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 27, 2025

Abstract

TO DO

Contents

1 Introduction

Since the creation of Transformers in 2017 [https://arxiv.org/abs/1706.03762] 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 [https://www.learntechlib.org/p/222408] 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 wasn't for free: billions were spent in larger datacenters, predictions that we soon wouldn't 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 [https://arxiv.org/abs/2001.08361] have 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". Secondly, it was also observed that different architectures could impact training performance, e.g. Transformers would have lower test loss than LSTMs. Those statements become 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's 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 it's 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) [https://arxiv.org/pdf/2005.14165] [https://huggingface.co/openai-community/gpt2-xl] [https://huggingface.co/openai-community/openai-gpt]. 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 [https://arxiv.org/abs/1009.1498], a metric that appears to be related to the model's performance and might help understanding the behaviour of such by proving insights about the weights distribution.

1.1 Work thesis

The main hypothesis of this work comes from Professor MURTA JUNIOR, Luiz Otavio study [Not published yet] 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 gates for new optimization processes during training, potentially reducing the amount of compute necessary to reach the best performance or even finding new maximums. Also, having a validated weights distribution x performance comparison should improve the data richness when studying a model's learning process in future studies. Thus, we can escalate 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 Computational Environment

2.2 Model Selection

2.3 LMC Complexity

According to [https://arxiv.org/abs/1009.1498], 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 Data Discretization

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** [⟨https://www.jstor.org/stable/2965501⟩] and **Rice Rule** [⟨https://www.scirp.org/pdf/ojs_2024022914191399.pdf⟩] 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 [⟨https://arxiv.org/abs/physics/0605197⟩].

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 [https://link.springer.com/content/pdf/10.1007/BF01025868.pdf]:

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

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.

Finally, the number of bins n can then be calculated as:

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

2.3.2 Extracting the Complexity Measure

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 [https://arxiv.org/abs/2001.08361], 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 [https://arxiv.org/abs/2401.04757], 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 [https://arxiv.org/abs/2203.08242] can influence the validity of results and makes 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 be 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 [https://arxiv.org/abs/2009.03300].
- **MMLU-Pro (5-shot)**: An enhanced version of MMLU that includes additional tasks and updated datasets to provide a better evaluation. It was created to address some limitations of the original MMLU [⟨https://proceedings.neurips.cc/paper_files/paper/2024/file/ad236edc564f3e3156e1b2feafb99a24-Paper-Datasets_and_Benchmarks_Track.pdf⟩].
- **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 [https://arxiv.org/abs/2406.07545].
- **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 users, using the Bradley-Terry system to estimate win rates and compute rankings/scores. [https://arxiv.org/abs/2403.04132].

2.4.2 Extracting the Inference Capability Measure

2.5 Comparing LMC Complexity and Inference Capability

References