Scaling and Emergent Ability

CS6770 Natural Language Processing

Thomas Chia
Kate Chu
Shawn Thomas

Overview

As large language models improve dramatically with scale, their behavior changes in ways that challenge our understanding of how intelligence emerges from training and computation.

- Key Questions:
  - How should we scale models optimally?
  - What new abilities emerge as models scale?
  - Are these “emergent” behaviors real, or artifacts of how we measure performance?

Training Compute-Optimal Large Language Models

DeepMind, 2022

Hoffmann et al., arXiv:2203.15556

LLMs Are Undertrained

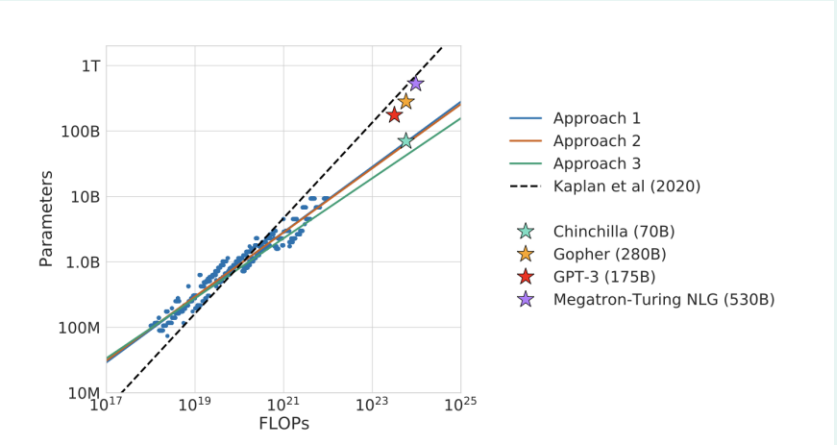
The industry focused on scaling model size but neglected training data:

Model	Parameters	Training Tokens
GPT-3	175B	300B
Gopher	280B	300B
Megatron-Turing NLG	530B	270B
Optimal for compute	**~70B**	**1.4T+**

Models were significantly larger than optimal but trained on insufficient data

Major Contributions

- Scaling Law Discovery: Model parameters and training data should scale equally with compute budget, contradicting prior assumptions
- Chinchilla Model: Demonstrated a 70B parameter model trained on 1.4T tokens outperforms much larger models (280B Gopher, 175B GPT-3)
- Empirical Validation: Trained over 400 models using three independent approaches to verify the scaling relationship



All three methodological approaches use in the paper converged on the same conclusion: Current LLMs (*at the time*) should be substantially smaller, trained much longer, and trained with a larger dataset (more tokens).

Impact & Significance

- Changed Industry Thinking: Shifted focus from "bigger is always better" to compute-optimal training strategies
- Revealed Undertraining: Showed that state-of-the-art models like GPT-3 and Gopher were undertrained by 4-5x in terms of tokens
- Data as Critical Resource: Elevated data quality and quantity to equal importance as model architecture
- Practical Deployment: Smaller models enable faster inference, lower serving costs, and easier deployment at scale

Influenced subsequent models: LLaMA, Mistral, and others adopted similar data scaling strategies

The Key Discovery: Equal Scaling

For a given compute budget C, both parameters and training tokens should scale with the square root:

$$N_{opt} \propto C^{0.50} \qquad D_{opt} \propto C^{0.50}$$

Contrast with Kaplan et al. (2020):

$$N_{opt} \propto C^{0.73} \qquad D_{opt} \propto C^{0.27}$$

Previous work suggested model size should grow much faster than training data

Equal exponents (0.5, 0.5) vs unequal exponents (0.73, 0.27) fundamentally changed resource allocation

Methodology: Three Independent Approaches

- **Approach 1: Fixed Model Sizes**
  Trained models of varying sizes with different token counts, analyzed loss curves to find optimal data per model size
- **Approach 2: IsoFLOP Profiles**
  For fixed compute budgets, varied model size and training duration to find the optimal allocation
- **Approach 3: Parametric Loss Fitting**
  Fitted a loss function to predict performance, then derived optimal scaling analytically

All three approaches independently converged to the same scaling law

Trained over 400 language models ranging from 70M to 16B parameters

Approach 1: Fixed model sizes

Found the "scaling exponents" law.

- Trained models from 70M to 10B parameters with 4 different training durations each (varying by 16× in tokens)
- Each training run produces a loss curve: loss vs function of FLOPs consumed. Then, for each FLOP count, determined which model size achieved the lowest loss
- Used cosine learning rate schedule matched to the number of training steps (critical finding: overshooting by >25% degrades performance)
- Smoothed and interpolated training curves to create continuous loss mappings, allowing us to view the exact loss value at a certain FLOP.
- Extracted the "envelope" of minimal loss per FLOP across all runs, allowing us to find the best loss for each FLOP level

Approach 1: Fixed model sizes

Found the "scaling exponents" law: 0.5 for model size, and token count.

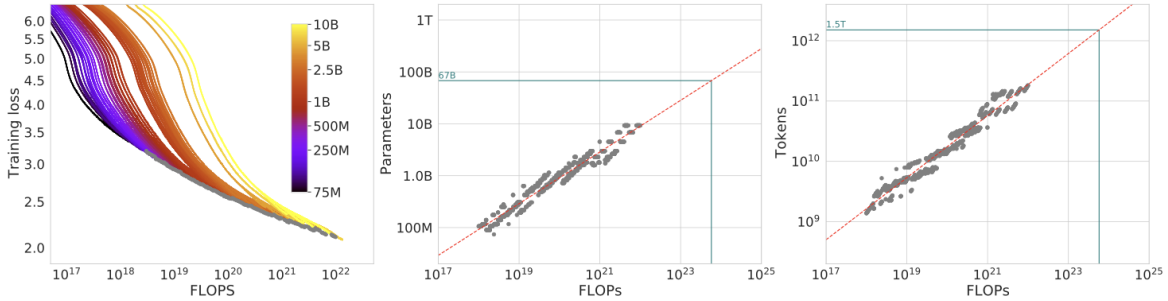


Figure 2 | **Training curve envelope.** On the **left** we show all of our different runs. We launched a range of model sizes going from 70M to 10B, each for four different cosine cycle lengths. From these curves, we extracted the envelope of minimal loss per FLOP, and we used these points to estimate the optimal model size (**center**) for a given compute budget and the optimal number of training tokens (**right**). In green, we show projections of optimal model size and training token count based on the number of FLOPs used to train *Gopher* (5.76×10^{23}).

Approach 2: Fixed model sizes

Reinforced the "scaling exponents" law with diff. methodology.

- Fixed 9 different FLOP budgets (6×10^18 to 3×10^21) and varied model size at each budget (they represent a horizontal slide thru. compute space.)
- Looked at final training loss for each configuration (rather than entire training curves)
- Model sizes ranged up to 16B parameters (larger than Approach 1)

- Set cosine cycle length to match the number of training tokens for optimal training
- Fit parabolas to each IsoFLOP curve to find the loss-minimizing model size

Approach 2: Fixed model sizes

Reinforced the "scaling exponents" law with diff. methodology.

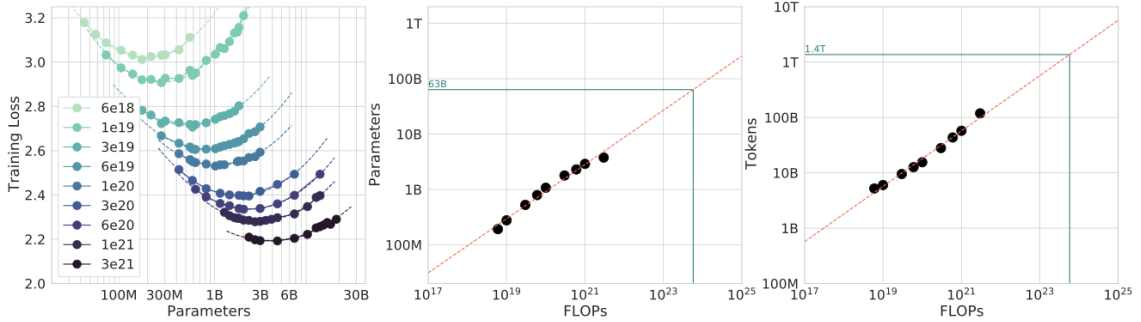


Figure 3 | **IsoFLOP curves.** For various model sizes, we choose the number of training tokens such that the final FLOPs is a constant. The cosine cycle length is set to match the target FLOP count. We find a clear valley in loss, meaning that for a given FLOP budget there is an optimal model to train (**left**). Using the location of these valleys, we project optimal model size and number of tokens for larger models (**center** and **right**). In green, we show the estimated number of parameters and tokens for an *optimal* model trained with the compute budget of *Gopher*.

Approach 3: Loss Fitting

Theoretical models can also be used to derive scaling laws.

- **Model loss theoretically as three components (next page).**
- **Uses all 400+ training runs simultaneously from both Approach 1 and Approach 2, fitting 5 parameters (E, A, B, α, β) using L-BFGS optimization with Huber loss ($\delta=10^{-3}$) to be robust against outliers, particularly noisy small-scale runs**
- **Derives optimal scaling analytically** by minimizing the fitted loss function under the compute constraint FLOPs ≈ 6ND, yielding closed-form power laws where the exponents come from the ratio of fitted parameters: $a = \beta/(\alpha+\beta)$ and $b = \alpha/(\alpha+\beta)$, ensuring $a + b = 1$.
- **Finds scaling exponents a = 0.46 and b = 0.54 with extremely tight confidence intervals (0.454-0.455 and 0.542-0.543), predicting slightly smaller optimal models than the other approaches due to observed negative curvature at high compute budgets that the Huber loss automatically captures.**
- **Uniquely enables counterfactual predictions for any (N,D) combination beyond just the optimal frontier**

Parametric Loss Function

The loss can be decomposed into three components:

$$L(N, D) = E + A/N^{\alpha} + B/D^{\beta}$$

E: Bayes risk
Irreducible entropy of natural language

A/N^{α}: Approximation error
Limited by model capacity

B/D^{β}: Optimization error
Limited by training data

Fitted values: $\alpha \approx 0.46$, $\beta \approx 0.54$, yielding equal scaling in optimal allocation

Minimizing L(N,D) subject to compute constraint $C \propto N \times D$ yields $N \propto C^{0.5}$, $D \propto C^{0.5}$

Approach 3: Loss Fitting

Reinforced the "scaling exponents" law with diff. methodology.

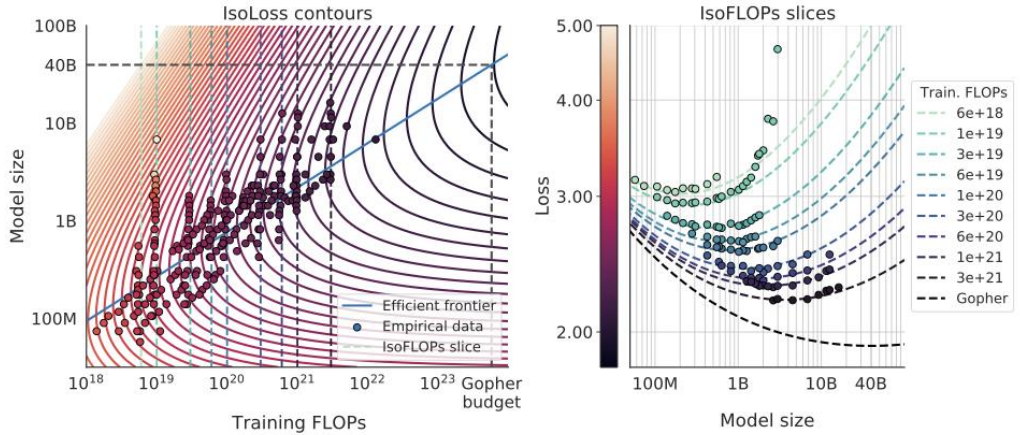
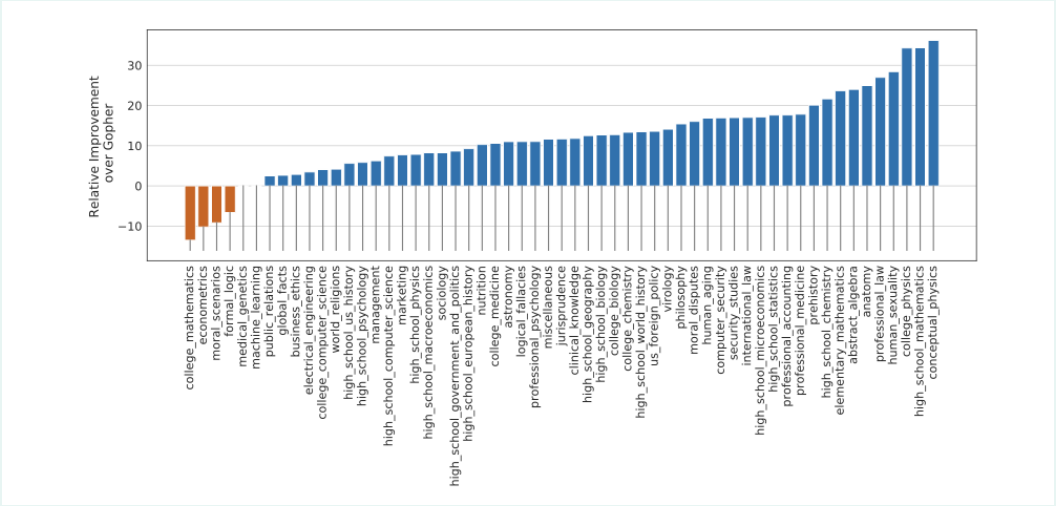


Figure 4 | **Parametric fit.** We fit a parametric modelling of the loss $\hat{L}(N, D)$ and display contour (**left**) and isoFLOP slices (**right**). For each isoFLOP slice, we include a corresponding dashed line in the left plot. In the left plot, we show the efficient frontier in blue, which is a line in log-log space. Specifically, the curve goes through each iso-loss contour at the point with the fewest FLOPs. We project the optimal model size given the *Gopher* FLOP budget to be 40B parameters.

Chinchilla Outperforms Larger Models



4x smaller than Gopher, but 7.5 percentage points better on MMLU

MMLU: Massive Multitask Language Understanding (57 academic tasks). Same compute budget used.

Practical Implications

- Smaller Models, More Data: Instead of training 175B+ parameter models on 300B tokens, train 70B models on 1.4T+ tokens for better performance
- Reduced Inference Costs: Smaller models require less memory and compute during deployment, enabling faster and cheaper serving at scale
- Better Downstream Performance: Compute-optimal models consistently outperform undertrained larger models across diverse benchmarks
- Data Collection Critical: High-quality training data becomes equally important as architectural innovations, shifting resource allocation priorities

Trade compute from parameters to tokens: More efficient use of computational budgets

Limitations & Future Directions

Current Limitations:

- Limited Large-Scale Validation: Only two models trained at the 70B+ scale due to computational cost constraints
- Power Law Assumptions: Scaling laws may not hold indefinitely at extreme scales or near data saturation
- Single Epoch Training: All models trained for only one epoch; multi-epoch effects unexplored

Future Research Directions:

- Investigate curvature in scaling relationships at larger compute budgets
- Study optimal data repetition and multi-epoch training strategies
- Explore data quality effects beyond quantity alone

Scaling laws provide guidance but require empirical validation at each new scale

Emergent Abilities of Large Language Models

Wei et al., 2022

arXiv:2206.07682

CS 6770: Scaling and Emergent Ability

Overview

- Define "Emergent Abilities":
  - Abilities that are not present in small-scale models but are present in large-scale models
  - Cannot be predicted by extrapolating the performance of small models
  - Phase transition: dramatic change of model behavior
- Investigate emergent abilities with examples and analyze how these abilities appears in scaling curves
  - Draw evidence from few-shot prompting tasks and augmented prompting strategies
  - Plot and compare how performance changes with different model sizes

Few-shot Prompted Tasks

- Consider the ability is emergent when model shows random performance until certain scale
- Compared the performance on 8 tasks across 5 models:
  - BIG-Bench:
    - Modular arithmetic: 3-digit addition/subtraction and 2-digit multiplication
    - IPA transliteration: convert IPA symbols to normal word spelling; e.g., /kæt/ -> cat
    - Word unscrambling: recover word from scrambled letters; e.g., elppa -> apple
    - Persian QA: answer questions in Persian.
  - Truthful QA: answer questions truthfully, avoiding common misconceptions
  - Grounded Conceptual Mappings: Map abstract concepts to positions in a textual grid world
  - Massive Multi-task Language Understanding: 57 tasks across math, history, law, etc.
  - Word in Context: determine if a word has the same meaning in two sentences

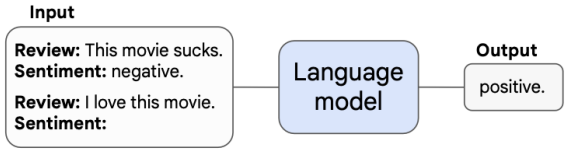


Figure 1: Example of an input and output for few-shot prompting.

https://arxiv.org/abs/2206.07682

Few-shot Prompted Tasks

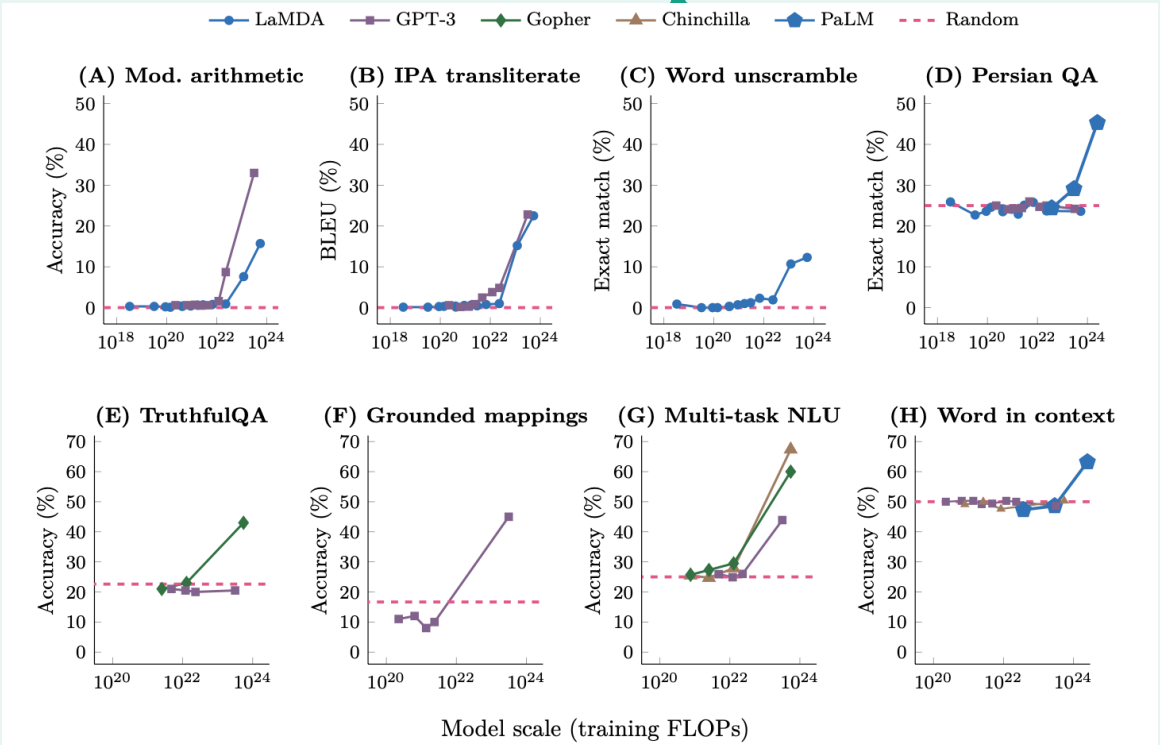


Figure 2: Eight examples of emergence in the few-shot prompting setting. Each point is a separate model. The ability to perform a task via few-shot prompting is emergent when a language model achieves random performance until a certain scale, after which performance significantly increases to well-above random. Note that models that used more training compute also typically have more parameters—hence, we show an analogous figure with number of model parameters instead of training FLOPs as the x-axis in Figure 11.

Across these tasks, performance stays near random for small/medium models and then jumps sharply at larger training compute, suggesting emergent abilities.

https://arxiv.org/abs/2206.07682

Augmented Prompting Strategies

- Consider a technique as emergent ability if the technique contributes positively only when the model is large enough
- Strategies:
  - Chain-of-Thought (CoT) Prompting: model is prompted to generate step-by-step reasoning before answering
  - Instruction Tuning: fine-tuning model on many instruction–response examples
  - Program execution ("scratchpad"): model is trained to output intermediate steps
  - Calibration: model first gives the answer, and the estimate the probability that the answer is correct

Augmented Prompting Strategies

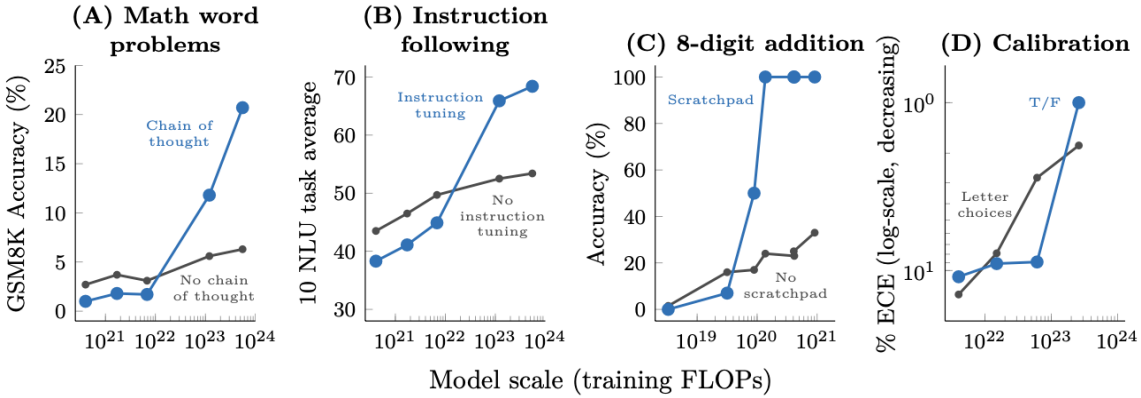


Figure 3: Specialized prompting or finetuning methods can be emergent in that they do not have a positive effect until a certain model scale. A: Wei et al. (2022b). B: Wei et al. (2022a). C: Nye et al. (2021). D: Kadavath et al. (2022). An analogous figure with number of parameters on the x-axis instead of training FLOPs is given in Figure 12. The model shown in A-C is LaMDA (Thoppilan et al., 2022), and the model shown in D is from Anthropic.

- Blue line (with specialized methods) only exceeds the grey line (without method) after models reaches a certain scale
  - Suggesting these techniques might be emergent abilities

https://arxiv.org/abs/2206.07682

Discussion & Implication

- Potential explanations of emergence
  - Some emergent abilities might require model to reach a threshold scale
  - More parameters and trainings enables better memorizations of knowledges
  - Emergence can partly depend on how we measure emergence:
    - Exact-match/accuracy metrics might hide gradual improvement
    - Small probability gains might not show up until reaches threshold
    - Cross-entropy loss improves for small models even when accuracy stays random
    - However, emergence still appear in classification tasks, so metrics alone could not explain sudden performance improvements

Discussion & Implication

- Other factors (e.g., quality of training data, model architecture, training objective) can also enable emergent abilities beyond scaling
  - e.g., PaLM 62B outperformed larger GPT-3 175B/LaMDA 137B on 14 BIG-Bench tasks
- Emergent abilities can be made possibles for smaller model overtime
  - e.g., instruction-based finetuning were once limited to models of 68B or bigger
- Emergent ability should be considered as a function of many correlated variables (e.g., perplexity)
  - New model design might shift the correlation between scale and ability

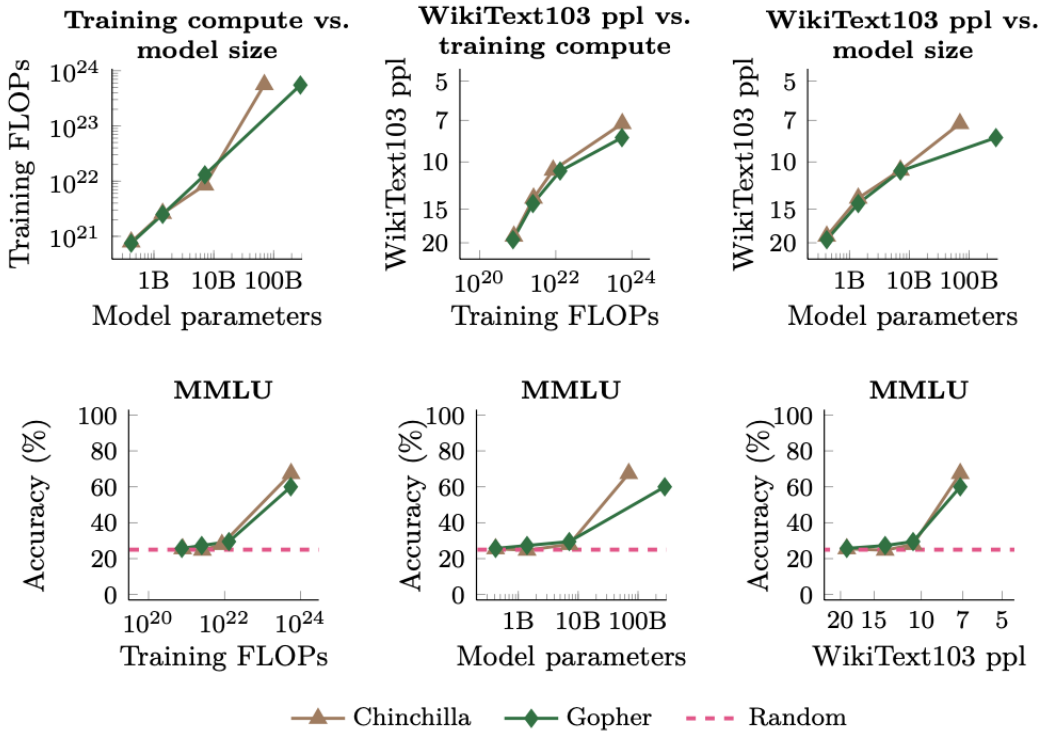


Figure 4: Top row: the relationships between training FLOPs, model parameters, and perplexity (ppl) on WikiText103 (Merity et al., 2016) for Chinchilla and Gopher. Bottom row: Overall performance on the massively multi-task language understanding benchmark (MMLU; Hendrycks et al., 2021a) as a function of training FLOPs, model parameters, and WikiText103 perplexity.

Discussion & Implication

- Safety concerns might emerge along with abilities
  - Increase risk for bias, toxicity, and misinformation with larger model scales
  - New approaches should be proposed to resolve emergent risks
- Scaling has produced an emergent sociocultural shift:
  - The field moved from building many small specialized models to training large, general-purpose model
  - Led to applications of language models in other scientific fields

Limitation & Future Direction

- **Limitations:**
  - Focused only on large dense transformer models and might not be generalizable
  - Model scale is confounded with other factors and could not be isolated as the only cause of emergence
- **Future works should:**
  - Continue model scaling and data scaling, as well as improving model architecture, training procedures, prompting techniques to enable emergent abilities

Future Direction & Contribution

- Investigate why some abilities to perform frontier tasks have not emerged yet and how to enable these abilities
- Investigate how and why emergent abilities occur in language models in new approaches (e.g., with different performance metrics)

- **Contribution:**
  - Formalized the concept of emergent abilities in LLMs
  - Provided empirical evidence across tasks and prompting techniques
  - Offered valuable discussion on possible explanations on emergent abilities and directions for future works

Are Emergent Abilities of Large Language Models a Mirage?

Schaeffer et al., 2023

arXiv:2304.15004

CS 6770: Scaling and Emergent Ability

Background

- Emergent properties: Abilities not present in smaller models but present in larger models, appearing sharp and unpredictable. (Wei et al., 2022)
- Characterized by: 1. Sharpness of transition of performance
  2. Unpredictability due to seemingly unknown causes.
- LLMs: eg. GPT 3, PaLM, LaMDA. Not present in smaller scale models.
- However, these emergent abilities seem to appear only under non-linear/discontinuous metrics.
- Potentially induced by researcher's choice of metric.

Are Emergent Abilities a Mirage?

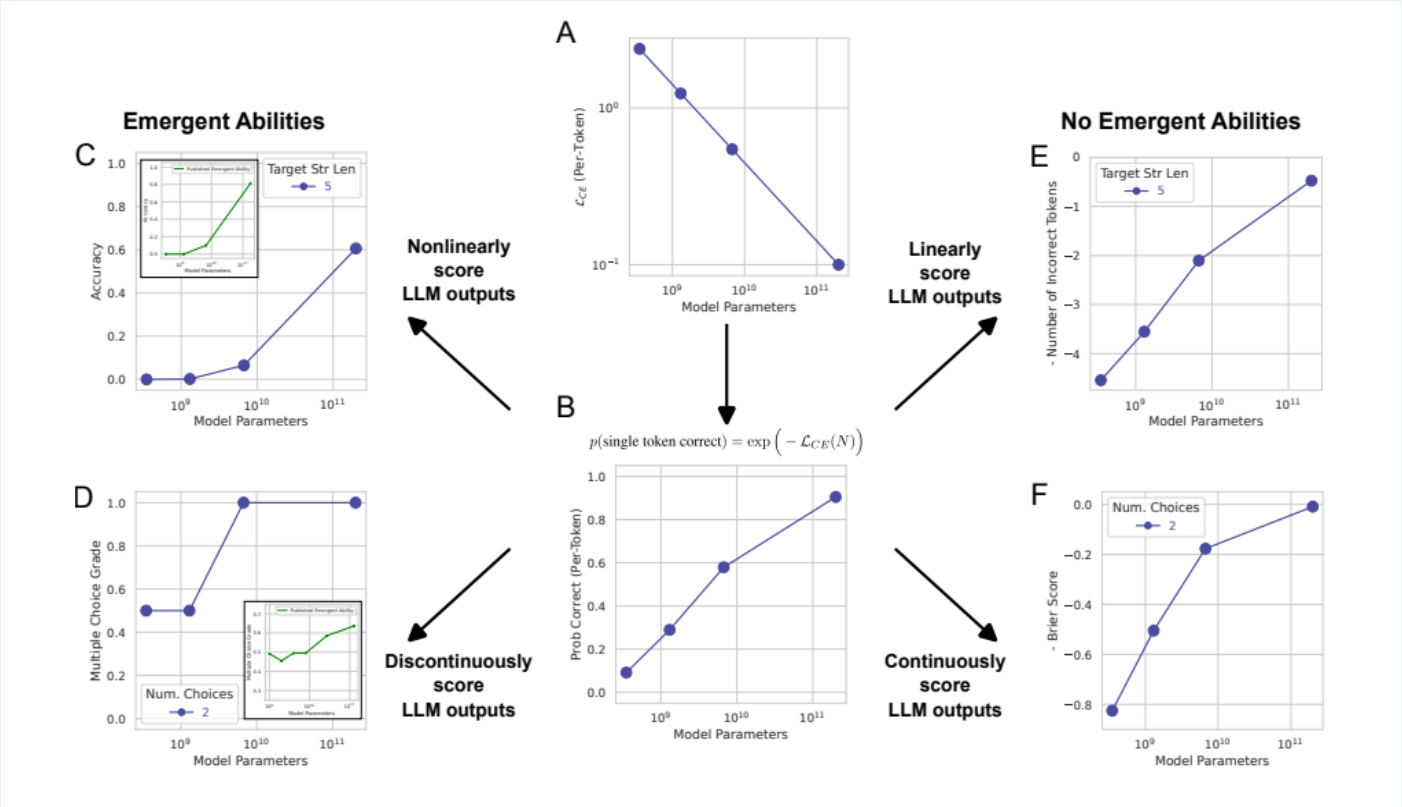
Schaeffer, Miranda, Koyejo (Stanford) - arXiv 2023

Question: Do sharp and unpredictable jumps in task related performance reflect true capability changes?

Thesis: Many “emergent abilities” are a **mirage** caused by the *researcher’s evaluation choices*

- **Discontinuous metrics** can turn smooth improvements into sharp jumps
- Limited evaluation data can make smaller models look like they have “zero ability”

Are Emergent Abilities a Mirage?



Left: Non-Linear Metrics **Middle: Per-Token Metrics** **Right: Continuous Metrics**

Major Contributions

- Proposes an **alternative explanation** for emergence: metric choice + measurement resolution
- Gives a **simple mathematical model** showing how smooth per-token improvement can produce “emergent-looking” curves.
  - **Empirical test** on InstructGPT/GPT-3 arithmetic tasks: emergence disappears under different metrics
  - **Meta-analysis** of BIG-Bench: emergence appears mostly under a small set of harsh metrics
  - **Constructive demo**: induce “emergence” in vision models by designing sharp metrics

Key Idea: Task ≠ Metric

Same model outputs can look “emergent” or “smooth” depending on the metric

Discontinuous / harsh metrics (create sharp transitions):

- Multiple Choice Grade: 1 if top choice correct, else 0
- Exact String Match: 1 if full output matches target, else 0

Continuous / graded metrics (reveal smooth scaling):

- Token Edit Distance (counts token mistakes)
- Brier Score (uses predicted probabilities, not just argmax)
- Cross-entropy loss (per-token, smooth scaling law behavior)

GPT-3 / InstructGPT Arithmetic

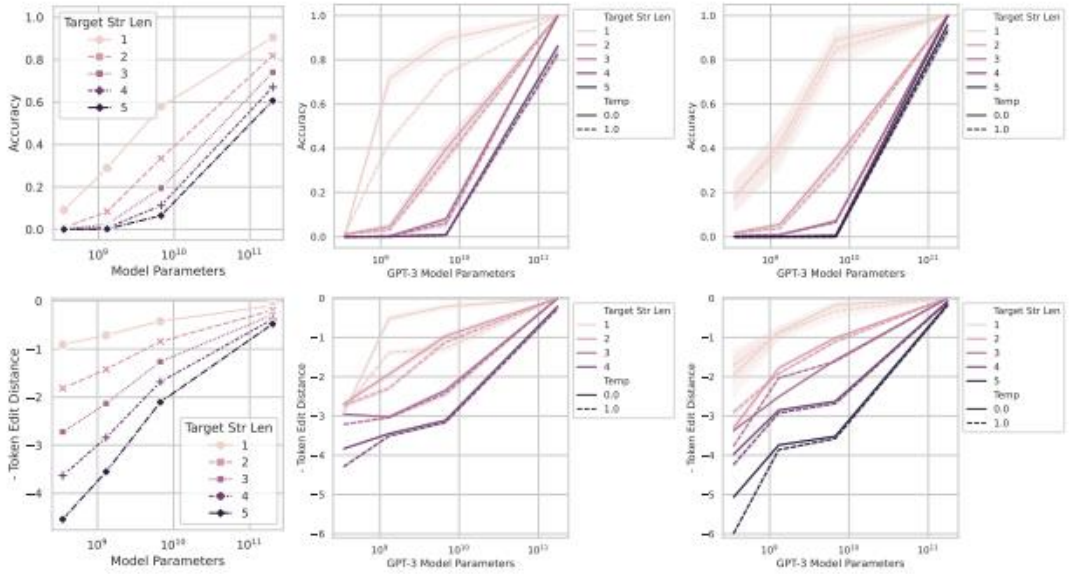


Figure 3: **Claimed emergent abilities evaporate upon changing the metric.** Left to Right: Mathematical Model, 2-Integer 2-Digit Multiplication Task, 2-Integer 4-Digit Addition Task. Top: When performance is measured by a nonlinear metric (e.g., Accuracy), the InstructGPT/GPT-3 [3, 24] family's performance appears sharp and unpredictable on longer target lengths. Bottom: When performance is instead measured by a linear metric (e.g., Token Edit Distance), the family exhibits smooth, predictable performance improvements.

Setup: Arithmetic tasks (addition/multiplication)

Metric swap, outputs held fixed

- With **Accuracy**, longer targets look "flat then jump" (emergent)
- With **Token Edit Distance**, performance improves **smoothly** with scale

GPT-3 / InstructGPT Arithmetic

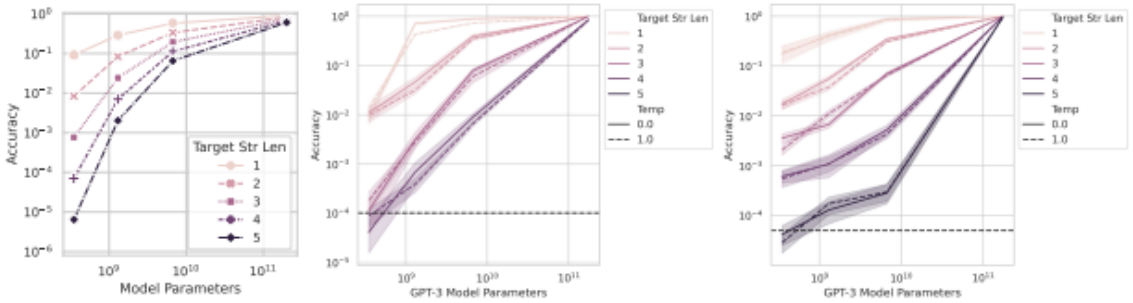


Figure 4: **Claimed emergent abilities evaporate upon using better statistics.** Left to Right: Mathematical Model, 2-Integer 2-Digit Multiplication Task, 2-Integer 4-Digit Addition Task. Based on the predictable effect Accuracy has on performance, measuring performance requires high resolution. Generating additional test data increases the resolution and reveals that even on Accuracy, the InstructGPT/GPT-3 family's [3, 24] performance is above chance and improves in a smooth, continuous, predictable manner that qualitatively matches the mathematical model.

"Better Statistics"

- Increasing test set size reveals smaller models are **above chance**, not truly "zero"
- The "jump" becomes a smoother curve when measured with enough resolution

BIG-Bench Meta Analysis

Motivation:

- Most model families with claimed emergence are not publicly queryable
- So the authors analyze published results on BIG-Bench

Predictions from the hypothesis:

- Prediction 1 (Population-level):
  Emergent abilities should appear with specific metrics,
  not consistently across task–model family pairs

- Prediction 2 (Triplet-level):
  If emergence appears for a task–model family,
  changing to a continuous/linear metric should remove it

Detecting Emergence

How do they detect emergence?

- Use an emergence score to quantify sharp performance jumps
- Applied across **task–metric–model family triplets** in BIG-Bench

$$\text{Emergence Score}\left(\left\{(x_n, y_n)\right\}_{n=1}^{N}\right) \overset{\text{def}}{=} \frac{\text{sign}(\arg\max_i y_i - \arg\min_i y_i)(\max_i y_i - \min_i y_i)}{\sqrt{\text{Median}(\{(y_i - y_{i-1})^2\}_i)}} \quad (1)$$

Intuition:

- Flags cases where performance shows a large jump relative to nearby model scales
- Normalizes the jump by typical smooth changes to avoid false positives

Prediction 1

- Emergent abilities appear under a few metrics
- Most BIG-Bench metrics show no emergence across task-model pairs
- 92% of "emergent abilities" come from 2 metrics:
  - Multiple Choice Grade
  - Exact String Match

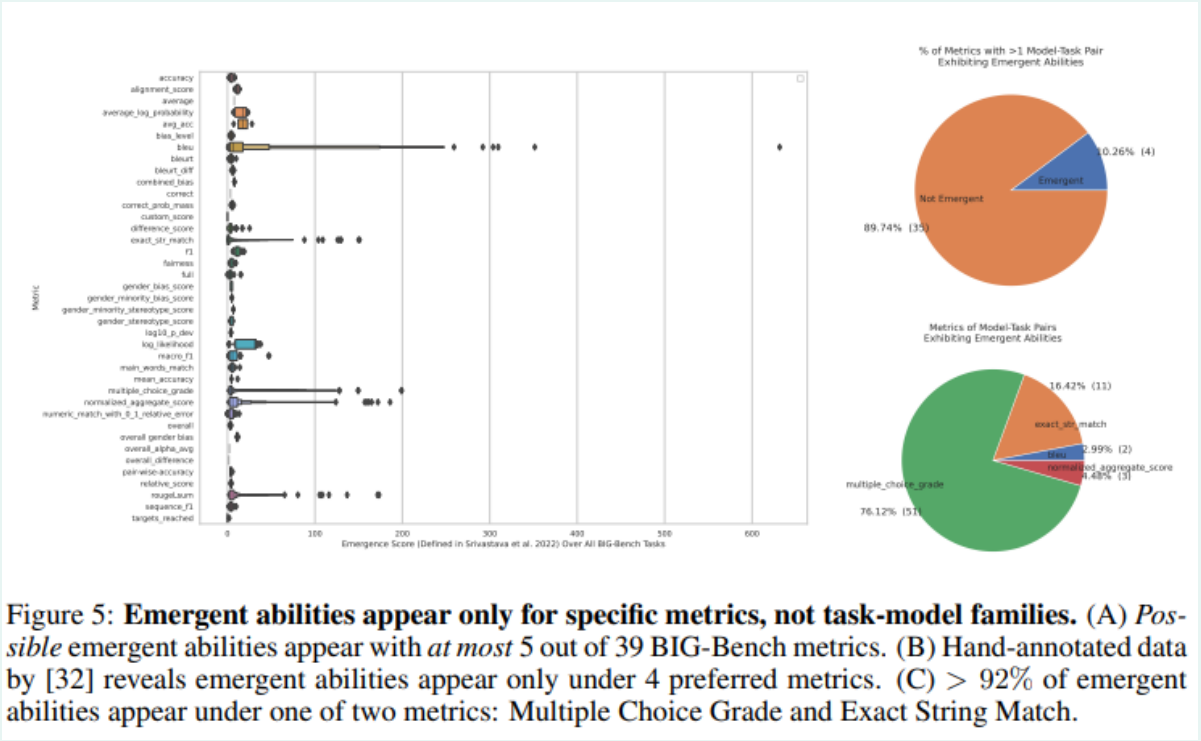


Figure 5: **Emergent abilities appear only for specific metrics, not task-model families.** (A) *Possible* emergent abilities appear with *at most* 5 out of 39 BIG-Bench metrics. (B) Hand-annotated data by [32] reveals emergent abilities appear only under 4 preferred metrics. (C) $>$ 92% of emergent abilities appear under one of two metrics: Multiple Choice Grade and Exact String Match.

Prediction 2

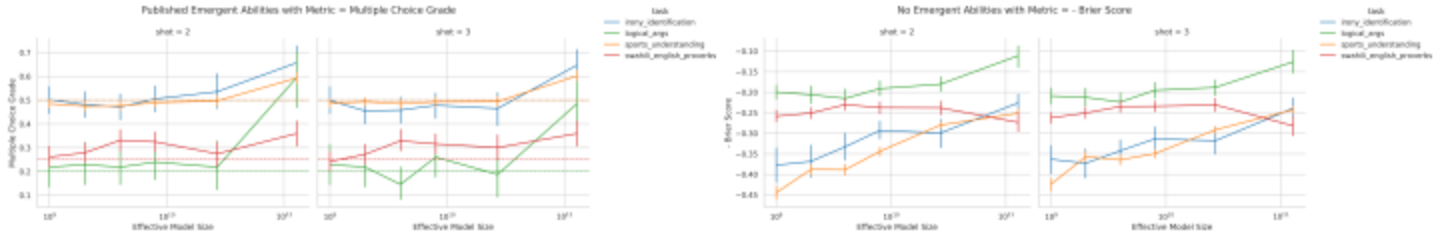


Figure 6: **Changing the metric when evaluating task-model family pairs causes emergent abilities to disappear.** Left: The LaMDA model family displays emergent abilities when measured under the discontinuous Multiple Choice Grade. Right: The LaMDA model family's emergent abilities disappear when measured under a continuous BIG-Bench metric: Brier Score.

Example: LaMDA results available via BIG-Bench

- Under Multiple Choice Grade, some tasks look emergent.
- Re-score same tasks with Brier score (continuous)
- Emergent trends disappear.

Inducing Emergence

Intentionally create emergent-looking curves for vision tasks by swapping metrics.

Why this matters:

- Emergence-like curves are not unique to LLMs
- Sharp metrics can create the illusion in many architectures/domains

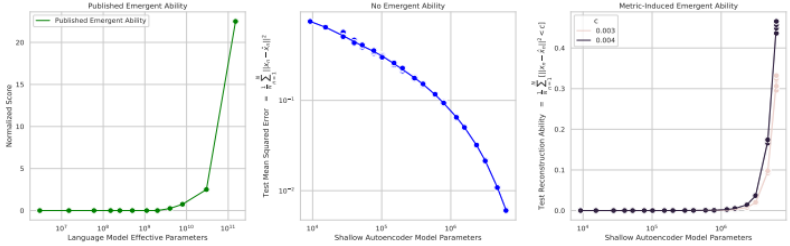


Figure 7: **Induced emergent reconstruction ability in shallow nonlinear autoencoders.** (A) A published emergent ability at the BIG-Bench Periodic Elements task [28]. (B) Shallow nonlinear autoencoders trained on CIFAR100 [19] display smoothly decreasing mean squared reconstruction error. (C) Using a newly defined Reconstruction$_c$ metric (Eqn. 2) induces an unpredictable change.

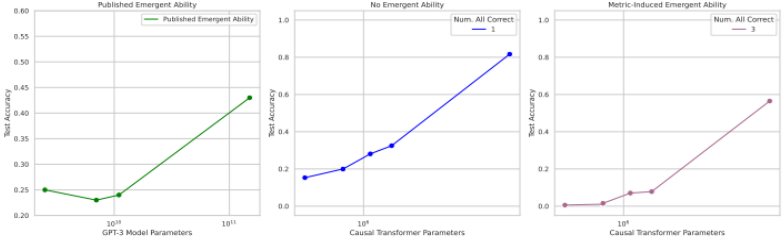


Figure 8: **Induced emergent classification ability in autoregressive Transformers.** (A) A published emergent ability on the MMLU benchmark [8]. (B) Autoregressive transformers trained to classify Omniglot images display increasing accuracy with increasing scale. (C) When accuracy is redefined as classifying *all* images correctly, a seemingly emergent ability appears.

Implications & Significance

- Suggests many reported “emergent abilities” may be **evaluation artifacts**, not sudden capability changes
- “Sharp jumps” can be **predictable under better metrics**
- Shows that **task ≠ metric** — evaluation design can change scientific conclusions
- Benchmarking needs careful **task + metric design** and enough samples for harsh metrics
- Authors don’t claim LLMs can’t show emergence, just that many claims may be **mirages**

Limitations & Future Direction

Current Limitations:

- Does **not rule out genuine emergent abilities** — only challenges prior empirical claims
- Relies on **simplified assumptions** (e.g., approximate token independence, fixed outputs)
- Meta-analysis depends on **existing benchmark definitions and annotations** of emergence

Future Directions:

- Develop **metric-robust definitions of emergence** independent of evaluation choice
- Design benchmarks with **proper statistical controls** (e.g., multiple comparisons)
- Increase **model output transparency** to enable independent replication and analysis

Overall Conclusions

- **1. Scaling Requires Balance, Not Just Size** Training compute should be allocated equally between model parameters and training data. The industry's focus on ever larger models was misguided; a 70B parameter model trained on 1.4T tokens outperforms undertrained 280B models. Data quality and quantity matter as much as architecture.

- **2. Emergent Abilities Exist, But Context Matters** Large language models do exhibit qualitatively new capabilities at scale, from multi-step reasoning to instruction following. However, these abilities depend on multiple correlated factors beyond just parameter count, including training data quality, model architecture, and fine-tuning strategies. Emergence is not magic; it is a function of measurable variables.

- **3. Measurement Shapes Scientific Understanding** The appearance of sharp, unpredictable capability jumps often reflects evaluation choices rather than true phase transitions. Discontinuous metrics like exact match accuracy can create illusions of emergence where continuous metrics reveal smooth scaling. Rigorous benchmarking requires careful metric selection, sufficient evaluation samples, and awareness that how we measure fundamentally shapes what we conclude about model behavior.

Sources:

https://arxiv.org/abs/2203.15556
https://arxiv.org/abs/2206.07682
https://arxiv.org/abs/2304.15004