Forecasting future gains due to post-training enhancements

Eli Lifland, Joel Becker, and Simeon Campos

This work has been done in the context of SaferAI's work on risk assessment. Equal contribution by Eli and Joel.

Disclaimer: this writeup is context for upcoming experiments, not complete work. As such it contains a lot of (not always well-justified) guess-work and untidy conceptual choices. We are publishing now despite this to get feedback.

If you are interested in this work — perhaps as a future collaborator or funder, or because this work could provide helpful input into e.g. risk assessments or RSPs — please get in touch with us at joel@qallys.com and/or simeon@safer-ai.org.

Summary

1. A recent report documented how the performance of AI models can be improved after training, via post-training enhancements (PTEs) such as external tools, scaffolding, and fine-tuning. The gain from a PTE is measured in compute-equivalent gains (CEG): the multiplier on training compute required to achieve equivalent performance to a model combined with a PTE.
2. We are interested in understanding the contribution that PTEs make to AI system capabilities over time.
    a. This question in turn is motivated by SaferAI's work on quantitative risk assessments of frontier models. In particular, any risk assessment of open-sourcing models or of having closed-source models stolen or leaked should take into account PTEs. A system's capabilities will increase over time as PTEs are added to the system built on top of a given base model.
3. We extend a recent analysis of PTEs in order to understand the trend in CEG over time, arriving at very rough estimates for the rate of improvement of PTEs. **Our primary takeaways are that current data is insufficient and experiments are needed to better forecast the effects of PTEs, as described below.**
4. There are serious limitations in our preliminary analysis, including: problems with the CEG metric, many uninformed parameter estimates, and reliance on an ill-defined "average task".
5. High-priority future work includes running experiments to get more evidence on important uncertainties for our forecasts of capability gains due to PTEs. In particular, we think it will be important to understand how well different PTEs combine, as well as to directly study performance on benchmarks relevant to dangerous capabilities rather than relying on the CEG and average task abstractions.

In this write-up, we will:

1. Outline our methodology. (More.)
2. Present CEG estimates for various PTEs. (More.)
3. Aggregate total CEG, using subjective estimates of 'composability.' (More.)
4. Note limitations of our analysis and important future work. (More.)

Methodology

Definitions

Post-training enhancements (PTEs) are methods to improve upon a model's performance without training a new model. For the purpose of this analysis, and consistent with prior literature, we will operationally define a PTE as requiring a one-time cost of <=10% of training the original model and <=100x of the inference cost.[1] Examples include fine-tuning and scaffolding.

Compute-equivalent gain (CEG) of a PTE is the training compute that a base model would have required to improve benchmark performance by as much as the PTE, divided by the compute used to train the base model.

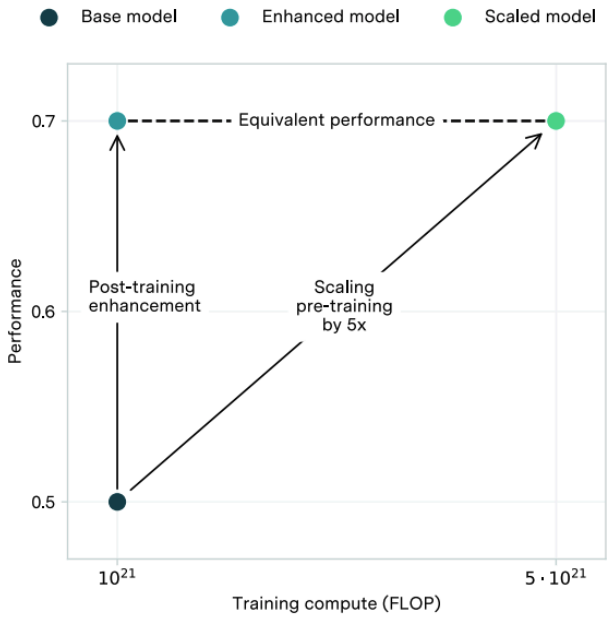
The **average task** is the hypothetical mean task that is an important input for forecasting risk levels.

Total CEG is the CEG due to *all* PTEs[2] available at some point in time, relative to some base model. See more details below.

CEG calculation

Davidson et al. uses the difference in compute between a lower-compute system with a PTE and a higher-compute system without the PTE (that has weakly greater performance than the lower-compute system with the PTE) to calculate CEG. (See their Figure 1 below; "the post-training enhancement improves performance by the same amount as increasing the training compute by 5x; so the CEG is 5.")

[1] These numbers are very arbitrary, open to arguments for adjusting. For the inference ones it really depends on the use case / threat model so it is hard to settle on a single number.
[2] Not literally all PTEs will be compatible (e.g. because some are substitutes, and because stacking them would go over the compute constraints). We mean the best combination of ones that is feasible within the compute constraints.



Note that this is a soft lower bound for this particular evaluation; if a system with slightly less compute than the higher-compute system would still have achieved greater performance than the lower-compute system with the PTE, then the actual CEG value would be higher than that estimated by this method[3].

We use the above method for some of our new CEG values. For the HumanEval benchmark, we estimate a sigmoid curve, using this curve to estimate the compute required to reach different levels of performance. For other CEG values, we use a local approximation (log-linear) to what we imagine is an underlying sigmoid curve of benchmark performance in log compute.

Data collection

We compile data on CEG for various PTEs studied in public research papers. We attempt to adjust these CEG values in order to estimate values for an 'average task,'[4] rather than on the particular benchmarks tested in research papers. The adjustment was done in an ad-hoc rather than systematic manner due to the lack of data for most PTEs.

---

[3] However, note that ,without further experiments, this method may naively overestimate results as papers showcase tasks and settings where their method performs best. See the average task limitation section for more.

[4] The way we thought about the ad-hoc adjustments was: we split the benchmarks from which CEG values were derived into several clusters of categories: Math, coding, QA, Common-sense reasoning (CSR), facts, classification, aggregations of many benchmarks, and a few others. QA, CSR, Math, aggregations, and coding seem most important to us of the things we're measuring in terms of datasets of interest for forecasting risk, though the others can be proxies for the ones we care about. Let's very naively say that a CEG that helps for only one of these counts for ⅕ of it in the 'average', 2 -> ⅖, etc. See below for visuals on how PTE varies by task type.

CEG aggregation

Within each category of PTEs, we specify highly subjective estimates of ‘composability’ of different PTEs. We use these to calculate the aggregate CEG from a given category of PTEs when only some PTEs within the category are activated. To calculate aggregate CEG across all PTE categories, we use an exponential weighting formula with another arbitrary composability constant.

Using our formula for aggregating CEG across PTE categories, and our estimates for aggregate CEG within a PTE category when only some PTEs are activated, we can estimate aggregate CEG for different combinations of PTEs. Then, using the release date of different PTEs, we can estimate aggregate CEG at different points in time. We adjust this historical trend in aggregate CEG based on various qualitative considerations.

Clarifying total CEG

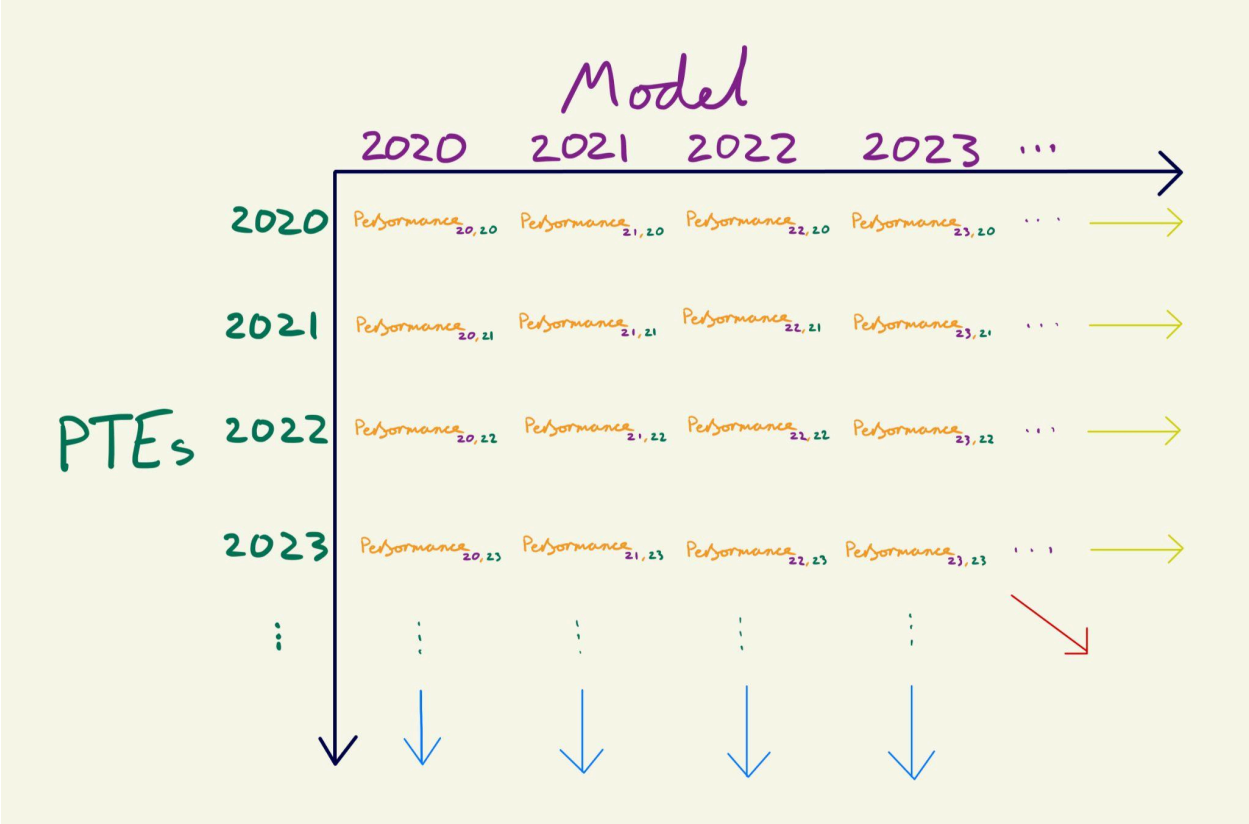


Figure 1: How performance changes with progress in both models and PTEs

The performance trend we care about

Since our motivation in this work is to understand how PTEs applied over time will increase performance of open-sourced or leaked AI systems, we are concerned with the performance trend given by combining a model released in year X with PTEs developed through years after X. **This is the blue trend in Figure 1 going from top to bottom.**

What total CEG measures

Ultimately, we care about the dangerous capabilities implied by performance levels in each cell. But since extrapolating performance for dangerous capabilities is difficult, CEG aims to anchor performance improvements against how much performance would increase with only scaling up compute. The idea is that if the improvement from (Model X, PTEs X) to (Model X, PTEs X+Y) is the same as (Model X, PTEs X) vs. (Model X + N OOMs compute, PTEs X), then Y PTEs have a total CEG of N OOMs.

This compares the blue top-to-bottom trend to a trend similar to the yellow left-to-right trend above (except without algorithmic improvements in training included in the yellow trend).

Note that we think there are serious issues with total CEG as a measure; see below.

Other things that would be valuable to measure

1. **Actual progress (red trend) as a reference point for the blue trend**: Arguably, comparing against actual progress might give better intuitions regarding performance. There might also be a case for using the yellow trend as a reference.
2. **Actual progress (red trend) compared to the yellow trend (progress holding PTEs fixed)**: This would be an important metric for understanding algorithmic progress of PTEs, in addition to understanding the blue trend.

Issues with our measurement of total CEG

There are many gaps between total CEG and what we've been able to measure/forecast thus far. Experiments will be valuable to gain insight into each of the all of the yellow/red/blue trends as well as lessening the need for a reference trend. See also limitations and future work below.

CEG estimates

Below is a table of all the post-training enhancements considered in our analysis. We have heavily drawn upon Davidson et al., for both which PTEs to look into and how they should be categorized. We add some more PTEs, adjust some CEG values, break CEG values down by dataset, order by date, and include subjective estimates of 'composability' of various PTEs in order to produce a running total over time of the total CEG for each type of PTE.[5]

Our full list of CEG estimates categorized by benchmark and task type are in CEGs across datasets. Reasoning for the CEG values by dataset are mostly in Davidson et al.. Otherwise they are linked from the table cells. Some are unfortunately not documented well

[5] For now we are just hackily averaging CEG values together in a hacky sort of logarithmic average, like biasing toward lower ones. Should think about if this is a principled approach, really should at least apply it consistently. Also in general all the intuitive steps in this table should be reduced as much as possible. TODO.

yet, especially in cases where we modified or added CEG values for PTEs considered in Davidson et al..

You can find plots comparing these CEG values across PTEs, task types, and benchmarks in the Plots of CEG values appendix section.

Date	Technique	Explanation	~CEG reasoning: very rough estimates of CEG for 'average dataset', and composability with prev. PTEs	~CEG running total for type	Compute multipliers (1-time cost, inference cost)
			Tools		
12/2021	WebGPT *Tools*	Fine-tune a model to use a web browser to answer factual questions and provide citations.	1 *8 = 8 CEG: 8 ELI5: >15 TruthfulQA: >220 Seems much more helpful for QA than other types of tasks, but not *only* helpful for QA. So let's say it's 20 CEG for QA -> 20*(⅖) = 8 for avg dataset.	8	(~0.01%, 1)
12/2021	RETRO *Tools*	The model retrieves text that is similar to the text it is predicting, and uses it to inform its predictions.	8 * 2^.2 = 9 Composability:[6] .2 V. similar to WebGPT so don't expect much composition CEG: 2 Varies from 0 to 43 on next-word prediction tasks, but the 43 is on text with very similar text in the training corpus and the others are substantially lower.	9	(<3.3%, <1.1)
02/2023	Toolformer *Tools*	Fine-tune a model to use a calculator, a	9 * 7^.3 = 16 Composability: .3	16	(~0.01%, 1)

[6] Composability is hackily defined as the exponent for the lesser number in the multiplication

		Q&A system, a search engine, a translation system, and a calendar.	Overlap with WebGPT, with some added tools. CEG: ~7 >20 in benchmarks for factual knowledge, math, and temporal questions. CEG = 7 for QA, ~1 for translation		
Prompting enhancements					
05/2020	Few-shot prompting *Prompting enhancements* InstructGPT paper also used for data points	Provide a few solved examples to the model.	1 * 26 = 26 CEG: 26 ~26 in SuperGLUE, an aggregative benchmark. Highly varying CEG values on other tasks: 2 times >200 and 6 times 1-2.	26	(0, >5 and <50 in three examples)
01/2022	Chain of thought *Prompting enhancements* Lanham et al. also used for later data points.	Encourage a model to make its reasoning chain explicit.	3^.5 * 26 = 45 Composability: .5 We think it does combine with few-shot some but doesn't stack super well. CEG: ~3 Got >9 on many benchmarks in the original paper, but more like ~1.5 on several tasks in a more recent paper using less undertrained models. It's debatable which of these is more relevant for our purposes, will take in between.	45	(0, 10)
Scaffolding enhancements. CEG calcs for HumanEval and HotpotQA					
07/2022	CodeT	Have a model generate test	1 * 2 = 2	2	

	Scaffolding enhancements	cases for code samples generated, choosing the best solution.	CEG: ~4 on HumanEval (coding). Let's say 2 for an average task.		
10/2022	ReAct *Scaffolding enhancements*	Have a model generate interleaving reasoning traces and task-specific actions.	2^.7 * 2 = 3 Composability: .7 Think it adds a decent amount on top of CodeT CEG: ~2 ~2x HumanEval, ~4x HotpotQA. Choose lower due to average task.	3	
12/2022	Parsel *Scaffolding enhancements*	The model decomposes a complex task into natural language function descriptions, generates modular implementations for each, and searches over combinations of these implementations by testing against constraints.	3 * 2 = 6 Composability: Builds on top of CodeT, so should just multiply by 4/2=2 CEG: ~4 ~7 on HumanEval including CodeT, ? on APPS. Then adjust down some to 4 as somewhat code-specific.	6	(0, ~32)
03/2023	Reflexion *Scaffolding enhancements*	A model makes use of reflection, memory, and evaluation to iteratively improve its output	6 * 5^.1 Composability: ~.1 Don't think it combines that well, Very similar to Parsel. CEG: ~10	7	(0, ~32)

			HumanEval: ~5 HotpotQA: ~86 So let's say ~5, taking lower for average task.		
10/2023	LATS *Scaffolding enhancements*	A model assigns sub-tasks to copies of itself, reads and writes to memory, has a chance to learn from their mistakes, etc.	7^.4 * 20 Composability: 0.4 We thought it wouldn't combine well in general but it apparently does with ReACT, at least on one dataset. CEG: ~20 CEG with ReACT on one dataset: ~80? Compromising between HumanEval, HotpotQA. Taking lower again for average task. ~20 CEG on HumanEval, ~400x on HotpotQA by itself vs. ~1800x with ReACT	44	(0, ~160 (for LATS at HumanEval))
10/2023	FireAct *Fine-tuning + Scaffolding*		Will leave the same as above for now since it requires a bigger model, and also is really a data point on how CEG aggregates. ~5 CEG on HotpotQA on top of ReACT	44	
Solution choice enhancements					
10/2021	Verification *Solution choice enhancements*	A verifier rates 100 candidate solutions and submits the one with the highest rating.	CEG: ~20 >26 on GS8MK benchmark, but adjusting down a little bit because it seems somewhat dataset-specific	20	(~0.05%, 200)

02/2022	AlphaCode sample selection *Solution choice enhancements*	Six techniques for choosing which coding solutions to submit out of 1000s of candidates.	20 * 3^.1 Composability: 0.1 Doesn't seem that composable CEG: ~3 ~6 on Codeforces problems, adjust down some since not that transferrable	22	(~0.45%, <2)
03/2022	Self-consistency / majority voting *Solution choice enhancements*	Sample multiple times then take the answer that is most common	1.5^.9 * 22 Composability: .9 It is pretty orthogonal to the other enhancements CEG: ~1.5 1.5-2 on various math/CSR benchmarks CEG calcs here.	32	
05/2023	Verification with process-based feedback *Solution choice enhancements*	Improves on a "outcomes based" verifier baseline by fine-tuning a verifier with "process based" feedback.	3^.1 * 1.5^.9 * 20 * 5 Composability: 0.1 with AlphaCode / self-consistency, .9 with self-consistency CEG: ~5 8 for MATH, on top of verification. Adjusting down to 5 for the same reason as verification above.	161	(~0.001%, ~1)
Data enhancements					
03/2022	Instruct GPT *Data enhancements*	Finetune a model on examples of humans following instructions; finetune against a reward model	CEG: ~30 >3900 at instruction following; >130 on some other NLP benchmarks; no gain on many NLP benchmarks.	30	(~0.3%, 1)

		trained to predict human preferences.	Chatbotting is the thing it was optimized for. It also gains a bunch on summarization and TruthfulQA, but little on commonsense reasoning. Overall it seems roughly fair to say average of 30x		
07/22	Generating your own fine-tuning data *Data enhancements*	Models write coding puzzles and solutions; solutions are automatically checked; fine-tune on correct solutions	Composability: 0 Some of the things above are ~strictly better if you put in the effort and compute, this is a way to do it cheaply. We're not really capturing that in our framework well right now. CEG: ~5 Requires automatic checking so reducing some, but could perhaps be done okay with verifiers. >22 in a coding benchmark, compared to a baseline with no finetuning for coding	134	(~0.04% , 1)
06/2023	Learning from a teacher model *Data enhancements*	Fine-tune a small model on detailed explanations produced by a larger model	Composability: 0 Not as relevant since it can't be used without a substantially bigger model. CEG: ~10 ~10 on a range of benchmarks	134	(~2.5%, 1)
08/2023	OctoPack *Data enhancements*	Fine-tune models on a large number of filtered git commits and multi-turn dialogues containing natural language and some code.	134 * 10^.1 Composability: 0.1 This doesn't really stack on top of SFT, we think. So we will set composability low CEG: ~10 Varied between ~4x and ~80x	169	(?, 1)

			for models on HumanEval. It is fairly code-specific so will adjust down but could be applicable to some other stuff. CEG calcs here.		

CEG aggregation

Methodology

We gathered the data from the overall table above into CEG over time, which tracks the aggregated CEG values for each category at each timestep.

To compose the aggregated CEG values for each category into a total CEG value at each timestep:

1. For each point in time, the aggregated CEG for each category (ad-hoc, using numbers from the above table) are sorted from highest to lowest as C_0, C_1, … C_4
2. Define an arbitrary composability constant M=.6
3. The overall CEG is (C_0 ^ (M^0)) * (C_1 ^ (M^1)) … * (C_4 ^ (M^4))

The motivation for this is that the overall CEG should always be at least as high as the highest category CEG, but there should of course be some diminishing returns for the rest.

Setting the composability constant

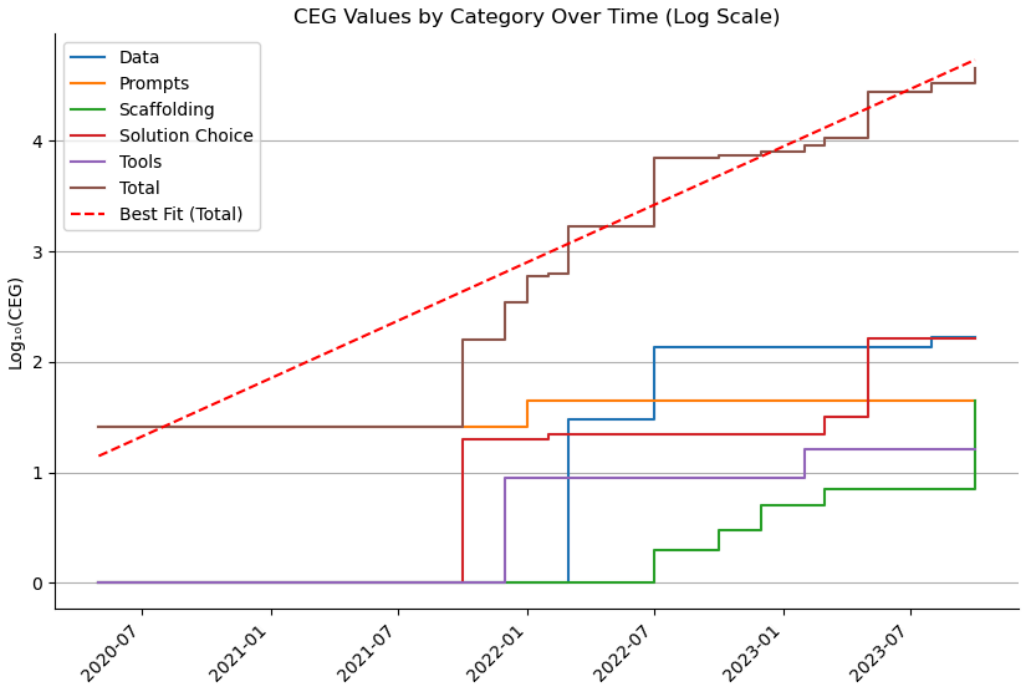
These points apply not only to the composability constant for across PTE groups, but also the composability values within PTE groups as in the overall table above.

1. **Points for lower composability**
    a. Even if some methods can theoretically be combined it might be too expensive to do so, at least for actors without access to huge amounts of compute. In practice, there aren't that many papers reporting results that combine many PTEs together in a way such that the CEG is easily measurable.
    b. CEG values appearing in published literature are selected for usefulness on the tasks/datasets they are tested on, and may not be as useful or even be able to be combined for use on other datasets (in some cases, this is very obvious; in other cases, less clear).
2. **Points for higher composability**
    a. We were quite pessimistic about the composability of LATS/ReACT and thought that LATS had basically just usurped ReACT, but in fact they seem to compose very well on HotpotQA i.e. it would have a constant of close to 1. This is just one data point but the first sample gives the most information.

We chose an estimate of .6 by (a) subjectively weighing these considerations (b) trying a few CEG combination spot checks and picking the median of what seems roughly right and (c) incorporating intuitions gained from the specific composability estimates that we gave for within categories above.

CEG over time

Using the above formula and composability constant, we estimate total CEG at different points in time.



Sensitivity analysis

The line of best fit above has a slope of **1 OOMs of CEG per year**. For various reasons — ambiguity about the right starting point, the disconnect between information we can find in the literature and the objects we want to measure, the trend above not obviously being log-linear, and the hacky choices made in the table above — we do not want readers to take this number too seriously.

Here are slopes for choices of composability constant that seem defensible:

Composability constant	.3	.45	**.6**	.8
Slope (OOMs of CEG/year)	0.6	0.8	**1**	1.6

Qualitative considerations

We would ideally want to estimate the rate of improvement in CEG with PTE compute held constant, but this is challenging without experimental data. We think that differences in PTE compute probably have a large effect. We are currently not adjusting for this in our quantitative methodology, which biases our estimates upwards for 2 reasons:

1. Compute usage on PTEs is generally going up over time, so the rate that CEG is going up is an overestimate of how much it would go up holding compute fixed.
    a. We think this is at least an issue with the scaffolding PTEs, not as sure for the others.
2. Our composability estimates lead to significant overestimation of the CEG gain that would happen with the compute level held constant. (If compute growth was paused, there would be further diminishing returns to new PTEs.) This is a huge deal in some cases, but it is a little double counting with (a) in 'Points for lower composability' above.

Still, there's value in a new method that is able to efficiently turn more compute into performance than was possible before. (Perhaps part of the reason more compute wasn't poured into the previous method was that it would be inefficient for performance.) We should keep this in mind in the absence of estimates with compute held constant.

We are probably missing out on many PTE improvements because their CEG can't be quantified and/or we didn't find them. While there are likely diminishing returns to combining more PTEs, this effect could still be very significant.

We expect the above considerations to roughly cancel out.

Another important factor, especially over a longer time period, is diminishing returns. This is not yet easy to observe, but would likely happen over time holding the base model for the PTEs constant. We should expect enhancements that help more on bigger models as opposed to smaller models being explored more over time, which hurts out ability to observe diminishing returns on a single model without more data and experiments.

At the extreme of diminishing returns, some important tasks may have effective 'maximum' levels of performance you can squeeze out with PTEs, capping the CEG. As an intuition pump, it would be practically impossible to find post-training enhancements that would make GPT-2 do as well as GPT-4 at competitive programming, capping its log10(CEG) at less than 4. We would expect similar effects on many tasks, though it would depend a lot on the capabilities of the base model and how effective scaling training compute is on the task.

Limitations and future work

Limitations

CEG as a measure

Over the course of our project, we realized that CEG has serious flaws.

Datasets with high CEG may just indicate that adding compute doesn't help much with respect to the measured skill, rather than indicating something about the 'absolute' level of gains from PTE relative to other datasets. Datasets with inverse scaling could lead to infinite CEG.

Some jumps in model performance enable qualitatively new capabilities that would be very hard to reach with a less capable base model, and vice versa for some post-training enhancements. This limits the usefulness of a compute-centered approach. It would probably be very hard to get GPT-6 with no internet access provided to do better than GPT-4 with internet access provided tasks requiring taking actions on the internet.

Relatedly, CEG could be extremely large in domains for which scaling is less helpful. This may reflect real phenomena: perhaps auto-regressive LLMs at current scale can have capabilities they inherently lack — mathematics, advanced domain expertise, long-horizon planning — greatly improved by PTEs, whereas capabilities that these models are already strong at — translation, code, summarization, classification, fact-remembering — can only be somewhat improved by PTEs. It is unclear how to best incorporate this consideration into forecasts, other than to potentially justify a long right tail.

Additionally, CEG becomes more problematic as the time horizon studied gets longer. When moving beyond local adjustments, it starts to be underdefined in a similar manner to the issues with effective compute.[7]

These limitations make it problematic in that a very high CEG is a bit of evidence for high performance but also substantial evidence for scaling slowing down (broadly, or in a domain) and/or (if you define CEG relatively strictly) just time passing moving away from the regime of local adjustments.

[7] Does CEG mean you literally just add GPUs and no other adjustments, even across OOMs of compute? If so, the CEG will probably get very high quickly. If the answer is you can't have new "fundamental algorithms" but you can make other adjustments to the training process (as with the TAI training requirements are defined in bio-anchors), how is the classification of fundamental algorithms defined, and how do you account for this in your CEG metrics? [Eli is eager for comments on this footnote if anyone might disagree about the severity of these issues]

Uninformed parameter estimates

We make many estimates which are quick educated guesses rather than grounded in empirical estimates or strong qualitative understanding. We hope to make progress on this via running experiments.

Average task

The 'average task' is not well-defined. Further, CEG values may vary greatly across datasets; if researchers are more likely to publish results for benchmarks on which a PTE does unusually well, our methods might naively overestimate CEG on typical tasks. (E.g. Minerva on MATH has an especially large CEG value, which is unrepresentative of the CEG value on other benchmarks.) For this reason, we ultimately think that for specific domains it's best to do as much analysis on specific benchmarks on that domain as possible.

Unspecified base model compute

The degree to which a given PTE boosts performance might depend on the pre-training compute of the underlying base model. For example, OpenAI has said that GPT-4 gets less value from fine-tuning than GPT-3.5.

Historical trend

The historical trend, even if it were cleanly estimated, may not be informative as to the future CEG. Current PTEs or domains might not be representative of future PTEs or domains.

Further limitations

See also the limitations section from Davidson et al.

Future work

The obvious and most informative next step is running experiments. This would provide evidence for how PTEs *combine*, which is crucial for our trend estimation and forecasting, and surprisingly absent from published research literature. Also, our experiments could keep base models fixed, which would aid comparison and help us forecast the right estimand.

Resources (e.g. code) for many PTEs are freely available online; running these experiments would be as simple as looping a benchmark performance calculation over benchmarks and combinations of PTEs. (Although note that this conceptually simple exercise might throw up significant practical difficulties.)

In experiments, we could more easily drop the CEG abstraction, measuring and forecasting trends in performance directly.

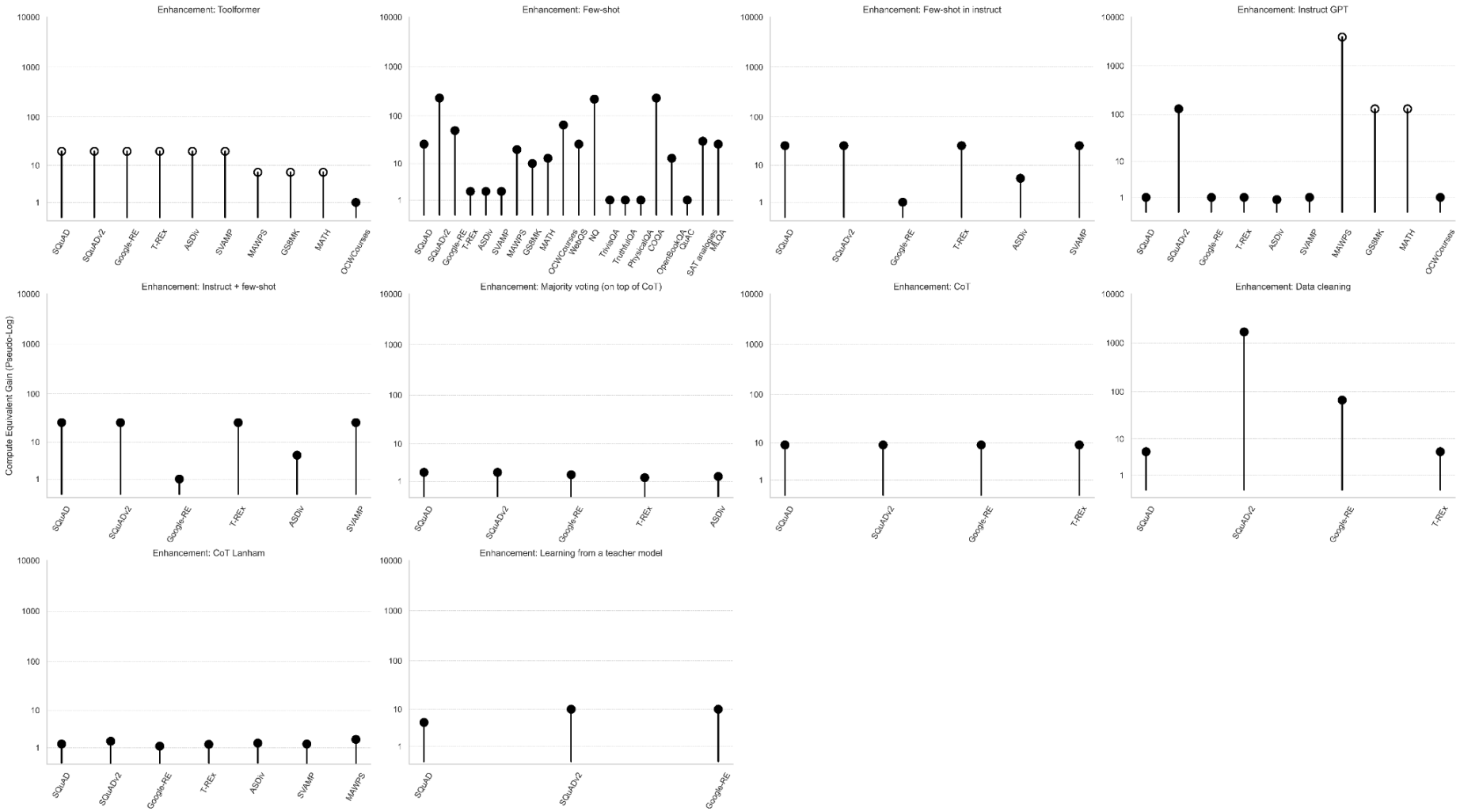
While there is some value in CEG especially in very local changes, we tentatively recommend that future work focus mostly on the implications of raw performance trends (especially if making

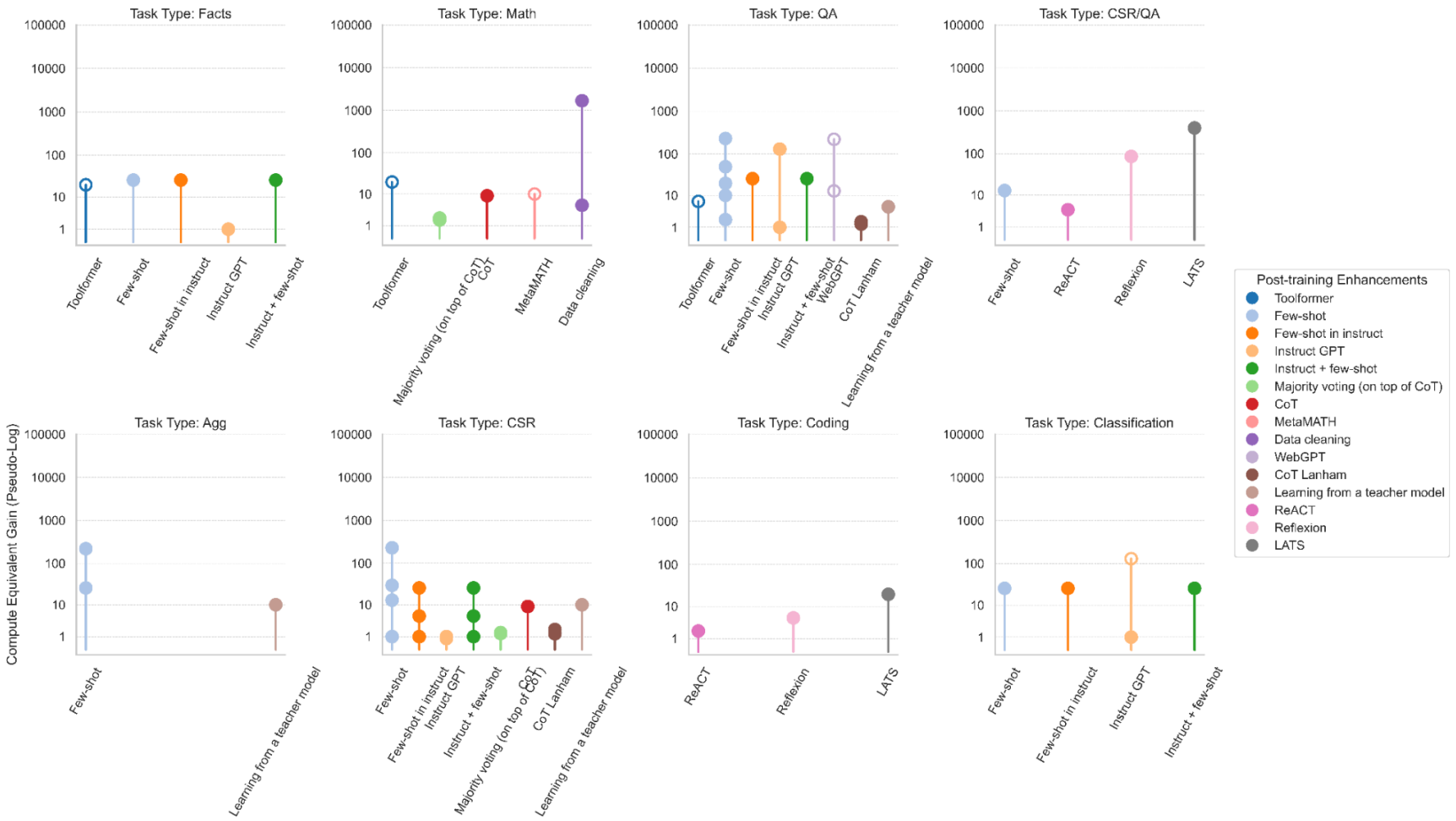
claims about performance changes over long time horizons) as CEG can be very hard to interpret and potentially misleading.

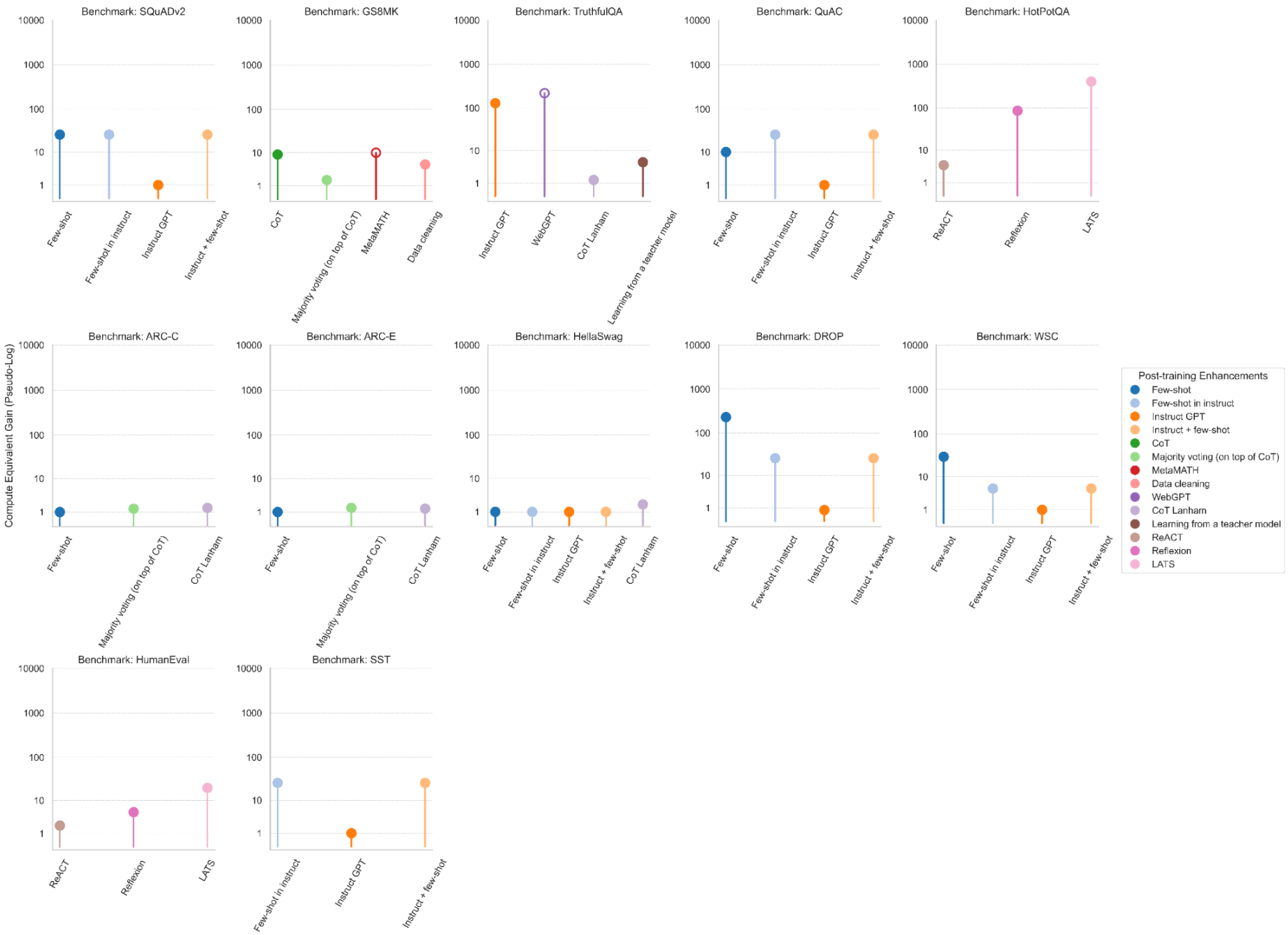
Appendix

Plots of CEG values

Broken down by PTE, then task type, then benchmark. The CEG values across task types and benchmarks are filtered for task types that have at least 3 recorded results.


Enhancement: Toolformer
Enhancement: Few-shot
Enhancement: Few-shot in instruct
Enhancement: Instruct GPT
Enhancement: Instruct + few-shot
Enhancement: Majority voting (on top of CoT)
Enhancement: CoT
Enhancement: Data cleaning
Enhancement: CoT Lanham
Enhancement: Learning from a teacher model
Compute Equivalent Gain (Pseudo-Log)
10000
1000
100
10
1
SQuAD
SQuADv2
Google-RE
T-REx
ASDiv
SVAMP
MAWPS
GSM8K
MATH
OCWCourses
WebQS
NQ
TriviaQA
TruthfulQA
PhysicalQA
CODA
OpenBookQA
QuAC
SAT analogies
MMLA


Task Type: Facts
Task Type: Math
Task Type: QA
Task Type: CSR/QA
Task Type: Agg
Task Type: CSR
Task Type: Coding
Task Type: Classification
Compute Equivalent Gain (Pseudo-Log)
100000
10000
1000
100
10
1
Toolformer
Few-shot
Few-shot in instruct
Instruct GPT
Instruct + few-shot
Majority voting (on top of CoT)
CoT
MetaMATH
Data cleaning
WebGPT
CoT Lanham
Learning from a teacher model
ReACT
Reflexion
LATS
Post-training Enhancements


Benchmark: SQuADv2
Benchmark: GS8MK
Benchmark: TruthfulQA
Benchmark: QuAC
Benchmark: HotPotQA
Benchmark: ARC-C
Benchmark: ARC-E
Benchmark: HellaSwag
Benchmark: DROP
Benchmark: WSC
Benchmark: HumanEval
Benchmark: SST
Compute Equivalent Gain (Pseudo-Log)
10000
1000
100
10
1
Few-shot
Few-shot in instruct
Instruct GPT
Instruct + few-shot
CoT
Majority voting (on top of CoT)
MetaMATH
Data cleaning
WebGPT
CoT Lanham
Learning from a teacher model
ReACT
Reflexion
LATS
Post-training Enhancements

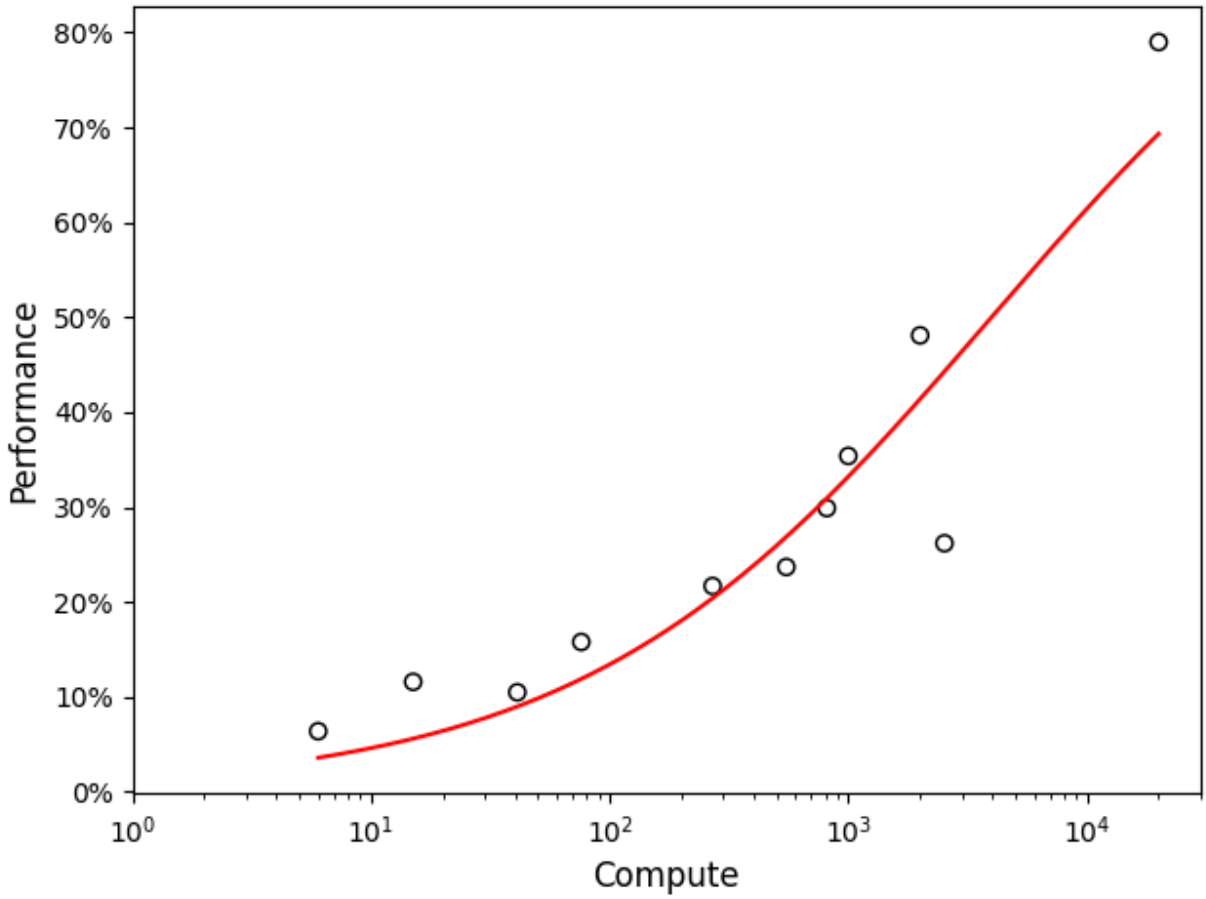
Explanations for CEG estimates

HumanEval

See [HumanEval Benchmark (Code Generation)] for most of the data we gather.

Training compute increase

We fit a baseline sigmoid as a function of training compute as done in [Extrapolating GPT-N performance] to more accurately estimate CEG values on HumanEval. See code for fitting the sigmoid [here], and the data below.



Sigmoid params:
k = 1.164308904678886
x0 = 3.5996951849783962

Compute estimates are from Parameter, Compute and Data Trends in Machine Learning

Date	Model	Performance (pass@1)	Approx. compute
Eval 02/2022	GPT-Neo-2.7B	6.4	6e21
02/2023	Llama-7B	10.5	4.1e22
Eval 02/2022	GPT-J-6B	11.6	1.5e22
02/2023	Llama-13B	15.8	7.6e22
02/2023	Llama-33B	21.7	2.7e23
02/2023	Llama-65B	23.7	5.5e23
04/2022	PaLM-540B	26.2	2.53e24
07/2023	LLaMA 2	29.9	8.13e23
06/2023	Inflection-1	35.4	~1e24 (just know as much or less than PaLM-540B)
06/2023	GPT-3.5	48.1	~2e24
Eval 03/2023, 03/2023, 08/2023, 10/2023	GPT-4 0-shot (OpenAI, 2023) (Bubeck et al., 2023) (Muennighoff et al., 2023) (Shinn et al., 2023)	Mean = 79 67, 82, 87, 80 (reported by 4 papers)	~2e25[8]

With scaffolding

Rate of change

See HumanEval Data for more complete data, including corresponding CEG calculations.

All data

Date	Method (pass@1 unless specified otherwise)	Performance	Approx. compute increase from base 0-shot

[8] GPT-4 was left as 2e25 in PaLM-2 & GPT-4 in "Extrapolating GPT-N performance" so will keep that. If we wanted to account for fine-tuning there is a sigmoid with GPT-4 as 2e26 in the appendix.

02/2022	A bunch; GPT-Neo, Codex, AlphaCode at pass@1,10,100		
Release 02/2022, Eval 07/2022	code-davinci-002, AlphaCode-C Pass@1 (Chen et al, 2022)	55.1	~100x
Release 02/2022, Eval 07/2022	code-davinci-002, AlphaCode-C Pass@10 (Chen et al, 2022)	84.4	~100x
Release 03/2022, Eval 07/2022	code-davinci-002 (Chen et al, 2022)	47.0	1x
Release 03/2022, Eval 07/2022	code-davinci-002, Pass@10 (Chen et al, 2022)	74.9	10x
Release 03/2022, Eval 07/2022	code-davinci-002, Pass@100 (Chen et al, 2022)	92.1	100x
07/2022	code-davinci-002 CodeT (Chen et al, 2022)	65.8	~100x
07/2022	code-davinci-002 CodeT, Pass@10 (Chen et al, 2022)	86.6	~100x
Eval 03/2023	GPT-3.5 0-shot (OpenAI, 2023)	48.1	1x
Eval 03/2023, 03/2023, 08/2023, 10/2023	GPT-4 0-shot (OpenAI, 2023) (Bubeck et al., 2023) (Muennighoff et al., 2023) (Shinn et al., 2023)	Mean = 79 67, 82, 87, 80 (reported by 4 papers)	1x
Eval 05/2023	GPT-4 pass@64 (Zelikman, 2023)	82.3	64x
Eval 05/2023	GPT-4 pass@128 (Zelikman, 2023)	84.1	128x
Method 07/2022, Eval 05/2023	GPT-4 CodeT (Zelikman, 2023)	81.1	32x

Method 12/2022, GPT-4 eval 05/28/2023	Parsel + CodeT + GPT-4 (Zelikman, 2023)	85.1	32x
Method 03/2023, Eval 10/2023	GPT-4 Reflexion (Shinn et al., 2023)	91.0	~30x (very roughly (8 (calls per trial) * 2 (prompt size increase) + 3) * num_trials). num_trials doesn't seem to be specified, based on the code it seems to be 2
Method 10/2022, Eval 10/2023	GPT-3.5 ReACT (Yao et al., 2022) / (Zhou et al., 2023)	56.9	? Not sure. Will email LATS authors
Method 03/2023, Eval 10/2023	GPT-3.5 Reflexion (Shinn et al., 2023) / (Zhou et al., 2023)	68.1	~30x (very roughly (8 (calls per trial) * 2 (prompt size increase) + 3) * num_trials). num_trials doesn't seem to be specified, based on the code it seems to be 2
10/2023	GPT-4 LATS (Zhou et al., 2023)	94.4	Eli copy of WIP… estimates 80x ~168x (8*(4*5+1)). LATS authors confirmed ~168x songs right
10/2023	GPT-3.5 LATS (Zhou et al., 2023)	86.9	Eli copy of WIP… estimates 80x ~168x (8*(4*5+1)). LATS authors confirmed ~168x songs right

HotpotQA: Scaffolding + fine-tuning

Note that HotpotQA perhaps benefits an unusually large amount from scaffolding due to its multihop nature.

Date	Method (pass@1 unless specified otherwise)	Estimated CEG	Approx. compute increase from base 0-shot
	GPT-3.5 CoT (https://arxiv.org/pdf/2310.05915.pdf) Table 1	10^(5.6/14.8)=2.4x	
10/2022	GPT-3.5 ReACT (https://arxiv.org/pdf/2310.05915.pdf) Table 1	10^(9/15)=4x	?
03/2023	Reflexion (Zhou et al., 2023) Table 2	10^(29/15)=~86x, ~22x over ReACT	
10/2023	GPT-3.5 LATS (Zhou et al., 2023) Table 2	10^(39/15)=~400x ~5x over Reflexion	
10/2023	GPT-3.5 LATS + ReACT (Zhou et al., 2023) Table 2	10^(49/15)=~1848x Approx Would have expected *4*400=1600x with full complements. So ReACT somehow slightly more on top of LATS than it added by itself, if we are interpreting correctly	80x ~168x (8*(4*5+1))
10/2023	GPT-3.5 FireAct (https://arxiv.org/pdf/2310.05915.pdf)	On top of react: 10^(10/15)*4=19x total, ~5x relative to ReACT	Same at inference time as ReACT? Not sure otherwise

Fine-tuning

OctoPack (HumanEval)

OctoPack: Instruction Tuning Code Large Language Models

[Data] OctoPack: Instruction Tuning Code LLMs

We looked at data from the OctoPack paper and were able to extract a few CEG data points, as in the above sheet.

MetaMATH (MATH and GSM8K)

MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models

[Data] MetaMATH has data and CEG calcluations

Majority voting

Table 2:

LAMDA-173B = 3.55e23 compute
PALM-540B = 2.53e24 compute
7x compute
ASDiv -> 7^(9/25) = 2
SVAMP -> 7^(14.4/39.9)=2
GS8MK -> 7^(10.6/39.4)=1.7
ARC-e -> 7^(4/20)=1.48
ARC-c -> 7^(4.7/30.1)=1.36

Extracting stuff from Table 5 would need to be too nice but would need to look into scaling laws for those tasks.