CS336 Language Modeling from Scratch

Lecture 3: Transformer Architectures and Hyperparameters

Stanford University, Spring 2025

Abstract

This lecture explores transformer architectures and hyperparameter selection for large language models, analyzing empirical evidence from 19 recent model releases. Key topics include architectural choices like layer normalization variants, activation functions, position embeddings, attention mechanisms, and the evolutionary convergence of successful design patterns. The lecture provides practical guidance for building state-of-the-art transformers based on consensus choices from leading research groups.

Enhanced Summary by:
GitHub: HtmMhmd — LinkedIn: Hatem Mohamed

Contents

1 Introduction and Course Context

1.1 Lecture Overview

This lecture explores the intricate details of transformer architectures and hyperparameter selection for large language models (LLMs), focusing on empirical evidence from 19 recent model releases. The primary goal is to understand what architectural choices and hyperparameters have proven effective through analyzing the ”evolutionary convergence” of successful models.

Key Point: Learning from Consensus

Since we cannot train all possible transformer variants ourselves, we learn from the collective experience of leading research groups who have invested billions in compute to find optimal configurations.

Learning Objectives:

1. Understand consensus architectural choices in modern transformers
2. Analyze empirical evidence for design decisions
3. Apply proven hyperparameter selection strategies
4. Recognize performance-critical vs. marginal optimizations

1.2 Lecture Methodology

Algorithm: Architectural Analysis Approach

1. Survey 19 recent high-performance language models
2. Identify convergent design patterns across research groups
3. Analyze empirical evidence for each architectural choice
4. Provide practical recommendations based on consensus

2 Architecture Variations and Consensus Choices

2.1 Layer Normalization: Pre-norm vs Post-norm

Definition: Layer Normalization Placement

Layer normalization can be applied either before (pre-norm) or after (post-norm) the main computational blocks in transformer layers, fundamentally affecting training stability and gradient flow.

$$\text{LayerNorm}(x) = \gamma \cdot \frac{x-\mu}{\sqrt{\sigma^2+\epsilon}} + \beta \tag{1}$$

where μ is the empirical mean, σ^2 is the variance, γ and β are learnable parameters.

Key Point: Pre-norm Consensus

Pre-norm has become the universal standard across all 19 surveyed models. This represents complete convergence in the field.

Performance: Pre-norm Advantages

Training Stability Benefits:

- More stable training with better gradient propagation
- Eliminates need for careful learning rate warm-up
- Preserves identity connections in residual streams
- Reduces loss spikes during training

Recent Innovation: "Double norm" - adding layer norms both before and after blocks (used in Grok and Gemma 2).

2.2 RMS Norm vs Layer Norm

Definition: RMS Normalization

RMS (Root Mean Square) Norm simplifies layer normalization by removing mean centering and bias terms:

$$\mathrm{RMSNorm}(x) = \gamma \cdot \frac{x}{\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2 + \epsilon}} \tag{2}$$

Key Point: RMS Norm Dominance

Most modern models use RMS norm due to equivalent performance with computational efficiency benefits.

Performance: RMS Norm Performance Impact

Despite being only 0.17% of FLOPs, normalization operations account for 25% of runtime due to memory movement costs. RMS norm provides:

- Equivalent performance to layer norm
- Faster computation (fewer operations)
- Reduced memory movement overhead
- Fewer parameters to load from memory

2.3 Bias Terms

Consensus Choice: Most modern transformers omit bias terms entirely.

Reasons for dropping bias terms:

- Equivalent performance without bias terms

- Improved optimization stability
- Cleaner matrix multiply operations
- Reduced parameter count

2.4 Activation Functions and Gated Linear Units

2.4.1 Standard Activations

- **ReLU:** $\text{ReLU}(x) = \max(0, x)$
- **GELU:** Gaussian Error Linear Unit - multiplies input by CDF of Gaussian
- **Swish:** $\text{Swish}(x) = x \cdot \sigma(x)$ where σ is sigmoid

2.4.2 Gated Linear Units (GLUs)

Definition: GLUs introduce gating mechanisms to MLPs:

$$\text{GLU}(x) = (xW_1 + b_1) \odot \sigma(xV + c) \text{ then } W_2 \tag{3}$$

where $\odot$ denotes element-wise multiplication.
Popular Variants:

- **GeGLU:** Uses GELU activation with gating
- **SwiGLU:** Uses Swish activation with gating (most popular)

Consensus Choice: SwiGLU has become dominant in modern models (LLaMA, PaLM, etc.).

Parameter Scaling: GLU variants typically scale hidden dimensions by 2/3 to maintain parameter count parity with non-gated counterparts.

2.5 Serial vs Parallel Layers

Serial (Standard): Attention → MLP sequentially **Parallel:** Attention and MLP computed simultaneously, then summed

$$\text{Parallel: } y = x + \text{Attention}(x) + \text{MLP}(x) \tag{4}$$

Trade-offs:

- Parallel enables better GPU utilization and systems efficiency
- Serial may be more expressive due to computation composition
- Most recent models prefer serial layers

2.6 Position Embeddings

2.6.1 RoPE (Rotary Position Embedding)

Definition: RoPE enforces that attention weights depend only on relative positions:

$$f(x_i, i) \cdot f(x_j, j) = g(x_i, x_j, i - j) \tag{5}$$

Implementation: Rotate embedding vectors by position-dependent angles:

$$\begin{pmatrix} q_{2k} \\ q_{2k+1} \end{pmatrix} = \begin{pmatrix} \cos(m\theta_k) & -\sin(m\theta_k) \\ \sin(m\theta_k) & \cos(m\theta_k) \end{pmatrix} \begin{pmatrix} x_{2k} \\ x_{2k+1} \end{pmatrix} \tag{6}$$

where m is the position and θ_k varies across dimension pairs.
Consensus Choice: RoPE has achieved universal adoption since 2023.
Key Properties:

- Operates at attention layer, not input embeddings
- Preserves relative position information
- Enables context length extrapolation
- No learned parameters required

3 Hyperparameter Guidelines

3.1 Feed-Forward Network Sizing

Standard Rule: $d_{ff} = 4 \times d_{model}$ for non-GLU models

GLU Adjustment: $d_{ff} = \frac{8}{3} \times d_{model} \approx 2.67 \times d_{model}$

Empirical Evidence: Kaplan et al. scaling laws show optimal ratios between 1-10, with 4 being well within the optimal basin.

Notable Exception: T5-11B used an extreme 64× ratio ($d_{model} = 1024$, $d_{ff} = 65536$), though T5v1.1 reverted to standard ratios.

3.2 Multi-Head Attention Configuration

Standard Rule: $d_{model} = n_{heads} \times d_{head}$ (ratio = 1)

Most successful models maintain this 1:1 ratio, keeping total attention parameters constant as head count increases.

Consensus Models: GPT-3, T5, LLaMA-2, PaLM all follow this guideline.

3.3 Model Aspect Ratio

Definition: Aspect ratio = $\frac{d_{model}}{n_{layers}}$

Optimal Range: Approximately 128 hidden dimensions per layer

Empirical Support: Kaplan et al. studies across multiple scales (50M to 1.5B parameters) consistently show optima around 100-128.

Systems Considerations:

- Depth affects pipeline parallelism efficiency
- Width affects tensor parallelism requirements
- Network bandwidth constraints influence optimal ratios

3.4 Vocabulary Size

Historical Trend: 30K-50K tokens → 100K-250K tokens

Driving Factors:

- Multilingual deployment requirements
- Emoji and special character support
- Production system needs
- Inference cost optimization for low-resource languages

Modern Standards: GPT-4 (100K), Command-R (256K)

3.5 Regularization: Weight Decay

Surprising Finding: Weight decay improves training loss, not just validation loss.

Mechanism: Complex interaction with learning rate schedules creates implicit acceleration during the "cooling" phase of cosine decay.

$$\mathcal{L}_{total} = \mathcal{L}_{CE} + \lambda \sum_i w_i^2 \tag{7}$$

Key Insight: In single-epoch pre-training, weight decay serves optimization purposes rather than traditional regularization.

4 Training Stability Innovations

4.1 Z-Loss for Output Softmax

Definition: Auxiliary loss to stabilize softmax normalizer:

$$\mathcal{L}_z = \alpha \log^2 \left(\sum_j e^{z_j} \right) \tag{8}$$

Goal: Force normalizer $Z(x) \approx 1$, making softmax numerically stable.

Adoption: PaLM (pioneer), Chinchilla, DCLM, OLMo-2

4.2 QK-Norm for Attention Softmax

Definition: Apply layer normalization to queries and keys before attention:

$$\text{Attention} = \text{softmax}\left(\frac{\text{LayerNorm}(Q)\text{LayerNorm}(K)^T}{\sqrt{d_k}} \right) V \tag{9}$$

Origin: Vision transformer stability research (Dehghani et al., 2023)

Benefits:

- Bounds softmax inputs
- Enables more aggressive learning rates
- Improves gradient norm stability

Adoption: Gemma-2, DCLM, OLMo-2

4.3 Soft Capping

Definition: Soft clipping of attention logits:

$$\text{soft\_cap}(x) = \text{soft\_cap} \times \tanh\left(\frac{x}{\text{soft\_cap}}\right) \tag{10}$$

Mixed Results: Some evidence of degraded performance compared to QK-norm.

5 Advanced Attention Mechanisms

5.1 Multi-Query Attention (MQA) and Group Query Attention (GQA)

Motivation: Inference efficiency during autoregressive generation.

Problem: KV-cache memory access patterns create poor arithmetic intensity:

$$\text{Arithmetic Intensity} = \frac{\text{FLOPs}}{\text{Memory Access}} = \frac{1}{\frac{n}{d} + \frac{1}{b}} \tag{11}$$

MQA Solution: Share key and value heads across multiple query heads.

GQA Solution: Intermediate approach - group queries share fewer KV heads.

Benefits:

- Reduces memory bandwidth requirements
- Enables longer sequence lengths at inference
- Improves throughput for autoregressive generation

5.2 Sparse Attention Patterns

Recent Innovation: Alternating attention patterns (LLaMA-4, Gemma, Command-A):

- Every 4th layer: Full self-attention without position embeddings
- Other layers: Sliding window attention with RoPE

Advantages:

- Controls computational costs for long contexts
- Enables aggressive length extrapolation
- Maintains global information flow

6 Summary of Consensus Choices

7 Key Takeaways

1. **Empirical Convergence:** Despite many possible choices, successful models have converged on similar architectures
2. **Memory vs Compute:** Modern architecture decisions increasingly consider memory movement, not just FLOPs
3. **Stability Focus:** Recent innovations emphasize training stability through normalization and regularization

Component	Consensus Choice
Layer Placement	Pre-norm (universal)
Normalization	RMS norm (dominant)
Bias Terms	Omitted (widespread)
Activation	SwiGLU (dominant)
Position Embedding	RoPE (universal since 2023)
FF Ratio	4× (non-GLU), 2.67× (GLU)
Head Configuration	$d_{model} = n_{heads} \times d_{head}$
Aspect Ratio	∼128 dims/layer
Vocabulary	100K-250K tokens
Regularization	Weight decay (optimization tool)

Table 1: Consensus architectural choices across modern LLMs

4. **Systems Integration:** Architecture choices must consider parallelization strategies and hardware constraints

5. **Conservative Innovation:** Most successful models follow established patterns with incremental improvements

8 Transformer Architecture Mind Map

8.1 Mind Map Legend

- Green branches: Consensus choices adopted universally or by most models
- Yellow branches: Emerging trends gaining adoption
- Red center: Core architectural foundation

8.2 Key Interconnections

The mind map emphasizes several critical relationships:

1. **Stability ↔ Normalization**: Pre-norm and RMS norm choices directly impact training stability

2. **Efficiency ↔ Hyperparameters**: Memory vs compute trade-offs influence optimal ratios and configurations

3. **Position Encoding ↔ Efficiency**: RoPE enables both stable training and efficient context extrapolation

4. **Activations ↔ Hyperparameters**: GLU variants require adjusted feed-forward ratios

5. **Stability ↔ Efficiency**: Modern stability techniques (QK-norm, Z-loss) enable more aggressive optimization

8.3 Architectural Decision Flow

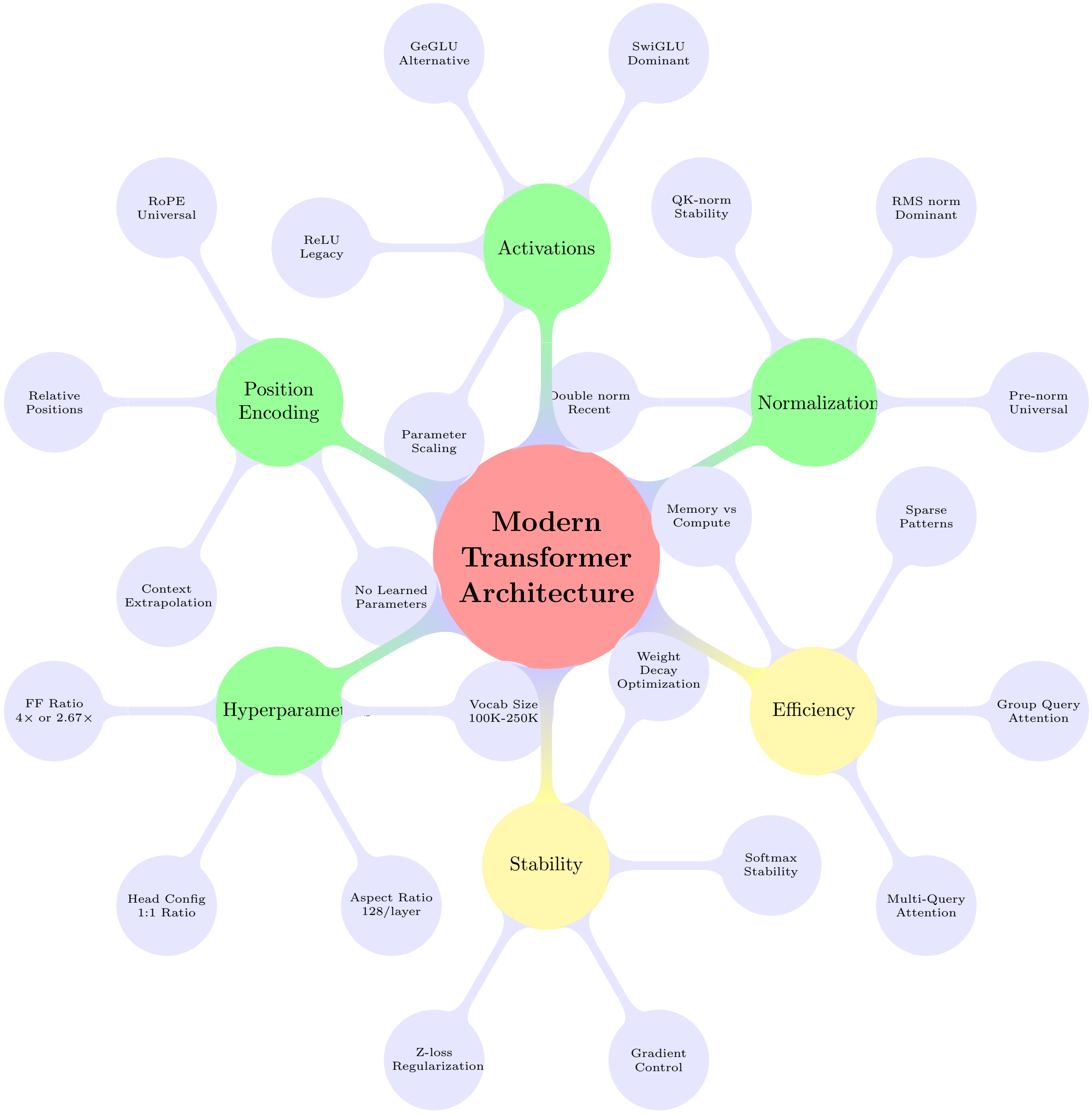

Figure 1: Mind Map of Modern Transformer Architecture Components

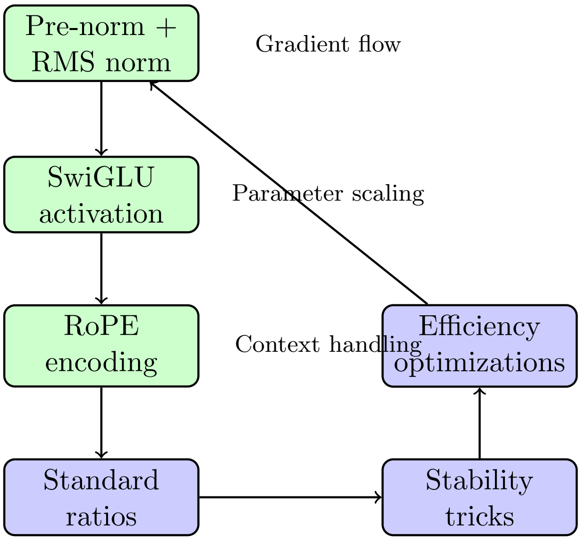

Figure 2: Architectural Decision Dependencies