Week 3

Learning Resources and Tasks

Transformers & Machine Translation

Implementation from Scratch

June 22, 2025

Contents

1 Foundational Paper and Theory

1.1 Primary Research Paper

- **Attention is All You Need** – The seminal 2017 paper by Vaswani et al. that introduced the Transformer architecture
- **Direct Link:** https://arxiv.org/abs/1706.03762
- **Authors:** Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin
- **Key Innovation:** Completely eliminates recurrence and convolutions, relying entirely on attention mechanisms

1.2 Essential Conceptual Understanding

- **Self-Attention Mechanism** – Understanding how tokens attend to each other within a sequence
- **Multi-Head Attention** – Parallel attention computations across different representation subspaces
- **Positional Encoding** – Injecting sequence order information without recurrence
- **Encoder-Decoder Architecture** – Processing input sequences and generating output sequences

2 Learning Resources

2.1 Interactive Implementation Guides

- **The Annotated Transformer (Harvard NLP)** – Line-by-line PyTorch implementation with detailed explanations
- **URL:** https://nlp.seas.harvard.edu/annotated-transformer/
- **Features:** Complete working notebook, visual attention examples, training procedures

2.2 Comprehensive Video Resources

- **3Blue1Brown – Attention in transformers, visually explained** – Mathematical intuition with beautiful visualizations
- **Andrej Karpathy – Let's build GPT** – Complete transformer implementation from scratch
- **DeepLearning.AI Transformer Courses** – Structured learning path covering theory and applications

2.3 Library Documentation and Tutorials

- **Hugging Face Transformers Documentation** – Comprehensive guide to using pre-trained models
- **PyTorch Transformer Tutorial** – Official PyTorch implementation examples
- **OpenNMT-py Documentation** – Production-ready neural machine translation framework

2.4 Advanced Learning Resources

- **CS224N Stanford NLP Course** – Transformer lectures and assignments
- **Fast.ai NLP Course** – Practical deep learning for coders focusing on transformers
- **Transformer Tutorials GitHub Repository** – Collection of notebooks covering basics to advanced topics

3 Transformer Building Blocks - Implementation Tasks

3.1 Task 1: Input Embeddings and Positional Encoding

- **Objective:** Implement token embeddings and sinusoidal positional encoding from scratch
- **Key concepts:** Word embeddings, positional encoding mathematics, sequence representation
- **Deliverable:** Standalone class that converts token sequences to position-aware embeddings
- **Estimated time:** 3-4 hours

3.2 Task 2: Scaled Dot-Product Attention

- **Objective:** Build the core attention mechanism with query, key, and value matrices
- **Key concepts:** Attention scoring, softmax normalization, value weighting
- **Deliverable:** Attention function with masking support and visualization capabilities
- **Estimated time:** 4-5 hours

3.3 Task 3: Multi-Head Attention

- **Objective:** Implement parallel attention heads with learned linear projections
- **Key concepts:** Multiple representation subspaces, concatenation, output projection

- **Deliverable:** Multi-head attention module with configurable number of heads
- **Estimated time:** 4-5 hours

3.4 Task 4: Feed-Forward Networks and Layer Normalization

- **Objective:** Create position-wise feed-forward networks and layer normalization
- **Key concepts:** Point-wise transformations, residual connections, normalization
- **Deliverable:** FFN module and LayerNorm implementation with residual connections
- **Estimated time:** 3-4 hours

3.5 Task 5: Encoder Layer

- **Objective:** Combine attention and feed-forward components into complete encoder layer
- **Key concepts:** Sublayer connections, dropout, layer composition
- **Deliverable:** Single encoder layer with self-attention and feed-forward sublayers
- **Estimated time:** 3-4 hours

3.6 Task 6: Decoder Layer with Masked Attention

- **Objective:** Implement decoder layer with masked self-attention and encoder-decoder attention
- **Key concepts:** Causal masking, cross-attention, autoregressive generation
- **Deliverable:** Decoder layer supporting training and inference modes
- **Estimated time:** 5-6 hours

3.7 Task 7: Complete Transformer Model

- **Objective:** Assemble encoder and decoder stacks into full transformer architecture
- **Key concepts:** Model composition, parameter initialization, forward pass
- **Deliverable:** Complete transformer model ready for training
- **Estimated time:** 4-5 hours

4 Machine Translation Implementation Project

4.1 Project Overview

Objective: Build a complete neural machine translation system using your transformer implementation to translate between two languages.

4.2 Dataset and Preprocessing

- **Primary Dataset:** Multi30k German-English parallel corpus (manageable size for learning)
- **Alternative:** WMT 2014 English-German subset (for advanced implementation)
- **Preprocessing:** Tokenization using spaCy, vocabulary building, sequence padding
- **Tools:** Hugging Face Datasets, TorchText, spaCy tokenizers

4.3 Training Infrastructure

- **Loss Function:** Cross-entropy with label smoothing
- **Optimizer:** Adam with custom learning rate scheduling
- **Learning Rate:** Warmup followed by inverse square root decay
- **Batch Processing:** Dynamic batching by sequence length

4.4 Evaluation Metrics

- **BLEU Score:** Primary metric for translation quality
- **Perplexity:** Model confidence measurement
- **Attention Visualization:** Qualitative analysis of learned alignments
- **Translation Examples:** Human-readable output assessment

5 Implementation Framework and Deliverables

Phase 1: Foundation (Days 1-2)

- Read and thoroughly understand the "Attention is All You Need" paper
- Set up development environment with PyTorch and required libraries
- Implement basic building blocks: embeddings and positional encoding

Phase 2: Attention Mechanisms (Days 3-4)

- Implement scaled dot-product attention from scratch
- Build multi-head attention with visualization capabilities
- Test attention mechanisms with simple examples

Phase 3: Encoder Implementation (Days 5-6)

- Create feed-forward networks and layer normalization
- Assemble complete encoder layers with residual connections
- Build encoder stack and test with dummy data

Phase 4: Decoder Implementation (Day 7)

- Implement masked self-attention for autoregressive generation
- Create decoder layers with encoder-decoder attention
- Complete full transformer model architecture

Phase 5: Translation System (Days 8-9)

- Prepare translation dataset and preprocessing pipeline
- Implement training loop with proper loss computation
- Train model on machine translation task

Phase 6: Evaluation and Analysis (Day 10)

- Evaluate model performance using BLEU scores
- Create attention visualizations and analysis
- Write comprehensive learning report

6 Deliverables

6.1 Code Deliverables

- **Component Modules:** Separate implementations for each transformer building block
- **Complete Model:** Fully assembled transformer for machine translation
- **Training Pipeline:** Data loading, training loop, and evaluation scripts
- **Utilities:** Attention visualization, BLEU scoring, and inference functions

6.2 Technical Report

Objective: Document your learning journey and technical insights from implementing transformers from scratch.

6.2.1 Required Report Sections

- **Executive Summary:** High-level overview of achievements and key insights
- **Architecture Analysis:** Detailed explanation of each transformer component
- **Implementation Challenges:** Technical difficulties encountered and solutions
- **Training Results:** Performance metrics, loss curves, and training dynamics
- **Translation Quality:** BLEU scores, example translations, and error analysis
- **Attention Analysis:** Visualization and interpretation of learned attention patterns
- **Comparative Study:** Comparison with existing implementations and frameworks
- **Future Improvements:** Identified areas for enhancement and optimization

6.3 Demonstration Requirements

- **Live Translation:** Working demo of your trained model translating new sentences
- **Attention Visualization:** Interactive plots showing attention weights
- **Code Walkthrough:** Presentation of key implementation details
- **Performance Analysis:** Discussion of training results and model capabilities

7 Additional Learning Support

7.1 Debugging and Development Tips

- Start with small models and simple datasets for initial testing
- Implement thorough unit tests for each component
- Use gradient checking to verify backpropagation implementation
- Monitor attention weights during training to ensure proper learning

7.2 Performance Optimization

- Implement efficient matrix operations using PyTorch
- Use mixed precision training for faster computation
- Optimize batch processing and data loading pipelines
- Consider model parallelization for larger architectures

7.3 Weekly Schedule Recommendation

Days	**Activities**
Days 1-2	Paper study and foundation components implementation
Days 3-4	Attention mechanisms and encoder development
Days 5-6	Decoder implementation and model assembly
Days 7-8	Machine translation system development
Days 9-10	Training, evaluation, and comprehensive report writing

7.4 Success Metrics

- **Technical:** Functional transformer model achieving reasonable BLEU scores
- **Understanding:** Clear demonstration of attention mechanism comprehension
- **Implementation:** Clean, well-documented code for all components
- **Analysis:** Insightful report documenting learning process and findings