Universidade de São Paulo
Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto
Departamento de Computação e Matemática

Comparing the LMC Complexity of Neural Networks with their Inference Capability

Author: Lucas Miranda Mendonça Rezende
Supervisor: Ph.D. Luiz Otavio Murta Junior

October 28, 2025

Abstract

TO DO

Contents

1 Introduction

Since the creation of Transformers in 2017 [1] and the subsequent usage of this new technique in the training of Large Language Models (LLMs), there has been a gold rush within the machine learning world. GPT-3.5, the original ChatGPT model, rapidly gained widespread adoption, becoming the fastest-growing consumer application in history after its launch in 2022 [2], sparking further interest in researchers, investors, and the general public for more powerful and cost-efficient models.

Since then, multiple new models have been developed by the biggest technology companies in the world such as Google, Microsoft, NVIDIA, Amazon, and amazingly also by some smaller ones such as DeepSeek. As better models in almost every metric emerged, it also became increasingly obvious that all this improvement was not for free: billions were spent in larger datacenters, predictions that we soon would not have enough data on the internet to keep building bigger models, increasingly expensive for marginal performance gains. But why?

In 2020, before the launch of the original ChatGPT, researchers at OpenAI [3] found that "performance improves smoothly as we increase the model size N (the number of parameters excluding embeddings), dataset size D, and amount of compute C" and that "performance depends strongly on scale and weakly on model shape, such as depth vs. width". Second, it was also observed that different architectures could impact training performance, e.g., Transformers would have lower test loss than LSTMs. Those statements became the basis for the scaling "laws" we currently accept.

It is evident that the most straightforward way to get a better model, at least considering performance in terms of lower test loss, is to increase the scale (N, C, and D sizes). However, there is a huge drawback: the scaling laws described in this case are power laws, meaning we would need an exponential amount of resources to achieve a constant gain in performance. The second approach would be creating better architectures or improving the existing ones, which demands research and is generally not as simple as increasing a number.

The first approach, being the easiest, was explored by the companies creating the new models, pushing those three variables to ludicrous amounts. The GPT family, mentioned above, serves as a good example: the original GPT had 117 million parameters, GPT-2 had 1.5 billion (a 12x increase from GPT), and GPT-3 had 175 billion (a 117x increase from GPT-2) [4] [5] [6]. It is noticeable that, just a few years later, the industry is already showing signs of exhaustion: new models do not exhibit performance improvements as dramatic as those observed in the recent past, although investment in training infrastructure has never been higher.

The second approach is what we intend to contribute in this work: an improvement in the architecture itself. Of course, before creating a new revolutionary architecture it would be useful to understand how machines learn. In a typical analysis setting, knowing how the process works is a requirement to engineer a better version of it. In Machine Learning, however, the process of learning, which is somehow connected to the well-known process of training, is still far from being fully understood.

Discussing the understanding of the learning process is out of the scope of this work. Yet, we are going to analyze what may be a piece of the puzzle: the LMC statistical complexity [7], a metric that appears to be related to the model's performance and might help in understanding the behavior of such by providing insights about the weights distribution.

1.1 Work thesis

The main hypothesis of this work comes from Professor Luiz Otavio Murta Junior [8] and can be summarized as: **"There exists a relationship between model complexity and its inference capability"**.

1.2 Objective

Validating the work thesis means we would have a new way of indirectly assessing a model's performance by just looking at the distribution of its weights, opening the door to new optimization processes during training, potentially reducing the amount of compute necessary to reach the best performance or even discovering new maxima. Also, having a validated weights distribution vs. performance comparison should improve the data richness when studying a model's learning process in future studies. Thus, we can establish the following objectives for this work:

- Validate the existence of a meaningful relationship between complexity of neural network weights and their inference performance.
- If possible, find the mathematical relation between those measures.
- Explore other dimensions of the problem that can affect complexity and performance measures such as parameter count.

2 Methodology

2.1 Testing Environment

We used the following computational environment for our experiments:

- OS: Linux 6.8.0-64-generic
- CPU: Intel Xeon Gold 6130 2.10GHz - 32 cores / 64 threads
- GPU: NVIDIA Quadro P5000 - 16GB VRAM
- RAM: 512GB
- DISK: 7.6TB

The hardware was provided for research purposes by Universidade de São Paulo. The choice of this environment was motivated by the availability of a high-performance GPU and large quantities of RAM necessary for handling large language models.

We used the following tools and libraries:

- Python 3.12.3
- PyTorch 2.8.0 [9]
- Transformers 4.56.2 [10]
- Hugging Face Hub 0.35.0 [11]
- Numpy 2.2.6
- Pandas 2.3.2
- Pysr 1.5.9
- SciPy 1.16.1
- Matplotlib 3.10.6
- Seaborn 0.13.2 [12]
- CUDA Toolkit 12.6

The choice of versions was motivated by compatibility with our computational environment GPU.

2.2 Model Selection

Due to the wide availability of open models combined with the ease of accessing them through the Transformers library [10] and Hub library [11], **Hugging Face** was chosen as the source for model selection in this study.

Hugging Face is a platform that hosts a variety of machine learning models, datasets, and tools. It is widely used in the AI research community for sharing and collaborating on machine learning projects [13, 14]. Other platforms such as Ollama [15] were considered, but ultimately not chosen due to the limited number of models and lack of company diversity.

Model selection proceeded in two stages:

1. First, we identified major technology companies by market capitalization that publish openly released language models. The companies considered were **OpenAI**, **Google**, **Meta**, and **Microsoft**.

2. Second, for each company we compiled a candidate set consisting of every model that satisfied the following criteria:

    - Must be available on the Hugging Face platform on the official company account.
    - The model is a transformer-based language model.
    - Model weights are publicly accessible (open weights), including models released behind gated access.
    - The model is text-only (no multimodal image/audio inputs).
    - The model is an original base model rather than a task-specific fine-tuned variant.
    - The total parameter count is below 150 billion. This upper bound was imposed due to hardware and inference limitations in the computational environment used for our experiments.
    - The model is supported by the Hugging Face Transformers library [10] (i.e., it can be instantiated via the AutoModel utility), which ensures consistent loading and preprocessing across the candidate set.
    - There is at least one publicly available benchmark result for the model among the selected benchmarks (section 2.4.1).

The final selection of models used in this study is listed in Table 1.

Meta	Google	Microsoft	OpenAI
meta-llama/Llama-4-Scout-17B-16E	google/gemma-3-27b-pt	microsoft/Phi-4-mini-reasoning	openai/gpt-oss-120b
meta-llama/Llama-3.2-3B	google/gemma-3-12b-pt	microsoft/Phi-4-reasoning	openai/gpt-oss-20b
meta-llama/Llama-3.2-1B	google/gemma-3-4b-pt	microsoft/Phi-4-reasoning-plus	openai-community/gpt2-xl
meta-llama/Llama-3.1-70B	google/gemma-3-1b-pt	microsoft/phi-4	openai-community/gpt2-large
meta-llama/Llama-3.1-8B	google/gemma-3-270m	microsoft/phi-2	openai-community/gpt2-medium
meta-llama/Meta-Llama-3-70B	google/gemma-2-27b	microsoft/phi-1_5	openai-community/gpt2
meta-llama/Meta-Llama-3-8B	google/gemma-2-9b	microsoft/phi-1	
meta-llama/Llama-2-70b-hf	google/gemma-2-2b		
meta-llama/Llama-2-13b-hf	google/gemma-7b		
meta-llama/Llama-2-7b-hf	google/gemma-2b		
	google/recurrentgemma-9b		
	google/recurrentgemma-2b		

Table 1: Selected language models included in this study.

Meta Models: [16] [17] [18] [19] [20] [21] [22] [23] [24] [25]
Google Models: [26] [27] [28] [29] [30] [31] [32] [33] [34] [35] [36]
Microsoft Models: [37] [38] [39] [40] [41] [42] [43]
OpenAI Models: [44] [45] [46] [47] [48] [49]

2.3 LMC Complexity

According to [7], LMC Statistical Complexity is the product of two other measures: Disequilibrium and Shannon entropy. It captures both the structured and unstructured aspects of the distribution:

$$C_{LMC} = H \times D$$

Disequilibrium measures how far a probability distribution is from being uniform, quantifying the "order" or structure in the data. It is calculated as:

$$D = \sum_{i=1}^{n} \left(p_i - \frac{1}{n} \right)^2$$

Shannon entropy measures the amount of uncertainty or randomness in a probability distribution. The normalized Shannon entropy is given by:

$$H = -K \sum_{i=1}^{n} p_i \log p_i$$

The values p_i represent the probabilities associated with each state i in the distribution, and n is the total number of states. K is a positive constant and, in our case, is set to 1 for simplicity. K can be changed later since $C_{LMC} = (-K \sum_{i=1}^{n} p_i \log p_i) \times D$ is equivalent to $C_{LMC} = K \times (-\sum_{i=1}^{n} p_i \log p_i) \times D$.

2.3.1 Reading Model Weights

Model weights are fetched using the Hugging Face Hub library [11] and loaded using the Transformers library [10]. For each selected model, we instantiate the model using the **AutoModel** utility, which automatically handles model architecture loading and weight initialization.

The models are loaded entirely into the main memory instead of a GPU since the amount of VRAM available in the computational environment is insufficient for larger models ($>$ 70 billion parameters). Other problems such as handling symbolic tensors also motivated this decision.

Using the main memory is considerably slower than using a GPU, but allows us to work with larger models without running into memory limitations. Since we had a large amount of RAM, we decided to cast all the model weights to float32 during the AutoModel instantiation as CPUs generally handle this format natively and, as a consequence, make calculations faster.

Once loaded, we extract all the parameters from the model using the `named_parameter()` method provided by the Transformers library. This method returns an iterator over all model parameters, each parameter being represented as a tensor.

We then flatten each tensor into a one-dimensional array and store it in a list. Each array is labeled in one of the following **weight-type categories**:

- Bias: if 'bias' is in the parameter name
- Norm: if 'norm' is in the parameter name
- Embedding: if 'embed' is in the parameter name
- Other: all other weights

These categories are the most common types of weights found in this architecture and will be used later as another dimension of analysis.

The choice of also analyzing by weight type is motivated by the fact that different weight types may exhibit different statistical properties and, as a consequence, different complexity characteristics. Studies such as the already cited OpenAI's [3] take into account different weight types when analyzing scaling laws.

2.3.2 Filtering

Casting different encoding formats to float32 may introduce rounding errors that manifest as extreme outliers in the weight distribution. This is rare (approximately 10 to 30 values in 100 billion) but can be problematic for our next step: histogram construction (section 2.3.3), since the number of bins will be affected by the range of values in the data. The reasons for those rounding errors could not be fully investigated within the scope of this work, but they are likely related to floating point rounding errors.

To mitigate the impact of outliers, we will apply a value removal approach. This process filters the data to retain only values within a configurable range centered around the mean:

$$\text{lower bound} = \mu - \sigma_{\text{filter}} \cdot \sigma$$
$$\text{upper bound} = \mu + \sigma_{\text{filter}} \cdot \sigma$$

where μ is the data mean, σ is the data standard deviation, and σ_{filter} is a configurable parameter that controls the filtering strength. Values falling outside this range are excluded from further analysis.

It is not trivial to choose the best value for σ_{filter}. A very low value may remove important parts of the distribution, while a very high value may not effectively mitigate the outlier problem. For this reason, we will experiment with different values of σ_{filter} and analyze how they affect the final complexity results. As a consequence, this becomes yet another dimension of analysis in our study.

The values of σ_{filter} tested will be: **0.125**, **0.25**, **0.5**, **1**, **2**, **3**, **4**, **5**, **10**, **20**. The choice of σ_{filter} that will be used as the data's filtering option for the next steps is the one that:

1. most reduces the bin count compared to the unfiltered data;
2. has the least amount of filtering or, in other words, the highest value of σ_{filter} among the ones chosen in the first criterion.

2.3.3 Data Discretization and Histogram

To compute the LMC complexity of a finite array of floating-point numbers, we first construct a histogram to discretize the data into a probability distribution. That's justified as the chance of finding two exact numbers is extremely low and, as a consequence, it is hard to determine the probability of each value.

We will call the set of all data points as S, the total amount of data points as N and the number of bins they will be distributed into as n. The probabilities p_i are then calculated as $p_i = \frac{f_i}{N}$ where f_i is the frequency count of data points in bin i.

As expected, this approach revisits a classic issue in histogram-based analysis: the choice of the number of bins n will impact the resulting probability distribution, which is specially problematic to the LMC complexity measure; variations in n can cause significant fluctuations in the final result. Selecting an inappropriate amount may produce misleading values, either by oversimplifying the distribution (too few bins) or by introducing noise (too many bins).

There are a variety of methods to determine n in a histogram, most of them with their own advantages and disadvantages. Commonly used methods such as **Sturges' formula** [50] and **Rice Rule** [51] rely only on the number of data points, while others like **Scott's normal reference rule** and **Freedman-Diaconis' choice** also take into account the data distribution by using standard deviation and interquartile range, respectively [52].

We chose to use the **Freedman-Diaconis' choice** as it adapts better to our needs. This is justified since N often consists of billions of numbers, and the distribution, although mostly concentrated between -1 and 1, can become sparse due to outliers and, as a consequence, require a larger number of bins to capture its characteristics accurately. The Freedman-Diaconis rule helps mitigate the influence of outliers by using the interquartile range IQR to determine bin width h. The rule is defined as follows [53]:

$$h = \frac{2 \times IQR}{N^{1/3}}$$

where IQR is the interquartile range of the data which is calculated as $Q3 - Q1$, $Q3$ and $Q1$ are the values at the 75th and 25th percentiles in the data, respectively. Since N is very large, the percentiles are computed based on a random sample of

100000 data points to reduce computational cost. The sample size was determined empirically to be large enough to provide stable estimates of the percentiles, the final number of bins showed no variance between tests.

The number of bins n can then be calculated as:

$$n = \frac{\max(S) - \min(S)}{h}$$

Finally, we can use the already filtered data and the calculated n to build a histogram using a simple function provided by PyTorch [9]. Then, compute the probabilities p_i for each bin i.

2.3.4 Complexity Calculation

With the probabilities p_i computed from the histogram, we can now calculate the LMC complexity C_{LMC} using the formulas provided in the beginning of section 2.3. This involves calculating the Disequilibrium D and the Shannon entropy H using the probabilities, and then multiplying them to obtain the final complexity measure C_{LMC}.

2.4 Inference Capability

Often called model performance, inference capability refers to how well a trained neural network performs on unseen data. This is typically measured using various metrics depending on the specific task the model is designed for.

In the previously cited research paper by OpenAI [3], performance was associated with test cross-entropy loss. This metric quantifies the difference between the predicted probability distribution output by the model and the true distribution of the target labels. Lower cross-entropy loss values indicate better model performance, as they reflect a closer alignment between predictions and actual outcomes.

Unfortunately, not all models we intend to analyze have publicly available training and test data. It's also not possible to run models in a training setting due to performance limitations of the computational environment available. Therefore, we will have to rely on imperfect proxies for performance such as **benchmarks**.

2.4.1 Benchmark Selection

Benchmarks are standardized tests designed to evaluate the performance of machine learning models across various tasks. They provide a common ground for comparison by measuring how well different models perform on the same datasets using predefined metrics.

According to [54], benchmarks such as the famous MMLU, correlate fairly well with the predicted test loss determined by scaling laws, making them suitable proxies.

They are, however, not perfect. Some benchmarks may fail to capture all aspects of a model's capabilities, leading to an incomplete assessment of performance. Other problems such as Data Contamination [55] can influence the validity of results and make it hard to fairly compare models made in different times. It is also hard to find benchmarks that are widely reported for all models we intend to analyze.

A nice proposal for future research to avoid the problems cited above would involve training a model from scratch twice: optimizing it initially for minimal loss and then for minimal loss + maximum LMC complexity, then comparing the results. This is however out of the scope of this work.

The benchmark selection followed three main heuristics:

- **Relevance**: The benchmark should be widely recognized and accepted in the machine learning community, e.g., used in major research papers.
- **Generality**: It should cover a range of tasks and data types to provide an assessment of model performance across different scenarios, that is, not being specialized in one task.
- **Availability**: The benchmark results should be publicly available. Benchmarks that cover more of the selected models were preferred.

Based on those heuristics, the benchmarks selected were:

- **MMLU (5-shot)**: Massive Multitask Language Understanding, a benchmark that tests models across 57 tasks spanning various subjects and difficulty levels. Widely used to evaluate LLMs [56].
- **MMLU-Pro (5-shot)**: An enhanced version of MMLU that includes additional tasks and updated datasets to provide a better evaluation [57].
- **OpenLLM (Average)**: A benchmark suite that evaluates models on a variety of tasks, including language understanding, generation, and reasoning. It aggregates results from multiple datasets to provide an overall performance score [58].
- **LMArena (Score)**: An online platform where users can submit questions and receive responses from two anonymous large language models (LLMs),

then vote on which answer they prefer, helping to crowdsource human preferences for evaluating and ranking LLMs. Its evaluation works by collecting thousands of pairwise comparisons from users, using the Bradley-Terry system to estimate win rates and compute rankings/scores. [59].

2.4.2 Benchmark Collection Procedure

Benchmark values were manually collected from multiple sources, in the following order of priority:

1. Official *Hugging Face* model pages
2. Original research papers
3. Official websites
4. Third-party websites

Meta Models: [16], [17], [18], [19], [20], [21], [22], [23], [24], [25]
Google Models: [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36]
Microsoft Models: [37], [38], [39], [40], [41], [42], [43], [60], [61]
OpenAI Models: [44], [45], [46], [47], [48], [49], [62],
Additional References: [14], [15], [63], [64],

If more than one source was available for the same model and benchmark, the one with higher priority was chosen. Not all models had results available for all benchmarks. Figure 1 shows the availability matrix.

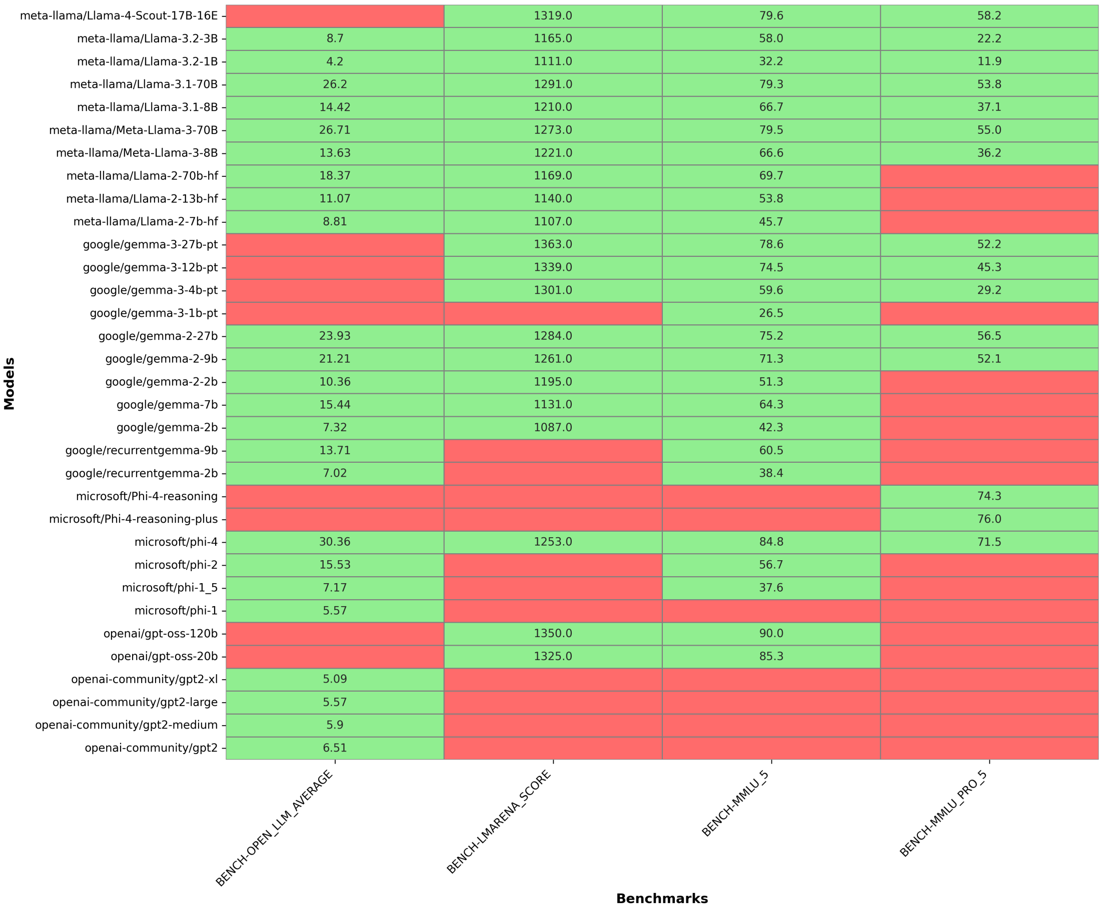

Figure 1: Availability of benchmark results for the selected models.

2.5 Comparing LMC Complexity and Inference Capability

3 Results

References

[1] A. Vaswani et al., "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[2] T. Trust, J. Whalen, and C. Mouza, "Editorial: Chatgpt: Challenges, opportunities, and implications for teacher education," *Contemporary Issues in Technology and Teacher Education*, vol. 23, no. 1, pp. 1–23, 2023. [Online]. Available: https://www.learntechlib.org/primary/p/222408/

[3] J. Kaplan et al., "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020. [Online]. Available: https://arxiv.org/abs/2001.08361

[4] T. B. Brown et al., "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[5] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019. [Online]. Available: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[6] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018. [Online]. Available: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

[7] R. Lopez-Ruiz, H. Mancini, and X. Calbet, "A statistical measure of complexity," *Physics Letters A*, vol. 209, no. 5-6, pp. 321–326, 1995.

[8] L. O. Murta, *Relationship between model complexity and inference capability*, Personal communication and ongoing research, 2025.

[9] A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," *arXiv preprint arXiv:1912.01703*, 2019. [Online]. Available: https://arxiv.org/abs/1912.01703

[10] T. Wolf et al., "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019. [Online]. Available: https://arxiv.org/abs/1910.03771

[11] H. Face, *Huggingface_hub: The hugging face hub python library*, https://huggingface.co/docs/huggingface_hub, 2021.

[12] M. L. Waskom, "Seaborn: Statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021. DOI: 10.21105/joss.03021 [Online]. Available: https://doi.org/10.21105/joss.03021

[13] U. R. Pol, P. S. Vadar, and T. T. Moharekar, *Hugging face revolutionizing ai and nlp*, https://www.researchgate.net/profile/Tejashree-Moharekar/publication/383497950, 2024.

[14] *Hugging face – the ai community building the future*, https://huggingface.co/, 2024.

[15] *Ollama – get up and running with large language models locally*, https://ollama.com/, 2024.

[16] *Meta llama 4 scout 17b 16e*, https://huggingface.co/meta-llama/Llama-4-Scout-17B-16E, 2024.

[17] *Meta llama 3.2 3b*, https://huggingface.co/meta-llama/Llama-3.2-3B, 2024.

[18] *Meta llama 3.2 1b*, https://huggingface.co/meta-llama/Llama-3.2-1B, 2024.

[19] *Meta llama 3.1 70b*, https://huggingface.co/meta-llama/Llama-3.1-70B, 2024.

[20] *Meta llama 3.1 8b*, https://huggingface.co/meta-llama/Llama-3.1-8B, 2024.

[21] *Meta llama 3 70b*, https://huggingface.co/meta-llama/Meta-Llama-3-70B, 2024.

[22] *Meta llama 3 8b*, https://huggingface.co/meta-llama/Meta-Llama-3-8B, 2024.

[23] *Meta llama 2 70b hf*, https://huggingface.co/meta-llama/Llama-2-70b-hf, 2024.

[24] *Meta llama 2 13b hf*, https://huggingface.co/meta-llama/Llama-2-13b-hf, 2024.

[25] *Meta llama 2 7b hf*, https://huggingface.co/meta-llama/Llama-2-7b-hf, 2024.

[26] *Google gemma 3 27b pt*, https://huggingface.co/google/gemma-3-27b-pt, 2024.

[27] *Google gemma 3 12b pt*, https://huggingface.co/google/gemma-3-12b-pt, 2024.

[28] *Google gemma 3 4b pt*, https://huggingface.co/google/gemma-3-4b-pt, 2024.

[29] *Google gemma 3 1b pt*, https://huggingface.co/google/gemma-3-1b-pt, 2024.

[30] *Google gemma 2 27b*, https://huggingface.co/google/gemma-2-27b, 2024.

[31] *Google gemma 2 9b*, https://huggingface.co/google/gemma-2-9b, 2024.

[32] *Google gemma 2 2b*, https://huggingface.co/google/gemma-2-2b, 2024.

[33] *Google gemma 7b*, https://huggingface.co/google/gemma-7b, 2024.

[34] *Google gemma 2b*, https://huggingface.co/google/gemma-2b, 2024.

[35] *Google recurrentgemma 9b*, https://huggingface.co/google/recurrentgemma-9b, 2024.

[36] *Google recurrentgemma 2b*, https://huggingface.co/google/recurrentgemma-2b, 2024.

[37] *Microsoft phi 4 mini reasoning*, https://huggingface.co/microsoft/Phi-4-mini-reasoning, 2024.

[38] *Microsoft phi 4 reasoning*, https://huggingface.co/microsoft/Phi-4-reasoning, 2024.

[39] *Microsoft phi 4 reasoning plus*, https://huggingface.co/microsoft/Phi-4-reasoning-plus, 2024.

[40] *Microsoft phi 4*, https://huggingface.co/microsoft/phi-4, 2024.

[41] *Microsoft phi 2*, https://huggingface.co/microsoft/phi-2, 2024.

[42] *Microsoft phi 1.5*, https://huggingface.co/microsoft/phi-1_5, 2024.

[43] *Microsoft phi 1*, https://huggingface.co/microsoft/phi-1, 2024.

[44] *Openai gpt oss 120b*, https://huggingface.co/openai/gpt-oss-120b, 2024.

[45] *Openai gpt oss 20b*, https://huggingface.co/openai/gpt-oss-20b, 2024.

[46] *Openai gpt-2 xl*, https://huggingface.co/openai-community/gpt2-xl, 2024.

[47] *Openai gpt-2 large*, https://huggingface.co/openai-community/gpt2-large, 2024.

[48] *Openai gpt-2 medium*, https://huggingface.co/openai-community/gpt2-medium, 2024.

[49] *Openai gpt-2*, https://huggingface.co/openai-community/gpt2, 2024.

[50] H. A. Sturges, "The choice of a class interval," *Journal of the American Statistical Association*, vol. 21, no. 153, pp. 65–66, 1926. [Online]. Available: https://www.jstor.org/stable/2965501

[51] J. Moral De La Rubia, "Rice university rule to determine the number of bins," *Open Journal of Statistics*, 2024. DOI: 10.4236/ojs.2024.141006 [Online]. Available: https://www.scirp.org/pdf/ojs_2024022914191399.pdf

[52] K. H. Knuth, "Optimal data-based binning for histograms," *arXiv preprint arXiv:physics/0605197*, 2006. [Online]. Available: https://arxiv.org/abs/physics/0605197

[53] D. Freedman and P. Diaconis, "On the histogram as a density estimator: L_2 theory," *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, vol. 57, pp. 453–476, 1981. [Online]. Available: https://link.springer.com/content/pdf/10.1007/BF01025868.pdf

[54] D. Owen, "How predictable is language model benchmark performance?" *arXiv preprint arXiv:2401.04757*, 2024. [Online]. Available: https://arxiv.org/abs/2401.04757

[55] I. Magar and R. Schwartz, "Data contamination: From memorization to exploitation," *arXiv preprint arXiv:2203.08242*, 2022. [Online]. Available: https://arxiv.org/abs/2203.08242

[56] D. Hendrycks et al., "Measuring massive multitask language understanding," *arXiv preprint arXiv:2009.03300*, 2020. [Online]. Available: https://arxiv.org/abs/2009.03300

[57] Y. Wang et al., "Mmlu-pro: A more robust and challenging multi-task language understanding benchmark," *arXiv preprint arXiv:2406.01574*, 2024. [Online]. Available: https://arxiv.org/abs/2406.01574

[58] A. Myrzakhan, S. M. Bsharat, and Z. Shen, "Open-llm-leaderboard: From multi-choice to open-style questions for llms evaluation, benchmark, and arena," *arXiv preprint arXiv:2406.07545*, 2024. [Online]. Available: https://arxiv.org/abs/2406.07545

[59] W.-L. Chiang et al., "Chatbot arena: An open platform for evaluating llms by human preference," *arXiv preprint arXiv:2403.04132*, 2024. [Online]. Available: https://arxiv.org/abs/2403.04132

[60] Y. Li, S. Bubeck, R. Eldan, A. Del Giorno, S. Gunasekar, and Y. T. Lee, "Textbooks are all you need ii: Phi-1.5 technical report," *arXiv preprint arXiv:2309.05463*, 2023. [Online]. Available: https://arxiv.org/abs/2309.05463

[61] M. Javaheripi and S. Bubeck, *Phi-2: The surprising power of small language models*, https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models/, 2023.

[62] OpenAI, *Open models*, https://openai.com/open-models/, 2024.

[63] *Llm explorer – compare language models*, https://llm-explorer.com/, 2024.

[64] *Ranked agi*, https://rankedagi.com/, 2024.