

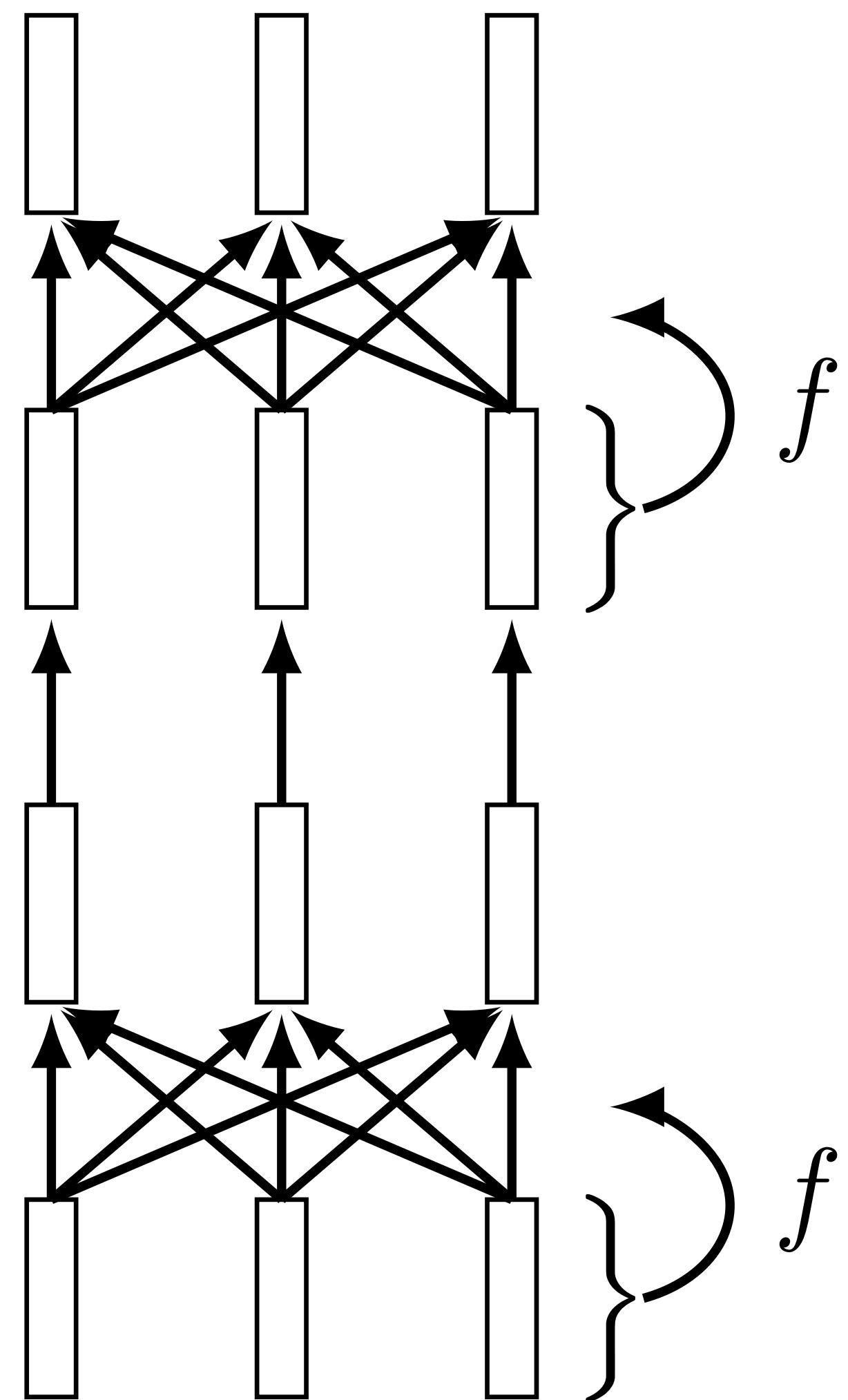
# Lecture 7: Transformers

Speaker: Sara Beery

self attn

MLP  
(token-wise)

self attn

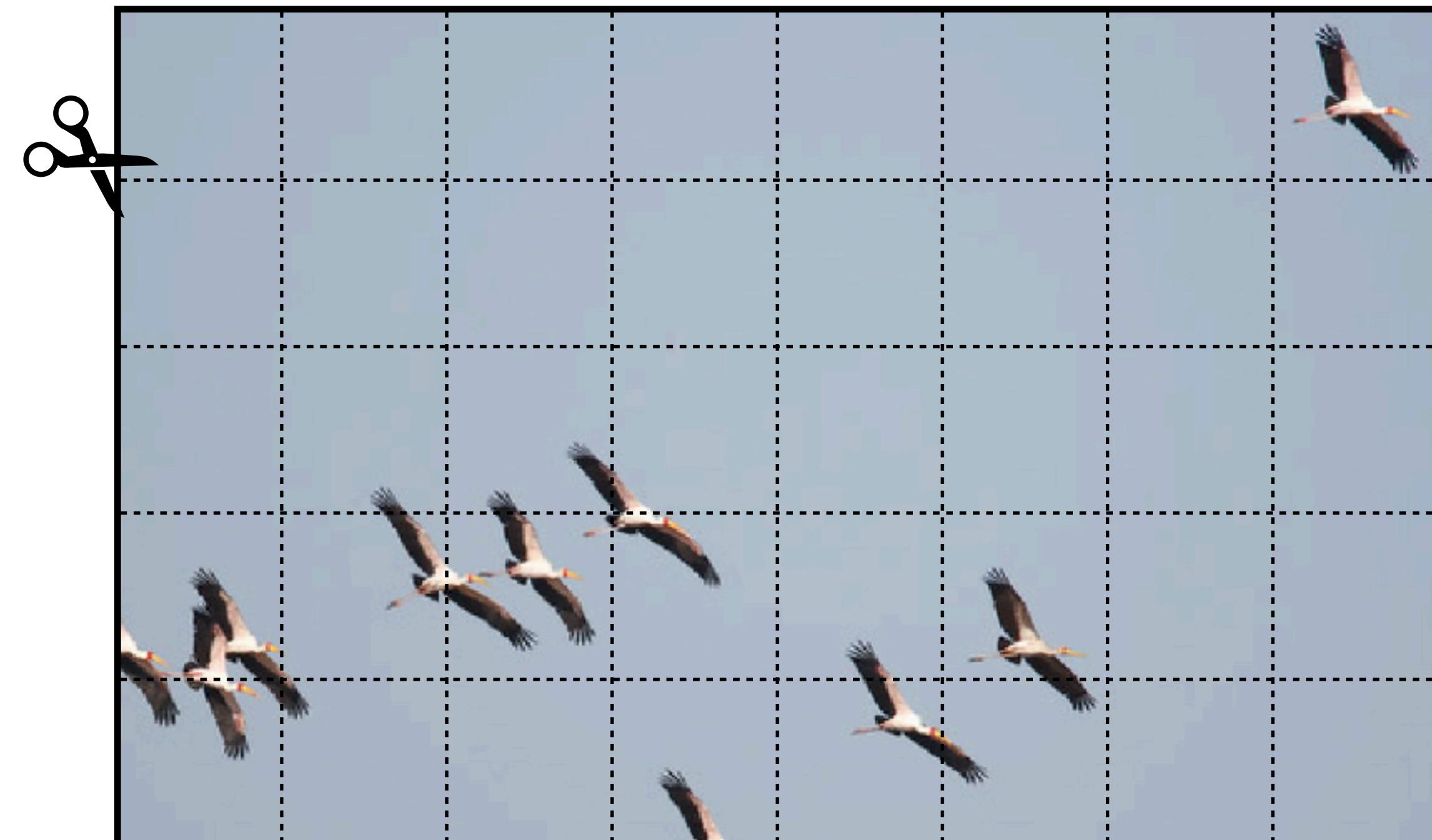


# 9. Transformers

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



# A Limitation of CNNs

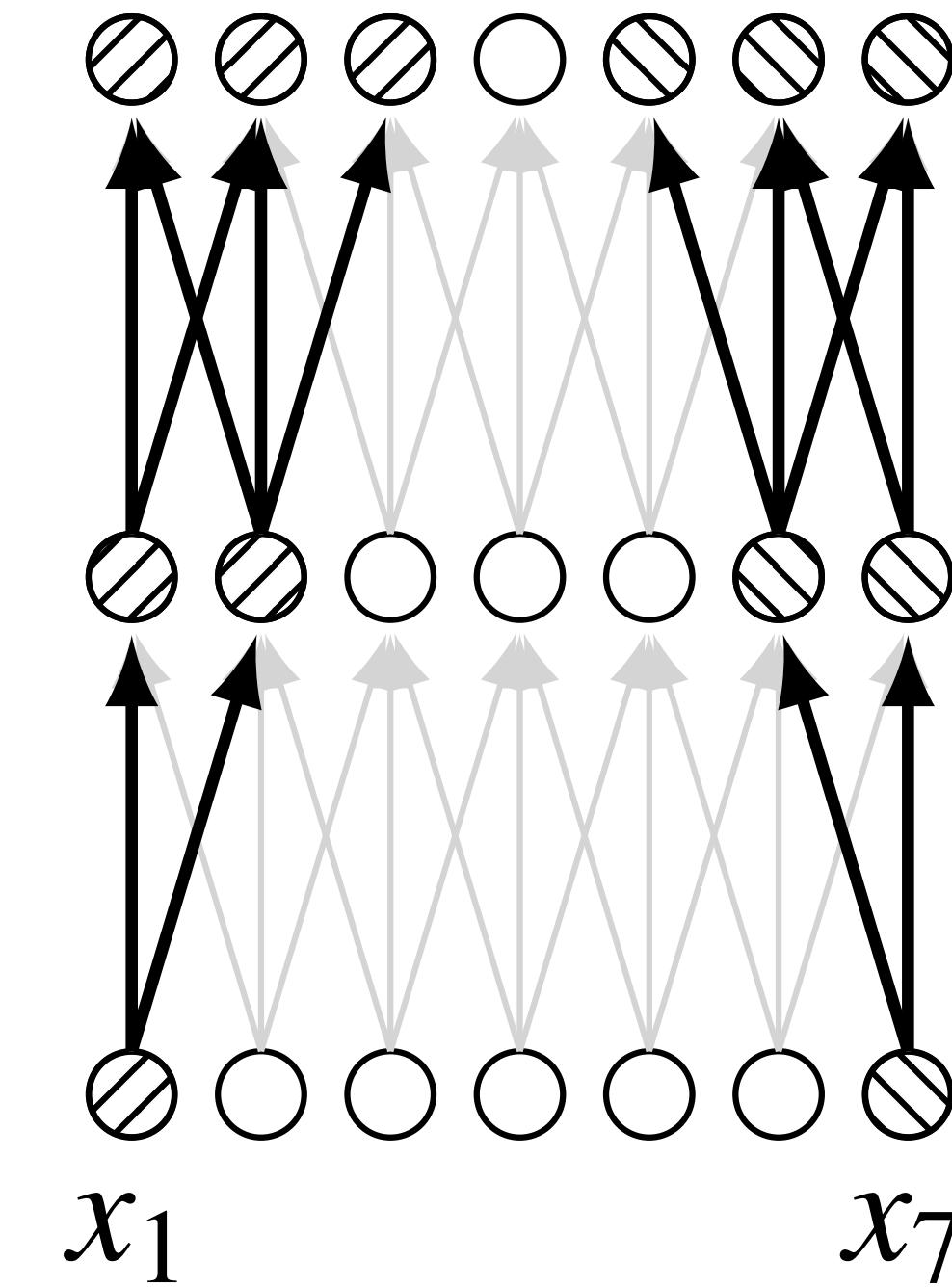


How many birds are in this image?

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

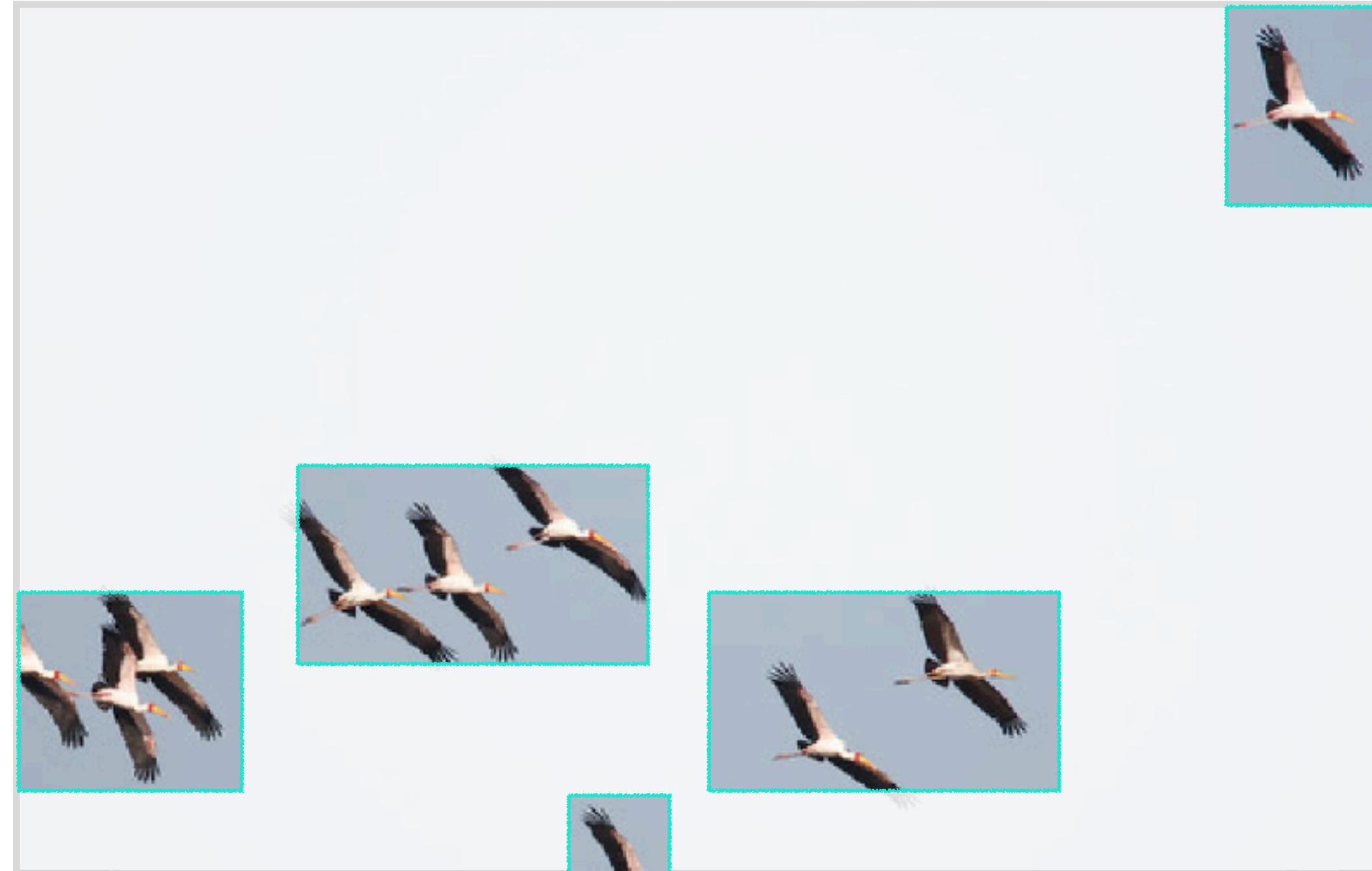
CNNs are built around the idea of locality, and are not well-suited to modeling long distance relationships

# A Limitation of CNNs



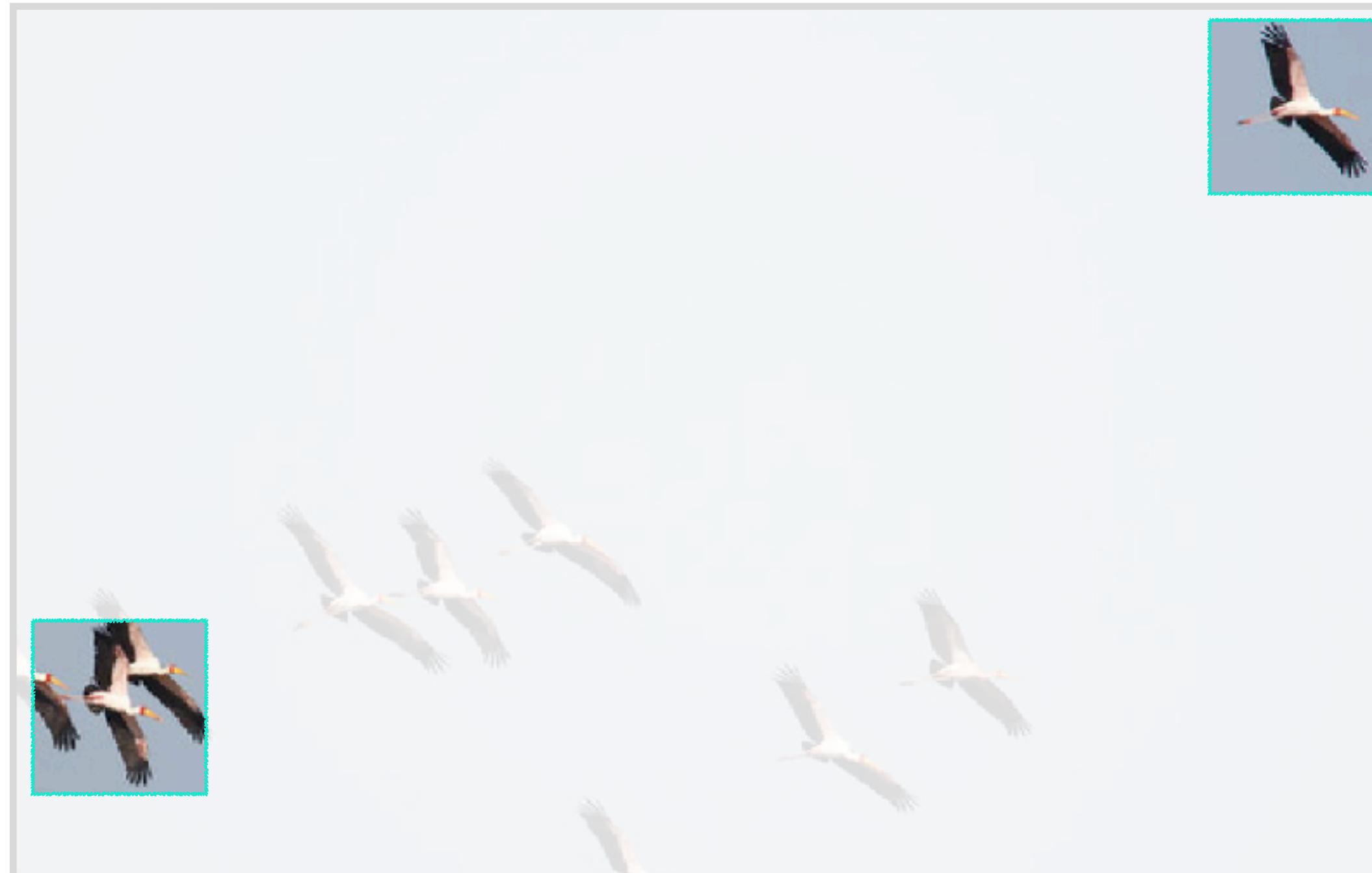
Far apart image patches do not interact

# The Idea of Attention



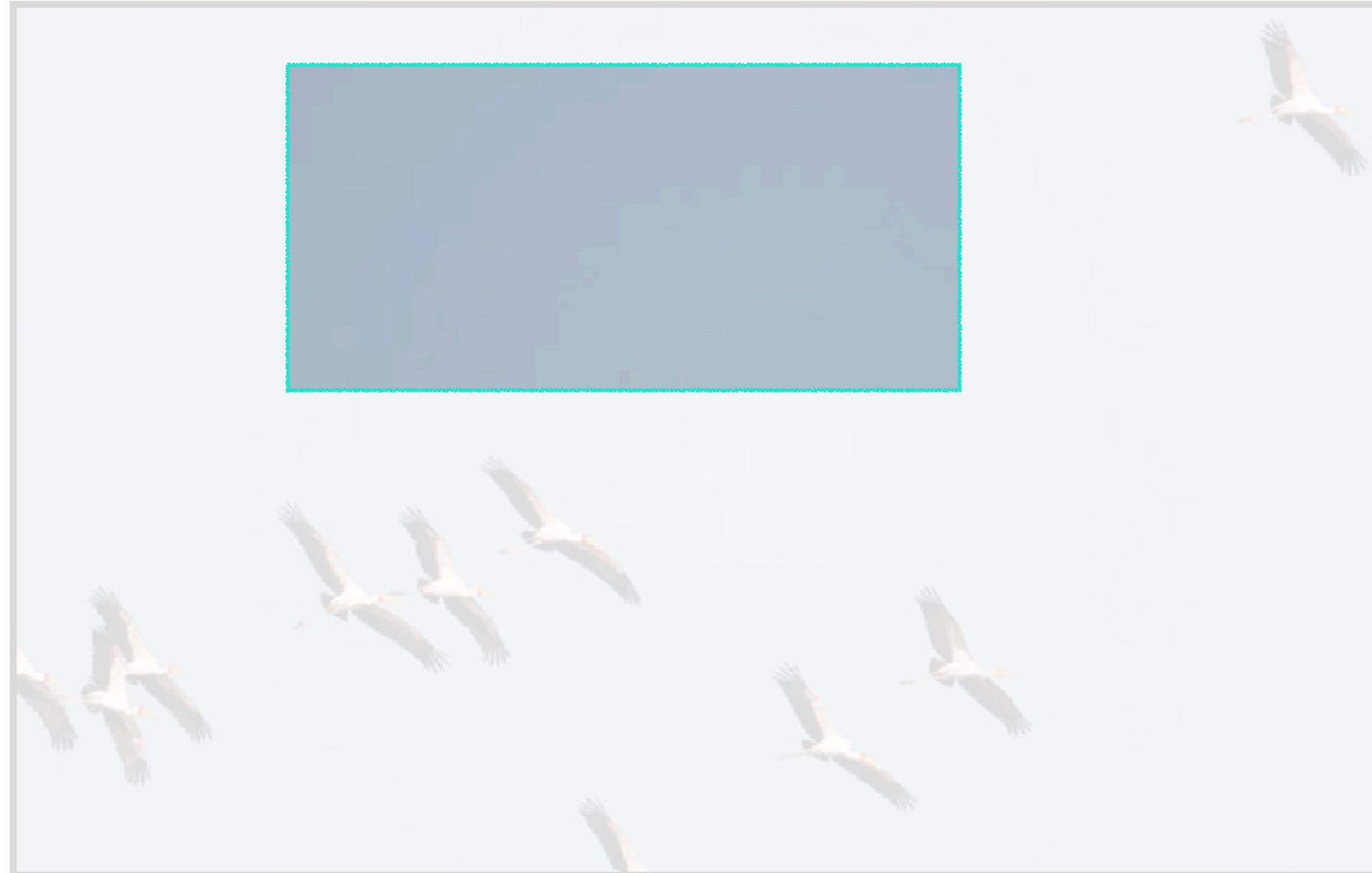
How many birds are in this image?

# The Idea of Attention



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

# The Idea of Attention



What's the color of the sky?

# Three Key Architectural Innovations

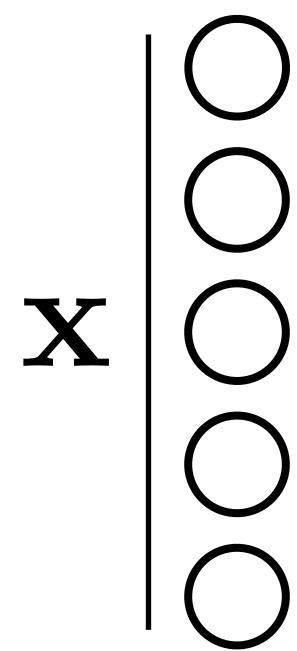
1. Tokens
2. Attention
3. Positional Codes

New idea #1: tokens

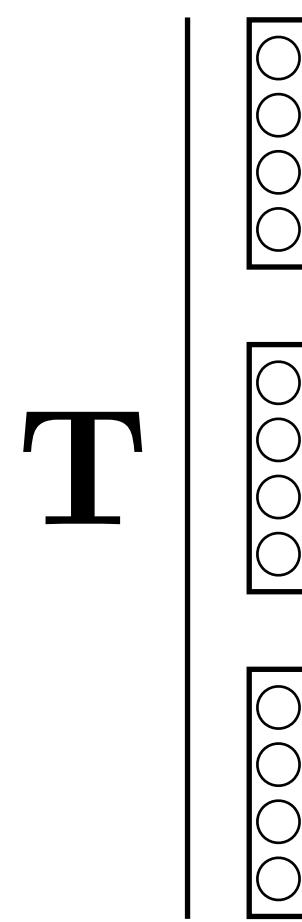
# A New Data Type: Tokens

- 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**

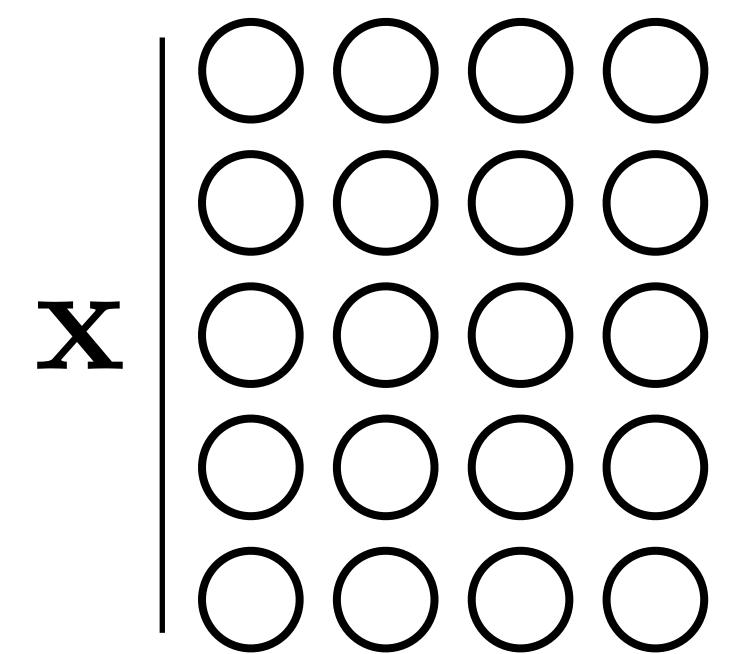


Note: sometimes the word “token” is instead used to refer to the atomic units of the data sequence we will model. In this usage tokens are the representation of the data only at the input and output layers. We use a more general definition where tokens are the representation of the data at *any* layer.

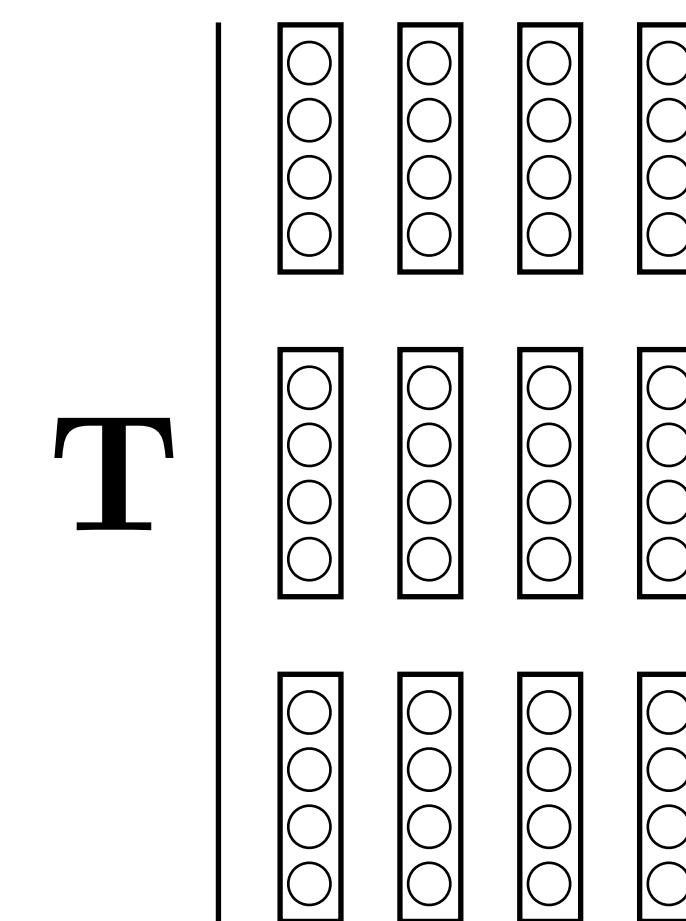
# A new data structure: Tokens

- 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**



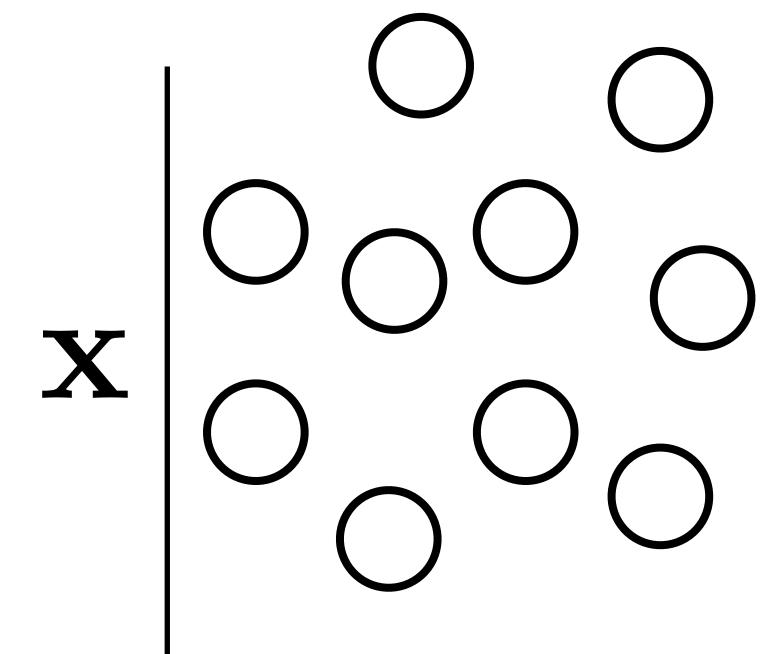
array of **tokens**



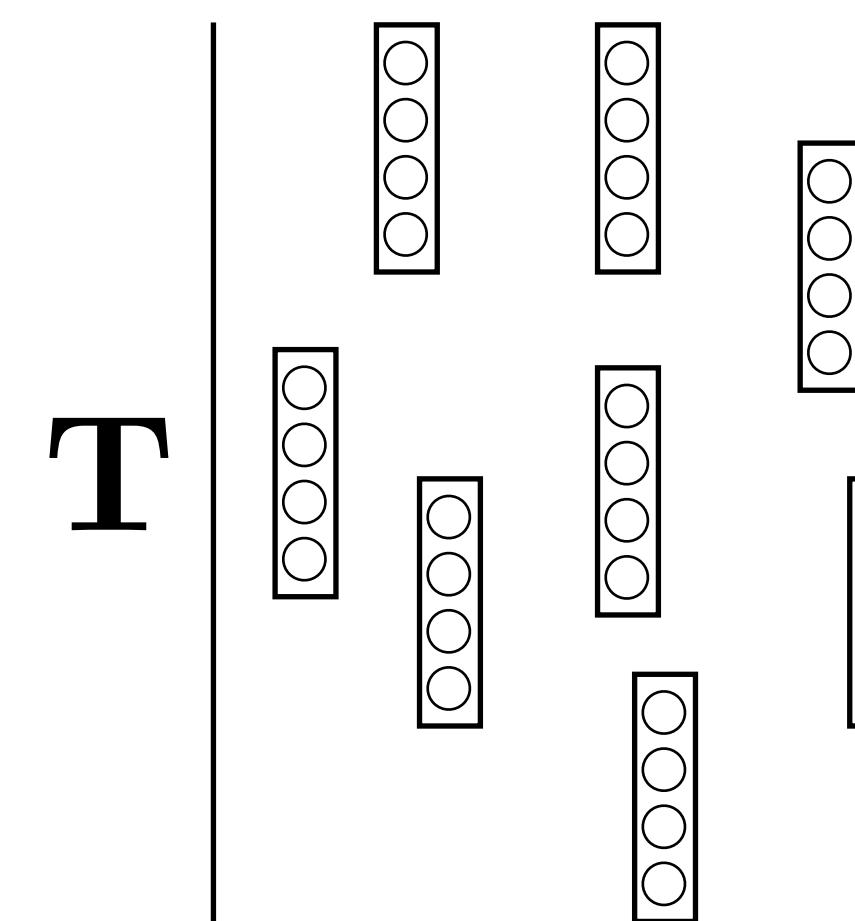
# A new data structure: Tokens

- 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.

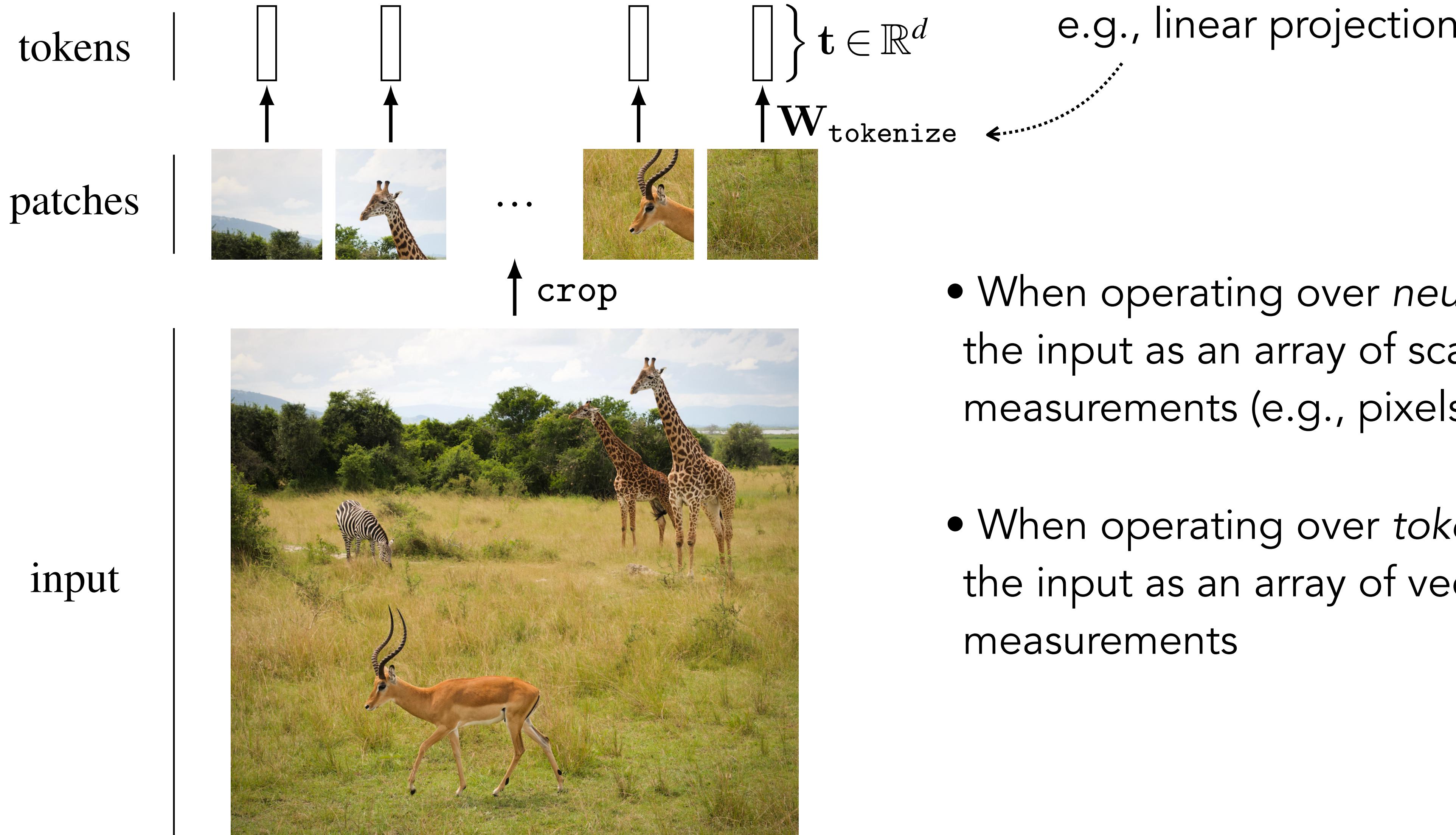
set of **neurons**



set of **tokens**



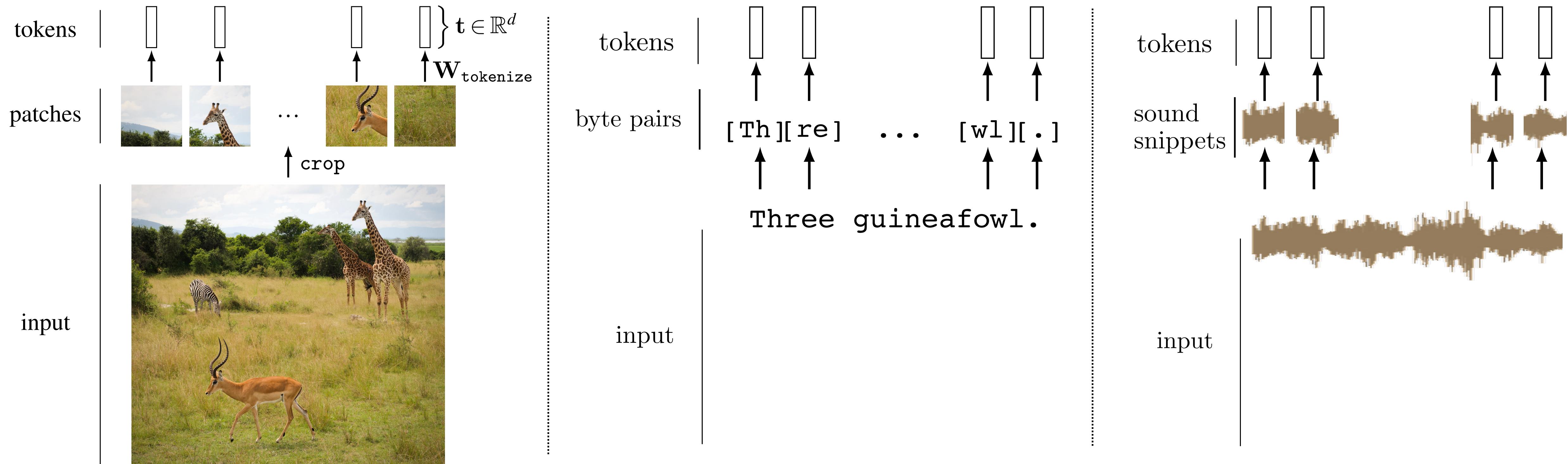
# Tokenizing the input data



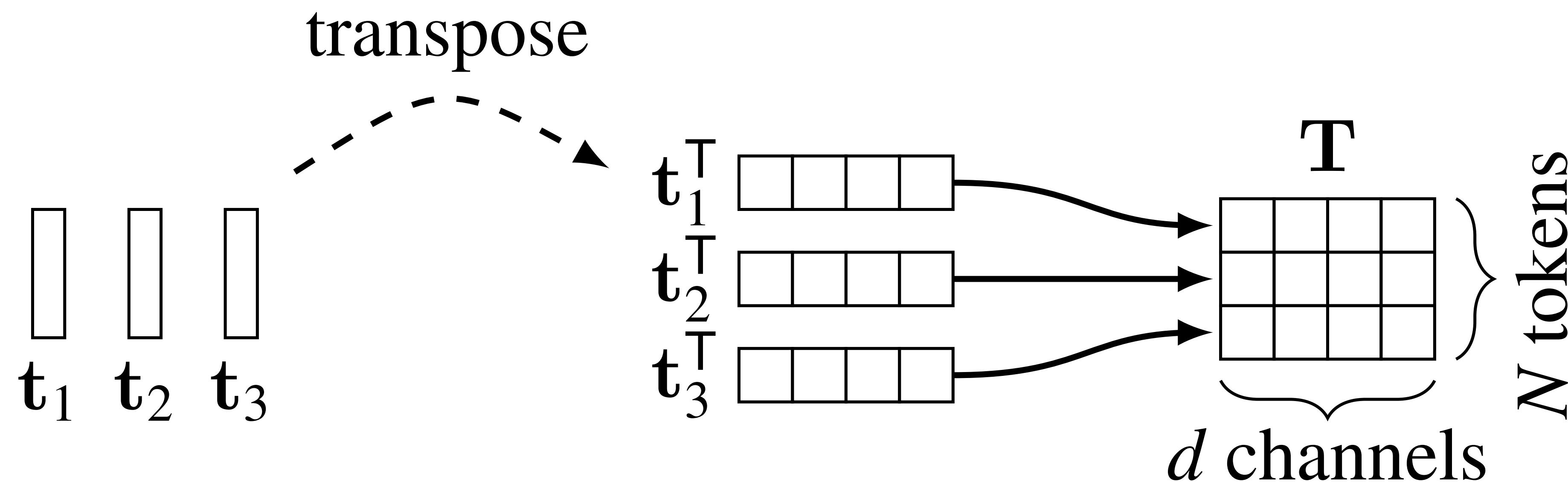
# Tokenizing the input data

You can tokenize anything.

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

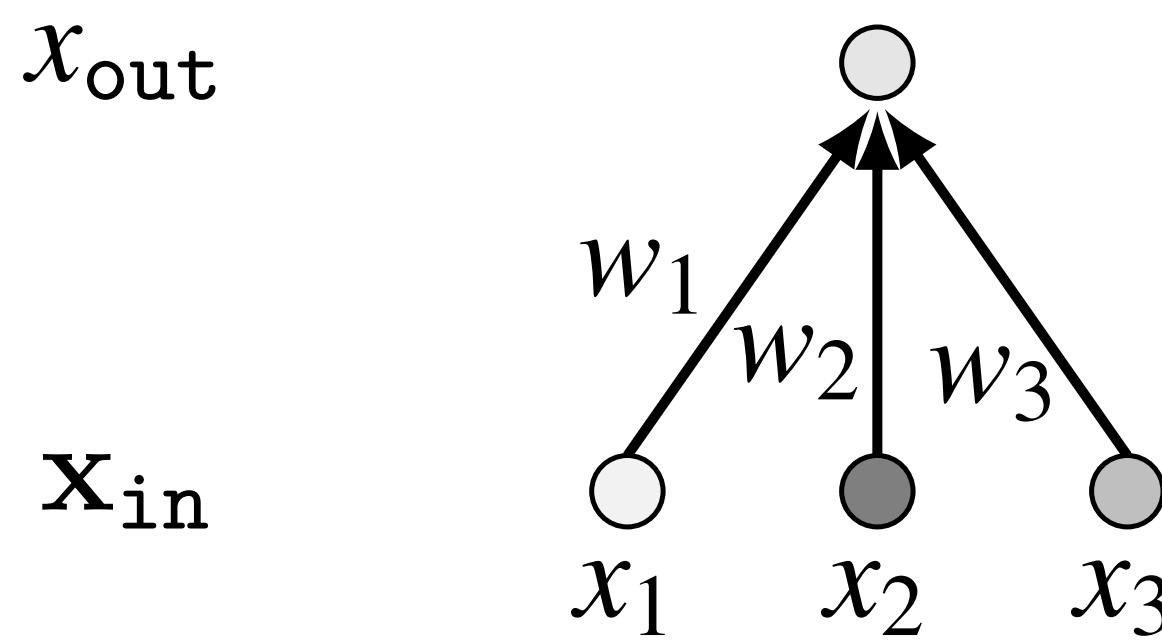


# Notation



# Linear combination of tokens

## Linear combination of neurons

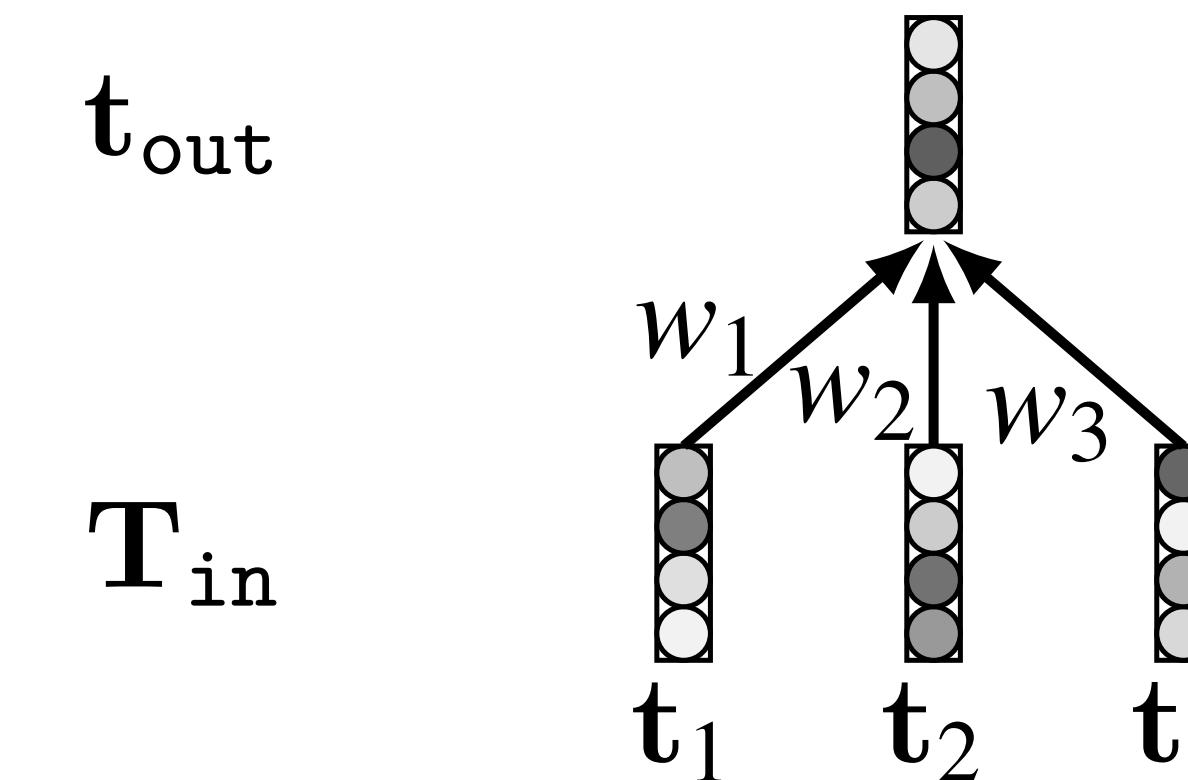


$$x_{out} = w_1x_1 + w_2x_2 + w_3x_3$$

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

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

## Linear combination of tokens



$$t_{out} = w_1t_1 + w_2t_2 + w_3t_3$$

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

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

# Token-wise nonlinearity

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

$F$  is typically an MLP

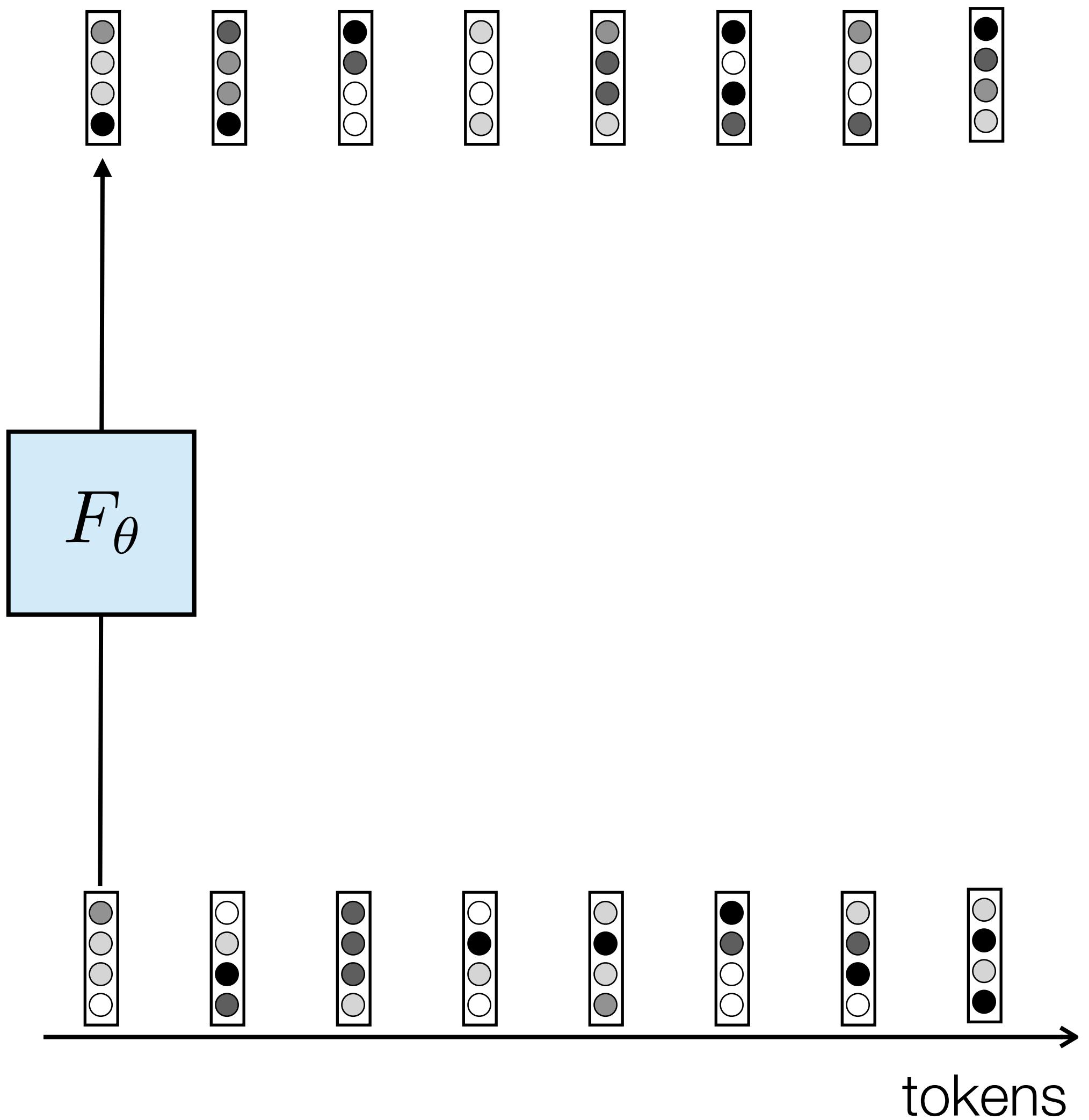
Equivalent to a CNN with  $1 \times 1$  kernels run over token sequence

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

# Token-wise nonlinearity

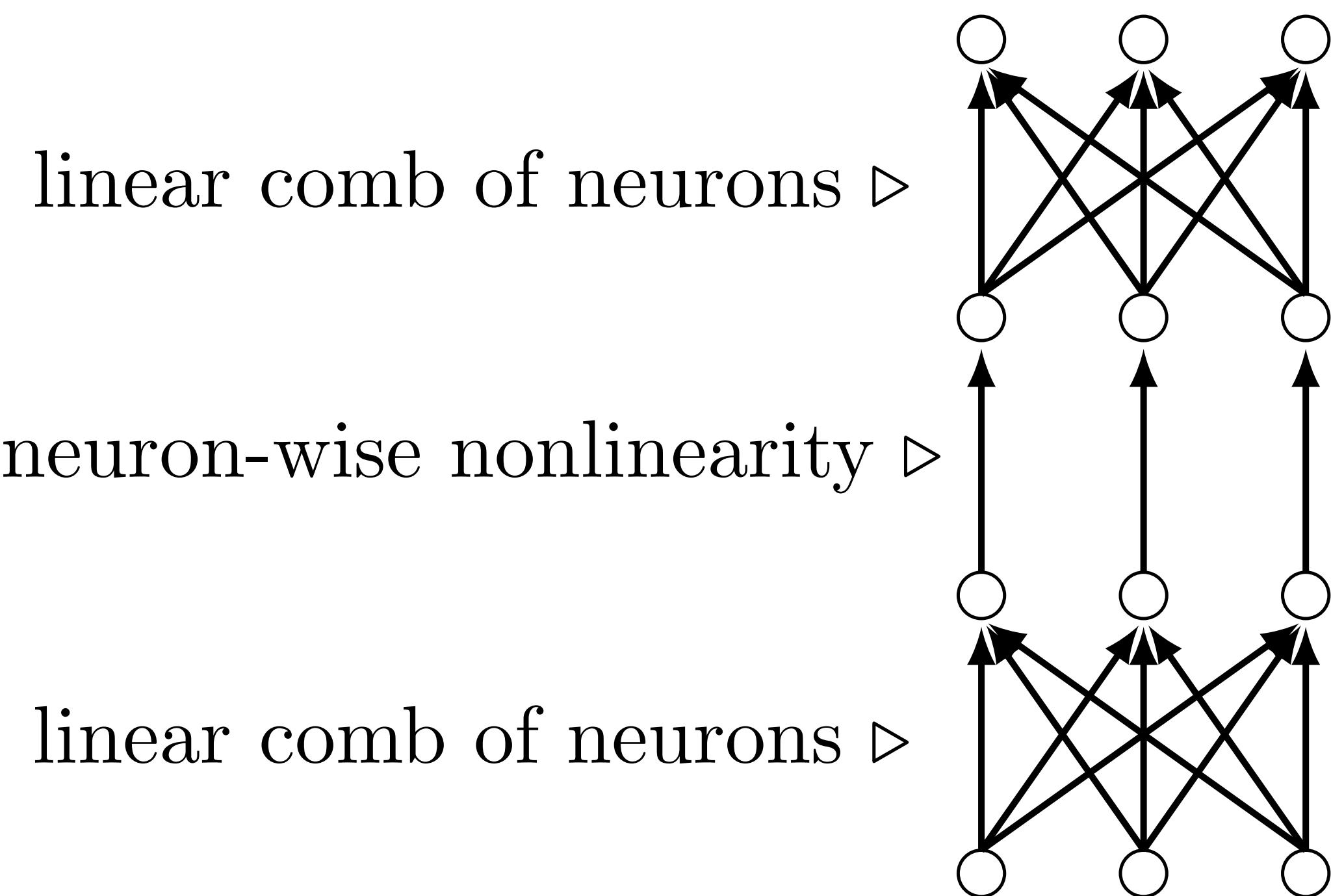
$$\mathbf{x}_{\text{out}} = \begin{bmatrix} \text{relu}(x_{\text{in}}[0]) \\ \vdots \\ \text{relu}(x_{\text{in}}[N-1]) \end{bmatrix}$$

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

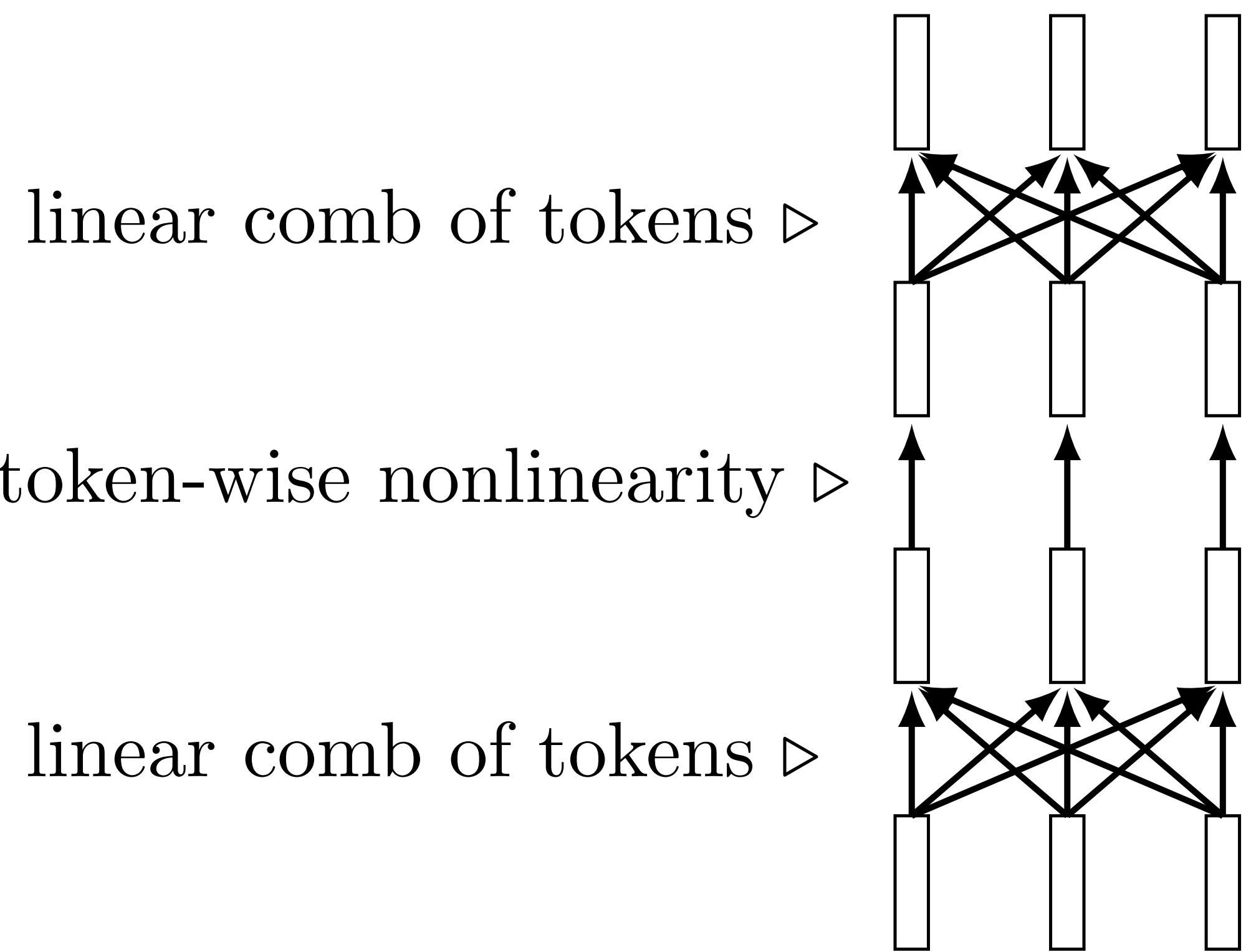


# Token nets

**Neural net**

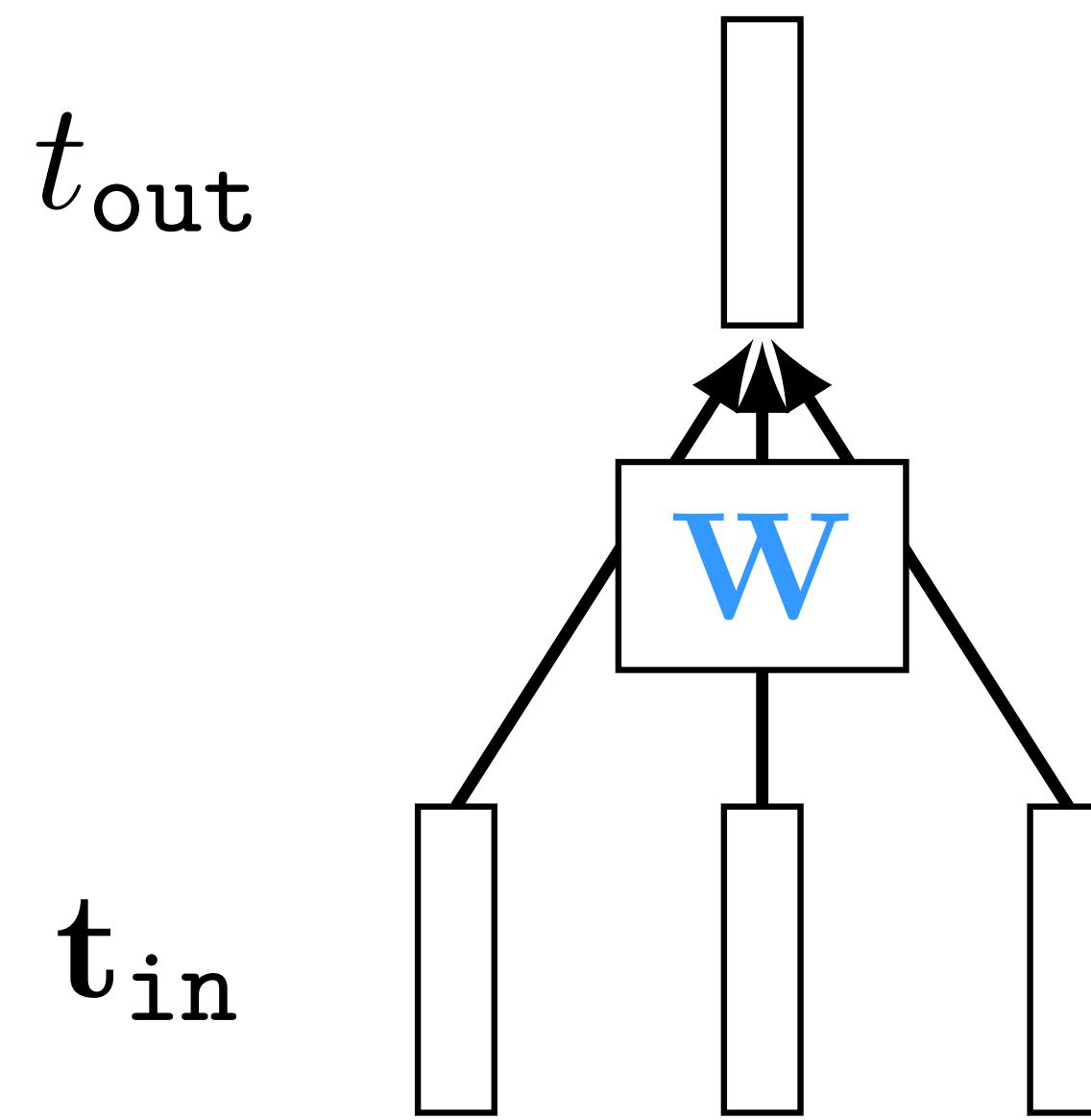


**Token net**

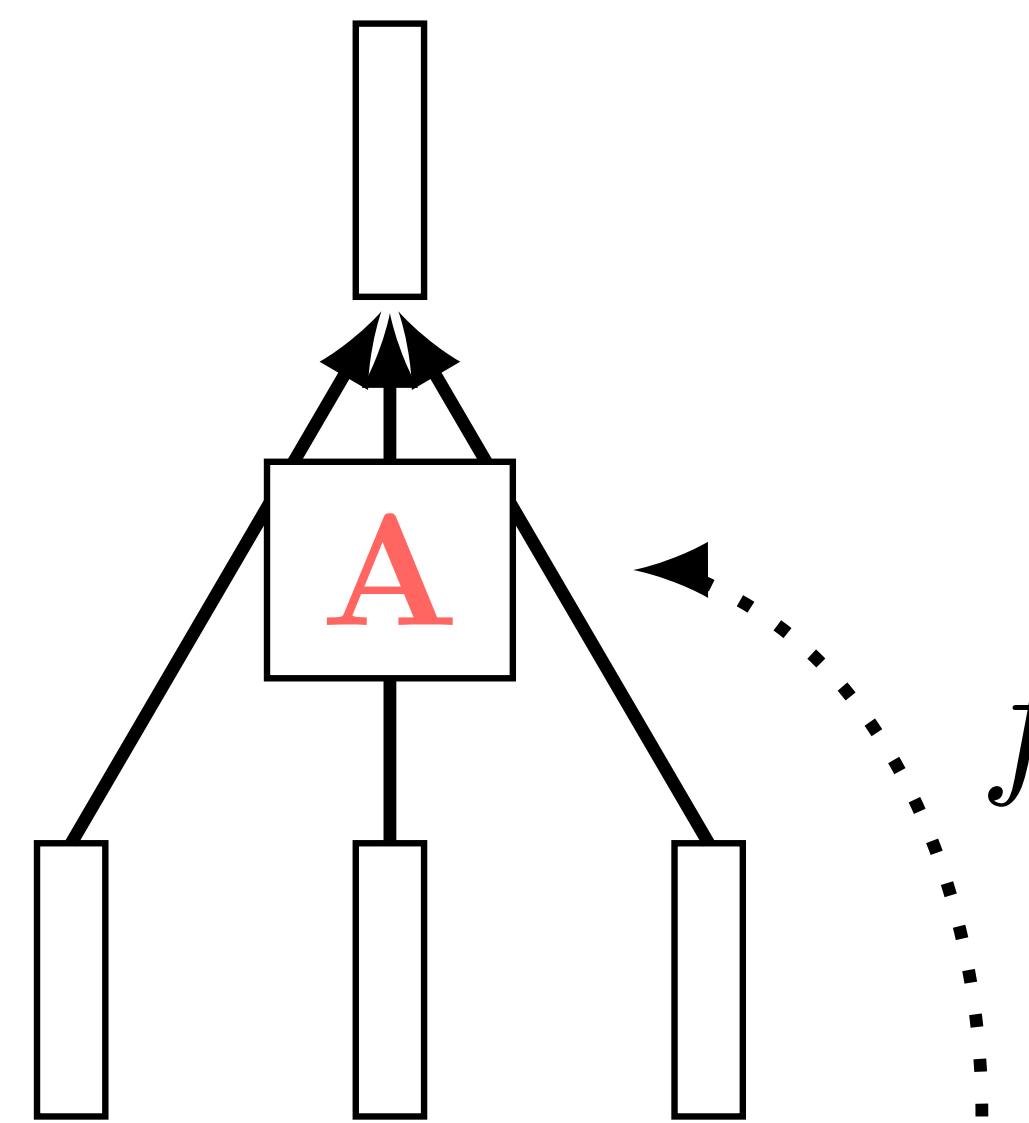


New idea #2: attention

fc layer



attn layer



$$A = f(\dots)$$

▷ attention

$$T_{\text{out}} = AT_{\text{in}}$$

$W$  is free parameters.

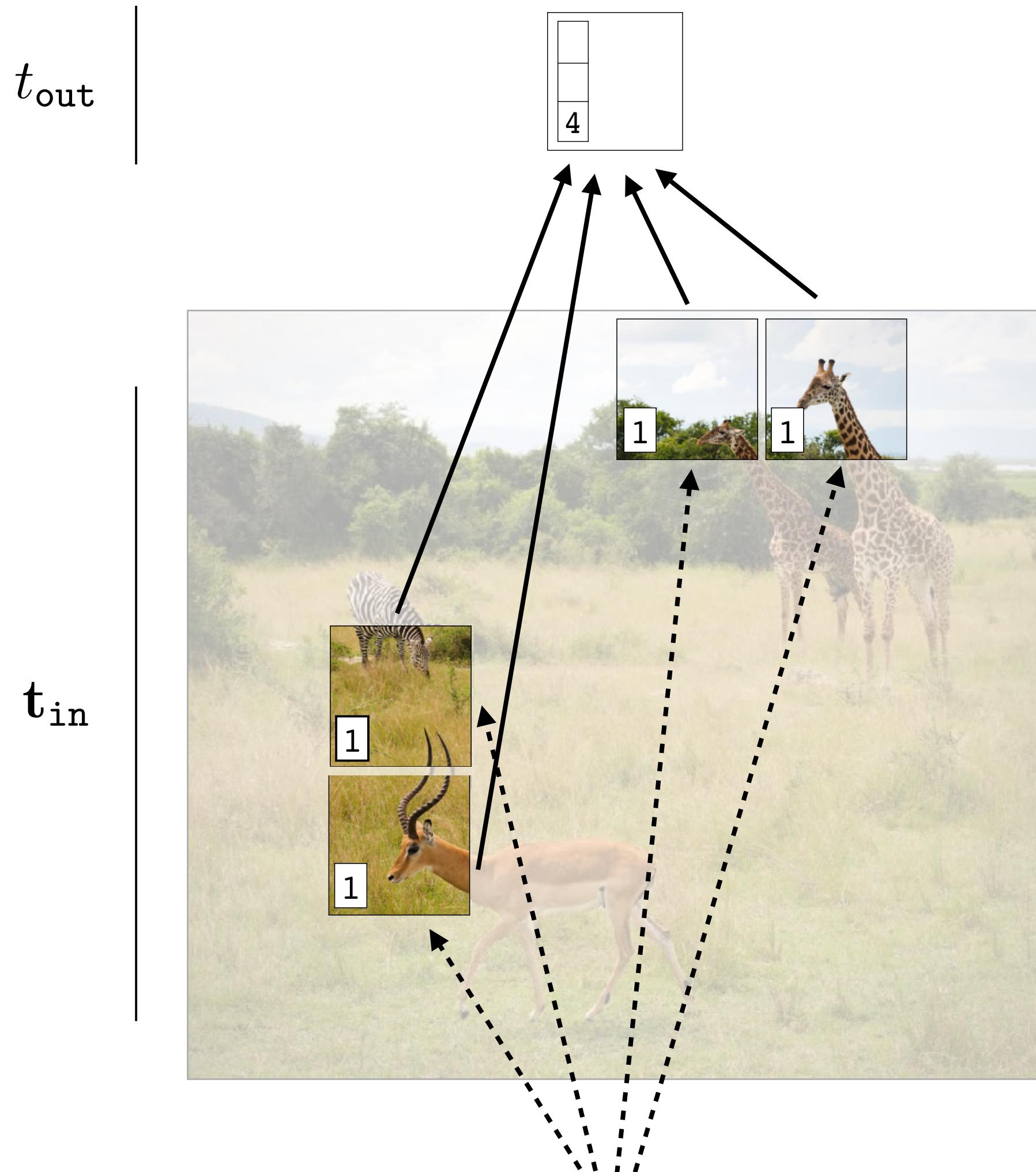
$A$  is a function of some input data. The data tells us which tokens to attend to (assign high weight in weighted sum)

$t_{\text{out}}$

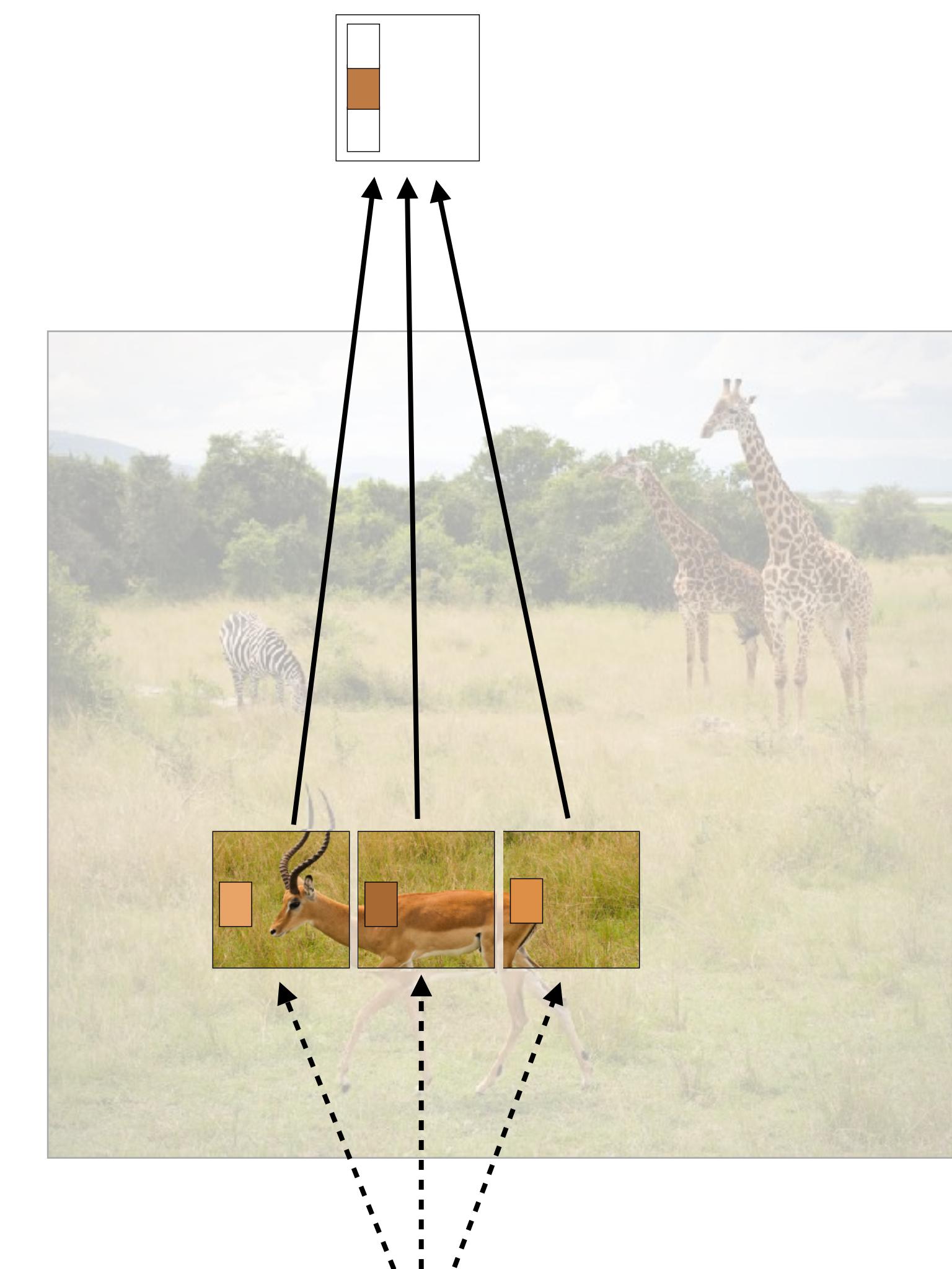
$t_{\text{in}}$



How many  
animals are  
in the photo?



How many  
animals are  
in the photo?



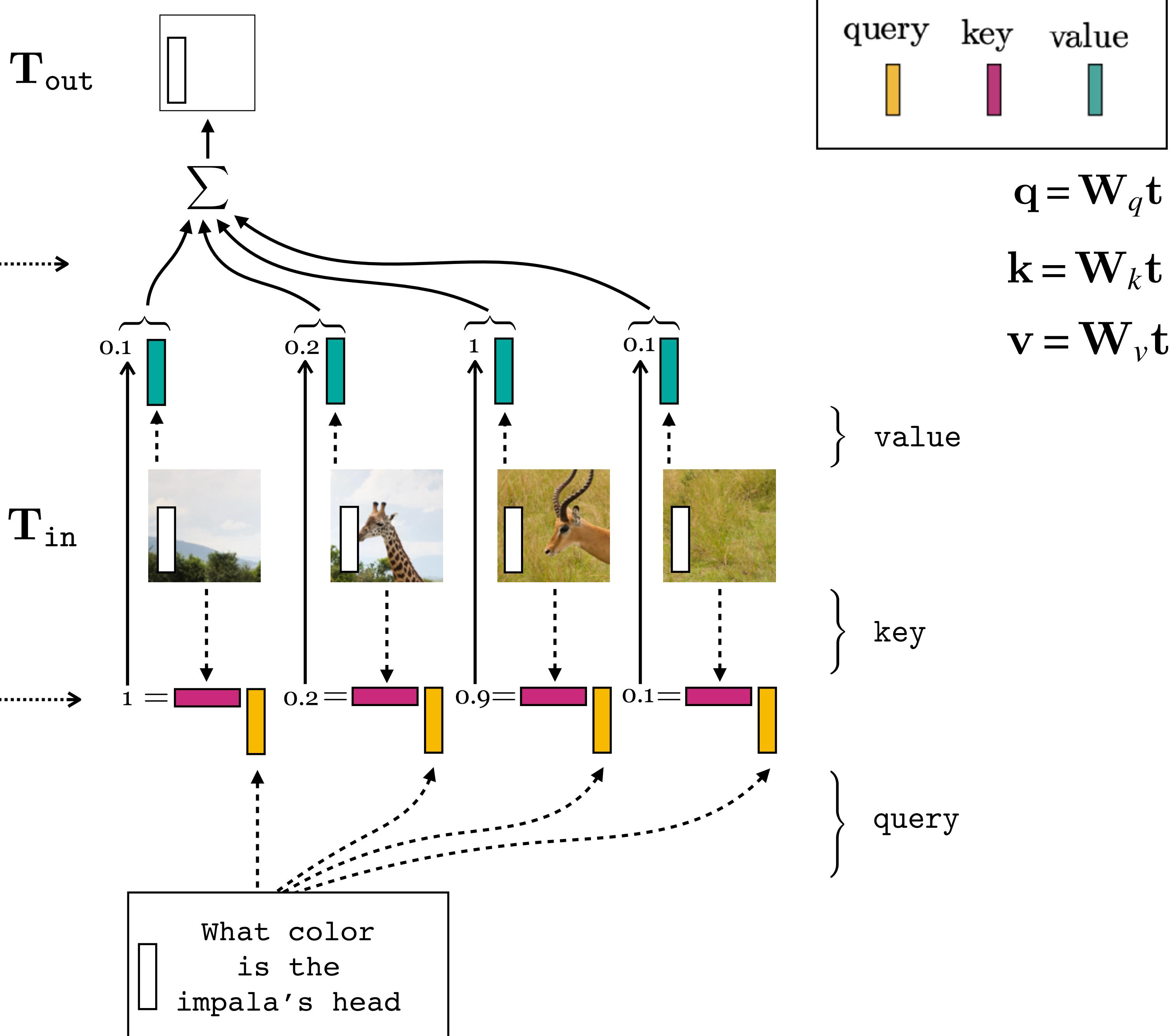
What is the  
color of the  
impala?

# query-key-value attention

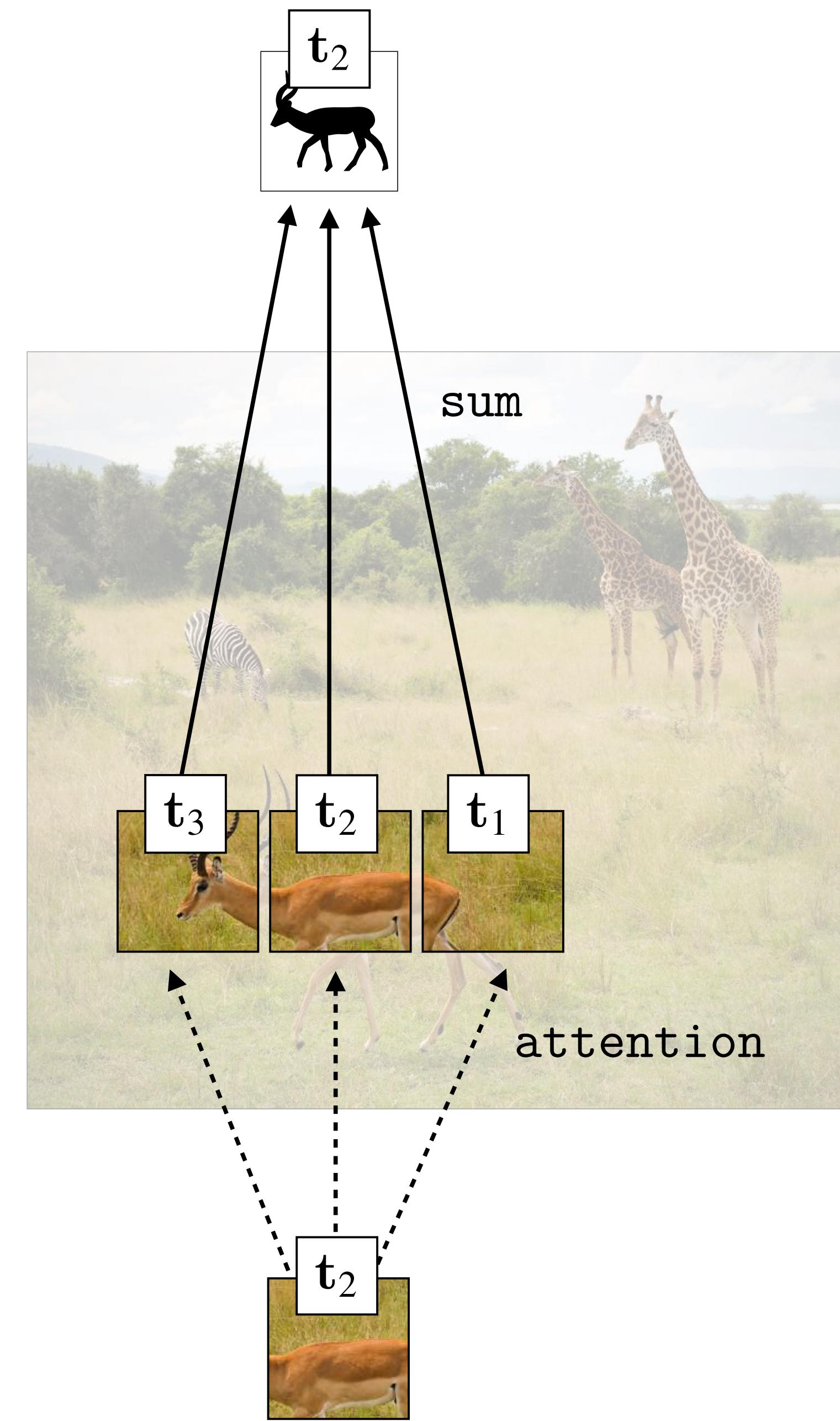
$$A = \text{softmax}(s)$$

$$T_{\text{out}} = \begin{bmatrix} a_1 v_1^T \\ \vdots \\ a_N v_N^T \end{bmatrix}$$

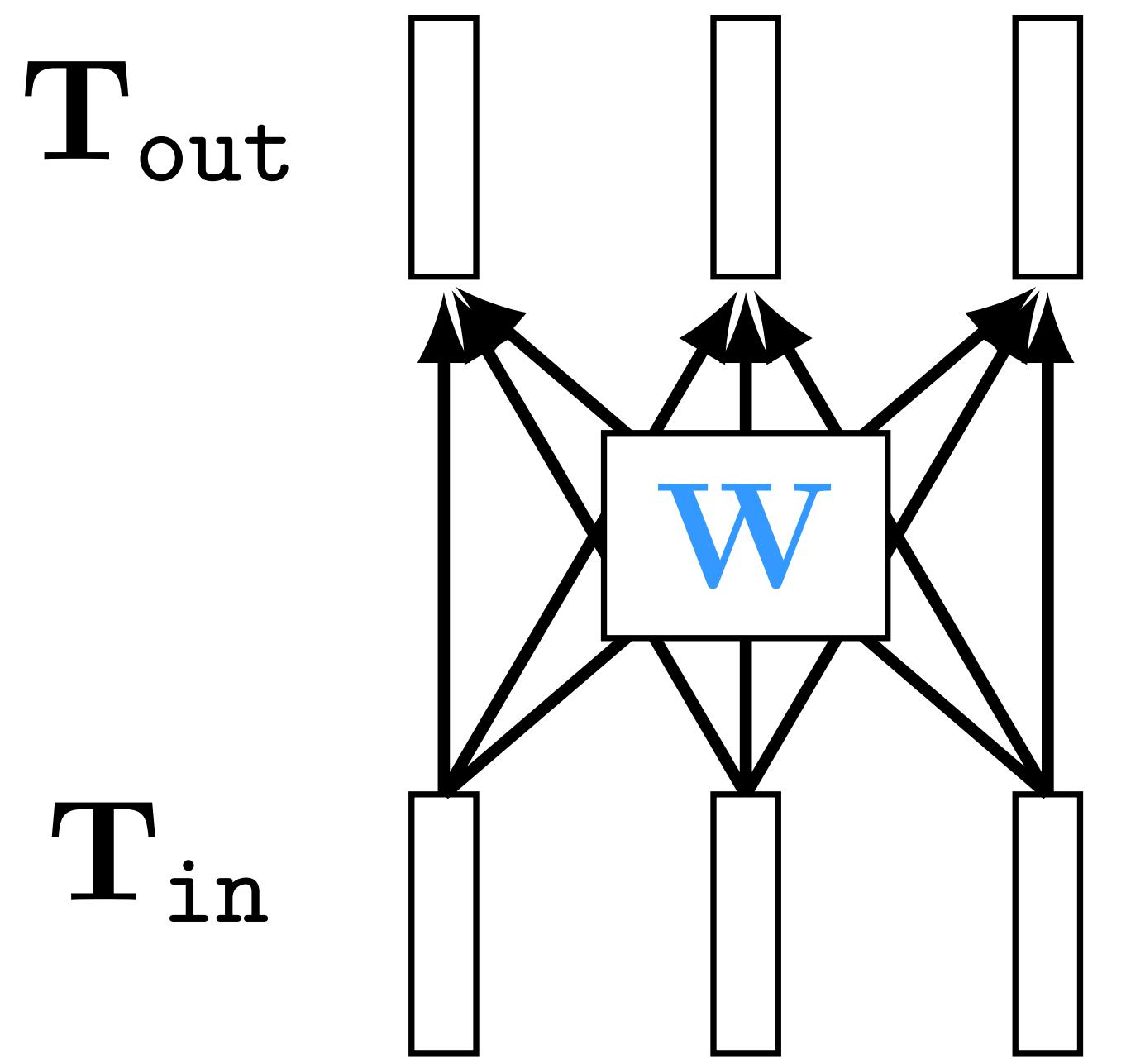
$$s = [q_{\text{question}}^T k_1, \dots, q_{\text{question}}^T k_N]$$



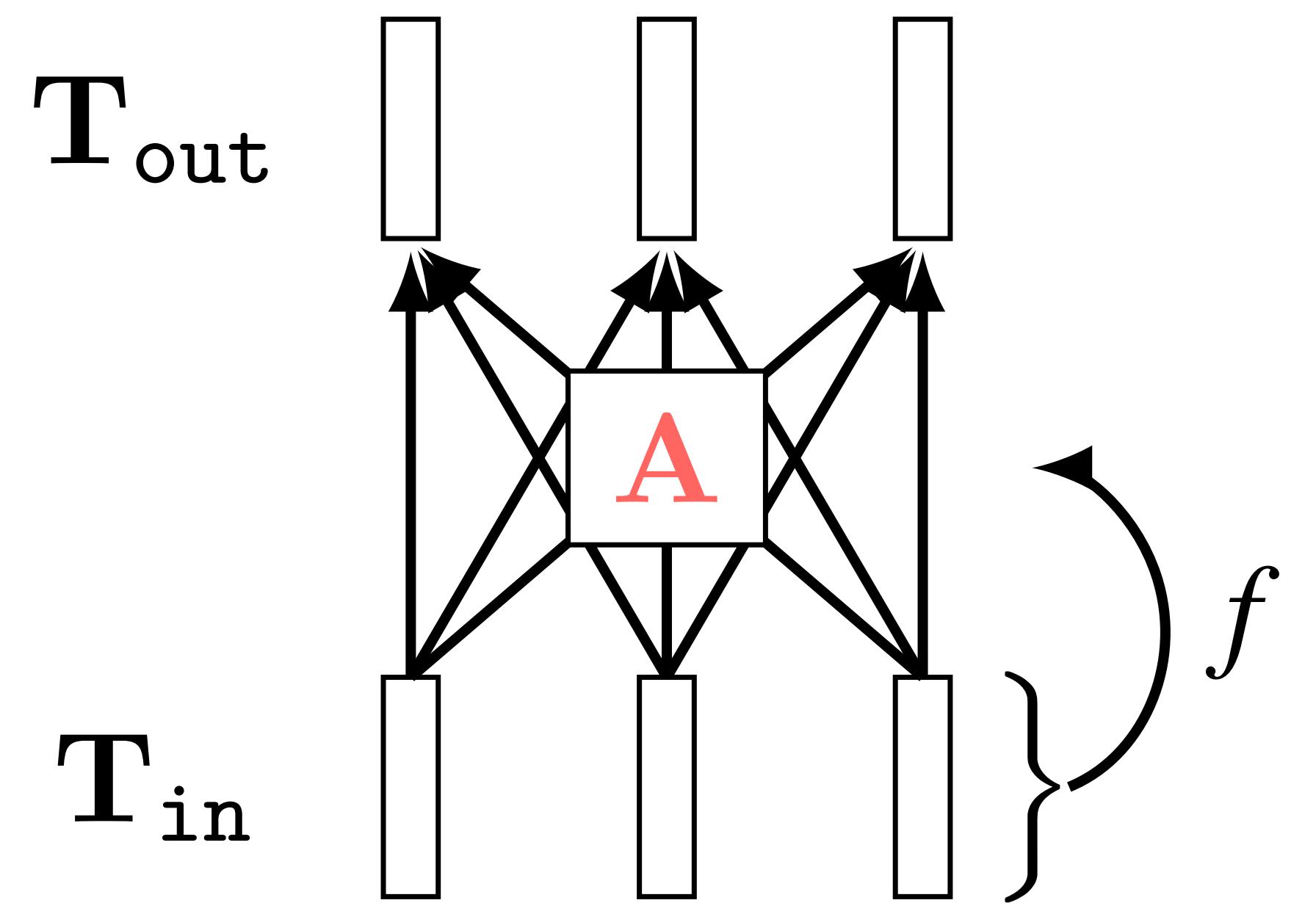
# Self-attention



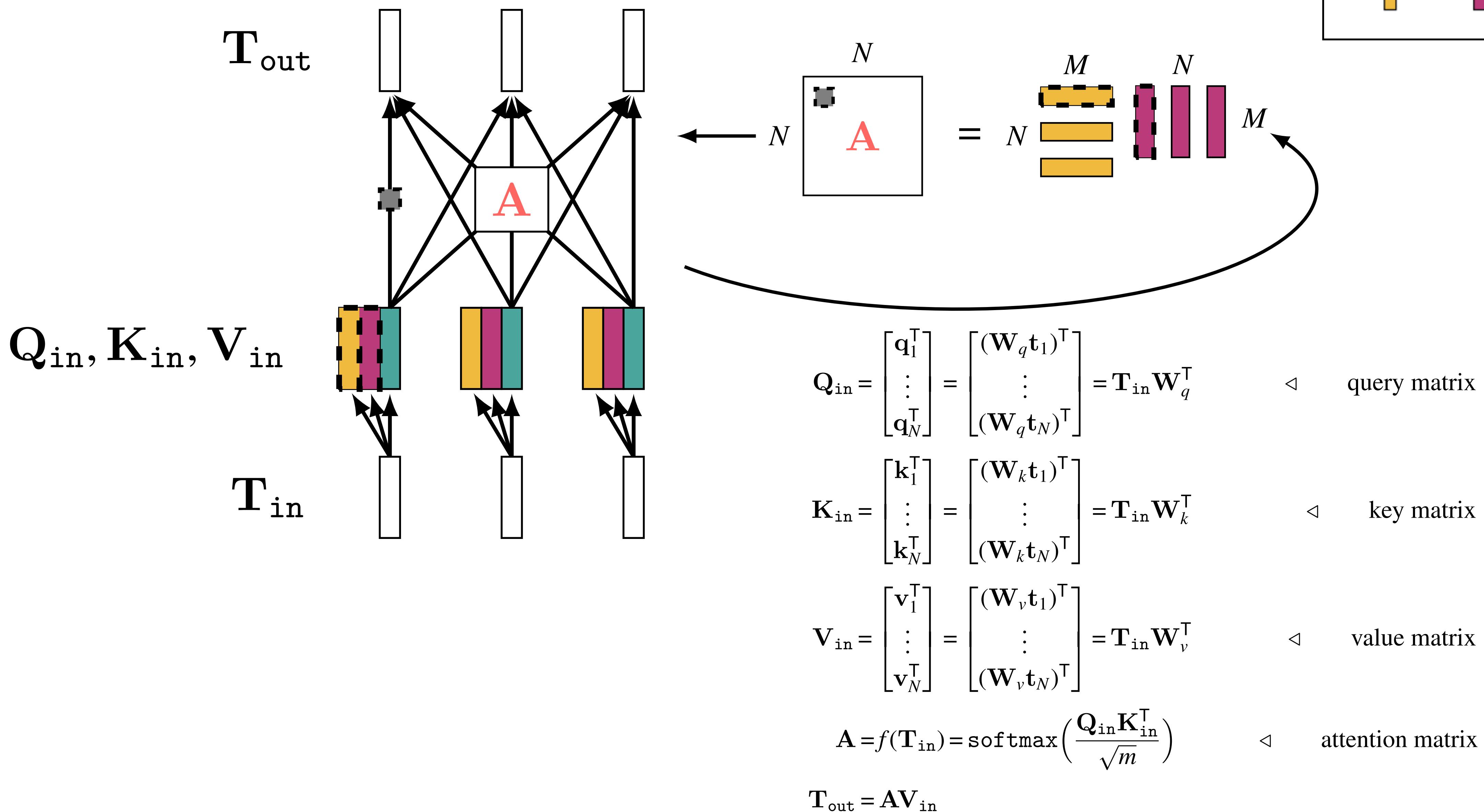
## fc layer



## self attn layer

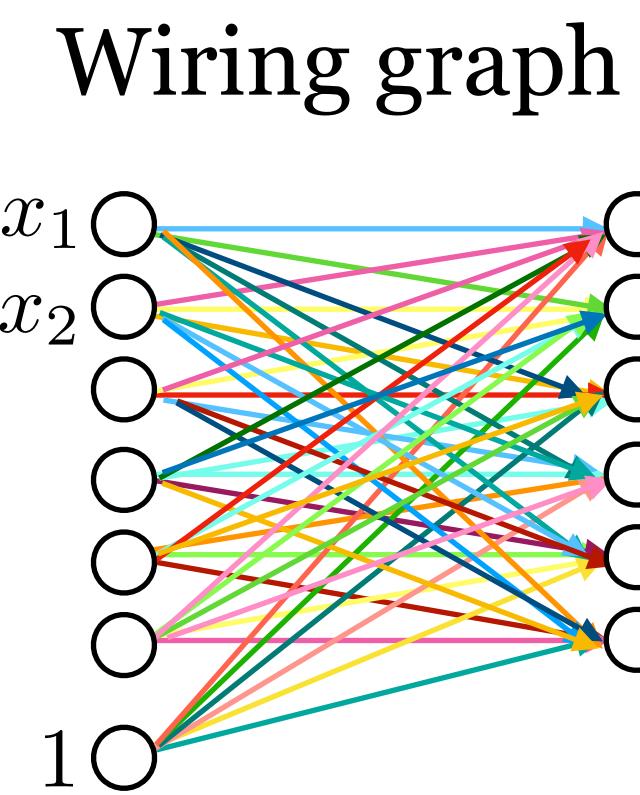


# self attn layer (expanded)



# A family of linear layers

fc

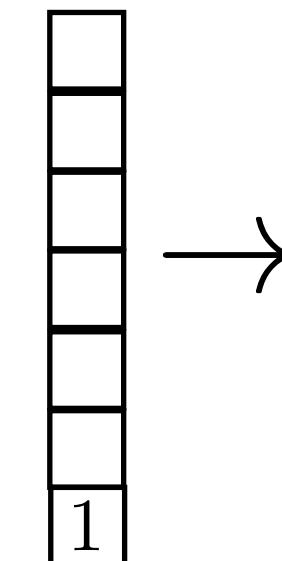
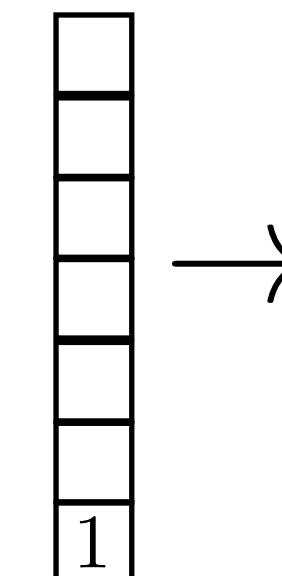
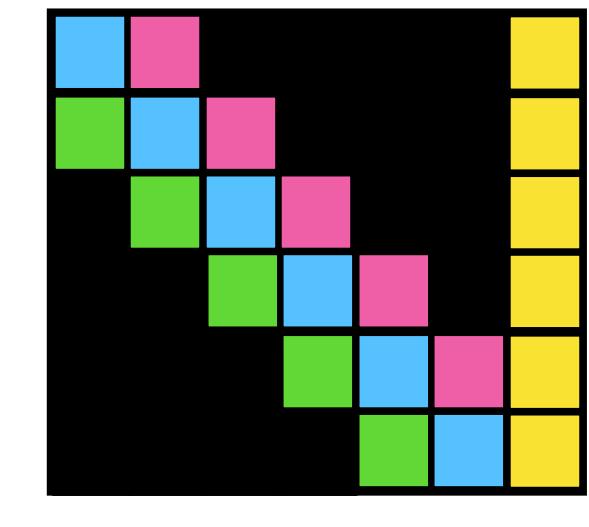
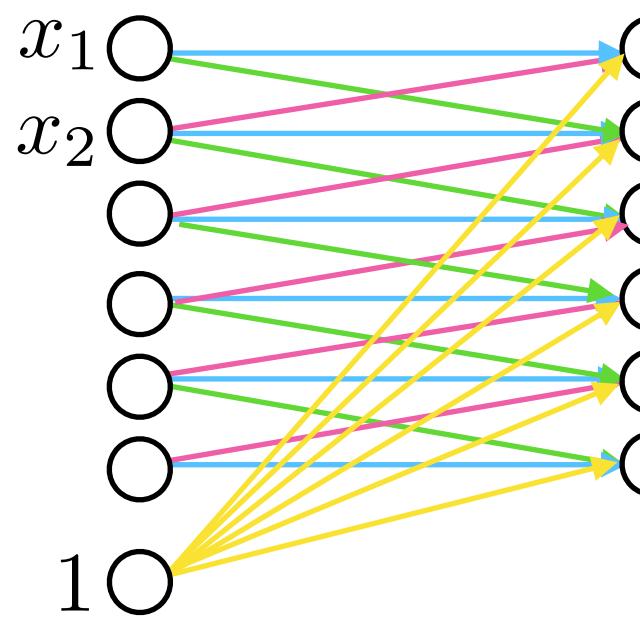


Properties

Fixed input dimensionality

$N^2$  learnable parameters

conv

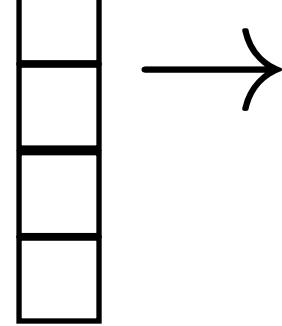
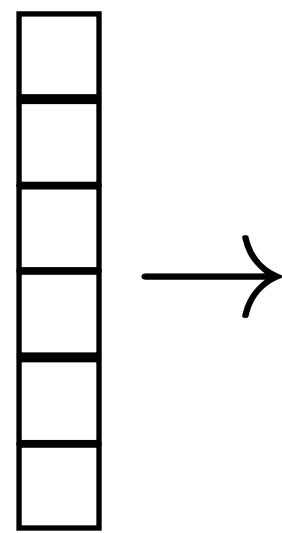
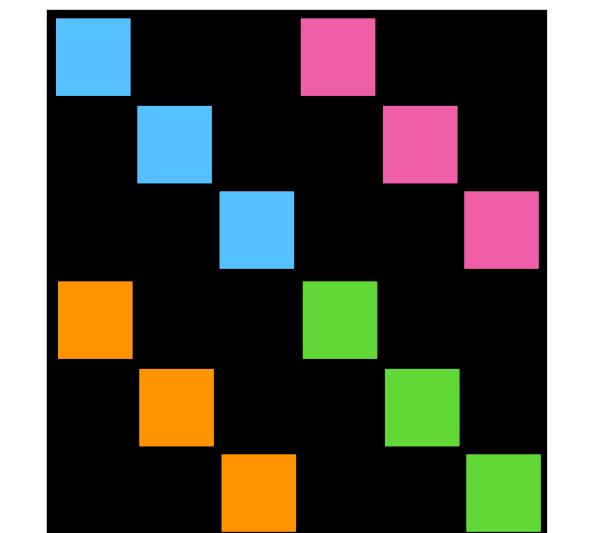
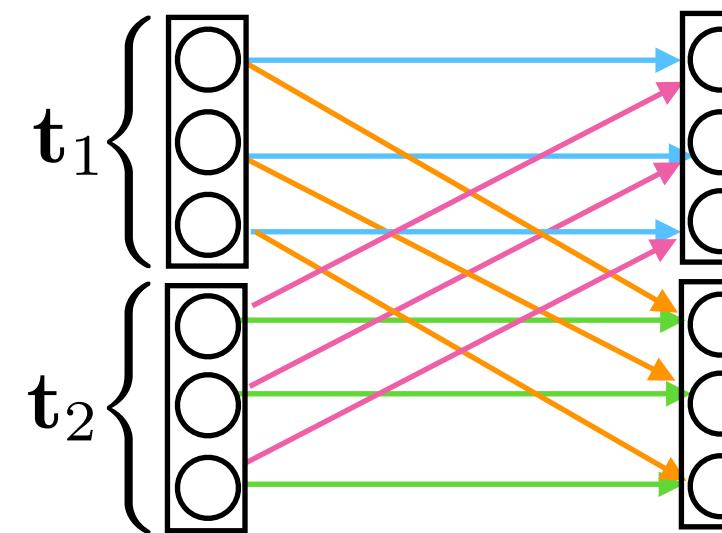


Variable input dimensionality

$k+1$  learnable parameters ( $k$  = kernel size)

$\text{conv}(\text{translate}(x)) = \text{translate}(\text{conv}(x))$

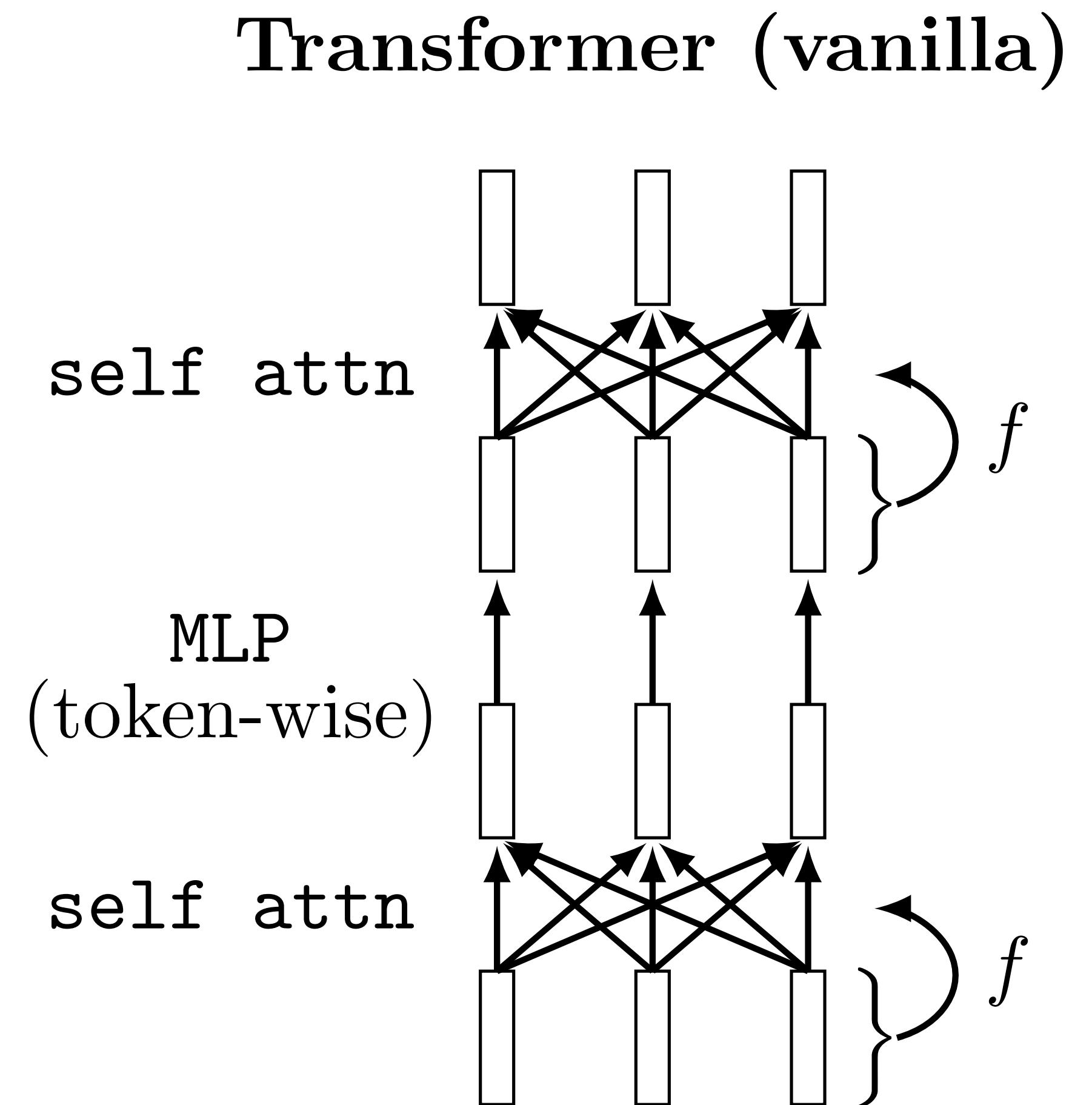
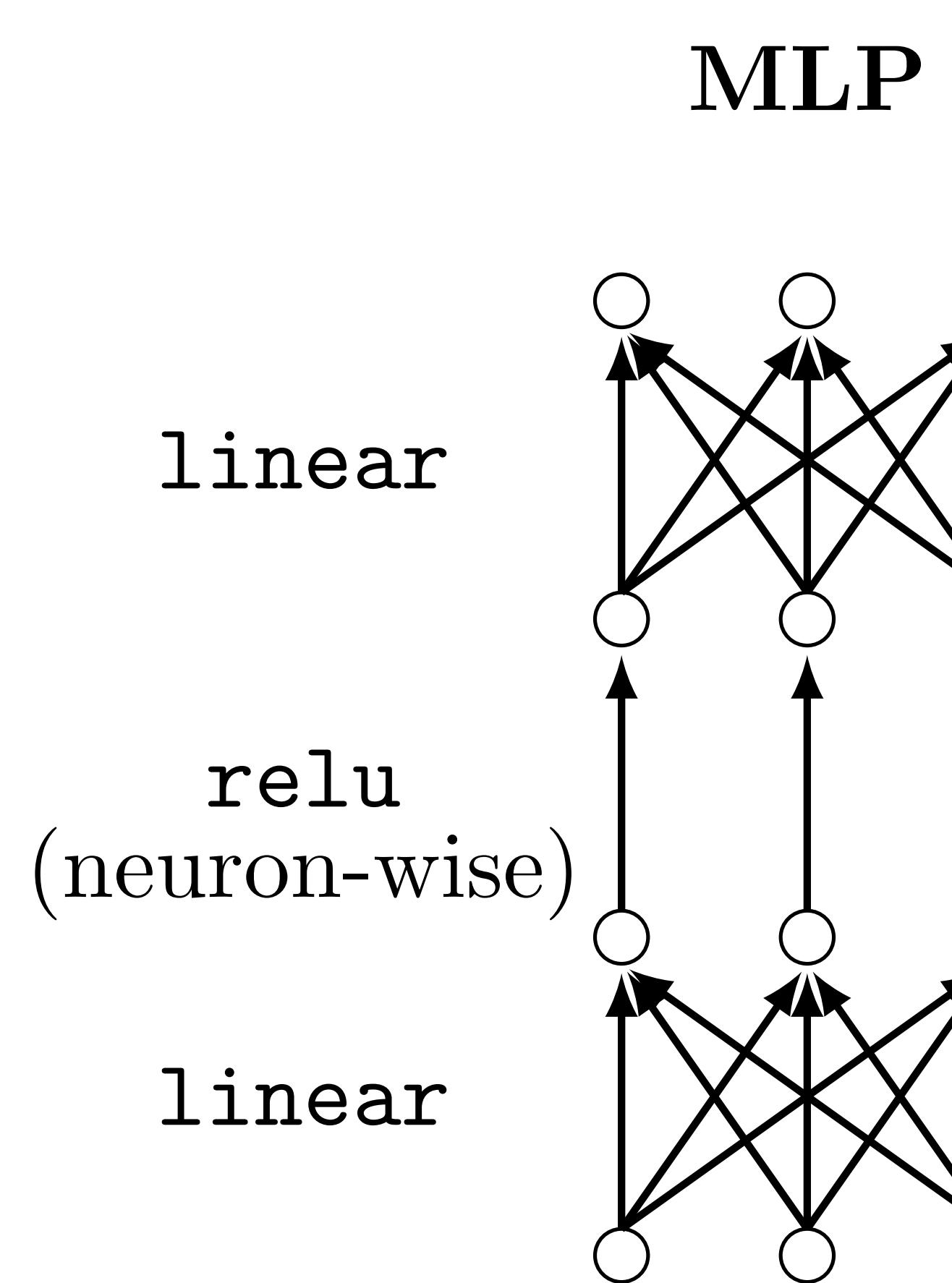
attn



Variable input dimensionality

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

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



# 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.

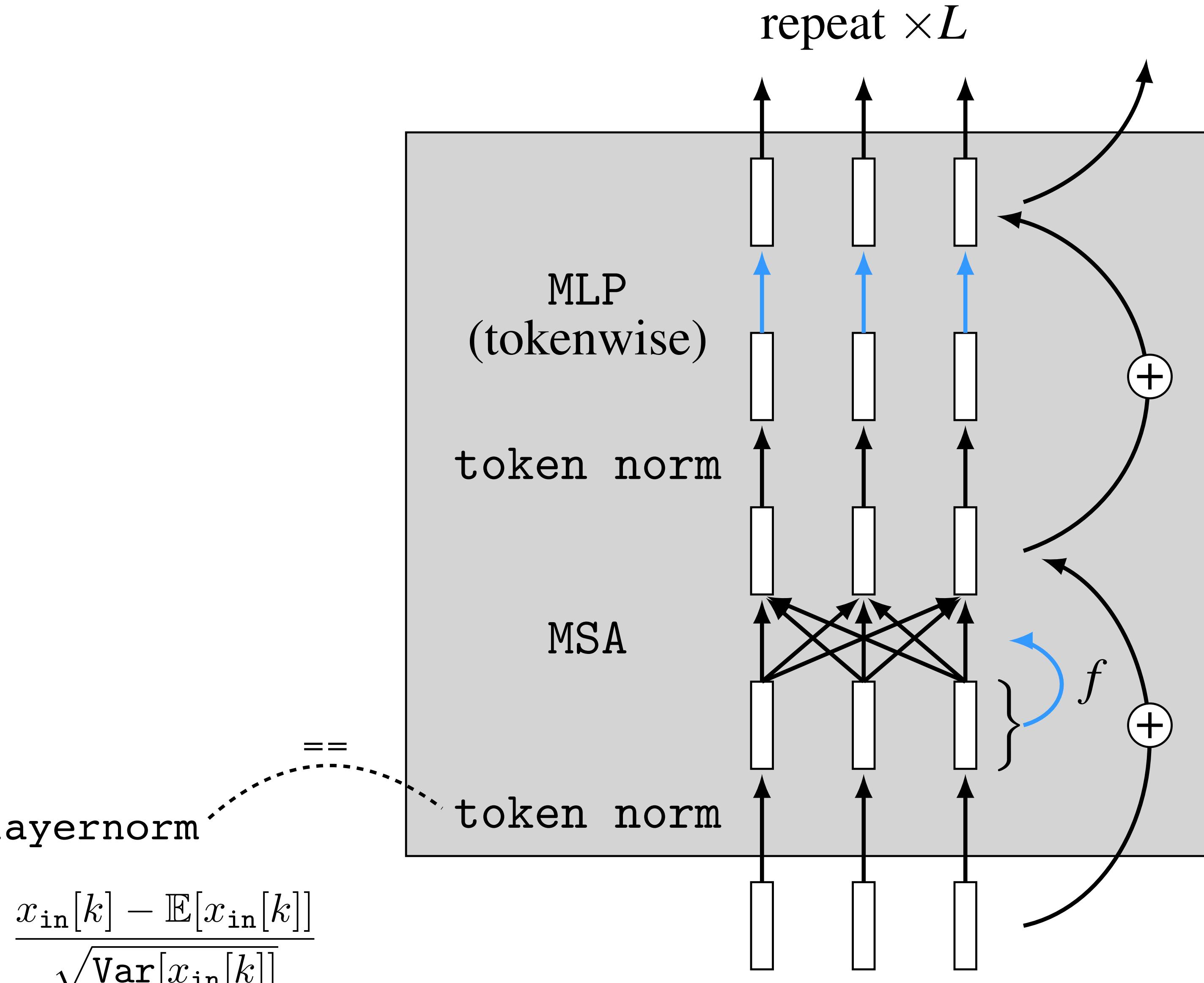
Run them all in parallel, then (weighted) sum the output token code vectors

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

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

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

# Transformer (ViT)



```
# 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):

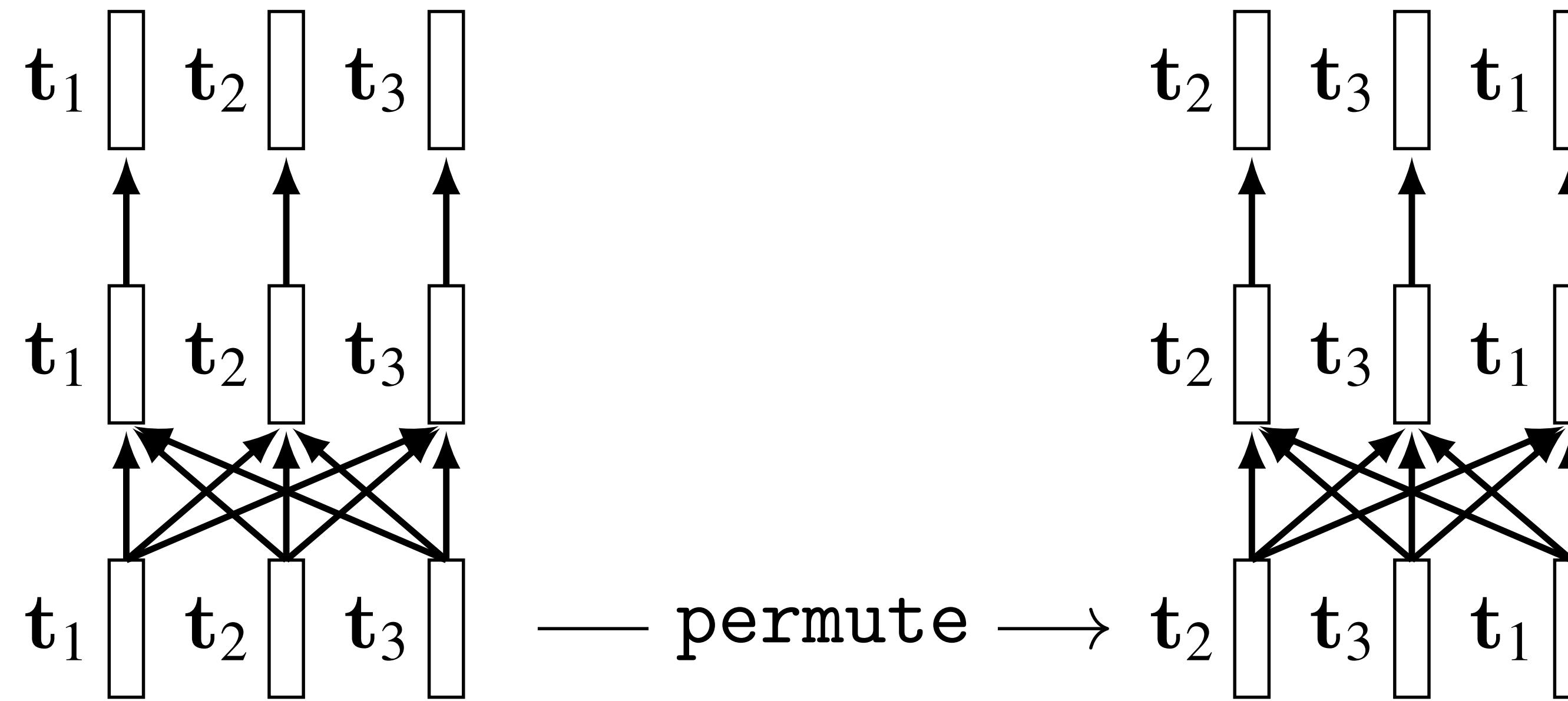
    # attention layer
    Q, K, V = nn.matmul(nn.layernorm(T), [W_q_T[l], W_k_T[l], W_v_T[l]])
    # nn.matmul does matrix multiplication
    A = nn.softmax(nn.matmul(Q, K.transpose()), dim=0)/sqrt(d)
    T = nn.matmul(A, V) + T # note residual connection

    # 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



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

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

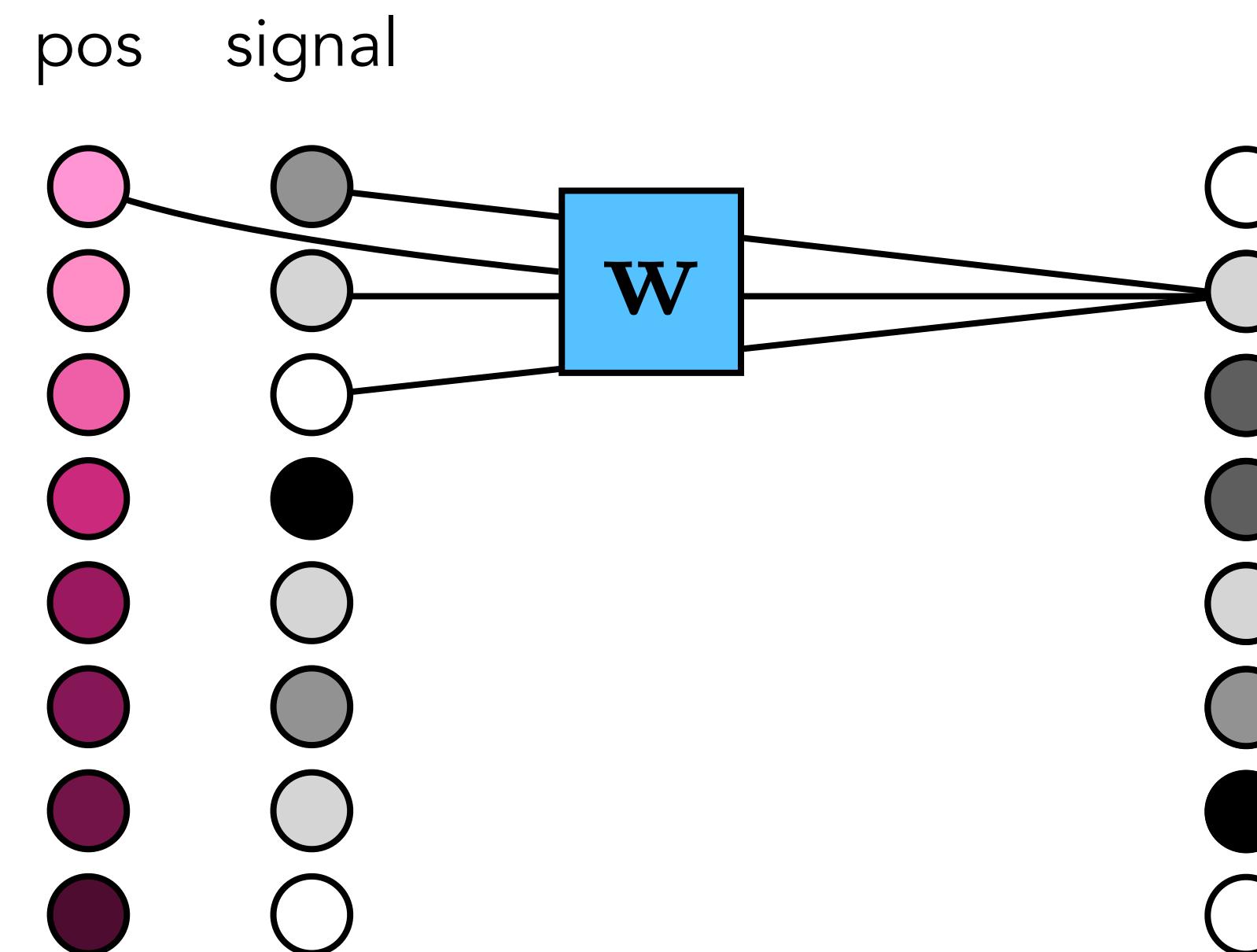
Set2Set



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

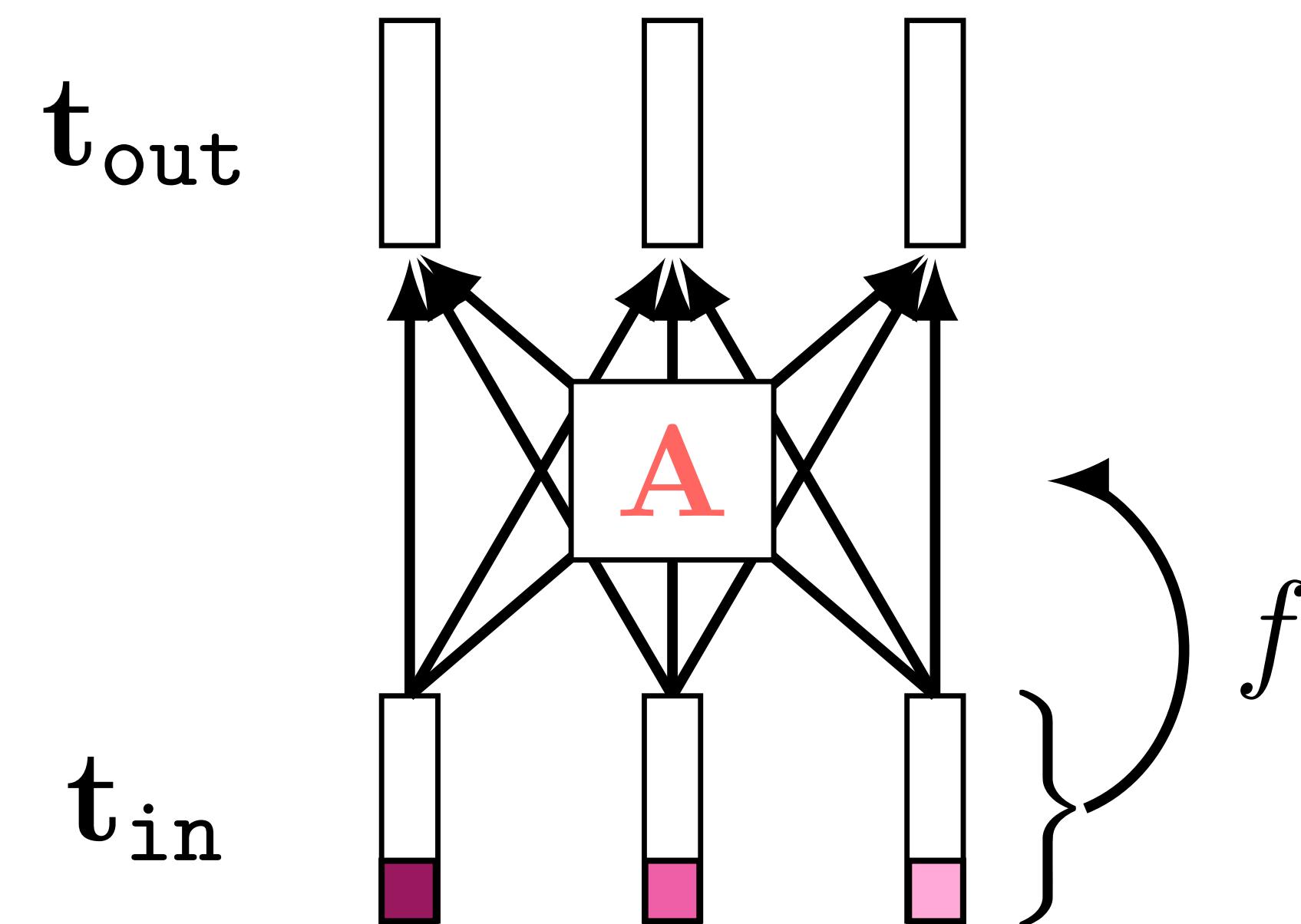
# What if you don't want to be shift invariant?

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**



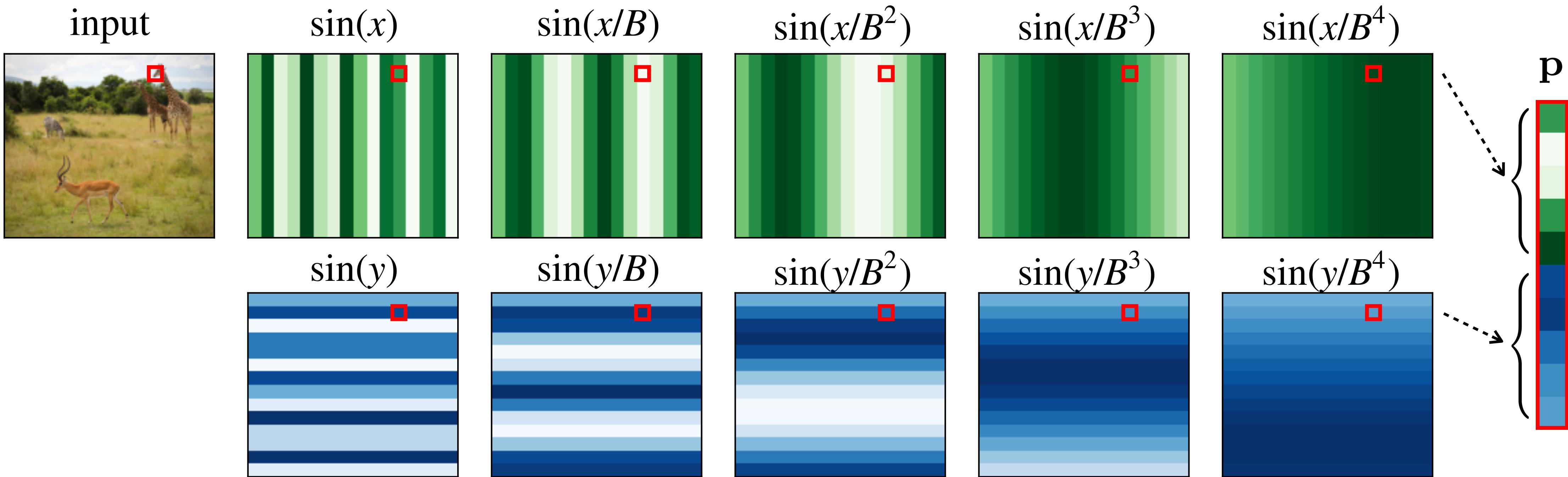
# What if you don't want to be permutation invariant?

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**

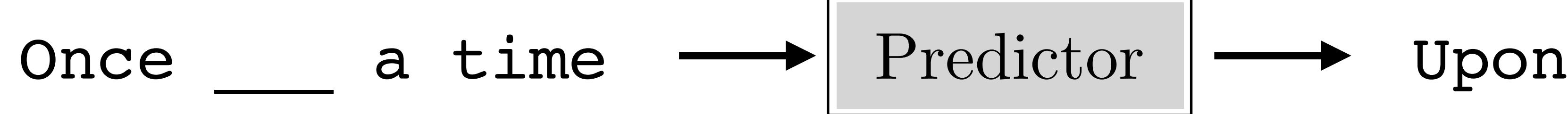
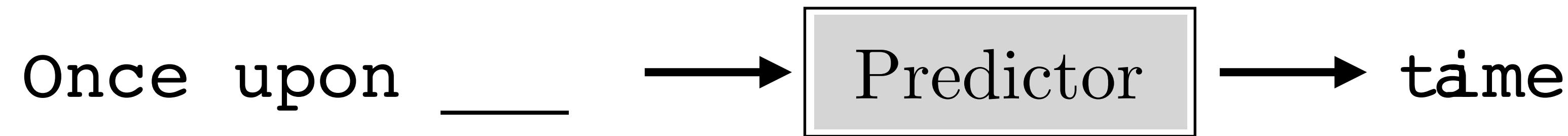


# Fourier positional codes

Represent coordinates on Fourier basis



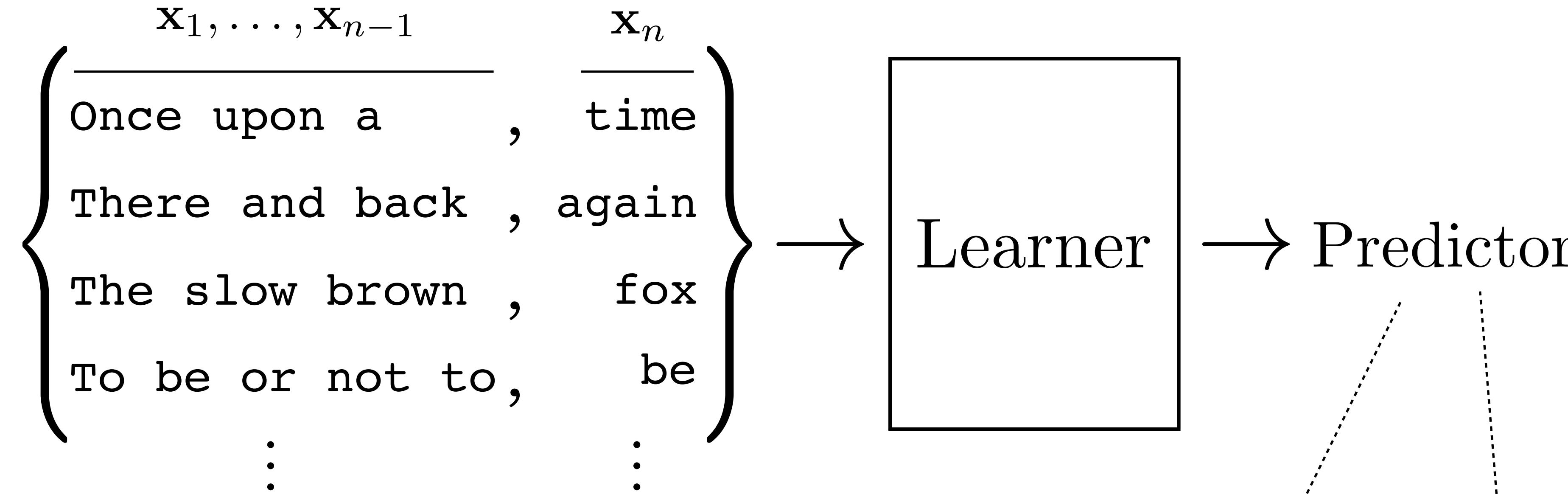
# Autoregressive models



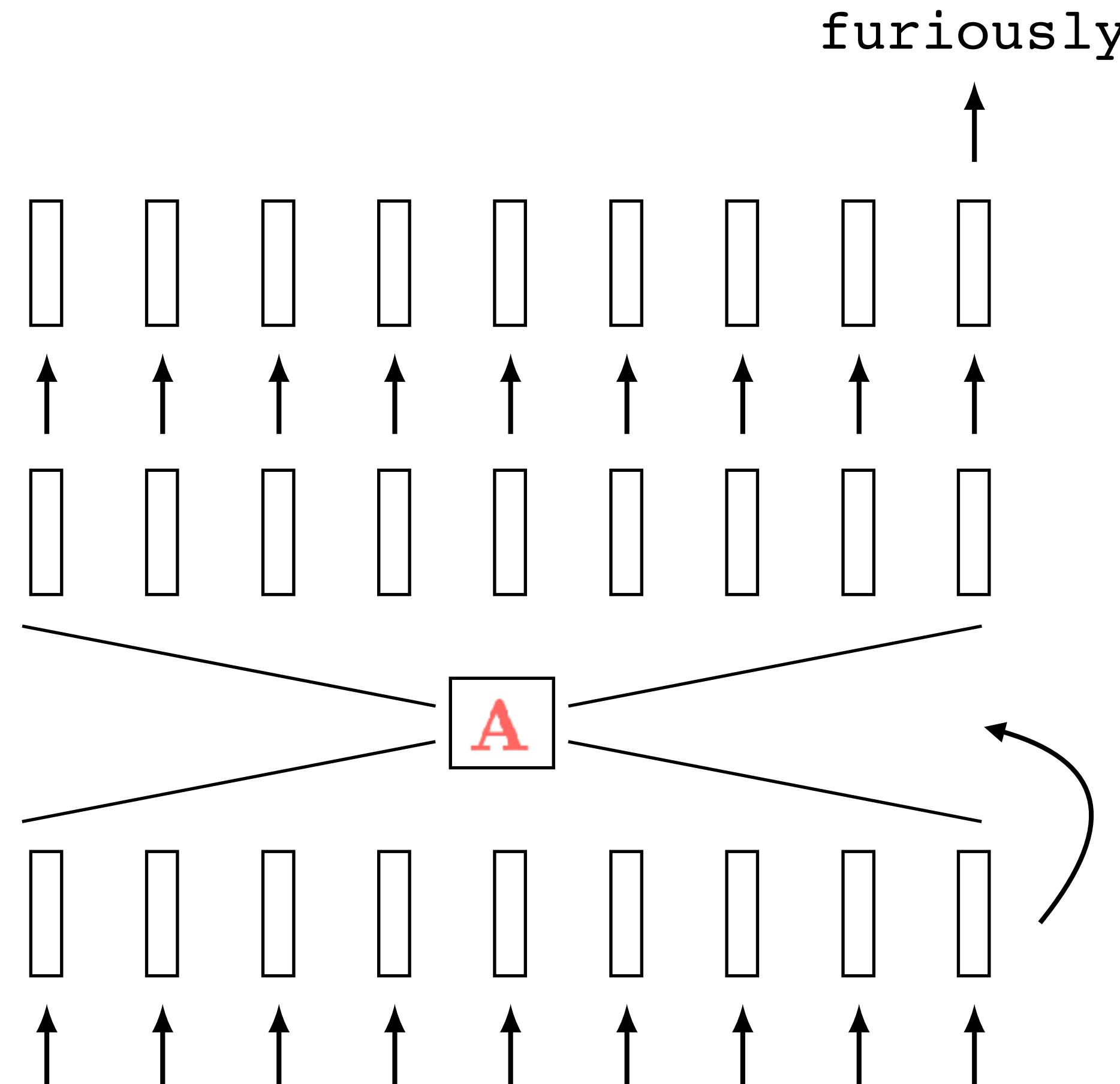
## Sampling

$\underline{x_1, \dots, x_{n-1}}$   
Colorless green ideas sleep → Predictor →  $\hat{x}_n$   
 $\underline{\hspace{10cm}}$   
furiously

## Training



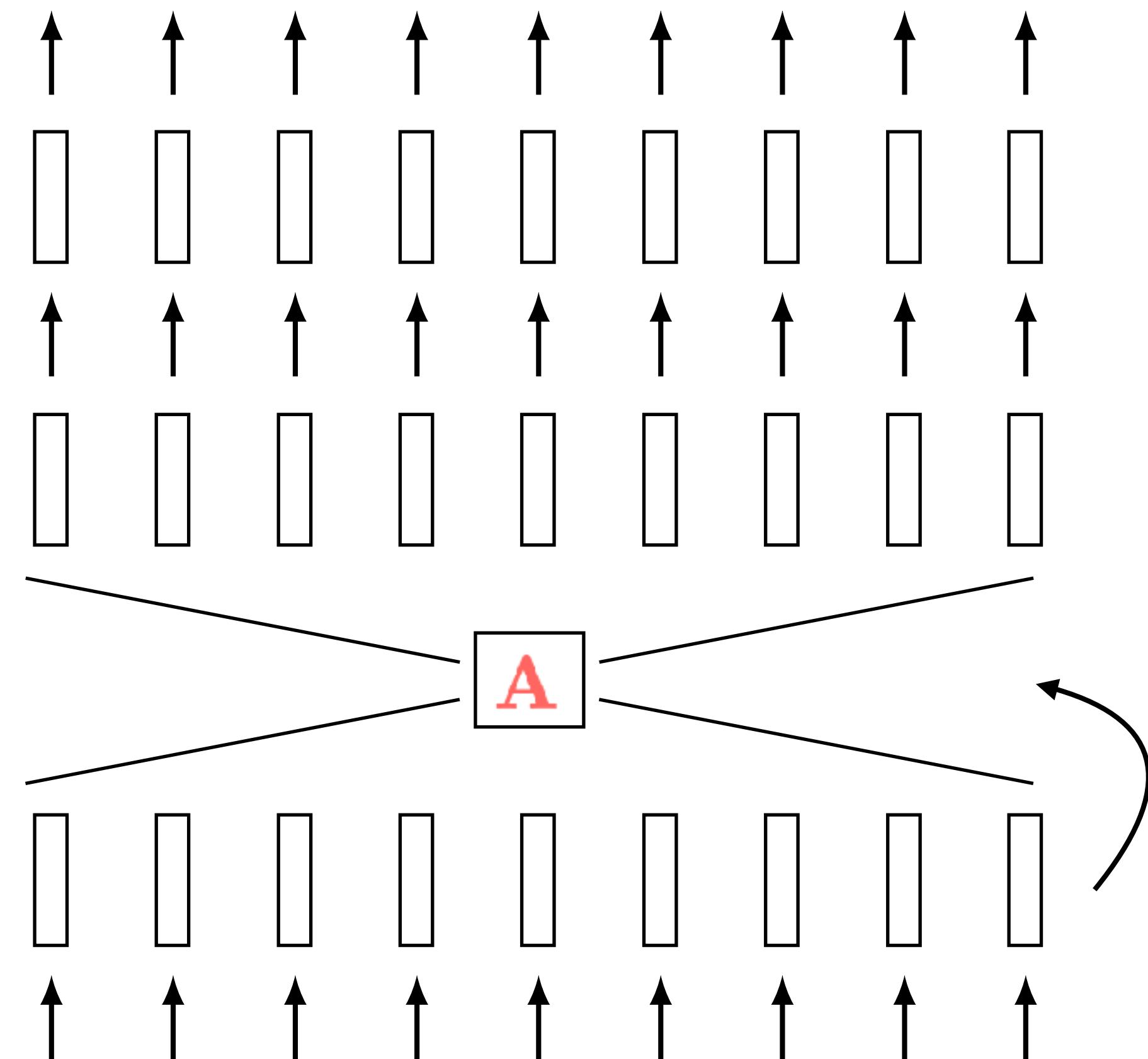
# GPT (and many other related models)



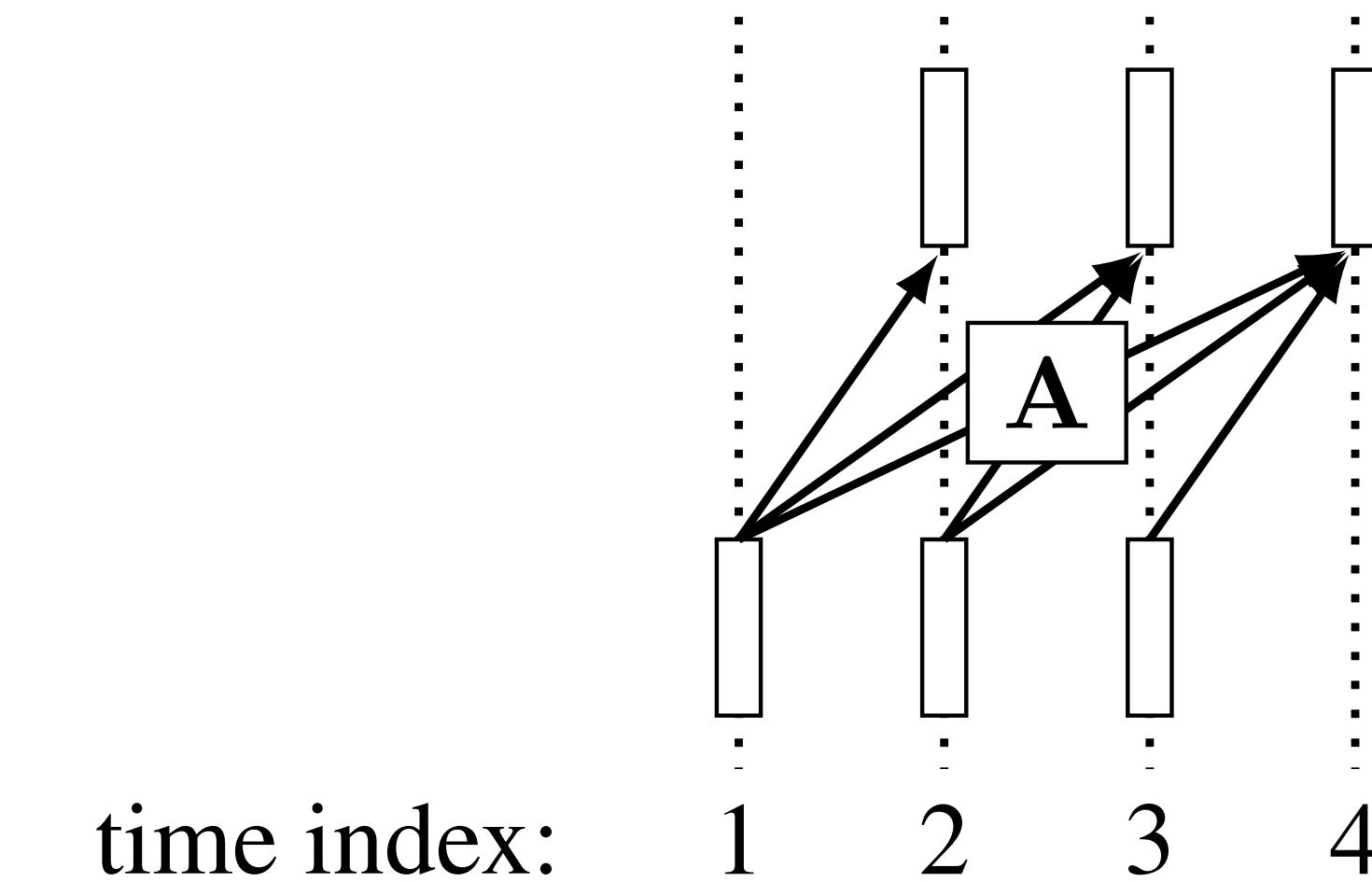
Colorless green ideas sleep \_\_

# GPT training (and many other related models)

Colorless green ideas sleep furiously

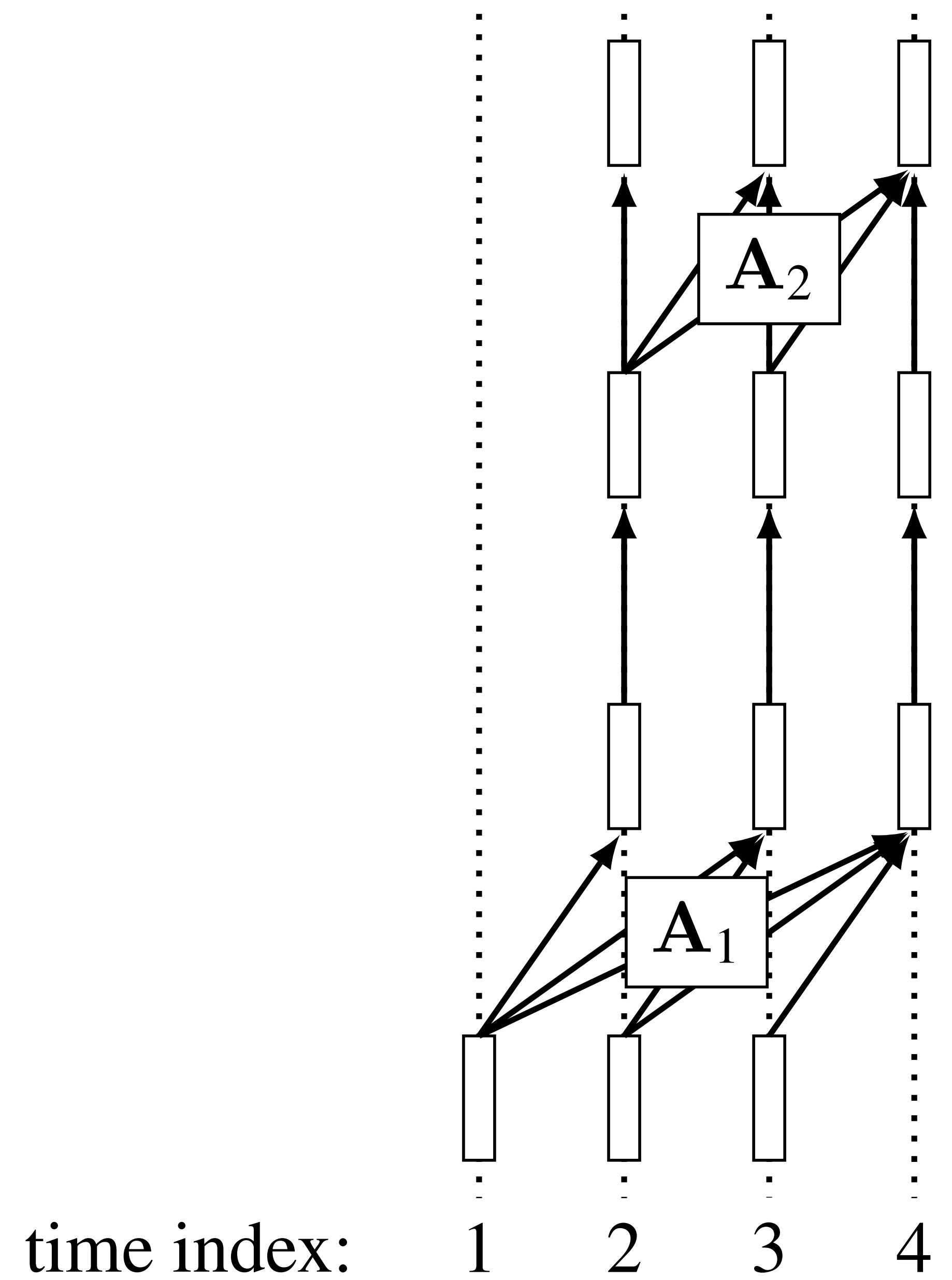


Colorless green ideas sleep furiously



$$A \cdot T_{in} = T_{out}$$

A diagram illustrating the computation of hidden states. On the left, a 4x4 matrix (A) is multiplied by a 4x1 vector (T<sub>in</sub>). The result is a 4x1 vector (T<sub>out</sub>). The first element of T<sub>out</sub> is white, while the others are black. This represents the transformation of the input sequence through the model's layers.



$$A_2 \quad T_{in} \quad T_{out}$$

$$\begin{matrix} & & \\ & & \end{matrix} = \begin{matrix} & & \\ & & \end{matrix}$$

$$A_1 \quad T_{in} \quad T_{out}$$

$$\begin{matrix} & & \\ & & \end{matrix} = \begin{matrix} & & \\ & & \end{matrix}$$

# 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

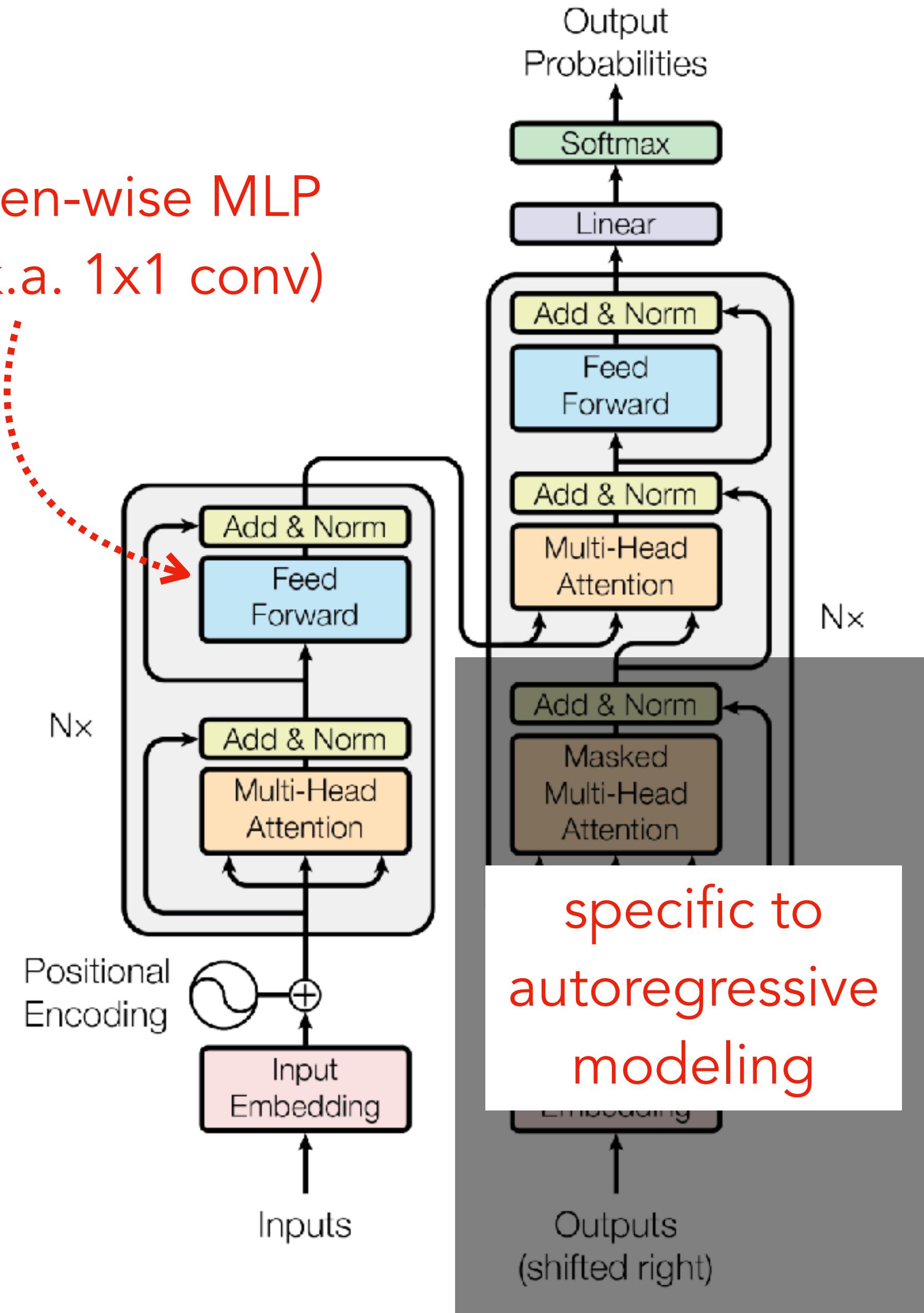
Lukasz Kaiser\*  
Google Brain  
lukaszkaiser@google.com

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.

token-wise MLP  
(a.k.a. 1x1 conv)



# Image-to-text architecture (autoregressive)

