



**Barcelona
Supercomputing
Center**

Centro Nacional de Supercomputación

Language Models at Scale

Jordi Armengol-Estapé (@jordiae)

Life Sciences Department

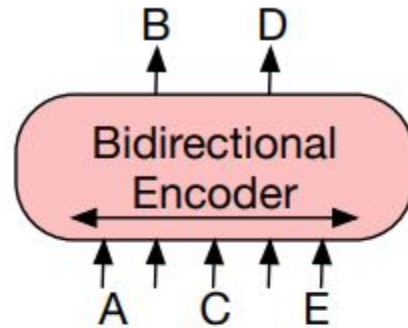
Recent work at TeMU

- Language-specific models:
 - Spanish: *Spanish Language Models*. Asier Gutiérrez-Fandiño*, Jordi Armengol-Estapé*, Marc Pàmies, Joan Llop-Palao, Joaquín Silveira-Ocampo, Casimiro Pio Carrino, Aitor Gonzalez-Agirre, Carme Armentano-Oller, Carlos Rodriguez-Penagos, Marta Villegas. Preprint.
 - ...
- Domain-specific models:
 - Spanish Legal Domain (coming soon).
 - Spanish Biomedical Domain (coming soon).
 - ...
- In this talk, we are going to go through the challenges and benefits of **scaling language models**.
- Disclaimer: Figures have been shamelessly stolen from OpenAI articles.

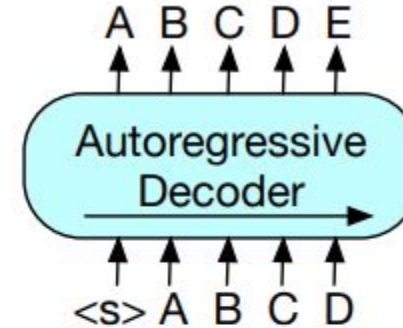
Outline

1. **Introduction**
2. Scale Science
3. Scale Engineering
4. What's next

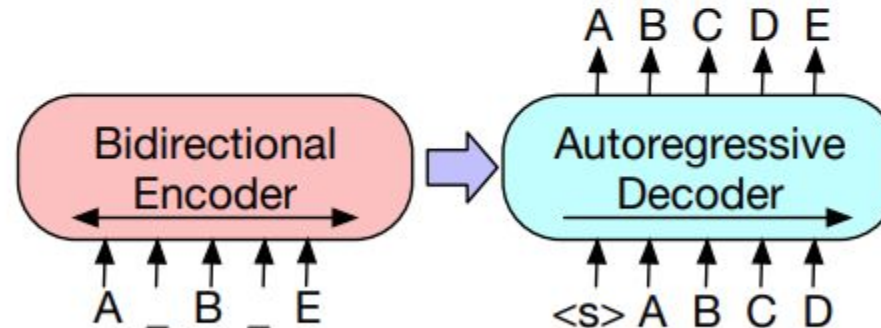
Language Modeling (the tasks)



Masked Language Modeling (BERT):
Good for discriminative tasks



(Classical, Autoregressive) Language
Modeling (GPT)
Good at unconstrained generation
(but virtually capable of any other task)



Denoising autoencoding (BART):
Good for sequence-to-sequence tasks

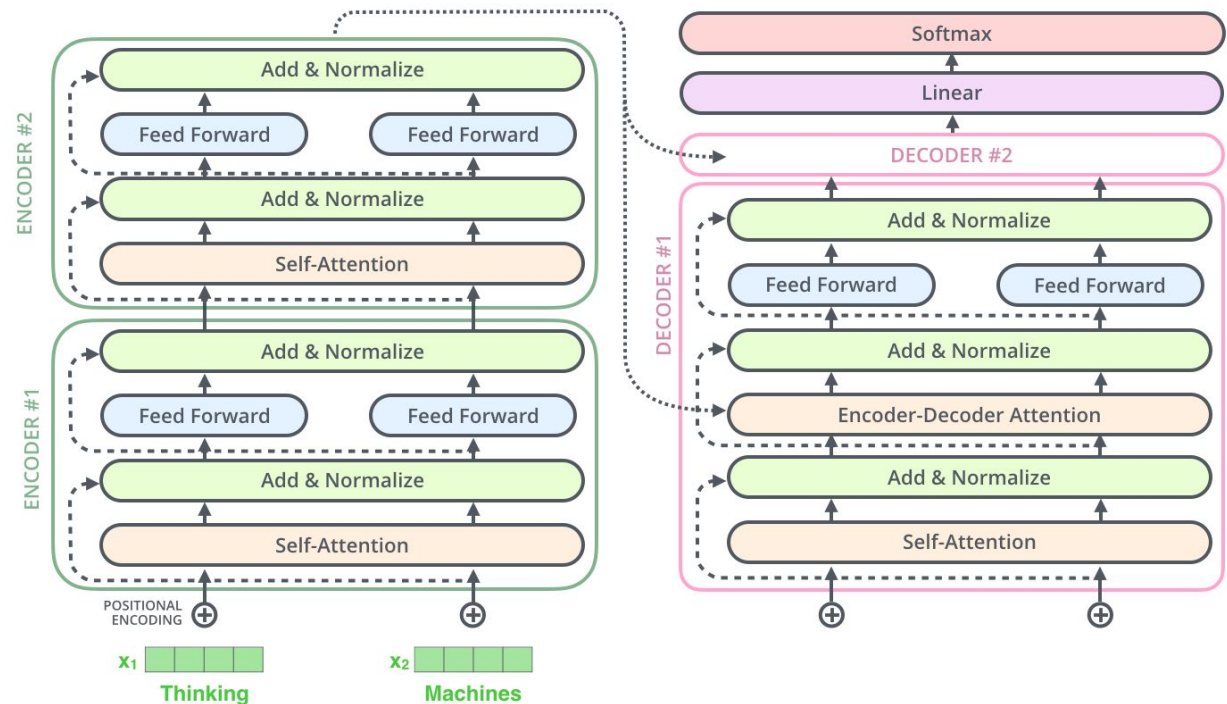
Language Models (the models themselves)

Many choices!

- Can we use a non-deep learning based algorithm? I *strongly* advise not to do that.
- RNNs are the intuitive, natural choice.
- Can we use convolutional neural networks? Yes, with remarkable success.
- But... you don't mean we can use MLPs? Yes, we can, with impressive results (MLP mixer, etc).

Transformers just **scale*** better.

*wait, but what does *scale* mean? We will see.



Why do we care about language modeling?

1. The task itself has interesting applications:
 - a. Smartphone predictive keyboards.
 - b. Code completion.
 - c. Etc.
2. Surrogate task for learning about our domain/modality.
Can be applied in downstream tasks (transfer learning)
3. Virtually infinite streams of data (self-supervised learning).
4. Virtually **all* computable tasks** can be casted as language (sequence) modeling tasks:
E.g., in vision: next pixel/patch/codework prediction.
5. Transformers scale well. Transformers can be used as language models out of the box.
Therefore, let's do language modeling!

*Can you think of a counter-example? Because I can't.

Downstream applications

- (Very) old: feature extraction.
- Old (?): Fine-tuning, adapters.
- New: **few-shot** learning.

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 cheese => ..... ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← example
3 cheese => ..... ← prompt
```

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← examples
3 peppermint => menthe poivrée ←
4 plush girafe => girafe peluche ←
5 cheese => ..... ← prompt
```

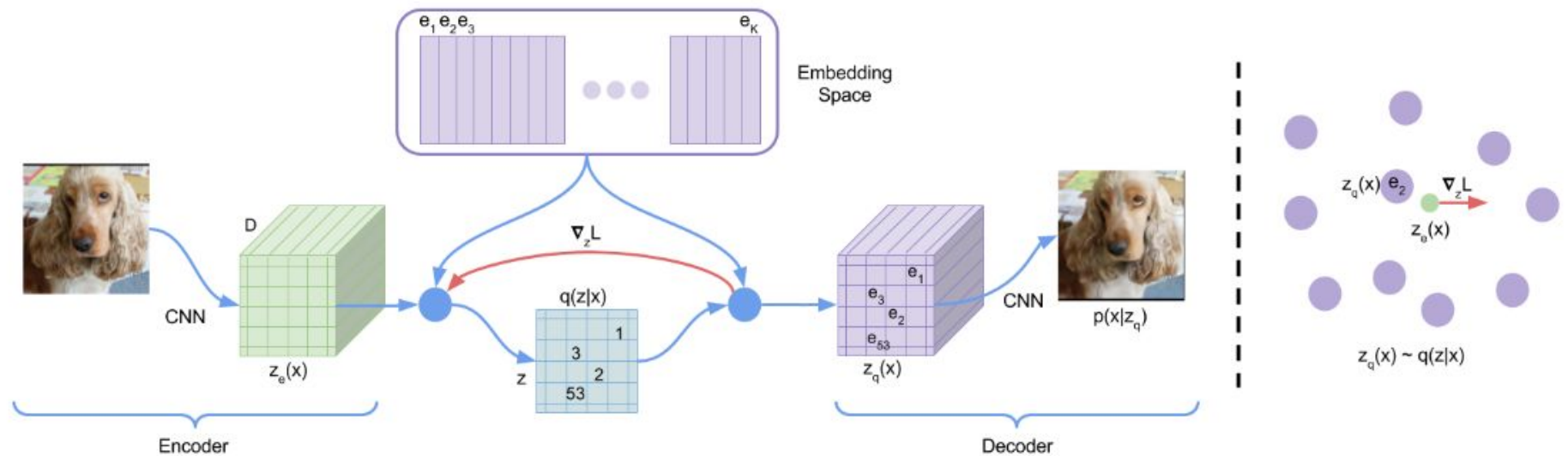
Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.



Modalities

1. Text-based (natural language, code, chemical compounds,...):
Use the Transformer out-of-box, optionally with (B)BPE encoding.
2. Non-differentiable (MDPs, POMDPs): i) Explore with some policy π . ii) Likelihood-based imitation learning.
3. Continuous:
 - a. Direct: Model raw data taking pixels, patches, etc.
 - b. **Indirect:**
 - i. Discretize your data with a VQ-based model (VAE, GAN).
 - ii. Use the
 - iii. Profit.
4. Multi-modal?
just concatenate.
5. Long sequences?
Efficient, even linear-time
Transformer
variants.



Scale

Just scaling language models up does **NOT** show the diminishing results one would expect.

More on that later.

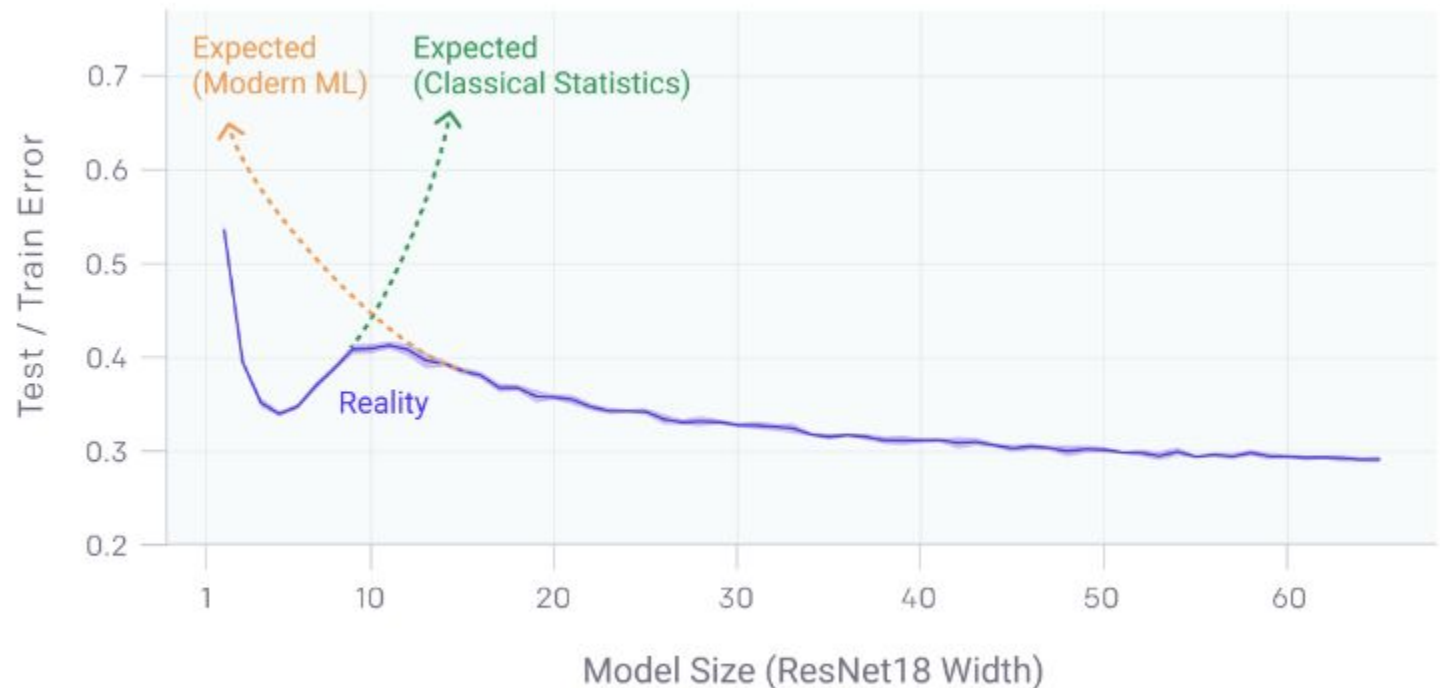
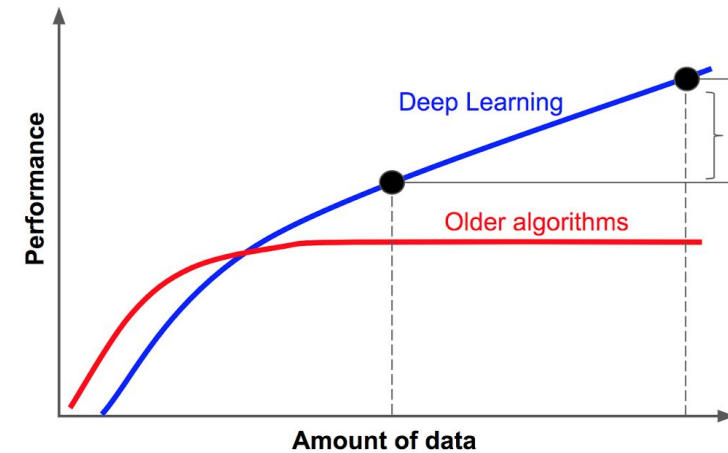
Outline

1. Introduction
2. **Scaling Science**
3. Scaling Engineering
4. What's next

Scaling Machine Learning

- Classical statistical/machine learning view: Normal things happen.
- Deep learning view: **very weird** (even *AGI-ish*) things happen at scale:
 - Double Descent.
 - (Grokking).
 - **Scaling Laws.**

Analogous to classical (macro) vs quantum (micro) mechanics



Scaling Laws

Originally formulated by **OpenAI** researchers:

- Scaling Laws for Neural Language Models (Kaplan, 2020) (OpenAI)
- Scaling Laws for Autoregressive Generative Modeling (Henighan, 2020) (OpenAI)
- Scaling Laws for **Transfer** (Hernandez, 2021) (OpenAI)

Now starting to gain momentum:

- Explaining Neural Scaling Laws (Bahri, 2021) (**Deepmind**)

Scaling Laws: Results

- *Performance depends strongly on the **scale**:*
 - Compute (FLOPs), C .
 - Data size (tokens), D .
 - Model size (number of non-embedding parameters), N .
- And very **weakly on the model shape (hyperparameters such as depth and width)**.
- They are called *laws* because they are physics-like formulae verified empirically.

Scaling Laws: The Basics

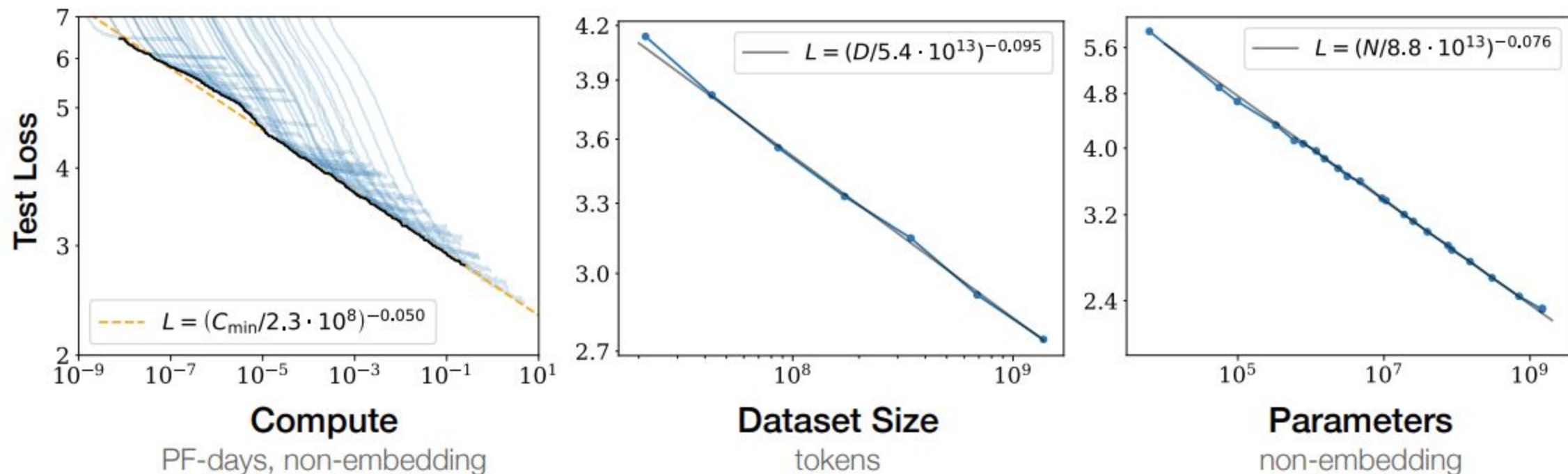
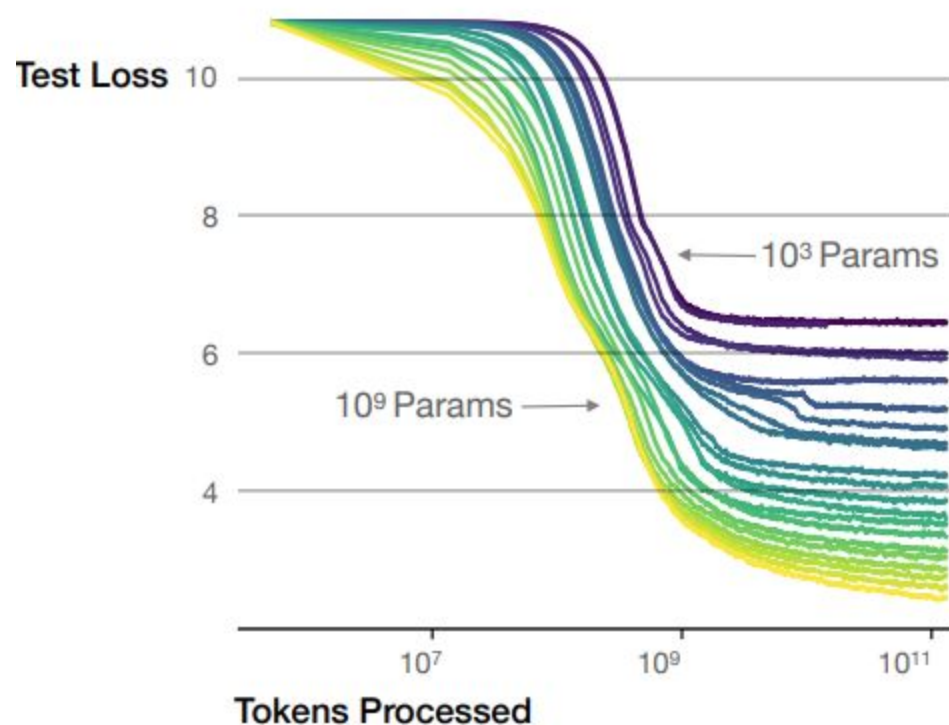


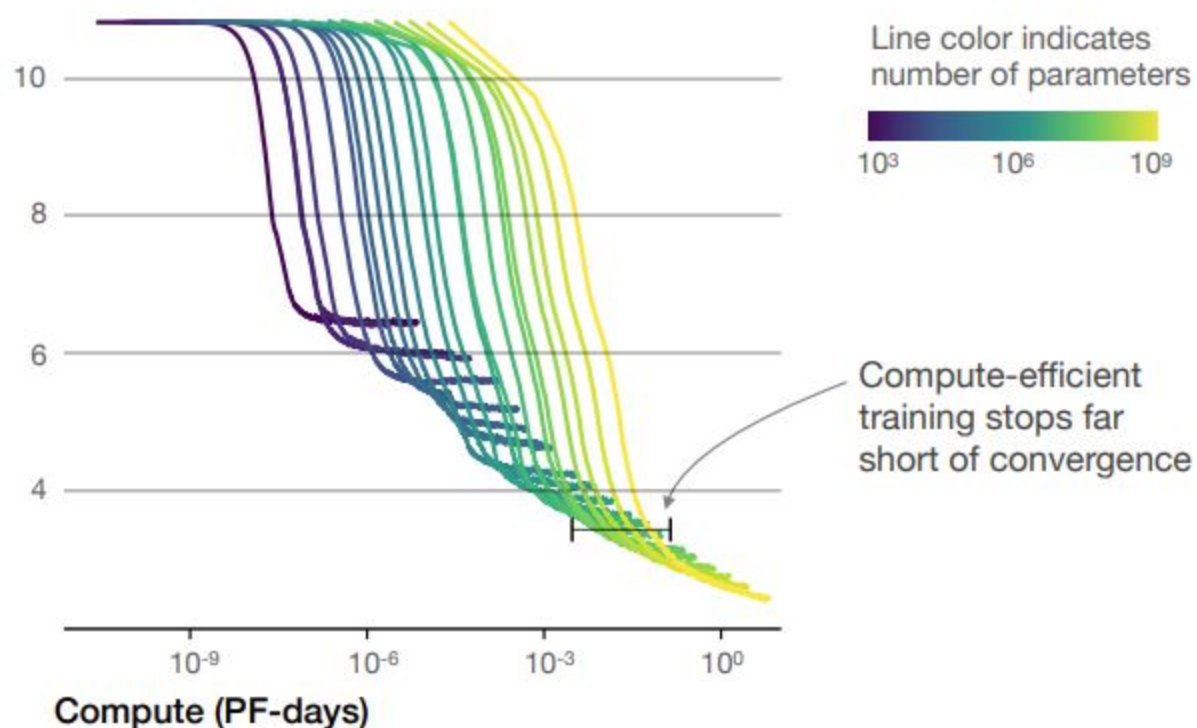
Figure 1 Language modeling performance improves smoothly as we increase the model size, dataset size, and amount of compute² used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

Scaling Laws: Optimal Compute Allocation

Larger models require **fewer samples** to reach the same performance



The optimal model size grows smoothly with the loss target and compute budget



Scaling Laws: Optimal Compute Allocation

For a given compute budget (e.g., GPU hours), it's better to train a larger model for less iterations.

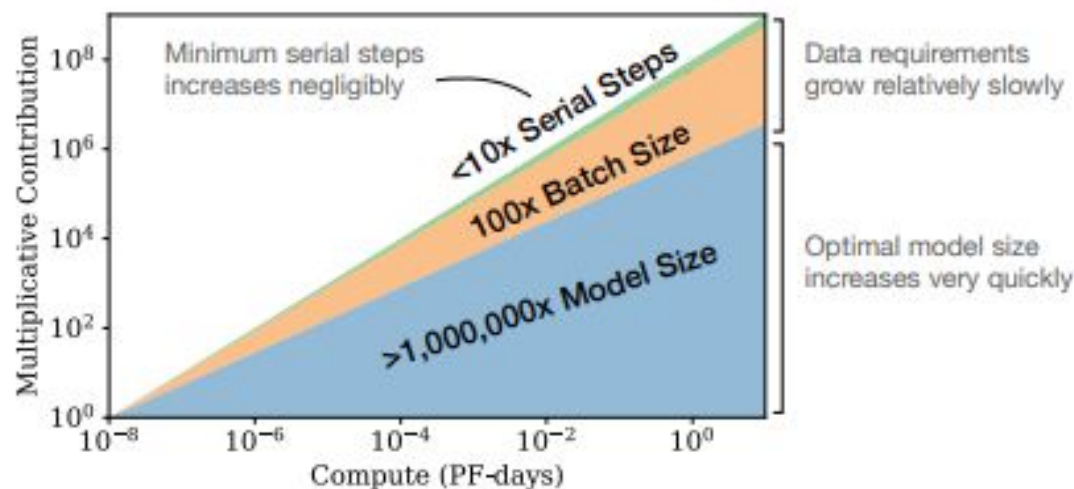


Figure 3 As more compute becomes available, we can choose how much to allocate towards training larger models, using larger batches, and training for more steps. We illustrate this for a billion-fold increase in compute. For optimally compute-efficient training, most of the increase should go towards increased model size. A relatively small increase in data is needed to avoid reuse. Of the increase in data, most can be used to increase parallelism through larger batch sizes, with only a very small increase in serial training time required.

Scaling Laws: Results

- *Convergence is inefficient.*
- Test loss correlates with transfer loss.
- ...

Scaling Laws: Extensions

- Other modalities: OpenAI applied the same setup to image modeling, video, text-to-image, image-to-text, and mathematics modeling and found scaling laws **holding across ALL modalities**.
 - They use the exact same model (GPT) for all modalities, with the exact same training procedure!
- OpenAI also did a more in-depth study of the scaling laws of transfer (specifically, from English to Python).

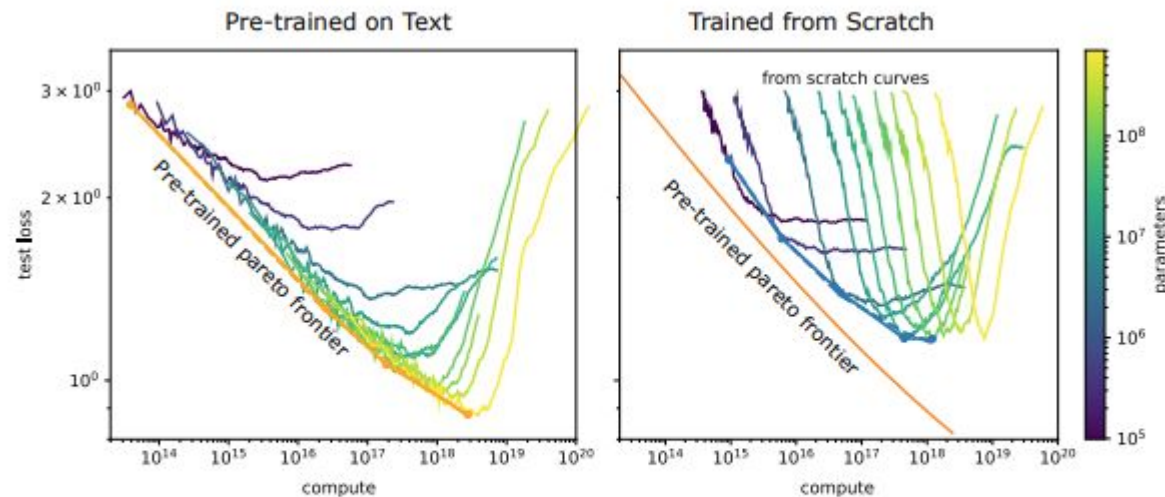


Figure 4 In the low data regime, fine-tuning gets better performance than training from scratch for a given amount of training compute and it's much easier to be on the compute efficient frontier. The performance gap widens severely as the model size grows with a fixed amount of python. (Curves for 3e8 python characters)

Outline

1. Introduction
2. Scale: Science (?)
3. **Scale: Engineering (?)**
4. What's next

Well-known facts in language modeling?

1. Data are the main bottleneck?
2. You need to train for many epochs?
3. You need regularization?
4. Large models are less sample-efficient?

Well-known facts in language modeling?

~~1. Data are the main bottleneck~~

OpenAI just used a tiny portion of the English CommonCrawl for training GPT-3.

Virtually infinite streams of data: Youtube, books, CCTV, football matches, music, large internet crawls.

OpenAI successfully fine tuned a huge model on a dataset with **less than 100 samples**.

~~2. You need to train for many epochs~~

One epoch is all you need (Komatsuzaki, 2019).

~~3. You need regularization~~

Virtually impossible to overfit in a huge, diverse dataset, in a single epoch.

Latest OpenAI models do NOT use dropout.

~~4. Large models are less sample-efficient~~

Larger models are MORE sample-efficient!

We've found we can improve language model behavior with respect to specific behavioral values by fine-tuning on a curated dataset of **< 100 examples** of those values. We also found that **this process becomes more effective as models get larger.** While the technique is still nascent, we're looking for OpenAI API users who would like to try it out and are excited to find ways to use these and other techniques in production use cases.

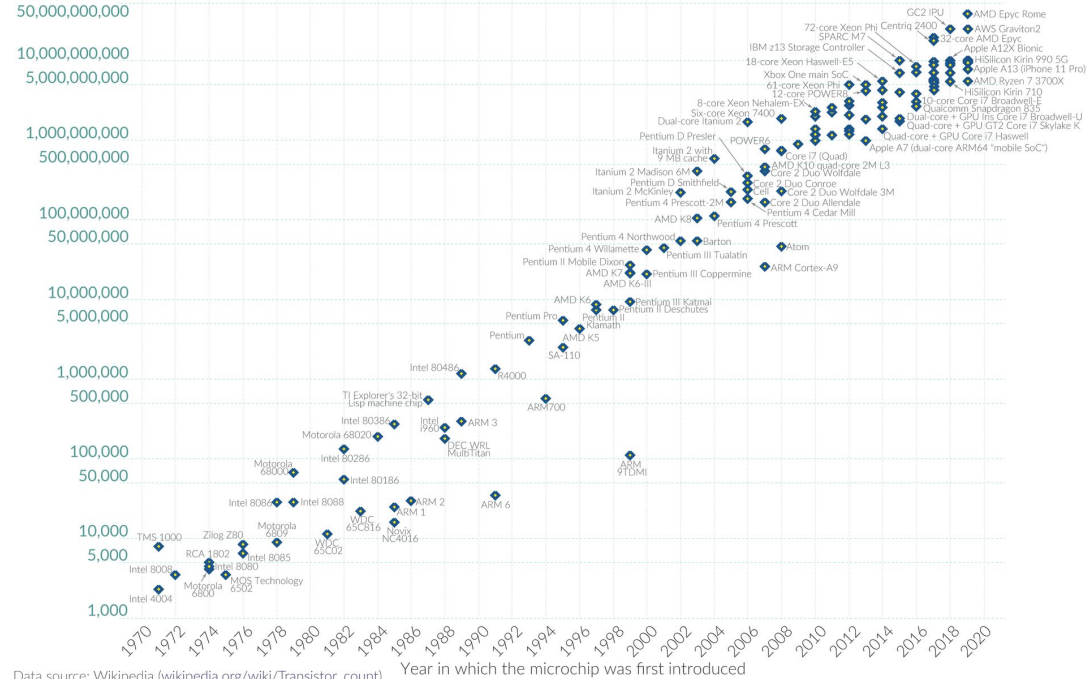
Moore's law?

Moore's Law: The number of transistors on microchips doubles every two years

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.

Our World
in Data

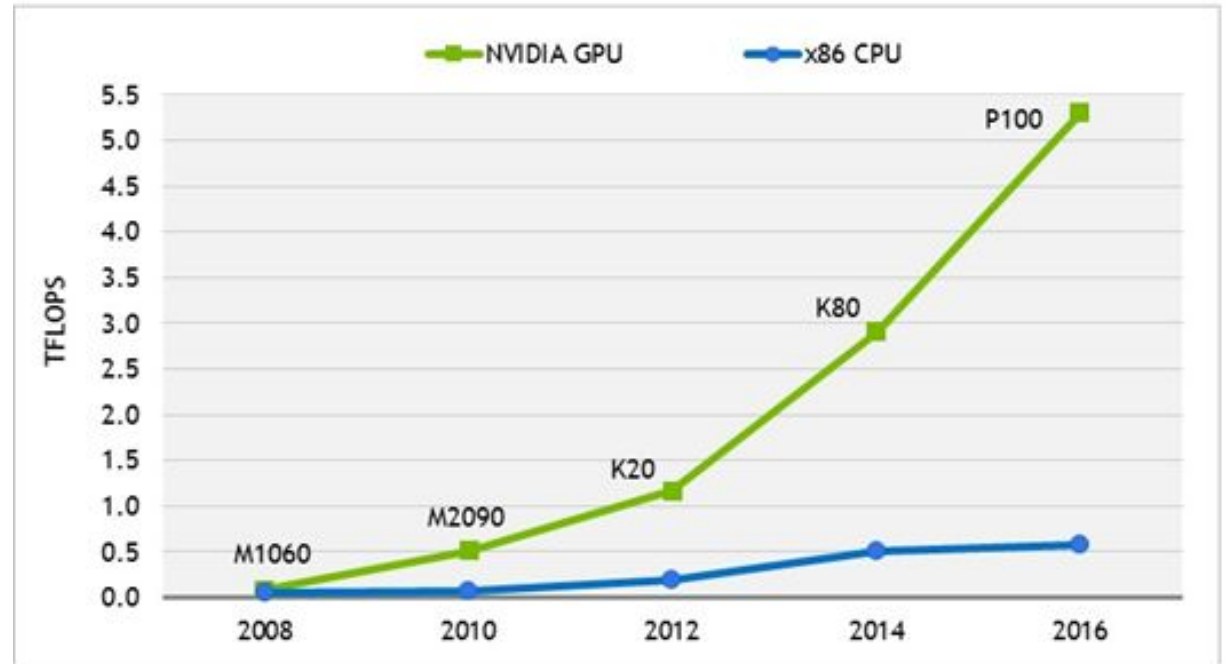
Transistor count



Data source: Wikipedia ([wikipedia.org/wiki/Transistor_count](https://en.wikipedia.org/wiki/Transistor_count))

OurWorldinData.org – Research and data to make progress against the world's largest problems.

Licensed under CC-BY by the authors Hannah Ritchie and Max Roser.



Moore's Law?

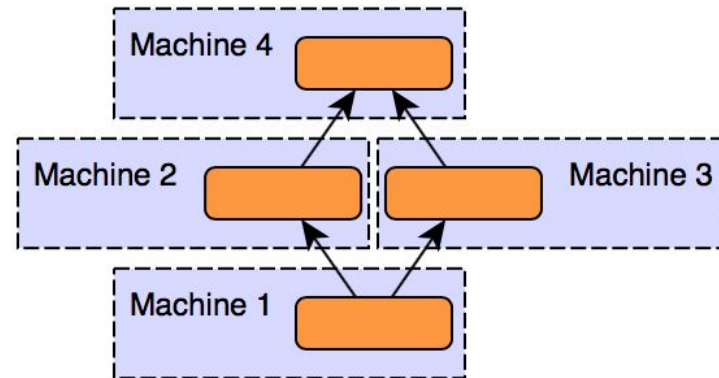
- Massively parallel GPU-based programming has kept Moore law kind of alive.
- However, both RAM and VRAM haven't keep up.
 - And we said that model size was the key ingredient!
- GPUs are **very** expensive (thanks, Dogecoin...).
- Hardware bottleneck.

Scaling model size

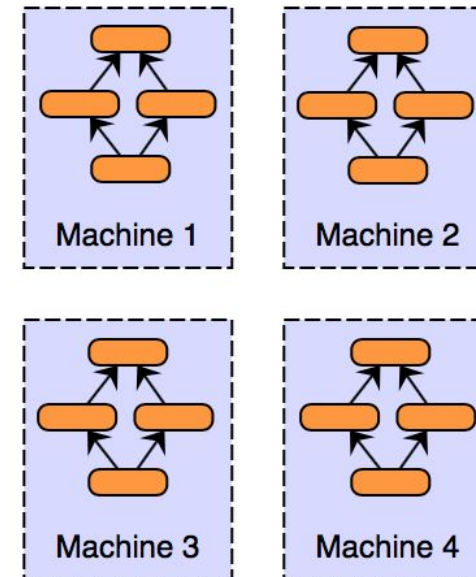
- The theory says that we should prioritize model scaling.
- In practice, this presents some challenges:
 - Memory!
 - Numerical instability
 - Batch size.

The limits of data parallelism

Model Parallelism

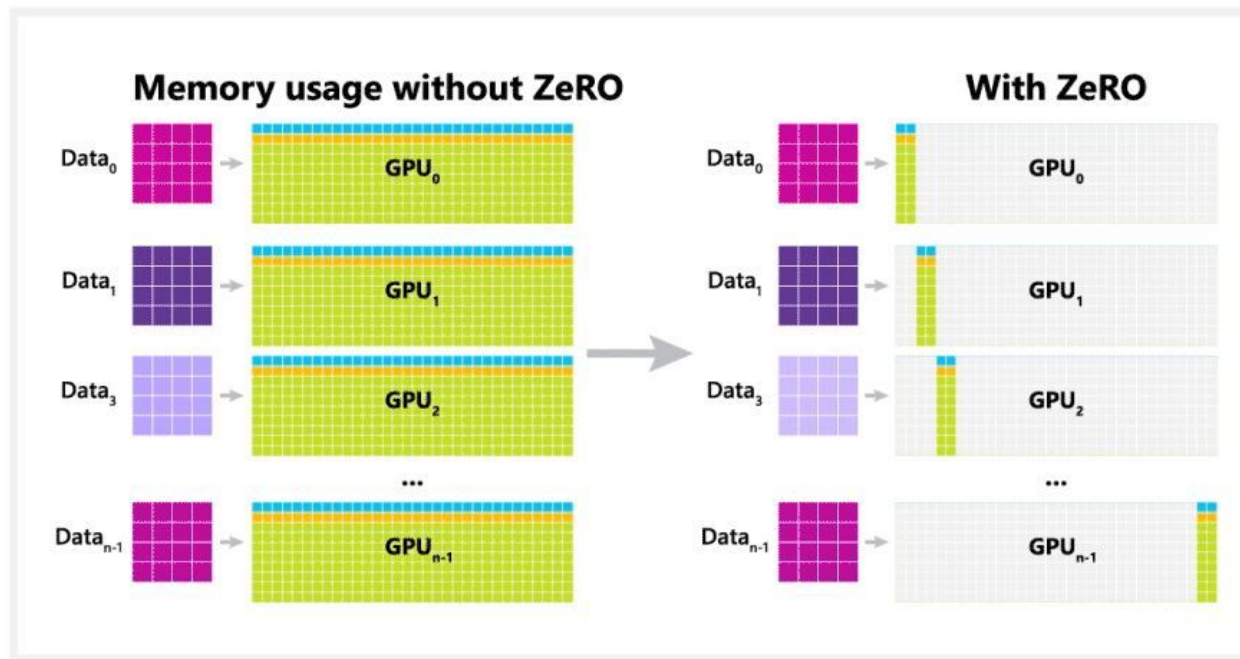


Data Parallelism



Deepspeed

DeepSpeed + ZeRO



Scale

- 100B parameter
- 10X bigger

Speed

- Up to 5X faster

Cost

- Up to 5X cheaper

Usability

- Minimal code change

Outline

1. Introduction
2. Scale: Engineering
3. Scale: Science
4. **What's next**

1. OpenAI & co plans

- OpenAI is probably working on multi-modal GPT and a scaled up version of video GPT (subtle hints on Twitter and papers).
- Google/Deepmind are jumping on the scaling laws bandwagon (Explaining Neural Scaling Laws).

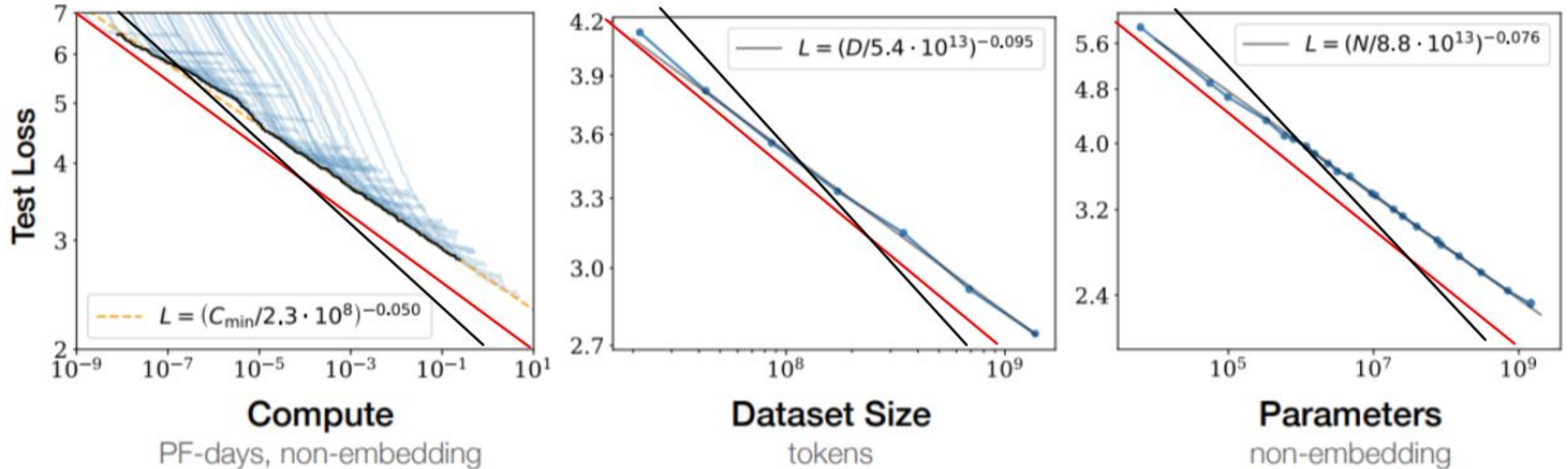
Towards Scaling-Laws-aware ML Research

- What if the majority of algorithms supposedly outperforming their baselines are actually just an **artifact** of the specific (typically small) scale in which their were evaluated?
- We can't* run OpenAI-scale models...
- ... but compute gets cheaper every year.
- Researchers should **test their method in different compute, model size, and data regimes** to show that it **scales** better than the baseline.
- Toy problems are unrealistic. But some “artificial” low-resource scenarios (standard benchmarks with a given data size) are unrealistic too.
- Still, there is no guarantee that the improvements will hold in OpenAI-like scales.
 - See “L11 Language Models -- guest instructor: Alec Radford (OpenAI) --- Deep Unsupervised Learning SP20”
- Plus, some computationally interesting methods are actually more difficult to scale than others due to hardware constraints.
 - *The Hardware Lottery* (S. Hooker, 2020).

*We could. It's just ~€1B/year \approx 0.2% of the Spanish Government annual budget.

Scaling Laws as a research field on its own

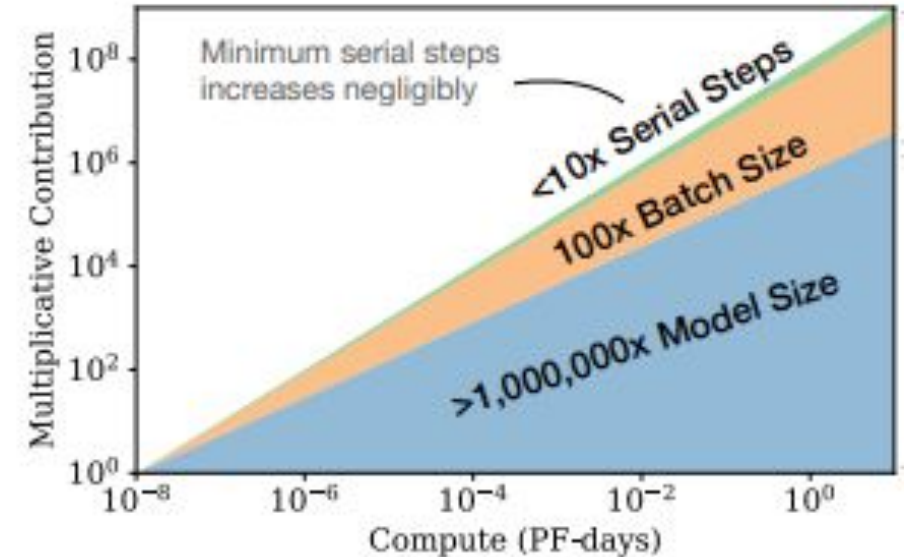
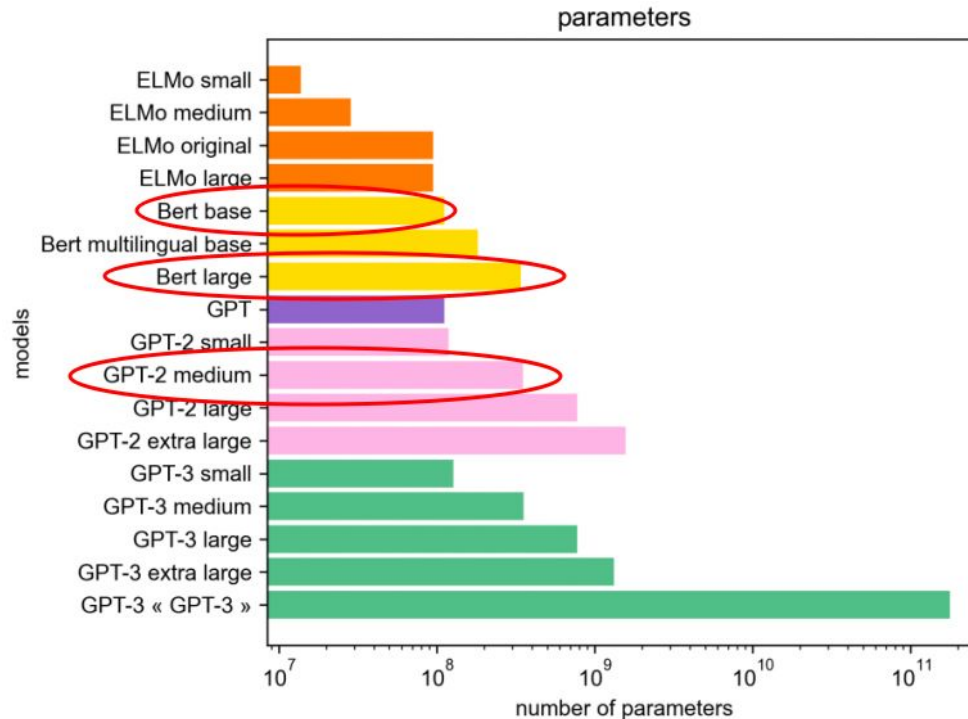
- Propose a method with **advantageous scaling laws**.



- Scale Scaling Laws Themselves: *Scaling Scaling Laws with Board Games* (Jones, 2021)


Back to TeMU models

- Data size: BNE dataset is almost the same order of magnitude than the dataset sampled in GPT-3!
 - Our clean crawling: 576 GB. OpenAI's sample: 570 gb
 - OpenAI added Books corpus, Wikipedia, etc.
- Batch size:
 - GPT-3 batch size is only 3 times bigger than TeMU's models (10,48,576 tokens).
- Model size: > 230x




Addendum: PyTorch DL/NLP libraries

- Consider *not* using PyTorch: JAX is gaining momentum.
- Huggingface:
 - Excellent for model sharing, inference, fine tuning.
 - Easy to convert from JAX-TF-PyTorch.
 - The trainer has subtle bugs.
 - In our experience, considerably slower than Fairseq.
- Fairseq:
 - Problem: it's a framework instead of a library.
 - If your use case fits, it's very fast.
 - Many tricks (e.g., loss scaling)
 - Convert trained models to HF
- Mistral:
 - Stanford's library for training big LMs.
 - Based on HF
- Eleuther AI libraries:
 - GPT Neo: TF.
 - GPT Neox: PyTorch.
 - GPT-J: Jax.
- Lightning Transformers?

 **Siddharth Karamcheti** @siddkaramcheti · Aug 24
Replying to @siddkaramcheti and @BramVanroy

To be more concrete; in our talk earlier today at the Workshop, @laurel_orr1 and I talked about stability issues with GPT training; these fixes involve things like upcasting the scaled dot-prod attn to FP32 when doing mixed precision: github.com/stanford-crfm/... (2/N)

**stanford-crfm/
mistral**



Mistral: A strong, northwesterly wind: Framework for transparent and accessible large-scale language model training, built with Hugging Face 🤗


Transformers...

| | | | |
|--------------|--------|-------|-------|
| 7 | 8 | 216 | 13 |
| Contributors | Issues | Stars | Forks |

[mistral/mistral_gpt2.py](#) at main · stanford-crfm/mistral

Mistral: A strong, northwesterly wind: Framework for transparent and accessible large-scale language model training, built with Hugging Fac...
[github.com](#)

2 ↕ ❤ ↑

 **Siddharth Karamcheti** @siddkaramcheti · Aug 24

There are other things; re-arranging the order of operations in scaled dot-prod -- scale K by 1/root(dk) first, prior to dot product github.com/stanford-crfm/...

Also, borrowing from Megatron-LM -- scale further by 1 / layer_idx to prevent overflow: github.com/stanford-crfm/... (3/N)

↩ ↕ 📄 ↑



**Barcelona
Supercomputing
Center**

Centro Nacional de Supercomputación

Language Models at Scale

Thanks. Questions?

Life Sciences Department Seminar