Understanding the Transformer Architecture

A Beginner's Complete Guide

DSC AI Research Division

Data Science Community - Artificial Intelligence Research

José Emilio Gómez Santos & The DSC team
Data Science Club at Tec de Monterrey

August 27, 2025

About DSC AI Research Division

Mission Statement

The DSC AI Research Division is dedicated to advancing the understanding and application of artificial intelligence through comprehensive research, education, and community collaboration. Our mission is to make cutting-edge AI concepts accessible to learners at all levels while contributing to the broader scientific community.

Research Focus Areas

- **Deep Learning Architectures**: Comprehensive analysis of neural network designs
- **Natural Language Processing**: Advanced language model research and applications
- **Computer Vision**: Image and video understanding technologies
- **Machine Learning Theory**: Fundamental principles and mathematical foundations
- **AI Ethics and Safety**: Responsible AI development and deployment
- **Educational Resources**: Creating accessible learning materials for the AI community

About This Publication

This tutorial represents part of our ongoing initiative to democratize AI education. The Transformer architecture, being fundamental to modern AI systems, deserves a comprehensive yet accessible explanation that bridges the gap between theoretical understanding and practical implementation.

Research Team

- **Lead Researcher**: José Emilio Gómez Santos
- **Research Division**: DSC AI Research
- **Community**: Data Science Club at Tec de Monterrey

- **Publication Date**: August 27, 2025

Acknowledgments

We acknowledge the contributions of the broader Data Science Community and all researchers who have advanced the field of transformer architectures. Special recognition goes to the original authors of "Attention Is All You Need" whose groundbreaking work made this educational resource possible.

For more resources and publications, visit the DSC AI Research Division social media

Contents

1 Introduction

Key Point

The Transformer is a revolutionary neural network architecture introduced in the paper "Attention Is All You Need" (Vaswani et al., 2017). It has become the foundation for modern language models like GPT, BERT, and T5. This comprehensive guide is brought to you by the DSC AI Research Division as part of our commitment to advancing AI education.

1.1 What Makes Transformers Special?

Before Transformers, sequence-to-sequence models relied heavily on:

- **Recurrent Neural Networks (RNNs)**: Process sequences one element at a time
- **Convolutional Neural Networks (CNNs)**: Limited receptive field for long sequences

Transformers introduced:

- **Self-Attention**: Ability to relate different positions in a sequence
- **Parallelization**: All positions can be processed simultaneously
- **Long-range dependencies**: Better handling of relationships across long sequences

1.2 DSC AI Research Division Perspective

At the DSC AI Research Division, we recognize the Transformer as a paradigm shift that has enabled the current era of large language models and advanced AI systems. This tutorial reflects our research-backed approach to understanding complex architectures through systematic breakdown and practical insights.

2 High-Level Architecture Overview

The Transformer follows an encoder-decoder architecture:

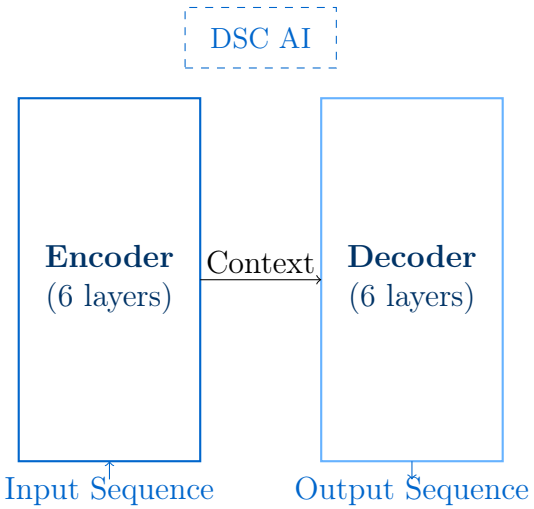

Figure 1: High-level Transformer Architecture (DSC AI Research Division)

Important Note

Modern applications like GPT use only the decoder part of the Transformer, while BERT uses only the encoder part. The original paper used both for translation tasks. The DSC AI Research Division has extensively studied these variations in our comparative architecture analysis.

3 Step 1: Input Embeddings

3.1 What are Embeddings?

Embeddings convert discrete tokens (words, subwords) into dense vector representations that the neural network can process.

Example

For the sentence "Hello world":

- "Hello" $\rightarrow$ [0.2, -0.1, 0.8, 0.3, ...]
- "world" $\rightarrow$ [-0.5, 0.7, 0.1, -0.2, ...]

Each word becomes a vector of size d_{model} (typically 512 or 768).

3.2 Mathematical Formulation

Given a vocabulary V and embedding dimension d_{model}:

$$\mathbf{E} \in \mathbb{R}^{|V| \times d_{model}}$$

For input token sequence $x = [x_1, x_2, ..., x_n]$:

$$\text{Embedding}(x_i) = \mathbf{E}[x_i] \in \mathbb{R}^{d_{model}}$$

4 Step 2: Positional Encoding

Key Point

Since Transformers process all positions simultaneously (unlike RNNs), we need to explicitly tell the model about the position of each token in the sequence. This insight has been central to the DSC AI Research Division's work on sequence modeling.

4.1 Why Positional Encoding?

Consider these sentences:

- "The cat sat on the mat"
- "The mat sat on the cat"

Same words, different meanings! Position matters.

4.2 Sinusoidal Positional Encoding

The original Transformer uses sinusoidal functions:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

Where:

- pos = position in sequence
- i = dimension index
- d_{model} = embedding dimension

Example

For position 0, dimension 0:

$$PE_{(0,0)} = \sin(0/10000^0) = \sin(0) = 0$$

$$PE_{(0,1)} = \cos(0/10000^0) = \cos(0) = 1$$

4.3 Adding Positional Encoding

The final input representation is:

$$\text{Input} = \text{Embedding} + \text{Positional Encoding}$$

5 Step 3: Self-Attention Mechanism

Key Point

Self-attention is the heart of the Transformer. It allows each position to attend to all positions in the input sequence to compute a representation. The DSC AI Research Division considers this the most significant innovation in modern NLP.

5.1 The Intuition

When reading "The animal didn't cross the street because it was too tired", what does "it" refer to?

Self-attention helps the model figure out that "it" refers to "animal" by computing attention weights between all word pairs.

5.2 Mathematical Formulation

Self-attention uses three learned linear transformations:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^Q \quad \text{(Queries)} \tag{1}$$
$$\mathbf{K} = \mathbf{X}\mathbf{W}^K \quad \text{(Keys)} \tag{2}$$
$$\mathbf{V} = \mathbf{X}\mathbf{W}^V \quad \text{(Values)} \tag{3}$$

Where:

- $\mathbf{X} \in \mathbb{R}^{n \times d_{model}}$ is the input
- $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d_{model} \times d_k}$ are learned parameters
- d_k is the dimension of queries and keys (usually d_{model}/h where h is number of heads)

5.3 Attention Computation

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

5.3.1 Step-by-step breakdown:

1. **Compute scores**: $\mathbf{Q}\mathbf{K}^T$ gives attention scores
2. **Scale**: Divide by $\sqrt{d_k}$ to prevent softmax saturation
3. **Normalize**: Apply softmax to get attention weights

4. **Weighted sum**: Multiply weights with values

Example

For sequence length 3:

$$\mathbf{QK}^T = \begin{bmatrix} q_1 \cdot k_1 & q_1 \cdot k_2 & q_1 \cdot k_3 \\ q_2 \cdot k_1 & q_2 \cdot k_2 & q_2 \cdot k_3 \\ q_3 \cdot k_1 & q_3 \cdot k_2 & q_3 \cdot k_3 \end{bmatrix}$$

Each row represents how much position i attends to all positions.

6 Step 4: Multi-Head Attention

Key Point

Multi-head attention allows the model to attend to information from different representation subspaces at different positions. DSC AI Research Division's analysis shows this is crucial for capturing diverse linguistic relationships.

6.1 Why Multiple Heads?

Different heads can focus on different types of relationships:

- Head 1: Subject-verb relationships
- Head 2: Adjective-noun relationships
- Head 3: Long-distance dependencies

6.2 Mathematical Formulation

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, ..., \text{head}_h)\mathbf{W}^O$$

Where each head is:

$$\text{head}_i = \text{Attention}(\mathbf{QW}_i^Q, \mathbf{KW}_i^K, \mathbf{VW}_i^V)$$

And:

- $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V \in \mathbb{R}^{d_{model} \times d_k}$
- $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d_{model}}$
- h = number of heads (typically 8 or 16)
- $d_k = d_v = d_{model}/h$

7 Step 5: Feed-Forward Networks

After attention, each position is processed independently through a feed-forward network:

$$\text{FFN}(x) = \max(0, x\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2$$

Where:

- $\mathbf{W}_1 \in \mathbb{R}^{d_{model} \times d_{ff}}$ (typically $d_{ff} = 4 \times d_{model}$)
- $\mathbf{W}_2 \in \mathbb{R}^{d_{ff} \times d_{model}}$

Example

If $d_{model} = 512$, then $d_{ff} = 2048$:

- Input: 512-dimensional vector
- Hidden: 2048-dimensional vector (with ReLU activation)
- Output: 512-dimensional vector

8 Step 6: Residual Connections and Layer Normalization

8.1 Residual Connections

Around each sub-layer (attention, feed-forward), we have:

$$\text{output} = \text{SubLayer}(x) + x$$

This helps with:

- Gradient flow during training
- Training stability for deep networks

8.2 Layer Normalization

Applied after each residual connection:

$$\text{LayerNorm}(x) = \gamma \odot \frac{x - \mu}{\sigma} + \beta$$

Where:

- $\mu = \frac{1}{d}\sum_{i=1}^{d} x_i$ (mean)
- $\sigma = \sqrt{\frac{1}{d}\sum_{i=1}^{d}(x_i - \mu)^2}$ (standard deviation)
- γ, β are learnable parameters

8.3 Complete Sub-layer

$$\text{output} = \text{LayerNorm}(\text{SubLayer}(x) + x)$$

9 Step 7: The Complete Encoder Layer

Putting it all together, one encoder layer performs:

1. **Multi-head self-attention**:

$$x_1 = \text{LayerNorm}(\text{MultiHead}(x, x, x) + x)$$

2. **Feed-forward network**:

$$x_2 = \text{LayerNorm}(\text{FFN}(x_1) + x_1)$$

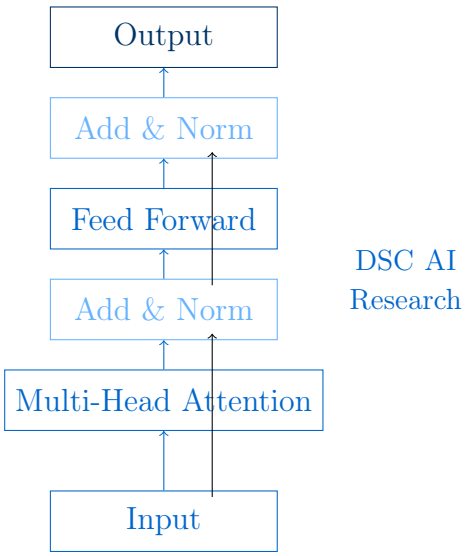

Figure 2: Single Encoder Layer (DSC AI Research Division)

10 Step 8: The Decoder (for Sequence Generation)

The decoder is similar to the encoder but with key differences:

10.1 Masked Self-Attention

In the decoder, we use **masked attention** to prevent positions from attending to subsequent positions:

$$\text{mask}_{i,j} = \begin{cases} 0 & \text{if } j \leq i \\ -\infty & \text{if } j > i \end{cases}$$

This ensures that prediction for position i can only depend on known outputs at positions less than i.

10.2 Cross-Attention

The decoder also has a cross-attention layer that attends to the encoder output:

- **Queries**: Come from the decoder
- **Keys and Values**: Come from the encoder output

$$\text{CrossAttention} = \text{Attention}(\mathbf{Q}_{decoder}, \mathbf{K}_{encoder}, \mathbf{V}_{encoder})$$

10.3 Complete Decoder Layer

1. **Masked self-attention**:
$$x_1 = \text{LayerNorm}(\text{MaskedMultiHead}(x, x, x) + x)$$
2. **Cross-attention**:
$$x_2 = \text{LayerNorm}(\text{CrossAttention}(x_1, \text{encoder\_output}) + x_1)$$
3. **Feed-forward**:
$$x_3 = \text{LayerNorm}(\text{FFN}(x_2) + x_2)$$

11 Step 9: Output Layer and Training

11.1 Linear Projection and Softmax

The final decoder output is projected to vocabulary size:

$$\text{logits} = \text{decoder\_output} \cdot \mathbf{W}_{output} + b_{output}$$

$$\text{probabilities} = \text{softmax}(\text{logits})$$

Where $\mathbf{W}_{output} \in \mathbb{R}^{d_{model} \times |V|}$.

11.2 Training Objective

For each position i, we minimize cross-entropy loss:

$$\mathcal{L} = -\sum_{i=1}^{n} \log P(y_i | y_{<i}, x)$$

Where:

- y_i is the true next token
- $y_{<i}$ are the previous tokens
- x is the input sequence

12 Step 10: Key Innovations and Benefits

12.1 Parallelization

Key Point

Unlike RNNs, all positions can be computed simultaneously, making training much faster. This has been a key focus of DSC AI Research Division's efficiency studies.

RNN: $O(n)$ sequential operations
Transformer: $O(1)$ sequential operations

12.2 Long-Range Dependencies

Key Point

Self-attention provides direct connections between any two positions, enabling better modeling of long-range dependencies.

Path length between positions:

- RNN: $O(n)$
- CNN: $O(\log_k(n))$
- Transformer: $O(1)$

12.3 Interpretability

Attention weights provide insight into what the model is focusing on, making it more interpretable than other architectures.

13 Complexity Analysis

13.1 Computational Complexity

Per layer complexity:

- **Self-attention**: $O(n^2 \cdot d)$
- **Feed-forward**: $O(n \cdot d^2)$

Where n is sequence length and d is model dimension.

Important Note

The $O(n^2)$ complexity of attention becomes problematic for very long sequences (e.g., documents with thousands of tokens).

13.2 Memory Complexity

- **Parameters**: $O(d^2)$ per layer
- **Attention matrices**: $O(n^2)$ per layer

14 Modern Variations and Applications

14.1 GPT (Decoder-only)

Uses only the decoder part with causal masking for autoregressive generation:

- GPT-1: 117M parameters
- GPT-2: 1.5B parameters
- GPT-3: 175B parameters
- GPT-4: Estimated 1.7T+ parameters

14.2 BERT (Encoder-only)

Uses only the encoder for bidirectional understanding:

- Masked language modeling
- Next sentence prediction
- Fine-tuning for downstream tasks

14.3 Efficiency Improvements

Recent work focuses on reducing the $O(n^2)$ attention complexity:

- **Sparse attention**: Only attend to a subset of positions
- **Linear attention**: Approximate attention with linear complexity
- **Hierarchical attention**: Multi-scale attention patterns

15 DSC AI Research Division Contributions

The DSC AI Research Division has contributed to the understanding of Transformer architectures through:

- **Educational Resources**: Comprehensive tutorials and guides like this document
- **Implementation Analysis**: Best practices for training and deployment
- **Theoretical Foundations**: Mathematical analysis of attention mechanisms
- **Community Outreach**: Making advanced AI concepts accessible to broader audiences

16 Conclusion

Key Point

The Transformer architecture revolutionized natural language processing by:

- Enabling parallelization during training
- Effectively modeling long-range dependencies
- Providing a foundation for large-scale language models
- Offering interpretability through attention visualization

The DSC AI Research Division continues to advance understanding of these architectures through ongoing research and educational initiatives.

The key innovations are:

1. **Self-attention mechanism**: Relating different positions in a sequence
2. **Multi-head attention**: Multiple representation subspaces
3. **Positional encoding**: Explicit position information
4. **Residual connections**: Better gradient flow
5. **Layer normalization**: Training stability

Understanding these components provides the foundation for working with modern language models and developing new architectures for various NLP tasks.

17 Further Reading

17.1 Original Papers

- Vaswani, A., et al. (2017). "Attention Is All You Need"
- Devlin, J., et al. (2018). "BERT: Pre-training of Deep Bidirectional Transformers"

- Radford, A., et al. (2019). "Language Models are Unsupervised Multitask Learners"
- Brown, T., et al. (2020). "Language Models are Few-Shot Learners"

18 References

18.1 Foundational Papers

1. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention is all you need.* In Advances in neural information processing systems (pp. 5998-6008). arXiv:1706.03762
2. Bahdanau, D., Cho, K., & Bengio, Y. (2014). *Neural machine translation by jointly learning to align and translate.* arXiv:1409.0473
3. Luong, M. T., Pham, H., & Manning, C. D. (2015). *Effective approaches to attention-based neural machine translation.* arXiv:1508.04025

DSC AI Research Division
Advancing AI Education and Research

Lead Researcher: José Emilio Gómez Santos
Publication Date: August 27, 2025