CS336: Language Modeling from Scratch

Lecture 9: Scaling Laws - Part 1

Stanford University - Spring 2025

Abstract

Abstract: This lecture introduces the fundamental concepts of scaling laws in language modeling, providing both theoretical foundations and practical engineering guidance. We explore the historical development of scaling laws from statistical machine learning theory to modern deep learning applications. The lecture covers data scaling laws, model scaling laws, joint optimization strategies, and the critical engineering decisions that emerge from scaling law analysis. Key topics include the Chinchilla scaling ratios, critical batch size analysis, and the practical methodology for using small-scale experiments to predict large-scale model behavior.

Enhanced Summary by:

GitHub: HtmMhmd | LinkedIn: Hatem Mohamed

Contents

1 Introduction and Motivation

Key Point

Core Scenario: You have 100,000 H100s for a month and must build the best open-source language model possible. How do you make optimal engineering decisions without wasting massive compute resources?

1.1 The Engineering Challenge

The fundamental challenge in modern language model development is optimization under compute constraints. Traditional approaches involve:

- Training many large models with different hyperparameters (extremely expensive)
- Copying existing architectures without innovation (limits frontier advancement)
- Making ad-hoc engineering decisions (unpredictable outcomes)

Definition

Scaling Laws: Simple, predictive mathematical relationships that describe how language model performance changes as a function of key variables like model size, dataset size, and compute budget.

1.2 The Scaling Law Paradigm

Algorithm/Method

Modern Scaling Approach:

1. Train many small models across different configurations
2. Establish predictive relationships from small-scale experiments
3. Extrapolate to large-scale optimal configurations
4. Execute single large-scale training run with high confidence

This approach transforms the economics of large model development from "trial and error at scale" to "predict and execute."

2 Historical Context and Theoretical Foundations

2.1 Statistical Learning Theory Origins

Definition

Classical Generalization Bounds: Traditional machine learning theory provides upper bounds on how error should decay with sample size:

- Finite hypothesis class: Error $\propto \frac{1}{\sqrt{m}}$
- Non-parametric density estimation: Error $\propto n^{-\frac{\beta}{2\beta+1}}$

Key Point

The key insight is that scaling laws represent the empirical realization of theoretical predictions that were previously only upper bounds.

2.2 Early Scaling Law Research

Performance Insight

Historical Milestone - Bell Labs 1993: The first scaling law paper by Vapnik, Cortes, and others established:

- Predictive methods for classifier performance without full training
- Functional form: Test Error = Irreducible Error + Polynomial Decay Term
- Validation that small model training can predict large model behavior

2.3 Modern Development Timeline

- **2001 - Banko & Brill:** NLP system performance scaling with data
- **2012:** Power law functional form validation
- **2017 - Hestness et al. (Baidu):** First large-scale neural scaling laws

Algorithm/Method

Hestness Three-Region Model:

1. **Random Performance Region:** Models perform at chance level
2. **Power Law Region:** Predictable scaling behavior
3. **Asymptotic Region:** Approaching irreducible error

3 Data Scaling Laws

3.1 Theoretical Foundation

Definition

Data Scaling Law: A mathematical relationship mapping dataset size n to excess error (error beyond irreducible minimum).

3.2 Statistical Motivation

Algorithm/Method

Simple Example - Mean Estimation:

- Input: $X \sim \mathcal{N}(\mu, \sigma^2)$
- Task: Estimate μ
- Error: $\text{Var}[\hat{\mu}] = \frac{\sigma^2}{n}$
- Log-log relationship: $\log(\text{error}) = -\log(n) + 2\log(\sigma)$
- Expected slope: -1

3.3 Non-Parametric Function Estimation

Performance Insight

Multi-dimensional Scaling: For estimating smooth functions in D dimensions:

- Partition space into $\sqrt{n}$ boxes per dimension
- Each box receives $\sqrt{n}$ samples
- Error scales as $n^{-\frac{1}{D}}$
- Scaling law slope: $-\frac{1}{D}$

3.4 Empirical Observations

Key Point

Observed vs. Expected Slopes:

- Machine Translation (Hestness): -0.13 (much slower than -0.5 or -1.0)
- Speech Recognition: -0.3
- Language Modeling: -0.095

The slower-than-expected rates suggest high intrinsic dimensionality in language tasks.

3.5 Practical Applications

Algorithm/Method

Data Composition Analysis:

1. Test different data mixtures at small scale
2. Observe that composition affects offset, not slope
3. Scale up optimal mixture with confidence
4. Achieve better performance without large-scale data experiments

Warning

Multi-Epoch Training: After approximately 4 epochs, diminishing returns set in rapidly. Effective sample size decreases significantly with repeated data.

4 Model Scaling Laws

4.1 Architecture Comparison

Performance Insight

Transformer vs. LSTM Scaling: Analysis shows consistent compute efficiency gaps:

- Transformers maintain 15x compute efficiency advantage
- Gap persists across all model scales (constant factor difference)
- Log-scale analysis reveals this as parallel scaling curves

4.2 Alternative Architecture Analysis

Algorithm/Method

Architecture Evaluation Methodology (Tay et al.):

1. Train multiple architectures across compute scales
2. Compare against Transformer baseline
3. Identify consistently superior architectures
4. Results: Only GLU and Mixture of Experts consistently outperform Transformers

4.3 Hyperparameter Scaling

Key Point

Aspect Ratio Analysis: Depth vs. width trade-offs show:

- Single layer performs poorly
- Wide basin of optimal ratios (4:1 to 16:1 width:depth)
- Scaling law analysis confirms architectural intuitions

Warning

Parameter Counting Subtlety: Embedding parameters don't follow the same scaling laws as non-embedding parameters. Always exclude embeddings from parameter scaling analysis.

4.4 Optimizer Scaling

Performance Insight

Adam vs. SGD: Consistent constant-factor efficiency gap favoring Adam across all scales, similar to architecture comparisons.

5 Batch Size and Learning Rate Scaling

5.1 Critical Batch Size Theory

Definition

Critical Batch Size: The threshold beyond which increasing batch size provides diminishing returns. Below this threshold, doubling batch size is equivalent to taking two gradient steps.

Algorithm/Method

Batch Size Scaling Regions:

1. **Perfect Scaling:** Batch size $<$ noise scale
2. **Diminishing Returns:** Batch size $\geq$ noise scale
3. **Critical Point:** Transition between regimes

5.2 Loss-Dependent Scaling

Key Point

Counter-intuitive Result: As target loss decreases (better models), critical batch size increases. This enables larger parallelism for better models.

5.3 Learning Rate Parameterization

Algorithm/Method

Standard Practice vs. μP (Mu Parameterization):
Standard:

- Optimal learning rate $\propto \frac{1}{\text{width}}$
- Requires re-tuning at each scale
- Predictive scaling law fitting needed

μP Approach:

- Reparameterize initialization and layer outputs
- Learning rate becomes scale-invariant
- Tune once at small scale, transfer directly

6 Loss vs. Downstream Performance

Warning

Critical Limitation: Scaling laws work excellently for training loss (log perplexity) but are much less predictable for downstream task performance.

Performance Insight

Empirical Evidence:

- Training loss: Clean log-linear relationship with compute
- Downstream tasks: Scattered, architecture-dependent performance
- State-space models: Good perplexity scaling, poor downstream scaling

7 Joint Data-Model Scaling Laws

7.1 Mathematical Framework

Definition

Joint Scaling Functional Forms:
Rosenfeld Form:

$$\text{Error} = A \cdot D^{-\alpha} + B \cdot N^{-\beta} + C$$

Kaplan Form:

$$\text{Error} = \left(\frac{A}{D^\alpha} + \frac{B}{N^\beta} \right)$$

Where D = data size, N = model parameters, α, β = scaling exponents.

7.2 Empirical Validation

Performance Insight

Rosenfeld Validation:

- Train on small models and data only
- Extrapolate to large models and datasets
- Prediction accuracy: Near-perfect correlation between predicted and actual performance
- Success across ImageNet and WikiText domains

8 The Chinchilla Analysis

8.1 The Compute Allocation Problem

Key Point

Central Question: For a fixed compute budget, what is the optimal trade-off between model size and training data size?

8.2 Historical Context

Warning

Kaplan vs. Chinchilla: Original Kaplan estimates were significantly off due to:

- Learning rate schedule effects (cosine schedules cannot be truncated)
- Incomplete exploration of the parameter space
- Methodological limitations in curve fitting

8.3 Chinchilla Methodology

Algorithm/Method

Three Independent Methods:
Method 1 - Minimum Envelope:

1. Train models of various sizes to completion
2. Identify lower envelope of optimal performance
3. Extract scaling relationship from envelope

Method 2 - IsoFLOP Analysis:

1. Fix total compute budget
2. Sweep model sizes (data size varies inversely)
3. Find minimum loss for each compute level
4. Extract optimal parameter/token ratio

Method 3 - Parametric Fitting:

1. Fit joint scaling law directly
2. Differentiate to find optimal allocation
3. Validate against empirical minima

8.4 Chinchilla Results

Performance Insight

Optimal Scaling Coefficients:

- Methods 1 & 2: $\alpha = \beta = 0.5$ (equal scaling in data and parameters)
- Method 3: Slightly different estimates (~ 0.03 difference)
- **Chinchilla Ratio:** ~ 20 tokens per parameter
- Kaplan estimate: Significantly sub-optimal

8.5 Learning Rate Schedule Impact

Warning

Cosine Schedule Constraint: Cosine learning rate schedules must complete their full cycle. Early truncation invalidates the training, contributing to Kaplan's estimation errors.

9 Practical Implementation Methodology

9.1 Scaling Law Design Process

Algorithm/Method

Engineering Workflow:

1. **Small-Scale Exploration:** Train models across 2-3 orders of magnitude in compute
2. **Relationship Establishment:** Verify log-linear relationships in target metrics
3. **Hyperparameter Optimization:** Use small-scale results to set large-scale parameters
4. **Validation:** Ensure non-crossing curves and consistent slopes
5. **Execution:** Scale up with high confidence

9.2 Key Validation Criteria

Key Point

Scaling Law Quality Indicators:

- **Non-crossing curves:** Different configurations maintain relative order
- **Parallel slopes:** Similar scaling exponents across configurations
- **Log-linear fit quality:** High R^2 values in log-log plots
- **Extrapolation stability:** Consistent predictions across multiple methods

9.3 Common Pitfalls

Warning

Methodological Cautions:

- Don't assume downstream task scaling matches perplexity scaling
- Account for learning rate schedule constraints
- Distinguish between different parameter types (embedding vs. non-embedding)
- Validate scaling laws across multiple architectures before generalizing
- Consider diminishing returns in multi-epoch training

10 Advanced Topics and Extensions

10.1 Multi-Epoch Data Scaling

Definition

Effective Sample Size: When training on repeated data, the effective amount of unique information decreases. After ~ 4 epochs, diminishing returns become severe.

10.2 Data Composition Optimization

Algorithm/Method

Mixture Optimization Strategy:

1. Establish that composition affects offset, not slope
2. Optimize data mixture at small scale
3. Scale up optimal mixture with confidence
4. Consider trade-offs between data quality and repetition

10.3 Compute-Optimal Training

Performance Insight

Resource Allocation Framework: Given a fixed compute budget, optimize across:

- Model size vs. training time
- Data collection vs. GPU procurement
- Data quality vs. data quantity
- Single large model vs. multiple smaller models

11 Future Directions and Open Questions

11.1 Scaling Law Limitations

Warning

Known Limitations:

- Breakdown for out-of-distribution tasks
- Inverse scaling behaviors in specific contexts
- Difficulty predicting emergent capabilities
- Architecture-specific downstream performance variations

11.2 Research Frontiers

Key Point

Active Research Areas:

- Multimodal scaling laws
- Post-training scaling (RLHF, fine-tuning)
- Mixture of Experts parameter counting
- Scale-invariant parameterizations (μP extensions)
- Data efficiency beyond Chinchilla ratios

12 Summary and Key Takeaways

Performance Insight

Core Insights:

1. **Predictive Power:** Scaling laws enable confident large-scale decisions from small-scale experiments
2. **Resource Optimization:** Chinchilla ratios (20 tokens/parameter) provide compute-optimal allocation
3. **Architecture Selection:** Scaling analysis reveals which innovations are worth pursuing
4. **Engineering Efficiency:** Small-scale hyperparameter tuning transfers to large scale
5. **Limitation Awareness:** Training loss scaling doesn't guarantee downstream task scaling

Algorithm/Method

Practical Checklist for Scaling Law Application:

1. Establish clean log-linear relationships at small scale
2. Verify non-crossing, parallel curves across configurations
3. Account for learning rate schedule constraints
4. Distinguish between different parameter types
5. Validate on training loss before extrapolating to capabilities
6. Plan for architectural innovations that show consistent scaling advantages

Key Point

The Scaling Law Paradigm: Transform language model development from expensive trial-and-error to predictive engineering through systematic small-scale experimentation and mathematical extrapolation.

13 Scaling Laws Mindmap

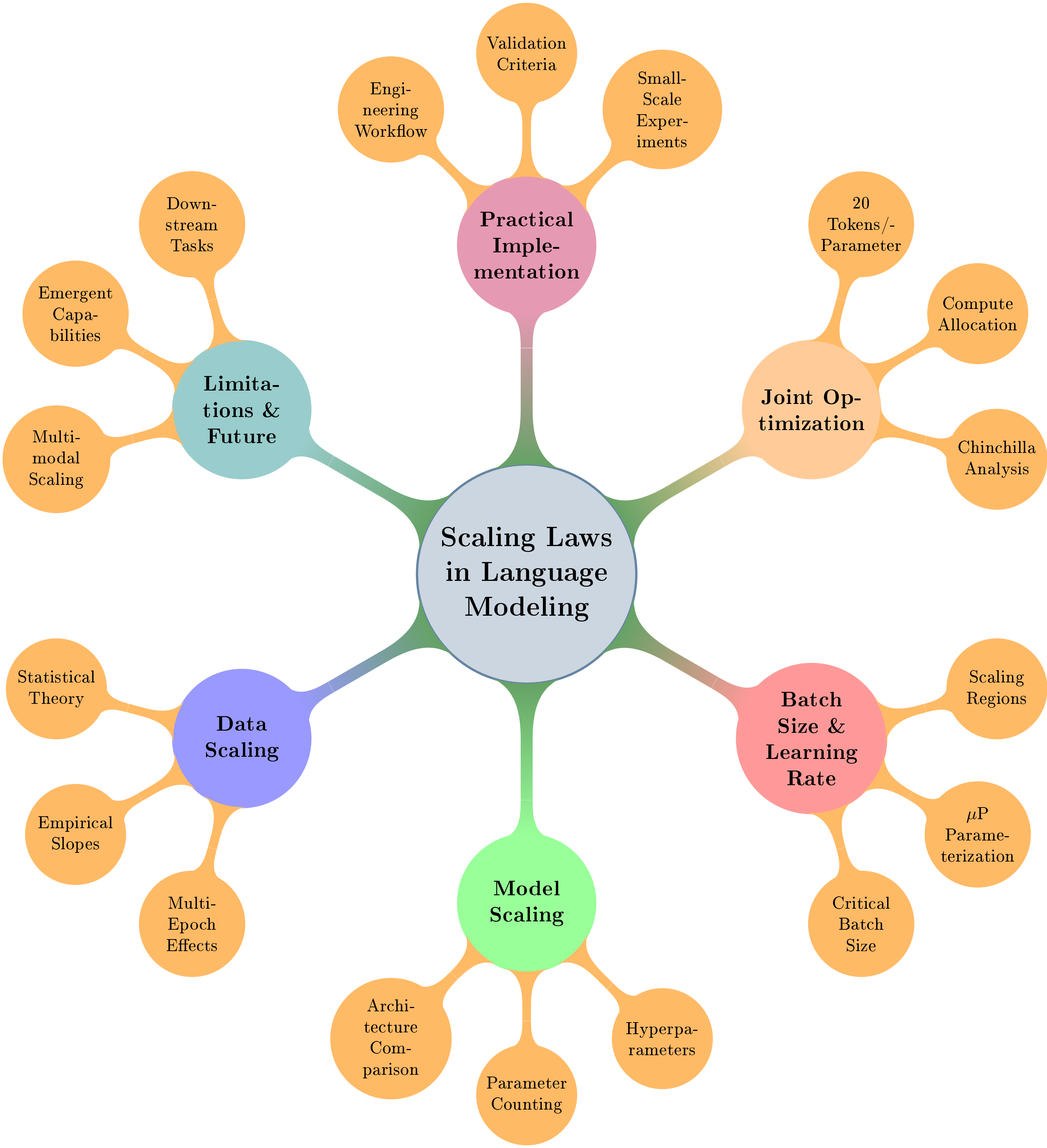

Figure 1: Comprehensive mindmap of scaling laws in language modeling, showing the six main areas covered in this lecture and their key components.

14 Detailed Mindmap Description

14.1 Scaling Laws Mindmap Overview

The mindmap in Figure 1 provides a comprehensive visual representation of the six major conceptual areas covered in this lecture on scaling laws for language modeling. Each branch represents a fundamental aspect of scaling law theory and practice, with the central concept being "Scaling Laws in Language Modeling" from which all specialized topics radiate.

14.2 Central Concept and Structure

The mindmap employs a radial structure with the core concept at the center, surrounded by six primary branches representing the main topic areas. Each branch uses distinct color coding to facilitate visual navigation and conceptual grouping. The hierarchical structure moves from general concepts at the first level to specific implementation details at deeper levels.

14.3 Branch-by-Branch Analysis

- **Data Scaling (Blue):** Encompasses the theoretical foundations from statistical learning theory, empirical observations of scaling slopes in real datasets, and practical considerations like multi-epoch training effects.
- **Model Scaling (Green):** Covers architectural comparisons between different model types, proper parameter counting methodologies, and hyperparameter optimization strategies that scale with model size.
- **Batch Size & Learning Rate (Red):** Addresses the critical batch size phenomenon, μP parameterization techniques for scale-invariant learning rates, and the identification of different scaling regions in the optimization landscape.
- **Joint Optimization (Orange):** Focuses on the landmark Chinchilla analysis, compute allocation strategies, and the practical rule of thumb for optimal training (20 tokens per parameter).
- **Practical Implementation (Purple):** Provides the engineering methodology including small-scale experimental design, validation criteria for scaling law quality, and the systematic workflow for applying scaling laws in practice.
- **Limitations & Future (Teal):** Acknowledges current limitations including downstream task prediction challenges, emergent capability scaling, and emerging research directions like multimodal scaling laws.

14.4 Visual Design Philosophy

The mindmap uses a color-coded approach where each major branch has its own color family, progressing from darker saturated colors at higher levels to lighter shades at more detailed levels. This creates visual hierarchy while maintaining conceptual cohesion within each topic area. The circular layout emphasizes that scaling laws represent an integrated approach to language model development rather than isolated techniques.