10 Attention, Self-Attention, and Transformer Algebra

Chapter Abstract. Recurrent models move information through a chain of hidden states. Attention changes the graph: each token can compare directly with other tokens and read a weighted mixture of their value vectors. This chapter develops attention from the phrase `not very good`. It defines queries, keys, values, score matrices, row-wise softmax, masks, positional information, multi-head attention, cross-attention, and the transformer block. The goal is practical mathematical literacy: a reader should be able to check transformer shapes, spot masking mistakes, understand why position signals are needed, and connect the algebra to common deep-learning design choices.

10.1 Running Example and Chapter Roadmap

The previous chapter ended with a limitation of recurrence. In a left-to-right recurrent model, information from an early token travels through a chain of hidden states before it reaches a later prediction. The phrase

```
not very good
```

shows why this matters. The last token `good` looks positive by itself, but a sentiment model should also account for the earlier token `not`. Attention gives a later position a direct algebraic route to earlier positions.



Figure 10.1: Attention can make a later representation depend strongly on earlier context. In the running example, `good` should not be read independently of `not`.

Transformer attention can be read through three matrices: scores, row-wise weights, and weighted value mixtures. The central formulas are

$$S = \frac{QK^\top}{\sqrt{d_k}}, \qquad A = \mathrm{softmax}_{\mathrm{row}}(S), \qquad H = AV.$$

Every symbol in these equations will be tied to a shape, an operation, and a reading of the running example.

The central object will be the row-weight matrix A.

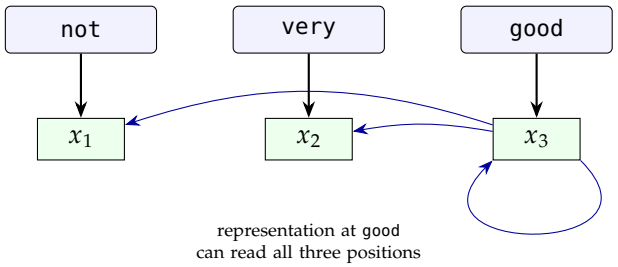


Figure 10.2: Self-attention lets each token representation compare directly with other token representations. The drawing shows one reading position, `good`, reading from all positions.

Definition 10.1.1 (Sequence matrix) *For a sequence of T embedded tokens, the sequence matrix is*

$$X = \begin{bmatrix} x_1^\top \\ x_2^\top \\ \vdots \\ x_T^\top \end{bmatrix} \in \mathbb{R}^{T \times d_{\text{model}}},$$

where row i contains the embedding or current representation of token position i.

For `not very good`, $T = 3$. The row $x_1^\top$ represents `not`, $x_2^\top$ represents `very`, and $x_3^\top$ represents `good`. The subscript marks position, not exponentiation.

Pause and predict

Before reading any equations, name one token that the representation of `good` should probably compare with in the phrase `not very good`. Give a short reason.

The phrase `not very good` shows why recurrence is not the only way to share context across positions. Attention lets each token compare directly with the others through queries, keys, and values. Masks decide which comparisons are legal, positional information tells the model where tokens sit, and multi-head attention repeats the comparison in several learned subspaces before the pieces are assembled into transformer blocks.

10.2 From Recurrence to Attention

Recurrent models are built from a state update such as

$$h_t = f_\theta(h_{t-1}, x_t).$$

This rule is natural for streaming input, because the model can update its state as new tokens arrive. It also creates a chain. Information from x_1 reaches h_T only after passing through $h_1, h_2, \ldots, h_{T-1}$.

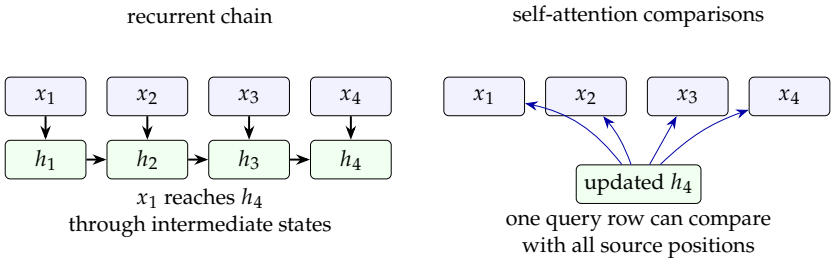


Figure 10.3: Recurrence moves information along a chain of hidden states. A self-attention layer forms direct query-source comparisons before mixing value vectors.

Attention compares positions before it mixes information.

Attention replaces the chain bottleneck with direct comparisons. For every query position i and source position j, the model computes a score that says how compatible those two positions are. Early neural sequence models used attention to let a decoder, the component producing output tokens, look back at encoder states, the input-side representations, in translation [6, 141]. Transformers made this comparison the main sequence operation [222].

Definition 10.2.1 (Position comparison) *A position comparison is a scalar score $S_{i,j}$ assigned to an ordered pair of positions (i, j). The first index i is the position doing the reading. The second index j is the position being considered as a source of information.*

In `not very good`, the comparison $S_{3,1}$ measures how much the query position `good` is compatible with the source position `not`. A high score does not mean the word `not` is copied verbatim. It means the learned algebra is making `not` available to the updated representation at the `good` position.

This gives self-attention a direct comparison path. In one self-attention layer over T positions, the representation at position i can compare directly with any source position $j \in \{1, \ldots, T\}$, unless the architecture explicitly forbids that pair.

The layer forms a table of position comparisons. If the pair (i, j) is allowed, the table contains an entry for source position j in row i. The later mixing step can therefore use information from j when updating position i. No intermediate hidden states between j and i are required in this layer.

This observation is about graph structure, not guaranteed model quality. Learning useful comparisons still depends on the loss, the data, the optimizer, and the architecture around the attention layer. The practical gain is that the architecture makes direct comparison possible and parallelizable.

10.3 Queries, Keys, and Values

Attention starts by projecting the sequence matrix into three new matrices:

$$Q = XW_Q, \qquad K = XW_K, \qquad V = XW_V.$$

The rows of Q are queries, the rows of K are keys, and the rows of V are values. The names are useful, but the objects are ordinary learned

linear projections. Their entries are trained through the same loss and backpropagation machinery as other neural-network weights.

Definition 10.3.1 (Query-key-value projections) *Given* $X \in \mathbb{R}^{T \times d_{\text{model}}}$, *define*

$$W_Q, W_K \in \mathbb{R}^{d_{\text{model}} \times d_k}, \qquad W_V \in \mathbb{R}^{d_{\text{model}} \times d_v}.$$

Then

$$Q = XW_Q \in \mathbb{R}^{T \times d_k}, \quad K = XW_K \in \mathbb{R}^{T \times d_k}, \quad V = XW_V \in \mathbb{R}^{T \times d_v}.$$

The row $q_i^\top$ is the query for position i, $k_j^\top$ is the key for position j, and $v_j^\top$ is the value for position j.

The query and key dimensions match because dot products $q_i^\top k_j$ will be used as compatibility scores. The value dimension may differ, because values are the vectors being mixed after the scores are normalized.

Queries and keys must share dimension d_k.

Object	Shape	Role
Q	$T \times d_k$	each row asks what one position should read
K	$T \times d_k$	each row is compared against queries
V	$T \times d_v$	each row is information that can be mixed into an output

For `not very good`, the query for `good` is q_3. The key for `not` is k_1. If $q_3^\top k_1$ is large relative to the other scores in row 3, then the value vector v_1 will have a strong effect on the updated representation h_3.

Pause and predict

Let $X \in \mathbb{R}^{3 \times 4}$, $W_Q \in \mathbb{R}^{4 \times 2}$, and $W_K \in \mathbb{R}^{4 \times 2}$. Predict the shapes of Q, K, and $QK^\top$. Which axis of the score matrix indexes the query token, and what shape error would appear if K were not transposed?

Shape checks are not cosmetic. A large fraction of implementation errors in attention layers are dimension errors: transposing the wrong axis, mixing batch and sequence axes, or giving queries and keys incompatible dimensions. Reading the algebra by shape is a practical debugging skill.

10.4 Scaled Dot-Product Attention

The most common attention formula in transformers is scaled dot-product attention [222]. It has three steps: score, normalize, and mix.

The object matters because it turns pairwise token comparisons into a routing table. Row i of the attention matrix says how position i reads value vectors from the available source positions.

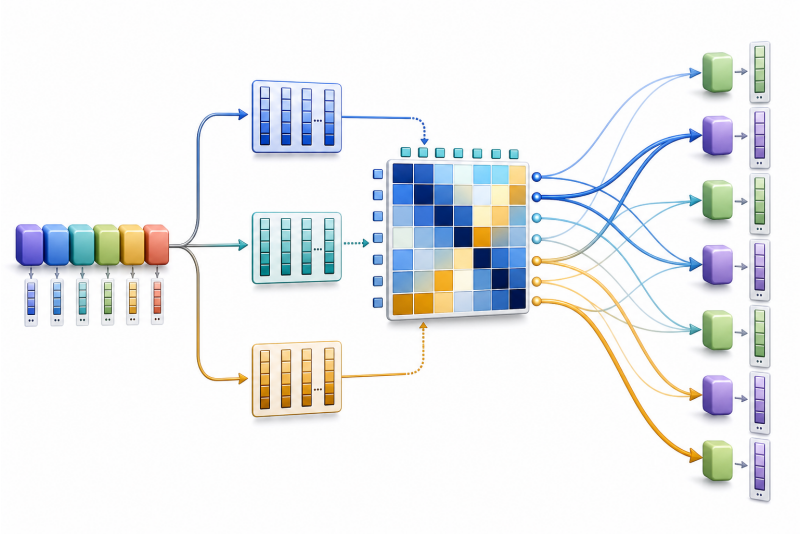


Figure 10.4: Self-attention routes information by comparing query rows with key rows, normalizing the scores into attention weights, and mixing value rows. Thicker paths indicate stronger routes from source positions into the updated representations.

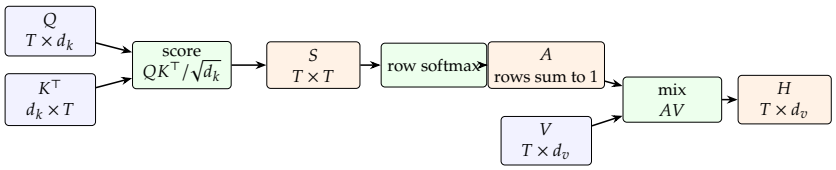


Figure 10.5: Scaled dot-product attention has a score-normalize-mix pipeline. The middle $T \times T$ matrix records which source positions each query position reads from.

Definition 10.4.1 (Scaled dot-product attention) *For $Q, K \in \mathbb{R}^{T \times d_k}$ and $V \in \mathbb{R}^{T \times d_v}$, scaled dot-product attention is*

$$S = \frac{QK^\top}{\sqrt{d_k}} \in \mathbb{R}^{T \times T},$$
$$A = \mathrm{softmax}_{\mathrm{row}}(S) \in \mathbb{R}^{T \times T},$$
$$H = AV \in \mathbb{R}^{T \times d_v}.$$

Row-wise softmax means

$$A_{i,j} = \frac{\exp(S_{i,j})}{\sum_{\ell=1}^{T} \exp(S_{i,\ell})}.$$

The scale factor $1/\sqrt{d_k}$ keeps dot products from becoming too large as d_k grows. Without it, the softmax can become very sharp early in training, which makes gradients less useful.

Proposition 10.4.1 (Variance of scaled dot products) *Let $q, k \in \mathbb{R}^{d_k}$ have independent coordinates with mean 0 and variance 1, and assume the coordinates of q are independent of the coordinates of k. Then*

$$\mathbb{E}[q^\top k] = 0, \qquad \mathrm{Var}(q^\top k) = d_k, \qquad \mathrm{Var}\left(\frac{q^\top k}{\sqrt{d_k}}\right) = 1.$$

Proof. Write $q^\top k = \sum_{\ell=1}^{d_k} q_\ell k_\ell$. For each coordinate, $\mathbb{E}[q_\ell k_\ell] = \mathbb{E}[q_\ell]\mathbb{E}[k_\ell] = 0$. Also,

$$\operatorname{Var}(q_\ell k_\ell) = \mathbb{E}[q_\ell^2]\mathbb{E}[k_\ell^2] = 1.$$

The coordinate products are independent under the assumptions, so the variance of their sum is d_k. Dividing by $\sqrt{d_k}$ divides the variance by d_k, leaving variance 1. □

The assumptions are an initialization-scale model, not a promise that every trained query and key vector has independent unit-variance entries. The result still explains the design pressure: without the scaling, score magnitudes tend to grow with the key dimension. In numerical code, the row maximum is often subtracted before exponentiation:

$$\operatorname{softmax}(s)_j = \frac{\exp(s_j - \max_\ell s_\ell)}{\sum_r \exp(s_r - \max_\ell s_\ell)}.$$

This produces the same probabilities but avoids unnecessary overflow.

Stable softmax subtracts a row maximum.

Example 10.4.1 (Attention output shape) If $Q, K \in \mathbb{R}^{T\times d_k}$ and $V \in \mathbb{R}^{T\times d_v}$, then

$$\operatorname{Attention}(Q, K, V) = AV \in \mathbb{R}^{T\times d_v}.$$

The product $QK^\top$ has shape $(T \times d_k)(d_k \times T) = T \times T$. Row-wise softmax does not change the shape, so $A \in \mathbb{R}^{T\times T}$. Therefore AV has shape $(T \times T)(T \times d_v) = T \times d_v$.

Row-wise softmax makes attention row-stochastic. Every row of $A = \operatorname{softmax}_{\text{row}}(S)$ has nonnegative entries that sum to 1.

For each fixed row i, $\exp(S_{i,j}) > 0$ for every j, and the denominator $\sum_{\ell=1}^{T} \exp(S_{i,\ell})$ is positive. Hence $A_{i,j} \geq 0$. Summing over j gives

$$\sum_{j=1}^{T} A_{i,j} = \frac{\sum_{j=1}^{T} \exp(S_{i,j})}{\sum_{\ell=1}^{T} \exp(S_{i,\ell})} = 1.$$

Example 10.4.2 (A tiny end-to-end attention pass) Let $T = 3$, $d_k = d_v = 2$, and use identity projections so that

$$Q = K = V = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}.$$

Then

$$QK^\top = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 2 \end{bmatrix}, \qquad S = \frac{QK^\top}{\sqrt{2}} \approx \begin{bmatrix} 0.71 & 0 & 0.71 \\ 0 & 0.71 & 0.71 \\ 0.71 & 0.71 & 1.41 \end{bmatrix}.$$

For the third query row, the softmax is approximately

$$\text{softmax}(0.71, 0.71, 1.41) \approx (0.25, 0.25, 0.50).$$

Therefore the third output row is

$$h_3 \approx 0.25(1, 0) + 0.25(0, 1) + 0.50(1, 1) = (0.75, 0.75).$$

The example uses identity projections only to keep the arithmetic visible; in a trained transformer, W_Q, W_K, and W_V are learned.

For a small hand calculation, suppose the score row for the query `good` is

$$s = (2, 0, 1).$$

Using $e^2 \approx 7.39$, $e^1 \approx 2.72$, and $e^0 = 1$, the softmax row is approximately

$$\frac{(7.39, 1, 2.72)}{7.39 + 1 + 2.72} \approx (0.67, 0.09, 0.24).$$

If the scalar values were $v_1 = -1$, $v_2 = 0$, and $v_3 = 2$, the attended output would be

$$0.67(-1) + 0.09(0) + 0.24(2) \approx -0.19.$$

The negative contribution from `not` can therefore change the representation at `good`.

Pause and predict

If the largest entry in one score row becomes much larger than the others, what happens to the corresponding softmax row?

10.5 The Attention Matrix as a Weighted Graph

The matrix A can be read as a directed weighted graph over token positions. The entry $A_{i,j}$ is the weight on the edge from query position i to source position j. The row $A_{i,:}$ describes where position i reads from.

Definition 10.5.1 (Attention graph) *An attention graph for T positions is the directed weighted graph with nodes $1, \ldots, T$ and edge weights $A_{i,j}$. The weights are row stochastic: $A_{i,j} \geq 0$ and $\sum_j A_{i,j} = 1$.*

For `not very good`, a plausible sentiment-oriented attention row for `good` might put substantial mass on `not`. A different head or task might emphasize `very` or `good` itself. These examples describe possible learned patterns, not rules built into the architecture.

Attention weights are diagnostics, not full explanations.

Query row	`not`	`very`	`good`	Interpretation
`not`	0.60	0.25	0.15	mostly self-focused
`very`	0.20	0.45	0.35	reads local context
`good`	0.67	0.09	0.24	reads negation strongly

Each value mixture is a convex combination. For each position i, the attended output

$$h_i = \sum_{j=1}^{T} A_{i,j} v_j$$

is a convex combination of the value vectors $v_1, \ldots, v_T$.

The reason is the row-wise softmax: $A_{i,j} \geq 0$ and $\sum_j A_{i,j} = 1$. A weighted sum with nonnegative weights that sum to one is a convex combination.

Attention heatmaps are useful for debugging. They can reveal that a padding token receives weight, that a causal model is reading future tokens, or that a head has nearly identical rows for many inputs. They are weaker as explanations of model reasoning. Different internal mechanisms can produce the same final prediction, and attention weights alone need not identify the causal reason for that prediction [99].

Pause and predict

If a padding token received attention weight 0.40 from several real-token rows, what debugging failure would that suggest? Predict how this could distort the value mixtures even if the final prediction looks plausible.

10.6 Masks for Padding and Causality

Attention scores are usually masked before the softmax. A mask changes which source positions a query position is allowed to read. Two masks are especially important in applied work: padding masks and causal masks.

Definition 10.6.1 (Attention mask) *An attention mask is a matrix* $M \in (\mathbb{R} \cup \{-\infty\})^{T \times T}$ *added to the score matrix before row-wise softmax:*

$$A = \mathrm{softmax}_{\mathrm{row}}(S + M).$$

The entry

$$M_{i,j} = \begin{cases} 0, & \text{if position } i \text{ may read position } j, \\ -\infty, & \text{if position } i \text{ may not read position } j. \end{cases}$$

A padding mask blocks fake positions introduced only to make a batch rectangular. A causal mask blocks future positions. A decoder is a model component that predicts output tokens; causal means that each position may read only positions available so far. For a three-token decoder sequence, the causal mask has the pattern

$$M = \begin{bmatrix} 0 & -\infty & -\infty \\ 0 & 0 & -\infty \\ 0 & 0 & 0 \end{bmatrix}.$$

Thus the representation at `very` may read `not` and `very`, but not `good`. The representation at `good` may read all three positions.

Mask before softmax, not after value mixing.

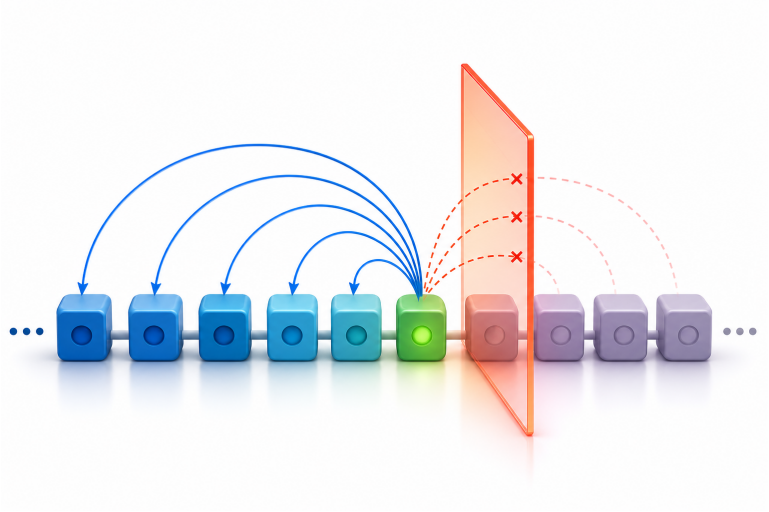


Figure 10.6: A causal decoder position can read only the positions available so far. Future positions must be blocked before the softmax to prevent leakage.

query row \ source column	not	very	good
not	0	$-\infty$	$-\infty$
very	0	0	$-\infty$
good	0	0	0

Figure 10.7: A causal mask is lower triangular. Hatched entries are future positions that receive zero attention after the masked softmax.

Example 10.6.1 (Masked row arithmetic) Suppose the unmasked score row for the second token, `very`, is

$$s = (2, 0, 1).$$

A causal mask blocks the future third position, so the masked row is

$$(2, 0, -\infty).$$

The softmax probabilities become

$$\frac{(e^2, e^0, 0)}{e^2 + e^0} \approx (0.88, 0.12, 0).$$

Without the mask, the future token `good` would receive positive attention weight. With the mask, its weight is exactly 0.

Blocked attention weights become zero. If $M_{i,j} = -\infty$, then the masked attention weight $A_{i,j}$ is 0.

The masked score for that entry is $S_{i,j} + M_{i,j} = -\infty$. Since $\exp(-\infty) = 0$, the numerator of the softmax term is 0. The denominator is positive as long as at least one source position in the row is allowed. Therefore $A_{i,j} = 0$.

Mask placement matters. If a causal mask is applied after $H = AV$, then future values have already contributed to the mixture. That is data leakage. The model can appear to perform well during training while using information that would not be available at prediction time.

Pause and predict

For a four-token causal decoder, which entries in row 2 are allowed and which entries are blocked?

10.7 Positional Information

Self-attention compares rows of X. By itself, it does not know that one row is first and another row is last. A transformer therefore needs position information.

Definition 10.7.1 (Positional representation) *A positional representation for length T is a collection of position-dependent objects $p_1, \dots, p_T$ together with a rule Γ that forms the position-aware row*

$$z_i = \Gamma(x_i, p_i).$$

In the absolute additive case,

$$Z = X + P,$$

where $P \in \mathbb{R}^{T \times d_{\text{model}}}$ contains one position vector per row. Attention is then applied to Z instead of to X.

Absolute positional encodings were used in the original transformer [222]. Later variants introduced relative position representations [192] and rotary position embeddings [203], which encode positions by rotating query and key coordinates. The common purpose is the same: token identity alone is not enough when order matters.

Attention needs a separate signal for order.

Proposition 10.7.1 (Permutation equivariance without positions) *Let P_π be a permutation matrix that reorders the T sequence positions. For self-attention without positional information,*

$$\begin{aligned}&\text{Attention}(P_\pi X W_Q, P_\pi X W_K, P_\pi X W_V)\\ &\quad = P_\pi \,\text{Attention}(X W_Q, X W_K, X W_V).\end{aligned}$$

Proof. Let $Q = XW_Q$, $K = XW_K$, and $V = XW_V$. After reordering rows, $Q' = P_\pi Q$, $K' = P_\pi K$, and $V' = P_\pi V$. The score matrix becomes

$$S' = \frac{Q'K'^\top}{\sqrt{d_k}} = \frac{P_\pi Q K^\top P_\pi^\top}{\sqrt{d_k}} = P_\pi S P_\pi^\top.$$

Row-wise softmax preserves this simultaneous row and column reordering, so $A' = P_\pi A P_\pi^\top$. Therefore

$$A'V' = (P_\pi A P_\pi^\top)(P_\pi V) = P_\pi A V.$$

□

The result states the limitation precisely: without a position signal, reordering the input rows reorders the output rows in the same way. If a later classifier pools over positions, it can lose the difference between `not very good` and `good very not`. Position vectors break this symmetry by giving the first, second, and third positions different mathematical identities.

Pause and predict

If positional vectors are removed and a later classifier mean-pools token representations, predict what should happen to the scores for `not very good` and `good very not`. Which symmetry is causing the problem?

10.8 Multi-Head Attention

One attention matrix gives one learned comparison pattern. Multi-head attention runs several such patterns in parallel and then combines their outputs [222].

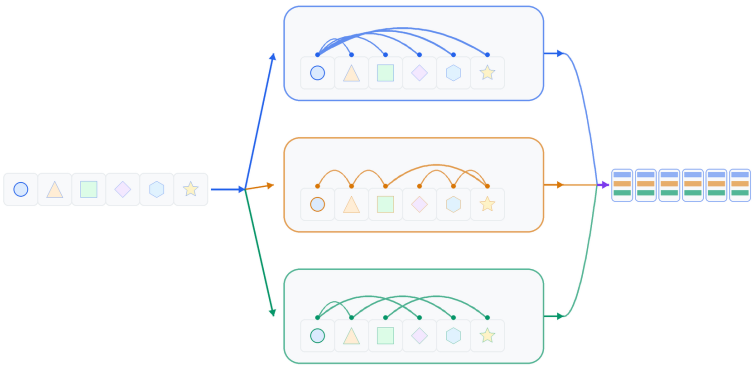


Figure 10.8: Multi-head attention gives the model several parallel comparison spaces over the same sequence before the results are merged back into one representation.

Definition 10.8.1 (Multi-head attention) *For m heads, define head-specific projections*

$$H^{(r)} = \text{Attention}\left(XW_Q^{(r)}, XW_K^{(r)}, XW_V^{(r)}\right), \qquad r = 1, \dots, m.$$

Then

$$\text{MHA}(X) = \text{Concat}(H^{(1)}, \dots, H^{(m)})W_O.$$

If each head has value dimension d_v, then concatenation produces a $T \times md_v$ matrix. The final projection W_O maps that concatenated representation back to the model dimension used by the rest of the block.

Heads are parallel learned comparison spaces.

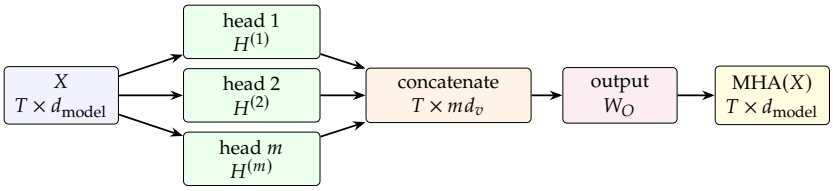


Figure 10.9: Multi-head attention runs several learned comparison spaces in parallel, concatenates their outputs, and projects back to the model dimension.

Example 10.8.1 (Multi-head output shape) If each head output $H^{(r)}$ has shape $T \times d_v$ and $W_O \in \mathbb{R}^{md_v \times d_{\text{model}}}$, then $\text{MHA}(X) \in \mathbb{R}^{T \times d_{\text{model}}}$. Concatenating m matrices of shape $T \times d_v$ along the feature axis gives a matrix of shape $T \times md_v$. Multiplication by $W_O \in \mathbb{R}^{md_v \times d_{\text{model}}}$ gives shape $T \times d_{\text{model}}$.

For `not very good`, one head might learn a comparison that is sensitive to negation, while another might focus on local modifiers. This is a useful mental model, not a promise about a particular trained model. Head behavior is learned from the task and can be redundant, specialized, or hard to interpret.

Pause and predict

If each of 4 heads returns a $T \times 8$ matrix, what is the concatenated shape before W_O? If the rest of the transformer block expects feature dimension d_{model}, why is the final projection needed after concatenation?

10.9 Cross-Attention and Encoder-Decoder Attention

Self-attention uses the same sequence to make queries, keys, and values. Cross-attention uses queries from one sequence and keys and values from another sequence. This distinction is central in encoder-decoder models and later multimodal systems.

Definition 10.9.1 (Cross-attention) *Let $Y \in \mathbb{R}^{T_y \times d_{\text{model}}}$ be a query-side sequence and $X \in \mathbb{R}^{T_x \times d_{\text{model}}}$ be a source-side sequence. A cross-attention layer may use*

$$Q = YW_Q, \qquad K = XW_K, \qquad V = XW_V.$$

The score matrix has shape $T_y \times T_x$, and the output has one row for each query-side position.

In translation, a decoder position can query the encoder representations of a source sentence [6, 141]. In image captioning or vision-language systems,

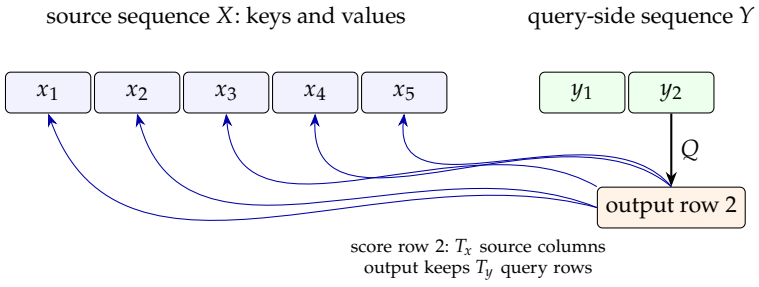


Figure 10.10: Cross-attention uses query rows from one sequence and reads keys and values from another sequence.

text positions may query image features. The same algebra handles both cases: queries determine the rows, keys and values determine the source columns and mixed information.

Cross-attention reads one representation set from another.

Pause and predict

If Y has 5 positions and X has 12 positions, which sequence controls the score-matrix rows and which controls the columns? Predict the score shape, then predict how many output rows remain after the values are mixed.

Attention is not limited to one sequence attending to itself. Later chapters can reuse the same query-key-value notation for language modeling, vision transformers, and multimodal alignment.

10.10 The Transformer Block

A transformer block combines attention with residual connections, normalization, and a position-wise feedforward network. The original transformer used residual connections around both sublayers [222], and layer normalization is a standard stabilizing component [4].

Definition 10.10.1 (Post-normalization transformer block) *For $X \in \mathbb{R}^{T \times d_{\text{model}}}$, a post-normalization transformer block is the map*

$$U = \mathrm{LN}(X + \mathrm{MHA}(X)), \qquad Y = \mathrm{LN}(U + \mathrm{FFN}(U)),$$

where LN *is layer normalization and the position-wise feedforward network is*

$$\mathrm{FFN}(u) = W_2 \phi(W_1 u + b_1) + b_2.$$

The same feedforward parameters are applied independently to each token position.

The attention sublayer communicates across positions. The feedforward sublayer transforms each position separately. This alternating structure is a useful way to read transformer blocks: communication, then nonlinear processing, then communication again in the next block.

Attention mixes positions; the FFN processes each row.

Residual connections make the block learn changes to a representation instead of replacing the representation from scratch. This connects to the residual architecture ideas used in deep convolutional networks [84]. Layer normalization rescales features within a token representation, which helps stabilize the depth of the stack.

Pause and predict

If MHA were removed but the feedforward network were made much wider, could token i use token j's information within the same block? Predict which algebraic role would still be missing.

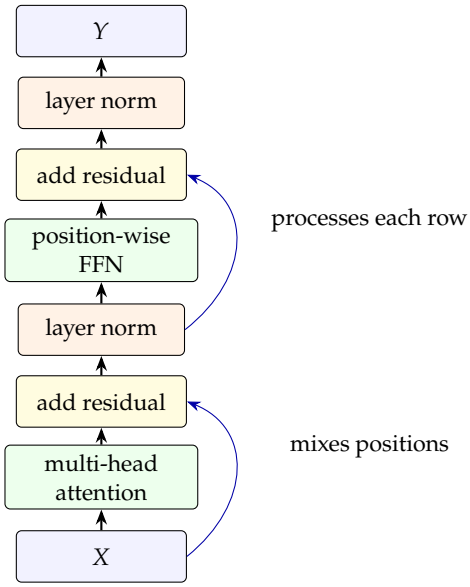


Figure 10.11: A transformer block alternates cross-position communication through attention with position-wise nonlinear processing, while residual paths carry the representation forward.

Many modern implementations use a pre-normalization variant, where normalization is applied before each sublayer. The exact placement affects optimization stability, especially in deep stacks. The algebraic roles remain the same: attention exchanges information across positions, the feedforward network transforms features within positions, and residual paths carry the representation forward.

10.11 Encoder, Decoder, and Decoder-Only Transformers

A transformer family is mostly a mask and source choice.

Transformer names are less important than the information pattern. Each block chooses a query sequence, a source sequence for keys and values, and a mask that decides which source columns each query row may read.

Definition 10.11.1 (Transformer information pattern) *For query-side rows $Y \in \mathbb{R}^{T_y \times d_{\text{model}}}$ and source-side rows $X \in \mathbb{R}^{T_x \times d_{\text{model}}}$, an attention sublayer computes*

$$Q = YW_Q, \qquad K = XW_K, \quad V = XW_V,$$
$$A = \operatorname{softmax}_{\text{row}}\left(\frac{QK^\top + M}{\sqrt{d_k}}\right),$$

where $M_{i,j} = 0$ for allowed reads and $M_{i,j} = -\infty$ for blocked reads. The output has T_y rows, one for each query position.

Pattern	Query rows	Key/value rows	Allowed columns
Encoder self-attention	input X	same input X	all non-padding positions
Decoder-only self-attention	generated prefix X	same prefix X	$j \le i$ in row i
Encoder-decoder cross-attention	decoder states Y	encoder states X	source positions not masked as padding

Encoder-style models therefore read a complete observed input [49]. Decoder-style language models keep the causal information set needed for next-token prediction [27]. Encoder-decoder models add a second source of keys and values for text-to-text conditioning [170].

Proposition 10.11.1 (Causal rows are prefix functions) *Suppose a decoder-only self-attention layer uses the mask $M_{i,j} = -\infty$ whenever $j > i$. If the remaining sublayers act row by row, then output row i can depend on $x_1, \dots, x_i$ but not on $x_{i+1}, \dots, x_T$.*

Proof. In row i, masked columns $j > i$ receive softmax weight 0. The attention mixture for row i is therefore a weighted sum only of $v_1, \dots, v_i$. Residual addition, layer normalization, and the feedforward network do not mix different rows. Thus no later token can enter row i through this block. □

The same argument applies by induction through a stack of causal blocks: if row i is a function only of the prefix before one block, it remains a function only of that prefix after the block.

After the access constraint is fixed, the separate question is expressivity: which ordered sequence maps can the architecture represent when positions are available to the model?

Theorem 10.11.2 (Transformer universal approximation, informal) *For fixed maximum sequence length and compact input domain, transformer architectures with positional encodings can approximate arbitrary continuous sequence-to-sequence functions to any desired accuracy. Without positional information, the natural universality statement is restricted to permutation equivariant sequence functions.*

This theorem answers an expressivity question, not a training guarantee. It says the architecture can represent broad sequence maps in principle; it does not say how many examples, heads, layers, or optimizer steps are needed to learn the desired map [233].

The decoding algorithms and probability objectives belong to the next chapter. The mathematical bridge is the prefix property: causal transformer blocks can be used inside an autoregressive factorization because each position's logits are computed from the prefix available at that position.

10.12 Cost, Diagnostics, and Applied Checks

Full self-attention over T positions forms a $T \times T$ score matrix. That matrix is the source of both power and cost. Direct all-pairs comparison is useful, but the number of scores grows quadratically with sequence length.

Length T	Scores T^2	Practical reading
16	256	tiny example
64	4096	small paragraph or patch grid
256	65536	moderate context
1024	1048576	more than one million pairwise scores per head

This table is not an implementation benchmark; it isolates the algebraic growth of the attention matrix. Real memory use also depends on batch size, number of heads, precision, activations saved for backpropagation, and kernel implementation.

Full attention has a $T \times T$ score matrix.

Example 10.12.1 (Full-attention scaling) For one attention head with $Q, K \in \mathbb{R}^{T \times d_k}$ and $V \in \mathbb{R}^{T \times d_v}$, full self-attention stores T^2 attention scores. Computing $QK^\top$ uses $O(T^2 d_k)$ arithmetic operations, and computing AV uses $O(T^2 d_v)$. Doubling T multiplies the number of scores by four. The score matrix has one entry for each ordered pair of query and key positions, so it has $T \cdot T = T^2$ entries. Each score is a dot product in $\mathbb{R}^{d_k}$, giving $O(T^2 d_k)$ work for all scores. The output row $(AV)_i$ is a weighted sum of T value rows in $\mathbb{R}^{d_v}$, giving $O(T^2 d_v)$ work across all i. Replacing T by $2T$ changes T^2 to $4T^2$.

Before training a transformer block, check the algebra:

- Verify that $QK^\top$ has the intended $T_{\text{query}} \times T_{\text{source}}$ shape.
- Check that padding columns receive attention weight 0 after masking.
- Check that causal row i assigns weight 0 to every column $j > i$.
- Check that a position signal is present when order matters.
- Inspect attention heatmaps as diagnostics, while remembering that they are not complete explanations.

Pause and predict

If the sequence length doubles from T to $2T$, by what factor does the number of full self-attention scores increase? Then predict what happens to memory pressure and why this is more than a merely twice-as-long input.

A model can have the right loss function and still be invalid if the score matrix has the wrong shape, if the mask leaves nonzero mass on future tokens, or if padded positions are treated as real input. Shape and mask audits catch those failures before a training curve hides them.

10.13 Summary

Attention lets sequence positions compare directly with other positions. Given a sequence matrix X, learned projections form queries, keys, and values: $Q = XW_Q$, $K = XW_K$, and $V = XW_V$. The score matrix $S = QK^\top/\sqrt{d_k}$ contains one compatibility score for each query-source pair.

Row-wise softmax turns scores into an attention matrix A. Each row of A is nonnegative and sums to 1, so each output vector is a weighted average of value vectors. The graph interpretation is direct: $A_{i,j}$ is the weight with which position i reads from position j.

Masks define which positions may be read. Padding masks block artificial positions introduced by batching. Causal masks block future positions and are essential for autoregressive models. Masks belong before softmax, because the softmax must assign zero probability to blocked entries.

Self-attention without position information is permutation equivariant. A transformer therefore needs positional representations when order matters. Multi-head attention runs several learned comparison spaces in parallel, then uses a final projection to return to the model dimension.

A transformer block alternates attention-based communication across positions with position-wise nonlinear processing. Encoder, decoder, and decoder-only models differ mainly in their masks and in whether queries read from the same sequence or from another sequence.

10.14 Exercises

Use the following eleven prompts to practice transformer shape reasoning, softmax calculations, masking, position information, and cost estimates. Include the calculation whenever a numerical answer is requested.

Exercise 1. Let $X \in \mathbb{R}^{4\times 6}$, $W_Q, W_K \in \mathbb{R}^{6\times 3}$, and $W_V \in \mathbb{R}^{6\times 5}$. Find the shapes of Q, K, V, $S = QK^\top$, $A = \mathrm{softmax}_{\mathrm{row}}(S)$, and $H = AV$.

Exercise 2. Compute the row-wise softmax of $s = (0, 1)$. Use $e^0 = 1$ and $e^1 \approx 2.72$, and round each probability to two decimals.

Exercise 3. Let one attention row be $(0.2, 0.5, 0.3)$, and let the scalar values be $v_1 = 4$, $v_2 = -2$, and $v_3 = 10$. Compute the attended scalar output.

Exercise 4. For `not very good`, describe one attention pattern that could help a sentiment model classify the phrase. State which query token reads from which source token.

Exercise 5. Explain why a padding or causal mask is added before the softmax rather than after the product AV.

Exercise 6. Write the allowed and blocked pattern for a four-token causal mask. Use 0 for allowed entries and $-\infty$ for blocked entries.

Exercise 7. Compare `not very good` with `good very not`. Explain why self-attention without position information treats one phrase as a row-reordered version of the other.

Exercise 8. Suppose $m = 4$ heads each return a matrix of shape $T \times 8$. What is the shape after concatenation? If the final projection returns to $d_{\mathrm{model}} = 32$, what is the shape after W_O? In cross-attention with U decoder queries and T encoder keys, what is the score-matrix shape?

Exercise 9. In a transformer block, identify which sublayer communicates across token positions and which sublayer transforms each token position independently. Then state one structural difference between an encoder-decoder decoder block and a decoder-only block.

Exercise 10. Full self-attention over T tokens forms T^2 scores per head. How many scores are formed for $T = 128$? If the length doubles to 256, by what factor does the score count change?

Exercise 11. Assume $q, k \in \mathbb{R}^{d_k}$ have independent zero-mean, unit-variance coordinates. Derive $\mathbb{E}[q^\top k]$, $\mathrm{Var}(q^\top k)$, and $\mathrm{Var}(q^\top k/\sqrt{d_k})$. Explain how this supports scaled dot-product attention.

10.15 Homework: Transformer Algebra

Required Work

This homework uses SymPy [145] to verify the chapter's matrix shapes and masking algebra.

Submission model. Submit the completed starter script, the main hand derivation, the SymPy output needed to verify it, and a short interpretation. External computer-algebra tools may be used only as optional sanity checks.

Task 1. Define symbolic dimensions T, d_{model}, d_k, and d_v, and create symbolic matrix expressions for X, W_Q, W_K, and W_V.

Task 2. Use SymPy matrix expressions to form $Q = XW_Q$, $K = XW_K$, $V = XW_V$, $S = QK^\top$, and $H = AV$, where A is a symbolic $T \times T$ attention matrix.

Task 3. Print the shapes of Q, K, V, S, A, and H, and verify that they match the chapter formulas.

Task 4. Build a three-token numerical example for `not very good` using the small matrices supplied in the starter file.

Task 5. Compute the scaled score matrix $S = QK^\top/\sqrt{2}$.

Task 6. Use the provided row-wise softmax helper to compute the unmasked attention matrix.

Task 7. Verify that each row of the unmasked attention matrix sums to 1.

Task 8. Add a causal mask to the three-token score matrix and compute the masked attention matrix.

Task 9. Verify that future positions have attention weight 0 after masking.

Task 10. Explain in words how the masked row for `very` differs from the unmasked row and why that difference matters.

Reference Values for Checking

The symbolic shape checks should report

$$Q, K \in \mathbb{R}^{T\times d_k}, \qquad V \in \mathbb{R}^{T\times d_v}, \qquad S, A \in \mathbb{R}^{T\times T}, \qquad H \in \mathbb{R}^{T\times d_v}.$$

For the causal mask in the three-token example, the first row may attend only to the first token, the second row may attend only to the first two tokens, and the third row may attend to all three tokens. The masked attention entries above the diagonal should simplify to 0.