# Machine Learning and Optimization

## Lecture 18: The Transformer Architecture

CSCI 4140/6960 — Spring 2026

Course website: `mlopt-s26.github.io`

# Learning Objectives

By the end of this lecture, you should be able to:

1. **Derive** the transformer block as a solution to a stated design problem: contextual representations via alternating communication (attention) and computation (FFN).
2. **Derive** scaled dot-product self-attention from the per-token definition to the matrix form, including the $1/\sqrt{d_k}$ scaling as a forced variance fix.
3. **Explain** why causal masking is *necessary* for autoregressive training, and derive the mask from the training objective.
4. **Distinguish** training-time parallelism (teacher forcing + causal mask) from inference-time sequentiality (autoregressive decoding), and explain exposure bias.
5. **Classify** transformer architectures by communication structure (bidirectional / causal / cross-sequence) and pretraining objective.

# Learning Objectives (continued)

6. **State** that self-attention is permutation equivariant and explain why positional encoding is a design requirement.
7. **Demonstrate** that a single attention head is a bottleneck and define MHSA as the resolution.
8. **Describe** the encoder transformer block and distinguish pre-norm from post-norm.
9. **Explain** why the FFN sublayer is necessary: attention alone cannot create features not linearly present in the input embeddings.
10. **Define** cross-attention and connect it to Bahdanau attention.
11. **Predict** the consequence of removing each major component from the architecture.

# Lecture Roadmap


The Pressure

The Design Problem and the Proto-Block

Designing the Communication Primitive

Richer Communication — Multi-Head Attention

The Computation Phase and the Full Block

Stress-Testing — What If We Broke It?

The Decoder Transformer

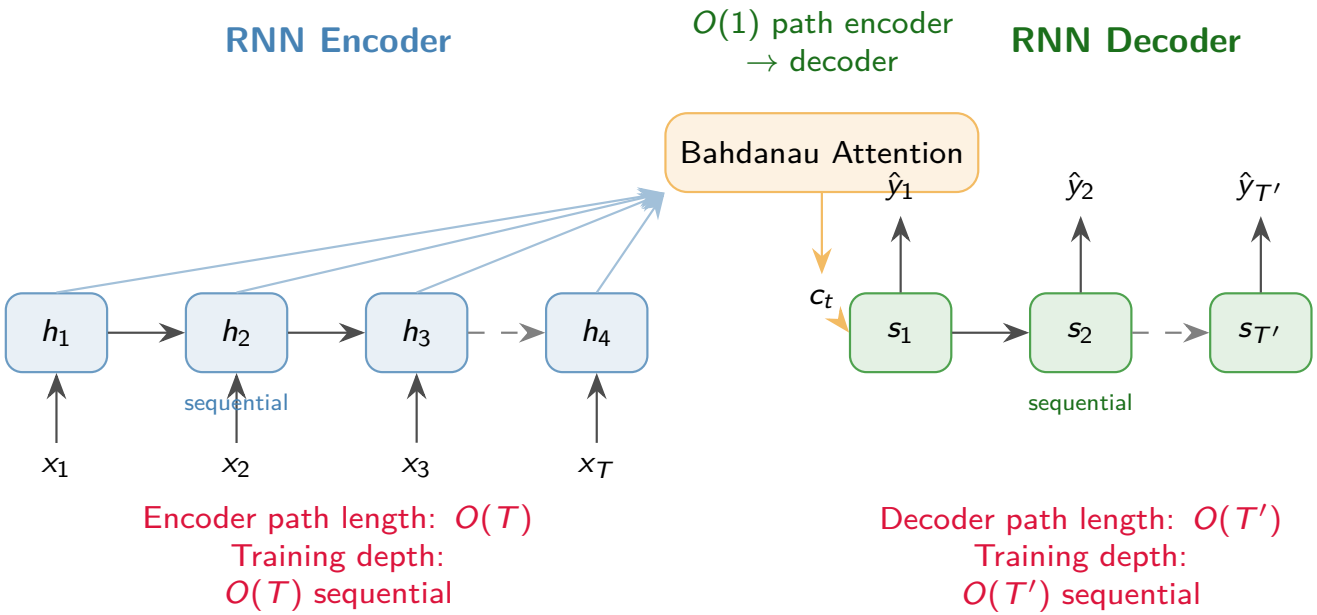
Training and Generating

Architecture Taxonomy — Three Communication Structures

Why Transformers Win

Summary

# Lecture 17 Recap — The Encoder–Decoder with Attention




Bahdanau attention solved the encoder-to-decoder bottleneck. But both encoder and decoder still have $O(T)$ sequential training depth.

**The question:** Can we eliminate recurrence entirely?

# What Are We Trying to Compute?

## Design Problem

Build a sequence-to-sequence system with two components:

1. **Encoder:** Produce contextual representations $c_1, \ldots, c_T$ of the input sequence *in parallel* — no sequential bottleneck.
2. **Decoder:** Reference the encoder representations to perform left-to-right autoregressive generation, also *without* sequential bottleneck at training time.

Everything we build today solves this problem. If a component doesn't serve one of these two goals, it doesn't belong.

# The Two-Phase Skeleton

Two operations are needed — and they are *different*:

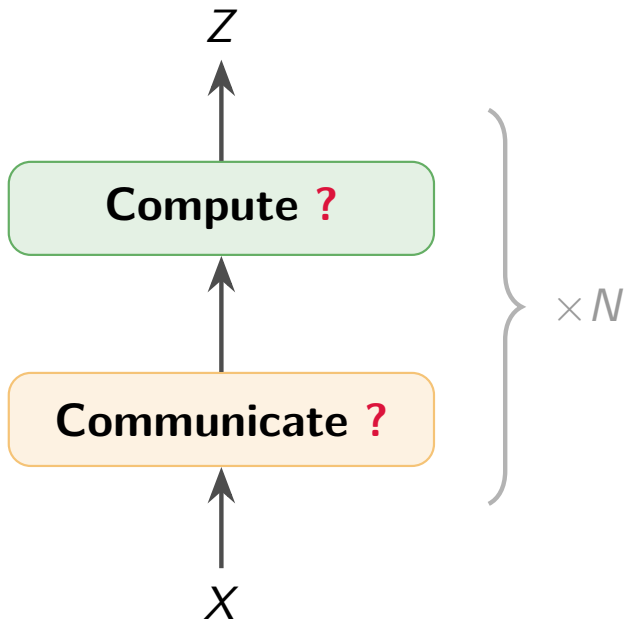
| Phase | What it does | Why it's needed |
|---|---|---|
| **Communication** | Tokens exchange information | Contextual: must see other tokens |
| **Computation** | Each token processes independently | Without this, tokens only copy — no transformation |

**Proto-block:**

$$X \xrightarrow{\text{communicate}} X' \xrightarrow{\text{compute}} Z$$

Stack $N$ copies: communicate, compute, communicate, compute, ...

# The Proto-Block



# From Cross-Attention to Self-Attention

Recall from last lecture: attention with encoder states $h_1, \ldots, h_T$ and decoder state $s_t$:

$$\alpha_t = \text{softmax}\big([k_1^\top q_t,\ \ldots,\ k_T^\top q_t]\big), \qquad c_t = \sum_{i=1}^{T} (\alpha_t)_i\, v_i$$

**Key conceptual leap:** What if a single sequence attends to *itself*?

**Self-attention** — keys, queries, and values all from the same sequence:

$$k_i = W_K x_i, \qquad q_i = W_Q x_i, \qquad v_i = W_V x_i$$

Does this satisfy the parallelism goal? Yes — every token's QKV computation is independent. The only coupling is through the softmax weights.

# Self-Attention — Per-Token Definition

For token $i$:

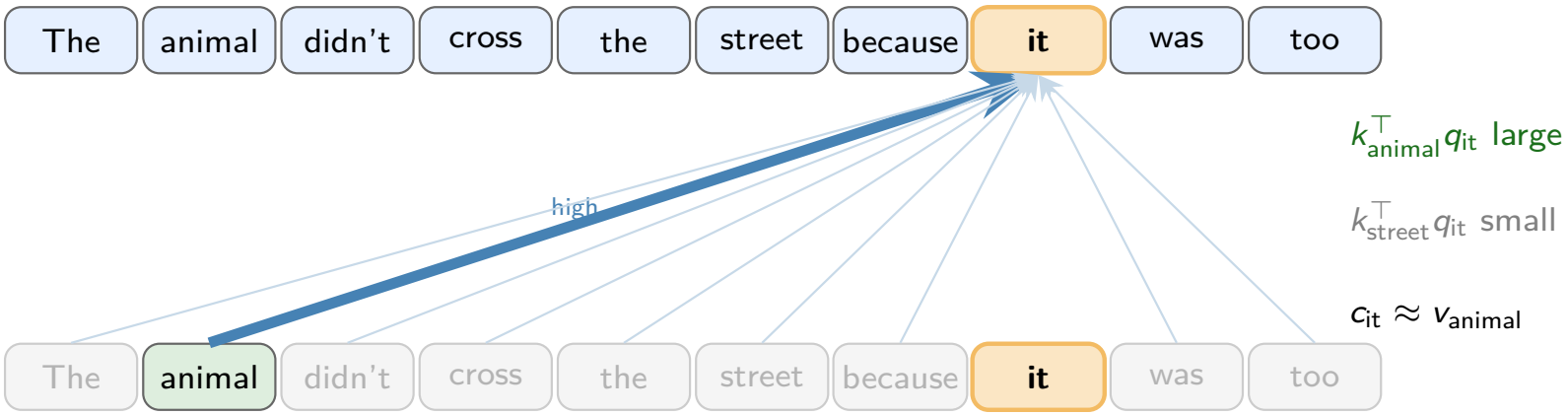
$$\alpha_i = \text{softmax}\left( \begin{bmatrix} k_1^\top \\ \vdots \\ k_T^\top \end{bmatrix} q_i \right) \in \mathbb{R}^T, \qquad c_i = \sum_{j=1}^{T} (\alpha_i)_j \, v_j$$

**What the parameters learn:**

- $W_Q$ learns what each token *looks for*
- $W_K$ learns what each token *advertises*
- $W_V$ learns what each token *passes along* when attended to

# Self-Attention — Pronoun Resolution

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




Token "it" produces query $q_{\text{it}}$. Self-attention resolves the pronoun: $c_{\text{it}}$ is dominated by $v_{\text{animal}}$.

# Design Check — Does This Know Word Order?

We designed a communication primitive. Before continuing: does it know that "animal" comes before "it"?

**Claim (Permutation Equivariance)**

Self-attention is permutation equivariant. If $\Pi$ is a permutation matrix and $X' = X\Pi$, then $\text{Attention}(X') = \text{Attention}(X)\Pi$.

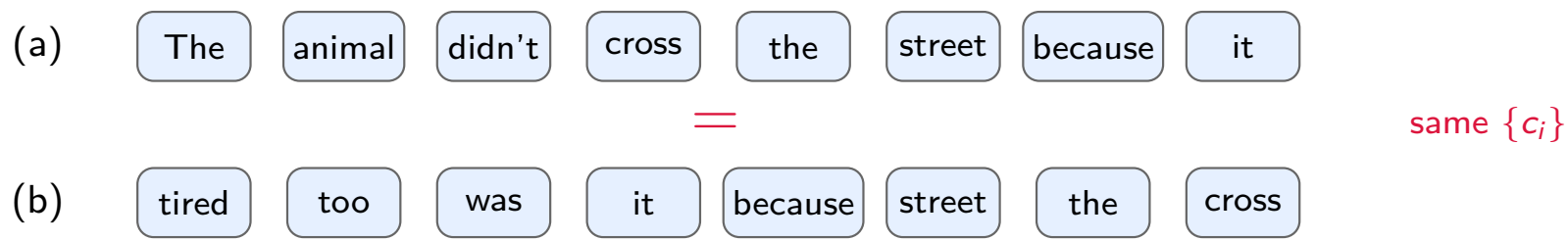
*Proof sketch.* $K' = W_K X\Pi$, $Q' = W_Q X\Pi$, $V' = W_V X\Pi$. Then $(K')^\top Q' = \Pi^\top K^\top Q\, \Pi$, and softmax applied column-wise commutes with symmetric permutation of rows and columns. $\square$

# Permutation Equivariance Breaks Pronoun Resolution

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

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

Same contextual representations (up to reordering). The model cannot distinguish these.




**Design failure.** "The dog bit the man" and "The man bit the dog" are indistinguishable. We must fix this before proceeding.

# Design Fix — Positional Encodings

**Fix:** Inject position into token representations *before* self-attention:

$$\tilde{X} = X + P = [\, x_1 + p_1 \mid \cdots \mid x_T + p_T \,]$$

This is a **design requirement**, not a patch. Any permutation-equivariant communication primitive needs external positional information.

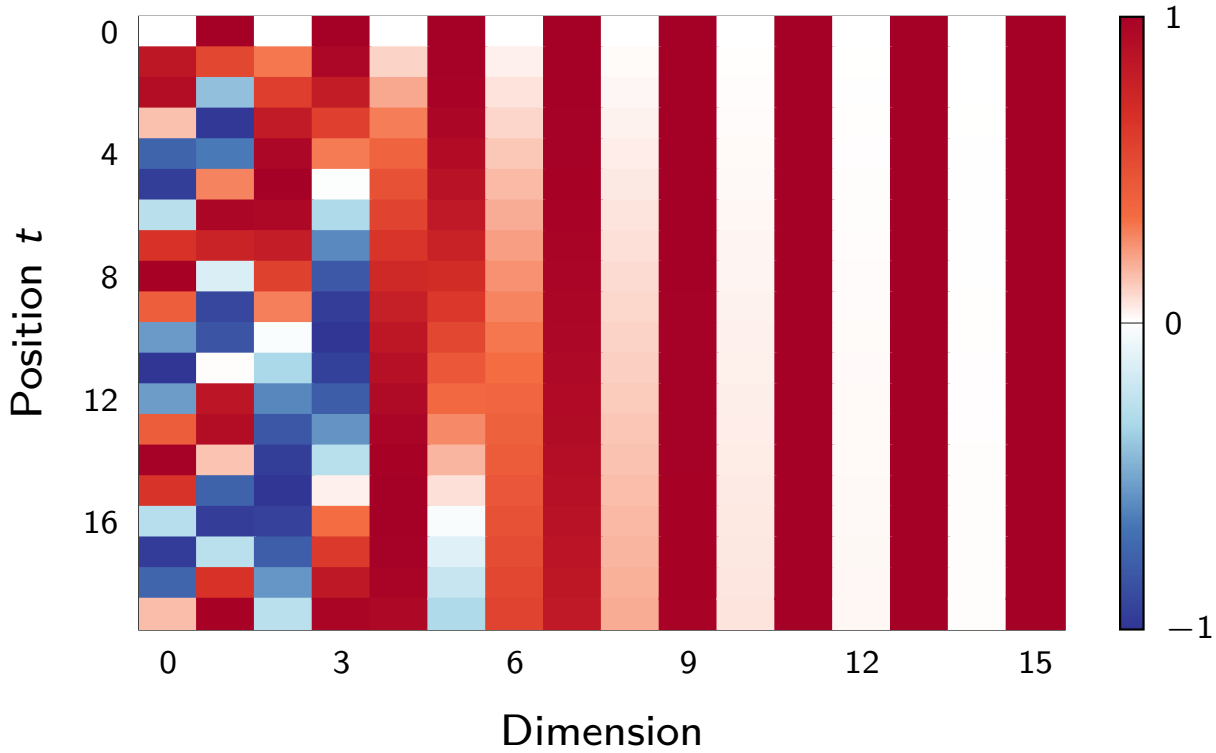
**Option 1 — Sinusoidal** (Vaswani et al., 2017):

$$(p_t)_{2i-1} = \sin\!\left(t \,/\, 10000^{2i/d_x}\right), \qquad (p_t)_{2i} = \cos\!\left(t \,/\, 10000^{2i/d_x}\right)$$

Different frequencies per dimension. Property: $p_{t+\Delta}$ is a linear (rotation) function of $p_t$.

**Option 2 — Learned** (BERT, GPT-2): $P$ is a learnable parameter. More flexible; cannot generalize beyond training length $T$.

# Positional Encoding — Intuition




Each position gets a unique signature. Nearby positions have similar encodings (high-frequency dimensions agree). The encoding is deterministic — no parameters to learn.

# Self-Attention — Matrix Form

With position-aware inputs $\tilde{X} = [\tilde{x}_1 \mid \cdots \mid \tilde{x}_T] \in \mathbb{R}^{d_x \times T}$:

$$K = W_K \tilde{X} \in \mathbb{R}^{d_k \times T}, \qquad Q = W_Q \tilde{X} \in \mathbb{R}^{d_k \times T},$$
$$V = W_V \tilde{X} \in \mathbb{R}^{d_v \times T}$$

The **score matrix** $K^\top Q \in \mathbb{R}^{T \times T}$:

$(K^\top Q)_{ji} = k_j^\top q_i$ ("how much should token $i$ attend to token $j$?")

Column $i$ of softmax$(K^\top Q)$ is $\alpha_i$.

Column $i$ of $C = V\,\text{softmax}(K^\top Q)$ is $c_i$.

All $T$ contextual representations computed in *one matrix multiply* — parallelism is satisfied.

# Scaled Dot-Product Attention

We have a working, position-aware design. But does it *train*?

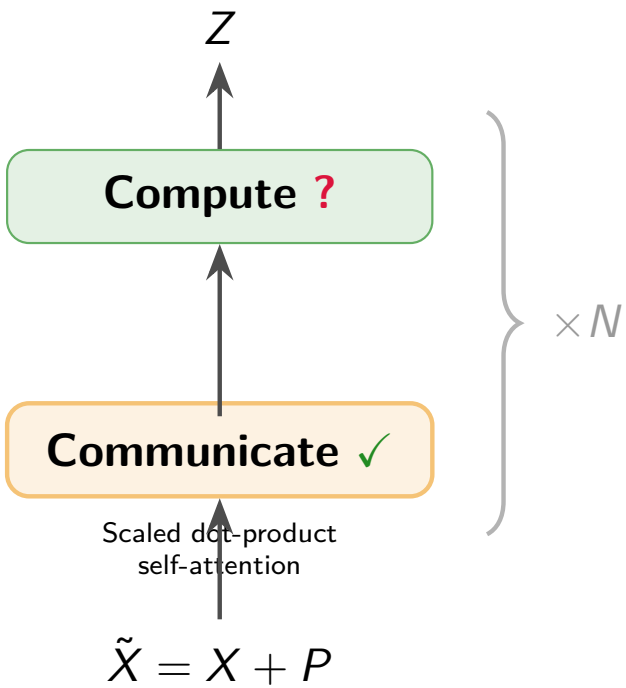
The dot product $k_i^\top q_j$ is a sum of $d_k$ terms. For large $d_k$ (e.g., 512), these scores grow in magnitude, causing softmax to produce near-one-hot outputs. Gradients through saturated softmax vanish.

**Fix:** Divide by $\sqrt{d_k}$ to restore unit variance:

$$\boxed{\text{Attention}(\tilde{X} \mid W_K, W_Q, W_V) = V \,\text{softmax}\Big(\frac{K^\top Q}{\sqrt{d_k}}\Big)}$$

Not a tunable hyperparameter — determined by variance scaling.

# Proto-Block Update — Communication Designed




The **Communicate** box is now filled:

- Position-aware (sinusoidal or learned $P$)
- Trainable ($W_K, W_Q, W_V$)
- Scaled ($1/\sqrt{d_k}$) for gradient health
- Fully parallel (one matrix multiply)

Next: the single-head bottleneck, then the **Compute** box.

# Pause & Check 1

**Q1.** In $K^\top Q \in \mathbb{R}^{T \times T}$, what does entry $(j, i)$ represent? What does column $i$ become after softmax?

$(K^\top Q)_{ji} = k_j^\top q_i =$ compatibility score between tokens $j$ and $i$. After softmax, column $i$ is $\alpha_i$ — the attention distribution for query token $i$.

**Q2.** We added positional encodings *before* the matrix form. Why couldn't we add them after computing attention weights?

By permutation equivariance: without position, the attention weights are the same for any permutation of the input. Adding position after attention doesn't help — the $\alpha_i$'s are already computed order-blind. Position must enter before QKV projections.

# The Single-Head Bottleneck — A Concrete Failure

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

Token "it" (position 8) needs to attend to multiple tokens *for different reasons*:

- **Coreference:** "it" $\rightarrow$ "animal" (referent)
- **Predicate:** "it" $\rightarrow$ "tired" (attributed property)
- **Negation scope:** "it" $\rightarrow$ "didn't" (negation)

A single head produces **one** distribution $\alpha_8 \in \Delta^9$ for token "it."

The contextual representation is a *single* attention-weighted average:

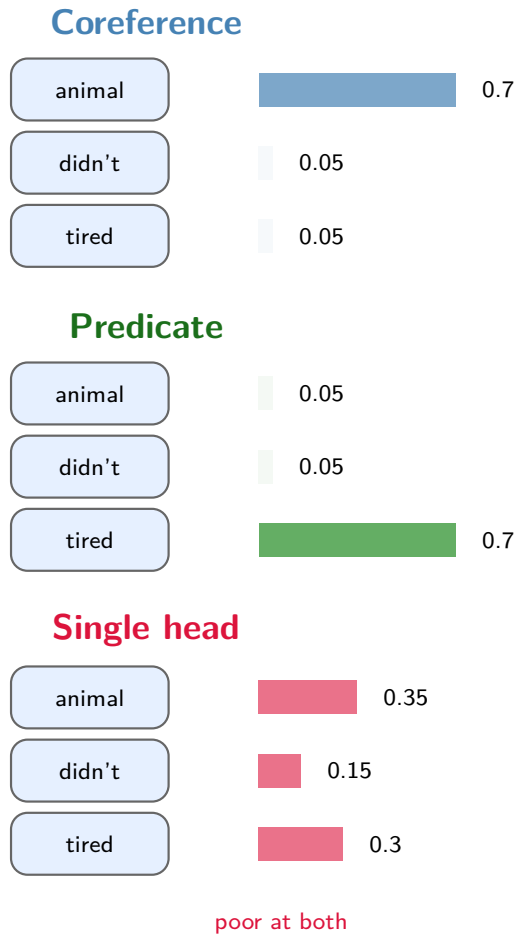
$$c_{\text{it}} = \sum_{j=1}^{10} (\alpha_8)_j \, v_j$$

# Why Single-Head Is an Impossibility

Since $(\alpha_8)_{\text{animal}} + (\alpha_8)_{\text{tired}} \leq 1$, increasing weight on "animal" (coreference) *necessarily* decreases weight on "tired" (predicate).

A single attention distribution cannot independently *control* multiple relationships via separate routing patterns.

One head provides one routing pattern per query position. Multiple simultaneous relations require multiple routing patterns.



# The Fix — Multi-Head Self-Attention (MHSA)

If one attention-weighted average can't carry all the information, compute several in parallel.

With $H$ heads, each head $h = 1, \ldots, H$ has its own projections $W_K^{(h)}, W_Q^{(h)}, W_V^{(h)}$:

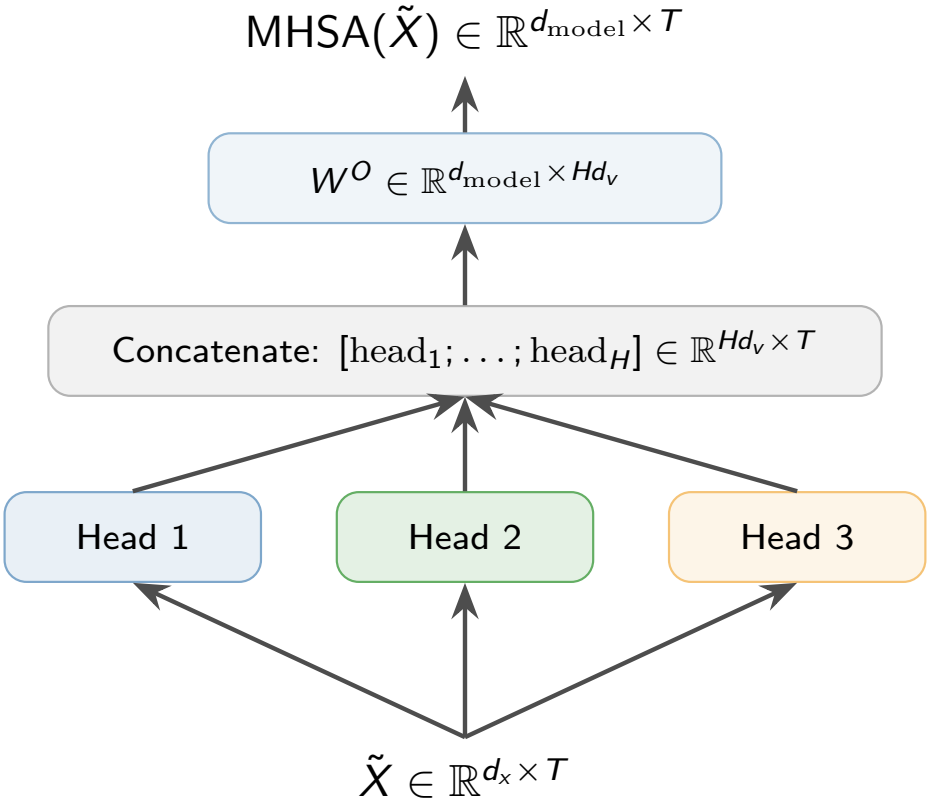
$$\mathrm{head}_h = \mathrm{Attention}(\tilde{X} \mid W_K^{(h)}, W_Q^{(h)}, W_V^{(h)}) \in \mathbb{R}^{d_v \times T}$$

$$\mathrm{MHSA}(\tilde{X}) = W^O \begin{bmatrix} \mathrm{head}_1 \\ \vdots \\ \mathrm{head}_H \end{bmatrix} \in \mathbb{R}^{d_{\mathrm{model}} \times T}$$

Head 1 attends to "animal" (coreference), head 2 to "tired" (predicate), head 3 to "didn't" (negation) — *independently*.

The output projection $W^O$ recombines: the final $c_i$ is a learned linear combination of $H$ independent attention-weighted averages.

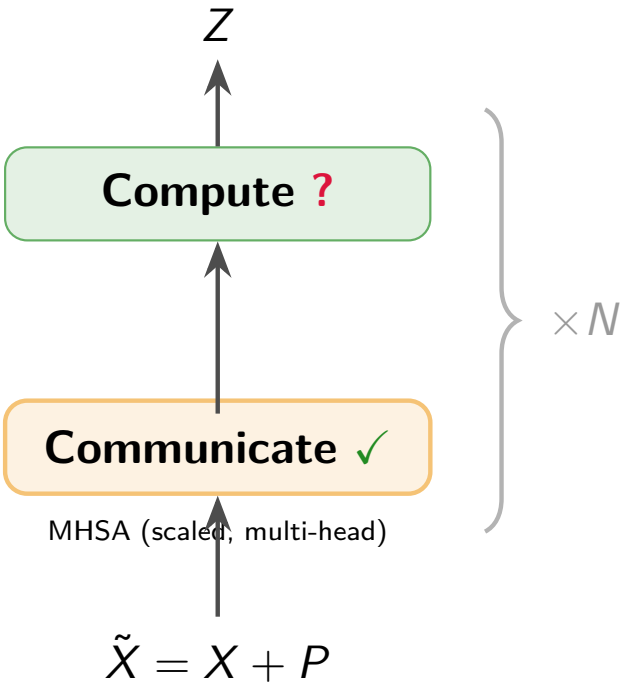
# MHSA — Schematic




Typical: $d_k = d_v = d_{\text{model}}/H$.
Total params $\approx$ single head with $d_k = d_{\text{model}}$.

**Key nuance:** MHSA is a reparameterization, not additional capacity. The multi-head structure biases learning toward diverse patterns.

# Proto-Block Update — Multi-Head Upgrade




**Communicate** upgraded to MHSA:

- Position-aware
- Scaled $(1/\sqrt{d_k})$
- Multi-channel ($H$ independent heads)
- Recombined via learned $W^O$

**Compute** is still unknown.

Analogy: multiple convolutional filters detect multiple spatial features. Multiple heads detect multiple token-to-token relationships.

# Pause & Check 2

**Q1.** With $H = 8$ heads and $d_{\text{model}} = 512$, what is $d_k$ per head? Is MHSA $8\times$ more expensive than single-head attention?

$d_k = 512/8 = 64$. Not $8\times$ — total dimension across heads equals single large head. $W^O$ adds cost, but overall compute is comparable.

**Q2.** In the pronoun resolution sentence, name two different attention patterns that two heads might learn.

Head 1: coreference (it $\rightarrow$ animal). Head 2: predicate attribution (it $\rightarrow$ tired). Any answer showing independent linguistic functions.

# Why Communication Alone Is Not Enough

We've designed MHSA — a powerful communication primitive. Could we just stack $N$ layers of MHSA and skip the computation phase?

**No.** Here is a precise failure:

Each MHSA layer produces, for each token, a weighted average of value vectors (one per head, linearly recombined). The residual connection adds the input. After $N$ such layers, the representation of each token is a (complicated) weighted combination of the original embeddings $\tilde{x}_1, \ldots, \tilde{x}_T$ — with no nonlinear transformation applied to any individual token.

# The "Is Plural" Argument

Suppose no token embedding linearly encodes the feature "is plural."

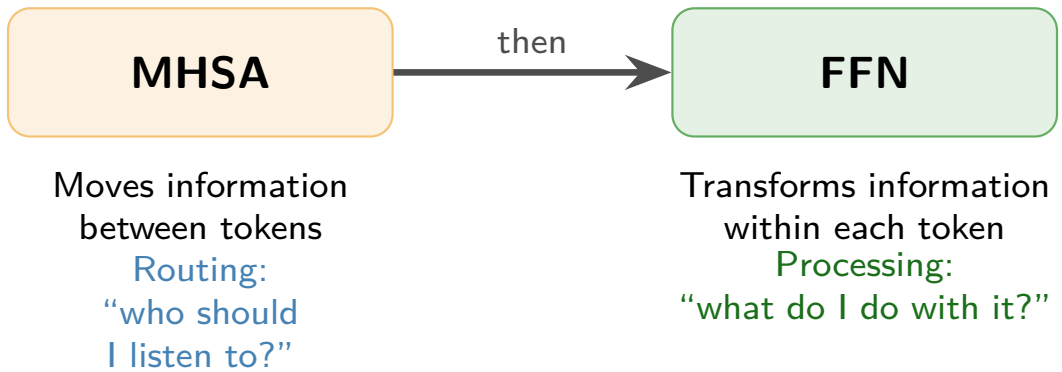
1. Each value vector $v_j$ does not encode "is plural."
2. Any attention-weighted average $\sum_j \alpha_j v_j$ also does not encode "is plural" — such averages cannot leave the affine span of the inputs.
3. Therefore MHSA *cannot create* the feature "is plural." It can only select among existing representations.

The same argument applies to "past tense," "negation scope," "semantic role," or any feature not already linearly present in the token embeddings.

MHSA *routes* information between tokens but cannot *transform* information within a token.

If a feature is not linearly present in the embeddings, no linear classifier on top of attention outputs can recover it.

# Communication Moves, Computation Transforms




Both are necessary. This is why the proto-block has two phases, not one.

# The Computation Phase — Position-wise FFN

$$\mathrm{FF}(y) = W^2\sigma(W^1 y + b^1) + b^2, \qquad \sigma = \mathrm{ReLU}$$

Applied independently to each token (each column of the matrix). Same parameters at every position.

**Role:** After MHSA enriches each token's representation with contextual information, the FFN applies a nonlinear transformation to *process* that information — extracting features, computing abstractions, mapping to useful intermediate representations.

**No inter-token interaction:** The FFN is the *only* sublayer where tokens are processed in complete isolation. All inter-token communication happens in MHSA.

The FFN introduces nonlinearity so that downstream representations become linearly separable for the task at hand.

# Assembling the Encoder Transformer Block

We now have both phases. The encoder block stacks them with residuals and normalization for trainability.

**Pre-norm** (standard in practice: GPT-2, LLaMA):

$$X' = X + \text{MHSA}(\text{LayerNorm}(X)) \quad \leftarrow \text{Communication (+ residual)} \tag{1}$$

$$Z = X' + \text{FF}(\text{LayerNorm}(X')) \quad \leftarrow \text{Computation (+ residual)} \tag{2}$$

**Post-norm** (original Vaswani et al., 2017):

$$X' = \text{LayerNorm}(X + \text{MHSA}(X)) \tag{3}$$

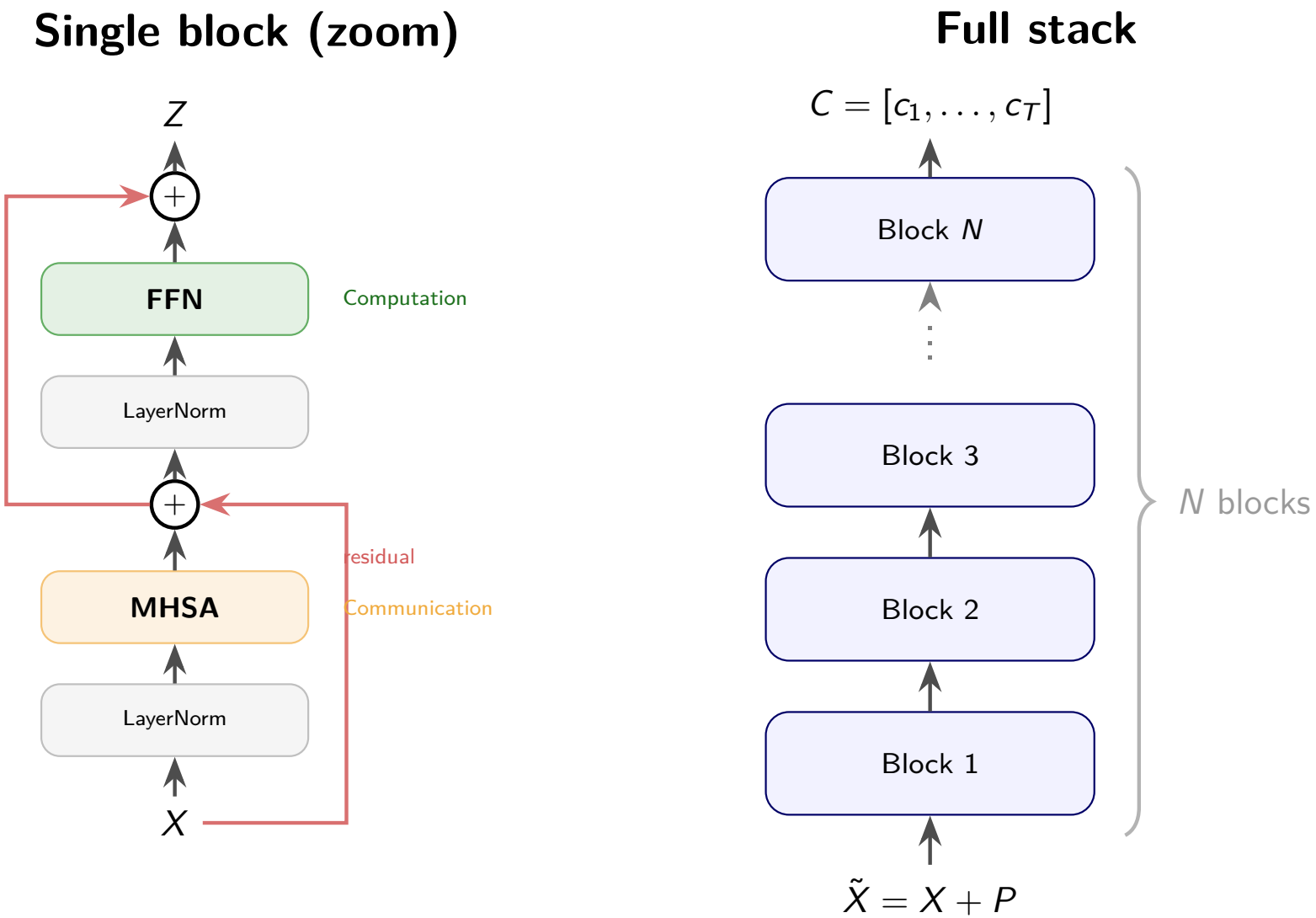
$$Z = \text{LayerNorm}(X' + \text{FF}(X')) \tag{4}$$

An **encoder transformer** is a stack of $N$ such blocks, each with its own parameters.

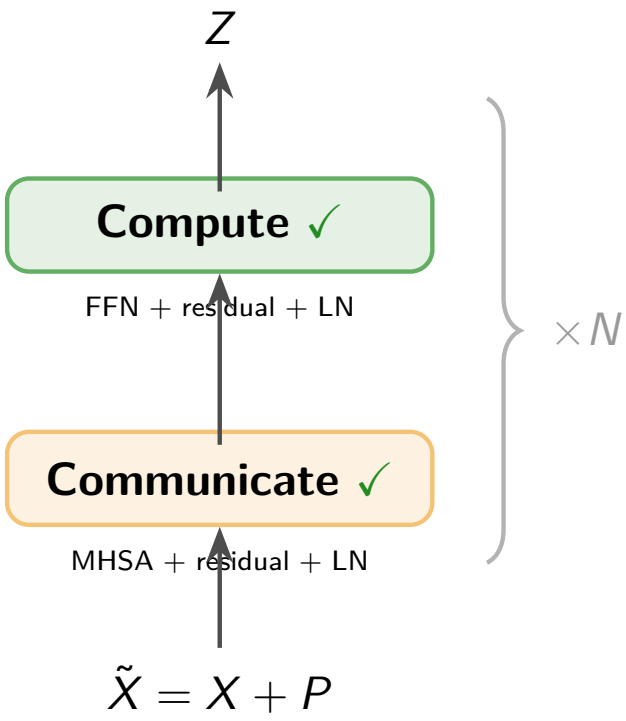
# Why Residuals and LayerNorm

- **Residual connections:** Each block learns a *perturbation* to the identity. Enables gradient flow through $N$ stacked blocks — same principle as ResNets (L15).
- **LayerNorm:** Normalizes across the feature dimension per token. Stabilizes training by controlling the scale of activations.
- **Pre-norm preferred:** Better-conditioned gradients at initialization (Xiong et al., 2020). Normalization before the sublayer means the sublayer inputs have controlled scale regardless of depth.

# Encoder Block Diagram



# Proto-Block — Complete




✓ **Block complete.**

- **Communication** = MHSA
- **Computation** = FFN
- Residuals + LayerNorm for trainability

Next: stress-test by breaking things, then constrain for generation.

# Pause & Check 3

**Q1.** Which sublayer allows token 3 to "see" token 7? Which does not?

MHSA (communication) — tokens interact via attention. FFN is per-token — tokens 3 and 7 are processed in complete isolation.

**Q2.** If we stack $N$ blocks of MHSA + residual *without FFN*, what can the model *not* do?

It cannot create new features. Outputs are weighted combinations of original embeddings — no nonlinear per-token transformation. The model can route information but not process it. Features like "this token is a verb in past tense" cannot be computed if they're not linearly present in the embedding.

# Four Ways to Break the Transformer

We claimed every component is necessary. Let's verify — watch the pronoun resolution sentence break in four different ways.

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

| **Remove** | **What breaks** | **Failure on pronoun sentence** |
|---|---|---|
| Positional encoding | Order-blind | "dog bit man" = "man bit dog" |
| Multi-head → single | One $\alpha$ per query | "it" cannot attend to both "animal" and "tired" |
| Residual connections | No skip paths | Gradients vanish; untrainable (cf. ResNets, L15) |
| FFN | No per-token nonlinearity | Cannot compute "this is a pronoun requiring resolution" |

Each component exists because removing it causes a *specific, identifiable failure*.

# From Encoder to Decoder — Two Requirements

We replaced the RNN *encoder* with self-attention blocks. Now we replace the RNN *decoder*.

The autoregressive objective $p(Y \mid X) = \prod_{t=1}^{T'} p(y_t \mid y_{1:t-1}, X)$ imposes two requirements:

1. **Access encoder representations.** Attend to $C = [c_1, \ldots, c_T]$ — the transformer analogue of Bahdanau attention. ← cross-attention
2. **Enforce causal communication.** Prediction for $y_t$ depends only on $y_{<t}$, not future tokens. In a parallel architecture, this must be imposed explicitly. ← causal masking

We solve these in order: cross-attention first (extends the previous lecture), then causal masking (new derivation).

# Cross-Attention — Definition

We now formalize cross-attention. Keys and values come from the encoder; queries from the decoder.

Given encoder representations $C = [c_1, \ldots, c_T]$ and decoder representations $Y = [y_1, \ldots, y_{T'}]$ (from $Y_{\text{aux}}$ at training, generated prefix at inference):

$$K = W_K C, \qquad V = W_V C, \qquad Q = W_Q Y$$

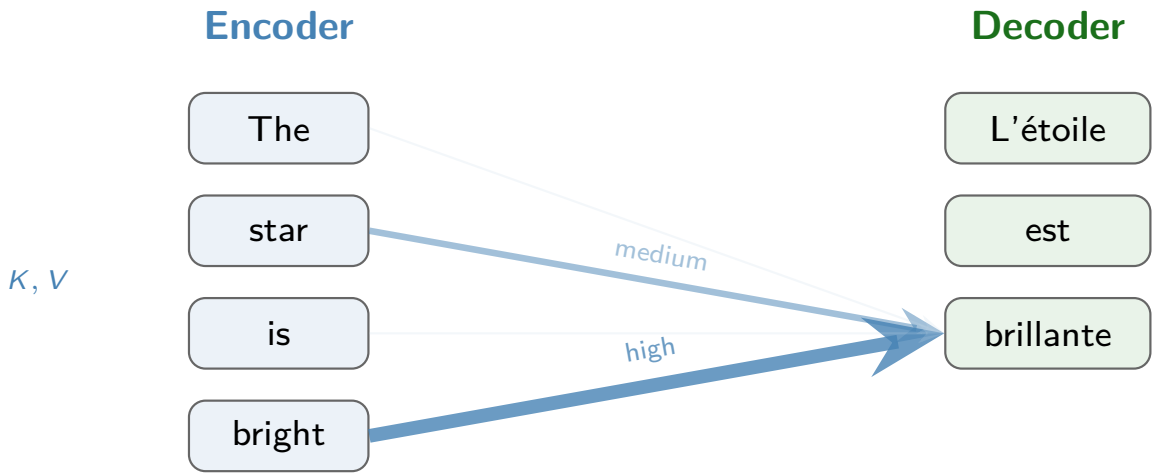
$$\text{CrossAttn}(Y, C) = V \ \text{softmax}\Big(\frac{K^\top Q}{\sqrt{d_k}}\Big) \in \mathbb{R}^{d_v \times T'}$$

Keys/values from the encoder ("memory"), queries from the decoder ("question").

Exactly Bahdanau attention from last lecture in QKV language.

# Cross-Attention in Translation

"The star is bright" $\rightarrow$ "L'étoile est brillante"




Decoder token "brillante" queries the encoder.
High weight on "bright" (semantics); medium on "star" (gender agreement).

# The Training Objective Forces Masking

The decoder will use self-attention to let output tokens communicate. But the autoregressive training objective constrains this communication: it must be **causal**.

**Goal:** Train a model to predict the next token:

$$\mathcal{L}(\theta) = -\sum_{t=1}^{T} \log p_\theta(y_t \mid y_1, \ldots, y_{t-1})$$

**Constraint:** The prediction for $y_t$ may depend only on $y_1, \ldots, y_{t-1}$.

**Problem:** With unrestricted self-attention in the decoder, MHSA lets every token attend to every other. When computing $c_t$, token $t$ reads from future tokens. This does not make sense at inference!

# The Causal Mask

$$M_{ji} = \begin{cases} 0 & \text{if } j \leq i \\ -\infty & \text{if } j > i \end{cases}$$

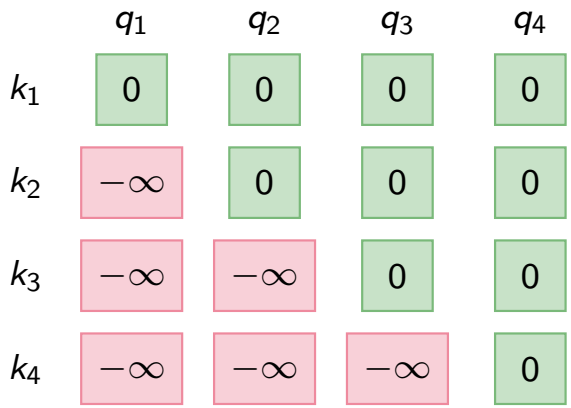
Since $e^{-\infty} = 0$, softmax zeroes out future positions:

$$\boxed{\text{CausalAttention}(X) = V\ \text{softmax}\Big(\frac{K^\top Q + M}{\sqrt{d_k}}\Big)}$$

Column $i$ of the masked score matrix: entries $j > i$ are $-\infty$. After softmax, token $i$ attends only to positions $1, \ldots, i$.

# Causal Mask Visualization

Sequence:
$[\langle \texttt{BOS} \rangle, \texttt{Dr.}, \texttt{Uzma}, \langle \texttt{EOS} \rangle]$




Green (0): attention allowed.
Red ($-\infty$): attention blocked.

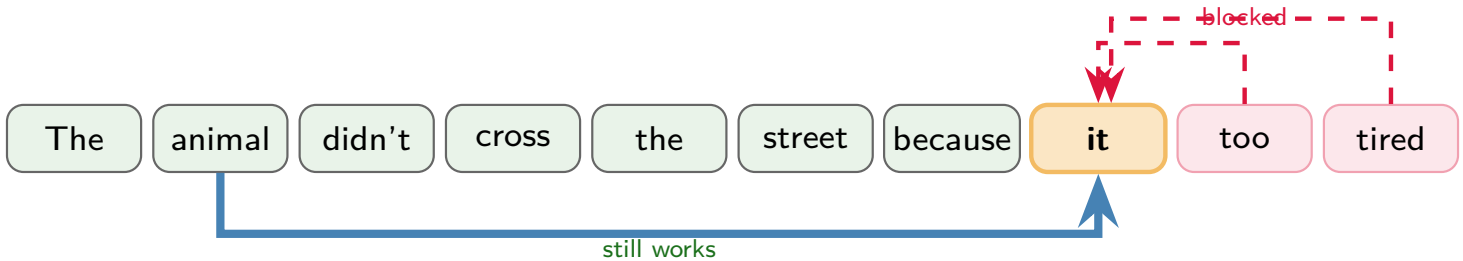
Lower triangle: token $i$ can attend to all $j \leq i$.
Upper triangle: future positions are masked out.

The mask is the same for every head and every layer — it encodes the *objective*, not a learned parameter.

# Pronoun Resolution Under Masking

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

Under causal masking, "it" (position 8) can attend to positions 1–8 but **not** 9–10 ("too," "tired").




Coreference "it" $\rightarrow$ "animal" still works (position $2 < 8$). But the model cannot use "tired" to help resolve it — must rely on left context alone.

This is strictly harder than bidirectional attention. This asymmetry separates BERT from GPT.

# Pause & Check 4

**Q1.** We add $-\infty$ before softmax. Why not multiply by 0 instead?

$e^{-\infty} = 0 \rightarrow$ zero weight, renormalize over the rest. Multiplying by 0 gives $e^0 = 1 \rightarrow$ *uniform* weight on masked positions. Opposite of masking.

**Q2.** We derived masking from the autoregressive objective. Does BERT need masking?

No. BERT uses masked language modeling (bidirectional context). No left-to-right factorization $\rightarrow$ no causal constraint $\rightarrow$ unrestricted communication.

# The Decoder Transformer Block

Three sublayers, each mapped to the communication/computation framework:

1. **Masked MHSA + residual + LN** ← constrained
   self-communication

$$Y' = Y + \text{MHSA}(\text{LayerNorm}(Y) \mid M)$$

2. **Cross-attention + residual + LN** ← cross-sequence
   communication

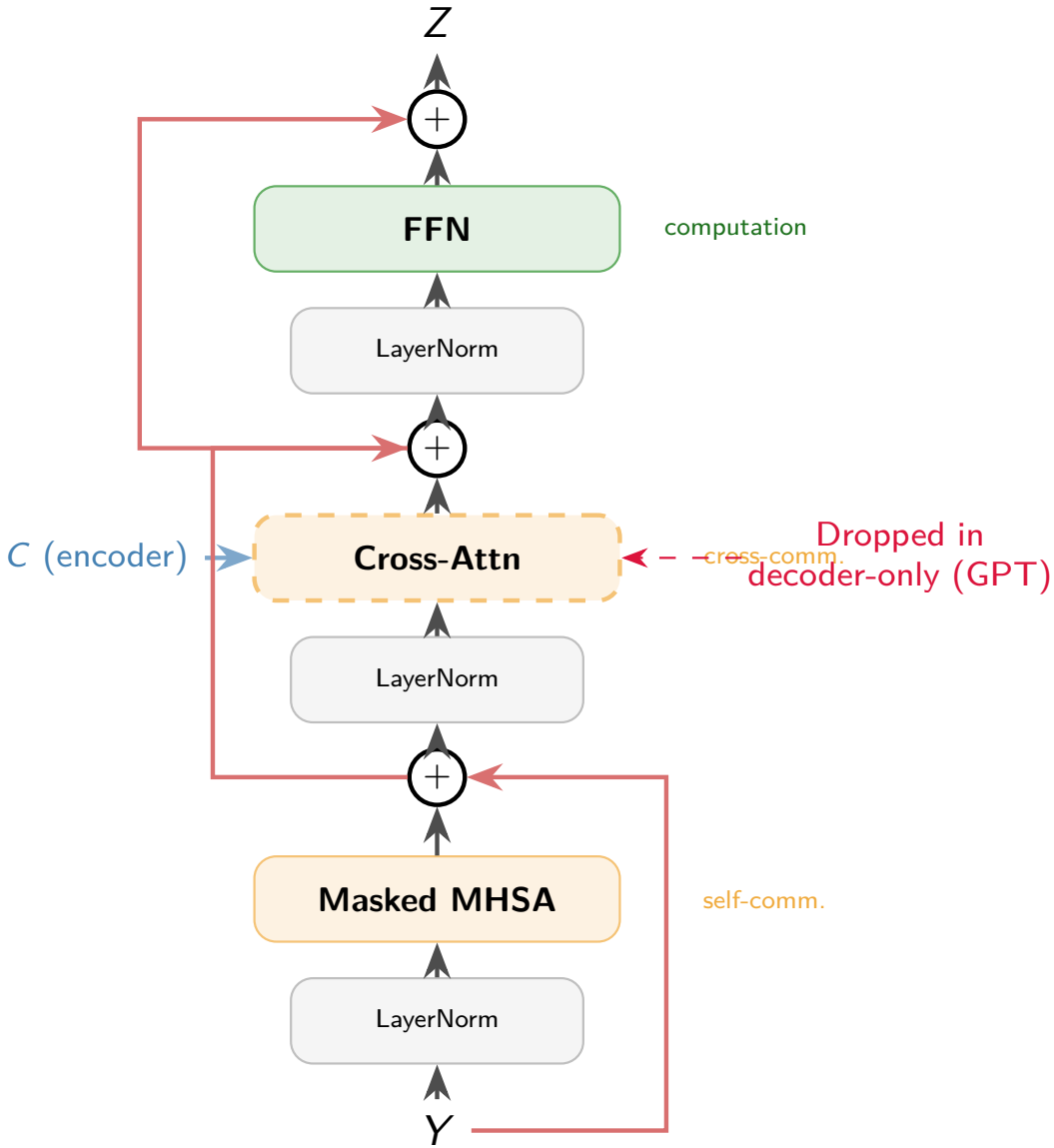
$$Y'' = Y' + \text{CrossAttn}(\text{LayerNorm}(Y'), C)$$

3. **FFN + residual + LN** ← computation

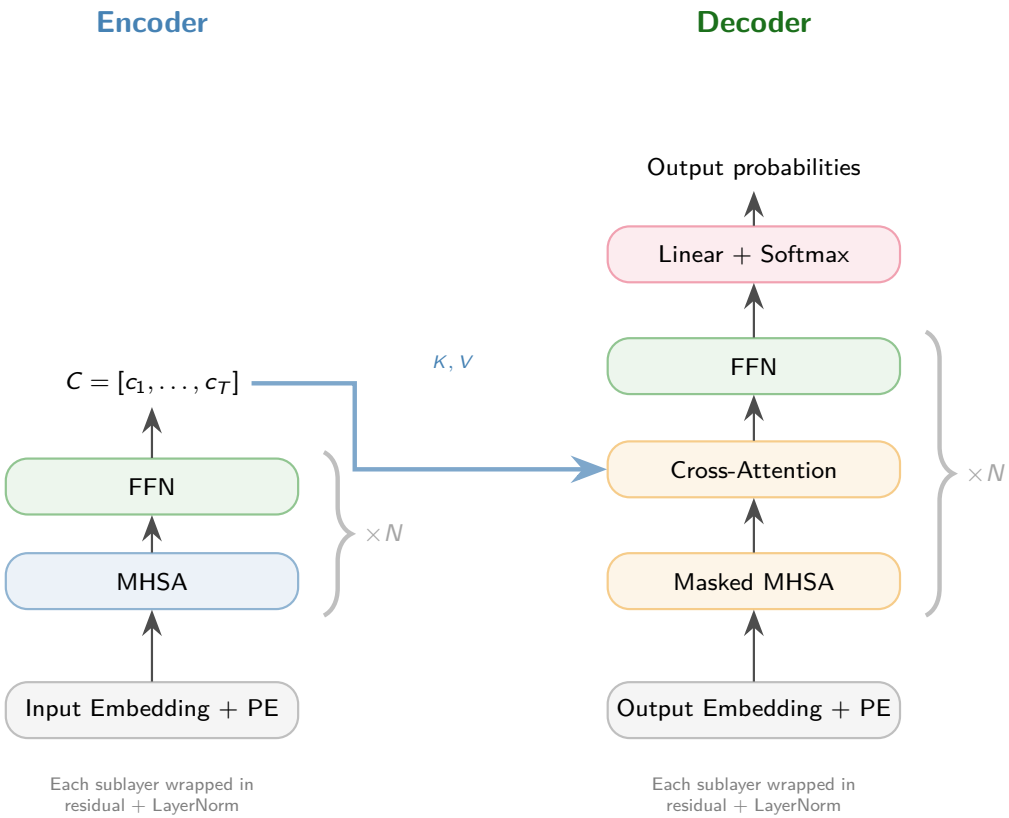
$$Z = Y'' + \text{FF}(\text{LayerNorm}(Y''))$$

For **decoder-only** models (GPT): sublayer (ii) is dropped — no encoder, no cross-attention.

# Decoder Block Diagram



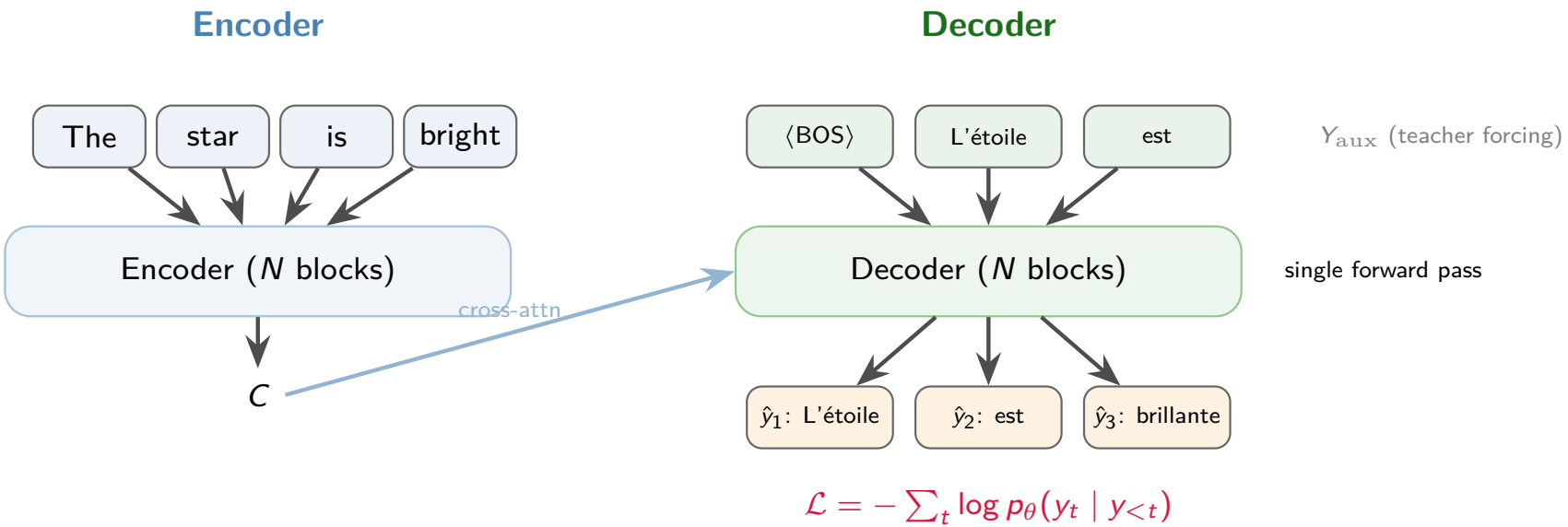
# The Full Encoder–Decoder Transformer




The encoder produces contextual representations $C$. Each decoder block applies three sublayers: causal self-attention (within the output sequence), cross-attention (to $C$), and FFN (per-token computation).

# Translation Example — Training

"The star is bright" → "L'étoile est brillante"



# Inference — Autoregressive Decoding

No ground truth at inference. Generate one token at a time:

1. $C = \text{Encoder}(X)$.
2. $Y^{(0)} = [\langle \texttt{BOS} \rangle]$.
3. For $t = 1, 2, \ldots$: decode $z_t$, predict $\hat{y}_t$, append. Stop at $\langle \texttt{EOS} \rangle$.

For **decoder-only** (GPT): no encoder; continue prompt autoregressively.

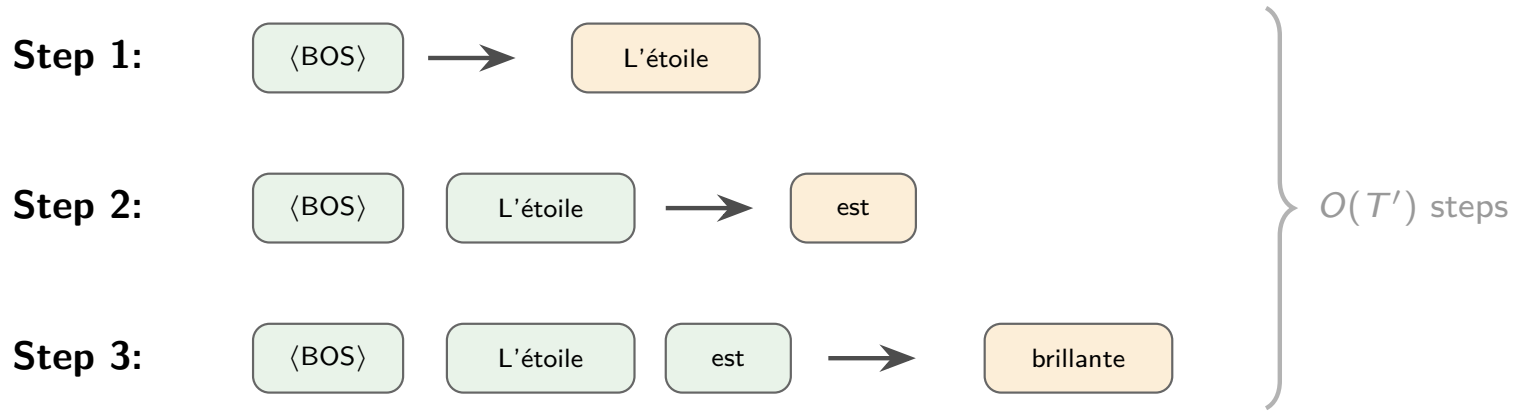
**Training/inference asymmetry:**

| | **Training** | **Inference** |
|---|---|---|
| Depth | $O(1)$ (parallel) | $O(T')$ sequential |
| Input | Ground truth (teacher forcing) | Own predictions |

The fundamental cost of autoregressive generation.

# Translation Example — Step-by-Step Decoding

"The star is bright" $\rightarrow$ ?




Each step requires a full decoder pass. Caching strategies amortize repeated computation.

# Pause & Check 5

**Q1.** How many forward passes for a target of length $T'$ during training? During inference?

Training: one (parallel via mask). Inference: $T'$ (sequential).

**Q2.** A student says: "Teacher forcing is a shortcut that hurts the model." Respond.

Not a shortcut — a consequence of the causal mask enabling parallelism. But it does create exposure bias (train/test mismatch). The tradeoff: massive training efficiency at the cost of a distributional gap at inference.

# Three Families — One Design Choice

The taxonomy reduces to one design choice: *what communication structure does the block allow?*

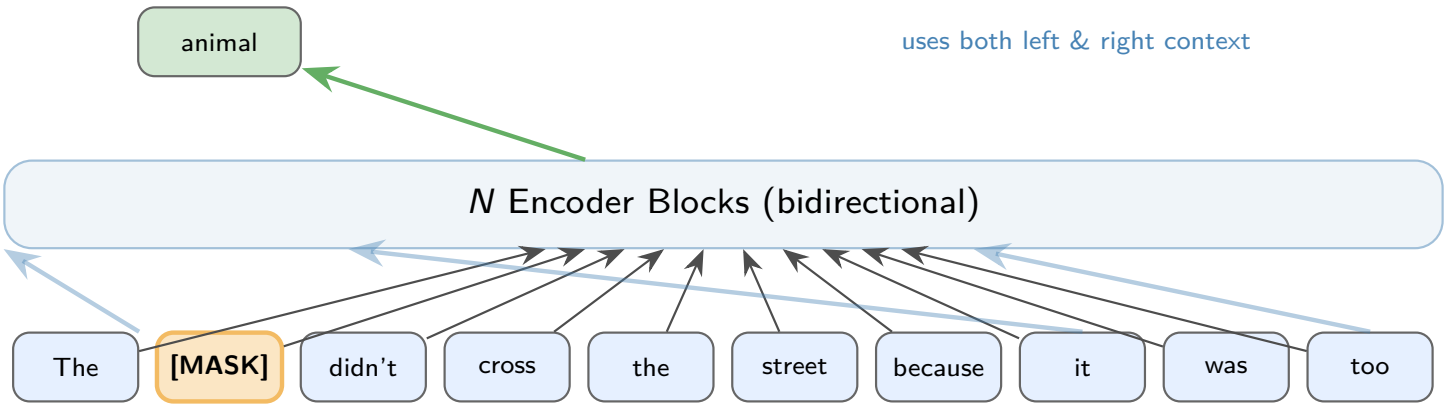
| | **Encoder-only** | **Decoder-only** | **Enc–Dec** |
|---|---|---|---|
| **Self-comm.** | Bidirectional | Causal | Enc: full; Dec: causal |
| **Cross-comm.** | None | None | Dec $\rightarrow$ Enc |
| **Pretraining** | Denoising (MLM) | Autoregressive | Seq2seq |
| **Exemplar** | BERT | GPT | T5 |
| **Strengths** | Understanding | Generation | Translation, summ. |

Bidirectional, causal, or cross-sequence. Everything else follows from this choice.

# BERT — Encoder-Only (Devlin et al., 2018)

$N$ encoder blocks. Unrestricted bidirectional communication.

**Masked Language Modeling (MLM):** Mask 15% of tokens; predict from full context. Framing: denoising autoencoder.



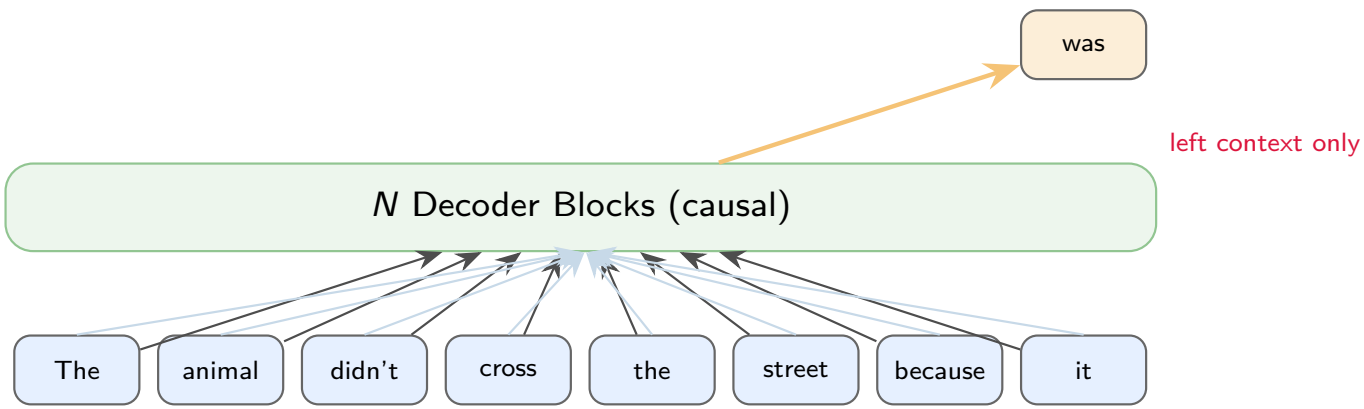

Pronoun resolution: "The [MASK] didn't cross the street because it was too tired" $\rightarrow$ predict "animal" using *both* left and right context.

# GPT — Decoder-Only (Preview)

$N$ decoder blocks with causal MHSA. No encoder, no cross-attention.

Autoregressive objective:

$$\mathcal{L}(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \log p_\theta(x_t \mid x_{1:t-1})$$



# T5 — Encoder–Decoder (Raffel et al., 2020)

**Text-to-Text Transfer Transformer:** Every task framed as text $\rightarrow$ text.

- Full encoder–decoder architecture.
- Pretraining: span corruption (mask random spans, predict them).
- Unifying: one architecture handles classification, generation, translation, summarization.

| **Task** | **Input** | **Output** |
|---|---|---|
| Translation | "translate: The star is bright" | "L'étoile est brillante" |
| Sentiment | "classify: Great movie!" | "positive" |
| Summarization | "summarize: [article]" | "[summary]" |

# Pause & Check 6

**Q1.** For the pronoun resolution sentence, which model resolves "it" $\rightarrow$ "animal" more easily: BERT or GPT? Why?

BERT — bidirectional communication uses "was too tired" (right context). GPT only uses left context.

**Q2.** Can BERT generate text autoregressively?

No. No causal mask $\rightarrow$ no left-to-right factorization $\rightarrow$ not a density model. BERT denoises; it doesn't model $p(x_1)\, p(x_2 \mid x_1) \cdots$.

# Return to the Design Requirements

Did we achieve the design requirements?

| Axis | RNN | Transformer |
|---|---|---|
| **Parallelism** | $O(T)$ sequential | Fully parallel (training) |
| **Path length** | $O(T)$ between distant tokens | $O(1)$ direct |
| **Inductive bias** | Strong recency bias | No built-in bias; learned |

**Cost:** $O(T^2 d_k)$ compute, $O(T^2)$ memory per attention layer.

Return to the design problem: we wanted $O(1)$ path length and $O(1)$ training depth. The transformer delivers both, at $O(T^2)$ cost.

# Key Takeaways

1. **Self-attention** gives $O(1)$ path length and full training parallelism, replacing RNN recurrence.
2. **Positional encoding** is a design requirement: self-attention is permutation equivariant.
3. **Multi-head attention** enables multiple independent routing patterns per token, resolving the single-head bottleneck.
4. **The FFN sublayer is necessary:** attention routes information but cannot create features not linearly present in the input embeddings.
5. **Causal masking** is forced by the autoregressive training objective, not a design choice.
6. **Cross-attention** connects encoder representations to the decoder, generalizing Bahdanau attention to the transformer setting.
7. **Teacher forcing + causal mask** enable fully parallel training; inference remains sequential.

# Questions?

Next time: How GPT works in detail, scaling laws, modern LLM design choices.

# Appendix Contents

**A.** Perplexity — Definition and Interpretation

# Appendix A: Perplexity — Definition

Perplexity is the standard evaluation metric in language modeling papers. It measures how well a model predicts held-out text — lower is better.

$$\mathrm{PPL} = \exp\Big(-\frac{1}{T'}\sum_{t=1}^{T'} \log p_\theta(y_t \mid y_{1:t-1})\Big)$$

Inverse geometric mean of conditional token probabilities.

# Appendix B: Perplexity — Interpretation

- PPL $\geq 1$. PPL $= |\mathcal{V}|$ for uniform predictions.
- PPL $= k \approx$ choosing uniformly among $k$ options at each step.
- $\log_2(\text{PPL}) =$ bits per token.

| **Model** | **PPL (WikiText-103)** |
|---|---|
| GPT-2 | $\approx 35$ |
| GPT-3 | $\approx 20$ |