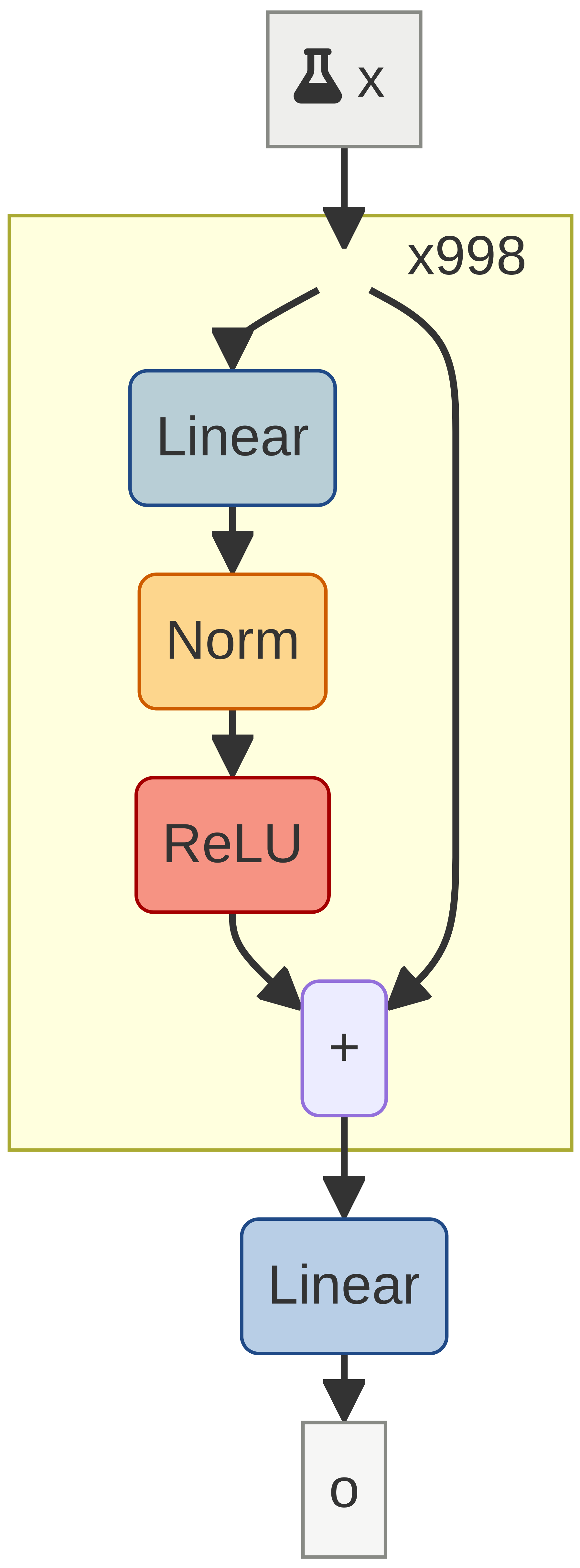
Transformers

Philipp Krähenbühl, UT Austin

Homework discussion

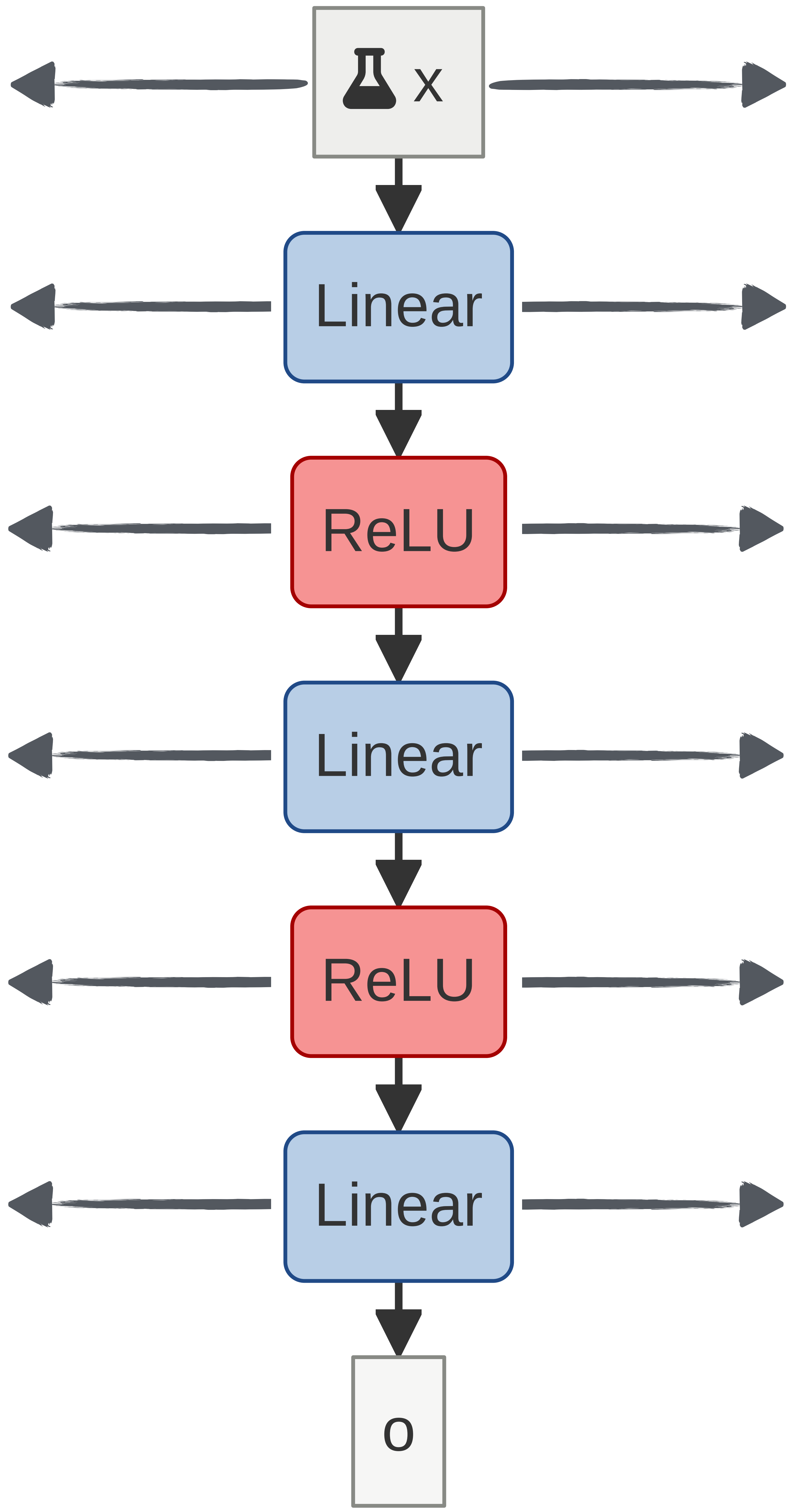
Recap: Scaling Deep

- Vanishing gradients and activations are normal
  - Better than explosions
- Residual connections and normalization deal with vanishing gradients



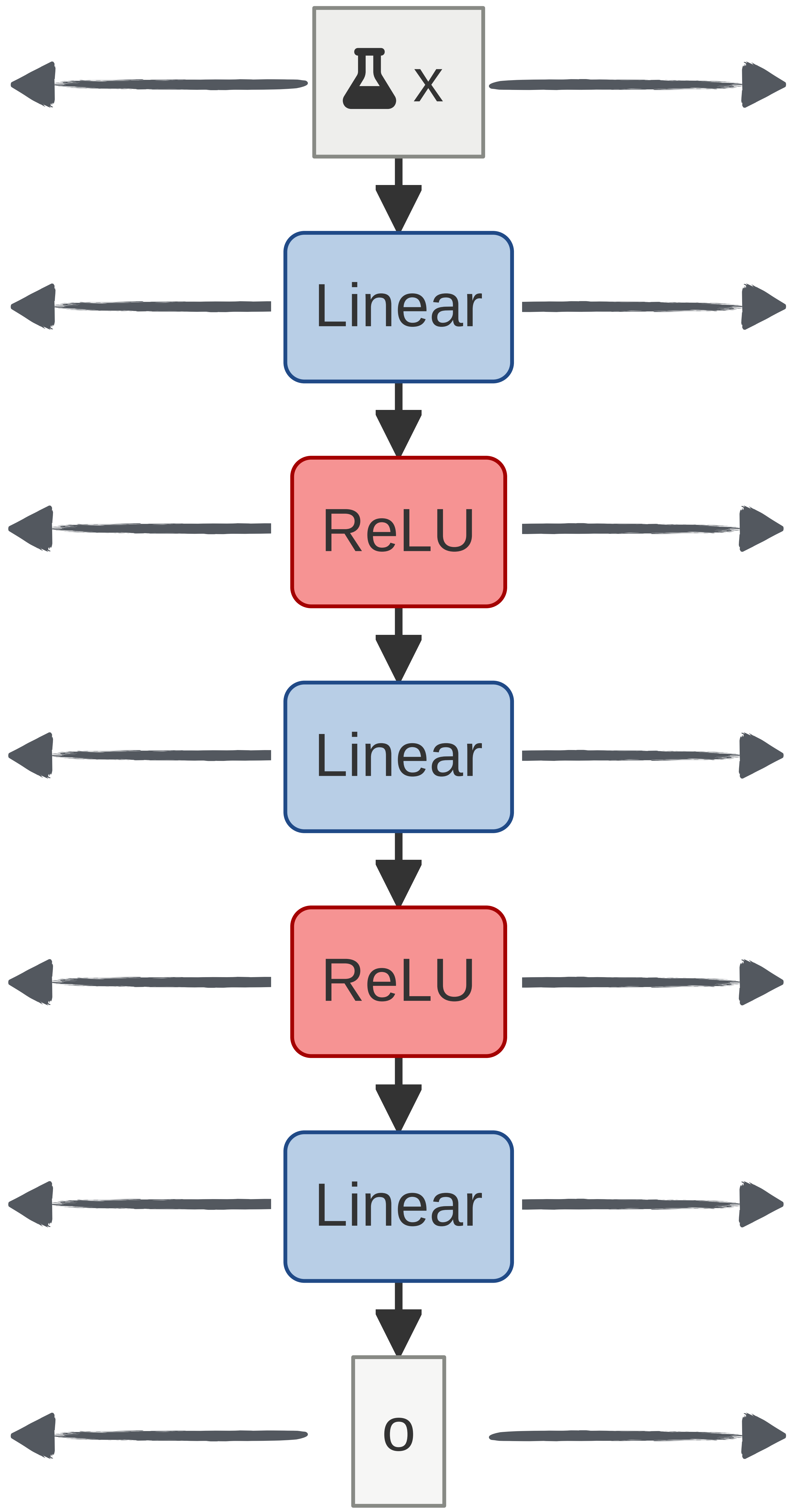
Scaling wide

- Why should we scale wide?
  - (Discussion)



Scaling wide

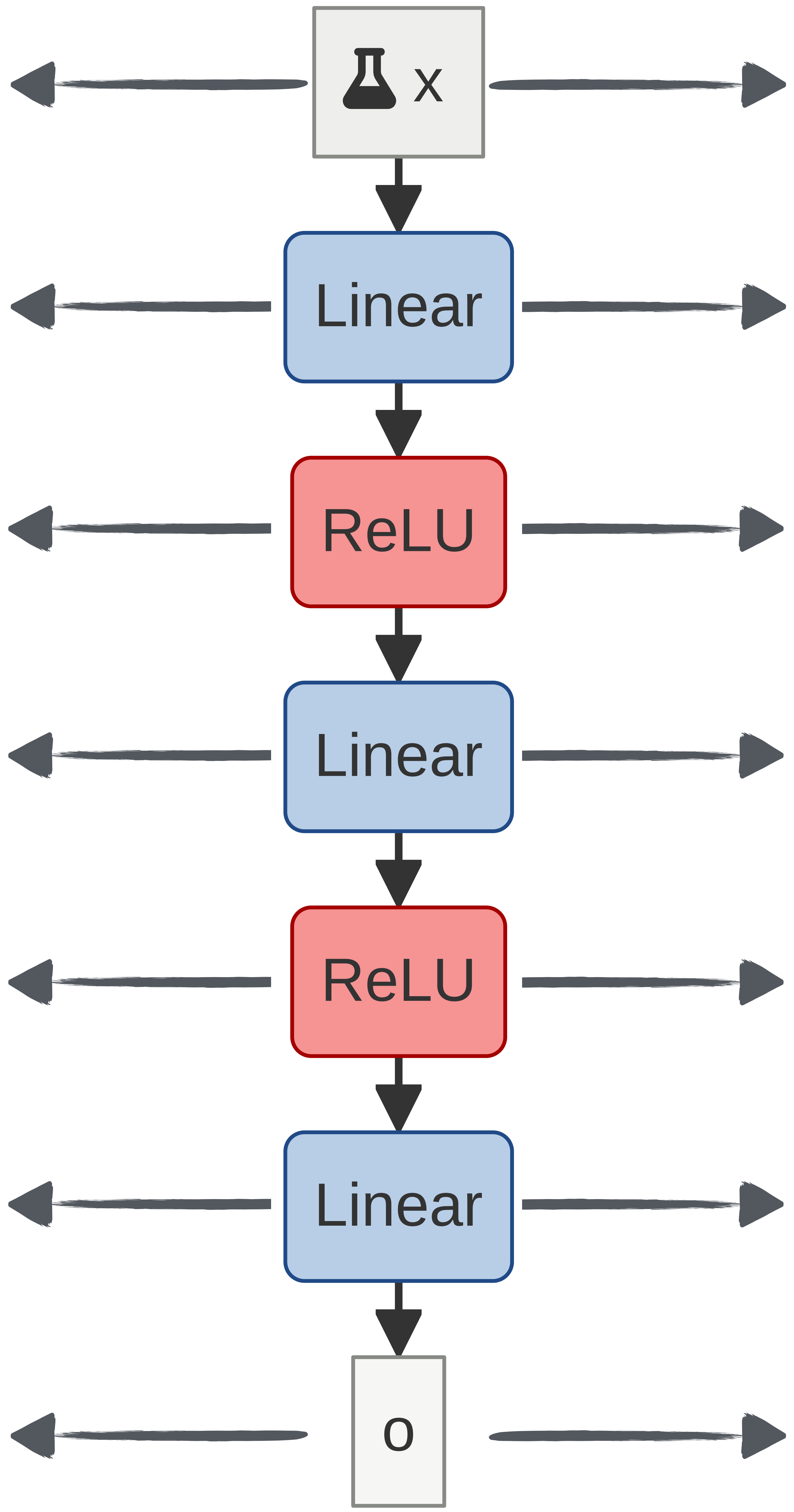
- Why should we scale wide?
  - Larger inputs
  - Larger immediate representations
  - Larger outputs
  - Better performing model



Scaling wide in PyTorch

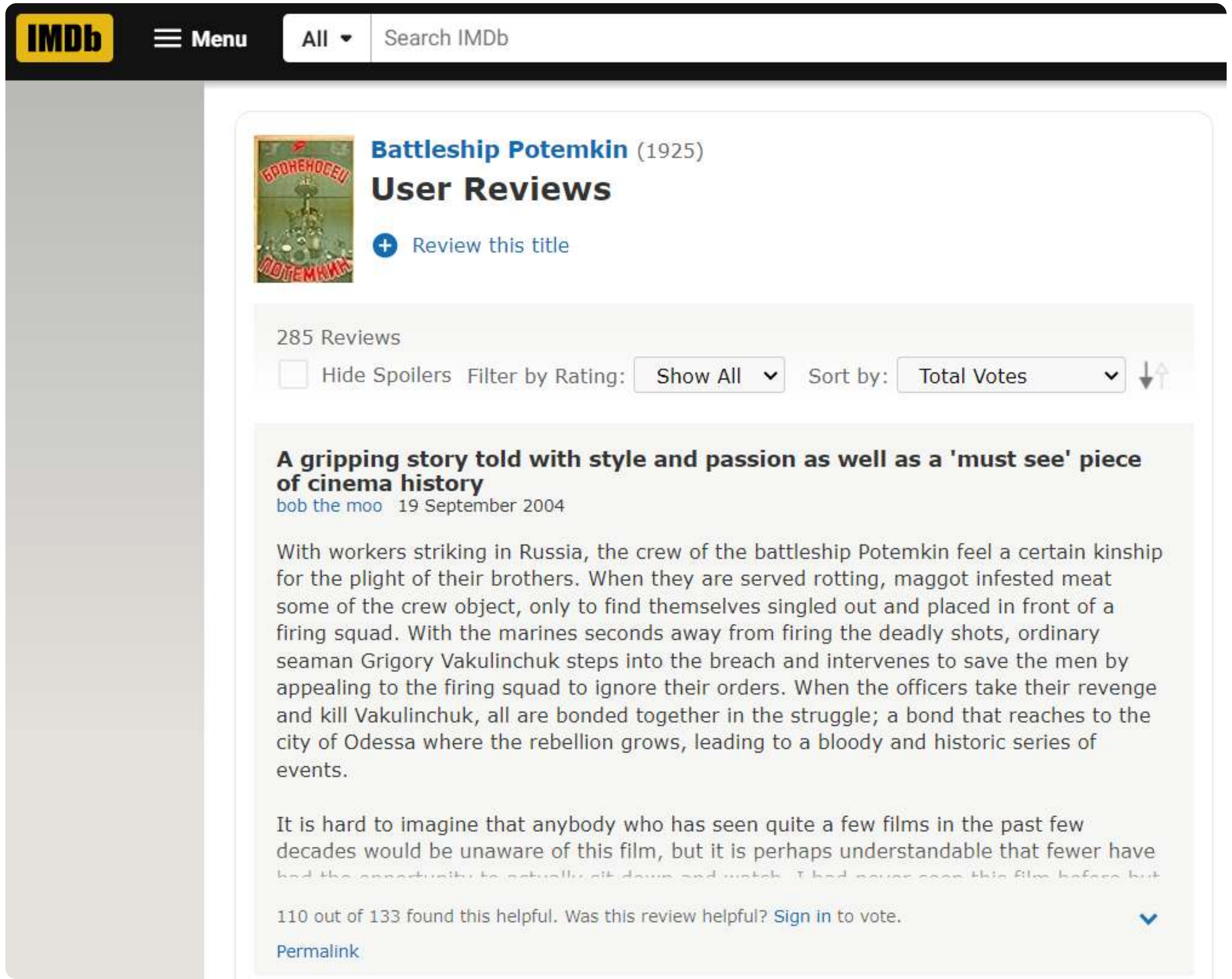
Scaling wide

- Why should we scale wide?
  - Larger inputs
  - Larger immediate representations
  - Larger outputs
  - Better performing model



A New Problem

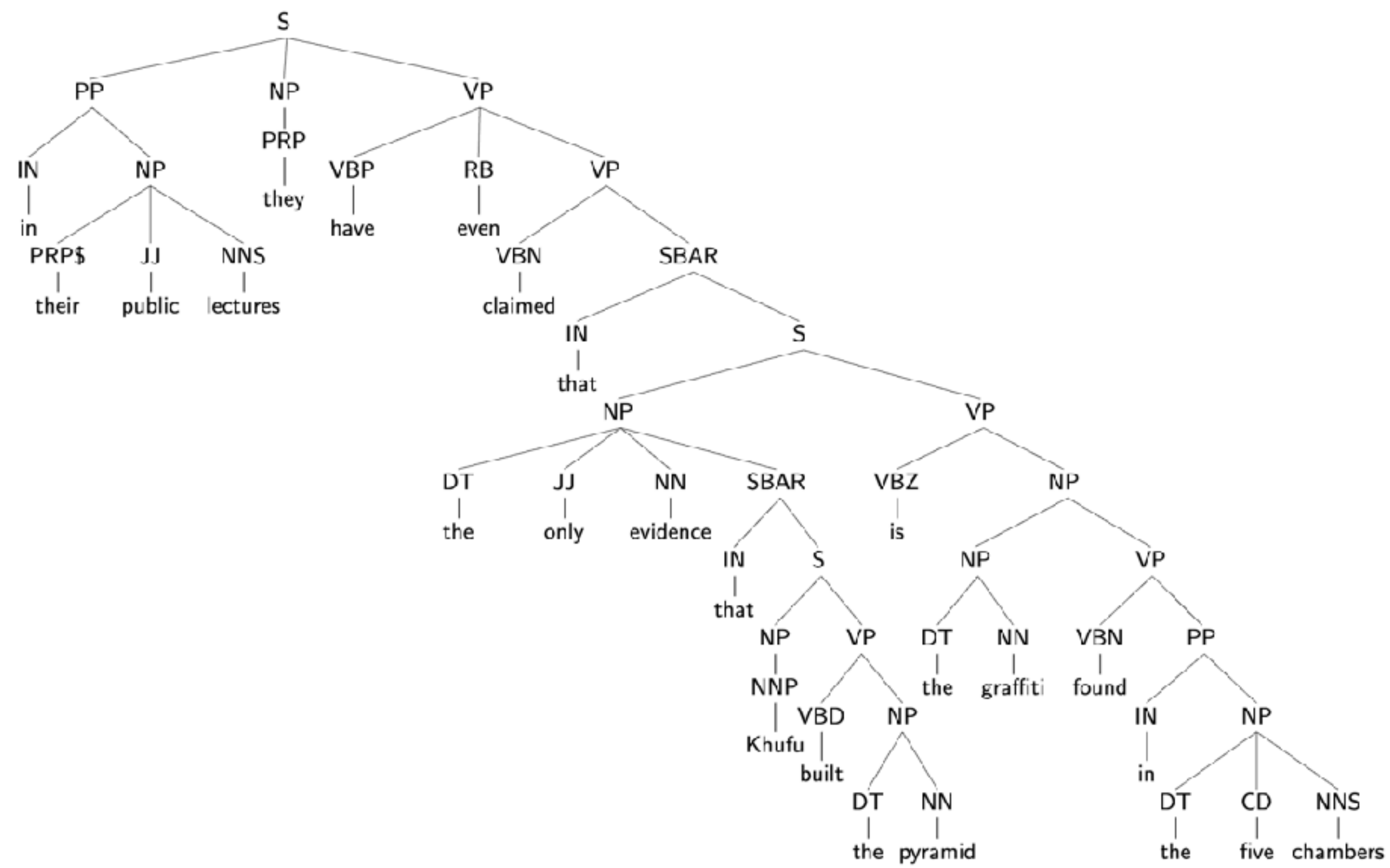
- Sentiment Analysis for Movie Reviews
  - Predict if review is positive 🙂 or negative ☹️
    - 🙂 My kid likes this movie
    - 🙂 My little kid likes this animated movie
    - ☹️ My kid does not like this movie
- Real-world examples: reviews on IMDB [1]



[1] https://www.imdb.com/title/tt0015648/reviews/?sort=totalVotes

How are language tasks different?

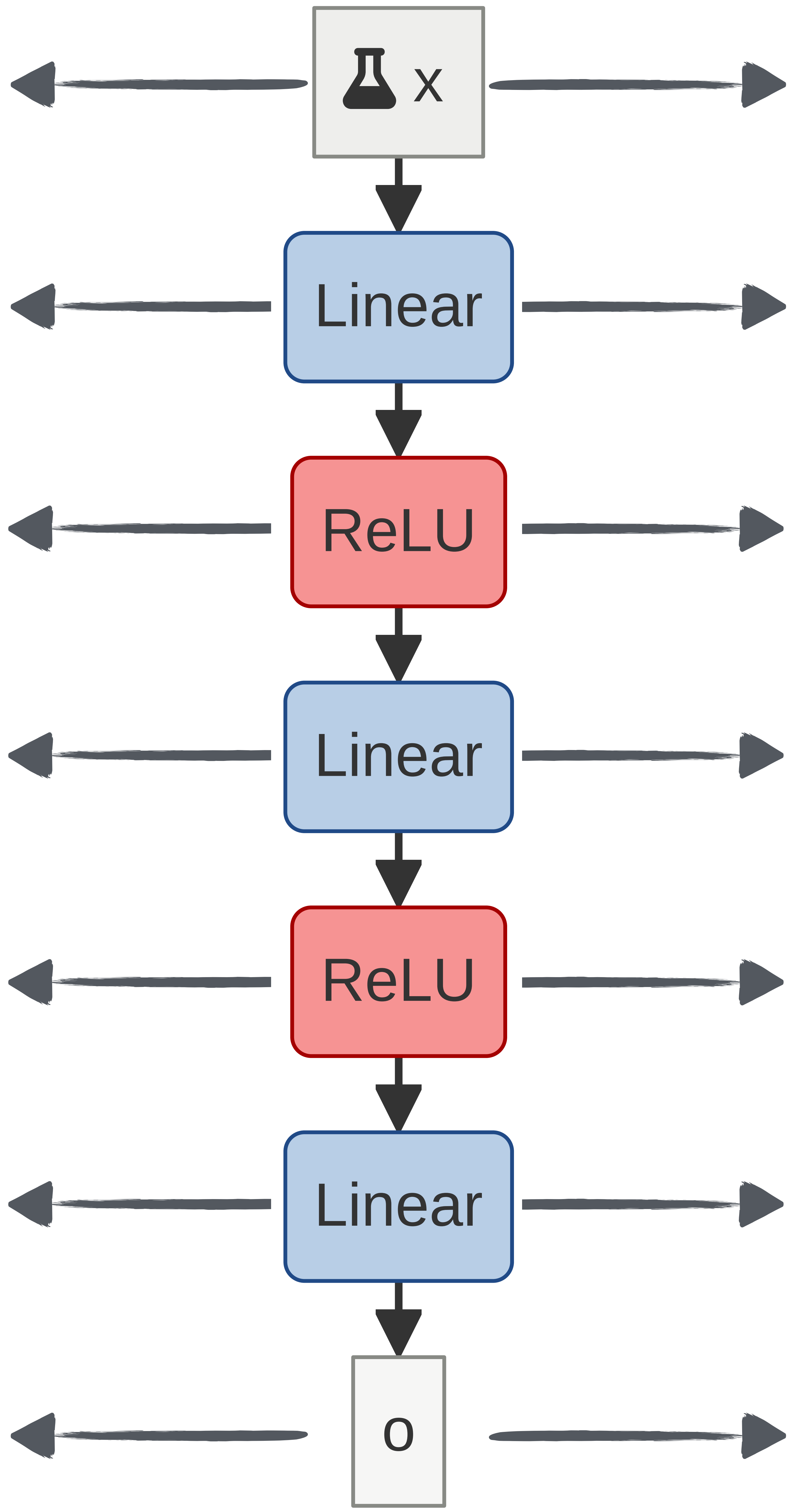
- Images
  - Fixed input (resolution)
  - Fixed structure
- Language
  - Variable length
  - Diverse structure (tree syntax) [1]



https://www.cs.cornell.edu/courses/cs474/2004fa/lec1.pdf

Scaling wide

- Why should we scale wide?
  - Larger inputs
  - Larger immediate representations
  - Larger outputs
  - Better performing model
  - Variable size inputs and outputs

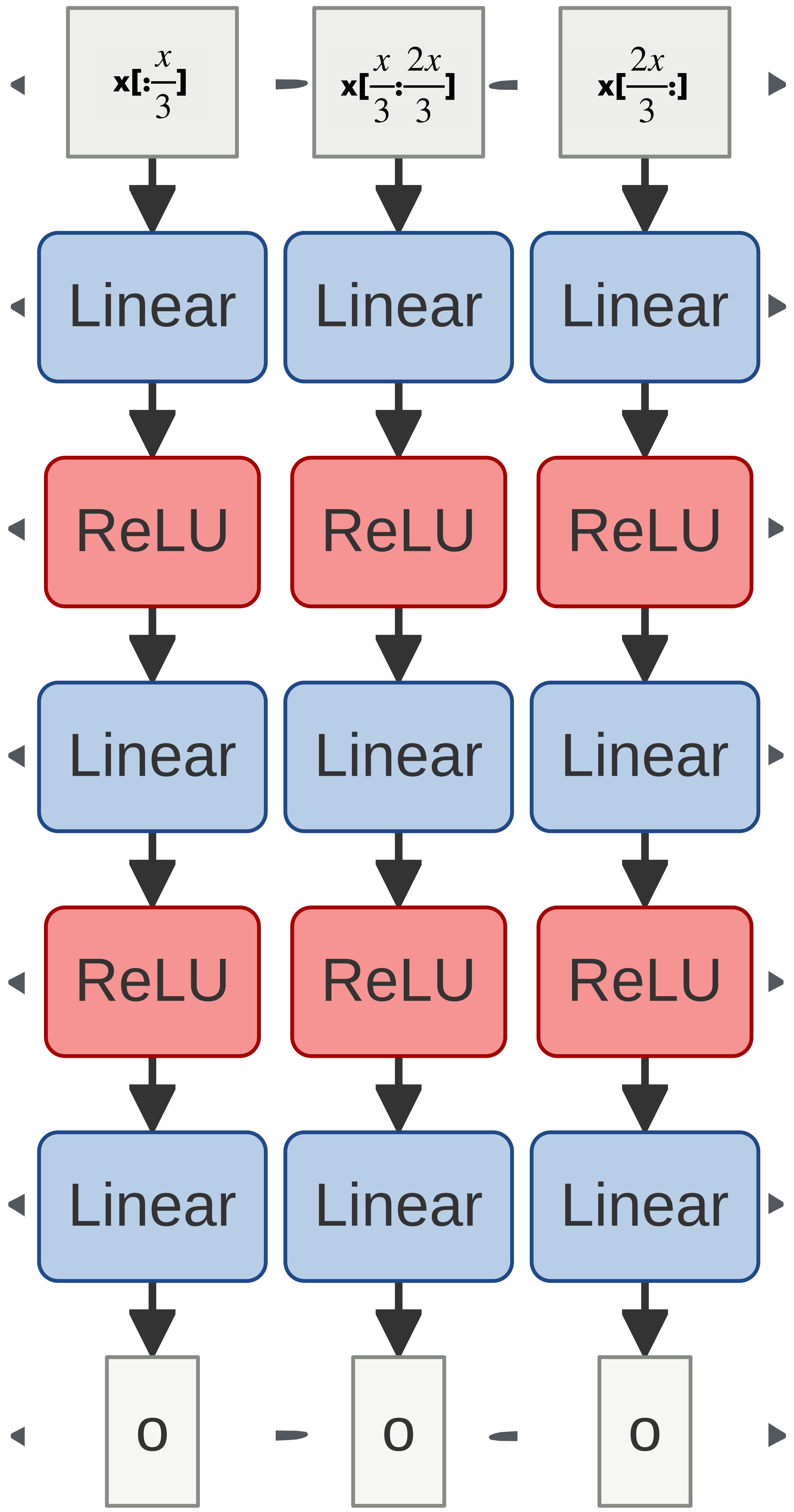


Scaling wide: A Simple Solution

Scaling wide

A Simple Solution

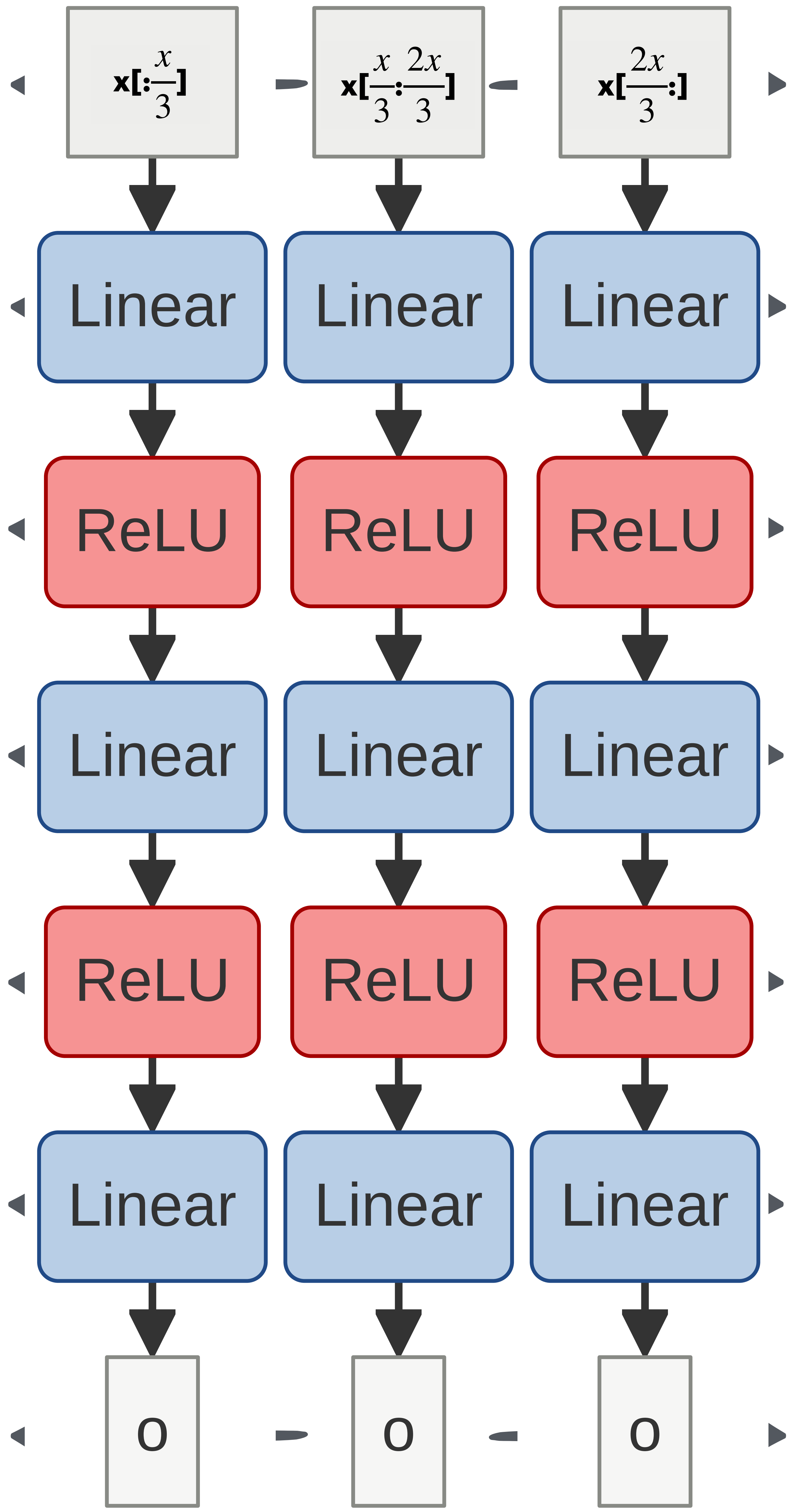
- Split input into k parts
- Run a identical smaller networks on each part



Scaling wide

A Simple Solution

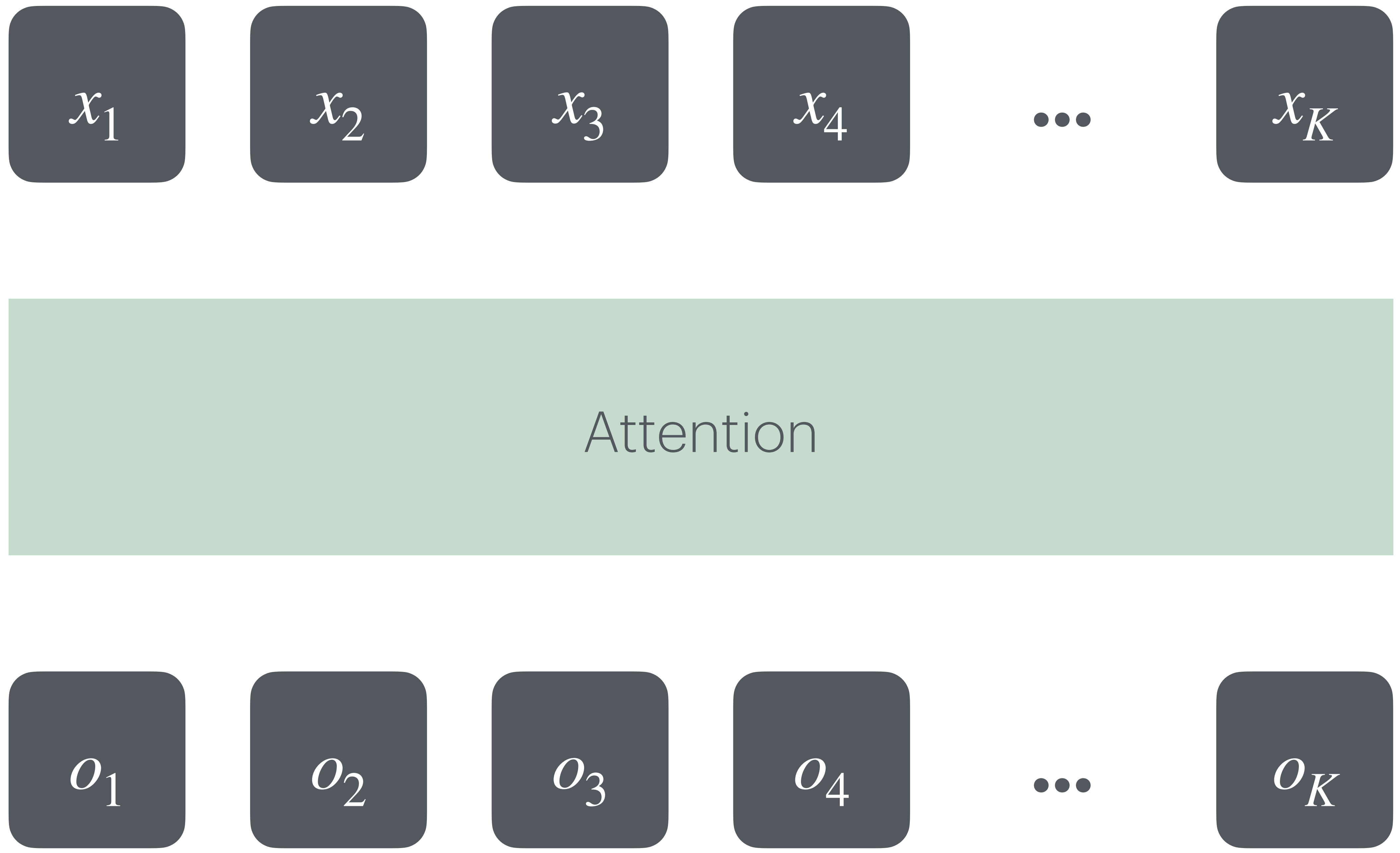
- Advantages (splitting in k parts)
  - k^2 times fewer parameters
  - k times faster
  - Good at processing local information
- Disadvantage
  - Different parts do not communicate



Attention

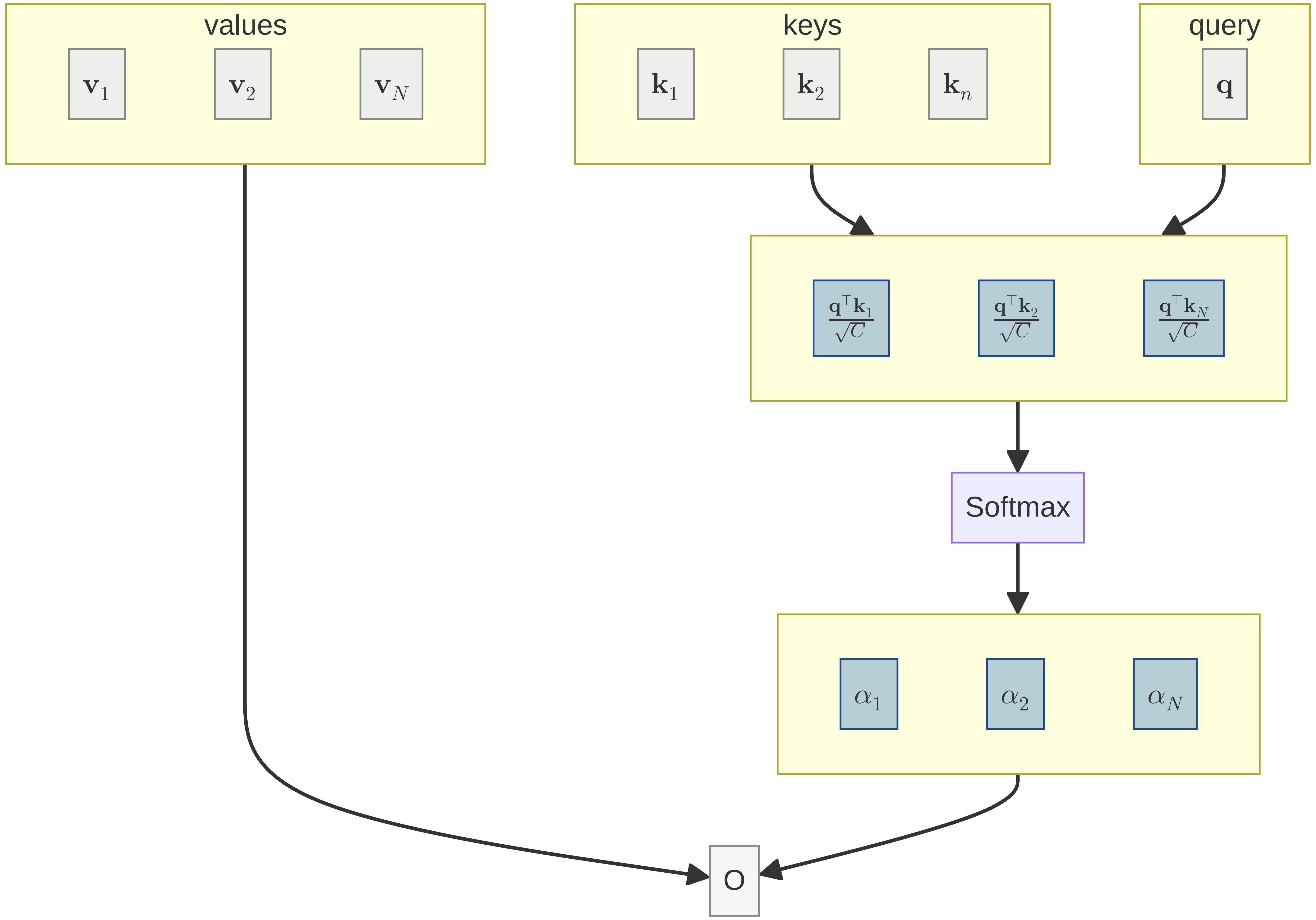
Attention

- A **set operator** that learns to reason about the structure of a set of elements



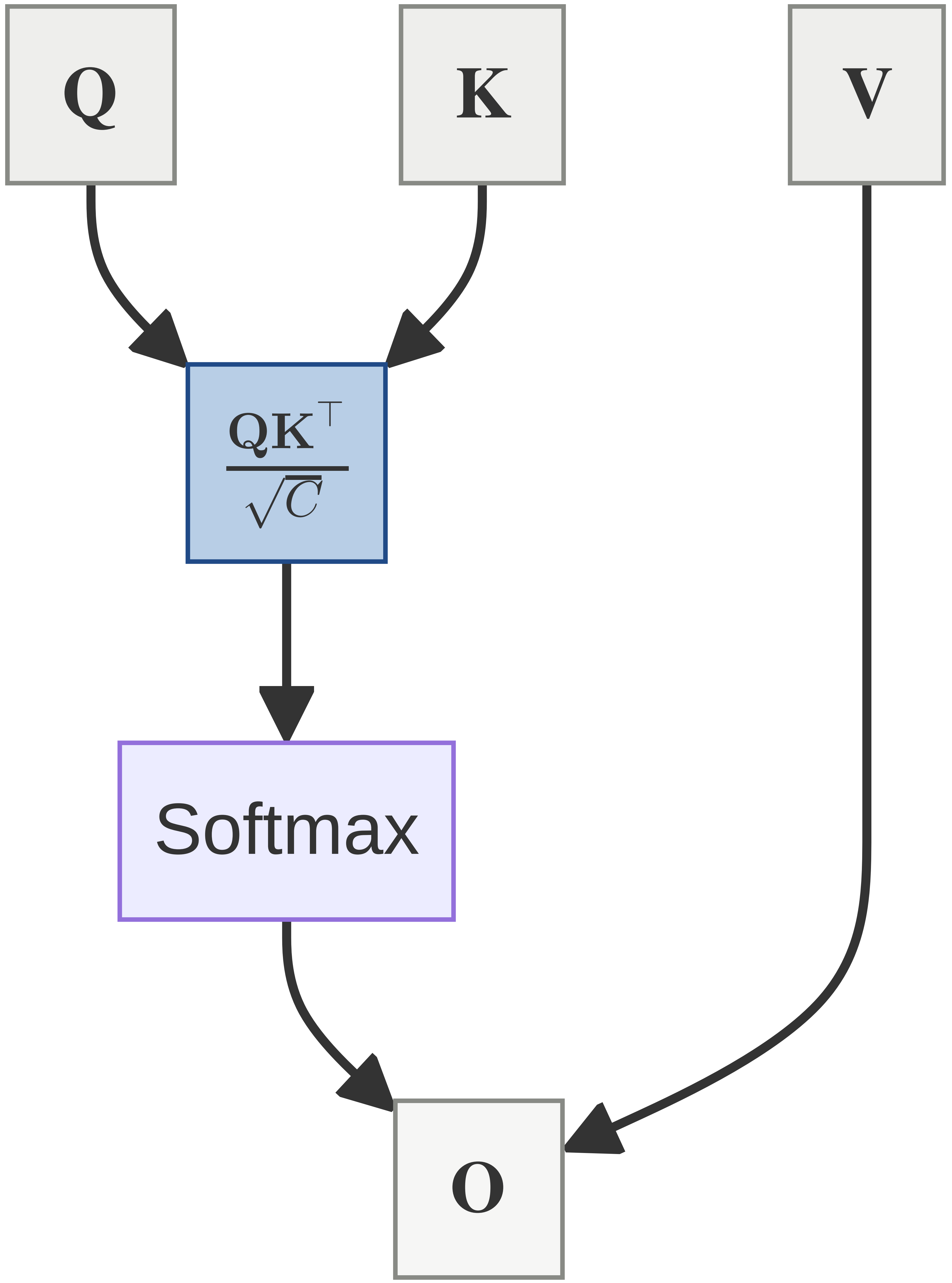
Attention

- **Inputs:**
  - query $\mathbf{q} \in \mathbb{R}^C$
  - a set of keys $\mathbf{K} = [\mathbf{k}_1, \cdots, \mathbf{k}_N], \mathbf{k}_i \in \mathbb{R}^C$
  - a set of values $\mathbf{V} = [\mathbf{v}_1, \cdots, \mathbf{v}_N], \mathbf{v}_i \in \mathbb{R}^C$
- **Output:** $\mathbf{o} \in \mathbb{R}^C$
  - $\mathbf{o} = \sum_i \alpha_i \mathbf{v}_i, \quad \text{where } \alpha_i = \frac{e^{\frac{\mathbf{q}^\top \mathbf{k}_i}{\sqrt{C}}}}{\sum_j e^{\frac{\mathbf{q}^\top \mathbf{k}_j}{\sqrt{C}}}}$



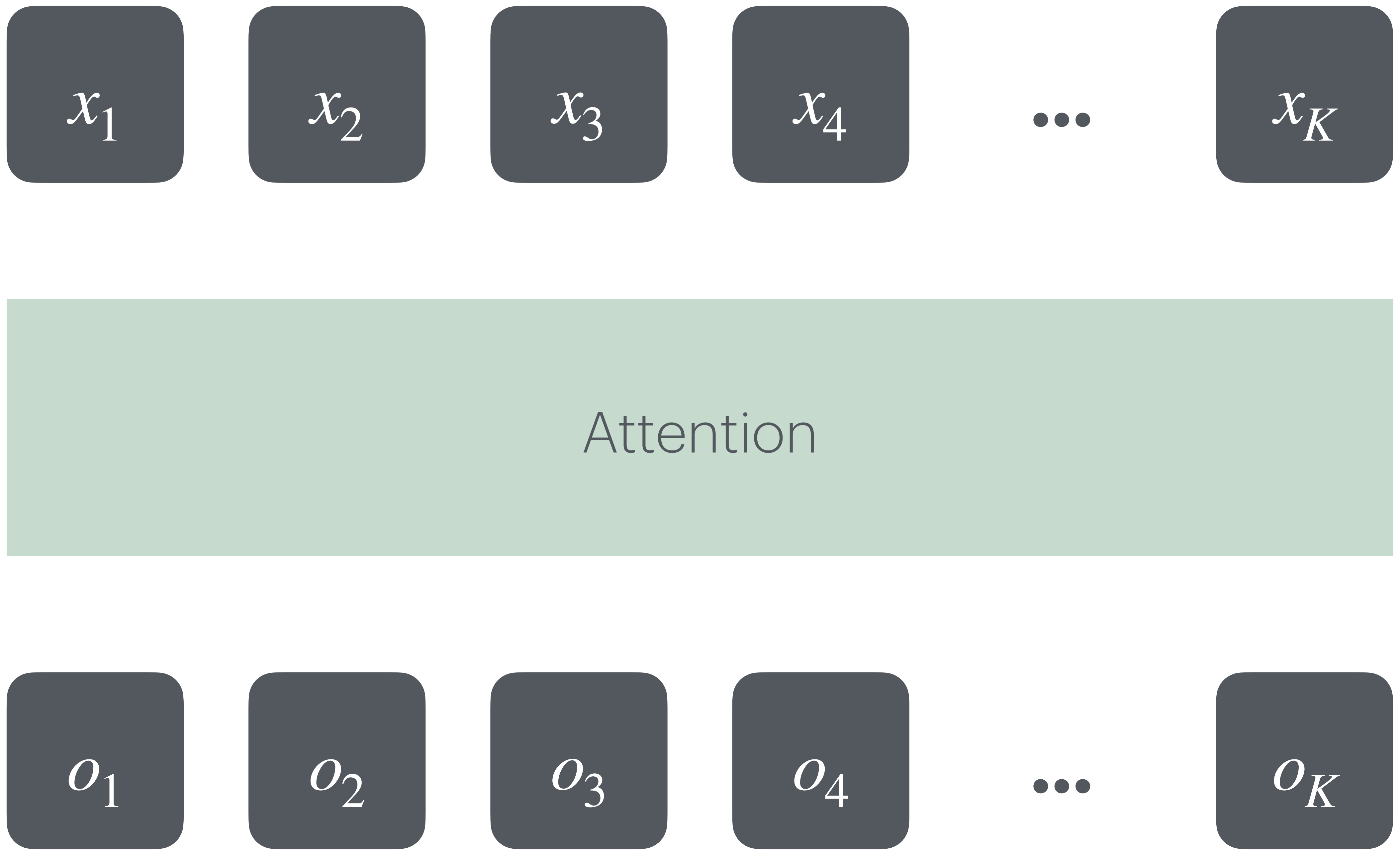
Attention: Matrix Form

- **Inputs:**
  - queries $\mathbf{Q} \in \mathbb{R}^{M \times C}$
  - keys $\mathbf{K} \in \mathbb{R}^{N \times C}$
  - values $\mathbf{V} \in \mathbb{R}^{N \times C}$
- **Output:** $\mathbf{O} \in \mathbb{R}^{M \times C}$
  - $\mathbf{O} = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{C}}\right)\mathbf{V}$
- $\text{softmax}(\cdot)$ is row-wise (each row sums to 1)



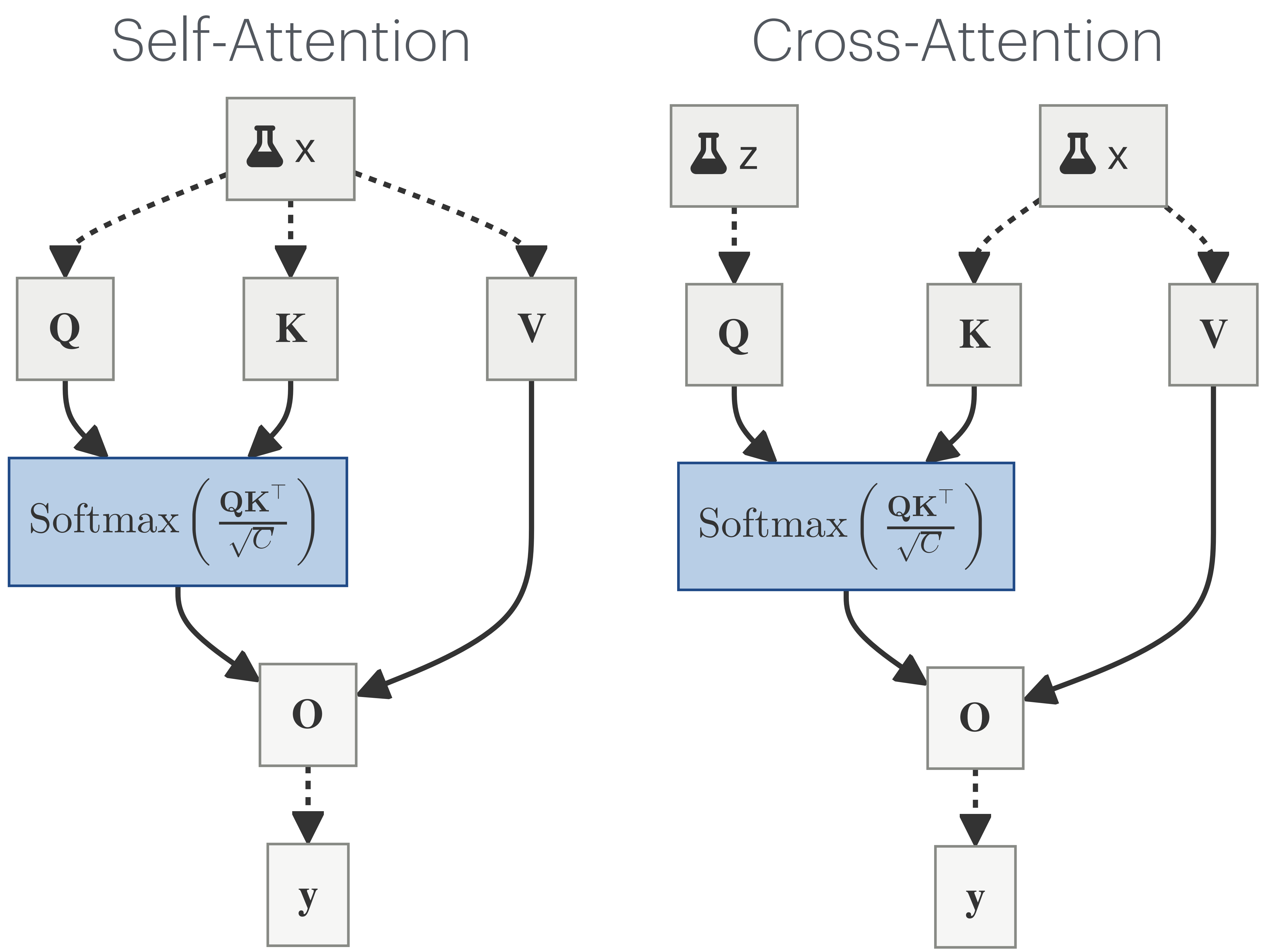
Attention

- A **set operator** that learns to reason about the structure of a set of elements
- Set 1
  - keys $\mathbf{K} \in \mathbb{R}^{N \times C}$
  - values $\mathbf{V} \in \mathbb{R}^{N \times C}$
- Set 2
  - queries $\mathbf{Q} \in \mathbb{R}^{M \times C}$
  - outputs $\mathbf{O} \in \mathbb{R}^{M \times C}$



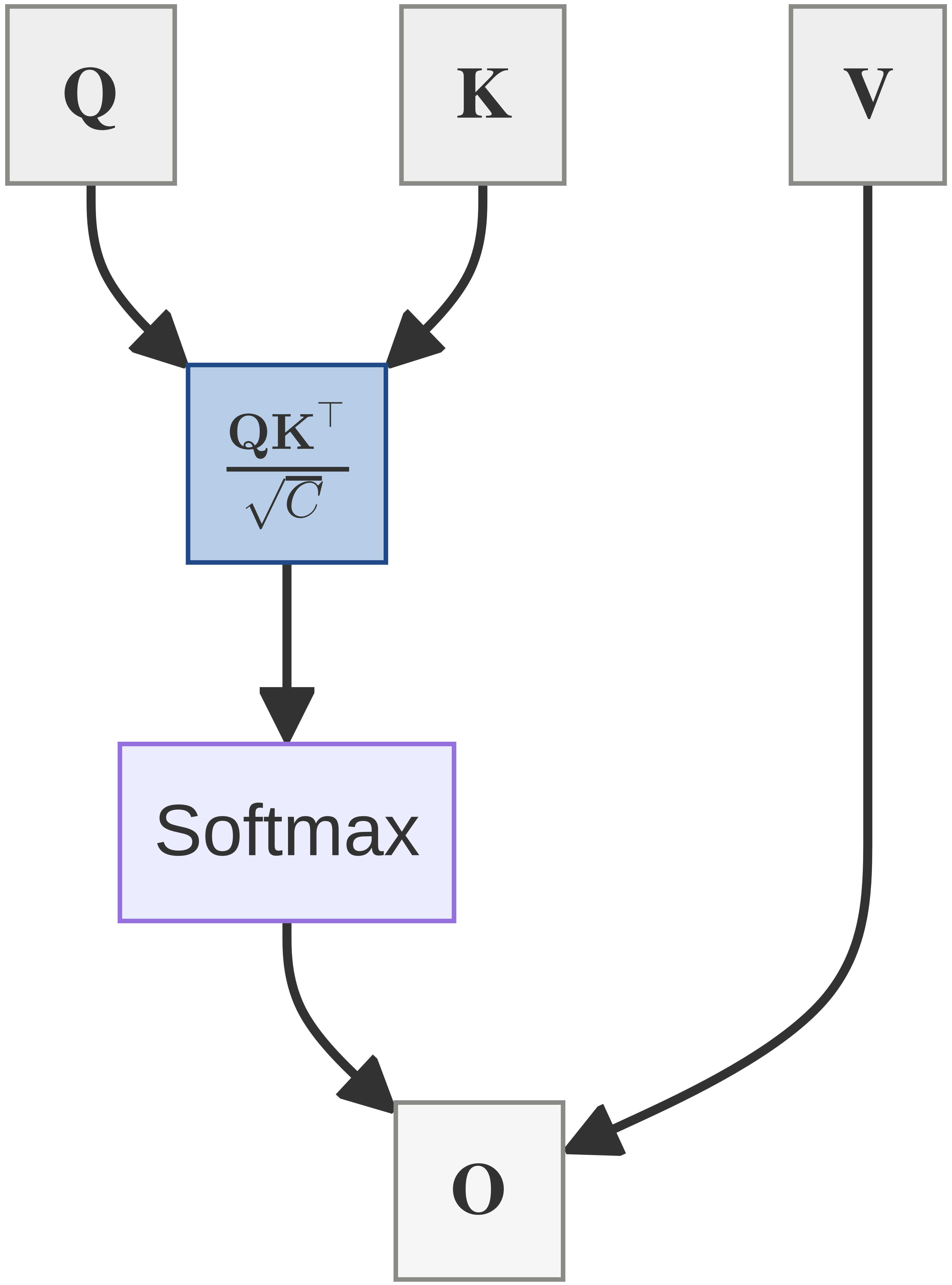
Self-Attention and Cross-Attention

- Self-Attention
  - queries, keys & values from the **same inputs**
- Cross-Attention
  - keys and values come from **one set of inputs**
  - queries come from **another set** of inputs



Self-Attention

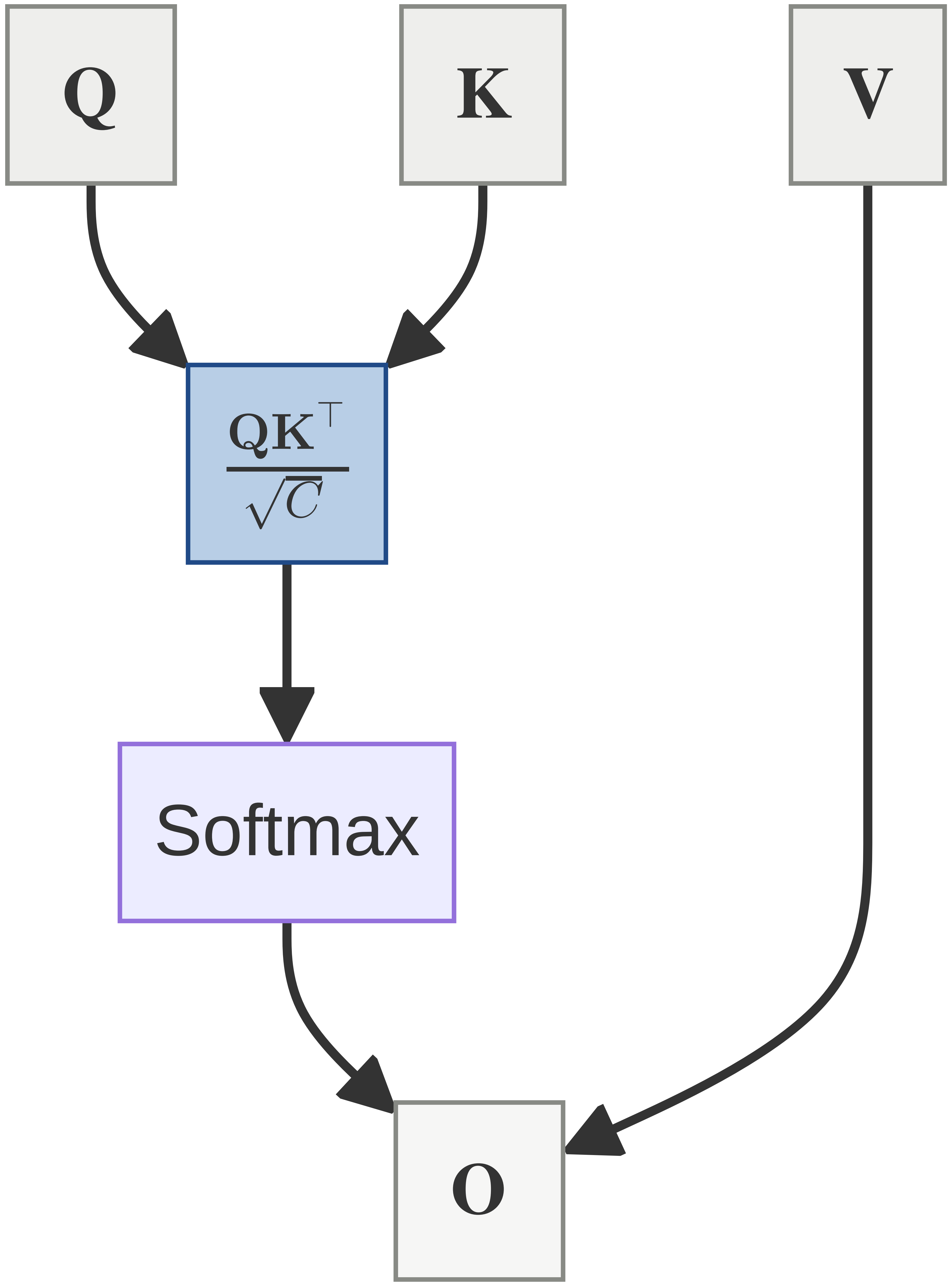
- **Attention:**
  - $\mathbf{O} = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{C}}\right)\mathbf{V}$
- **Self-Attention:** $\mathbf{O} \in \mathbb{R}^{M \times C}$
  - $\mathbf{O} = \text{Attention}(\mathbf{X}, \mathbf{X}, \mathbf{X}) = \text{softmax}\left(\frac{\mathbf{X}\mathbf{X}^\top}{\sqrt{C}}\right)\mathbf{X}$



Issues With Self-Attention

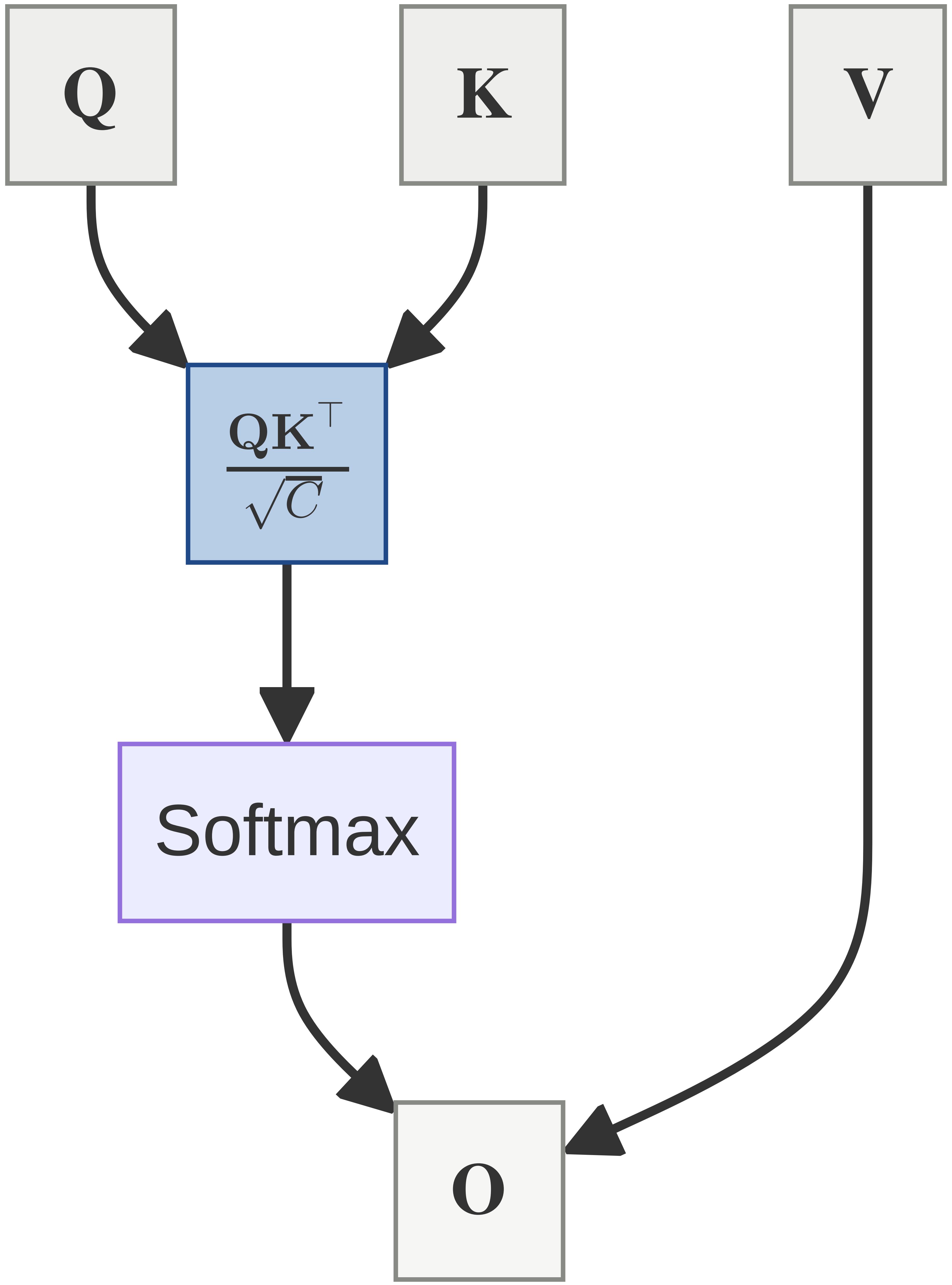
- $\mathbf{O} = \text{Attention}(\mathbf{X}, \mathbf{X}, \mathbf{X}) = \text{softmax}\left(\frac{\mathbf{X}\mathbf{X}^\top}{\sqrt{C}}\right)\mathbf{X}$
- For any $(\mathbf{x}_i, \mathbf{x}_j)$ pair where $i \neq j$,
- $$\begin{aligned} \mathbf{x}_i\mathbf{x}_j^\top &\leq \frac{1}{2}(\mathbf{x}_i\mathbf{x}_i^\top + \mathbf{x}_j\mathbf{x}_j^\top) \\ &\leq \max(\mathbf{x}_i\mathbf{x}_i^\top, \mathbf{x}_j\mathbf{x}_j^\top) \\ \alpha_{i,j} &\leq \max(\alpha_{i,i}, \alpha_{j,j}) \quad \text{(softmax preserves order)} \end{aligned}$$

  for $\boldsymbol{\alpha} = \text{softmax}\left(\frac{\mathbf{X}\mathbf{X}^\top}{\sqrt{C}}\right)$



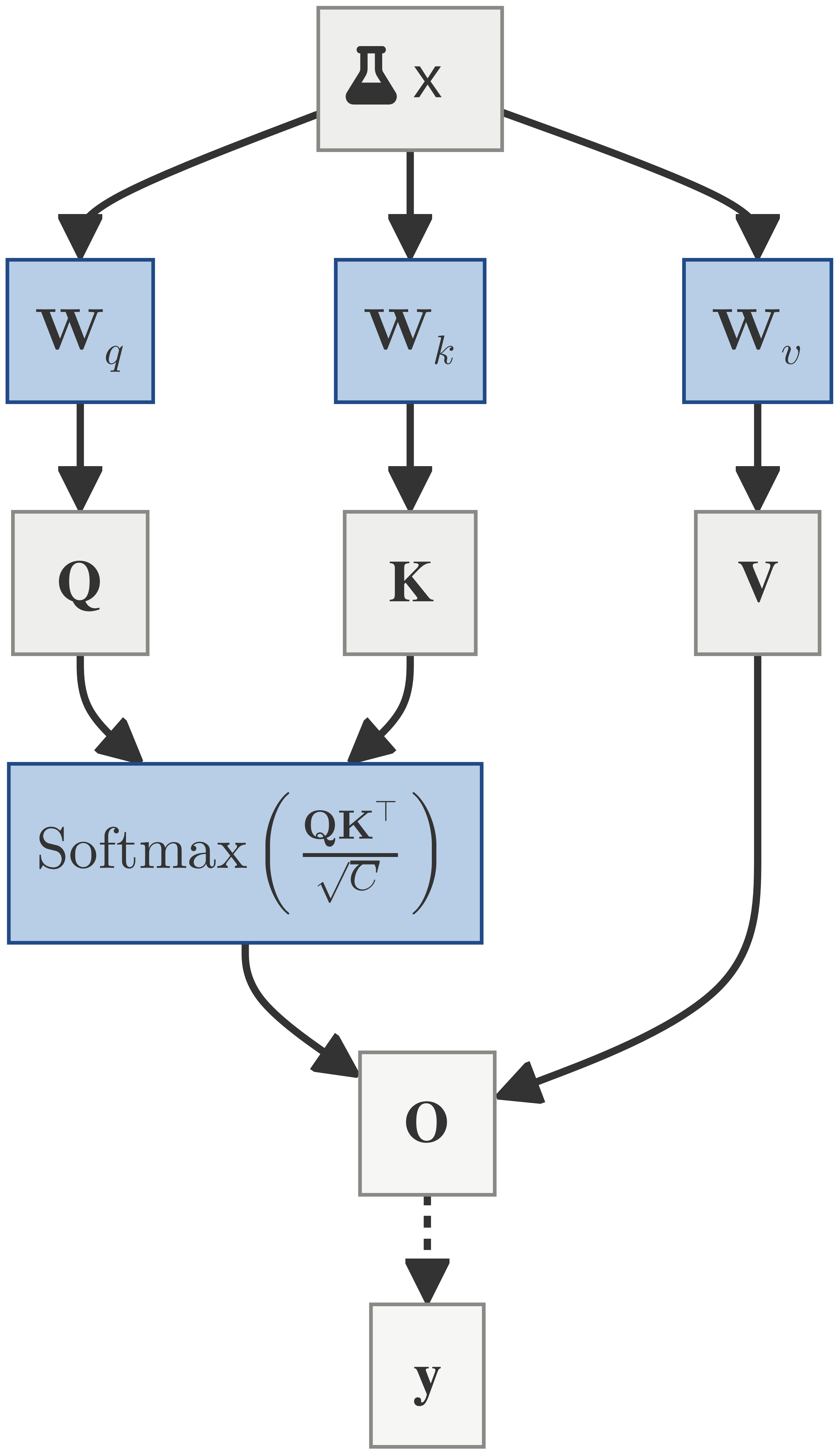
Issues With Self-Attention

- $\mathbf{O} = \text{Attention}(\mathbf{X}, \mathbf{X}, \mathbf{X}) = \text{softmax}\left(\frac{\mathbf{X}\mathbf{X}^\top}{\sqrt{C}}\right)\mathbf{X}$
- $\alpha_{i,j} \leq \max(\alpha_{i,i}, \alpha_{j,j})$ for $\boldsymbol{\alpha} = \text{softmax}\left(\frac{\mathbf{X}\mathbf{X}^\top}{\sqrt{C}}\right)$
- diagonal always be greater than off-diagonals
- significantly limits expressive power
- How do we fix this?



Issues With Self-Attention

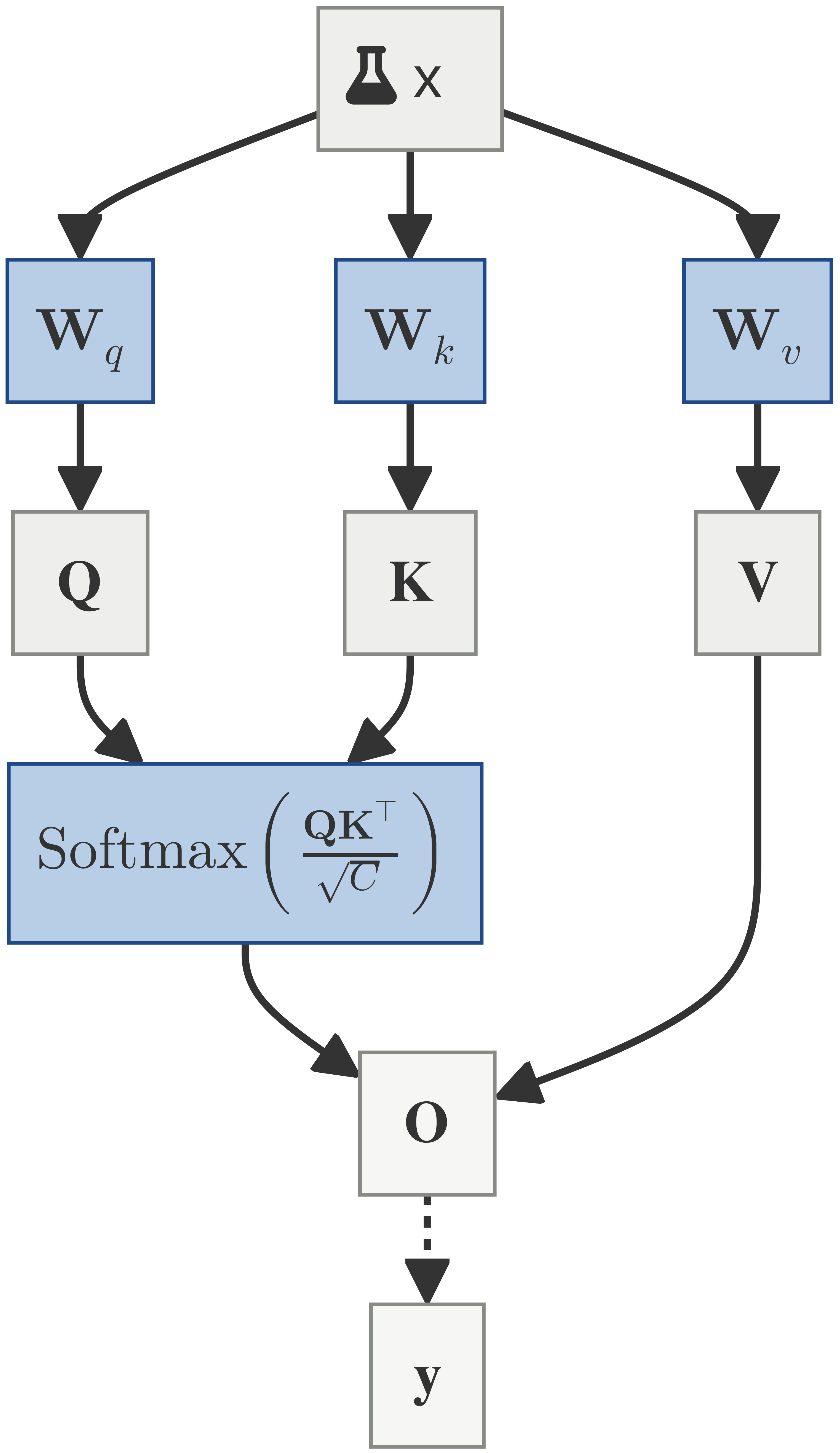
- $\mathbf{O} = \text{Attention}(\mathbf{X}, \mathbf{X}, \mathbf{X}) = \text{softmax}\left(\frac{\mathbf{X}\mathbf{X}^\top}{\sqrt{C}}\right)\mathbf{X}$
- $\alpha_{i,j} \leq \max(\alpha_{i,i}, \alpha_{j,j})$
- diagonal always be greater than off-diagonals
- significantly limits expressive power
- **Solution:** Apply linear transformation to **Q**, **K**, **V**



Attention with weights

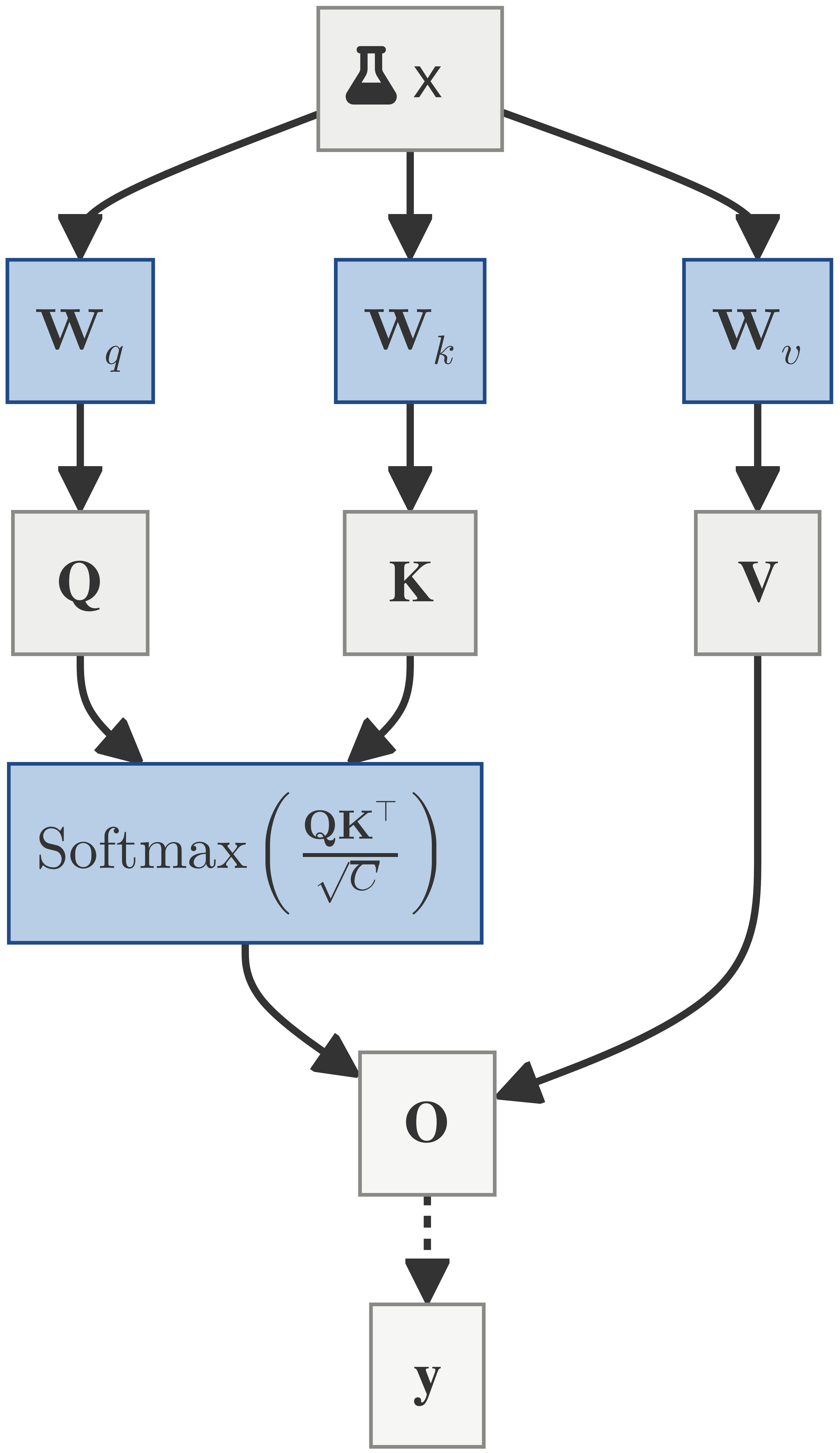
- $$\text{Attention}(\mathbf{X}; \mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V) = \text{Attention}(\mathbf{X}\mathbf{W}_Q, \mathbf{X}\mathbf{W}_K, \mathbf{X}\mathbf{W}_V) = \text{Softmax}\left(\frac{\mathbf{X}\mathbf{W}_Q(\mathbf{X}\mathbf{W}_K)^\top}{\sqrt{d_k}}\right)\mathbf{X}\mathbf{W}_V$$

- **Learnable parameters:** $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$



Issues with attention with weights

- Single attention weight / softmax
  - Can only attention to one other element or average values of a few



Multi-Head Attention

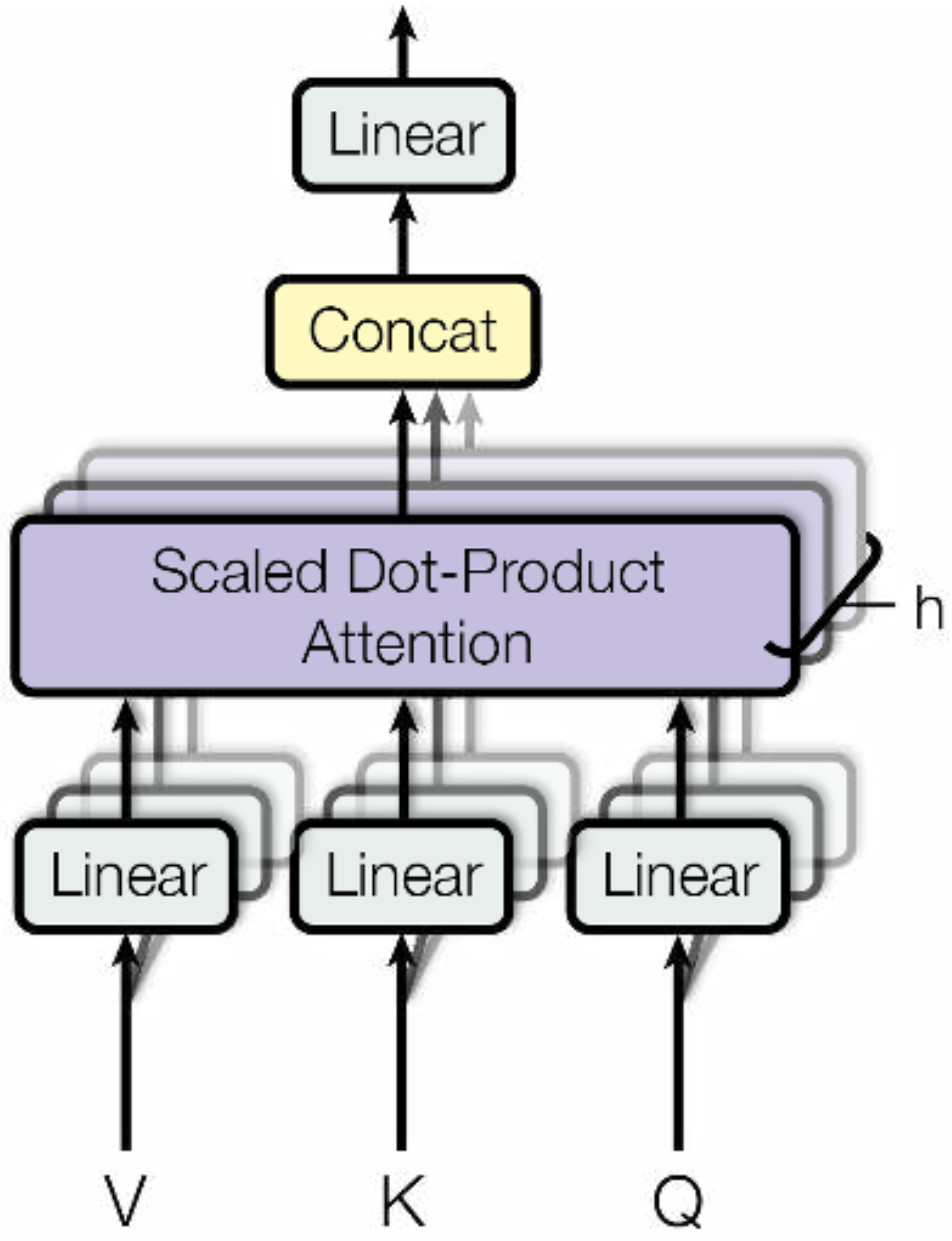
- Simple concatenation of multiple attention layers (“heads")

$$\text{Attention}(\mathbf{X}; \mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V)$$

$$= \text{Attention}(\mathbf{X}\mathbf{W}_Q, \mathbf{X}\mathbf{W}_K, \mathbf{X}\mathbf{W}_V)$$

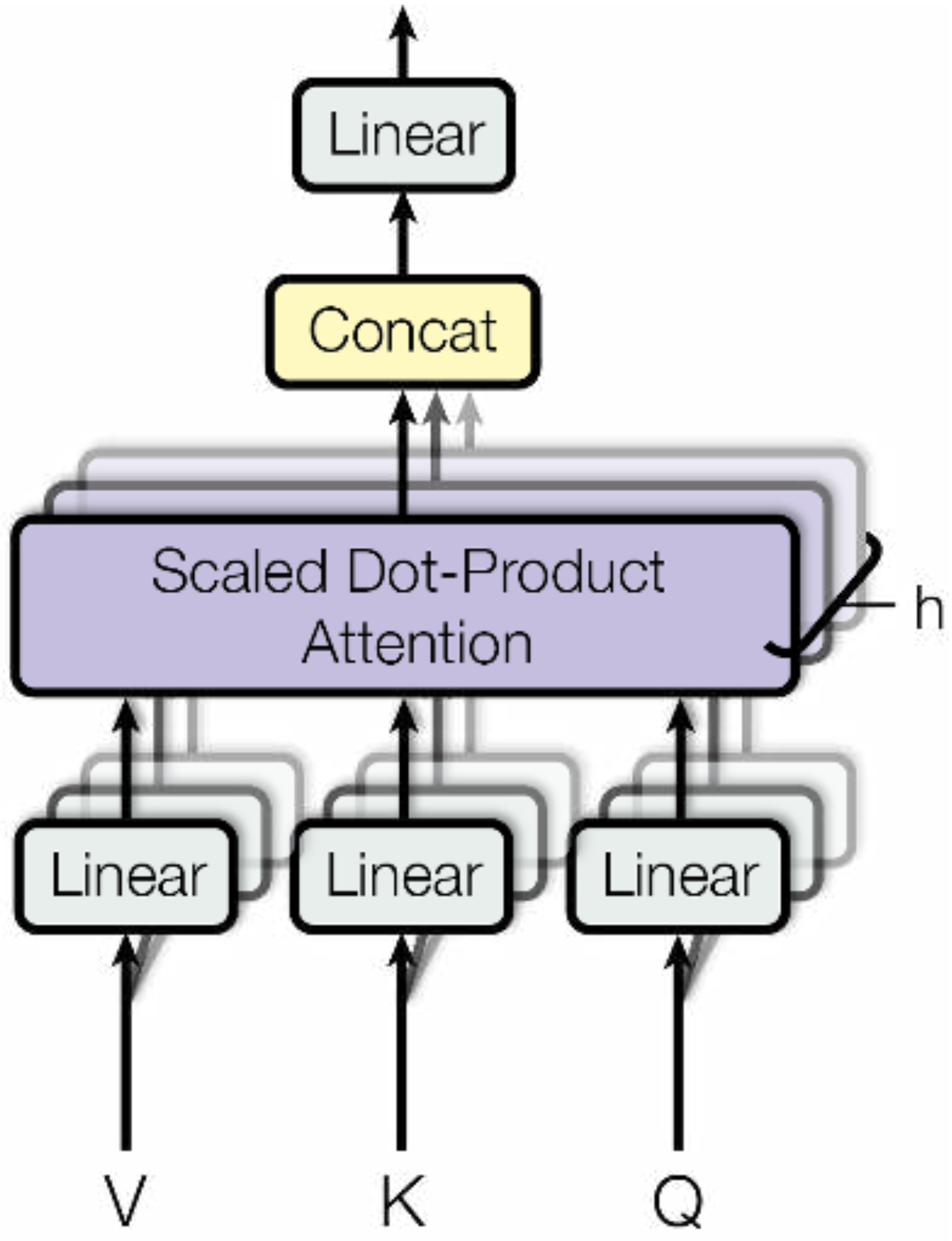
- $$= \text{Softmax}\left(\frac{\mathbf{X}\mathbf{W}_Q(\mathbf{X}\mathbf{W}_K)^\top}{\sqrt{d_k}}\right)\mathbf{X}\mathbf{W}_V$$

- **Learnable parameters per head:** $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$



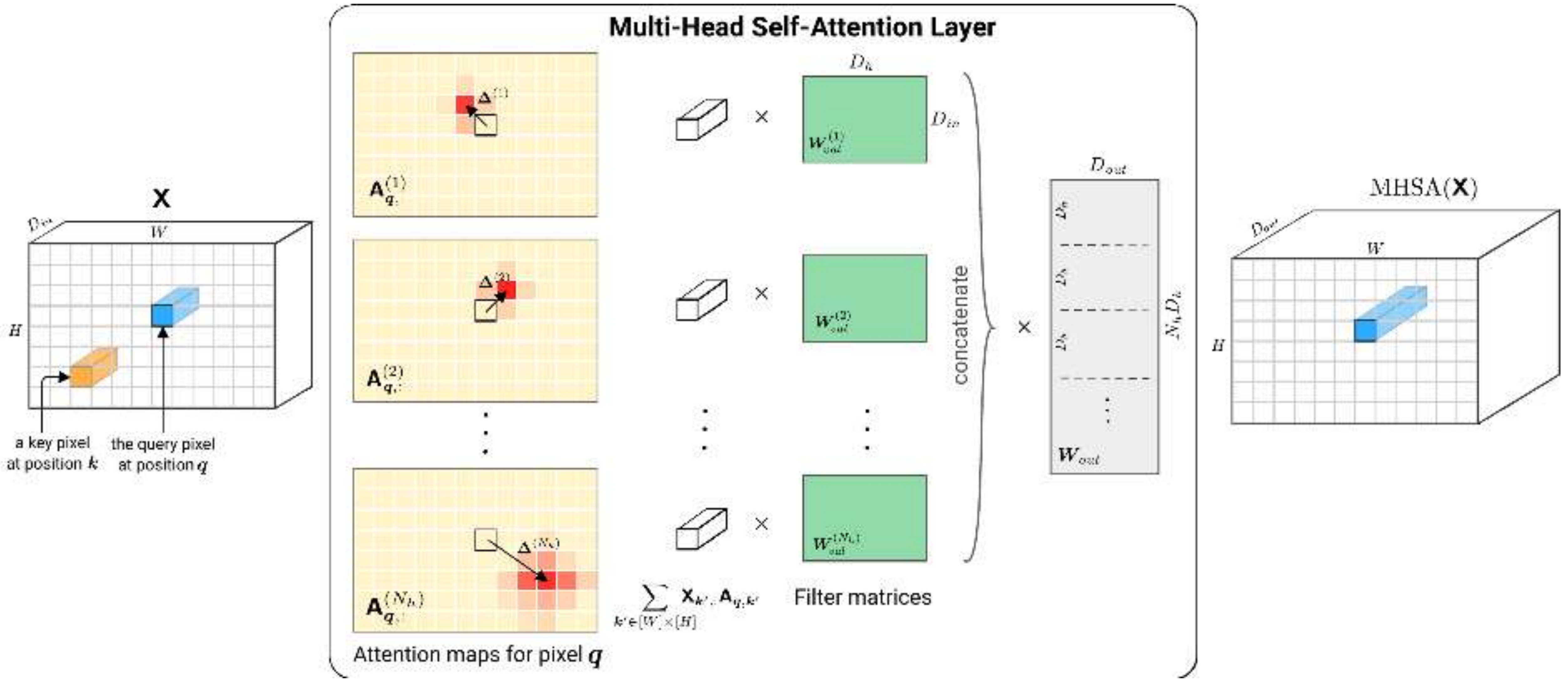
Multi-Head Attention

- h heads, each with a set of linear projections
  - $\mathbf{W}_{K,h} \in \mathbb{R}^{C \times d_k}$
  - $\mathbf{W}_{Q,h} \in \mathbb{R}^{C \times d_k}$
  - $\mathbf{W}_{V,h} \in \mathbb{R}^{C \times d_v}$
- A final linear projection to map to output dimension
  - $\mathbf{W}_O \in \mathbb{R}^{hd_v \times C}$
- $\begin{bmatrix} \text{Attention}(\mathbf{XW}_{Q,1}, \mathbf{XW}_{K,1}, \mathbf{XW}_{V,1}) \\ \vdots \\ \text{Attention}(\mathbf{XW}_{Q,h}, \mathbf{XW}_{K,h}, \mathbf{XW}_{V,h}) \end{bmatrix} W_O$



Connection to Convolution

- Multi-head attention with h heads is **more expressive** than a $\sqrt{h} \times \sqrt{h}$ 2D conv [1]



Cordonnier et al. On the relationship between self-attention and convolutional layers. ICLR 2020

Attention - TL;DR

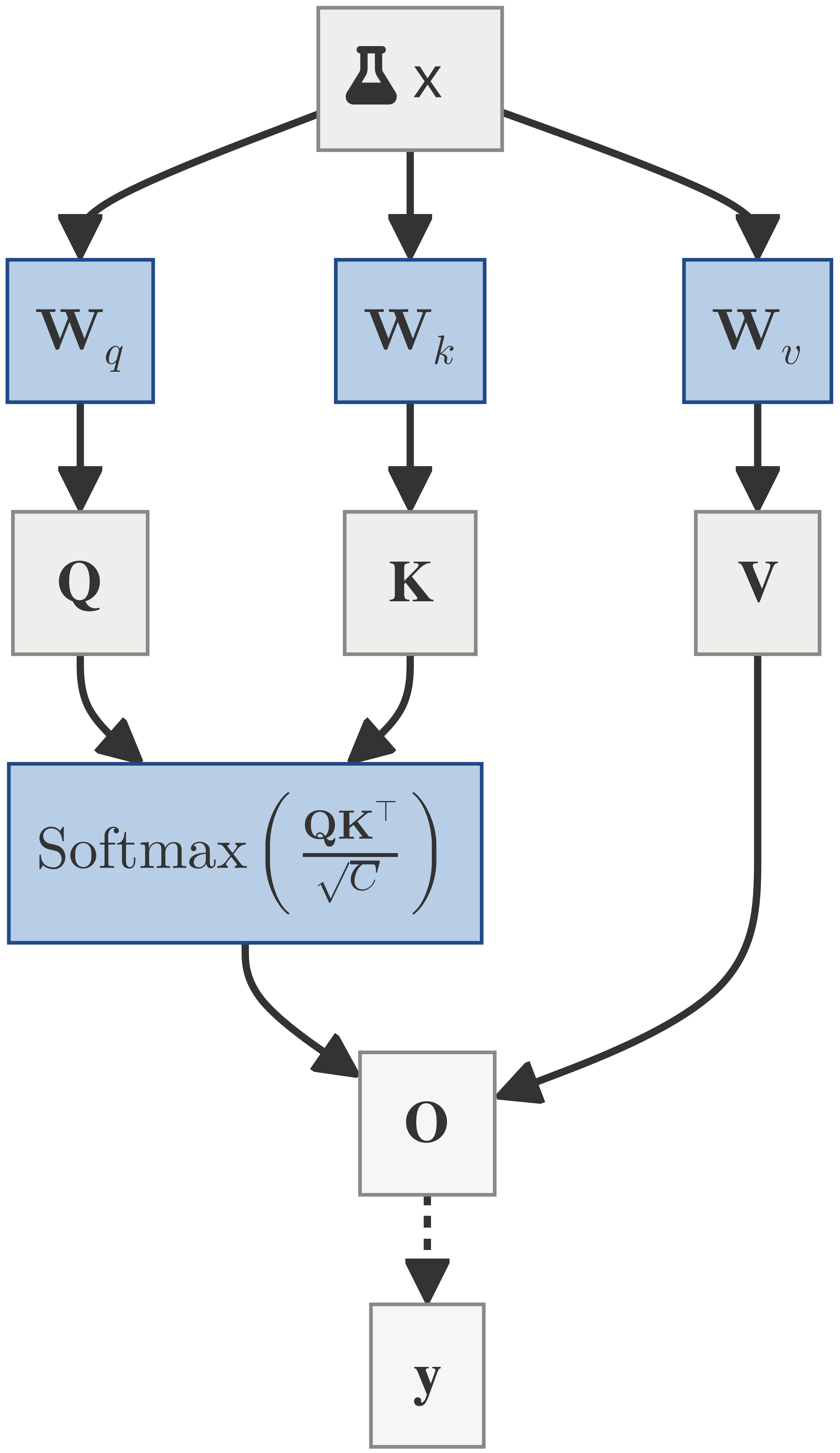
- Attention is a **set operation** which reasons about set elements
- Attention takes three inputs - **queries**, **keys** and **values**
- Always use **multi-head attention** (with weights)

Positional embeddings

Recap: Attention with weights

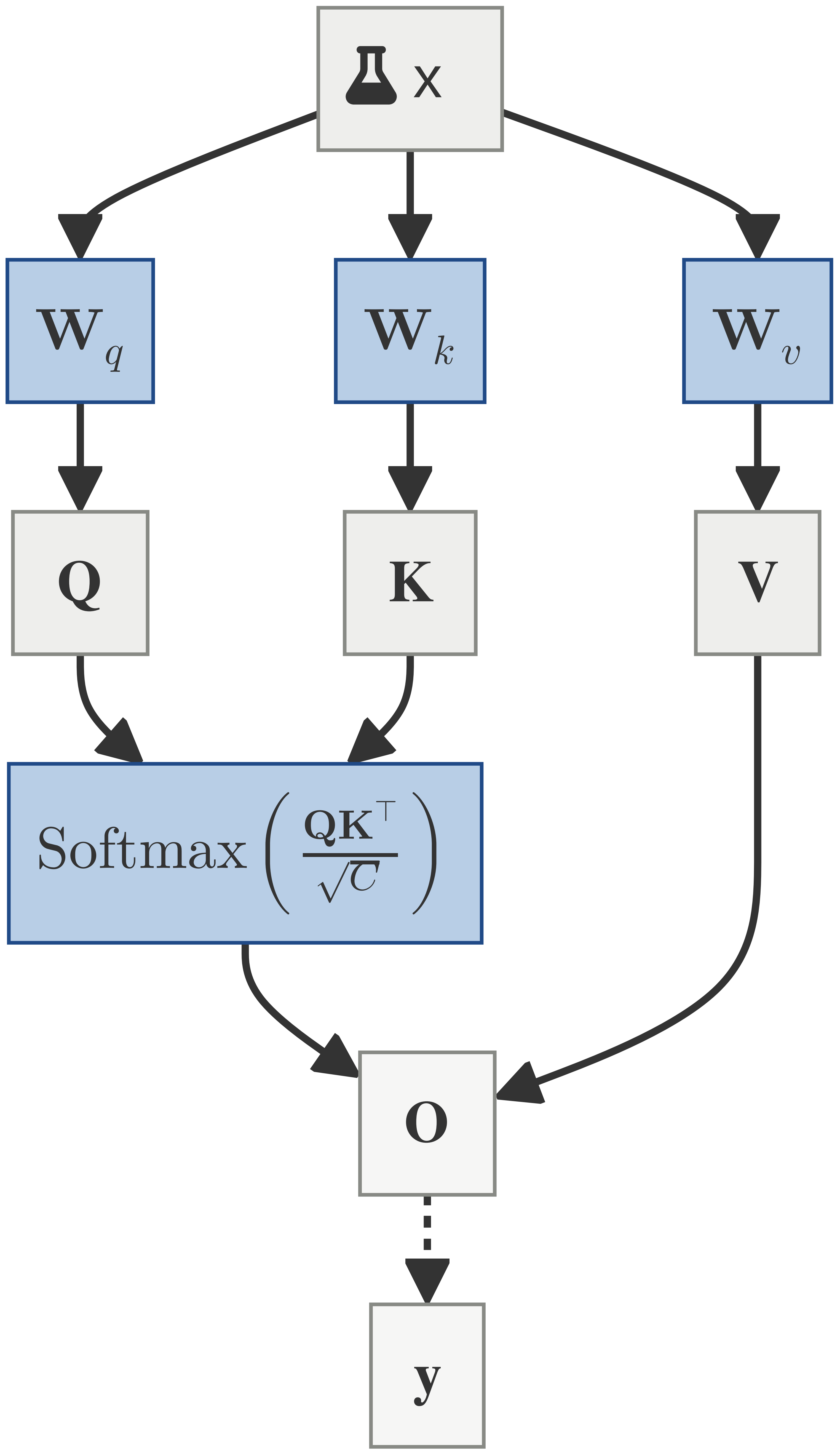
- $$\begin{aligned}&\text{Attention}(\mathbf{X}; \mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V)\\ &= \text{Attention}(\mathbf{X}\mathbf{W}_Q, \mathbf{X}\mathbf{W}_K, \mathbf{X}\mathbf{W}_V)\\ &= \text{Softmax}\left(\frac{\mathbf{X}\mathbf{W}_Q(\mathbf{X}\mathbf{W}_K)^\top}{\sqrt{d_k}}\right)\mathbf{X}\mathbf{W}_V\end{aligned}$$

- **Learnable parameters:** $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$



Permutation Invariance

- Attention is a *set* operation
  - shuffling keys/values gives the same output
- Let $\mathbf{Perm}(\mathbf{M})$ denote an arbitrary permutation operation over the rows of a matrix $\mathbf{M}$
- $$\begin{aligned}\text{Attention}(\mathbf{XW}_Q, \text{Perm}(\mathbf{XW}_K), \text{Perm}(\mathbf{XW}_V)) \\ = \text{Attention}(\mathbf{XW}_Q, \mathbf{XW}_K, \mathbf{XW}_V)\end{aligned}$$



Example: Set Operations

- Given a set of reported power outages in the last few days, predict the number of power outages in the future

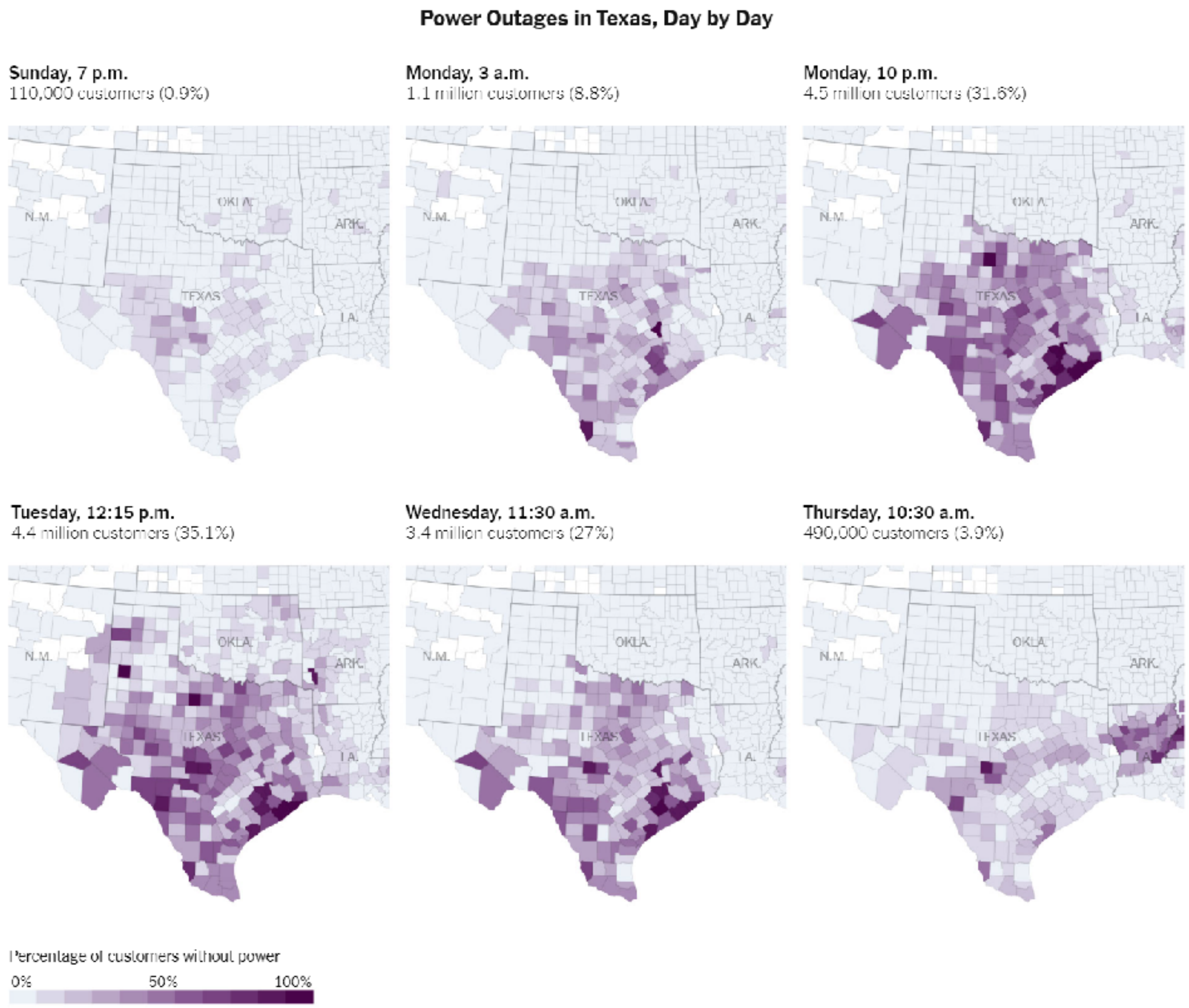
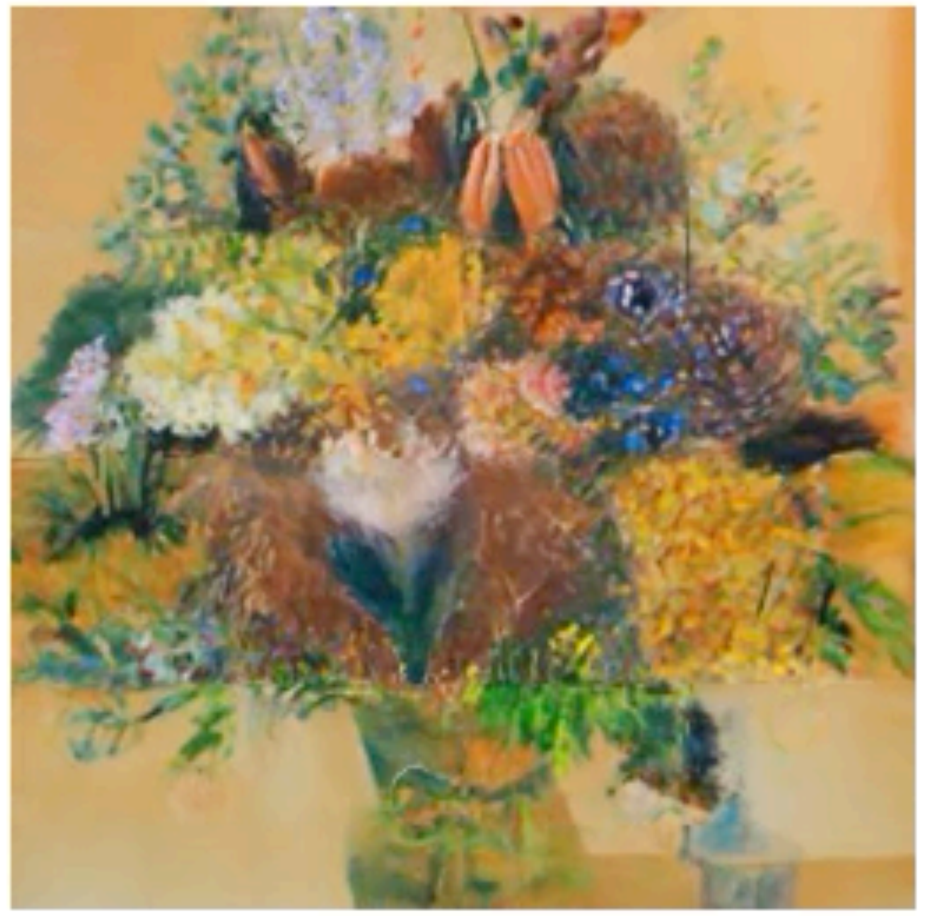


Image credit: New York Times

Example: Ordered Operations

- **Natural Language** - ordered array of words
  my kid likes the movie ≠ the movie likes my kid
- **Speech** - sequence of sound waves
- **Images** - ordered set of smaller patches
  - What happens when you shuffle an image?
  - Fun demo: Visual Anagrams [1]

a painting of a deer



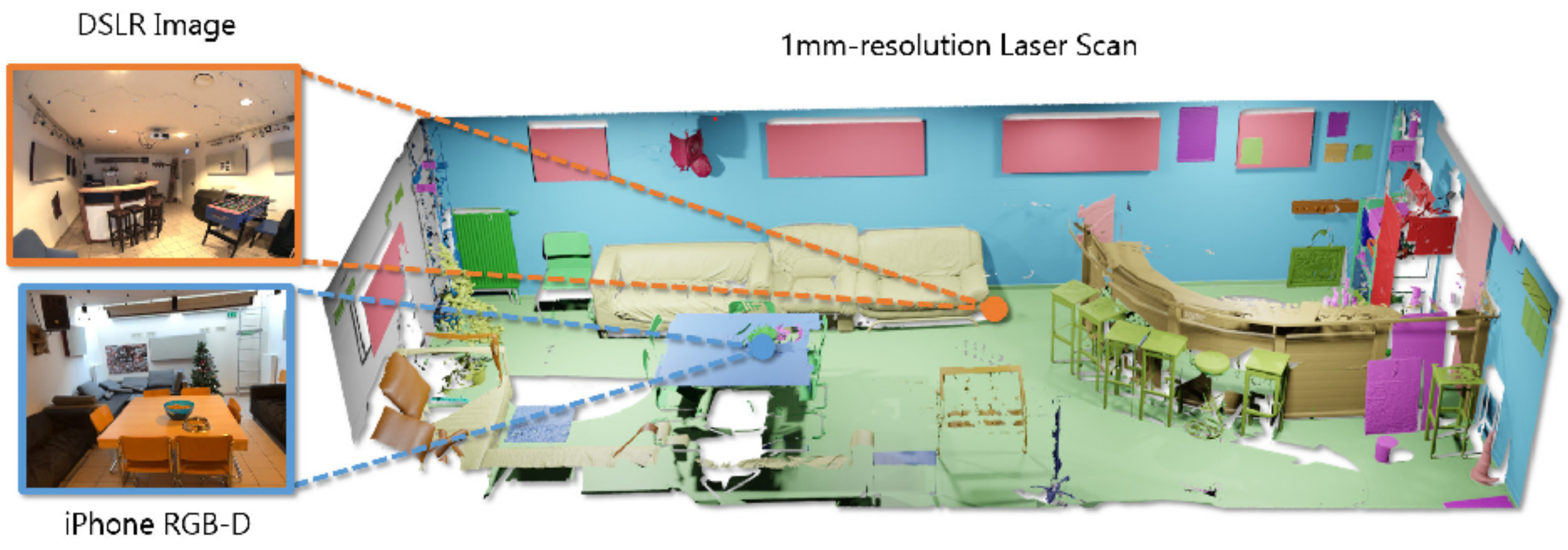
an oil painting of flowers

a painting of houseplants

[1] Geng, et al. Visual Anagrams: Generating Multi-View Optical Illusions with Diffusion Models. CVPR 2024

Example: Somewhat Ordered Operations

- Point Clouds
  - Set of N points $p_i \in \mathbb{R}^3$ - $P = \{p_1, p_2, \ldots, p_N\}$

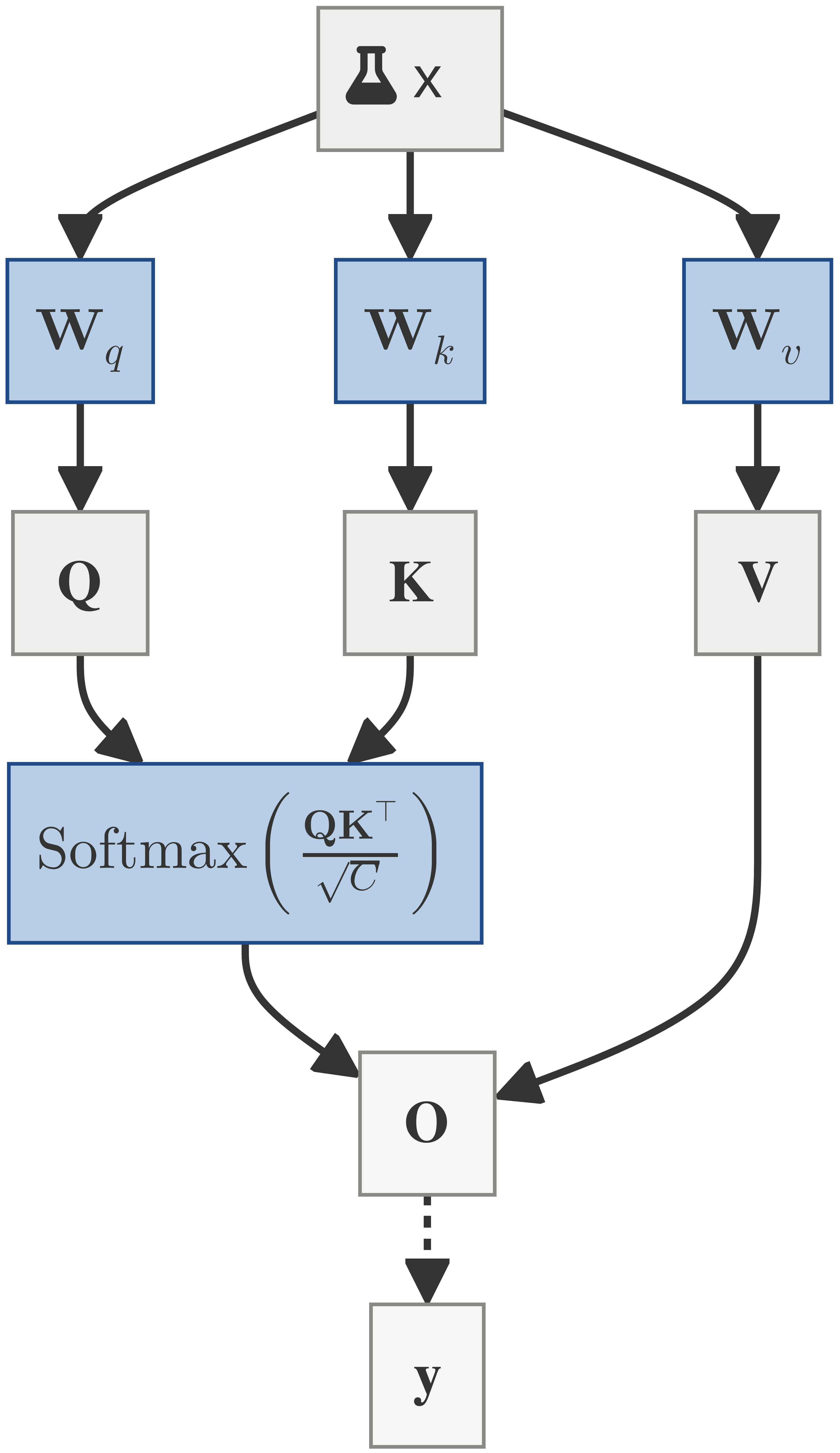


Attention without Positional Embeddings

$$\text{Attention}(\mathbf{X}; \mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V)$$

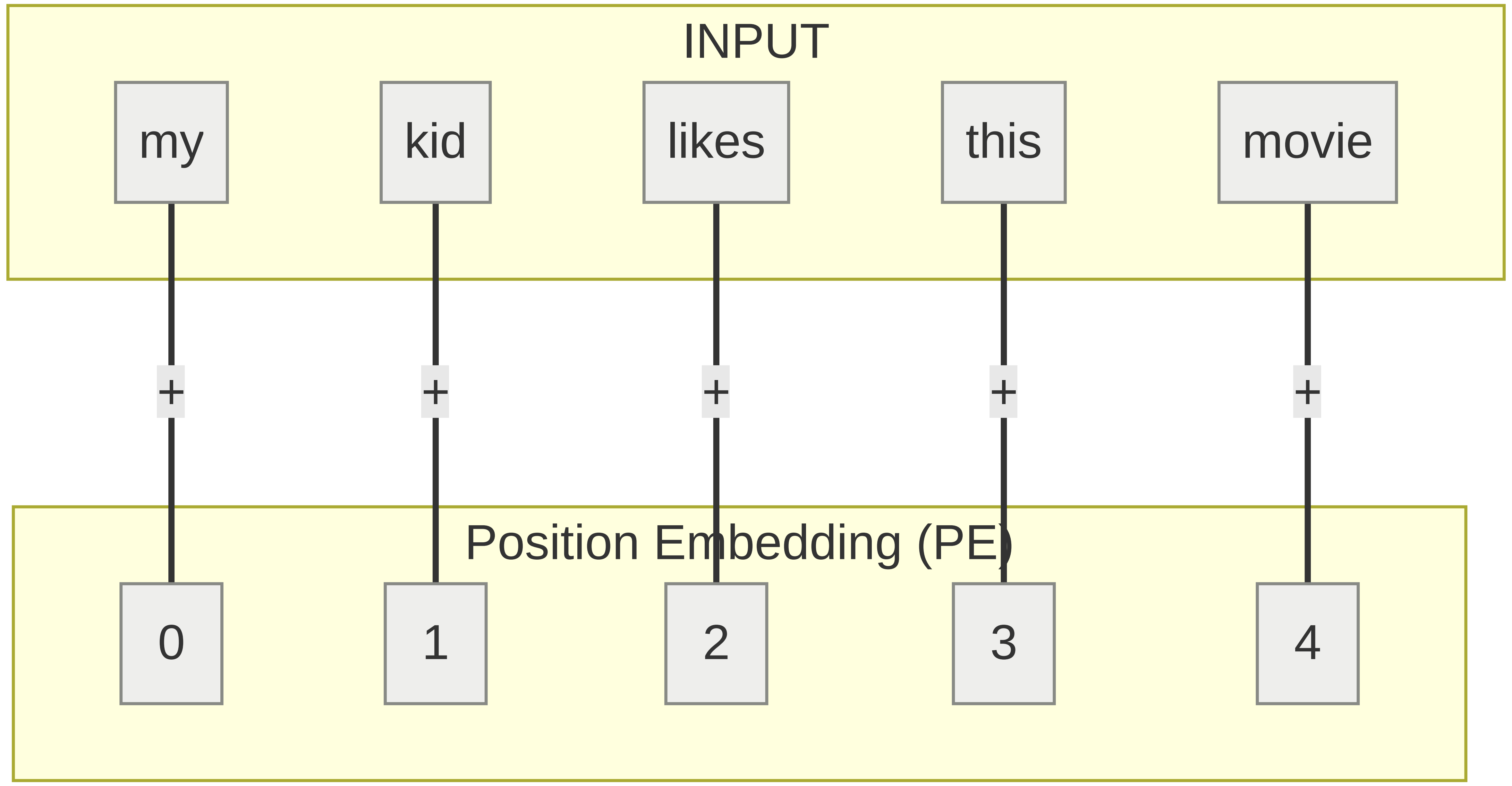
$$= \text{Attention}(\mathbf{X}\mathbf{W}_Q, \mathbf{X}\mathbf{W}_K, \mathbf{X}\mathbf{W}_V)$$

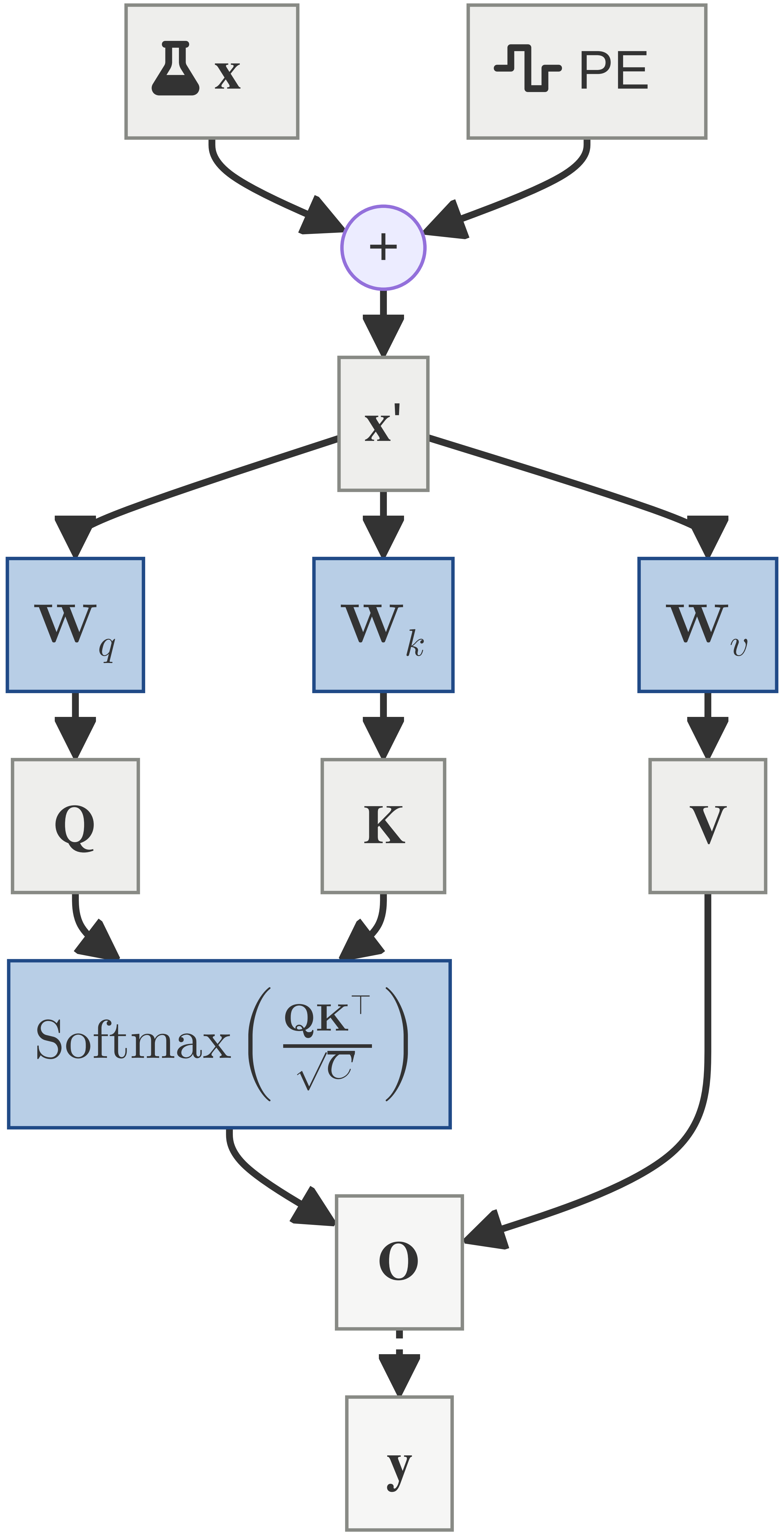
- $$= \text{Softmax}\left(\frac{\mathbf{X}\mathbf{W}_Q(\mathbf{X}\mathbf{W}_K)^\top}{\sqrt{d_k}}\right)\mathbf{X}\mathbf{W}_V$$



Attention With Positional Embedding

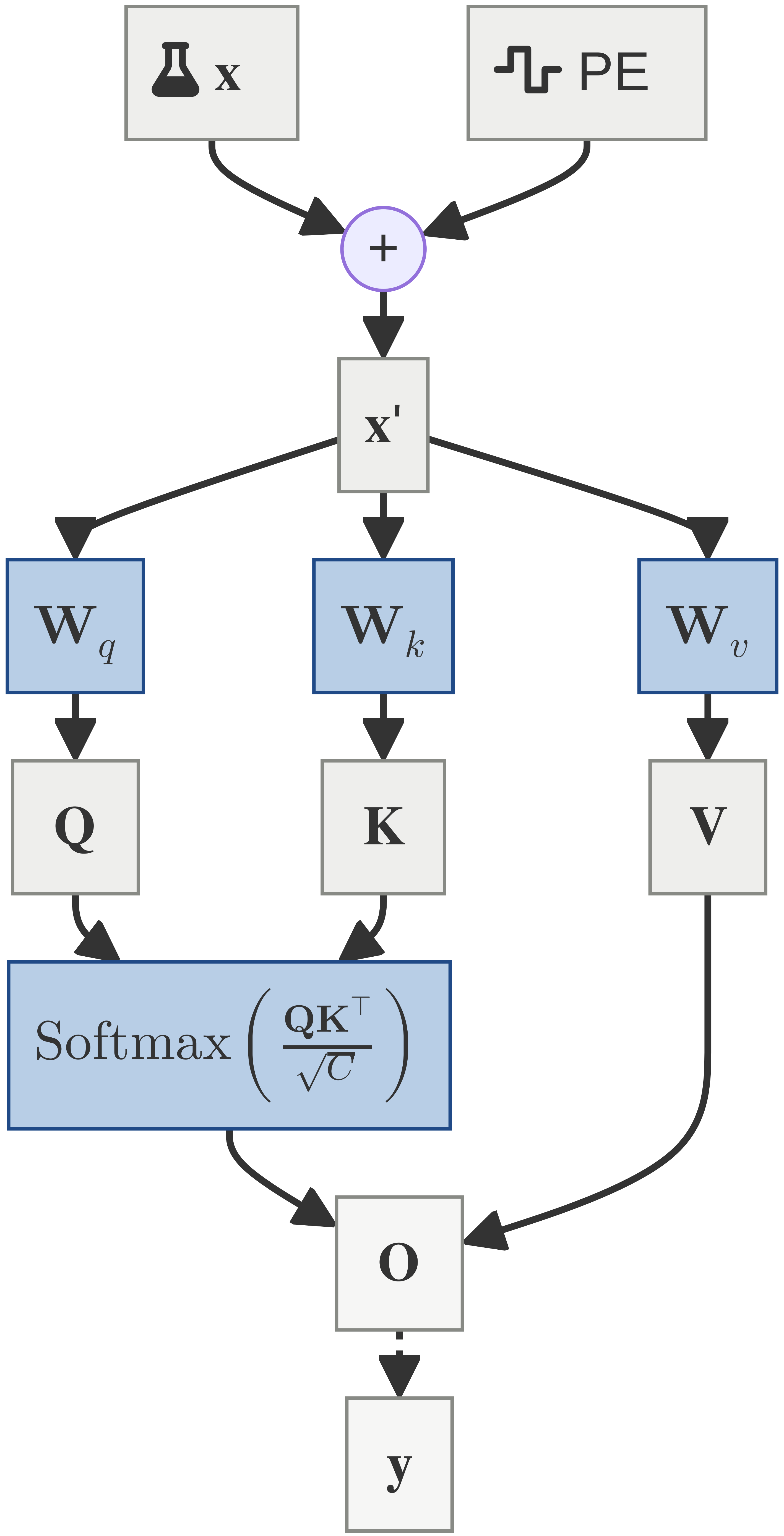
- Describes the location of elements in a sequence
  - add position information to Q, K, V





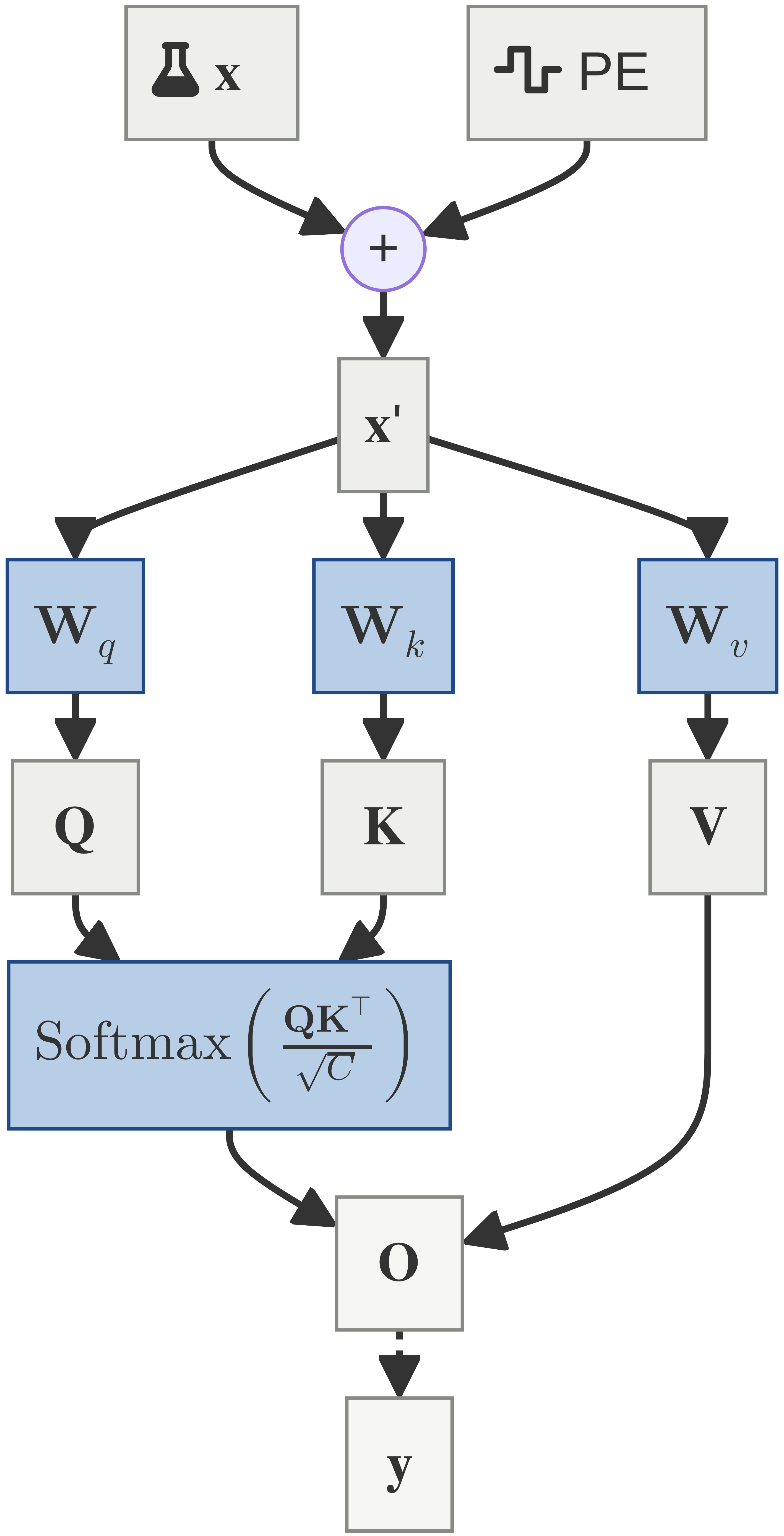
Absolute Positional Embeddings

- **Option 1**: Raw position

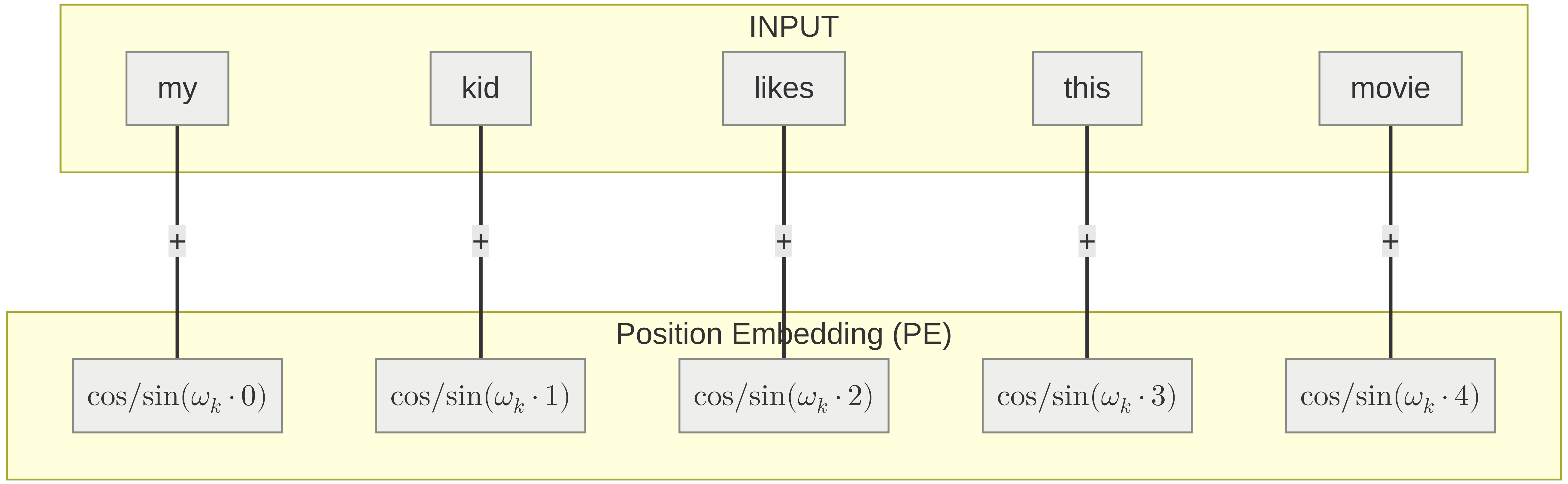


Absolute Positional Embeddings

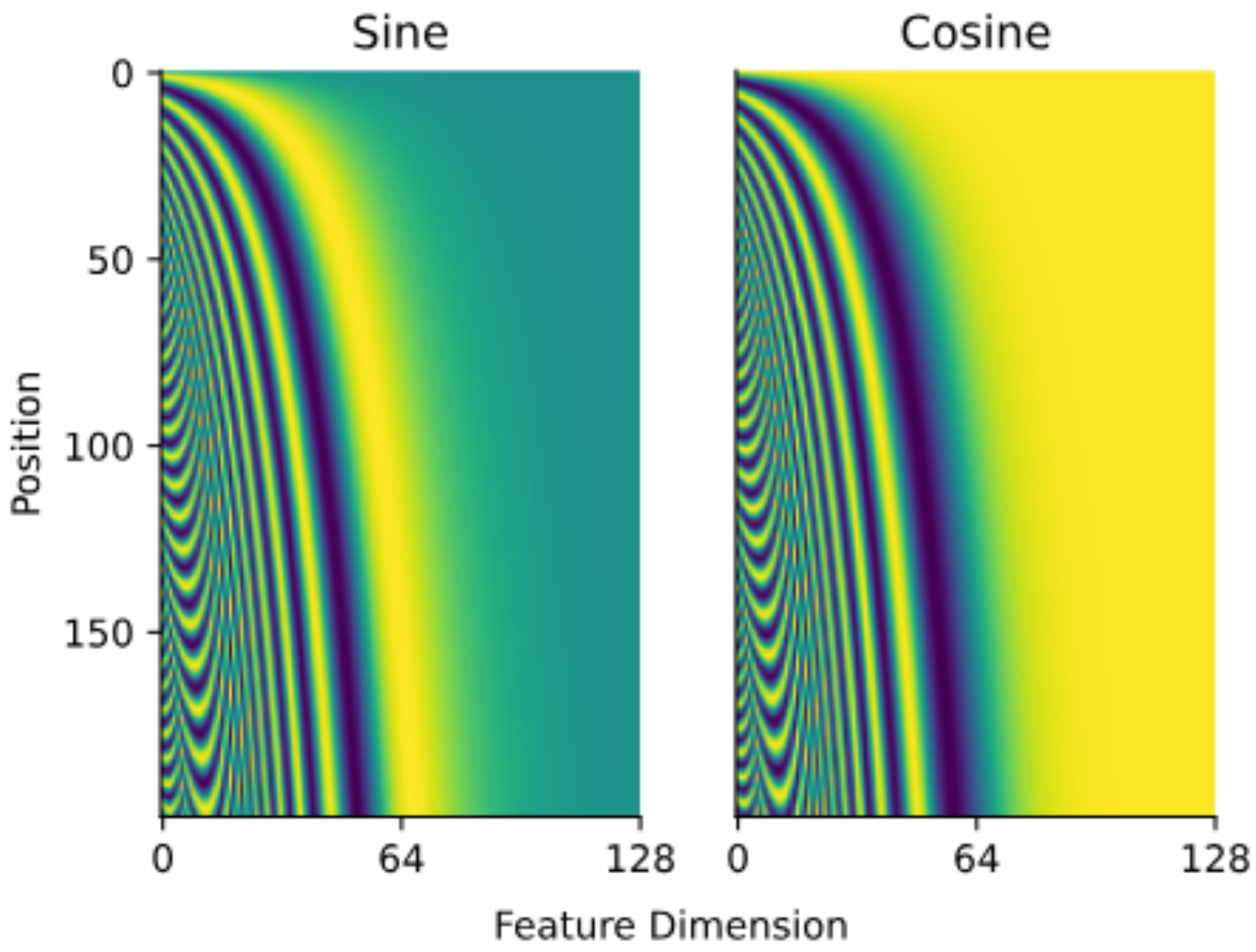
- **Option 1**: Raw position
  - Bad idea 👎: Hard to interpret



Sinusoidal Embeddings

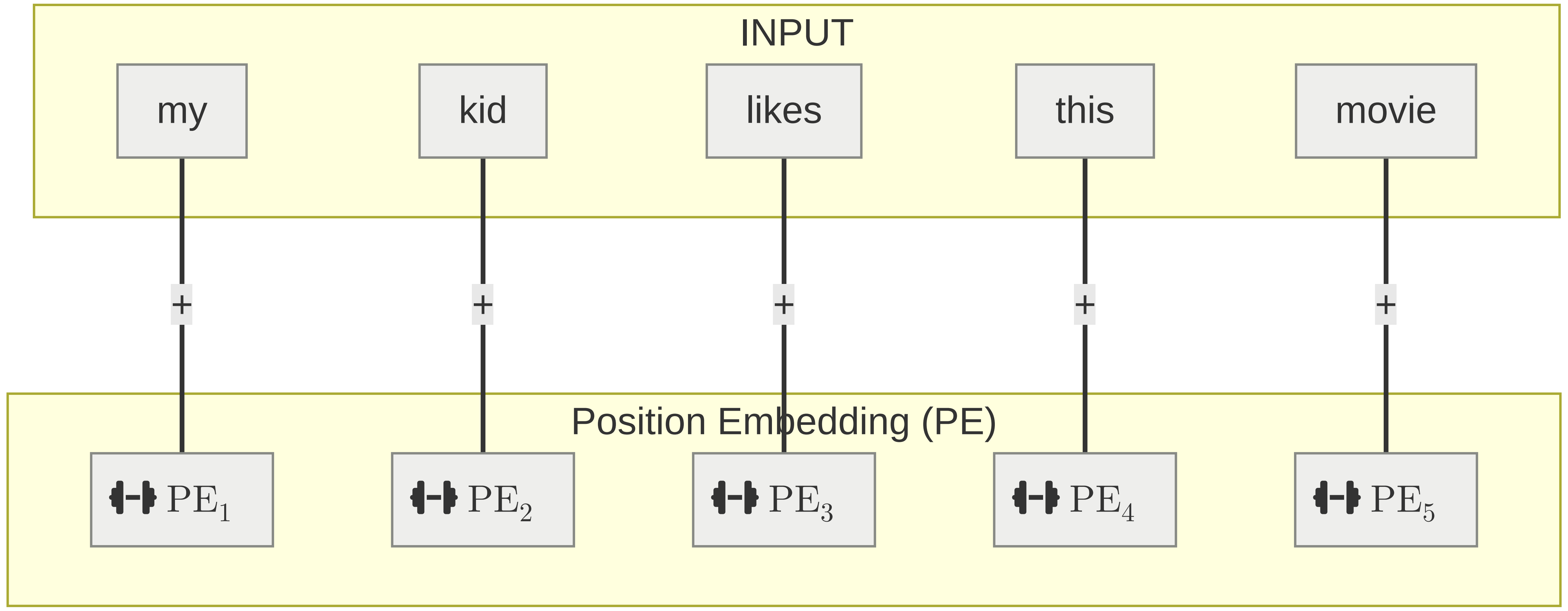


- **Option 2**: encode position with sine/cosine $\mathrm{PE} \in \mathbb{R}^{N \times C}$
  - Use sine & cosine functions with varied frequencies
- $\mathrm{PE}(n, 2i) = \sin\left(\frac{n}{10000^{2i/C}}\right)$
- $\mathrm{PE}(n, 2i+1) = \cos\left(\frac{n}{10000^{2i/C}}\right)$
- 👍 "Kind of" absolute
- 👍 Position represented well by frequency/phase



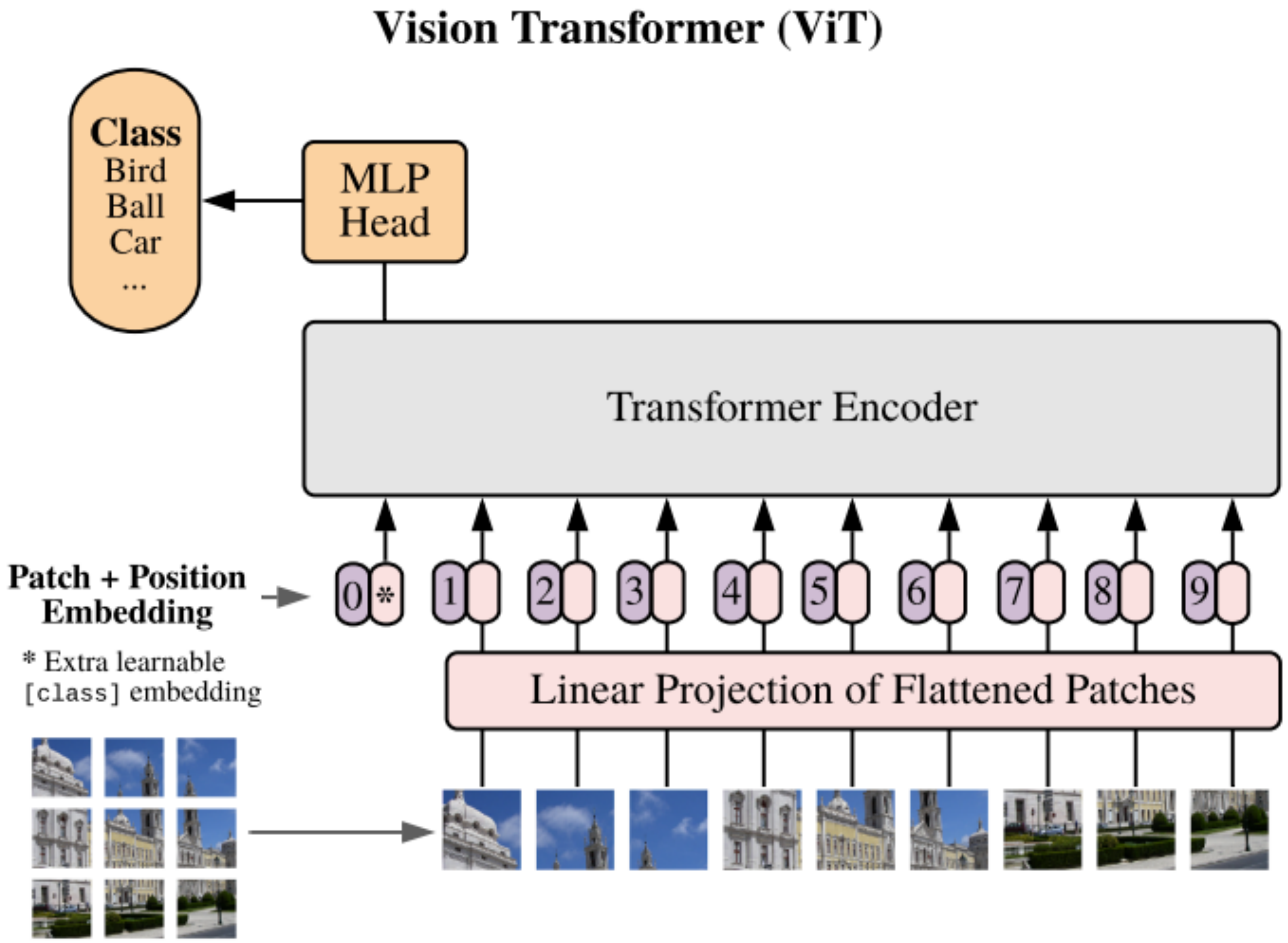
Learnable Positional Embedding

- **Option 3**: learned encoding
  - Embeddings $\mathbf{PE} \in \mathbb{R}^{N \times C}$ randomly initialized
  - Learned through training
- 👍 Fully learned
- 👍 Use when frequency information is not obvious
- 👎 Performance drops when the sequence length varies between train/test



Learnable positional embeddings

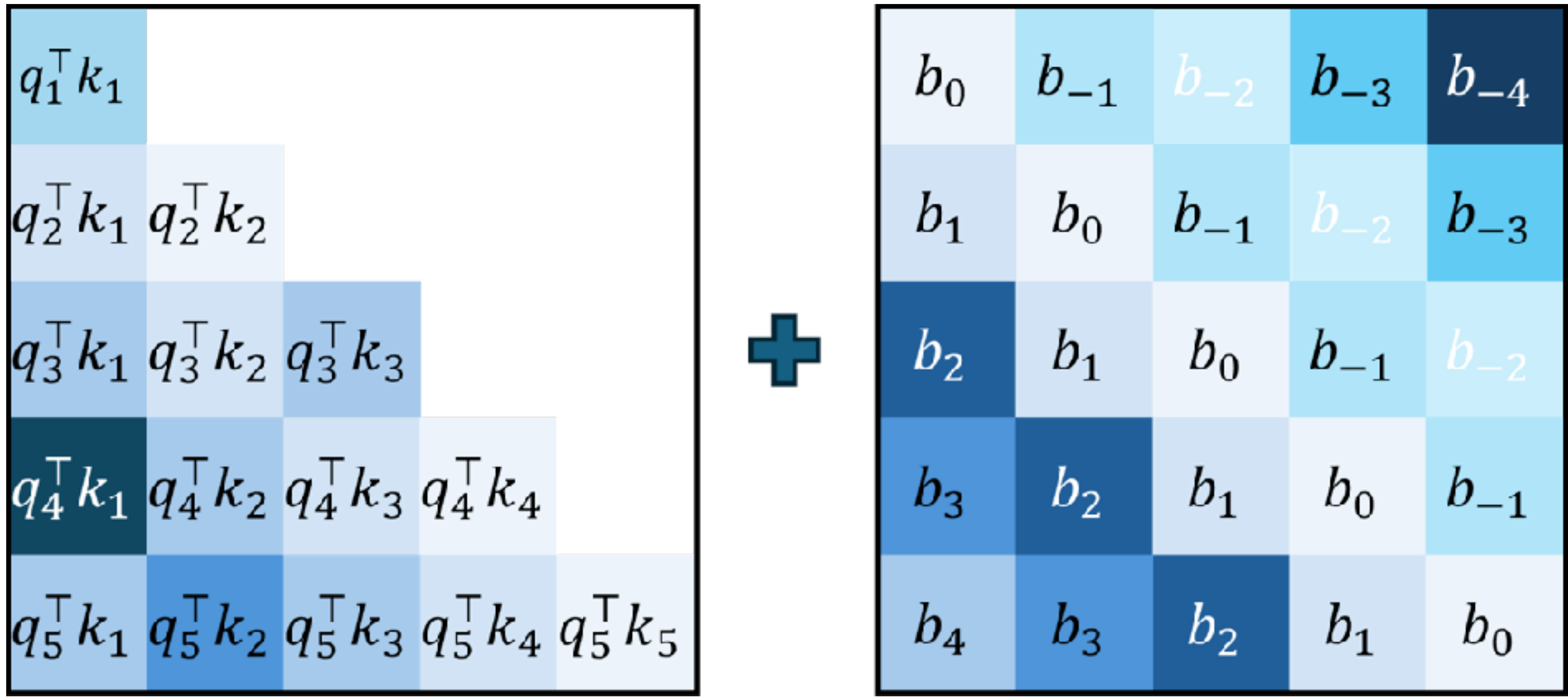
- Example: Vision Transformer (ViT)



Dosovitskiy, et al. "An image is worth 16x16 words: Transformers for image recognition at scale. ICLR 2021

Relative Positional Embedding (1): T5-bias

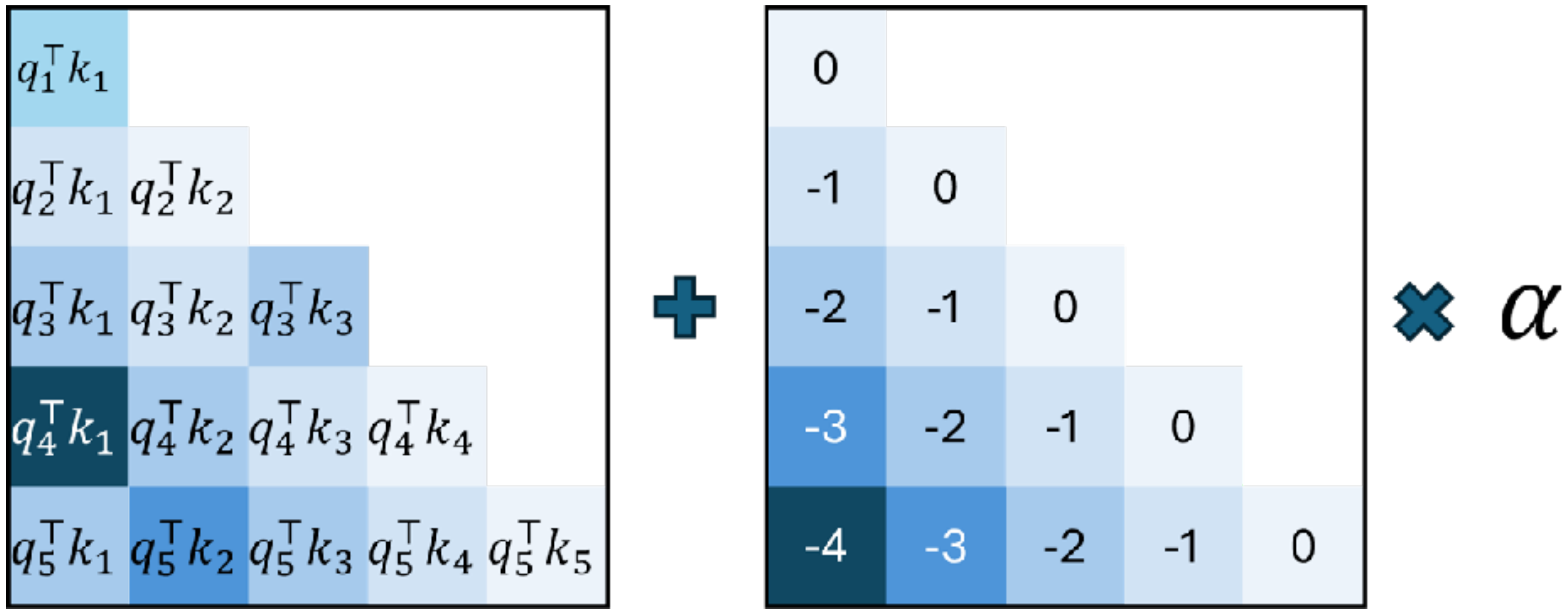
- **Option 4a**: pairwise/relative encoding
  - Takes the form of $\mathrm{PE}(m, n) = f(m, n)$
- $$\mathbf{O} = \mathrm{softmax}\left(\frac{\mathbf{XW}_Q(\mathbf{XW}_K)^\top + \mathbf{B}}{\sqrt{C}}\right)(\mathbf{XW}_V)$$
- $$\mathbf{B} = \begin{bmatrix} b_0 & b_{-1} & b_{-2} & \cdots & b_{-N+1} \\ b_1 & b_0 & b_{-1} & \cdots & b_{-N+2} \\ b_2 & b_1 & b_0 & \cdots & b_{-N+3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{N-1} & b_{N-2} & b_{N-3} & \cdots & b_0 \end{bmatrix}$$
- 👍 Generalizes better to sequences of unseen lengths



Raffel, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. JMLR 2020.

Relative Positional Embedding (2): Alibi

- **Option 4b**: pairwise/relative encoding
  - Takes the form of $\mathrm{PE}(m, n) = f(m, n)$
- $$\mathbf{O} = \mathrm{softmax}\left(\frac{\mathbf{XW}_Q(\mathbf{XW}_K)^\top + \mathbf{B}}{\sqrt{C}}\right)(\mathbf{XW}_V)$$
- $$\mathbf{B} = \alpha \cdot \begin{bmatrix} 0 & \cdots & & & \\ -1 & 0 & \cdots & & \\ -2 & -1 & 0 & \cdots & \\ \vdots & \vdots & \vdots & \ddots & \\ -N+1 & -N+2 & -N+3 & \cdots & 0 \end{bmatrix}$$
- 👍 Generalizes better to sequences of unseen lengths



Press et al. Train short, test long: Attention with linear biases enables input length extrapolation. ICLR 2022

Relative Positional Embedding (3)

- **Option 4c**: pairwise/relative encoding
  - Encode edge between two arbitrary positions i and j
- $$\mathbf{O} = \text{softmax}\left(\frac{\mathbf{XW}_Q(\mathbf{XW}_K + \mathbf{P}_{ij}^K)^\top}{\sqrt{C}}\right)(\mathbf{XW}_V + \mathbf{P}_{ij}^V)$$

Shaw et al. Self-Attention with Relative Position Representations. NAACL 2018

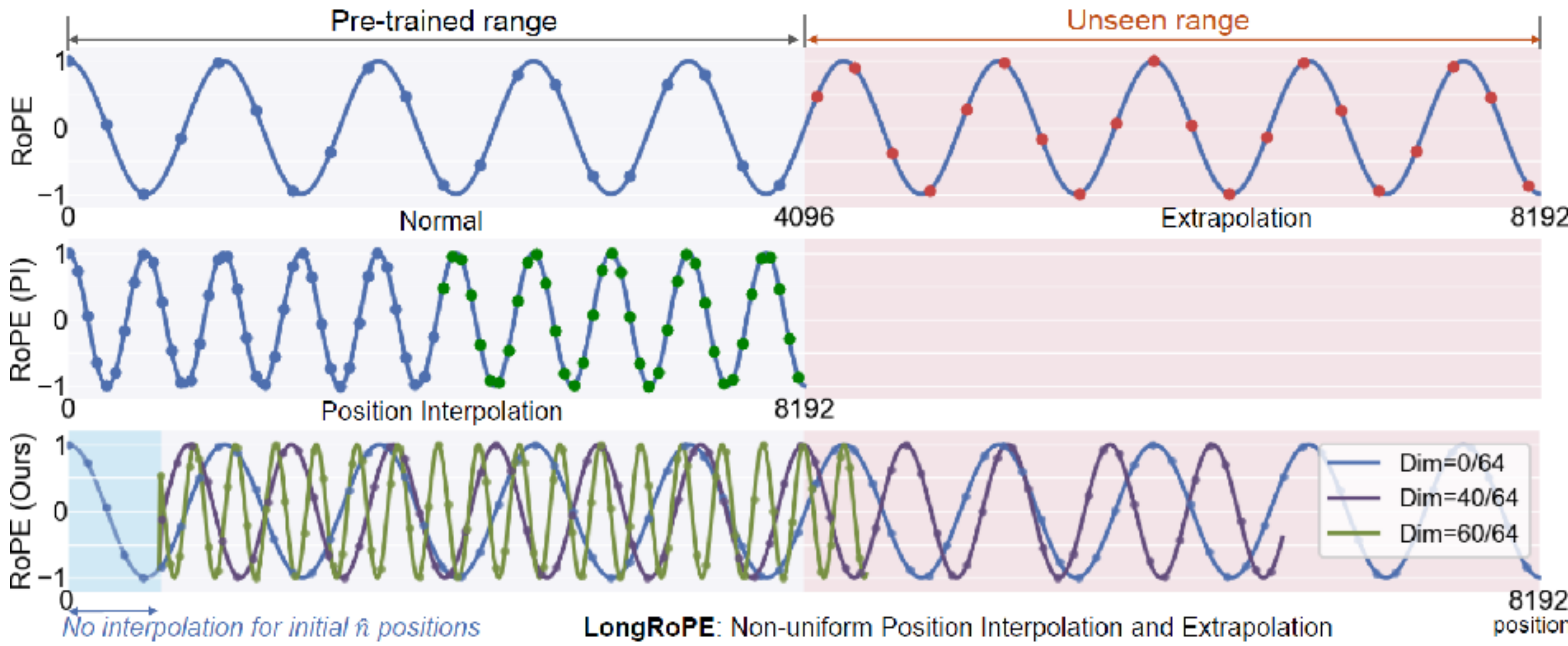
Rotary Positional Embedding: RoPE

- **Option 5**: rotary encoding
  - Both absolute PE and relative PE
  - Goal: find a kernel function such that
- $h(\mathbf{q}_m, \mathbf{k}_n) = \mathbf{q}_m^\top \mathbf{k}_n = g(\mathbf{q}_m, \mathbf{k}_n, m - n)$
- $\mathbf{q}_m = \mathbf{R}_m \mathbf{W}_Q \mathbf{x}_m \qquad \mathbf{k}_n = \mathbf{R}_n \mathbf{W}_K \mathbf{x}_n$
- where $\mathbf{R}_m = \begin{bmatrix} \cos(m\theta_1) & -\sin(m\theta_1) & 0 & 0 & \cdots & 0 & 0 \\ \sin(m\theta_1) & \cos(m\theta_1) & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos(m\theta_2) & -\sin(m\theta_2) & \cdots & 0 & 0 \\ 0 & 0 & \sin(m\theta_2) & \cos(m\theta_2) & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos(m\theta_{C/2}) & -\sin(m\theta_{C/2}) \\ 0 & 0 & 0 & 0 & \cdots & \sin(m\theta_{C/2}) & \cos(m\theta_{C/2}) \end{bmatrix}$

Su et al. RoFormer: Enhanced Transformer with Rotary Position Embedding. Neurocomputing 2024

Rotary Positional Embedding

- $$\begin{aligned} \mathbf{q}_m &= \mathbf{R}_m \mathbf{W}_Q \mathbf{x}_m \\ \mathbf{k}_n &= \mathbf{R}_n \mathbf{W}_K \mathbf{x}_n \\ \mathbf{q}_m^\top \mathbf{k}_n &= (\mathbf{R}_m \mathbf{W}_Q \mathbf{x}_m)^\top \mathbf{R}_n \mathbf{W}_K \mathbf{x}_n \\ &= \mathbf{x}_m^\top \mathbf{W}_Q^\top \mathbf{R}_{n-m} \mathbf{W}_k \mathbf{x}_n \end{aligned}$$

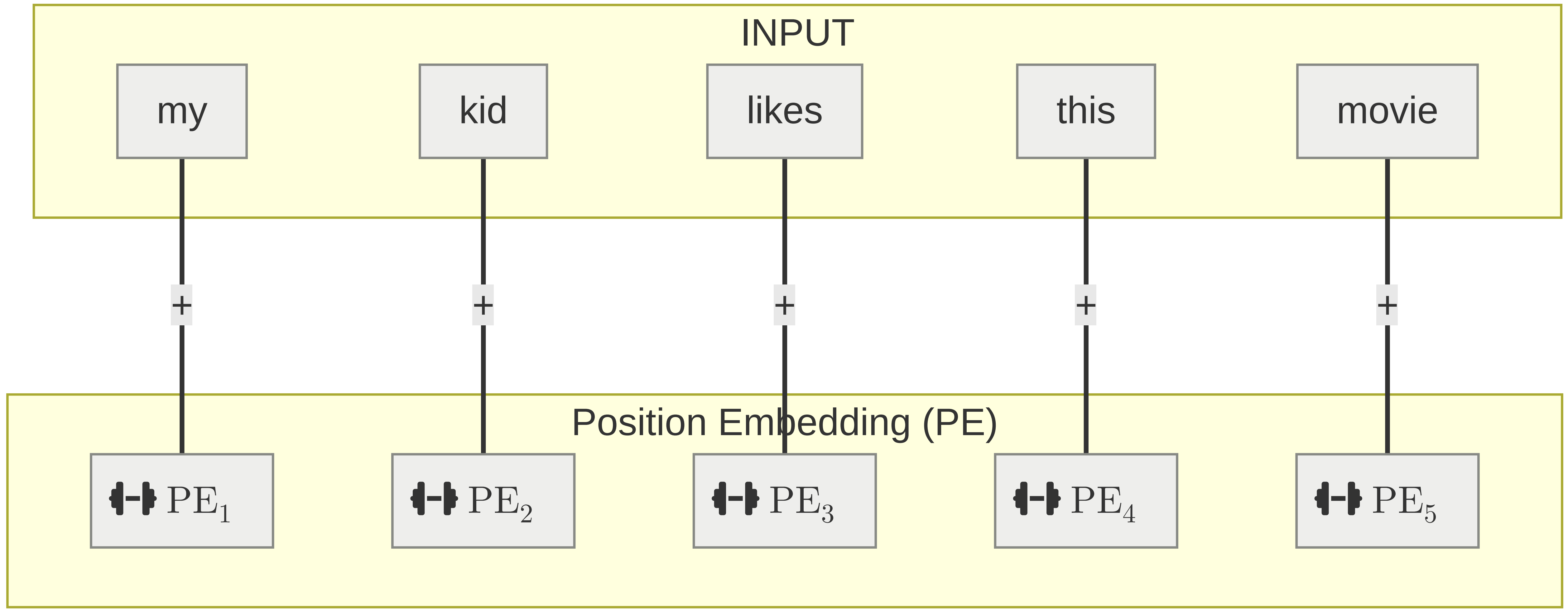


- Great extrapolation capability when context length between train/test varies
- Widely adopted in Large Language Models (LLMs) such as LLaMA

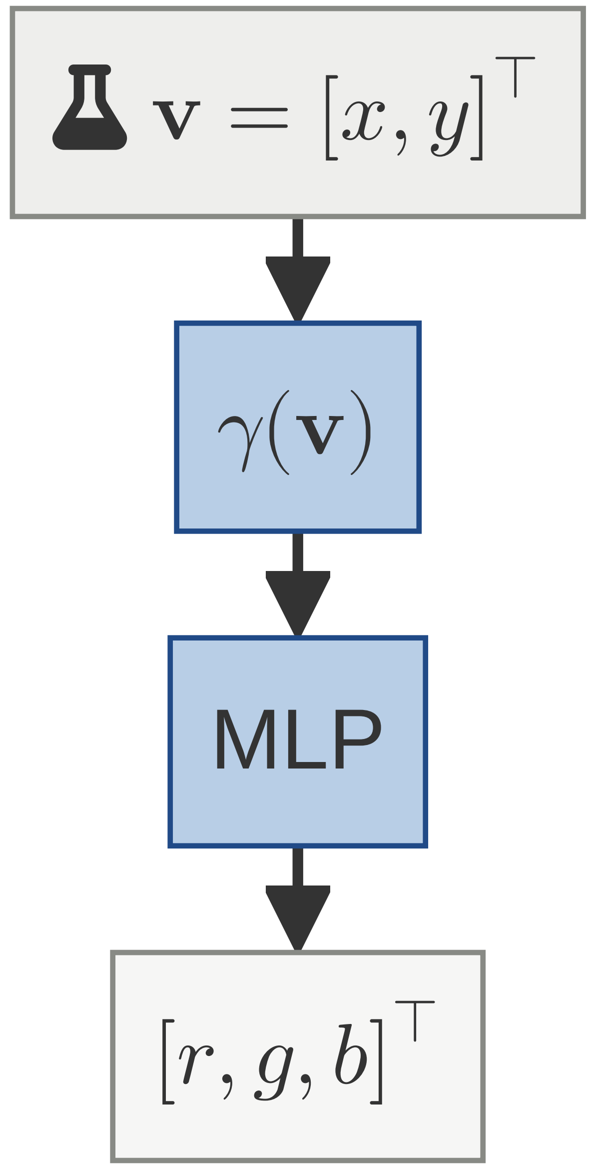
Touvron, et al. LLaMA: Open and efficient foundation language models. arXiv:2302.13971

Applications of PE: LLMs

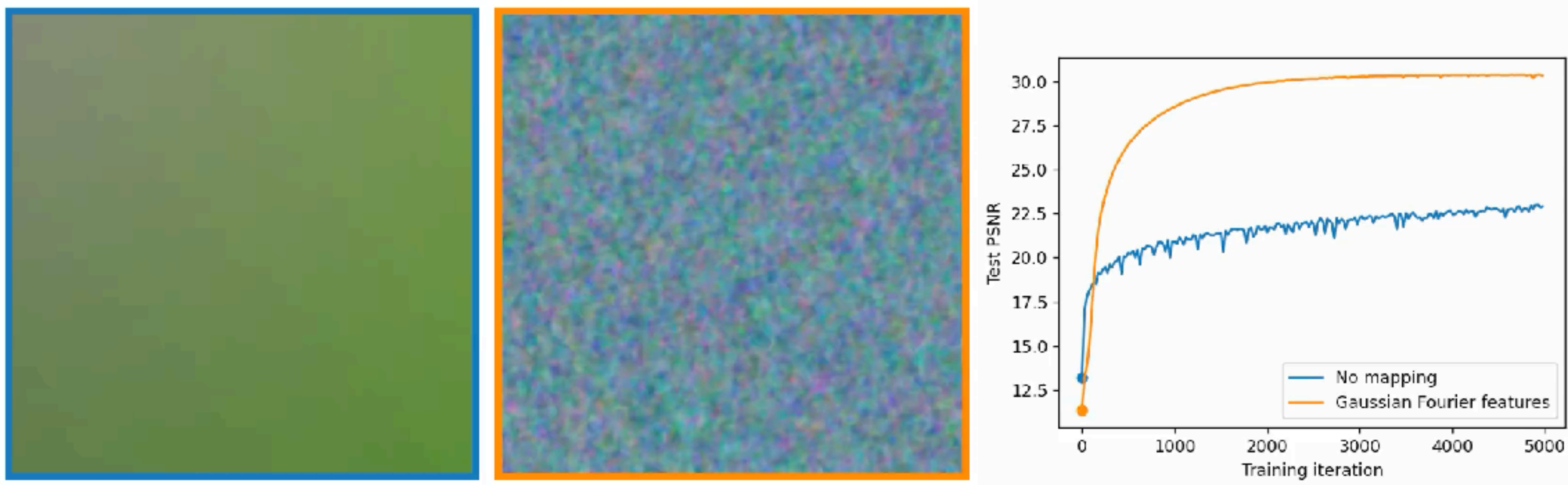
- PEs are widely used in Large Language Models (LLM)



Applications of PE: Implicit functions



- $f : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ modeled by a network (e.g. MLP)
- **Input**: pixel coordinate $\mathbf{v} = [x, y]^\top$
- **Output**: color value $[r, g, b]^\top$



Tancik et al. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. NeurIPS 2020

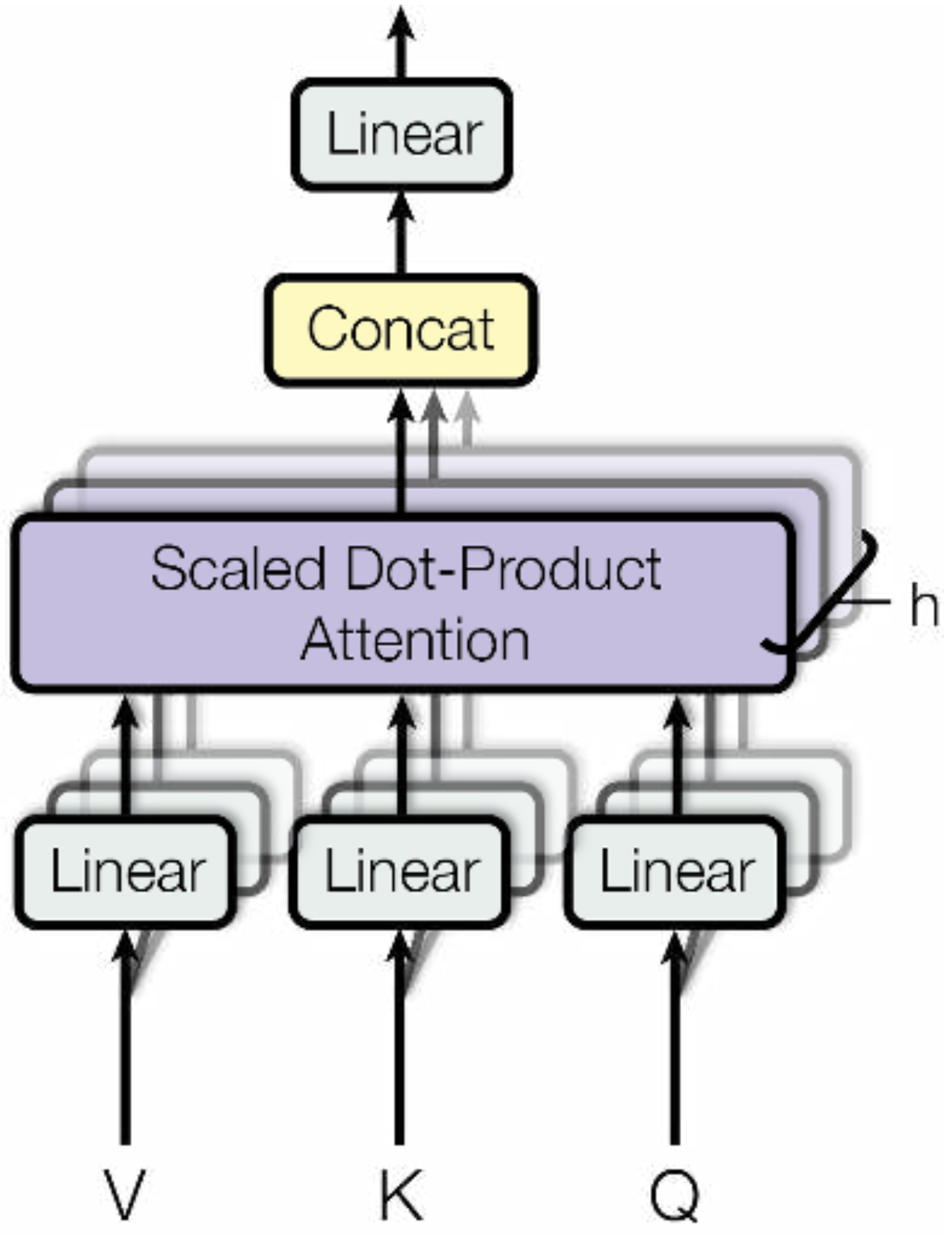
Positional Embeddings - TL;DR

- Positional embeddings are used to break permutation invariance
- Positional embeddings encode location-related information
- Many types of PEs: absolute, sinusoidal, learnable, relative, rotary

The Transformer Architecture

Recap: Multi-Head Attention

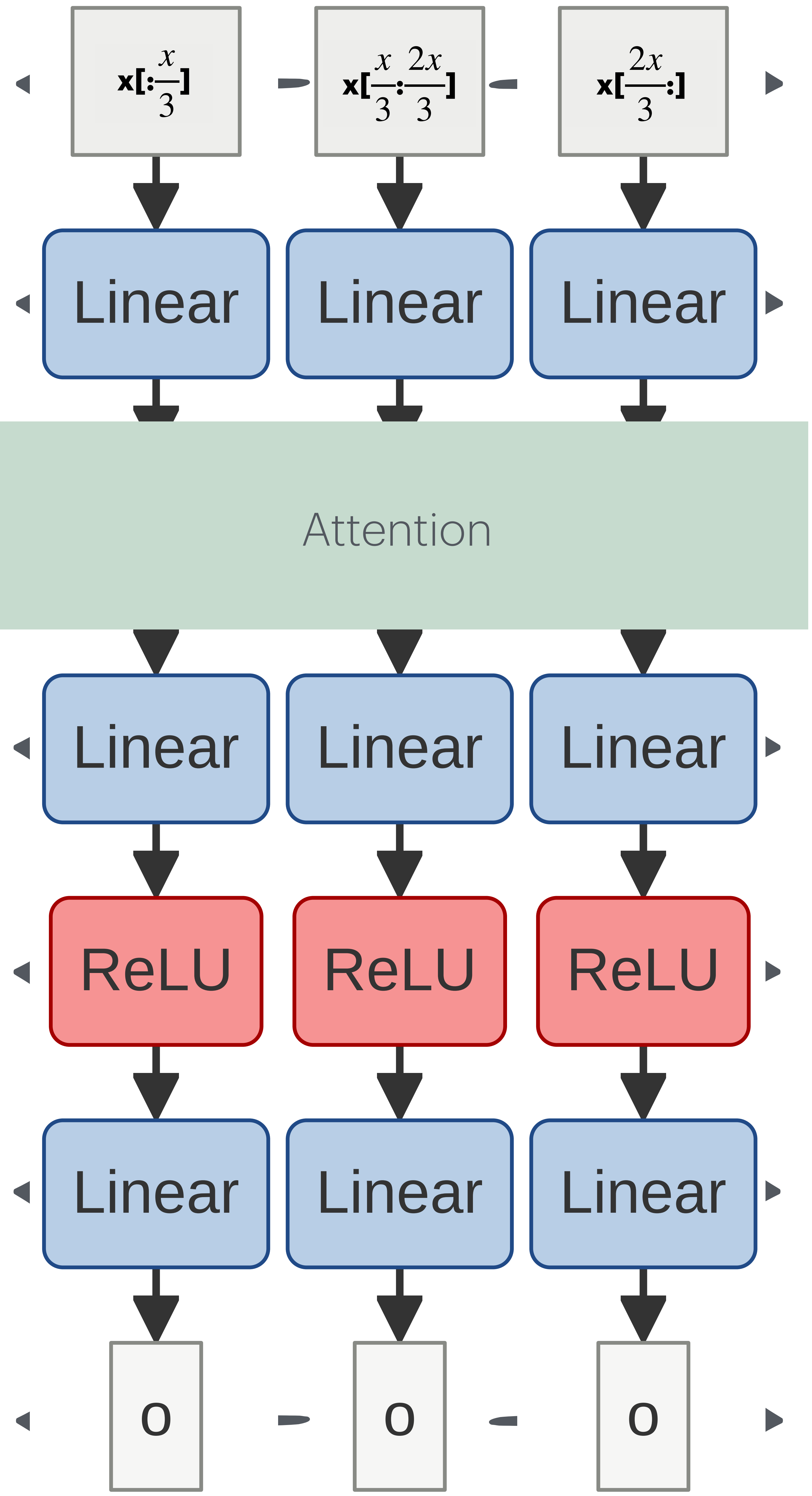
- h heads, each with a set of linear projections
- $$\begin{bmatrix} \text{Attention}(\mathbf{XW}_{Q,1}, \mathbf{XW}_{K,1}, \mathbf{XW}_{V,1}) \\ \vdots \\ \text{Attention}(\mathbf{XW}_{Q,h}, \mathbf{XW}_{K,h}, \mathbf{XW}_{V,h}) \end{bmatrix} W_O$$
- Good at **mixing information** across various inputs / tokens



Recap: Scaling wide

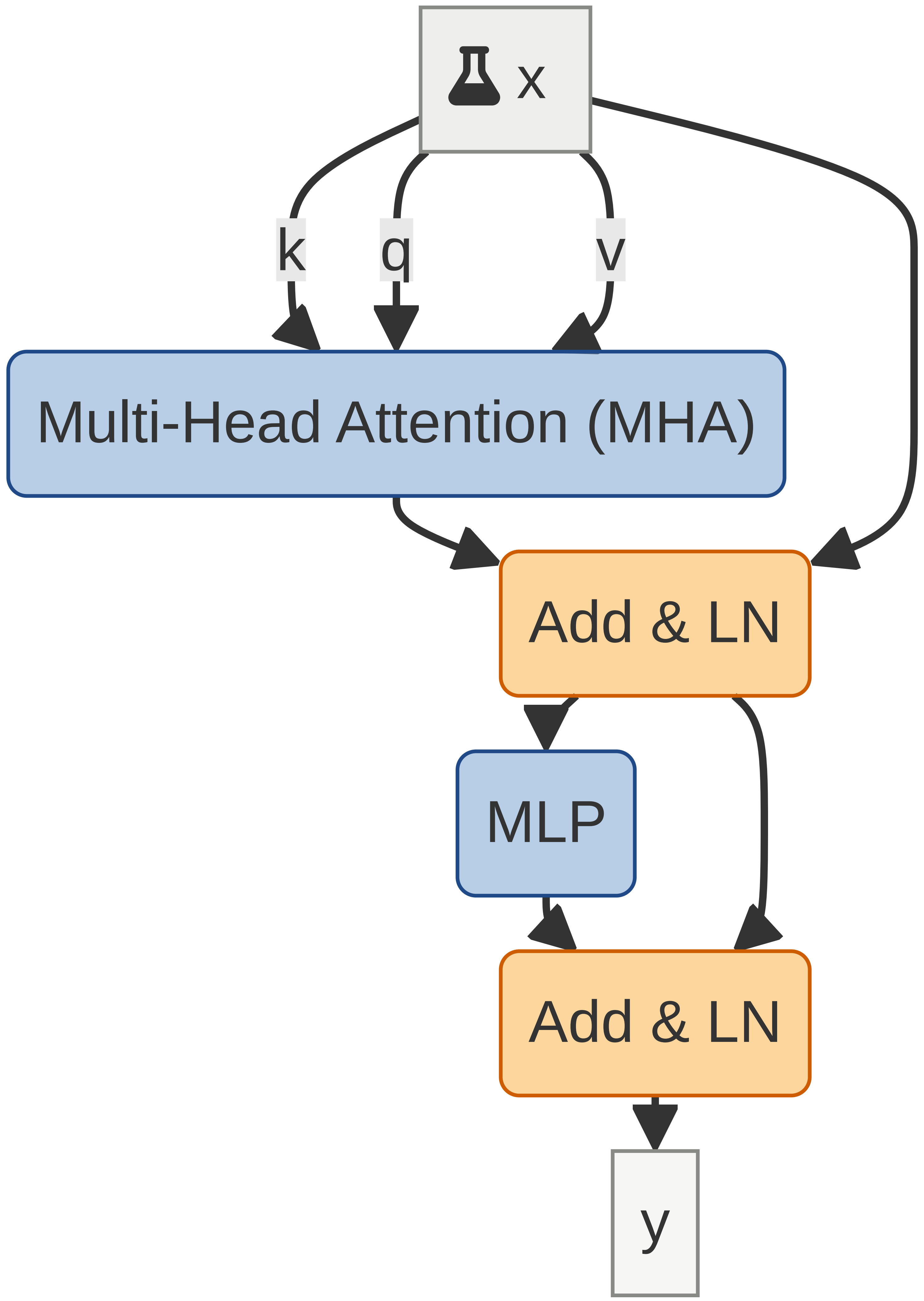
A Simple Solution

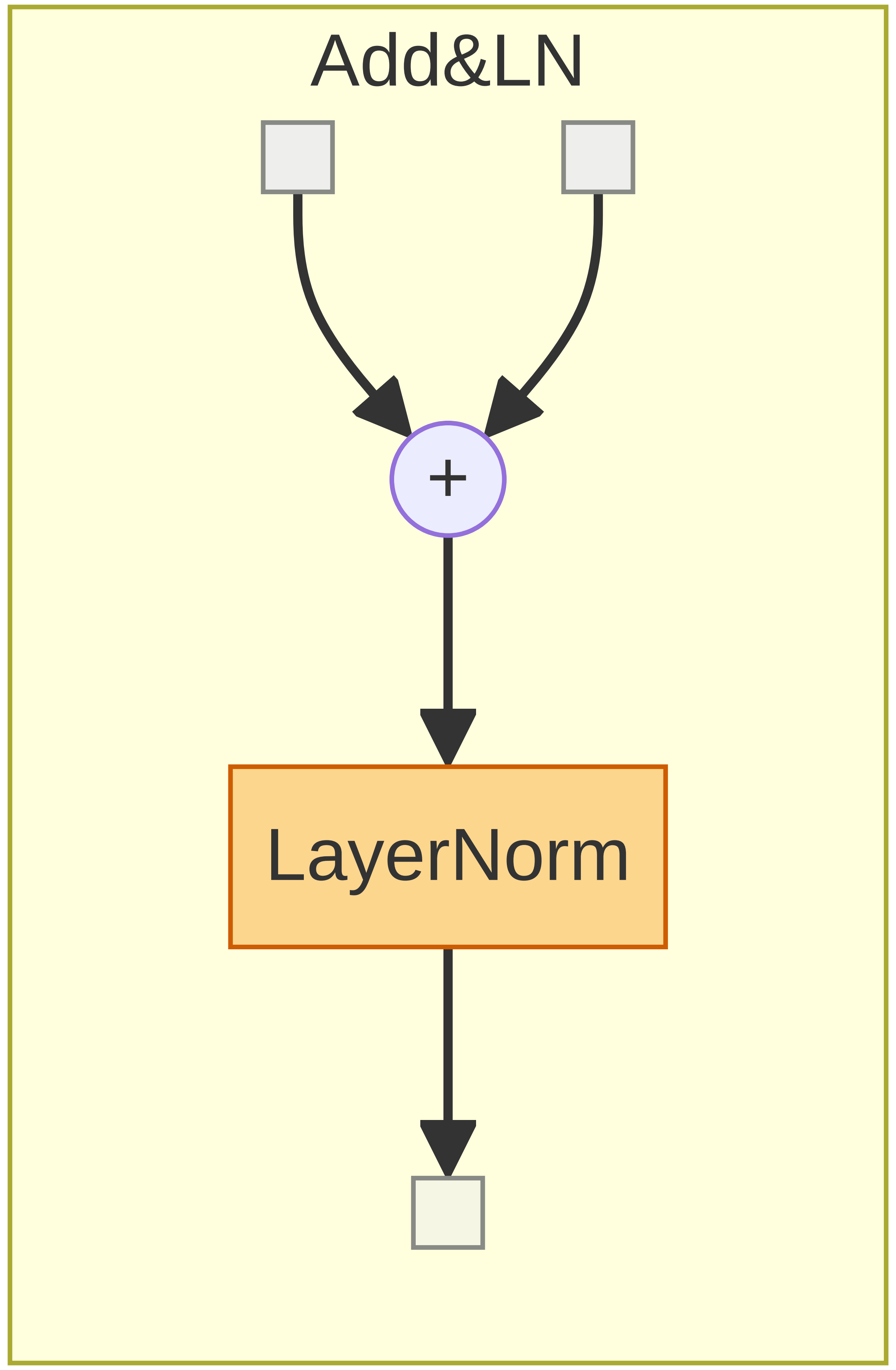
- Token-level MLP
  - Good at processing local information



Transformer Layer

- Attention + MLP + Residual / Normalization

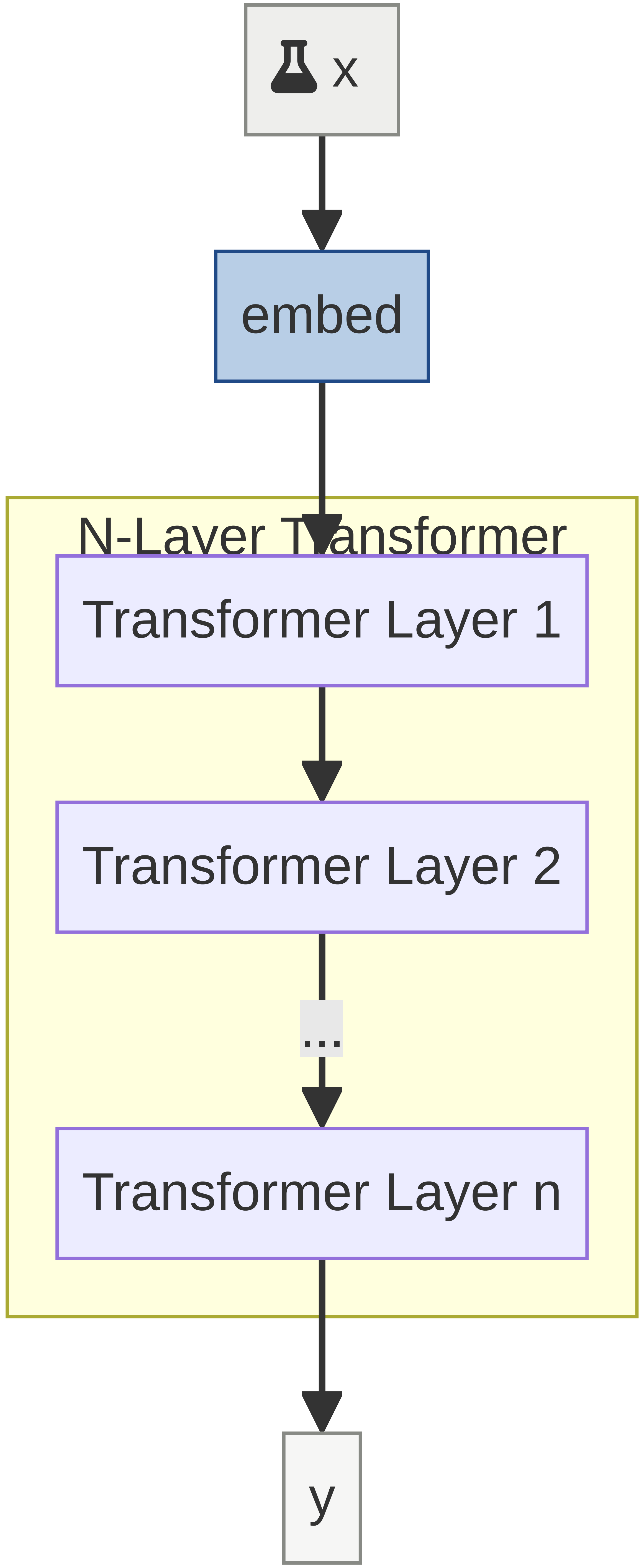




Vaswani, et al. "Attention is all you need." NeurIPS 2017

Transformer

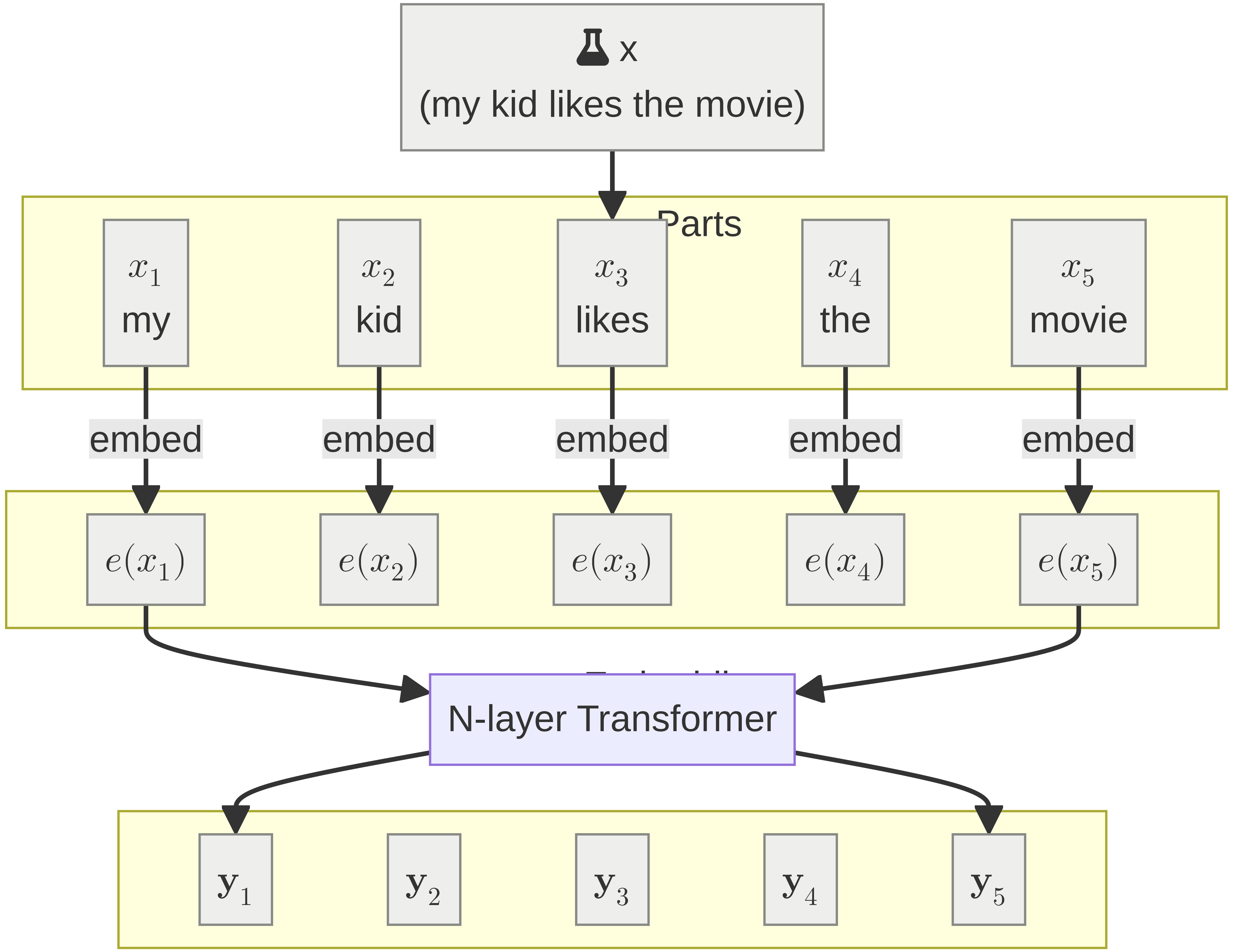
- **Input**: Set of tokens $\{x_i\}$
- **Output**: Set of tokens $\{y_i\}$
- Stack N transformer layers



Vaswani, et al. "Attention is all you need." NeurIPS 2017

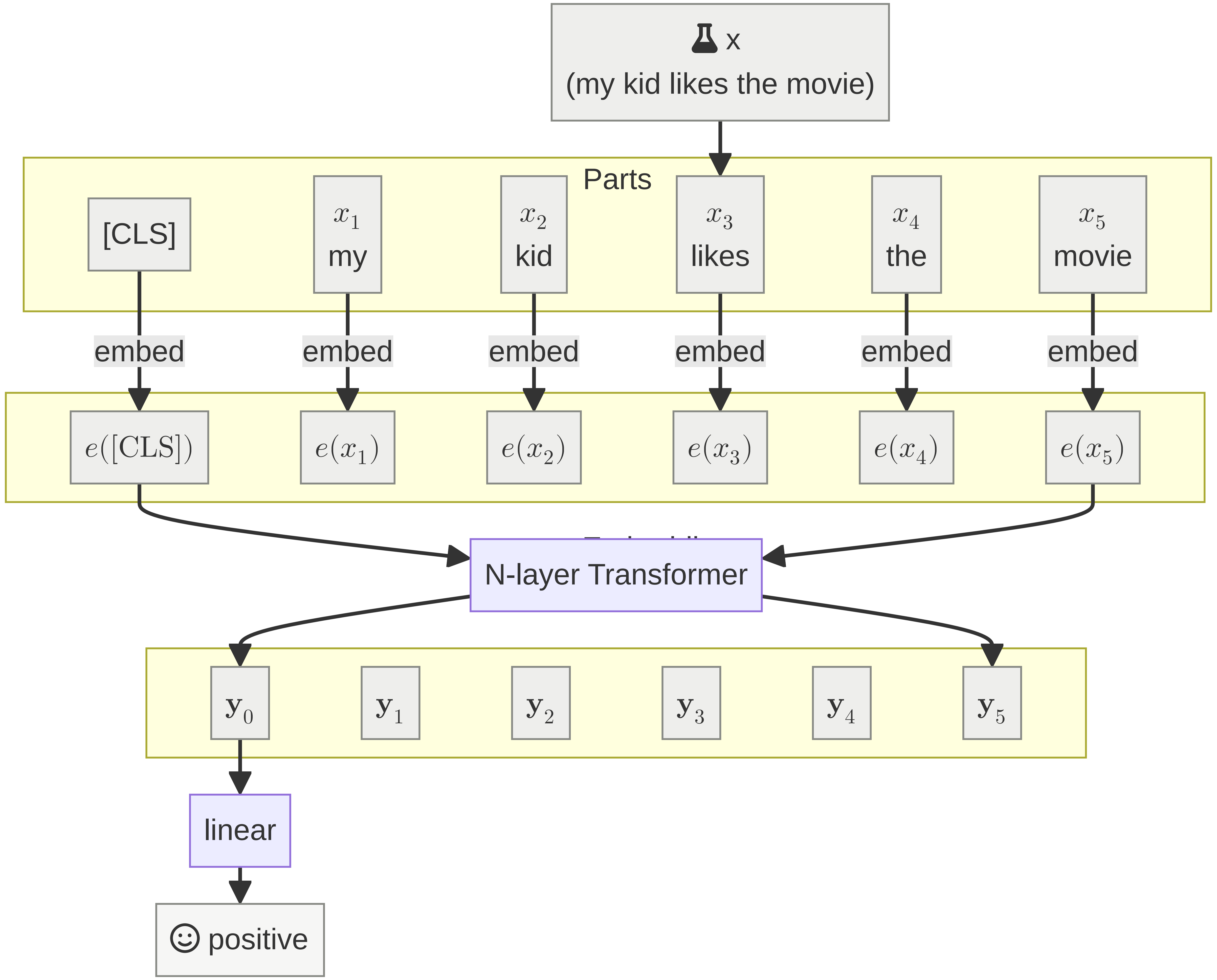
Transformer

- **Input**: Set of tokens $\{x_i\}$
- **Output**: Set of tokens $\{y_i\}$
- Stack N transformer layers



Classification with transformers

- **Input**: Set of tokens $\{x_i\}$
- **Output**: Set of tokens $\{y_i\}$
- Prepend "classification" token



The Transformer Architecture - TL;DR

- Transformer layer = MHA + MLP + LN + residual connection
- A Transformer is a stack of N transformer layers

Applications of transformers

What Else Can We Do With Transformers?

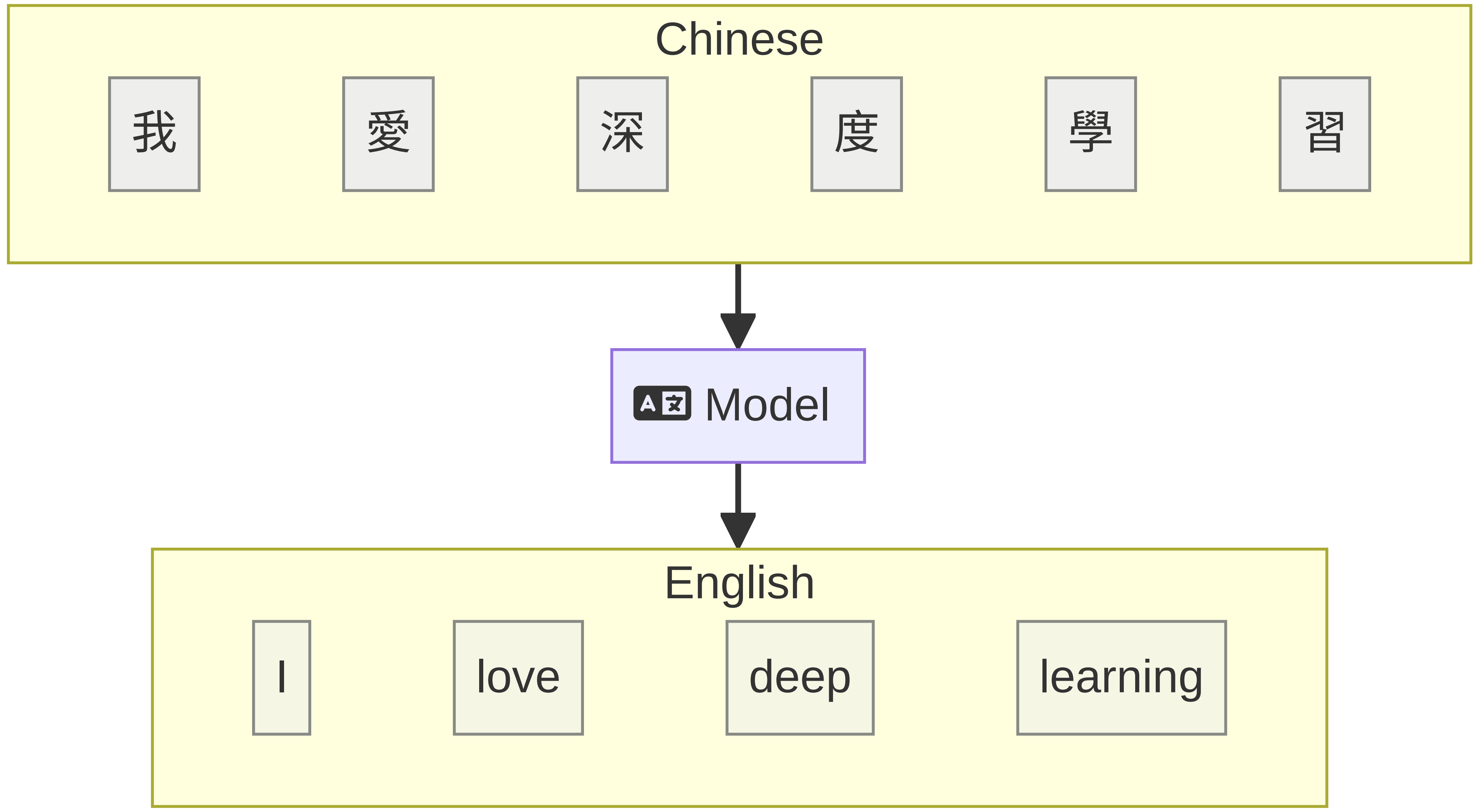
Machine Translation

Input: a sentence in a given language

Output: translation to another language

English: I love deep learning

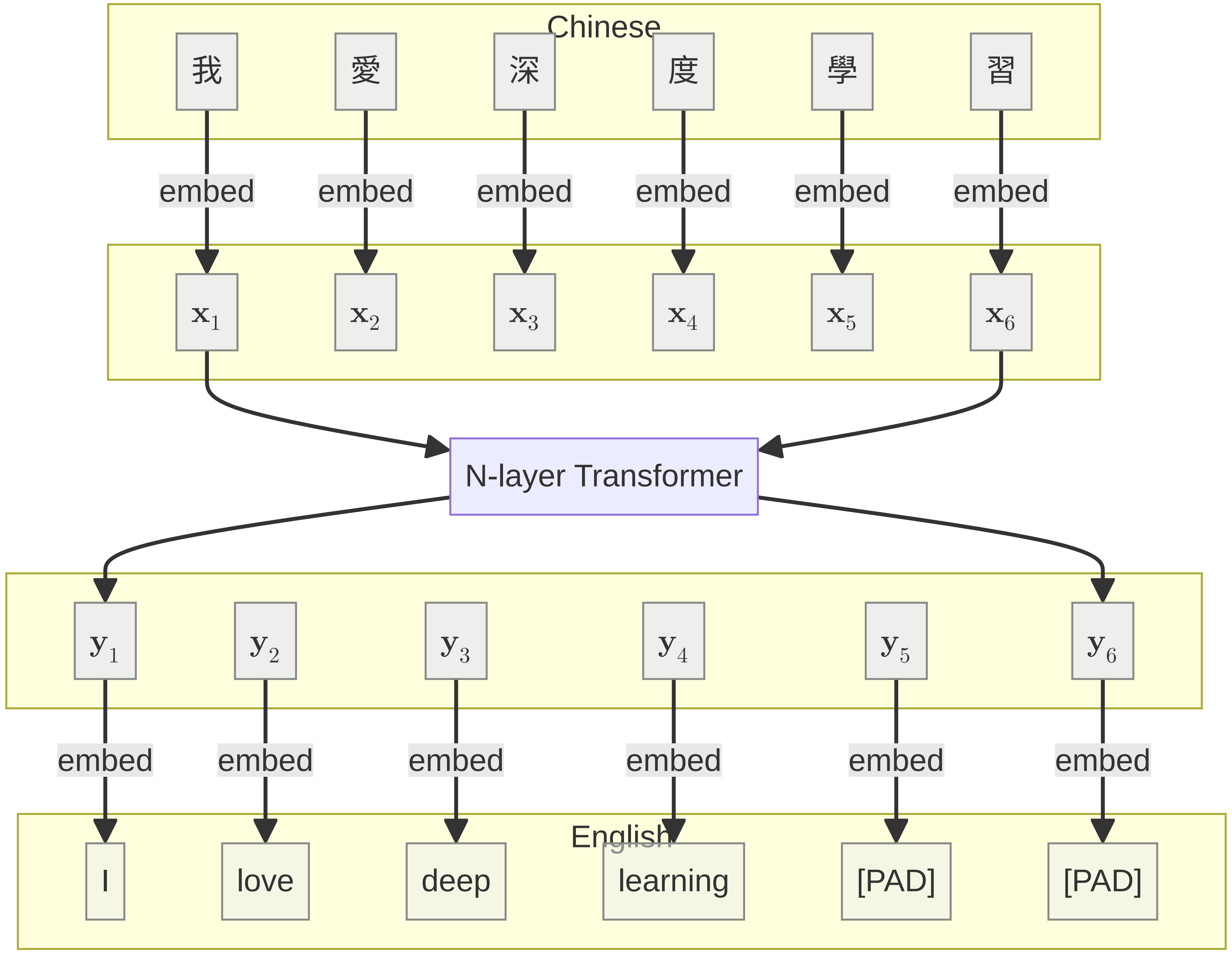
Chinese: 我愛深度學習



Challenges of Machine Translation

⚠ Length of input != length of output

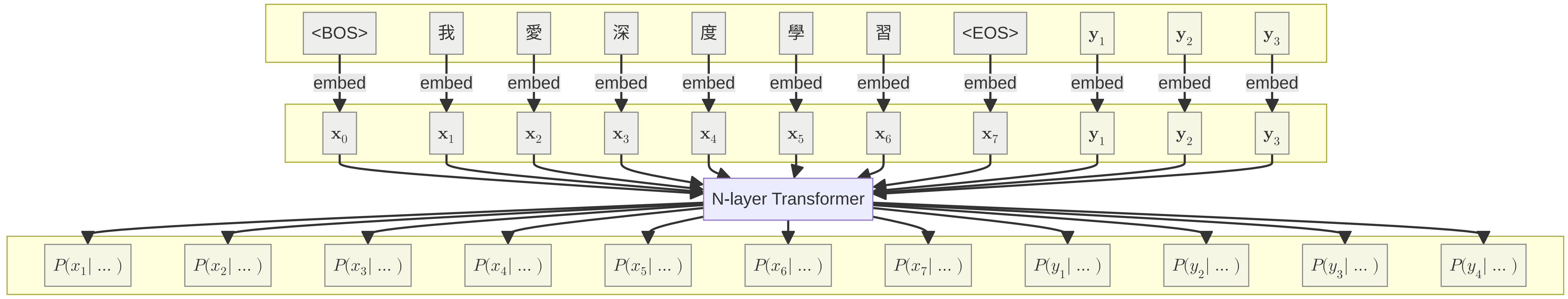
⚠ Hard to produce coherent output tokens simultaneously



Auto-Regressive Prediction

Predict one token (word) at a time

1. $P(\tilde{\mathbf{y}}_1 | \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6)$
2. $P(\tilde{\mathbf{y}}_2 | \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \tilde{\mathbf{y}}_1)$
3. $P(\tilde{\mathbf{y}}_3 | \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_2)$
4. $P(\tilde{\mathbf{y}}_4 | \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_2, \tilde{\mathbf{y}}_3)$



...

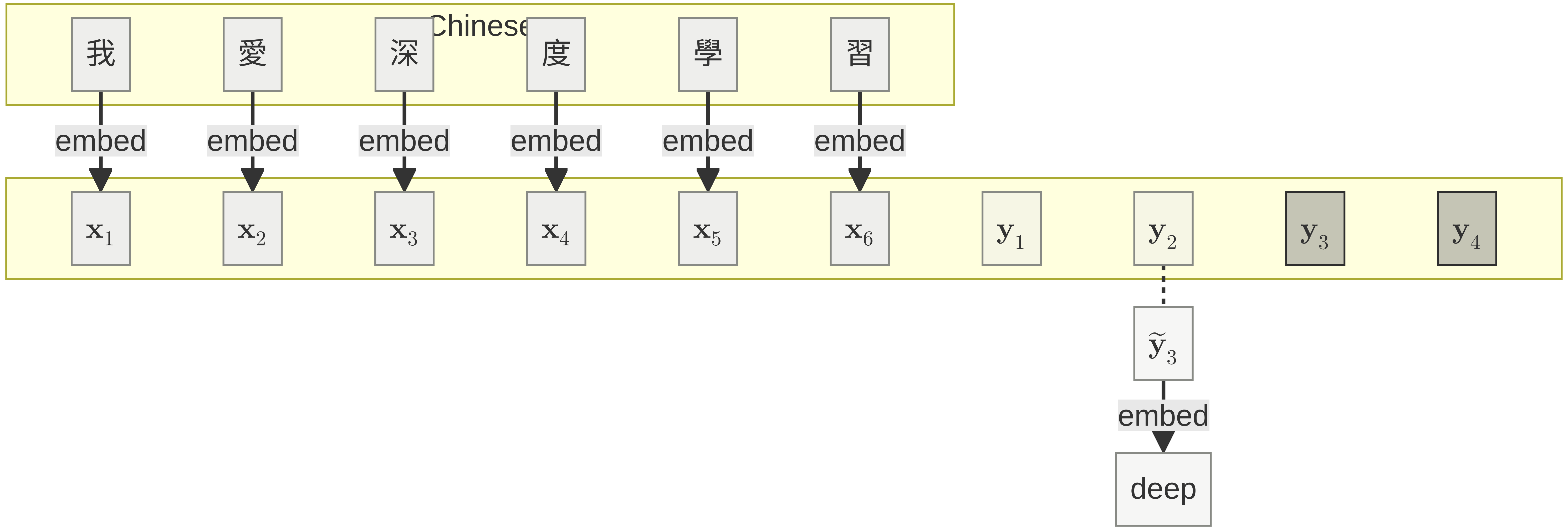
Until $\tilde{\mathbf{y}}_t$ hits an end-of-sequence (EOS) token

Here, $p(\tilde{\mathbf{y}}_t | \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_2, \tilde{\mathbf{y}}_{t-1})$ is modeled by an N-layer Transformer

Masked Attention

Output sequence is offset by one compared to input

Model can easily cheat by looking at future tokens

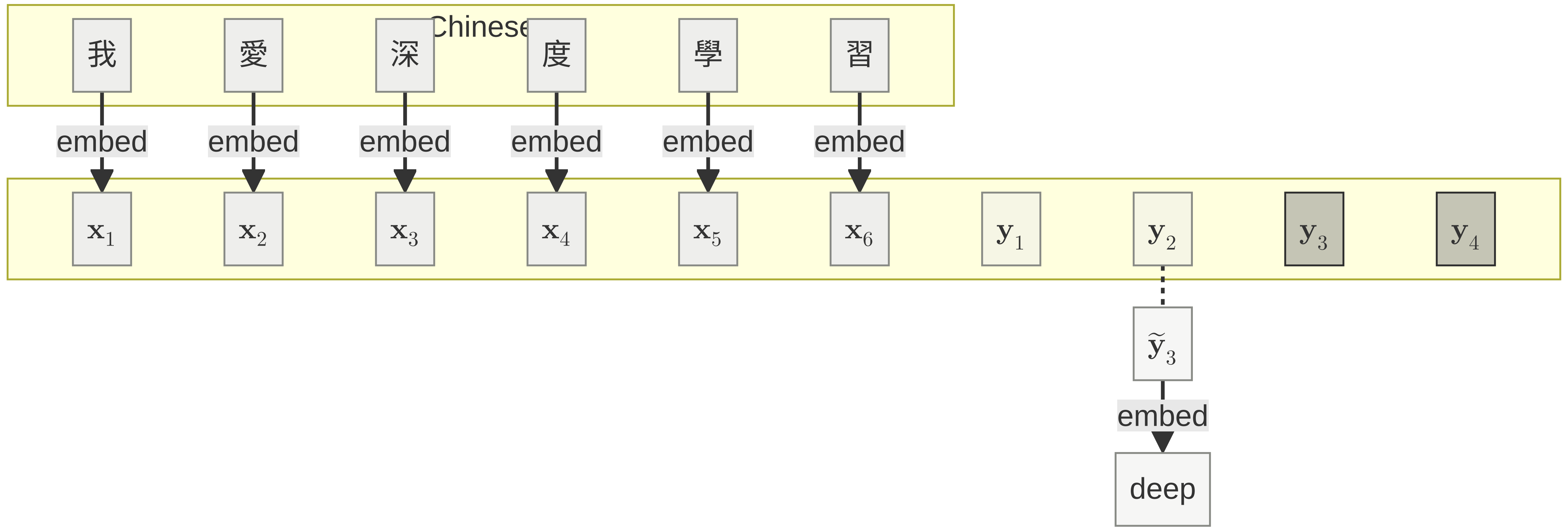


Masked Attention

$$\begin{aligned}\mathbf{O} &= \mathrm{MaskedAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \\ &= \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{C}} + \mathbf{M}\right)\mathbf{V}\end{aligned}$$

where the mask $\mathbf{M}$ is defined by

$$\mathbf{M} = \begin{bmatrix} 0 & -\infty & \cdots & -\infty \\ 0 & 0 & \cdots & -\infty \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$



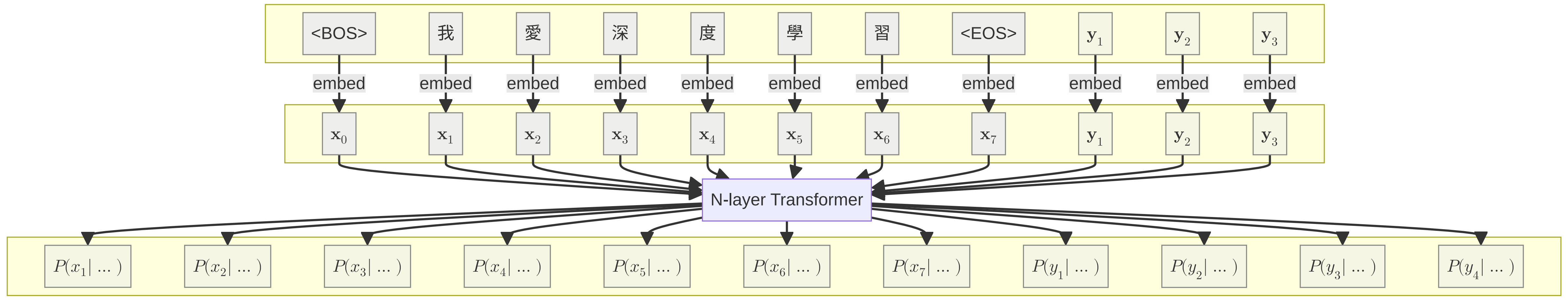
Auto-Regressive Prediction

Test Time:

Sample one token (word) at a time

$$P(\tilde{\mathbf{y}}_t | \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_2, \cdots, \tilde{\mathbf{y}}_{t-1})$$

until $\tilde{\mathbf{y}}_t$ hits an end-of-sentence (`<EOS>`) token



☹ Very slow during training

Teacher Forcing

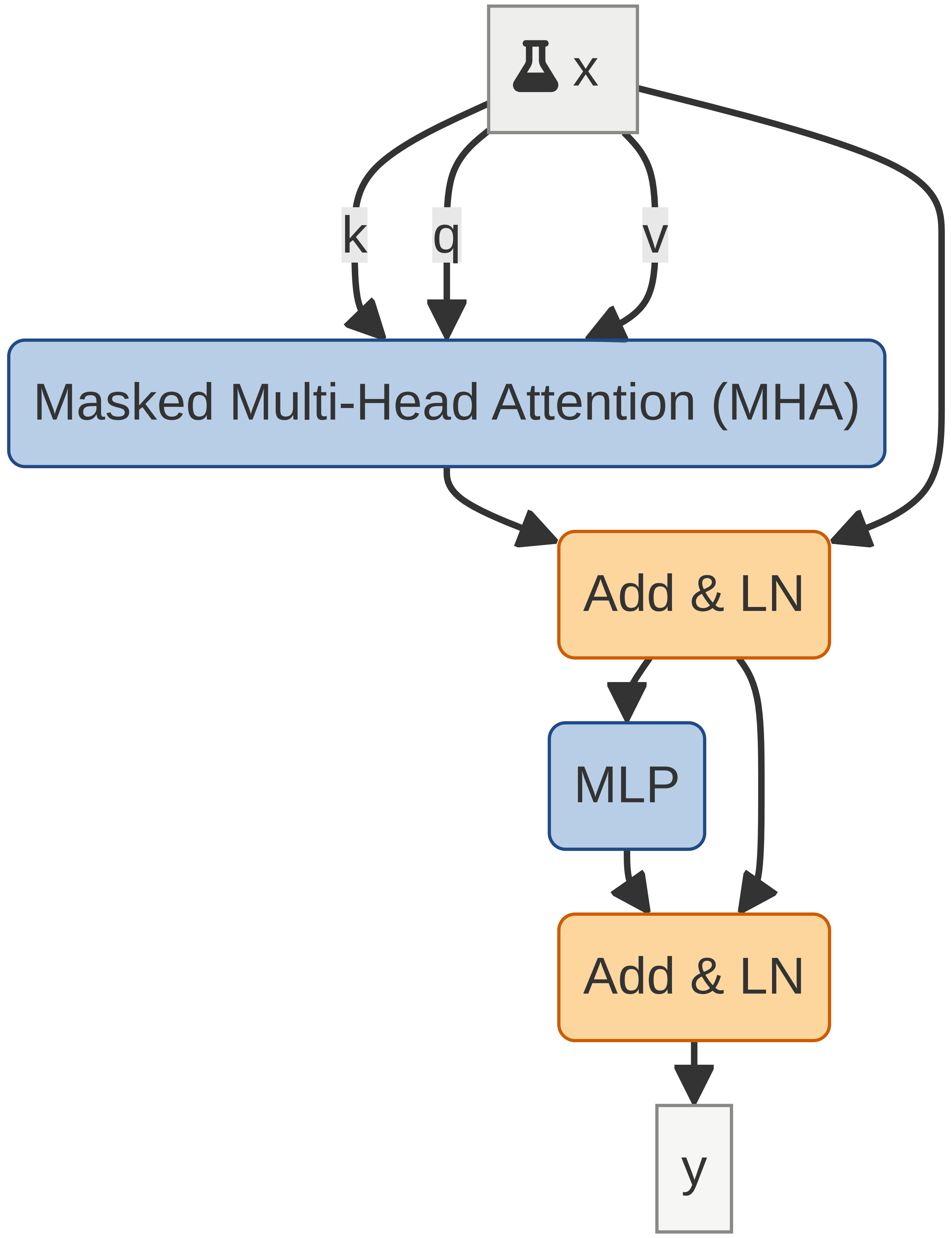
Fast training

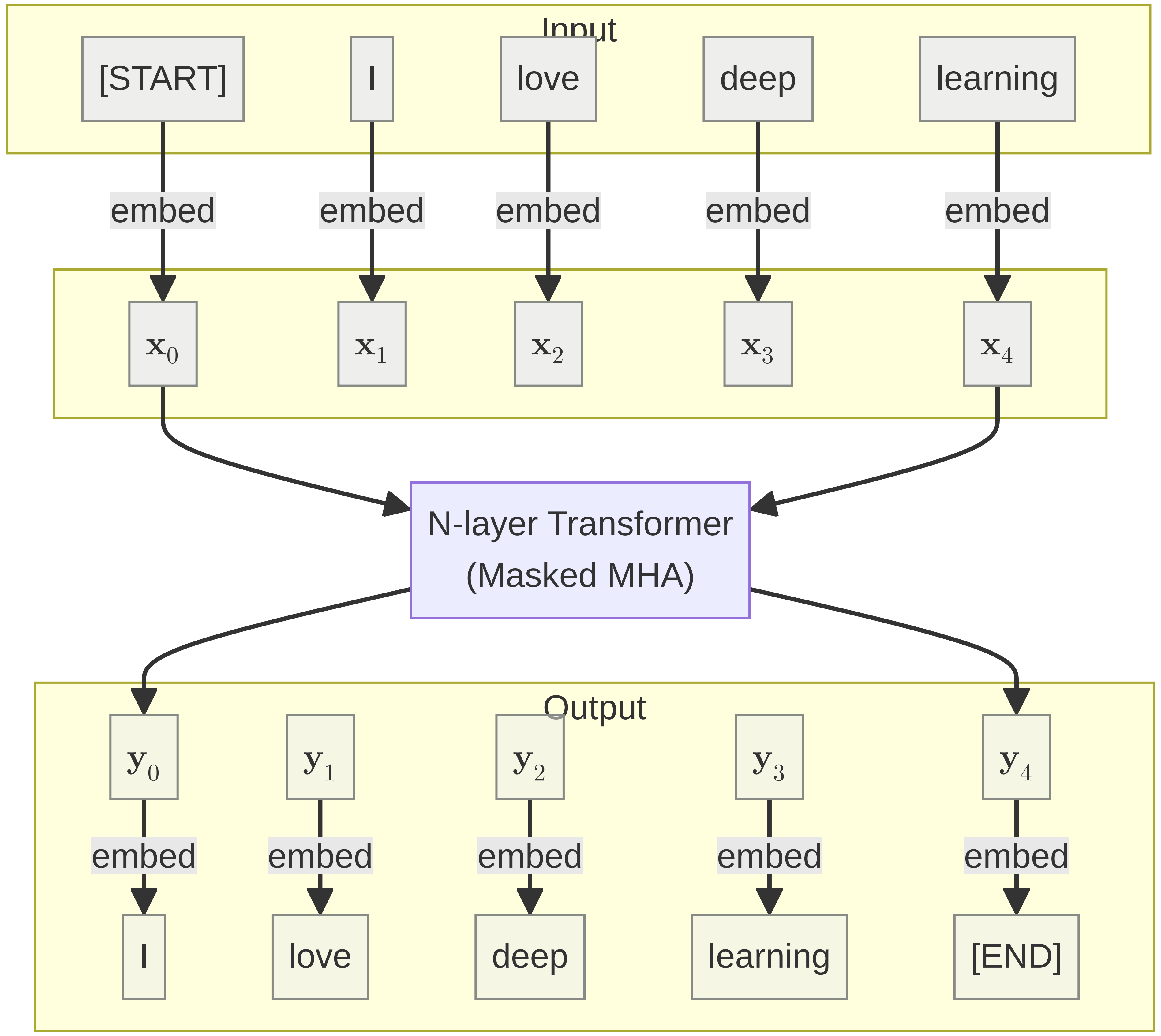
- Condition on ground truth inputs
- Different from what is seen during generation (sampling vs ground truth)
- Fine in practice
- Parallel training of all predictions

$$P(\tilde{\mathbf{y}}_t | \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \mathbf{y}_0, \mathbf{y}_1, \cdots, \mathbf{y}_{t-1})$$

$\mathbf{y}_0$ is a special end-of-sentence (`<EOS>` = start-of-translation) token

Transformer Layer With Masked Attention





Types of Transformers

Decoder-only

Masked auto-regressive prediction

Encoder-only

No prediction, just understanding

Encoder-Decoder

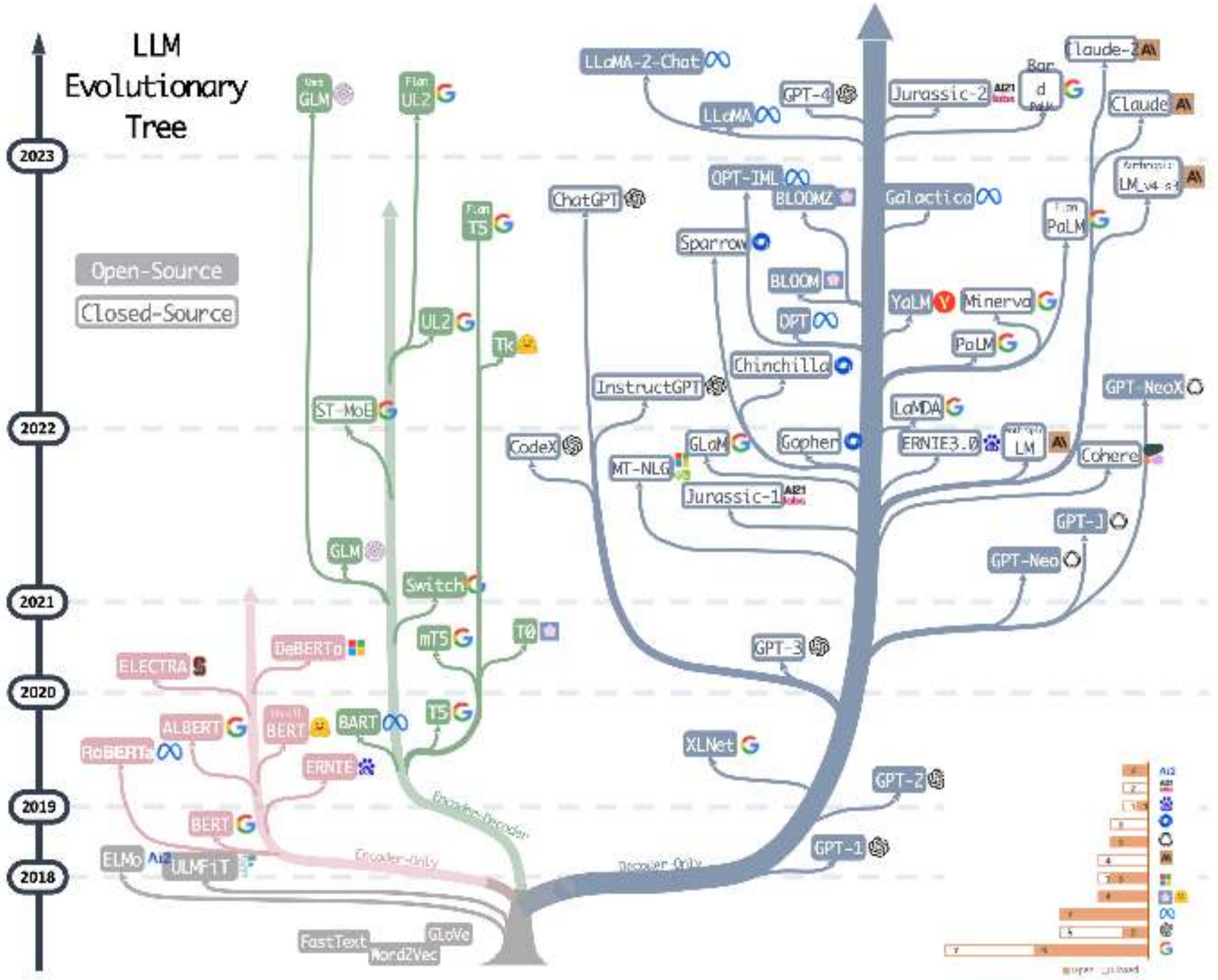


Image credit https://github.com/Mooler0410/LLMsPracticalGuide ↩️

Types of Tokens

Tokens

- words or sub-words (tokenization)
- visual (e.g. image patches)
- discrete or continuous

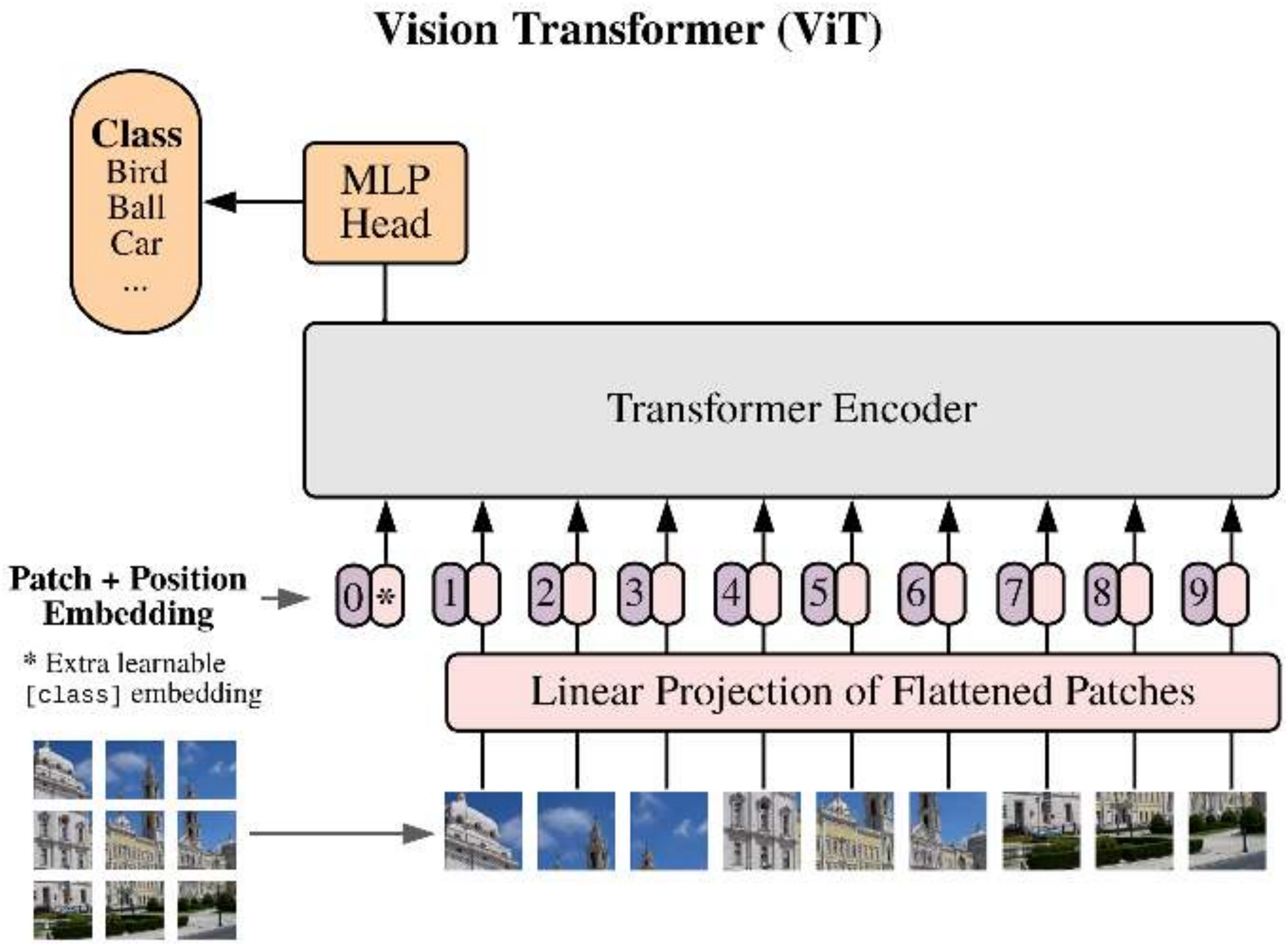


Image credit: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale ↩️

Applications of Transformers - TL;DR

- Transformers are suitable language models, vision model, audio models, etc
- Auto-regressive next word prediction
- Efficient parallel training through teacher forcing
- Transformers process many forms of tokens