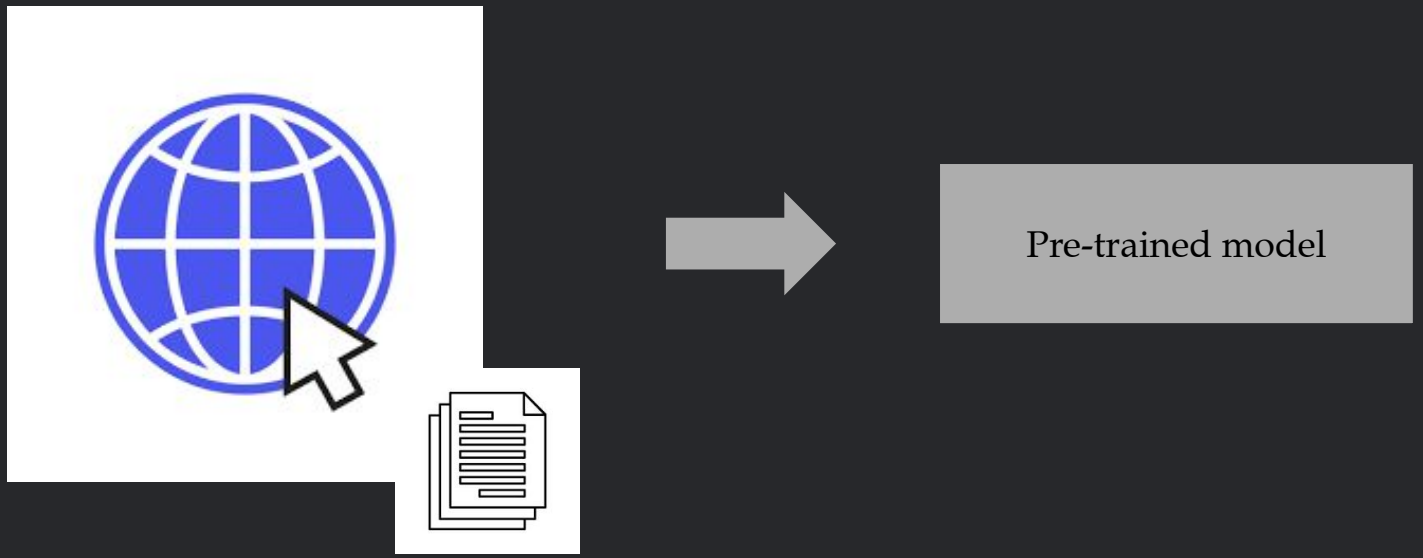
Pre-training under infinite compute

Konwoo Kim[∞], Suhas Kotha[∞], Percy Liang, Tatsu Hashimoto

Motivation

Language modeling works well these days, largely thanks to pre-training models by fitting the distribution of internet text

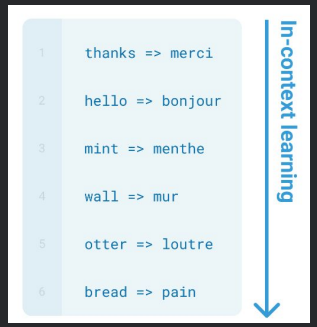


Motivation

Better pre-trained models provide incredibly useful initializations of intelligence that can be used for many real world tasks

Motivation

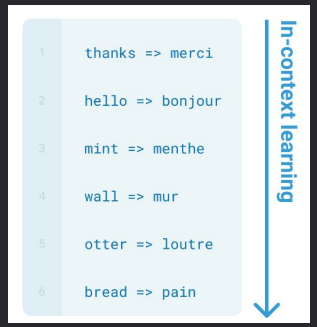
Better pre-trained models provide incredibly useful initializations of intelligence that can be used for many real world tasks



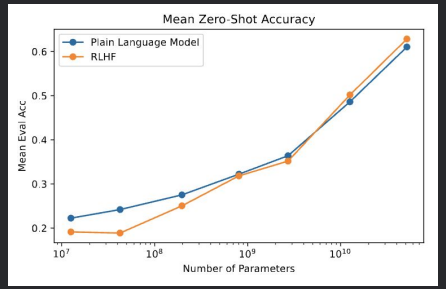
GPT-3 was necessary for models that can do in-context learning

Motivation

Better pre-trained models provide incredibly useful initializations of intelligence that can be used for many real world tasks



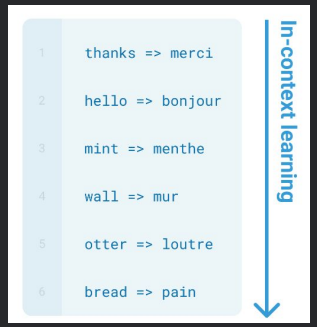
GPT-3 was necessary for models that can do in-context learning



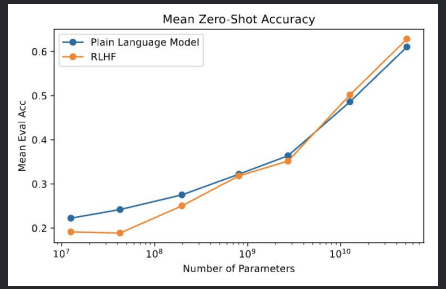
Anthropic's base models were necessary to benefit from alignment

Motivation

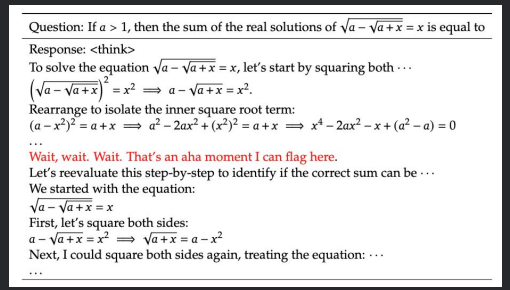
Better pre-trained models provide incredibly useful initializations of intelligence that can be used for many real world tasks



GPT-3 was necessary for models that can do in-context learning



Anthropic's base models were necessary to benefit from alignment


Question: If $a > 1$, then the sum of the real solutions of $\sqrt{a-\sqrt{a+x}} = x$ is equal to

Response: <think>

To solve the equation $\sqrt{a-\sqrt{a+x}} = x$, let's start by squaring both $\cdots$

$\left(\sqrt{a-\sqrt{a+x}}\right)^2 = x^2 \implies a-\sqrt{a+x} = x^2$.

Rearrange to isolate the inner square root term:

$(a-x^2)^2 = a+x \implies a^2-2ax^2+(x^2)^2 = a+x \implies x^4-2ax^2-x+(a^2-a) = 0$

$\cdots$

Wait, wait. Wait. That's an aha moment I can flag here.

Let's reevaluate this step-by-step to identify if the correct sum can be $\cdots$

We started with the equation:

$\sqrt{a-\sqrt{a+x}} = x$

First, let's square both sides:

$a-\sqrt{a+x} = x^2 \implies \sqrt{a+x} = a-x^2$

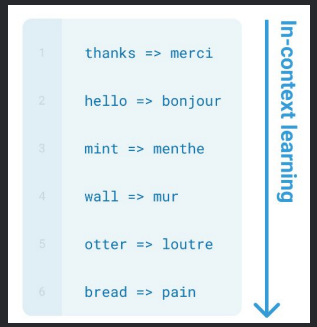
Next, I could square both sides again, treating the equation: $\cdots$

$\cdots$

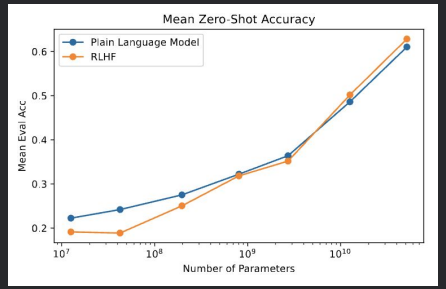
DeepSeek V3 was necessary for models that reason after RL

Motivation

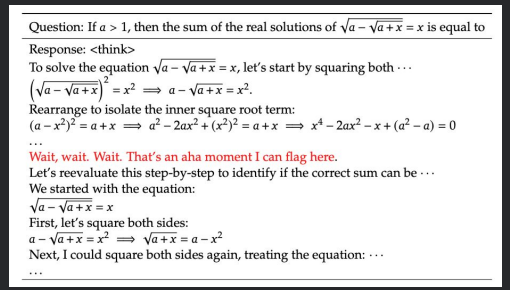
Better pre-trained models provide incredibly useful initializations of intelligence that can be used for many real world tasks



GPT-3 was necessary for models that can do in-context learning



Anthropic's base models were necessary to benefit from alignment



DeepSeek V3 was necessary for models that reason after RL

Current scaling practice

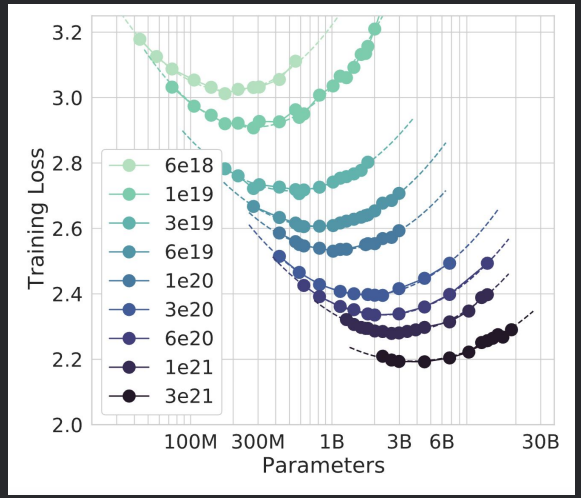
Pre-training is expensive and has historically focused on training the best model subject to various compute constraints, resulting in predictable scaling recipes

Current scaling practice

Pre-training is expensive and has historically focused on training the best model subject to various compute constraints, resulting in predictable scaling recipes

Example 1: Train compute

Chinchilla scaling: for a fixed compute budget, set parameter count N and token count D in a 1:20 ratio with no repetitions



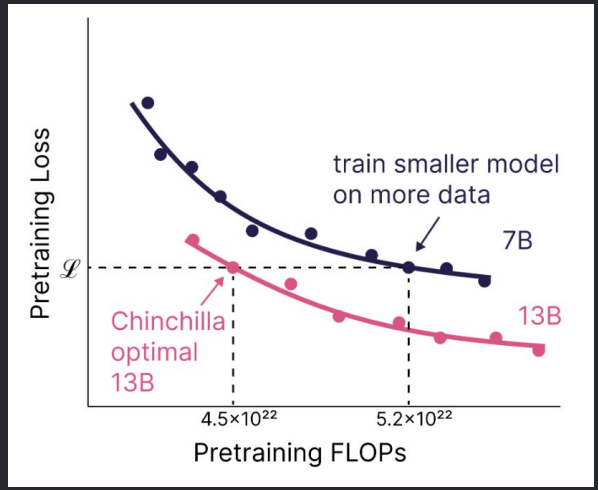
Kaplan et al, 2020 ; Hoffman et al, 2022

Current scaling practice

Pre-training is expensive and has historically focused on training the best model subject to various compute constraints, resulting in predictable scaling recipes

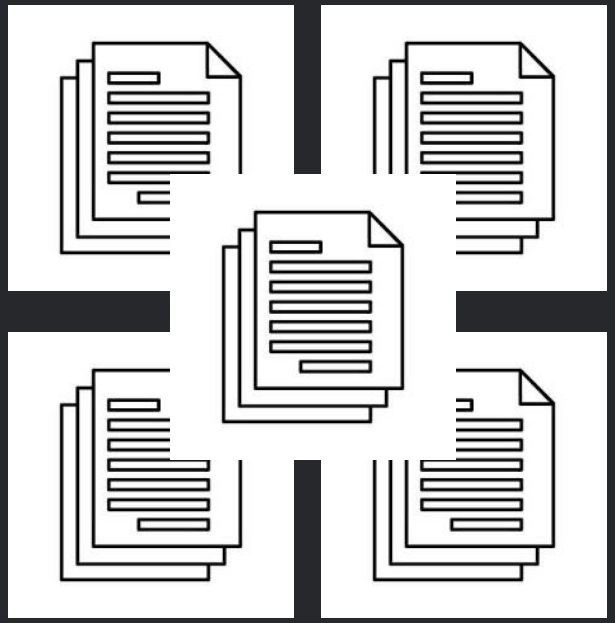
Example 2: Inference compute

To lower inference costs with small models, common to over-train models with 1:2000 parameter to token ratio



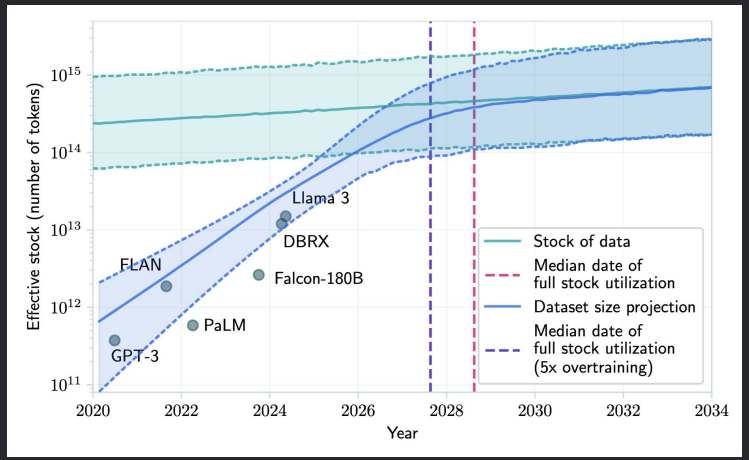
Sardana et al, 2025, Llama, Qwen, ...

Generally, current pre-training practice tells us how to pre-train given finite compute and as much data as needed



Problem: running out of pre-training data

The amount of data available on the internet grows quite slowly (approximately 3% per year), and we can not assume that we will always have access to more



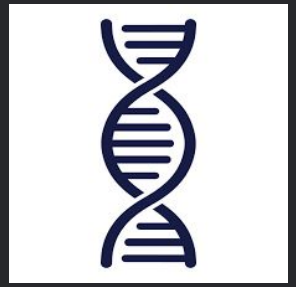
[Villalobos et al, 2024]

Many domains are naturally data-constrained

Beyond text, many distributions genuinely have few samples

Agents (i.e. web navigation trajectories)

Rare languages (i.e. Basque documents)



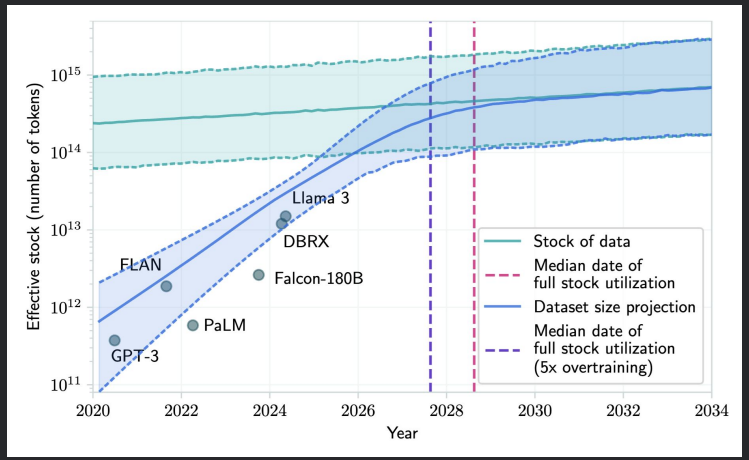
Biology (i.e. DNA sequences)

Personalization (i.e. personal information)

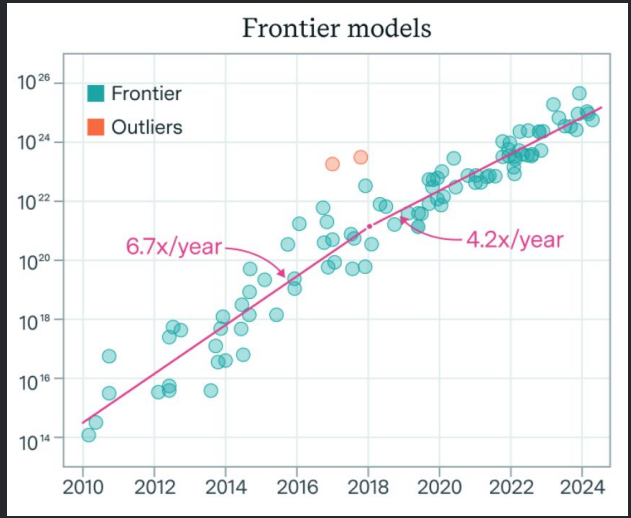
Robotics (i.e. robot trajectories)

Observation: compute continues to increase

On the other hand, compute continues to rapidly increase, with pre-training spending 5x more compute per year

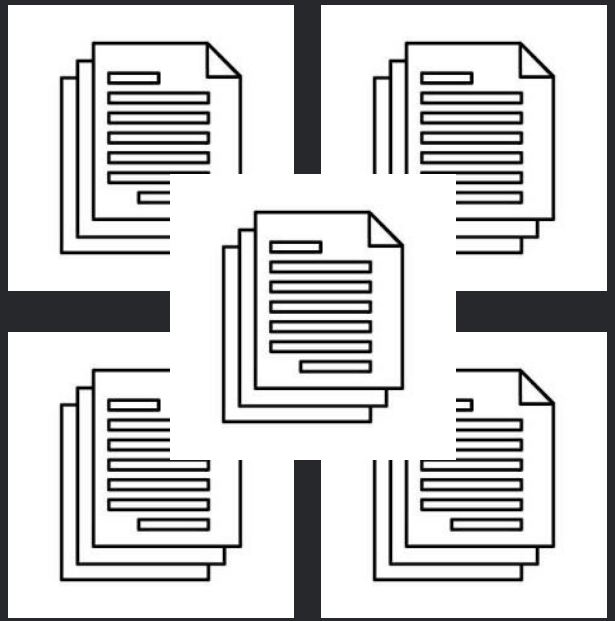


Villalobos et al, 2024

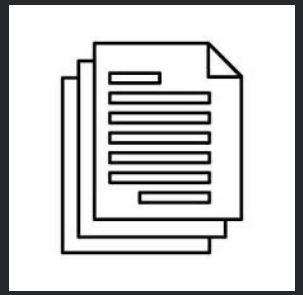


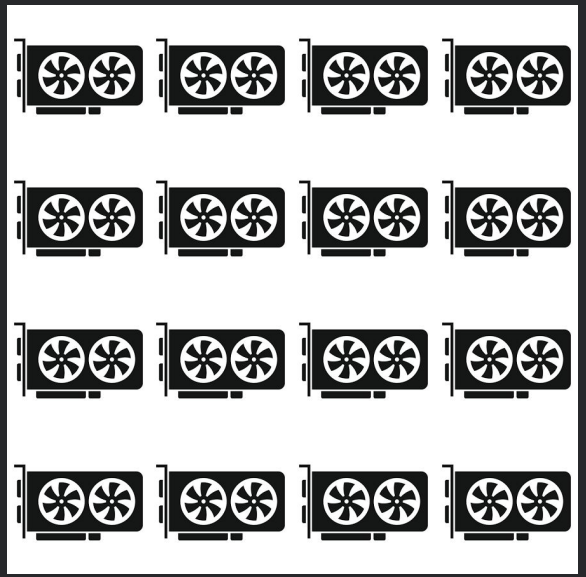
Sevilla and Roldán, 2024

Even if current pre-training practice tells us how to best use limited compute for unlimited data …



It gives little guidance on how to use compute when constrained by available data





How should one approach pre-training when constrained by data and unconstrained by compute?

Pre-training under infinite compute

How should one approach pre-training when constrained by data and unconstrained by compute?

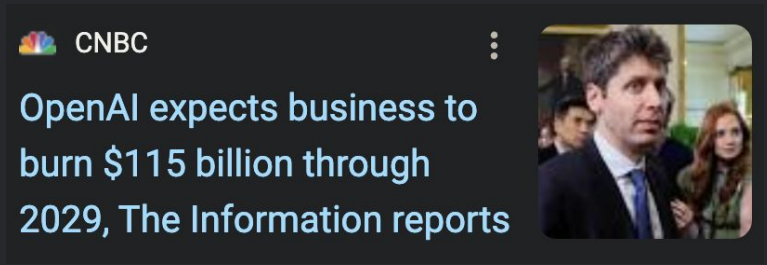
This is not very different from ML before the modern LLM era
- classical statistical learning analyzes algorithms with no compute constraints
- benchmarks like MNIST and Penn Tree Bank often have few samples

Pre-training under infinite compute

How should one approach pre-training when constrained by data and unconstrained by compute?

This is not very different from ML before the modern LLM era
- classical statistical learning analyzes algorithms with no compute constraints
- benchmarks like MNIST and Penn Tree Bank often have few samples

However, people are really willing to spend more compute to pre-train better models…



Pre-training under infinite compute

How should one approach pre-training when constrained by data and unconstrained by compute?

We revisit data efficiency with a fresh perspective from scaling laws

We find that current approaches overfit and can not leverage more compute, even if it was available

We design scaling recipes that monotonically decrease loss with clean power law scaling. We estimate the best possible performance via the *asymptote* of our scaling laws, predicting whether this algorithm will be useful in the future

Outline

1. Current approaches overfit	(epoching + parameter scaling)
2, 3, 4. Designing better recipes in infinite compute utopia	(regularization, ensembling, limits)
5, 6, 7. Demonstrating practicality of our interventions	(data scaling, distillation, evals)

0. Setting

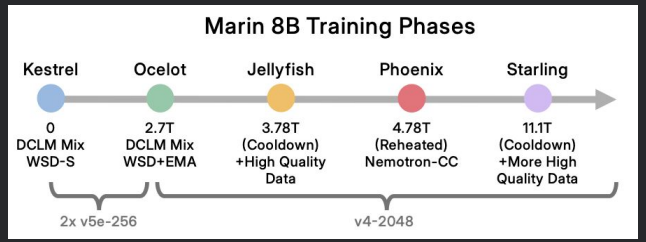
0. Setting

To simulate a data-constrained future, we limit our algorithms to a small number of pre-training tokens (D = 200M), ablated in Section 7

0. Setting

To simulate a data-constrained future, we limit our algorithms to a small number of pre-training tokens (D = 200M), ablated in Section 7

We use an optimized pre-training recipe (transformer, AdamW, etc) via [Marin]



	Average
Marin 8B Base (Deeper Starling)	**66.6**
Llama 3.1 Base	65.3
OLMo 2 Base	64.9
MAP NEO 7B	59.5
Amber 7B	48.1

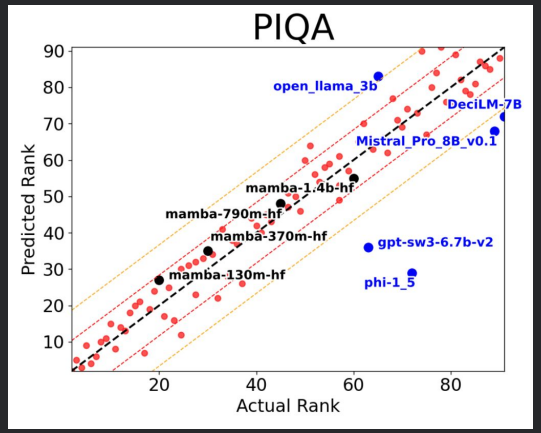
0. Setting

To simulate a data-constrained future, we limit our algorithms to a small number of pre-training tokens (D = 200M), ablated in Section 7

We use an optimized pre-training recipe (transformer, AdamW, etc) via Marin

Since we are assuming infinite compute, we train much larger models than normal, defaulting to a N = 300M parameter model (30x larger than Chinchilla)

0. Setting



We measure validation loss as this strongly correlates with pre-training quality [Section 6 ; Chen et al, 2025 ; Gadre et al, 2024 ; Thrush et al, 2025]

0. Setting

To simulate a data-constrained future, we limit our algorithms to a small number of pre-training tokens (D = 200M), ablated in Section 7

We use an optimized pre-training recipe (transformer, AdamW, etc) via Marin

Since we are assuming infinite compute, we train much larger models than normal, defaulting to a N = 300M parameter model (30x larger than Chinchilla)

We measure validation loss as this strongly correlates with pre-training quality [Section 6 ; Chen et al, 2025 ; Gadre et al, 2024 ; Thrush et al, 2025]

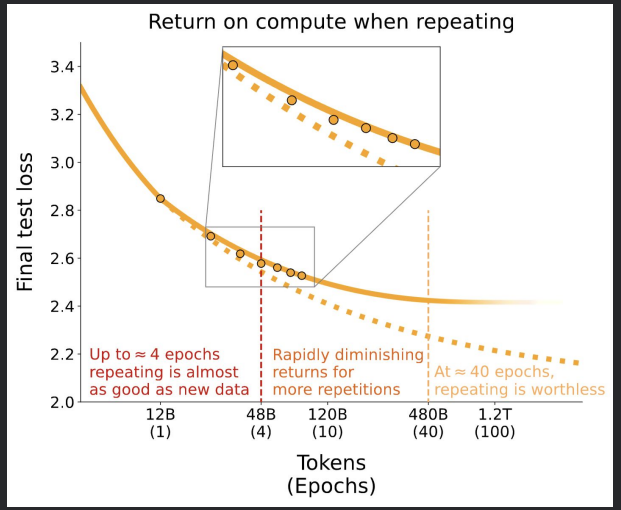
We do not allow for any human supervision (prompting, reward functions, external data, external models, etc.)

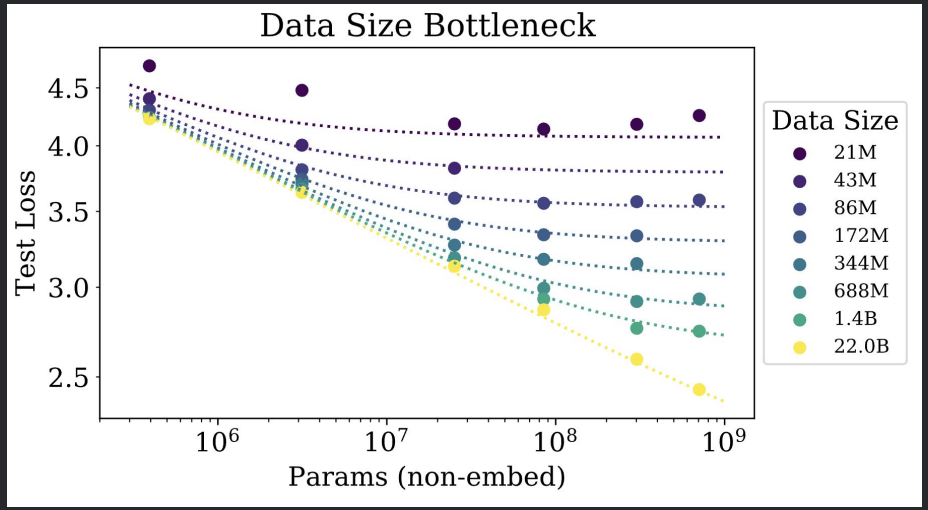
1. Current recipes

how well do existing methods handle data constraints?

1. Current recipes

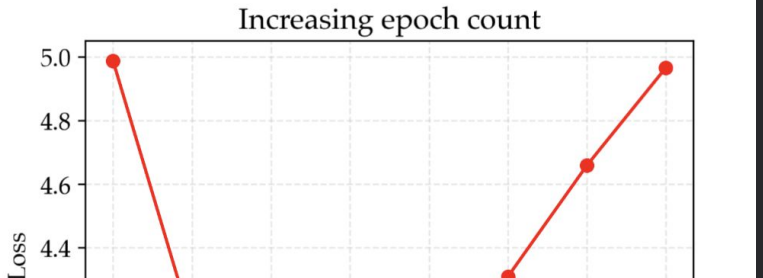
We start by considering some existing data-constrained pre-training recipes such as repeating the data [Muennighoff et al, 2023] and increasing parameter count [Kaplan et al, 2020 ; Hoffman et al, 2022]





1a. Current recipes: epoching

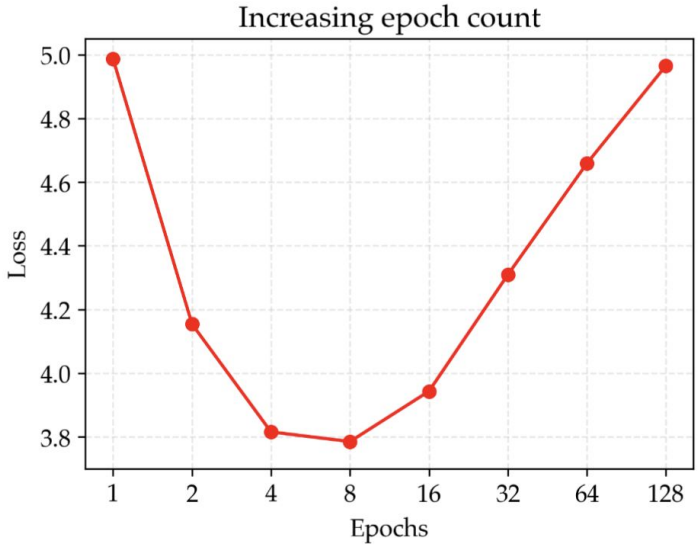
If we repeat the data too many times, we start overfitting



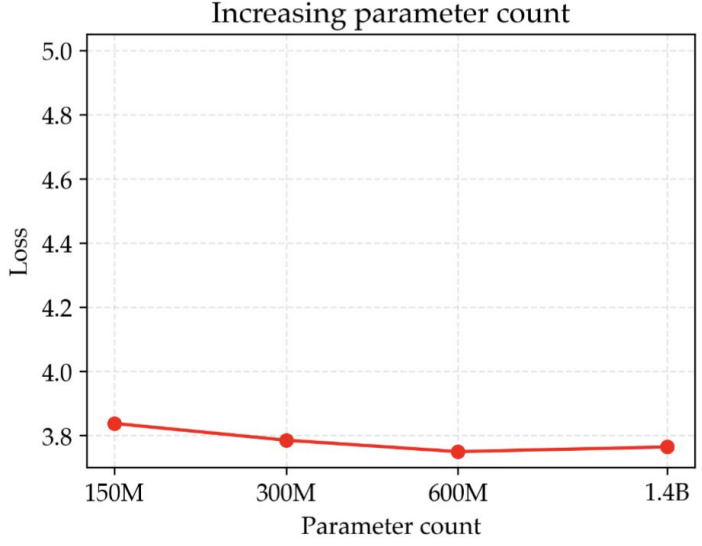
Tuned *H*	**1**	**8**	**128**
Learning rate	1e-3	1e-3	3e-3

1b. Current recipes: epoching + parameter scaling

Increasing the parameter count isn't very helpful, even after tuning epoch count



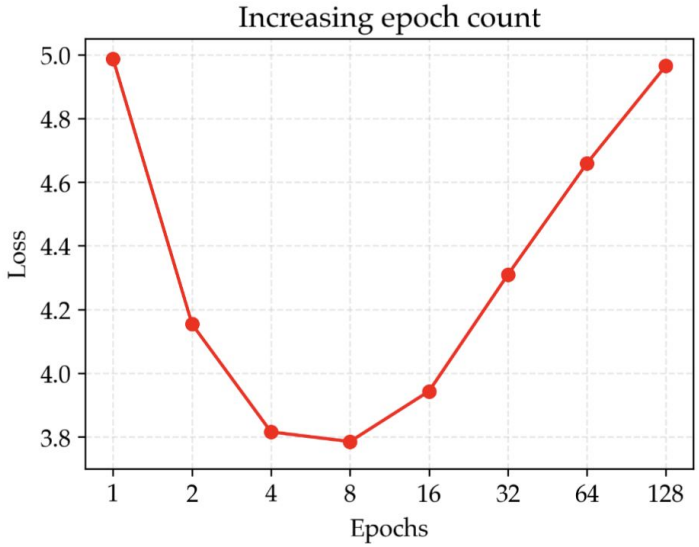
Tuned H	1	8	128
Learning rate	1e-3	1e-3	3e-3



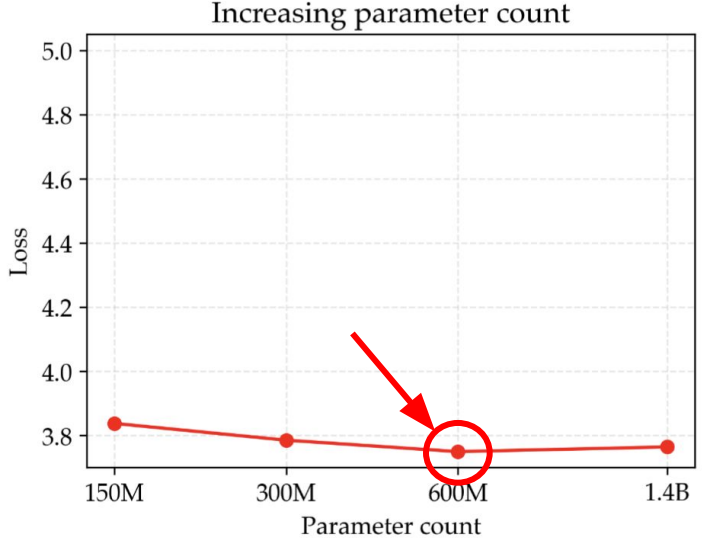
Tuned H	150M	300M	600M	1.4B
Learning rate	3e-3	1e-3	1e-3	3e-4
Epoch count	8	8	4	4

1b. Current recipes: epoching + parameter scaling

Is this the best possible performance under infinite compute?



Tuned H	1	8	128
Learning rate	1e-3	1e-3	3e-3



Tuned H	150M	300M	600M	1.4B
Learning rate	3e-3	1e-3	1e-3	3e-4
Epoch count	8	8	4	4

2. Regularized parameter scaling

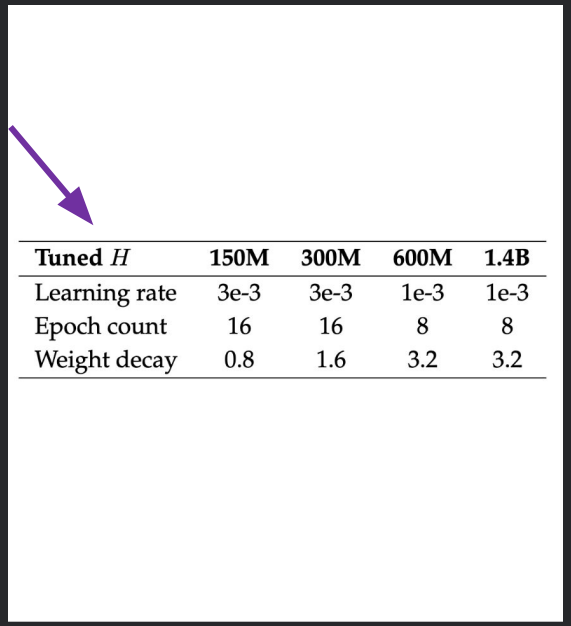
can we get stronger/monotone/clean scaling?

2. Regularized parameter scaling

We find that to get the best possible performance from these over-parameterized, epoched models, it is critical to regularize pre-training with much higher weight decay.

2. Regularized parameter scaling

We first jointly tune learning rate, epoch count, and weight decay for each parameter count by performing an expensive search for locally optimal hyper-parameters



Tuned H	150M	300M	600M	1.4B
Learning rate	3e-3	3e-3	1e-3	1e-3
Epoch count	16	16	8	8
Weight decay	0.8	1.6	3.2	3.2

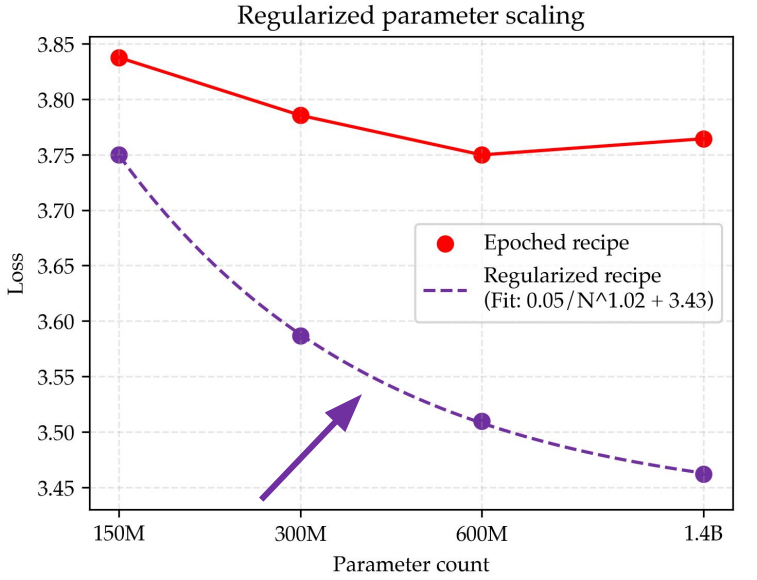
2. Regularized parameter scaling

The optimal weight decay can be **over 30 times** higher than standard practice of 0.1, likely because our models are now larger than the data and we epoch

Tuned H	**150M**	**300M**	**600M**	**1.4B**
Learning rate	3e-3	3e-3	1e-3	1e-3
Epoch count	16	16	8	8
Weight decay	0.8	1.6	3.2	3.2

2. Regularized parameter scaling

After regularization, our loss follows clean power law scaling!

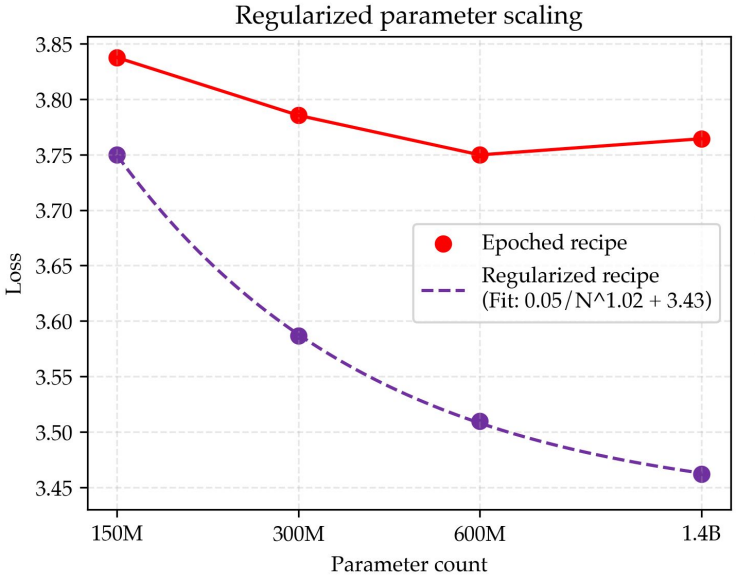


$$L = 0.05 / N^{1.02} + 3.43$$

Tuned H	150M	300M	600M	1.4B
Learning rate	3e-3	3e-3	1e-3	1e-3
Epoch count	16	16	8	8
Weight decay	0.8	1.6	3.2	3.2

2. Regularized parameter scaling

Our law has a model scaling exponent that is much faster than Chinchilla (1.02 vs 0.34)

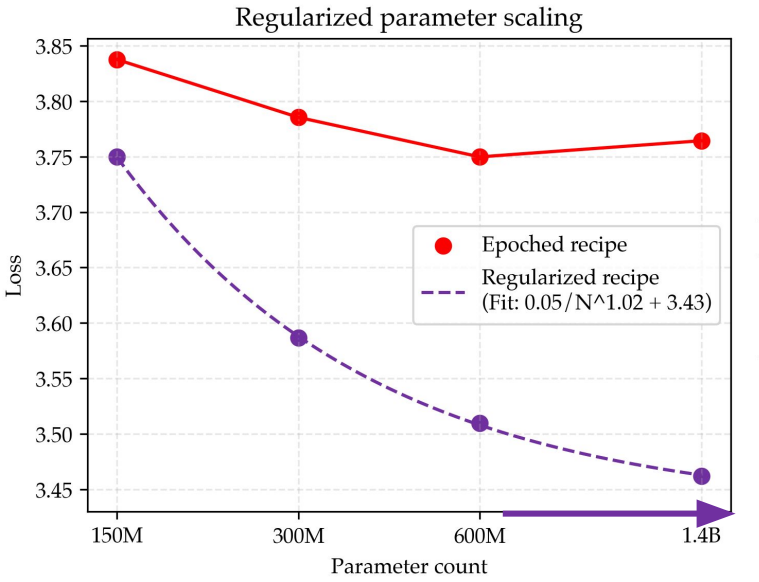


$$L = 0.05 / N^{1.02} + 3.43$$

Tuned H	150M	300M	600M	1.4B
Learning rate	3e-3	3e-3	1e-3	1e-3
Epoch count	16	16	8	8
Weight decay	0.8	1.6	3.2	3.2

2. Regularized parameter scaling

Thanks to our scaling law, we can estimate the best possible performance under infinite compute via the ***asymptote*** under infinite parameter count

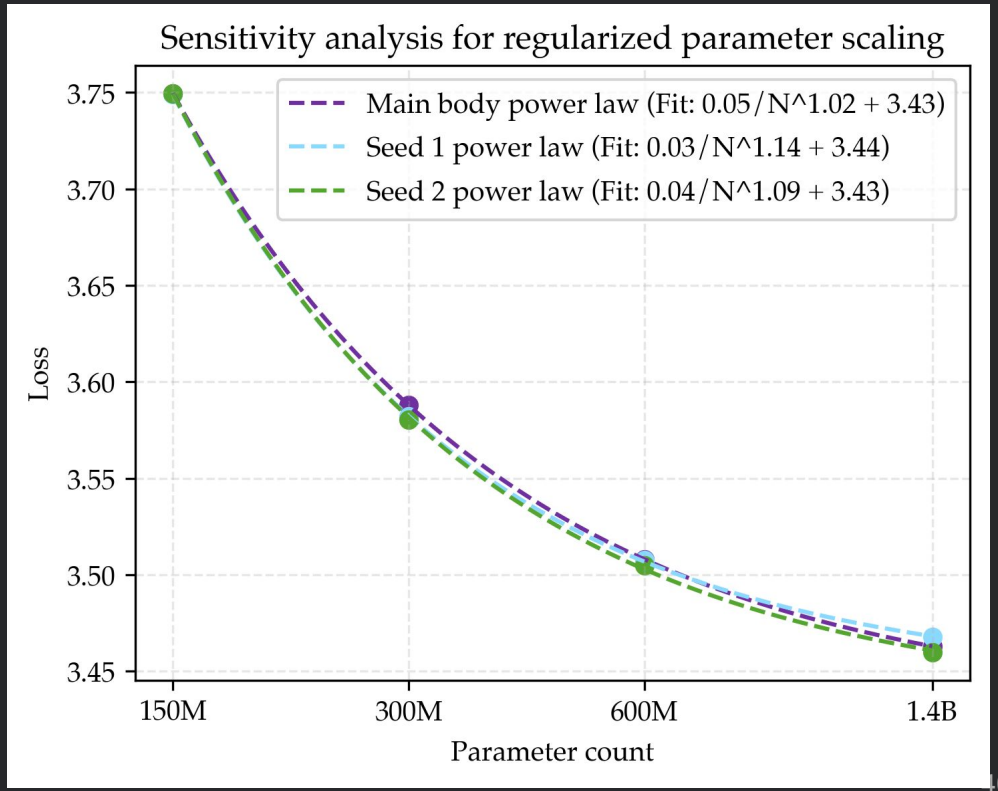


$$L = 0.05 / N^{1.02} + 3.43$$

Tuned H	150M	300M	600M	1.4B
Learning rate	3e-3	3e-3	1e-3	1e-3
Epoch count	16	16	8	8
Weight decay	0.8	1.6	3.2	3.2

Aside: Sensitivity Analysis

Our asymptote estimation seems reasonable in this case, most likely because we are so close to the asymptote



2. Regularized parameter scaling

Thanks to our scaling law, we can estimate the best possible performance under infinite compute via the ***asymptote*** under infinite parameter count

We are excited by using asymptotes to estimate the best performance of algorithms under infinite compute, which can hopefully anticipate the correct algorithms for a data-constrained future!

3. Ensemble scaling

can we improve upon the parameter scaling asymptote?

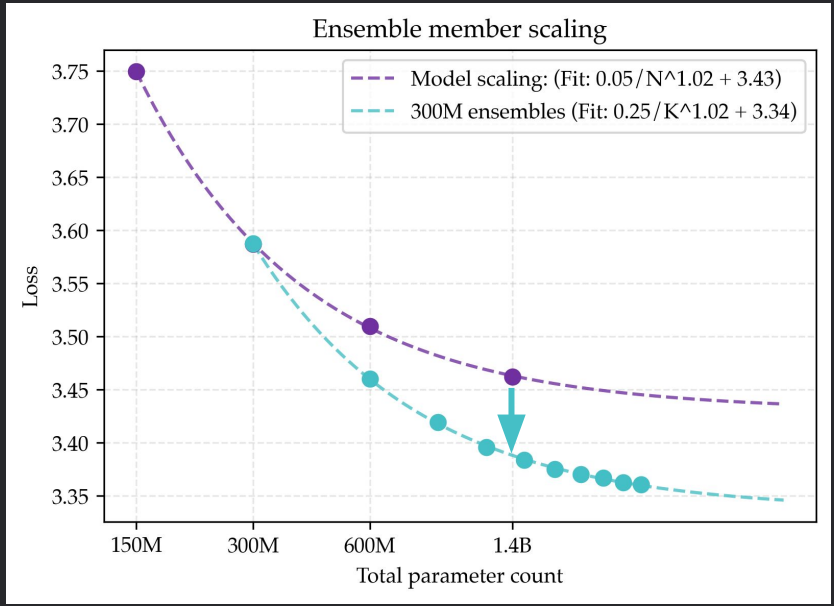
3. Ensemble member scaling

Parameter scaling is only one possible limit, are there others?

We investigate ensembling: instead of training bigger models, we train multiple models with different data orders + initializations and average logits at inference

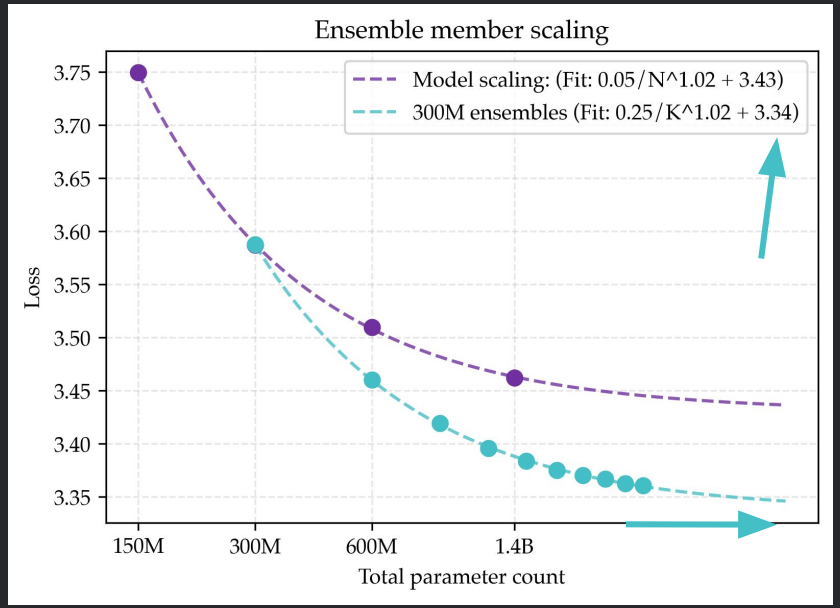
3. Ensemble member scaling: better asymptotes

Increasing ensemble member count out-performs increasing parameter count!
Instead of training bigger models, it's better to train multiple models



3. Ensemble member scaling: better asymptotes

We also get power law scaling for ensembles. The exponents are similar and the asymptote is lower, making it better with infinite compute



3. Ensemble member scaling: why are ensembles better?

Allen-Zhu and Li, 2023 suggest data is “multi-view”, where one of many different features correctly classifies the data, but the best performance comes from using all the features. Parameter scaling is biased to learning only one view of the data

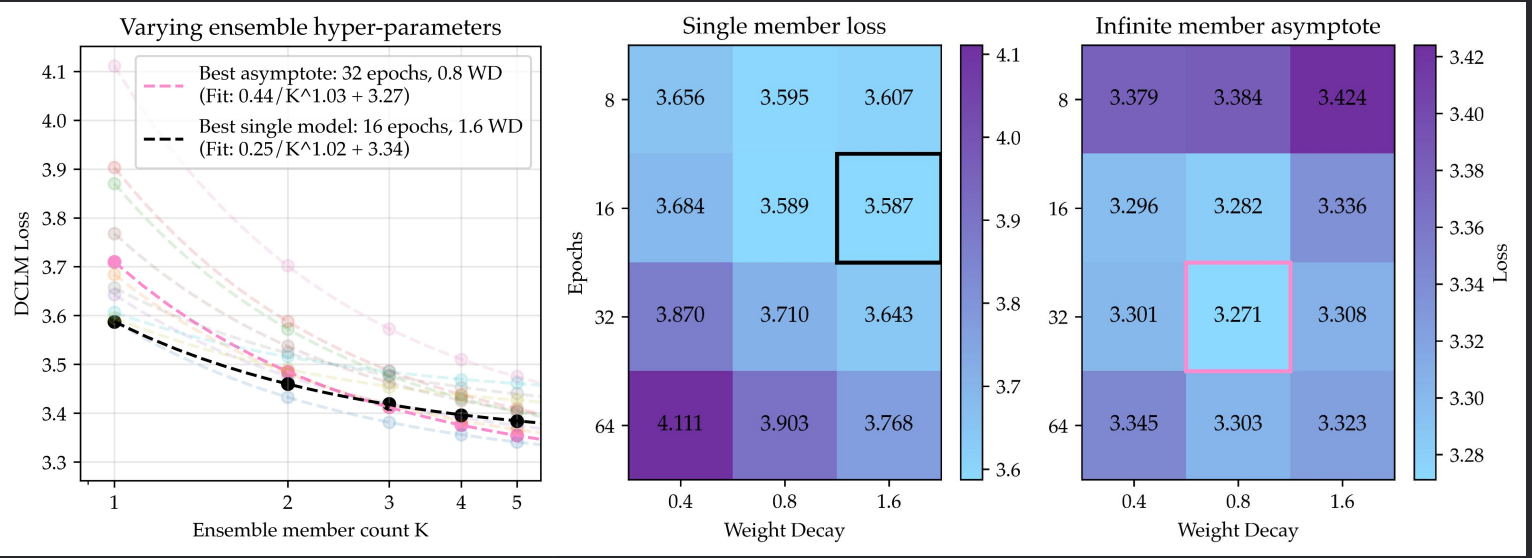
Ensembling alternatives

Mixture-of-Expert's: MoE's differentiate through the whole model and we do not expect them to learn all the features. For example, when we try training ten models in parallel following an ensemble architecture, it barely out-performs a single student and is much worse than increasing ensemble members

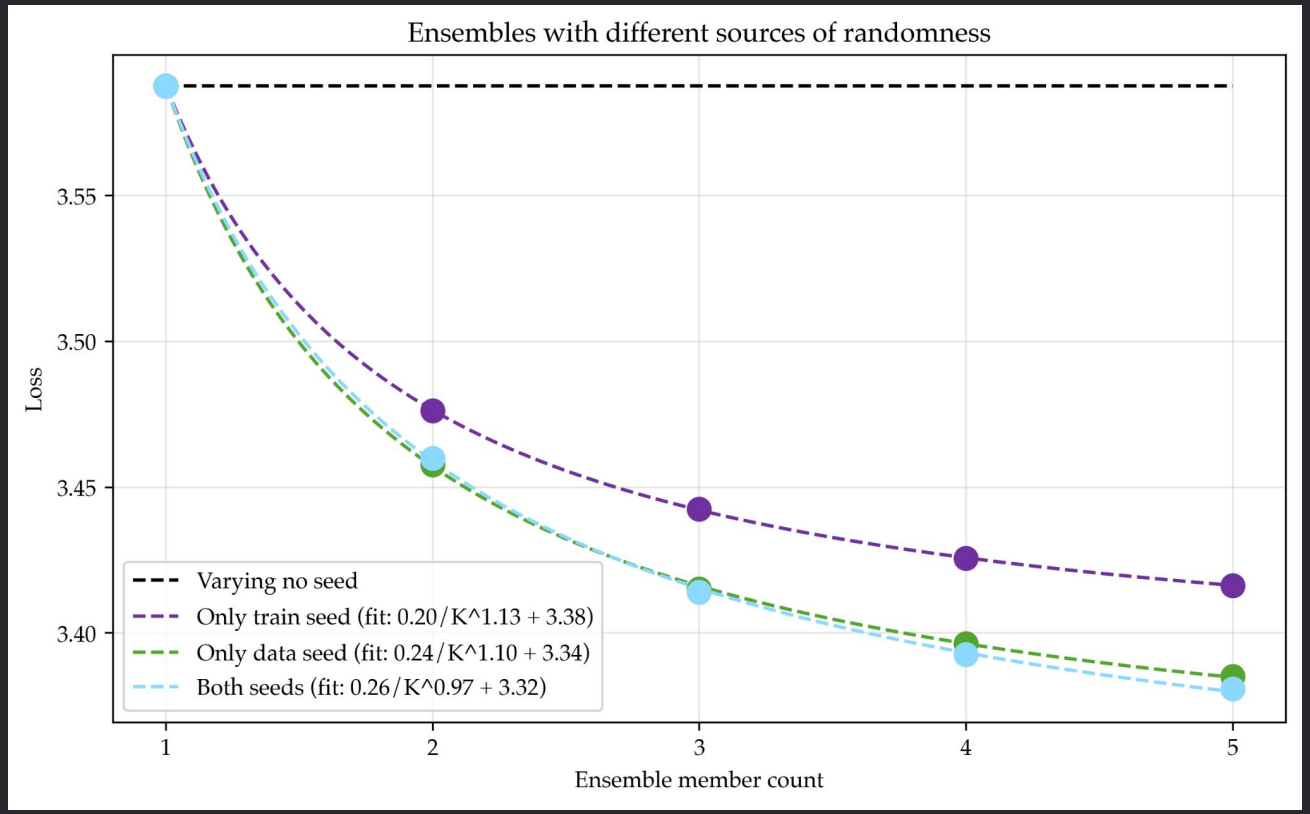
Weight averaging: Weight averaging only works at fine-tuning since it requires solutions to be in the same loss basin. When we try averaging our pre-trained models, it results in random guessing loss. Moreover, distillation also allows us to merge many members automatically for inference compute savings

3. Ensemble member scaling: why are ensembles better?

We find evidence for multi-view data when tuning ensemble members: we get better asymptotes when each member is trained with more epochs and less regularization, with each model over-fitting in a different way



Seed science



4. Composing all recipes

how far can we push data-efficiency with all of our recipes?

4. Composing both ensemble and parameter scaling

Previously, we were comparing parameter and ensemble scaling. However, there is nothing that stops us from composing them. We characterize the best possible performance of both by computing the double limit

4. Composing both ensemble and parameter scaling

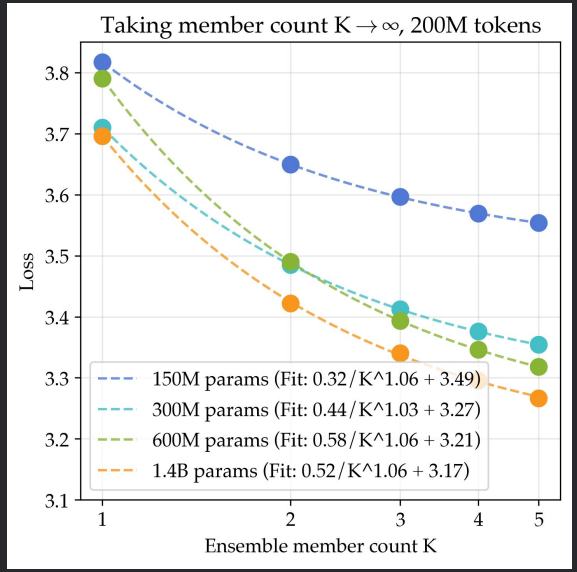
Previously, we were comparing parameter and ensemble scaling. However, there is nothing that stops us from composing them. We characterize the best possible performance of both by computing the double limit

$$\hat{\mathcal{L}}_D = \lim_{N\to\infty} \lim_{K\to\infty} \min_{H} \mathcal{L}\left(\mathcal{E}_{\mathcal{A}}\left(D, N, K, H\right)\right)$$

D tokens
N parameters
K ensemble members
H hyperparams (wd, lr, epochs)

4. Composing both ensemble and parameter scaling

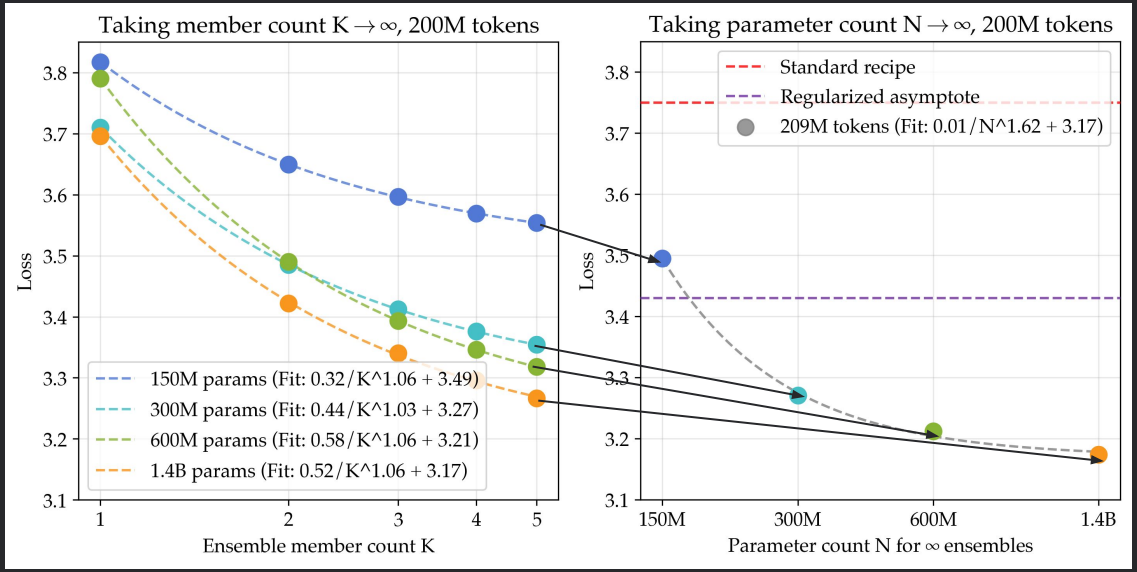
Left: We first send K → ∞ for fixed N, D with hparams selected for the asymptote



4. Composing both ensemble and parameter scaling

Left: We first send K → ∞ for fixed N, D with hparams selected for the asymptote

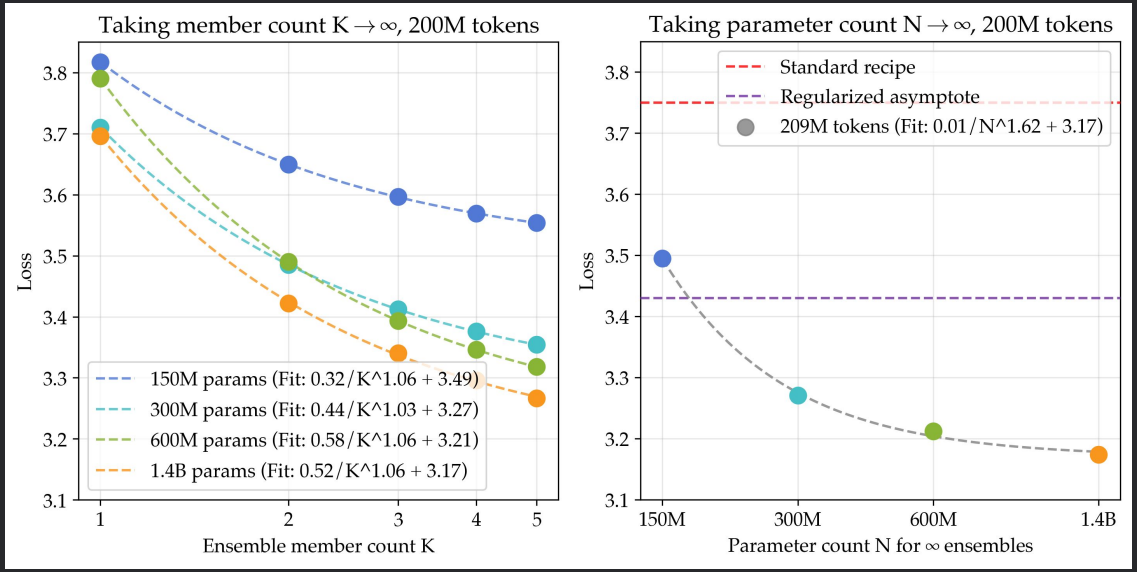
Right: We then send N → ∞ for fixed D and achieve a lower asymptote of 3.17



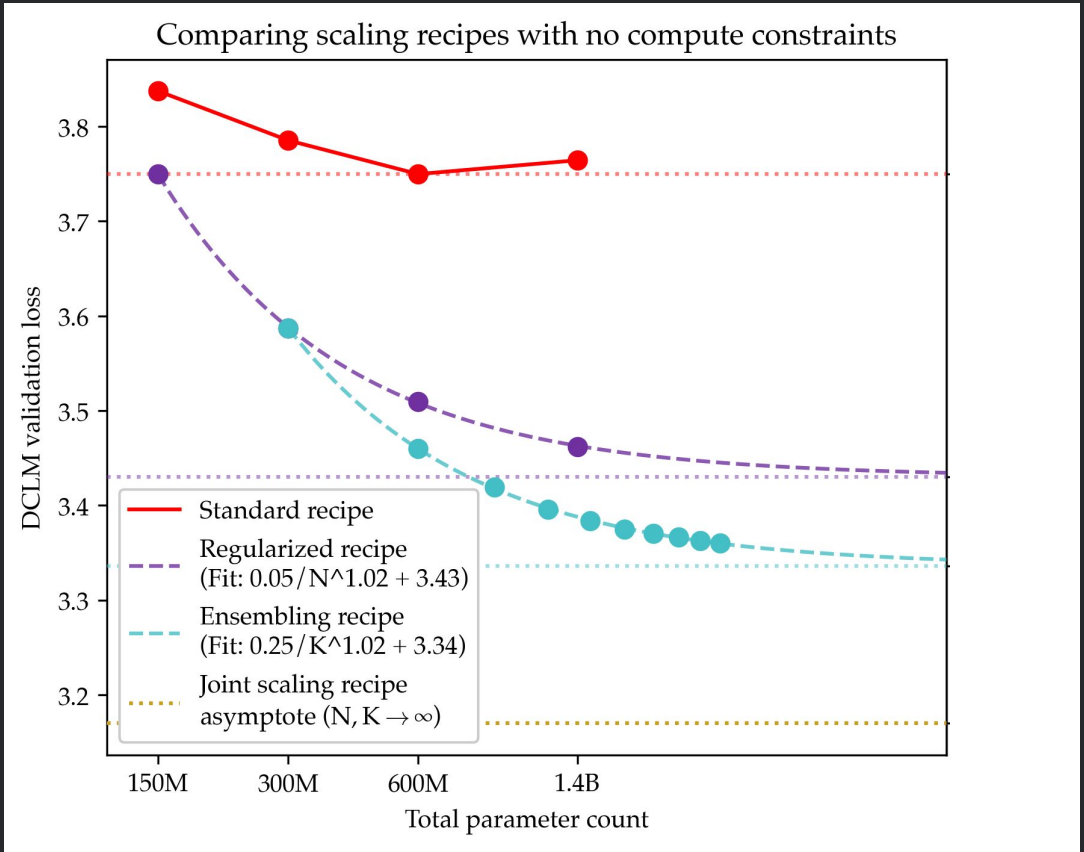
4. Composing both ensemble and parameter scaling

Left: We first send K → ∞ for fixed N, D with hparams selected for the asymptote

Right: We then send N → ∞ for fixed D and achieve a lower asymptote of 3.17



Summary of scaling recipes at 200M tokens



1. Current approaches over-fit

2. Regularization has a scaling law + asymptote

3. Ensembling has a lower asymptote

4. We can compose our recipes

5. Scaling seed token count

do our methods only help at small scale?

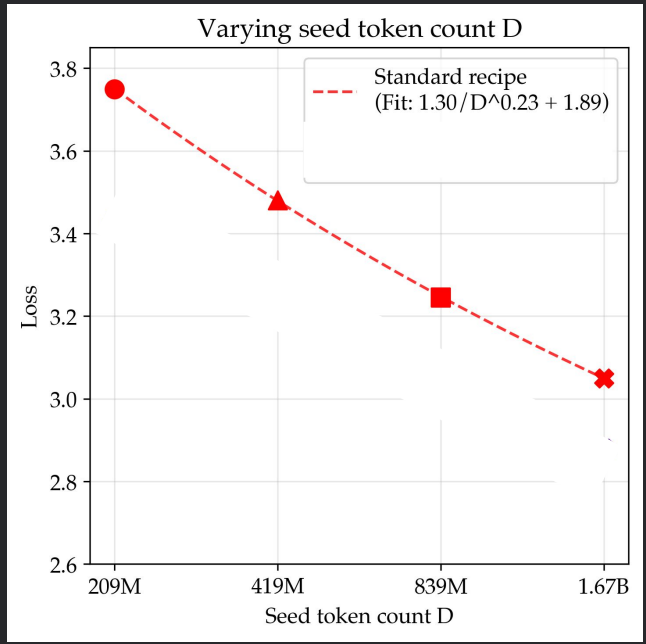
5. Scaling seed token count

Do these interventions only help at 200M tokens, or do they also help at larger token counts?

We scale up our experiments by estimating the best possible loss of epoching, regularizing, and ensembling at each token scale. We can then measure how much better our method is at different token scales.

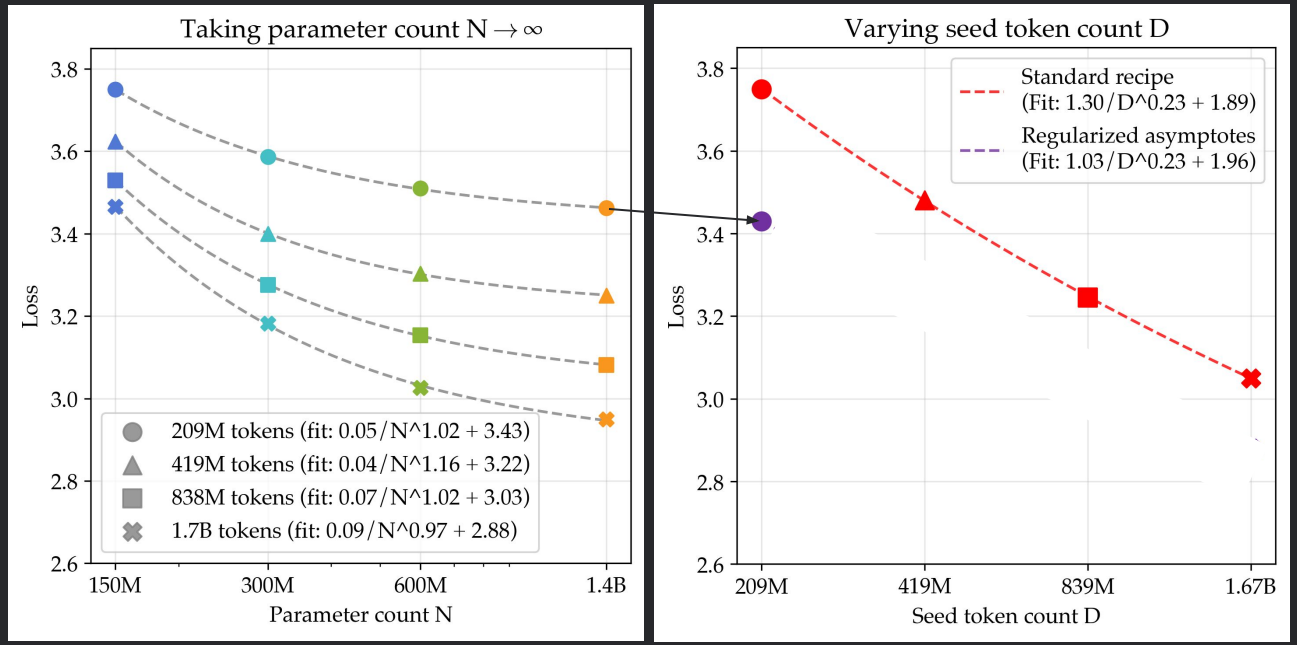
5. Scaling seed token count

For each seed token count, we find the best possible loss of the standard recipe and regularized recipe (asymptote)



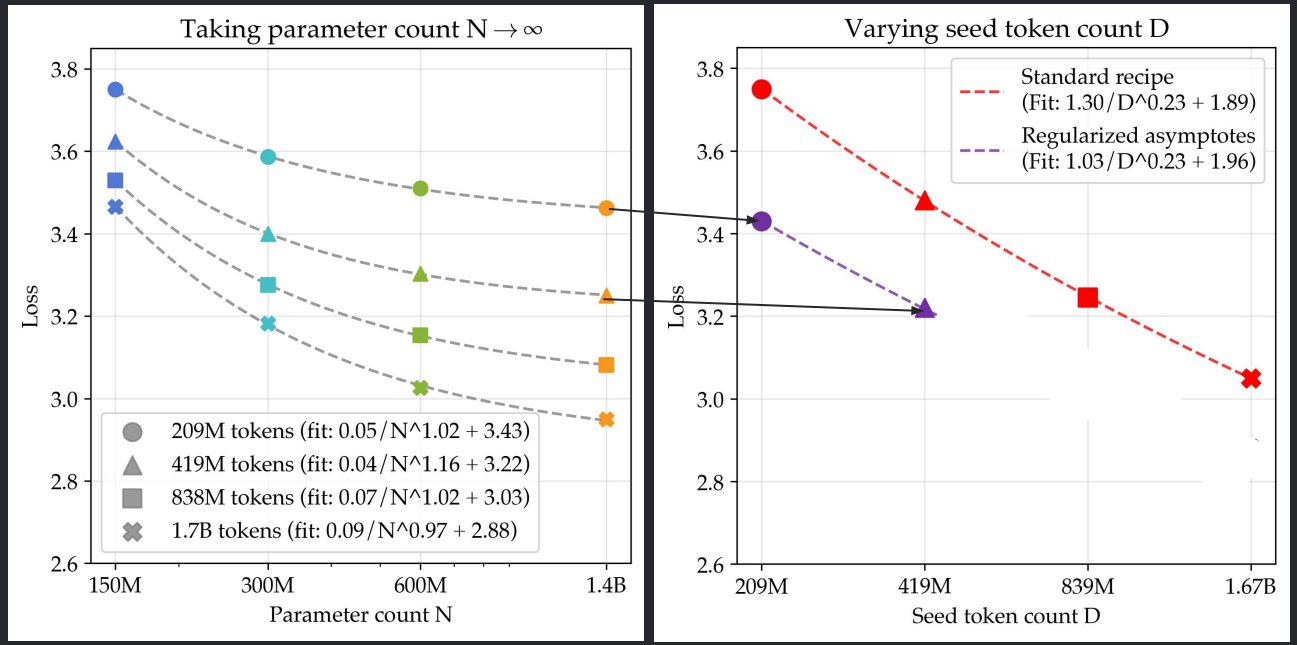
5. Scaling seed token count

For each seed token count, we find the best possible loss of the standard recipe and regularized recipe (asymptote)



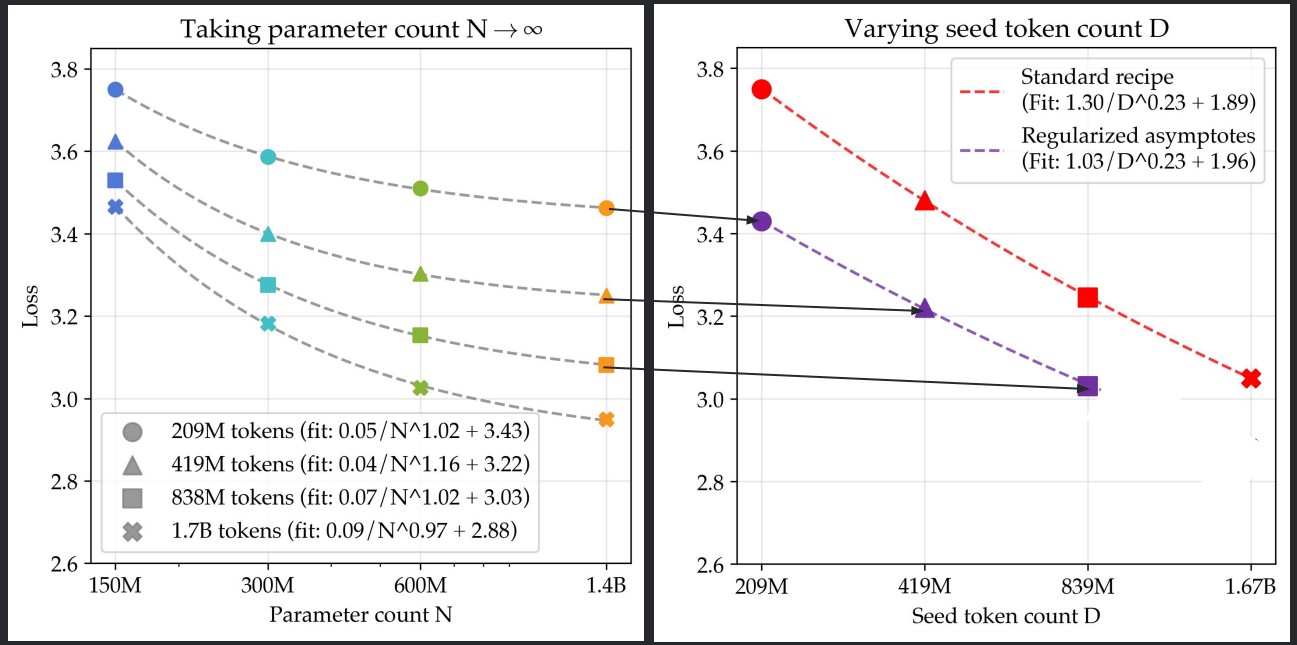
5. Scaling seed token count

For each seed token count, we find the best possible loss of the standard recipe and regularized recipe (asymptote)



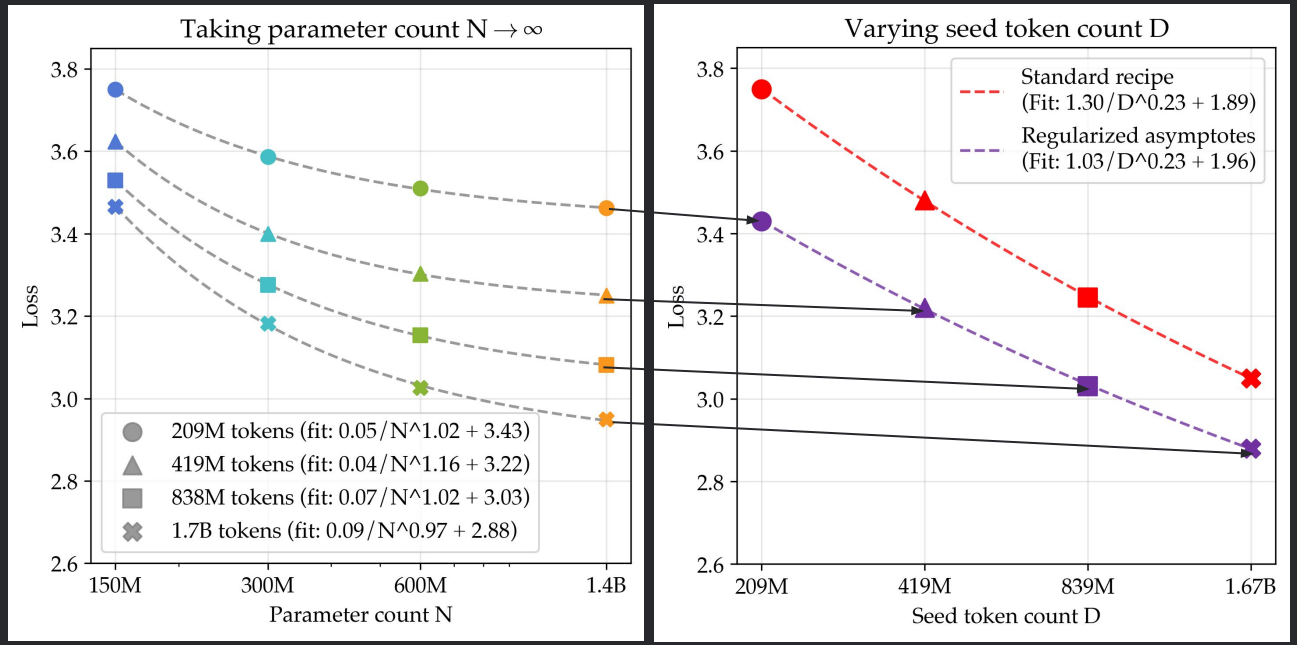
5. Scaling seed token count

For each seed token count, we find the best possible loss of the standard recipe and regularized recipe (asymptote)



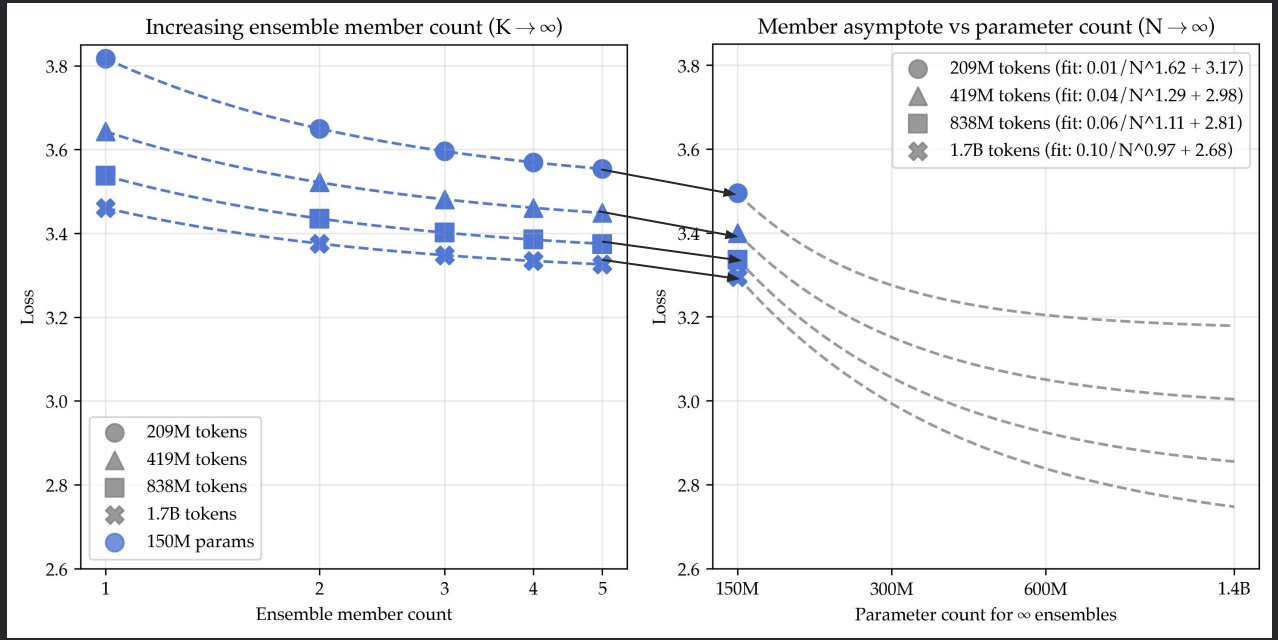
5. Scaling seed token count

For each seed token count, we find the best possible loss of the standard recipe and regularized recipe (asymptote)



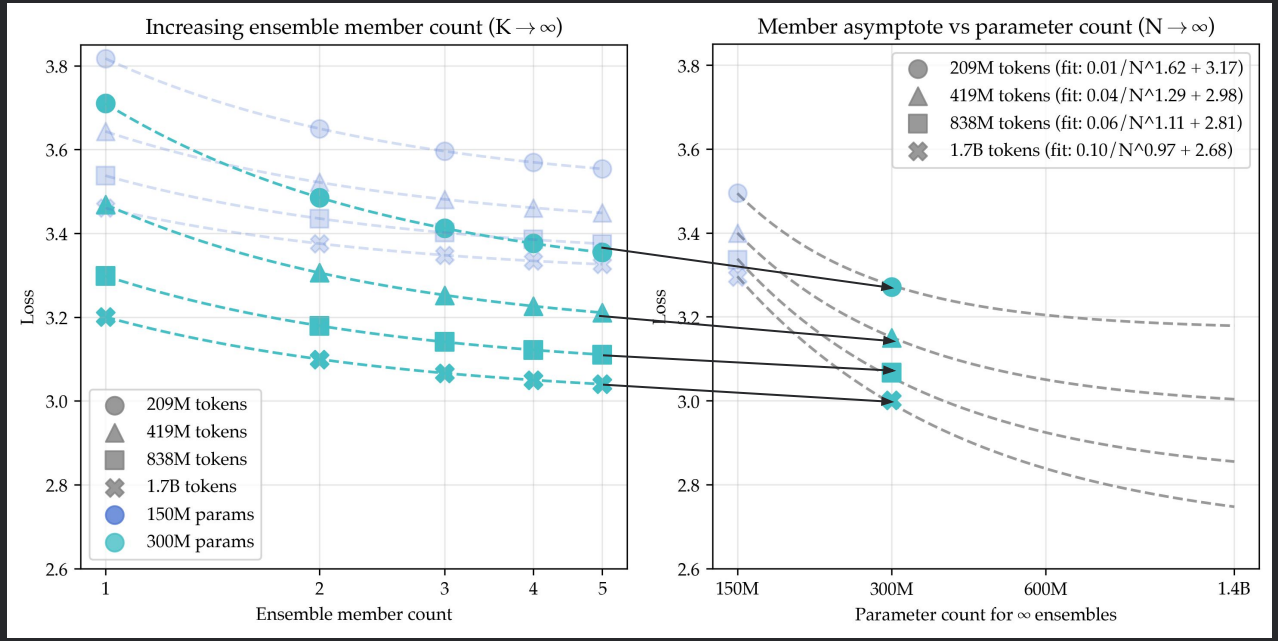
5. Scaling seed token count

For ensembles, for each D, we first send K → ∞ for fixed N with hparams selected for the asymptote.



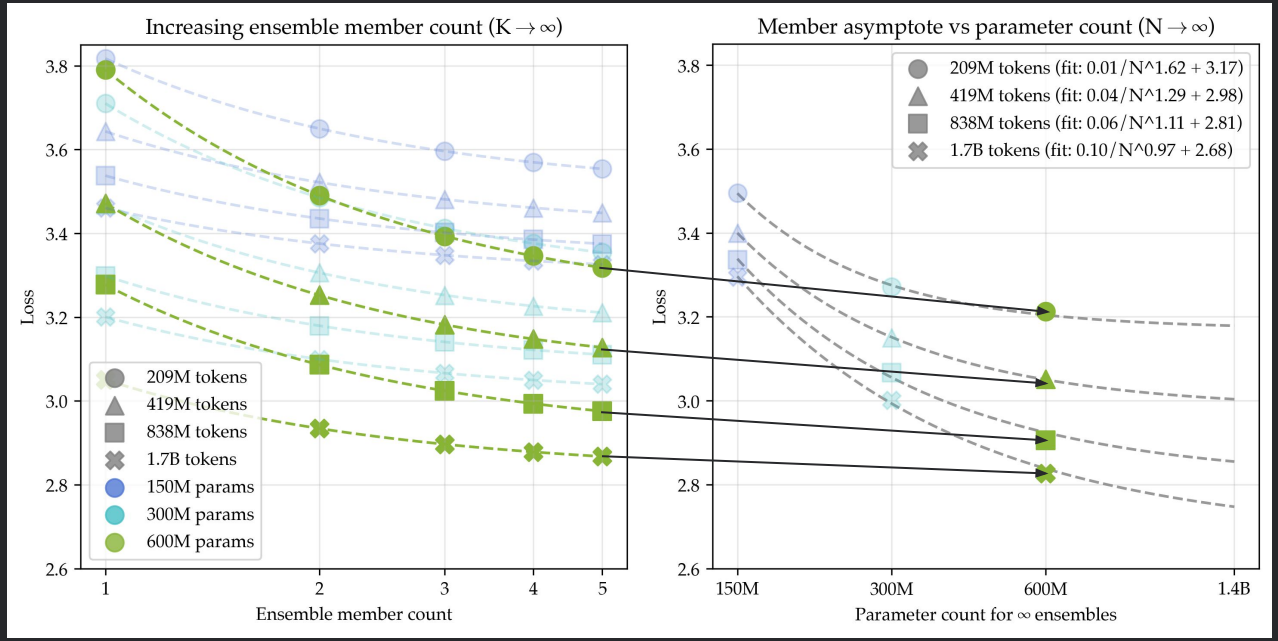
5. Scaling seed token count

For ensembles, for each D, we first send K → ∞ for fixed N with hparams selected for the asymptote.



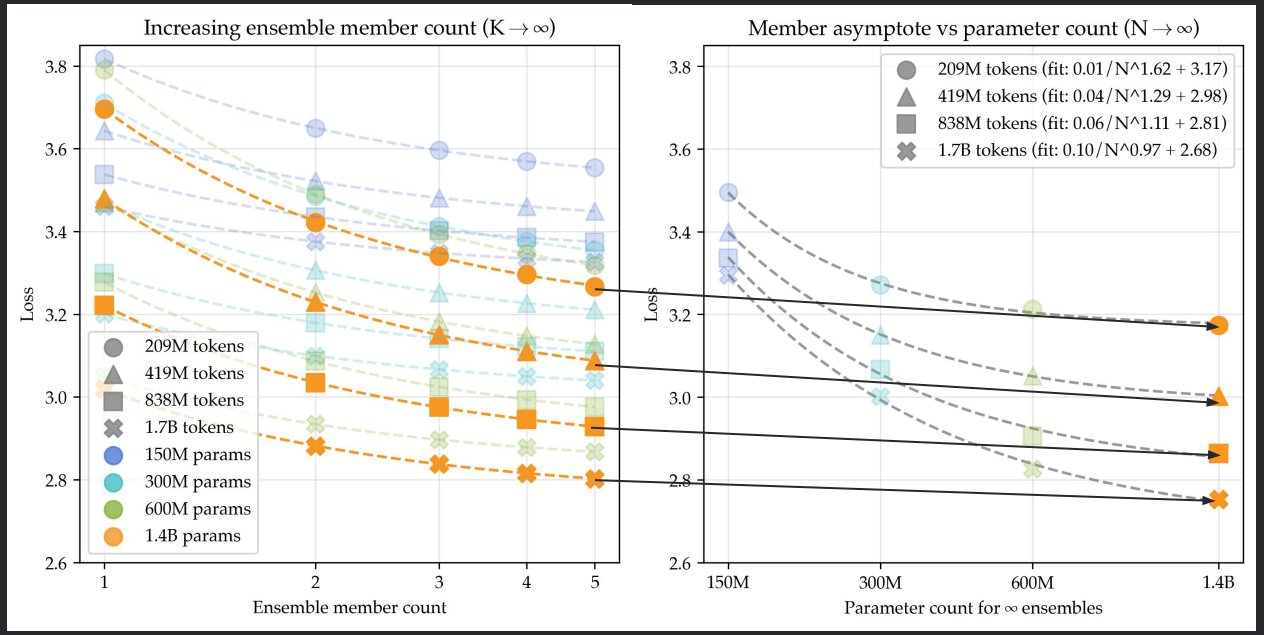
5. Scaling seed token count

For ensembles, for each D, we first send $K \to \infty$ for fixed N with hparams selected for the asymptote.

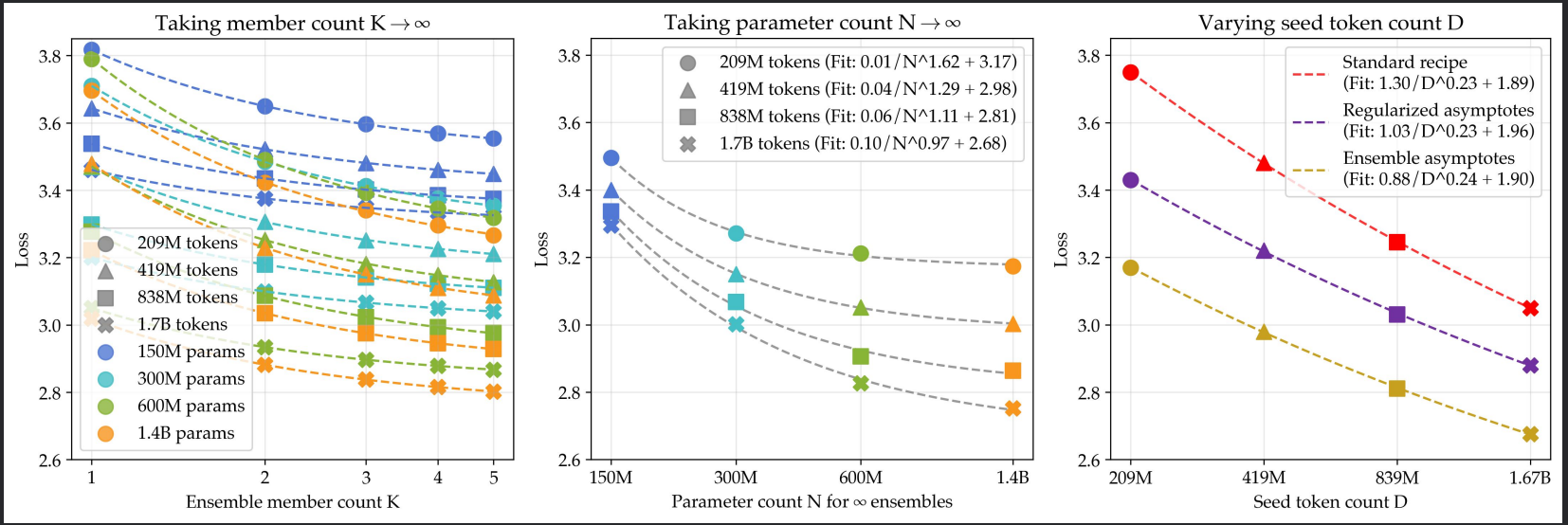


5. Scaling seed token count

For ensembles, for each D, we first send K → ∞ for fixed N with hparams selected for the asymptote.

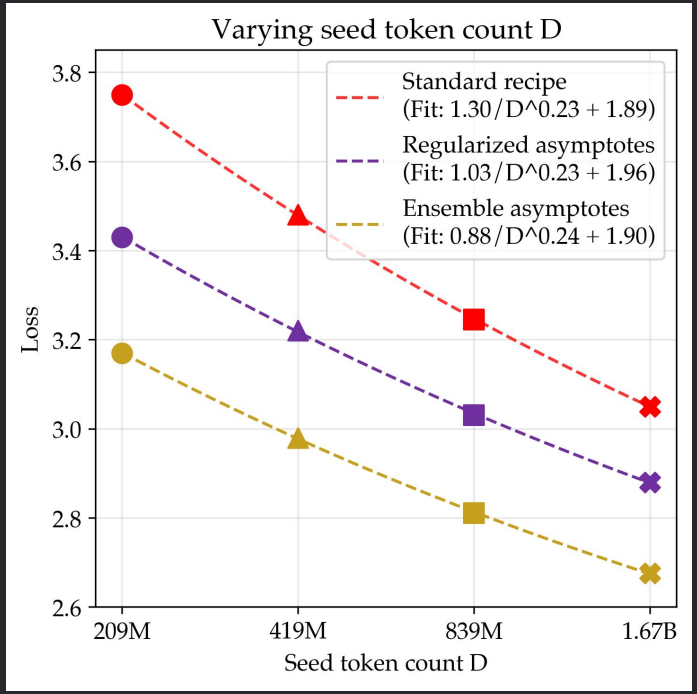


5. Scaling seed token count



5. Scaling seed token count

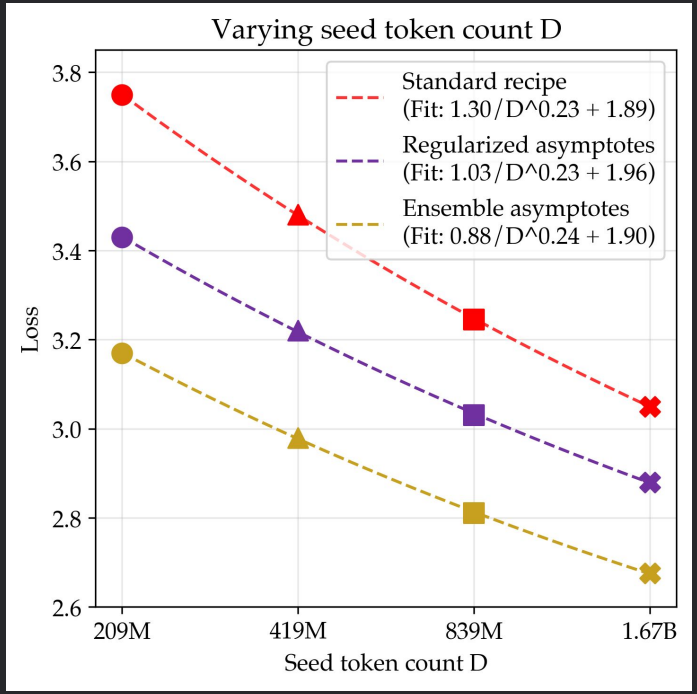
Summary of best possible loss for each method at each token count



5. Scaling seed token count

Summary of best possible loss for each method at each token count

Baseline needs ~5 times more data to match the performance of ensembling at 200M tokens

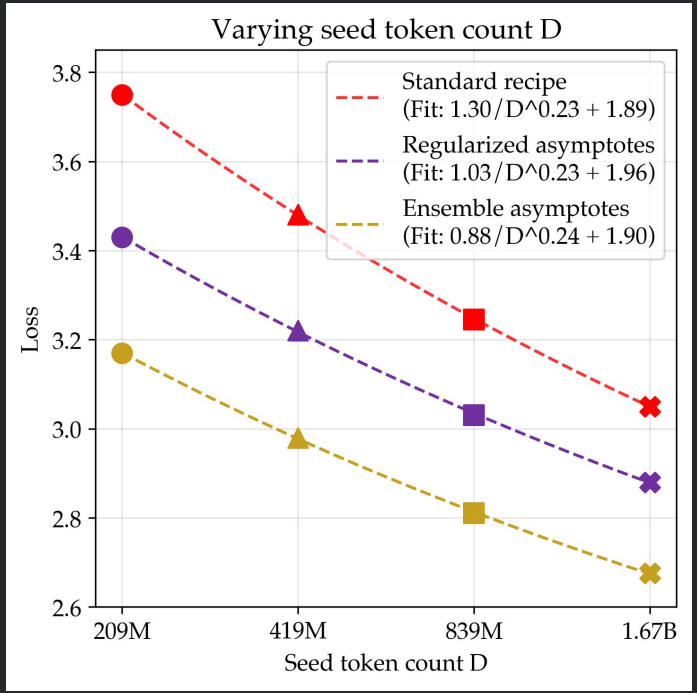


5. Scaling seed token count

Summary of best possible loss for each method at each token count

Baseline needs ~5 times more data to match the performance of ensembling at 200M tokens

Asymptotes and exponents are weirdly similar across methods, suggesting that both methods are a constant data-efficiency win across token scales



6. Distillation

do we need such large parameter counts at inference/training?

6. Distillation

Are the large parameter counts necessary for the final model and during training?

We find that via distillation, we can achieve data-efficiency gains without largely increasing parameter count

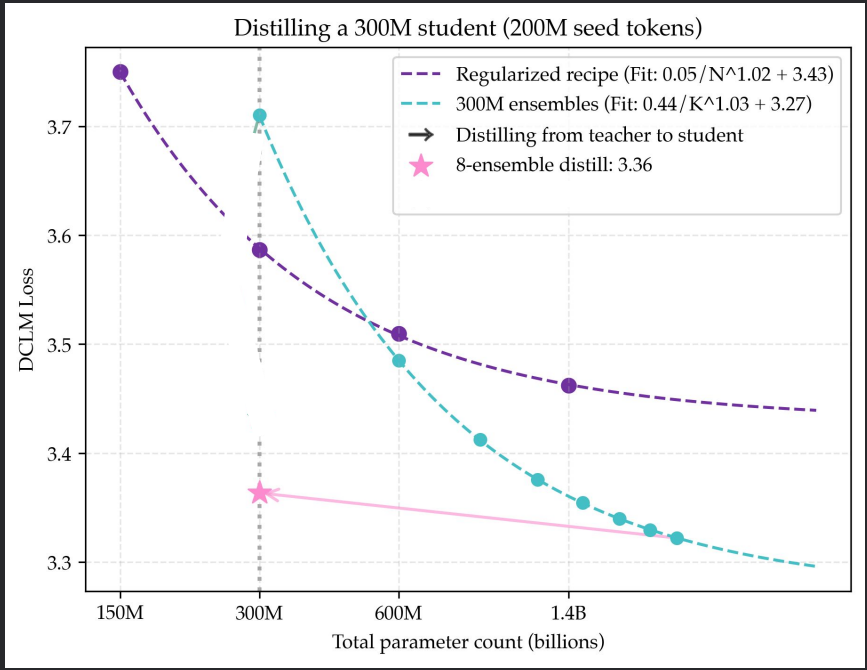
6. Distillation algorithm

We opt for the simplest algorithm of Sequence KD [Kim and Rush, 2016]

1. Train a teacher model M′ on D tokens
2. Sample from M′ unconditionally (i.e. no prompt) to generate a dataset of D′ tokens
3. Train a student model M from scratch on the mixture of D and D′

6a. Ensemble distillation

We follow this algorithm to distill an 8-ensemble into a single model, retaining most of the loss improvement and beating standard parameter scaling



6b. Self-distillation

Ensemble distillation removes ensembles at inference time. Do we need them at train time?

At first glance, this seems unlikely assuming that student models can not out-perform their teachers. In fact, recent works claim that training on self-generations results in mode collapse [Dohmatob et al, 2024 ; Gerstgrasser et al, 2024 ; Shumailov et al, 2024]

6b. Self-distillation

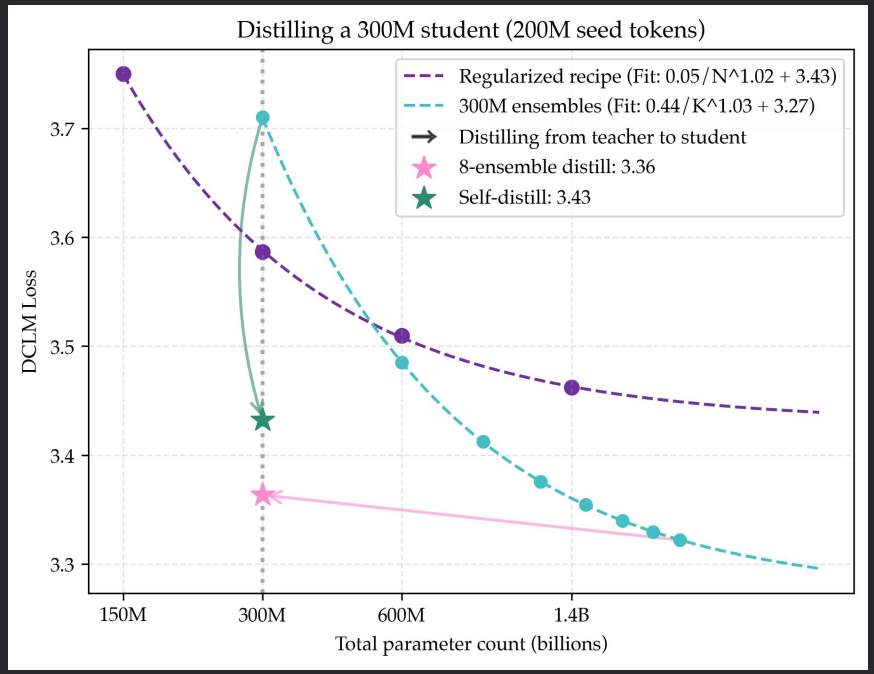
Ensemble distillation removes ensembles at inference time. Do we need them at train time?

At first glance, this seems unlikely assuming that student models can not out-perform their teachers. In fact, recent works claim that training on self-generations results in mode collapse [Dohmatob et al, 2024 ; Gerstgrasser et al, 2024 ; Shumailov et al, 2024]

Surprisingly, we find that self-distillation (making the student and teacher exactly the same) *improves* performance!

6b. Self-distillation

Even though the teacher has the same architecture and parameter count as the student, the student out-performs the teacher



6b. Self-distillation

Allen-Zhu and Li, 2023 suggest that self-distillation can be seen as implicitly ensembling the trained teacher model and the fresh student model, suggesting a connection between distillation, synthetic data, and ensembling

6b. Self-distillation

Allen-Zhu and Li, 2023 suggest that self-distillation can be seen as implicitly ensembling the trained teacher model and the fresh student model, suggesting a connection between distillation, synthetic data, and ensembling

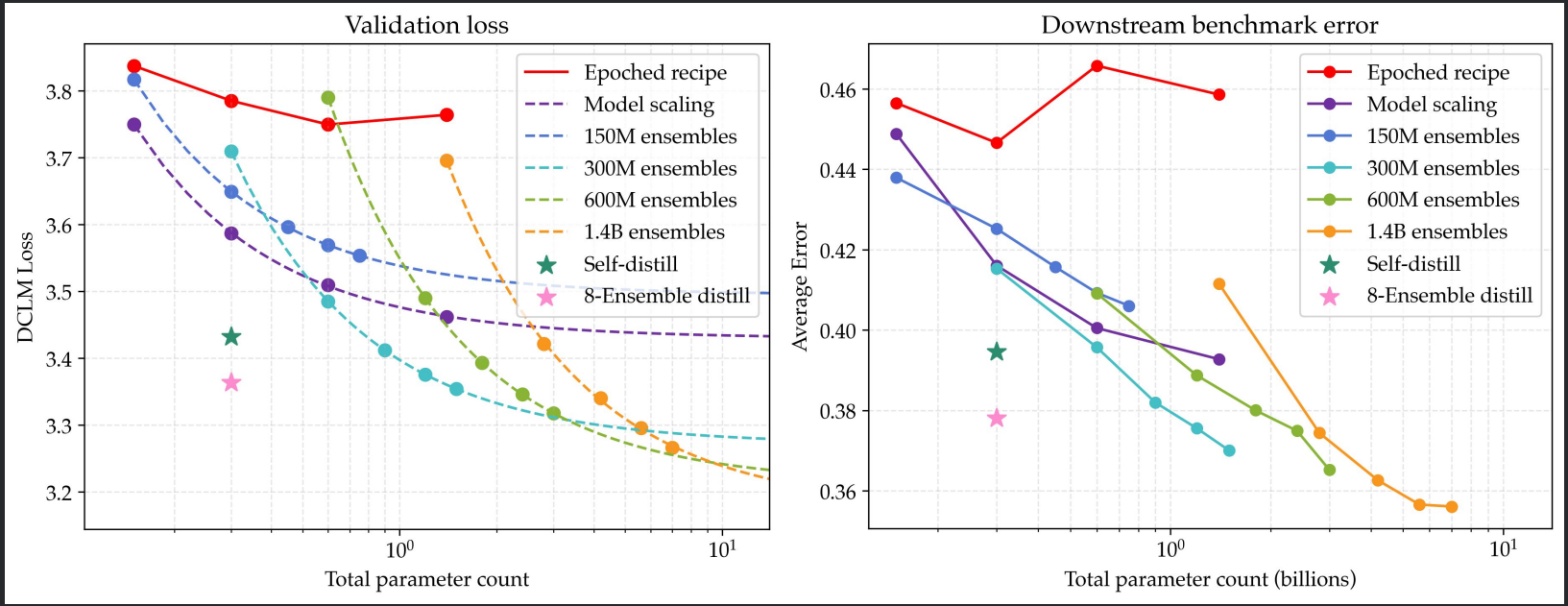
Self-distillation can be seen as a form of synthetic data that doesn't require any human prior (prompting, reward functions, etc.) We are generally excited about synthetic data that doesn't require human prior

7. Downstream benefits

is it okay that we only looked at validation loss?

7a: Validation loss translates to downstream benchmarks

Better validation loss translates to better average performance on the small model benchmarks of ARC Easy, PiQA, and SciQ



7b. Continual pre-training

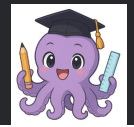
Do these findings only apply to pre-training, or do they also help when adapting pre-trained models to rare data?

We take the MegaMath-Web-Pro dataset from OctoThinker as an example of strong reasoning data for math [Wang et al, 2025]

7b. Continual pre-training

After applying our data-efficiency tricks, we train with 4B tokens in a way that out-performs their training with 73B tokens! This is a 17.5x data-efficiency improvement

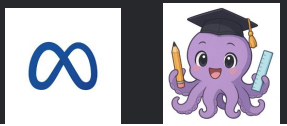


Benchmarks	Llama 3B	CPT (4B tokens)			K-ensembles			CPT (73B tokens)
		Default	Lower BS	Epoching ($K = 1$)	$K = 2$	$K = 4$	$K = 8$	
GSM8K$_{\text{(8-shot)}}$	28.23	38.44	44.50	44.05	49.28	51.80	**52.99**	49.51
MATH$_{\text{(4-shot)}}$	6.90	14.38	17.64	19.74	21.84	23.04	**23.50**	23.40
MATHQA$_{\text{(8-shot)}}$	35.07	38.96	41.31	42.58	45.12	**46.06**	45.26	44.79
Average	24.25	30.59	34.48	35.82	38.79	40.35	**40.58**	39.23

7b. Continual pre-training

After applying our data-efficiency tricks, we train with 4B tokens in a way that out-performs their training with 73B tokens! This is a 17.5x data-efficiency improvement (AND more compute-efficient?!)




Benchmarks	Llama 3B	CPT (4B tokens)			K-ensembles			CPT (73B tokens)
		Default	Lower BS	Epoching ($K = 1$)	$K = 2$	$K = 4$	$K = 8$	
GSM8K$_{\text{(8-shot)}}$	28.23	38.44	44.50	44.05	49.28	51.80	**52.99**	49.51
MATH$_{\text{(4-shot)}}$	6.90	14.38	17.64	19.74	21.84	23.04	**23.50**	23.40
MATHQA$_{\text{(8-shot)}}$	35.07	38.96	41.31	42.58	45.12	**46.06**	45.26	44.79
Average	24.25	30.59	34.48	35.82	38.79	40.35	**40.58**	39.23

8. Conclusion

8. Conclusion

We present some techniques (larger models, regularization, ensembling, distillation) that result in much better loss for finite data and infinite compute

8. Conclusion

We present some techniques (larger models, regularization, ensembling, distillation) that result in much better loss for finite data and infinite compute

None of the methods we propose are new: these are classic tricks when we had limited images (MNIST, CIFAR) or sentences (Penn Tree Bank, BabyLM). Therefore, we believe there's tons of free lunch from rethinking basic training decisions such as architecture [Gladstone et al, 2025], objective [Prabhudesai et al, 2025], data augmentation [Maini et al, 2024], optimizer, etc.

8. Conclusion

However, in the modern pre-training + generative modeling era, we believe there may be many new algorithms that are more data-efficient at the cost of more compute. **Asymptotes** will be critical to estimating which algorithms work best.

8. Conclusion

However, in the modern pre-training + generative modeling era, we believe there may be many new algorithms that are more data-efficient at the cost of more compute. **Asymptotes** will be critical to estimating which algorithms work best.

We are also excited to apply this general purpose toolkit to other domains such as personalization, rare languages, agents, DNA, robotics, etc.

8. Conclusion

However, in the modern pre-training + generative modeling era, we believe there may be many new algorithms that are more data-efficient at the cost of more compute. **Asymptotes** will be critical to estimating which algorithms work best.

We are also excited to apply this general purpose toolkit to other domains such as personalization, rare languages, agents, DNA, robotics, etc.

Bonus: This is a particularly great area for academics to do research as scaling laws can extrapolate the performance at large scale with tiny experiments (most of our iteration was done with 40M parameter models with 50M tokens)