Scaling Laws for Neural Language Model

Jared Kaplan *
Johns Hopkins University, OpenAI
jaredk@jhu.edu

Sam McCandlish*
OpenAI
sam@openai.com

Tom Henighan
OpenAI
henighan@openai.com

Tom B. Brown
OpenAI
tom@openai.com

Benjamin Chess
OpenAI
bchess@openai.com

Rewon Child
OpenAI
rewon@openai.com

Scott Gray
OpenAI
scott@openai.com

Alec Radford
OpenAI
alec@openai.com

Jeffrey Wu
OpenAI
jeffwu@openai.com

Dario Amodei
OpenAI
damodei@openai.com

HUMANE Lab
김건수

2025.01.17

Abstract

- **Objective:** Study empirical scaling laws for language model performance on cross-entropy loss
- **Key Findings:**
  - **Power-law Scaling:** Loss decreases as a power-law with model size, dataset size, and trainin g compute.
  - **Minimal Architectural Impact:** Variations in network width or depth have negligible effects within a broad range.
  - **Overfitting Dependence:** Overfitting scales predictably with model and dataset size.
  - **Training Speed Dependence:** Training speed scales predictably with model size.
- **Compute Budget Optimization:**
  - Larger models are more sample-efficient.
  - Compute-efficient training uses large models with modest data sizes, halting before converg

Introduction: Key Points

- **Language Modeling and AI:** Language is a natural domain for AI due to its ability to express reasoning tasks and leverage vast text datasets for unsupervised generative modeling.
- **Advances in Deep Learning:** Recent progress in deep learning has led to near-human performance in tasks like coherent multi-paragraph text generation.
- **Scaling Factors:** Language modeling performance is influenced by model size, compute power, data availability, and architecture, with a focus on the Transformer architecture
- **Power-law Scalings:** Performance trends span over seven orders of magnitude, with precise power-law relationships observed across model size, dataset size, and compute

Introduction: Summary of Findings

1. **Performance and Scale:**
   - **Strong Dependence on Scale:** Model performance primarily depends on parameters (N), dataset size (D), and compute (C), with minimal sensitivity to architectural details.
   - **Smooth Power-law Trends:** Performance scales predictably with N, D, C, showing no si gns of deviation over six orders of magnitude.
2. **Overfitting and Training Efficiency:**
   - **Universality of Overfitting:** Overfitting scales predictably; optimal balance between N a nd D requires$D \propto N^{0.74}$.
   - **Universality of Training:** Training curves follow power-laws, enabling loss predictions b ased on early training progress.

Introuduction: Summary of Findings

3. **Sample Efficiency and Convergence:**
   - **Sample Efficiency:** Larger models are significantly more efficient, requiring fewer steps and data to reach equivalent performance.
   - **Inefficient Convergence:** Optimal compute-efficient training involves large models and early stopping, with data requirements growing slowly ($D \propto C^{0.27}$).
4. **Optimal Batch Size:**
   - The ideal batch size is determined by the gradient noise scale, approximately 1-2 millio n tokens at convergence for large models.

Introduction: Scaling Laws

1. **Loss Relationships:**
   - Loss vs. Model Size: $L(N) = (N_c/N)^{\alpha_N}, where\ \alpha_N \sim 0.076.$
   - Loss vs. Dataset Size: $L(D) = (D_c/D)^{\alpha_D}, where\ \alpha_D \sim 0.095.$
   - Loss vs. Compute: $L(C_{min}) = (C_{min,c}/C_{min})^{C_{min,c}}, where\ \alpha_{min,C} \sim 0.050.$
2. **Combined Loss Equation:**
   - $L(N, D) = (\frac{N_C}{N})^{\alpha_N} + (\frac{D_C}{D})^{\alpha_D}.$
3. **Optimal Training under Fixed Compute Budget:**
   - Scaling relationships: $N \propto C^{\alpha_{min,C}/\alpha_N}$, $\mathrm{B} \propto C^{\alpha_{min,C}/\alpha_B}$, $\mathrm{S} \propto C^{\alpha_{min,C}/\alpha_S}$, $D \propto B \cdot S$
   - With $\alpha_C^{min} = 1/(\frac{1}{\alpha_S} + \frac{1}{\alpha_S} + \frac{1}{\alpha_S}).$

Introduction: Practical Implications

- Larger models should be prioritized for improved performance and sample efficiency.
- Training should allocate most compute to increasing model size while keepi ng dataset size and training steps relatively modest.
- Power-law relationships provide predictive tools for loss optimization, comp ute allocation, and scaling strategies.

Methods: Model Training Setup

- **Dataset:**
  - Trained on WebText2, an extended version of WebText, which includes Reddit outbound links from Jan–Oct 2018 (minimum 3 karma).
  - Dataset stats:
    - **Size:** 96 GB of text (~20.3M documents).
    - **Tokens:** 2.29×10^{10} tokens (6.6×10^{8} reserved for testing).
    - **Vocabulary:** 50,257 tokens (byte-pair encoding).
  - Also tested on datasets like Books Corpus, Common Crawl, English Wikipedia, and publi c Internet books.

Methods: Model Training Setup

- **Architecture:**
  - Focused on **decoder-only Transformers**; comparisons made with LSTMs and Universal Transformers.
  - Performance metric: Autoregressive log-likelihood (cross-entropy loss) over a 1024-toke n context.

Methods: Transformer Parameterization

- **Hyperparameters:**
  - Layers (n_{layers}), residual stream dimension (d_{model}), feed-forward layer (d_{ff}), attention o utput (d_{attn}), and attention heads (n_{heads}).
  - Context size (n_{ctx}) = 1024.
- **Model Size** (N) **Approximation:**
  - $N \approx 12 \cdot n_{layers} \cdot d_{model}^2$.
  - Embedding and positional parameters excluded for cleaner scaling laws.
- **Compute Estimate:**
  - Forward pass: $C_{forward} \approx 2N + 2n_{layers} \cdot n_{ctx} \cdot d_{attn}$
  - Training compute: $C \approx 6N$FLOPs per token (accounts for forward and backward passes).

Methods: Training Procedures

- **Optimizer:**
  - Used Adam for models ≤ 1 billion parameters; Adafactor for larger models.
- **Training Steps:** Fixed at 250,000 steps.
- **Batch Size:** 512 sequences of 1024 tokens.
- **Learning Rate:**
  - Schedule: Linear warmup (3,000 steps) followed by cosine decay to zero.
  - Convergence results largely independent of learning rate schedules.

Methods: Key Compute and Parameter Observations

- **Efficiency Considerations:**
  - For $d_{model} \gg n_{ctx}/12$, context-dependent terms contribute negligibly to compute.
- **Parameter Counts:**
  - Table summarizes contributions from embedding, attention, and feed-forward layers to total parameters and FLOPs.

Operation	Parameters	FLOPs per Token
Embed	$(n_{\text{vocab}} + n_{\text{ctx}})\, d_{\text{model}}$	$4d_{\text{model}}$
Attention: QKV	$n_{\text{layer}} d_{\text{model}} 3d_{\text{attn}}$	$2n_{\text{layer}} d_{\text{model}} 3d_{\text{attn}}$
Attention: Mask	—	$2n_{\text{layer}} n_{\text{ctx}} d_{\text{attn}}$
Attention: Project	$n_{\text{layer}} d_{\text{attn}} d_{\text{model}}$	$2n_{\text{layer}} d_{\text{attn}} d_{\text{embd}}$
Feedforward	$n_{\text{layer}} 2d_{\text{model}} d_{\text{ff}}$	$2n_{\text{layer}} 2d_{\text{model}} d_{\text{ff}}$
De-embed	—	$2d_{\text{model}} n_{\text{vocab}}$
Total (Non-Embedding)	$N = 2d_{\text{model}} n_{\text{layer}} (2d_{\text{attn}} + d_{\text{ff}})$	$C_{\text{forward}} = 2N + 2n_{\text{layer}} n_{\text{ctx}} d_{\text{attn}}$

Empirical Results and Basic Power Laws

Factors Studied

- **Model size:** Ranged from 768 to 1.5 billion non-embedding parameters.
- **Dataset size:** Spanned from 22 million to 23 billion tokens.
- **Model shape:** Included variations in depth, width, attention heads, and feed -forward dimensions.
- **Context length:** Typically 1024 tokens but also shorter contexts were tested.
- **Batch size:** Varied from 2^{19} to measure critical batch size effects.

Key Findings

1. **Transformer Shape Independence:**
   - Performance is weakly dependent on shape parameters $(n_{layer}, n_{heads}, d_{ff})$ when total non-embedding parameters (N) are fixed.
   - Small differences in shape (e.g., depth-to-width ratios) have minimal impact on loss.
2. **Performance Scaling with Model Size (N):**
   - Test loss follows a predictable power-law: $L(N) \approx (\frac{N_c}{N})^{\alpha N}$.
   - Including embedding parameters obscures trends; excluding them reveals clear scaling.
   - Transformers outperform LSTMs for longer contexts, leveraging improved use of long-range dependencies.

Key Findings

3. **Generalization Across Data Distributions:**
   - Loss on out-of-distribution datasets (e.g., Wikipedia) scales smoothly with model size.
   - Generalization is strongly tied to in-distribution validation loss and independent of train ing duration or proximity to convergence.

4. **Performance Scaling with Dataset Size** (D)**:**
   - Test loss decreases predictably with dataset size: $L(D) \approx (\frac{D_c}{D})^{\alpha D}$.
   - Training on larger datasets improves performance but shows diminishing returns withou t scaling model size.

Key Findings

5. **Performance Scaling with Compute** (C)**:**

- Test loss follows a power-law relationship: $L(C) \approx (\frac{C_c}{C})^{\alpha C}$.
- Larger models are more sample-efficient, achieving better performance with fewer toke ns processed.

Comparison

- • **LSTMs vs. Transformers:**
  - LSTMs match Transformer performance for early tokens in context but plateau with lon ger sequences.
  - Transformers maintain improvement throughout the entire context window.
- **Generalization:**
  - Model size consistently improves test loss on other datasets, with minimal offsets from training distribution loss.
  - Generalization trends remain stable across different training phases.

Charting the Infinite Data Limit and Overfitting

Objective

- Investigate how test loss scales with model size (N) and dataset size (D) sim ultaneously.
- Empirically validate the proposed scaling law for L(N, D).

Proposed Equation for $L(N, D)$

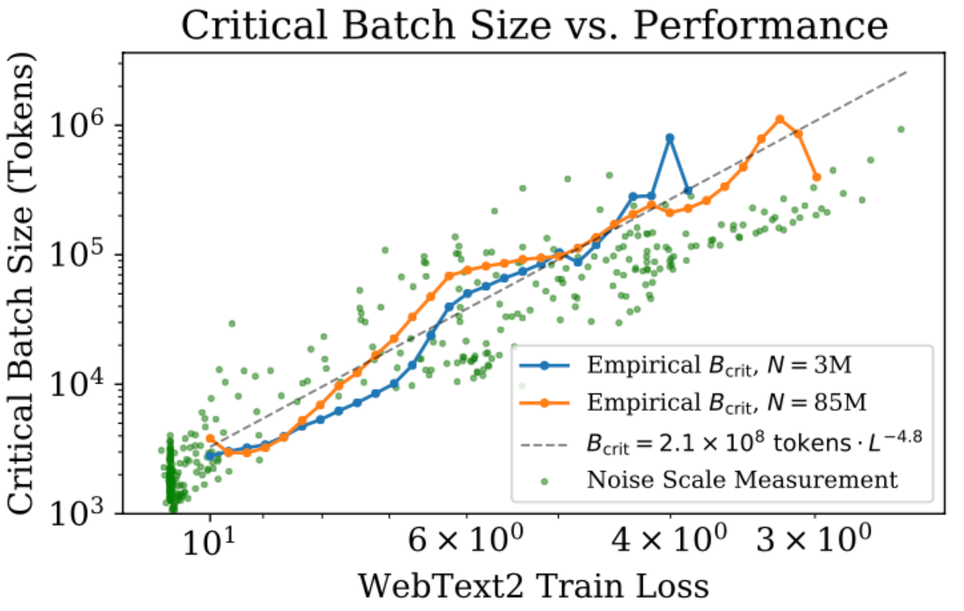
- **Equation:** $L(N, D) = \frac{N_C}{N^{\alpha_N}} + \frac{D_C}{D^{\alpha_D}}$.
- **Principles Behind the Equation:**
  1. Allows rescaling with changes in vocabulary size or tokenization.
  2. Models loss limits: $D \to \infty \Longrightarrow L(N); N \to \infty \Longrightarrow L(D)$.
  3. Analytic at $D \to \infty$, allowing series expansion in 1/D.

Results

- **Fit Parameters:**
  - $\alpha_N = 0.076, \alpha_D = 0.103, N_c = 6.4, D_c = 1.8 \times 10^{13}$.
- **Key Findings:**
  - **Overfitting:**
    - For large D, loss follows a power law in N.
    - For small D, performance plateaus as N increases, showing overfitting.
  - **Critical Dataset Size:**
    - To avoid overfitting within a 0.02 loss, dataset size grows sub-linearly with model size: $D \propto N^{0.74}$.
  - Models < 10^9 parameters show minimal overfitting on a 22B token dataset.

Critical Batch Size

- **Observation:**
  - Critical batch size (B_{crit})follows a power law with loss.
  - B_{crit} doubles for every 13% decrease in loss.
  - Independent of model size; aligns with predictions from gradient noise scale.



Implications

- Dataset size can grow sub-linearly with model size to avoid overfitting.
- Larger datasets mitigate overfitting but are not always compute-efficient.
- Regularization (e.g., dropout) was not optimized, leaving room for further im provements.

Scaling Laws with Model Size and Training Time

1. Critical Batch Size ($\boldsymbol{B_{crit}}$)

- **Definition:**
  - Critical batch size allows optimal time/compute tradeoff for training.
  - Increasing batch size (B):
    - $B \leq B_{crit}$: Minimal degradation in compute efficiency.
    - $B > B_{crit}$: Diminishing returns with increased batch size.
- **Relation with Loss:**
  - $B_{crit}(L) \approx \frac{B_*}{L^{1/\alpha B}}$ (where $B^* \approx 2 \times 10^8, \alpha_B \approx 0.21$).
  - Critical batch size is independent of model size and only depends on the loss (L).

Scaling Laws with Model Size and Training Time

2. Universal Training Step (S_{min})

- **Relation Between Training Steps and Data:** $\left(\frac{S}{S_{min}} - 1\right)\left(\frac{E}{E_{min}} - 1\right) = 1$
- S_{min} : Minimum steps to reach a target loss.
- E_{min} : Minimum data examples required.
- Training at B_{crit} ensures optimal time/compute tradeoff ($2S_{min}, 2E_{min}$).

Scaling Laws with Model Size and Training Time

3. Loss Scaling with Model Size and Steps

- **Equation:**
  - $L(N, S_{min}) = (\frac{N_C}{N})^{\alpha N} + (\frac{S_C}{S_{min}})^{\alpha S}$
- **Fit Parameters:**
  - $\alpha_N = 0.077, \alpha_S = 0.76, N_c = 6.5 \times 10^{13}, S_c = 2.1 \times 10^{3}$.
- **Key Observations:**
  - Loss scales predictably with both model size (N) and training steps (S_{min}).
  - Larger models trained for fewer steps can outperform smaller models trained longer.

Scaling Laws with Model Size and Training Time

4. Early Stopping and Data Efficiency

- **Early Stopping Criterion:**
  - $S_{stop}(N,D) \geq \frac{S_c}{[L(N,D)-L(N,\infty)]^{1/\alpha S}}$
  - Ensures efficient training by minimizing overfitting.
  - Larger datasets and models reduce required steps for optimal loss.

Scaling Laws with Model Size and Training Time

5. Implications

- **Efficiency Insights:**
  - Train at B_{crit} for optimal compute usage.
  - Scaling laws provide a framework for balancing model size, batch size, and training step s.
- **Practical Applications:**
  - Predictive power of scaling laws aids in optimizing training compute allocation.
  - Early stopping reduces unnecessary computation and data usage.

Optimal Allocation of the Compute Budget

1. Key Observations

- **Optimal Compute Allocation:**
    - Training efficiency improves when compute is allocated optimally between model size (N) and data processed ($2B_{crit}, S_{min}$).
    - Loss scaling improves when adjusted for critical batch size (B_{crit}).

Optimal Allocation of the Compute Budget

2. Optimal Model Size ($N(C_{min})$)

- **Scaling Relation:**
  - $N(C_{min}) \propto (C_{min})^{0.73}$
  - A 10x increase in compute results in a 5x increase in model size, while data usage grow s modestly (~2x).
- **Training Steps:**
  - $S_{min} \propto (C_{min})^{0.03}$, indicating very slow growth in steps.

Optimal Allocation of the Compute Budget

3. Predictions from $\mathbf{L}(\boldsymbol{N}, \boldsymbol{S_{min}})$

- **Loss Scaling with Compute:**
  - $L(C_{min}) = (\frac{C_C^{min}}{C_{min}})^{\alpha_C^{min}}.$
- **Scaling laws:**
  - $N(C_{min}) \propto {C_{min}}^{\alpha_C^{min}/\alpha N} \approx (C_{min})^{0.71}$
- **Empirical Agreement:**
  - Predictions align closely with observed data, validating the scaling laws.

Optimal Allocation of the Compute Budget

4. Contradictions and Limitations

- **Scaling Breakdown:**
  - At extreme scales, predictions from $L(C_{min})$ and $L(D)$ intersect, indicating a breakdown.
  - Intersection estimates:
    - Compute: $C^{*}{\sim}10^{4}$ PF-Days | Model size:$N^{*}{\sim}10^{12}$ parameters.
    - Dataset size: $D^{*}{\sim}10^{12}$ tokens, Loss: $L^{*}{\sim}1.7$ nats/token.
- **Interpretation of Intersection Point:**
  - May represent the maximum achievable performance under current scaling laws.
  - Suggests the limit of reliable information extractable from natural language data.

Optimal Allocation of the Compute Budget

5. Implications for Compute-Efficient Training

- **Model Size Priority:**
  - Scaling compute should focus on increasing model size (N) rather than training steps.
- **Future Challenges:**
  - Dataset growth must match model scaling to avoid overfitting.
  - Current scaling laws may need adjustments to address data bottlenecks at extreme scal es.

Discussion

1. Key Observations

- **Consistent Scaling Laws:**
  - Loss scales predictably with non-embedding parameters (N), dataset size (D), and optim ized compute (C_{min})
  - Weak dependence on architectural and optimization hyperparameters.
  - Diminishing returns observed with increasing scale.
- **Predictive Framework:**
  - Scaling laws predict compute scaling, overfitting magnitude, early stopping steps, and d ata requirements.
  - Analogous to the “ideal gas law” in physics, providing universal macroscopic insights in dependent of system specifics.

Discussion

2. Broader Implications

- **Generative Modeling:**
  - Scaling laws may apply to other domains (e.g., images, audio, video) and tasks (e.g., ran dom network distillation).
  - Requires exploration to distinguish language-specific results from universal patterns.
- **Theoretical Insights:**
  - Developing a theoretical framework akin to "statistical mechanics" for scaling laws could offer precise predictions and deeper understanding.

Discussion

3. Qualitative vs. Quantitative Improvements

- **“More is Different”:**
  - Smooth improvements in loss may mask qualitative leaps in language model capabiliti es.
  - Continued loss reduction may lead to significant breakthroughs in task performance.

Discussion

4. Larger Models and Efficiency

- **Big Models > Big Data:**
  - Larger models are more sample-efficient than previously realized.
  - Focus on scaling models rather than solely increasing data size.
- **Model Parallelism:**
  - Promising approaches to train large models efficiently:
    - **Pipelining:** Depth-wise parameter splitting across devices.
  - **Wide Networks:** Better suited for parallelization.
  - **Sparsity/Branching:** Enable faster training via model parallelism .
  - **Dynamic Networks:** Growing networks during training to maintain compute efficiency.

Discussion

5. Future Directions

- Investigate scaling laws in diverse domains and tasks.
- Develop theoretical underpinnings for observed scaling laws.
- Explore the qualitative impact of quantitative improvements in loss.
- Innovate parallelism techniques for efficient training of very large models.