Transformers: Comprehensive Summary

Mathematical Notes Collection

October 28, 2025

Contents

1 Introduction to Transformers

The Transformer architecture, introduced in "Attention Is All You Need" (Vaswani et al., 2017), revolutionized natural language processing and deep learning. Unlike previous sequence-to-sequence models that relied on recurrent or convolutional layers, Transformers use only attention mechanisms to process sequential data.

1.1 Key Innovation

Definition 1.1 (Transformer). *A Transformer is a neural network architecture based entirely on attention mechanisms, dispensing with recurrence and convolutions entirely. It consists of an encoder-decoder structure where both components are stacks of identical layers.*

1.2 Core Components

1. **Multi-Head Attention**: Parallel attention mechanisms
2. **Position Encoding**: Injecting positional information
3. **Feed-Forward Networks**: Point-wise fully connected layers
4. **Residual Connections**: Skip connections for gradient flow
5. **Layer Normalization**: Stabilizing training

2 Attention Mechanism

2.1 Self-Attention

Self-attention allows each position in a sequence to attend to all positions in the same sequence:

Definition 2.1 (Self-Attention). *Given input sequence $X = (x_1, x_2, \ldots, x_n)$, self-attention computes:*

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

where $Q = XW_Q$, $K = XW_K$, $V = XW_V$ are query, key, and value matrices.

2.2 Scaled Dot-Product Attention

Algorithm 1 Scaled Dot-Product Attention

```
1: Input: Queries $Q$, Keys $K$, Values $V$
2: Compute attention scores: $S = QK^T$
3: Scale by $\sqrt{d_k}$: $S = S/\sqrt{d_k}$
4: Apply softmax: $A = \text{softmax}(S)$
5: Compute output: $\text{Attention}(Q, K, V) = AV$
```

Mathematical Formulation:

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

2.3 Multi-Head Attention

Multi-head attention allows the model to jointly attend to information from different representation subspaces:

Definition 2.2 (Multi-Head Attention).

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O$$

where $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$

Algorithm 2 Multi-Head Attention

```
1: Input: $Q$, $K$, $V$, number of heads $h$
2: for $i = 1$ to $h$ do
3:     Compute $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$
4: end for
5: Concatenate: $\text{MultiHead} = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$
6: Output: $\text{MultiHead}(Q, K, V)$
```

3 Transformer Architecture

3.1 Encoder-Decoder Structure

Definition 3.1 (Transformer Encoder). *The encoder consists of* $N = 6$ *identical layers. Each layer has two sub-layers:*

1. *Multi-head self-attention mechanism*

2. *Position-wise fully connected feed-forward network*

Each sub-layer has a residual connection and layer normalization.

Definition 3.2 (Transformer Decoder). *The decoder consists of* $N = 6$ *identical layers. Each layer has three sub-layers:*

1. *Masked multi-head self-attention mechanism*

2. *Multi-head attention over encoder outputs*

3. *Position-wise fully connected feed-forward network*

Each sub-layer has a residual connection and layer normalization.

3.2 Position Encoding

Since Transformers have no inherent notion of position, positional encodings are added to input embeddings:

Definition 3.3 (Sinusoidal Position Encoding). *For position pos and dimension i:*

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

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

3.3 Feed-Forward Networks

Each layer contains a position-wise feed-forward network:

Definition 3.4 (Position-wise Feed-Forward).

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

This consists of two linear transformations with a ReLU activation in between.

4 Training and Optimization

4.1 Training Objectives

4.1.1 Language Modeling

Definition 4.1 (Next Token Prediction). *Given a sequence of tokens $(x_1, x_2, \ldots, x_n)$, predict the next token x_{n+1}:*

$$\mathcal{L} = -\sum_{i=1}^{n} \log P(x_{i+1}|x_1, \ldots, x_i)$$

4.1.2 Masked Language Modeling

Definition 4.2 (Masked Language Modeling). *Randomly mask 15% of input tokens and predict the masked tokens:*

$$\mathcal{L} = -\sum_{i \in masked} \log P(x_i|context)$$

4.2 Optimization Techniques

1. **Adam Optimizer**: Adaptive learning rates
2. **Learning Rate Scheduling**: Warmup and decay
3. **Gradient Clipping**: Preventing exploding gradients
4. **Dropout**: Regularization technique
5. **Label Smoothing**: Improving generalization

4.3 Learning Rate Schedule

Definition 4.3 (Warmup Schedule).

$$lr = d_{model}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5})$$

5 Transformer Variants

5.1 BERT (Bidirectional Encoder Representations)

Definition 5.1 (BERT). *BERT uses bidirectional training of Transformers to learn representations that can be fine-tuned for various downstream tasks.*

Key Features:

- Bidirectional context understanding

- Masked language modeling objective
- Next sentence prediction task
- Pre-training on large corpora

5.2 GPT (Generative Pre-trained Transformer)

Definition 5.2 (GPT). *GPT uses a decoder-only Transformer architecture for autoregressive language generation.*

Key Features:

- Autoregressive generation
- Causal attention masking
- Unsupervised pre-training
- Fine-tuning for specific tasks

5.3 T5 (Text-to-Text Transfer Transformer)

Definition 5.3 (T5). *T5 treats every NLP problem as a text-to-text problem, using a unified encoder-decoder architecture.*

5.4 Vision Transformer (ViT)

Definition 5.4 (Vision Transformer). *ViT applies the Transformer architecture directly to image patches, treating them as a sequence of tokens.*

Process:

1. Split image into patches
2. Linear projection to patch embeddings
3. Add positional embeddings
4. Process with Transformer encoder
5. Classification head for final prediction

6 Attention Variants

6.1 Sparse Attention

Definition 6.1 (Sparse Attention). *Sparse attention mechanisms reduce computational complexity by attending to only a subset of positions.*

Types:

- **Local Attention**: Attend only to nearby positions
- **Strided Attention**: Attend to every k-th position
- **Random Attention**: Attend to random positions
- **Block Sparse Attention**: Attend to blocks of positions

6.2 Linear Attention

Definition 6.2 (Linear Attention). *Linear attention reduces quadratic complexity to linear by approximating the softmax operation.*

Formulation:

$$\text{LinearAttention}(Q, K, V) = \frac{Q(K^T V)}{QK^T \mathbf{1}}$$

6.3 Cross-Attention

Definition 6.3 (Cross-Attention). *Cross-attention allows the decoder to attend to encoder outputs:*

$$\mathit{CrossAttention}(Q, K, V) = \mathit{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where Q comes from decoder and K, V from encoder.

7 Positional Encoding Variants

7.1 Learned Positional Embeddings

Definition 7.1 (Learned Positional Embeddings). *Instead of sinusoidal functions, learn position embeddings as parameters:*

$$PE_{pos} = \mathit{Embedding}(pos)$$

7.2 Relative Positional Encoding

Definition 7.2 (Relative Positional Encoding). *Encode relative distances between positions rather than absolute positions:*

$$\mathit{Attention}_{ij} = \frac{(q_i + r_{i-j})^T k_j}{\sqrt{d_k}}$$

where r_{i-j} is the relative position encoding.

7.3 Rotary Position Embedding (RoPE)

Definition 7.3 (RoPE). *RoPE encodes absolute positional information with rotation matrix:*

$$f_{RoPE}(x, m) = \begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

8 Applications

8.1 Natural Language Processing

1. **Machine Translation**: Sequence-to-sequence translation
2. **Text Summarization**: Abstractive summarization
3. **Question Answering**: Reading comprehension tasks
4. **Text Classification**: Sentiment analysis, topic classification
5. **Language Generation**: Creative writing, code generation

8.2 Computer Vision

1. **Image Classification**: Vision Transformers (ViT)
2. **Object Detection**: DETR (Detection Transformer)
3. **Image Generation**: DALL-E, Imagen
4. **Video Understanding**: Video Transformers

8.3 Multimodal Applications

1. **Image Captioning**: Describing images with text
2. **Visual Question Answering**: Answering questions about images
3. **Multimodal Generation**: Creating content across modalities
4. **Cross-Modal Retrieval**: Finding relevant content across modalities

8.4 Scientific Applications

1. **Protein Folding**: AlphaFold uses attention mechanisms
2. **Drug Discovery**: Molecular property prediction
3. **Climate Modeling**: Weather prediction
4. **Astronomy**: Galaxy classification

9 Mathematical Foundations

9.1 Complexity Analysis

Theorem 9.1 (Attention Complexity). *The computational complexity of self-attention is $O(n^2 \cdot d)$ where n is sequence length and d is model dimension.*

Proof. For each attention head, we compute:

- QK^T: $O(n^2 \cdot d)$
- Softmax: $O(n^2)$
- Attention weights $\times V$: $O(n^2 \cdot d)$

Total: $O(n^2 \cdot d)$ □

9.2 Expressiveness

Theorem 9.2 (Universal Approximation). *A Transformer with sufficient depth and width can approximate any continuous function on compact sets.*

9.3 Gradient Flow

Proposition 9.3 (Residual Connection Benefit). *Residual connections help maintain gradient flow in deep Transformer networks by providing direct paths for gradients.*

10 Optimization Techniques

10.1 Mixed Precision Training

Definition 10.1 (Mixed Precision). *Using both 16-bit and 32-bit floating-point representations during training to reduce memory usage and increase speed.*

10.2 Gradient Accumulation

Definition 10.2 (Gradient Accumulation). *Accumulate gradients over multiple mini-batches before updating parameters, effectively increasing batch size.*

10.3 Model Parallelism

1. **Tensor Parallelism**: Split tensors across devices
2. **Pipeline Parallelism**: Split layers across devices
3. **Data Parallelism**: Replicate model across devices

11 Evaluation Metrics

11.1 Perplexity

Definition 11.1 (Perplexity). *Perplexity measures how well a language model predicts a sequence:*

$$\mathit{Perplexity} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(x_i)\right)$$

11.2 BLEU Score

Definition 11.2 (BLEU). *BLEU measures the quality of machine-translated text by comparing it to reference translations:*

$$\mathit{BLEU} = \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

where p_n is the n-gram precision.

11.3 ROUGE Score

Definition 11.3 (ROUGE). *ROUGE measures the quality of summaries by comparing them to reference summaries:*

$$\mathit{ROUGE\text{-}N} = \frac{\sum_{S\in ref}\sum_{gram_n\in S} Count_{match}(gram_n)}{\sum_{S\in ref}\sum_{gram_n\in S} Count(gram_n)}$$

12 Challenges and Limitations

12.1 Computational Challenges

1. **Quadratic Complexity**: Attention scales quadratically with sequence length
2. **Memory Requirements**: Large models require significant GPU memory
3. **Training Time**: Long training times for large models
4. **Inference Speed**: Slower inference compared to RNNs

12.2 Theoretical Limitations

1. **Positional Encoding**: Limited ability to generalize to longer sequences
2. **Inductive Bias**: Less inductive bias compared to CNNs/RNNs
3. **Interpretability**: Difficulty in understanding attention patterns

12.3 Practical Challenges

1. **Data Requirements**: Need for large datasets
2. **Hyperparameter Sensitivity**: Many hyperparameters to tune
3. **Overfitting**: Tendency to overfit on small datasets

13 Recent Advances

13.1 Large Language Models

Definition 13.1 (Large Language Model)**.** *A Transformer-based model with billions or trillions of parameters, trained on massive text corpora.*

Examples:

- GPT-3: 175 billion parameters
- GPT-4: Estimated 1+ trillion parameters
- PaLM: 540 billion parameters
- Chinchilla: 70 billion parameters

13.2 Emergent Capabilities

1. **Few-shot Learning**: Learning from few examples
2. **Chain-of-thought**: Step-by-step reasoning
3. **Code Generation**: Writing computer programs
4. **Mathematical Reasoning**: Solving mathematical problems
5. **Creative Writing**: Generating creative content

13.3 Efficient Transformers

1. **Linformer**: Linear attention mechanism
2. **Performer**: Random feature attention
3. **Reformer**: Locality-sensitive hashing attention
4. **Longformer**: Sparse attention for long sequences
5. **BigBird**: Sparse attention with global tokens

14 Future Directions

14.1 Architectural Improvements

1. **More Efficient Attention**: Reducing computational complexity
2. **Better Positional Encoding**: Handling longer sequences
3. **Modular Architectures**: Composable components
4. **Multimodal Integration**: Better cross-modal understanding

14.2 Training Improvements

1. **More Efficient Training**: Reducing training time and cost
2. **Better Optimization**: New optimization algorithms
3. **Continual Learning**: Learning new tasks without forgetting
4. **Few-shot Learning**: Learning from minimal examples

14.3 Applications

1. **Scientific Discovery**: Accelerating research
2. **Education**: Personalized learning systems
3. **Healthcare**: Medical diagnosis and treatment
4. **Creativity**: AI-assisted creative processes

15 Key Algorithms

15.1 Transformer Training

Algorithm 3 Transformer Training

```
1: Initialize model parameters $\theta$
2: Initialize optimizer (Adam with warmup)
3: for epoch = 1 to max_epochs do
4:     for batch in dataloader do
5:         Forward pass: $y = \text{Transformer}(x)$
6:         Compute loss: $\mathcal{L} = \text{Loss}(y, \text{target})$
7:         Backward pass: $\nabla_\theta \mathcal{L}$
8:         Update parameters: $\theta = \theta - \alpha \nabla_\theta \mathcal{L}$
9:     end for
10: end for
```

15.2 Attention Computation

Algorithm 4 Efficient Attention Computation

```
1: Input: $Q$, $K$, $V$ matrices
2: Compute attention scores: $S = QK^T$
3: Apply scaling: $S = S/\sqrt{d_k}$
4: Apply causal mask (for decoder): $S_{ij} = -\infty$ if $i < j$
5: Apply softmax: $A = \text{softmax}(S)$
6: Compute output: $O = AV$
7: Apply dropout: $O = \text{Dropout}(O)$
8: Output: $O$
```

16 Key Theorems

Theorem 16.1 (Attention Expressiveness)**.** *Multi-head attention can approximate any continuous function with sufficient heads and dimensions.*

Theorem 16.2 (Positional Encoding Universality)**.** *Sinusoidal positional encodings can represent any position in a sequence up to the maximum sequence length.*

Proposition 16.3 (Layer Normalization Stability)**.** *Layer normalization helps stabilize training by normalizing activations within each layer.*

Conjecture 16.4 (Attention Mechanism Necessity)**.** *Attention mechanisms are necessary for achieving state-of-the-art performance on sequence modeling tasks.*

17 Software and Tools

17.1 Popular Frameworks

1. **Hugging Face Transformers**: Pre-trained models and training scripts
2. **TensorFlow**: TensorFlow implementation
3. **PyTorch**: PyTorch implementation
4. **JAX**: JAX-based implementations
5. **Fairseq**: Facebook's sequence modeling toolkit

17.2 Training Libraries

1. **DeepSpeed**: Microsoft's deep learning optimization library
2. **FSDP**: PyTorch's Fully Sharded Data Parallel
3. **Accelerate**: Hugging Face's training acceleration library
4. **Megatron-LM**: NVIDIA's large-scale training framework

18 Conclusion

The Transformer architecture has fundamentally transformed the field of deep learning, particularly in natural language processing and beyond. Its key innovation of using attention mechanisms instead of recurrence or convolution has enabled unprecedented performance on a wide range of tasks.

Key strengths of Transformers include:

- Parallelizable training
- Long-range dependency modeling
- Strong performance on sequence tasks
- Flexibility for various applications

However, challenges remain in computational efficiency, especially for long sequences, and the need for large amounts of training data.

Future research will likely focus on:

- More efficient attention mechanisms
- Better handling of long sequences
- Reduced computational requirements
- Enhanced interpretability

The Transformer architecture continues to evolve and influence new developments in AI, making it one of the most important innovations in modern machine learning.