ACADEMIC RESEARCH PAPER

Scaling Laws for Neural Language Models

An Analysis of Performance Scaling with Model Size, Dataset Size, and Compute

Jared Kaplan[1,2], Sam McCandlish[2], et al.

[1]Johns Hopkins University [2]OpenAI

Overview

01 Introduction & Motivation

Research goals and key findings overview

02 Background & Methods

Experimental setup and notation

03 Core Scaling Laws

Power laws for N, D, and C

04 Overfitting & Data Limits

Universality of overfitting and data requirements

05 Training Time & Compute

Critical batch size and learning curves

06 Optimal Resource Allocation

Compute-efficient training strategies

07 Key Findings & Implications

Practical takeaways and future directions

INTRODUCTION

The Study of Scaling

Research Motivation

Language modeling provides a natural domain for studying artificial intelligence:

- Most reasoning tasks can be efficiently expressed and evaluated in language
- The world's text provides abundant data for unsupervised learning via generative modeling
- Deep learning has seen rapid progress, approaching human-level performance on many tasks

Key Research Question

How does language model performance (cross-entropy loss) depend on:

Model Architecture	Model Size (N)
Compute (C)	Dataset Size (D)

Focus: Transformer Architecture

This work focuses on the Transformer architecture due to its:

- **High ceiling:** State-of-the-art performance on many tasks
- **Low floor:** Easy to train small models
- **Wide range:** Allows study over 7+ orders of magnitude in scale

Empirical Approach

Throughout the paper, the authors observe **precise power-law scalings** for performance as a function of:

Training Time | Context Length | Dataset Size | Model Size | Compute Budget

These scaling relationships allow prediction of model performance before training.

KEY FINDINGS

Eight Major Results

1 Performance Depends on Scale, Not Shape

Performance depends most strongly on three scale factors: **N** (parameters), **D** (dataset), and **C** (compute). Within reasonable limits, performance depends very weakly on ar chitectural hyperparameters such as depth vs. width.

2 Smooth Power Laws

Performance has a power-law relationship with each scale factor when not bottleneck ed by others, spanning **6+ orders of magnitude**. No signs of deviation on the upper end , though performance must flatten eventually.

3 Universality of Overfitting

Performance improves predictably when N and D scale together, but enters diminishin g returns if one increases while the other is fixed.

Penalty depends on ratio $N^{0.74}/D$

4 Universality of Training

Training curves follow predictable power-laws with parameters roughly independent of model size. Early training can predict final loss if trained much longer.

5 Transfer Improves with Performance

When evaluating on different text distributions, results correlate strongly with training validation performance. Transfer incurs a **constant penalty** but improves in line with tr aining set performance.

6 Sample Efficiency

Larger models are significantly more sample-efficient, reaching the same performance with **fewer optimization steps** and **fewer data points**.

7 Convergence is Inefficient

With fixed compute budget C, optimal performance comes from training **very large m odels** and stopping **significantly before convergence**. Data requirements grow very slo wly as $D \propto C^{0.27}$.

8 Optimal Batch Size

Ideal batch size is roughly a power of the loss only, determinable by gradient noise sca le. For largest models, $B_{crit} \approx$ **1-2 million tokens** at convergence.

SCALING LAWS

The Three Core Power Laws

1 $L(N)$

Parameters Only

$$L(N) = (N_c/N)^{\alpha_N}$$

$\alpha_N \approx 0.076$

$N_c \approx 8.8\times10^{13}$

For models trained to convergence on sufficiently large datasets.

2 $L(D)$

Dataset Only

$$L(D) = (D_c/D)^{\alpha_D}$$

$\alpha_D \approx 0.095$

$D_c \approx 5.4\times10^{13}$

For large models trained with limited data and early stopping.

3 $L(C_{min})$

Compute Only

$$L(C_{min}) = (C_{minc}/C_{min})^{\alpha_{minC}}$$

$\alpha_{minC} \approx 0.050$

$C_{minc} \approx 3.1\times10^{8}$ PF-days

With optimal model size, sufficiently large dataset, and small batch siz
e.

Power-Law Exponents

The exponents specify the degree of performance improvement when scaling:

Doubling parameters: Loss decreases by factor $2^{-\alpha_N} \approx 0.95$

Doubling data: Loss decreases by factor $2^{-\alpha_D} \approx 0.93$

Doubling compute: Loss decreases by factor $2^{-\alpha_{minC}} \approx 0.97$

Important Notes

- Relationships hold across 8 orders of magnitude in C, 6 in N, and 2 in D.
- Depend very weakly on model shape and other hyperparameters.
- Numerical values are specific to WebText2 tokenization; N_c, D_c have no fundam ental meaning.

METHODOLOGY

Experimental Setup

Dataset: WebText2

Extended version of WebText dataset:

Documents 20.3M	Text Size 96 GB
Words 16.2B	Tokens 22.9B

Test set: 660M tokens. Also evaluated on Books, Common Crawl, Wikipedia.

Tokenization

- Byte-pair encoding (BPE)
- Vocabulary size: 50,257
- Reversible tokenizer from Radford et al. (2019)

Architecture

Primarily decoder-only Transformer models. Also trained LSTMs and Universal Tr ansformers for comparison.

Training Procedures

Optimizer

Adam (Adafactor for >1B params)

Steps

Fixed 2.5×10^5 steps

Batch Size

512 sequences (524,288 tokens)

Learning Rate

3000-step warmup, cosine decay to 0

Evaluation Metric

Autoregressive log-likelihood (cross-entropy loss) averaged over 1024-token con text.

Tested on WebText2 and other distributions.

METHODOLOGY

Architecture & Compute Estimation

Model Size Definition (N)

Defined as **non-embedding parameters**:

$$N \approx 2 d_{model}\, n_{layer}\, (2 d_{attn} + d_{ff})$$

With standard ratios, this simplifies to:

$$N \approx 12\, n_{layer}\, d^2_{model}$$

Key: Excluding embeddings produces cleaner scaling laws.

Compute Estimation (C)

Forward pass compute:

$$C_{forward} \approx 2N + 2 n_{layer}\, n_{ctx}\, d_{model}$$

Total training compute (forward + backward):

$$C \approx 6NBS$$

Where one PF-day = 8.64×10^{19} FLOPs.

Parameter & Compute Breakdown

Operation	Parameters	FLOPs/Token
Embed	$(n_{vocab} + n_{ctx}) d_{model}$	$2 d_{model}$
Attention: QKV	$4 d^2_{model}$	$2 d^2_{model}$
Attention: Mask	—	$2 n_{ctx} d_{model}$
Attention: Project	$4 d^2_{model}$	$2 d^2_{model}$
Feedforward	$8 d^2_{model}$	$8 d^2_{model}$
De-embed	—	$2 d_{model} n_{vocab}$
Total (Non-Emb)	**$N = 12 d^2_{model} n_{layer}$**	**$C_{forward} \approx 2N$**

Assumption: For $d_{model} >> n_{ctx}/12$, context-dependent compute is small fraction of total. This h olds for most models studied.

EMPIRICAL RESULTS

Model Shape Independence

Key Finding

When total non-embedding parameter count **N** is held fixed, Transformer perfor mance depends very weakly on shape parameters:

- Number of layers (n_{layer})
- Number of attention heads (n_{heads})
- Feed-forward dimension (d_{ff})

Aspect Ratio Flexibility

Aspect ratio (width vs. depth) can vary by a factor of **40×** while only slightly impac ting performance.

Example: (n_{layer}, d_{model}) = (6, 4288) reaches loss within **3%** of (48, 1600)

Why Exclude Embeddings?

When including embedding parameters, performance appears to depend strongl y on layers. When excluding them, different depths converge to a single trend.

Implication: Embedding matrix can be made smaller without impacting performance.

Possible Explanation

Deeper Transformers may effectively behave as ensembles of shallower models, similar to findings in ResNets. This would explain why depth has minimal effect wi thin wide ranges.

Practical Implication: Model architecture hyperparameters are less important than the overall model scale (total parameter count N).

EMPIRICAL RESULTS

Scaling with Model Size N

Power-Law Relationship

When trained to near convergence on full WebText2 dataset (no overfitting), test loss follows:

$$L(N) \approx (N_c/N)^{\alpha_N}$$

Exponent	Constant
$\alpha_N \approx 0.076$	$N_c \approx 8.8\times10^{13}$

Range of Validity

- Models from **768** to **1.5 billion** non-embedding parameters
- Shape variations: (2, 128) to (207, 768)
- Spanning **8 orders of magnitude** in compute
- No signs of deviation from power law on upper end

Cross-Dataset Validation

Models trained on WebText2 show the same power-law scaling when evaluated on other datasets:

- Books Corpus ✓
- Common Crawl ✓
- English Wikipedia ✓
- Internet Books ✓

All show nearly identical power-law exponents.

Comparison: Transformers vs. LSTMs

LSTMs perform as well as Transformers for **early tokens** in context, but cannot match Transformer performance for **later tokens**.

Interpretation: Transformers asymptotically outperform LSTMs due to improved use of long contexts. Larger Transformers show increasingly steep power-law improvements per token.

EMPIRICAL RESULTS

Scaling with Dataset Size D

Power-Law Relationship

When training a fixed model on subsets of WebText2 with early stopping, test loss follows:

$$L(D) \approx (D_c/D)^{\alpha_D}$$

Exponent	Constant
$\alpha_D \approx 0.095$	$D_c \approx 5.4\times10^{13}$

Experimental Method

1. Fixed model: $(n_{layer}, d_{model}) = (36, 1280)$
2. Train on subsets: 22M to 22B tokens
3. Stop when test loss ceases to decrease
4. Fit final losses to power law

What This Represents

This scaling law represents the **data-limited regime** where:

- Models have sufficient capacity to fit the data
- Performance is bottlenecked by dataset size
- No overfitting occurs (early stopping prevents it)
- Additional data would improve performance

Key Implications

Predictable Improvement

Larger datasets continue to improve performance predictably across 2+ orders of magnitude.

No Plateau

No evidence of performance plateau, though loss must eventually reach entropy lower bound.

EMPIRICAL RESULTS

Scaling with Compute C

Empirical Trend (Fixed Batch Size)

By scanning over models of various N and finding optimal performance at each co mpute budget (fixed batch size $B = 2^{19}$):

$$L(C) \approx (C_c/C)^{\alpha_C}$$

Exponent	Constant
$\alpha_C \approx 0.057$	$C_c \approx 2\times10^7$

The 1→2 Layer Transition

A conspicuous 'lump' at 10^{-5} PF-days marks transition from 1-layer to 2-layer networks, revealing that very shallow networks behave differently.

Cmin: The Corrected Compute

Training at fixed batch size is **not optimal**. We correct for this using **Cmin**:

$C_{min} = C / (1 + B/B_{crit})$

Compute needed if training at $B \ll B_{crit}$

This gives a **cleaner power law**:

$$L(C_{min}) = (C_{minc}/C_{min})^{\alpha_{minC}}$$

Exponent	Constant
$\alpha_{minC} \approx 0.050$	$C_{minc} \approx 3.1\times10^8$

Why Cmin matters: $L(C_{min})$ should be used for predictions, not $L(C)$.

EMPIRICAL RESULTS

The Combined L(N,D) Equation

Unified Scaling Law

$$L(N,D) = \left[\left(\frac{N_c}{N}\right)^{\alpha_N/\alpha_D} + \frac{D_c}{D}\right]^{\alpha_D}$$

This equation combines the individual scaling laws and governs the simultaneous dependence on model size N and dataset size D.

α_N exponent 0.076	α_D exponent 0.103
N_c constant 6.4×10^{13}	D_c constant 1.8×10^{13}

Three Design Principles

1. Rescaling

Must allow rescaling N_c, D_c for vocabulary/tokenization changes.

2. Limits

Must approach L(N) when $D\to\infty$ and L(D) when $N\to\infty$.

3. Analyticity

Should be analytic at $D=\infty$, with series expansion in 1/D with integer powers.

Excellent Fit

The equation provides an excellent fit to empirical data, with one exception: runs with only ~2×10^7 tokens.

Conjecture: This functional form may parameterize trained log-likelihood for other generative modeling tasks.

OVERFITTING ANALYSIS

Universality of Overfitting

Measuring Overfitting

Define overfitting relative to infinite data limit:

$$\delta L(N,D) \equiv L(N,D)/L(N,\infty) - 1$$

Where $L(N,\infty)$ is loss with full 22B token dataset (no overfitting).

Universal Dependence

Empirically, δL depends only on a specific combination of N and D:

$$\delta L \propto N^{0.74}/D$$

This follows from the L(N,D) scaling law, suggesting overfitting scales as 1/D.

The N:D Ratio Rule

To avoid overfitting penalty when increasing model size:

Increase data by ~5× for every 8× increase in N

Since $\alpha_N/\alpha_D \approx 0.74$, and $8^{0.74} \approx 5$.

Data Requirement Formula

To avoid overfitting when training to within threshold of convergence:

$$D > (5\times10^3) \times N^{0.74}$$

Example: Models < 10^9 params can be trained on 22B tokens with minimal overfitting.

Key Insight: Dataset size can grow sub-linearly in model size while avoiding overfitting. This does not represent maximally compute-efficient training (see Section 6).

TRAINING DYNAMICS

Critical Batch Size

Concept

From McCandlish et al. (2018): **B_{crit}** is the batch size where time and compute effi ciency are optimally balanced.

- **$B \ll B_{crit}$:** Minimizes compute, more serial steps
- **$B \gg B_{crit}$:** Minimizes steps, more compute
- **$B \approx B_{crit}$:** Optimal balance (2× steps, 2× examples)

Power-Law in Loss

$B_{crit}(L)$ is independent of model size, depending only on the loss:

$$B_{crit}(L) \approx B^* / L^{1/\alpha_B}$$

B^*	α_B
2×10^8	0.21

Adjustment Equations

To compare training runs at different batch sizes:

Minimum steps:

$$S_{min} = S / (1 + B_{crit}/B)$$

Minimum compute:

$$C_{min} = C / (1 + B/B_{crit})$$

Key Findings

- B_{crit} **independent of model size**, depends only on loss
- Critical batch size roughly matches gradient noise scale
- Doubles for every ~13% decrease in loss

Learning Curves: $L(N, S_{min})$

Universal Learning Curve

In infinite data limit, after warmup, loss follows:

$$L(N,S_{min}) = (N_c/N)^{\alpha_N} + (S_c/S_{min})^{\alpha_S}$$

α_S	S_c
0.76	2.1×10^{3}

Universality

- Parameters roughly **independent of model size**
- Fits best **late in training** (after warmup)
- Universal across all model sizes after transient

Extrapolation

By extrapolating early training, we can roughly predict the loss if trained much longer.

> **Practical Use:** Early training curves can predict final performance without full training runs.

Implications

Optimizer Dynamics

Power-law reflects interplay of optimizer and loss landscape.

Hessian Spectrum

Suggests Hessian eigenvalue density is roughly independent of model size.

Optimal Compute Allocation

The Key Finding

With fixed compute budget C, optimal performance comes from training **very large models** and stopping **significantly before convergence**.

> **Why?** Larger models are more sample efficient, reaching same loss with fewer steps per parameter.

Optimal Allocations

For compute-efficient training:

N ∝ C0.73min (Model size)

B ∝ C0.24min (Batch size)

S ∝ C0.03min (Serial steps)

D ∝ C0.27min (Dataset size)

What This Means

1. Model size grows very rapidly
5× larger model for 10× more compute

2. Batch size grows modestly
2× larger batch for 10× more compute

3. Serial steps grow negligibly
< 1.1× more steps for 10× more compute

4. Data requirements grow slowly
D ∝ C0.27 for efficient training

Contrast with Typical Practice

Researchers typically train smaller models to convergence. Compute-efficient training uses **2.7× more parameters** and **7.7× fewer steps** to reach same loss.

OPTIMAL ALLOCATION

Efficient vs. Inefficient Training

Two Training Regimes

Compute-Efficient

Stop at **$f = \alpha N/\alpha S \approx 10\%$** above converged loss

Typical Practice

Stop at **$f' \approx 2\%$** above converged loss

The Comparison

To reach the same fixed loss L, compute-efficient training uses:

2.7×	7.7×	65%
More parameters	Fewer steps	Less compute

Suboptimal Model Sizes

Models between **0.6× and 2.2×** the optimal size can be trained with only 20% incr ease in compute budget.

Smaller Models

Useful when inference cost matters

Larger Models

Train in fewer steps, allowing more parallelism

Trade-offs

- A 2.2× larger model requires 45% fewer steps at a cost of 20% more compute
- Smaller models: better for deployment, worse for training speed
- Larger models: faster training with sufficient hardware, higher inference cost

Sample Efficiency & Transfer

Sample Efficiency

Larger models are dramatically more sample-efficient:

- Reach same performance with **fewer optimization steps**
- Use **fewer data points** to reach target loss
- Improvement factor of ~100× from smallest to largest models

Transfer Learning

When evaluating on different text distributions:

- Results **strongly correlate** with training validation performance
- Transfer incurs a **constant offset** in loss
- Performance improves in **parallel** to training set

Why It Matters

This suggests that **big models may be more important than big data**. As models gr ow larger, they become increasingly efficient at extracting information from each data point.

Generalization Insights

Generalization depends almost exclusively on:

In-distribution validation loss

Does NOT depend on:

- Duration of training or proximity to convergence
- Model depth (when N is fixed)

LIMITATIONS

Contradictions & Caveats

The Apparent Contradiction

Two different data scaling requirements:

To avoid overfitting:

$D \propto C_{min}^{0.54}$

Compute-efficient training:

$D \propto C_{min}^{0.27}$

This implies our scaling laws must break down before the intersection point.

Intersection Point

Where $L(D(C_{min}))$ and $L(C_{min})$ intersect:

Compute $C^* \sim 10^4$	Parameters $N^* \sim 10^{12}$
Data $D^* \sim 10^{12}$	Loss $L^* \sim 1.7$

A Deeper Conjecture

The intersection may have deeper meaning:

If we cannot increase model size beyond N* without different data requirements...

Perhaps we've extracted all reliable information from natural language data.

In this interpretation, **L* provides a rough estimate for the entropy-per-token of natural language**.

Key Caveats

1. **No Theory:** No solid theoretical understanding for scaling laws.
2. **Bcrit Uncertainty:** Predictions far outside explored range are uncertain.
3. **Small Data:** Fits were poor for smallest datasets.
4. **Compute Estimate:** $C \approx 6NBS$ excludes context-dependent terms.
5. **Hyperparameters:** Some hyperparameters may not have been tuned optimally.

CONTEXT

Related Work & Connections

Power Laws in ML

Power laws arise from diverse sources and may be connected to our results:

- Density estimation and random forest models
- Exponents may relate to inverse of relevant features

Early Scaling Work

Power laws found between performance and dataset size in early work.

Model Size vs. Data Size

Closest to our work, but found **super-linear** scaling of data with model size, whereas we find **sub-linear** scaling.

Architecture Scaling

EfficientNet advocates scaling depth and width exponentially for optimal performance.

> **Our finding:** For language models, precise hyperparameters are unimportant compared to overall scale.

Deep Models as Ensembles

Deep models can function as ensembles of shallower models, potentially explaining our shape independence finding.

Large-Batch Training

Our work builds on empirical model of large-batch training, applying it to determine optimal allocations.

DISCUSSION

Future Directions & Open Questions

Universal Scaling?

Do these scaling relations apply to other generative modeling tasks?

Images | Audio | Video | Random Network Distillation

Unknown which results depend on natural language structure vs. which are universal.

Theoretical Framework

Need a 'statistical mechanics' underlying the observed 'thermodynamics'.

Such a theory might derive more precise predictions and systematic understanding of limitati ons.

Loss vs. Capability

Does continued loss improvement translate to task improvement? Smooth quantit ative change can mask major qualitative improvements: **"More is different"**.

Implications for Practice

Results strongly suggest that **larger models will continue to perform better** and be more sample efficient than previously appreciated.

Warrants further investigation into:
Model parallelism, sparsity, and methods for training giant models efficiently.

Model Parallelism Directions

Pipelining:
Splits parameters depth-wise, but requires larger batches

Wide Networks:
More amenable to parallelization, less serial dependency

Sparsity/Branching:
May allow faster training of large networks

Growing Networks:
Remain on compute-efficient frontier for entire training run

PRACTICAL GUIDANCE

Key Takeaways for Practitioners

1 Prioritize Model Size

When scaling up, allocate most resources to **model size**. Architectural details matt er less than total parameters N.

2 Don't Train to Convergence

Stop training when loss is **~10% above converged loss**. This is maximally compute -efficient.

3 Scale Data Sub-linearly

Increase dataset by **~5× for every 8×** increase in model size to avoid overfitting. $D \propto N^{0.74}$.

4 Use Large Batch Sizes

Use batch size **~1-2 million tokens** for largest models. $B_{crit} \propto L^{-4.8}$.

5 Architectural Flexibility

Depth, width, and attention heads matter less than total parameters. Aspect ratio can vary **40×** with minimal impact.

6 Exploit Sample Efficiency

Larger models are dramatically more sample efficient. **Big models > big data**.

7 Predict Performance

Use the scaling laws to **predict performance** before training. $L(N,D)$, $L(N,S_{min})$, and $L(C_{min})$ provide predictive framework.

Conclusion

Key Findings

Language model performance improves **smoothly and predictably** as we scale model size, data, and compute according to **power laws**.

These relationships hold across **many orders of magnitude** and provide a predictive framework for training large language models.

Optimal Training

Optimally compute-efficient training involves:

- Training **very large models**
- On **relatively modest amounts of data**
- And stopping **significantly before convergence**

The Scaling Hypothesis

We expect that **larger language models will continue to perform better** and be **much more sample efficient** than has been previously appreciated.

Thank you for your attention.