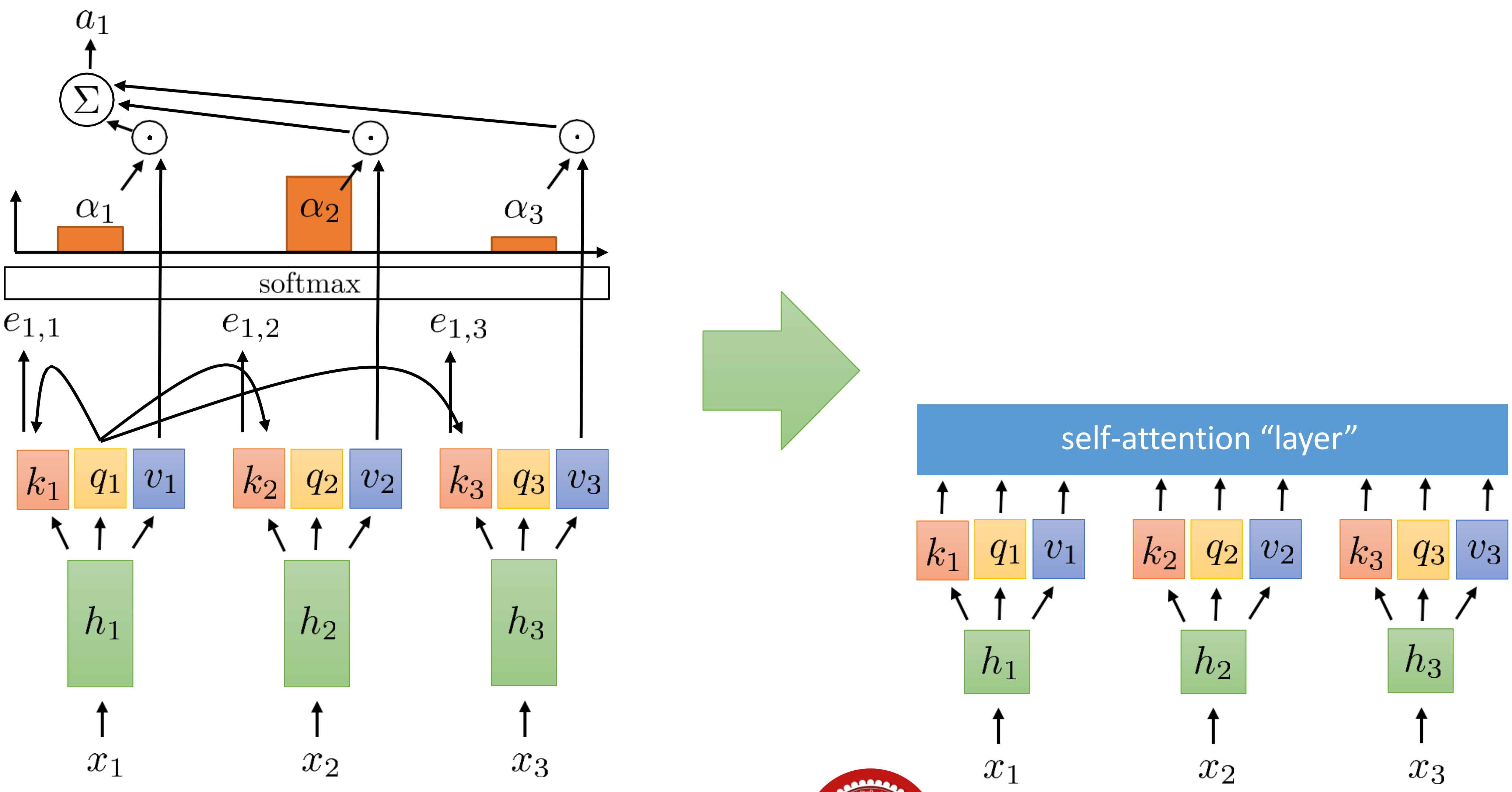
Self-Attention



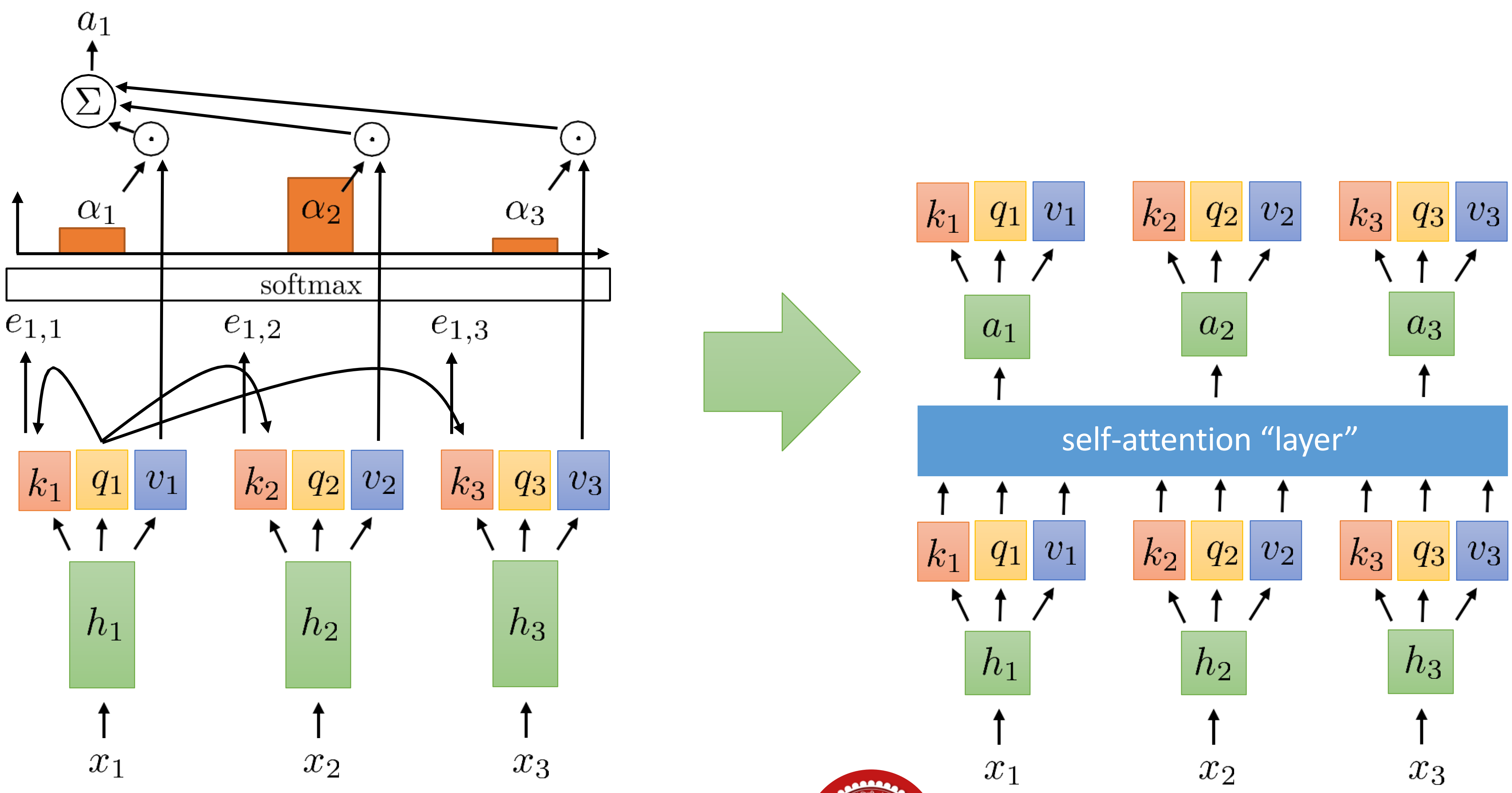




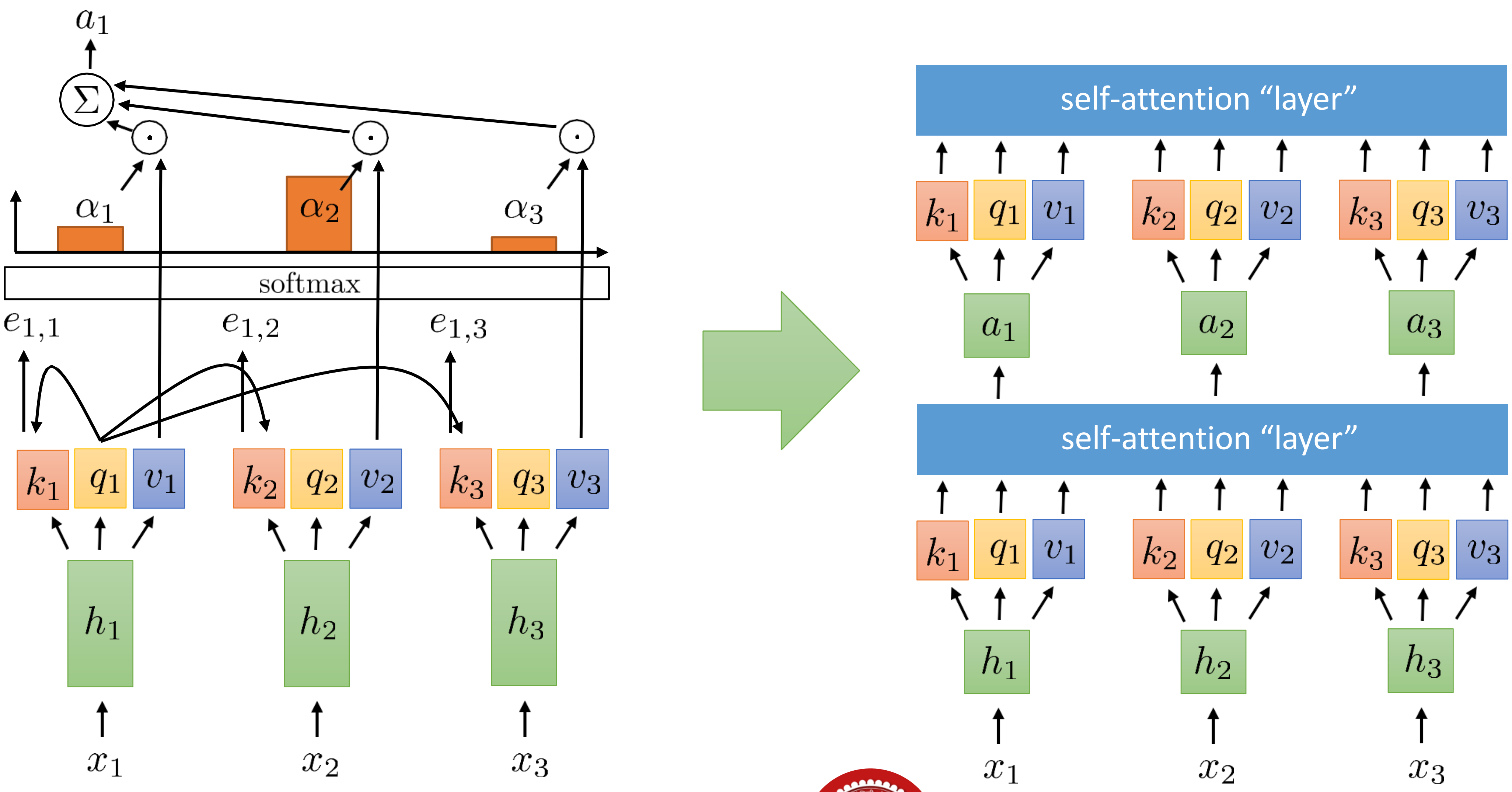
Self-Attention



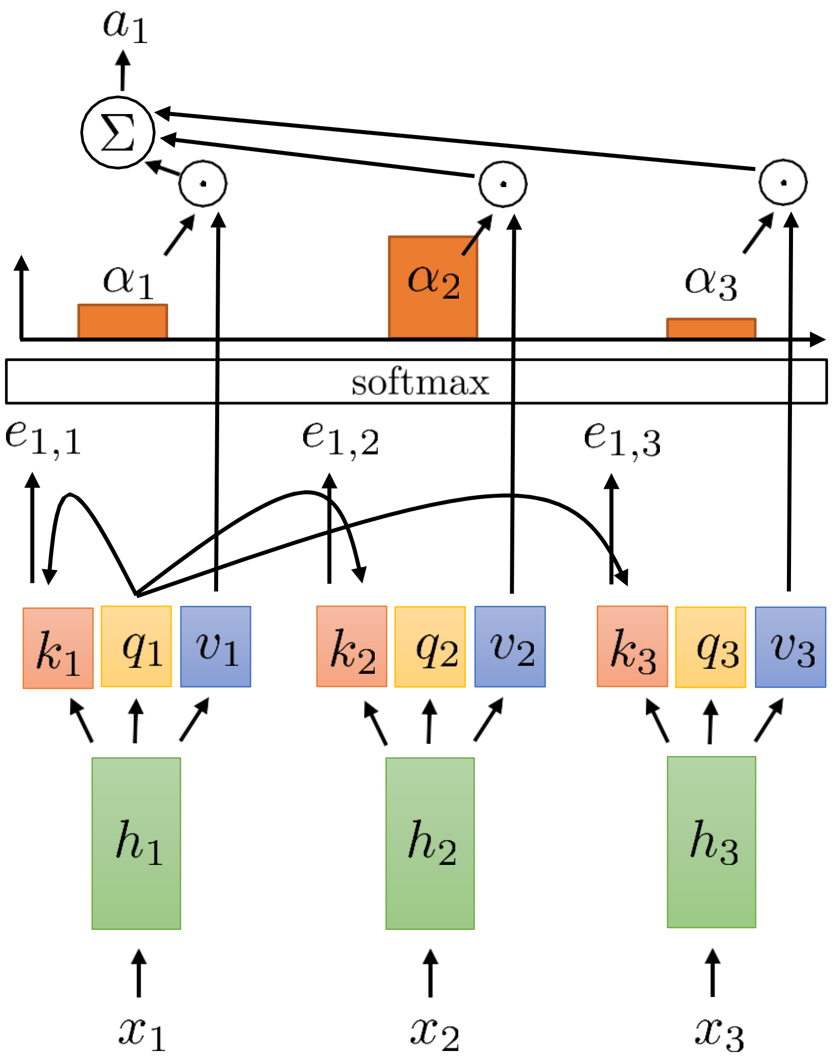
Self-Attention

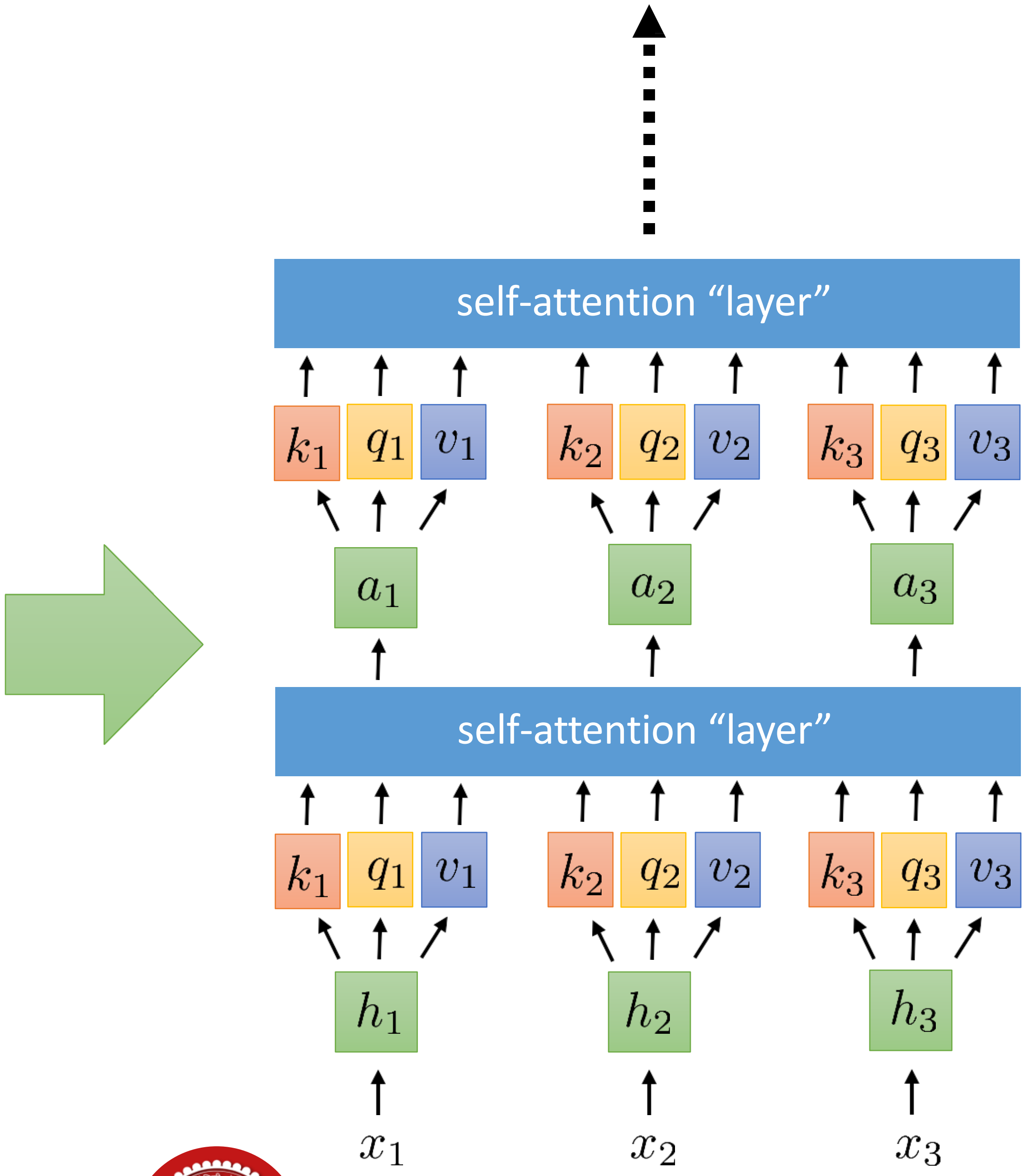


Self-Attention

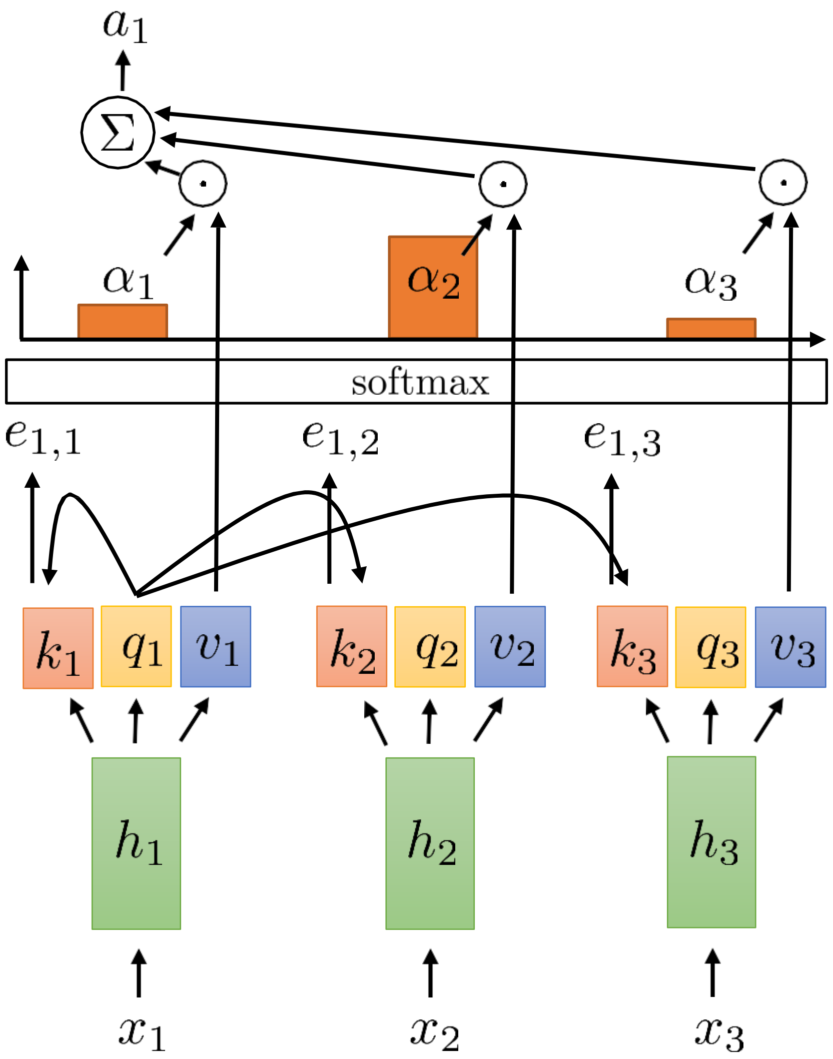


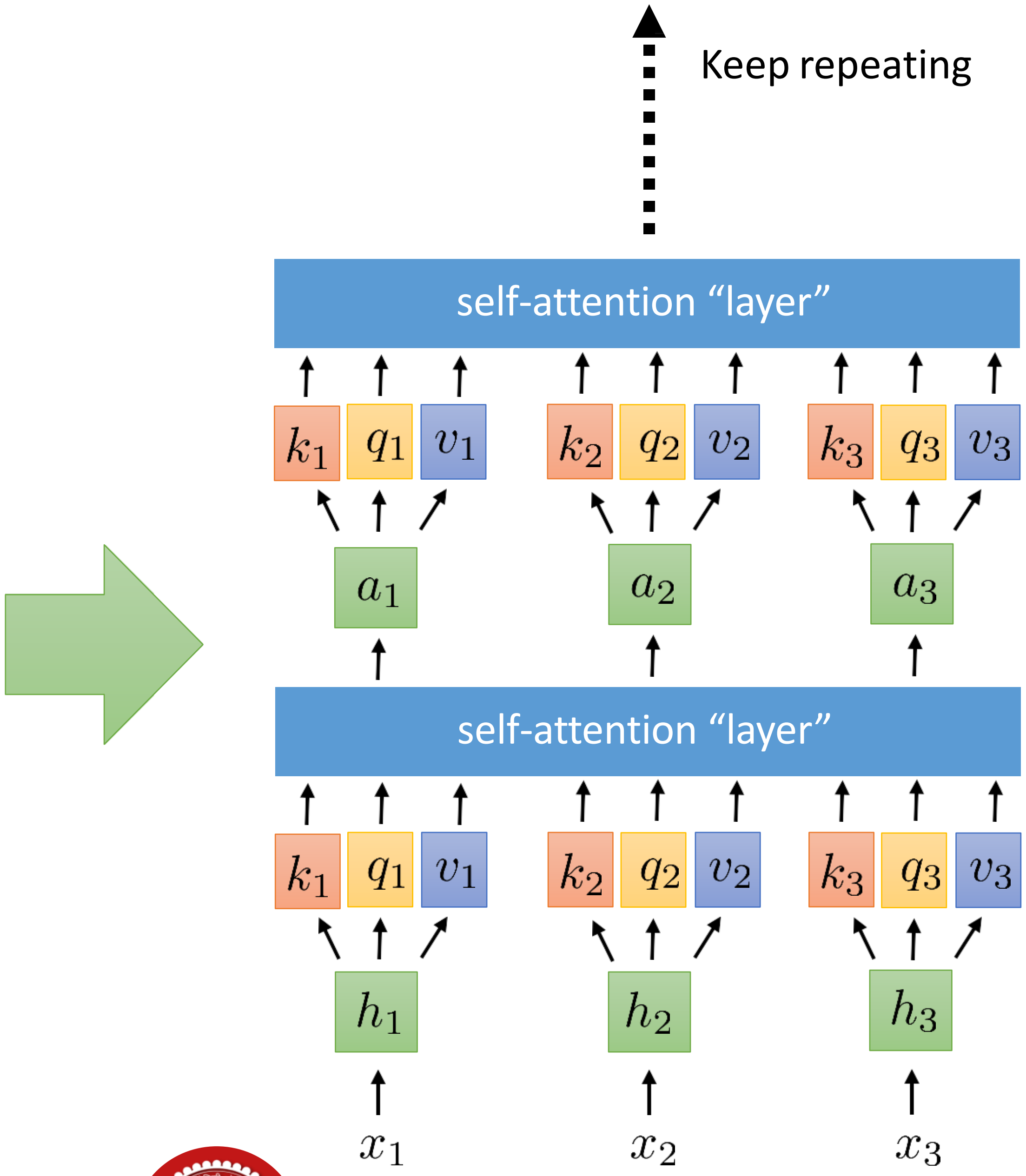
Self-Attention





Self-Attention





From Self-Attention to Transformers

- We will talk about a class of models for processing sequences that does not use recurrent connections but instead relies entirely on attention and will build up towards a class of models called **Transformers**.
- To address a few key limitations, we need to add certain elements:

1. Positional encoding	addresses lack of sequence information
2. Multi-headed attention	allows querying multiple positions at each layer
3. Adding nonlinearities	so far, each successive layer is *linear* in the previous one
4. Masked decoding	how to prevent attention lookups into the future?

From Self-Attention to Transformers

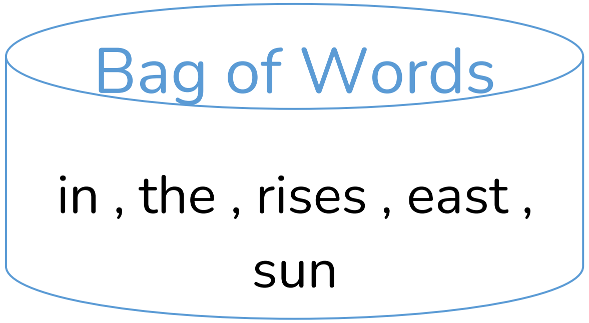
- We will talk about a class of models for processing sequences that does not use recurrent connections but instead relies entirely on attention and will build up towards a class of models called **Transformers**.
- To address a few key limitations, we need to add certain elements:

1. Positional encoding	addresses lack of sequence information
2. Multi-headed attention	allows querying multiple positions at each layer
3. Adding nonlinearities	so far, each successive layer is *linear* in the previous one
4. Masked decoding	how to prevent attention lookups into the future?

Positional Encoding - Motivation

- **Problem :** Self-attention processes all the elements of a sequence in parallel without any regard for their order.
  - Example : the sun rises in the east
  - Permuted version : rises in the sun the east

    the east rises in the sun
  - Self-attention is permutation invariant.
  - In natural language, it is important to take into account the order of words in a sentence.



Bag of Words

in , the , rises , east , sun

- **Solution :** Explicitly add positional information to indicate where a word appears in a sequence

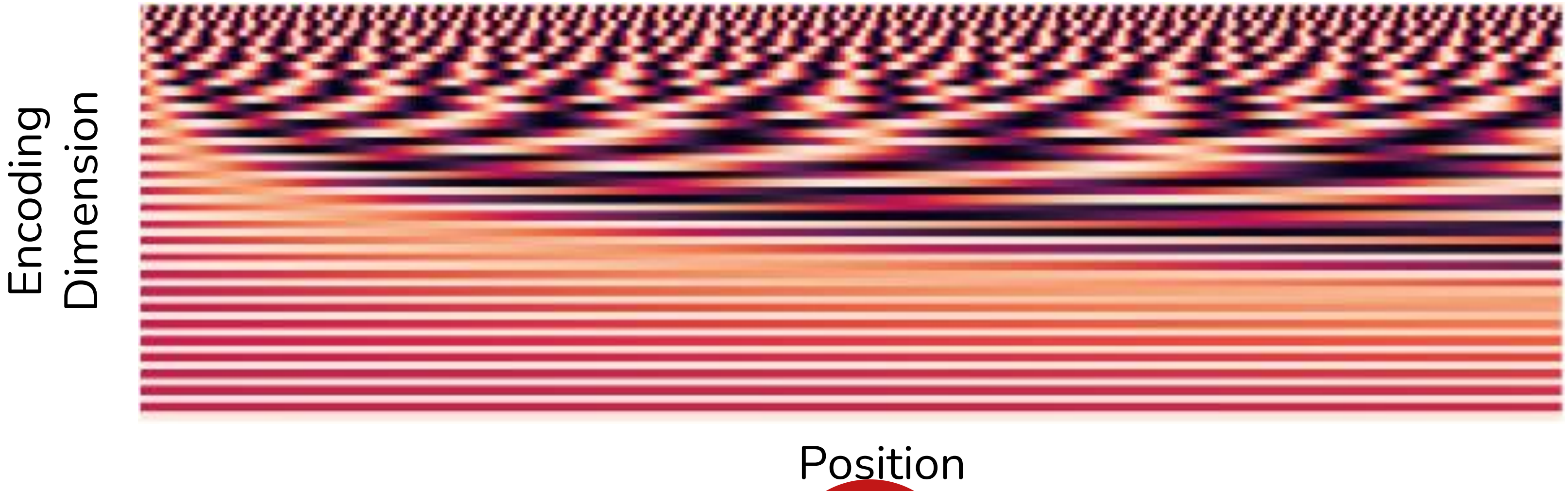
Sinusoidal Positional Encoding

- Helps it determine the position of each word (absolute positional information), or the distance between different words in the sequence(relative positional information)
- The frequency decreases along the encoding dimension.

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

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

We will see this in more detail later !



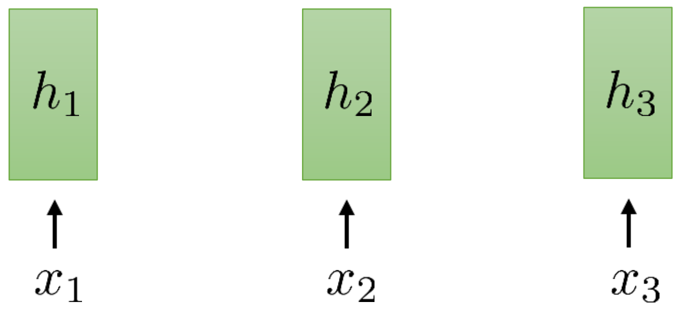
From Self-Attention to Transformers

- We will talk about a class of models for processing sequences that does not use recurrent connections but instead relies entirely on attention and will build up towards a class of models called transformers.
- To address a few key limitations, we need to add certain elements:

1. Positional encoding	addresses lack of sequence information
2. Multi-headed attention	allows querying multiple positions at each layer
3. Adding nonlinearities	so far, each successive layer is *linear* in the previous one
4. Masked decoding	how to prevent attention lookups into the future?

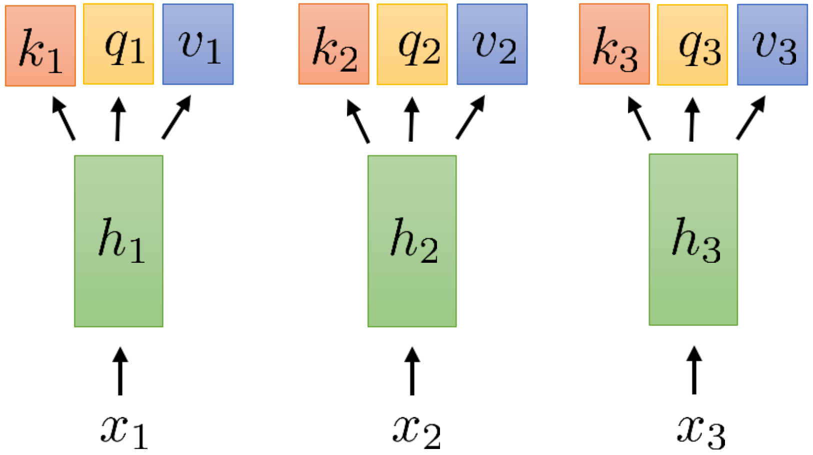
Multi-Head Attention

Given that we're fully depending on attention now, it could be beneficial to include more than one time step.



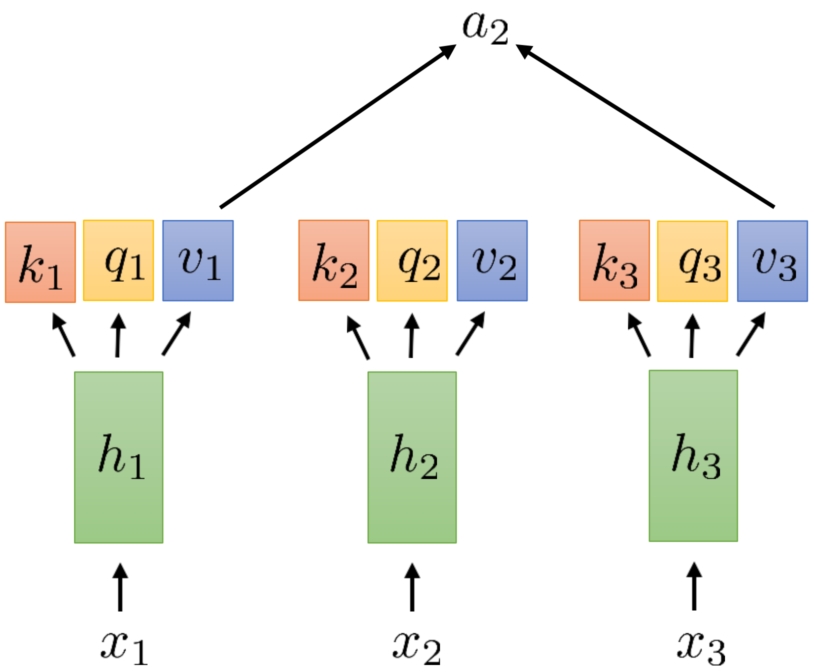
Multi-Head Attention

Given that we're fully depending on attention now, it could be beneficial to include more than one time step.



Multi-Head Attention

Given that we're fully depending on attention now, it could be beneficial to include more than one time step.

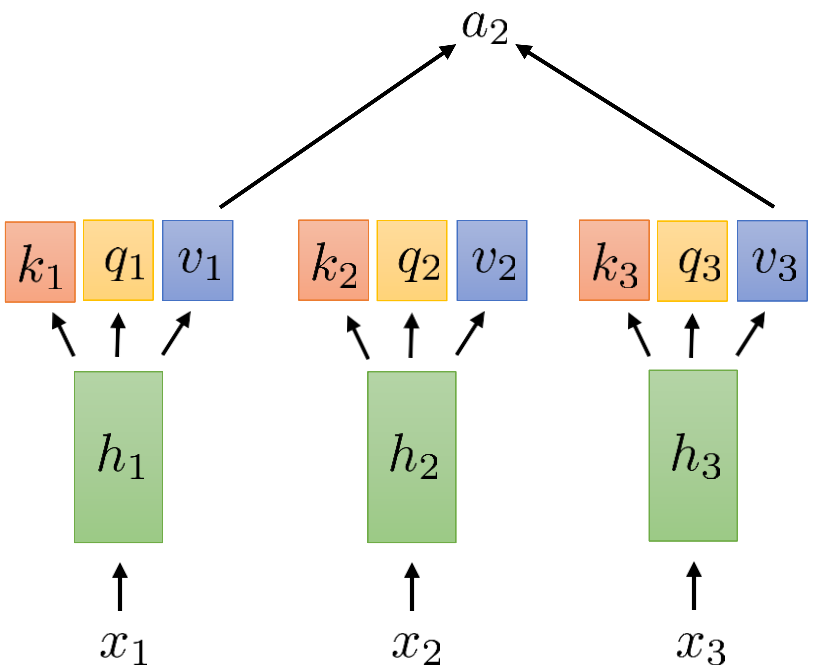


$$a_l = \sum_t \alpha_{l,t} v_t$$

Due to the softmax function, this will be heavily influenced by a single value.

Multi-Head Attention

Given that we're fully depending on attention now, it could be beneficial to include more than one time step.



$$a_l = \sum_t \alpha_{l,t} v_t$$

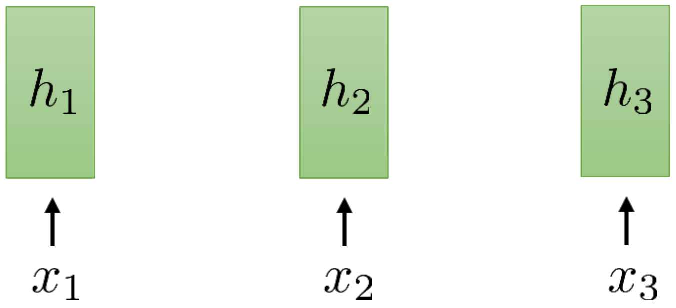
Due to the softmax function, this will be heavily influenced by a single value.

$$e_{l,t} = q_l \cdot k_t$$

It's challenging to clearly specify that you want two distinct elements, like the subject and object in a sentence.

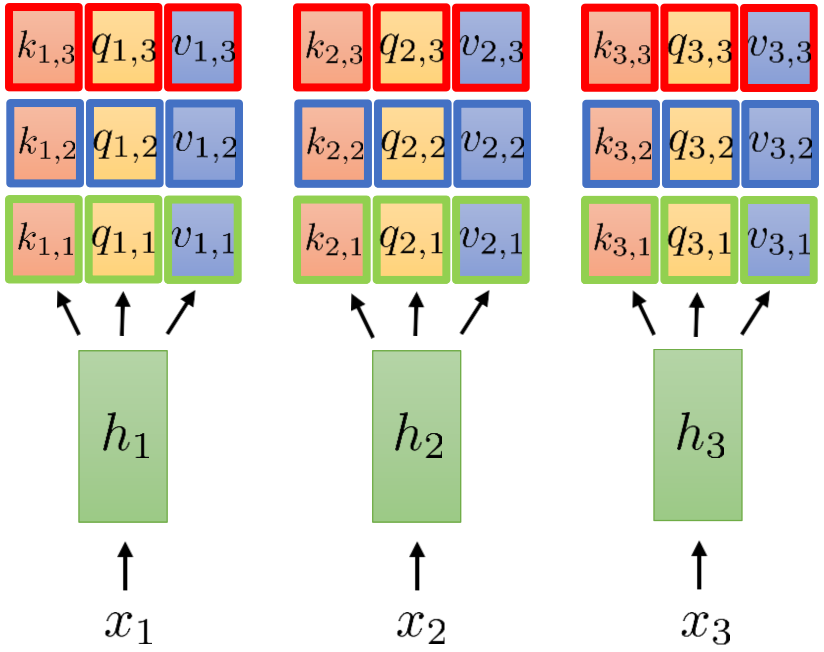
Multi-Head Attention

Solution: Use multiple keys, queries, and values for each time step



Multi-Head Attention

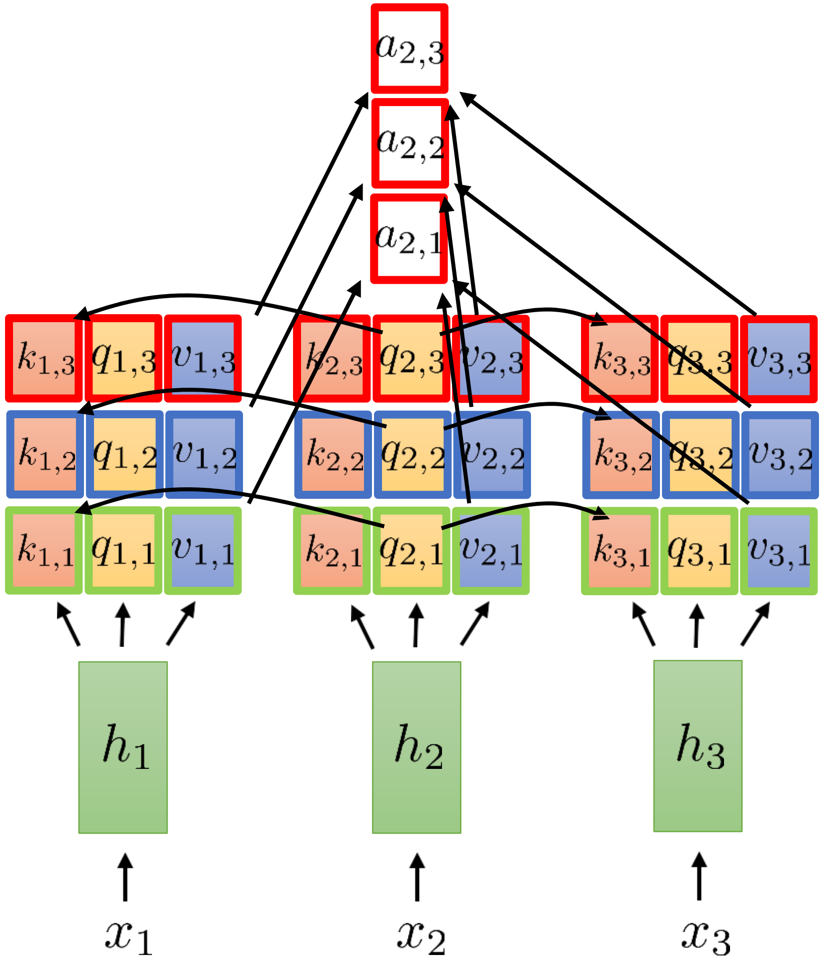
Solution: Use multiple keys, queries, and values for each time step



$k_{1,3}$ $q_{1,3}$ $v_{1,3}$
$k_{1,2}$ $q_{1,2}$ $v_{1,2}$
$k_{1,1}$ $q_{1,1}$ $v_{1,1}$
$k_{2,3}$ $q_{2,3}$ $v_{2,3}$
$k_{2,2}$ $q_{2,2}$ $v_{2,2}$
$k_{2,1}$ $q_{2,1}$ $v_{2,1}$
$k_{3,3}$ $q_{3,3}$ $v_{3,3}$
$k_{3,2}$ $q_{3,2}$ $v_{3,2}$
$k_{3,1}$ $q_{3,1}$ $v_{3,1}$
h_1
h_2
h_3
x_1
x_2
x_3

Multi-Head Attention

Solution: Use multiple keys, queries, and values for each time step



$a_{2,3}$
$a_{2,2}$
$a_{2,1}$
$k_{1,3}$ $q_{1,3}$ $v_{1,3}$
$k_{2,3}$ $q_{2,3}$ $v_{2,3}$
$k_{3,3}$ $q_{3,3}$ $v_{3,3}$
$k_{1,2}$ $q_{1,2}$ $v_{1,2}$
$k_{2,2}$ $q_{2,2}$ $v_{2,2}$
$k_{3,2}$ $q_{3,2}$ $v_{3,2}$
$k_{1,1}$ $q_{1,1}$ $v_{1,1}$
$k_{2,1}$ $q_{2,1}$ $v_{2,1}$
$k_{3,1}$ $q_{3,1}$ $v_{3,1}$
h_1
h_2
h_3
x_1
x_2
x_3

full attention vector formed by concatenation:

$$a_2 = \begin{bmatrix} a_{2,1} \\ a_{2,2} \\ a_{2,3} \end{bmatrix}$$

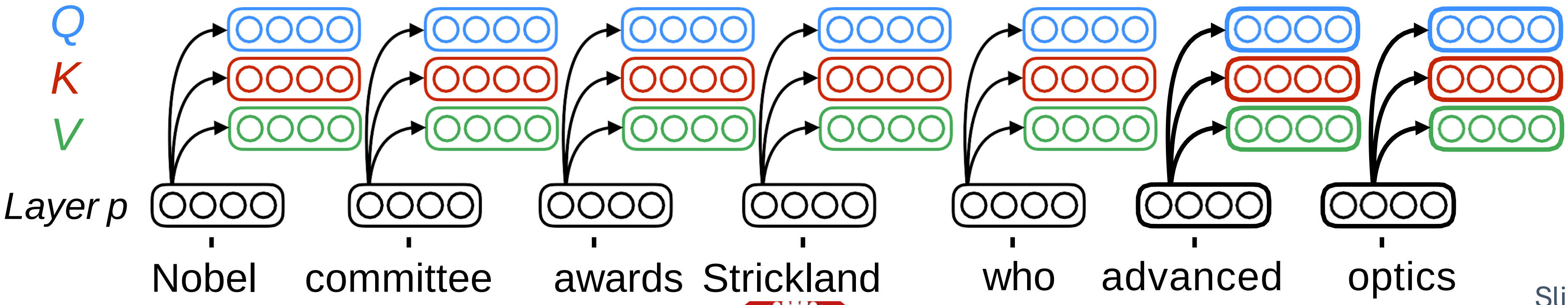
compute weights **independently** for each head

$$e_{l,t,i} = q_{l,i} \cdot k_{l,i}$$

$$\alpha_{l,t,i} = \exp(e_{l,t,i}) / \sum_{t'} \exp(e_{l,t',i})$$

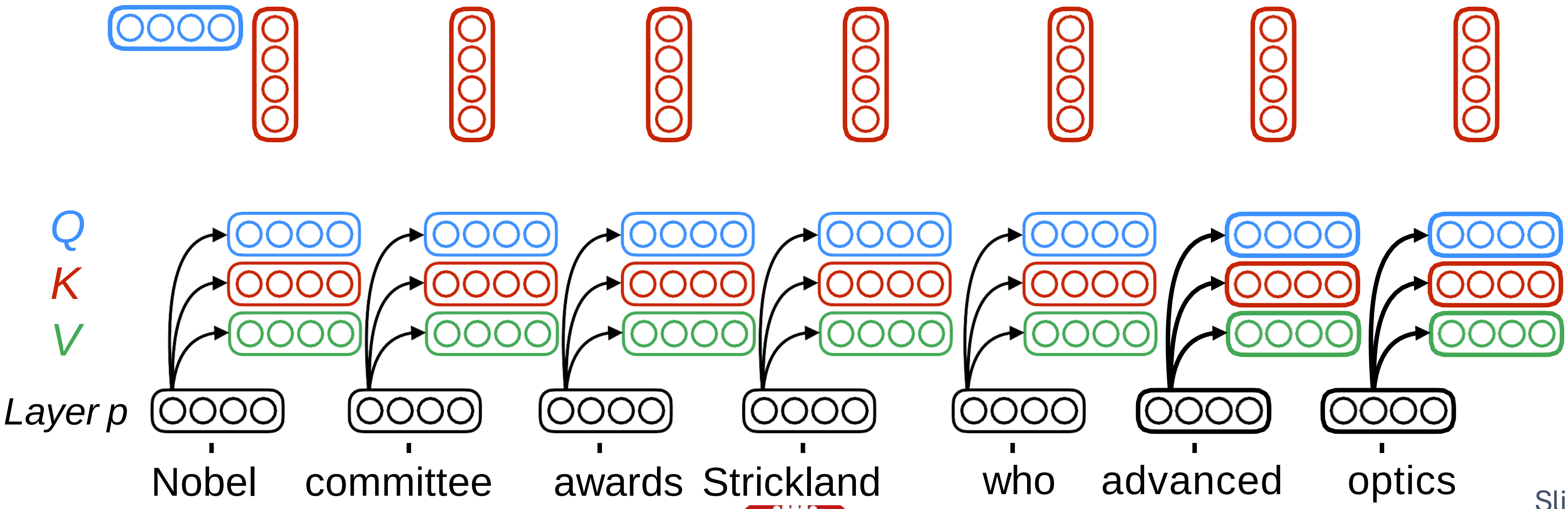
$$a_{l,i} = \sum_t \alpha_{l,t,i} v_{t,i}$$

Self-Attention (In Encoder)



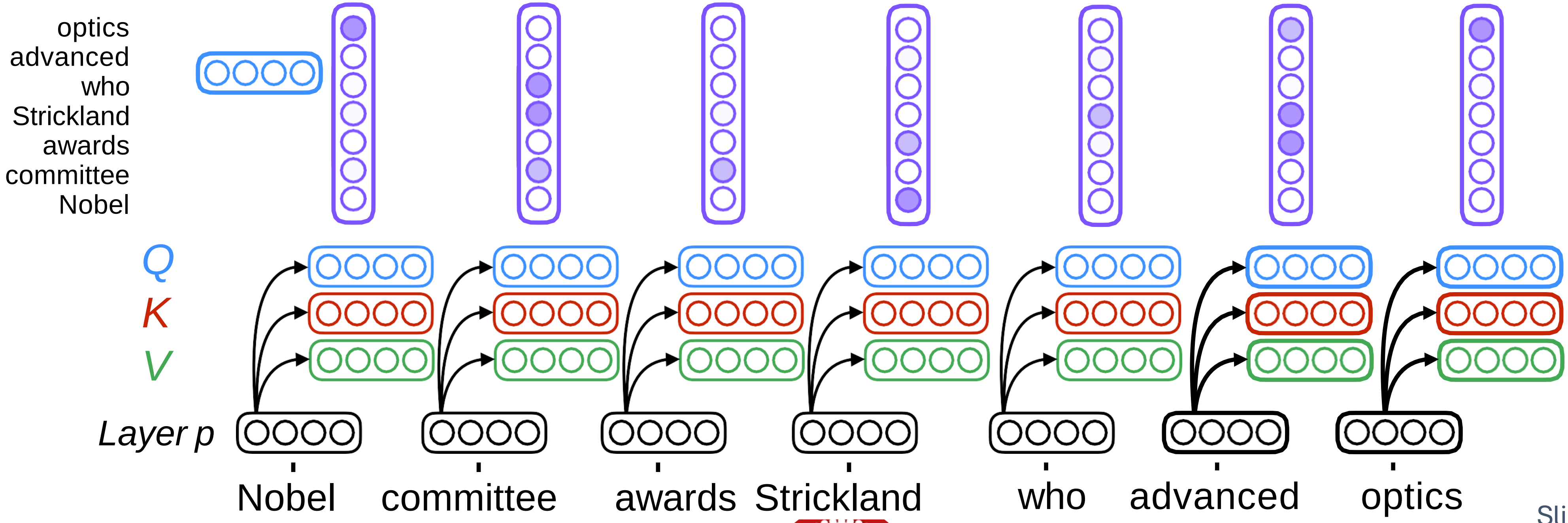
Slides by Emma Strubell

Self-Attention (In Encoder)



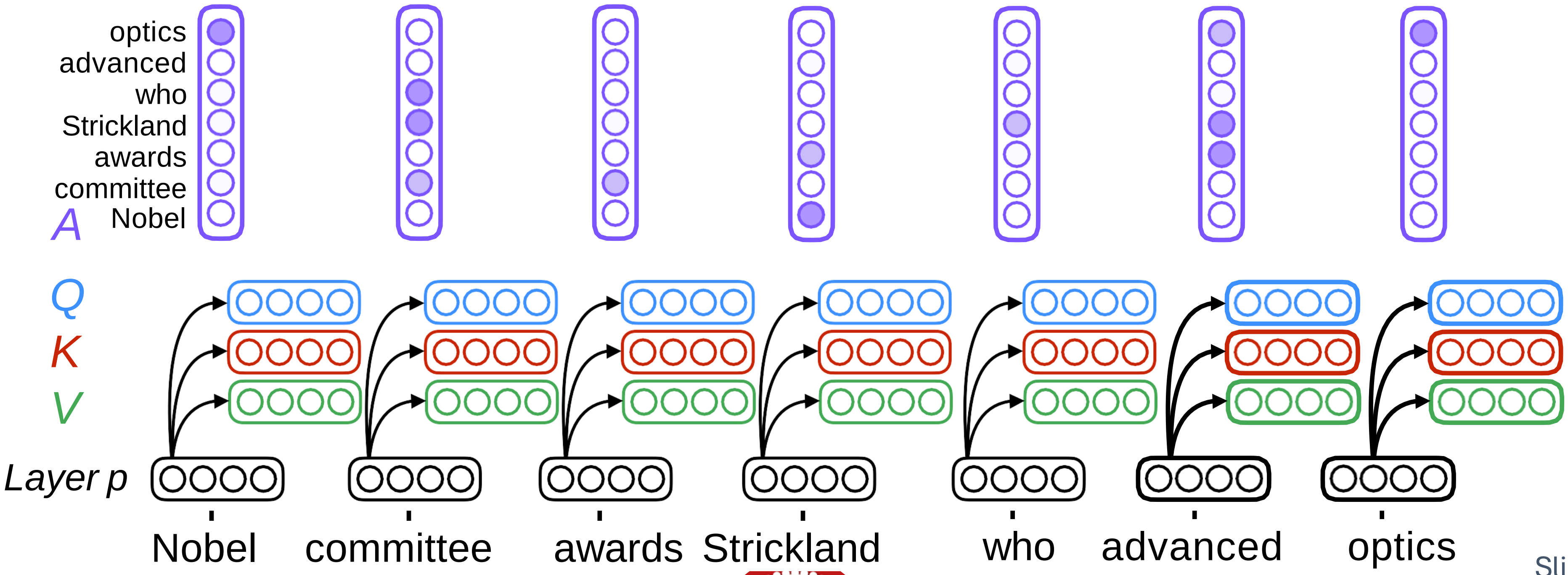
Slides by Emma Strubell

Self-Attention (In Encoder)



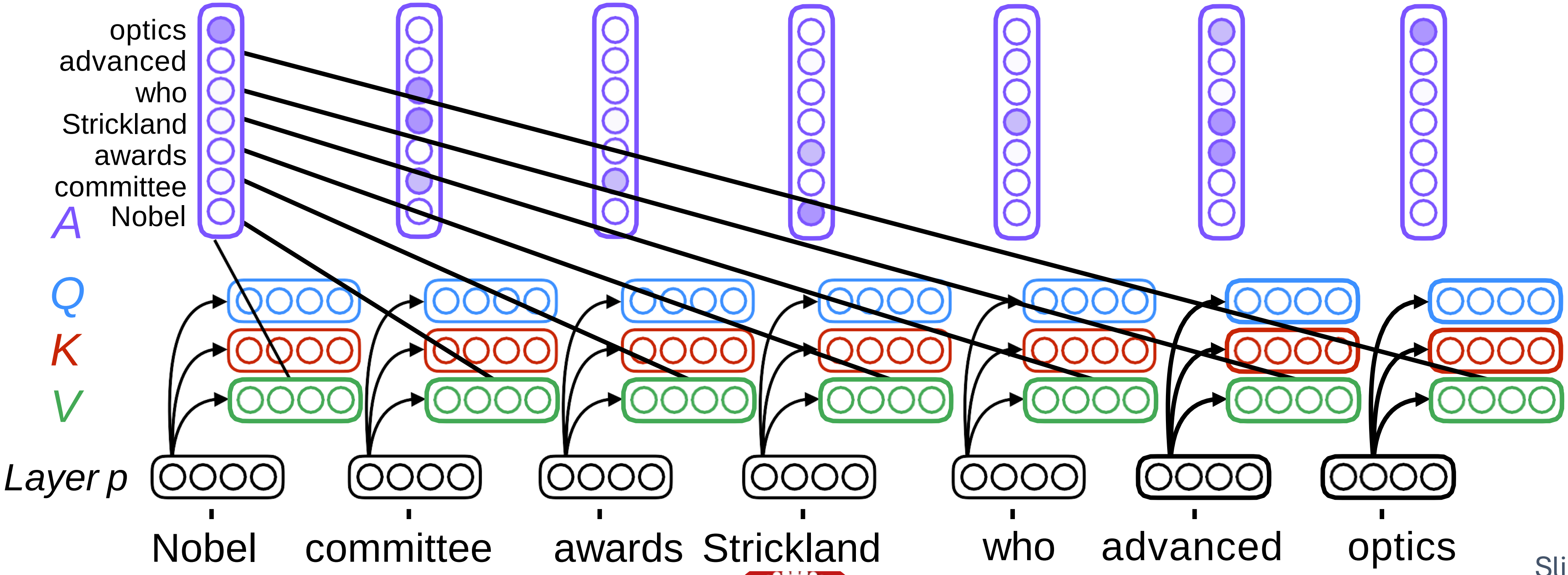
Slides by Emma Strubell

Self-Attention (In Encoder)



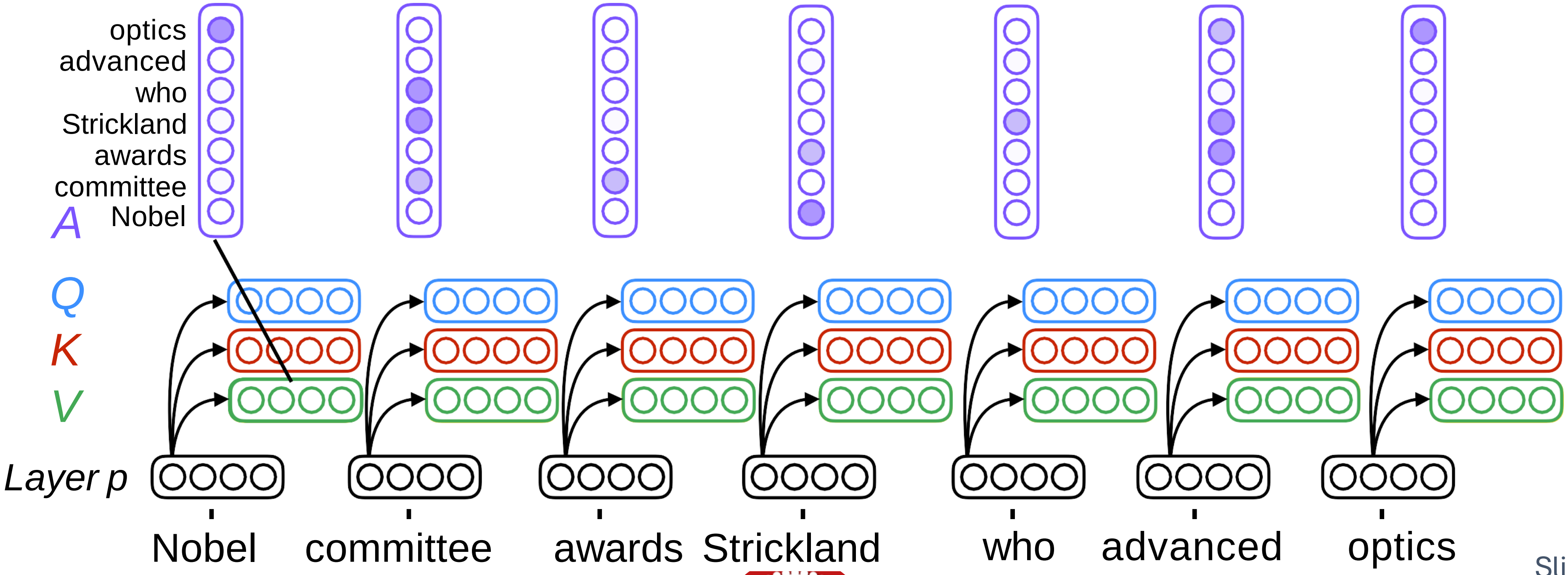
Slides by Emma Strubell

Self-Attention (In Encoder)



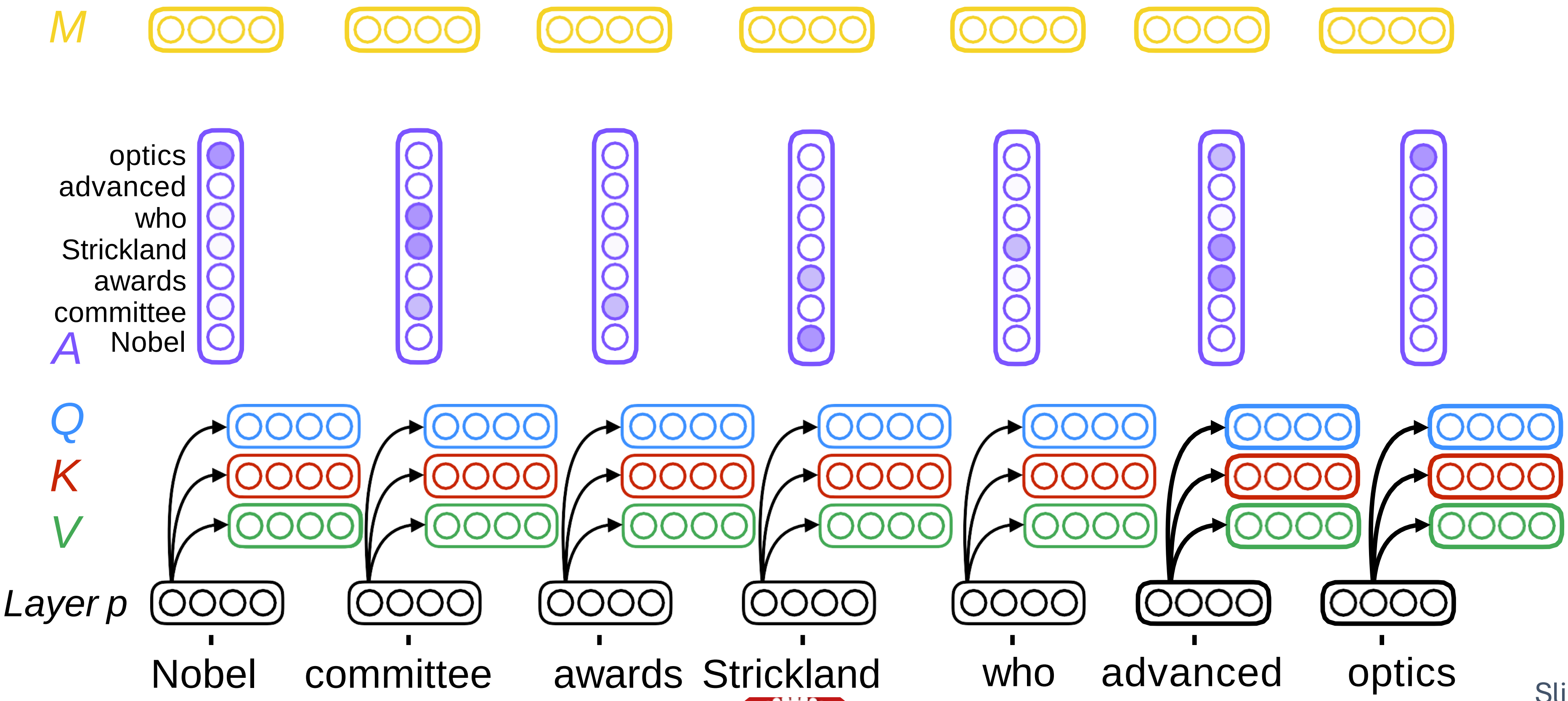
Slides by Emma Strubell

Self-Attention (In Encoder)



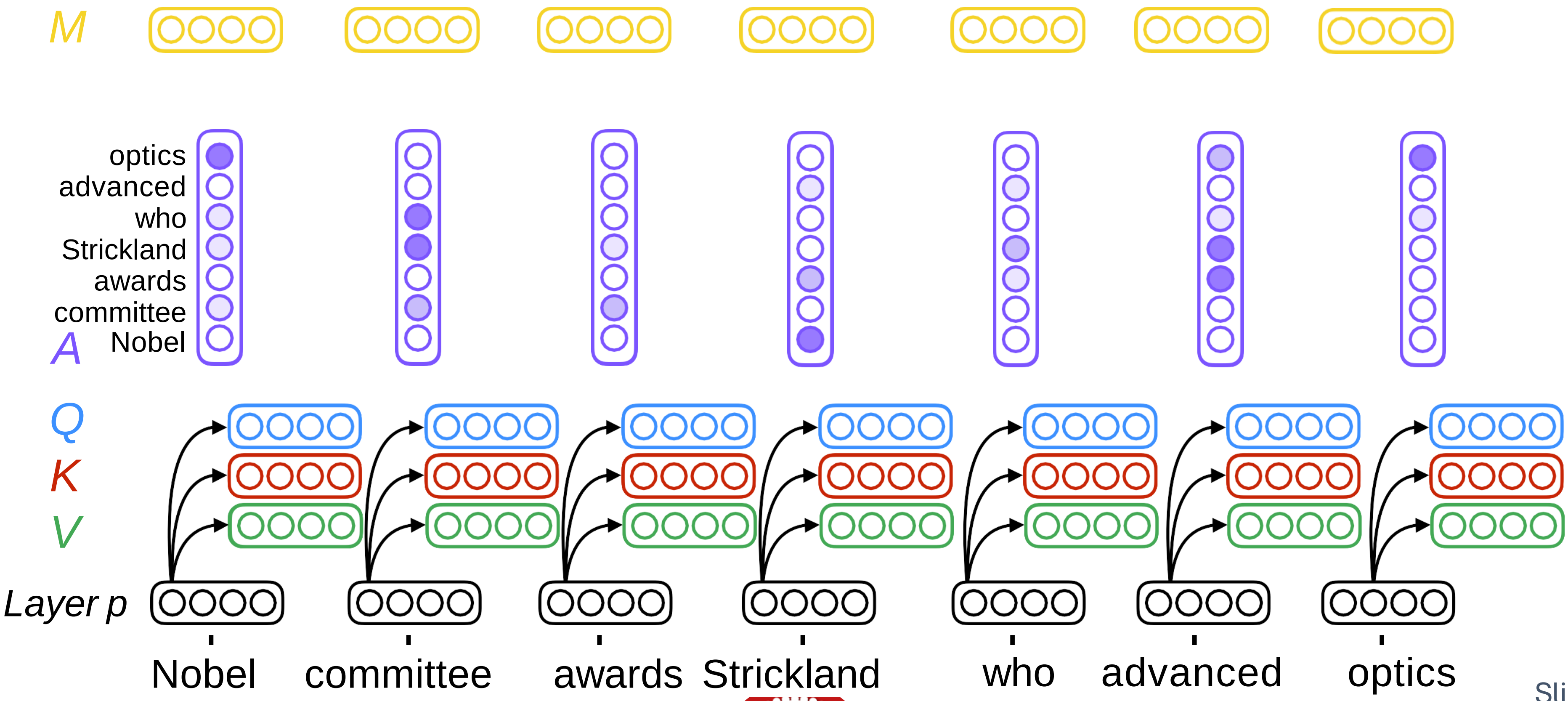
Slides by Emma Strubell

Self-Attention (In Encoder)



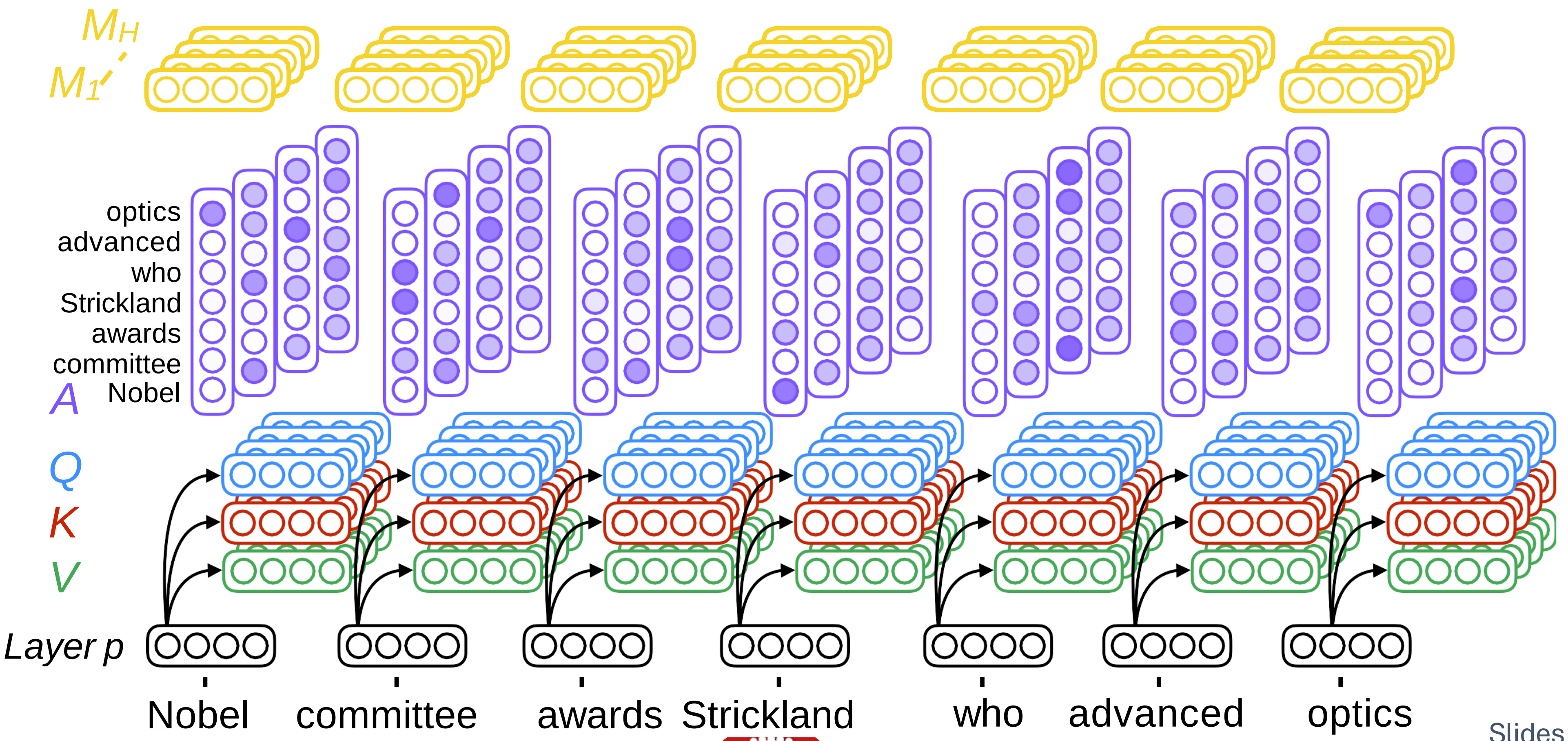
Slides by Emma Strubell

Self-Attention (In Encoder)



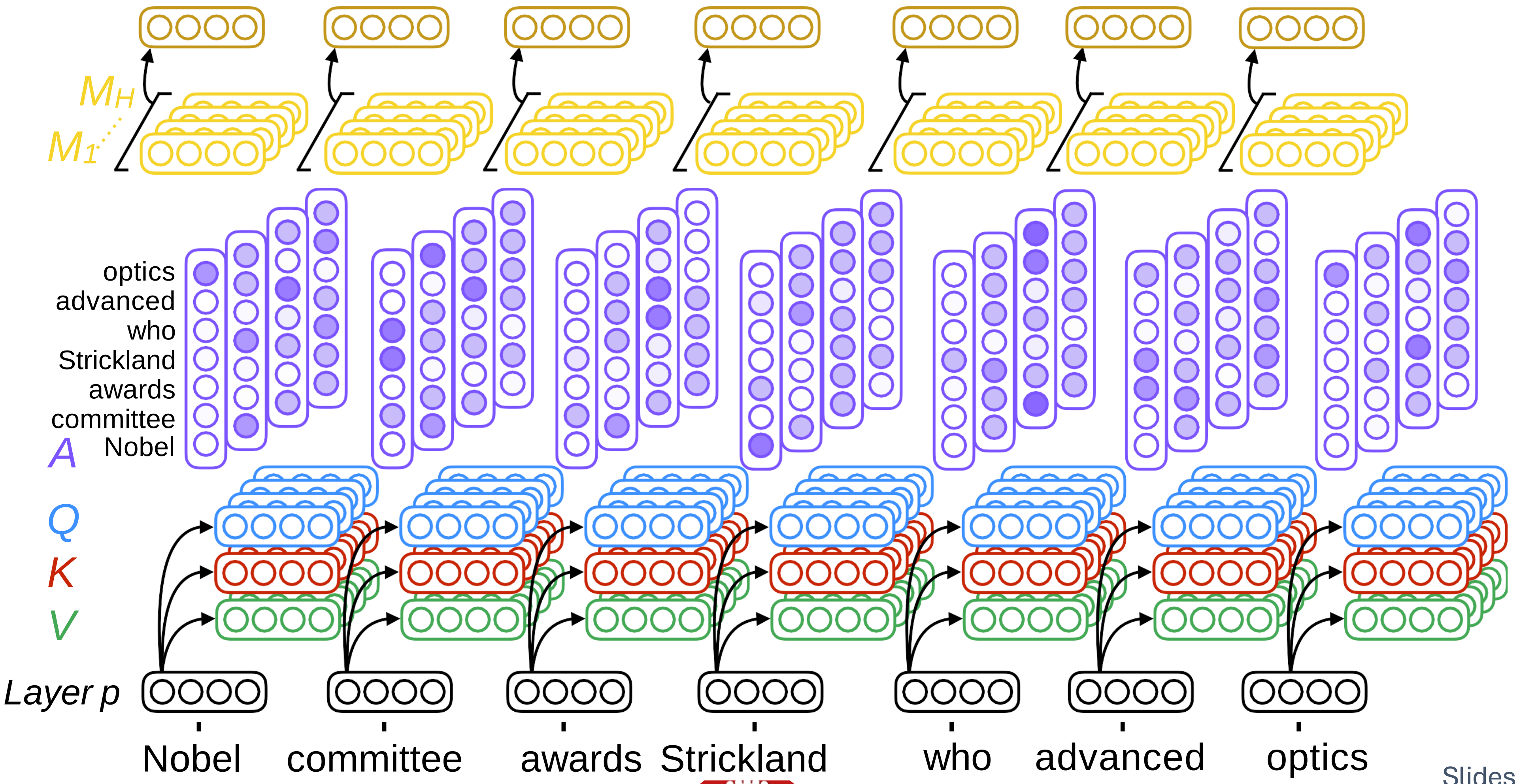
Slides by Emma Strubell

Multi-Head Self-Attention



Slides by Emma Strubell

Multi-Head Self-Attention



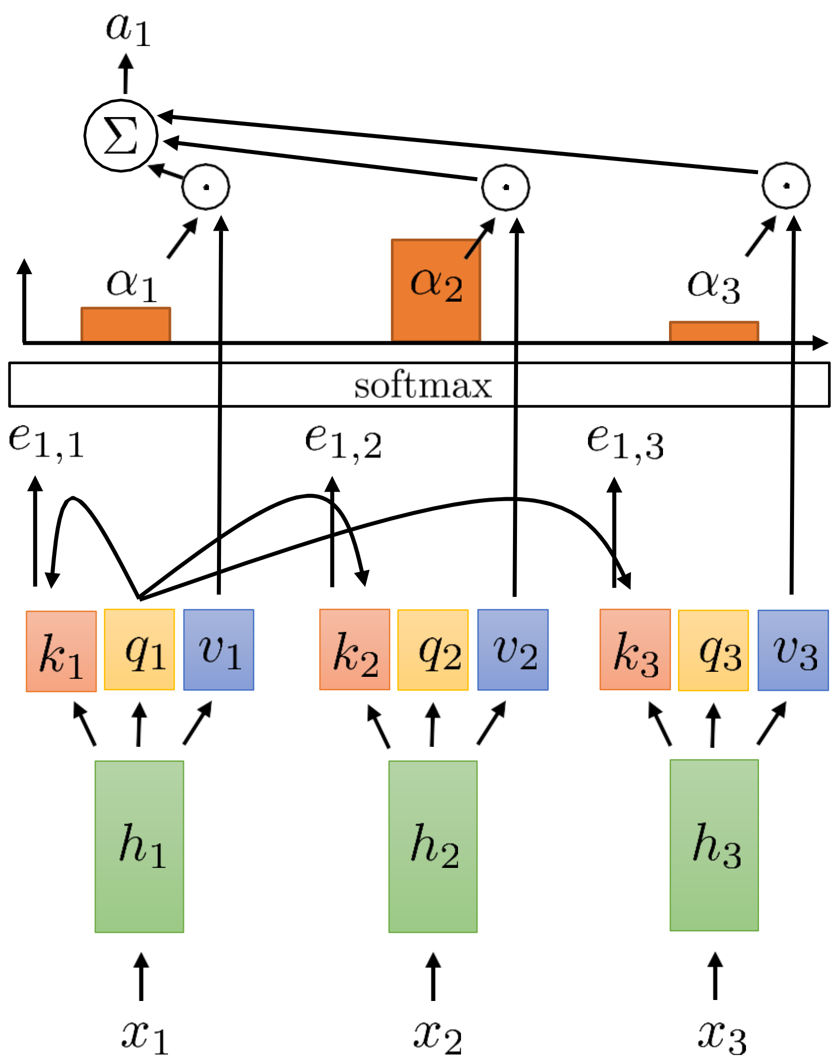
Slides by Emma Strubell

From Self-Attention to Transformers

- We will talk about a class of models for processing sequences that does not use recurrent connections but instead relies entirely on attention and will build up towards a class of models called transformers.
- To address a few key limitations, we need to add certain elements:

1. Positional encoding	addresses lack of sequence information
2. Multi-headed attention	allows querying multiple positions at each layer
3. Adding nonlinearities	so far, each successive layer is *linear* in the previous one
4. Masked decoding	how to prevent attention lookups into the future?

Self-Attention Is "Linear"



$$k_t = W_k h_t \qquad q_t = W_q h_t \qquad v_t = W_v h_t$$

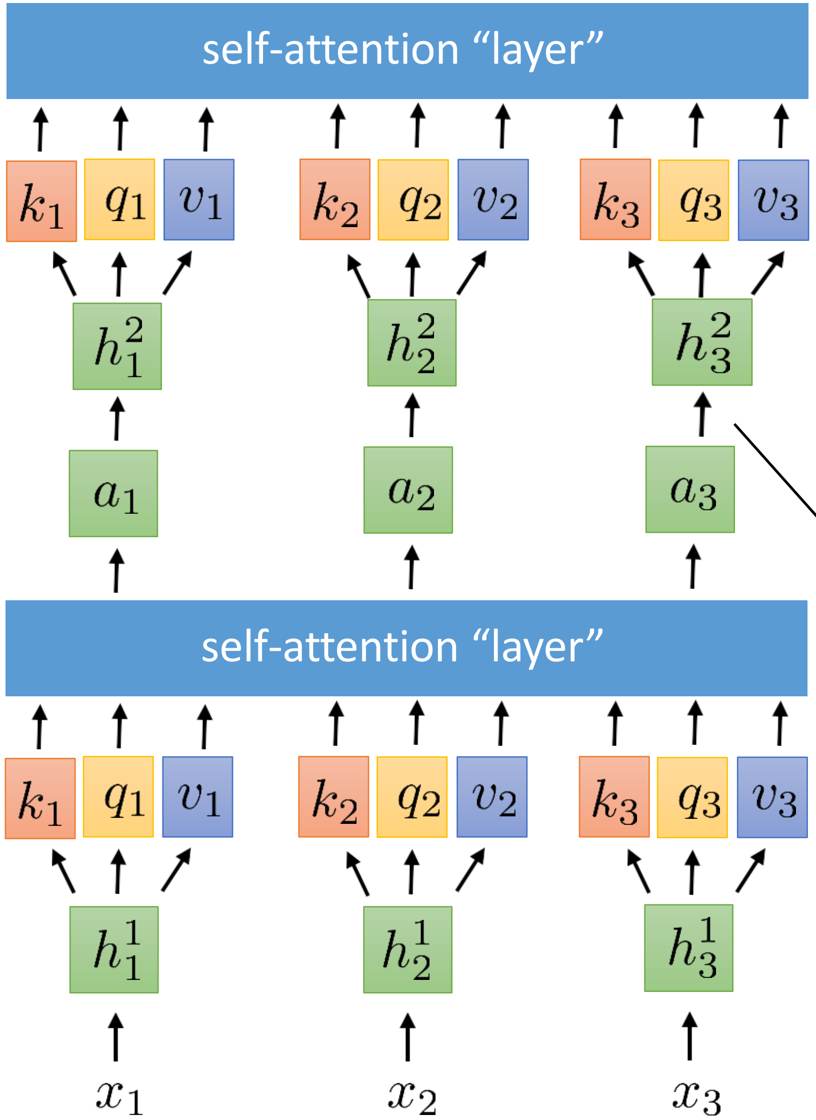
$$\alpha_{l,t} = \exp(e_{l,t}) / \sum_{t'} \exp(e_{l,t'})$$

$$e_{l,t} = q_l \cdot k_t$$

$$a_l = \sum_t \alpha_{l,t} v_t = \sum_t \alpha_{l,t} W_v h_t = \underbrace{W_v}_{\text{linear transformation}} \sum_t \underbrace{\alpha_{l,t}}_{\text{non-linear weights}} h_t$$

Problem: Every self-attention layer is a linear transformation of the previous layer with non-linear weights.

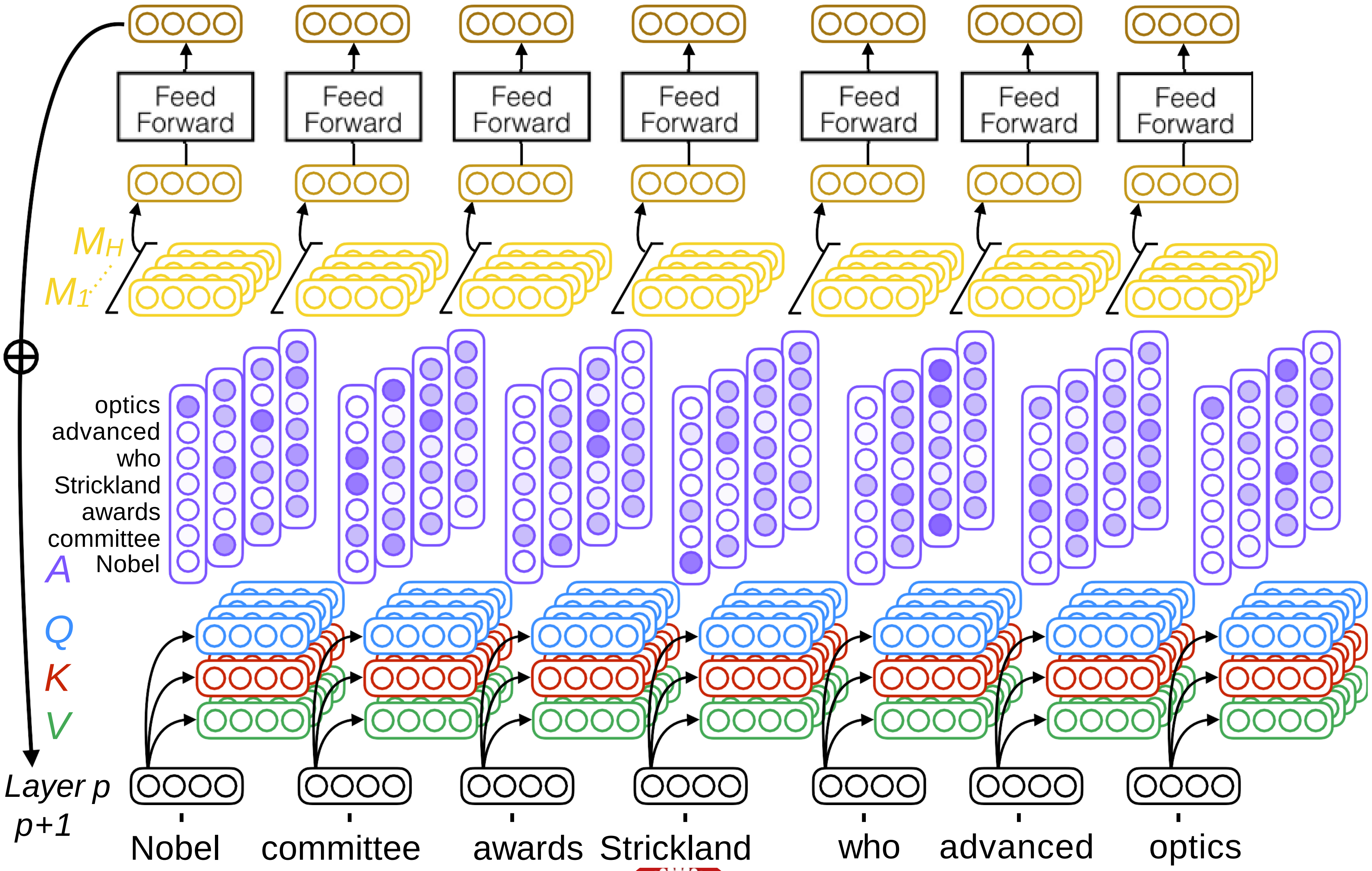
Position-wise Feed-Forward Networks



- **Solution :** Make the model more expressive is by alternating use of self-attention and non-linearity.
- Non-linearity is incorporated by means of a feed-forward network which consists of two linear transformations with a ReLU activation in between.

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

- The same non-linearity is utilized across various positions but they differ from layer to layer.

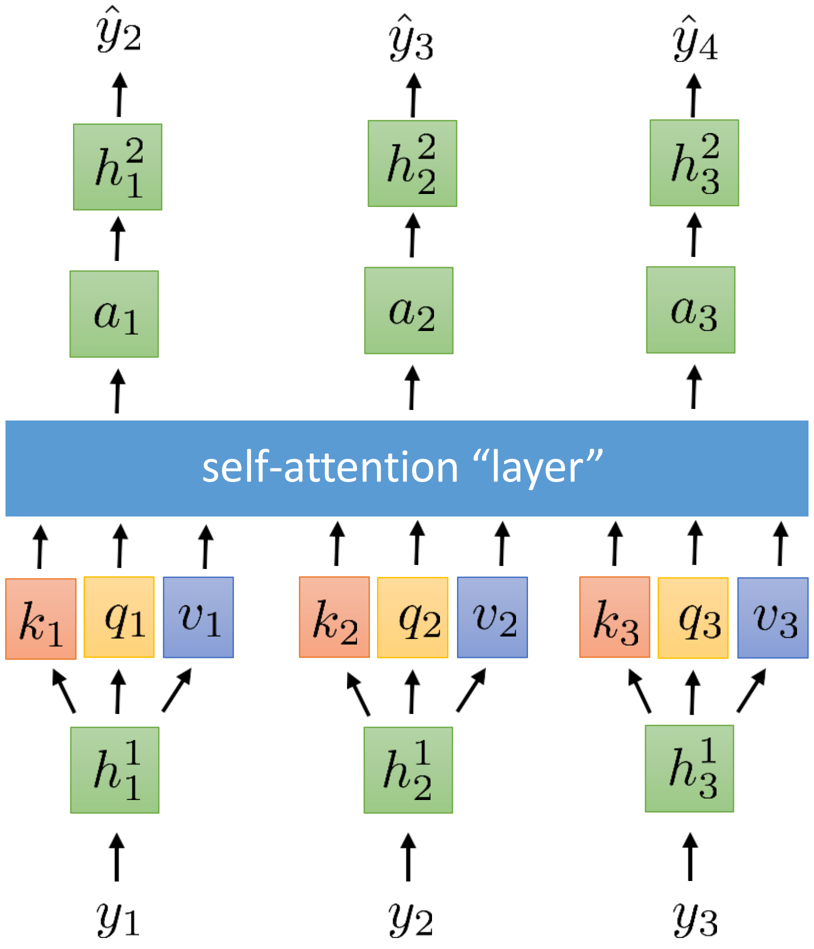

Feed Forward
M_H
M_1
optics
advanced
who
Strickland
awards
committee
Nobel
A
Q
K
V
Layer p
p+1
Nobel
committee
awards
Strickland
who
advanced
optics

From Self-Attention to Transformers

- We will talk about a class of models for processing sequences that does not use recurrent connections but instead relies entirely on attention and will build up towards a class of models called transformers.
- To address a few key limitations, we need to add certain elements:

1. Positional encoding	addresses lack of sequence information
2. Multi-headed attention	allows querying multiple positions at each layer
3. Adding nonlinearities	so far, each successive layer is *linear* in the previous one
4. Masked decoding	how to prevent attention lookups into the future?

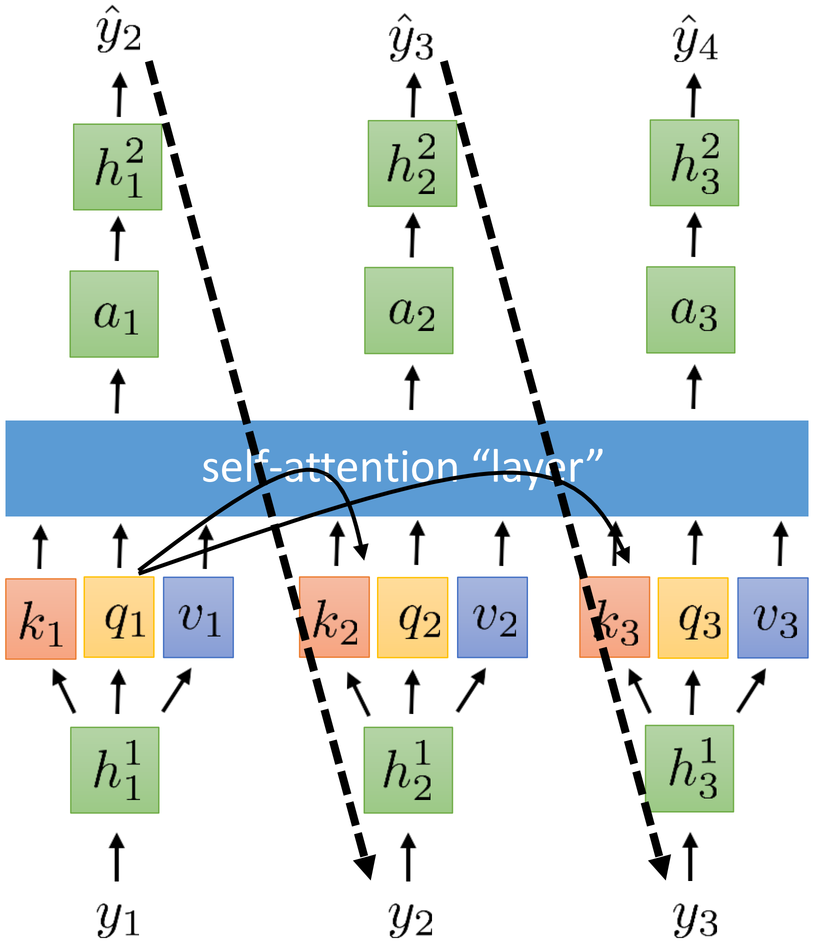
Self-attention can see the future!



A **crude** self-attention "language model":

In practice, there would be several alternating self-attention layers and position-wise feedforward networks

Self-attention can see the future!



A **crude** self-attention "language model":

In practice, there would be several alternating self-attention layers and position-wise feedforward networks

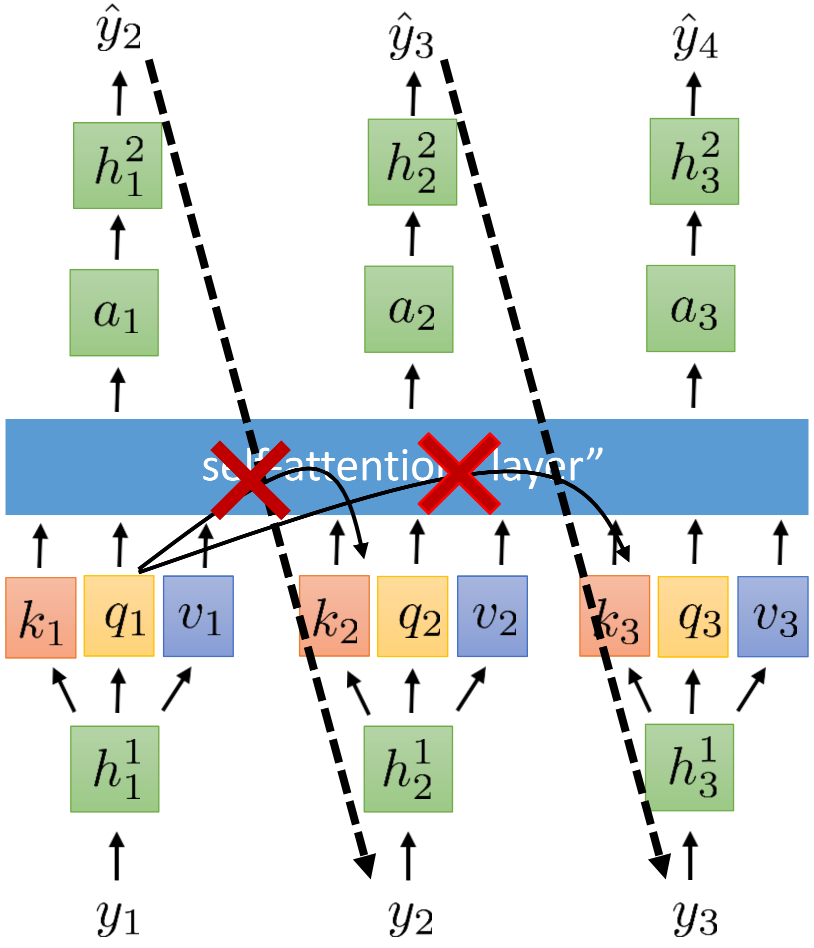
Big problem: self-attention at step 1 can look at the value at steps 2 & 3, which is based on the **inputs** at steps 2 & 3

At test time (when decoding), the **inputs** at steps 2 & 3 will be based on the output at step 1…

…which requires knowing the **input** at steps 2 & 3

Masked Attention

A **crude** self-attention "language model":



At test time (when decoding), the **inputs** at steps 2 & 3 will be based on the output at step 1…

…which requires knowing the **input** at steps 2 & 3

Must allow self-attention into the **past**…

…but not into the **future**

Easy solution:

$$\cancel{e_{l,t} = q_l \cdot k_t}$$

$$e_{l,t} = \begin{cases} q_l \cdot k_t & \text{if } l \geq t \\ -\infty & \text{otherwise} \end{cases}$$

in practice:

just replace $\exp(e_{l,t})$ with 0 if $l < t$

inside the softmax