ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

CENG 463: Introduction to Natural Language Processing
Transformer and Attention

Asst. Prof. Çağrı Toraman
Computer Engineering Department
ctoraman@ceng.metu.edu.tr

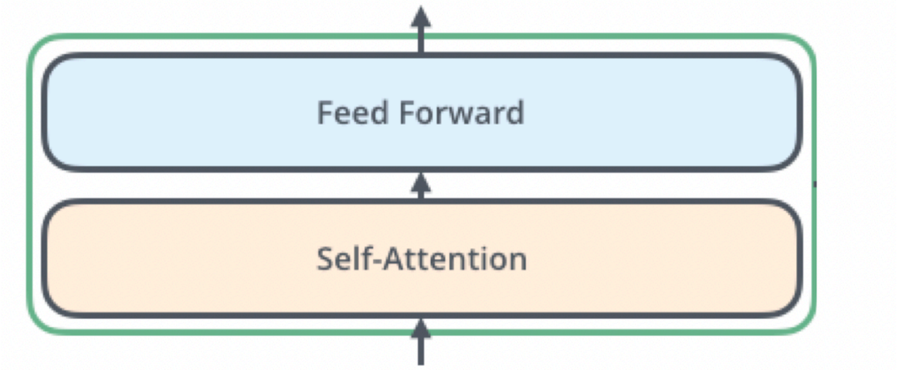
02.12.2025

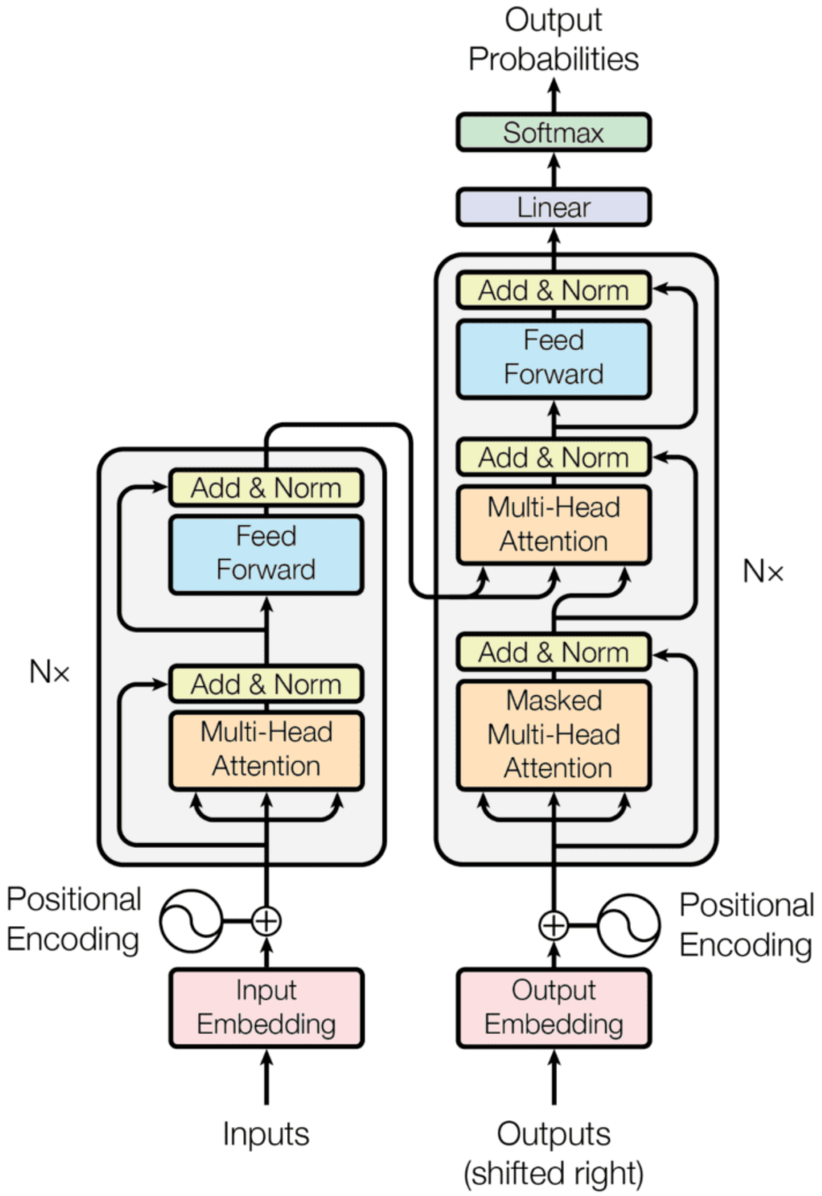
** The Course Slides are subject to **CC BY-NC**. Either the original work or a derivative work can be shared with appropriate attribution, but only for noncommercial purposes.*

Transformer (Recap)

Build whole model out of self-attention

Uses only point-wise processing and attention
(no recurrent units or convolutions)



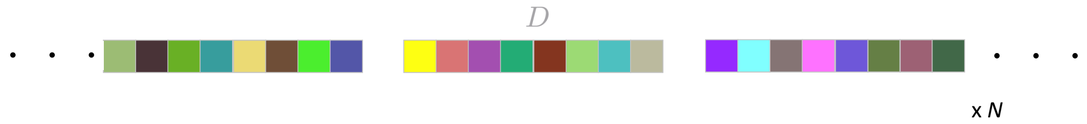


A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, NeurIPS 2017

Attention

Design neural network to encode and process text:

The restaurant refused to serve me a ham sandwich, because it only cooks vegetarian food. In the end, they just gave me two slices of bread. Their ambience was just as good as the food and service.



The word their must “attend to” the word restaurant.

Conclusions:

- There must be connections between the words.
- The strength of these connections will depend on the words themselves.

Attention

Dot-product self attention

Takes N inputs of size Dx1 and returns N inputs of size Dx1

Computes N values (no ReLU)

Attention

Dot-product self attention

Takes N inputs of size Dx1 and returns N inputs of size Dx1

Computes N values (no ReLU)

$$\mathbf{v}_n = \boldsymbol{\beta}_v + \boldsymbol{\Omega}_v \mathbf{x}_n$$

Attention

Dot-product self attention

Takes N inputs of size Dx1 and returns N inputs of size Dx1

Computes N values (no ReLU)

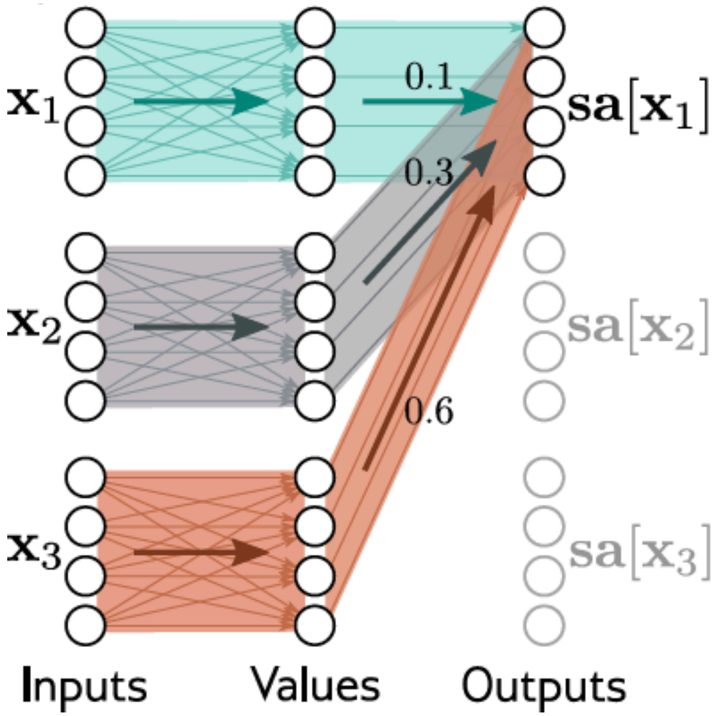
$$\mathbf{v}_n = \boldsymbol{\beta}_v + \boldsymbol{\Omega}_v \mathbf{x}_n$$

$$\mathbf{sa}[\mathbf{x}_n] = \sum_{m=1}^{N} a[\mathbf{x}_n, \mathbf{x}_m]\mathbf{v}_m$$

N outputs are weighted sums of these values

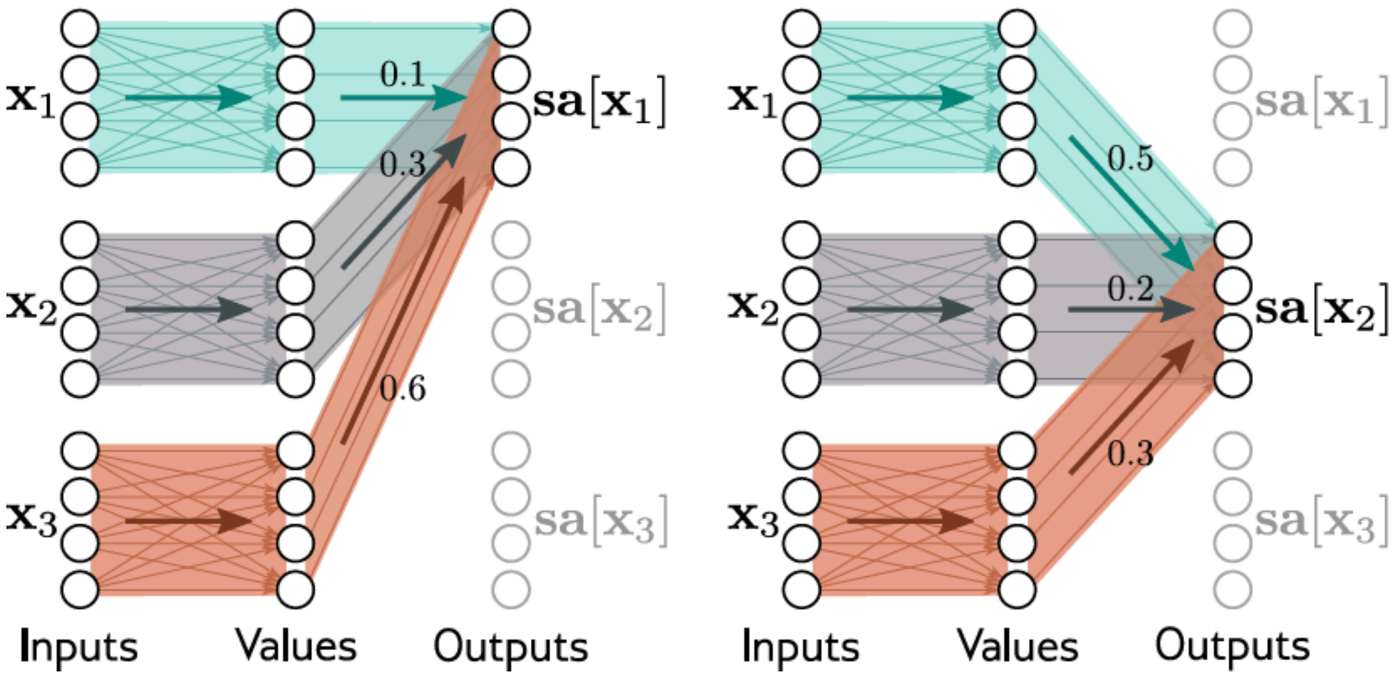
Attention

Attention as routing



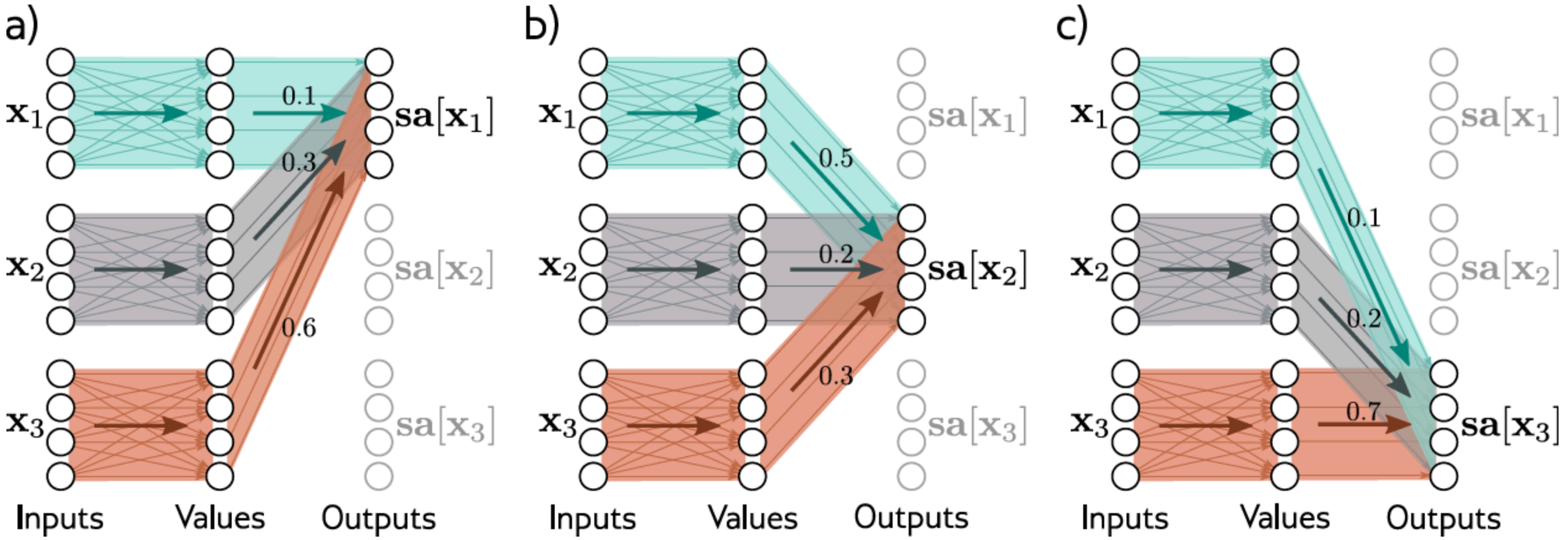
Attention

Attention as routing

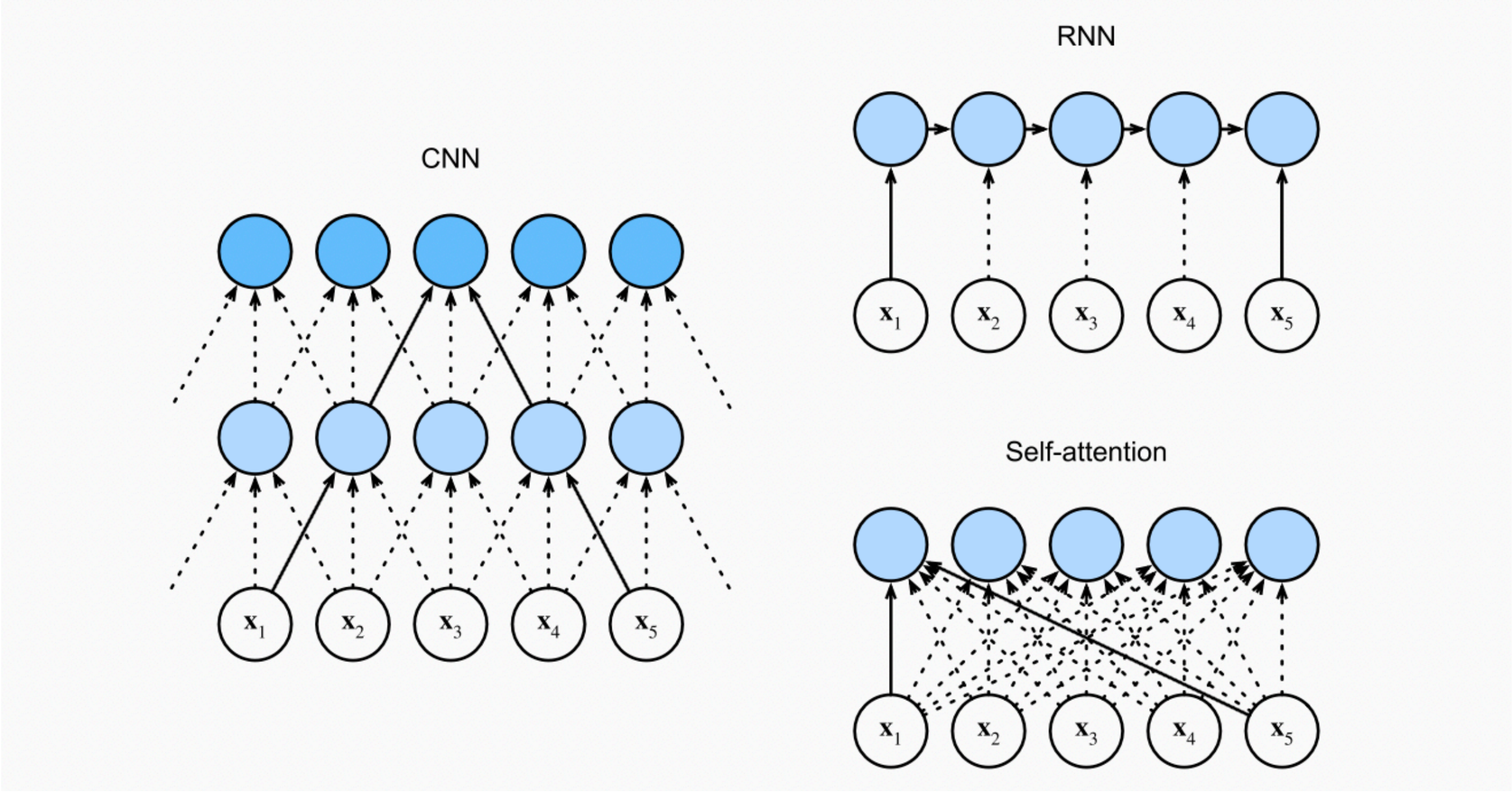


Attention

Attention as routing



Attention



Attention

Attention weights

Compute N “queries” and N “keys” from input

$$\mathbf{q}_n = \boldsymbol{\beta}_q + \boldsymbol{\Omega}_q \mathbf{x}_n$$
$$\mathbf{k}_n = \boldsymbol{\beta}_k + \boldsymbol{\Omega}_k \mathbf{x}_n,$$

Calculate similarity and pass through softmax:

$$a[\mathbf{x}_n, \mathbf{x}_m] = \text{softmax}_m\left[\text{sim}[\mathbf{k}_m \mathbf{q}_n]\right]$$
$$= \frac{\exp\left[\text{sim}[\mathbf{k}_m \mathbf{q}_n]\right]}{\sum_{m'=1}^{N} \exp\left[\text{sim}[\mathbf{k}'_m \mathbf{q}_n]\right]};$$

Attention

Attention weights

Compute N “queries” and N “keys” from input

$$\mathbf{q}_n = \boldsymbol{\beta}_q + \boldsymbol{\Omega}_q \mathbf{x}_n$$
$$\mathbf{k}_n = \boldsymbol{\beta}_k + \boldsymbol{\Omega}_k \mathbf{x}_n,$$

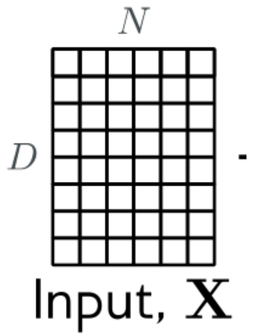
Calculate similarity and pass through softmax:

$$a[\mathbf{x}_n, \mathbf{x}_m] = \text{softmax}_m\left[\mathbf{k}_m^T \mathbf{q}_n\right]$$
$$= \frac{\exp\left[\mathbf{k}_m^T \mathbf{q}_n\right]}{\sum_{m'=1}^{N} \exp\left[\mathbf{k}_{m'}^T \mathbf{q}_n\right]}$$

Attention

Matrix form

Store N input vectors in matrix X



Compute values, queries and keys:

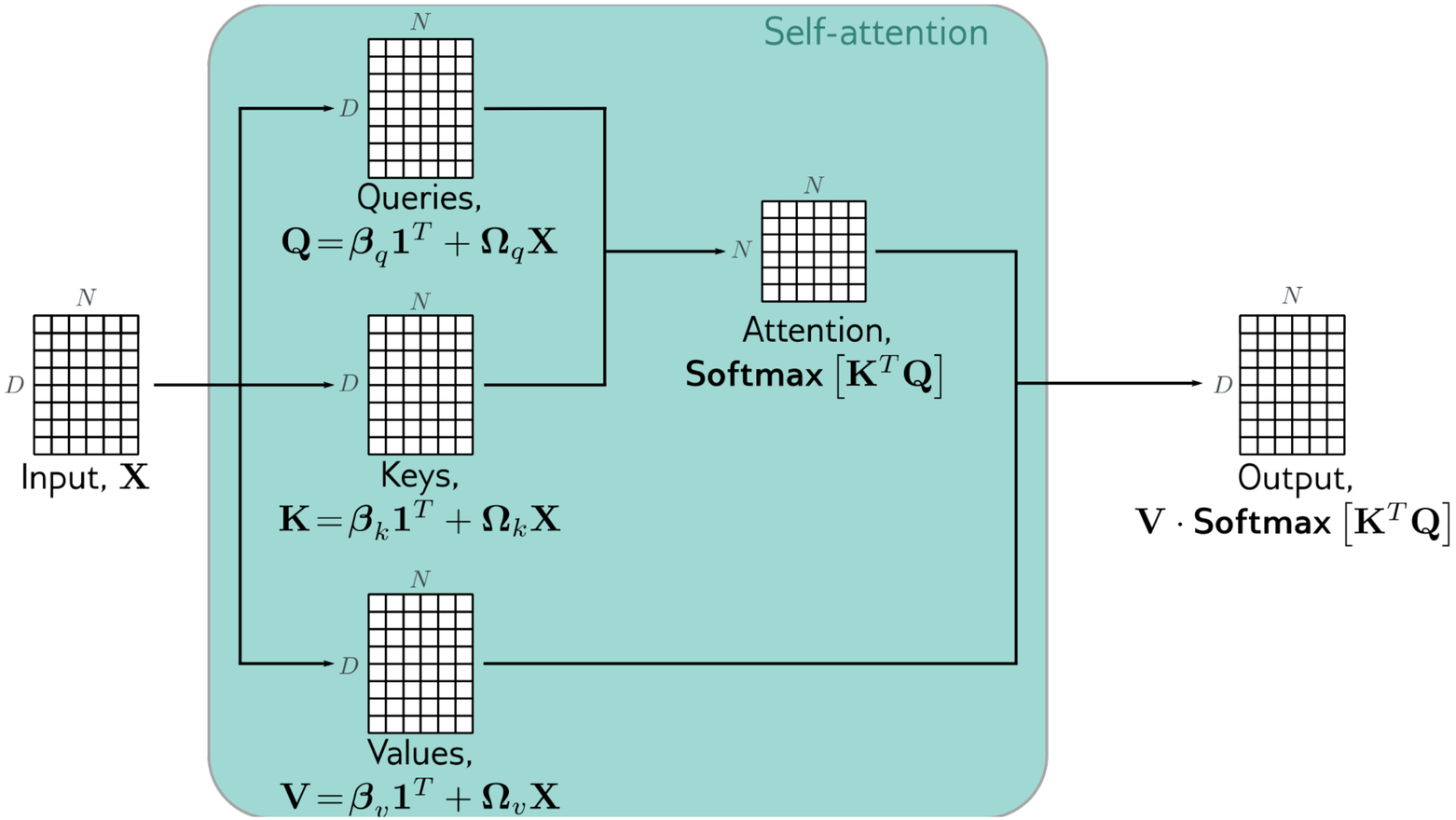
$$\mathbf{V}[\mathbf{X}] = \boldsymbol{\beta}_v \mathbf{1}^{\mathbf{T}} + \boldsymbol{\Omega}_{\mathbf{v}} \mathbf{X}$$
$$\mathbf{Q}[\mathbf{X}] = \boldsymbol{\beta}_q \mathbf{1}^{\mathbf{T}} + \boldsymbol{\Omega}_{\mathbf{q}} \mathbf{X}$$
$$\mathbf{K}[\mathbf{X}] = \boldsymbol{\beta}_k \mathbf{1}^{\mathbf{T}} + \boldsymbol{\Omega}_{\mathbf{k}} \mathbf{X},$$

Combine self-attentions

$$\mathbf{Sa}[\mathbf{X}] = \mathbf{V}[\mathbf{X}] \cdot \mathbf{Softmax}\Big[\mathbf{K}[\mathbf{X}]^T \mathbf{Q}[\mathbf{X}]\Big]$$

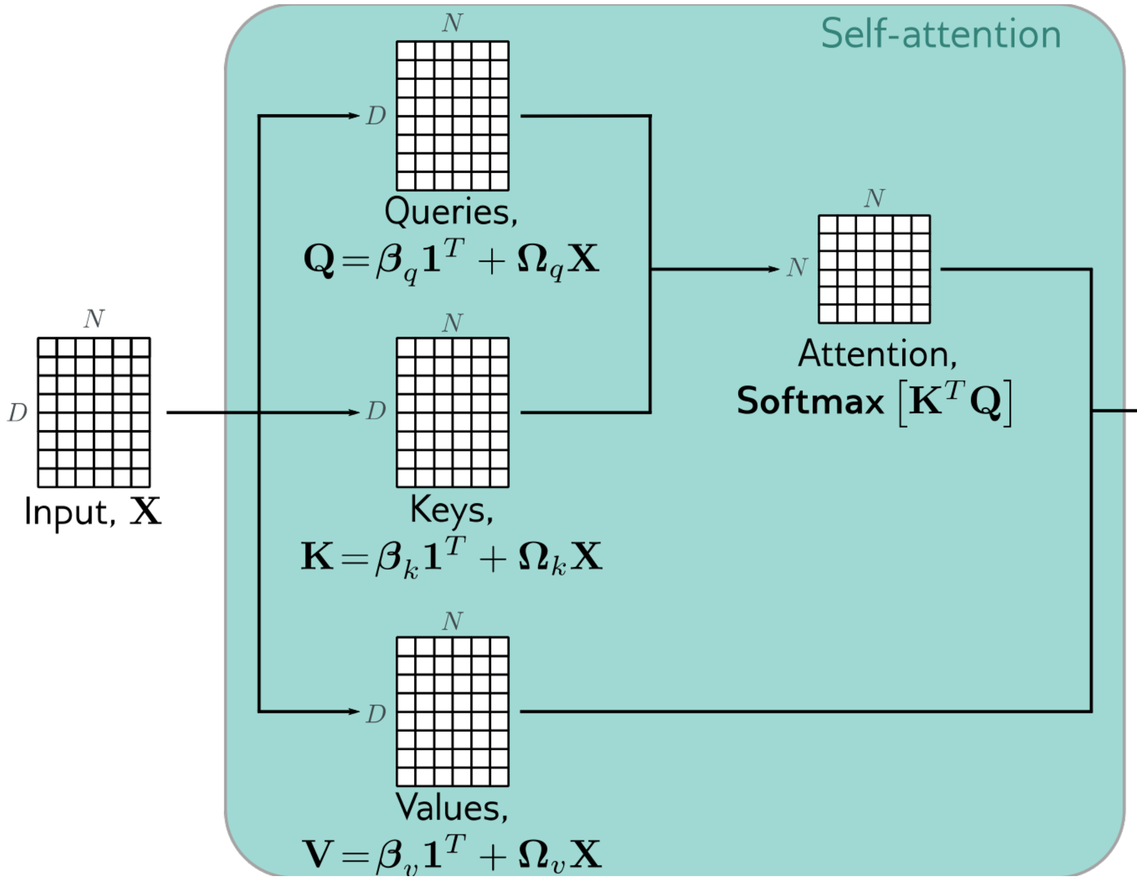
Attention

Matrix form



Attention

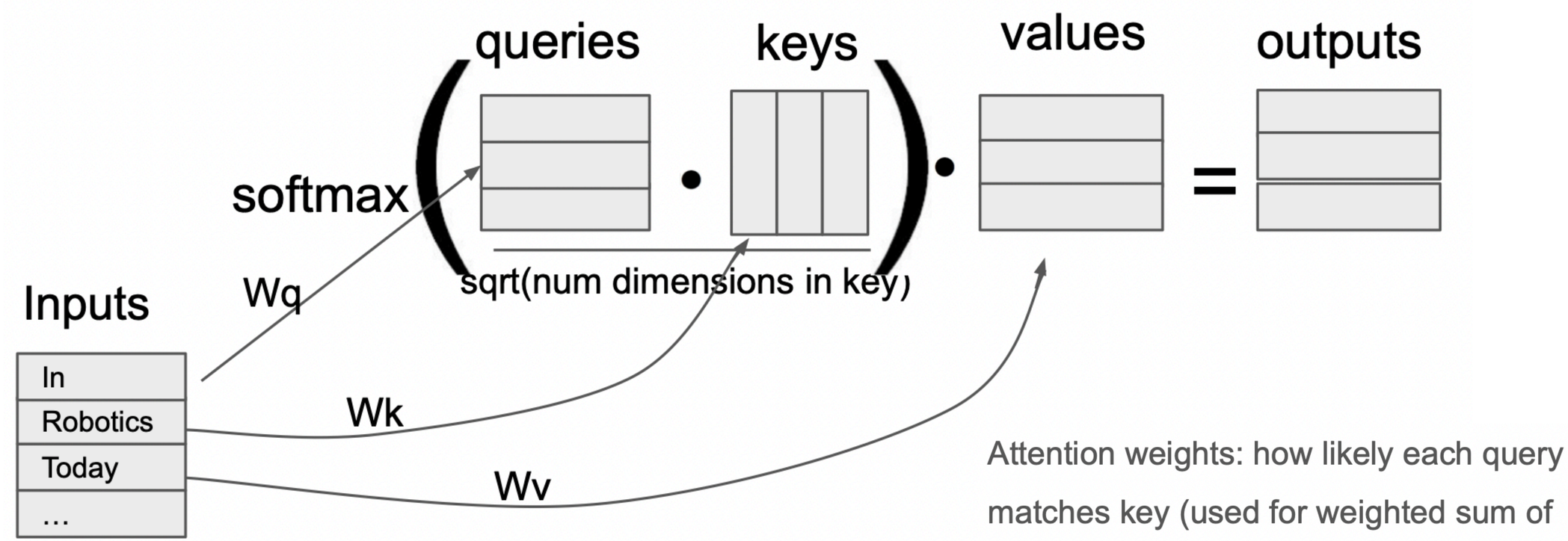
Matrix form



$$\text{Attention}(Q,K,V)=\text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Dividing by d_k makes algorithm easier to train (controlling the magnitude of the attention scores)

Attention



Attention weights: how likely each query matches key (used for weighted sum of values)

Multi-Head Attention

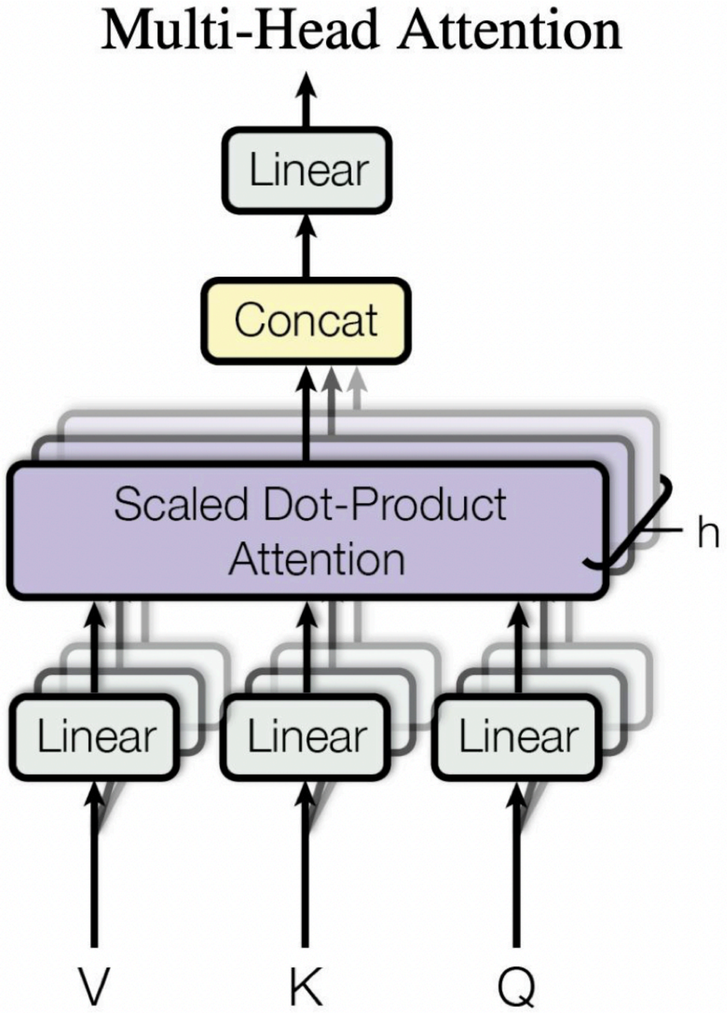
- Idea:
  a. Stack linear layers (weight matrices without biases) that are independent each for keys, queries, values.
  b. Concatenate output of attention heads to form (plus non-linearity) output layer
- Why?
  a. Allows for model to focus on different positions
  b. Gives attention layer multiple “representation subspaces”
  c. No longer need to oversaturate one attention mechanism

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$

$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$h = 8$ parallel attention layers, or heads.

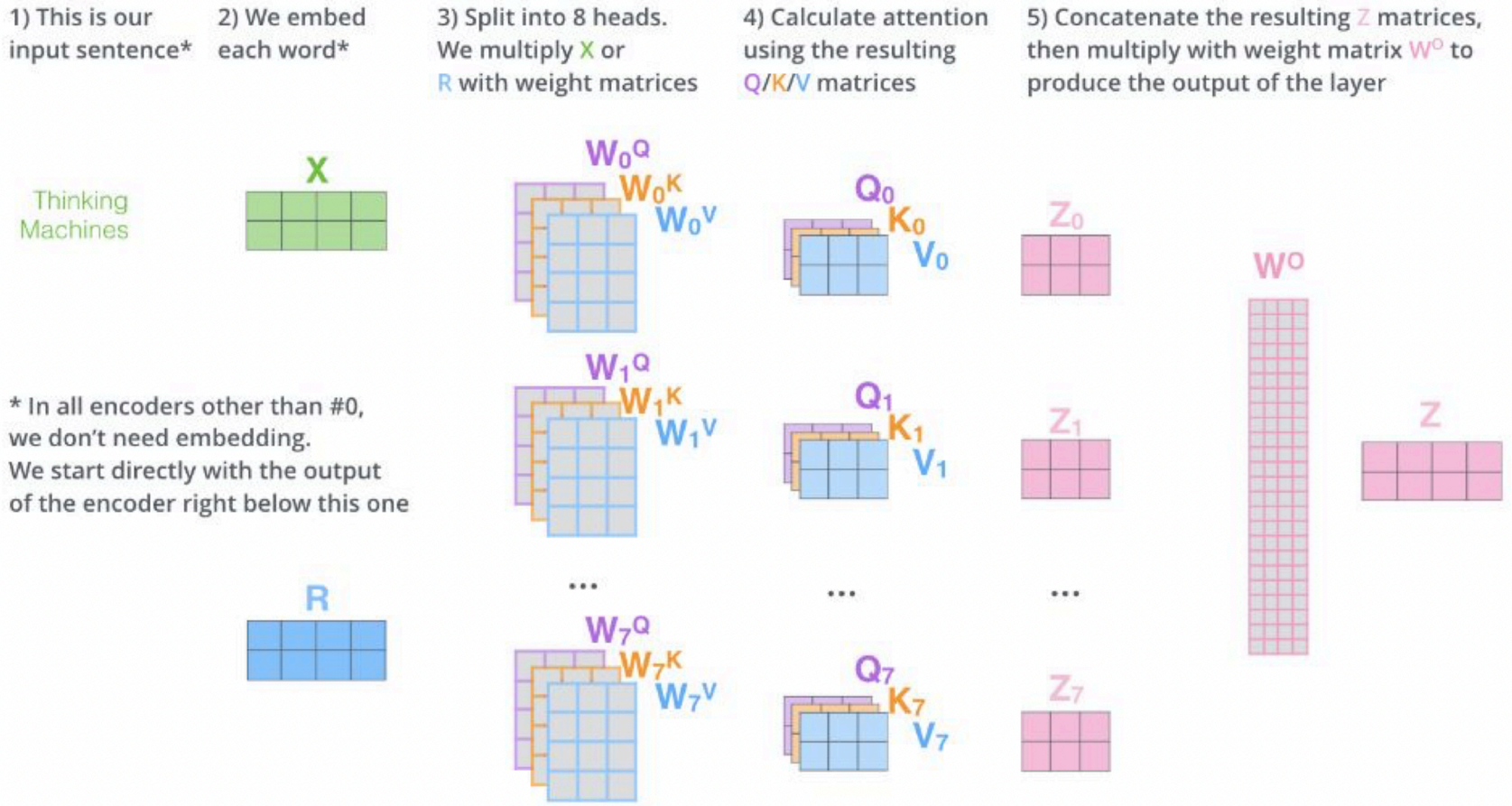
Learnable parameter matrices



Multi-Head Attention

Key points:

- We calculate 8 different attention heads, but we need to combine them
- Attention heads are independent of each other



https://jalammar.github.io/illustrated-transformer/

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

Thanks for your participation!

Çağrı Toraman
02.12.2025