
Forecasting AI Time Horizon Under Compute Slowdowns

Parker Whitfill¹, Ben Snodin², and Joel Becker³

¹MIT

²Independent

³METR

Abstract

METR’s time horizon metric has grown exponentially since 2019, along with compute. However, it is unclear whether compute scaling will persist at current rates through 2030, raising the question of how possible compute slowdowns might impact AI agent capability forecasts. Given a model of time horizon as a function of training compute and algorithms, along with a model of how compute investment spills into algorithmic progress (which, notably, precludes the possibility of a software-only singularity), and the empirical fact that both time horizon and compute have grown at constant rates over 2019–2025, we derive that time horizon growth must be proportional to compute growth. We provide additional, albeit limited, experimental evidence consistent with this theory. We use our model to project time horizon growth under OpenAI’s compute projection, finding substantial projected delays in some cases. For example, 1-month time horizons at 80% reliability occur 7 years later than simple trend extrapolation suggests.

Keywords: AI scaling, compute, time horizon, METR, algorithmic progress

1 Introduction

Kwa et al. [2025] show that, from 2019 to 2025, the time horizon of software tasks that AI agents can accomplish with 50% accuracy has grown exponentially, with a steady 7-month doubling time. However, this trend is arguably driven by exponential growth in compute over the same period [Sevilla and Roldán, 2024]. This research note provides a framework for forecasting METR’s time horizon trend given potential compute slowdowns [Edelman and Ho, 2025, Sevilla et al., 2024], conditional on no software-only singularity.

We begin in Section 2 with a simple theoretical model in which time horizon depends on compute and algorithms, and algorithmic progress follows an ideas-production model where the input is experimental compute. (Note that this assumption is contestable, and precludes the possibility of a software-only singularity [Davidson, 2025].) We show that, given this model, the basic facts that time horizon and (experimental + training) compute have been growing at constant rates imply that the time horizon growth rate is causally proportional to the compute growth rate — if the growth rate of compute halves, the growth rate of time horizon will also halve.

In Section 3, we test an empirical prediction of the theoretical model by comparing how log training compute relates to log time horizon within the Llama 3.1 and Qwen 2.5 families. We show that the relationship is approximately linear and the same across model families with different algorithms, consistent with the theory. This analysis, along with our theoretical framework, enables us to make an estimate of algorithmic progress in Section G — approximately $3.5\times$ per year, consistent with Ho et al. [2024].

Lastly, we combine our model with a compute forecast to see how much delay a compute slowdown would cause. In particular, we construct a time series of OpenAI’s compute spend from 2018–2025 and use OpenAI’s own forecasts of their compute spend up to 2030 as reported in Efrati and Muppidi [2025]. Figure 1 shows the effect of taking into account compute slowdowns for time horizon. While this scenario is primarily illustrative, it shows that some milestones can be substantially delayed — for example, 1-month, 50% reliable AI arrives in 2033 instead of 2029. In general, we find that delays are more severe for higher time horizons and higher reliability thresholds.

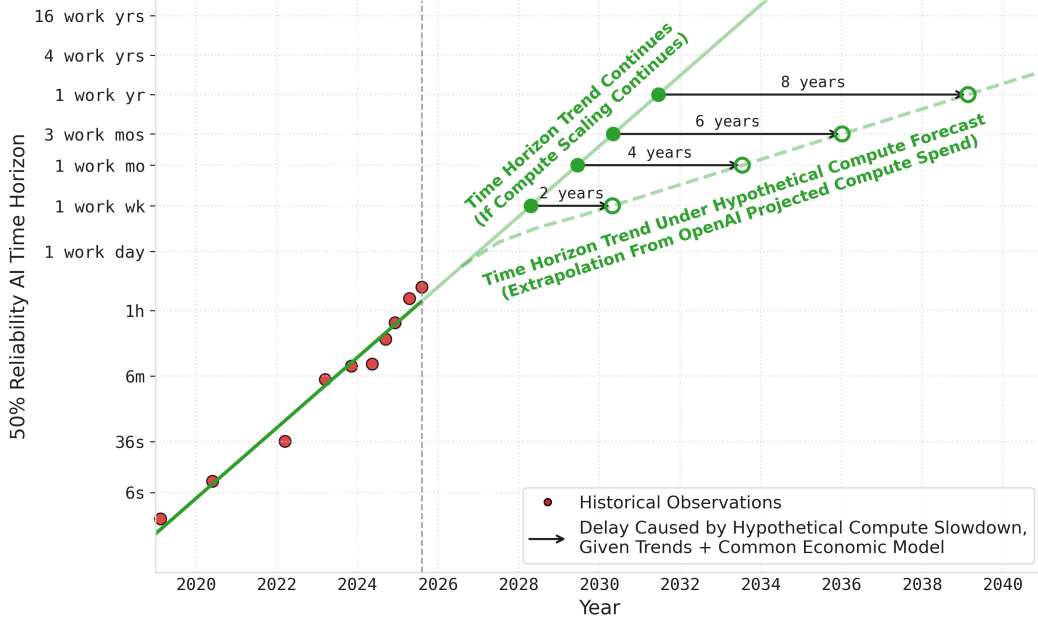


Figure 1: 50% reliability time horizon forecast implied by our model.

2 Theory

The goal of this section is to formally derive the relationship between compute growth and time horizon growth using the following two stylized facts. The first stylized fact comes directly from Kwa et al. [2025].

Stylized Fact 1. *Time horizon is growing at a constant rate.*

The next stylized fact is that total R&D compute spend is growing at a constant rate.

Stylized Fact 2. *Total R&D compute is growing at a constant rate.*

The best available data for this stylized fact is for training compute [Sevilla and Roldán, 2024], but total R&D compute (experimental plus training compute) also appears to be growing at an exponential rate.

Figure 2 shows both trends for OpenAI. The time horizon data is taken directly from Kwa et al. [2025]. The compute estimate comes from first constructing a time series of OpenAI R&D compute spend from their IRS Form 990s and Epoch AI [2025]. The compute estimates include training and experimental compute usage but not, for example inference compute.¹ We then use the methodology of Cottier et al. [2024] to convert from compute spend in dollars to FLOPs.²

¹Epoch AI [2025] already separates inference and R&D compute, and for 2018 and 2022 we assume 0 inference costs.

²More precisely, we use the inverse methodology, as they convert from FLOPs to dollars, while we convert from dollars to FLOPs. This conversion involves guesswork on which GPUs were used, the utilization rates and the prices, but it should give a roughly accurate picture.

Both trends are approximately straight lines on a log plot, indicating exponential growth.³ Combined with these stylized facts, our theoretical model will imply that the growth rate of time horizon is causally proportional to the growth rate of compute.

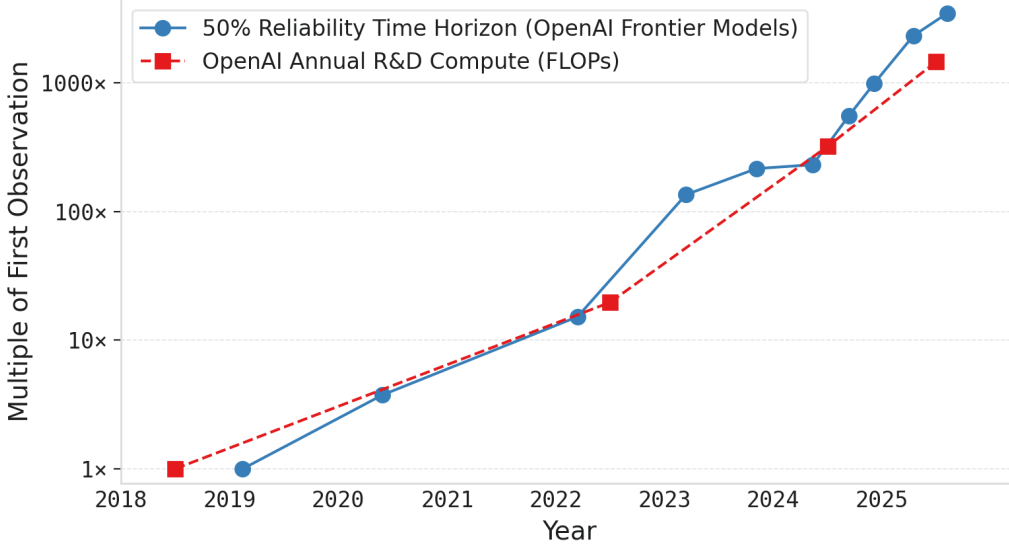


Figure 2: Total R&D compute and AI agent time horizon for OpenAI, 2018–2025.

2.1 Setup

Let

- Y = METR’s 50% accuracy time horizon in minutes,
- C = training compute spend in FLOPs
- E = experimental compute spend in FLOPs,
- A = algorithmic multiplier that scales training compute to effective compute,
- t = calendar time in years.

We start with two equations. First, we assume there is some function F that maps effective training compute to time horizon. We make no assumptions on F except differentiability:

$$Y = F(AC). \quad (1)$$

Next, we assume algorithmic progress follows the standard economic model of ideas [Jones, 1995]:

$$\frac{dA}{dt} \propto A^{1-\beta} E^\lambda, \quad \lambda, \beta > 0. \quad (2)$$

This equation says that the rate of algorithmic progress is governed by two factors: $A^{1-\beta}$, which denotes that over time ideas may get harder to find, and E , which denotes how much compute is dedicated to researching algorithms.

In this model, we assume that the effective inputs to algorithmic progress are well-approximated by experimental compute only. This could be justified if the primary bottleneck to progress is compute, not research labor. The primary reason we make this assumption is that it is plausible and it simplifies the relation between the compute growth and time horizon.⁴ Note that this explicitly assumes that

³Interestingly, the lines have nearly the same slope, indicating that compute and time horizon are growing at the same rate. The theory will not require this.

⁴Section C considers relaxing this assumption by adding a human researcher labor term.

a software-only singularity is not possible because labor (either human or AI provided) is not the constraint to more algorithmic progress. Therefore, our model is accurate only if the compute bottleneck approximation is good, or up to the point in time at which a software-only singularity kicks in.

2.2 Theoretical Results

Using (1) and (2), we link the pace of time horizon growth to that of compute growth. Recall that the growth rate of a variable is equal to its log derivative with respect to time. So our first step is to log-differentiate time horizon:

$$\begin{aligned}
\frac{d \log Y}{dt} &= \frac{1}{Y} \frac{dY}{dt} \\
&= \frac{1}{Y} F'(AC) \left(A \frac{dC}{dt} + \frac{dA}{dt} C \right) \\
&= \frac{C}{Y} AF'(AC) \left(\frac{d \log C}{dt} + \frac{d \log A}{dt} \right) \\
&= \frac{C}{Y} \frac{\partial Y}{\partial C} \Big|_A \left(\frac{d \log C}{dt} + \frac{d \log A}{dt} \right) \\
&= \frac{\partial \log Y}{\partial \log C} \Big|_A \left(\frac{d \log C}{dt} + \frac{d \log A}{dt} \right). \tag{3}
\end{aligned}$$

This equation says the growth rate of time horizon is equal to the growth rate of training compute plus the growth rate of algorithms, all multiplied by the log derivative of time horizon with respect to compute.

To simplify this expression further, we can note that equation (2) relates algorithm growth to compute investment. Furthermore, after enough time, the growth rate of equation (2) converges to

$$\frac{d \log A}{dt} = \frac{\lambda}{\beta} \frac{d \log E}{dt}. \tag{4}$$

While there are transition dynamics to this steady state, compute is currently growing extremely quickly, so the half-life of convergence is on the order of months.⁵ Therefore, it is without much loss to ignore transition dynamics and work with equation 4 directly.⁶

Plugging equation (4) into equation (3) yields

$$\frac{d \log Y}{dt} = \frac{\partial \log Y}{\partial \log C} \Big|_A \left(\frac{d \log C}{dt} + \frac{\lambda}{\beta} \frac{d \log E}{dt} \right) \tag{5}$$

This equation relates the growth rate of time horizon with the growth-rate of training compute plus the growth rate of experimental compute weighted by a few other terms.

Notice that $E + C$ denotes R&D compute spend. If we are willing to assume that training compute and experimental compute are proportional to total R&D spend⁷, then we can simplify this equation to our main equation of this note:

$$\frac{d \log Y}{dt} = \frac{\partial \log Y}{\partial \log C} \Big|_A \left(1 + \frac{\lambda}{\beta} \right) \frac{d \log(E + C)}{dt}. \tag{6}$$

From this equation, we can derive a few interesting results. First, because the stylized facts imply that the growth rate of time horizon and compute are constant, we can derive that $\frac{\partial \log Y}{\partial \log C} \Big|_A$ is constant, i.e., it doesn't itself depend on algorithms or compute.

⁵When we consider compute scaling slowdowns, the half-life will be slightly longer, but still fairly short while the inputs grow so fast.

⁶All further results are therefore implicitly asymptotic. For simplicity, the theory assumes instantaneous transitions. In reality there must be some delay between obtaining compute, algorithmic progress, and having deployed models, if only due to the time allocated to model training.

⁷In Appendix Section D we show what changes when we drop this assumption.

Lemma 1. *Stylized Facts 1 and 2 $\implies \frac{\partial \log Y}{\partial \log C} \Big|_A$ is constant.*

Proof. See Section B.1.

Second, this implies that the growth rate of time horizon is proportional to the growth rate of total R&D compute.

Proposition 1. *Stylized Facts 1 and 2 $\implies \frac{d \log Y}{dt} = c \frac{d \log(E + C)}{dt}$, where $c \in \mathbb{R}$ and $c = \frac{\partial \log Y}{\partial \log C} \Big|_A (1 + \frac{\lambda}{\beta})$*

Proof. See Section B.2.

Because $c = \frac{\partial \log Y}{\partial \log C} \Big|_A (1 + \frac{\lambda}{\beta})$, the substantive implication is that $\frac{\partial \log Y}{\partial \log C} \Big|_A$ is a constant. Notice $\frac{\partial \log Y}{\partial \log C} \Big|_A$ is a property of F , the function that maps effective training compute to time horizon, as $\frac{\partial \log Y}{\partial \log C} \Big|_A$ measures the returns for time horizon when increasing training compute. This term is constant if and only if F is a power-law function.

Our empirical results only speak to our data from 2019–2025, like any empirical study we cannot guarantee the relationship holds out of distribution after 2025. However, F has exhibited power-law-like behavior over many orders of magnitude, so we believe it is reasonable to conclude that F is a power-law function. If F is indeed a power-law, we have a causal model that says the growth rate of time horizon and compute are proportional. Therefore, under this model, if the growth rate of total R&D compute drops by half, then the growth rate of time horizon will also drop by half.

3 Empirics

In this section, we test an empirical prediction of the theory as a sanity check on our results. In particular, we study whether Lemma 1 holds, i.e., that $\frac{\partial \log Y}{\partial \log C} \Big|_A$ is constant⁸. The following proposition gives us a simple test for this.

Proposition 2. $\frac{\partial \log Y}{\partial \log C} \Big|_A \text{ is constant} \iff \log Y = \alpha + \gamma \log A + \gamma \log C.$

Proof. See Section B.3.

The equation $\log Y = \alpha + \gamma \log A + \gamma \log C$ implies that, for any fixed algorithm, $\log Y$ vs. $\log C$ should be linear. Therefore, we test this by taking families of language models that have the same algorithms but with varying compute. We selected the Llama 3.1 and Qwen 2.5 families, since these models have multiple versions with varying compute, while the training algorithms are held constant (there are minimal distillation or data-quality changes). We measure Y via imputation from benchmarks using the methodology from Kwa [2025], except for the Llama family, where we recorded the METR time horizon on the METR task suite directly.⁹ However, this direct measurement was conducted somewhat less carefully than is standard for METR’s time horizon evaluations and should therefore be given lower weight.

⁸Note that the observations that follow will not provide distinguishing evidence for our model relative to other possible models for which $\frac{\partial \log Y}{\partial \log C} \Big|_A$ may be constant.

⁹We exclude Qwen 2.5 0.5B for GPQA, as 1.5B approaches the guessing bound on the benchmark.

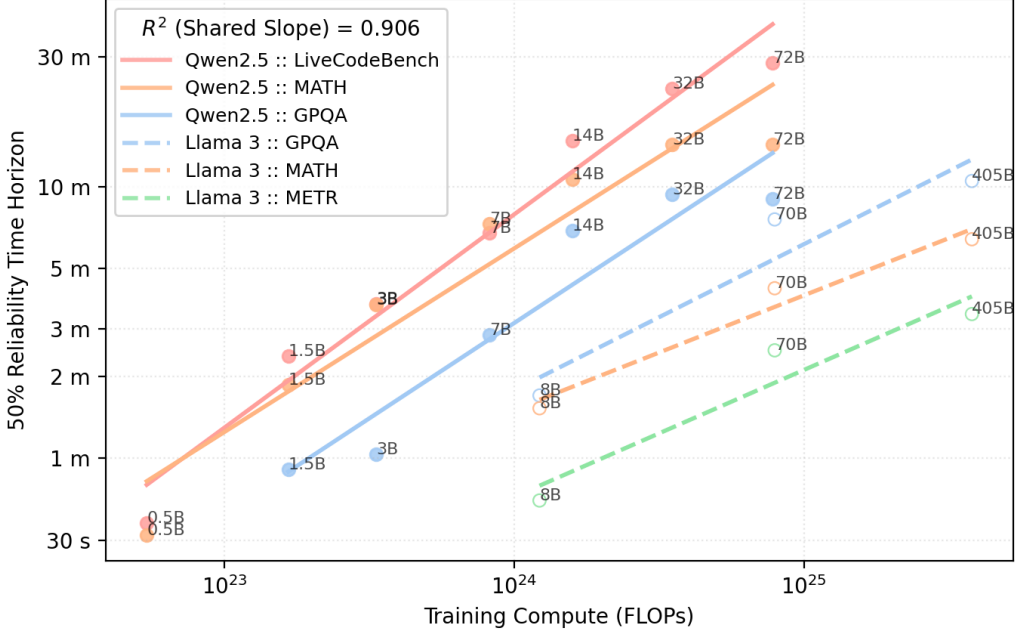


Figure 3: Log-log fit of time horizon vs. compute.

Figure 3 shows the fit. The slopes across models and benchmarks are fairly similar, and the linear fit appears reasonable, albeit with a leveling off at the top end. To summarize the results quantitatively, we estimate the equation

$$\log \text{ time horizon} = \text{constants for each model-benchmark} + \beta \log \text{ training compute}$$

and report the partial R^2 that the model achieves versus just the constants. We call this the shared slope R^2 as we are only fitting a single slope for all 30 data points. The shared slope R^2 would be 1 if and only if all model-benchmarks have the same log time horizon vs. log training compute slope and a linear fit was perfect. We obtain a shared slope R^2 of 0.906. An equivalent and visual way to evaluate this shared slope is to residualize both time horizon and training compute by model-benchmark and compute the R^2 in the standard way, which is done in Appendix Section E.

One possible explanation for the leveling off at the top in Figure 3 is that compute-optimality is not held fixed. Llama 3, for example, is trained on the same pre-training corpus regardless of model size: 8B \rightarrow 70B \rightarrow 405B just increases the corresponding number of parameters. This is plausibly different from what has happened for models in the METR sample, where the ratio of training data to parameters has likely been held closer to constant. To adjust for this effect, we replot the same relationship, but instead of using raw compute, we use the amount of compute needed to reach the same loss if the training were compute-optimal.¹⁰ We make this adjustment using the Chinchilla loss formula [Hoffmann et al., 2022] to calculate the training loss for each model given its training dataset size and number of parameters. We then calculate the minimum amount of compute needed to reach that same loss.

¹⁰The important part is not that the models in the METR sample were themselves compute-optimal, but that the degree of compute optimality was being held constant.

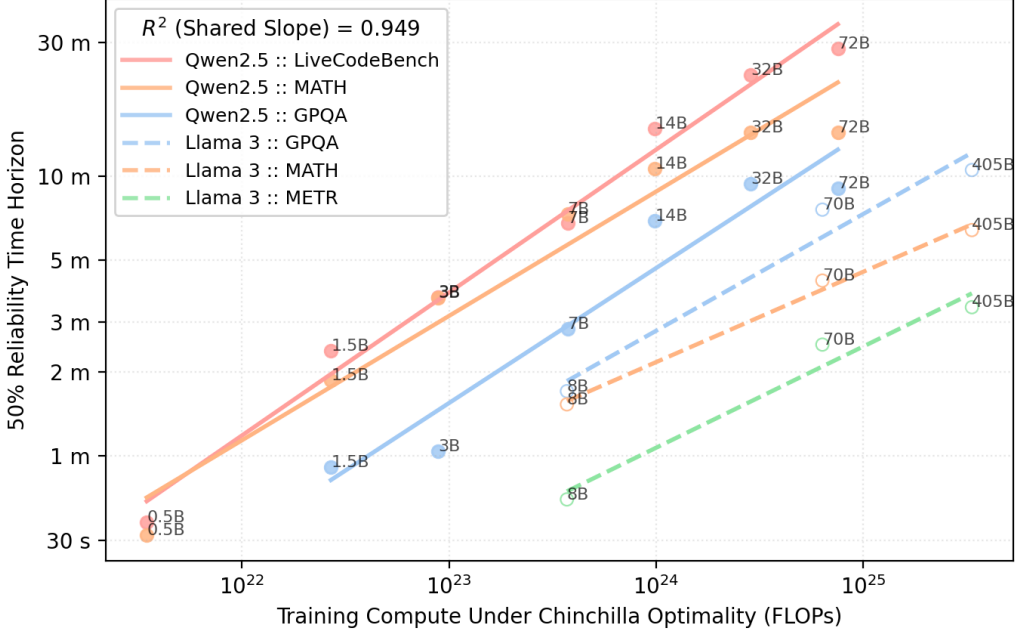


Figure 4: Re-estimated log-log fit, using compute adjusted for compute-optimality.

The resulting fit (Figure 4) is better, with the shared slope R^2 increasing to 0.949.

Overall, the linear fit (in log space) is not perfect, but is reasonably accurate over about 4 orders of magnitude of training compute. Therefore, our interpretation is that the experimental evidence is broadly consistent with the theory, but since it comes from only 2 model scale-ups and there is some evidence of concavity, we do not consider it decisive evidence. However, for our forecasts to be accurate, we need $\left. \frac{\partial \log Y}{\partial \log C} \right|_A$ estimated in 2018–2025 to be a good estimate of $\left. \frac{\partial \log Y}{\partial \log C} \right|_A$ from 2025 onwards. If the linear fit was perfect, this would imply that these terms are exactly equal, but even with some error our forecasts would still be reasonably accurate. Appendix Section F checks the robustness of our forecasts by allowing our estimate of $\left. \frac{\partial \log Y}{\partial \log C} \right|_A$ to fall as C increases.

4 Forecasting

Proposition 1 gives us the following relationship between compute growth and time horizon growth.

$$\text{future horizon growth} = \frac{\text{past horizon growth}}{\text{past compute growth}} \cdot \text{future compute growth} \quad (7)$$

We can estimate the past growth rate ratio from the data in Figure 2. We subset this data to only OpenAI, as they have the largest data coverage on both time horizon and compute. Once we have this estimated, equation 7 lets us convert from an implied growth rate change in compute to an implied growth rate change in time horizon.

To get an estimate of implied future compute growth, we use OpenAI’s own forecasts of their compute spend up to 2030 as reported in Efrati and Muppidi [2025] and then assume compute growth stabilizes at the 2029-2030 rate. To account for computing improvements, we assume that FLOPs per \$ improves at the same rate per year it did from 2018–2025.

Figure 5 plots the projected compute path, along with the historical path we described in previous sections. The ‘Compute Trend Continues’ line shows estimated trend of compute if OpenAI compute

continues to grow at the 2018–2025 rate, while the ‘OpenAI Compute Projection + Extrapolation’ grows at a similar rate until about 2028 at which point compute growth slows.¹¹

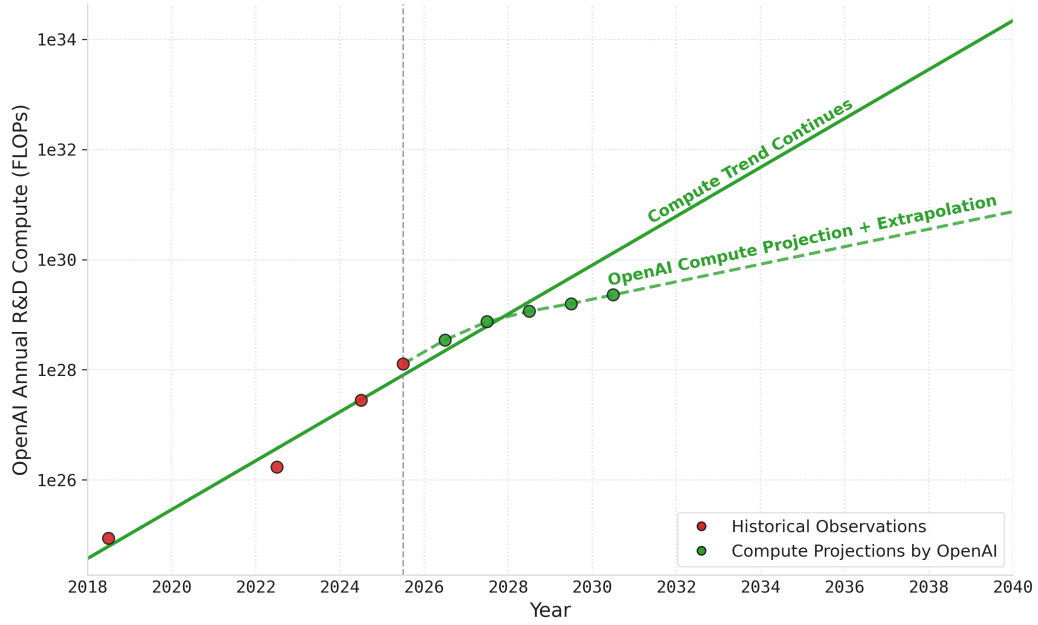


Figure 5: OpenAI projected compute.

Figure 1 shows the corresponding 50% time horizon path. Figure 6 shows the corresponding 80% time horizon path.

¹¹Note the difference in levels from 2025-2027 is due to ‘Compute Trend Continues’ starting from the trend line since it is a linear extrapolation, while ‘OpenAI Compute Projection + Extrapolation’ starts from the last point since it is a piecewise linear fit between points. This is not material for our time horizon forecast, which depends only on the slope.

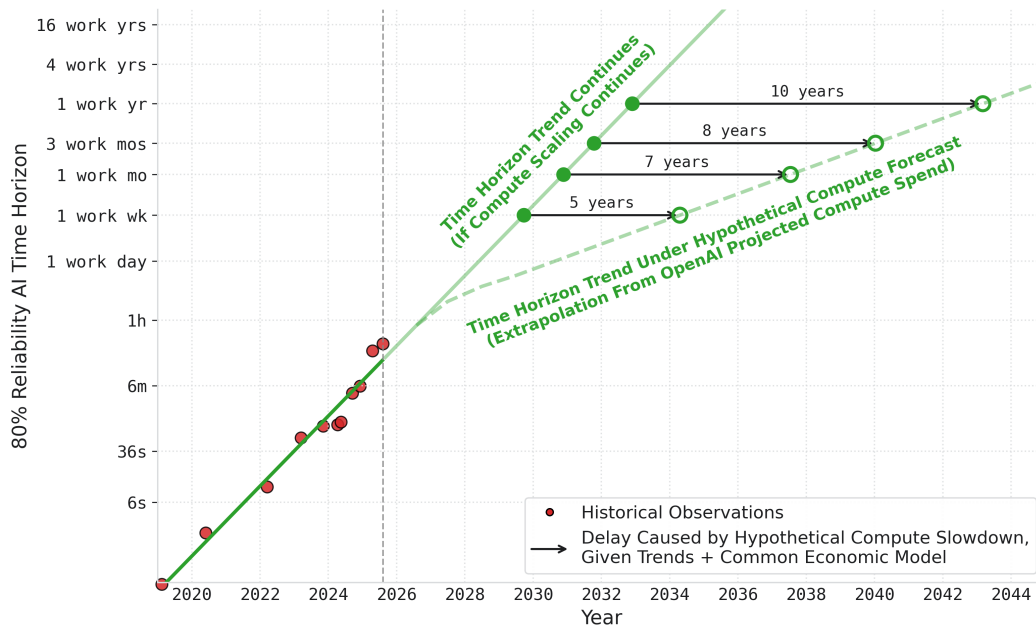


Figure 6: 80% reliability time horizon forecast.

Notice that Figure 6 has longer delays than Figure 1. The intuition for this is that because 50% time horizon is ahead of 80% time horizon, the 50% time horizon gets to spend more relative time in the current fast growth mode than the slow growth mode. This explanation is generally why we see longer delays for higher time horizons and higher reliability. Figure 7 captures this dynamic in a graph, where we display the delay introduced for various time horizons at various reliabilities.

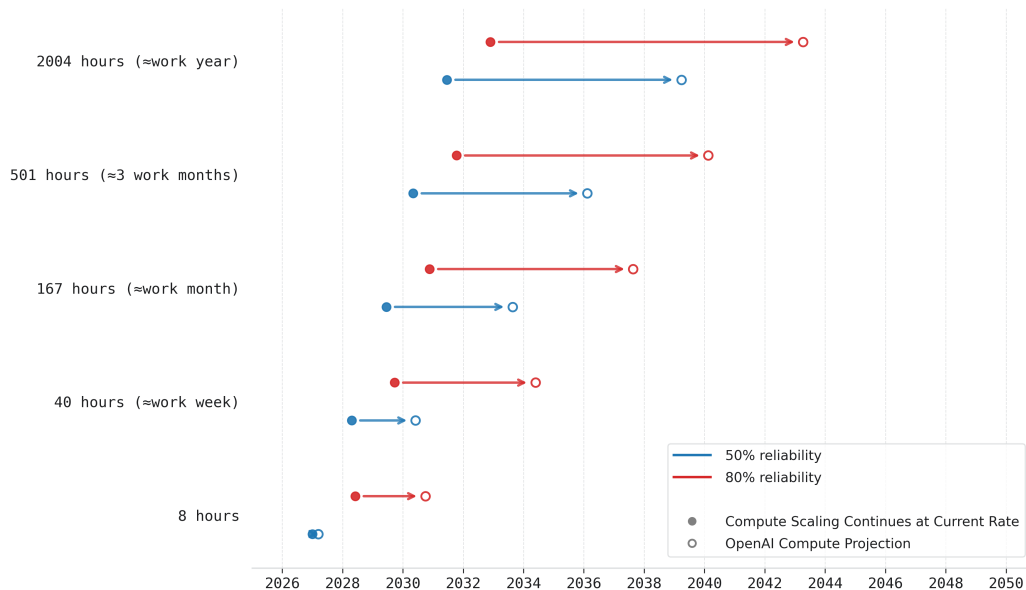


Figure 7: Implied milestone delay from compute slowdown

4.1 Webapp

We release a web app <https://timehorizonforecast.com/> where users can convert any compute scaling path to the corresponding path for METR time horizon. The compute path can either be supplied in FLOPs directly or in 2025 USD.¹²

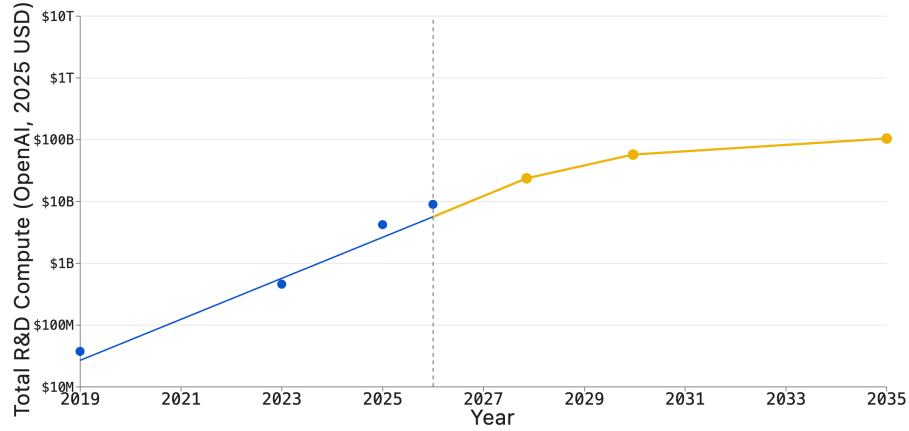


Figure 8: Example compute path in USD on the Web App.

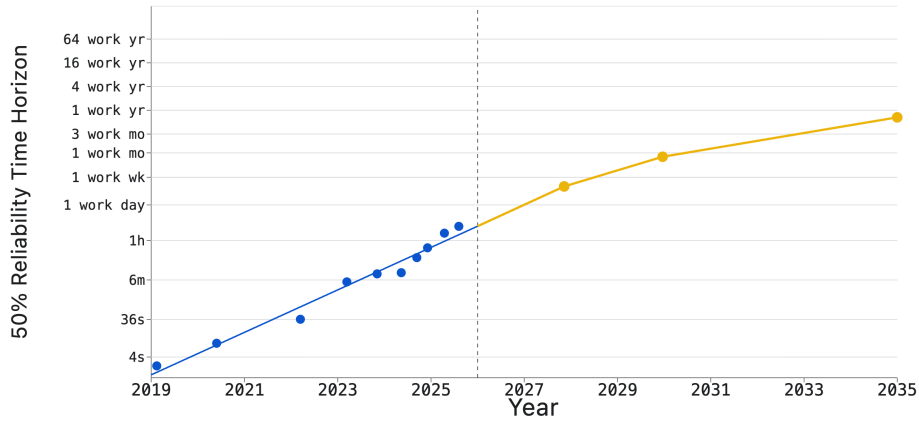


Figure 9: Example METR time horizon path implied by compute path in Figure 8.

¹²If the path is supplied in USD, we use the same technique described previously to convert to FLOPs, where we assume that FLOPs per dollar improves at the 2018–2025 rate.

A Author contributions

Parker Whitfill produced theoretical results and contributed the large majority of data analysis and draft writing. **Ben Snodin** collected time horizon data necessary for the figures in section 3. **Joel Becker** gave guidance on the project and contributed to data analysis and draft writing.

B Proofs

B.1 Proof of Lemma 1

Rearranging (6) yields

$$\left. \frac{\partial \log Y}{\partial \log C} \right|_A = \frac{\frac{d \log Y}{dt}}{\frac{d \log C}{dt} \left(1 + \frac{\lambda}{\beta}\right)}. \quad (8)$$

Stylized Fact 1 implies $\frac{d \log Y}{dt}$ is constant and Stylized Fact 2 combined with our constant training compute proportion assumption implies $\frac{d \log C}{dt}$ is constant. Therefore, the entire right-hand side of (8) is constant. Hence the left-hand side is constant. \square

B.2 Proof of Proposition 1

By Lemma 1, $\left. \frac{\partial \log Y}{\partial \log C} \right|_A$ is constant. Therefore, let

$$c = \left. \frac{\partial \log Y}{\partial \log C} \right|_A \left(1 + \frac{\lambda}{\beta}\right).$$

Using (6), we have

$$\frac{d \log Y}{dt} = c \frac{d \log(E + C)}{dt}.$$

\square

B.3 Proof of Proposition 2

(\Rightarrow) Recall,

$$\left. \frac{\partial \log Y}{\partial \log C} \right|_A = \frac{AC F'(AC)}{F(AC)} = \frac{d \log F(AC)}{d \log(AC)}.$$

Therefore, if $\left. \frac{\partial \log Y}{\partial \log C} \right|_A \equiv \gamma$ is constant, then

$$\frac{d \log F(AC)}{d \log(AC)} = \gamma.$$

Integrating with respect to $\log(AC)$ gives

$$\log F(AC) = \alpha + \gamma \log(AC),$$

hence $\log Y = \alpha + \gamma(\log A + \log C)$.

(\Leftarrow) If $\log Y = \alpha + \gamma \log A + \gamma \log C$, then holding A fixed yields

$$\left. \frac{\partial \log Y}{\partial \log C} \right|_A = \gamma.$$

\square

C Adding Labor to Algorithmic Progress

In the main text, we assumed that the only input to algorithmic progress research is experimental compute E , as in equation (2). Here we briefly sketch how the results might change if algorithmic ideas also depend on human researcher labor L .

Suppose instead that the Jones law of motion has as input a Cobb-Douglas production function between compute and labor.

$$\frac{dA}{dt} \propto A^{1-\beta} (E^{1-\alpha} L^\alpha)^\lambda, \quad 0 < \alpha < 1, \lambda, \beta > 0. \quad (9)$$

As before, after enough time the log derivative of A with respect to t converges to

$$\frac{d \log A}{dt} = \frac{\lambda}{\beta} \left((1-\alpha) \frac{d \log E}{dt} + \alpha \frac{d \log L}{dt} \right). \quad (10)$$

Substituting equation (10) into equation (3) gives

$$\frac{d \log Y}{dt} = \frac{\partial \log Y}{\partial \log C} \Big|_A \left[\frac{d \log C}{dt} + \frac{\lambda}{\beta} \left((1-\alpha) \frac{d \log E}{dt} + \alpha \frac{d \log L}{dt} \right) \right]. \quad (11)$$

Maintaining the assumption from the main text that experimental compute and training compute grow at similar rates, then we get

$$\begin{aligned} \frac{d \log Y}{dt} &\approx \frac{\partial \log Y}{\partial \log C} \Big|_A \left[\frac{d \log(E+C)}{dt} + \frac{\lambda}{\beta} \left((1-\alpha) \frac{d \log(E+C)}{dt} + \alpha \frac{d \log L}{dt} \right) \right] \\ &= \frac{\partial \log Y}{\partial \log C} \Big|_A \left[\left(1 + \frac{\lambda}{\beta} (1-\alpha) \right) \frac{d \log(E+C)}{dt} + \frac{\lambda}{\beta} \alpha \frac{d \log L}{dt} \right]. \end{aligned} \quad (12)$$

First, notice that if $\frac{d \log(E+C)}{dt}$ and $\frac{d \log L}{dt}$ grow at similar rates, then we get the same equation from the main text. This just says that if compute slowdowns are paired with labor slowdowns, then the main text equation is accurate. This is plausible because both are driven by the same underlying investment process.

Second, even if we model compute slowdowns as separate from labor slowdowns, we get fairly similar results. Ho and Whitfill [2025] estimate that at OpenAI, $\alpha \approx .67$, $\frac{\lambda}{\beta} \approx .95$, and we estimate

$\frac{\partial \log Y}{\partial \log C} \Big|_A \approx .454$ in the main paper. Finally, [Epoch AI, 2025] finds that OpenAI staff has been 1.6x-ing a year. Putting these terms together, a back of the envelope calculation implies that dropping compute growth by half would drop time horizon growth by $\approx 45\%$ instead of the full 50%. Therefore, while including staff does break the perfect proportionality result, the predictions do not change by much.

D Training And Experimental Compute

We obtained equation (6) from equation (5) by assuming that C and E move proportionally with total R&D compute $E+C$. Here we show exactly what changes when we drop that assumption.

Write the shares

$$s_C := \frac{C}{E+C}, \quad s_E := \frac{E}{E+C}.$$

Then the following identity holds:

$$\left(\frac{d \log C}{dt} + \frac{\lambda}{\beta} \frac{d \log E}{dt} \right) = \left(1 + \frac{\lambda}{\beta} \right) \frac{d \log(E+C)}{dt} + \underbrace{\left(s_E - \frac{\lambda}{\beta} s_C \right) \left(\frac{d \log C}{dt} - \frac{d \log E}{dt} \right)}_{:= \varepsilon_t}.$$

Plugging into equation (5) gives

$$\frac{d \log Y}{dt} = \frac{\partial \log Y}{\partial \log C} \Big|_A \left[\left(1 + \frac{\lambda}{\beta} \right) \frac{d \log(E+C)}{dt} + \varepsilon_t \right]. \quad (13)$$

When is the extra term small or zero?

- If $d \log C/dt \approx d \log E/dt$. This is the constant proportions of total R&D compute assumption we make in the main paper.
- If $s_E \approx \frac{\lambda}{\beta} s_C$. Since the returns to growing E are $\frac{\lambda}{\beta}$, intuitively this says s_E divided by its return to effective compute equals s_C divided by its return to effective compute (1). This would occur in a simple Cobb-Douglas model, although it does not formally occur here without further assumptions.

Both conditions seem plausible and if either of them is true then we have zero error. Moreover, the error is proportional to the differences of these terms, so even if equality does not hold exactly in practice, it may hold approximately.

Since total R&D compute is necessarily a sum of experimental and training compute, we suspect that using this as the input measure is relatively accurate. We suspect that this is more accurate than using training compute estimates from [Sevilla and Roldán, 2024] and then inferring experimental compute by the proportionality assumption.

E Shared Slopes

In Section 3 we reported the shared slope R^2 . Here we show visually what this fit corresponds to. We residualize both log time horizon and log training compute by model-benchmark. For example, residualized log time horizon equals log time horizon minus the mean log time horizon among the model-benchmark group. By the Frisch-Waugh theorem, this is equivalent to regressing log time horizon on a single log training compute slope term and model-benchmark constants.

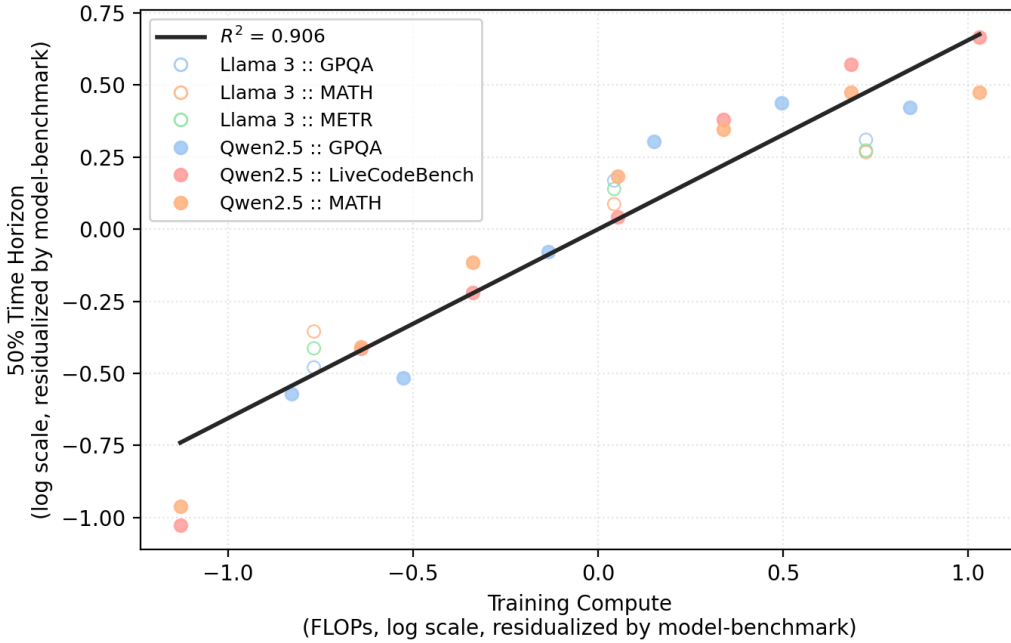


Figure 10: Residualized version of Figure 3.

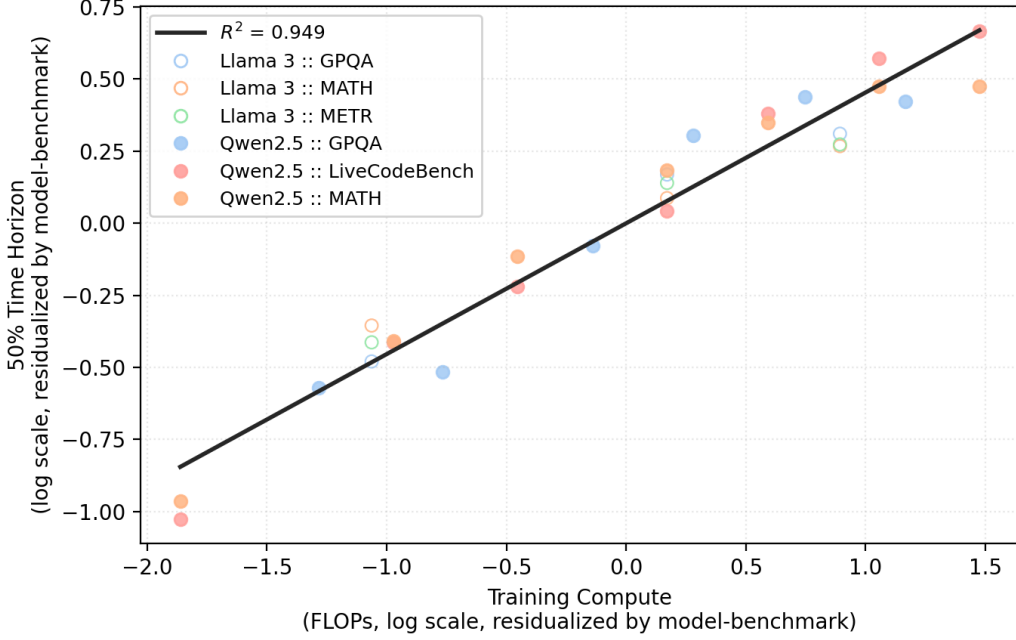


Figure 11: Residualized version of Figure 4.

F Forecasting Time Horizon Under Concavity

Our results in Figure 4 could reasonably be viewed as weak evidence that $\left. \frac{\partial \log Y}{\partial \log C} \right|_A$ declines over time. As a robustness check to our main results, we go back to the same data in Figure 4 and estimate via NLS:

$$\log Y = \text{model-benchmark constants} + \beta \left(\frac{X^\rho - 1}{\rho} \right)$$

In this model, $\left. \frac{\partial \log Y}{\partial \log C} \right|_A = \beta C^\rho$, which generalizes from the $\rho = 0$ case in the main text. This more flexible form allows us to fit slightly better, with a partial R^2 of 0.972. Moreover, our estimate $\rho < 0$ so $\left. \frac{\partial \log Y}{\partial \log C} \right|_A$ declines over time, giving us concavity.

To account for this concavity in our forecast, we return to equation 6 and substitute in our estimated elasticity

$$\begin{aligned} \frac{d \log Y}{dt} &= \left. \frac{\partial \log Y}{\partial \log C} \right|_A \left(1 + \frac{\lambda}{\beta} \right) \frac{d \log(E + C)}{dt} \\ &= \beta C^\rho \left(1 + \frac{\lambda}{\beta} \right) \frac{d \log(E + C)}{dt} \end{aligned}$$

Given our path of total R&D compute from 2018–2025, we then estimate $\frac{\lambda}{\beta}$ to make the predicted values for $\log Y$ most closely match what we observe in the data.¹³

Figure 12 shows a version of Figure 1 that incorporates concavity in F . Interestingly, time-horizon milestone delays lengthen, with 1-week time horizon being delayed by 7 years instead of 2 years in Figure 1. The intuition is that introducing concavity means the compute needed to reach a given

¹³Note that, because of the concavity, we explicitly assume how training compute and total compute relate: we assume training compute is $\frac{1}{10}$ following You [2025].

capability level rises as you need to overcome the decreasing returns. We discussed earlier in the paper that harder capability levels are more delayed by a compute slowdown. Moreover, this effect bites harder the more concave the returns are. Therefore, our main results under no concavity provide a lower bound on the delay introduced if there is some concavity.¹⁴

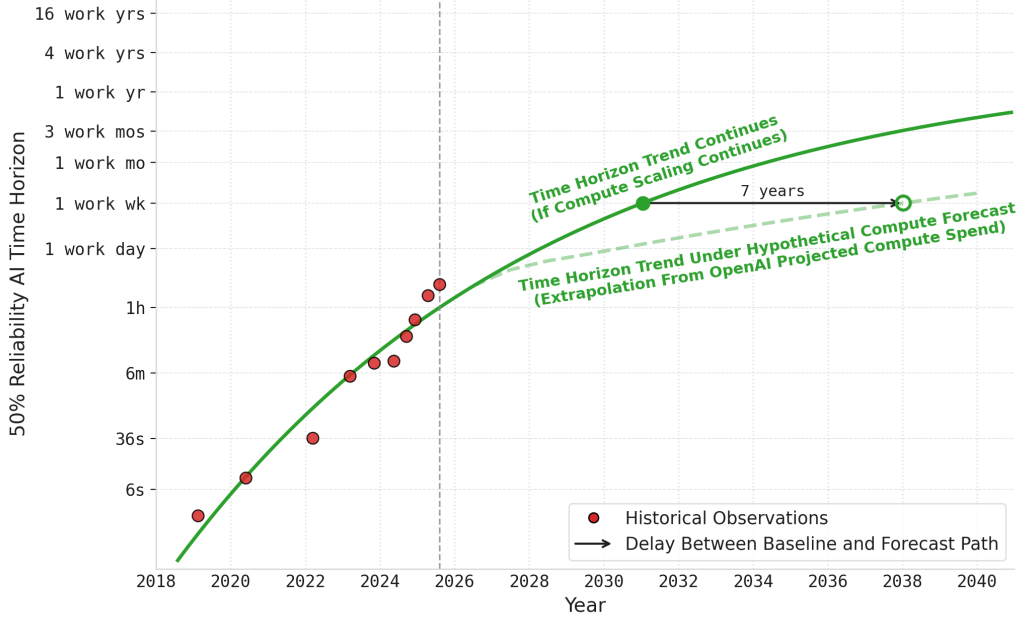


Figure 12: Concave Version of Figure 1

G Algorithmic Progress

The framework also implies a back-of-the-envelope estimate of algorithmic progress. Starting from (3),

$$\frac{d \log Y}{dt} = \frac{\partial \log Y}{\partial \log C} \bigg|_A \left(\frac{d \log C}{dt} + \frac{d \log A}{dt} \right),$$

solve for the algorithmic progress rate:

$$\frac{d \log A}{dt} = \frac{1}{\frac{\partial \log Y}{\partial \log C} \bigg|_A} \frac{d \log Y}{dt} - \frac{d \log C}{dt}. \quad (14)$$

We already estimated $\frac{\partial \log Y}{\partial \log C} \bigg|_A$ in Figure 4, $\frac{d \log Y}{dt}$ can be directly estimated in the METR data, while $\frac{d \log C}{dt}$ can be estimated in the METR data by merging it with training compute estimates from Epoch AI [2025]. For these estimates, we consider all non-Chinese models in the METR data that have compute estimates, not just those from OpenAI.

The following table shows the estimate of algorithmic progress we get from our data.

Table 1: Algorithmic progress implied by the METR p50% horizon.

$d \ln A / dt$	95% CI
1.265	[0.667, 3.614]

Confidence interval from 10000 paired METR resamples with Normal draws around the clustered elasticity estimate.

¹⁴Symmetrically, if returns are increasing over time, then the main results are an upper bound.

The point estimate corresponds to roughly $3.5\times$ per year, which closely matches the estimate in Ho et al. [2024]. One advantage of this estimate over Ho et al. [2024] is that it includes post-training and measures progress towards an economically relevant term, time horizon. However, because we estimate $\left.\frac{\partial \log Y}{\partial \log C}\right|_A$ using only Qwen-2.5 and Llama-3.1, we should be cautious about the external validity of these estimates.

H Acknowledgments

Thanks to Hjalmar Wijk, Phil Trammell, Thomas Kwa, David Rein, Charles Foster, Nikola Jurkovic, Ryan Greenblatt, Anson Ho, Jacob Hilton, David Farhi, Tom Cunningham, Beth Barnes and others for helpful comments.

References

- Ben Cottier, Robi Rahman, Loredana Fattorini, Nestor Maslej, and David Owen. The rising costs of training frontier AI models, 2024. URL <https://arxiv.org/abs/2405.21015>.
- Tom Davidson. Will compute bottlenecks prevent a software intelligence explosion?, 2025. URL <https://www.forethought.org/research/will-compute-bottlenecks-prevent-a-software-intelligence-explosion>. Accessed: 2025-11-20.
- Yafah Edelman and Anson Ho. Compute scaling will slow down due to increasing lead times, 2025. URL <https://epoch.ai/gradient-updates/compute-scaling-will-slow-down-due-to-increasing-lead-times>. Accessed: 2025-10-14.
- Amir Efrati and Sri Muppidi. OpenAI’s \$350 billion computing cost problem. *The Information*, September 2025. URL <https://www.theinformation.com/articles/openais-350-billion-computing-cost-problem>. Accessed: 2025-11-14.
- Epoch AI. Data on AI models, 07 2025. URL <https://epoch.ai/data/ai-models>. Accessed: 2025-11-08.
- Anson Ho and Parker Whitfill. The software intelligence explosion debate needs experiments, 2025. URL <https://epoch.ai/gradient-updates/the-software-intelligence-explosion-debate-needs-experiments>. Accessed: 2025-11-19.
- Anson Ho, Tamay Besiroglu, Ege Erdil, David Owen, Robi Rahman, Zifan Carl Guo, David Atkinson, Neil Thompson, and Jaime Sevilla. Algorithmic progress in language models, 2024. URL <https://arxiv.org/abs/2403.05812>.
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022. URL <https://arxiv.org/abs/2203.15556>.
- Charles I Jones. R&D-based models of economic growth. *Journal of Political Economy*, 103(4): 759–784, 1995.
- Thomas Kwa. How does time horizon vary across domains?, 2025. URL <https://metr.org/blog/2025-07-14-how-does-time-horizon-vary-across-domains/>. Research blog post.
- Thomas Kwa, Ben West, Joel Becker, Amy Deng, Katharyn Garcia, Max Hasin, Sami Jawhar, Megan Kinniment, Nate Rush, Sydney Von Arx, et al. Measuring AI ability to complete long tasks. *NeurIPS*, 2025.

Jaime Sevilla and Edu Roldán. Training compute of frontier AI models grows by 4-5x per year, 2024. URL <https://epoch.ai/blog/training-compute-of-frontier-ai-models-grows-by-4-5x-per-year>. Accessed: 2025-10-14.

Jaime Sevilla, Tamay Besiroglu, Ben Cottier, Josh You, Edu Roldán, Pablo Villalobos, and Ege Erdil. Can AI scaling continue through 2030?, 2024. URL <https://epoch.ai/blog/can-ai-scaling-continue-through-2030>. Accessed: 2025-10-14.

Josh You. Most of OpenAI's 2024 compute went to experiments, 2025. URL <https://epoch.ai/data-insights/openai-compute-spend>. Accessed: 2025-10-19.