Scaling Laws for Neural Language Models

Jared Kaplan, Sam McCandlish
OpenAI

Jennifer Zhuang

Mathematical Foundations of Machine Learning

Fall 2024

Introduction

- Research Project
- Leading Question

Scaling Laws for Neural Language Models

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

What if we could predict how improving model size, data, and compute directly impacts the performance of language models?

In this paper, our researchers explore the scaling laws that govern these relationships and how they shape the future of neural language models

Background

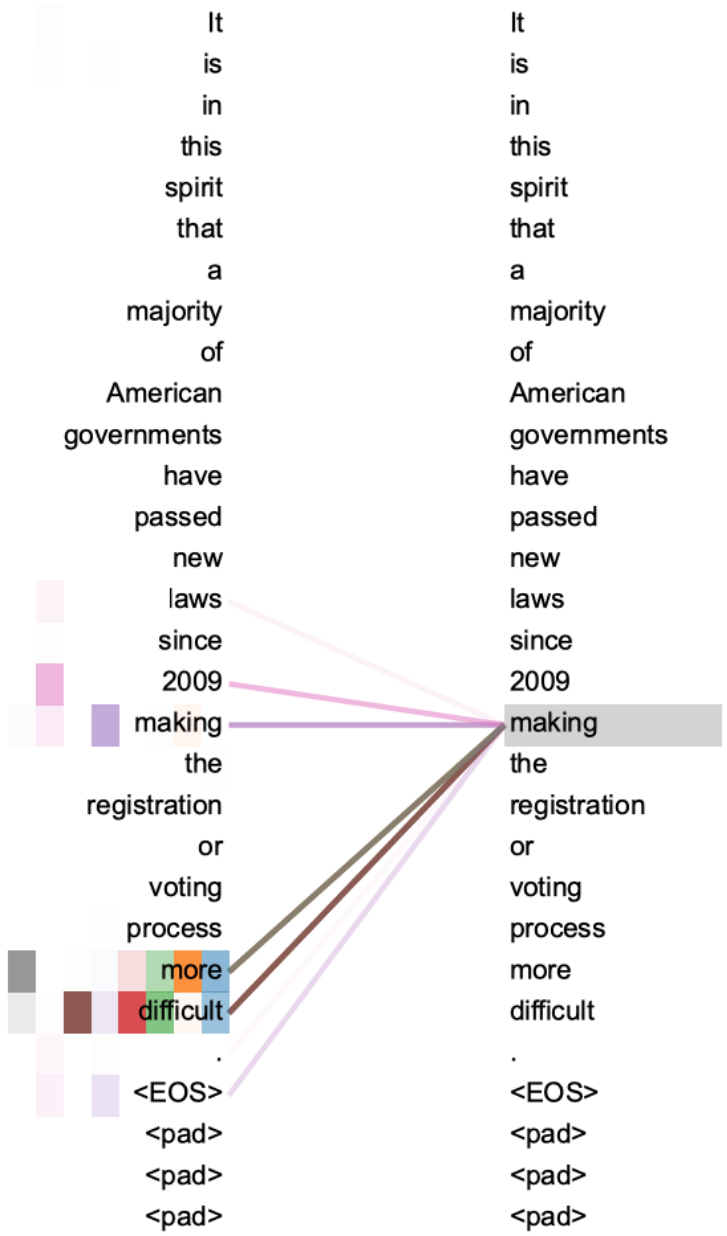
- Transformer Based Neural Language Models
- Loss Function
- Scaling Laws
- Compute Budget
- The Potential of LLMs

What is a Transformer Based Neural Language Model

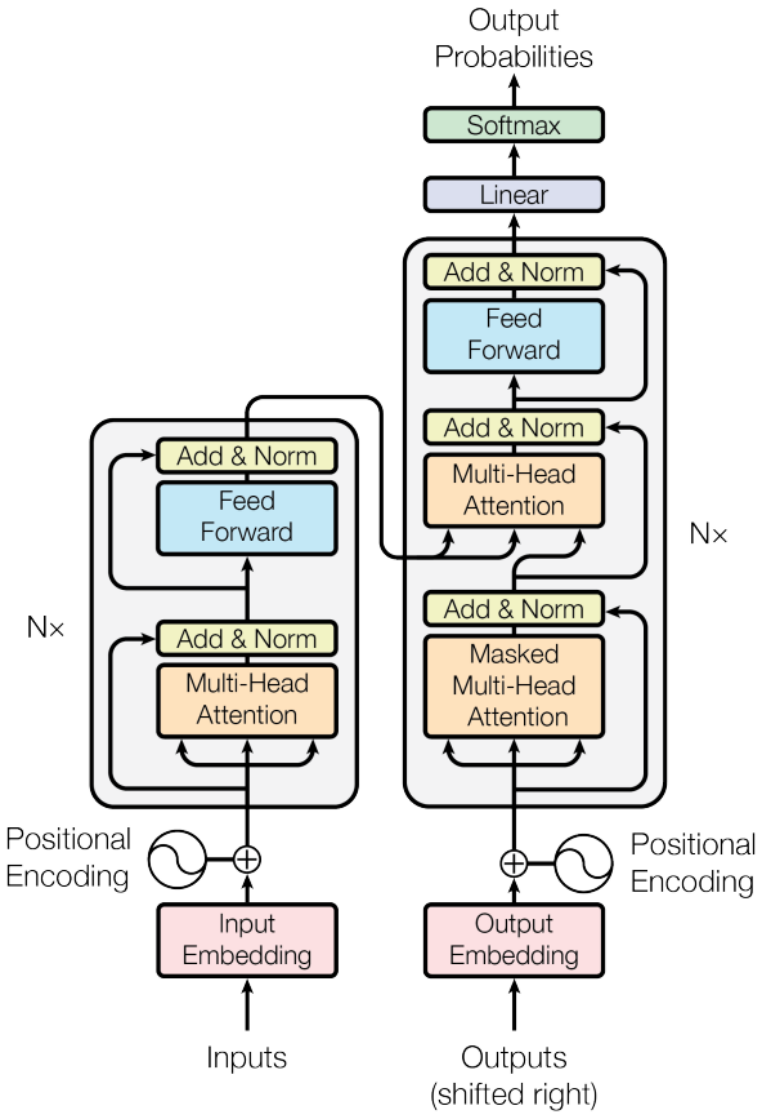
- A type of machine learning model designed to process and generate natural language using a mechanism called ***self-attention***
- The foundation for language models like GPT, BERT, and T5
- Key Components:
  - **Self-Attention Mechanism**: Allows the model to weigh the importance of different words in a sentence relative to each other, enabling it to understand context better.
  - **Encoder-Decoder Architecture**: The encoder processes input sequences, while the decoder generates output sequences
  - **Positional Encoding**: Transformers process words in parallel (not sequentially), using positional encodings to keep track of the order of words

Attention Mechanism

- The model calculates the importance of each token relative to every other token in the input
- Every word considers all other words in the sequence, enabling the model to identify relationships, even for distant tokens and understand the context of the entire input
- The model process entire sequences in parallel, unlike RNNs, and captures relationships between tokens effectively
- Since transformers process tokens in parallel, positional encoding adds information about the order of the tokens, ensuring the model understands sequence relationships



(Vaswani et al., 2017)



(Vaswani et al., 2017)

Transformer Model Architecture

- **Input Processing**: Input (e.g., a sentence) is split into smaller units called tokens, and converted into numerical representations (vectors) using an embedding layer
- **Encoder**: processes the input and generates a representation that captures the relationship between tokens
  - N identical layers working in parallel
  - Attention mechanism feeds data to feedforward neural network which extracts deeper patterns and relationships between tokens
- **Decoder**: generates the output sequence
  - Also has N identical layers working in tandem
  - Decoder uses contextual representation from encoder with previous generated tokens to predict next token
- **Output and Softmax:** highest probability token selected
- **Training:** The transformer adjusts its weights by minimizing a *loss function* using techniques like gradient descent and backpropagation

What is a Loss Function

- "We optimize the autoregressive log-likelihood (i.e. cross-entropy loss)" (Kaplan et al.)
- Cross-entropy loss: a mathematical measure used to evaluate the difference between the model's predicted probabilities and the actual probabilities
- Serves as a measure of model performance
    - Lower loss means model's predicted probabilities align more closely with the true probabilities

$$L = -\frac{1}{N}\sum_{i=1}^{N}\log p(y_i|y_{<i})$$

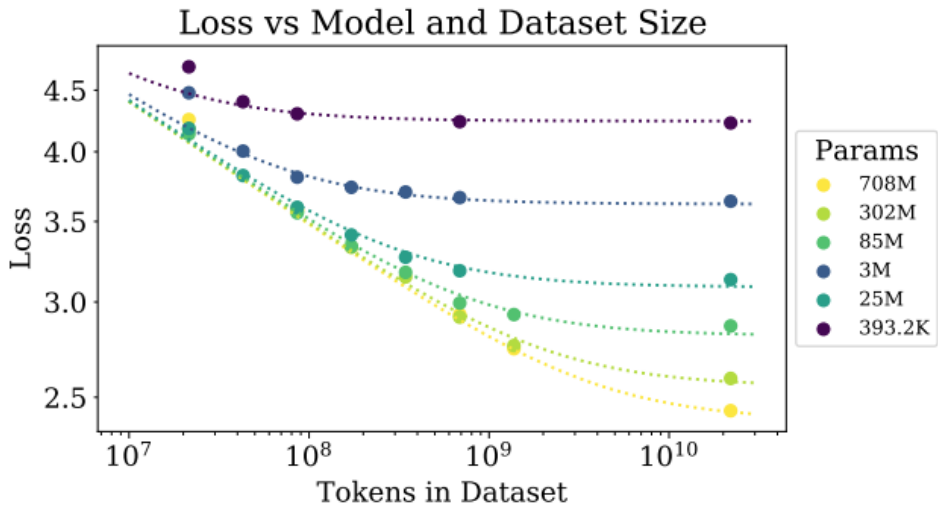
y_i : The i^{th} token in the sequence

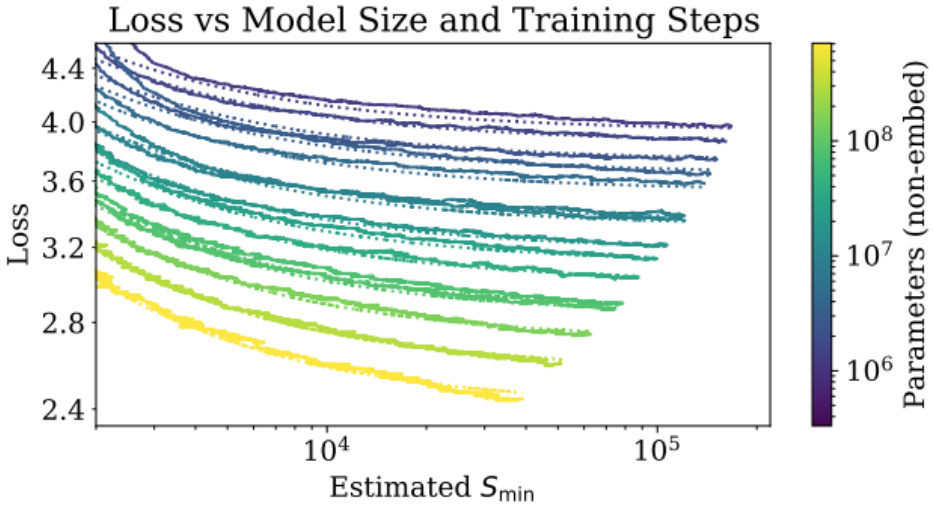
$p(y_i|y_{<i})$: The model's predicted probability of token y_i given all previous tokens in the sequence $(y_{<i})$

The loss is averaged over all tokens in the sequence

What are Scaling Laws

- "We study empirical scaling laws for language model performance" (Kaplan et al.)
- "The ***loss scales as a power-law*** with model size, dataset size, and the amount of compute used for training" (Kaplan et al.)
- Scaling laws: Mathematical rules that describe how changes in certain variables (e.g., model size, data size) affect performance.
- Empirical: Based on observations or experiments rather than purely theoretical ideas
- In simpler terms: We are observing trends in how performance scales as models grow larger or are trained on more data or provided more compute



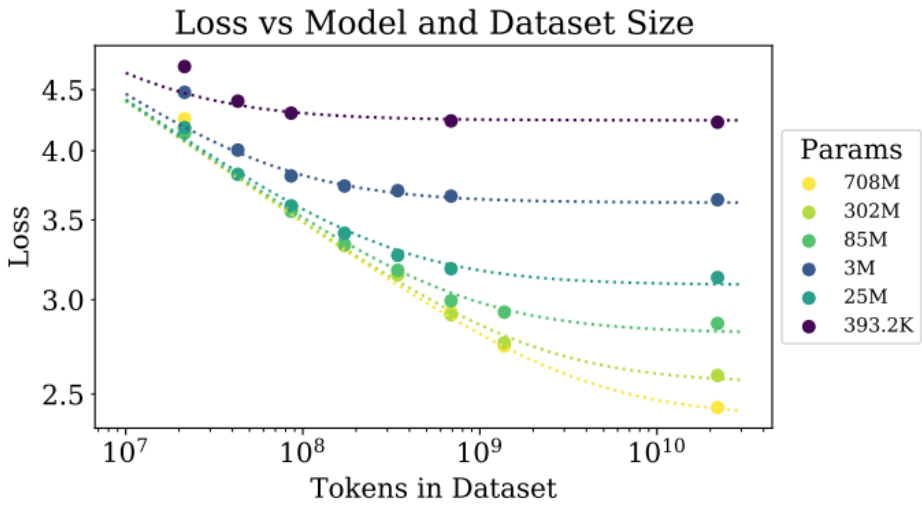


What are Power Laws?

- A **power law** describes a relationship where one quantity varies as a power of another.
- For language models, **loss** (L) scales with variables like model size (N), dataset size (D), and compute (C)

$$L(X) \propto \frac{1}{X^{\alpha_X}}$$

α_X : Scaling exponent that determines how quickly loss decreases as X grows



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

Role of Scaling Laws in LLMs

- Guiding Optimal Resource Allocation
  - Scaling laws help determine how to optimally use compute, data, and model size to achieve maximum performance
  - Ensures that LLMs are trained effectively without wasting resources
- Maximizing Performance
  - Provide a roadmap for designing and training LLMs to minimize loss and improve accuracy
- Impact of Vast Data Availability
  - The abundance of text data
  - Scaling laws guide how to leverage this data efficiently without overloading compute resources

The Potential of Large Language Models

- Language as a Universal Medium
  - Most reasoning tasks can be efficiently expressed and evaluated in language
  - Focusing on LLMS can open vast possibilities across numerous domains
- Unsupervised Learning Potential
  - There are vast amounts of text available globally, providing rich data for training large models
  - We can capitalize on the wealth of data without requiring costly human-labeled annotations
- Recent Advancements & State of the Art Performance
  - Progress in deep learning has driven significant improvements in language models
  - Models now approach human-level performance on tasks like text composition, reasoning, and contextual understanding
  - Key to unlocking emergent capabilities: Sparks of Artificial General Intelligence (OpenAI, 2024)

Experimental Design

- Training and Testing Procedures
- Parameterizing Transformers and Compute
- Scale Factors

Training and Testing Dataset

- **Training Dataset:** (WebText2)
  - **Source**: Web scrape of outbound links from Reddit (2017–2018) with ≥3 karma (heuristic for quality)
  - **Processing**:
    - Extracted text using Newspaper3k (Python library).
    - Applied a reversible tokenizer to process text into tokens.
  - **Statistics:**
    - Documents: 20.3M
    - Text Size: 96 GB (~16.2 billion words).
    - Tokens: 22.9 billion tokens.
    - Vocabulary Size: 50,257
- **Testing Dataset**
  - **Subset from WebText2:** Reserved 660M tokens as a test set.
  - **Additional Sources**: Books Corpus, Common Crawl, English Wikipedia, Public Internet Books

Training and Testing Procedures

- Training Procedures
  - **Model Type**: Decoder-only Transformer models
  - **Optimizer**:
    - Adam Optimizer: Adjusts model parameters to minimize errors, efficient for large datasets
    - AdaFactor: Used for training the largest models (1B+ parameters)
  - **Training Process**:
    - Steps: Trained for 250,000 steps (each step updates model parameters after processing a batch of data)
    - Batch Size: Processes 512 sequences per step, each with 1024 tokens
- Testing Procedures
  - **Evaluation Metric**: autoregressive log-likelihood averaged over a 1024-token context

Factor	Range/Details	Notation
Model Size	768 to 1.5 billion parameters	N
Dataset Size	22 million to 23 billion tokens	D
Shape	Depth, width, attention heads, feed-forward dimensions	d_{ff}, n_{layer}, n_{heads}
Context Length	1024 tokens (default), shorter contexts	n_{ctx}
Batch Size	2^{19}, varied for experiments	B
Training Compute	Estimated as $C \approx 6NBS$, quoted in PF-days (1 PF-day = 8.64×10^{19} FLOPs)	C

Scaling Factors

- To understand how performance changes with model size, dataset size, and compute used for training we vary several key factors
- These factors can be thought of as the "independent variables" in the experiment and performance as the "dependent"

Parameterizing Model Size

- How to define the size of a transformer model such that we can measure scaling behavior as the model increases in size and complexity
- First, define the structure by specifying a set of key hyperparameters
- Then, we use N to denote the model size, defined as the number of non-embedding parameters

- n_{layer} : Number of layers
- d_{model} : Dimension of the residual stream (main data flow)
- d_{ff} : Dimension of the intermediate feed-forward layer
- d_{attn} : Dimension of the attention output
- n_{heads} : Number of attention heads per layer

↓

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

↓

$$N = 12 \cdot n_{\text{layer}} \cdot d_{\text{model}}^2$$

$$\text{where } d_{\text{model}} = \frac{d_{\text{ff}}}{4} = d_{\text{attn}}$$

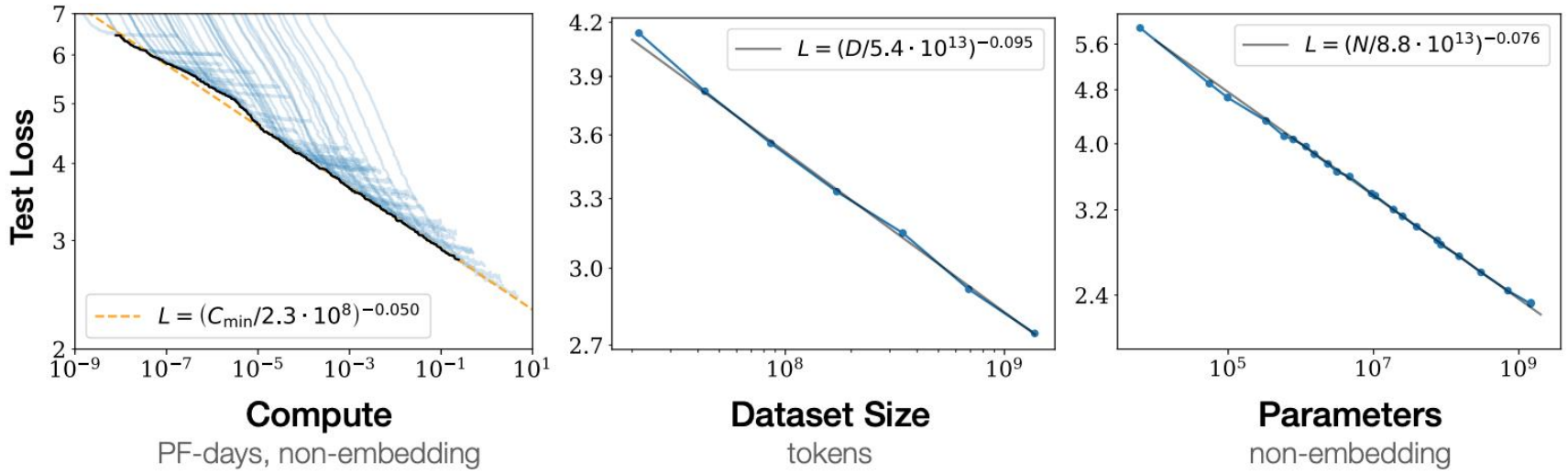
Operation	Parameters	FLOPs per Token
Embed	$(n_{\text{vocab}} + n_{\text{ctx}})\, d_{\text{model}}$	$4d_{\text{model}}$
Attention: QKV	$n_{\text{layer}} d_{\text{model}} 3d_{\text{attn}}$	$2n_{\text{layer}} d_{\text{model}} 3d_{\text{attn}}$
Attention: Mask	—	$2n_{\text{layer}} n_{\text{ctx}} d_{\text{attn}}$
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)	$N = 2d_{\text{model}} n_{\text{layer}} (2d_{\text{attn}} + d_{\text{ff}})$	$C_{\text{forward}} = 2N + 2n_{\text{layer}} n_{\text{ctx}} d_{\text{attn}}$

Parameterizing Compute

- Compute refers to the amount of computational effort required to train or run a machine learning model.
- It's typically measured in terms of floating-point operations (FLOPs), which count the mathematical operations (like addition and multiplication) that the model performs.
- Higher compute requirements mean more time, energy, and hardware resources are needed to train the model.
- Larger models generally require more compute, but they can achieve better performance if sufficient compute is available

Key Findings

- Smooth Power Laws
- Loss' weak dependence on model shape
- Charting the Infinite Data Limit and Overfitting
- Sample Efficiency

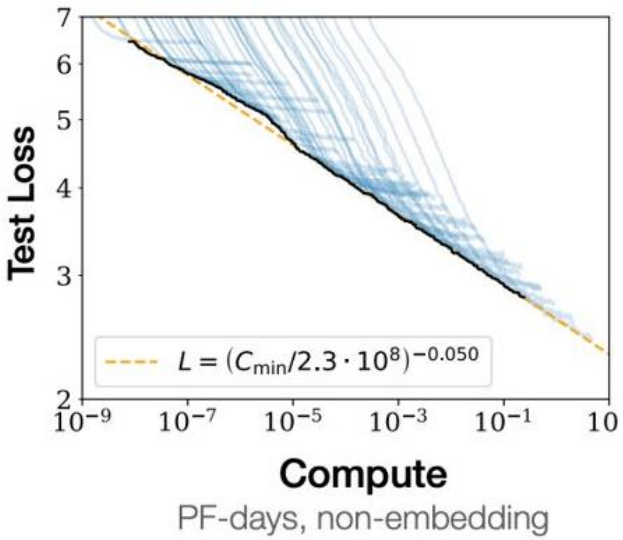


Smooth Power Laws

- Language modeling performance improves smoothly as we increase the model size N, dataset size D, and amount of compute C used for training
- All three factors must be scaled up in tandem
- Observed no signs of deviation from these trends on the upper end

Performance vs Compute

- Compute is measured in **PF-days** (PetaFLOP-days), representing the total computational effort used for training.
- Each blue line represents a **different training run** with unique configurations of model size, dataset size, and batch size.
- The lines follow a downward trend, indicating that increasing compute generally improves performance (reduces test loss).
- The black line is the **aggregate trend** across all training runs, showing the average relationship between compute and test loss.
- It smooths out variations from the individual runs.
- The dashed line represents a **power-law fit** to the data, defined by the equation

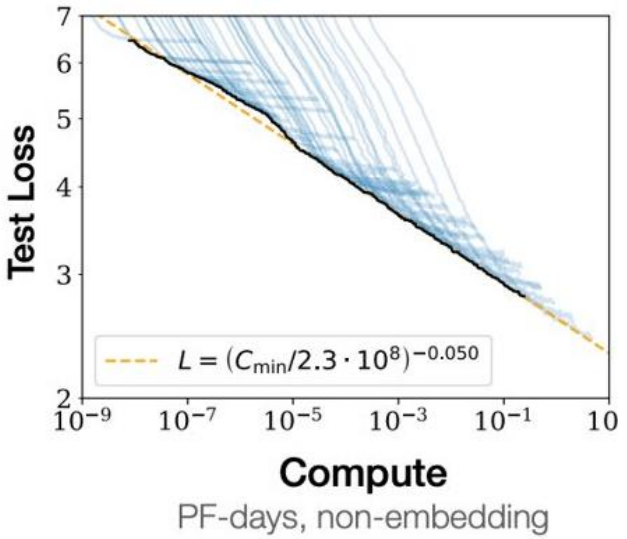


Performance vs Compute

- The power law fit to the data is:

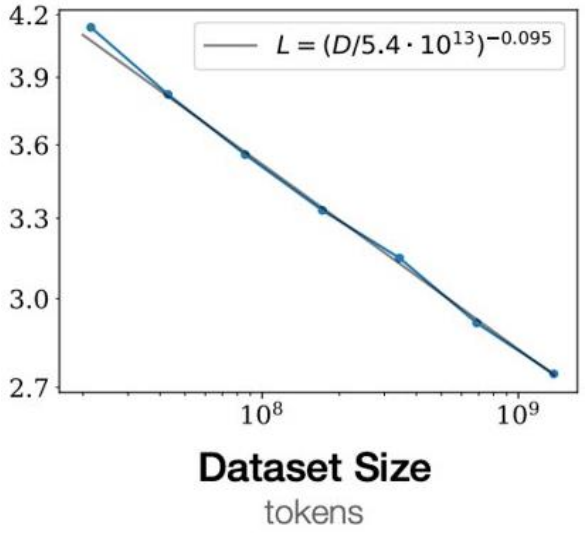
$$L = \left(\frac{C_{\text{min}}}{2.3 \times 10^8} \right)^{-0.050}$$

  - C_{min}: The minimum compute required to achieve a specific loss value.
  - The exponent (-0.050) indicates how quickly the loss decreases as compute increases.
- The data strongly suggests that sample efficiency improves with model size



Performance vs Dataset Size

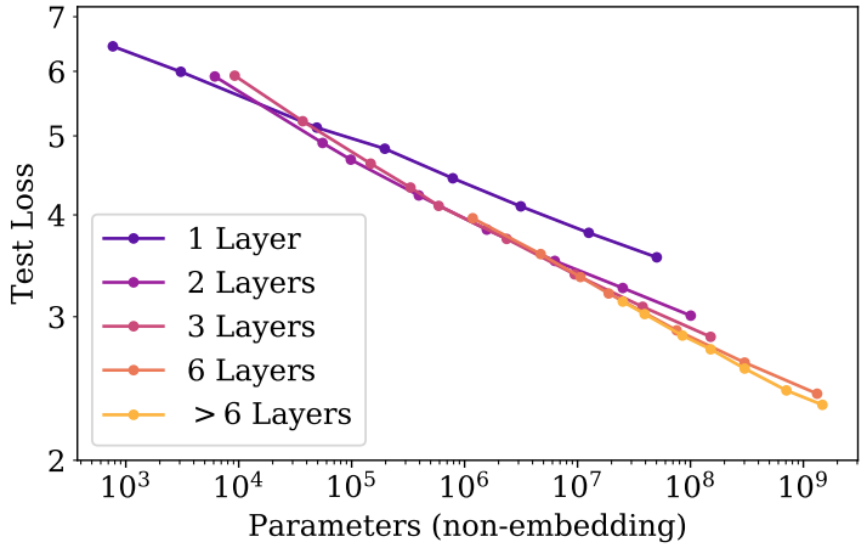
- For this trend, a model with fixed size (nlayer, nembd) = (36, 1280) is trained on fixed subsets of the WebText2 dataset
- Training is stopped once the test lost ceased to decrease
- We can see the resulting losses can also be fit with a simple, predictable power law curve



$$L = \left(\frac{D}{5.4 \times 10^{13}} \right)^{-0.095}$$

Performance vs N

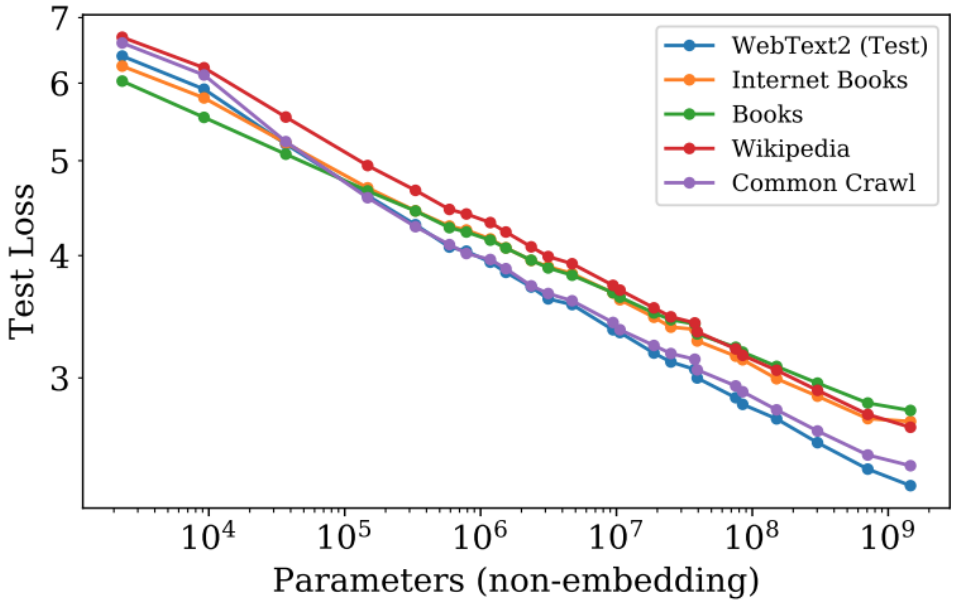
- The performance of a wide variety of models, ranging from small models with shape (nlayer, dmodel) = (2, 128) through billion-parameter models
- Here we have trained to near convergence on the full WebText2 dataset and observe no overfitting (except possibly for the very largest models).
- As shown in the figure earlier, we find a steady trend with N



$$L(N) \approx \left(\frac{N_c}{N}\right)^{\alpha_N}$$

Performance vs N

- Although the models were trained on the WebText2 dataset, their performance was measured on other datasets as well
- Their test loss on a variety of other datasets is also a power-law in N with nearly identical power



Performance depends strongly on scale, weakly on model shape

- Model shape parameters include
  - n_{layer} : Number of layers
  - d_{ff} : Dimension of the intermediate feed-forward layer
  - n_{heads} : Number of attention heads per layer
- While holding the model size N constant, different models were trained while varying a single hyperparameter

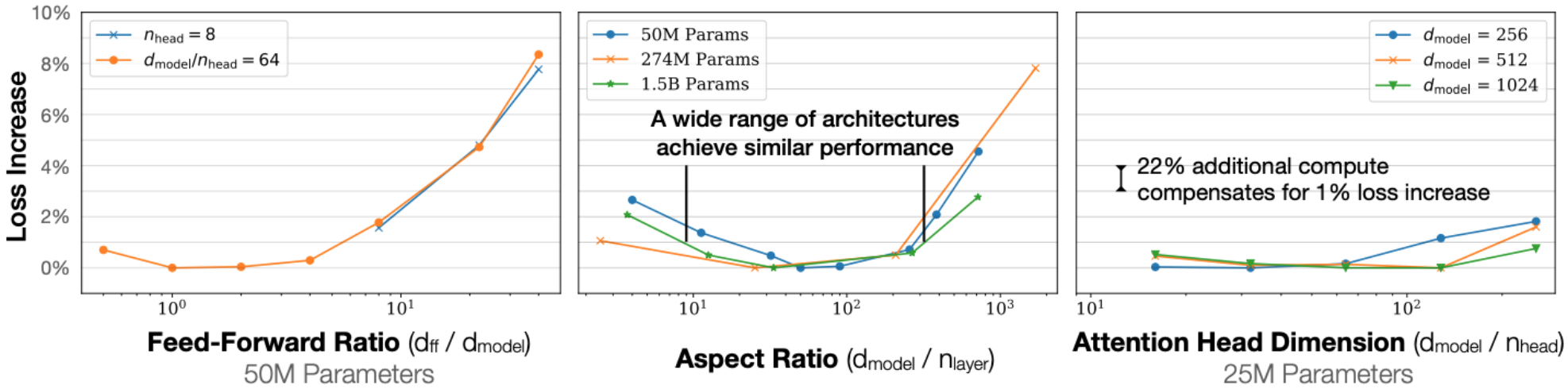
$$N = 12 \cdot n_{\text{layer}} \cdot d_{\text{model}}^2$$

$$\text{where } d_{\text{model}} = \frac{d_{\text{ff}}}{4} = d_{\text{attn}}$$

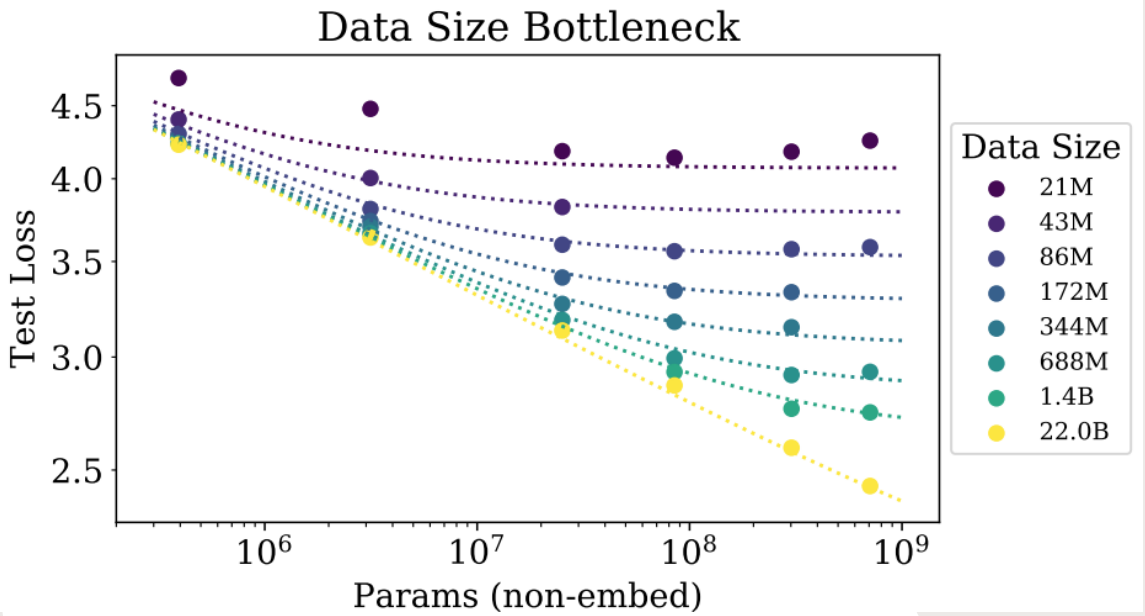
- The loss varies only a few percent over a wide range of shapes

Performance depends strongly on scale, weakly on model shape

- The loss varies only a few percent over a wide range of shapes

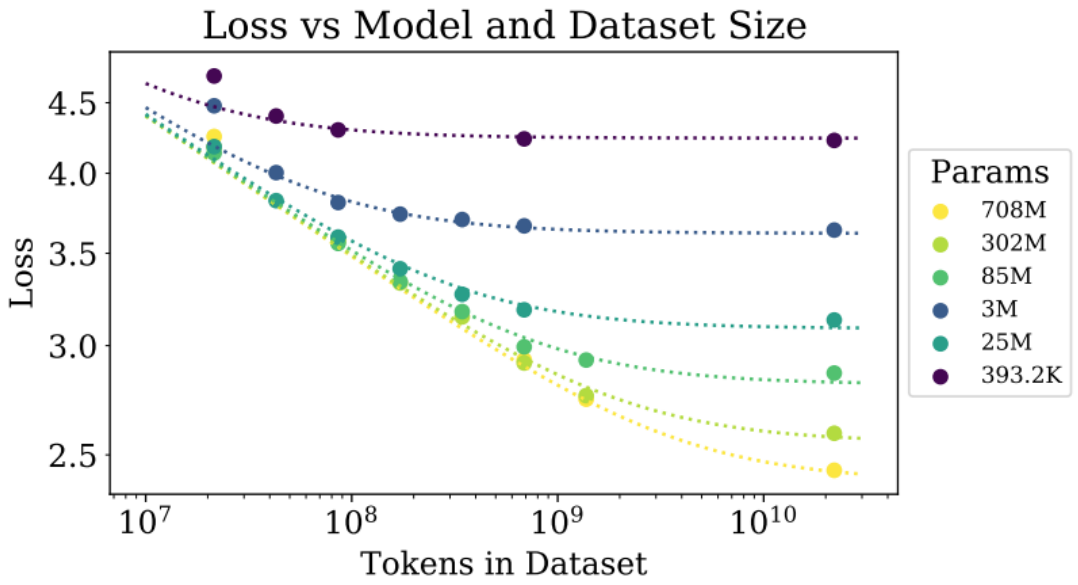


- n_{layer} : Number of layers
- d_{ff} : Dimension of the intermediate feed-forward layer
- n_{heads} : Number of attention heads per layer



Charting the Infinite Data Limit and Overfitting

- Exploring what happens to a model's performance if it were trained on an infinite amount of data
- The relationship between the performance of a model of size N trained on a dataset with D tokens is measured
  - while varying N and D simultaneously
- For large D, performance is a straight power law in N
- For a smaller fixed D, performance stops improving as N increases and the model begins to overfit
- Performance improves predictably as long as N and D are scaled up tandem
- Performance enters a pattern of diminishing returns if either N or D is held fixed while the other increases

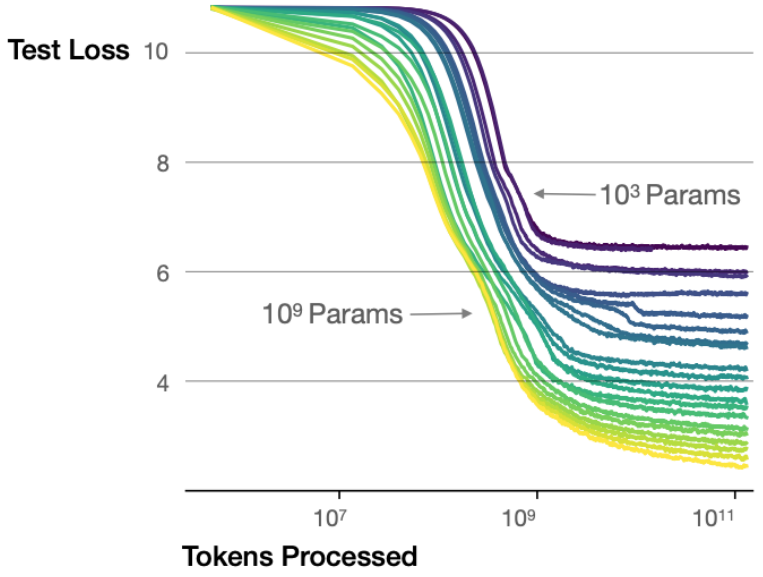


Charting the Infinite Data Limit and Overfitting

- Here, the reverse is also true: for a smaller fixed N, performance stops improving as D increases and the model begins to overfit
- From analyzing how test loss scales with N and D, the performance penalty depends predictably on the ratio N0.74/D
- Suggests that every time we increase the model size 8x, we only need to increase the data by roughly 5x to avoid a penalty

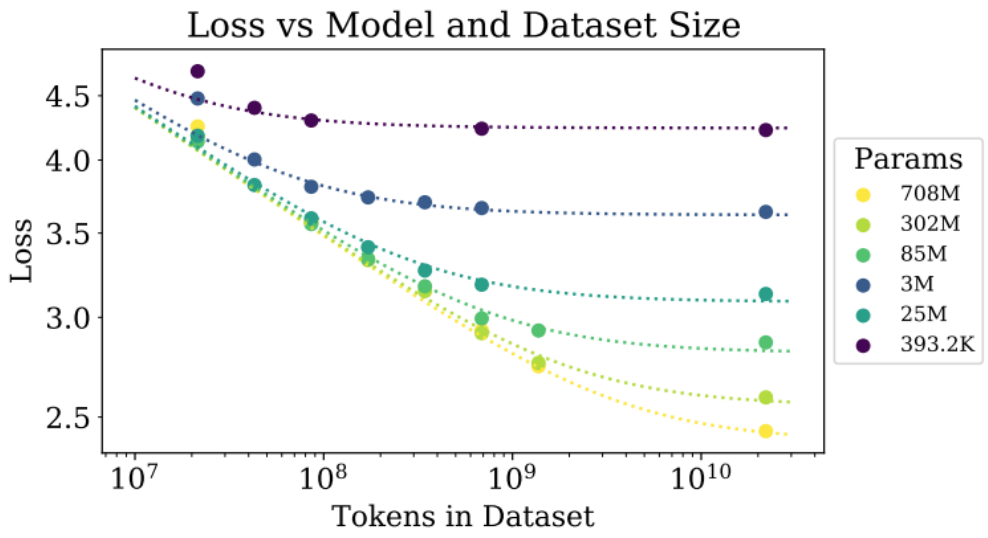
Sample Efficiency

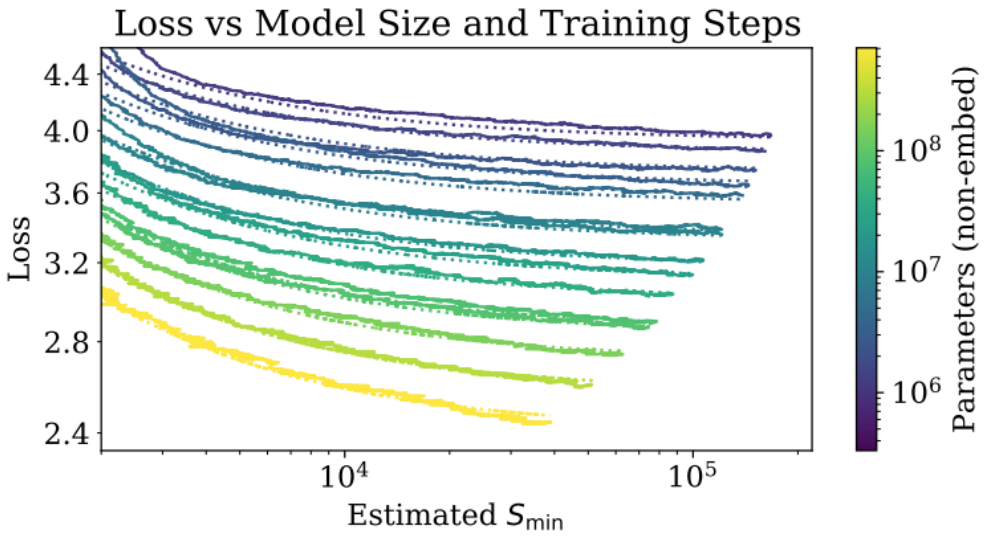
- Large models are more sample-efficient than small models, reaching the same level of performance with fewer steps



Sample Efficiency

- Large models reach the same level of performance with using fewer data points





Insights

Implications of Scaling Laws

- Predictable Scaling Behavior
  - Loss scales as a **power-law function** of model size, dataset size, and compute
  - Key takeaway: Larger models and datasets improve performance but yield diminishing returns—scaling further provides smaller improvements
- Balancing Model Size and Dataset Size
  - Performance Penalties: Penalties occur when model size or dataset size is under-scaled relative to the other
  - Scaling Relationship: For an **8x** increase in model size, dataset size must grow **5x** to maintain efficiency
- Larger Models Are Worthwhile
  - Consistent Gains: Larger models consistently outperform smaller ones when trained effectively
  - Scaling up to models with **billions or trillions of parameters** is worthwhile within reasonable compute and dataset limits

Roadmap for Future Models

- Scaling laws provide a roadmap for developing increasingly powerful language models like GPT and LLaMA.
- They explain why models like GPT-4 continue to improve with larger sizes and datasets.
- They provide actionable insights into how to design, train, and deploy neural language models efficiently and effectively, helping push the boundaries of AI capabilities.

Works Cited

- Kaplan, J., Mccandlish, S., Tom, Benjamin, Rewon, Scott, Alec, Jeffrey, & Dario. (2020). Scaling Laws for Neural Language Models. https://arxiv.org/pdf/2001.08361
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. https://arxiv.org/pdf/1706.03762