# 📘 Transformers – Complete Guide

## 1. Background

### RNN / LSTM / GRU

* **RNNs:** Sequentially process words but are slow and suffer from vanishing gradients.
* **LSTM/GRU:** Improve RNNs with gating mechanisms but still process sequentially → no parallelization.

### Encoder–Decoder (Seq2Seq)

* Used in translation.
* **Encoder:** Converts source sentence into context vector.
* **Decoder:** Generates output sentence step by step.
* **Limitation:** Fixed-size context struggles with long sentences.

### Attention Mechanism

* Decoder can attend to all encoder states instead of a single vector.
* Still limited with long sequences and slow sequential nature.

### Transformers

* Introduced in *“Attention Is All You Need” (2017).*
* Remove RNNs, use **only self-attention + feed-forward layers**.
* Enables full **parallelization**.

## 2. Why Transformers?

1. Fast → Parallel processing of tokens.
2. Better handling of long-range dependencies.
3. Scalable → stackable architecture.
4. Foundation of modern models (GPT, BERT, T5, ViT).

## 3. Transformer Architecture

### Encoder

* Input Embedding + Positional Encoding.
* Multi-head Self Attention.
* Feed-Forward Neural Network.
* Residual + Layer Normalization.

### Decoder

* Input Embedding + Positional Encoding.
* **Masked** Multi-head Self Attention.
* Encoder–Decoder Attention.
* Feed-Forward Neural Network.
* Residual + Layer Normalization.

## 4. Core Concepts

### Self-Attention (Scaled Dot-Product)

* Inputs: **Q (Query), K (Key), V (Value)**.
* Formula:  
  Attention(Q,K,V) = softmax((Q·Kᵀ)/√dₖ) · V

Steps: 1. Compute scores with dot product Q·Kᵀ. 2. Scale by √dₖ. 3. Apply softmax. 4. Weighted sum with V.

### Multi-Head Attention

* Multiple projections of Q,K,V.
* Learn different relationships (syntax, semantics, dependencies).
* Outputs concatenated + linear projection.

### Positional Encoding

* Injects order information.
* **Types:**
  + Sinusoidal (fixed, sin/cos functions).
  + Learned (trainable embeddings).

### Feed-Forward Network

* Two-layer MLP with activation (ReLU/GELU).
* Applied independently to each token.

### Normalization & Residuals

* **Residual connections:** help gradient flow.
* **Layer Normalization:** stabilizes training.

### Masking in Decoder

* **Look-ahead mask:** Prevents seeing future tokens.
* **Padding mask:** Ignores padding tokens in batches.

### Encoder–Decoder Attention

* Encoder output → Keys & Values.
* Decoder hidden state → Query.
* Helps decoder focus on relevant parts of the input.

## 5. Contextual Embeddings

* **Traditional:** word2vec, CBOW → static embeddings.
* **Transformers:** contextual embeddings (different meaning depending on context).

Example: “bank” = river bank vs financial bank → depends on sentence.

## 6. Advantages & Limitations

### Advantages

1. Parallelized training.
2. Long-range dependency capture.
3. Scalable to billions of parameters.
4. Multi-modal extensions.

### Limitations

1. Quadratic complexity (O(n²) for attention).
2. Requires huge data and compute.
3. Lacks strong inductive bias (e.g., CNNs for images).

## 7. Applications

* **BERT (Encoder-only):** Understanding tasks (QA, classification).
* **GPT (Decoder-only):** Text generation.
* **BART / T5:** Encoder–decoder hybrids for translation, summarization.
* **Vision Transformer (ViT):** Image tasks.
* **Multi-modal Transformers:** Text + image + audio tasks.

## 8. Process Flow

1. Input → Token embeddings + positional encodings.
2. Encoder → Self-attention + FFN.
3. Decoder → Masked self-attention + encoder–decoder attention + FFN.
4. Output → Vocabulary probabilities → Generated tokens.

✅ This document covers transformers from basics → architecture → details → applications.