Deconstructing Scaling Law: The Trio of Optimization, Architecture, and Data

Su Jianlin

2026-07-29

Training a large neural network is influenced by many factors; changing the optimizer, the model architecture, or the training set can lead to drastically different results. In engineering practice, we jokingly call the process of tuning these factors based on empirical results "alchemy". But how can we elevate experience to laws, and describe their relationships more accurately and quantitatively? If we can figure out this problem, our alchemy will become more grounded.

"Scaling Law" attempts to answer this question in a relatively quantitative way. Since OpenAI's foundational work in 2020 (Kaplan Law), Scaling Law has become one of the most reliable empirical laws in deep learning. It can be used to estimate model performance, guide hyperparameter selection, and judge the effectiveness of certain modifications. Therefore, many works have attempted to provide a more essential interpretation of Scaling Law.

This article will also share some of the author's understanding of Scaling Law.

Preparation

Later in the text, we will repeatedly encounter the problem of "finding the minimum of a power-law combination under certain constraints", so we will first prepare two basic conclusions here.

Different-Power Inequality

First is the most frequently used inequality. Let $a, b, p, q, x > 0$, then

$$ax^p + bx^{-q} \geq (p+q)\left(\frac{a^q b^p}{p^p q^q}\right)^{\frac{1}{p+q}} \tag{1}$$

The condition for equality is

$$x = \left(\frac{bq}{ap}\right)^{\frac{1}{p+q}} \tag{2}$$

It is a generalization of the simple inequality $x + x^{-1} \geq 2$, which can be proven by taking derivatives or using the Weighted AM-GM inequality. Below we demonstrate the latter. The weighted AM-GM inequality states that $w_1 x_1 + w_2 x_2 \geq (w_1 + w_2)(x_1^{w_1} x_2^{w_2})^{\frac{1}{w_1+w_2}}$, so we have

$$ax^p + bx^{-q} = q \cdot \frac{ax^p}{q} + p \cdot \frac{bx^{-q}}{p} \geq (q+p) \left[\left(\frac{ax^p}{q} \right)^q \left(\frac{bx^{-q}}{p} \right)^p \right]^{\frac{1}{p+q}} = (p+q) \left(\frac{a^q b^p}{p^p q^q} \right)^{\frac{1}{p+q}} \quad (3)$$

The condition for equality is $\frac{ax^p}{q} = \frac{bx^{-q}}{p}$, i.e., $x = \left(\frac{bq}{ap} \right)^{\frac{1}{p+q}}$. For convenience, we call this inequality the "different-power inequality". Its main feature is having two power functions with opposite-signed exponents. More interestingly, the minimum point and the minimum value itself are still power laws, which is one of the cornerstones of the subsequent series of derivations.

Optimal Allocation Ratio

The different-power inequality finds the minimum of the sum of two power-law terms after fixing their product ($(x^p)^q (x^{-q})^p = 1$). There are also scenarios (such as finding the optimal ratio of parameters) where we need to fix their "sum" to find the minimum. Specifically, let $a, b, p, q, x, y > 0$, we want to find

$$\min_{x,y} ax^{-p} + by^{-q} \qquad \text{s.t.} \qquad x + y = 1 \quad (4)$$

Unfortunately, this problem does not have an elementary analytical solution, but numerical solution is not a problem. It is easy to see that the minimum must exist. Substituting $y = 1 - x$ and taking the derivative gives $bq(1-x)^{-q-1} - apx^{-p-1}$. Setting it to zero yields

$$\frac{x^{p+1}}{(1-x)^{q+1}} = \frac{ap}{bq} \quad (5)$$

The left side is obviously monotonically increasing with respect to x in $(0, 1)$, and when $x \to 1$, the left side approaches ∞, while when $x \to 0$, the left side approaches 0. By the intermediate value theorem, there is a unique solution in $(0, 1)$, which can be found directly using the bisection method.

Ideology

The training process of a model can be formalized as:

> Under the given data $\mathcal{D}$, architecture $\mathcal{A}$, and optimizer $\mathcal{O}$, minimize the loss function $L(\mathcal{E}|\mathcal{D}, \mathcal{A}, \mathcal{O})$.

Here $\mathcal{E}$ represents an ideal distribution, which can also be imagined as an immensely large test set. Any training set $\mathcal{D}$ we can construct is a subset of it, or a sampling result; $L(\mathcal{E}|\mathcal{D}, \mathcal{A}, \mathcal{O})$ represents the loss value the model can achieve on the ideal distribution $\mathcal{E}$ under these conditions.

Triple Decomposition

We consider the following decomposition

$$\begin{aligned} L(\mathcal{E}|\mathcal{D}, \mathcal{A}, \mathcal{O}) = & \underbrace{L(\mathcal{E}|\mathcal{D}, \mathcal{A}, \mathcal{O}) - L(\mathcal{D}|\mathcal{A}, \mathcal{O})}_{\text{Data}} \\ & + \underbrace{L(\mathcal{D}|\mathcal{A}, \mathcal{O}) - L(\mathcal{D}|\mathcal{A}, \infty)}_{\text{Optimization}} \\ & + \underbrace{L(\mathcal{D}|\mathcal{A}, \infty) - L(\mathcal{D}|\infty, \infty)}_{\text{Architecture}} \\ & + L(\mathcal{D}|\infty, \infty) \end{aligned} \tag{6}$$

This decomposition seems to complicate the problem, but actually it breaks down the distance from the current training state to the ideal target into three progressively deeper steps. Under the conventional assumption of "the more, the better", each bracket is non-negative, so it is equivalent to writing the total gap as the sum of three independently explainable distances, decoupling the impact of each variable on the loss function as much as possible, allowing us to more reasonably infer the dependencies between them.

Now let's explain the meaning of each term in the equation one by one.

Data Error

The first layer of decomposition is

$$L(\mathcal{E}|\mathcal{D}, \mathcal{A}, \mathcal{O}) = \Big[L(\mathcal{E}|\mathcal{D}, \mathcal{A}, \mathcal{O}) - L(\mathcal{D}|\mathcal{A}, \mathcal{O})\Big] + L(\mathcal{D}|\mathcal{A}, \mathcal{O}) \tag{7}$$

where $L(\mathcal{D}|\mathcal{A}, \mathcal{O})$ represents the loss value of the model on the training set $\mathcal{D}$ given the architecture $\mathcal{A}$ and optimizer $\mathcal{O}$.

Note that by definition, $L(\mathcal{E}|\mathcal{D}, \mathcal{A}, \mathcal{O})$ represents the loss of the model on the ideal distribution $\mathcal{E}$, which is our ultimate goal. However, $\mathcal{E}$ is untouchable during the training process; we can only deal with the training set $\mathcal{D}$, so we can only obtain the training loss $L(\mathcal{D}|\mathcal{A}, \mathcal{O})$, and then try to describe the gap between them via $L(\mathcal{E}|\mathcal{D}, \mathcal{A}, \mathcal{O}) - L(\mathcal{D}|\mathcal{A}, \mathcal{O})$.

This term is usually called the "generalization error", and its key influencing factor is data—such as the quantity, quality, diversity of the data, etc. In addition, the architecture $\mathcal{A}$ and optimizer $\mathcal{O}$ may also change the generalization error. Which variables to consider specifically depends on our analysis goals.

Optimization Error

The second layer of decomposition is

$$L(\mathcal{D}|\mathcal{A},\mathcal{O}) = \Big[L(\mathcal{D}|\mathcal{A},\mathcal{O}) - L(\mathcal{D}|\mathcal{A},\infty)\Big] + L(\mathcal{D}|\mathcal{A},\infty) \tag{8}$$

where $L(\mathcal{D}|\mathcal{A},\infty)$ represents the ideal loss achievable on the training set when optimization is pushed to the extreme—such as having a perfect super-optimizer, or pushing the training steps and tuning times to infinity.

Therefore, $L(\mathcal{D}|\mathcal{A},\infty)$ represents the ceiling of the optimizer, while $L(\mathcal{D}|\mathcal{A},\mathcal{O})-L(\mathcal{D}|\mathcal{A},\infty)$ represents the distance between the practical optimizer and this ceiling. It measures whether the optimizer is good enough—for instance, whether the learning rate is appropriate, whether the training steps are sufficient, whether the batch size is large enough to stabilize the gradient, etc.

Architecture Error

The third layer of decomposition is

$$L(\mathcal{D}|\mathcal{A},\infty) = \Big[L(\mathcal{D}|\mathcal{A},\infty) - L(\mathcal{D}|\infty,\infty)\Big] + L(\mathcal{D}|\infty,\infty) \tag{9}$$

where $L(\mathcal{D}|\infty,\infty)$ represents the most ideal loss achievable on the training set when both optimization and architecture are pushed to the extreme—any excellent optimizer, any powerful model.

Therefore, $L(\mathcal{D}|\infty,\infty)$ is the theoretical limit determined by the current data itself, while $L(\mathcal{D}|\mathcal{A},\infty)-L(\mathcal{D}|\infty,\infty)$ represents the performance gap between the practical model and its theoretical limit. It measures whether the architecture is good enough—whether the parameter count is sufficient, whether the depth and width are sufficient, whether there is still room for residual improvement, etc.

Optimization

In this section, we first explore the optimization gap $F_{\text{opt}} = L(\mathcal{D}|\mathcal{A},\mathcal{O})-L(\mathcal{D}|\mathcal{A},\infty)$. Given an optimizer (such as Adam or Muon), we mainly care about the impact of three core parameters: learning rate η, batch size B, and training steps T. Of course, theoretically, other parameters like momentum and other details can also be considered, but here we mainly focus on these three parameters.

Relationship Analysis

First, a reasonable assumption is "the more you train, the better the effect", where training more includes both meanings of "more steps" and "larger learning rate". Alternatively, we

can intuitively believe that $T\eta$ is the "distance" the model has traveled, and the longer the distance, the better the effect. Thus, we can guess that one contribution is $\alpha_1(T\eta)^{-\gamma_1}$.

On the other hand, the training effect is also affected by noise. A reasonable assumption is that the larger the noise, the worse the effect. Noise comes from two sources: first, the batch size B (the smaller it is, the larger the noise); second, the learning rate, which represents the non-smoothness of the training process (the larger it is, the larger the noise). So we guess there is another term $\alpha_2 B^{-\gamma_2} + \alpha_3 \eta^{\gamma_3}$.

Adding these two parts together yields

$$F_{\text{opt}} \sim \alpha_1(T\eta)^{-\gamma_1} + \alpha_2 B^{-\gamma_2} + \alpha_3 \eta^{\gamma_3} \tag{10}$$

We can also understand this formula by dividing the training process into two stages: in the early stage of training, noise is secondary, and the term $\alpha_1(T\eta)^{-\gamma_1}$ brings the loss function down quickly. Subsequently, noise gradually starts to play a role, and the model begins to oscillate near the target point, similar to a spiral descent trajectory.

This form is consistent with what is used in 2503.12645 and 2603.15958. It is worth pointing out that these two works are not experimental fits, but rather direct theoretical analyses of the convergence of optimizers like SignSGD and Muon, concluding that $\gamma_1 = \gamma_3 = 1, \gamma_2 = 1/2$. In the subsequent derivations, we can substitute these values for simple verification.

Optimal Learning Rate

Equation (10) has 6 parameters. Fitting it directly would require taking many data points, and the cost would be very high. It is also very prone to overfitting. To address this, we can use the optimal parameter assumption to further simplify the form.

First, according to the different-power inequality, the optimal learning rate that minimizes the right-hand side can be found as

$$\eta^* = \left(\frac{\gamma_1\alpha_1 T^{-\gamma_1}}{\gamma_3\alpha_3}\right)^{\frac{1}{\gamma_1+\gamma_3}} \sim T^{-\frac{\gamma_1}{\gamma_1+\gamma_3}} \tag{11}$$

The corresponding minimum value is

$$F^*_{\text{opt}} = \underbrace{(\gamma_1+\gamma_3)\left(\frac{(\alpha_1 T^{-\gamma_1})^{\gamma_3}\alpha_3^{\gamma_1}}{\gamma_1^{\gamma_1}\gamma_3^{\gamma_3}}\right)^{\frac{1}{\gamma_1+\gamma_3}}}_{\sim T^{-\frac{\gamma_1\gamma_3}{\gamma_1+\gamma_3}}} + \alpha_2 B^{-\gamma_2} \tag{12}$$

This tells us two results: 1. There exists some $0 < c < 1$ such that the optimal learning rate is inversely proportional to T^c; 2. Assuming we can always find the optimal learning rate for each configuration, the asymptotic law of the optimization error can be simplified

to the form $\tilde{\alpha}_1 T^{-\tilde{\gamma}_1} + \alpha_2 B^{-\gamma_2}$, reducing the number of parameters to 4. This is exactly the decoupled form proposed by 2607.01487, while 2605.09154 derived the same conclusion of modeling B and T separately starting from a noisy quadratic form.

Optimal Batch Size

Now, starting from the assumption of the optimal learning rate, let the optimization error be

$$F_{\text{opt}} \sim \tilde{\alpha}_1 T^{-\tilde{\gamma}_1} + \alpha_2 B^{-\gamma_2} \tag{13}$$

Let $K = BT$, which represents the number of samples learned during the training process. Note that we do not restrict Multi-Epoch, so it is possible that some samples are learned repeatedly. If K is fixed, the right-hand side becomes $\tilde{\alpha}_1 (B/K)^{\tilde{\gamma}_1} + \alpha_2 B^{-\gamma_2}$. Continuing with the different-power inequality, the minimum value can be found as

$$F^*_{\text{opt}} = (\tilde{\gamma}_1 + \gamma_2) \left(\frac{(\tilde{\alpha}_1 K^{-\tilde{\gamma}_1})^{\gamma_2} \alpha_2^{\tilde{\gamma}_1}}{\tilde{\gamma}_1^{\tilde{\gamma}_1} \gamma_2^{\gamma_2}} \right)^{\frac{1}{\tilde{\gamma}_1 + \gamma_2}} \sim K^{-\frac{\tilde{\gamma}_1 \gamma_2}{\tilde{\gamma}_1 + \gamma_2}} \tag{14}$$

Equality holds at

$$B^* = \left(\frac{\alpha_2 \gamma_2 K^{\tilde{\gamma}_1}}{\tilde{\alpha}_1 \tilde{\gamma}_1} \right)^{\frac{1}{\tilde{\gamma}_1 + \gamma_2}} \sim K^{\frac{\tilde{\gamma}_1}{\tilde{\gamma}_1 + \gamma_2}} \tag{15}$$

This also leads to two conclusions: given the total number of training samples K, the optimal batch size B^* is proportional to K^c, where $c \in (0, 1)$; and under the optimal batch size, the Scaling Law simplifies to $\hat{\alpha}_1 K^{-\hat{\gamma}_1}$, which is the classic Scaling Law form.

Brief Summary

Now we can summarize: the general Scaling Law for an optimizer is $\alpha_1 (T\eta)^{-\gamma_1} + \alpha_2 B^{-\gamma_2} + \alpha_3 \eta^{\gamma_3}$. If we assume it always runs at the optimal learning rate, it can be simplified to $\tilde{\alpha}_1 T^{-\tilde{\gamma}_1} + \alpha_2 B^{-\gamma_2}$. If the total number of training samples K is further given, and we assume the optimal batch size can always be found, it simplifies to $\hat{\alpha}_1 K^{-\hat{\gamma}_1}$.

As for the optimal parameters, the optimal batch size is proportional to some power of K that is not greater than 1, which aligns with Step Law; the optimal learning rate is inversely proportional to some power of T that is not greater than 1. Converting according to $T^* = K/B^*$, the optimal learning rate is also inversely proportional to some power of K that is not greater than 1, which aligns with Microsoft Law, but is opposite to Step Law.

If we substitute the previously mentioned theoretical values $\gamma_1 = \gamma_3 = 1, \gamma_2 = 1/2$, we get $\tilde{\gamma}_1 = 1/2$, and subsequently $B^* \sim K^{1/2}$, which is quite close to the $B^* \sim K^{0.571}$ given by Step Law; furthermore, $\eta^* \sim T^{-1/2}$, converting via $T^* = K/B^*$ gives $\eta^* \sim K^{-1/4}$, which is not far from the $\eta^* \sim K^{-0.32}$ of Microsoft Law. Finally, $F^*_{\text{opt}} \sim K^{-1/4}$ is also very close to the $\sim K^{-0.28}$ given by Chinchilla Law.

Architecture

Next, we turn to the model gap $F_{\text{arch}} = L(\mathcal{D}|\mathcal{A}, \infty) - L(\mathcal{D}|\infty, \infty)$. This purely discusses the contribution of the architecture $\mathcal{A}$ to the loss function. Classic variables include the parameter count N, width W, depth H, etc. At the same time, the introduction of architectural variables will in turn affect the variation law of the optimization gap. We will try to sort out all this content as much as possible.

Model Parameters

Under the premise of a fixed overall architecture, the main variable of the model is the parameter count N. Assuming that a larger parameter count yields better results, it is reasonable to believe that

$$F_{\text{arch}} \sim \alpha_4 N^{-\gamma_4} \tag{16}$$

This is the most straightforward assumption of Scaling Law regarding parameter count, consistent with Kaplan Law and Chinchilla Law. Kaplan Law gives the result $\gamma_4 = 0.076$, while Chinchilla Law gives a fitted result of $\gamma_4 = 0.34$. It is currently widely believed that Chinchilla Law is more accurate as the training scale becomes larger.

A point of contention here is whether the count of parameters N should include Embedding. The mainstream practice is not to include it, but this may cause significant deviations at small scales. This might be one of the reasons for the difference between Kaplan Law and Chinchilla Law results (in the era of Kaplan's experiments, training scales were generally small). The paper 2406.12907 provides a detailed analysis of this. For how to more accurately consider the contribution of Embedding, please refer to the later "Memory Layers" section.

Instead of compressing the entire architecture into a single parameter count N, we can be a bit more refined. For example, we can separate width W and depth H to study whether a "tall and thin" or "short and fat" model is better:

$$F_{\text{arch}} \sim \alpha_W W^{-\gamma_W} + \alpha_H H^{-\gamma_H} \tag{17}$$

Based on the approximate model parameter count $N \sim W^2 H$, we can find the optimal width, depth, and the corresponding F^*_{arch} under a fixed parameter count:

$$W^* \sim N^{\frac{\gamma_H}{\gamma_W + 2\gamma_H}}, \qquad H^* \sim N^{\frac{\gamma_W}{\gamma_W + 2\gamma_H}}, \qquad F^*_{\text{arch}} \sim N^{-\frac{\gamma_W \gamma_H}{\gamma_W + 2\gamma_H}} \tag{18}$$

The paper 2606.25008 theoretically proposes (from 2505.10465 and 2602.05970) that $\gamma_W = \gamma_H = 1$. Substituting this yields

$$W^* \sim N^{1/3}, \qquad H^* \sim N^{1/3}, \qquad F^*_{\text{arch}} \sim N^{-1/3} \tag{19}$$

The final $F^*_{\text{arch}} \sim N^{-1/3}$ is still quite close to Chinchilla Law.

Optimization Law

At the same time, changes in parameter count also affect the optimization process. In Equation (10), the three coefficients $\alpha_1, \alpha_2, \alpha_3$ are treated as constants, which is under the assumption of a given architecture $\mathcal{A}$. Now that we have introduced the parameter count N, $\alpha_1, \alpha_2, \alpha_3$ are naturally also functions of N.

We also understand this in two parts: on the one hand, a larger parameter count means stronger model capability and faster loss descent, so we replace α_1 with $\alpha_1 N^{-\gamma_5}$; on the other hand, a larger parameter count means a more complex model and larger noise, so we replace α_2, α_3 with $\alpha_2 N^{\gamma_6}$ and $\alpha_3 N^{\gamma_7}$ respectively. Thus

$$F_{\text{opt}} \sim \alpha_1 N^{-\gamma_5} (T\eta)^{-\gamma_1} + \alpha_2 N^{\gamma_6} B^{-\gamma_2} + \alpha_3 N^{\gamma_7} \eta^{\gamma_3} \tag{20}$$

A natural question here is: why only consider the variation of $\alpha_1, \alpha_2, \alpha_3$ with N, and not the variation of the exponents $\gamma_1, \gamma_2, \gamma_3$? We will defer this question to the "Power Law Section". Now, repeating the calculations from the "Optimization" section, we get

$$\eta^* \sim T^{-\frac{\gamma_1}{\gamma_1+\gamma_3}} \cdot N^{-\frac{\gamma_5+\gamma_7}{\gamma_1+\gamma_3}} \sim K^{-\frac{\gamma_1\gamma_2}{\gamma_1\gamma_3+\gamma_2(\gamma_1+\gamma_3)}} \cdot N^{\frac{\gamma_1\gamma_6-\gamma_7(\gamma_1+\gamma_2)-\gamma_5\gamma_2}{\gamma_1\gamma_3+\gamma_2(\gamma_1+\gamma_3)}} \tag{21}$$

$$B^* \sim K^{\frac{\gamma_1\gamma_3}{\gamma_1\gamma_3+\gamma_2(\gamma_1+\gamma_3)}} \cdot N^{\frac{\gamma_6(\gamma_1+\gamma_3)+\gamma_5\gamma_3-\gamma_7\gamma_1}{\gamma_1\gamma_3+\gamma_2(\gamma_1+\gamma_3)}} \tag{22}$$

$$F^*_{\text{opt}} \sim K^{-\frac{\gamma_1\gamma_2\gamma_3}{\gamma_1\gamma_3+\gamma_2(\gamma_1+\gamma_3)}} \cdot N^{\frac{\gamma_1\gamma_2\gamma_7+\gamma_1\gamma_3\gamma_6-\gamma_2\gamma_3\gamma_5}{\gamma_1\gamma_3+\gamma_2(\gamma_1+\gamma_3)}} \tag{23}$$

If we agree with the results of Kaplan Law, Chinchilla Law, and Step Law, then B^* and F^*_{opt} are independent of N. By setting the exponents of N in these two terms to 0, we can solve and get

$$\gamma_3\gamma_5 = \gamma_1\gamma_7, \qquad \gamma_6 = 0 \tag{24}$$

Substituting into the formulas, we find only one new parameter γ_7 is added:

$$F_{\text{opt}} \sim \alpha_1 N^{-\frac{\gamma_1\gamma_7}{\gamma_3}} (T\eta)^{-\gamma_1} + \alpha_2 B^{-\gamma_2} + \alpha_3 N^{\gamma_7} \eta^{\gamma_3} \tag{25}$$

$$\eta^* \sim T^{-\frac{\gamma_1}{\gamma_1+\gamma_3}} \cdot N^{-\frac{\gamma_7}{\gamma_3}} \sim K^{-\frac{\gamma_1\gamma_2}{\gamma_1\gamma_3+\gamma_2(\gamma_1+\gamma_3)}} \cdot N^{-\frac{\gamma_7}{\gamma_3}} \tag{26}$$

$$B^* \sim K^{\frac{\gamma_1\gamma_3}{\gamma_1\gamma_3+\gamma_2(\gamma_1+\gamma_3)}} \tag{27}$$

$$F^*_{\text{opt}} \sim K^{-\frac{\gamma_1\gamma_2\gamma_3}{\gamma_1\gamma_3+\gamma_2(\gamma_1+\gamma_3)}} \tag{28}$$

Interestingly, the form of η^* exactly matches Microsoft Law, i.e., it is negatively correlated with both K and N. This is not trivial, because we only assumed that B^* and F^*_{opt} are independent of N, and made no assumption about η^*. Finally, substituting the theoretical values $\gamma_1 = \gamma_3 = 1, \gamma_2 = 1/2$, we get

$$\eta^* \sim K^{-1/4} N^{-\gamma_7}, \qquad B^* \sim K^{1/2}, \qquad F^*_{\text{opt}} \sim K^{-1/4} \tag{29}$$

As for γ_7, Microsoft Law gives 0.23, while Step Law gives 0.713. Considering that these two laws have completely opposite dependencies on K, it is normal for them to have significant differences in this exponent. The theory generally leans more towards Microsoft Law. From some convex optimization results, N^{γ_7} is related to the standard deviation of the gradients of all parameters, so it is guessed to be between 0 and 0.5. Combined with Microsoft Law, we can take a guess of $1/4$.

Given Compute

Combining F^*_{opt} and F_{arch}, we have

$$F^*_{\text{opt}} + F_{\text{arch}} \sim \hat{\alpha}_1 K^{-\hat{\gamma}_1} + \alpha_4 N^{-\gamma_4} \tag{30}$$

For dense models, the computation per step is basically proportional to the parameter count N, and K is the number of samples learned during training, which is also proportional to the training computation. So the total computational cost consumed by the training process is $C \sim NK$. The proportionality constant for standard architectures is roughly 6, including $2NK$ for forward propagation and $4NK$ for backward propagation.

In actual training, compute is usually limited. We want to achieve maximum intelligence under a fixed compute budget C, which means we need to find the optimal K^* and N^* under the constraint $NK \sim C$. Substituting $K \sim C/N$ into the above equation (absorbing constant factors into the coefficients), we get

$$F^*_{\text{opt}} + F_{\text{arch}} \sim \hat{\alpha}_1 C^{-\hat{\gamma}_1} N^{\hat{\gamma}_1} + \alpha_4 N^{-\gamma_4} \tag{31}$$

This is once again in the form of the different-power inequality. Directly applying the formula, we get

$$N^* \sim C^{\frac{\hat{\gamma}_1}{\hat{\gamma}_1+\gamma_4}}, \qquad K^* \sim C^{\frac{\gamma_4}{\hat{\gamma}_1+\gamma_4}}, \qquad F^* \sim C^{-\frac{\hat{\gamma}_1\gamma_4}{\hat{\gamma}_1+\gamma_4}} \tag{32}$$

If we substitute the previous theoretical values $\hat{\gamma}_1 = 1/4$ and $\gamma_4 = 1/3$, then

$$N^* \sim C^{3/7}, \qquad K^* \sim C^{4/7}, \qquad F^* \sim C^{-1/7} \tag{33}$$

This is very close to the core conclusion of Chinchilla Law: the optimal model size and optimal data amount should scale roughly proportionally (the paper's fitted results are $N^* \sim C^{0.46}$, $K^* \sim C^{0.54}$). Finally, substituting the expressions for N^* and K^* into $\eta^* \sim K^{-1/4} N^{-\gamma_7}$ and $B^* \sim K^{1/2}$ from the previous section, we get

$$\eta^* \sim C^{-(1+3\gamma_7)/7}, \qquad B^* \sim C^{2/7} \tag{34}$$

Here, $B^* \sim C^{2/7}$ is quite close to the $B^* \sim C^{0.3271}$ of DeepSeek Law. However, DeepSeek Law gives $\eta^* \sim C^{-0.1250}$, while here, even substituting $\gamma_7 = 1/4$, we can only get $\eta^* \sim C^{-1/4}$, which is still quite far off. It seems that the results of different groups are relatively consistent on the optimal batch size, but there is a large divergence on the optimal learning rate, which may be closely related to specific optimization settings and learning rate schedules.

Sparse Architecture

Just now we said "computation is roughly proportional to the parameter count N", which holds for Dense models. Assuming the core operation of the model is the Linear layer, an input of $a \times b$ is multiplied by parameters of $b \times c$. The computation is $\mathcal{O}(abc)$, and the parameter count is bc, both proportional to bc, so the parameter count is the computation. But in recent years, everyone has also been committed to researching architectures that decouple parameter count and computation, such as MoE.

MoE is naturally well-deservedly the mainstream architecture today; almost every open-source large model is MoE. An important new parameter is the sparsity S, which can be defined as the ratio of total parameters to activated parameters, or the ratio of total Experts to activated Experts. Theoretically, the theoretical computation of MoE roughly depends only on the activated parameters, meaning that increasing the sparsity S theoretically does not increase computation, but can reduce the loss, so increasing sparsity is always cost-effective.

A simple scheme to incorporate sparsity into the Scaling Law is:

$$F_{\text{arch}} \sim \alpha_4 N_{act}^{-\gamma_{act}} N_{total}^{-\gamma_{total}} = N_{act}^{-(\gamma_{act}+\gamma_{total})} S^{-\gamma_{total}} \tag{35}$$

where N_{act}, N_{total} are the activated parameter count and total parameter count respectively, and $S = N_{total}/N_{act}$. A similar form also appears in 2501.12370. Viewed from Equation (16), this is equivalent to assuming that S only affects the coefficient α_4 in a power-law manner, but does not affect the exponent γ_4. From the above equation, we can also derive the concept of "equivalent parameter count":

$$N_{eff} = N_{act}^{\frac{\gamma_{act}}{\gamma_{act}+\gamma_{total}}} N_{total}^{\frac{\gamma_{total}}{\gamma_{act}+\gamma_{total}}}, \qquad F_{\text{arch}} \sim \alpha_4 N_{act}^{-\gamma_{act}} N_{total}^{-\gamma_{total}} = \alpha_4 N_{eff}^{-(\gamma_{act}+\gamma_{total})} \tag{36}$$

That is, an MoE model with an activated parameter count of N_{act} and a total parameter count of N_{total} is equivalent to a Dense model with a parameter count of N_{eff}. This concept originates from the Effective Parameter Count proposed by 2202.01169, and later Ling Law further expanded it into an "efficiency lever" and explored its variation patterns.

However, the computation is only proportional to the activated parameter count N_{act}, which means we can let N_{eff} remain unchanged while letting $N_{act} \to 0$, in other words, reducing the computation to nearly zero without changing the effect, but this seems unrealistic. Therefore, a reasonable guess is that F_{arch} should have an additional penalty term regarding N_{act} to guarantee a certain amount of activated parameters:

$$F_{\text{arch}} \sim \alpha_4 N_{act}^{-\gamma_{act}} N_{total}^{-\gamma_{total}} + \alpha_8 N_{act}^{-\gamma_8} \tag{37}$$

Works on modeling sparsity also include 2309.08520, 2501.12370, 2502.05172, etc., and the forms of the Scaling Laws they set are all different. In addition to sparsity, the granularity of the Expert (first modeled by 2402.07871) and the Shared Expert also have a certain impact

on the effect. 2509.23678 blended these factors into a very massive form for experimental fitting.

Of course, "arbitrarily increasing sparsity is cost-effective" is only theoretical. In practice, routing overhead, inference efficiency, etc., must also be considered. Increasing sparsity is not completely free and requires the joint design of algorithms and infrastructure.

Memory Layers

In addition to MoE, other ways to decouple parameter count and computation include sparse Memory layers, classics such as PKM and UltraMem, while emerging ones like Over-Encoding and Engram can also be classified into this category.

In fact, this type of work can also be seen as another extreme of MoE: their "Experts" are simplified into trainable vectors without any computation, and then a trainable Router (Pointer) or N-gram Hash is used to select the "Experts" to be activated. From this perspective, their Scaling Law should be similar to MoE. For example, after pairing a Dense model with several Memory layers, the Scaling Law should similarly take the form of Equation (37).

If two different types of sparse designs, MoE and Memory, are used simultaneously, what should the Scaling Law look like? Let's first define a few notations $N_{act}, N_{moe}, N_{mem}, N_{total}$, representing "activated parameter count (only counting parameters that generate computation, i.e., Dense part plus activated Experts)", "total parameter count after removing Memory", "total parameter count after removing inactive Experts", and "total parameter count", respectively. They satisfy the identity

$$N_{moe} + N_{mem} = N_{total} + N_{act} \tag{38}$$

If we guess that the roles of MoE and Memory are complementary, then their Scaling Law might be additive, i.e.,

$$F_{\text{arch}} \sim \alpha_{4a} N_{act}^{-\gamma_{act}} N_{moe}^{-\gamma_{moe}} + \alpha_{4b} N_{act}^{-\gamma_{act}} N_{mem}^{-\gamma_{mem}} + \alpha_8 N_{act}^{-\gamma_8} \tag{39}$$

This also gives rise to a new optimization problem: if we must fix the activated parameter count N_{act} (compute bottleneck) and the total parameter count N_{total} (memory bottleneck), how should we allocate the parameters between MoE and Memory? According to the identity (38), $N_{moe} + N_{mem}$ is a constant. Minimizing the above equation under this constraint leads to the sum-constrained minimization problem introduced in the "Optimal Allocation Ratio" section, which can be solved numerically.

That is, it is not optimal to allocate all parameters to either MoE or Memory; instead, there exists an optimal allocation ratio, which is similar to the findings of Engram.

Brief Summary

Similar to the Optimization section, this section also first makes power-law assumptions based on experience, and then mainly uses the different-power inequality for optimization

calculations.

The simplest variable to quantify a model is the total parameter count N; for more detail, one can consider width W and depth H separately. Changes in the model will also cause changes in the optimization error, so we also briefly discussed the joint effect of various optimization parameters and the parameter count. Under the assumption of a Dense model, the parameter count itself also represents the computation, and the optimizer's training steps T and batch size B are also positively correlated with computation. Thus, we can also explore how to determine the optimal parameter count and the corresponding optimizer parameters when the total budget C is fixed.

If we further refine, there are many more variables in the model architecture. For example, the MoE architecture distinguishes between "activated parameters" and "total parameters", and designs like Engram are similar. When they appear together, there is a problem of optimal parameter allocation. We haven't even mentioned residual improvements like MHC and AttnRes, etc. In short, there are many variables that can be considered for the model, and we can only briefly touch upon them here.

Data

Finally, we turn to the data gap $F_{\text{data}} = L(\mathcal{E}|\mathcal{D},\mathcal{A},\mathcal{O}) - L(\mathcal{D}|\mathcal{A},\mathcal{O})$. By definition, this term requires us to focus on the effect on the ideal distribution $\mathcal{E}$, but we also said that the ideal distribution is theoretically untouchable. If so, how can we measure it?

In fact, there is no good way. We can only choose a dataset that has not been trained on and that we consider sufficiently representative as a test set, and use the test loss calculated from it as an approximation for $L(\mathcal{E}|\mathcal{D},\mathcal{A},\mathcal{O})$. Therefore, similar to benchmarks, the construction of the test set is particularly important, as it represents the accuracy of our understanding and description of the ideal target.

Data Size

The core hyperparameter of data is the size of the training set D. We believe that the larger the training set, the better the generalization performance (i.e., "seeing more leads to broader knowledge"), so it contributes a term $\alpha_9 D^{-\gamma_9}$.

If the data for each step is uniformly randomly sampled from the training set, then K/D is the average number of times each sample is trained, i.e., the number of training Epochs. We believe that the more severe the Multi-Epoch, the worse the generalization performance, so it contributes a term $\alpha_{10}(K/D)^{\gamma_{10}}$. Thus, a basic form is

$$F_{\text{data}} \sim \alpha_9 D^{-\gamma_9} + \alpha_{10}(K/D)^{\gamma_{10}} \tag{40}$$

Most Scaling Law works treat K and D as the same under the Single-Epoch assumption. This article distinguishes them precisely to be able to consider the impact of Multi-Epoch.

The recent paper "Prescriptive Scaling Laws for Data Constrained Training" also introduces a similar power law, but replaces K/D with $K/D-1$, and simultaneously considers the impact of the model parameter count. It generally believes that a larger parameter count makes the model easier to overfit, so it should not be trained for Multi-Epoch; thus, α_{10} is made positively correlated with N. These modifications can also be adjusted as needed.

On the other hand, articles like "Scaling Data-Constrained Language Models" correct the Scaling Law by introducing the concept of "value decay", converting the amount of data after Multi-Epoch into an "effective data amount". However, the author feels that this term is only needed if one does not distinguish between K and D. Since we have already distinguished K and D, we only need to directly penalize the overfitting risk brought by Multi-Epoch.

Optimal Number of Epochs

Combining the optimization error and data error (fixing D and N, and assuming training is always under optimal hyperparameters), the result is:

$$F_{\text{opt}}^* + F_{\text{data}} \sim \hat{\alpha}_1 K^{-\hat{\gamma}_1} + \alpha_{10}(K/D)^{\gamma_{10}} \tag{41}$$

This clearly shows that the more you train, the smaller the optimization error, but the more severe the data repetition, and the larger the generalization error. Thus, again according to the different-power inequality, the optimal value of K can be found as

$$K^* = \left(\frac{\hat{\alpha}_1\hat{\gamma}_1}{\alpha_{10}\gamma_{10}}D^{\gamma_{10}}\right)^{\frac{1}{\hat{\gamma}_1+\gamma_{10}}} \sim D^{\frac{\gamma_{10}}{\hat{\gamma}_1+\gamma_{10}}} \qquad \frac{K^*}{D} \sim D^{-\frac{\hat{\gamma}_1}{\hat{\gamma}_1+\gamma_{10}}} \tag{42}$$

This provides a scaling law for Multi-Epoch, concluding that: with less data, one should actually train for more epochs; with more data, the optimal number of Epochs is smaller. The conclusion is exactly opposite to 2511.13421. A possible improvement scheme is to generalize $\alpha_{10}(K/D)^{\gamma_{10}}$ to $\alpha_{10}K^{\gamma_{10}}D^{-\gamma_{11}}$. In this case, we have

$$K^* = \left(\frac{\hat{\alpha}_1\hat{\gamma}_1}{\alpha_{10}\gamma_{10}}D^{\gamma_{11}}\right)^{\frac{1}{\hat{\gamma}_1+\gamma_{10}}} \sim D^{\frac{\gamma_{11}}{\hat{\gamma}_1+\gamma_{10}}} \qquad \frac{K^*}{D} \sim D^{\frac{\gamma_{11}-\hat{\gamma}_1-\gamma_{10}}{\hat{\gamma}_1+\gamma_{10}}} \tag{43}$$

In this way, it is possible for the optimal number of Epochs to either increase or decrease as data increases.

But even if we switch to $\alpha_{10}K^{\gamma_{10}}D^{-\gamma_{11}}$, there are still some unreasonable aspects here. For instance, when $K\to\infty$, it also approaches infinity, but empirically, even if we keep training, the test set loss should not be infinite. However, if we only fit within a small range (assuming the number of Multi-Epochs cannot be too large), the power-law assumption can still yield practically useful results.

Extended Thoughts

Besides the core parameter of data size D, many works have made detailed distinctions regarding the composition of data, such as domain ratios (2403.16952, 2507.09404, 2603.19149, 2605.12715, 2606.08167), quality levels (2510.03313), modality ratios (2607.22043), etc. These works are diverse and it's hard to distill a unified form, so we won't expand on them one by one.

Compared to optimization and architecture, the Scaling Law on the data side indeed gives a feeling of being "messy" and "blurry". The reason is not hard to understand: the variables of the optimizer (η, B, T) and architecture (N, W, H, S) are relatively clear numbers, whereas for data, apart from the data size D which can be quantified relatively accurately, dimensions like specialized domains, quality, and modality inherently lack clear boundaries and cannot be accurately depicted by a single scalar.

In addition, from the notation $L(\mathcal{E}|\mathcal{D}, \mathcal{A}, \mathcal{O})$, we can see that $\mathcal{D}, \mathcal{A}, \mathcal{O}$ are all its conditions. That is to say, changes in optimizer variables and architecture variables will in principle also affect the Scaling Law on the data side. Various variables interact with each other, making it difficult for us to clearly study the dependency laws of data under a "clean" setting.

Even more critically, if we think carefully, we will find that the proposition of "Data Scaling Law" itself is confusing: in order to measure the effect fairly, we need to first prepare a sufficiently representative test set, and then "pretend" not to know what the test set looks like, trying to study the "Scaling Law" on the training data to make the test set perform as well as possible. This process seems quite nonsensical no matter how you look at it.

Brief Summary

In this section, we briefly introduced the Scaling Law of the data gap, mainly introducing the parameter of data size D, considering the positive effect of increasing data size and the overfitting risk brought by Multi-Epoch. Combined with the optimization error, we derived the result for the optimal number of Epochs. But overall, there are still many confusing aspects about the Scaling Law on the data side that urgently require deeper thought.

Power Law

So far, we have assumed all variation laws to be in the form of sums or products of "power laws", and then deduced some optimal choices through the "different-power inequality". Looking back now, there are two questions worth pondering: 1. Why do we assume the variation law to be a power-law form? 2. When we introduce other conditions, why do we only consider changes in the coefficients of the power law, but not changes in the exponents?

The Question of Power Law

Why power laws? Many researchers have tried to provide more "essential" explanations, but the author feels that many attempts merely swap one assumption for another without substantial change, such as "Deriving the Scaling Law of Models Based on the Quantization Hypothesis".

The author believes the most direct explanation is: when we determine that a variable's dependency is monotonically decreasing and we only care about asymptotic behavior, there are not many functions to choose from. Generally, there are only two classes: power functions and exponential functions. Exponential functions decay too fast—in the language of distributions, they are "short-tailed"—which means that with a slight investment of some resource, the return quickly hits a ceiling, which is inconsistent with our "intuition" of the real world. Power functions, on the other hand, are "long-tailed"; they decay more slowly and better describe the phenomenon of "continuous investment, continuous improvement".

Speaking more philosophically, if this world were dominated by exponential functions, it would be too boring: various things would quickly reach their ceilings, and various investments would quickly lose meaning. It is precisely because power laws dominate that the story of Scale Up becomes vivid.

Another perspective is that power laws are equivalent to "scale invariance": $f(\lambda x) = \lambda^{-\gamma} f(x)$. The function looks the same at any scale, and the very meaning of the word "Scaling" is cross-scale laws—the only elementary function that can play this role seems to be the power law. From a practical point of view: a power law is a straight line under log-log coordinates, making it easier to fit and visualize. Furthermore, the existence of the "different-power inequality" ensures that its minimum point and minimum value are still power laws. These are all important reasons why it can become an "empirical law".

Of course, the power law is not the only answer. In the "Data Section", there are papers that use exponential functions to describe the value decay of Multi-Epoch. One could say that the power law is a basic law that can be tried first in most cases. If we encounter problems that are particularly difficult to fit, we can then consider making adjustments. Or, if the interval we want to extrapolate to is not large, the monotonicity of the function might be more important, and the differences between different function forms might not be that significant.

The Question of Coefficients

The second question first appeared in the "Architecture Section". We assumed that changes in conditions at most affect the coefficients of the power law. For example, architectural changes only affect the coefficients of the optimization error power law, and changes in MoE sparsity only affect the power law of the architecture error, while keeping the exponents unchanged. What is the rationale for this? Is there a more essential truth behind it?

First is mathematical pragmatism: if the exponents also change with conditions, then

the power-law form itself is destroyed—the result is no longer a power law, the different-power inequality no longer applies, and the entire framework loses its simplest operability. Letting only the coefficients change is the minimal generalization while keeping the form unchanged, making it an assumption worth trying first.

Second is a physical analogy: in statistical physics, the critical exponents of phase transitions are universal. Different materials within the same universality class share the same set of exponents, and material details only change the non-universal prefactors. Similarly, the exponents of the Scaling Law can also be understood as the "difficulty" of the problem itself—determined by the data distribution and the task; whereas the coefficient is the "engineering level" of the solution—it changes with improvements in the optimizer and architecture. This is essentially doing better on the same problem, so what should change is the coefficient in front of the power law, not the exponent.

Thinking in reverse, if a finite engineering improvement could really change the exponent, then in an asymptotic sense, the relative advantage it brings would be magnified unboundedly as the scale grows—finite investment yielding unbounded relative returns is equivalent to an exponential level of progress appearing out of thin air, which is usually unrealistic. Unless this improvement changes the problem itself, but that is no longer an "engineering improvement on the same problem", but rather a different problem.

Of course, this is more of a conjecture by the author based on simplicity and self-consistency. It can be considered an open question, and discussions are welcome.

Conclusion

After reading this blog post, some readers might feel that "it seems like everything was said, yet it seems like nothing was said."—we did not prove any theorems; we simply wrote down a triple decomposition, wrote down a bunch of power-law assumptions based on experience, and then did some basic analysis using optimization methods.

We attempted to use this approach to find the commonalities among various Scaling Law results and clarify their interaction mechanisms. Fortunately, this article did yield some heuristic results. However, due to the author's limited level, most of the content can only be introduced superficially, especially the "Data Section", where the author's understanding is still at a very shallow stage.

Going through the entire derivation process, it personally feels a bit like "dimensional analysis" in physics: it is not as rigorous as a first-principles derivation, but rather relies on intuition and monotonicity to guess the power-law forms of each term, further derives the laws of the optimal solution, and finally compares them with classic results for simplification or correction.

I hope the perspective of this article can provide some help for readers to understand and apply Scaling Law.