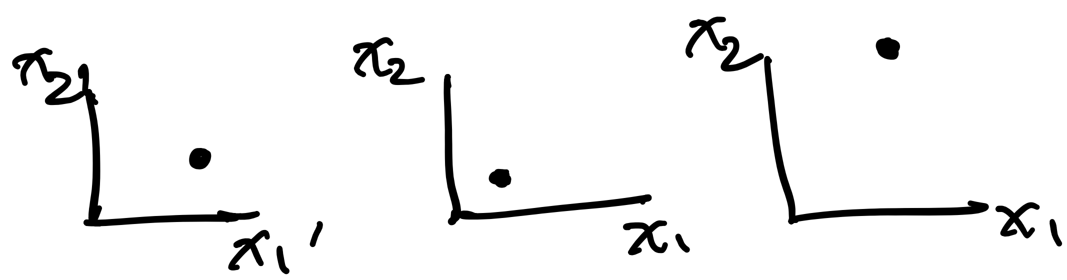
Tokenization

Thursday, November 7, 2024 8:10 AM

- Map a raw sequence to a sequence of tokens
  - "Train a neural network" → "Train", "a", "neural network"

- Map tokens to real- or vector-valued representation
  - "Train", "a", "neural network"

    ↳ 

    Tool: word2vec

  - ACGTAACG → (1,0,0,0), (0,1,0,0), (0,0,1,0), (0,0,0,1), (1,0,0,0), (1,0,0,0) ...

Attention (SLP 9)

Sunday, November 3, 2024 12:11 PM

- Consider two sentences

The chicken didn't cross the road because it was too tired

The chicken didn't cross **the road** because it was too wide

  - "it" refers to "The chicken" in the 1st sentence, and "the road" in the 2nd

- For each token, "attention" is used to represent the contextual meaning given other tokens

Transformers (SLP 9)

Thursday, November 7, 2024 8:26 AM

Simplified "attention head"

- Let $\underline{x}_i$ be the input representation of token i
- The attention vector for token i is

$$\underline{a}_i = \sum_j \alpha_{ij} \underline{x}_j ,$$

a & x have same dimensionality

where α_{ij} is the similarity of representations $\underline{x}_i$ & $\underline{x}_j$:

$$\alpha_{ij} = e^{\underline{x}_i^T \underline{x}_j} / \sum_j e^{\underline{x}_i^T \underline{x}_j}$$

Softmax

- $\underline{a}_1, \underline{a}_2, \underline{a}_3 \ldots$ incorporate contextual information

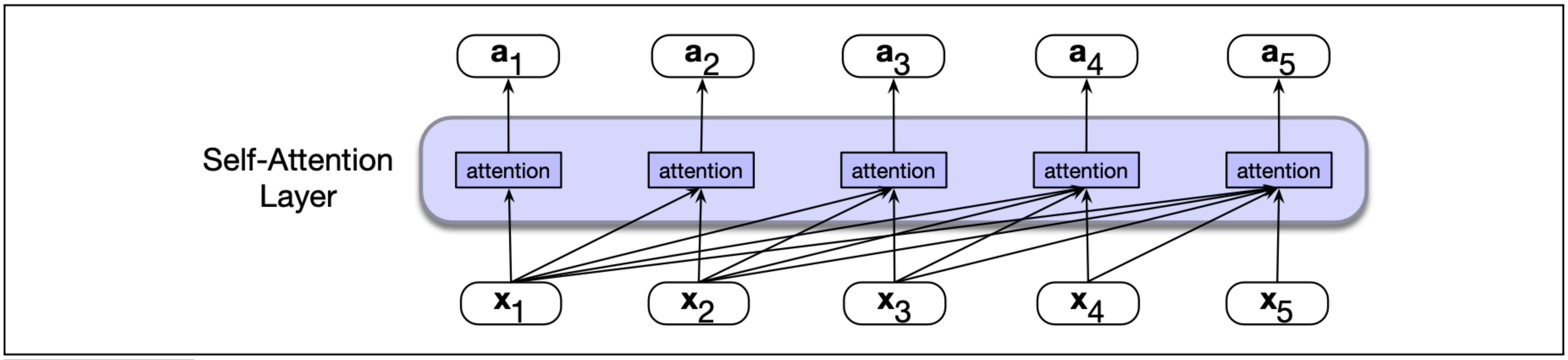


Figure 9.3 Information flow in causal self-attention. When processing each input $\mathbf{x}_i$, the model attends to all the inputs up to, and including $\mathbf{x}_i$.

$$\underline{a}_i = \sum_j \alpha_{ij} \underline{x}_j ,$$

$$\alpha_{ij} = e^{\underline{x}_i^T \underline{x}_j} / \sum_j e^{\underline{x}_i^T \underline{x}_j}$$

Transformers (SLP 9)

Thursday, November 7, 2024 8:39 AM

Actual "attention head"

- Query: Element for which we are creating context
- Key: Other elements used to provide context
- Value: Vectors representing the elements

Weight matrices, W's are trained

$$\underline{q}_i = W^Q \underline{x}_i, \quad \underline{k}_i = W^K \underline{x}_i, \quad \underline{v}_i = W^V \underline{x}_i,$$

map the vector for token i, $\underline{x}_i$ to vectors representing the query, key and value.

$$\alpha_{ij} = e^{\underline{q}_i^T \underline{k}_j / \sqrt{d_k}} \Big/ \sum_j e^{\underline{q}_i^T \underline{k}_j / \sqrt{d_k}}$$

d_k = dimensionality of q, k,

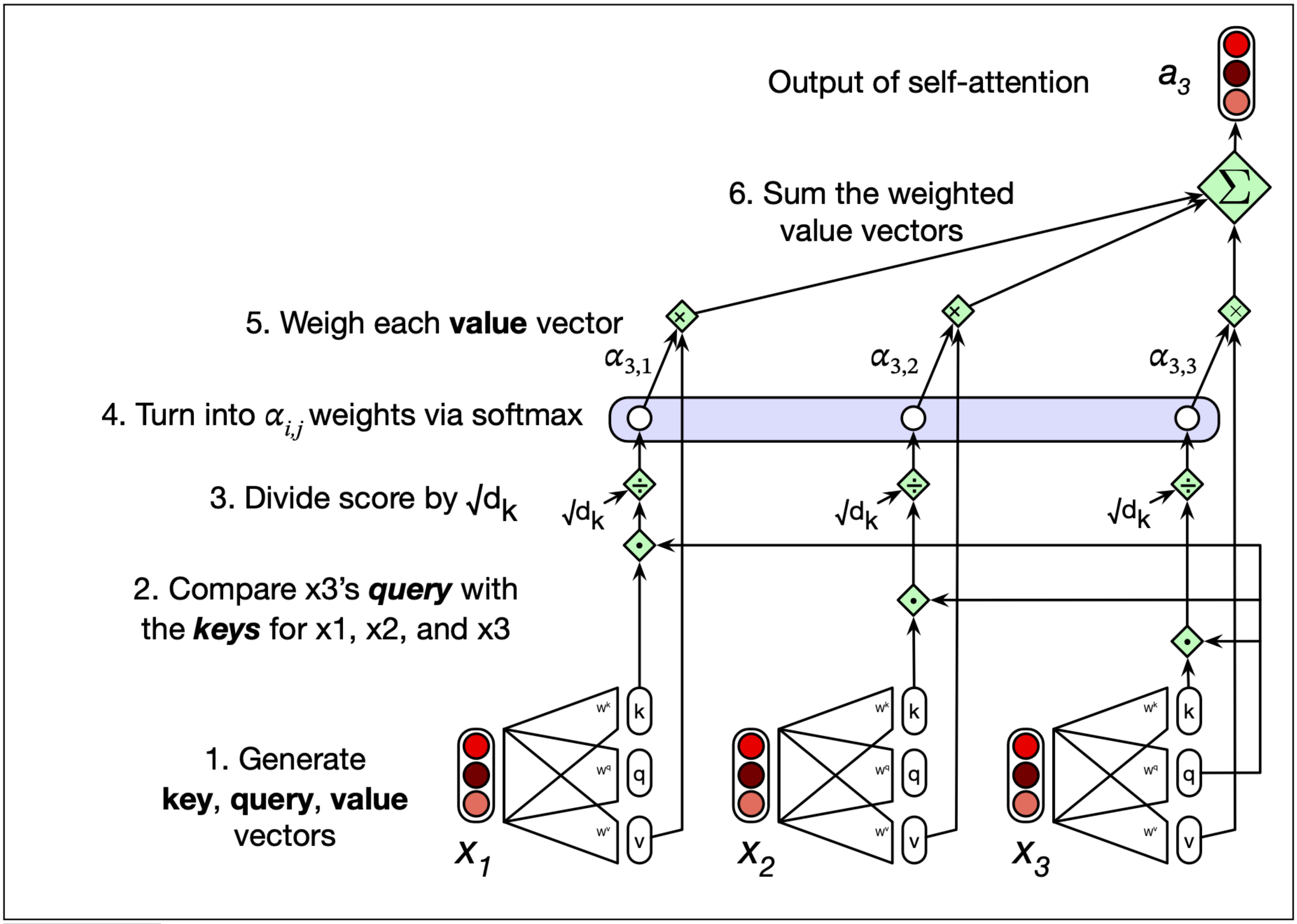


Figure 9.4 Calculating the value of $\mathbf{a}_3$, the third element of a sequence using causal (left-to-right) self-attention.

Multi-head attention

Thursday, November 7, 2024 8:53 AM

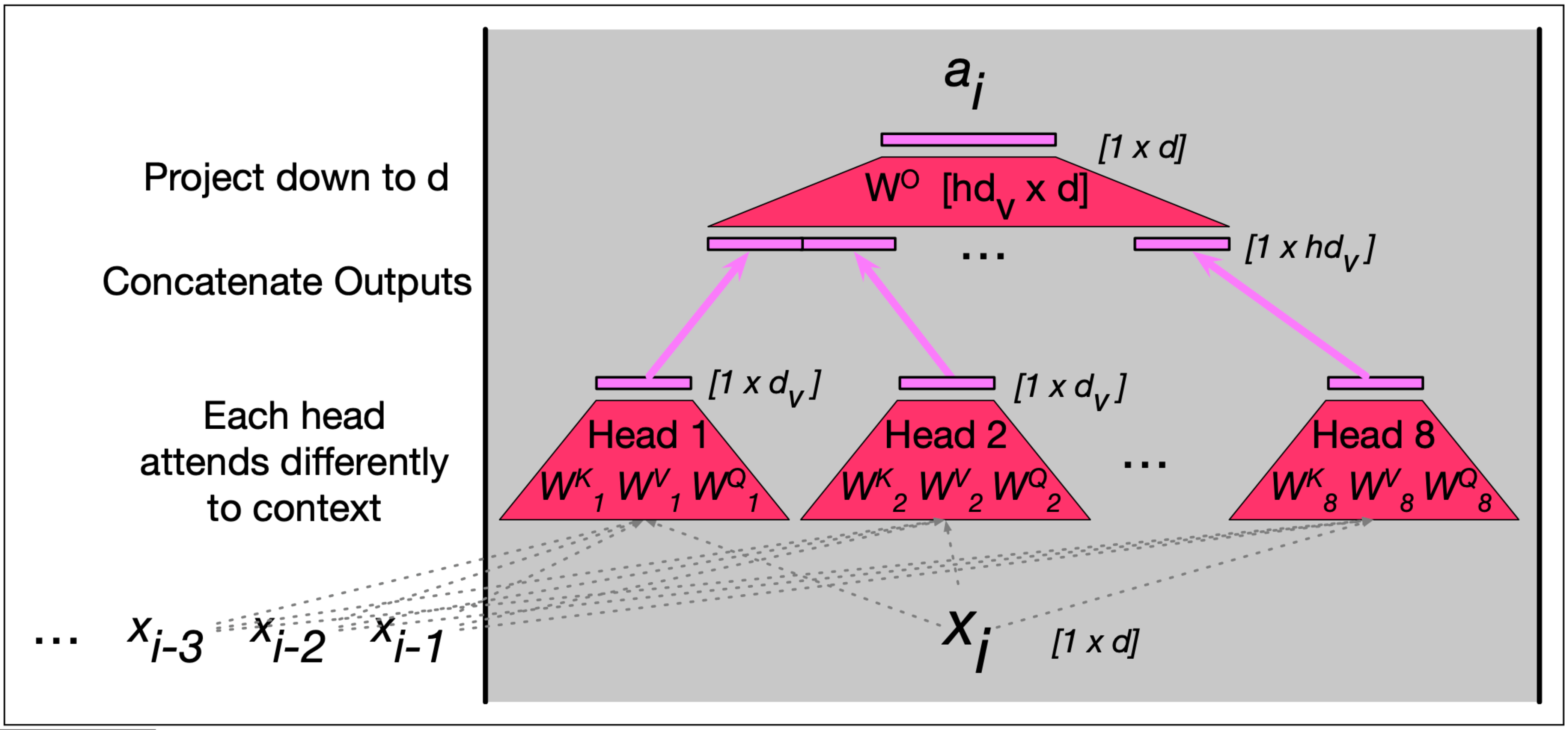


Figure 9.5 The multi-head attention computation for input $\mathbf{x}_i$, producing output $\mathbf{a}_i$. A multi-head attention layer has h heads, each with its own key, query and value weight matrices. The outputs from each of the heads are concatenated and then projected down to d, thus producing an output of the same size as the input.

Residual streams

Thursday, November 7, 2024 8:56 AM

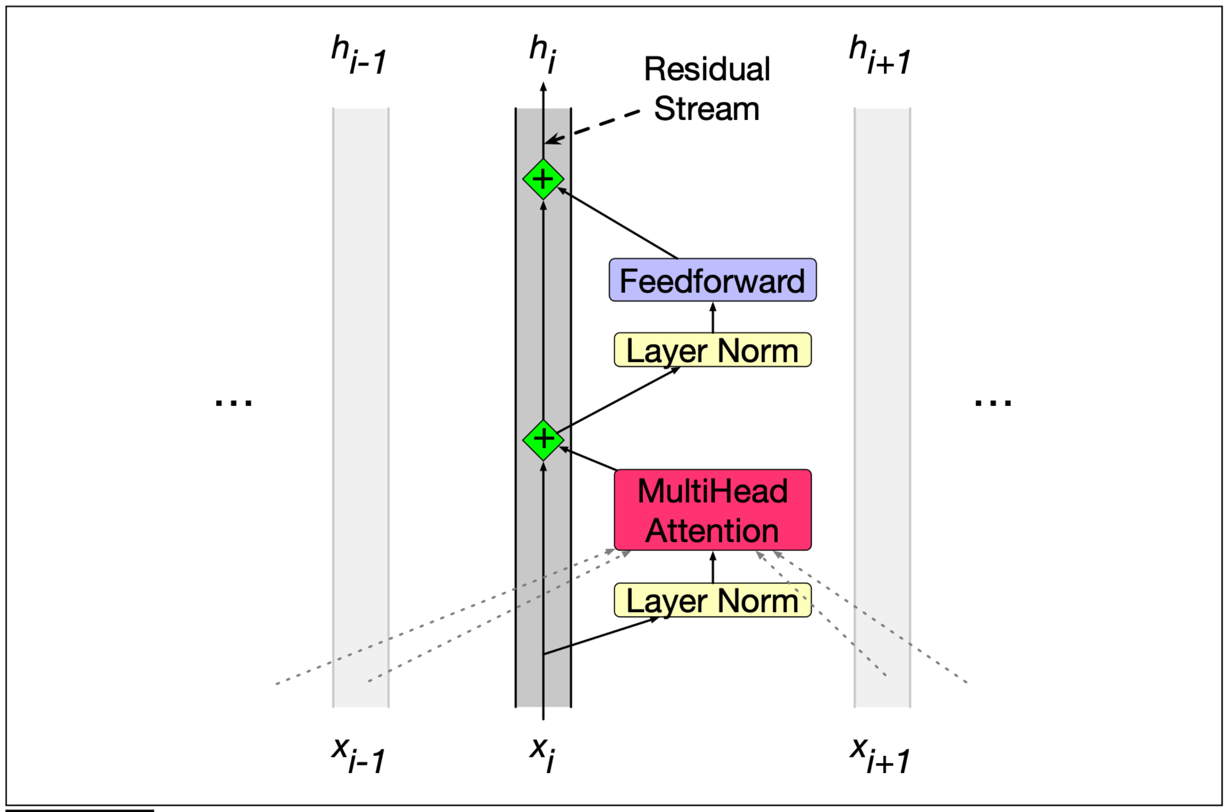


Figure 9.6 The architecture of a transformer block showing the **residual stream**. This figure shows the **prenorm** version of the architecture, in which the layer norms happen before the attention and feedforward layers rather than after.

$$\mathbf{q}_i^c = \mathbf{x}_i\mathbf{W}^{\mathbf{Q}c}; \quad \mathbf{k}_j^c = \mathbf{x}_j\mathbf{W}^{\mathbf{K}c}; \quad \mathbf{v}_j^c = \mathbf{x}_j\mathbf{W}^{\mathbf{V}c}; \quad \forall c \quad 1 \le c \le h \tag{9.14}$$

$$\text{score}^c(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{q}_i^c \cdot \mathbf{k}_j^c}{\sqrt{d_k}} \tag{9.15}$$

$$\alpha_{ij}^c = \text{softmax}(\text{score}^c(\mathbf{x}_i, \mathbf{x}_j)) \;\forall j \le i \tag{9.16}$$

$$\mathbf{head}_i^c = \sum_{j \le i} \alpha_{ij}^c \mathbf{v}_j^c \tag{9.17}$$

$$\mathbf{a}_i = (\mathbf{head}^1 \oplus \mathbf{head}^2 \ldots \oplus \mathbf{head}^h)\mathbf{W}^O \tag{9.18}$$

$$\text{MultiHeadAttention}(\mathbf{x}_i, [\mathbf{x}_1, \cdots, \mathbf{x}_N]) = \mathbf{a}_i \tag{9.19}$$

Putting it all together

Thursday, November 7, 2024 8:54 AM

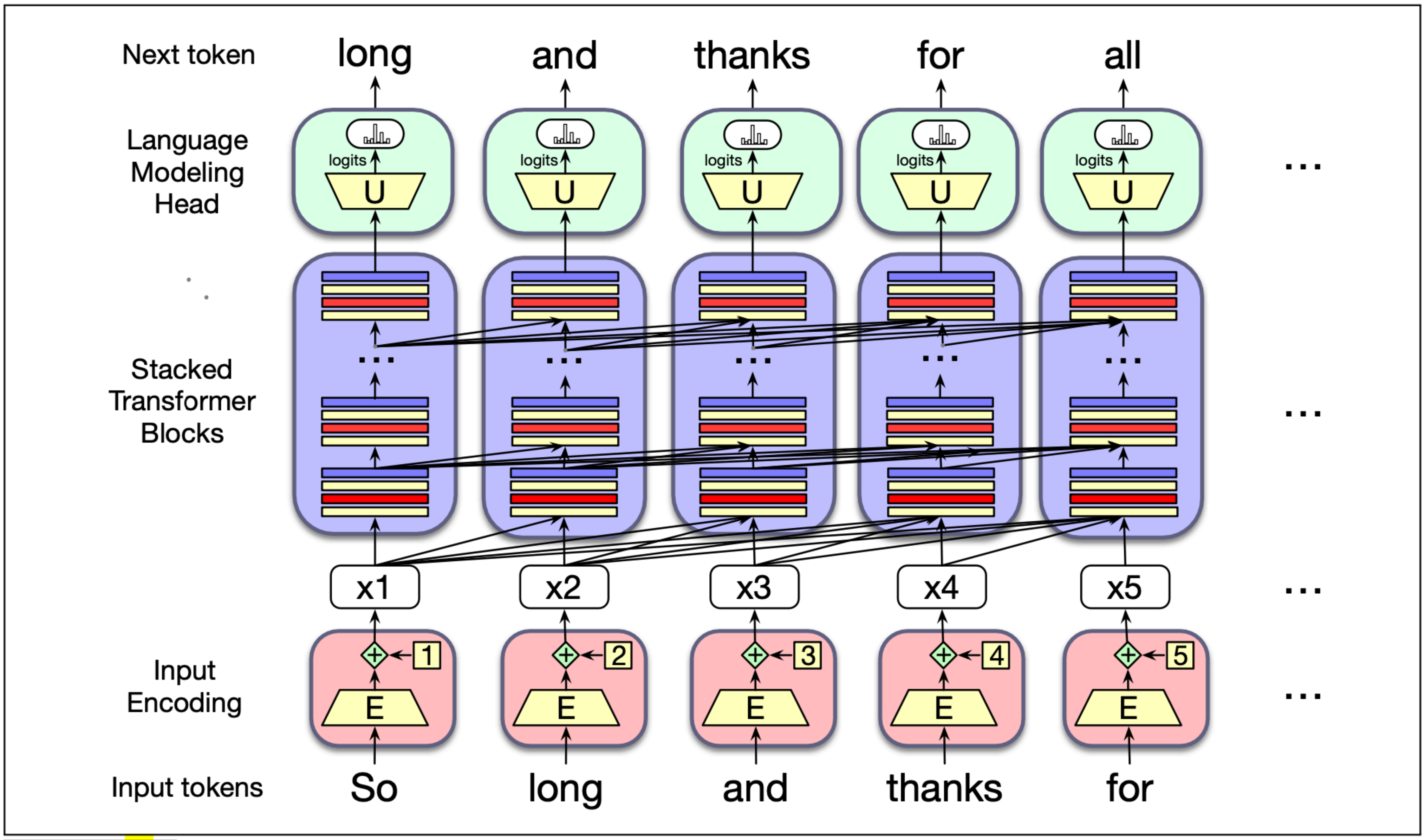


Figure 9.1 The architecture of a (left-to-right) transformer, showing how each input token get encoded, passed through a set of stacked transformer blocks, and then a language model head that predicts the next token.