Empirical Scaling Laws in Neural Models: Literature Analysis

Allegra Strippoli*
all.strippoli@stud.uniroma3.it
Universita degli studi Roma Tre
Rome

ACM Reference Format:
Allegra Strippoli. 2024. Empirical Scaling Laws in Neural Models: Literature Analysis.

1 ABSTRACT

This report discusses the empirical scaling laws observed in neural models. Scaling laws are a particular kind of power law that describe how deep learning models scale. [1]. Researchers find out that some key parameters, such as model size, dataset size, cost and loss, actually obey to these laws. Even though they explore different domains, they extract general laws that can be applied to multiple contexts. Through an analysis of literature from 2017 to 2023, the most significant findings are shown with particular interest in how it is possible to exploit these laws to improve performance and reduce cost. The strategies adopted over the years not always lead to the same conclusions and the discussion can still be considered open. The goal of the report is to show an overview of the main insights with special emphasis on the results obtained by Hoffman et al. that aim to find compute-optimal LLMs. Mathematical values found for coefficients are often glossed over, because the focus remains on model behavior and trends rather than the actual values found.

2 INTRODUCTION

A neural model can generally be described using four key parameters [7]:

- **N**: **N**umber of parameters of the model;
- **D**: training **D**ataset size;
- **C**: **C**omputing cost, it can be computed in different ways, here it is estimated in FLOPs;
- **L**: **L**oss, it evaluates the ability to accurately predict the output based on the input data;

Each of these four variables can be precisely defined into real numbers, and they are empirically found to be related by simple statistical laws. Within reasonable limits, performance depends very weakly on other architectural hyperparameters such as depth and width.

3 STATE OF THE ART

Some important steps for the analysis and understanding of scaling laws are given. In particular, four papers are cited. When scaling laws are found empirically, it becomes clear how they can be used as equations of an optimization problem to find the best trade-off between computational cost (number of FLOPs) and performance (loss value). Over the years, the study becomes more and more focused on finding the optimal solution.

*The drawings are self-made, based on graphics found in the paper [4]. They may not be accurate.

Scaling is Predictable, Empirically - 2017

The first work referred to is from 2017 [3] . Although previous works exist, they remain theoretical and involve scenarios of lower orders of scale, hence, they do not always explain results obtained by real applications. Hestness, Narang et al, in 2017, use the proportionality relationship between L and D that was already known in literature, but they are the first to predict the exponent empirically.

$$L \propto \left(\frac{1}{D}\right)^{\alpha}$$

Measurements are carried out in 4 learning domains: machine translation, language modeling, image processing, and speech recognition. They conclude that, among the factors that might change, only the task assigned to the model can lead to a different value of α. For instance, with LSTM, machine translation leads to $\alpha \sim 0.13$, while generative language modelling leads to $\alpha \in [0.06, 0.09]$. Changing the architecture optimizers, regularizers and loss functions, would only change the proportionality factor, not the exponent. One architecture might have $m = 1000$, another one $m = 500$ in $L = m * D^{-\alpha}$).

Scaling Laws for Autoregressive Generative Modeling - 2020

A 2020 study by Henighan, Kaplan, et al extend the discussion, identifying empirical scaling laws for autoregressive generative models in 4 domains: generative image modeling, video modeling, multimodal image-text models, and mathematical problem solving. They start from two ideas [2]:

(1) the benefits of scaling are highly predictable;
(2) when the loss L of a language model is bottlenecked by either the compute budget C, dataset size D, or model size N, the loss scales with each of these quantities as a simple power-law.

They study statistical relations between C, N, D, L over a wide range of values. They fix one of the two C, N, and vary the other one. D varies along using $D = C/6N$. The scaling relation for the loss is:

$$L(x) = L_0 + \left(\frac{x_0}{x}\right)^{\alpha}$$

- α is the modality-dependent scaling exponent;
- x is the varied variable (either N or C, occasionally D);
- L_0 is the irreducible loss, the entropy introduced by true data;
- $\left(\frac{x_0}{x}\right)^{\alpha}$ is the reducible loss, the entropy introduced by the model. Imagining a scenario where D, N, C tend to infinity, the reducible loss tends to zero. The researchers assume that an infinitely large transformer could model the data distribution exactly.

These scaling relations often hold to high precision, even when the reducible loss is much smaller than the irreducible loss. They demonstrate that a single architecture (the Transformer) with an autoregressive cross-entropy loss, scales smoothly with only minimal changes to hyperparameters such as width, depth, or learning rate. They also observe that larger models consistently learn faster, achieving any given value of the loss in fewer steps. This statement led researchers to invest more in the size of the model than in the quantity of tokens. This has been later criticized by Hoffman et al.

Scaling Laws for Neural Language Models - 2020

A second paper [5] published in 2020 by Kaplan, McCandlish et al. deeps dive into the study of empirical scaling laws for language model performance on the cross-entropy loss. They state that optimal training (the most computationally efficient) involves training very large models on a relatively small amount of data and stopping before convergence.

They also find a single equation that combines the dependence of N, D with L:

$$L(N,D) = \left[\left(\frac{N_C}{N}\right)^{\frac{\alpha_N}{\alpha_D}} + \frac{D_C}{D}\right]^{\alpha_D}$$

This law can be useful to keep track on overfitting. When the model's capacity N is too high relative to the dataset size D, the model could start to memorize the training data rather than generalizing from it. This would result in poor performance on unseen data.

Training Compute-Optimal Large Language Models - 2022

Hoffmann et al, in 2022 [4], investigate the optimal model size and number of tokens for training a transformer language model, under a given compute budget. So they assume to have a fixed FLOPs budget, and thet try to find a trade-off between model size and number of training tokens.

They train over 400 language models for one epoch, each ranging from 70 million to over 16 billion parameters. With a less number of parameters, Chinchilla outperforms models like GPT-3 (175 billion parameters), exploiting a very similar architecture but a greater number of training token. According to the researchers, this is evidence that optimality can be achieved with a smaller model size as long as more tokens are used.

4 KAPLAN AND HOFFMANN COMPARISON

The strategies adopted by the two teams have two main differences.

First, Kaplan et al. use a fixed number of training tokens and learning rate schedule for all models.

In contrast, Hoffmann et al. set the learning rate schedule to approximately match the number of training tokens. To their knowledge, a different approach could lead to undestimate the effectiveness of models and contribute to the conclusion that model size should increase faster than training data size.

Specifically, given a 10× increase computational budget, Kaplan et al. suggest that the size of the model should increase 5.5× while the number of training tokens should only increase 1.8×.

Instead, Hoffmann et al, find that for every doubling in model size, the number of training tokens should also double.

Secondly, the majority of the models used in Hoffman et al. analysis have more than 500M parameters, in contrast the majority of runs in Kaplan et al. are significantly smaller, many being less than 100M parameters.

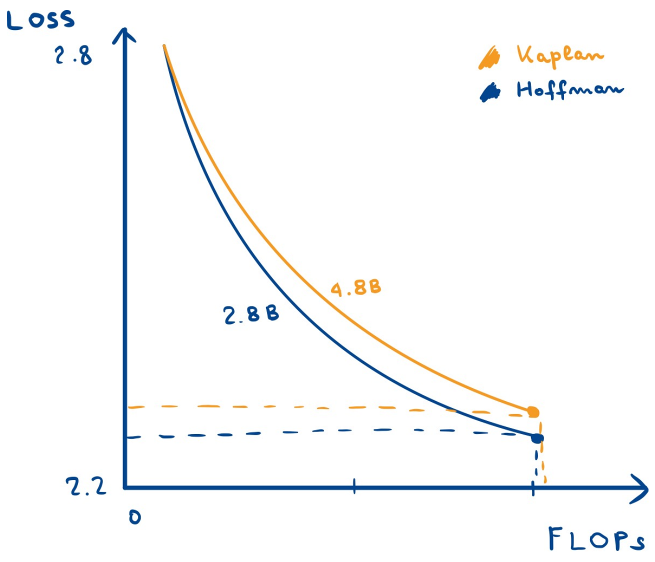

Figure 1: The yellow line describes the behaviour of a model with 4.8B parameters that follows Kaplan's approach. The blue one has 2.8B parameters and follows Hoffman's approach. The blue one reaches a lower value of loss, with the same FLOPs.

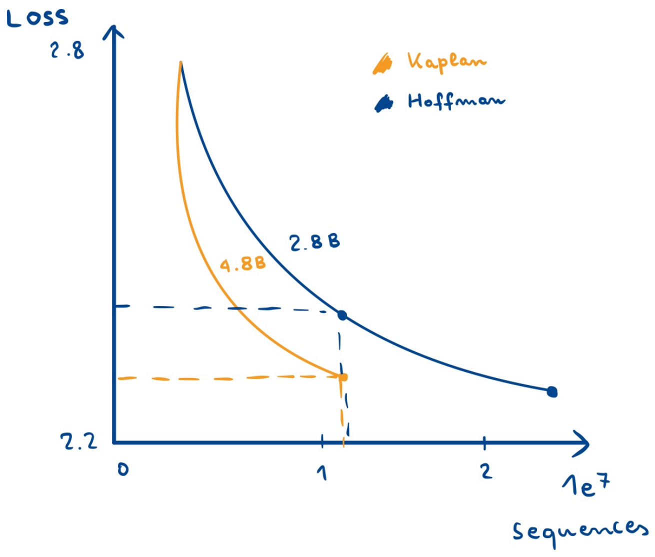

Figure 2: At an early stage, the bigger model (yellow) learns faster than the smaller one. The blue model needs a longer amount of time for training. But, at the end, it reaches a lower value of loss.

These results make Hoffman et al. believe that current LLMs are significantly under-trained, as a consequence of the focus on scaling the model size, whilst keeping the amount of training data constant.

5 CHINCHILLA'S APPROACH

For a greater understanding, three different approaches used by Hoffman et al are shown.

1: Fix model size and vary number of training tokens

For a fixed-size model family, researchers vary the amount of tokens used for training. For example, a fixed-size model might have 70 million parameters. They take four instances of that model and train each one with a different number of tokens (e.g., 1 billion, 5 billion, etc.). After training they know exactly the number of training FLOPs and loss, so they can recognize the most convenient one.

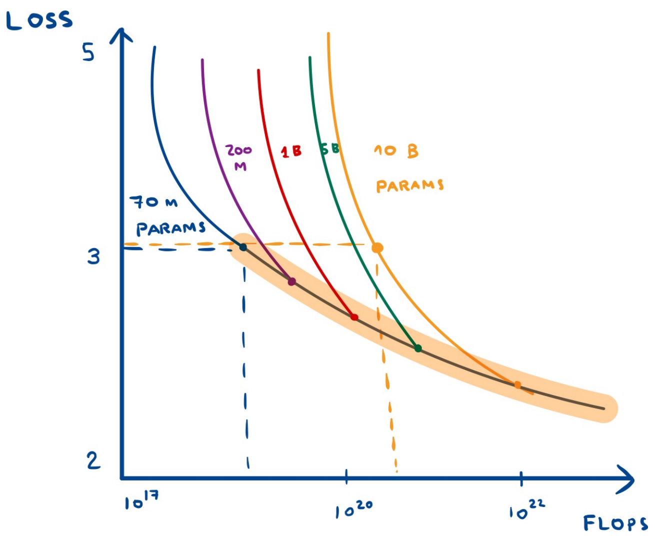

Figure 3: When the blue curve reaches its minimum, it's more compute-efficient compared to the yellow one, achieving the same loss with fewer FLOPs. However, as the number of FLOPs increases, the yellow curve enables reaching a lower loss value than the blue curve. Moreover, on the line marked in orange lie all the minimum points.

2: Vary model size and fix number of FLOP count

They use 9 distinct training FLOP counts, essentially by selecting the number of training tokens to ensure a constant final FLOP value. For each of these 9 FLOP levels, the training loss forms a valley-shaped curve. The point where the training loss is minimized indicates the optimal parameter count. Researchers note that all points of minimum lie roughly on the same oblique line (green in Figure 2). Compared with the previous approach, which derived FLOPs from the number of parameters and tokens, this one is complementary. From the number of FLOPs and tokens they derive the optimal number of parameters.

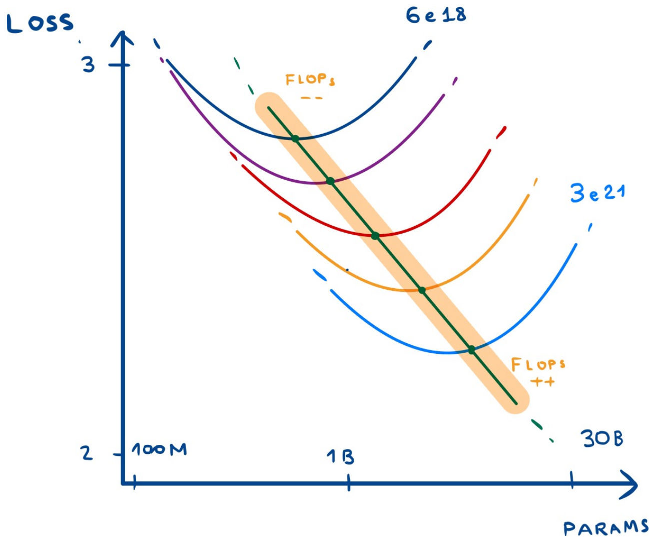

Figure 4: IsoFLOP curves show how to get the minimum loss, given a fixed budget.

3: Combine approach 1, 2 to get a parametric function L(N,D) and C(N,D)

The two laws presented in the paper concern the variation of cost and loss as N and D change. The first law:

$$L(N,D) = \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}} + L_0$$

- The first term captures the fact that a perfectly trained transformer with parameters underperforms an ideal generative process;
- The second term indicates that the transformer is not trained to convergence;
- L_0 is the entropy of natural text;

The second law:

$$C(N,D) = 6 * N * D$$

- It costs 6 FLOPs per parameter to train on one token;
- It costs 1 to 2 FLOPs per parameter to infer on one token;

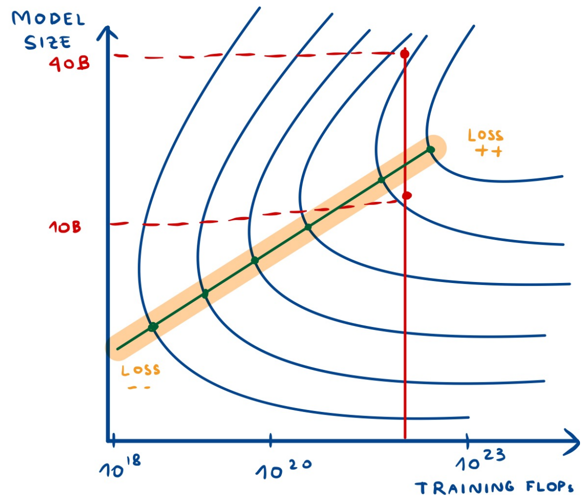

Figure 5: The two red points belong to the same IsoLoss curve. It means that a model with 40B parameters and another one with 10B parameters can achieve the same loss. This is possible due to the fact that a greater number of training token can be used to reduce the loss.

6 CHINCHILLA'S RESULTS

Researchers question the recent trend where the model size is increased without also increasing the number of training tokens. They hypothesize that the race to train larger and larger models is resulting in models that are underperforming compared to what could be achieved with the same compute budget. Their expectation is that scaling to larger datasets is only beneficial when the data is high-quality.

To prove their point, they have two comparable models trained on a large scale: Chinchilla and Gopher. They use the three approaches mentioned above [section 5] and end up claiming that Gopher is oversized. They estimate that for the same compute budget a smaller model trained on more data would perform better. According to their estimates, Gopher could achieve optimality with 67 billion parameters, rather than the 280 billion parameters that were actually employed.

Parameters	FLOPs in Gopher Unit	Tokens
67 Billion	1	1.5 Trillion
280 Billion	17.2	5.9 Trillion

Table 1

Even tough their performance prediction is validated by their experiments, researchers claim that the work can still be expanded by considering multiple epochal regimes, instead of a single epoch.

7 BEYOND CHINCHILLA

Some aspects of Chinchilla have been questioned. Hoffman et al. try to determine how best to scale the dataset and model size for a particular computational training budget. Instead, it may be more cost-effective to focus on the cost of inference rather than training. For instance, a smaller model trained longer could be cheaper at inference. This idea was advocated by some Meta AI researchers in a 2023 paper "LLaMA: Open and Efficient Foundation Language" [6]. Although Hoffmann et al. recommend training a 10B model on 200B tokens, they find that the performance of a 7B model continues to improve even after 1T tokens. Although the results are interesting, both the trained models are competitive and no one seems to have the upper hand.

8 CONCLUSION

Athough the discussion is wide-ranging and is only briefly covered in this report, a few key points were covered:

- The model performance depends most strongly on scale, which consists of three factors: N, D, and C;
- Given a fixed training sequence, larger models require fewer samples to reach the desired performance (Kaplan, 2020);
- Given a fixed budget, the right trade-off between number of tokens and parameters can be found following Hoffman et al's approach;
- The number of tokens and parameters should scale at roughly the same rate (Hoffman, 2022);
- Model performance can improve even if some parameters deviate from Chinchilla laws (Meta AI, 2023);

REFERENCES

[1] EpochAI. [n. d.] Scaling laws: a literature review. (). %5Curl%7Bhttps://epochai.org/blog/scaling-laws-literature-review%7D.
[2] Tom Henighan et al. 2020. Scaling laws for autoregressive generative modeling. (2020). arXiv: 2010.14701 [cs.LG].
[3] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. 2017. Deep learning scaling is predictable, empirically. (2017). arXiv: 1712.00409 [cs.LG].
[4] Jordan Hoffmann et al. 2022. Training compute-optimal large language models. (2022). arXiv: 2203.15556 [cs.CL].
[5] Jared Kaplan et al. 2020. Scaling laws for neural language models. (2020). arXiv: 2001.08361 [cs.LG].
[6] Hugo Touvron et al. 2023. Llama: open and efficient foundation language models. (2023). arXiv: 2302.13971 [cs.CL].
[7] Wikipedia. [n. d.] Neural scaling law. (). %5Curl%7Bhttps://en.wikipedia.org/wiki/Neural_scaling_law#cite_note-10%7D.