Deep Learning II

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 ambiance was just as good as the food and service.

Quote from Prince, Understanding Deep Learning, 2023

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** ambiance was just as good as the food and service.

What does each **bolded** word refer to?

Is this a positive review or a negative one?

What kind of food do they serve at this restaurant?

Quote from Prince, Understanding Deep Learning, 2023

Types of NLP problems

Text Classification
- Sentiment Analysis
- Topic categorization
- Extractive question answering

Sequence-to-sequence
- Machine translation (text-to-text)
- Summarization (text-to-text)
- Speech-to-text, text-to-speech

- **Generative models**
  - Chatbots
  - Generative question answering
  - Code generation

Encoder models

BERT (Bidirectional Encoder Representations from Transformers)

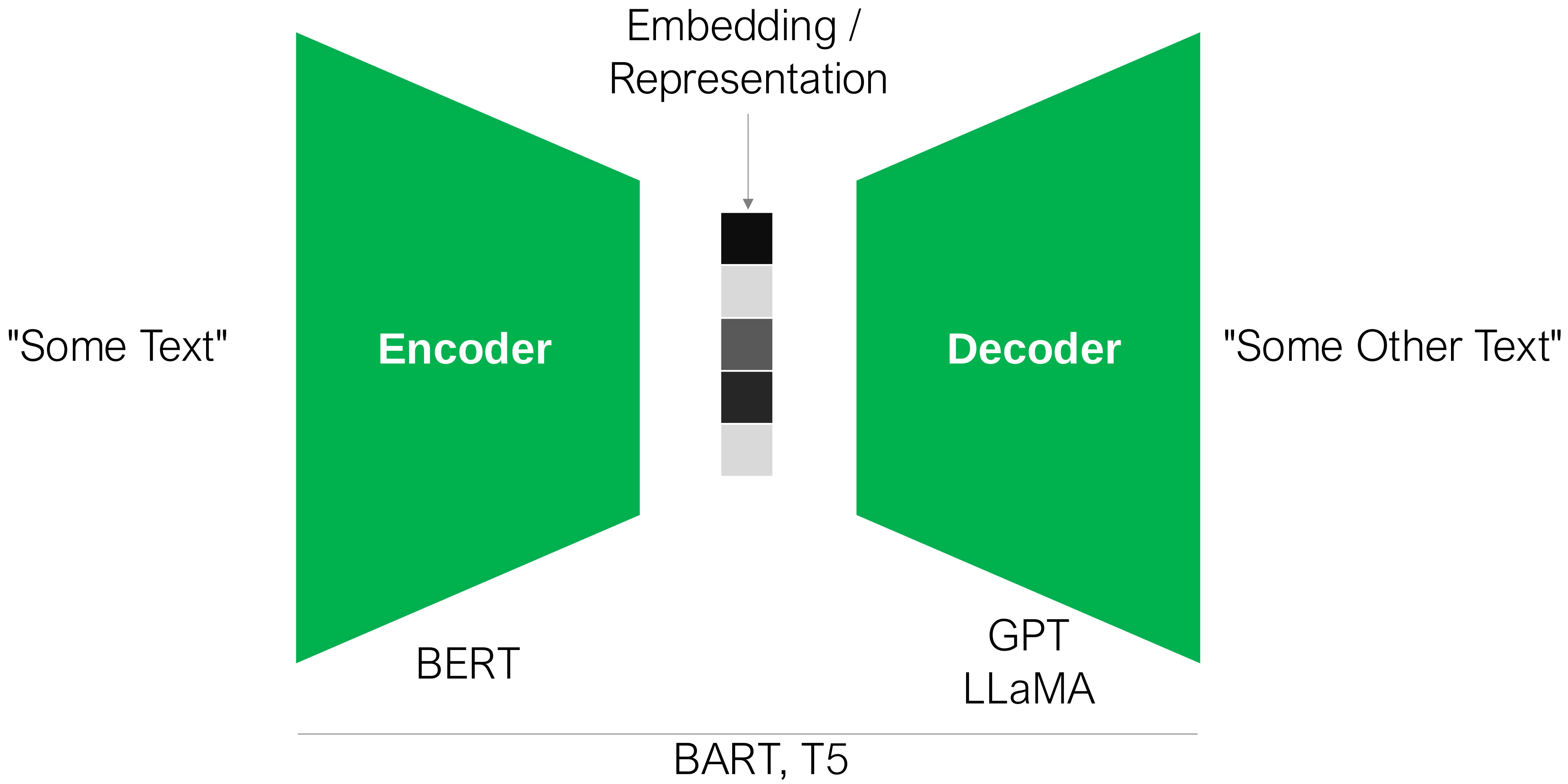
Encoder-decoder models

BART (Bidirectional and Auto-Regressive Transformers)
T5 (Text-to-Text Transfer Transformer)

Decoder models

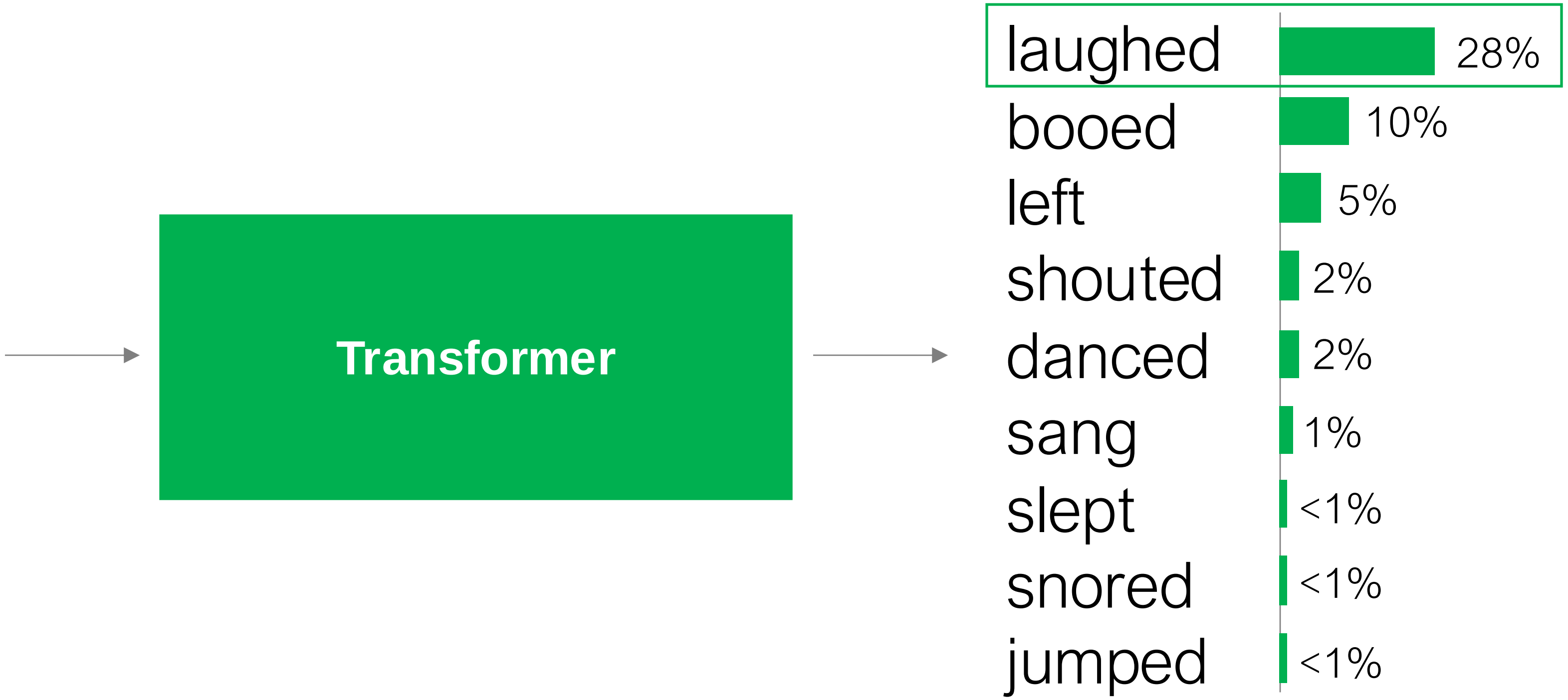
GPT (Generative Pre-trained Transformer)
LLaMA (Large Language Model Meta AI)

Types of Transformers



Example: Predict the next word

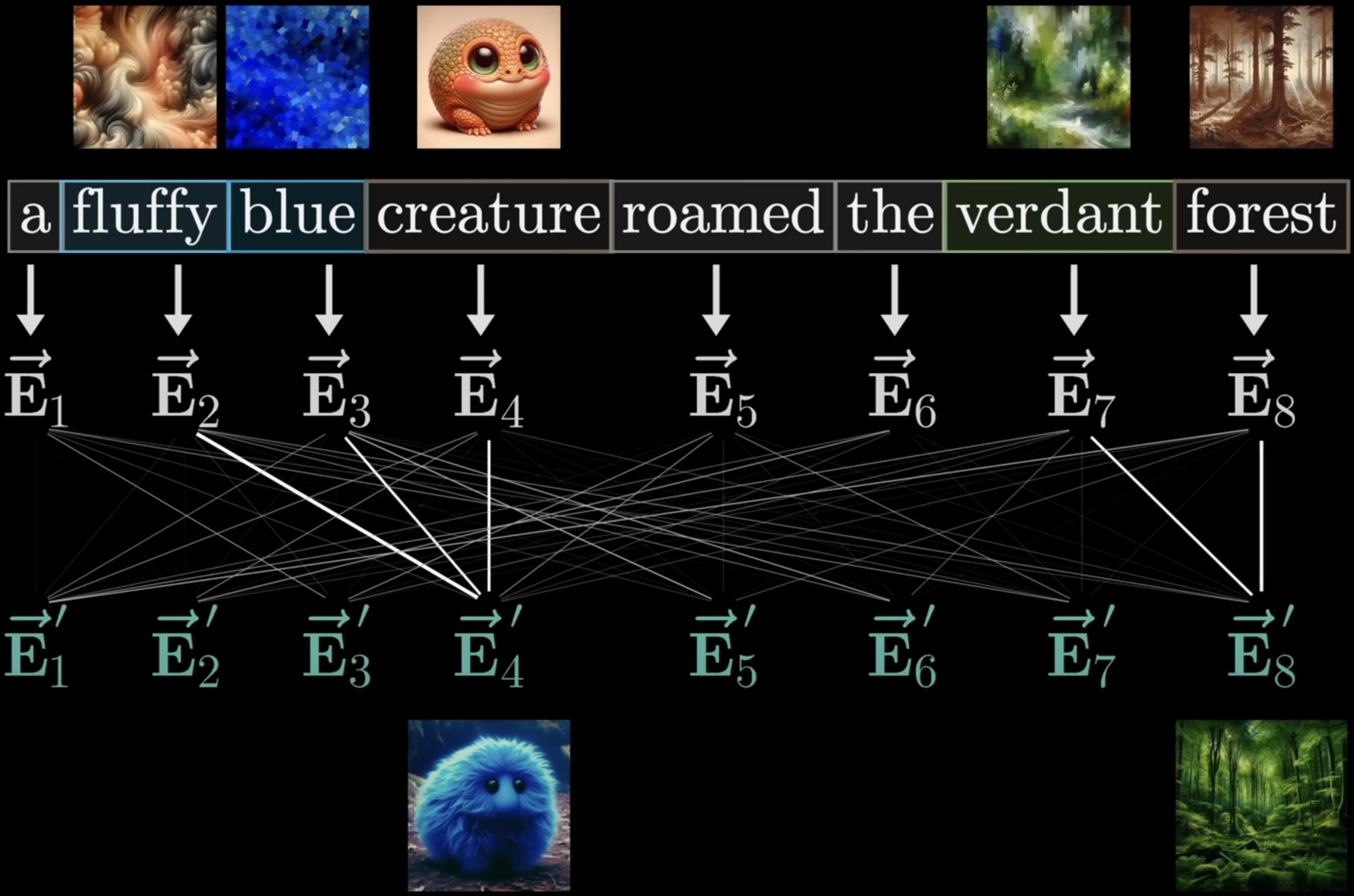
After hearing the terrible pun, the data science students, to the surprise of their instructor, all suddenly ________



If we repeat this again and again, we can generate text

Transformer intuition

The quick brown fox jumped over the lazy dog



Create improved embeddings by incorporating meaning from other words

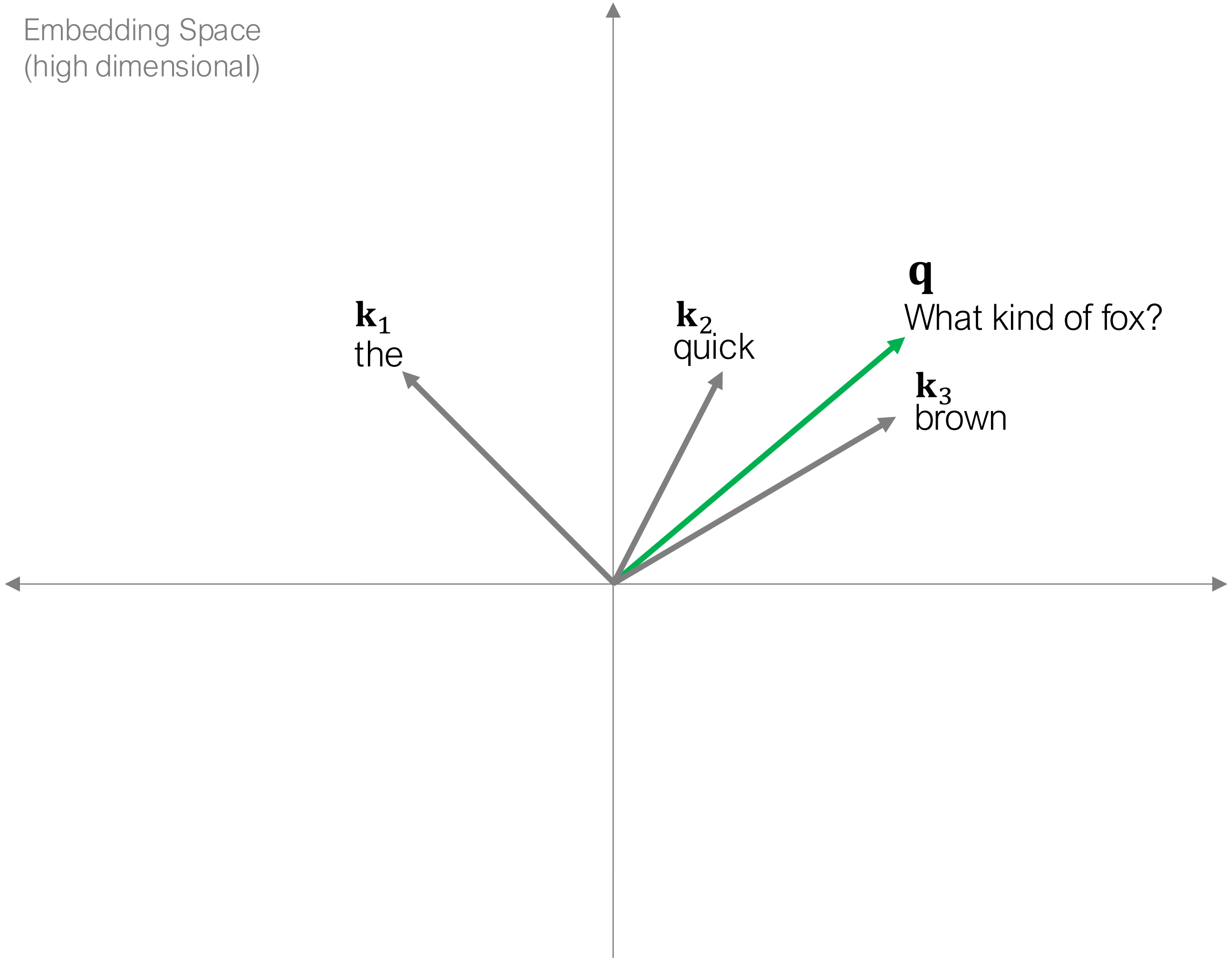
Image from 3Blue1Brown (link)

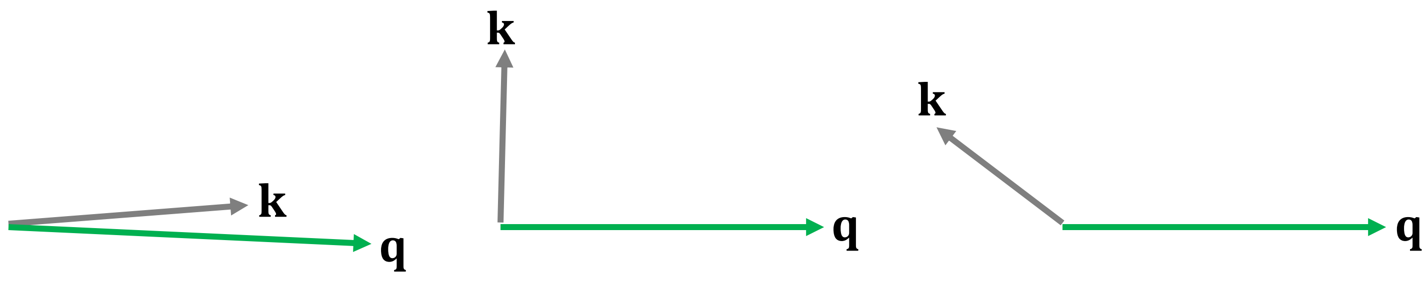
Queries and Keys

The quick brown fox jumped over the lazy dog

We measure the similarity between the key ($\mathbf{k}$) and the query ($\mathbf{q}$) through the dot product:

$$\mathbf{k} \cdot \mathbf{q} = \mathbf{k}^T \mathbf{q}$$





$\mathbf{k}^T \mathbf{q}$ positive

$\mathbf{k}^T \mathbf{q}$ zero

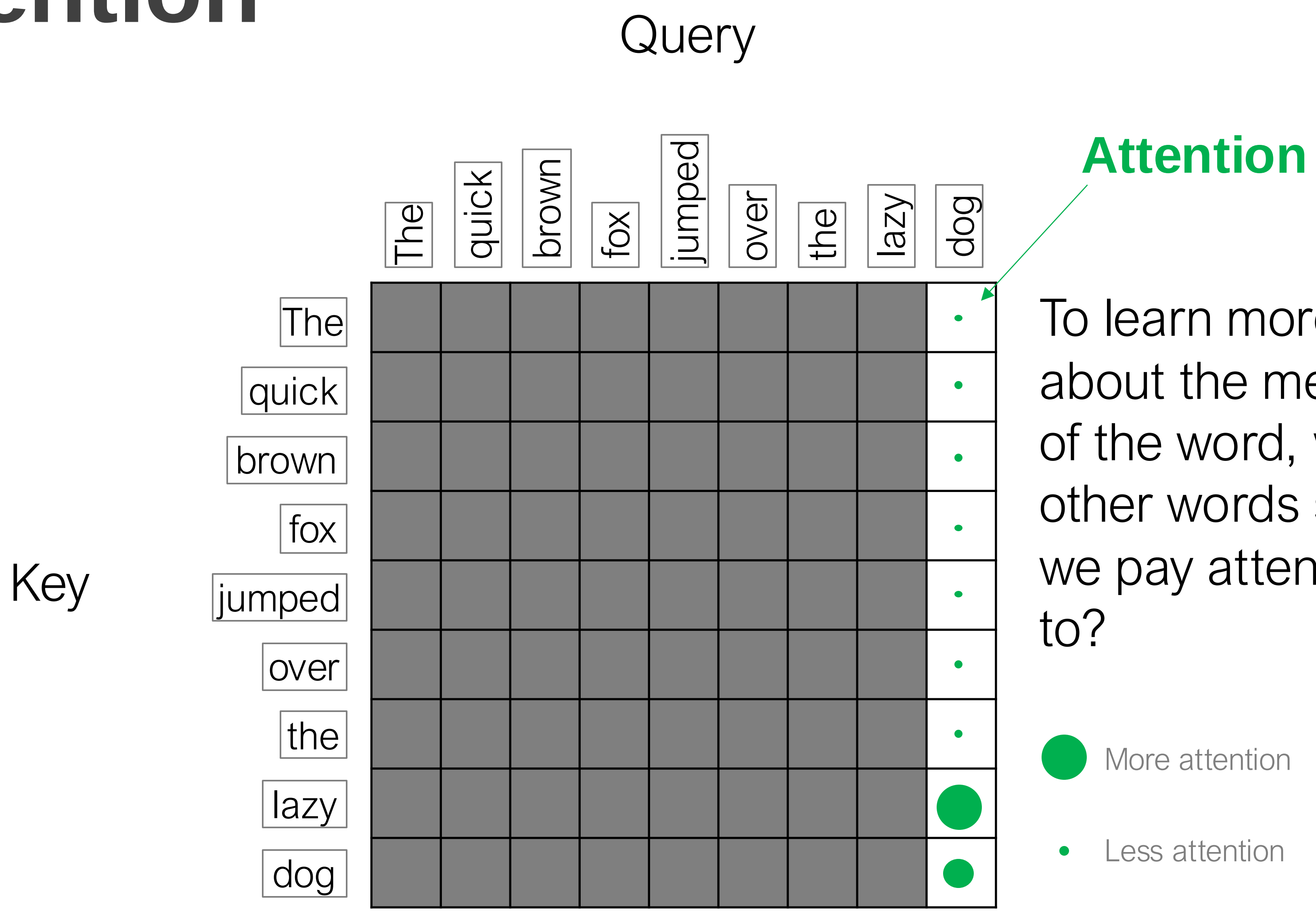
$\mathbf{k}^T \mathbf{q}$ negative

This tells the model what to pay attention to for a particular concept

Transformer steps and components

1. Tokenization – convert text to numbers
2. Position embedding – encode the order of the tokens
3. Self-attention – enable learning from context
    1. Queries
    2. Keys
    3. Values
4. Layer normalization
5. Multilayer perceptron

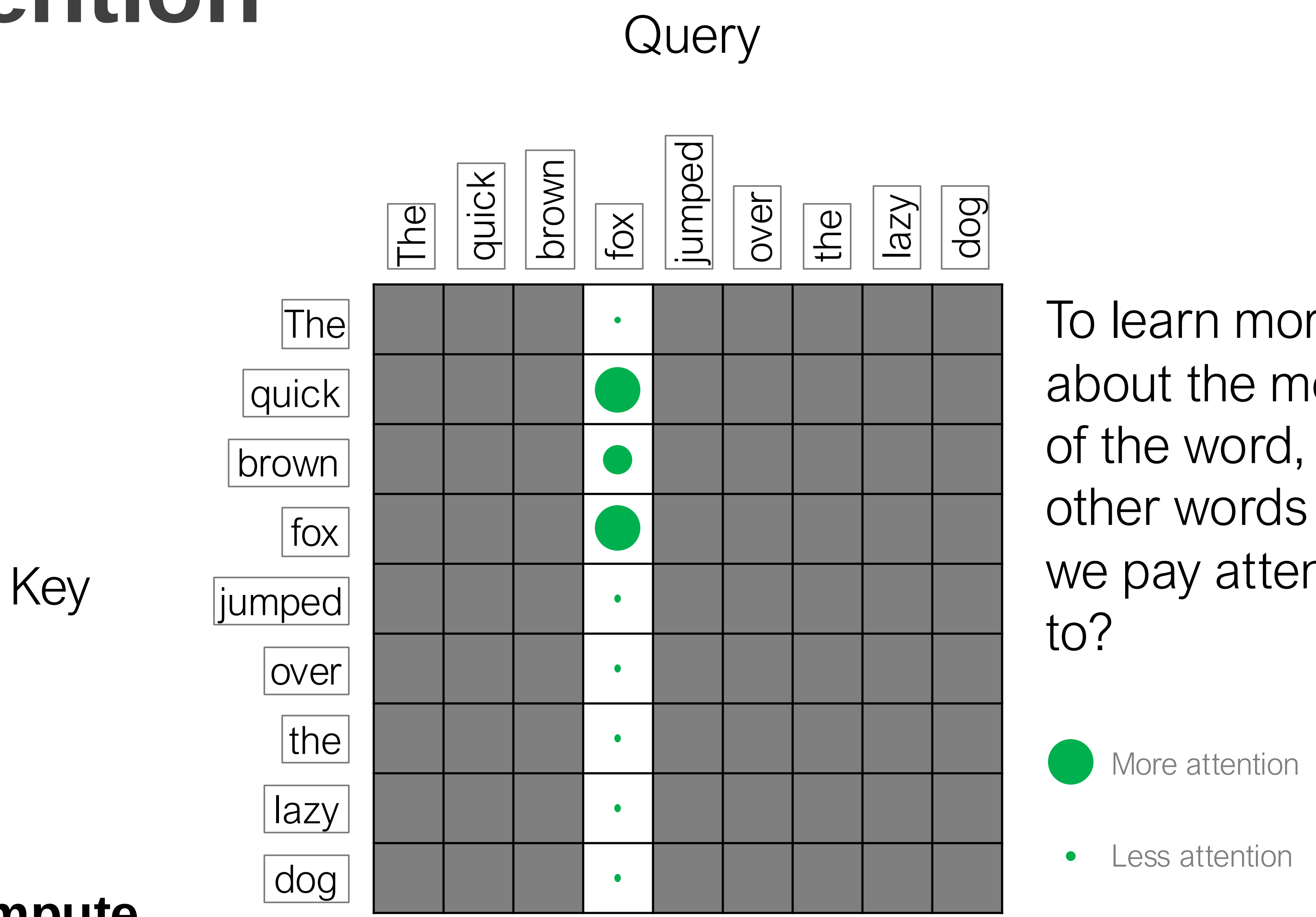
Self-attention



To learn more about the meaning of the word, which other words should we pay attention to?

Figure adapted from 3Blue1Brown (link)

Self-attention

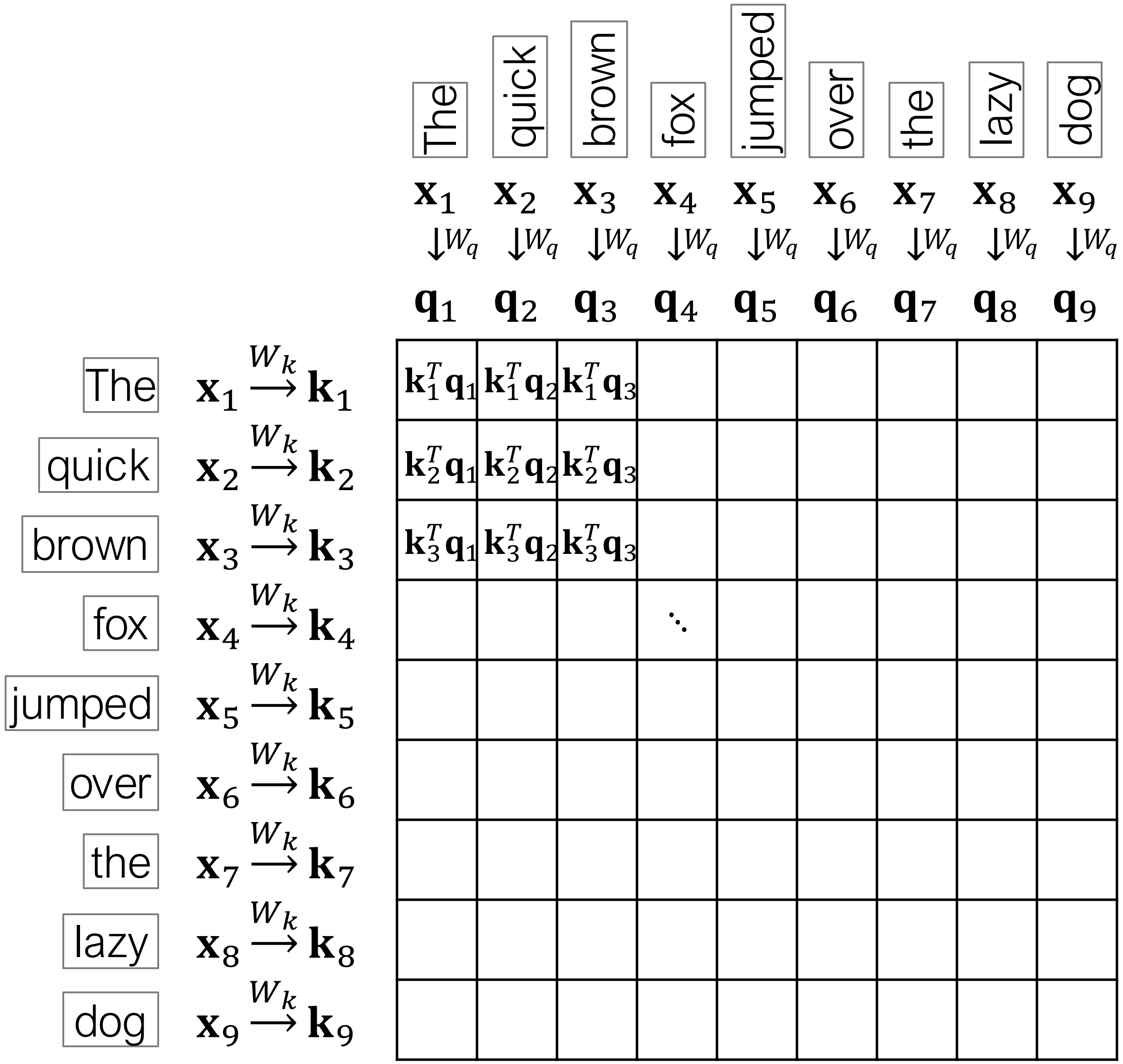


To learn more about the meaning of the word, which other words should we pay attention to?

How do we compute self-attention?

Figure adapted from 3Blue1Brown (link)

Query



Keys $\mathbf{k}_i = W_k \mathbf{x}_i$

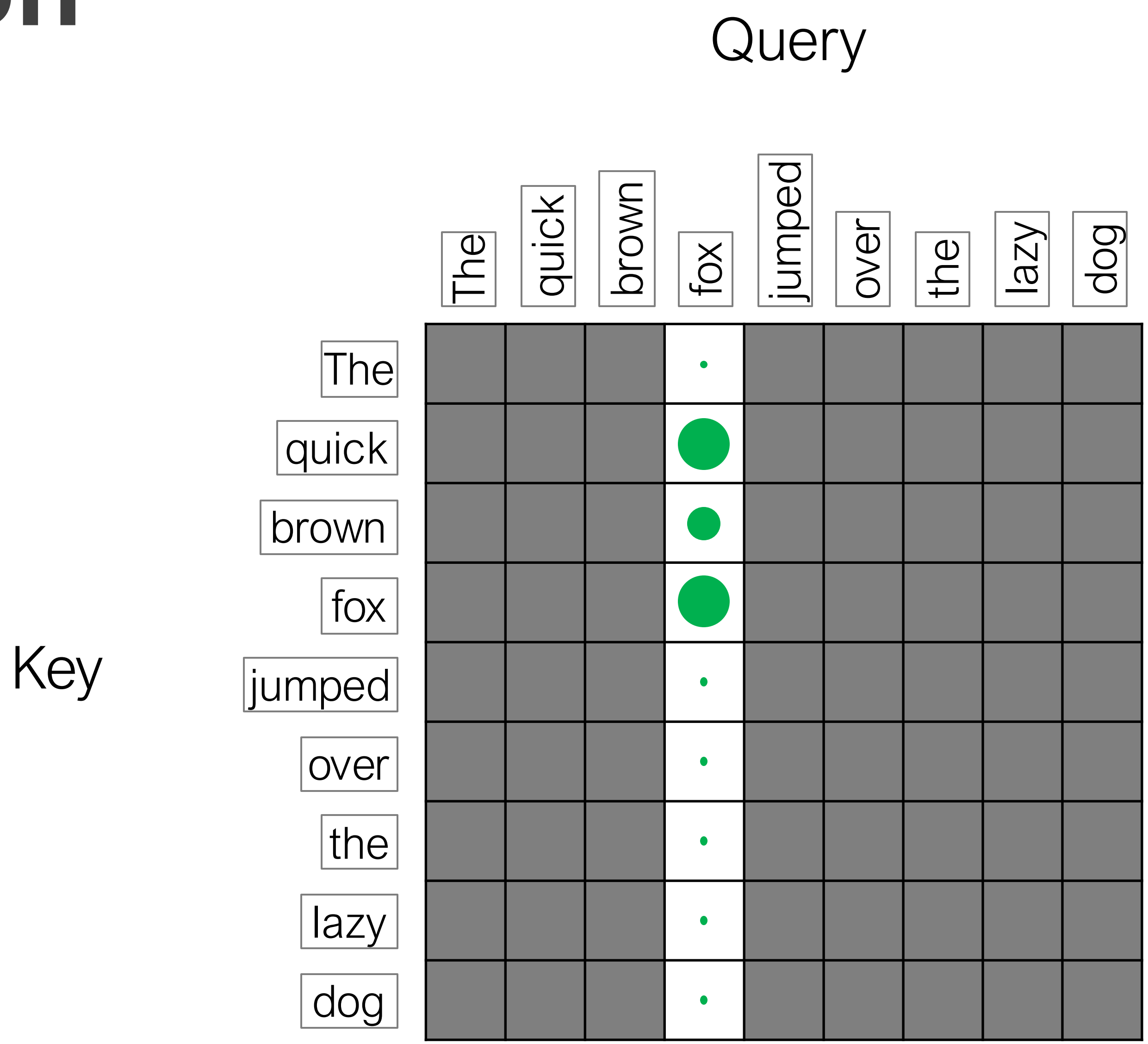
$\mathrm{K} = [\mathbf{k}_1, \mathbf{k}_2, \dots, \mathbf{k}_9]$

Queries $\mathbf{q}_i = W_q \mathbf{x}_i$

$\mathrm{Q} = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_9]$

Figure adapted from 3Blue1Brown (link)

Attention



To learn more about the meaning of the word, which other words should we pay attention to?

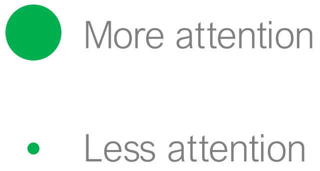
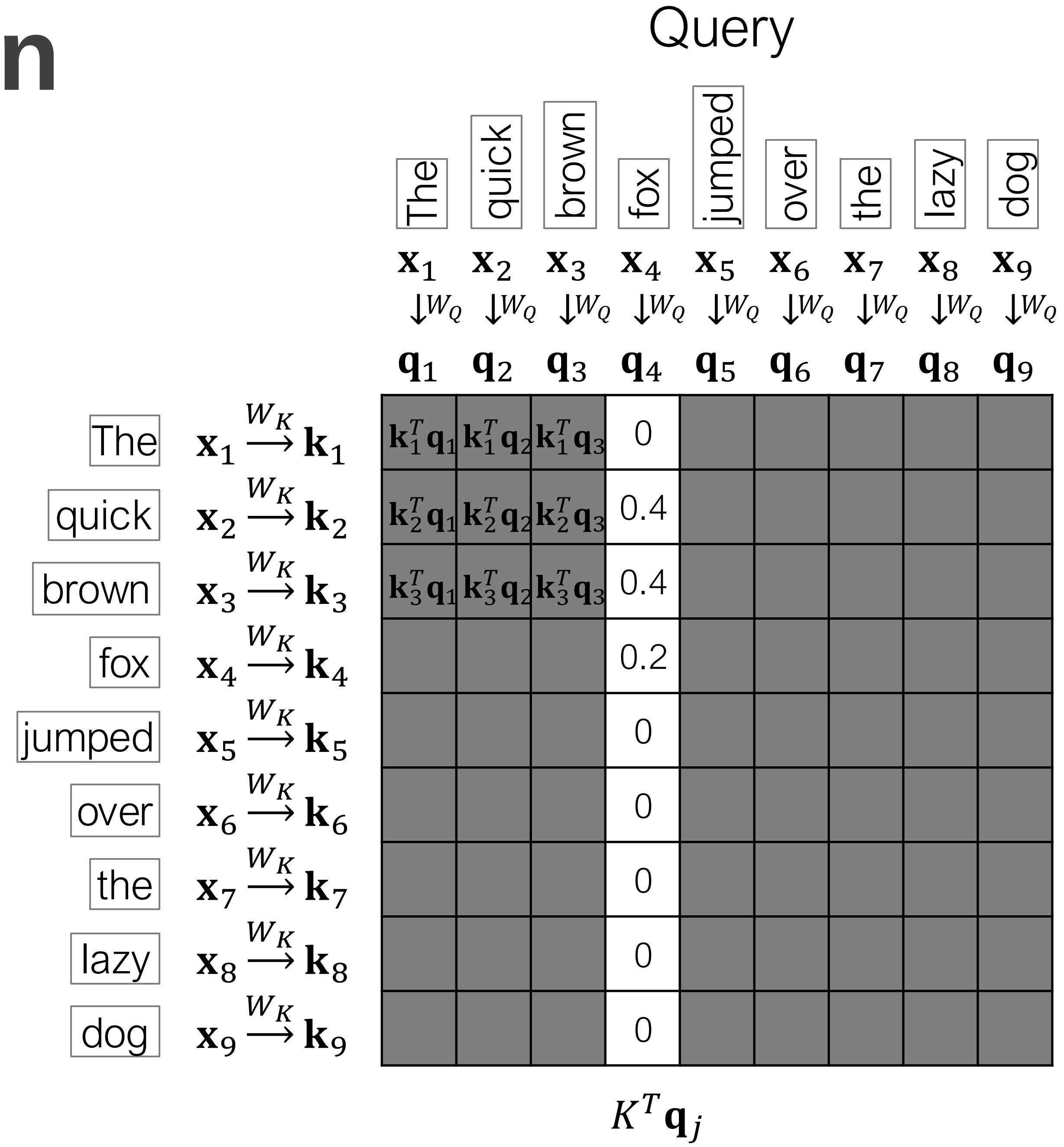


Figure adapted from 3Blue1Brown (link)

Attention



Keys $\mathbf{k}_i = W_K\mathbf{x}_i$

$\mathrm{K} = [\mathbf{k}_1, \mathbf{k}_2, \dots, \mathbf{k}_9]$

Queries $\mathbf{q}_i = W_Q\mathbf{x}_i$

$\mathrm{Q} = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_9]$

Attention weights

$a[\mathbf{x}_i, \mathbf{x}_i] = \mathrm{softmax}_i\left(K^T\mathbf{q}_j\right)$

Note: we apply softmax to each column so it sums to 1

Figure adapted from 3Blue1Brown (link)

Self-attention

Weighted sum of values

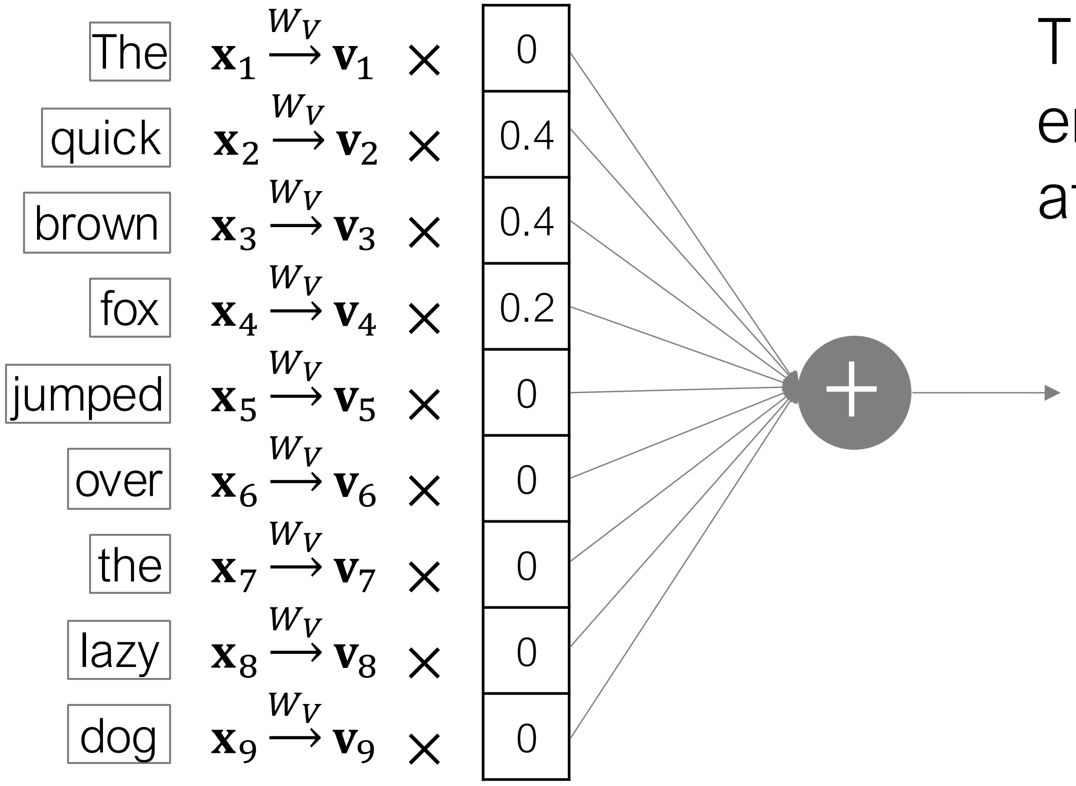
(based on self-attention weights)

Values

$$\mathbf{v}_i = W_V \mathbf{x}_i$$

$$\mathrm{V} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_9]$$

$a[\mathbf{x}_i, \mathbf{x}_i]$



This simply says – weight the embeddings based on the self-attention weights

$$\mathbf{sa}_n[x_1, \dots, x_N] = \sum_{m=1}^{N} a[\mathbf{x}_m, \mathbf{x}_n]\mathbf{v}_m$$

Figure adapted from 3Blue1Brown (link)

Self-attention

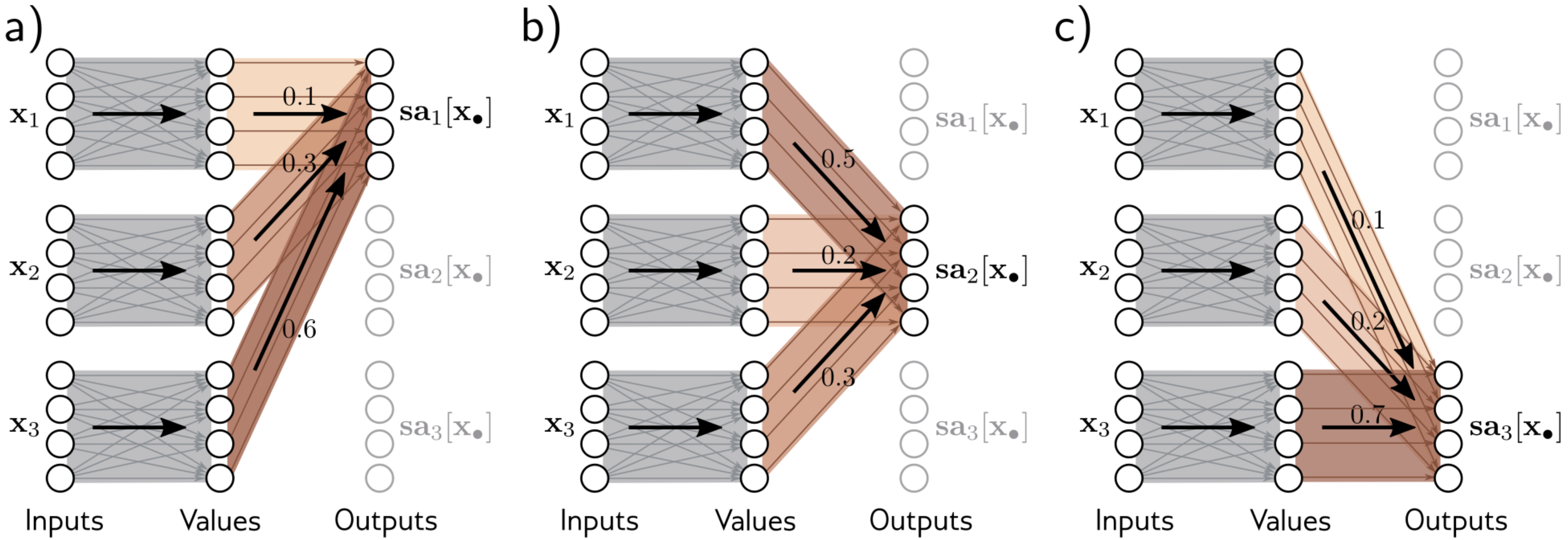


Image from Prince, Understanding Deep Learning, 2023

Self-attention in matrix form

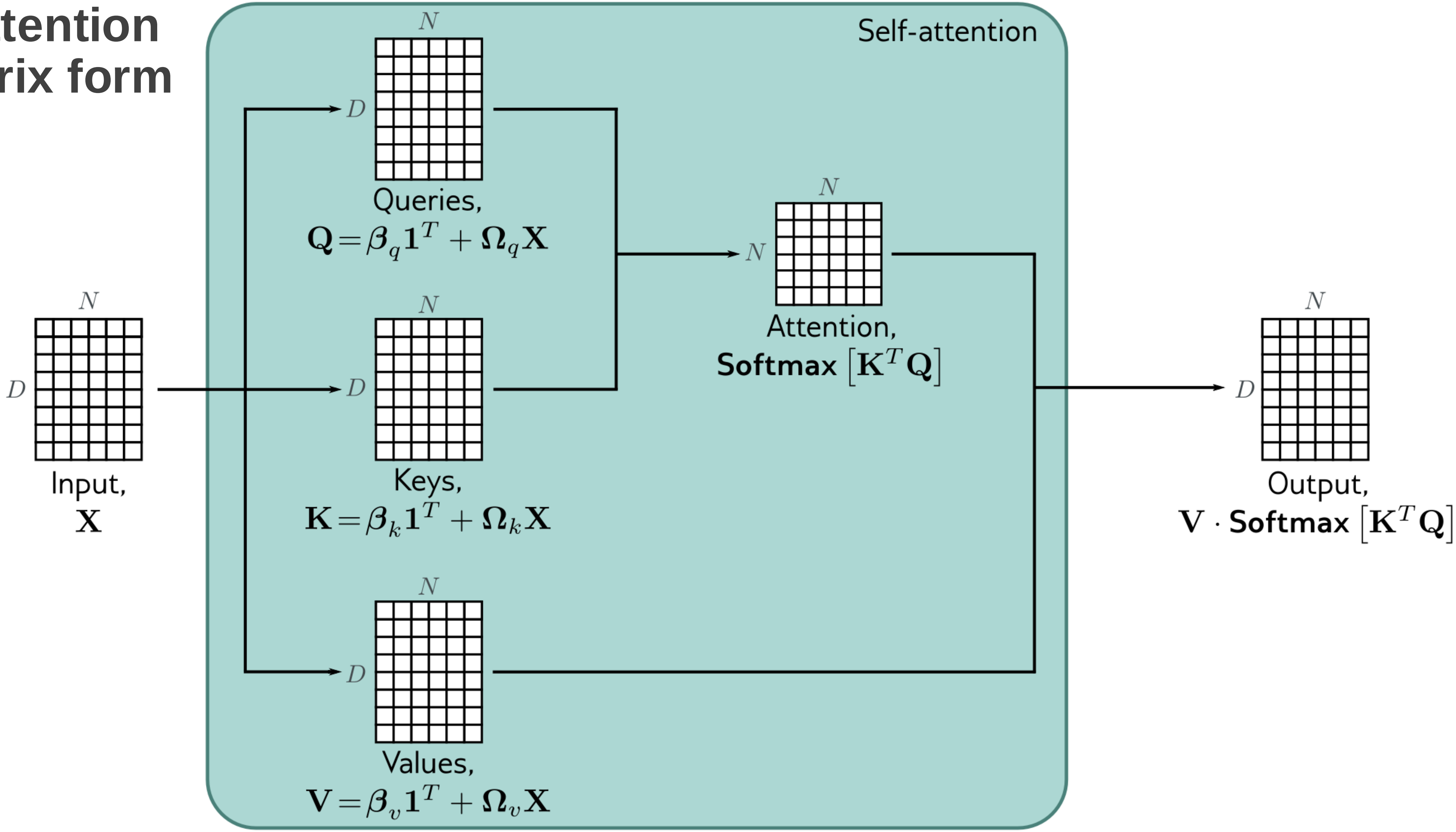


Image from Prince, Understanding Deep Learning, 2023

Multiheaded attention

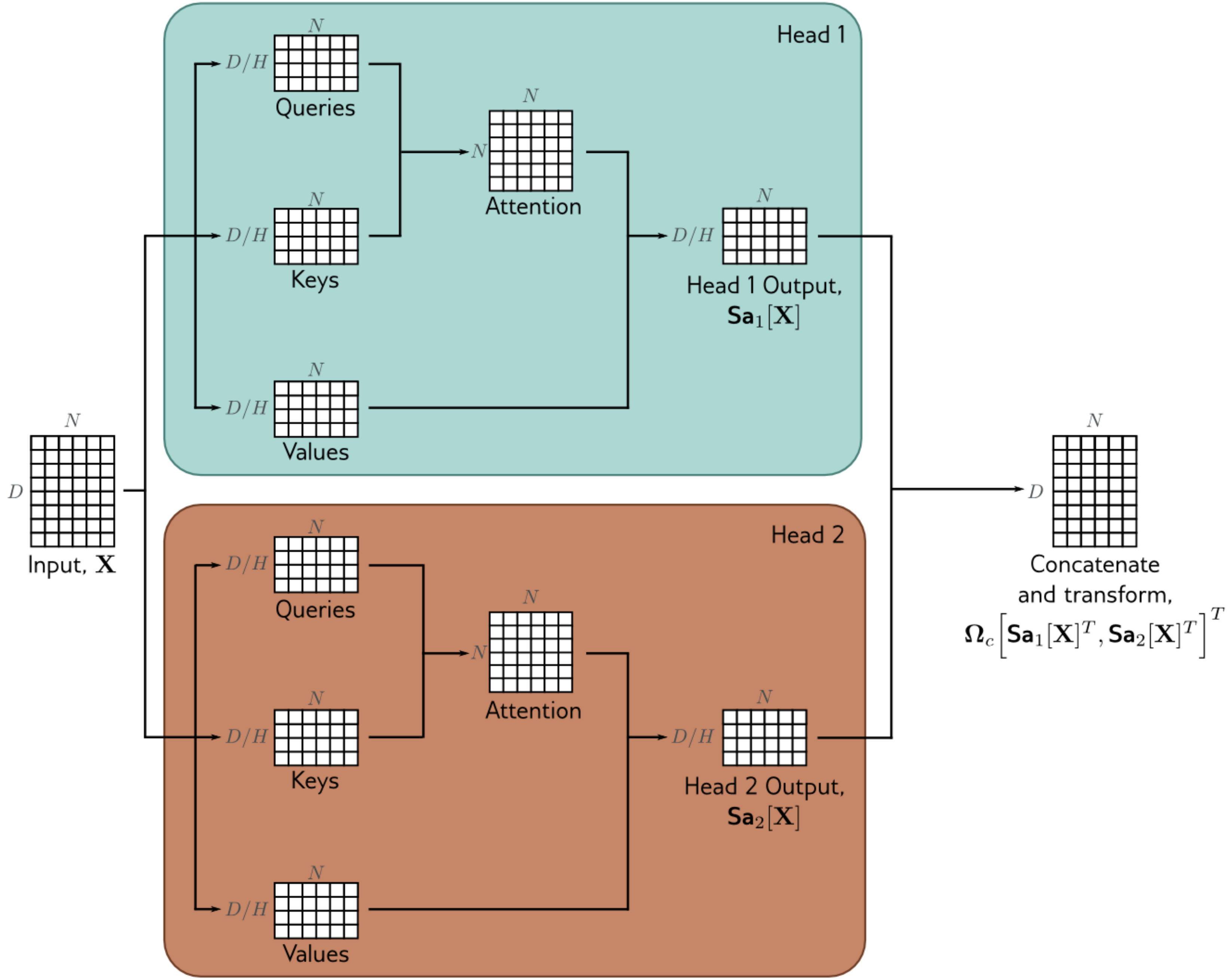


Image from Prince, Understanding Deep Learning, 2023

Tokenization

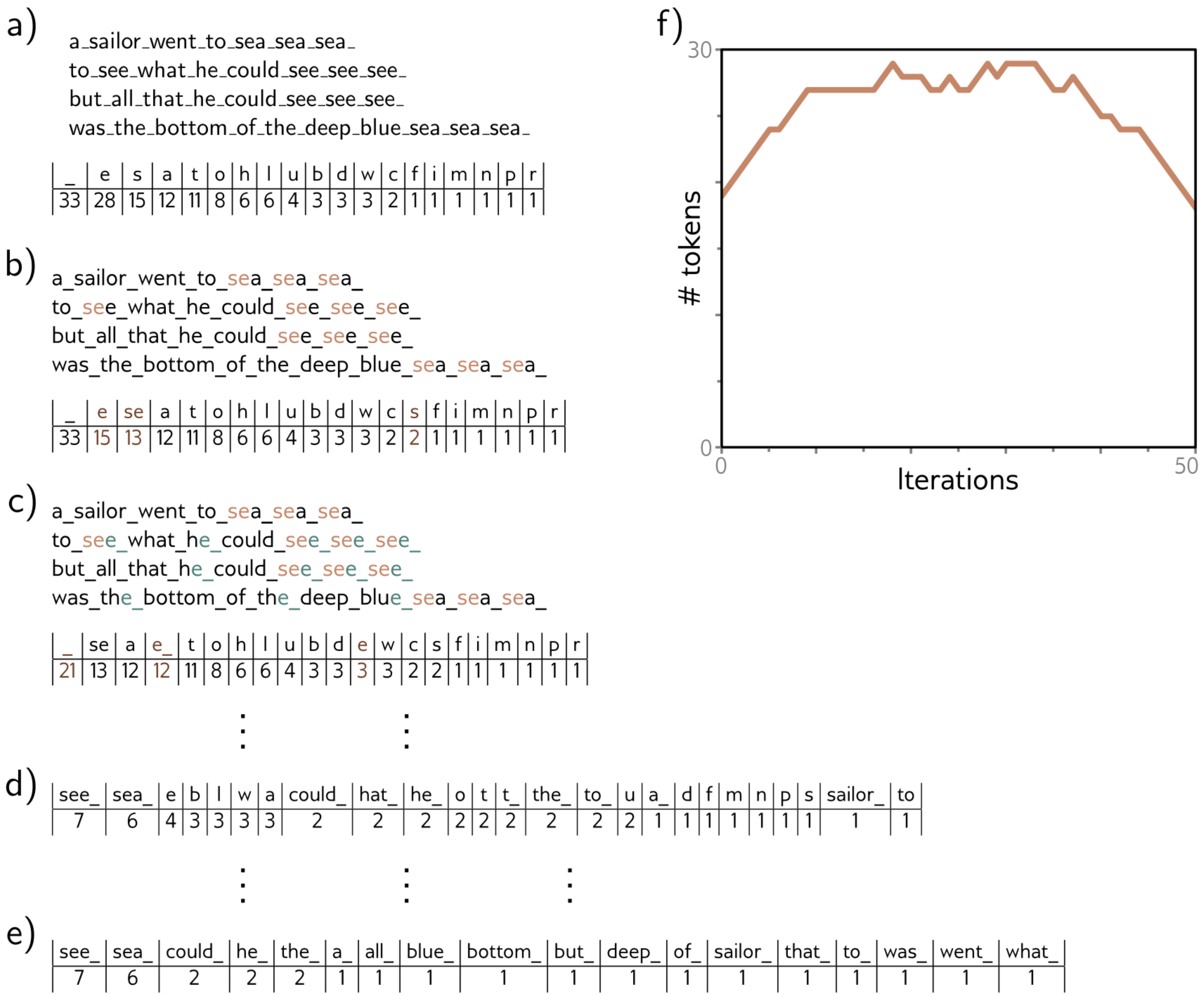
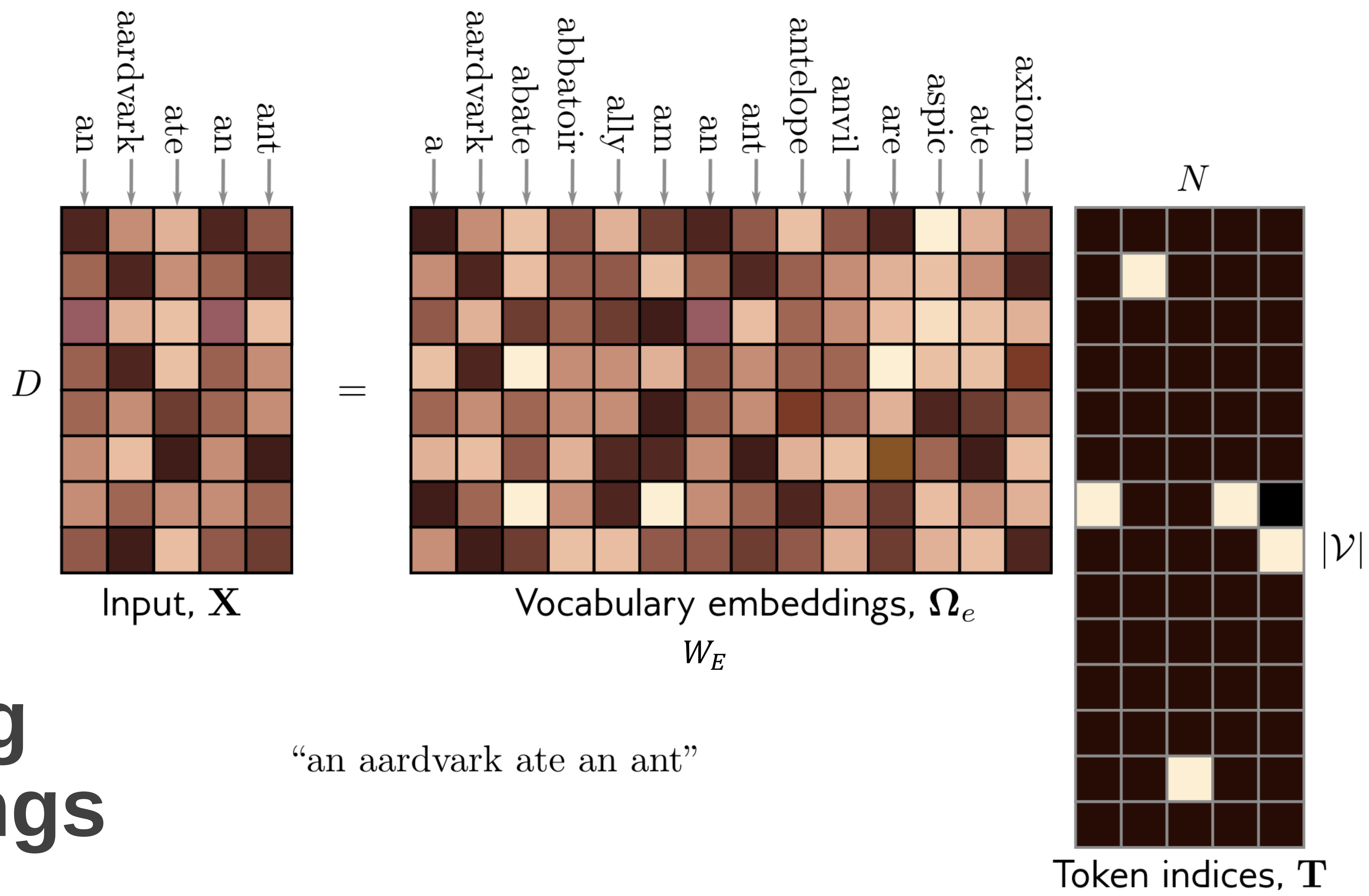


Image from Prince, Understanding Deep Learning, 2023



"an aardvark ate an ant"

Producing embeddings

Image from Prince, Understanding Deep Learning, 2023

Positional Encoding

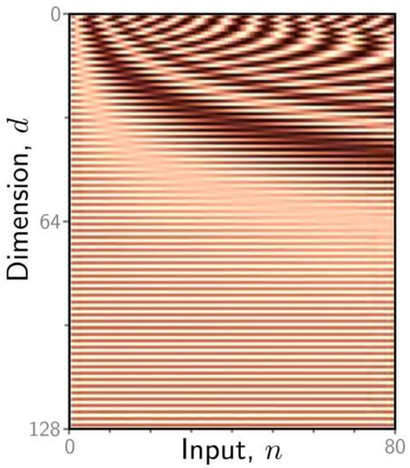
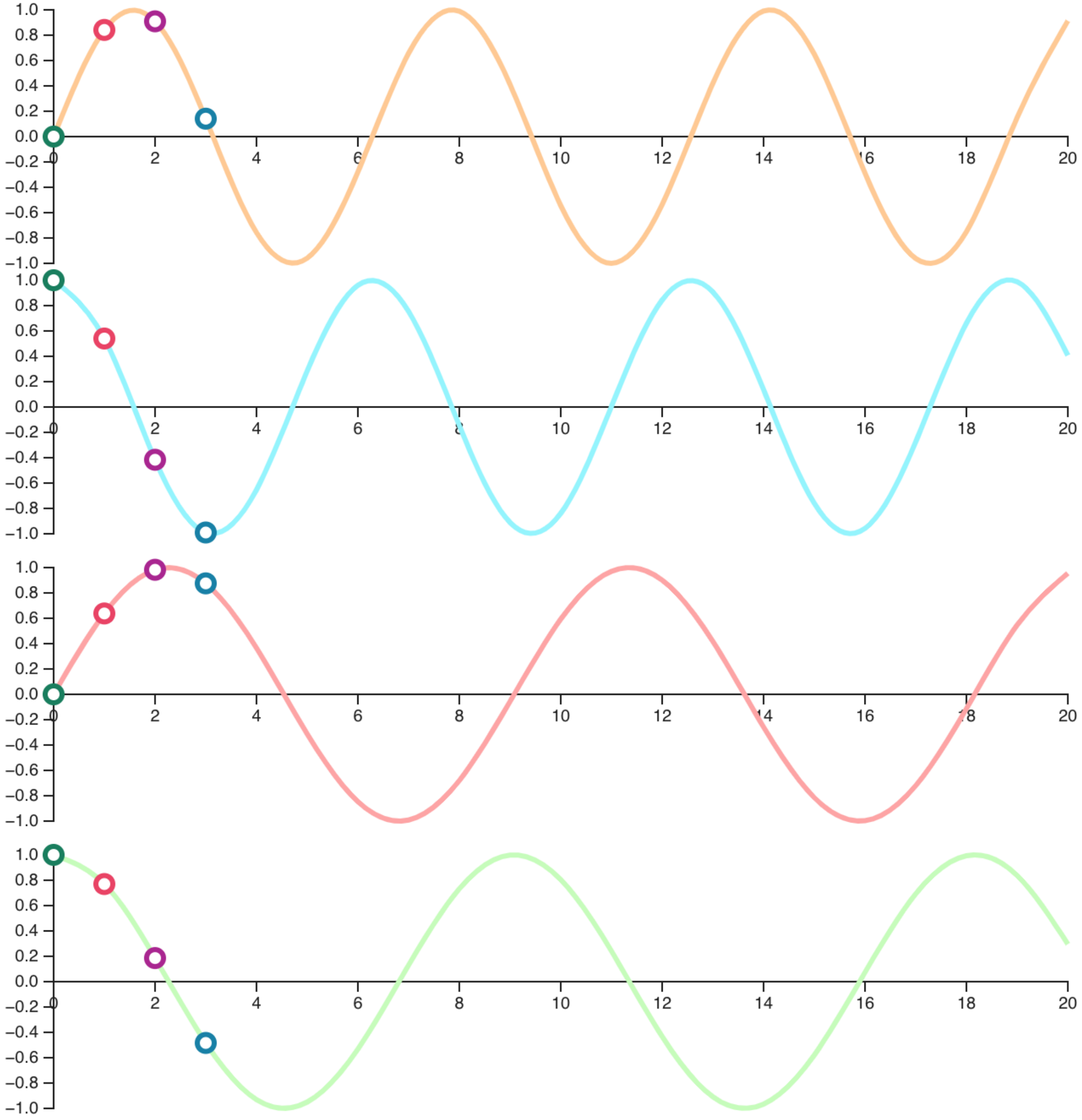


Image from Prince, Understanding Deep Learning, 2023



	p0	p1	p2	p3
i=0	0.000	0.841	0.909	0.141
i=1	1.000	0.540	-0.416	-0.990
i=2	0.000	0.638	0.983	0.875
i=3	1.000	0.770	0.186	-0.484

Positional Encoding

$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{2i/d_{\text{model}}}})$$

$$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{2i/d_{\text{model}}}})$$

Settings: d = 50

The value of each positional encoding depends on the *position* (*pos*) and *dimension* (*d*). We calculate result for every *index* (*i*) to get the whole vector.

Image source: https://erdem.pl/2021/05/understanding-positional-encoding-in-transformers

The Transformer

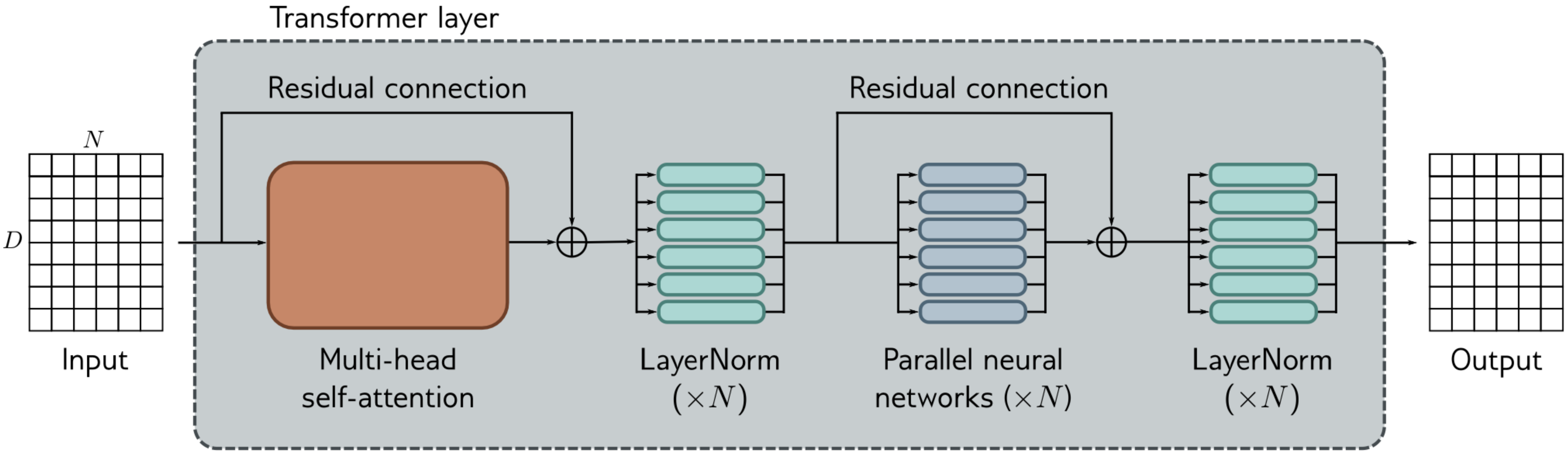


Image from Prince, Understanding Deep Learning, 2023

You can learn an "unembedding" matrix to then map the output to the full vocabulary list, apply softmax, and using that generate the next entry in the text sequence

Vision Transformer (ViT)

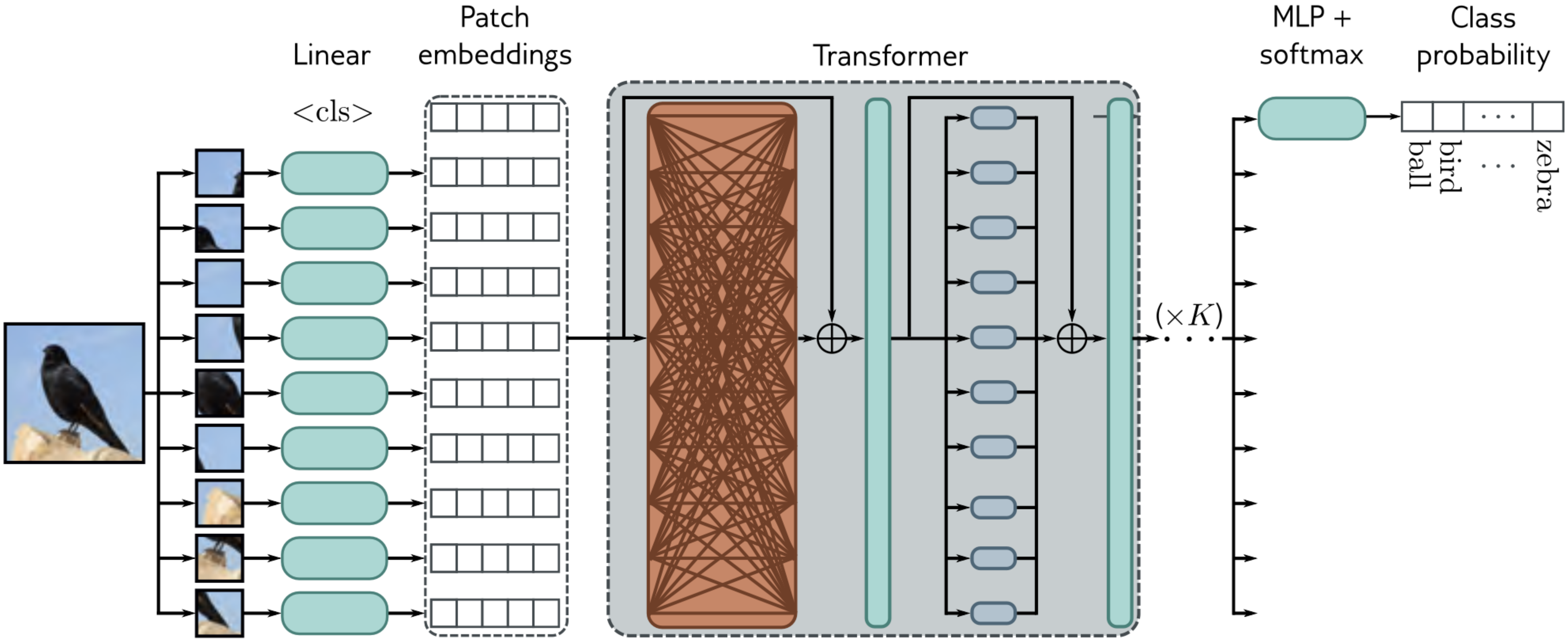


Image from Prince, Understanding Deep Learning, 2023

Supervised Learning Techniques

Covered so far

- Linear Regression
- K-Nearest Neighbors
- Perceptron
- Logistic Regression
- Linear Discriminant Analysis
- Quadratic Discriminant Analysis
- Naïve Bayes
- Support Vector Machines
- Decision Trees and Random Forests
- Ensemble methods (bagging, boosting, stacking)
- Neural Networks

Appropriate for:
- Classification
- Regression

Can be used with many machine learning techniques