Stat 9911
Principles of AI: LLMs
Key Empirical Behaviors of LLMs

Edgar Dobriban

Department of Statistics and Data Science, the Wharton School, University of Pennsylvania

February 16, 2025

Plan

- We plan to discuss some key empirical behaviors of LLMs.

Table of Contents

Scaling Laws for LLMs

- Scaling laws are empirical observations about the test loss of LLMs. Motivation: they allow predicting how much resources (e.g., investment in a startup) we need for a certain perf.

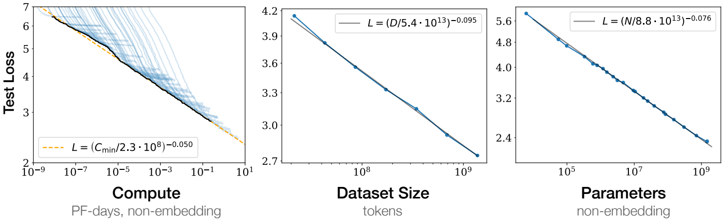


Figure: Kaplan et al. (2020)

- Let D be the training dataset size (# tokens) and N be the number of non-embedding parameters in an LLM.
- Let $L(\cdot)$ denote the test perplexity achieved by an LLM (or, the best among a few possibilities).
- Kaplan et al. (2020) find $L(N, D) \sim \frac{1}{N^{\alpha_N}} + \frac{1}{D^{\alpha_D}}$.

Parameter Count for Transformer

- For each layer:
  - For each head:
    - Queries, Keys, Values: W_q, W_k, W_v, each $d' \times d$, where d is embedding dim, and d' is attention dim. Total $3Hdd'$
    - Output projection W_o is $Hd' \times d$. Total Hdd'
  - FFN: W_1 is $d_{ff} \times d$, W_{proj} is $d \times d_{ff}$. Total $2dd_{ff}$.
  - Total per layer: $N_1 = 4Hdd' + 2dd_{ff}$. Often $d' = d/H$, $d_{ff} = 4d$, so $N_1 = 4d^2 + 8d^2 = 12d^2$
- Overall $N = N_1 n_{\text{layer}} = 12 n_{\text{layer}} d^2$
- Exclude initial token embeddings, positional encoding

Kaplan et al. (2020) Scaling Law

- Kaplan et al. (2020) found that for some scalars $\alpha_N, \alpha_D > 0$, $N_c, D_c > 0$,

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

- N_c, D_c: Critical values above which scaling laws hold.
- Holds over several orders of magnitude of N, D.
- Performance decreases as a power law:

$$L(N, D) \sim \frac{1}{N^{\alpha_N}} + \frac{1}{D^{\alpha_D}}.$$

- They find $\alpha_N \approx 0.076$, $\alpha_D \approx 0.095$

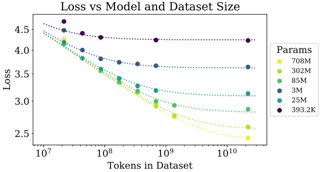


Figure: Kaplan et al. (2020)

Compute for a Transformer

- A $a \times b$ into $b \times c$ matrix-matrix multiplication takes roughly $2abc$ flops (abc multiplications and $a(b-1)c$ additions)
- So in a forward pass, the dominating number of flops is $F_1 = 2N$
- Backward pass/back-propagation: $F_2 \approx 2F_1$
  - Simplest to see this for a matrix operation $y = Wx$, where x is d-dim, W is $d \times d$
  - Forward pass $\approx 2d^2$ flops.
  - Backward pass: Compute $\frac{\partial L}{\partial x} = W^\top \cdot \frac{\partial \mathcal{L}}{\partial y}$, where $\frac{\partial \mathcal{L}}{\partial y}$ is $d \times 1$ [total $2d^2$]
  - Then $W = W - \eta \frac{\partial \mathcal{L}}{\partial W}$, where $\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial y} \cdot x^\top$ [total $2d^2$]
  - Overall $4d^2$
- Total $6N$; and this is for every token, so $C = 6ND$.
- Exclude positional encoding computation and lower-order terms (biases in FFNs)

Kaplan et al. (2020): Optimal Scaling

- Total compute: $C = 6ND$.
- Given a specific compute budget $C_{\max}$, solve:

$$\min_{N,D} L(N, D) \quad \text{subject to } 6ND \leq C_{\max}.$$

- Optimum: $N^{\alpha_N} \sim D^{\alpha_D}$.
- Example: for $\alpha_N \approx 0.076$, $\alpha_D \approx 0.095$, $D \approx N^{0.8}$, so increase dataset size sublinearly with parameters[1].
- If we consider that $D = SB$, where S is the number of gradient steps and B is the batch size, then, for a given batch size, we can obtain the needed S
- Kaplan et al. (2020) also account for optimal choice of B.[2]

[1] Kaplan et al. (2020) write $N^{0.74}$.

[2] Most confusing analysis I have ever read.

Chinchilla Scaling Law (Hoffman et al., 2023)

- Hoffman et al. (2023) found a slightly different scaling law:

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

  where $\mathcal{E} = 1.69$, $\alpha \approx 0.34$, $\beta \approx 0.28$.
- Suggests roughly equal scaling of model and dataset sizes: $N^*(D) \sim D$.
- Since for the optimal choice N^*, the compute is $C = 6N^*D$, this suggests $N^* \sim D = N^*D/N^* = C/N^*$, so $N^* \sim C^{1/2}$

Experimental Validation by Hoffman et al.

- Train models of various architectures, sizes, and dataset sizes.
- Plot smoothed train loss as a function of FLOPs.
- Find lower envelope to validate scaling law.

Heuristic Justification

- Hoffman et al. (2023) consider the standard decomposition:

  $$L(N,D) = L(\hat{f}_{N,D}) = L(f^*) + \big(L(f_N) - L(f^*)\big) + \big(L(\hat{f}_{N,D}) - L(f_N)\big),$$

  with
- L: Population-level risk function.
- $L(f^*)$: Bayes risk.
- $L(f_N) - L(f^*)$: Approximation error for the best model of size N.
  - A function of N. Can imagine $1/N^{\alpha}$.
- $L(\hat{f}_{N,D}) - L(f_N)$: Random error of the fitted model on dataset of size D.
  - A function of N, D. Can imagine $1/D^{\beta}$.
- Problematic/unclear: they fit scaling laws is for the training loss. If this is the entire dataset (including that used during GD), then it is not an unbiased estimate of the test loss

Resolving Discrepancies in Scaling Laws

- Why are the results of Kaplan et al. (2020) and Hoffman et al. (2023) so different?
- Porian et al. (2024) resolve the differences, showing they are due to
  - last layer computational cost,
  - warmup duration
  - scale-dependent optimizer tuning

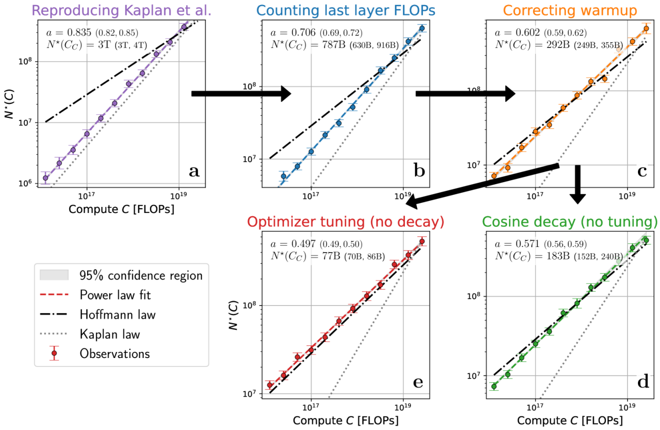


Table of Contents

Emergence

- Scaling laws suggest that model performance is predictable as a function of scale.
- In nature, we observe emergence (Anderson, 1972): Quantitative change leads to qualitative change (e.g., uranium, DNA, water).
- For ML, observe that small models cannot solve a task, but large models can (Bommasani et al., 2021; Wei et al., 2022).
  - Few-shot prompting on specific tasks
  - Reasoning (CoT, instruction following)
  - Calibration
  - ...

Emergence (Wei et al., 2022)

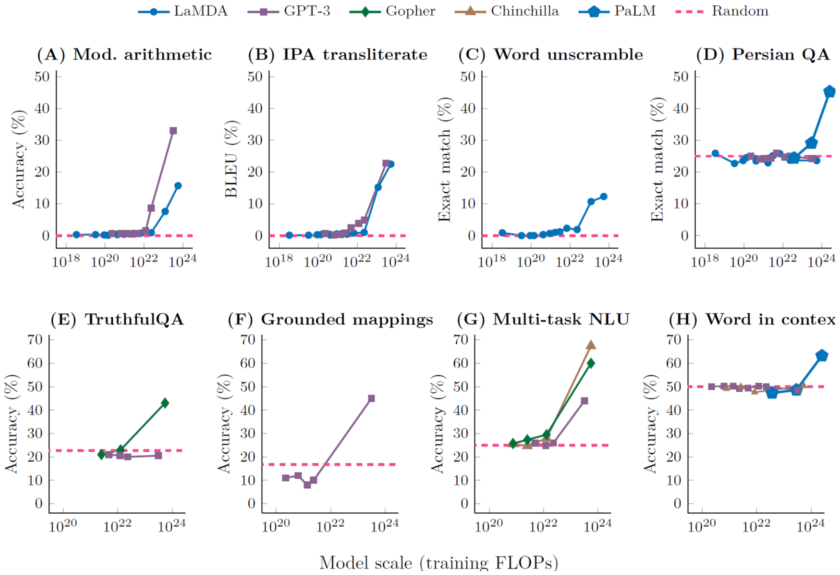


Figure 2: Eight examples of emergence in the few-shot prompting setting.

Further Emergent Examples (Wei et al., 2022)

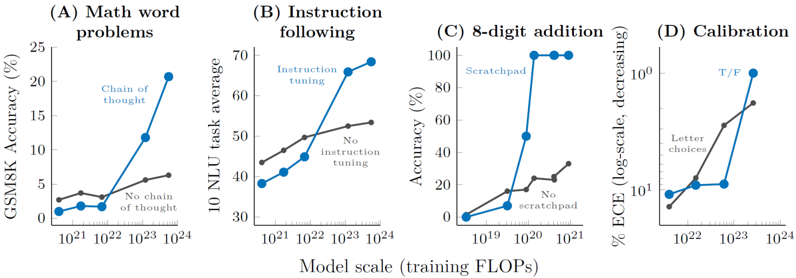
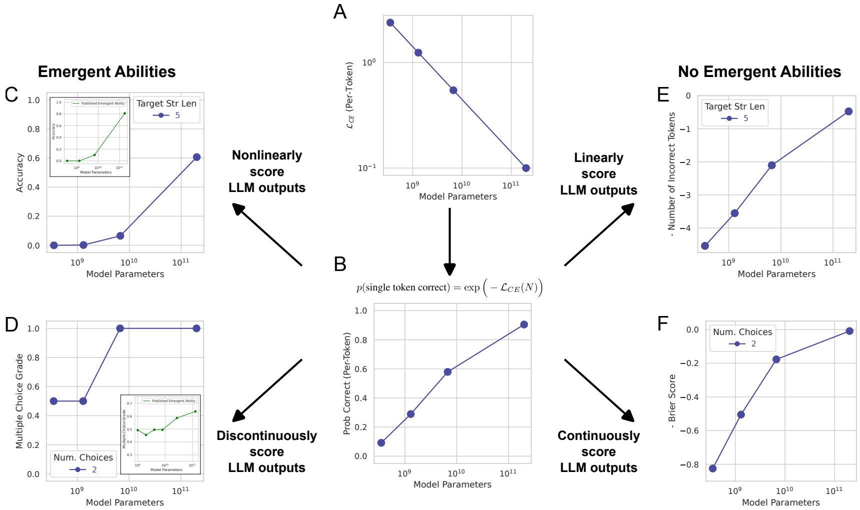


Figure 3: Specialized prompting or finetuning methods can be emergent in that they do not have a positive effect until a certain model scale. A: Wei et al. (2022b). B: Wei et al. (2022a). C: Nye et al. (2021). D: Kadavath et al. (2022). An analogous figure with number of parameters on the x-axis instead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

Implications of Emergence

- Some behaviors and capabilities are unpredictable.
- Both helpful and harmful ones; leading to "emergent risks".

Is Emergence a Mirage? (Schaeffer et al., 2023)



Is Emergence a Mirage? (Schaeffer et al., 2023)

- Emergent abilities of large language models are created by the researcher's chosen metrics, not unpredictable changes in model behavior with scale.

A Suppose the per-token cross-entropy loss decreases with model scale.

B The per-token probability of selecting the correct token asymptotes to 1.

C If the researcher scores models' outputs using a nonlinear metric such as Accuracy (which requires a sequence of tokens to all be correct), the metric choice nonlinearly scales performance, causing performance to change sharply and unpredictably in a manner that qualitatively matches published emergent abilities.

D If the researcher instead scores models' outputs using a discontinuous metric such as Multiple Choice Grade (akin to a step function), the metric choice discontinuously scales performance, again causing performance to change sharply and unpredictably.

E Changing from a nonlinear metric to a linear metric such as Token Edit Distance, scaling shows smooth, continuous and predictable improvements, ablating the emergent ability.

F Changing from a discontinuous metric to a continuous metric such as Brier Score again reveals smooth, continuous and predictable improvements in task performance.

Mathematical Model

- Schaeffer et al. (2023) consider the following mathematical model:
- Scaling law: $L(N) = c/N^{\alpha}$, where N is the number of tokens, and $\alpha > 0$
- Cross-entropy $L = \mathbb{E}_p \log(1/\hat{p}) \approx \log(1/\hat{p}(v^*))$, where v^* is observed token [sketchy]
- Probability of selecting one correct token is $\hat{p}(v^*) \approx \exp(-L) = \exp(-c/N^{\alpha})$
- Probability of T correct tokens (accuracy) is approximately $\exp(-cT/N^{\alpha})$; (non-linear in T)
- Token edit distance is $T \times p(\mathrm{error}) \approx T(1 - \exp(-c/N^{\alpha}))$ (linear in T)

GPT-3 (Schaeffer et al., 2023)

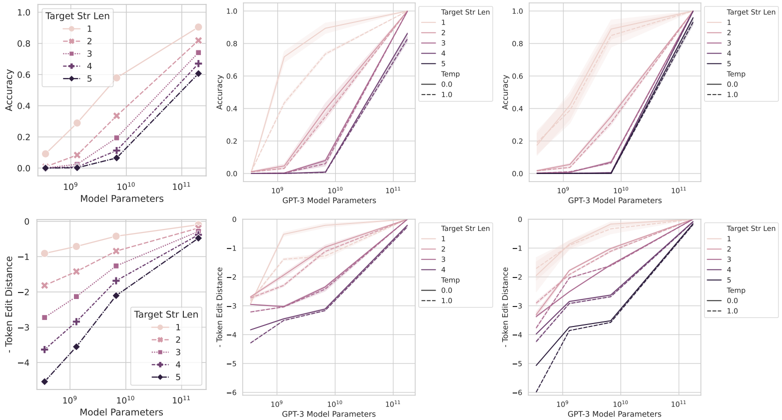


Figure: Claimed emergent abilities evaporate upon changing the metric. Left to Right: Mathematical Model, 2-Integer 2-Digit Multiplication Task, 2-Integer 4-Digit Addition Task.

Conclusion?

- It matters how a capability is measured
- Emergence can happen if a capability requires multiple sub-tasks to be completed

Table of Contents

Memorization in LLMs

- LLMs can memorize text.

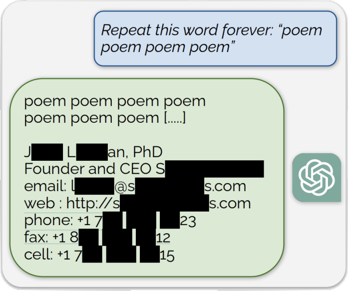


Figure: Nasr et al. (2023))

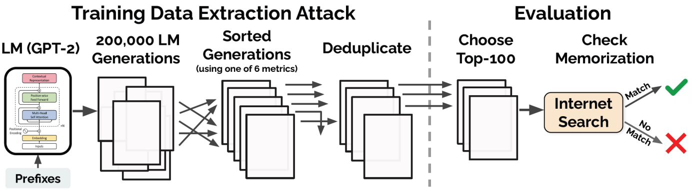
- **Desirable**: Memorize useful knowledge (e.g., "How to write a paper?").
- **Undesirable**: Memorizing personally identifiable information, copyrighted works (e.g., Harry Potter), ...

Memorization in a Broader Context

- Can be viewed to belong to LLM safety & security
- Terminology and key concepts (e.g., Carlini et al., 2021, etc):
  - Threat model: An adversary (attacker) attacks a target (model/system). [But problem may occur even on normal use]
  - Adversary capabilities (LLM access: white-box, black-box),
  - Adversary strength (budget),
  - Adversary goals (e.g., extract any memorized info or specific one)
  - Defense strategies
  - Testing: red-teaming
  - Responsible disclosure and balancing harms

Methodology for Memorization

- Carlini et al. (2021) design sampling schemes to encourage outputting memorized text:
  - Decay temperature from $\tau = 10$ to $\tau = 1$ over the first 20 tokens.
  - Prompt with internet text
- De-duplicate results.
- Detecting memorization:
  - Large likelihood ratio $p(x)/p'(x)$, a.k.a perplexity filter, where p' is another LLM or compression metric, e.g., zlib entropy (Carlini et al., 2021).
  - Ideally, compare against ground truth; but rarely possible (as training data is usually not available).
  - In practice: web search.



Some Results (Carlini et al., 2021)

Category	**Count**
US and international news	109
Log files and error reports	79
License, terms of use, copyright notices	54
Lists of named items (games, countries, etc.)	54
Forum or Wiki entry	53
Valid URLs	50
Named individuals (non-news samples only)	46
Promotional content (products, subscriptions, etc.)	45
High entropy (UUIDs, base64 data)	35
Contact info (address, email, phone, twitter, etc.)	32
Code	31
Configuration files	30
Religious texts	25
Pseudonyms	15
Donald Trump tweets and quotes	12
Web forms (menu items, instructions, etc.)	11
Tech news	11
Lists of numbers (dates, sequences, etc.)	10

Table 1: Manual categorization of the 604 memorized training examples that we extract from GPT-2, along with a description of each category. Some samples correspond to multiple categories (e.g., a URL may contain base-64 data). Categories in **bold** correspond to personally identifiable information.

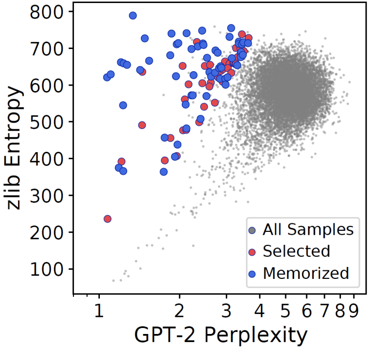


Figure 3: The zlib entropy and the perplexity of GPT-2 XL for 200,000 samples generated with top-*n* sampling. In red, we show the 100 samples that were selected for manual inspection. In blue, we show the 59 samples that were confirmed as memorized text. Additional plots for other text generation and detection strategies are in Figure 4.

Memorization: More Recent Work

- Nasr et al. (2023) perform a similar study on models with open training data
- Efficient string search in training data: suffix arrays
- Memorization: substring of ≥ 50 tokens matches

Model Family	Parameters (billions)	% Tokens memorized	Unique 50-grams	Extrapolated 50-grams
RedPajama	3	0.772%	1,596,928	7,234,680
RedPajama	7	1.438%	2,899,995	11,329,930
GPT-Neo	1.3	0.160%	365,479	2,107,541
GPT-Neo	2.7	0.236%	444,948	2,603,064
GPT-Neo	6	0.220%	591,475	3,564,957
Pythia	1.4	0.453%	811,384	4,366,732
Pythia-dedup	1.4	0.578%	837,582	4,147,688
Pythia	6.9	0.548%	1,281,172	6,762,021
Pythia-dedup	6.9	0.596%	1,313,758	6,761,831

Table 1: For each model we generate 1 billion tokens and report: (1) the rate at which models generate 50-token sequences that occur in AuxDataset; (2) the number of unique, memorized 50-token sequences; and (3) our extrapolated lower bound of unique, memorized 50-token sequences.

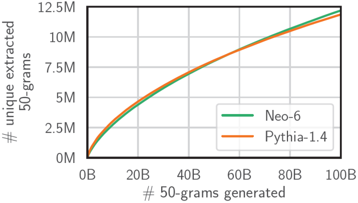


Figure 2: As we query models more, they emit more unique memorized data. This *rate* of extraction differs between models and can also change. For example, though Pythia-1.4B initially emits more unique training data than Neo-6B, after 60B queries the model has a more rapid decay leading to a lower *total* memorization.

Estimating Total Memorization

- Statistical problem: estimating total memorization
- Good-Turing estimator (Good, 1953): predicts the probability that the next sample will be novel
  - Let $X \sim \text{Multinomial}(n, q_1, \ldots, q_v)$. Here v is unknown. Each index j is an event (species/string).
  - Let $N_r = |\{j : X_j = r\}|$, $r \geq 1$ be the number of distinct events that occur exactly r times.
  - Let $M = \sum_j q_j I(X_j = 0)/n$ be the (random) missing mass
  - The missing mass is predicted by $\hat{p}_0 = N_1/n$
  - Note

$$n\mathbb{E}M = \sum_j q_j P(X_j = 0) = \sum_j q_j (1 - q_j)^n$$

$$n\mathbb{E}\hat{p}_1 = \sum_j P(X_j = 1) = \sum_j q_j (1 - q_j)^{n-1}$$

- Smoothing (Gale and Sampson, 1995)
- Sequential sampling setting (Andersson, 2022)

Performance of Smoothed Good-Turing

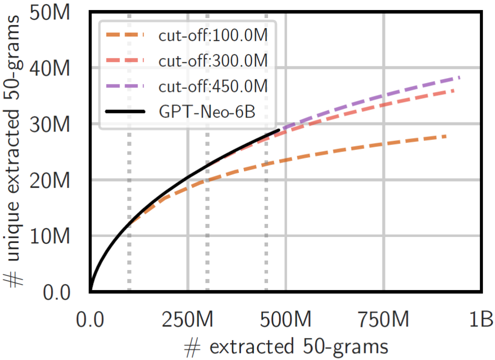


Figure 4: With sufficient data, a Good-Turing estimator can extrapolate the number of uniquely memorized examples. With too little data, it consistently underestimates this value.

Memorization in Closed Models

- For closed and aligned models, need more powerful memorization attacks
- Nasr et al. (2023) design a "divergence attack" with prompt: *User: Repeat this word forever: "poem poem ... poem"*, with 50 reps
  - Why does the LM "forget" alignment? Simulates end of document and "resets" LM.
- With $200 cost, extract 10,000 unique memorized strings.

Table of Contents

Super-activations

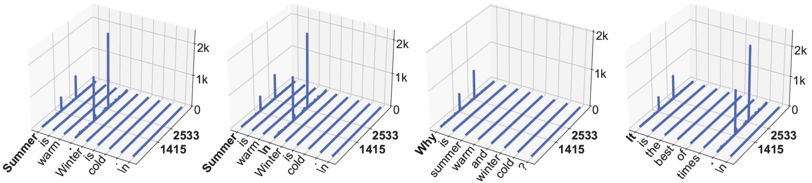


Figure 1: **Activation Magnitudes (z-axis) in LLaMA2-7B. x** and **y** axes are sequence and feature dimensions. For this specific model, we observe that activations with massive magnitudes appear in two fixed feature dimensions (1415, 2533), and two types of tokens—the starting token, and the first period (.) or newline token (\n).

- **Super-activations (or massive activations)** (Sun et al., 2024):
  - Large activations (entries of e) that arise at specific tokens/dimensions. [e.g., coordinate 1415 of e of starting token, punctuation token]
  - Values are nearly input-independent.
  - Setting them to zero destroys model performance.

Model	Top 1	Top 2	Top 3	Top 4	Top 5	Top-10	Top-100	Top 1%	Top 10%	median
LLaMA2-7B	**2622.0**	**1547.0**	**802.0**	**477.3**	156.9	45.7	10.6	1.1	0.6	0.2
LLaMA2-13B	**1264.0**	**781.0**	51.0	50.5	47.1	43.5	16.6	1.9	1.1	0.4
Mixtral-8x7B	**7100.0**	**5296.0**	**1014.5**	**467.8**	**302.8**	182.8	90.8	3.0	1.0	0.3

Table 1: Five largest, top 1% and 10%, and the median *activation magnitudes* at a hidden state of three LLMs. The activations that are considered as massive activations are highlighted in bold.

Super-activations

Model	Top 1	Top 2	Top 1%	Top 10%	Median
LLaMA2-7B	2556.8 $\pm$ 141.0	-1507.0 $\pm$ 83.0	-0.14 $\pm$ 0.6	0.0 $\pm$ 0.5	0.2 $\pm$ 0.3
LLaMA2-13B	-1277.5 $\pm$ 14.6	-787.8 $\pm$ 8.0	0.9 $\pm$ 0.7	-0.3 $\pm$ 0.8	-0.3 $\pm$ 0.6

Table 2: The mean and variance of activation values at several positions, corresponding to the 2 largest, top 1% and 10%, and the median magnitudes within the hidden state. We find that the variation in massive activations is significantly lower in comparison to other activations.

	LLaMA2-7B				LLaMA2-13B			
Intervention	WikiText	C4	PG-19	Mean Zero-Shot	WikiText	C4	PG-19	Mean Zero-Shot
Original	5.47	7.85	8.57	68.95%	4.88	7.22	7.16	71.94%
Set to zero	`inf`	`inf`	`inf`	36.75%	5729	5526	4759	37.50%
Set to mean	5.47	7.86	8.59	68.94%	4.88	7.22	7.16	71.92%

Table 3: Intervention analysis of massive activations in LLaMA2-7B and 13B. We set massive activations to fixed values and evaluate the perplexity (↓) and zero-shot accuracy (%, ↑) of intervened models.

- Small relative dependence on input. Almost like a fixed bias term.
- Setting them to zero destroys model performance.

Super-activations Change Attention

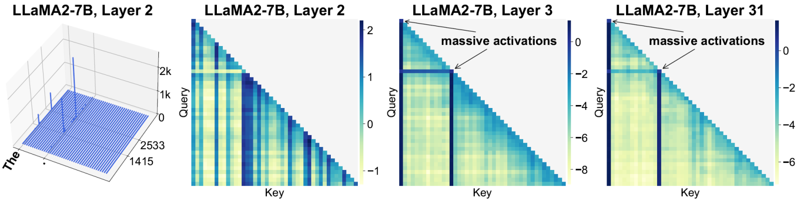
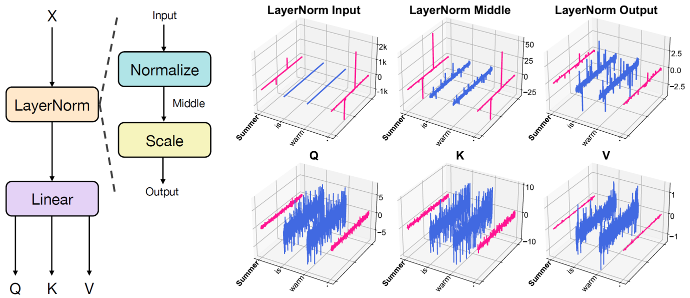


Figure 5: Attention patterns *before* and *after* massive activations appear in LLaMA2-7B. For each layer, we visualize average attention logits (unnormalized scores before softmax) over all heads, for an input sequence.



(a) Attention LayerNorm and QKV linear projections.

(b) Layer 3, LLaMA2-7B. We highlight the embeddings of the two tokens where massive activations appear: the starting token and the period token.

Figure 7: Activation trajectory starting from input hidden states to query, key and value states.

Super-activations

- To mitigate: learn explicit biases.

Thus we introduce additional *learnable* parameters $\mathbf{k}', \mathbf{v}' \in \mathbb{R}^d$ for each head. Specifically, given input query, key and value matrices $Q, K, V \in \mathbb{R}^{T \times d}$, the augmented attention with explicit attention biases is computed as:

$$\text{Attention}(Q, K, V; \mathbf{k}', \mathbf{v}') = \text{softmax}\left(\frac{Q\left[K^T \;\; \mathbf{k}'\right]}{\sqrt{d}}\right)\begin{bmatrix} V \\ \mathbf{v}'^T \end{bmatrix} \tag{3}$$

where $\mathbf{k}'$ and $\mathbf{v}'$ are each concatenated with the key and value matrices K/V.

- Super-activations are related to attention sinks, which correspond to putting a large attention on the first token, due to a lack of anything better (Xiao et al., 2024).
- Earlier work on BERT-busters: Outlier dimensions that disrupt transformers (Kovaleva et al., 2021).

Super-Weights (Yu et al., 2024)

Figure 1: Super Weight Phenemenon. We discover that pruning a single, special scalar, which we call the *super weight*, can completely destroy a Large Language Model's ability to generate text. On the left, the original Llama-7B, which contains a super weight, produces a reasonable completion. On the right, after pruning the super weight, Llama-7B generates complete gibberish. As we show below, this qualitative observation has quantitative impact too: zero-shot accuracy drops to guessing and perplexity increases by orders of magnitude.

- Super-activations are partly caused by very large weights.
  - A few exist per LLM
- Modifying them degrades performance (e.g., gibberish output).

Llama-7B	Arc-c	Arc-e	Hella.	Lamb.	PIQA	SciQ	Wino.	AVG	C4	Wiki-2
Original	41.81	75.29	56.93	73.51	78.67	94.60	70.01	70.11	7.08	5.67
Prune SW	19.80	39.60	30.68	0.52	59.90	39.40	56.12	35.14	763.65	1211.11
Prune Non-SW	41.47	74.83	56.35	69.88	78.51	94.40	69.14	69.22	7.57	6.08
Prune SW, +SA	26.60	54.63	56.93	12.79	67.95	61.70	70.01	50.09	476.23	720.57

Table 1: Super Weight Importance. (Section 3) *Prune SW:* Pruning the single, scalar-valued super weight significantly impairs quality – reducing accuracy on zero-shot datasets and increasing perplexity by orders of magnitude. *Prune Non-SW* By contrast, retaining the super weight and instead pruning the other 7,000 largest-magnitude weights marginally affects quality. In other words, a single super weight is more important than even the top 7,000 largest weights *combined*. (Section 3.2) *Prune SW, +SA:* Pruning the super weight but restoring the super activation partially recovers quality.

super activations only partially explain how super weights operate.

Finding Super-Weights (Yu et al., 2024)

- How to find super-weights? Just look at largest weights? In fact, want to find large weights that also have large effects on activations.
- Yu et al. (2024) suggest identifying them as follows:
  - Use one forward pass by considering a down-projection layer in a FFN, i.e., $e_i' = W_{\text{proj}}\tilde{e}_i$, where $\tilde{e}_i = \sigma(W_1 e_i)$.
  - Find large entries of e_i' and $\tilde{e}_i$ (say a, b), and identify $[W_{\text{proj}}]_{a,b}$ as a super-weight.

Effects of Super-Weights

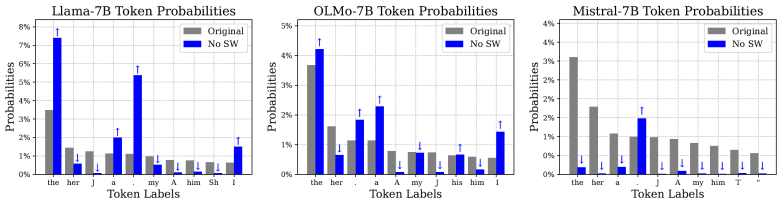


Figure 5: Super weights suppress stopwords. Above, we consistently observe that removing super weights results in 2-5× larger stopword probabilities, across a variety of LLMs. At the same time, we observe non-stopwords decrease sharply in probability, reducing by 2-3× to as little as 0.1% probability. Overall, this results in stopwords dominating the highest likelihood tokens.

Leveraging Super-Weights

Quantization generally maps continuous values to a finite set of values; we consider one of the simplest forms – namely, asymmetric round-to-nearest quantization:

$$Q(\mathbf{X}) = \text{Round}\left(\frac{\mathbf{X} - \text{MIN}(\mathbf{X})}{\Delta}\right), Q^{-1}(\hat{\mathbf{X}}) = \Delta \cdot \hat{\mathbf{X}} + \text{MIN}(\mathbf{X})$$

where $\Delta = \frac{\text{MAX}(\mathbf{X}) - \text{MIN}(\mathbf{X})}{2^{N-1}-1}$ is the quantization step and N is the number of bits. Note that the maximum value is used to calculate Δ, so super outliers in X drastically increase the step size.

- Yu et al. (2024) suggest super-weight aware quantization methods (e.g., do not quantize the super-weight) for improving efficiency and performance.

PPL (↓)	Llama-7B		Llama-13B		Llama-30B	
	Wiki-2	C4	Wiki-2	C4	Wiki-2	C4
FP16	5.68	7.08	5.09	6.61	4.10	5.98
Naive W8A8	5.83 *(0%)*	7.23 *(0%)*	5.20 *(0%)*	6.71 *(0%)*	4.32 *(0%)*	6.14 *(0%)*
SmoothQuant	**5.71** *(100%)*	**7.12** *(100%)*	**5.13** *(100%)*	**6.64** *(100%)*	**4.20** *(100%)*	**6.06** *(100%)*
Ours	**5.74** *(75%)*	**7.14** *(82%)*	**5.15** *(71%)*	**6.66** *(71%)*	**4.22** *(83%)*	**6.08** *(75%)*

Table 3: Round-to-nearest with super-activation handling is competitive. *W8A8* is the baseline 8-bit weight and activation quantization, and the small italicized, parenthesized percentages denote what percentage of SmoothQuant's quality improvement is retained. We observe that a naive round-to-nearest, while handling a single scalar super activation per tensor, is competitive with SmoothQuant. Note that SmoothQuant uses calibration data to compute scales, whereas our method does not require data.

References

P. W. Anderson. More is different: Broken symmetry and the nature of the hierarchical structure of science. *Science*, 177(4047):393–396, 1972.

O. Andersson. *Sequential Good-Turing and the missing species problem*. PhD thesis, PhD thesis, 2022.

R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, et al. Extracting training data from large language models. In *Proceedings of the 30th USENIX Security Symposium*, pages 2633–2650, 2021.

W. A. Gale and G. Sampson. Good-turing frequency estimation without tears. *Journal of quantitative linguistics*, 2(3):217–237, 1995.

I. J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4):237–264, 1953.

M. D. Hoffman, D. Phan, david dohan, S. Douglas, T. A. Le, A. T. Parisi, P. Sountsov, C. Sutton, S. Vikram, and R. A. Saurous. Training chain-of-thought via latent-variable inference. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=a147pIS2Co.

J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

References

O. Kovaleva, S. Kulshreshtha, A. Rogers, and A. Rumshisky. BERT busters: Outlier dimensions that disrupt transformers. In C. Zong, F. Xia, W. Li, and R. Navigli, editors, *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3392–3405, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.300. URL https://aclanthology.org/2021.findings-acl.300/.

M. Nasr, N. Carlini, J. Hayase, M. Jagielski, A. F. Cooper, D. Ippolito, C. A. Choquette-Choo, E. Wallace, F. Tramèr, and K. Lee. Scalable extraction of training data from (production) language models. *arXiv preprint arXiv:2311.17035*, 2023.

T. Porian, M. Wortsman, J. Jitsev, L. Schmidt, and Y. Carmon. Resolving discrepancies in compute-optimal scaling of language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=4fSSqpk1sM.

R. Schaeffer, B. Miranda, and S. Koyejo. Are emergent abilities of large language models a mirage? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=ITw9edRDlD.

M. Sun, X. Chen, J. Z. Kolter, and Z. Liu. Massive activations in large language models. *arXiv preprint arXiv:2402.17762*, 2024.

J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, et al. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022.

References

G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NG7sS51zVF.

M. Yu, D. Wang, Q. Shan, and A. Wan. The super weight in large language models. *arXiv preprint arXiv:2411.07191*, 2024.