1. Why Transformers

Background: Sequence Modeling Before Transformers

- **Sequential nature of data**: Many real-world problems (language, speech, DNA sequences, stock data) require models that handle ordered inputs.
- **Traditional approaches**:
  - **RNNs (Recurrent Neural Networks)**
    - Process sequences step-by-step.
    - Struggled with **long-term dependencies** → vanishing/exploding gradients.
    - Hard to train on long texts (like an entire article).
  - **LSTMs/GRUs (1997–2014)**
    - Added memory gates → solved some issues.
    - But still **sequential** (slow to train, no parallelism).
    - Example: translating one sentence could take seconds/minutes.

Problem: As datasets and tasks grew (translation, summarization, Q&A), these models became bottlenecks.

The Breakthrough: Attention & Transformers

- In **2014**, "Attention" was introduced (Bahdanau et al.) for machine translation.
  - Allowed models to **focus on relevant words** instead of compressing everything into a single hidden state.
- In **2017**, Google researchers published *"Attention is All You Need"*.

    - **Removed recurrence entirely.**
    - Proposed the **Transformer architecture** built only on **attention + feedforward layers**.
    - **Key benefits:**
        - **Parallel training** on GPUs (much faster than RNNs).
        - **Better at long-range dependencies** (can link the first and last word easily).
        - **Scalable** → enabled training of massive language models.

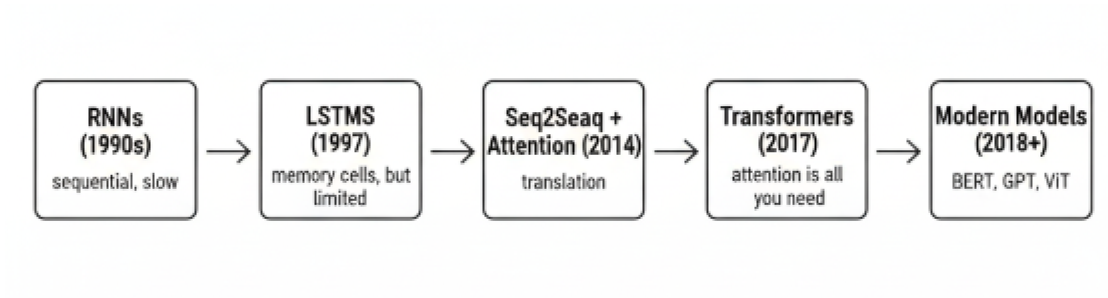
Why Is This a Revolution?

- Training time dropped from **weeks to days** for large translation tasks.
- Transformers became the **standard backbone** for NLP, vision, speech, and multimodal AI.
- Enabled the wave of **Large Language Models (LLMs)** like GPT, BERT, T5, and eventually ChatGPT.
- Quote from Vaswani et al. (2017): *“Attention is all you need.”* → captured the core idea that **we don’t need recurrence anymore**.

🔹 Timeline of Key Milestones

- **1997 → LSTM introduced**: better memory but still sequential.
- **2014 → Seq2Seq + Attention**: machine translation breakthrough.
- **2017 → Transformers introduced**: attention-only model.
- **2018 → BERT, GPT-1 released**: NLP benchmark dominance.
- **2020 → GPT-3, Vision Transformer (ViT)**: scaling beyond text.

- **2023–24 → GPT-4, multimodal transformers (CLIP, Whisper, GPT-4V)**: AI across text, images, audio.



2. Attention

What Do We Mean by "Attention"?

When humans read or listen, we do not process every word with equal importance. Instead, we **focus attention** on the most relevant parts of the information.

Example:
Sentence → *"The cat sat on the mat because it was tired."*
If asked *"Who was tired?"*, your brain automatically connects **"it"** → **"cat"**, not "mat".

This same idea is applied in machine learning:
Attention is a mechanism that allows a model to focus on the most important parts of the input when making predictions.

The Query, Key, and Value (Q, K, V) System

To understand attention, think of how a **search engine** works:

- **Query (Q):** What you are searching for.
- **Key (K):** The labels attached to all documents.
- **Value (V):** The actual documents themselves.

The search engine calculates how similar the query is to each key, and based on that, it retrieves the relevant values.

In transformers:

- Each word in a sentence is represented as Q, K, and V vectors.
- Attention finds how much one word should “attend” to other words.

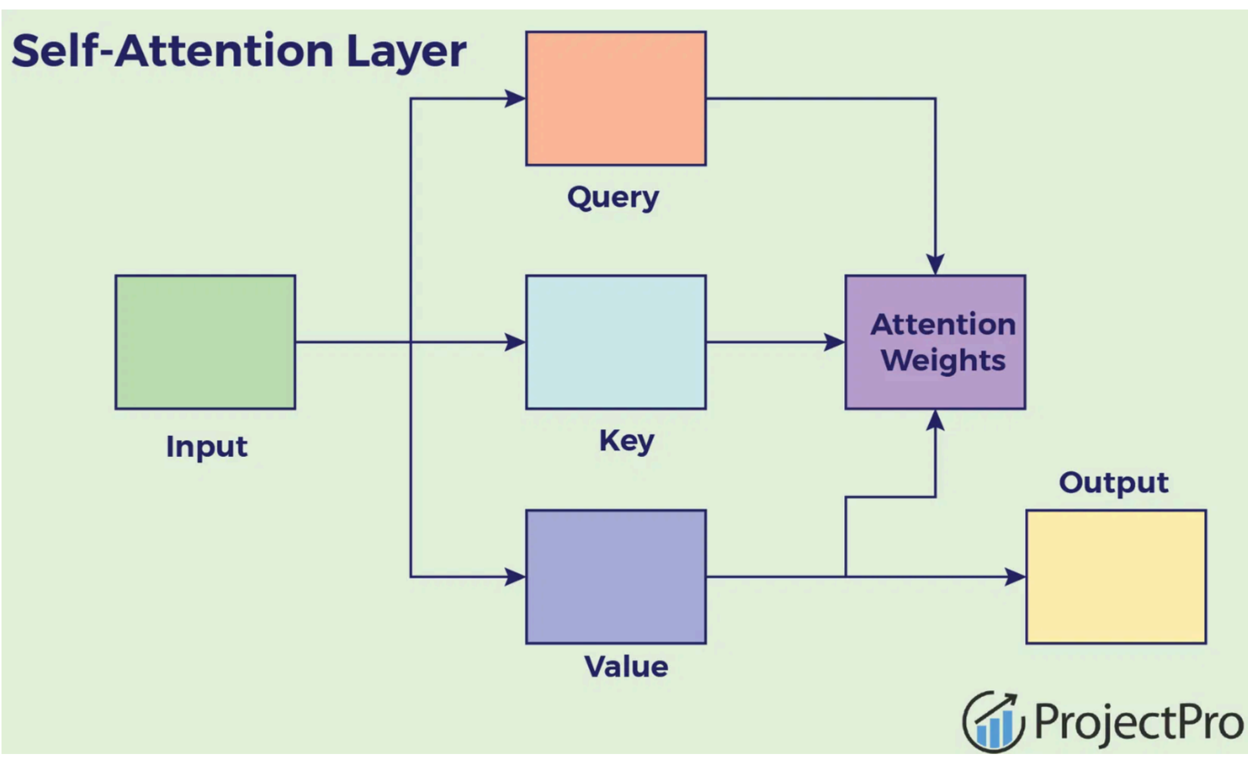
Self-Attention: The Key Innovation

In transformers, attention is applied **within the same sentence**.
This is called **self-attention**.

Example:
Sentence → *“The bank can guarantee deposits will cover future housing loans.”*
The word **“bank”** could mean a *river bank* or a *financial institution*.

- By attending to the words “deposits” and “loans”, the model understands that here **bank = financial institution**.

Self-attention gives every word the ability to directly connect with every other word in the sequence.

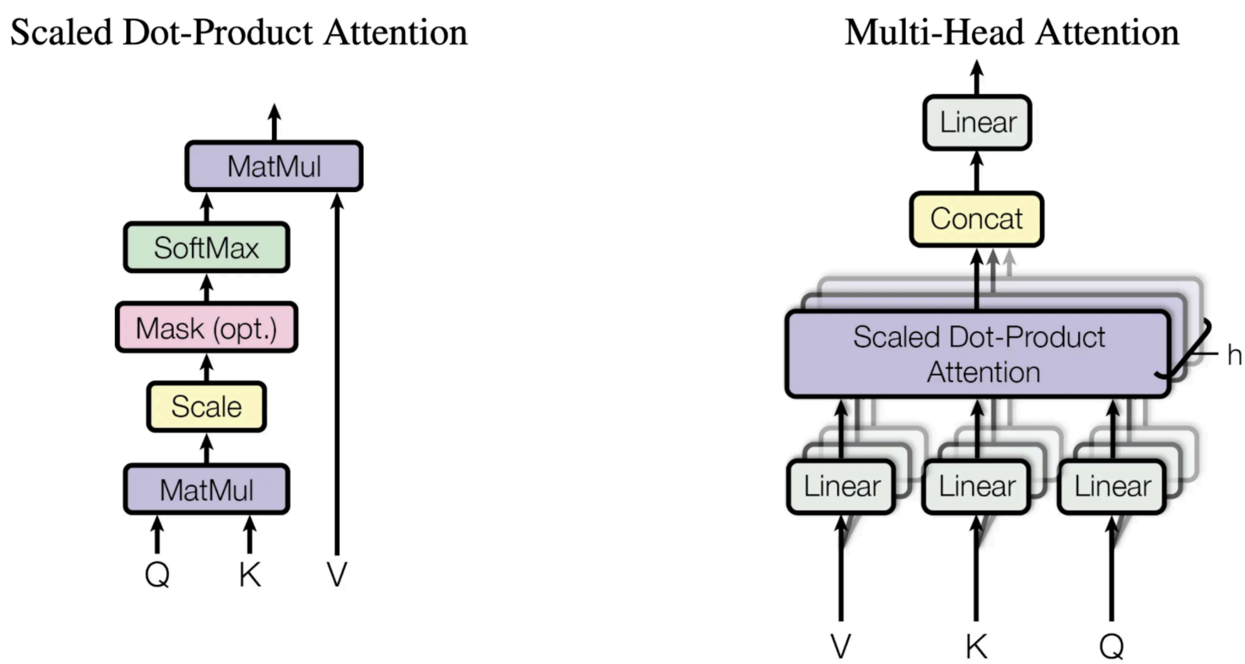


Multi-Head Attention

One “attention calculation” might not be enough. That’s why transformers use **multiple attention heads**.

- Each head learns different kinds of relationships.
- For example:
    - Head 1 → focuses on grammatical structure.
    - Head 2 → focuses on meaning.
    - Head 3 → focuses on long-distance connections.
- The outputs of all heads are combined → richer understanding.

👉 Think of it like asking several experts to analyze a sentence, then combining their insights.

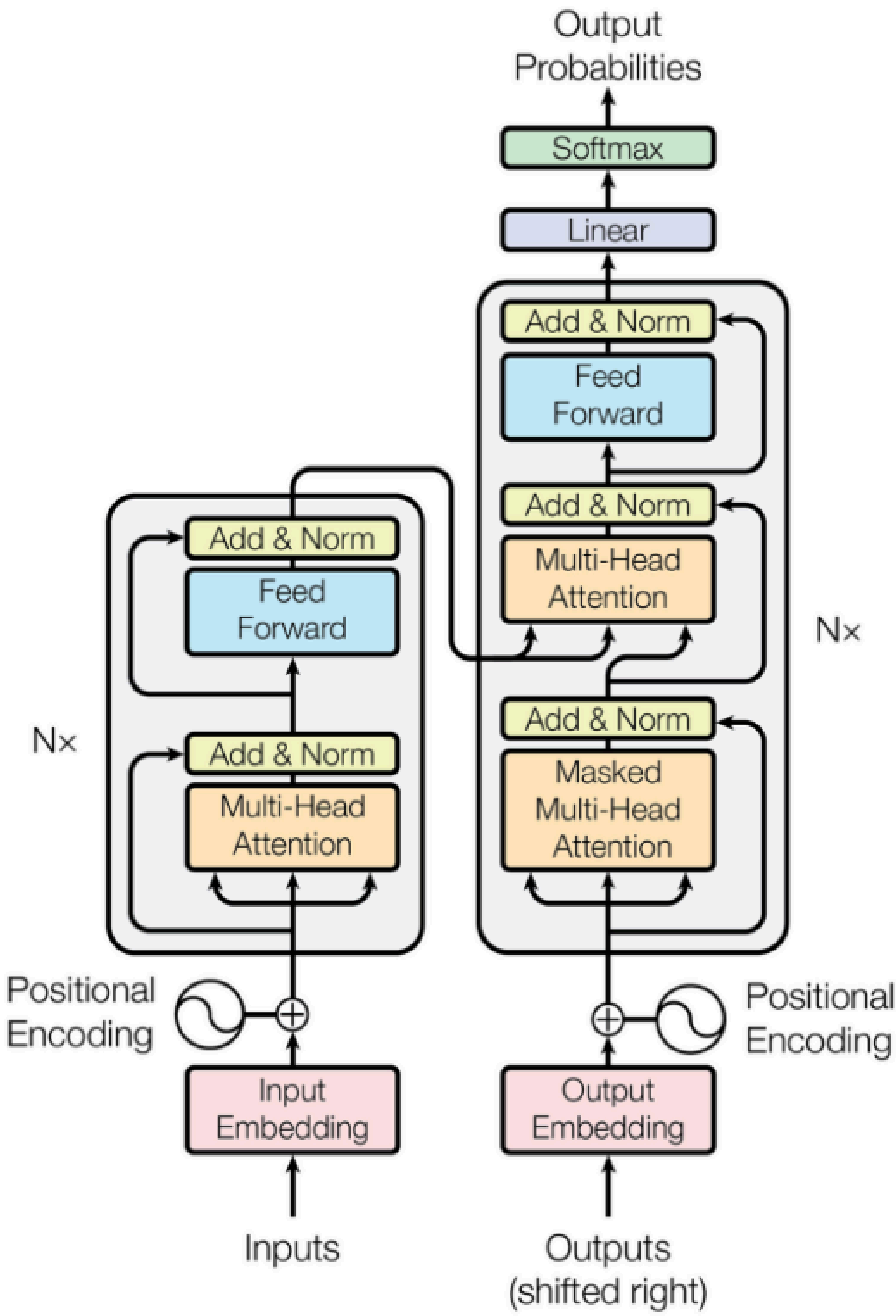


Why Attention is Better than RNNs/LSTMs

- **RNNs/LSTMs:** Pass information step by step → slow + forget long-term dependencies.
- **Attention:** Connects all words directly → fast + remembers distant context.
- **Parallelization:** Unlike RNNs, attention allows training on GPUs for entire sentences at once.

3. Transformer Architecture

The Transformer architecture (from *Attention Is All You Need*, 2017) is the foundation for most modern LLMs.
It's based on **an Encoder–Decoder design**, where both parts are built from layers of attention + feedforward networks.



3.1 Encoder–Decoder Structure

- **Encoder**: Reads the input (e.g., a source sentence in English).

- **Decoder**: Produces the output (e.g., translated sentence in French).
- Information flow:
  Input → Encoder → Encoded Representation → Decoder → Output

Think of the encoder as the "reader" and the decoder as the "writer."

3.2 Encoder (Input Side)

Each encoder block has **3 main parts**:

1. **Multi-Head Self-Attention**
   - Each word in the input looks at all other words.
   - Helps capture context — e.g., in "the animal didn't cross the street because it was too tired," *"it"* can attend to *"animal."*
2. **Feed Forward Network (FFN)**
   - A small 2-layer MLP applied to each token independently.
   - Adds non-linearity and expressive power.
3. **Residual Connections + Layer Normalization**
   - Helps stabilize training.
   - Shortcut connections let gradients flow easily.

3.3 Decoder (Output Side)

Each decoder block has **4 main parts**:

1. **Masked Multi-Head Self-Attention**

    - Prevents "cheating" by ensuring the model only looks at **past words** when generating the next word.

2. **Encoder–Decoder Attention**

    - Aligns input and output.

    - Example: While translating *“chien”* → “dog,” the decoder attends to “chien” in the encoder output.

3. **Feed Forward Network (FFN)** (same as encoder).

4. **Residual Connections + Layer Normalization** (same as encoder).

3.4 Positional Encoding

- Unlike RNNs, Transformers don’t know sequence order naturally.

- To fix this, **positional encodings** (sine/cosine wave patterns) are added to word embeddings.

- These encodings help the model distinguish order:

    - “Cats chase dogs” ≠ “Dogs chase cats.”

3.5 Why It Works (Key Insights)

- **Parallelization**: Unlike RNNs (which process one step at a time), Transformers process **all tokens at once** → much faster training.

- **Scalability**: Easy to stack layers and scale up → GPT, BERT, T5, etc.

- **Context Capture**: Attention allows long-range dependencies (RNNs forget after a few steps).

4. Types of Transformers

After the original **Encoder–Decoder Transformer (2017)**, researchers created specialized variants tailored for different tasks.
These can be grouped into **3 main categories**:

4.1 Encoder-Only Models (Understanding)

- Examples: **BERT, RoBERTa, DistilBERT, ALBERT**
- Goal: Learn deep **representations of text** for understanding tasks.
- Applications:
  - Sentiment Analysis
  - Question Answering
  - Classification tasks
- Mechanism:
  - Use only the **encoder stack**.
  - Learn bidirectional context (word looks at both left & right).

4.2 Decoder-Only Models (Generation)

- Examples: **GPT, GPT-2, GPT-3, GPT-4, LLaMA**
- Goal: Generate coherent text (auto-completion, dialogue, story writing).
- Applications:
  - Chatbots
  - Text completion

    - Code generation (Codex, GPT-4)
- Mechanism:
    - Use only the **decoder stack**.
    - Masked self-attention ensures left-to-right text generation.

4.3 Encoder–Decoder Models (Seq2Seq Tasks)

- Examples: **T5, BART, MarianMT**
- Goal: Transform input text into another text format.
- Applications:
    - Machine Translation
    - Summarization
    - Text-to-text tasks (T5 treats everything as text-in → text-out).
- Mechanism:
    - Full **encoder + decoder stacks**.
    - Encoder compresses input → Decoder generates output.

4.4 Key Insight

- **Encoder-only → Understanding tasks**
- **Decoder-only → Generation tasks**
- **Encoder–Decoder → Transformation tasks**

This classification helps you choose the **right Transformer type** for your problem.

5. Applications of Transformers

5.1 Text Classification (Encoder-only Models like BERT)

Use case: Sentiment Analysis, Spam Detection, Topic Classification.

📌 Example: Sentiment Analysis

```
from transformers import pipeline

classifier = pipeline("sentiment-analysis")
print(classifier("The new iPhone is awesome!"))
# [{'label': 'POSITIVE', 'score': 0.99}]
```

5.2 Text Generation (Decoder-only Models like GPT)

Use case: Chatbots, Story generation, Email writing.

📌 Example: GPT-2 Text Generation

```
from transformers import pipeline

generator = pipeline("text-generation", model="gpt2")
print(generator("Once upon a time", max_length=30, num_return_sequences=1))
```

🔗 Demo: Hugging Face GPT-2 Text Generation

5.3 Translation (Encoder–Decoder Models like MarianMT, T5)

Use case: English → French, Spanish → German, etc.

📌 Example: Translation

```
from transformers import pipeline

translator = pipeline("translation_en_to_fr")
print(translator("Machine learning is fascinating!"))
# [{'translation_text': 'L'apprentissage automatique est fascinant !'}]
```

🔗 Demo: Hugging Face Translation

5.4 Summarization (Encoder–Decoder like BART, T5)

Use case: News summarization, Document compression.

📌 Example: Summarization

```
from transformers import pipeline

summarizer = pipeline("summarization")
print(summarizer("Transformers are a type of neural network... very long text",
                 max_length=50, min_length=20, do_sample=False))
```

🔗 Demo: Hugging Face Summarization

5.5 Beyond Text → Images & Multimodal

- **Vision Transformers (ViT)** → Image classification.
- **CLIP** → Connects images + text.

- **Whisper** → Speech-to-text.