Link to part 2. Link to part 3. Link to folder. Link to summary.

Takeoff speeds report

Acknowledgements

- To Carl Shulman especially for introducing many of the core ideas.
- Jaime Sevilla, Eduardo Infante Roldán, and others at Epoch for coding up the Full Takeoff Model, creating the playground and many helpful suggestions.
- Dan Kokotajlo, Holden Karnofsky, Paul Christiano, Ajeya Cotra, Luke Muehlhauser, Lukas Finnveden, Leopold Aschenbrenner, Joe Carlsmith, Peter Favaloro, Tamay Besiroglu, Owen Cotton-Barratt, Ege Erdil, Eliezer Yudkowsky, Nate Soares, Basil Halpern, Marius Hobbhahn, Ali Merali, Alyssa Vance, Eric Neyman, Lukas Gloor, Anton Korinek, Jakub Growiec, Eli Lifland, Charlotte Siegman, Ben Levinstein, David Schneider-Joseph and Jacob Trefethen for helpful feedback.

Contents

You should read the short and long summaries before reading sections in this document.

2. What is takeoff speed? Why does it matter?

I recommend skipping this section unless you're interested to hear about ways of quantifying takeoff speeds that I didn't end up focusing on.

This section discusses what I mean by AI takeoff speed, why it matters, and how we might quantify takeoff speed. Ultimately, I think there are multiple reasons to care about takeoff speed and multiple reasonable ways to quantify it. I introduce some ways of quantifying takeoff speed that are both decision-relevant and that I can forecast using the framework of this report.

Takeoff speed of AI capabilities need not be the same as takeoff speed of AI impacts

What question is takeoff speeds trying to answer? One vague version of the question is: *How long will it take to go from significantly capable AI to billions of AGIs[1]?*

If it takes a month then takeoff is fast; if it takes 50 years then takeoff is slow. There are many ways you could define “significantly capable AI”; I’ll discuss some possibilities below.

If we want to avoid reference to the (arbitrary) startpoint and endpoint we can phrase the question as: *How quickly will AI capabilities improve as AI systems collectively approach and surpass human intelligence?*

This refers to AI capabilities directly rather than AI’s effect on the world so I call it “capabilities takeoff speed”.

Another version of the takeoff speeds question is: *How long will it take to go from AI having a significant impact on the world to AI having a truly transformative impact on the world?*

Again, if this happens in a month then takeoff is fast; if it takes many decades then takeoff is slow.

We could define “significant impact” in different ways depending on whether we’re interested in economic impact, unemployment, military power, technological progress, or something else. And takeoff speed will plausibly differ between these different domains based on how much AI’s effects are bottlenecked by regulations or scarce physical equipment (more). Again, in the case of economic impact, I’d define “significant” as “adding ~$5trillion/year to global GDP”.

By “transformative impact” I mean causing a transition comparable to (or more significant than) the agricultural or industrial revolution, e.g. by significantly increasing (~10X) the rate of economic growth (more).

This definition refers not just to AI capabilities but to its actual impact on the world; let’s call this impact takeoff speed.

If capabilities takeoff is fast, then impact takeoff is more likely to be fast. But they can come apart in either direction.

- Suppose there’s fast capability takeoff, but regulations, safety concerns and other bottlenecks prevent advanced AI being used in the economy. If these bottlenecks are

[1] By AGI I mean an AI system, or a collection of AI systems, that can do virtually all cognitive tasks that a human can do. By “cognitive task” I mean “any part of the workflow that could in principle be done remotely or is done by the human brain”. So it includes ~all knowledge work but also many parts of jobs where you have to be physically present. E.g. for a plumber it would include “processing the visual and audio inputs relating to the problem, choosing a plan to solve it, and deciding what specific actions to take second by second”.

removed or overwhelmed gradually over many decades, you could have a slow impact takeoff.
- Suppose delays to broad adoption get shorter for more advanced AIs. Then impact takeoff will be faster than capability takeoff. An extreme case of this is where regulations stop AI have ~any economic impact until misaligned AGI forcibly and suddenly disempowers humanity.

I think the question of impact takeoff is probably more important than capabilities takeoff, but harder to forecast as there are more factors that influence it. The framework here is most reliable for predicting capabilities takeoff. I will report takeoff metrics that relate to both impact and capabilities, but the impact metrics don't account for various possible delays. Before I discuss these metrics precisely, I discuss why takeoff speeds matter.

Some reasons to care about takeoff speeds

Takeoff speed is correlated with a few factors that are strategically important. For example:

- **Warning shots.** How long before the point of no return do we get clear evidence of AI risk[2]? What about clear evidence that AI will be transformative? All things equal, faster takeoff means we'll have less time to respond and there will be fewer people paying attention.
- **Time for high-impact alignment work.** Before we develop AIs that pose existential risk, we might develop AIs that are similar but do not pose existential risk. Alignment work on these systems will probably be particularly impactful for reducing risk, because they'll be more similar in structure and behavior to the systems that later pose existential risk. All things equal, faster takeoff means less time to do this high-impact alignment work.
- **Concentration of power**. Faster takeoff means less time for AI progress to spread around the world. All things equal, this will lead to fewer relevant AI actors and a higher chance of an actor getting a decisive strategic advantage[3].
- **Changes in the strategic landscape.** Slower takeoff means more time for the world to be transformed by pre-AGI systems, e.g. dramatically changing the geopolitical landscape of the defense-offense balance in cyber. This might favour increasing longtermist influence in generic ways over making specific plans.
- **AI timelines.** Holding fixed AGI training requirements, faster takeoff means *more* time to AGI because earlier systems do less to accelerate AI development. This has a number of strategic implications. Indeed, this framework allows us to quantify this tradeoff.

These factors affect how large AI risk is overall and what actions we should take to reduce it.

[2] In particular, risks that could be existential as AI capabilities improve. E.g. clear evidence of misaligned power-seeking.

[3] A decisive strategic advantage is "a level of technological and other advantages sufficient to enable it to achieve complete world domination", Bostrom (2014), p. 78.

In addition, having a view about takeoff speeds might allow us to make predictions about precursors to AGI. If we're right, we gain credibility and confidence in our views; if we're wrong we can update our views.

Quantifying takeoff speed

Summary: I use GDP metrics to quantify impact takeoff speed, and consider a few different ways to quantify capabilities takeoff speed.

Quantifying *impact* takeoff speed

GDP is a useful impact metric because of its correlation with things like military power, technological progress, and the total productive capacity of civilization.

I believe sufficiently advanced AI would dramatically accelerate GDP growth. Moreover, I expect GDP growth to be steady or slow over time absent the development of sufficiently advanced AI or a small handful of other possible breakthroughs such as some forms of radical biotechnology. If GDP growth accelerates and we don't observe other compelling causes besides AI advancement, then I think it will be reasonable to attribute the vast majority of that GDP acceleration to AI advances, and thus to use GDP growth acceleration as a measure of AI impact takeoff speed.

With this background, it is natural to identify fast takeoff with a sudden increase in GDP growth and slow takeoff with a gradual increase in GDP growth.

Paul Christiano operationalises slow takeoff as follows:

There will be a complete 4 year interval in which world output doubles, before the first 1 year interval in which world output doubles. (Similarly, we'll see an 8 year doubling before a 2 year doubling, etc.)

Intuitively, this is slow takeoff because AI has a moderately transformative impact for 8 years before it begins to have a massively transformative effect.

Paul's key metric here is the ***ratio between successive GDP doubling times***. E.g. suppose GDP doubling times are as follows: 24 years (~current rate) → 8 years → 2 years → 1 year. The ratios in this example are 3, 4, and 2. Paul's best guess in 2017 was that the ratios would equal ~2. So I call ratios of 4 or more 'fast takeoff' and ratios of 2 or less 'slow takeoff'.

Currently, my best guess is that there are a couple of ratios that are 3 or 4 during the transition, and then we settle down into ratios slightly less than 2. But it also seems plausible (>20%) that we get a ratio >8, and also plausible that all the ratios are <2.

How do these ratios compare with those observed historically in global GDP? I calculated these from David Roodman's data set.

Year	Doubling time (years)	Ratio between successive doubling times
-5000		
-3000	1293	
-2000	1014	1.3
-1000	1081	0.9
-500	480	2.3
-200	493	1.0
1100	1055	0.5
1500	539	2.0
1820	223	2.4
1870	83	2.7
1913	33	2.5
1940	31	1.1
1962	26	1.2
1977	15	1.7
2000	23	0.7
2019	19	1.2

The 5 doublings since 1913 all had ratios <2; the four doublings from 1100 - 1913 all had ratios between 2 and 3. AI impacts found that there was plausibly a ratio of >4 around the agricultural revolution, where my table begins.

We can also use GDP to define serial time metrics of takeoff speed. The metric I currently use is: ***time from 5% GDP growth to 20% GDP growth***. 5% is significantly higher than the recent rate of 3%, so it seems like a good indicator of "crazy stuff is happening".[4] [5] 20% is fast enough that I expect humans are struggling to keep up with developments.[6]

[4] The last time annual growth for one year exceeded 5% was 2006. The last time the 5-year average for GWP growth exceeded 5% was 1974. Source.

[5] We should exclude 5% growth if it's driven by recovery from a disaster or war here. The growth should be driven primarily by frontier technological progress.

[6] For serial time metrics, AI value-add to GPD is a concrete way to define a startpoint. E.g. "AI is adding $5 trillion / year to global GDP".

Which type metric is better? Serial time metrics are easier to understand, but the startpoint and endpoint are pretty arbitrary. The ratios between successive GDP doubling times is less arbitrary, but it's a more abstract quantity and so harder to think about.

It's worth noting that, like any impact metric, GDP can in principle decouple strongly from AI capabilities. For example, in the growth model I ultimately use in this report, physical capital can strongly bottleneck GDP even as the amount of cognitive labour from AIs becomes extremely large.[7] This reflects the idea that certain physical inputs are essential to (e.g.) building a house, and no amount of cognitive labour can replace them. As a result, going from 100 billion to 1 trillion AGIs might increase GDP by much less than 2X. I do expect this bottleneck dynamic to apply to some extent. But I think tracking AI capabilities explicitly when they diverge from GDP impacts is very important, so I primarily emphasise AI capability metrics (discussed below) while also reporting GDP metrics.

Another limitation of economic metrics is that they are lagging indicators of AI capabilities. Economic signs may only appear long after dangerous capabilities are developed.[8]

What about quantifying impact takeoff speed without using GDP? You could consider other domains, e.g. military power or level of SOTA technology. I haven't thought about how to precisely quantify takeoff speed in these domains, but the rough idea is time from "AI makes significant difference to the domain" to "AI is making abilities in this domain go through the roof". Takeoff speed can be different in different domains (more).

Quantifying *capability* takeoff speed

Metrics of AI capabilities typically measure performance of specific systems at narrow tasks, e.g. error rate on a specific benchmark or cluster of benchmarks.

For takeoff speeds, though, I'd like a quantity that describes the collective capabilities of all AI systems across all cognitive tasks. I'm not aware of a way of quantifying this that is straightforwardly measurable. The quantities I'm using come from the growth modelI use to estimate the effects of partial AI automation on R&D and GDP, and I think they will have meaningful analogues in the real world. However, they are mostly not straightforwardly measurable, at least not today. I sometimes describe these quantities as 'metrics of takeoff'; but by this I just mean that they quantify takeoff speed, not that they're straightforwardly measurable.

[7] More precisely, if we hold the levels of physical capital and technology fixed and increase cognitive labour to infinity, GDP only increases by ~4X in any amount of time. It only increases further once the level of technology, or amount of physical capital, increases. I discuss this further in section 6.

[8] Dan Kokajlo critiques GDP metrics of takeoff speed along similar lines.

The % of cognitive tasks that AI can readily perform

In this report, I will use "AGI" to refer to an AI system, or collection of systems[9], that can readily perform ~all cognitive tasks.

We can generalise this notion to that of *AI that can readily perform x% of cognitive tasks*.[10]

If AI can perform some tasks, but not others, how can we quantify the exact % of tasks it can perform? To answer this, we need some way to assign a *weight* to each task that quantifies its importance. I weight each task by its economic value in 2020, as measured by the total $ that people earn while performing the task.[11] **Throughout the report, whenever I refer to the % of cognitive tasks – or the fraction of cognitive tasks – I am weighting different tasks by their 2020 economic value.** I explain this concept in more detail and discuss its weaknesses in an appendix.

With this notion at hand, we can define metrics of the form: *Years from when AI can readily perform x% of cognitive tasks to AI that can readily perform y%*.

This metric ignores the question: *At what cost can AI perform the task?* My reason is compute is already cheap enough that we could run a human brain for ~$10/hour.[12] Compute prices will continue to fall, so I expect that once AI can perform the task it will be able to do so more cheaply than a human.[13]

Throughout this report, whenever I say that “AI can perform” a task, I mean that it can *readily* perform the task. The phrase “readily” here indicates that i) it would be profitable to do the engineering and workflow adjustments necessary for AI to perform the task in practice, and ii) these adjustments could be done within 1 year if organisations made it one of their priorities.

[9] I drop the “or collection of AI systems” henceforth for brevity, even though this framework is most naturally interpreted as implying that AGI will take the form of many AIs.

[10] Example coarse-grained tasks include proofreading a document, writing a poem, checking a maths proof, writing code to perform a specified function, generating a strategy to meet a specified objective, giving medical advice, etc. Each of these tasks has many subtasks, which may themselves have subtasks. The tasks in this document should be thought of as the lowest level subtasks, as then we need not consider cases when AI can partially perform a task by performing some but not all its subtasks.

[11] More precisely, the weight of task T is proportional to the total $ people earn while performing T. For each person, this is given by the time they spend on T multiplied by their hourly salary. In mathematical notation: weight_T = SUM_i($ earned performing T by person i) / SUM_i($ earned by person i).

[12] You can rent an A100 for $1/hour and it produces ~1e14 FLOP/s. A standard median estimate of human brain FLOP/s (to the extent that’s a meaningful concept) is 1e15 FLOP/s. That implies you could run a human brain for $10/hour.

[13] Two caveats.
First, if there’s large demand for AI chips when AIs are adding $trillions to the economy, this could drive prices back up somewhat. I haven’t analysed how significant this effect could be; it will depend on how quickly chip production can be increased to meet demand. This implies that the *supply* of computer chips will be the key bottleneck of how many AIs we run.
Second, it may be that the first time AI is able to perform a valuable economic task it is very expensive to run, and then the price falls over time. I discuss a model along these lines here. In this case, it is more meaningful to track when it becomes profitable to actually *automate* cognitive tasks, as opposed to when AI can first perform the tasks.

That is, if the AI *could in principle* perform the task if humans did a lot of work restructuring workflows and generating suitable inputs, but in practice it would take a lot of work for the AI to do this task in practice, then the AI can **not** readily perform the task (as I'm using the phrase). If only a relatively small amount of work is needed, however, then the AI can perform the task.[14]

The % of cognitive tasks that AI could fully automate

Imagine some AI can perform 50% of cognitive tasks, but there's only enough runtime compute to run one such system. In this case, AI could not fully automate 50% of tasks because we can't run enough AIs to replace all the human workers.

If AI can readily perform x% of tasks, and there's enough runtime compute[15] for AI to replace all the human workers in those tasks,[16] then I'll say that *AI could readily fully automate x% of tasks*. (I sometimes omit "readily", but it is always implied.)

With this notion we can define takeoff metrics of the form: *Time from AI that could readily automate x% of cognitive tasks to AI that could readily automate y%*.

This is a similar metric to the last subsection, but it relates not only to the capabilities of individual AIs, but to how many AIs we can run in total.

The metric I mostly focus on in the report is: *Time from AI that could readily automate 20% of cognitive tasks to AI that could readily automate 100%*.

How many AGIs can we run?

The Full Takeoff Model (FTM), discussed in the summary, makes assumptions or predictions about:

- When we'll train AGI for the first time (AI that can perform 100% of cognitive tasks).
- The FLOP/s to run AGI.

[14] Somewhat more precisely, it should take <1 year of engineering and adjusting workflows before AI can perform the task in practice, and it should be profitable for organisations to make necessary workflow adjustments.

[15] But at what price? As discussed above, I expect that once AI can perform the task it will be able to do so more cheaply than humans. (Though see earlier caveats in a footnote.)

[16] For AI to replace all human workers at a task, the new AI output at the task must exceed the previous human output at that task. For example, suppose humans worldwide write 1 billion emails per day. Then for AI to replace all human workers at the task of email writing, AIs must be able to write more than 1 billion (similarly productive) emails per day. In the model of automation I use, which I explain below, once AI output rises a little above this level it becomes profitable for all human workers to work on new tasks that haven't yet been automated (e.g. to spend all their time doing things other than emails). The numbers I report for this metric correspond to this profitability point, so are model dependent.
If previous automation has already concentrated human workers on some cognitive task, then this raises the bar for replacing all humans at that task. E.g. if automating emails causes humans to spend more time coding, then you'll need more AIs to replace humans at coding.

- How many FLOP/s we can do at each point of time.

This means it can calculate *the first year in which we have trained AGI and can run X AGIs*, for any X. I like this as "***year when we can run 10 billion AGIs"*** as an endpoint signifying when AIs' collective cognitive abilities significantly exceed the collective cognitive abilities of humans.[17] [18]

Cognitive output

By "cognitive output" I mean the progress per unit time on software R&D and in other cognitive domains like maths, strategy, persuasion, etc. The restriction to *cognitive* domains, to the exclusion of tasks that require physical labour, captures the idea that disembodied AI will automate the stuff done by the human brain but won't (without robotics) automate physical human labour.

My preferred unit for AI cognitive output, or for AI+human combined cognitive output, is "How many remote[19] human workers would it take to add the same amount of value?" So if AIs + humans make some software progress in one month, and you'd have needed 1000 human workers to make the same amount of progress in one month without AI, then the total cognitive output of AIs + humans is "1000 remote human worker equivalents".

Notice that in this example I looked at the total cognitive output from both humans and AIs *combined*. Until we have AGI, humans and AIs are complementary to each other, so it's hard to separate out the cognitive output that's due to "AI alone".[20] I view it as a benefit of this metric that it naturally incorporates this complementarity. Another benefit is that it avoids privileging an arbitrary capability level like 'AGI'.

We can separate out a notion of the cognitive 'value add' of AI by comparing the cognitive output that *would* obtain if you only had human workers (with no AIs[21]) with the actual cognitive output produced by the combination of humans and AIs. If the latter quantity is twice as high,

[17] I say "significantly" because AGI will have a number of significant cognitive advantages over humans. To list a few: run faster in serial time, smaller % of AIs in education and retirement, smaller % of time spent on leisure or sleeping, can use smaller models for easier tasks rather than doing all tasks with a fixed brain size.

[18] Do AI's individual cognitive abilities also exceed those of humans by this point? Not necessarily. The framework sits most naturally with a comprehensive AI services interpretation of AGI, where no single AI has abilities as general as an individual human (more). But my personal expectation is that very soon in calendar time after AIs can *collectively* do all tasks a human can do, we'll be able to develop a unified AI system that exceeds humans at ~all cognitive tasks. So I *do* think that some AIs' individual cognitive abilities will exceed humans' by this point.

[19] Remote human workers because disembodied AIs won't be able to do tasks involving physical labor.

[20] As a concrete example, let L be the number of humans and C the number of AIs. Suppose cognitive output is given by L*C. It's hard to attribute a fraction of this output to humans vs AIs, due to the complementarity (in this case represented via multiplication).

[21] Or, more precisely, with no AIs developed after 2020. (We *already* use AI to help us perform cognitive tasks, and I don't want to exclude them. I just want to exclude new AIs that automate additional cognitive tasks.)

then the AI's cognitive value add is 2X. With this in hand, we can define the following takeoff speed metric: time from AI value add being 2X to it being 10X. This period begins when cognitive output is twice what it would be absent AI, and ends when it is 10X what it would be absent AI.[22]

I explain how the FTM (Full Takeoff Model) calculates cognitive output here.

Impact metrics vs capability metrics

This piece will make forecasts about impact metrics and capability metrics. How much stock should we place in each?

I have greater trust in the forecasts of capability metrics. Forecasts of impact metrics involve forecasting capabilities *and* making substantial additional[23] assumptions about how those capabilities translate into impact. Example assumptions:

- How much does **lack of physical equipment** or physical labour delay or reduce the impact of advanced AI? (We discussed this briefly above.)
- How much do **regulations** delay or reduce the impact of advanced AI?
- How much **schlep** is involved in integrating advanced AI in the economy?

These additional assumptions will tend to make forecasts of impact metrics more uncertain than forecasts of capability metrics.

In addition, the correct additional assumptions might differ in different domains. For example, perhaps lack of physical equipment will significantly bottleneck how much AGI accelerates **technological progress**, but won't prevent AGI from giving its controller a huge **military advantage**. Or perhaps regulations will prevent AGI impacting **goods and services** but not **software R&D.** So a second advantage of capability metrics is we can make separate judgements about how AI capabilities impact multiple different domains.

On the other hand, the capability metrics are less meaningful. In particular, they will be much harder to measure and track over time, and are at some risk of involving made-up concepts derived from a growth model but not grounded in reality.

[22] Of course, AI may automate cognitive tasks without being agentic. If AI cognitive value add is 10X, but AIs do not make plans and are not strategically aware of humans and the levers of power (see Joe Carlsmith's draft report on AI risk), this may be much less risky than if AIs *do* make plans and *are* strategically aware.

[23] To some extent, these additional assumptions also affect forceasts of AI capabilities. The impacts of AI on GDP and R&D accelerate future capability developments. However, I model the effect on bottlenecks on this feedback, and don't expect large delays from regulations and schlep in back-end industries that will spur further AI development like AI R&D and chip manufacturing.

Other metrics of takeoff speed

There are a number of other serial time metrics of takeoff speed that seem plausibly useful. For example:

- Time from "*AI that causes >30% of world leaders to realise that AI will be transformative*" to "*AI that gives its controller a decisive strategic advantage"*
- Time from "*misaligned AI that blatantly seeks power*" to "*AI that causes existential catastrophe if it's misaligned*".
- Time from [weaker AI capability that is strategically significant] to [stronger AI capability that is strategically significant]

There will be no straightforward way to get predictions about these metrics from my framework. To do this, we'll have to translate the AI capabilities that feature in these metrics into the language of the framework. This means mapping them to the rate of GDP growth, the % of 2020 cognitive tasks that have been automated, the number of AGIs that can be run, [the AI multiplier on cognitive output], or some other quantity that can be calculated by the model.

Summing up

We can distinguish between capabilities takeoff speed and impact takeoff speed, and have reasons to care about both. Impact takeoff speed might be more important, and it can be quantified using GDP metrics that are well grounded. Capabilities takeoff speed might be easier to predict, and it can be quantified using a few different metrics that are less well grounded. The Full Takeoff Speeds Model I'll explain during the next few sections will make predictions about all the metrics I've mentioned.

3. Basic framework for calculating takeoff speed

I recommend skipping this section except that part that estimates the size of the effective FLOP gap. The rest just recaps Bio Anchors and explains the basic framework for thinking about takeoff speeds a little more slowly than in the long summary.

This section presents a very basic first-pass framework for thinking about takeoff speeds. It describes a simple extension you can make to the bio-anchors framework to get an estimate of takeoff speed.

In short, we first use the biological anchors framework to estimate the FLOP needed to train AGI. Then we add an additional assumption about the FLOP needed to train some weaker AI. Lastly, we estimate how quickly we can ramp-up training FLOP between these two points. This gives us the calendar time from the weaker AI to AGI, one metric of takeoff speed.

The rest of this section explains this basic framework in more detail. Later sections expand upon it by i) analysing how increased AI investment and incremental AI automation might affect the ramp-up of training FLOP, and ii) modelling the effect of AI automation on economic growth.

Bio anchors recap[24]

Ajeya Cotra's biological anchors report (hereafter, "Bio Anchors") articulates a framework that can be used to estimate when we'll train AGI,[25] via estimating when we'll have enough compute and software to do so.[26]

In particular, it uses analogies with biological systems and trends in ML to estimate the **FLOP required to train AGI using 2020 algorithms.** That is, if AI algorithms had frozen at their 2020 levels and a multi-year concerted effort had been made to train AGI, how many FLOP would have been sufficient to succeed?

The bio-anchors report also estimates how the size of our training runs will change over time. One tricky element here is algorithmic progress: we can achieve more with each FLOP in 2025 than in 2020. To incorporate this, we can measure the size of training runs in units of *2020-FLOP*, meaning "How many FLOP would have been needed to train a system with these capabilities using 2020 algorithms?" Software progress increases the number of 2020-FLOP that are available from a fixed budget of FLOP.

The 2020-FLOP used in a training run can be calculated by multiplying together three quantities:

1. **$ on training FLOP**. How much is spent on the training run?
2. **FLOP/$.** How many FLOP does each $ buy us? This increases over time due to hardware progress.
3. **Software multiplier.** How many times more efficient are today's algorithms than 2020 algorithms? E.g. if we could train AGI today using half as many FLOP as we'd have needed in 2020, the software multiplier equals 2.
    a. The unit for software is **2020-FLOP per FLOP.** I.e. each FLOP today corresponds to multiple 2020-FLOP because algorithms have improved.

Writing this as an equation:
2020-FLOP = $ on FLOP * FLOP/$ * 2020-FLOP per FLOP

[24] Here I give a dense summary of the relevant points; readers without familiarity might want to read a summary (here or here), listen to part of this podcast, or read the full report.

[25] The report actually focuses on forecasting a slightly different target: *transformative AI*, defined as AI which increases the rate of economic growth by ~10X. But the same framework can be used for forecasting AGI, and this use-case will be more useful for our purposes. In what follows, I'll talk as if the report was forecasting AGI, to simplify the exposition. The difference will matter later because AGI is plausibly harder to develop than TAI, and we'll adjust the report's output for this fact.

[26] Specifically, Bio Anchors estimates when we'll have enough computation to train *one unified AGI system*. This is aggressive because we might actually achieve AGI earlier via many distributed cheaper systems, but it's conservative because there are inputs to developing AGI other than computation (e.g. data).

Let’s call the FLOP needed to train AGI using 2020 algorithms the **AGI training requirement** (notice that it’s in units of 2020-FLOP). When the 2020-FLOP used in a training run exceeds the AGI training requirement, bio anchors forecasts that we will train AGI.

Extending bio anchors to estimate one metric of takeoff speed

Bio anchors estimates the training requirements for AGI, measured in 2020-FLOP. If we add an additional assumption about the training requirements for some weaker AI system, we can estimate the *calendar time* between training the weaker system and training AGI via the growth of the 2020-FLOP used in training runs. This in turn depends on the growth of its three components: $ on FLOP, FLOP/$ and 2020-FLOP per FLOP.

Concrete example

Let’s go through a concrete example to illustrate this idea.

Suppose bio anchors estimates that the 2020-FLOP AGI training requirement = 1e36.[27] I.e. it would take 1e36 FLOP to train AGI using 2020 algorithms. Then we additionally estimate that some weaker AI would take 1e30 FLOP to train using 2020 algorithms.

Then the serial time between the weaker AI and AGI is simply the time to increase the 2020-FLOP used in training runs by 6 OOMs.[28] How long will this take? It depends on how quickly the three components of 2020-FLOP grow after we’ve trained the weaker AI. Let’s make the following assumptions:

1. **$ on training FLOP** has a growth rate of 30%.
2. **FLOP/$** has a growth rate of 40%.
3. **2020-FLOP per FLOP** has a growth rate of 40%.

The growth rate of 2020-FLOP used in the largest training run is related to the growth rates of its components as follows:
g(2020-FLOP) = g($ on FLOP) + g(FLOP/$) + g(2020-FLOP per FLOP)

So the 2020-FLOP used in a training run has a growth rate of 30+40+40 = 110%.[29] This implies that it takes 13 years to increase 2020-FLOP by 6 OOMs.[30] So we’d estimate the time from the weaker AI system to AGI as 13 years.

[27] Bio anchors places a probability distribution over the 2020-FLOP AGI training requirement, and I will ultimately do the same. For now though, I will proceed using point estimates to simplify the exposition.

[28] OOM = order of magnitude

[29] Note, this is an *instantaneous growth rate,* distinct from the *annual growth*. The former equals e^gt; the latter equals (1 + g)^t. The benefit of using the former is that you can add growth rates of the components to get the growth rate of the 2020-FLOP. They’re similar when g < 0.1.

[30] e^(1.1*12.6) = 1 million.

Comments on the concrete example

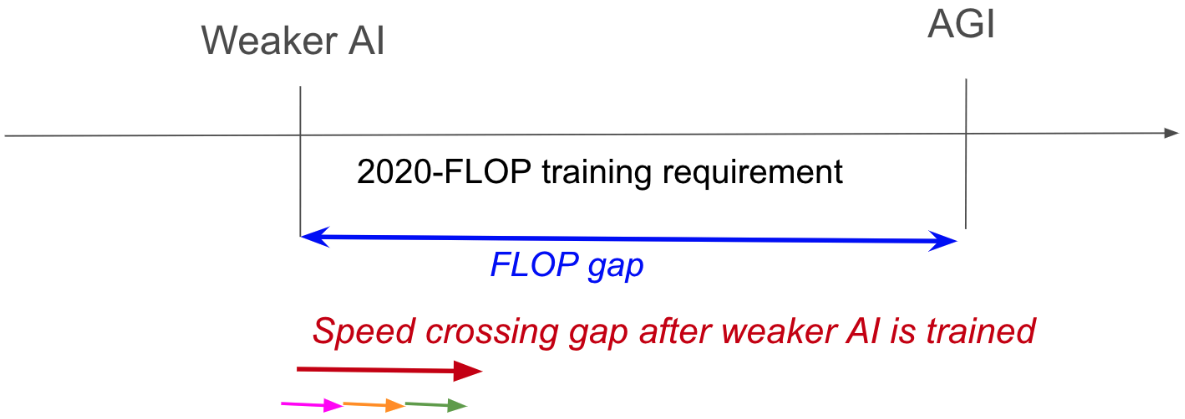
Firstly, the specific endpoint (AGI) and startpoint (some weaker AI) that I used here could be changed. E.g. you could use the startpoint "misaligned AI that blatantly seeks power" and the endpoint "AI that causes existential catastrophe if misaligned".

However, the startpoints and endpoints that we can use are still fairly limited at this stage. They must both correspond to some 2020-FLOP training requirement for the methodology to work. This is plausible if they refer to some *AI capability level*. We can't yet use startpoints/endpoints that refer to the *number* of AIs; e.g. we can't use the endpoint "*can run 10 billion AGIs*". Endpoints like this have a runtime computation requirement as well as a training requirement and we're not yet modelling the available runtime computation.[31] Also, we can't yet use startpoints/endpoints that refer to GDP growth, because I haven't introduced the constructs needed to calculate GDP. Later, with the Full Takeoff Speeds Model, we'll have the option to use these additional startpoints and endpoints.

Secondly, it's worth highlighting the structure of the calculation in the concrete example. It has two key inputs.

A. The **effective FLOP gap** between the startpoint and endpoint. In our example this was 6 OOMs. The effective FLOP gap's precise meaning is: how many more FLOP would it take to train [endpoint AI] than [startpoint AI], using 2020 algorithms.
    a. ***Note: I sometimes just refer to it as the "FLOP gap" for short rather than the "effective FLOP gap", but I always mean to refer to the effective FLOP gap.***
B. The speed crossing the gap, **g(2020-FLOP)**. In our example, 2020-FLOP had a growth rate of 110%, increasing by ~0.5 OOMs per year. The precise meaning is: what is the average growth rate of 2020-FLOP between [startpoint AI] and [endpoint AI].

[31] In the Full Takeoff Speeds Model, it turns out that the endpoint "can run 10 billion AGIs" typically comes very quickly (<2 years) after training AGI because AI automation causes hardware and software to improve extremely rapidly around this time *and* AGI training compute is so high that you're not too far off being able to run 1 billion AGIs by the time you've trained AGI. So the endpoint "train AGI" approximates the endpoint "can run 10 billions AGIs".


Weaker AI
AGI
2020-FLOP training requirement
FLOP gap
Speed crossing gap after weaker AI is trained

g(2020-FLOP) = g($ on FLOP)+g(FLOP/$)+ g(2020-FLOP per FLOP)

If we measure the effective FLOP gap in *OOMs* and measure g(2020-FLOP) in *OOMs per year*, we get the simple equation:

serial time to cross FLOP gap = FLOP gap / g(2020-FLOP)
= FLOP gap / [g($ on FLOP) + g(FLOP/$)
+ g(2020-FLOP per FLOP)]

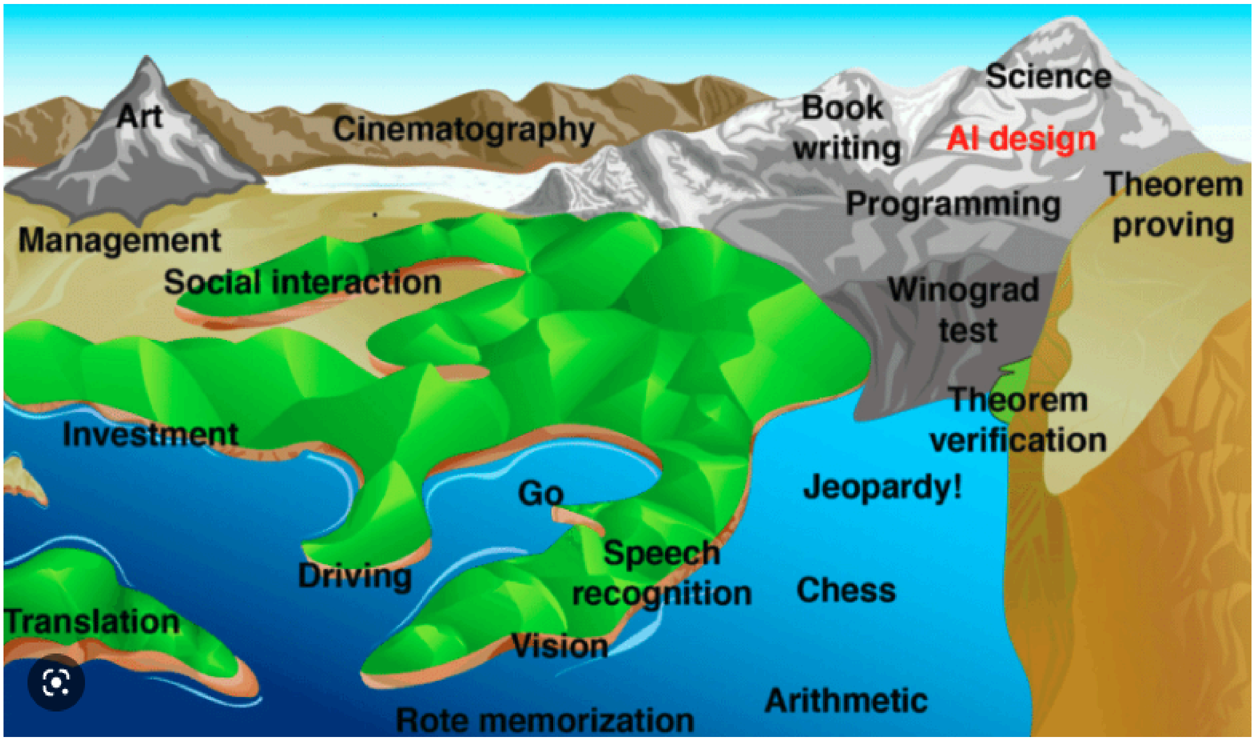
The rest of this section is organised as follows. First I briefly discuss considerations informing the effective FLOP gap. Then I state my bottom line about g(2020-FLOP) and compare it with the view implicit in bio-anchors.

Note, the report's main metric "time from AI that could readily automate 20% of cognitive tasks to AI that could readily automate 100%" implicitly makes reference both to whether AI could perform the tasks and to whether we can run enough AIs to replace humans at the tasks. So at this stage I can only calculate this metric by i) defining "weaker AI" as AI that can readily perform 20% of tasks, and ii) assuming that there will be enough runtime compute to automate tasks once AI can readily perform them (which is not true when AGI training requirements are low).

Evidence about the size of the effective FLOP gap

The choice of effective FLOP gap presupposes some startpoint and some endpoint. For concreteness I'll use startpoint = AI that can perform 20% of cognitive tasks, endpoint = AGI (AI that can perform ~100% of cognitive tasks). Crossing the effective FLOP gap involves going most of the way in capability space from today's AI to AGI. This input to takeoff speeds is second only to AGI training requirements in terms of being very important but very uncertain.

I think Hans Moravec's "rising tide of AI capacity" visualisation is useful for framing this discussion (h/t David Schneider-Joseph for this point). Currently AI can only do a small fraction of cognitive tasks – the areas of the map that are currently underwater. Over time the AI capabilities improve (the tide rises) and AI can perform more and more tasks. Eventually, AI can readily perform all cognitive tasks (everything is under water).



Hans Moravec's "rising tide of AI capacity" can help us think about the meaning of the effective FLOP gap

For our purposes, we should imagine the surface area of the landscape to be proportional to the tasks' economic value in 2020. (Or, if we're thinking about R&D automation, proportional to the task's share of R&D.) Then the effective FLOP gap tells us how much more effective training compute we need to cover *all* the landscape compared to just 20% of it.

There are a few factors that can weakly inform the choice of effective FLOP gap:

- **AGI training requirements** bound it from above.
- **SOTA AI capabilities** weakly bound it from below.
- **Horizon length** suggests it could be pretty big.
- **How AI capabilities vary with training FLOP between different domains** provides an estimate.
- **How AI capabilities vary with training FLOP within a domain** provides a low-end estimate.
- **How animal capabilities vary with brain size** provides a low-end estimate.
- **How human capabilities vary with brain size** provides a low-end estimate**.**
- **Practical barriers to partially automating tasks** suggest it could be very small.

I'll discuss each factor in turn.

- **AGI training requirements.** As of today the largest training run is ~3e24 FLOP.[32] My median guess is that AGI training requirements are 1e36 2020-FLOP,[33] so **my median effective FLOP gap can be no bigger than 12 OOMs**. More generally, shorter AI timelines lead to a smaller effective FLOP gap.
- **SOTA AI capabilities.** In my opinion, today's AI systems are not close to being able to readily perform 20% of all cognitive tasks done by human workers. (Actually automating these tasks would add ~$10tr/year to GDP.[34])
    - **a.** Based on this, and my rough sense for how much progress we're getting with each additional OOM (informed by looking at scaling papers and playing around with GPT-2 and GPT-3), I'd want to put my startpoint above 1e27 2020-FLOP.
    - b. Another rough-and-ready approach is to naively extrapolate recent trends in AI value-add and model size. This suggests the startpoint should be >3e28 2020-FLOP.
        - The data I could find suggests AI value-add is doubling roughly every 2 years,[35] over which time training runs have increased by ~1 OOM.[36] The largest training run as of July 2022 is ~3e24. If today's systems could readily add $500b/year to the economy, that would correspond to automating ~1% of cognitive tasks.[37] If each doubling of value-add continues to take ~1 OOM, AI won't automate 20% until >3e28 FLOP.[38]
    - c. Overall, **I'd want to put my startpoint above 1e27 2020-FLOP and probably above 1e28 2020-FLOP.** But if I condition on AGI requiring (say) 1e30 2020-FLOP then I'd want to make it lower.
- **Horizon length.[39]**
    - a. Suppose we automate some economic tasks using a horizon length of 1 second but training AGI requires a horizon length of 1 year with the same model size or bigger. This implies a effective FLOP gap of >7.5 OOMs.[40]

[32] I believe these were the requirements for PaLM.

[33] The Bio Anchors best-guess median training requirement for TAI is 1e35; I add 1 OOM to account for AGI being harder than TAI.

[34] World GDP is ~$100tr, about half of which is paid to human labour. If AI automates 20% of that work, that's worth ~$10tr/year.
[This is a bit aggressive, as many tasks have a component of physical labour (though all have some cognitive component). On the other hand, AI will probably produce *more* output at those tasks than the humans they replace (as they're cheaper to run), increasing their value-add.]

[35] E.g. here, here, here, here. I don't know how reliable these estimates are, or understand their methodologies.

[36] Epoch's piece.

[37] World GDP is ~$100tr, about half of which is paid to human labour. If AI automates 1% of that work, that's worth ~$500b/year.

[38] The bound would be higher if I used a number below $500b, or if I included software progress. OTOH, value-add per OOM of training FLOP may rise.

[39] This concept is from bio anchors. Ajeya defines it as follows: *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*.

[40] There are 30 million seconds in a year.

  b. My median guess would be that, compared to this, the startpoint will require longer horizons and we will train AGI with shorter horizons. So my extremely tentative takeaway would be that the effective FLOP gap is **>5 OOMs**.[41]
  c. I put (even) less weight on this consideration than the others because the usefulness of the 'horizon length' concept is debatable and I don't know whether the startpoint-AI will use a very different horizon length to AGI.
  d. If I condition on very large training requirements for AGI, however, I become more convinced by this consideration. At that point I think AGI requires very long horizons, but still think that short horizons will produce significant economic value.

- **How AI capabilities vary with training FLOP *between different domains*.** How much do training FLOP requirements vary across *different tasks or domains*?
  a. **AI has strong comparative advantages in some domains relative to others.** Probably, less training FLOP will be needed for automating human work in domains like these. Indeed, this is how AI can *already* perform at super-human level in some domains. There are many mechanisms that can give AI comparative advantages at some tasks but not others:
     - Some tasks can be performed by AI with "short horizon training", others require "long horizon training".[42]
     - Some tasks require strong sim2real transfer.
     - Some tasks are more similar to tasks used in AI pre-training.
     - Some tasks have much more available data from demonstrations or human feedback.
     - Some tasks benefit from memorising lots of information.
     - For some tasks it's important to "always be on" (no sleep), or to consistently maintain focus (no getting bored or slacking).
     - For some tasks, it's much easier to verify that an answer is correct than to generate the correct answer.
     - People choose to develop specialized AI architectures and training processes for some tasks but not others.
     - Some tasks require very high levels of reliability (e.g driving vs drafting an email).
  b. **GPT-N task performance.**
     - Lukas Finnveden has extrapolated the performance of GPT-N on a variety of benchmarks. You could look at the graphed extrapolations to compare the (predicted) training FLOP needed to solve different benchmarks. In the linear extrapolation, the first benchmark exceeds a score of 90% ~4 OOMs before the last benchmark. In the sigmoid extrapolation, the gap is ~5 OOMs. So we're on ~4-5 OOMs.

[41] There are ~5 OOMs between a "10 second" horizon length and a "1 month" horizon length.

[42] This concept is from Bio Anchors. Short horizons means that the model only needs to "think" for a few seconds for each data point; long horizons means the model needs to "think" for months for each data point and so training requires much more compute.

        - Both extrapolations omit the task with the worst SOTA performance (ANLI).
        - On the other hand, they both include the text-completion task, which is very similar to GPT's training task. Omitting this lowers the estimated effective FLOP gap to **~4 OOMs**.
    - As before, this scaling would be smaller using Chinchilla scaling laws, and perhaps smaller still the scaling law we're on when we start crossing the effective FLOP gap. So I'll reduce that gap to **~3 OOMs**.
    - How does the gap for the extrapolated benchmarks compare to the gap we might see for economically valuable tasks? I expect the latter gap to be bigger. These benchmarks were selected (in part) for being appropriately challenging to SOTA LMs, which will narrow the range of difficulty between them. Economically useful tasks will not have this selection pressure, and are generally much more varied in general (e.g. they're not all language based). I'll tentatively add another 2 OOMs for this.
    - So overall I take this consideration to suggest that the effective FLOP gap is >3 OOMs and use a tentative best-guess of **~5 OOMs**.

c. **RL vs transformer training FLOP**
    - AlphaStar had 139 million parameters[43] and took ~2e23 FLOP to train.[44] GPT-1 had a similar number of parameters (117 million) but only took 1e19 FLOP to train. That's a difference of 4 OOMs.
        - The difference is probably partially due to AlphaStar having a longer horizon length, and I discussed horizon length above. But other factors contribute, like the use of league-based training and training data being more noisy.
    - I think this is representative of a broader pattern of RL systems having several OOMs more training FLOP than similarly-sized LMs (where size is measured in parameter count, or in FLOP per forward pass).
        - [Someone could check this by looking at Xland.]
    - *If* some economically valuable tasks will be automated by transformer architectures of a certain size but other tasks are only automated by RL systems of a similar size, *then* I'd expect the effective FLOP gap to be **>4 OOMs**.
    - This would combine with other sources of a wide effective FLOP gap.

d. To sum up:
    - AI comparative advantages: suggests that the gap is wide in general.
    - GPT-N: >3 OOMs, ~5 OOMs.
    - RL vs transformer: *maybe* implies an additional ~4 OOMs.

e. Overall, I interpret this 'training FLOP differences in different domains' consideration as suggesting that the effective FLOP gap is **>3 OOMs**, weakly

[43] See page 3 here.
[44] From eyeballing figure 3 of *Compute Trends Across Three Eras of Machine Learning*.

suggesting a best guess of **~5 OOMs**, and lending plausibility to amounts as high as ~8 OOMs.

- **How AI capabilities vary with training FLOP *within one domain*.** When you increase training 2020-FLOP within one domain, how much do capabilities improve? What increase in training FLOP is needed to cross the human range?
    a. **GPT-N**. I believe GPT-3 required about ~2.5 OOMs more training 2020-FLOP than GPT-2.[45] People familiar with both systems can use the difference to intuit how much performance improves with more training FLOP. Intuitively, the difference is pretty big! My extremely rough sense is that once AI can readily perform 20% of language based tasks, you'd need ~two similar-sized improvements before it could readily perform all language based tasks,[46] suggesting an effective FLOP gap of **~5 OOMs**.
        - The difference in training FLOP between GPT-2 and GPT-3 would be smaller with the new Chinchilla scaling law, and by the time we're crossing the effective FLOP gap we may be using better scaling law still. (For example, certain prompting or fine-tuning or aggregation techniques may improve how much total performance increases with scale, as suggested by figure 3 here.) So I'll put my overall estimate here at **~4 OOMs**.

    b. Other papers on "scaling laws" could potentially be similarly informative here.
        - The LM-scaling papers I've glanced at, seem broadly consistent with the above, with ~2 OOMs of training FLOP improving the score on large aggregations of benchmarks (e.g. BIG-Bench, MMLU) by ~10-40%.

    c. **Go.** The difference between an intermediate amateur Go player and the best in the world is ~2800 Elo.[47] Marginal doublings of training FLOP improved AlphaGo's Elo by ~300 - 700.[48] This implies that ~4 - 9 doublings of training FLOP are needed to cross the human range, or **~1 - 3 OOMs**. I put more weight on this than the GPT comparison, as the comparison to human abilities is more grounded.
        - Though given that we're talking about the effective FLOP gap for *economic value*, we should make the startpoint when you can get paid for your performance. If we measured the difference from "a bit below professional" to "best in the world", the Elo gap would be less than half as big.[49] Then the FLOP to cross the range would be **~0.5-1.5 OOMs**.

    d. So to sum up:

[45] This data suggests GPT-3 took ~100X more training FLOP; I'm assuming ~3X algorithmic improvements on top of that.

[46] Obviously, comparing the qualitative capability gap between GPT systems to the gap between top-performing and low-performing humans is fraught. One worry is that we're less able to notice intelligence differences between systems much dumber than us, compared to systems similarly intelligent to us.

[47] See this sheet.

[48] From figures 3, 4 and 5 of the AGZ paper. See reasoning on this sheet.

[49] The world's best Elo is 3800, Go's "professional" range starts at 2700 Elo, while and Elo corresponds to "advanced amateur". Using 2400 Elo as the start point would give us a "human range" of 1400 Elo.

        - GPT-N scaling: ~4 OOMs
        - Go: ~1 OOM
    e. These results underestimate the effective FLOP gap because crossing the human range *in one domain* is easier than crossing the range *across all economically valuable tasks.* I.e. they ignore the above point of AI having comparative advantages at some domains over others.
        - This is especially true for Go, which is just one game.
        - It's only somewhat true for GPT-N, as the full space of cognitive tasks isn't that much broader than language-based tasks.
    f. So overall I interpret this as suggesting **>4 OOMs** as a soft lower bound, and **~5 OOMs** as a very tentative best guess.

- **How animal capabilities vary with brain size.**
    a. One comparison here is humans vs chimps. Human brains are probably ~3X bigger than chimp brains (in terms of FLOP/s),[50] and there is arguably a large gap in cognitive abilities. Perhaps humans have an additional ~3X from software improvements. If you think "chimp → human" is enough to cross the effective FLOP gap, this implies a effective FLOP gap of **~1 - 2 OOMs**.[51]
    b. Rat brains are 2 OOMs smaller than human brains (in terms of synapses and so FLOP/s). Perhaps accounting for software, the difference is 3 OOMs. It's not crazy to me that rats could perform 20% of cognitive tasks if they'd been selected to do so. This suggests an effective FLOP gap of **~4-6 OOMs**.[52]
    c. I find the chimp comparison more plausible.
    d. Both approaches ignore AI having comparative advantages at some tasks over others, so underestimate the gap.
    e. Overall, I see this as lending weight to effective FLOP gaps as low as **1 OOM**, and weakly suggesting a best-guess of **~3 OOMs**.

- **Brain size - IQ correlations within humans.** [53]
    a. There's a few decent-seeming papers that estimate the correlation between brain volume and IQ. My conclusion from a few hours looking at these was that a 10% increase in brain volume might cause a gain of ~4.5 IQ points. More.

[50] Chimps have about 3X fewer neurons than humans, and the data in figure 4C of this paper suggests they have a little over 3X fewer synapses. (Synapses are closer to what we care about for estimating FLOP/s.)

[51] Two methods: Method 1 anchors to *runtime* FLOP/s, method 2 anchors to total *lifetime learning* FLOP.
Method 1: Human runtime FLOP/s is ~10X bigger (including software gains), implying 100X more training FLOP with Chinchilla scaling.
Method 2: Human lifetime learning uses ~3X more FLOP than chimps as the learn for about the same length of time. Bump this up to 10X for better software for learning. Then assume training FLOP will be proportional to lifetime learning FLOP. Method 2 assumes ML training-run scaling will be as good as hominid life-learning scaling. Hominid scaling might be *better* because the life-learning algorithm is better than our training algorithms; it might be *worse* because it doesn't scale data optimally with brain size but we will be able to do this in ML.

[52] Two methods: Method 1 anchors to *runtime* FLOP/s, method 2 anchors to total *lifetime learning* FLOP.
Method 1: 3 OOMs bigger brain → 3 OOMs more runtime FLOP/s → 6 OOMs more training FLOP with Chinchilla scaling.
Method 2: 3 OOMs bigger brain and 1.5 OOM longer childhood → 4.5 OOMs more training FLOP.

[53] I've largely taken this line of reasoning from this doc by Paul Christiano.

b. If we assume that brain volume is proportional to brain FLOP/s, then a 10% increase in brain FLOP/s causes a gain of 4.5 IQ points.
   - Between primates, it seems like # neurons and # synapses are roughly proportional to brain volume, confirming this assumption.[54]
   - It seems that *within humans*, brain volume is more slightly anti-correlated with neuron density, implying that a 10% increase in brain FLOP/s could have a larger effect on IQ.[55] For now, I'll err conservative and leave this out.

c. In line with the spirit of Bio Anchors, I'll *assume that a 10% increase in AI model size (measured in FLOP/s) has the same impact on IQ as a 10% increase in human FLOP/s*.[56] [I'll later consider a more aggressive assumption.]

d. So 10% bigger AI model size → ~4.5 IQ points.

e. So a 10X bigger AI model → ~24 10% increases in model size[57] → ~110 IQ points.

f. Let's assume training FLOP increases with the square of model size.[58] Then 100X more training FLOP → 10X bigger model → ~24 10% increases in model size[59] → ~110 IQ points.

g. Intuitively, going IQ 45 → IQ 155 would cross the effective FLOP gap (initially able to perform <20% of economic tasks → then able to perform ~100%). So **this naively suggests an effective FLOP gap of 2 OOMs**.

h. You could plausibly end up with a smaller effective FLOP gap:

[54] I expect # synapses to be proportional to FLOP/s. Figure 4A of this paper finds the # synapses per unit volume is constant. If neurons matter, this paper claims the # neurons is slightly less than proportional to volume (# neurons = volume^0.9). (Would be interesting to check if these two claims are consistent with the slight increase in synapses per neuron observed in figure 4C of the first paper.)
This paper claims the number of neurons is proportional to brain volume and

[55] Pakkenberg & Gundersen 1997 (N=94) is the only thing I'm aware of studying correlations between brain volume and neuron or synapse count in humans (thanks to Tegan for sharing). Bottom line: I think the data naively imply a 10% increase in FLOP/s would add 5.8 IQ points. I may have made a math mistake though.
Figure 4 shows that brain volume is anti-correlated with neuron density (# neurons per unit volume), so that an 85% increase in brain volume is only associated with a 61% increase in neuron count. This implies that each 10% increase in brain volume increases neuron count by 7.7%, with humans. (I.e # neurons = volume^0.77.) Assuming FLOP/s per neuron is constant, each 10% increase in brain volume increases FLOP/s by 7.7%. So a 10% increase in FLOP/s would be equivalent to a 13% volume increase and increase IQ by ~5.8 IQ points. (4.5*1.3=5.8.)
We can sense-check this using the raw correlation between neuron count and brain volume reported in table 3. The correlation is 0.71. This is broadly consistent with 0.77 from above, as neuron count had larger variance than brain volume. (E.g. If 1 SD of brain volume was 5% of the mean brain volume, then 1 SD of neuron count was >5% of the mean neuron count.)

[56] An increase in AI model size might be better than human brain size due to the ability to do more calculations in series, or because we'll make small and easy adjustments to AI algorithms to make them suited to the new scale.

[57] ln(10)/ln(1.1) = 24.2

[58] I.e. Chinchilla scaling. The basic rationale here is double the FLOP/s → double the # parameters → double the # data points needed for training. Also double the FLOP/s → double the FLOP per data point. With 2X the data points and 2X FLOP per data point, training FLOP increases 4X.

[59] ln(10)/ln(1.1) = 24.2

        - *Assume training compute is proportional to lifetime learning compute.* Humans with bigger brains don't get more "data" (experience) to learn from, so a 10X bigger human brain would use only 10X as much compute to learn. Anchoring to this, we might need to only use 10X as much training compute, an effective FLOP gap of **1 OOM**.[60]
        - *Use a smaller IQ gap.* If we used IQ 70 → IQ 125 then the effective FLOP gap would halve to **1 OOM**.
        - Doing both of the above reduces the effective FLOP gap to **0.5 OOMs**!

    i. Overall, I take the estimate here to be **~1 OOM**.

    j. Again, a *very significant* counterpoint is that early AIs will have strong comparative advantages at some tasks over others. E.g. someone with IQ 45 can't multiply 7 digits numbers in their heads, and yet we have calculators.

    k. Overall, I see this as lending some plausibility to gaps as low as 0.5 OOMs as well as weakly suggesting a best-guess of **~2-3 OOMs**.

- **Practical barriers to partially automating tasks.**
    a. A high-level task like "present on topic X" might have many subtasks like "learn about X", "plan the presentation", "write the presentation", "check for errors", "deliver the presentation". These subtasks themselves have many further subtasks ("search for relevant articles", "extract relevant information from articles"), and so on.
    b. Before AGI, a lot of AI's economic impact will probably come from partially automating high-level tasks; i.e. from automating some subtasks but not others.[61]
    c. But partial automation may be difficult in practice, e.g. if it involves integrating AIs in complex workflows. For example, suppose an AI can write a presentation from a suitably formatted plan. Using this AI to partially automate "present on topic X" would require somehow creating a plan in a format suitable for the AI.
    d. This could mean that, in practice, it is only when AI is close to being able to *fully* automate a high-level task that it does significant amounts of partial automation.
    e. In addition, it may be the case that AI is able to fully automate most high-level tasks at about the same time because they require similar capabilities (or similar subtasks).
    f. Combining the above two bullets, we may automate most high-level tasks at about the same time and only achieve significant partial automation of high-level tasks shortly before full automation.[62] This implies there might only be a short gap

[60] This implicitly assumes human lifetime-learning scaling via increasing brain size is as good as ML scaling by increasing training FLOP. Lifetime learning might be *better* because the life-learning algorithm scales better than ML training algorithms; it might be *worse* because ML algorithms can scale the amount of data optimally with training FLOP, unlike lifetime-learning.

[61] AI may also allow us to restructure the high-level task entirely so the necessary subtasks change, and automate some of the new subtasks. It may also allow us to do new kinds of high level tasks.

[62] One concrete way to think about this is as follows. Suppose there are 100 tasks on the lowest level. Each high-level task requires 90 of them, so there's lots of overlap between high level tasks. But it's only possible to partially automate a high-level task once 80 out of its 90 lower level tasks can be automated, due to practical difficulties with partial automation. Now imagine low-level tasks are automated one by one in random order. In this toy model, all the high-level tasks will be automated at a similar time (when we're

from “begin to significantly automate some high-level tasks” to “fully automate ~all high level tasks”.

g. This all pushes towards a smaller effective FLOP gap, somewhat smaller and perhaps significantly smaller than we’d have otherwise thought.

h. This effect is *significantly increased* by the fact that my definition of “AI can readily perform task X” is “it would be profitable for organisations to do the engineering and workflow adjustments necessary for AI to perform task X in practice and they could make these adjustments within 1 year if they prioritised them”.

    - If it takes many decades to cross the effective FLOP gap, there would be much more time to rearrange workflows to allow for partial automation. However, the numbers I’m getting out of the full takeoff speeds framework suggest we’d cross even a large effective FLOP gap in <20 years.

This table summarises my very tentative takeaways from each factor.

Factor	**How it informs the effective FLOP gap**	**My tentative takeaway for the effective FLOP gap**	**How much relative weight I place on each consideration (1-5)**
AGI training requirements	Constrains the endpoint	Low AGI training requirements bound the effective FLOP gap from above.	5
SOTA AI capabilities	Constrains the startpoint		3
Horizon length	Directly informs choice of effective FLOP gap	>5 OOMs	2 (more if training requirements are large)
How AI capabilities vary with training FLOP between different domains	Directly informs choice of effective FLOP gap.	>3 OOMs ~5 OOMs (best guess) ~8 OOMs is plausible	4
How AI capabilities vary with training FLOP within a domain	Directly informs choice of effective FLOP gap.	>4 OOMs ~5 OOMs	3
How animal capabilities vary with brain size	Directly informs choice of effective FLOP gap	~1 OOM is plausible ~3 OOM best-guess	2
How human capabilities vary with	Directly informs choice of effective	~0.5 OOM is plausible ~2-3 OOM best-guess	3

close automating all 100 low-level tasks) and each partial automation of a high-level task only happens shortly before full automation.

brain size	FLOP gap		
Practical barriers to partially automating tasks	Argument for a small effective FLOP gap	Effective FLOP gap smaller	4

Overall, before taking into account AGI training requirements my best guess for the effective FLOP gap is ~4 OOMs,[63] but I wouldn't be surprised by 1 OOMs or 8 OOMs.

If I condition on low AGI training requirements of 1e30 2020-FLOP the first two factors bite hard and my best guess is ~2-3 OOMs; if I condition on large AGI training requirements of >=1e38 2020-FLOP the "horizon length" factor and "How AI capabilities vary with training FLOP between different domains" comes into play more and my best guess is ~5-6.

What bio-anchors says about speed crossing effective FLOP gap

The effective FLOP gap is measured in 2020-FLOP. The bio anchors report projects its three components over time. In Ajeya's best guess sheet their growth rates are as follows:

1. $ on FLOP for a training run initially doubles every 2.5 years (growth rate 28%) until it reaches 1% of US GDP. Then its growth rate is 3%.
2. FLOP/$ doubles every 2.5 years (growth rate 28%) until it reaches a maximum.[64]
3. 2020-FLOP per FLOP doubles every 2-3 years (growth rate ~28%) until it reaches a maximum.[65]

This implies that g(2020-FLOP in the largest training run) is initially ~84%[66] and then slows down to ~59%[67]. This corresponds to 0.36 OOMs per year initially followed by 0.26 OOMs per year. At the faster pace, it would take 11 years[68] to cross an effective FLOP gap of 4 OOMs; at the slower pace it would take 15 years.[69]

Summing up and looking ahead

I've introduced a first-pass framework for calculating some metrics of takeoff speed. At the moment, it can calculate the calendar time between training some weaker AI and some stronger AI; it cannot calculate metrics relating to the *number* of AIs or their effects on GDP.

[63] One thing informing this is calculating a weighted average using an adjusted version of the above table.
[64] The growth rate slows somewhat when the maximum is near.
[65] The precise doubling time depends on the biological anchor. For medium and long horizon anchors it is 2 years, for short horizon anchors it is 3 years.
[66] 28 * 3 = 84
[67] 3 + 28 + 28 = 59
[68] 4/0.36 = 11
[69] 4/0.26 = 15

In this framework, the takeoff speed depends on the size of the effective FLOP gap and our speed crossing it. Our speed crossing it is given by the equation g(2020-FLOP) = g($ on FLOP) + g(FLOP/$) + g(2020-FLOP per FLOP).

The next few sections extend this basic model. Section 4 estimates g(2020-FLOP) in the run-up to AGI, based on the effect of fast rising AI investments. Section 5 models the effect of incremental AI automation in the run-up to AGI. The infrastructure introduced here allows us to calculate metrics based on the number of AIs and on their effects on GDP. Section 6 discusses bottlenecks. Later sections perform sensitivity analyses, discuss its many limitations, and come to an all-things-considered bottom line.

4. Rising AI investments

I would treat this section as providing detailed parameter estimates for important inputs to the Full Takeoff Model. In particular, it estimates the pace at which human investments on the largest AI training run, hardware R&D and software R&D will grow. It also estimates the returns to hardware and software R&D. It then infers how fast takeoff will be from human investment alone, essentially calibrating this toy model.

Summary

I believe pre-AGI systems have the potential to increase GDP by $10s trillions and probably $100s trillions *per year*. Given this, and the fact that AI investments are currently in the $10s billions (sources below), I expect human investments in AI to grow very rapidly after key actors "wake up" to the potential economic and strategic[70] value of AI. While governments and chip manufacturers and investors are aware that AI is strategically important, I claim their actions implicitly significantly underestimate its transformative potential.

This section analyses the effect of this fast-rising investment on the three components of 2020-FLOP.

The numbers in this section are best characterised as "wild guesses informed by weakly relevant empirical data". As such, the uncertainties are very high. That said, here are my tentative central estimates.[71] After "wake up" I guess that:

- **$ on FLOP in the largest training run** will initially grow at ~97%, and then later at ~22%
    - **$ on FLOP globally** will grow at ~22%, based on how quickly I guess we'll be able to expand chip production after "wake up".

[70] e.g. via military and security applications.

[71] For now I'm leaving these all at 2 significant figures so I don't lose information about the centres of my subjective distributions, but 1 significant figure would be much more appropriate given the magnitude of the uncertainties involved.

    - **% global FLOP on the largest training run** will grow at ~75%, and have 1 - 3 OOMs room to grow in total.
- **FLOP/$** will eventually grow at up to ~88% (recently ~25%), driven by real $ hardware investment growing at ~17% (recently 5%). But it will take many years before it grows this quickly, gradually accelerating from its recent pace of doubling every ~3 years.
- **2020-FLOP per FLOP** will grow at ~31%, driven by real $ software investment growing at ~25% (recently ~20%).

This section often refers to growth rates. If you prefer thinking in terms of doubling times, use this converter.

Background - "waking up" to advanced AI's economic potential

I believe AGI would drive explosive economic growth - Gross World Product (GWP) growing at >= 30%/year. I think less powerful AI could drive multiple doublings of GWP. Nominal GWP is $85tr, and so doubling GWP even once involves adding ~$80tr to the global economy (starting from now). In other words, on my view, pre-AGI systems have the potential to generate many $10s or $100s of trillions *per year*.

Annual investments into hardware and software for AI development are 2-4 OOMs smaller:

- *All hardware.* Total semiconductor industry revenues are $550 billion; semiconductor R&D is ~$70 billion and semiconductor capex was ~$130 billion.
- *AI hardware.* The size of the AI chip market is probably ~$20 billion;[72] AI targeted chip R&D is maybe a few billion $.[73]
- *AI software.* I'd guess annual spending on software workers for SOTA AI is $10-100 billion.[74]

I haven't dug into these numbers and this is important further work.

I believe that before AGI is developed, many key actors[75] will "wake up" to the potential for advanced AI to generate $10s trillions per year. At the very latest, this will happen when pre-AGI systems *actually produce* this much value; probably it will happen much earlier, as the potential to automate large swathes of cognitive labour becomes apparent from impressive demos.

Once this "wake up" occurs, I expect investments in AI to scale up as quickly as possible until they are worth $ trillions per year.

[72] See footnote 48 of this CSET report; also informed by an unpublished memo by a CSET researcher.
[73] NVIDIA R&D in 2021 was ~$4 billion, and they're estimated to be 80% of the market share for AI chips. I'm not sure how much of the $4 billion was spent on R&D for AI chips vs other R&D.
[74] DeepMind annual spending is ~$1b. I'd guess total AI spend is 10 - 100X this.
[75] Governments of powerful nations, leaders of large tech companies, militaries.

This section analyses the consequences of this fast-rising investment for the growth of 2020-FLOP used in the largest training run, g(2020-FLOP). I consider each of its three components separately: $ on FLOP, FLOP/$, and 2020-FLOP per FLOP. First, though, two sections about how I'm thinking about the dynamics of investment ramp up once the "wake up" has occurred.

In what follows, I'll assume that we have had "wake up" by the time we start crossing the effective FLOP gap. This need not be true. But for my startpoint of "automate 20% of cognitive tasks", I think it's very likely. Automating 20% of cognitive tasks, on a naive calculation, would increase GDP by 25% which is ~$20 trillions.[76]

$ on FLOP for the largest training run

I break this component down into two subcomponents:

$ on FLOP for the largest training run = $ on FLOP globally * fraction of global FLOP on the largest training run

Let's discuss each in turn.

$ on FLOP globally

I believe that, after "wake up", the total amount of global FLOP will be bottlenecked by how quickly we're able to design better chips and manufacture them rather than by willingness to pay.[77]

For this reason, I'm thinking primarily in terms of "how quickly can the world overcome production bottlenecks to produce more FLOP" rather than "how quickly will people increase their $ spending on FLOP". In line with this, and as a part of a toy simplification discussed in this appendix, I'm interpreting "$ on FLOP globally" as meaning something like "number of chips globally". Analogously, I'm interpreting FLOP/$ as "FLOP per chip". This means, among other things, that I'm ignoring the fact that after "wake up" rising demand will increase the price of chips of a fixed quality.[78]

[76] If the only effect [of automating 20% of cognitive tasks] is to concentrate human workers on the remaining 80% of tasks, you'll have 25% more workers per task on that remaining 80%. (100/80 = 1.25.) So the effective labour supply is 25% bigger; as a result you'll accumulate 25% more capital and so GDP will increase 25%. (Here I make the standard assumption there are constant returns to labour and capital in combination.) This bottom line is too high in ignoring the non-cognitive labour that is not automated; this might reduce it by 2X. However, it is too low in ignoring the benefits of improving performance and throughput on the automated cognitive tasks, which could increase it by 2X.

[77] I say more about this in an appendix.

[78] See more.

How fast will the world be able to ramp up production of chips, when it's doing so (roughly speaking) as quickly as possible?[79] This is a thorny empirical question, which deserves much more attention than I've given it. This table summarises some weakly relevant quantities:

Quantity	**Growth rate**[80]	**Comment on relevance to g($ on on FLOP globally) after "wake up"**
Semiconductor revenue growth.	6 - 18%	Demand will be much higher after "wake up".
TSMC revenue growth.	14 - 29%	Demand will be much higher after "wake up". TSMC can potentially steal experts from rival companies.
Growth of munitions production at war time.	20 - 45%	Semiconductor supply chain is much more complex than munitions.
Time to build a fab. (Assumes that this equals the doubling time in the number of fabs.)	14 - 35%	My assumption about doubling times seems aggressive.

In a little more detail:

- **Semiconductor revenue growth.**
  - This grew at a rate of 6% from 2012-20 and a rate of 18% from 2018-20.
  - The higher rate might be a better indicator of the maximum capacity for growth, and thus of the growth after "wake up". On the other hand, it could reflect a temporary effect or a spike in demand (that wasn't reflected in an increase in supply).
  - My takeaway is a very weak pull to numbers in the ballpark 5%-20%.
- **TSMC revenue growth.**
  - Semiconductor revenue growth is constrained by demand. This is less true for an individual corporation like TSMC that can take market share from its rivals by growing its own output. So TSMC's growth may mirror growth after "wake up" in that neither is constrained by demand.

[79] To simplify the main text, I elide the distinction between the *annual production* of FLOP, and the *total stock* of FLOP. Mathematically, it turns out that if the former grows at a constant rate, so does the latter. So here I estimate the growth rate of the former and use this as a proxy for the latter. However, this can lead us astray: if the growth rate of the former increases, there is a lag before the growth rate of the latter increases in step. So this distinction *is* included in the Full Takeoff Model (FTM), so is incorporated in the results of the sensitivity analysis in section 8. The effect is to make takeoff slightly slower.

[80] Reminder: the growth rates here and throughout this report are *instantaneous growth rates*, not per-year growth figures. For example, if the instantaneous growth rate is 40% then the increase in one year is e^0.4 = 1.49X, which corresponds to a per-year growth of 49%.

  - This grew at 14% from 2011-21 and 29% from 2019-21.
  - Its capital expenditures will grow at 35% from 2018-2022. This suggests the fast revenue growth is not mainly “more demand leads to higher prices for the same amount of real output”.
  - On the other hand its number of employees only grew at 8% from 2009-2020 and at the same rate from 2018-2020. I’d be surprised if you could sustain 30% growth in output for very long without growing your employees faster than this.
  - TSMC is potentially able to hire talent from other companies to help it scale, which may make these numbers too high. On the other hand, after “wake up” there will be (much) more willingness to pay and so I expect expansion efforts to be more aggressive even than TSMC’s recent expansion.
  - My takeaway is a **weak pull towards numbers in the ballpark 15-30%**. I find this more informative than the semiconductor numbers.
- **Growth of munitions production at war time.**
  - This is an example of “growth of a specific industry’s output when there is very large demand”.
  - I looked at munitions output for countries involved in WW2. Growth rates were mostly between 20% and 45%, with the US as high as 80% (though from a much lower base as a fraction of their GDP).
  - I’d expect it to be harder to grow semiconductor output because its supply chain is notoriously complicated, probably much more so than munitions. E.g. this growth involved a lot of refitting existing factories to make munitions, which won’t be possible for cutting edge computer chips.
  - My takeaway is a **weak pull towards numbers in the 20 - 30% range**.
- **Time to build a fab.**
  - From a quick google, estimates vary from 2 - 5 years.[81]
  - If this is also the *doubling time* for the number of fabs after “wake up”,[82] that implies a growth rate of 14% - 35%. The assumption that the doubling time for fabs is the time it takes to build them feels aggressive to me.
  - My takeaway is a **weak pull towards numbers in the 10 - 25% range**.
- **Growth of the AI chip market.** It’s growing at a rate of **~30%**. Naively, you’d expect growth to slow after it becomes a majority of semiconductors, but maybe the demand after “wake up” will allow it to continue at its current pace. Not convincing to me as its current growth is probably enabled by displacing production of other chips rather than creating additional production capacity.

I find the first quantity least informative, and the last three quantities similarly (un)informative as each other. My best-guess central estimate here is a growth rate (for $ on FLOP globally) of **~22%**.[83] I’d be surprised if the true figure was <10% or >40%. <10%, as well as falling outside

[81] This website says >2 years; this one says 3-5 years; this discusses an example that took 2.5 years.
[82] The toy model here is that we’re able to leverage the expertise of each existing fab to build another one, and meanwhile train people up to work there.
[83] Here’s how I arrived at this number. For the last three quantities my takeaway was a weak pull towards the ranges 15-30%, 20-30%, and 10-25%. Taking the average of the midpoint of each range gives (22.5 + 25 + 17.5)/3 = 22.

the range of the last three quantities, just feels very low for a world that's trying its best to expand chip production. >40% implies a production doubling time of <21 months, which seems very quick given the staggering complexity of the production process and the requirement for workers with specialized skills.

When will this growth in $ on FLOP globally top out? The Full Takeoff Speeds Model currently assumes it continues until we're spending 10% GWP annually on FLOP, ~$10 trillion today. Why 30%? Currently ~60% of GDP is paid to human wages; when AI plays a similar role to human brains we might pay a comparable % of GDP to rent chips to run AIs on. (I reduce the 60% to 10% to account for other costs of running AI systems and the possibility that we pay less for running AIs than we currently pay to human wages, e.g. due to being bottlenecked by non-cognitive inputs to production.)[84]

Ok, I've explained how I'm thinking about the first component of $ on FLOP for the largest training run; now let's discuss the second component.

Fraction of global FLOP on the largest training run

In my mind there are two important sub-questions here. First, how much room will there be in total to scale up the fraction of global FLOP used on the largest training run? This determines how many OOMs of the effective FLOP gap we can cross without even increasing the number of chips in the world.

Second, how quickly will we be able to scale up in this way? This determines how quickly we'll cover those OOMs.

If there's many OOMs here, and we can cover them quickly, this could drive a very fast takeoff.

How much room to scale up the fraction of global FLOP used on a training run?

Drew Lohn from CSET has contracted with us and spent ~10 hours estimating the amount of FLOP currently available globally. Measured in terms of the FLOP we could perform by running the chips non-stop for a year for perfect utilisation, he estimates **~2e28 FLOP from discrete GPUs,[85] ~4e28 FLOP from discrete+integrated GPUs, and ~2e29 if you include the large numbers of lower performance CPUs** used in (e.g.) phones.[86] In my view, these numbers are very uncertain and could easily be wrong by an OOM.

[84] In practice, in the Full Takeoff Model we typically get AGI before spending 10% GWP on FLOP.
[85] As opposed to *integrated GPUs,* which are incorporated as part of a device like a laptop.
[86] Drew Lohn actually estimated FLOP/s from the chips produced annually. I obtained these numbers by assuming the global stock of chips is 2X annual production.

I think a realistic maximum for the fraction of global FLOP that could be used in the largest training run is probably **~10%**.[87] That could involve, for example, using 30% of the world's compute in a training run that lasts for 4 months. This already feels somewhat aggressive, but there will be economic incentives to combine compute together in big training runs and online learning will be a significant factor. Using 10%, the realistic maximum for FLOP used in the largest training run is currently 2e27 - 2e28 FLOP.

I believe the largest publicly available training run as of July 2022 is ~2e24,[88] implying that the room for scale up is currently ~**3 - 4 OOMs**.

Of course, this quantity will change over time. I think the room for scale up will reduce over the next decade. Here's an extremely rough estimate. Extrapolations of growth of the AI chip market imply growth of 15X by 2030; but the total quantity of global FLOP will grow by less, let's say ~5X.[89] Meanwhile, I expect the $ spent on the largest training run to increase by ~400X in that time.[90] This implies that the room to scale will fall by ~2 OOMs to **~1 - 2 OOMs** by 2030.[91] For now, the Full Takeoff Speeds Model is assuming that this won't change after 2030 until "wake up".[92]

For our purposes – estimating the time it will take to cross the effective FLOP gap – what matters is how much room for scale up will remain once we reach the startpoint. If we reach the startpoint around or after 2030, I expect ~1 - 2 OOMs room for scale. If we reach the startpoint before 2030, I expect somewhere between that and the current room for scale up (~3 - 4 OOMs); I'll call it ~1 - 3 OOMs.

When will we reach the startpoint? The startpoint is measured in 2020-FLOP used for a training run. A $1b training run in 2030 would use ~2e29 FLOP, based on the bio-anchors extrapolations of FLOP/$ and 2020-FLOP per FLOP.[93] So people with startpoints lower than 2e29 2020-FLOP should expect us to reach the startpoint before 2030.

Startpoint, 2020-FLOP	Reach before 2030?	Room left for scale up when we reach the startpoint

[87] This, together with my earlier assumption that we might ultimately spend 30% GWP on FLOP, implies that we'd spend 3% GWP on FLOP for the largest training run. This is a lot; Bio Anchors caps this at 0.25%. But remember that an AGI training run will be happening in a world where AI systems are already generating $10s trillion per year.

[88] I believe these were the requirements for PaLM.

[89] The 5X is for $ on "discrete GPUs", which can still grow significantly as a fraction of fab production. If we used the broader category of all GPUs and CPUs, I'd expect lower growth.

[90] It's estimated that PaLM cost ~$10m to train. My median guess is that the largest 2030 spend is ~$4b, in line with Bio Anchors' projections. That's a 400X increase. (I believe AI companies can buy FLOP ~3x cheaper than if I were to rent GPUs on the cloud, so in practice it may cost them more like ~$1b.)

[91] In the FTM the median room for scale in 2030 is ~1 OOM because it uses the "discrete GPU" category as its median.

[92] I.e. it assumes that after 2030 training runs will scale up at the same pace as global FLOP until "wake up".

[93] See calc.

<2e29 2020-FLOP	Yes	1 - 3 OOMs
>2e29 2020-FLOP	No	1 - 2 OOMs

How quickly will we increase the fraction of FLOP used on a training run?

A naive approach here is to anchor off the scale up over the last decade. I roughly estimate that, in 2012-2018, the fraction of AI chips used in a training run grew at a rate of ~150%.[94] That's an OOM every 18 months.[95] If we scale similarly fast after "wake up", we'll reach the maximum in just a few years.

However, I believe that the engineering barriers to further scaling will be much more significant than in the past, e.g. in building the infrastructure for training to be efficiently distributed over many chips in parallel. I don't know how long these problems will take to overcome; this is an area where further empirical research is needed.

Another big uncertainty is how hard it will be to adjust manufacturing processes to produce AI specialised chips rather than other chips. I'd guess that this will be easier for discrete GPUs than for CPUs.

My current wild guess is that after "wake up" enough effort will go towards this that we will scale up at a pace of an OOM every ~2 years. That amounts to a **growth rate in the fraction of chips on the largest training run of ~110%**.[96] But it's conceivable to me that we will do 2 OOMs in one year with sufficient effort (~450%), and conceivable that we can only do an OOM every 4 years (~55%).

The strategic importance of scale up

This dynamic of scale up could drive an *extremely* fast takeoff. If the effective FLOP gap is small (<=2 OOMs), room for scale up is large (>=2 OOMs) and scale up happens quickly, we may cross the effective FLOP gap *extremely* quickly. Concretely, this might look like one big actor making a deal with TSMC or NVIDIA to buy the majority of their output for a year, using this to 100X the largest training run so far, and thereby blasting through the OOMs for FLOP around which AI capabilities are most concentrated.

There's a counter-intuitive implication here. One sure-fire way to avoid this scenario is to scale up training runs *before* we reach the startpoint (before AIs become really capable). That way there's no room left to scale when we hit the startpoint. Of course, there are considerations

[94] In 2012 - 2018 training FLOP increased by 300,000X, corresponding to a growth rate of 210%. exp(2.1*6) = ~300,000. To get the growth in the *fraction* of FLOP we need to subtract out the growth of total FLOP. Let's subtract 30% for growth of FLOP/$ and 30% from growth in $ on FLOP, leaving 150%. [Do this again with Jaime's numbers for the longer period?]

[95] e^(150%*1.5) = 9.5.

[96] e^(1.1*2) = 9.

which argue in the opposite direction,[97] so this doesn't imply that we should in fact scale up training runs today.

Summing up $ on FLOP for a training run

There are two sources of growth for this quantity: increasing the **$ on FLOP globally[98]** and increasing the **fraction of FLOP on a training run**. Mathematically:
g($ on FLOP for a training run) = g($ on FLOP globally) + g(fraction of FLOP on a training run)

My very rough best guesses are:

- g($ on FLOP globally) = ~22%, probably between ~10% and ~40%.
- g(fraction of FLOP on a training run) = ~75%, with room for 1 - 4 OOMs of growth here in total (more likely towards the lower end).
- So g($ on FLOP for a training run) is initially ~97% and later ~22%.

This concludes the discussion of how rising AI investment after "wake up" might affect $ on FLOP;[99] the next section analyses the same question for FLOP/$.

FLOP/$

How might rising AI investments after "wake up" affect FLOP/$?[100] The most salient mechanism is that they increase R&D efforts towards making better chips and increasing FLOP/$. In 2000 - 2020, real $ inputs to hardware R&D have grown slowly at ~4% a year. Faster input growth after "wake up" should accelerate growth of FLOP/$.

How can we estimate the size of this effect? My approach is to:

1. Use historical data to fit an economic model relating hardware R&D inputs to increases in FLOP/$. The fitted model implies that **each doubling of cumulative inputs leads to 5.2 doublings in FLOP/$.**
2. Forecast how quickly inputs will grow after "wake up".
   a. I guess that annual R&D inputs might grow at a rate of **17%.**
   b. If *cumulative* inputs grew at 17%, the model implies that FLOP/$ will grow at 5.2 * 17% = **88%**, ~9 month doubling.
   c. But, importantly, I'm currently forecasting that cumulative inputs will initially grow at their current pace of ~4%, and that their growth rate will gradually increase to

[97] For example, scaling up training runs today might cause "wake up" and thus bring AI timelines forward by many years, reducing time for safety work, movement building, and other preparations.
[98] As mentioned above, I'm excluding the effect of "bidding up the price of compute".
[99] I compare my $ on FLOP forecasts with those from Bio Anchors in this appendix.
[100] Remember I'm ignoring the possibility that compute prices are significantly bid up, and using a simplification in which all increases in FLOP/$ are due to hardware R&D (and all increases in $ on FLOP and due to scaling up production of chips).

17%. This implies growth in FLOP/$ will gradually increase from its recent rate of doubling every ~3 years.

I'll discuss each step in turn.

Economic model relating hardware R&D inputs to outputs

I'm using what economists typically consider to be the best model of R&D - the semi-endogenous growth model (SEG).[101] Its distinctive advantage, compared to the alternatives, is that it quantifies the extent to which *ideas are getting harder to find*, or the extent to which there are *diminishing returns to doing more research.*[102]

One way to understand the core of the model is as saying:[103]

Each time cumulative inputs double, the output metric doubles r times

So for each x% increase in cumulative R&D inputs, the output metric will increase by r*x%.[104] The inputs could be measured in $ or in (quality-adjusted) researcher-years; the output metric can similarly vary.

Estimating *r* for hardware

What happens when we fit a SEG model to this data on hardware inputs and outputs? I did this, taking the inputs to be real $ invested in semiconductor R&D[105] and the output to be measured FLOP/$.[106]

[101] To my mind most of the alternatives are pretty implausible, so being "the best" means something like "being broadly consistent with the empirical evidence and not having significant noticeable flaws" rather than being well confirmed.

[102] As such, it's the model used in the famous paper by that name.

[103] How does this relate to the standard presentation of the theory in terms of the stepping on toes parameter lambda and the fishing out parameter phi? r = 1 / (1 - phi). If lambda < 1 then we should adjust our description of what the core of the model to '*For each x% increase in cumulative **effective** R&D inputs, the output metric will increase by r*x%*', and add that the **effective** input in each year is input^lambda.

[104] Then in what sense are "ideas are getting harder to find"? Somewhat confusingly, economists define "one idea" as a 1% increase in the output metric. Early R&D models implied that every such increase would require the same absolute increase in cumulative inputs. But this model implies that the cumulative input increase required grows over time. The smaller *r*, the more effort must increase from one idea to the next. More precisely, compare the effort needed to double the output metric on two consecutive occasions (e.g. increasing it from 5 to 10, and from 10 to 20). The effort for the second doubling is greater by a factor of 2^(1/r).

[105] I got my dollar input data from *Are ideas getting harder to find*. I took their numbers for nominal $ hardware investment, using their category 'PatentNarrow (include equipment)'. I extend their data after 2015 based on recent growth in R&D inputs (rows 41 - 46). Then I adjust these nominal $ inputs for inflation. See calcs here.

[106] I got my FLOP/$ from the Bio Anchors appendix, top of p.29. I read off FLOP/$ values for 1970, 2008 and 2018, see rightmost columns here. (Note, this data doubles-up as data on the growth of FLOP/s per

The following table summarises the calculation:

Time period	**g(cumulative inputs)**	**g(FLOP/$)**	**Estimated hardware returns, *r***
1970 - 2018	7.9%	53%	6.7
1970 - 2008	8.5%	61%	7.1
2008 - 2018	5.4%	23%	4.3
2006 - 2022 (GPUs)	5.4%	28%	5.2

If you prefer thinking in terms of doubling times rather than growth rates, use this converter.

The calculation is simple. For each time period, *r* = g(FLOP/$) / g(cumulative inputs).

So the data suggest that returns to hardware R&D were very good from 1970 - 2008, with each doubling of cumulative inputs leading to **7** doublings for FLOP/$. Returns have been less good recently, with each doubling of inputs driving a little over **4** doublings of FLOP/$. Though if you focus on GPUs, returns look a little better.

I'll take the GPU figure, 5.2 as my median estimate of the current value of *r*. It could be higher if there's a reversion to the longer-run historical returns, or it could be lower if returns are worse today than in the period 2008-2018. Additionally, the Full Takeoff Speeds Model assumes that *r* decreases towards 0 as FLOP/$ approaches physical limits.

(There is a subtlety in accounting for the *stepping on toes* effect, where doubling the real $ investment in a given year less than doubles the progress that year due to barriers to parallelising research. I discuss this in an appendix.)

How quickly might hardware R&D inputs grow after "wake up"?

How quickly could we grow real $ inputs to R&D when we're trying very hard after "wake up"?[107] This is another input where the best I have been able to do is point to weakly relevant empirical trends. Here are some:[108]

- **Growth of US federal R&D around WW2.**
    - The US made a notable effort to expand R&D during and after WW2, so may be an indicator of how quickly R&D efforts can be expanded.

$, as the appendix applies a constant conversion factor from FLOP/s to FLOP, assuming chips are deprecated after 2 years). For GPU FLOP/$, I got data from Epoch's analysis.

[107] I'm explicitly *not* including the $ value of AI systems used to do hardware R&D in this section. That is, I'm forecasting the growth in the real inputs of human labour and non-AI capital. The next section gives a separate treatment of the effect of incremental AI automation on the pace of R&D progress.

[108] All figures give the growth of real $ inputs unless stated otherwise in a footnote.

  - Examples:
    - Defense R&D grew at **15%** from 1950 to 1962.
    - Civilian R&D (which includes many sectors) grew at **17%** from 1950 to 1966, though probably *total* R&D grew more slowly than this.
    - Space R&D grew at **36%** from 1954 to 1966, though from a small base.
    - Total federal R&D grew at **11%** from 1937 - 1953 (I think this number is sketchy).
  - I think the financial incentive for hardware R&D will be greater after “wake up”, but it will be growing from a higher base. My takeaway is a weak anchor to 10-30%.
- **Historical growth of semiconductor R&D inputs.**
  - These grew at **10%** initially, and this growth rate has gradually declined over time (see above).
  - Again, the incentive to grow R&D will be much greater after “wake up”, but it will be growing from a much larger base. I’m not sure how these net out.
- **Historical growth rate for other areas of R&D.**
  - *Are Ideas Getting Harder to Find* contains estimates of the growth of inputs for a number of fields; numbers range from **2%** to **10%**.[109]
  - I’d expect hardware inputs to grow at or above the higher end of this after “wake up”.
- **Recent growth of R&D in hardware companies whose revenues are growing quickly.**
  - These provide evidence about how quickly hardware companies grow R&D when there is lots of demand for their output.
  - Examples:
    - ASML R&D grew at **17%** from 2016 - 2021.
    - NVIDIA R&D grew at **19%** from 2016 - 2021, and **14%** from 2005 - 2021.
    - TSMC R&D is growing at **4 - 5%**. (Their capital costs are growing much more quickly than this.)
  - My takeaway is a weak anchor to ~20%, as the demand will be higher still after “wake up”.
- **Growth of R&D relating to covid-19.**
  - From January 2020 to June 2020, the number papers published related to covid grew by almost 2 OOMs.[110]
  - While I doubt the new authors were adding nearly as much value as existing experts, and the initial base is clearly much smaller for covid, this was an update for me towards scientists’ ability and willingness to pivot to new fields.
- [Any other ways ppl think of to inform a guess at this?]

[109] The growth rate of real $ are probably slightly higher, as the paper uses a different measure of inputs. Its units are “salary of a high-skilled worker”, and I think a “high-skilled worker” is operationalised as a graduate. So if graduate salaries have gone up in real terms, the paper’s growth rates will **underestimate** growth as measured in real $.

[110] See the “Sell it” section of this Matt Clancy substack post.

My best-guess central estimate here is an input doubling time of ~3.5 years, which is a **growth rate of ~17%**. This is a little higher than the trends above, but I expect there to be huge demand once $ trillion training runs are on the table,[111] and sufficient numbers of high-skill people who can add value (e.g. who currently work in finance, physics, other areas of materials science) for the field to double every 3 - 4 years. I'd be surprised if this <8% as 8% is not unusual for the growth of R&D fields. Similarly I'd be surprised by >35% because i) even space R&D did not grow faster than this from a small base in the 1950s, and ii) doubling the semiconductor R&D field in just 2 years (35% growth) feels like a pretty tall order.

When will this growth top out? The Full Takeoff Speeds Model currently assumes it continues until we're investing 3% GWP annually, ~$2.5 trillion today. That's ~35X larger than today's figure of $70b, allowing for ~20 years of 17% growth.

Why a ceiling of $2.5 trillion? One anchor is that annual semiconductor revenues today are ~$550b while R&D is ~$70b: a ratio of **six**. If annual semiconductor revenues are growing towards being worth $30tr, as I assume above, then that same ratio implies that R&D should be growing towards $5tr. I'm reducing that to $2.5trillion because total global R&D spend today is only ~$2 trillion.

Growth of FLOP/$ after "wake up"

If cumulative inputs grow at 17%, and *r* = 5.2, FLOP/$ will grow at 5.2 * 17% = **88%.** That's roughly a 9-month doubling.

There is an important caveat. There is a difference between *annual* and *cumulative* inputs. If annual inputs suddenly start growing at 17% (rather than their current ~5%), there is a lag before *cumulative* inputs grow at the new faster rate.[112] So **if faster growth of annual R&D inputs coincides with the startpoint, there will be a >10 year lag before FLOP/$ grows at the new quicker rate**.

This dynamic – the distinction between annual and cumulative inputs – is modelled explicitly in the Full Takeoff Model (FTM).

I worry the FTM is conservative for modelling the response to "wake up" merely via a faster growth rate in annual hardware R&D spending, rather than also including a one-time jump in spending. A one-time jump would reduce the lag before FLOP/$ grows at the new quicker rate, and could significantly reduce the time crossing the effective FLOP gap.

[111] If you're going to spend $1 trillion on FLOP for a training run, it's worth spending $500 billion on R&D to double FLOP/$. Current semiconductor R&D is only $70 billion.

[112] In fact, the growth rate of cumulative inputs gradually increases from the old growth rate of annual inputs to the new growth rate of annual inputs. Also, it turns out a larger "stepping on toes" effect causes a longer lag before output grows at the new pace, see more here.

This concludes the discussion of how AI investments after “wake up” might affect trends in FLOP/$. Now I turn to how they might affect 2020-FLOP per FLOP.

How big is the relevant bucket of hardware R&D?

The Full Takeoff Model (FTM) uses ~$80b as the hardware R&D spend in 2022, the figure for semiconductor R&D spend.

But you might choose to use a larger figure if you include the potentially-larger buckets of computing and electronics or materials science.[113] The logic for inclusion would be that, in the long run, semiconductor R&D progress is reliant in progress in these broader areas. Using a This would leave less room for investment to hardware R&D to grow before reaching a cap; it might also change the historical growth rates of R&D spending.

Alternatively, you might choose to use a *smaller* figure if you restrict to hardware R&D specifically targeted at improving chip design for AI use-cases. This would leave more room for R&D spending to grow, and probably imply recent historical growth of R&D spending was higher.

In reality, the smaller bucket is probably more relevant over shorter timescales (where existing node sizes can be specialised for AI algorithms) but the larger bucket will become increasingly relevant over longer time periods (where entirely new computing paradigms must be invented). This means the FTM might *underestimate* R&D progress in the short term but *overestimate* progress in the long term.

This model implies that hardware progress will continue to slow down before “wake up”

The above data show that growth of FLOP/$ has slowed over time. The semi endogenous growth model I’m using predicts *some but not all* of that slowdown, because the growth of cumulative hardware inputs has also slowed over time (but not by as much). This gives us some reason to prefer it to a simple trend extrapolation.

But, in the near term, it seems like the growth of cumulative R&D hardware inputs will continue to slow.[114] So the semi endogenous growth model predicts that the growth of FLOP/$ will continue to slow during the 2020s. In fact, it predicts its *current* growth rate is lower than its

[113] This chart puts Computing and Electronics at ~20% of global R&D spend, which would be 0.2*$2tr = $400b.

[114] Why? Historically, *annual* R&D inputs grew quickly at a rate of ~10%. Recently, they’ve grown slower, at a rate of ~4%. As a result, *cumulative* R&D inputs have gradually been growing more slowly over time, with their growth rate moving gradually down from 10% towards 4%, currently at ~6%. If annual R&D inputs continue to grow at ~5% (as I’m forecasting before “wake up”), then the growth of cumulative R&D inputs will continue to slow from 5% to 4%.

recent average (i.e. the averages reported in the table above) because the *current* growth of cumulative inputs is lower than the recent average growth rate.

If I had used a smaller bucket of R&D, restricted to designing AI-specialised chips, this model might have the opposite conclusion. Plausibly, the cumulative hardware inputs in this narrower bucket will grow more quickly in the near future than the recent past. This is another way in which the FLOP/$ forecast of this model could be considered to be conservative.

But aren't we approaching the physical limits of the current paradigm?

It was beyond the scope of this report to do an investigation into the details of how long progress could continue within the current hardware paradigm, and how promising new paradigms are. Instead this report takes a zoomed out "outside view" approach to forecasting hardware progress, extrapolating the observed relationship between inputs and outputs.

Would investigating the details of the current paradigm imply that this report overestimates future hardware progress? Plausibly, but it's not obvious to me.

- As mentioned above, the report assumes that hardware innovations are getting harder to find, with more research effort required for each successive doubling of FLOP/$. This captures the intuition that progress will become more difficult as we approach the end of the current paradigm.
- In addition, the FTM assumes that the *rate* at which hardware innovations become harder to find itself increases. The returns diminish increasingly steeply as we make more hardware progress. Mathematically, this corresponds to reducing *r* as FLOP/$ increases. This is an additional conservative adjustment to naive trend extrapolation.
- It's possible that the rate of progress will be *faster* in a new paradigm, rather than slower.

2020-FLOP per FLOP

The approach here is the same as in the last section. This time I use inputs to **software** R&D rather than hardware, and measure output as 2020-FLOP per FLOP rather than FLOP/$.

In particular I:

1. Use historical data to calibrate an economic model relating cumulative inputs to output.
    a. This time the data is significantly more uncertain, especially the output data.
    b. Ultimately I assume that **each doubling of software inputs drives ~1.25 doublings of 2020-FLOP per FLOP**.
2. Forecast how quickly inputs will grow after "wake up". I guess that these will grow at ~25%, slightly faster than their recent rate of 20%. This implies that **2020-FLOP per FLOP will also grow at ~31%**.[115]

[115] 1.25 * 25% = 31%.

Economic model relating software inputs to outputs

As last time, the core of the model can be expressed as:

*For each x% increase in cumulative R&D inputs, the output metric will increase by r*x%.*

Our inputs are real $ invested in software R&D; our output metric is 2020-FLOP per FLOP.

The data for software inputs and outputs is not comparably good as for hardware. On the input side, the best I know of are the numbers from Tamay Besiroglu's dissertation. He uses data on the number of authors of papers in three subfields of ML as a proxy for the number of researchers. After some adjustments, we end up with following estimates:[116]

ML subfield	Growth in the annual inputs to software R&D, 2012 - 2020
Computer vision	20%
Natural language processing	36%
Graphs	42%

My guess is that the real growth of inputs in this period is lower, mostly because these growth rates seem very high and I think this estimate is very uncertain.[117] (If these numbers *are* correct, it suggests I've been too conservative with my estimates earlier in this section.)

On the output side, the ideal situation would be to have trustworthy estimates of how many FLOP would be needed to train AGI at different points in time. This would be a direct estimate of the growth in 2020-FLOP per FLOP.

In actual fact, the best I'm aware of is to track the training compute needed to achieve a fixed score on a specific benchmark over time. The best example of this type of analysis that I know for AI is AI and Efficiency, which finds that runtime compute needed for a fixed performance on AlexNet halved every 16 months (growth rate of **52%**) between 2012 and 2020.[118]

[116] Tamay multiplies (an estimate of) the number of paper authors from each country by the average salary of scientists in that country, measured in nominal $. I adjust these numbers downwards somewhat to account for inflation. So the R&D inputs in the table are measured in real $. See my calcs here.

[117] Uncertain for at least two reasons. Firstly, I don't expect the "number of distinct authors" to correlate perfectly with "number of full time researchers". Secondly, the attempt to adjust for the quality of the researchers seems unconvincing: they multiply by the salary of the researcher's country but I'd expect that within each country the new researchers are mostly young.

In addition to the numbers looking high and being uncertain, there's another reason I think the true growth is probably lower. The growth rate of *cumulative* inputs will be lower than the growth rate of *annual* inputs, assuming that growth of annual inputs was slower before 2012. And it's growth of cumulative inputs that matters for this economic model linking R&D inputs to outputs.

[118] I discuss non-AI software trends here.

I believe that **other benchmarks show similar or faster rates of software progress** (with doubling times of 1 - 2 years) when this has been measured.[119]

Estimating *r* for software

If we naively combine the g(2020-FLOP per FLOP) = 52% estimate from AI and Efficiency with Tamay's estimate of the growth rate of real $ inputs to computer vision, we get *r* = 52%/20% = **~2.5**. In other words, each doubling of cumulative investment doubles the efficiency of algorithms 2.5 times.

If we use this as our central estimate we are assuming that the software progress on ImageNet will match that on AGI (in expectation). But in Bio Anchors, Ajeya writes:

> "... *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.*"

Another reason for the same adjustment is that we might imagine AGI software progress is the average of all areas of AI, and that the areas where we're measuring progress have faster progress than the areas we're not interested in measuring. On the other hand, some algorithmic progress seems to reduce the compute needed large training runs *more* than the compute to smaller training run, suggesting the compute needed to train AGI may be falling *more* quickly than that for ImageNet.

Ultimately, I follow Bio Anchors and assume ~half the rate of software progress as observed in ImageNet.[120] For now I will remain consistent with Bio Anchors and make an equivalent adjustment. This halves my estimate to *r* = **1.25.**

Using *r* = 2.5 would bring forward AGI timelines by ~3 years as well as making takeoff faster. In the Monte Carlo I use large uncertainty bounds for this parameter: **0.8 - 5**.

How quickly might software R&D inputs grow after "wake up"?

I don't have much to add here to the analogous section for hardware. In that case, my central estimate was 17%. I want to use a higher number here, for two reasons:

1. The AI software sector is growing from a smaller base. I guessed AI software spend is ~$10-20 billion, vs $70 billion for semiconductor R&D.

[119] For example, table 2 of OpenAI's paper shows similar or faster software gains on other select tasks as on ImageNet, and people I've spoken to suggest recent progress on language models has been faster than the ImageNet progress.

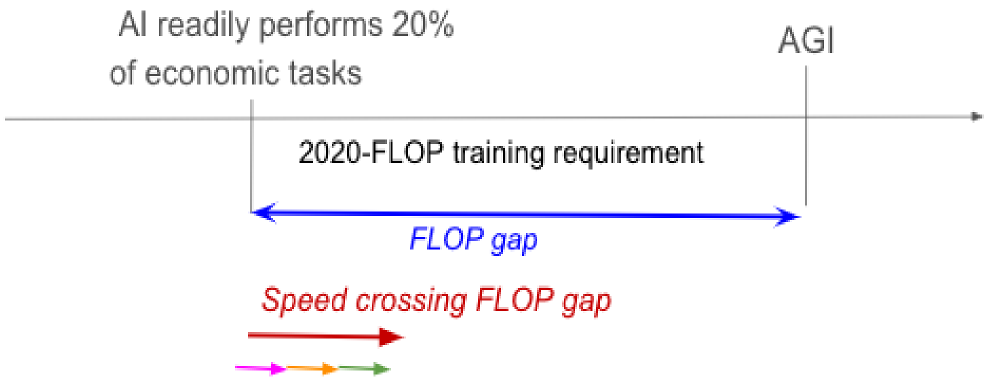
[120] She replaces the observed 16 doubling time with a 2 - 3 year (i.e. ~30 month) doubling time.

2. The AI software sector is apparently *already* growing faster than 17%: the numbers above range from 20% to 42%. I expect that after “wake up” inputs to software will grow as fast or faster as it is currently, based on the huge demand.
    a. I’m going to anchor to the 20%, rather than the 42%, because I used the 20% to estimate *r*.[121]

Based on the above, my central estimate here is that **real $ invested in software R&D will grow at a rate of ~25%**. For similar reasons to above, I’d be surprised if this is <15% or >40%.

This implies a central estimate of g(2020-FLOP per FLOP) = 1.25 * 25% = 31%. This is very similar to the Bio Anchors extrapolation of ~28%; just a touch higher as I expect inputs to grow somewhat faster than they are currently.

Summing up


AI readily performs 20% of economic tasks
AGI
2020-FLOP training requirement
FLOP gap
Speed crossing FLOP gap

g(2020-FLOP) = g($ on FLOP in training run) + g(FLOP/$) + g(software)
g(2020-FLOP) = g(fraction FLOP on training run) + g($ on FLOP globally) + g(FLOP/$) + g(software)
g(2020-FLOP) = Initially ~75%, then ~0% + ~22% + ~25% (increasing over time towards 88%) + ~31%
g(2020-FLOP) = Initially ~153%, then ~78% (increasing over time towards 141%)
= Initially ~0.7 OOMs/year, then ~0.3 OOMs/year (increasing over time towards ~0.6) [122]

This section analysed the effect of fast-rising AI investments on the speed crossing the effective FLOP gap, summarised in the diagram above. This has implications for takeoff speeds and for timelines.

The next section analyses the effect of incremental AI automation on speed crossing the effective FLOP gap. In doing so, it introduces the theoretical machinery for calculating new metrics for takeoff speed that relate to the *number* of AIs and their effects on GDP.

[121] If I’d used the 42% to estimate r, I’d probably be using a lower value for *r*. (Though it’s possible that software progress has simply been twice as fast in Graphs than in Computer Vision.)
[122] Link to diagram.

5. AI automation

I think this section of the long summary summarises the important takeaways from sections 5 and 6 in a just few pages. I'd only read this section if you really want to understand the math behind the automation models I'm using further, but aren't familiar enough with growth economics already to read the mathematical description of the Full Takeoff Model.

Summary

This section analyses the effect of incremental AI automation on takeoff speeds.

By "incremental" I just mean that I model AI automation as a *continuous* process of automating more and more tasks, without any discrete "jumps" in which AI suddenly automates lots of cognitive tasks in one fell swoop. I do *not* mean to imply that this process happens slowly; in fact it may happen very quickly. The speed depends on the size of the effective FLOP gap and how quickly we cross it.

As a recap, the two key inputs to takeoff speed are the **effective FLOP gap** (measured in 2020-FLOP) and our speed crossing the effective FLOP gap, **g(2020-FLOP)**. Section 3 described this framework and laid out evidence informing the effective FLOP gap, and section 4 analysed the effect of rising AI investment on g(2020-FLOP).

This section analyses the effect of AI automation on g(2020-FLOP). As in section 4, I'll analyse each of the three components of 2020-FLOP separately. This time I'll take them in reverse order: 2020-FLOP per FLOP, then FLOP/$, and finally $ on FLOP.

For each component, I will use a certain economic model to analyse the effect of AI automation.[123] In this section I use a simple version of the model which excludes certain bottlenecks.[124] Bottlenecks are discussed in section 6.

Then I'll explain how we can calculate some additional metrics of takeoff speeds using our model of AI automation. In particular, metrics relating to the number of AIs and their effects on GDP.

The key takeaways from all this are:

- AI automation causes growth of the three components to accelerate continuously. By the time we reach full automation of cognitive labour (AGI), they can double in months or faster.

[123] I'll use a task-based model, and explain it below.
[124] This is the Cobb Douglas version of the task-based model.

    - This includes a feedback loop where more 2020-FLOP leads to training better AIs and running more AIs, which in turn allows us to produce more 2020-FLOP.
- Initially, when a small fraction of cognitive tasks have been automated, AI automation has a small effect on the growth of the three components compared to the rising investment discussed in section 4.
    - The effect on FLOP/$ and software becomes significant, relative to rising human investment, when ~25% of cognitive tasks have been automated.
    - The effect on $ on FLOP becomes significant when ~45% of cognitive tasks have been automated.
    - This doesn't account for bottlenecks, which would increase these percentages somewhat.
- Unfortunately, I'm not aware of a simple, analytically tractable way to calculate overall takeoff speed metrics given the feedback loops involved here. My approach instead is to simulate the model and do a sensitivity analysis, which I'll present in section 7.

This section is more technical than other sections. Many readers will prefer to skip to the next section.

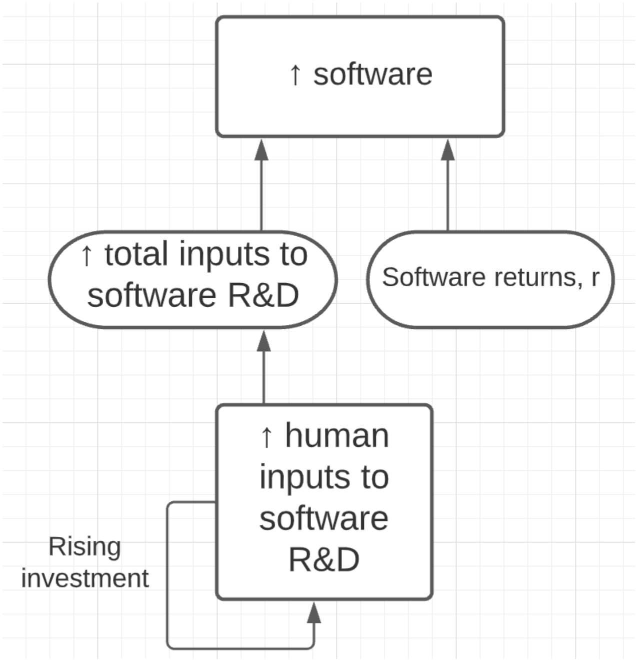
2020-FLOP per FLOP

2020-FLOP per FLOP, my operationalism of the quality of AI algorithms, increases because of software research. I'm going to model incremental AI automation as a continuous transition from "world 1", where humans do ~all the software R&D, to "world 2", where AIs do ~all the software R&D. First I'll discuss the dynamics of world 1; then those of world 2. Lastly I'll explain how I'm modelling the transition between the two.

World 1

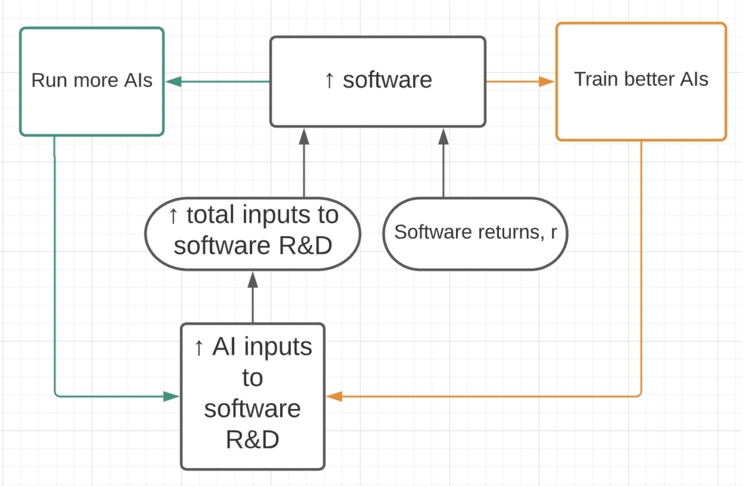
We're currently in a world where software research is overwhelmingly done by human workers. Let's call this world 1. In section 4, I forecasted how quickly these human inputs to software R&D might grow after "wake up", and what effect this might have on 2020-FLOP per FLOP.[125]

[125] I guessed that each doubling of cumulative software R&D inputs would cause 2020-FLOP per FLOP to double 1.25 times. I measured in the inputs as real $ spent on R&D.

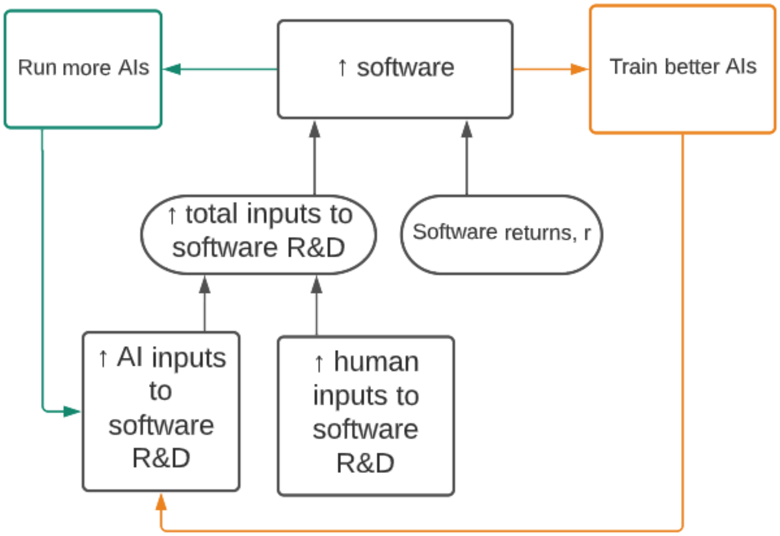

↑ software
↑ total inputs to software R&D
Software returns, r
↑ human inputs to software R&D
Rising investment

World 2

Once we have billions of AGIs – remembering that AGI is AI that can automate 100% of cognitive tasks – we'll be in a world where software research is overwhelmingly done by AIs. Let's call this world 2.



Ultimately, we'll model the transition from world 1 to world 2 by assuming progress is driven by a combination of AI and humans.


Run more AIs
↑ software
Train better AIs
↑ total inputs to software R&D
Software returns, r
↑ AI inputs to software R&D
↑ human inputs to software R&D

Before this, let's think about what will happen to the level of software world 2?

We can understand this dynamic by answering two questions:

1. How long does the first doubling of software (i.e. 2020-FLOP per FLOP) take in world 2?
2. How do the lengths of the software doublings change over time in world 2?

How long is the first software doubling?

The answer to the first question depends on i) how many human researcher-years are needed to double software when we first get AGI, and ii) how many AGIs you can run (where each AGI is as productive as a human per day).[126]

Here's a very rough estimate of (i). If there are 20,000 high-quality human researchers on software today[127] and software doubles every ~2 years[128] then it currently takes 40,000 researcher years to double 2020-FLOP per FLOP. Let's assume this is ~100X higher by the time we get AGI due to diminishing returns from the research that happens before then.[129] That implies ~4 million researcher-years to double software when we get AGI.

[126] A more precise formulation of (ii) is: *how many human workers you'd need to make the same software progress per day as the collection of AIs you can run*. This formulation reflects the fact that there may be a variety of different AIs doing different software tasks, rather than just AGIs doing all of them. (Indeed, this is what happens in the Full Takeoff Model!)

[127] DeepMind has 1000 employees, and I earlier assumed that total AI software input was 20X that of DeepMind. (The 20X is a guess, and someone could probably easily get a better number.) Facts that are potentially relevant to a more careful estimate: 200 new AI PhDs in 2020; 80,000 AI journal publications in 2020.

[128] In the ImageNet example, 2020-FLOP per FLOP doubled every 16 months.

[129] 100X corresponds to cumulative research inputs growing by 100X by the time we develop AGI, which could be from growing 23% per year for 20 years before we develop AGI. e^(0.23*20) = 100. I'm using a model in which the effort needed to double software is proportional to the total cumulative input so far.

To estimate (ii), suppose you trained AGI with 1e32 2020-FLOP, the training run took 4 months, afterwards you used 10% of your training compute to run AGIs doing software research, and running an AGI required 1e16 2020-FLOP/s.[130] With these conservative assumptions,[131] you'll have 100 million AGIs doing software research and so the first software doubling will take ~1 months.

Our estimate here could easily have been more aggressive:

- If instead AGI requires 1e36 2020-FLOP to train but has the same runtime requirements (e.g. due to a long horizons), the first software doubling will be OOMs quicker as we'll have more 2020-FLOP to run AGIs.
- If AGI has significant "one-off" productivity advantages over humans for R&D (run faster in serial time, no sleep or leisure, better motivation and coordination, all AGIs are copies of the most productive AGI) then this will speed up software progress. My current guesstimate of these advantages for R&D is ~60X.

A more aggressive estimate of 10X naively implies software doubling in a couple of weeks, though at that point bottlenecks from computational experiments become salient.

The point here is not to trust these exact numbers.[132] It is to see the way in which the time for the first software doubling depends on the AGI's training compute, AGI's runtime compute, and the amount of software research that has happened before AGI. It is secondly to make plausible the idea that the initial software doubling in world 2 could happen in months or much less.

How do the lengths of the software doublings change over time?

In world 2, the annual inputs to software R&D are proportional to the 2020-FLOP used for this purpose.[133] This means that there is a very direct feedback loop between the inputs and outputs of software research, unlike today. Doubling the software R&D output metric also doubles the input to software research.

[130] I.e. 1e16 2020-FLOP are needed to do as much quality-adjusted cognitive labour as a human does in 1 second.

[131] Small training 2020-FLOP for AGI; a large runtime compute for AGI; only 10% of compute on software work; ignores the fact that inference is more efficient than training; ignores the possibility of AIs perform some tasks much more compute efficiently than human brains (e.g. like calculators are OOMs more compute efficient than human brains at arithmetic, or facial recognition systems are OOMs more efficient at recognising faces, or GPT-3 is OOMs more compute efficient at writing poems).

[132] In fact, the first doubling time depends on many interacting factors (like "how much will returns to software research have diminished by the time we get to AGI?") and is hard to estimate analytically. Ultimately, I get around this by simulating the system and doing a sensitivity; software doubling times are almost always extremely fast (<6 months) by the time we have AGI, even if there isn't a "software singularity" (discussed below). But the simulation omits bottlenecks from computational experiments.

[133] We can in principle distinguish between 2020-FLOP for *training* AGI (my main focus thus far in the report) and the 2020-FLOP for *running* AGI. It is possible that they grow at different rates, e.g. if a new algorithm reduces training FLOP but not runtime FLOP. For our present purposes, it is the *runtime* 2020-FLOP that concerns us. For ease of exposition, I won't explicitly distinguish between these two unless it is relevant. [Describe what Full Takeoff Model does here. TODO]

The feedback loop is:

Better software → more 2020-FLOP → more software R&D → better software

It turns out that, with this feedback loop, there are two broad possibilities.

1. Software singularity - quicker and quicker doublings. If returns to software R&D exceed a certain threshold, the feedback loop is so powerful that there's a "software only singularity". The level of software, quantified here as 2020-FLOP per FLOP, grows faster and faster, theoretically going to infinity in finite time. And this happens even using a fixed quantity of physical FLOP to run the AIs. In practice, of course, the software returns become worse before we go to infinity and we move to the second possibility.

2. Software fizzle - slower and slower doublings. If returns to software R&D are below a certain threshold, the level of software grows more and more slowly over time,[134] assuming a fixed quantity of physical FLOP. (If the amount of physical FLOP is in fact growing increasingly quickly, then the level of software can do the same. But software progress is reliant on the growth of physical FLOP.)

Which possibility will obtain? It turns out that there is a software singularity just if r > 1, where r is defined as in section 4:

For each doubling of cumulative R&D inputs, the output metric will double r times.

[134] There is technically a "knife edge" third possibility where software grows at a constant exponential rate, if software returns are *exactly* equal to the threshold. I'm setting this aside because it's a knife edge result.

r > 1 means that doubling cumulative software inputs causes 2020-FLOP per FLOP to *more* than double.[135] I argue that this is plausible here, considering various estimates of r and the fact that r will likely fall over time.

What are the implications of a software singularity for takeoff? In short, it would not guarantee a fast takeoff in every important metric, but it would make takeoff faster.

- **Make takeoff faster.** A software singularity would lead to very fast software progress as we approach AGI, significantly accelerating software growth. This means we cross the effective FLOP gap more quickly and AI capabilities improve more quickly immediately after AGI. This progress in AI capabilities wouldn't be bottlenecked by the need to print new chips.
- **Increase the peak capabilities reached shortly after AGI.** If there's a software singularity, AI software could rapidly grow by many OOMs and approach physical limits in the months after AGI, without needing to wait on the design and production of new AI chips. This has implications for how a small calendar lead in developing AGI could translate into a total capabilities advantage shortly after developing AGI.
- **No guarantee of fast takeoff.** During a software singularity, each doubling of software need only happen slightly faster than the previous doubling. In fact, only under extreme assumptions does each doubling happen twice as quickly as the last.[136] This means that on a metric of takeoff speed in terms of the ratio between successive software doubling times, world 2 does not involve a fast takeoff even if there's a software singularity. That said, a fairly *rapid transition* from world 1 to world 2 would be more likely to drive a fast takeoff if there's a software singularity.

I've just discussed the internal dynamics of 2020-FLOP per FLOP in world 2 in their implications for takeoff speed; I analysed the dynamics of world 1 in section 4; now I describe a model of a gradual transition from world 1 to world 2.

[135] Why is this the condition for software singularity? Suppose that you initially have 1000 AGIs doing software work. Let's say it takes them 1 year to double cumulative software inputs. By this time, 2020-FLOP per FLOP has increased by a factor of 2^r. (This follows straight from the definition of r.) If r > 1 then 2020-FLOP per FLOP has *more* than doubled, and so your software research input has *more* than doubled to >2000 AGIs. How long will it take you to double cumulative inputs a second time? If your population of AGIs were still 1000, it would take you twice as long (each doubling of cumulative inputs takes twice as much effort as the previous doubling). But because you now have >2000 AGIs, it will take you *less* long and you'll double cumulative inputs in less than a year. This means the growth rate of cumulative inputs is increasing. g(output) = r * g(cumulative inputs), so the growth rate of output is also increasing.

How does introducing a "stepping on toes" assumption change this analysis? Not much. Stepping on toes is expressed mathematically as I = C^lambda, lambda < 1. In this case, the condition for software singularity becomes r*lambda > 1. If we held our estimate of r fixed, then a software singularity would be less likely. However, consistency with the historical data requires us to raise our estimate of r to exactly compensate if we lower the value of lambda. This is because the historical data constrain r*lambda directly. For example, in section 4 I said the data suggested **r = 2.5**. But if I'd accounted for stepping on toes, I'd have instead said that the data suggests **r*lambda = 2.5**. The implication of the historical data for the software singularity is unchanged. In both cases the singularity happens just if the quantity > 1, and the historical data suggest the quantity equals 2.5.

[136] I analyse this dynamic more in appendix TODO.

Transition from world 1 to world 2

We can represent the annual inputs to software research mathematically in worlds 1 and 2.

In world 1, the annual inputs to software research are given by:

$$I_S = L_S \qquad (1)$$

where L_S is the number of human software workers.[137] [138]

In world 2, the annual inputs to software research are given by:

$$I_S = C_S \qquad (2)$$

where C_S is the quantity of 2020-FLOP used by AI systems for software R&D. (We will continue to use the same economic model to predict how the annual inputs I_S will affect the output metric, 2020-FLOP per FLOP.)

How can we model the move from world 1 to world 2? The best approach I know of here is the task based model.[139] This model supposes that software R&D involves a large number of distinct cognitive tasks.[140] Total R&D input depends on the inputs to each task.[141]

[137] This is slightly different from section 4 where I measured software inputs in units of real $. In this section it will be simpler to talk in terms of # researchers, rather than real $. We can relate these two input metrics by estimating the annual rise in real salaries during this period, which I'll assume is 2%. Real $ inputs should grow 2% quicker than # researchers. For example, I estimated that real $ inputs to software would grow at 25%, and this would correspond to 23% growth in # researchers.

[138] If we want to account for the "stepping on toes" effect we could instead write I = L^lambda. I won't do this for simplicity of exposition, but will note in footnotes or appendices when the stepping on toes effect would meaningfully change the dynamics.

[139] During my previous investigation into whether AI could drive explosive growth I didn't come across anything more promising despite studying most mainstream growth models and most growth models of transformative AI (e.g. this review). This model also seems to be favoured by economists studying the economic implications of advanced AI, e.g. Aghion et al. (2017) and Hanson (2001). Some advantages of this model are: quantifying what % of the way from world 1 to world 2 we've travelled at each point in time and quantifying the effect of this on software R&D, allowing flexible incorporation of the degree of bottlenecking between different tasks (which I'll use in section 6); being fairly intuitive to explain. With bottlenecks, the model looks like (see section 9.2) it can explain a good chunk of the growth of the last 150 years as resulting from automation. Thus we are using a model in which future AI automation is a continuation (and significant acceleration) of past automation (which for the first time ever leads to full automation).

[140] In the model these tasks are fixed over time. However, in the version that incorporates bottlenecks (that I'll introduce next section), the relative importance of these tasks *does* change over time. In particular, if we automate a task our output on that task increases and so the task becomes *less* important. The result is that the non-automated tasks grow in importance. This matches the recent trend of hard-to-automate sectors like education and healthcare growing as a fraction of GDP while automatable sectors like agriculture fall as a share of GDP. Also, the growing importance of non-automated tasks can represent the possibility that *entirely new* tasks are introduced that AI cannot (initially) perform. In our model, we'll think of these new tasks as new applications of existing tasks that AI couldn't perform.

[141] This mathematical footnote is not needed to understand what follows. This section uses the Cobb Douglas version of the task-based model, which is simple and doesn't include bottlenecks. In that model,

In world 1 humans do ~all the tasks[142] and, when you do the maths of the task based model, this results in equation (1). In world 2 AIs do all the tasks and this results in equation (2). We model intermediate worlds as ones where AI performs a fraction f of tasks, 0 < f < 1.[143] It turns out that, in the Cobb Douglas version of the task-based model (that excludes certain bottlenecks)[144], total R&D input is given (up to an unimportant constant) by:

$$I_S = L_S^{(1-f)} * C_S^{f} \qquad (3)$$

Notice that when f = 0 this becomes equation (1) and when f = 1 it becomes equation (2). As we continuously automate a greater fraction of tasks, the exponent on C increases gradually from 0 to 1.

(*The Full Takeoff Model assumes a constant fraction of tasks are performed by physical compute – physical FLOP. I omit this here for simplicity.*)

How should we model the process by which tasks are incrementally automated? There are two components here:

1. When will we develop AI that can perform various cognitive tasks?
2. When do you have enough runtime compute to actually automate various tasks?

the total R&D input depends on the input to each task as follows. You multiple the inputs of every task together to get the total R&D input. Mathematically, suppose there are N tasks, and input to each task is given by X_1, X_2,..., X_N. Then total R&D input I = X_1 * X_2 *... * X_N. The implication is that you want to spread your inputs evenly across the tasks, as a tiny input on any task could really let you down (and zero input on any task will mean zero total input).

[142] Is it really true that human workers do all the tasks necessary for software development today? Perhaps conducting AI experiments is a crucial part of the process, and this "task" is already performed by computers. I discuss this in appendix TODO.
In the Full Takeoff Model, I actually assume that, even if AI experiments are not part of software development, initially a *small* percentage of software tasks are performed by 2020-FLOP. I am thinking here of the way in which software developers offload certain cognitive tasks to calculators and spreadsheets. These are only a small fraction of the relevant tasks because only a small fraction of the money invested in software development goes to buying the machines that do these types of tasks. (E.g. calculators are very cheap and use of google sheets are very cheap compared to a developer's salary.) The assumption that a small percentage of software tasks, rather than 0 tasks, are initially performed by 2020-FLOP does not materially affect the results.

[143] The model implies that an equal fraction of output is paid to each task in 2021. This means that the 'fraction of tasks' in the model matches my earlier definition of the "% of cognitive tasks", where I weighted each task by the salary-weighted time spent on it in 2021. (I am assuming that all tasks performed by software workers are cognitive – as opposed to partly-physical tasks like "building a table" – and so can all ultimately be performed by AI.)

[144] This section analyses the implications of the Cobb Douglas version of the task-based model because it is (relatively) simple to understand and tractable to analyse analytically. The Cobb Douglas version excludes certain bottlenecks, some of which are included in the CES version of the task-based model. I'll flag when results from the Cobb Douglas version might not carry over to the CES version. I discuss the CES version, and bottlenecks more broadly, in section 6.

When will we develop AI that can perform various cognitive tasks?

My proposal is to extend the Bio Anchors model for when we train AGI. Let's say that we use Bio Anchors to predict that we'll need 1e36 2020-FLOP to train AI that can automate 100% of cognitive tasks (my definition of AGI). We can extend it to **estimate the 2020-FLOP needed to train AI that can perform x% of cognitive tasks, for various values of x**.[145] For example, we might assume that you automate 50% tasks with 1e34 2020-FLOP and automate 20% of tasks with 1e31 2020-FLOP.

What evidence can we use to inform these assumptions?

- Bio Anchors, or perhaps other methods, can inform how much 2020-FLOP we think is needed to train AI that can perform 100% of tasks.
- My earlier discussion of the effective FLOP gap can then inform how much 2020-FLOP we think is needed to train AI that can perform 20% of tasks.
- The Full Takeoff Model (FTM) then uses a pretty hacky method to extrapolate to the training run needed to automate x% of tasks for any x.

The spread of these tasks in 2020-FLOP space, together with g(2020-FLOP), will dictate how quickly new tasks are automated. It is a hugely important and uncertain input to this framework, which strongly influences how suddenly we transition from world 1 where human workers are the key input to economic production to world 2 where AI is the key input.

When do you have enough runtime compute to actually automate various tasks?

As discussed in section 2, having AI that can perform a task is not sufficient to fully automate it. In addition, you must have enough runtime compute to actually replace the human workers that currently do the task.[146] [147]

To know whether we have enough runtime compute to fully automate a task, we need to know:

1. How many human workers are currently performing that task?
2. How much compute are we using to run AIs doing software R&D?
3. What are the runtime compute requirements for AIs to have the same output at the task as a human worker?
4. What one-off productivity gains do AIs have over humans?

[145] In fact, to fully specify the model we'll need to determine the 2020-FLOP needed for every value of x between 0 and 100, though the results are not sensitive to small variations.

[146] Indeed, you must be able to run enough AIs doing the task that it becomes more profitable for human workers to perform a different task instead.

[147] Of course, even this is not sufficient. You might have AI that can readily perform a task, and enough compute to affordably automate all instances of that task, but not *actually* automate it due to some other bottleneck like regulations, incumbents resisting automation, or just the minor effort involved in introducing the AI into the workflow. The Full Takeoff Model does *not* incorporate delays between getting enough training and runtime compute to fully automate a task and actually fully automating it; I discuss this weakness in section 10[TODO link]. (This is why the training compute threshold should be interpreted as the 2020-FLOP training requirement for AI to be able to "readily perform" x% of cognitive tasks.

Our Full Takeoff Model (FTM) makes assumptions about these quantities. The important high level points are that:[148]

1. FTM assumes 1.6 million people[149] doing software R&D in 2022, which grows at 20% before "wake up" and 25% after "wake up". Also workers' time is split evenly over equally important non-automated tasks (so if there are 5 equally important non-automated tasks, workers spend 20% of their total time on each).
2. FTM tracks total global compute and, after "wake up", the fraction of this used to run AIs doing software R&D rises very rapidly to ~10%. Why so high? After "wake up", demand for AI software R&D will be high, and so a notable fraction of AIs will be assigned to it if they can be useful.
3. FTM assumes that the *runtime* requirements for different tasks are spread out over multiple OOMs, just as the *training* requirements.
    a. My central estimate has AGI at 1e17 2020-FLOP/s and 20% of tasks at 1e15 2020-FLOP/s.
    b. The spread of runtime requirements is smaller than the spread of training requirements for two reasons.
        i. A 10X increase in runtime compute typically corresponds to a 100X increase in training, e.g. for Chinchilla scaling.
        ii. Increasing the horizon length of training tasks will increase training compute but not runtime.
4. The FTM assumes ~60X one-time gains for AGIs over humans doing R&D.

The software sector is relatively small and so lack of runtime FLOP only prevents task automation if AI training requirements are extremely low.

To summarise the above two subsections, a task is fully automated when we i) have done a big enough training run to develop AI that can perform the task, **and** ii) we have enough runtime compute for AIs to replace all humans at that task. I believe this is a natural way to integrate Bio Anchors with a task-based model of incremental automation.

Section 7 considers a model in which having more runtime compute than you need to automate a task can compensate for not having enough training compute. The consequence is that software R&D can be fully automated ***much*** earlier due to an abundance of runtime FLOP.

This simplistic model obviously omits many factors that in practice affect the automation of tasks; I'll discuss this in section 8.

[148] Some additional details are included in this appendix.

[149] In fact, the true number of people doing AI software R&D is lower by 1-2 OOMs. My methodology was to multiply world population by an estimate the fraction of GWP spent on software R&D (0.02%); but in fact these workers' salaries are much higher than average and per-person capital costs for this industry are also unusually high. It doesn't matter much to the results as the bottleneck to automation is nearly always training compute rather than runtime compute.

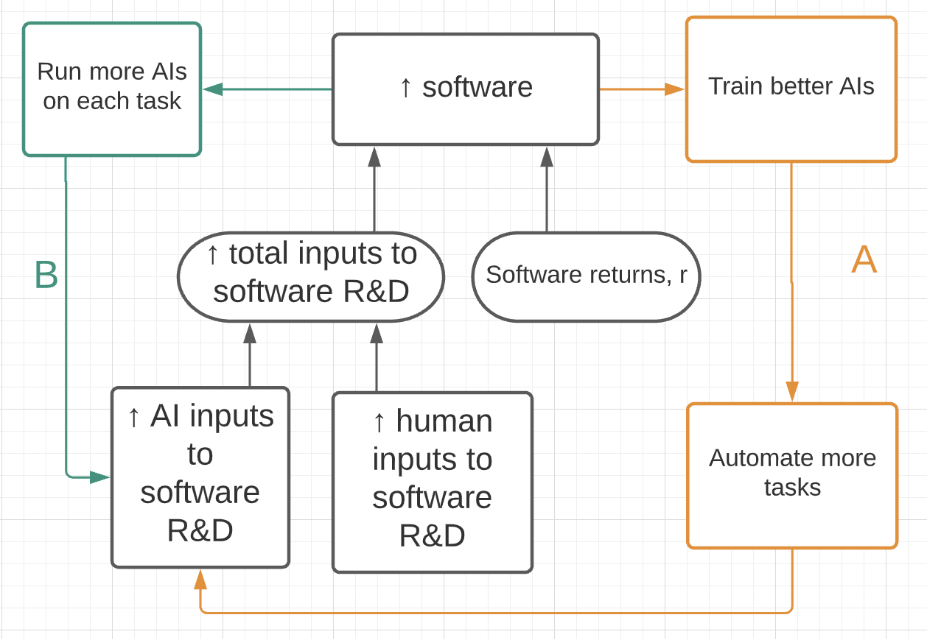
The dynamics during the transition from world 1 to world 2

During the transition, there is the following two qualitative feedback loops:

(A) Better software[150] → more 2020-FLOP **in largest training run** → **more tasks automated** → more input to software R&D → better software

(B) Better software → more 2020-FLOP **for running AIs** → **more AIs per automated task** → more input to software R&D → better software

Both loops increase inputs to software R&D; I've highlighted the differences in **bold**.


Run more AIs on each task
↑ software
Train better AIs
B
↑ total inputs to software R&D
Software returns, r
A
↑ AI inputs to software R&D
↑ human inputs to software R&D
Automate more tasks

[151]

Qualitatively, the result of this feedback loop is that software R&D inputs and 2020-FLOP grow at increasingly fast rates – super exponential growth – as software tasks are automated.[152]

When does this super exponential growth become quantitatively significant?

[150] I use this interchangeably with "more 2020-FLOP per FLOP".

[151] Link to chart.

[152] Equation (3) implies the growth rate of software inputs is given by g(I) = (1-f)*g(L) + f*g(C). We know that g(C) > g(L): 2020-FLOP grows much quicker than human inputs to software R&D. So as f increases, g(I) increases. If the growth rate of software *inputs* increases, so does the growth rate of software *output:* 2020-FLOP per FLOP. (This follows from the SEG). And if g(2020-FLOP per FLOP) increases then so does g(2020-FLOP), as long as g(physical FLOP) is not falling (in fact it will be rising). This establishes that both g(I) and g(2020-FLOP) increase as f increases. In short: f increases → g(I) increases → g(2020-FLOP per FLOP) increases → g(2020-FLOP) increases. This argument could fail for two reasons. Firstly, in the CES version the importance of tasks done by AI falls over time and so we must automate tasks quickly enough to counteract this for the argument to go through. Secondly, if g(C) falls for some other reason (e.g. we stop ramping up the fraction of compute used for software R&D) then g(I) may fall and the argument is blocked.

In section 4 I guessed that the growth rate of real $ inputs to software R&D would be 25% after “wake up”. We can ask: **What fraction of tasks must be automated before the effect of AI on software inputs is larger than this?**

Equation (3) implies that the growth rate of software inputs due to AI equals f * g(2020-FLOP). If g(2020-FLOP) = 100%[153], then this first exceeds 25% when **f = 0.25.** In other words, it is when roughly ~25% of cognitive tasks have been automated that rising AI inputs become more important to software R&D than rising human inputs. I’ll revisit this question in section 6 when we discuss bottlenecks, which will push towards a somewhat larger fraction.[154]

Summing up

Incremental AI automation will increase g(2020-FLOP per FLOP) as we cross the effective FLOP gap. This effect is smaller than the effect of rising human inputs until AI has automated ~25% of cognitive tasks. By the time we reach AGI, 2020-FLOP per FLOP will be doubling in months or much less.

More generally, as we cross the effective FLOP gap, g(2020-FLOP per FLOP) depends on:

1. The returns to software R&D, quantified by the parameter r.
2. The rate at which inputs to software R&D grow. There are two sources of growth:
    a. Increasing number of people doing software R&D.
    b. Increasing fraction of tasks done by AI, and an increasing number of AIs doing each task.

This completes my discussion of the effect of incremental AI automation on 2020-FLOP per FLOP. The next section discusses the effect on FLOP/$.

FLOP/$

In section 4 I analysed how fast-rising AI investment might affect FLOP/$ after “wake up”. I guessed that inputs to hardware R&D might grow at 17%, eventually driving FLOP/$ to grow at ~88%.

This section extends that analysis by additionally considering the effect of incremental AI automation on FLOP/$. Like with software, we’ll see that g(FLOP/$) increases as more tasks are automated such that FLOP/$ may be doubling in months or quicker by the time we have AGI.

[153] A ballpark figure from section 4.

[154] In fact, this analysis involved a few simplifications. Firstly, the bottleneck dynamic introduced in the next section will reduce the effect of AI automating 20% of tasks. Secondly, the effect of rising human inputs becomes less important as AI automates more tasks, which pushes in the other direction. Overall, I think the answer “f = 0.2” given here is too low, a more realistic answer is maybe f = ~0.35. [Maybe Jaime can investigate the relative contributions from AI automation vs rising human inputs for rho = -0.5?]

The analysis is very similar as for software, so I start by noting the points of similarity and difference.

Comparing the effect of AI automation on hardware vs software R&D

The effects of AI automation on g(FLOP/$) is similar to its effects on g(2020-FLOP per FLOP). Here are the key similarities in my analysis of each:

- We start in world 1, with humans doing 100% of the cognitive tasks needed for hardware R&D.
- We end up in world 2, where AIs do 100% of those cognitive tasks.
- There's a transition from world 1 to world 2 where the fraction of cognitive tasks done by AIs increases continuously from 0% to 100%. g(FLOP/$) increases significantly during this transition.
    - During the transition, a task is automated when i) we've done a big enough training run that the resultant AI can perform the task, **and** ii) we have enough runtime compute to replace all humans doing that task.[155] [156]

There are a few important changes:

- **Delays before innovation can boost AI capabilities.**
    - Software improvements can be rolled out immediately over all existing compute, increasing AI inputs to software R&D without delay. By contrast, there are significant lags between designing new chips and using the new chips to run AIs. At the least you need to manufacture new chips from an existing fab; you may also need to build new manufacturing equipment (e.g. for making chips of a new node size).
    - The FTM models the need to manufacture new chips, tracking both the stock of chips and the new chips produced each year. It also includes an optional lag between designing new chips and beginning to manufacture them. More.
- **Physical capital is needed for hardware R&D.** Hardware R&D sometimes requires experiments to test the behaviour of materials and new chip designs. To incorporate this, a fixed fraction α of tasks are performed by capital; the rest are cognitive tasks.[157] My best guess is $\alpha = 0.3$.[158]

[155] FTM assumes 16 million people doing hardware R&D in 2022, which grows at 7% before "wake up" and 17% after "wake up"; people are split evenly over non-automated tasks. (16 million is too high by 1-2 OOMs; the reason is that we multiply world population by an estimate the fraction of GWP spent on software R&D (0.02%). It doesn't matter much because the bottleneck to task automation is nearly always training compute rather than runtime compute.

[156] The hardware R&D sector employs a relatively small number of people (though more than AI software) and so lack of runtime FLOP only prevents task automation if AI training requirements are low (e.g. AGI trained with <1e31 FLOP [TODO confirm]).

[157] So this simplistic model assumes that all R&D tasks performed by labour are cognitive tasks, not requiring physical actuators. For labour tasks that aren't cognitive, it is probably best within this framework to include them as tasks done by physical capital. More.

[158] Alpha gives the fraction of R&D costs paid to physical capital (as opposed to cognitive labour) in 2021. This data suggests labour share is 65% and maybe as high as 90%, depending on whether you label

- The three equations from last section become:
  $I_H = K_H^{\alpha} L_H^{(1-\alpha)}$ (1*) - humans do all cognitive tasks, world 1
  $I_H = K_H^{\alpha} C_H^{(1-\alpha)}$ (2*) - AI does all cognitive tasks, world 2
  $I_H = K_H^{\alpha} L_H^{(1-\alpha)(1-f)} C_H^{(1-\alpha)f}$ (3*) - AI does fraction f of cognitive tasks
  where K_H, L_H and C_H gives the amount of physical capital, labour and 2020-FLOP used in hardware R&D; and I_H gives the resultant real input to hardware R&D.
- The Full Takeoff model incorporates a similar dynamic into its model of hardware R&D. A constant fraction of tasks are performed by physical compute. The total input to these tasks is proportional to the amount of available physical FLOP – not 2020-FLOP.

Having made this comparison with hardware vs software, I'll briefly describe the dynamics affecting g(FLOP/$) in world 2 and in the transition from world 1 to world 2.

World 2

As with software, we consider two questions.

1. How long does the first doubling of hardware (FLOP/$) take in world 2?
2. How do the lengths of the software doublings change over time in world 2?

How long does the first doubling of hardware (FLOP/$) take?

The considerations influencing this are similar as for software. We forecast how many researcher-years will be needed to double hardware, and compare this to how many AGIs we'll be able to run. If we'll need 1 million researcher-years but we'll have 2 million AGIs, the first doubling takes 6 months. The ballpark estimate here goes the same as for software, with the result that the first doubling could happen in months or less.

One additional complication here is that hardware R&D progress might be bottlenecked by limited physical capital. This could cause the first doubling to happen more slowly.

How do the lengths of the hardware doublings change over time

As with software, we can distinguish two scenario:

1. Hardware singularity - quicker and quicker doublings. If returns to hardware R&D exceed a certain threshold, the feedback loop is so powerful that there's a "hardware only singularity". The level of FLOP/$ grows faster and faster, theoretically going to infinity in finite time. This

certain subcategories are spent on labour vs capital. However, the data is for **generic** R&D rather than **hardware** R&D.

dynamic is initially curtailed by the difficulty of printing new chip designs fast enough to quickly match the current stock of hardware.

2. Hardware fizzle - slower and slower doublings. If returns to hardware R&D are below a certain threshold, FLOP/$ grows more and more slowly over time,[159] assuming a fixed $ spend on FLOP.

Which scenario will obtain? The condition for **hardware singularity** is $(1 - \alpha)r > 1$. My best-guess values of α=30% and r = ~5 imply it would happen comfortably. Though of course r may be lower by the time we reach AGI. The question is less important than for software because any hardware singularity would be slowed by delays printing chips, as mentioned above.[160]

Transition from world 1 to world 2

Just like last time, there are two feedback loops at play during this transition:

(A*) Better hardware[161] → more 2020-FLOP **in largest training run** → **more tasks automated** → more input to hardware R&D → better hardware

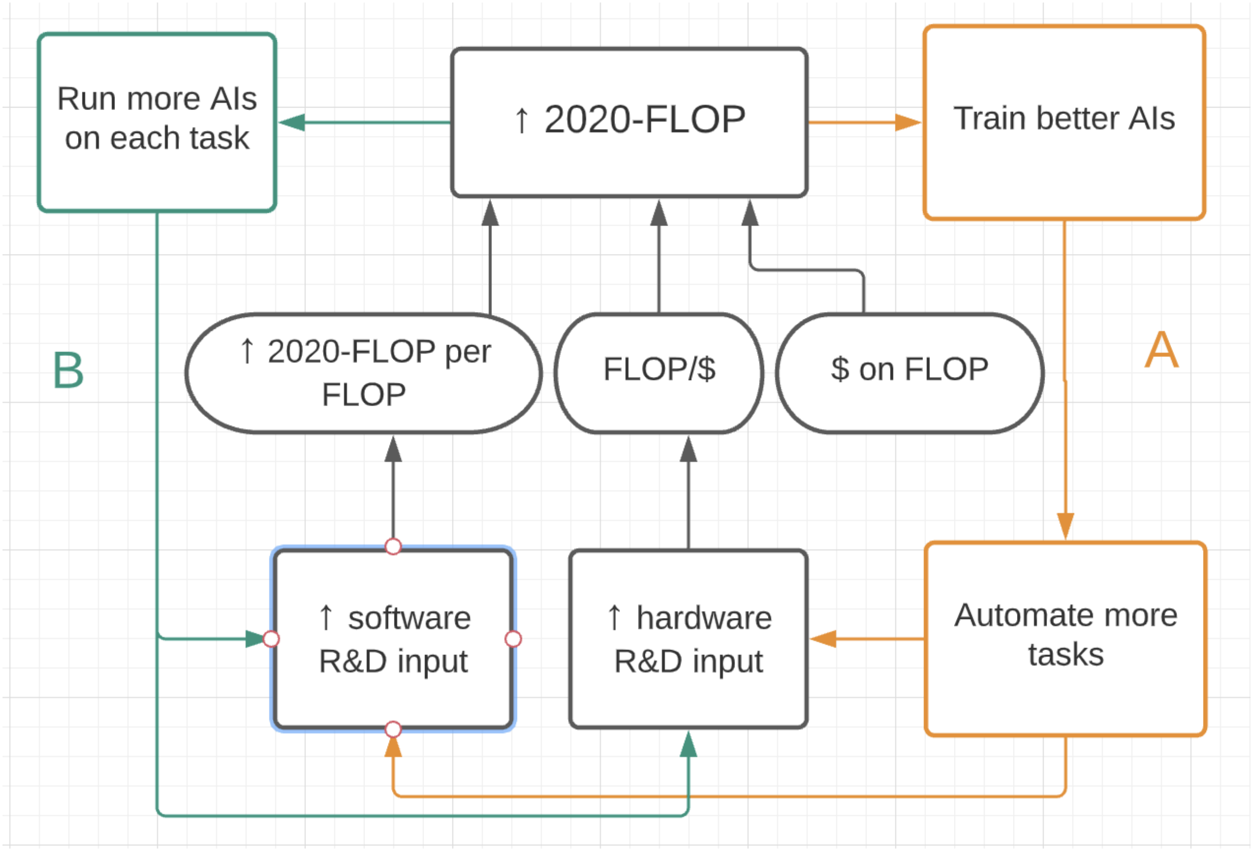
(B*) Better hardware → more 2020-FLOP **for running AIs** → **more AIs per automated task** → more input to hardware R&D → better hardware

We can show these feedback loops alongside those for software:

[159] There is technically a “knife edge” third possibility where software grows at a constant exponential rate, if software returns are *exactly* equal to the threshold. I’m setting this aside because it’s a knife edge result.

[160] Even if the conditions for software singularity don’t obtain and the conditions for hardware singularity don’t obtain, there can still be a joint hardware-and-software singularity if the combined returns are high enough. And even if *this* doesn't obtain, I expect both FLOP/$ and software to eventually grow increasingly quickly due to GWP growth accelerating (which I discuss later).

[161] I use this interchangeably with “more FLOP/$”.



[162]

It’s worth emphasising that the fast growth of 2020-FLOP is playing a dual role: bigger training runs which lead to greater automation **and** more runtime compute to run AIs doing tasks that have been automated.

AI automation becomes the dominant source of R&D input growth at about the same time as for software. In section 4 I guessed that human inputs to hardware R&D would rise at 17%. Equation (3*) implies that the growth rate of software inputs due to AI is given by the expression $(1 - \alpha)$ * f * g(2020-FLOP). (Recall f is the fraction of cognitive tasks automated by AI.) If g(2020-FLOP) = 100% (from section 4), then this first exceeds 17% when **f = 0.25**.[163]

[162] Link to diagram. There are of course human inputs to hardware and software R&D, but these aren’t represented explicitly in the diagram.

[163] 100%*0.7*0.25 = 17.5%. Notice that f = 0.25 is the same as the result for software R&D. There are in fact two differences between software and hardware here, which happen to roughly cancel out. First, I projected slower growth of human investments in hardware R&D than in software R&D. Second, I model physical capital (which AI can’t replace) as having an important role in hardware R&D but not in software R&D.

Summing up

Incremental AI automation will increase g(FLOP/$) as we cross the effective FLOP gap. This effect is smaller than the effect of rising human inputs until AI has automated ~25% of cognitive tasks (though the bottlenecks considered in the next section will increase this % somewhat).

More generally, as we cross the effective FLOP gap, g(FLOP/$) depends on:

1. The returns to hardware R&D, quantified by the parameter r.
2. The rate at which inputs to hardware R&D grow. There are two sources of growth:
    a. Increasing number of people and physical capital used in hardware R&D.
    b. Increasing fraction of cognitive tasks done by AI, and an increasing number of AIs doing each task.

This completes my discussion of the effect of incremental AI automation on FLOP/$. The next section discusses the effect on $ on FLOP.

$ on FLOP

In section 4 I analysed the impact of rising AI investments on **$ on FLOP**. I guessed that, after "wake up", $ on FLOP would initially grow at a rate of ~97% as we ramp up the fraction of global FLOP used on the largest training run, and then at ~22% after this when we're just expanding global chip production.

This section extends this analysis by incorporating the impact of incremental AI automation on $ on FLOP.

My approach here is to:

1. Use the same task-based model to forecast the effect of AI automation on Gross World Product (GWP) as I previously used to forecast its effect on software R&D and hardware R&D.
    a. This will tell us GWP in each year as we cross the effective FLOP gap and AI automates more and more cognitive tasks.
2. Assume that any acceleration in GWP growth accelerates growth in all economic sectors to the same degree. In particular, **the increase in g($ on FLOP globally) equals the increase in g(GWP)**.
    a. For example, suppose AI automation causes GWP growth to be 5% in some year rather than 3% - an additional 2% growth. Then I'll assume that g($ on FLOP globally) is 2% larger than I was previously assuming: 24% rather than 22%.
    b. This is a conservative assumption. AI automation of the economy will probably be disproportionately directed towards manufacturing more chips (more fabs, more chips per fab), given the large demand for FLOP that will exist.
        i. Importantly, the FTM **does** assume that AI automation will be disproportionately focussed on software and hardware R&D. Only on this

third component, $ on FLOP, am I making this conservative assumption.[164]

c. For more detail see this appendix.

I'll now say a bit more about #1, the effect of AI automation on GDP.

How I'm modelling the effect of AI automation on GWP

Just like I did for hardware and software R&D, I model incremental AI automation of the economy as a continuous transition between a world where cognitive tasks are performed by humans (world 1) to a world where they're performed by AIs (world 2).

The equations here are the same as for hardware R&D, in that they include a constant fraction of tasks α performed by physical capital. GWP is given by:

$$Y = K_g^{\alpha} L_g^{(1-\alpha)} \quad \text{(1') - humans do all cognitive tasks}$$

$$Y = K_g^{\alpha} C_g^{(1-\alpha)} \quad \text{(2') - AI does all cognitive tasks}$$

$$Y = K_g^{\alpha} L_g^{(1-\alpha)(1-f)} C_g^{(1-\alpha)f} \quad \text{(3') - AI does fraction f of cognitive tasks}$$

Y gives GWP in each year;[165] K_g, L_g and C_g give the amount of (physical) capital, human labour and 2020-FLOP used to produce goods and services (i.e. GDP) that year.

Equation (1') is the standard Cobb Douglas formula for GDP. Each time you double the quantity of labour (L), GDP (Y) doubles $(1 - \alpha)$ times. And similarly, each time you double the quantity of capital (K), GDP (Y) doubles α times.

Equation (2') simply replaces **number of human workers** with **2020-FLOP**, indicating that AI has fully automated the cognitive tasks previously done by humans.

Equation (3') uses the task-based Cobb Douglas model to allow for a continuous transition between (1') and (2').

[164] In other words, we are imagining that for each of the three components of 2020-FLOP ($ on FLOP, FLOP/$, 2020-FLOP per FLOP) there is an equivalent sub-sector of the economy (chip manufacturing, hardware R&D, software R&D). FTM assumes AIs are disproportionately focussed on the latter two areas but not the first. While advanced AIs are heavily concentrated on improving software and chip design, they're *not* concentrated on building new fabs and expanding the capacity of existing fabs. Of course, the sharp division between hardware R&D and chip manufacture the model makes here is not entirely realistic; the two will often merge together in practice like when a new fab must be built to manufacture a new type of chip.

[165] If growth is fast, Y can increase significantly over the course of a single year. To account for situations like this, there is a more precise definition of Y: Y gives the GWP that *would* be produced *if* output remained constant for 1 year. Mathematically, Y = ($ value of goods and services produced per second) * seconds in a year.

α turns out to be the fraction of GDP paid to capital, and $(1 - \alpha)$ is the share paid to cognitive labour. The actual share of GDP paid to labour in developed countries is ~0.65 and ~0.5 globally, but this includes physical labour as well as cognitive labour.[166] I'll assume the share going to cognitive labour globally is 0.5, and so use $(1 - \alpha) = 0.5$.[167]

$$Y = K_g^{0.5} L_g^{0.5} \qquad (1')$$

$$Y = K_g^{0.5} C_g^{0.5} \qquad (2')$$

$$Y = K_g^{0.5} L_g^{0.5(1-f)} C_g^{0.5f} \qquad (3')$$

The conditions under which tasks are automated are the same as for the task-based models I'm using for software and hardware R&D. A task is automated when i) we've done a training run large enough that AI can perform the task,[168] and ii) we have enough runtime 2020-FLOP to run enoughs AIs to replace all humans at the task.[169]

There are the same feedback loops as before, as more $ on FLOP causes more tasks to be automated **and** more AIs to perform each task.

(A') Bigger GWP → more $ on FLOP → more 2020-FLOP **in largest training run → more tasks automated** → bigger GWP

(B') Bigger GWP → more $ on FLOP → more 2020-FLOP **for running AIs → more AIs per automated task** → bigger GWP

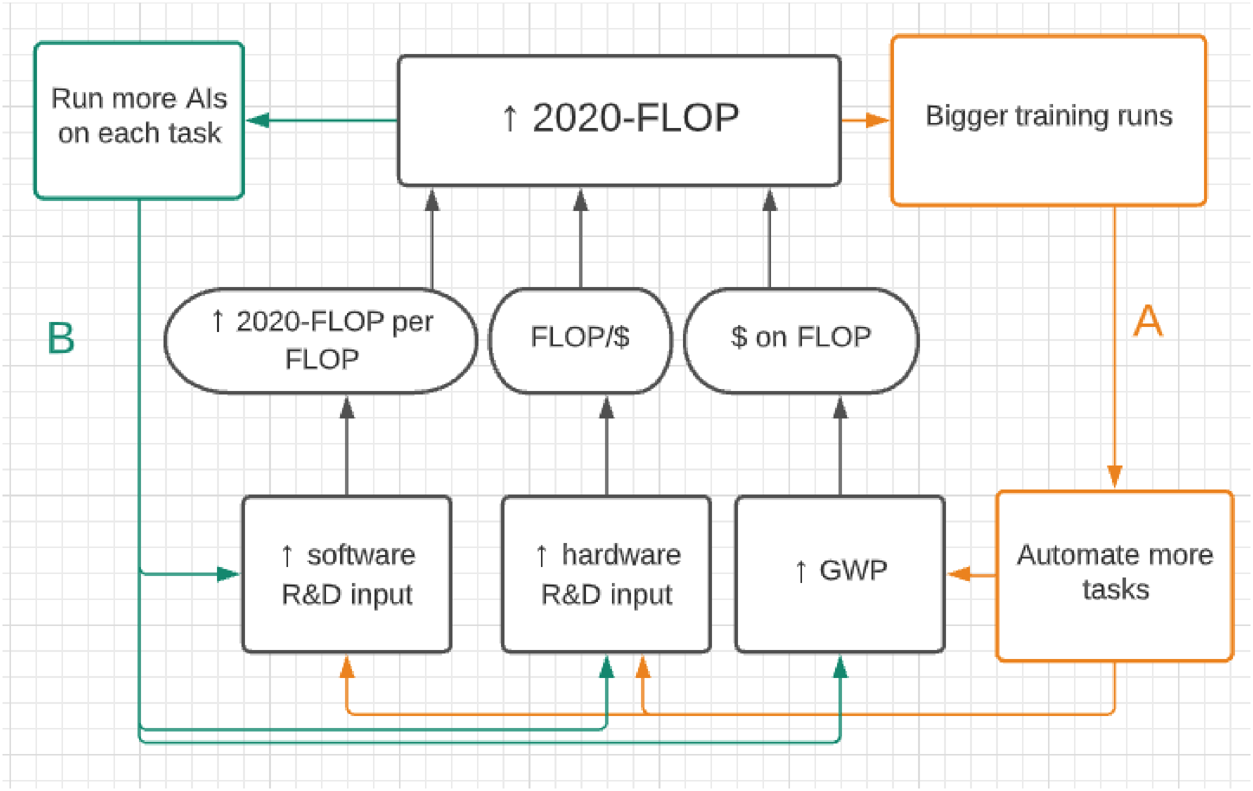
We can show these feedback loops alongside those for hardware and software.

[166] I'm using a simplistic model that *completely ignores tasks that require physical labour*. Each task is either done by physical capital or it's a cognitive task that's initially performed by humans and later performed by disembodied AI. I discuss this in this appendix.

[167] Though 0.5 is too high for world GDP, I actually care more about the share of cognitive labour in the semiconductor industry: that's what's relevant for **$ on FLOP**.
My thinking more generally is that cognitive labour is economically more valuable than physical labour by a wide margin, and so the share of GDP paid to cognitive labour should only be slightly lower than that paid to labour in total. Importantly, even jobs involving "manual labour" have a very significant component of cognitive labour to them. E.g. a plumber needs to figure out which changes to make and know how to make them; their distinctive skills mostly relate to these cognitive abilities rather than in their body's ability to execute particular physical movements when instructed to do so by the brain.
Empirical research could inform a better estimate of this parameter. You could look at the wages paid to various jobs in the US economy, and estimate the extent to which each job could in principle be done remotely (and so is purely cognitive) vs requires physical labour.

[168] As before, I adjust the Bio anchors estimate of 2020-FLOP training requirement for performing 100% of cognitive tasks (AGI) to estimate the 2020-FLOP needed to train AI that performs only x% of cognitive tasks for 0 < x < 100. How widely distributed these thresholds are in FLOP space is very important, and informed by these considerations.

[169] As before, I adjust the Bio anchors estimate of the 2020-FLOP/s needed to run AGI to estimate the 2020-FLOP/s needed to run AI that performs only x% of cognitive tasks for 0 < x < 100.



[170]

Quantitative implications of the model

What does this model imply about how GWP growth changes over time? Equation (3') implies that the contribution to GWP growth from AI equals 0.5 * f * g(2020-FLOP). (Recall f is the fraction of cognitive tasks automated by AI; it increases over time.) Assuming g(2020-FLOP) = ~100%, this equals ~f * 50%.

With full automation (f = 1), this implies GWP growth is 63%, doubling roughly every year.[171] [172]

We get "explosive" GWP growth of >30% from AI when **f = 0.6**, i.e. with AI automates 60% of cognitive tasks. However, f will be higher once we take into account bottlenecks, as we'll do in the next section.

Even so, if we quickly transition from a world where f < 0.1 to one where f > 0.8 then GWP could quickly go from doubling every ~20 years to doubling every ~2 years. In other words, there could be a fast takeoff according to the GWP doubling metric discussed above.

[170] Link to diagram. The diagram doesn't show inputs of human labour and physical capital to R&D and GWP. It only shows the AI inputs for simplicity.
[171] e^0.63 = 1.9.
[172] By the time f = 1, g(2020-FLOP) is much higher and so this model will predict a faster GWP growth in practice. On the other hand, the model omits certain bottlenecks that will make growth slower.

When does the effect of AI automation on **g($ on FLOP)** become more significant than rising human investment? In section 4 I guessed that rising human investment would drive g($ on FLOP) = 22%. The contribution of AI to g($ on FLOP) is the same its contribution to GWP growth: f * 50%. This exceeds 22% when **f = 0.45**.

By contrast, we estimated AI automation would dominate human investment in software and hardware R&D when f = 0.25. So it seems like AI automation will dominate rising human investment for g(2020-FLOP per FLOP) and g(FLOP/$) before g($ on FLOP).[173]

I analyse why the Full Takeoff Model (FTM) can easily predict fast takeoff in GWP, in the context of what generic growth models say about takeoff speed, in this appendix.

I'm not modelling AI automation of generic R&D

The FTM does *not* include AI automating generic R&D and thereby causing a productivity explosion.[174] This means I'm modelling the role of AI in producing goods and services but not in developing new technologies (other than those relating to software and hardware).

If I included this, it would increase the impact of AI automation on GWP and make takeoff faster. However, perhaps not *that* much faster: I believe the economic effects of AI automation via generic R&D will initially be smaller than its effects via goods and services.[175]

[173] What's driving the difference here? Firstly, I've assumed that capital is more important to GWP than to software or hardware R&D inputs. (Capital does 50% of GWP tasks, 30% hardware tasks, 0% of software tasks.) AI automation doesn't (directly) affect this capital component, so has smaller effects on GWP than on hardware and software R&D inputs. Secondly (and less importantly), I assume human inputs grow at 22% for $ on FLOP vs only 17% for hardware R&D. This means there's a lower bar for AI automation to dominate inputs for hardware, compared with $ on FLOP.

[174] This is imagined in the Cold Takes description of PASTA.

[175] Why? In short, because "doubling inputs to goods and services" immediately doubles GDP while "doubling inputs to R&D" only doubles the *rate of tech progress*, which takes many years to actually translate to a doubling of GDP.

In more detail: Imagine there are two sources of GWP growth: more production inputs and more R&D inputs. We want to compare using AI to increase production inputs via using AI to increase R&D inputs. There are two reasons to think the effect on GWP would be bigger and quicker via increasing production inputs. Firstly, data suggests that each doubling of R&D inputs causes *less* than 1 doubling of TFP; but in standard models doubling production inputs doubles GWP. This implies that doubling production inputs has a bigger effect on GWP. Secondly, to double GWP via production you only need to double *annual* production inputs. But via R&D what matters is *cumulative* R&D inputs, which are harder to double (even if your annual inputs instantaneously doubled, it would take a while for cumulative inputs to double). Combining these two points: to double GWP via production you need to double *annual* production inputs, but via R&D you need to *more than* double *cumulative* R&D inputs.

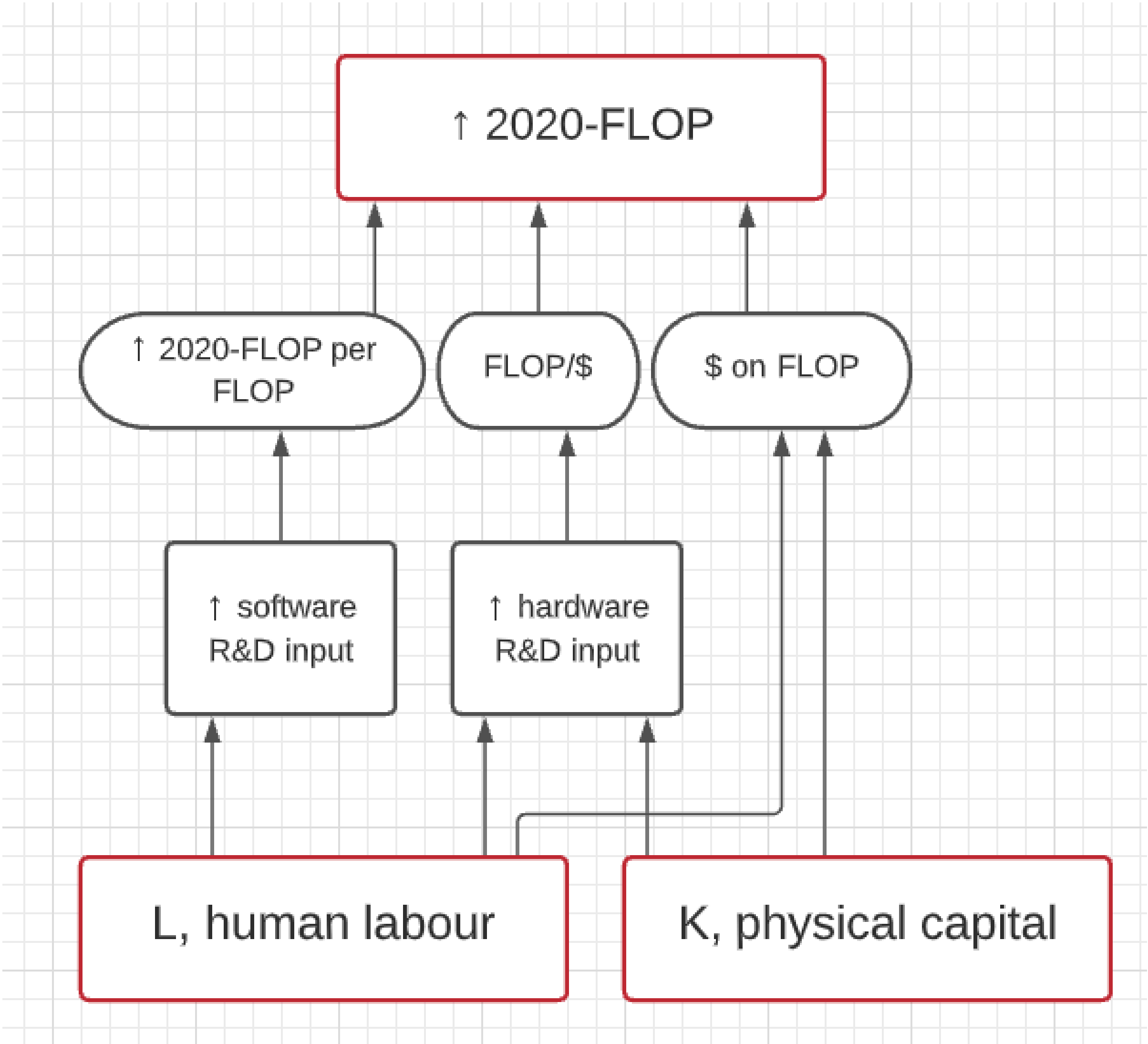
There are some good reasons to think increasing production inputs would have smaller effects than I'm claiming. i) Past a certain point people don't want more of the same goods and services, they want new types of good; so increasing production inputs won't help unless you've used R&D to invent new goods. ii) More technological, social and regulatory barriers to automating the provision of goods and services than to automating R&D. I discuss these further in TODO.

The takeoff model so far

The takeoff speeds model so far can be summarised as follows:

- We're forecasting the *calendar time* to cross the effective FLOP gap.
  - effective FLOP gap = How much bigger does your training run need to be to automate 100% of cognitive tasks rather than just 20%?
- This depends on the size of the effective FLOP gap and on how quickly we're able to increase the 2020-FLOP used in the largest training run.
- 2020-FLOP has three components such that:
  g(2020-FLOP) = g($ on FLOP) + g(FLOP/$) + g(2020-FLOP per FLOP)
- Section 4 analysed the effects of fast rising human investments on each of the three components. It can be thought of as estimating the effect of rising human labour and physical capital inputs on each component.[176]

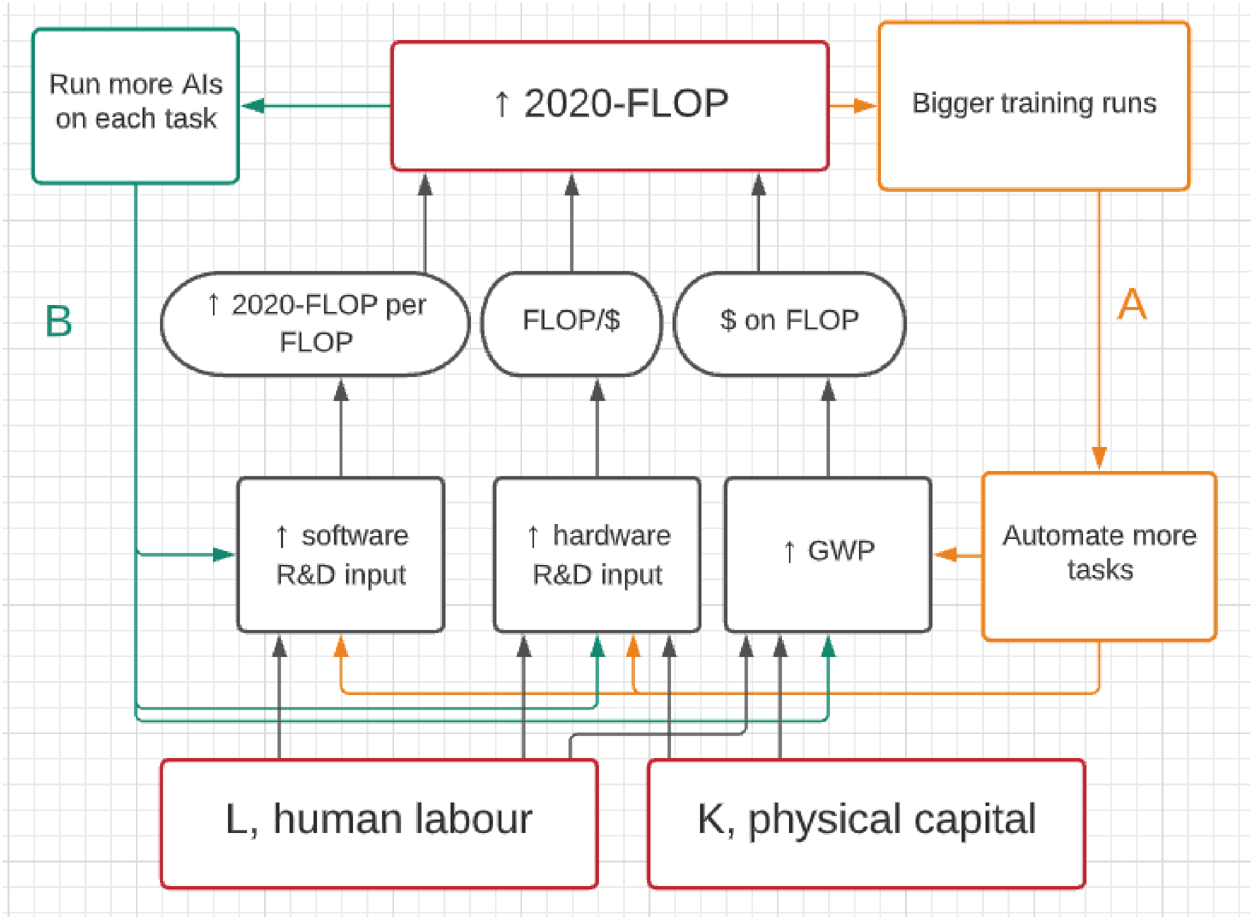
[176] Section 4 estimated the growing input as measured in real $, rather than separately estimating the growth of labour and capital. The simplest assumption is that real inputs of both labour and capital are growing at this same rate.



[177]

- This section analysed the effect of AI automation on each of the three components. As 2020-FLOP increases, we automate more tasks and have more AIs performing each task. These AI inputs are combined with human labour and capital inputs for each component.

[177] Link to diagram.



[178]

- The growth of each component accelerates as we cross the effective FLOP gap and automate more tasks. By the time we have fully automated all the cognitive tasks for any given component, it doubles in a year or much less.
  - We found that, for software and hardware R&D, the effects of AI automation are smaller than those of fast rising human investments (L and K) until ~25% of cognitive tasks have been automated. For $ on FLOP the threshold was ~45%.
  - These thresholds will be somewhat higher after we take bottlenecks into account in the next section.
  - Once we've reached these thresholds, the feedback loops in green and orange have become more significant in increasing g(2020-FLOP).

New metrics

The modelling introduced in this section, and the additional assumptions we must now make, allow us to calculate some other metrics of takeoff speed that were discussed in section 2.

- **Successive GWP doubling times.**

[178] Link to diagram.

- Equation (3’) calculates GWP based on the inputs of capital, (human) labour, and 2020-FLOP. As AI automates more tasks the importance of labour falls, that of 2020-FLOP rises, and GWP growth accelerates.
- If we transition quickly from world 1 to world 2 GWP growth can quickly go from its current level (~3%) to much faster (>60%). The speed of this transition depends on the effective FLOP gap and on the average g(2020-FLOP) as we cross the gap.
- The sensitivity analysis will report the largest ratio between successive GWP doubling times during this transition. Remember I’m calling ratios > 4 a “fast” takeoff, ratios <=2 a “slow” takeoff, and ratios between 2 and 4 a “medium” takeoff.

- **Time from “AI could readily automate x% of tasks” to “AI could readily automate y% of tasks”.**
    - To determine when tasks can be automated I make assumptions about i) the 2020-FLOP training requirements for each task and ii) the 2020-FLOP/s runtime requirements for each task.
- **How many AGIs can we run?**
    - The training and runtime requirements for the final task are the highest and can be loosely interpreted as the AGI training and runtime requirements.[179]
    - The model calculates the largest training run in each timestep, so can calculate when we first train AGI
    - The model also calculates the total quantity of 2020-FLOP/s in each timestep, so can calculate how many AGIs we could run in each timestep.
    - This allows the model to calculate the first timestep in which we can train AGI and run X AGIs, for any X.
        - In fact it can calculate the number of AI that automates x% of tasks, for any x. AGI is the special case when x = 100.
    - I use this to calculate the metric **time from AI that could automate 20% of cognitive tasks to when we can run 10 billion AGIs.**
        - My startpoint of “automating 20% of cognitive tasks”, requires the training 2020-FLOP to train AI that can automate 20% of cognitive tasks and the runtime 2020-FLOP to run enough AIs to replace human workers at those tasks.
-
- **Cognitive output.**
    - As AI automates more cognitive tasks the model calculates total cognitive output in each timestep.
    - This is simply the output on tasks *not* performed by capital. The formula is:

$$Cognitive\ output\ =\ L_g^{(1-f)} C_g^{\ f}$$

[179] Those requirements are sufficient to train an AI and run it to do any task as well as a human worker. In practice, AIs with lower requirements are used to perform most tasks in the model as this is more efficient. But you could in principle use a more expensive system to perform every task.

  - This notion embraces the complementarity between human and AI cognitive labour by tracking the output from both of them combined. It aims to side-step the complementarity between cognitive labour and physical capital in order to get a metric of AI capabilities that is independent of physical capital bottlenecks.
  - The notion also avoids leaning on any specific and arbitrary capability level like "AGI"; total cognitive output is the result of AI systems of varying levels of generality and capability, some of which may be tools and some of which may be agentic.
  - The units of cognitive output are **remote human equivalents.** More precisely, "How many remote human workers would be needed to generate the same economic value per day as we're getting from the combined cognitive output of humans and AIs?"[180]
  - I use this to calculate the metric: **time from AIs being a 2X multiplier on human cog output to being a 10X multiplier.**

Significant AI automation need not happen before AI risk

If you believe that the most likely path to AI causing existential risk is via accelerating economic growth, or having vastly superior cognitive capabilities, then there will be significant effects from AI automation before this happens. In this case, the dynamics discussed in this section are potentially extremely important as they precede x-risk.

On the other hand, if you think it's likely existential risk comes from pre-AGI systems that perform some tasks excellently but cannot perform most tasks, the analysis of this section is much less relevant. AI won't be sufficiently capable and general to accelerate software R&D, hardware R&D, or GWP before it poses x-risk.

My own view is that AI would probably need very significant degrees of autonomy in a very wide range of cognitive tasks to pose existential risk via power-seeking. For example, it would need to be able to perform ~all tasks in one or more broad areas like AI R&D, social manipulation, hacking and business/military strategy. And my guess is that this in turn would require AI to perform a large percentage of total cognitive tasks, probably >70%.[181]

Summing up

This section analysed the effect of AI automation on the speed crossing the effective FLOP gap, g(2020-FLOP). I did this separately for each of the three components – g(2020-FLOP per

[180] So "cognitive output = 1 billion remote human workers" means "if we continued to produce cognitive output at the current rate for a year, then the total cognitive output produced would have the same value as that produced by 1 billion remote human workers working for 1 year".
[181] In particular, *fully* automating any of these high-level tasks requires many capabilities which will also help to fully automate or partially automate many other high-level tasks.

FLOP), g(FLOP/$) and g($ on FLOP) – using the same task-based model for each. Unsurprisingly, AI automation increases these growth rates significantly as we cross the effective FLOP gap. By the time AI has automated all cognitive tasks in a component, it doubles in months or faster.

When does the effect of AI automation become significant, compared to the rising human investments discussed in section 4? In the model used here, the answer depends on the component in question. For g(2020-FLOP per FLOP) and g(FLOP/$), i.e. for software and hardware progress, AI automation becomes significant when roughly ~25% of cognitive tasks have been automated; for g($ on FLOP) it was roughly when ~45% of cognitive tasks have been automated.

These AI automation dynamics are less relevant for takeoff speeds if you think AI will pose an existential risk before it automates a significant fraction of tasks.

Modelling AI automation required additional assumptions about the compute needed to train and run AIs that can perform x% of cognitive tasks for 0 < x < 100. These additional assumptions allow us to calculate the way in which the % of tasks performed by AI increases continuously over time.

The result model can calculate metrics of takeoff speed relating to GWP, the *number* of AIs, and the total cognitive output of AIs and humans.

What is the bottom line here for takeoff speeds? In this framework the values of takeoff speed metrics – e.g. how long from 20% automation to AGI – depend on the training and runtime requirements of pre-AGI systems, and how these combine with the rising human investments from section 4 and the bottlenecks described in the next section. Unfortunately, I'm not aware of a simple analytically tractable way to estimate them, and so this section did not make new central best-guess estimates of takeoff speed.[182] My approach instead is to conduct a sensitivity analysis on a simulation of the model, which I'll present in section 7.

The main body of the report continues in a new doc.

Appendices

Appendices to add:

- The shape of the task distribution over log(FLOP); maybe there's a tail of tasks?
- Are we assuming "AGI is one big model" vs "lots of little models"?

[182] Though I do take the numbers in section 4 to be very rough best guesses even accounting for AI automation. That's because the numbers in section 4 are too high for when we start crossing the effective FLOP gap, and too low after AI automation kicks in.

Literature on brain size - IQ correlations

Edit: a new meta analysis has been released, which comes down a little more conservative than I did here.

My current guess is that a **10% more brain volume → 4.5 more IQ points**.

I skimmed three studies.

1. Gignac & Bates (2017) is a large recent meta analysis.
    a. Its headline figure is a correlation of **0.29** (95% CI = 0.24, 0.33) between brain volume and IQ.
    b. It found the result depended on the accuracy of the IQ measurement used. 'Fair', 'good', and 'excellent' measurements had correlations of 0.23, 0.32 and 0.39.
        i. It claims the adjustments it makes here are probably too small, as empirical measurements tend to be less reliable than normative samples used to rate measurement procedures.
    c. At a glance, it doesn't seem to discuss confounders much. Health and education seem like possibilities. I don't know how carefully the object level studies controlled for this.
    d. If the above two factors cancel (under-adjusting for mismeasurement of IQ and not including confounders) then the estimate of correlation due to causation is 0.3 - 0.4. (I'd guess this cancelling assumption leaves the correlation too high.)
    e. We can use the correlation to estimate that **a 10% bigger brain increases intelligence by 4.5 - 6 IQ points**.
        i. A correlation of z between X and Y means: increase X by 1 standard deviation → increase Y by z standard deviations. (This assumes the correlation is causal.)
        ii. I couldn't see data about the standard deviation of brain sizes for the study participants; but this seems to be ~10% in the general population. So 1 standard deviation of brain size = 10% bigger brain.
        iii. So this study is saying a 10% bigger brain → 0.3 - 0.4 standard deviations of IQ, or 4.5 - 6 IQ points. (A standard deviation of IQ is 15 IQ points.)
    f. I'd guess this estimate is too high due to the seeming lack of effort to adjust for confounds, but I'm not confident about this.
2. Nave et. al (2018) is (I believe) the largest empirical study to date, bigger than all previous investigations combineed (N = 13,608).
    a. The most relevant figure here is a correlation of **0.25**. (They get this after including various confounders and trying to adjust for mismeasurement of IQ.)
    b. This paper includes a few confounds (social deprivation, place of birth, height, genetics) and did other robustness checks.
    c. I expect they have under-adjusted for mismeasuring IQ, for the same reasons as Gignac & Bates (2017).
    d. We can use the correlation to estimate that **a 10% bigger brain increases intelligence by 4 IQ points**.

        i. This is just as above, except that the standard deviation of brain volume in this study was 9.3%.[183]
        ii. The correlation means that a 9.3% larger brain → 0.25 standard deviations of IQ, or 3.75 IQ points. (Again assuming the measured correlation is causal.)
        iii. So a 10% larger brain → 4 IQ points.[184]
    e. I don't see a strong reason to think this is biased in either direction overall.

3. This 2019 sibling study (N = 1022) finds a correlation of **0.18** within families and **0.33** overall.
    a. They of course control for family environment, which will include health and education.
    b. Again, I expect they have under-adjusted for mismeasuring IQ. Indeed, they find strong evidence that their IQ tests are less reliable than they assume in their adjustment.[185]
    c. We can use the correlation to estimate that **a 10% bigger brain increases intelligence by 3.5 IQ points.**
        i. This is just as above, except that the standard deviation of brain volume in this study was 8%.[186]
        ii. So a 10% larger brain → 3.5 IQ points.[187]
    d. I think this is too low, due to under-adjusting for the mismeasurement of IQ.

To summarise the above evidence on the effect of a 10% bigger brain:

- Gignac & Bates (2017): 4.5 - 6 IQ points. I'd guess this is too high, but I'm not sure. I don't have a good understanding of this sprawling meta analysis.
- Nave et. al (2018): 4 IQ points. This doesn't seem biased in either direction and is the biggest study out there.
- The sibling study: 3.5 IQ points. I think this estimate is too low, so I see this as easily consistent with the true effect being 4 - 5 IQ points.

Overall, I'd guess a **10% more brain volume → 4.5 more IQ points**. I've adjusted slightly upwards from Nave et. al (2018) due to Gignac & Bates (2017). This corresponds to 1 standard deviation of brain size → 0.3 standard deviations of IQ.[188]

[183] See figure S3 in the supplementary material.
[184] 3.75 * 10/9.3 = 4.03.
[185] For reasons that are unclear, the correlation between Verbal and Performance observed in the MCTFR does not seem consistent with such high reliability. We nevertheless used a conservative value of 0.82 in calculating the disattenuated associations; the assumption of a lower value would lead to larger apparent effects.
[186] Figure S1 finds brain volume mean (standard deviation) is 1270 (101) for males, and 1100 (88) for females. 1270/101 = ~1100/88 = ~ 8%.
[187] 10% bigger brain → 0.18 * 10/8 = 0.23 standard deviations of IQ = 3.45 IQ points.
[188] Assuming 1 standard deviation is a 10% increase in brain size and a 15 point increase in IQ.

Ramp up will be bottlenecked by supply of FLOP

In this regime of fast rising investment, I expect the primary bottleneck for investment is not going to be willingness to pay, but instead supply constraints.

For example, if someone wanted to spend $100 billion on AI chips today, they simply couldn't (NVIDIA data center revenue in 2022 was ~$10b, and they're a large fraction of the AI chip market). If the person insisted on spending that much, they'd be forced to buy non-AI chips that are much less well suited for AI. In this example, the bottleneck on "get a quantity of AI chips that would be worth $100 billion at current prices" is not willingness to pay but instead how quickly chip manufacturers can scale up production of AI chips. Large willingness to pay can expedite this process, but only to some extent.

Similarly, if someone wanted to spend $100 billion on AI software researchers today, the key bottleneck would be talent availability. If they wanted to hire good quality people, they'd be limited by the time it takes to attract and train good people to grow a small field. In this example, the bottleneck to "get a quantity of AI software workers that would cost $100 billion at today's prices" is how quickly you can attract and train high quality talent.

So I analyse how quickly we can ramp-up AI investments by focussing primarily on the supply side.[189]

I ignore the rising price of inputs to AI development after wake up

One factor I (try to) put to one side is the likelihood that the cost of inputs to AI investment will rise significantly as demand far outstrips supply. If the actual number of AI software workers remains constant, but their salaries have all doubled, I don't want to say that software investment has doubled. Instead, I would say that *real* software investment has stayed constant. The numbers in this report should all be interpreted in this vein as referring to the growth of *real* inputs to AI, measured in the number of quality-adjusted workers, physical capital and computer chips. In this way, I (try to) sidestep the way in which high demand will drive up the price of real AI investments.[190]

[189] I don't think supply places a strict bottleneck on annual AI investment. If there's higher willingness to pay on the margin, that will somewhat increase real AI investment by inducing more people to abandon otherwise lucrative activities. So demand does make a difference on the margin. But, past a certain point, that marginal difference is small and you approach hard limits in terms of (e.g.) the time it takes to find and train additional people, and the limited number of people who have the expertise to deliver that training.

[190] I feel confused about whether this move will lead to unrealistic predictions about the things I care about. E.g. I will end up talking about "FLOP/$" numbers that, because I'm ignoring the effect of high demand on prices, are predictably too low. But what I ultimately care about is the total FLOP available in the world, not the amount that is paid for; I'm only using "FLOP/$" as a measure of hardware progress. The question is whether my forecast of the available FLOP is distorted by not explicitly modeling this factor.
I'll forecast "$ on FLOP" numbers that are predictably too low for the same reason, and again it's unclear how much this matters.

Bold assumption to make the analysis somewhat tractable

One thing I'll need to forecast is the growth of FLOP produced each year globally.[191] By analogy with section 3, I calculate this as:[192]

FLOP per year = $ on FLOP per year * FLOP/$

I forecast each of the two components by mapping them to two sources of growth in FLOP production. The two sources are:

1. **More chips**. Increases in the *number of chips produced per year*. E.g. more fabs, more production lines within each fab.
2. **Better chips**. Increases in *FLOP per chip.* E.g. smaller node sizes, specialised chip designs.

In reality, I suspect these two sources can't always be cleanly separated.[193]

My bold assumption is that "more chips" corresponds exactly to more $ on FLOP, and that "better chips" corresponds exactly to more FLOP/$. In particular, I assume g($ on FLOP per year) = g(number of chips produced per year), and that g(FLOP/$) = g(FLOP per chip).[194]

Then my strategy is to:

- Forecast g($ on FLOP per year) via forecasting how quickly we will expand chip production after "wake up".
- Forecast g(FLOP/$) by assuming that hardware R&D has driven historical growth in FLOP/$, and forecasting inputs to hardware R&D after "wake up".

Perhaps the right thing to do is to replace "FLOP/$" with "FLOP/s per chip" and replace "$ on FLOP" with "number of chips", to avoid the reference to "$".

[191] What do I mean by "total FLOP produced each year"? Take all the chips produced over the course of one year, run them all non-stop for one year, and ask: *How many FLOP did you do*? This is what I mean. A more precise statement would be "the annual FLOP capacity of 1 year's chip production". I use this unit so we can talk about FLOP rather than FLOP/s. This is useful because I'm ultimately concerned with how many FLOP we have available for the largest training run in each year, and only indirectly concerned with FLOP/s.

[192] In section 3 I calculateed FLOP for the largest training run = $ on FLOP for the largest training run * FLOP/$. (I'm putting aside software progress for now.)

[193] Imagine we build a new fab with smaller node size, and compared its FLOP production to an old fab. We ask: *Is the new fab's greater FLOP production due to better chips, or due to expanded production?* The new node size may use a completely different kind of chip that doesn't map cleanly to the old chip. As a result, it may be ambiguous whether the new fab has more chips, relative to the old fab. So it's ambiguous to what extent the greater FLOP production of the new fab comes from more chips vs better chips.

[194] I.e. I assume that the price of chips is constant. People create better chips so that they can sell more chips at the same price, not to increase the price per chip. I think this assumption is more accurate over long timescales than short timescales. Over short timescales, you might be able to sell better chips for more. But in the long run, the price of the most recent cutting edge chips may be constant at ~$10,000 per chip.

Source of growth	**Equivalent quantity in takeoff framework**	**Cause of growth**
More chips	g($ on FLOP per year)	Build more fabs and bigger fabs
Better chips	g(FLOP/$)	Hardware R&D

I discuss why my bold assumption might be wrong, and how a more realistic assumption might change the results, in the following footnote.[195] This part of the framework feels conceptually confused, and I'd welcome suggestions for improvement.[196]

Details about assumptions of the Full Takeoff Model

This appendix gives some additional details on assumptions made by the Full Takeoff Model (FTM). For additional information you could:

- *See the FTM's behaviour for your chosen deterministic inputs here, including justifications for my preferred values.*
- *Inspect the functionality of this old spreadsheet version of the FTM.*
- *Ask Epoch for the most up to date python code.*
- *Read this concise mathematical description of the FTM (courtesy of Epoch).*

Accounting for the "stepping on toes" effect when estimating the returns to hardware R&D

Suppose you invest $X in R&D. If there's a stepping on toes effect, then the *effective* R&D input is only X^lambda, lambda < 1. Some effort is duplicated (or otherwise wasted due to the difficulty of parallelising R&D effort). So doubling investment only increases *effective* inputs by 2^lambda.

[195] I suspect my assumption gives too much credit to "more chips", and not enough credit to "better chips", in explaining the historical growth of FLOP production. I assume that all the increase in [$ on FLOP] is due to "more chips". But the price of chips has probably increased over time, so that some of the increase is due to "better chips". How would giving more credit to "better chips" change the results? Firstly, it would mean giving historical hardware R&D more credit for the growth in FLOP production, and so increase my forecast of how quickly hardware R&D will increase FLOP production after "wake up". Secondly, it would slightly lower my estimate of how quickly we'll be able to expand chip production after "wake up". I'm not sure how these effects would net out; my guess is that the first would be larger and that the net effect would be fairly small.
I assume that the only reason why [$ on FLOP] increases is because of *more chips;* but in reality I'd guess that it also increases due to *better chips*. People pay more for SOTA chips over time. This means that

[196] Repeating from an earlier footnote, perhaps I should replace "FLOP/$" with "FLOP per chip" and replace "$ on FLOP" with "number of chips", eliminating the reference to "$" entirely.

How does this affect our empirical estimate of the returns to hardware R&D?

Let g_I be the growth of R&D inputs, g_O be the growth of the relevant R&D output metric, lambda be the stepping on toes parameter and *r* be the returns to hardware. For hardware R&D increasing FLOP/$, we have historical observations of g_I and g_O and must infer lambda and *r* from the data. With the semi-endogenous growth model, if g_I grows at a constant rate, the equation linking these quantities is:

g_O / g_I = lamba * r (1)

Here's what happens in the main text. Let g_I_h and g_O_h be the observed historical growth rates of R&D inputs and FLOP/$. In the main text I assume lambda = 1 and then use (1) to infer r = g_O_h / g_I_h. I then make a hypothesis about the future growth of inputs after "wake up" g_I_f, and infer future growth of the FLOP/$ g_O_f. Mathematically, this is:

g_O_f = g_I_f * r = g_I_f * g_O_H / g_I_h

(g_I_h and g_O_h give historically observed growth of hardware inputs and FLOP/$; g_I_f and g_O_f are forecasts of the same quantities after "wake up".)

If instead I'd assumed some stepping on toes then I'd used (1) to infer r = g_O_h / (g_I_h * lambda). My estimate of *r* would increase by a factor 1 / lambda, as the growth of effective inputs have increased more slowly due to stepping on toes. Then I'd have made the same forecast about growth of inputs after "wake up" and inferred future growth of FLOP/$ as follows:

g_O_f = g_I_f * lambda * r = g_I_f * lambda * (g_O_H / g_I_h * lambda) = g_I_f * g_O_H / g_I_h

g_O_f is exactly the same. My estimate of *r* is higher in a way that exactly offsets stepping on toes. So stepping on toes doesn't affect the predicted growth of FLOP/$ after "wake up".

There is one significant caveat. Equation (1) assumes annual inputs are growing at a constant rate. If annual inputs start growing more quickly than they used to – like they will after "wake up" – things are more complex. In this case, a stepping on toes dynamic (lambda < 1) will increase the lag between the faster growth of annual inputs and the faster growth of the output. You can see this dynamic play out in this sheet. So the stepping on toes effect increases the lag between "faster growing hardware R&D inputs" and "faster growing FLOP/$" and so makes takeoff slower.

The diminishing returns to hardware and software become steeper over time; ideas become *increasingly* hard to find

By the time we reach physical limits of hardware, further progress is impossible. This corresponds to r = 0 (each doubling of cumulative R&D inputs causes 0 doublings of FLOP/$).

The FTM assumes r decreases towards 0 by a constant amount each OOM of FLOP/$ increase between now and the physical limit. E.g. if we've 6 OOMs from the physical limit and currently r = 2 then by the time FLOP/$ has increased by 3 OOMs, the FTM assumes r = 1.

This dynamic can capture the expectation that hardware returns should trend back to the average R&D returns across all sectors of the economy.[197] As long as the physical limits are assumed to be at or above 1e25 FLOP/$, introducing physical limits in this way doesn't significantly affect the results.

And the same model is used for software.

You can see the assumptions about these physical limits, and justifications, in the "additional parameters" tab here.

Full derivation of the equation for hardware R&D

It's the standard semi-endogenous equation, with a "stepping on toes" effect, and with two complications.

Here's a derivation of the equation ignoring "stepping on toes" and the additional complications.

[197] *Are ideas getting harder to find* estimates the average returns to the overall economy to be r = ⅓, much lower than the returns for hardware.

$$\dot{A} = \delta r(t) A^{\phi} \qquad \text{(Assumption 1)}$$

$$\dot{A}/A^{\phi} = \delta r(t)$$

$$\int_{-\infty}^{t} \dot{A}/A^{\phi}\, dt = \delta \int_{-\infty}^{t} r(t)\, dt$$

Let *R(t)* be the total researcher-years by time *t*:

$$R(t) \equiv \int_{-\infty}^{t} r(t)\, dt$$

$$\int_{-\infty}^{t} \dot{A}/A^{\phi}\, dt = \delta R(t)\, dt$$

$$A^{1-\phi}/(1-\phi) = \delta R(t) \qquad \text{Assumption 2 ensures } \varphi \neq 1$$

$$A^{1-\phi} = (1-\phi)\delta R(t)$$

$$A = (1-\phi)\delta R(t)^{1/(1-\phi)}$$

$$\boxed{A(t) = kR(t)^{m}, \text{with positive constants } k = [(1-\phi)\delta]^{1/(1-\phi)} \; and \, m = 1/(1-\phi)}$$

Notice this implies that g(A) = m * g(R). In the report I use different variable names, so let's stick with those of the report: g(O) = r * g(I).

Now to include stepping on toes in this set-up I alter the semi-endogenous equation in the standard way by adding a parameter lambda: A = delta * r^lambda * A^phi.

I then define $R(t) \; = \; \int_{-\infty}^{t} r(t)^{\lambda} dt$

This (I claim) doesn’t change the result of the above derivation. We again get A = constant * R^(1/(1 - phi)) = constant * R^r. So g(A) = r * g(R).

But my new definition of R implies g(R) = lambda * g(r). So g(A) = r * g(R) = r * lambda * g(r).

In the report's notation: **g(O) = r * lambda * g(I).**

Then two complications are added:

1. For hardware R&D (but not software R&D) I replace r^lambda with CES(r^lambda, K^lambda) where K is the physical capital used for R&D and r continues to be the number of researchers. This allows for physical capital to play a role in R&D, and potentially allows for physical capital to bottleneck R&D progress.

2. The returns to R&D r decreases towards 0 as we approach physical limits (more).

For a full implementation reach out to Epoch or check out this (messy!) sheet.

Aggregating one-time productivity gains from different sources

My approach is to quantify the individual effects mentioned, combine the relevant ones together, and maybe do a final adjustment based on my “gut”.

Productivity gains from different sources:

- Faster serial speed
  - Paul and Carl think a 1000X speed-up is possible
  - E.g. 1 AGI running for 1000 subjective years rather than 1000 humans working for 1 year each
  - I’m calling this **~10X**[198]
- No leisure / sleep: **3X** (people spend 8 hours a day working)
- Better motivation: **2X**
- Average vs top productivity
  - Among humans, average vs top productivity is >100X (global average income is ~$10k, most productive people can earn >$1m)
  - But all AGIs are as productive as the most productive AGI
  - So this naively gives AGI a gain of **~100X**

Multiplying these gains together gives 10 * 3 * 100 * 2 = 6,000.

One-off gains for AGI in R&D. I will exclude "average vs top productivity" as researchers are mostly close to the top global average productivity. That leaves me on 10* 3 * 2 = **60X**.

One-off gains for AGI in goods production. I don't think serial speed will apply very much in goods production. I also feel pretty suspicious of the “average vs top productivity” figure, in particular that AGIs could increase everyone’s productivity by 100X despite lacking the context of their jobs. This leaves me 3 * 2 = **6X**.

In the FTM these assumptions are combined with a starting estimate of the AGI’s runtime compute of 1e17 FLOP/s using 2020 algorithms. See rows 5 and 7 here.

Modelling $ on FLOP

Roughly speaking, I assume:

$ on FLOP for training run = GWP * fraction of GWP spent on FLOP * fraction of FLOP on largest training run

FLOP for a training run = $ on FLOP for training run * FLOP/$

[198] It turns out that you can derive the size of this effect from the size of the “stepping on toes” effect. If you quantify that effect in the usual way with lambda < 1, then a 1000X speed up increases productivity by 1000^(1-lambda). 10X gain corresponds to lambda = ⅔, which is roughly my best guess for lambda.

The quantities are forecast as follows:

- GWP: use the task-based model for how AI automation affects GWP
- Fraction of GWP on FLOP: section 4 analysis of growth of $ on FLOP globally
    - I guessed $ on FLOP globally might grow at ~22% after "wake up"
    - GWP growth is 3%, so this implies that the fraction of GWP on FLOP will grow at ~19%.
    - So my central estimate is ~19%.
- FLOP/$: grows due to hardware R&D. Section 4 analyses the effect of human inputs; section 5 analyses the effect of AI automation.
- fraction of FLOP on largest training run: analysed in section 4.

Annual production vs stock

The above equation ignores that we can use chips bought in previous years in training runs. In fact the equations used in the FTM are:

global FLOP year y+1 = global FLOP year y **+** GWP * fraction of GWP on FLOP * FLOP/$

FLOP for training run = global FLOP * fraction of FLOP used on largest training run

In essence, the above equation pretended that we produce all our chips from scratch each time-step, while these ones allow us to accumulate a stock of chips over time. Each year's production simply adds a little to that stock.

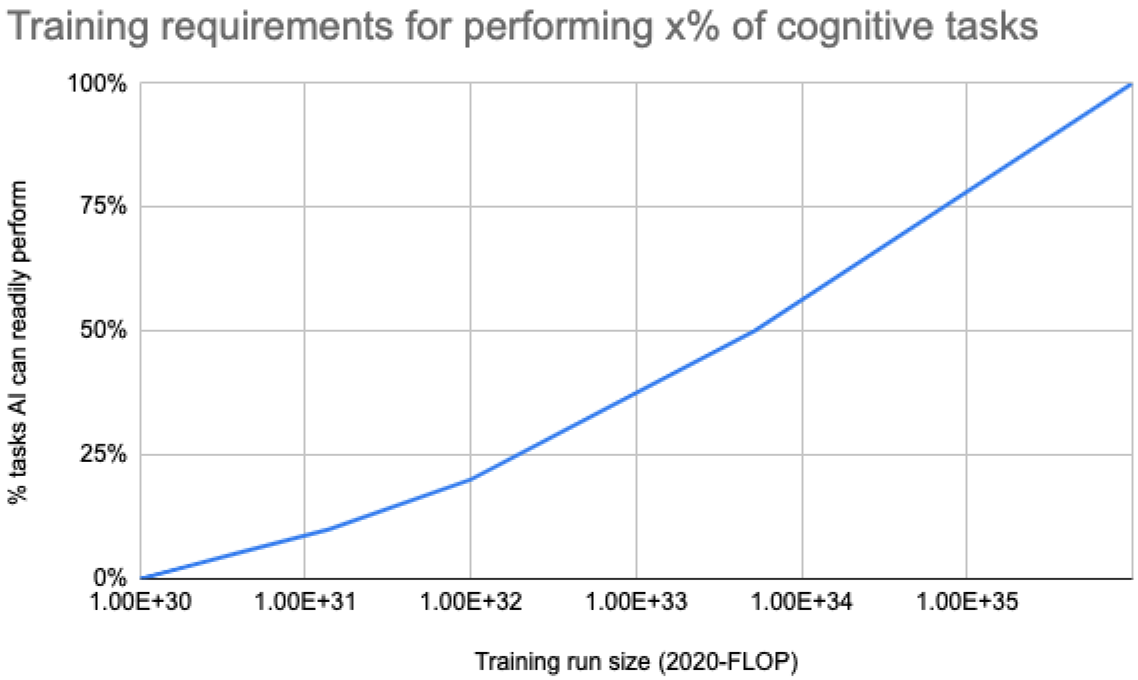
I do not explicitly distinguish between FLOP used for AI and other FLOP. I think this is a weakness of the FTM as it stands, which I explain more in this doc.

FTM assumes no tasks are done by physical labour

- The better model would have:
    - Some tasks done by physical labour, some by cognitive labour, some by physical capital
    - One process whereby AI automates cognitive labour.
    - A second process whereby robotics automates physical labour.
- But my model only includes the first process: AI automating cognitive labour. I ignore physical labour.

Calculating the training requirements for "AI can readily perform x% of cognitive tasks"

The Full Takeoff Model (FTM) makes assumptions about the training requirements for "AI that can readily perform x% of cognitive tasks" for all x. In other words, it makes an assumption about the full shape of the following graph:



The FTM calculates the full shape of the curve mechanically from just one input, the effective FLOP gap from 20% of tasks to 100%. The method, described mathematically here, creates a curve shaped like that in the picture above. In particular:

- Each additional OOM of training unlocks more tasks than the last.
  - This seems very likely as recently each OOM of training has seemingly unlocked very few tasks.[199] I'd expect a gradual transition with each OOM unlocking more tasks than the last.
- The effective FLOP gap from ~1% to 20% is half the effective FLOP gap from 20% to 100%.
  - My best guess would actually be that ~1% to 20% is roughly *as big* as 20% to 100%, or bigger.
  - But if we used that assumption, we'd have less flexibility in specifying the 20%-100% effective FLOP gap. E.g. suppose we wanted to say AGI is 1e30 FLOP and the effective FLOP gap is 4 OOMs. Then our assumption would imply that ~1% of tasks required training of ~1e22 FLOP. But we've already seen training runs of 3e24 FLOP, implying that today's systems can readily perform >>1% of cognitive tasks. The model would predict that AI could readily[200] add >>$500 brillion/year to GDP, in contrast to observed AI revenues.

[199] Annual revenues from AI are estimated at ~$10 - 100b. (E.g. here, here, here, here; I don't know how reliable these estimates are, or even their methodologies.) But performing even 1% of tasks would be worth ~$500b because about ~$50tr is paid in wages each year globally.

[200] Reminder: the phrase "readily" here indicates that i) it would be profitable for organisations to do the engineering and workflow adjustments necessary for AI to perform the task *in practice*, and ii) they could make these adjustments within 1 year if they made this one of their priorities.

  - Our actual assumption allows us to specify scenarios with fairly large effective FLOP gaps without this implied inconsistency. This is a reason to expect the actual effective FLOP gap to be smaller, than the maximum effective FLOP gap allowed by the model.

Different assumptions about the exact shape of the curve would change the bottom line somewhat. And perhaps some useful insight could come from thinking more carefully about implications of different curves, and testing out different possibilities (e.g. a log-normal distribution). But I suspect most of the 'action' is in the single scalar parameter I've pulled out – the effective FLOP gap – that describes how spread-out in FLOP space different tasks are.

How does physical capital changes over time in the FTM?

- Cognitive output is grows fast due to AI automation.
- Initially, this increases GWP a lot because cognitive tasks are important to GWP (they're paid a high fraction GWP).
- More GWP → more reinvestment → faster growth of physical capital.
- But after a while, the abundance of cognitive tasks reduces their importance to GWP (they're paid a much lower share of GWP). GWP is bottlenecked by physical capital. Further cognitive output growth doesn't affect GWP much at this point. So the reinvestment in physical capital stops rising: ~constant GWP → ~constant reinvestment → ~consant growth of physical capital.
- Eventually physical capital grows faster and faster because of tech progress
  - Higher TFP allows you to accumulate capital more quickly.
  - This takes unrealistically long in the FTM.
  - The reason it takes so long is that the FTM assumes TFP grows exogenously rather than modelling AI automation's effect on generic R&D; it takes many decades before physical capital is doubling every year.
  - In fact I expect it would take much less long that for billions of AGIs to design robot-factories that collectively self-replicate in a year.

How much does AI automation accelerate our progress through the effective FLOP gap overall?

We re-ran all the scenario analyses from this section, excluding the speed-up effect of AI automation. This allowed us to compare the takeoff speed with and without AI automation.

	Takeoff speed	
	Time from AI that could readily	Time from AI that could readily

	automate 20% economic tasks to AI that could readily automate 100%.		automate 20% R&D tasks to AI that could readily automate 100%.	
Scenario	*Including* AI automation	*Excluding* AI automation	*Including* AI automation	*Excluding* AI automation
Short, best guess	3.7	16.5	4.4	8.8
Medium, best guess	5.0	14.0	5.5	12.5
Long, best guess	15.7	63.6	18.0	56.8
Medium, aggressive	0.4	0.9	0.6	0.9
Medium, conservative	27.0	46.5	32.3	49.4

Overall, it looks like AI automation increases our speed crossing the effective FLOP by ~2.5X.

Am I assuming AGI will take the form of one unified system, or many narrow systems working together?

Early writing about AGI tended to assume that it would take the form of one system with general capabilities. Others have since suggested that AGI could instead consist of many individually narrow AIs which work together to *collectively* have general capabilities.

When developing this framework, I tried not to take a strong stance on this question. Ultimately, the mathematical form of the Full Takeoff Model (FTM) fits best with the latter view. The FTM has dozens of different tasks, which each have different training and runtime requirements. The most natural interpretation is that different AIs perform each task.

If you want to make the FTM consistent with the "one general system" interpretation of AGI, you could say that there's one AI that i) learns to do more tasks as we increase the size of our training runs, and ii) can match human performance at some tasks with OOMs less compute than other tasks. But this interpretation seems less natural to me.[201]

But then why do you talk about the "number of AGIs we could run" as if AGI was a unified system?

[201] In particular, how is AGI able to match human performance on some tasks with OOMs less compute than others? Perhaps it does them in much less time, but the more natural explanation is that it delegates those tasks to a smaller model, in which case we are back to having many different AIs.

To calculate the number of AGIs we could run, I divide the globally available FLOP/s by the FLOP/s required to run the AI that performs the most compute-intensive task (task with the highest runtime compute requirements).

This calculation would be accurate if one unified system can use this amount of FLOP/s to match human performance on *any task*, and can't perform any tasks with less FLOP/s.

This calculation is conservative by the lights of the FTM, in that it *underestimates* the number of remote human workers whose output we could match by running AIs. This is because (according to the FTM) many tasks can be done with much less runtime compute than the most compute-intensive task.

But your assumption that a massive training run will be needed to train AGI implies AGI will be one unified system.

All that is strictly-speaking implied is that a massive training run will be needed to automate the hardest-to-automate tasks. I expect many tasks will be performed by smaller and more specialised AIs than this. That said, I do expect the AI trained in that massive training run to have fairly general capabilities.

Value-weighted cognitive tasks

Throughout the report, whenever I refer to the % of cognitive tasks – or the fraction of cognitive tasks – I am using a particular method for weighting different tasks.

Roughly and intuitively, each task is weighted by how important it is to the economy in 2020. A task that many people perform, and are paid lots of money to perform, has more weight than a task performed by fewer people on lower wages. More precisely, a task's weight is proportional to the total $ that people earn while performing the task.[202]

Even more precisely, a task's weight is given by the elasticity of GDP to that task in 2020. If you performed 1% more of that task, how much would that increase GDP? If the answer is "GDP would increase by x%", then the task's weight is x. The task-based models I'm using assume that performing 1% more of *every* task (both cognitive and non-cognitive tasks) would increase GDP by 1%, so the total weight of all tasks equals 1. If economic inputs are allocated efficiently, this definition should match the one relating to wages.[203]

The weights are pinned to a particular year (in my case 2020 for convenience) because the relative economic importance of tasks changes over time. In particular, tasks that are automated and so can be performed in higher volumes and quality tend to become less economically

[202] For each person, this is given by the time they spend on the task multiplied by their hourly wage.
[203] For tasks performed by physical capital, a task's weight should equal the total amount paid to rent capital to perform the task.

important over time (e.g. producing food). Tasks that cannot be automated or made more productive tend to become more important (e.g. those in healthcare and education).

This same effect happens in the Full Takeoff Model (FTM) as AI automates cognitive tasks. Tasks that AI can perform in great quantities become less economically important – their weight decreases – ones still performed by humans become more important. The weights change over time, and so I pin my weights to 2020 when referring to the % of cognitive tasks.

An analogous definition of task weights applies to R&D as to the broader economy. As before, the precise definition ties a task's weight to how much performing more of the task would increase R&D output (R&D output in each timestep is proportional to the rate of R&D progress in that timestep). If performing 1% more of a task would increase R&D output by x%, the task's weight is x. Again, if R&D resources are allocated efficiently, this should correspond to weighting each cognitive task by the total wages paid to people while they perform it.

Objections talking in terms of the "% of cognitive tasks"

Objection 1: We have already automated >70% of 1700 tasks? And there was no explosion of economic growth. So why do you think this time will be different?

In economic models of automation, the growth effects of automation depend on *how quickly* new tasks are automated. If you automate 90% of tasks, the models I've seen predict GDP/capita should rise by >10X.[204] But if this is spread out over 300 years, it might correspond to 1% growth per year.[205] So the effect would not be extremely high rates of growth. But if this was spread out over 10 years, it would correspond to >20% growth.[206] So the first key difference here is that I'm considering much faster rates of automation than we've seen historically.

The second difference is that I think we could ultimately see *full automation*. Many growth models predict a qualitatively different long-run outcome from this compared to mere partial automation. With full automation, models tend to predict accelerating economic growth.[207]

Objection 2: Won't we introduce new types of tasks into the economy? If so, then AI that can perform 100% of *2020* cognitive tasks won't necessarily be able to perform the new cognitive tasks we've introduced.

I think this objection has some force.

[204] You get 10X from having all human workers concentrated on the remaining 10% of tasks, and having enough machines to increase per-task output on the automated tasks by 10X. This increases output of all tasks by 10X, so increases total output by 10X. You can get further gains from increasing the output of automated tasks even further. This all assumes you're investing enough in capital accumulation to get enough machines to do this.
[205] 300 years of 1% annual growth is a 20X increase.
[206] 10 years of 20% growth is a 7X increase.
[207] I discuss this point further in a previous report on whether AI could drive >30% GDP growth.

One attempt to dodge is to reiterate that the FTM implies that, over time, the automated tasks will become less important and non-automated tasks will become more important. We can claim that the model represents the introduction of new tasks (that AI can't perform) via the increased importance of non-automated tasks. More precisely, the apparently new task is really just a new application of a pre-existing non-automated task and this new application makes the non-automated task more important (as the FTM predicts).

I like this dodge. I think it suggests that the FTM's predictions need not go badly wrong because of this objection. If we think many important new tasks will be introduced that AI can't perform, we can increase the degree to which automated tasks become less important (by decreasing the parameter ρ) and keep in mind that our training requirements for AGI (AI that can perform ~all cognitive tasks) should include any newly introduced tasks.

But I do worry that the FTM's misleading ontology, in which the tasks needed for GDP and for R&D are fixed over time, may introduce other issues. This objection makes me view the abstraction of "% of cognitive tasks that AI can perform" less useful and meaningful.

Objection 3: AI will add value by enabling entirely new workflows as well as by automating existing ones.

As with the last objection, I think it has some force but there's a dodge that I like.

The growth model allows that, in addition to replacing humans on automated tasks, AI can have *additional* economic impact by producing more output on automated tasks than humans previously produced. If AI has economic impact by enabling new workflows, we can say the model represents this via AI producing additional output and already-automated tasks. The apparently new task is just a new application of a pre-existing automated task. If we think this effect will be significant, we should increase the parameter ρ.

Objection 4: In practice it won't be possible to actually measure what % of cognitive tasks AI could perform.

I agree that it will be very hard to evaluate precise claims about the "% of cognitive tasks" that AI could readily perform in each year. But it isn't meaningless; it's the kind of thing that economists have tried to measure. In principle you measure it by going through the concrete cognitive tasks that each person is in fact performing in each year (and how much they're being paid to do it, as implied by the time they spend and their salary), and ask the technological question of whether AI could perform that task instead (with a limited amount of engineering work and rearranging of workflows). Any measurement would necessarily involve many arbitrary judgment calls about what to include, but that doesn't render it meaningless or prevent us reaching rough conclusions.

Even if it didn't map to reality well at all, the "% cognitive tasks" abstraction would still be the best way I'm aware of to model AI continuously improving from today when it (seemingly) can readily perform <1% of economic tasks to a future world where it can perform ~all cognitive tasks. A skeptic can just think of the framework as giving some arbitrary one-dimensional scale on which "AI capabilities" improve between today and AGI.

Objection 5: Whether AI can or can't perform a task depends in part on the extent to which nearby tasks are already automated. But in your model, whether AI can perform a task depends solely on SOTA AI capabilities (measured via the biggest training run to date).

I agree with this. Automating one task can "unlock" a nearby task for automation by standardising the workflow. If nearby tasks have been automated, this reduces the AI capabilities needed to automate any given task.

My current interpretation of the Full Takeoff Model is that AI can "readily perform" a task if it can perform it with <1 year of engineering effort and work changing workflows, and it's profitable to make these workflow changes. This ignores the question of "But are adjacent tasks already automated?'

I'm not sure there's a way to get around this without significantly complicating the model. A better version might say that, if there's been a few years since AI could perform 20% of tasks then this reduces the training requirements for AI to automate 30% of tasks, since AI automating 20% of tasks will make it easier to automate further tasks. More generally, the training requirement for performing x% of tasks fall over time once AI can perform x - e% of tasks because we expect nearby tasks to be automated.

I *think* the effect of this change would be to make very fast takeoff somewhat less likely (by raising the training requirements for AI suddenly performing 100% of tasks without any nearby tasks being automated) and make very slow takeoff less likely (by lowering the training requirements for eventually automating 100% of tasks via iteratively automating more and more nearby tasks).

The truth is, though, that the evidence about training requirements for different levels of automation is already extremely rough. It consists of first estimating the requirements for 100% automation (perhaps via Bio Anchors), then adjusting this based on evidence about the effective FLOP gap for lower levels of automation. My uncertainties here are already so big that this doesn't feel like a significant contributor.

How this report relates to previous thinking about takeoff speeds

Paul Christiano's 2018 blog post

Paul Christiano argued for a slow takeoff in an influential 2018 blog post. Most of the post counters various arguments that have been made for fast takeoff. His central argument for slow takeoff is:

- *Before we have an incredibly intelligent AI, we will probably have a slightly worse AI.*
- *A slightly-worse-than-incredibly-intelligent AI would radically transform the world, leading to growth (almost) as fast and military capabilities (almost) as great as an incredibly intelligent AI.*

I agree with this argument, but I think its conclusion is that takeoff will be *continuous* rather than that takeoff should be *slow*.

The argument precludes a discontinuous jump in capabilities or impact, because some slightly-worse AI would have caused an intermediate level of impact first. This I agree with (though I assign some probability to discontinuous jumps in capability nonetheless).

But the argument doesn't preclude AI capabilities and impacts increasing continuously but extremely rapidly. It doesn't speak to whether the slightly-worse AI will occur 1 year vs 1 second before the slightly-better AI. And this can make a big difference to takeoff speed. If AI capabilities improve continuously but go from today's capabilities to AGI in one month, then it seems possible that we go straight from world GDP doubling in 24 years to it doubling in 1 year, which is a very fast takeoff by Paul's definition. And the GDP trajectory underlying this could also be entirely continuous.

The framework of this report is (an example of) one in which AI progress is assumed to be continuous but this can still give rise to fast takeoff if the rate of continuous AI improvement is sufficiently fast. More concretely, AI capabilities improve continuously as you cross the effective FLOP gap, but if you cross that gap sufficiently quickly then takeoff is fast.

Paul predicts that takeoff will be slow enough that there's a full 4-year doubling of world GDP before the start of a 1-year doubling (and a full 8-year doubling before a 2-year doubling, etc). The Monte Carlo analyses calculate the probability that this is the case:

Median AGI training requirements in simulation (FLOP using 2020 algorithms)	Probability of a full 4-year doubling of world GDP finishing before a 1-year doubling begins	Probability of a full 8-year doubling of world GDP finishing before a 2-year doubling begins
~1e31	49%	32%
~1e36	74%	47%

In a nutshell, then, my reply is: *Yes takeoff will probably be mostly continuous, but it could still be fast.*

Eliezer Yudkowsky's Intelligence Explosion Microeconomics

These are my opinions, and Eliezer might disagree with my characterisation of his thinking.

Intelligence explosion microeconomics doesn't argue for takeoff happening in weeks rather than in years.

My impression is that Eliezer Yudkowsky expects takeoff to be very fast, happening in time scales of days or months. By contrast, this framework puts the bulk of its probability on takeoff taking multiple years.

Does Eliezer give arguments for a transition taking weeks or months, rather than years? *Intelligence Explosion Microeconomics* (IEM), Eliezer's most detailed piece on this topic, gives various qualitative arguments for thinking that an intelligence explosion would not fizzle out but instead involve intelligence growing super-exponentially. There are also arguments for thinking AI will only need to outcompete the very small fraction of humans who do AI software R&D, rather than outcompeting the whole world, for them to kickstart an intelligence explosion. But these arguments don't (attempt to) quantify either the length of the transition to AGI or the pattern of software progress during and after the transition. So they don't speak to whether we should expect the transition to take days vs years; and thus to whether the accelerating AI progress will take the form of a slow takeoff vs a fast takeoff. Therefore I view those arguments as all wholly compatible with the full range of scenarios sketched in this report from 1 year takeoff to 30 year takeoff.

For example, Eliezer argues that the comparison of chimps and humans suggests that returns to improving the algorithms for general intelligence are very favourable in the human range. This can be interpreted as claiming that the returns to software R&D, which I quantify with r, will be favourable around human-level AI. But the argument doesn't (attempt to) quantify i) how good these returns are, or ii) the time it will take AI to transition from "comparably useful to software R&D as today's AIs" to "fully automates software R&D", or iii) the rate of software progress during and after this transition. Nor is there any attempt to make a bounding argument - to argue that any takeoff respecting these constraints must be extremely fast. Without this, the argument doesn't support takeoff happening in days vs years.

Perhaps the closest thing in IEM is the analogy to uranium. To simplify, when the density of uranium is below a critical threshold, no chain fission reaction occurs. But once it rises even a tiny bit above that threshold, the chain reaction quickly explodes.[208] If AI follows an analogous trajectory then the transition from "AI can't really help with software R&D" to "AI recursively self-improves, doubling its own abilities every few hours" would be very quick indeed. While I find the analogy suggestive of the *logical possibility* of a very quick transition, I think more work

[208] The threshold was k = 1. When k = 1.0006, the neutron level doubled every 2 minutes.

is needed to show that this is plausible or probable in the case of AI. (I'd be excited about someone doing this work, teasing apart why the transition is so sudden in the case of uranium and what analogous assumptions would need to hold in AI for a comparably quick transition.)

I think of this report as providing a quantitative framework for Intelligence Explosion Microeconomics

IEM *qualitatively* discusses a number of factors:

- Moore's law would go faster if humans ran on computer chips.
- The importance of algorithmic improvements to AI progress.
- The (large) returns to higher quality intelligence

As I said above, he doesn't use these factors to predict takeoff speed, or even to *bound* takeoff speed (e.g. he doesn't argue takeoff must take less than 1 year to be consistent with this evidence).

This report quantities these factors, wherever possible using relevant empirical data:

- Moore's law would go faster if humans ran on computer chips.
  - **Data on the returns to hardware R&D.**
  - **An economic model where AI increasingly automates the cognitive work of hardware R&D.**
- The importance of algorithmic improvements to AI progress.
  - **Data on returns to software R&D.**
  - **An economic model where AI increasingly automates the cognitive work of hardware R&D.**
- The (large) returns to higher quality intelligence
  - **Correlations between brain size, IQ and output.**
  - **Data from ML on how much less thinking time models with "bigger brains" need to achieve the same performance.**
- The report also incorporates the fact that AI might have strong comparative on some tasks over others, which tends to slow down takeoff speed.

So I think of the report as providing a quantitative framework to think about the factors that are discussed qualitatively in IEM.

Other arguments for discontinuous AI progress around AGI

I put ~6% on a substantial discontinuity in AI progress around the human range

I agree that takeoff could happen in mere days *if* there is a transition in days from "AI that can perform <20% of cognitive tasks in today's software R&D" to "AI that fully automates software R&D" *and* there are no compute bottlenecks to a software-only singularity. In this case, the pace of software progress could jump from its current pace (doubling every few years) to a 1000X faster pace (doubling in hours).[209]

I put low probability on such a large discontinuous jump occuring, for familiar reasons. In particular

1. Such large discontinuous jumps are rare in technological progress in general, rare in software progress and rare in AI,[210]
2. Large discontinuities seem more plausible in narrow areas than in broad areas and achieving general intelligence seems like a very broad area (though there's more room for doubt about how broad the skills for software R&D are).
3. It seems to me the rebuttals[211] of specific arguments for why there might be discontinuities around AGI are strong, and no good responses have been given 4 years on.

That said, I do think that small jumps in AI capabilities are likely to occur, and that we should attach *some* probability to substantial discontinuous jumps in AI capabilities.[212] How much probability would I assign to a large jump or kink in capabilities around AGI? Based on a rough outside view argument, I'd maybe assign ~6%?

- AI impacts looked at 38 trends, 20 of which had substantial discontinuities, ~50%
- They went looking for discontinuities though, so i'd put this probability 4X lower, ~12%.
- But the generality of AGI and the continuity of recent AI progress provides some reason to think big jumps are less likely. And even if there is a big jump, there's no particular reason for it to happen just before AGI. So I'll lower this another 4X: ~3%.
- I update up to ~6% based on an argument from Nate Soares about the chimp-human transition.

This has not been the main focus of my work, which was exploring the implications of a compute-centric approach that doesn't have substantial discontinuities (beyond those implied by a small effective FLOP gap).

[209] Let's assume software currently takes ~2 years to double with ~20,000 high quality software workers. After developing AGI that's equally good at software development as a high quality software worker, it may be possible to run ~10 million such AGIs. (This depends on the AGI runtime requirements.) This implies the first post-AGI software doubling could happen 10 million / 10,000 = 1000X more quickly than a current software doubling. So it could happen in < 1 day. If a software only singularity is possible, subsequent software doublings would be faster than this so that within a few days very many doublings have occurred: an 'intelligence explosion'. I model out this scenario here.

[210] Though progress can be somewhat discontinuous in particular narrow applications of AI like Go, progress in entire domains (like games) is more continuous and progress in the field as a whole is even more continuous.

[211] See Paul Christiano's blog post and AI Impact's page.

[212] By " substantial discontinuous jump" I mean ">10 years of progress at previous rates occurred on one occasion".

Takeoff speed can differ in different domains

Suppose it takes 1 month to go from today's AI capabilities to a disembodied AGI, but it then takes decades for this to affect GDP due to various bottlenecks. Is takeoff fast or slow? If you measure AI capabilities by the ability to perform cognitive tasks that a human worker could perform remotely, takeoff was very fast. But if you measure AI capabilities by their impact on the economy, takeoff was slow.

Or suppose that overnight we develop AI that massively increases military power, giving its controller a decisive strategic advantage, but this AI doesn't accelerate technological progress. In one relevant strategic sense takeoff is fast, but in another it is slow.

In cases like these, it can be useful to talk about takeoff speed *in domain X*. The question of takeoff speed becomes: *How long will it take to go from "AI has a minor impact on X" to "AI is making X go through the roof".* Here are some example domains:

- **Cognitive output**. Annual output on tasks that a human worker could do remotely. E.g. software development, math, strategy, knowledge work, writing.
- **SOTA technological progress**. The speed at which we're developing new technologies (distinct from how quickly they are integrated into the global economy).
- **Military power**. Ability to win a hot war.
- **GDP.** GDP as measured by the incomes and consumption of fleshy humans.
- **AI-inclusive GDP.** GDP as measured by the incomes and consumption of fleshy humans *and* digital agents (including human uploads and AIs).

We care about different domains for somewhat different reasons, and fast takeoff is more plausible in some domains than others.

Conditions under which simple growth models predict fast takeoff in GDP

Let's first take a simple case where we're not modelling technology or TFP.

In standard growth models output Y (which represents GDP) is a function of capital K and labour L. The equation is:
$Y = g(K, L)$

Capital cannot replace labour in these models. There are diminishing returns to capital which makes it hard to get a large output by building more capital alone. An extreme but simple way to model this is:
$Y = \min(K, L)$

These models can represent full automation from AI via capital being able to replace labour in production. After the transition to full automation there aren't diminishing returns to capital. If you can amass enough capital, output can become very high. To increase output significantly you'll need enough capital to replace all the labour many times over. One simple way to model this is: Y = K + L

There are a few ways to model a continuous transition here. You can model the economy as containing many tasks, and have capital perform and continuously increasing fraction of them. (This is my approach in this report.) Or you can model the elasticity of substitution between capital and labour (as in a CES production function); it starts <1 (so that capital and labour are complements) and continuously increases until it approaches infinity (so that capital and labour are perfectly sustitutable).

Fast takeoff means that the growth rate of Y increases very suddenly during the and immediately after transition (see earlier discussion). This will only happen if both:

1. The transition happens very quickly.
    - Output can only become very high once that transition has occurred and capital can replace the limited supply of labour. If the transition is gradual then the gain in Y will be spread out over time.
2. Shortly after the transition there is *lots* of capital, including enough to replace labour many times over.
    - Even after the transition, output won't become significantly higher unless there is enough capital to replace labour many times over. (If one type of capital replaces labour (e.g. AI robots) and another doesn't (e.g. trucks) then there must also be lots of *both* types of capital, as there are diminishing returns to each type.)

If both these conditions hold then you quickly transition from a world where capital can't replace labour to a world where it can and there's enough capital to replace it many times over. After the transition, the inputs to production are many times higher than before the transition.

Growth models would predict that Y increases extremely rapidly during such a transition, implying a very rapid increase in the growth of Y. The simple example equations I gave clearly imply that if conditions 1 and 2 hold, there will be a very rapid increase in the growth of Y.

What about technology? Well, the best models of technological progress are similar to the model of output I've been discussing. You simply replace Y with dA/dt, the rate of technological progress (you also model diminishing returns, but we can set that aside). So you'll get a sudden and dramatic increase in the rate of technological progress if conditions 1 and 2 hold for the tasks involved in technological progress. So this doesn't really change the analysis.

How does this all relate to the broader report? The historically fast growth of both computer hardware and software give reason to think conditions 1 and 2 could hold for cognitive tasks.

- Condition 1 (fast transition) holds if we rapidly cross the effective FLOP gap, achieving 100% automation of cognitive tasks. Fast growth of hardware and software (i.e. of FLOP/$ and 2020-FLOP per FLOP) allow us to cross the effective FLOP gap quickly.

- Condition 2 (enough capital to replace humans many times over) holds if there's enough computer capital to replace human cognitive labour many times over shortly after the transition. Fast growth of hardware and software allow us to quickly increase the numbers of AGIs we can run after crossing the effective FLOP gap.
    - In fact, condition 2 nearly always holds in my framework because either i) by the time we have enough FLOP to train AGI we already have enough to run 10s of billions of AGIs, *or* ii) around when we develop AGI the amount of 2020-FLOP is increasing *extremely* rapidly due to AI automation of hardware and software R&D.
        - For factor (i) it's essential that compute can be easily reassigned from one purpose to another. After inventing AGI we won't need to print new specialised AGI chips to run the new AGI algorithms.
    - So the main determinant of whether there's fast takeoff of GDP is normally condition 1.

This gives an interesting perspective on why hard takeoff in GDP can happen in this framework. It's because it allows for the possibility of a rapid transition to a world in which computer capital can replace human cognitive labour and there is very soon enough computer capital to replace human cognitive labour many times over.

If I had captured this dynamic with a different sort of growth model, I would have got a comparable result. E.g. if instead of a task-based model, I could have used a standard CES growth model with 3 inputs: human labour, computer capital, and physical capital. Rather than the 'transition' corresponding to AI performing a greater fraction of tasks, it would correspond to increasing substitutability between human labour and computer capital. I'd peg this substitutability to the biggest training run that has occurred. The overall result would be the same: the possibility of a fast transition to a world where AI can completely replace human cognitive labour.

This also gives an interesting perspective on a key bottleneck of the report: physical capital. While the quantity of computer capital has grown extremely rapidly in recent decades, this is unusual. The quantity of physical capital generally grows much more slowly. This means that it could prevent Y from becoming extremely large, even if there's enough computer capital to replace human cognitive labour many times over.

Why a fast takeoff probably has its origins in the transition *to* AGI rather than purely in the *aftermath* of AGI

I'll analyse a simple toy model of growth dynamics in the aftermath of AGI. This toy model suggests that the internal dynamics of the aftermath themselves won't by themselves lead to a fast takeoff. But it also suggests that if the transition to AGI was sufficiently quick, there would be a fast takeoff.

Here's the toy model. We've developed AGI, and can run a certain number of AGIs. Let's say that, initially, the AGIs' total cognitive output equals 10. For now, let's say the only way to increase their cognitive output is via software R&D to improve the AGI algorithms (i.e. recursive self-improvement).

After a certain amount of time, the AGIs will have doubled their own cognitive output to 20. Let's call this amount of time D_1. D_1 is the duration of the first doubling of cognitive output after AGI. ("D" for "Doubling time".)

After some additional amount of time, the AGIs will have doubled their cognitive output a second time to 40. Let's call this additional amount of time D_2. D_2 is the duration of the second doubling of cognitive output after AGI.

And similarly, D_3 is the duration of the third doubling of cognitive output, and D_4 the duration of the fourth. More generally, D_n is the duration of the nth doubling of cognitive output after AGI.

The key question for takeoff speeds is: **what is the ratio between D_n and D_{n+1}?** That is, what is the ratio between successive doubling times? If the ratio is very large, e.g. $D_n = 10D_{n+1}$, then there is a fast takeoff. We go very quickly from AI self-improving at a moderate rate to it self-improving 10X faster. But if the ratio is relatively small, e.g. $D_n = 2D_{n+1}$, then takeoff is slow. We move gradually from AI self-improving at a moderate rate to it self-improving at a slightly faster rate.

This metric of takeoff speeds mirrors Paul's definition in terms of GDP doubling times, except that we're replacing "GDP" with "cognitive output".

Ok then, what *is* the ratio between D_n and D_{n+1}? In this toy model, **there's good reason to think the ratio is <2**. In particular, if there's *any diminishing returns whatsoever* to increasing cognitive output, the ratio is <2.

Let's compare D_2 and D_1. At the start of the second doubling, cognitive output is 2X higher than at the start of the first doubling. (By the definition of a "doubling of cognitive output".) If the amount of cumulative cognitive output needed to achieve the second doubling exactly equalled that needed to achieve the first doubling, the second doubling would take half the time. With twice the cognitive output (per second), it takes half as long to do a fixed quantity of work. So *if* (work needed for first cognitive doubling) = (work needed for second cognitive doubling), *then* $D_1 = 2D_2$.

But in fact diminishing returns are fairly ubiquitous.[213] It will probably take *more* absolute effort to double output the second time than it did the first time. The lowest hanging fruit, the biggest improvements that are easiest to find, will have already been taken. This implies that the second doubling will take *more* than half as long as the first doubling. So *if* (work needed for first cognitive doubling) < (work needed for second cognitive doubling), *then* $D_1 < 2D_2$. The ratio between successive doubling times is <2.[214]

So, at least in this toy model, it seems like takeoff will be slow in the aftermath of AGI. The rate of recursive self-improvement will increase gradually with each doubling of cognitive output, rather than suddenly.

But this still leaves open the question of how the rate of AI improvement in the aftermath of AGI compares to the rate of improvement before AGI. If D_1 = 1 week (AI cognitive output doubles in a week immediately after AGI) but the doubling in cognitive output just before AGI took one year, then the ratio between successive doubling times on either side of AGI would equal 52! The argument above still leaves open the possibility of a very fast takeoff during the transition to, and immediate aftermath of, AGI.

This explains why I believe that a fast takeoff is unlikely to arise purely from the internal dynamics of a post-AGI world, but could easily arise from a *rapid transition* to AGI.

There are considerations omitted from the toy model which could lead to a fast takeoff dynamic occurring purely from the internal dynamics of a post-AGI world. If the fraction of the world's compute used to run AIs working to improve AI algorithms increases very rapidly in the immediate aftermath of AGI, then the ratios between successive doublings could be larger. For example, if twice as much compute is used for each successive doubling, then that could double the ratio. Or, similarly, if the data AGIs can access increases very rapidly between doublings, this could also increase the ratio between successive doublings. More generally, the pattern is that there's some input to AI development (compute, data, something else) that grows very rapidly in the immediate aftermath of AGI.

I think though, that this kind of dynamic is likely to only be significant if there is a rapid transition to AGI. If the transition lasts many decades, it seems likely that we'll already be using AIs to improve AI algorithms and already be leveraging all the data we can to train our AIs. In this scenario, I wouldn't expect very significant gains to be left on the table from reallocating the world's inputs just after we cross the threshold for AGI. If so, it would raise the question: why expect those inputs to be reallocated just after AGI but not before?

[213] *Are Ideas Getting Harder to Find* finds diminishing returns in the economy as a whole and many particular areas of it. See also this blog post by Matt Clancy examining the empirical evidence. To my mind, there are also strong *a priori* reasons to expect diminishing returns. If some improvements are harder to discover than others, then most simple models will exhibit diminishing returns as the easier-to-discover improvements are found earlier on.

[214] This is just a bound. I make my best guess about what happens in this scenario when I assess whether a software only singularity might occur.

These arguments are not conclusive. But they do lead me to expect that, conditional on a fast takeoff, the fast takeoff probably doesn't arise purely from the internal dynamics of a post-AGI but also relies on a fairly quick transition to AGI.

How likely is a software-only singularity?

Suppose you have a fixed amount of hardware that's capable of doing a particular number of physical FLOP/s.[215] You use this hardware to run AGIs (one or more AIs that collectively automate all cognitive labour) that do software R&D. In particular, they try to improve the algorithms that the AGIs are running on.

In this scenario, how would the total cognitive output[216] of the AGIs change over time?

In section 5 I said this depended on i) how long it takes to double the AGIs' cognitive output the first time, ii) how the doubling time for cognitive output changes over time.

This appendix discusses (ii), in particular whether there will be a **software-only singularity** with doublings becoming quicker over time and, if so, how long this might last before doublings become slower over time.

While current returns to software R&D suggest a software-only singularity would happen comfortably, returns may become worse as we approach AGI and being limited to a fixed amount of physical FLOP/s could bottleneck software progress in a couple of ways.

Overall, I'm roughly ~65% on a software-only singularity occurring, and my median best guess is that it would last for ~2-3 OOMs if it happened. What would 2.5 OOMs of a software singularity mean? My unit of software in this section is "useful cognitive output per FLOP". So 2.5 OOMs means you can 300X the rate of progress on software development, persuasion, and any other cognitive task. One way to imagine this is that it's as if the software improvements allowed all your AGIs to think 300X more quickly; though in fact the progress will come from a combination of "you can run more AIs in parallel" and "AIs can think in new and qualitatively better ways". (And I expect some of the progress to allow AI to do entirely new things that they previously couldn't have done even with ~arbitrarily long to think.)

[215] In practice, I expect physical FLOP to be growing very rapidly during any period where there might be a software only singularity. However, the simplification of imagining that physical FLOP is constant is still useful. This is because it can tell us about whether software (2020-FLOP per FLOP) might grow *much much* faster than physical FLOP during this time. If it does so, then physical FLOP will be roughly constant on timescales over which software grows very significantly. So the question "Would there be a software singularity?" maps to the question "Would software grow much much faster than physical FLOP?".

[216] By this I mean their ability to make progress on software R&D and their output in other cognitive domains like maths, strategy, persuasion, etc. My preferred unit for cognitive output is "How many remote human workers would it take to add the same amount of value?" So if the AGIs make some software progress in one month, and you'd have needed 1000 human workers to make the same amount of progress in one month, then the AGIs' cognitive output is "1000 remote human worker equivalents". More.

Note, even without a software singularity I expect software progress to become extremely fast by the time we have AGI.

In the rest of this section I:

- Recap the mathematical condition under which a software only singularity occurs.
- Distinguish between pure efficiency improvements and capability improvements.
- Argue that a singularity via only efficiency improvements seems plausible, ~50%.
- Argue that including capability improvements makes it significantly more plausible, ~85%.
- Suggest potential bottlenecks do not rule out a software only singularity but do make it less plausible, leaving me on ~65%.

Recap: the mathematical condition for a singularity

Whether there is a software singularity depends on the returns to software R&D. These returns can be quantified by the parameter r. The meaning of r is: each time cumulative software inputs double, 2020-FLOP per FLOP doubles r times. During a (potential) software only singularity these inputs are provided by AGIs and they only increase due to the AGI's improved software.

As discussed above, the mathematical condition for a software-only singularity is **r > 1**. Each doubling of cumulative cognitive R&D inputs must *more* than double 2020-FLOP per FLOP.

Two types of software improvements

One type of software improvement that AGIs might make is simply to make the algorithms on which they're running **more efficient**. The same level of intelligence is then produced with fewer physical FLOP/s. An example of this type of improvement would be *pruning*, where some of the connections in a dense neural network are removed ('pruned') but the performance of the system is (mostly) maintained.

This contrasts with the second type of improvement, which **increases the capabilities** of the AGIs. A greater level of intelligence is produced, perhaps with the same or more physical FLOP/s. For example, GPT-3 performs much better at a range of language modelling tasks than GPT-2.

If AGIs are trying to achieve a software only singularity, they will be able to make both kinds of improvement. They will presumably work on both improvements in (roughly) whichever combination best improves software.

I will first assess whether a software only singularity could be achieved by the 1st type of improvement alone, and then discuss the effects of the second type of change.

An efficiency-only singularity

We are restricting ourselves only to *efficiency* software improvements, i.e. ones that decrease the physical FLOP/s to achieve a given capability. With this restriction, the mathematical condition for a singularity here is the same as before: each doubling of cumulative inputs must *more* than double the efficiency of AI algorithms. If this holds, then the efficiency of running AGIs (of fixed ability) will double faster and faster over time. Let's call this an "efficiency-only singularity", which is of course an example of a software-only singularity.

Estimating r from 'AI and efficiency'

What data do we have on this? Recall that to estimate r we need data on cumulative inputs and on output.

Let's start with outputs. AI and efficiency, an OpenAI blog, looks at how the runtime FLOP/s needed to achieve a given level of performance on ImageNet has changed over time. They observe an 18X decrease from 2012 - 2017. This corresponds to an efficiency doubling time of 15 months and an efficiency growth rate of 58%.[217]

What about inputs? Tamay Besiroglu's dissertation suggests that the number of computer vision researchers grew at 19% over the same period.[218] If the cumulative research effort on ImageNet grew at the same pace, that implies that that **r = 2.9**.[219] (Recall, this means that each doubling of cumulative R&D inputs doubles runtime efficiency 2.9 times.)

My impression is that similar rates of software improvement have been achieved in other domains of ML, with efficiency doublings happening every 1 - 2 years.[220] But gathering more data points on this would be a very tractable and useful exercise.

If the value of r when we first get AGI is similar to this estimate, then there would comfortably be an efficiency-only singularity. However, there are a few reasons to think that r will be smaller than this.

1. **ImageNet inputs rose more quickly than 19%.** I don't have data on the amount of research done specifically on ImageNet. It's plausible that it rose faster than the number of computer vision researchers overall after 2012 did. ImageNet rose in prominence, as did approaches to it that used large amounts of compute. On the other hand, the growth of *quality-adjusted* researchers is probably slower than the growth of researchers if many

[217] e^(0.58*5) = 18 -- 18X increase in 5 years. e^(0.58*15/12) = 2 -- doubling time of 15 months

[218] Calcs.

[219] 55/19 = 2.9.

[220] For example, table 2 of OpenAI's paper shows similar or faster software gains on other select tasks as on ImageNet (though this is for training compute, not runtime).

new researchers entered the field.[221] In addition, even if *annual* research inputs rose faster than 19%, *cumulative* inputs would have risen more slowly than annual inputs.[222] Let's say that quality-adjusted cumulative ImageNet inputs actually grew at a rate of 25%; this still implies **r = 2.2**.[223]

2. **There was low hanging fruit to improve ImageNet in 2012 as algorithms were using more physical FLOP than previously**.[224] If you suddenly have access to much more physical FLOP/s than previously, new algorithms will become available[225] and people won't previously have been able to pluck the low-hanging improvements to improving them. This seems correct, but I don't think it suggests we should use a lower value of r.
    a. Firstly, the model I'm using already incorporates low-hanging fruit. Each efficiency doubling is harder to achieve than the last. (Indeed, the FTM normally predicts that the first post-AGI software doubling will take >~100X effort as a software doubling takes today, though this depends on the parameter choices.) So it is consistent with the observation that efficiency improvements were easy in 2012 and have become harder since. The objector here would have to further claim that the value of r itself, which controls the rate of diminishing returns, should decrease over time.
    b. Secondly, AGI will plausibly be in an analogous situation to ImageNet. More physical FLOP will be used to train AGI, and more physical FLOP/s to run it, than with previous systems. So you'd expect there to be low-hanging fruit here for the same reasons as with ImageNet.[226]
3. **Minimal efforts made to make vision algorithms compute-efficient before 2012.** Before 2012, computer vision algorithms used much less compute. In particular, compute was a small fraction of the total costs of a project, much smaller than human

[221] This is the relevant comparison, because any efficiency gains will allow us to run more AGIs of a fixed quality.

[222] ImageNet had been going since 2010, and its predecessor since 2005 (source). Earlier work on computer vision also contributed to the stock of relevant cumulative inputs. So there would have been a notable stock of cumulative inputs in 2012. If the growth rate of *annual* inputs increased in 2012 then, it turns out mathematically, the growth rate of *cumulative* inputs is initially lower than this and catches up only after a few years.

[223] 55% / 25% = 2.2.

[224] In particular, I believe AlexNet, the system that famously won the competition in 2012, used significantly more training and runtime compute than had historically been used.

[225] Here's a toy example of how this might happen. To train algorithm 1 on D data points requires physical FLOP of 10*D^1.2. To train algorithm 2 on D data points requires physical FLOP of 1000*D^0.8. The second algorithm only becomes more efficient than the first once you are using a sufficiently large number of data points. It scales better with data but has a larger up-front cost, so only becomes 'available' when we are using enough physical FLOP to process lots of data points.

[226] In fact, this gets at an important way in which my model may *underestimate* the speed of software progress around AGI. It implies that the *first* software doubling after AGI will take *much* more effort than the first such doubling after ImageNet 2012, because of algorithmic progress inbetween pushing us further out the curve of diminishing returns. (~100X more effort, depending on what much software inputs increase before AGI.) But if each new OOM of physical FLOP "resets" the low-hanging fruit, then the first software doubling in each case may require *equal* effort. This would mean that the initial post-AGI software doublings would happen *much* faster than I'm predicting, and even if r<1 there would be many very rapid doublings of software.

labour. So there was minimal incentive to optimise algorithms for compute-efficiency. But after 2012, the cost of compute for projects rose very rapidly, increasing the financial incentives to make computer vision algorithms efficient. So there may have been one time gains from transitioning from a “we don’t care about efficiency” to a “we do care about efficiency” regime that will not be repeated again. If we ignored these one-time gains, our estimate of r would have been lower.

I think this point has some merit, but it doesn’t seem to justify a much lower value of *r*. There are often “one-time gains” that drive progress, and my model of software progress is really just aggregating together many such one-time gains. And, to repeat, the model incorporates diminishing returns and so it *expects* the one-time gains to become smaller and less common over time. And there will plausibly be comparable “one time gains” in the future: as researchers spend $1 millions and much more on training runs, the financial incentives to make AI algorithms more efficient will grow significantly. The question is whether the transition from pre-2012 to post-2012 is part of a series of one-time gains that we should include in the model as part of a pattern of diminishing returns that will continue into the future, or whether they constitute an outlier from that pattern.[227] I currently lean towards the former. This is influenced i) by a suspicion that, even before 2012, algorithm designers in computer vision were at least somewhat concerned with efficiency, and ii) by a sense that similar rates of software progress have happened in other ML domains until the current day (2022).

If I discount 25% of the observed efficiency gains as due to a one-off effect that should be treated as an outlier, then my estimate drops from r = 2.2 to **r = 1.7**.[228]

4. **r will fall as we approach ultimate limits of software efficiency, and will be lower by the time we get to AGI.** There is some ultimate limit to how efficient software can become; e.g. you can’t run AGI on 10 FLOP/s. Once we reach this limit, further progress is impossible. This corresponds to r = 0.[229] So r falls towards 0 as we approach ultimate limits, and may have fallen somewhat by the time we get to AGI.[230]

   The longer your AI timelines, the stronger this argument as there is more time for software to approach ultimate limits before AGI.

   This seems broadly correct to me, and I expect r will be lower when we get AGI than today. I don’t expect this effect to be huge because I don’t think we’ll have reached

[227] One way to settle this is empirical. Look at whether rates of software progress were significantly higher just after 2012 than in periods since; if so it suggests the transition was an exception. Another way is speaking to practitioners in the field about whether they feel there is a continued pattern of this kind, or whether there was a regime change around 2012 that will never be repeated.

[228] 2.2 * 0.75 = 1.65.

[229] Each doubling of cumulative inputs causes 0 doublings in efficiency.

[230] An important question is whether r falls based on our *linear* distance from the limit, or our log-distance from it. If the former, then only in the last OOM of software improvement will r fall to 0 and r probably won’t change much before AGI. If the later, then r will fall somewhat during each software doubling along the way and r may decrease significantly before AGI.

ultimate limits by the time we get to AGI (more). In addition, AGI will probably be trained using more physical FLOP and run using more physical FLOP/s than previous systems. So it seems unlikely that the first AGIs will be maximally efficient, given our lack of experience optimising systems with that level of compute.[231] I think we'll have made 2 - 3 OOMs progress, with more than this still remaining before reaching ultimate limits.

My guess would be that r falls ⅓ of the way towards 0 by the time we get AGI; so my estimate drops from r = 1.7 to **r = 1.1.**[232] I think people could reasonably expect r to fall ½ or even ⅔ of the way towards 0, which would imply r < 1.

After considering those 4 objections, my best guess for r fell from 2.9 to ~1. This matches my gut feeling that, once we have AGI, returns will be worse than the naive ImageNet data suggest, but not *way way* worse, and that means there could well be an efficiency only singularity. I'm about 50 -50 on whether r > 1 at this point. (Later, I'll discuss how many software doublings a singularity might last for, if it happens.)

Estimating r from Computational Limits of Deep learning

Thompson et al. (2022) find that "3 years of algorithm improvement is equivalent to an increase in computing power of 10X" in image models. This corresponds to a growth rate of 77%. The paper does not estimate growth of inputs, but using the 19% from above implies r = 4. This is higher than the equivalent r = 2.9 we estimated previously. Applying the same penalties as in the last section would leave us on **r = 1.5**.

Estimating r from 'How Fast do Algorithms Improve'

How Fast do Algorithms Improve, by Sherry and Thompson, is another source of data on efficiency improvements. They survey a wide range of algorithms, most of which are not specific to machine learning, and calculate the annual rate of efficiency improvement. The rate of improvement depends on the size of the problem - how many examples or data points must be processed. They find that, at a problem size of n = 1 billion, the efficiency of the median algorithm had a growth rate of 25%.[233]

To estimate r, we also need data on software investments across this period. A couple of data sources imply software investment, measured in real $, grew at a rate of 6 - 14% during this

[231] Although if you think that AGI will consist of multiple interacting AIs, we may have already trained most of those AIs before training the final AI that allows the AIs to collectively perform all tasks. (OTHO, the tasks performed by the final AI could well be the main bottleneck, so that final AI's capacity for improvement may be most relevant.)

[232] 1.65 * 2/3 = 1.1.

[233] They say "28% per year" on p.5, which corresponds to a growth rate of 25%: e^0.25 = 1.28.

period. Let's say the number of quality adjusted researchers had a growth rate of 10%.[234] That implies r = 2.5.

This methodology avoids objections 1, 2 and 3 from above, as these were specific to the ImageNet data being used. Objection 4, that returns may be worse once we get to AGI, still applies. The same ⅓ adjustment as before leaves us at **r = 1.6**.

There are a couple of big uncertainties here.

Firstly, Sherry and Thompson observe very large disparities on progress in different types of algorithms.[235] If we have similar uncertainty about AGI we should be open to an efficiency-only singularity happening comfortably, or to it not happening at all.

Secondly, the result is sensitive to the problem size used. A problem size of 1 million, rather than 1 billion, reduces the median rate of progress from 25% to 14%,[236] which would leave us at r = 0.9.[237] I don't know a principled way to choose the problem size. I think using the size of current SOTA AI models (e.g. # params or # data points) would imply a somewhat higher problem size than 1 billion. The problem size will be larger still for AGI.[238]

I see this second estimate as broadly consistent with the first; both suggest r = ~1 is plausible for efficiency improvements around AGI. I find the second estimate slightly less informative because it looks at algorithms in general rather than focussing on AI.

In both cases, my biggest uncertainty is how much r will decrease between today and when we get AGI. It seems fairly clear that today r > 1 by some margin, but that could easily stop being the case by the time we get to AGI.

A software-only singularity (including capability improvements)

The above analysis assumed we were restricted to only using software improvements that increase the efficiency of running systems with ~fixed capabilities. I guessed there was a ~50% chance these improvements would happen increasingly quickly; and if so, that might be ~10X total improvement before progress began to slow.

[234] This is probably slightly too high, as it looks like real $ grew at ~10% and so quality adjusted people probably grew ~2% more slowly (due to rising real salaries). As a result, the estimate of r will be slightly too low.

[235] They say (p.4) that just under half the families show little to no improvement, while 14% of algorithms improved by more than 11X each year (on average).

[236] e^0.14 = 1.15.

[237] 1.6 * 14%/25% = 0.9.

[238] This gives another reason to think there won't be extremely harsh diminishing returns at this time (larger problem sizes lend themselves to faster algorithmic improvements, measured in % terms).

In reality, AGIs would also try to make improvements to make *more capable* AI, perhaps running on as much or more FLOP/s. This could only *increase* the chance of a software-only singularity occurring and its duration.

How large might this effect be? I think there are good reasons to think the effect will be big:

- New capabilities are plausibly a much bigger source of progress than efficiency improvements on existing capabilities.
  - OpenAI argue[239] for this. For example, they suggest that AlexNet – the system that famously won ImageNet in 2012 – achieved its level of performance much more efficiently than pre-existing algorithms could have. More generally, they argue that the first time a capability is achieved, the algorithm used is typically *much* more efficient than pre-existing algorithms at achieving that capability. Their arguments seem reasonable to me; the key question then becomes how useful these new capabilities are. (Who cares about dramatically increasing the efficiency of new capabilities if those capabilities aren't useful?)
  - My impression is that the growth in AI's economic importance since 2012 has mostly come from new capabilities, rather than merely from increasing the efficiency of capabilities that already existed before 2012.[240]
  - There is very large variation among humans in terms of effectiveness at software R&D.[241] This suggests that, around the human level, there are very large gains to software R&D from increased capabilities.
  - More speculatively, perhaps AGI whose (collective) capabilities surpass any human will identify new kinds of software improvements that humans cannot see. Perhaps many such improvements will exist, as humans haven't been able to see any of them. If so, there could be extremely rapid progress once AIs surpass the best humans.
  - Overall, this gives me a prior that capability improvements will be a much bigger deal than efficiency improvements during a software-only singularity. So if we previously thought r = 1 for efficiency improvements only, you might think r = 3 when you include capability improvements (so that capability improvements are twice as big a deal as efficiency improvements).
- Grace (2013) measures algorithmic progress in 6 domains, and finds that in many areas about half of all progress is due to software and half due to hardware.
  - I believe that these improved algorithms often used constant or increasing amounts of compute, so her evidence speaks to non-efficiency gains from software.
  - One extremely hacky way is to assume that her measure of 'hardware progress' maps to increases in FLOP/$, and assume that inputs to software R&D have grown at the same rate as inputs to hardware R&D. Then the returns to software R&D will be the same as we calculated earlier for hardware: **r = ~7**.

[239] Section 5.3 of the paper that accompanied the AI and Efficiency blog.

[240] People could dig into this: the specific use cases that generate revenue and the algs used for them.

[241] I've heard there is SMPY data on this but I couldn't find them with an hour or so looking. Salary differentials are indicative, but they may underestimate true productivity differences for social reasons.

  - This is very far above the threshold for a singularity (r = 1). In the last section I adjusted the efficiency estimate down from r = 2.9 to r = 1.1 based on a few objections; similar adjustments here[242] would still leave the overall estimate at **r = ~2.5**.[243]
  - An important caveat, discussed more below, is that these returns might not be possible without *increasing* the amount of physical training FLOP and runtime FLOP/s.
- Eliezer Yudkowsky argues that the evolution of humans suggests that there are favourable returns to improving the algorithms for general intelligence around the human range, and that those returns aren't sharply diminishing. Roughly speaking, this corresponds to the claim that r is large when software reaches human levels. It's hard to translate this into a quantitative claim, but the next footnote argues from Yudkowsky's claim to the conclusion that **r > 2** (around the human range).[244]
  - I find this evidence fairly unconvincing for the same reasons given in Paul Christiano's blog post. Selection for intelligence, in particular for learning from and communicating to others, may have increased significantly during this period due to the massively increased importance of culturally accumulated knowledge for survival.[245]
- Evidence from within ML suggests 'cleverer' models make much better use of compute.
  - Jones (2021) finds that, when training AlphaZero on the game Hex, using 10X more training compute reduces the runtime compute[246] needed to achieve a given test result by 15X.[247] In other words, a model that is "10 times smarter" (as quantified by its training FLOP) can achieve the same result with 15X less thinking (as quantified by its runtime FLOP).[248]

[242] Are these objections applicable? The first (inputs rose more quickly than we assumed) might apply if the problems studied received faster growing investment than is typical for software. The second (low hanging fruit due to more physical FLOP) applies more strongly as I believe the amount of physical FLOP by the systems studied in Grace (2013) was continually increasing. (By contrast the physical FLOP used on Imagenet didn't increase after 2012 in the OpenAI data.) The third objection (minimal effort to make algorithms compute-efficient before 2012) doesn't apply, as it was specific to 2012.

[243] 7 * 1.1 / 2.9 = 2.65

[244] Yudkowsky argues that the effort needed to increase intelligence didn't significantly increase during the evolution from Australopithecus to Homo erectus to Homo sapiens. In this period, he claims, brain size increased by a factor of four. If software increased by a similar factor over this period (i.e. if better software was responsible for the same share of cognitive improvement as bigger brains), then software too increased by 4X. According to the model of this report, the effort needed to improve software increased by 4^(1/r) during the period. Suppose we accept Yudkowsky's claim that the effort needed didn't increase much; let's commit to saying it increased by <2X. To meet this commitment, we'd need 4^(1/r) < 2, which implies r > 2.

[245] *The Secret of Our Success* is the best account of the importance of culture to the biological evolution of humans that I'm aware of.

[246] The reduction in runtime compute comes from reducing the depth of the tree search. (It must be reduced by more than enough to compensate for the mdoe being larger.)

[247] This implies that, with a fixed budget for both training and runtime, it's optimal to spend ~55% on training and ~45% on runtime. (The system is Cobb Douglas: test result = Train^0.55 * Runtime^0.45. I have verified that this Cobb Douglas equation roughly reproduces Jones' results.)

[248] We can relate this to model size, i.e. FLOP/s at runtime, if we make an assumption relating training FLOP to runtime FLOP/s. Let's assume that when you double model size you need 4X the training FLOP. In this case, 10X more training FLOP corresponds to a ~3X bigger model. Jones' result is then that **a 3X**

- On a simple intuitive level, this suggests that the returns to training more capable AIs could be large. Increases in training FLOP, or simply training efficiency, could result in significantly more capable AIs.
- A simple toy model supports this intuition.
    - To simplify the model, let's suppose 10X more training FLOP reduces the runtime FLOP needed for a given result by 10X, rather than 15X. I.e. doubling training FLOP means that only half the runtime FLOP is needed to achieve a given result.
    - Suppose we have 1000 FLOP available to us in each timestep. Each timestep we must use the latest algorithms to train AGI from scratch and then run AGIs to improve AI algorithms.
    - It turns out that, given our assumptions, it's optimal to use 50% of our FLOP on training and 50% on runtime. 500 FLOP each.
    - First, let's walk through an example where we don't increase training efficiency and so don't train more capable AIs.
        - Suppose that in the first timestep AGIs double cumulative R&D inputs. Further, assume that this doubles runtime efficiency (i.e. assume r_runtime=1) but doesn't change training efficiency (assume r_training=0). The new algs take the same amount of FLOP to train, but run on half as many FLOP/s.
        - Then in the second timestep we'll again use 500 FLOP to train AGIs of the same ability, but we can run twice as many of them with the other 500 FLOP. We get twice as much R&D done as in the first timestep, so we double cumulative R&D inputs again. As before, this doubles runtime efficiency (r_runtime=1) but doesn't change training efficiency (r_training=0).
        - The process continues: in the third timestep we again use 500 FLOP to train our newly designed AGIs, use the other 500 FLOP to run 4X as many AGIs as in the first timestep, double cumulative R&D inputs again, so double runtime efficiency (r_runtime=1) but don't change training efficiency (r_training=0).
        - The process can continue indefinitely. We chose the knife-edge r=1, and so the software doubling times are constant over time. If we'd chosen r > 1, each doubling would have taken less long.
    - Now let's walk through an example where we *do* increase training efficiency.
        - Like last time, AGIs double cumulative R&D inputs on the first timestep and this doubles runtime efficiency (r_runtime=1). This time let's assume it also doubles training efficiency (r_training=1). The new algs take half as much FLOP to train *and* run on half as many FLOP/s.

bigger model achieves the same result with 15X less runtime FLOP/s. This means it thinks for 45X less long! Or, equivalently, a 2X bigger model achieves the same result with 6X less FLOP/s and 12X less thinking time.

- Then in the second timestep we *could* train AGIs of the same capability with only 250 FLOP, and use the other 750 FLOP to run them twice as efficiently as before. We would get *three* times as much R&D done as in the first timestep. (2X as efficient, using 1.5X the runtime FLOP.) Call this option 1.
- *Alternatively,* we could use 500 FLOP for training. Compared to the option 1, this doubles training FLOP. We will train more capable AIs than in option 1.[249] How much more capable? Above, we assumed that doubling the training FLOP halves the runtime FLOP needed to achieve a given result. (This was based on Jones (2021).[250]) So our more-capable-AGIs will achieve the same output with half as many runtime FLOP, compared to option 1. We could achieve the same software progress as option 1 by running them with 375 FLOP, but in fact we can run them with 500 FLOP. This means we'll get 500/375 = 4/3 times as much R&D done as in option 1, and *four* times as much as in the first timestep. Call this option 2.
  - There's a factor of 2 from reducing the runtime FLOP of our old AGIs, and a factor of 2 from using more efficient training to train more capable AGIs. Based on the result from Jones (2021), improvements to runtime and training combine multiplicatively.
- Option 2 is better than option 1, by a factor of 4/3, because it exploits the ability to train *smarter* AGIs. The actual numbers from Jones (2021) suggests the true effect of increasing training efficiency would be slightly larger. (He found 10X more training compute drives **15X** more runtime efficiency, whereas we assumed it would drive only **10X**.)
- In this toy model, there's a software only singularity just if r_runtime_efficiency + r_training_efficiency > 1; we saw earlier

[249] By analogy with Jones (2021), this happens via training a bigger model (one with more FLOP/s) than in option 1. Remember, option 1 itself involved AGIs using half as many FLOP/s as in timestep 1, so in option 2 AGIs will use more than half as many FLOP/s. How much exactly? If we assume model size goes with sqrt(training FLOP) then the model size will be 0.5*sqrt(2) = ~0.7 times as big as in timestep 1. So models still get smaller, but they also get smarter, due to the combined effects of training improvements and runtime improvements.

[250] [Weedsy fn.] We are applying the result from Jones (2021) in a subtly different context here. The original result showed that doubling the physical training FLOP (slightly less than) halved runtime FLOP to achieve a given result. There was only one algorithm used (AlphaZero). Here we are again imagining doubling the physical training FLOP, but we *also* imagining that we just halved training FLOP by making algorithmic improvements. You *could* object that the doubling training FLOP won't halve runtime FLOP if you've just made some algorithmic improvement to make training more efficient. Maybe that efficiency improvement only improves training at the new smaller scale, but not so much at the original scale? This objection doesn't seem convincing to me. My guess is that the training algorithms developed since AlexNet (the 2012 ImageNet system) also function well at the training FLOP used for AlexNet. Much more significant to my mind is the fact that Jones (2021) is a toy environment, while we're here imagining AI that can do 100% of cognitive tasks.

that an efficiency only singularity occurs just if r_runtime_efficiency > 1. So this toy model suggests that a software only singularity is considerably more plausible.

- Above I estimated r_runtime_efficiency = 2.9, based on data about ImageNet (though revised it downwards to 1.1). What about r_training_efficiency? The same AI system driving the runtime estimate produces an estimate of r_training_efficiency = 3.2,[251] which a similar discount would reduce to 1.2.
- More generally, most improvements in runtime efficiency also increase training efficiency,[252] but not vice versa.[253] So I'd expect r_training_efficiency > r_runtime_efficiency. This implies that we get a software singularity as long as r_runtime_efficiency > 0.5. This is definitely the case now, and I expect it will still be true when we get to AGI, but I'd still assign >25% to the contrary.

- All this is to say that a toy model implies that being able to train cleverer models would make a software-only singularity significantly more plausible. It uses a tradeoff between runtime and training FLOP that Jones (2021) observed in a toy environment, but that type of tradeoff does seem plausible.
- This suggests that if we thought r = 1 only including runtime efficiency improvements, we should think **r > 2** once we include training efficiency improvements that can lead to more capable models. (Because in the toy model the contribution of the latter was expected to be bigger, r_training_efficiency > r_runtime_efficiency.)

- If we discover a learning algorithm that scales as efficiently with training and runtime FLOP as the human lifetime-learning algorithm, then it seems plausible we could do a software-only singularity just by making that algorithm more efficient.
  - Correlations between brain size and IQ, and IQ and productivity, suggest a relationship between brain size and productivity in humans. In particular, a 10% bigger brain is ~5 IQ points smarter, and so ~30% more productive. Extrapolating heroically, a 2X bigger brain is ~8X more productive (see calcs and data source).
  - Suppose you had (an inefficient version of) the human learning algorithm, and were able to make it 2X as efficient. That would mean that, using the same amount of physical FLOP as before, you could train and run a model that was like a "2X bigger brain" and so was 8X more productive.
  - Whether you succeed in doing a software-only singularity or not depends on whether you become faster or slower at making 2X efficiency improvements of

[251] While the system's runtime efficiency increased 18X (growth rate 55%), its training efficiency increased by 21X (growth rate 61%). With the same assumption that cumulative inputs grew at 19%, this implies r_training_efficiency = 61/19 = 3.2.

[252] Training consists in doing multiple forward passes. If you increase runtime efficiency, you decrease the compute for each forward pass.

[253] For example, improving the optimiser or the hyper parameters don't affect runtime efficiency.

that kind. To succeed, each 2X efficiency improvement must take <=8X as much effort as the last.

- This condition will hold unless diminishing returns to efficiency improvements are *much* steeper than they are today.
    - For context, the estimate of r_efficiency = 2.9 from Imagenet models corresponds to each 2X efficiency improvement taking 27% more effort than the last. Much less than 8X more effort![254]
    - We can use a model like before, where r_brain_alg means: when you double cumulative R&D inputs you double the efficiency of the human learning algorithm r_brain_alg times. The condition for software-only singularity is **r_brain_alg > 0.3**. This is a *lot* lower than the estimates we've been seeing.
        - Translating back to the condition on overall r ('When we double cumulative software R&D inputs, how many times do we double productivity?'), I see this as evidence that **r > 1**, perhaps comfortably so.
    - Here's a toy model of this dynamic.
- A qualification here is that perhaps the human learning algorithm scales well within the human range of variation (±10%), but no further. Or perhaps by the time we find anything that scales this well, we'll have already hit the ultimate limits to software. On the other hand, you might think we could get *better* scaling than the human learning algorithm by scaling data in proportion to model size. (The human learning algorithm keeps data fixed as brain size increases.)

I said I was 50-50 on an efficiency only singularity happening, at least temporarily. Based on these additional considerations I'm now at more like ~85% on a software only singularity. And I'd guess that initially **r = ~3** (though I still think values as low as 0.5 or as high as 6 as plausible). There seem to be many strong ~independent reasons to think capability improvements would be a really huge deal compared to pure efficiency problems, and this is borne out by toy models of the dynamic.

How long might a software-only singularity last?

Even if a software only singularity occurs, there's a further question of *how much* software improves before software doublings start to slow down. I don't have much to say here. There are a few sources of evidence that I'm aware of:

- **How big were the total efficiency improvements on ImageNet?** Runtime efficiency increased 18X from 2012 to 2017; training efficiency increased 44X from 2012 to 2019. Perhaps returns to increasing the efficiency at which we achieve AlexNet-level performance become much worse shortly after this (though returns for making more capable models more efficient might be better). We can anchor to this and predict total gains of **2 - 5 OOMs** before returns become worse and doublings start to slow down.

[254] 2^(1/2.9) = 27%.

    - Why 2 OOMs? Conservatively only include training efficiency increases (44X) and assuming these ran out soon after 2019 (at 100X).
    - Why 5 OOMs? Combine training and runtime increases multiplicatively as in the toy model above: 18 * 44 = 800, ~3 OOMs. Then assume the trend could continue for another ~2 OOMs before running out.
- **How far away are ultimate limits to software efficiency (at this level of physical FLOP)?**
    - *Runtime efficiency.*
        - I expect that when we first train AGI, its runtime efficiency will be less than the human brain. The first version of an AI system with a new capability is typically not well optimised for runtime efficiency. AGI might initially be 10X or 100X less efficient than the human brain, perhaps much more.
        - In addition, I'd guess that the ultimate limits for runtime software efficiency are significantly better than that of the brain:
            - The brain does specialised cognitive tasks using general thinking software that is much less efficient than specialised software would be (e.g. doing mathematics using neural networks).
            - There's significant variation between humans in IQ, even holding brain size fixed.
            - In evolutionary time, we have not had brains our size for that long; and they have not been optimised for doing the cognitive tasks needed for science for long.
            - AI will have a some significant structural advantages over humans that make them more productive; e.g. faster serial speed, no leisure (though there are potential ethical concerns here), more motivated to work hard and coordinate effectively. More.
        - Overall, **3 OOMs** or more increase here seems likely before hitting limits.
    - *Training efficiency.*
        - When we first train AGI, its training efficiency will be *many* OOMs below human learning efficiency. Human lifetime-learning takes ~1e24 FLOP,[255] and training AGI with 1e30 FLOP would be less than my median. Naively, that suggests 6 OOMs improvement available just in training efficiency.
            - Even if the human learning algorithm is extremely complicated and evolution has learned thousands of clever tricks, in principle AI could discover and hardcode them themselves.
        - It seems like human learning efficiency is not close to physical limits.
            - We could do much better to fully optimise people's experiences for learning, e.g. by providing better and more personalised learning curricula.
            - Again, there's large variation in learning efficiency between humans.

[255] Quoting from Bio Anchors: "*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.*"

      - Again, there’s only a limited amount of time (on evolutionary timescales) that human-sized brains have been optimised for learning. And much less time still being optimised to learn in our current cultural environment (e.g. from books).
      - So if there’s 2 - 5 OOMs of software gains still to be had once we get AGI, perhaps returns become worse (ending the software singularity) after ~2 OOMs.
    - Overall, **5 OOMs** or more increase here seems likely before hitting limits.
  - *Bigger brains*
    - Even if human brain learning and runtime efficiency is at physical limits, you could increase total productivity simply by training bigger brains. Above I discussed the naive estimate that doubling brain size would 8X productivity; this means 4X more output per FLOP.
    - If we trained 100X bigger-than-human brains using the human lifetime learning algorithm, this would take 100X the compute (people with bigger brains don’t take longer to learn), 1e26 FLOP. That would increase productivity by 10,000X, **4 OOMs**.
  - *What do ultimate physical limits tell us about how long the software-only singularity will last?*
    - If initially r= 2 and we’re Y OOMs from ultimate physical limits to software, and the software-only singularity requires r>1, then a really simple model might say that the singularity will last for Y/2 OOMs. The idea is that r=0 at ultimate limits, and we assume it falls a constant amount towards 0 with each OOM of software improvement.
    - We guesstimated 3 OOMs to improve runtime software efficiency, 5 OOMs to improve training efficiency and at least 4 OOMs to increase output by training bigger models. I think these are multiplicative,[256] summing to 12 OOMs.
    - So then if we start at r = 2, then the software singularity would last for 6 OOMs. (Above my best guess was r=3, so i’m being somewhat conservative here.)
    - Clearly, a lot of work is being done by the assumption that r = 2 when the software singularity starts (i.e. when we fully automate software R&D). Assumptions of r = 1.2 would lead to a much shorter singularity; r = 4 would lead to a much longer one.
    - A lot of work is also done by the linearity assumption, that r decreases steadily OOM by OOM. Maybe r falls quickly to ‘close to 0’ and then slowly approaches 0 over subsequent OOMs.
- **Directly estimate r at human levels of software.** Above I discussed Yudkowsky’s claim that returns to software aren’t sharply diminishing around the human level of intelligence, and suggested we could very roughly parse the claim as r > 2. This implies

[256] Training and runtime improvements are multiplicative in the toy model above, and then training bigger brains is clearly a distinct type of improvement.

that, even when software is at human levels, returns would have to become notably worse before a software only singularity stopped.

I don't trust any of these lines of evidence, but my best guess is that, based on the evidence discussed so far, a software singularity, if it started, would last ~5 OOMs before software doublings become slower. It could be 1 OOMs, or even 10 OOMs.

Importantly, even after software doublings become slower ($r < 1$), there may still be very fast software progress for multiple doublings. For example, if $r = 0.5$ then each doubling takes twice as long as the previous one. If the fastest software doubling took 1 week, after which each doubling is twice as long as the previous one, there would still be 16X software progress over the next 30 weeks.[257]

Bottlenecks from a fixed supply of physical FLOP

Two bottlenecks are salient to me.

1. The need to experiment to find better algorithms
2. The dependence of software progress on using more physical compute.

The need to experiment to find better algorithms

As discussed in section 6, one seemingly important contributor to software R&D is doing experiments to see which algorithms have good performance in practice. Across all the data series considered in this section, the physical FLOP available for doing such experiments was increasing exponentially while software progress happened. Perhaps this exponential growth in physical FLOP was needed to run enough experiments to maintain the observed pace of software progress. Perhaps we'd have seen slower software progress if the amount of physical FLOP had remained constant (as it would in a software singularity). If so we'd have estimated a lower value for r and judged the software singularity to be less likely.

For example, I estimated that $r = 2.9$ for runtime efficiency improvements on Imagenet. But perhaps we'd have only seen half these improvements had the physical FLOP used for experimentation remained constant. In which case I'd have instead estimated $r = 1.45$, and lowered my probability of a singularity accordingly. And similarly, perhaps the software improvements observed in Sherry and Thompson (2021) and Grace (2013) would have been smaller had the physical FLOP for experimentation remained constant. Again this would reduce the estimates of r that I derived and make a software singularity look less likely.

To make the potential bottleneck here concrete, let's imagine trying to achieve an efficiency-only singularity. Each doubling of efficiency will require a certain number of experiments. We can compare the number required for one efficiency doubling with the number required for the next efficiency doubling. The key question is: *How does the number of experiments required change*

[257] The next four doublings take 2 weeks, 4 weeks, 8 weeks, and 16 weeks. That's 2^4=16X progress in 30 weeks.

for successive efficiency doublings? If we needed a constant number of experiments to achieve each efficiency doubling, the physical FLOP needed for experimentation would actually *decrease* over time. After the first doubling, each experiment would take half as much physical FLOP.[258] If we needed twice as many experiments for each new efficiency doubling, the physical FLOP needed for experimentation would be constant over time. Each successive doubling would require twice as many experiments, but each experiment would use half as much compute. The effects would cancel. Lastly, if we needed *more than* twice as many experiments for each new efficiency doubling, the physical FLOP needed for experimentation would increase over time.

If we instead imagined a software-only singularity that included improvements in the capability of AGIs, then this analysis would shift. In the previous paragraph, after each software (efficiency) doubling, the physical FLOP per experiment halved. But capability improvements would make experiments more computationally expensive. So the physical FLOP per experiment would not halve after each software doubling; it might decrease more slowly than this, or even increase if new models use more physical FLOP/s.[259] This makes it comparatively more likely that experiments would significantly bottleneck progress.

Still, we could get significant capability gains while doing a fixed number of experiments per software doubling, by holding physical runtime FLOP/s fixed. And we can adjust how we conduct software R&D to reduce the reliance on large experiments (e.g. conducting experiments on a smaller scale, reasoning more from first principles, inferring the outcome of a training run from the first 100 timesteps, a move back to "good old fashioned AI" where AI runtime software is handwritten). I think experiments would probably eventually bottleneck capability improvements, but this might not happen until we've seen multiple OOMs of improvements.

One way to model this would be to have physical FLOP perform some fraction of software R&D tasks;[260] this input would stay fixed during the software singularity and so eventually (if R&D tasks are complementary) bottleneck progress. I believe Epoch are investigating a model of this kind.

My takeaways from the previous few paragraphs are:

- The number of experiments for each software doubling has to increase at a fast exponential rate for it to block an efficiency only singularity. This doesn't seem very likely.
- It is more likely to be a bottleneck for capability increases, but this is not guaranteed.

[258] I'm assuming that the physical FLOP required for an experiment is proportional to the runtime FLOP/s of the system the experiment is investigating.

[259] This depends on the balance between runtime efficiency improvements and capability improvements, and on how capability improvements affect the AGI's runtime FLOP/s. If we are increasing the physical FLOP/s of our SOTA AIs, then we will have fewer experiments at that scale; but capability improvements can also come from using a fixed amount of FLOP/s more effectively.

[260] You could use the fraction of lab spending on physical FLOP vs talent to decide the fraction of software R&D tasks performed by physical FLOP. (Although physical FLOP allows labs to develop and run AIs, as well as improving their algorithms; so this is problematic.)

- Overall, I’d be surprised if experimental bottlenecks block a software-only singularity in its early stages, but wouldn’t be surprised if they blocked it after a couple of OOMs of improvements.
- I think this consideration should lower our estimates of r; if I had to say it would lower r from 3 to 2.
- It also lowers my probability that a software-only singularity will occur at all from ~85% to ~70% and makes me think any software singularity would last less long (~2-3 OOMs rather than ~5 OOMs).

The dependence of software progress on using more physical compute

A decent chunk of software progress may be the result of software adapting to larger hardware scales (h/t Paul Christiano). In other words, there are fast diminishing returns to improving algorithms that use a fixed budget of (physical) FLOP/s, but using more FLOP/s allows us to find new algorithms that are much better adapted to the additional FLOP/s than our previous algorithms.

As a concrete example, suppose alg1 has efficiency 100 when run on 1e9 FLOP/s. alg2 has a very similar efficiency of 105 when run on 1e9 FLOP/s. But when run on 1e10 FLOP/s, alg2 has an efficiency of 200, compared to alg1’s efficiency of 110.

Efficiency of algorithms	alg1	alg2
1e9 FLOP/s	100	105
1e10 FLOP/s	200	110

alg2 is much better adapted to the new FLOP/s budget than alg2, even though their performance was similar on the old budget.

If much historical algorithmic progress is of this sort, then algorithmic progress would become much slower if our budget of FLOP/s remained constant (as during a software-only singularity).[261]

There’s a couple of reasons it could be easier to improve algorithms are larger hardware scales:

- Less effort has been made to optimise algorithms for that large scale historically, so there’s more low-hanging fruit.
- Improvements in scaling behaviour (e.g. moving from O(n^2) to O(nlogn), or moving to Chinchilla scaling) have bigger effects at larger levels of FLOP/s.

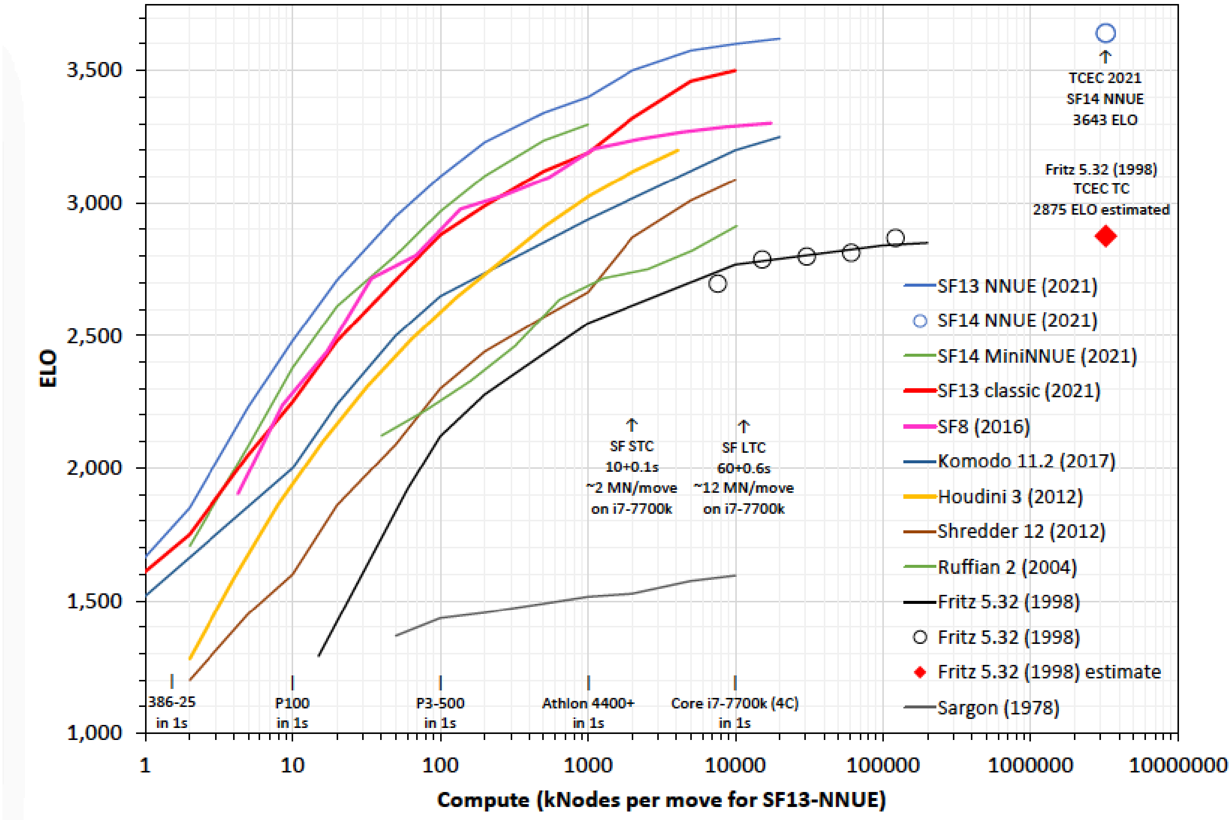
How does this consideration affect the estimates of r that I’ve used in this section?

[261] *How Fast do Algorithms Improve* supports the idea that we’ve only maintained our overall pace of algorithmic progress by increasing our physical FLOP budgets. It finds that algorithmic progress is faster at larger problem sizes.

- The *AI and Efficiency* and the *How Fast do Algorithms Improve* estimates are affected in similar ways.
    - They both measure algorithmic progress as a *reduction* in the FLOP/s to achieve a given capability. Their measured software progress does *not* rely on using new algorithms that are better adapted to new scales of FLOP/s.
    - *However*, the fast software progress they measure may be a result of adapting to a new large hardware scale, as happened with ImageNet in 2012. This could mean that the researcher inputs to software R&D *for that new hardware scale* grew especially quickly during that time, because the scale was previously neglected.
    - I already adjusted for this consideration for *AI and Efficiency,* where I ended up on **r = 1**.
    - I didn't adjust for this in *How Fast do Algorithms Improve,* so will make an additional adjustment from r = 1.6 to r = ~1.2.
- The (very rough) estimate based on Grace (2013) would be more affected. It looked at software progress over periods of time when the FLOP/s used by systems increased; if instead the FLOP/s used had remained constant, software progress may have been slower.
    - I estimated r = ~7, then adjusted down to r = ~2.5. I'd now adjust this further to ~r = 2.

So this mostly affects the estimate of r from Grace (2013), the only one that suggested a software singularity would happen comfortably. The estimates of r = ~1 from ImageNet and *How Fast do Algorithms Improve* aren't affected much.

Paul Christiano and Carl Shulman commissioned work to investigate this objection. They compared the performance of an old chess algorithm to a new algorithm at both old levels of (physical) FLOP and new levels of FLOP. The old algorithm was 1998 Fritz [black line] and the new algorithm was 2021 Stockfish [blue line].



[x axis shows compute used by a system; y axis shows its Elo rating. Each line corresponds to a different algorithm.]

If the objection is correct, 2021 Stockfish should have a bigger advantage at the new hardware scale (~10,000 kNodes) than the old scale (~100 kNodes).

It's ambiguous whether this is the case. It depends how you measure it.

- The Elo differences between the two algorithms are slightly bigger at the *old* hardware scale. At 100 kNodes, the Elo gap is ~1000, at 10,000 kNodes it's ~800. This suggests algorithmic progress didn't rely on increasing the hardware scale.
- But the efficiency improvements are bigger at the *new* levels of FLOP than at old levels of FLOP. Suppose you ask "how many times fewer FLOP does the new algorithm need to match the performance of the old algorithm?". 2021 Stockfish needs 100X fewer FLOP to match the performance of 1998 Fitz at 100 kNodes, vs ~500X fewer to match its performance at 10,000 kNodes.[262]

[262] The data also suggests that capability improvements are more significant than efficiency improvements. New algorithms achieve capabilities that would have taken old algorithms >10 OOMs of extra FLOP. By contrast the efficiency improvements are 2 - 3 OOMs. This confirms OpenAI's claim, above, that the first time a capability is achieved, the algorithm used is typically *much* more efficient than pre-existing algorithms at achieving that capability.

Overall, it seems plausible that software progress does depend on moving to new hardware scales to some extent. This mostly affects the estimate based on Grace et al, which was already extremely rough. This consideration slightly decreases my probability that a software singularity occurs, down from ~70% to ~65%.

Summing up

I've tried to assess the plausibility of a software-only singularity by looking at data about the historical returns to software R&D. I proceeded in a few steps:

- ImageNet data left me thinking that there was ~50% chance of a singularity based on efficiency improvements alone.
- Including significant potential gains from capability improvements increased this to ~85%.
- If a software-only singularity *does* occur, I guessed it might last for ~5 OOMs.
- Considering two potential bottlenecks, neither of which seemed compelling to me, lowered my estimates somewhat:
    - ~65% chance of a software-only singularity
    - I expect it to last ~2-3 OOMs if it does occur.
- Importantly, even if there is no software-only singularity, software progress might still be *extremely* rapid just after we fully automate software R&D due to the huge rise in R&D inputs. There could be multiple OOMs of fast progress on a fixed hardware base even if software doublings are slowing down over time. In addition, I expect the quantity of hardware to be increasing rapidly, driving further software progress.

Open questions

I've organised these open questions according to which components of the model they'd inform. I highlighted the ones that I thought had the best combination of importance and tractability in yellow.

- **To inform g($ on FLOP globally)** - semiconductor production scale up
    - If people were willing to spend $ trillions expanding semiconductor production, how long would it take to double the number of chips produced per year?
        - Via new fabs and more efficient use of existing fabs. *Not* via better chip designs - this falls under R&D scale up.
        - Are there fundamental physical bottlenecks to increasing manufacturing throughput above a certain level? E.g. a certain crystal needs months to be grown and this can't be expedited.
        - How would the above answers change if there was also abundant specialised cognitive labour (from AIs) to help with the expansion?
        - Use expert networks to speak to someone at TSMC / Intel / Samsung

- **To inform g(fraction of FLOP on a training run) -** prospects for rapid scale up of training runs
    - If you wanted to actually use 30% of global FLOP/s in a training run, how would you do that? What bottlenecks would there be? How long before you can start the training run?
        - What fraction of existing FLOP could you rent? What fraction of new production could you buy without being blocked?
        - How much engineering effort would it take to distribute training over ~1000X more chips than we've done to date?
            - Speak to people at labs about the engineering barriers.
        - What is the highest fraction of FLOP/s that could be used in a single training run in various scenarios?
    - Improve our empirical estimates of how the fraction of FLOP on a training run is likely to change over time.
        - What FLOP/s is currently available from all the world's AI chips?
        - How quickly will the FLOP/s globally and from AI chips grow over the next 10 years?
        - How easy would it be to reappropriate production lines currently producing non-AI chips to make AI chips?
        - How will the FLOP on the largest training run grow over the next 10 years?
- **To inform g(FLOP/$) -** prospects for future hardware progress.
    - Near-term forecasts of FLOP/$ from speaking to industry experts.
        - In a 'business as usual' scenario where AI improvements are modest.
        - What happens with hardware if we get AGI in 2030?
            - How many gains are still available from fabless R&D, improving the designs of chips made with existing fab production processes?
            - Once these gains have been taken, what's the next lowest hanging route to hardware progress?
    - Alternative paradigms: are quantum computing or optical computing plausible over the next few decades? What magnitude of improvement might they bring?
    - If people were willing to spend $ trillions on hardware R&D, how would that affect the rate of progress?
        - How much money could the field usefully absorb?
        - How many people could move in from adjacent fields and usefully contribute?
        - How sharp would the diminishing returns be to increased spending within each year?
        - What are the current bottlenecks to R&D progress, to what extent could they be relieved by more $?
    - If people were willing to spend $ trillions on hardware R&D *and* there was super-abundant expert cognitive labour (from AIs), how would that affect the rate of progress?

        - What are the current bottlenecks to R&D progress, to what extent could they be relieved by more $ *and* abundant cognitive labour?
- **To inform g(2020-FLOP per FLOP) -** prospects for future software progress
    - Gather up to date versions of the data from 'AI and efficiency' paper, for a variety of AI benchmarks.
        - To inform r_software.
    - Do more experiments where you run both old and new hardware using both old and new algorithms. Investigate whether the new algorithms only help with the new hardware, vs whether they help equally with old and new hardware.
        - To inform r_software and whether software progress is dependent on hardware.
    - Think of a new and better way to conceptualize (and ideally quantify) software progress that allows us to achieve new capabilities.
        - The first time a new capability is achieved, the algorithm that achieves it often does so using orders of magnitude less compute than any pre-existing algorithm.
            - E.g. these chess graphs, and section 5.3 of the 'AI and Efficiency' paper.
        - This is at odds with our formalism, in which the compute requirements for new and old capabilities decrease gradually year on year, halving every ~2 years.
        - The challenge here is simply to suggest a new framework for software progress that better captures the nature of software improvements that unlock new capabilities.
            - The new framework may imply, contra Bio Anchors, that we could *not* have trained AGI with ~1e36 FLOP using 2020-algs.
        - To improve the way I'm modelling software progress.
    - Estimate the correlations between IQ and output on key tasks like R&D.
        - We can combine this with IQ-brain size correlations discussed above.
        - The relationship between brain size and output informs the effective FLOP gap, whether a software singularity is likely to occur, and takeoff speed according to a one-dimensional model of intelligence.
    - Empirically, how 'jumpy' is algorithmic progress? What fraction of the total gains happen in unusually large discrete jumps vs normal progress.
        - To inform whether I should put more probability on large discontinuous jumps in capability.
    - During a software-only singularity, might it be possible to avoid retraining each generation of AGIs?
        - What techniques for making AI systems more capable/efficient don't require retraining from scratch? How big are the efficiency gains from these techniques? How long do they take?
            - E.g. chain of thought prompting.

        - If the model size increases significantly, is it possible to avoid retraining the system from scratch (e.g. by initializing the weights of the new larger system using the weights of a smaller system)?
        - If a new architecture is introduced, is it possible to avoid retraining the system from scratch?
        - To what extent could ~all AI training by done via online learning, so that precious compute is not “wasted” on training rather than running AGIs?
        - How can we integrate the answers to these questions in my analysis of whether a software singularity will occur?
    - What would ‘automating 20%, 50%, or 80% of software R&D’ look like in practice?
        - Speak to AI researchers about what tasks they perform. Estimate the time spent on each type of task. Describe what it might look like for AI to perform tasks that currently take x% of researchers’ time.
        - What percentage of these tasks could SOTA AI profitably perform today?
        - What percentage of tasks will AI be able to perform with a training run of (e.g.) 1e27 FLOP.
        - To inform whether it will be possible to get large productivity gains from partial software automation in practice.
    - How much easier will it be for AI to readily automate a large fraction of AI R&D tasks compared to a large fraction of the broader economy?
- **To inform the speed-up from automating AI R&D sooner than the global economy.**
    - How much easier will it be for AI to perform all cognitive tasks in AI R&D than all cognitive tasks in the broader economy?
- **To inform the size of the effective FLOP gap**
    - My research into evidence about the effective FLOP gap was fairly shallow. Two factors in particular could be investigated further.
        - How AI capabilities vary with training FLOP.
            - How does the performance of AI systems vary as we increase the training FLOP 10X - 1000X, but hold algorithms constant? What does this suggest about the increase in training FLOP needed to cross the effective FLOP gap?
            - Are there some domains where it takes significantly more FLOP to train AI than others? E.g. perhaps achieving human level at some band of games takes more FLOP than achieving human level for a comparably narrow band of language tasks.
        - How animal capabilities vary with brain size.
            - Pick animals with 3X, 10X, 30X 100X smaller brains than humans. Learn about the cognitive capabilities of these animals.
            - First ask: Could the animal do useful economic tasks (or help with R&D) if they were motivated to help (i.e. if we could perfectly control their second by second desires).
            - Second ask: Could the animal do useful economic tasks (or help with R&D) if their brain had been optimised for this by evolution?

                - This is a weirder counterfactual so harder to think about, but ultimately more relevant to the effective FLOP gap I think.
            - To the extent the answers are “no, they don’t have the cognitive capabilities to be helpful”, this suggests the effective FLOP gap is small.
    - What is the current $ value-add of AI? How is it changes over time, or with model size?
        - Various ways of operationalising this: investment, revenues, effect on GDP.
        - Relevant for when AI will first be capable enough to readily add $trillions / year to GDP.
    - Why do MIRI people think there’ll be a rapid (< years) transition from “shallow systems” to “deep systems” where the former aren’t very helpful to science and the latter can ~fully automate science? In the language of this framework, they think the effective FLOP gap is small.
    - Dig into AI impacts’ finding that it took decades to cross the human range in chess, Go and checkers.
        - This is in tension with the findings of this report. Some possible resolutions of the tension:
            - The effective FLOP gap is on the high end of my estimates, implying high AGI training requirements.
            - Progress in those games is slower due to slower investment growth and the absence of speed-ups from AI automation.
            - The effective FLOP gap is narrower than in those games, e.g. because “capabilities scale especially quickly in the human range” or “it’s difficult to partially automate jobs”.
        - This is also in tension with the one-dimensional model of takeoff.
        - The first step is probably finding data about how inputs to these domains (compute at training / runtime, software R&D effort) changed while they crossed the human range.
- **To inform thinking about bottlenecks**.
    - Do bottlenecks raised by economists suggest growth won’t *ever* accelerate?
    - To what extent do bottlenecks push towards slow takeoff in areas of strategic importance?
    - Critique and improve my analysis of bottlenecks in sections 6 and 9.
    - To what extent is AI progress driven by running big experiments vs software R&D labour? How fast would progress become if we had ~unlimited supply of the latter?
    - Some slightly more fleshed out ideas here.
- **Validate this model of takeoff speeds.**
    - David Schneider-Joseph makes some suggestions here.