Self-attention
Probability score matrix

(last week)

	Hello	I	love	you	
Hello	0.8	0.1	0.05	0.05	Softmax()
I	0.1	0.6	0.2	0.1	...
love	0.05	0.2	0.65	0.1	
you	0.2	0.1	0.1	0.6	Softmax()

(source) ENGLISH → (target) KOREAN

<CROSS Attention>

The cat sat

1 2 3 chunk

고양이가 앉았다

1 2 chunk

	The	cat	sat
고양이가	0.2	0.75	0.05
앉았다	0.05	0.10	0.85

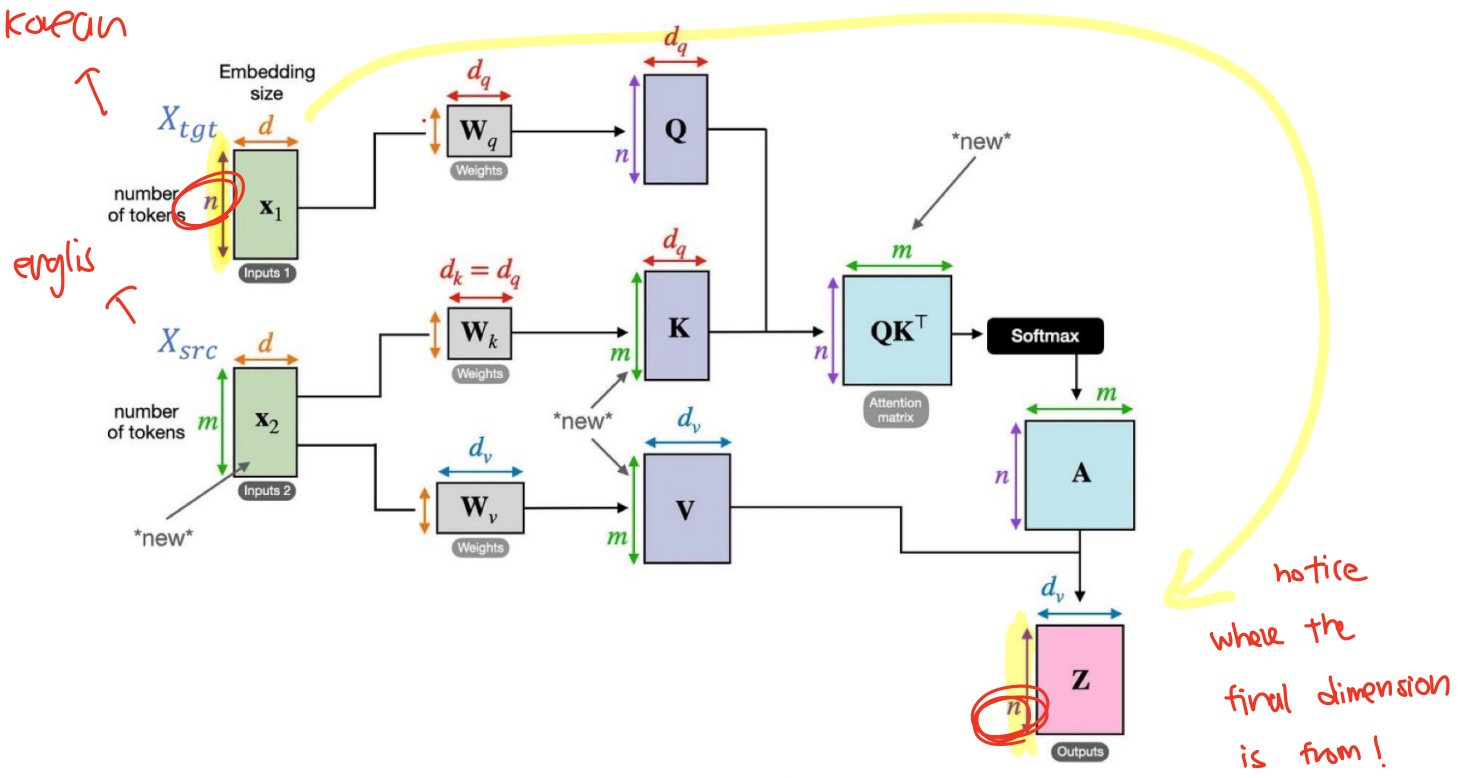
The cat sat → 고양이가

Quiz : Q, K, V

↳ comes from (a) source OR (b) target ?

korean ↑

english ↑



notice where the final dimension is from!

Figure 1: Cross-attention block.

<Some Details : FFN>

attn block from last week

attn_out = (concat(H1 ... Hn) W^(O))

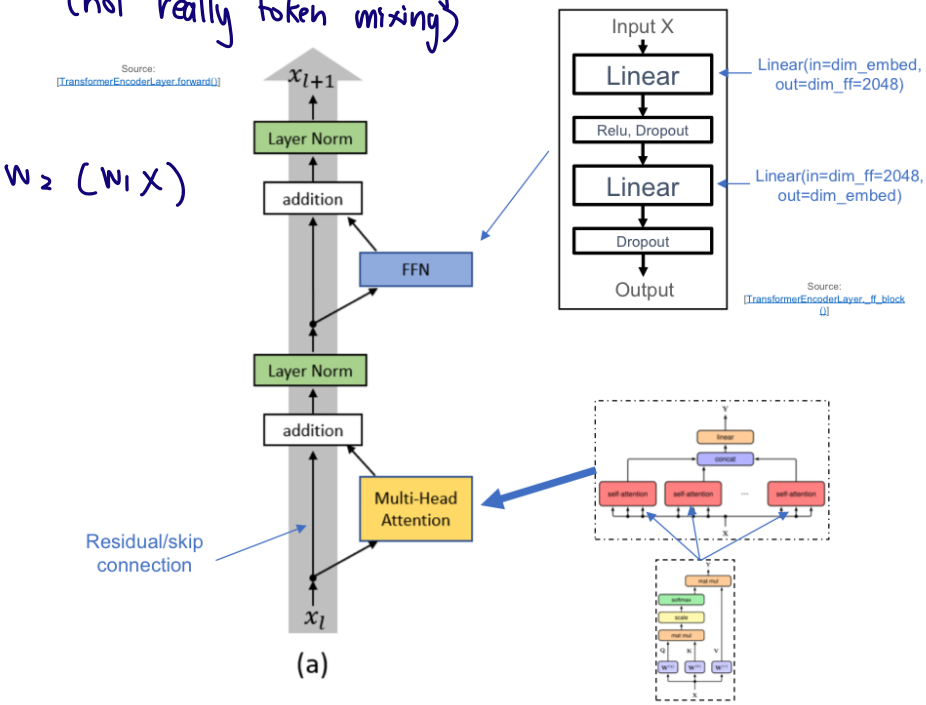
↳ x + attn_out

↳ FFN

FFN after the MHA → why?

further feature mixing
(not really token mixing)

$W_2 (W_1 X)$



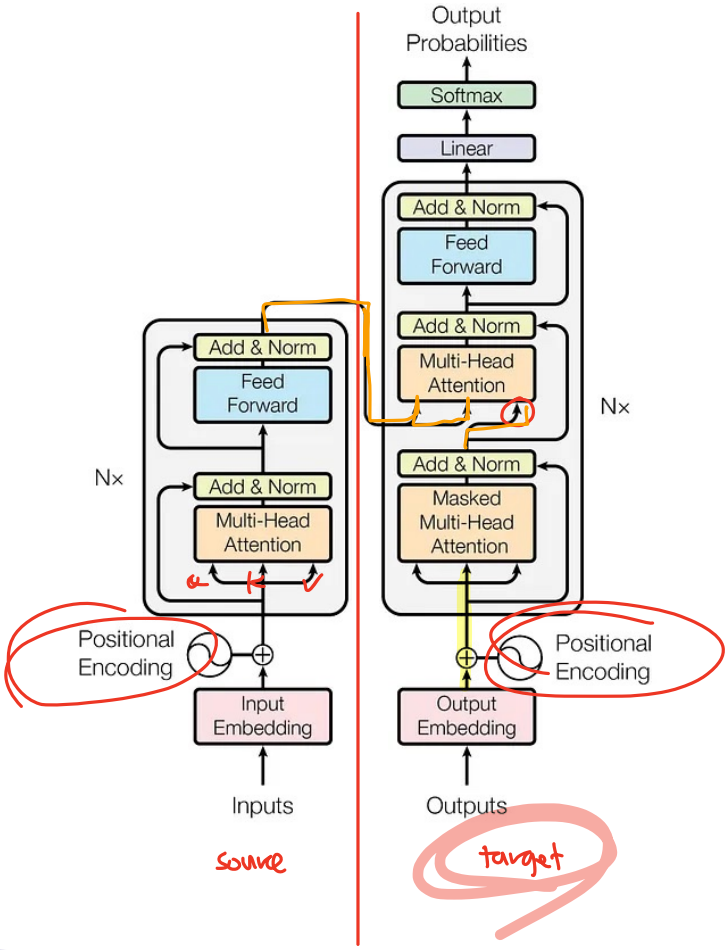
Important: FFN is broadcasted to each token, independently.
In other words, the same Linear layer(s) are applied to each token independently (not a different Linear layer for each token position, nor a giant Linear layer that mixes all token embeddings, that would be too expensive!)

<Transformer as a whole>

BERT

embeddin

Encoder



GPT

Decoder

<Casual Mask>

A B C D

A → predict B

AB → predict C

ABC → predict D

⋮

A → only A

B → can see A, B

A B C
↓ ↓ ↓
B C D

C → can see A, B, C

⋮

Self-attention
Probability score matrix

	Hello	I	love	you	
Hello	0.8	0.1	0.05	0.05	Softmax()
I	0.1	0.6	0.2	0.1	...
love	0.05	0.2	0.65	0.1	
you	0.2	0.1	0.1	0.6	Softmax()

in practice

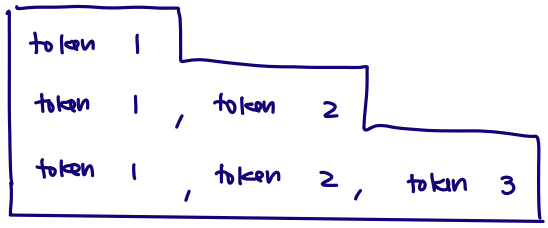
you add a

-inf	-inf	-inf
	-inf	-inf
		-inf

since

softmax(-inf) = 0

predict next tokens →



→ token 2
token 3
token 4

<Inference>

t1 t2 t3

	t1	t2	t3
t1	✓	X	X
t2	✓	✓	X
t3	✓	✓	✓

Data 188 Introduction to Deep Learning
Spring 2026 Eric Kim

Discussion 09

This week we continue our study of transformers by focusing on the encoder-decoder setting.

1. Cross-Attention Shapes

Recall that in multi-head cross-attention, the queries come from the target sequence, while the keys and values come from the source sequence. For head $h \in \{1, \dots, H\}$,

$$Q_h = X_{\text{tgt}} W_h^{(q)}, \qquad K_h = X_{\text{src}} W_h^{(k)}, \qquad V_h = X_{\text{src}} W_h^{(v)},$$

$$\mathrm{H}_h = \mathrm{Softmax}\left(\frac{Q_h K_h^\top}{\sqrt{d_q}}\right) V_h,$$

and the full multi-head output is

$$Y = \mathrm{Concat}(\mathrm{H}_1, \dots, \mathrm{H}_H)\, W^{(o)}.$$

In the figure below, we will focus on a *single* cross-attention block (that is, one head). As a result, each target token computes attention weights over the source tokens, and these weights depend on the particular input sequences in the batch.

Assume $X_{\text{tgt}} \in \mathbb{R}^{B \times n_{\text{tgt}} \times d}$, $X_{\text{src}} \in \mathbb{R}^{B \times n_{\text{src}} \times d}$, and assume the query/key dimension is d_q (so $d_k = d_q$) and the value dimension is d_v.

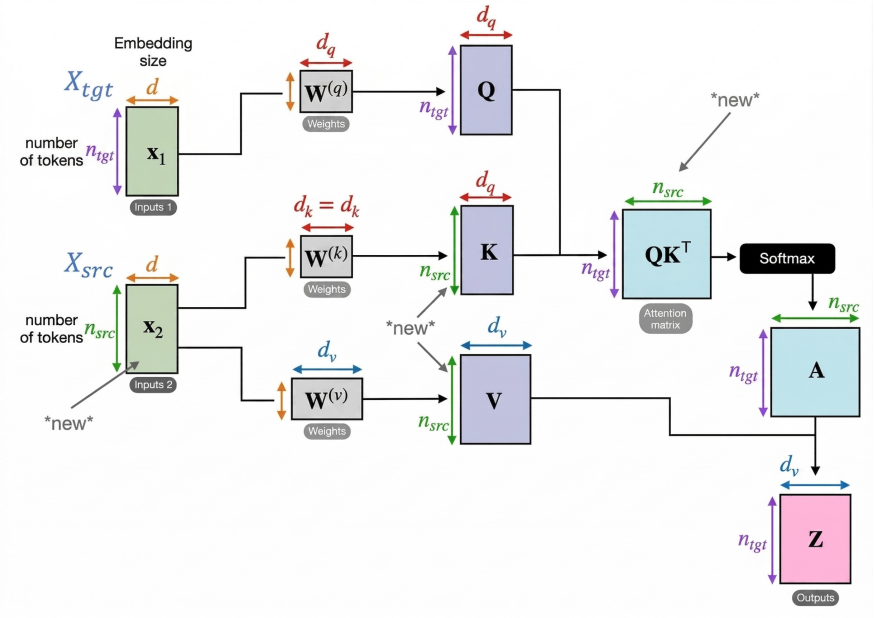


Figure 1: Cross-attention block.

You may annotate the figure as you work, but please write your final answers below.

(a) What are the shapes of the projection weight matrices $W^{(q)}$, $W^{(k)}$, and $W^{(v)}$?

Solution: Since each token starts in the model embedding space of dimension d, the projection matrices map from d to the corresponding query/key/value dimensions:

$$W^{(q)} \in \mathbb{R}^{d \times d_q}, \qquad W^{(k)} \in \mathbb{R}^{d \times d_q}, \qquad W^{(v)} \in \mathbb{R}^{d \times d_v}.$$

(b) What are the shapes of the projected activations Q, K, and V?

Solution: The target sequence produces the queries, and the source sequence produces the keys and values:

$$Q = X_{\text{tgt}} W^{(q)} \in \mathbb{R}^{B \times n_{\text{tgt}} \times d_q},$$

$$K = X_{\text{src}} W^{(k)} \in \mathbb{R}^{B \times n_{\text{src}} \times d_q}, \qquad V = X_{\text{src}} W^{(v)} \in \mathbb{R}^{B \times n_{\text{src}} \times d_v}.$$

(c) What are the shapes of the score matrix $QK^\top$, the attention matrix

$$A = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_q}}\right),$$

and the final output

$$Z = AV?$$

Then, in one or two sentences, explain what the entries of A represent. Why can A be viewed as a set of *data-dependent* weights?

Solution: For each batch element, each target token compares itself against every source token, so

$$QK^\top \in \mathbb{R}^{B \times n_{\text{tgt}} \times n_{\text{src}}}.$$

The scaling and softmax do not change shape, so

$$A \in \mathbb{R}^{B \times n_{\text{tgt}} \times n_{\text{src}}}.$$

Finally, multiplying by V collapses the source-token dimension and leaves one output vector per target token:

$$Z = AV \in \mathbb{R}^{B \times n_{\text{tgt}} \times d_v}.$$

Each row of A gives a distribution of attention weights over the n_{src} source positions for one target position. In particular, A_{ij} tells us how much target token i attends to source token j.

These weights are *data-dependent* because they are not fixed after training like the weights of a standard linear layer. Instead, they are computed from the current input sequences through $QK^\top$, so changing X_{tgt} or X_{src} changes A.

2. Decoder Block Shapes in an Encoder-Decoder Transformer

The figure below zooms in on one decoder block inside an encoder-decoder transformer for machine translation.

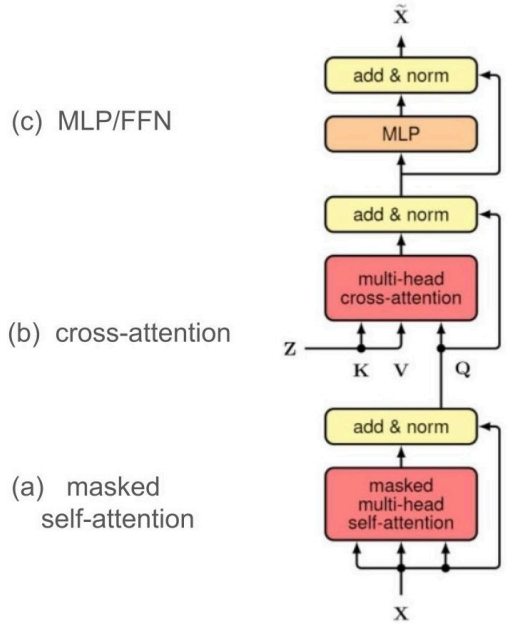


Figure 2: One decoder block in an encoder-decoder transformer.

You may annotate the figure as you work, but please write your final answers below.

Assume LayerNorm and residual connections do not change tensor shapes.

For both attention sublayers, assume query/key dimension d_q (so $d_k = d_q$), value dimension d_v, and an output projection back to model dimension d.

(a) Recall the single head attention mechanism (these are the same equations you've seen before, but we've just added the subscript "self" to distinguish from the cross-attention sublayer):

$$
\begin{aligned}
Q_{\text{self}} &= XW_{\text{self}}^{(q)} \\
K_{\text{self}} &= XW_{\text{self}}^{(k)} \\
V_{\text{self}} &= XW_{\text{self}}^{(v)} \\
A_{\text{self}} &= \text{Softmax}\left(\frac{Q_{\text{self}}K_{\text{self}}^{\top}}{\sqrt{d_k}}\right) \quad \text{(Attention matrix)} \\
H_{\text{self}} &= A_{\text{self}}V_{\text{self}} \quad \text{(Attention head)} \\
Y_{\text{self}} &= H_{\text{self}}W_{\text{self}}^{(o)} \quad \text{(Attention block output, single head)}
\end{aligned}
$$

In the masked self-attention sublayer, what are the shapes of

$$W_{\text{self}}^{(q)},\ W_{\text{self}}^{(k)},\ W_{\text{self}}^{(v)},\ W_{\text{self}}^{(o)},$$

the activations

$$Q_{\text{self}},\ K_{\text{self}},\ V_{\text{self}},$$

the masked attention matrix A_{self}, and the projected sublayer output Y_{self} right before the first add & norm?

Assume the decoder takes target-side hidden states $X \in \mathbb{R}^{B \times n_{\text{tgt}} \times d}$ as input to this block.

Solution: Since masked self-attention uses the decoder sequence for queries, keys, and values:

$$W_{\text{self}}^{(q)} \in \mathbb{R}^{d \times d_q}, \qquad W_{\text{self}}^{(k)} \in \mathbb{R}^{d \times d_q}, \qquad W_{\text{self}}^{(v)} \in \mathbb{R}^{d \times d_v}$$

Thus, the projected activations are

$$Q_{\text{self}}, K_{\text{self}} \in \mathbb{R}^{B \times n_{\text{tgt}} \times d_q}, \qquad V_{\text{self}} \in \mathbb{R}^{B \times n_{\text{tgt}} \times d_v}$$

To determine the shape of A_{self}, you can either reason from the definition of the attention matrix (e.g. use matrix multiplication dimension rules) or note that each target position attends over target positions, so: $A_{\text{self}} \in \mathbb{R}^{B \times n_{\text{tgt}} \times n_{\text{tgt}}}$.

From $H_{\text{self}} = A_{\text{self}} V_{\text{self}}$, we know the attention head H_{self} has shape $\mathbb{R}^{B \times n_{\text{tgt}} \times d_v}$.

From Figure 2, notice that the output of this block Y_{self} is added to X, so Y_{self} must have the same shape as X:

$$Y_{\text{self}} \in \mathbb{R}^{B \times n_{\text{tgt}} \times d}$$

which means the output projection matrix $W_{\text{self}}^{(o)}$ must have shape $\mathbb{R}^{d_v \times d}$.

(b) Assume the encoder has already processed the source sequence and produced $E \in \mathbb{R}^{B \times n_{\text{src}} \times d}$.

The cross-attention sublayer is the same as the previous part, except that the queries (Q) come from the decoder stream (same shape as Y_{self}), while the keys (K) and values (V) come from the encoder output E.

What are the shapes of

$$W_{\text{cross}}^{(q)},\ W_{\text{cross}}^{(k)},\ W_{\text{cross}}^{(v)},\ W_{\text{cross}}^{(o)},$$

the activations

$$Q_{\text{cross}},\ K_{\text{cross}},\ V_{\text{cross}},$$

the attention matrix A_{cross}, and the projected sublayer output Y_{cross} right before the second add & norm?

Solution: The projection weight matrices have the same shapes as before:

$$W_{\text{cross}}^{(q)} \in \mathbb{R}^{d \times d_q}, \qquad W_{\text{cross}}^{(k)} \in \mathbb{R}^{d \times d_q}, \qquad W_{\text{cross}}^{(v)} \in \mathbb{R}^{d \times d_v},$$

Queries come from the decoder stream, so

$$Q_{\text{cross}} \in \mathbb{R}^{B \times n_{\text{tgt}} \times d_q}$$

Keys and values come from the encoder output, so

$$K_{\text{cross}} \in \mathbb{R}^{B \times n_{\text{src}} \times d_q}, \qquad V_{\text{cross}} \in \mathbb{R}^{B \times n_{\text{src}} \times d_v}$$

To determine the shape of A_{cross}, you can either reason from the definition of the attention matrix

(e.g. use matrix multiplication dimension rules) or note that each **target** position attends over **source** positions, so: $A_{\text{cross}} \in \mathbb{R}^{B \times n_{\text{tgt}} \times n_{\text{src}}}$.

From $H_{\text{cross}} = A_{\text{cross}} V_{\text{cross}}$, we know the attention head H_{cross} has shape $\mathbb{R}^{B \times n_{\text{tgt}} \times d_v}$.

From Figure 2, notice that the output of this block Y_{cross} is added to the output of the self-attention block, so Y_{cross} must have the same shape as Y_{self}:

$$Y_{\text{cross}} \in \mathbb{R}^{B \times n_{\text{tgt}} \times d}$$

which means the output projection matrix $W_{\text{cross}}^{(o)}$ must have shape $\mathbb{R}^{d_v \times d}$.

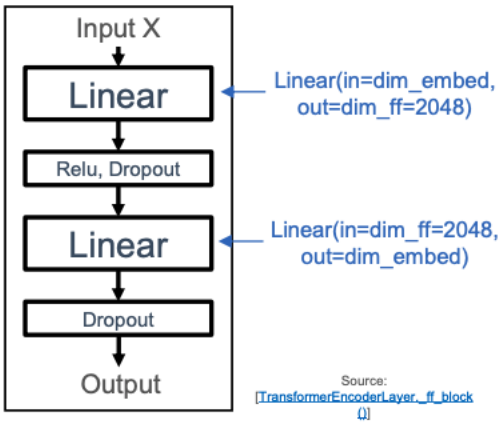


Figure 3: FNN Block (Lec 16 Slide 36).

(c) In the above **MLP/FFN sublayer**, the first linear layer expands from model width d to hidden width d_{ff}, a nonlinearity (ReLU) and dropout is applied, and the second linear layer projects back to width d.

Note: The input matrix X to the MLP block is the output of the previous add & norm (see Figure 3), which has the same shape as the output of the cross-attention block Y_{cross}. In other words, $X \in \mathbb{R}^{B \times n_{\text{tgt}} \times d}$.

$$Z = \text{Dropout}(\text{ReLU}(XW_1))$$
$$\tilde{X} = ZW_2$$

What are the shapes of the two MLP weight matrices, the hidden activation Z, and the final decoder-block output $\tilde{X}$?

Solution: The first linear layer expands the last dimension:

$$W_1 \in \mathbb{R}^{d \times d_{\text{ff}}}, \qquad XW_1 \in \mathbb{R}^{B \times n_{\text{tgt}} \times d_{\text{ff}}}.$$

The nonlinearity (ReLU) and dropout do not change shape, so the hidden activation Z is still

$$Z \in \mathbb{R}^{B \times n_{\text{tgt}} \times d_{\text{ff}}}.$$

The second linear layer projects back to the model width:

$$W_2 \in \mathbb{R}^{d_{\text{ff}} \times d}.$$

Therefore the MLP output, and hence the final output of the decoder block after the last add & norm

(after the FFN block), is

$$\tilde{X} \in \mathbb{R}^{B \times n_{\text{tgt}} \times d}.$$

3. Why Attention Is More Than a Linear Layer

Recall the equations for a single attention head:

$$Q = XW^{(q)}$$
$$K = XW^{(k)}$$
$$V = XW^{(v)}$$
$$H = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_q}}\right)V.$$

In this question, ignore batching and assume $X \in \mathbb{R}^{n \times d}$, where n is the sequence length.

(a) For this part only, ignore the softmax and normalization ($\sqrt{d_q}$) and set $W^{(q)} = W^{(k)} = W^{(v)} = I$. Show that the attention output simplifies to $H = (XX^\top)X$.

Solution: Under these assumptions we have

$$Q = X, \qquad K = X, \qquad V = X.$$

Ignoring the softmax gives

$$H = (QK^\top)V = (XX^\top)X.$$

(b) Why does the expression $(XX^\top)X$ show that attention is *nonlinear* in X? Why does it also show that attention *mixes information across tokens*?

Solution: The matrix $XX^\top$ contains all pairwise dot products between tokens:

$$(XX^\top)_{ij} = x_i^\top x_j,$$

where x_i is the ith row of X.

This already shows token mixing: entry (i, j) measures how strongly token i interacts with token j, and the product $(XX^\top)X$ uses those interaction strengths to form new token representations from *all* rows of X.

It is also nonlinear in X. A standard linear map has the form XW and is linear in the entries of X. In contrast, $XX^\top$ contains products of entries of X, and the full expression $(XX^\top)X$ contains higher-order products of entries of X. So the mapping $X \mapsto (XX^\top)X$ is not linear.

(c) Now restore the learned projections and the softmax. Rewrite the attention output in the form $H = A(X)XW^{(v)}$, where $A(X)$ is a function that takes in X as input and returns an $n \times n$ matrix that depends on the current input. **Hint:** $(AB)^\top = B^\top A^\top$ for any matrices A and B.

In what sense can attention be viewed as a *data-dependent* linear layer? How is this different from an ordinary linear layer XW?

Solution: We can write

$$\begin{aligned} H &= \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_q}}\right)V \\ &= \text{Softmax}\left(\frac{XW^{(q)}(XW^{(k)})^\top}{\sqrt{d_q}}\right)XW^{(v)} \\ &= \text{Softmax}\left(\frac{XW^{(q)}W^{(k)\top}X^\top}{\sqrt{d_q}}\right)XW^{(v)}. \end{aligned}$$

So if we define

$$A(X) = \text{Softmax}\left(\frac{XW^{(q)}W^{(k)\top}X^\top}{\sqrt{d_q}}\right),$$

then

$$H = A(X)XW^{(v)}.$$

This looks like a linear transformation applied to X, except the token-mixing matrix $A(X)$ is not fixed in advance: it is computed from the current input sequence. So different inputs induce different effective weights.

By contrast, an ordinary linear layer applies the same weight matrix W to every example:

$$X \mapsto XW.$$

The weights do not depend on the input, and there is no analogous data-dependent token-to-token interaction matrix.

4. Why the Decoder Looks the Way It Does

This final question focuses on several design choices that appear in encoder-only, decoder-only, and encoder-decoder transformers.

(a) Suppose we remove positional encodings entirely from a transformer. What kind of information would the model lose? Give one concrete example of two sentences that would become difficult to distinguish based on meaning.

Solution: Without positional encodings, the model has no direct way to tell which token came first, second, third, and so on. In other words, it loses word-order information.

For example, the sentences

```
the cat chased the mouse
```

and

```
the mouse chased the cat
```

contain the same tokens but have different meanings because of their order. Without positional information, a transformer would struggle to represent that distinction.

(b) Give one common use case for each transformer variant below.

i. encoder-only

ii. decoder-only

iii. encoder-decoder

Solution: Typical examples are:

i. encoder-only: building strong token or sequence representations for tasks such as classification or tagging,

ii. decoder-only: autoregressive generation tasks such as next-token prediction, chat, or text completion (ChatGPT is an example),

iii. encoder-decoder: sequence-to-sequence tasks such as machine translation or summarization.

(c) In the decoder, why do we apply a *causal mask* in the self-attention block for tasks such as machine translation?

Solution: The decoder predicts the target sequence one token at a time. When predicting token t, it should only be allowed to use earlier target tokens, not future ones.

The causal mask prevents the model from *looking ahead*. Without it, during training the decoder could directly attend to the correct future target tokens and learn to rely on information that will not be available at inference time.

(d) Do we also apply a *causal mask* in the cross-attention block of an encoder-decoder transformer for machine translation? Why or why not?

Solution: No. In cross-attention, the decoder queries the *encoder output*, which represents the full source sentence. For machine translation, when generating a target token, we generally want the decoder to be able to attend to *any* source position. For example, languages can have different subject-verb-object word orders. In English, "I gave her the apple" has the subject ("I"), verb ("gave"), and object ("her") in that order, while in Spanish the equivalent sentence would be "Le di la manzana" with the object ("Le"), verb ("di"), and subject (implied to be "Yo" in Spanish or "I" in English) in a different order. If the decoder could only attend to earlier source positions, it might struggle to learn such cross-lingual word-order differences.

Therefore, cross-attention does not use a causal mask over source positions. However, we may still use a *padding mask* so that the decoder ignores padded source tokens.

Contributors:

- Eric Kim.
- Terry Kim.

