Training method	Human feedback	Training incentive	Level of PH
Behavioral cloning	base task demonstrations	imitate human at base task	0
RLHP [CLB+17]	base task evaluations (i.e. critique demonstrations)	give outputs humans don't find flaws in	1
2-step debate [ICA18]	critique task evaluations (i.e. helpfulness demonstrations)	give outputs without critiques that humans judge as valid	2
2-step RRM [LKE+18]	critique task evaluations and assisted base task evaluations	give outputs humans don't find flaws in, with assistance from a critique model	2
Iterative refinement	base task edits (i.e. refinement demonstrations)	give outputs humans don't find improvements in	0

Table 5: A summary of training methods which seem potentially viable with recent ML methods. Based on the human feedback required, each corresponds to a different level of the polynomial hierarchy (PH)

This lets us easily, for example, run the critique task and evaluate critiqueability score at the same time. Due to this format, we also train on critiqueability and critique tasks in a single example. So the input to critique tasks always starts with "Yes" (the token corresponding to critiqueability).

Note that since we mask out all but the human-written demonstration during fine-tuning, this prevents us from sharing even more tasks in the same example. In this example, the base task would be done by a model rather than a human.

We briefly explored reordering in the synthetic RACE domain, and found it to make little difference. Though we believe the optimal format may be task-dependent, we leave investigation to future work.

For direct refinement tasks, we skip straight from showing critiqueability to refinement. For conditional refinement and helpfulness tasks, we omit critiqueability.

We also start and end model responses with an end-of-text token.

B Complexity theory analogy

In this section, we argue that critiques are a particularly natural form of assistance, from the perspective of proof systems.

B.1 Theory background

We can imagine humans as having some fixed computational ability while models can learn to have arbitrary computational ability, and ask what kinds of questions we can reliably train models to do. This motivates [ICA18] an analogy between scalable oversight and interactive proof systems from complexity theory [GMR89]: if humans are analogous to polynomial time Turing machines and models are computationally unbounded, what classes of decision problems can be solved by humans interacting with models?

For example, models trained with behavioral cloning can only solve problems in P, since humans must solve a problem to demonstrate its solution. Models trained against human evaluations can solve problems in NP/co-NP. [ICA18] proposes an AI safety via debate scheme which solves problems analogous to PSPACE [Sha92], with simpler n-step versions corresponding to the nth level of the polynomial hierarchy. In both the complexity theory setting and the scalable oversight analogy, a single prover can engage in deceptive strategies that can be prevented by cross-checking a second prover [BCOI20, BFL91].

B.2 Proof systems in practice

More recently, reinforcement learning from human preferences (RLHP) has become more common [CLB+17, BJN+22, MTM+22, SOW+20], demonstrating empirically a technique that lets us reach NP/co-NP. RLHP asks models to find solutions and humans merely to verify them.

A simple way to go past NP is to go one step up the polynomial hierarchy (PH), to Σ_2^P/Π_2^P. The definition of Σ_2^P corresponds precisely to 2-step recursive reward modeling (RRM): we give the verifier access to a model trained with RLHP (analogous to an NP/co-NP oracle). In general, n-step recursive reward modeling corresponds to the nth level of the polynomial hierarchy, just like n-step debate.

We can interpret the assistance needed to move up the polynomial hierarchy as inherently critique-like. The canonical Σ_2^P-complete problem essentially asks for existence of an answer such that there is no recognizable critique of the answer. Thus we ask humans only to recognize critiques, i.e. do the helpfulness task. For Σ_3^P, we would train an assistant model to critique critiques, and ask humans to evaluate critiques of critiques.

Note that it is not apparent whether iterative refinement lets us solve problems outside of P, since it asks humans to directly give answers. However they also serve as another comparison point algorithmically: iterative refinement may be very powerful if computation of the model is the limiting factor, rather than computation of the human. Overall, the proof systems view suggests the technique will become less useful as models become more powerful.

For a summary of possible approaches discussed, see Table 5.

C GDC gaps: discussion and extra results

We have defined the GDC gaps in Section 5. Here we discuss more intuitions and motivations for studying gaps, as well as subtleties in their measurement.

C.1 Informal intuition

How should we expect and want the difficulty of generating, discriminating, and critiquing answers to relate to each other?

First, for many tasks of interest we expect G $\leq$ D, meaning it is harder to write a good answer than to tell that an answer is wrong. This is analogous to the hypothesis that NP is larger than P. We call the gap between these abilities the ***GD gap*** (the generator-to-discriminator gap). The size of the gap may strongly depend on the nature of the problem: the problem of computing a hash might have no gap, but hash inversion might have a large gap. Note also that the GD gap is computational in nature: allowing rejection sampling against a discriminator can improve generator performance to close the gap.

Second, we expect C $\leq$ D. The ***CD gap*** roughly represents the gap from ability to articulate flaws (to humans), to the ability to recognize flaws. Again, the size of the gap may strongly depend on the nature of the problem – heuristic or statistical arguments might be often correct but hard to explicate or justify. For example, it may be easy for humans to discriminate cats versus dogs using intuitive (system 1) thinking, but not to explain why a certain cat photo is not a dog. However, even for more logical arguments, such as verifying a math proof, it's unclear whether the gap closes completely.

Finally the direction of the ***GC gap*** seems less clear; the considerations for the other two gaps are now competing. For example, G $>$ C: someone might know how to smile photogenically, but have trouble articulating why another smile looks inauthentic. But G $<$ C: it is easier to critique a poor story plot than to write a good story.

C.2 Relevance to model training and scalable oversight

Disclaimer: highly speculative

Our definitions assume access to ground truth for both the base task and critique task evaluations. But suppose for the sake of argument that we have some difficult task for which humans cannot determine ground truth, but that they can determine ground truth with critiques. (See Appendix B for motivation of this setting.)

In this setting, a negative GC gap seems undesirable in the following sense. Suppose the model could reach a certain level of performance for G if trained on ground truth. However, there is a GC gap, so that the model is able to generate some good outputs but is unable to always explain why some

worse output is worse. This means we may not have been able to elicit those good outputs to begin with, since we would not have provided a training signal to discourage the worse output. Optimizing against critiqueability will then yield model outputs that look good to humans but aren't actually good; in other words, the model will over-optimize [MG18] and produce misleading outputs. To avoid this, we should train the critique model to push the GC gap to be positive ($C \geq G$).

Closing the CD gap ($C = D$) is a more ambitious goal, and gets at an intuitively stronger property. The CD gap should be closed if a model is able to explain to humans why all poor outputs are poor. While weaker than "explaining to humans everything it knows," this would let humans straightforwardly notice misleading outputs and plausibly be a substantial step towards solving technical alignment, if it were possible. This stronger assurance may be relevant in practice: we could imagine a discriminator trained against flawed critiqueability labels favoring misleading generations, even if the generator cannot easily mislead itself. Rejection sampling against such a discriminator would lead to a misleading generator. Finally, a pragmatic reason to aim at closing the CD gap is that critiquing compares more naturally to discrimination (relative to generation).

Although we expect a non-negative CD in theory, in the absence of ground truth, we may be able to train better critique models than we can discriminators. Imagine, for example, if no human in the world knows about some undiscovered but common security vulnerability. Then training a discriminator on human labels for "this code has a security vulnerability" would not result in code avoiding that vulnerability. However, a helpfulness model might be able to predict that a human would be able to verify the vulnerability. And thus a critique model may point out the vulnerability. Of course, there is a risk that there are vulnerabilities that a model cannot explain to humans, if there is a CD gap, so it would of course be better to have correct labels for "this code has a security vulnerability" in the first place. But that may simply be impossible.

Thus, overall, we believe it is useful to push critique ability as far as we can. In practice, for difficult tasks, we don't expect to know what G and D are, since we cannot produce ground truth labels – we might push C without necessarily knowing where the limits are and without truly knowing if G or D are misleading. However, we can study these questions using problems with known ground truth such as synthetic tasks.

Relaxing ground truth

A different workaround is to relax this assumption of having ground truth. We could instead have more knowledgeable humans provide "ground truth" and less knowledgeable humans serve as labelers. We can use various methodologies to create the capability disparity, such as restricting time budget, hiring humans lacking expertise, etc. This is the setting introduced by [Cot21].

Then we have the analogy:

- D labels come from more knowledgeable humans trying to evaluate an answer
- C labels come from less knowledgeable humans evaluating an answer using assistance from an assistant model trained against labels from less knowledgeable humans

This framing gives an alternative way to motivate our gap measurement definitions.

Process versus outcomes

But what if we *do* have some ground truth and our discriminators look genuinely great, generalize well, etc? If $C < D$ then should we not just use a discriminator as training signal?

[SB22] argue that training to have "good process", i.e. a system with human-understandable steps, is better than directly training it against "good outcome". A different framing for closing the CD gap might be that we are trying to make a human-understandable process be competitive with an outcome-based training signal.

If we primarily train outcome-based systems, it will be tempting to use proxy ground truth when working on tasks with outcomes that are expensive or difficult to observe, such as long-horizon tasks. Furthermore, we will have very little insight into how these systems will generalize. So it could be important to begin improving process-based systems, even if they do not work as well as outcome-based systems today.