Forecasting TAI with biological anchors

Part 1: Overview, conceptual foundations, and runtime computation

Author: Ajeya Cotra
Date: July 2020

This report emerged from discussions with our technical advisors Dario Amodei[1] and Paul Christiano.[2] However, it should not be treated as representative of either of their views; the project eventually broadened considerably, and my conclusions are my own.

This is a work in progress and does not represent Open Philanthropy's institutional view. We are making it public to make it easier to gather feedback, to help inform others' thinking in the effective altruism community, and to allow for follow-on work outside of Open Phil. However, we may edit it substantially in the future as we gather feedback from a broader audience and investigate open questions.[3] Accordingly we have not done an official publication or blog post, and would prefer for now that people not share it widely in a low-bandwidth way (e.g., just posting key graphics on Facebook or Twitter).

The report has been divided into four Google docs to load faster. ***This is Part 1, the second part is here, the third part is here, and the fourth part is here.*** *Additional materials (collected in this folder):*

- *Quantitative model: the Python notebook Biological anchor hypotheses for 2020 training computation requirements; a template spreadsheet When required computation may be affordable; and my best guess, conservative, and aggressive forecasts.*
- *Supplemental materials: a document containing various appendices; a folder of figures for the report; the spreadsheet Extrapolations of data and compute to train models; and the Python notebook Compute price trends, which draws on data in this folder.*

Open Philanthropy (Open Phil) is a grantmaking organization whose mission is to do as much good as possible per dollar. One of our major focus areas is technical research and policy work aimed at reducing potential risks from advanced AI. In this focus area, we aim to anticipate and

1 Note that I think it is somewhat unlikely that a single model would have as large an impact as the Industrial Revolution on its own; I am focusing on this relatively unrealistic path to TAI because it is simpler to analyze, and because I think the estimate of when the amount of computation required to train a transformative model could become affordable may still serve as a decent proxy estimate for TAI timelines even if TAI does not take the form of a single unified model. See here for more detail.

2 A general model which must be fine-tuned for various specific purposes (e.g. a language model pre-trained on a huge corpus of text) would count as a transformative model *if and only if* the fine-tuning is cheap enough and fast enough that it would be economically feasible and desirable to fine-tune the model immediately to automate a wide array of jobs (or a few key jobs sufficient for transformative impact). A salient way this could happen is if fine-tuning the model to do a particular job is no more expensive than teaching an intelligent human being to do that job. If instead the step of "fine-tuning" the model to have a transformative impact requires millions, billions, or trillions of additional data points, that model is likely not transformative by itself. See here for more detail.

3 This particular vision for a transformative model is very similar to the idea of "artificial general intelligence" or AGI. However, not all of the connotations of "AGI" necessarily apply here. On the one hand, some models that could reasonably be considered "general" may not be powerful enough to have a transformative impact (e.g. a personal assistant AI). On the other hand, "AGI" is often defined as a program that can do *anything that any human can do* at least as well as the best human at that task, which seems sufficient but not necessary for transformative impact. See here for more discussion.

influence the development and deployment of *transformative artificial intelligence* (TAI), which we informally define as "software" -- a computer program or collection of computer programs -- that has at least as profound an impact on the world's trajectory as the Industrial Revolution did (see below for a more detailed description of TAI and the relationship between TAI and the concept of "artificial general intelligence" or AGI).

The question of when -- if ever -- software will have had a transformative impact, which we refer to as **TAI timelines**, is crucial for how we should prioritize between potential risks from advanced AI and other focus areas, as well as what interventions we should prioritize within the AI focus area.

One salient path to TAI is to train a large machine learning model[4] that is capable of causing such a dramatic transformation by itself (e.g., by quickly and cheaply automating a very wide array of economically valuable labor simultaneously), which I will call a **transformative model**.[5]In this report, I (Ajeya Cotra) propose a framework for estimating when the amount of computation required to train a transformative model may become affordable, which is one important consideration relevant to estimating the probability that some entity trains a transformative model.

Some basic terms

This section introduces basic terms related to machine learning (ML) and computer hardware; people familiar with ML can skip this section.

Machine learning usually means using a lot of trial and error to create a ***model*** (a computer program) that accomplishes a certain task (such as classifying images). Typically we start with a model that is very bad at the task (e.g. one which takes as input an image and spits out a classification entirely at random) and then iteratively adjust that model using a ***learning algorithm***. First, the model tries classifying an image -- say it guesses that a particular image is a dog. If the image *was* of a dog, the learning algorithm tweaks the model so that it's slightly *more* likely to label similar-looking images as "dogs"; if it *wasn't* a dog, the learning algorithm tweaks the model so it's slightly *less* likely to label similar-looking images as "dogs."

Eventually, after enough rounds of attempting to classify an image and tweaking the model, the model gets to be good at image classification. The process by which an initial model is improved through this kind of trial-and-error is called ***training***. It often takes a lot of examples (also called ***"samples"*** or ***"data points"***) to train a model until it's competent at the relevant task. In the case of an image classification model, it may take millions or tens of millions of images.

Reinforcement learning is a type of machine learning. Rather than attempting a discrete task such as classifying an image and then learning whether the attempt was correct, a

4 Here I mean a single piece of software which can be copied and run by many different people on many different machines, rather than a single *instance* of the program run only on one machine.
5 I am thinking of top1 accuracy, for those familiar with that term.

reinforcement learning model is acting in an environment (e.g. playing StarCraft) and receives intermittent rewards which will be larger (on average) if the model is performing better. Over time, the model is iteratively adjusted using a learning algorithm to take actions that maximize expected reward.

Neural networks are one broad category of machine learning models. Many recent advances in image classification, game playing, and language modeling were accomplished using very large-scale neural networks (also called ***"deep"*** neural networks); they are capable of representing a very broad class of functions. Neural networks are characterized by a large set of numbers called ***parameters***; the learning algorithm tweaks a neural network by changing these parameter values.[6]

Hardware power (how much useful work a computer does per unit time) is commonly measured in terms of ***floating point operations per second*** *or* ***FLOP/s****.* Larger models run on hardware that performs more FLOP/s. In my report, I also use *floating point operations* or ***FLOP*** to measure *total* computation; a "FLOP" is equivalent to one addition, subtraction, multiplication, or division of two decimal numbers. Training a larger model generally takes more total FLOP.

I deal with very large numbers, particularly large numbers of training FLOP, throughout this writeup, so I generally use scientific notation. For example, "1e6" refers to 1,000,000 or 1 million, "1e15" refers to 1,000,000,000,000,000, which is 1×10^{15}, or 1 quadrillion; "1e16" is ten times larger than 1e15, "1e17" is one hundred times larger than 1e15, and so on.

Overview of the framework and estimates

This is a relatively terse outline of the entire argument; it introduces key concepts and figures with only brief explanation, and may be hard to follow for non-technical readers. The main report provides a slower and more detailed explanation.

I first generate a subjective probability distribution over how much computation it would take to train a transformative model if we had to do it imminently, using 2020 ML architectures and algorithms. I call this the **2020 training computation requirements distribution** (see here for a more precise definition and discussion of this concept). I measure training computation in

6 Because the model would run 100 times faster than humans, it may only need to cost << $10,000 per hour.

floating point operations or FLOP.[7] I use this Python notebook to generate the 2020 training computation requirements distribution.

To estimate the probability that the computation to train a transformative model is affordable in future years, I then model three additional considerations: how training computation requirements are likely to fall over time due to **algorithmic progress**, how the amount of computation available for a given price is likely to increase over time due to falling **computation prices**, and how the amount of money an AI project is **willing to spend on computation** to train a potentially transformative model would increase over time.

For each future year Y, the "year Y training computation requirements distribution" can be calculated by modifying the 2020 training computation requirements distribution according to the algorithmic progress forecast to place greater probability on low levels of computation. The amount of total computation that would be available for training a transformative model in year Y can be calculated by multiplying the forecast for how much computation can be purchased per dollar in year Y with the forecast for how many dollars an AI project would be willing to spend on computation in year Y. The probability that the amount of computation required to train a transformative model is affordable in year Y is the probability that the second quantity exceeds the first quantity. This is modeled in this Google Sheet.

The 2020 training computation requirements distribution is the most uncertain of the four key quantities, and most of my research to date has focused on it. In the rest of this overview, I describe:

- How I generate my 2020 training computation requirements distribution (more).
- How I generate forecasts of algorithmic progress, computation prices, and willingness to spend on computation, and use them to estimate when the amount of computation required to train a transformative model may be affordable (more).
- How I think about the translation from "when the amount of computation required to train a transformative model is affordable" to "when TAI may be developed" (more).

7 In some cases, the cheapest currently-discoverable solution to a task may require physically unattainable amounts of resources, e.g. a computer the size of Jupiter. In that case, the "price" is underdefined -- I am generally imagining a simple operationalization such as "multiply the current unit price of some resource by the amount required." So for example, if solving a certain task would require more computational power than all of the computers on Earth could provide, we could estimate the current "cost per operation" and multiply it by the number of operations required to solve the task. If solving a task would require astronomical amounts of labeled images, we could imagine how much we would have to pay a human to generate and label one image, and multiply by the number of images. This works because the key point of introducing this concept is to have a unified number representing "difficulty" that goes down over time as algorithms improve, even before it crosses over from "unattainable" to "attainable."

2020 training computation requirements distribution

To estimate training computation requirements, I focus on one particularly salient vision of a transformative model: a model that can perform a large majority of economically valuable jobs more cheaply than human workers can.[8]

I see the human brain as an "existence proof" found in nature for this type of transformative model, so I use evidence from biology to help estimate the computation needed to train it. Additionally, while some possible transformative models may only automate a relatively narrow set of highly valuable jobs, I think the computation required to train a relatively human-like model is a reasonable proxy for the computation required to train any kind of transformative model (see here for more discussion of this).

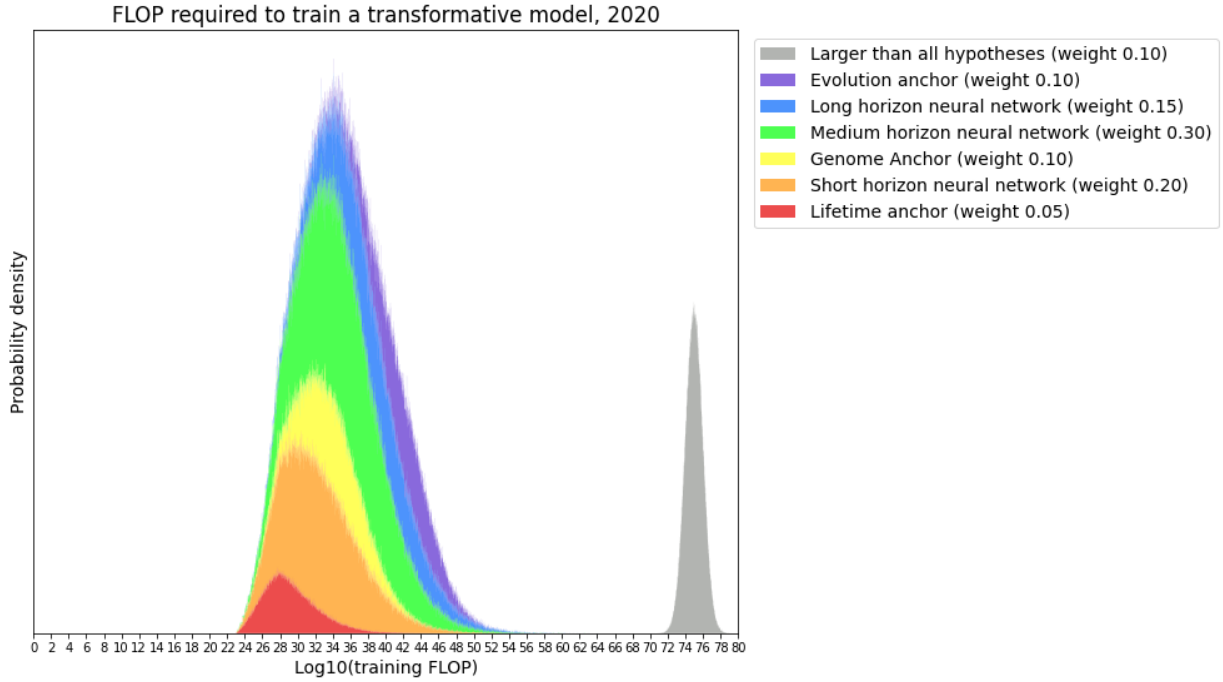
This leads me to use a **biological anchors framework** to estimate training computation requirements for a transformative model:

- I first lay out four hypotheses about 2020 training computation requirements, each of which anchors on a key quantity estimated from biology: total computation done over evolution, total computation done over a human lifetime, the computational power of the human brain, and the amount of information in the human genome (more).
- Then, I associate each biological anchor hypothesis with a conditional probability distribution representing what 2020 training computation requirements would be if that hypothesis were entirely correct (more). The *Lifetime Anchor* hypothesis estimates that a median of ~1e29 FLOP[9] will be required by anchoring on the amount of FLOP done in a human lifetime. The *Evolution Anchor* hypothesis estimates a median of ~1e41 FLOP will be required by anchoring on the amount of FLOP done over the course of evolution. The *Neural Network* hypothesis extrapolates (from existing models) the amount of computation required to train a neural network whose size is anchored to the human brain, estimating somewhere between ~1e32 FLOP and ~1e37 FLOP depending on how long the model would need to run to process one "data point." The *Genome Anchor* hypothesis uses a similar extrapolation but anchors to the number of bytes in the human genome to estimate parameter count, resulting in a median of ~1e33 FLOP.
- Some hypotheses predict that we should have already been able to afford to train a transformative model with reasonable probability; I think this is unlikely, so I execute an update against low levels of FLOP (more).
- Finally, I assign probabilities to the different hypotheses to produce a mixture distribution representing my overall belief about 2020 training computation requirements (more).

8 The setup of reinforcement learning is slightly different, with no explicitly defined "data distribution"; see Part 2 for a brief description.

9 This is an informal definition. Pinning it down would require more clarity and specificity on the type and size of the perturbation in the model being evaluated as well as some constant confidence level which we are seeking re: whether the perturbation improves or worsens performance. Also note that the effective horizon length should increase over the course of training -- as the model gets closer to optimal, there is less room for improvement and it takes more data to be able to tell which of two similar models is very slightly higher-performing. I try to get around this complexity by assuming that the "effective horizon length" of the language modeling problem is 1 token, and thinking about the effective horizon lengths of other ML problems in relation to this. See here for more discussion.

In the Python notebook, I follow the above steps to generate the distribution shown below:



Articulating biological anchor hypotheses

I consider four high-level biological anchor hypotheses for the amount of computation that would be required to train a transformative model using 2020 architectures and algorithms. All of them rely (directly or indirectly) on an estimate of "the amount of computation performed by the human brain," measured in floating point operations per second (FLOP/s).

The question of whether there is a sensible notion of "brain computation" that can be measured in FLOP/s -- and if so, what range of numerical estimates for brain FLOP/s would be reasonable -- is conceptually fraught and empirically murky. I lean heavily on a detailed investigation into this topic conducted by my colleague Joe Carlsmith. For the purposes of this report, I use the following definition of "brain FLOP/s":

> *Suppose we redo evolutionary history at the point when neurons first emerged, but in every animal we replace each neuron with N floating-point units that each perform 1 FLOP per second. For what value of N do we still get roughly human-level intelligence over a similar evolutionary timescale?*

Under this definition, my median estimate for human brain computation is ~1e15 FLOP/s. (See here for more detail on this definition and my subjective distribution over brain FLOP/s.)

The first two "biological anchor" hypotheses estimate the total FLOP that would be required to train a transformative model by anchoring to the total FLOP done over the course of a naturally-occuring "training" process:

- **Evolution Anchor:** This hypothesis states that before examining detailed evidence, we should assume on priors that training computation requirements will resemble the amount of computation done over the course of evolution from the earliest animals with neurons to modern humans, because we should expect our architectures and optimization algorithms to be about as efficient as natural selection. This hypothesis anchors to evolution computation (~1e41 FLOP), and adjusts by a relatively modest factor to account for qualitative considerations about how sophisticated our architectures and algorithms seem to be as of 2020.
- **Lifetime Anchor:** This hypothesis states that we should assume on priors that training computation requirements will resemble the amount of computation done by a child's brain over the course of growing to be an adult, because we should expect our architectures and optimization algorithms to be about as efficient as human learning. This hypothesis anchors to lifetime computation (~1e24 FLOP), and adjusts from this anchor by a relatively modest constant factor to account for qualitative considerations about how sophisticated our architectures and algorithms seem to be as of 2020.

The final two hypotheses use a biological anchor to estimate the *size* of a transformative model, rather than to directly estimate training FLOP. Both hypotheses state that we should anchor to human brain FLOP/s to estimate the **FLOP per *subjective* second** performed by a transformative model. By "subjective second", I mean *the amount of elapsed real time that a model would take to process as much data (e.g. words, sounds, images) in serial as a typical human can process in one second.* For example, a typical human reads about 3-4 words per second for non-technical material, so "one subjective second" for a language model would correspond to however much time that the model takes to process about ~3-4 words of data. If it runs on 1000 times as many FLOP/s as the human brain, but also processes 3000-4000 words per second, it would be performing about as many FLOP per *subjective* second as a human. Throughout, **I will abbreviate "FLOP per subjective second" as "FLOP / subj sec."**

The final two hypotheses differ only in how they generate an estimate for the number of *parameters* required to characterize a transformative model:

- **Neural Network:** This hypothesis states that we should assume on priors that a transformative model would perform roughly as many FLOP / subj sec as the human brain (~1e15 FLOP/s) and *have about as many parameters as we would expect if we simply scaled up the architectures of the largest current neural networks to run on that many FLOP / subj sec*. It adjusts from the anchor point of human brain FLOP/s by a relatively modest constant factor to account for qualitative considerations about how sophisticated our architectures seem to be as of 2020, and estimates parameter count by assuming that a transformative model would have a similar ratio of computation to parameters as the most expensive neural networks we have trained so far. It then extrapolates the amount of FLOP required to train such a model using an empirically-derived scaling law that expresses training data as a function of parameter count.
- **Genome Anchor:** This hypothesis states that we should assume on priors that a transformative model would run on roughly as many FLOP / subj sec as the human brain *and have about as many parameters as there are bytes in the human genome (~7.5e8*

bytes). This hypothesis adjusts from the anchor points of human brain FLOP/s and human genome parameters by a relatively modest constant factor to account for qualitative considerations about how sophisticated our architectures seem to be as of 2020, and uses the same extrapolation as the Neural Network hypothesis to estimate training FLOP.

Generating conditional training computation requirements distributions

For each biological anchor hypothesis, I generate a probability distribution representing what that hypothesis would predict about the amount of FLOP sufficient to train a transformative model in 2020. Below I cover:

- How I generate Lifetime Anchor and Evolution Anchor distributions, and a brief summary of my quantitative estimates for each (more).
- How I generate distributions for the Neural Network and Genome Anchor hypotheses, and a brief summary of my quantitative estimates for these (more).

For the Lifetime Anchor and Evolution Anchor hypotheses

To estimate the 2020 training computation requirements for the Evolution Anchor hypothesis and the Lifetime Anchor hypothesis, I multiply two factors:

- The **anchor distribution**, a probability distribution over the relevant biological anchor -- i.e., how much computation was performed over the course of evolution or over the course of a human lifetime, respectively.
- The **adjustment distribution**, a probability distribution that describes how many times larger or smaller 2020 training computation requirements could be compared to evolution computation or lifetime computation, respectively.

In the case of the **Lifetime Anchor hypothesis**, I took the anchor distribution to be the number of total FLOP that a human brain performs in its first 1 billion seconds (i.e. up to age ~32); my median estimate is (1e15 FLOP/s) * (1e9 seconds) = 1e24 FLOP. I chose the adjustment distribution to have a median of ~3 OOM, because a) models we currently train already require ~3 OOM more data than humans, b) “manufacturing costs” of natural artifacts tend to be ~3-5 OOM lower than manufacturing costs of human goods, and c) human babies are likely born with various priors that we must instead train. This distribution is right-skewed, resulting in a **median of ~3e27 FLOP**, slightly more than 3 OOM larger than the anchor; this is then adjusted upward to ~1e29 after updating against low-end FLOP. See here for more detail.

In the case of the **Evolution Anchor hypothesis**, I estimated the anchor distribution to be ~1e41 FLOP, by assuming about 1 billion years of evolution from the earliest neurons and multiplying by the average population size and average brain FLOP/s of our evolutionary ancestors. I did not shift away from this anchor, resulting in a distribution **centered around ~1e41 FLOP.** See here for more detail.

For the Genome Anchor and Neural Network hypotheses

Generating distributions conditional on the Neural Network and Genome Anchor hypotheses is more complicated, because the relevant biological anchors don't directly refer to the total FLOP required to carry out an optimization or learning process that occurs in nature. Instead, we must estimate and multiply two different factors:

1. A distribution over the number of FLOP that a transformative model would perform per subjective second of data, F, according to the hypothesis.
2. A distribution over the total amount of data required to train the model, T, expressed in subjective seconds (a unit of data corresponding to the amount of information that a typical human can process in one second).

That is, the total training FLOP is given by:

Train FLOP = (F FLOP / subj sec) x (T subj sec of training)

The FLOP per subjective second, F, is simply the product of an anchor distribution and an adjustment distribution, as above: both hypotheses anchor to human brain FLOP/s. As I mentioned above, my median estimate for human brain FLOP/s is ~1e15 FLOP/s. Based on comparisons between the efficiency of man-made artifacts and natural artifacts (e.g. solar panels vs plants or batteries vs fat cells), I adjusted this estimate upward by 1 OOM for a transformative model, resulting in a distribution **centered around ~1e16 FLOP / subj sec.** See below for more detail.

The subjective seconds of training, T, must be extrapolated using other information.

Extrapolating training data requirements

Evidence from ML theory and experimental data suggests that T depends on two key quantities:

- The **number of parameters,** P, that would be required to characterize a transformative model according to the hypothesis. In the case of the Genome Anchor hypothesis, P is anchored to an estimate of the amount of information contained in the human genome, while in the case of the Neural Network hypothesis, P is chosen so that the ratio of computation to parameters is similar to the ratio in commonly-used neural network architectures such as the transformer. ML theory suggests that T should be roughly linear in P, while experimental evidence finds a power law scaling (that is, $T \propto P^{\alpha}$ for some exponent α, which seems to lie between ~0.4 and ~1.2 for different ML problems).
- How much data the model must process (on average) to tell with a given level of confidence whether a perturbation to the model improves performance or worsens performance.[10] I call this the **"effective horizon length"**, measured in subjective

10 Note that this is a measure of *total computation* used in the course of training the model, not computational *power* or *throughput* (which is measured by "floating point operations per second" or FLOP/s). I have seen PFLOP/s-days used as an alternative measure of total computation -- see this post from OpenAI for an example. I will use FLOP throughout this report because it makes arithmetic simpler. I am making no particular assumptions about how much elapsed real time it may take to perform a certain number of total FLOP.

seconds. Effective horizon length can vary by orders of magnitude across different ML problems. There are many potential sources of variation in the amount of data it takes to tell whether a perturbation improves or worsens performance: how sparse "ground truth" loss signals are and how noisy and/or biased proxy loss signals are as a reflection of ground truth, how much subjective time it takes for the consequences of a single action to fully play out, how stochastic the consequences of actions are, whether there are categories of inputs which are very rare but very important to performance, etc. Reinforcement learning problems tend to have longer effective horizon lengths than supervised learning or generative modeling problems, but there can be substantial variation within each broad category as well.

Prima facie, I would expect that if we modify an ML problem so that effective horizon length is doubled (i.e, it takes twice as much data on average to reach a certain level of confidence about whether a perturbation to the model improved performance),[11] the total training data required to train a model would also double. That is, **I would expect training data requirements to scale linearly with effective horizon length** as I have defined it. Using this as an assumption, I break up the two factors described above into three factors:

1. A distribution over the number of FLOP / subj sec of a transformative model, F, according to the hypothesis.
2. An distribution over the subjective seconds T required to train the model, broken into:
    a. An estimate of the effective horizon length, H, in subjective seconds according to the hypothesis.
    b. An extrapolation of the number of "samples" (chunks of H subjective seconds of data) that would be required to train the model, D, as a function of parameter count. I model D as a power law function of parameter count, $K\,P^{\alpha}$, where the constant factor K and exponent α are derived from experimental and observational evidence about existing models.

I therefore model total training FLOP requirements according to the Genome Anchor and Neural Network hypotheses as:

$$\text{Train FLOP} = (F \text{ FLOP / subj sec}) \times (H \text{ subj sec / sample}) \times (D \text{ samples})$$
$$= (F \text{ FLOP / subj sec}) \times (H \text{ subj sec / sample}) \times (K\,P^{\alpha} \text{ samples})$$

I am not confident that linear scaling is the appropriate functional form, or that this concept as defined is precisely what I'm looking for. I could imagine coming to believe that there should be a different relationship between "effective horizon length" as defined and "total training data requirements" (at least for the sorts of ML problems that matter for this forecast), or that another concept or definition is more suited for the role that "effective horizon length" plays

11 Note that I begin this forecast in 2025 rather than 2021. This is because I expect there will be a short-lived period of extremely rapid scaleup in spending on computation for ML training runs over the next few years, and that growth in spending over the subsequent few decades will likely be substantially slower (see this appendix for more detail). Beginning the forecast after I expect this spurt in spending to have ended helps to simplify the modeling.

in this argument. **This is currently the most important thread of further research we are pursuing.** For the rest of this report, I will work with the functional form given above.

The Genome Anchor and Neural Network hypotheses differ in their estimate of parameter count, which impacts the number of effective horizons D they predict would be required. For the three Neural Network hypotheses, I have assumed a parameter count of ~3e14, because architectures commonly used to train large models tend to perform 1-100 FLOP per subjective second (more). For the Genome Anchor hypothesis, I assumed a parameter count equivalent to the number of bytes in the human genome, or ~7.5e8 parameters (more).

Estimating the effective horizon length

The effective horizon length H is a huge source of uncertainty, because it cannot be directly extrapolated or calculated from any of the other relevant quantities -- it is a property of the task that the model is trained to solve and the strategy used to train it, rather than the computational power or size of the model. Generating a guess for the effective horizon length necessarily involves some amount of speculation about the details of a potential training strategy for a transformative model.

The **Genome Anchor hypothesis** is implicitly making a claim that training a transformative model will be structurally analogous to natural selection, albeit more computationally efficient -- that it will involve searching for a "genome" with high "fitness" by optimizing over a large number of "generations." Since signals about the fitness level of a particular candidate genome are sparse, this implies that the effective horizon length for the Genome Anchor hypothesis should be relatively long. I assumed a uniform distribution between ~1 subjective year (3e7 subjective seconds) and ~32 subjective years (1e9 subjective seconds). This results in a distribution **centered around ~1e33 FLOP.** See here for more detail.

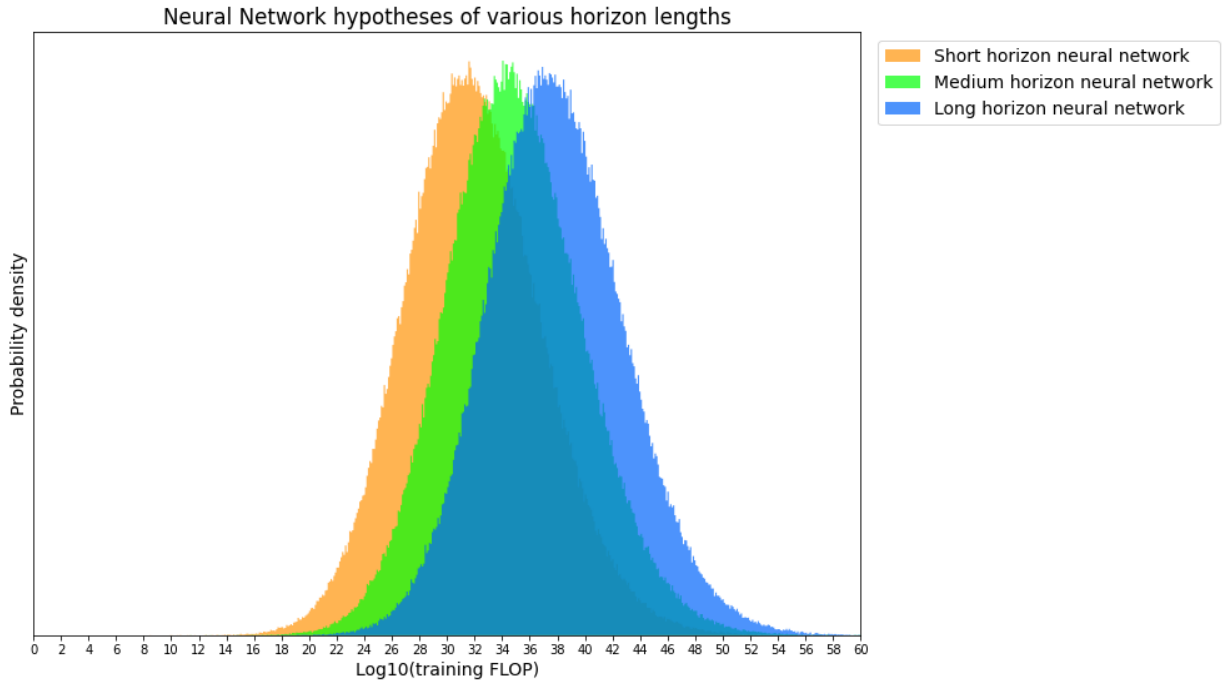
It is more ambiguous what the horizon length should be in the case of the Neural Network hypothesis. At the high end, we can imagine a training strategy structurally identical to the long horizon, natural selection-like training strategy implied by the Genome Anchor hypothesis (but less efficient, as it would require setting more parameters). At the low end, we can imagine a highly dense training signal with an effective horizon length of ~1 subjective second; this is similar to my assumption about the effective horizon length of contemporary language models and shorter than my best guess for the effective horizon lengths of contemporary RL models.

I have arbitrarily divided the space of possible horizon lengths for the Neural Network hypothesis into three buckets which are evenly-sized in log-space:

- "Short horizon": Log-uniform from 1 subjective second to 1e3 subjective seconds (~17 mins), with a median of ~32 subjective seconds. This results in a distribution centered around ~1e32 FLOP.
- "Medium horizon": Log-uniform from 1e3 subjective seconds to 1e6 subjective seconds (~12 days), with a median of ~9 subjective hours. This results in a distribution centered around ~3e34 FLOP.

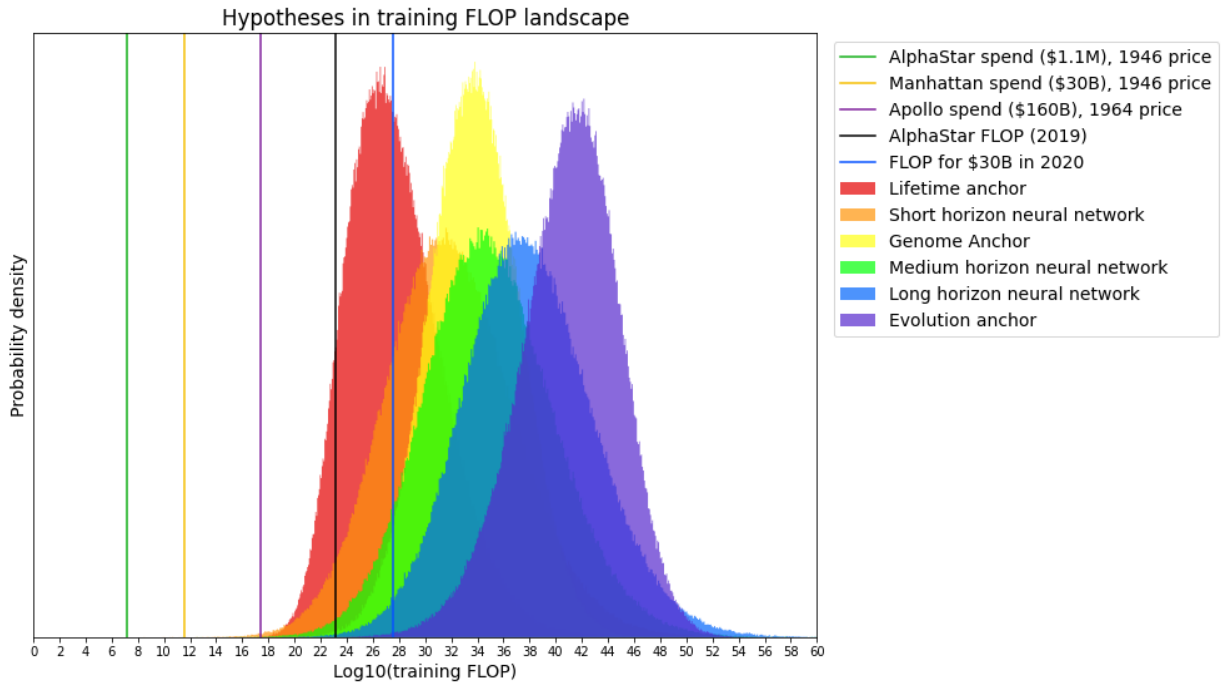
- “Long horizon”: Log-uniform from 1e6 subjective seconds to 1e9 subjective seconds (~32 years), with a median of ~1 subjective year. This results in a distribution centered around ~1e37 FLOP.

These distributions are identically shaped and spaced 3 OOM apart:



Updating against levels of training FLOP that are already attainable

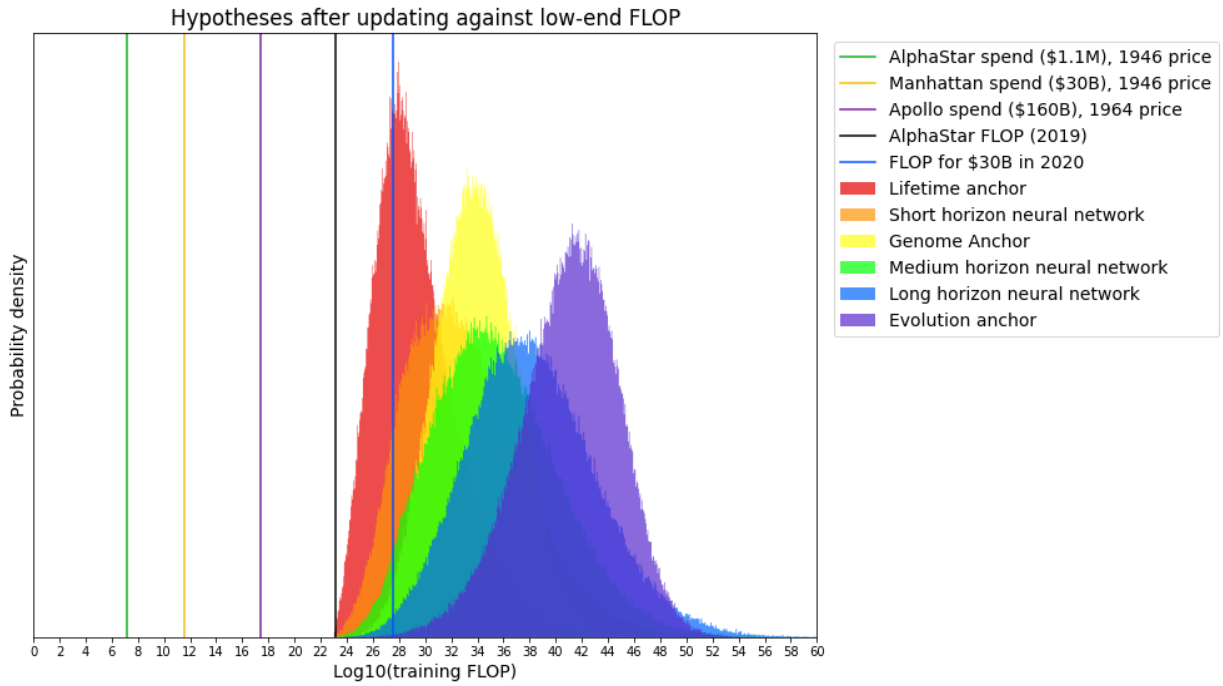
Some hypotheses -- especially the Lifetime Anchor hypothesis -- assign non-trivial probability to levels of computation that are already affordable as of July 2020 (for example, less than the amount of computation required to train AlphaStar or GPT-3):



The lines in this landscape mark levels of computation that could theoretically have been purchased at various points in the past with various levels of investment; see this appendix for details on that landscape. The black line is the amount of computation that the AlphaStar training run used (~1e23 FLOP); the blue line further to the right is the amount of computation that I estimate could be purchased for ~$30B in 2020.

I think it is unlikely that the amount of computation that would be required to train a transformative model lies between the black and blue lines. AlphaStar and GPT-3 were not close to prohibitively expensive for DeepMind and OpenAI. If it were possible to train a transformative model with only a few OOMs more FLOP, I would expect some company to have already trained a transformative model, or at least to have trained models that have had massive economic impact at a scale we have not yet seen from ML models.

To account for this, I truncated each hypothesis distribution:



Here, I have assumed:

- There is overwhelming evidence that the required amount of compute is larger than the amount of compute used to train AlphaStar (~1e23 FLOP, which cost ~$1M in 2019). In other words, I set the probability at 1e23 FLOP and below to 0.
- There is no particular evidence that the required amount of compute is greater than 1e27 FLOP (which would cost ~$10B in 2020). In other words, the probability at 1e27 FLOP and above is identical to the prior probability before truncation.
- There is an intermediate amount of evidence against FLOP values from 1e23 to 1e27. (Specifically, I interpolated log-linearly between those two values.)

To the extent that the prior (untruncated) distribution of a particular hypothesis assigned probability to levels of computation that we have updated against, we should consider that hypothesis less credible overall. The truncation removed[12] ~30% of the probability mass in the Lifetime Anchor distribution and ~10% of the mass from the Short Horizon Neural Network distribution; the others were virtually unaffected, losing <5% probability mass.

Combining into a mixture distribution

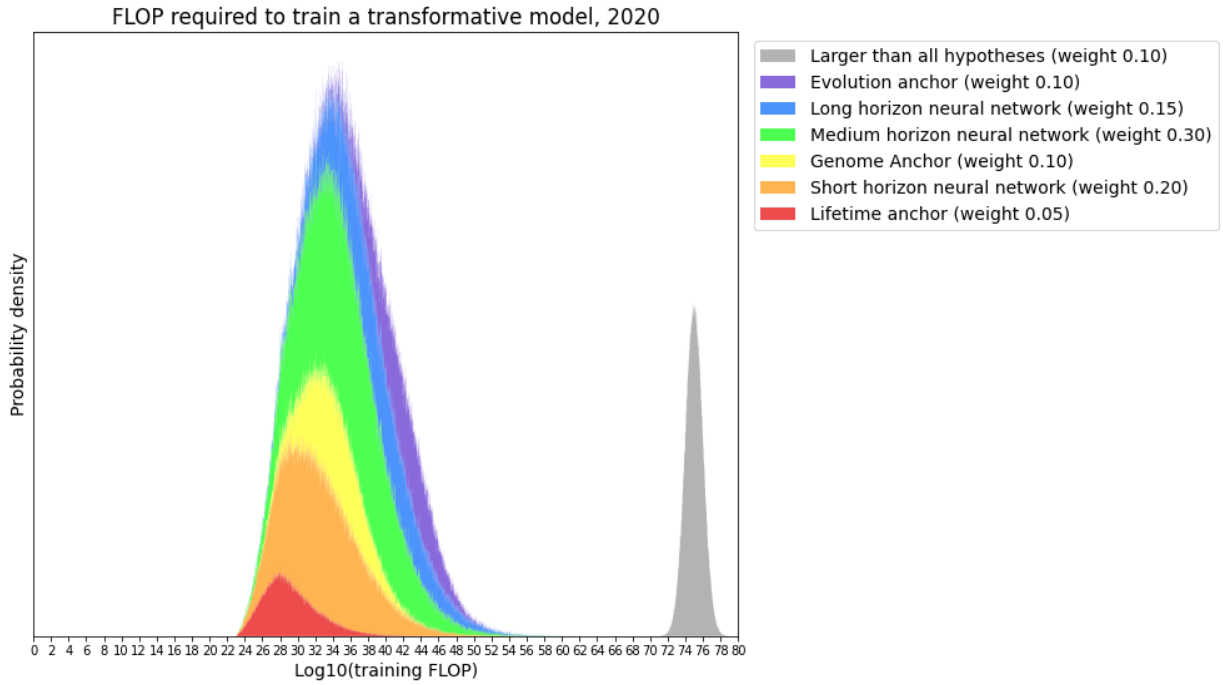
After generating the six conditional distributions for 2020 training computation requirements and updating against low-end FLOP values using the truncation procedure described above, I assign probabilities to the different hypotheses to produce a mixture distribution representing my overall belief about 2020 training computation requirements.

12 As the probability of “larger than all hypotheses” decreases, I increase the probability assigned to each of the six hypotheses proportionally.

I think that the Neural Network hypotheses collectively are the most plausible on priors, because they make the naively appealing assumption that the architecture and learning algorithm that would be used to train a transformative model (if sufficient compute were affordable today) would be broadly similar to the ones used to train the largest models so far. I assign 65% probability to the three hypotheses collectively, but I am deeply uncertain about horizon length for this hypothesis. I tentatively consider "medium" horizon lengths (of several subjective hours) to be most plausible; I assigned 20% to Short Horizon, 30% to Medium Horizon, and 15% to Long Horizon.

Additionally, I assign 5% probability to the Lifetime Anchor hypothesis, 10% probability to the Genome Anchor hypothesis, 10% probability to the Evolution Anchor hypothesis, and 10% probability to the possibility that the amount of computation that would be required to train a transformative model with 2020 architectures and algorithms is higher (perhaps astronomically higher) than any of the hypotheses predict.

The resulting distribution is very wide.[13] Conditional on one of the hypotheses being true, the distribution places non-trivial probability mass on a range of 26 orders of magnitude, from 1e24 FLOP to 1e50 FLOP:



13 Paul inspired the concept of "effective horizon length" and the Short Horizon Neural Network hypothesis, and I use his research into how human-created technological artifacts compare to their natural counterparts as a key input into estimating the FLOP / subjective second that a transformative model may run on. I'm also in a romantic relationship with Paul and extensive discussions with him considerably influenced the report; however, I don't consider this to call for a traditional "conflict of interest" disclosure, because this report is not directly contributing to any funding decisions which affect Paul.

Hardware prices, spending, and algorithmic progress

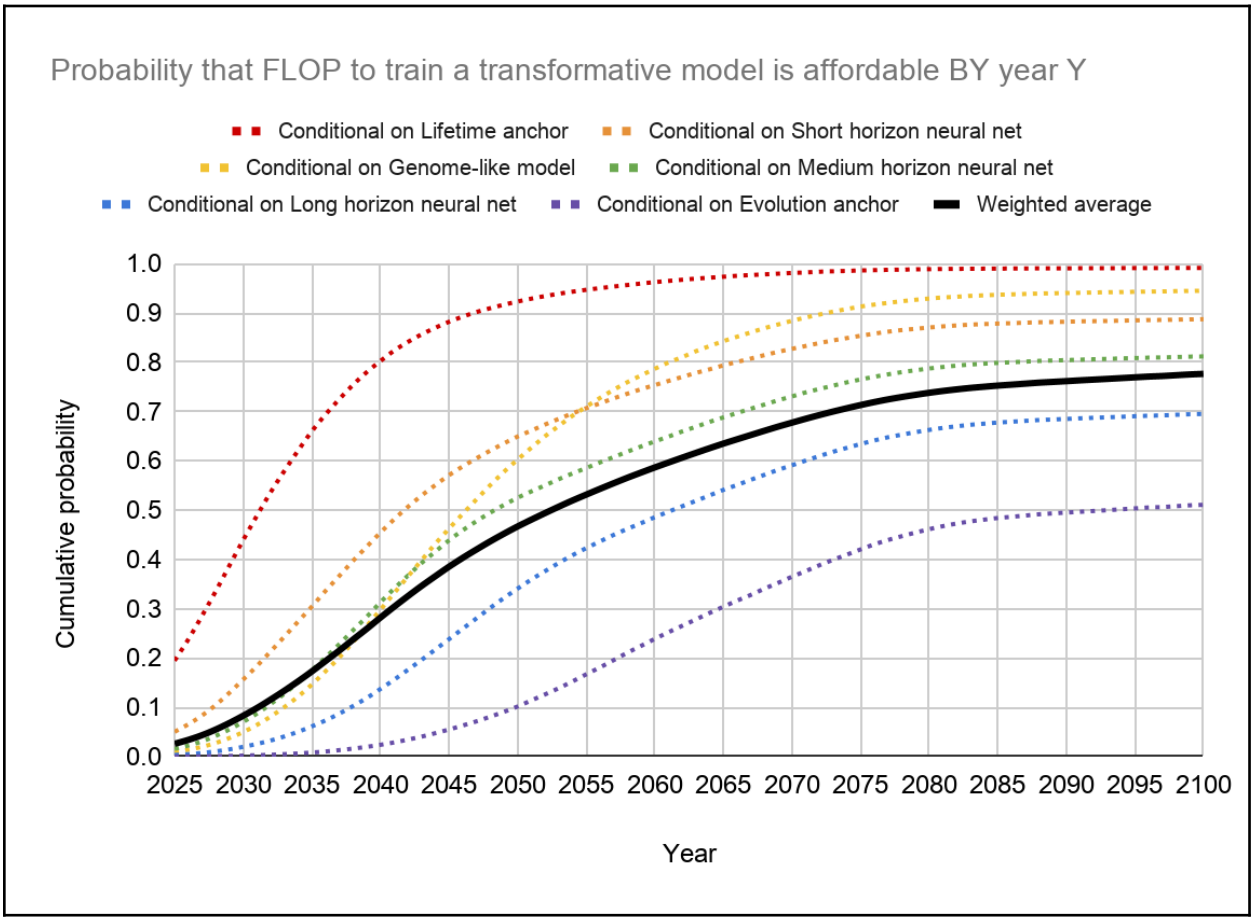
I generate forecasts for effective computation available per dollar, willingness to spend on computation, and algorithmic progress -- and use them to calculate the final probability distribution over when the amount of computation required to train a transformative model will become affordable -- in this spreadsheet. I have spent substantially less time on these quantities than on the 2020 training computation requirements distribution.

I model all three quantities as logistic curves: that is, I assume they are improving at some exponential constant rate right now, but will level off and saturate at some maximum value, after which they will no longer improve. A brief summary of my reasoning:

- **Hardware prices:** I assume that the cost of computation is halving every ~2.5 years (which is somewhat slower than the historical Moore's law trend but roughly in line with the more recent trend), and will level off after about 6 OOM of progress (which is about half as many orders of magnitude of progress as has occurred over the last 50 years).
- **Spending on computation:** As of July 2020, I estimate that the most expensive training run *in a published paper* was the final training run for AlphaStar, at roughly $1M; I expect somewhat more expensive proprietary training runs have been done. In the near-term I expect this to rise rapidly to about $1B by 2025 (a doubling time of about 6 months), then slow to a doubling time of 2 years as AI labs run into more material capital constraints. I assume that spending would saturate at 1% of the GDP of the largest country, by anchoring to national and international megaprojects such as the Manhattan Project and the Apollo Project, each of which were >1% of US GDP. Because the GDP of the largest country is growing (I assume at the long-run US average of 3%), spending does not saturate at a maximum value, growing indefinitely at the same pace as GDP.
- **Algorithmic progress:**
    - I forecast relatively "incremental" algorithmic progress separately for each of the six hypotheses. The blog post AI and Efficiency finds that ImageNet architectures and algorithms improve enough to halve the amount of computation required to achieve AlexNet level performance every 16 months. However, researchers have strong feedback loops on ImageNet, and I would expect them to be less efficient at reducing computation costs for something which has never been done before, such as "training a transformative model." I increase the halving time to ~2-3 years. I assume that hypotheses which predict 2020 training computation requirements are *higher* would also expect more *room to improve* over time, so the cap on progress is different for each hypothesis, ranging from 1 OOM (for Lifetime Anchor) to 5 OOM (for Evolution Anchor).
    - Additionally, I assume that the probability that the required amount of computation is larger than all hypotheses would predict is falling linearly over time, from 10% in 2025 to 3% by 2100,[14] to model the possibility of algorithmic breakthroughs that bring a task from "essentially impossible" to "possible with non-astronomical amounts of computation."

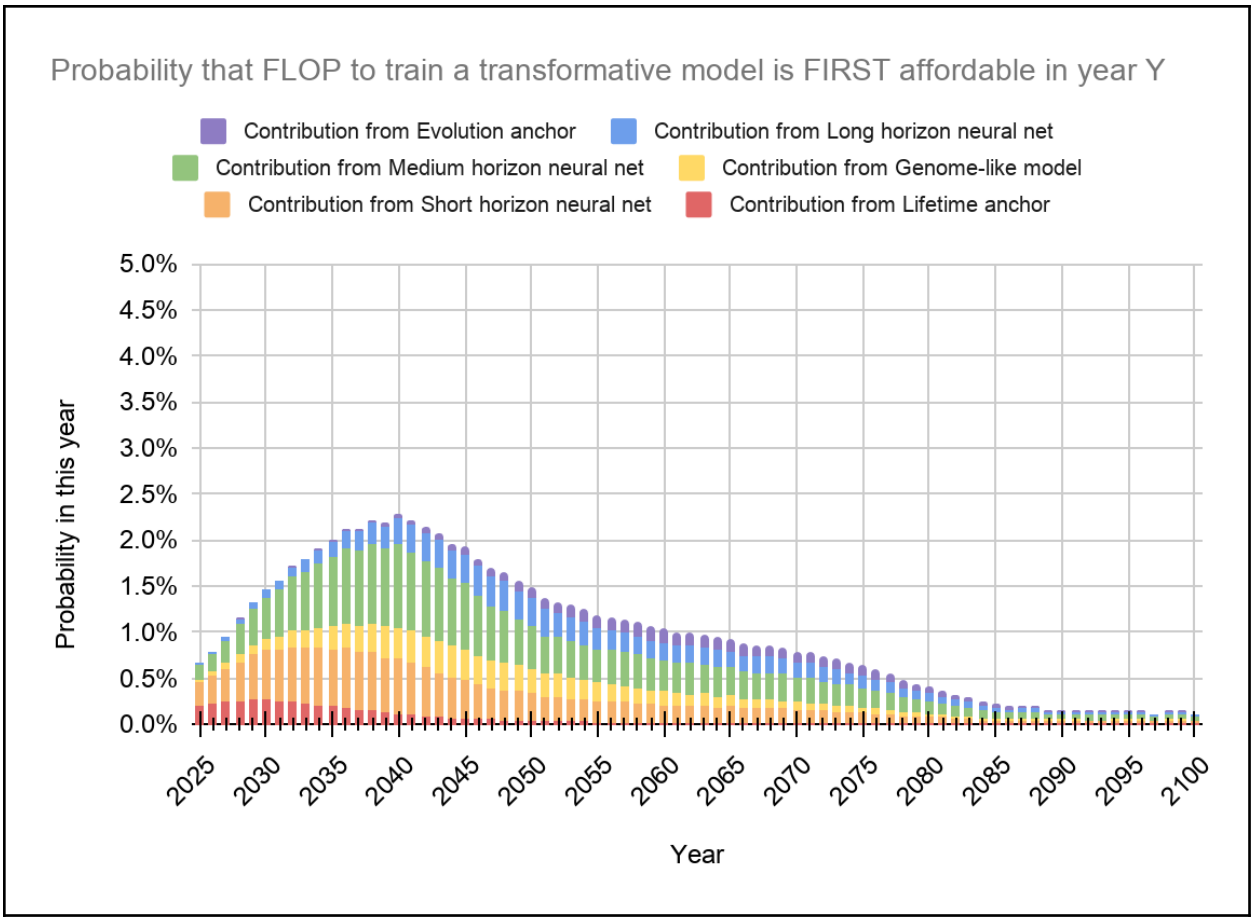
14 In particular, I first encountered the basic concept of attempting to forecast TAI by estimating the amount of computation required to a brain-sized ML model, as well as the Lifetime Anchor and Genome Anchor hypotheses, through Dario's discussion of the topic.

These forecasts, combined with the 2020 training computation requirements distribution shown above, generate the following cumulative distribution estimating when the amount of computation required to train a transformative model will be affordable[15] (bold black line):



Below is the probability that the computation required to train a transformative model *first* becomes affordable in each future year (the probability mass function):

15 As an example of how the report may evolve going forward, most of this report was written before the release of GPT-3, and we haven't thoroughly edited it to incorporate new arguments that have come up in that context (although we have addressed such arguments somewhat). Additionally, the concept of effective horizon length is critical to the argument but we are not fully convinced we have formulated the best version of it or are using it correctly.



The median estimate for when the amount of computation required to train a transformative model will be affordable is 2052 according to this distribution. However, I think that defensible choices of inputs to this model could result in a median as soon as 2036 or as late as 2100 (see this section of Part 4 for aggressive and conservative medians).

Relationship to transformative AI timelines

This model is not directly estimating the probability of transformative AI, but rather the probability that the amount of computation that would be required to train a transformative model using contemporary ML methods would be attainable for some AI project, assuming that algorithmic progress, spending, and compute prices progress along a "business-as-usual" trajectory. The model doesn't incorporate:

- The possibility of other paths to TAI (including more distributed paths in which "TAI" is not one unified piece of technology, but looks more like an Industrial Revolution driven by AI technologies).
- Whether the training data and/or environments to train a transformative model are available by the time the computation is available.
- The possibility of exogenous events halting the normal progress of AI research, such as a global catastrophic risk, severe economic downturn, government regulation of AI, etc.

- Additional effort and testing which might be required to ensure that the model meets safety and robustness standards, regulatory requirements, etc.

How does the probability distribution output by this model relate to TAI timelines? In the very short-term (e.g. 2025), I'd expect this model to overestimate the probability of TAI because it feels especially likely that other elements such as datasets or robustness testing or regulatory compliance will be a bottleneck even if the raw compute is technically affordable, given that a few years is not a lot of time to build up key infrastructure. In the long-term (e.g. 2075), I'd expect it to underestimate the probability of TAI, because it feels especially likely that we would have found an entirely different path to TAI by then.

In the medium-term, e.g. 10-50 years from now, I feel unsure which of these two effects would dominate, so I am inclined to use the output of this model as a rough estimate of TAI timelines within that range. **To choose a round number that avoids signaling too much precision, I have tentatively adopted 2050 as my median estimate for TAI.**

Definitions for key abstractions used in the model

In this section, I define two key concepts that underpin this model:

- The concept of a "transformative model" (more).
- The concept of the "2020 training computation requirements" of a transformative model (more).

How do we define "transformative AI" and "transformative model"?

I define "transformative artificial intelligence" (transformative AI or TAI) as "software" (i.e. a computer program or collection of computer programs) that has at least as profound an impact on the world's trajectory as the Industrial Revolution did. This is adapted from a definition introduced by CEO Holden Karnofsky in a 2016 blog post.

How large is an impact "as profound as the Industrial Revolution"? Roughly speaking, over the course of the Industrial Revolution, the rate of growth in gross world product (GWP) went from about ~0.1% per year before 1700 to ~1% per year after 1850,[16] a tenfold acceleration. By analogy, I think of "transformative AI" as software which causes a tenfold acceleration in the rate of growth of the world economy (assuming that it is used everywhere that it would be economically profitable to use it).[17]

16 It may be the case that some group acquires a monopoly on software that *would* have multiplied the growth rate by ten if it were simply sold on the open market, but the group in question uses its monopoly for other purposes, such as gaining a geopolitical advantage. I suspect that AI technology powerful enough to dramatically accelerate growth would also allow a relatively small group (e.g. a large company or a small nation) with a monopoly on that technology to militarily dominate the rest of the world -- this could potentially serve as an alternative definition of "transformative AI." However, I will use the economic definition of TAI going forward because I think it is more probable.

17 See Christensen et al 2018 for a survey of economists' views on GWP per capita (because these economists tend to expect relatively steady population growth, views about GWP can be backed out from

Currently, the world economy is growing at ~2-3% per year, so TAI must bring the growth rate to 20%-30% per year if used everywhere it would be profitable to use. This means that if TAI is developed in year Y, the entire world economy would *more than double* by year Y + 4. This is a very extreme standard -- even 6% annual growth in GWP is outside the bounds of what most economists consider plausible in this century.[18]

This notion of "transformative AI" is defined solely in terms of its impact on the world, without making reference to any specific properties or capabilities of the relevant pieces of software or how they relate to one another. While this definition best captures what we ultimately care about as philanthropists, it can be difficult to reason about concretely because it is so unconstrained -- it is intuitively unclear what "transformative AI" could look like, whether it's appropriate to think of it as "one technology" (rather than the cumulative effect of many different technologies), and what it would mean to "develop TAI."

In this report, I focus on analyzing a relatively concrete and easy-to-picture way that TAI could manifest: as a *single computer program*[19] which performs a large enough diversity of intellectual labor at a high enough level of performance that it alone can drive a transition similar to the Industrial Revolution. Let's call this a **transformative program**. A transformative *model* is simply one type of transformative program. In this section I spell out what a transformative program is and how it relates to transformative AI in greater detail:

- I first define what I mean by a "task", and give examples of tasks (more).
- I then define the concept of a "transformative task" and provide examples of what a transformative task or model may look like (more).
- Finally, I discuss the relationship between the concept of a "transformative program" and "artificial general intelligence" or AGI (more).

What makes something a well-defined "task"?

Here, a *task* T is a natural language description of the desired input-output behavior of a computer program and the desired cost (in time and/or money) of running the program, that is specific enough that a reasonable person who understands the description could relatively confidently classify whether or not a candidate program meets the desiderata (assuming they could test the program on whatever inputs they wanted for as long as they wanted).

Here are examples of descriptions which are too ambiguous to be "tasks":

- "A computer that can see."

views about GWP). In contrast with the results of this survey, we currently believe that the best model of long-run economic growth actually predicts that a dramatic acceleration in growth over this century is likely, because historical growth has been super-exponential and there have been many past increases in the growth rate; see Senior Advisor David Roodman's post on the topic. We are currently working on a project to explore the disagreements between Roodman's model and economists' beliefs.

18 This is assuming that a highly intelligent human who graduated from a top university could do the course in a few months.

19 This is assuming that a highly intelligent human who graduated from a top university could do this in the course of a few years.

- “A computer that can understand natural language.”
- “AI that can do everything a human can do.”
- “AI that can make some company at least $10 trillion.”

Here are some examples of “tasks” which seem to roughly meet the desired level of precision:

- “A program which costs <$0.01 / hour to run that takes as input images randomly selected from the ImageNet corpus and within 200 ms outputs predicted class labels such that the expected fraction of images it labels correctly[20] on this distribution is greater than the fraction of images the average Mechanical Turk worker would label correctly on the same distribution given the same 200 ms time limit per image, assuming the Turkers all previously had 10 hours to familiarize themselves with the labeled ImageNet corpus.”
- “A program which costs <$0.05 / hour to run and that can take as input *any image* (including e.g. adversarially selected images, image containing words, images depicting famous people or sites, etc) and within 2 min output an English-language description of the image such that the average Mechanical Turk worker would determine that the program’s description of the image was at least as accurate as a description written by a different randomly selected Mechanical Turker who had the same 2 min time limit.”
- “A program which costs <$0.10 / hour to run and that takes as input the first half of a randomly-selected post from Reddit, and within 2 min outputs a prediction of the next half of the post, such that an average Mechanical Turk worker could not distinguish the model’s completion from the true completion with success probability greater than random chance, assuming the worker had not read that specific Reddit post before.”
- “A program which costs <$10 / hour to run and that takes as input the blurb on the back cover of a randomly-selected New York Times Bestselling novel, and within 10 hours outputs a non-plagiarized novel fitting that description, such that if Amazon consumers who made a purchase based on the blurb were randomly assigned to either the program’s novel or the human author’s novel, the average customer would rate the program’s novel at least as highly out of 5 stars as the human author’s novel.”
- “A program which costs <$50 / hour to run and that takes as input a stream of live price data for various stocks and outputs a stream of trades such that a top tier New York hedge fund would increase its expected revenue over the course of the average year by replacing all its human traders with instances of this program.”

There is still some ambiguity in each of the descriptions above, and they could all be tightened further, particularly the last one. For many of them, it would require running (potentially expensive or impractical) real-world experiments involving humans to verify whether a candidate program constitutes a solution to the task. Some desired behaviors are inherently harder to describe succinctly as a “task”, and inherently harder to verify, than others. These examples are intended to be broadly illustrative of the level of specificity I mean when I talk about a “task.”

20 The entire economy likely won’t progress a full 100 times faster at first, because there may be various physical bottlenecks, but I think it is very likely the economy would be progressing ten times faster (especially because much of the virtual professionals’ attention would be focused on alleviating remaining bottlenecks).

What would it take for an individual task to be “transformative”?

A “transformative task” is a specific task such that a solution to that task (i.e. a computer program that displays the desired input/output behavior within the specified resource constraints) would, if used in all contexts that it would be profitable, accelerate GWP growth ten-fold compared to the 2020 growth rate. Such a computer program would be a “transformative program.”

Note that a single instance of the program running on a single computer does not need to have this impact on its own; the program could be copied and run on billions of different computers, with each instance of the program working on a different set of inputs (just as millions of people can simultaneously use different instances of the Microsoft Word program on their personal computers to write different documents).

If transformative AI is developed by training a machine learning model to solve a transformative task, one of the major challenges to its development is likely to be identifying a transformative task that is “ML friendly” -- a task for which we can readily develop efficient architectures, gather or produce abundant training data, craft low-noise and well-shaped reward signals, etc., that has the tremendous scale of impact sought after. I will try to sketch out what a transformative task could look like in this section, although a) these descriptions will not be as crisp as the examples of “tasks” given above, and b) it is unclear whether any of these tasks will turn out to be ML friendly, which is a big source of uncertainty for this whole analysis.

Examples of potentially transformative tasks

The simplest transformative task to think about intuitively is the **“virtual professional”** task. Let us define a virtual professional as a program running on a computer connected to the internet which:

- Takes as input a stream of bits representing the contents of the computer’s monitor.
- Produces as output a stream of mouse clicks and keyboard strokes to interface with the computer.
- Can learn to perform any kind of economically valuable work that could be done purely remotely, using only as much information (words or images) or trial-and-error as a highly intelligent graduate of a top university would require to learn to do that kind of work.
- **Processes inputs and produces outputs 100 times faster than a human**: i.e., “reads” and “writes” hundreds of words per second, when humans generally read and write only a few words per second.
- Is substantially cheaper to run than a highly intelligent human would be to employ (e.g. it costs << $100 / subjective hour[21] to run).

It is relatively easy to think about what a “virtual professional” means. We have strong intuitions about what behaviors the program should be able to display (for example, it should be able to

21 This is assuming that a highly intelligent human who graduated from a top university could do this in the course of 6-7 years, and that any physical experiments that must be conducted or observations that must be recorded do not bottleneck the total time required to get a PhD. For example, conducting longitudinal studies on humans may slow down the serial time required by a lot.

teach itself computer programming in only one day of elapsed real time,[22] it should be able to pass the bar by attending law school classes remotely over only a few weeks of elapsed real time,[23] it should be able to acquire a PhD in less than a month of elapsed real time for many disciplines such as computer science or physics,[24] it should be able to generate reasonable startup ideas given only a few real-time days of dedicated effort, it should be able to offer valuable judgments on complex questions of military strategy or policy after a few real-time months of study, etc). It's also fairly easy to see how a virtual professional would slot into the world economy to have a transformative impact, because we would not need to substantially restructure the economy to extract value from it -- the program could quickly become a direct "drop-in" replacement for a wide variety of top-level professionals who can do their work remotely (such as software engineers, most CEOs, many types of scientific researchers, most government advisors and policymakers, etc).

The availability of a computer program that solves the "virtual professional" task would likely dramatically accelerate economic growth if sold on the open market, simply because a) different instances of this program running on different computers would collectively be able to automate a large fraction of the economy, and b) this means the economy would be operating much faster than it does now because most of its component parts are operating much faster.[25]

By construction, we are positing that a virtual professional would have strengths and weaknesses that roughly match an intelligent and educated human's profile of abilities, most crucially the ability to quickly learn how to perform new types of jobs. However, there is a much larger space of possible programs that -- like existing ML models -- are vastly superhuman in

22 It may be the case that a large expensive fine-tuning run on just one particularly crucial downstream task (e.g. ML research itself) would create a feedback loop that allows the economy to double in a few years, but this seems unlikely to me.

23 By 2025, I think we could easily have decent AI personal assistants that autonomously compose emails and handle scheduling and routine shopping based on users' desires, great AI copy-editors who help a lot with grammar and phrasing and even a little bit with argument construction or correctness, AIs who summarize bodies of information like conversations in meetings or textbooks and novels, AI customer service reps and telemarketers, great AI medical diagnosis, okay AI counseling/therapy, AI coding assistants who write simple code for humans and catch lots of subtle bugs not catchable by compilers / type checkers, even better AI assisted search that feels like asking a human research assistant to find relevant information for you, pretty good AI tutors, AIs that handle elements of logistics and route planning, AIs that increasingly handle short-timescale trading at hedge funds, AIs that help find good hyperparameter settings and temperature settings for training and sampling from other AIs, and so on. However, I don't think this will count as transformative: I don't think the world economy in 2029 will be double what it will be in 2025.

24 There's a common argument that the gap between "AGI" in the first sense and the second sense should be extremely short -- e.g. months rather than years. The most common reason given for this is that the gap in evolutionary time between chimps and humans is very short on evolutionary timescales, and an "uploaded chimp" would not be transformative while an "uploaded human" would be. I don't find this compelling for two main reasons. First, humans (unlike evolution) will be optimizing directly for economic value, and I expect that models well short of human brain size will have some economic value. Second, for a fixed model size I expect it to take less computation to solve tasks with shorter effective horizons, and I expect passing a conversational Turing test to be a shorter horizon task than the easiest transformative task.

25 See this section for a somewhat more technical explanation of how "parameters" work.

some areas and vastly subhuman in others. Let's call these kinds of programs **"unbalanced" programs**.

Some possible unbalanced AI programs would be able to have as profound an impact on the world economy as a virtual professional would have, albeit potentially by *complementing* human skills very effectively rather than serving as straightforward *substitutes* for certain types of human labor. For example, it seems plausible to me that a **virtual scientist** specializing only in physics and chemistry R&D might have a transformative impact by quickly discovering and developing other technologies that then have a transformative impact (e.g. drastically cheaper energy sources, atomically precise manufacturing, or whole brain emulation). Additionally a **virtual AI researcher** model which specializes in conducting ML research could potentially quickly develop another type of transformative AI.

Most AI systems (and other inventions more broadly) have historically been highly unbalanced, so it seems fairly likely that this pattern will continue and that the first "transformative task" we solve will be unbalanced. However, it is harder to generate specific descriptions of unbalanced tasks that would likely have a transformative impact if solved, because it is harder to intuitively understand how any given unbalanced program would interface with the rest of the world to generate impact.

What about a model which is pre-trained on text and fine-tuned on many tasks?

As of July 2020, training large generic language models such as GPT-3 on a broad corpus of text and then **fine-tuning** those models to solve specific downstream tasks seems to be a salient direction for future AI progress. Here, "fine-tuning" means that developers use the parameter values at the end of the first training run -- which were selected to accurately predict what would come next in a given piece of text -- as the starting values for a new training run, where the model is trained to solve a task that requires good language understanding, such as text summarization.

If transformative AI takes the form of a large language model fine-tuned to solve a variety of tasks, would this count as a single "transformative model" or a large number of different models? Each instance of the model would be fine-tuned for a different task and would therefore have a different set of parameter values. However, while the fine-tuned models would not be identical copies of the same computer program, **the generic language model could be a "transformative model" if the fine-tuning step is sufficiently cheap.**

Suppose that fine-tuning the generic model to be competent at any particular job is much faster in serial time and less expensive per subjective second than teaching an intelligent and educated human how to do that job, and the training process is fairly trivial to execute in terms of engineering (e.g. the model may ask its human overseers for the information it needs to make parameter updates in plain English). In that case, the **generic model together with the learning algorithm used to update weights is solving the virtual professional task** and would constitute a transformative model -- I would expect that the economy would be growing radically faster within a few years of training the generic language model. (Similarly, a sped-up

simulation of a human would be a transformative program, even though if we created a billion copies of that simulated human to perform a billion different jobs, the contents of each copy's "brain" would quickly diverge.)

However, if it would take thousands of subjective years to "fine-tune" the generic model to do each new economically valuable job, and setting up the fine-tuning process is often bottlenecked on the time of human engineers and ML researchers, it would most likely[26] not count as a single transformative model. Gradual automation through this process may eventually culminate in the world economy growing at 20-30% per year, but simply training the language model would probably not allow the economy to reach that level of growth right away.

If you have the view that it would be cheap enough and simple enough to fine-tune a large generic language model that the generic model would constitute a "transformative model", the Short Horizon Neural Network hypothesis may represent your view well (more).

How does this concept relate to "artificial general intelligence"?

While "transformative" is defined in terms of its impact on the world, "artificial general intelligence" or "AGI" is usually defined in terms of a certain set of cognitive abilities. I think some uses of the word AGI are very similar to what I call a "transformative program", while others are too strong or too weak.

Sometimes "AGI" is used to refer to a computer program that would pass a relatively "casual" Turing test -- e.g., having a one-hour conversation with the program should feel like talking to a human. This would likely mean it would have to be fluent in natural language, displays a reasonable amount of general knowledge about the world, its responses display "common sense" and understanding of context, its statements are coherent and consistent with one another, it can make simple inferences, and so on. This definition is too weak for transformative AI, and I expect we will have systems that pass this bar many years before AI systems have driven ~20-30% annual growth.[27]

Other times, "AGI" is used to refer to a computer program that can do *any cognitive task* that *any* human can do at least as well as the best human at that task.[28] This definition is somewhat similar to the "virtual professional" concept, but stronger. I think it is likely that we will have transformative AI in advance of "AGI" in this sense, although I am unsure how far in advance.

26 This median value is the median after updating against low-end FLOP; before the update, it was ~3e27 FLOP.

27 For this exercise to have utility, the compute budgets need to be small enough that training the models is affordable today -- this could mean that we would need to use models substantially smaller than the size that would be required to actually have a transformative impact. Note that for any given ML problem, there is likely to be a specific range of model sizes that are "reasonable" to use on that problem, and scaling behavior might not be very smooth or predictable outside that range.

28 This is sometimes called wetware.

How do we define “2020 training computation requirements”?

This section provides a bit more detail on how I am operationalizing the “2020 training computation requirements” -- i.e. “the amount of computation it would take to train a transformative model if we had to do it using only current knowledge.” I will:

- Define the concept of “technical difficulty”, the amount of money it would take to solve a task in a given year (more).
- Explain why I believe that “technical difficulty” of solving a task with ML is a rough upper bound on technical difficulty overall (more).
- Operationalize “training computation requirements” and explain why they seem like a recent proxy for the technical difficulty of solving a task with ML (more).
- Discuss how to define “training data requirements”, which are required for estimating training computation requirements for the Genome Anchor and Neural Network hypotheses (more).

Defining the “technical difficulty of task T in year Y”

I will use *technical difficulty* to refer to the notion of how “hard” it would be to produce a computer program to solve a given task. This concept needs to be defined relative to a certain *well-defined task* and a certain *level of algorithmic understanding.* I will define the *technical difficulty of some task T for the CS field in a certain year Y* as follows:

> The price[29] of the bundle of resources that it would take to implement the cheapest solution to T that researchers could have readily come up with by year Y, given the CS field’s understanding of algorithms and techniques at that time.

By “bundle of resources”, I mean the collection of hardware, tools, data, engineering labor, and wall-clock time that the project requires -- anything other than fundamental algorithms developed in the research community. I am imagining a large team of top researchers is focused specifically on designing a solution to the task for a short period of time (e.g. ~2-5 years), although the “implementation” of the solution may take a much longer period of time (for example, the cheapest solution that researchers could come up with within ~2-5 years might involve training a machine learning model for ten years).

It’s tricky to estimate the current (2020) technical difficulty for a task T that hasn’t been solved yet, particularly if solving the task is far enough out of reach that no one is seriously attempting it

29 Potential costs to large brains substantially more severe than the caloric cost may manifest especially at large sizes. E.g., there is a hypothesis that human mothers’ greater risk of complications in childbirth relative to other animals is due to human babies’ large heads, and I have heard of a related hypothesis that human children’s much longer period of dependence on parents relative to other animals is in part due to the need for us to be born essentially “premature” in order to fit through the birth canal (see the Neoteny in humans Wikipedia page for some potentially relevant discussion). Additionally, it seems likely that bird brains have been optimized to pack in a large number of neurons per pound of neural matter relative to mammals in order to be able to afford to have large brains and still fly; this appears to be a large amount of “optimization effort” on the part of natural selection to allow for large brains. Large birds already have neuron counts comparable to small primates, and it’s plausible to me that the importance of maintaining the ability of flight for fitness in birds’ ecological niches is one of the main things preventing their brains from growing even larger.

at all right now. Technical difficulty of tasks which have the form “improve upon the current state-of-the-art score on X popular benchmark by Y%” may be relatively easy to estimate by extrapolating from existing trends, but getting a good estimate of technical difficulty for a task that’s much further out of reach requires speculating in substantial detail how we might produce an extraordinarily impactful system assuming we had access to unlimited resources. Most programmers and researchers are generally not aiming to do this kind of exploratory engineering work -- in fact, they are generally working on the *inverse* problem of designing and solving tasks that are as impressive or useful as possible *given* current resource constraints.

Technical difficulty conditional on ML is a rough upper bound

I think ML can be thought of as an extension to the toolkit of ordinary programming which makes the job of the programmer easier at the expense of costing more computation and/or data.

“Machine learning” is essentially a cluster of techniques for searching over a search space H of candidate computer programs (also known as “a model class” or “a hypothesis space”) to find a particular program within that space which empirically performs well according to a certain loss function $L(h,d)$ on a certain data distribution[30] $d \sim D$. I’ll use **“machine learning problem”** to refer to a specified combination of objective function and data distribution $(L(h,d),D)$. We can construct a well-specified task from a machine learning problem by specifying a *target performance level* -- a value $L^{¿}$ such that if $L(h^{¿},d) \leq L^{¿}$, the program $h^{¿}$ successfully solves the task.

A very common way to set up a model class H for an ML problem is by using some number P of free parameters: a parametrization function takes as input a list of P numbers, and arranges those numbers together into a computer program that takes as input an element from D (e.g. a representation of an image or Go board) and produces an output of the desired type (e.g. a label for the image or a representation of the next move in the game). Given a parametrization function, each candidate program in H is then fully specified by a particular setting of the P parameters, and the optimization process will select a particular set of numbers which represent a high-performing candidate program. If each of the P parameters has b possible values, then the model class H will contain b^P candidate programs.

Defining a parametrized model class to solve a certain task essentially amounts to writing down a program to solve the task while allowing ourselves to leave parts of the program we are uncertain about “blank” and letting the optimization “fill in the blanks” -- the more parameters we can use, the more “blanks” we can leave in the initial solution we write down. In this sense, **ML**

30 My understanding from Joe’s reports of discussions with neuroscientists is that neurons are widely accepted as the fundamental unit of information processing in the brain, with other types of cells serving various supporting functions. Additionally, according to this informal survey done by AI Impacts, it seems that human respondents tend to rate the behavior of large-brained birds as similarly “intelligent” to the behavior of primates with a similar brain size when the behavior is described in a way that leaves the animal ambiguous; since those birds had a similar neuron count to those primates but substantially smaller physical brain sizes, I take that to be a small amount of additional evidence in favor of neuron count being the key “resource” in this analogy.

simply adds the option to "leave something blank" (filling it in via a particular search-and-optimize process) **to the existing repertoire of programming options**.

Because of this, the amount of money it would take to solve a task using "pure ML" at any given point in time is often substantially higher than the actual technical difficulty of that task at that time: ML is essentially the most brute-force option in our programming toolkit, and if we happen to have deeper insight into the task, we should expect to get away with leaving less of the program "blank" and thereby spending fewer resources on optimization.

Training computation requirements are a decent proxy for technical difficulty

Technical difficulty is defined as:

> The price of the bundle of resources that it would take to implement the cheapest solution to T that researchers could have readily come up with by year Y, given the CS field's understanding of algorithms and techniques at that time.

By analogy, we can define the "year Y training computation requirements" for a transformative model:

> The number of FLOP that would be used to train a transformative model in the cheapest training run that researchers could have readily designed by year Y, given the ML field's understanding of architectures and optimization techniques at that time.

I focus on estimating training computation requirements in this report because:

- There seem to be meaningful biological analogues for computation that we can lean on, which is less obviously the case for e.g. training data or engineering costs.
- Currently, training computation costs tend to make up a substantial fraction of the total cost of generating state-of-the-art ML results, and it seems likely to continue that way, making it a reasonably strong proxy for "technical difficulty conditional on ML."
- Compute usage, price, and spending is unusually tractable to track and forecast over time.

With that said, I will occasionally touch on other important aspects of technical difficulty (particularly data and human labor requirements) qualitatively, and these considerations are important in my judgments about how plausible various biological anchor hypotheses are.

How do we define and estimate "training data requirements"?

The Neural Network and Genome Anchor hypotheses involve estimating 2020 training computation requirements by first extrapolating how many subjective seconds[31] of data are required to train a model of a certain size on a transformative task and then multiplying by the number of FLOP performed per subjective second by a transformative model. How do we define

31 Technically speaking, there is also a separate question of how frequently these neurons are capable of firing, which also impacts energy usage, but I will assume here that they fire roughly ~0.1-2 times per second, per Joe's analysis here.

2020 training data requirements by analogy to 2020 technical difficulty and 2020 training computation requirements?

There are a great diversity of possible “transformative tasks” and “transformative models” -- and for any particular transformative task there are a wide variety of ways to set up an ML problem to solve it: how large a model to use and how long to train it for, exactly what kind(s) of data and environments to use, how much data to collect, what precise objective function(s) to use, and the settings of various hyperparameters can all be varied based on the tradeoffs researchers face and the resources that are available to them. The technical difficulty formulation assumes a hypothetical in which researchers design an ML problem and choose hyperparameters to solve a transformative task while minimizing all-things-considered costs (balancing engineering effort against data against computation against wall-clock time) as best as they can, using their algorithmic understanding as of 2020; our definition of “training data requirements” follows from that.

Let’s say that the cheapest way to train a transformative model (that researchers can think of in 2020) involves training a model with $P^{¿}$ parameters to optimize the objective function $L^{¿}$ to a level of target performance $L^{¿}_{T}$, using $T^{¿}$subjective seconds’ worth of data collected or generated in some specified way (e.g. drawn from some specified distribution, or generated as the result of interaction with some specified training environment).

The estimate for $P^{¿}$ is different for the Neural Network hypothesis and the Genome Anchor hypothesis. In the former case, my median is roughly ~3e14 parameters because the most common architectures as of 2020 seem to involve a large number of parameters each doing a small number of FLOP per subjective second of experience (more), and in the latter case my median estimate is ~7.5e8 parameters because I am anchoring to the amount of information found in the human genome (more). In both cases, we would like to estimate $T^{¿}$ given an estimate of $P^{¿}$.

If we already knew what this ML problem would look like (that is, if we could specify $L^{¿}$, the process of data collection or generation, and the model architecture), then we may be able to run the following ML experiment:

- Perform a sequence of training runs with increasing FLOP budgets[32] $C_1, C_2, C_3, ... C_N$ using models of a similar architecture on data collected according to the specification. For each FLOP budget C_i, perform a systematic sweep for the model size P_i and the total amount of training data T_i that results in the lowest loss value for that computation budget. This will associate each model P_i with a particular optimal amount of experience (T_i subjective seconds).
- Fit a function $T_{eff}(P)$ to the collection of pairs $[(P_1, T_1), ... (P_N, T_N)]$ to describe the number of subjective seconds of data that would be required to train a model of a given size in a compute-cost-efficient way on this ML problem.

32 My understanding is that neuroscientists also sometimes use the word “brain architecture” to refer to this concept.

- Use the function $T_{eff}(P)$ to extrapolate the number of subjective seconds of data that a model large enough to have a transformative impact would be trained on, assuming a cost-efficient training run: $T^{¿}=T_{eff}(P^{¿})$, i.e. $T_{eff}(10^{15})$ for the Neural Network hypotheses and $T_{eff}(7.5\times10^{8})$ for the Genome Anchor hypothesis.

I'll refer to $T^{¿}$ as "the amount of data required to train a transformative model", and $T_{eff}(P)$ (the function that could be fit using the results of this hypothetical experiment) as "the scaling behavior" of a transformative ML problem.

As I said above, the scaling behavior for current ML problems seems to have the functional form

$T^{¿}$ = (H subj sec / effective horizon) x ($K\,P^{\alpha}$ effective horizons)

where the constant factor K is in the range of ~10-1000 and the exponent α is in the range of ~0.4 to 1.2. I assume that the scaling behavior of a transformative task would have the same form, and choose subjective probability distributions over values of K and α to be similar to current ML problems. See Part 2 for more detail.

What does "brain FLOP/s" mean and why anchor to the brain?

All six biological anchor hypotheses rely on an estimate of "brain FLOP/s" in some way. The Lifetime Anchor and Evolution Anchor hypotheses use an estimate of brain FLOP/s to calculate the total amount of computation done over a human lifetime and over the course of evolution, respectively. The Neural Network and Genome Anchor hypotheses instead anchor to human brain FLOP/s to estimate the number of FLOP a transformative model may perform per *subjective* second. In this section, I cover:

- Why I view the human brain as an "existence proof" in nature for a transformative model and the motivation for using the brain to help estimate training FLOP (more).
- The "evolutionary hypothetical" operationalization for "human brain FLOP/s" (more).
- My subjective probability distribution over human brain FLOP/s, informed heavily by a report prepared by my colleague Joe Carlsmith (more).

The brain is an "existence proof" of a transformative model

I see the human brain as **an existence proof in nature** that could be used to calibrate our expectations for the FLOP/ subj sec of a transformative ML model and/or the FLOP required to train such a model. I said above that perhaps the easiest transformative model to concretely picture is a **virtual professional,** a model that can do roughly everything economically productive that an intelligent and educated human could do remotely from a computer connected to the internet at a hundred-fold speedup, for costs similar to or lower than the costs of employing such a human. That is, one salient example of a transformative model is a

computer program that displays a large subset of the behaviors of an intelligent human (albeit running more quickly).

As I explained above, there is also a potentially large space of specialized models which would have a transformative impact by being vastly superhuman in some areas and vastly subhuman in others -- call these "**unbalanced" models**, where a virtual professional (or another potential transformative model that has a profile of abilities very similar to a human) would be "**balanced".** Out of all the theoretically possible transformative models (both balanced and unbalanced), the ones that are trained *earliest* will probably be the ones that turn out to be *cheaper and easier* to train. That means that if and when someone first trains a transformative model of *some kind*, that model will likely have been easier to train than a virtual professional -- otherwise, someone else would have been able to train a virtual professional earlier.

That means that in theory, the technical difficulty of training a transformative model should be at *most* the technical difficulty of training a virtual professional. If we accept this argument, that means that training a transformative model should not be harder than training a model that can display a strict subset of the behaviors that an intelligent human can display -- even if that model has a profile of abilities nothing like a human, and is capable of doing many things that are completely beyond any human's ability.

However, in practice, it seems fairly likely to me that a single model which can accelerate growth so dramatically would *either* need to be fairly "generally intelligent" and capable of quickly acquiring a wide variety of skills like the virtual professional, *or* need to be extremely superhuman at a key set of very difficult skills such as technological R&D. I can't easily think of versions of the latter which seem like they should clearly have substantially lower technical difficulty than solving the virtual professional task. Additionally, I think that explicit quantitative estimates like this are more likely to be overly aggressive than overly conservative, even if some parameters are deliberately chosen to be conservative. Given this, I am inclined to **treat the difficulty of training a "balanced" transformative model as a rough proxy for the difficulty of training *any* transformative model.**

How big would a model need to be in order to be able to solve some *balanced* transformative task if trained using roughly current ML techniques? Once trained, such a model would be able to do a large subset of what an intelligent human could do, so it feels natural to ask if we could somehow quantify "brain computation" as a starting point to anchor our estimates around. If we could cast the brain as a "software" which was "written" by the process of human evolution, running on biological "hardware",[33] then it would be doing something very similar to what a balanced transformative model would be doing. If we had a guess for how many "FLOP/s" the "brain's software" could run on, we could start asking questions like "Should we expect our model architectures to be more or less efficient than the brain's software once trained? By how much?"

33 Note that when I talk about the "efficiency" of an architecture, I am referring to how large a model needs to be in order to be able to fit some data distribution to a desired level of accuracy; **I am not referring to how long it takes to train the model** to do that.

The “evolutionary hypothetical” definition of brain FLOP/s

There are a number of possible ways to try to cash out what we mean by “brain computation”, all of which come with some conceptual messiness (see this section of Joe’s report for an overview). The definition of brain computation that feels most appealing to me from the perspective of this argument is something like *“How much computation would an animal’s brain have to run on for natural selection to be able to select that animal to be as intelligent as humans?”* In this section, I will:

- Provide some caveats and clarifications about this definition (more).
- Attempt a more detailed operationalization of the informal statement (more).
- Generate a subjective probability distribution, based on my reading of Joe’s report, over how much computation the human brain performs according to this definition (more).

Important caveats about this attempted definition

- I am not aspiring to the same level of formalism or detail as I am for other operationalizations of concepts given in this document, which themselves have considerable room for interpretation and potential for confusion.
- I find a couple of alternative definitions (which I don’t discuss here because they are covered in Joe’s report) relatively compelling as well; my current guess is that using any of the definitions I find relatively compelling would result in roughly similar *central estimates* for brain computation, but would likely lead to different behavior in the tails; I find this to be a point that “robustifies” my case somewhat.
- This definition is fairly complex and unorthodox, and relatively hard to think about intuitively and/or test empirically; I am not sure that I have fully played out the logical implications in my mind, which might be causing me to make some sort of conceptual error.
- Technical advisor Paul Christiano originally proposed this way of thinking about brain computation; neither he nor I have a background in neuroscience and I have not attempted to talk to neuroscientists about this. To the extent that neuroscientists who talk about “brain computation” have a specific alternative definition of this in mind, this proposal may not line up well with their way of thinking about it; this might make it more hazardous to rely as much as I do on evidence Joe gathered from discussions with neuroscientists.
- I am not confident that my subjective estimates below are adhering strictly to the definition I give here (although as I said above, my guess is that swapping in an alternative definition which I still consider to be fairly attractive would mainly shift around the tails of the probability distribution rather than the bulk).

With all that said, the question *“How much computation would natural selection have needed access to in order to produce beings as intelligent as humans?”* still feels most appealing to me from the perspective of doing work in this argument about TAI timelines.

More detail on how I imagine the "evolutionary hypothetical" working

Human brains can be thought of as the product of a large-scale optimization process -- natural selection optimizing a certain population of creatures for inclusive genetic fitness within a particular ecological niche. In many ecological niches, having a larger brain turned out to be valuable for an animal's genetic fitness due to the increased versatility and sophistication of behaviors that it enabled; human evolutionary history in particular is characterized by a very large increase in the brain sizes of our ancestors sustained over a very long period of time. This is despite the fact that an animal with a larger brain would need to consume more calories and may face other complications,[34] which would (other things being equal) *reduce* fitness in a resource-scarce environment.

For these purposes, I think "brain size" mostly manifests as "neuron count."[35] Because of the dynamic described above, I think it would be reasonable to **model neurons as a "resource" that can be "purchased at some price"**[36] from the perspective of natural selection as an optimizer:

- When larger brains confer substantially more of a fitness advantage on average than the fitness losses (from calorie needs and various complications) within a particular niche, there will be a natural selection pressure toward having brains with more neurons.
- However, there is also natural selection pressure toward "making the most" out of a certain brain size -- e.g., toward arranging a fixed number of neurons in a highly efficient wiring pattern, allowing for the largest possible qualitative improvement in behavioral sophistication per increment of brain size.

This is similar to the sense in which computation is a resource for training ML systems -- an AI project can purchase more compute when this would be worth the cost for its research and/or commercial goals, but researchers also attempt to design architectures that are expressive, scalable, and well-suited to the task of interest in order to leverage each unit of compute effectively. By analogy to model architectures, I will refer to the set of genetic adaptations determining how an animal's neurons are wired together and "put to use" to generate behaviors as the animal's "**brain architecture.**"[37]

34 It's possible that the origin of human-level intelligence was the result of a rare "fluke" rather than representing a typical outcome from recapitulating evolution on Earth many times; I am setting aside that possibility for now. If you believe that either human-level intelligence or some lower level of intelligence was an evolutionary fluke, you can assume that I mean to include whatever conditions created that "fluke" in this hypothetical, such that the baseline probability that intelligent life originates out of this process at all is high.

35 The definition he considers which is most relevant is likely the question of the number of FLOP/s that would be required to implement a "reasonably brain-like" model. On priors, it seems reasonable to believe that if we replaced neurons with FPUs, natural selection would have still found brain architectures that are fairly similar at a macroscopic level (e.g., simply using FPUs to construct simulated "neurons", or using one FPU in the same functional role as multiple neurons).

36 All of the content about neuroscience in this section is taken from Joe's report.

37 Although given that the number of synapses dominate the number of neurons in most animals' brains, it would be possible to use artificial neurons that compute much more complicated nonlinear functions of their inputs than standard reLUs in order to make firing decisions while still allowing the multiply-adds from the weights to dominate total computation. It may be that human designers and neuroscientists are currently unsure what the appropriate non-linearities should be, but that evolution would have found a

With this in mind, a key question that I think should play into an estimate of transformative model FLOP / subj sec for the Genome Anchor and Neural Network anchor hypotheses is something like "**How efficient[38] are human-designed ML model architectures for a given human goal** (in this case, for training a balanced transformative model), **compared to how efficient the evolutionarily-selected human brain architecture was for an analogous "goal"** (in this case, producing an unusually intelligent animal in an ecological niche where intelligence conferred high degrees of genetic fitness)?"

A more specific (though strange) counterfactual that could be used to operationalize the question of "brain architecture efficiency" is something like this: Imagine that we restarted the process of natural selection on Earth from the origin of neurons with small jellyfish, while making the following change: whenever the jellyfish's genome would have instructed its body to produce a biological neuron cell, we instead produce some number *N* of floating point units each capable of doing one calculation per second. Assume that:

- There is some interface that translates data from its normal biological sensory organs into numerical inputs for some set of FPUs, and transfers numerical outputs from another set of FPUs into motor commands for the animal's normal muscles.
- The hypothetical jellyfish's genome has the ability to arrange these FPUs into various circuits and produce more FPUs over evolutionary time about as "flexibly" as the genome of the actual jellyfish could rearrange neurons and add more neurons. Adding N FPUs to the hypothetical nervous is about as "costly" as adding one neuron was in its actual nervous system.
- Everything else about biological and physiology and ecological niches on Earth remains unchanged.

How many FPUs, *N,* would need to be provided per neuron on average for this hypothetical version of natural selection to produce animals that we would regard as roughly equally "intelligent" to the animals of Earth, given similar amounts of time to evolve[39] and similar levels of natural selection pressure toward "intelligence"?

In potentially simpler terms: if we anthropomorphize evolution as a being who is consciously optimizing for the "goal" of creating intelligent creatures and is willing to use whatever physical "component parts" that it has available to do the job, we can ask **how well on average it would be able to work with the component of a "floating point unit" vs the component of a "biological neuron"** to achieve its high-level goal. The number of FLOP/s performed by the brain of some animal X is then *N* times the number of neurons in the brain of animal X.

way to use FPUs to construct the necessary behavior using <100,000 FLOP/neuron/sec.

38 Also, while this isn't strictly logically implied by the evolutionary hypothetical definition of brain computation, I am moved by the fact that humans are training models of ~1e11 FLOP / subj sec (e.g. Google T5) that are clearly well below human-level at economically valuable tasks. Because humans are often able to be roughly "competitive" with evolution (see below), we should be surprised if evolution could have done dramatically better than human engineers.

39 My understanding is that reversible computing is extremely difficult to make work, and as far as we know it requires highly-precise, very cold environments that are highly unlike the brain's environment.

As I mentioned at the top of this section, this is an admittedly unorthodox definition which is very challenging to think about concretely -- nonetheless, I feel that it best captures the motivation behind my use of biological anchors in this document. I am seeing humanity as *competing* with this anthropomorphized version of evolution -- can we manage to design pieces of software that achieve similar goals to the "software written by evolution" while consuming less computation? Or is our design ability poor enough that our software artifacts would need to consume much more computation to do similarly impressive things? Similarly, will the *training* process be as expensive as evolution, or are our optimization algorithms more efficient such that we would need to perform much less computation to find a similarly capable model?

Subjective probability distribution over brain FLOP/s

Joe lays out several possible definitions of brain computation in his report, rather than focusing narrowly on the "evolutionary hypothetical" definition I laid out above.[40] He is not aiming to estimate "brain FLOP/s" directly, instead asking the broad question "What can we learn from the brain about the FLOP/s sufficient, in principle, to carry out the tasks that the brain does?" Nonetheless, his research is highly relevant for informing estimates based on more specific and opinionated definitions such as mine.

Based on reading his report, my current best guess is that the number *N* of FPUs that would be required to replace one neuron is about ~0.01-1,000; it seems about 50% likely that the value is 1 or smaller. A brief summary of the argument for this:[41]

- The average neuron fires about ~0.1-2 times per second; this sends an electrical signal across a synapse to between ~1,000 and ~10,000 neurons that are "downstream" of it. Neurons are theoretically *capable* of firing much more often than that -- roughly speaking, whether or not a given neuron fires at a given time is a function of somehow integrating all the input signals it is currently receiving from all of its "upstream" neurons.
- It seems plausible that a "firing event" could be represented with ~1 FLOP per "downstream" neuron that receives the signal, that the "decision-making" about whether to fire could be represented with <100,000 FLOP per neuron per second, and that these computations dominate other computations which may be necessary to represent. This results in ~1,000-100,000 FLOP/neuron/sec, or in other words ~1-10 FLOP / synapse / sec. This is roughly equivalent to the amount of computation it would take to run a model of the brain in which we had perfect knowledge of the connectome (the pattern of neuronal connections) and simply replaced every biological neuron with a standard artificial neuron.[42]
- It also seems plausible that biological neurons are a *less* helpful building block than artificial neurons. For example, my understanding is that because "firing" is only a binary

40 Joe uses 20 watts (Joules / second); my understanding is that estimates of the brain's energy budget tend to vary from 10-20 watts, and that some of this budget is not going toward "computational" activities.
41 Joe's upper bound assumes ~1 FLOP per bit erasure, whereas I would guess that because a FLOP is a substantially more complicated operation, we should assume hundreds or thousands of bit erasures are equivalent to one FLOP.
42 For example, artificial neural networks were inspired by biological neurons, as the name suggests; my understanding is that convolutional neural networks were inspired by the visual cortex in particular.

signal, biological neurons use cues like firing frequency to communicate what conceptually feels like only one real-valued number such as “How hot is this object?” I also believe that at least some neuroscientists consider it plausible that the *average* firing rate of a patch of biological neurons is the appropriate level of granularity at which to understand the brain (e.g. because it may be using such mechanisms to reduce the impact of various kinds of noise). This suggests it may be possible to replicate the function of e.g. ~10-100 biological neurons using one artificial neuron composed of low-noise FPUs capable of sending real-valued signals.

- Furthermore, only some subset of neurons in the whole brain may be most contributing to our subjective understanding of human “intelligence”, such that we may consider an animal with somewhat fewer neurons than a human equally “intelligent”, because the facilities that it sacrificed were e.g. balance or other types of motor control.

The human brain has 86 billion (8.6e10) neurons which are each connected to 1000-10,000 other neurons via synapses; at 1-10 FLOP / synapse / firing and 0.1-2 firing/second, that would be 8.6e10 * (1e3-1e4) * (1-10) * (0.1-2) = 8.6e12 - 1.7e16 FLOP/s. **My best guess for a median is 1e15 FLOP/s**; I would be quite surprised by values lower than ~1e11 FLOP/s because my understanding of the potential reasons why artificial neurons may be a better building block than biological neurons don’t seem to imply that they are likely to be multiple OOM better.[43]

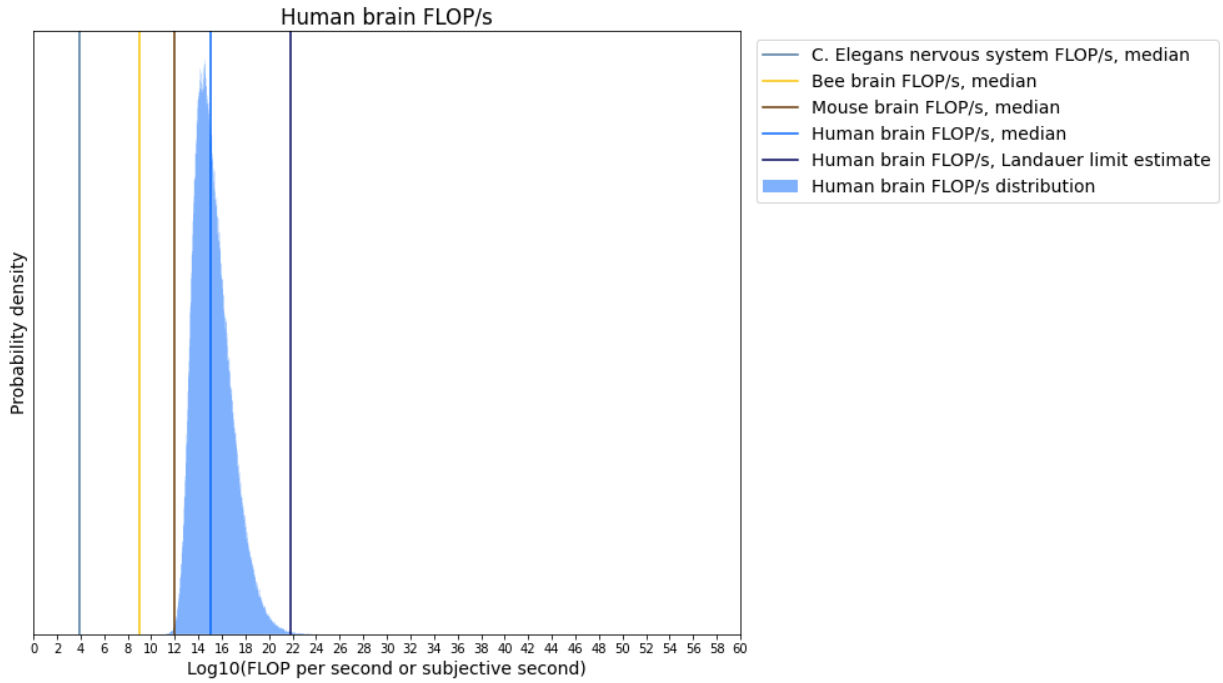
There is a longer tail to the right of my median than to the left given my epistemic state; I find it harder to feel confident in an upper bound for brain computation. My understanding is that there is a wide array of different models of neural computation used by neuroscientists, and many models that are useful for understanding neuron function would result in a full brain computation estimate of >> 1e15 FLOP/s if applied to all 86 billion neurons.

One candidate upper bound that I do find quite compelling is the limit method described in Joe’s report. This is given by the Landauer limit, which is a bound on the number of logical bit erasures that could occur per unit of time in a non-reversible computer that consumes a certain amount of power to run. Given that we can estimate how much power the brain runs on from caloric intake estimates and/or cellular activity measures, the Landauer limit bounds the amount of bit erasures which could be occurring in the human brain -- assuming that it is not doing reversible operations, which seems highly likely.[44] Using what I consider to be a relatively conservative estimate for the brain’s energy budget[45] and a highly conservative estimate for the number of “logical bit erasures” which are equivalent to 1 “FLOP”,[46] Joe estimates that the

43 The Wikipedia list of animals by number of neurons indicates that chimpanzees have ~2.8e10 neurons, compared with humans’ 8.6e10 neurons, about a factor of 3.
44 Ibid; the house mouse is listed as having ~1e12 synapses, compared with humans’ ~1e14.
45 These “cost measures” were chosen to be costs that would be meaningful to both evolution and human designers -- needing to use more energy costs more money and also reduces genetic fitness, other things being equal.
46 Some people have suggested that humans ought to have a comparative advantage over evolution in software relative to physical objects because of the greater ease of iterating and experimenting with software. However, Paul points out that in the case of ML, this may be a sign of comparative *disadvantage* -- most physical devices operate on simple, clean principles which allow humans to plan

Landauer limit for human brain computation is ~7e21 FLOP/s. I would guess that a “reasonable high end” estimate is therefore >1 OOM smaller than this.

I have translated my subjective uncertainty into this probability distribution:



I have also indicated the median of the distribution and the estimate of the Landauer limit upper bound described above, as well as my best guess for the median number of FLOP/s performed in mouse brains, bee brains, and *C. Elegans* nervous systems.

To be fully principled, I should be explicitly accounting for my subjective credence that “this basic way of thinking about brain computation is somehow broken”, including that we are misunderstanding the Landauer limit argument and the true right tail should be longer and thicker. I have not done that here because it would have been too cognitively demanding to determine how best to incorporate that kind of model uncertainty, which feels different-in-kind from my uncertainty over this quantity from *within* a particular model.

This distribution should be thought of as *conditional* on the approach being sound, the brain being very unlikely to do reversible computation, and the estimate of the Landauer being correct and applicable in the way we believe it is. The possibility that these assumptions are mistaken will ultimately determine how much weight I give to the whole biological anchors framework relative to other big picture ways of getting at the TAI timelines question.

designs effectively on paper, but in ML it seems to be harder to predict which architectures will work well and why, forcing the use of trial-and-error (which is more analogous to the slower evolutionary process) rather than principled design.

Note that this does not necessarily represent Joe's subjective probability distribution over brain computation (even given the caveats above), although he has looked at it and considers it to be a roughly reasonable state of subjective uncertainty for someone to be in given the evidence.

Transformative model FLOP / subj sec

How much computation would a model need to perform to have a "transformative" impact given current architectural sophistication as of 2020? I argued above that because of the possibility of training a "balanced" transformative model, it is reasonable to use the human brain as an existence proof in nature when thinking about the potential FLOP/subj sec of a transformative model, and anchor to "brain FLOP/s" when estimating transformative model computation.

My subjective estimate after reading Joe's report is that there is a ~50% chance that evolution would have been able to "work with" ~1e15 or fewer FLOP/s to produce a system roughly as intelligent as a human. If we think of humanity as competing with evolution to design efficient model architectures for similar goals, how much larger or smaller would our models need to be than the human brain before they can be trained to solve a transformative task?

My central estimate is that a transformative model would need to be **~1 OOM larger than the brain (~1e16 FLOP / subj sec) given the sophistication of current ML architectures**. I use this estimate both for the Neural Network hypotheses and the Genome Anchor hypothesis (more in Part 3).

In this section I go over:

- What I think we should believe on priors, before examining empirical evidence (more).
- What sources of empirical evidence could potentially shed light on this, and what I believe they imply (more).
- What my overall subjective distribution is over the amount of computation that a transformative model may run on (more).

What should we believe about model FLOP/ subj sec on priors?

At a high level, I think there are a few key dynamics that seem relevant:

- Evolution has had vastly more time to optimize brain architectures than human researchers have had to optimize ML architectures.
- However, human researchers a) have the benefit of forethought and explicit backwards-chaining from their desired goals which should allow them to explore the space of possible architectures vastly faster, and b) can potentially benefit from examining the brain's architecture and attempting to reverse-engineer it.[47]
- The goal of designing a model architecture suitable for solving a balanced transformative task is probably at least slightly "easier" in some sense than the analogous task of

47 These distributions were generated by simply shifting the distribution of human brain computation X orders of magnitude to the left for an animal with X OOM fewer synapses; synapse counts are from this list on Wikipedia, which I have not vetted.

designing a brain architecture suitable for "being as intelligent as a human", since a model which displays a strict subset of the behaviors an intelligent human can display would still be transformative; if evolution were selecting for "transformativeness" rather than "intelligence" in humans, it could likely have been able to work with a smaller brain. Additionally, because energy and space are relatively cheap, a transformative model could be physically much larger and consume more energy than an animal whose brain performs a similar amount of computation.

Overall, my gut-level synthesis is as follows:

- The advantages and disadvantages of evolution relative to human design should net out somewhere between a wash, and a slight advantage to evolution given its sheer scale and the intuitive intricacy and "impressiveness" of natural systems, and the relative "newness" of deep learning compared to other fields of technological R&D. This points toward making no adjustment or a slight upward adjustment, **say adding something like -0.5 to +1.5 OOM** to the brain compute estimate in order to derive an estimate of the required model compute.
- The consideration that a balanced transformative task is somewhat easier than the "task of being a human" points in the direction of a downward adjustment from brain compute; **I could see this being -1 OOM, but it feels unlikely to be -2 OOM.** This is based on an intuitive judgment of the seeming "intelligence" of animals with different numbers of neurons. It seems to me that a chimp-sized brain (which is ~0.5 OOM smaller than a human brain)[48] could have relatively easily been optimized over evolutionary time to do scientific research given the right fitness signal, since chimps are already able to use tools and learn rudimentary sign language. On the other hand, mice (whose brains are ~2 OOM smaller than human brains)[49] don't seem to me to display the kind of flexible learning and "problem-solving" behavior that indicates the same potential.

I think these kinds of judgments are particularly likely to be different for different people, and often come down to "intellectual aesthetic" in significant part. I encourage readers to generate their own probability distributions for this quantity in particular.

What empirical evidence might bear on model FLOP / subj sec?

I can think of two potentially promising avenues to getting a better handle on how ML architectures would likely compare to brain architectures:

- We can potentially turn to existing examples of human-designed artifacts "competing" with artifacts produced by evolution in non-AI domains (e.g. artificial organs), which are

48 The TD-Gammon evaluation function performs about 16,000 FLOP per move it evaluates. I was unsure how many moves it could evaluate per second; I assumed ~4 moves per second because this is in the range of the number of actions humans take per second in video games, as well as being similar to the number of words humans read per second. See this appendix for details.

49 The compute estimate is for the raw model (i.e. without search); my understanding is that this model achieved "decent" play, while adding two-ply search (which Paul guesses would have increased compute by ~100x) caused its play to be comparable to top human backgammon players.

often much more straightforward to evaluate, to get a sense for how human designs compare to evolutionary "designs" in general.

- We can estimate the distribution of brain FLOP/s for various smaller animals, and try to qualitatively gauge how the capabilities of ML models today compare to the capabilities of the animals they are closest to in size. This has the potential to be more satisfying if we can compare functional subparts of an animal's brain (e.g. the visual system) with ML models of a similar size which are attempting to solve a similar task (e.g. object recognition or video prediction).

How has human technology compared to nature in non-AI domains?

Paul Christiano has spent a day looking into the first question, producing this document, which briefly explores:

- How various artificial organs (e.g. artificial hearts or dialysis machines) compare in energy cost[50] to their natural counterparts.
- How solar panels compare to plants at converting sunlight into stored energy, in terms of efficiency (what fraction of the incoming sunlight is successfully converted to electrical or chemical energy by a solar panel or leaf) and payback period (how long it would take for a solar panel or leaf to capture as much energy as it took to manufacture it)..
- How energy storage in chemical batteries compares to energy storage in fat cells, in terms of payback period.
- How computer *hardware* (as opposed to software) compares to estimated hardware power in the brain, in terms of FLOP/s per Watt (which is equivalent to FLOP/Joule) and manufacturing cost in energy.
- How photodetectors in cameras compare to the photodetectors in the human retina, in terms of a combination of desired qualities such as pixel resolution, brightness needed to recognize colors, and temporal resolution vs energy consumption.
- How actuators compare to limbs and muscles in terms of energy cost to do a comparable-seeming type of locomotion.

The findings are summarized in this table (put together by Danny Hernandez on the basis of Paul's research):

How Much Worse are Human Engineered Artifacts than Evolution		
	Performance (powers of 10)	*Metric*
Dialysis Machine	3 worse	Energy cost
Artificial Hearts	1-2 worse	Energy cost
Solar power	1 better	Efficiency
Solar power	4 worse	Payback period
Chemical Energy Storage	2 worse	Payback period

50 I have not researched this much, but e.g. this study suggests temporal resolution is around 100 frames per second for color vision, while this study suggests that discrimination between two different stimuli is perfect around 50 frames per second and decays to being no better than random chance at 200 frames per second.

V100 GPU	1-2 worse	Flops/w
V100 GPU	4 worse	Manufacturing energy cost
Photodetector	3-4 worse	Overall performance given power
Locomotion	2 worse	Energy cost

I have not attempted to vet or reproduce Paul's research on this question; I hope to investigate this more thoroughly in the future. Additionally, the evidence this kind of analysis can provide about the question of interest is inherently fairly limited:

- It is very difficult to properly take into account cases in which humans have more or less completely failed at replicating a certain piece of "natural technology" -- for example, biological systems appear to be remarkably good at producing very small-scale machinery of a level of complexity that human nanotechnology does not seem close to replicating, even allowing for many OOM increases in energy costs.
- It is similarly difficult to take into account cases in which it seems that humans might have designed technologies with no clear natural analogue. Arguably, long-range high-speed land transportation such as cars and trains (which seem to be able to cover long distances substantially more effectively than animals walking) could fall into this category, as could highly lethal weapons such as guns and bombs (which seem substantially more effective at dealing damage than claws/fangs/stingers/etc).
- The dynamics of research and development are highly different across different fields of technology; it's unclear how ML research should compare to these examples.

With that said, at face value the technologies Paul selected seem like reasonably salient "central examples" of human designers attempting to compete with a natural artifact to me, and I think that they would provide a relatively valuable central estimate, because *a priori* I don't have a strong instinct about whether I should expect human designers to have a comparative advantage or disadvantage at designing model architectures compared to designing these physical objects.[51]

Of the metrics he examined, the measures of *utility per unit power consumption* (as opposed to manufacturing costs, payback period, or efficiency) seem to be the most relevant source of evidence for thinking about how many FLOP / subj sec model architectures would need to run on to compete with brain architectures.

The average of these measures for the dialysis machine (~3 OOM worse), artificial hearts (~1.5 OOM worse), the V100 GPU (~1.5 OOM worse), the photodetector (~3.5 OOM worse), and locomotion (~2 OOM worse) from the table above is ~3.2 OOM worse, and the standard deviation is ~1 OOM. Additionally, I am inclined to place a little bit more weight on the V100

51 According to Table 2 of Tan and Le 2019, models with strong ImageNet performance range from ~1e9 to 1e10 FLOP/forward pass; my understanding is that models which achieve a comparable performance on CIFAR-100 are more in the range of ~1e7-1e8 FLOP/image.

data point because the ML field intuitively seems more correlated with the computer hardware field than the others.

However, I am unsure exactly how relevant this is. It might be the case that evolution has optimized for energy efficiency to a degree that human designers generally don't need to, because energy is relatively abundant for us while it is scarce and dangerous to acquire for many animals. The fact that evolutionary artifacts tend to be more energy-efficient than their human-designed counterparts may simply reflect that evolution was under a stronger constraint there. With that said, evolutionary artifacts tend to be superior on multiple dimensions, and not obviously deficient on any particular dimension, so I am still inclined to assign some penalty for that. My central estimate from this methodology downward to **something like ~2.5 OOM worse, +/- 1 OOM.**

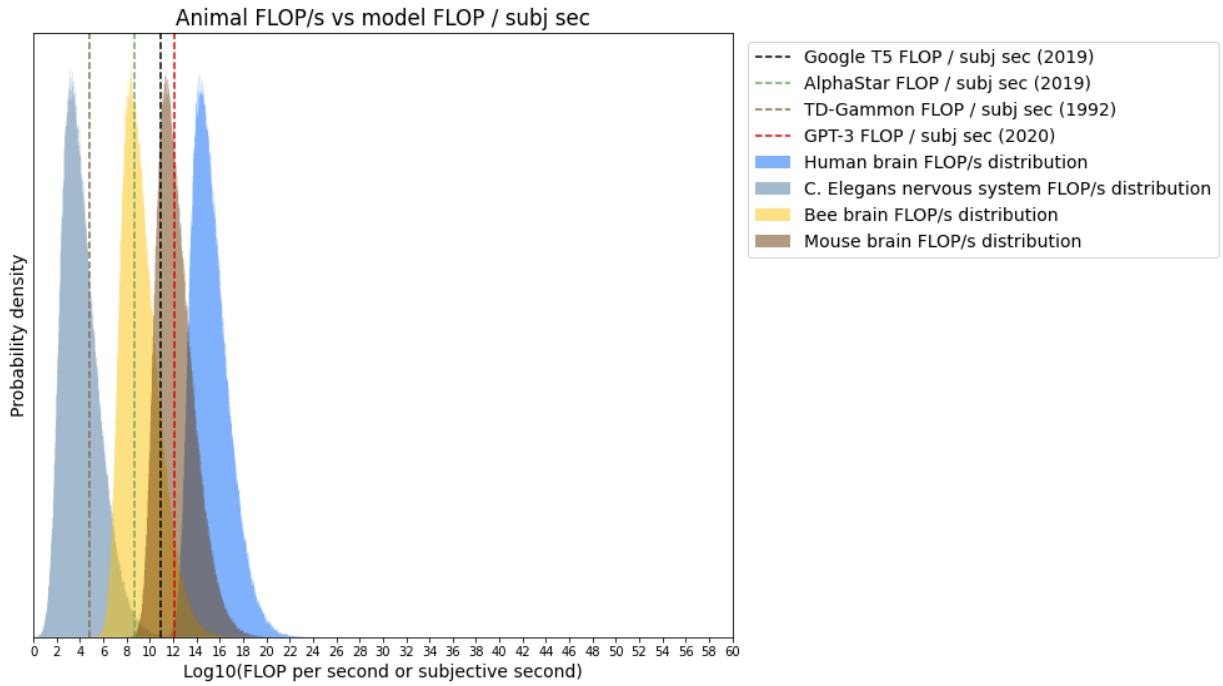
How do the capabilities of models compare to those of animals?

Another, substantially rougher, way to get an intuition for how human designed model architectures may compare with brain architectures is to qualitatively examine the "intelligence" / "impressiveness" of various models with that of various small animals.

Note that it is *not* a prediction of this framework that we ought to be able to *closely simulate* particular animals using a model roughly similar to the size of their brain. This doesn't isolate the impact of *model size* from other factors -- a failure to closely simulate a bee might be due to human researchers being unable to generate a good simulation of the bee's body or environment, or being unable to design an objective function that incentivizes "bee-like-ness" (which might need to involve expensive human feedback from bee experts). The relevant prediction is something like "*Given* a well-specified environment and objective function, how large of a model would we need to train such that the model optimizes its objective 'about as well' as an animal with a certain brain size appears to be 'optimizing' for its genetic fitness?" This is very murky to think about, because the tasks that ML models are solving are generally very different from the tasks that animals are implicitly solving, and it is very difficult to judge how "well" an animal is doing at "optimizing its overall fitness."

Below, I show the probability distributions for the brain computation levels of three smaller animals (*C. elegans*, honeybee, and mouse) along with the probability distribution of human brain computation and the runtime FLOP / subj sec of various ML models[52]:

52 Roodman 2020 ("Modeling the Human Trajectory") gives a table of GWP figures on pg 40. The GWP was $350B in 1600, $397B in 1700, $741B in 1820, $1128B in 1870, and $2767B in 1913) all in 1990 USD. The rate of exponential growth from 1600 to 1700 is given by ln($397B / $350B) / (1700 - 1600) = 0.12%, the rate of growth from 1700 to 1820 is given by ln($741B / $397B) / (1820 - 1700) = 0.52%, the rate of growth from 1820 to 1870 is given by ln($1128B / $741B) / (1870 - 1820) = 0.84%, and the rate of growth from 1870 to 1913 is given by ln($2767B / $1128B) / (1913 - 1870) = 2.1%; the average in the 1820 to 1913 period is ln($2767B / $741B) / (1913 - 1820) = 1.4%.



Intuitively, this comparison makes me inclined to evaluate human-designed ML architectures somewhat more favorably relative to brain architectures than the earlier methodology:

- I am particularly struck by TD-Gammon, which would run on about as much computation as my median estimate for *C. Elegans* if it is able to evaluate four moves per second.[53] It feels intuitively impressive to me that an RL model running on as much compute as a microscopic worm was able to achieve decent play at backgammon while playing multiple moves every second.[54] Even if I attempt to take into account the fact that *C. Elegans* has to navigate a more diverse continuous environment with a greater number of possible actions, it feels relatively implausible to me that playing backgammon quickly at the level of reasonably-skilled human players is e.g. hundreds of times "easier" than the "task" of being a small worm.
- It also seems intuitively plausible to me that AlphaStar is not drastically less sophisticated than a honeybee, or that a very large-scale transformer would be somewhat more sophisticated than a bee (given the grammatically coherent, idiomatic, and roughly on-topic text samples that substantially smaller transformers can generate).

53 One way to do this could be to introduce "filler" data. For example, we could modify the StarCraft environment so that a blank screen is inserted in between every frame of the normal game. Another way to do this could be doubling the noise in the loss signal. For example, every time the model would have received a reward signal with value r, we could instead flip a coin, and give the model r reward if it comes up heads and 0 reward if it comes up tails.

54 Note that this loss of probability mass is not shown visually in the plots above, because the plotting software automatically renormalizes the histograms so that the total area under their curve adds up to 1. This means that when a distribution gets *narrower* -- as the Lifetime Anchor distribution did when low-end values were cut off -- it also gets *taller.* However, I take this into account when I assign weights to different hypotheses below.

Additionally, Paul has spent a few hours investigating the question of how insect vision compares to various image models (which has the potential to be a much tighter analogy than the above analogies) in order to inform his personal views on timelines. While this analogy has the potential to be more satisfying, it is still highly conceptually fraught and empirically messy to make such comparisons. Paul looked into this question for personal purposes almost two years ago, and he has not carefully written up his reasoning and evidence in a way that would allow others to examine it; he no longer necessarily endorses the precise numerical conclusions. Nonetheless, I felt it might be valuable to nonetheless report his findings because it is informing my intuitions. We hope to conduct our own investigation into the vision analogy in the future (though the functional method section in Joe's report explores this to some extent).

Paul's remembered impression from his research into the visual capabilities of various insects is that honeybee vision seems to require heuristics that are somewhat more sophisticated than ones that would be sufficient to perform well on CIFAR-10, but somewhat less sophisticated than would be required for ImageNet. Additionally, CIFAR-10 and ImageNet models tend to be evaluated ~1-10 times per second, while bee vision appears to have a temporal resolution of ~50-200 frames per second.[55] Based on the size of models that achieve high performance on these respective datasets, **that would suggest that architectures which run on between ~1e7 and ~1e10 FLOP / subj sec could "compete with" bee vision.**[56] My median estimate for the total FLOP/s of a bee's brain is ~1e9 FLOP/s, and my understanding from Paul is that >= 10% of the neurons in a bee's brain deal with visual processing; this is consistent with image recognition architectures being only slightly worse than, or similar to, the architecture of a bee's visual system.

As with judgments about priors**, I think judgments about "model impressiveness" or "task sophistication" are particularly likely to be different for different people;** I do not necessarily think readers should place much weight on my or Paul's estimates, and should potentially lean more heavily on their priors.

Distribution over FLOP / subj sec for a transformative model

In summary,

- I think considerations of priors indicate that the median estimate for model computation requirements should be somewhere between brain computation (~1e15 FLOP / subj sec) and ~1 OOM smaller (~1e14 FLOP / subj sec).
- Empirical evidence from comparing non-AI technologies to their natural analogues, and from comparing insect visual systems to image recognition models, seem to suggest that we should expect model architectures to be worse than brain architectures by ~1-3 OOM; an intuitive examination of less-easily-comparable capabilities makes me inclined to believe that our architectures are unlikely to be *too much* worse than a couple OOM.

55 Note that the location of the grey distribution is irrelevant and does not impact future calculations.
56 Recall that one "subjective second" of data corresponds to the amount of information of a certain type (e.g. words, sounds, images) that a human brain could process in one second.

- The evidence feels thin enough and murky enough that it seems reasonable to rely substantially on our priors.

I represented my uncertainty over the number of orders of magnitude larger or smaller I expect current model compute requirements to be for a transformative model, compared to my estimate for the brain as a lognormal distribution with **a mean of 1 OOM larger and a standard deviation of 2 OOM** (the choice of this standard deviation is informed by my beliefs about algorithmic progress; see this discussion for an explanation).

Incorporating this, the resulting subjective distribution over transformative model FLOP / subj sec looks like this (shown along with the distribution over human brain FLOP/s for comparison):

Guide to the rest of the report

- Part 2 reviews what ML theory, scaling experiments, and observational evidence imply about how training data requirements scale with model size, how I arrive at subjective probability distributions over the scaling exponent α and constant K, and how I think about effective horizon length. These estimates are used to estimate training FLOP requirements for the Genome Anchor hypothesis and the three Neural Network hypotheses.
- Part 3 goes into more detail on the six biological anchors hypotheses and how I estimate the combined 2020 training FLOP requirements distribution.
- Part 4 goes into more detail on my forecasts for algorithmic progress, willingness to spend on computation, and hardware prices, and how they combine to form an estimate

of when the amount of computation required to train a transformative model may become affordable; I also address several common questions and objections, which I will describe below.

Responses to questions and objections

In Part 4, I address the following high-level questions and objections to this framework:

- Does this framework assume that transformative AI is "hardware bottlenecked" (more)?
- What if training data or training environments will not be available by the time the requisite computation is available, or the computation to run training environments exceeds the computation to train the model (more)?
- What if training a transformative model would take too much wall-clock time even if the computation is available (more)?
- Is there a risk of selection bias if we're extrapolating training data requirements from existing ML problems, because we are more likely to have solved tasks with ML when the scaling behavior is favorable (more)?
- The extrapolation of willingness to spend assumes continuous growth, but what if there is another "AI winter" causing spending to stop rising or to drop (more)?
- Would the biological anchors approach generate the same estimates for training FLOP requirements at all points in the past, even though algorithmic progress means that each year should be different (more)?
- How could we test the predictions of the biological anchors framework and potentially falsify it (more)?
- How applicable is this model if transformative AI arrives by a very different path, e.g. via distributed automation or through a new "paradigm" (more)?
- Instead of using biological anchors, why not simply directly gauge how capable or impressive AI systems are and extrapolate this progress forward (more)?

Forecasting TAI with biological anchors

Part 2: How training data requirements scale with parameter count

Author: Ajeya Cotra
Date: July 2020

This report emerged from discussions with our technical advisors Dario Amodei and Paul Christiano. However, it should not be treated as representative of either of their views; the project eventually broadened considerably, and my conclusions are my own.

This is a work in progress and does not represent Open Philanthropy's institutional view. We are making it public to make it easier to gather feedback, to help inform others' thinking in the effective altruism community, and to allow for follow-on work outside of Open Phil. However, we may edit it substantially in the future as we gather feedback from a broader audience and investigate open questions. Accordingly we have not done an official publication or blog post, and would prefer for now that people not share it widely in a low-bandwidth way (e.g., just posting key graphics on Facebook or Twitter).

This report has been split into four Google docs in order to load faster. ***This is Part 2; the first part is here, the third part is here, and the fourth part is here.*** *Additional materials (collected in this folder):*

- *Quantitative model: the Python notebook Biological anchor hypotheses for 2020 training computation requirements; a template spreadsheet When required computation may be affordable; and my best guess, conservative, and aggressive forecasts.*
- *Supplemental materials: a document containing various appendices; a folder of figures for the report; the spreadsheet Extrapolations of data and compute to train models; and the Python notebook Compute price trends, which draws on data in this folder.*

In Part 1, I provided an overview of the framework and estimates, provided definitions for key abstractions used in the model, and generated an estimate for the number of FLOP / subj sec of a transformative model. In Part 2, I will provide an overview of the evidence we can use to think about training data requirements for a transformative model, by generating a scaling law relating the number of parameters that characterizes a model to the number of data points required to train it.

The scaling law derived in this part will be used in Part 3 to estimate 2020 training computation requirements for the Genome Anchor hypothesis and the Neural Network hypotheses. Specifically, the amount of computation required to train a transformative model for each of those hypotheses will be estimated as

$$\text{Train FLOP} = (F \text{ FLOP / subj sec}) \times (T \text{ subj sec of training})$$

I generated an probability distribution over F, the FLOP / subj sec of a transformative model, in Part 1 here; my median estimate is ~1e16 FLOP / subj sec. Here, I will focus on how to extrapolate T, the number of subjective seconds of data that a model must be trained on, as a function of the parameter count P of a transformative model -- the larger the parameter count,

the more data is needed to train the model. The Neural Network hypotheses and Genome Anchor hypothesis estimate P differently; this will be covered in Part 3.

In the rest of Part 2, I will:

- Explain why machine learning theory would predict that data requirements tend to scale linearly with parameter count (more).
- Examine two papers which ran controlled experiments attempting to elicit scaling laws relating dataset size and model parameter count, both of which conclude that dataset size D scales can be described as a power law function of parameter count P (i.e. $D \approx K P^{\alpha}$), with the more relevant result predicting that this scaling is sub-linear (more).
- Introduce the concept of a "horizon length" to operationalize what "one data point" means for different ML problems (more).
- Present some estimates of the number of data points various recent RL models were trained on, which is consistent with a linear or slightly sub-linear scaling (more).
- Summarize my thinking on how data requirements scale with parameter count, expressed as a power law function with a probability distribution over the exponent α and constant factor K (more).

How sample complexity scales in ML theory

Machine learning is essentially *search over programs*: we set up a search space containing a large set of candidate computer programs (usually parametrized by some number of numerical parameters), and use an optimization algorithm to select a program from that space which performs well according to a certain loss function.

Machine learning theory suggests that **the amount of data required to find a model that is "close to optimal" is a linear function of parameter count and the variance of the loss function**. In this section, I provide a more precise description of how we should expect data requirements to scale from a theoretical perspective:

- I'll first define "sample complexity" and explain theoretical results that bound the sample complexity of a learning task under certain conditions (more).
- I'll discuss the extent to which the conditions required for these theoretical sample complexity bounds would obtain in real-world machine learning problems (more).
- I'll briefly address how these bounds may apply to other optimization algorithms besides stochastic gradient descent, the most commonly used algorithm in modern ML (more).

Later in this document, I will cover empirical evidence about training data requirements for modern ML models from both controlled experiments and observational data.

Definition of "sample complexity" and theoretical bounds

I first learned the information in this section from the textbook Understanding Machine Learning: From Theory to Algorithms, by Shalev-Shwartz and Ben-David; it is standard and can be found in many other sources.

Consider a finite search space over candidate programs, denoted as H (which stands for "hypothesis space"). For any loss function[1] L and data distribution D, there is at least one program $h^¿ \in H$ such that $h^¿$ achieves the lowest loss on data distributed according to D of any program in H. Written in mathematical notation this looks like:

$$\exists h^¿ \in H \vee \forall h \in H, E_{d \sim D}[L(h^¿, d)] \leq E_{d \sim D}[L(h, d)]$$

Where $E_{d \sim D}[f(d)]$ denotes "the expected value of the expression $f(d)$ if the variable d is distributed according to the distribution D."

An optimization algorithm draws samples of data $d_1, d_2, d_3, \ldots$ from the distribution D (for example, images from a dataset of images)[2] and attempts to select a candidate program that has high average performance on *that finite sample*, in the hope that it would also have high average performance *on the full distribution*. The goal is to select a candidate program $\hat{h}$ that we are confident will achieve a loss that is "close" to the loss that $h^¿$ achieves. Specifically, we would like the probability that $E_{d \sim D}[L(\hat{h}, d)] \leq E_{d \sim D}[L(h^¿, d)] + \epsilon$ to be $\geq 1 - \delta$, where $\epsilon > 0$ and $\delta > 0$ are small constants that we choose.[3]

The **sample complexity** is defined as the number of data points that our optimization algorithm needs such that with probability **at least** $1 - \delta$, the expected loss of the model returned by the optimization algorithm on the full data distribution, $E_{d \sim D}[L(\hat{h}, d)]$, will be **at most** ϵ larger than the expected loss of the best model in the hypothesis space on the full distribution, $E_{d \sim D}[L(h^¿, d)]$.

When certain "learnability" conditions apply, it is possible to calculate an upper bound on sample complexity for a certain optimization algorithm given only the size of the hypothesis space H, the loss function L, the desired accuracy ϵ, and the desired confidence level δ, while making only relatively weak assumptions about the data distribution D. I cover two sample complexity bounds and their relevant conditions below: the empirical risk minimization bound and the stochastic gradient descent bound.

Empirical risk minimization bound

This sample complexity bound holds under the following conditions:

- There are a **finite** number of candidate programs in H. This condition is obtained for physically instantiated machine learning models because they are characterized by a

1 It is conventional to cast the optimization problem as the problem of *minimizing* an objective rather than maximizing one; I'll do the same.

2 Note that it's not always clear what constitutes "one data point" or "one sample", particularly in reinforcement learning settings. I will address this complication later in the document. For now, it is easiest to think about these theoretical results in the context of supervised learning (e.g. image classification), where it is relatively unambiguous what "one sample" means (e.g. "one image").

3 This definition of learnability is called "probably approximately correct" learnability or PAC learnability.

finite number of parameters that have a finite precision. For example, if a neural network has 1 million parameters stored as 32-bit precision floating point numbers, then each parameter can then take on 2^{32} possible values, meaning that there are ¿possible settings of the parameters overall -- a huge but finite quantity.[4] More generally, a model parametrized by P values each with a precision of b bits has size ¿$H \vee$¿2^{bP}.

- The loss function L is **bounded**, with a maximum at L_{max} and a minimum at 0.[5]
- The data distribution D is unchanging, and each data point d_i is drawn independently at random from D; i.e., they are independent and identically distributed (i.i.d.) random variables.
- The optimization algorithm implements the **empirical risk minimization** (ERM) rule: given a sample $S=(d_1, d_2, d_3, ... d_N)$ where d_i are i.i.d., it returns the hypothesis $\hat{h}$ which achieved the **lowest average loss on the sample** S (this is called the "empirical loss" or "empirical risk"). We could clearly implement the ERM rule using exhaustive search (i.e. computing the empirical loss of every model $h \in H$ on the sample S); for some learning problems it is possible to write a much more efficient optimization algorithm which finds the hypothesis that minimizes empirical risk.

The law of large numbers states that in the limit of infinite data, the *empirical* loss of every model $h \in H$ on the sample will converge to the theoretical mean almost surely:

$$\forall h \in H, \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} L(h, d_i) = E_{d \sim D}[L(h, d)], \text{ with probability 1}$$

Once the number of samples N is large enough that all of the individual empirical losses $\frac{1}{N} \sum_{i=1}^{N} L(h, d_i)$ associated with each hypothesis $h \in H$ are very close to their respective expectations $E_{d \sim D}[L(h, d_{\square})]$ with very high probability, then the hypothesis $\hat{h}$ that minimizes empirical risk will have a true loss very close to the best hypothesis $h^{¿}$ with high probability.

Further assumptions about the probability distributions are required to estimate *how quickly* this convergence will occur. There are a number of inequalities in probability theory that upper-bound the probability δ that a sample mean estimated from N independent samples of a random variable X deviates from the true mean $E[X]$ by more than some small ϵ given certain assumptions about the properties of X.

4 An abstract (as opposed to physically instantiated) neural network would be an infinite hypothesis space because its parameters could theoretically take on any real value. Infinite hypothesis spaces are not necessarily PAC learnable with finite data using the empirical risk minimization optimization rule, but neural networks actually are PAC learnable. The sample complexity of learnable infinite hypothesis spaces can be bounded using their VC dimension; this usually results in a tighter bound than the simple empirical risk minimization bound I describe in this section, but I've left it out because it is more mathematically complicated, and physically instantiated models satisfy the finiteness condition this bound demands.

5 Any bounded loss function whose minimum loss is something other than 0 can simply be shifted by a constant factor so that it fits this form without affecting learnability.

The proof of the ERM sample complexity bound leans on one of these inequalities, Hoeffding's inequality, which makes the assumption that the probability distribution over the loss is bounded on both ends. Because the loss function L is bounded, each of the random variables $L(h, d_{\square}), d \sim D$ are also bounded, and Hoeffding's inequality can be applied to bound the rate at which each of them will converge to their respective expectations.

This gives a sample complexity bound of $2 \times L_{max}{}^{2} ¿$ data points. This bound grows:

- *Logarithmically* with $¿ H \vee ¿$, the size of the hypothesis space. As shown above, a model parameterized by P parameters each with b bits of precision has a hypothesis space of size $¿ H \vee ¿ 2^{bP}$ -- this means that $\ln ¿$. This means that the sample complexity upper bound grows **linearly in the number of parameters.** My understanding is that tighter bounds can be proven which exploit the fact that the hypotheses in a hypothesis space are usually not independent of one another (for example, two neural networks with very similar weights will have highly correlated losses and loss variances). These bounds will use a more sophisticated measure of the complexity of the hypothesis space (such as the VC dimension) rather than literally counting possible hypotheses, and they are typically independent of the precision b.
- *Inverse-quadratically* with ϵ / L_{max}. If the best hypothesis in the space has an expected loss very close to 0 and the worst has an expected loss very close to L_{max}, then ϵ / L_{max} is roughly "what fraction of the way from the worst hypothesis to the best hypothesis we are aiming to get." In most cases it is possible to prove a tighter sample complexity bound which replaces the $L_{max}{}^{2}$ term with something that looks more like the *worst-case variance* of the loss function for any hypothesis in the space,[6] $\sigma_{max}{}^{2}{}_{\square} = \max_{h \in H} \sigma^{2}{}_{d \sim D} L(h, d)$.

  In this bound, the ratio ϵ / σ_{max} can be interpreted as "how close we want to be to the best model *expressed in units of standard deviations* of the loss", and the maximum sample complexity grows quadratically in the inverse of "desired accuracy in worst-case standard deviation units." **This is essentially a signal-to-noise ratio** where the strength of the "signal" corresponds to how finely we want to discriminate between different models -- the more finely we would like to discriminate between models in the hypothesis space relative to the worst-case noise in the loss signal, the smaller the signal we would like to pick up, and the more data we will need.
- *Logarithmically* with $1/\delta$, the inverse of the desired confidence level. In practice this ends up mattering very little: for hypothesis spaces as big as those typically used in deep learning, the $2 \ln 2 - \ln \delta$ term in the numerator will be very small compared to $\ln ¿$ and make very little difference unless extreme confidence is sought. Many learning

6 For this bound, the relevant assumption about the loss function is that it has a *finite variance* on the data distribution in question for all hypotheses in the hypothesis space. If the loss function is bounded, it automatically has a finite variance. If the support set of the loss function is unbounded - i.e., if the loss function could take on values ranging from -infinity to infinity -- then it must be sub-Gaussian, i.e. its tails must decay at least as fast as the tails of a Gaussian distribution (which is the maximum entropy distribution for a specified variance).

theory bounds simply directly bound the *expectation* $E_{d \sim D}[L(\hat{h}, d)]$, entirely dropping the δ term.

Dropping the dependence on b, replacing the maximum loss with the worst-case loss standard deviation σ_{max}, and dropping the term involving δ, the sample complexity bound becomes:

$$N_{ERM}(\epsilon, P, \sigma_L) \leq C \times P \times ¿$$

Here the constant factor in the scaling law (C) depends on the hypothesis space and is usually quite small relative to the other terms. In later sections, we will essentially roll together both C and the signal-to-noise ratio ¿ into a single constant factor K, which we will attempt to determine empirically, either from experiment or observational data about the number of data points required to train models of various sizes.

Stochastic gradient descent bound

Most real-world machine learning uses some variant of stochastic gradient descent (SGD) as the optimization algorithm. The basic version of SGD works roughly as follows:

1. Start with a random assignment of parameters $\theta_0 = [\theta_0^{(1)}, \theta_0^{(2)}, ... \theta_0^{(P)}]$.
2. For $t = [1, ...T]$ timesteps,
    a. Draw a new sample d_t from some distribution of data D_t.
    b. Compute the gradient vector[7] of the loss function with respect to the current parameter vector θ_t on this data point: $g_t = \nabla_\theta L(\theta_t, d_t)$.
    c. Update the parameter vector to move a small distance η_t in the direction *opposite* the gradient (i.e. "downhill"): $\theta_t = \theta_{t-1} - \eta_t g_t$.
3. Return the model characterized by the final parameter values θ_T.

For some problems, SGD will return the hypothesis that minimizes empirical risk. I discuss a set of criteria under which SGD is guaranteed to find the global minimum in this appendix. If the data points are all i.i.d., then the ERM bound can be applied directly to SGD for such problems:

$$D_{ERM}(\epsilon, P, L) \leq K \times P ¿$$

A more sophisticated argument can show that a similar bound applies even if the data points d_t are not i.i.d.[8] The main difference is in the "signal-to-noise ratio", where we substitute the worst-case loss standard deviation σ_{max} with the worst-case *gradient magnitude,* $¿ \vee \nabla_\theta L \vee ¿_{max}$:

$$D_{SGD}(\epsilon, P, L) \leq K' \times P ¿$$

7 This is also a vector of length *P.*

8 This setting is known as the online convex optimization setting; the typical assumption made in proving these sample complexity bounds is that each data point is selected adversarially to maximize the loss of the current model.

Because one sample is processed in each step, these bounds on sample complexity are also bounding the number of steps, and therefore the running time of the algorithm.[9]

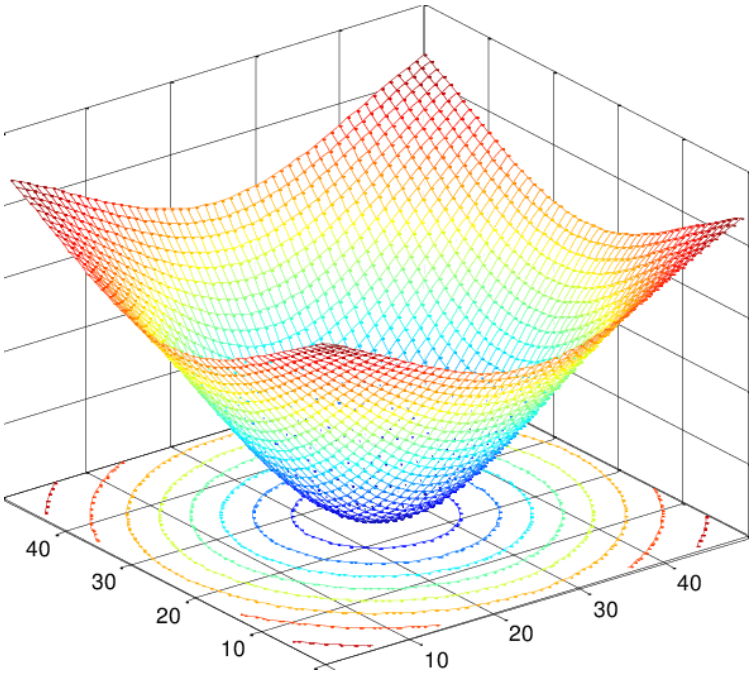
Intuitively, $¿ \vee \nabla_\theta L \vee ¿_{max}$ governs the most that the parameter values could move based on the influence of a single data point -- the larger the possible move from the next data point d_{t+1}, the harder it is to guarantee that θ_T will have good average loss. Because a single data point generally provides a noisy estimate of the true gradient, $¿ \vee \nabla_\theta L \vee ¿_{max}$ is usually (but not always) dominated by the magnitude of the gradient *noise*, rather than the magnitude of the *expected* gradient -- and the gradient noise, in turn, tends to be larger when the noise in the loss signal (σ_{max}) is larger.

Can these bounds apply to real-world problems?

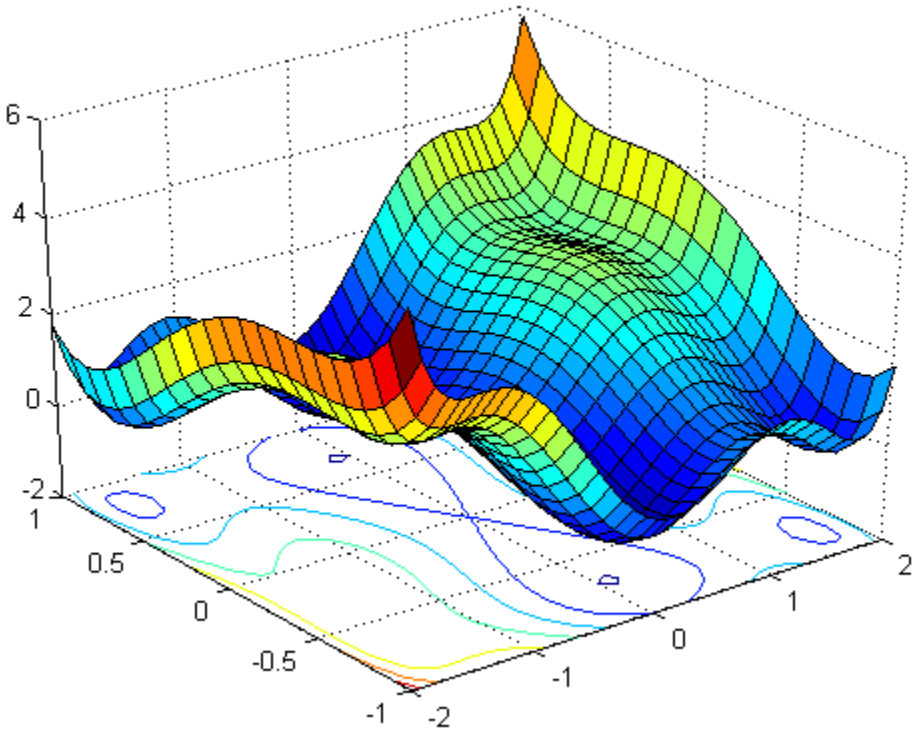
In most real-world ML problems, the conditions required for this bound to provably apply are *not* met. The true expected loss function over the space of parameter settings is often called a **loss landscape**. Roughly speaking,[10] the loss landscape needs to be "convex" to be learnable with SGD (I explain the criteria in more detail in this appendix). Intuitively, this means that the loss landscape has a unique lowest point and it smoothly and gradually curves up from that point, with the curve getting steeper as you move further from the lowest point. With only two parameters p_1 and p_2, the loss landscape $E_{d \sim D}[L(p_1, p_2, d)]$ would basically look like a bowl, e.g. like this function:

9 As I argued in Part 2, the computation required to do one forward pass of a neural network tends to scale linearly with parameter count, which means that for learning problems in which SGD implements the ERM rule, training *computation* should scale roughly *quadratically* with parameter count.

10 Convexity isn't the only criterion, but it is the most salient. I use "convex" as a short-hand here for "convex-smooth-bounded" or "convex-Lipschitz-bounded", for readability. The "bounded" criterion is automatically satisfied for finite hypothesis spaces such as physically instantiated neural networks.



A ball set anywhere in this loss landscape will roll to the valley at (25, 25), which is analogous to SGD finding the empirical risk minimizing hypothesis. However, most loss landscapes for real learning problems aren't convex:

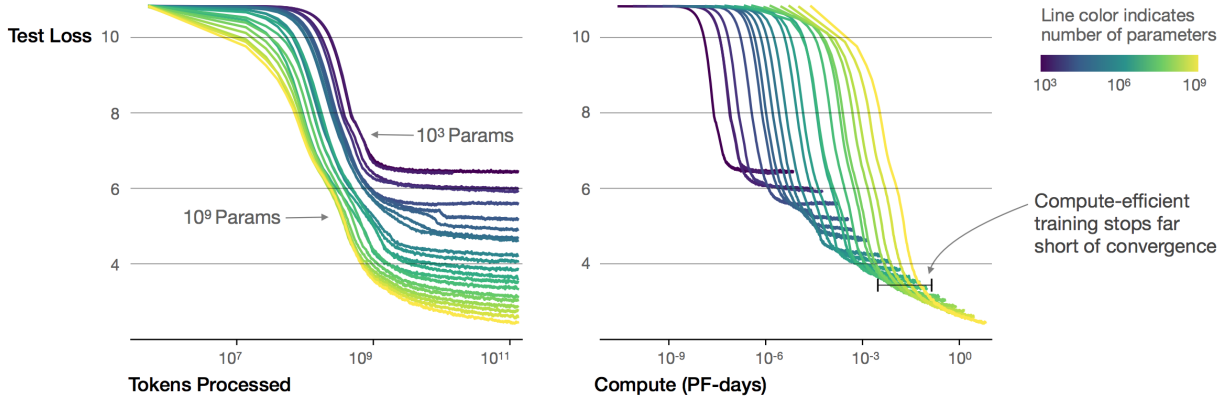


Notice that the function in this image has multiple local minima: "valleys" in the landscape where the function curves upward everywhere. If the loss landscape is not convex, SGD is not guaranteed to find the empirical risk minimizing hypothesis, or get anywhere close.

SGD can work well in practice in non-convex loss landscapes

With that said, **SGD does work reasonably well on a wide variety of practically-relevant non-convex loss landscapes.** Running SGD over loss landscapes that are almost certainly highly non-convex has produced models which generate idiomatic English text samples, play professional-level Go, and so on -- this is "good performance" in a practical, informal sense that is not defined relative to the best possible model in the model class.

When SGD "works well in practice", the average long-run loss tends to decrease fairly steadily as more data and computation is used, often converging toward some "lowest loss achievable by SGD" on that particular landscape.[11] This produces training curves like this:[12]



The lowest loss achievable by SGD may be a local minimum[13] that is *not* the loss that would be achieved by the best possible setting of parameters, analogous to how a ball placed at a random point on the landscape shown above would roll downward until it settled into one of the many valleys (which often won't be the deepest valley). Converging toward *some* setting of parameters θ_N which achieves a "good loss" turns out to be a much weaker desideratum than convergence to the *best* possible parameter setting.[14]

11 How smoothly loss decreases over time, and the precise value of the "lowest loss achievable by SGD", often depends on hyper-parameter settings.
12 Kaplan et al 2020, Figure 2, pg 4.
13 Or more often toward a saddle point (in which the local gradient is 0 but the loss landscape is curving upward on some dimensions and downward on others); saddle points are much more common in high-dimensional spaces such as the parameter spaces of neural networks than local minima.
14 My understanding is that in practice, for a surprisingly large variety of tasks neural network training converges to a model that gets 0 loss on the training dataset -- i.e. achieves the globally minimizing setting of parameters for the training distribution (although generally not for the test distribution). E.g., see Li et al 2018, pg. 1: "Training neural networks requires minimizing a high-dimensional non-convex loss function – a task that is hard in theory, but sometimes easy in practice. Despite the NP-hardness of training general neural loss functions [2], simple gradient methods often find global minimizers (parameter configurations with zero or near-zero training loss), even when data and labels are randomized before training [42]. However, this good behavior is not universal; the trainability of neural nets is highly dependent on network architecture design choices, the choice of optimizer, variable initialization, and a variety of other considerations."

It may well be the case that there is some possible setting of the parameters that describe GPT-2 which would generate dramatically more believable sentences than GPT-2 but was in practice difficult to reach using SGD, or some setting of the parameters that describe AlphaGoZero which would play dramatically better Go than AlphaGoZero but would be very difficult to reach using SGD, and so on. But my goal is to estimate the number of parameters that ML researchers would *in fact* need to use to design an architecture such that an optimization algorithm ML researchers could write down (like SGD or one of its variants) would *in fact* select a transformative model. In other words, **the estimate of parameter count should price in the possibility of landing at a suboptimal parameter setting** that achieves worse performance than the true "best setting of parameters" for that architecture but is nonetheless transformative.

The theoretical bound can provide a "prior" for real-world scaling behavior

Let's refer to the loss level that SGD converges toward (when it "works well") as the "reachable minimum loss." For these learning problems, we want to know how much data it would take to achieve a loss within ϵ of the reachable minimum loss.

For convex learning problems, the reachable minimum loss is simply the global minimum loss, and the worst-case data requirements are described by the SGD sample complexity bound. The bound doesn't directly tell us how much data is necessary for SGD to get within ϵ of the reachable minimum loss for non-convex problems. However, in the absence of specific contradictory information, I think it's reasonable to have a **lightly held prior that sample complexity will scale roughly linearly in parameter count** for machine learning problems on which SGD "works well in practice."

For any given non-convex problem, the reachable minimum loss could be substantially higher than the global minimum loss, but the formulation of this question controls for that possibility. Under this formulation, it's not clear *a priori* how non-convexity should impact sample complexity:

- It might be that because the loss landscape is non-convex, SGD needs to take a longer and more circuitous path from the initial model to the reachable minimum loss than it would for convex problems. If the ratio between the path taken by SGD and the shortest path from initialization to the reachable minimum loss gets larger as parameter count increases -- that is, if SGD takes more circuitous and "winding" paths for bigger models -- this would mean that data requirements scale super-linearly in parameter count. There is not a lot of empirical data on this question, but my understanding is that the data so far implies that SGD doesn't tend to find long and winding optimization paths for current ML

problems.[15] This could certainly change for future ML problems, but the evidence so far makes the linear extrapolation seem like a reasonable starting point.

- It might be that SGD is effectively optimizing over some simpler *convex subspace* of the non-convex loss landscape, meaning the model that it finds could have been fully characterized by a smaller number of parameters[16] $P' < P$. If $P' = \alpha P$ for some $\alpha < 1$, then we should expect sample complexity to scale linearly in αP. If α does not depend on P for the relevant learning problem, then we should expect sample complexity to scale linearly in P. If α shrinks as Pgrows, the scaling should be sub-linear,[17] and vice versa if α grows with P.

Overall, I don't see strong theoretical reasons to believe that the worst-case number of data points required for SGD to get within ϵ of the reachable minimum loss should scale differently in the non-convex case than the convex case. My understanding from discussions with technical advisors is that machine learning researchers and engineers also tend to expect linear scaling to hold up roughly in practice -- for example, I have seen ML tutorial websites cite the "rule of 10", the heuristic that ML models need roughly 10x as much data as they have parameters to train well.[18]

If SGD works well in practice for transformative ML problems in the same informal sense that it works well in practice for image classification, game-playing, language modeling, and so on, my prior is that data requirements would scale linearly in parameter count, although I have a lot of uncertainty and consider substantially sub-linear and substantially super-linear scaling plausible. In the rest of this document, I will describe how I attempt to use empirical information to reduce uncertainty about this scaling behavior (while keeping this prior in mind).

15 McCandlish et al 2018 find that when training at the "critical batch size", the number of optimization steps has a very weak dependence on model size for a variety of different tasks. Kaplan et al 2020 (from the same team) demonstrates this again specifically for language models, over a larger range of model sizes. Both papers find that to the extent that bigger models need more data to train on, this is mostly because they need to get a less noisy estimate of the gradient, rather than because they need to take a longer path from the initialization to the final setting of parameters. I discuss both papers in more detail in this section. I am not aware of other papers that specifically study this question, although I have not done an exhaustive literature review.

16 As a simple example, suppose the reachable minimum loss is achieved at a saddle point where P' dimensions are curving upward and P - P' dimensions are curving downward. If we removed the latter dimensions, then the local loss landscape would be a convex function in P' parameters and its minimum would be located in the same place as the saddle point in the larger space.

17 See for example, this Medium post by Malay Haldar, a machine learning engineer at Airbnb, this post on the ML tutorial website fastml.com, this blog post by the ML tutorial website machinelearningmastery.com, and this Caltech lecture by Prof Yaser Abu-Mostafa. These were simply a few hits that I found on the first page from Googling "data points parameters rule of thumb" for a few minutes in April 2020; I did not conduct more thorough research because I already felt relatively confident that this was a fairly popular rule of thumb from discussions with machine learning practitioners.

18 These are the only recent papers that I am aware of which systematically estimate sample complexity scaling behavior for reasonably large state-of-the-art models; both papers cite prior empirical work, but I didn't examine that work closely because it generally seemed more out-of-date. We may do a more exhaustive literature review at some point in the future.

What if we use a different optimization algorithm besides SGD?

In most of the document, I focus on SGD because most large-scale modern machine learning mostly relies on SGD. But there are a wide variety of other optimization algorithms -- future models could be trained using alternatives such as black-box search (which uses only loss value information and does not use loss gradient information), methods that use second-order derivatives such as Newton's method, and so on.

There are certain conditions under which each of these algorithms is guaranteed to return the hypothesis that minimizes empirical risk -- in those cases, the ERM sample complexity bound could be applied directly. As with SGD, we can likely prove tighter bounds that exploit the special structure of these problems, but my understanding from a very brief discussion with our technical advisor Paul is that the dependence on P and $1/\epsilon^2$ is likely to remain.

Of course, any of these alternative optimization algorithms may be applied even when the conditions required for them to implement the ERM rule clearly do not hold, just as SGD is routinely applied to non-convex problems. In these cases, my first inclination would again be to treat these algorithms as optimizing over a smaller subspace on which they *do* implement the ERM rule, and thus start with the prior that sample complexity would scale linearly in parameter count, using reasoning analogous to the reasoning given above for SGD.

However, I have spent much less time thinking about non-SGD optimization algorithms, and I am much less confident in my judgments about them. Ideally, we would collect empirical data about the sample complexity of alternative optimization algorithms rather than relying on pure theory; in the rest of this document, I review empirical evidence for the sample complexity of SGD.

Experimental scaling laws from ML papers

Note: *Since drafting this report, I encountered another relevant scaling experiment paper, Rosenfeld et al 2019, which focuses on image classifiers. I have not read this paper yet, but technical advisor Jacob Hilton believes that the scaling behavior it finds is broadly consistent with my estimates.*

In reality, ML practitioners rarely specify a particular ϵ value to hit. Instead, they simultaneously choose a model architecture with a certain number of parameters $P^{¿}$, a dataset of size $D^{¿}$, and an objective function $L^{¿}$ such that they expect training that model on that dataset using that loss function will satisfy their real-world goal while making good use of their resources. How dataset size scales with parameter count in practice is therefore an implicit consequence of these practical choices, and might be very different from the theoretical predictions.

Recall from Part 1 that my definition of training data requirements for a transformative model refers to the amount of data that would be roughly "optimal" (according to the way researchers value different resources) for training a model with a certain number of parameters.

In this section, I'll summarize findings from two recent machine learning papers[19] which attempt to derive sample complexity scaling laws for specific ML problems using controlled experiments, both of which find that sample complexity scales according to a power law, $D \approx K P^{\alpha}$. I'll cover:

- Kaplan et al 2020 ("Scaling Laws for Neural Language Models"), which finds that the amount of data required to train generative language models scales sub-linearly with parameter count (more).
- Hestness et al 2017 ("Deep Learning Scaling is Predictable, Empirically"), which finds that sample complexity scales *super-linearly* with parameter count for a variety of supervised learning tasks including language modeling (more).
- My interpretation of these two papers and what I take away that seems relevant to the question of transformative model data requirements (more).

In Part 3, I will estimate the parameter count of a transformative model according to the Neural Network hypotheses and the Genome Anchor hypothesis, and extrapolate training data requirements from parameter count using scaling laws similar to the ones derived in these papers, especially Kaplan et al 2020.

Kaplan et al 2020: Scaling Laws for Neural Language Models

This paper comes out of OpenAI, and one lead author is Jared Kaplan, an Open Philanthropy technical advisor. It describes a suite of experiments designed to isolate how the prediction loss achieved by a transformer-based generative language model changes as model size, dataset size, and training computation are varied over multiple orders of magnitude.

Kaplan et al 2020 derives two scaling laws that are especially relevant to the data requirements question: the target accuracy law and the compute-optimal scaling law. I implement these in this sheet. The key takeaways are:

- The empirical relationship between dataset size and model size can be closely fit with power law curves for the range of model and dataset sizes tested in the paper.
- Both scaling laws find that for the language modeling problem, data requirements scale sub-linearly as a function of model size.
- These scaling laws contradict one another -- the former suggests that dataset size scales as parameter count to 0.74, while the latter suggests a scaling exponent of 0.37.

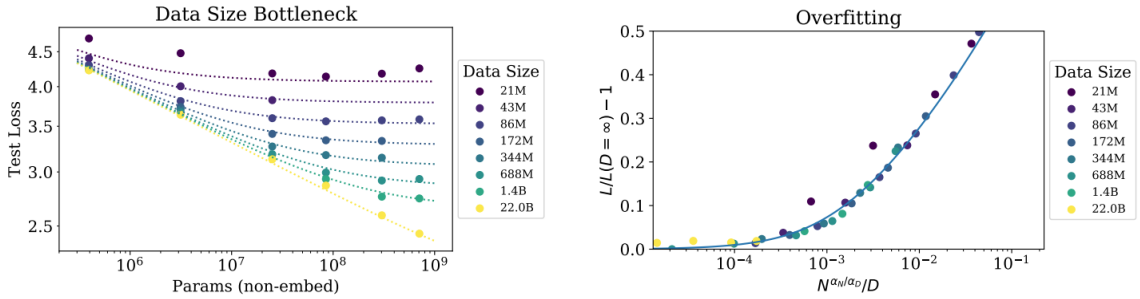
I consider Kaplan et al 2020 to be by far the most relevant piece of experimental evidence for how data requirements are likely to scale with parameter count for a transformative task.

19 Kaplan et al use N to refer to model size, but I use P for consistency with above and because it feels slightly more intuitive to me. Note that all of these parameter counts exclude the parameters used in word embedding, which turns each word into a vector that is then processed by the language model: "Model performance depends most strongly on scale, which consists of three factors: the number of model parameters N (excluding embeddings), the size of the dataset D, and the amount of compute C used for training." pg. 3.

The target accuracy law

The target accuracy law describes the minimum number of tokens $D(P,q)$ which are required to train a model with P parameters[20] to achieve the same loss that a smaller model of size qP could achieve with infinite training data (where q is some fraction such as 0.5 or 0.1).[21]

They estimate the target accuracy law by comparing test loss achieved by training on a dataset large enough to be functionally "infinite" with the test loss achieved by training on various smaller dataset sizes. They generate this data by training six different sizes of model (ranging from 1e3 parameters to 1e9 parameters) each on eight different sizes of dataset (ranging from 2.1e7 to 2.2e10 tokens), for a total of 8 x 6 = 48 different training runs:[22]



In each of these training runs, they use early stopping (i.e., they stop training as soon as the loss on a held-out test set ceased to decrease). I will refer to this as the "early-stopped test loss"; stopping training just as the loss on a test set begins to plateau roughly minimizes test loss while preventing overfitting and ensuring that larger models always achieve a lower test loss.[23] Using the data from the charts above, Kaplan et al fit a function $L_{early}(P,D)$ describing the early-stopped test loss; this function is a power law in both P and D.

No overfitting or plateau was observed for the yellow line (22B tokens) for models below 1B parameters, so this was taken to be representative of the "infinite" data limit for those smaller models.[24] For each model size P below 1 billion and dataset size D that was tested, we can use this assumption to estimate $P' < P$, the smaller model size such $L_{early}(P,D)$ is equal to

20 Note that the formulation involving q is not the one originally given in the paper; I discuss this in more detail in this appendix.
21 See the left half of figure 9 on pg. 11 (reproduced above). The eight different colored lines correspond to the eight different dataset sizes, and the six points plotted on each line represent the six model sizes.
22 "Since we stop training early when the test loss ceases to improve and optimize all models in the same way, we expect that larger models should always perform better than smaller models...We regularize all our models with 10% dropout, and by tracking test loss and stopping once it is no longer decreasing." pg. 11.
23 "To chart the borderlands of the infinite data limit, we can directly study the extent of overfitting. For all but the largest models, we see no sign of overfitting when training with the full 22B token WebText2 dataset, so we can take it as representative of D = ∞." pg. 11.
24 See this sheet for the calculations.

$L_{early}(P', 2.2 \times 10^{11})$, the loss the smaller model achieves on the functionally "infinite" 22B token dataset.

This, in turn, can be used to derive a function for dataset size $D(P, q)$ in terms of P and $q = P'/P$ which describes how much data would be needed to train a model of size P to achieve the same loss as a model of size $P' = qP$. Kaplan et al show that $D(P, q)$ has the form $K(q)P^{0.74}$: it is a power law function in which the exponent is 0.74 and the constant factor is a function of q.

Note that $K(q)$ quantifies a tradeoff between model size and training data: setting q closer to 0 means training a larger model on a small amount of data, and setting q closer to 1 means training a smaller model on relatively more data. If we choose to minimize training computation, then $q \approx 0.11$ and $K(q) \approx 288$;[25] I will use this value going forward. (See this appendix for more details.)

The compute-optimal training law and the contradiction

While Kaplan et al don't attempt to choose the compute-optimizing q in the overfitting scaling law, they do generate a scaling law for compute-optimal training using an alternative set of experiments (details in this appendix):

$$D_{opt}(P) \approx 4.7 \times 10^6 \times P^{0.37} \text{ tokens}$$[26]

As you can see, this is in direct contradiction to the equation above which describes the minimum amount of data needed to achieve a target loss. The target loss scaling law predicts that at least $D_{min}(P, q = 0.11) \geq 288\, P^{0.74}$ are needed to achieve the same loss as the best reachable loss of a model ~11% the size, and that $q \approx 0.11$ approximately minimizes the use of computation for a given target loss. The two equations reach a crossing-over point at ~1e12 parameters, at which point compute-optimal training according to the $D_{opt}(P)$ scaling law is using fewer data points than the minimum required according to the $D_{min}(P, q)$ scaling law.

I consider the target accuracy scaling law to be more credible than the compute-optimal scaling law because it was derived from a simpler experimental procedure with less room for error, and is more consistent with both theoretical predictions and the observational evidence from RL models that I discuss below. I discuss the comparison between these two scaling laws and the contradiction in more detail in this appendix.

25 Language modeling also happens to give the tightest fits of the problems Hestness et al studied: "LM learning curves and model size scaling relationships are the most robust; word and character language models show clear and predictable power-law learning curves, and the power-law exponents tend to be small." pg. 6.

26 The general procedure is described in more technical detail on pg. 4 of the paper.

Hestness et al 2017: Deep Learning Scaling is Predictable, Empirically

This paper comes out of Baidu AI research. It describes experiments deriving scaling laws for four different supervised learning or generative modeling problems (neural machine translation, language modeling, image classification, and speech recognition). I will focus on analyzing the language modeling results of Hestness et al 2017 because they are directly comparable to the results in Kaplan et al 2020.[27] I implement the scaling laws Hestness et al derived for language modeling in this sheet. Their procedure was roughly as follows:[28]

1. Generate random subsets of a vocabulary-restricted version[29] of the One Billion Word language modeling dataset ranging in size from ~1e6 to ~4e8 words,[30] plus a held-out test set.
2. For each dataset size D_i:
    a. Generate a set of candidate models of increasing size $M_{i,1}, M_{i,2}, M_{i,3}...$
    b. Train the candidate models on the dataset of size D_i with no dropout or other regularization, until the loss the model achieves on the test set *either* flattens out and stops changing (because the model has saturated its capacity) *or* starts to increase (because the model has started to overfit to the training dataset).
    c. Search for the **smallest** candidate model $\widehat{M}_i$ such that the test loss starts to increase with marginal epochs rather than continuing to stay flat.
3. Fit a scaling law $P_{max}(D) \approx K \times D^{\alpha}$ to the data points $(D_i, \widehat{M}_i)$ which describes the smallest model that "barely overfits" a dataset of size D.

Comparison to Kaplan et al 2020

Critically, the goal of this paper is very different from the goal of Kaplan et al 2020.

Kaplan et al aim to find a scaling law that describes how much data would be required for a model of size P to achieve the same test loss that a smaller model of size qP would achieve with infinite data, *assuming that training is always stopped early so that no model overfits and larger models always achieve lower loss than smaller models*.

On the other hand, Hestness et al don't try to avoid overfitting because they aim to find the smallest model size that can "barely overfit" a given dataset -- or equivalently, **the largest model size that does not overfit to the dataset even if trained with infinite epochs.** Accordingly, Hestness et al specifically sought to avoid all regularization because this would

27 "To reduce the computational requirements of the models, we restrict the vocabulary to the top 10,000 most frequent words in the Billion Word Dataset." pg. 7.
28 "We train the models on shards ranging from 0.1% up to 40% of the Billion Word Dataset." pg. 7.
29 "Next, we aim to understand the importance of model capacity to fit a training set, so we control our experiments by removing regularization schemes that might reduce the model's effective capacity (e.g., weight decay)." pg. 4.
30 "We regularize all our models with 10% dropout, and by tracking test loss and stopping once it is no longer decreasing." Kaplan et al 2020, pg. 11.

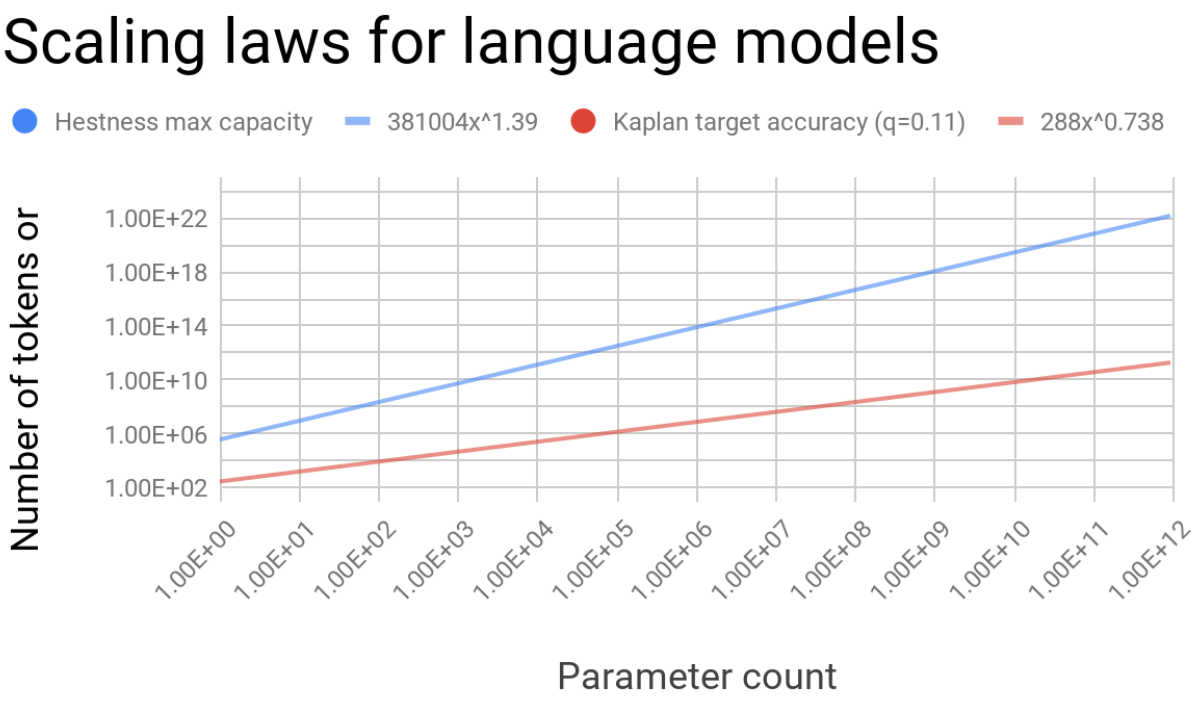
diminish model capacity,[31] whereas Kaplan et al regularize not only with early stopping but also dropout.[32]

Hestness et al find that the smallest model size $P(D)$ that barely overfits a dataset of size D scales as $D^{0.72}$. Inverting this, that means that the largest dataset size $D(P)$ that a model of size P can barely overfit scales as $P^{1/0.72} = P^{1.39}$ -- this is highly super-linear scaling. The scaling laws for other learning problems that Hestness et al derive are qualitatively similar, with scaling exponents ranging from ~1.28 to ~1.75. On the other hand, Kaplan et al find sub-linear scaling. The table below summarizes the differences between the language model scaling behaviors according to the two papers:

	Hestness et al 2017	**Kaplan et al 2020** (target accuracy law)
Form	$D_{max}(P) = K' \times P^{\beta}$	$D_{min}(P, q) = K(q) \times P^{\alpha}$
Exponent	1.39	0.738
Constant[33]	381,000	Parametrized by q. To optimize training computation q = 0.11 and K(q) = 288.
Procedure	On a fixed dataset, train small models of increasing size with no regularization to find the smallest model for which test loss starts to get worse with marginal epochs through that dataset. Repeat for different dataset sizes. Using this, estimate P(D) -- how the smallest model that overfits a dataset scales with dataset size. Invert to find D(P).	Train every model on every dataset size with 10% dropout; when training large models on small datasets use early stopping to ensure that it doesn't overfit. Taking the largest size to represent "infinite data", estimate q(P, D) -- the fraction q such that a model of size qP trained on "infinite" data achieves the same loss as a model of size P trained on D data points with early stopping. Rearrange to find D(P, q).
Meaning	**Maximum model capacity:** The smallest dataset size D(P) such that when a model of size P is trained on D(P) data points, its unregularized test loss will never get worse even in the limit of infinite epochs.	**Minimum data for a target loss:** The dataset size D(P, q) such that if a model of size P is trained on D(P, q) data points with early stopping, it will achieve the same test loss as the best loss that a model of size qP would achieve with infinite data.

I charted the two scaling laws side by side here:

31 The constant factor is less inherently meaningful than the exponent, depending on how the raw text is broken up into tokens. The tokenization scheme is likely slightly different for the Billion Word benchmark used in Hestness et al 2017 and the WebText2 dataset used in Kaplan et al 2020, so they are not strictly comparable. However, tokenization differences do not change the exponents, and in practice I expect them to make only a modest difference to the constant factor as well.
32 "The precise numerical values of [the constant factors] depend on the vocabulary size and tokenization and hence do not have a fundamental meaning." Kaplan et al 2020, pg 5.
33 "Strikingly, although these model architectures differ appreciably, they all show the same learning curve profile characterized by the power-law exponent." Hestness et al 2017, pg. 7.



I think the question that Kaplan et al ask is substantially more relevant for thinking about training data requirements for a transformative model (as operationalized in Part 1) than the question Hestness et al ask.

It is generally not relevant whether a model trained without any regularization measures would start to overfit to a dataset in the limit of infinite epochs, because in practice most ML training runs are heavily regularized with early stopping, dropout, normalization, and other techniques (see this appendix for details). Training a small model past the point of saturation with no regularization would minimize the size of the trained model for a given target loss, but only at a major cost in terms of dataset size, training computation, and wall-clock time. This is rarely a worthwhile tradeoff compared to training a somewhat larger model with substantially less data, computation, and time (potentially compressing the trained model later if model size is crucial for the application).

With that said, I am unsure how much of the difference between these two scaling laws is due to the fact that Hestness et al do not regularize the models they train and/or train them for more epochs than is generally practical. Some of the difference may instead be attributable to other aspects of the training procedure, or various details of the model architectures or datasets. I don't know of a simple way to estimate what the Hestness et al 2017 experiments would imply about the number of data points required to reach a target accuracy, or what the Kaplan et al 2020 experiments would imply about the smallest dataset that a model could never overfit. Without a systematic comparison reconciling the two results, I am inclined to put some weight on both sources of evidence.

Summary and interpretation of experimental evidence

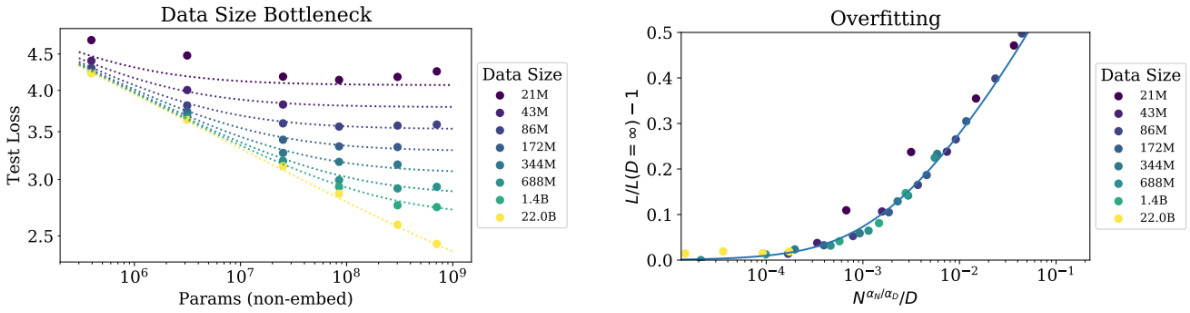
We are trying to glean information about how much data might be required to train a model on some unknown future learning problem that would have a transformative impact at a certain threshold of performance. While it's not appropriate to directly apply the specific scaling laws derived in Kaplan et al 2020 or Hestness et al 2017 for other ML problems, these scaling laws could help calibrate our intuitions about transformative model data requirements.

Absent other evidence, it is plausible that data requirements for unknown future problems scale in a broadly similar way to data requirements for problems that *have* been systematically studied (with the caveat that the possibility of selection bias means we should not necessarily give this empirical evidence strong weight relative to theoretical considerations). In this section, I summarize the high-level qualitative expectations that I take away from these two papers:

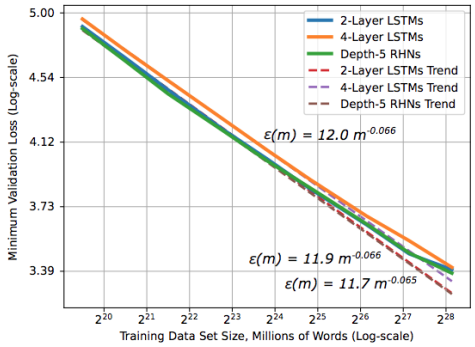
- Transformative model data requirements may be well-expressed as a simple power law function of model size (more).
- Highly super-linear and highly sub-linear scaling both seem possible, although sublinear scaling seems more likely (more).

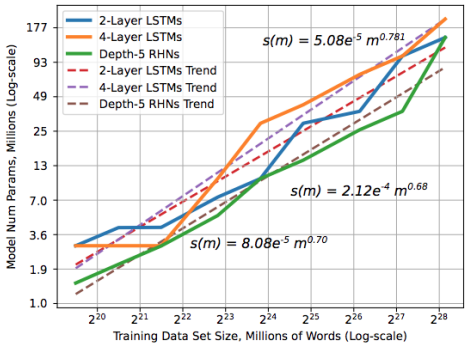
Scaling may be governed by a simple power law in the relevant regime

All of the key empirical results in both papers imply that the very simple functional form of $D(P,q)=K(q)\times P^{\alpha}$ can describe how dataset size scales with model size, at least over the range of model sizes tested. These are the fits for the experiments run in Kaplan et al 2020 to generate the target accuracy law:



These are the fits for the experiments run in Hestness et al 2017 for word language models (the closest analogue to the ML problem studied in Kaplan et al 2020):





Goodness of fit is not quantified in the papers, but we can see from visual inspection that these fits are fairly tight over a wide range of model sizes and/or dataset sizes; my understanding is that empirical fits this clean are relatively unusual. Furthermore, both papers find that the *exponent* of the power law is a function only of the parameter count and the data distribution, whereas the constant is dependent on factors like the tokenization,[34] the target accuracy (see above), and the model architecture.[35]

My understanding is that there is also some additional evidence that power law functions are an appropriate way to model empirical sample complexity scaling behavior:

- Hestness et al 2017 claims that theoretical work supports this functional form,[36] providing a literature review of the theory which I have not examined.
- Both papers cite related empirical work which demonstrates power law scaling for various ML problems, including for other architectures besides neural networks. I have not read any papers deriving empirical scaling laws other than these two because my understanding is that the rest of the literature is less current and studies smaller models.
- In Appendix D.7, Kaplan et al report that they experimented with a number of functional forms, and found that power law fits were more accurate than other functions.[37] However, I do not know which other functional forms they tried, or what the full set of *a priori* reasonable functional forms would be.

34 “Many studies theoretically predict that generalization error "learning curves" take a power-law form, ε(m) ∝ alpha * m^βg. Here, ε is generalization error, m is the number of samples in the training set, α is a constant property of the problem, and βg...is the scaling exponent that defines the steepness of the learning curve—how quickly a model family can learn from adding more training samples.” pg. 2. If the generalization error scales as a power law in sample complexity, model size is also likely to scale as a power law because the latter is derived from the former.

35 “We experimented with a number of functional forms for the fits to L(N), L(C), and L(D); the power-law fits were qualitatively much more accurate than other functions such as logarithms (see Figure 23).” pg. 26.

36 Kaplan et al 2020 cites this textbook as an introduction to the ways in which power laws can arise naturally in complex systems; I have not read it.

37 A pure power law function is not capable of expressing a linear relationship with non-zero intercept, f(x) = mx + b. However, this is not an issue because we know that a model with 0 parameters (i.e. an ordinary computer program) requires 0 data points to achieve its optimal loss.

More generally, my understanding is that the power law $f(x)=Cx^{\alpha}$ is a very simple and basic type of relationship between two variables that describes a wide range of phenomena reasonably well, and can arise due to a number of possible underlying dynamics.[38] (This is similar to the ubiquity of Gaussian distributions in modeling noise.) The power law fit is in a small class of functions with a similar level of simplicity and practical significance, making me more inclined to privilege it as a prior:

- More complex functional forms would seem relatively unmotivated given our ignorance about the dynamics of a potentially transformative ML problem, and ML training in general.
- Other similarly simple and ubiquitous functional forms seem less appropriate, given what we do know. In addition to the theoretical and empirical evidence cited in the two papers, it is important that the power law is capable of representing the linear relationship[39] $f(x)=mx$. This is the relationship between sample complexity and parameter count predicted by the ERM bound, with the slope m being roughly equivalent to the signal-to-noise ratio of the ML problem.

Note that smooth power law scaling likely only applies within a certain regime of model sizes for any given learning problem, but this is likely not an issue for using a power law functional form to estimate transformative model data requirements (more in this appendix).

Highly super-linear and highly sub-linear scaling are both possible

In Kaplan et al 2020, both the target accuracy law (exponent of 0.74) and the compute-optimal training law (exponent of 0.37) predict that the number of tokens required to train language models scales substantially sub-linearly in the parameter count, while Hestness et al 2017 predicts scaling with an exponent of ~1.39.

As I explained above, I consider the results of Kaplan et al 2020 to be more informative than the results of Hestness et al 2017 because the latter employs no regularization, but I am uncertain how much of the difference between the two results can be attributed to this, and place some weight on the possibility that Hestness et al 2017 is more representative of how sample complexity would scale for a transformative model.

Below, I generate a subjective probability distribution over the scaling exponent after incorporating observational evidence from RL training times.

38 These were the StarCraft agent AlphaStar (Vinyals et al 2019), two versions of the DOTA agent OpenAI Five (Berner et al 2019), the ShadowHand agent trained on Rubik's Cube (OpenAI et al 2019), two versions of a Capture-The-Flag bot (Jaderberg et al 2019), the Hide and Seek agent (Baker et al 2020), a DQN agent trained on Atari games (Mnih et al 2015), an IMPALA agent trained on Atari games (Espeholt et al 2018), the poker agent DeepStack (Moravcik et al 2017), three Hanabi agents (Bard et al 2019), and two versions of the Go agent AlphaGoZero (Silver et al 2017). To save time, we chose only ~1-3 "representative results" per distinct RL problem, e.g. we did not include the earlier Go results like AlphaGo Lee or the additional Atari results besides IMPALA and DQN.

39 Kaplan et al 2020, discussed below, may be evidence for this.

Effective horizon length

So far we have discussed *sample complexity*, i.e. the number of independent "samples" we would need to draw from a specified data distribution D (such as WebText or ImageNet) to train a model with a certain number of parameters P. This does not directly estimate the number of *subjective seconds*[40] *of data* that would be required to train a model of size P, and the translation between "number of samples" and "subjective seconds" may be very different for different machine learning problems.

I'll define the **"effective horizon length" of an ML problem as the amount of data it takes (on average) to tell whether a perturbation to the model improves performance or worsens performance**. If we believe that the number of "samples" required to train a model of size P is given by $K P^{\alpha}$, then the number of subjective seconds that would be required should be given by $H \times K P^{\alpha}$, where H is the effective horizon length expressed in units of "subjective seconds per sample."

This is an informal and somewhat fuzzy definition. Pinning it down precisely would require specifying how large a perturbation we are considering (e.g. "randomly perturbing all parameter values by 0.1%") as well as how confident we would like to be about which of two different models is higher-performing (e.g. "⅔ probability of choosing the correct model"). Furthermore, the effective horizon length should likely increase over the course of training -- as the model gets closer to optimal, there is less room for improvement so it should take more data to tell whether a particular tweaking of parameter values slightly improves performance.

However, if we make an assumption about the effective horizon length for one *particular* ML problem, we can use that as a reference point for thinking about the effective horizon length of other problems. I will be using the problem of "predict the next token of text given the context so far" (which GPT-3 was trained on) as my reference ML problem. **I will assume that the effective horizon length for the next-token prediction problem is simply 1 token**. At average human reading speeds, this would be roughly **~¼ of a subjective second.**

This assumption can give a starting point for thinking about the effective horizon length of other ML problems. In other words, we can ask "In this ML problem, how much data does it take to get 'about as much useful information' about whether a perturbation to the model was helpful for performance as one token provides in the next-token prediction problem?" This is still a fuzzy and incomplete definition,[41] but it seems possible to get a rough order-of-magnitude sense using qualitative heuristic arguments. I think it is easiest to illustrate this in the context of some examples. In the rest of this section, I will discuss effective horizon length in the context of:

- Supervised learning (more).
- Reinforcement learning (more).

40 The optimal value of q is a constant which does not depend on P.

41 This scaling exponent is quite close to the scaling exponent in the target accuracy law from Kaplan et al 2020, which I find somewhat encouraging.

- A strategy that involves training a generative language model (or other type of generative model) to predict what it will see next, but *evaluating its performance* on downstream tasks such as translation or summarization, potentially with fine-tuning (more).

For a supervised learning problem

Suppose we train a supervised learning model on a hypothetical dataset of **binary advice questions.** Each binary advice question describes a context and two proposed courses of action A and B, and asks which is better; all of the answers are simply one character, either "A" or "B." The context is intended to give a generally intelligent and knowledgeable human the information they need to get up to speed on the question, so it could be arbitrarily long: general knowledge questions could be zero context, relationship advice could be a couple of pages of context, complex military strategy questions could be hundreds of pages of context, and so on.

Suppose we train a model that can process sequences of text (such as a transformer)[42] on a large dataset of binary advice question and answer pairs. In contrast to a model such as GPT-3 which attempts to predict *every word*, this model would simply process the entire question and then attempt to predict whether the answer is "A" or "B." While it must perform a forward pass for every word that it processes, it will only receive a loss signal once per *question* rather than once per *token*.

I would predict that the compute-optimal number of tokens required to train this model will scale linearly with the length of the question -- if we performed experiments exactly like the Kaplan et al 2020 scaling experiments for this supervised learning problem, I believe we would find that doubling the average question length in the dataset roughly doubles the optimal number of tokens required to train a model of a fixed size.[43] **I would guess that the "effective horizon length" of this problem is the length of the average question**.

More generally, since the notion of a "sample" is well-defined for supervised learning, it seems like it would make sense most of the time to simply assume that the "effective horizon length" is equivalent to the amount of data in the average sample. For example, I would consider the "effective horizon length" of an ImageNet model to be "one image" (which is a fraction of a subjective second), although the concept is mainly useful for ML problems that involve sequential data such as text, code, video, audio, game-playing, and navigating the physical world.

For a reinforcement learning problem

In this section, I will:

42 Recall that one "subjective second" of data corresponds to the amount of information of a certain type (e.g. words, sounds, images) that a human brain could process in one second. For example, the average human can read ~3-4 words per second, so one "subjective second" of language data would be ~3-4 words.
43 See this appendix for a simple explanation of how transformers work.

- Briefly cover the formal definition of a reinforcement learning (RL) problem (more).
- Describe how I estimate the effective horizon length of an RL problem in theory, using the discount rate (more).
- Discuss how well the concept applies to RL training runs in practice (more).
- Explain the functional form, $H \times K \times P^{\alpha}$ which I will be using to estimate the number of timesteps required to train an RL model (more).

Formal description of a reinforcement learning problem

Reinforcement learning problems are often formally described as Markov Decision Processes (MDPs), in which an agent must maximize its expected sum of rewards in a stochastic environment. An MDP consists of:

- A set of possible *states* $s \in S$ that the environment could be in.
- A set of possible *actions* $a \in A$ available to the agent.
- A *transition function* $T(s' \vee s, a)$, a probabilistic function which takes as input a state and an action taken by the agent in that state, and outputs a probability distribution over the possible subsequent states of the environment. The transition function of an MDP must satisfy the Markov property: it must be the case that the probability of a subsequent state depends *only* on the current state and action (rather than any past states or actions). This means that the states must be rich enough to capture all possibly-relevant information from the past.
- A *reward function* $R(r \vee s, a, s')$, a probabilistic function which takes as input a state, action, and next state, and outputs a real-valued *transition reward* that the agent receives[44] when it transitions from state s to state s' after taking action a. (The distribution over rewards the agent can receive conditional taking action a on state s is therefore $R(r \vee s, a, T(s' \vee s, a))$, incorporating uncertainty about the transition function.)
- A *discount factor* γ indicating the agent's time-preference.

The agent must implement a *policy* $\pi(a \vee s)$, a probabilistic function which takes as input the current state and outputs a probability distribution over actions to take in that state. The agent's goal at any given timestep t_0 is to choose actions a_t which maximize its expected discounted sum of rewards r_t over all possible subsequent states s_{t+1}:

$$E[R(\pi) \vee s_{t_0}] = \sum_{t=t_0}^{\infty} \sum_{a_t \in A}^{\square} \pi(a_t \vee s_t) \sum_{s_{t+1} \in S}^{\square} \gamma^{t-t_0} T(s_{t+1} \vee s_t, a_t) R(r_t \vee s_t, a_t, s_{t+1})$$

Most RL problems in modern practice are episodic. This means that there are one or more *terminal states* $S_{END} \subseteq S$ such that when the environment transitions into a terminal state, it never transitions out: $s \in S_{END} \Rightarrow T(s' \vee s) = 0, \forall s' \in S$. In a game, terminal states are generally "win", "lose", and "draw" states. An "episode" consists of the complete sequence of states,

44 More specifically, I expect that the scaling behavior will be similar to the scaling behavior Kaplan et al 2020 finds, except that the relevant unit will be "one question" rather than "one token." That is, I would guess that we would need ~300 * P^0.74 *questions* in the dataset.

actions, and rewards (s_t, a_t, r_t) from a starting state to a terminal state (e.g. one game of chess). An *episode transcript* $\tau = \{s_0, a_0, r_0, s_1, ... s_{T-1}, a_{T-1}, r_{T-1}, s_T\}$ is the complete sequence of states, actions, and transition rewards that occur within a particular episode.

Different episodes are fully independent of one another: the agent retains no memories from one episode to the next, and the agent's actions in one episode do not affect how a subsequent episode plays out.

Deriving effective horizon length of an RL problem from discount rate

Because it would usually make sense in the context of supervised learning to estimate the effective horizon length as the average length of one sample, naively it might be reasonable to define the effective horizon length of an RL problem to be the average length of one episode, since different episodes are independent from one another just as samples are in supervised learning.

However, I think that average episode length would usually overestimate the amount of data required to tell whether a perturbation to the model improves performance in an episodic RL problem (and it doesn't help generate an estimate for non-episodic problems). The agent could be receiving a reward r_t at every timestep, such that a single episode may consist of hundreds or thousands of reward signals. While I don't think it makes sense to treat each such reward as its own "sample", I also don't think we should count one episode as equivalent to one "sample" regardless of how much useful information is contained in the intermediate rewards.

I believe the discount rate γ can help to crudely quantify the value of these intermediate rewards, if we assume that the discount rate was set roughly optimally for the problem in question (more on that below). In RL, the optimization algorithm is selecting for a policy which maximizes the expected *discounted sum of future rewards* from a randomly selected[45] state. If $\gamma = 1$, the agent values a reward it may receive arbitrarily far into the future just as much as a reward it may receive in the current timestep, and the optimization algorithm is simply finding a policy that maximizes the expected sum of rewards over an entire episode. In this case, it seems that the effective horizon length should be the average length of an episode,[46] in accordance with the naive intuition. If $\gamma = 0$, the agent only values reward in the current timestep, and the optimization algorithm is simply finding a policy which maximizes immediate reward. In this case, it seems that the effective horizon length should probably be 1 timestep.

45 Many problems of interest may have *sparse* rewards, meaning that the reward function outputs 0 for most possible state, action, next state triples.

46 The formal specification of reinforcement learning does not directly specify any single "data distribution" (such as ImageNet) from which observations and rewards are drawn -- there is simply an environment which produces observations and rewards in response to the agent's own actions. By "randomly selected timestep", I mean a state drawn from the distribution of states that are generated by running the optimal policy pi_star itself -- that is, a state selected uniformly at random from the full list of states that an agent running pi_star visited over the course of a large number of episodes. It is necessary to define it this way because two different policies pi_j and pi_k will end up visiting the same state in the MDP with different probabilities.

Prima facie, the more important distant future rewards are to overall performance (i.e. the closer γ is to 1), the more time we should expect it to take to tell whether a perturbation to the model is an improvement. I expect that **for $0<\gamma<1$, the expected horizon length can be approximated by the *minimum* of average episode length and $1/(1-\gamma)$.**

The goal of RL is to produce a policy $\pi^{¿}$ which maximizes the discounted sum of future rewards from a randomly selected[47] timestep t_0: $\pi^{¿} = arg\,max_{\pi} \sum_{t=t_0}^{\infty} \gamma^{t-t_0} r_t$. It turns out that we can express the discounted sum of future rewards as a weighted average of reward sums over increasingly long timescales. If the expected episode length is infinite, the sum looks like this:

$$\sum_{t=t_0}^{\infty} \gamma^{t-t_0} r_t = \sum_{H=1}^{\infty} \gamma^{H-1}(1-\gamma)¿¿¿$$

The right hand side is an infinite sum of the form $p_0 r_{t_0} + p_1(r_{t_0}+r_{t_0+1}) + p_2(r_{t_0}+r_{t_0+1}+r_{t_0+2})+\ldots$ where the p_i sum to 1. We can potentially think of this as the **weighted average of undiscounted performance** on a horizon-1 subproblem, a horizon-2 subproblem, a horizon-3 subproblem, and so on, starting from a randomly selected point in time t_0. The average horizon length across these subproblems would then be given by

$$\bar{H} = \sum_{H=1}^{\infty} \gamma^{H-1}(1-\gamma) H = 1/(1-\gamma)$$

Note that we assumed an infinite sum in the derivation above. If the expected episode length is finite, the equation for the probability distribution is slightly more complicated, but $1/(1-\gamma)$ remains a very good approximation of the average horizon length as long as the average episode length is substantially larger (e.g. >2x) than the value of $1/(1-\gamma)$.

In general for practical RL applications, γ can be set such that $1/(1-\gamma)$ is substantially smaller than the expected episode length to speed up training (see below). This implies that across a large number of episodes,[48] we should expect to receive the equivalent of one "independent

47 As above, by "randomly selected timestep", I mean a state drawn from the distribution of states that are generated by running the optimal policy pi_star.

48 Note that because it is important for the policy to perform well at a randomly-selected timestep drawn from a typical episode, getting the very *first* "independent signal" of a policy's quality will require running one full episode and selecting a timestep t0 at random from that episode. However, if the policy is simply continuously running, starting a new episode once the current episode episode ends, we can get an unbiased estimate of performance with the following procedure:

1. Draw a horizon H from the probability distribution P(H) = gamma^(H-1)(1-gamma).
2. Pause the continuously-running policy and grab the most recent H rewards.
3. Take the sum to be the unbiased estimator of performance.

The expected value of H will be 1/(1-gamma) and the overhead required to run the policy at all will shrink as more estimates are collected. Real-world RL training runs generally do work by periodically grabbing excerpts of a transcript from a continuously-running policy; see this appendix for more detail.

signal" about an RL policy's[49] performance roughly once every $\bar{H} = 1/(1-\gamma)$ timesteps on average.

This argument isn't fully formal and relies substantially on intuition; I suspect there is something more refined and mathematically satisfying that could be said about the notion of horizon length and **would be curious for technical work which explores this concept further and attempts to pin it down.**

Researchers strategically select the discount rate and training strategy

Using the discount rate to estimate effective horizon length of an RL problem works best if the discount rate and the overall training algorithm or strategy is set strategically to be roughly optimal for the higher-level task that ML researchers are trying to solve with the RL problem. In this section, I'll describe what I mean by this.

In most real-world RL training runs, there is no "discount rate" intrinsically baked into the ultimate task. Researchers are trying to accomplish goals like "Produce a model that beats human professionals at StarCraft"; they must *design* an RL environment and choose an appropriate discount rate such that optimizing for performance in that environment using that discount rate is likely to produce a program that accomplishes this goal.

Sometimes there will be a simple "naive" RL problem corresponding to a particular high-level task. For example, if the task is to achieve a high win rate on a zero-sum game like chess, DOTA, or StarCraft, the natural way to directly encode what we care about into a reward signal is something like "+1 when the agent wins, -1 when it loses, and 0 when a game ends with a draw." An agent trained with vanilla policy gradients to maximize expected total reward over an episode with no discounting (i.e. $\gamma = 1$) using the win/lose/draw reward signal is optimizing for precisely what we want, but it seems like it would take an unnecessarily large amount of training experience / computation to train the model with such a sparse reward signal.

Often, it is possible to get a good win rate in practice by taking various measures to improve on the naive training strategy, such as by:

- Using an actor-critic training method, in which one model (the "critic") is trained to predict the value of being in a certain state (i.e., the expected probability of winning from that state), and another model (the "actor") is trained to optimize for the critic's assessment of value.[50] I have attempted to informally explain how these methods work in this appendix.
- Introducing **proximate reward signals** while bringing the discount rate closer to 0. For example, the OpenAI Five DOTA bot was receiving some form of intermediate reward signal essentially every timestep of play during training, providing points for various metrics such as[51] "[money], kills, deaths, assists, last hits, and the like."[52]

49 In real RL training runs, a fixed policy is generally not run for a large number of episodes; the policy is being frequently updated with SGD as it is running. However, I don't expect this to make a big difference to the bottom line.
50 OpenAI Five blog post.

If it is possible to solve a given task without displaying much sophisticated long-term behavior,[53] then researchers can get away with using steep discount rates to reduce the total timesteps of training. Similarly, discount rates can be steep if researchers can design an RL problem with a very useful proximate reward signal, or if it turns out to be simple for a "critic" to learn to predict the long-term reward very well in an actor-critic model. On the other hand, if it's not possible to solve the task without displaying sophisticated long-term behaviors such as learning or planning, and it's hard to design good proximate reward signals that capture those behaviors, researchers will need to use sparser rewards and shallower discount rates, and accept the resulting longer training times.

When training large RL models, I will assume that researchers are generally attempting to use the steepest discount rates they can while still solving the practical task they set out to solve, and make sure of strategies such as actor-critic methods to extract more information from each piece of data. This means that the $1/(1-\gamma)$ estimate using the discount rate that researchers choose in an RL problem they design is likely to roughly track the the longest timescale of behaviors that are actually important for the agent to learn given the available budget.

The assumption that γ is chosen "optimally" is more likely to be true the more that computation is an important constraint, because researchers may not be sufficiently incentivized to bring timestep complexity down if training computation is a tiny fraction of the overall cost of the project to begin with.

Functional form for extrapolating "timestep complexity" of an RL problem

We can define **"timestep complexity"** in the same way we defined sample complexity in the discussion of the Kaplan et al 2020 paper above: the "timestep complexity" of an RL problem (S, A, T, R, γ) (together with a number of parameters P and desired accuracy q) is the number of timesteps of interaction with the environment that would be required to train a model of size P to achieve the best discounted sum of rewards that could be achieved by a smaller model of size qP on the same problem in the limit of infinite training timesteps.

Other things being equal, **I would expect timestep complexity to scale linearly with effective horizon length**, where "effective horizon length" is defined as the minimum of the average episode length and $1/(1-\gamma)$.

Consider an RL environment ENV, defined by a set of states S, a set of possible actions A, a transition function $T(s' \vee s, a)$, and a reward function $R(r \vee s, a, s')$. Suppose we want to train the same model θ on two different RL problems from ENV that differ only in their discount rates: $\gamma_1 = 0.99$ for the first RL problem, and $\gamma_2 = 0.999$ for the second RL problem. *Prima facie*, I would

51 For example, several commenters (e.g. game AI researcher Mike Cook in this tweet) have argued that the DOTA agent seems weak on big-picture strategy (known as "macro") but was nonetheless able to defeat a professional team largely because of its superhuman reflexes (known as "micro").
52 Note that this may or may not involve acting in a rich simulated world; it could also involve fine-tuning language models with human feedback, as in OpenAI's recent work on summarization.
53 Brown et al 2020, the GPT-3 paper, quantifies this effect for many downstream tasks.

expect that the timestep complexity would be ten times longer for the second problem, because I expect that it would require a similar number of "independent performance signals" to train on both problems, and it takes ten times longer[54] (in subjective time) to collect each signal if $\gamma = 0.999$ than if $\gamma = 0.99$.

This is a very loose approximation -- even if everything else about an RL problem is held fixed, a tenfold increase in $1/(1-\gamma)$ may not translate to a precisely tenfold increase in timestep complexity:

- It could be the case that even though the optimization process is *theoretically* intended to converge toward a policy which maximizes the discounted sum of future rewards, the policy that SGD *actually* converges toward is different. SGD might always produce a policy that effectively only optimizes ~10 timesteps into the future, even if the discount rate during training was set well above 0.9. In that case, the timestep complexity when $\gamma = 0.99$ may be quite similar to the timestep complexity when $\gamma = 0.999$.
- It could be the case that the sum of total reward over longer timescales has lower variance than the sum of total reward over shorter timescales for this problem. This would mean that even though it takes ten times as long to get an unbiased estimate of performance with γ_2, each such signal is less noisy than the analogous signal using γ_1 and it takes fewer of them to learn the right behavior.
- Conversely, it could be the case that total reward over longer timescales has *higher* variance, so timestep complexity is much *more* than ten times worse using γ_2 (or perhaps learning can't get off the ground at all with a discount rate of 0.999 because it introduces too much noise).[55]

In general, real-world reinforcement learning is substantially more messy and complicated than supervised learning, and a whole host of factors can make learning difficult and affect realized timestep complexity.[56]

With that said, the basic intuition that "effective horizon length" for RL problems should be defined as above seems reasonably sound to me. In that case, "timestep complexity" should simply be "sample complexity" times the effective horizon length. Given this, **I make the structural assumption going forward that timestep complexity can be modeled with the functional form** $T(P;H,q) = H \times K \times P^{\alpha}$, where H is the effective horizon length of the problem, P is parameter count, and q describes the target accuracy. This combines the

54 As in the original definition, there are two major ambiguities that would need to be pinned down to make it totally precise: what does "reasonably confident" mean, and what does "small perturbation" mean? You can imagine an arbitrary desired confidence level (such as being able to correctly determine which model is higher-performing ⅔ of the time, or ¾ of the time) or perturbation size (such as shifting each weight by 0.1%).

55 The concept of an "epoch" doesn't strictly apply to reinforcement learning, because there is no fixed "training dataset" in an RL problem; the agent generates the experience it trains on through interaction with an environment. However, "experience replay" is structurally analogous

56 To take an average, the distribution over episode lengths is defined (as above) by the distribution of states that are generated by running the optimal policy. If the expected episode length is infinity, that means that the discounted sum of future rewards is infinite when gamma = 1, and effective horizon length should also be infinite.

assumption that the average length of a "sample" is given by the horizon length, and that sample complexity for RL will scale according to a power law function, as in supervised learning and generative modeling.

For generative models which transfer to downstream skills

I made the assumption above the effective horizon length *for the next-token prediction problem* in particular is 1 token. However, most of the time we don't intrinsically care about the model's performance on "predicting the next token" *per se*, but rather whether it can use language to do useful things such as question-answering, summarization, translation, programming, etc.

Let's call these the model's **downstream skills**; ability to predict the next token is usefully correlated with many downstream skills. We may evaluate downstream skills in several ways:

- "Zero-shot", i.e. just presenting the language model with an instance of the task with no additional information. For example you might simply type a question into the model's context window, such as "What does it mean if my air conditioner is leaking?" The model would respond by attempting to predict what would come next if this text were found on the internet, which will sometimes look like a helpful answer.
- Zero-shot with prompting or context-setting, which means including natural language instructions in the context to increase the probability that "predicting what comes next" is likely to yield a helpful response. For example, we may prompt with something like "Please provide a helpful and accurate answer to the following question: _____" or "This is the answer that a technician knowledgeable about air conditioners would give to the following question: _____".
- Using "Within-context learning" (or few-shot learning), which means providing a few examples of the desired downstream skill within the context window. We might provide three examples of questions about appliances followed by accurate and helpful answers before asking the question we would like to know about.
- After fine-tuning on a much larger number of examples of the relevant skill -- unlike any of the above strategies, fine-tuning actually updates the model's weights and tends to lead to better performance.

As next-token prediction accuracy improves, zero-shot and few-shot performance for a large number of downstream skills improves simultaneously in a relatively predictable way (with or without context-setting).[57] This makes it meaningful to ask about the effective horizon length *for the auxiliary ML problem of learning a particular downstream skill in the course of predicting text*, as opposed to the primary ML problem of learning to predict the next token.

I'll informally define the **effective horizon length of a downstream skill** analogously to the other definition: as the amount of data it takes (on average) to tell whether a perturbation to the

57 For the purposes of this hypothetical, I am assuming that the average episode length is much larger than 1000 timesteps, such that 1/(1-gamma) << episode length for both gamma_1 and gamma_2.

model improves performance or worsens performance *at the downstream skill of interest.*[58] Again, we can try to think about this in relation to the reference of next-token prediction.

The effective horizon length will get longer over the course of training because it is generally easier to tell which of two incompetent models is better for most skills. When the model is still relatively incompetent at basic grammar, any small perturbation which causes the model to make fewer grammatical mistakes is likely to improve performance at the downstream skill of interest, because basic grammar is a prerequisite for most of them. But later in training, both models will be equally excellent at the basics of writing, and it will likely take much more time to tell which is better at complex downstream skills.

Consider a large language model which is already competent at basic writing. Suppose we want to evaluate the model on one of two possible downstream skills: writing stories which have a creepy atmosphere, or writing stories which have a satisfying twist ending. If we perturb each of the model's weights by some ϵ, how many tokens do we need to see it produce before we can tell if that perturbation improved each of these downstream skills? *Prima facie,* it should be easier to tell if the perturbation made the model better at creating a creepy atmosphere (something which is generally demonstrated continuously throughout a story) than to tell if it made the model better at crafting a twist ending (something generally observed only once per story). Because of this, **the "create a creepy atmosphere" skill probably has a shorter effective horizon length than the "write a satisfying twist ending" skill**.

The perplexity loss is correlated with both skills -- getting better at prediction will smoothly improve the model's zero-shot and few-shot ability to write creepily and to write a good twist ending. However, I expect that getting to excellent zero-shot or few-shot performance at writing a twist ending will require training on more tokens than getting to excellent zero-shot or few-shot performance at creating a creepy atmosphere. I also expect that fine-tuning the model to achieve excellent performance at writing a twist ending will require more computation and data than fine-tuning it to achieve excellent performance at creating a creepy atmosphere.

Improving the accuracy of next-token prediction will simultaneously improve zero-shot and few-shot-within-context performance at many downstream skills like these two, and in general I would predict that the ones which seem to have a longer effective horizon (in the sense of requiring more words to tell which of two similar models is better) will require achieving closer-to-perfect prediction to master zero-shot (or with few-shot learning), and/or require more data and computation to master with fine-tuning.

However, actually quantitatively estimating the effective horizon length of any given skill is an extremely messy empirical question, because there is likely to be some amount of transfer between shorter-horizon skills and longer-horizon skills. Consider the binary advice questions problem described above. We can imagine two different training strategies for training a language model if we have a data distribution of questions and answers:

58 In some real-world training runs, the discount rate starts off steep and slowly moves closer to 1 over time.

1. **Supervised learning**, as described above: we train the model to predict only the answers to questions, although it must process the question first and uses more computation on longer questions.
2. **Pre-training and fine-tuning:** We pre-train the model to predict every token of the question and the answer, and then fine-tune using the supervised learning method, but using no more data or computation than the pre-training step took.

A supervised learning model with N parameters will be much better at answering binary advice questions than a fine-tuned language model with N parameters if both models are trained with a compute-optimal amount of data for their respective task. This is because the supervised learning model has seen a lot more data and performed a lot more computation -- if we assume that the scaling is similar to what Kaplan et al find, the supervised learning model has seen $288 \times N^{0.74}$ whole questions, while the fine-tuned language model has seen only one set of $288 \times N^{0.74}$ tokens in the pre-training step and another dataset of the same size in the fine-tuning step. If the average length of a question was 1000 tokens, then the fine-tuned language model has seen 500 times less data and performed 500 times fewer FLOP than the supervised learning model.

If we had a fixed compute budget C and were not data constrained, and we wanted to train a model to be good at answering long binary advice questions, we could either train a small model on a lot of data using supervised learning or a large model on a smaller amount of data using pre-training and fine-tuning. It's an empirical question which one would give better advice -- it's clear that being good at modeling the question will help to some extent with the task of answering questions well, but it's unclear how much, and whether that will make up for the fact that the fine-tuned language model would have seen a lot less data and would be trained to devote space and attention to many aspects of language modeling besides figuring out whether to answer "A" or "B" to a binary advice question.

Observational evidence of RL model training times

The empirical evidence I've examined so far comes from supervised learning and generative modeling problems. As of July 2020, I am not aware of any papers that systematically study sample complexity scaling behavior for RL problems.

This is an important gap, because if a transformative model is trained with techniques similar to current ML techniques, it seems likely that reinforcement learning will play some role.[59] In order to have a transformative impact, it is likely that the model will have to have substantially *superhuman* abilities in at least some domains, and it is difficult to use supervised learning alone to discover strategies fundamentally different from the ones humans use.

In this section,

59 See this blog post by Alex Irpan and this blog post by Himanshu Sahni (both written in early 2018) for an introduction to the practical problems of making RL work.

- I'll describe the process I followed to collect observational evidence about various RL training runs (more).
- I'll show that a power law fit similar to the target accuracy law from Kaplan et al 2020 describes this data reasonably well (more).

Selection criteria and process for RL training data points

Research Analyst Tom Davidson and I collected several examples of RL model training times. We gathered examples using a loose informal process. We started with the most widely publicized RL results between 2015 and February 2020, which we were already aware of through the media and our social networks: StarCraft, Go, the Rubik's Cube, DOTA, and Atari. We then moderately expanded our search by a) browsing recent DeepMind and OpenAI publications for additional RL results, and b) searching arxiv for recent reinforcement results, particularly in other competitive games such as poker. We selected papers that were relatively highly cited and appeared to achieve state-of-the-art RL results in a relatively important domain.

We selected 15 RL models during this search.[60] In this sheet, we recorded several pieces of relevant information about each one (e.g. the parameter count, the number of timesteps and/or episodes of training, the number of FLOP performed per timestep, the discount rate, etc). Once we had collected this information, I performed an additional filter, hiding rows in the spreadsheet if any of the following conditions applied:

- **Error or confusion:** The paper did not give me enough information to confidently estimate one of the key variables, I had good reason to suspect one of my estimates was incorrect in some way, or there was something conceptually confusing to me about the result.
- **Small training run:** The training run was small enough that the cost of computation and data were likely a negligible fraction of the overall project costs. For such training runs, it likely wouldn't have been worthwhile to put effort into training in a compute- and data-efficient way, and it is more likely that models were trained for an inefficiently long time, with poorly tuned hyperparameters, and so on. As I explained in Part 1, my operationalization of "data requirements" for a transformative model assumes a well-optimized training run, and smaller training runs provide less evidence about this quantity.
- **Poorly optimized training run:** I had strong reason to believe that a large-scale training run was nonetheless not well-optimized -- for example, a confident expert opinion or a clear example in which someone achieved substantially better performance per unit cost by training more efficiently.

There were five remaining models: AlphaStar, the "Rerun" training run from OpenAI's DOTA project, the large IMPALA model trained on Atari-57, and both the small and large AlphaGoZero models. The filter view named "Mainline" contains these models -- to see it, click on "Data" in the toolbar, then select "Filter views", then select "Mainline." There are two other Filter view

60 "The intersection point of L(D(C_min)) and L(C_min) occurs at C* = 10^4 PF-Days, N* = 10^12 parameters, D* = 10^12 tokens, L* = 1.7 nats/token…" pg 17.

options -- “None” (which includes all data points) and “High end” (which displays a set of points that result in a much steeper scaling law).

This search was not highly systematic or exhaustive, but I expect that we have captured a significant fraction of the well-optimized large-scale state-of-the-art RL results, simply because large-scale deep RL is still relatively new and largely restricted to a small number of well-resourced labs as of mid-2020. We may expand this dataset in the future; I expect we will see a number of new RL results in the coming years.

Sample complexity scaling behavior and interpretation

Above, I suggested that we should expect the number of timesteps needed to train an RL model to be described by $T(H,q,P)=H\times K(q)\times P^{\alpha}$, where P is the parameter count, H is the horizon length, and $q<1$ means that we want to train the model long enough to achieve the same performance as a smaller model of size qP.

If each “sample” is one horizon length, then the number of samples D needed to train a model of a certain size would be given by $D(q,P)=T(H,q,P)/H=K(q)\times P^{\alpha}$. The key pieces of information we need to interpolate a sample complexity scaling law of this form are parameter count P, total timesteps of training T, horizon length H, and desired accuracy q. In practice, researchers don’t explicitly choose q, so we won’t be able to explicitly disentangle that factor: we will try to fit an approximate scaling law $\widetilde{D}(P)\approx\widetilde{K}\,P^{\alpha}$, where the constant factor $\widetilde{K}$ incorporates researchers’ implicit choices about q.

I’ve collected information about the three variables (P, T, and H) required to fit this approximate law in columns D-G of the sheet. Horizon length was calculated to be $1/(1-\gamma)$ for the IMPALA and DOTA agents; AlphaStar had $\gamma=1$ so the horizon length was taken to be the average length of a game of StarCraft (roughly ~10 minutes according to the paper); horizon length was taken to be 1 move for AlphaGoZero.

As predicted, a simple power-law function provides a reasonable fit for the number of “horizons” as a function of parameter count, considering the small sample size; I have plotted this here (the chart would be difficult to see if pasted in the document). The green line is the best-fit power law trendline for the number of horizons in terms of parameter count:[61] $\widetilde{D}(P)\approx 2300\,P^{0.782}$. The reported correlation between P and the number of horizons is given by $r=\sqrt{\square}$, although we should not take this literally given the extremely small sample size (adding or removing one data point can dramatically change this value).

61 “The numerical values [of the intersection points] are highly uncertain, varying by an order of magnitude in either direction depending on the precise values of the exponents from the power-law fits.” pg. 17. See this appendix for my derivation of the crossover point, which arrives at a slightly different answer.

The precise scaling behavior is sensitive to the set of models included. If all models are included, the scaling exponent is about 1. If we include large models trained on an unusually large amount of experience such as OpenAI's Rubik's Cube model and include the DOTA original run rather than the DOTA clean run data point the scaling exponent can climb well above 1. Viewers of the spreadsheet can switch between the provided filter views or create their own filter views, which will be reflected in the two charts. The data points included in the "High end" filter view results in an exponent of 1.41, and the vertical lines in the exponent plot here give a sense of the spread.

How I think about data requirements overall

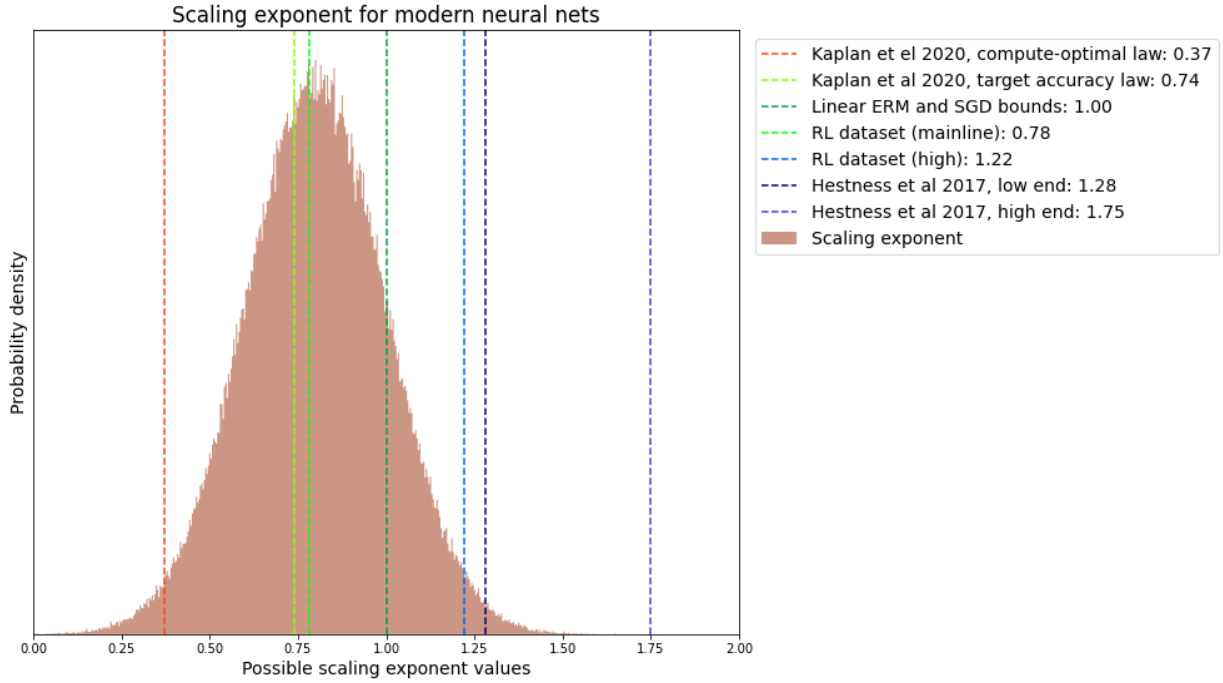
Overall, I think there is moderate theoretical and empirical evidence supporting the following claims about data requirements for learning problems similar to current learning problems:

- **Power-law fit as a function of P:** It is sensible to model the function relating parameter count P to number of training data points D as a power-law $\widetilde{D}(P)=\widetilde{K}P^{\alpha}$ -- this is a simple functional form which produces a strong fit in scaling laws derived from controlled experiments, and also fits observational data from RL training runs reasonably well. The two papers I examined also claim that this functional form is grounded in theory or intuition, although I have not investigated this. The constant factor will depend on how close the model gets to the best possible loss; this is an implicit choice researchers make that we generally don't know precisely.
- **Somewhere around linear scaling:** From ML theory and informal heuristics used by ML practitioners, we should have a prior that $\alpha \approx 1$ (i.e. linear scaling) and that the constant factor will scale with the variance in the loss signal. Experimental evidence from Kaplan et al 2020 and observational evidence from RL suggests that the exponent may be closer to ~0.7-0.8 for well-optimized training runs on problems similar to current learning problems. Hestness et al 2017 shows that superlinear scaling (~1.25-1.75) is possible, but the experiments in that paper were a lot less representative of the likely tradeoffs researchers would make in a realistic commercial setting.
- **Horizon length adjustment:** The length of one "data point" can be very different for different learning problems. To estimate the total amount of *subjective time* that a model needs to spend in training, we have to multiply the number of data points it requires by the effective horizon length of the problem it is training on. For supervised learning models, the horizon length is unambiguous; for RL models, $1/(1-\gamma)$ and/or the episode length can serve as good proxies. Horizon length is largely exogenous, and we need to think through the details of a learning problem (including options for reward shaping) to estimate it rather than calculating it from other quantities.
- **RL penalty:** Even after adjusting for horizon length, sample complexity seems to scale more poorly for RL models than for supervised learning models, likely because RL is more noisy.

Subjective distributions over scaling exponent and constant factor

In this section, I only generate a probability distribution over the number of "horizons" needed to train a transformative model if training is done in an optimized way; I will discuss considerations feeding into what horizon length may be necessary in Part 3. I will assume that the number of horizons follows the power law function $D(P) \approx KP^{\alpha}$.

My subjective distribution over the scaling exponent α has a median of 0.8 because I am inclined to mostly anchor to the theoretical predictions but place somewhat more weight on sublinear scaling than superlinear scaling because both Kaplan et al 2020 and the highest-quality RL data points seem to suggest somewhat sublinear scaling (and I don't consider Hestness et al 2017 to be as representative):



The constant factor is trickier to estimate, because it is more arbitrary (e.g. dependent on tokenization and the precise value of q), and it also seems more likely to vary wildly across tasks. Because the constant factor does not have inherent meaning, it's most intuitive to first specify a reference model with a certain number of parameters $\hat{P}$, generate a probability distribution over the number of samples $\hat{D}$ required to train the reference model, and then back out the constant factor K from this and the exponent distribution α: specifically $K = \hat{D}/\hat{P}^{\alpha}$.

I chose to use language models as the relevant reference class, because we currently have the clearest experimental evidence in that domain, and because it is plausible to me that a transformative model would be purely or mostly language-based. As discussed above, there are two scaling laws discovered in Kaplan et al 2020 which contradict one another: the compute-

optimal scaling law is shallower (exponent of 0.37) compared to the target accuracy scaling law (exponent of 0.74).

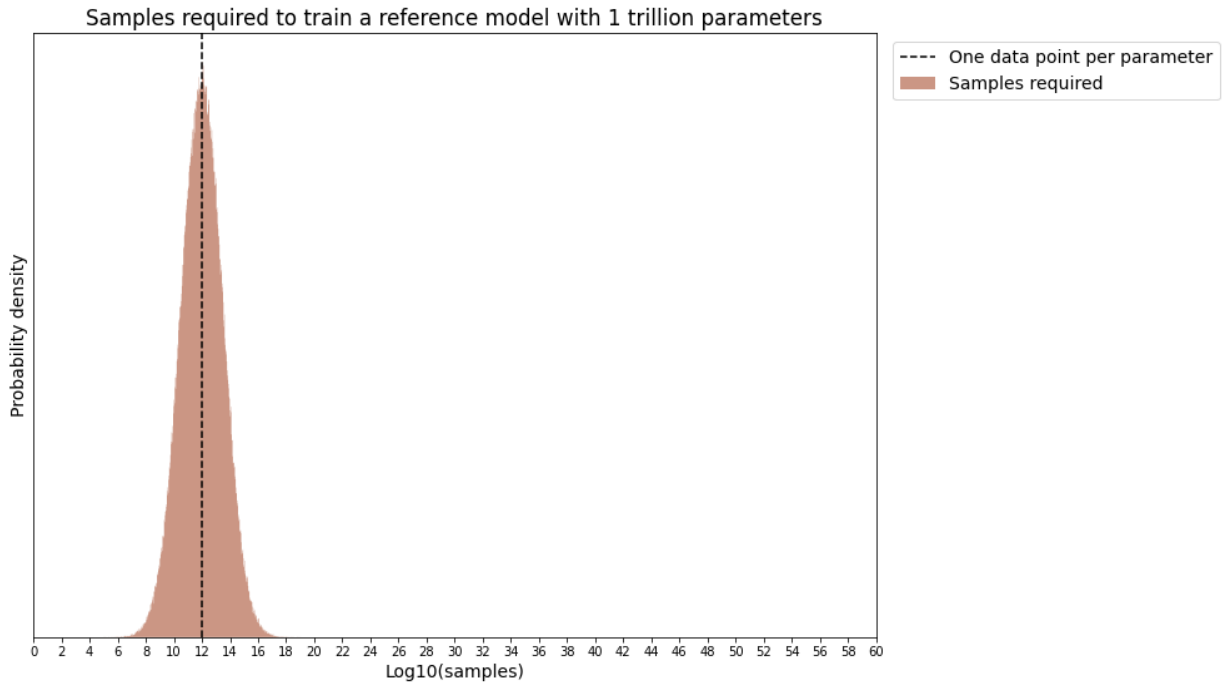
My expectation is that language models will follow the shallower compute-optimal scaling law for the next several doublings but eventually transition into the steeper target accuracy scaling law around the crossover point between the two scaling laws.

According to the paper, the crossover point occurs at around ~1e12 parameters and ~1e12 tokens of data,[62] although the precise value is highly sensitive to the precise exponents in the scaling law.[63] I chose $\widehat{P}=10^{12}$ as my reference parameter count, and assumed that the number of samples $\widehat{D}$ required to train a model with 1e12 parameters could be represented as a log-normal distribution with mean 1e12 and a standard deviation of 1.5 OOM:

62 We cannot precisely define this in terms of something like "the ratio between the optimal number of subjective seconds to train a language model and the optimal number of subjective seconds to train the other type of model" because different ML problems can have different scaling behaviors. For example, suppose that we wanted to know the effective horizon length of training a model to play StarCraft (expressed in units of "average number of frames of StarCraft per effective horizon"). Additionally, suppose that the number of tokens required to train a language model scales up proportionally to (parameters)^0.7, while the number of frames of StarCraft required to train a StarCraft-playing model scales up proportionally to (parameters)^1. There will be no fixed ratio between the optimal number of tokens to train a language model and the optimal number of game frames to train a StarCraft model that holds across all model sizes. For small models, the optimal number of game frames of StarCraft will be smaller than the optimal number of tokens, and vice versa for larger models; they can diverge arbitrarily far from one another.

I use a heuristic based on episode length and discount rate, described below, to estimate the effective horizon length of AlphaStar to be ~10 subjective min, the length of an average game of StarCraft, which is about ~2400 game frames. See here for a description of scaling data I collected from RL models. I am exploring whether there is an alternative way to cash out the notion of "providing as much useful information as one token" (the appropriate definition may need to be tied to a particular model size).

63 None of these metrics are intrinsically valuable, but they are fairly well-correlated with winning the game, and including these proximate reward signals will speed up learning because they mean that the DOTA agent can get away with optimizing more myopically for short-term reward while still achieving good performance. Essentially, proximate reward signals introduce some bias relative to the "natural" reward signal, but reduce variance and speed up training.



Note that I make the simplifying assumption that each "data point" is used about once over the course of training: that is, I will calculate total training FLOP as simply the product of the number of "data points", the "horizon length" in seconds of the average data point, and model FLOP/s.

While many models today were trained with substantial reuse of data (e.g. on dozens or hundreds of epochs),[64] I think that this is a reasonable assumption because training with one epoch tends to be the strategy which minimizes training computation, and I expect that researchers will aim to economize heavily on training FLOP as models get larger. To the extent you disagree with this assumption, you can reflect your beliefs by simply multiplying the value of the number of data points required to train the reference model by the number of epochs you believe will be used.

You may want to use another reference point (for example, an RL model such as AlphaStar or OpenAI Five). I suggest a few different options for a reference model in this section of the Colab notebook; different choices tend to shift the median value of the Neural Network and Genome Anchor hypotheses by about ~1 OOM.

64 Often the actor and the critic are actually the same model, sharing most of their weights; I have described them as separate for simplicity.

Forecasting TAI with biological anchors

Part 3: Hypotheses and 2020 training computation requirements

Author: Ajeya Cotra
Date: July 2020

This report emerged from discussions with our technical advisors Dario Amodei and Paul Christiano. However, it should not be treated as representative of either of their views; the project eventually broadened considerably, and my conclusions are my own.

This is a work in progress and does not represent Open Philanthropy's institutional view. We are making it public to make it easier to gather feedback, to help inform others' thinking in the effective altruism community, and to allow for follow-on work outside of Open Phil. However, we may edit it substantially in the future as we gather feedback from a broader audience and investigate open questions. Accordingly we have not done an official publication or blog post, and would prefer for now that people not share it widely in a low-bandwidth way (e.g., just posting key graphics on Facebook or Twitter).

The report has been divided into four Google docs to load faster. ***This is Part 3; the first part is here, the second part is here, and the fourth part is here.*** *Additional materials (collected in this folder):*

- *Quantitative model: the Python notebook Biological anchor hypotheses for 2020 training computation requirements; a template spreadsheet When required computation may be affordable; and my best guess, conservative, and aggressive forecasts.*
- *Supplemental materials: a document containing various appendices; a folder of figures for the report; the spreadsheet Extrapolations of data and compute to train models; and the Python notebook Compute price trends, which draws on data in this folder.*

In Part 1, I provided an overview of the framework and estimates, provided definitions for key abstractions used in the model, and generated an estimate for the number of FLOP / subj sec of a transformative model. In Part 2, I reviewed theoretical and empirical evidence about training data requirements for a transformative model, introduced the concept of horizon length, and estimated how training data requirements may scale with parameter count for a transformative ML problem.

In this part, I will discuss each of the six biological anchors hypotheses in more detail, and combine them to generate my 2020 training FLOP requirements distribution:

- I will start with the Neural Network hypotheses which I place the most weight on (more).
- I will then cover the Evolution Anchor, Genome Anchor, and Lifetime Anchor hypotheses in less detail (more).
- Finally, I will describe in more detail how I update against low-end FLOP levels and assign probabilities to each hypothesis to generate my 2020 training FLOP requirements distribution (more).

Then in Part 4, I will explain how I generate my estimate for when the amount of computation required to train a transformative model may become available, and answer several questions and objections about the framework.

Neural network hypotheses

This family of hypotheses states that we should assume on priors that a transformative model would perform roughly as many FLOP / subj sec as the human brain and have about as many parameters as we would expect if we simply scaled up the architectures of the largest current neural networks (e.g. transformer architectures) to run on that many FLOP / subj sec.

In Part 1 I generated a probability distribution centered around **~1e16 FLOP / subj sec** for the amount of computation that a transformative model is likely to run on; this is 1 OOM larger than my central estimate for brain FLOP/s. This estimate will be used for the Neural Network hypotheses and the Genome Anchor hypothesis below.

It adjusts from the anchor point of human brain FLOP/s by a relatively modest constant factor to account for qualitative considerations about how sophisticated our architectures seem to be as of 2020, and estimate parameter count by assuming that a transformative model would have a similar ratio of computation to parameters as the most expensive neural networks we have trained so far. With this it extrapolates the amount of FLOP required to train such a model using an empirically-derived scaling law that expresses training data as a function of parameter count.

This results in a median estimate somewhere between ~1e31 FLOP and ~1e38 FLOP depending on the effective horizon length of the learning problem. In this section, I will:

- Generate an estimate for the number of parameters this model may require if its architecture is similar to existing neural networks (more).
- Discuss what a reasonable range of effective horizon lengths could be for a transformative ML problem (more).
- Generate subjective probability distributions for training FLOP requirements conditional on this family of hypotheses, broken into “short”, “medium”, and “long” effective horizon lengths (more).

Parameter count of a transformative neural network

I have focused first on inference FLOP / subj sec because computation feels somewhat more “fundamental” and universal than parameter count (it is more comparable across different model architectures and across the whole spectrum from “mostly classical programming” to “mostly pure ML”).

The more computationally powerful a neural network is, the more parameters it is likely to use, so we can try to derive an estimate of parameter count given our estimate for inference computation.

For example, in a simple feedforward neural network architecture, a model which performs twice as many FLOP per forward pass will straightforwardly have twice as many parameters, because each parameter performs roughly one multiplication and one addition in a feedforward architecture. Convolutional neural networks (CNNs) perform substantially more FLOP per

parameter per forward pass than a generic feedforward neural network with the same number of parameters can, but also generally maintain a consistent ratio as they are scaled up. Other common architectures such as recurrent neural networks, transformers, etc also exhibit a broadly similar scaling behavior within their respective architecture class -- there is generally a fixed number of FLOP performed per parameter per forward pass.

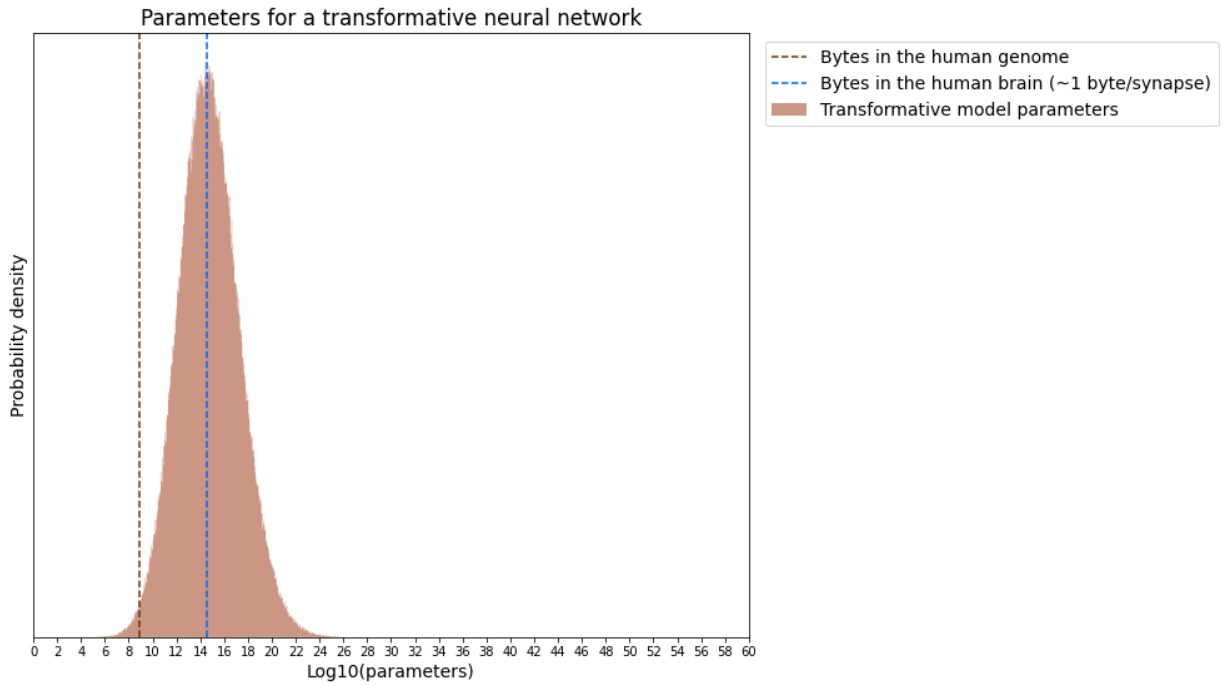
For typical neural network architectures, this ratio is somewhere between ~1 and ~100 FLOP per parameter per forward pass.[1] How many forward passes would a transformative model need to perform per second?[2] I would expect that ~0.1-10 forward passes per second is sufficient:

- According to Joe's report, each of the ~1e14-1e15 synapses in the human brain experiences a firing event roughly ~0.1-2 times per second. If we treat the parameters of a transformative model as broadly analogous to the synapses in the brain, that would suggest that each parameter should be used ~0.1-2 times per second.
- Average English speaking speed in conversation works out to about 2.5 words per second, while average reading speed works out to about 4 words per second and a good typing speed is slightly more than one word per second. If we imagine that a transformative model is primarily language-based, processing or generating one token of text with each forward pass, then matching human speaking, reading, and writing speeds would suggest performing a few forward passes per second.
- Average human reaction time is about 250 milliseconds, while especially skilled humans such as fighter pilots or competitive video game players reach reaction times closer to 100 milliseconds; if each forward pass of a transformative model constitutes an "action" and the model would need to match human reaction times in order to have a transformative impact, that would imply that it would need to perform 4-10 forward passes per second.

These assumptions imply that a transformative neural network should perform (1 to 100) * (0.1 to 10) = 0.1 to 1,000 FLOP per parameter per second, meaning that a model running on ~1e16 FLOP / subj sec would require ~1e13 to ~1e17 parameters. I expect that researchers would attempt to increase this ratio if possible, since reducing parameter count reduces data requirements, so I expect higher values to be somewhat more likely than lower values. For simplicity, I have assumed that the parameter count distribution for a transformative neural network is identical to the FLOP / subj sec distribution, but shifted to the left by 1.5 OOM (i.e. a factor of ~30), resulting in a distribution **centered around ~3e14 parameters**, a value slightly smaller than the number of synapses in the human brain:

1 Models which reach ratios higher than this are usually pure image processing models, which I think are likely less representative of a transformative model than language models or RL models.

2 Note that researchers are often able to control how many forward passes they run per second -- "forward passes per second" is not an inherent feature of a neural network, unlike "parameter count" or "FLOP per forward pass." For example, I expect that it would be necessary to perform thousands of forward passes per second to make the Long Horizon Neural Network hypothesis work, because otherwise training would take too much wall-clock time. In this section, though, I am assuming that the model is running at "human speed."



Note that this ratio is simply based on a rough empirical relationship between model FLOP per forward pass and model parameters. it is certainly possible in theory to design a very computationally intensive architecture with a very small number of parameters. For example, according to Wikipedia, Deep Blue (generally considered a “classical computer program” rather than a “machine learning model”) employed some number of learned parameters representing the relative value of particular positions in its evaluation function.[3] Conversely, it is also possible to design models which only use a small fraction of their parameters in each forward pass, such as the mixture-of-experts model GShard.

The chart above marks a point for the estimated number of bytes in the human genome; the Genome Anchor hypothesis (discussed below) assumes that the number of parameters required to characterize a transformative model is in this range, and therefore that the ratio of FLOP per parameter per forward pass is much more extreme.

Effective horizon length of a transformative ML problem

Recall that I am assuming we can estimate training data requirements with the following functional form:

Train FLOP = (F FLOP / subj sec) x (H subj sec / effective horizon) x ($K\,P^{\alpha}$ effective horizons)

3 “Deep Blue's evaluation function was initially written in a generalized form, with many to-be-determined parameters (e.g. how important is a safe king position compared to a space advantage in the center, etc.). The optimal values for these parameters were then determined by the system itself, by analyzing thousands of master games.”

In the previous two sections, I generated probability distributions for F and P for the Neural Network family of hypotheses; in Part 2 I generated probability distributions for the scaling parameters α and K.

The effective horizon length H of the Neural Network hypothesis is currently my biggest source of uncertainty. I currently think all values in the range from 1 subjective second to multiple subjective years are plausible. In this section, I will:

- Discuss why neural networks trained on ML problems with short effective horizon lengths (e.g. a few subjective minutes) could likely do useful work of various kinds (more).
- Explain why a transformative ML problem may nonetheless require a long effective horizon length (e.g. a few subjective *years*), because having a transformative impact may require replicating cognitive abilities that natural selection may have optimized with multi-year generation times (more).
- Sketch out one form that a long horizon transformative model training run could take (more).
- Briefly describe several strategies that researchers may be able to use to reduce the effective horizon length of an ML problem which may naively seem to be long horizon (more).
- Discuss how we might estimate the effective horizon length of an ML problem which consists of many sub-skills learned over varying time horizons (more).

Short horizon models can probably do a lot of economically valuable work

It seems likely to me that a non-trivial fraction[4] (>5%) of economically valuable labor currently[5] done by humans could be made a lot more efficient or automated entirely by neural networks trained on short horizon ML problems using techniques very similar to current ML techniques. For example, here's a potential hypothesis to automating the handling of customer service calls (which have already been automated to a large degree using simpler computer programs over the last decade or two):

1. Train a generative language model on a large, generic corpus of text scraped from the internet, such that it is able to write basically grammatical, superficially sensible English in response to a broad range of different contexts.
2. Fine-tune the model to predict customer service representatives' responses in transcripts of recorded customer service calls.
3. Feed the model real-time transcripts of customers' speech as new calls come in, and have it predict responses. Convert the model's text to speech in real time to carry on a conversation with the customer, potentially allowing a human to jump in when necessary.

4 My guess is that this is probably true for most possible ways of defining "fraction of human labor", e.g. weighting by hours, wages, value-added measures, etc.

5 Note that automating X% of the labor humans do as of 2020 will likely result in substantially less than X % reduction in labor force participation because of complementarity: technological innovations can create new work. My understanding is that historical waves of automation have tended to have little to no long-term effect on the fraction of humans who are employable -- e.g., technology has already automated away an outright majority of the human labor done in 1500 (mostly farming).

4. Continually improve the model using RL from human preferences: elicit quantitative ratings of the model's quality of service from call transcripts from human overseers or customers themselves, train a reward model to predict humans' quality ratings from call transcripts, and train the model to optimize predicted human score with reinforcement learning.

This is likely to be a relatively short horizon learning problem. Conservatively, the horizon length might be the length of an average customer service phone call, which are often quite short (e.g. a few minutes long). However, my best guess is that it would be substantially shorter than the average call length. Simply learning to generate idiomatic English that makes even vague sense in context is likely to be a significant fraction of the work of training the whole model, and the horizon length for "grammatical and vaguely sensical English" is probably closer to 1 token than multiple minutes. Additionally, it seems likely that in a lot of cases, the appropriate response to a statement by the customer can more-or-less be determined locally, without explicitly looking ahead to the end of the entire interaction (for example, a single interaction may involve dealing with multiple separate issues).

A similar process could be done for generating code. As of the time of writing in July 2020, language models are already being trained to predict what code humans will type next:

- TabNine, trained in 2019, is a code autocompletion service based on the open-source version of GPT-2 (~1.5B parameters) and fine-tuned on ~2 million GitHub files.[6]
- Microsoft's IntelliCode Compose (paper released in May 2020) is a code completion service based on a generative model GPT-C (~0.37B parameters) trained on ~1.2 billion lines of code.
- Very soon after, OpenAI (in a partnership with Microsoft) revealed an improved code-prediction model in a demo at the May 2020 Microsoft Build conference.

Such models seem to already be a little bit useful for speeding up routine coding, and potentially helping to catch some mistakes. They could likely be further improved with short horizon reinforcement learning, using a reward model trained on signals constructed from a variety of cues such as whether the code compiles/runs without errors, whether it passes unit tests (which may sometimes be partially written by a version of the model itself), human reviewer ratings, what edits are suggested by a human collaborator, etc.

Other categories of work that seem as if they could likely be done substantially or entirely by short horizon neural networks include:

- **Customer service and telemarketing.** Each interaction with a customer is brief, but ML is often required to handle the diversity of accents, filter out noise, understand how different words can refer to the same concept, deal with customization requests, etc.

6 "Deep TabNine is trained on around 2 million files from GitHub. During training, its goal is to predict each token given the tokens that come before it….Deep TabNine is based on GPT-2, which uses the Transformer network architecture. This architecture was first developed to solve problems in natural language processing. Although modeling code and modeling natural language might appear to be unrelated tasks, modeling code requires understanding English in some unexpected ways." July 2019 blog post Autocompletion with deep learning.

This is currently being automated for drive-thru order taking by the startup Apprente (acquired by McDonald's).

- **Personal assistant work:** This could include scheduling, suggesting and booking good venues for meetings such as restaurants, sending routine emails, handling routine shopping or booking medical and dental appointments based on an understanding of user needs, and so on.
- **Research assistant work:** This could involve things like copy-editing for grammar and style (e.g. Grammarly), hunting down citations on the web and including them in the right format, more flexible and high-level versions of "search and replace", assisting with writing routine code or finding errors in code, looking up relevant papers online, summarizing papers or conversations, etc.
- **Repetitive but dextrous manual labor** done by humans in factories, such as tying knots, painting, glueing, folding, etc. My understanding is that these types of work have been difficult to automate using classical robotics because there are tiny variations across different instances of the task that make it difficult to program a simple trajectory for robotic actuators. Such models could potentially be trained partially in simulation, as as the OpenAI Rubik's cube robot was, to cut down on physical capital costs.

The general theme is that work that is naturally broken up into short, fairly repetitive chunks for which we can clearly define either a training data distribution (as in medical diagnostics or customer service) or a training environment (as in factory work) can likely be cast as a short horizon ML problem. Note that it might not currently be economically or logistically feasible to train some of these models -- the only claim I am making is that it's likely we could *design a short horizon ML problem* such that a sufficiently large model trained on that problem using an amount of data/experience in line with what we would expect from existing training runs would successfully automate the relevant type of work.

We may need long horizons for meta-learning or other abilities that evolution selected for

Training a model with SGD to solve a task generally requires vastly more data and experience than a human would require to learn to do the same thing. For example, esports players generally train for a few years to reach professional level play at games like StarCraft and DOTA; on the other hand, AlphaStar was trained on 400,000 subjective years of StarCraft play, and the OpenAI Five DOTA model was trained on 7000 subjective years of DOTA. GPT-3 was trained on 300 billion tokens,[7] which amounts to about 3000 subjective years of reading given typical human reading speeds; despite having seen many times more information than a human about almost any given topic, it is much less useful than a human for virtually all language-based jobs (programming, policymaking, research, etc).

I think that for a single model to have a transformative impact on its own, **it would likely need to be able to learn new skills and concepts about as efficiently as a human, and much more efficiently than hand-written ML algorithms like SGD.** For a model trained in 2020 to

7 That is, the way that is most likely to be discovered by the optimization algorithm (e.g. SGD).

accelerate the prevailing rate of growth by 10x (causing the economy to double by ~2024), it seems like it would have to have capabilities broadly along the lines of one of the following:

- Automate a wide swathe of jobs such that large parts of the economy can ~immediately transition to a rate of growth closer to the faster serial thinking speeds of AI workers, or
- Speed up R&D progress for other potentially transformative technologies (e.g. atomically precise manufacturing, whole brain emulation, highly efficient space colonization, or the strong version of AGI itself) by much *more* than ten-fold, such that once the transformative model is trained, the relevant downstream technology can be developed and deployed in only a couple of additional years in expectation, and then *that* technology could raise the growth rate by ten-fold. For AI capable of speeding up R&D like this, I picture something like an "automated scientist/engineer" that can do the hardest parts of science and engineering work, including quickly learning about and incorporating novel ideas.

Both of these seem to require efficient learning in novel domains which would not have been represented fully in the training dataset. In the first case, the model would need to be a relatively close substitute for an arbitrary human and would therefore probably need to learn new skills on the job as efficiently as a human could. In the second case, the model would likely need to efficiently learn about how a complex research domain works with very little human assistance (as human researchers would not be able to keep up with the necessary pace).

Humans may learn more efficiently than SGD because we are able to use sophisticated heuristics and/or logical reasoning to determine *how* to update from a particular piece of information in a fine-grained way, whereas SGD simply executes a "one-size-fits-all" gradient update step for each data point. Given that SGD has been used for decades without improving dramatically in sample-efficiency, I think it is relatively unlikely that researchers will be able to hand-design a learning algorithm which is in the range of human-level sample efficiency.

Instead, I would guess that **a transformative ML problem would involve meta-learning** (that is, using a hand-written optimization algorithm such as SGD to find a model which itself uses its own internal process for learning new skills, a process which may be much more complex and sophisticated than the original hand-written algorithm).

My best guess is that human ability to learn new skills quickly was optimized by natural selection over many generations. Many smaller animals seem capable of learning new skills that were not directly found in their ancestral environment, e.g. bees, mice, octopi, squirrels, crows, dogs, chimps, etc.

The larger animals in particular seem to be able to learn complex new tasks over long periods of subjective time: for example, dogs are trained over a period of months to perform many relatively complex functions such as guiding the blind, herding sheep, assisting with a hunt, unearthing drugs or bombs, and so on. My understanding is that animals trained to perform in a circus also learn complex behaviors over a period of weeks or months. Larger animals seem to exhibit a degree of logical reasoning as well (e.g. the crow in the linked video above), which seems to help speed up their learning, although I'm less confident in this.

This makes me believe it's likely that our brain's architecture, our motivation and attention mechanisms, the course of brain development over infancy and childhood, synaptic plasticity mechanisms, and so on were optimized over hundreds of millions of generations for the ability to learn and perhaps reason effectively.

The average generation length was likely several months or years over the period of evolutionary history that seems like it could have been devoted to optimizing for animals which learn efficiently. I consider this a *prima facie* reason to believe that the effective horizon length for meta-learning -- and possibly for training other cognitive abilities which were also selected over evolutionary time -- may be in the range of multiple subjective months or years. It could be much lower in reality for various reasons (see below), but anchoring to generation times seems like a "naive" default.

Here I am not saying we should expect that training a transformative model would take as much computation as natural selection (that view is represented by the Evolution Anchor hypothesis which I place substantially less weight on than the Neural Network hypotheses). I am instead saying:

1. A transformative model would likely need to be able to learn new skills and concepts as efficiently as a human could.
2. Hand-written optimization algorithms such as SGD are currently much less efficient than human learning is, and don't seem to be on track to improve dramatically over a short period of time, so training a model that can learn new things as efficiently as a human is likely to require meta-learning.
3. It seems likely that evolution selected humans over many generations to have good heuristics for learning efficiently. So naively, we should expect that it could take an amount of subjective time comparable to the average generation length in our evolutionary history to be able to tell which of two similar models is more efficient at learning new skills (or better at some other cognitive trait that evolution selected for over generations).

My understanding is that meta-learning has had only limited success so far, and there have not yet been strong demonstrations of meta-learning behaviors which would take a human multiple subjective minutes to learn how to do, such as playing a new video game.[8] Under this hypothesis -- assuming that training data is not a bottleneck -- the implicit explanation for the limited success of meta-learning would be some combination of a) our models have not been large enough, and b) our horizons have not been long enough.

8 "All eutherian mammals have genomes of essentially the same size. It is instructive to consider the size of the mammalian genome in terms of the amount of computer-based memory that it would occupy. Each base pair can have one of only four values (G, C, A, or T) and is thus equivalent to two bits of binary code information (with potential values of 00, 01, 10, 11). Computer information is usually measured in terms of bytes that typically contain 8 bits. Thus, each byte can record the information present in 4 bp. A simple calculation indicates that a complete haploid genome could be encoded within 750 megabytes of computer storage space." Mouse Genome Informatics, Chapter 5 section 1.

This seems like a plausible explanation to me. Let's estimate the cost of training a model to learn how to play a new video game as quickly as a human can:

- **Effective horizon length:** Learning to play an unfamiliar video game well takes a typical human multiple hours of play; I will assume the effective horizon length for the meta-learning problem is one subjective hour.
- **Model FLOP / subj sec and parameter count:** Even if our ML architectures are just as good as nature's brain architectures, it seems plausible that models much smaller than the size of a mouse brain aren't capable of learning to learn complex new behaviors at all -- my understanding is that we have some solid evidence of mice learning complex behaviors,[9] and more ambiguous evidence about smaller animals. According to Wikipedia, a mouse has about ~1e12 synapses in its brain, implying that its brain runs on ~1e12 FLOP/s.[10] I will assume we need a model larger than the equivalent of a bee but smaller than the equivalent of a mouse (say at least ~3e9 parameters and 1e11 FLOP / subj sec) to perform well on the "learning to learn new video games" ML problem.

If the scaling behavior follows the estimate generated in Part 2, the amount of computation required to train a model that could quickly master a new video game should be (3600 subj sec) * (1e11 FLOP / subj sec) * (1700 * 1e11^0.8) = 2e25 FLOP. At ~1e17 FLOP per dollar,[11] that would cost $200 million, which makes it unsurprising this hasn't been successfully demonstrated yet, given that it is not particularly valuable.

Note that while meta-learning seems to me like the single most likely way that a transformative ML problem could turn out to have a long horizon, there may be other critical cognitive traits or abilities that were optimized by natural selection which may have an effective horizon length of several subjective months or longer.

What might a long horizon transformative ML problem look like?

A particularly salient vision for training a long horizon transformative model involves multiplayer self-play: the idea is to create a population of several instances of the agent who must compete and cooperate with one another to accumulate reward on a wide variety of rich and complex simulated environments over the course of episodes lasting multiple subjective years.[12] The agent is trained over many episodes with traditional reinforcement learning and/or evolutionary methods to maximize the total reward it gets in an average long horizon self-play episode.

9 Specifically, I assumed that there was no evidence against values greater than 1e27 FLOP, that there was perfect evidence against values less than 1e23 FLOP, and interpolated log-linearly between those two values.

10 I intend to construe the hypotheses broadly here, meaning that if researchers come up with any training run which ends up using an amount of computation that is squarely in-distribution for hypothesis X, that would count as "hypothesis X worked out." It does not necessarily have to look like the sketches I presented in this section.

11 *C. Elegans* is estimated to have ~7,500 synapses.

12 Note that because each effective horizon must be multiple subjective years long, it would likely take too much wall-clock time to run the model in real time and train it using a lot of interaction with humans; it would be necessary to speed up the model relative to real time and rely on human interaction for only a tiny proportion of its data.

This is intended to create a selection pressure toward “general intelligence” similar to the selection pressure over evolutionary history. The idea is that the need to compete with and cooperate with other humans to successfully navigate a complicated physical environment resulted in substantial fitness benefits accruing to humans who were slightly more intelligent than their peers (e.g. they could acquire more allies by being more persuasive and more useful, they could more easily deceive and defeat enemies, etc), resulting in a “general intelligence arms race” and an explosion of increasingly sophisticated strategies and counter-strategies.

Most zero-sum games are trained with self-play. OpenAI’s “Emergent Tool Use from Multi-Agent Autocurricula” is a very small-scale application of this idea to more than two players -- two teams with at least two agents per team play hide-and-seek in a simple simulated environment, and spontaneously generate strategies and counter-strategies that they weren’t explicitly taught over the course of many games.

Note that we don’t strictly need to have a competitive, interactive environment for this hypothesis to work; we could instead train a model on much the same types of problems as in the short horizon hypothesis (e.g. writing code, generating text, manipulating robots) while receiving feedback at much longer intervals. For example, the model might spend a few subjective years (perhaps corresponding to a few hours in wall-clock time) attempting to prove a certain theorem or write a novel, and a human overseer could provide feedback on the result of this attempt. This approach could also be combined with self-play.

Reasons to think effective horizon length may be shorter

Despite the *a priori* argument given above that solving a transformative task with ML is likely to involve long horizon training, there are reasons to think researchers could devise transformative ML problems with shorter effective horizon lengths:

- **Human feedback or demonstrations could provide short horizon proxies for long-term value:** Even if the policy’s actions have to be optimized to maximize expected reward over the next several minutes, we may be able to design the reward signals themselves to reflect much longer-term consequences.
    - One option is imitation learning. A short horizon model might be trained to generate behaviors that are hard to distinguish from behavior transcripts generated by human experts.
    - Another option is RL from human feedback. If human overseers have a good understanding of what immediate behaviors are likely to result in good long-term outcomes, they can directly reward those behaviors. As an analogy, small children are generally myopically optimizing for adults’ short-term approval, but the adults in their lives can provide approval for behaviors (e.g. homework or tooth-brushing) which will pay off over much longer timescales.

  Imitation and human feedback can be particularly powerful if learning a reward model (learning to accurately predict how much humans would approve of an action) or a discriminator (learning to accurately predict how much an action deviates from what a human would do in the same context) requires many fewer parameters than learning a policy (learning how to *actually perform* actions that humans would approve of and/or

that are hard to distinguish from humans' actions). In that case, a relatively small amount of human feedback or small number of behavior transcripts can be used to train a small reward model or discriminator, which can then be called a much larger number of times to train a large policy.

- **We might be able to train composable short horizon reasoning operations:** When humans make decisions optimizing for our long-run goals (e.g. pursuing a line of research or deciding whether to sell a company), we generally use some amount of explicit verbal planning and reasoning -- in other words, our decisions are governed by an internal structure, and we could explain how they were built up from shorter steps of reasoning if we needed to.[13] We may be able to train models to execute the kinds of short reasoning steps we use in our explicit verbal thinking, and compose those steps over and over again (e.g. in a long chain of deductive logic or a large tree search) to make a complex decision. We could then update the model to directly imitate the ultimate output of this large, slow process. In this way, what seems like a long horizon problem (making a decision based on long complex reasons) can be learned as a short horizon problem. This is essentially the same idea as Christiano's Iterated Distillation and Amplification, Anthony et al's Expert Iteration algorithm, and the algorithm used to train DeepMind's MuZero agent and earlier AlphaZero and AlphaGoZero agents.
- **Some short horizon behaviors might naturally generalize to longer horizons:** For some types of problems, it might be the case that the most "natural" way[14] to perform well on shorter instances of the problem is to discover a strategy that generalizes robustly to larger instances. For example, it seems likely to me that a sufficiently large model trained on a distribution of arithmetic problems of increasing size (one-digit, two-digit, … N-digit addition, subtraction, multiplication, and division) would eventually start to perform perfectly on arbitrarily large (>> N-digit) arithmetic problems, because it seems likely that implementing the underlying rules of arithmetic is the most natural way to perform well on arithmetic problems of a wide variety of sizes. We may be able to design a transformative ML problem which has this property -- or has certain important subproblems with this property which reduce the effective horizon length of the whole problem. On priors, we should expect this to be the case to some extent, since humans are capable of acting to optimize outcomes over periods of time substantially longer than one generation.
- **Longer-horizon behaviors may require a much smaller number of examples to learn:** For some ML problems, achieving strong performance might requiring learning *some* very long horizon behaviors, but those behaviors might be very simple -- representable by a small number of parameters and therefore requiring a very small number of long horizon data points to learn. As an artificial example, consider a problem in which the RL agent must learn to control many complicated actuators in its body in

13 e.g., "I decided to sell my company because they made a good offer. I think their offer is good because I don't expect my company is actually worth that much. I don't think it's worth that much because I would need to expand my user base a lot to justify that valuation, and I don't have any good ideas for expanding my user base…."

14 A more careful accounting of this hypothesis would likely need to consider both how human-designed architectures would compare to the genome and what "fraction" of the genome is likely devoted to relevant features of brain development.

order to move (e.g. a much larger version of QWOP). When the agent can get moving at all, it has the option of moving left or right. Let's say that moving left for 10,000 timesteps will result in a reward of +1, and all other behavior sequences result in a reward of 0. The subproblem of "Figure out which direction to move" has a horizon length of 10,000 but the optimal policy ("Keep going left") is very simple to represent; on the other hand, the subproblem of "Figure out how to move" is extremely complex, but has a much shorter horizon length. The effective horizon length of the entire problem might be something like $(N \times 1 + M \times 10^4)/(N+M)$, where N is much larger than M; this could end up being much closer to 1 than 10,000. It is unclear how much this toy example is representative of real-world RL training runs because it is very difficult to understand what "fraction" of an RL agent's "capacity" is being directed to certain sub-skills (or whether that's a reasonable ontology in the first place). It is plausible to me that it is somewhat representative -- for example, while a typical game of DOTA takes ~20 minutes to play, the OpenAI DOTA bot achieved strong play with a horizon length of only ~2 minutes, largely because its very strong "micro" play (i.e. skill at fast-paced combat tactics) made up for its mediocre "macro" play (i.e. whole-game strategy). It achieved *decent* "macro", but did not end up having to learn its intricacies really well in order to defeat humans at DOTA; it is plausible that a smaller fraction of its parameters were devoted to macro and it therefore needed a smaller number of long horizon examples than short horizon examples. The relatively small information content of the human genome may be a hint that there is relatively little to learn over horizons of multiple subjective years; see the discussion of multiple horizon lengths below and the Genome Anchor hypothesis further down for more detail.

- **Some key skills may naturally have a somewhat shorter effective horizon length:** Some economically valuable behaviors may have a "natural" horizon length closer to the range of "several hours or days" rather than "several years" because ground-level feedback may be easier to collect. For example, a significant fraction of the role of a venture capitalist might be well-described as a series of roughly independent "episodes" in which the VC learns about a new company before deciding whether to invest in it and if so how much; a model that was superhuman at this aspect of venture capital might add a lot of value and complement the skills of human VCs, even if it is not able to e.g. carry on long-running professional relationships. Similarly, law, consulting, IT, security, and medicine have some "episodic" elements of quickly orienting to a fresh problem and providing advice or making decisions about it on a limited timeframe. Particularly if we are looking for ways that models can *complement* humans rather than replace them one-for-one, many important aspects of many valuable roles could potentially be automated without necessarily requiring the model to have abilities that might have required long horizon meta-learning to acquire.
- **We may not need to run the agent the entire length of the task horizon:** For some tasks, we may need to see the consequences of an agent's actions play out in the world over a long period of time to estimate the reward, but the agent itself may only need to take actions for a relatively small portion at the beginning of each episode, after which the consequences take their own course until the agent's reward can be determined.

Because I expect running the environment to be substantially cheaper computationally than running the agent, this could substantially reduce the effective horizon length.

These possibilities are not mutually exclusive. We may first select types of work that seem to have "natural horizon lengths" of several hours or days rather than months or years. It might then turn out to be the case that most of the behaviors learned over a horizon of 5 minutes generalize naturally to the timescale of days, and many of the remaining longer-horizon behaviors are simple enough that they can be learned with a very small number of day-long data points, not much impacting the average horizon length. We might then tackle the remaining difficult long horizon behaviors through a combination of human feedback, demonstrations, decomposition / Expert Iteration, and other techniques. If there are some important long horizon behaviors that we are still not able to train with an affordable amount of data and computation, we might figure out ways for humans to make up for the weaknesses of the model while still extracting a huge amount of economic value from the model that might count as "transformative" impact.

I expect that exploiting these possibilities fully will require building up a lot of infrastructure and know-how, such that it is more plausible that tactics like these could be used to bring effective horizon lengths down to a few subjective minutes in 2030 or 2040 than it is in 2020. A more accurate version of this model would incorporate this by increasing the probability of shorter effective horizon lengths over time.

What might effective horizon length be if a mixture of horizons is required?

Reinforcement learning problems and generative modeling problems tend to involve simultaneously learning a number of sub-skills that play out over varying timescales. For example,

- A real-time strategy game like StarCraft will involve exercising strategic skills (which play out over the course of a whole game), tactical skills (which come up several times per game), and reflex-based skills (exercised continuously throughout a game).
- Language modeling involves modeling very local correlations between words (e.g. various grammatical rules or common phrases and idioms) as well as regularities that hold across much longer chunks of text (e.g. the fact that character names and descriptions must remain consistent over a novel).

In general, I expect that a model would have stronger performance on skills that it has had more opportunity to exercise in the training distribution -- e.g., I expect that language models will be more likely to make errors in logical consistency across a long narrative than grammatical errors, and I expect models trained to play real-time strategy games to be much more impressive at aspects of the game that require fast and precise reflexes than aspects which require good strategy.

If excellent performance at short horizon skills combined with lower performance at longer horizon skills is sufficient for overall success at the relevant task -- or if short horizon skills transfer very well to longer horizon skills -- then the total cost of training will be dominated by the

cost of learning the skills which play out over short timescales, and the effective horizon length of the overall problem will be short; if not, the effective horizon length may be several orders of magnitude higher, dominated by the longest-horizon skills in the distribution.

It is likely that a transformative ML problem would also involve learning different behaviors over different timescales. How can we model the way these different timescales would average out into an overall effective horizon length? I have heard of three high-level perspectives, although many more are possible:

- **Short horizon training (especially generative modeling) will dominate:** Under this perspective, almost all of the difficulty and expense of training most practically valuable behavior with ML will come from generative modeling (i.e. training the model to predict what it will see next), which has a very dense reward signal. According to this perspective, generative training will cause the model to build up useful concepts and representations that can then be transferred to a wide variety of downstream skills, such that the model can achieve human-level performance on those skills with very little additional fine-tuning.
- **Long horizon training (especially meta-learning) will dominate:** Under this perspective, a non-trivial fraction p (e.g. 1% or 10%) of the training data must be focused on key long horizon behaviors -- particularly the behavior of efficiently learning complex new skills over subjective months of years -- for the model to learn those behaviors adequately enough to have a transformative impact. The overall effective horizon length is therefore at least equal to p times the horizon length of the long horizon behaviors, a term which will likely dominate over the short horizon portion of training. Note that a perspective which expects training data to be distributed log-uniformly across several orders of magnitude of effective horizon length would be equivalent to this perspective, because roughly $1/n^{th}$ of the data would have a horizon length of 10^n. The Genome Anchor hypothesis, explored below, could be framed as a variant of this hypothesis which uses the information content of the human genome as an estimate of how large the “hard core” of long horizon behavior may be.
- **The number of data points at each timescale will decay exponentially:** Under this perspective, some fraction q of the training data can be focused on the shortest-horizon behaviors, and q of the remaining data can be focused on the second-shortest-horizon behaviors, and q of the remaining data can be focused on the third-shortest-horizon behaviors, and so on. This is one very simple way to model the idea that shorter-timescale skills may transfer to longer-timescale skills, but only partially or imperfectly; this could be how curriculum learning might turn out to work. The overall effective horizon length on this perspective is highly sensitive to the value of q. If we assume a range from 1 subjective second to 1 billion subjective seconds (~32 subjective years), setting $q=0.2$ will result in an effective horizon length of ~4.4 subjective years, whereas setting $q=0.8$ will result in an effective horizon length of 15 subjective minutes. This spreadsheet implements the exponential decay model of horizon lengths, allowing the user to choose different values of q.

I was personally most intuitively sympathetic to the second perspective when I began this investigation, but after reflecting and discussing more I am ultimately very uncertain which of these high-level stories will turn out to be most correct. Differing intuitions about this question seems to be an important driver of disagreement about TAI timelines for the set of people I have talked with.

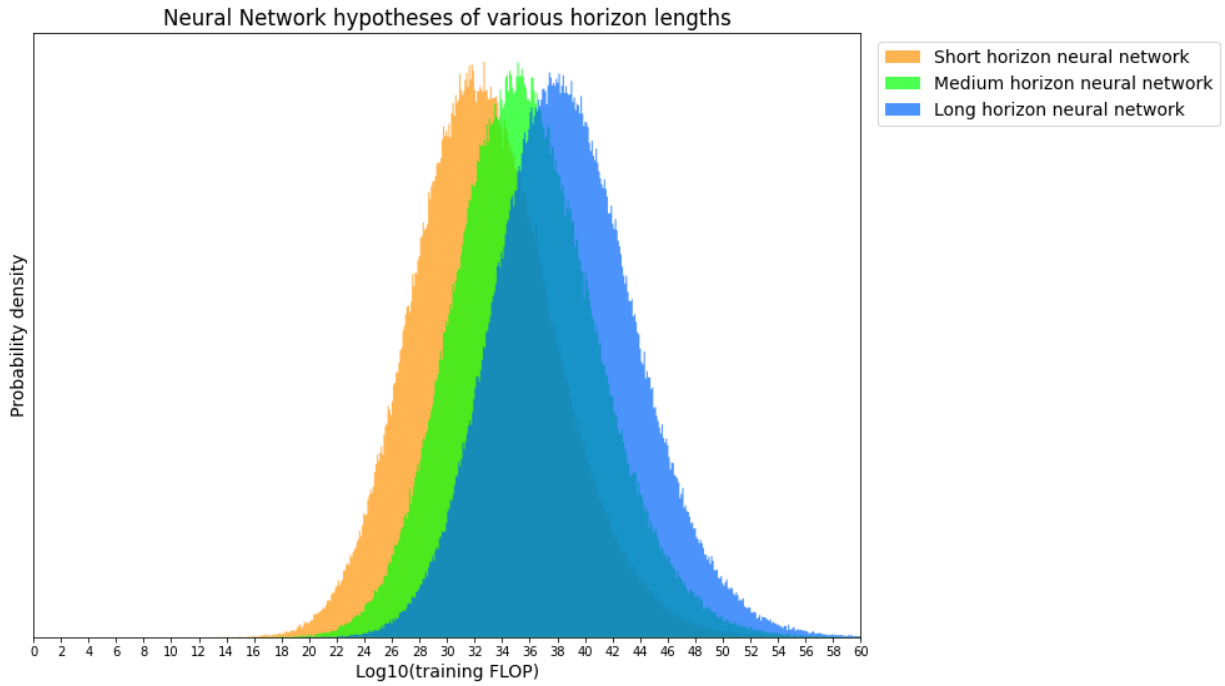
Training FLOP distributions for neural network hypotheses

I divide the spectrum of possible horizon lengths somewhat arbitrarily into three categories:

1. **Short Horizon:** a log-uniform distribution between 1 subjective second and 1000 subjective seconds (~17 subjective minutes), with a median of **~32 subjective seconds**.
2. **Medium Horizon:** a log-uniform distribution between 17 subjective minutes and 1,000,000 subjective seconds (~1.7 subjective weeks), with a median of **~8.8 subjective hours**.
3. **Long Horizon:** a log-uniform distribution between ~8.8 subjective hours and 1,000,000,000 subjective seconds (~32 subjective years), with a median of **~1 subjective year.**

The categories were chosen to span an equal range in log-space (3 OOMs). The low end of the short horizon category was chosen because it is close to the effective horizon lengths of the shortest-horizon problems we solve today. The high end of the long horizon category was chosen because it is more than the amount of time that it would have taken for someone to grow up and successfully reproduce in the human ancestral environment (i.e., it is longer than the typical “horizon length” used in human evolution).

Here are my subjective distributions over the amount of computation it would take to train a transformative model according to each of these three hypotheses:



The median 2020 training FLOP requirements predicted by the Short Horizon hypothesis is **~1e31 FLOP**; the Medium Horizon hypothesis predicts a median of **~3e34 FLOP**; and the Long Horizon hypothesis predicts a median of **~1e38 FLOP.**

Other biological anchor hypotheses

I have thought most deeply about the three Neural Network hypotheses, and they collectively drive my views the most. In this section I describe my briefer investigations into three alternative biological anchor hypotheses:

- The Genome Anchor hypothesis, which anchors to the amount of information in the human genome and arrives at a median estimate of ~3e33 FLOP for 2020 training computation requirements (more).
- The Lifetime Anchor hypothesis, which anchors to the amount of computation done over the course of a human lifetime, and arrives at a median estimate of ~1e27 FLOP before updating against low-end FLOP levels, which pushes the median to ~1e28 FLOP (more).
- The Evolution Anchor hypothesis, which anchors to the amount of computation done over the course of evolution from early neurons, and arrives at a median estimate of ~1e41 FLOP (more).

Genome Anchor hypothesis

This hypothesis states that we should assume on priors that a transformative model would run on roughly as many FLOP / subj sec as the human brain *and have about as many parameters as there are bytes in the human genome (~7.5e8 bytes)*, and that we can extrapolate the amount of FLOP required to train such a model using an empirically-derived scaling law that expresses training data as a function of parameter count. It adjusts from the anchor points of

human brain FLOP/s and human genome parameters by a relatively modest constant factor to account for qualitative considerations about how sophisticated our architectures seem to be as of 2020.

Like the Neural Network hypotheses, this hypothesis anchors on human brain FLOP/s to estimate model FLOP / subj sec. I will be using the same probability distribution over FLOP / subj sec derived in Part 1, centered around ~1e16 FLOP / subj sec, which is ~1 OOM higher than my estimate for human brain FLOP/s. Like the Long Horizon Neural Network hypothesis, this hypothesis assumes that **we need to train a transformative model to have some critical cognitive ability (such as sample-efficient learning) that evolution optimized for,** and that the ML problem of finding a model with this ability would have a long effective horizon length, comparable to the generation length over evolutionary history.

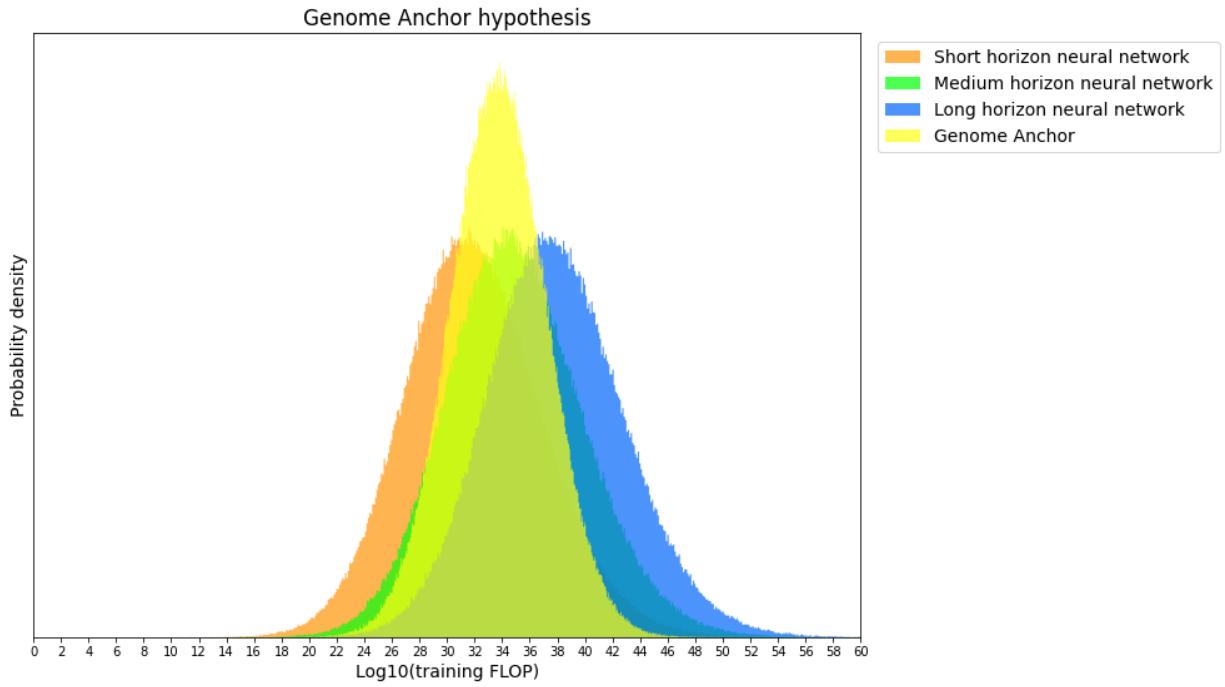
The key difference is that this hypothesis assumes that researchers could execute a long horizon training run like the one described above using many orders of magnitude fewer parameters than the Long Horizon Neural Network hypothesis assumes, anchoring parameter count to the amount of information we believe can be stored in the human *genome*. The motivating intuition is that evolution performed a search over a space of small, compact genomes which coded for large brains rather than directly searching over the much larger space of all possible large brains, and human researchers may be able to compete with evolution on this axis.

According to Wikipedia, the human genome contains 750 megabytes (7.5e8 bytes) of information; my impression is that this figure is not very uncertain, and the conception of "information in the genome" is much less conceptually fraught than the concept of "brain FLOP/s", but I have not dug into this question. For simplicity, I centered the distribution of parameter count for this hypothesis around 7.5e8 rather than attempting to think about how I would shift it,[15] and assumed ~3 OOM uncertainty on either side to match the amount of uncertainty I assumed when estimating how much larger or smaller a transformative model would be compared to the brain.

I assumed that the effective horizon length was log-uniform between ~3e7 subjective seconds (~1 subjective year) and 1e9 subjective seconds (~32 subjective years), for an average of ~5 subjective years. I also assumed that the scaling behavior is the same as the one derived in Part 2 and used in the Neural Network hypotheses.

The yellow distribution in the image below is the subjective distribution over the amount of FLOP it would take to train a transformative model that runs on ~1e16 FLOP / subj sec under these assumptions:

15 The behaviors I have seen mice learn seem substantially shorter horizon and less complex than learning new video games, but the mice are managing to do this despite having been optimized for survival and reproduction in the mouse ancestral environment rather than directly for "learning new tricks", so I would expect a mouse-sized meta-learned model to be substantially better than mice at learning, while being worse at various other mouse-survival-and-reproduction-relevant behaviors.



This distribution is narrower than the corresponding distributions for the Neural Network hypotheses primarily because the parameter count distribution is narrower since I took a simple point estimate for the number of bytes in the human genome rather than the wide distribution I used for the human brain FLOP/s anchor. Secondarily, uncertainty over the sample complexity scaling exponent doesn't translate into quite as much uncertainty over the number of total data points required because the parameter counts in question are similar to the number of parameters in existing models.

How plausible is this hypothesis? There are at least two distinct ways to interpret this hypothesis:

- One possibility is that the hypothesis is claiming that researchers in 2020 would be able to design an architecture for a transformative model which only contains ~7.5e8 parameters given ~2-5 years of trying (per the definition of technical difficulty in Part 1).
- Another possibility is that the Genome Anchor hypothesis is not directly making a claim that researchers will quickly create a genome-like architecture, but simply using the genome as a way to bound what fraction of data points would need to be long horizon when training an ordinary neural network. This would make it a more specific version of the "Long horizon training will dominate" view described above, which takes a view on the exact number of long horizon data points.

I am quite skeptical of the first interpretation, that researchers could quickly design a genome-like architecture given the current state of ML algorithms:

- Such an architecture is very different from typical architectures we use today, which perform a much smaller number of FLOP per parameter per timestep (e.g., ~1-100

rather than millions). A normal neural network in which each parameter performs a small number of operations per timestep is a very agnostic architecture that doesn't require researchers to understand how the brain works much; reducing parameter count by a factor of over 1 million from that agnostic starting point is a substantial step toward ordinary programming on the spectrum from "pure ML" to "pure programming", and seems like it would require much more specific knowledge on the part of researchers.

- If it were easy to achieve impressive results in language modeling, games, image classification, and so on using such a small ratio of parameters to computation per timestep -- while still training on a small number of data points per parameter -- researchers likely would have been doing this already because it would have saved a lot on training computation. The fact that they haven't suggests that this version of the hypothesis would require that researchers either come to understand how the human genome codes for the development of the human brain and design an architecture that imitates this mapping, or else develop a new architecture at least as effective as that genotype-to-phenotype mapping despite not understanding the biological mechanisms in detail and not having found a similarly efficient architecture so far.
- Even if researchers could discover a mapping from ~7.5e8 parameters to a space of models running on ~1e16 FLOP / subj sec which contains a transformative program, we have very little evidence about whether current local optimization techniques like SGD could effectively search through that space. The optimization hypothesis may need to be more winding, or we may need to train much closer to convergence to achieve the desired level of performance, or we may not be able to use gradients, all of which would likely make sample complexity scale more poorly with parameter count than it does for today's architectures.

I find the second possibility more compelling, but am still somewhat skeptical on net. It seems like if either version of this hypothesis is correct, we should have seen more impressive results in meta-learning so far. Above, I argued that if a model needs to be at least 10% the size of a mouse brain to be able to learn new video games as well as a human, then training a typical neural network of that size on that problem would cost ~$200M in 2020. However, if the cost of meta-learning would be dominated by the cost to train as many parameters as there are in the mouse *genome*, this cost would be much lower.

Mice and humans have essentially the same number of base pairs in their genome,[16] implying 7.5e8 parameters would be required to represent a mouse. Assuming as above that the model could run on ~1e11 FLOP / subj sec and we need about one 1-subjective-hour-long data point per parameter, this would cost (1e11 FLOP / subj) * (3600 seconds / data point) * (7.5e8 data points) = $2.25 million; the fact that we do not yet have models that are capable of learning complex new skills such as games over the course of several subjective minutes despite various attempts feels like substantial evidence against this hypothesis.

16 This is optimistic for a well-optimized training run in 2020, and definitely too optimistic for earlier time periods.

Lifetime Anchor hypothesis

This hypothesis states that we should assume on priors that training computation requirements will resemble the amount of computation done by a child's brain over the course of growing to be an adult, because we should expect our architectures and optimization algorithms to be about as efficient as human learning. It anchors to this estimate and adjusts from this anchor by a relatively modest constant factor to account for qualitative considerations about how sophisticated our architectures and algorithms seem to be as of 2020.

Suppose it takes on average about 1 billion seconds (~32 years) for an intelligent human to go from an infant[17] to their peak level of productivity. If a human brain performs ~1e15 FLOP/s, that would be (1e15 FLOP/s) * (1e9 seconds) = 1e24 FLOP, only about 1 OOM larger than the amount of computation used to train AlphaStar. I am quite skeptical that this is the most appropriate anchor (see below), but here I will do my best to condition on the assumption that this is the best biological anchor to work with and training FLOP will be somewhere in this region, and generate the most plausible version of the hypothesis given that assumption.

If ~1e24 FLOP is the most relevant biological *anchor*, where should we expect our training computation requirements would fall *relative* to that anchor? For the Neural Network and Genome Anchor hypotheses, I assumed that a transformative model would need to run on about ~1 OOM more FLOP / subj sec than the brain; I expect that if we are anchoring training FLOP on the human lifetime, we would need to shift the distribution by *more* than 1 OOM to the right. This is because:

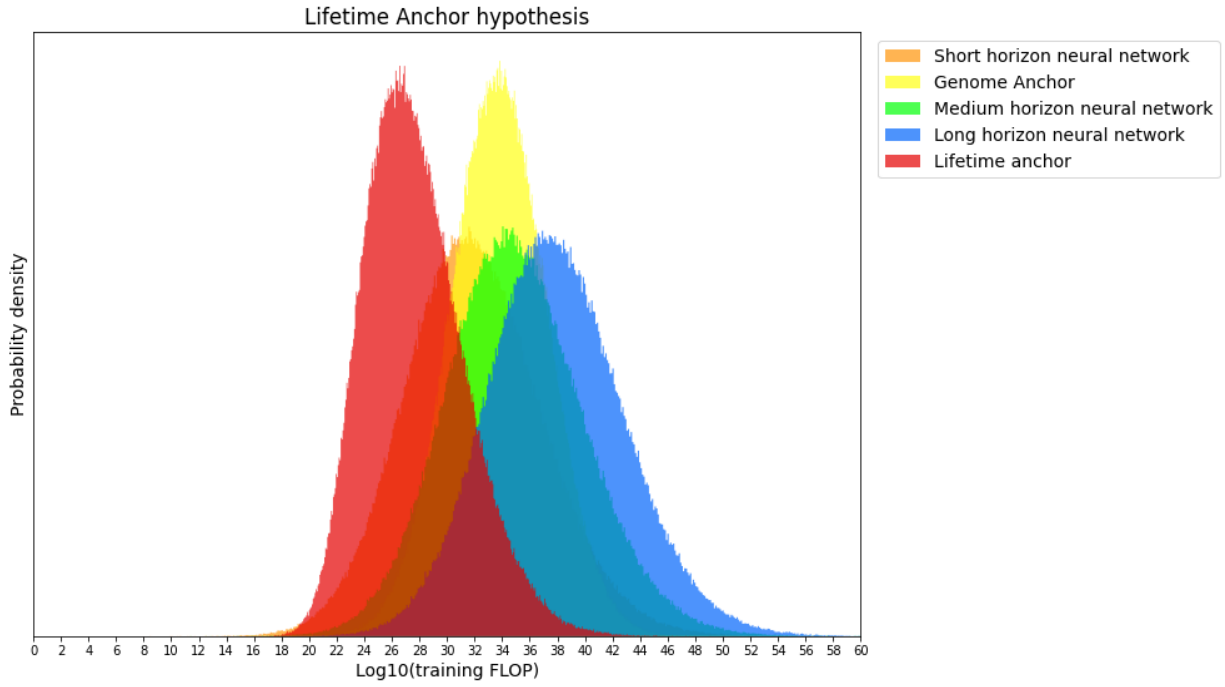
- Many models we are training currently already require orders of magnitude more data than a human sees in one lifetime. For example, GPT-3 was trained on 300 billion tokens,[18] which amounts to about 100 billion subjective seconds or ~100 subjective lifetimes of experience given human reading speeds. AlphaStar was trained on 10,000 subjective lifetimes of experience, and the OpenAI Five DOTA model was trained on 200 subjective lifetimes. These models are sub-transformative, and scaling to TAI would likely not involve a major *reduction* in training data requirements, even if trends suggesting steep *increases* in data requirements are too pessimistic.
- In the neural network hypotheses, model FLOP / subj sec is correlated with model parameters, and model parameters determine how much data the model needs to train on. This means that the model FLOP / subj sec distribution essentially impacts the training FLOP distribution *twice*, because it also partially determines how many seconds of training the model requires. A shift of 1 OOM in model FLOP / subj sec translates into a shift of about ~2 OOM in training FLOP.

17 While GPT-3 displays impressive within-context few shot learning at a wide variety of natural language processing tasks, my impression is that so far it has not performed tasks which would require the typical *human* to learn something new, or would otherwise require humans to think over a period of multiple subjective minutes. For the most part, these seem to be tasks that a human would know how to do very well from the description alone, and which would take a human <1 minute to execute, consistent with the effective horizon length of GPT-3 being relatively short on downstream tasks.

18 That is, assuming that mice neurons fire about the same number of times per second on average as human neurons, and that the conversion from "synaptic firing events" to "FLOP" is the same for both species.

- Brain FLOP/s seems to me to be somewhat more analogous to “ongoing energy consumption of a biological artifact” while lifetime FLOP seems to be more analogous to “energy required to *manufacture* a biological artifact”; Paul’s brief investigation comparing human technologies to natural counterparts, which I discussed in Part 1, found that the manufacturing cost of human-created artifacts tend to be more like ~3-5 OOM worse than their natural counterparts, whereas energy consumption tends to be more like ~1-3 OOM worse.
- Human babies may be born with various “baked-in priors” from evolution that make learning faster -- for example, they may have an intuitive understanding of what faces look like, or an intuitive grasp of the structure of human languages, intuitive priors about the visual world (for example that objects will be separated from other objects with “edges”) or an intuitive understanding of physics. Even if it is relatively trivial to somehow imbue ML models with these same priors (such that the human lifetime is still the most appropriate biological anchor to use), it may require somewhat more computation -- either because researchers may need to do some trial-and-error to determine how to “hard-code” these priors into a learning setup or because the model may need to do a constant amount of extra learning at the beginning to “catch up.”

Somewhat arbitrarily, I settled on a median of ~3 OOM larger than the anchor. Here, I have generated a subjective probability distribution over the amount of computation that would be required to train a transformative model conditional on the Lifetime Anchor hypothesis:



To generate the compute requirements distribution, I multiplied the distribution over brain FLOP/s by 1e9 seconds, and multiplied the result by a skew-log-normal distribution with a median of ~3 OOM, a standard deviation of ~5 OOM, and a right skew (recall that I chose ~5 OOM as the spread based on my beliefs about algorithmic progress; see this section from Part

1). This expresses the view that there is a ~50% chance researchers can train a transformative model using at most ~1000 times the amount of computation done by the human brain from birth to age 32 -- and if that doesn't work, there is a long tail of larger levels of computation that might turn out to be necessary.

Note that this hypothesis does not make any direct assumption about the model size, instead directly anchoring on a certain value for total training FLOP. We can see that if a transformative model would require ~1e15 parameters and run on ~1e16 FLOP / subj sec, the Lifetime Anchor hypothesis predicts that the amount of computation required to train it is orders of magnitude smaller than the amount of data we would expect to need based on the extrapolation from current models, even if we allow for an extremely short (<1 second) horizon length.

I think the most plausible way for this hypothesis to be true would be if a) it turns out we need a smaller model than I previously assumed, e.g. ~1e11 or ~1e12 FLOP / subj sec with a similar number of parameters, *and* b) that model could be trained on a *very* short horizon ML problem, e.g. 1 to 10 seconds per data point. Condition a) seems quite unlikely to me because it implies our architectures are much more efficient than brain architectures discovered by natural selection; I don't think we have strong reason to expect this on priors and it doesn't seem consistent with evidence from other technological domains. Condition b) seems somewhat unlikely to me because it seems likely by default that transformative ML problems have naturally long horizon lengths because we may need to select for abilities that evolution optimized for, and possible measures to get around that may or may not work.

The second way this hypothesis could turn out to be true would be if we need to use a large model (e.g. ~1e15 parameters) but manage to vastly outperform ML sample complexity trends on at least one transformative ML problem within ~2-5 years of effort. This also seems unlikely to me -- I don't think there are compelling reasons to believe that transformative ML problems would naturally have more favorable sample complexity scaling than current problems, and I am not aware of any plausible paths to dramatically improving sample complexity quickly.

Another major reason for skepticism is that (even with a median ~3 OOM larger than the human lifetime) this hypothesis implies a substantial probability that we could have trained a transformative model using less computation than the amount used in the most compute intensive training run of 2019 (AlphaStar at ~1e23 FLOP), and a large probability that we could have done so by spending only a few OOMs more money (e.g. $30M to $1B). I consider this to be a major point of evidence against it, because there are many well-resourced companies who could have afforded this kind of investment already if it would produce a transformative model, and they have not done so. See below for the update I execute against it.

More broadly, the Lifetime Anchor hypothesis contradicts the efficient-market hypothesis, implying that AI companies and the inputs to AI research are radically mispriced. It is also in tension with the general pattern observed across domains that technological progress tends to be broadly continuous -- it implies that a transformative model could be trained in the very near

term, increasing the annual economic value-added of ML models from hundreds of billions of dollars to hundreds of trillions of dollars in a very short period, and taking the world from a ~3% growth rate to a ~30% growth rate all at once without first passing through intermediate levels of growth. While I don't believe that markets are always fully efficient or that technological progress is always continuous, I would still expect AI to be adding a lot more economic value than it is today, and to be priced much higher on the market, if this hypothesis were actually feasible.

Evolution Anchor hypothesis

This hypothesis states that we should assume on priors that training computation requirements will resemble the amount of computation performed in all animal brains over the course of evolution from the earliest animals with neurons to modern humans, because we should expect our architectures and optimization algorithms to be about as efficient as natural selection. It anchors to this estimate and adjusts by a relatively modest factor to account for qualitative considerations about how sophisticated our architectures and algorithms seem to be as of 2020.

Like the Long Horizon Neural Network hypothesis and the Genome Anchor hypothesis, this hypothesis assumes that we need to train a transformative model to have some critical cognitive ability (such as sample-efficient learning) that evolution optimized for. Unlike those hypotheses, it assumes that human-designed architectures and optimization algorithms have done very little to reduce the amount of computation that meta-learning would take compared to a baseline of simulating natural selection from a very primitive starting point such as a randomly-connected nerve net with dozens of cells.

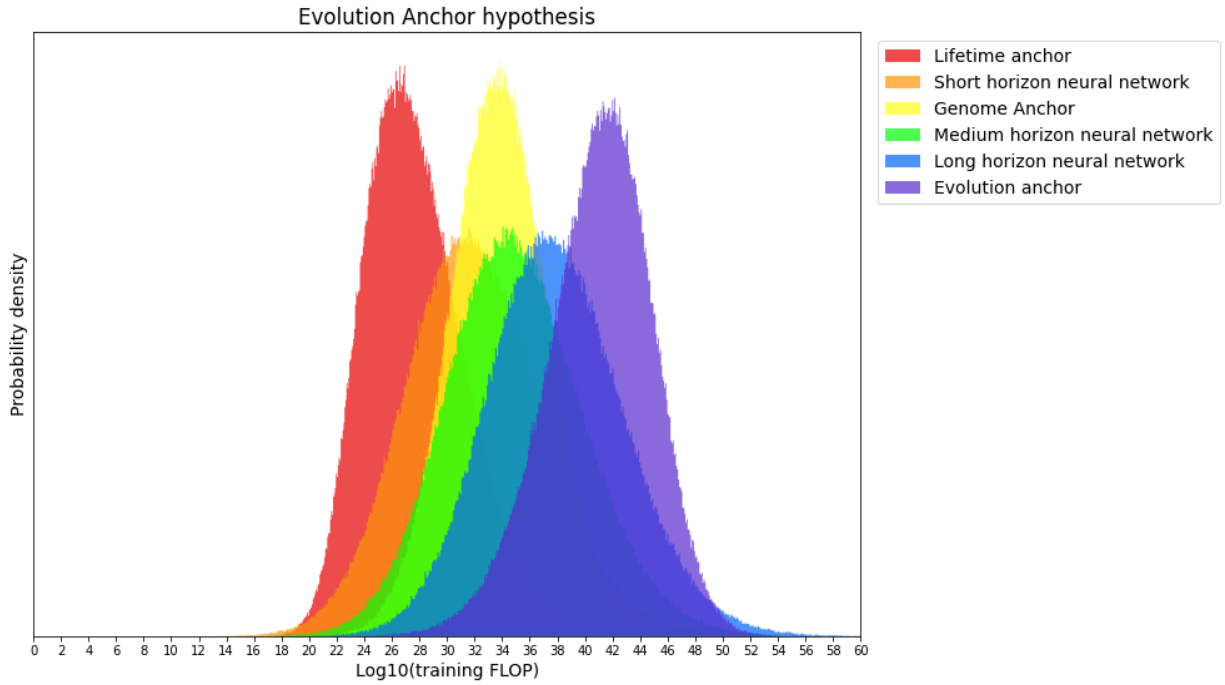
As with the Lifetime Anchor hypothesis, I am skeptical that this is the most appropriate biological anchor to lean on in generating our 2020 compute requirements distribution, but in this section I will try to condition on the assumption that it is the most informative anchor.

The amount of computation done over evolutionary history can roughly be approximated by the following formula: (Length of time since earliest neurons emerged) * (Total amount of computation occurring at a given point in time). My rough best guess for each of these factors is as follows:

- **Length of evolutionary time:** Virtually all animals have neurons of some form, which means that the earliest nervous systems in human evolutionary history likely emerged around the time that the Kingdom *Animalia* diverged from the rest of the Eukaryotes. According to timetree.org, an online resource for estimating when different taxa diverged from one another, this occurred around ~6e8 years ago. In seconds, this is ~1e16 seconds.
- **Total amount of computation occurring at a given point in time:** This blog post attempts to estimate how many individual creatures in various taxa are alive at any given point in time in the modern period. It implies that the total amount of brain computation occurring inside animals with very few neurons is roughly comparable to the amount of brain computation occurring inside the animals with the largest brains. For example, the population of nematodes (a phylum of small worms including *C. Elegans*) is estimated to

be ~1e20 to ~1e22 individuals. Assuming that each nematode performs ~10,000 FLOP/s,[19] the number of FLOP contributed by the nematodes every second is ~1e21 * 1e4 = ~1e25; this doesn't count non-nematode animals with similar or fewer numbers of neurons. On the other hand, the number of FLOP/s contributed by humans is (~7e9 humans) * (~1e15 FLOP/s / person) = ~7e24. The human population is vastly larger now than it was during most of our evolutionary history, whereas it is likely that the population of animals with tiny nervous systems has stayed similar. This suggests to me that the *average* ancestor across our entire evolutionary history was likely tiny and performed very few FLOP/s. I will assume that the "average ancestor" performed about as many FLOP/s as a nematode and the "average population size" was ~1e21 individuals alive at a given point in time. This implies that our ancestors were collectively performing ~1e25 FLOP every second on average over the ~1 billion years of evolutionary history.

This implies that the total amount of computation done over the course of evolution from the first animals with neurons to humans was (~1e16 seconds) * (~1e25 FLOP/s) = **~1e41 FLOP**. Here, I generated a distribution over training computation requirements for the Evolution Anchor hypothesis in purple:



To generate this distribution,

- I first generated a distribution for the amount of computation done in evolutionary history, incorporating my uncertainty about the length of the evolutionary period, the amount of computation that might be occurring in a nematode's nervous system, the average

19 The appropriate starting point for the purposes of this exercise may be conception, given that the work of translating from the genetic code into a brain development plan begins there, but it wouldn't make a material difference to the estimate.

ancestor population size, and how much more or less computation we might need compared to the evolution anchor (using the same distribution generated in Part 1 for how much better or worse ML architectures might be compared to brain architectures) -- I have not investigated these estimates very thoroughly and I expect they could easily be improved with more research.
- I then multiplied that distribution by a skew-log-normal distribution with a median of ~0 OOM, a standard deviation of ~5 OOM, and a left skew. This expresses the view that that researchers could train a transformative model using as much computation as the amount of computation done in evolutionary history, and there is a substantial chance that smaller amounts would be sufficient as well, with the probability decaying slowly at smaller and smaller levels of computation.

Note that depending on the particular spirit of the hypothesis we are conditioning on (that the anchor of "FLOP done over the course of evolution" is the most appropriate biological anchor to use), I think we could argue for shifting the distribution further to the left rather than centering it exactly where the evolution anchor is centered -- just as I argued above that the Lifetime Anchor hypothesis distribution should be centered ~3 OOM to the right of its biological anchor. Below, I discuss some reasons to expect that we should be able to train a transformative model with substantially less computation than evolution. However, I have chosen to leave it alone in this context because:

- I wanted to simplify the report somewhat, and this hypothesis is the one I have spent the least time thinking about.
- There are plausible arguments that I have underestimated true evolutionary computation here in ways that would be somewhat time-consuming to correct: for example, it may be that we should somehow attempt to incorporate the evolution of precursors to true neurons, or that for early ancestors with very few neurons the amount of computation it would take to appropriately simulate their DNA would dominate the amount of computation to simulate their nervous systems. Not shifting the Evolution Anchor hypothesis distribution further to the left -- despite believing that there are relatively straightforward ways we could improve upon evolution, such as reducing population sizes or crafting less noisy fitness signals -- feels like an easy fudge to attempt to make up for this.
- The Long Horizon Neural Network hypothesis (in blue above) makes a number of reasonable assumptions, and I consider to be an attractive "naive default" view -- and it is actually relatively close to the Evolution Anchor hypothesis already. Shifting the Evolution Anchor hypothesis multiple OOMs to the left based on inside-view considerations of how we might improve on natural selection would cause it to be centered to the *left* of the Long Horizon Neural Network hypothesis. This feels somewhat strange because the spirit of this hypothesis involves believing that our optimization algorithms will be *less* efficient than SGD.
- I expect that some ML researchers would want to argue that we would need substantially *more* computation than was performed in the brains of all animals over evolutionary history; while I disagree with this, it seems that the Evolution Anchor hypothesis should place substantial weight on this possibility.

Like the Human Lifetime Anchor hypothesis, this hypothesis makes no assumptions about the size of the model, estimating training FLOP directly from a biological anchor. This could be achieved by training a very large model (e.g. one with ~1e18 FLOP / subj and a similar number of parameters) using an amount of data that is in-line with extrapolations from existing models, or by training a smaller model using a lot more data than would be predicted from these extrapolations, or by training a model that grows in size over the course of the training run as animals' brains did over the course of evolutionary history, etc.

I think it is very likely that this hypothesis or some cheaper hypothesis would work out given 2020 architectures and algorithms; I discuss this more below.

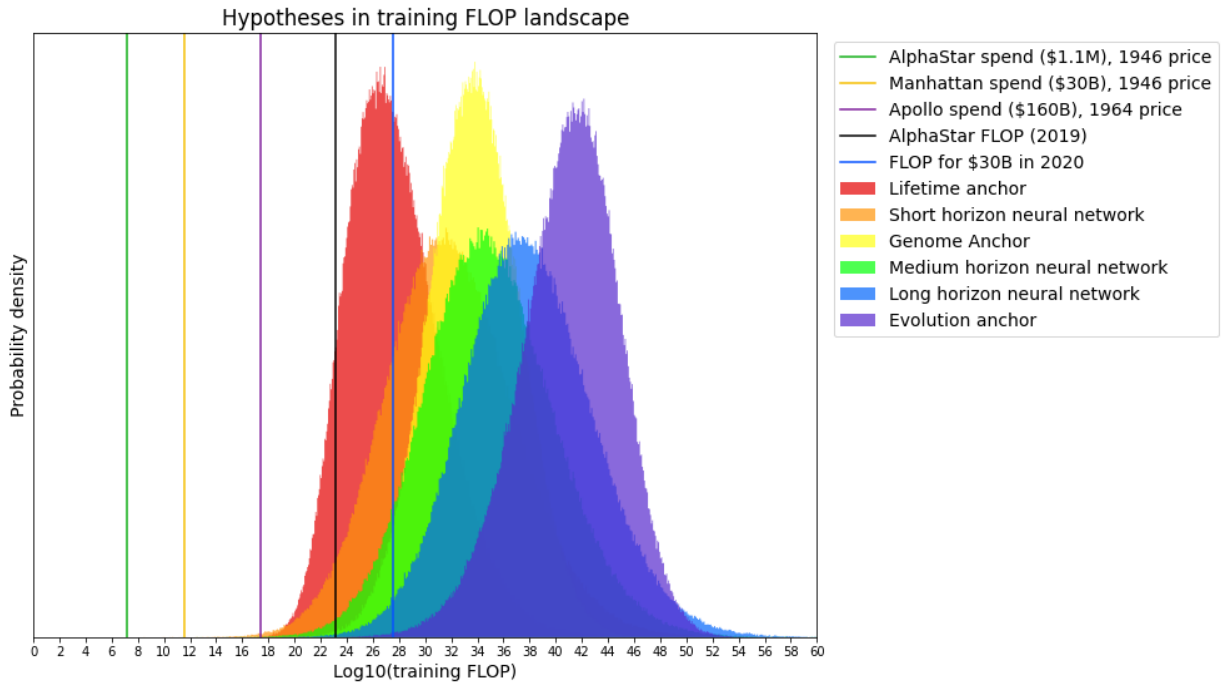
2020 training computation requirements distribution

In this section, I will attempt to use the hypothesis distributions generated in the previous sections to construct an overall 2020 compute requirements distribution. I will:

- Adjust the hypothesis distributions so that they assign a lower probability to the possibility that very low levels of computation (e.g. as much computation as models today are trained with) would be sufficient (more).
- Attempt to estimate the probability the amount of computation that would be required to train a transformative model in 2020 is much larger than any hypothesis would predict (more).
- Assign weights to each of the hypothesis distributions based on the qualitative discussions of the plausibility of each hypothesis given in the previous sections (more).
- Generate a probability distribution by creating a weighted mixture of the different biological anchor hypothesis distributions and the hypothesis representing "none of the above" (more).

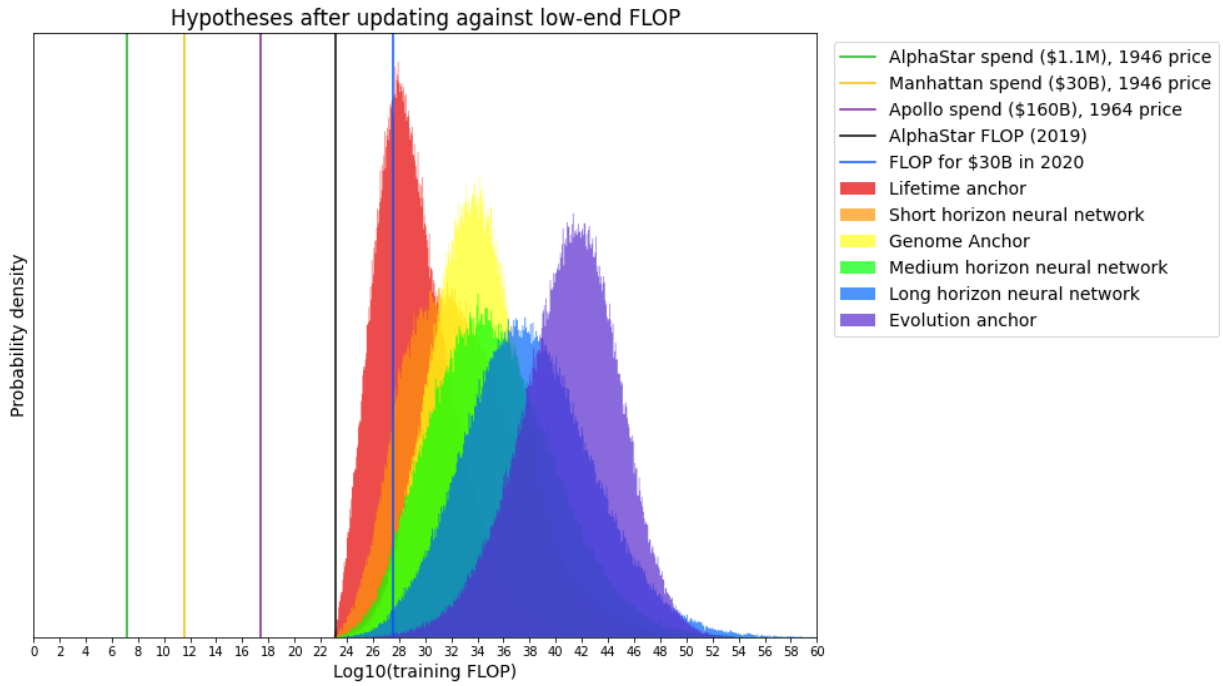
Updating against levels of computation that are already affordable

As I mentioned above, some hypotheses -- most notably the Lifetime Anchor hypothesis -- assign non-trivial probability to levels of computation small enough that we can already afford to do training runs of that size. To see this more clearly, we can display the hypotheses within the context of a landscape of FLOP levels that would have been attainable for various prices in the present or past:



The dotted lines in this landscape mark levels of computation that could theoretically have been purchased at various points in the past with various levels of investment (as well as markers for the median estimate of lifetime computation and evolution computation); see this appendix for details on that landscape. The black dotted line is the amount of computation that the AlphaStar training run used (~1e23 FLOP); the grey dotted line further to the right is the amount of computation that I estimate could be purchased for ~$30B in 2020.

I think it is unlikely that the amount of computation that would be required to train a transformative model is in the range of AlphaStar or a few orders of magnitude more -- if that were feasible, I would expect some company to have already trained a transformative model, or at least to have trained models that have already had massive economic impact. To loosely approximate a Bayesian update based on the evidence from this “efficient markets” argument, I truncate and renormalize all the hypothesis probability distributions:



Here, I have assumed that the probability that the required amount of compute is smaller than the amount of compute used to train AlphaStar (~1e23) should be set to ~0. Additionally, I assume that we have some amount of evidence that it's unlikely to be between 1e23 and 1e27, with the strength of that evidence getting weaker for higher values.[20] AlphaStar cost around ~$1M to train, and it seems likely that AI companies would have been willing and able to spend much more than that if they had reason to expect substantial profit. I assume that there is no particularly strong evidence against values larger than ~1e27 purely from an "efficient markets" standpoint.

To the extent that the prior (untruncated) distribution of a hypothesis assigned probability to levels of computation that we have updated against, we should consider that hypothesis less credible overall. The truncation removed ~30% of the probability mass in the Lifetime Anchor distribution and ~10% of the mass from the Short Horizon Neural Network distribution; the others were virtually unaffected, losing <5% probability mass.

Note that this loss of probability mass is not shown visually in the plots above, because the plotting software I use automatically renormalizes the histograms so that the total area under their curve adds up to 1; this means that when a distribution gets *narrower* -- as the Lifetime Anchor hypothesis did when low end values were cut off -- it also gets *taller.* Instead, I take this into account below when I assign weights to different hypotheses and combine them into a mixture distribution.

20 "All models were trained for a total of 300 billion tokens." Brown et al 2020, pg 8.

Probability that the required FLOP is larger than all hypotheses predict

What is the probability that 2020 training computation requirements for a transformative model are larger than the amount predicted by the Evolution Anchor hypothesis -- that is, that none of the biological anchor hypotheses would work out in any form[21] (even given access to the requisite data and environments)?

I would estimate a ~10% probability that 2020 compute requirements are larger than the Evolution Anchor hypothesis. From my current perspective, I think it could be defensible to assign a probability closer to 25% instead, but I don't currently see how someone would defend a probability of 50% or more (although I may be missing some relevant arguments). My reasoning is as follows:

- ML models have already made substantial progress on several tasks analogous to the types of behaviors animals were under strong natural selection pressure to perform well (e.g. object recognition and motor control) using vastly less training computation than it took to evolve those abilities. For example, if our ancestors collectively performed ~3e24 FLOP per second, then the amount of computation performed in the entire AlphaStar training run (~1e23) is the equivalent of less than one second of evolutionary history, implying that AlphaStar should only be as capable as the earliest animals with neurons (believed to be tiny jellyfish-like creatures).
- There are also some specific ways it seems that we could improve upon the "simulate natural selection" baseline, *prima facie*. For example, population sizes are a consequence of the carrying capacity of an ecological niche rather than being tuned to minimize the amount of computation used to evolve intelligent animals; it seems likely that they were far too large from a computational-efficiency standpoint. Additionally, the genetic fitness signal in the natural environment was often highly noisy, whereas we could plausibly design artificial environments which provide much cleaner tests of precisely the behaviors we are looking to select for.
- The Evolution Anchor hypothesis posits that training a transformative model would require computation levels almost *twenty orders of magnitude* larger than anything we could reasonably afford today; as mentioned above, this allows for room to train a model more than *eight orders of magnitude* larger than models we are training today, *and* to train it using much more data than extrapolations from current models would predict. I think there is little reason to expect that even domain experts would be justified in making a confident judgment that such an unprecedentedly huge training run couldn't accomplish a transformative task; most experts most of the time are not entertaining this possibility. If the Evolution Anchor hypothesis would work out, this is easily compatible with the views of experts who are highly pessimistic about machine learning-based transformative AI.

21 "All models were trained for a total of 300 billion tokens." Brown et al 2020, pg 8.

Assigning probabilities to each of the biological anchor hypotheses

If there is a ~90% probability that the Evolution Anchor hypothesis or something cheaper would work out given 2020 architectures and algorithms, how should we distribute that credence across hypotheses? Which hypothesis distribution is most likely to represent the *minimum* required amount of computation given 2020 architectures and algorithms?

This is more debatable and dependent on intuitive priors that may differ from person to person than the question of whether the Evolution Anchor hypothesis would work out. I have given some qualitative commentary in the sections above on how plausible each of the different hypotheses seems to me; here I will attempt to translate that into rough quantitative weights.

Long Horizon Neural Network as a "naive" default view

I consider the Long-Horizon Neural Network hypothesis to be a "naive default" view that errs conservative. This hypothesis assumes that we would need a model roughly ~1 OOM larger than the human brain to solve a transformative task, and then makes fairly standard or somewhat conservative assumptions about every other parameter:

- It assumes that a transformative model would look similar to contemporary architectures, and so would perform about ~1-100 FLOP per parameter per subjective second.
- It assumes that a transformative ML problem would likely involve meta-learning of some kind, and that meta-learning would have an effective horizon length of multiple subjective months or years.
- It assumes that the amount of data required to train this model will scale in a similar way to existing models -- that the number of instances of "trying to learn a new skill over the course of multiple subjective months" that the model will need to observe will scale close to linearly with its parameter count (an exponent of ~0.8).
- It assumes that there is no "shortcut" to training the relevant long horizon cognitive skills using decomposition, human feedback, and so on.

If it is the case that the computation that would be required to train a transformative model in 2020 is larger than the amount estimated by this hypothesis, then the reason is likely some combination of:

- **Our optimization techniques are fundamentally limited:** It won't be possible to use SGD (or other optimization techniques researchers could design in a few years of effort) to find a transformative model from a search space that contains one -- perhaps because there would be astronomically many bad local minima in the loss landscape for every transformative model, and any optimization technique researchers try would likely get stuck in one of those local minima.
- **The quantitative estimates are too small:** Contemporary optimization techniques *would* work for finding a transformative model at *some* scale, but the amount of computation required would be larger than the amount estimated by any hypothesis distribution -- perhaps because we would actually need a model many orders of magnitude larger than the human brain (or the human brain is much more powerful than

we think), or because the amount of training experience needed would be orders of magnitude larger than ten data points per parameter.

Both of these are possible, but I don't see a strong affirmative reason to be confident in either of these possibilities:

- Simple variants of SGD have been able to produce models capable of a wide variety of useful behaviors -- and some of these behaviors, most notably language modeling, seem like important sub-skills for many possible transformative tasks.
- There are some skills that seem important to many transformative tasks that our models have not yet displayed clearly, notably quickly learning new skills. However, as I argued above, it is plausible (and consistent with the biological anchors framework) that this is because models are not large enough for meta-learning to work well rather than because of a fundamental limitation.
- More generally, this framework predicts that machine learning models today should be substantially less "capable" than mice and only somewhat more capable than bees; this does not seem obviously inconsistent with what we in fact observe.

I would estimate an ~80% probability that the Long-Horizon Neural Network hypothesis or a cheaper hypothesis would work out given 2020 algorithms and architectures. I can see how someone would assign a probability closer to ⅓, but I don't currently see how someone could arrive at a probability much lower than that.

How likely is it that more aggressive hypotheses are right?

Hypotheses which predict that smaller amounts of computation will be sufficient are committed to denying one of the "naive" assumptions laid out above:

- The other Neural Network hypotheses assume that models can have a transformative impact without being able to plan over long horizons, or else that we can train models to plan adequately over long horizons without actually giving them direct experience with many long horizon episodes.
- The Genome Anchor hypothesis assumes that the amount of training data required to learn long horizon behaviors can best be estimated by anchoring to the amount of parameters in the human genome, rather than the amount of parameters in a typical neural network running on ~1e16 FLOP / subj sec.
- The Lifetime Anchor hypothesis does not make specific statements about training methods, but the most likely ways it could turn out to be true is if a) the model size required to solve a transformative task will be substantially smaller than the human brain *and* very short horizons will be sufficient, or b) researchers can relatively easily design a large model which learns some transformative task as efficiently as a human baby could (which may involve "hard-coding" a number of the priors such as intuitive physics, intuitive psychology, or proclivity for learning languages that human babies may have acquired over the course of natural selection).

On the other hand, the Evolution Anchor hypothesis relaxes the "default assumptions" further, allowing for the model size to be much larger than the human brain, and/or for sample

complexity to scale much more poorly than for existing ML problems, and/or for horizons to be much longer than one subjective year.

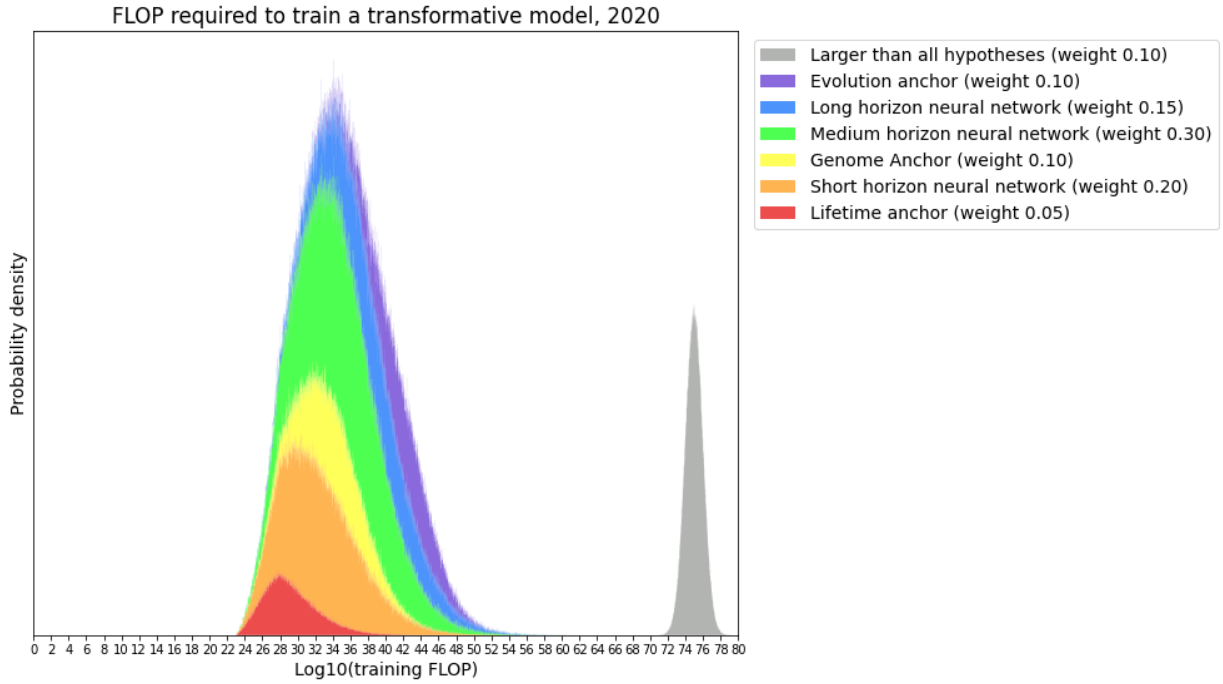
For each of these six hypotheses, I'll assign a probability estimate to the event that the *minimum* amount of computation required to train a transformative model is distributed roughly as that hypothesis would predict:

- **Lifetime Anchor hypothesis:** I consider the Lifetime Anchor hypothesis to be unlikely. As we saw above, there is almost a factor of ~2 update against this hypothesis from the efficient markets argument; it also seems to be out of line with other evidence we have about ML systems -- for example, we have substantial evidence that training a model to solve a task consistently requires many more examples than teaching a human an analogous skill. I'll assign a ~5% probability to this hypothesis.
- **Short Horizon Neural Network hypothesis:** I think that there is a substantial possibility that effective horizon length may be much shorter than the conservative assumption made by the Long Horizon Neural Network hypothesis from some combination of proxy reward signals, decomposition, and short horizon sub-skills transferring to longer horizons (see above). In particular, several researchers I have spoken to consider it plausible that learning the skill of "predict what will happen next" will dominate the cost of training. I'll assign a ~20% probability to this hypothesis.
- **Genome Anchor hypothesis:** While I think it is unlikely that researchers could quickly discover an architecture which structurally behaves like the genome while having a similar scaling behavior to neural networks, I do find the genome anchor to be somewhat plausible as a way to think about what fraction of the training data for a neural network may need to be long horizon if a mixture of horizon lengths is required (although I don't currently think of it as strongly privileged over other ways of thinking about that question). I'll assign a ~10% probability to this hypothesis. This makes the total probability assigned to the three lowest-computation hypotheses ~35%.
- **Medium-Horizon Neural Network hypothesis:** I currently consider this hypothesis to be the most likely. It feels as if we should be able to solve substantially more impactful tasks with an effective horizon length in the range of subjective hours or days compared to subjective seconds or minutes. If the "naive" effective horizon length of meta-learning is multiple subjective months, it seems highly plausible that some combination of the reasons described above may bring that down by ~2-3 OOM into the Medium Horizon range (as opposed to ~5-6 OOM into the Short Horizon range). I'll assign a ~30% probability to this hypothesis.
- **Long Horizon Neural Network hypothesis:** The above implies that the probability assigned to this hypothesis is ~15%, for a total of ~80% probability assigned to all the hypotheses which estimate lower training computation requirements than Evolution Anchor.
- **Evolution Anchor hypothesis:** The above implies that the probability assigned to this hypothesis is ~10%, for a total of ~90% probability assigned to all of the biological anchor hypotheses combined.

The weighted mixture of biological anchor hypotheses

To generate a combined 2020 compute requirements distribution from the hypotheses, I need to first somehow represent the possibility that the minimum amount of computation required to train a transformative model given current architectures and algorithms is larger than the Evolution Anchor hypothesis.

Because that possibility is outside the biological anchors framework, I don't have a way to estimate exactly *how* much larger the required amount of computation would be in that world. As a stand in, I simply generate an arbitrary "dummy distribution" to represent this possibility, and create a weighted sum of the seven different distributions using the weights described above:



This dummy distribution plays no role in the calculation of timelines and its location is completely irrelevant**;** it is purely a visual aid which allows the total size of the combined distribution containing the six hypotheses to grow or shrink based on the total probability assigned to "at least one hypothesis is correct" (otherwise the plotting software would automatically enforce that the total density is 1).

In the Part 4, I describe how I generate a probability distribution over when the amount of compute required to train a transformative model may become affordable; in that calculation, I make the somewhat conservative assumption that **if 2020 compute requirements are larger than the Evolution Anchor, we will not develop TAI anytime in the rest of this century.**

Again, it is important to emphasize that the distribution above is conditioned on 2020 architectures and algorithms; compute requirements distributions for future years will put more

probability mass on low levels of computation due to algorithmic improvement, and may have probability mass on amounts of computation even smaller than ~1e23.

Forecasting TAI with biological anchors

Part 4: Timelines estimates and responses to objections

Author: Ajeya Cotra
Date: July 2020

This report emerged from discussions with our technical advisors Dario Amodei and Paul Christiano. However, it should not be treated as representative of either of their views; the project eventually broadened considerably, and my conclusions are my own.

This is a work in progress and does not represent Open Philanthropy's institutional view. We are making it public to make it easier to gather feedback, to help inform others' thinking in the effective altruism community, and to allow for follow-on work outside of Open Phil. However, we may edit it substantially in the future as we gather feedback from a broader audience and investigate open questions. Accordingly we have not done an official publication or blog post, and would prefer for now that people not share it widely in a low-bandwidth way (e.g., just posting key graphics on Facebook or Twitter).

The report has been divided into four Google docs to load faster. ***This is Part 4; the first part is here, the second part is here, and the third part is here.*** *Additional materials (collected in this folder):*

- *Quantitative model: the Python notebook Biological anchor hypotheses for 2020 training computation requirements; a template spreadsheet When required computation may be affordable; and my best guess, conservative, and aggressive forecasts.*
- *Supplemental materials: a document containing various appendices; a folder of figures for the report; the spreadsheet Extrapolations of data and compute to train models; and the Python notebook Compute price trends, which draws on data in this folder.*

In Part 1, I provided an overview of the framework and estimates, provided definitions for key abstractions used in the model, and generated an estimate for the number of FLOP / subj sec of a transformative model. In Part 2, I reviewed theoretical and empirical evidence about training data requirements for a transformative model, introduced the concept of horizon length, and estimated how training data requirements may scale with parameter count for a transformative ML problem. In Part 3, I discussed in more detail the Neural Network hypotheses and other biological anchor hypotheses, and combined them into a 2020 training FLOP requirements distribution.

In this part, I will:

- Explain my best guess, conservative, and aggressive forecasts for when the amount of computation required to train a transformative model may become affordable, incorporating forecasts for algorithmic progress, hardware prices, and spending on computation (more).
- Explain how I translate the outputs of this exercise into views on timelines, including a median estimate for TAI, probability of TAI in this century, and probability of TAI by 2036 (more).
- Address several high-level questions and objections to the framework (more).
- Briefly describe several open questions for further investigation (more).

Timelines for when required computation is available

Generating estimates for when the compute to train a transformative model may become affordable involves modeling three additional considerations besides the current compute requirements distribution: how compute requirements are likely to fall over time due to **algorithmic progress**, how the amount of computation available for a given price is likely to increase over time due to falling **compute prices**, and how the amount of money an AI project is **willing to spend on compute** to train a potentially transformative model would increase over time.

This spreadsheet demonstrates how I model these three quantities to generate an estimate for when the amount of computation required to train a transformative model may become affordable. In this section, I will:

- Explain how the quantitative model works at a high level and describe my forecasts for effective FLOP per dollar, willingness to spend on computation to train a transformative model, and algorithmic progress (more).
- Describe the probability distribution over when the computation required to train a transformative model will be available according to these best guess forecasts (more).
- Briefly discuss what I consider to be conservative and aggressive inputs to the model and the distributions generated by these inputs (more).

As I explained in Part 1, the focus of the vast majority of my research has been articulating the framework, generating the hypotheses and generating the 2020 compute requirements distribution; I have spent substantially less time thinking about these other considerations. **These numbers are much more tentative and unstable than my hypothesis distributions**, and I expect many readers to have substantially more information and better intuitions about them than I do; **I strongly encourage interested readers to generate their own versions of these forecasts with this template spreadsheet.**

Best guess forecasts for hardware, spending, and algorithms

My spreadsheet generates forecasts **starting from 2025 through the end of the century (2100).** The reason I begin at 2025 rather than in the current moment (mid-year 2020) is that I expect there will be a short-lived period of extremely rapid scaleup in spending on computation for ML training runs over the next few years, and that growth in spending over the subsequent few decades will likely be substantially slower (see this appendix for more detail). Beginning the forecast after I expect this spurt in spending to have ended helps to simplify the modeling.

In 2025, I assume that it is possible to purchase ~4e17 FLOP per dollar, and that the most expensive training run in that year would cost ~$1B; this means that I am expecting the largest training by 2025 would perform $1e9 * 4e17 FLOP/$ = ~4e26 total FLOP.

For each year in the period from 2025 through 2100, the total FLOP that could be used to train a transformative model is computed by multiplying the FLOP per dollar in that year with the

willingness to spend. The total FLOP value for a given future year Y must be compared to the year Y training computation requirements distribution: the probability mass that lies to the left of the total FLOP available represents the probability that the amount of computation required to train a transformative model will be affordable by year Y. Computing the year Y training computation requirements distribution requires estimating algorithmic progress.

In the rest of this section, I will cover:

- My forecast for hardware prices (more).
- My forecast for willingness to spend on computation to train a transformative model (more).
- My forecast for incremental and "breakthrough" algorithmic progress (more)

FLOP per dollar forecast

See this appendix for details on historical FLOP per dollar and my forecast up to 2040. I draw on research conducted by former Open Philanthropy Research Analyst Kathleen Finlinson in this section.

As of 2020, I believe a well-optimized ML training run can perform a bit over ~1e17 "useful" operations per dollar spent on GPUs or TPUs, assuming that they are able to achieve about ~⅓ utilization of each GPU or TPU on average over the training run. In this section, I describe my tentative best guess forecasts for how FLOP per dollar will increase from ~2025 to 2100.

Initial FLOP per dollar and growth rate in 2025

I assume that in 2025, AI labs will be able to perform about ~4e17 useful FLOP per real dollar spent, which is about 4x my estimate for the number of useful FLOP per dollar available in 2020. I chose 4e17 FLOP per dollar because two of our technical advisors felt that a roughly ~4x improvement over the current value was plausible over the next five years given promising very recent improvements. I expect that more closely examining recent developments in GPUs and speaking to hardware industry experts could relatively easily refine this number.

I also assume that effective FLOP per dollar is doubling roughly once every 2.5 years around 2025. This is slower than Moore's law (which posits a ~1-2 year doubling time and described growth reasonably well until the mid-2000s) but faster than growth in effective FLOP per dollar from ~2008 to 2018 (a doubling time of ~3-4 years). In estimating this rate, I was attempting to weigh the following considerations:

- Other things being equal, the recent slower trend is probably more informative than older data, and is fairly likely to reflect diminishing returns in the silicon chip manufacturing industry.
- However, the older trend of faster growth has held for a much longer period of time and through more than one change in "hardware paradigms." I don't think it makes sense to extrapolate the relatively slower growth from 2008 to 2018 over a period of time several times longer than that
- Additionally, a technical advisor informs me that the NVIDIA A100 GPU (released in 2020) is substantially more powerful than the V100 that it replaced, which could be more consistent with a ~2-2.5 year doubling time than a ~3.5 year doubling time.

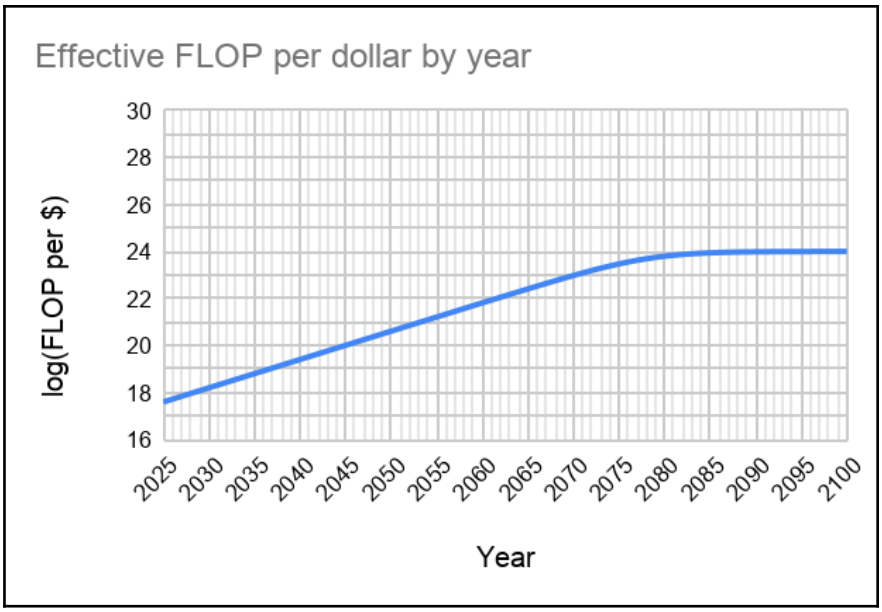
Maximum FLOP per dollar achievable by 2100

How much room is there for computing technology to improve over the course of 75 years -- a period of time about as long as the entire history of computing from early vacuum tube computers in the forties to 2020?

My very uncertain best guess estimate is that we could achieve a maximum of ~1e24 FLOP per dollar by the end of the century, a factor of about ~2 million improvement from 4e17. **This estimate feels particularly unstable and under-informed to me.** Some of the considerations going into my estimate:

- It seems that a factor of ~140 improvement can be achieved by around ~2040 through a variety of relatively straightforward optimizations such as squeezing out the last bits of transistor efficiency improvement, reducing precision, and taking advantage of economies of scale (more detail in this appendix). If this is true, that means that a factor of about ~15,000 improvement would need to be achieved via more exotic computing technologies such as optical computing, three-dimensional computing, reversible computing, or quantum computing to achieve a factor of 2 million improvement overall. Intuitively, this seems plausible or perhaps somewhat conservative to me.
- From 1964 to 2018, a period of 54 years, effective FLOP per dollar improved by ~11 OOM adjusting for inflation (details in this appendix). The improvement over that period was about twice as large, on a log scale, as the improvement I am predicting will be possible in the ~75 years from 2025 to 2100. It feels likely to me that the total improvement from 2025 to 2100 should be substantially smaller than the improvement from 1964 to 2018 due to diminishing returns to technological innovation and the possibility of hard physical limits such as Landauer's limit, but I am very uncertain how much smaller; about half as much progress in log-space (a little over 6 OOM vs 11 OOM) seems intuitively plausible.

I would need to more closely investigate proposed future computing substrates and/or claimed physical limits to improving computing efficiency to make a better-informed estimate of this factor; Open Philanthropy is likely to do some of this investigation in the future. In the meantime, my best-guess forecast for future FLOP per dollar looks like this:



In reality, I would expect that rather than transitioning from a ~2.5 year doubling time to some hard maximum value at ~1e24, we would slowly transition to a slower growth mode (or several slower growth modes) over the course of the century, and that computing technology would likely still be improving at some non-trivial rate in the year 2100. This forecast feels most solid and plausible out to ~2040 or so, beyond which it feels substantially more murky and likely incorrect.

Willingness to spend on computation forecast

See this appendix for more detail on this estimate.

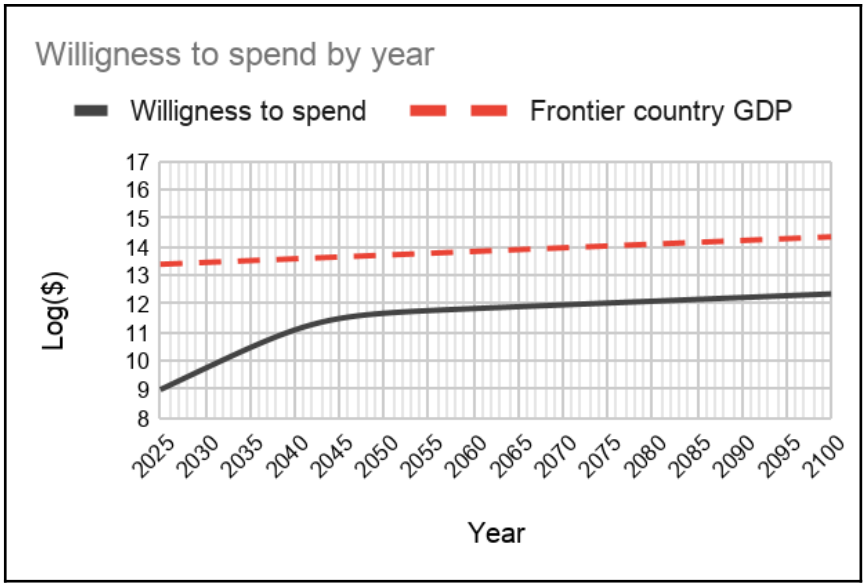
I have broken up my thinking about willingness to spend on computation to train a transformative model into short, medium, and long-run growth (links to more detailed discussion in the appendices):

- **Short-run growth:** By 2025, my best guess is that AI companies could scale up to spending ~$1B in computation for a single large-scale training run,[1] a ~3 OOM scaleup from the cost of computation for the final AlphaStar training run. This would require doubling spending on the most expensive training run about once every 6 months, which is consistent with what I understand of the recent pace of spending scaleup and the existing resources of AI companies such as Google.
- **Medium-run growth:** Beyond ~$1B training runs, I expect that raising money and justifying further spending would become noticeably more difficult for even very well-resourced labs, meaning that growth would slow after 2025. I made the assumption that it would slow to a 2 year doubling time, reaching $100B by 2040.

1 "Our survey population was all researchers who published at the 2015 NIPS and ICML conferences (two of the premier venues for peer-reviewed research in machine learning). A total of 352 researchers responded to our survey invitation (21% of the 1634 authors we contacted)." pg 1.

- **Long-run growth:** Anchoring to the costs of major technological megaprojects such as the Manhattan Project (which cost about ~1.7% of a year of GDP over five years) and the Apollo Project (which cost about ~3.6% of a year of GDP over its four peak years), I assumed that the maximum level of spending on computation for a single training run that could be reached is ~1% of the GDP of the largest country.

This results in the following forecast for willingness to spend on computation to train a transformative model (from 2025 to 2100):



Algorithmic progress forecasts

Note: *I have done very little research into algorithmic progress trends. Of the four main components of my model (2020 compute requirements, algorithmic progress, compute price trends, and spending on computation) I have spent the least time thinking about algorithmic progress.*

I consider two types of algorithmic progress: relatively incremental and steady progress from iteratively improving architectures and learning algorithms, and the chance of "breakthrough" progress which brings the technical difficulty of training a transformative model down from "astronomically large" / "impossible" to "broadly feasible."

For **incremental progress**, the main source I used was Hernandez and Brown 2020, "Measuring the Algorithmic Efficiency of Neural Networks." The authors reimplemented open source state-of-the-art (SOTA) ImageNet models between 2012 and 2019 (six models in total). They trained each model up to the point that it achieved the same performance as AlexNet achieved in 2012, and recorded the total FLOP that required. They found that the SOTA model in 2019, EfficientNet B0, required ~44 times fewer training FLOP to achieve AlexNet performance than AlexNet did; the six data points fit a power law curve with the amount of computation required to match AlexNet halving every ~16 months over the seven years in the

dataset.[2] They also show that linear programming displayed a similar trend over a longer period of time: when hardware is held fixed, the time in seconds taken to solve a standard basket of mixed integer programs by SOTA commercial software packages halved every ~13 months over the 21 years from 1996 to 2017.[3]

Grace 2013 ("Algorithmic Progress in Six Domains") is the only other paper attempting to systematically quantify algorithmic progress that I am currently aware of, although I have not done a systematic literature review and may be missing others. I have chosen not to examine it in detail because a) it was written largely before the deep learning boom and mostly does not focus on ML tasks, and b) it is less straightforward to translate Grace's results into the format that I am most interested in ("How has the amount of computation required to solve a fixed task decreased over time?"). Paul is familiar with the results, and he believes that algorithmic progress across the six domains studied in Grace 2013[4] is consistent with a similar but slightly slower rate of progress, ranging from 13 to 36 months to halve the computation required to reach a fixed level of performance.

Additionally, it seems plausible to me that both sets of results would overestimate the pace of algorithmic progress on a transformative task, because they are both focusing on relatively narrow problems with simple, well-defined benchmarks that large groups of researchers could directly optimize.[5] Because no one has trained a transformative model yet, to the extent that the computation required to train one is falling over time, it would have to happen via proxies rather than researchers directly optimizing that metric (e.g. perhaps architectural innovations that improve training efficiency for image classifiers or language models would translate to a transformative model). Additionally, it may be that halving the amount of computation required to train a transformative model would require making progress on multiple partially-independent sub-problems (e.g. vision *and* language *and* motor control).

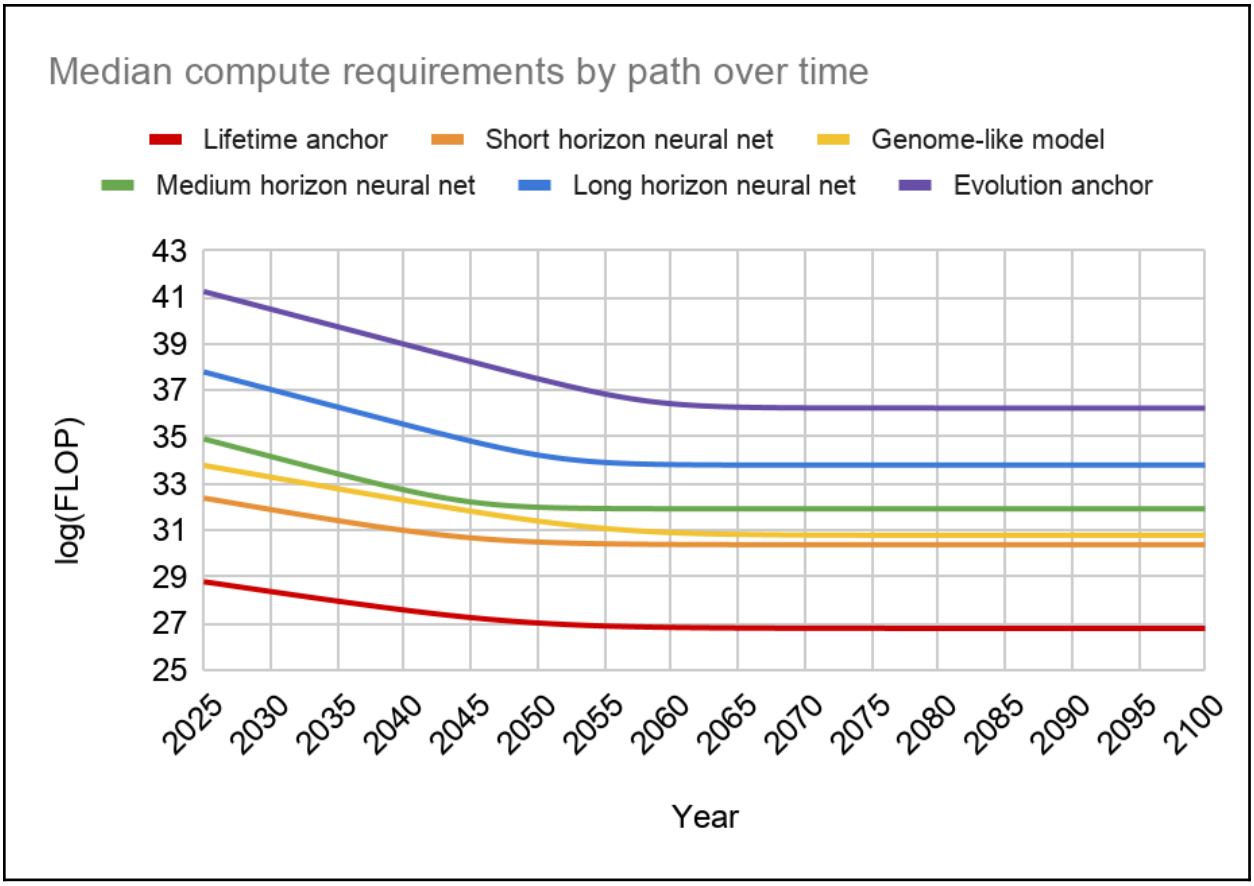
I have attempted to take the Hernandez and Brown 2020 halving times (and Paul's summary of the Grace 2013 halving times) as anchoring points and shade them upward to account for the considerations raised above. There is massive room for judgment in whether and how much to shade upward; **I expect many readers will want to change my assumptions here, and some will believe it is more reasonable to shade *downward*.**

In my spreadsheet, I chose to break down the algorithmic progress forecast by hypothesis rather than use a single value describing how the 2020 compute requirements distribution shifts to the left in future years. This is because hypotheses which predict that the amount of

2 Pg. 1.
3 Figure 1, pg. 2.
4 Note that this metric could be increasing without the labor force participation rate decreasing, because old jobs might be automated at the same rate that new jobs are being created. For example, self-driving cars may displace truck drivers, but many of those people who would have become truck drivers could instead move into other industries (including perhaps providing data or feedback to ML models).
5 Although my understanding is that some key complementary or responsive technologies, such as intercontinental ballistic missiles and various missile defense technologies, substantially affected how nuclear diplomacy played out.

computation required to train a transformative model is already very low (such as the Lifetime Anchor hypothesis) seems like they should also predict that further algorithmic progress would be difficult and there is not as much room to reduce compute requirements even further.

As with compute prices and willingness to spend, I assumed that algorithmic progress for each hypothesis could be described as a logistic curve, shown below:



I assumed that:

- Training FLOP requirements for the *Lifetime Anchor* hypothesis (red) are halving once every 3.5 years and there is only room to improve by ~2 OOM from the 2020 level -- moving from a median of ~1e28 in 2020[6] to ~1e26 by 2100.

6 Because this analysis is predicated on business-as-usual AI research continuing to progress, there should be a discount factor for the probability that there is an exogenous event that effectively halts AI progress, such as a large-scale disaster like nuclear war or a pandemic which affects enough people to stunt routine activity in many industries, sweeping government regulation of AI research such as blanket moratoriums on certain kinds of research or a large-scale and long-term withdrawal of investment due to a severe economic recession, etc. My guess is that the discount factor for "exogenous events that derail research" should be quite close to 1 for at least the next couple decades (e.g, I believe there is less than 10% probability of such an event occurring by 2040, which would result in a discount factor > 0.9).

Note that the way I have set up this analysis, **there should not actually be any further discount factor for the probability that a company would attempt the project;** that uncertainty is meant to be priced in to the definition of "willingness-to-spend" probability distribution which is multiplied by the cost of compute to determine the compute threshold for an attainable training run: the willingness-to-spend distribution is meant to be the probability for every level of spending S that if solving a transformative task had S

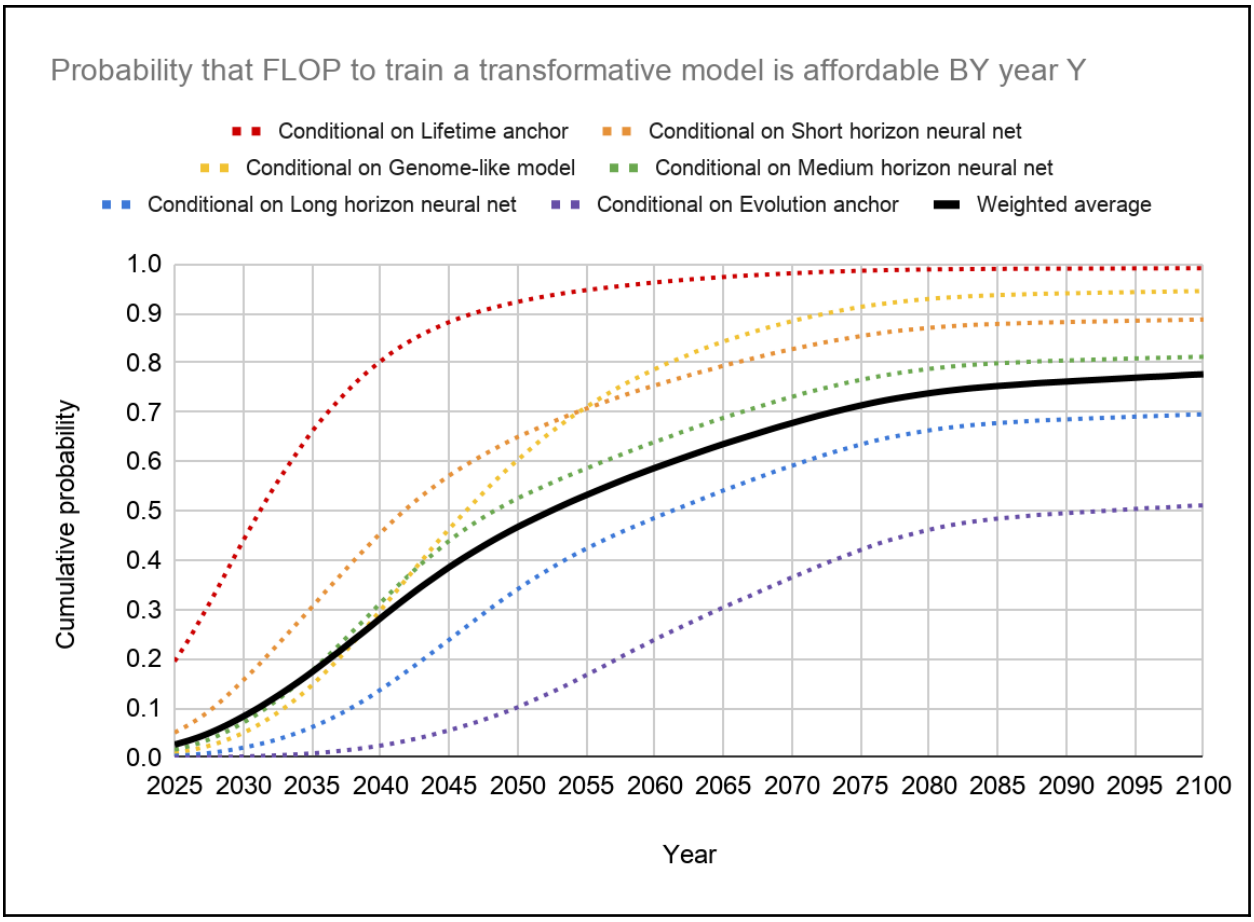
- Training FLOP requirements for the *Short horizon neural network* hypothesis (orange) are halving once every 3 years and there is room to improve by ~2 OOM from the 2020 level -- moving from a median of ~1e31 in 2020 to ~3e29 by 2100.
- Training FLOP requirements for the *Genome Anchor* hypothesis (yellow) are halving once every 3 years and there is room to improve by ~3 OOM from the 2020 level -- moving from a median of ~3e33 in 2020 to ~3e30 by 2100.
- Training FLOP requirements for the *Medium-horizon neural network* hypothesis (green) are halving once every 2 years and there is room to improve by ~3 OOM from the 2020 level -- moving from a median of ~3e34 in 2020 to ~3e31 by 2100.
- Training FLOP requirements for the *Long-horizon neural network* hypothesis (blue) are halving once every 2 years and there is room to improve by ~4 OOM from the 2020 level -- moving from a median of ~1e38 in 2020 to ~1e34 by 2100.
- Training FLOP requirements for the *Evolution Anchor* hypothesis (purple) are halving once every 2 years and there is room to improve by ~5 OOM from the 2020 level -- moving from a median of ~1e41 in 2020 to ~1e36 by 2100.

For **breakthrough progress**, I simply decrease the probability that the amount of computation required to train a transformative model is greater than all biological anchor hypotheses over time. In 2025, I estimate this probability is 10%, and by 2100 I assume that it drops to ~3%. For simplicity, I do not shift the relative share of probabilities across different hypotheses over time. A more thorough model of breakthrough progress would likely shift probability mass from higher-compute hypotheses to lower-compute hypotheses to reflect that over time researchers may discover how to make those hypotheses work out (for example, that they may discover a qualitatively different learning algorithm which is as efficient as the one used by the human brain).

Distribution generated by best guess assumptions

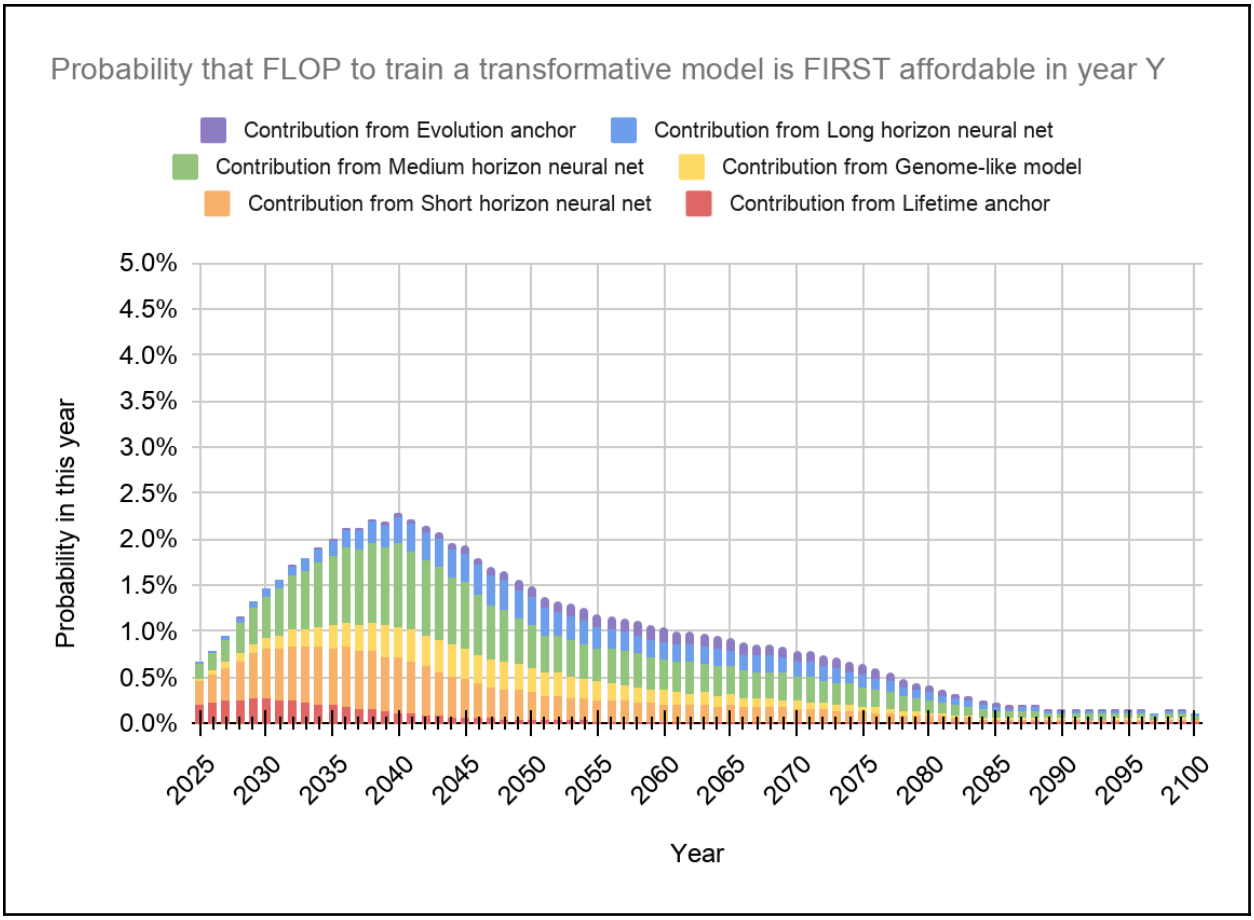
When I input the assumptions described in the last few sections into my model, the cumulative distribution over when compute required to train a transformative model may be affordable looks like this:

difficulty, some group would actually do it. Relatedly, if investment stalls or diminishes in the near-term because ML systems are not delivering enough value to justify it, that is also meant to be handled within the context of the framework -- it should simply increase our credence that the technical difficulty of TAI is unattainably high and we must wait for research progress to shift the distribution further to the left before ramping up investment will make sense. In general I have tried to set up the framework such that for the most part we only need to adjust for events that are relatively exogenous to the all-in technical difficulty of TAI after the fact. (I think it's unlikely that I've fully succeeded in this aim, however.)



The black curve depicts the probabilities output by the weighted combination of hypotheses; the colored curves correspond to what the probability would be if we conditioned on a particular hypothesis. As discussed in the previous sections, the probabilities in the latter half of the century are likely slightly too low,

We can convert the cumulative distribution above into a distribution which depicts the probability for each year that the compute required to train a transformative model *first* becomes affordable in that year (with the contribution of each hypothesis to the total probability in that year shown):



As a rough guess, I would say that this distribution is a fairly good representation of my TAI timelines between ~2030 and ~2050. Before that point, my best guess probabilities would be slightly lower than what is depicted because of the possibility of other bottlenecks such as data or human labor. After that point, my best guess probabilities would be slightly higher than the ones depicted, because a) the FLOP per dollar curve and algorithmic progress curves saturate at a hard maximum value when in reality they would likely continue growing slowly, and b) as more time passses it is more likely that TAI may have arrived through other means besides training a unified transformative model (e.g. a collection of several pieces of software which collectively have a massive economic impact). In particular, it feels very unrealistic that this distribution depicts very little chance that TAI first arrives between ~2080 and 2100.

Conservative and aggressive estimates

In addition to the mainline distribution discussed above, I also generated aggressive and conserative probability distributions for when the computation required to train a transformative model may be affordable.

The “aggressive” set of assumptions are meant to capture the soonest estimate I could imagine arriving at simply by thinking more deeply about the information and high-level considerations I’m already aware of as I attempt to arrive at reflective equilibrium, as opposed to what I may come to believe if I learn surprising empirical information or try to take a qualitatively different

high-level approach to the question. Similarly, the “conservative” assumptions are meant to reflect the longest timelines I could realistically imagine arriving at without learning surprising information or taking a different approach.

In this section, I will:

- Briefly describe my aggressive and conservative assumptions (more).
- Discuss what probabilities would be reasonable to assign to TAI being developed by 2036 (more).
- Discuss what a reasonable *median* estimate for TAI timelines would look like (more).
- Discuss what probabilities would be reasonable to assign to TAI being developed by the end of the century (more).

For the most part, I generated the numerical inputs that generate my “aggressive” and “conservative” probability distributions by perturbing my “best guess” values, rather than reasoning about them afresh. I’ll go over them very briefly in this section.

Range of estimates for hypothesis probabilities

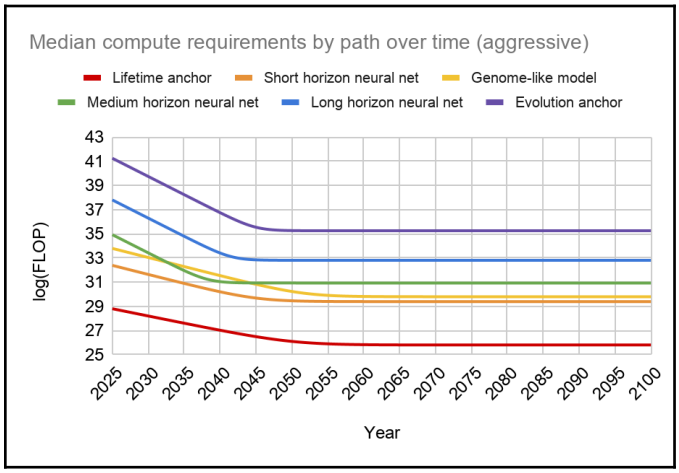
Under my best guess probabilities, the collective probability assigned to the low-compute hypotheses (Lifetime Anchor, Short Horizon Neural Network, and Genome Anchor) was 35%; under aggressive assumptions this is doubled to 70% and under conservative assumptions this is reduced to 15%. Conversely, the probability that no hypothesis will work is set to 10% on my best guess assumptions; my aggressive assumptions reduce this probability to 5% and my conservative assumptions increase it to 20%. The weights overall are given by this table:

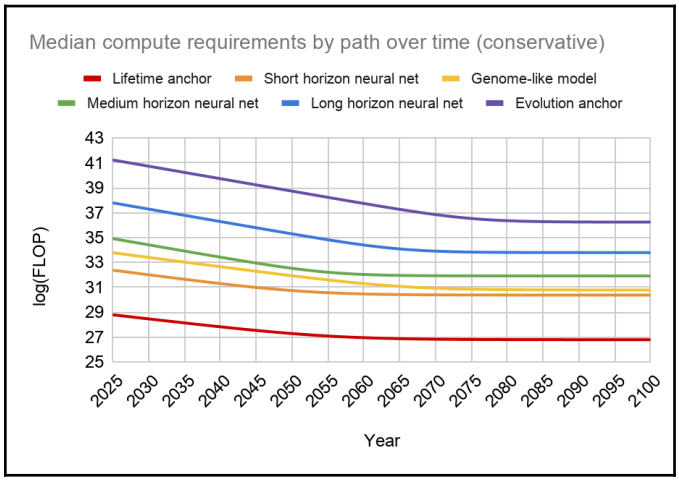
	Aggressive	**Best guess**	**Conservative**
Lifetime Anchor	10%	5%	2%
Short Horizon NN	40%	20%	8%
Genome Anchor	20%	10%	5%
Medium Horizon NN	10%	30%	15%
Long Horizon NN	10%	15%	30%
Evolution Anchor	5%	10%	20%
No hypothesis	5%	10%	20%

Range of forecasts for algorithmic progress

To generate my aggressive assumptions for incremental progress, I have simply assumed that an extra order of magnitude of progress is possible relative to my best guess assumptions about maximum possible progress for each hypothesis, and that the doubling time of algorithmic

efficiency for each of the hypotheses is one year shorter. Conversely, I assumed that the doubling time of algorithmic efficiency is one year *longer* for each hypothesis and that the maximum possible progress is 1 OOM *less* to generate conservative assumptions. Below are the curves for the aggressive and conservative assumptions, respectively:

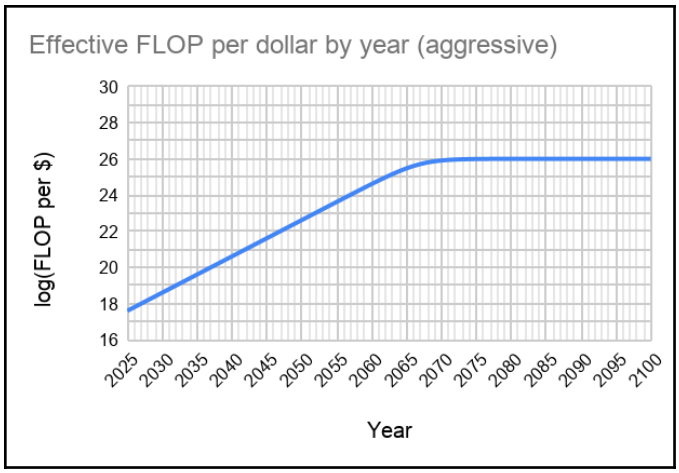


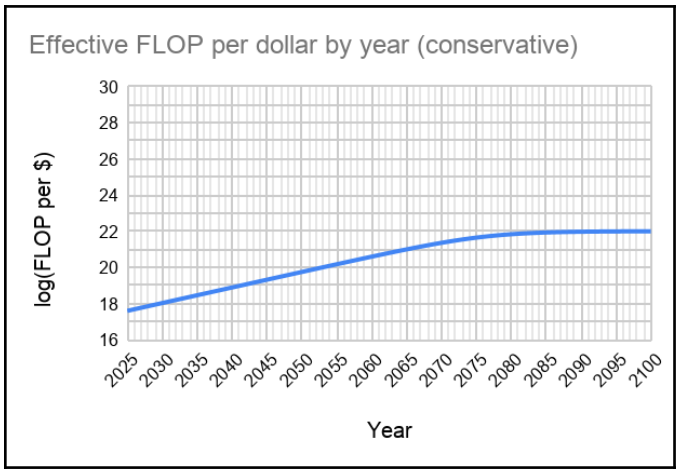


For “breakthrough progress”, I decrease the probability of “no hypothesis” from ~5% to ~1% over the course of the century in the aggressive model, and from ~20% to ~5% for the conservative model.

Range of forecasts for effective FLOP per dollar

To generate my aggressive assumptions about compute prices, I assumed that an extra 2 OOM of progress is possible relative to my best guess assumptions about maximum FLOP per dollar (for a maximum of ~1e26 FLOP / $) and that the doubling time of FLOP per dollar is one year shorter (~1.5 years). Conversely, I assumed that the doubling time of FLOP per dollar is one year *longer* (~3.5 years) and that the maximum possible progress is 2 OOM *less* (~1e22 FLOP / $) to generate conservative assumptions. Below are the FLOP per dollar curves for the aggressive and conservative assumptions, respectively:





Range of forecasts for willingness to spend

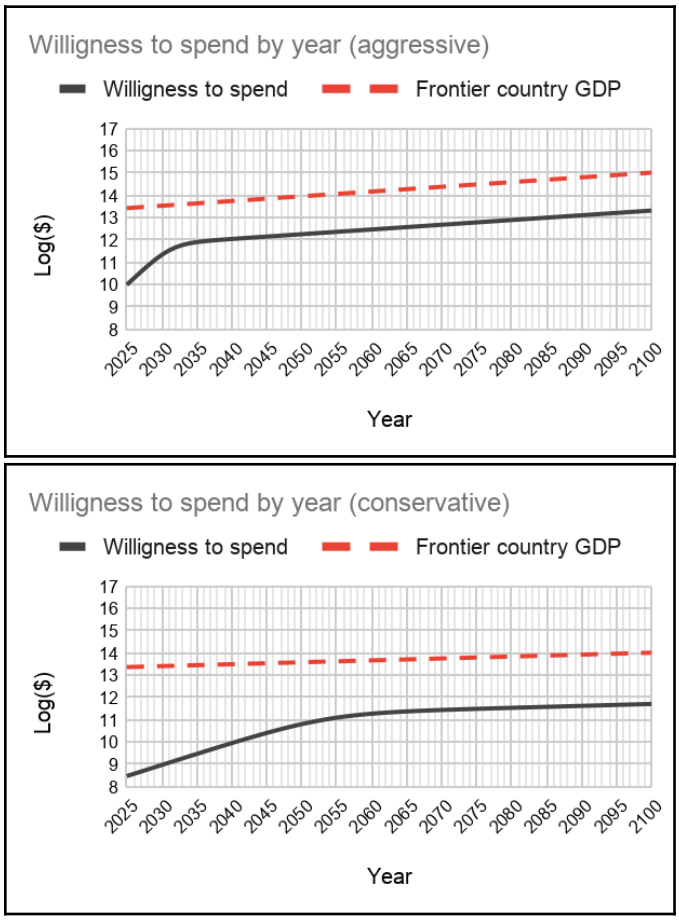
To generate my aggressive and conservative assumptions about spending, I assumed the following:

- **Short-term scaleup:** My best guess assumption was that spending on computation for the largest training could get to $1B by 2025; I bumped this up by 1 OOM to $10B for the aggressive assumption, and down 0.5 OOM to $300M for the conservative assumption.
- **Medium-term scaleup:** My best guess assumption was that willingness to spend on computation to train a potentially transformative model would double once every two years in the medium term to reach $100B by ~2040. My aggressive assumption is that it would double every year and reach $100B by ~2030, and my conservative assumption is that it would double every three years to reach $100B by ~2055.
- **Long-term spending:** My best guess assumptions were that the maximum willingness to spend on computation to train a transformative model would be ~1% of the GDP of the largest national economy, and that this figure would grow ~3% annually. My aggressive assumptions are that the maximum willingness to spend will be ~2% of GDP and GDP will grow ~5% per year in the long run;[7] my conservative assumptions are that

7 In particular, they don't attempt to tease apart the amount of computation required to replicate short-horizon and longer-horizon animal behaviors.

the maximum willingness to spend will be ~0.5% of GDP and GDP will grow ~2% per year in the long run.

Below are the willingness-to-spend curves for the aggressive and conservative assumptions, respectively:



Translating into views on TAI timelines

As discussed in Part 1, this model is not directly estimating the probability of transformative AI, but rather the probability that the amount of computation that would be required to train a transformative model using contemporary ML methods would be attainable for some AI project, assuming that AI research progresses along a “business-as-usual” trajectory. I would expect this model to overestimate the probability of TAI in the very short-term, but underestimate it in the long-term. In this section, I discuss:

- What I consider to be a reasonable median estimate for when TAI may arrive (more).
- What I consider to be a reasonable range of probabilities for TAI by 2036 (more).
- What I consider to be a reasonable range of probabilities for TAI by 2100 (more).

What is a reasonable range of *medians* for when TAI may be developed?

My best guess assumptions result in a median year of ~2052 (32 years from the time of writing) for when the amount of computation required to train a transformative model may become affordable. My aggressive assumptions place this median at ~2036, and my conservative assumptions place it at ~2100.

As I described in Part 1, 2052 feels far enough in the future that I feel more or less comfortable assuming that the probability that *the computation required to train a transformative model is affordable by then* is roughly similar to the probability that *TAI is developed by then.* To be maximally precise, we would need to adjust this probability *downward* by some amount to account for the possibility that other key resources such as datasets and environments are not available by the time the computation is available, and adjust *upward* by some amount to account for the possibility that TAI is developed by 2052 through other means besides training a single transformative ML model. To avoid false precision and choose a round number, **I am tentatively adopting ~2050 as my median forecast for TAI.**

On the other hand, 2035 feels like it is soon enough that the probability TAI is developed through a qualitatively different paradigm is substantially lower, meaning that the downward adjustment seems like it should carry more weight than the upward adjustment. Somewhat arbitrarily, I will shift this estimate upward a few years and say that **my "most aggressive plausible median" is ~2040.**

Finally, 2087 feels far enough in the future that the possibility of other paths to transformative AI seem to clearly outweigh the possibility of other bottlenecks. I am inclined to shift the estimate forward in time significantly to **~2090 for my "most conservative plausible median."**

What probabilities would be reasonable to assign to TAI by 2036?

In a 2016 blog post, Open Philanthropy CEO Holden Karnofsky stated that he would personally estimate >10% chance of TAI within 20 years (i.e. by 2036); at the time, he considered this to be moderately robust, meaning he would be moderately surprised if further research brought his central estimate below 10%. Vetting this claim more deeply was one of the motivations for doing this research.

Having done this exercise, I would say that this forecast looks reasonable, although the robustness may have been slightly overstated. My best guess inputs to the model suggest that there is a ~23% probability that the computation required to train a transformative model would be available by 2036, and I would adjust that downward to ~18% for my all-things-considered probability of TAI by 2036; this is in line with Holden's estimate. However, conservative assumptions which still seem defensible might assign a probability under 5%. **I think a very broad range, from ~2% to ~45%, could potentially be defensible.**

According to my **best guess assumptions**, AI labs could spend up to ~3e29 FLOP training a model in 2036 if they believed it would be highly lucrative (this is about 1 million times more computation than it took to train AlphaStar). Performing ~3e29 FLOP would cost ~$3T in 2020, but my best guess compute price forecasts imply that it would instead cost $30B by 2036. In other words, I assume that there will be a ~100x improvement in compute prices from now to then, while spending will be ~30,000 times larger than spending on AlphaStar, multiplying to a ~3 million-fold scaleup in training computation.

By 2036, my best guess algorithmic progress assumptions imply that the various hypotheses would each require ~30 to ~300 times less computation than they do in 2020. According to the Lifetime Anchor hypothesis, there is a ~75% probability that ~3e29 FLOP is enough to train a transformative model in 2020; according to the Evolution Anchor hypothesis the probability is virtually 0, with other hypotheses in between. Given my best guess weighting across hypotheses, there is a ~19% probability that the computation required to train a transformative model will be less than ~3e29 in 2036. Because 2036 is so near and there may be data bottlenecks and/or engineering bottlenecks, I am inclined to shade this estimate downard to **~12%-17%**.

However, according to my **conservative assumptions**, there is only a ~5.3% probability that the amount of compute required to train a transformative model is affordable by 2036. This is because compared to my best guess assumptions, there is about 30 times less computation available (~1e28), each hypothesis makes somewhat less algorithmic progress, and I place less weight on hypotheses which assume the computation requirements are lower (with only ~15% of the weight going to the Lifetime Anchor, Short Horizon Neural Network, and Genome Anchor hypotheses collectively, compared to ~35% under my best guess assumptions). Adjusting downward for the probability that other resources are also in place could bring this down to a **2%-4% probability of TAI by 2036.**

According to my **aggressive assumptions**, there is a ~50% probability that the amount of compute required to train a transformative model is affordable by 2036. This is because compared to my best guess assumptions, there is about 10 times more computation available (~3e30), each hypothesis makes somewhat more algorithmic progress, and I place more weight on hypotheses which assume the computation requirements are lower (with ~70% of the weight going to the Lifetime Anchor, Short Horizon Neural Network, and Genome Anchor hypotheses collectively). Adjusting downward for the probability that other resources are also in place could result in **~35%-45% probability by 2036**.

What probabilities would be reasonable to assign to TAI by 2100?

My best guess assumptions result in a ~78% probability that the amount of computation required to train a transformative model is affordable by the end of this century. My aggressive assumptions result in a probability of ~96%, and my conservative assumptions result in a probability of ~50%.

Based on the reasoning given here in Part 1, I am inclined to bump up at least the conservative number to some extent:

- I expect that FLOP per dollar and, and algorithmic progress will continue at a slower pace rather than flattening out entirely by the end of the century.
- 80 years feels like plenty of time for TAI to be developed through a qualitatively new paradigm and/or through gradual distributed automation. In particular, various technical advisors and Open Philanthropy staff place substantial credence on technologies such as whole brain emulation and atomically precise manufacturing being developed sometime this century which could accelerate progress toward transformative AI if developed.

However, I don't think it would be appropriate to have as much confidence as the aggressive estimate would indicate, given meta-uncertainty about whether this model is reasonable or whether we are missing an important consideration, and the possibility of very long-term delays due to extreme algorithmic bottlenecks, training environment bottlenecks, or business-as-usual progress being disrupted due to investment drying up[8] or exogenous events such as global catastrophic risks or disruptions to governance.

Ultimately, I could see myself arriving at a view that assigns anywhere from **~60% to ~90% probability** that TAI is developed this century; this view is even more tentative and subject to revision than my view about median TAI timelines. My best guess right now is about 80%.

Responses to common questions and objections

In this section, I'll address the following high-level questions and objections to this framework:

- Does this framework assume that transformative AI is "hardware bottlenecked" (more)?
- What if training data or training environments will not be available by the time the requisite computation is available, or the computation to run training environments exceeds the computation to train the model (more)?
- What if training a transformative model would take too much wall-clock time even if the computation is available (more)?
- Is there a risk of selection bias if we're extrapolating training data requirements from existing ML problems, because we are more likely to have solved tasks with ML when the scaling behavior is favorable (more)?
- The extrapolation of willingness to spend assumes continuous growth, but what if there is another "AI winter" causing spending to stop rising or to drop (more)?
- Wouldn't the biological anchors approach generate the same estimates for training FLOP requirements at all points in the past, even though algorithmic progress means that each year should be different (more)?
- How could we test the predictions of the biological anchors framework and potentially falsify it (more)?
- How applicable is this model if transformative AI arrives by a very different path, e.g. via distributed automation or through a new "paradigm" (more)?

8 From this table on Wikipedia; "house mouse" entry.

- Instead of using biological anchors, why not simply directly gauge how capable or impressive AI systems are and extrapolate this progress forward (more)?

Does this framework assume TAI is “hardware bottlenecked”?

A salient position in discussions of AI forecasting is that AI progress is “hardware-bottlenecked”, in the sense that the main (or only) factor limiting further progress in AI capabilities is progress in the availability of hardware, with progress in architectures and algorithms playing a very limited role. This position is controversial, and many experts hold the opposite position, that progress is bottlenecked by key algorithmic insights, and dramatically increasing hardware without acquiring those insights would have very limited value.

I don’t currently believe that AI progress is entirely “bottlenecked” by either hardware or algorithms in a strong sense; I expect that these resources essentially complement and substitute for one another in a relatively smooth way. I expect that either a massive windfall of computation or a massive windfall of research talent would accelerate AI capabilities and bring TAI closer.

My framework reflects this view -- it does not make a structural assumption that increases in the total amount of computation available (due to falling hardware prices and/oir rising spending) are the primary way that the probability of transformative AI increases over the years. I model both increasing availability of computation and algorithmic progress decreasing the amount of computation required to train a transformative model. (I have chosen particular parameter values which imply that hardware prices are falling at roughly the same rate as algorithmic progress is improving, but these views are unstable and I would not be surprised if I revised my estimate of the rate of algorithmic progress upward.)

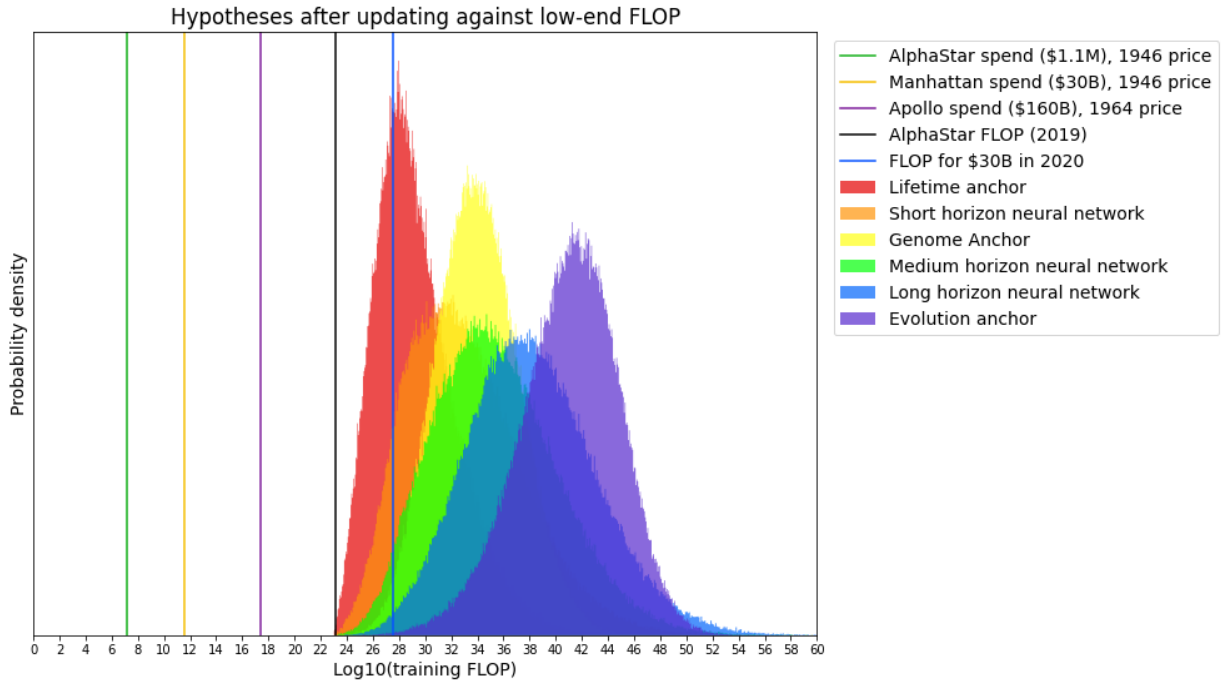
However, I am committed to a weaker claim, which is that it is likely that researchers could figure out how to combine 2020 architectures and algorithms with an amount of computation between human lifetime computation and evolution computation (as well as arbitrary amounts of training data) to train a transformative model within a few years of trying. My model wouldn’t be a useful tool for thinking about TAI timelines if you assigned a small probability to this, because the “larger than Evolution Anchor” hypothesis always estimates that the amount of computation required to train a transformative model is out of reach by construction. For example, if you assigned only a 10% probability that 2020 training computation requirements are at or below the Evolution Anchor level, your beliefs about TAI timelines would likely be dominated by your forecasts for when researchers figure out how to bring down 2020 training FLOP requirements down to evolutionary levels or lower. My model incorporates this only very coarsely (by allowing the user to decrease over time the probability that no biological anchor hypothesis is correct), and someone with this view should likely take a different approach which emphasizes estimating the rate of major algorithmic breakthroughs in greater detail.

With that said, I don't currently see a good argument in favor of a probability of 50% or more for the view that the amount of computation required to train a transformative model is outside this range, because:

1. Sufficiently large neural nets are capable of expressing arbitrary programs with the right setting of the parameters. As long as you believe that a transformative task (e.g. the virtual professional task) would be computable, there exists *some* neural network that can solve it. This is almost certainly the case, and not generally disputed.
2. Therefore, the two main questions are:
    a. *How large would the neural network need to be?* The analysis done in Joe Carlsmith's report about the human brain is very suggestive that there exists some neural network which runs on about 1e15 FLOP / subj sec that is capable of solving the "virtual professional" transformative task (see the next section for more detail). The architecture of the human brain may be more efficient than the architectures that researchers can currently design, but I don't think we have much reason to be confident that it is many orders of magnitude more efficient.
    b. *Can 2020 optimization algorithms find the right setting of parameters?* Simple variants of stochastic gradient descent (SGD) have so far been able to find parameter settings which are capable of expressing some important human and animal behaviors (e.g. object recognition, language processing, and elements of strategy and competition). As I argue below, the capabilities of models today don't seem highly inconsistent with the predictions that this framework would make about when we should be able to match the capabilities of various animals.

This suggests that we should assign substantial probability to one of the Neural Network hypotheses being correct. When including the full range of effective horizon lengths which seem plausible (from ~1 subjective second to ~1 billion subjective seconds or ~32 subjective years), the Neural Network hypotheses collectively span most of the range from the Lifetime Anchor

hypothesis to the Evolution Anchor hypothesis:



Most of this range is outside what would be affordable in 2020 even with a megaproject spending tens of billions of dollars on hardware, and I don't think it makes sense to be confident in specific intuitions about what would happen if training run sizes were suddenly scaled up to be this much larger anything we have observed so far when they contradict the high-level argument given above.

My personal estimate is 90% that the Evolution Anchor hypothesis or some cheaper hypothesis correctly describes 2020 training FLOP requirements, and 80% that one of the hypotheses that predict lower computation than Evolution Anchor is correct. See the discussion of probabilities in Part 3 for more detail.

What if training data or environments will be a bottleneck?

If it is much slower to generate enough relevant data or training environments than it is to scale up the computation available for training, the outputs of this model will be a less useful guide for thinking about TAI timelines.

I have not investigated this question thoroughly, but I currently don't expect it to be a major issue for the framework. It is plausible that training data is a limiting factor in the near-term (giving some reason to adjust the near-term forecasts output by this model downward), but I don't expect it to be a major obstacle for a long period of time, such that it would make sense to shift the median estimate to the right by several years or more to account for this.

In this section I discuss:

- The possibility that data collection or generation will be a bottleneck (more).
- The possibility that the computational expense of *running* training environments will be a bottleneck (more).

Collecting or generating the right data and environments

Training a transformative model will likely require a rich and broad distribution of data and experience, and hypotheses which predict that total computation to train a transformative model will be very high (the Long Horizon Neural Network hypothesis and the Evolution Anchor hypothesis) also predict that a very large amount of training data would be required.

Particularly for these hypotheses, generating a sufficiently large, broad, and complex distribution of training data or RL environments may prove to be a bottleneck. One piece of suggestive evidence for this is that solving robotics tasks entirely in simulation tends to be much easier and cheaper than training them to transfer well to the real world, because the latter often requires randomizing across a large number of different simulations with slightly different physical parameters (see for example OpenAI's Rubik's Cube paper).

However, it seems fairly plausible to me that the relevant data and environments could be gathered even for high-compute and high-data hypotheses:

- Existing passive data may go a long way. For generative modeling, trillions of words of text and petabytes of images and videos exist on the internet. For reinforcement learning, researchers may be able to figure out how to make relatively simple modifications or extensions of existing video games, board games, and procedurally generated RL environments to be more difficult or have a much longer effective horizon. Some of these environments may already support a very high "ceiling" of performance (for example, it is likely that neither humans nor AIs are anywhere near perfect StarCraft play) or a rich diversity of ways for the agents to modify their own environments to make things more difficult for other agents (for example, Minecraft). These passive data stores are likely to grow over time -- even though more data will be required under the higher-computation hypotheses, there will also be more time to accumulate it because it will take longer to get to the point of affording the requisite computation.
- Researchers may be able to construct simple reinforcement learning environments in which the central challenge is some abstract mathematical problem that is very easy to generate but very difficult to solve (e.g. "Factor this product of two large primes"), or a simple language- or code-based task that is relatively easy to generate and evaluate but difficult to execute (e.g. "Build a web application with the following properties").
- Researchers may be able to automate some parts of environment generation, for example by using a generative adversarial network setup in which a separate model is trained to generate an RL environment that the current population of agents is likely to perform poorly on. This would be especially useful if the problem of generating difficult RL environments is substantially less complex (requiring a much smaller model) than the problem of performing well in difficult RL environments.
- If the high-computation hypotheses turn out to be correct, then by the time the computation for a transformative model is affordable, it is likely to be at a very high price

point (e.g. several hundred billion dollars) after decades of scaleup (see above for my willingness to spend forecasts). Once an AI project is spending this much money on hardware, it also makes financial sense to pay a large amount of money for data generation, for example by paying for a lot of human labor to provide demonstrations of tasks or feedback to models. Conversely, if low-computation hypotheses (e.g. the Lifetime Anchor hypothesis) turn out to be correct, it is more likely that total spending on computation will be lower (e.g. $1B), but the challenge of data collection will also be much lower.

On balance, I expect that training data and environments are likely to be available roughly around the time that the requisite computation is affordable, but I do consider it plausible that data/environments will cause several years of delay, particularly if high-compute and high-data hypotheses are true. I think this consideration is mostly a reason to believe that the right tail of the timelines distribution is longer than the output of this model would imply and that the probability of AI this century may be somewhat lower than the model implies. I am not inclined to revise my median upward much on the basis of this consideration, partially because it is offsetting some of the ways the model is conservative (e.g., not taking into account different paths to TAI and not fully accounting for the possibility of "breakthrough progress" increasing the probability of low-compute hypotheses over time).

Having computation to run training environments

An implicit assumption made by all the biological anchor hypotheses is that the overwhelming majority of the computational cost of training will come from running the model that is being trained, rather than from running its training environment.

This is clearly the case for a transformative model which only operates on text, code, images, audio, and video since in that case the "environment" (the strings of tokens or pixels being processed) requires a negligible amount of computation and memory compared to what is required for a large model. Additionally, as I mentioned above, it seems possible that some highly abstract mathematical environments which are very cheap to run could nonetheless be very rich and support extremely intelligent agents. I think this is likely to be sufficient for training a transformative model, although I am not confident.

If reinforcement learning in a rich simulated world (e.g. complex physics or other creatures) is required to train a transformative model, it is less clear whether model computation will dominate the computation of the environment. Nonetheless, I still believe this is likely. My understanding is that the computation used to run video game-playing agents is currently in the same ballpark as the computation used to run the game engine. Given these models are far from perfect play, there is likely still substantial room to improve on those same environments with a larger model. It doesn't seem likely that the computational cost of environments will need to grow faster than the computational cost of agents going forward.[9] (If several intelligent

9 My median estimate for the scaling exponent is around ~0.8. The constant factor implied by the reference model that I use is around 10^2.6 = ~400, and 400 * (1e12)^0.8 = ~1.6e12.

agents must interact with one another in the environment, it seems likely that all agents can be copies of the same model.[10])

In the main report, I assume that the computation required to train a transformative model under this path can be well-approximated by $F \times H \times K \times P^{\alpha}$, where F is the model's FLOP / subj sec, H is the model's horizon length in subjective seconds, P is the parameter count of the model, and α and K describe scaling behavior. I do not add an additional term for the computational cost of running the environment.

What about the wall-clock time to execute a training process?

If a training run takes too long to run in elapsed real time or "wall-clock time" (as opposed to subjective time), it will be impractical to execute even if the hardware is affordable to purchase and the right training environments are available and affordable to run. For the output of this model to be a useful guide to TAI timelines, training a transformative model must not be bottlenecked for a long time by the need to bring wall-clock training time down to a reasonable level.

I have not investigated this question deeply, but my current belief is that this likely won't turn out to be a major obstacle. Most large-scale training runs of the last several years already involved processing large amounts of data in parallel to limit the wall-clock time taken to train models. The number of data points processed in parallel per gradient update is known as the **batch size**, and the number of successive gradient updates performed in series is called the **step count.** My understanding is that the evidence so far suggests it would be possible to scale up batch sizes tremendously as data requirements scale up, scaling up step count only slightly or not at all:

- McCandlish et al 2018, "An Empirical Model of Large-Batch Training", experimentally derives the **critical batch size** (the batch size that best optimizes for total data requirements and total computation simultaneously) for a number of different ML problems. The paper finds that when training at the critical batch sizes, all of the models tended to train in a step count of ~100-10,000, with no clear dependence on model size.[11]
- Kaplan et al 2020, a paper by the same research group studying the scaling behavior of language models, implies that when training language models at the critical batch size, the number of serial steps of gradient descent scales extremely slowly with total

10 From this table on Wikipedia; "honey bee" entry.

11 If a bee's neurons fire at a similar rate to a human's neurons, that would imply that a bee brain runs on ~1e9 FLOP/s and a model with ~1e9 parameters running on ~1e10 FLOP / subj sec could replicate most of its behaviors. Such a model would train on about 400 * (1e9)^0.8 = 6e9 data points according to my best estimate of scaling behavior. A 1 subjective second horizon task would therefore take (6e9 data points) * (1 subj sec / data point) * (1e10 FLOP / subj sec) = ~6e19 FLOP, and a task with an effective horizon length of 1 subjective hour would take 3.5 OOM more.

computation, an amount consistent with zero scaling given measurement error.[12] This paper is discussed in more detail here in Part 2.

In other words, these results suggest that most of the increased data requirements for larger models are due to needing a larger batch size to get a good estimate of the gradient, rather than needing to take a longer optimization path from the starting point to find a good model. I think it is likely that this would be true for some transformative ML problem as well, such that the total amount of wall-clock time taken to train a transformative model would likely be manageable.

I do expect that there will be non-trivial algorithmic and engineering work required to scale up data parallelism, but I don't currently see a strong reason to believe it will be a major obstacle in a way that is not already implicitly captured by the fact that the model assumes the AI field will need to ramp up spending over a period of decades before reaching peak spending (see the discussion of willingness-to-spend above).

In the main report, I assume that training a large transformative model will not be prohibitive in wall-clock time, focusing primarily on estimating the cost of computation required.

Is selection bias an issue for extrapolating data requirements?

We are intrinsically limited in what we can learn from observed scaling behavior, *because we can only observe scaling behavior for learning problems which researchers have chosen to pursue* (and publish about). If getting decent performance on a certain type of learning problem would require an unaffordable amount of data or computation, it's likely that researchers will not attempt to train a model on that problem at all, or that it would not make it into a paper if they tried and got lackluster results. For example, no one attempted to train a neural network to play Go until 2014, even though Go was a relatively prestigious research area and people had been programming Go AIs using classical techniques for decades.

That is, it could be the case that sample complexity will scale differently for a transformative learning problem than for today's learning problems -- in addition to the fact that a transformative model would need more training data because it has more parameters than current models, the *function relating parameter count to dataset size* might be completely different for transformative learning problems.

However, given the facts on the ground, I think selection bias is probably only a mild-to-moderate issue for using existing ML problems to extrapolate the training data requirements for a transformative model. This is because:

12 My best guess (based on publicly-known information) is that the final training run for AlphaStar was ~1e23 FLOP, and cost roughly ~$1M (see this appendix for details). As far as I know the AlphaStar training run is roughly the most expensive training run in a published paper as of July 2020, and similar to the amount of computation used in the final training run of GPT-3 (although I expect that training AlphaStar was overall more expensive, counting the computation used to run the environment and do earlier training runs). I expect that unpublished models would have been trained using somewhat more computation.

- **We don't *need* to invoke selection bias to explain a lack of progress:** One reason we could hypothesize that scaling behavior is unfavorable for a transformative task is because we have not yet solved any transformative task, and have not gotten a large amount of economic value out of ML systems in general. However, even if the scaling behavior for a transformative problem were exactly the same as the scaling behavior for current learning problems, we likely wouldn't have been close to being able to train a transformative model under the Genome Anchor hypothesis or any of the Neural Network hypotheses. The vast bulk of the probability mass for all of these hypotheses before updating against low levels of FLOP was placed on levels of FLOP that would be unaffordable in 2020.[13] Model size requirements alone are likely sufficient to explain why we haven't already seen transformative AI or extremely economically valuable AI systems. It is possible that *the scaling itself* will be worse for future learning problems, but positing that doesn't seem necessary to explain the current state of the AI field.[14]
- **We could have seen problem categories that scale more poorly than RL by now:** We already see a significant difference in scaling behavior across different types of problems, between RL models and supervised learning models. We can train huge generative models (such as GPT-3 with ~200 billion parameters) for about the same expense as training substantially smaller RL models (such as AlphaStar with ~50 *million* parameters): an RL model of size P is roughly as affordable to train as a generative model of size $1000\,P$. If there were a large attractive category of learning problem for which sample complexity scales moderately worse than RL, it's plausible that researchers would have discovered this by now -- perhaps they would currently be training models in the ~10-100 *thousand* parameter range on this new category of problem. My understanding is that we haven't confidently identified such a problem category yet.[15] I consider this to be a modest amount of evidence against the possibility that learning problems we will pursue in the future will turn out to scale more poorly than current learning problems.
- **The horizon length formulation allows for huge variation in *total timesteps*:** Finally, the effective horizon length is a free parameter which is exogenous to model size. The fact that different ML problems have different horizon lengths implicitly allows for

13 From this table on Wikipedia; "fruit fly" entry. Fruit flies have ~1e7 synapses, which would mean that we should expect an ML model with ~1e7 parameters and ~1e8 FLOP / subj sec to have the capacity to emulate fruit fly behavior. Emulating 1-second-horizon fruit fly behaviors such as motor control should therefore take 400 * (1e7)^0.8 * 1e8 = ~1e16 FLOP. If this estimate should actually be ~10 OOM larger, that would mean that we would need ~1e26 training FLOP to replicate even short-horizon fruit fly behaviors. This seems implausible to me -- for example, the motor control involved in fruit fly wrestling does not seem *obviously* much more sophisticated as a behavior than the manual manipulation involved in solving a Rubik's cube (which was trained on ~1e21 FLOP).

14 According to this list on Wikipedia, small monkeys such as the capuchin monkey and large birds such as macaws have brains ~1.5 OOM smaller than human brains. Because my median estimate for the computation required to train a transformative model is ~1e37, that implies my median estimate for the computation required to train a model around as capable as a baboon or chimp should be ~3e35 or so. If that value is off by ~10 OOM, that implies that a model capable of solving long-horizon tasks as well as monkeys or large birds should take ~3e25 FLOP to train, and our current largest models should be capable of solving short- and medium-horizon tasks better than monkeys and large birds.

15 See "When will computer hardware match the human brain?", a 1997 article by Hans Moravec, as an example.

substantial variation in the scaling behavior of *total timesteps* $T(H,q,P)$ across different ML problems even if the scaling behavior in terms of "number of horizons" remains constant. We can observe this already to some extent in this chart: AlphaStar has only ~3x as many parameters as AlphaGoZero, but trained on almost 9000 times as many timesteps.[16] If we expect that transformative learning problems likely have much longer horizons than current learning problems, then the number of *timesteps* it takes to train a transformative model could vastly exceed the number of timesteps it would take to simply train a larger version of AlphaStar or OpenAI Five. The medium horizon and long horizon Neural Network hypotheses make this assumption.

- **Tractability is difficult to predict, and ML seems to have had a good hit rate on "prestige tasks":** While I do expect that researchers generally don't attempt to solve tasks that are *clearly* out of reach, my understanding is that it is generally very hard to show that new categories of tasks (e.g. a new video game or board game) will *clearly* be easy, because there is simply a lot of noise and risk inherent to trying something novel. Given that, I take it to be a modest amount of positive evidence against the selection bias hypothesis that sample complexity has not exploded when ML was applied to a handful of high profile tasks (Go, DOTA, StarCraft, language modeling) that seem to have been selected largely for prestige rather than tractability.

Overall, I don't feel that selection bias strongly undermines the evidence from existing models; I am inclined to take this information as a starting point, include wide error bars, and only adjust upward modestly to generate a best guess estimate for transformative model data requirements.

Would we have made the same estimates in the past despite algorithmic progress?

At a high level, the methodology of forecasting from biological anchors is fairly "timeless." Someone could have advanced the general hypothesis "Maybe if we could somehow create an AI system as computationally powerful as the human brain that would constitute transformative AI" at any point in the past, because the basic hypothesis doesn't obviously depend on any specific properties of the current state-of-the-art in AI. (Indeed, some futurists in the 1990s like Hans Moravec and Ray Kurzweill did advance biological anchor hypotheses[17] that helped to inspire this work and more immediate precursors to this work done by Open Philanthropy technical advisors.)

The "timelessness" of using biological anchors to estimate resource requirements for TAI is a potential problem for the strategy. Advancing algorithmic progress means that the resource requirements for producing a transformative model (whatever they may be at a given moment)

16 Uncertainty *across* different hypotheses is driven by more fundamental uncertainties such as which biological anchors make the most sense and what horizon length is likely to be required; uncertainty *within* a single hypothesis is driven by the types of considerations described in this section.
17 The Lifetime Anchor hypothesis was the only one which was materially affected by that update.

should be decreasing over time -- this means that model size requirements can only be close to the brain at *one particular point in time*, which might not be the *current* point in time.

As an example, say we believe that transformative model size requirements would have fallen by 5 OOM from 1970 to 2020 based on our understanding of the rate of algorithmic progress. If we were analyzing the resources required to train a transformative model in 1970, it seems likely that we would have assumed that a transformative model would be roughly the size of the human brain because the brain is an *a priori* plausible anchor, and there wouldn't have been much reason to deviate from it. But when we attempt to do the analysis in 2020, the brain *still* seems like an *a priori* plausible anchor and it feels unmotivated to assume that a model 5 OOM smaller than the brain would be sufficient. Either our naive 1970 estimate would have been wrong or our naive 2020 estimate would be wrong, with the discrepancy being more severe the more algorithmic progress we believe has occurred in the last fifty years.

I have tried to mitigate this issue in two main ways. **First, I attempt to incorporate empirical evidence from other technological domains** where the comparison between human-created artifacts and natural artifacts is clearer to help set where I think transformative model size requirements fall relative to the biological anchor. Based on this, I estimate that a transformative model would run on about ~1 OOM more FLOP / subj sec than the human brain in 2020. Doing this exercise at various points in the past would have resulted in a higher central estimate for where model size requirements lie relative to the human brain anchor, because the collection of reference technologies such as solar panels and artificial organs would have been less efficient relative to their natural counterparts.

Second, I attempt to make the *spread* of my distribution consistent with my beliefs about the pace of algorithmic progress. It's clear that I should incorporate *some* degree of uncertainty about where transformative model size requirements lie relative to the brain FLOP/s anchor, because if I were doing this analysis in 2010 or 2030 rather than 2020, I would still likely have guessed that a model ~1 OOM larger than the brain would be sufficient. In other words, even after trying to incorporate empirical evidence from other technologies, my intuitions don't feel very sensitive to a difference of ~10 years in either direction -- I have no substantive reason to expect that 2020 (rather than 2023 or 2029 or 2012 or 2017) is the *precise* year when a model *precisely* 10 times larger than the human brain (rather than 7 times or 13 times larger) could be trained to solve a transformative task.

If I separately expect that transformative model size requirements would fall by ~X OOM over a typical period of 10 years, then it wouldn't make sense for my subjective distribution over where transformative model size requirements lie relative to the brain FLOP/s anchor to have a standard deviation of X OOM or smaller. My beliefs about algorithmic progress would require that I shift the 2020 model size requirements distribution to the left X OOM to generate the 2030 model size requirements distribution -- but because my judgments are so coarse-grained, I would also want the 10-years-shifted distribution to look "more or less the same" as the original distribution. These two goals can only be approximately reconciled if the *width* of the 2020

distribution is substantially larger than the *shift* from 2020 to 2030 -- if that is the case, the 2020 and 2030 distributions will assign fairly similar probabilities to most values.

I have briefly investigated algorithmic progress trends above; my tentative best guess is that the amount of total FLOP required to *train* (as opposed to run) a transformative model is halving once every ~2-3 years. This corresponds to a decrease of 1-1.5 OOM every 10 years. Because I want the training FLOP requirements distribution in year Y to look very similar to the distribution in year Y+10 or year Y-10, I want the standard deviation to be much larger than 1.5 OOM. Somewhat arbitrarily, I chose to aim for a standard deviation of ~5 OOM in the training FLOP requirements distribution for each individual biological anchor hypothesis.[18]

I did not research what fraction of reductions in training computation requirements can be attributed to reductions in model size requirements in particular (as opposed to e.g. how data requirements scale with model size). I ended up going with a standard deviation of ~2 OOM for the distribution of where model size requirements fall relative to the brain, which results in roughly the spread I want to see when combined with other distributions into an estimate of training computation requirements.

What if there is an "AI winter" and spending stops growing?

In the willingness to spend section, I estimated that:

- Over the next ~5 years AI companies would likely be able to scale up to spending about $1B on computation for the largest training runs, in the process building up a lot of infrastructure and know-how around training very large models.
- It is likely that in the medium term, spending on computation for a large training run could scale up further to about $100B if there were a strong possibility of training a transformative model as a result.
- In the long run, computation to train a transformative model could stabilize at about 0.5% of gross world product.

However, it's possible that the next ~1-2 OOM of scaleup will prove disappointing, causing interest and investment to dry up. This would likely substantially reduce the size of the largest-scale training runs that can be attempted and may precipitate a shift away from the "deep learning paradigm." In this scenario, my near- and perhaps medium-term spending forecast would be an overestimate, meaning that the near- and medium-term probability of TAI is likely to be an overestimate.

However, even if we enter such an "AI winter" in the near term, I think the long-run average willingness-to-spend forecast (~1% of GDP) is not likely to be a huge overestimate, because I expect companies to anticipate TAI long enough ahead of time to ramp up spending to that level.

18 In fact, it's possible that the scaling behavior might be better for a transformative learning problem than for current problems, while still implying that training a transformative model requires levels of computation and data that we can't access right now.

If investment dries up in the near-term, **I would take it as fairly strong evidence that the 2020 technical difficulty of training a transformative model turned out to be far out of reach** -- that it wouldn't have been possible even with a $100B investment in computation. This is because I expect that if spending $100B would have led to a transformative model, spending $30-300M will look qualitatively exciting and potentially add some economic value, spurring greater investment, which will have greater returns and spur even more investment, and so on.

In other words, I generally expect that **we will see "signs of life" along the way to being able to train a transformative model.** Even if this current spending boom ends in an AI winter, I expect that once algorithmic progress starts to bring down the technical difficulty of solving a transformative task, we will probably start to see some qualitatively impressive progress on interesting "toy problems" that were previously difficult to solve, and investment will soon pick up again in anticipation of further progress. Once it becomes possible to solve a transformative task at ~1% of GDP, I think some AI project will probably be willing and able to effectively spend that much, having already ramped up its capacity.

Others may disagree with me about the degree to which there will be signs of life along the way to TAI that spurs high levels of investment. If you disagree with my position, you should likely place more weight on relatively low willingness-to-spend levels, and consider them to be more uncertain. This means you should assign a higher probability that some group will "stumble upon" the solution to a transformative task at a relatively modest level of spending (because they or their potential investors did not properly anticipate that it would actually lead to transformative AI).

How can we test this model's predictions and falsify it?

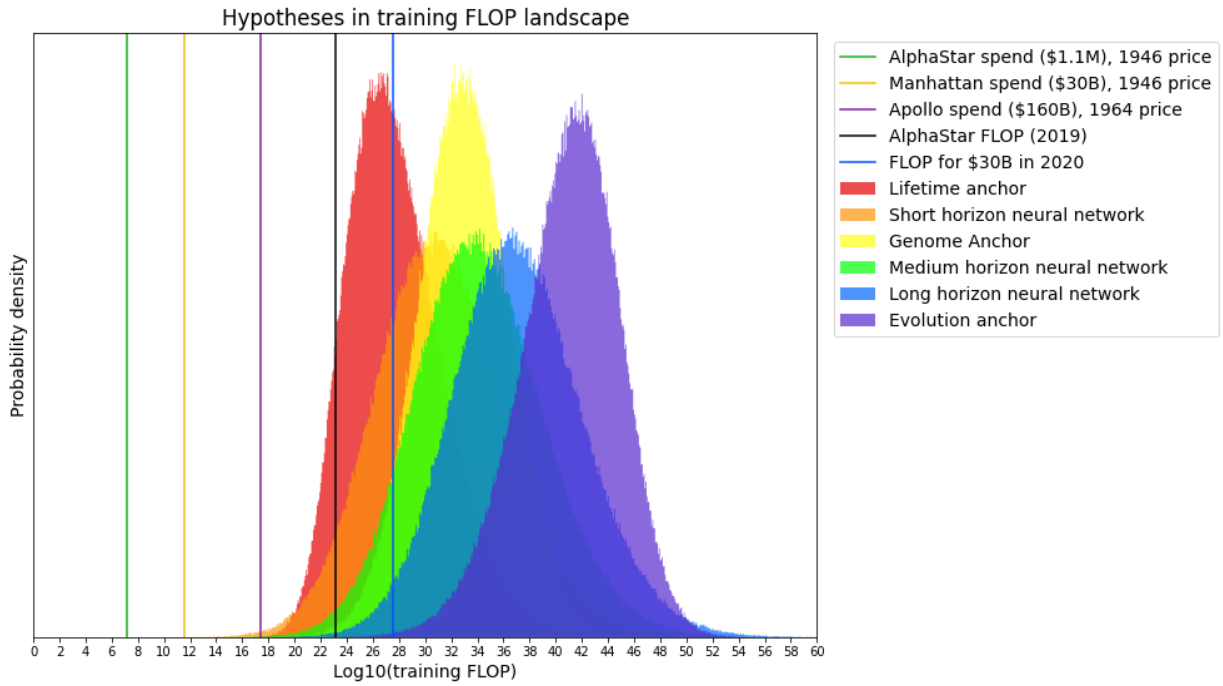
I have focused in this report on articulating the biological anchors framework at a conceptual level rather than digging deeply into what it would have predicted about the past or present; we hope to do more of this in future work. With that said, I will touch on two high-level ways to back-test this framework:

- What would the framework have predicted about whether it would be possible to train a transformative model in the past (more)?
- What would it predict about the amount of computation required to match the abilities of smaller animals (more)?

It would not have confidently predicted TAI in the past

As I said above, the high level analysis strategy of anchoring to biological quantities (particularly the Lifetime Anchor and the Evolution Anchor) would have been available to hypothetical AI forecasters at most points in the past, and some forecasters used a version of it.

I believe that a realistically-accessible version of this exercise, if done in the 1950s, would have correctly predicted that transformative AI is most likely several decades away. As this chart shows, only the Lifetime Anchor hypothesis assigns any substantial probability to levels of computation that were available decades ago:



With that said, it would have been reasonable to have more ignorance and to assign higher probabilities to low-end levels of computation in the past. In particular, the Lifetime Anchor hypothesis that I generated is not centered around my median estimate for human lifetime computation (~1e24 FLOP), but is shifted up a few OOMs; this is due to intuitions and evidence about ML that would not have been accessible in the past (more).

However, shifting the median for the Lifetime Anchor hypothesis distribution back to ~1e24 FLOP would not have resulted in the model estimating a high probability of TAI by 2020, assuming that the probability mass was still distributed between the Lifetime Anchor and Evolution Anchor hypotheses. Additionally, as I mentioned above, attempting to account for how technological artifacts compared to natural artifacts in the 1950s may have pushed estimates out somewhat further.

Many researchers felt optimistic about developing AGI relatively quickly in the early years of the AI field, and adopting this framework would have likely pushed against this in a way that turned out to be more correct than those researchers' intuitions at the time; I take this to be a moderate amount of evidence in favor of this approach, particularly compared to approaches that have a heavier element of subjectively gauging AI progress.

It is not obviously inconsistent with the capabilities of animals

We should also be able to use the biological anchors framework to estimate the amount of computation that would be required to train ML models to replicate the behaviors of various *smaller* animals (such as insects and mice). We can then use these estimates to check which animal(s) current ML systems should be on par with, as a way of sanity-checking whether the

training computation requirements distribution for a transformative model is in roughly the right place.

I'll only consider the three "mainline" Neural Network hypotheses, because the other hypotheses make less fine-grained predictions[19] and I haven't assigned them much weight in total. How much computation should it take to replicate the behaviors of a mouse according to the mainline Neural Network hypotheses (assuming we had the necessary environments and evaluation functions)?

- A mouse's brain has around ~1e12 synapses;[20] if mouse neurons fire at a similar rate to human neurons, that would suggest that a mouse brain runs on ~1e12 FLOP/s and a model that runs on ~1e13 FLOP / subj sec with ~1e12 parameters would have the capacity to replicate most mouse behaviors given proper training.
- According to the way I expect data points to scale relative to model size, we would need about ~1e12 data points to train such a model.[21] This would imply that to replicate mouse behaviors that occur over a period of ~1 second (e.g. object recognition), we would need about (~1e13 FLOP / subj sec) * (~1e12 effective horizons) * (~1 subj sec / effective horizon) = ~1e25 FLOP. On the other hand, to replicate mouse behaviors for which ground-level feedback is only given over the scale of an hour (e.g. learning to solve a complex novel obstacle course), we would need ~3e28 FLOP.

On the other hand, replicating the 1-second-horizon behaviors of a bee (~1e9 synapses)[22] should take around ~1e20 FLOP, and replicating 1-hour-horizon bee behaviors should take around ~3e23 FLOP.[23]

19 It is possible that meta-learning is a candidate for a category of task which scales more poorly, but I would guess that this is captured to a large extent by the effective horizon length parameter (see the discussion of likely effective horizon length in Part 3).

20 Here, a "timestep" for AlphaGoZero (AGZ) is defined as one move that the model *actually makes*, and does *not* count the hypothetical moves considered by the Monte Carlo Tree Search (MCTS). The vast majority of the computation of AGZ went into MCTS rather than making actual moves and updating the model.

21 With that said, environment simulation tends to run on CPUs rather than GPUs, so it may benefit less from improvements in FLOP/$, particularly if they are based on increasing parallelization of GPUs.

22 Note that the AlphaStar model was trained with the help of nine "exploiter agents", each of which specialized in defeating the main agent when it was playing as a particular race in StarCraft. These exploiter agents increased the total cost of training the AlphaStar model by a factor of 4, because they were about as computationally expensive as the main model but are not actually good StarCraft players because they were not being optimized for winning the game in general. My best guess is that the "main agent plus specialized exploiter agents" training structure made sense mainly because StarCraft is a relatively narrow game with particular rules, and that it would make sense to train many instances of the same agent, and pool updates across them, when training to elicit generally-intelligent behavior. Therefore, I did not include the cost of running the exploiter agents in my estimate of the amount of computation to train the StarCraft agent for the purposes of the extrapolation; see this cell in the spreadsheet for more explanation.

23 Figures 9 (pg 20), 11 (pg 21), 12, 13 (pg 22), and 14 (pg 23). They show how serial optimization steps (x-axis) and total examples processed (y-axis) relate for different target performance levels (different colors) for several different ML problems. The point at the *elbow* of each curve gives the optimization steps and the examples processed at the critical batch size. The number of optimization steps moves to the right for different target performance levels on a given ML problem, but there is no clear pattern in the number of optimization steps used *across* problems. The total examples processed increases for harder

As of July 2020, the largest-scale ML training runs are likely around 1e23 to 1e24 FLOP,[24] and the most computationally powerful published model is GPT-3 at around ~2e11 FLOP / subj sec. This suggests that today's most expensive models ought to be somewhat more capable than bees on both short-horizon and medium-horizon problems, not yet quite as capable as mice (even over very short horizons), and *much* less capable than bigger-brained animals such as ravens and crows, cats and dogs, or primates, on any timescale.

It is very difficult and subjective to evaluate whether this is actually the case, both because it's hard and research-intensive to pin down these animals' behavioral repertoires and because most ML models are optimized to solve problems (e.g. Go, StarCraft, and language modeling) which are completely different from the problems these animals were selected to solve in their ancestral environments (e.g. selecting mates or hunting prey).

With that said, it does not strike me as *obviously unreasonable* to describe today's ML models as "somewhat more capable than honeybees on most problems and somewhat less capable than mice on most problems." As an example, learning a new task or pattern from a small number of examples ("few-shot learning") is a behavior that animals are likely selected for to some extent, and a research problem in ML that gets a substantial amount of attention. Both mice and bees seem to be capable of few-shot learning to some extent -- but I have only seen bees learn very simple tasks (e.g. pulling strings or rolling balls) so far, whereas mice seem to be able to reliably learn a wide variety of complex behaviors. Intuitively, the breadth and complexity of the within-context few-shot learning displayed by GPT-3 and T5 seem plausibly more impressive than bee learning and less impressive than mouse learning. Additionally, my understanding is that few ML researchers claimed to have made major strides on transfer learning or few-shot learning before models got to be quite large, which feels like additional suggestive evidence for the basic framework.

We hope to do additional research comparing the capabilities of animals with the capabilities of various machine learning systems in the future, and I could easily see that research shifting my beliefs by a few orders of magnitude -- for example, I could easily come to believe that current systems are not quite as capable as bees once the breadth and robustness of the bee behavioral repertoire is fully taken into account. Nonetheless, this cursory examination makes me believe that it's fairly unlikely that my current estimates are off by *several* orders of magnitude. If the amount of computation required to train a transformative model were (say) ~10 OOM larger than my estimates, that would imply that current ML models should be *nowhere near* the abilities of even small insects such as fruit flies (whose brains are 100 times smaller than bee brains).[25] On the other hand, if the amount of computation required to train a

problems which call for larger models, but the number of optimization steps doesn't seem to increase (see for example the MNIST vs CIFAR curves, or the Atari vs DOTA curves).

24 "This leads us to conclude that the optimal number of steps will only grow very slowly with compute...In fact the measured exponent is sufficiently small that our results may even be consistent with an exponent of zero." Kaplan et al 2020 pg. 16.

25 Here I am implicitly making the assumption that AI labs are currently scaling up spending roughly as fast as they can manage logistically speaking, and if a transformative model is trained between 2020 and

transformative model were ~10 OOM *smaller* than my estimate, our models should be as capable as primates or large birds (and transformative AI may well have been affordable for several years).[26]

What if TAI is developed through a different path?

The analysis in the main document focused on estimating the training computation requirements of a "transformative model": a single computer program found through local program search techniques broadly similar to the ones popular in machine learning research and practice as of 2020.

However, I believe that this analysis can provide a useful median estimate even if TAI is produced through a very different path: essentially, by the time it is affordable to develop TAI through a *particular* highlighted route, it is plausible that somebody develops it through that route *or any cheaper route.* I consider the example of a distributed economic transition facilitated by a broad range of different technologies below, but the same reasoning applies to the possibility that a unified transformative program may be developed using a qualitatively different "AI paradigm" that can't be usefully considered a descendant of modern machine learning.

"TAI" may be a collection of several technologies

I think it is fairly likely that "transformative AI" will be developed without any individual transformative program solving any single, unified "transformative task."

Consider that it doesn't seem correct to characterize the Industrial Revolution as being precipitated entirely (or even mostly) by one "transformative invention" or "transformative discovery." No single device like the steam engine -- something specific enough and discrete enough that one company or individual could receive a patent for it -- seems to have driven the bulk of the economic growth and attendant societal changes that happened from ~1650 to ~1850. Rather, the Industrial Revolution is probably best characterized as driven by the gradual build-up of a large number of different inventions, bodies of scientific knowledge, bodies of practical tacit knowledge and habits, extracted raw resources, markets for various goods and services, ways of organizing society and workplaces, public goods and feats of infrastructure, etc.

2025, it will have to be at a relatively low level of spending, because it will happen in the middle of a period when labs are still figuring out how to scale effectively. Note that earlier, I made the assumption that there is a >50% chance we would have already trained a transformative model if the computation would have cost somewhere between $1M and $100M to train in 2020, and a substantial probability that we would have already trained a transformative model if the computation would have cost somewhere between $100M and $10B. That was reflecting my belief that if it were so cheap to train a transformative model, we would have probably seen signs of it years ahead of time and therefore managed to build up the relevant infrastructure to train models at scale by now. However, for forecasts of the *future* (as opposed to updates from the past) I am conditioning on the fact that the training runs executed so far are relatively small and labs must figure out how to scale up over the next few years. See this appendix for more detail.

26 Figure 3, pg 8.

This large interconnected system of people, objects, and practices (which spanned a large fraction of the whole economy and touched many other parts of society) was retroactively given the label of "industry." Many individual elements of "industry" are complementary with, substituting for, or otherwise responsive to other elements, such that most reasonable methods of assigning counterfactual impact to particular discrete inventions, discoveries, or key decisions would likely result in assigning relatively limited impact to any one factor (in particular it is likely that no one factor analogous to our notion of a "task" would be credited with having "transformative" impact). Similarly, "agriculture" seems to be best understood as the label for a conceptually related cluster of practices and technologies (e.g. plowing, irrigation, domestication and breeding, etc) which were discovered and re-discovered in different versions concurrently across different isolated peoples, and which were improved slowly over multiple generations to reach their modern form.

In contrast with the Industrial and Agricultural Revolutions, the nuclear bomb was a single, discrete invention built over the course of one focused project that (although it did not have a *transformative* impact the way we use that term) had a relatively immediate major impact on the nature of international conflict, without the need for a long integration period or the development of extensive auxiliary or complementary technologies and services before it had that impact.[27] This is more analogous to developing transformative AI via solving a single "transformative task." Many discussions of highly advanced future AI systems seem to implicitly assume that it will look like this -- that at some point in the future, a particular company or government project will develop an "artificial general intelligence" (AGI) or "superintelligence", which is modeled as a single unified computer program that can do everything that an intelligent human can do at least as well as (or possibly much better than) a human can.

I am broadly skeptical of the "unified AGI" assumption -- I think it is unusual for a single discrete piece of technology to have such a clear and large impact as quickly as the nuclear bomb did, and by default I expect that new civilizational heights of technological sophistication will be acquired via **the gradual accretion of individually low- or moderate-impact innovations distributed throughout a broad economy** and spaced out in time. I have not seen strong signs that TAI will be different, and have seen some weak signs that it will be similar -- e.g., recent substantial investment in self-driving cars and AI for medical diagnostics may well end up looking like the beginnings of a broad economic transition enabled in significant part by AI and other software.

In theory, this implies this analysis underestimates P(TAI)

However, despite the fact that I don't expect transformative AI to be developed in one fell swoop by one group, I still think that the strategy of estimating when it would become affordable to produce a transformative program has value. Because this model estimates when one *particular path* toward transformative AI (let's call it the "big model path") out of many will be attainable, that means **if this analysis is correct** (i.e., if I am correct to assume the big model path is possible at all due to the theoretical feasibility of local search, and if we correctly

27 Figure 1, pg 5.

estimated the probability that it would be attainable in year Y for all Y), **then the probability estimates generated should be underestimates**.

Suppose that we estimate that a large company could afford to train a large ML model to do a transformative task in year Y with probability *p.* If we think this is accurate, we should believe the probability that TAI is developed by year Y is greater than *p*: either TAI will be developed in year Y through the big model path with probability almost[28] *p*, or it will already have been developed before year Y through some *cheaper* path (which may be much less tractable to analyze directly) with some non-zero probability *q*, so the probability of TAI by year Y should be somewhere between almost *max(p*, *q)* and almost *p* + *q.*

The further away Y is, the more time there is for gradual innovation to culminate in transformative change and/or for algorithmic breakthroughs to unlock a substantially less expensive path to solving a transformative task, and the more this is an underestimate of the true probability.

In practice, my interpretation depends on the year in question

However, once sources of distortion (many of which tend to push our estimates upward) are properly taken into account, **I think it is fairly unclear whether these estimates should actually be considered underestimates.**

Firstly, all of the potential biological anchor hypotheses we highlighted have substantial question marks associated with them, such as “Will robustness and reliability work out well enough that the system is usable?”, “Will we be able to generate enough data of the right kinds?”, and “Will the training data extrapolation based on small models continue to hold?” Trying to make any particular hypothesis concrete enough to analyze in detail tends to surface a proliferation of potential pitfalls, for which I haven’t heard immediately compelling solutions from technical advisors or other researchers.

Despite this, I have assigned non-trivial or moderate probabilities to most of the biological anchor hypotheses I have considered. This is because these anchors feel plausible to me on priors and have not been obviously contradicted by existing (relatively thin) evidence from existing models, and thus I am assigning substantial weight to the possibility that various potential pitfalls I see now turn out not to be issues or turn out to be easy to resolve or work around. For each biological anchor hypothesis, I am acting on the assumption that there is a relatively broad space of “unknown unknown” paths to solving a transformative task within that range of technical difficulty, not just the particular concrete path I have written down for illustration in association with each hypothesis (which is often fairly conjunctive).

I expect reasonable people to have a wide range of instincts about how to handle the possibility of such “unknown unknowns.” In particular many reasonable people would argue that when estimating the probability of a qualitatively “extraordinary” or “surprising” event, it would be

28 SAT solving, board games (chess and Go), physics simulations, factoring large numbers, mixed integer programs, and machine learning. Results summarized on pg 2.

inadvisable to fudge estimates upward for the possibility of unknown unknown factors, because rough estimates will already tend to be overestimates and we should expect that unknown unknown factors are more likely to end up lowering rather than raising the probability. Despite choosing in this case to account for unknown unknowns that would increase my probability estimate, I have a substantial amount of sympathy for this view and it has generally been my experience that deeper analysis tends to result in downward adjustments to many quantitative estimates (e.g. for the cost-effectiveness of a potential grant).

Secondly, I've focused almost entirely on estimating the amount of the total FLOP that would be sufficient to train the model according to each hypothesis, under the assumption that this is a good proxy for the all-in technical difficulty of training a machine learning model. However, this will not always be the case -- in some scenarios, it will be the case that the raw FLOP will be available well before some other key resource becomes available. Throughout the document, I have tried to flag when I noticed that another resource (e.g. human labor to generate data or memory bandwidth to keep wall clock training time manageable) could plausibly be a bottleneck, but because that was not my focus in this research project it seems fairly likely that trying to do a more careful accounting of all-in technical difficulty will cause me to increase my median estimate for when training a transformative model will become feasible.

Finally, the possibility of exogenous events such as global catastrophic risks, severe recessions, or changes in regulatory regime that could derail business-as-usual AI research is not factored into this model and should slightly decrease the probability estimates generated.

Despite these considerations, some of our technical advisors are still relatively confident these probability estimates are low-end estimates. This is partly because they would assign a higher probability to some of the low-end biological anchor hypotheses than I do, partly because they are overall more confident in the argument given above that these numbers ought to be considered underestimates, and partly because they believe that more carefully accounting for the correlations between different variables (which I have not attempted to model in detail) should increase our estimate that TAI is developed soon relative to the estimate generated by this model.

For now, I feel that the most reasonable way to interpret the probability estimates generated by the biological anchors framework is as a **rough central estimate for when TAI will be developed rather than as particularly conservative or particularly aggressive.** In making this judgment, I am admittedly mentally running together a large cloud of heterogeneous considerations which in a maximally-principled and transparent analysis should be handled separately. I feel very unstable on this judgment, and I expect many other Open Philanthropy staff members and/or readers to disagree.

In the coming years, we hope to explicitly quantify more and more pieces of the overall analysis (for example, by reworking some parts of the structure of this model to better highlight key correlations, or combining this analysis with analysis of the probability distribution of other global catastrophic risks to estimate the annual discount from exogenous events).

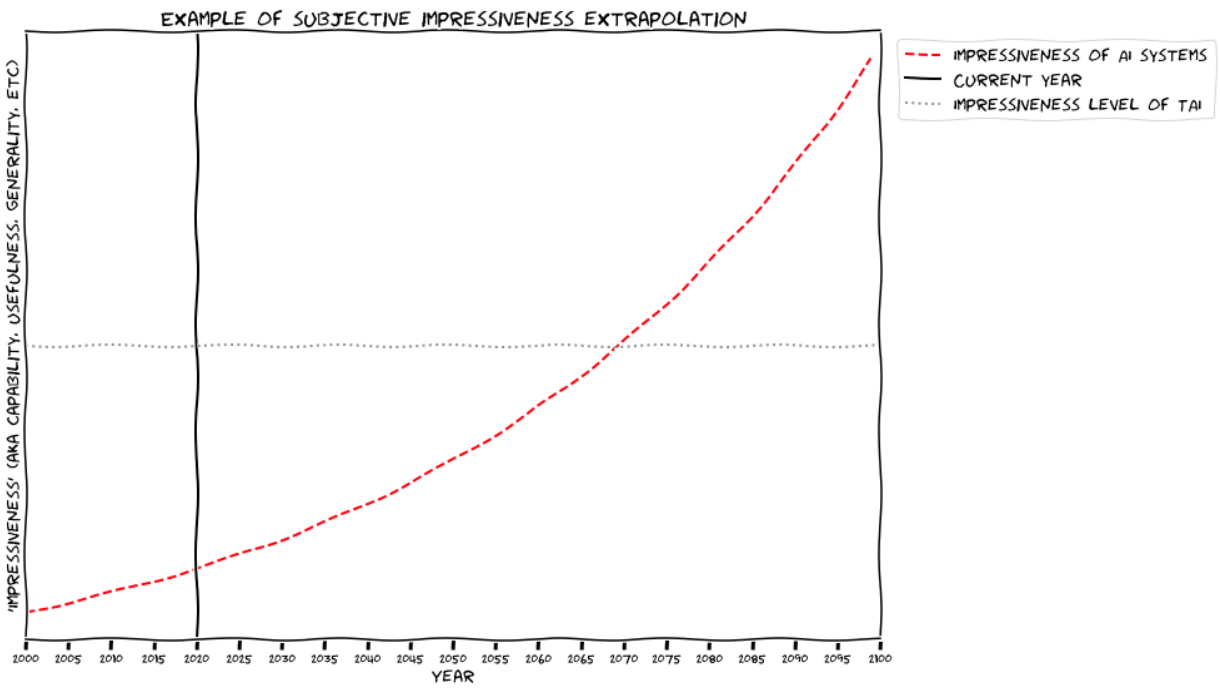
More immediately, we are planning to write more about the all-things-considered subjective TAI timelines views of various key staff members, after receiving feedback on this document from a broader range of machine learning experts and completing currently ongoing investigations into two other promising high-level perspectives on the question of when transformative AI may be developed (uninformative priors and extrapolations from long-run gross world product data). Ultimately, our all-things-considered subjective probabilities will be determined by some combination of the output of this framework, the output of other explicit frameworks such as the two we are currently investigating, and various clusters of considerations which are harder to fit into any explicit framework (such as deference to the views of various experts whose intuitions may not be easily expressible in terms of explicit probabilities).

Why not directly extrapolate AI capabilities instead?

A very different high-level approach to estimating TAI timelines (which in our experience most people initially gravitate toward) involves more holistically assessing progress in AI systems' capabilities, rather than leaning heavily on biological anchors. Essentially, this approach is to:

1. Judge how "impressive", "capable", "general", or "useful" state-of-the-art (SOTA) AI systems currently are (for example by synthesizing information from various key benchmarks and AI challenges such as performance on board games, Winograd schemas, adversarial examples, etc).
2. Assess how quickly the impressiveness of AI systems has been improving recently.
3. Extrapolate how many years of progress at the current pace would be required to reach the level of impressiveness required for TAI.

I'll call this approach the **subjective impressiveness extrapolation** approach, which stands in contrast with the biological anchors framework used in this report. Here is a visualization of a hypothetical TAI timelines forecast using a subjective impressiveness extrapolation approach, where the x-axis is the year (from 2000 to 2100), and the red line represents the holistic "impressiveness" of AI systems in that year (which reaches the level of TAI around ~2065-2070):



To most people, the subjective impressiveness extrapolation may initially seem more natural, more direct, and less complicated than the biological anchors framework. However, I think it actually has substantial disadvantages and is ultimately less reliable as a forecasting methodology.

The most important disadvantage of the subjective impressiveness extrapolation is that it is **extremely unclear what exactly the y-axis refers to**, and different people will have different intuitions about it. People's intuitions about the y-axis will a) dramatically impact resulting timelines estimates, and b) be hard to fully articulate and reconcile with one another. In our experience, two people (including two ML experts) can come to wildly different conclusions about how impressive current SOTA is and how rapidly it is improving because they focus on very different indicators of "impressiveness."

We have heard ML experts with relatively short timelines argue that AI systems today can essentially see as well as humans, understand written information, and beat humans at almost all strategy games, and the set of things they can do is expanding rapidly, leading them to expect that transformative AI would be attainable in the next decade or two by training larger models on a broader distribution of ML problems that are more targeted at generating economic value. Conversely, we have heard ML experts with relatively long timelines argue that ML systems require much more data to learn than humans do, are unable to transfer what they learn in one context to a slightly different context, and don't seem capable of much structured logical and causal reasoning; this leads them to believe we would need to make multiple major breakthroughs to develop TAI. At least one Open Philanthropy technical advisor has advanced each of these perspectives.

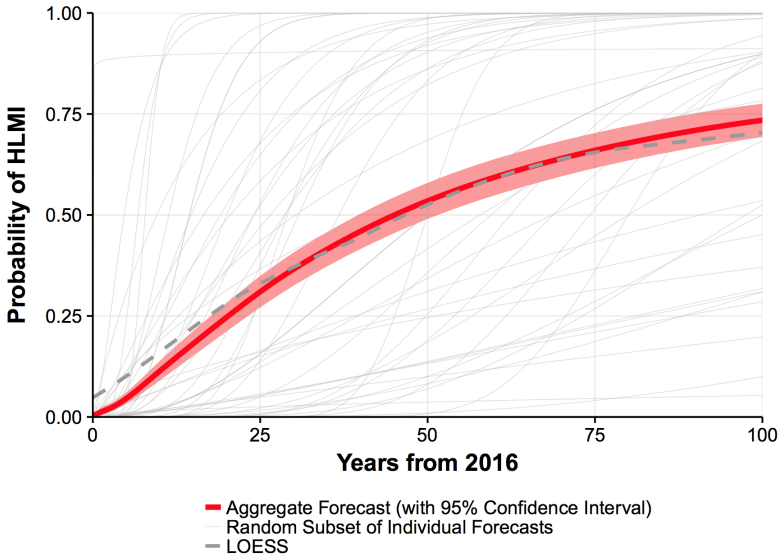
What would a representative expert survey look like?

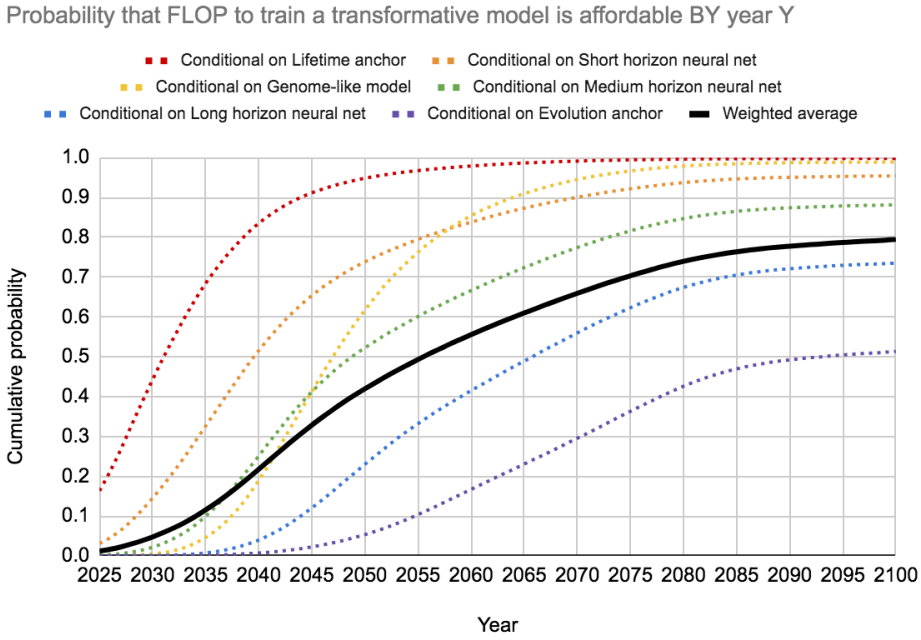
Because different experts' positions can vary so much, we don't think it makes sense for generalists to lean too heavily on their own subjective impressiveness extrapolation or that of any one ML researcher in forming their views on timelines. In our view, the best way to forecast TAI using subjective impressiveness extrapolations would be to survey a large representative group of ML experts. This has the drawback that the information we can extract is relatively crude and high-level -- it doesn't help articulate the underlying reasons different experts have different views, or pinpoint places where they would make differing predictions about the results of an experiment. Nonetheless, it would be very reasonable for a generalist to lean on the outcome of a well-designed survey of experts in coming to their beliefs about TAI timelines.

Grace et al 2017 is the best expert survey on timelines that I am aware of; the authors reached out to all researchers who published at 2015 NIPS or ICML conferences, ultimately surveying 352 of them.[29] The main forecasting target researchers were asked about in this survey was **"high-level machine intelligence"**, defined as the time when "when unaided machines can accomplish every task better and more cheaply than human workers."[30] This is a stronger condition than transformative AI, which can be achieved by machines which merely complement human workers (see the discussion of transformative tasks in Part 1).

Nonetheless, the aggregated estimate across survey respondents of when HLMI would be achieved[31] (shown at the top below) is remarkably similar to my best-guess forecast for when the computation required to train a transformative model will be affordable (bottom):

29 On the other hand, arguably tasks that are extremely impractical currently have the most low-hanging fruit for improving them.
30 This median shifted to the right due to the truncation; before truncation, it was at roughly ~3e27.
31 See this appendix for reasons why the long-run growth rate of the largest national economy may be higher than the historical rate of growth of the United States.





Additionally, researchers' responses in this survey indicated that small changes in framing lead to very different answers for questions that are substantively equivalent:

- Some researchers were asked to forecast "HLMI" as defined above, while a randomly-selected subset was instead asked to forecast "full automation of labor", the time when "all occupations are fully automatable." Despite the fact that achieving HLMI seems like it should quickly lead to full automation of labor, the median estimate for full automation of labor was ~2138 while the median estimate for HLMI was ~2061, almost 80 years earlier.
- Random subsets of respondents were asked to forecast when individual milestones (e.g. laundry folding, human-level StarCraft, or human-level math research) would be

achieved. The median year by which respondents expected machines to be able to automate AI research was ~2104, while the median estimate for HLMI was ~2061 -- another clear inconsistency because “AI research” is a task done by human workers.

The Grace et al results suggest to me that a) ML experts in aggregate seem to have views about the future of AI that are broadly consistent with estimates output by the biological anchors framework, and b) many researchers’ forecasts do not seem highly robust or self-consistent. At some point in the future, we hope to do a more detailed updated version of the Grace et al survey which focuses on transformative AI rather than HLMI and potentially takes measures to reconcile inconsistencies.

Could the y-axis be operationalized more objectively?

Another potential way to improve the standard subjective impressiveness extrapolation framework is to attempt to better define the y-axis, and collect objective information about how it has varied over time. For example, we could define “impressiveness” as the amount of revenue generated by machine learning models in inflation-adjusted dollars, or as the fraction of jobs done by humans as of 2000 which have now been automated by machines,[32] or as the economic productivity of the tech sector.

“Impressiveness” could also be operationalized in terms of benchmarks such as SuperGLUE or the unrestricted adversarial examples challenge, rather than purely in economic terms. However, it seems important that a wide range of researchers who currently disagree on timelines at the object level can agree that the benchmark(s) chosen are reasonable, so that we avoid a situation where researcher A and researcher B both expect to see a certain score on a particular benchmark achieved by a certain year but disagree on whether that constitutes much evidence that TAI may be developed soon. I am very interested in this type of exercise, particularly if the operationalization focuses on revenue or productivity, and think it could serve as a very valuable supplement and sanity check to the outputs of this framework. We may eventually do some research along these lines.

Open questions for further investigation

Most of these are investigations that a generalist researcher could do, but some of them involve machine learning experiments.

- Can we further refine and formalize the definition of “effective horizon length”, and get more mathematical clarity on whether the functional form proposed in the report (linear scaling) is appropriate?
- Will it take longer to train the same model to solve a task that seems intuitively “longer horizon” compared to a task that seems intuitively “shorter horizon”? For example, will it take more training time for a language model to be able to do within-context few-shot

32 This might occur in worlds in which the amount of computation required to train a transformative model is very high (as the Evolution Anchor or Long Horizon Neural Network hypotheses would suggest), and it is very difficult to create large amounts of economic value with levels of computation much smaller than that.

learning for complex skills vs simpler skills? I am hoping to register my predictions about the likely horizon lengths of various RL tasks at some point in the future.

- What would an experiment analogous to Kaplan et al 2020 for reinforcement learning models imply about scaling behavior for RL? Would a systematic sweep over the discount rate γ be consistent with total training timesteps scaling with $1/(1-\gamma)$?
- What will various machine learning experts think of the biological anchors framework? Will the framework help articulate some of the underlying causes of disagreement about TAI timelines? What are the most important disagreements about timelines which are difficult or impossible to capture in the language of this framework?
- How do the capabilities of machine learning models compare to those of various animals, particularly on tasks that seem closely analogous to the types of tasks that animals solve? Examples of such tasks could include:
    - Object recognition
    - Motion detection
    - Motor control
    - Sound recognition and localization
    - Making simple predictions grounded in intuitive physics (e.g. anticipating that a cup will fall if the person holding it lets go)
    - In larger animals, learning novel behaviors with a small number of trials (e.g. solving mazes or obstacle courses)

  More carefully documenting animals' abilities and their limits in these domains, and getting a better sense of how they likely compare to various ML models, would help generate a narrower and more accurate distribution over how animal brain architectures compare to ML architectures which run on a similar number of FLOP / subj sec (where animal brain FLOP/s is defined using the "evolutionary hypothetical" definition from Part 1).
- Similarly, what would a more careful and comprehensive version of Paul's investigation into how human-made technologies compare to natural counterparts imply?
- Could we use recent results in large-scale image recognition[33] to extrapolate how large a model would need to be to recognize approximately any object as well as a typical human seeing the object for an instant (e.g. 200 ms), without being fooled by adversarial examples or noise? Would this forecast be consistent with a "truly human-level object recognition model" being only ~1 OOM larger than the human visual cortex (or whatever "portions" of the visual cortex -- or potentially other parts of the brain -- we believe are involved in the task of "object recognition")? Thanks to our technical advisor Jacob Steinhardt for this idea.
- What should we expect about the future of compute prices, particularly about the shape of diminishing returns and potential hard limits to the number of FLOP per dollar that could be achieved, in light of the possibility of shifting to more exotic computing paradigms such as optical computing, three-dimensional circuits, reversible computing, and so on?

33 Particularly Mahajan et al 2018 (which pre-trained a large model to predict hashtags on ~3B images from Twitter and then fine-tuned on ImageNet), Xie et al 2019a (which pre-trained on 250 million images), and Xie et al 2019b (which used a model large enough that they could introduce a small amount of adversarial noise while increasing accuracy on the ImageNet dataset).

- What should we expect about the increase in willingness to spend on computation going forward? Would it be possible for training runs in the latter half of the century to cost much more than I currently assume, due to factors such as increasing GWP, increasing inequality and concentration of wealth, the possibility that pre-transformative AI systems may have generated huge amounts of revenue that could be reinvested into training larger systems, the ability to spread spending out over many years, the possibility of national or international conglomerates, and so on?
- What would a more comprehensive investigation imply about the rate of algorithmic progress for various tasks, particularly a) tasks that researchers have been working on for several decades such as chess, and b) tasks within machine learning such as NLP?
- How much does hardware progress for a particular application accelerate as demand for that application spikes? For example, how much progress was made on hardware accelerators for Bitcoin mining as a function of demand for that application?
- Relatedly, how much does increased R&D investment accelerate algorithmic progress? Are there historical cases we could examine of a particular subfield in CS attracting a lot of talent and money as machine learning is doing today? How much algorithmic progress did such sub-fields make, as measured by how quickly the technical difficulty of particular tasks dropped?
- How much does access to cheaper hardware for experimentation speed up algorithmic progress? Can we generate a "production function" for algorithmic progress in terms of hardware and labor inputs?
- What would a more careful estimate of the amount of computation done over the course of evolution look like? Could we incorporate an estimate of the amount of computation done within the DNA itself, rather than just within animals' brains? Are there plausible high-end anchors that are more expensive than evolution, but less expensive than fully brute-force search (which would cost on the order of ~2^(10^15) FLOP)?
- Can we get a more solid sense of what datasets and/or training environments would be sufficient to execute each of the hypotheses in this document? How likely is it that we will be able to develop these datasets by the time the computation is affordable? How much human labor would go into generating them, and could the human labor costs dominate the hardware costs?
- What can we learn about the amount of computation required to run environments? Is it plausible that the computation required to run the training environment would dominate the amount of computation required to run the model(s) being trained?
- What groups of tasks might be collectively transformative? What horizon length would AI systems need to do these tasks? How expensive would it be, and how long would it take, to gather the data needed to train AI systems to do these tasks?
- Will current techniques, or something broadly similar, allow us to learn medium and long horizon tasks, or will we run into fundamental obstacles attempting to scale horizon length so far?
- How long in wall-clock time would it take to train transformative systems with medium and long horizon lengths? Would that be prohibitive?

Acknowledgements

Firstly, I owe a huge thanks to Paul Christiano, who profoundly shaped this report by conducting substantial prior conceptual and empirical research that I drew on, having extensive discussions with me that heavily informed the structure of the model and my intuitions about key quantitative inputs, and providing many rounds of detailed feedback on this report as well as several early stage research documents leading up to this draft. In addition to the tangible research value he added in his capacity as a technical advisor, Paul provided steadfast emotional support in his capacity as my partner, consistently believing in me when I didn't always believe in myself.

I am grateful to the following people whose research informed the construction of this framework or its key quantitative inputs:

- Technical advisor Dario Amodei for his early thinking on the high-level biological anchors framework, brain computation, and sample complexity extrapolation.
- My colleague Joe Carlsmith, for his philosophically careful and empirically thorough investigation into brain computation (which is at the foundation of every biological anchor hypothesis) and his feedback on early stage research documents.
- Kathleen Finlinson for her thorough investigation into the history of hardware prices.
- Tom Davidson for assisting with research into RL scaling behavior and providing thought partnership from an early stage.

Major thanks to Collin Burns, Jacob Hilton, Jared Kaplan, Richard Ngo, and Rohin Shah for extensive feedback and thought partnership that inspired substantial revisions.

Thanks also to Alexander Berger, Bastian Stern, Buck Shlegeris, Carl Shulman, Catherine Olsson, Daniel Kokotaljo, Danny Hernandez, David Roodman, Jacob Steinhardt, Jacob Trefethen, John Schulman, Luke Muehlhauser, Mike Levine, Nick Beckstead, Otis Reid, Peter Favoloro, Sam McCandlish, Zach Robinson, and others for feedback and discussion.

Thanks to Andreas Stuhlmueller, Jungwon Byun, and Owain Evans of Ought for improving my code and integrating Elicit.

Finally, I am deeply grateful to my manager Holden Karnofsky for providing excellent thought partnership, feedback, mentorship, project management and emotional support throughout the course of this research project, and for helping me to become a much more effective researcher and communicator over the last four years.