Towards a Universal Scaling Law of Training and Inference: Opinions on LLM Intelligence

Chuhan Wu
LLM Researcher

Ruiming Tang
LLM Researcher

Abstract

Guided by the prophecy of scaling law, large language models (LLMs) demonstrate higher levels of intelligence with increased sizes and computational power. Meanwhile, the overall outcome of small LLMs seems to show a scaling trend when a higher inference cost is paid in prompting and sampling. However, the inherent relatedness between training and inference in the path of scaling up is less studied. In this article, we present a universal theory on the joint computational scaling of LLM training and inference, which characterizes the general behaviors of LLMs in various settings. Based on simple modeling of several key hyperparameters, we give intuitive explanations for the effectiveness of various techniques at both training and inference time. We hope this work can provide insights for LLM research, development, and applications.

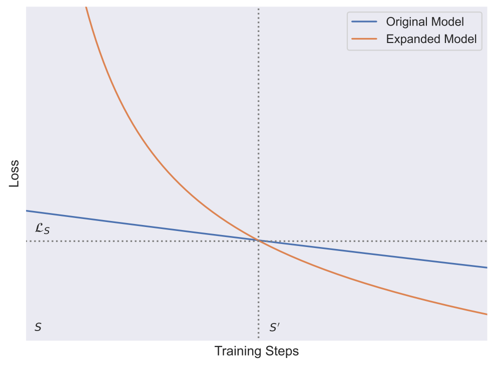
1 Introduction

On the upscaling path of computation power given by the scaling law [9, 7], the general capabilities of large language models (LLMs) have become more impressive and appealing [6]. Training larger models on more tokens has allowed LLMs to develop abilities beyond mere data memorization, such as solving math problems and demonstrating logical reasoning [1]. However, researchers face significant challenges, including limitations in high-quality data, computational resources, and the instability of training large models [15, 14, 20, 19]. These issues may restrict the upscaling potential of ultra-large language models.

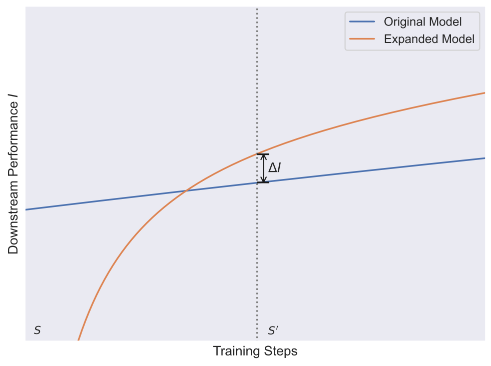
Faced with the difficulties in upscaling LLM training, an alternative way has emerged, i.e., investing more in the inference stage. Techniques such as chain-of-thought (CoT) prompting [18] and its successors [23, 3] using advanced search and prompting techniques have shown that LLMs can perform better in real-world applications by optimizing inference strategies. Based on the successful practice of existing works, an inference scaling law of LLMs is primarily formed, suggesting the scaling potential at test time [5, 13]. In particular in domains such as mathematics and code, smaller models can outperform larger ones given adequate trials [2, 4]. The release of GPT-o1[1] is also a successful signal to create the second scaling curve at the test time. This shift highlights the practicality of turning more computation into model inference under the stagnated growth of the training cost budget. This naturally leads to an interesting question: is there a unified scaling law that connects both training and inference?

Here, we provide an initial answer to this question with a qualitative theory that connects the joint effects of training and inference. We propose a unified scaling formulation to describe the joint effects of training and inference on the final outcome of LLMs in downstream tasks. We argue that both model size and model depth have major impacts on downstream performance, meanwhile, proper sampling and prompting strategies are also important in achieving optimal results. Our qualitative analysis can provide rational explanations for various techniques and phenomena in the practical use and development of LLMs. It is expected to offer new insights on optimizing the design of LLM architectures under limited training budgets to boost the scaling effects at test time.

[1]https://openai.com/index/introducing-openai-o1-preview/



(a) Training loss.



(b) Downstream performance

Figure 1: An illustrative example of the training loss and downstream performance of a base model and an expanded version from the base model at the S-th step. Although the two models have the same loss at step S', the expanded model has slightly better performance in downstream tasks.

2 Universal Scaling Law

2.1 Preliminary: Is Model Expansion a "Free Lunch"?

We start our deduction from a practical problem in LLM development, i.e., model expansion. As shown in Fig. 1, we expand a model from a basic model at step S through self-merging [22] techniques by copying a certain number of layers. After training on a relatively small amount of data ($\tilde{1}$00B tokens), the expanded model achieves a lower loss than the original model trained on the same data. However, we observe that the performance of the expanded model in downstream tasks is slightly better than that of the original model, although they have the same scale of loss at step S'. This phenomenon is difficult to explain by the scaling law, as it seems almost like a free lunch of LLM training. Therefore, a question arises: What factor leads to this surprising performance improvement?

In fact, the main difference between the two models at step S' is that the expanded model pays more cost at the inference stage, since it has more layers. Thus, we can assert that there exists a joint scaling effect on downstream performance that depends on the settings of both training and inference. In this unified scaling law, the depth of LLMs may generate a critical impact on the final performance.

2.2 Formulation of Universal Scaling Law

Motivated by the standard LLM training scaling law, we propose a unified scaling law that synthesizes the effects of both training and inference techniques. The vanilla scaling law is mainly based on the size of the training data and the size of the model, where the influence of the model shape is neglected. In fact, previous studies [8, 21] have demonstrated the importance of model depth in performing complex tasks. Thus, we reformulate the standard scaling law into a finer-grained one that considers the number of layers N and the hidden dimension d. It predicts the loss function $\mathcal{L}$ as follows:

$$\mathcal{L} = \frac{x_1}{N^{w_1}} + \frac{x_2}{d^{w_2}} + \frac{x_3}{S^{w_3}} + b, \tag{1}$$

where $w_{\{1,2,3\}}$, $x_{\{1,2,3\}}$ and b are parameters, and S reflects the training tokens. For mixture-of-expert (MoE) models, inspired by [10] we further modify this loss prediction by introducing an additional coefficient that reflects the expansion of model parameters:

$$\mathcal{L} = (\frac{x_1}{N^{w_1}} + \frac{x_2}{d^{w_2}})\frac{x_4}{E^{w_4}} + \frac{x_3}{S^{w_3}} + b, \tag{2}$$

where w_4 and x_4 are parameters, and E is the expansion factor that depends on the design of MoE architecture (e.g., number of experts and expert granularity).

In downstream tasks, the probability p of generating correct responses is determined by the corresponding loss, i.e., $p = e^{-\mathcal{L}}$ (we assume that the final answer is a single token). Denote the logits of the i-th token as μ_i (the correct token logits is μ_j), then the correct probability is computed below:

$$p = \frac{\exp(\mu_j)}{\sum_{i=1}^{V} \exp(\mu_i)} = \frac{1}{1 + \exp(-\mu)}, \tag{3}$$

where V is the vocabulary size and we use a single term $\exp(-\mu)$ to replace the accumulated term for simplicity. Thus, the relatedness between loss and the equivalent "logits" of the correct token is:

$$\mu = -\log(e^{\mathcal{L}} - 1). \tag{4}$$

In the non-CoT mode, the response accuracy I is given by p since it is already the expectation over data distribution. However, for more complicated decoding and sampling scenarios, we assume that the equivalent correct token logits are sampled from a Gaussian distribution $y \sim \mathcal{N}(\mu, \sigma^2)$, where σ depends on the data distribution and decoding strategies. Based on this assumption, we can model the behaviors of LLMs in CoT prompting strategies. Since most tokens in the thinking steps are less informative, we assume that the entire generation process has $C \geq 2$ critical steps, and the final response is corrected only when all critical steps are correct (here we ignore cases where wrong steps lead to correct answers). In CoT generation, the depth of reasoning inherent in LLMs is determined by the number of critical steps C and the model depth N. We use the following equation to adjust the sampled equivalent logits y by introducing an additional term that characterizes the effects of CoT:

$$\begin{aligned} y' &= y + y_0 \cdot \tanh(\frac{C \log(N)}{C_0}) \\ &= y + y_0 \cdot \frac{N^{\frac{2C}{C_0}} - 1}{N^{\frac{2C}{C_0}} + 1} \end{aligned} \tag{5}$$

where y_0 and C_0 are task-dependent constants, where a larger y_0 value means the task can benefit more from CoT prompting, while a larger C_0 indicates more complex reasoning steps. In general, more difficult tasks usually produce smaller coefficients y_0 and larger C_0. In addition, deeper models and more CoT steps usually bring a higher increase in the probability of generating correct tokens, whereas this benefit is not unlimited when we increase the reasoning step and model depth. The overall accuracy expectation is computed as follows:

$$I = \mathbb{E}_{y'}(\frac{1}{1 + e^{-y'}})^C. \tag{6}$$

Repeated sampling without verification. Based on the above modeling, we further discuss scenarios with repeated sampling in the inference phase. If the system only conducts repeated inference, its main effect can be regarded as reducing the variance of equivalent logits across data distributions:

$$y \sim \mathcal{N}(\mu, \frac{\sigma^2}{f(R)}), \tag{7}$$

where R is the number of trials, $f(\cdot)$ is usually a sublinear function since different trials may not be independent. This formula implies that repeated sampling techniques such as self-consistency [16] may improve the performance of LLMs to some extent under certain scenarios, but the improvement is not unlimited and does not show a scaling effect.

Repeated sampling with perfect verification.

In some domains, such as mathematics and code, we have gold standards to verify the correctness of generated responses. For example, on the HumanEval benchmark, we can run the generated code to check whether it passes all the test cases. Repeated sampling may show a scaling effect under this setting, where the final accuracy I' is formulated as follows:

$$I' = 1 - (1 - I)^{f(R)}. \tag{8}$$

We see that all test cases can be correctly solved under sufficient trials regardless of their difficulties, as shown by the infinite monkey theorem.

Repeated sampling with imperfect verification.

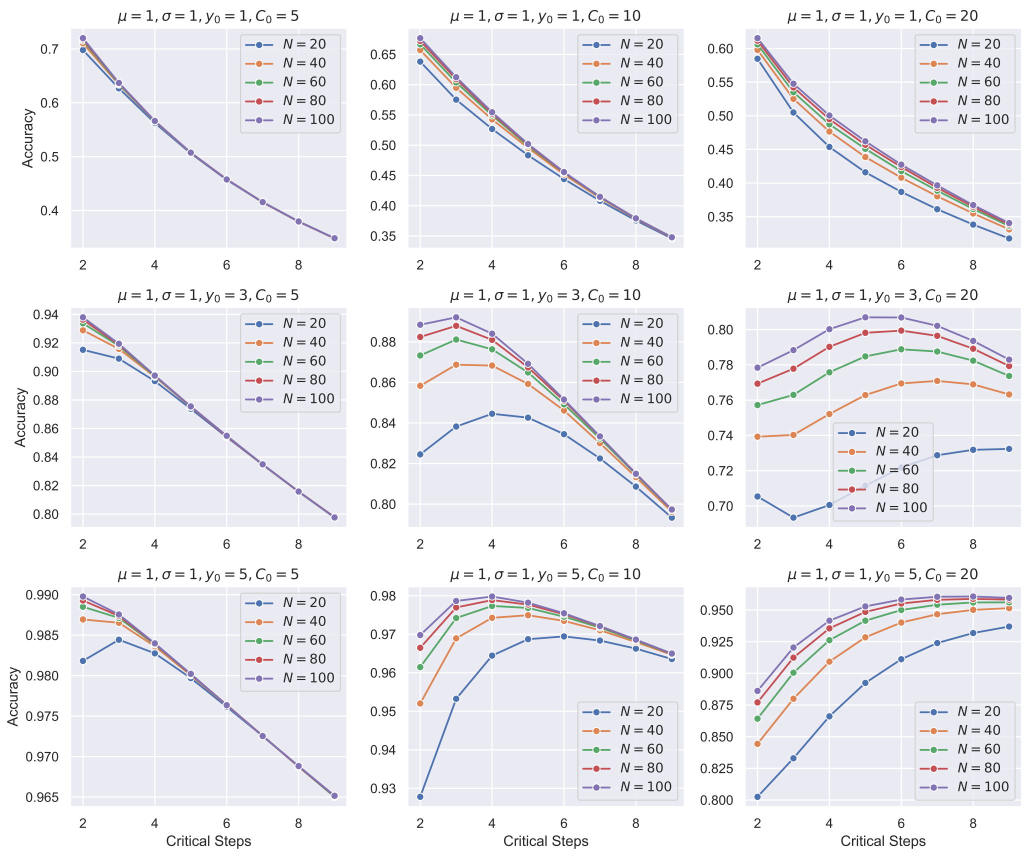


Figure 2: The answer accuracy w.r.t. different task constants, model depths, and critical steps.

In most scenarios such as RLHF [11] and LLM-as-judge [25, 24], LLMs generate multiple responses and evaluate them with an imperfect judger. We assume that the judge has an ϵ probability of mistakenly rejecting a correct response. In this setting the final accuracy I' is computed as follows:

$$I' = (1 - (1 - I)^{f(R)})(1 - \epsilon)^{f(R)}. \tag{9}$$

This equation indicates that too many trials may be suboptimal for the final performance since the judger system may mislead the model to pick the wrong results.

3 Analysis and Implications

Here we use our qualitative model to explain the phenomena in practical LLM usage.

3.1 Effects of CoT in Different Types of Tasks

We vary the value of the task-dependent constants y_0 and C_0 to see the accuracy change w.r.t. different numbers of critical steps and model depths, as shown in Fig. 2 (we set $\mu = \sigma = 1$ here). We have several implications from the results:

- Small y_0, large C_0 (upper right figure): tasks with high reasoning difficulty and large exploration space in the reasoning step. MMLU is a representative benchmark of this type, where the CoT mode may not be better than the non-CoT mode in a few-shot setting. Solving these tasks usually requires sufficient knowledge storage, and step-by-step reasoning does not help much. Model depth is quite important in solving MMLU-like problems [21].

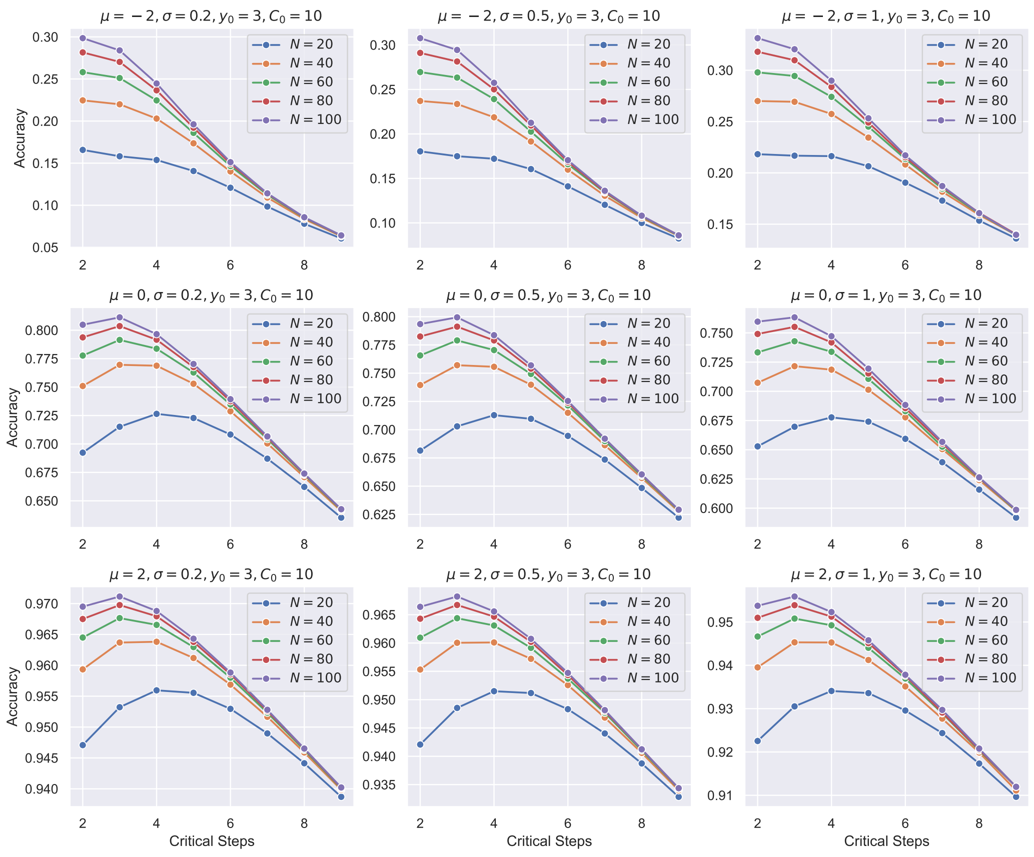


Figure 3: Influence of μ and σ on the effectiveness of CoT reasoning.

- Large y_0, large C_0 (lower right figure): tasks with a complex combination of multiple easy steps. GSM8K can be regarded as this type, where the performance can substantially benefit from using more layers and finer-grained reasoning chains.

- Small y_0, small C_0 (upper left figure): tasks with high reasoning difficulties but limited steps. Riddle solving is a task of this type, where step-by-step reasoning does not help, and strong creative thinking ability is needed.

- Small y_0, small C_0 (lower left figure): easy tasks such as simple arithmetic. Only a few steps are sufficient to give accurate results, while too long reasoning steps may be confusing.

Based on these discussions, we can see that our formulation can explain the behaviors of models in various types of tasks.

3.2 Influence of Training Loss and Repeated Sampling

In our qualitative modeling, the equivalent correct token logits are determined by the training loss. Thus, we can vary the value of μ and σ to simulate the performance of models with different training losses and different inference trials (Fig. 3). Based on our modeling, LLMs may not benefit from CoT reasoning if they have a high loss in a given task. We also observe this phenomenon in our practice on very difficult benchmarks like MMLU-Pro [17] and GPQA [12]. Combining responses in different trials fails to solve problems better than using fewer chances in this situation, which has been verified by existing work [5]. In contrast, voting strategies are more effective in tasks when LLMs have a good mastery of task-related knowledge, which is consistent with our intuition.

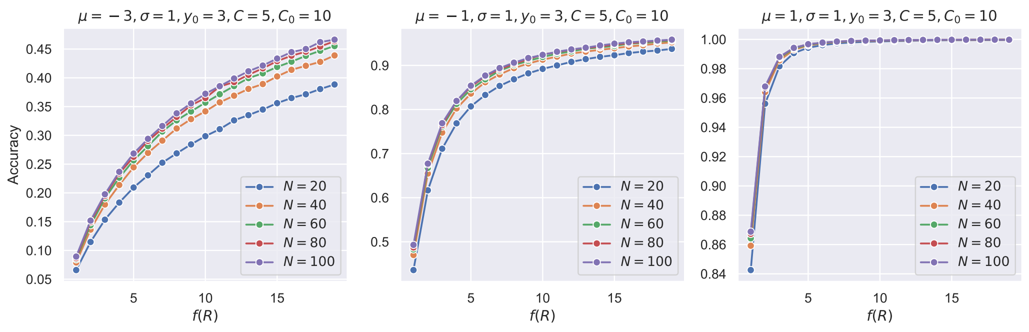


Figure 4: Model performance with perfect verification under different μ and $f(R)$.

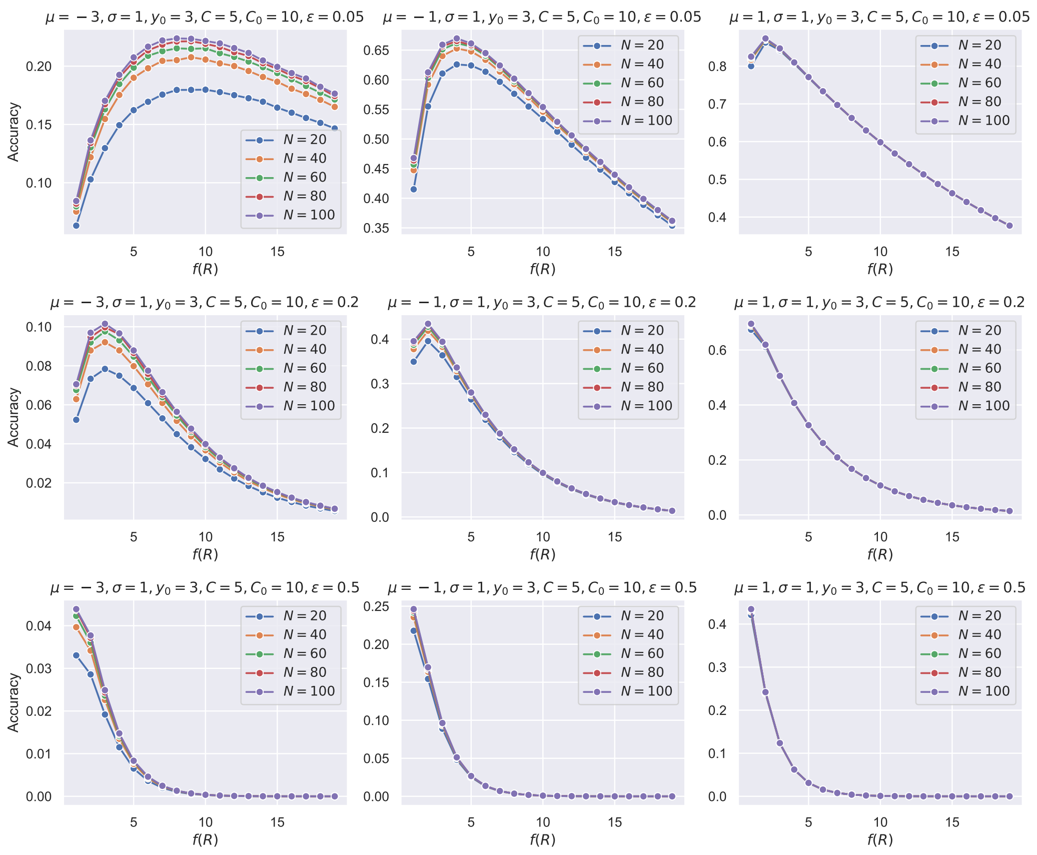


Figure 5: Influence of the verification error on the final performance.

3.3 Influence of Response Verification

Finally, we discuss the effectiveness of response verification in different settings. As shown in Fig. 4, the performance is consistently better when $f(R)$ is larger, which is intuitive because even monkeys can almost surely type any text given an infinite time. A more practical simulation with imperfect verification is shown in Fig. 5. When the judger is relatively reliable, using complicated sampling techniques at the inference stage may achieve substantial performance gains, especially when the model is weak on the given task. However, it is ineffective to scale up the inference cost if the judger is inaccurate, which is consistent with our intuition that low-quality verification will mislead the system to pick wrong or inaccurate results. This can explain why the new GPT-o1 system does not improve impressively in subjective tasks.

4 Conclusion

In this paper, we present a unified scaling law that connects the impact of model architecture, data size, and inference strategies on the final model performance. We show that the depth of LLMs is a critical factor in training and inference that generates substantial impacts on overall performance. By setting different parameters in our qualitative model, we can explain various phenomena in practical LLM development and applications. We hope that our modeling can offer useful insight to help optimize resource allocation and strategies in the unified scaling path of LLM training and inference.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023).

[2] Hritik Bansal, Arian Hosseini, Rishabh Agarwal, Vinh Q Tran, and Mehran Kazemi. 2024. Smaller, Weaker, Yet Better: Training LLM Reasoners via Compute-Optimal Sampling. arXiv preprint arXiv:2408.16737 (2024).

[3] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In AAAI, Vol. 38. 17682–17690.

[4] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. arXiv preprint arXiv:2407.21787 (2024).

[5] Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. 2024. Are more llm calls all you need? towards scaling laws of compound inference systems. arXiv preprint arXiv:2403.02419 (2024).

[6] Zhengxiao Du, Aohan Zeng, Yuxiao Dong, and Jie Tang. 2024. Understanding emergent abilities of language models from the loss perspective. arXiv preprint arXiv:2403.15796 (2024).

[7] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. In NeurIPS. 30016–30030.

[8] Mingyu Jin, Qinkai Yu, Jingyuan Huang, Qingcheng Zeng, Zhenting Wang, Wenyue Hua, Haiyan Zhao, Kai Mei, Yanda Meng, Kaize Ding, et al. 2024. Exploring Concept Depth: How Large Language Models Acquire Knowledge at Different Layers? arXiv preprint arXiv:2404.07066 (2024).

[9] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. arXiv preprint arXiv:2001.08361 (2020).

[10] Jakub Krajewski, Jan Ludziejewski, Kamil Adamczewski, Maciej Pióro, Michał Krutul, Szymon Antoniak, Kamil Ciebiera, Krystian Król, Tomasz Odrzygóźdź, Piotr Sankowski, et al. 2024. Scaling laws for fine-grained mixture of experts. arXiv preprint arXiv:2402.07871 (2024).

[11] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. NeurIPS 35 (2022), 27730–27744.

[12] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. arXiv preprint arXiv:2311.12022 (2023).

[13] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. arXiv preprint arXiv:2408.03314 (2024).

[14] Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. 2023. Spike No More: Stabilizing the Pre-training of Large Language Models. arXiv preprint arXiv:2312.16903 (2023).

[15] Pablo Villalobos, Jaime Sevilla, Lennart Heim, Tamay Besiroglu, Marius Hobbhahn, and Anson Ho. 2022. Will we run out of data? an analysis of the limits of scaling datasets in machine learning. arXiv preprint arXiv:2211.04325 (2022).

[16] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. arXiv preprint arXiv:2203.11171 (2022).

[17] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. arXiv preprint arXiv:2406.01574 (2024).

[18] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. NeurIPS 35 (2022), 24824–24837.

[19] Sarah Wild. 2024. Millions of research papers at risk of disappearing from the Internet. Nature 627, 8003 (2024), 256–256.

[20] Mitchell Wortsman, Peter J Liu, Lechao Xiao, Katie Everett, Alex Alemi, Ben Adlam, John D Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, et al. 2023. Small-scale proxies for large-scale transformer training instabilities. arXiv preprint arXiv:2309.14322 (2023).

[21] Chuhan Wu and Ruiming Tang. 2024. Performance Law of Large Language Models. arXiv preprint arXiv:2408.09895 (2024).

[22] Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. 2024. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. arXiv preprint arXiv:2408.07666 (2024).

[23] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. NeurIPS 36 (2024).

[24] Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. arXiv preprint arXiv:2401.10020 (2024).

[25] Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2024. Generative Verifiers: Reward Modeling as Next-Token Prediction. arXiv preprint arXiv:2408.15240 (2024).