Intuition Behind Self-Attention

Attending to the most important parts of an input.

Intuition Behind Self-Attention

Attending to the most important parts of an input.

1. Identify which parts to attend to
2. Extract the features with high attention

Intuition Behind Self-Attention

Attending to the most important parts of an input.

1. Identify which parts to attend to
2. Extract the features with high attention

Similar to a search problem!

Understanding Attention with Search

Query (Q)

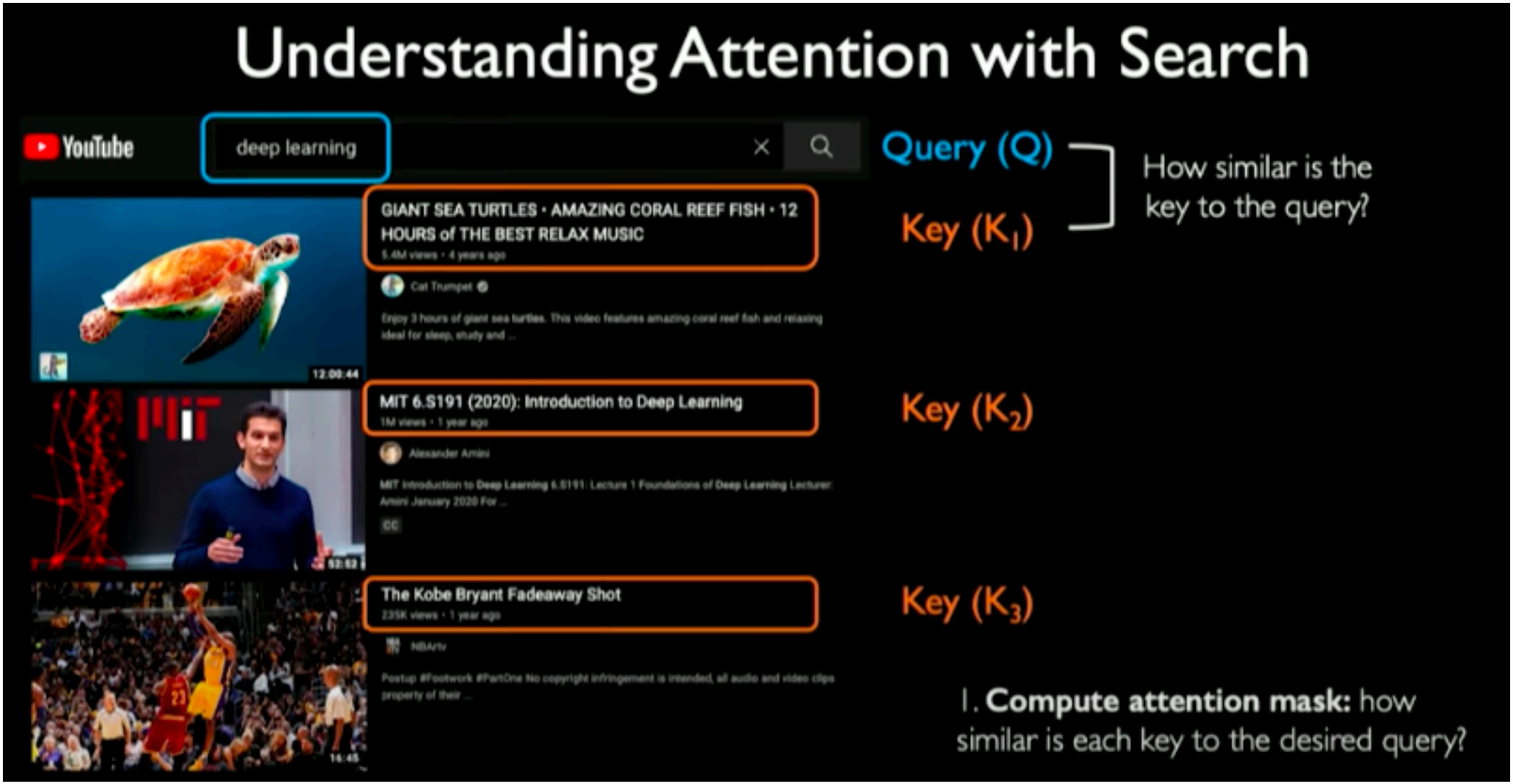
Key (K_1)

Key (K_2)

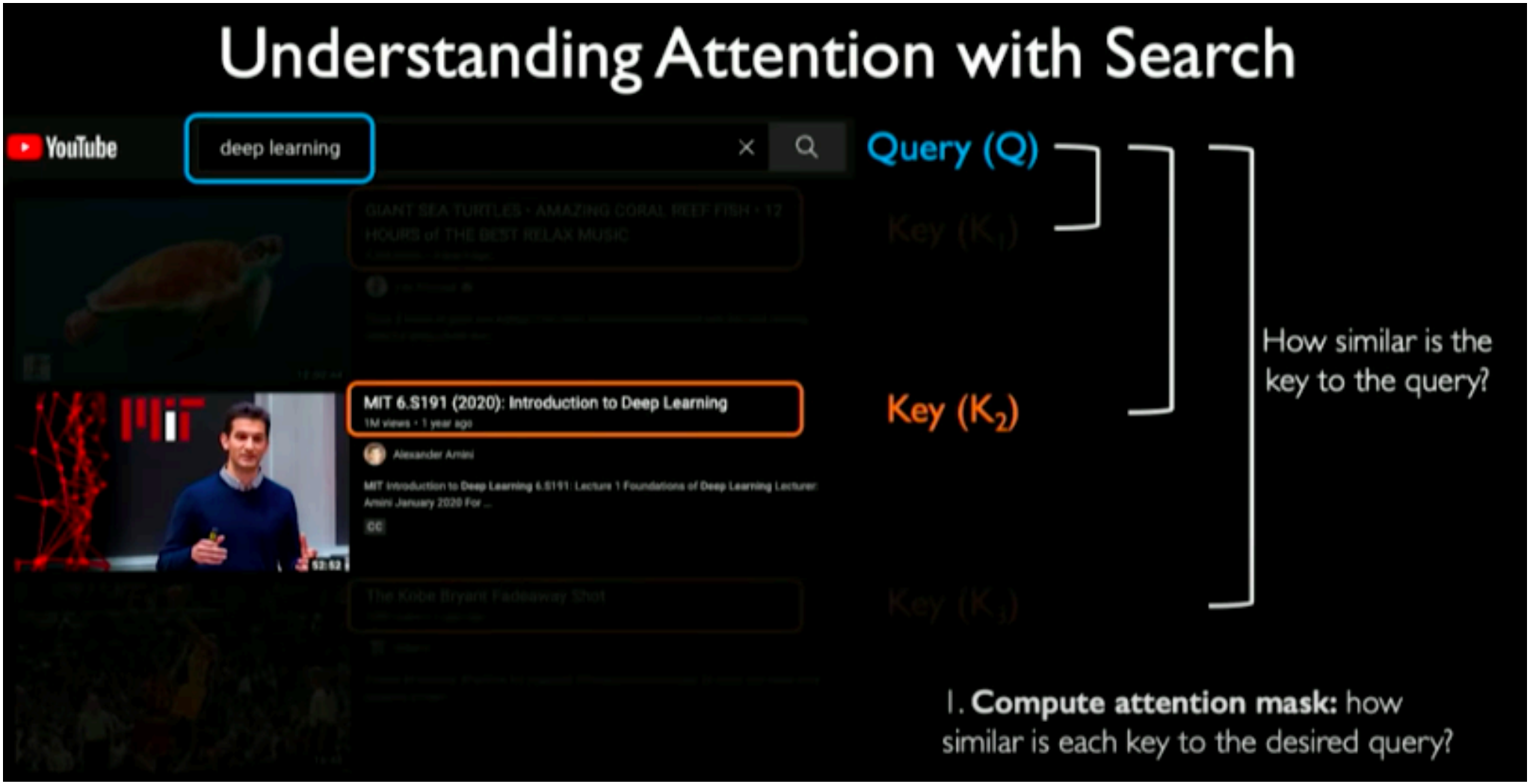
Key (K_3)

1. **Compute attention mask:** how similar is each key to the desired query?

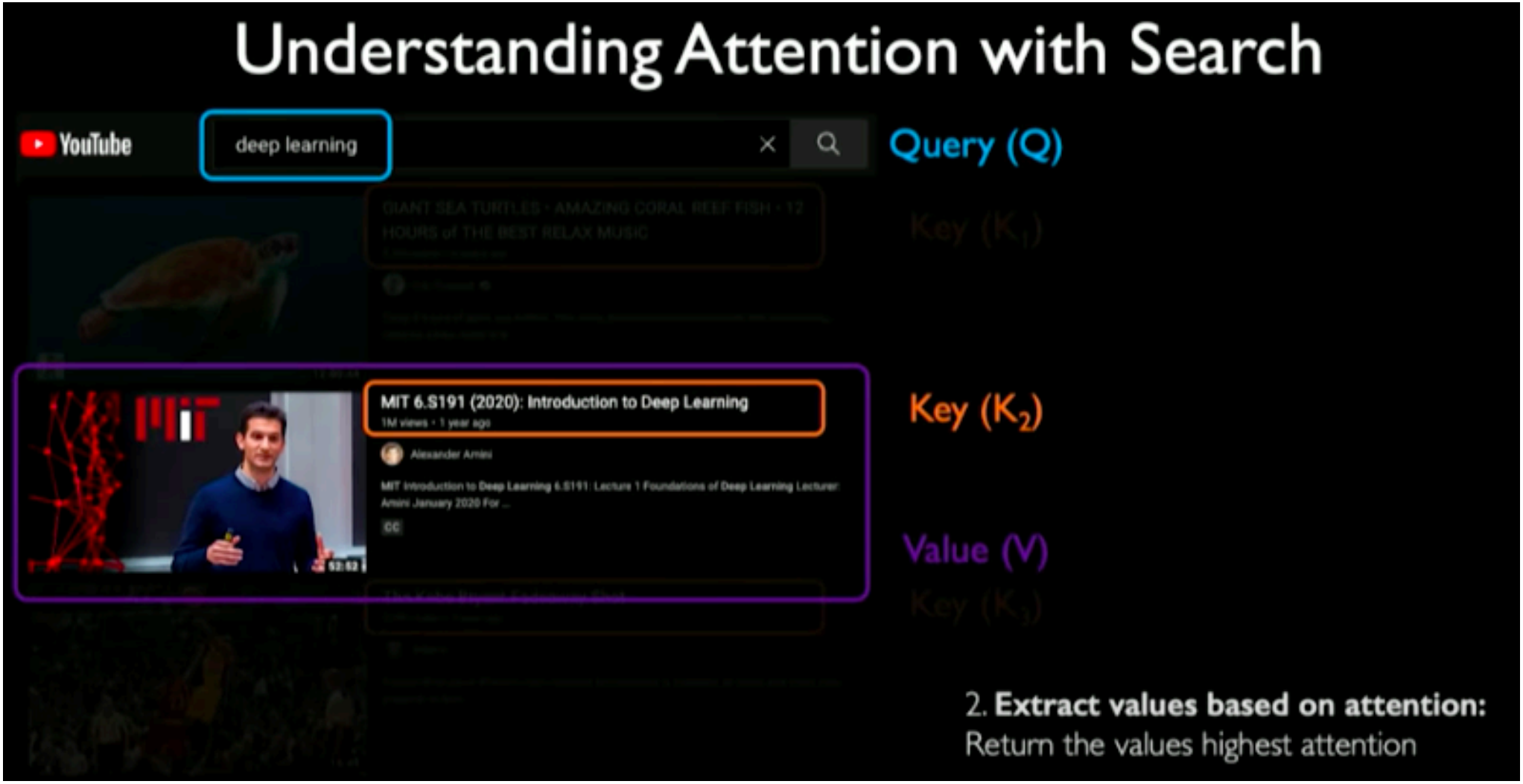
Understanding Attention with Search



Understanding Attention with Search



Understanding Attention with Search



A Sequence Modeling Problem: Predict the Next Word

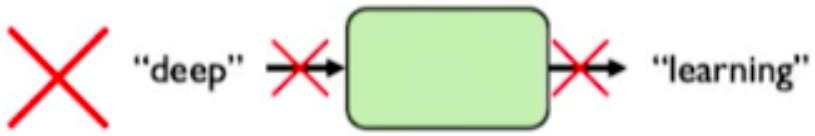
"This morning I took my cat for a walk."

given these words

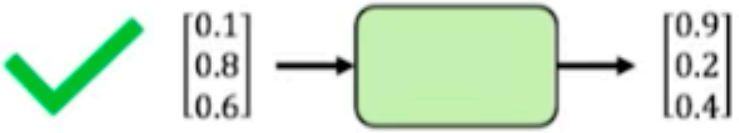
predict the next word

Representing Language to a Neural Network

Encoding Language for a Neural Network

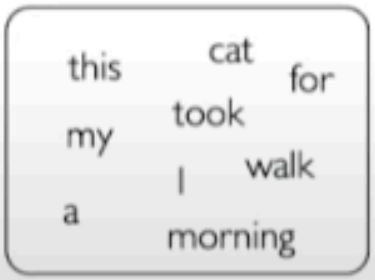


Neural networks cannot interpret words

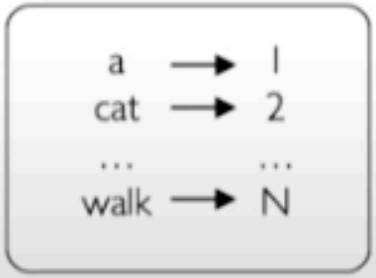


Neural networks require numerical inputs

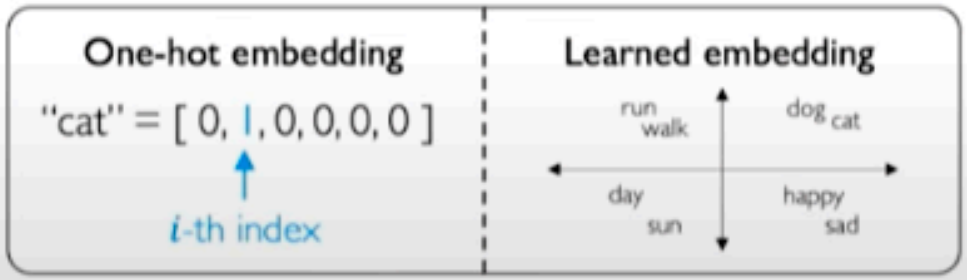
Embedding: transform indexes into a vector of fixed size.



1. Vocabulary:
Corpus of words



2. Indexing:
Word to index

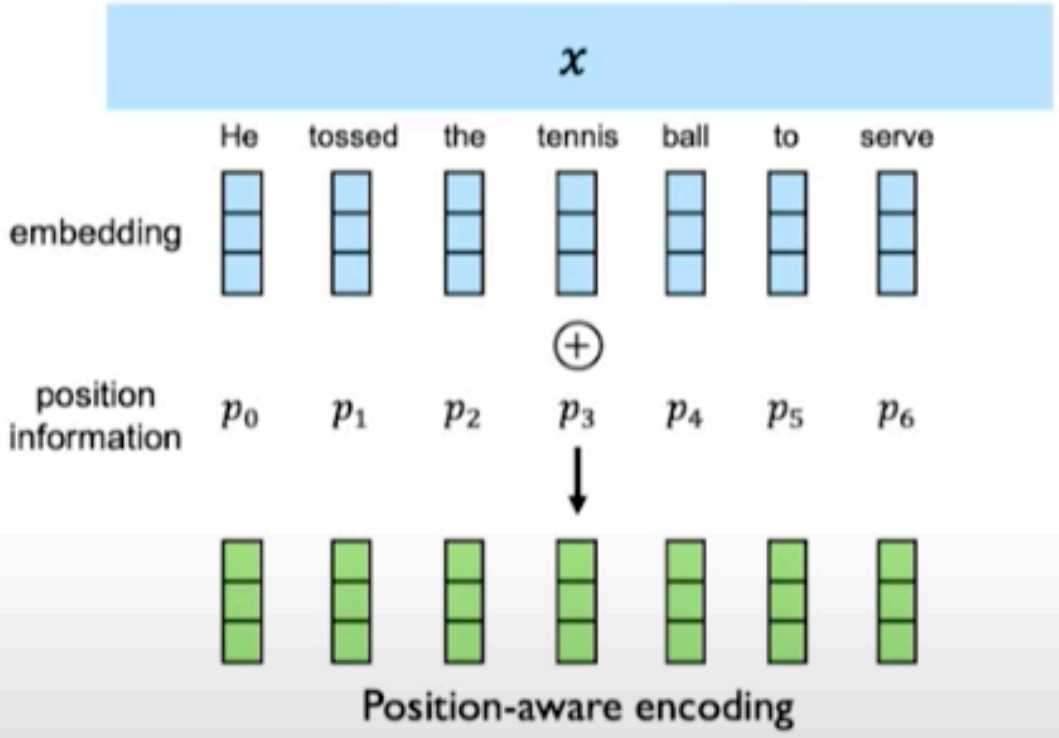


3. Embedding:
Index to fixed-sized vector

Learning Self-Attention with Neural Networks

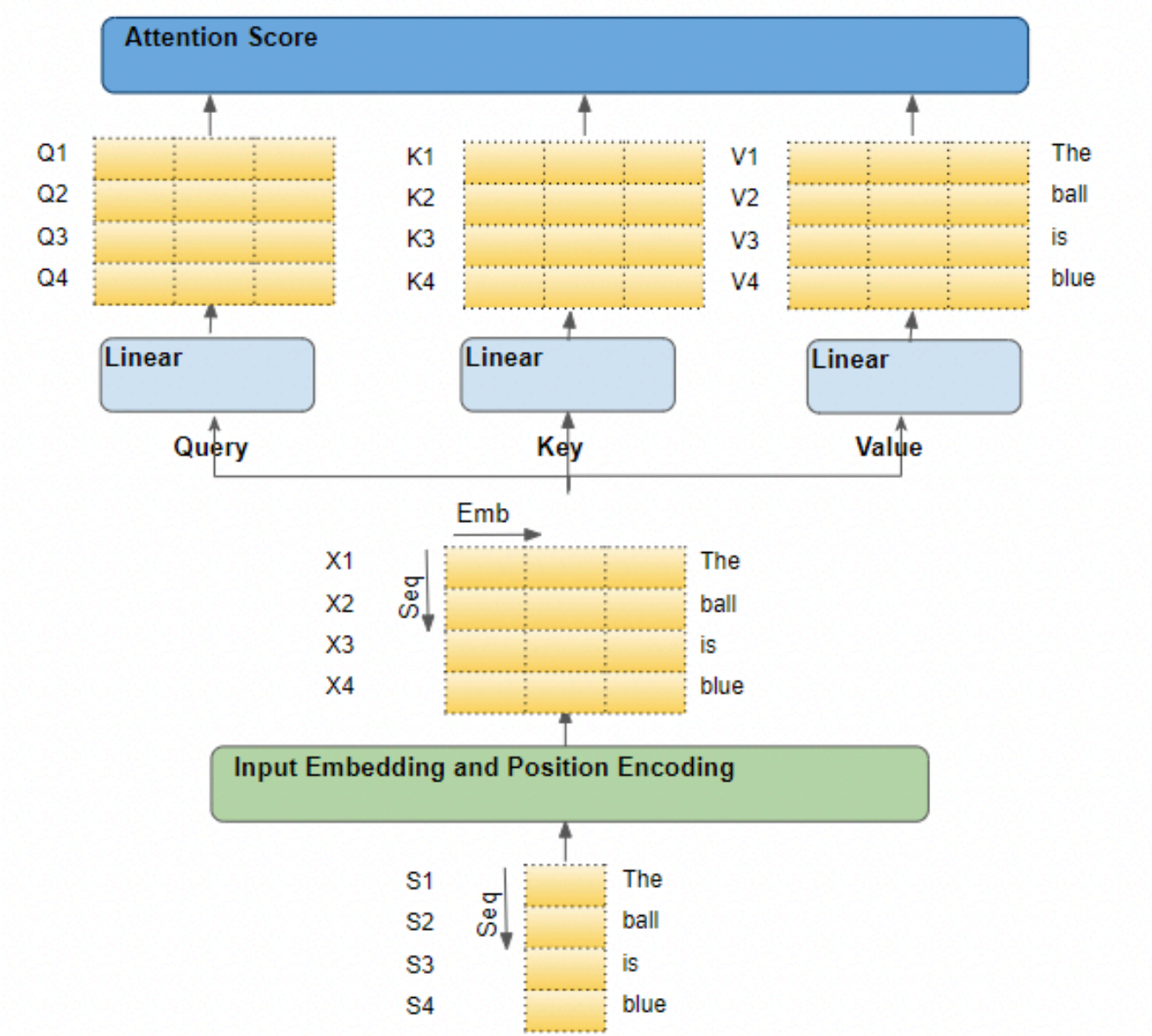
Goal: identify and attend to most important features in input.

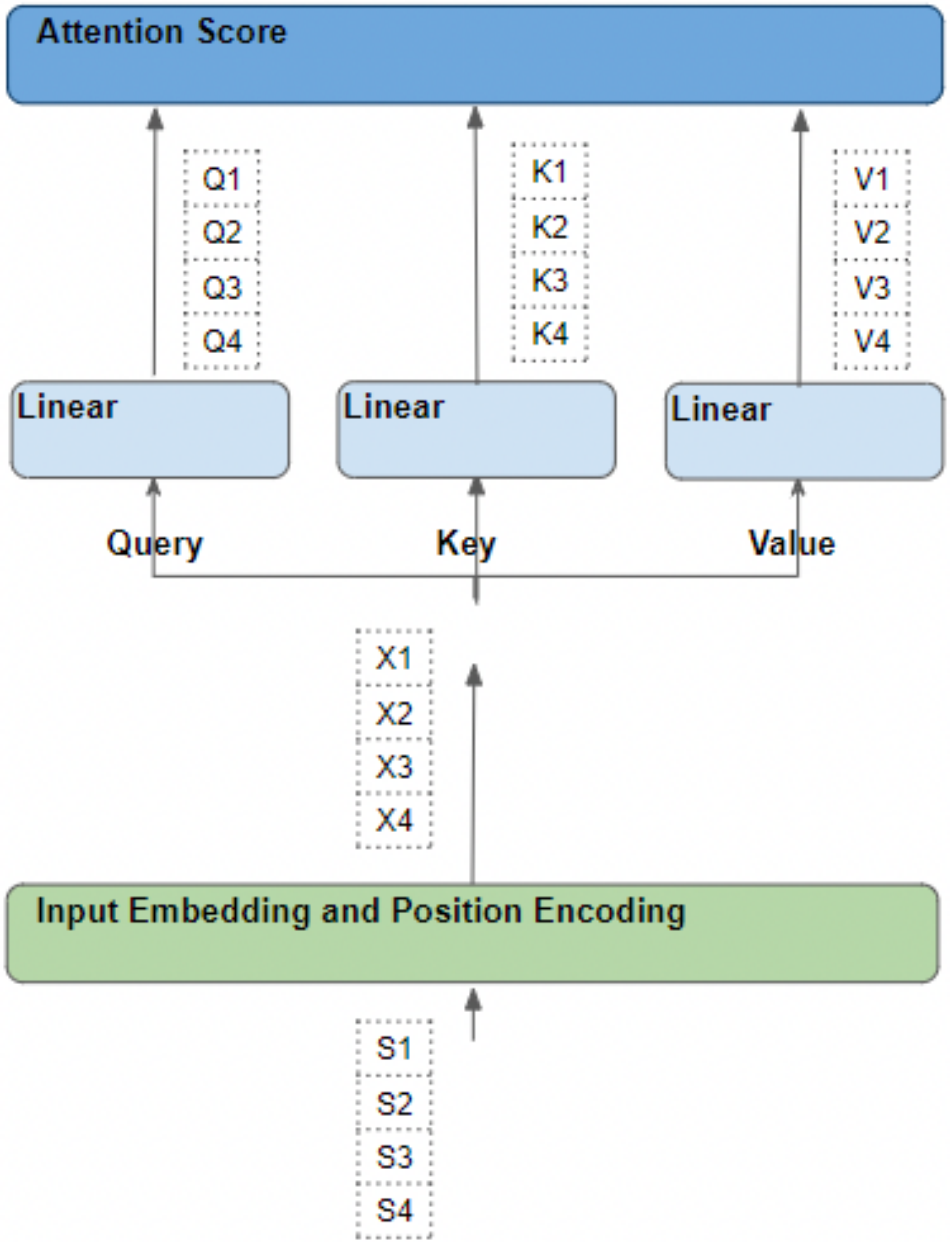
1. Encode **position** information

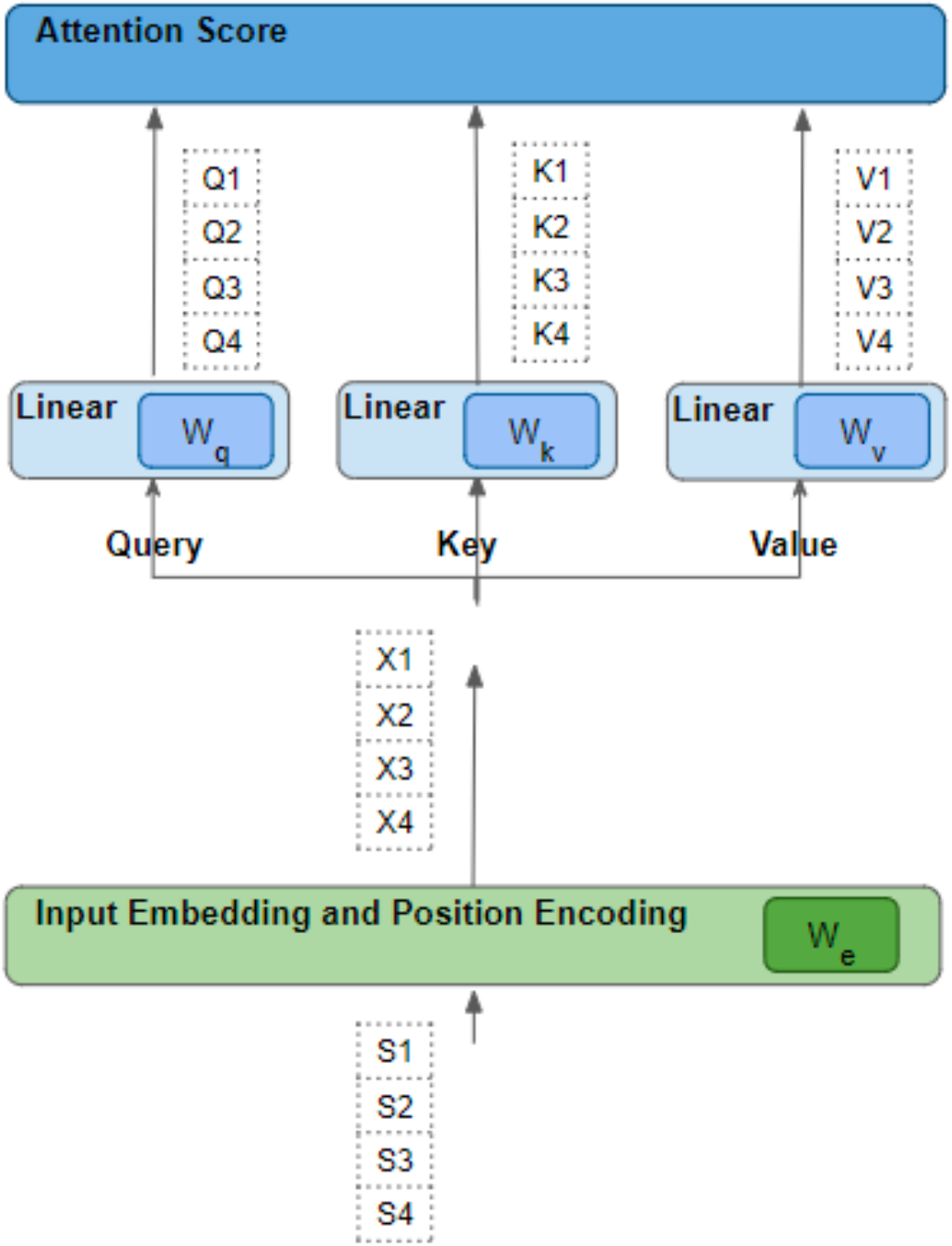


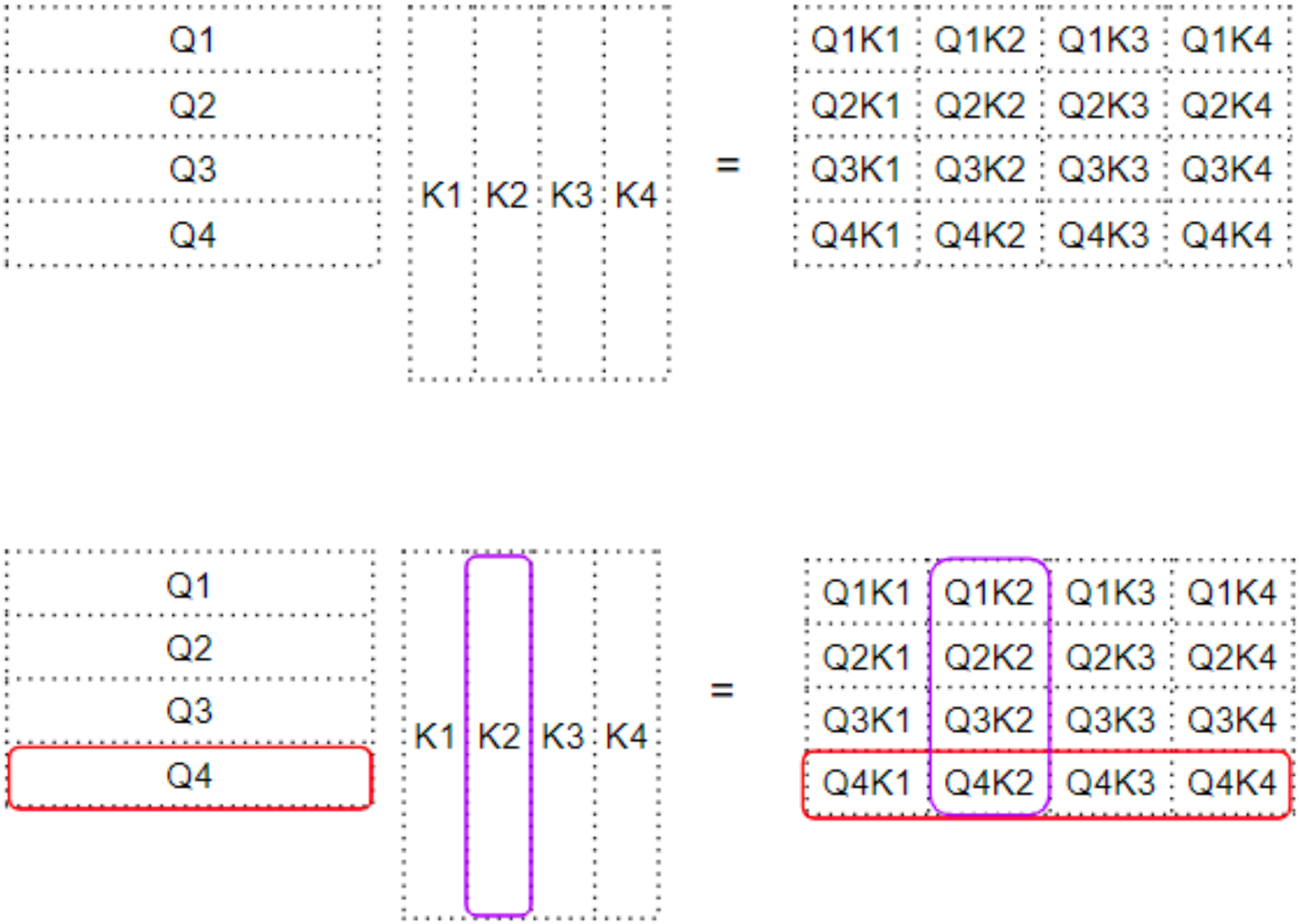
Data is fed in all at once! Need to encode position information to understand order.

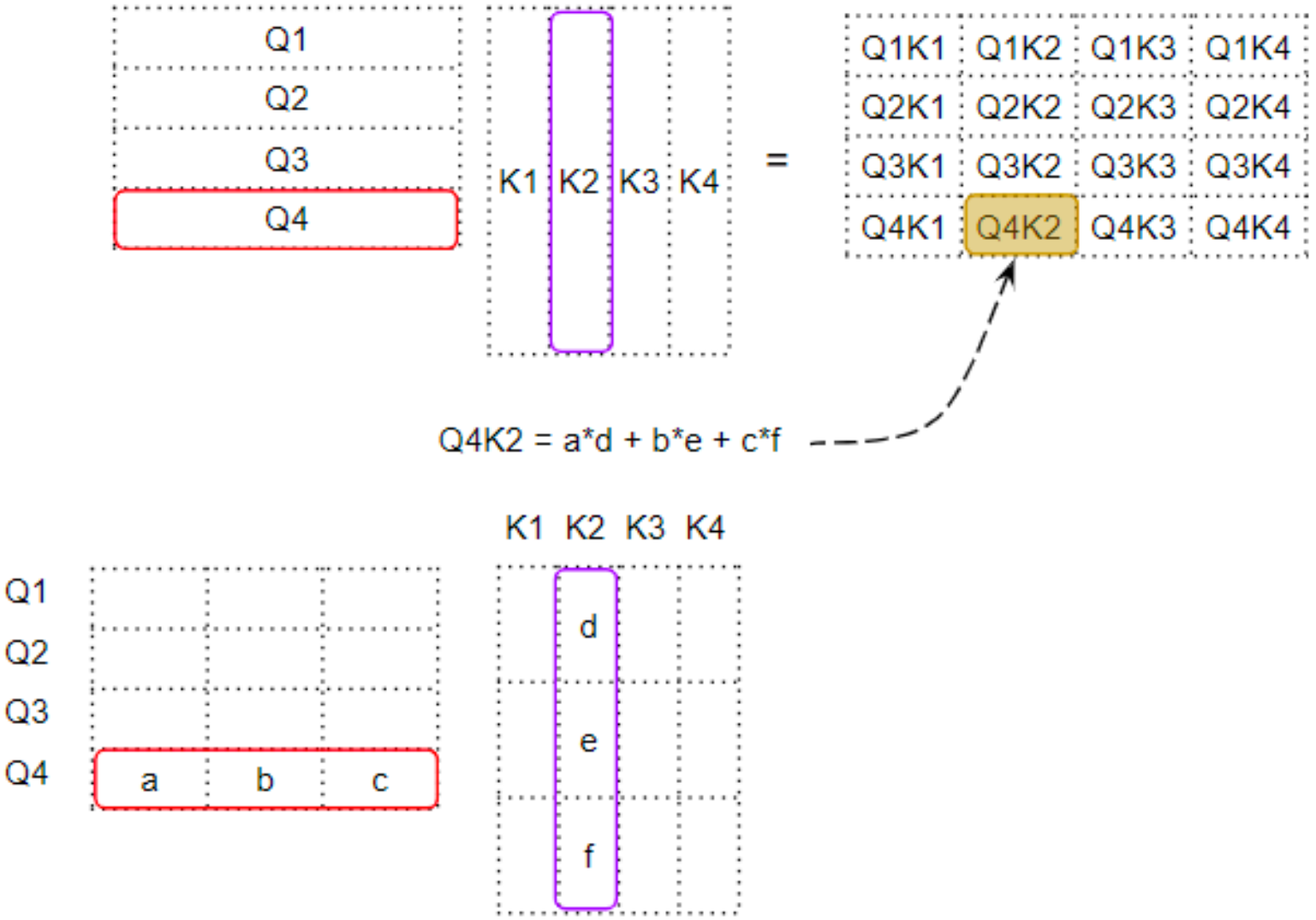
The Attention Module




Attention Score
Q1
Q2
Q3
Q4
K1
K2
K3
K4
V1
V2
V3
V4
Linear
Linear
Linear
Query
Key
Value
X1
X2
X3
X4
Input Embedding and Position Encoding
S1
S2
S3
S4


Attention Score
Q1
Q2
Q3
Q4
K1
K2
K3
K4
V1
V2
V3
V4
Linear
W_q
Linear
W_k
Linear
W_v
Query
Key
Value
X1
X2
X3
X4
Input Embedding and Position Encoding
W_e
S1
S2
S3
S4


Q1
Q2
Q3
Q4
K1
K2
K3
K4
=
Q1K1
Q1K2
Q1K3
Q1K4
Q2K1
Q2K2
Q2K3
Q2K4
Q3K1
Q3K2
Q3K3
Q3K4
Q4K1
Q4K2
Q4K3
Q4K4
Q1
Q2
Q3
Q4
K1
K2
K3
K4
=
Q1K1
Q1K2
Q1K3
Q1K4
Q2K1
Q2K2
Q2K3
Q2K4
Q3K1
Q3K2
Q3K3
Q3K4
Q4K1
Q4K2
Q4K3
Q4K4


Q1
Q2
Q3
Q4
K1 K2 K3 K4
=
Q1K1 Q1K2 Q1K3 Q1K4
Q2K1 Q2K2 Q2K3 Q2K4
Q3K1 Q3K2 Q3K3 Q3K4
Q4K1 Q4K2 Q4K3 Q4K4
Q4K2 = a*d + b*e + c*f
K1 K2 K3 K4
Q1
Q2
Q3
Q4
a b c
d
e
f

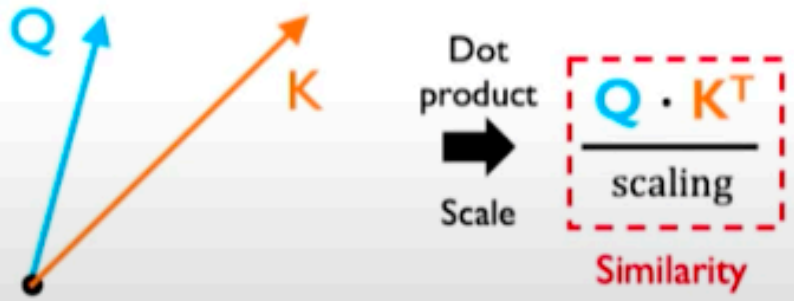
Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract features with high attention

Attention score: compute pairwise similarity between each query and key

How to compute similarity between two sets of features?



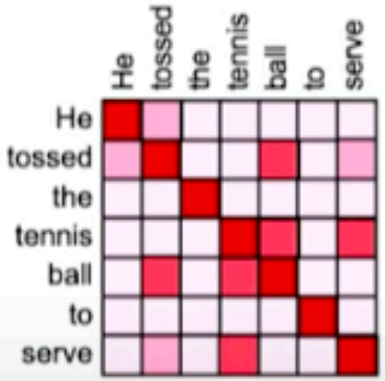
Also known as the "cosine similarity"

Learning Self-Attention with Neural Networks

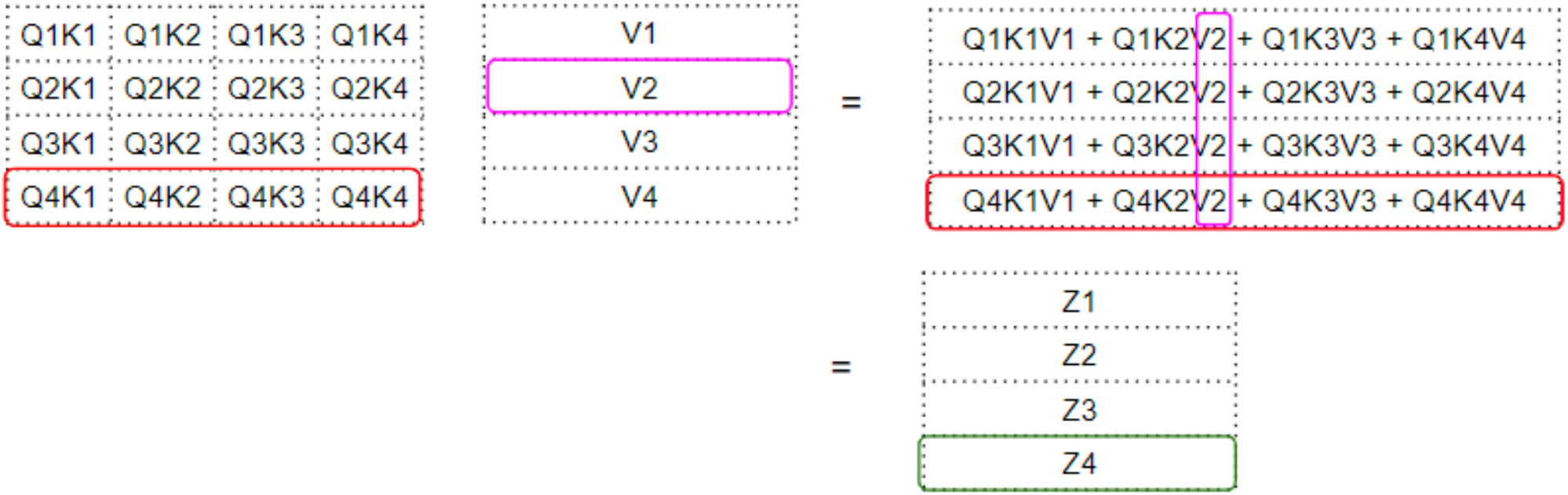
Goal: identify and attend to most important features in input.

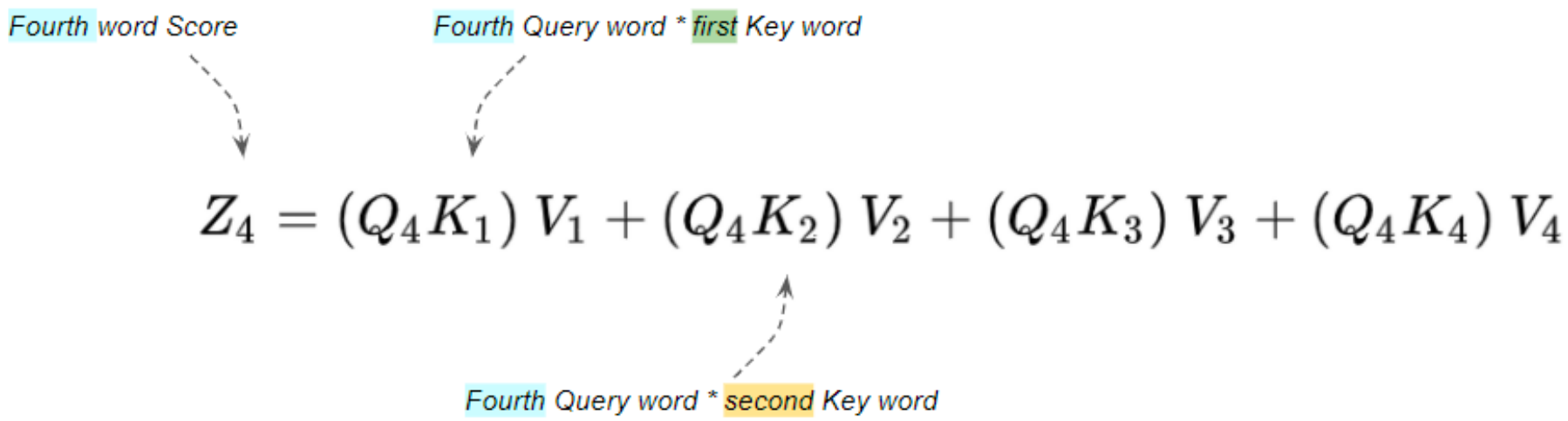
1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract features with high attention

Attention weighting: where to attend to! How similar is the key to the query?

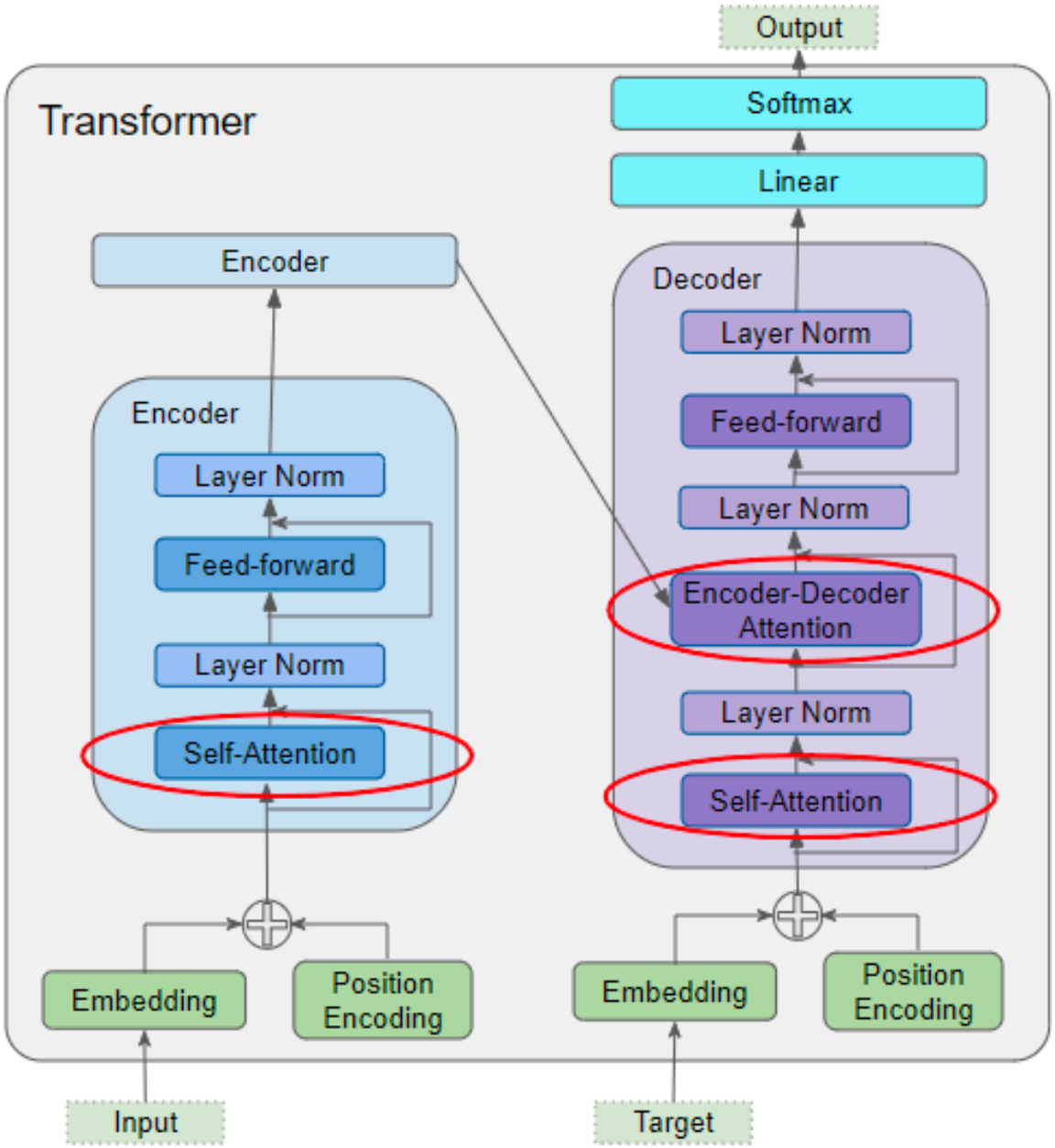


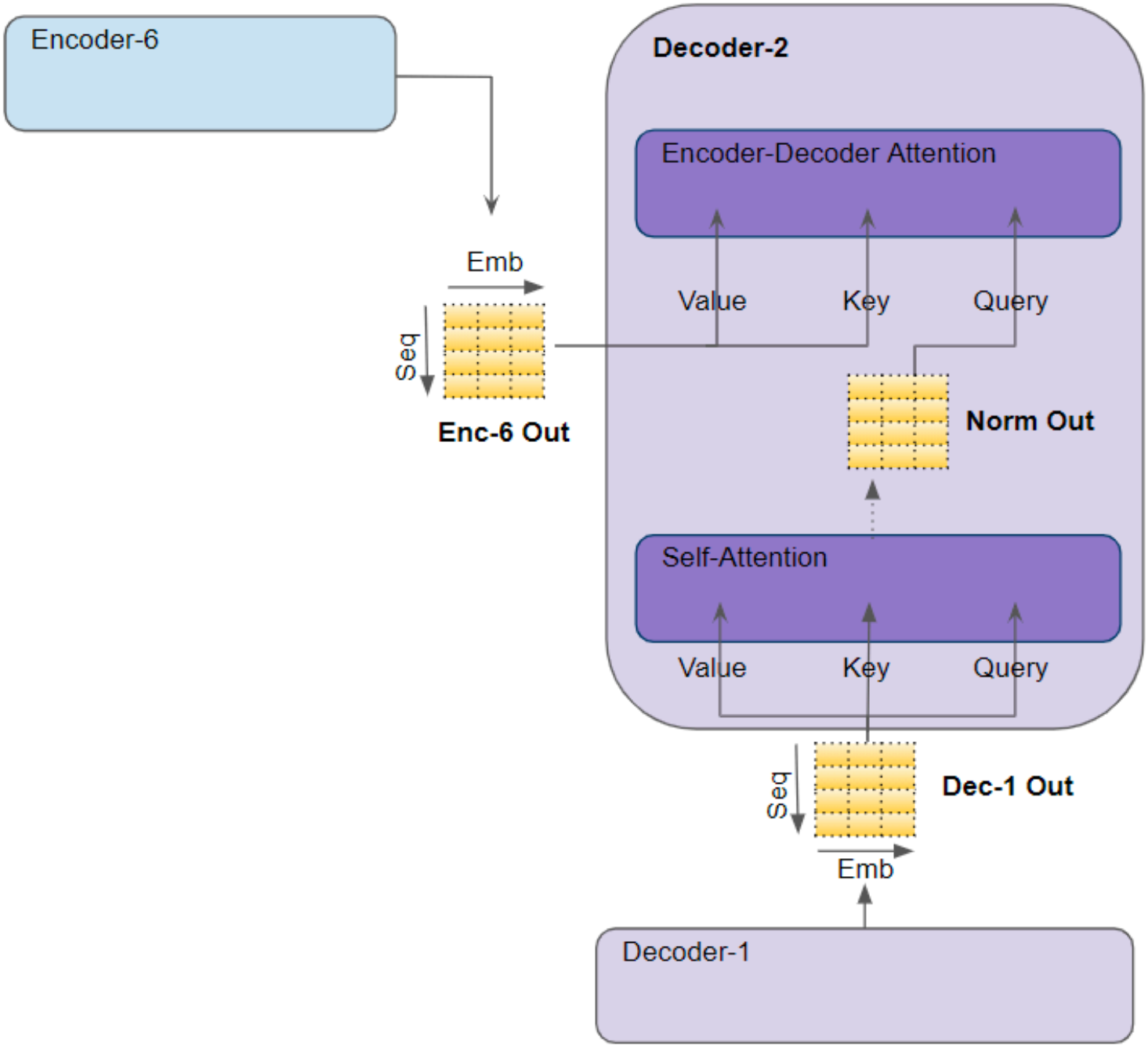
$$softmax\left(\frac{Q \cdot K^T}{scaling}\right)$$


Q1K1 Q1K2 Q1K3 Q1K4
Q2K1 Q2K2 Q2K3 Q2K4
Q3K1 Q3K2 Q3K3 Q3K4
Q4K1 Q4K2 Q4K3 Q4K4
V1
V2
V3
V4
=
Q1K1V1 + Q1K2V2 + Q1K3V3 + Q1K4V4
Q2K1V1 + Q2K2V2 + Q2K3V3 + Q2K4V4
Q3K1V1 + Q3K2V2 + Q3K3V3 + Q3K4V4
Q4K1V1 + Q4K2V2 + Q4K3V3 + Q4K4V4
=
Z1
Z2
Z3
Z4



$$Z = Softmax(\frac{QK^T}{\sqrt{d_k}})V$$


Output
Transformer
Softmax
Linear
Encoder
Decoder
Layer Norm
Feed-forward
Layer Norm
Encoder-Decoder Attention
Layer Norm
Self-Attention
Encoder
Layer Norm
Feed-forward
Layer Norm
Self-Attention
Embedding
Position Encoding
Embedding
Position Encoding
Input
Target

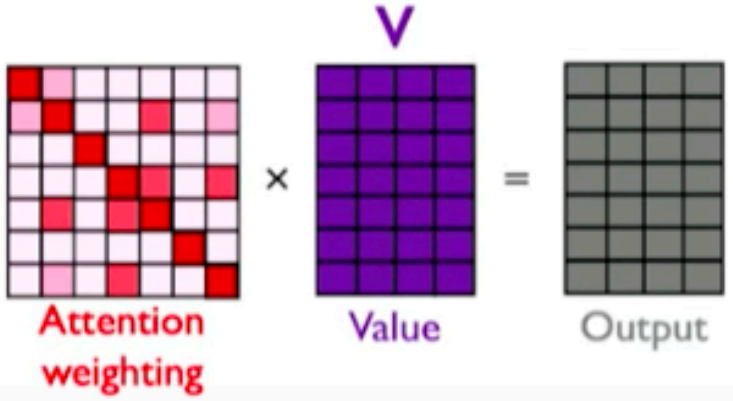

Encoder-6
Decoder-2
Encoder-Decoder Attention
Emb
Value
Key
Query
Seq
Enc-6 Out
Norm Out
Self-Attention
Value
Key
Query
Seq
Dec-1 Out
Emb
Decoder-1

Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

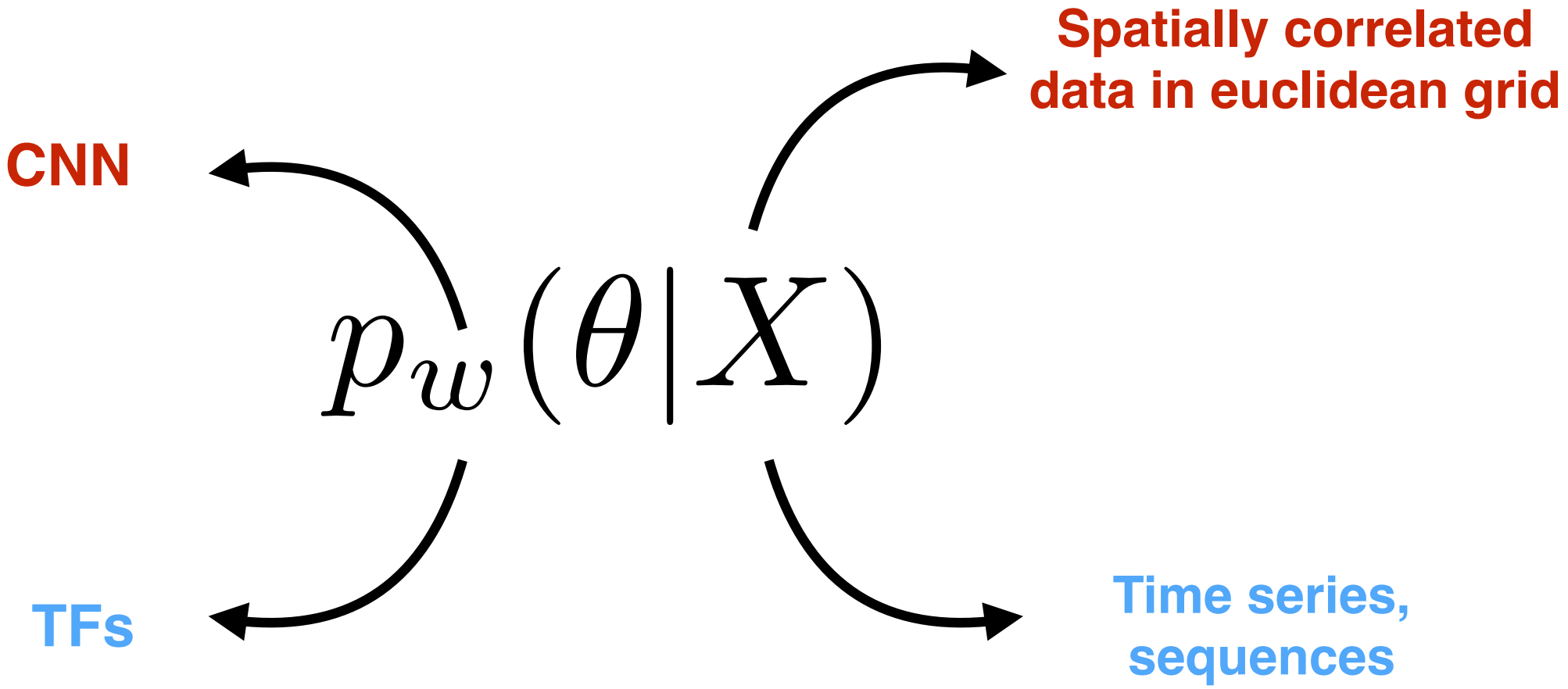
1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**

Last step: self-attend to extract features



$$softmax\left(\frac{Q \cdot K^T}{scaling}\right) \cdot V = A(Q,K,V)$$

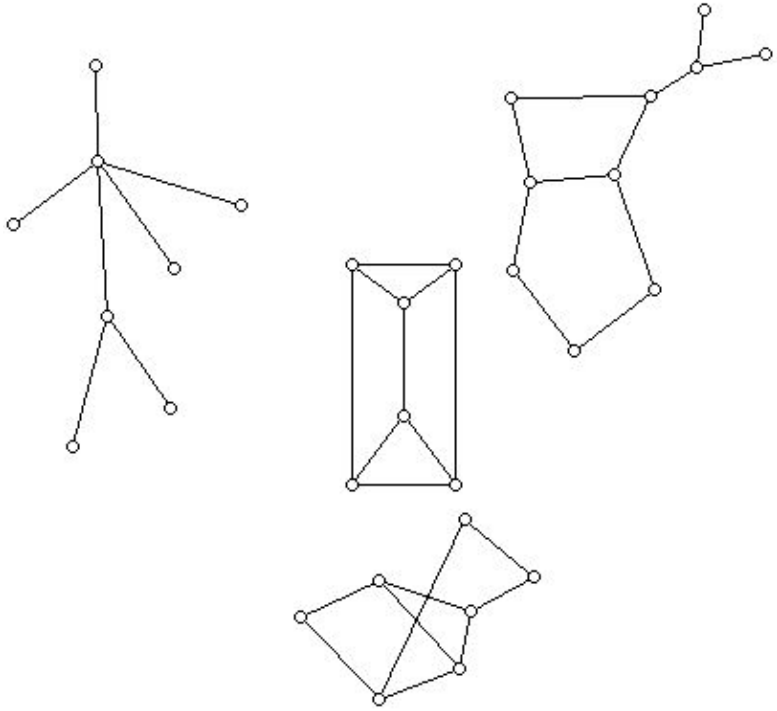
RECAP4:

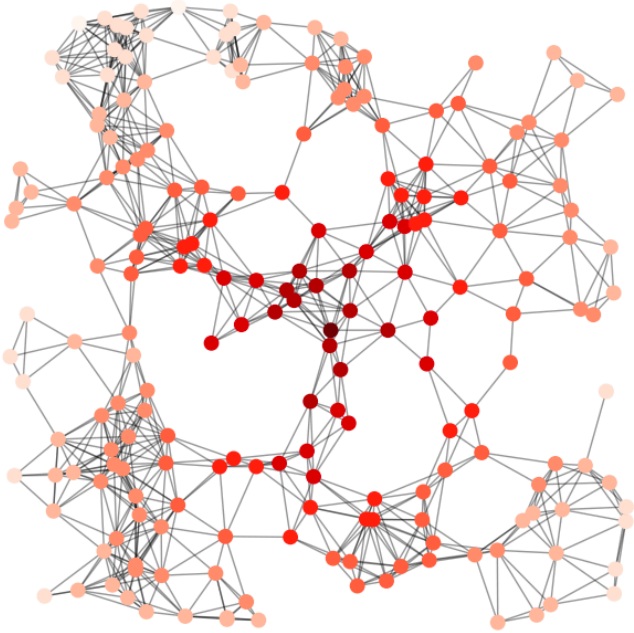


Approach	Discriminative		Generative	
Data type	Target Function	Method	Target Function	Method
Labels	$p(y\|x)$	All supervised networks with bottleneck	$p(x\|y)$	Conditional generative models
No Labels	$p(\hat{y}\|x)$	Self-supervised learning	$p(x)$	Generative models, Autoencoders

A very brief incursion into GNNs…

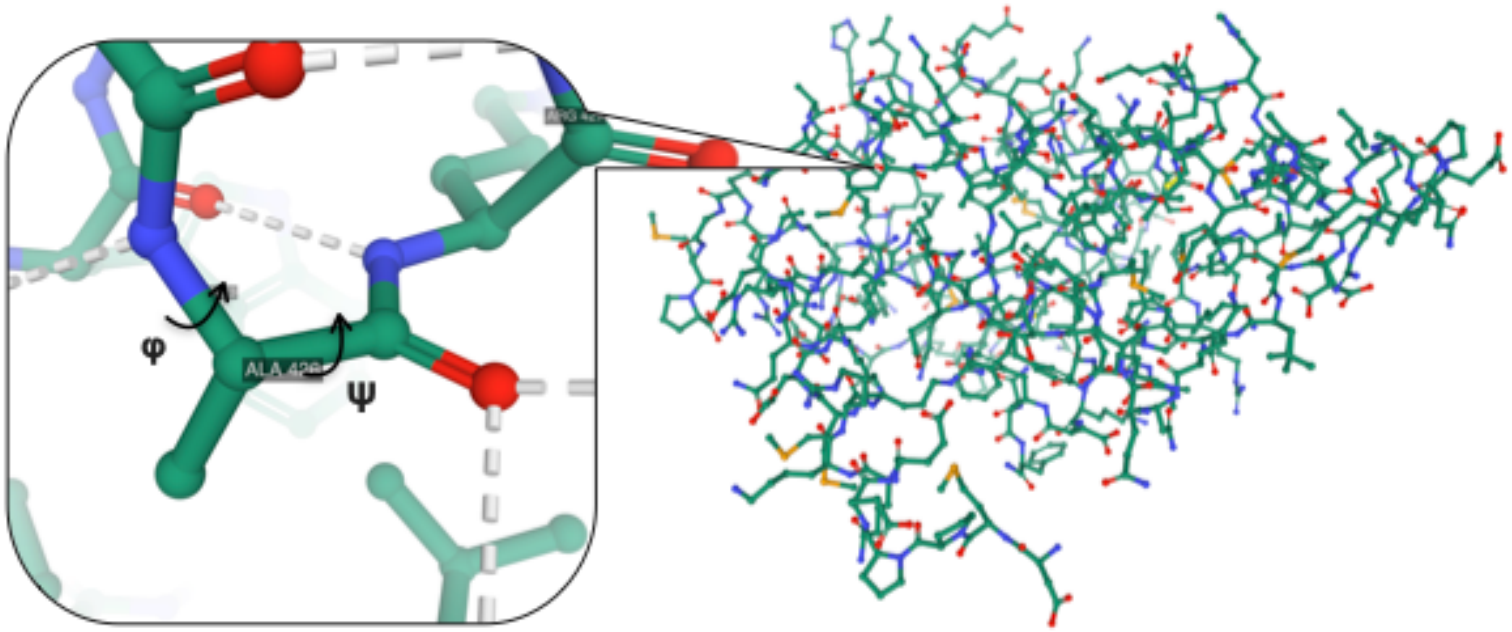
What is a graph?





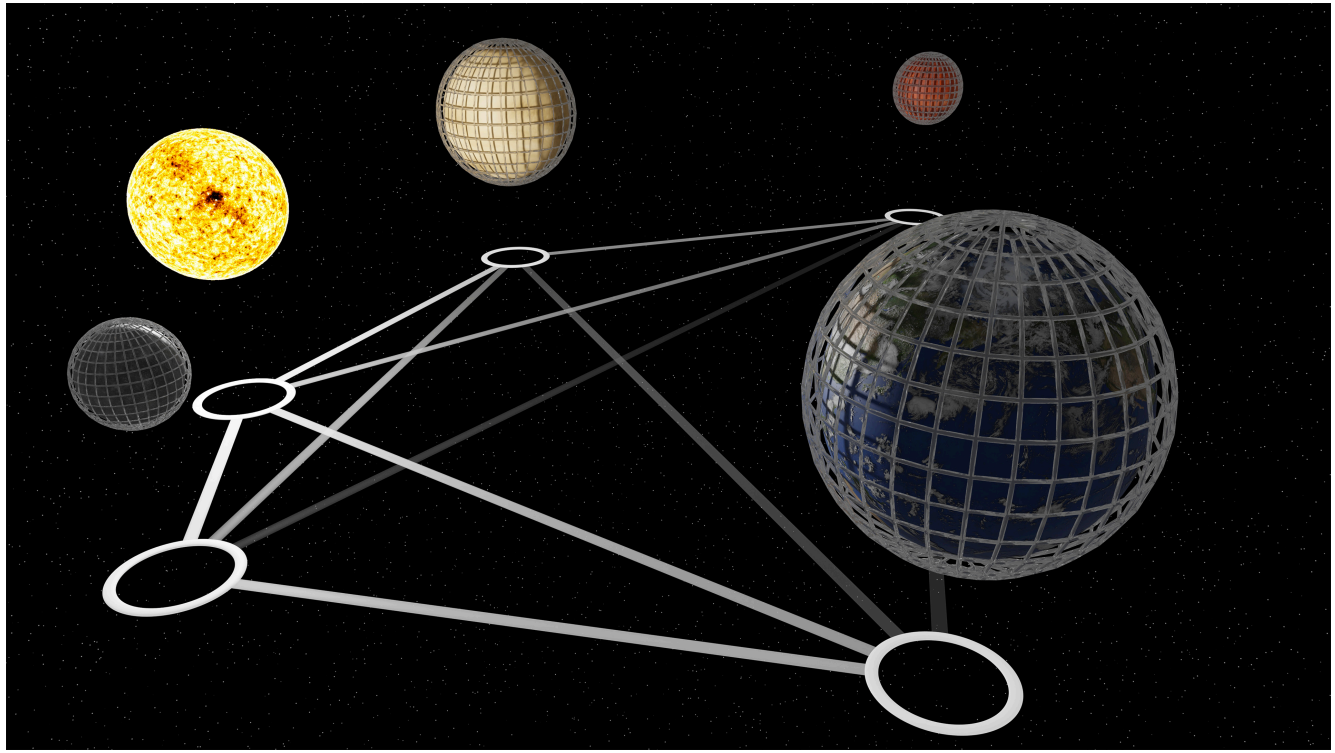
!!!

Molecules are Graphs

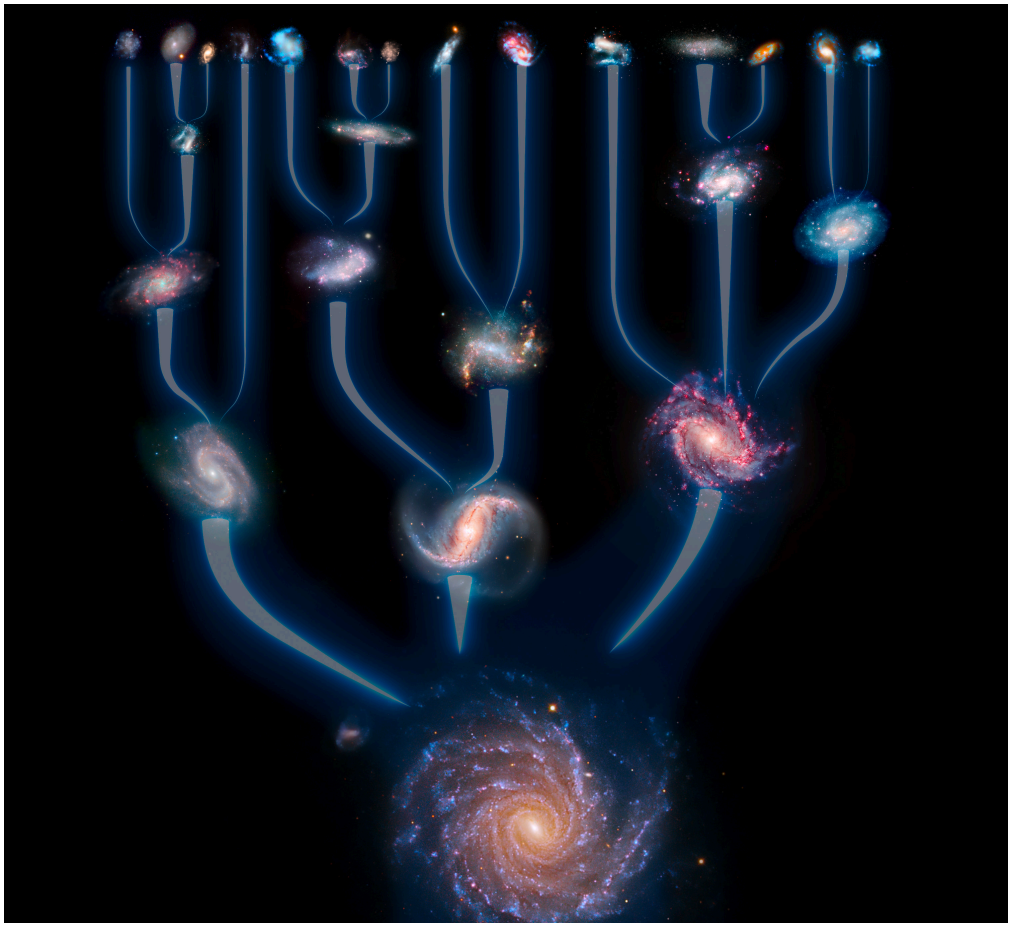


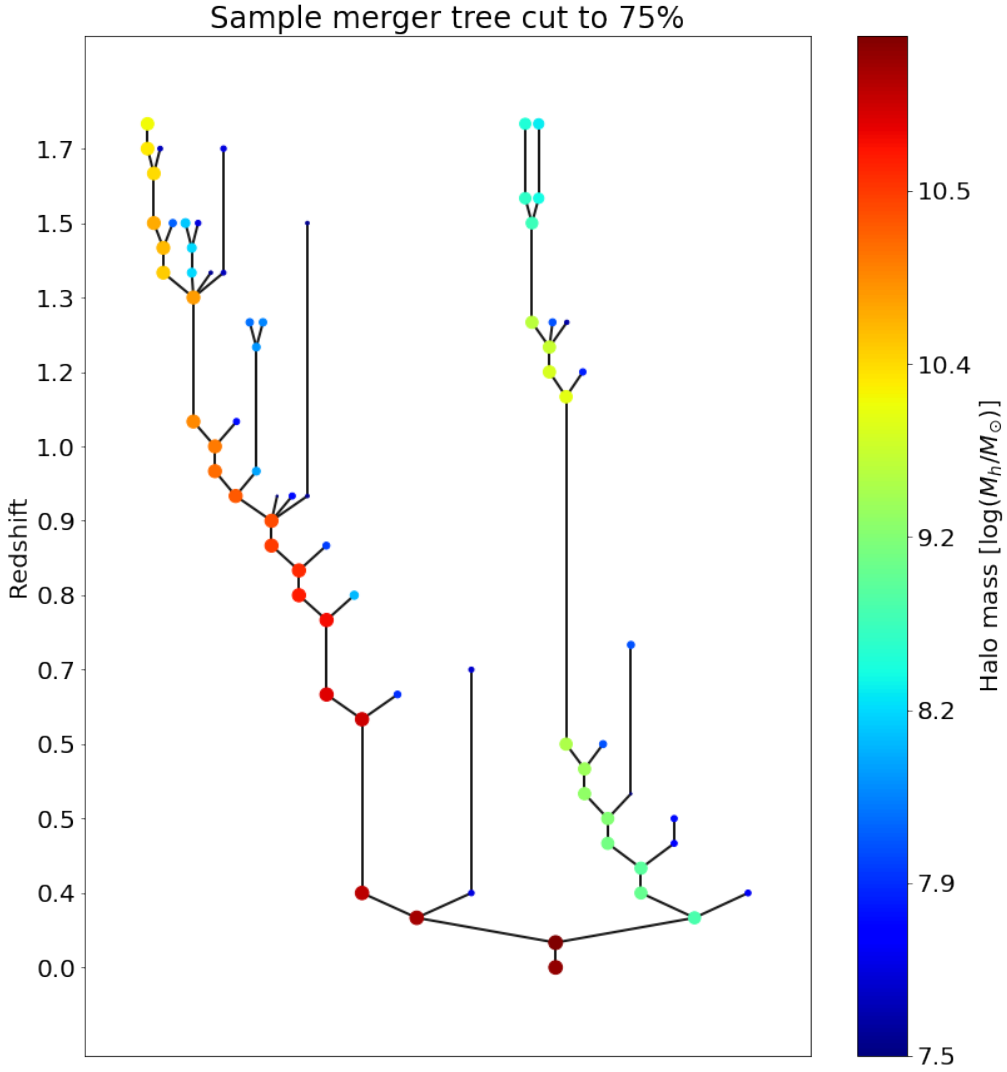
The Solar System is a Graph

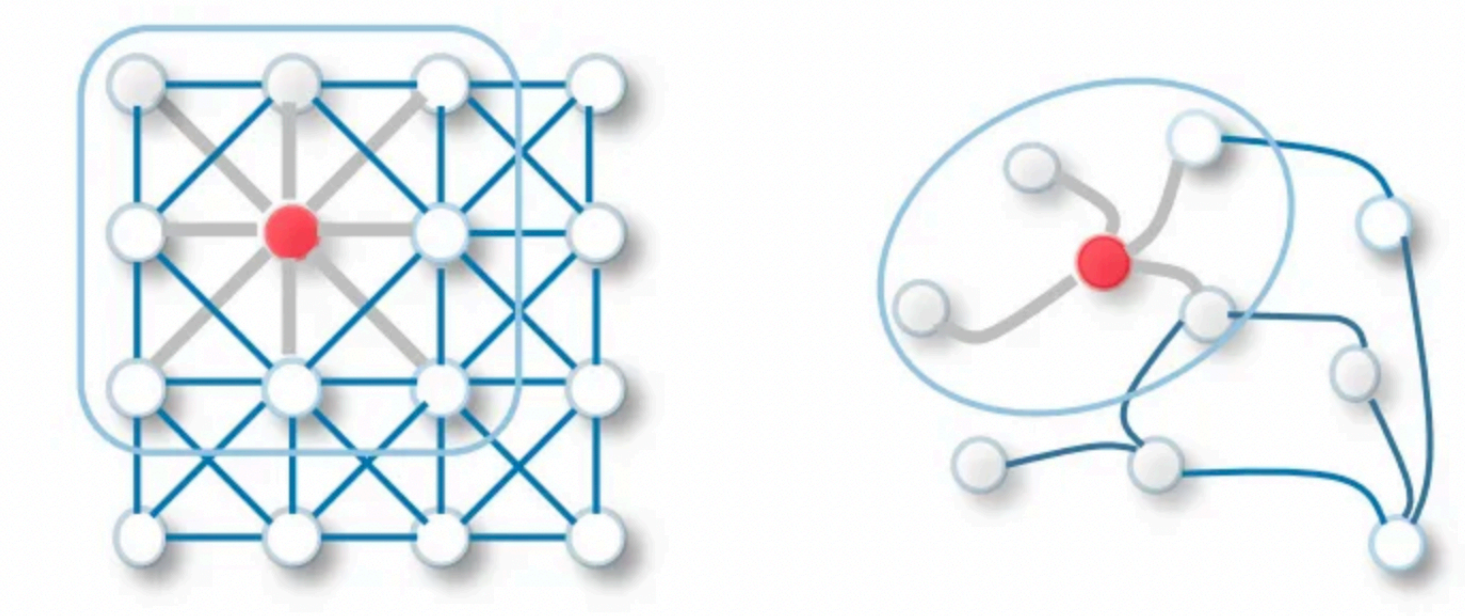
Or at least a natural abstraction



Galaxy Merger trees are Graphs



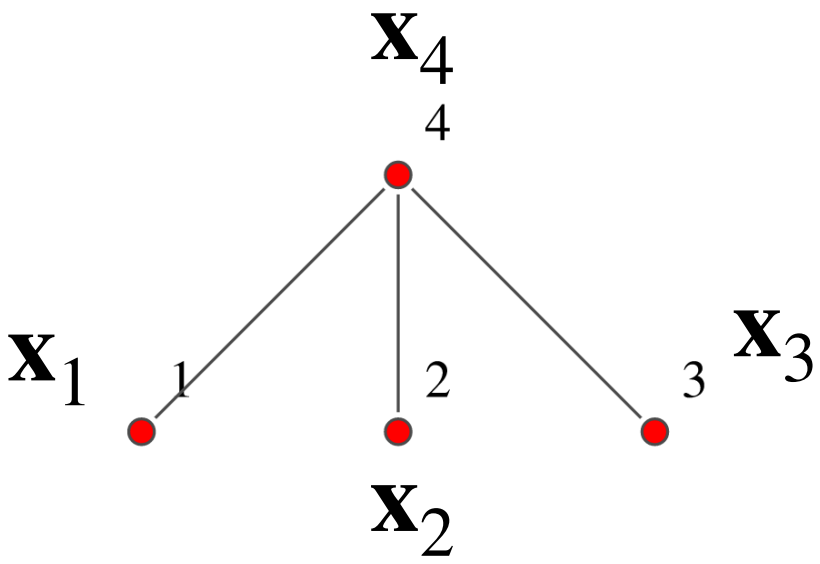

Sample merger tree cut to 75%
Redshift
1.7
1.5
1.3
1.2
1.0
0.9
0.8
0.7
0.5
0.5
0.4
0.0
Halo mass [log($M_h/M_\odot$)]
10.5
10.4
9.2
8.2
7.9
7.5



The main difference compared to images
is in the definition of neighbours

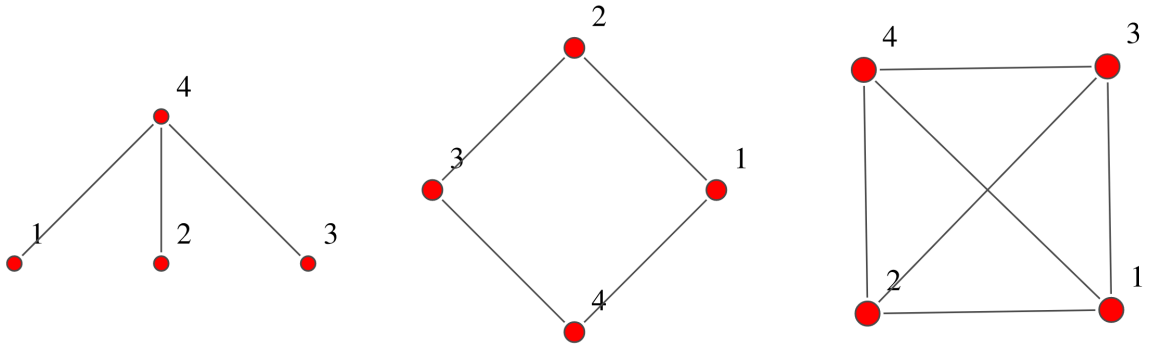
What makes up a graph?

Nodes and node features



What makes up a graph?

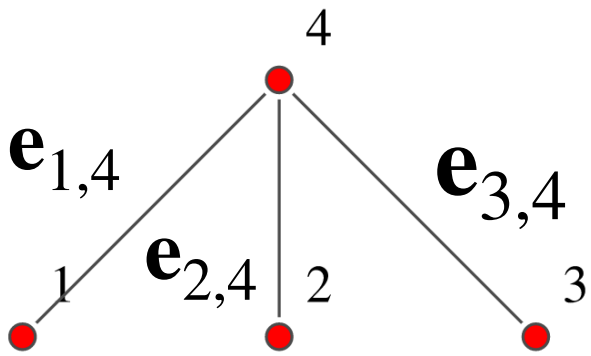
Nodes and node features



$$\mathbf{X} = \left[\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\right]^\top \rightarrow \quad \mathbf{X} = \left[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{|\mathcal{V}|}\right]^\top \rightarrow \quad \mathbf{X} = \left\{\mathbf{x}_i \mid i \in N_v\right\} \in \mathbb{R}^{N_v \times d_v}$$

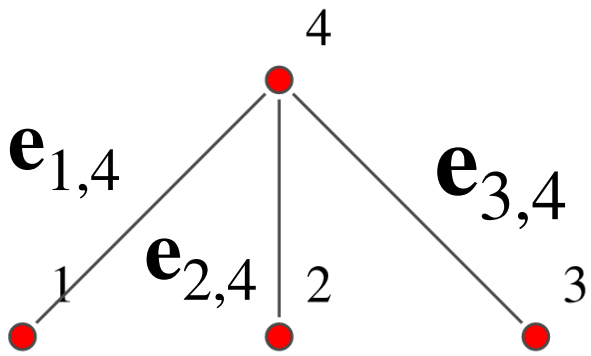
What makes up a graph?

Edges and edge features



What makes up a graph?

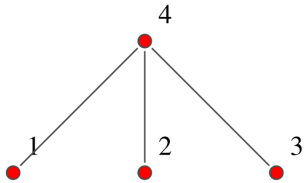
Edges and edge features

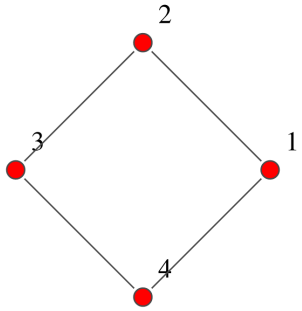


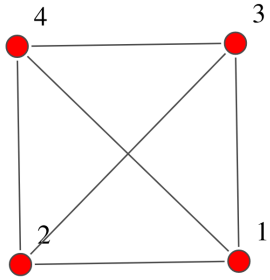
$$\mathbf{E} = \left[\mathbf{e}_{12}, \mathbf{e}_{13}, \mathbf{e}_{14}\right]^\top \rightarrow \quad \mathbf{E} = \left[\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{|\mathscr{E}|}\right]^\top \rightarrow \quad \mathbf{E} = \left\{\mathbf{e}_i \mid i \in N_e\right\} \in \mathbb{R}^{N_e \times d_e}$$

What makes up a graph?

The Adjacency Matrix

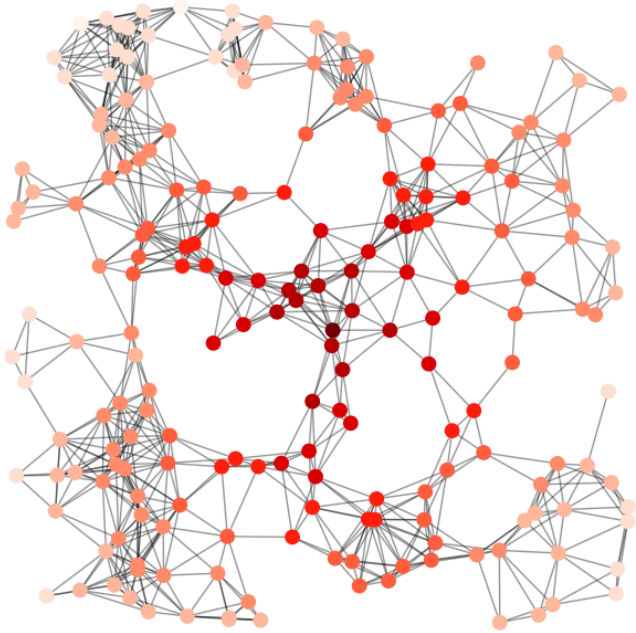






$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Global Features



Could be physical properties e.g.

$$\mathbf{U} = [\Omega_m, \sigma_8]$$

Statistical properties,
e.g. manually inserted

$$\mathbf{U} = \left[\mu_{\mathbf{x}}, \sigma_{\mathbf{x}}, max(\mathbf{x}), min(\mathbf{x}) \ldots P_n(\mathbf{x})\right]$$

Or completely learnable

Summarizing…

- Graph defined as $G(V, E, U)$ with G being the graph, V being the set of nodes, E being the set of edges and U being the global feature vector
- Adjacency matrix (A) encodes the structure of the graph and is a binary matrix

Locality constraints

The edges of a graph naturally define the concept of a local **neighborhood**

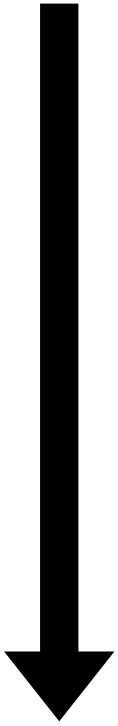
$$\mathcal{N}_u = \{v \mid (u, v) \in \mathbf{E} \vee (v, u) \in \mathbf{E}\}$$

Which allows us to allow us to define local functions, which work on a given node and its neighborhood.

$$\phi\left(\mathbf{x}_u, \mathbf{X}_{\mathcal{N}_u}\right)$$

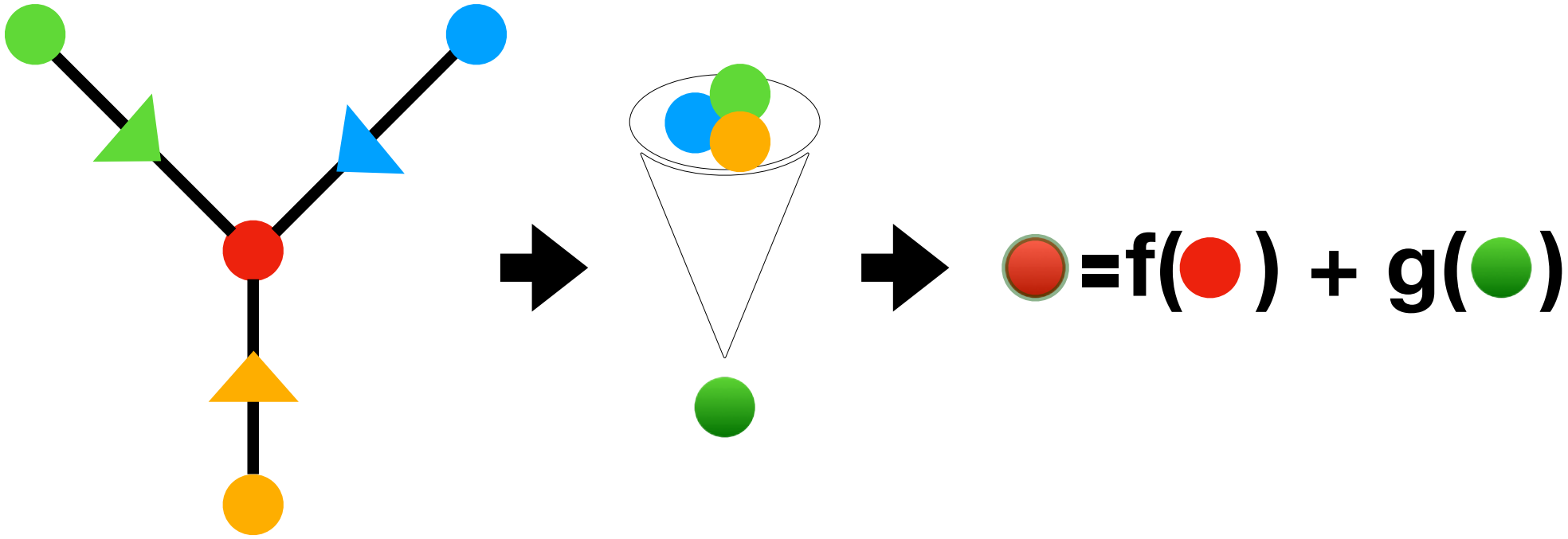
Graph Neural Network Layers

Increasing complexity/ expressivity/ training time



- Convolutional Layer $\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} c_{vu}\psi\left(\mathbf{x}_v\right)\right)$
- Attentional Layer $\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} a(\mathbf{x}_u, \mathbf{x}_v, \mathbf{e}_{uv})\psi\left(\mathbf{x}_v\right)\right)$

Looks complicated, but!



Looks complicated, but!

Essentially, these are all variants of:

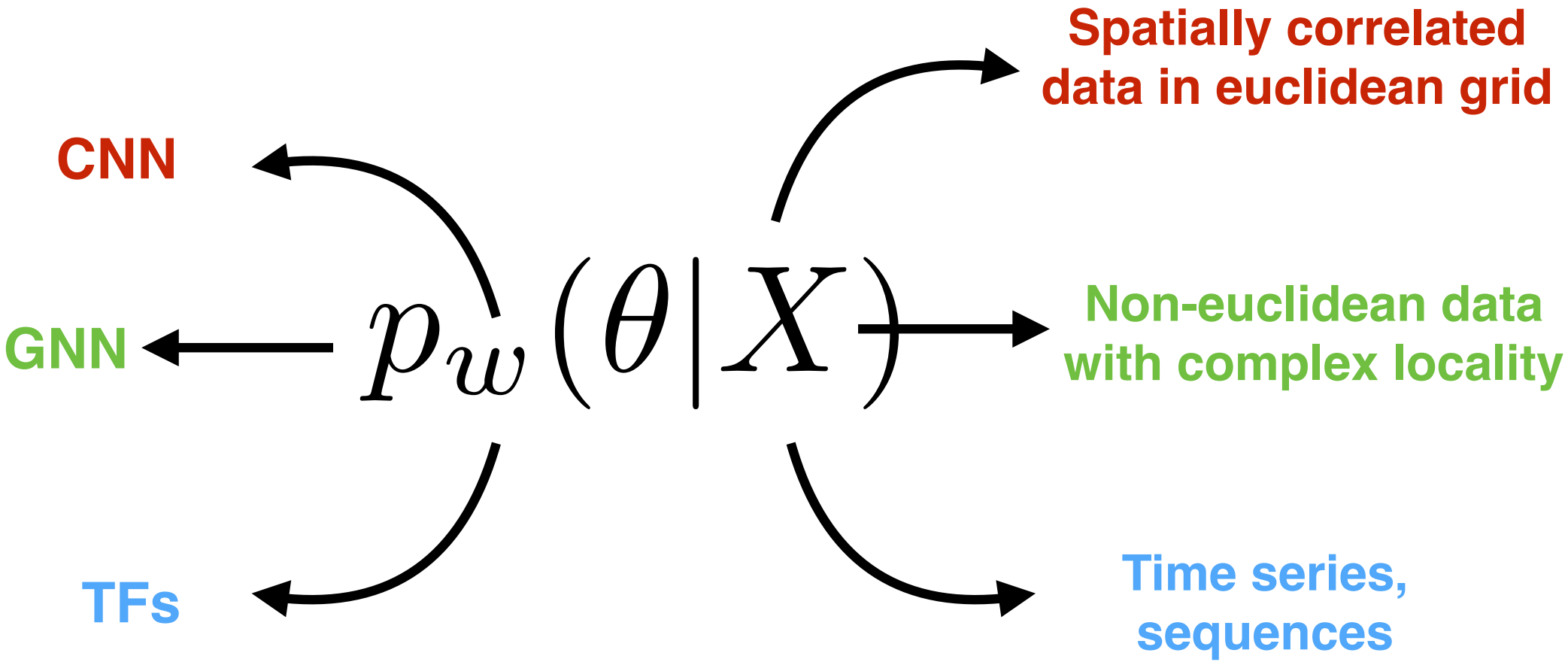
1. Define a node function f and a neighborhood function g (f and g can be the same function)
2. Act with f on the node and with g on the neighborhood

Graphs are good representations of physical systems

In modern physics … a central theme will be a Geometric Principle: The laws of physics must all be expressible as geometric (coordinate-independent and reference frame-independent) relationships between geometric objects (scalars, vectors, tensors, ...) that represent physical entities.

Physics on Graphs

- A natural abstraction of physical systems and inherently local -> easier physical interpretations/separability
- Embeds inductive biases easily by restructuring the graph -> more efficient learning and no need to learn things we already know
- Can embed permutational, rotational, translational and reflectional symmetries


Spatially correlated
data in euclidean grid
CNN
GNN
$p_w(\theta|X)$
Non-euclidean data
with complex locality
TFs
Time series,
sequences