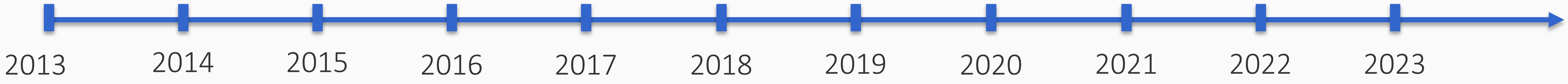
The impact of scale

A lot of this material is based on the ACL 2022 tutorial on Zero- and Few-Shot NLP with Pretrained Language Models by Iz Beltagy, Arman Cohan, Robert L. Logan IV, Sewon Min, Sameer Singh

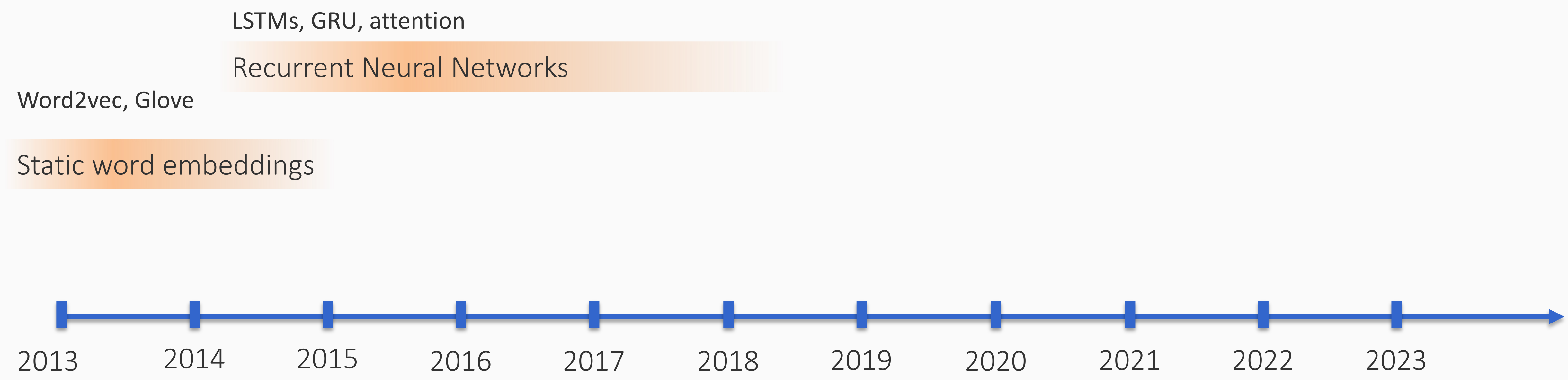
Where are we?

Word2vec, Glove

Static word embeddings

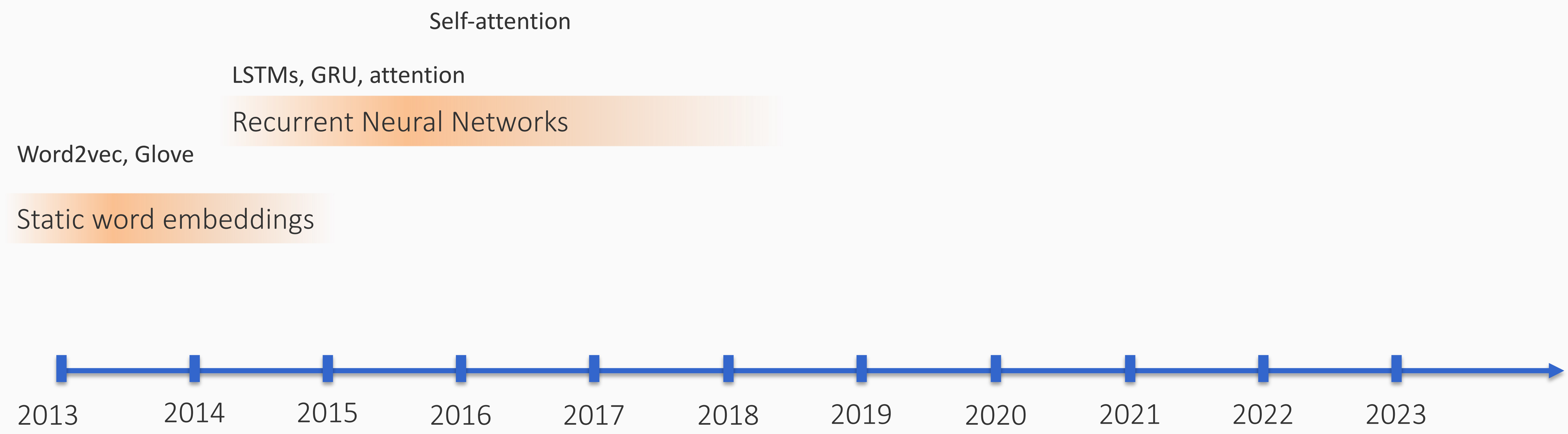


Where are we?

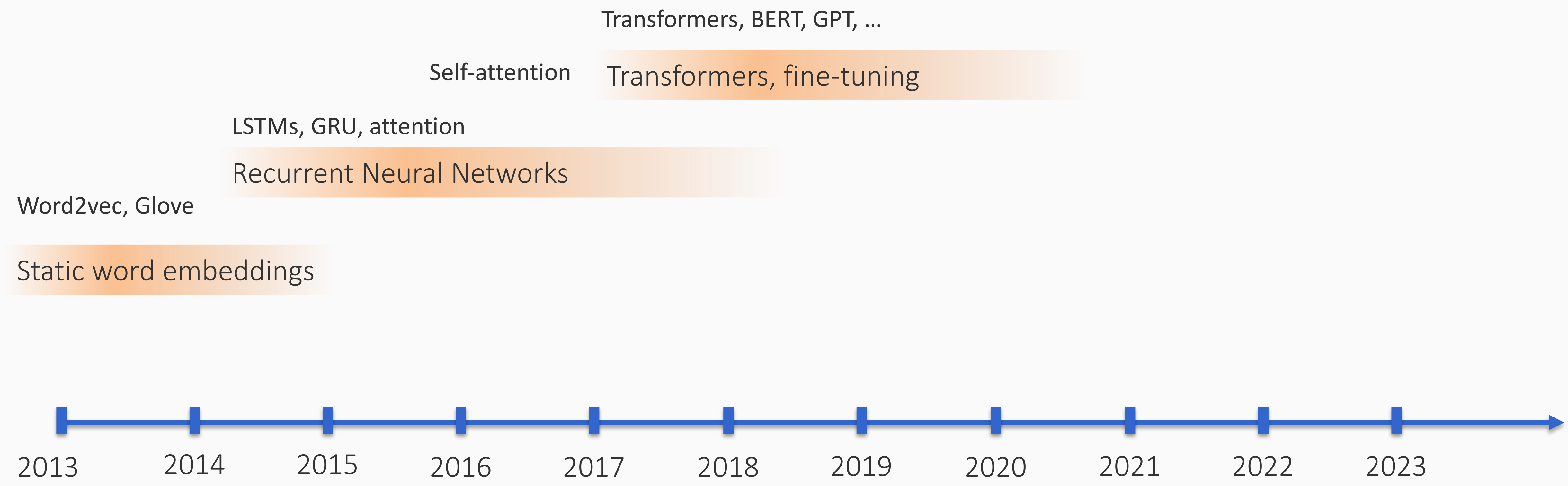


LSTMs, GRU, attention
Recurrent Neural Networks
Word2vec, Glove
Static word embeddings
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023

Where are we?

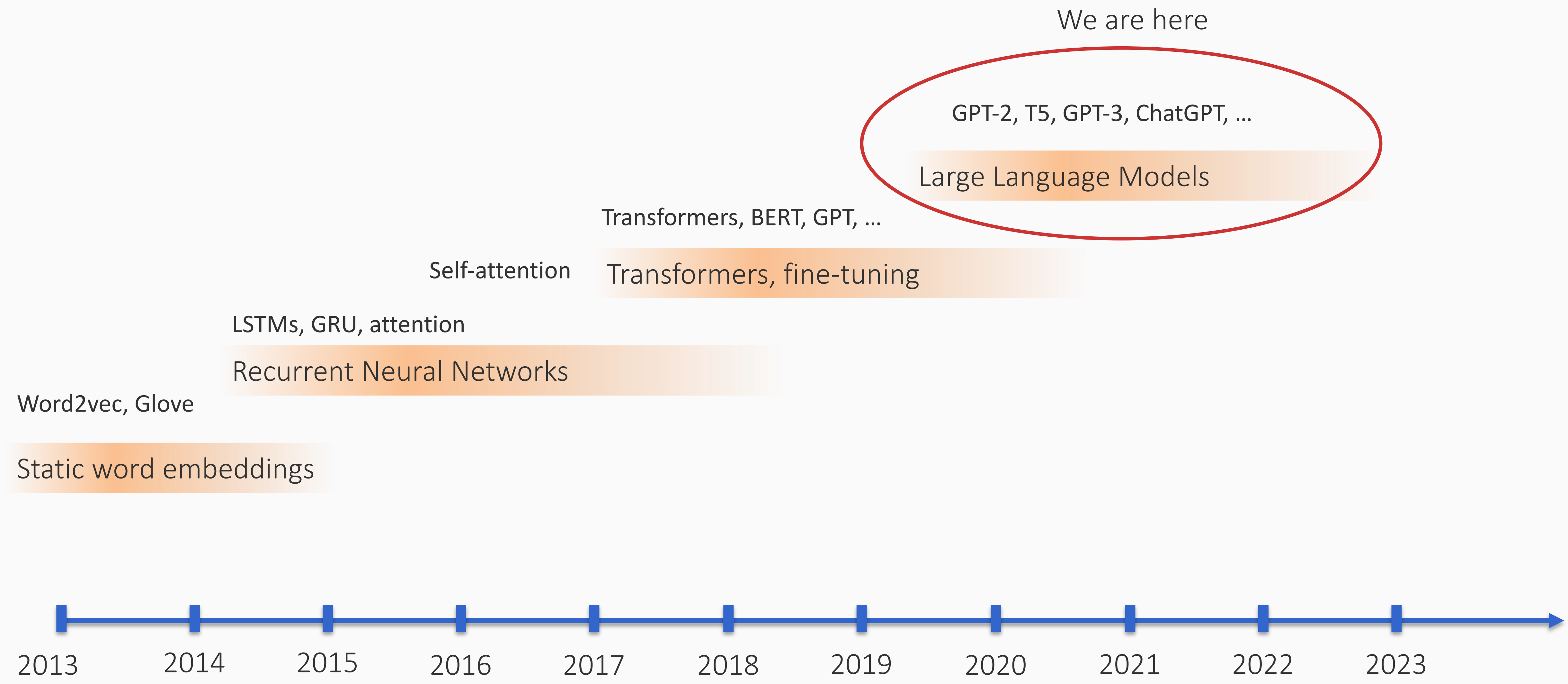

Self-attention
LSTMs, GRU, attention
Recurrent Neural Networks
Word2vec, Glove
Static word embeddings
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023

Where are we?

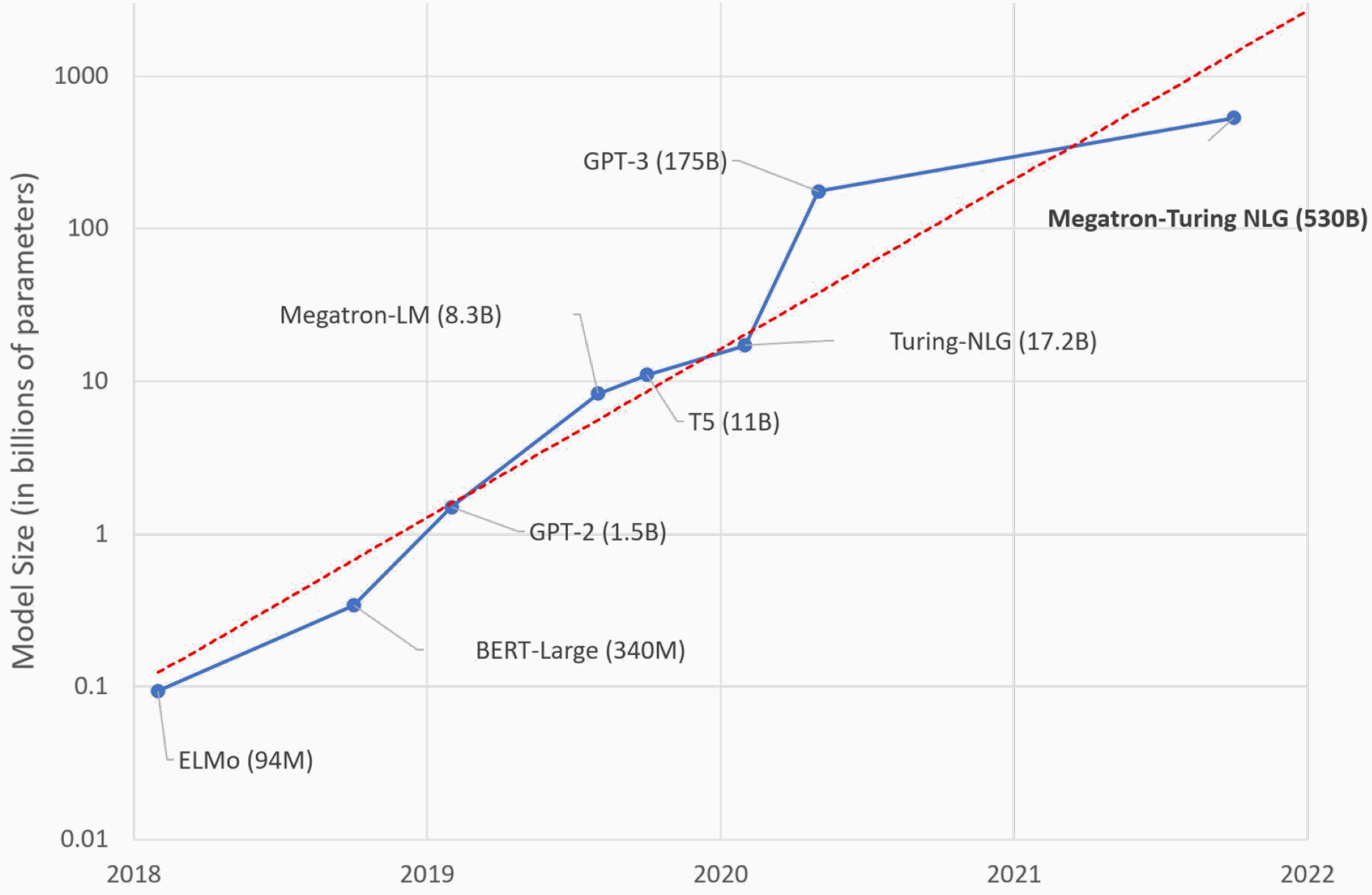


Transformers, BERT, GPT, …
Self-attention
Transformers, fine-tuning
LSTMs, GRU, attention
Recurrent Neural Networks
Word2vec, Glove
Static word embeddings
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023

Where are we?

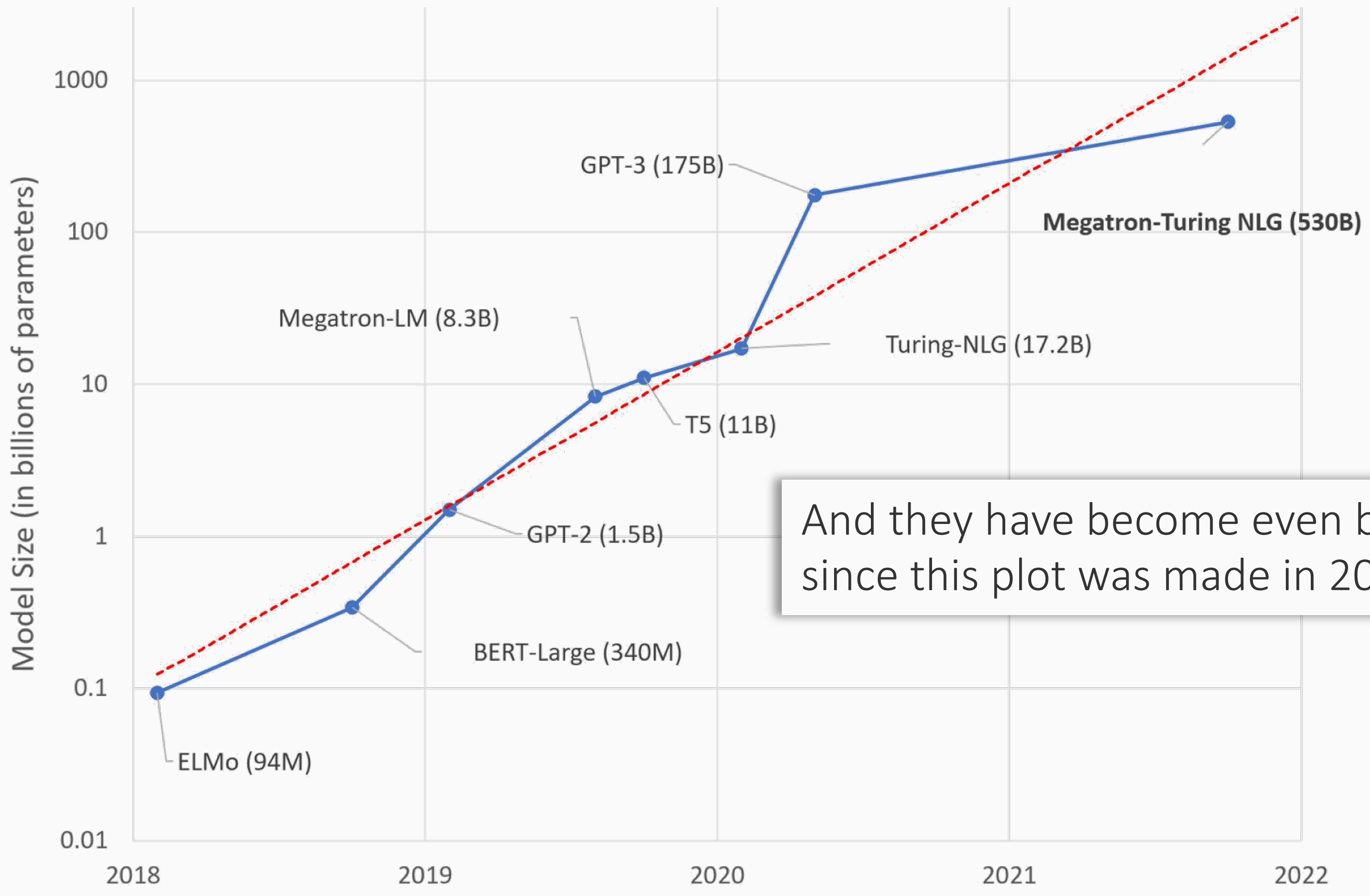

We are here
GPT-2, T5, GPT-3, ChatGPT, …
Large Language Models
Transformers, BERT, GPT, …
Self-attention
Transformers, fine-tuning
LSTMs, GRU, attention
Recurrent Neural Networks
Word2vec, Glove
Static word embeddings
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023

Models for language have become bigger



Plot from Microsoft Research blog (Alvi & Kharya 2021)

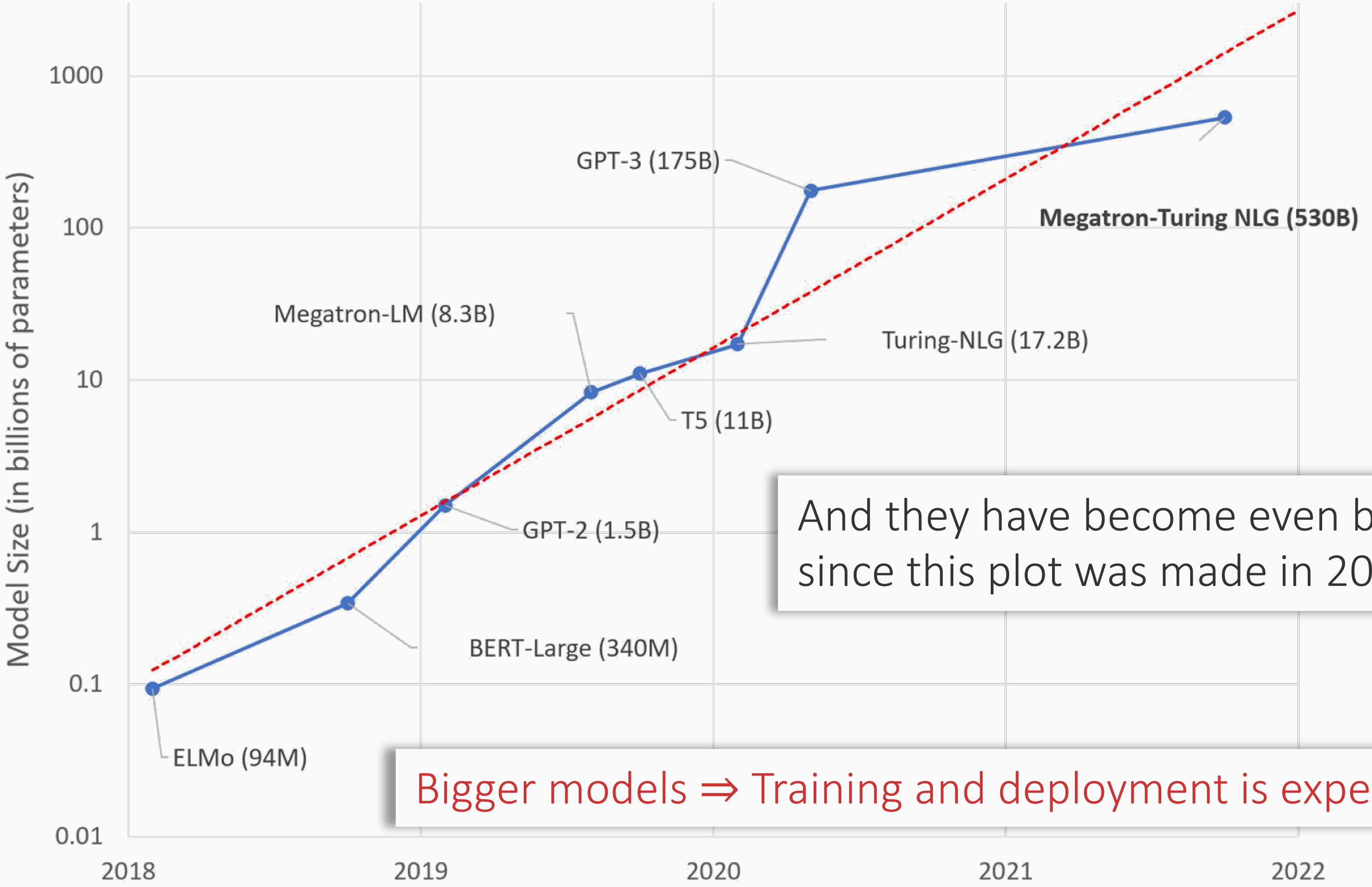
Models for language have become bigger



And they have become even bigger since this plot was made in 2021

Plot from Microsoft Research blog (Alvi & Kharya 2021)

Models for language have become bigger



And they have become even bigger since this plot was made in 2021

Bigger models ⇒ Training and deployment is expensive

Plot from Microsoft Research blog (Alvi & Kharya 2021)

Bigger models

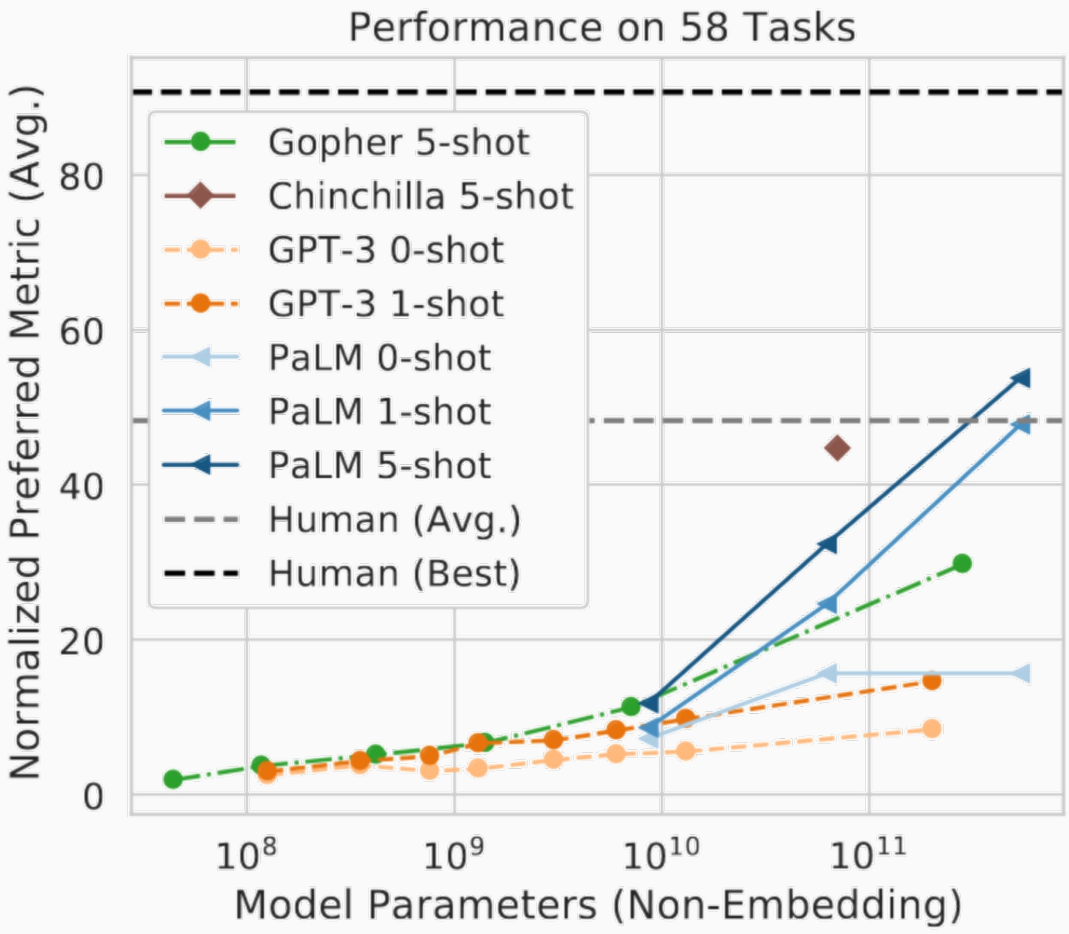
⇒ Training and deployment is expensive

Bigger models

⇒ Training and deployment is expensive

More parameters → better performance?

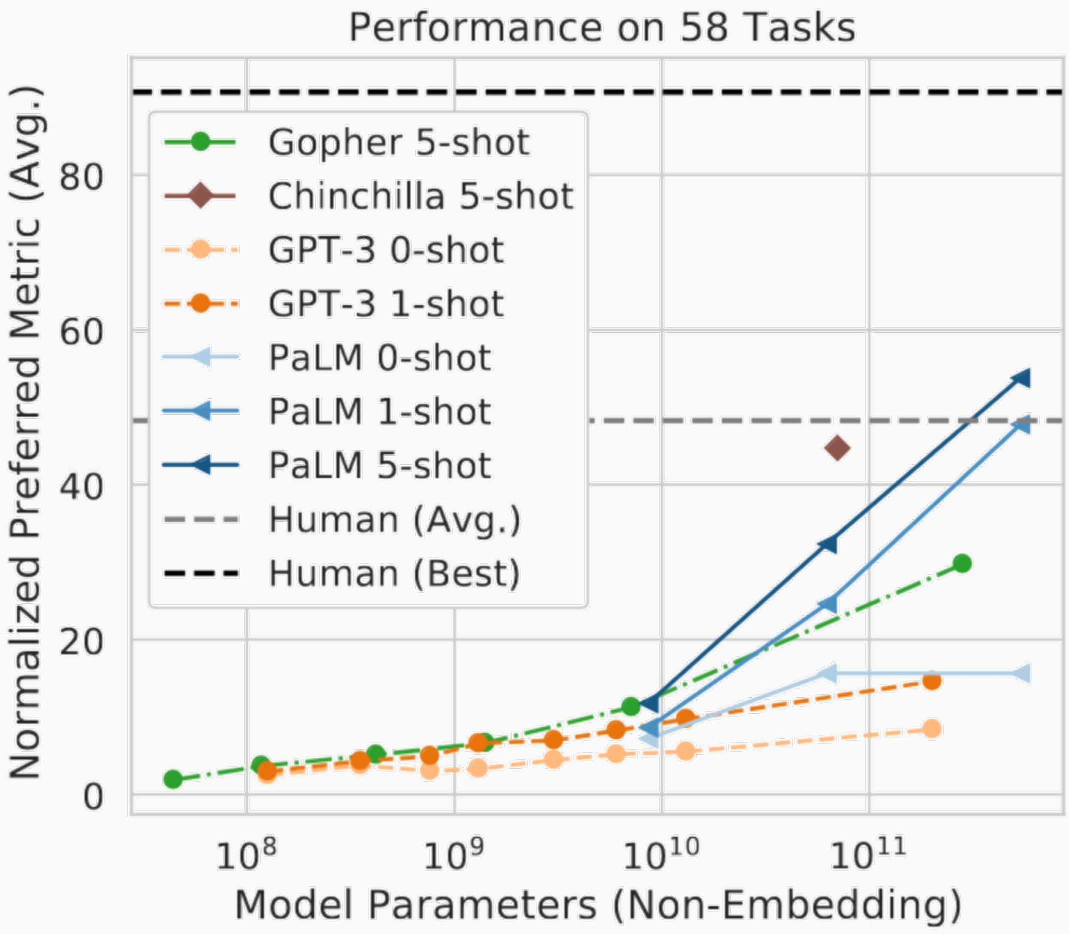
Especially in the zero- and few-shot setting

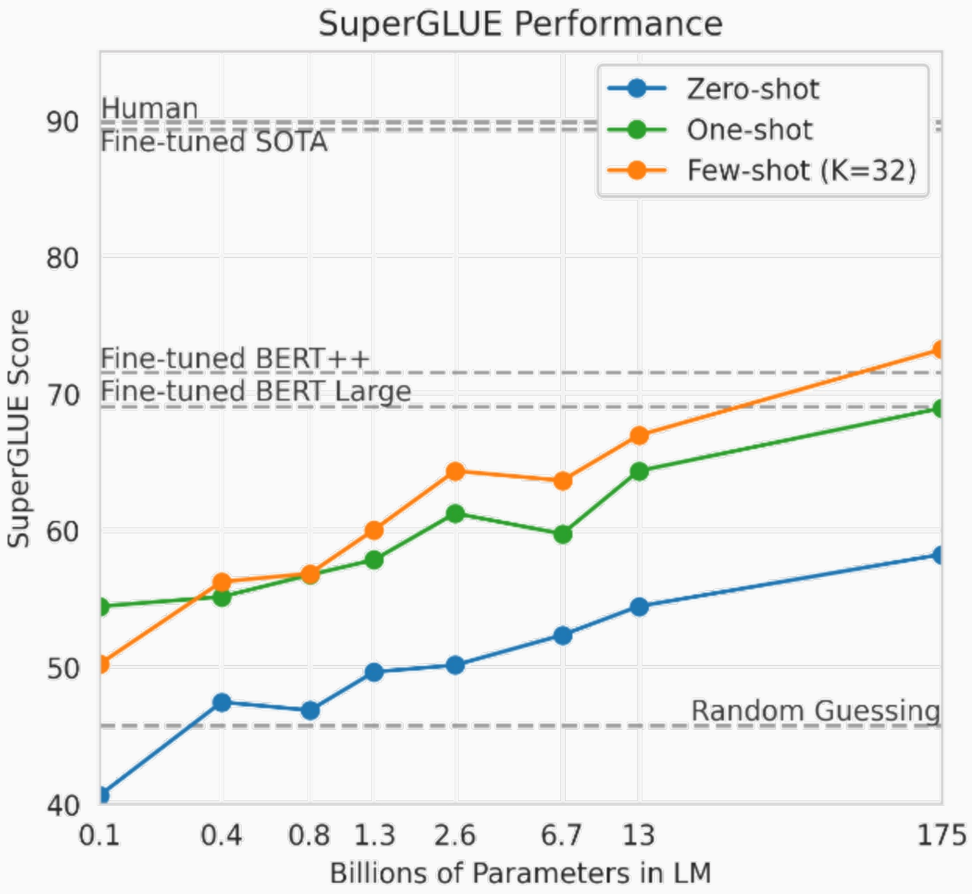


Chowdhery et al. "Palm: Scaling language modeling with pathways." *arXiv preprint arXiv:2204.02311* (2022).

More parameters → better performance?

Especially in the zero- and few-shot setting





Chowdhery et al. "Palm: Scaling language modeling with pathways." *arXiv preprint arXiv:2204.02311* (2022).

Brown et al. "Language models are few-shot learners." *Advances in neural information processing systems* 33 (2020).

What does scaling mean?

Scaling is not *just* about models with more parameters

What does scaling mean?

Scaling is not *just* about models with more parameters

Scaling is about using more compute

What does scaling mean?

Scaling is not *just* about models with more parameters

Scaling is about using more compute

- More compute for model forward and backward passes
- More compute for training iterations also

What does scaling mean?

Scaling is not *just* about models with more parameters

Scaling is about using more compute

- More compute for model forward and backward passes
- More compute for training iterations also

- Of course, only a large enough model can take advantage of the additional training
  - Think about model capacity

What does scaling mean?

Scaling is not *just* about models with more parameters

Scaling is about using more compute

- More compute for model forward and backward passes
- More compute for training iterations also

- Of course, only a large enough model can take advantage of the additional training
  - Think about model capacity
  - So scaling tends to be associated with larger models

Scale: Model size × # training tokens

Model name	Model size (billions of parameters)	Training tokens (billions of tokens)	Compute (in GPT-3 terms)
GPT-NeoX	20	472	0.18x
GPT3	175	300	1x
Gopher	280	300	1.6x
Chinchilla	67	1,400	1.6x
Megatron-Turing-NLG	530	270	2.7x
PaLM	540	780	8x

Larger models present new problems

We cannot find the best hyperparameters by training multiple models

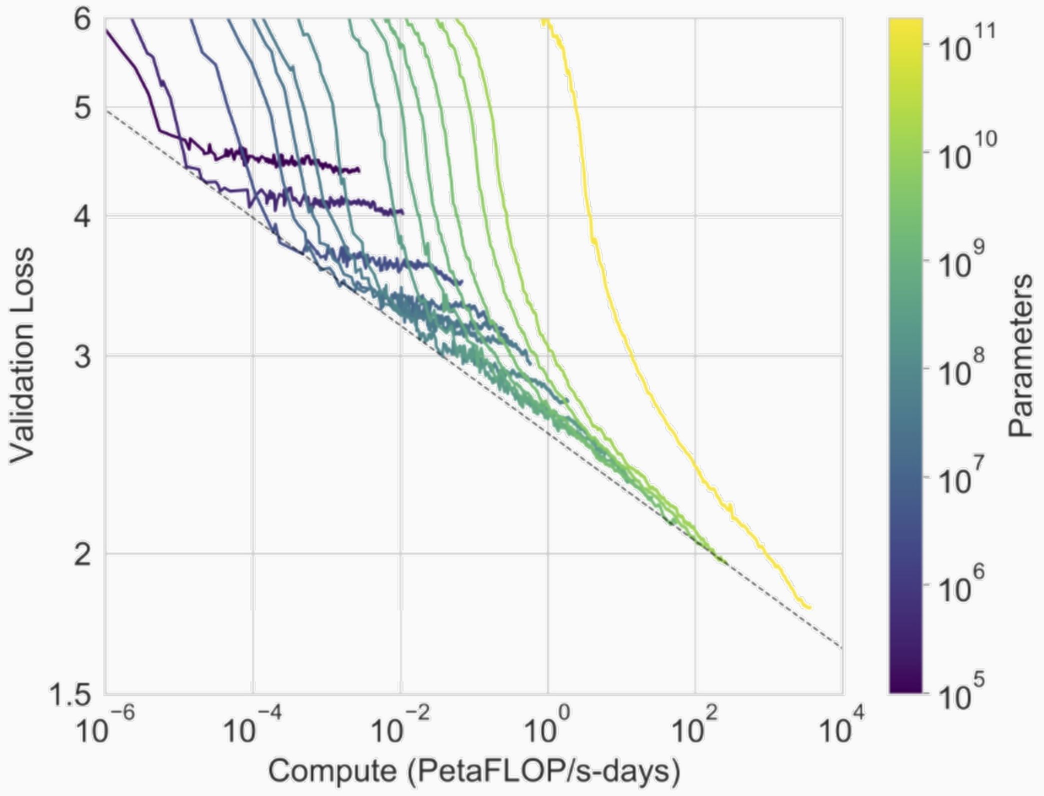
We don’t know when to stop training

Given a budget for compute, should we increase the model size or the number of training steps using that budget?

Can we develop a theory that connects loss with the model sizes and the number of training steps?

How big should your model be?

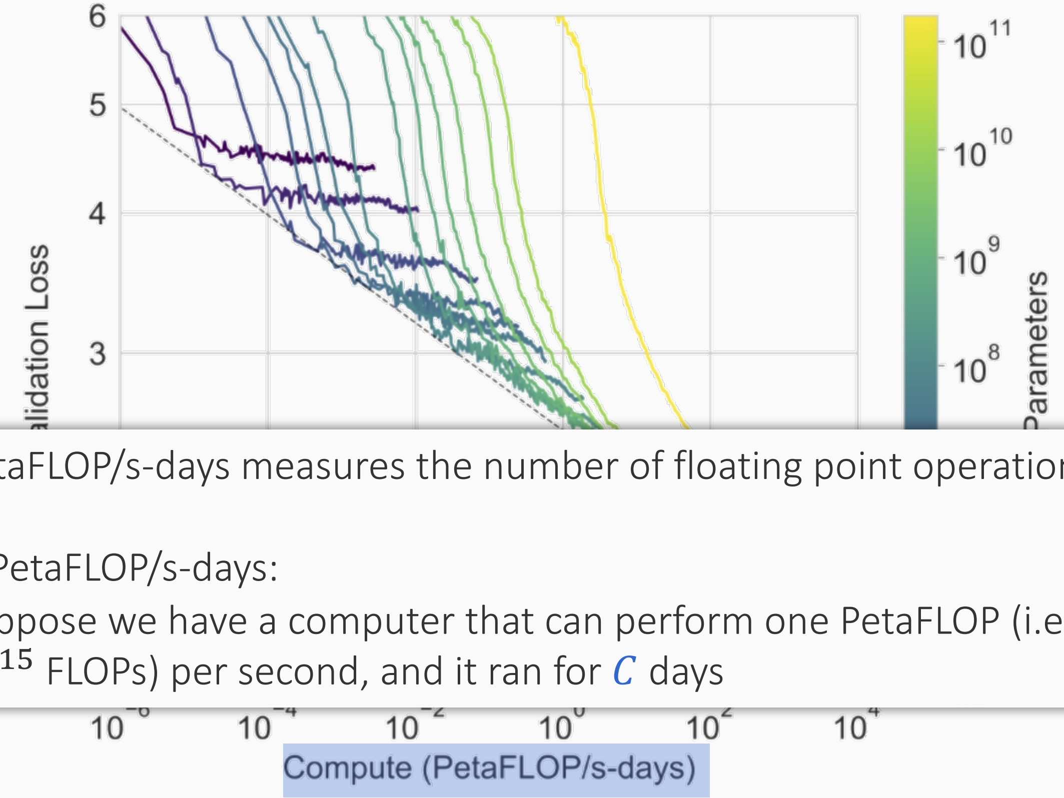
Suppose you have a compute budget, what model size should you use?



Brown, Tom B., et al. "Language Models are Few-Shot Learners." *arXiv preprint arXiv:2005.14165* (2020).z

How big should your model be?

Suppose you have a compute budget, what model size should you use?



PetaFLOP/s-days measures the number of floating point operations

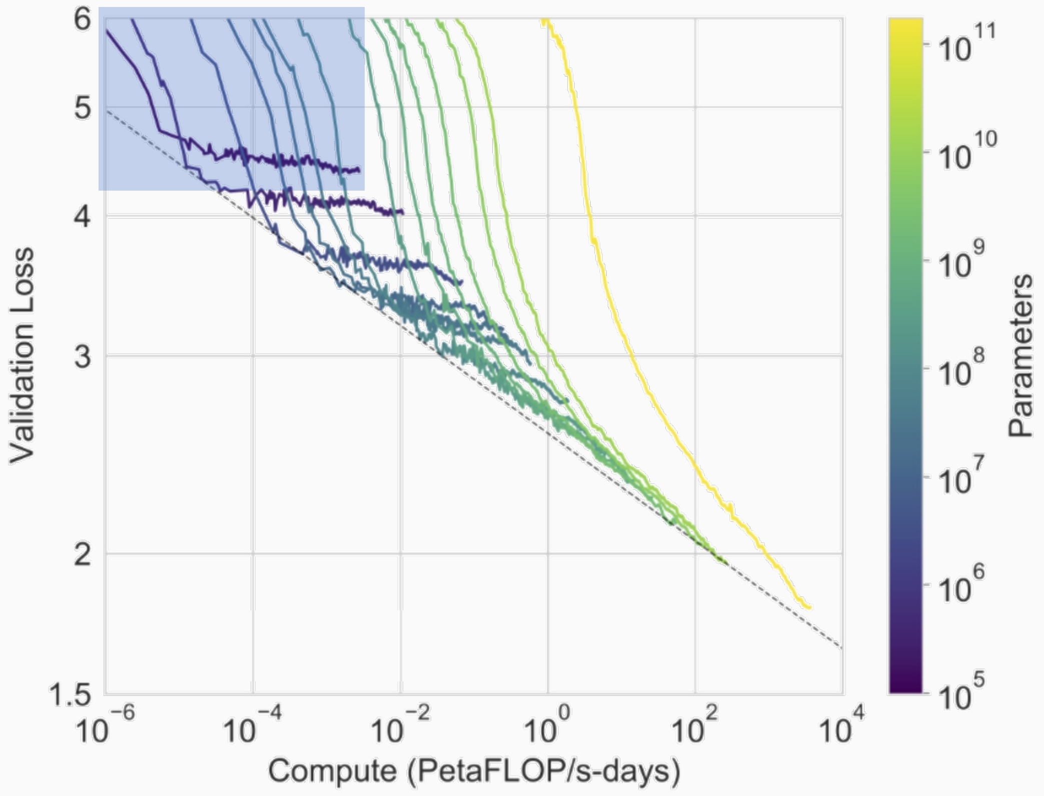
C PetaFLOP/s-days:
Suppose we have a computer that can perform one PetaFLOP (i.e., 10^{15} FLOPs) per second, and it ran for C days

Brown, Tom B., et al. "Language Models are Few-Shot Learners." *arXiv preprint arXiv:2005.14165* (2020).z

How big should your model be?

Suppose you have a compute budget, what model size should you use?

- Smaller models don’t have enough capacity to use the extra compute. They plateau early



Brown, Tom B., et al. "Language Models are Few-Shot Learners." *arXiv preprint arXiv:2005.14165* (2020).z

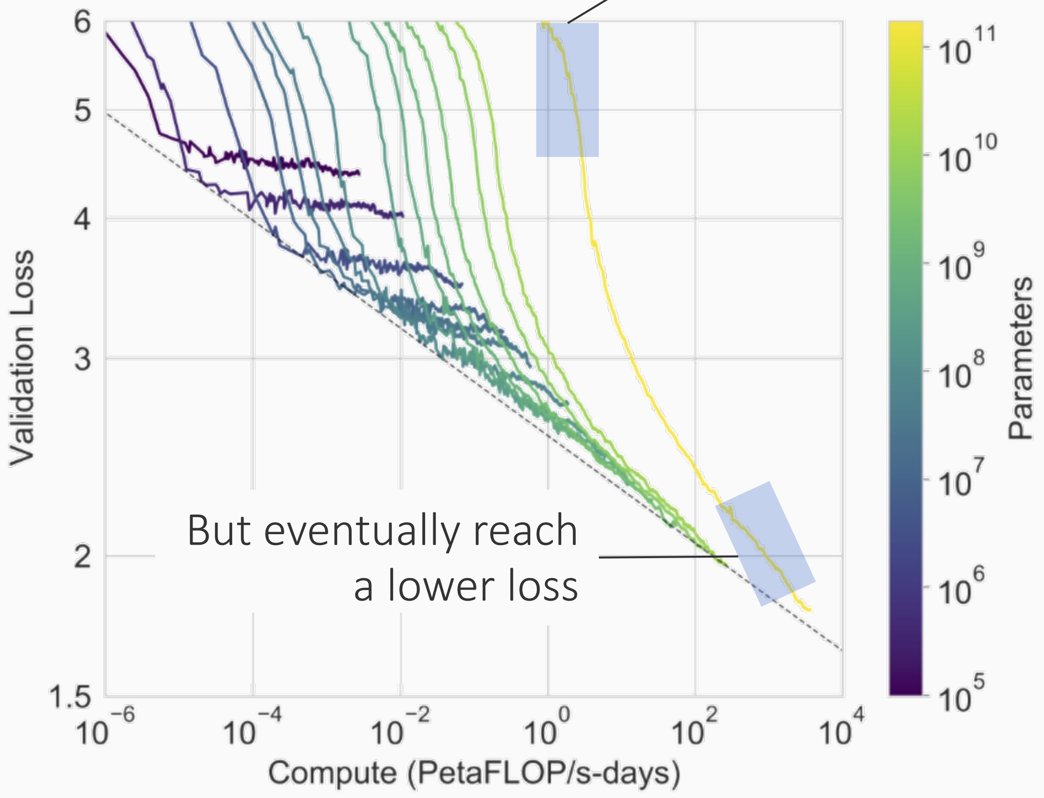
How big should your model be?

Initially take longer to get to the same loss as a smaller model

Suppose you have a compute budget, what model size should you use?

- Smaller models don't have enough capacity to use the extra compute. They plateau early

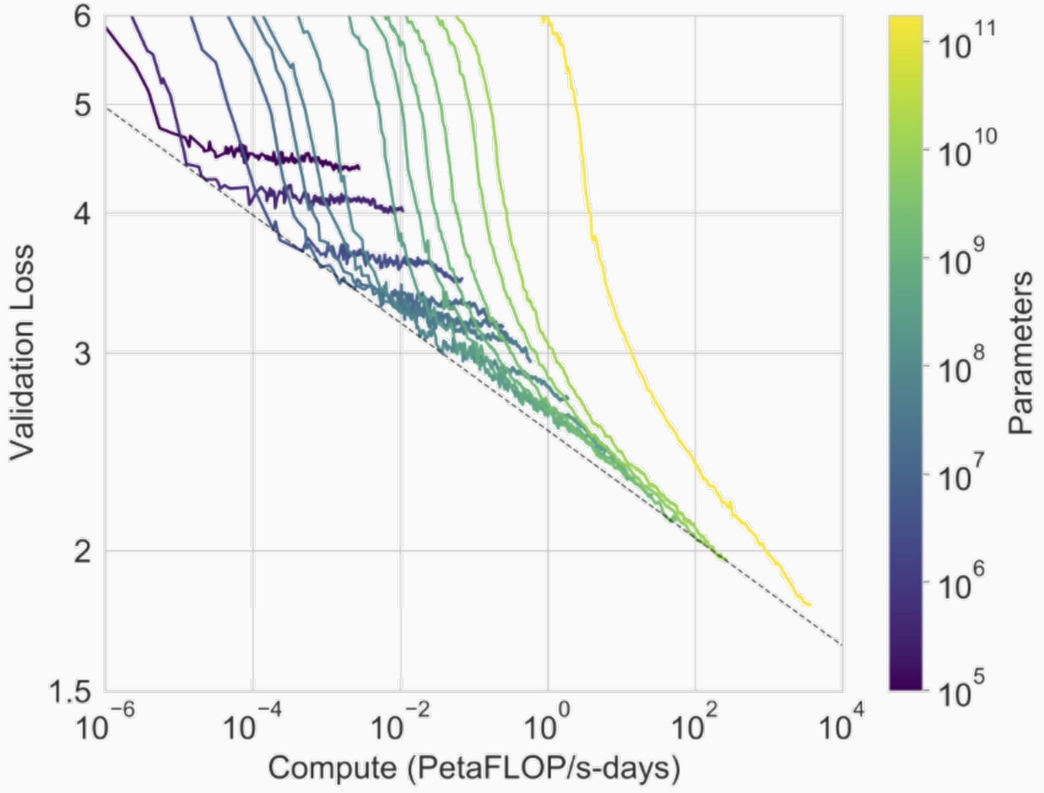
- Larger models take longer initially, but with more compute get to lower losses



Brown, Tom B., et al. "Language Models are Few-Shot Learners." *arXiv preprint arXiv:2005.14165* (2020).z

How big should your model be?

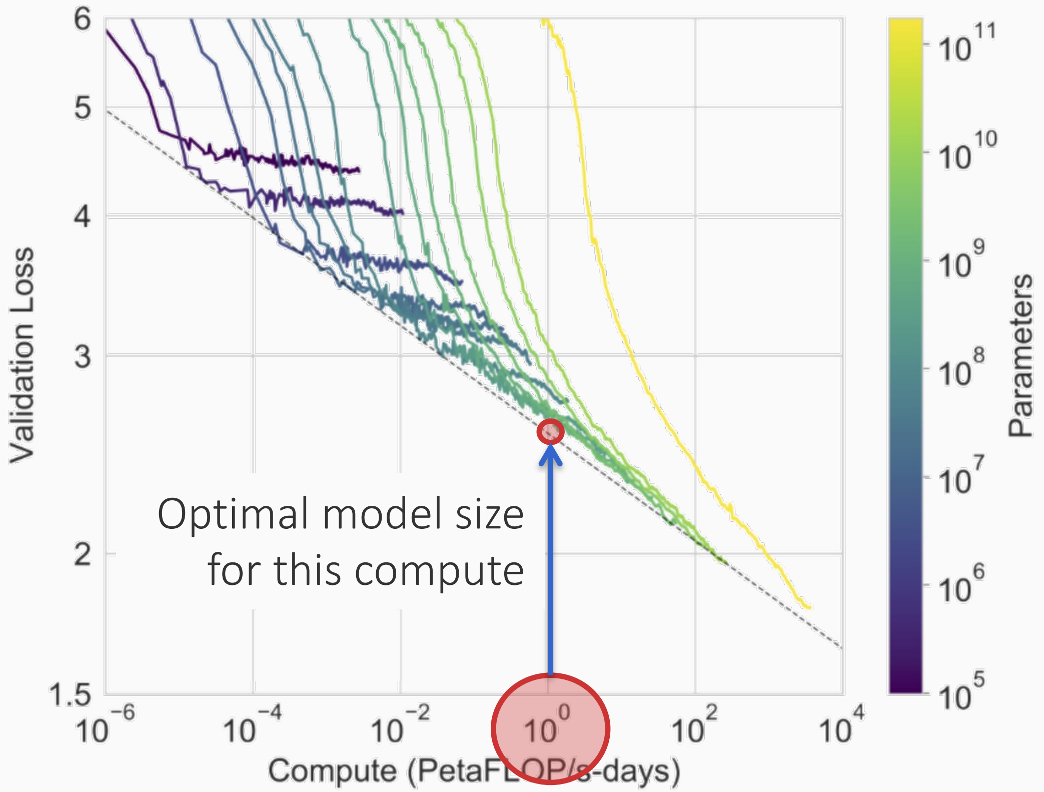
For a given compute, we can ask: What is the optimal model size?



Brown, Tom B., et al. "Language Models are Few-Shot Learners." *arXiv preprint arXiv:2005.14165* (2020).z

How big should your model be?

For a given compute, we can ask: What is the optimal model size?



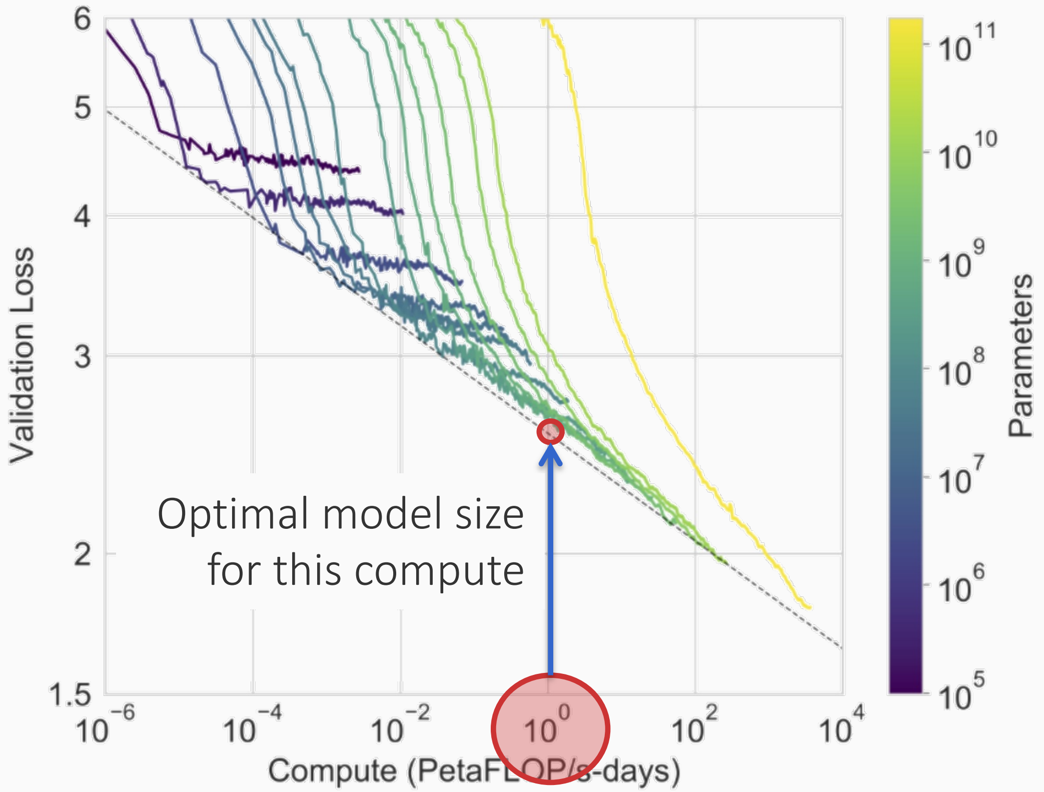
Brown, Tom B., et al. "Language Models are Few-Shot Learners." *arXiv preprint arXiv:2005.14165* (2020).z

How big should your model be?

For a given compute, we can ask: What is the optimal model size?

Rather than training models to convergence, train them to optimality (which occurs earlier)

- Extra effort is not worth it because you can get a better model for the effort by picking a larger model



Brown, Tom B., et al. "Language Models are Few-Shot Learners." *arXiv preprint arXiv:2005.14165* (2020).z

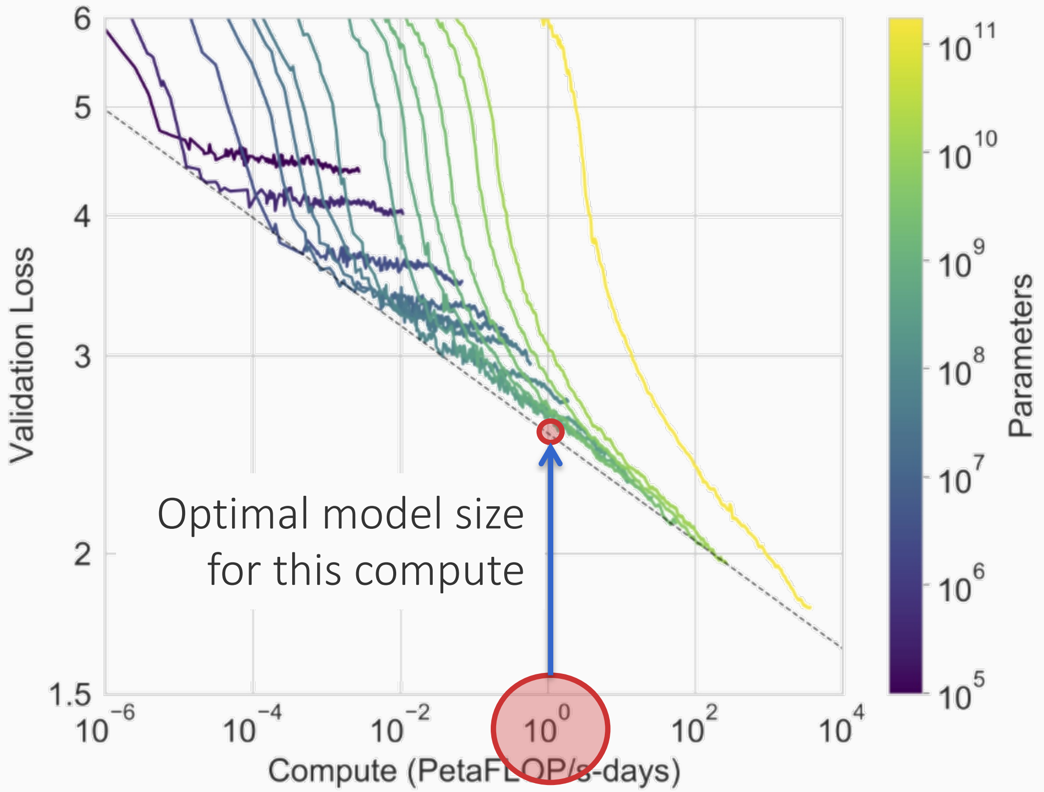
How big should your model be?

For a given compute, we can ask: What is the optimal model size?

Rather than training models to convergence, train them to optimality (which occurs earlier)

Extra effort is not worth it because you can get a better model for the effort by picking a larger model

But to make this choice, we need to know all these learning curves. How can we get them without training a model? *Or when the budget only allows training one LARGE model?*

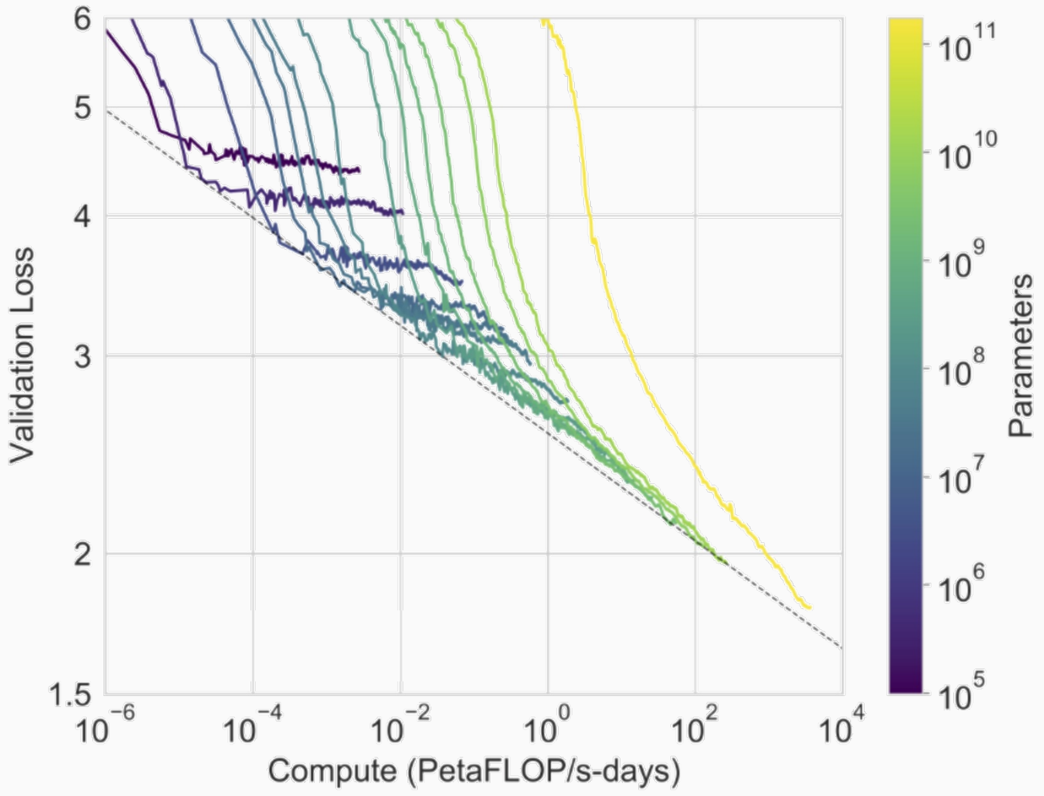


Brown, Tom B., et al. "Language Models are Few-Shot Learners." *arXiv preprint arXiv:2005.14165* (2020).z

Scaling laws

[Kaplan et al 2020]

The claim: Test loss are power law functions of model size and compute



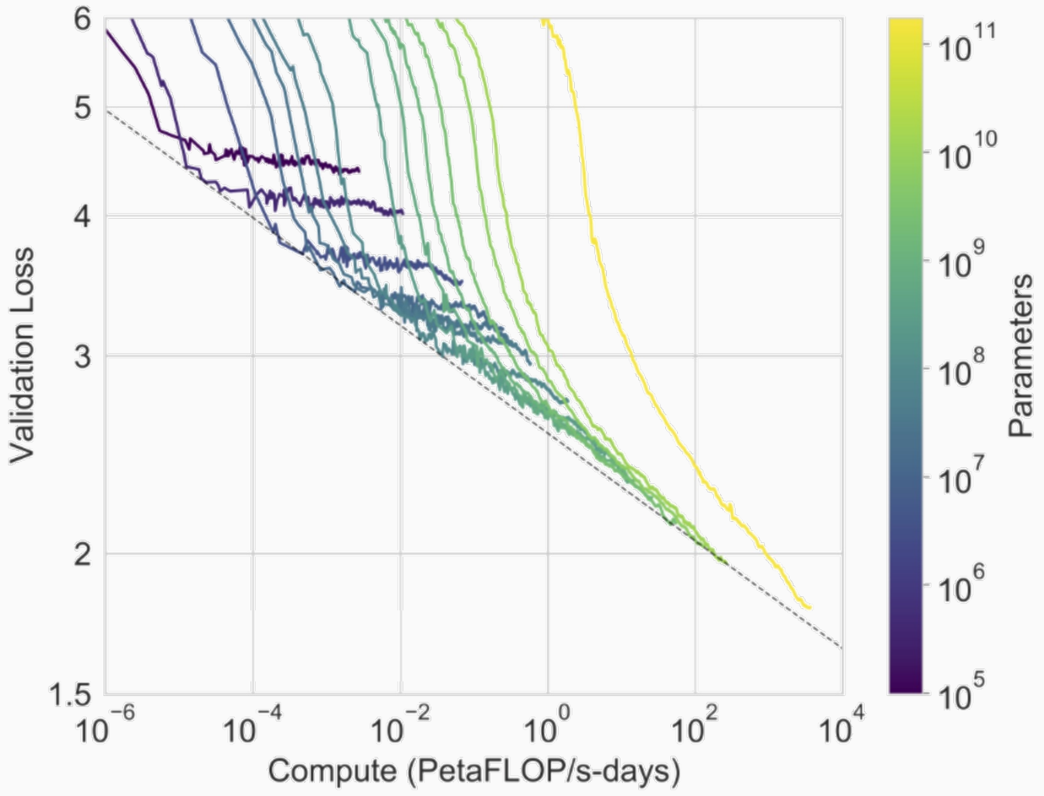
Kaplan et al. "Scaling laws for neural language models." *arXiv preprint arXiv:2001.08361* (2020).

Scaling laws

[Kaplan et al 2020]

The claim: Test loss are power law functions of model size and compute

If this were true, then use small models to fit the constants of the power law function, and then extrapolate to large sizes



Kaplan et al. "Scaling laws for neural language models." *arXiv preprint arXiv:2001.08361* (2020).

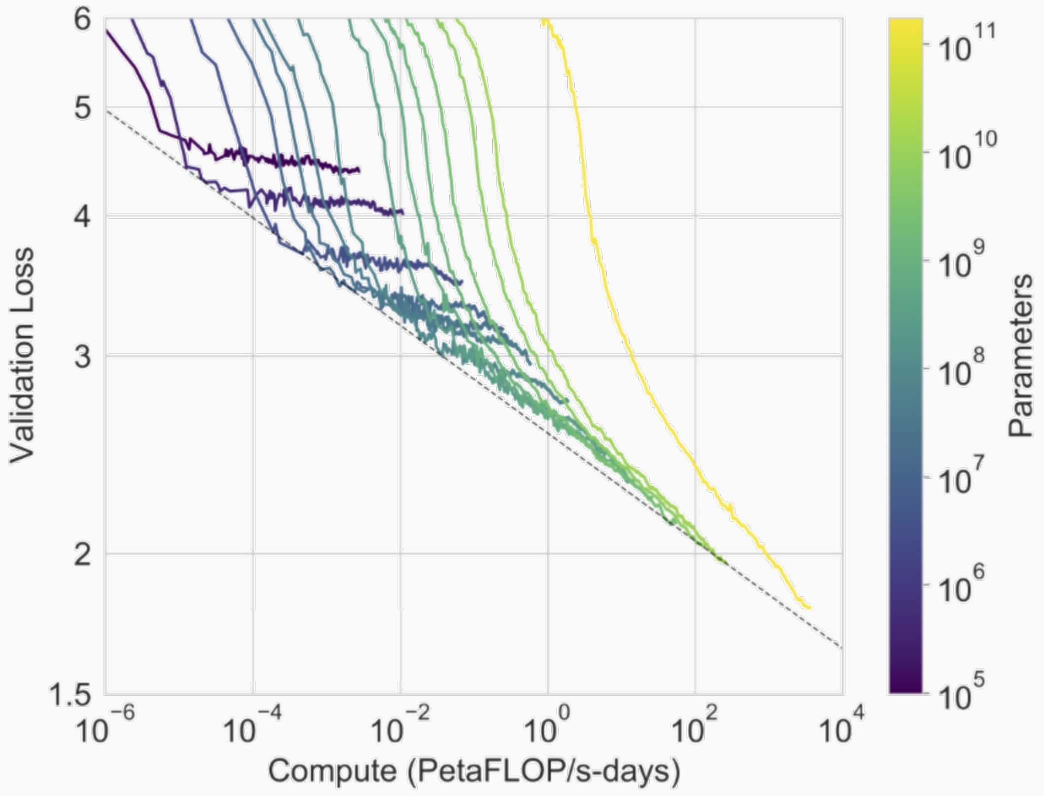
Scaling laws

[Kaplan et al 2020]

The claim: Test loss are power law functions of model size and compute

If this were true, then use small models to fit the constants of the power law function, and then extrapolate to large sizes

Kaplan et al showed empirical support for the existence of such power laws



Kaplan et al. "Scaling laws for neural language models." *arXiv preprint arXiv:2001.08361* (2020).

Scaling law according to Kaplan et al

$$L(N,S) = \left(\frac{N_c}{N}\right)^{\alpha_N} + \left(\frac{S_C}{S_{\min}(S)}\right)^{\alpha_S}$$

Scaling law according to Kaplan et al

$$L(N, S) = \left(\frac{N_c}{N}\right)^{\alpha_N} + \left(\frac{S_C}{S_{\min}(S)}\right)^{\alpha_S}$$

Number of model parameters excluding token embeddings and positional embeddings

Scaling law according to Kaplan et al

$$L(N, S) = \left(\frac{N_c}{N}\right)^{\alpha_N} + \left(\frac{S_C}{S_{\min}(S)}\right)^{\alpha_S}$$

Number of training steps

Number of model parameters
excluding token embeddings
and positional embeddings

Scaling law according to Kaplan et al

$$L(N, S) = \left(\frac{N_c}{N}\right)^{\alpha_N} + \left(\frac{S_C}{S_{\min}(S)}\right)^{\alpha_S}$$

Cross entropy loss of a transformer language model of size N when trained for S steps

Number of training steps

Number of model parameters excluding token embeddings and positional embeddings

Scaling law according to Kaplan et al

$$L(N,S) = \left(\frac{N_c}{N}\right)^{\alpha_N} + \left(\frac{S_C}{S_{\min}(S)}\right)^{\alpha_S}$$

Cross entropy loss of a transformer language model of size N when trained for S steps

Number of model parameters excluding token embeddings and positional embeddings

Number of training steps

An estimate of the minimal number of training steps needed to reach the given loss (Proportional to the number of steps, see paper for details)

Scaling law according to Kaplan et al

Constants estimated by training many small models and fitting the loss as this function of N and S

$$L(N, S) = \left(\frac{N_c}{N}\right)^{\alpha_N} + \left(\frac{S_c}{S_{\min}(S)}\right)^{\alpha_S}$$

Cross entropy loss of a transformer language model of size N when trained for S steps

Number of training steps

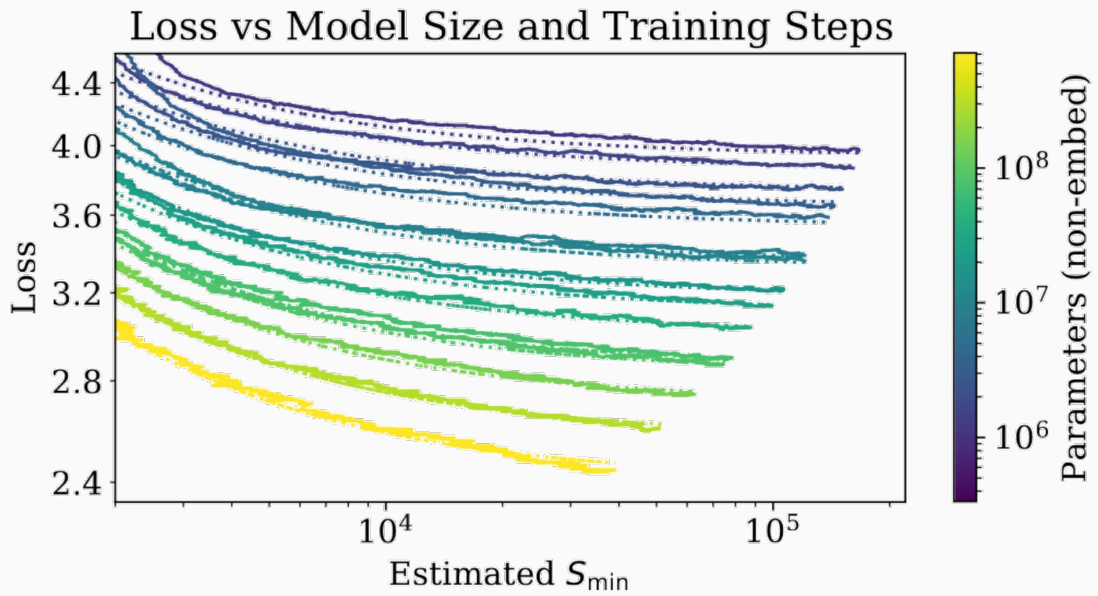
Number of model parameters excluding token embeddings and positional embeddings

An estimate of the minimal number of training steps needed to reach the given loss (Proportional to the number of steps, see paper for details)

The scaling laws seem to be empirically valid

The setup: Constants estimated by training many small, and then predict the learning curves of larger models

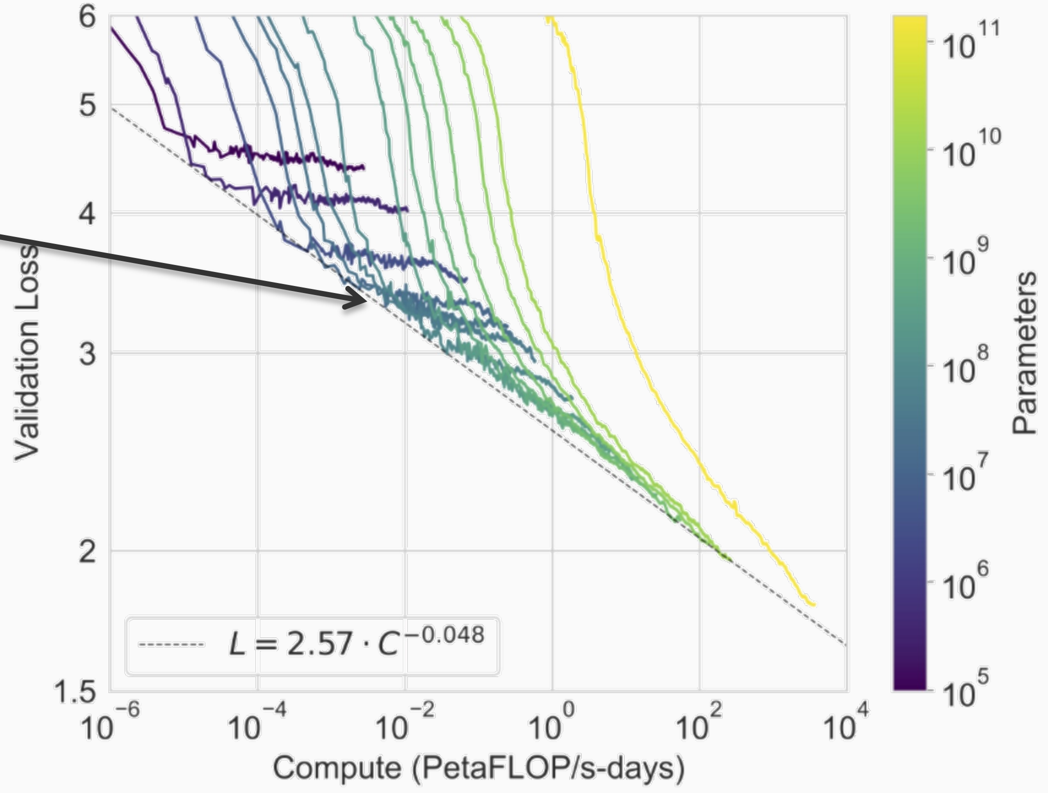
The predicted losses (dotted curves) matches the empirical learning curves (solid)



More empirical observations

A compute efficient loss frontier: For a given amount of compute, what is the best loss we can obtain?

$L \propto C^{-0.048}$



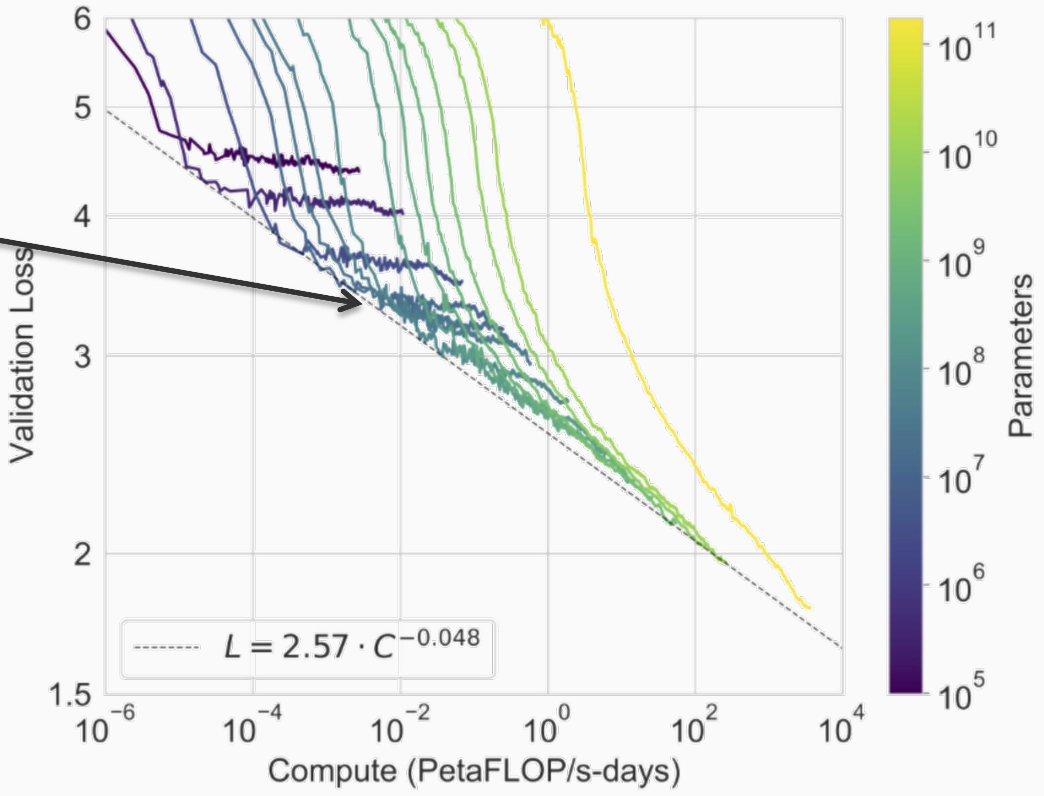
More empirical observations

A compute efficient loss frontier: For a given amount of compute, what is the best loss we can obtain?

$$L \propto C^{-0.048}$$

For a given amount of compute C

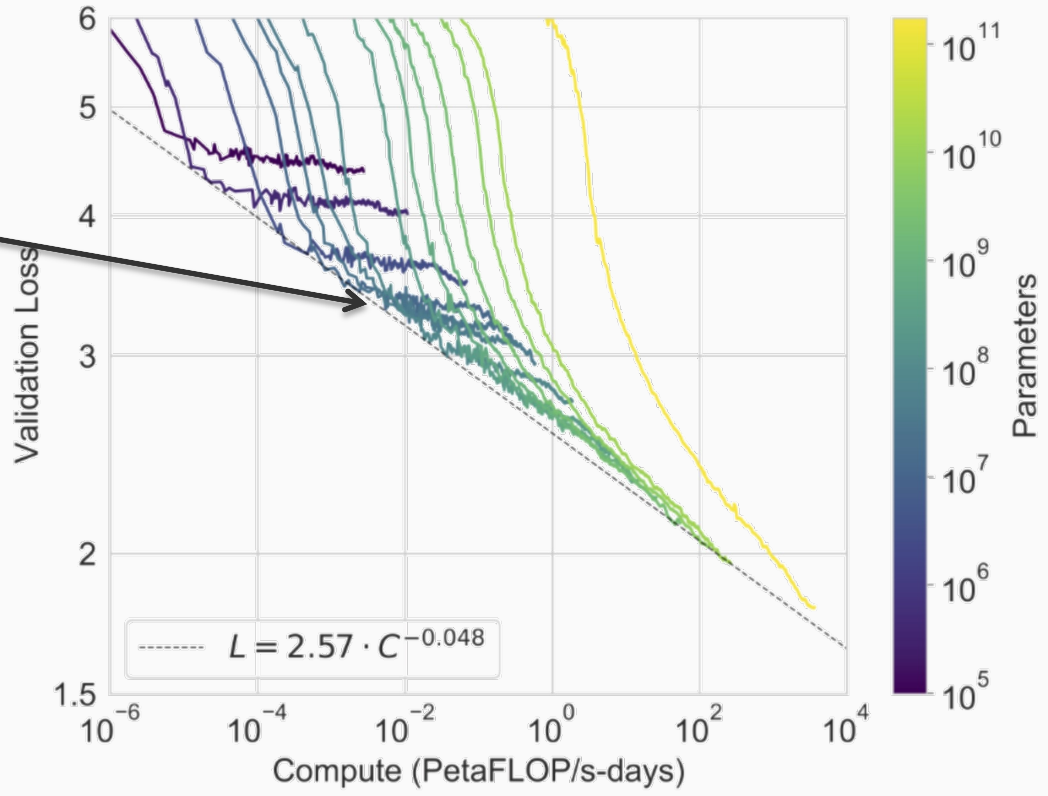
- The optimal model size is
$$N_{opt} \propto C^{0.73}$$



More empirical observations

A compute efficient loss frontier: For a given amount of compute, what is the best loss we can obtain?

$$L \propto C^{-0.048}$$



For a given amount of compute C

- The optimal model size is
  $$N_{opt} \propto C^{0.73}$$
- The optimal number of tokens
  $$D_{opt} \propto C^{0.27}$$

More empirical observations

A compute efficient loss frontier: For a given amount of compute, what is the best loss we can obtain?

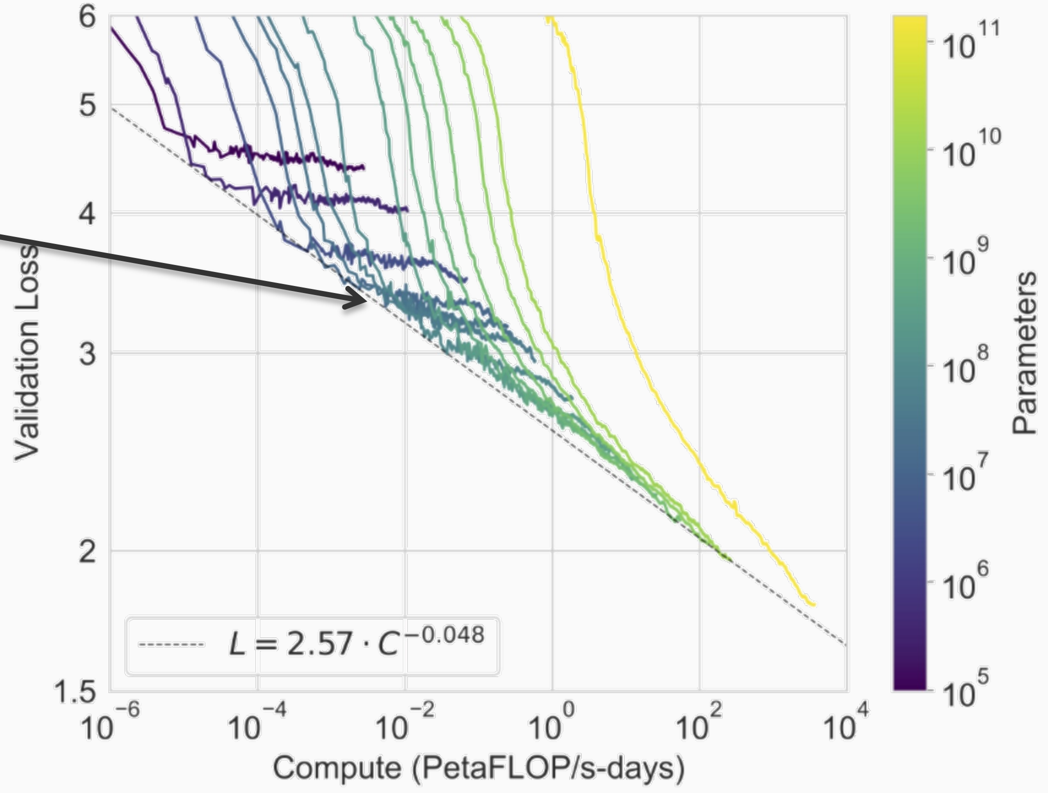
$$L \propto C^{-0.048}$$

For a given amount of compute C

- The optimal model size is
$$N_{opt} \propto C^{0.73}$$

- The optimal number of tokens
$$D_{opt} \propto C^{0.27}$$

Since the data is so large, there is only one epoch. No tokens are ever seen by the model twice during training



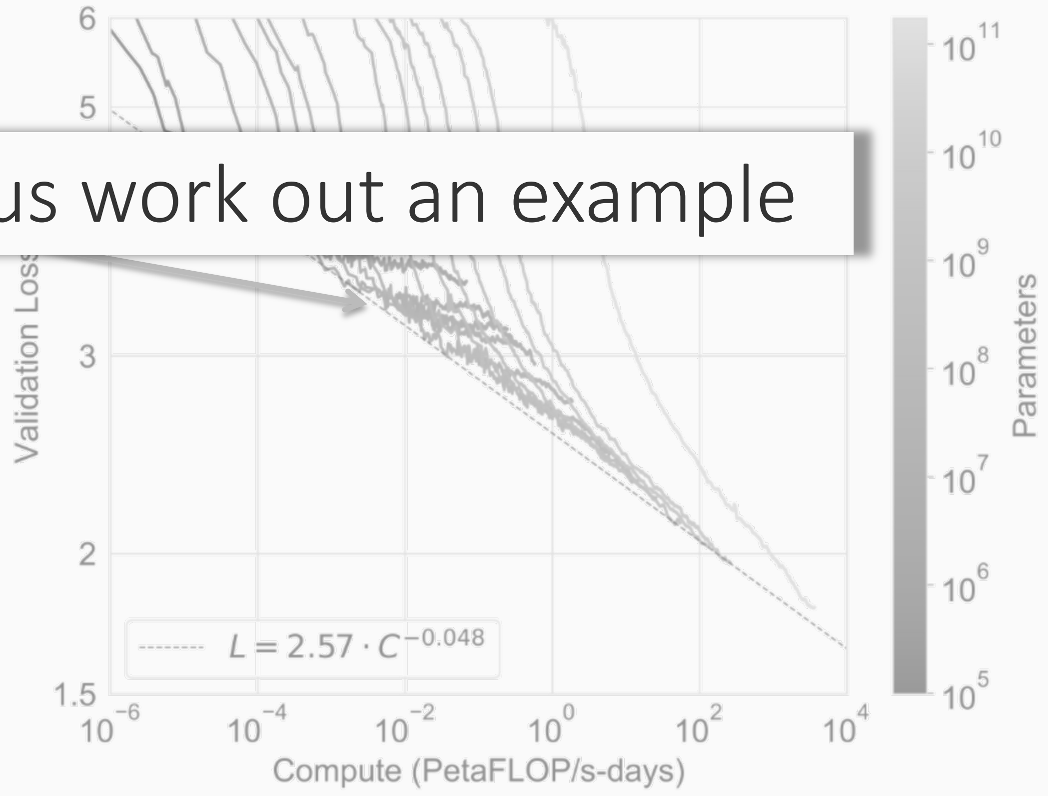
More empirical observations

A compute efficient loss frontier: For a given amount of compute, what is the best loss we can obtain?

What do these mean? Let us work out an example

For a given amount of compute C

- The optimal model size is
$$N_{opt} \propto C^{0.73}$$
- The optimal number of tokens
$$D_{opt} \propto C^{0.27}$$



Since the data is so large, there is only one epoch. No tokens are ever seen by the model twice during training

With more compute, should you increase model size or number of tokens?

Kaplan et al: For a given amount of compute C

- The optimal model size is
$$N_{opt} \propto C^{0.73}$$

- The optimal number of tokens
$$D_{opt} \propto C^{0.27}$$

With more compute, should you increase model size or number of tokens?

Kaplan et al: For a given amount of compute C

- The optimal model size is
  old $N_{opt} \propto C^{0.73}$
- The optimal number of tokens
  old $D_{opt} \propto C^{0.27}$

Suppose we have access to 100x more compute.

- new $N_{opt} \propto (100C)^{0.73}$
- new $D_{opt} \propto (100C)^{0.27}$

With more compute, should you increase model size or number of tokens?

Kaplan et al: For a given amount of compute C

- The optimal model size is
  $\text{old } N_{opt} \propto C^{0.73}$

- The optimal number of tokens
  $\text{old } D_{opt} \propto C^{0.27}$

Suppose we have access to 100x more compute.

- new $N_{opt} \propto (100C)^{0.73}$
- new $D_{opt} \propto (100C)^{0.27}$

$$\frac{\text{new } N_{opt}}{\text{old } N_{opt}} = \frac{(100C)^{0.73}}{C^{0.73}} = 100^{0.73} \approx 28.8$$

With more compute, should you increase model size or number of tokens?

Kaplan et al: For a given amount of compute C

- The optimal model size is
  $\text{old } N_{opt} \propto C^{0.73}$

- The optimal number of tokens
  $\text{old } D_{opt} \propto C^{0.27}$

Suppose we have access to 100x more compute.

- $\text{new } N_{opt} \propto (100C)^{0.73}$
- $\text{new } D_{opt} \propto (100C)^{0.27}$

$$\frac{\text{new } N_{opt}}{\text{old } N_{opt}} = \frac{(100C)^{0.73}}{C^{0.73}} = 100^{0.73} \approx 28.8$$

$$\frac{\text{new } D_{opt}}{\text{old } D_{opt}} = \frac{(100C)^{0.27}}{C^{0.27}} = 100^{0.27} \approx 3.47$$

With more compute, should you increase model size or number of tokens?

Kaplan et al: For a given amount of compute C

- The optimal model size is
  old $N_{opt} \propto C^{0.73}$
- The optimal number of tokens
  old $D_{opt} \propto C^{0.27}$

Suppose we have access to 100x more compute.

- new $N_{opt} \propto (100C)^{0.73}$
- new $D_{opt} \propto (100C)^{0.27}$

$$\frac{\text{new } N_{opt}}{\text{old } N_{opt}} = \frac{(100C)^{0.73}}{C^{0.73}} = 100^{0.73} \approx 28.8$$

$$\frac{\text{new } D_{opt}}{\text{old } D_{opt}} = \frac{(100C)^{0.27}}{C^{0.27}} = 100^{0.27} \approx 3.47$$

Increase the number of training steps by ~29x and the number of tokens seen during training only by 3.5x
We can estimate these without having to actually train the model.

With more compute, should you increase model size or number of tokens?

Kaplan et al: For a given amount of compute C

- The optimal model size is
  $\text{old } N_{opt} \propto C^{0.73}$
- The optimal number of tokens
  $\text{old } D_{opt} \propto C^{0.27}$

Suppose we have access to 100x more compute.

- new $N_{opt} \propto (100C)^{0.73}$
- new $D_{opt} \propto (100C)^{0.27}$

$$\frac{\text{new } N_{opt}}{\text{old } N_{opt}} = \frac{(100C)^{0.73}}{C^{0.73}} = 100^{0.73} \approx 28.8$$

$$\frac{\text{new } D_{opt}}{\text{old } D_{opt}} = \frac{(100C)^{0.27}}{C^{0.27}} = 100^{0.27} \approx 3.47$$

Increase the number of training steps by ~29x and the number of tokens seen during training only by 3.5x
We can estimate these without having to actually train the model.

GPT-3 used this recipe to train a 175B model on 300B tokens

Subsequent developments

Hoffman et al (2022) noted that the Kaplan results were based on all experiments using the same learning rate schedule

Subsequent developments

Hoffman et al (2022) noted that the Kaplan results were based on all experiments using the same learning rate schedule

- Changing the learning rate schedule so that the learning rate reaches zero at the end of training gives different constants in the expression

Subsequent developments

Hoffman et al (2022) noted that the Kaplan results were based on all experiments using the same learning rate schedule

- Changing the learning rate schedule so that the learning rate reaches zero at the end of training gives different constants in the expression

- Both N and D are equally important

$$N_{opt} \propto C^{0.5}, D_{opt} \propto C^{0.5}$$

  - Implication: If we have more compute, grow number of steps and number of tokens equally

Subsequent developments

Hoffman et al (2022) noted that the Kaplan results were based on all experiments using the same learning rate schedule

- Changing the learning rate schedule so that the learning rate reaches zero at the end of training gives different constants in the expression

- Both N and D are equally important

$$N_{opt} \propto C^{0.5}, D_{opt} \propto C^{0.5}$$

  - Implication: If we have more compute, grow number of steps and number of tokens equally

- Trained a 70B model on 1.4T tokens (Chinchilla) outperforming their previous 280B model trained on 300B tokens (Gopher)

Final words

- Scaling laws: Empirical observations that relate model size, compute in FLOPs, training size and loss functions. Typically power law relationships

- These are empirical observations. There is very little theoretical understanding

- But why did we not see this coming? Because learning theory does not really like overparameterized models
  - Learning theory: “Overparameterization = high capacity = low generalization”
  - Empirical evidence: Making models bigger makes them generalize better!

  Perhaps there is room for new theory. A promising direction involves the so called “double descent” curve of Belkin et al 2018.