Title: Neural scaling law

URL: https://en.wikipedia.org/wiki/Neural_scaling_law

PageID: 73713333

Categories: Category:Artificial intelligence, Category:Artificial neural networks, Category:Deep learning, Category:Power laws, Category:Statistical laws

---

In machine learning , a neural scaling law is an empirical scaling law that describes how neural network performance changes as key factors are scaled up or down. These factors typically include the number of parameters, training dataset size, [1] [2] and training cost. Some models also exhibit performance gains by scaling inference through increased test-time compute (TTC), extending neural scaling laws beyond training to the deployment phase. [3]

Introduction

In general, a deep learning model can be characterized by four parameters: model size, training dataset size, training cost, and the post-training error rate (e.g., the test set error rate). Each of these variables can be defined as a real number, usually written as N , D , C , L {\displaystyle N,D,C,L} (respectively: parameter count, dataset size, computing cost, and loss).

A neural scaling law is a theoretical or empirical statistical law between these parameters. There are also other parameters with other scaling laws.

Size of the model

In most cases, the model's size is simply the number of parameters. However, one complication arises with the use of sparse models, such as mixture-of-expert models . [4] With sparse models, during inference, only a fraction of their parameters are used. In comparison, most other kinds of neural networks, such as transformer models, always use all their parameters during inference.

Size of the training dataset

The size of the training dataset is usually quantified by the number of data points within it. Larger training datasets are typically preferred, as they provide a richer and more diverse source of information from which the model can learn. This can lead to improved generalization performance when the model is applied to new, unseen data. [5] However, increasing the size of the training dataset also increases the computational resources and time required for model training.

With the "pretrain, then finetune" method used for most large language models , there are two kinds of training dataset: the pretraining dataset and the finetuning dataset. Their sizes have different effects on model performance. Generally, the finetuning dataset is less than 1% the size of pretraining dataset. [6]

In some cases, a small amount of high quality data suffices for finetuning, and more data does not necessarily improve performance. [6]

Cost of training

Training cost is typically measured in terms of time (how long it takes to train the model) and computational resources (how much processing power and memory are required). It is important to note that the cost of training can be significantly reduced with efficient training algorithms, optimized software libraries, and parallel computing on specialized hardware such as GPUs or TPUs .

The cost of training a neural network model is a function of several factors, including model size, training dataset size, the training algorithm complexity , and the computational resources available. [5] In particular, doubling the training dataset size does not necessarily double the cost of training, because one may train the model for several times over the same dataset (each being an " epoch ").

Performance

The performance of a neural network model is evaluated based on its ability to accurately predict the output given some input data. Common metrics for evaluating model performance include: [5]

Negative log-likelihood per token (logarithm of perplexity) for language modeling ;

Accuracy , precision, recall , and F1 score for classification tasks;

Mean squared error (MSE) or mean absolute error (MAE) for regression tasks;

Elo rating in a competition against other models, such as gameplay [9] or preference by a human judge . [10]

Performance can be improved by using more data, larger models, different training algorithms, regularizing the model to prevent overfitting , and early stopping using a validation set.

When the performance is a number bounded within the range of [0 , 1] {\displaystyle [0,1]} , such as accuracy, precision, etc., it often scales as a sigmoid function of cost, as seen in the figures.

Examples

(Hestness, Narang, et al, 2017)

The 2017 paper [2] is a common reference point for neural scaling laws fitted by statistical analysis on experimental data. Previous works before the 2000s, as cited in the paper, were either theoretical or orders of magnitude smaller in scale. Whereas previous works generally found the scaling exponent to scale like L ∝ D − α {\displaystyle L\propto D^{-\alpha }} , with α ∈ { 0.5 , 1 , 2 } {\displaystyle \alpha \in \{0.5,1,2\}} , the paper found that α ∈ [0.07 , 0.35] {\displaystyle \alpha \in [0.07,0.35]} .

Of the factors they varied, only task can change the exponent α {\displaystyle \alpha } . Changing the architecture optimizers, regularizers, and loss functions, would only change the proportionality factor, not the exponent. For example, for the same task, one architecture might have L = 1000 D − 0.3 {\displaystyle L=1000D^{-0.3}} while another might have L = 500 D − 0.3 {\displaystyle L=500D^{-0.3}} . They also found that for a given architecture, the number of parameters necessary to reach lowest levels of loss, given a fixed dataset size, grows like N ∝ D β {\displaystyle N\propto D^{\beta }} for another exponent β {\displaystyle \beta } .

They studied machine translation with LSTM (α ~ 0.13 {\displaystyle \alpha \sim 0.13}), generative language modelling with LSTM (α ∈ [0.06 , 0.09] , β ≈ 0.7 {\displaystyle \alpha \in [0.06,0.09],\beta \approx 0.7}), ImageNet classification with ResNet (α ∈ [0.3 , 0.5] , β ≈ 0.6 {\displaystyle \alpha \in [0.3,0.5],\beta \approx 0.6}), and speech recognition with two hybrid (LSTMs complemented by either CNNs or an attention decoder) architectures (α ≈ 0.3 {\displaystyle \alpha \approx 0.3}).

(Henighan, Kaplan, et al, 2020)

A 2020 analysis [11] studied statistical relations between C , N , D , L {\displaystyle C,N,D,L} over a wide range of values and found similar scaling laws, over the range of N ∈ [10 3 , 10 9] {\displaystyle N\in [10^{3},10^{9}]} , C ∈ [10 12 , 10 21] {\displaystyle C\in [10^{12},10^{21}]} , and over multiple modalities (text, video, image, text to image, etc.). [11]

In particular, the scaling laws it found are (Table 1 of [11]):

For each modality, they fixed one of the two C , N {\displaystyle C,N} , and varying the other one (D {\displaystyle D} is varied along using D = C / 6 N {\displaystyle D=C/6N}), the achievable test loss satisfies L = L 0 + (x 0 x) α {\displaystyle L=L_{0}+\left({\frac {x_{0}}{x}}\right)^{\alpha }} where x {\displaystyle x} is the varied variable, and L 0 , x 0 , α {\displaystyle L_{0},x_{0},\alpha } are parameters to be found by statistical fitting. The parameter α {\displaystyle \alpha } is the most important one. When N {\displaystyle N} is the varied variable, α {\displaystyle \alpha } ranges from 0.037 {\displaystyle 0.037} to 0.24 {\displaystyle 0.24} depending on the model modality. This corresponds to the α = 0.34 {\displaystyle \alpha =0.34} from the Chinchilla scaling paper. When C {\displaystyle C} is the varied variable, α {\displaystyle \alpha } ranges from 0.048 {\displaystyle 0.048} to 0.19 {\displaystyle 0.19} depending on the model modality. This corresponds to the β = 0.28 {\displaystyle \beta =0.28} from the Chinchilla scaling paper.

When N {\displaystyle N} is the varied variable, α {\displaystyle \alpha } ranges from 0.037 {\displaystyle 0.037} to 0.24 {\displaystyle 0.24} depending on the model modality. This corresponds to the α = 0.34 {\displaystyle \alpha =0.34} from the Chinchilla scaling paper.

When C {\displaystyle C} is the varied variable, α {\displaystyle \alpha } ranges from 0.048 {\displaystyle 0.048} to 0.19 {\displaystyle 0.19} depending on the model modality. This corresponds to the β = 0.28 {\displaystyle \beta =0.28} from the Chinchilla scaling paper.

Given fixed computing budget, optimal model parameter count is consistently around N o p t (C) = (C 5 × 10 − 12 petaFLOP-day) 0.7 = 9.0 × 10 − 7 C 0.7 {\displaystyle N_{opt}(C)=\left({\frac {C}{5\times 10^{-12}{\text{petaFLOP-day}}}}\right)^{0.7}=9.0\times 10^{-7}C^{0.7}} The parameter 9.0 × 10 − 7 {\displaystyle 9.0\times 10^{-7}} varies by a factor of up to 10 for different modalities. The exponent parameter 0.7 {\displaystyle 0.7} varies from 0.64 {\displaystyle 0.64} to 0.75 {\displaystyle 0.75} for different modalities. This exponent corresponds to the ≈ 0.5 {\displaystyle \approx 0.5} from the Chinchilla scaling paper.

It's "strongly suggested" (but not statistically checked) that D o p t (C) ∝ N o p t (C) 0.4 ∝ C 0.28 {\displaystyle D_{opt}(C)\propto N_{opt}(C)^{0.4}\propto C^{0.28}} . This exponent corresponds to the ≈ 0.5 {\displaystyle \approx 0.5} from the Chinchilla scaling paper.

The scaling law of L = L 0 + (C 0 / C) 0.048 {\displaystyle L=L_{0}+(C_{0}/C)^{0.048}} was confirmed during the training of GPT-3 (Figure 3.1 [12]).

Chinchilla scaling (Hoffmann, et al, 2022)

One particular scaling law (" Chinchilla scaling ") states that, for a large language model (LLM) autoregressively trained for one epoch, with a cosine learning rate schedule, we have: [14] { C = C 0 N D L = A N α + B D β + L 0 {\displaystyle {\begin{cases}C=C_{0}ND\\L={\frac {A}{N^{\alpha }}}+{\frac {B}{D^{\beta }}}+L_{0}\end{cases}}} where the variables are

C {\displaystyle C} is the cost of training the model, in FLOPS .

N {\displaystyle N} is the number of parameters in the model.

D {\displaystyle D} is the number of tokens in the training set.

L {\displaystyle L} is the average negative log-likelihood loss per token (nats /token), achieved by the trained LLM on the test dataset. L 0 {\displaystyle L_{0}} represents the loss of an ideal generative process on the test data A N α {\displaystyle {\frac {A}{N^{\alpha }}}} captures the fact that a Transformer language model with N {\displaystyle N} parameters underperforms the ideal generative process B D β {\displaystyle {\frac {B}{D^{\beta }}}} captures the fact that the model trained on D {\displaystyle D} tokens underperforms the ideal generative process

L 0 {\displaystyle L_{0}} represents the loss of an ideal generative process on the test data

A N α {\displaystyle {\frac {A}{N^{\alpha }}}} captures the fact that a Transformer language model with N {\displaystyle N} parameters underperforms the ideal generative process

B D β {\displaystyle {\frac {B}{D^{\beta }}}} captures the fact that the model trained on D {\displaystyle D} tokens underperforms the ideal generative process

and the statistical parameters are

C 0 = 6 {\displaystyle C_{0}=6} , meaning that it costs 6 FLOPs per parameter to train on one token. This is estimated by Kaplan et al. [15] Note that training cost is much higher than inference cost, as training entails both forward and backward passes , whereas inference costs 1 to 2 FLOPs per parameter to infer on one token.

α = 0.34 , β = 0.28 , A = 406.4 , B = 410.7 , L 0 = 1.69 {\displaystyle \alpha =0.34,\beta =0.28,A=406.4,B=410.7,L_{0}=1.69} .

Although Besiroglu et al. [16] claims that the statistical estimation is slightly off, and should be α = 0.35 , β = 0.37 , A = 482.01 , B = 2085.43 , L 0 = 1.82 {\displaystyle \alpha =0.35,\beta =0.37,A=482.01,B=2085.43,L_{0}=1.82} .

The statistical laws were fitted over experimental data with N ∈ [7 × 10 7 , 1.6 × 10 10] , D ∈ [5 × 10 9 , 5 × 10 11] , C ∈ [10 18 , 10 24] {\displaystyle N\in [7\times 10^{7},1.6\times 10^{10}],D\in [5\times 10^{9},5\times 10^{11}],C\in [10^{18},10^{24}]} .

Since there are 4 variables related by 2 equations, imposing 1 additional constraint and 1 additional optimization objective allows us to solve for all four variables. In particular, for any fixed C {\displaystyle C} , we can uniquely solve for all 4 variables that minimizes L {\displaystyle L} . This provides us with the optimal D o p t (C) , N o p t (C) {\displaystyle D_{opt}(C),N_{opt}(C)} for any fixed C {\displaystyle C} : N o p t (C) = G (C 6) a , D o p t (C) = G − 1 (C 6) b , where G = (α A β B) 1 α + β , a = β α + β , and b = α α + β . {\displaystyle N_{opt}(C)=G\left({\frac {C}{6}}\right)^{a},\quad D_{opt}(C)=G^{-1}\left({\frac {C}{6}}\right)^{b},\quad {\text{ where }}\quad G=\left({\frac {\alpha A}{\beta B}}\right)^{\frac {1}{\alpha +\beta }},\quad a={\frac {\beta }{\alpha +\beta }}{\text{, and }}b={\frac {\alpha }{\alpha +\beta }}{\text{. }}} Plugging in the numerical values, we obtain the "Chinchilla efficient" model size and training dataset size, as well as the test loss achievable: { N o p t (C) = 0.6 C 0.45 D o p t (C) = 0.3 C 0.55 L o p t (C) = 1070 C − 0.154 + 1.7 {\displaystyle {\begin{cases}N_{opt}(C)=0.6\;C^{0.45}\\D_{opt}(C)=0.3\;C^{0.55}\\L_{opt}(C)=1070\;C^{-0.154}+1.7\end{cases}}} Similarly, we may find the optimal training dataset size and training compute budget for any fixed model parameter size, and so on.

There are other estimates for "Chinchilla efficient" model size and training dataset size. The above is based on a statistical model of L = A N α + B D β + L 0 {\displaystyle L={\frac {A}{N^{\alpha }}}+{\frac {B}{D^{\beta }}}+L_{0}} . One can also directly fit a statistical law for D o p t (C) , N o p t (C) {\displaystyle D_{opt}(C),N_{opt}(C)} without going through the detour, for which one obtains: { N o p t (C) = 0.1 C 0.5 D o p t (C) = 1.7 C 0.5 {\displaystyle {\begin{cases}N_{opt}(C)=0.1\;C^{0.5}\\D_{opt}(C)=1.7\;C^{0.5}\end{cases}}} or as tabulated:

Discrepancy

The Chinchilla scaling law analysis for training transformer language models suggests that for a given training compute budget (C {\displaystyle C}), to achieve the minimal pretraining loss for that budget, the number of model parameters (N {\displaystyle N}) and the number of training tokens (D {\displaystyle D}) should be scaled in equal proportions, N o p t (C) ∝ C 0.5 , D o p t (C) ∝ C 0.5 {\displaystyle N_{opt}(C)\propto C^{0.5},D_{opt}(C)\propto C^{0.5}} .

This conclusion differs from analysis conducted by Kaplan et al., [15] which found that N {\displaystyle N} should be increased more quickly than D {\displaystyle D} , N o p t (C) ∝ C 0.73 , D o p t (C) ∝ C 0.27 {\displaystyle N_{opt}(C)\propto C^{0.73},D_{opt}(C)\propto C^{0.27}} .

This discrepancy can primarily be attributed to the two studies using different methods for measuring model size. Kaplan et al.: [17]

did not count the parameters in the token embedding layer, which when analyzed at smaller model sizes leads to biased coefficients;

studied smaller models than the Chinchilla group, magnifying the effect;

assumed that L ∞ = 0 {\displaystyle L_{\infty }=0} .

Secondary effects also arise due to differences in hyperparameter tuning and learning rate schedules. Kaplan et al.: [18]

used a warmup schedule that was too long for smaller models, making them appear less efficient;

did not fully tuning optimization hyperparameters.

Beyond Chinchilla scaling

As Chinchilla scaling has been the reference point for many large-scaling training runs, there had been a concurrent effort to go "beyond Chinchilla scaling", meaning to modify some of the training pipeline in order to obtain the same loss with less effort, or deliberately train for longer than what is "Chinchilla optimal".

Usually, the goal is to make the scaling law exponent larger, which means the same loss can be trained for much less compute. For instance, filtering data can make the scaling law exponent

larger. [19]

Another strand of research studies how to deal with limited data, as according to Chinchilla scaling laws, the training dataset size for the largest language models already approaches what is available on the internet. [20] found that augmenting the dataset with a mix of "denoising objectives" constructed from the dataset improves performance. [21] studies optimal scaling when all available data is already exhausted (such as in rare languages), so one must train multiple epoches over the same dataset (whereas Chinchilla scaling requires only one epoch). The Phi series of small language models were trained on textbook-like data generated by large language models, for which data is only limited by amount of compute available. [22]

Chinchilla optimality was defined as "optimal for training compute", whereas in actual production-quality models, there will be a lot of inference after training is complete. "Overtraining" during training means better performance during inference. [23] LLaMA models were overtrained for this reason. Subsequent studies discovered scaling laws in the overtraining regime, for dataset sizes up to 32x more than Chinchilla-optimal. [24]

Broken neural scaling laws (BNSL)

A 2022 analysis [25] found that many scaling behaviors of artificial neural networks follow a smoothly broken power law functional form:

y = a + (b x – c 0) ∏ i = 1 n (1 + (x d i) 1 / f i) – c i ∗ f i {\displaystyle y=a+{\bigg (}bx^{-c_{0}}{\bigg)}\prod _{i=1}^{n}\left(1+\left({\frac {x}{d_{i}}}\right)^{1/f_{i}}\right)^{-c_{i}*f_{i}}}

in which x {\displaystyle x} refers to the quantity being scaled (i.e. C {\displaystyle C} , N {\displaystyle N} , D {\displaystyle D} , number of training steps, number of inference steps, or model input size) and y {\displaystyle y} refers to the downstream (or upstream) performance evaluation metric of interest (e.g. prediction error, cross entropy , calibration error, AUROC , BLEU score percentage , F1 score , reward, Elo rating , solve rate, or FID score) in zero-shot , prompted , or fine-tuned settings. The parameters a , b , c 0 , c 1 . . . c n , d 1 . . . d n , f 1 . . . f n {\displaystyle a,b,c_{0},c_{1}...c_{n},d_{1}...d_{n},f_{1}...f_{n}} are found by statistical fitting.

On a log–log plot , when f i {\displaystyle f_{i}} is not too large and a {\displaystyle a} is subtracted out from the y-axis, this functional form looks like a series of linear segments connected by arcs; the n {\displaystyle n} transitions between the segments are called "breaks", hence the name broken neural scaling laws (BNSL) .

The scenarios in which the scaling behaviors of artificial neural networks were found to follow this functional form include large-scale vision , language , audio, video, diffusion , generative modeling , multimodal learning , contrastive learning , AI alignment , AI capabilities, robotics , out-of-distribution (OOD) generalization, continual learning, transfer learning , uncertainty estimation / calibration , out-of-distribution detection , adversarial robustness , distillation , sparsity, retrieval, quantization, pruning , fairness , molecules, computer programming/coding, math word problems, arithmetic, emergent abilities , double descent , supervised learning , unsupervised / self-supervised learning, and reinforcement learning (single agent and multi-agent).

The architectures for which the scaling behaviors of artificial neural networks were found to follow this functional form include residual neural networks , transformers , MLPs , MLP-mixers , recurrent neural networks , convolutional neural networks , graph neural networks , U-nets , encoder-decoder (and encoder-only) (and decoder-only) models, ensembles (and non-ensembles), MoE (mixture of experts) (and non-MoE) models, and sparse pruned (and non-sparse unpruned) models.

Inference scaling

Other than scaling up training compute, one can also scale up inference compute (or "test-time compute" [3]). As an example, the Elo rating of AlphaGo improves steadily as it is allowed to spend more time on its Monte Carlo Tree Search per play. [26] : Fig 4 For AlphaGo Zero , increasing Elo by 120 requires either 2x model size and training, or 2x test-time search. [27] Similarly, a language model for solving competition-level coding challenges, AlphaCode, consistently improved (log-linearly) in performance with more search time. [28]

For Hex , 10x training-time compute trades for 15x test-time compute. [9] For Libratus for heads up no-limit Texas hold 'em , and Cicero for Diplomacy , and many other abstract games of partial information, inference-time searching improves performance at a similar tradeoff ratio, for up to 100,000x effective increase in training-time compute. [27]

In 2024, the OpenAI o1 report documented that o1's performance consistently improved with both increased train-time compute and test-time compute, and gave numerous examples of test-time compute scaling in mathematics, scientific reasoning, and coding tasks. [29] [30]

One method for scaling up test-time compute is process-based supervision , where a model generates a step-by-step reasoning chain to answer a question, and another model (either human or AI) provides a reward score on some of the intermediate steps, not just the final answer. Process-based supervision can be scaled arbitrarily by using synthetic reward score without another model, for example, by running Monte Carlo rollouts and scoring each step in the reasoning according to how likely it leads to the right answer. Another method is by revision models , which are models trained to solve a problem multiple times, each time revising the previous attempt. [31]

Other examples

Vision transformers

Vision transformers , similar to language transformers, exhibit scaling laws. A 2022 research trained vision transformers, with parameter counts $N \in [5 \times 10^6, 2 \times 10^9]$, on image sets of sizes $D \in [3 \times 10^7, 3 \times 10^9]$, for computing $C \in [0.2, 10^4]$ (in units of TPUv3-core-days). [32]

After training the model, it is finetuned on ImageNet training set. Let L be the error probability of the finetuned model classifying ImageNet test set. They found $\min_{N,D} L = 0.09 + 0.26 (C + 0.01)^{0.35}$.

Neural machine translation

Ghorbani, Behrooz et al. [33] studied scaling laws for neural machine translation (specifically, English as source, and German as target) in encoder-decoder Transformer models, trained until convergence on the same datasets (thus they did not fit scaling laws for computing cost C or dataset size D). They varied $N \in [10^8, 3.5 \times 10^9]$ They found three results:

L is a scaling law function of N_E, N_D , where N_E, N_D are encoder and decoder parameter count. It is not simply a function of total parameter count $N = N_E + N_D$. The function has form $L(N_e, N_d) = \alpha \left(\frac{\bar{N}_e}{N_e}\right)^{p_e} \left(\frac{\bar{N}_d}{N_d}\right)^{p_d} + L_\infty$, where $\alpha, p_e, p_d, L_\infty, \bar{N}_e, \bar{N}_d$ are fitted parameters. They found that $N_d / N \approx 0.55$ minimizes loss if N is held fixed.

L "saturates" (that is, it reaches L_∞) for smaller models when the training and testing datasets are "source-natural" than "target-natural". A "source-natural" data point means a pair of English-German sentences, and the model is asked to translate the English sentence into German, and the English sentence is written by a natural English writer, while the German sentence is translated from the English sentence by a machine translator. [34] To construct the two kinds of datasets, the authors collected natural English and German sentences online, then used machine translation to generate their translations.

As models grow larger, models trained on source-original datasets can achieve low loss but bad BLEU score . In contrast, models trained on target-original datasets achieve low loss and good BLEU score in tandem (Figure 10, 11 [33]).

The authors hypothesize that source-natural datasets have uniform and dull target sentences, and so a model that is trained to predict the target sentences would quickly overfit.

[35] trained Transformers for machine translations with sizes N ∈ [4 × 10 5 , 5.6 × 10 7] {\displaystyle N\in [4\times 10^{5},5.6\times 10^{7}]} on dataset sizes D ∈ [6 × 10 5 , 6 × 10 9] {\displaystyle D\in [6\times 10^{5},6\times 10^{9}]} . They found the Kaplan et al. (2020) [15] scaling law applied to machine translation: L (N , D) = [(N C N) α N α D + D C D] α D {\displaystyle L(N,D)=\left[\left({\frac {N_{C}}{N}}\right)^{\frac {\alpha _{N}}{\alpha _{D}}}+{\frac {D_{C}}{D}}\right]^{\alpha _{D}}} . They also found the BLEU score scaling as B L E U ≈ C e − k L {\displaystyle BLEU\approx Ce^{-kL}} .

Transfer learning

Hernandez, Danny et al. [36] studied scaling laws for transfer learning in language models. They trained a family of Transformers in three ways:

pretraining on English, finetuning on Python

pretraining on an equal mix of English and Python, finetuning on Python

training on Python

The idea is that pretraining on English should help the model achieve low loss on a test set of Python text. Suppose the model has parameter count N {\displaystyle N} , and after being finetuned on D F {\displaystyle D_{F}} Python tokens, it achieves some loss L {\displaystyle L} . We say that its "transferred token count" is D T {\displaystyle D_{T}} , if another model with the same N {\displaystyle N} achieves the same L {\displaystyle L} after training on D F + D T {\displaystyle D_{F}+D_{T}} Python tokens.

They found D T = 1.9 e 4 (D F) .18 (N) .38 {\displaystyle D_{T}=1.9e4\left(D_{F}\right)^{.18}(N)^{.38}} for pretraining on English text, and D T = 2.1 e 5 (D F) .096 (N) .38 {\displaystyle D_{T}=2.1e5\left(D_{F}\right)^{.096}(N)^{.38}} for pretraining on English and non-Python code.

Precision

Kumar et al. [37] study scaling laws for numerical precision in the training of language models. They train a family of language models with weights, activations, and KV cache in varying numerical precision in both integer and floating-point type to measure the effects on loss as a function of precision. For training, their scaling law accounts for lower precision by wrapping the effects of precision into an overall "effective parameter count" that governs loss scaling, using the parameterization N ■ N eff (P) = N (1 − e − P / γ) {\displaystyle N\mapsto N_{\text{eff}}(P)=N(1-e^{-P/\gamma })} . This illustrates how training in lower precision degrades performance by reducing the true capacity of the model in a manner that varies exponentially with bits.

For inference, they find that extreme overtraining of language models past Chinchilla-optimality can lead to models being more sensitive to quantization, a standard technique for efficient deep learning. This is demonstrated by observing that the degradation in loss due to weight quantization increases as an approximate power law in the token/parameter ratio D / N {\displaystyle D/N} seen during pretraining, so that models pretrained on extreme token budgets can perform worse in terms of validation loss than those trained on more modest token budgets if post-training quantization is applied. Other work examining the effects of overtraining include Sardana et al. [38] and Gadre et al. [39]

Densing laws

Xiao et al. [8] considered the parameter efficiency ("density") of models over time. The idea is that over time, researchers would discover models that use their parameters more efficiently, in that models with the same performance can have fewer parameters.

A model can have an actual parameter count N {\displaystyle N} , defined as the actual number of parameters in the model, and an "effective" parameter count N ^ {\displaystyle {\hat {N}}} , defined as how many parameters it would have taken a previous well-known model to reach he same performance on some benchmarks, such as MMLU . N ^ {\displaystyle {\hat {N}}} is not measured directly, but rather by measuring the actual model performance S {\displaystyle S} , then plugging it

back to a previously fitted scaling law, such as the Chinchilla scaling law, to obtain what N ^ {\displaystyle {\hat {N}}} would be required to reach that performance S {\displaystyle S} , according to that previously fitted scaling laws.

A densing law states that ln ■ (N ^ N) m a x = A t + B {\displaystyle \ln \left({\frac {\hat {N}}{N}}\right)_{max}=At+B} , where t {\displaystyle t} is real-world time, measured in days.

See also

Large language model

Foundation model

Artificial general intelligence

References