Attention Mechanisms and Transformers

September 28, 2025

1 Motivation and Formulation of Attention

Attention mechanisms allow models to focus on relevant parts of the input when producing outputs. Given queries $\mathbf{Q}$, keys $\mathbf{K}$, and values $\mathbf{V}$, the additive attention score between a query $\mathbf{q}_i$ and key $\mathbf{k}_j$ is

$$e_{ij} = \mathbf{v}^\top \tanh(\mathbf{W}_q \mathbf{q}_i + \mathbf{W}_k \mathbf{k}_j), \tag{1}$$

while scaled dot-product attention simplifies the score to

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}. \tag{2}$$

The softmax weights act as adaptive alignment coefficients. Figure **??** illustrates how attention redistributes focus across sequence elements.

1.1 Alignment and Context Vectors

For sequence-to-sequence tasks, attention forms context vectors $\mathbf{c}_i = \sum_j \alpha_{ij} \mathbf{v}_j$ with α_{ij} from the softmax weights. Attention improves convergence, mitigates information bottlenecks, and handles variable-length inputs.

1.2 Variations

Common variants include additive (Bahdanau) attention, multiplicative (Luong) attention, and location-based attention. Monotonic attention enforces ordered alignments for speech recognition. Sparse and hard attention restrict selections to top-k elements for efficiency.

2 Self-Attention and Multi-Head Attention

Self-attention treats the same sequence as queries, keys, and values, enabling long-range dependencies without recurrence. Multi-head attention (MHA) projects inputs into h subspaces and attends in parallel:

$$\text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V), \tag{3}$$

$$\text{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}^O. \tag{4}$$

Multi-heads capture diverse relational patterns (e.g., syntax, positional cues). Figure **??** visualizes head-specific attention maps.

2.1 Positional Encoding

Because self-attention is permutation-invariant, transformers add positional encodings. Sinusoidal encodings use fixed frequencies:

$$\mathrm{PE}_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\mathrm{model}}}}\right), \tag{5}$$

$$\mathrm{PE}_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\mathrm{model}}}}\right). \tag{6}$$

Learned positional embeddings or rotary positional embeddings (RoPE) adapt positions during training and improve extrapolation.

2.2 Efficiency Considerations

Self-attention has $\mathcal{O}(n^2)$ complexity. Sparse, local, or linear attention variants (e.g., Performer, Linformer) reduce cost for long sequences by approximating attention weights or restricting receptive fields.

3 Transformer Architecture

The transformer encoder-decoder architecture comprises stacked layers with MHA and feed-forward networks (FFNs). Each encoder layer performs:

$$\mathbf{z} = \mathrm{LayerNorm}(\mathbf{x} + \mathrm{MHA}(\mathbf{x})), \tag{7}$$

$$\mathbf{y} = \mathrm{LayerNorm}(\mathbf{z} + \mathrm{FFN}(\mathbf{z})), \tag{8}$$

where FFN is typically a two-layer MLP with ReLU or GELU activation. Decoder layers include masked self-attention and cross-attention to the encoder outputs. Residual connections and layer normalization stabilize deep stacks.

3.1 Feed-Forward Networks

The position-wise FFN expands dimensionality (e.g., $d_{\mathrm{model}} = 512$ to 2048) before projecting back. Variants like gated linear units (GLU), SwiGLU, or depthwise convolutions enhance expressive power. Dropout and stochastic depth regularize training.

3.2 Training Strategies

Transformers rely on large-scale data and parallel computation. Techniques include label smoothing, warmup schedules, adaptive optimizers (Adam, Adafactor), gradient accumulation, and mixed-precision training.

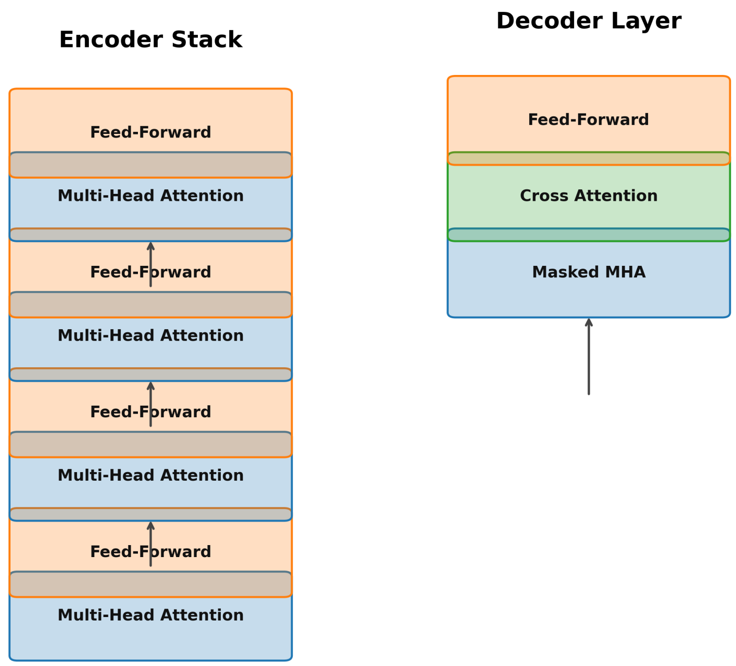

Figure 1: Transformer encoder-decoder stack with self-attention, cross-attention, and position-wise feed-forward networks.

4 Pretrained Language Models: BERT and GPT

Pretrained transformers revolutionized NLP by learning contextual representations.

4.1 BERT

Bidirectional Encoder Representations from Transformers (BERT) pretrains using masked language modeling (MLM) and next sentence prediction (NSP). The model uses only the encoder stack with bidirectional self-attention, enabling deep contextual understanding. Fine-tuning BERT for downstream tasks requires minimal adaptation (task-specific heads atop [CLS]).

4.2 GPT Series

Generative Pretrained Transformers (GPT) employ decoder-only architectures trained with causal language modeling. Autoregressive training predicts the next token given past context. Scaling laws reveal predictable improvements with larger models and data. GPT-2/3 introduced zero-shot and few-shot prompting; GPT-4 incorporated multi-modal inputs and tool integration.

4.3 Comparison and Extensions

BERT excels at understanding tasks (classification, QA), whereas GPT excels at generation. Hybrid models (T5, BART) unify encoder-decoder training objectives. Instruction tuning, reinforcement learning from human feedback (RLHF), and retrieval augmentation further enhance performance.

5 Applications in NLP and Cross-Modal Learning

Attention and transformers underpin modern AI systems.

5.1 Natural Language Processing

Transformers dominate machine translation, summarization, sentiment analysis, and question answering. Pretrained encoders provide embeddings for information retrieval; sequence-to-sequence transformers power neural machine translation and abstractive summarization.

5.2 Cross-Modal and Multimodal Learning

Vision transformers (ViT) apply self-attention to image patches. CLIP aligns text and images via contrastive pretraining. Video transformers model spatio-temporal dependencies. Audio transformers handle speech recognition and music generation. Multimodal large language models integrate vision, text, and audio for grounded reasoning.

5.3 Knowledge Integration

Retrieval-augmented models (RAG), memory-augmented transformers, and adapters incorporate external knowledge bases. Prompt tuning and LoRA (low-rank adaptation) provide parameter-efficient fine-tuning.

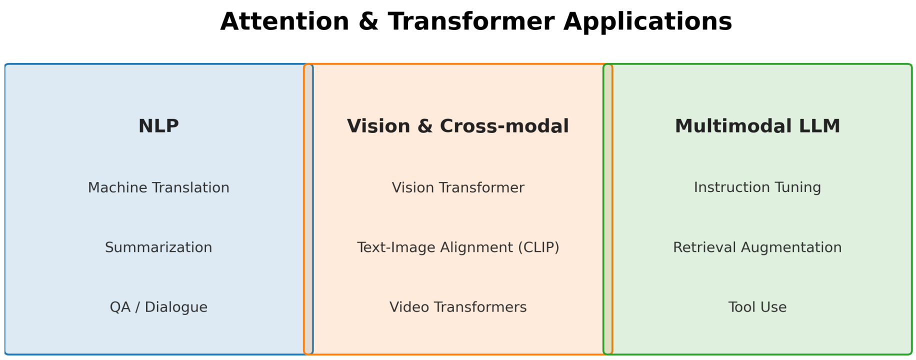

Figure 2: Applications of attention and transformers in NLP and multimodal settings.

6 Practical Tips

- **Memory management:** Use gradient checkpointing, mixed precision, or sequence chunking to handle long sequences.
- **Regularization:** Apply dropout, attention dropout, label smoothing, and weight decay to prevent overfitting.
- **Scaling:** Monitor loss scaling laws; adjust batch size, learning rate schedule (cosine with warmup), and gradient clipping.
- **Evaluation:** Track task-specific metrics (BLEU, ROUGE, accuracy) and perplexity. Analyze attention maps and calibration.
- **Deployment:** Distill large models into smaller students, quantize weights, or apply sparse attention for latency-sensitive applications.