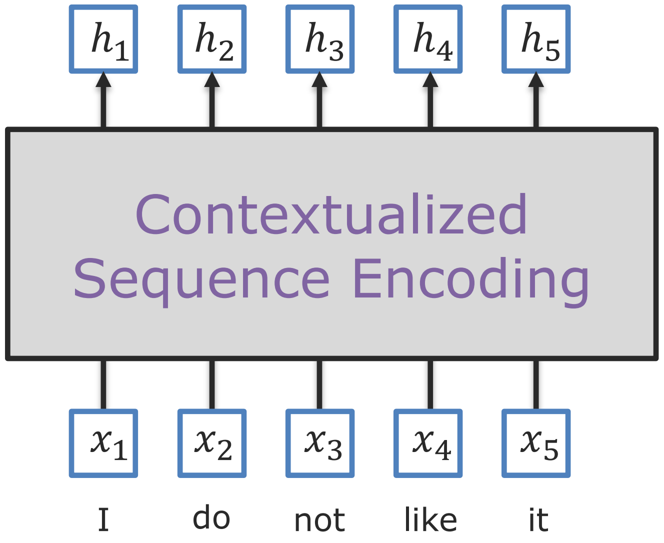
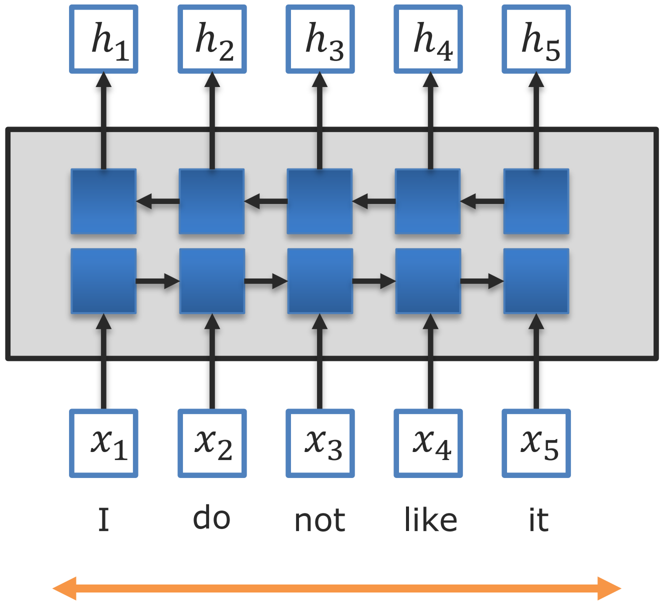
Contextualized Sequence Representations

Sequence Encoding - Contextualization



How to encode this sequence while modeling the interaction between elements (e.g., words)?

Option 1: Bi-directional LSTM:
(e.g., ELMO)

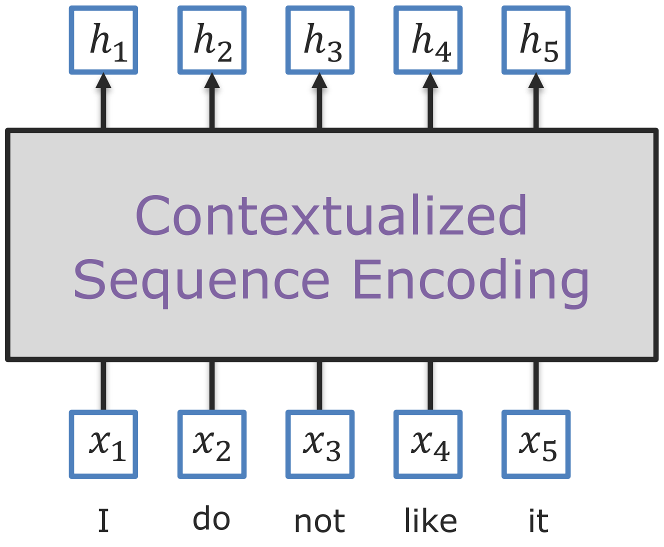


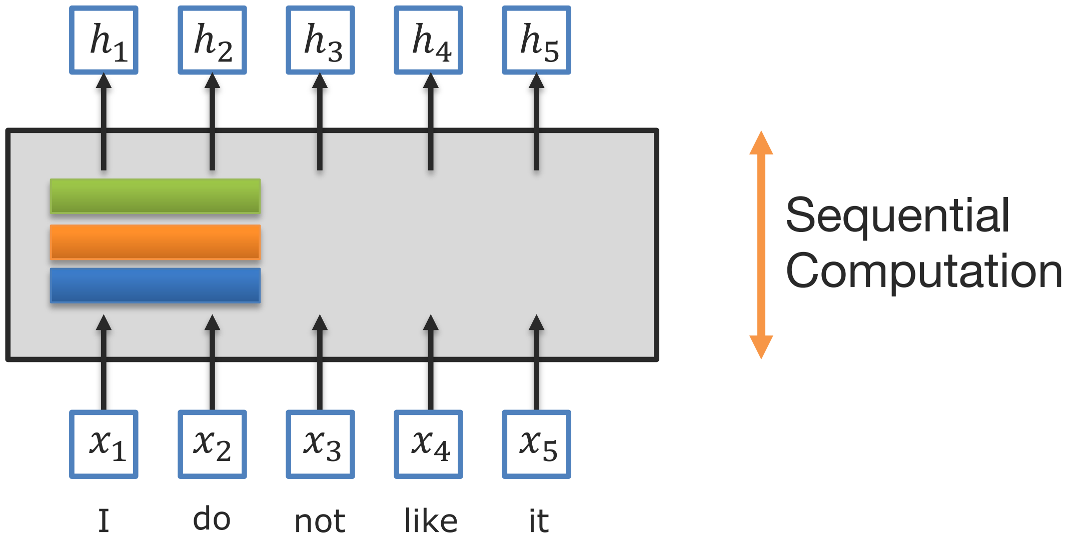
Sequential Computation

But harder to parallelize…

Sequence Encoding - Contextualization

Option 2: Convolutions



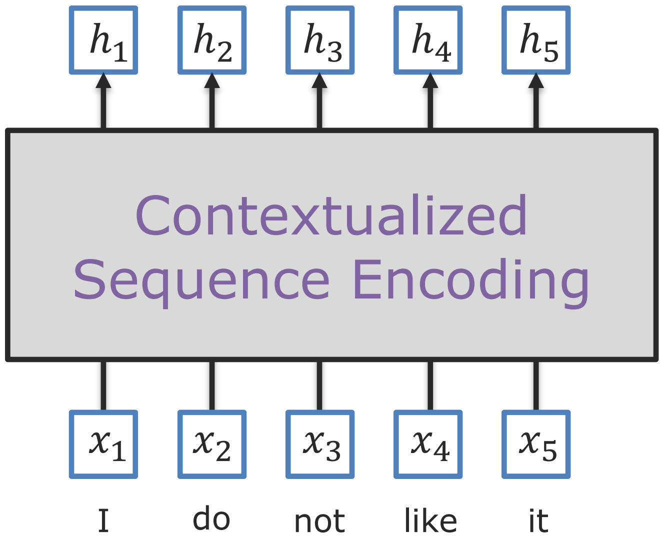


Can be parallelized!

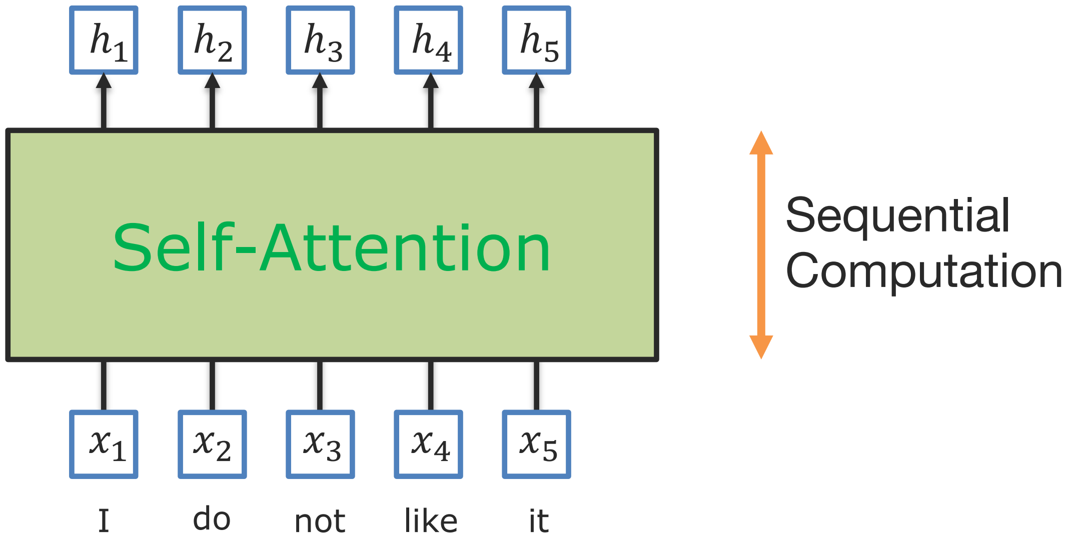
But modeling long-range dependencies requires multiple layers.

And convolutional kernels are static.

Sequence Encoding - Contextualization



Option 3: Self-attention



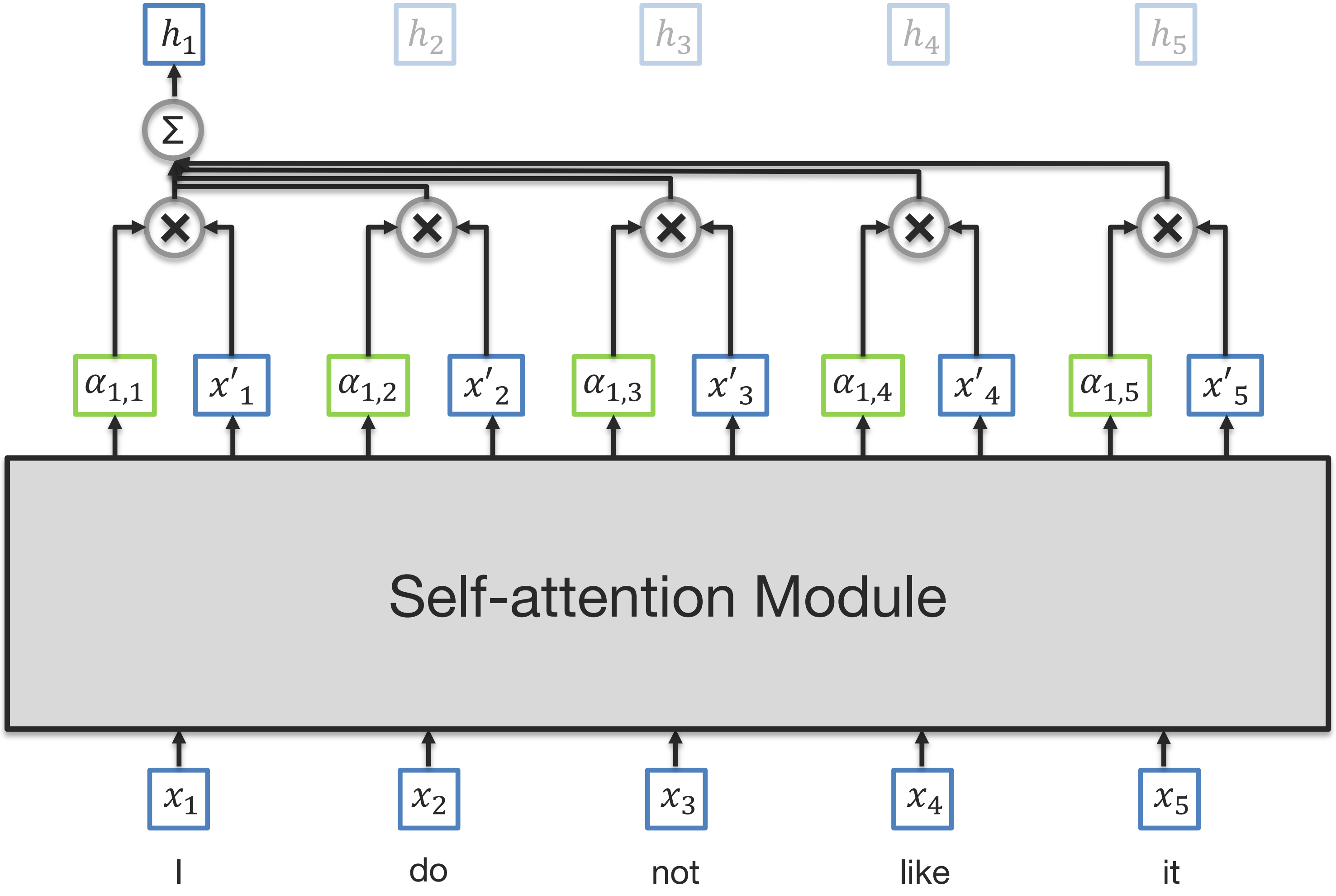
Can be parallelized!

Long-range dependencies

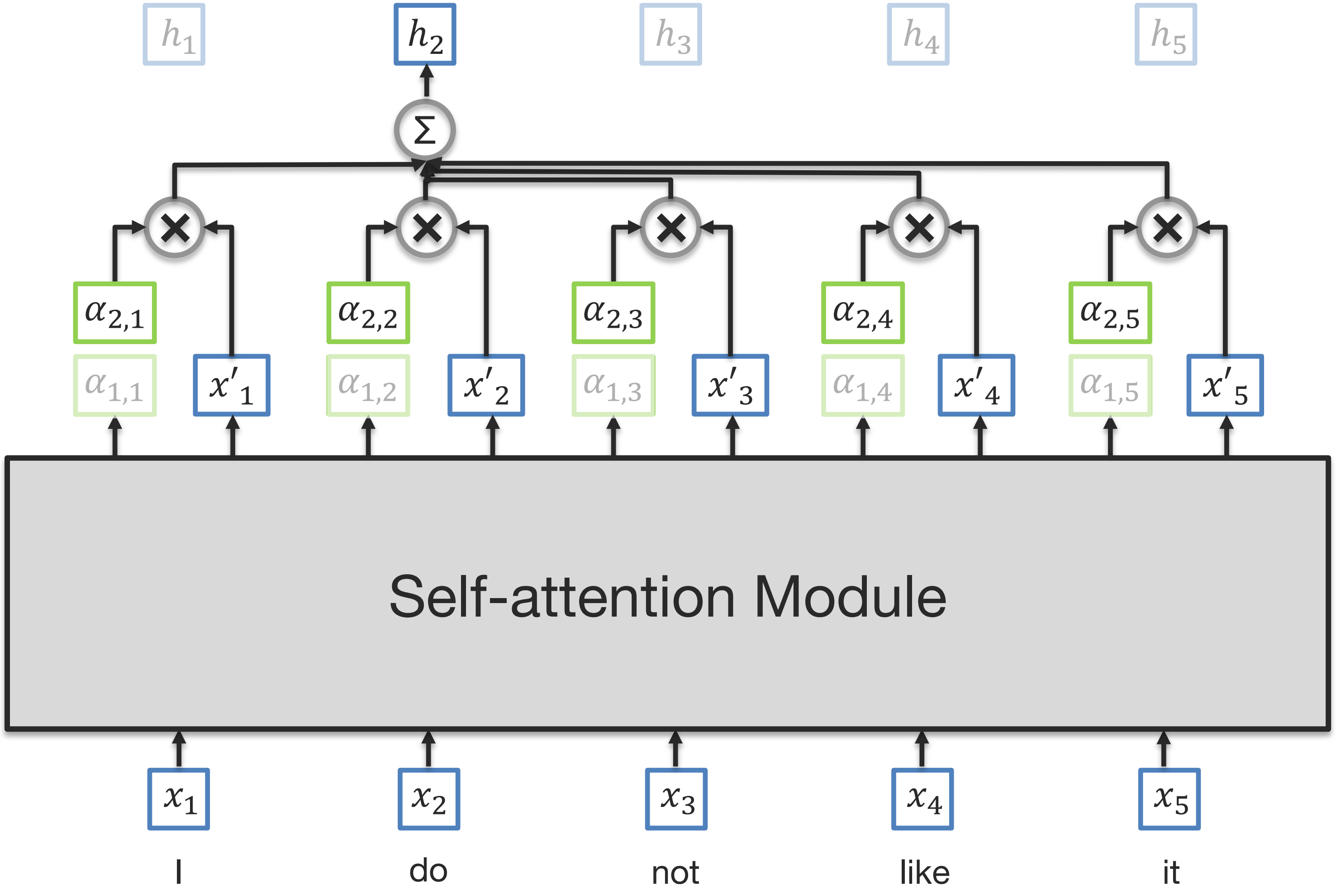
Dynamic attention weights

Self-Attention

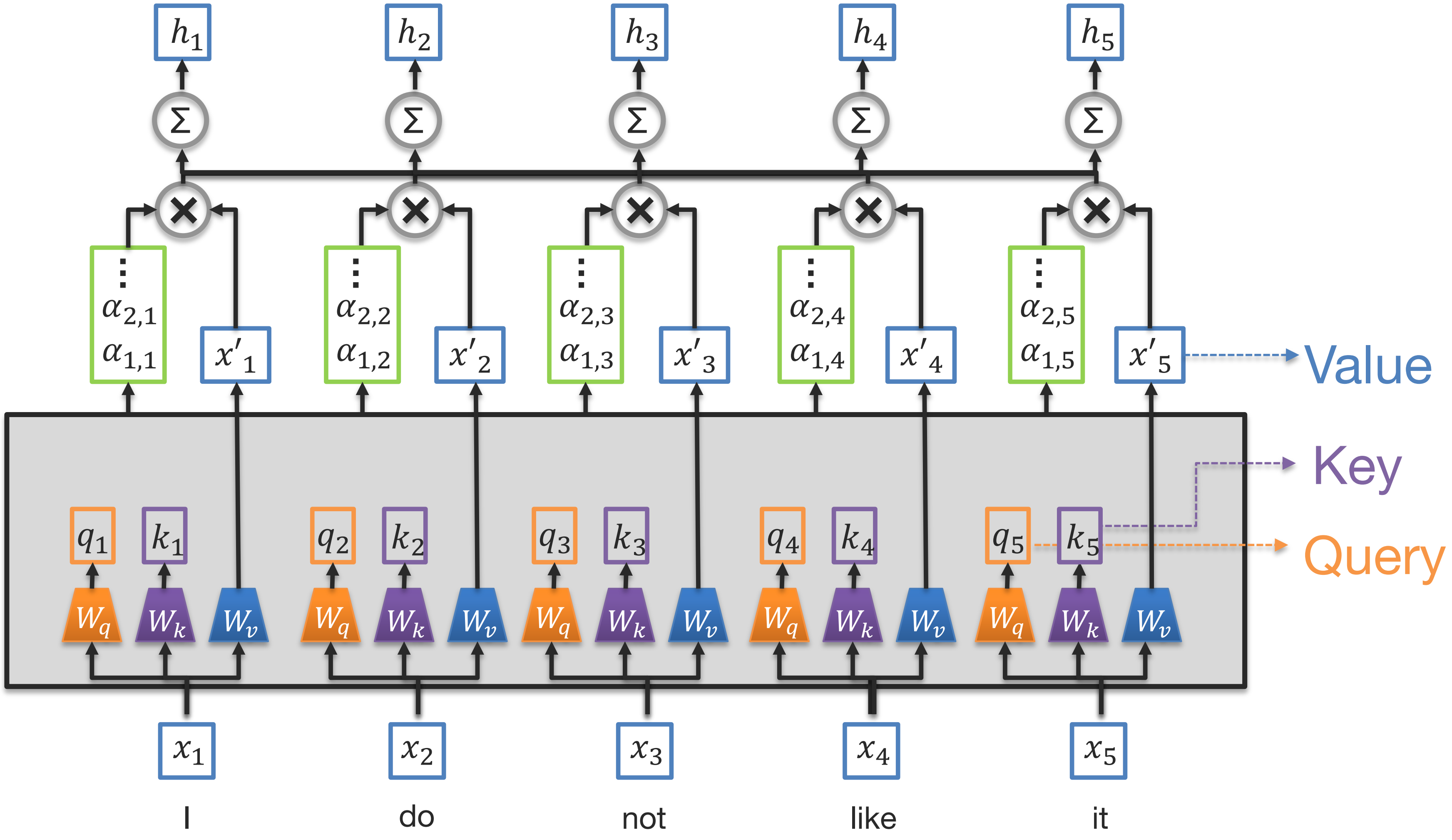
Self-Attention



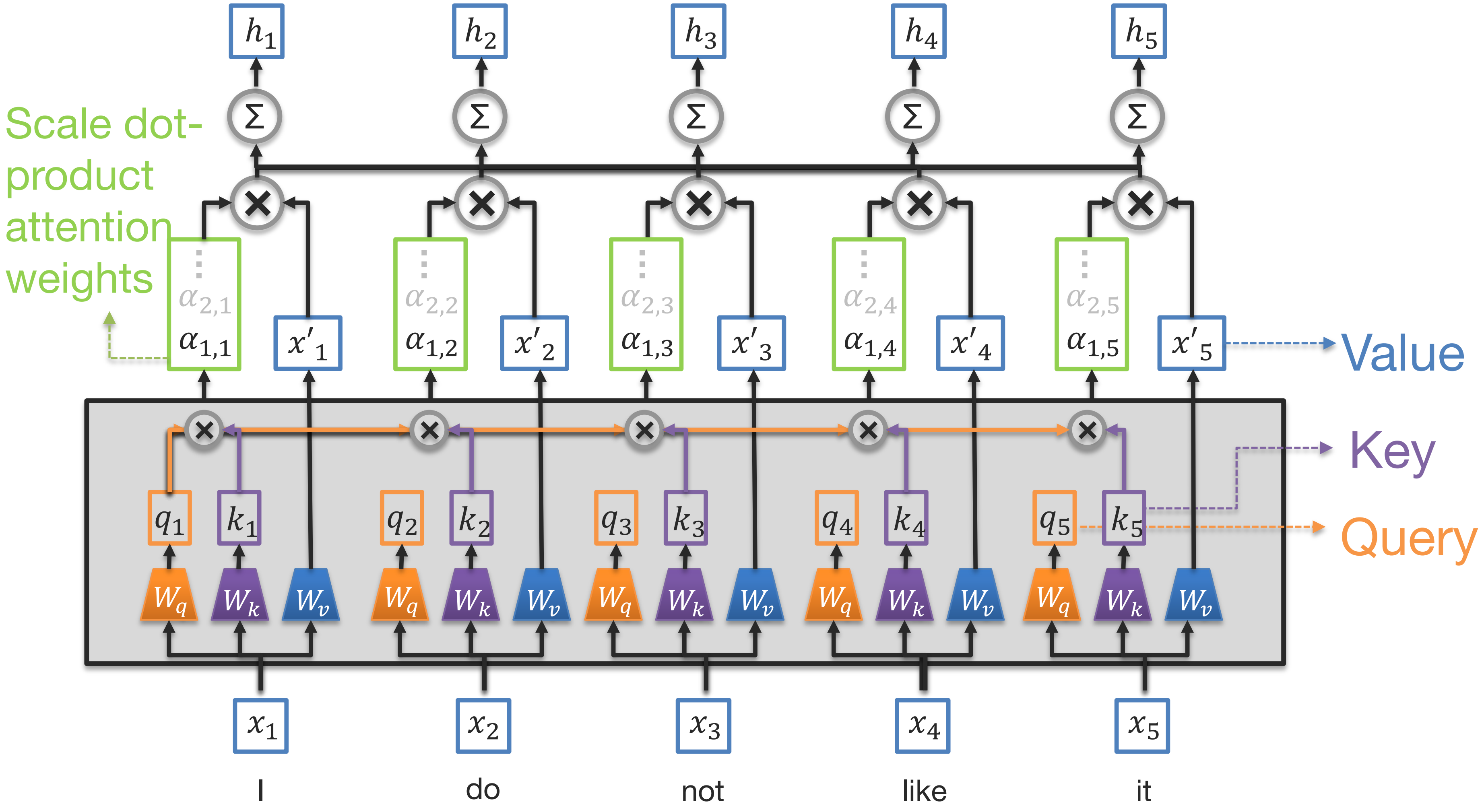
Self-Attention



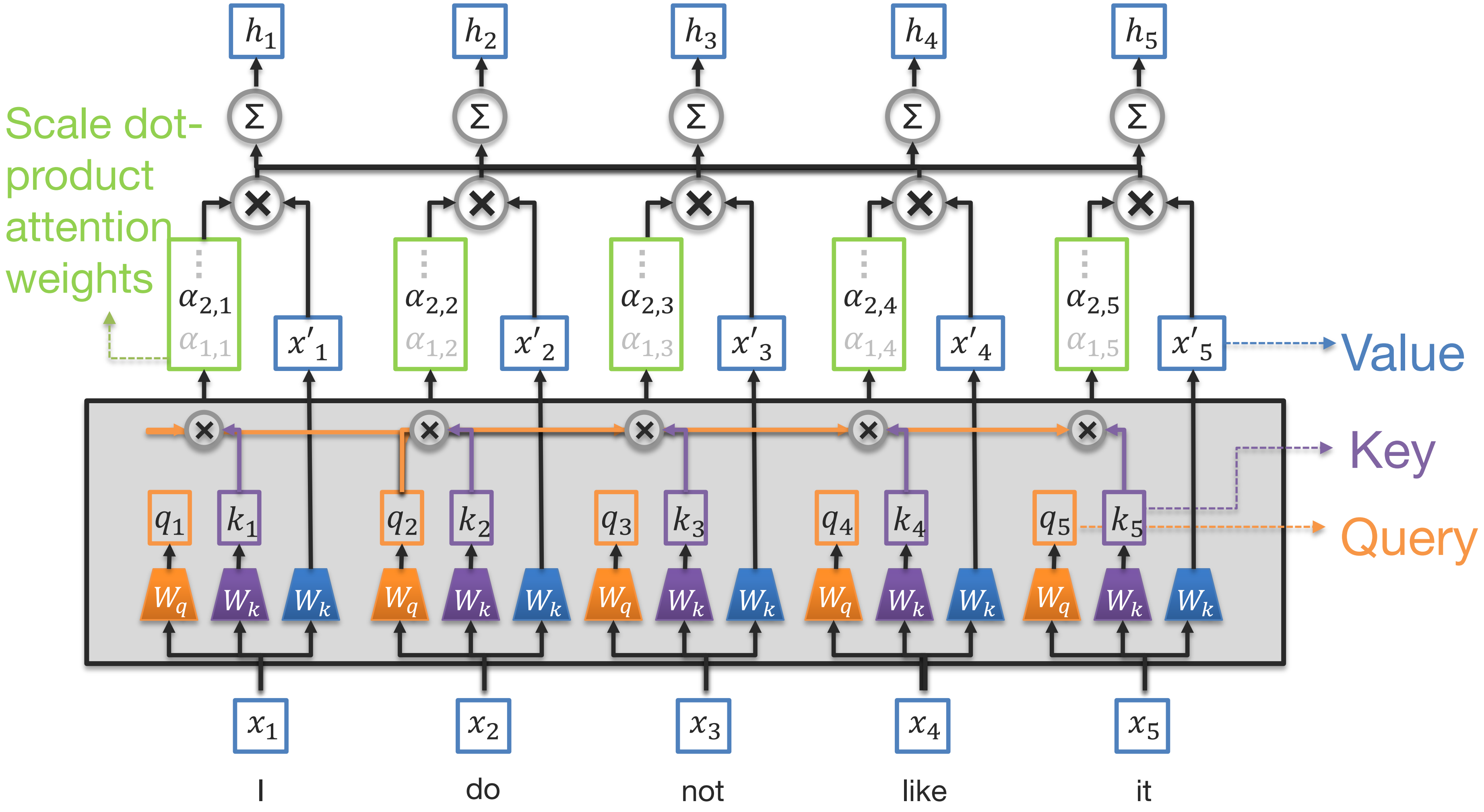
Transformer Self-Attention



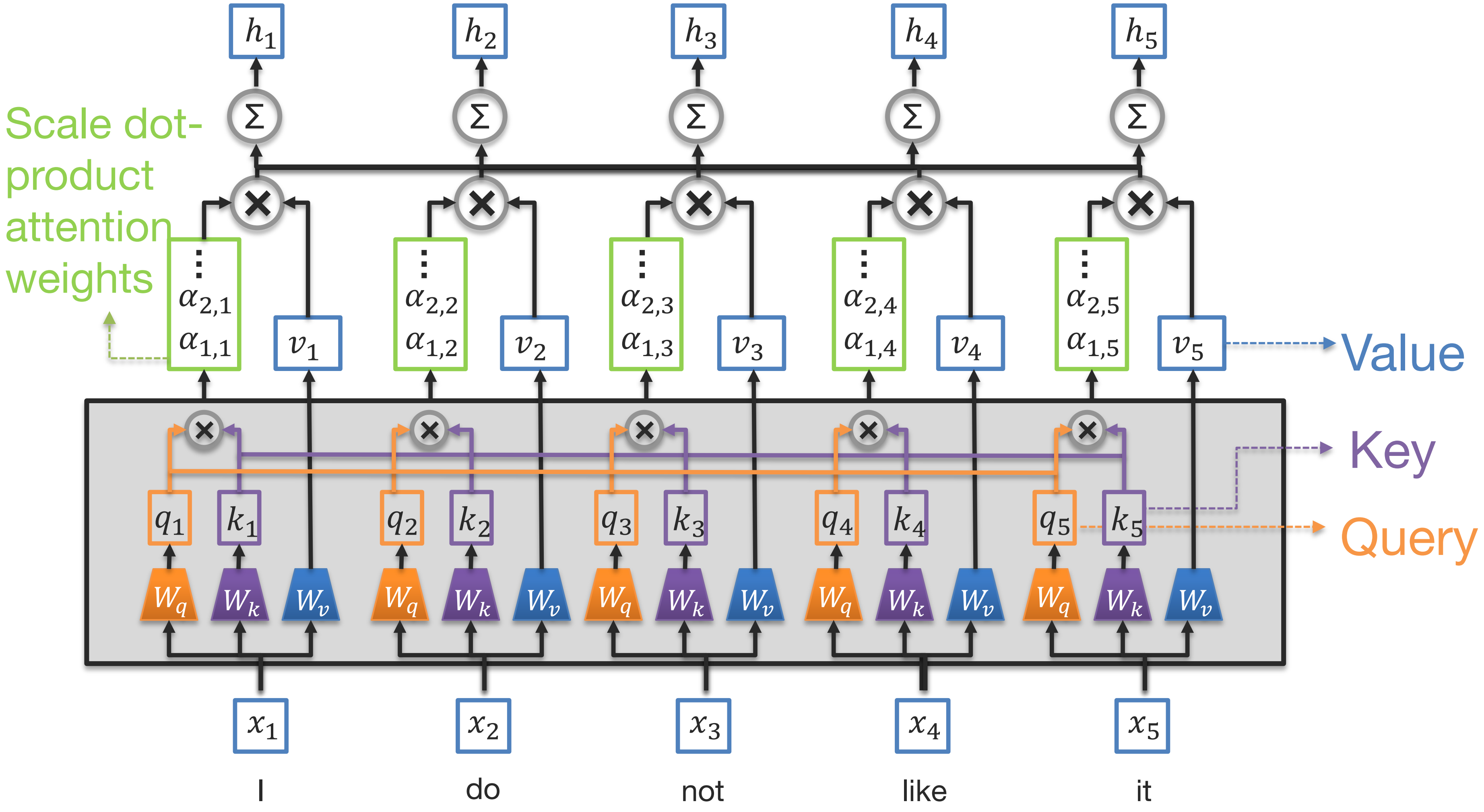
Transformer Self-Attention



Transformer Self-Attention

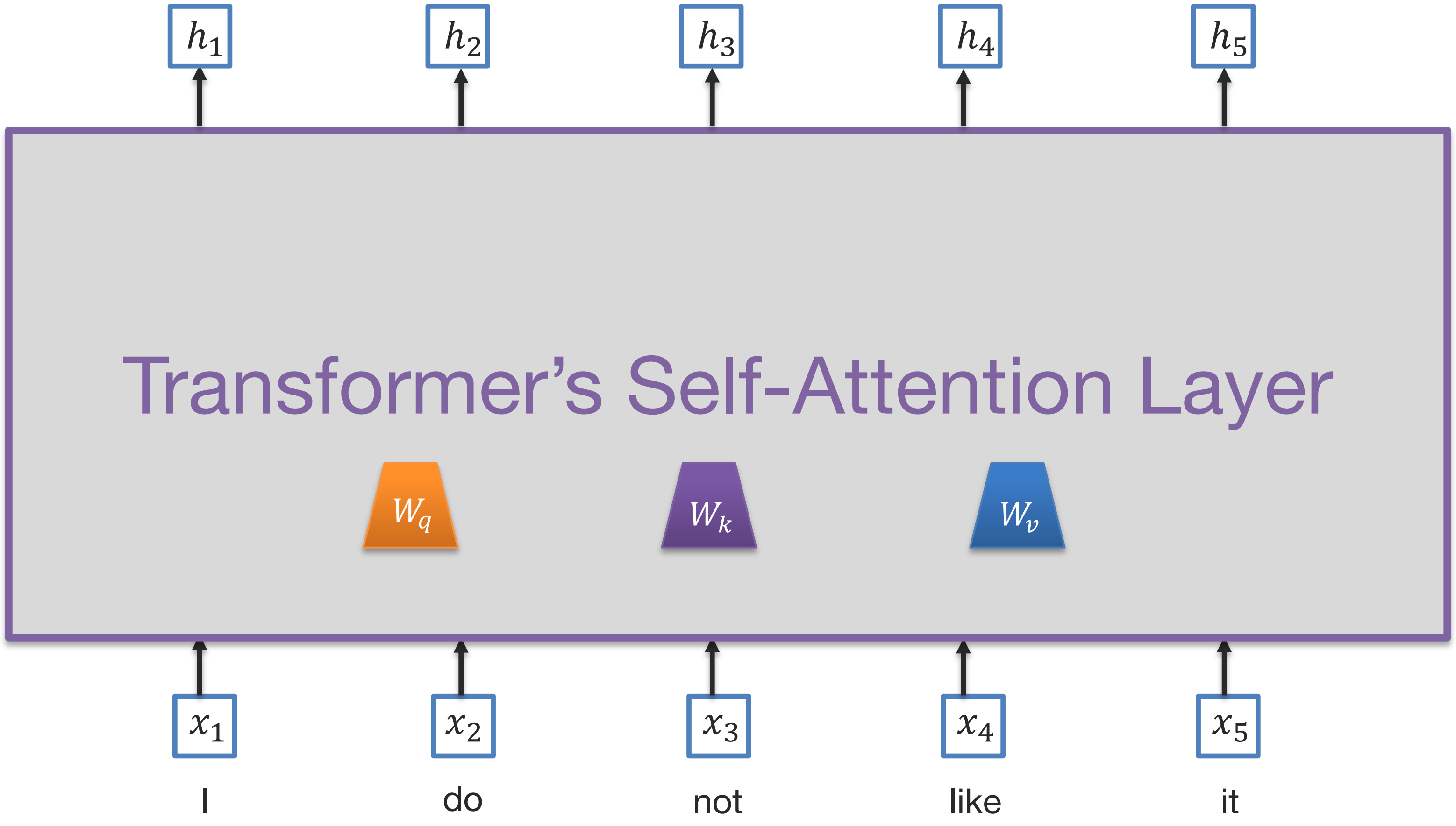


Transformer Self-Attention

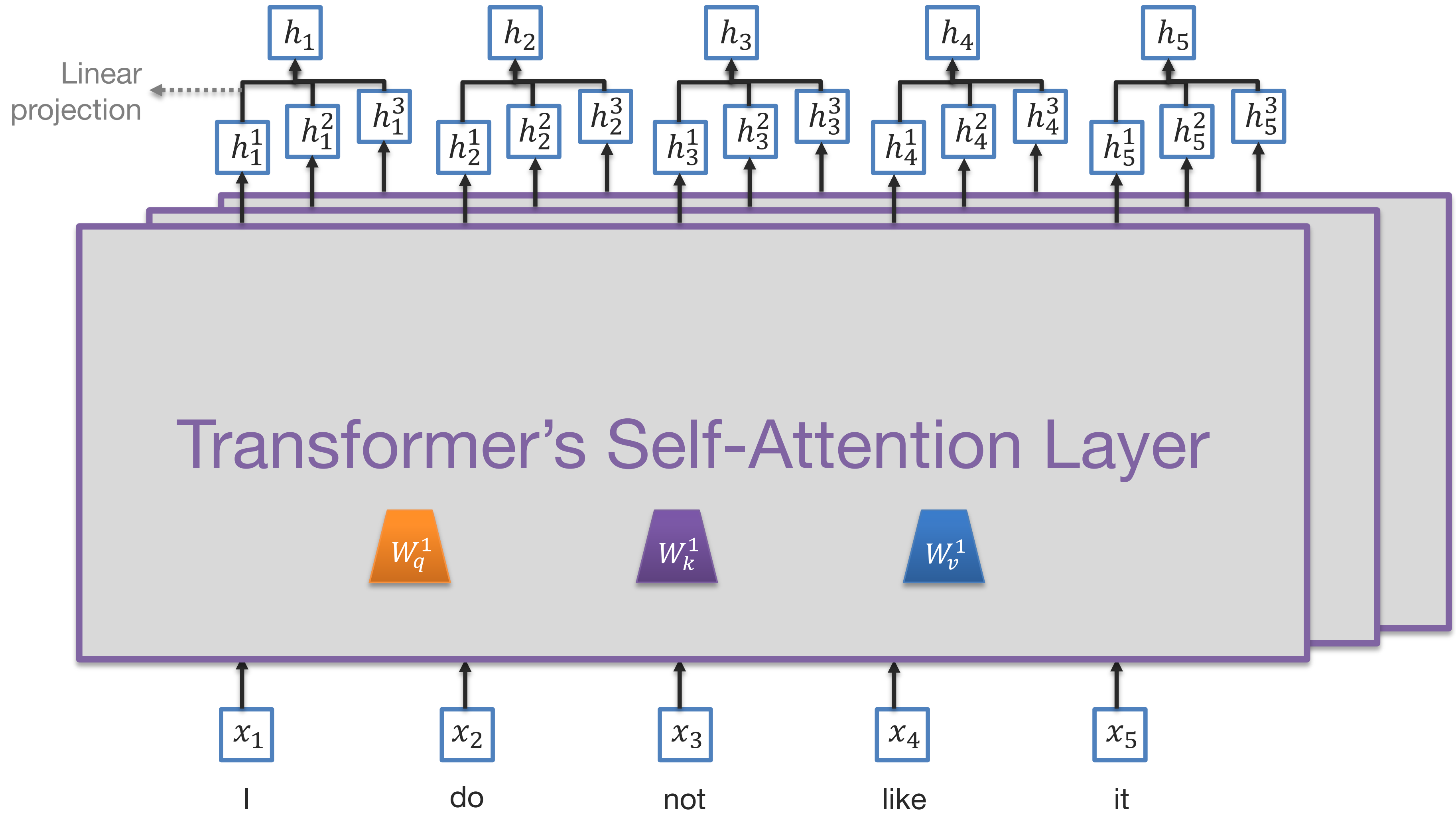


Transformer Self-Attention

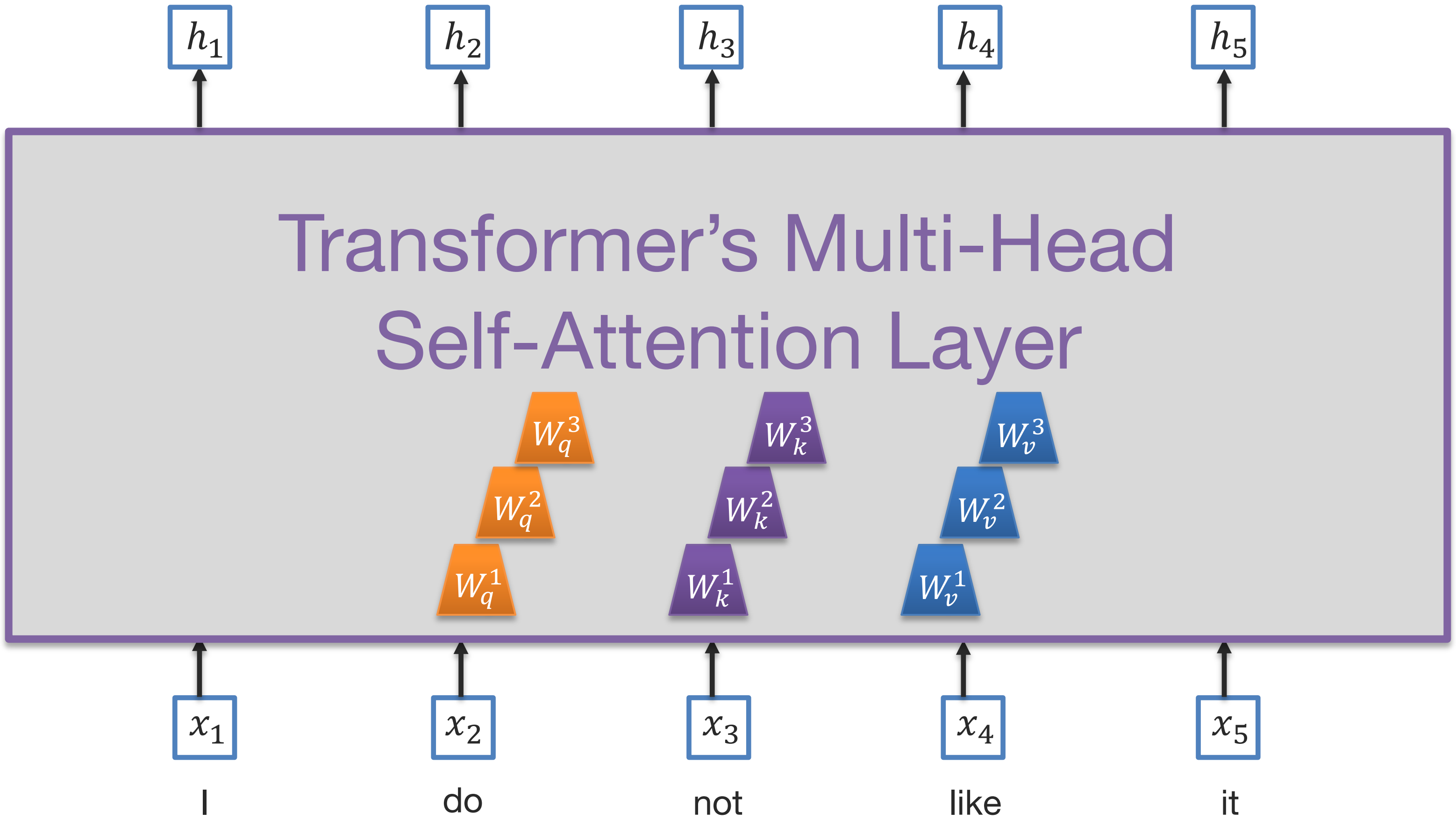
What if we want to attend simultaneously to multiple subspaces of x?



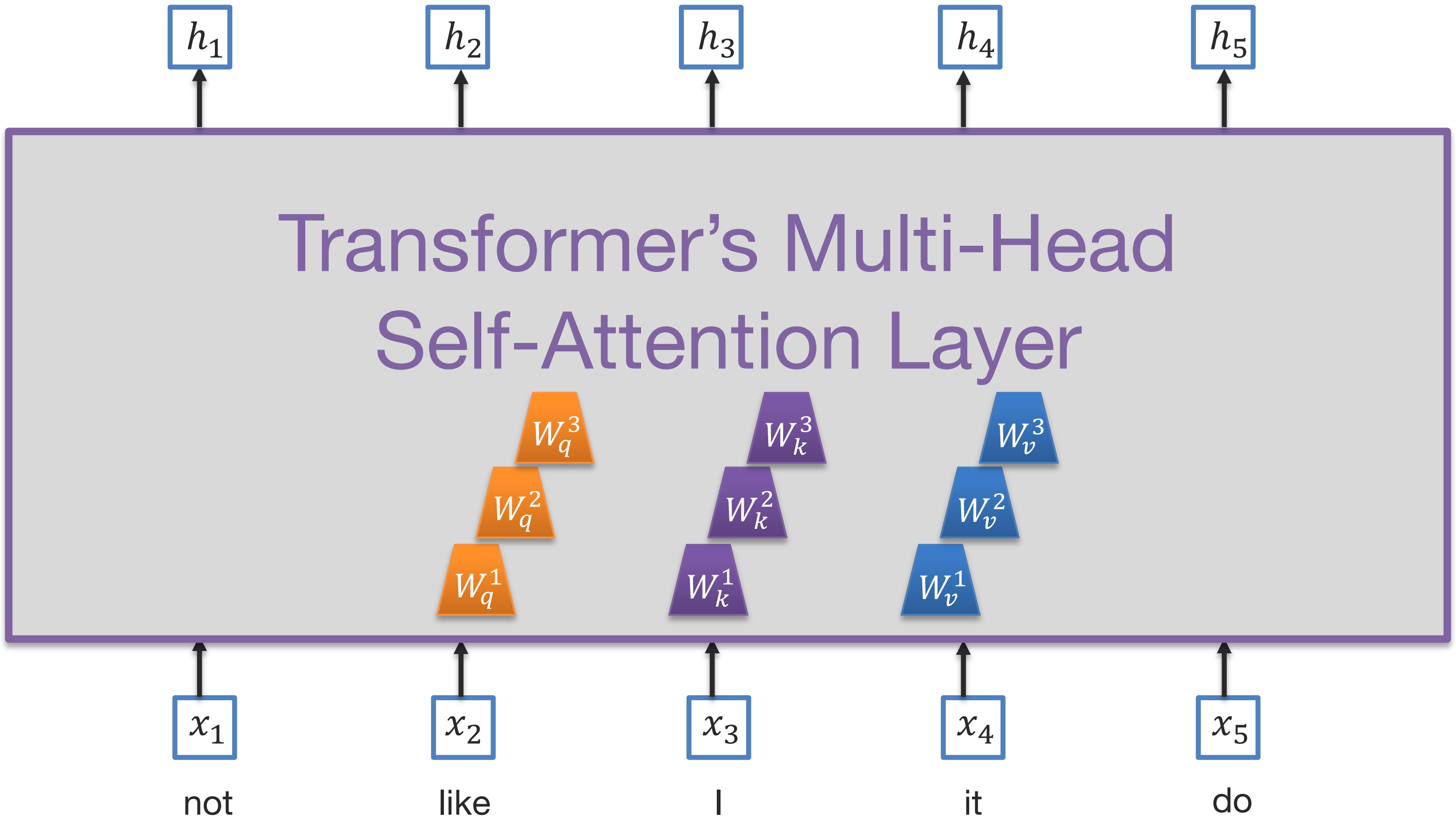
Transformer Multi-Head Self-Attention



Transformer Multi-Head Self-Attention



Transformer Multi-Head Self-Attention



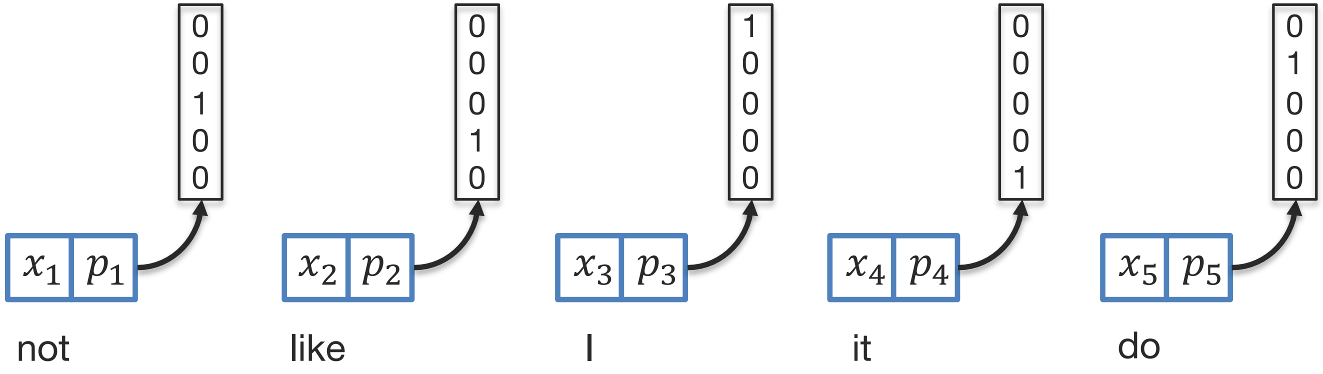
What happens if the words are shuffled?

Position embeddings

- ❑ Position information is not encoded in a self-attention module

How can we encode position information?

Simple approach: one-hot encoding

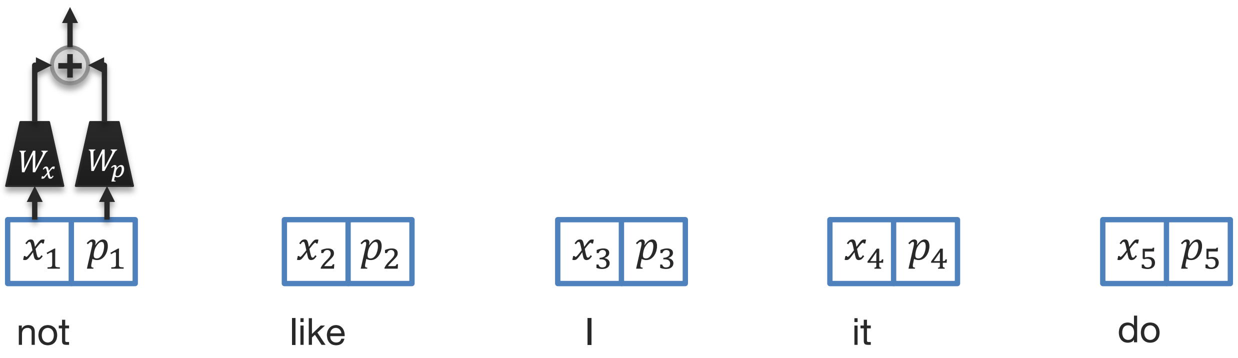


Position embeddings

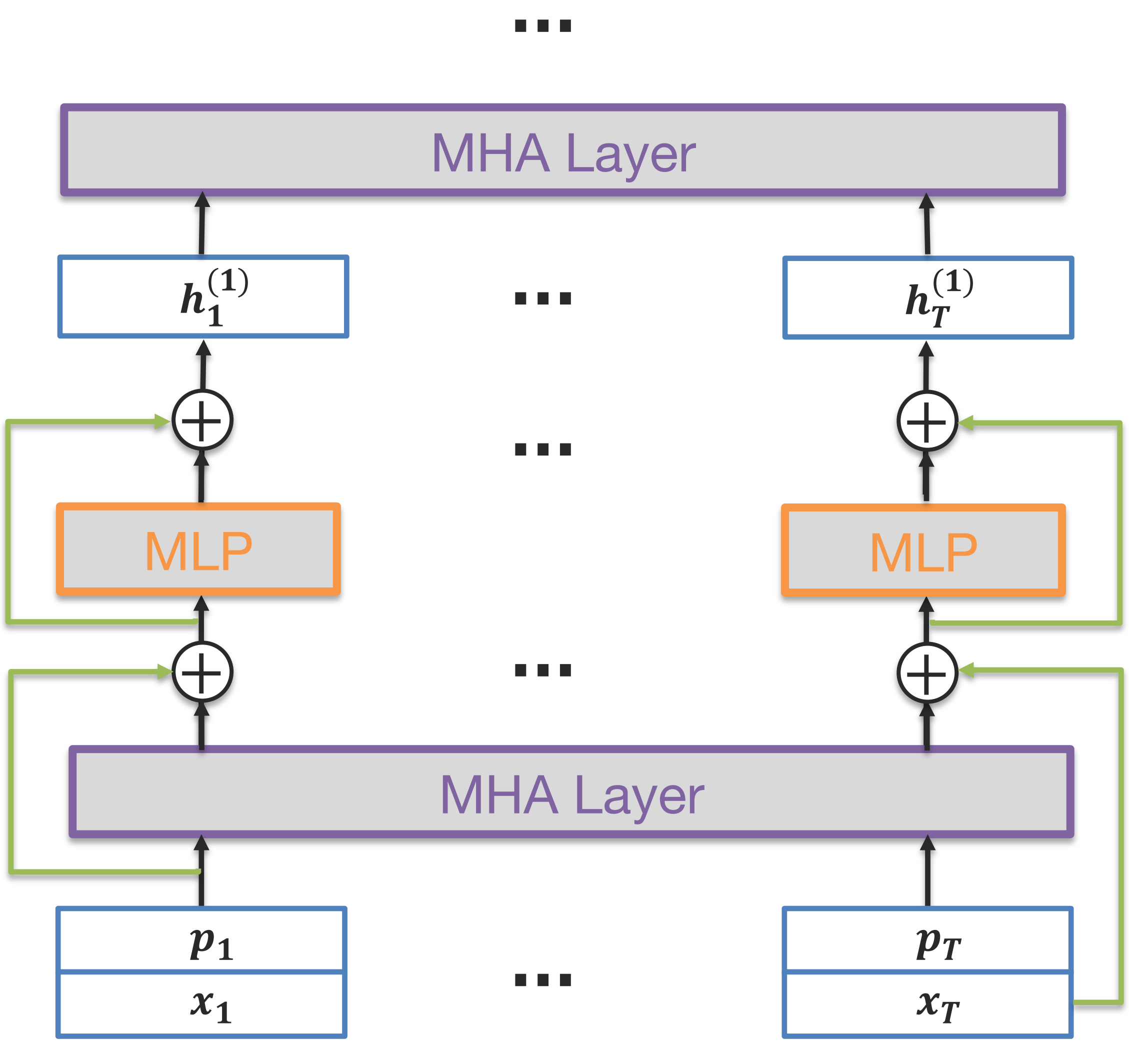
❑ Position information is not encoded in a self-attention module

How can we encode position information?

Simple approach: one-hot encoding + linear embeddings + { Sum - or - concat



Transformer – Layers



Multi-head attention is basically linear
=> need **MLPs**, and many layers (12 in BERT, 96 in GPT-3)

Residual connections allow each layer to update only a small subspace of the hidden space
=> more than just contextualized word vectors

Transformer Key Ideas

1. Learn to route information, then apply local computation to compose it.
2. No fixed-weight similarity kernels; attention is input-dependent similarity
3. Compose multiple simple learned operations
4. Highly parallelizable; allows data to trump inductive biases

For more intuition: https://transformer-circuits.pub/2021/framework/index.html