

**What are the main components of a transformer model**

The Transformer architecture revolutionized machine learning with its ability to process sequential data in parallel. Here are its core components:

**1. Encoder-Decoder Architecture**

The original Transformer uses stacked encoder and decoder layers:

* **Encoder**: Processes input sequences into contextual representations
* **Decoder**: Generates output sequences using encoder information

*Some variants like BERT (encoder-only) and GPT (decoder-only) use partial stacks*[[1]](#fn1)[[2]](#fn2).

**2. Key Structural Components**

**Input Processing**

* **Embedding Layer**: Converts tokens to dense vectors (e.g., word → 512-dim vector)
* **Positional Encoding**: Injects sequence order information using sinusoidal functions[[3]](#fn3)[[1]](#fn1)

**Attention Mechanisms**

1. **Multi-Head Self-Attention**:
   * Processes sequences through parallel attention "heads"
   * Each head learns different contextual relationships
   * Uses Query/Key/Value matrices to compute attention scores[[4]](#fn4)[[5]](#fn5)
2. **Masked Attention** (Decoder):
   * Prevents attending to future tokens during generation
   * Enables autoregressive prediction[[5]](#fn5)
3. **Cross-Attention** (Decoder):
   * Links decoder to encoder outputs
   * Aligns target and source sequences[[6]](#fn6)[[5]](#fn5)

**Feed-Forward Networks**

* Position-wise fully connected layers
* Typically expands to 4x embedding dimension (e.g., 512 → 2048)
* Uses ReLU activation for non-linearity[[5]](#fn5)[[2]](#fn2)

**Normalization & Residual Connections**

* **Layer Normalization**: Stabilizes training by normalizing outputs
* **Skip Connections**: Preserves gradient flow through deep networks
* *Post-LN vs Pre-LN* variants affect training dynamics[[5]](#fn5)[[2]](#fn2)

**3. Layer Stacking**

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Original Paper | BERT-like | GPT-like |
| Encoder Layers | 6 | 12-24 | - |
| Decoder Layers | 6 | - | 12-48 |
| Attention Heads | 8 | 12-16 | 16-96 |

*Modern models scale these parameters significantly*[[5]](#fn5).

**4. Output Processing**

* **Final Linear Layer**: Maps decoder outputs to vocabulary size
* **Softmax**: Converts logits to token probabilities[[5]](#fn5)

This architecture enables transformers to handle long-range dependencies while maintaining parallel computation efficiency. The combination of self-attention mechanisms with position-aware processing forms the foundation for modern LLMs like ChatGPT and Claude[[7]](#fn7)[[4]](#fn4).

⁂

1. <https://www.hpe.com/emea_europe/en/what-is/transformer-model.html>

1. <https://towardsdatascience.com/attention-is-all-you-need-e498378552f9/>

1. <https://www.datacamp.com/tutorial/how-transformers-work>

1. <https://www.ibm.com/think/topics/transformer-model>

1. <https://en.wikipedia.org/wiki/Transformer_(deep_learning_architecture)>

1. <https://myscale.com/blog/key-components-transformer-models-understanding/>

1. <https://www.run.ai/guides/generative-ai/transformer-model>