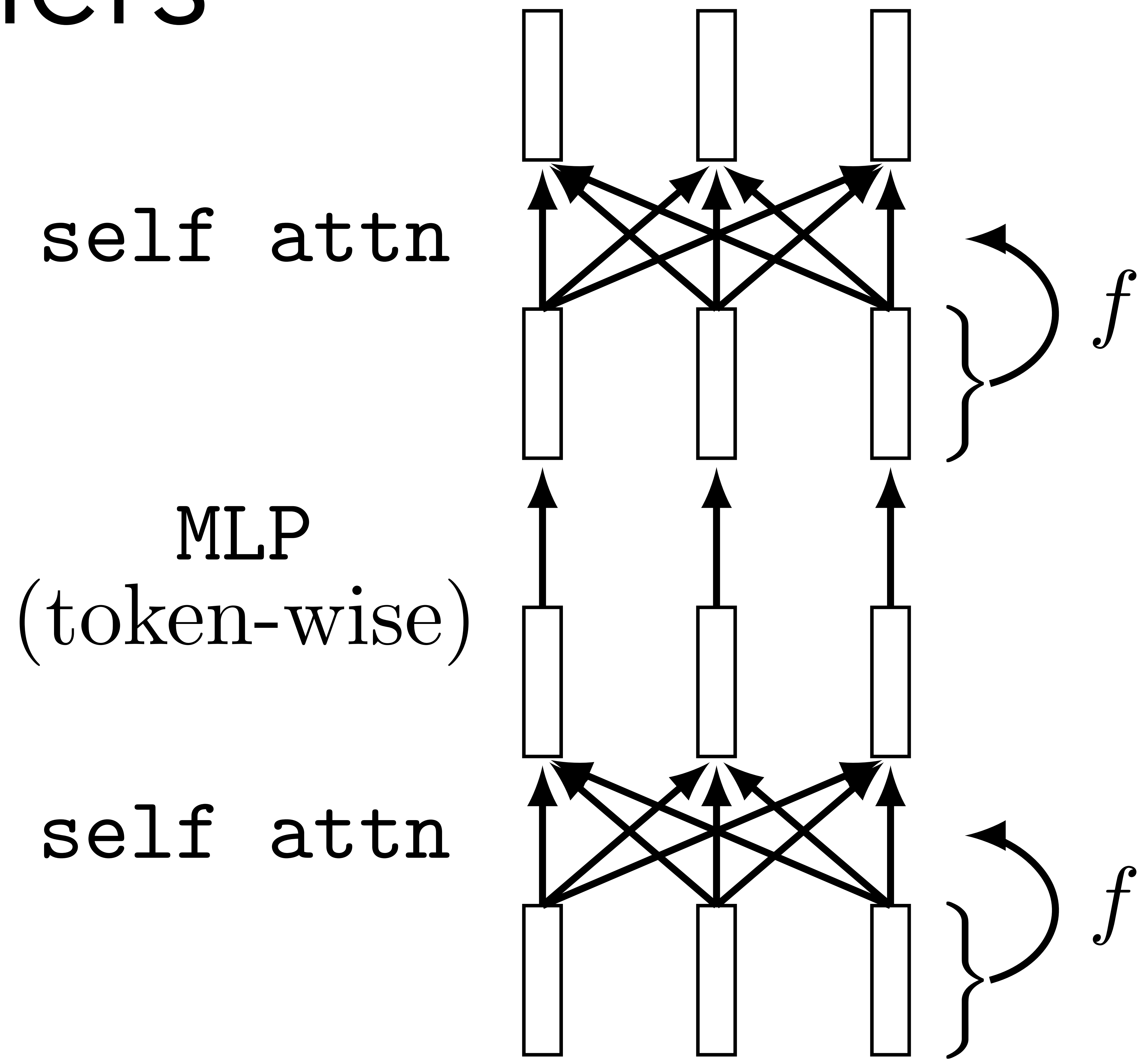
Lecture 8: Transformers

Speaker: Phillip Isola

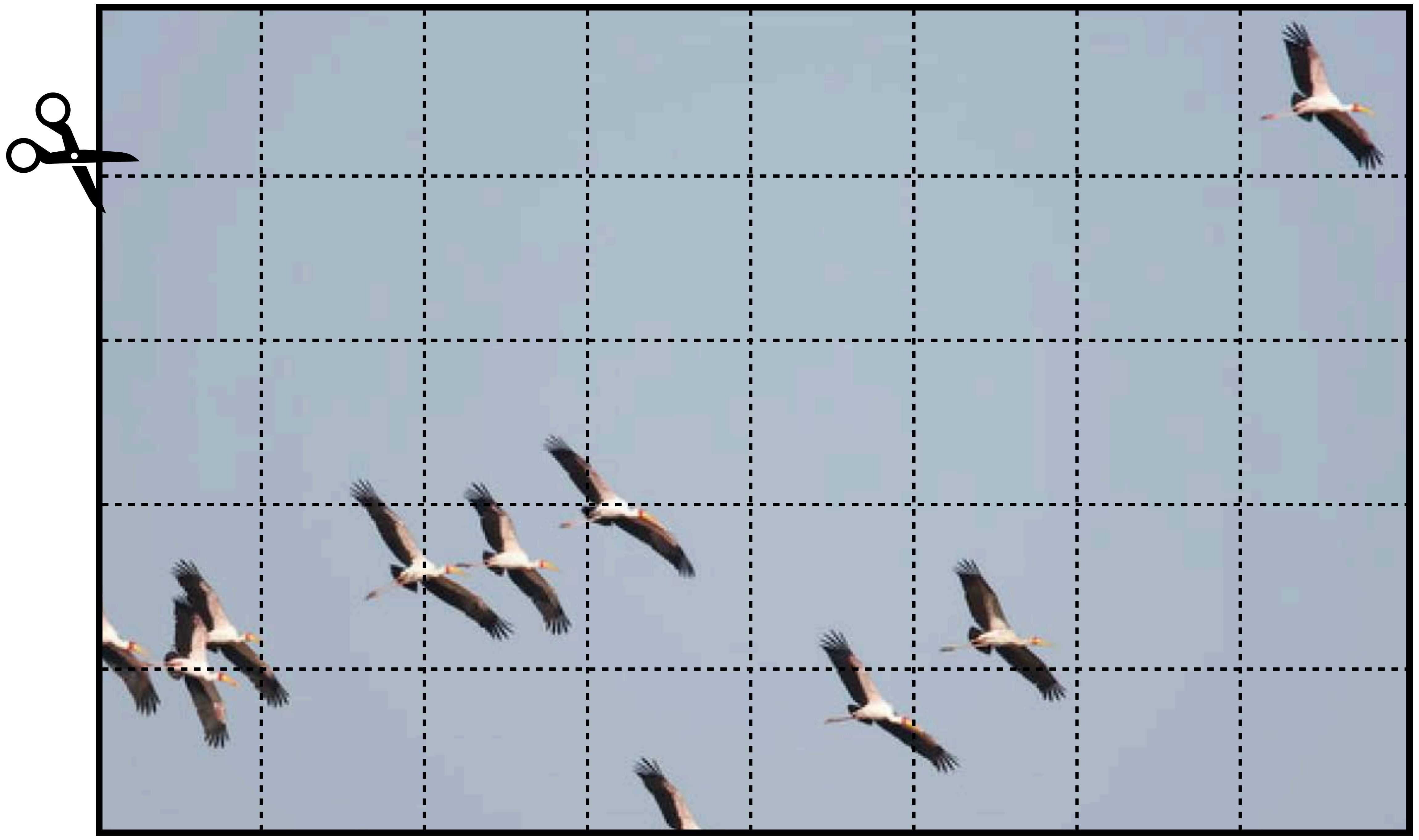


9. Transformers

- Three key ideas
  - Tokens
  - Attention
  - Positional encoding
- Examples of architectures and applications

Don Quixote by Pierre Menard

A Limitation of CNNs



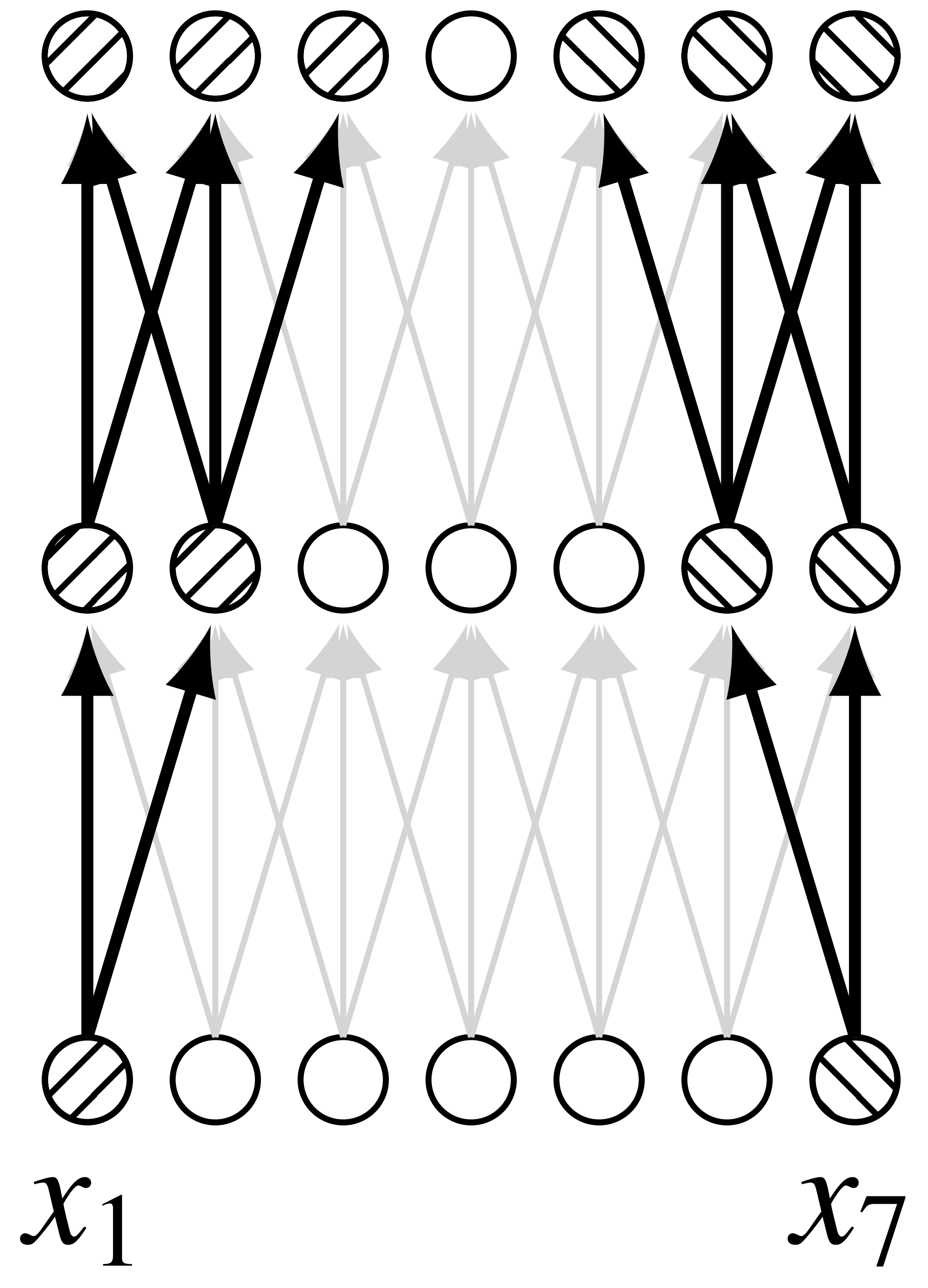
© Fredo Durand. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use/

How many birds are in this image?

Is the top right bird the same species as the bottom left bird?

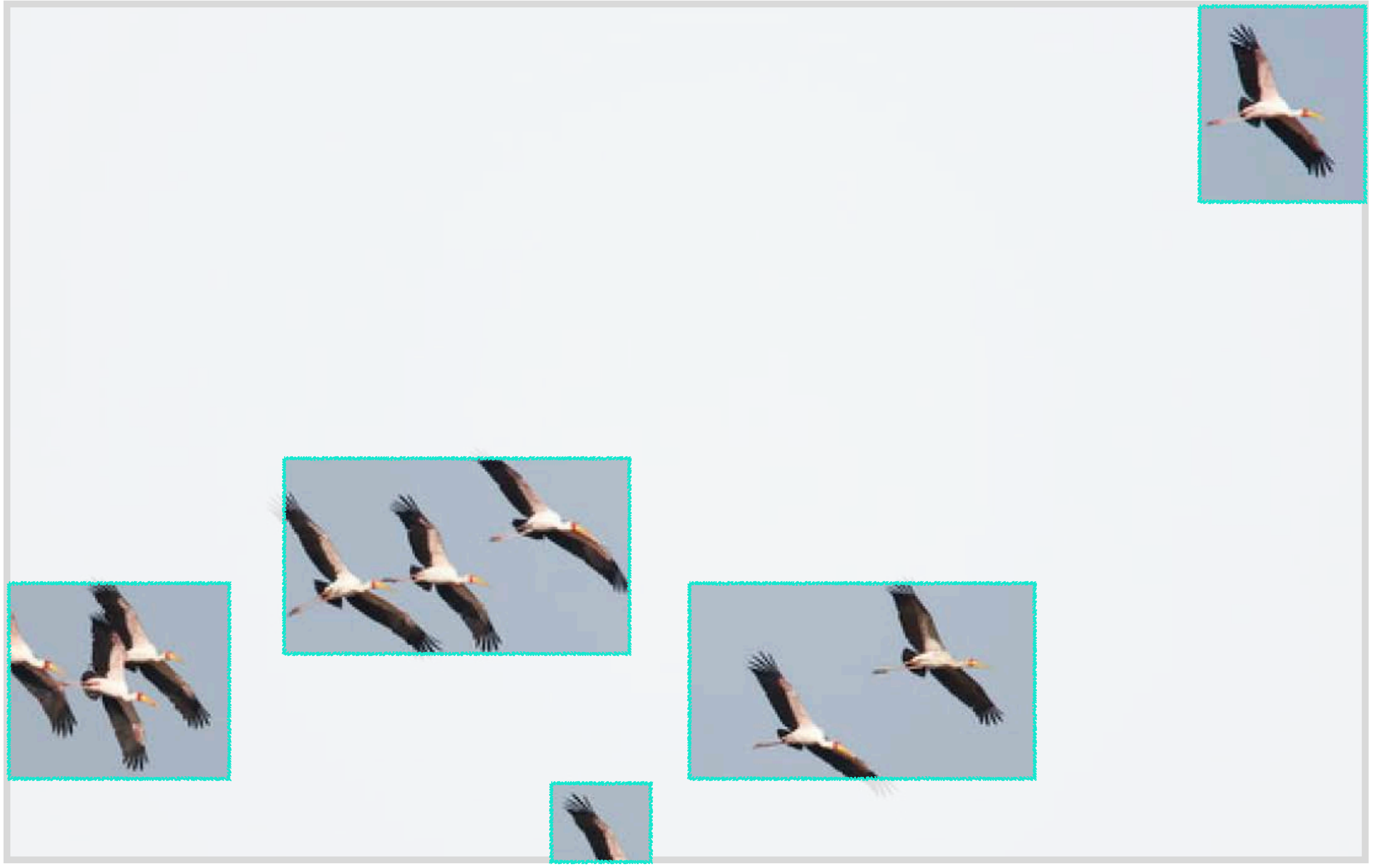
CNNs are built around the idea of locality, and are not well-suited to modeling long distance relationships

A Limitation of CNNs



Far apart image patches do not interact

The Idea of Attention



© Fredo Durand. All rights reserved. This content is excluded from our Creative Commons license. For more information, see
https://ocw.mit.edu/help/faq-fair-use/

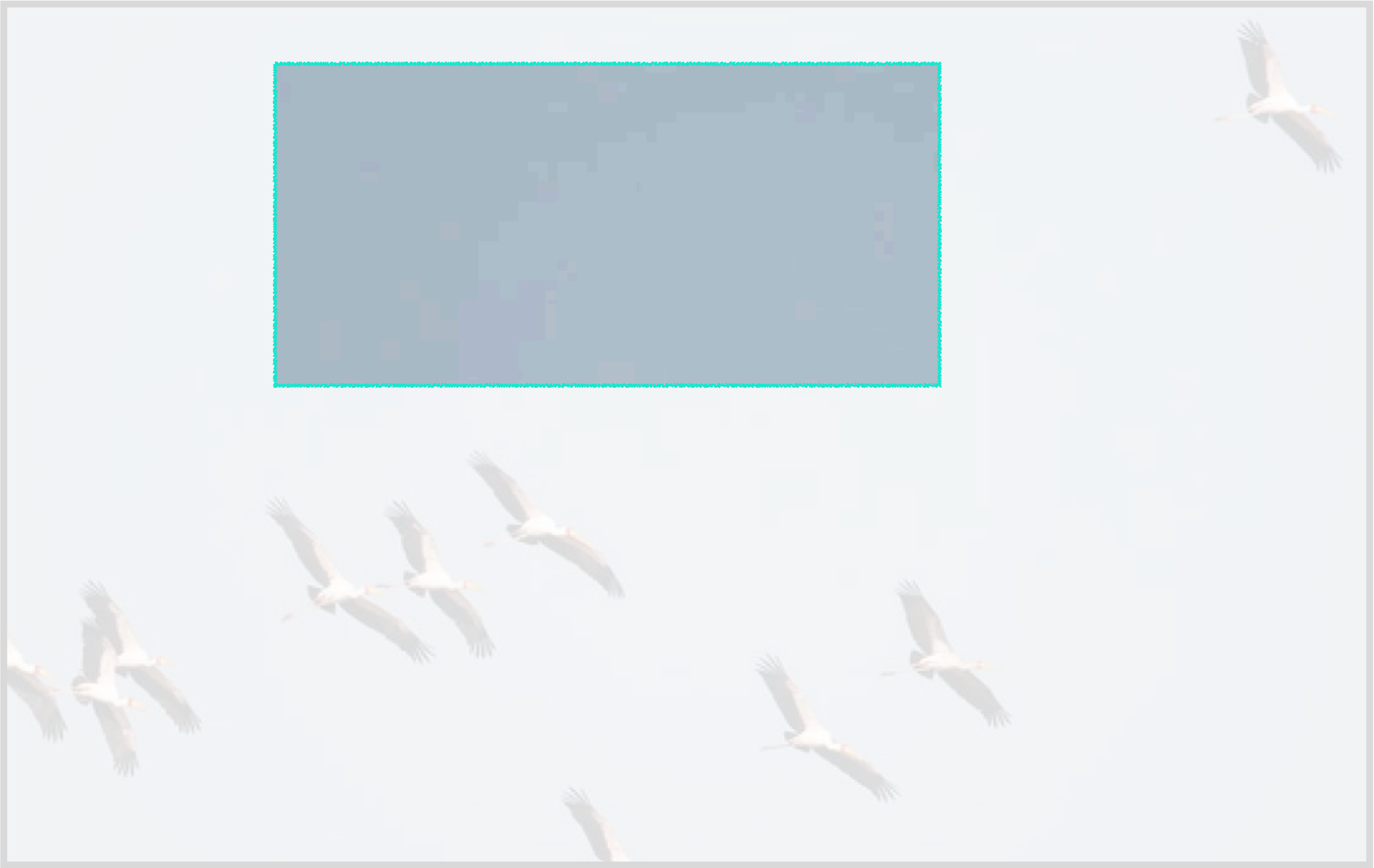
How many birds are in this image?

The Idea of Attention

© Fredo Durand. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use/

Is the top right bird the same species as the bottom left bird?

The Idea of Attention



© Fredo Durand. All rights reserved. This content is excluded from our Creative Commons license. For more information, see
https://ocw.mit.edu/help/faq-fair-use/

What's the color of the sky?

Three Key Architectural Innovations

1. Tokens
2. Attention
3. Positional Codes

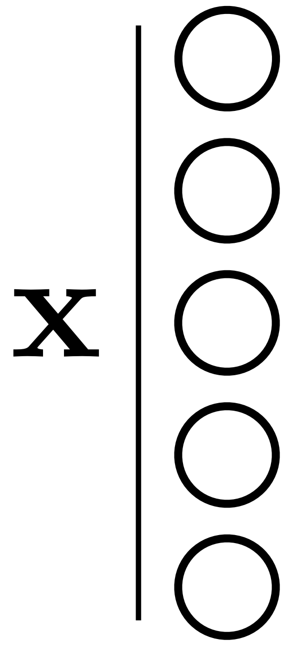
New idea #1: tokens

A New Data Type: Tokens

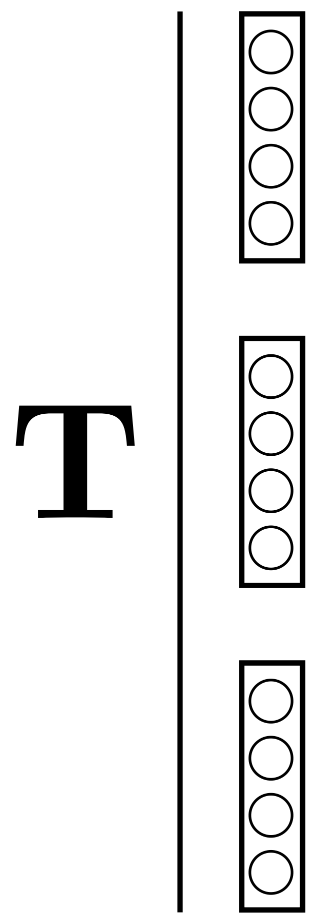
- A **token** is just a vector of neurons. (note: GNNs also operate over tokens, but over there we called them “node attributes” or node “feature descriptors”)

- But the connotation is that a token is an encapsulated bundle of information; with transformers we will operate over tokens rather than over neurons.

array of **neurons**



array of **tokens**



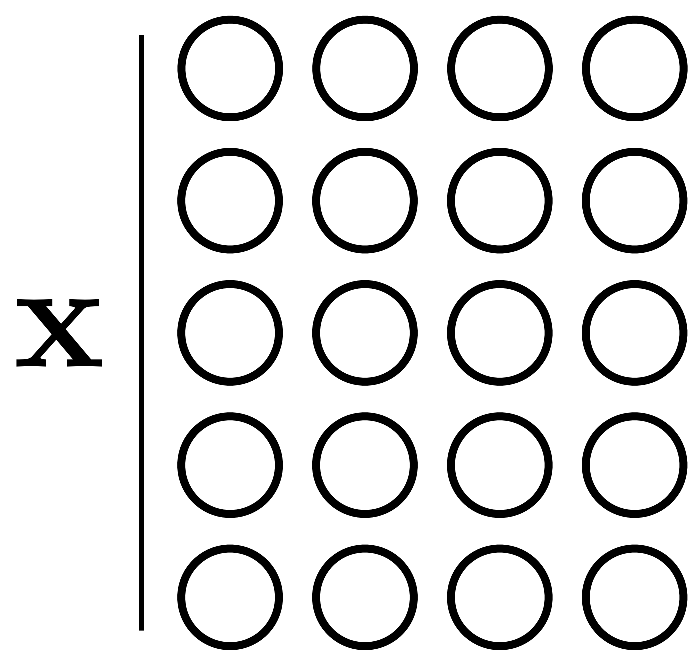
Note: sometimes the word “token” is instead used to refer to the atomic units of the data sequence we will model. In this usage tokens are the representation of the data only at the input and output layers. We use a more general definition where tokens are the representation of the data at *any* layer.

A new data structure: Tokens

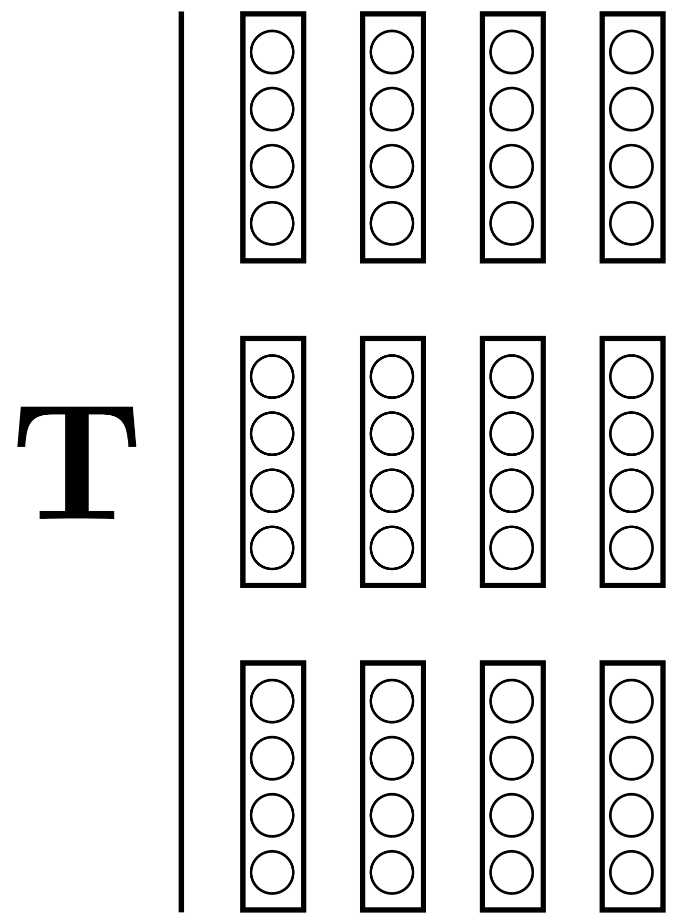
- A **token** is just a vector of neurons. (note: GNNs also operate over tokens, but over there we called them “node attributes” or node “feature descriptors”)

- But the connotation is that a token is an encapsulated bundle of information; with transformers we will operate over tokens rather than over neurons.

array of **neurons**



array of **tokens**

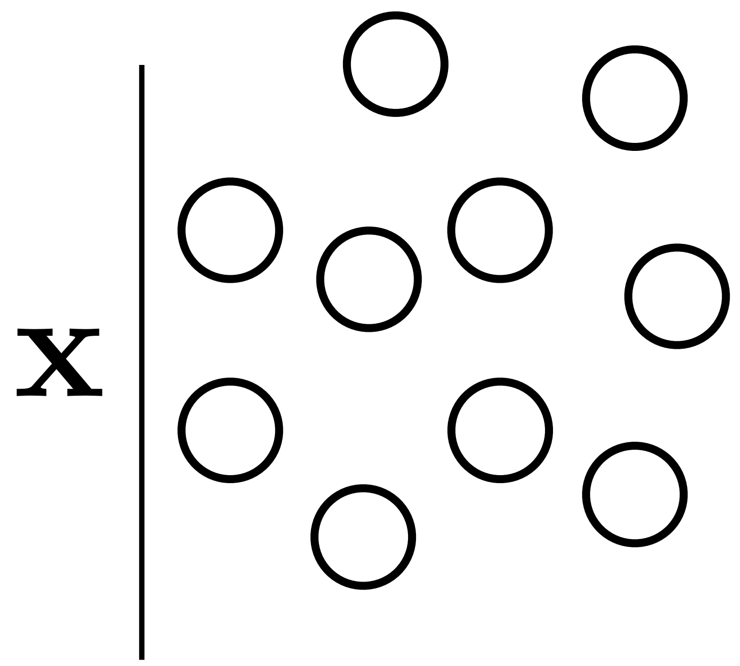


A new data structure: Tokens

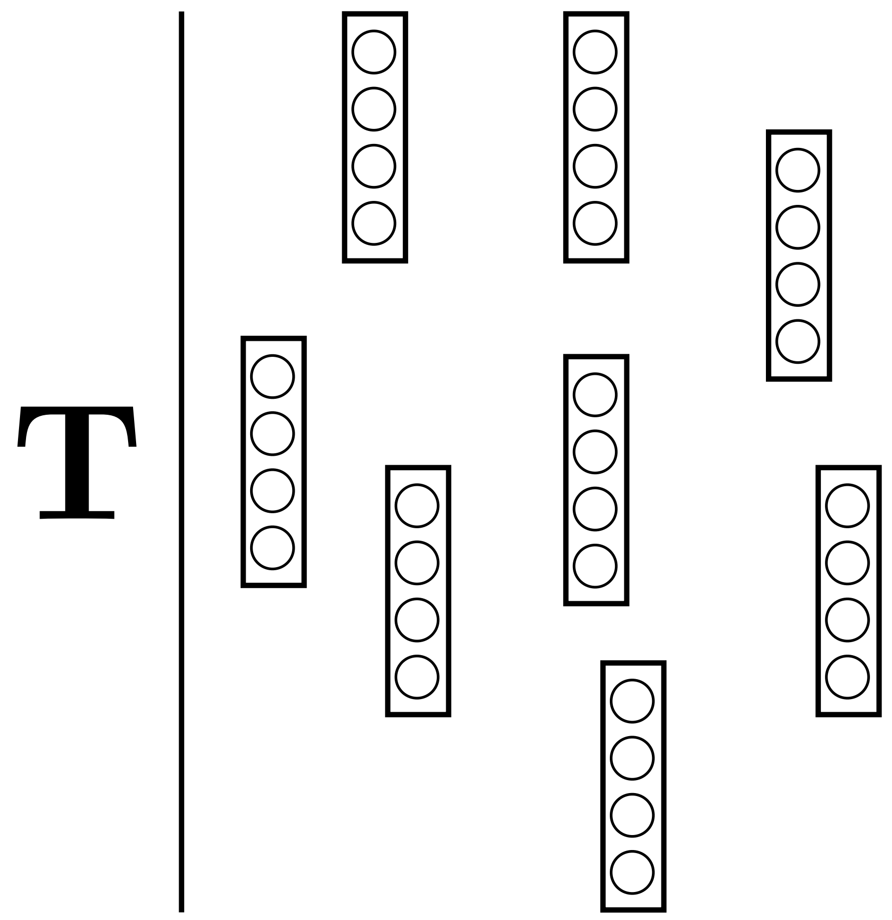
- A **token** is just a vector of neurons. (note: GNNs also operate over tokens, but over there we called them “node attributes” or node “feature descriptors”)

- But the connotation is that a token is an encapsulated bundle of information; with transformers we will operate over tokens rather than over neurons.

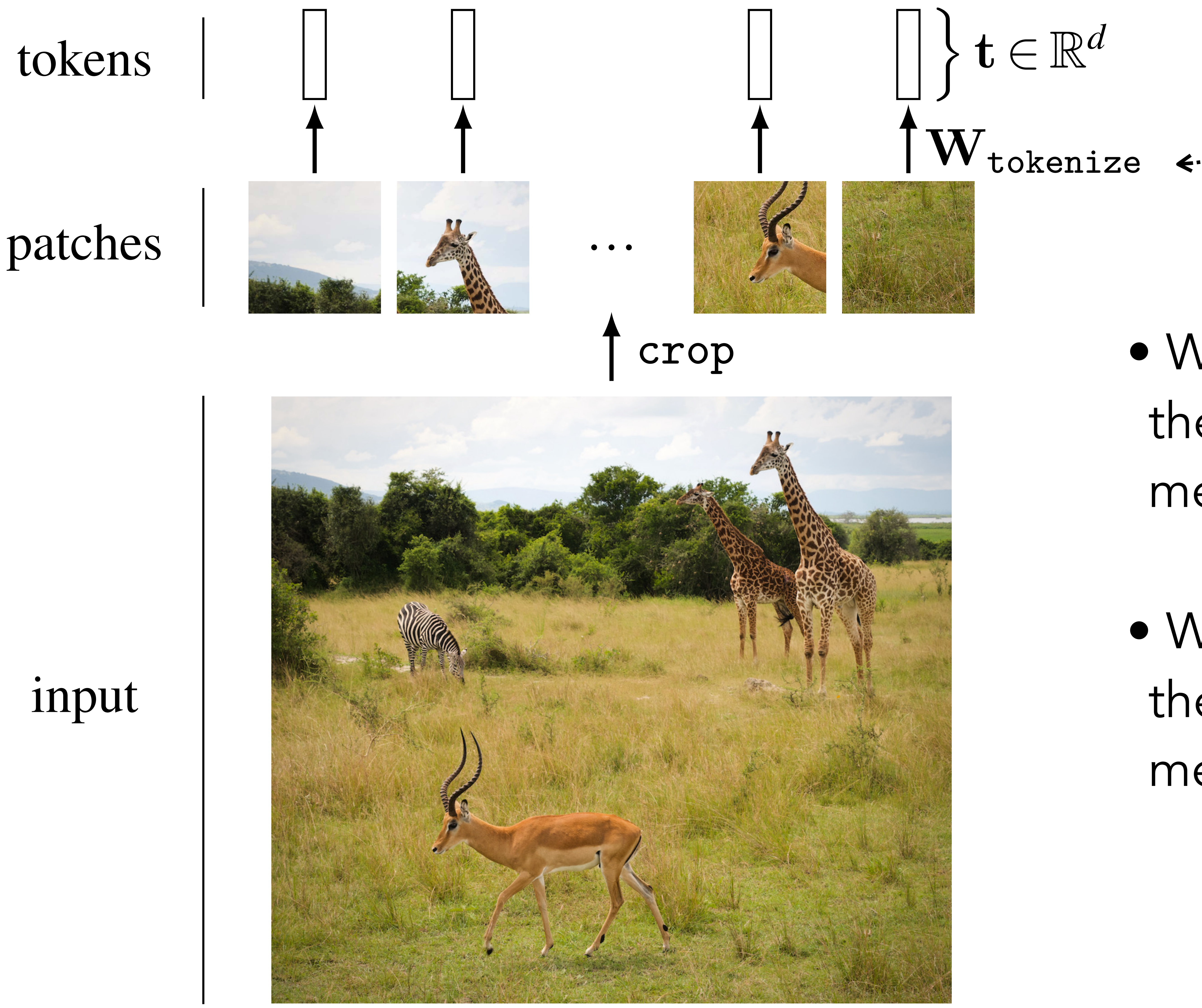
set of **neurons**



set of **tokens**



Tokenizing the input data



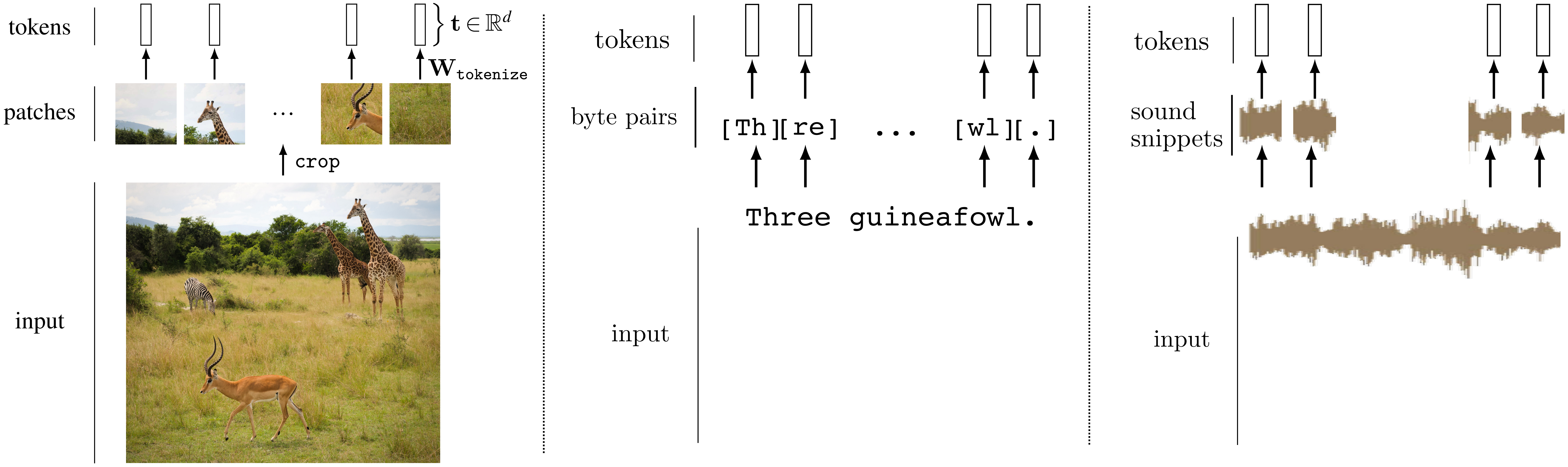
- When operating over *neurons*, we represent the input as an array of scalar-valued measurements (e.g., pixels)

- When operating over *tokens*, we represent the input as an array of vector-valued measurements

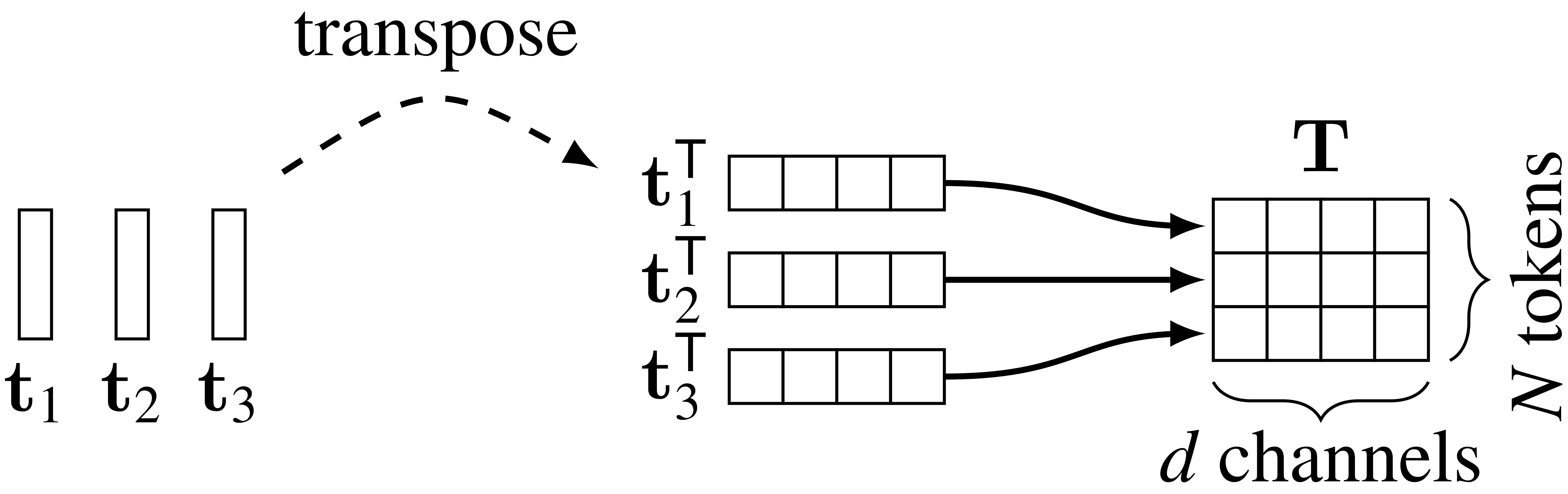
Tokenizing the input data

You can tokenize anything.

General strategy: chop the input up into chunks, project each chunk to a vector.



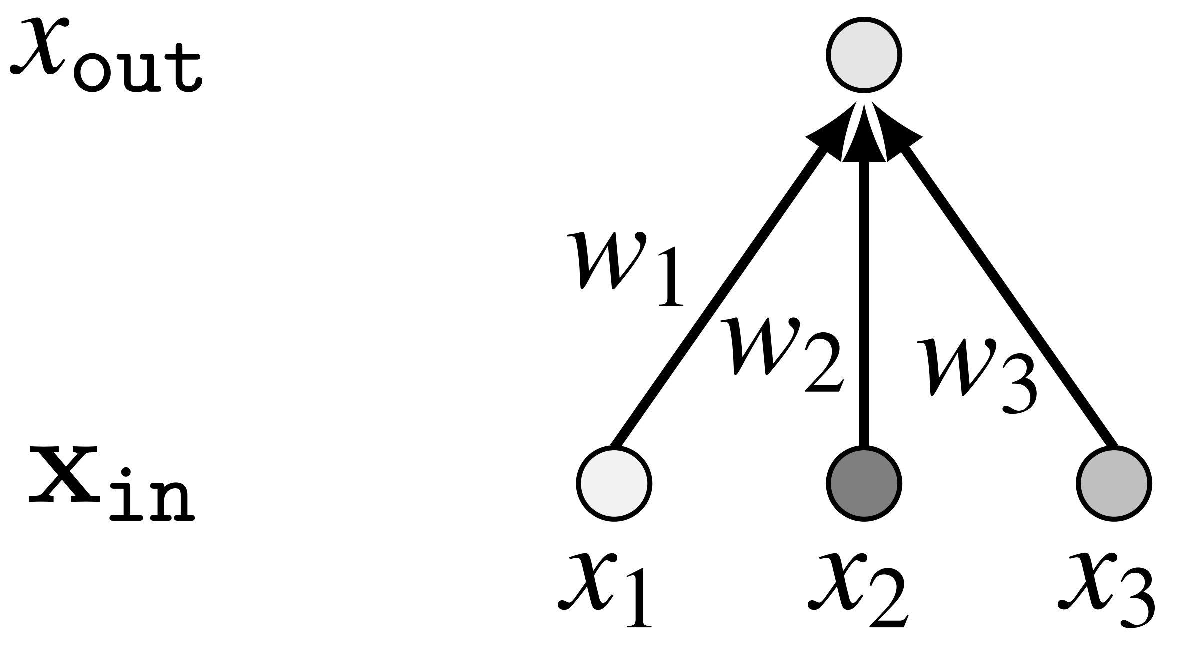
Notation



transpose
$\mathbf{t}_1$ $\mathbf{t}_2$ $\mathbf{t}_3$
$\mathbf{t}_1^\top$
$\mathbf{t}_2^\top$
$\mathbf{t}_3^\top$
$\mathbf{T}$
N tokens
d channels

Linear combination of tokens

Linear combination of neurons

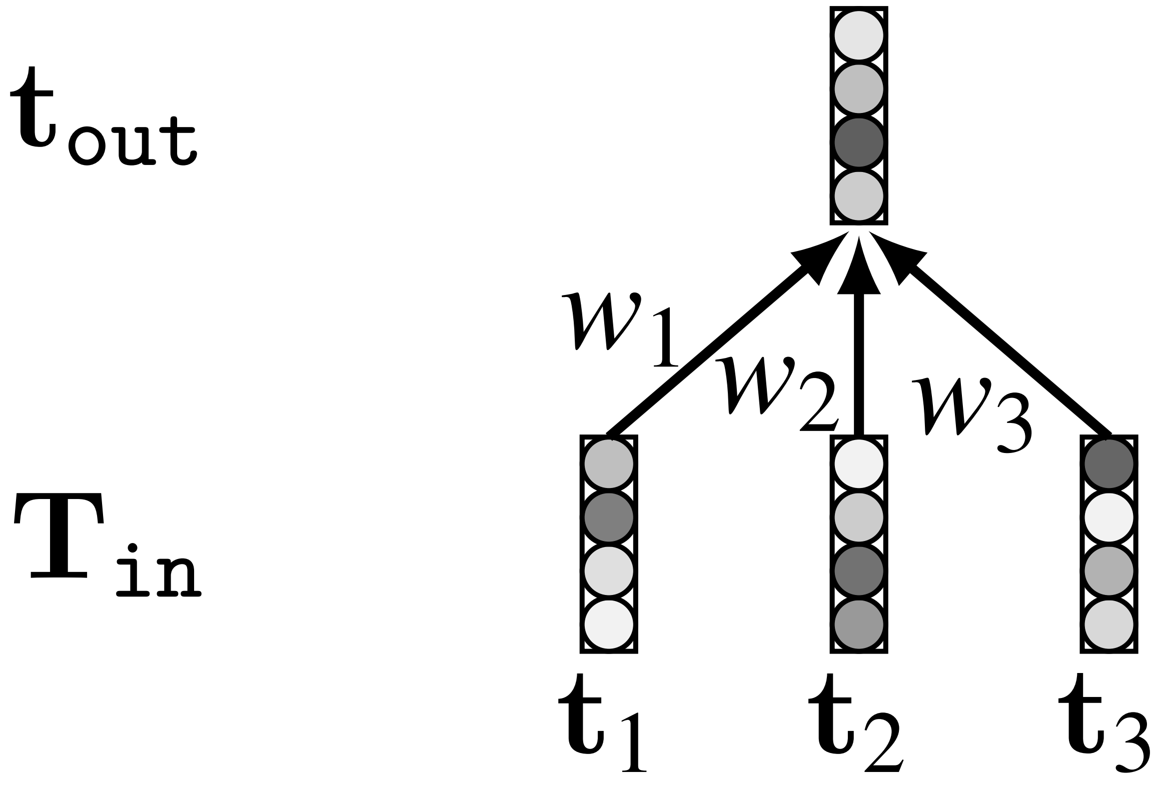


$$x_{\texttt{out}} = w_1 x_1 + w_2 x_2 + w_3 x_3$$

$$x_{\texttt{out}}[i] = \sum_{j=1}^{N} w_{ij} x_{\texttt{in}}[j]$$

$$\mathbf{x}_{\texttt{out}} = \mathbf{W}\mathbf{x}_{\texttt{in}}$$

Linear combination of tokens



$$\mathbf{t}_{\texttt{out}} = w_1 \mathbf{t}_1 + w_2 \mathbf{t}_2 + w_3 \mathbf{t}_3$$

$$\mathbf{T}_{\texttt{out}}[i,:] = \sum_{j=1}^{N} w_{ij} \mathbf{T}_{\texttt{in}}[j,:]$$

$$\mathbf{T}_{\texttt{out}} = \mathbf{W}\mathbf{T}_{\texttt{in}}$$

Token-wise nonlinearity

$$\mathbf{x}_{\texttt{out}} = \begin{bmatrix} \texttt{relu}(x_{\texttt{in}}[0]) \\ \vdots \\ \texttt{relu}(x_{\texttt{in}}[N-1]) \end{bmatrix}$$

$$\mathbf{T}_{\texttt{out}} = \begin{bmatrix} F_{\theta}(\mathbf{T}_{\texttt{in}}[0,:]) \\ \vdots \\ F_{\theta}(\mathbf{T}_{\texttt{in}}[N-1,:]) \end{bmatrix}$$

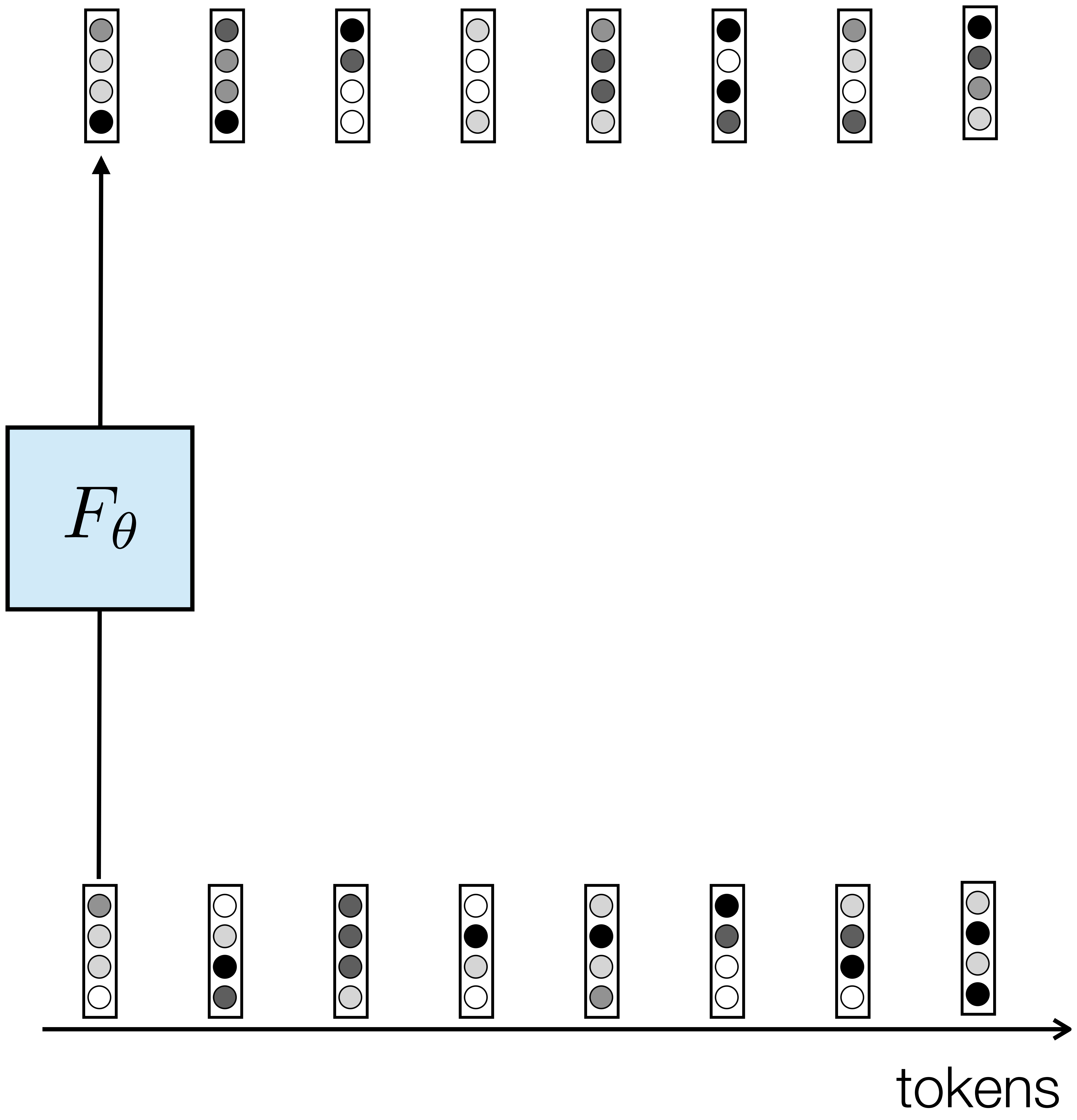
F is typically an MLP

Equivalent to a CNN with 1x1 kernels run over token sequence

Token-wise nonlinearity

$$\mathbf{x}_{\texttt{out}} = \begin{bmatrix} \texttt{relu}(x_{\texttt{in}}[0]) \\ \vdots \\ \texttt{relu}(x_{\texttt{in}}[N-1]) \end{bmatrix}$$

$$\mathbf{T}_{\texttt{out}} = \begin{bmatrix} F_\theta(\mathbf{T}_{\texttt{in}}[0,:]) \\ \vdots \\ F_\theta(\mathbf{T}_{\texttt{in}}[N-1,:]) \end{bmatrix}$$



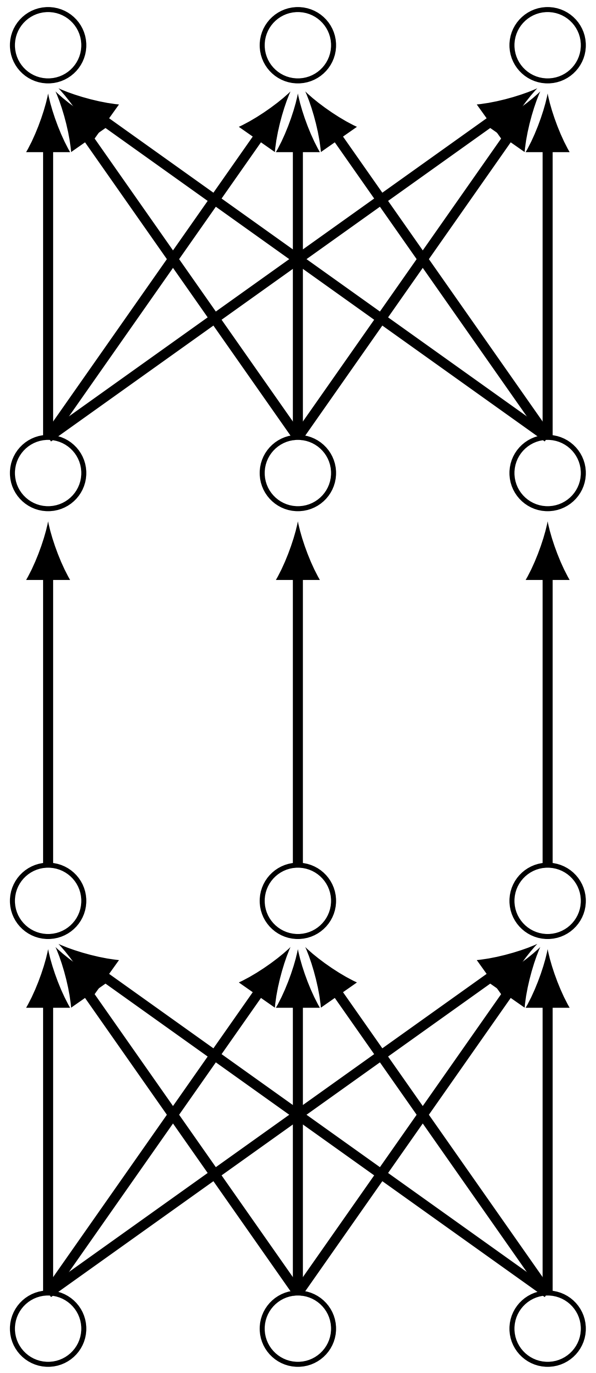
Token nets

Neural net

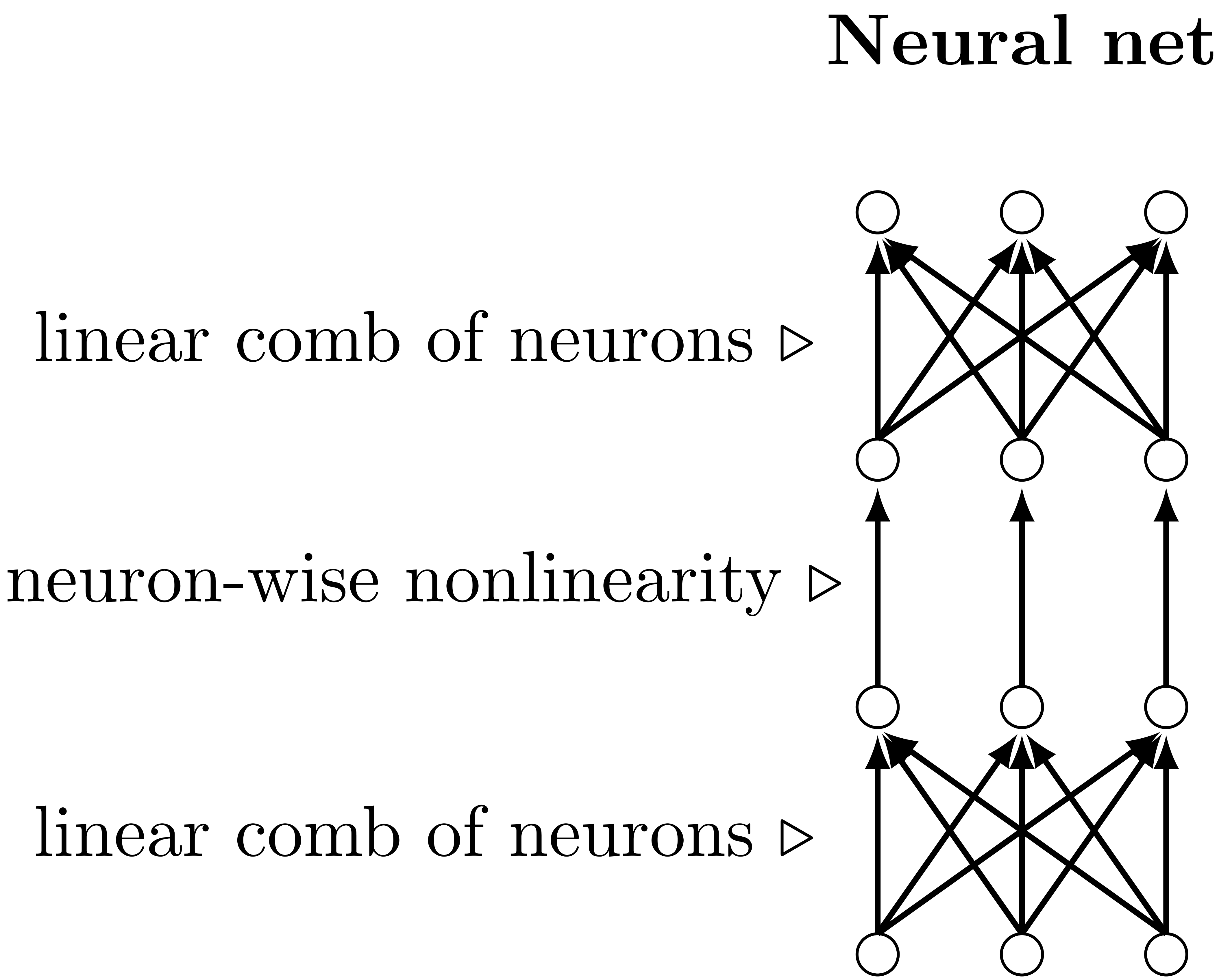
linear comb of neurons ▷

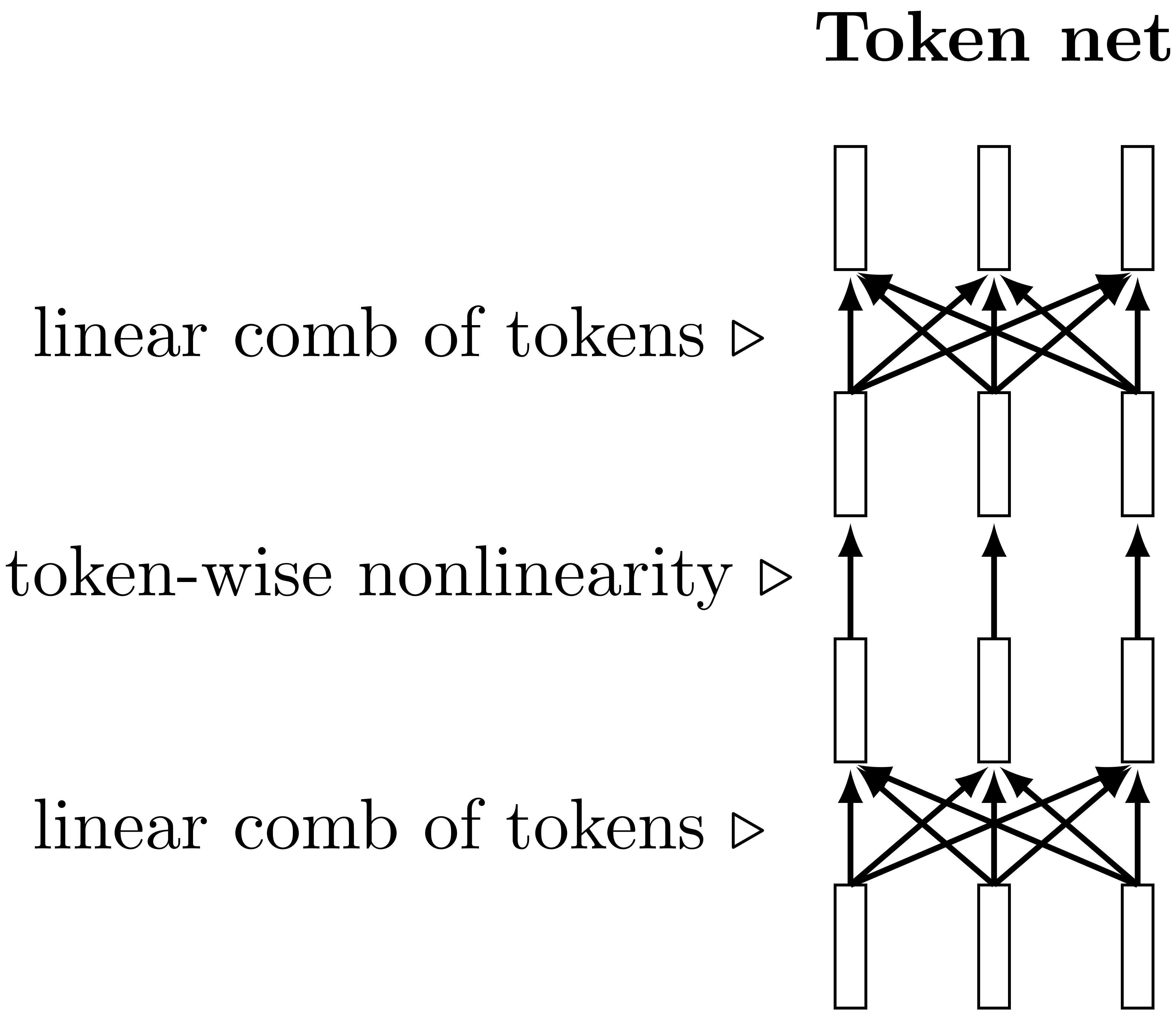
neuron-wise nonlinearity ▷

linear comb of neurons ▷

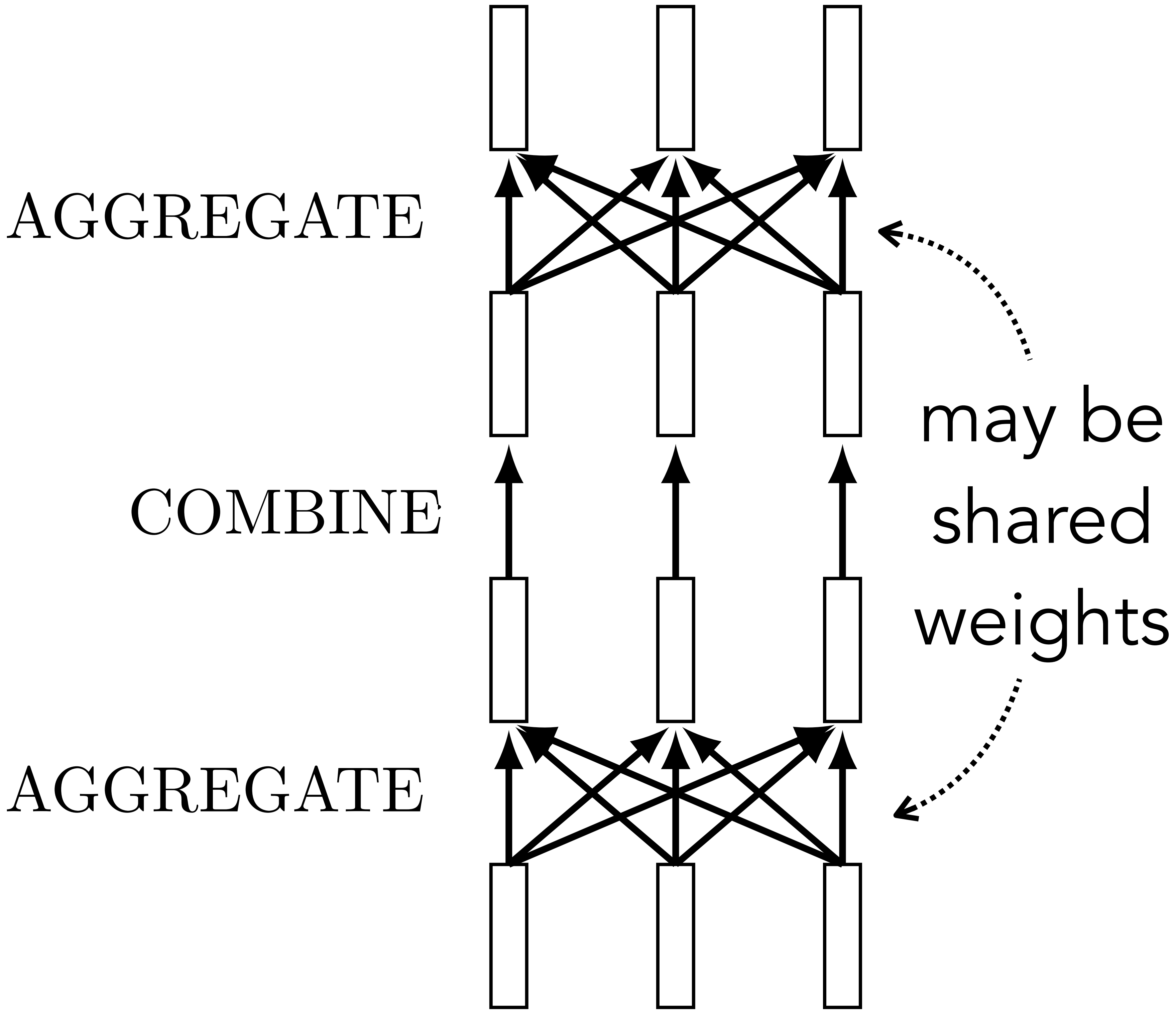


Token nets

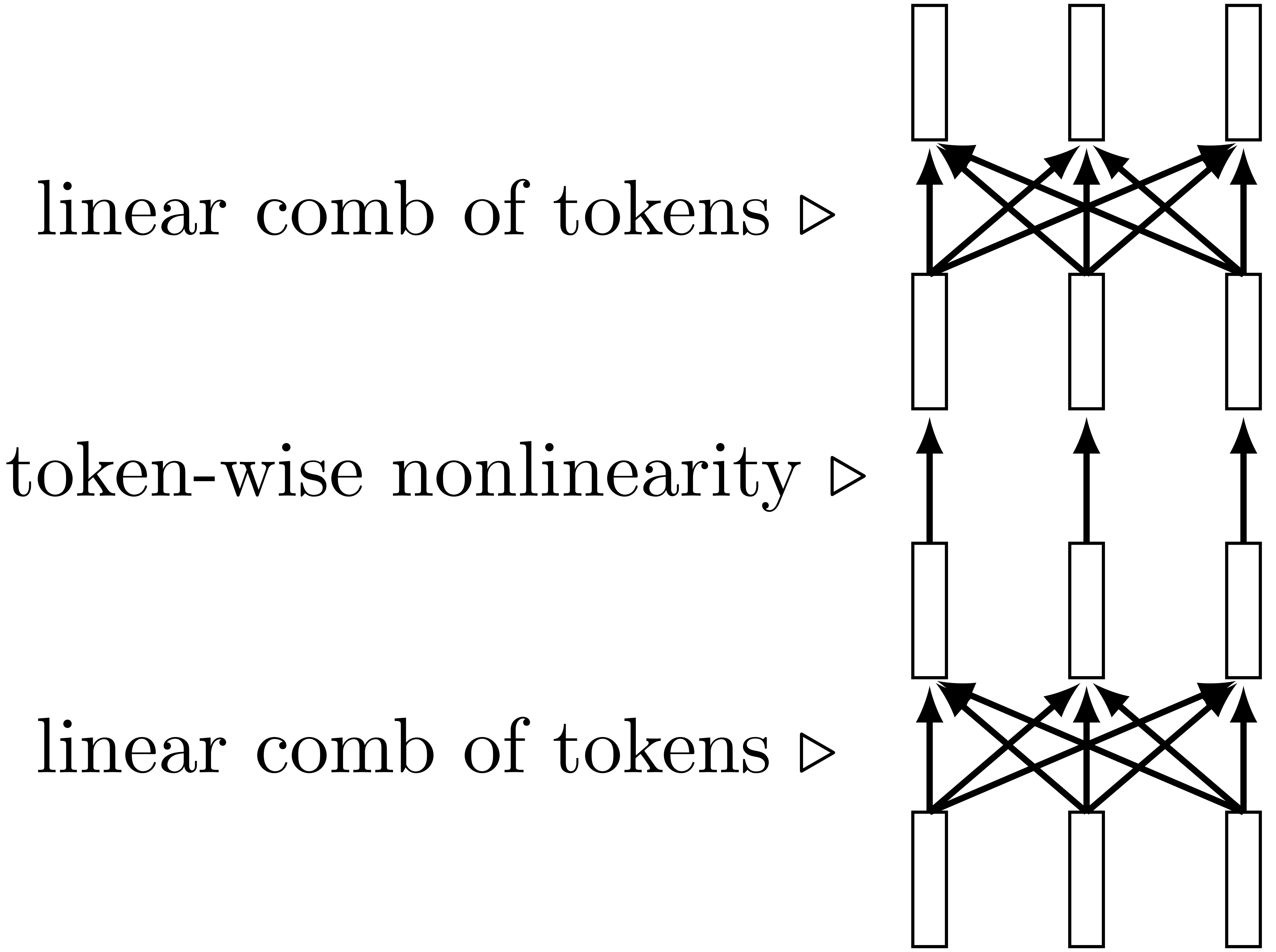

Neural net
linear comb of neurons ▷
neuron-wise nonlinearity ▷
linear comb of neurons ▷


Token net
linear comb of tokens ▷
token-wise nonlinearity ▷
linear comb of tokens ▷

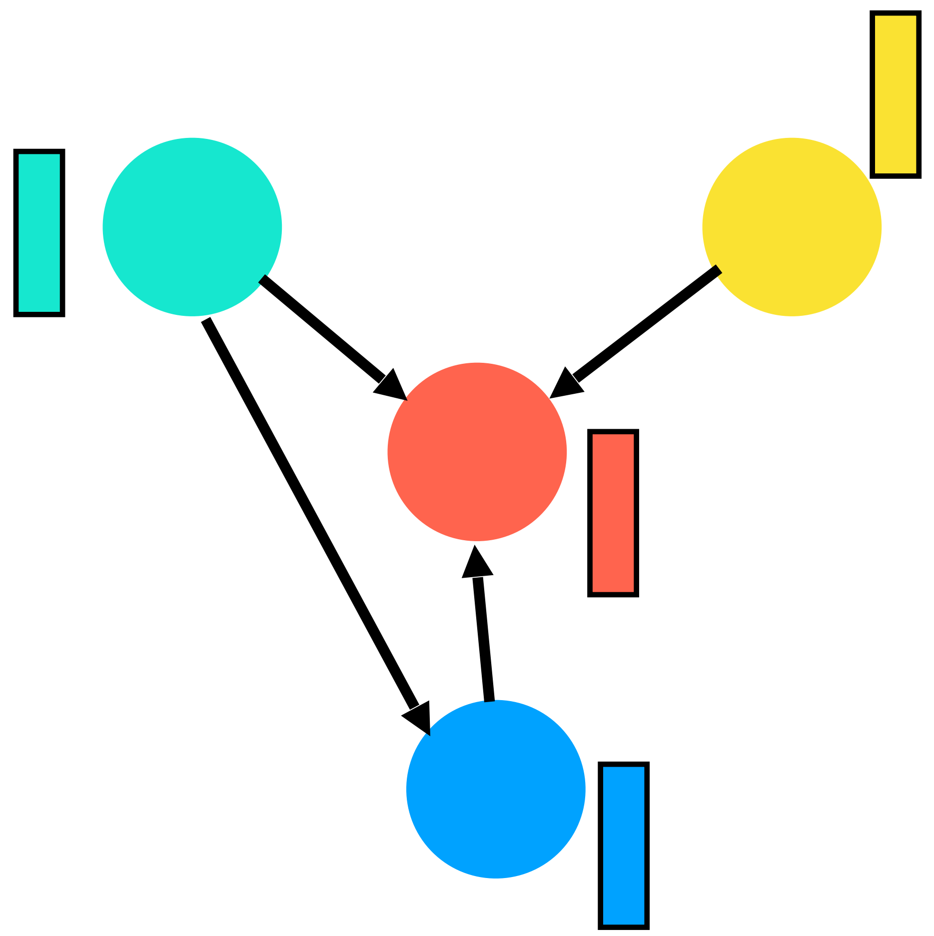
GNN

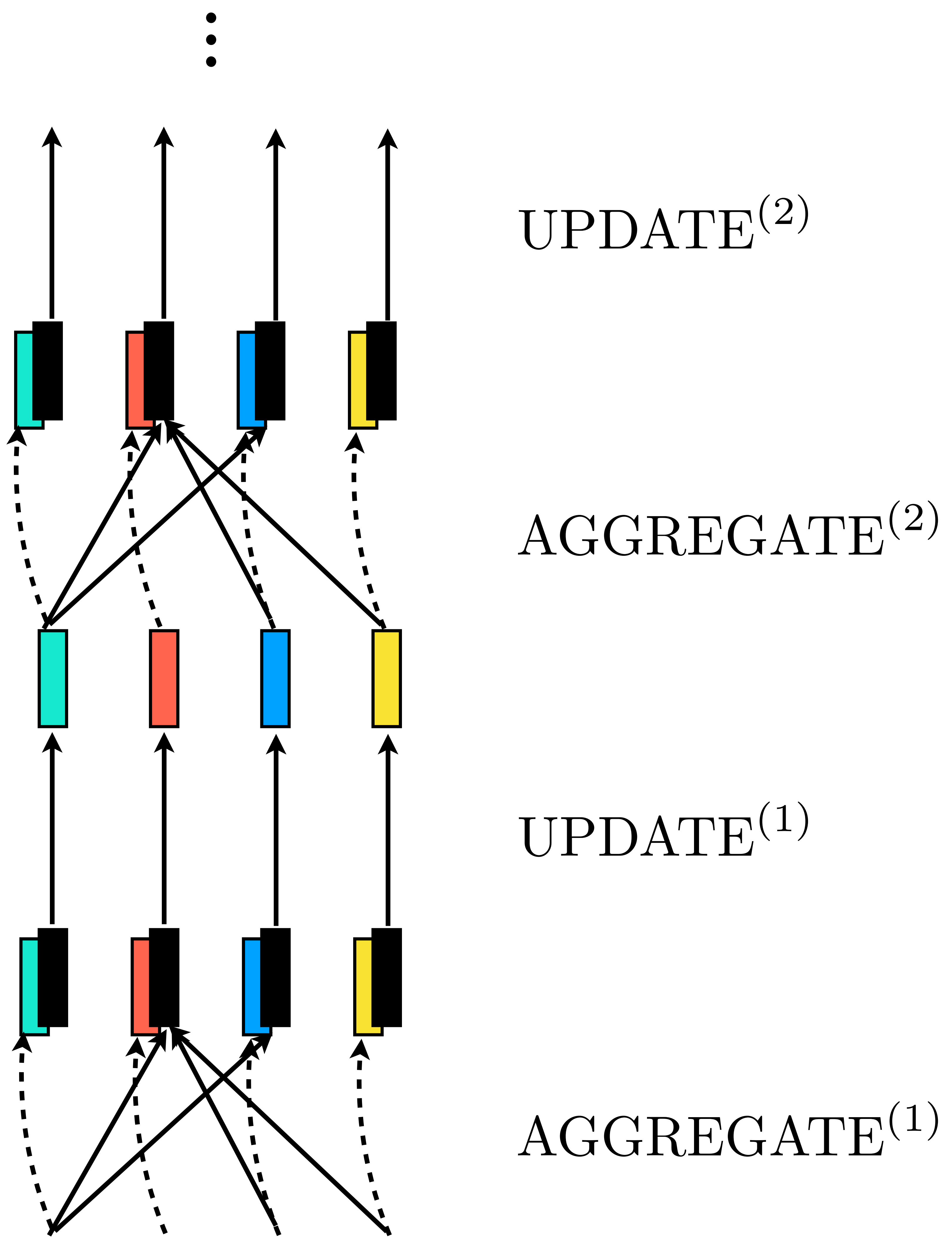


Token net



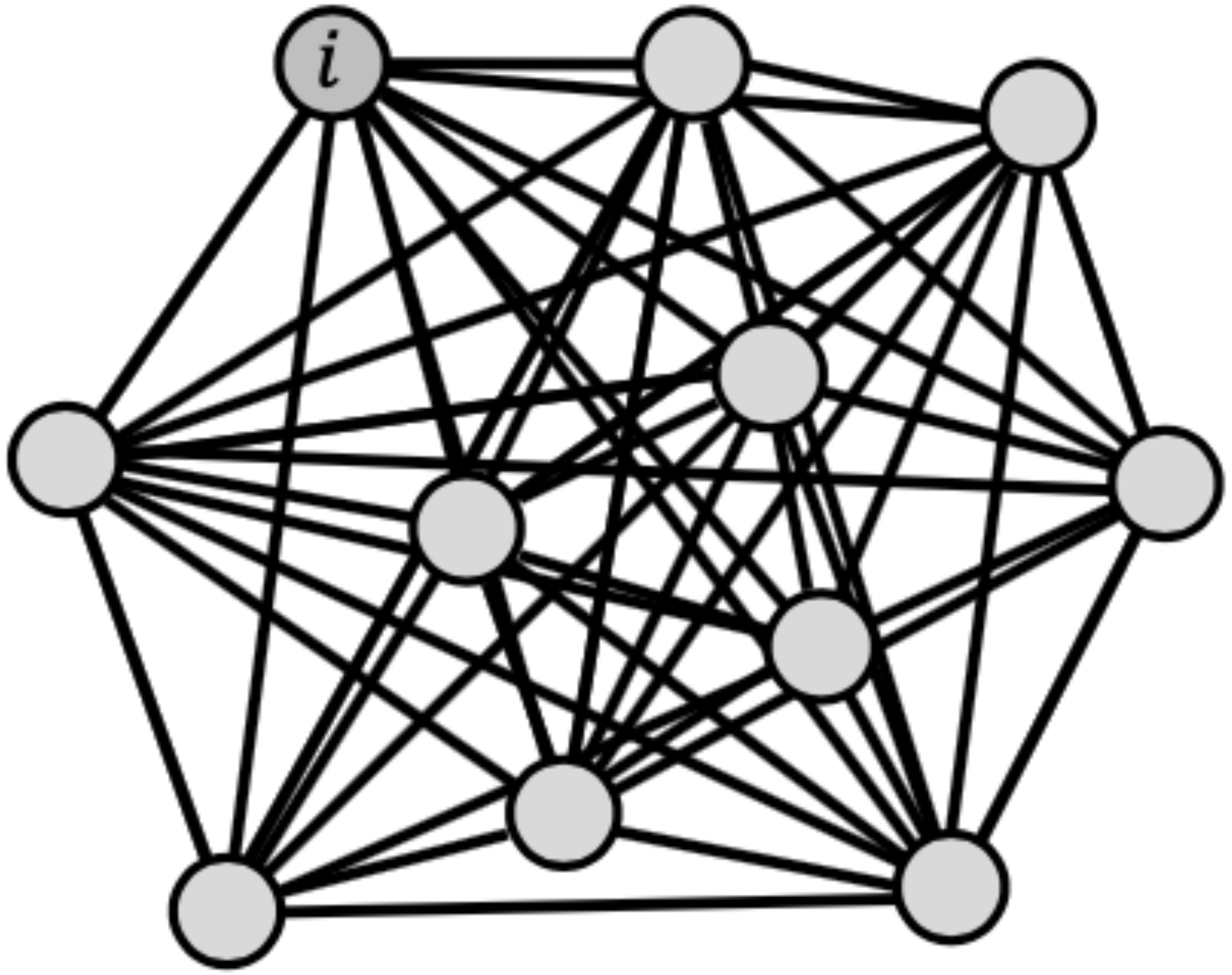
GNNs unrolled





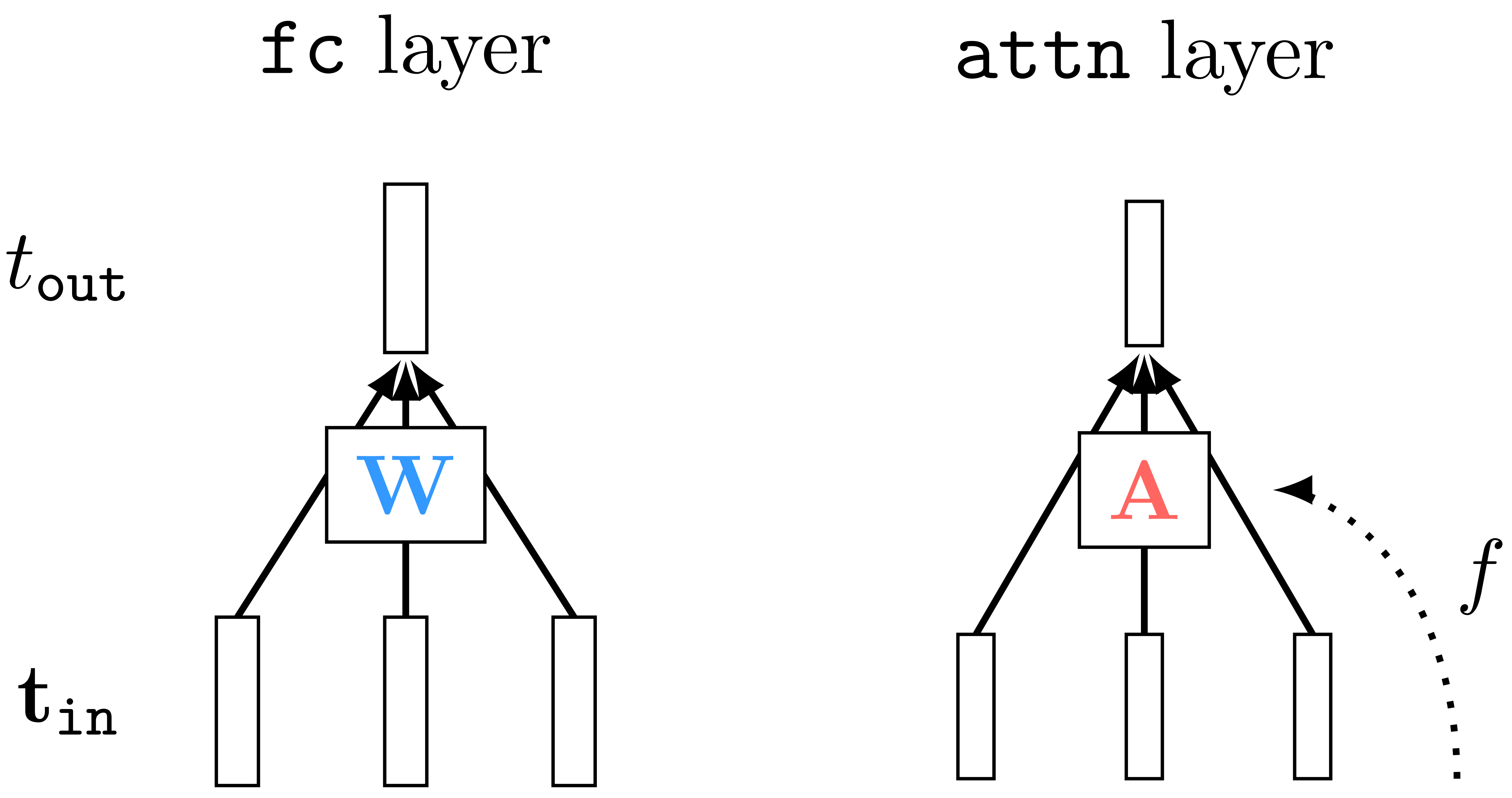
- Like an MLP, but nodes are vectors rather than scalars, edges are potentially complex functions (e.g., an edge can be an MLP)
- Each iteration of GNN message passing is a layer
  - AGGREGATE is akin to a linear layer
  - UPDATE is akin to a pointwise layer

A view from the graph perspective



Transformers may be viewed as Graph Neural Networks over fully-connected graphs

New idea #2: attention



$$\mathbf{A} = f(\ldots) \qquad \triangleleft \text{ attention}$$

$$\mathbf{T}_{\texttt{out}} = \mathbf{A}\mathbf{T}_{\texttt{in}}$$

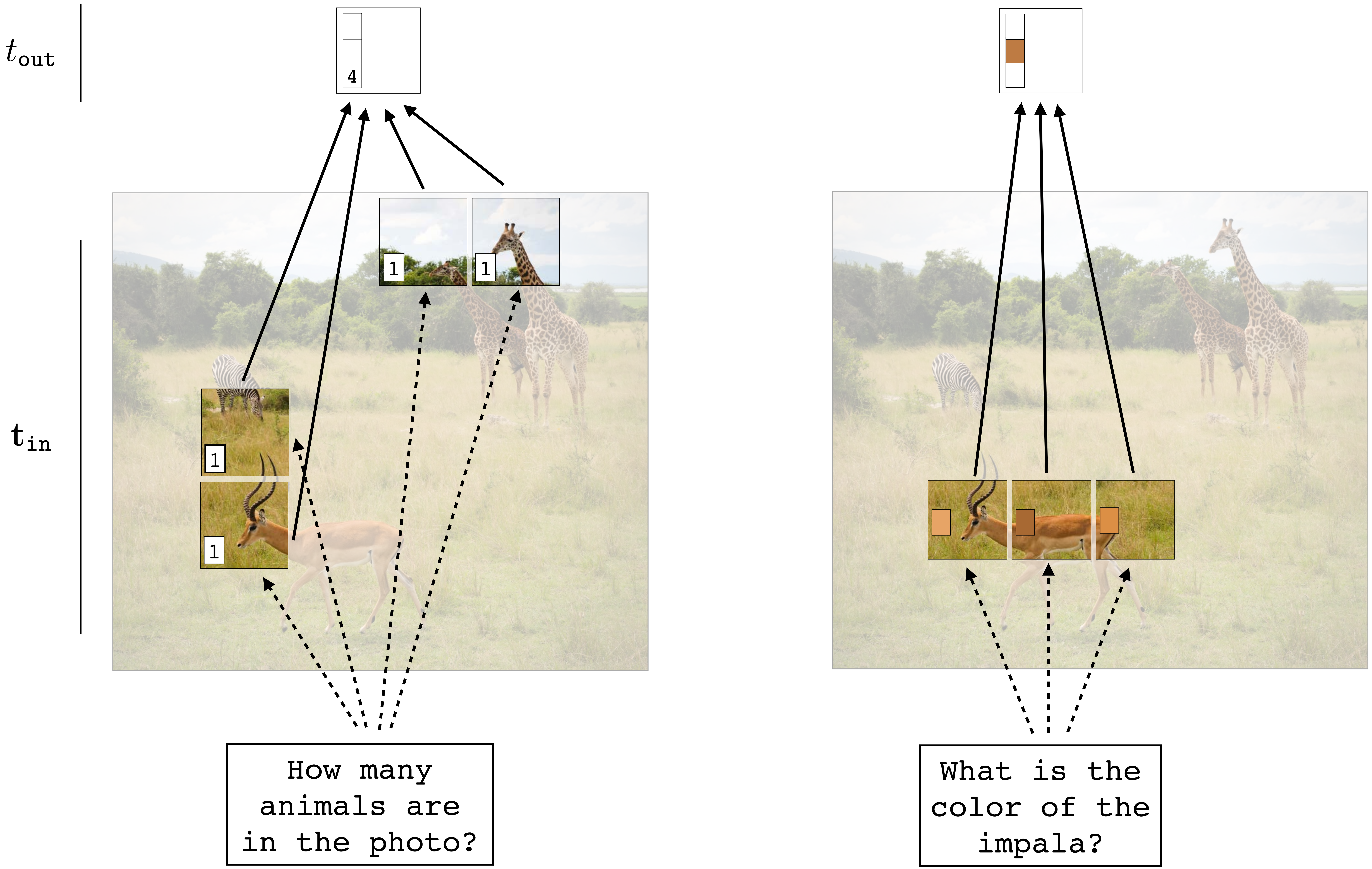
W is free *parameters*.

A is a function of some input *data*. The data tells us which tokens to attend to (assign high weight in weighted sum)

t_{out}

$\mathbf{t}_{\mathrm{in}}$

How many animals are in the photo?



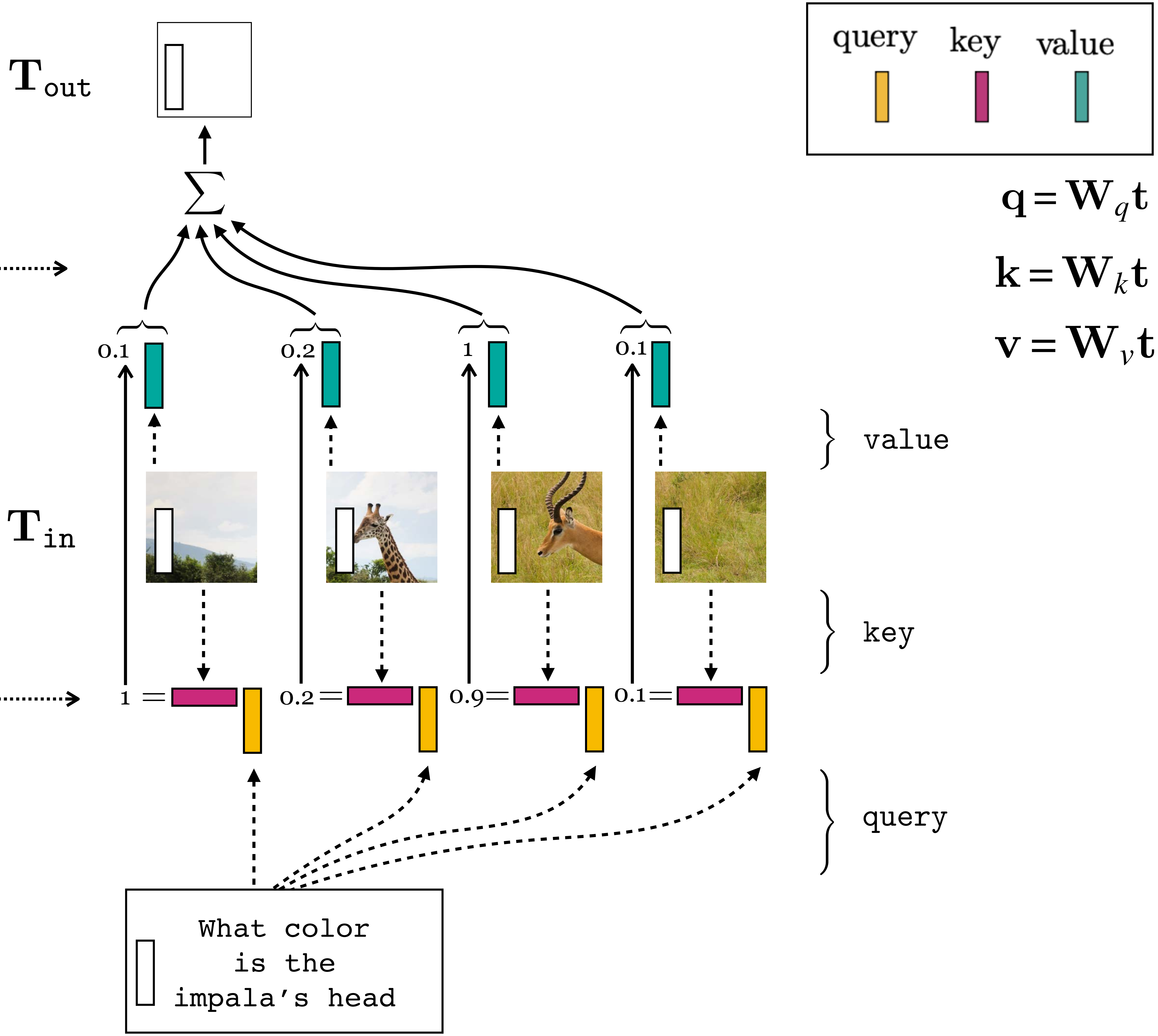
t_{out}
$\mathbf{t}_{\text{in}}$
4
1
1
1
1
How many animals are in the photo?
What is the color of the impala?

query-key-value attention

$$\mathbf{A} = \texttt{softmax}(\mathbf{s})$$

$$\mathbf{T}_{\texttt{out}} = \begin{bmatrix} a_1\mathbf{v}_1^\top \\ \vdots \\ a_N\mathbf{v}_N^\top \end{bmatrix}$$

$$\mathbf{s} = [\mathbf{q}^T_{\texttt{question}}\mathbf{k}_1, \ldots, \mathbf{q}^T_{\texttt{question}}\mathbf{k}_N]$$

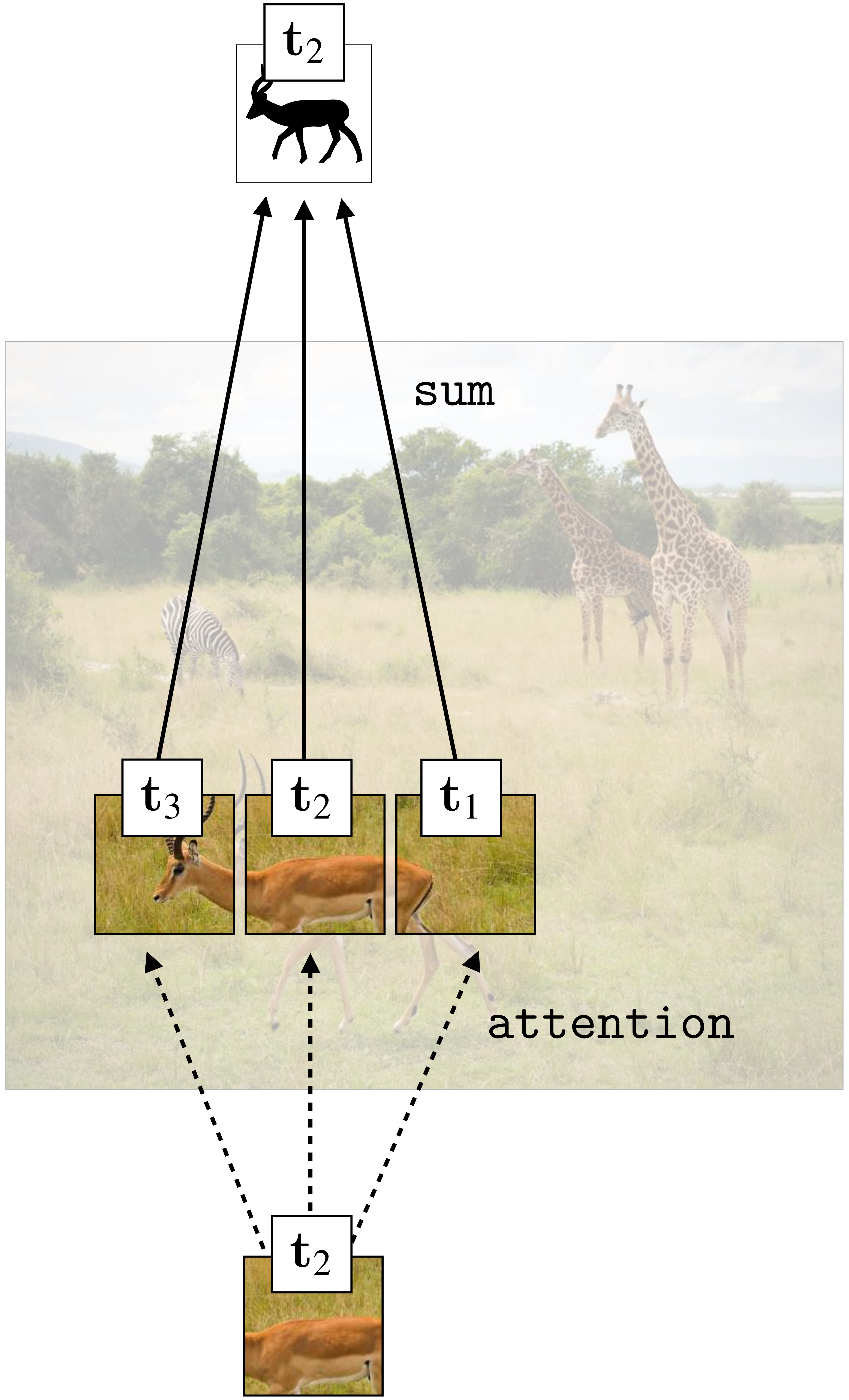


$$\mathbf{q} = \mathbf{W}_q\mathbf{t}$$

$$\mathbf{k} = \mathbf{W}_k\mathbf{t}$$

$$\mathbf{v} = \mathbf{W}_v\mathbf{t}$$

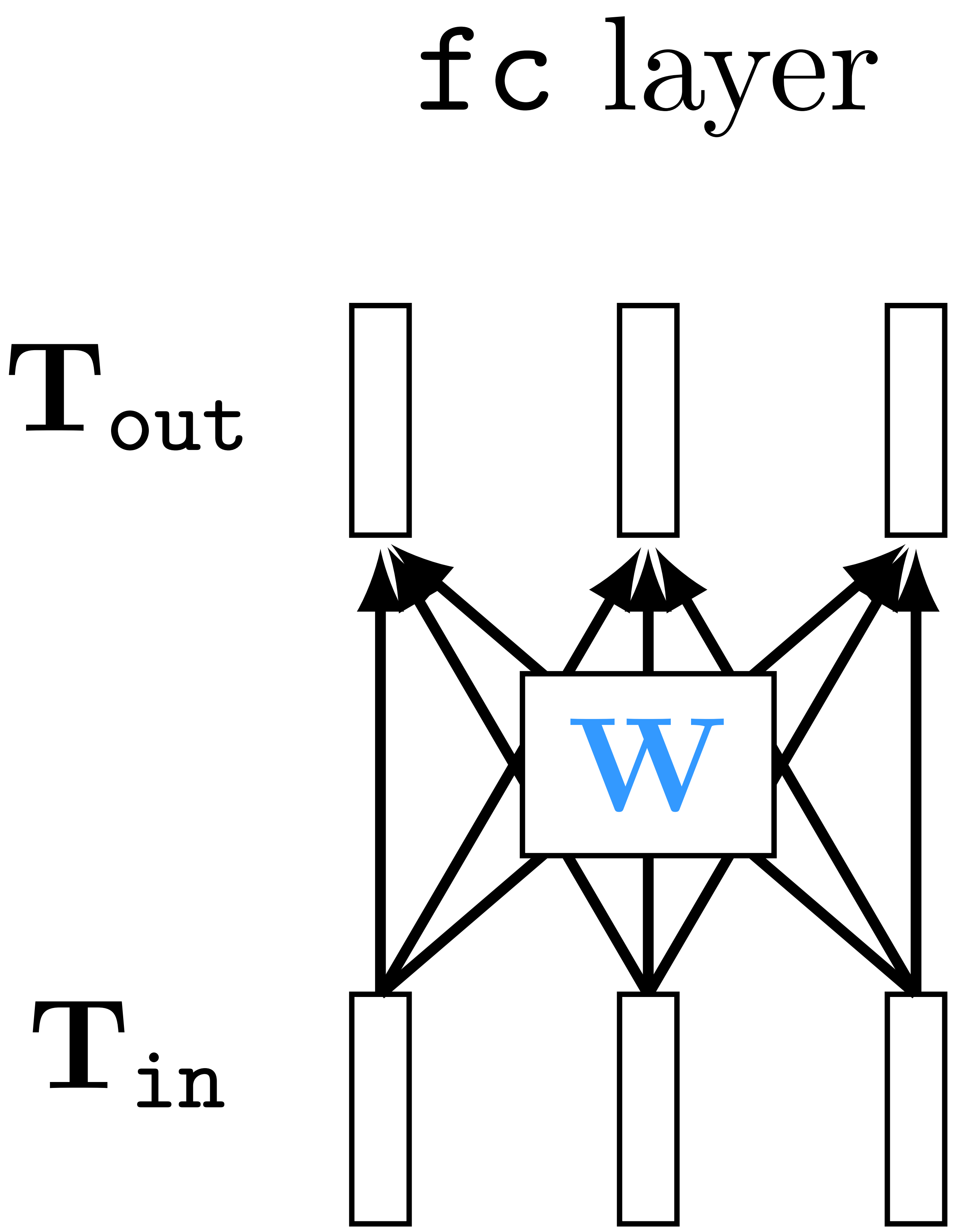
Self-attention



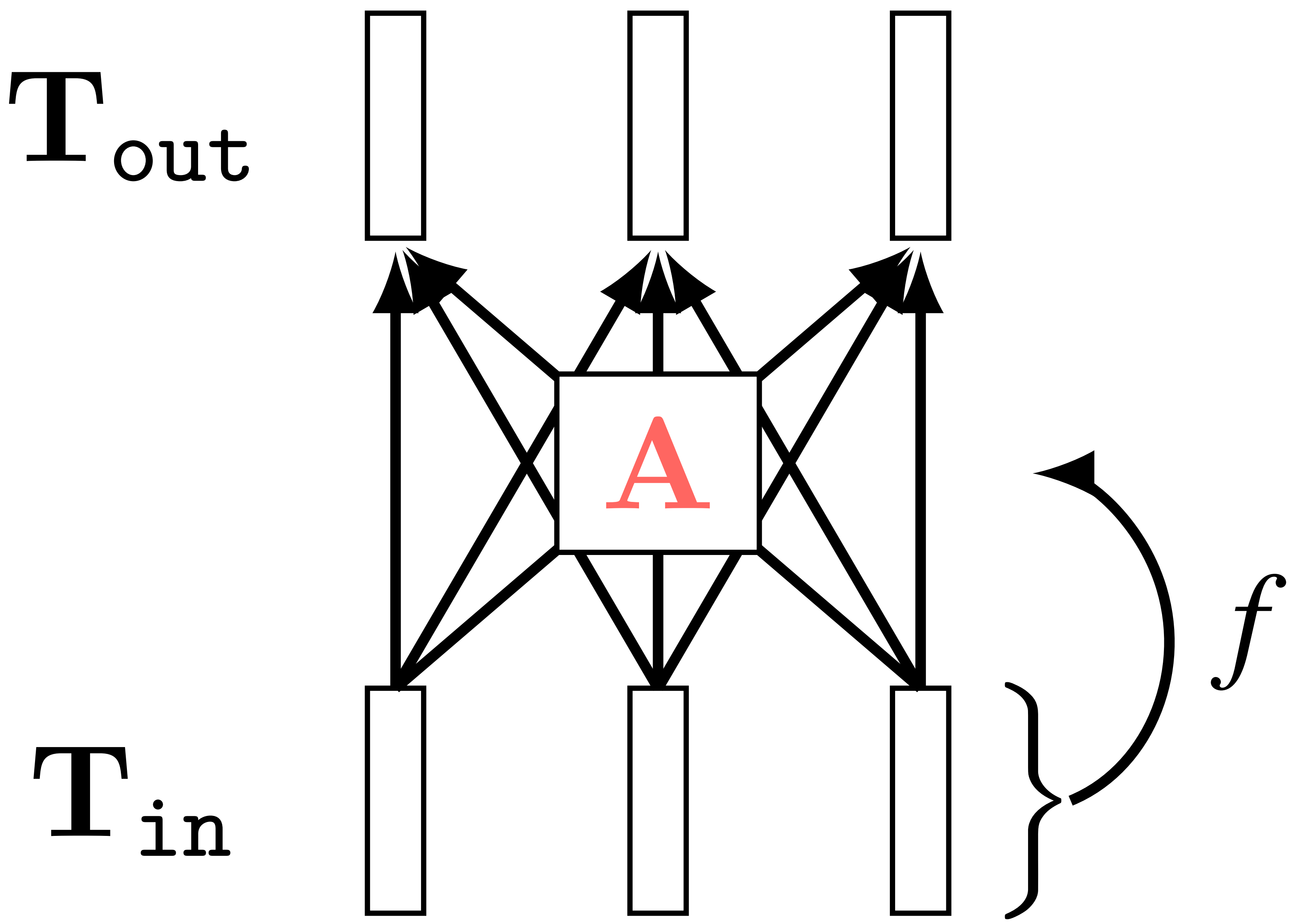
Attention maps in a trained transformer

© AI at Meta. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use/

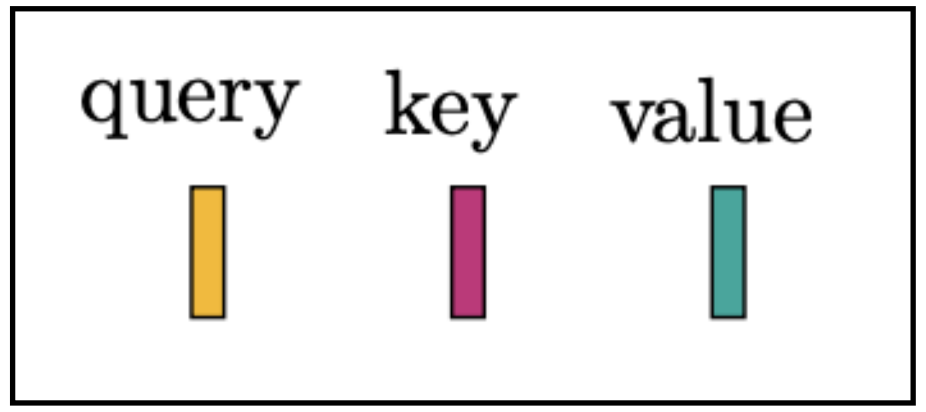
["DINO", Caron et all. 2021]

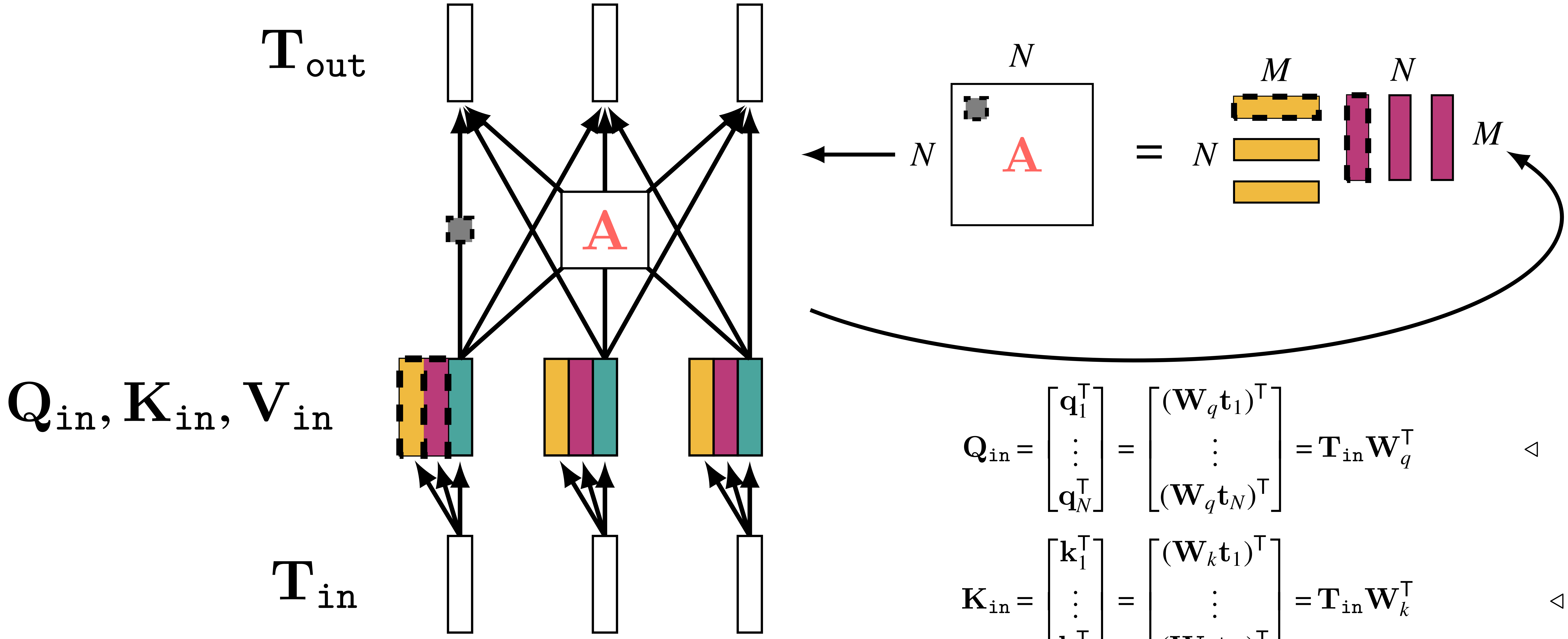

fc layer
$\mathbf{T}_{\texttt{out}}$
W
$\mathbf{T}_{\texttt{in}}$

self attn layer



$\texttt{self attn}$ layer (expanded)





$$\mathbf{Q}_{\texttt{in}} = \begin{bmatrix} \mathbf{q}_1^\top \\ \vdots \\ \mathbf{q}_N^\top \end{bmatrix} = \begin{bmatrix} (\mathbf{W}_q \mathbf{t}_1)^\top \\ \vdots \\ (\mathbf{W}_q \mathbf{t}_N)^\top \end{bmatrix} = \mathbf{T}_{\texttt{in}} \mathbf{W}_q^\top \qquad \triangleleft \quad \text{query matrix}$$

$$\mathbf{K}_{\texttt{in}} = \begin{bmatrix} \mathbf{k}_1^\top \\ \vdots \\ \mathbf{k}_N^\top \end{bmatrix} = \begin{bmatrix} (\mathbf{W}_k \mathbf{t}_1)^\top \\ \vdots \\ (\mathbf{W}_k \mathbf{t}_N)^\top \end{bmatrix} = \mathbf{T}_{\texttt{in}} \mathbf{W}_k^\top \qquad \triangleleft \quad \text{key matrix}$$

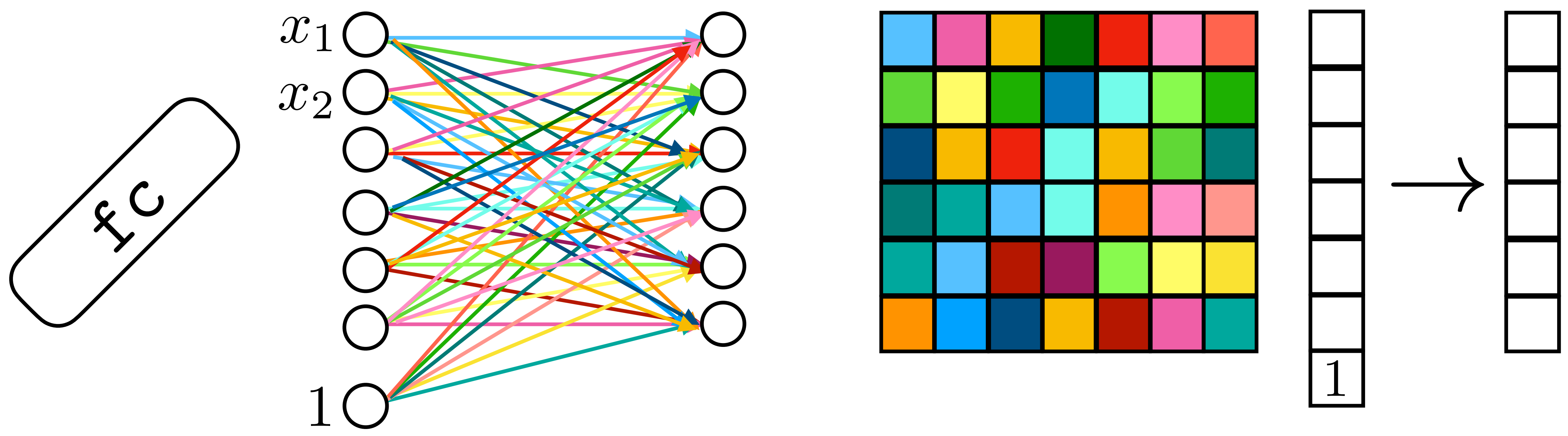
$$\mathbf{V}_{\texttt{in}} = \begin{bmatrix} \mathbf{v}_1^\top \\ \vdots \\ \mathbf{v}_N^\top \end{bmatrix} = \begin{bmatrix} (\mathbf{W}_v \mathbf{t}_1)^\top \\ \vdots \\ (\mathbf{W}_v \mathbf{t}_N)^\top \end{bmatrix} = \mathbf{T}_{\texttt{in}} \mathbf{W}_v^\top \qquad \triangleleft \quad \text{value matrix}$$

$$\mathbf{A} = f(\mathbf{T}_{\texttt{in}}) = \texttt{softmax}\left(\frac{\mathbf{Q}_{\texttt{in}} \mathbf{K}_{\texttt{in}}^\top}{\sqrt{m}}\right) \qquad \triangleleft \quad \text{attention matrix}$$

$$\mathbf{T}_{\texttt{out}} = \mathbf{A}\mathbf{V}_{\texttt{in}}$$

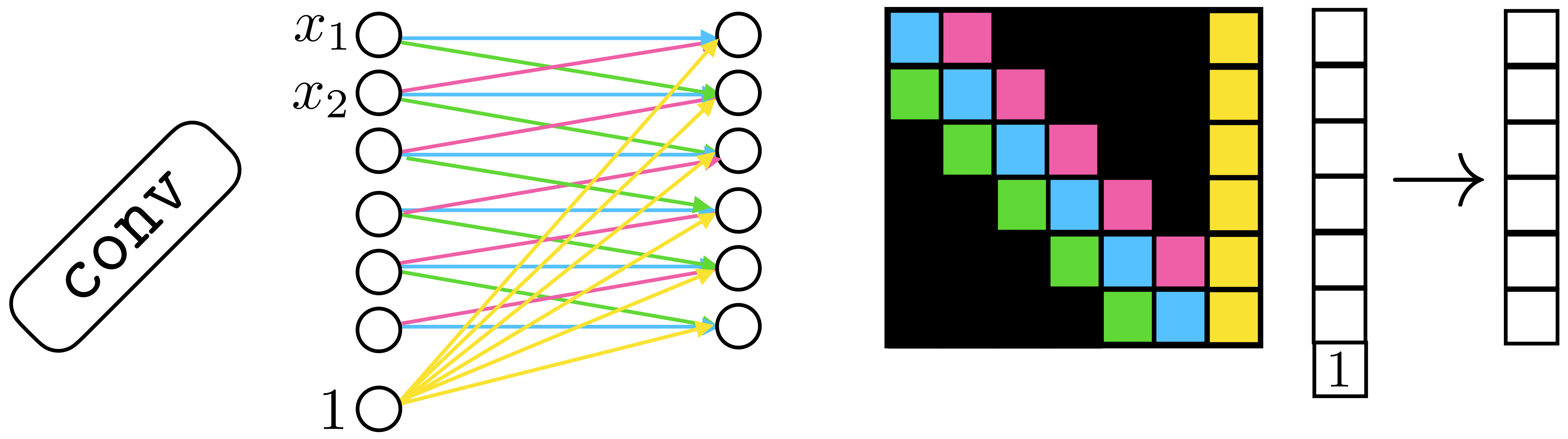
A family of linear layers

Wiring graph | Matrix | Properties



Fixed input dimensionality

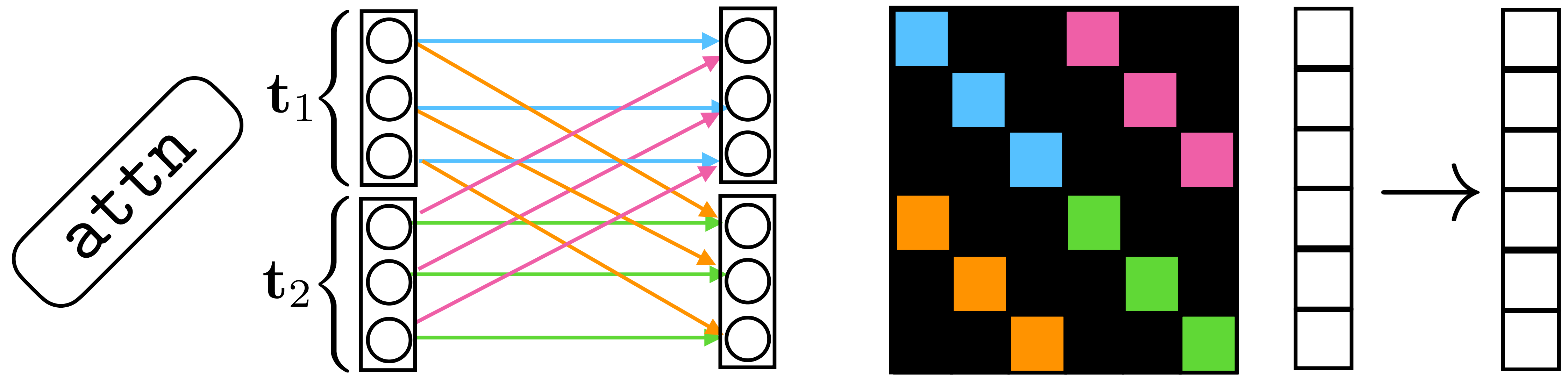
N^2 learnable parameters



Variable input dimensionality

k +1 learnable parameters (k = kernel size)

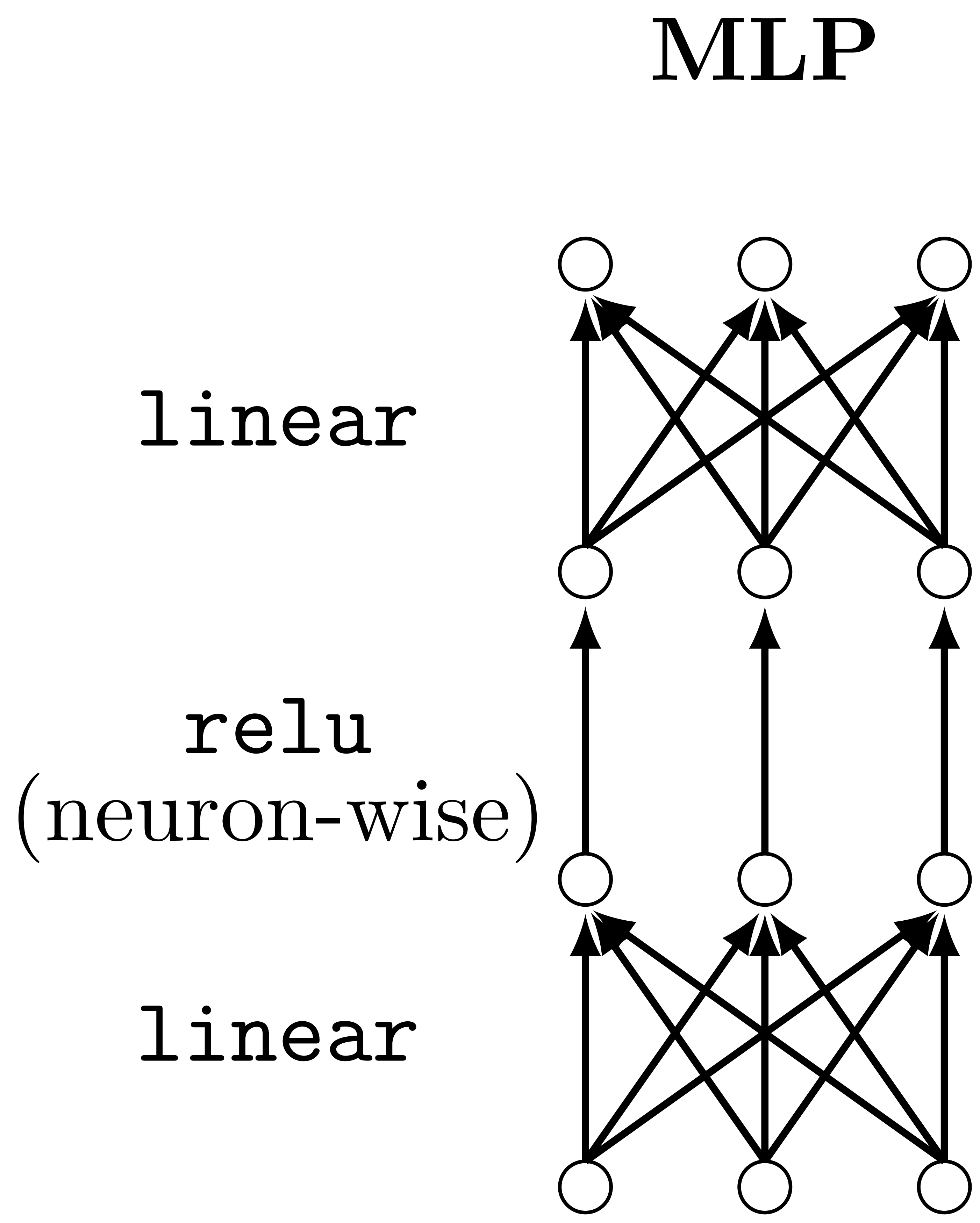
$\texttt{conv}(\texttt{translate}(\mathbf{x})) = \texttt{translate}(\texttt{conv}(\mathbf{x}))$

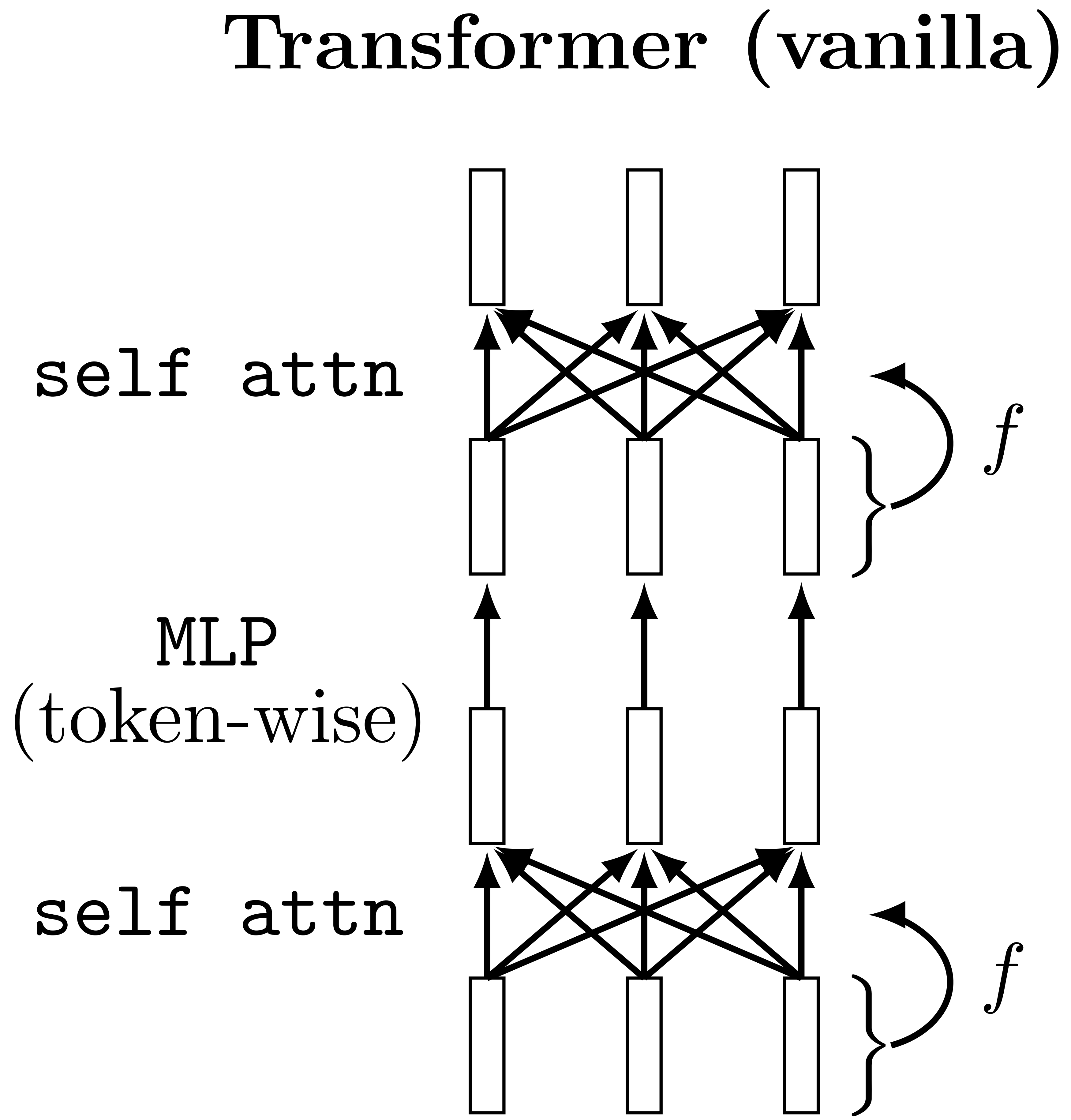


Variable input dimensionality

$|\mathbf{W}_q| + |\mathbf{W}_k| + |\mathbf{W}_v|$ learnable parameters

$\texttt{attn}(\texttt{permute}(\mathbf{T})) = \texttt{permute}(\texttt{attn}(\mathbf{T}))$


MLP
linear
relu
(neuron-wise)
linear


Transformer (vanilla)
self attn
f
MLP
(token-wise)
self attn
f

Multihead self-attention (MSA)

Rather than having just one way of attending, why not have k?

Each gets its own parameterized query(), key(), value() functions.

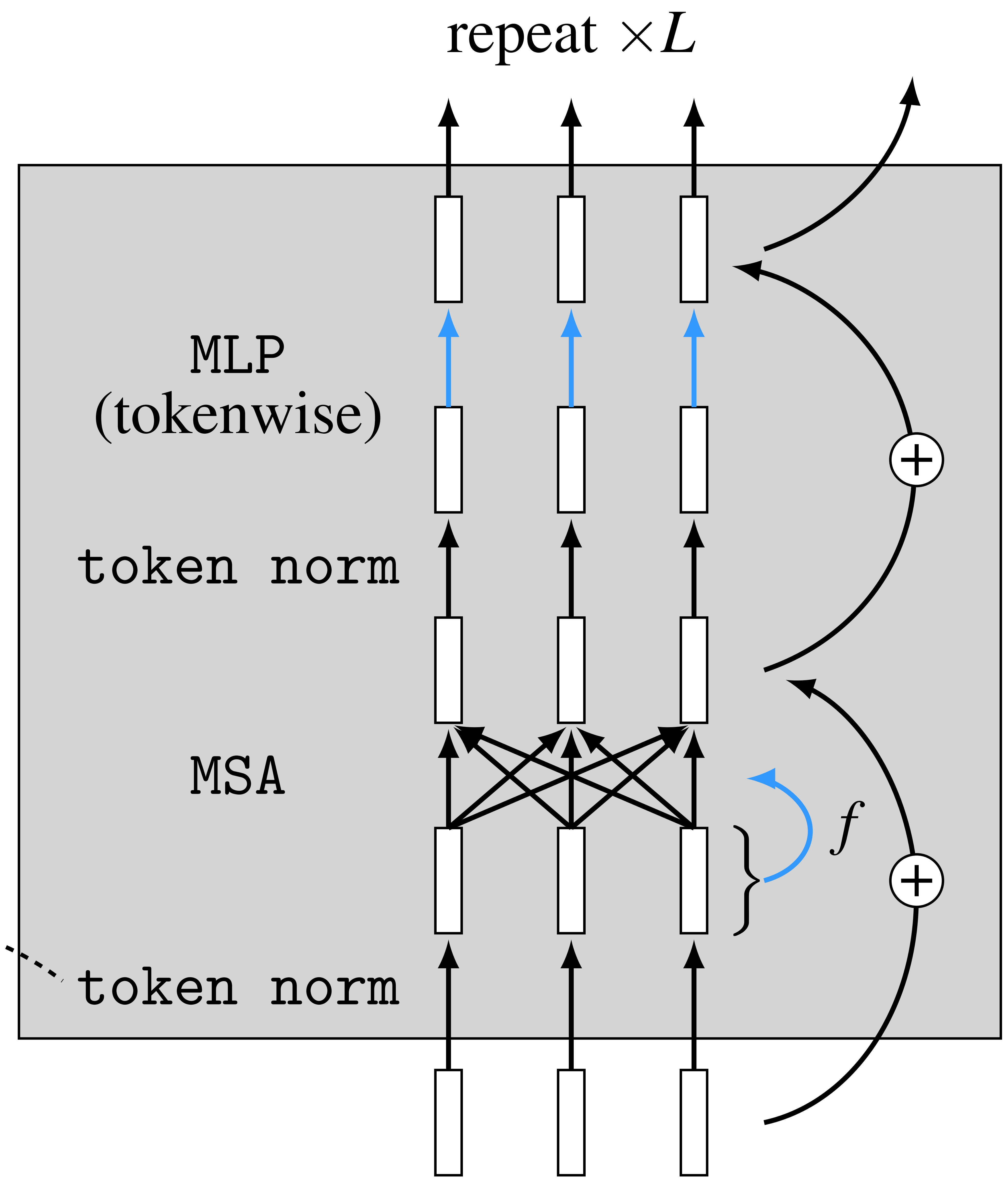
Run them all in parallel, then (weighted) sum the output token code vectors

$$\mathbf{T}^i_{\texttt{out}} = \texttt{attn}^i(\mathbf{T}_{\texttt{in}}) \quad \text{for } i \in \{1, \ldots, k\}$$

$$\bar{\mathbf{T}}_{\texttt{out}} = \begin{bmatrix} \mathbf{T}^1_{\texttt{out}}[0,:] & \ldots & \mathbf{T}^k_{\texttt{out}}[0,:] \\ \vdots & \vdots & \vdots \\ \mathbf{T}^1_{\texttt{out}}[N-1,:] & \ldots & \mathbf{T}^k_{\texttt{out}}[N-1,:] \end{bmatrix} \qquad \triangleleft \quad \bar{\mathbf{T}}_{\texttt{out}} \in \mathbb{R}^{N \times kv}$$

$$\mathbf{T}_{\texttt{out}} = \bar{\mathbf{T}}_{\texttt{out}} \mathbf{W}_{\texttt{MSA}} \qquad \triangleleft \quad \mathbf{W}_{\texttt{MSA}} \in \mathbb{R}^{kv \times d}$$

Transformer (ViT)



==

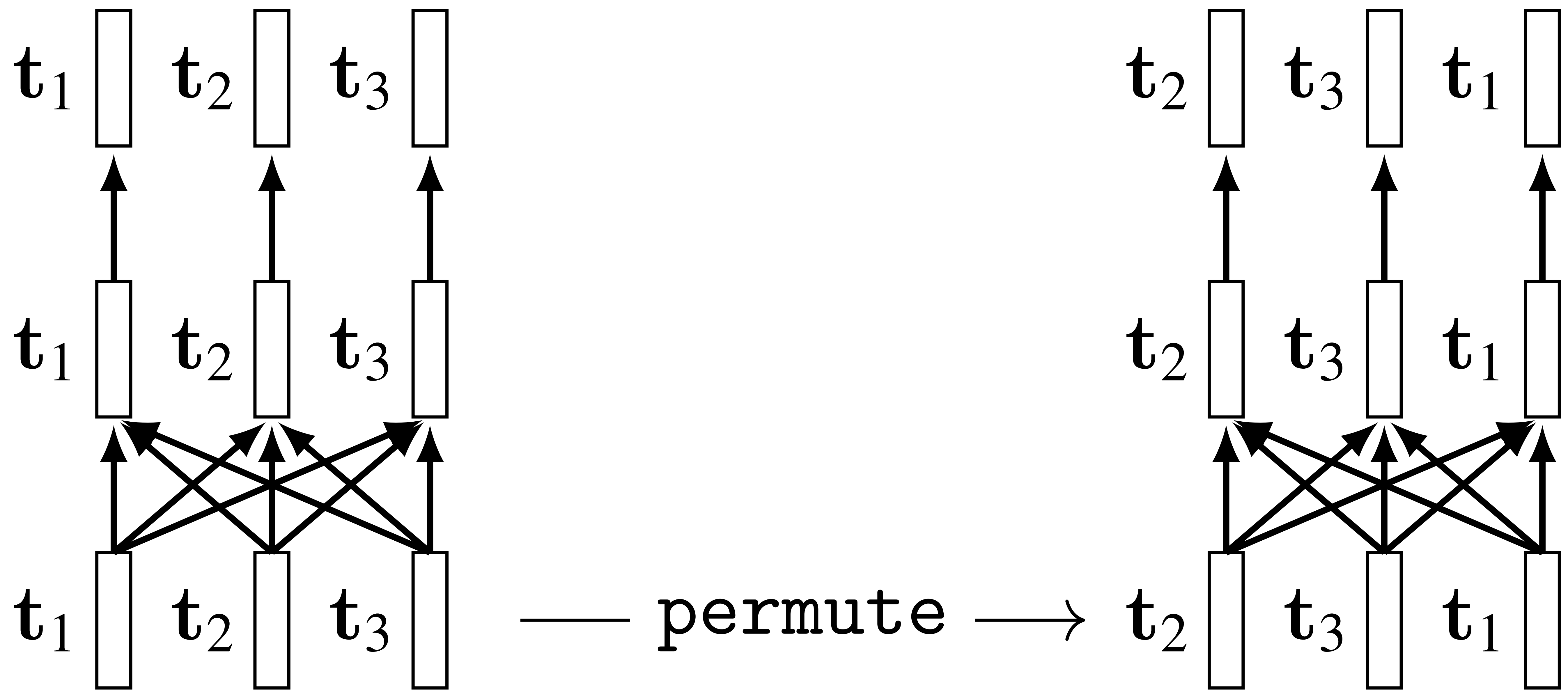
layernorm

$$x_{\mathtt{out}}[k] = \frac{x_{\mathtt{in}}[k] - \mathbb{E}[x_{\mathtt{in}}[k]]}{\sqrt{\mathtt{Var}[x_{\mathtt{in}}[k]]}}$$

```python
# x : input data (RGB image)
# K : tokenization patch size
# d : token/query/key/value dimensionality (setting these all as the same)
# L : number of layers
# W_q_T, W_k_T, W_v_T : transposed query/key/value projection matrices
# mlp: tokenwise mlps

# tokenize input image
T = tokenize(x,K) # 3 x H x W image --> N x d array of token code vectors

# run tokens through all L layers
for l in range(L):

    # attention layer
    Q, K, V = nn.matmul(nn.layernorm(T),[W_q_T[l], W_k_T[l], W_v_T[l]])
    # nn.matmul does matrix multiplication
    A = nn.softmax(nn.matmul(Q,K.transpose()), dim=0)/sqrt(d)
    T = nn.matmul(A,V) + T # note residual connection

    # tokenwise mlp
    T = mlp[l](nn.layernorm(T)) + T # note residual connection

# T now contains the output token representation computed by the transformer
```

New idea #3: positional encoding

Permutation equivariance



$$F_\theta(\texttt{permute}(\mathbf{T}_{\texttt{in}})) = \texttt{permute}(F_\theta(\mathbf{T}_{\texttt{in}}))$$

$$\texttt{attn}(\texttt{permute}(\mathbf{T}_{\texttt{in}})) = \texttt{permute}(\texttt{attn}(\mathbf{T}_{\texttt{in}}))$$

$$\downarrow$$

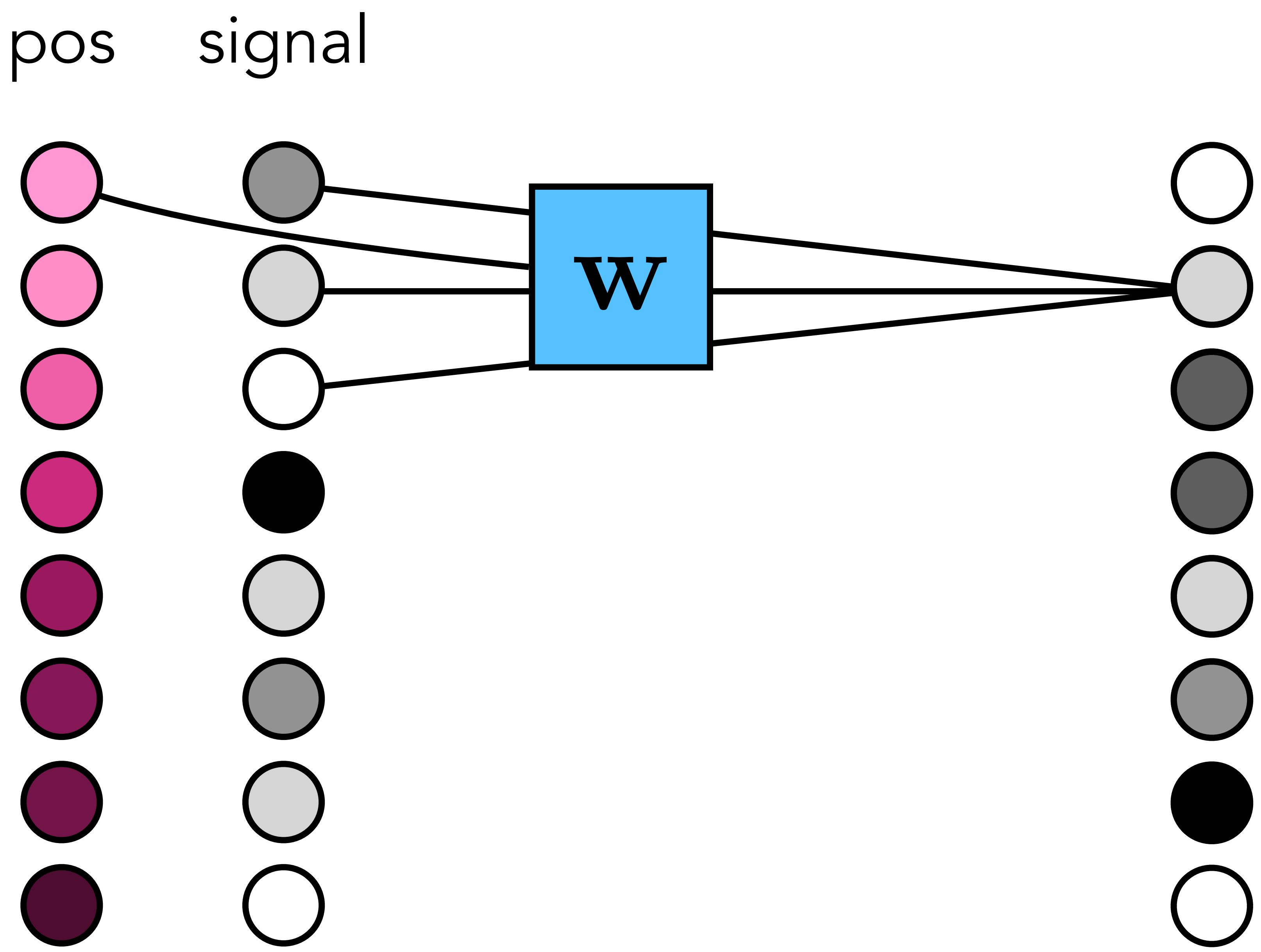
$$\texttt{transformer}(\texttt{permute}(\mathbf{T}_{\texttt{in}})) = \texttt{permute}(\texttt{transformer}(\mathbf{T}_{\texttt{in}}))$$

Set2Set

What if you *don't* want to be shift invariant?

1. Use an architecture that is not shift invariant (e.g., MLP)

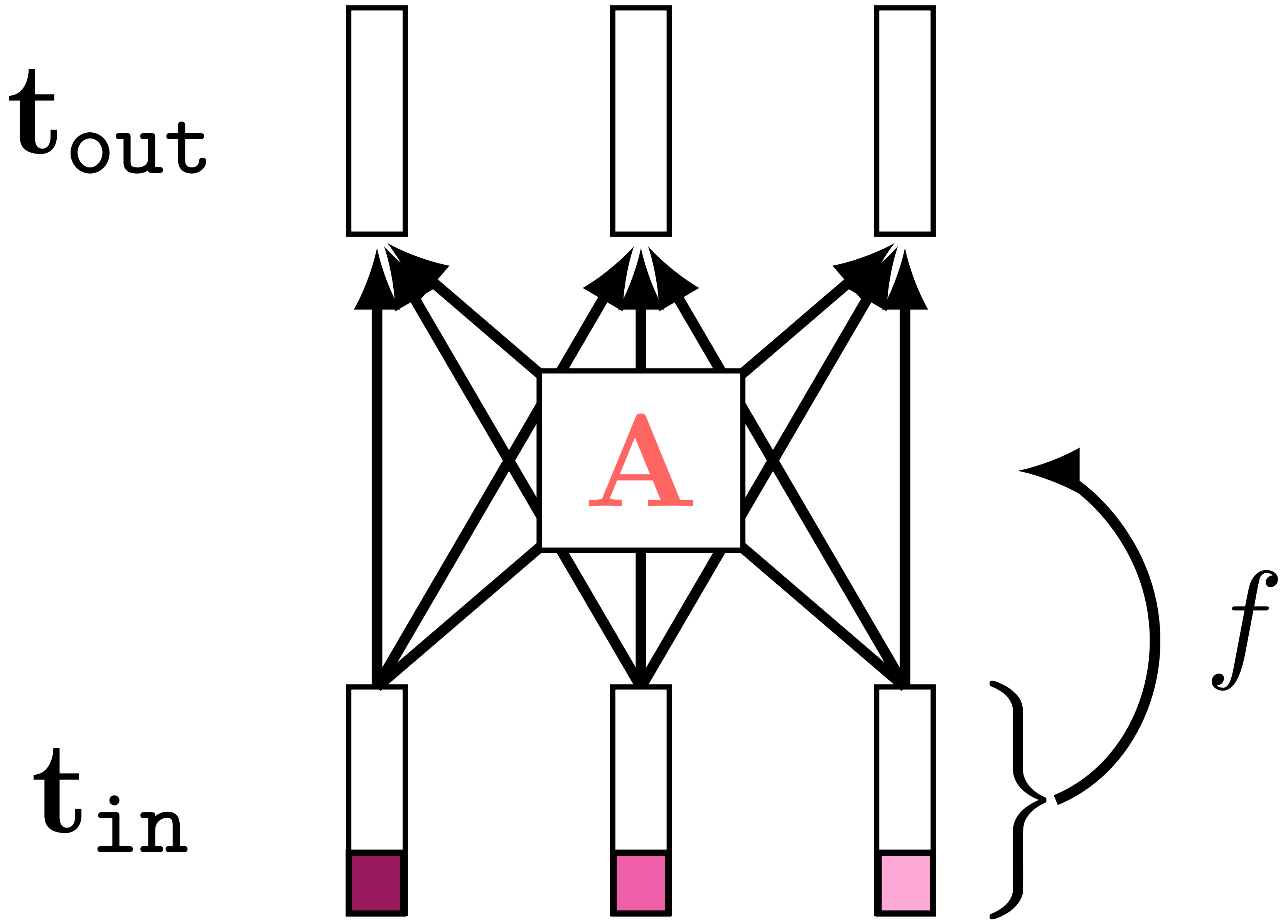
2. Add location information to the *input* to the convolutional filters — this is called **positional encoding**



What if you *don't* want to be permutation invariant?

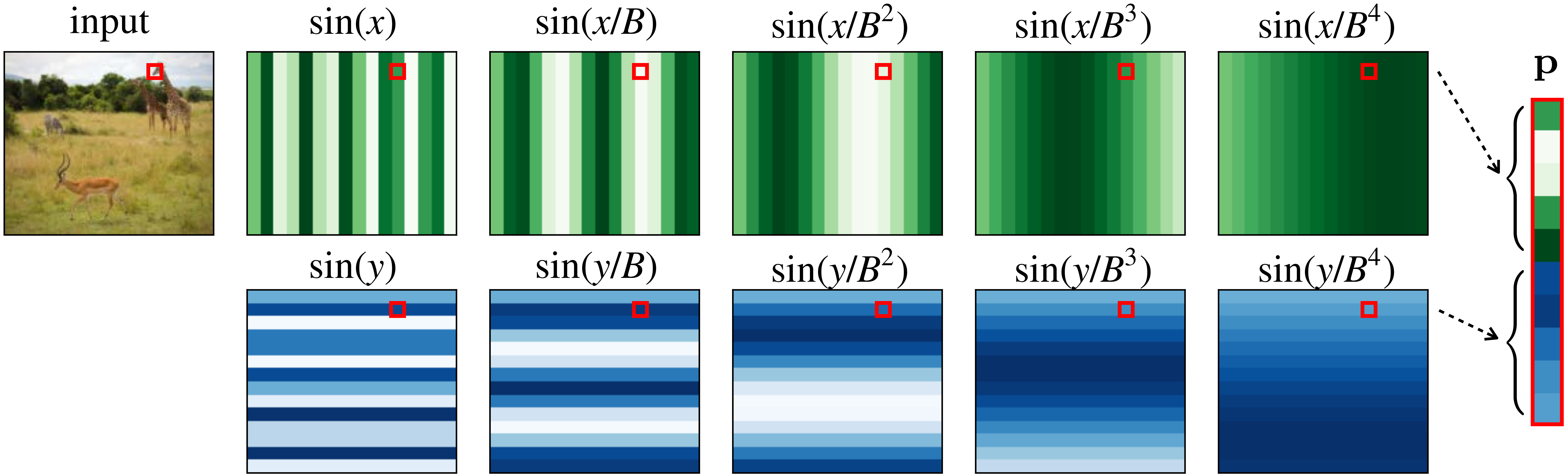
1. Use an architecture that is not permutation invariant (e.g., MLP)

2. Add location information to the token code vectors — this is called **positional encoding**



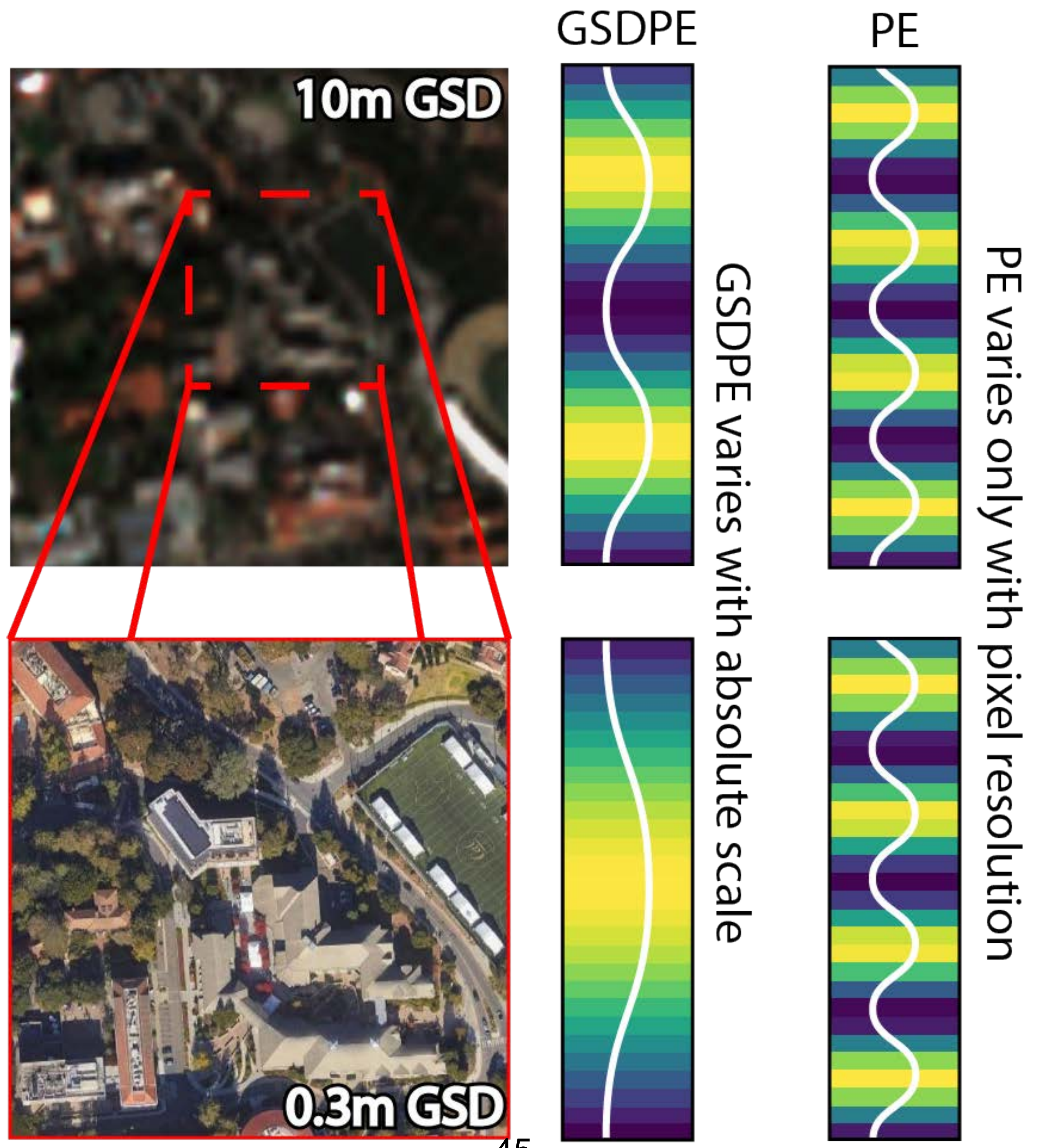
Fourier positional codes

Represent coordinates on Fourier basis



Other positional encodings

ScaleMAE uses ground sample distance positional encoding to train an MAE across spatial scales of remote sensing data

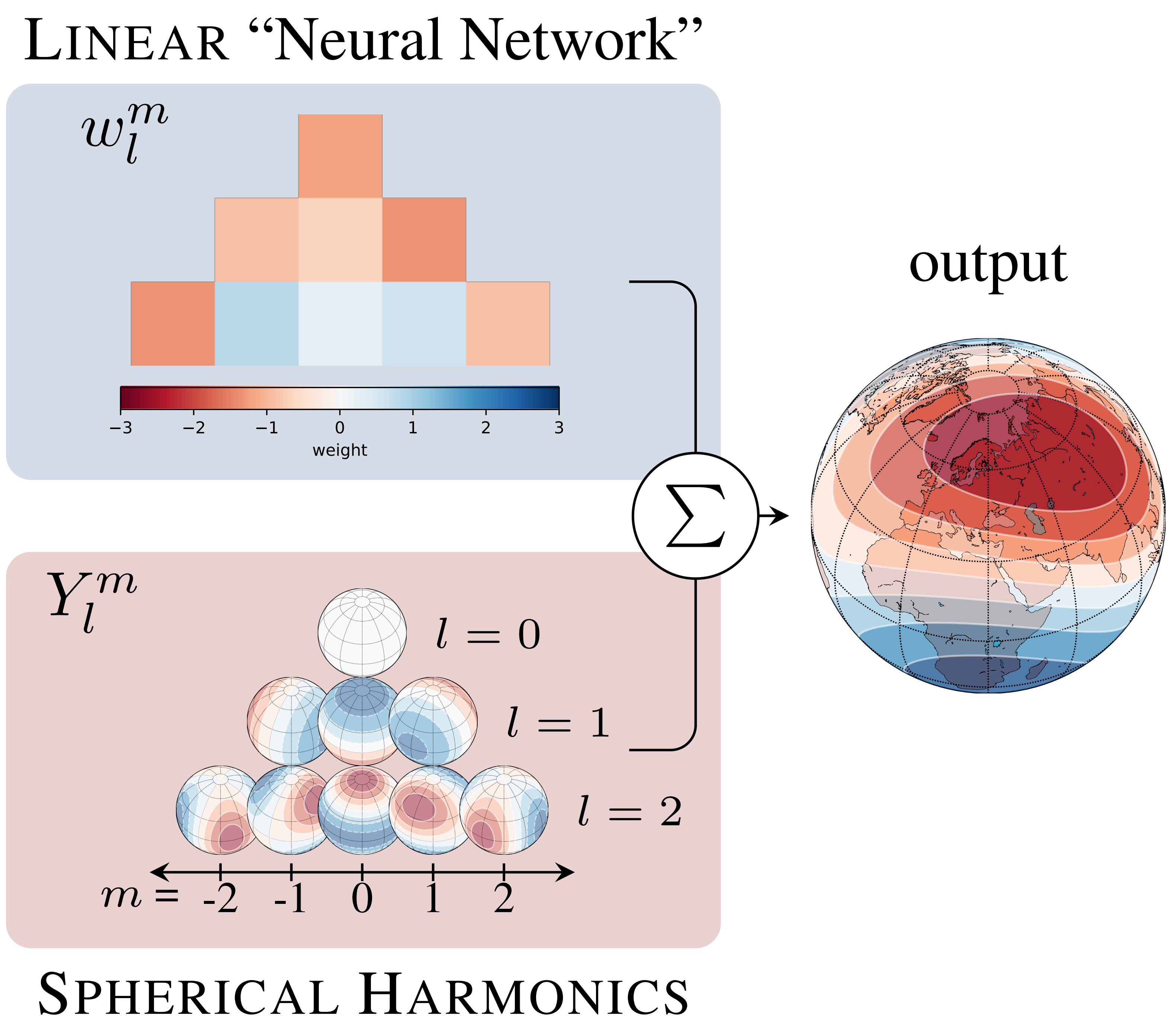


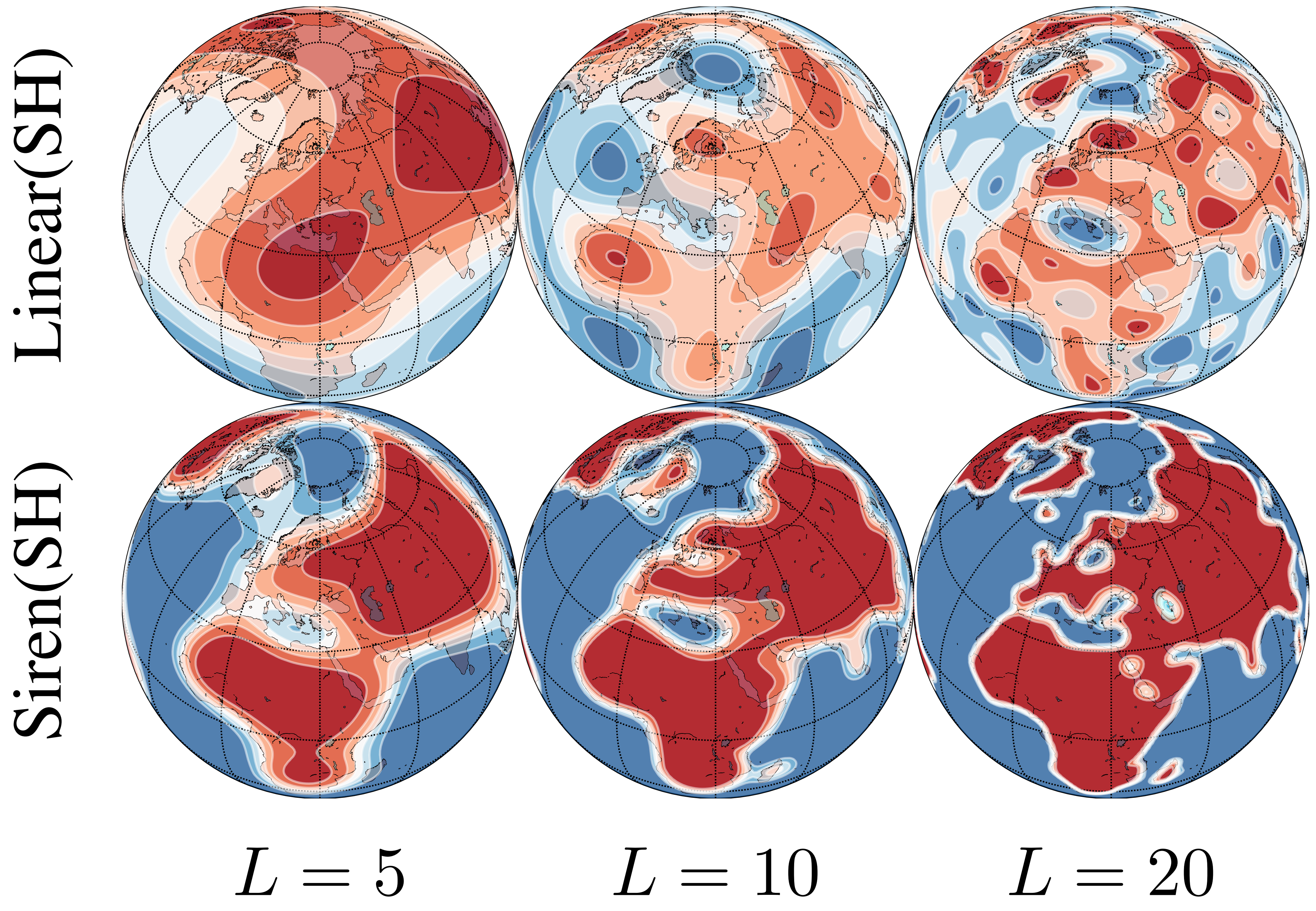
© Reed, et al. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use/

https://arxiv.org/abs/2212.14532

Other positional encodings

Geographic location encoding with spherical harmonics and sinusoidal representation networks



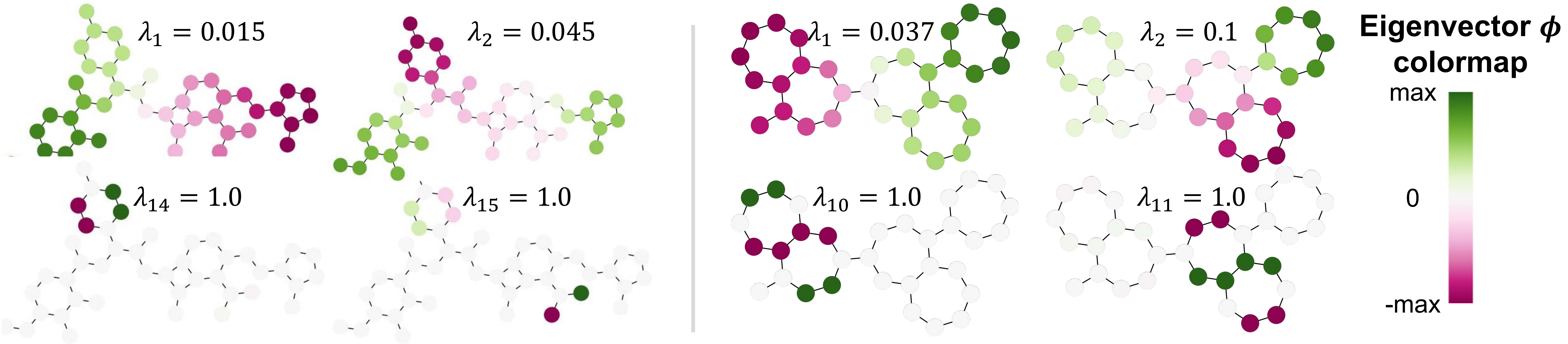


Courtesy of Rußwurm, et al. Used under CC BY.

https://arxiv.org/abs/2310.06743

Other positional encodings

Laplacian positional encodings to encode node positions in a graph



Courtesy of Kreuzer, et al. Used under CC BY-NC-SA.

https://arxiv.org/abs/2106.03893

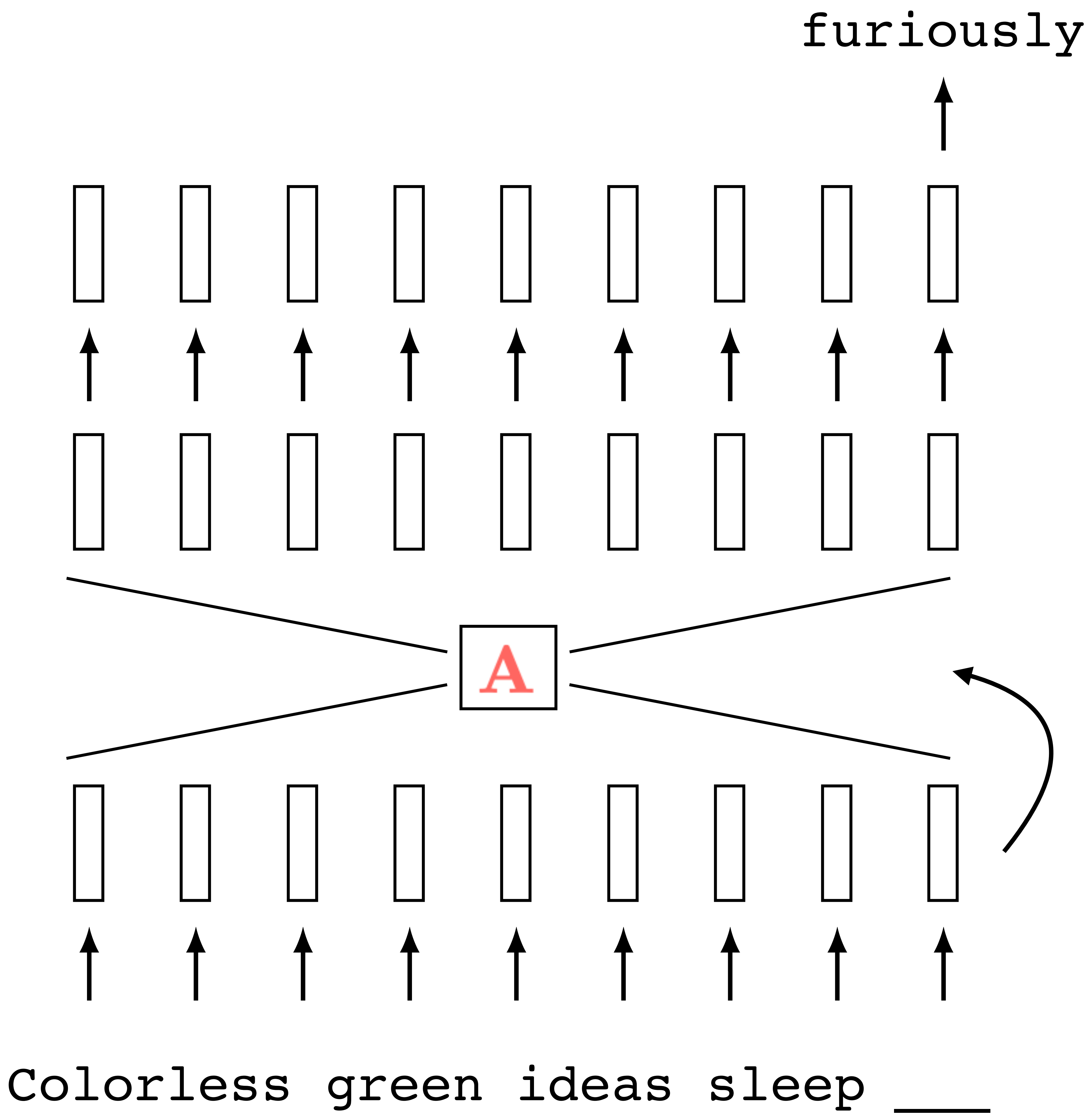
Autoregressive models

Once upon ___ → Predictor → time a

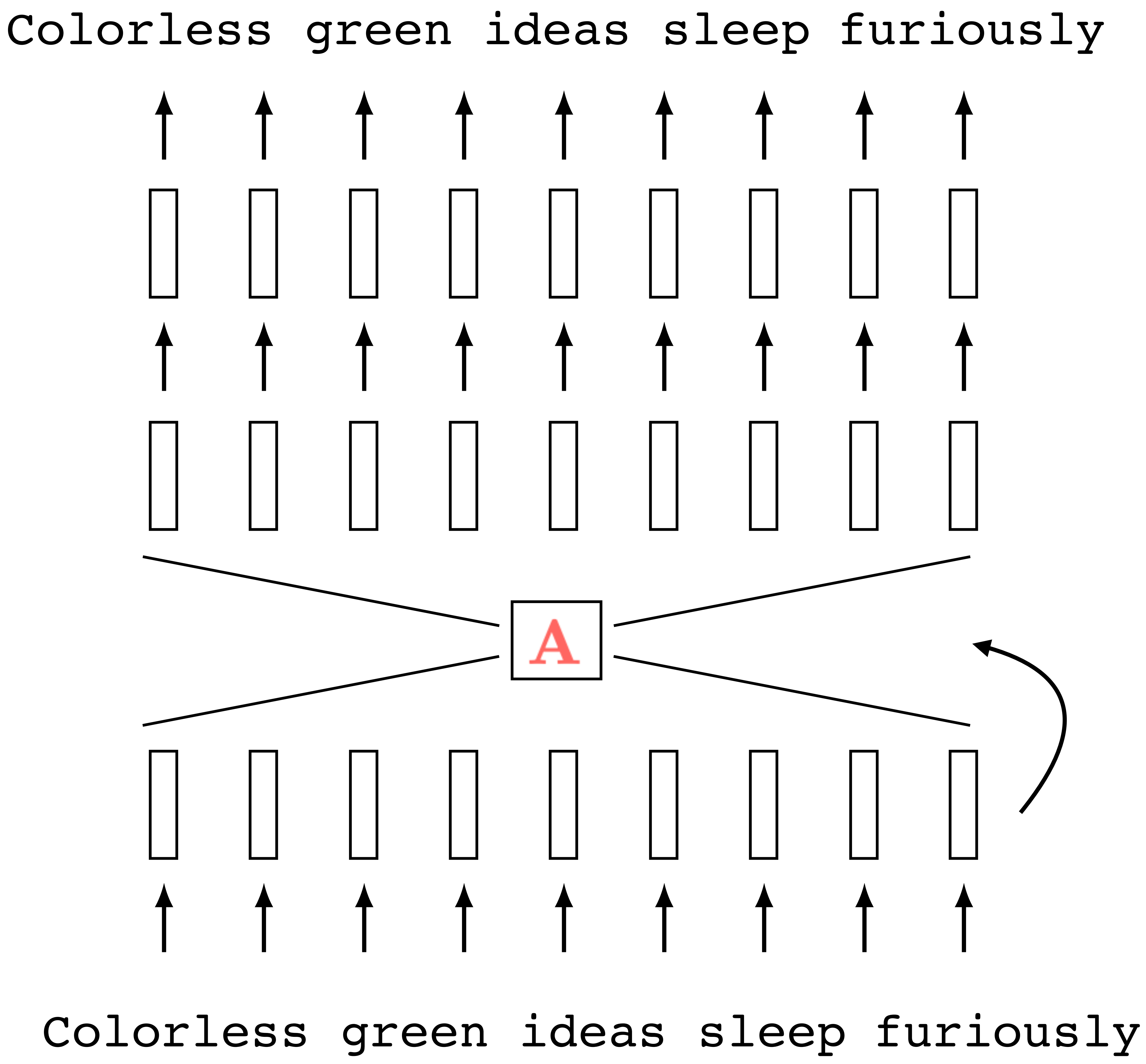
Once ___ a time → Predictor → Upon

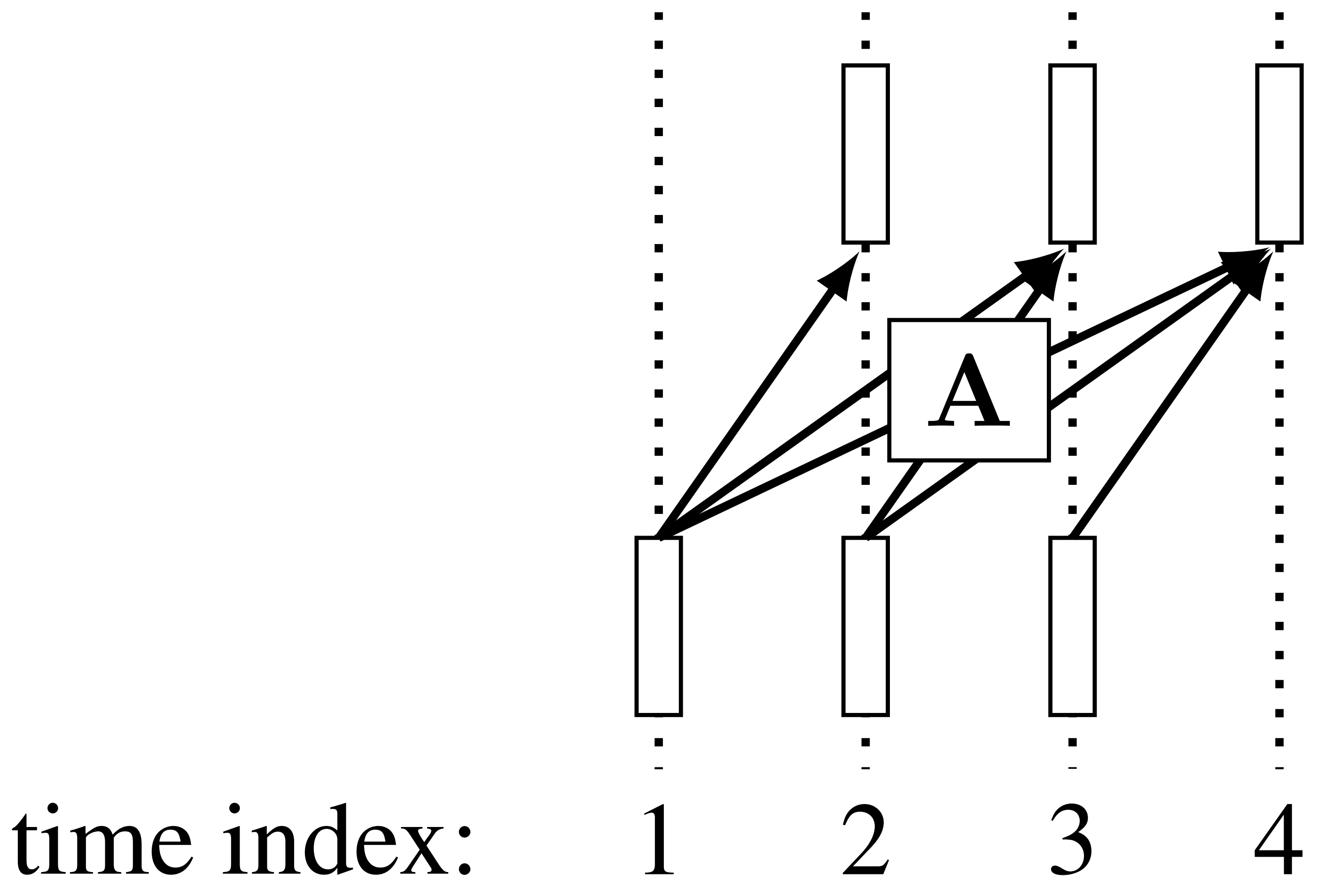

Training
$\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}$
$\mathbf{x}_n$
Once upon a , time
There and back , again
The slow brown , fox
To be or not to, be
Learner
Predictor
Sampling
$\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}$
Colorless green ideas sleep
Predictor
$\hat{\mathbf{x}}_n$
furiously

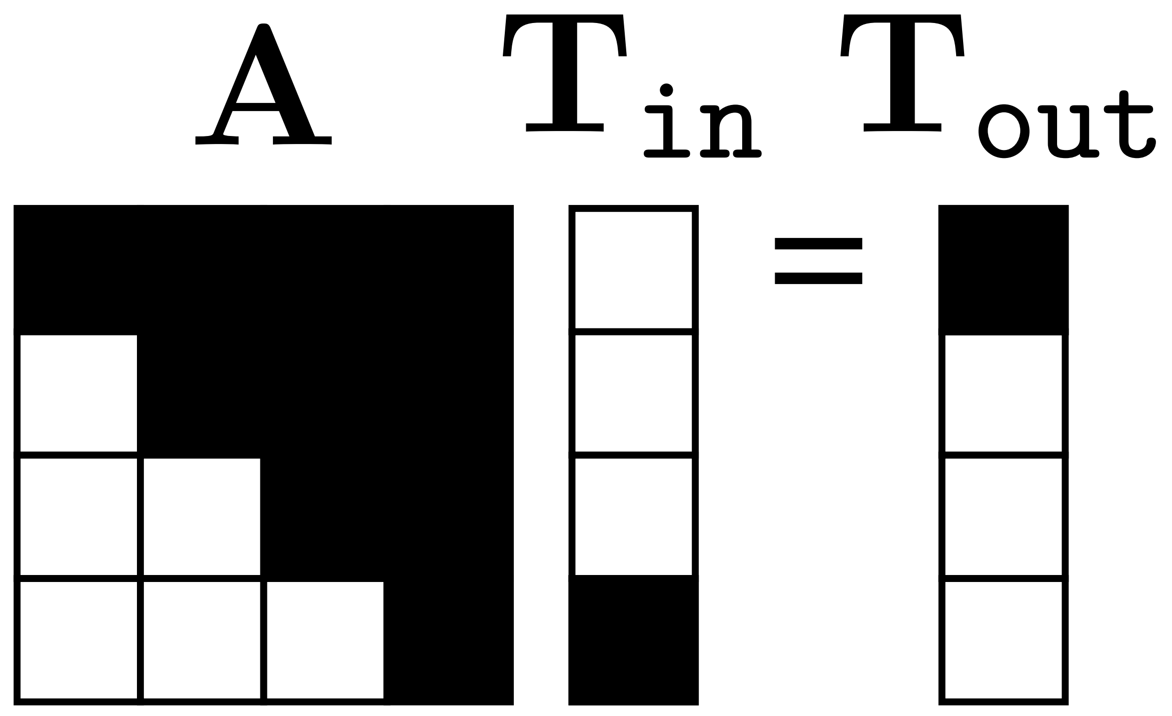
GPT (and many other related models)

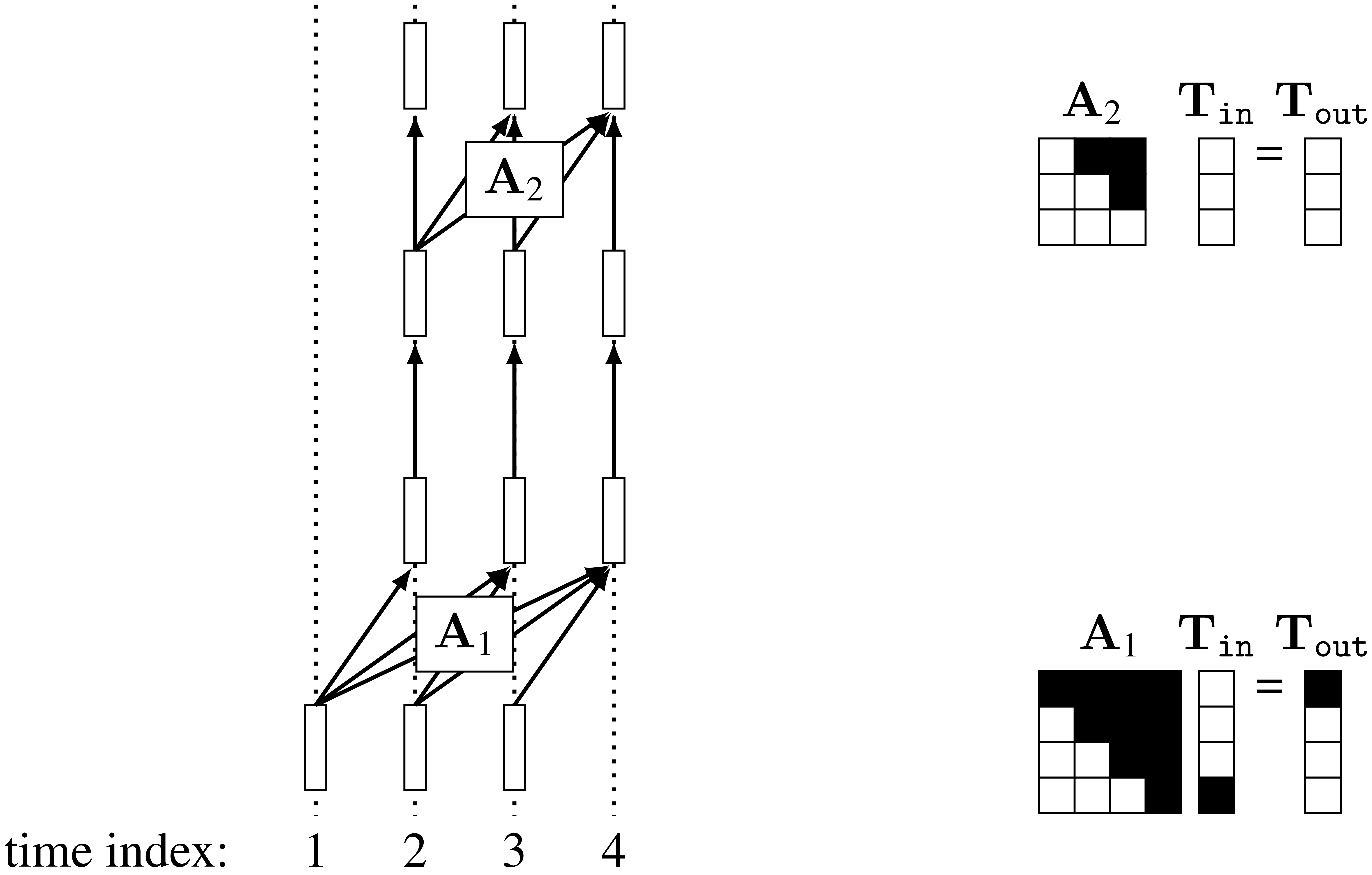


GPT training (and many other related models)








A2
A1
time index: 1 2 3 4
A2 Tin Tout
=
A1 Tin Tout
=

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

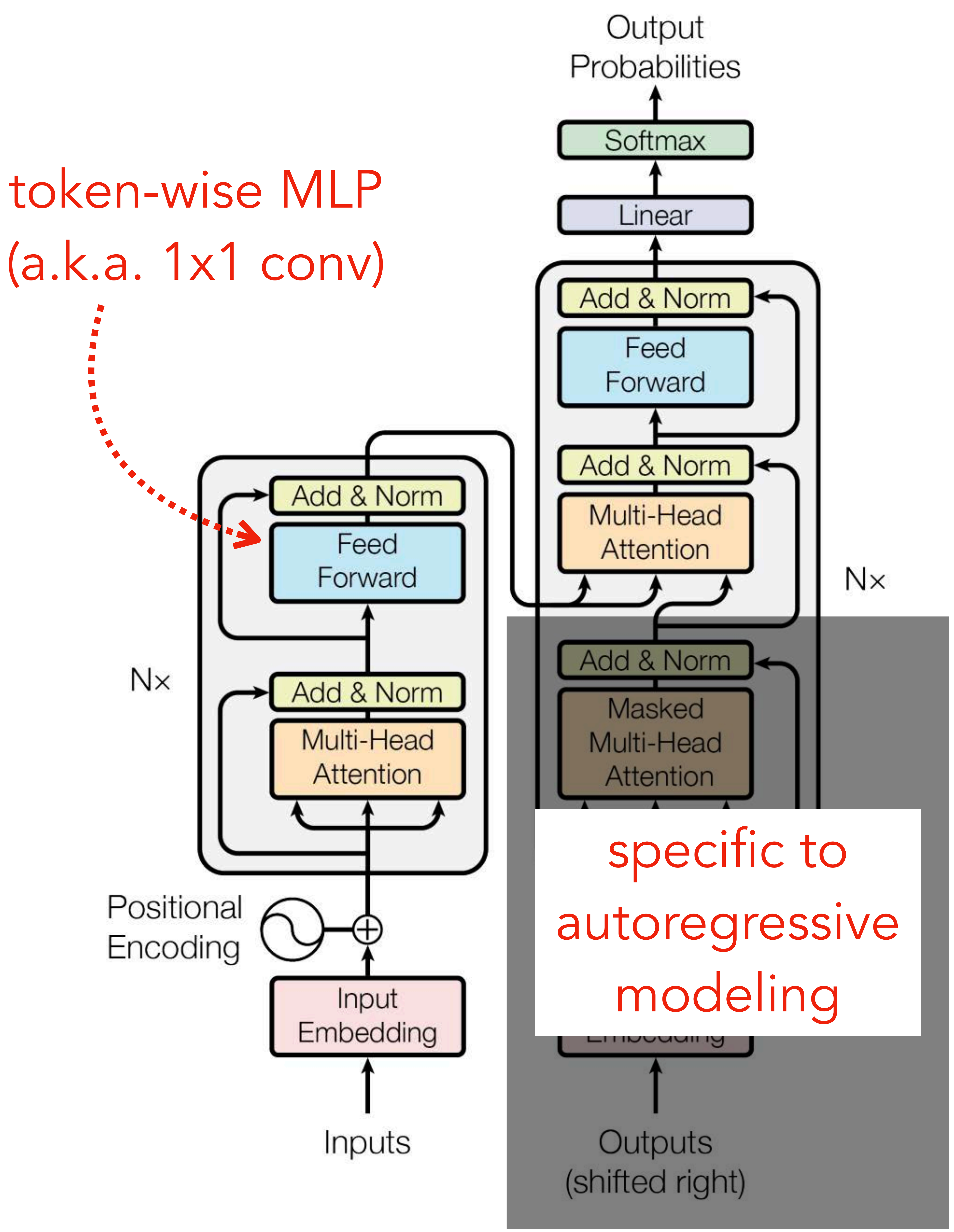
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

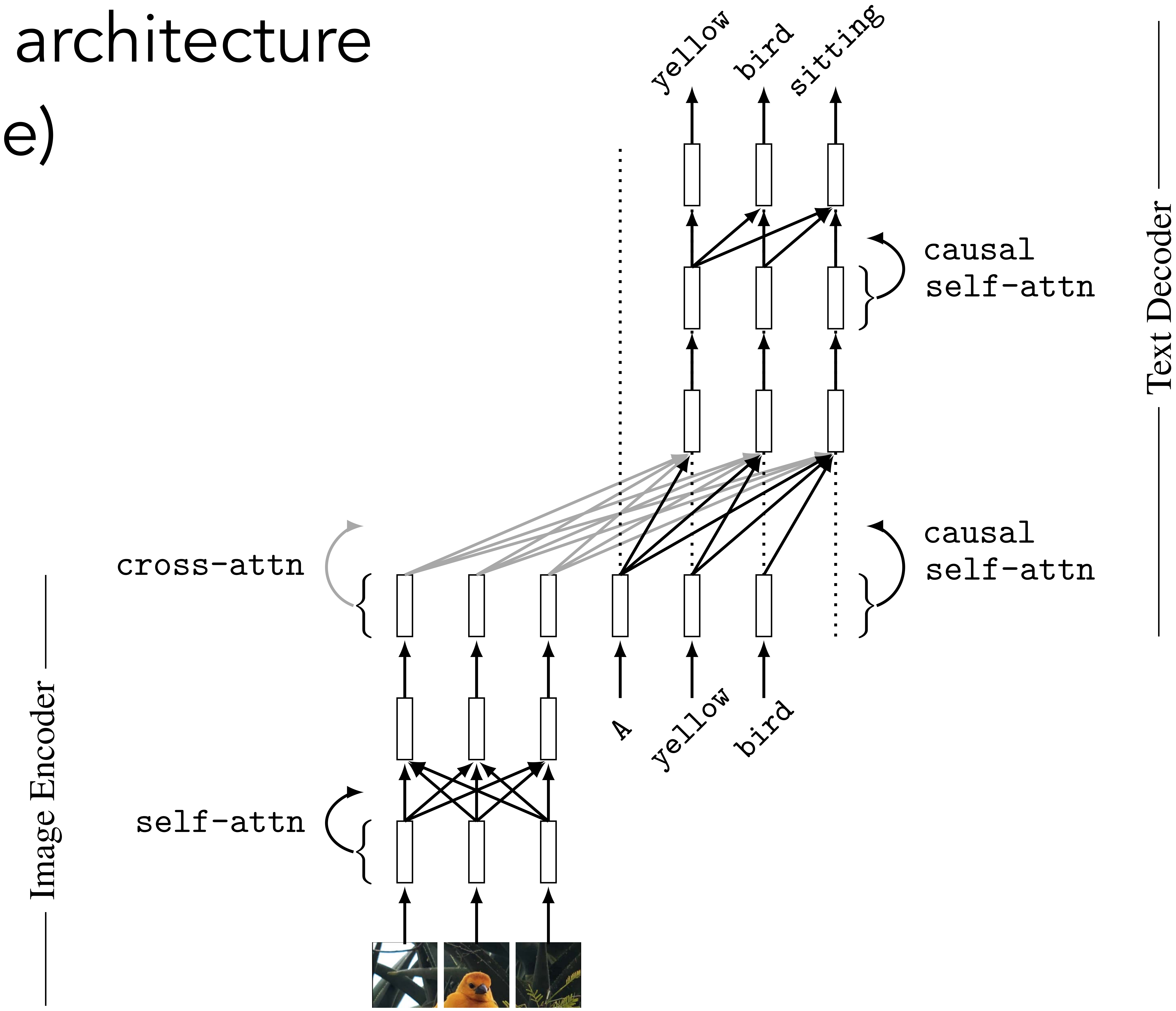
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.



© Vaswani, et al. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use/

Image-to-text architecture (autoregressive)



MIT OpenCourseWare
https://ocw.mit.edu

6.7960 Deep Learning
Fall 2024

For information about citing these materials or our Terms of Use, visit: https://ocw.mit.edu/terms