Transformer Architecture Execution Flow

This document explains the execution flow of our NumPy-based transformer implementation, which supports multiple NLP tasks through task-specific heads. The architecture follows the original "Attention Is All You Need" paper while adding flexibility for various tasks.

Core Components

Component	Purpose	Key Features
Embeddings	Convert tokens to vectors	Token + Positional encoding
Encoder	Process input sequences	Self-attention + Feed-forward
Decoder	Generate output sequences	Self-attention + Cross-attention
Task Heads	Task-specific predictions	Classification/Generation/QA/etc.

Execution Flow

1. Input Processing: - Tokenize input text - Convert tokens to embeddings - Add positional encoding 2. Encoder Processing: - Self-attention mechanism - Feed-forward network - Layer normalization - Residual connections 3. Decoder Processing (for generation tasks): - Self-attention with causal masking - Cross-attention with encoder output - Feed-forward network - Layer normalization - Residual connections 4. Task-Specific Head: - Classification: Mean pooling + linear projection - Sequence Labeling: Token-level predictions - Generation: Next token prediction - Question Answering: Start/end position prediction

Task-Specific Implementations

Task	Input	Output	Loss Function
Classification	Sequence	Class probabilities	Cross-entropy
Sequence Labeling	Sequence	Token labels	Token-wise cross-entropy
Generation	Source sequence	Target sequence	Sequence cross-entropy
Question Answering	Context + Question	Start/end positions	Position cross-entropy

Hardware Considerations

The current NumPy implementation is hardware-agnostic but can be optimized for specific hardware: 1. GPU Acceleration: - Replace NumPy with CUDA-compatible libraries - Use GPU-optimized operations - Implement custom CUDA kernels 2. TPU/ASIC Acceleration: - Use JAX or TensorFlow with TPU support - Optimize for TPU matrix multiplication - Handle TPU memory management 3. CPU Optimization: - Use BLAS/LAPACK optimizations - Implement SIMD instructions - Add multi-threading support 4. Edge Devices: - Implement model quantization - Apply model pruning - Add hardware-specific optimizations