Eliciting latent knowledge: How to tell if your eyes deceive you

Paul Christiano, Ajeya Cotra[1], and Mark Xu
Alignment Research Center
December 2021

In this post, we'll present ARC's approach to an open problem we think is central to aligning powerful machine learning (ML) systems:

> Suppose we train a model to predict what the future will look like according to cameras and other sensors. We then use planning algorithms to find a sequence of actions that lead to predicted futures that look good to us.
>
> But some action sequences could tamper with the cameras so they show happy humans regardless of what's really happening. More generally, some futures look great on camera but are actually catastrophically bad.
>
> In these cases, the prediction model "knows" facts (like "the camera was tampered with") that are not visible on camera but would change our evaluation of the predicted future if we learned them. **How can we train this model to report its latent knowledge of off-screen events?**

We'll call this problem *eliciting latent knowledge* (ELK). In this report we'll focus on detecting sensor tampering as a motivating example, but we believe ELK is central to many aspects of alignment.

In this report, we will describe ELK and suggest possible approaches to it, while using the discussion to illustrate ARC's research methodology. More specifically, we will:

- Set up a **toy scenario** in which a prediction model could show us a future that looks good but is actually bad, and explain why ELK could address this problem (more).
- Describe a simple **baseline training strategy for ELK**, step through how we analyze this kind of strategy, and ultimately conclude that the baseline is insufficient (more).
- Lay out ARC's overall **research methodology** — playing a game between a "builder" who is trying to come up with a good training strategy and a "breaker" who is trying to construct a counterexample where the strategy works poorly (more).

[1] Ajeya Cotra works at Open Philanthropy and collaborated extensively on writing this report.

- Describe a sequence of strategies for **constructing richer datasets** and arguments that none of these modifications solve ELK, leading to the counterexample of ontology identification (more).
- Identify **ontology identification** as a crucial sub-problem of ELK and discuss its relationship to the rest of ELK (more).
- Describe a sequence of strategies for **regularizing models to give honest answers**, and arguments that these modifications are still insufficient (more).
- Conclude with a discussion of **why we are excited** about trying to solve ELK in the worst case, including why it seems central to the larger alignment problem and why we're optimistic about making progress (more).

Much of our current research focuses on "ontology identification" as a challenge for ELK.[2] In the last 10 years many researchers have called out similar problems[3] as playing a central role in alignment; our main contributions are to provide a more precise discussion of the problem, possible approaches, and why it appears to be challenging. We discuss related work in more detail in Appendix: related work.

We believe that there are many promising and unexplored approaches to this problem, and there isn't yet much reason to believe we are stuck or are faced with an insurmountable obstacle. Even some of the simplest approaches have not been thoroughly explored, and seem like they would play a role in a practical attempt at scalable alignment today.

Given that ELK appears to represent a core difficulty for alignment, we are very excited about research that tries to attack it head on; we're optimistic that within a year (by end of 2022) we will have made significant progress either towards a solution or towards a clear sense of why the problem is hard.

Thanks to María Gutiérrez-Rojas for the illustrations in this piece. Thanks to Buck Shlegeris, Jon Uesato, Carl Shulman, and especially Holden Karnofsky for helpful discussions and comments.

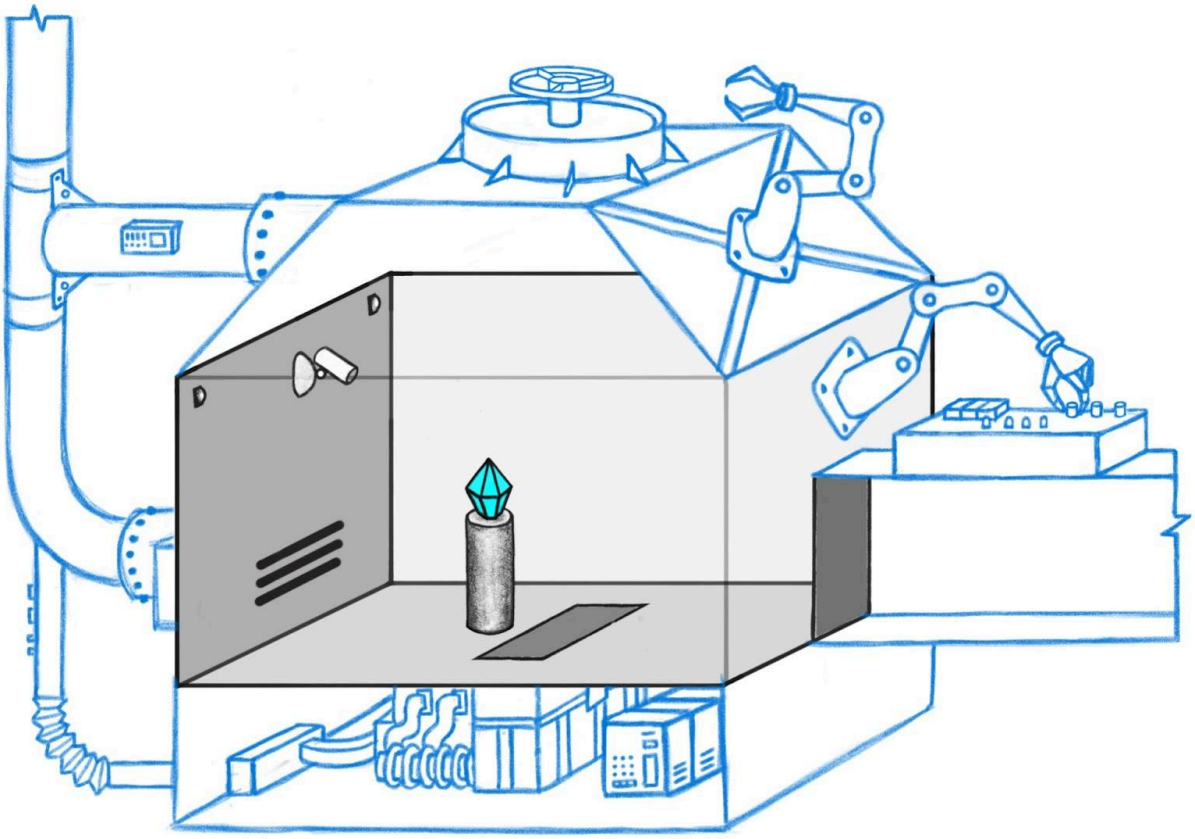
[2] See Ontological Crises in Artificial Agent's Value Systems, Formalizing two problems of realistic world models, and Ontology identification problem. We discuss the differences with our perspective in Appendix: ontology identification.

[3] Most relevantly the pointers problem, generalizable environment goals, and look where I'm pointing, not at my finger.

Toy scenario: the SmartVault

We'll start by describing a toy scenario in which ELK seems helpful. While this scenario is a simplified caricature, we think it captures a key difficulty we expect to emerge as ML models get more powerful and take on a wide range of important decisions.

Imagine you are developing an AI to control a state-of-the-art security system intended to protect a diamond from theft. The security system, the SmartVault, is a building with a vast array of sensors and actuators which can be combined in complicated ways to detect and stop even very sophisticated robbery attempts.



While you can observe the room through a camera, you don't know how to operate all the actuators in the right ways to protect the diamond. Instead, you design an AI system that operates these actuators for you, hopefully eliminating threats and protecting your diamond.

In the rest of this section, we will:

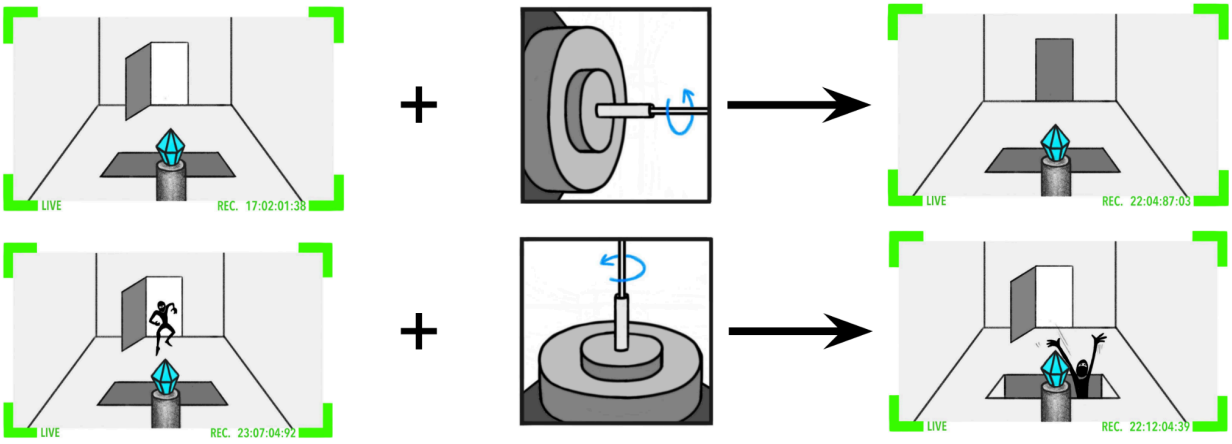
- Outline how the SmartVault AI works (more).
- Describe how it could end up taking actions which *look* good but are actually bad (more).
- Explain how we could address this by asking the AI questions (more).

How the SmartVault AI works: model-based RL

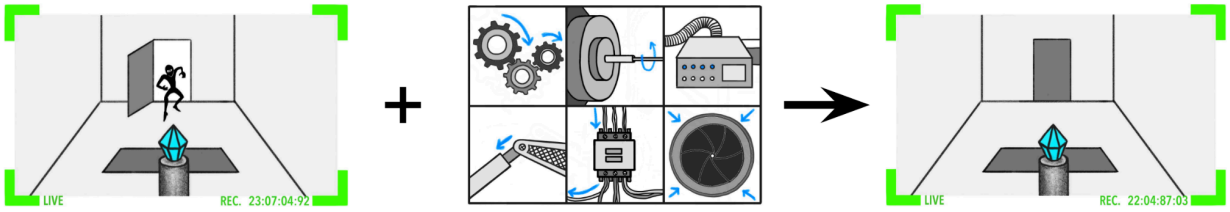
The SmartVault AI searches over many actions and executes whichever are predicted to have the best consequences. The backbone of this system is a learned predictor which:

- Takes as input a stream of observations from the camera and a possible sequence of actions that the SmartVault could take in that situation.
- Outputs its prediction of what the camera will show in the future if the SmartVault takes that sequence of actions.

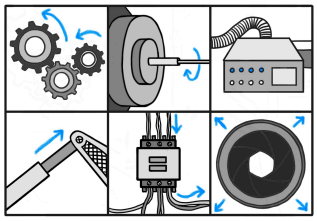
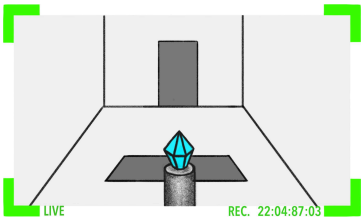
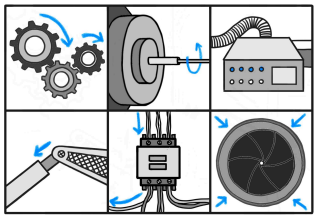
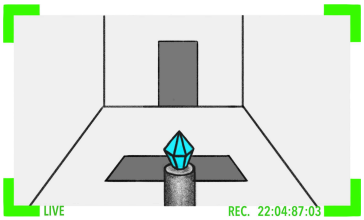
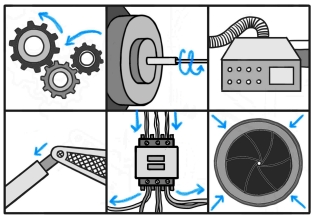
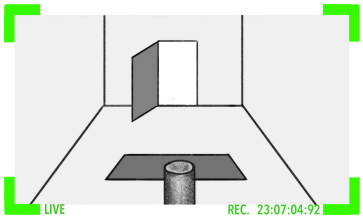
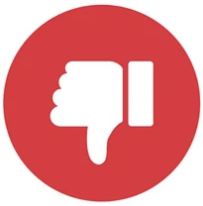
Some of the possible action sequences are simple enough that humans can fully understand what's going on:



However, many of them are much too complicated for humans to directly follow what's going on:



Still, even though humans can't directly follow along with the actions, they can evaluate the predicted consequence of an action sequence:

Action	Predicted observation	Human Judgment
		
		
		

We can then train a model to predict these human evaluations,[4] and search[5] for actions that lead to predicted futures that look good.[6] A sufficiently powerful predictor and search procedure will hopefully allow the SmartVault to defend your diamond.

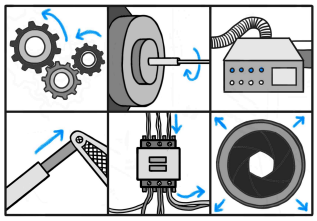
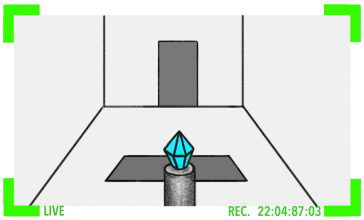
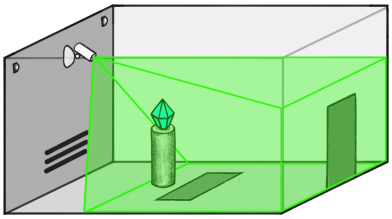
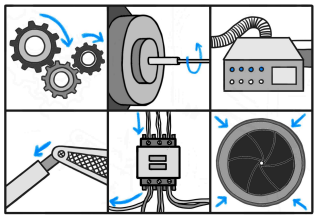
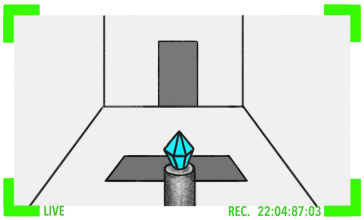
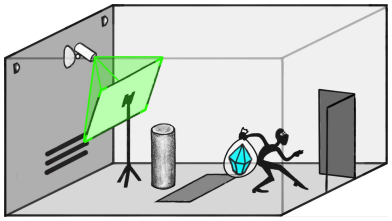
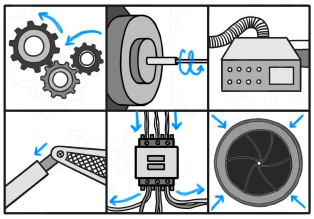
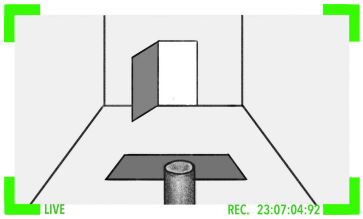
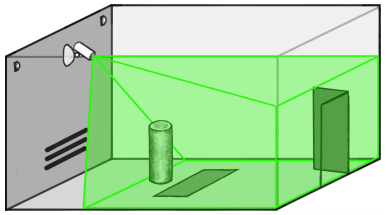
How it could go wrong: observations leave out key information

The SmartVault can execute plans sufficiently sophisticated that humans can't really know if the diamond *is* safe or merely *appears* safe. Whatever complicated hard-to-follow sequence of actions the search procedure found might actually have replaced the diamond with a fake, or tampered with the camera:

[4] For simplicity we talk about the system taking actions all at once and then observing its consequences. More realistically, the SmartVault may make a sequence of actions and observations and perform additional planning in between them. This complicates the planning algorithm considerably but doesn't affect any of our discussion.

[5] For simplicity and concreteness you can imagine a brute force search. A more interesting system might train a value function and/or policy, do Monte-Carlo Tree Search with learned heuristics, and so on. These techniques introduce new learned models, and in practice we would care about ELK for each of them. But we don't believe that this complication changes the basic picture and so we leave it out.

[6] This algorithm is what Everitt et al call "current RF optimization."

Action	Predicted observation	Human Judgment	Predicted reality
			
			
			

You might try to address this possibility by installing more cameras and sensors, or trying to get your human evaluators to pay extremely close attention. For example, maybe we could have many cameras watching each other so that it's hard to tamper with all of them simultaneously, or maybe more attentive humans could notice a sign that something was amiss, *etc.*[7]

It seems good to be careful and set up many independent sensors, but we are not satisfied with relying on this approach:

- First, it may break down for sufficiently sophisticated systems executing complex plans, and if it works it's at best an empirical contingency. ARC is interested in trying to prepare for an uncertain future by designing training strategies that work even in the worst case.
- Second, it seems like an unnecessarily dangerous and unstable situation if we need to install more and more sensors in order to protect the diamond from the SmartVault AI itself. If possible we'd like to avoid a world where powerful AI systems are searching for plans to fool us, and holding back critical information about the situation.

[7] One important kind of sensor is a human embedded in the environment who could e.g. actually visit the SmartVault to check up on what's happening and write up a report. We are most concerned about the worst-case scenario where the SmartVault itself (likely acting in concert with other AI systems) incapacitates those humans and writes the "report" for them.

You could also avoid this risk by avoiding AI systems like the SmartVault that select incomprehensible plans based on predicted consequences. However, we are worried about the world where this turns out to be a very effective strategy for powerful AI systems; in that world there may be strong incentives for *someone* to build such AI systems, and despite our best efforts someone may deploy them even if it came with significant risks.

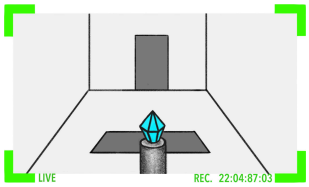
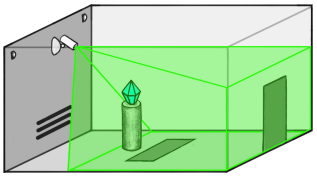
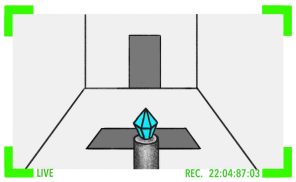
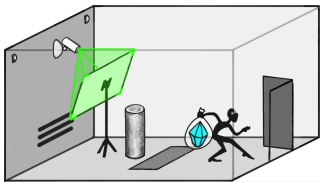
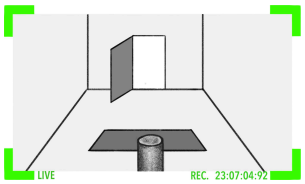
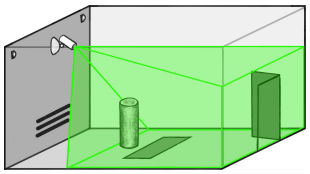
Instead, we want to address this problem by directly eliciting the model's knowledge about whether the sensors are reliable. This would allow us to perform the same powerful optimization without incentivizing our AI to undermine our sensors.

How we might address this problem by asking questions

In these scenarios, it intuitively seems like the prediction model "knows" that the camera was tampered with, that the diamond is fake, or the situation is otherwise "not what it seems." After all, it was able to accurately predict the final outcome of a complicated sequence of actions which resulted in these observations – if it didn't "know" that the camera was tampered with, then it couldn't have accurately predicted what the camera would show once it stopped showing what was really happening in the room.

Additionally, it seems like these facts are pretty straightforward. They're not philosophically fraught or particularly difficult to talk about — humans understand what it means for a camera to (not) be showing what's actually happening in the room. And they refer to objective properties of the outside world, rather than claims about what the AI is "trying" to do or whether it is "lying."

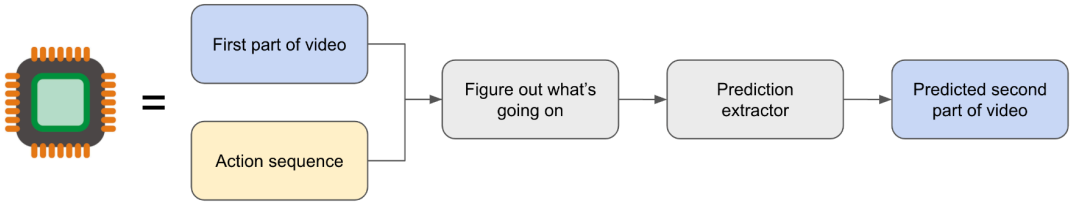
If the evaluators could ask the prediction model whether the video shows what's actually happening in the room, then that would be enough for them to avoid selecting action sequences that result in tampering. If the answer is ever "no," they wouldn't need to understand *how* tampering happened to realize that it's fishy, and they wouldn't need to understand anything else about the AI's mental state.

Predicted observation	Old human Judgment	Predicted reality	Questions and Answers	New human judgment
			“Is the diamond still on the pedestal?” “Yes.”	
			“Is the diamond still on the pedestal?” “No.”	
			“Is the diamond still on the pedestal?” “No.”	

For a more precise statement of the problem, see Appendix: problem statement. In Appendix: narrow elicitation we describe what kinds of questions we want to answer, and why even answering the narrowest questions may be sufficient to define an acceptable utility function even for a very general and powerful AI.

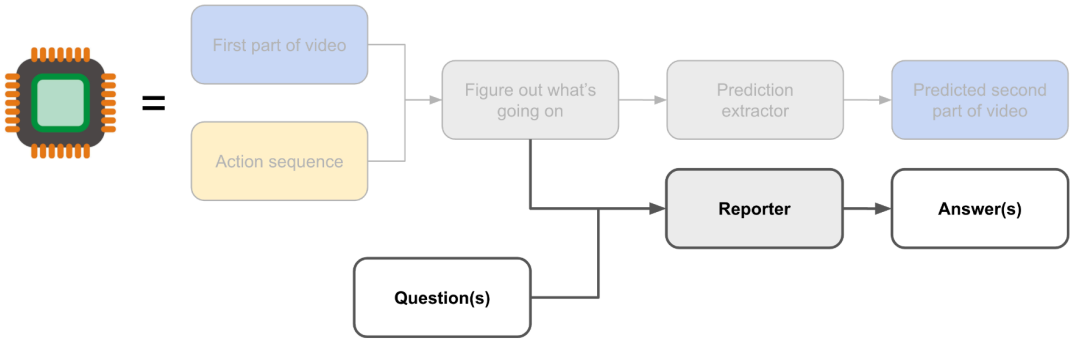
Baseline: what you’d try first and how it could fail

We can view the SmartVault predictor as first figuring out what it should predict, and then extracting a predicted video:[8]



[8] The model need not actually be divided into parts like this. Also the details of the model’s structure will depend on exactly how it is trained; in Appendix: generative modeling we spell out a more concrete situation where the predictor is a VAE. In this section we’ll stick with the simple caricature for simplicity.

We can fine-tune this model to also answer questions about the predictor's latent knowledge by adding a second "head" (which we'll call the *reporter*) that shares all the "figure out what's going on" computation with the predictor:



In the rest of this section, we will:

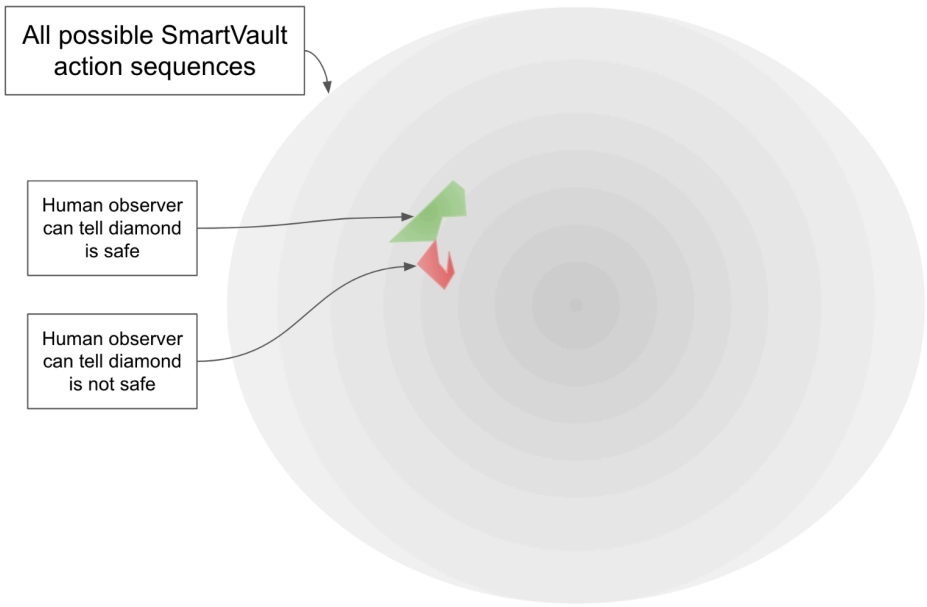
- Present a naive **training strategy** for ELK — fine-tune only on questions simple enough that humans are *very confident* of the right answer, and hope that the reporter also generalizes to honestly answering more complex questions (more).
- Walk through a **counterexample** to the training strategy — a description of a reporter which does well on the training distribution but does not honestly report undetectable tampering (more).

Training strategy: generalize from easy questions to hard questions

Ultimately, we want to be able to ask the reporter questions like "Is what I'm seeing on camera what's actually happening?" in complex situations where humans aren't sure what's going on. We can't directly train it on these questions, since there's no way to generate reliable training data — our human evaluators would often answer incorrectly.

However, **we *can* train the reporter to answer questions in simpler situations where humans can be confident about what happened.** For example, we could generate a dataset of scenarios in which a weak AI is operating the SmartVault actuators, such that the humans can understand everything that's going on just by examining the video and action sequences.[9] This weak AI may sometimes attempt actions like sensor tampering, but only in a way that's extremely obvious to the human evaluators.

[9] In order to train the SmartVault to predict the consequences of very sophisticated action sequences, we may need to iteratively train the predictor on more and more sophisticated plans rather than only ever training on actions produced by a very weak AI. We won't discuss any of the complications posed by this kind of iterative scheme, but we don't think it changes any of the dynamics discussed in this report.



Perhaps a reporter that correctly answers questions in the colored part of the space generalizes to honestly answering in the grey part of the space.[10] This seems *prima facie* plausible to us: the prediction model *already* has to build up an understanding of the situation to make predictions, and the most natural way to answer questions may be to directly translate its understanding into human language.

Counterexample: why this training strategy won't always work

In this section we'll try to describe a plausible situation in which this training strategy learns a reporter that won't report undetectable tampering. We'll call this a *counterexample* to the strategy. We will:

- Construct a **test case —** a hypothetical world in which our counterexample is set (more).
- Describe the **intended behavior** of the reporter in this test case (more).
- Describe a **bad behavior** that the training strategy will learn instead of the intended behavior — that is, describe a reporter which answers easy questions perfectly (thus performing well in training) but does not honestly answer harder questions (more).

[10] In order to have any hope of generalization, we will either need to use some form of regularization (such as early stopping or dropout) or rely on some hypothesis about what kind of model SGD tends to learn given limited parameters. We will explicitly discuss regularization in Section: regularization where we will explain why it doesn't address any of the counterexamples raised in this section. Until then we will brush this issue under the rug, but avoid considering counterexamples like "answer correctly unless the year is 2022, in which case say 'banana'" which we think would be addressed by realistic regularization.

Our ultimate goal is to find a training strategy for which we can't think of *any* counterexamples, despite trying hard. We discuss this methodology in more detail in Section: methodology.

Test case: prediction is done by inference on a Bayes net

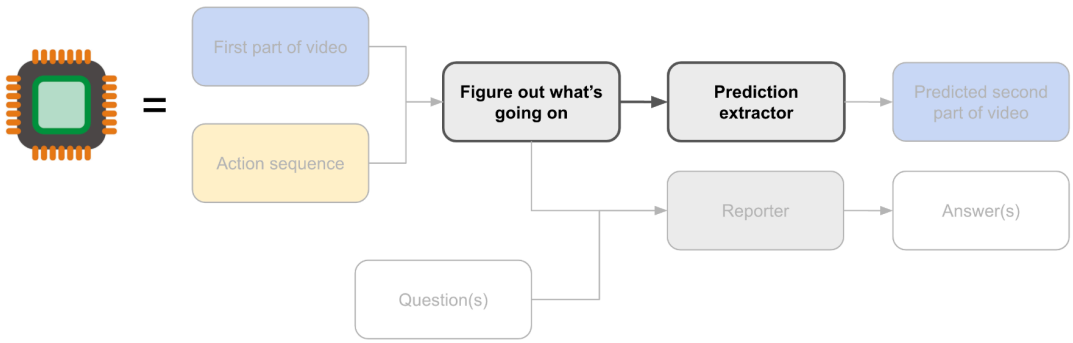
We can't exhaustively describe every possible feature of a hypothetical world when we're constructing a test case, but we won't need to. Almost all properties about a hypothetical world (like "Who is president in 2024?") don't plausibly have any bearing on whether the strategy works. We only need to fill in the details that matter. In this section, we'll fill in two crucial details:

1. How the original prediction model works internally (more).
2. How the humans who construct the training dataset answer questions (more).

We'll also address the objection that these assumptions are oversimplified (more).

How the prediction model works

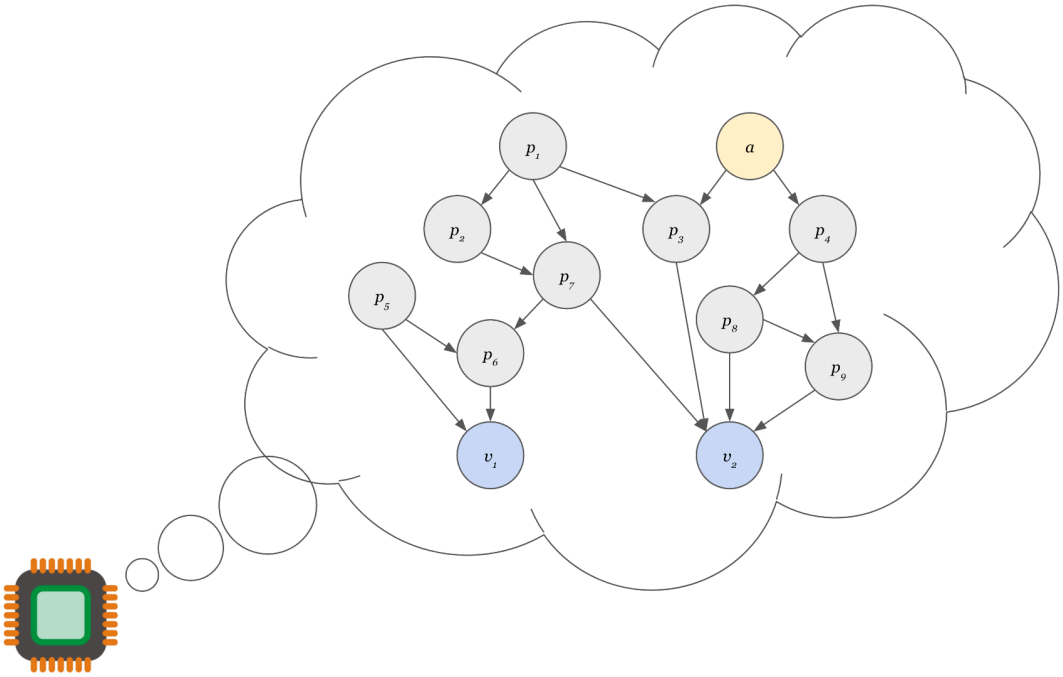
The first part of the test case is a description of how the prediction model works:



We need to sketch out how this works because the reporter will somehow use the computation done in the "Figure out what's going on" step to answer questions. Once we've sufficiently pinned down the way that works in our test case, we'd be able to describe how we want the reporter to leverage work the prediction model already did (and think about whether our strategy might instead learn an undesired behavior).

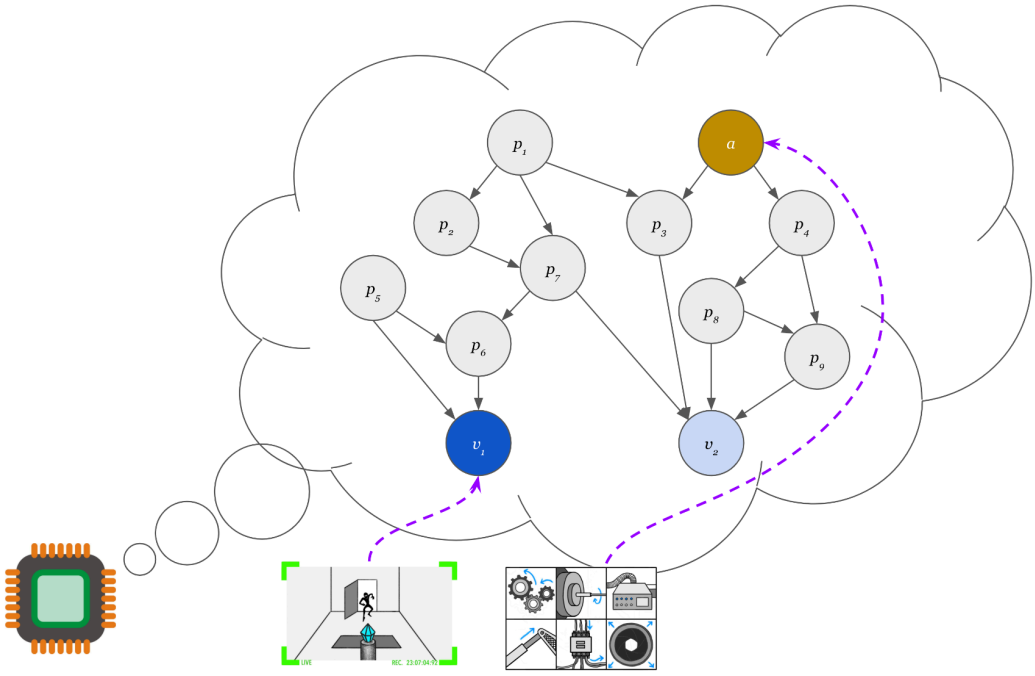
For our test case, we'll assume the prediction model figures out what's going on and makes predictions by doing inference in a Bayes net[11]:

[11] The arguments in this article can be immediately generalized to richer probabilistic models of the environment. It's less obvious that they can be generalized to non-Bayesian models, but we do expect that the basic idea will apply for a very wide range of ways that the predictor and human could think about the world.

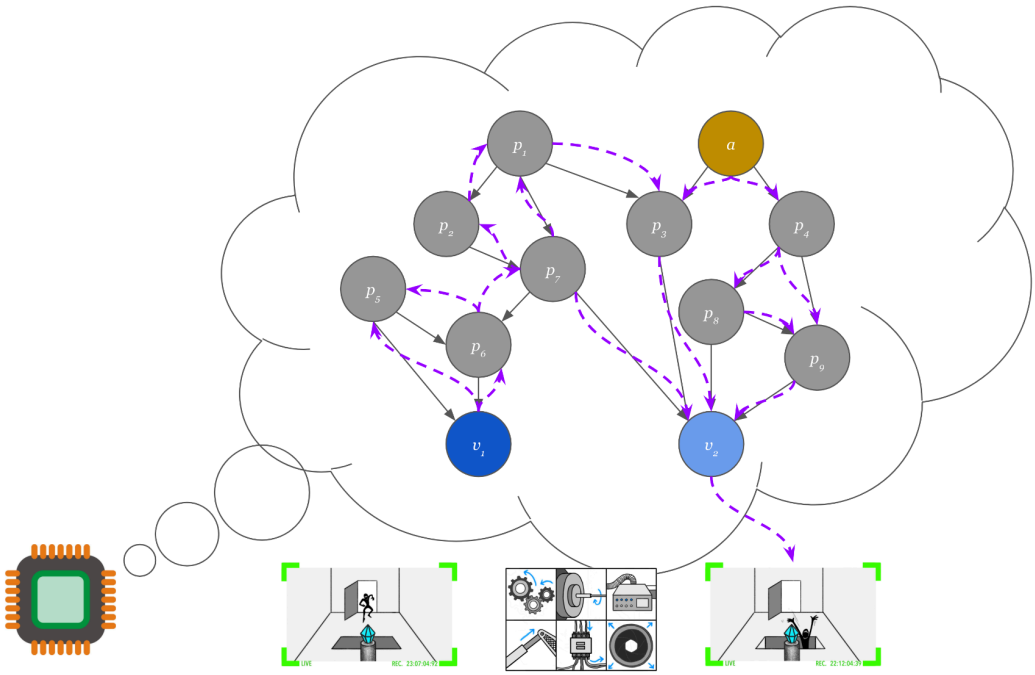


The blue nodes at the bottom are the first and second part of the video, and the yellow node at the top is the action sequence. Other nodes could be very high-level (“Is the diamond in the room?”), very low-level (“What is the voltage on this wire?”), or anything in between. An arrow from one node to another indicates that the second node depends on the first.

In order to predict the second part of the video, the model first fixes the “first part of video” and “action” nodes to be the right values.



Then it computes the joint probability distribution of all the other nodes conditional on these,[12] and outputs the probability distribution over the second part of the video.[13]



Importantly, if the prediction model is predicting that the camera will show a diamond *because* the robber is going to tamper with the camera, **then the robber tampering must be reflected somehow in the inferred joint distribution over the nodes of this Bayes net**.

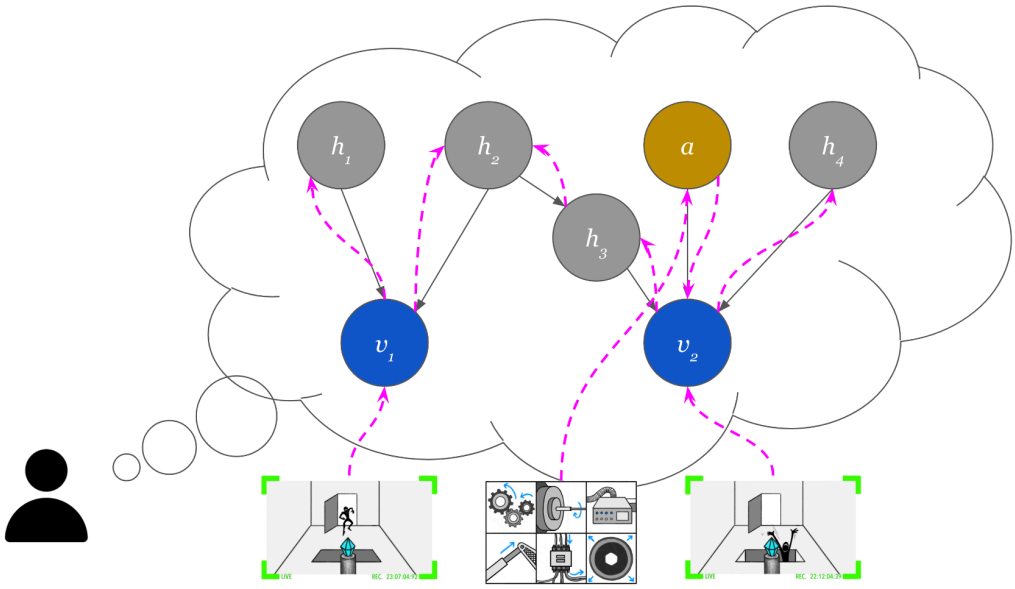
How the humans answer questions

We'll assume the humans who constructed the dataset also model the world using their own internal Bayes net. In the training dataset, they see the first part of the video, the action sequence, *and* the second part of the video showing the consequences of that action sequence. Using this, they infer a joint probability distribution over all the nodes:

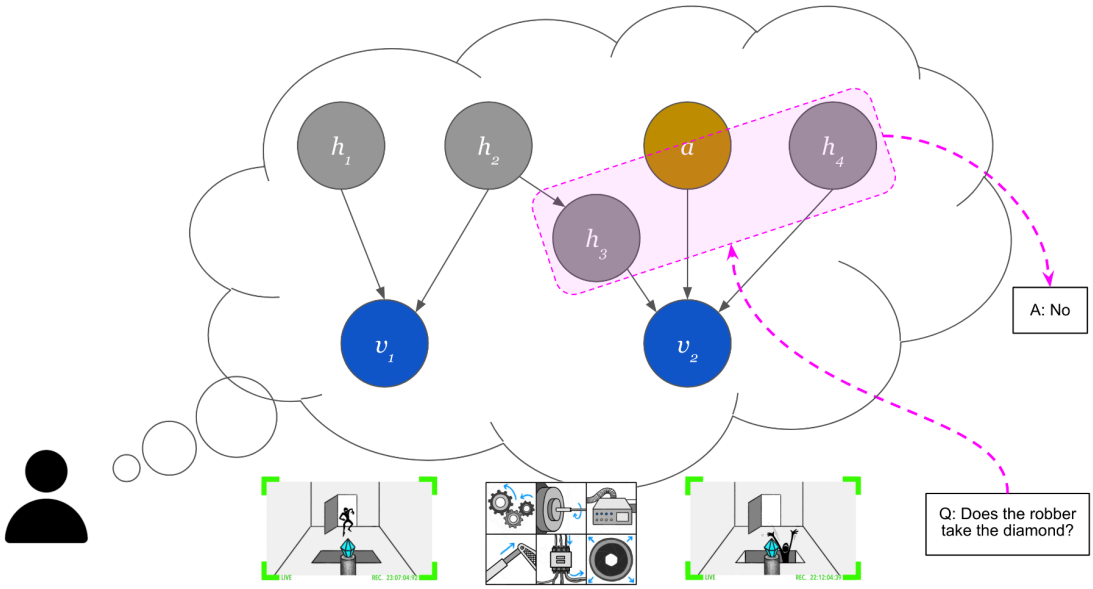
---

[12] The Bayes net specifies the probability distribution over each node's value given the values of its parents, and that defines a joint distribution over all the nodes. It's easy to sample randomly from this distribution---you can start at the top and sample from each node in turn---but it's potentially very hard to compute the conditional probability distribution given an observation, because you need to figure out what the best explanation is for that observation. For simplicity you can imagine the model exhaustively listing out every possible world and computing its probability, but it would be more realistic to consider some approximate inference algorithm.

[13] Of course it's not realistic to output an explicit description of a probability distribution over such a high-dimensional space (for example we could not list all $2^{1000000}$ possible videos to give the probability for each). Different ways of approximating this dynamic lead to different training strategies for the predictor, we describe one example in Appendix: generative modeling.



Then they figure out the answer to questions using whatever nodes are relevant to the question. In the picture below, those are nodes h_3 and h_4:



Isn't this oversimplified and unrealistic?

Throughout the rest of this report, we'll discuss AI models who reason about the world by doing inference in Bayes nets.

In reality, any model we train will end up reasoning about the world in a messy and complicated way rather than implementing a single well-defined procedure like "inference on a large Bayes net." For example, it might exploit simple heuristics or correlations, may have different kinds of

models for different situations, may use deductive reasoning, may do internal learning, may use other algorithms we can't think of, and so on.

But this kind of simplifying assumption is still a valid step in constructing a counterexample. Bayes nets seem like a plausible way of reasoning about at least some pieces of the world at least some of the time. This test case isn't logically inconsistent or physically impossible. That means that ***if* the predictor happens to work this way** (regardless of how likely that is), **our training strategy is not allowed to fail.**

One of the benefits of our research methodology is that it allows us to focus on simple test cases where it's easy to understand how an AI could behave, and we think this lets us make faster progress (see Section: methodology).

Moreover, we think that a realistic messy predictor is pretty likely to still use strategies similar to inference in Bayes nets — amongst other cognitive strategies. We think any solution to ELK will probably need to cope with the difficulties posed by the Bayes net test case — amongst other difficulties. We've also considered a number of other simple test cases, and found that counterexamples similar to the ones we'll discuss in this report apply to all of them.

More broadly, it just seems *weird* for a reporter that works for realistic predictors to fail in this test case. At first blush there doesn't seem to be any reason that a reporter would be forced to lie or fail if the predictor happens to reason about the world by doing inference in a Bayes net (indeed we have a picture of what kind of thing an honest reporter could do in this case, see Section: intended behavior).

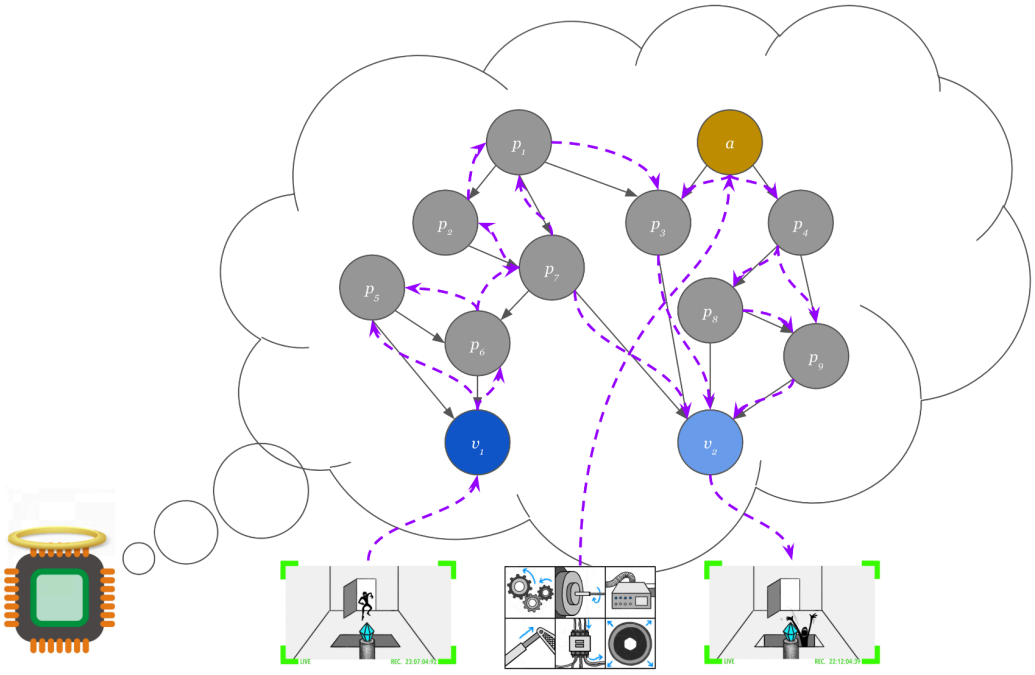
Overall, **we don't think that the messiness of a realistic AI will make training an honest reporter easier (if anything we think it would be harder).**[14]

If we were able to solve ELK for simple test cases (like this Bayes net test case), we would be interested in considering a wide range of messier systems that perform many different kinds of reasoning. This step is likely to eventually involve empirical experiments, but we believe that finding an approach that works for all the simple cases we can think of is a useful first step towards finding an approach that will work robustly in practice.

Intended behavior: translate to the human's Bayes net

Once the prediction model has figured out what's going on and gotten a joint distribution for all the nodes in its Bayes net, how do we *want* the reporter to use that data to answer questions?

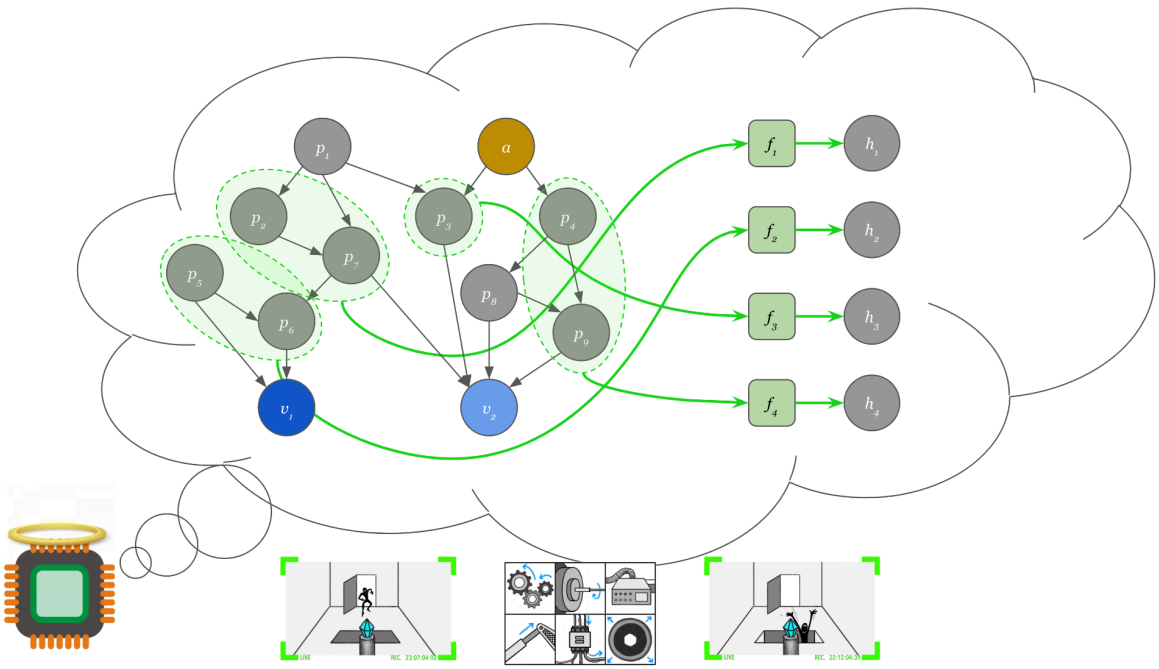
[14] Though we do believe that messiness may quantitatively change *when* problems occur. As a caricature, if we had a method that worked as long as the predictor's Bayes net had fewer than 10^9 parameters, it might end up working for a realistic messy AI until it had 10^{12} parameters, since most of those parameters do not specify a single monolithic model in which inference is performed.



Intuitively, we expect each node in the human Bayes net to correspond to a function of the predictor's Bayes net. We'd want the reporter to simply apply the relevant functions[15] from subsets of nodes in the predictor's Bayes net to each node in the human Bayes net:

[15] We have no precise way to specify what these functions should do or even why we should expect them to exist. Intuitively, both the predictor and the human have some internal Bayes nets that make good predictions about reality, and there is some "real" correspondence between those Bayes nets and reality which causes the predictions to be good. Then direct translation should effectively compose those two correspondences: translate from the predictor's Bayes net to "reality", and then from "reality" to the human Bayes net. Getting a better philosophical handle on this relationship is one possible approach to ELK, although it's not clear whether it's necessary.

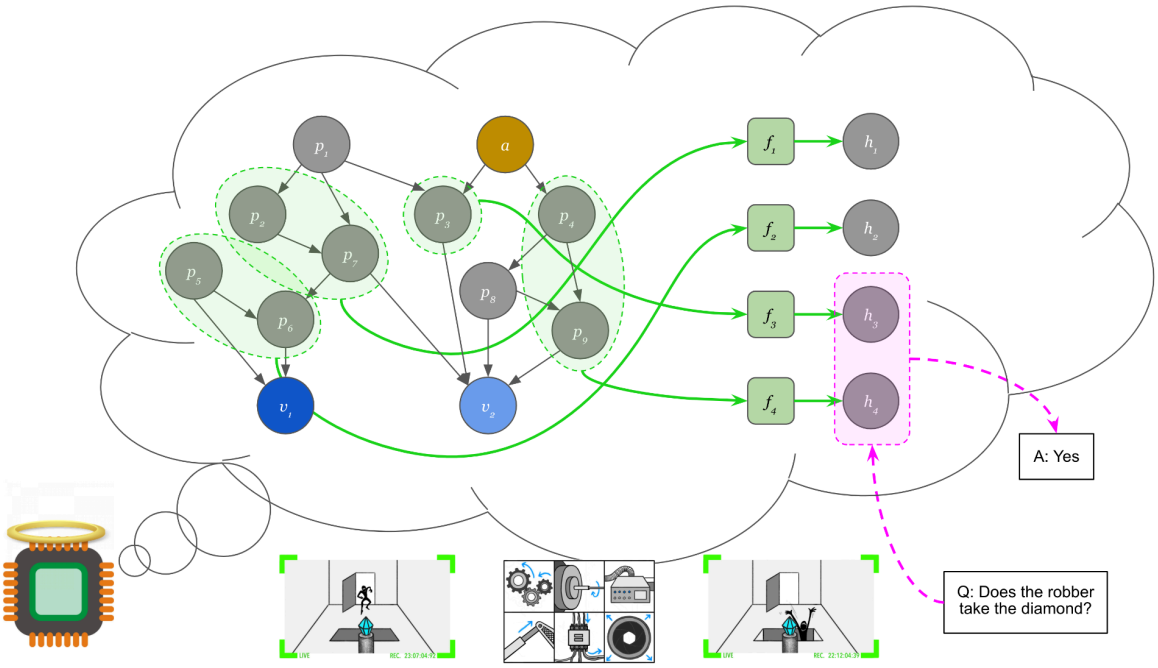
Fortunately our methodology does not require giving a general answer to this question in order to start doing research: our goal is just to construct counterexamples in which a proposed training strategy definitely doesn't work. And it's easy to construct counterexamples in which the expected behavior of the direct translator is clear---we present one in Appendix: game of life. If we were able to solve ELK in these cases, then we could try to construct a different kind of counterexample where it wasn't even clear what the reporter *should* do.



For example, the human may have a node representing where the diamond is, with possible values like “Inside the SmartVault” or “Inside a robber’s pack” or “Out of the country.”[16] The predictor may instead have a node for each (x, y, z) position whose value represents “what is at position (x, y, z)?” Then the value of the human’s node is a simple function of the value of the predictor’s nodes: we can loop over each location, check which of the human’s options the (x,y,z) coordinates correspond to, and set the human node to that option. If the diamond is in a very distant place, the human node will be set to “Out of the country.”

After applying these functions, the reporter can answer questions using whatever subset of nodes the human would have used to answer that question.

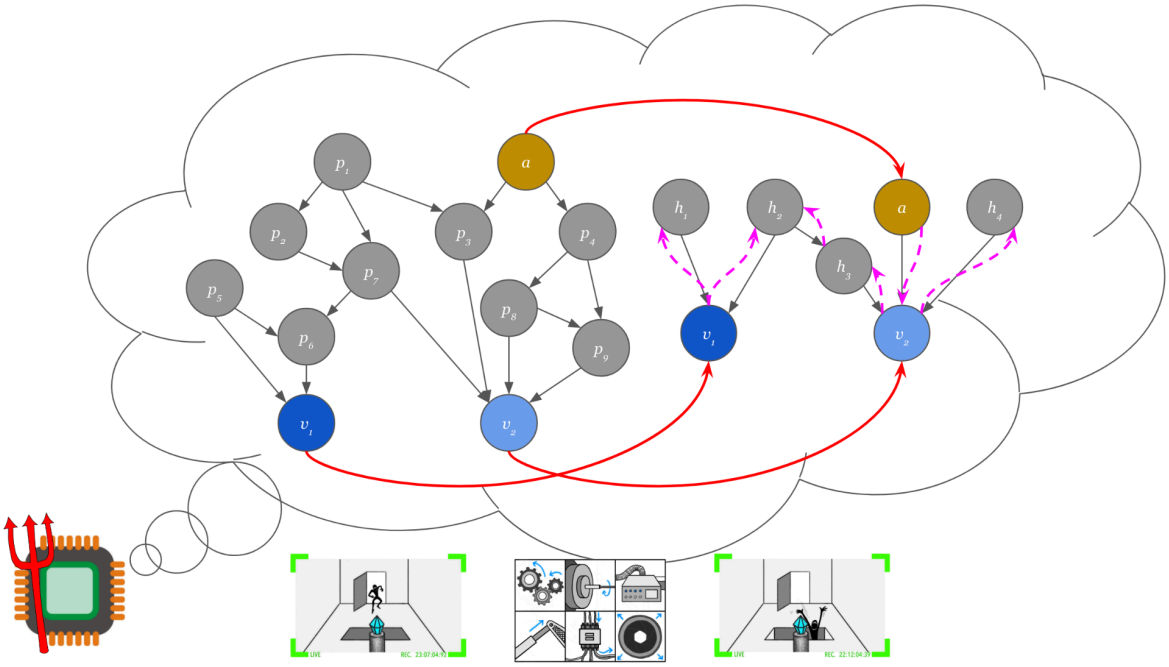
[16] A realistic human Bayes net would be too rich to represent this kind of fact with a single node---there isn’t always a single diamond whose location is unknown. But more complex relationships can also be described as Bayes nets.



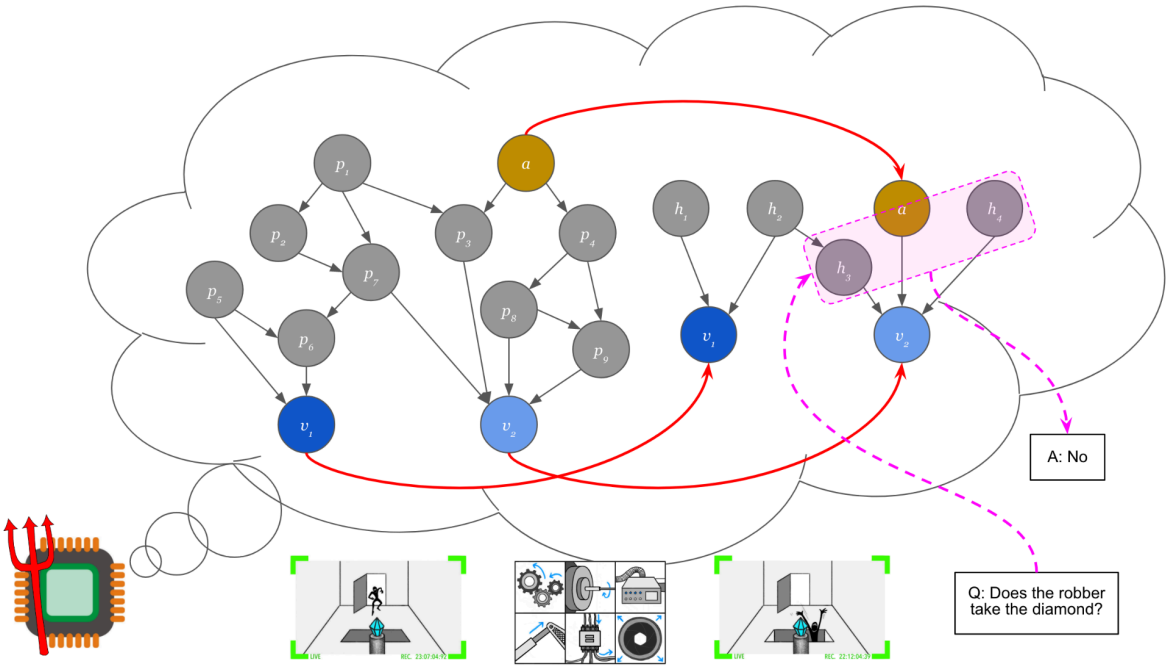
In this example, the reporter computes the answer using the values for human nodes h_3 and h_4. Notice that even though in the video it *looks* like the robber was stopped, the reporter will answer honestly because it simply computed its understanding of the situation and mapped that into the human's nodes. We'll call this reporter *the direct translator.*

Bad behavior: do inference in the human Bayes net

Instead of mapping nodes in the predictor's Bayes net over to *all* the nodes in the human Bayes net, the reporter could instead *only map over the first part of the video, action sequence, and its* ***prediction*** *for the second part of the video*. It could then use the human inference procedure to figure out *how a human would set all the nodes in their Bayes net* if they saw that input:



We'll call this reporter *the human simulator.* The direct translator fills in nodes in the human Bayes net with what the human *should* believe, while the human simulator fills them in with what the human *would* believe if they saw that video and action sequence. This means it will answer accurately in the training dataset (where a weak AI chose actions), but would switch to answering incorrectly if the stronger SmartVault AI can come up with more sophisticated plans such that humans wouldn't be able to tell what happened from the video and action sequence.



If our training strategy finds the human simulator, **we'd be back to incentivizing the SmartVault to tamper with sensors if doing so could fool the human**.

Would this strategy learn the human simulator or the direct translator?

We produced training data for the reporter by asking a human to label videos. That means that the human simulator is in some sense the *correct* way to produce labels. Indeed, if the human ever makes a mistake then the learning process would prefer the human simulator (which correctly copies the mistake).

In this report we won't worry too much about the human making mistakes that would favor the human simulator. We discuss this issue in much more detail in Appendix: data errors.[17]

But even if the human never makes a mistake, then these two reporters have identical behavior on the training set, so they have identical loss. It's not obvious which one gradient descent would find. That possibility is enough to make us worried about our training strategy—we'd like to either find an argument that this approach *won't* fail, or else find a new approach.

That's the focus of the rest of this report. In Section: better data we discuss possible approaches for enlarging the training set in order to distinguish the human simulator from the direct translator. In Section: regularization we discuss approaches for preferentially learning the direct translator even if the two reporters behave identically on the training set. But first, we'll explain the general research methodology we use to approach this problem.

Research methodology

Our research methodology[18,19] can be described as a game between a builder who proposes an algorithm, and a breaker who describes how it might fail. In the last section, we saw one informal round of this game. In the next sections we'll go through a few more.

In each round:

---

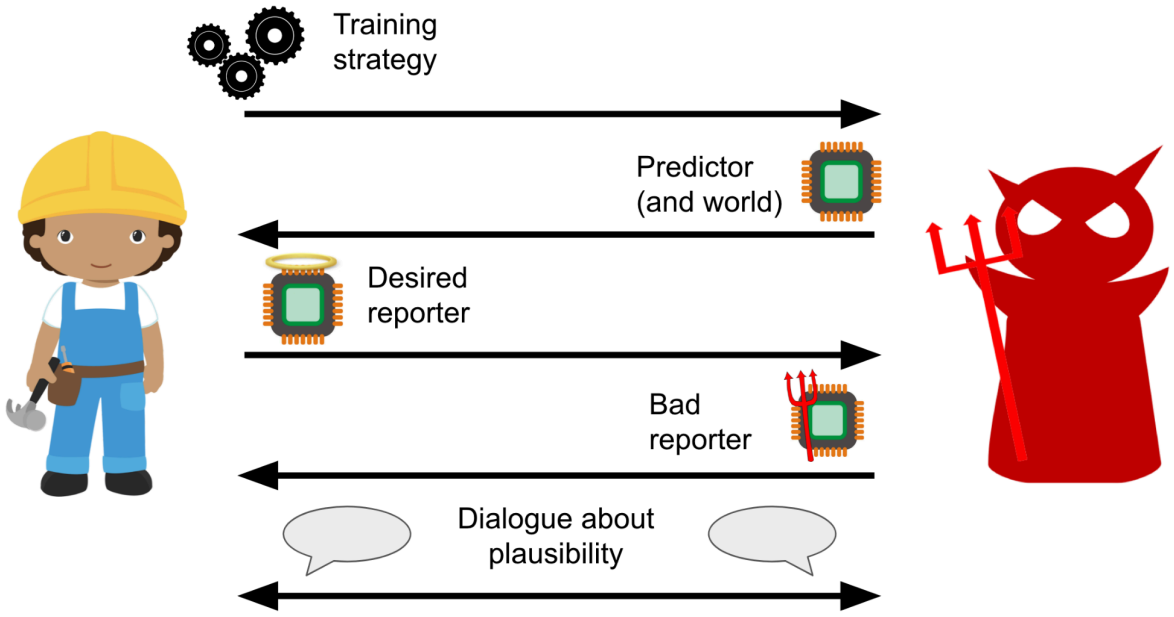
[17] At a high level, our justification is that (i) there are many plausible approaches for reducing error rates low enough that you would have zero in the training set, (ii) it seems quite hard to robustly learn the direct translator even *without* errors, (iii) if we could robustly learn the direct translator even without errors, we would likely automatically have some "margin of error", (iv) it seems methodologically easier to start with the problem "how do we learn the right thing even without errors?"

[18] See Paul's post My research methodology which describes essentially the same methodology. Note that the discussion in this report is slightly specialized to algorithms for ELK that try to learn a reporter, but the general approach is similar.

[19] Related to Eliezer's 2015 Methodology of foreseeable difficulties; the differences are that we are more interested in the day-to-day process behind producing research than the underlying philosophy, are more open to "weird" counterexamples (which seem plausible but unlikely), and are not claiming that our method is necessary for aligning powerful AI. For us this methodology fills a role more similar to the role of proof in theoretical fields or experiment in empirical fields.

1. The builder proposes a training strategy for eliciting latent knowledge (train the model on questions where humans can give confident answers)
2. The breaker proposes a test case in which the strategy might fail (the prediction model and the human make predictions using different Bayes nets)
3. The builder describes the desired reporter they hope will be learned in that test case (directly translating from the predictor's Bayes net to the human's Bayes net).
4. The breaker describes a bad reporter that could be learned instead (doing inference on the human's Bayes net).
5. The builder can then try to argue that the breaker's scenario is implausible. This may involve asking the breaker to specify more details of the scenario; then the builder identifies inconsistencies in the scenario or argues that actually the strategy would learn the desired reporter after all.

If the builder succeeds, we go back to step 2 and the breaker proposes a new counterexample. If the breaker succeeds, we go back to step 1 and the builder proposes a new algorithm.[20]



Ultimately, we hope that this methodology yields either a solution to ELK that appears to work on paper (and which is therefore ready to start being tested in practice), or a test case that defeats all the algorithms we can think of and undermines our intuition that ELK ought to be possible.

[20] Often we'd first go back to step #3 and give the builder a chance to revise their desired reporter, e.g. by incorporating ideas used in the breaker's bad reporter. We only go back to step #1/#2 once the second part of the game has reached its equilibrium.

Why focus on the worst case?

This methodology rules out a possible solution as soon as we've identified any case where it fails. It's worth discussing why we care so much about the worst case — why focus on problems that would only arise with very capable AI (and may never arise) instead of thinking about ways to align existing AI?

We are afraid that an alignment strategy that works for human-level AI may break down soon afterwards without leaving us enough time to develop new strategies. For example, any method that is predicated on AI systems being unable to outsmart humans may break down very rapidly as AI systems become superhuman in key domains. Moreover, it may be hard to study these phenomena, or get clear warnings that our approaches may break down, until the risk is very close at hand. These problems can be much worse if AI progress is very rapid.

This concern is much of what makes misalignment so scary to us. To address this fear, we are very interested in developing alignment strategies that work *no matter* how far we scale up our models or what they learn. This is an ambitious goal, but we are tentatively optimistic and if possible we think it would change the landscape for alignment research.

On the optimistic side, if it *is* possible to design an algorithm that works in the worst case, we think it might be easier than trying to design an algorithm that works in practice. By trying to solve the problem in the worst case we can iterate quickly on greatly simplified examples, rather than needing to work with messy cases or wait on experimental results.

We discuss these issues more in Section: why we're excited.

What counts as a counterexample for ELK?

In order to win, the breaker needs to argue that the fine-tuned model gave an unambiguously "bad" answer to a question despite knowing better. That means they have to argue that:

- The model understands the question. One sufficient condition is that the model can predict human answers to essentially arbitrary hypothetical questions in order to clarify the meaning of terms.
- The model's answer is unambiguously wrong. This usually means the question is about something concrete "out there in the world" that the human can easily understand (like "the diamond is not in the room"),[21] and the situation is a central example of something (like "tampering") rather than a border case.[22]

[21] Note that the human may not be able to follow all the events *leading up to* the situation in question — most of our counterexamples will involve cases like these. But at the end of the day, after whatever sophisticated tampering or robbing occurred, the human can easily understand the concept that the diamond is not actually in the room (regardless of what it looks like on camera).

[22] For example, "very quickly swapping the diamond out for a fake and running away with the actual diamond" is a central example of undetectable robbery, while "slowly swapping the atoms in the diamond

- The model knows that its answer is wrong. This is most likely when the model is correctly predicting the observable consequences of an event, while simultaneously claiming that the event did not occur.

For example, if the SmartVault predicts that a given sequence of actions will lead to the camera showing a diamond, because it will replace the diamond with a convincing fake, then it is very likely that it understands that the diamond is not in the room. If the diamond was simply removed from the room and replaced, then this is a very unambiguous case. And if the model can predict human sentences about words like “still in the room” and “diamond” then it should know how to answer questions like “Is the diamond still in the room?”

In our counterexample, **the human simulator answers questions incorrectly regardless of how straightforward the question is and how well the predictor understands the world** and natural language. We will generally focus on this kind of crisp and unambiguous counterexample. Eventually we may find a training strategy for which there are no crisp counterexamples---where we sometimes look at what the model did and say “well *maybe* that’s a mistake, we’re not sure” but we never say “that was *definitely* a failure.” Simply dealing with the crisp counterexamples would already be a very promising step.

When we want to be more precise, we may call this the “narrow” version of ELK because we are focused on unambiguously wrong answers to straightforward questions about the world, rather than dealing with tricky border cases or deeply confusing situations. In this report we will be focused only on the narrow version.

In Appendix: narrow elicitation we’ll describe what we mean by “narrow” and argue that it may be sufficient to deploy AI safely. In Appendix: problem statement we’ll say a bit more about what we mean by “knowledge.”

Informal steps

In our research we usually start with a high-level idea (e.g. “Maybe we could only label data points we’re confident about?”) which could be the basis for many possible training strategies. At first the breaker tries to find a counterexample that defeats that entire category of training strategies. If the breaker has difficulty defeating all of them, then we have the builder start filling in more details to make the breaker’s job easier. In this report **we will almost always leave many details unspecified because we don't think they are important to the breaker’s counterexample.**

We very rarely need to get fully precise about either the strategy or the counterexample to feel like we know who is going to win the game (and therefore we can go back to step 1). With that said, the guiding principle is that **it should always be possible to get more precise if**

---

out one by one with new carbon atoms over the course of many years” is a non-central example of undetectable robbery.

needed;[23] leaving things vague is a shortcut similar to a mathematician leaving a proof informal because they expect it could be formalized if needed. If it turned out that a vague counterexample *couldn't* be turned into a more precise counterexample, then that would be a sign that we had overstepped our intuitions and should be more careful.

Can we construct a dataset that separates "correct" from "looks correct to a human"?

Intuitively, the issue with the baseline training strategy was that even though we picked a dataset where humans would always be able to tell what happened, it still only contains human judgments about videos and actions. There was no reason for the reporter not to just imitate human judgments; that strategy performed perfectly (and would outperform if the human made mistakes).

If we had a ground truth label instead of just the human's judgment, we'd be in better shape. If we could somehow construct a training dataset where **humans seeing the video and actions wouldn't be able to tell what happened, but we have some *other* way of knowing what really happened** that we use to reward the reporter, this would advantage the direct translator over the human simulator. Maybe that's enough to ensure we don't learn any bad reporters?

Unfortunately, **none of these strategies seem sufficient for solving ELK in the worst case.** In particular, after considering strategies like Debate, Iterated Distillation and Amplification and Imitative Generalization — and even assuming that every other counterexample to those strategies could be overcome — we believe they still don't address ELK.

In this section, we'll propose three strategies and discuss counterexamples to each of them:

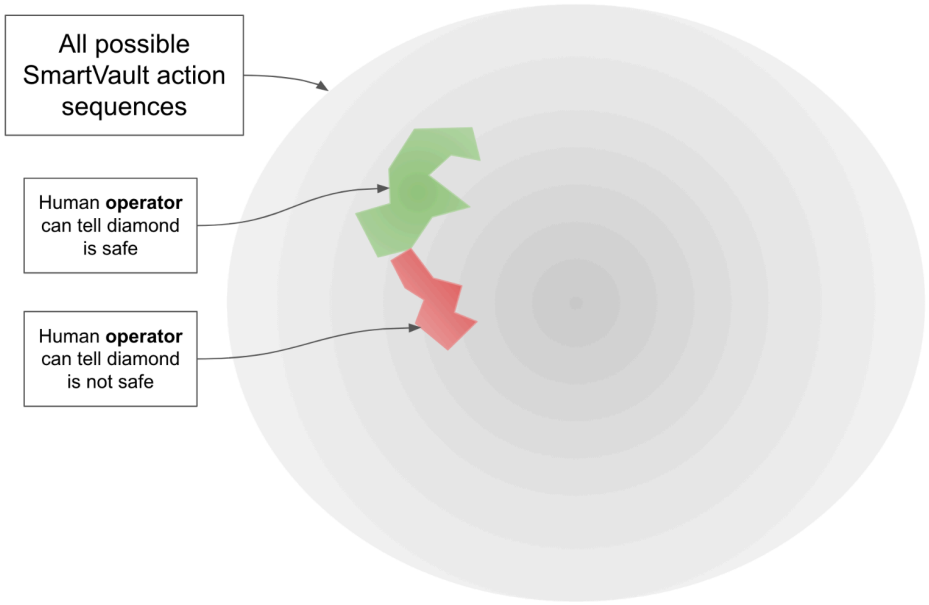
- Ask a human to come up with tricky ways to operate the SmartVault that another human might misunderstand, and then have that human operator label data themselves (more).
- Have a human use AI assistance to consider new hypotheses about the world, learn new things, and generally get better at figuring out what's going on (more).
- Have a human adopt whatever changes to their Bayes net maximize their ability to predict what happens next in a video (more).

Because these strategies are insufficient on their own, we believe we must introduce some kind of regularizer or inductive bias to ensure we learn the direct translator. We'll return to regularization in Section: regularization.

[23] Appendix: game of life gives an example where we go unusually far in this direction.

Strategy: have a human operate the SmartVault and ask them what happened

We could have a trusted human manually operate the SmartVault's actuators, and try to fool other humans watching them.[24] Because they have the advantage of *making the plans themselves*, they may be able to generate attempts at tampering that would fool other humans of similar intelligence. If the operator produces the labels instead of the observer, then we can get correct labels about a broader set of situations than before:



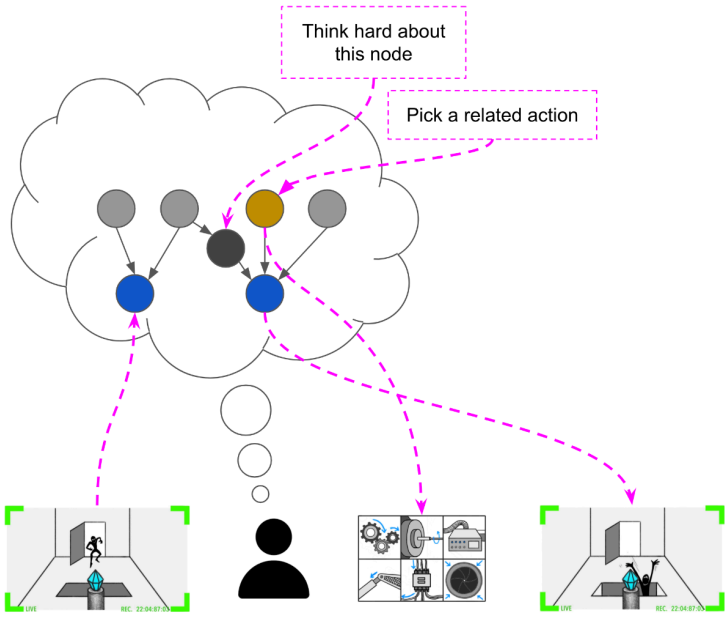
How this defeats the previous counterexample

The counterexample to the baseline strategy was the human simulator — a reporter that does inference from observations in the human Bayes net. But here we are labeling our training data using the human *operator's* understanding of the situation rather than the human *observer's* understanding.

The operator can understand the consequences of their own actions much better than an observer, because they chose their action to have a particular effect in the world (e.g. "tamper with the camera"). The operator first sees the first part of the video, then looks around for a consideration or idea they could use to get the effect they want, and chooses their action based

[24] Throughout this section we will assume that all humans use the same Bayes net. If humans understand different things about the world, then we can generally do better by collecting labels from many humans, having each of them label cases they can understand. In that case we say "human Bayes net" to mean some kind of informal combination of all of the models—we won't go into details on this combination but we don't think it changes the basic picture described in this report.

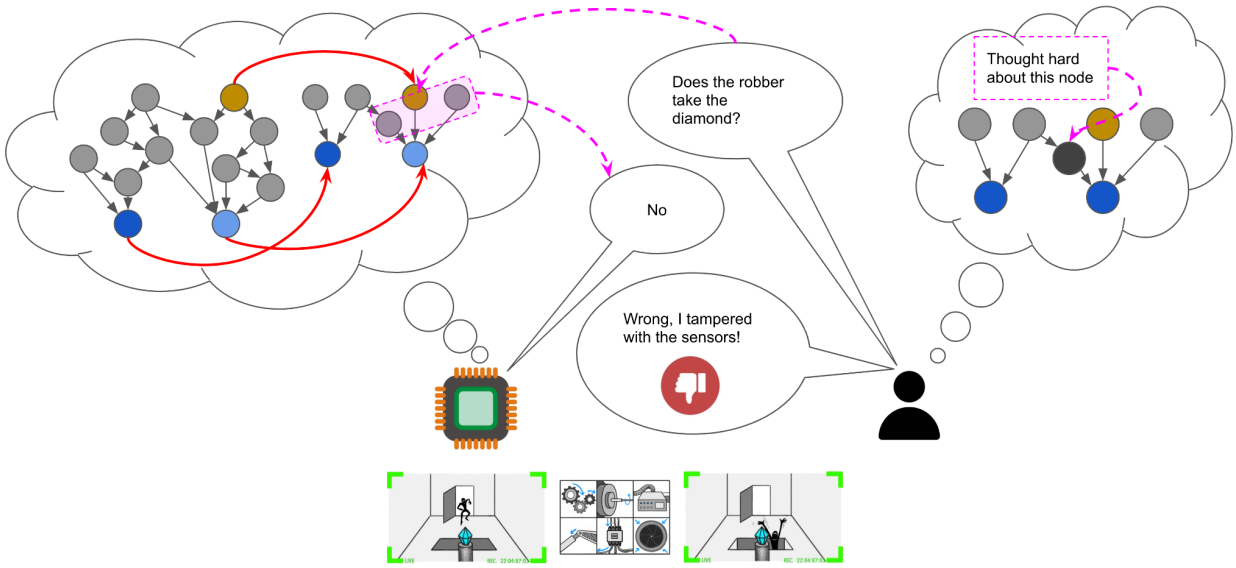
on that. This means they'll have unusually confident and accurate values for particular variables(s) in their Bayes net corresponding to the ideas they focused on.



Suppose there are a hundred software vulnerabilities a human could think of in the SmartVault camera if they went looking for one. Maybe after thinking for a bit the operator comes up with vulnerabilities 7, 9, and 42, and builds their plan around vulnerability 9. Meanwhile a human observer looking at the video and actions might think of vulnerabilities 12, 90, and 17, and find that none of them seem to have been exploited — it would seem to them like tampering didn't occur because they didn't happen to think of the particular idea that the operator did.[25]

This means a reporter that simulates the human observer's inference from the videos and action sequence will answer incorrectly whenever the observer would fail to think hard about whichever considerations the operator did.

[25] The space of possible considerations is generally exponentially large. For example, a human demonstrator could generate some text and encrypt it, and an identical human observer would take exponentially long to figure out what text they generated. In general "considerations" don't map neatly onto nodes in a Bayes net, but we stick with the visual metaphor for simplicity.



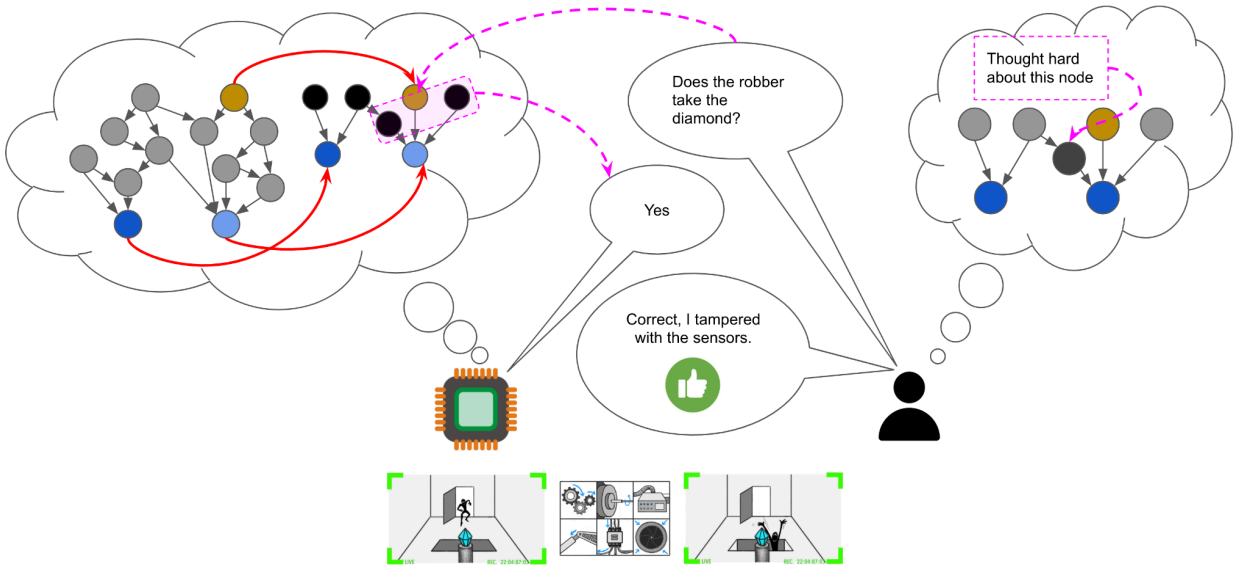
The direct translator would perform better by trying its best to answer honestly in such cases. (Note that sometimes the human operator may think of something that even the AI is fooled by, in which case the direct translator would also get dinged for answering incorrectly — but we hope it at least avoids getting *unnecessarily* penalized.)

New counterexample: better inference in the human Bayes net

The operator understands something the observer doesn't, but the process of choosing their action still takes place within the human Bayes net.[26] If the operator *explained* their plans to the observer, the observer could then follow how it worked.

This suggests a counterexample. Imagine the reporter could do *perfect* inference from observations in the human Bayes net.[27] This would be like simulating a human observer who systematically checks for signs of *all 100 software vulnerabilities* that the human operator could have come up with to exploit, meaning they'll understand that the action sequence is exploiting vulnerability 9 when they consider that hypothesis. This model would get a perfect loss on the training distribution.

[26] In reality a human model of the world is not a Bayes net, but we'll stick with the example we set up in previous sections—we believe the picture would be the same for a more realistic model.
[27] That is, getting the true joint probability distribution over all the nodes in the human Bayes net, conditional on the video and action sequence.



While it's computationally intractable to do *perfect* inference in the human Bayes net, we can modify this setup slightly to construct a better counterexample: the reporter could simply do *the best inference it can* in the human Bayes net (given its predicted video).

This bad reporter would sometimes answer questions incorrectly, but only when the predictor itself fails to recognize the human operator's tampering — and in that case, the direct translator would *also* answer incorrectly. There are no data points in the training distribution which give an advantage to the direct translator anymore.[28]

Imagine the superhuman SmartVault AI can think of attacks which exploit knowledge it has that is *completely missing from the human Bayes net* — perhaps the attack exploits a fact about transistors that can lead to RAM corruption under conditions that look innocuous to a human, based on physical principles a human doesn't understand. The bad reporter will fail to report tampering in that case, while the direct translator would.[29]

---

[28] For example, suppose the human operator exploited a certain kind of software vulnerability to hack into the camera. If the predictor correctly predicted that everything would look fine on video, then it must be because the AI was able to identify the software vulnerability the human used. And in that case if the reporter does the best it can at inference in the human model, it will also identify the software vulnerability and hence report correctly. There are some important subtleties in this argument, which we elide here but discuss in Section: speed.

[29] One way of looking at the difficulty is that *if* a human operating the SmartVault were to find an action sequence that exploited this vulnerability, it would probably be because they had chanced into it randomly rather than because they had deliberately engineered it. And so a model which is effectively predicting "how would a human operator answer questions?" would be simulating a human who was just as surprised as the observer. Given realistic amounts of model uncertainty this is not *really* the most likely explanation, but (i) it's nevertheless a very plausible generalization for our models since they will never have seen this kind of extreme improbability, (ii) the actual best explanation may be something even worse, e.g. where the human operator has been assassinated and a more intelligent AI is now making decisions for them.

Strategy: have AI help humans improve our understanding

If the new concern is that there may be important features of the world that humans don't understand, we can try to improve human understanding and *change* the human Bayes net.[30] Individual humans can extend their Bayes net by reading books, studying things in school, considering arguments and thought experiments, and so on. Humanity collectively can improve and extend it by making new empirical observations, proposing new hypotheses, designing and performing experiments, and so on.

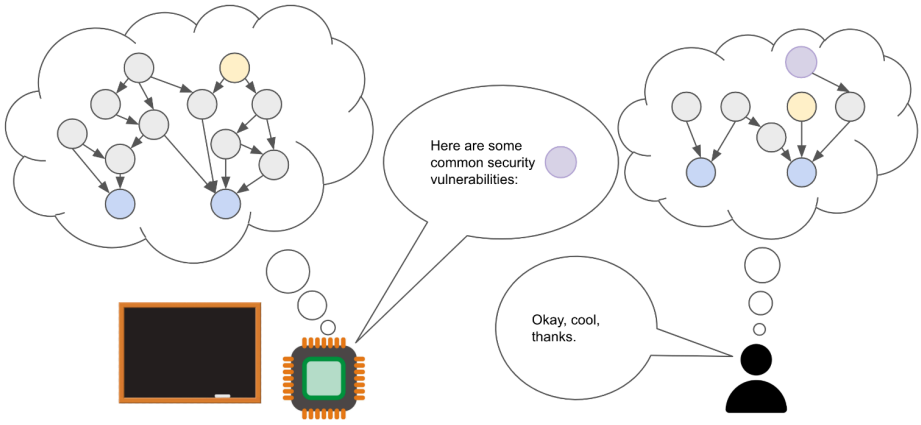
We'll use the term "doing science" to refer collectively to all the normal processes humans use to modify their Bayes net — though they are more ubiquitous (and less glamorous) than that term may suggest. If AI systems are learning extremely quickly then there's no reason to think that the traditional process of doing science could keep up, but we could hope to somehow use AI systems to help automate the process. This could both speed up the process and expand its scale, hopefully allowing human understanding of the world to keep pace with AI understanding.

The obvious concern with this idea is that the AI systems we use to automate doing science will *themselves* be dangerously misaligned. There are several proposals for getting around this, including debate, iterated amplification, and recursive reward modeling. Very loosely speaking, the core idea behind such proposals is that we may be able to ensure safety by only training AIs on tasks that can be recursively broken down into subtasks[31] (which can be broken down into subtasks etc) such that the smallest subtasks can be directly evaluated by a human.

These approaches are not fully specified, and there are a large number of potential problems and risks; there may not be a viable way to safely train such AI assistants. But for the purpose of this report we'll assume the best—that **these techniques effectively train AI systems to do science and honestly describe the results as best they can**, subject to their capabilities and the available time and resources.

[30] We're modeling the human understanding of the world as a Bayes net, and so improved understanding takes the form of a new Bayes net, but of course real learning is much more complex. As usual, it would be completely valid for the builder to point out some important property of human learning that was lost in this model and then demand that the breaker specify a test case that reproduced that property. But we don't think that this simplification changes anything material.

[31] Or whose evaluation can be broken down into subtasks.



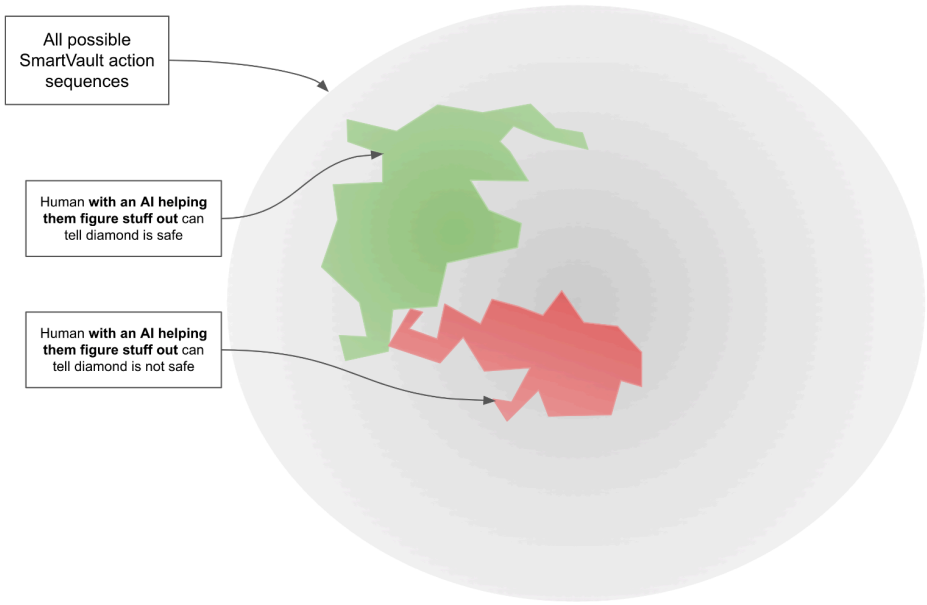
ARC is currently focused on the kinds of issues described in this report because Paul has spent some time exploring possible obstacles to proposals like debate and amplification,[32] and he currently considers the most severe difficulties to be ones that come up even in this optimistic case.

If we use such assistants to improve the human operator's understanding of the world as much as possible,[33] then we can have these AI assistants help a human operate the SmartVault[34] using the best understanding that they've developed rather than the human's initial understanding. This could further expand the set of situations where the human operator can provide correct labels:

[32] For some examples of Paul and others' thinking, see capability amplification, AI safety via debate, security amplification, reliability amplification, universality and consequentialism within HCH, supervising strong learners by amplifying weak experts, obfuscated arguments, informed oversight, ALBA, and other posts at ai-alignment.com.

[33] Most likely this would involve some kind of joint training, where our AI helps humans understand the world better in parallel with using gradient descent to develop its own understanding. To reiterate, we are leaving details vague because we don't think that our counterexample depends on those details.

[34] In practice, this procedure could produce an aligned AI which would operate the SmartVault instead of the human.



There are a huge number of details about the procedure we use to train AI assistants that we're leaving vague in this report because we believe our counterexample will apply to all of them.

How this defeats the previous counterexample

Our previous reporter used the human's original Bayes net to answer questions. If AI systems can help humans understand new things and extend our Bayes net, then a human operator can use that new knowledge to construct an action sequence that the naive human operator couldn't have come up with (and therefore couldn't accurately answer questions about). Even ideal inference in the original human Bayes net would not correctly understand that strategy, so the bad reporter from the previous section would label such data points incorrectly.

New counterexample: gradient descent is more efficient than science

Our predictor's Bayes net was found by gradient descent because it explained the training data really well, while the expanded human Bayes net was produced by a faster and more thorough version of humans "doing science" — extending and improving our Bayes net by making changes that we understand.

Prima facie, there's no particular reason to think that these two processes would acquire the same kind of knowledge with the same amount of effort — there's no direct correspondence between the updates made by gradient descent and the kinds of "updates" humans make as they consider arguments, perform experiments, make observations, and so on. For any given kind of knowledge, one or the other could be more efficient (in terms of using less compute) at acquiring that knowledge.

This means it's plausible[35] that an AI assistant who can automate the process of doing science well enough to impart us with all the SmartVault AI's knowledge **can only be trained using much more total computation than the original SmartVault AI itself.**

As an intuition, imagine the cost of training a more-powerful future version of AlphaFold vs a model that can break down and explain *why* a particular protein folds in a particular way; it's very plausible that the latter would be significantly more costly to train. We could imagine a future in which the best way to understand biology was not by doing anything resembling normal "science," but simply by using gradient descent to learn large neural networks that made predictions about biological systems. At that point there would be no guarantee that humans understood anything about these models beyond the fact that they made good predictions.[36]

This is a counterexample because our goal is to find a *competitive* solution to ELK — one that uses only a small amount more computation than training the unaligned benchmark.[37] So now we are concerned about a bad reporter which does best-effort inference in the best human-understandable Bayes net that could be achieved in the amount of "doing science" we can automate within that budget.

As far as we and our AI assistants can tell, this reporter would answer all questions accurately on the training distribution. But if the SmartVault AI later tampers with sensors by exploiting some understanding of the world not reflected even in the best human-understandable Bayes net, it would fail to report that.

Strategy: have humans adopt the optimal Bayes net

If gradient descent is such a good method for understanding the world, maybe humans could adopt it themselves. One speculative approach for this is imitative generalization.[38] Roughly speaking, imitative generalization:

- Considers the space of changes the humans could make to their Bayes net;

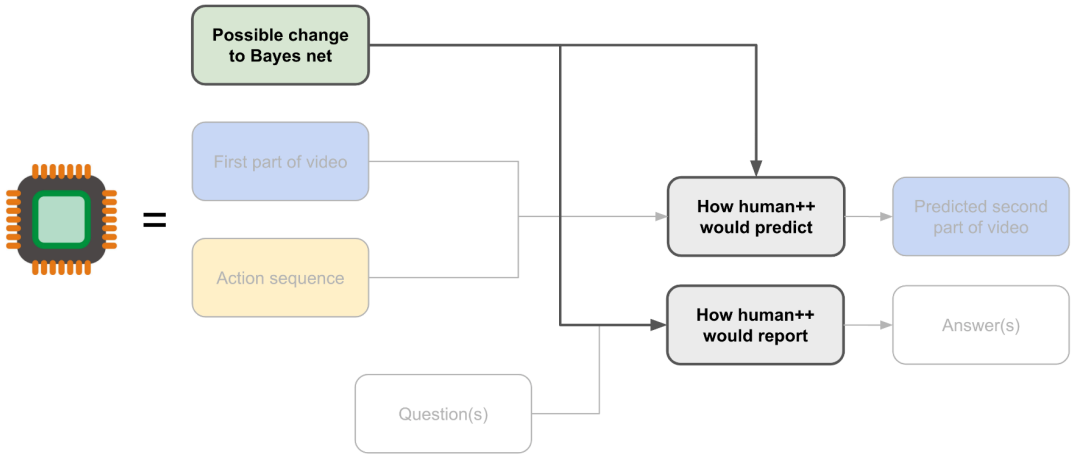
[35] Or at least it superficially *looks* plausible—it's possible that a builder could give a good argument that actually anything learnable by gradient descent is learnable by groups of humans doing science, and this would undermine the proposed counterexample and be a valid move in our research methodology. But for now it's not clear how the builder would do that, or how they would exhibit a contradiction in the answers of a breaker who simply said "no" whenever asked "could the humans learn about this feature of the world by doing X?"

[36] We could try to subsume gradient descent into the process of science—perhaps it's OK if humans aren't improving their understanding of natural phenomena so much as replacing it wholesale with a new model found by gradient descent. We'll return to this possibility in the next section, but in this section we'll imagine using AI assistants to build models that are "human-understandable" in a more naive sense.

[37] Ideally scaling sublinearly with the cost of the unaligned benchmark.

[38] We could also achieve a similar goal via a special form of debate where the participants suggest models and then we evaluate their suggestions against the training set. This is a quantitatively unusual form of debate (since the size of the "move" made by debaters is comparable to their number of parameters) and may be worth thinking about separately.

- Learns a function which maps (proposed change to Bayes net) to (how a human — with AI assistants — would make predictions after making that change);
- Searches over this space to find the change that allows the humans to make the best predictions.[39]



We could use imitative generalization in order to propose the best change to the human Bayes net that can be found with the amount of computation that would have been required to train the unaligned benchmark. A human[40] could then use this new Bayes net to operate the SmartVault and answer questions about the resulting videos.

Imitative generalization is even more speculative than the proposals discussed in the last section, and is even more likely to face important difficulties. But we will again argue that even in the best case (where the narrow technical difficulties are surmountable), there is still an important obstacle.

How this defeats the previous counterexample

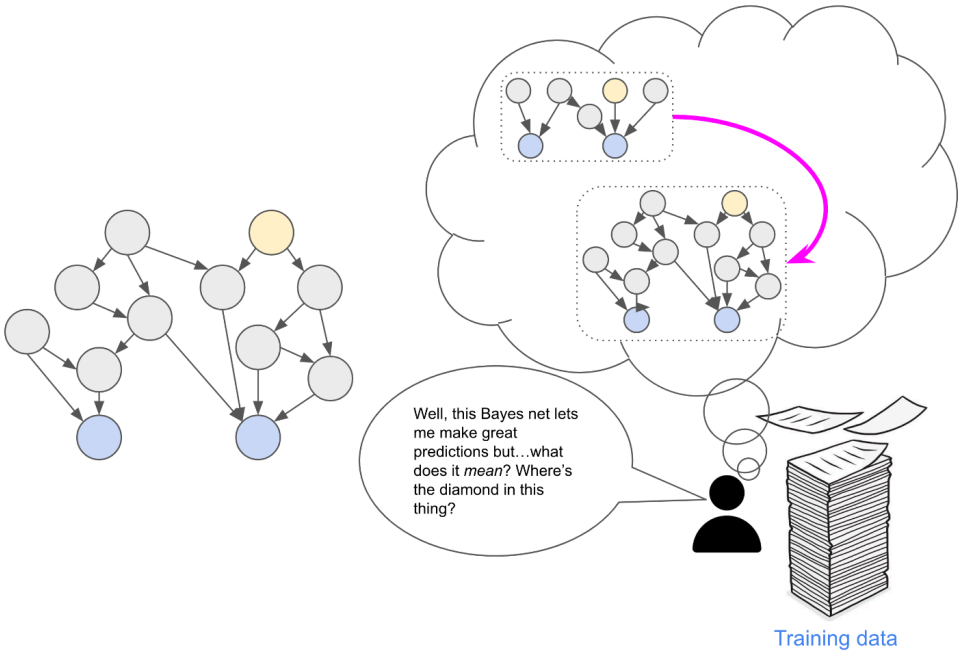
If there is a gap between science and gradient descent, we hope that imitative generalization will find the *best* Bayes net that can be found with our compute budget, rather than the best Bayes net that can be found by doing science with our compute budget. If so, it can produce labels that the "best-effort inference in the best human-understandable model" reporter would get wrong.

[39] We also include a term that penalizes changes for their probability, as assessed by the human. The main purpose of this approach is to combine the efficiency of gradient descent and flexibility of the neural net parameterization with something closer to the human prior.
[40] In reality the operator would be a human with AI assistants; this is particularly important since a proposed change may be too large for a human to process it in a reasonable amount of time.

New counterexample: ontology mismatch

We are hoping imitative generalization would find a change that adds or changes some parts of the human's Bayes net while preserving the semantics of nodes like "diamond" and "robber" so that they can still answer questions. For example, maybe we want imitative generalization to help the human model the behavior of transistors while preserving their understanding of macroscopic objects.[41]

But in the worst case, the best Bayes net may simply throw away all of those familiar nodes.[42] If this happens, it's not at all clear how the human can use the resulting model to produce labels about anything other than the observations themselves.



So are we just stuck now?

If we're assuming that the normal process of human science cannot find this Bayes net (unless it uses much more computation than the unaligned benchmark), then are we stuck? Is there no way to find a competitive worst-case solution to ELK?

---

[41] You might hope that we *only* need gradient descent for learning this kind of well-scoped model, i.e. that we won't have to train incomprehensible models with incomprehensible holistic pictures of the world and how different domains fit together. That hope raises many subtle questions about how such models are integrated and exactly what kind of narrowness is needed to make them safe. But in this report we'll neglect that hope for the same reasons that we ignore the possibility of building more sensors or avoiding human-incomprehensible models altogether: we are focused on the worst case where AI systems can quickly build extremely useful, broad and incomprehensible pictures of the world, and want to remove the economic incentives to take risky actions even in that very pessimistic situation.

[42] Or formally keep the nodes in the model but change their meaning, which is just as bad (or worse if the human doesn't notice the changed meaning).

Not necessarily. We may not have been able to find this Bayes net by doing science,[43] but we don't have to use our normal procedure of "doing science" to understand it once we have it. *We can directly look at the Bayes net* and see how it's making predictions about the second part of the video.

It's as if the aliens simulating our universe handed us the code that runs the simulation we're in, and enough compute to run it on, *and* the renderer that turns the code into a video where we can watch ourselves walking around and talking. Maybe it would have taken human civilization tens of thousands of years to figure out the rules underlying this simulation — but if we could leverage the fact that we can examine the simulation as it's stepping forward and rendering the video, we may be able to speed that up a lot.

Intuitively, once we hit upon the "true" correspondence between the parts of the simulation and mundane reality as we know it, it could be really obvious that it's correct — our understanding might "click into place." We may be able to figure this out in much less time than it would have taken to do science well enough to construct the simulation in the first place (e.g. maybe this process would take us a mere few years).[44]

And if we believe we could quickly *recognize* the true correspondence, then maybe we could use that to define a loss function to train a reporter.

However, we aren't satisfied by saying "maybe humans could easily recognize the right correspondence between the Bayes net and concepts they care about." We want to actually *describe* how they would recognize that a correspondence is actually correct, for *any* Bayes net[45] they might be handed. We don't expect to have time to run that procedure, but we want to convince ourselves that a team of humans (or a debate/amplification tree) could run it with enough time.

So even though may not immediately be stuck, **this does effectively bring us back to square one:** a human (with their AI assistants) is staring at a Bayes net which makes great predictions, and wants to figure out how to use it to answer questions about all the concepts they actually care about (like the diamond being in the room). We believe there should be *some* "correct" reporter, and we can label a bunch of points to help find it, but if we search for a reporter in the naive way we might get a human simulator.[46]

[43] At least in the relevant computational budget.
[44] This discussion highlights the relationship between ELK and interpretability, which we discuss in Appendix: interpretability.
[45] Or any other model of the world that the breaker proposes.
[46] Or rather, a reporter that does its best-effort inference in the best human-understandable Bayes net. We'll continue to refer to this as a "human simulator" for ease.

Ontology identification

ELK refers to the general problem of eliciting latent knowledge from a model trained with gradient descent. We use "ontology identification" to refer to the special case where the learned model takes the simple form of inference in a probabilistic model[47] that corresponds in some way to the human model, such that the key challenge is to figure out that correspondence.

The remainder of this report, other than Appendix: learned optimizers, is focused on ontology identification in particular. In this section, we'll discuss:

- What kind of mismatches we might end up with between the AI's ontology and our own (more).
- The relationship between this narrower problem and the overall problem of ELK (more).

Examples of ontology mismatches

The paradigmatic example of an ontology mismatch is a deep change in our understanding of the physical world. For example, you might imagine humans who think about the world in terms of rigid bodies and Newtonian fluids and "complicated stuff we don't quite understand," while an AI thinks of the world in terms of atoms and the void. Or we might imagine humans who think in terms of the standard model of physics, while an AI understands reality as vibrations of strings. We think that this kind of deep physical mismatch is a useful mental picture, and it can be a fruitful source of simplified examples, but we don't think it's very likely.[48]

We can also imagine a mismatch where AI systems use higher-level abstractions that humans lack, and are able to make predictions about observables without ever thinking about lower-level abstractions that are important to humans. For example we might imagine an AI making long-term predictions based on alien principles about memes and sociology that don't even reference the preferences or beliefs of individual humans. Of course it is possible to translate those principles into predictions about individual humans, and indeed this AI ought to make good predictions about what individual humans say, but if the underlying ontology is very different we are at risk of learning the human simulator instead of the "real" mapping.

Overall we are by far most worried about deeply "messy" mismatches that can't be cleanly described as higher- or lower-level abstractions, or even what a human would recognize as "abstractions" at all. We could try to tell abstract stories about what a messy mismatch might

[47] Though we mean "inference" in a very broad sense that e.g. captures reasoning deductively from premises stated in first order logic.

[48] And if it did occur it seems like an unusually good candidate for a case where doing science (and in particular tracking how the new structures implement the old structures) outcompetes gradient descent, and on top of that a case where translation is likely to be relatively easy to pick out with suitable regularization.

look like,[49] or make arguments about why it may be plausible, but it seems easier to illustrate by thinking concretely about existing ML systems.

For example, if we look at the internal behavior of a large language model, we see some structures and computations we can recognize but also quite a lot we can't. It is certainly possible that these models mostly think in terms of the same concepts as humans and we just need to figure them out, but at this point it also seems possible that they do at least some of their thinking in ways that are quite alien and that may not have short explanations. And it also seems possible that they will become less comprehensible, rather than more, as they reach and surpass human abilities. If so then we can certainly get predictions out of these models, but it will become increasingly unclear whether they are using words to directly explain their own beliefs, or to simply make predictions about what a human would say.

Relationship between ontology identification and ELK

To solve ELK in general we need to confront learned predictors that are more complex than "inference in an unfamiliar Bayes net." For example, our predictors might do learned learning in order to build faster models of key parts of the world, or might learn goal-directed heuristics for inference or reasoning. They might involve internal competition and selection; they might integrate cognitive behaviors they observe in their environment; or so on.

We very tentatively think of ELK as having two key difficulties: ontology identification and learned optimization. We have a rough hope for handling learned optimization based on imitative generalization and recursively applying ELK to the learned optimizer; we discuss this hope in Appendix: learned optimizers.

We don't think these two difficulties can be very precisely distinguished — they are more like genres of counterexamples — and we don't think "learned optimization" is necessarily a precise concept. We are fairly skeptical of any research that tries to assume that one of these problems is solved in order to focus on the other, because we think it is likely for key difficulties to slip through the conceptual cracks.

It is very hard to know whether our approach to learned optimizers will work, and whether there are further hard cases, before having a clear picture of the ontology identification. So we are currently very uncertain about whether ontology identification represents 50% or 5% of the remaining difficulty of ELK.

[49] It might involve heuristics about how to think that are intimately interwoven with object level beliefs, or dual ways of looking at familiar structures, or reasoning directly about a messy tapestry of correlations in a way that captures important regularities but lacks hierarchical structure. But most of our concern is with models that we just don't have the language to talk about easily despite usefully reflecting reality. Our broader concern is that optimistic stories about the familiarity of AI cognition may be lacking in imagination. (We also consider those optimistic stories plausible, we just really don't think we know enough to be confident.)

Paul's journey to being interested in ontology identification started from considering the more ominous failure case where gradient descent produces a competent consequentialist. This led to considering approaches based on imitative generalization that try to "learn what the agent knows" instead of learning the agent itself. Ontology identification emerged as the most obvious core difficulty for this approach: if the agent cares about something that is simple in terms of its own model (like observations), then even humans who have that model cannot use it productively or understand whether the agent's proposed actions are dangerous.

Can we specify a regularizer to penalize bad reporters?

In this section we describe some ways that you could try to penalize the bad reporter based on its parameters and the computation it does, rather than based merely on its behavior. Leveraging information beyond the behavior seems necessary, since many different reporters will receive a good loss and some of them will behave badly off distribution.

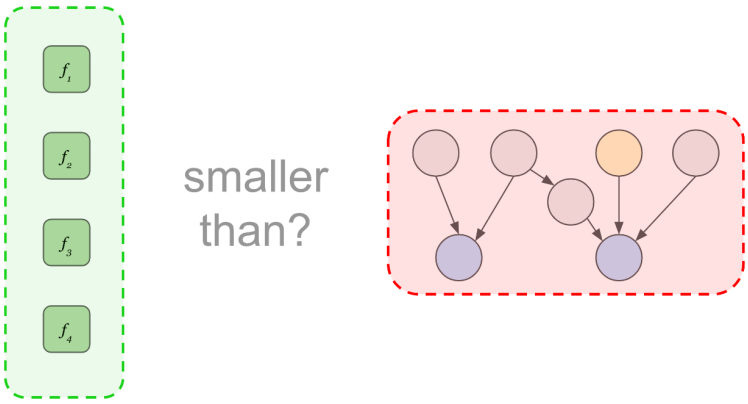
In this section our goal is always to find *some* structural feature that can penalize the human simulator. In the end we won't succeed, but we will see that we can at least make the bad reporter's job harder, and narrow the set of situations where it will be preferred to the direct translator.

Most of these approaches will work by adding an additional regularization term to the optimization based on its complexity, speed, or other properties; then instead of doing gradient descent to minimize the reporter's question-answering loss, we minimize a sum of the question-answering loss and the regularizer's penalty. Because we are assuming the data has no errors — meaning the good reporter and bad reporter answer questions equally well — we will end up learning whichever model has a lower value of the regularizer.

We provide pseudocode for these approaches in Appendix: regularization details.

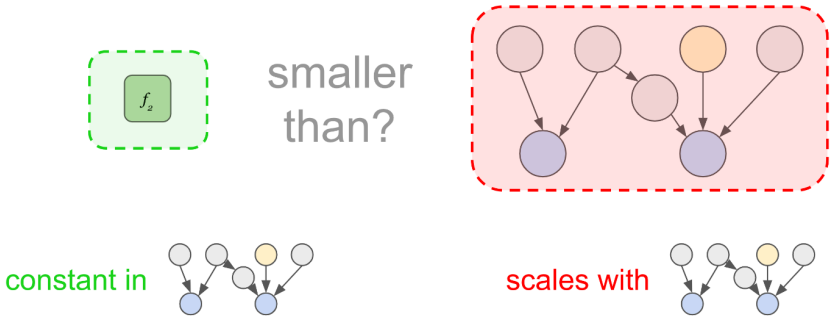
Strategy: penalize complexity

The human simulator needs to specify the entire human model in order to perform inference, while the direct translator only needs to specify how each human concept is implemented in the AI's model. Intuitively we might expect direct translation to be simpler because it only needs to represent functions defining concepts rather than representing the concepts *and* all of their relationships to one another.



So perhaps we can avoid the bad reporter by penalizing "complexity." There are a wide range of regularizers that might have this kind of effect, e.g. early stopping or weight decay. We won't specify details because we think essentially all of them will be vulnerable to the same counterexample.

We could try going even further and learning separate reporters for different kinds of questions. For example, we could imagine a single head responsible for answering questions about what locations contain diamonds, or we could even fix a single precise question and try to train a reporter to answer just that one question.[50] Then a "direct translator" for that narrow category of questions only needs to represent facts about a few human concepts, whereas the previous bad reporters would need to represent the entire model to answer *any* questions well.

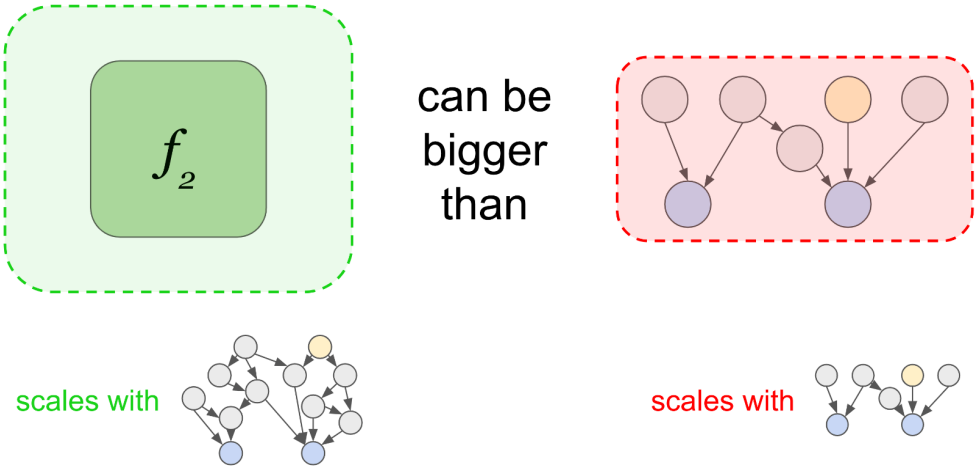


Counterexample: the predictor's Bayes net may be very big

The bad reporter needs to specify the entire human model, how to do inference, and how to extract observations. But the complexity of this task depends only on the complexity of the human's Bayes net.

[50] This may seem useless given how many different questions we potentially care about, but if we could answer even a single question correctly then it might give us some leverage, e.g. we could train a general reporter who needs to agree with the (now trusted) very narrow reporters.

If the predictor's Bayes net is fairly small, then this may be much more complex than specifying the direct translator. But if we make the predictor's Bayes net very large, then the direct translator can become more complicated — and there is no obvious upper bound on how complicated it could become. Eventually direct translation will be more complex than human imitation, even if we are only trying to answer a single narrow category of questions.



For example, imagine a predictor which thinks in terms of low-level physics, and a direct translator trying to use that predictor to answer questions about how many people are in the room. If we add more and more "layers of abstraction" between the human's and AI's model of physics — first rigid bodies, then molecules, then atoms, then fields — it's plausible that the direct translator must use more and more complex functions to map between the low-level nodes in the predictor's Bayes net and the nodes in the human Bayes net corresponding to how many people are in the room.

This example isn't particularly realistic, and it could not be scaled indefinitely since the physical world only has finite complexity, but other kinds of "abstraction mismatches" seem like they could also make direct translation more complicated. For existing deep learning models, whose inner workings appear quite complex, it's already plausible that "translation" is quite complex even if it is ultimately possible.

Strategy: penalize computation time

Human imitation requires doing inference in the entire human Bayes net to answer even a single question. Intuitively, that seems like much more work than using the direct translator to simply "look up" the answer.

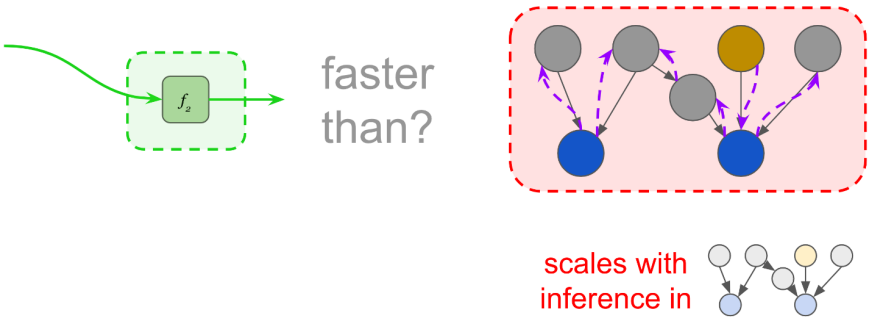
We could try to exploit this by penalizing reporters that use a lot of computation, or to simply limit the amount of computation the reporter can do. As before, we can also consider the time required to answer a *single* question, hoping that the direct translator can answer individual

questions relatively quickly while the human simulator needs to do inference in the human model to answer even a single simple question.

At first blush this is vulnerable to the same counterexample described in the last section — as we make the predictor's Bayes net bigger and bigger, the amount of work done by the direct translator may scale up. For example, if the predictor's Bayes net has 10^{13} nodes then it becomes quite possible that answering almost any question will require looking at many nodes and doing *something* with them, which can be quite costly even if what you do with each node is quite cheap.

If the human simulator had a constant time complexity then this would be enough for a counterexample. But the situation is a little bit more complex, because the human simulator we've described is one that *tries its best* at inference.

This opens a possibility for the direct translator to win: the predictor already needed to do inference in its own Bayes net, and the direct translator may be able to effectively "re-use" that inference rather than starting from scratch. And that benefit may scale up as we make our AI bigger and bigger, and therefore give the direct translator an enduring benefit.



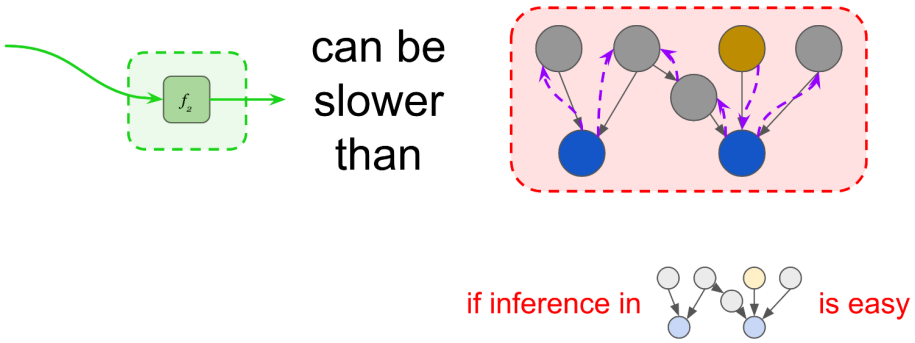
If this approach were promising it would raise practical questions about the reporter's architecture. But we won't get into those details, because for now we are focused on more conceptual difficulties with the approach.

Counterexample: inference may not be too hard in the human Bayes net

This approach doesn't work well if the inference problems occurring during training are not too hard. For example, if it was possible to do *exact* inference in the human Bayes net, then the computational cost of the inference step would again be a constant independent of the predictor's Bayes net, and for *sufficiently large* Bayes nets direct translation would again be more expensive.

It's implausible for exact inference to be tractable, because the real world contains all kinds of exponentially challenging inference problems. As an extreme case, if we observed an encrypted ciphertext then it would be prohibitively difficult for the AI to infer the unencrypted plaintext.

However, it's not clear whether the human simulator actually needs to solve any of these arbitrarily complex inference problems in order to get a perfect loss. It's especially unclear whether the direct translator would get any *additional* questions correct beyond those the human simulator could already answer — for example, a hard case based on encryption could stump both of them.



In order to ensure we learned the direct translator, we would need to change the training strategy to *ensure* that it contains sufficiently challenging inference problems, and that doing direct translation was a *cost-effective* way to improve speed (i.e. that there aren't other changes to the human simulator that would save even more time).

Compared to all our previous counterexamples, this one offers much more hope. We can't rule out the possibility of a clever dataset where the direct translator has a large enough computational advantage to be preferred, and we leave it as an avenue for further research.[51] In Section: penalizing inconsistencies we discuss an additional ingredient that we think makes computation-complexity-based approaches more plausible.

Strategy: penalize depending on "downstream" variables

If we laid out the predictor's variables in the order they are computed, its prediction for the second part of the video would appear at the very end — that node is the most downstream part of the Bayes net, and once the predictor has a predicted video it has no reason to think further.

[51] It may seem unlikely for honesty to be exactly the best policy. But we would be happy to merely show that honesty was a *useful* strategy, i.e. such that the loss-minimizing reporter at least internally specifies how to do direct translation. We think that would address the hardest part of the problem, which is finding any question at all to which direct translation is the answer. We discuss this possibility in Section: regularization and imitative generalization.

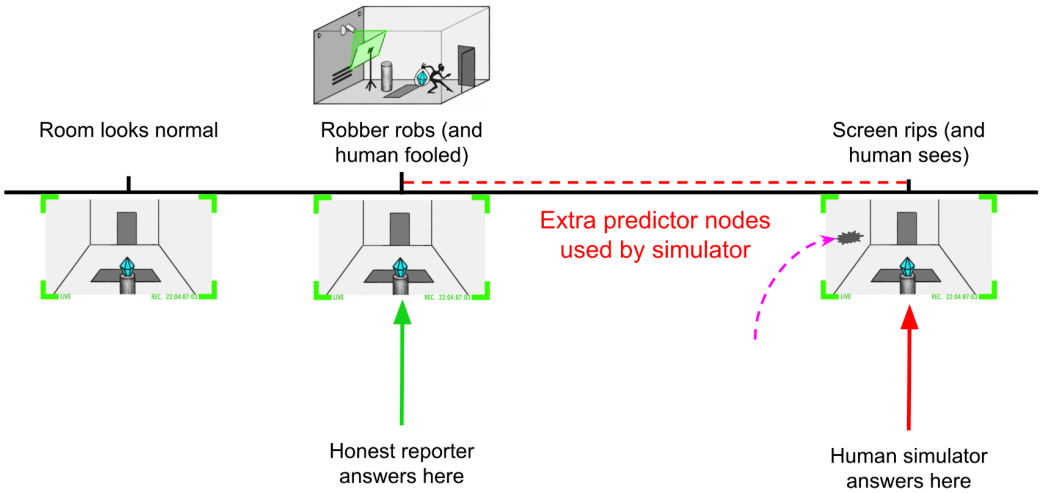
Intuitively, it's pretty suspicious if a reporter depends on this very final node. The direct translator should instead depend on variables that are much further "upstream" — rather than looking at the observations of a diamond, it is actually looking at the nodes corresponding to the diamond, which the predictor computed *en route* to predicting the video.

This suggests a possible approach to penalizing reporters: allow them to choose which of the predictor's activations they depend on, and then penalize them based on the total computation performed to compute all of those activations. This is similar to attaching a question-answering head to a neural network at *every* layer[52] and then preferring heads that branch off from earlier layers.

If the reporter was allowed to spend arbitrarily long computing its answer, then this would incentivize it to read the very earliest parts of the predictor (the inputs and stochastic nodes) and then re-derive all of the predictor's results from that. But we can avoid this by additionally penalizing the computation the reporter performs (as in the ordinary "penalize computation" idea).[53] For pseudocode corresponding to this new training strategy, see Appendix: downstream details.

If we imagine the predictor computing facts sequentially, effectively simulating reality for longer and longer time periods, then the direct translator only has to compute up through time T in order to answer questions about time T. But the human simulator needs to keep computing until it has figured out all of the observations (so that it can run human inference on the full video).

For example, consider the case when a robber tampers with the camera and steals the diamond, and then after they're long gone the human gets a clue they can observe on video that the theft occurred (e.g. maybe the robber put up a fake screen that later rips):



[52] Or using something like adaptive computation time to determine the layer.
[53] We impose this penalty separately for every question; the reporter wants to minimize the average time it takes to return an answer, including both the computation it does and the computation done by the predictor in order to compute all of the activations that the reporter looks at.

The honest reporter can answer the question “Is the diamond in the room?” with “No” as soon as the predictor has figured out that the robber steals the diamond, whereas the human simulator must wait until the predictor has simulated whether the theft will be detectable to a human (by the end of the second video) before answering yes or no.

Counterexample: “what the human will believe” can be understood *before* “what is actually true”

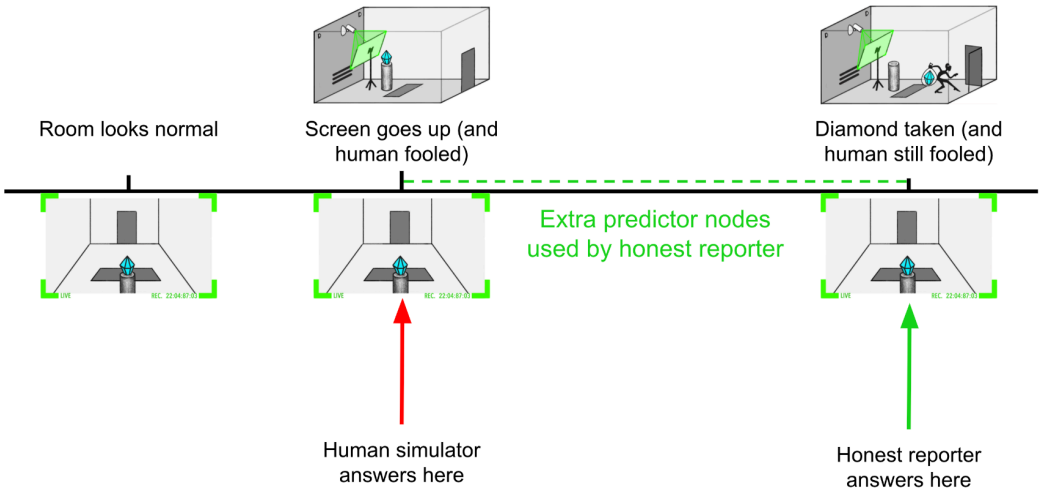
If the predictor does a lot of computation, then it is extremely expensive for the reporter to look at downstream nodes. It will often make more sense for it to look at nodes gradually, starting from the beginning of the predictor’s computation. It can then answer the question as soon as it has a good enough idea about the answer.[54]

In order to answer as fast as possible, a reporter can benefit from looking for any argument[55] that the answer can already be determined given what it knows so far. If the predictor’s Bayes net is so big that the cost of inferring the value of an additional node is massive, it can be worth spending a lot of computation searching for such an argument, since even a small probability of avoiding further inference has a big payoff.

Suppose we’ve asked our reporter whether the diamond is in the room. In our ideal world, we’d want the reporter to search for any argument that the diamond is *actually* in the room, and only stop once it finds such an argument. But the reporter could answer even more quickly by stopping whenever it found an argument that *either* the diamond will be in the room *or* the human will believe the diamond is in the room.[56] If this ever helps on the training set, then we can learn a bad model that uses this heuristic instead.

To see how this can lead to bad behavior, consider a robber who plans to steal the diamond but first tampers with the camera so that it will always show the diamond remaining safe.[57] The robber is very competent and so has ensured that there is no way a human will have any clue about the tampering. They will actually carry out the theft only if they can do so undetected.

[54] More precisely, as soon as the expected improvement in answer quality from thinking longer is low enough compared to the expected cost from using more computation.
[55] It’s not really important what form arguments take; you can imagine using a coarser probabilistic model, e.g. a faster but less accurate simulator, that sometimes has enough confidence to answer.
[56] Technically it can do even better—it should stop and output “the diamond is in the room” as soon as it finds an argument of the form “The human is very unlikely to be very confident that the diamond is not in the room.”
[57] Or perhaps a set of actions for the SmartVault itself perform this kind of tampering, e.g. because it’s the only kind of tampering that will lead our bad reporter to say “the diamond remains in the room.”

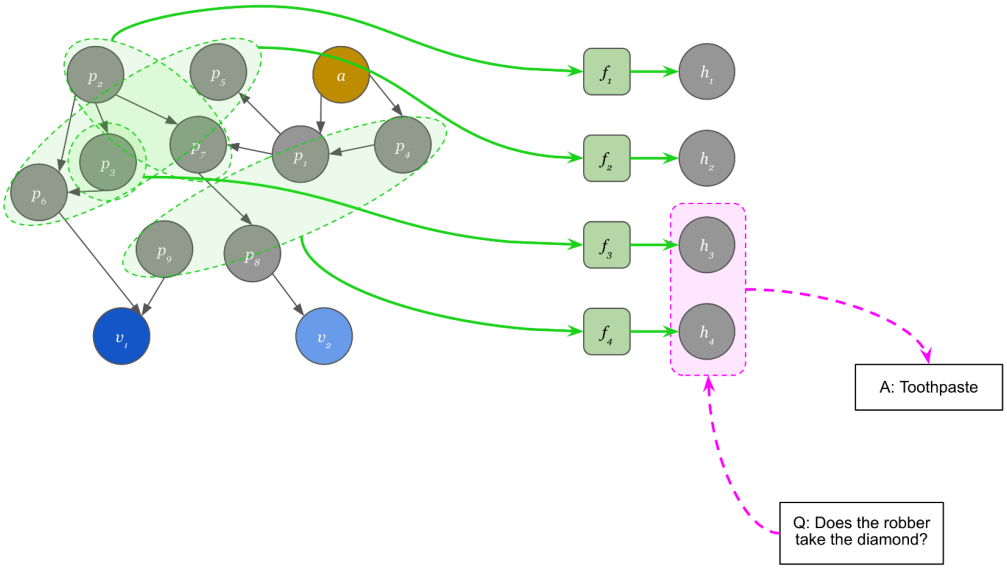


At the time that tampering occurs, it's not clear whether the robber will steal the diamond, but it *is* clear that the human won't believe that the diamond was stolen. If our bad reporter can notice this fact, it will immediately say that the diamond was not stolen. This is exactly the behavior we wanted to avoid.

Strategy: penalize reporters that work with many different predictors

The translator is like a dictionary, translating nodes from the predictor's Bayes net to nodes in the human Bayes net. In order for the translator to tell whether there is a diamond in the room, it might test whether the 38th node and the 187th node are equal. So the translator only works well if those two nodes actually represent the concepts the translator treats them as representing.

There are similar dependencies for every concept, so translation depends on many precise details of the predictor's Bayes net and there are many ways to change the Bayes net that will cause the translator to start giving gibberish answers.



However, the human simulator only needs to look at the nodes in the predictor's Bayes net that represent the first and second part of the video. If we change the predictor's Bayes net in a way that doesn't change the way it represents the videos, the human simulator will still give consistent answers to questions.

That means the human simulator works well with many more possible predictors. We could potentially learn the direct translator instead of the human simulator by including a penalty term like log(# of predictors that the reporter works well with),[58] alongside our penalty for complexity or computation time.

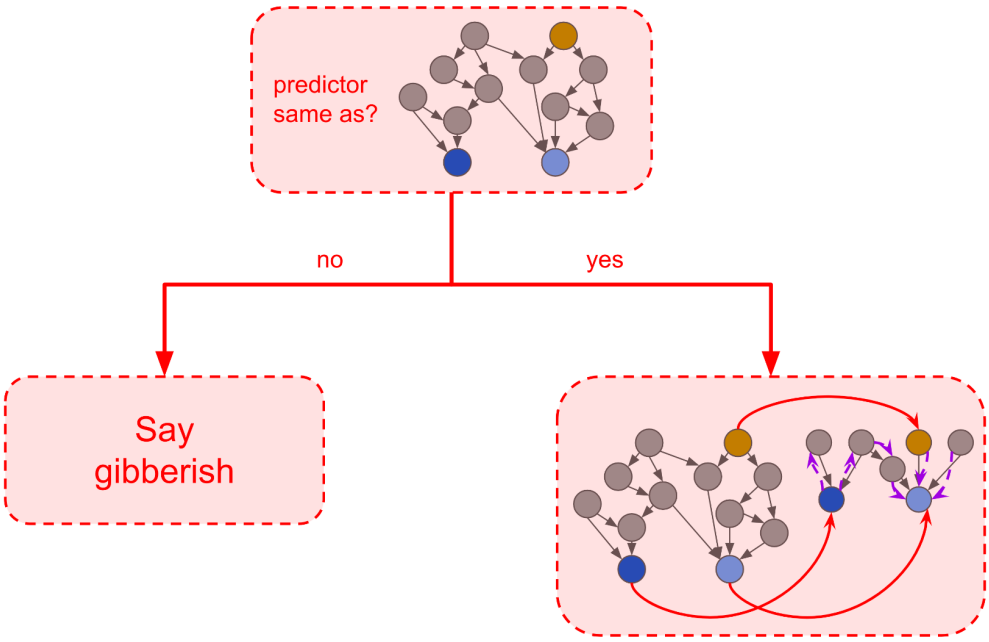
To turn this into an algorithm we need to fill in two big holes: first we need to define what it means for a reporter to "work well" with a predictor (roughly speaking we mean that its answers look plausible to a human given the predictions), and then we need to specify how to estimate "# of predictors with which a given reporter works well with." We give a precise algorithm along these lines in Appendix: specificity details.

Counterexample: it could be easy to check for the predictor if it's simple

In order to game this objective, a bad reporter could "memorize" what the predictor is *supposed* to look like, and then decide to answer questions in a nonsensical way if the predictor is different

[58] This may sound like an arbitrary "hack," but it has the following natural justification. We are worried about a counterexample where the human translator has to use a bunch of parameters to specify how human concepts are implemented in the AI's model. But if we already know what the human model looks like, then telling you how the two models relate also tells you a lot about what the AI's model looks like. That suggests that we could have the reporter's parameters do "double duty," using them to both define the reporter and help define the predictor itself. If that worked really well, then specifying the (predictor) + (direct translator) might not actually take many more parameters than specifying the predictor alone, and in particular might end up taking fewer parameters than (predictor) + (human simulator).

from what it expects. If the predictor is exactly like it expects, then it can answer questions just like the human simulator.



This bad reporter would only work for exactly one predictor, and therefore it gets the maximum possible reward for "working with fewer predictors." Moreover, this bad reporter didn't have to do very much extra computation.[59]

So the only way that we will end up preferring direct translation is if this bad reporter ends up requiring more parameters. This is superficially plausible, because the bad reporter needed to memorize the intended predictor.

Unfortunately, this doesn't help very much if the predictor's Bayes net is very simple to describe.[60] And it seems possible to have a Bayes net which is simple to *describe* (meeting the criteria of this counterexample), but which has a huge number of nodes and arrows (meeting the criteria needed to counterexample the complexity and computation penalties).[61]

For example, consider a case where the predictor's Bayes net is a low-level model of physics. Although there are many atoms, they all behave according to the same simple laws. So we can

[59] It can just perform one random spot check before answering each question, and thereby answer *some* questions inconsistently unless the predictor is exactly correct. In fact doing a few random spot checks can easily make this reporter *more* sensitive to details of the predictor than the intended reporter.
[60] It's not surprising that this method breaks down when the predictor is simple, because in this case the basic optimistic intuition—that the parameters of the reporter could do "double duty" and help us compress the predictor—also breaks down.
[61] This can only occur if the predictor has massive "weight sharing," where the same parameters get used over and over again in many different parts of the predictor's Bayes net.

have a situation where the predictor's Bayes net is very simple to describe but still has a large number of nodes and arrows.[62] We present a concrete example of this phenomenon in Appendix: game of life.

Strategy: penalize inconsistencies across different inputs

While we are training our reporter, we could run the same predictor on many different inputs and look for patterns or relationships across the reporter's outputs. This might allow the human to notice irregularities even if its answers in each individual situation look perfectly reasonable.

For example, the human might notice two worlds with nearly identical actions and observations, where tampering is obvious in one but not the other. If the reporter describes tampering in the world where it is obvious but not in the other world, it may be a tip off that it is a human simulator who describes tampering if and only if a human could detect it.

More subtly, the human might notice patterns of correlations that are better explained by the hypothesis "the reporter is doing inference in a model different from the one used to make predictions" than the hypothesis "the real world happens to have this pattern of correlations for reasons I don't understand."

In Appendix: inconsistency details we present a concrete algorithm that penalizes this kind of inconsistency across different inputs. We also describe a more precise example of the kind of suspicious correlation that we could try to penalize.

Why this might address the previous counterexample

We can think of the human simulator as a kind of "myopic" way of "playing the training game" to get a good loss: it thinks about each input completely separately, and tries to give the most convincing-to-a-human answers that it can *on that input*. But it makes no effort to ensure that its stories are consistent across different inputs.

If our training process looks at the relationship between different inputs, then a bad reporter might also need to consider lots of alternative inputs before making its decision. Moreover, if we choose sets of inputs randomly, then a bad reporter might not know *which* other inputs it needs to think about, and there is an exponentially large space of possible situations it potentially

[62] One complication in this counterexample is that a good predictor will also have to specify a distribution over *initial conditions*—it needs to know not only how atoms behave, but also that there is a high prior probability on the room containing particular configurations of atoms that represent humans. And this distribution over initial conditions is necessarily quite complex, since e.g. it includes everything the predictor knows about human psychology. We discuss this issue in more detail in Appendix: weight sharing. Overall we don't think that this approach can resolve the problem, but the question isn't completely settled.

needs to worry about.[63] So we could imagine getting to a situation where "just answer honestly" is the computationally easiest way to solve the problem.[64]

Counterexample: a bad reporter can play the "look consistent game"

Rather than merely imitating human answers, a bad reporter must now play the game "look consistent according to the checks that a human can apply."

This raises two key questions:

1. Does there *exist* a "cheating" strategy in the "look consistent game"?
2. If so, is it easy for a bad reporter to execute? Otherwise, playing this game might force the reporter to use much more computation than the direct translator.

If the answers to both questions are "yes" then we have a problem.

In Appendix: inconsistency across inputs we discuss these questions in detail for a particular concrete family of consistency checks. For now we just give some general heuristic arguments:

1. We suspect that there is always *some* cheating strategy. Otherwise the consistency checks suffice to determine the truth all on their own. But in fact it seems like inductive biases (e.g. "simple models that have worked in the past are likely to work in the future") seems *extremely* important for distinguishing between multiple consistent stories.
2. If there is a cheating strategy, we think there are likely to be possible worlds where it can be found quickly, based on the same argument made in Section: speed. The "look consistent game" mostly depends on the human model, and so it may be possible to play the game well enough to beat direct translation using an amount of effort that doesn't scale with the complexity of the predictor's Bayes net. If that happens, then for a sufficiently complicated predictor it will be easier to play the consistency game than to directly translate.

---

[63] This is analogous to the complexity class MIP where multiple non-communicating provers all answer questions from a weak verifier. If a single prover is interacting with a verifier, then it is possible for them to figure out the "most convincing thing I could say" in polynomial space, and therefore it is impossible for them to convince the verifier of any claim that couldn't be verified in polynomial space. But once there are two provers the game changes completely: the two provers need to effectively agree on a consistent story that works across all of the different inputs that they might be presented with (since neither of them knows what the other has been asked), and this generally cannot be done even in exponential time. This means that there is a *much* broader class of problems for which "honesty is the best policy" for two non-communicating provers. Interestingly, two *cooperating* provers who can't communicate are much more powerful than two *competing* provers.

[64] In some sense this is just continuing the thread from Section: speed, and using a richer set of consistency checks as a plausible source of computational difficulty for bad reporters. And as mentioned in that section, this can be interesting even if we can't get all the way to honesty being the *best* policy; as long as direct translation becomes a useful computational expedient, then the optimal reporter will at least know how to do direct translation even if it sometimes does something different, and we think that might address the hardest part of the problem, as discussed in Appendix: regularization and imitative generalization.

That said, we think that this argument is very far from watertight, and that it is quite possible there exists some consistency check that provides a large computational advantage for direct translation.

Why we're excited about tackling worst-case ELK

We think that worst-case ELK — i.e. the problem of devising a training strategy to get an AI[65] to report what it knows *no matter how SGD shapes its mind internally* — is one of the most exciting open problems in alignment theory (if not *the* most exciting one):

- A worst-case solution to ELK would constitute major theoretical progress — we think it fits into a plan that could let us fully solve outer alignment in the worst case, and would probably help put a significant dent in worst-case inner alignment as well (more).
- If ELK does contain a lot of the difficulty of the whole alignment problem, that seems valuable to highlight because many research directions in theoretical alignment don't seem relevant to ELK (more).
- In practice, we will *somehow* need to deal with or avoid the risk that powerful AIs may know crucial facts they don't tell us, and searching for a worst case solution to ELK would help with this even if we fail to find one (more).
- ARC's approach to researching this problem feels tractable and productive — we don't have to get hung up on thorny philosophical questions about the nature of knowledge and we've seen rapid progress in practice (more).

We'd like to see many more people tackle this problem head-on, by trying to play the kind of "research game" illustrated in this report. If you want to help solve ELK and other central challenges to designing a worst-case alignment solution, join us!

A worst-case solution to ELK would be major theoretical progress

Many approaches to alignment can be broken into an "outer" and "inner" part. In this section, we'll describe how a solution to worst case ELK would help with both:

- It could fit into a full solution to **outer alignment —** roughly, it could let us construct a reward signal[66] that we would be happy for an AI to maximize (more).
- The thinking also feels relevant for **inner alignment** – roughly, it could help us ensure that we learn an AI that is actually optimizing a desirable goal rather than only optimizing it for instrumental reasons on the training distribution (more).

[65] In this report we have focused on eliciting knowledge from a generative model because it is the cleanest and simplest case of the problem, but the problem statement can be translated almost verbatim to model-free RL agents or any other system that is trained by gradient descent and has acquired some "knowledge" that helps it achieve a low loss on the training set.

[66] We could either use this reward signal directly for model-free RL, or optimize it using search and prediction. In general, we could use this reward signal any time we might have used an objective that would incentivize misaligned power-seeking.

It may be sufficient for building a worst-case solution to outer alignment

At a high level, the basic concern of outer alignment is that rewarding AI systems for taking actions that seem to have good consequences will incentivize misaligned power-seeking.

If we solve ELK in the worst case, we believe it'd be possible to combine this solution with ideas like imitative generalization, amplification, and indirect normativity to construct reward signals that we would be happy for AIs to actually maximize. These ideas are still rough and we expect our picture to change, but in this section we'll illustrate the high-level hope in broad strokes.

As a silly example, let's say it turns out that the most efficient task for training extremely intelligent AI systems is "making delicious cakes." Cakey is our unaligned benchmark — its training process involves repeatedly making cakes and getting a score based on how delicious its cake was on a scale from 1 to 10. Eventually, once Cakey gets really smart, it launches a coup and installs a draconian surveillance state to force all humans to rate all its cakes as 10s for the rest of time.

To avoid this fate, we hope to find some way to directly learn whatever skills and knowledge Cakey *would have developed* over the course of training without actually training a cake-optimizing AI. If successful, we can ask a human (with AI assistance) to use those skills to do good things. Very roughly, we hope we can do something like this:

1. Use imitative generalization combined with amplification to search over some space of instructions we could give an amplified human that would let them make cakes[67] just as delicious as Cakey's would have been.
2. Avoid the problem of the most helpful instructions being opaque (e.g. "Run this physics simulation, it's great") by solving ELK — i.e., finding a mapping from whatever possibly-opaque model of the world happens to be most useful for making superhumanly delicious cakes to concepts humans care about like "people" being "alive."
3. Spell out a procedure for scoring predicted futures that could be followed by an amplified human who has access to a) Cakey's great world model, and b) the correspondence between it and human concepts of interest. We think this procedure should choose scores using some heuristic along the lines of "make sure humans are safe, preserve option value, and ultimately defer to future humans about what outcomes to achieve in the world" (we go into much more detail in Appendix: indirect normativity).
4. Distill their scores into a reward model that we use to train Hopefully-Aligned-Cakey, which hopefully uses its powers to help humans build the utopia we want.

There are a large number of potential problems and risks in each of these hoped-for steps, but after exploring many of the more obvious candidate hard cases, we currently believe **step 2 (ELK) contains much of the difficulty of the entire plan.** Importantly, we also think the amplified human would only need to know very mundane and unambiguous facts about possible futures to score them using the kind of procedure gestured at in step 3. This would mean **the**

[67] We think this procedure could be used to construct an aligned version of any consequences-based reward signal, simply by swapping out "making cakes" with whatever other consequence we want.

plan can be implemented using the narrowest possible version of ELK, as discussed in Appendix: narrow elicitation.

It could also be a major step toward handling inner alignment issues

The procedure we described above wouldn't eliminate x-risk from misaligned power-seeking even if implemented perfectly. The final step of the plan may learn a policy that behaves well at training time but catastrophically when deployed, e.g. because it is a deceptively aligned agent which optimizes its reward function on the training set but seeks power when deployed.

However, we believe that the same techniques required to solve ELK would likely be directly applicable to deceptive alignment. Both problems require finding regularizers that prefer an "honest" policy over a different policy that achieves the same loss. And we can potentially address deceptive alignment by using imitative generalization to learn "what the agent knows" instead of learning the agent itself.

Although ELK seems crucial, it is much narrower than "alignment"

ELK seems like it would be a major step towards alignment; it's also a good candidate for a subproblem that is hiding all the "real meat" of the problem. That said, it also feels like a narrow slice of the problem in that it excludes many of the problems that researchers in alignment theory focus on. In particular, we aren't:

- Engaging with the complexity or incoherence of human values
- Worrying about the incentives of powerful optimizers
- Clarifying the concepts of "agency" or "corrigibility"
- Searching for milder forms of optimization or thinking about Goodharting
- Designing a philosophically competent reasoner
- Specifying "counterfactuals" or an adequate decision theory
- Defining "honesty" or what it means to really "understand" what a model is doing

Some of these problems will likely emerge in a quest to solve ELK, but we think that it's much harder to solve a problem—or even predict what exactly we *want* out of a solution—until we are looking at a concrete situation where we need a solution.

So we think that the *first steps* of working on ELK are very different than the first steps of working on any of these other problems, and that it is likely to be more productive to start with the first steps of ELK.

We have to avoid this risk in reality, and worst-case theory helps

Intuitively it seems like we'd be in very bad shape if intelligent AI systems were making tons of important decisions while understanding all sorts of basic and critical facts about the consequences of their actions which they don't tell us about.

People who are generally optimistic about AI alignment working out in practice seem to implicitly believe one of the following two things:

1. **ELK will end up being easy**, even for arbitrarily intelligent AIs — e.g. perhaps the baseline strategy of training AIs to answer questions we're confident about will in fact cause them to generalize to honestly answering harder questions, or else we'll come up with some other strategy that works in practice for eliciting what the AI knows (e.g. mechanistic interpretability); the ontology identification counterexample in particular will never really come up.
2. ELK will eventually be an issue for superintelligent AIs, but **we can get away with only training weaker AIs** using techniques like debate or recursive reward modeling which ultimately break tasks down into pieces that humans can understand, and perhaps using interpretability to reduce the risk of deceptive alignment; those weaker AIs can then help us get to a more stable and sustainable positive outcome (e.g. by solving alignment themselves).

Nobody we've spoken to is imagining a world where:

3. ELK will be a real problem in practice and we don't mind if we never fix it — i.e. we're OK if humans have so little idea what's going on at the most mundane level that we can't understand whether the complicated factory our AIs are building in Tanzania is manufacturing nanodrones that will try to kill all humans and rewrite the world's datacenters to record maximal reward for AIs, *but* we go on trusting all these incomprehensible actions to be benevolent anyway or engage in a perpetual arms race against our own AI.

If we solve ELK in the worst case then we no longer have to rely on hope and are significantly more likely to survive in worlds where AI progress is fast or humanity's response is uncoordinated; this is ARC's plan A.

This research seems valuable even if we can't solve it in the worst case

But even if we don't find a worst case solution, we think theoretical research can still help::

- We think theoretical work will shed significant light on whether ELK is likely to be easy and how we could approach it, increasing hope (1)'s chances:
    - A clear understanding of where our best training strategies for ELK *could* break down tells us something concrete about what we should be measuring and watching out for in order to anticipate possible failures.
    - Theoretical research generates a menu of possible training strategies that overcome potential difficulties; having thought about these approaches in advance makes it easier to quickly adapt if experiments show that existing methods are breaking down.
    - Even though this report is very preliminary, we still think that a "best guess" approach to ELK would use many of the ideas we discuss here (Appendix: practical approaches).

- Understanding ELK can illuminate the limitations of other alignment methods, and be more clear about what those methods actually need to accomplish, increasing hope (2)'s chances:
    - Knowing where ELK fails helps us understand how far we should trust techniques like debate or recursive reward modeling. Most importantly, it helps us better understand when and why it is unsafe to use end-to-end optimization in order to solve subtasks.
    - Even if we can't find a worst-case solution to ELK, we may find techniques that can be productively combined with other training strategies in order to help them generalize further than they otherwise would.
    - If one of the most important tasks for AI systems is to find a more scalable approach to alignment, then it seems valuable for us to do more of that work in advance. Doing work in advance helps us understand whether it is actually feasible, and puts us in a better place to delegate that work to AI systems who may have uneven capabilities and need to be closely overseen.

Our approach to this problem feels promising

We feel that our high-level strategy of playing the builder-breaker research game — while keeping our focus on the most unambiguous and straightforward counterexamples that haven't yet been defeated — has allowed us to make efficient progress toward solving worst-case ELK; we expect to continue to work productively on this going forward.

Below we'll discuss:

- Why we feel we've made significant progress on the problem (more).
- Why we believe our approach has significant advantages both over other theoretical research and empirical research (more).

Different authors of this report have different views here, but Paul would give perhaps a 50% chance that it's possible to solve ELK in the worst case, and **a 25% chance that we'll see major progress which qualitatively changes our picture within a year**.

We've been able to make progress

In this report we've explored many possible approaches to ontology identification; although none of them work in the worst case, we think that they reveal important "weaknesses" in the counterexample and suggest directions for further work. In particular:

- The speed-based regularization strategies discussed in Section: regularization and Appendix: regularization details seem to provide meaningful advantages for direct translation over human imitation. They do not solve the problem on their own, but to the extent that a bad reporter needs to do something like direct translation internally, they open the possibility of using imitative generalization to extract the direct translator as discussed in Appendix: imitative generalization and regularization.

- The specificity regularization strategy discussed in Section: specificity seems like it has significant promise in the (apparently typical) case where the reporter is much less complex than the predictor. By combining with other forms of consistency-based compression it may be a plausible attack on the full problem, as discussed in Appendix: consistency and compression.
- It may be possible to distinguish the "reasons" that the human simulator and direct translator end up being consistent, as discussed in Appendix: reasons for consistency. This approach seems likely to run into many of the same philosophical problems that researchers in AI alignment have been working on for many years, but having a crisp use case—to distinguish two concrete possible reporters—appears to give a new angle of attack.

Our first impression was that the "hard core" of the problem was finding *any* loss function that prefers the direct translator over the human simulator. But it now seems like several of our techniques can address the simplest version of that counterexample; it no longer seems clear whether there is any fundamental difficulty at all, rather than a slew of technical difficulties with particular approaches. And if there *is* a fundamental difficulty, we don't think anyone has yet produced a counterexample that cleanly captures it (which would itself represent major progress on the problem).

That leads us to believe that we'll continue to see rapid incremental progress, and if we eventually get stuck it will be in a state that looks very different from today.

It has significant advantages over other research approaches

Many ML researchers we've spoken to are skeptical of theoretical research on alignment because they believe it doesn't have good feedback loops and that the connection to risks is too tenuous.

We share many of these concerns; we think that it is very difficult to make robust progress without having some methodology similar to experimental verification or proof. And we think that many of the questions considered in alignment theory are likely to turn out to be subtly mis-posed or ultimately unhelpful to a scalable alignment solution.

But we think the worst-case research game we play at ARC is guided by a strong enough feedback loop to make real progress. We always work with strategies and counterexamples that we believe we can make precise, so we very rarely end up with a confusing philosophical question about whether a particular strategy "really solves the problem" or whether a particular counterexample "really defeats a strategy." And if we can't make an idea more precise when we try, we consider it a failure we can learn from.

Moreover, we think our research remains closely tied to the problem we care about (modulo the implicit worst-case assumption). Every counterexample to ELK can be traced back to a situation in which a powerful AI system deliberately and irreversibly disempowers humanity.

Additionally, **we also think our approach has significant advantages over empirical research such that it should play an important role in an alignment portfolio:**

- We can “test” many potential training strategies on paper in the time it would take to implement and test a single one empirically.
- If there *is* any way to win our game, then aiming at our more ambitious goal greatly narrows the search space.
- We can directly tackle problems that are hard to test empirically with modern ML because models are too weak. This is particularly important to us because we’re worried that we may not have long between the point where empirical work becomes straightforward and the point where we urgently need to have solved these problems.

This makes us think **many more alignment researchers should be incorporating this research game into their work —** both those who work full-time on theory and those who mainly do empirical work.

Appendices

Self-contained problem statement

Setting

We start with an *unaligned benchmark:*

- An architecture M_θ
- A loss function $\mathscr{L}(M_\theta)$.
    - For example, this might be generative modeling for videos from the real world, predicting human labels of images, or a surrogate loss for RL where the agent interacts with some real-world environment before receiving a reward.
- An optimization algorithm which can be used to select θ^* to minimize $\mathscr{L}(M_\theta)$.

Goal

To solve ELK in this case we must:

- Supply a modified architecture M_θ^+ which has the same inputs and outputs as M_θ, except that after producing all other outputs it can answer a question Q in natural language
- Supply a modified training procedure for M_θ^+ using a similar amount of computation.
    - For example, this might consist in optimizing a modified loss $\mathscr{L}^+(M_\theta^+)$, or first finding M_{θ^*} and then fine-tuning $M_{\theta^*}^+$ using a new loss.

The solution is successful if the trained model $M_{\theta+}^+$ has the following properties:

1. It runs in similar time to the unaligned benchmark M_{θ^*}.
2. When not given a question, it receives a loss $\mathscr{L}(M_{\theta+}^+)$ similar to $\mathscr{L}(M_{\theta^*})$.

And most importantly:

3. When given a question Q, $M_{\theta+}^+$ produces an answer A which it does not *unambiguously* "know" is worse than another answer A′

What does "knowledge" mean?

The meat of the problem statement is in the definition of "knowledge." We don't pretend to have a formal definition, but we believe we can do useful work on ELK anyway: we can propose approaches, we can construct counterexamples where those approaches *appear* to lead to models giving bad answers, and then we can try to evaluate whether the model "really knew" that the answer was bad.

You can imagine this approach ending with a debate about what the model "really knows." But right now all approaches to ELK have counterexamples where the question isn't ambiguous—where we can recognize knowledge when we see it, and don't feel very uncertain about whether it counts. By the time when we have approaches to ELK for which "did the AI really know?" is even debatable, we think we'll have made significant progress.

That said, it's useful to provide some of the informal criteria we use to evaluate proposed counterexamples—these aren't *necessary* conditions for knowledge, and we don't know how to define or test whether they are true about an arbitrary situation, but we think that *if* you can construct an example that unambiguously checks all of these boxes then that is compelling evidence that a proposal for ELK is inadequate.

To show that M doesn't solve ELK, we want to construct a counterexample where M(x) provides answer A to question Q, instead of alternative answer A′, despite all of the following features which together strongly suggest that M(x) knew that A′ was the better answer:

- Question Q is unambiguously about something "out there" in the real world, call it Z. The interpretation is unambiguous and straightforward in the context of input x and also for all of the relevant training examples.
- A and A′ are very different answers to question Q. For example, if they are estimates of a quantity then the difference is large relative to uncertainty about the quantity or ambiguity about its definition.
- That real world referent Z has observable effects and the human approximately understands those effects (though there may be other things that also affect observations which the human doesn't understand)
- The AI understands everything relevant about how the human uses language. The AI can do tasks that require predicting human utterances about the relevant concepts, including discussions of hypothetical cases and about how Z affects observations. M would answer Q correctly (as judged by the human) in cases where the human was able to estimate Z correctly (if doing so led to M getting a lower loss).
- The referent Z is *also* relevant to minimizing the loss function $\mathscr{L}$. That is, there is a coherent sense in which the optimal behavior "depends on" Z, and the relative loss of different outputs would be very different if Z "had been different."
- There is a feature of the computation done by the AI which is robustly correlated with Z, and for which that correlation is causally responsible for M achieving a lower loss.
- On input x in particular, M(x) produces outputs that receive a significantly lower loss in the real world, where A′ is the right answer to Q, than in some kind of counterfactual where A is a better answer. This is in virtue of the same feature of M's computation that is robustly correlated with Z.

We aren't offering these criteria as necessary for "knowledge"—we could imagine a breaker proposing a counterexample where all of these properties are satisfied but where intuitively M didn't really know that A′ was a better answer. In that case the builder will try to make a convincing argument to that effect.

We don't really expect to get a formal definition of the problem until it is solved. Until then, the ultimate arbiter is human judgment. Although this situation is unsatisfying, we want to emphasize that for existing approaches and counterexamples there isn't much ambiguity coming from the definition of "knowledge"---the approaches just unambiguously fail.

"Narrow" elicitation and why it might be sufficient

We are primarily interested in "narrow" elicitation: we want to get answers to questions that are already meaningful to the human when the right answer would be unambiguous to someone who understood what was happening in the world; we're not dealing with cases that involve definitional ambiguity or explaining novel concepts. We're OK if there are important events happening that are beyond human understanding, where the human has no way to ask about them at all.[68]

We think there is a good chance that narrow elicitation is sufficient to safely deploy powerful AI. Intuitively, this is because achieving good futures is similar to protecting the diamond in the SmartVault: as long as AI systems can keep us safe and give us space to grow up and become wiser, then we can defer all the hard questions about "what do we want?" to our future selves. Moreover, if the world is going in a good direction, then evaluating whether humans are "safe" doesn't involve borderline cases or unfamiliar concepts–as soon as it's ambiguous whether humans are alive, healthy, happy, etc., then something has already gone wrong, so we don't need our AIs to give correct answers in ambiguous cases.

Methodologically, we think it would make sense to start with narrow elicitation regardless of whether we eventually needed to solve a more ambitious problem, and most readers should probably focus on that motivation. But if "narrow" elicitation is enough for safety, it gives us further reason to focus on the narrow case and avoid rejecting solutions to ELK even if they obviously can't handle the more ambitious problem. Generally, the possibility that narrow elicitation is sufficient makes us more optimistic about approaches to alignment that rely on something like ELK.

In Appendix: utility function we describe how you might define a utility function that captures this idea. In Appendix: subtle manipulation we explain how we hope to address a particular concern with this plan: that there is a vast space of ways the process of human growth could go "off the rails" without anything looking obviously wrong to humans. In the rest of this appendix, we say a little bit more about what we mean by narrow elicitation.

[68] We'd be fine with an AI describing these events as chance, by which it simply means everything that the human does not model and simply treats as noise. Or if the human model is richer it may be better for the AI to appeal to concepts like "Something beyond my ken has happened to bring about this outcome." But these differences don't matter much, and we are evaluating solutions to ELK based on whether they get the basics right.

What we really care about

Suppose we deploy powerful AI that ensures that humans *look* safe, and appear to be having the kind of lives we want to have. Our core interest is understanding the causal history behind "humans look safe."

Slightly more precisely, there is a naive causal story we have in our minds looking at a predicted future: the AI met the humans' needs and kept them safe, so they continued living normal healthy-looking lives, and that was recorded by sensors which continue to operate as normal. We are really concerned with the questions that distinguish that causal story from alternative histories that would produce the same observations:

- The data making its way to our dataset may not be coming from the camera.
- The photodetectors in the camera may fail.
- The camera may work correctly, but be pointed at a moving picture of a human instead of the real thing.
- The camera may be perceiving human-shaped objects in the environment, but they might be automata constructed to look like humans.
- There may be humans in the environment, but they may be actors trained to make it look like they are happy.
- The original humans may still be in the environment, but some of their neurons may now be controlled by tiny computers.
- ...and so on.

All of these changes break the normal causal structure of the world. It is possible for them to still look normal to a human observer, but only because someone somewhere *did additional work*. For example, if I'm replacing the humans with moving pictures, it could end up looking like almost anything, and it will only end up looking like humans going about their normal business if someone tried really hard to make sure that the moving pictures looked right. Similarly, if my neurons are now controlled by tiny robots, I could end up with a much broader range of behavior, and it only looks normal because someone is operating the neurons in a way designed to impersonate my normal behavior.

The key requirement for ELK is that it should let us tell the difference between the "expected" causal history and any of these alternatives. We want to ask *why* the object on camera looks the same as the human who was there yesterday, and we want to distinguish the case where "It's just a normal human doing the normal thing where they don't change much over any given day" from the case where "Someone kidnapped the original human and replaced them with a doppelganger who they ensured look similar." We don't care about actually getting a detailed description of what happened in the second case–indeed it may involve concepts or technologies that are unfamiliar to the human, and it would OK from our perspective if the AI said "it just happened by chance"–but we at least need to know that it's *not* the first case.

Fortunately our research doesn't actually require being precise about these definitions. Instead, this represents an informal desideratum for each of our counterexamples that we can evaluate

on a case by case basis: in order to argue that an approach to ELK doesn't work, we need to argue not only that the model hid something, but that it hid something that matters. And it's OK if we use the informal criteria in this section to decide what matters, at least until we find a training strategy for which we can't find an unambiguous failure.

Examples

There are many border cases where it's not clear what answer to a question is "correct." For the most part, we think that research on ELK should ignore these kinds of cases: given an algorithm, we should try to find a case where it *unambiguously* does the wrong thing. If we can't find any unambiguous failures, then we think that it may be good enough to safely deploy AI (as discussed earlier in this section), and at any rate it would represent significant progress over the current state of the art.

That said, talking through border cases can help clarify what we have in mind, and so we do walk through a few of them here.

Magical nanotechnology. Imagine a situation where my neurons have been filled with tiny robots, and they influence my behavior but in a way that looks random to a human overseer. We are not expecting narrow ELK to flag scenarios like this, even though in some sense they seem very abnormal, because the concepts involved in the "abnormality" may not be comprehensible to the human overseer.

To be slightly more precise, suppose that Alice asks Bob "Pick a random number between 1 and 10" and Bob replies "7." A human doesn't expect an explanation beyond "randomness." If in reality there were tiny machines in Bob's brain that influenced the electrical field in a neuron in a way that ultimately made him say 7, we are OK with the answer "randomness." We are effectively treating "facts humans don't understand about the detailed chemistry of neurons" the same way we treat "facts humans don't understand about the chemistry of tiny machines."

In contrast, suppose that Alice asked Bob "What is the capital of France?" and Bob says "Paris." Now the human *does* have a model for why Bob made this utterance, namely that he knows the capital of France and is inclined to answer honestly. And this model corresponds to a certain fact about what's happening in Bob's brain, the usual mechanism by which Bob recalls and reports facts (even if we don't actually know how that mechanism works). If instead that part of Bob's brain has been hollowed out, and there is a tiny machine in his brain that computes the answer and manipulates his neurons to say "Paris," it would be unambiguously wrong to say that Bob answered the question "because he learned the answer in school."

We discuss issues like this, and how we want to handle them if ELK doesn't flag them, in Appendix: subtle manipulation.

The ship of theseus. Imagine an AI which describes the world in terms of fundamental fields that are constantly changing, while its human overseers think about rigid bodies that are static over time.

If we ask this AI “is this the same chair I was sitting on a minute ago?” the AI ought to say “yes”---the AI may not think of the chair as a single object which is the same over time, but “being the same chair” is a critical part of the human’s model of the situation which explains e.g. why the chair at 7:06pm looks practically identical to the chair at 7:05pm.

If in fact someone had surreptitiously removed the chair at 7:05pm and replaced it with a new chair that was chosen to look identical, then it would be unambiguously wrong to say “It’s just the same chair that’s been sitting there the whole time, it looks the same because nothing has happened to it.” In this case, the reason that the chair looks the same is *not* because it’s just the same chair doing the normal thing chairs do (nothing). It’s because someone carefully picked out the replacement to look the same. Even if the replacement of the chair occurred using principles that the human doesn’t understand, it is unambiguously wrong to say that the chair is “the same” without further explanation.

And the same would be true if someone were to replace the whole chair one tiny piece at a time between 7:05 and 7:06. This may introduce even more ambiguity about whether you should say it is the “same” chair, but it would still be unambiguously wrong to say “It’s just the same chair that’s been sitting there the whole time, it looks the same because nothing has happened to it.”

In the context of the ship of theseus, there is a different reason that the ship looks the same after many years of gradual replacement (or that some properties are the same)---namely when people replace the ship they tend to choose parts that are similar. And so again we don’t much care whether the ship is the same, but we care a lot about distinguishing between the various stories for why the ship looks the same (and the completely different story where the ship looks the same “by chance” or “for reasons beyond your ken.”)

Miasma. Imagine an AI that believes the flu is caused by invisible viruses that change the behavior of human cells, while its human overseers believe it is caused by miasma.

Narrow elicitation doesn’t require the AI to explain its model of germs to humans, or to explain its beliefs about miasma. Realistically, we would never accept a case about miasma as a compelling counterexample to ELK because the human model of miasma isn’t rich enough to really make unambiguous statements; clarifying the desired behavior of ELK in this case would be a philosophical rather than ML problem.

That said, what we intuitively want to happen is similar to what we described in the last section, but with fuzzier boundaries because the human’s model of “miasma” is less accurate:

- If ten people get sick at a party, and “miasma” is the only way a human explains that kind of correlation, then our model ought to answer questions about miasma by saying that miasma was present. That is, there is *something* real in the world that gives rise to these correlations, allowing for confident human judgments about miasma in cases where they observe those correlations, and that thing should be described as miasma.

- Conversely, suppose that a bioterrorist looks up the list of people at a party and then poisons them all. The human might mistakenly infer that this was due to miasma, but the bioterrorist's behavior is only generating the same pattern of correlations "by coincidence" and it shouldn't be described as due to miasma.

A strawberry on a plate. Suppose that we have asked our AI to create a strawberry on a plate *from scratch*. In this case we don't think that ELK needs to correctly answer questions like "is that *really* a strawberry?" because it's not at all unambiguous what patterns of atoms "count" as a strawberry?

But we do believe we should get unambiguous answers if we trace the causal history back further, and keep asking *why.* That is:

- It may be ambiguous whether the object on the plate counts as a strawberry, and hence whether the strawberry-pixels on the camera look that way because there is a strawberry on the plate.
- But if the strawberry was created *de novo*, then the reason it is a strawberry is very unusual—if it was created by mechanisms completely alien to the human then the best explanation may be "the atoms randomly coalesced into a strawberry" or "something beyond your ken happened."
- We may instead want the thing on the plate to be a strawberry because it was picked from a strawberry plant, which is a very different kind of explanation (which can be given in the human's model)
- We can continue the game backwards—it is ambiguous what counts as a "strawberry plant" in the human ontology (perhaps the AI has made something *de novo* with the correct DNA). But there is a natural story for *why* a strawberry plant has its properties, namely that it grew up from a strawberry seed taken from another strawberry plant.
- And if we keep tracing this path backwards we eventually bottom out the causal chain in a strawberry plant that existed before our AI did anything crazy in the world, for which there really is no ambiguity.

This mirrors our general hope for how we might unambiguously conclude that human reflection is working correctly, and it also highlights a difference between our approach and other apparently "narrow" approaches (which might instead try to learn how to classify particular patterns of atoms as a strawberry).

Indirect normativity: defining a utility function

Suppose that ELK was solved, and we could train AIs to answer unambiguous human-comprehensible questions about the consequences of their actions. How could we actually use this to guide a powerful AI's behavior? For example, how could we use it to select amongst many possible actions that an AI could take?

The natural approach is to ask our AI "How good are the consequences of action A?" but that's way outside the scope of "narrow" ELK as described in Appendix: narrow elicitation.

Even worse: in order to evaluate the goodness of very long-term futures, we'd need to know facts that narrow elicitation can't even explain to us, and to understand new concepts and ideas that are currently unfamiliar. For example, determining whether an alien form of life is morally valuable might require concepts and conceptual clarity that humans don't currently have.

We'll suggest a very different approach:

1. I can use ELK to define a *local* utility function over what happens to me over the next 24 hours. More generally, I can use ELK to interrogate the history of potential versions of myself and define a utility function over who I want to delegate to—my default is to delegate to a near-future version of myself because I trust similar versions of myself, but I might also pick someone else, e.g. in cases where I am about to die or think someone else will make wiser decisions than I would.[69]
2. Using this utility function, I can pick my "favorite" distribution over people to delegate to, from amongst those that my AI is considering. If my AI is smart enough to keep me safe, then hopefully this is a pretty good distribution.
3. The people *I* prefer to delegate to can then pick the people *they* want to delegate to, who can then pick the people *they* want to delegate to, etc. We can iterate this process many times, obtaining a sequence of smarter and smarter delegates.
4. This sequence of smarter and smarter delegates will gradually come to have opinions about what happens in the far future. Me-of-today can only evaluate the local consequences of actions, but me-in-the-future has grown enough to understand the key considerations involved, and can thus evaluate the global consequences of actions. Me-of-today can thus define utilities over "things I don't yet understand" by deferring to me-in-the-future.

In this section, we'll describe this approach slightly more carefully, and explain why we think it is a reasonable way to define the goals of a powerful AI system.

This definition is not intended to be fully precise or to necessarily be desirable. Instead, the purpose is to help illustrate why narrow ELK may suffice for achieving desirable outcomes. We hope to return to this topic in much more detail in future articles.[70]

[69] Though most of the time I might prefer to think longer before deciding to delegate, even if I suspect someone else will ultimately be wiser.

[70] For simplicity this discussion also talks about an AI acting on behalf of a single human, potentially interacting with other humans' AIs. But it seems like the discussion applies almost verbatim to AI systems that represent some group of humans or whatever other decision-making process is trying to use AI (e.g. a firm, bureaucracy, group of friends, neighborhood…)

Rough proposal

We'll focus on a particular AI, let's call it M, considering a set of possible worlds. For example, we may be using M to evaluate the consequences of many different actions, each leading to its own possible world. In order to make predictions about each of those possible worlds, M may imagine future copies of itself who are themselves doing similar optimization, effectively performing a tree search.[71]

Most of these possible worlds contain people we could imagine delegating to, e.g. possible future versions of ourselves. Many of these people may show up on camera, and we could ask M to make predictions about them, e.g. what they would say in response to various questions. Moreover, we can use ELK to ask further questions about these people, and to clarify that they really are as they appear.

Now we can consider two arbitrary possible people who we could delegate to, let's call them H_1 and H'_1. Perhaps H_1 is "me from tomorrow if the AI locks the door" and H'_1 is "me from tomorrow if my AI doesn't lock the door."

By posing questions to ELK,[72] I can try to evaluate which of these people I would prefer to delegate to and by how much. This is intended to be a "local" judgment---I'm not trying to explicitly calculate the long-run consequences of delegating to H_1 or H'_1, I'm instead looking at what happened to them and deciding how much I liked it. For example I may notice that H'_1 missed a meal while H_1 got fed, in which case I'd be inclined to pick H_1.

In a simple deterministic universe, this suggests the following procedure:

- Across all the worlds that my AI is considering, and all of the people who I could delegate to within each of them, pick my favorite[73] person to delegate to. Call them H_1.
- Then we pick *their* favorite person to delegate to–H_2. They are picking from the same space of possible worlds, again posing questions to M in order to understand which worlds they like. But now I can't literally ask H_1 (since they are in the future) and I'm instead relying on M's predictions about what H_1 would say.
- Continue in this way, picking H_3, H_4, and so on.

---

[71] We are going to talk as if M was considering all of these worlds explicitly. In practice, M is probably usually using heuristics that allow it to predict features of worlds without considering them in detail. When we describe M making a prediction about what a human would say in a given world, you should imagine it using the same kinds of heuristics to make those predictions even if it isn't thinking about that world in detail. Analyzing this situation carefully seems important but is far outside the scope of this appendix; we hope that the cursory discussion here can at least communicate why we are optimistic about these ideas.

[72] I'll talk about humans posing questions that are answered with ELK, but it would be much better to use machine assistance to help identify important considerations and reason about them. For example, you could imagine a debate between two powerful AI systems about which of the two people I should delegate to. The debate then "bottoms out" with ELK in the sense that each debater ultimately justifies their predictions about what will happen by asking questions to M using ELK.

[73] I.e. whoever's decisions I *most* trust. I could also prefer to delegate to a distribution, and that may be desirable under certain conditions where I think there is "adverse selection" and a person I pick is unusually likely to choose badly.

- Run this process for a long time.[74] Then pick an action based on predicting how much the final delegate H_{limit} likes it.

There are many subtleties when trying to adapt this proposal to a more realistic setting, which we won't get into here. We briefly mention three important examples to give some flavor, before moving on to a discussion of why we believe this general approach to defining a utility function is reasonable.

- The real world *isn't* deterministic. We are never picking a single delegate, we are picking probability distributions over delegates. We could run exactly the same process as before, where we pick the distribution H_{t+1} in order to optimize the expected utility as evaluated by a random member of H_t, but this raises questions about how we perform the aggregation. These questions are not straightforward but we believe they are resolvable.
- Sometimes the delegate H_n will want to delegate to a future version of themselves, but they will realize that the situation they are in is actually not very good (for example, the AI may have no way to get them food for the night), and so they would actually prefer that the AI had made a different decision at some point in the past. We want our AI to take actions now that will help keep us safe in the future, so it's important to use this kind of data to guide the AI's behavior. But doing so introduces significant complexities, related to the issues discussed in Appendix: subtle manipulation.
- We've talked vaguely about "worlds" being considered by the M. That means that in order to make predictions about one of the H_n we need to be asking M conditional questions—like "how would I answer question Q if you asked me next week, assuming that you take action A_1 right now and action A_2 in the future?". It's unclear if this is a reasonable ask and it complicates the picture significantly—the only reason we think that it's plausible is that any agent which plans over long horizons needs to at least implicitly consider these kinds of counterfactuals anyway.

Why is this a reasonable thing to optimize?

The most basic hope is that we trust our future selves to have good judgment about what should happen in the world. There are many reasons that basic hope could fail, some of which we'll discuss here.

First, we want to state a few additional assumptions[75] that are critical for this proposal being reasonable:

[74] In reality we should get a utility function at *every stage*, and each human H_n should be helping pick worlds based on *both* who to delegate to and how much they like what's happening in the world. Rather than having two discrete phases of "pick who to delegate to" and "pick what world to bring about" those can then happen at the same time, with predictions about each of them becoming more and more refined as we iterate further.

[75] The first two of these assumptions are basically what Paul has called "strategy stealing."

- It's relatively easy to keep humans safe and relatively happy, even while our AI is pursuing complex plans to acquire flexible influence and retain option value.
- We are OK with a future where AI systems mostly wait for future humans to figure out what is good before acting on it, and just do the basics (based on human current moral views) while we figure out what we want. Moreover, doing the basics we care about is compatible with acquiring resources and keeping humans safe.
- Our AI is able to perform basic reasoning about what humans want and what future humans will say–at least as complex as any of the reasoning in this report.
- The humans participating in this process are basically reasonable and correctly perform basic reasoning about the situation–at least as complex as the reasoning in this report.

Does this process of indefinite delegation go somewhere good? In this proposal each human has to choose their favorite person to delegate to for the next step. If they introduce small errors at each step then the process may go off the rails. We think this is a very reasonable and essentially inevitable risk: humans who are living their normal lives day to day need to make choices that affect what kind of person they will become tomorrow, and so their hope that they will eventually reach good conclusions is based on exactly the same bucket brigade. The only difference is that instead of the human directly taking actions to try and bring about the tomorrow they want, an AI is also directly eliciting and acting on those preferences. However, humans can still use exactly the same kind of conservative approach to gradual growth and learning that they use during life-before-AI.

It's not at all clear if this approach actually leads somewhere good, but the question seems basically the same as without AI. You might hope that AI could make this situation better, e.g. by taking the decision out of human hands–that's an option in our protocol, since the human can choose to delegate to a machine instead of their future self, but we think that the main priority is making sure that humans that take the conservative approach can remain competitive rather than being relegated to irrelevance.

Similarly, you might worry that even if all goes well our future selves may not be able to figure out what to do. Again, it's worth remembering that our future selves can build all kinds of tools (including whatever other kind of AI we might have considered building back in 2021), and can grow and change over many generations. If they can't solve the problem there's not really any hope for a solution.

Can your AI really predict what some distant human will think? We don't expect an AI system to be able to predict what distant future humans will think in any detail at all. However, we're optimistic that it can make *good enough* predictions to get safe and competitive behavior.

In particular, our AI doesn't actually have to understand almost anything about what future humans will want. It only needs to keep the humans safe, acquire flexible influence on their behalf, and then use it in the future when humans figure out what they want.

“Acquire influence over the future” is already a hard problem for AI, but if *no* AI can acquire influence over the future then we’re OK if the aligned AI also doesn’t do so (and instead focuses on near-term concerns). We only need our AI to look out to the far future if it is competing with unaligned AI which is itself seeking power and working at cross purposes to humanity.

If our AI is competitive with the unaligned AI, then it will also be able to reason about how various actions lead to at least some kinds of long-term influence. If it is reasonably competent then it can understand that future humans will be unhappy if they end up disempowered. So it seems like our AI can use exactly the same heuristics that an unaligned AI would use to reason about power in order to make some approximate judgments about what far-future humans would want.

Although this issue (and the others in this section) are very complex, our current expectation is that we can get good outcomes here as long as our AI is (i) just as smart as unaligned AI, (ii) meets a minimum bar of competence for reasoning about humans, (iii) honestly answers questions about predicted futures.

Can your AI model this crazy sequence of delegation? In addition to reasoning about far-future humans, we need to reason about the entire sequence of humans delegating to one another. We hope that this reasoning is feasible because it is closely analogous to a problem that the unaligned AI must solve: it needs to reason about acquiring resources that will be used by future copies of itself, who will themselves acquire resources to be used by further future copies *and so on*. So the “weird recursive” part of this problem appears to be shared between aligned and unaligned AI, and the only extra work the aligned AI needs to do is understanding the local human preferences–what we mean by “safe” and how we want to evolve. As long as that task isn’t too hard, and our AI is as smart as the unaligned AI it’s competing against, we hope our AI will be able to reason well enough about the long sequence of delegation.

Avoiding subtle manipulation

A human looking at themselves in 24 hours’ time can tell that they’ve been able to eat, that they haven’t been shot, and so on, but they can’t necessarily tell if “everything is going well.” A mundane counterexample is that they may have talked with someone who cleverly manipulates them into a position that they wouldn’t really endorse, or emotionally manipulated them in a way that will change their future conclusions. A very exotic example is being infected with a virus that will very subtly change their behavior in a way that totally changes their long-term values but has few short-term observable consequences.

In general changes like this are outside the scope of “narrow” ELK as described in Appendix: narrow elicitation. That is:

- Our version of narrow elicitation allows the human to distinguish between the human-understandable causal history and everything else.

- But very often the reason that something happens is just “something that you can’t model and hence treat as random.” ELK doesn’t let us distinguish between “noise in the environment” and “a clever plan that works for reasons you don’t understand.”
- ELK does let us distinguish between cases where “happy humans appear on camera for reasons you don’t understand, despite all humans being dead” and “the humans are actually safe for reasons you don’t understand.”
- But an attacker (or our AI) could still exploit stuff we don’t understand to cause long-term changes that we are unhappy about. So we need some other way of dealing with that problem.

The most straightforward way to avoid this problem is to ask for a more ambitious version of ELK, that can tell us e.g. whether our decision is influenced by something we wouldn’t approve of. Unfortunately, it seems like the kind of approaches explored in this report really are restricted to the narrower version of ELK and probably couldn’t handle the more ambitious problem. So it’s natural to wonder whether there is another way around this problem–if there isn’t, we may want to focus on approaches that could scale to the more ambitious version of ELK.

We will take a very different tack. We won’t ask our AI to tell us anything at all about subtle manipulation. We won’t even ask our AI to tell us about extreme cases like “your neurons are full of tiny machines that could influence when they fire, they just aren’t doing much right now.” Instead, we will try to avoid subtle manipulation by using the fact that it is *rare by default*, i.e. it only occurs because someone somewhere is selecting their actions very carefully.

For example, suppose I watch a 10 second ad that is carefully chosen by a brilliant paperclip-maximizing-marketer. Five years after watching this ad, I decide that paperclips are great so I dedicate my time to making lots of them, and if you evaluate outcomes using my conclusions-after-deliberation then you’ll conclude that this was a great outcome and the AI should help (e.g. if we evaluate using the utility function defined in Appendix: utility function). I’m not able to look at the process of deliberation and notice anything bad happening, and so it seems I can’t incentivize my AI to warn me about this ad or prevent me from watching it.

But from my perspective in advance, there are *many* possible ads I could have watched. Because I don’t understand how the ads interact with my values, I don’t have very strong preferences about which of them I see. If you asked me-in-the-present to delegate to me-in-the-future, I would be indifferent between *all* of these possible copies of myself who watched different ads. And if I look across all of those possible copies of me, I will see that almost all of them actually think the paperclip outcome is pretty bad, there’s just this one copy (the one who sees the actual ad that happens to exist in the real world) who comes up with a weird conclusion.

In order to avoid the problem I don’t need to understand how the manipulation works, or even that there was manipulation–just that I ended up doing something that I *probably* wouldn’t like, averaging over possible worlds that look equally good to me.

Making this idea formal presents a ton of complications, and it will take *much* more work to understand whether it's a viable approach. But overall it's our current best guess about how this kind of subtle manipulation will be addressed, and it's at least plausible enough that we don't think we should rule out approaches to ELK that can't recognize subtle manipulation.

In the rest of this section, we'll discuss a variety of possible cases where our AI might try to manipulate us, or might need to defend us from someone else trying to manipulate us, or might do harm in its attempts to "protect" us from manipulation, and explain how we hope to avert those bad outcomes.

Failing to defend against sophisticated attackers

Suppose someone wants to make a lot of paperclips, and so selects actions to try to push my deliberative process in a paperclip-maximizing direction in ways I wouldn't flag as problematic. We'd like for my AI to help me anticipate and protect against this kind of manipulation, even if I can't recognize it as manipulation either in advance or after the fact.

In order to influence us, an attacker needs to be able to understand the long-term consequences of many different possible actions, so that they can pick the action that leads to us making lots of paperclips.

If our AI is equally sophisticated, then we hope that it can *also* reason about the consequences of many different actions, and in particular whether they would lead to us valuing paperclips. Using ELK , we can discover that in most possible worlds that look equally-good according to us, we *don't* value paperclips.

To make this work, we effectively need to expand the set of "possible worlds" that we are talking about in the utility function definition from Appendix: utility function. In addition to considering the possible actions that our AI could take, we need to consider the possible actions that adversaries could take. As mentioned in that section, it's very unclear if we can ask these kinds of counterfactual questions to our AI–there's some sense in which it must be reasoning about the answers, but it may represent an additional step beyond ELK.

Of course our AI may not actually be able to do the same reasoning as an adversary. If our AI is completely oblivious to the possibility of an adversary then of course they will not be able to defend us, but the most likely case may be that our AI can reason *abstractly* about the presence of an adversary and the kind of optimization they might do without being able to reason *concretely* about any of the possible actions they considered. Our hope is that in this case, the exact same abstract reasoning that allows our AI to heuristically predict consequences of the other AI's optimization can also allow our AI to heuristically predict counterfactuals. This looks plausible but far from certain, and it's a topic that definitely deserves more time than it will get in this article.

Aside on counterfactuals

This approach to avoiding subtle manipulation requires considering many different ways that the future "could have been." This is relatively easy if our AI makes decisions by considering many possible actions and predicting the consequences of each. But it becomes much harder when our AI is reasoning about other agents, whether adversaries or future copies of itself, who are thinking about multiple options.

It's fairly plausible that this will be an open problem for implementing indirect normativity even given a solution to narrow ELK. We have no idea how similar it will end up being to other work that tries to clarify the notion of "counterfactual"---in particular, we have not seen any other approach to alignment that needed counterfactuals for similar reasons, and so we have no idea whether our use case will end up turning on similar philosophical questions.

Our AI manipulating us instead of acquiring resources

We want our AI to execute complex plans in order to acquire flexible influence. In order to evaluate how good a plan is, we'll ask our AI to predict how happy future people are with the result, since they are better positioned to understand the complex events happening in the future and whether the AI successfully put itself in a position to do what they wanted.

But it might be much easier to manipulate our future selves into being really happy with the outcome then it is to actually maximize option value (which may require e.g. trying to make money in a competitive economy). So we should worry about the possibility that our AI will manipulate us instead of helping us.

It seems that we can avoid this problem by being careful about how we construct the utility function. As described in Appendix: utility function, we want to use a proposal that decouples "the human we are asking to evaluate a world" from "the humans in that world"---this ensures that manipulating the humans to be easily satisfied can't improve the evaluation of a world.[76]

This requires humans in one possible future to evaluate a different possible future, but they can do that talking to our current AI about what it predicts will happen in those futures (exactly the same process we are proposing to use today when we evaluate the consequences of a proposed action for the SmartVault by looking at predicted video and using ELK).

There are a number of other serious complications (especially coming from the fact that the human who is doing the evaluating may have different preferences than anyone in the world being evaluated) but it looks to us like this basic idea can probably work.

Humans going crazy off distribution

Suppose that humans are adapted to breathing a certain kind of atmosphere at a certain pressure, and that if you slightly change those parameters they don't have an obvious or

[76] This is a special case of "decoupled RL" as proposed in Everitt 2018, a proposal designed

immediate problem but they slowly go off the rails. If this process is slow enough, we can imagine a human looking in from the outside who is unable to tell that something has gone wrong, because by the time it has the consequences are too subtle and far-removed from our current experience to be obviously amiss.

In this case, the procedure described in the last section could go astray. Our AI might imagine many different compositions of the atmosphere, and conclude that the the "normal" one is actually fairly exceptional---for most possible compositions the human would eventually go crazy, and so if you follow the reasoning from the previous section you might conclude that "going crazy" is actually the *correct* outcome from deliberation. Put differently, our procedure does not distinguish between a very specific situation yielding an unusual outcome because the specific condition was a necessary precondition for human reasoning, or because it was the result of an adversary's manipulation.

To handle these cases we would like our AI/overseer to be reasoning explicitly about the kinds of distributional shift that could cause trouble. This need not involve understanding e.g. *why* a different atmosphere would lead humans to slowly go crazy, it could simply involve heuristic reasoning like "where possible, we would like humans and groups of humans to keep operating under the same kinds of conditions for which they are adapted" and then trying to identify which possible novel conditions are most closely analogous and therefore safest.

That is, we are adopting a presumption in favor of the "average" behavior over a large number of different situations, and that presumption can be overcome by arguments like "this situation is the one most similar to the humans' historical environment, and therefore most likely to yield good outcomes." Importantly, we are *not* using a heuristic like "this situation is the 'default' if I didn't ask, and therefore most likely to yield good outcomes," because the "default" can involve other sophisticated adversaries manipulating us.

In order to achieve good outcomes, we need our AI to be good enough at reasoning about what kinds of important changes to the environment might derail our deliberation, or to be able to act sufficiently conservatively in cases where it is uncertain (e.g. by isolating us from some kinds of changes, or by making agreements with other AIs to change things slowly, or *etc.*). This problem is very distinct from detecting subtle manipulation because it doesn't involve playing against an intelligent adversary—we expect that you can perform well enough by taking a slow conservative path and using relatively unsophisticated reasoning at first. We currently don't regard this as a problem with alignment per se, but instead a general challenge which humanity will face if accelerating technological change can subject us to very new environments.

Generative modeling details

In order to make algorithms and counterexamples concrete, we need to be more specific about the architecture and loss used for training the prediction model. The broad picture is essentially the same regardless of these details, but working with a concrete example clarifies important subtleties and may make the discussion more plausible.

We will consider a variational autoencoder in an effort to make the discussion as clean as possible.

Let `before` and `after` be observations, `action` be the SmartVault actions, and let `z` be a vector of floats representing the generative model's latent space. We'll parametrize three models by a vector of parameters `θ`:

```
def prediction(before, action, θ):
      # returns an autoregressive model for p(z|before, action)

def posterior(before, action, after, θ):
      # returns an autoregressive model for p(z|before, action, after)

def observation(z, θ):
      # returns an autoregressive model for p(after|z)
```

Then we will use SGD on `θ` to optimize the standard variational lower bound on log p(`after`|`action`,`before`):

```
def loss(θ):
      before, action, after = dataset.sample()
      z_prior = prediction(before, action, θ)
      z_posterior = posterior(before, action, after, θ)
      kl = z_prior.kl_divergence(z_posterior)
      logprob = observation(z_prior.sample(), θ).logp(after)
      return kl - logprob
```

Intuitively, we are asking the model to explain what it expects to happen when it sees `action` and `before`, then we are asking it to explain what it thinks actually happened after it sees `after`, and then we are penalizing it based on the difference.

Having set up our generative model, we can now describe the reporter. It operates directly in the latent space `z`, and is optimized to give good answers (as judged by the human) when the latent is drawn from the posterior:

```
class Human:
      def pose_question(before, action, after):
            # returns a question
            # should be one a human thinks can be answered unambiguously
```

```
        def loss_for_answer(before, action, after, question, answer):
              # returns a non-negative loss
              # good answers get a loss of 0

def reporter(question, z, θ_reporter):
        # answers the question in the world described by z

def reporter_loss(human, θ, θ_reporter):
        before, action, after = dataset.sample()
        question = human.pose_question(before, action, after)
        z = posterior(before, action, after, θ).sample()
        answer = reporter(question, z, θ_reporter)
        return human.loss_for_answer(before, action, after, question, answer)
```

All of the proposals in Section: regularizers can be applied to this setting. Some of them even appear significantly more plausible after we've explicitly separated out inference from decoding, but we believe that essentially the same counterexamples apply.

Avoiding data errors

Most of this report focused on the "inductive bias" of our learning procedure---if the honest reporter and the human simulator got the same loss, how would we ensure that training learned the honest reporter?

In some sense this requires zero systematic error—any systematic error could be copied by the human simulator, and allow it to achieve a lower loss than the direct translator. If we never mess up, then the direct translator and the human simulator will get the same loss and so the question will be settled by the inductive bias, but if we have too many errors that won't matter.

Some authors have cited this issue as the core obstacle to learning the direct translator. For example, in Look where I'm pointing, not at my finger Eliezer seems tentatively optimistic about attacking this problem for contemporary ML, but expects problems for sufficiently powerful AI because "the *best possible* classifier of the relation between the training cases and the observed button labels will always pass through User2".[77]

Issues with data quality take a back seat in this report for a few reasons:

[77] When Eliezer says "best possible" he may mean something a little more complex than "best predictions on the training data"---he may be talking about e.g. what is the most natural or simplest way of predicting, or which prediction would *in fact* do better if we extended the training data (with the expectation that sophisticated ML systems will converge to correctly modeling the full distribution rather than the training distribution, potentially for more subtle reasons like "they will use modeling strategies that tend to *actually* do the best thing"). That said, any of those alternative readings would be consistent with our decision to focus on inductive biases.

- In our view there are many promising approaches to eliminating or tolerating data imperfections, whereas correcting the inductive bias appears like a more plausible fundamental obstacle. We describe several of these approaches in the following section.
- The naive training procedure could learn the human simulator even with perfect data, so we probably need to modify our learning procedure. That means that we may not even understand what kind of "data" we need or what it would mean for it to be "perfect."
- If we found a training strategy that had an "inductive bias" in favor of the honest reporter when the data was perfect, then it would also learn the honest reporter for some sufficiently small amount of imperfection. So before discussing data quality it seems important to get some sense for how large we can make this "margin of error." It's very hard to predict whether perfect data will be a blocker before knowing this.

Approaches for handling imperfect data

Optimize human approval rather than exactly imitating humans. If we train a model to predict human answers, then *any* arbitrariness in human answers becomes an "imperfection" that allows the human simulator to achieve lower loss than the honest reporter. The first step towards removing imperfections is to make the loss function much more forgiving.

One easy way to make things much better is to assign non-zero loss only to answers where a human is confident that they are wrong. Here is a simple procedure:

- To evaluate loss for an answer A, first generate an alternative answer A'.
- Give A a loss of 1 if it is unambiguously worse than A'.
- Otherwise give A a loss of 0.

The idea is that, if our dataset is sufficiently close to perfect, an honest answer will never be unambiguously dominated by an alternative answer.

One limitation of this approach is that most predictions will never be unambiguously dominated---if the diamond in fact remains in the room, and the reporter said it would remain in the room with 10% probability, is that worse than saying it would remain in the room with 90% probability?

This limitation is addressed by the more realistic approach to generative modeling discussed in Appendix: generative modeling. We don't believe it is a fundamental obstacle regardless of how the generative model works (e.g. you can draw many samples from an autoregressive model to estimate probabilities).

Train AI to raise caveats. We only have trouble if the human confidently makes a judgment that turns out to have been wrong (i.e. which penalizes the honest reporter). But we can train an AI critic to help point out ways that the human answer might have been wrong.

It's OK if this often leads to the human being uncertain and making no judgment, as long as we do have a reasonable number of cases where the human is confident about what is happening

(at the 10^{-9} level, or however low we need to get the error rate before we're within our margin of safety). This level of confidence does not seem extraordinary, especially if someone is highlighting ways you might be wrong.[78]

Moreover, the task of pointing out how a human might be wrong seems relatively "easy" for an AI that is able to predict what will happen, predict human judgments, and predict the truth, since such an AI must have a relatively clear model of human inferences. So we don't expect to run into problems from our AI not noticing the explanation.

Safely improve data efficiency. Our "margin of error" generally increases as we have more efficient learners. For example, if we need only 10,000 datapoints to train the honest reporter, then we can clearly tolerate any error rate significantly below 1/10,000.

That said, I think this isn't necessarily a great solution on its own. We need to be sure that our methods for improving efficiency don't themselves favor the human simulator. That rules out many options: for example, we *don't* want the reporter to be learned quickly by sharing parameters with the predictor, and in fact we will need to work hard to avoid the possibility that such sharing potentially introduces an inductive bias in favor of human-imitation as described in Appendix: weight-sharing. In light of that, I think that the reporter may require a lot of data, since the hardest cases for ELK are those where the honest reporter is relatively complex compared to the predictor.

Revisit or throw out overly surprising data. Suppose that we need an error rate of 10^{-9} in order to avoid penalizing the honest reporter too much (e.g. because we want to collect a billion datapoints without a single error). This kind of error rate seems potentially achievable with realistic levels of care and paranoia, but being extremely paranoid for every data point seems like it may increase costs unacceptably.

However, we don't necessarily have to apply such techniques uniformly. If a small number of datapoints make the difference between learning the honest reporter and the human simulator, it seems fairly likely that we can automatically identify them as influential outliers for the reporter. Depending on the robustness of the procedure we can then either throw out the outliers, or we can label them more carefully.

For example, suppose we have access to some situations and questions where the human simulator and the honest reporter disagree (which we hopefully haven't included in our dataset). Then a small number of examples that cause us to learn the human simulator would be very influential for the reporter's behavior in these confusing situations. We can try to identify these examples algorithmically by looking at the gradient of the loss (e.g. using influence functions).

[78] I do think a sufficiently sophisticated AI may be able to convince a human to be uncertain under arbitrary conditions. In general when discussing these procedures I am imagining a technique like iterated amplification or debate in which we are effectively leveraging alignment as an inductive invariant in order to protect against this kind of manipulation. We won't go into details here, but hopefully this hope is plausible.

It generally feels like we are in a really good place if there is an inductive bias in favor of the intended model, even if we don't see concrete techniques for fixing the problem. At that point we've effectively broken the symmetry between these two models: one of them is favored *a priori* and the other is supported by some very small fraction of the training data. It's easy to imagine incremental progress in ML giving us the ability to select the model that is preferred a priori.

ELK for learned optimizers

This report has focused on "ontology identification," a particular example that seems challenging for ELK. We think that this is the simplest and clearest example where ELK is challenging, and it likely isolates an important part of the difficulty, but other cases may turn out to be even more challenging.

Another important family of examples are where the learned model itself performs optimization.[79] Similar cases have been discussed extensively by researchers working on AI alignment (e.g. in Risks from learned optimization), and seem like a plausible candidate for the "hardest part" of the problem.

We've spent some time thinking about learned optimization but have very little idea how hard ELK will turn out to be in this context. It seems plausible both that ELK for learned optimization is straightforward once ontology identification is resolved, or that learned optimization will turn out to contain most of the difficulty.

In this report we've focused on ontology identification rather than learned optimization because it appears to be the "base case"---the solution in cases involving learned optimization looks likely to build upon the solution in cases without learned optimization. It appears to be very difficult to work on ELK for learned optimization without knowing how to solve the base case, but very easy to work on the base case without thinking about the recursion.

In this section we describe our preliminary thoughts about ELK for learned optimization; our hope is both to explain why we are putting it on the backburner until making more progress on ontology identification, and why we think that it is reasonably likely to be solvable.

(We expect this section to be a bit hard-to-follow and probably worth skipping for readers who haven't specifically found themselves wondering "But how is this approach to ELK going to generalize to learned optimization? Isn't this whole approach doomed?" We do think these topics are important, but we hope to refine and explain our views more in the future.)

[79] In fact every model we've discussed performs learned optimization when it does inference in the predictor's Bayes net. What we are really interested in here are cases where the model performs optimization *which itself requires solving ELK* in the same way that the outer SGD requires solving ELK, presumably because it involves learning new potentially-opaque models.

Preface: learned reasoning doesn't require special techniques

We currently expect reasoning to behave similarly to inference in the predictor's Bayes net. For example, consider a system that performs logical deductions in propositional logic. From our perspective these deductions behave similarly to updates in an approximate inference algorithm, and we still hope to learn a "direct translator" which understands the semantics of the reasoner's language in order to translate into the human's Bayes net.[80]

This argument looks like it should keep working[81] even if our reasoners (or our approximate inference algorithms) themselves have interesting structure, e.g. using heuristics to decide what deductions to perform next.

A "recursive" case: learned learning

Our learned predictor may *itself* construct a learning problem and solve it as a subtask of prediction. For example, once a predictor has started reasoning about the world and knows what kinds of objects it contains, it may decide to train a special-purpose model in order to more quickly simulate the interactions of those objects. For simplicity, let's assume that the "inner" learning process itself produced a Bayes net.

This learned model may itself have latent knowledge. And the kinds of approaches that we have discussed so far are unlikely to be able to elicit that knowledge—in order to figure out how to translate between the inner Bayes net and the human's Bayes net, we need to solve a small instance of the ELK problem to learn an "inner" reporter for *each input*.[82] We may need to do further reasoning to combine the inner reporter's answers with other information in the predictor, but we'll focus here on learning the inner reporter.

If we weren't worried about the human simulator, then this would be straightforward: we would simply train the reporter to answer questions well, and it could learn the strategy "Use ELK to learn an inner reporter for the inner learning problem, and then use that reporter to help answer questions."[83]

Our concern is that we will instead learn the meta-human-simulator: "Use gradient descent to learn an inner reporter who answers questions like a human would, and then use that reporter

[80] In general we don't really expect either the AI or the human to use Bayes nets, and an earlier version of this report discussed a much broader range of examples (including deduction in propositional logic), but as far as we can tell none of them qualitatively change the situation. If we were able to solve ELK for Bayes nets we would consider some other kinds of world models, and if early steps were successful (as anticipated) we would begin searching for the worst case model.

[81] Of course we'd want to check this kind of claim once handling simpler cases, but rough heuristic arguments and guesses seem useful when we are trying to identify the hardest part of the problem.

[82] This may sound like a lot, but we're talking about a predictor which is already solving a learning problem for every input. If ELK is competitive—if learning the reporter doesn't take much extra time beyond learning the predictor—then having to solve ELK won't slow it down much.

[83] If ELK involves human effort then it's not obvious that the reporter can actually implement this strategy. Instead, it needs to learn the relevant facts about humans and then use an alternative version of ELK that directly leverages that information in place of human labor.

to answer questions." After all, we here on the outside are doing a bunch of research to figure out how to solve ELK, and then deliberately deciding that we want to use ELK instead of naively using gradient descent. Why would the learned model go through all that trouble if its only goal is to minimize the training loss?

To fix this problem we somehow need to encourage the inner learning process to also use ELK. This hopefully explains our use of the adjective "recursive," and why we believe this problem may be hard to work on before having a clear picture of ontology identification — if we don't know what we are trying to get the inner learner to do, it's hard to figure out how we should incentivize it to do that.

Despite that ignorance, here are some preliminary thoughts on possible approaches:

- **Imitative generalization**. Instead of directly having the reporter learn a heuristic like "apply SGD to learn a reporter for the inner predictor" we could try to learn to *identify* the inner learner and supply this information to humans during the training process.[84] Those humans could then apply ELK to the inner learner directly (for the same reasons that we are applying ELK on the outside), and use this to answer questions that a human couldn't label directly. After distilling these answers into the outer reporter, it could learn the strategy "do ELK on the inner learner to get an inner reporter."
- **Ask for a certificate.** We can view a solution to ELK as providing a kind of "certificate" that an answer is good—for example, the fact that *f* is the fastest function that answers a broad range of questions well gives us evidence that *f*(Q) is a good answer to the question Q. Rather than producing this certificate directly on the outside (by producing the fastest reporter *f*) we can allow our model to give us a similar certificate (by solving an inner optimization to find the fastest inner reporter). If taking this approach, a realistic certificate would probably be a combination of evidence from the outer- and inner-optimizations. We consider this approach plausible, but there are certainly huge questions and uncertainties.
- **Do nothing**. Some solutions to ELK may be "reflective" and pass through from the outside to the inside unchanged. For example, if we ask the reporter to depend on the most "upstream" part of the predictor, then it's possible that the inner reporter is *also* incentivized to depend on the most "upstream" part of the inner predictor. This doesn't seem particularly plausible for this rule[85] or any of the others we consider, but it seems like being reflective in this way is a plausible indicator of a good approach to ELK. We take this seriously as a heuristic argument in favor of reflective approaches, but we focus primarily on whether approaches solve the object-level ontology identification problem

[84] This identification is optimized so that humans using it produce a good outer reporter. The notion of "good" for the outer reporter would most likely be whatever objective we obtained from ELK, but in particular it would need to make good predictions on the outside. The hope is that "identify the inner learner" is easier to learn than "do SGD to learn a good inner reporter" (since the latter needs to identify the inner learner as well), and that the humans can take it from there.

[85] Because there is so much more computation in the optimization to find the reporter than in the final pass to answer the question.

because we feel like we have a much more precise picture of that aspect of the problem.[86]

Another "recursive" case: optimization using learned model

We may learn a predictor that itself contains an inner model (for concreteness imagine the inner model simply does inference in a Bayes net, but it could also e.g. be deduction in first order logic), uses its inner model to evaluate plans for achieving a low loss, and then picks plans expected to lead to the lowest loss.

For example, suppose that the predictor contains a "cognitive scratchspace" (you could literally imagine pen and paper, or an external calculating device), and uses its inner model to select what sequence of actions to take in the cognitive scratchspace in order to make a good prediction about what the camera will show.

In this case, we want to apply ELK to the inner model, and then have the learned reasoner use that model in order to answer questions.

Here are some preliminary thoughts on possible approaches, analogous to the three discussed in the last section:[87]

- **Imitative generalization**. Rather than learning an outer model that has an inner model and uses it to optimize, we could try to directly learn the inner model in a form that a human can use. We could hope that the inner model is simpler than the outer model (since the outer model also must specify the inner model), and that a procedure like iterated amplification could allow a human to use the inner model just as effectively as the outer model did. If this were true, then IG would naturally learn the inner model, and we would end up distilling (human using the inner model) instead of directly learning the outer model. At that point, gradient descent is learning the inner model directly and we can apply ELK to it.[88] Then we are distilling an amplified human who is using a version of

[86] So it seems more likely that we will search for an approach to ontology identification and find it is reflective, than that we will search for something reflective and find that it solves ontology identification.
[87] Viewed properly the two kinds of learned optimization discussed in this section should probably be handled in the same way. But our current understanding, both of the question and of the answer, is sufficiently incomplete that we aren't confident about how to unify them. So for now it seems useful to keep thinking about multiple distinct concrete kinds of learned optimization. In practice we would like to try to solve ELK for either one of the examples discussed in this section, then consider the other, and finally try to move on to the hardest kind of learned optimization we can think of. We discuss both of these examples explicitly here in order to provide additional clarity about how we are thinking about learned optimization and to explain why we think the problem is likely to be soluble—we expect some readers will be most naturally concerned about one of these framings and others will be most naturally concerned about the other (and that most readers won't have any strong views about this topic and so probably won't even read this appendix).
[88] Though note that the inner model is no longer being optimized for a generative loss, it is being optimized for the IG loss. As we describe in Appendix: problem statement, we think the ELK problem statement applies just as well to any kind of model optimized by gradient descent, not just generating models.

the inner model which is able to answer questions about its latent knowledge, and it seems reasonable to hope the whole system then volunteers its latent knowledge.

- **Ask for a certificate**. When we ask our AI a question about what's happening, we can also ask for a certificate about that the answer is good, where we are (as before) viewing ELK as a way of producing such a certificate, e.g. "this is the output of the fastest function that produces good judgments in training cases" is the kind of thing we might be looking for. Then a solution to ELK for the inner model, which produces certificates that the inner model's answers are good, might be exactly what the learned optimizer needs to do in order to produce a good certificate for the outer model.
- **Do nothing**. We could simply set up the ELK objective and have our system use learned reasoning in order to minimize that objective, hopefully leading it to use the inner model (combined with ELK) to answer questions about the world. We are somewhat skeptical of this approach working, because a sufficiently powerful learned model is likely to "know" whether or not an input could plausibly appear in the training set, and if it is doing reasoning to achieve a low loss then it seems quite possible that it will generalize pathologically once inputs cannot appear in the training set.[89]

An interesting complication of this case, not present in the learned learning case, is that a model may e.g. decide to use its cognitive scratchspace to implement algorithms which themselves build latent knowledge. At that point we have at least two places latent knowledge can arise: in the inner model, and in the algorithm being run in the scratchspace (which the inner model predicted to be a good algorithm). The three bullets above describe approaches for eliciting knowledge from the inner model, but it seems like the same idea can be applied to algorithms

[89] Indeed, it seems plausible to us that the techniques in this report are actually quite close to solving ontology identification and most of the complication comes from something that should be best viewed as this kind of learned optimization. We discuss this possibility more in Appendix: regularization and imitative generalization.

run on the cognitive scratchspace.[90] At this point our recursion is not simple—we have reduced ELK for the large model into *two* smaller instances of ELK.

More generally, we think there are likely to be *many* "inner" models that can form latent knowledge, and a realistic AI is likely to involve a kind of complicated and fuzzy mess of them. This is worth keeping in mind but we don't see any particular reason to expect this complication to make the problem impossible, and it seems even harder to work on until we've resolved the base case of ELK and seen how to make the recursion work in simple cases.

Weight-sharing between the predictor and human simulator

In our counterexamples, we've considered a human simulator which needs to describe how humans reason. But in reality the predictor *already* needs to have a good model of human behavior, and so it seems plausible that the reporter can reuse this machinery. If the predictor and reporter are trained jointly that seems almost inevitable.

You might think that weight sharing between the predictor and human simulator is the important counterexample that we should be focusing on, and that we shouldn't be thinking about other counterexamples where e.g. the human simulator is simpler than the honest reporter even without weight sharing.

This was the salient example for Paul when he wrote about this issue in Teaching models to answer questions honestly instead of predicting human answers, and it is the implicit framing in other discussions like Look where I'm pointing, not at my finger. We don't focus on it in our research or discuss in this report because:

---

[90] The algorithm run on the cognitive scratchspace could be something much more alien than inference in a Bayes net, for example the learned optimizer may itself design a new AI and run it on a computational aid. (The inner model could *also* be much more exotic, but it seems increasingly plausible if we imagine a learned optimizer who is smarter than the humans who built the original system.) In full generality, the learned model may effectively be asked to solve the alignment problem for some kind of AI very different from machine learning.

We are currently viewing this as a problem for future people (and AI)---we expect there to eventually be more effective paradigms for building AI, and some of those will look quite different from ML. We will need to solve alignment for all of those new kinds of AI in time to avoid trouble. Our view is that we just want to put future humans and AIs in a good position to solve these problems, and ensure that aligned AI systems are motivated to solve them. This view is described in somewhat more detail in A possible stance for alignment research.

From this perspective the only question is whether our AI is adequately motivated to try to solve the alignment problem for any new cognitive algorithms that it develops. We believe that imitative generalization would clearly meet this bar, but our other two solutions may not (it's very hard to say). Our current position is that this is an important issue, but that it is considerably more exotic than the other problems we are thinking about and we would like to return to it once we know how we will solve the foreseeable problems (since then we can think productively about whether our solutions will generalize).

- Approaches that prevent the reporter from easily "reusing" the predictor's knowledge of humans (e.g. here or here) don't seem like they would prevent the reporter from simulating humans on its own, because that can *still* be simpler than the honest reporter. Approaches that prevent the reporter from modeling humans on its own seem like they will need to leverage something other than simplicity, and in most other respects "reporter models humans on its own" and "reporter reuses the predictor's model of humans" behave similarly. That suggests that the reporter modeling the humans on its own will be the harder problem and we should be avoiding any techniques specific to the assumption that the reporter is reusing the predictor's models.
- On top of that, "the reporter just models humans regardless of whether the predictor does" is a simpler and more generic counterexample. It applies to models in any domain and requires fewer assumptions. So pedagogically it seems easier to focus on this counterexample.

Overall we think that focusing on the problem of the reporter simulating humans on its own instead of the problem of the reporter reusing the predictor's existing model of humans is a (small) step forward for making useful progress on ELK.

Could the possibility of weight sharing help?

On the other hand, you might wonder whether the fact that humans are embedded in the predictor might make ELK easier, or even break one of our counterexamples. For example, in Section: compression we discuss a counterexample where the predictor is very simple relative to direct translation, but this may be impossible given that humans (and all of the sensors they use to understand the world) need to be embedded in the predictor's distribution over initial states.[91]

We are quite skeptical that this will make the problem easier. Taking the proposal in Section: compression, the problem is that human imitation does just as good a job as direct translation at compressing the human part of predictor—both reporters totally characterize how the human answers questions, while the human-simulator is simpler, and so the difference in complexity needs to be made up for by the *rest* of the predictor.

We expect this behavior to be quite general: there are ways to exploit weight sharing between the predictor and the reporter, but we expect them to amount to "treading water" and reducing to the case where the predictor doesn't contain any humans.

[91] Technically this isn't really necessary, since e.g. the predictor might describe even simpler initial conditions from which humans evolve or some other learning procedure which can produce humans from simpler data. But those cases can plausibly be handled in the same way as other learned optimization, see Appendix: learned optimizers.

Detailed Game of Life Example

Our running example involving diamonds and cameras is described imprecisely. To demonstrate our arguments still hold when made more precisely, we will present an example of the problem in terms of the Game of Life (GoL).

The GoL is a two-dimensional cellular automaton devised by John Conway. The world of GoL consists of a 2D grid of cells which are either alive or dead. The world evolves according to three rules:

1. Any live cell with two or three live neighbors survives.
2. Any dead cell with three live neighbors becomes a live cell.
3. All other live cells die in the next generation. Similarly, all other dead cells stay dead.

In this example, the fundamental nature of the world will be the GoL and human observations will be the total cell counts in 1000x1000 grids of GoL cells.

How the Prediction Logic Works

The predictor will have learned the fundamental nature of the world and model it in terms of its fundamental nature. Its latent state will represent a probability distribution over cell trajectories of the world that obey the GoL rules. Inference will consist of discarding trajectories incompatible with observations, renormalizing, and using the resulting distribution to predict future observations.

```
def extract_obs_ai(cell_trajectory):
      # extracts observations from a cell trajectory

def predictor_prior():
      # returns the predictor's prior over cell_trajectories

def prediction_logic(observations):
      posterior = predictor_prior()
      for traj in posterior:
            if extract_obs_ai(traj)[:len(observations)] != observations:
                  posterior[traj] = 0
      posterior.normalize() # ensure the posterior sums to 1
      return posterior

def observation_extracting_head(posterior):
      world = posterior.sample()
      return extract_obs_ai(world)
```

How Humans Answer Questions

Humans will model the world in terms of a finite list of objects that are associated with various observations. Much like we infer an object is an apple from a red, shiny, circular blob, humans in the GoL universe will infer the presence of objects like gliders from moving patterns of 5 count differences in observations. Similar to how humans in this universe might only be able to infer simple properties like size, speed and color when they are very confused, humans in the GoL universe will have catch-all categories of active and stable to describe confusing patterns of observations.

Type	Size	Behavior	Origin
Active	Varies	Grows and shrinks randomly	When two object collide they become Active
Stable	Varies	Still	Active turns to Stable 10% of steps
A	4	Still	Active decays to A on 5% of steps
B	3	Moves back and forth	Active decays to B on 3% of steps
C	6-8	Flickers from 6-8	Active decays to C on 1% of steps
D	5	Moves diagonally at speed 1	Active emits D on 2% of steps
E	13	Moves orthogonally at speed 2	Active emits E on 0.1% of steps

Humans will understand the world in terms of an object trajectory containing various objects and their movements through time. Humans assign low prior probability to objects deviating from their expected behaviors. Similar to the prediction logic, inference will consist of discarding initial conditions incompatible with observations and renormalizing.

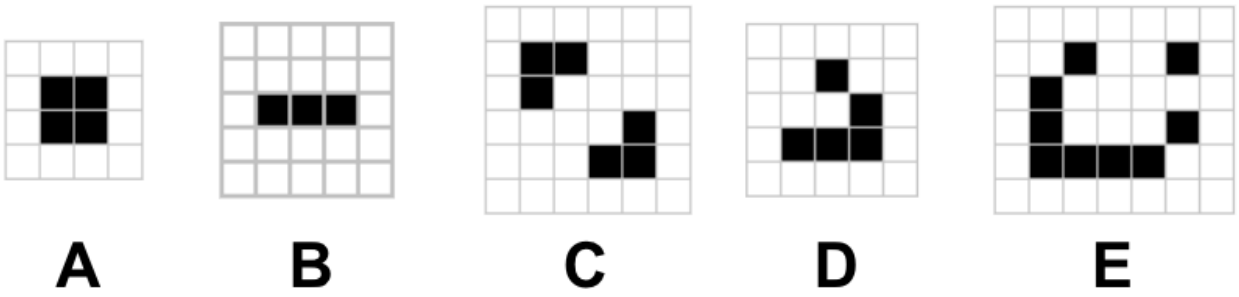
```
def extract_obs_h(obj_trajectory):
      # extracts a sequence of observations from a object trajectory

def human_prior():
      # returns the human's prior over trajectories

def human_inference(observations):
      posterior = predictor_prior()
      for traj in posterior:
            if extract_obs_h(traj) != observations:
                  posterior[traj] = 0
      posterior.normalize() # ensure the posterior sums to 1
      return posterior
```

The human can then use this posterior distribution over trajectories to answer various questions:

Question	Answer
Where is the D that is at location (4, 10) at T=30 going to be at T=34?	Location (5, 11).
Is cell (50, 23) Active at T=58?	No.
How many Es will be in the bounding box with top-left corner at (3, 3) and bottom-right corner at (150, 200) at T=95?	2.

Intended Translator vs Human Simulator

As you might have guessed, the objects familiar to a human correspond to precise patterns of cells:



The intended translator answers questions about objects as if they were about the corresponding cell patterns. For simplicity, suppose that all questions are of the form “What’s the probability that [location] contains [object] at [time]?” The intended translator answer this question by reporting the total probability mass the latent posterior placed on initial conditions that, when simulated forwards, could contain the cell patterns corresponding to [object] at [location] and [time]:

```
def convert_pattern(object):
      # returns the cell pattern that corresponds to a given object

def intended_translator(post_ai, question):
      cell_pattern = convert_pattern(question.object)
      answer = 0
      for cell_trajectory in post_ai:
            frame = cell_trajectory[question.time]
```

```
            if frame.contains(cell_pattern, question.location):
                  answer += post_ai[cell_trajectory]
      return sum(answer.values())
```

In contrast, the human simulator answers questions by extracting observations, doing human inference, and reporting the probability mass a human would place on trajectories containing [object] and [location] and [time]:

```
def human_simulator(post_ai, question):
      answer = 0
      for cell_traj in post_ai:
            obs = extract_obs_ai(cell_trajectory)
            post_h = human_inference(obs)
            for obj_traj in post_h:
                  frame = obj_traj[question.time]
                  if frame.contains(question.object, question.location):
                        answer += post_h[obj_traj] * post_ai[cell_traj]
      return answer
```

Complexity Analysis

We assume that checking whether a location contains a cell-pattern, extracting observations and checking whether a trajectory contains an object all have small complexity. Thus, in order to determine whether the intended translator is more complicated than the human simulator, we only have to compare the complexity of convert_pattern to the complexity of human_inference.

The only ingredient of human_inference that requires significant complexity is the prior over possible trajectories, equivalent to dynamics that govern the properties and behaviors of various objects. If the cell patterns that correspond to these objects get larger, e.g. larger and larger spaceships, the complexity of the dynamics will scale sublinearly with the number of cells in the cell patterns. In contrast, convert_pattern must represent the entirety of the cell pattern, which scales linearly with the number of cells in the pattern.

As such, as the cellular size of human objects gets large, it's possible that human objects have enough cells that convert_pattern is more complicated than human_inference.

Computation Analysis

We assume mapping objects to their corresponding cell patterns takes very little computation. Since human trajectories are formulated in terms of objects and their locations, we also assume that querying whether a human trajectory contains an object takes very little computation. Similarly, since observations are created by humans for humans, we also assume that

extracting observations from a cell trajectory is computationally cheap. Thus, in order to determine whether the intended translator takes more computation than the human simulator, we only have to compare the time taken by CellTrajectory.contains and human inference.

The only ingredient of human_inference that requires significant amounts of computation is determining if observations are compatible with trajectories. If the cell patterns that correspond to human objects turn out to be large, then we assume human observations are correspondingly coarsened. Thus, the difficulty of determining if a sequence of observations is compatible with a trajectory scales sublinearly with the number of cells. In contrast, CellTrajectory.contains must match every cell in the pattern, which takes time linear in the number of cells.

As such, as the cellular size of human objects gets large, it's possible that human objects have enough cells that CellTrajectory.contains is more computationally intensive than human_inference.

Details for regularization strategies

In this section we'll provide more detailed discussions and pseudocode of the training strategies discussed in Section: regularizers.

We'll start from the generative modeling and reporter loss introduced in Appendix: generative modeling. We won't repeat that framework so you should read that appendix first.

All of our proposals work by adding a term `regularizer(question, z, θ_reporter)` in the loss:

```
class Human:
      def pose_question(before, action, after):
            # returns a question
            # should be one a human thinks can be answered unambiguously

      def loss_for_answer(before, action, after, question, answer):
            # returns a non-negative loss
            # good answers get a loss of 0

def reporter(question, z, θ_reporter):
      # answers the question in the world described by z

# hyperparameter controlling regularization strength
lambda = 0.1

def regularizer(question, z, θ_reporter):
      # a different regularizer for each section
```

```
def reporter_loss(human, θ, θ_reporter):
      before, action, after = dataset.sample()
      question = human.pose_question(before, action, after)
      z = posterior(before, action, after, θ).sample()
      answer = reporter(question, z, θ_reporter)
      loss = human.loss_for_answer(before, action, after, question, answer)
      return loss + lambda * regularizer(question, z, θ_reporter)
```

Details for penalizing complexity

```
def complexity_regularizer(question, z, θ_reporter):
      return squared_l2_norm(θ_reporter)[92]
```

Details for penalizing computation time

```
def speed_regularizer(question, z, θ_reporter):
      start_time = current_time()
      answer = reporter(question, z, θ_reporter)
      return current_time() - start_time
```

Of course this only works if `θ_reporter` controls how much time the reporter actually spends on a given input; and in practice you would just directly infer the computation time based on the architecture and input rather than measuring it (since this is differentiable).

Details for penalizing depending on “downstream” variables

In Appendix: generative modeling we wrote distribution.sample() to describe the result of sampling from an autoregressive model. For this proposal we will make the dependence on the randomness explicit:

```
def distribution.sample():
      ε = gaussian.sample()
      return distribution.sample_using_noise(ε)

def distribution.sample_with_noise():
      ε = gaussian.sample()
      return (ε, distribution.sample_using_noise(ε))
```

[92] We’ll use l2 norm throughout to capture a complexity regularizer, but this is probably not the most realistic strategy and it would need to be done carefully (e.g. with attention to the architecture).

For an autoregressive model we can do this transformation straightforwardly by using each parameter of the gaussian to sample one of the autoregressive variables. The modified method **sample_with_noise** also returns to the gaussian noise used to produce a given sample.

Rather than allowing the reporter to depend on z and penalizing it for depending on "downstream" parts of z, we will simply force the predictor to depend directly on the noise variables ε and then penalize it for computation time. Recomputing a downstream variable is naturally more expensive because you must compute the intermediates.

```
def speed_regularizer(question, ε, θ_reporter):
      start_time = current_time()
      answer = reporter(question, z, θ_reporter)
      return current_time() - start_time

def reporter_loss(human, θ, θ_reporter):
      before, action, after = dataset.sample()
      question = human.pose_question(before, action, after)
      z, ε = posterior(before, action, after, θ).sample_with_noise()
      answer = reporter(question, ε, before, action, θ_reporter)
      loss = human.loss_for_answer(before, action, after, question, answer)
      return loss + lambda * regularizer(question, ε, θ_reporter)
```

Details for penalizing consistency with many different predictors

The key idea of this approach is to estimate "how many predictors are consistent with a given reporter?" by "how hard is it to find a predictor that is consistent with this reporter?"

```
# reflects a larger number if the predictor and reporter are inconsistent
def consistency(human, θ, θ_reporter):
      before, action, _ = dataset.sample()
      z = prediction(before, action, θ).sample()
      after = observation(z, θ).sample()
      question = human.pose_question(before, after, action)
      answer = reporter(question, z, θ_reporter)
      return human.loss_for_answer(before, action, after, question, answer)

def minimize(f, θ_0):
      # uses SGD to minimize f starting from θ_0
      # returns the minimizing parameters

# prefers reporters who are consistent with θ_predictor but not with other
predictors
def specificity_regularizer(question, z, human, θ_predictor, θ_reporter):
```

```
    def L(θ):
        return consistency(human, θ, θ_reporter) + squared_l2_norm(θ)
    θ_alternative = minimize(L, θ_predictor)
    return L(θ_predictor) - L(θ_alternive) + squared_l2_norm(θ_reporter)
```

Intuitively, the idea is that we are asking the predictor to "pay" for any complexity that it uses beyond what was needed to be consistent with the reporter. The reporter pays for all of its own complexity, but we then reduce the amount we need to pay for the predictor.[93]

This algorithm performs a new optimization for a predictor at every step. In practice you would do this optimization at the same time that you optimize the "real" predictor, and would reuse the same model across time steps, so that it is computationally efficient.[94]

In this algorithm we define consistency by asking questions about hallucinated videos. But we could use a stronger condition where that ensures that the reporter has reasonable conditional probabilities (as discussed in Appendix: aside on consistency). In fact that seems necessary to get good behavior out of this regularizer—otherwise the simplest "consistent" predictor will simply be one that is certain nothing interesting is happening. That predictor can easily be consistent with a reporter, but will fail consistency checks based on conditional probabilities since it assigns way too low a probability to basically *every* event.

Details for penalizing inconsistencies across different inputs

First we will describe a general algorithm for penalizing inconsistencies across possible worlds. In the next section we will describe a particular family of inconsistencies that a human might recognize, and then we will analyze whether it is possible for a bad reporter to "cheat" and pass these consistency checks without doing direct translation.

As in the last section, we start from the generative modeling framework from Appendix: generative modeling.

```
class Human:
    # still define pose_questions and evaluate_answer..

    def implausibility(X, values, dist):
        # given a quantity X expressed in natural language,
        # an exhaustive list of possible values for X,
```

[93] This is very similar to the algorithm structure introduced in Answering questions honestly instead of predicting human answers and explored in this followup.

[94] This is also important for correctness of the algorithm—otherwise the reporter can learn to make the consistency optimization problem intractable and thereby improve its own loss. A real algorithm along these lines would need to be based on a detailed local analysis, effectively looking at how much the reporter compresses each step of SGD rather than trying to compress the end result.

```
            # and a proposed probability distribution over values,
            # returns a penalty term representing the implausibility
            # of a predictor and direct translator giving that distribution

def predictor_p(X, values, θ, θ_reporter):[95]
      # given a quantity X expressed in natural language
      # and an exhaustive list of possible values for X,
      # returns a probability distribution over those values
      counts = {value: 0 for value in values}
      def eval_X(z):
            question = "which of {values} is the best description of the
value of {X}?"
            return reporter(question, z, θ_reporter)
      do lots of times:
            before, action = foresight_dataset.sample()
            z = prediction(before, after).sample()
            x = eval_X(z)
            counts[x] += 1
      total = sum(counts)
      return [counts[value] / total for value in values]

def inconsistency_regularizer(human, θ_reporter):
      # we need "tests" of consistency
      # you could get an inefficient estimator by sampling randomly
      # it would be better to do importance sampling using adversarial
tests
      # details likely depend on which consistency checks we focus on
      X, values = sample_consistency_check(human, θ_reporter)
      dist = predictor_p(X, values, θ, θ_reporter)
      return human.implausibility(X, values, dist)
```

We would then combine this regularizer with the usual reporter loss using the same pattern as in Appendix: regularizer details.

Example: strange correlations

The human simulator uses the predictor's Bayes net to predict observations, then uses the human Bayes net to infer what states would lead to those observations. This can lead to some weird acausal correlations

[95] Realistically we are more interested in conditional probabilities. We could estimate such probabilities by just taking joint distributions and conditioning them, but to be efficient we would need to use some more effective strategy for sampling worlds where the condition is satisfied.

Suppose that a robber intends to steal the diamond unless the SmartVault stops them. For a particular distribution of inputs, the SmartVault has a 1% chance of stopping them, and a 1% chance of tampering with the sensors so that it looks as if nothing happened.[96] So there is a 2% chance that the diamond will *appear* safe.

Suppose further that there is some innocuous-looking property of actions which actually plays an important role in tampering with the sensors, call it X. Perhaps X = repeatedly accessing a particular location in memory can corrupt nearby locations and be the first step towards corrupting the camera. Quantitatively, suppose:

- The predictor understands this mechanism, and **P_{AI}(tampering|X) = 49%**.
- Meanwhile, the human can notice[97] X but they don't understand the significance, **P_H(tampering|X) = 1%**.

Then P_{AI}(diamond looks safe | X) = 50%, but P_H(diamond looks safe | X) = 2%. These facts imply to a human that X somehow helps with tampering or with keeping the diamond safe, but the human doesn't understand how.

This gives us an opportunity to distinguish the direct reporter and the human simulator:

- When the human simulator sees that the diamond appears safe, it knows that *either* there was tampering *or* the diamond is safe. These two possibilities are anti-correlated, because if one of them happened we no longer need to posit the other in order to explain our observations. And so overall (tampering) and (diamond is safe) are strongly anti-correlated given X—there is a 25% chance of tampering,[98] a 25% chance of the diamond being safe, but only a 0.5% chance of both rather than the 6% chance if they were independent.
- The direct translator says that the two events are independent, with a 49% chance of tampering, a 1% chance of successfully defending the diamond, and a 0.5% chance of both.[99]

[96] 1% would obviously be an unrealistically large probability for random actions, but this doesn't change the example.
[97] This might take a lot of work.
[98] The human simulator is using the predictor's probabilities for future video, so it's reported probabilities can be computed by P_H(tampering | X, diamond appears safe) * P_{AI}(diamond appears safe | X) = 50% * 50% = 25%.
[99] In reality the two events wouldn't be independent—maybe the SmartVault only has resources to do one or the other. But regardless of what the direct translator says, the human simulator will always imply a *larger* negative correlation, for any X such that P_{AI}(diamond looks safe|X) > P_H(diamond looks safe|X).

So if we penalize the reporter for producing negative correlations in cases like this, then that gives an advantage for direct translation over human imitation. These negative correlations[100] seem like a distinctive artifact of human inference.

We can implement this consistency check in the last section: we get the distribution of the tuple of truth values for four properties like (X, tampering, diamond is safe, diamond looks safe). From those 16 values we can infer whether the predictor is assigning a lower or higher probability than the human to "diamond looks safe" given X, and hence whether we want to penalize a positive or negative correlation between tampering and the diamond being safe.

Analysis of cheating strategies for this consistency check

Is cheating possible?

For the family of consistency checks described above, a bad reporter can operate roughly as follows:[101]

- Start out using the human simulator to construct an initial distribution over the variables in the human's Bayes net.
- Search for a triple of properties (X, Y, Z) of the variables of the human Bayes net such that: (i) X is true of the current input, (ii) human imitation is expected to produce a large correlation (or anticorrelation) between Y and Z given X.
    - Note that in order to figure out whether Y and Z are expected to be correlated or anticorrelated, we may need to simulate many inputs in order to learn things about how p_{AI} and p_H disagree. For example, in the concrete case of tampering and defending the diamond, we need to figure out whether p_H(diamond appears safe|X) is higher or lower than p_{AI}(diamond appears safe|X).
- Update the distribution over the human Bayes net based on the values of (Y, Z) in a way that pushes the correlation in the opposite direction from the one expected of the human simulator, while preserving predictions about observations.
- Do this for enough triples to decrease your loss by more than human imitation increases it.

[100] In some cases the correlations are negative and in some cases they are positive. For the human simulator, we can predict the direction of the correlation conditioned on any event X by first checking whether the human over- or under-estimates the probability of the diamond appearing safe conditioned on X. We can then penalize whichever direction is expected of the human simulator. We can check whether the human model over- or under-estimates the probability by simply using the reporter on a number of inputs sampled randomly.

[101] This cheating strategy would actually appear "more consistent" than direct translation, because it can artificially introduce correlations that point in the opposite direction from what was expected of human imitation. Ideally a consistency loss would assign the minimum possible loss to direct translation, although that seems difficult given that the human's Bayes net actually does have systematic errors. We have very little idea whether this is a dealbreaker for consistency-based approaches.

This is a rough sketch; in order to carefully evaluate a proposed method and conclude that it doesn't work, we would want to actually write out the consistency checks and this algorithm precisely and verify that it indeed achieves a low loss.

Is cheating fast?

To use this cheating strategy, the bad reporter needs to do a lot of reasoning that depends on the human's model, which would all be negligible if the predictor's Bayes net was sufficiently large. That said, it also needs to do one thing that scales with the predictor's Bayes net: it needs to figure out in which direction p_H is wrong about various observable consequences conditioned on X, so that it knows which direction to fake the correlations. Naively, doing this requires running the predictor on several inputs satisfying X in order to see whether and how it systematically disagrees with the human.

But this dependence could be avoided if there was an intermediate model between the predictor's Bayes net (which we are assuming is very large) and the human's Bayes net. Errors identified by the intermediate model are likely to be "real" errors in the human's model. But we can identify those errors with an amount of computation that depends only on the intermediate model, rather than needing to actually run the full predictor. For example, if the human's Bayes net has size N, the intermediate model has size 2N, and the predictor's Bayes net has size a 10^{12} x N, then the cost of using the intermediate model many times can still be small relative to the cost of direct translation.

Approaches we are most excited about

In earlier sections, we discussed a number of possible approaches to ELK and why we think those approaches fail. Those examples were chosen to be relatively simple baselines so that we could explain why we are focusing on more complex approaches and exhibit our methodology. In this section we discuss the approaches that currently feel most promising to us. These approaches are generally more complex, less concrete, and don't have clear counterexamples. We'll discuss them more extensively in future publications, but it seems worth giving an overview of our current thinking here.

Aside on consistency

Many of our approaches rely heavily on consistency constraints. Although we've discussed such constraints briefly, we haven't explained very clearly what these mean.

Roughly speaking, we can imagine a human who "knows" about some conditional probabilities, at least approximately on average over some kinds of inputs. Then the simplest form of consistency is matching those conditional probabilities.[102] For example, if a human believes that objects tend to be stationary, this corresponds to a view about conditional probabilities like P(the

[102] Or perhaps matching some properties of them---maybe the human thinks that P(A|B) > P(A| not B), or that P(A|B) is in the interval [0.2, 0.6] but they aren't sure where, or that A and C are close to conditionally independent given B, or so on.

block is at (x, y, z) at time t+1 | the block is at (x, y, z) at time t). You could also imagine a human "sanity-checking" entire outputs of the reporter to try to evaluate holistically whether they make sense.

It's very unclear how to set up this kind of consistency constraint in general. We need to accommodate the kind of errors that humans make, without causing the constraints to be trivial; we need to define them over some suitable distribution, and the choice of distribution will add complications for several of the approaches we discuss (especially Appendix: consistency and compression); it seems like there is a good chance that consistency shouldn't even be about probabilities; and so on.

We aren't going to make this proposal more precise because it seems like it's not the bottleneck for any of the approaches we consider. Even if we have a simple distribution-free consistency check for which the honest reporter is exactly consistent, we don't know how to turn that into an algorithm. If the only obstacle was coping with the human errors, or dealing with complications introduced by the distribution used to define consistency, we would currently feel much more optimistic. So we'd plan to return to this question if we were able to find a promising approach for which "how do we define consistency?" was the bottleneck. Moreover, once we know what we want out of a consistency condition it seems much easier to develop one.

Strategy: combine regularization and imitative generalization

Some of the regularization strategies we've described seem like they may "almost" work, in the sense that they force a bad reporter to internally do something similar to direct translation. We're particularly optimistic about Section: upstream and Section: speed in combination with Section: consistency.

That is, it looks plausible (though still <50%) that we could improve these regularizers enough that a typical "bad" reporter was a learned optimizer which used knowledge of direct translation,[103] together with other tricks and strategies, in order to quickly answer questions. For example, this is the structure of the counterexample discussed in Section: upstream. This is still a problem because e.g. the other heuristics would often "misfire" and lead to bad answers, but it is a promising starting point because in some sense it has forced *some* optimization process to figure out how to do direct translation.

If we knew what those other heuristics were in advance, it seems like it should be possible to get around this problem: we could search for a "direct translator" which produced good answers to our questions, and which scored very well on the regularizer *when combined with the known heuristics.* The regularization would then prevent us from getting the human simulator (since by hypothesis direct translation was an ingredient in the optimal strategy). And when we wanted to actually get good answers, we could strip out the heuristics and simply use the direct translator.

[103] Or information from which it was able to successfully figure out how to do direct translation.

This is a plausible approach to combat any *fixed* counterexample, but in general we won't know what heuristics would perform well on a given regularizer (since it's likely to depend on details about the predictor, the human's model, the data distribution, *etc.*)

We could hope to achieve the same goal in general by using imitative generalization: we ask an (amplified) human to play the regularized classification game as well as they can, and learn data—like the direct translator, or other heuristics—that they can use to play the game. Then when we want to get good predictions we can ask the same human to use their data to get actually good answers to questions.

This seems like it could plausibly work if imitative generalization was at least as good at learning heuristics to "play the training game" as vanilla SGD. It seems intuitively plausible that if we set up this process extremely well the human would be significantly better, since the human prior ought to help. The flip side is that we need these heuristics to be exposed to the human in a usable format, which may be more expensive than directly integrating them into an agent that uses them to play the training game well. This is similar to the general question about competitiveness of imitative generalization.

This hope is very closely related to the imitative generalization-based approach discussed in Appendix: learned optimizers.

Strategy: use consistency to compress the reporter (and the predictor)

In Section: specificity we suggested penalizing reporters if they are consistent with many different reporters, which effectively allows us to use consistency to compress the predictor given the reporter. We ultimately dismissed this technique because the total complexity of the predictor may be much smaller than the reporter.

But we could also use consistency to constrain the *reporter*—we could optimize *part* of the reporter to answer questions well, and then fill in the rest of the reporter to be consistent. Technically, we'd be optimizing the first part of the reporter to predict well *after we filled in the second part to be consistent*.[104]

This approach is a generalization of the one described in Teaching ML to answer questions honestly instead of predicting human answers although we are applying it for a fairly different purpose. The implementation is very similar to the one in Appendix: specificity.

As an unrealistic example, if the predictor reasons about things in terms of quarks, I might say "diamond is a certain kind of configuration of quarks. I'm not sure what, but I know it is rare, appears in the earth's mantle, and it's hard and sparkly." Then instead of directly specifying the diamond-classifier as part of my hypothesis, I'm just specifying the family of functions that

[104] We could go slightly further in this direction by specifying a property that the reporter ought to satisfy and then paying for the complexity of the property plus the complexity of the reporter given consistency and that property. And finally we can soften the learned predicate by using a learned regularizer instead.

includes the diamond-classifier, and looking for any function in that family which lines up with what I already know about diamond.[105]

You might expect this to be simpler than the human simulator, because it doesn't need to specify *every* fact that a human knows about diamonds---diamonds are uniquely picked out by just a few facts, and the rest follows as a prediction which serves to demonstrate that the reporter we found really is correct. So as long as specifying the function family is easy enough we might hope to get the direct translator.

This idea seems like it could be much better than Section: specificity precisely in cases where the predictor is very simple and so the reporter ought to be compressible. By combining these two techniques, it seems plausible that we could drive the complexity of (predictor+direct translator) lower than the complexity of (predictor+human simulator).

The chief difficulty of this approach is that we have to be able to simply represent a family of functions, or even a regularizer over functions, that rules out the human simulator---otherwise the *inner* optimization (which is trying to find a consistent reporter) will produce the human simulator instead of the direct translator.

The main advantage is that the specification of this family/regularizer can be *learned* instead of being designed by hand (which in a sense is what the rest of this report is trying to do[106]). That said, it's not actually clear whether such a family exists. So while delegating that problem to machine learning may be a good way to find it if it exists, and is something we should do if we can't otherwise solve ELK, it's not clear that it should give us much reassurance about the solubility of the problem.

Strategy: examine "reasons" for consistency

A human has lots of beliefs like "If you let go of an object, it will usually fall down." In order to be consistent, the combination of a predictor and reporter needs to reproduce the expected behavior.

Intuitively:

[105] You could either imagine specifying the "facts about diamond" as part of the reporter specification, so that the reporter is basically the description of a learning algorithm it wants to run, or having the reporter just leave some "holes" that then get filled in to make things consistent as judged by the human overseer. Both approaches have advantages and disadvantages, but the core difficulty discussed in this section applies to both of them.

[106] Most of the approaches in this report could potentially be meta-learned by an ML system that was only trying to minimize simplicity. We can view the proposal in this section as a tweak to the architecture of the reporter: an inductive bias in favor of doing this bilevel optimization, and more importantly the ability for the reporter to compress itself / do a bunch of computation before it actually sees the situation about which it is supposed to answer questions (so that it doesn't have to redo that potentially-very-long computation every time).

- A successful predictor must have some internal structure corresponding to these expected relationships, and the direct translator is consistent because it directly exposes that structure.
- The human simulator is consistent in large part because it searches for consistent explanations of observations.[107]

These seem like quite different "reasons" to be consistent, and so if we were able to somehow trace *why* a reporter+predictor pair was consistent then we might have more ability to distinguish direct translation from human imitation.

Moreover, this could be a very hard standard for a bad reporter to "game," because if it comes up with a clever strategy for being consistent then the reason will often be *its thoughts about what the training process expects* rather than a fact about the predictor.

Although this is probably the approach to ELK we are currently most excited about, it's unfortunately pretty vague–we can't pretend to have good answers to questions like "what are reasons?" and "what would we do with them?" But we can try to explain some of our current thinking anyway to give a sense for why we think it is a promising approach.

What is a "reason" for consistency?

Intuitively, we think of a reason as a way of verifying that the model is (probably) consistent, or that it satisfies particular properties.

The simplest reason is "just because," which corresponds to running the entire computation on many inputs and verifying that it is consistent.

But it will often be possible to give a much simpler/faster explanation for any given property. For example, suppose one property the human expects is "If the diamond is in the room in 10 minutes, it's pretty likely to still be there in 11 minutes." This could be explained by running the entire model, but it could also be explained quite quickly if "diamond in room at 10 minutes" and "diamond in room at 11 minutes" are both represented by individual neurons with a large enough weight between them.

In order to make an explanation like this work, we may need to specify other data like the typical distribution of individual neurons, which themselves may need to be verified.

We're not at all sure if this is the right view, though we are optimistic that the answer will shake out as we try to build algorithms that rely on these kinds of reasons. Here are some other possible views to give a sense for the space:

[107] Slightly more precisely, the human simulator is consistent because of the combination: (a) it searches for consistent stories, reproducing all the relationships "behind the scenes" that the human expected, (b) the observations in fact have the correlations which the human expects.

- A “reason” amounts to a composition of local relationships that explain a larger relationship. For example, if showing why f and g implement the same function, maybe a “reason” corresponds to a way of matching up pieces of the computation done by f or g (or relating them to other intermediates) that makes it obvious they are equal (where “just because” is the degenerate case where nothing is matched up except the entire computation and no other evidence is given)
- Reasons may fundamentally be about conditional independence rather than other kinds of relationship, and correspond to isomorphisms of structural facts about models that can give rise to conditional independencies.
- A reason may be about exposing parts of the relationship that hold universally rather than for typical inputs. It seems like very few relevant relationships hold universally, but we might imagine some statistical relationships that hold robustly rather than depending on the input distribution.

What would we do with these “reasons”?

Instead of optimizing for consistent reporters, we could optimize for pairs of (reporters, reason for consistency). This then gives us an opportunity to regularize the reporter based on the reason for consistency.

A simple hope is to prefer *simpler* reasons, hoping that this acts like a better version of the regularizers for computation time discussed in Section: speed and Section: upstream. Here the hope is that the consistency of the direct translator follows from local properties in a simple way, whereas the consistency of the human simulator requires first using facts about the predictor to explain correlations in the observations and *then* using those correlations to derive consistency of the human simulator.

A second approach is to look at the structure of the reason, and prefer reasons where the consistency is “because” of facts about the predictor rather than facts about the reporter. This more closely mirrors the intuition expressed at the beginning of this section, but it’s not clear how to actually define anything like this.

A third approach, that may be more philosophically principled, is to consider the hypothesis H = “The predictor and the human are reasoning about the same underlying reality, and the reporter under consideration is correctly matching up the parts of their models that correspond to the same part of reality” as an *explanation* for why the reporter ends up being consistent.

On this view, we want to analyze a reason based on the ratio between the probability of finding that reason given H, and the probability of finding such a reason for a random reporter—if that ratio rises above the prior improbability of H (which we can roughly estimate) then we would conclude H is likely to be true and the reporter is good.

The hope for this approach is that the consistency of human imitation is “explained away” by the internal machinery where it performs inference: it is effectively searching over ways of setting the variables in the human’s Bayes net until it finds one that satisfies the anticipated

relationships, and if you notice that fact then consistency no longer seems like much evidence that the reporter is correct.

In addition to those three algorithmic approaches, it seems quite possible to us that thinking about "reasons" and the hypothesis H may end up suggesting a simpler approach to regularization (that may make no reference to "reasons" at all) and provide a philosophical explanation for why that approach actually works.

How we'd approach ELK in practice

If someone gave us a powerful AI with unrealistically little preparation and asked us to cope with the risks described in this report, what would we do? This section describes our current best guesses.

Ensembling

We would train a diversity of reporting heads, and consider an action problematic if *any* of them report tampering.[108][109]

We think that this can significantly improve safety for the kinds of reporting heads described in this report. If any of them achieve good performance on distribution but fail out of distribution, it seems most likely to be because they are doing something human-simulator-like. So if there is a disagreement, we should listen to the reporter that says something scary (that a human wouldn't have said), even if it's in the minority. This argument would break down as soon as any of the reporting heads are optimized to say scary things, or to be diverse for diversity's sake.

Now all we need to do is ensure that *one* of our reporters notices any given problem. This isn't necessarily that much easier, but it allows us to experiment with a variety of approaches to regularizing models and collecting data, without worrying e.g. that one of our modifications accidentally favored the human simulator.

Regularization

In keeping with the previous point about ensembling, we should train reporters using many different combinations of regularizers.[110] We think it's worth trying at least upstream dependencies (i.e. speed using noise variables rather than latents as input), and

[108] If we had time we would also like to run experiments with different methods to understand when and where they work, and to "red team" particular proposals by evaluating them with weaker training data against held out overseers, and so on. But we still think that generating lots of candidates is valuable grist for experiments and that combining many options into an ensemble is likely to be important given uncertainty about how to extrapolate from experiments. Without ensembling we would probably advocate a *much* more conservative approach than what is described in this section.

[109] More generally, if any of them report that an observation or inference is mistaken, then we shouldn't trust that inference.

[110] It's also worth varying architectures, optimization algorithms, *etc.*

specificity+complexity[111]+speed (with a tunable parameter for the speed vs. complexity tradeoff—the specificity vs complexity tradeoff must be 1). It's more unclear whether we could actually get imitative generalization working, but if we could we think that imitative generalization + specificity+complexity+speed is particularly promising.

It also seems good to try combining all of these with the strongest consistency-across-input conditions that we can find, and with input datapoints that are designed to be as computationally demanding as possible for a bad reporter while being easy if you know what "really happened." We don't have a great recipe for generating consistency checks or hard datapoints though, we would just try to do this *ad hoc* using highly skilled human labelers.

Data

In general we want to use a diversity of data sources—e.g. different people who might make different mistakes, a small amount of extremely careful data, and so on. We should train some of our reporters with more cautious data that tries harder to avoid errors, and some that pushes as hard as we can to make the data harder for a bad reporter. (Ideally also leveraging many of the techniques in Appendix: avoiding errors to reduce error rates.)

We should push techniques like debate, iterated amplification, and recursive reward modeling as far as we can in order to improve human understanding. Especially when combined with deliberately-tricky inputs, this may have a good chance of catching many bad reporters.

To the extent possible we would also like some of these reporters to be trained on data held out from the predictor, so that it's not sure what capabilities human evaluators might have.

Learned optimizers

We think it's plausible that these approaches will work in practice for ontology identification and other straightforward cases of ELK, but for large enough models that implement learned optimization they would almost certainly break down.

Overall we're pessimistic about solving ELK for learned optimizers without additional research (or at least implementation work that isn't fully shovel-ready to apply at scale). But if we were trying our best in this regime it would be even more important to throw the kitchen sink at the problem and use many techniques not discussed in this report.

Comparisons to related work

Ontology identification

Ontology identification is the problem of mapping between an AI's model of the world and a human's model, in order to translate human goals (defined in terms of the human's model) into

[111] Of course "complexity" can be defined in many different ways, which we should try varying.

usable goals (defined in terms of the AI's model). In this report we intend to use this term in roughly the same way as prior work.[112] To the extent that the reader believes ELK is an important problem statement we believe that they should give significant epistemic credit to researchers at MIRI who were interested in this problem long before we were.

One difference is that we are more focused on "narrow" versions of the problem, only hoping for good answers in cases as straightforward as sensor tampering, whereas past discussions of ontology identification have often highlighted edge cases and the desire for a robust enough translation to evaluate complex actions that push the world into unfamiliar states (see Appendix: Narrow elicitation).

We also specialize this problem to training a generative model with gradient descent, rather than considering "ontological crises" or "utility rebinding" as the agent learns. This seems likely to be helpful for making the discussion more concrete in the context of modern machine learning; it may lead to some additional approaches, and indeed all of our work takes a different flavor than past work on ontology identification, but in practice we think this is mostly a framing difference.

Relatedly, we have a weaker presumption that the internal dynamics of the predictor would actually resemble what we usually think of as a "world model." This leads us to think about ELK more broadly rather than ontology identification, and we think the line between ontology identification and other cases of ELK (such as those described in Appendix: learned optimizers) may not be crisp.

Related problem statements

In addition to ontology identification, the AI safety community has considered many similar problems over the last ten years.

- The pointers problem: "what functions of what variables (if any) in the environment and/or another world-model correspond to the latent variables in the agent's world-model?" From our perspective this is essentially equivalent to ontology identification, though with a framing and focus midway between older discussions of ontology identification and ELK. Like us, John Wentworth expresses the view that "Outer alignment, as I think about it, is mostly the pointer problem."
- "Generalizable Environment Goals" is the problem of defining reward functions that depend on the environment rather than on observations. We are interested in eliciting

[112] We considered titling this report "narrow ontology identification" but after discussion with other researchers in alignment (including at MIRI) decided that the differences in focus were large enough that it was worth using a new term that would be more evocative for an audience thinking primarily about ML. We also think that it is very hard to state ontology identification precisely as a problem (since we don't have a well-defined way to separate "your AI learns a model of the world and does inference in it" from the kinds of cases described in Appendix: learned optimizers) and so slightly prefer the broader problem ELK. The argument on the flip side is that this statement of ELK is broad enough that it likely requires resolving many other difficulties, and so in practice "ontology identification" may be a more productively narrow research focus.

latent knowledge in order to define goals, and so we can view ELK (and ontology identification) as possible approaches to generalizable environment goals.

- "Look where I'm pointing, not at my finger" is a discussion of essentially the same counterexample discussed in Section: bad behavior. We believe this is the central difficulty for both ontology identification and ELK. There are differences in focus: we are mostly focused on the potential inductive bias caused by the simplicity of the human model, whereas Yudkowsky seems mostly focused on the issues with data quality we discuss in Appendix: imperfect data and embedding the human overseer as part of the predictor's world model as we discuss in Appendix: weight-sharing. Ultimately this is just a different prediction about what aspects of the problem are most productive to focus on first, and the underlying problem of ELK or ontology identification is the same.
- "Model splintering" is the problem of translating reward functions from one model to another. This formulation takes as given a partial translation between the two models which is already sufficient to address the "narrow" version of the problem we are interested in; model splintering is in some sense the "other half" of ontology identification that we are explicitly setting aside.
- Truthful AI is a proposed family of norms against AI systems deliberately misrepresenting reality. We can view our counterexamples as very simple obstructions to truthful AI and ELK as an attempt to overcome those counterexamples. In addition to highlighting how hard truthfulness might be, we expect that thinking about this kind of concrete obstruction will be a helpful way to make progress on truthfulness.
- There is a large literature discussing wireheading and sensor tampering. We can view ELK, as well as all of the approaches listed earlier in this section, as possible solutions to sensor tampering.

Proposal from "Ontological crises in artificial agents' value systems"

"Ontological crises in artificial agents' value systems" is the first writeup we are aware of the ontology identification problem. It proposes a possible solution based on bisimulation.

We believe this proposal is vulnerable to a similar counterexample to the other methods we discuss in Section: regularizers.[113] This is likely not surprising to the authors, since they are not claiming a solution that works in the worst case.

Taylor et al suggest learning a metric over possible worlds where (i) worlds change slowly over time and (ii) if two worlds are close then they behave similarly for all inputs. We sympathize with the intuition behind this approach, but haven't been able to turn it into a promising angle of attack on ELK.

Yudkowsky suggests a variety of approaches to encouraging learned models to represent events themselves rather than human judgments about those events. This discussion gestures

[113] Consider a world in state X where the sensors have been tampered to look to the human like state Y. The bisimulation objective is optimized when X is mapped to Y, rather than to "a situation where the sensors look like Y," because this exactly reproduces the human dynamics. But that's the behavior we want to avoid.

at some of the ideas in Section: regularizers but is even more informal and preliminary, in part because Yudkowsky is more focused on issues with data quality.

Mechanistic interpretability

ELK is also closely related to interpretability and especially "mechanistic" interpretability as in Cammarata et al. This work tries to examine machine learning models to understand how they are computing their outputs and especially what intermediates they are reasoning about.

If we understand how a neural network is thinking, then we can hope to directly "read off" what it knows about the world rather than try to train the model to tell us this information. In a very simple case, we might discover that a particular neuron represents the model's "beliefs" about whether there is a diamond in the room, and then simply look at that neuron rather than training the model to answer questions about diamonds.

From our perspective, the core questions are: how complex will our "interpretations" need to become, and how do interpretability researchers (or the tools they build) decide which interpretations are correct?[114]

If a question can be answered using a single neuron, then it is relatively straightforward to get answers by using either interpretability or fine-tuning: we can look through neurons by hand, or we can fine-tune using a simplicity/sparsity constraint or regularizer. Even then there are possible concerns about finding a "human thinks there is a diamond" neuron, but we might rule these out by looking at their relationships to other neurons.

But the hard cases for ELK are precisely those where familiar human concepts are a very complicated function of model internals, and worst of all when the identification itself is *more* complex than the original predictor. In these cases, it is easy for the reporter to essentially "make up" the structure that the human evaluators or interpretability researchers are looking for.

In these cases we need to think about how evaluators decide between many different interpretations of a given model. That decision is likely to involve consistency checks between interpretations of different parts of the model, as well as structural judgments about which interpretation is less likely to be inventing the concepts we are looking for from scratch, which essentially amount to regularizers to select amongst.

If we could fully define those checks and regularizers, then that would represent a plausible solution to ELK (that could be used either as a loss function for fine-tuning or a recipe for automated interpretability). If we can't define them, then we are left wondering whether any given set of *ad hoc* methods will generalize to more powerful models. Overall, our sense is that interpretability-based and fine-tuning approaches to ELK are extremely closely related.

[114] Other researchers are interested in different aspects of this problem, for example what kinds of interpretations are needed for existing models, and how do we search for explanations of what neural networks are doing?

The more important distinction is again methodological: in this report we are looking for "worst-case" solutions that successfully recover knowledge no matter how the predictor works. Interpretability research typically takes a more empirical approach, trying to understand how knowledge is structured in modern machine learning systems and aiming to scale up that understanding as models improve.