Scaling Laws; In-Context Learning; Emergent Abilities

What happens when a language model is made bigger?

CSCI 567 — Machine Learning • Lecture 10 • Nov 13, 2026

Today's question: what does growing parameters, data, and compute buy?

Lecture 08, Part II built the Transformer language model, trained on next-token prediction. Today adds no new architecture: it asks what happens to that model as three measurable quantities grow.

N	D	C
number of parameters	training tokens	compute (FLOPs)

The three are tied together: training N parameters on D tokens costs about $C \approx 6ND$ floating-point operations, so a budget C fixes the product ND and leaves one real choice, how to split it.

Arc. A measured power law in N, D, C $\rightarrow$ the compute-optimal split of a budget $\rightarrow$ learning a task with no weight update $\rightarrow$ are abilities that appear suddenly with scale real, or produced by the metric?

Outline

1. **Scaling laws** (Kaplan et al., 2020): test loss is a power law in N, D, and C; how to read the exponents; the joint law in N and D.
2. **Compute-optimal training** (Hoffmann et al., 2022): the cost model $C \approx 6ND$, a fitted loss $\mathcal{R}(N, D)$, the optimal split in closed form, and Chinchilla.
3. **Scaling laws and double descent**: two places where the classical picture of Lectures 03 and 04 stops predicting what is measured.
4. **In-context learning** (Brown et al., 2020): a task specified in the prompt, zero-shot to few-shot, and how it differs from fine-tuning.
5. **Emergent abilities** (Wei et al., 2022; Schaeffer et al., 2023): sharp jumps with scale, and the argument that an all-or-nothing metric produces them.

1. Scaling Laws

Loss as a power law in size, data, and compute

Next-token prediction is one loss over many skills

Skill needed	Context	Correct continuation
Grammar	"The keys to the cabinet ___ on the table"	are
World knowledge	"The capital of Japan is ___"	Tokyo
Arithmetic	"$14 + 27 =$___"	41
Translation	"Bonjour, comment allez-___"	vous

No row is labeled as a grammar or an arithmetic example; each is ordinary text. The pretraining loss, the average next-token cross-entropy $-\frac{1}{T}\sum_t \log p_\theta(x_t \mid x_{<t})$, averages over all of these skills at once.

Two consequences. (1) One scalar, the test loss in nats per token, tracks many skills, so loss against scale is a meaningful curve. (2) Any input-output task can be written as text to continue: the basis of in-context learning (Section 4).

Kaplan et al. (2020): three power laws, one per factor

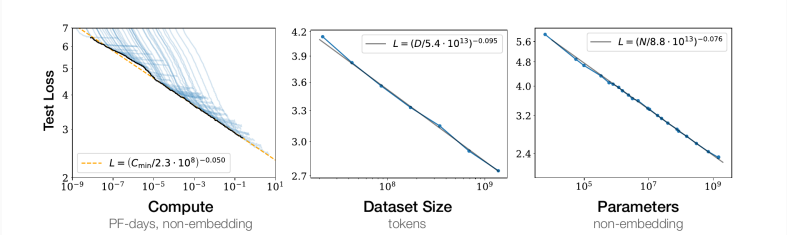
Train Transformer language models over a wide range of sizes, data, and compute, and measure the test loss $\mathcal{R}$ in nats per token (Lecture 01's population risk, written as a function of scale).

$$\mathcal{R}(N) = \left(\frac{N_c}{N}\right)^{\alpha_N}, \qquad \mathcal{R}(D) = \left(\frac{D_c}{D}\right)^{\alpha_D}, \qquad \mathcal{R}(C_{\min}) = \left(\frac{C_c^{\min}}{C_{\min}}\right)^{\alpha_C^{\min}}$$

Factor	Measured when	Exponent	Constant
N: non-embedding parameters	trained to convergence, data not limiting	$\alpha_N \approx 0.076$	$N_c \approx 8.8 \times 10^{13}$
D: training tokens	large model, early stopping	$\alpha_D \approx 0.095$	$D_c \approx 5.4 \times 10^{13}$
$C_{\min}$: compute in PF-days	best model size for each budget	$\alpha_C^{\min} \approx 0.050$	$C_c^{\min} \approx 3.1 \times 10^{8}$

The fits span eight orders of magnitude in $C_{\min}$, six in N, and over two in D (one PF-day is 8.64×10^{19} operations). The constants depend on the tokenizer and have no meaning of their own; the exponents are the result.

The measured curves: straight lines on log-log axes

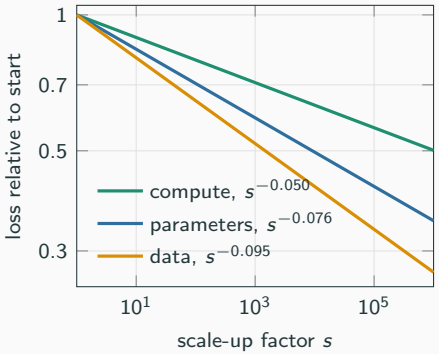


Kaplan et al. (2020), Figure 1. Test loss against compute, dataset size, and parameters.

Taking logs, $\log \mathcal{R} = \alpha_N \log N_c - \alpha_N \log N$: a power law is a straight line on log-log axes with slope $-\alpha_N$, which is what each panel shows. The paper reports "no signs of deviation from these trends on the upper end".

Scale, not shape. At a fixed parameter count the loss depends "very weakly on model shape and other Transformer hyperparameters (depth, width, number of self-attention heads)".

Reading the exponents: each factor of 10 buys the same ratio



Computed from the three fitted exponents.

Multiplying N by s multiplies the loss by $s^{-\alpha_N}$, whatever the starting size.

	N	D	$C_{\min}$
loss factor at $2\times$	0.95	0.94	0.97
loss factor at $10\times$	0.84	0.80	0.89
scale-up to halve the loss	9×10^3	1.5×10^3	10^6

Halving needs $s = 2^{1/\alpha}$, for example $2^{1/0.076} \approx 9100$.

The exponents are small, so each gain is expensive. They are constant, so returns do not stop inside the measured range.

Limit. Text has non-zero entropy, so the loss "must flatten out eventually before reaching zero".

The joint law in N and D: data must grow with the model, but more slowly

One equation covers both factors (Kaplan et al., Eq. 1.5; joint fit $\alpha_N = 0.076$, $\alpha_D = 0.103$):

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

Limits. $D \to \infty$ recovers $\mathcal{R}(N)$ and $N \to \infty$ recovers $\mathcal{R}(D)$. With D fixed, growing N stops helping once the first term falls below D_c/D: the model overfits the finite dataset (the paper's *universality of overfitting*).

How much data avoids the penalty. The two terms keep a fixed ratio when $D \propto N^{\alpha_N/\alpha_D} = N^{0.74}$. Scale the model by 8 and solve for the data factor that keeps $N^{0.74}/D$ unchanged:

$$\frac{N^{0.74}}{D} = \frac{(8N)^{0.74}}{D'} \implies \frac{D'}{D} = 8^{0.74} = e^{0.74 \ln 8} = e^{1.54} \approx 4.7.$$

Result. An $8\times$ larger model needs only about $5\times$ the data (the paper's headline), and a $1000\times$ larger model about $1000^{0.74} \approx 166\times$: data needs are sublinear in model size.

Kaplan et al.'s budget advice: put the compute into model size

Cost model. A forward pass costs about $2N$ floating-point operations per token and the backward pass about twice that, so training costs $C \approx 6N$ per token.

Finding. For a fixed budget, the best loss comes from a very large model stopped "significantly short of convergence". The measured optimal allocation is

$$N_{\text{opt}} \propto C_{\text{min}}^{0.73}, \qquad D_{\text{opt}} \propto C_{\text{min}}^{0.27}.$$

Budget grows by	model grows by	data grows by
$10\times$	$10^{0.73} \approx 5.4\times$	$10^{0.27} \approx 1.9\times$
$1000\times$	$1000^{0.73} \approx 155\times$	$1000^{0.27} \approx 6.5\times$

What the field did with it. GPT-3 (175B parameters), Gopher (280B) and their peers grew N and kept D near 300 billion tokens. Section 2: a larger, more careful study finds this split wrong.

2. Compute-Optimal Training

Given a budget C, how large a model and how much data?

The problem: minimize the loss over (N, D) subject to $6ND = C$

Hoffmann et al. (2022, DeepMind) state the question as a constrained optimization:

$$N_{\text{opt}}(C),\ D_{\text{opt}}(C) \;=\; \underset{N,\,D\,:\,\mathrm{FLOPs}(N,D)=C}{\arg\min} \mathcal{R}(N, D), \qquad \mathrm{FLOPs}(N, D) \approx 6ND.$$

Model	**Parameters** N	**Training tokens** D	$6ND$ **(FLOPs)**
GPT-3	175B	300B	3.2×10^{23}
Jurassic-1	178B	300B	3.2×10^{23}
Gopher	280B	300B	5.0×10^{23}
Megatron-Turing NLG	530B	270B	8.6×10^{23}
Chinchilla	**70B**	**1.4T**	5.9×10^{23}

(N, D from Hoffmann et al., Table 1; last column computed here.) All but Chinchilla sit near $D = 300$B while N varies by $3\times$. To find the optimum, the paper trains over 400 models, 70M to 16B parameters on 5B to 500B tokens, and estimates $N_{\text{opt}}(C)$ three ways.

A fitted loss model with three terms

The paper's third approach fits one parametric loss to all runs (Hoffmann et al., Eqs. 2 and 10; the symbols E, A, B, α, β are theirs and are used only in this section):

$$\mathcal{R}(N, D) = \underbrace{E}_{\text{irreducible}} + \underbrace{\frac{A}{N^{\alpha}}}_{\text{finite model}} + \underbrace{\frac{B}{D^{\beta}}}_{\text{finite training}}$$

$$E = 1.69, \qquad A = 406.4, \qquad B = 410.7, \qquad \alpha = 0.34, \qquad \beta = 0.28.$$

- E: the loss of an ideal model, the entropy of text; no scale removes it.
- A/N^{α}: what a perfectly trained model with only N parameters still loses.
- B/D^{β}: what is lost by stopping after D tokens instead of training to convergence.

Plug in. Gopher's split (280B, 300B) gives $1.69 + 0.05 + 0.25 = 1.99$; Chinchilla's (70B, 1.4T) gives $1.69 + 0.08 + 0.16 = 1.94$. Gopher's loss is dominated by the training term: too few tokens for its size.

The optimal split in closed form: balance the two reducible terms

Derivation. Substitute the constraint $D = C/(6N)$ and set the derivative in N to zero:

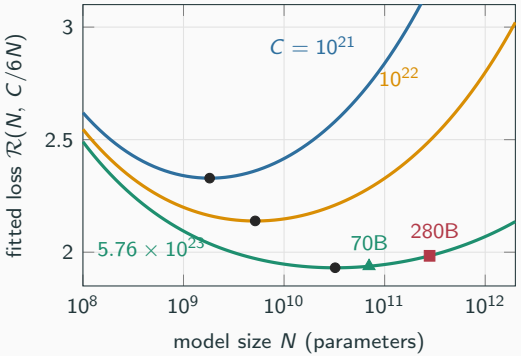
$$\frac{d}{dN}\left[\frac{A}{N^{\alpha}} + B\left(\frac{6N}{C}\right)^{\beta}\right] = -\frac{\alpha A}{N^{\alpha+1}} + \frac{\beta B}{N}\left(\frac{6N}{C}\right)^{\beta} = 0 \quad \Longrightarrow \quad \alpha\,\frac{A}{N^{\alpha}} = \beta\,\frac{B}{D^{\beta}}.$$

At the optimum the model term and the training term balance, weighted by their exponents. Solving for N (Hoffmann et al., Eq. 4):

$$N_{\text{opt}} = G\left(\frac{C}{6}\right)^{a}, \quad D_{\text{opt}} = G^{-1}\left(\frac{C}{6}\right)^{b}, \qquad a = \frac{\beta}{\alpha+\beta}, \;\; b = \frac{\alpha}{\alpha+\beta}, \;\; G = \left(\frac{\alpha A}{\beta B}\right)^{\frac{1}{\alpha+\beta}}.$$

Numbers. $a = 0.28/0.62 \approx 0.45$ and $b = 0.34/0.62 \approx 0.55$ (the paper reports 0.46 and 0.54): model and data should grow at nearly the same rate with compute. Kaplan et al.'s 0.73 and 0.27 put most of the growth in the model.

The trade-off at a fixed budget: too small underfits, too large is undertrained



Computed from the fitted $\mathcal{R}(N, D)$ with $D = C/6N$. Dots: the closed-form optimum $N_{\text{opt}}(C)$.

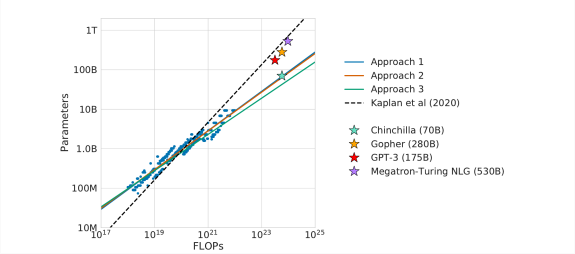
Each curve fixes C and varies the split. Left of the minimum A/N^{α} dominates (too few parameters); right of it B/D^{β} dominates (too few tokens seen).

At $C = 5.76 \times 10^{23}$	N	**loss**
closed-form optimum	32B	1.931
Chinchilla's size	70B	1.938
Gopher's size	280B	1.984

The rounded constants give 32B; the paper's own fit projects 40B and its Approach 1 projects 67B.

The optimum moves right by $10^{0.45} \approx 2.8\times$ per $10\times$ compute. The curve is flat near its minimum: a $2\times$ error in N costs little, an $8\times$ error costs a lot.

Three independent estimates agree: scale N and D in equal proportion



Hoffmann et al. (2022), Figure 1. Predicted optimal parameter count against training FLOPs; Kaplan et al.'s projection dashed; stars mark Chinchilla, Gopher, GPT-3, and Megatron-Turing NLG.

Approach ($N_{\text{opt}} \propto C^a$, $D_{\text{opt}} \propto C^b$)	a	b
1. Minimum over training curves	0.50	0.50
2. IsoFLOP profiles	0.49	0.51
3. Parametric loss fit	0.46	0.54
Kaplan et al. (2020)	0.73	0.27

"For every doubling of model size the number of training tokens should also be doubled."

The existing large models lie *above* the frontier: too many parameters for their compute. By the paper's Table 3, a 175B model is compute-optimal only when trained on 3.7 trillion tokens; GPT-3 saw 300 billion, about 12× fewer. In this sense those models were **undertrained**.

The test: Chinchilla, a quarter of Gopher's size at the same compute

Gopher
280B parameters
300B tokens

Chinchilla
70B parameters
1.4T tokens

same budget,
about 5.8×10^{23} FLOPs:
$N/4$ and roughly $4 \times D$

Result. Chinchilla "uniformly and significantly outperforms Gopher (280B), GPT-3 (175B), Jurassic-1 (178B), and Megatron-Turing NLG (530B) on a large range of downstream evaluation tasks". On MMLU it reaches 67.5% average accuracy, more than 7 points above Gopher.

A second benefit. Inference costs about $2N$ per token, so the $4\times$ smaller model is also about $4\times$ cheaper to serve and to fine-tune.

The prediction came from small runs (at most 16B parameters) and was confirmed at 70B: the fitted law was used to plan an experiment $4\times$ beyond the data, and it held.

Why the first estimate was off, and what survived

The paper's own account of the difference from Kaplan et al.

1. **One schedule for all runs.** Kaplan et al. used a fixed number of training tokens and one learning-rate schedule for every model. A loss read off partway through such a run is worse than what a shorter run with a matched schedule reaches, so small-data runs look worse than they are, and the optimum is pushed toward large N.
2. **Small models.** Most of Kaplan et al.'s runs were under 100M parameters. The frontier curves slightly at larger sizes; Hoffmann et al. fit models up to 16B, most above 500M.

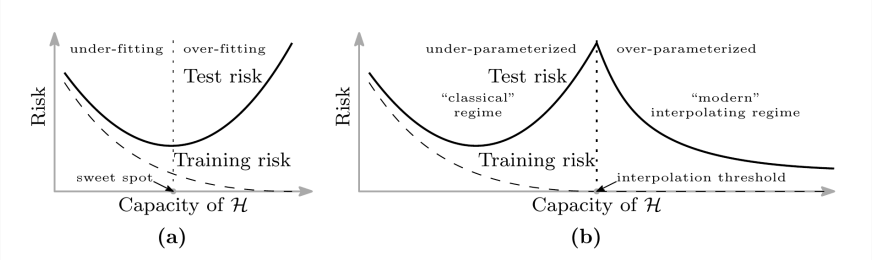
What survived. The power-law form, the cost model $C \approx 6ND$, and the weak dependence on shape. Only the allocation exponents changed, from (0.73, 0.27) to about (0.5, 0.5).

Lesson. An empirical law is as good as the experiments behind it. How to split a budget is a measurable question, and its first answer was wrong by enough to matter: the same compute, split differently, gave a better model.

3. Scaling Laws and Double Descent

Two places the classical picture stops predicting

Recall double descent (Lecture 04): test error falls again past interpolation



Belkin, Hsu, Ma, Mandal (2019), Figure 1. Left: the classical U-shaped curve. Right: the second descent past the interpolation threshold.

Classical expectation (Lecture 03). Test error is bias plus variance; past the best capacity, variance grows, and a model large enough to fit noisy training data exactly should test badly.

What is measured. Past the interpolation threshold, where training error first reaches zero, test error comes down a second time as capacity keeps growing. The U-curve is correct up to the threshold and silent beyond it.

Scaling laws are a second measured regularity the classical picture does not give

	Double descent (Lecture 04)	**Scaling laws (today)**
Classical expectation	error rises once the model can fit the training data exactly	returns from more parameters fade as variance takes over
What is measured	a second descent past the interpolation threshold	a power law over six orders of magnitude in N, with no deviation seen
What varies	capacity, at fixed data	N, D, C, usually together
Source	Belkin et al. (2019)	Kaplan et al. (2020); Hoffmann et al. (2022)

The mechanisms differ: double descent is about one model class as capacity crosses the interpolation threshold; scaling laws track the loss as model, data, and compute grow together, with language models usually far from fitting their training data exactly.

Connection. The status is the same. Neither curve follows from the bias-variance trade-off of a fixed model class; both had to be measured, and practice at this scale is planned from such fits, like $\mathcal{R}(N, D)$ in Section 2.

4. In-Context Learning

Learning a task from the prompt, with no weight update

In-context learning: the training set goes in the prompt, the weights stay fixed

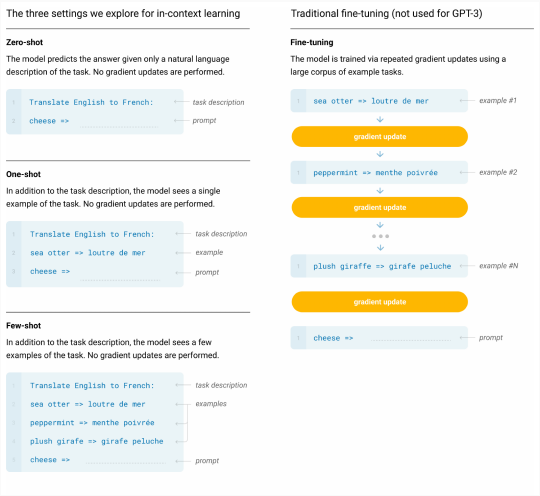
Definition. Given K demonstrations $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_K, y_K)$ of a task and a new input $\mathbf{x}$, write them as one token sequence and let a pretrained model p_θ continue it:

$$\hat{y} \;=\; \arg\max_{y} \; p_\theta\big(y \;\big|\; \underbrace{\text{instruction},\; \mathbf{x}_1, y_1,\; \ldots,\; \mathbf{x}_K, y_K,\; \mathbf{x}}_{\text{the prompt}}\big), \qquad \theta \text{ unchanged.}$$

Example ($K = 2$). The prompt `sea otter => loutre de mer; peppermint => menthe poivrée; cheese =>` is continued with `fromage`. Nothing told the model the task is translation except the two demonstrations.

> **Terms** (Brown et al., 2020). **Zero-shot**: $K = 0$, an instruction only. **One-shot**: $K = 1$. **Few-shot**: as many demonstrations as fit the context window, typically 10 to 100 for GPT-3's 2048 tokens. All three are in-context learning.

What the prompts look like, next to fine-tuning



Brown et al. (2020), Figure 2.1.

Left: the three prompt formats differ only in how many solved examples precede the query. The model reads the prompt once and predicts the next tokens.

Right: fine-tuning shows the same examples one at a time and takes a gradient step after each.

Same examples, two uses: as *training data* that changes θ, or as *input* that a fixed θ conditions on.

In-context learning against fine-tuning, and why it can work at all

	Fine-tuning (Lecture 08, Part II)	**In-context learning**
Where the task data goes	into the weights, by gradient steps	into the input, as text
Update	$\theta \leftarrow \theta - \eta \nabla_\theta \hat{\mathcal{R}}(\theta)$, many times	none; one forward pass
Labeled examples used	thousands and more	$K = 0$ to about 100
What persists afterwards	a new model per task	nothing; the prompt is sent again
Limit	needs a dataset per task	examples must fit the context

Why it can work. Section 1: any task can be phrased as text continuation. Ordinary text also contains many passages of the form "example, example, example, query" (lists, tables, question-answer pages), and predicting their endings well requires working out the pattern from the earlier entries.

Brown et al. describe this as two loops: pretraining is an outer loop that builds general pattern-recognition skills, and the forward pass is an inner loop that recognizes the task at hand; "in-context learning" names the inner loop.

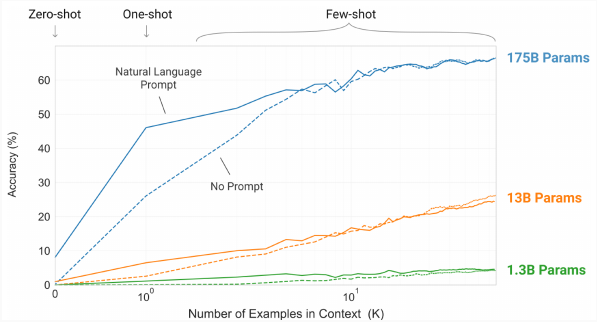
GPT-1 to GPT-3: how a new task is specified changed with scale

	Size	Pretraining data	How a new task is specified
GPT-1 (2018)	117M, 12 layers	BooksCorpus, over 7,000 books	labeled set and fine-tuning per task
GPT-2 (2019)	1.5B, 48 layers	WebText, about 40 GB	a text prompt, zero-shot
GPT-3 (2020)	175B, 96 layers	filtered Common Crawl and curated sources; 300B tokens seen	demonstrations in the prompt

> *"For all tasks, GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model."* (Brown et al., 2020)

Measured on TriviaQA (question answering with no documents, GPT-3 175B): 64.3% zero-shot, 68.0% one-shot, 71.2% few-shot. The few-shot number is above the fine-tuned retrieval system of the time (RAG, 68.0%, the method of today's discussion), with no task-specific training.

In-context learning improves with more demonstrations, and with model size



Brown et al. (2020), Figure 1.2. Accuracy on removing random symbols from a word against the number K of examples in context, for three model sizes, with (solid) and without (dashed) a task description.

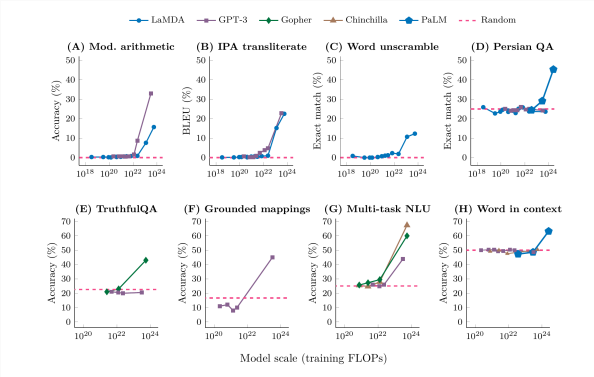
- Accuracy rises with K: the model uses the demonstrations.
- At the same K, the 175B model passes 60%, the 13B model stays near 25%, the 1.3B model below 10%.
- A task description helps most when K is small.

Learning from context is itself a function of scale: nearly absent at 1.3B, strong at 175B, while the pretraining loss falls smoothly. Section 5 asks how to read that.

5. Emergent Abilities

Real discontinuity, or measurement artifact?

The claim (Wei et al., 2022): abilities absent in small models, present in large



Wei et al. (2022), Figure 2. Eight few-shot tasks, five model families; the horizontal axis is model scale, logarithmic.

> *"We consider an ability to be emergent if it is not present in smaller models but is present in larger models. Thus, emergent abilities cannot be predicted simply by extrapolating the performance of smaller models."*

In each panel the score stays near the random baseline over orders of magnitude of scale, then rises: the change looks **sharp**, and its location **unpredictable** from the smaller models.

Yet the loss of the same model families falls smoothly with scale (Section 1).

The critique (Schaeffer et al., 2023): a smooth loss under a harsh metric

"Nonlinear or discontinuous metrics produce apparent emergent abilities, whereas linear or continuous metrics produce smooth, continuous, predictable changes in model performance."

Their model. (1) Assume only what Section 1 measured: per-token cross-entropy falls as a power law, $\mathcal{R}(N) = (N_c/N)^{\alpha_N}$. (2) Cross-entropy is $-\log$ of the probability given to the correct token, so one token is right with probability

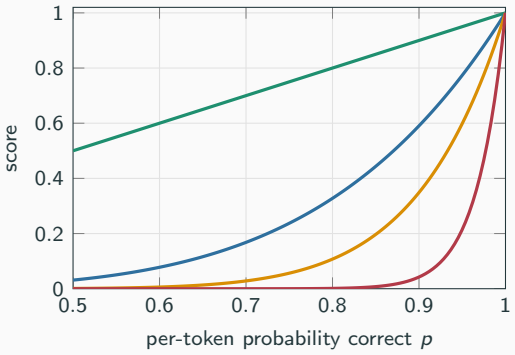
$$p(N) = \exp\big(-\mathcal{R}(N)\big), \quad \text{which rises smoothly toward 1.}$$

(3) Score a task whose answer is k tokens long (say k-digit addition) two ways, treating tokens as independent:

$$\underbrace{\text{exact-match accuracy} = p(N)^k}_{\text{all } k \text{ right, or zero}} \qquad \underbrace{\text{expected wrong tokens} = k\,\big(1 - p(N)\big)}_{\text{partial credit, linear in } p}$$

Both scores are functions of the *same* smooth $p(N)$. The first is geometric in k: near 0 until p is close to 1, then rising fast. The second moves in proportion to p.

Computed example: one skill under exact match and under partial credit



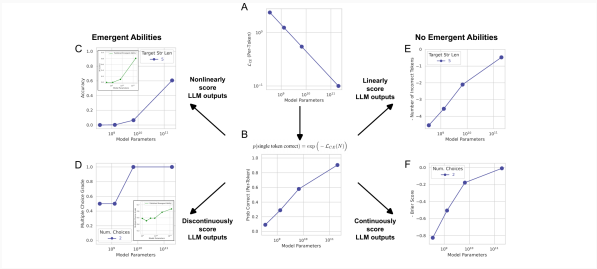
Computed from p^k, not measured model data. Colors match the table rows.

p	0.80	0.90	0.95	0.99
tokens right	80%	90%	95%	99%
exact, $k = 5$	33%	59%	77%	95%
exact, $k = 10$	11%	35%	60%	90%
exact, $k = 30$	0.1%	4%	21%	74%

From $p = 0.80$ to 0.99 the partial-credit score moves by 19 points. Exact match on a 30-token answer moves from 0.1% to 74%, nearly all of it above $p = 0.9$.

Plotted against $\log N$ at a few model sizes, the red curve reads as "absent, then present". The longer the answer, the later and sharper the jump.

The paper's figure and its evidence on real models



Schaeffer, Miranda, Koyejo (2023), Figure 2. (A) loss falls smoothly with scale; (B) per-token probability correct; (C, D) nonlinear and discontinuous metrics show a jump; (E, F) linear and continuous metrics on the same outputs do not.

Tested three ways.

- GPT-3 family arithmetic: token edit distance on the *same outputs* improves smoothly.
- BIG-Bench: over 92% of claimed emergent abilities appear under two metrics, Multiple Choice Grade and Exact String Match.
- Vision models: a harsh metric produces "emergence" on demand.

Scope. The claim is about how emergence is measured. It does not show that every sharp change is an artifact, and exact match is often the metric that matters: a sum with one wrong digit is wrong.

A measured limit: multi-step errors remain at the largest scale

Dziri et al. (2023) test compositional tasks such as long multiplication, where each output digit depends on several earlier partial results.

59%
GPT-4, zero-shot,
3-digit $\times$ 3-digit

55%
ChatGPT, zero-shot,
3-digit $\times$ 3-digit

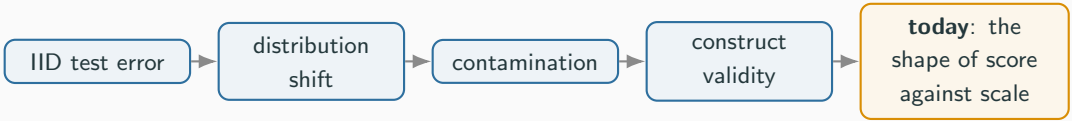
a task people do reliably by hand once they know the algorithm; accuracy falls further with more digits

This is the p^k picture on real models: a per-step error rate that is small but not zero, compounded over many dependent steps, leaves the whole product wrong more than 40% of the time.

The metric critique does not say large models have no limits. Scale moved p toward 1; it has not made long chains of exact steps reliable. Today's discussion covers one remedy: hand the arithmetic to a tool.

Back to Lecture 01: a clean score can still be misread

Lecture 01 asked whether a benchmark number means what it is used to mean, and the question became harder at each step.



Today's case is one step further: the test set can be clean and the task well chosen, and the reading still wrong, because whether a curve looks like a jump or a slope depends on the metric. A sharp rise may be a new capability or p^k at work.

Practice. Before calling a jump a new capability, plot a continuous metric on the same outputs (per-token accuracy, edit distance, log-likelihood) and use enough test examples to resolve small accuracies. Which claimed abilities survive that check is still open.

Summary: what scale buys, and how to measure it

- **Scaling laws.** $\mathcal{R}(N) = (N_c/N)^{0.076}$ and its analogues in D and C are straight lines on log-log axes over six or more orders of magnitude; $10\times$ the parameters lowers the loss by 16%; shape matters little.
- **Compute-optimal training.** With $C \approx 6ND$ and $\mathcal{R} = E + A/N^{\alpha} + B/D^{\beta}$, the optimum balances $\alpha A/N^{\alpha} = \beta B/D^{\beta}$, so N and D grow about equally with C. Chinchilla (70B, 1.4T tokens) beat Gopher (280B, 300B tokens) at equal compute.
- **Double descent, again.** Like Lecture 04's curve, the scaling law is a measured regularity that the bias-variance picture does not predict.
- **In-context learning.** Demonstrations go in the prompt and θ stays fixed; TriviaQA goes from 64.3% zero-shot to 71.2% few-shot; the ability grows with model size.
- **Emergent abilities.** Exact match on a k-token answer is p^k, so a smooth p gives a sharp curve. Check a continuous metric before reading a jump as a new capability.

Coming up

Discussion 10 (today): two ways to extend a language model past its weights at inference time. Retrieval-augmented generation looks facts up; agents call tools in a reason-act-observe loop (ReAct).

HW4 is due Monday 11/30, 11:59pm.

Lecture 11 (11/20): reinforcement learning. Bandits, Q-learning, policy gradients, and how RLHF and RLVR use them to post-train the language models of today's lecture.

Colab. `scaling_laws_emergent_abilities_demo.ipynb` fits $\mathcal{R}(N) = aN^{-b} + c$ to noisy synthetic data, then scores one smoothly improving skill curve with a linear metric and with exact match.

References

- Kaplan, McCandlish, Henighan, Brown, Chess, Child, Gray, Radford, Wu, Amodei. *Scaling Laws for Neural Language Models.* arXiv:2001.08361, 2020.
- Hoffmann, Borgeaud, Mensch, et al. *Training Compute-Optimal Large Language Models.* NeurIPS 2022; arXiv:2203.15556. (Chinchilla.)
- Belkin, Hsu, Ma, Mandal. *Reconciling Modern Machine-Learning Practice and the Classical Bias-Variance Trade-off.* PNAS 116(32), 2019; arXiv:1812.11118.
- Radford, Narasimhan, Salimans, Sutskever. *Improving Language Understanding by Generative Pre-Training.* OpenAI, 2018. (GPT-1.)
- Radford, Wu, Child, Luan, Amodei, Sutskever. *Language Models are Unsupervised Multitask Learners.* OpenAI, 2019. (GPT-2.)
- Brown, Mann, Ryder, et al. *Language Models are Few-Shot Learners.* NeurIPS 2020; arXiv:2005.14165. (GPT-3.)
- Wei, Tay, Bommasani, et al. *Emergent Abilities of Large Language Models.* TMLR, 2022; arXiv:2206.07682.
- Schaeffer, Miranda, Koyejo. *Are Emergent Abilities of Large Language Models a Mirage?* NeurIPS 2023; arXiv:2304.15004.
- Dziri, Lu, Sclar, et al. *Faith and Fate: Limits of Transformers on Compositionality.* NeurIPS 2023; arXiv:2305.18654.