



Data Analysis Block Course: Statistics and Machine learning



Transformers by Hand

Prof. Dr. Mikhail Mikhasenko

Assistants: Hendric Jonas, Christoph Tönnis

30 September 2026

Indico schedule

RUB Moodle

Learning goals

After this exercise, you should be able to

- explain why tokens need an embedding lookup before entering a network,
- explain why attention alone is permutation equivariant and why order must be added back in,
- compute scaled dot-product attention by hand for a small example,
- track tensor shapes through query/key/value projections, multi-head splitting and recombination,
- explain the difference between an encoder block and a decoder block,
- construct a causal attention mask,
- perform a full dimensionality and parameter-count analysis of a transformer encoder layer,
- match encoder-only, decoder-only and encoder-decoder architectures to typical tasks.

The calculations in this sheet are designed to be done by hand.

Part I - Embeddings

We work with a tiny vocabulary of six words:

$$\{\text{"the"}, \text{"cat"}, \text{"sat"}, \text{"on"}, \text{"mat"}, \text{"dog"}\}$$

so the vocabulary size is $V = 6$. Each word is represented by an integer index $0, \dots, 5$ (in the order above).

Exercise 1.1 - The embedding matrix

An embedding layer stores one trainable vector of length $d = 4$ per vocabulary entry, collected into a matrix

$$E \in \mathbb{R}^{V \times d}$$

1. What are the dimensions of E for our vocabulary?

2. How many trainable parameters does this embedding layer have?
3. Looking up the embedding for word index i means selecting row i of E . Why is this equivalent to multiplying a one-hot vector of length V with E ?
4. Why do we use a lookup instead of feeding the raw integer index directly into a dense layer?

Exercise 1.2 - Embedding a sentence

We embed the sentence "the cat sat on the mat" (6 tokens, with "the" appearing twice) using the following rows of E :

index	word	embedding row
0	the	(1, 0, 0, 1)
1	cat	(0, 2, 0, 0)
2	sat	(0, 0, 2, 0)
3	on	(1, 1, 0, 0)
4	mat	(0, 1, 1, 0)
5	dog	(2, 0, 0, 1)

1. Write out the resulting embedded sequence as a matrix X , one row per token in sentence order.
2. What are the dimensions of X ? Express them as (sequence length L) $\times$ (embedding dimension d).
3. If we embed a **batch** of 8 such sentences (all padded to length $L = 6$), what are the dimensions of the batched tensor?
4. The word "the" appears twice in the sentence. Are its two embedding vectors identical after the lookup? Explain why this alone is not enough to distinguish the two occurrences.

Exercise 1.3 - Why attention needs position information

Self-attention (Part II) computes, for every token, a weighted average of *value* vectors, where the weights come from comparing *query* and *key* vectors.

1. Suppose we shuffle the tokens of a sentence (and their padding mask) together, in the same order. Argue informally that the set of pairwise query-key comparisons is unchanged, just relabelled.
2. Based on 1., is plain self-attention sensitive to the order of the input tokens?
3. Give one concrete example of a sentence pair where the two sentences contain the same words but the order changes the meaning.
4. Positional encodings add a vector that depends only on the position $t = 0, 1, 2, \dots$ (not on the token identity) to each embedding before it enters the encoder. Why does this restore order-sensitivity?

Part II - Scaled dot-product attention by hand

For one attention head, with queries $Q \in \mathbb{R}^{L \times d_k}$, keys $K \in \mathbb{R}^{L \times d_k}$ and values $V \in \mathbb{R}^{L \times d_v}$,

$$A = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) \quad Z = AV$$

Softmax is taken **row-wise**, i.e. separately for each query, over the key dimension.

Exercise 2.1 - A three-token example

Consider the three tokens "the", "cat", "sat" with $d_k = 4$ key/query vectors and $d_v = 2$ value vectors:

token	key k_i	value v_i
the	(2, 0, 0, 0)	(1, 0)
cat	(0, 2, 0, 0)	(0, 1)
sat	(0, 0, 2, 0)	(1, 1)

We compute the output for the query of token "cat", $q_{\text{cat}} = (0, 2, 0, 0)$.

Step 1 - Raw scores

Calculate the three dot products

$$s_i = q_{\text{cat}} \cdot k_i \quad i \in \{\text{the}, \text{cat}, \text{sat}\}$$

Your answers:

$$s_{\text{the}} = \underline{\quad} \quad s_{\text{cat}} = \underline{\quad} \quad s_{\text{sat}} = \underline{\quad}$$

Step 2 - Scale

Divide each score by $\sqrt{d_k} = \sqrt{4} = 2$.

$$\tilde{s}_{\text{the}} = \underline{\quad} \quad \tilde{s}_{\text{cat}} = \underline{\quad} \quad \tilde{s}_{\text{sat}} = \underline{\quad}$$

Step 3 - Softmax

Use $e^0 = 1$ and $e^2 \approx 7.39$. Compute

$$a_i = \frac{e^{\tilde{s}_i}}{\sum_j e^{\tilde{s}_j}}$$

Complete the table (round to 3 decimal places):

i	$\tilde{s}_i$	$e^{\tilde{s}_i}$	a_i
the			
cat			
sat			

Check that your three weights sum to 1.

Step 4 - Weighted output

Calculate

$$z_{\text{cat}} = a_{\text{the}} v_{\text{the}} + a_{\text{cat}} v_{\text{cat}} + a_{\text{sat}} v_{\text{sat}}$$

Your answer:

$$z_{\text{cat}} \approx \underline{\hspace{2cm}}$$

Step 5 - Interpretation

1. Which token does "cat" attend to most strongly? Relate this to the dot product $q_{\text{cat}} \cdot k_{\text{cat}}$.
2. Would the result change if we scaled every key vector by the same positive constant c before the scaled dot product? Would it change the *weights* a_i ?
3. Why do we divide by $\sqrt{d_k}$ rather than, say, d_k or not scaling at all? (Think about what happens to the scores as d_k grows, if q and k have entries of typical size $\mathcal{O}(1)$.)

Exercise 2.2 - Shapes of scaled dot-product attention

For a batch of B examples, sequence length L , key dimension d_k and value dimension d_v :

Complete the table with tensor shapes:

quantity	shape
Q	
K	
V	
$QK^\top$	
$A = \text{softmax}(QK^\top / \sqrt{d_k})$	
$Z = AV$	

1. Does the shape of Z depend on d_k ?
2. In self-attention, Q , K and V are all linear projections of the same input $X \in \mathbb{R}^{B \times L \times d_{\text{model}}}$. What operation produces Q from X , and what is the shape of the weight matrix that does it?

Exercise 2.3 - Multi-head attention dimensionality

A multi-head attention layer with h heads and model dimension d_{model} splits Q , K , V into h smaller pieces, each of dimension

$$d_k = d_v = \frac{d_{\text{model}}}{h}$$

runs scaled dot-product attention independently per head, concatenates the h outputs, and applies one more linear projection back to d_{model} .

Take $d_{\text{model}} = 8$, $h = 2$, $L = 5$, $B = 1$.

1. What is d_k per head?
2. What is the shape of Q , K , V **per head**?
3. What is the shape of the attention output Z_{head} per head?

4. What is the shape after concatenating all h heads?
5. What is the shape after the final output projection?
6. Why must d_{model} be divisible by h ?
7. If we increased h from 2 to 4 while keeping $d_{\text{model}} = 8$ fixed, what happens to d_k per head? Does the total number of attended values change?

Part III - Encoder and decoder blocks

Exercise 3.1 - Labelling a transformer encoder layer

A transformer **encoder layer** consists of, in order:

1. Multi-head self-attention over the input sequence,
2. a residual (skip) connection adding the block's input, followed by layer normalisation,
3. a position-wise feed-forward network (two linear layers with a nonlinearity in between),
4. another residual connection followed by layer normalisation.

Draw this as a block diagram with boxes and arrows, labelling the input X , the intermediate tensors, and the two "Add & Norm" steps.

1. What does "residual connection" mean here, concretely, in terms of a tensor addition?
2. Why is a residual connection useful for training deep stacks of these layers (connect this to what you learned about backpropagation through many layers)?
3. Self-attention in an encoder is **bidirectional**: a token can attend to tokens both before and after it. Why is this appropriate for, e.g., understanding the meaning of a sentence, but not appropriate for generating text left to right?

Exercise 3.2 - A decoder layer and causal masking

A transformer **decoder layer** (as used for language generation) additionally uses a **causal mask**: token i is only allowed to attend to tokens $j \leq i$.

For a sequence of $L = 4$ tokens, complete the mask table. Use **True** to mean "query i may attend to key j ", **False** otherwise:

query i \ key j	$j = 0$	$j = 1$	$j = 2$	$j = 3$
$i = 0$				
$i = 1$				
$i = 2$				
$i = 3$				

1. In practice, the mask is implemented by setting the disallowed scores to $-\infty$ before the softmax, rather than by zeroing weights afterwards. Why does this matter (think about what softmax does to a row of otherwise-unmasked scores if some scores are simply removed vs. set to $-\infty$)?
2. Explain in one or two sentences why an *encoder-decoder* architecture (e.g. for translation) additionally uses **cross-attention**, where the decoder's queries attend to the encoder's keys and values, without any masking on that step.
3. An encoder-only model (like BERT) is typically used for classification or understanding tasks. A decoder-only model (like GPT) is typically used for text generation. Which one uses the causal mask, and why does that follow from how it is trained/used?

Exercise 3.3 - Full dimensionality and parameter count

Consider one encoder layer with:

$$d_{\text{model}} = 512 \quad h = 8 \quad d_{\text{ff}} = 2048$$

The multi-head attention sub-layer has four weight matrices of shape $(d_{\text{model}}, d_{\text{model}})$: for the query, key, value and output projections (ignore biases for this exercise).

The feed-forward sub-layer has two weight matrices, of shape $(d_{\text{model}}, d_{\text{ff}})$ and $(d_{\text{ff}}, d_{\text{model}})$ respectively (again ignore biases).

1. Calculate the number of parameters in the four attention projection matrices combined.
2. Calculate the number of parameters in the two feed-forward matrices combined.
3. Calculate the total number of weight parameters in one encoder layer.
4. A full model stacks $N = 6$ identical encoder layers. Estimate the total number of weight parameters (ignore the embedding matrix and any output head).
5. For an input batch with $B = 32$, sequence length $L = 50$: give the shape of the tensor at (a) the embedding output, (b) after the multi-head attention sub-layer, (c) inside the feed-forward network's hidden representation, (d) at the output of the encoder layer.
6. Does the number of parameters in an encoder layer depend on the sequence length L ? Does the amount of *computation* (and memory for the attention matrix A) depend on L ? Explain the difference.

Exercise 3.4 - Matching architectures to tasks

Match each task to the most natural architecture choice: **encoder-only**, **decoder-only**, or **encoder-decoder**. Justify each in one sentence.

task	architecture	justification
Sentiment classification of a movie review		
Autoregressive next-word text generation		
Machine translation (English $\rightarrow$ German)		
Producing a single embedding vector to search a document database		

Short conceptual questions

1. Why is self-attention, on its own, permutation equivariant, and what two mechanisms fix that in a real transformer (one for the encoder input, one for the decoder's generation order)?
2. Why do queries, keys and values need separate learned projections rather than using the raw input embeddings directly for all three?
3. If the embedding dimension d_{model} is increased while keeping the number of heads h fixed, what happens to d_k per head, and what does that imply for the granularity of what each head can represent?
4. Why does layer normalisation typically operate over the feature dimension d_{model} rather than over the batch dimension, unlike batch normalisation?
5. Explain, without equations, what would go wrong if a decoder used bidirectional (unmasked) self-attention during training for next-token prediction.
6. The feed-forward sub-layer of an encoder layer is applied identically and independently to every token position. What is doing the "mixing" of information *between* token positions in a transformer, if not the feed-forward network?