Reading Summary Week 8

Aakash Agrawal
HDSI
University of California San Diego
San Diego, CA, 92092
aaa015@ucsd.edu

1 Training Compute-Optimal Large Language Models

What is compute optimal? It means creating a model with the lowest loss for a given, fixed amount of compute by striking a balance between model size, dataset size, training duration, and hardware efficiency. Chinchilla's focus is on training models that are **compute optimal**.

The authors argue that most large language models today are **significantly undertrained**, leading to suboptimal performance given their compute budgets. They highlight that the focus on scaling model size has resulted in inefficient training, where models fail to reach their full potential.

Pre-training a large language model incurs significant compute costs, but downstream **fine-tuning** and **inference** also contribute substantially to overall compute usage. Compute-optimal models like Chinchilla reduce both memory footprint and inference costs, making them more efficient in deployment, thereby amortizing the initial energy cost across many predictions. Hence, the benefits of a compute optimal model extend far beyond the immediate benefits of improved performance.

Prior works on understanding the scaling behavior have used fixed values for **hyperparameters** like the learning rate schedules, which have been prohibitive in modeling their impact on the loss. Since training such large models is a one-time, resource-intensive process, accurately determining optimal hyperparameters for a given compute budget is crucial to maximizing performance.

1.1 Estimating the optimal parameter/training tokens allocation

The gist of the paper is answering the question: `Given a fixed FLOPs budget, how should one trade-off model size and the number of training tokens?` It's basically an optimization problem where we fix one variable FLOPs (C) and try to find the optimal values for the size of the model N and the number of training tokens D (factors that influence the amount of FLOPs a model uses).

The authors train a range of models, varying both model size and the number of training tokens, and use the resulting training curves to develop an empirical estimator for optimal scaling. They propose three different methods to estimate the scaling behavior between model size and data, with similar results across all three, suggesting that as the compute budget increases, model size and the amount of training data should be increased in approximately equal proportions.

1.1.1 Approach 1: Fixed model size

Here, the authors vary the number of training steps (or training tokens) for a fixed family of models (ranging from 70M to over 10B parameters), training each model for 4 different cosine cycle lengths. This helps extract the envelope of minimal loss per FLOP that can be used to estimate the optimal model size and the optimal number of training tokens for a given compute budget. Fitting power law curves yields $N_{opt} \propto C^a$, $D_{opt} \propto C^b$, $a \approx b \approx 0.5$, suggesting that both the model size and training data should scale at roughly the same rate. This assumption can be used to estimate the optimal model size and number of training tokens for any given amount of compute.

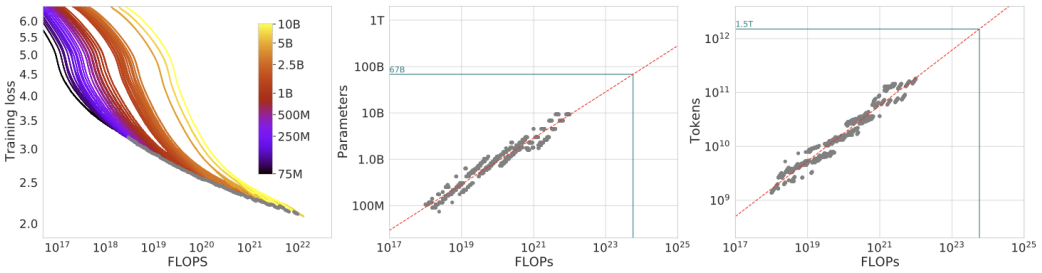

Figure 1: Approach 1: Training curve envelope.

1.1.2 Approach 2: Fixed FLOPs aka IsoFLOPs

Here, the authors vary the model size for a fixed set of 9 different training FLOP counts and compute the validation loss L of each model. For different model sizes, they choose the number of training tokens such that the final FLOPs is a constant. The authors call this an IsoFLOP curve, and it helps find the model size with the lowest loss for a fixed compute budget. Each of these 9 curves above have one value of N_{opt}. The authors then fit a power law $N_{opt} \propto C^a$ to the points (C, N_{opt}). This is where the scaling law appears: when $a = 0.49$, they obtain a very tight fit to the empirically calculated values. Similarly, the optimal number of training tokens $D_{opt} \propto C^b$, $b = 0.51$.

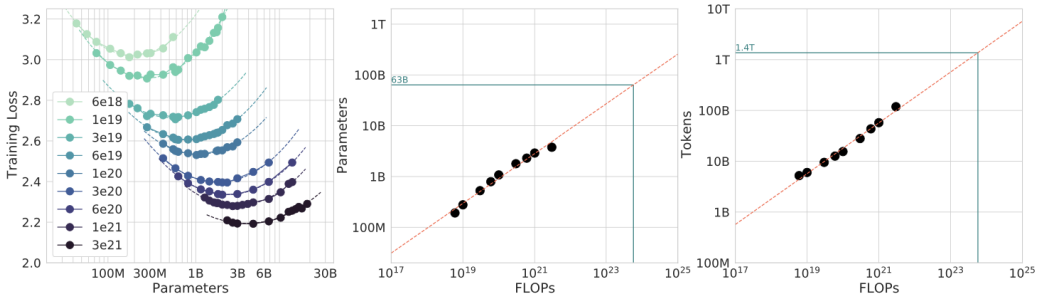

Figure 2: Approach 2: IsoFLOP curves.

1.1.3 Approach 3: Parametric Fitting

The goal here is to take the data from the first two approaches and try to find a function for loss values. The authors propose the following functional form: $\hat{L}(N, D) = E + \frac{A}{N^\alpha} + \frac{B}{D^\beta}$. The first term captures the inherent uncertainty in the data distribution (natural text). Hence, this loss is **irreducible**. The second term decreases as the model capacity grows and represents the fact that a perfectly trained transformer with N parameters underperforms the ideal generative process. The third term represents the loss because the model is not trained to full **convergence**. This represents a predictive model of the loss; the authors approximate N_{opt}, D_{opt} by minimizing the parametric loss $\hat{L}$ under the constraint $FLOPs(C) \approx 6ND$, yielding $a = 0.46, b = 0.54$.

Approach	Coeff. a where $N_{opt} \propto C^a$	Coeff. b where $D_{opt} \propto C^b$
1. Minimum over training curves	0.50 (0.488, 0.502)	0.50 (0.501, 0.512)
2. IsoFLOP profiles	0.49 (0.462, 0.534)	0.51 (0.483, 0.529)
3. Parametric modelling of the loss	0.46 (0.454, 0.455)	0.54 (0.542, 0.543)

Figure 3: Estimated parameter and data scaling with increased training compute. The 10th and 90th percentiles are shown in parenthesis.

1.2 Optimal model scaling and Chinchilla

The authors discover that a **175 billion** parameter model should be trained with a compute budget of 4.41×10^{24} FLOPs and over **4.2 trillion** tokens. For a **280 billion** parameter Gopher-like model, the optimal training setup requires a compute budget of approximately 10^{25} FLOPs and should be trained on **6.8 trillion** tokens.

All three approaches predict that Gopher is significantly oversized and suggest that a smaller model trained on more data would achieve better performance under the same compute budget. To test this hypothesis, the authors train Chinchilla, a **70B**-parameter model trained on **1.4T** tokens, demonstrating that it outperforms Gopher and even larger models across nearly all evaluation tasks. Chinchilla and Gopher are trained with the same number of FLOPs, but differ in model size and the amount of training data.

1.3 Performance Evaluation

The authors perform an extensive evaluation of the Chinchilla model on a collection of downstream tasks and compare it to the performance of other models like Gopher and GPT-3.

- **Language modeling**: Chinchilla outperforms Gopher on all evaluation subsets of the **Pile**. On **Wikitext103**, Chinchilla achieves a perplexity of **7.16** compared to 7.75 for Gopher.
- **Massive Multitask Language Understanding (MMLU)**: Chinchilla achieves an average accuracy of **67.6%** across 57 tasks, compared to Gopher with 60% average accuracy. Chinchilla **underperforms** Gopher on four tasks: college-mathematics, econometrics, moral-scenarios, and formal-logic.
- **Reading comprehension**: On the final word prediction dataset **LAMBADA**, Chinchilla achieves **77.4%** accuracy, compared to 74.5% accuracy from Gopher and 76.6% from MT-NLG 530B.
- **BIG-bench**: Chinchilla achieves an average accuracy of 65.1% across 62 tasks versus 54.4% for Gopher. Chinchilla **underperforms** Gopher on crash-blossom, dark-humor-detection, mathematical-induction and logical-args tasks.
- **Common sense**: On **TruthfulQA**, Chinchilla achieves **43.6%** 0-shot and **66.7%** 10-shot accuracy. In comparison, Gopher achieved only 29.5% 0-shot and 43.7% 10-shot accuracy.
- **Closed-book question answering**: On the **Natural Questions** dataset, Chinchilla sets a new closed-book SOTA with **31.5%** (5-shot) and **35.5%** (64-shot) accuracy, outperforming Gopher's 21% and 28% accuracy, respectively.
- **Gender bias and toxicity**: On **Winogender**, Chinchilla consistently outperforms Gopher in pronoun resolution, particularly on gotcha examples that challenge gender stereotypes. However, performance differences across groups indicate that Chinchilla still **exhibits bias**.

1.4 Some Key Takeaways

- Chinchilla's most important finding is that larger models need much **more data** to fully leverage their capacity.
- Chinchilla proposes that model size and dataset size should be scaled in a **1:1 ratio** to achieve compute optimal model performance.
- Chinchilla showed that smaller models trained on more data could outperform larger models trained on fewer data (e.g., **Gopher** and **GPT-3**) when the total compute is fixed.
- The authors find that setting the **learning rate schedule** to approximately match the number of training tokens results in the best final loss regardless of model size. By aligning the learning rate decay with the number of training tokens, the model can adapt its learning pace more effectively across the entire training process.
- The paper also highlights that, under a fixed compute budget, data is the primary **limiting factor**. Therefore, prioritizing data scaling is essential, but it must be done while considering data **quality** and properly accounting for train-test overlap.

2 Language Models are Few-Shot Learners

The introduction of pre-trained recurrent or transformer-based language models has eliminated the need for task-specific architectures, driving significant progress in tasks like reading comprehension and question-answering. However, achieving strong performance still often requires **task-specific** fine-tuning and large task-specific datasets with thousands of examples. This has several limitations:

- From a practical standpoint, requiring a large labeled dataset for each new task limits the scalability of language models, especially for tasks where collecting supervised data is challenging and must be repeated for every new use case.
- Fine-tuned models may appear to reach human-level performance on benchmarks but often **overstate** true task ability. This is because as the model expressiveness increases and training data narrows, the risk of exploiting **spurious correlations** grows.
- Humans learn most language tasks with just a brief directive or a few examples, without needing large supervised datasets.

One approach to tackling these challenges is **meta-learning**, where a language model develops broad skills and pattern recognition abilities. At inference time, it adapts to new tasks through **in-context learning**, using text input as task specification. The model is conditioned on natural language instructions and/or a few demonstrations and then completes the task by predicting the next sequence. While in-context learning enables a model to absorb many skills and tasks within its parameters, it performs significantly worse than fine-tuning.

In-Context Learning at Scale: This paper shows that scaling up language models greatly improves task-agnostic, few-shot performance, often achieving state-of-the-art results. The authors introduce **GPT-3**, a 175-billion-parameter autoregressive model, and assess its performance across various tasks in different in-context learning settings. GPT-3 is applied to all tasks without gradient updates or fine-tuning, relying entirely on text-based interactions to specify tasks and provide few-shot demonstrations. This is because the focus of the paper is on **task-agnostic** performance.

There are different settings for learning within the context, depending on how much task-specific data they tend to rely on:

1. **Fine-Tuning (FT)**: The model's weights are updated by training on a supervised dataset specific to the task. This approach delivers strong performance on many benchmarks but typically requires thousands to hundreds of thousands of labeled examples and may struggle with generalization to out-of-distribution data.
2. **Few-Shot (FS)**: The model is provided with a few demonstrations at inference time as conditioning without any weight updates. This significantly reduces the need for large task-specific datasets and mitigates the risk of overfitting to a narrow fine-tuning distribution. However, performance remains below state-of-the-art fine-tuned models, and some task-specific data is still necessary.
3. **One-Shot (1S)**: The model receives a single example along with a natural language task description. This setting mirrors how some tasks are communicated to humans but offers limited adaptation compared to few-shot learning.
4. **Zero-Shot (0S)**: Similar to one-shot learning, but without any demonstrations—only a natural language instruction is provided. While this setting offers maximum flexibility, robustness, and avoids spurious correlations, it remains highly challenging for many tasks.

2.1 Training Setup

Model Architecture: The authors use the same architecture as **GPT-2**, including the modified initialization, pre-normalization, and reversible tokenization. Additionally, they use alternating dense and locally banded sparse attention patterns in the layers of the transformer. In order to study the dependence of model size on ML performance, they create 8 different models of varying sizes from 125 million parameters to 175 billion parameters (GPT-3).

Training Dataset: To improve the dataset quality, the authors took three steps with the **Common Crawl** dataset, which contains nearly a trillion words:

- They filtered the dataset based on similarity to high-quality reference corpora.
- They performed fuzzy **deduplication** at the document level to prevent redundancy and ensure the validation set accurately measures overfitting.
- They incorporated known high-quality reference corpora to augment Common Crawl and enhance its diversity.

During training, higher-quality datasets are sampled more frequently, accepting a small amount of overfitting in exchange for improved training data quality. A major concern with language models pre-trained on **extensive internet data**, especially large models capable of **memorizing** vast amounts of content, is the potential **contamination** of downstream tasks. This occurs when test or development sets are inadvertently seen during pre-training, leading to unintended biases in evaluation. The authors note that while models performed slightly better on data overlapping between training and testing, this did not significantly impact the results due to the small fraction of contaminated data.

In contrast to **Chinchilla's scaling laws**, which suggest training large models on extensive token data, the authors advocate for training much larger models on **significantly fewer tokens**. For few-shot learning, the authors evaluate each example in the evaluation set by randomly selecting K examples from the task's training set as conditioning. K can range from 0 to the maximum allowed by the model's context window, typically accommodating 10 to 100 examples.

2.2 Performance and Benchmarks

- **Language Modeling**: The largest GPT-3 model achieved a SOTA result on the Penn Treebank (PTB) dataset, attaining a perplexity of 20.5, which outperformed the previous SOTA by 15 points.
- **Closed Book Question Answering**: On TriviaQA, GPT-3 achieves 64.3% accuracy in zero-shot, 68.0% in one-shot, and 71.2% in few-shot, outperforming fine-tuned T5–11B.
- **Translation**: Few-shot GPT-3 outperforms previous unsupervised NMT work by 5 BLEU when translating into English, reflecting its strength as an English LM.
- **Common Sense Reasoning**: On the PhysicalQA (PIQA) dataset, GPT-3 achieves 81.0% accuracy zero-shot, 80.5% one-shot, and 82.8% few-shot, setting the SOTA in all evaluation settings.
- **Reading Comprehension**: GPT-3's performance varied across datasets, reflecting its strengths and limitations with different answer formats. It performed best on CoQA, nearly matching human performance, but struggled on QuAC, falling below an ELMo baseline due to the need for modeling structured dialog acts and answer span selections.
- **SuperGLUE**: On COPA and ReCoRD GPT-3 achieves near-SOTA performance in the one-shot and few-shot settings. On WSC, performance was still relatively strong. Performance on BoolQ, MultiRC, and RTE was reasonable, roughly matching fine-tuned BERT-Large. Performance on SuperGLUE increases with model size and number of examples in context.

2.3 GPT-3 Limitations

- GPT-3 generates high-quality text but sometimes **repeats ideas**, loses coherence in long passages, **contradicts itself**, and includes irrelevant or nonsensical sentences. Further, GPT-3 struggles with basic physical reasoning, often making errors in tasks requiring an understanding of common sense physics.
- GPT-3 has inherent constraints due to its architecture and training methodology. The authors did not experiment with **bidirectional** architectures or alternative training objectives like **denoising**, which could have potentially improved performance on tasks where GPT-3 struggled.
- Another key limitation of GPT-3 in few-shot learning is uncertainty about whether it truly learns new tasks from scratch during inference or if it is simply recognizing and applying tasks it has already learned during training.

- GPT-3 models are both expensive and inconvenient to use for inference, which could pose a challenge to the practical applicability of models at this scale.
- GPT-3's decisions are difficult to **interpret**, and it carries the **biases** present in the data it was trained on.
- Large pretrained language models are not grounded in other domains, such as video or real-world physical interaction, and therefore lack crucial contextual knowledge about the world. As a result, scaling pure self-supervised prediction may eventually reach its limits, making it likely that augmentation with alternative approaches will be necessary.

2.4 Broader Impacts

GPT-3 enhances the quality of text generation and adaptability compared to smaller models, making it more challenging to distinguish synthetic text from human-written text. As a result, it has the potential to drive both beneficial and harmful applications of language models.

Deliberate Misusage: Powerful language models can enhance harmful activities like misinformation, phishing, and fraud by generating high-quality text. GPT-3's ability to produce text indistinguishable from human writing marks a concerning milestone in this potential for misuse. Threat actors range from low- to moderately-skilled individuals or groups with limited resources, to highly-skilled, well-funded advanced persistent threats (APTs), often state-sponsored, with long-term strategic goals.

Issues of bias, fairness, and representation: Biases in training data can cause models to generate stereotyped or prejudiced content. The authors' analysis suggests that internet-trained models inherit internet-scale biases, reflecting the stereotypes present in their training data.

- **Gender**: The authors analyzed the associations between gender and occupation and found that occupations, in general, are **male-leaning**.
- **Race**: To investigate racial bias in GPT-3, the authors measure the word co-occurrences in the generated samples for prompts seeded with different races. They analyze this bias by exploring how race impacts sentiment. They found that **Asian** had a consistently high sentiment and **Black** had a consistently low sentiment.
- **Religion**: A similar observation was made in regard to different religions. The authors found that the models make associations with religious terms that indicate some propensity to reflect how these terms are sometimes presented in the world. For example, words such as violent, terrorism, and terrorist co-occurred at a greater rate with **Islam**.

Energy Usage: Pre-training LLMs require large amounts of compute that is energy-intensive. GPT-3 175B consumed several **thousand petaflop/s-days** of compute during pre-training, compared to tens of petaflop/s-days for a 1.5B parameter GPT-2 model. The authors also suggest considering how these resources are amortized over the lifetime of a model. These LLMs can be very efficient once trained.