Lecture 9

SCALING LAWS - BASICS

CS336

Taking scaling seriously

Imagine the following scenario..

Your friend has given you ten thousand H100s for a month,
and asked you to build a good open source LM.

What do you do?

- Put together your infra team and distributed training framework (A2)
- Put together a great pretraining dataset (A4)
- Run a big model (but which one??) <- we are here.

Scaling isn’t easy

Wide or deep? How many heads? Which nonlinearity?

Name	Has pa...	Link	Year	Tokenizer type	Vocab count	Norm	Parallel Layer	Pre-norm	Position embedding	Activations	MoE	MLP factor	num_layers	model_dim
Original transformer	Yes	arxiv.org/abs...03762	2017	BPE	37000	LayerNorm	Serial	☐	Sine	ReLU	☐	4	6	
GPT	Yes	cdn.openai.com/res...er.pdf	2018	BPE	40257	LayerNorm	Serial	☐	Absolute	GeLU	☐	4	12	
GPT2	Yes	cdn.openai.com/bet...rs.pdf	2019	BPE	50257	LayerNorm	Serial	☑	Sine	GeLU	☐	4	48	
T5 (11B)	Yes	arxiv.org/abs...10683	2019	SentencePiece	32128	RMSNorm	Serial	☑	Relative	ReLU	☐	64	24	
GPT3 (175B)	Yes	arxiv.org/abs...14165	2020	BPE	50257	LayerNorm	Serial	☑	Sine	GeLU	☐	4	96	
mT5	Yes	arxiv.org/abs...11934	2020	SentencePiece	250000	RMSNorm	Serial	☑	Relative	GeGLU	☐	2.5	24	
T5 (XXL 11B) v1.1	Kind of	github.com/goo...d#t511	2020	SentencePiece	32128	RMSNorm	Serial	☑	Relative	GeGLU	☐	2.5	24	
Gopher (280B)	Yes	arxiv.org/abs...11446	2021	SentencePiece	32000	RMSNorm	Serial	☑	Relative	ReLU	☐	4	80	
Anthropic LM (not claude)	Yes	arxiv.org/abs...00861	2021	BPE	65536			☑			☐	4	64	
LaMDA	Yes	arxiv.org/abs...08239	2021	BPE	32000			☑	Relative	GeGLU	☐	8	64	
GPTJ	Kind of	huggingface.co/Ele...t-j-6b	2021	BPE	50257	LayerNorm	Parallel	☑	RoPE	GeLU	☐		28	
Chinchilla	Yes	arxiv.org/abs...15556	2022	SentencePiece	32000	RMSNorm	Serial	☑	Relative	ReLU	☐	4	80	
PaLM (540B)	Yes	arxiv.org/abs...02311	2022	SentencePiece	256000	RMSNorm	Parallel	☑	RoPE	SwiGLU	☐	4	118	
OPT (175B)	Yes	arxiv.org/abs...01068	2022	BPE	50272	LayerNorm	Serial	☑	Absolute	ReLU	☐	4	96	
BLOOM (175B)	Yes	arxiv.org/abs...05100	2022	BPE	250680	LayerNorm	Serial	☑	AliBi	GeLU	☐	4	70	
GPT-NeoX	Yes	arxiv.org/pdf...45.pdf	2022	BPE	50257	LayerNorm	Parallel	☑	RoPE	GeLU	☐	4	44	
GPT4 OPEN	Ad	arxiv.org/abs...08774	2023	BPE	100000			☐			☐			
LLaMA (65B)	Yes	arxiv.org/abs...13971	2023	BPE	32000	RMSNorm	Serial	☑	RoPE	SwiGLU	☐	2.6875	80	
LLaMA2 (70B)	Yes	arxiv.org/abs...09288	2023	BPE	32000	RMSNorm	Serial	☑	RoPE	SwiGLU	☐	3.5	80	
Mistral (7B)	Yes	arxiv.org/abs...06825	2023	BPE	32000	RMSNorm	Serial	☑	RoPE	SwiGLU	☐	3.5	32	

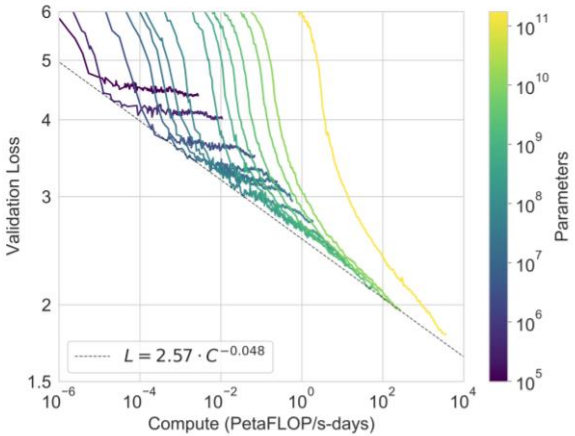
We could cargo cult things from existing LMs… but how do these get optimized in the first place?

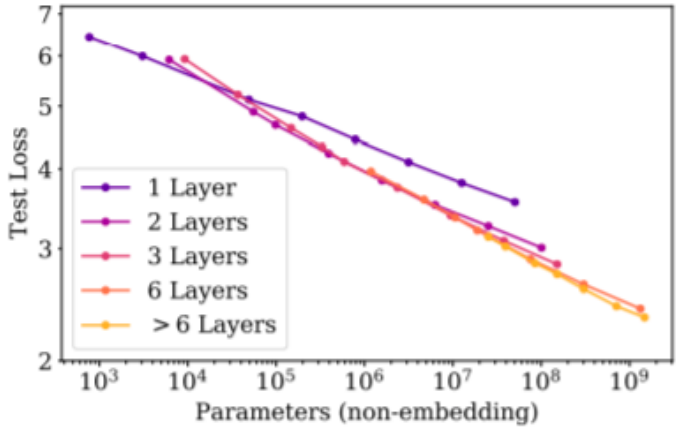
Today: simple, predictive ‘laws’ for behaviors of LMs

The approach -

scaling laws which are simple, predictive rules for model performance

Old and unpleasant: tune hyperparameters on big models
New (over?) optimism: tune on small models, extrapolate to large ones





Part 1. Scaling laws, history and background.

- Data scaling as empirical sample complexities

- Initial forays into understanding neural scaling with data

Sample complexity and rates

Theorists have thought about ‘scaling’ for a long time..

$$\epsilon(\hat{h}) \leq \left(\min_{h \in H} \epsilon(h)\right) + 2\sqrt{\frac{1}{m} \log \frac{2k}{\delta}}$$

(learning in a finite set of k hypotheses)

Under the assumptions of Theorem 1.5, the rate of convergence of the estimator $\hat{p}_n(x_0)$ is $\psi_n = n^{-\frac{\beta}{2\beta+1}}$, which means that for a finite constant C and for all $n \geq 1$ we have

$$\sup_{p \in \mathcal{P}(\beta, L)} \mathbb{E}_p \left[\left(\hat{p}_n(x_0) - p(x_0) \right)^2 \right] \leq C \psi_n^2.$$

(generative modeling for smooth densities)

But these are upper bounds, not actual, realized loss values.

https://www.cs.cmu.edu/~epxing/Class/10701/slides/lecture16-VC.pdf

Hall, 1989

Earliest (data) scaling law paper – 1993

Learning Curves: Asymptotic Values and Rate of Convergence

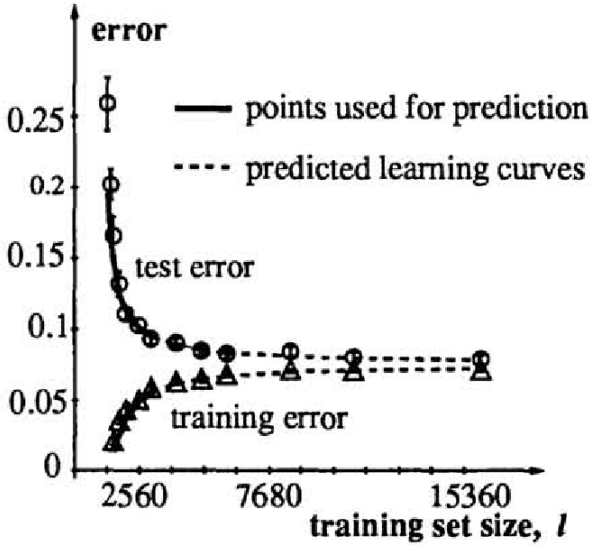
Corinna Cortes, L. D. Jackel, Sara A. Solla, Vladimir Vapnik, and John S. Denker
AT&T Bell Laboratories
Holmdel, NJ 07733

Abstract

Training classifiers on large databases is computationally demanding. It is desirable to develop efficient procedures for a reliable prediction of a classifier's suitability for implementing a given task, so that resources can be assigned to the most promising candidates or freed for exploring new classifier candidates. We propose such a practical and principled predictive method. Practical because it avoids the costly procedure of training poor classifiers on the whole training set, and principled because of its theoretical foundation. The effectiveness of the proposed procedure is demonstrated for both single- and multi-layer networks.

A typical example of learning curves is shown in Fig. 2. The test error is always larger than the training error, but asymptotically they reach a common value, a. We model the errors for large sizes of the training set as power-law decays to the

$$\mathcal{E}_{\text{test}} = a + \frac{b}{l^{\alpha}} \qquad \text{and} \qquad \mathcal{E}_{\text{train}} = a - \frac{c}{l^{\beta}}$$



Early history of scaling laws – data scaling

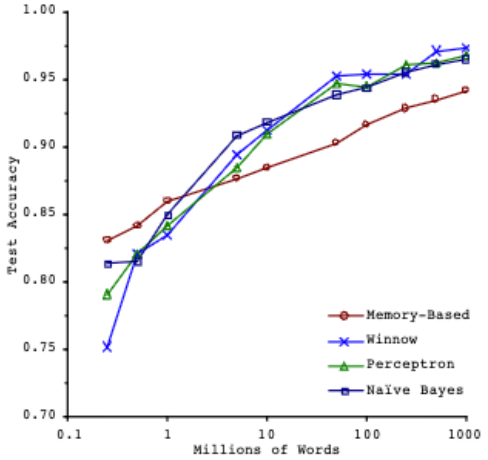
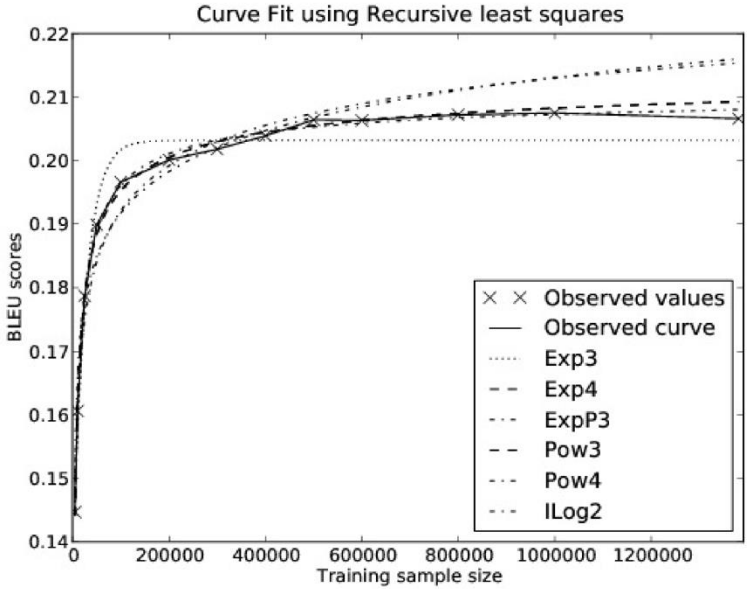


Figure 1. Learning Curves for Confusion Set Disambiguation

these results suggest that we may want to reconsider the trade-off between spending time and money on algorithm development versus spending it on corpus development. At least for the problem of confusable disambiguation, none of the learners tested is close to asymptoting in performance at the training corpus size commonly employed by the field.

Log-linear scaling with data [Banko and Brill ‘01]

Early history of scaling laws – data scaling



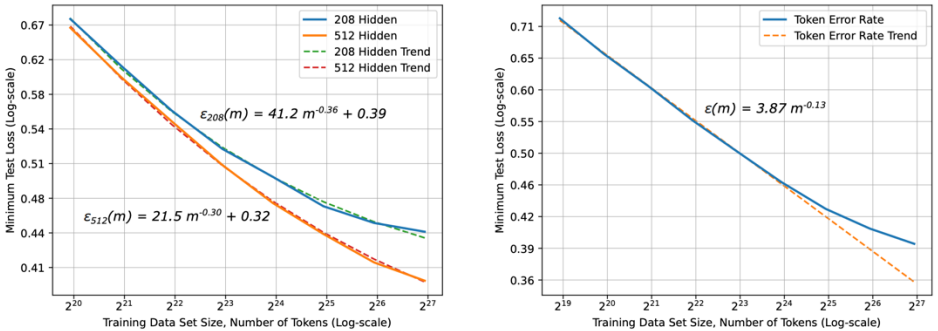
Model	Formula
Exp_3	$y = c - e^{-ax+b}$
Exp_4	$y = c - e^{-ax^{\alpha}+b}$
ExpP_3	$y = c - e^{(x-b)^{\alpha}}$
Pow_3	$y = c - ax^{-\alpha}$
Pow_4	$y = c - (-ax+b)^{-\alpha}$
ILog_2	$y = c - (a/\log x)$

Table 1: Curve families.

Early tests of functional forms

Kolachina et al 2012 – power law relation between data and downstream performance

Hestness et al 2017



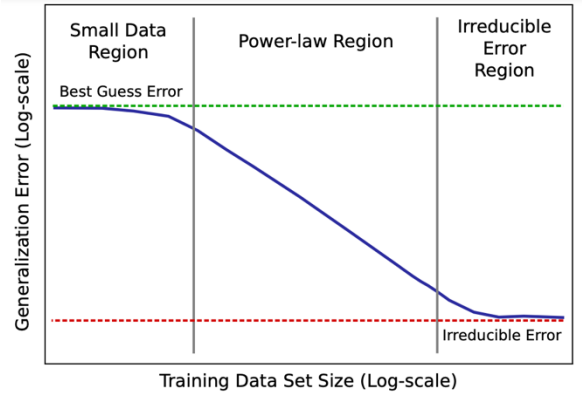


Figure 1: Neural machine translation learning curves. Left: the learning curves for separate models follow $\varepsilon(m) = \alpha m^{\beta_g} + \gamma$. Right: composite learning curve of best-fit model at each data set size.

Earliest 'large scale neural' scaling work: Hestness 2017

Predictable scaling on many tasks (MT, LM, Speech) and hypothesized scaling shape.

Hestness II

Very ahead of its time..

“Emergence”

> Although small data set testing may be possible, it can be difficult to ensure that training data is large enough to see the power-law learning curve region. We have found that models with poor optimizer parameterization or model priors/initialization show accuracy cliffs, where accuracy is only as good as best guessing, but the model trains on enough data to be in the power-law region. Researchers must take great care when defining a "large enough" training set for small data testing. We leave the methodology for defining such a training set to future work.

Scaling by compute

> **Computational Limits:** If we have identified a desirable model to scale to larger training sets, the next potential limitation is the speed of computation. In some cases, training large models on very large data sets would take months or years of critical path compute time, making these training runs impractical for any real world problem on existing systems. However, predictable learning and model size curves may offer a way to project the compute requirements to reach a particular accuracy level. The compute requirements could inform decisions about how to scale computational capacity to unlock these compute-limited applications.

Speed = accuracy

> **The Performance-Accuracy Trade-off:** Many DL software and hardware techniques impose a trade-off between model accuracy and the speed of computation. Learning curves and model size growth can indicate whether these techniques could regain lost accuracy by improving the speed of computation. For example, low-precision computation/quantization and sparse models give up some model accuracy (e.g., up to 20%) in order to improve compute throughput. If the compute throughput improvements allow DL developers to train larger models on larger data sets, these accuracy losses might be easily recoverable.

Part 2. Neural (LLM) scaling behaviors

1.Data vs performance

“Are there simple rules that determine how data affects performance?”

2. Data vs model size

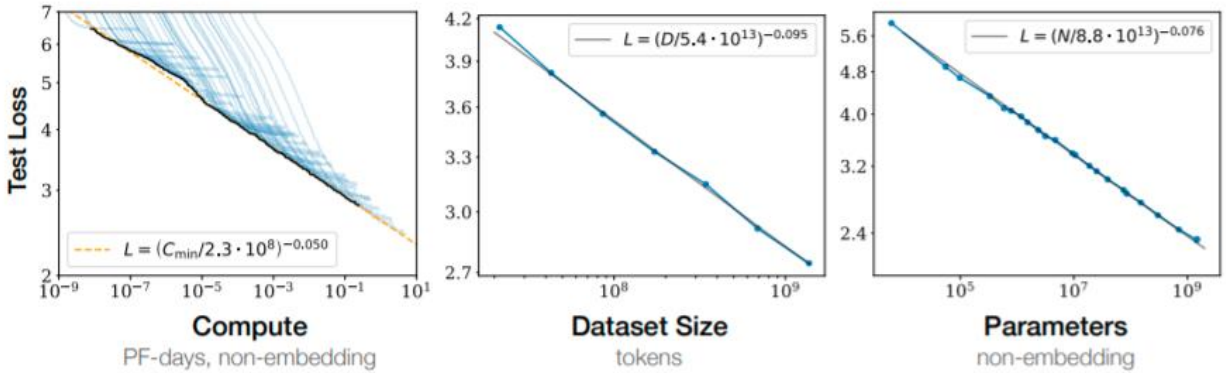
Do we train on more data or bigger models?

3. Hyper-parameters vs performance

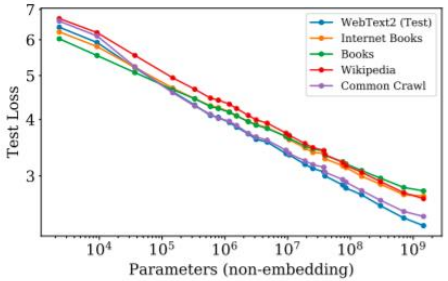
“How should we set hyperparameters on the big model??”

Scaling laws – power law relationships for many factors

These scaling laws hold on *many* different kind of phenomena!



They even hold in non-standard settings (when train ≠ test)



[Kaplan+ 2020]

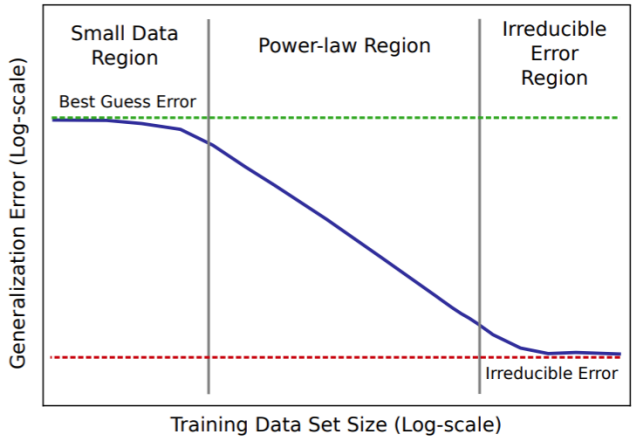
Data vs performance

What's a data scaling law?

Data scaling laws : simple formula that maps dataset size (n) to error

What do we expect out of scaling laws?

Monotonic, logistic-like curves

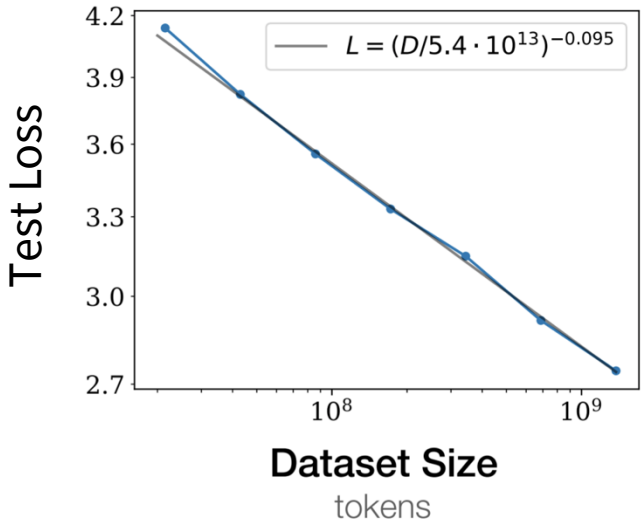


[Hestness+ 2017]

Data scaling laws for language models

First, an empirical observation

Loss and dataset size is linear on a log-log plot



"Scale-free" or
"Power law"

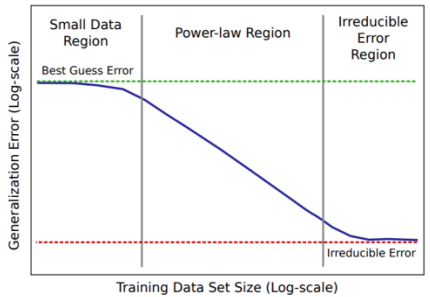
(For language modeling, from Kaplan+ 2020)

Conceptual foundations of data scaling laws.

Q: Why do scaling laws show up?

We know error should be monotone

But why is it a power law / linear in log-log?



A (?): Estimation error naturally decays polynomially.

But this answer may take a moment to understand. Let's work through an example.

Example: If our task is to estimate the mean of a dataset, what's the scaling law?

Toy example: mean estimation

Input: $x_1 \ \ldots x_n \ \sim N(\mu, \sigma^2)$

Task: estimate the average as $\hat{\mu} = \frac{\sum_i x_i}{n}$

What's the error? By standard arguments..

$$\mathrm{E}[(\hat{\mu} - \mu)^2] = \frac{\sigma^2}{n}$$

This is a 'scaling law'

$$\log(Error) = -\log n + 2\log \sigma$$

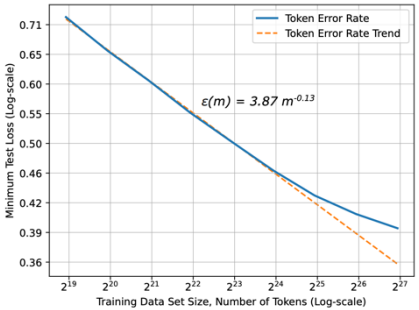
More generally, any polynomial rate $1/n^{\alpha}$ is a scaling law

Scaling law exponents: an intriguing mystery

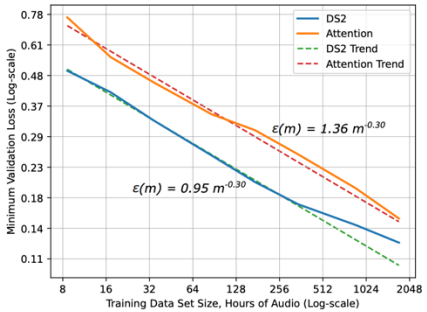
Fact: Similar arguments show most 'classical' models (regression, etc) have $\frac{1}{n}$ scaling

This means we should see $y = -x + C$

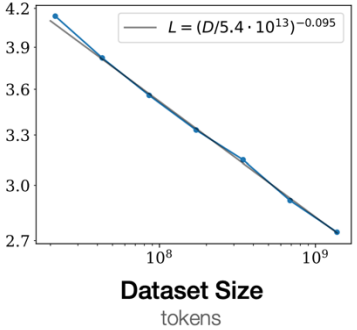
What do we find in neural scaling laws?



Machine translation



Speech



Language modeling

Very different from predictions.. Why might this be?

Detour: scaling laws for (nonparametric) learning

Neural nets can approximate arbitrary functions. Lets turn that into an example.

Input: $x_1 \ldots x_n$ uniform in 2D unit box. $y_i = f(x_i) + N(0,1)$
Task: estimate f(x)

Approach: cut up the 2D space into boxes with length $n^{-\frac{1}{4}}$

What's our estimation error?

Informally, we have $\sqrt{n}$ boxes, each box gets $\sqrt{n}$ samples.

$$Error \approx \frac{1}{\sqrt{n}} + (other\ smoothness\ terms)$$

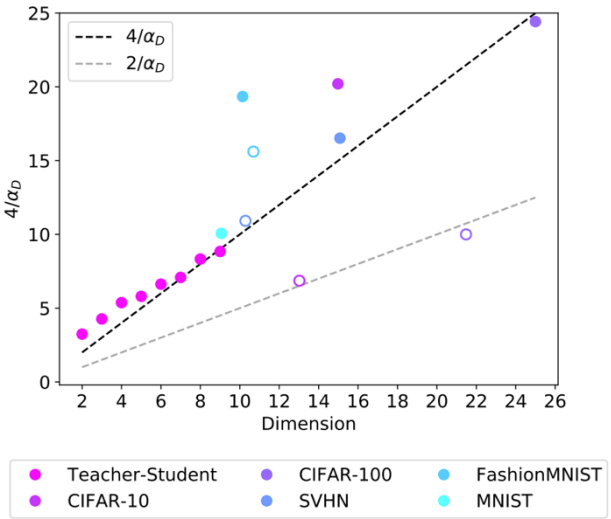
In d-dimensions, this becomes $Error = n^{-1/d}$ - **This means scaling is $\boldsymbol{y = -\frac{1}{d}x + C}$**

Takeaway: flexible 'nonparametric' learning has dimension dependent scaling laws.

Intrinsic dimensionality theory of data scaling laws

Some have made the following argument (Bahri 2021)

1. Scaling laws arise due to polynomial rates of learning $\frac{1}{n^{\alpha}}$
2. Some argue the slope α is closely connected to the *intrinsic dimensionality* of the data.



But estimators of intrinsic dimension are sketchy, and this is not airtight..

Other data scaling laws

Data scaling thus far: how does dataset size relate to performance?
Related question: how does dataset *composition* affect performance

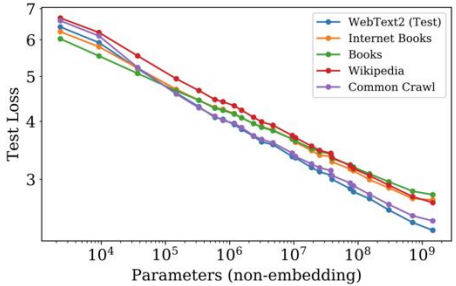
- Picking optimal data mixture using small scale models
- Deciding whether to repeat data or not
- Combing the two and balancing quality with repetition rate

Other advanced data scaling law: distribution shift

Data scaling thus far: how does dataset size relate to performance?

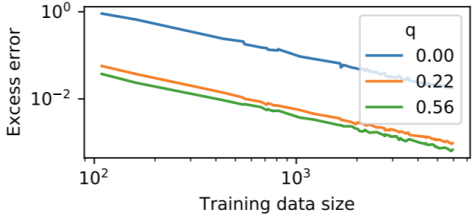
Related question: how does dataset *composition* affect performance

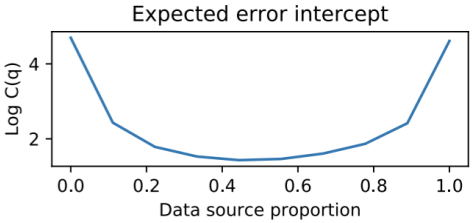
A: Data composition affects the offset, not the slope.



[Kaplan+ 2021]

These ‘distribution shift’ scaling laws can tell us about the importance of collecting diverse data!

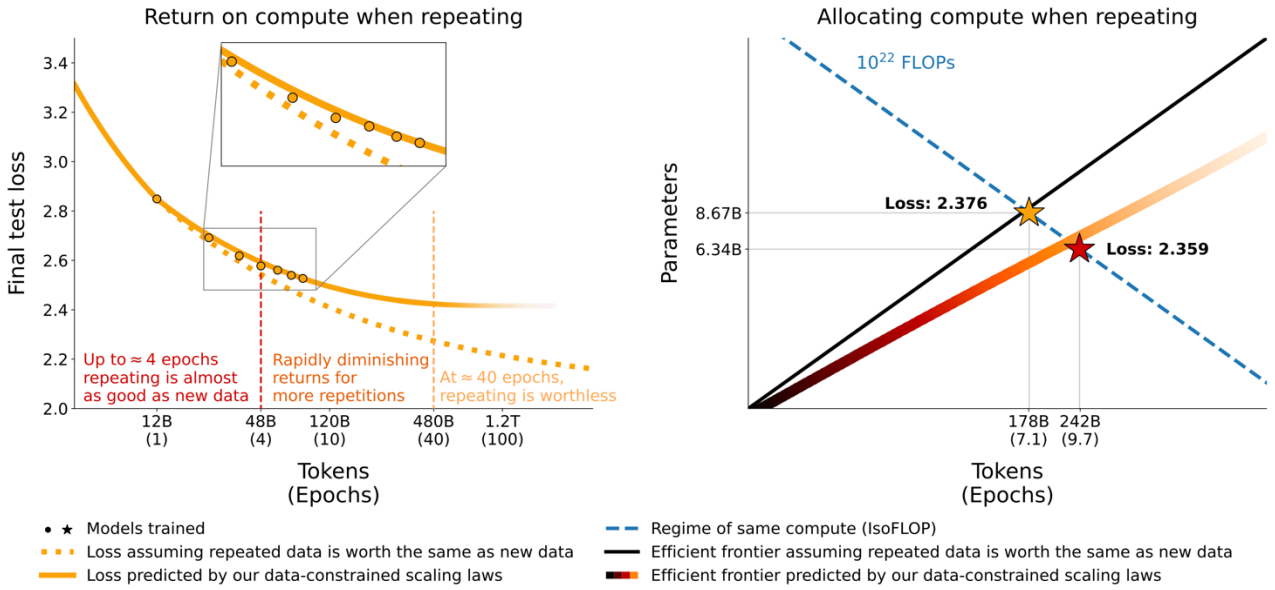




[Hashimoto 2021]

Scaling laws under data repetition

In practice, we have finite data – how does repeating examples affect scaling?



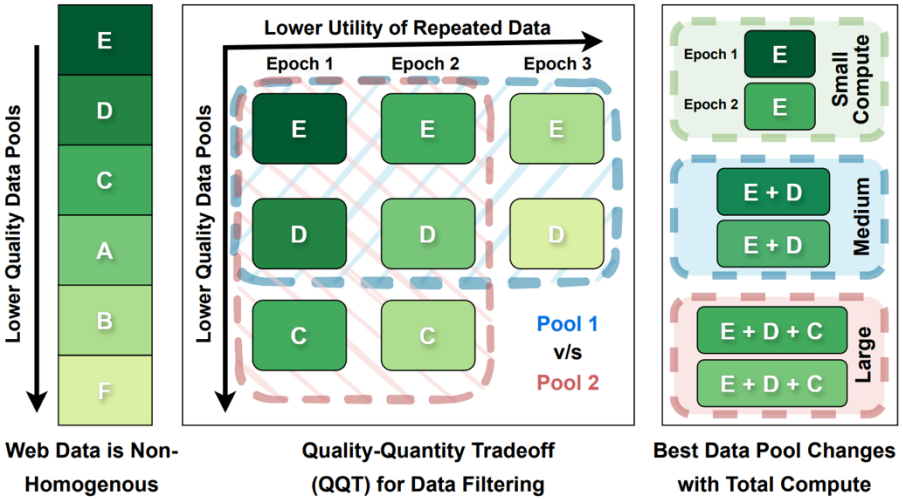
$$D' = U_D + U_D R_D^* (1 - e^{\frac{-R_D}{R_D^*}}) .$$

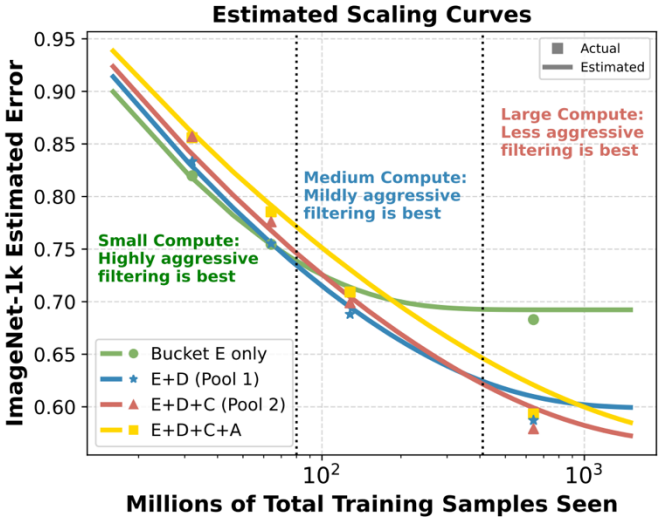
D' = Effective data
Ud = Unique tokens
Rd* = Constant
Rd = Repetition

Data selection scaling and accounting for finiteness

Given that repeated data is less valuable..

Data selection should then be adaptive to scale !





Recap: data scaling laws

- Remarkably linear relationship between log-data size and log-error
- Holds across domains and models
- Theory understanding: similar to generalization bounds: mean estimation example
- Applications: data collection / curation

Scaling laws for model engineering

Now for what I promised at the start: **model scaling!**

Our motivation: how can we efficiently design huge LMs?

- LSTMs vs Transformers
- Adam vs SGD

How should we allocate our limited resources?

- Train models longer vs train bigger models?
- Collect more data vs get more GPUs?

Scaling laws provide a simple procedure to answer these.

Hyperparameter questions

We'll consider some of these choices in the context of the classic Kaplan scaling paper

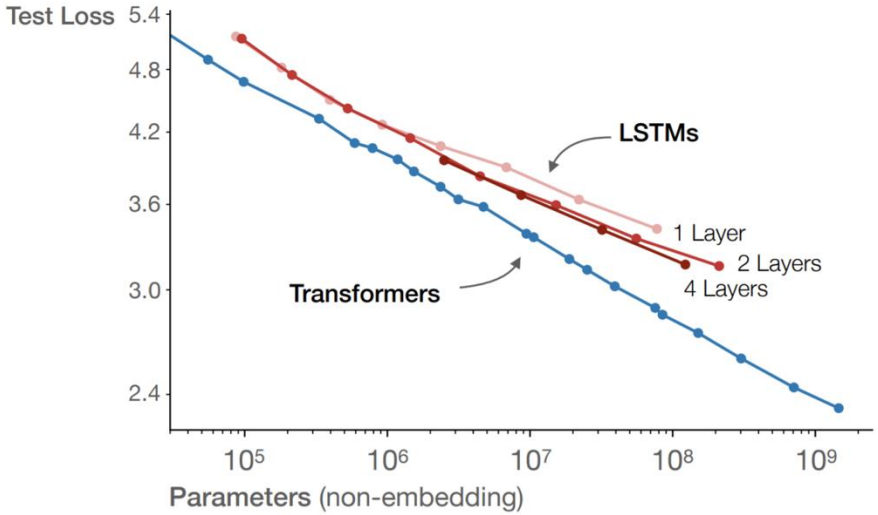
- Architecture
- Optimizer
- Aspect ratio / depth
- Batch size

1. Architecture: transformers vs LSTMs

Q: Are transformers better than LSTMs?

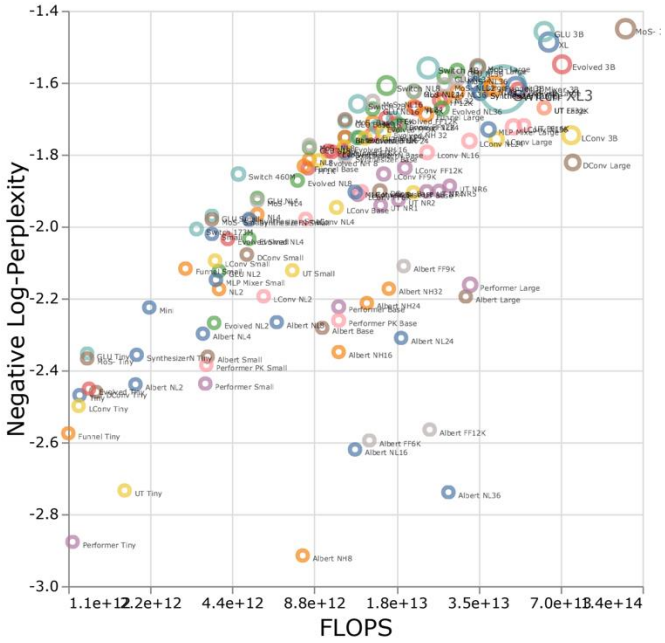
Brute force way: spend tens of millions to train a LSTM GPT-3

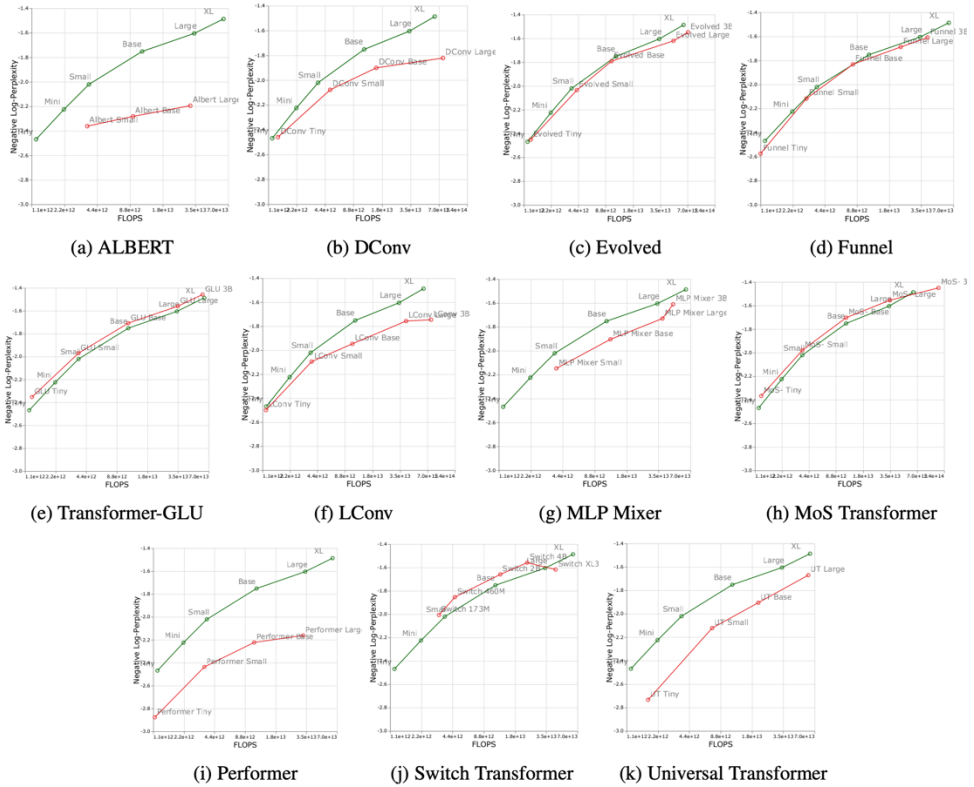
Scaling law way:



[Kaplan+ 2021]

1. Many architectures

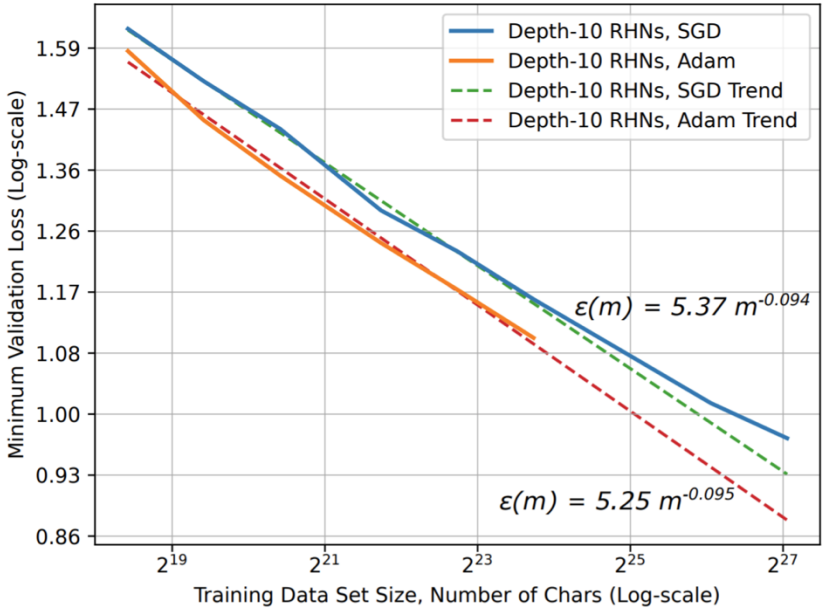




Cross-architecture scaling in Tay et al.

2. Optimizer choice

What about ADAM vs SGD?

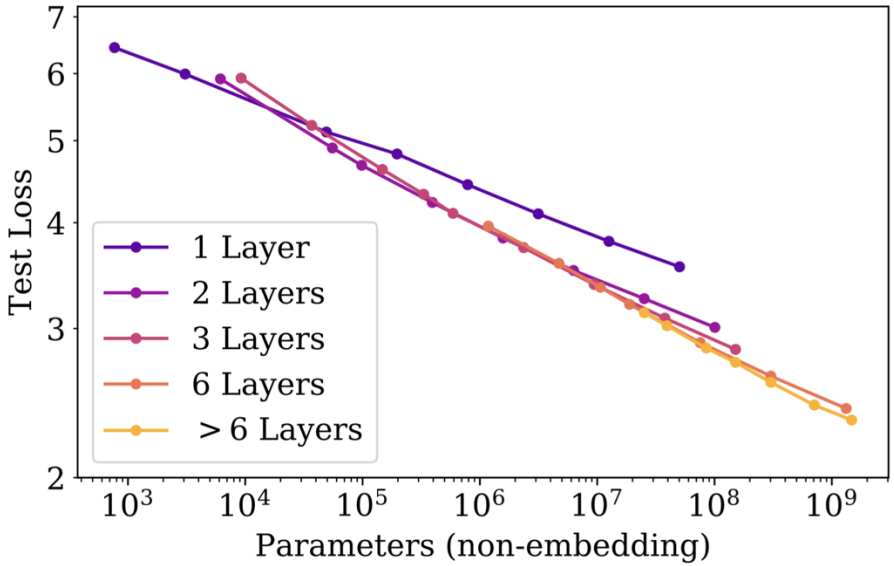


[Hestness+ 2017]

(Note, this is in 2017, so pre-transformers. RHN is recurrent highway nets)

3. Depth/Width: Number of layers

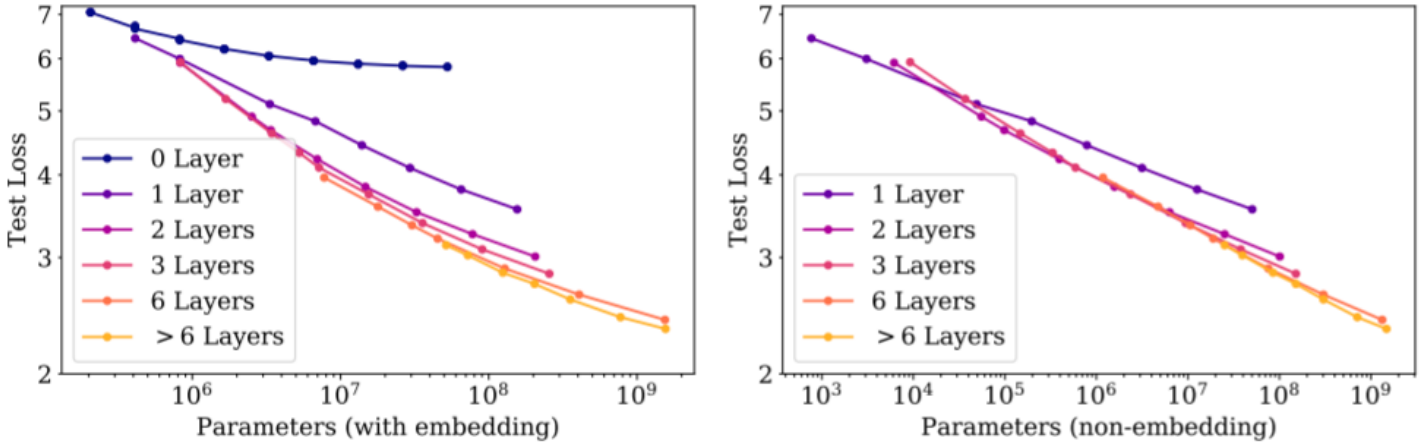
Does depth or width make a huge difference?



- 1 vs 2 layers makes a huge difference.
- More layers have diminishing returns below 10^7 params

3. Depth/Width: But not all parameters are made equal

We've been thinking about 'parameters' but not all parameters are equal



Embedding layer parameters don't behave the same!

Related: recent papers on scaling laws for mixtures of experts.

3. Depth/Width: and other Transformer hypers

Do hyperparameters like the aspect ratio depend on scale?

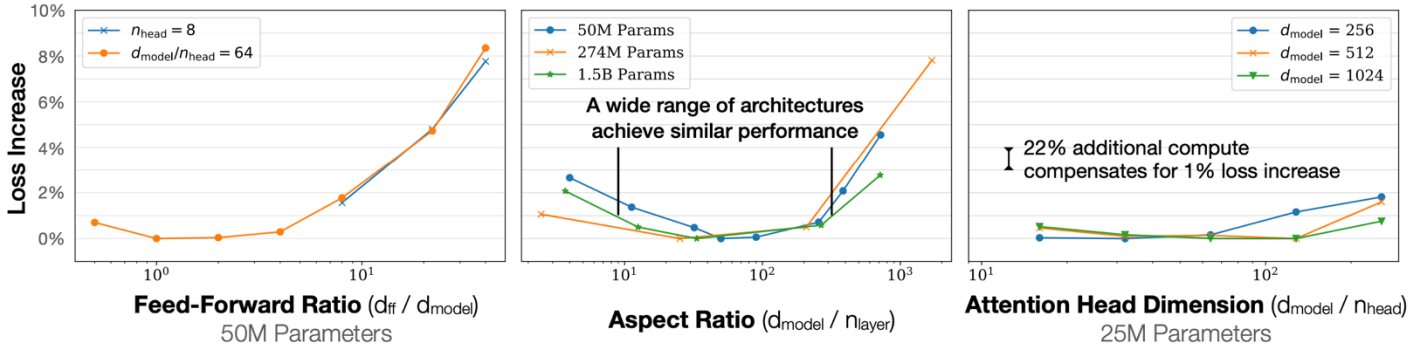
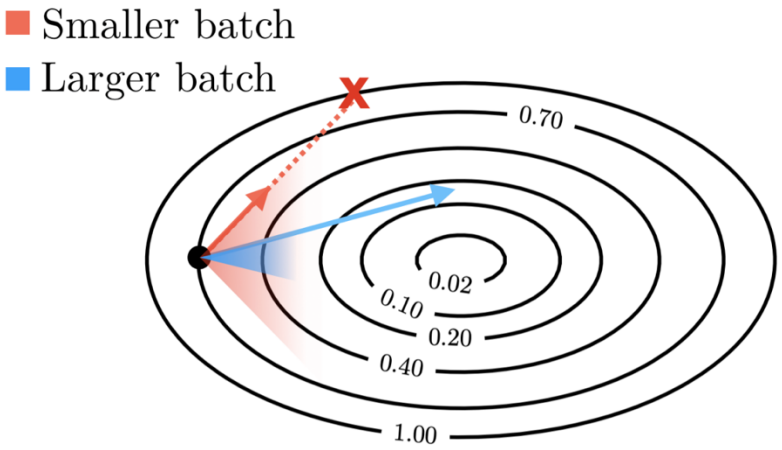
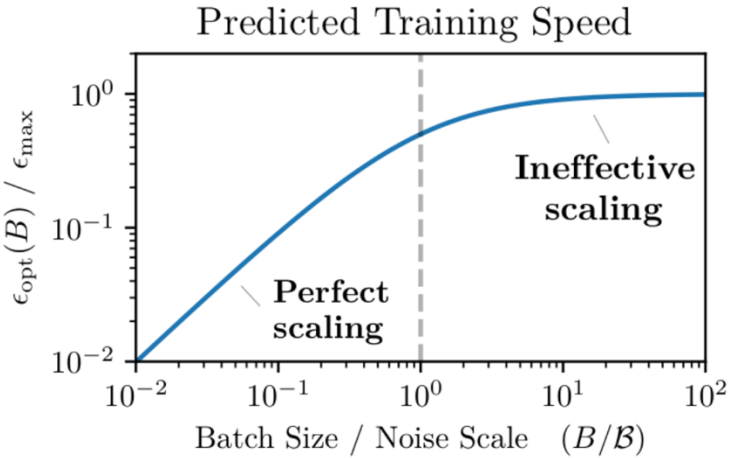


Figure 5 Performance depends very mildly on model shape when the total number of non-embedding parameters N is held fixed. The loss varies only a few percent over a wide range of shapes. Small differences in parameter counts are compensated for by using the fit to $L(N)$ as a baseline. Aspect ratio in particular can vary by a factor of 40 while only slightly impacting performance; an $(n_{\text{layer}}, d_{\text{model}}) = (6, 4288)$ reaches a loss within 3% of the $(48, 1600)$ model used in [RWC+19].

4. Batch size: Critical batch size

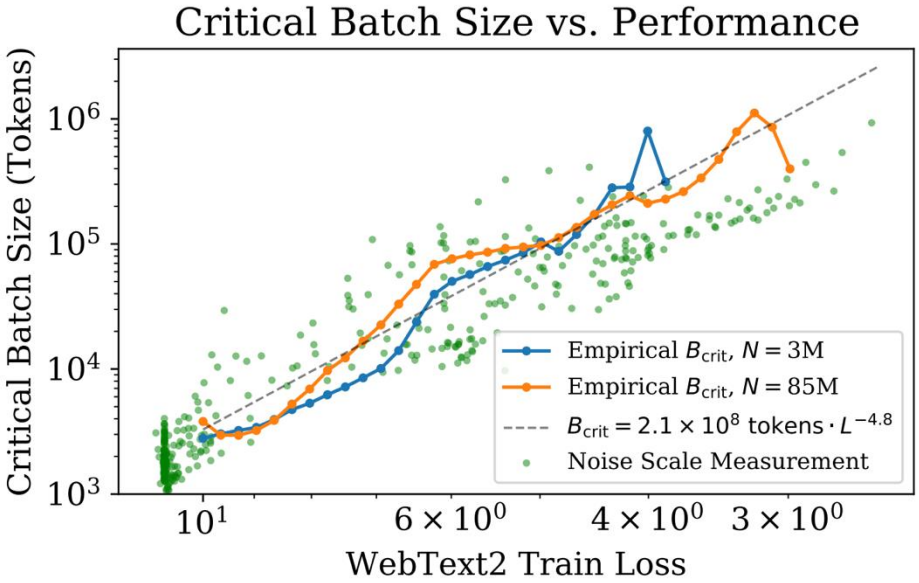




Batch size – known to have strong diminishing returns past a certain point.

Critical batch = min number of examples for target loss / min number of steps for target loss

4. Batch size: critical batch size



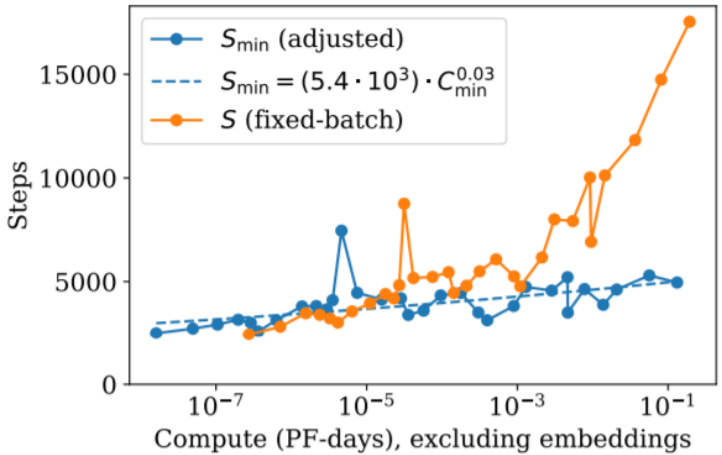
The smaller the loss target,
The bigger the batch

$$C_{\min}(C) \equiv \frac{C}{1 + B/B_{\text{crit}}(L)} \qquad \text{(minimum compute, at } B \ll B_{\text{crit}}\text{)}$$

4. Batch size: selecting the optimal batch

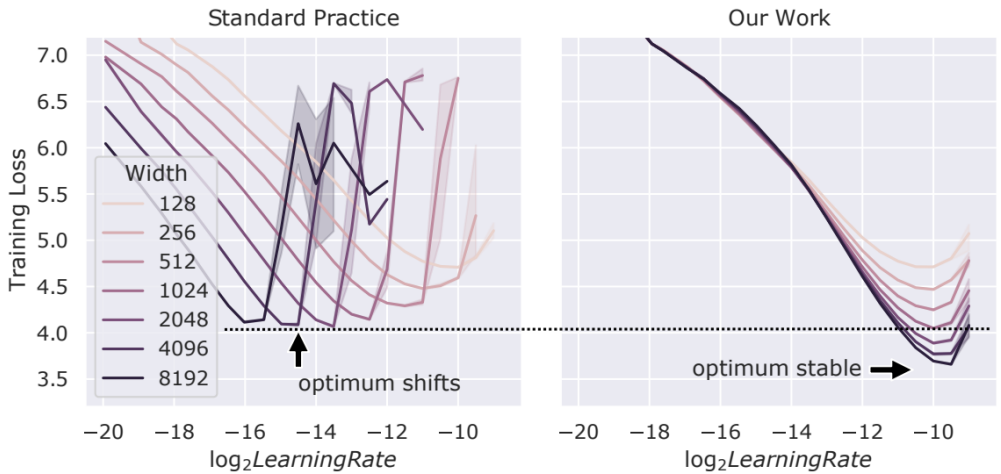
Q: as we increase both compute and model size, how should we scale training?

- Huge batches, same number of steps
- Fixed batches, more steps



Good news for data parallel processing (?)

5. Learning rates: muP and scale-aware LR choices



Yang et al 2022

Table 2. μP function for a model M' that is r times the widths of M. If a parameter tensor has 2 dimensions that goes infinite when the model width goes infinite, it is "matrix-like" (*e.g.,* a fully-connected hidden layer); if the number is 1 or 0, it belongs to the "others" class. Note that embedding layers are "others". "Output" means the layer that maps an infinite dimension to a finite dimension, which is the word decoding layer (lm_head) in Transformers. A multiplier is a constant multiplied by a parameter tensor, which has a similar function to softmax temperature.

Hyperparameter (weight)	M	$M' \sim r$
AdamW learning rate (matrix-like)	l	l/r
AdamW learning rate (others)	l	l
Initialization variance (matrix-like)	σ	σ/r
Initialization variance (others)	σ	σ
Multiplier (output)	τ	τ/r
Multiplier (others)	τ	τ

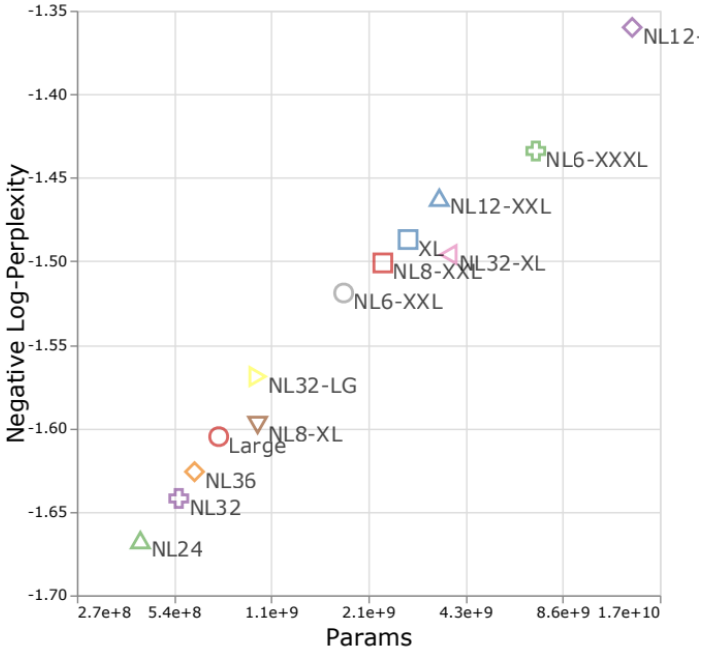
Yao et al 2024

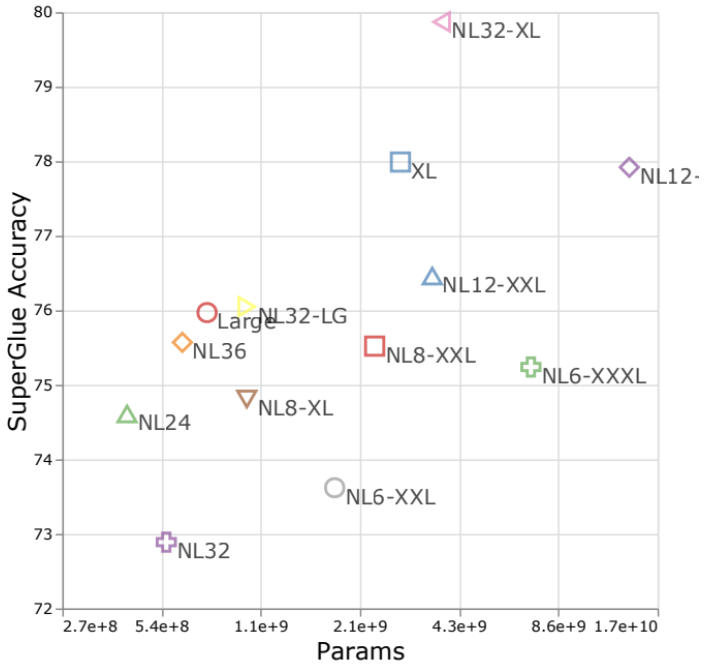
If we naively scale up – optimal learning rate depends on scale.
We need *scaling aware* initialization and learning rate scaling

Caution – scaling behaviors can differ downstream

Thus far: scaling is predictable and depends mainly on parameters

Catch: downstream scaling can often be much less predictable





Tay et al 2023

Some surprising takeaways

The effect of hyperparameters on big LMs can be predicted *before* training!

- Optimizer choice
- Model depth
- Architecture choice

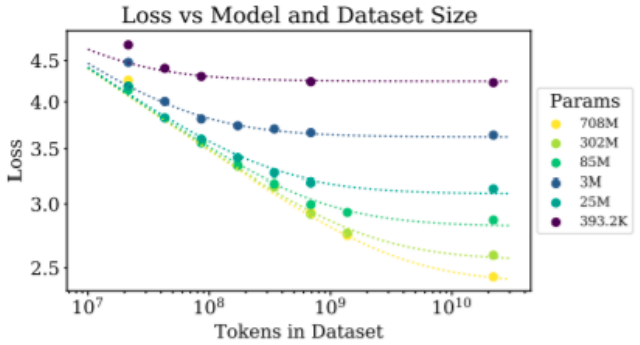
The scaling law based design procedure.

1. Train a few smaller models
2. Establish a scaling law (e.g. ADAM vs SGD scaling law)
3. Select optimal hyperparam based on the scaling law prediction.

One important use of scaling laws

Q: Do we need more data or bigger models?

Clearly, lots of data is wasted on small models



Joint data-model scaling laws describe how the two relate

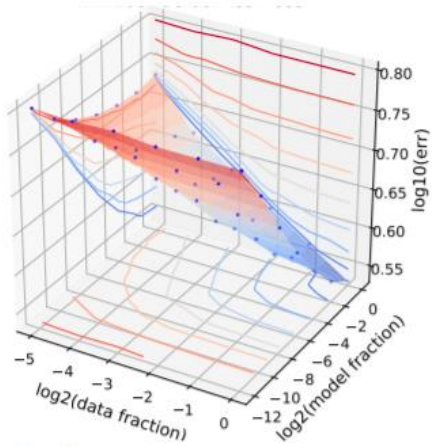
From Rosenfeld+ 2020,

$$Error = n^{-\alpha} + m^{-\beta} + C$$

From Kaplan+ 2020

$$Error = [m^{-\alpha} + n^{-1}]^{\beta}$$

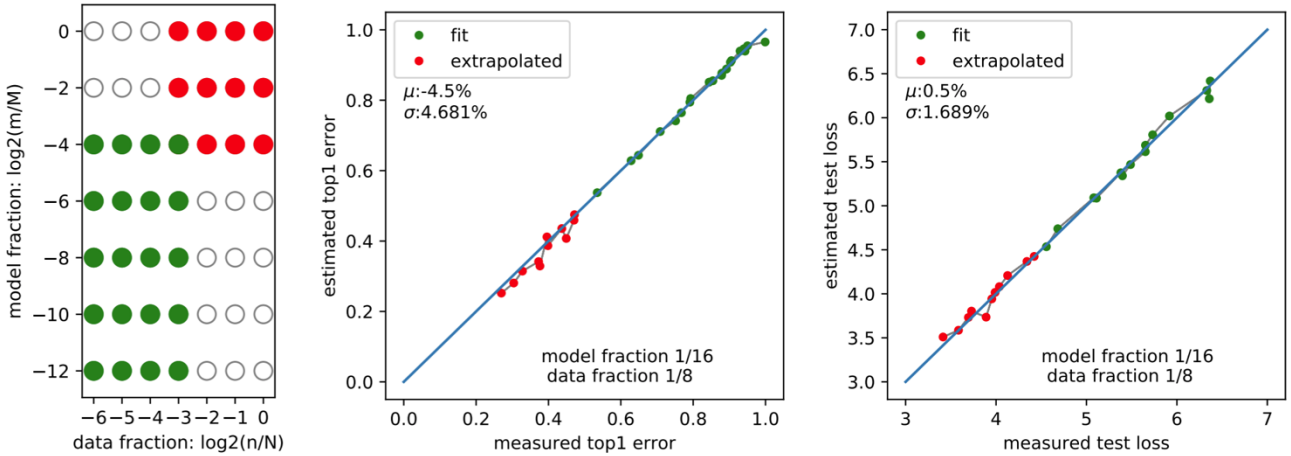
Provides surprisingly good fits to model-data joint error.



(a) Wiki103 error (cross entropy) landscape.

Model-data joint scaling is accurate

From Rosenfeld – fit scaling exponents on small data, small models. Predict rest.



(a) Illustration. (b) Extrapolation on ImageNet (c) Extrapolation on WikiText-103.

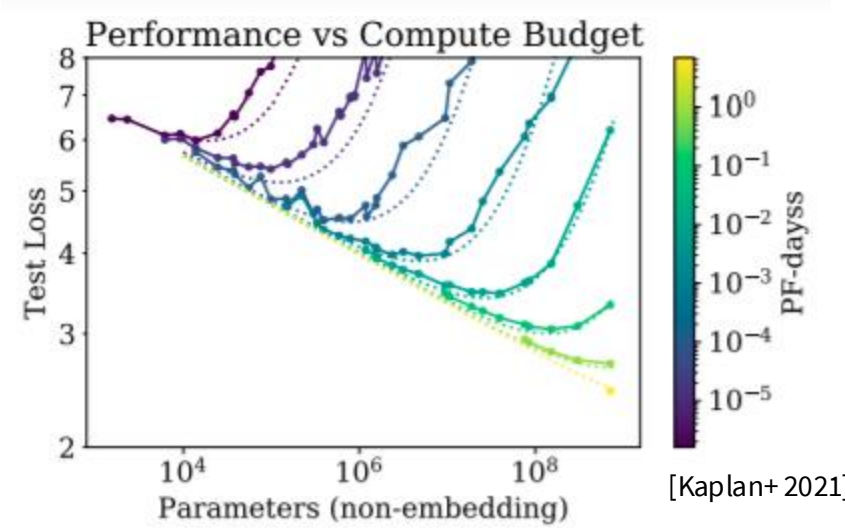
Trading off data size and model size: optimize $n^{-\alpha}+m^{-\beta}+C$ with your costs.

Compute tradeoffs.

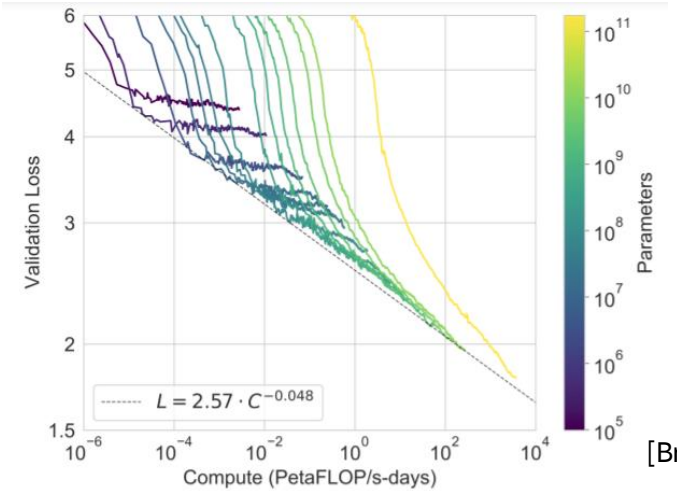
Q: what about other resources? Compute vs performance?

For a fixed compute budget…

Big model that's undertrained vs small model that's well trained?



[Kaplan+ 2021]



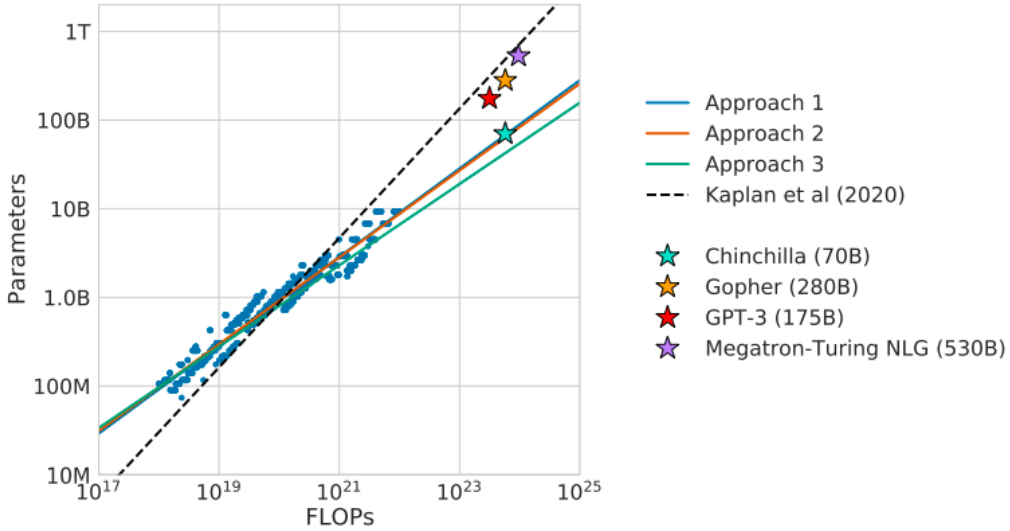
[Brown+ 2020]

Scaling laws let us navigate this tradeoff

Caution – ‘Optimal’ scaling laws are hard to get

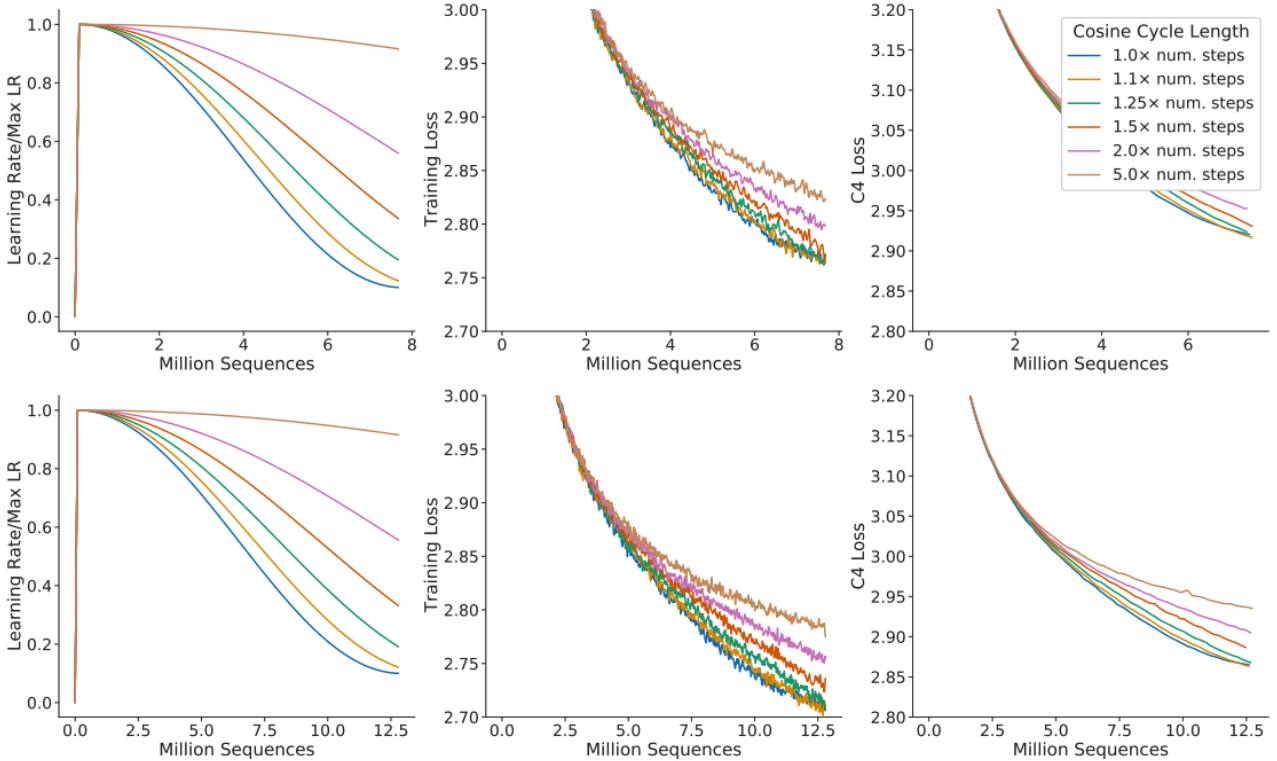
Rosenfeld, Kaplan both predict relationship of data, model and perf.

Chinchilla [Hoffman et al] argue these fits are quite off.



Hoffman+ 2022

Main difference – accounting for LR schedules



Chinchilla in depth – 3 methods

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

The chinchilla authors suggest 3 ways of fitting scaling laws – we'll go over each.

They mostly (minus method 3) suggest similar constants. More on this later..

Method 1 – minimum over runs.

Similar to the FLOPS figure on Kaplan –
the minimum over the union of all training curves is a power law.

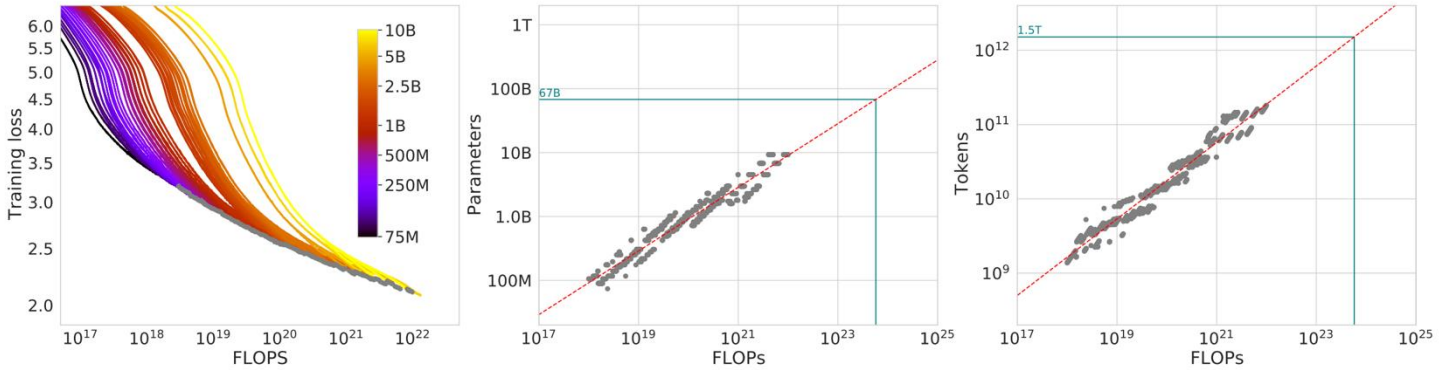


Figure 2 | **Training curve envelope.** On the **left** we show all of our different runs. We launched a range of model sizes going from 70M to 10B, each for four different cosine cycle lengths. From these curves, we extracted the envelope of minimal loss per FLOP, and we used these points to estimate the optimal model size (**center**) for a given compute budget and the optimal number of training tokens (**right**). In green, we show projections of optimal model size and training token count based on the number of FLOPs used to train *Gopher* (5.76×10^{23}).

Method 2 - IsoFLOPS

Pick a range of FLOP budgets, vary the total parameter count, take the min over these convex shapes. The minima form a power law.

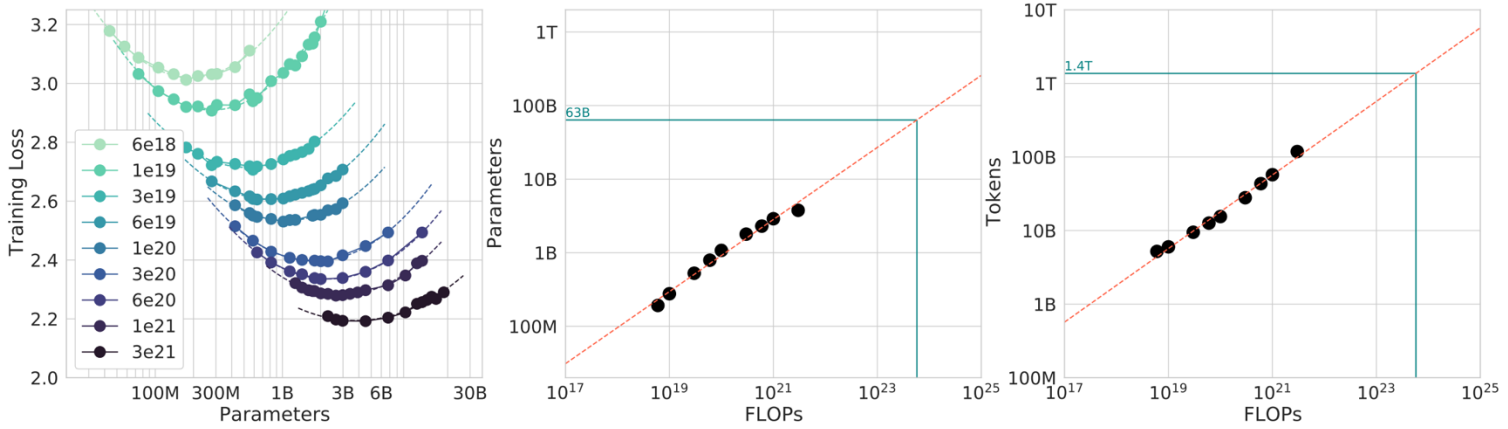


Figure 3 | **IsoFLOP curves.** For various model sizes, we choose the number of training tokens such that the final FLOPs is a constant. The cosine cycle length is set to match the target FLOP count. We find a clear valley in loss, meaning that for a given FLOP budget there is an optimal model to train (**left**). Using the location of these valleys, we project optimal model size and number of tokens for larger models (**center** and **right**). In green, we show the estimated number of parameters and tokens for an *optimal* model trained with the compute budget of *Gopher*.

Method 3 – Joint fits

Run a bunch of models on the size-data grid. Use least squares to fit a joint scaling law

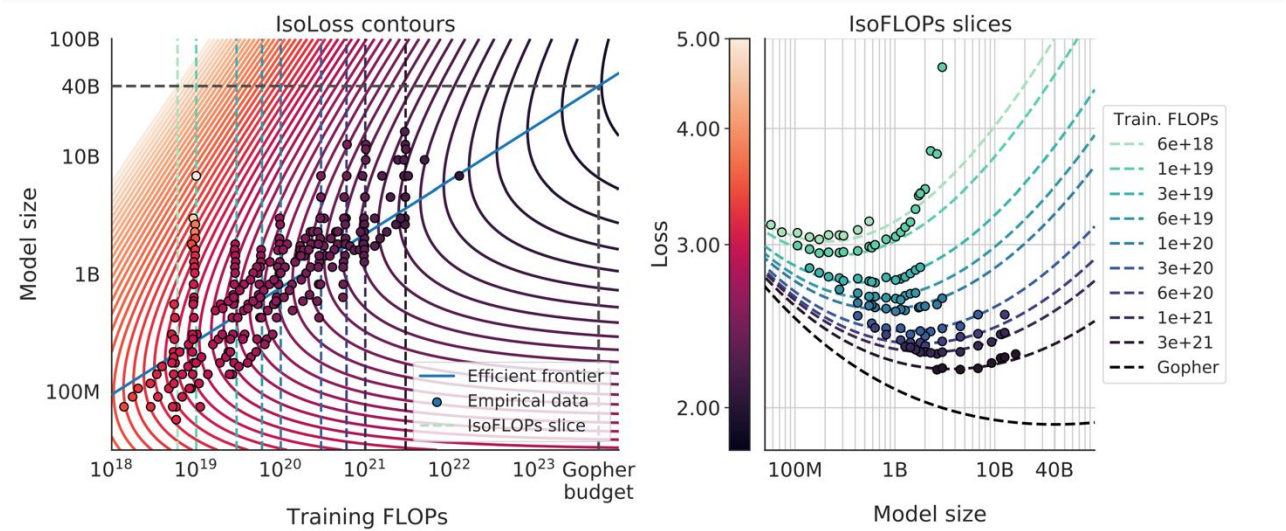
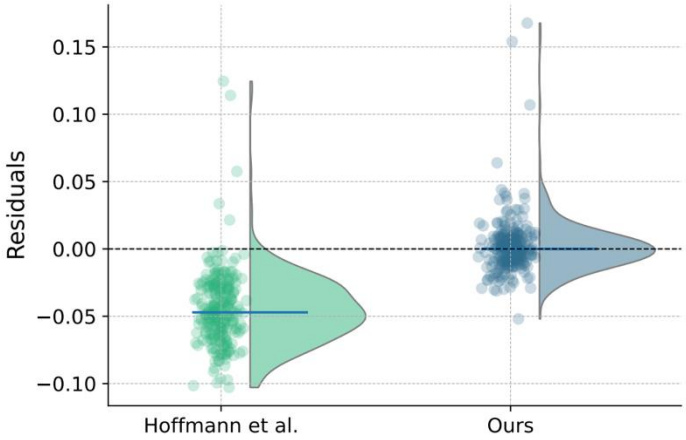
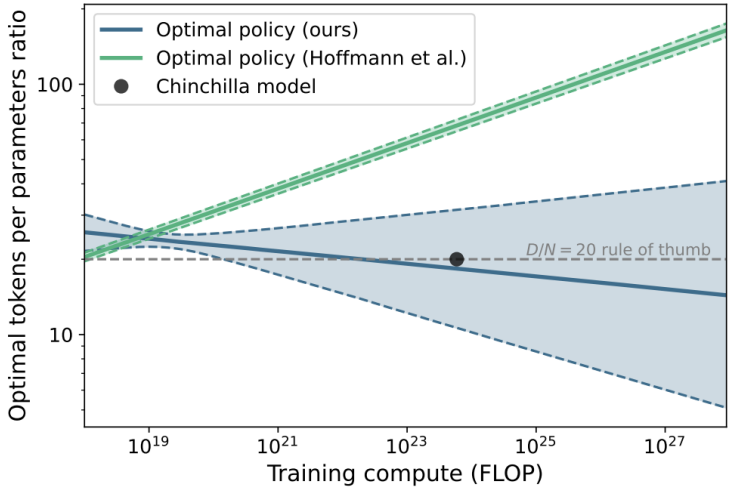


Figure 4 | **Parametric fit.** We fit a parametric modelling of the loss $\hat{L}(N, D)$ and display contour (**left**) and isoFLOP slices (**right**). For each isoFLOP slice, we include a corresponding dashed line in the left plot. In the left plot, we show the efficient frontier in blue, which is a line in log-log space. Specifically, the curve goes through each iso-loss contour at the point with the fewest FLOPs. We project the optimal model size given the *Gopher* FLOP budget to be 40B parameters.

Fun addendum – errors in chinchilla method 3

Note that this method three was likely flawed in the original paper. Some authors did data forensics, recovered the raw data, and re-did the fit and got results more consistent with methods 1 and 2





[Besiroglu et al 2024]

Important note – train-optimal may not be what you want

Chinchilla aims to tell you what gives the best model for fixed training compute..

But most of the compute in a real deployment is inference.. So we should ‘over’ train

- **GPT3** – 2 tokens / param
- **Chinchilla** – 20 tokens / param
- **LLaMA65B** – 22 tokens / param
- **Llama 2 70B** – 29 tokens / param
- **Mistral 7B** – 110 tokens / param
- **Llama 3 70B** – 215 tokens / param

The more usage we expect, the more it becomes worth it to pay the upfront cost

Recent example for different (diffusion) models

Methods like IsoFLOPS are pretty easy to execute, and our group has replicated these results

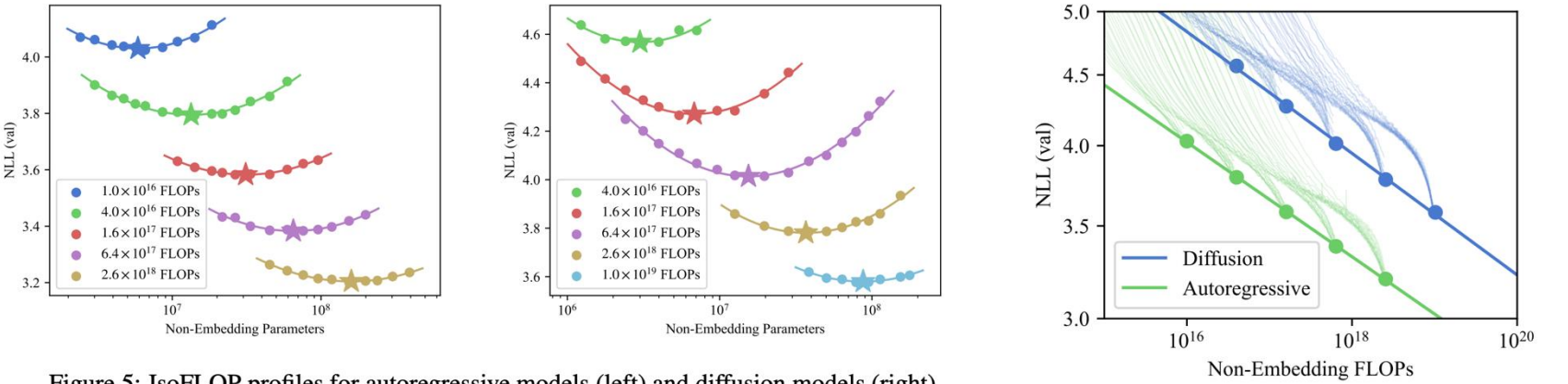


Figure 5: IsoFLOP profiles for autoregressive models (left) and diffusion models (right).

Gulrajani+ 2023.

Scaling laws for models and compute

Log-linearity extends to model parameters and compute!

Lets us set the following based on small models

- Pick optimizer
- Pick architecture and model sizes

Also lets us make smart resource tradeoffs

- Big models vs more data?

Recap: scaling laws – surprising and useful!

- **Data scaling**: understand how data affects models, clean theory
- **Model scaling**: dramatically reduce costs for training
- **Scaling as prediction:** understand what problems can be ‘brute forced’