CS-461

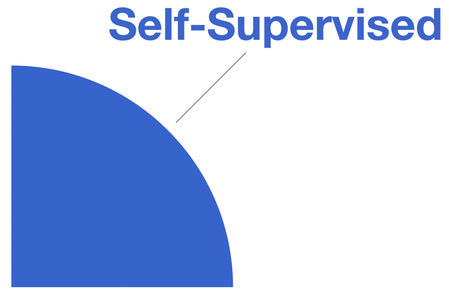
Foundation Models and Generative AI

Exercise Session 1

Xiuying Wei, Fall Semester 2025/26

Contrastive Learning

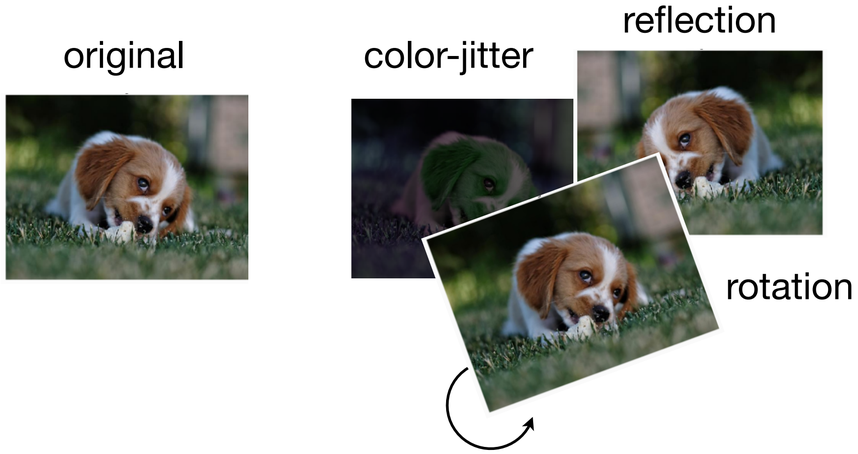
Self-Supervised Learning: Contrastive Learning



Defining Positive and Negative Pairs Without Labels

… in **vision**

+ Positive Pairs

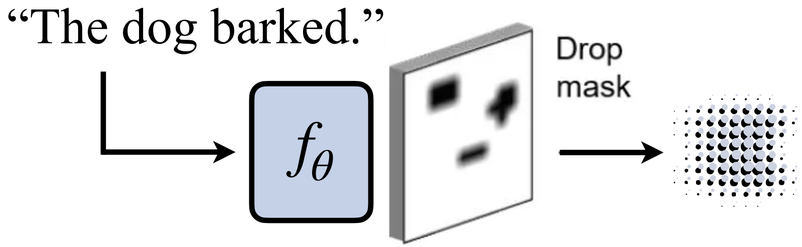


- Negative Pairs

different image

Why? A cropped dog photo is still a dog; different photos are likely different objects.

… in **language**

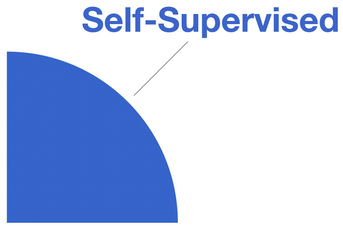


embedding of the same sentence with different dropout masks

“The cat meowed.” different sentence

Why? “The dog barked" ≈ “The dog woofed” ≠ random other sentences

Self-Supervised Learning: Contrastive Learning



InfoNCE Loss

$$\mathcal{L}_{\text{InfoNCE}} = -\mathbb{E}_{c\sim p(c)}\mathbb{E}_{x\sim p(x|c)}\mathbb{E}_{\{x_i\}\sim p(x)}\left[\log \frac{\exp\left(f'(x)^\top f(c)/\tau\right)}{\exp\left(f'(x)^\top f(c)/\tau\right) + \sum_{i=1}^{N-1}\exp\left(f'\left(x_i\right)^\top f(c)/\tau\right)}\right]$$

Derivation: Noise Contrastive Estimation (NCE)

i.e., a method for estimating parameters of a statistical model

Idea: Instead of learning $p_{\text{data}}(x)$, it learns to distinguish data from noise ← contrastive!

How? Logistic regression with binary crossentropy loss

log-odds ratio

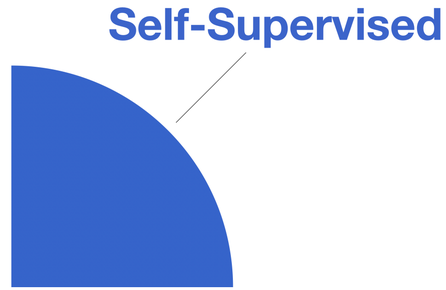
$$s_\theta(x) = \log \frac{p_\theta(x)}{p_n(x)}$$

NCE Loss

$$\mathcal{L}_{\text{NCE}} = -\mathbb{E}_{x\sim p_{\text{data}}}\left[\log \sigma\left(s_\theta(x)\right)\right] - k \cdot \mathbb{E}_{x\sim p_n}\left[\log\left(1-\sigma\left(s_\theta(x)\right)\right)\right]$$

number of noise samples per real sample

Self-Supervised Learning: Contrastive Learning



Generalize binary classification in NCE to N-way classification.
InfoNCE asks 'which one of these N samples is the positive one?'

InfoNCE Loss

$$\mathcal{L}_{\text{InfoNCE}} = -\mathbb{E}_{c\sim p(c)}\mathbb{E}_{x\sim p(x|c)}\mathbb{E}_{\{x_i\}\sim p(x)}\left[\log \frac{\exp\left(f'(x)^\top f(c)/\tau\right)}{\exp\left(f'(x)^\top f(c)/\tau\right) + \sum_{i=1}^{N-1}\exp\left(f'\left(x_i\right)^\top f(c)/\tau\right)}\right]$$

van den Oord et al., (2018)

- encoders f, f'
- context c (anchor) and positive sample x (e.g., its augmentation)
- positive pair: (c, x) where $x \sim p(x \mid c)$ and negative samples: $\left\{x_i\right\}_{i=1}^{N-1}$ where $x_i \sim p(x)$
- score function: $s(x, c) = \exp\left(f'(x)^\top f(c)/\tau\right)$

temperature that controls how "sharp" or "smooth" the similarity distribution is

Exercise 1 · Task 1

→ connection to **information theory**

InfoNCE and Mutual Information

Mutual information: The mutual information between two random variables is a measure of how dependent they are on one another. Mutual information between two random variables c and x is defined as:

$$I(x;c) = \sum_{x,c} p(x,c) \log \frac{p(x|c)}{p(x)}$$

InfoNCE loss: c_t as the context**,** x_{t+k} as the corresponding positive sample, f is the neural network that we learn to model the density ratio, which preserves the mutual information.

$$\mathcal{L}_{\mathrm{N}} = -\mathop{\mathbb{E}}_{X}\left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)}\right] \qquad f_k(x_{t+k}, c_t) \propto \frac{p(x_{t+k}|c_t)}{p(x_{t+k})}$$

Oord, Aaron van den, Yazhe Li, and Oriol Vinyals. "Representation learning with contrastive predictive coding." *arXiv preprint arXiv:1807.03748* (2018).

InfoNCE and Mutual Information

Minimizing the InfoNCE loss maximizes a lower bound on mutual information

$$I(x_{t+k}, c_t) \geq \log(N) - \mathcal{L}_{\mathrm{N}},$$

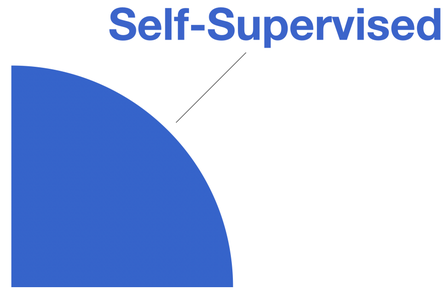
$$\begin{aligned}
\mathcal{L}_{\mathrm{N}}^{\mathrm{opt}} &= -\mathop{\mathbb{E}}_{X} \log \left[\frac{\frac{p(x_{t+k}|c_t)}{p(x_{t+k})}}{\frac{p(x_{t+k}|c_t)}{p(x_{t+k})} + \sum_{x_j \in X_{\mathrm{neg}}} \frac{p(x_j|c_t)}{p(x_j)}} \right] \\
&= \mathop{\mathbb{E}}_{X} \log \left[1 + \frac{p(x_{t+k})}{p(x_{t+k}|c_t)} \sum_{x_j \in X_{\mathrm{neg}}} \frac{p(x_j|c_t)}{p(x_j)} \right] \\
&\approx \mathop{\mathbb{E}}_{X} \log \left[1 + \frac{p(x_{t+k})}{p(x_{t+k}|c_t)} (N-1) \mathop{\mathbb{E}}_{x_j} \frac{p(x_j|c_t)}{p(x_j)} \right]
\end{aligned}$$

$$\begin{aligned}
&= \mathop{\mathbb{E}}_{X} \log \left[1 + \frac{p(x_{t+k})}{p(x_{t+k}|c_t)} (N-1) \right] \\
&\geq \mathop{\mathbb{E}}_{X} \log \left[\frac{p(x_{t+k})}{p(x_{t+k}|c_t)} N \right] \\
&= -I(x_{t+k}, c_t) + \log(N),
\end{aligned}$$

Oord, Aaron van den, Yazhe Li, and Oriol Vinyals. "Representation learning with contrastive predictive coding." *arXiv preprint arXiv:1807.03748* (2018).

Perplexity

Self-Supervised Learning: Autoregressive Learning



Principle: Learn by **predicting next** elements in sequence.

Autoregressive Loss

$$\mathcal{L}_{\mathrm{AR}} = -\mathbb{E}_{x\sim\mathcal{D}}\left[\sum_{t=1}^{T}\log p_\theta\left(x_t \mid x_{<t}\right)\right]$$

Any joint distribution can be factored as a product of conditionals:

chain rule of probability

prefix/history $x_{<t} = (x_1, \ldots, x_{t-1})$

$$p\left(x_1, \ldots, x_T\right) = \prod_{t=1}^{T} p\left(x_t \mid x_{<t}\right)$$

Characteristics:

1. **Exact factorization**, i.e., no approximation, allows to objectively compare models, detect anomalies, etc.
2. **Causal structure** that enforces temporal/sequential dependencies, i.e., foundation to generate sequences, one token at a time, left to right.
3. Deep **connection to information theory**, i.e., ideal AR model achieves Shannon-optimal compression.

Excursion: **Perplexity**

Perplexity (PPL) is the exponential of the average negative log-likelihood per token:

Perplexity

$$\mathrm{PPL} = \exp\left(-\frac{1}{T}\sum_{t=1}^{T}\log p\left(x_t \mid x_{<t}\right)\right) = p\left(x_1, x_2, \ldots, x_N\right)^{-1/T}$$

Intuition: How "surprised" the model is on average

- PPL = 10 → Model is as confused as choosing uniformly from 10 options
- PPL = 1 → Perfect prediction (no surprise)

Historical Origin:

- **Information Theory** (1948): Claude Shannon introduced as $2^{H(X)}$ (exponential of entropy H)
- **Language Modeling** (1970s-80s): Frederick Jelinek and IBM team popularized for speech recognition
- **Modern NLP**: Standard metric since statistical language models replaced rule-based systems

Excursion: **Pseudo-Likelihood**

Autoregressive strategies directly learn the joint probability through the chain rule, allowing to compute the exact likelihood *and* perplexity → direct way to evaluate models!

What probability distribution do approaches based on masking learn?

→ Masked language models (MLM) optimize something called the **pseudo-likelihood**!

What is the difference? True joint probablilty: $p\left(x_1, x_2, x_3\right) = p\left(x_1\right) \cdot p\left(x_2 \mid x_1\right) \cdot p\left(x_3 \mid x_1, x_2\right)$

Pseudo joint probability: $p_{\text{pseudo}}\left(x_1, x_2, x_3\right) = p\left(x_1 \mid x_2, x_3\right) \cdot p\left(x_2 \mid x_1, x_3\right) \cdot p\left(x_3 \mid x_1, x_2\right)$

Pseudo-Likelihood

$$p_{\text{pseudo}}\left(x_1, \ldots, x_T\right) = \prod_{t=1}^{T} p\left(x_t \mid x_{\setminus t}\right)$$

→ approximation to the joint probability distribution

Perplexity and Entropy

Entropy: the entropy of a random variable quantifies the average level of uncertainty or information associated with the variable's potential states or possible outcomes

Continuous case

$$H(p) = \mathbb{E}[I(X)] = \mathbb{E}[\log_b p]$$

Discrete case with N discrete states

$$H(p) = -\sum_{i=1}^{N} p(x_i)\ \log_b p(x_i)$$

Perplexity (PP)

$$PP(p) = b^{H(p)} = b^{\mathbb{E}[\log_b p]}$$

$$PP(p) = b^{H(p)} = b^{-\sum_{i=1}^{N} p(x_i)\ \log_b p(x_i)}$$

https://leimao.github.io/blog/Entropy-Perplexity/

Perplexity and Entropy in Language Models

Language Models (LM): we build a system with state distribution q which tries to mimic the real-world system with the state distribution p as much as possible. Therefore, we need the cross entropy between q and $\tilde{p}$. ($\tilde{p}$ is the proxy of p which is a sampled distribution from the real-word).

Cross-entropy: equals the entropy of p plus the KL divergence of p from q.

$$H(p, q) = -\sum_{i=1}^{N} p(x_i) \log_b q(x_i)$$

Perplexity in LM: By further uniformly sampling sentences x_i ($\tilde{p}(x_i) = \frac{1}{N}$) and factoring it as a product of conditional, we finally can see perplexity = exp(average per-token CE) in practice.

$$PP'(p, q) = b^{-\sum_{i=1}^{n} \tilde{p}(x_i) \log_b q(x_i)} = b^{\frac{1}{N} \sum_{i=1}^{N} \frac{1}{T_i} \sum_{t=1}^{T_i} -\log_b q(x_t^i | x_{<t}^i)}$$

https://leimao.github.io/blog/Entropy-Perplexity/

Scaling Laws

Scaling Laws for Neural Networks

Neural scaling laws establish a mathematical relationship between the following variables in training a model.

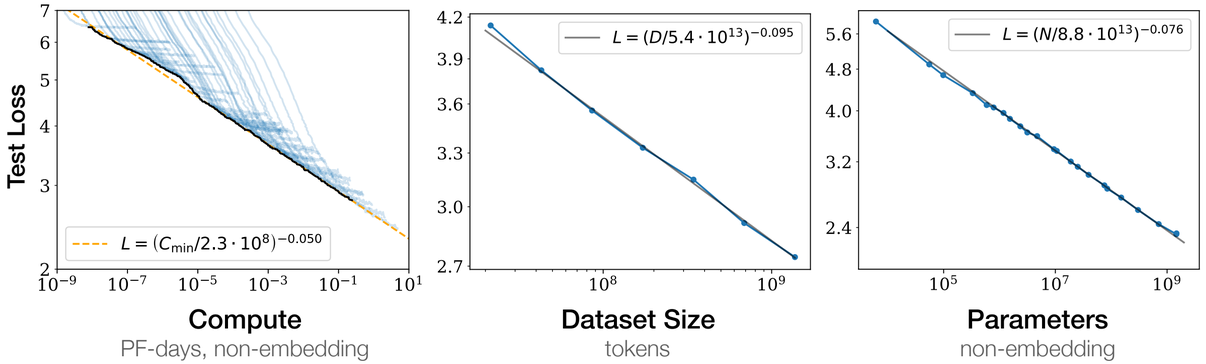
Performance has a **power-law relationship** with each of the three scale factors N, D, C when not bottlenecked by the other two.

$\mathcal{L}$ prediction loss

N number of model parameters

D size of the dataset used to train the model

C total compute used to train the model



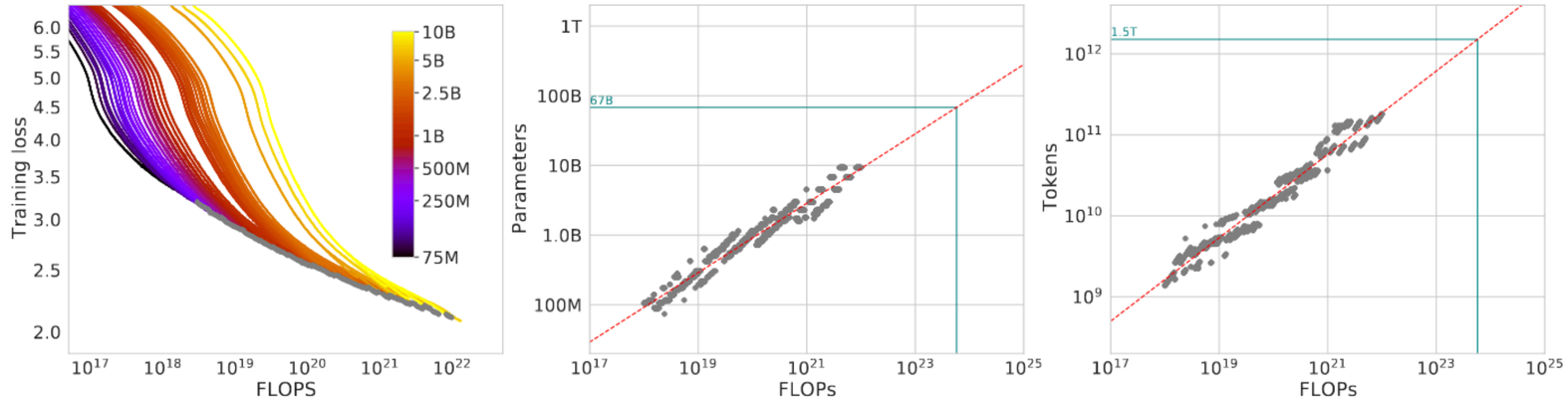
Kaplan et al., (2020)

Chinchilla Scaling Laws

Revisit the question: Given a fixed FLOPs budget, how should one trade-off model size and the number of training tokens?

$$N_{opt}(C), D_{opt}(C) = \underset{N,D \text{ s.t. } \text{FLOPs}(N,D)=C}{\text{argmin}} L(N, D)$$

Approach 1: Fix Model Sizes and Vary Number of Training Tokens



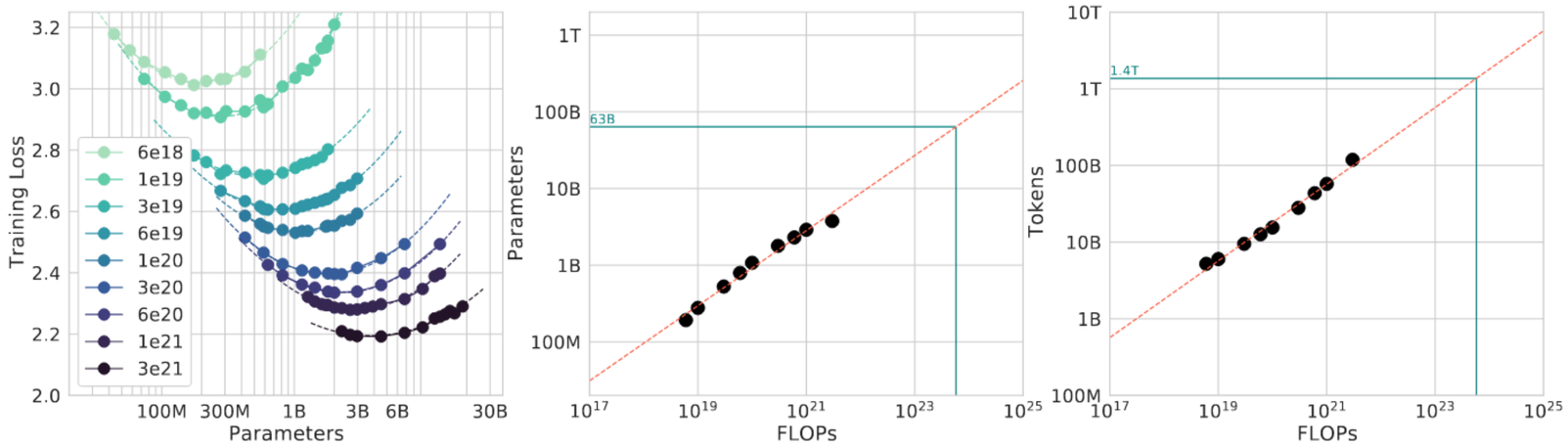
Hoffmann, Jordan, et al. "Training compute-optimal large language models." *arXiv preprint arXiv:2203.15556* (2022)

Chinchilla Scaling Laws

Revisit the question: Given a fixed FLOPs budget, how should one trade-off model size and the number of training tokens?

$$N_{opt}(C), D_{opt}(C) = \underset{N,D \text{ s.t. } \text{FLOPs}(N,D)=C}{\text{argmin}} L(N, D)$$

Approach 2: IsoFLOP Profiles



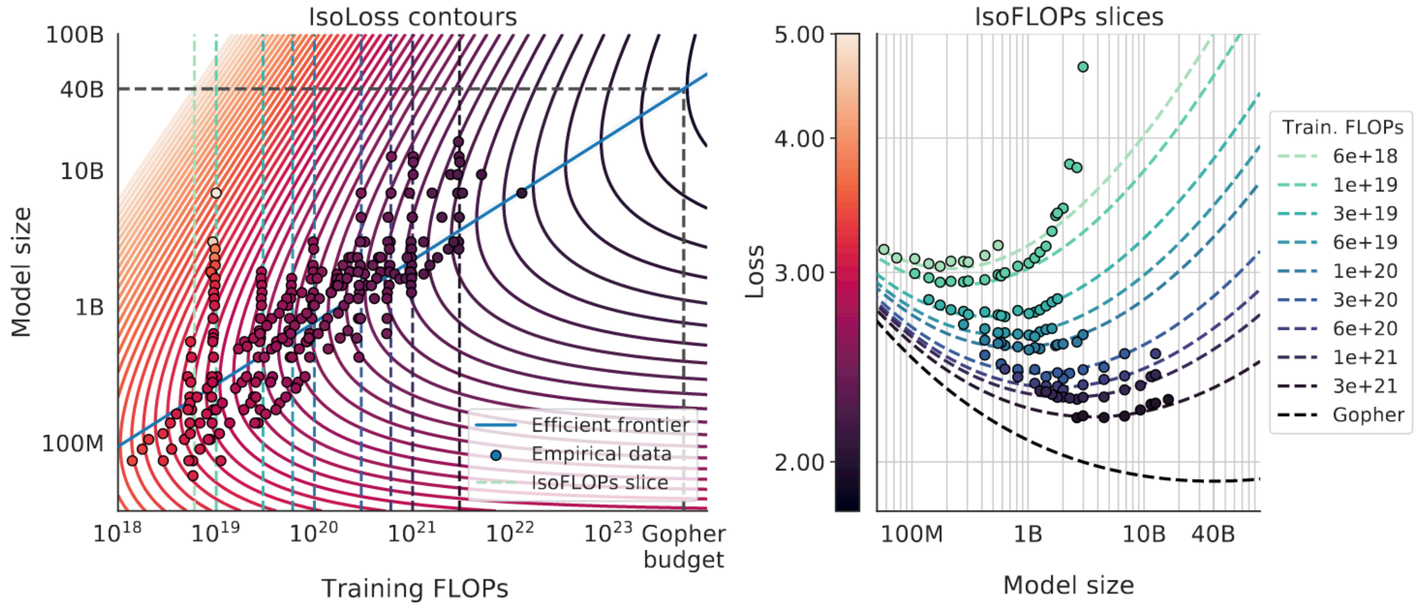
Hoffmann, Jordan, et al. "Training compute-optimal large language models." *arXiv preprint arXiv:2203.15556* (2022)

Chinchilla Scaling Laws

$$N_{opt}(C), D_{opt}(C) = \underset{N,D \text{ s.t. } \text{FLOPs}(N,D)=C}{\text{argmin}} L(N, D)$$

Approach 3: Fitting a Parametric Loss Function

$$\hat{L}(N, D) \triangleq E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}.$$



Hoffmann, Jordan, et al. "Training compute-optimal large language models." *arXiv preprint arXiv:2203.15556* (2022)

Chinchilla Scaling Laws

$$N_{opt}(C), D_{opt}(C) = \underset{N,D \text{ s.t. } \text{FLOPs}(N,D)=C}{\text{argmin}} L(N, D)$$

Table 2 | **Estimated parameter and data scaling with increased training compute.** The listed values are the exponents, a and b, on the relationship $N_{opt} \propto C^a$ and $D_{opt} \propto C^b$. Our analysis suggests a near equal scaling in parameters and data with increasing compute which is in clear contrast to previous work on the scaling of large models. The 10^{th} and 90^{th} percentiles are estimated via bootstrapping data (80% of the dataset is sampled 100 times) and are shown in parenthesis.

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)
Kaplan et al. (2020)	0.73	0.27

Hoffmann, Jordan, et al. "Training compute-optimal large language models." *arXiv preprint arXiv:2203.15556* (2022)

Math and Code Exercises

Next Steps

1. Math and code: Go over solutions.
2. Code: Have discussions.
3. Code: Have some students present their solutions.