Explainable Transformers via Graph-Based Operator Notation

A Beginner-Oriented, Concept-Only Guide

Anonymous

Abstract

We present a compact, pedagogy-first description of Transformers using a graph-based operator notation. Nodes denote operators, edges denote tensors, and double arrows mark the second operand of matrix multiplications. This article focuses on concepts (no code), pairing each formula with shape reasoning so a newcomer can translate diagrams into implementations after reading.[1]

Keywords Transformer; attention; backpropagation; LayerNorm; softmax; diagram notation; pedagogy.

1 Introduction

Transformers have become the dominant sequence model in modern ML. However, many introductions assume prior exposure to neural networks and autodiff. We provide a beginner-friendly, concept-only path: (i) a minimal diagram legend, (ii) operator primitives with forward/backward summaries, (iii) multi-head attention (MHA) forward/backward, and (iv) assembly of a Transformer block (Input Embedding $\rightarrow$ MHA $\rightarrow$ MLP $\rightarrow$ Output Projection).

1.1 Contributions

(1) A consistent diagrammatic convention that separates operators from data. (2) Shape-first backprop summaries, including broadcasting and reduce axes. (3) A compact Transformer walkthrough suitable for first-time readers.

2 Diagram Legend

Nodes: $\bullet$ (MatMul), $\oplus$ (elementwise add), yellow rectangles (nonlinear / non-invertible, e.g., LN, Softmax, Dropout).
Edges: single arrow = dataflow; double arrow = second MatMul operand.
Helpers: R (reshape/split-merge heads), T (transpose), C (concat), DO (dropout), S (scale/softmax), SM (masking).

3 Operator Primitives

3.1 General Backprop Rule for $z = f(x, y)$

Let $g = \partial L/\partial z$. Then

$$\frac{\partial L}{\partial x} = \texttt{reduce\_like}\big(g \odot \tfrac{\partial f}{\partial x},\ x\big), \quad \frac{\partial L}{\partial y} = \texttt{reduce\_like}\big(g \odot \tfrac{\partial f}{\partial y},\ y\big), \tag{1}$$

where `reduce_like` sums over broadcast axes to match the input shape. For MatMul $Z = AB$ with upstream G, $\partial L/\partial A = GB^\top$ and $\partial L/\partial B = A^\top G$.

[1] A practical, code-first companion is intentionally excluded in this version.

3.2 Elementwise Add / Bias Add

$Y = A + B$. Gradients reduce along broadcast axes.

3.3 Linear / Projection

$Y = XW + b$. $\partial L/\partial X = (\partial L/\partial Y)W^\top$, $\partial L/\partial W = X^\top(\partial L/\partial Y)$, $\partial L/\partial b = \sum_{B,S}(\partial L/\partial Y)$.

3.4 Softmax (Stable)

With $Y = \mathrm{softmax}(Z)$ along the last axis, $\partial L/\partial Z = (G - \langle G, Y\rangle) \odot Y$, where $G = \partial L/\partial Y$ and the inner product is along the softmax axis.

3.5 LayerNorm (LN)

Summarize forward $(\mu, \sigma, \hat{X})$ and use the standard derivative form with cached statistics.

4 Multi-Head Attention (MHA)

4.1 Forward

Given $Q, K, V \in \mathbb{R}^{[B,H,S,D_h]}$, scores $A = QK^\top/\sqrt{D_h}$, probabilities $P = \mathrm{softmax}(A)$ (with masking), outputs $O = PV$. Heads are merged (concat) and projected. Shapes: $A, P \in \mathbb{R}^{[B,H,S,S]}$, $O \in \mathbb{R}^{[B,S,D]}$.

4.2 Backward (Summary)

$dO \to dV = P^\top dO$, $dO \to dP = dO\, V^\top$, $dP \to dA = (dP - \mathrm{sum}(dP \odot P)) \odot P$, masking zeros gradients on masked entries, and $dA \to dQ = dA\, K/\sqrt{D_h}$, $dA \to dK = dA^\top Q/\sqrt{D_h}$.

5 Transformer Block (Concept-Only)

Pipeline: Input Embedding → MHA → MLP (FFN) → Output Projection, with residual connections and LayerNorm.
Input Embedding: token table $E \in \mathbb{R}^{V\times D}$ mapping $[B, S]$ to $[B, S, D]$; positional signal (absolute/learned or RoPE).
MLP: Linear–GELU–Linear (or SwiGLU), expansion factor α.
Output Projection: logits via $W_{lm} \in \mathbb{R}^{D\times V}$; optionally tie with E.

6 Figures

Figure 1: Forward pass of multi-head attention using the proposed legend.

7 Discussion and Limitations

Scope is conceptual; we omit code, datasets, and training details. Numerical stability notes (softmax, LN) are summarized rather than proven.

8 Conclusion

We provided a concise, beginner-first exposition of Transformers grounded in operator graphs and shape reasoning. The framework is intended to be directly translatable to implementations once readers are ready to code.