“Attention Is All You Need” (Transformers)

1) Big idea

- A **Transformer** turns an input sequence (e.g., a sentence) into an output sequence (e.g., a translation) using only **attention** and small feed-forward networks—**no recurrence, no convolutions**.
- It processes the **whole input in parallel** (great for speed) and uses attention to decide **which tokens should look at which other tokens**.

2) Model layout (encoder–decoder)

- The original paper stacks **N = 6 encoders** and **N = 6 decoders**.

N is a **hyperparameter**; you can choose other depths.

- **Each encoder layer** has:
    1. **Self-Attention**
    2. **Position-wise Feed-Forward Network (FFN)**
       Each sublayer is wrapped with **Add & LayerNorm** (residual connection + normalization).
- **Each decoder layer** has **three** sublayers:
    1. **Masked Self-Attention** (can’t look at future tokens)
    2. **Encoder–Decoder (Cross) Attention** (queries from the decoder attend to the encoder’s outputs)
    3. **Feed-Forward Network**
       Again, each sublayer uses **Add & LayerNorm**.

3) From tokens to vectors

1. **Tokenization** → integers.
2. **Embedding** maps each token to a vector of size **d_model** (paper uses **d_model = 512**).
3. **Positional encoding** is **added** to embeddings so the model knows order and relative distance (see §7).

So the input to the first encoder layer is a matrix **X ∈ ℝ^{L × d_model}** where L is the sequence length.

4) Scaled dot-product self-attention (inside a layer)

For each token's vector x we make three projections:

- **Queries**: $Q = XW^Q$
- **Keys**: $K = XW^K$
- **Values**: $V = XW^V$

Typical shapes (paper defaults):

- $W^Q, W^K \in \mathbb{R}^{d_{model} \times d_k}$, $W^V \in \mathbb{R}^{d_{model} \times d_v}$
- With **8 heads**, usually $d_k = d_v = d_{model}/8 = 64$.

Scores (how much each token should attend to each other token):

$$\text{Scores} = \frac{QK^\top}{\sqrt{d_k}} \quad (\text{shape } L \times L)$$

- The division by $\sqrt{d_k}$ keeps the dot products from getting too large, so **Softmax** doesn't saturate and gradients stay healthy.

Attention weights:

$$A = \text{Softmax}(\text{Scores})$$

Weighted combination of values:

$$Z = AV \quad (\text{shape } L \times d_v)$$

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

- When computing the representation for "it", attention will allocate **higher weights** to the token that best explains "it" (likely "animal") and lower weights elsewhere. This disambiguation emerges from training.

Everything above is done **for all tokens at once** using matrices (parallel, not step-by-step like RNNs).

5) Multi-head attention (why and how)

- One attention head may focus on only a **single relation**.
- **Multi-head attention** uses **h heads** (paper: **h = 8**), each with its own W^Q, W^K, W^V.
- Each head sees **lower-dimensional** projections (e.g., 64 dims) and learns to focus on **different patterns** (syntax, long-range links, etc.).

Process:

1. Compute $Z_1, Z_2, \ldots, Z_h$ from the h heads.
2. **Concatenate**: $\text{Concat}(Z_1, \ldots, Z_h) \in \mathbb{R}^{L \times (h \cdot d_v)}$.
3. Project back to model size with $W^O \in \mathbb{R}^{(hd_v) \times d_{model}}$.

6) Feed-Forward Network (FFN)

- Applied **independently to each position** (same weights for all positions).
- Two linear layers with a nonlinearity:

$$\mathrm{FFN}(x) = \max(0,\ xW_1 + b_1)\, W_2 + b_2$$

  Paper defaults: **d_ff = 2048** (hidden size), going back to **d_model = 512**.
- Think of FFN as a small MLP that **mixes features** within a token after attention has mixed information **across tokens**.

7) Positional encoding (ordering & distance)

Because attention itself is order-agnostic, we add a **positional signal** to each embedding.

- **Sinusoidal** encodings in the paper: fixed sine/cosine waves with different frequencies so the model can infer **relative distances**.
- Many implementations use **learned** positional embeddings; both work.

Result: the network knows that “animal” is **before** “it”, and how far apart they are.

8) Residual connections & LayerNorm (Add & Norm)

- For every sublayer, we do:

$$\mathrm{LayerNorm}(x + \mathrm{sublayer}(x))$$

- **Residuals** don’t “skip if a part fails”; they **always** add the original input back in.

  Benefits:
  - Preserve the original signal,
  - Improve gradient flow,
  - Let sublayers learn **refinements** instead of having to recreate the whole representation.
- The original paper used **post-norm** (norm after adding). Many modern models use **pre-norm** (norm before) for training stability; both follow the same idea.

9) Masks (very important)

Two kinds are common:

1. **Padding mask**: prevents attending to padding tokens added to equalize lengths. Used in encoder & decoder.
2. **Look-ahead (causal) mask**: in **decoder self-attention**, prevents a position from attending to **future** positions.

- This ensures generation is **autoregressive**: token ttt can only use tokens ≤t\leq t≤t.

Masks are applied by setting the corresponding score entries to a very negative number **before Softmax** so their attention weight becomes (essentially) zero.

10) Encoder vs Decoder roles

Encoder

- Self-attention is **unmasked** (except for padding), so each token can look **anywhere** in the input.
- The stack outputs a **memory** (context representations) for all input positions.

Decoder

1. **Masked self-attention** produces a representation for each output position that only sees **previous** outputs.
2. **Encoder–decoder (cross) attention**:
   - **Queries** come from the decoder's current representations,
   - **Keys/Values** come from the **encoder memory**.
     This lets the decoder pull in the relevant parts of the input while generating each word.
3. **FFN**, then a final **Linear + Softmax** over the vocabulary to pick the next token.

Training: uses **teacher forcing** (the gold previous token is provided), so all time steps can be computed **in parallel** with the mask.
Inference: generates **one token at a time** (greedy, beam search, etc.), feeding each new token back in.

11) Why attention helps with pronouns ("it" example)

- The representation for "it" forms a **query** that matches best with the **key** for "animal" (and poorly with "street").
- After Softmax, the **weight on "animal" is high**, so the resulting vector is a mixture that strongly reflects "animal".
- Deeper layers refine this: early layers may capture **local** relations; later layers can integrate **long-range** and **semantic** cues.

12) Typical default hyperparameters (paper)

- **Layers**: N = 6 encoders, 6 decoders
- **Model size**: $d_{model} = 512$
- **Heads**: $h = 8 \rightarrow d_k = d_v = 64$
- **FFN hidden**: $d_{ff} = 2048$
- **Dropout**: 0.1 (applied in attention and FFN)
- **Optimization**: Adam with a "warm-up then decay" learning-rate schedule

$$lr = d_{model}^{-0.5} \cdot \min(step^{-0.5}, step \cdot warmup^{-1.5})$$

- **Label smoothing** around 0.1 is often used.

These are **starting points**, not rules.

13) Putting it all together (encoder–decoder pass)

1. **Inputs → embeddings → add positional encodings**.
2. **Encoder stack** (Self-Attention → Add&Norm → FFN → Add&Norm) × N → **memory**.
3. **Start token** enters the decoder.
4. For each output position (masked):
   - Decoder **self-attention** attends to earlier generated tokens,
   - **Cross-attention** attends over the encoder memory,
   - **FFN** refines the representation,
   - **Linear + Softmax** chooses the next token.
5. Stop at **end-of-sequence** token.

14) Practical notes & limits

- Complexity of attention is **O(L^2)** in time and memory because of the L×LL \times LL×L score matrix; very long sequences are expensive. (Many later variants reduce this, but that's beyond the original paper.)
- Residual paths mean information can **flow unchanged** if a sublayer isn't helpful, while still letting the sublayer add useful adjustments.
- Multi-head lets the model track **different relations simultaneously** (syntax, agreement, coreference, etc.).

15) Mini glossary (plain language)

- **Query (Q)**: “What am I looking for?”
- **Key (K)**: “What do I offer?”
- **Value (V)**: “What information should be taken if I’m chosen?”
- **Softmax**: turns raw scores into **probabilities** that sum to 1.
- **Masked attention**: forces the model to **ignore** some positions (future tokens or padding).
- **Residual connection**: add input back to output: **output + input**, then normalize.