5 years of GPT progress

A short history

Finbarr Timbers, Artificial Fintelligence
finbarrtimbers.substack.com
@finbarrtimbers

What is a GPT?

What is a GPT?

Generative

Pretrained

Transformer

Why do we care?

Why do we care?

It works *really* well.

Background: tokens

A token is a numerical representation of text.

Background: tokens

Mental model:

A = 1, B = 2, C = 3, etc.

Background: tokens

Mental model:

A = 1, B = 2, C = 3, etc.

(Not quite accurate)

Background: tokens

Plain text:

“Hello, world!”

Encoded text (using GPT-4 tokenizer):

[9906, 11, 1917, 0]

Timeline

2018: GPT (117M parameters, 20M tokens, multiple epochs)

- Decoder architecture, context length 512, GELUs

“Improving Language Understanding by Generative Pre-Training”

Timeline

2018: GPT (117M parameters, 20M tokens)

- Decoder architecture, context length 512, GELUs

2019: GPT-2 (1.5B parameters, 9B tokens, 1 epoch)

- Slightly different norms, context length 1024, larger batch size

“Language Models are Unsupervised Multitask Learners”

Timeline

2018: GPT (117M parameters, 20M tokens)

- Decoder architecture, context length 512, GELUs

2019: GPT-2 (1.5B parameters, 9B tokens)

- Slightly different norms, context length 512, larger batch size

2020: GPT-3 (175B parameters, 300B tokens)

- Context length of 2048, and alternating sparse/dense attention

“Language Models are Few-Shot Learners”

Timeline

2018: GPT (117M parameters, 20M tokens)

- Decoder architecture, context length 512, GELUs

2019: GPT-2 (1.5B parameters, 9B tokens)

- Slightly different norms, context length 512, larger batch size

2020: GPT-3 (175B parameters, 300B tokens)

- Context length of 2048, and alternating sparse/dense attention

2022: ChatGPT (GPT-3 + RLHF)

Observations

- The original GPT paper, and even GPT-2 to a certain extent, seem like side projects, and their positioning isn't particularly exciting
- The major difference between these is "just" scale, not really any architectural differences
- Very few ablations were done
- Not a lot of *science*
- None of this was published at a peer reviewed venue!

Simultaneously...

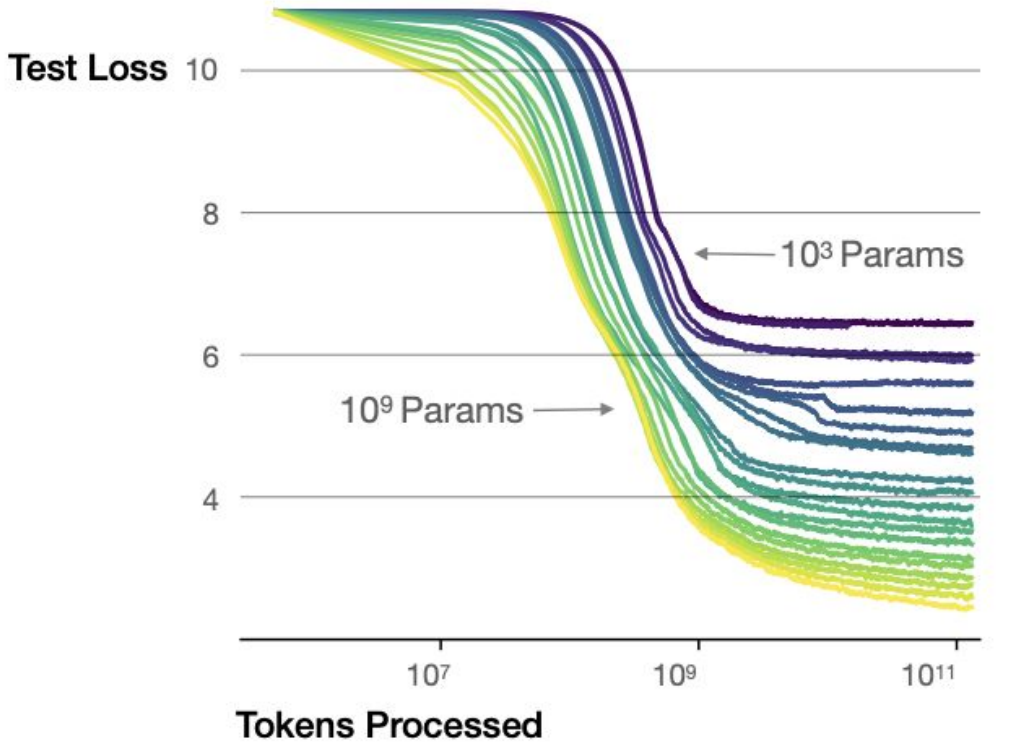
Kaplan et. al was published in 2020. Some conclusions:

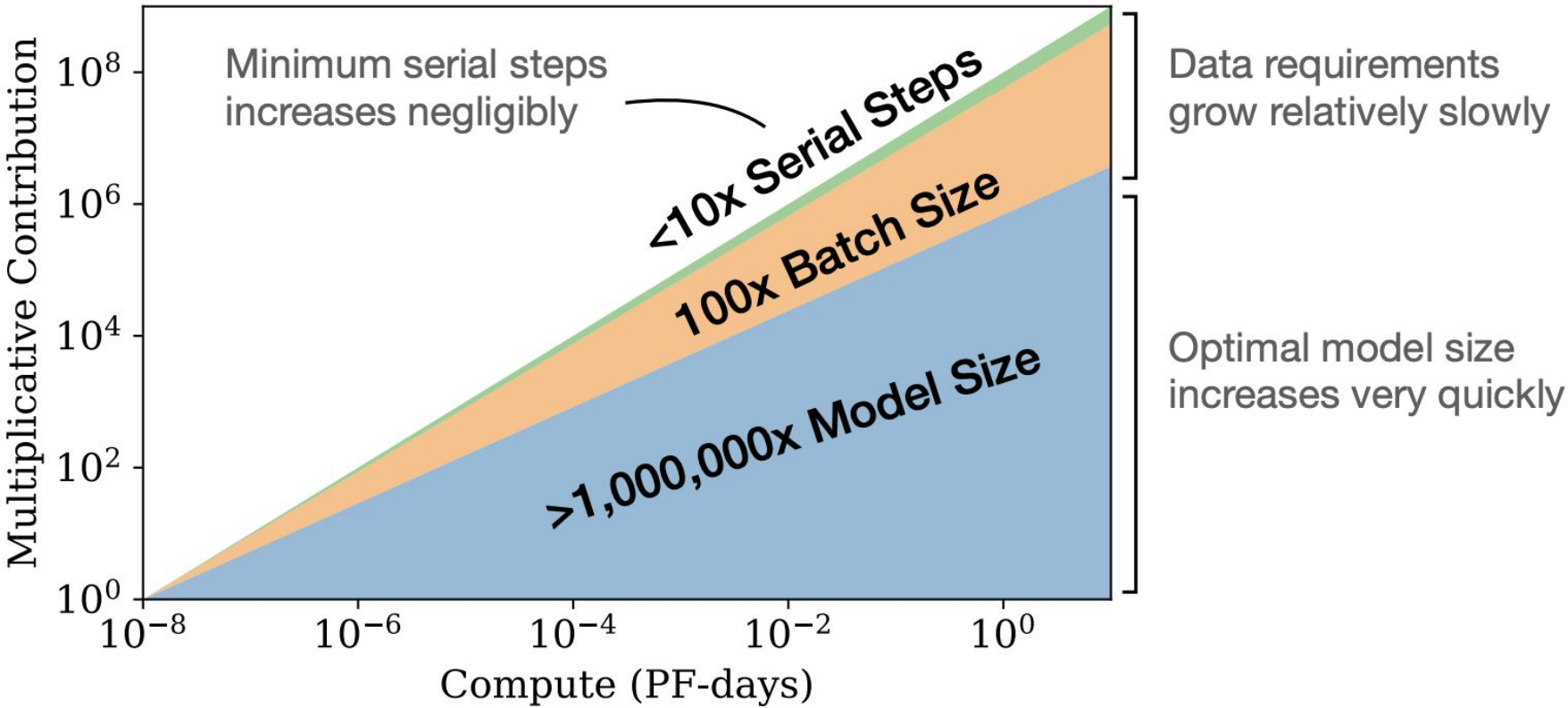
- Model architecture (width/depth) doesn't really matter
- Scale is basically all that matters (# of parameters)
- Sample efficiency increases with scale

Kaplan

$$L(N, D) = \left[\left(\frac{N_c}{N} \right)^{\frac{\alpha_N}{\alpha_D}} + \frac{D_c}{D} \right]^{\alpha_D}$$

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




Minimum serial steps
increases negligibly
<10x Serial Steps
100x Batch Size
>1,000,000x Model Size
Multiplicative Contribution
10^8
10^6
10^4
10^2
10^0
10^{-8}
10^{-6}
10^{-4}
10^{-2}
10^0
Compute (PF-days)
Data requirements
grow relatively slowly
Optimal model size
increases very quickly

Kaplan's conclusion

Model size is all you need.

Kaplan's conclusion

Model size is all you need.

But…

This wasn't entirely correct.

Chinchilla

Another scaling law paper from DeepMind. Conclusions:

- Data is the constraint in a lot of cases.
- Pushing model size above 300B parameters has very diminishing returns to scale

They trained a model which was **better** with “only” 70B params to show this.

Chinchilla

nostalgebraist:

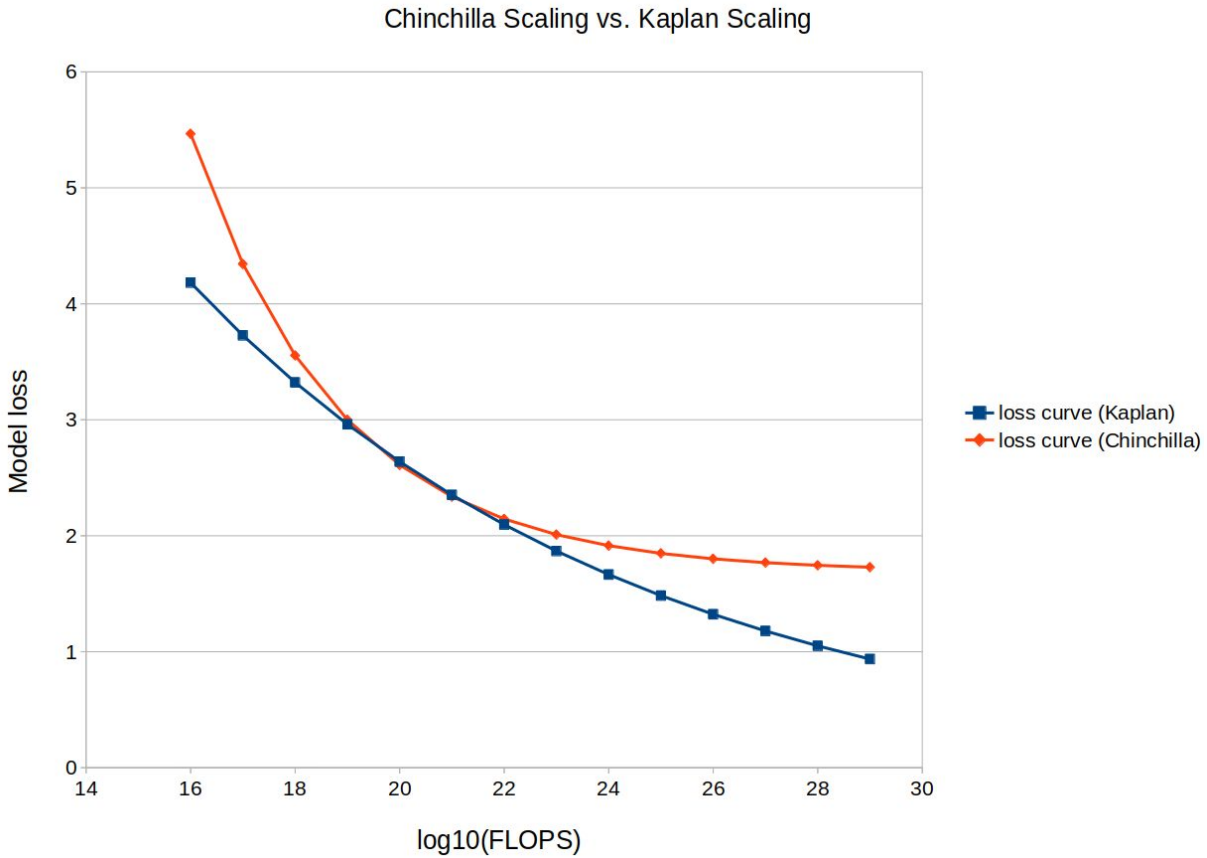
$$L(N, D) = \underbrace{\frac{A}{N^{\alpha}}}_{\text{finite model}} + \underbrace{\frac{B}{D^{\beta}}}_{\text{finite data}} + \underbrace{E}_{\text{irreducible}}$$

Kaplan vs Chinchilla

$$L(N, D) = \left[\left(\frac{N_c}{N} \right)^{\frac{\alpha_N}{\alpha_D}} + \frac{D_c}{D} \right]^{\alpha_D}$$

$$L(N, D) = \underbrace{\frac{A}{N^{\alpha}}}_{\text{finite model}} + \underbrace{\frac{B}{D^{\beta}}}_{\text{finite data}} + \underbrace{E}_{\text{irreducible}}$$

Two scaling laws, two conclusions



Chinchilla

nostalgebraist:

$$L(N, D) = \underbrace{\frac{406.4}{N^{0.34}}}_{\text{finite model}} + \underbrace{\frac{410.7}{D^{0.28}}}_{\text{finite data}} + \underbrace{1.69}_{\text{irreducible}}$$

Chinchilla

Nostalgebraist: If we insert the values for Gopher…

$$L(280 \cdot 10^9,\ 300 \cdot 10^9) = \underbrace{0.052}_{\text{finite model}} + \underbrace{0.251}_{\text{finite data}} + \underbrace{1.69}_{\text{irreducible}} = 1.993$$

Post GPT-3

A succession of papers came out:

- Megatron (describes the computational architecture of GPT-3)
- Gopher (bigger, better GPT-3 variant)
- Chinchilla (smaller, better allocation of resources)
- LLaMa (open source GPT-3, trained for longer)

Post GPT-3

We start to see the conditions emerge for competition:

- Megatron tells people how to train large models
- Chinchilla shows you how to allocate your resources
- Llama gives open source weights

Modern era

Then, the modern era:

- GPT-4
- Claude (Claude Long)
- PaLM 2

We don’t know anything about these architectures, but we know they were *very* expensive.

Speculation on modern architectures

1. Conditional routing
2. Sparse attention
3. Long context length
4. Many, *many* GPUs

Conditional routing

- Transformer variants which allocate different amounts of compute depending on the token
- E.g. Mixture of Experts, which selects different *parameters* for each incoming token (e.g. Switch Transformers)
- Creates a sparsely-activated model with a very large number of parameters, but constant computational cost
- Scales nicely to multiple machines: shards at the first point of contact
- Allows for specialization

Sparse Attention

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Sparse Attention

Softmax:

$$\text{softmax}([760, 752, 750])$$

$$=\text{softmax}([10, 2, 0])$$

$$= [0.99962, 0.00034, 0.00005]$$

Sparse Attention

- Everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it
- H100s are 2x faster with sparsity, but, unclear how to use
- GPT-3 alternates sparse & dense attention layers, in practice, I don’t see others using this
- Difficult to see how anyone is scaling to massive context lengths without sparse attention
- Active subject of research!

Long context lengths

For a long time, context length was 2048, *maybe* 4096, 8192.

Then, two major developments:

1. Flash Attention
2. Lots of investor $$$ (= lots of H100s)

Now: 32k, 100k context length! Naively: 16x-100x more expensive.

Lots of anecdotal reports that Claude 100k doesn't attend well to the tokens in the middle.

Where is research heading?

1. Inference cost (memory consumption, smaller models, latency)
2. Factual accuracy (solve hallucinations)
3. APIs
4. Better RLHF

Inference Cost

1. Attention is expensive
2. But ironically, so are large feed forward networks
3. Just very very expensive to serve requests from a 175B parameter model, especially with multiple GPUs

Factual accuracy

- Models are predicting the next token, no notion of factual accuracy
- If “average advice on Reddit” is useful, they’re great
- APIs/retrieval will solve this
- E.g. I have two degrees in math, but I’m still bad at multiplication- so I use a calculator!

Factual accuracy

More broadly, the “predict the next token” setting means that models find it very difficult to, e.g., learn math:

$$1 + 1 = 2$$

Becomes the sequence:

[16, 489, 220, 16, 284, 220, 17]

Have to learn to predict 17 given[16, 489, 220, 16, 284, 220]!

APIs

We want our models to use tools

Biggest gap between human performance

E.g. humans are bad at math. We use tools!

Unclear how to do this without paying $$$ for raters to build datasets that are immediately out of date.

RLHF

- Most use PPO, which isn't close to a SOTA RL algorithm
- Can't train for long without overfitting the data

Lots of opportunity for RL researchers to contribute!

As a practitioner- this is very fresh. Lots of space!

Questions?

Twitter: @finbarrtimbers

Newsletter: blog.finbarr.ca