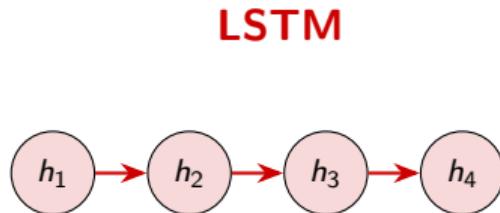


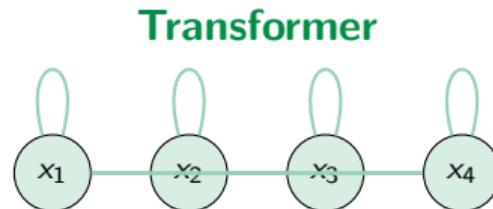
The Transformer Architecture

Self-Attention · Multi-Head Attention · Positional Encoding · Encoder–Decoder

The key idea



Sequential: $O(n)$ path



Parallel: $O(1)$ path

"Attention Is All You Need" (Vaswani et al., 2017)

Parallel

Process all tokens simultaneously

Direct access

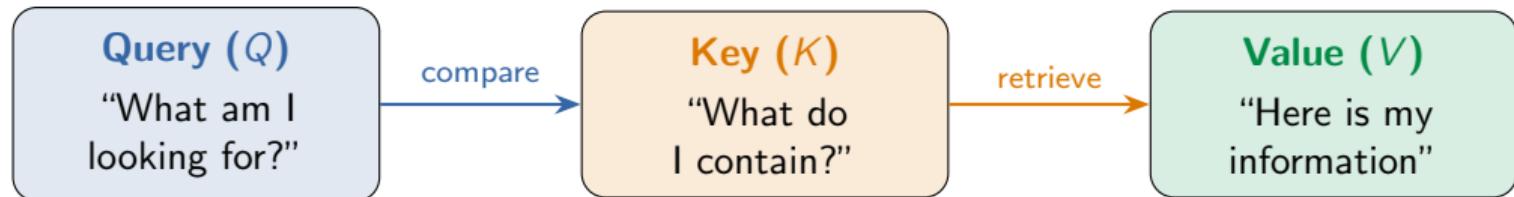
Any token can attend to any other

Contextual

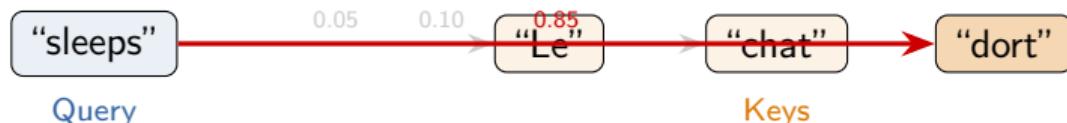
Representations change with context

Attention — intuition

Think of it as a soft dictionary lookup



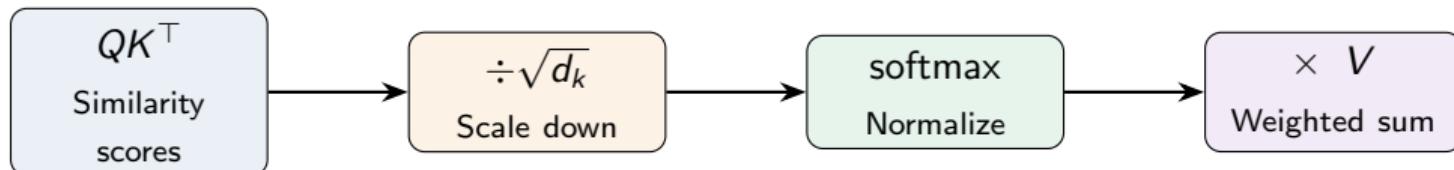
Example: translating “Le chat dort” → “The cat sleeps”



Output = weighted sum of values: $0.05 \cdot v_{\text{Le}} + 0.10 \cdot v_{\text{chat}} + 0.85 \cdot v_{\text{dort}}$

Scaled dot-product attention

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$



Why $\sqrt{d_k}$?

Without scaling, dot products grow with d_k , pushing softmax into saturation (near 0 or 1) where gradients vanish

Dimensions:

$Q: (n \times d_k)$ $K: (m \times d_k)$
 $V: (m \times d_v)$
 $QK^\top: (n \times m)$ Output: $(n \times d_v)$

Each output row is a **weighted average** of value vectors, where weights come from query-key similarity

Attention — worked example

Tokens: The cat sat

	$QK^\top / \sqrt{d_k}$			Weights			
	The	cat	sat				
The	1.2	0.5	0.3	.49	.28	.23	
cat	0.4	2.1	0.8	softmax →	.12	.63	.25
sat	0.2	0.9	1.5		.15	.30	.55

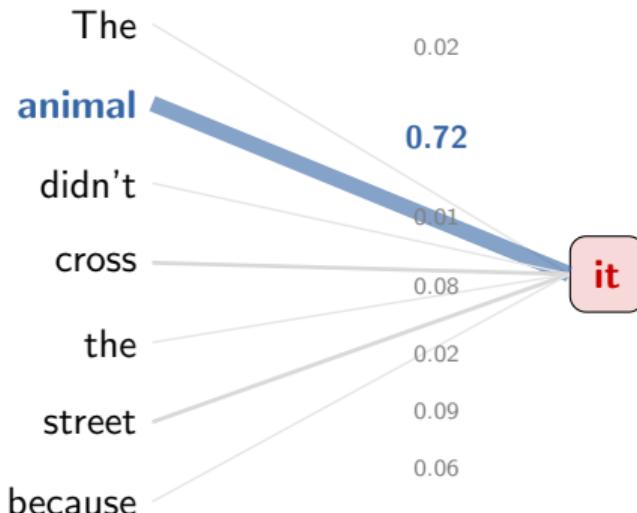
Row for “cat”: attends 63% to itself, 25% to “sat”, 12% to “The”

Output for “cat” =
 $0.12 \cdot v_{\text{The}} + 0.63 \cdot v_{\text{cat}} + 0.25 \cdot v_{\text{sat}}$

Each output is a **context-aware** representation — unlike Word2Vec, the same word gets different embeddings in different contexts

Attention learns meaningful relationships

"The **animal** didn't cross the street because **it** was too tired"



Coreference resolved!

"it" attends most strongly to "animal" — the model learned that "it" refers to "animal"

No explicit rule

This emerges from training — the model learns which tokens are relevant for each query

Different attention heads specialize in different relationships (see next slides)

Self-attention vs. cross-attention

Self-Attention

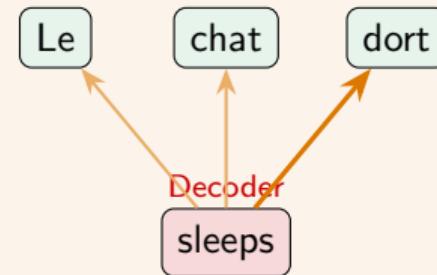


Q, K, V all come from
the **same** sequence

Used in: encoder, decoder (masked)

Cross-Attention

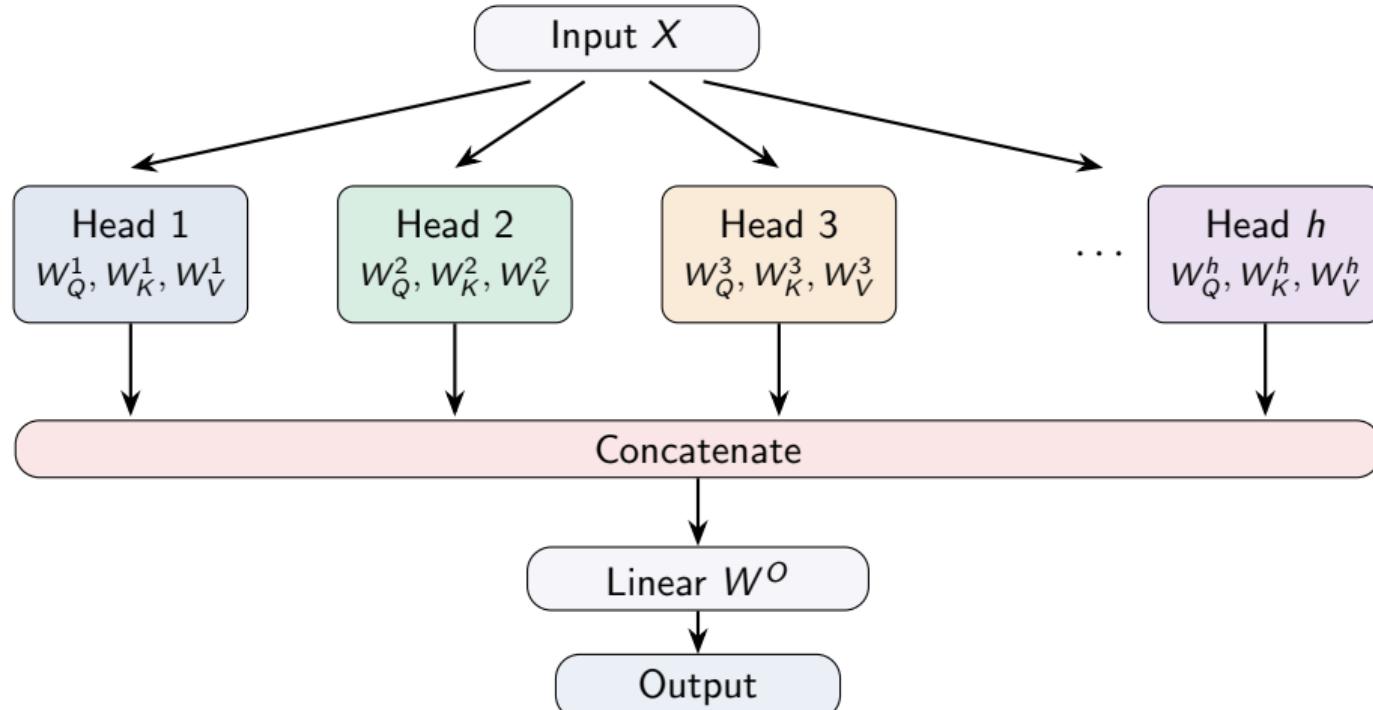
Encoder



Q from decoder
 K, V from encoder

Used in: decoder (enc-dec models)

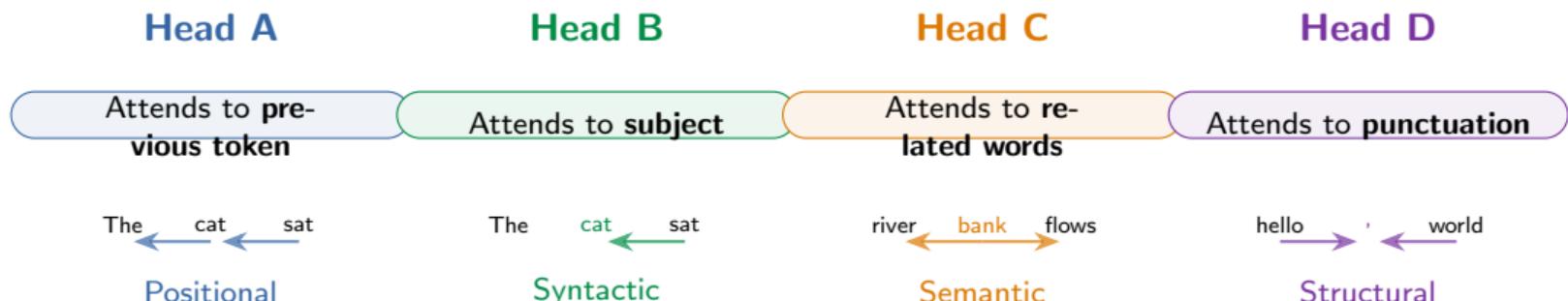
Multi-head attention



$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O$
where $\text{head}_i = \text{Attention}(QW_Q^i, KW_K^i, VW_V^i)$

What different heads learn

Different heads specialize in different types of relationships:

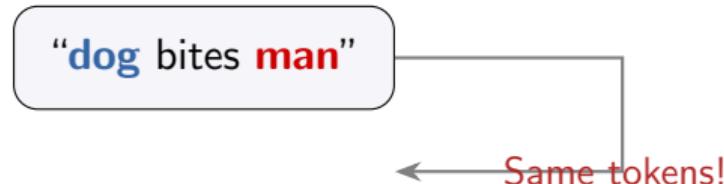


Multiple heads = multiple “perspectives” on the same input.

Typical: $h = 8$ (BERT-base) or $h = 12\text{--}96$ (larger models). $d_k = d_{\text{model}}/h$

Total compute is the same as single-head attention with full d_{model} , since each head uses $d_k = d_{\text{model}}/h$

Positional encoding — why we need it



Problem: Self-attention computes dot products between token pairs.

It's **permutation-invariant** — swapping token order doesn't change the output!

But word order matters: these two sentences mean very different things.

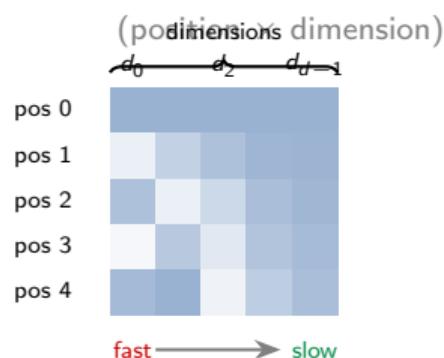
Solution: Add **positional information** to the input embeddings

$$\text{input}_i = \text{token_embedding}_i + \text{position_encoding}_i;$$

Sinusoidal positional encoding

$$\begin{aligned} \text{PE}(\text{pos}, 2i) &= \sin\left(\frac{\text{pos}}{10000^{2i/d}}\right) \\ \text{PE}(\text{pos}, 2i + 1) &= \cos\left(\frac{\text{pos}}{10000^{2i/d}}\right) \end{aligned}$$

PE matrix



Unique pattern

Each position gets a distinct encoding

No learned params

Fixed, deterministic, works for any length

Relative positions

$\text{PE}_{\text{pos}+k}$ can be expressed as linear function of PE_{pos}

Multi-scale

Low dims = fine position, high dims = coarse

$$\text{input}_i = \text{embedding}(x_i) + \text{PE}(i) \quad (\text{element-wise addition})$$

Modern positional encodings

Sinusoidal

Vaswani et al., 2017

Added to embeddings

Absolute position

Fixed (not learned)

Used by: original Transformer

RoPE

Su et al., 2021

Rotates Q and K vectors

Relative position

No extra parameters

Used by: LLaMA, Mistral

ALiBi

Press et al., 2022

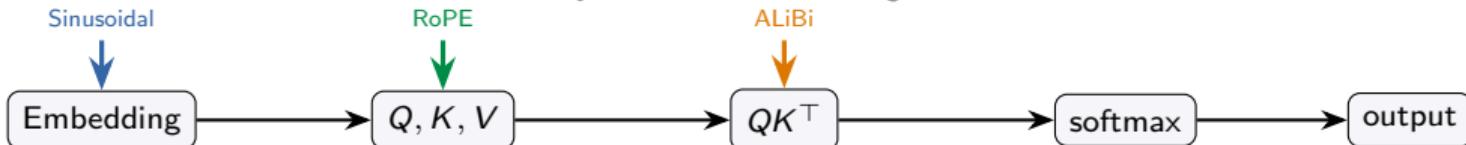
Bias in attention scores

Linear distance penalty

No extra parameters

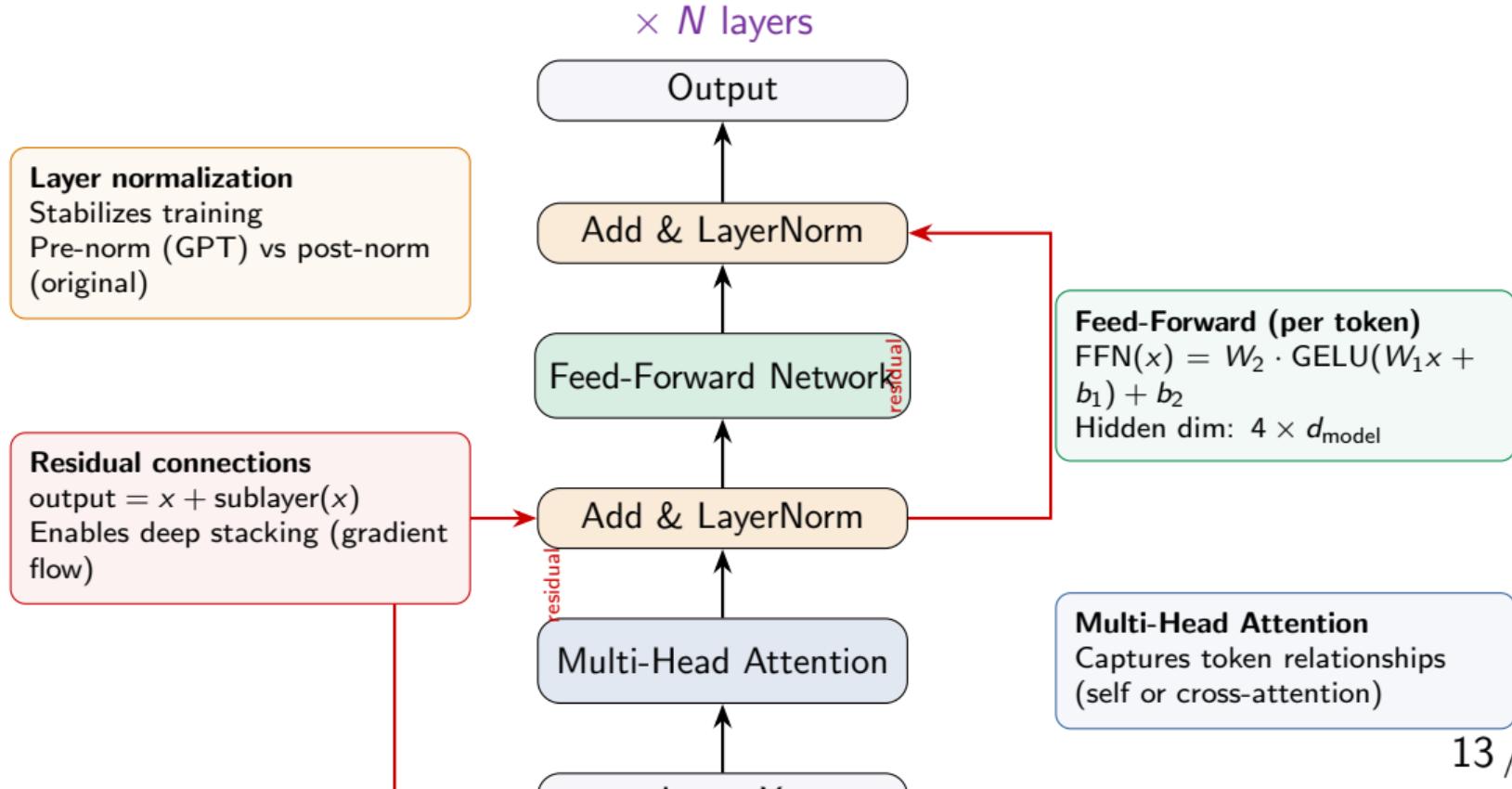
Used by: BLOOM, MPT

Where position info is injected:

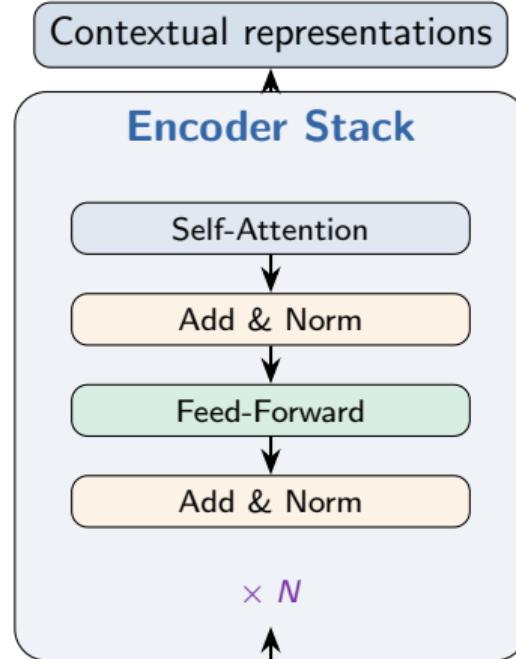
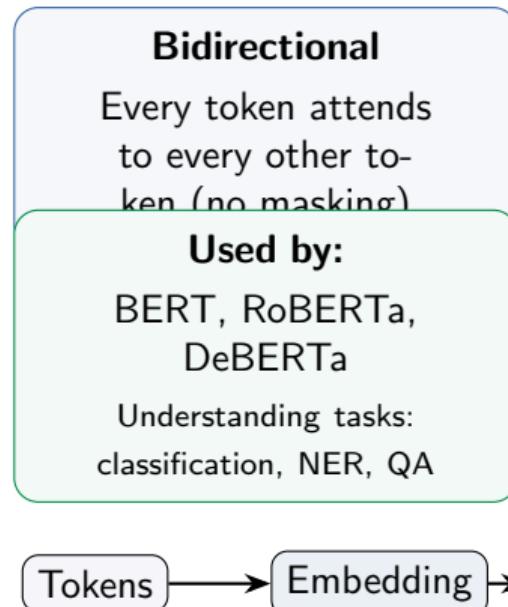


RoPE is the most popular today — it naturally encodes relative position and supports length extrapolation with techniques like YaRN

The Transformer block

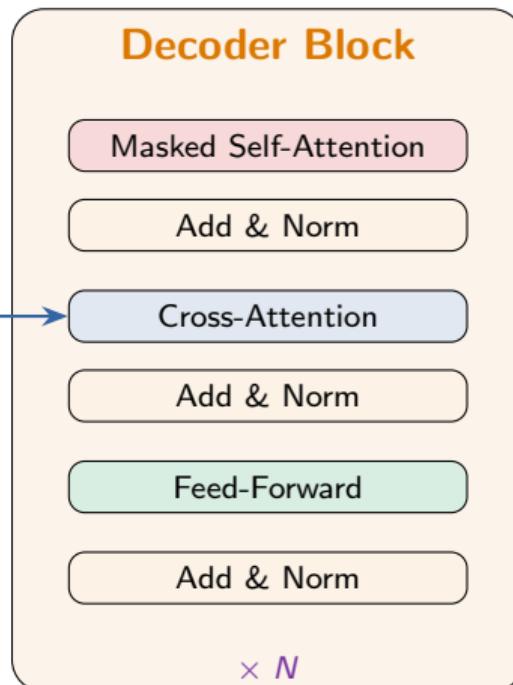


The Transformer encoder



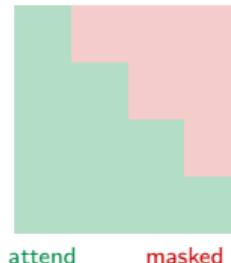
The Transformer decoder

From encoder
(K, V)



Decoder-only models (GPT) skip cross-attention — only masked self-attention + FFN

Causal Mask

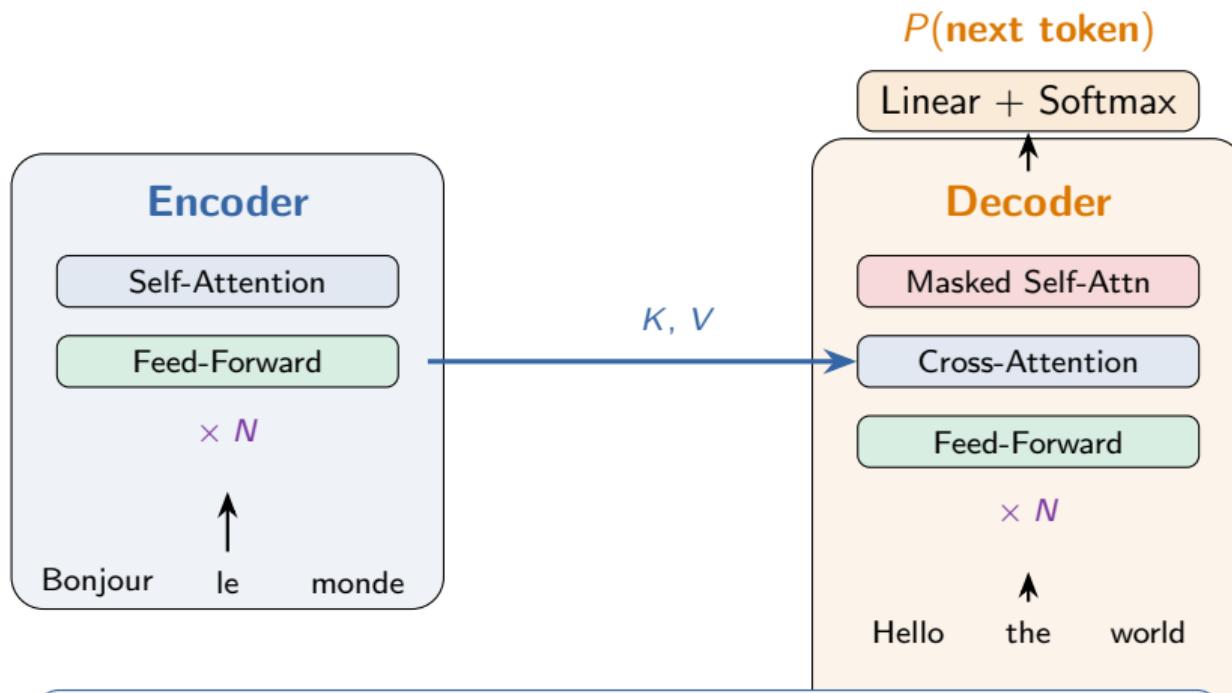


Token t can only attend to tokens $< t$

Used by:

GPT (decoder-only)
T5 decoder
Generation tasks

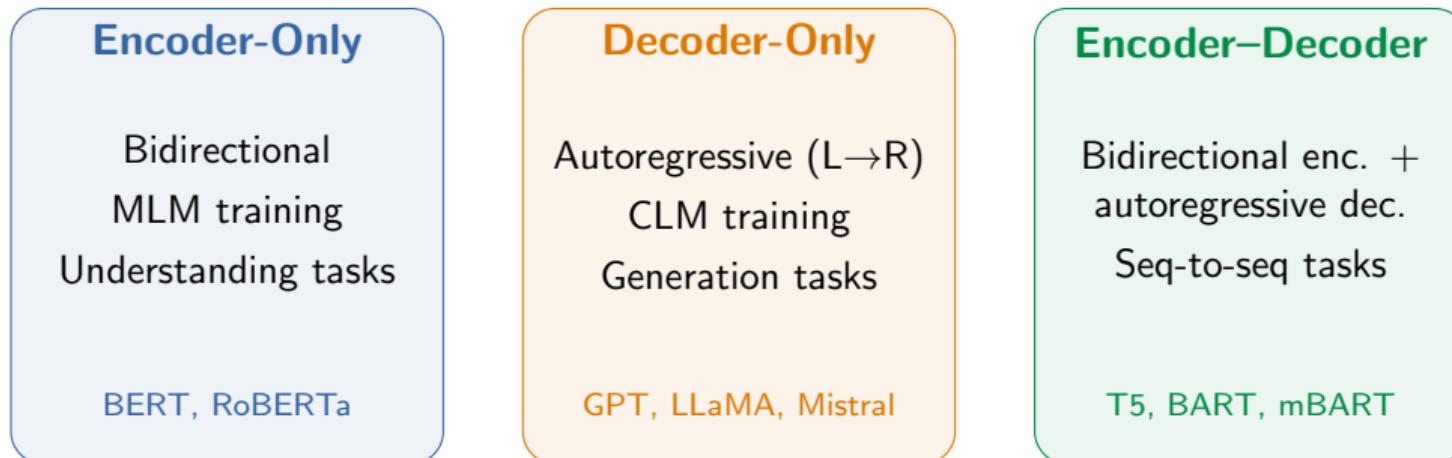
Encoder–Decoder architecture



Used for: machine translation, summarization, question answering

Models: T5, BART, mBART, original Transformer

Three Transformer architectures

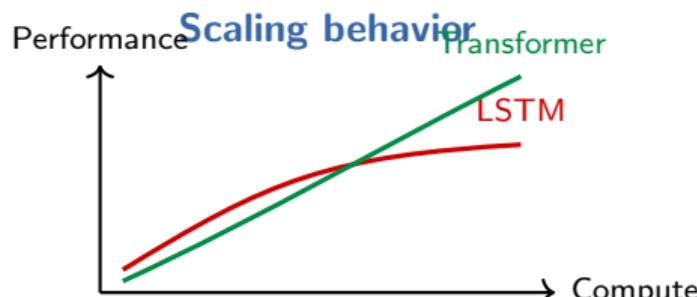


	Enc-Only	Dec-Only	Enc-Dec
Direction	Bidirectional	Left-to-right	Both
Best for	Classification, NER	Text generation	Translation, summary
Today's trend	Less common	Dominant	Niche

The trend since GPT-3 (2020): decoder-only models at scale can do almost everything, including understanding tasks.

Why Transformers won

	LSTM	Transformer
Parallelizable	No (h_t needs h_{t-1})	Yes (all tokens at once)
Long-range path	$O(n)$ steps	$O(1)$ (direct attention)
Context	Limited by hidden size	Full context window
Scaling	Diminishing returns	Log-linear improvement
Training speed	Slow (sequential)	Fast (GPU-friendly)



The scaling insight:
Transformers reliably improve with
more data, parameters, and compute

This enabled the LLM revolution:
GPT-3, PaLM, LLaMA, Claude, ...

Transformer dimensions in practice

Model	Layers	d_{model}	Heads	d_{ff}	Params
BERT-base	12	768	12	3072	110M
BERT-large	24	1024	16	4096	340M
GPT-2	12	768	12	3072	117M
GPT-3	96	12288	96	49152	175B
LLaMA-2 7B	32	4096	32	11008	7B
LLaMA-2 70B	80	8192	64	28672	70B

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

Head dimension

$$d_{\text{ff}} \approx 4 \times d_{\text{model}}$$

FFN hidden size

$$\text{Params} \approx 12 \cdot N \cdot d^2$$

Rough estimate

Questions?

Next: Early Notable Models — GPT, BERT, T5