Paths to AGI through Self-Improvement

Eddie Zhang

December 2024

Abstract

This will be an attempt to give some rough sketches of how one might tractably (within the next 5 to 10 years with O(100M) dollars) achieve AGI. There is no guarantee of safety.

1 Assumptions

Assumption 1.1. We assume an ego-centric view of AGI, meaning there exists some boundary between the *environment* $E = \big(S, A, p(s'|a, s)\big)$, and the AGI $\pi : S \mapsto A$ (I reserve capital P for the set of all possible transition dynamics $p \in P$). Not all forms of AGI need follow ego-centricity, as one could imagine some self-assembling environment which performs useful computation, or multi-agent systems such as hive minds and human organizations where other agents are both part of the AGI as well as the environment. Although what follows could likely be extended to other forms of AGI that do not respect an agent-environment interface, we restrict our attention to ego-centric AGI for the sake of intuition and ease.

Assumption 1.2. We assume that within both the state space S and A there is a representation of π: $\text{rep}(\pi)$. For example, this could be computer code and/or neural network weights. This is crucial, as it allows the possibility for π to become a self-improvement operator. In other words, we could choose a π such that $\pi(s) = \pi(\text{rep}(\pi), ..) \implies a = (\pi', ...)$.

Assumption 1.3. There exists some *learnable* and non-learnable part of our policy. Learnable means that it changes as the AGI interacts with the environment, while the non-learnable piece does not change. We initialize our policy *tabula rasa*, with no priors or knowledge. This means the learnable piece is sampled randomly from a distribution that with high probability receives performance equal to a uniform random policy. Performance is defined below.

2 Definitions

Definition 2.1 (Task, aka Problem)**.**
A task T is defined as an MDP: $(S, A, r(s, a), \gamma, p(s'|a, s))$. That's it.

A trajectory is defined as $\tau = (s_0, a_0, r_0...)$, return $R(\tau) = \sum_t^\infty \gamma^t r_t$, and performance $J(\pi) = \mathbb{E}_{\tau\sim\pi,p}[R(\tau)]$.

Claim 2.1. Almost all 'problems' defined in the loose real-world sense can be formalized as an MDP. This is also known as the 'reward hypothesis', from Sutton. See this paper [2] for a more detailed and nuanced exposition from people who have thought about this for much longer than I have.

Although it may not initially seem like it, even problems like social ones such as homelessness and climate change, as well as philosophical ones like the meaning of life and the existence of free will can be written as MDPs. The reason is that if we abstractly define a problem as the absence of a solution, where a solution is defined as some different state than the one we currently reside in, then any problem can simply be thought of as goal-reaching, which can immediately be written as an MDP. One may protest: What if my solution can only be specified as a trajectory, rather than a final goal state? What if I break the Markovian assumption? What about the transition function stationarity assumption? Both the goal specified as trajectory and the Markovian assumption worry can easily be solved by defining the state space S to be a history of all prior states and actions, as is done in RLHF. Stationarity can also be removed, be adding the current timestep to the state. In fact, a fun problem is to try and imagine a problem which *cannot* be written as an MDP.

To see an alternative formulation for someone who disagrees with me and whom I respect greatly, see [1].

Definition 2.2 (Learning Algorithm)**.** Define a learning algorithm $\mathcal{A} : \text{rep}(\Pi) \times \mathcal{T}^n \mapsto \text{rep}(\Pi)$. Here, $\mathcal{T}^n$ refers to the space of all prior experience (space of a set of trajectories), while rep(Π) refers to the representation space of all possible policies. Notably, due to Assumption 1.2, $\mathcal{A}$ *can be part of* π.

Definition 2.3 (Intelligence)**.** All that is meant by intelligence can likely be formalized as simply the expected discounted sum of performance over time. Concretely, this means taking the policy at every timestep (after each sample in the environment) and calculating its performance $J(\pi)$ at that timestep, and getting the discounted sum of all performances. In other words, we want the highest asymptotic performance, as fast as possible. Note π here may be a random variable, due to possible environment and/or learning stochasticity. Thus, Intelligence for a task T is $I(\mathcal{A}, T) = \mathbb{E}_{\pi_t\sim\mathcal{A},p}[\sum \gamma^t J(\pi_t)]$.

Some may argue that Intelligence is simply $J(\pi)$ of a converged policy from $\mathcal{A}$. After all, when we evaluate how 'smart' the newest language model is from our favorite lab, the policy has already been fixed and is no longer changing. However, I argue that this misses a crucial aspect of Intelligence, namely that it also matters *how quickly* (how many samples) it took to get to that level of performance. To me, it is necessary but not sufficient to have high performance; perhaps one can think asymptotic performance as 'crystallized' intelligence and sample efficiency as 'fluid' intelligence. In my mind, you require both to be considered intelligent. You could perhaps test for both using just a fixed policy

if you evaluate on some sort of meta-MDP you've defined where actions are policies and reward is set as the discounted sum of performance of the policy, although at this point I believe we've just returned to the same spirit of my original definition.

On the other hand, if you care only about 'crystallized' intelligence, then evaluating $J(\pi)$ would be fine, although you may want to consider using a different test environment transition function. For example, with a different inital state fn $\rho(s_0)$ than the one you trained on, to evaluate OOD generalization. Note even the original train environment will test for generalization given that it's large enough, as during rollout you'll likely encounter new states that were not seen during training.

Definition 2.4 (General Intelligence)**.** Define a new general goal-conditioned MDP $\mathcal{M}_g$. Define the set of all tasks T from 2.1 as $\mathbf{T}$ and assign each a description $d_T \in D$ using some function d. d_T should ideally be as short as possible (see Kolmogorov complexity). Possible specifications of d_T could be a set of goal states, a series of demonstrations from π^*, a reward model, or a natural language description. Next, define a distribution G over all tasks. The most natural distribution is likely one which is inversely proportional to the description length of the task. Optionally, one could also weight this distribution proportionate to economic utility, difficulty, or to tasks deemed especially important. Then, define $\mathcal{M}_g$ as $(S_g, A_g, r_g) := (\bigcup_{T\in\mathbf{T}} S_T \bigcup D \ , \ \bigcup_{T\in\mathbf{T}} A \ , \ r(s,a) = \sum_{T\in\mathbf{T}} r_T(s,a) \cdot \mathbb{1}\big(d_T = d(T)\big)$, and let γ_g, p_g be defined similarly to r as just selecting for the given task at hand, weighted by G. Then General Intelligence is defined as $I_g(\mathcal{A}) = \mathbb{E}_{T\sim G, p_g}[I(\mathcal{A}, T)]$.

3 Safety

3.1 Human Bad Actors

Between individuals, organizations, and countries. Societal-level risk with the last one.

3.2 Internal Misalignment and Loss of Control

Between the model and the user of the model.

4 Theory

5 Methods

5.1 RL on space of RL algorithms

The key idea here is to create a self-improvement operator by defining an MDP where S is chosen to be $\text{rep}(\mathcal{A}) \times \mathcal{T}^n$ and A to be $\text{rep}(\mathcal{A})$. Now a policy π

becomes an algorithm improvement operator, and we can improve π using the current best $\mathcal{A}$. In other words, we have a policy which searches for the best possible algorithms for improving itself. Define the set of possible algorithms as those that could be proven to hold for some desired conditions, such as guaranteed convergence to optimal policy, monotonic improvement, sample complexity bounds, etc. One possible choice could be the Mirror Learning Space introduced in [4], which encompasses a broad set of algorithms such as PPO and DDPG.

Unfortunately, solving such a MDP is likely computationally intractable, as the set of possible algorithms is combinatorially large, and the feedback mechanisms (running several rounds of RL in parallel) could result in a single iteration taking hours to days running on 4-8 H100s (at least in 2024).[1]

The reward function is this case would be defined as the general Intelligence metric, which is a bandit reward not dependent on state: $r(s, a) = r(a) = I_g(a = \mathcal{A})$.

The regular RL agent-environment dependency graph is drawn below:

$$s_i \xrightarrow{\pi_i} a_i \xrightarrow{p} r_i, s_{i+1}$$

We can improve π_i by choosing an algorithm: $a_{i+1} = \underset{a \in \{a_0, a_1 .. a_i\}}{\arg\max} r(a)$. Then we sample trajectories $\{\tau_\pi^i\}_{i=0}^n$ from G and π_i, and finally get $\pi_{i+1} = a_{i+1}(\pi_i, \{\tau_\pi^i\}_{i=0}^n)$. This likely could all be done swapping π for Q^π values instead, as there is likely a general equivalence that can be drawn between the two [7].

5.2 Imitation learn on *everything*, wait a few years for society to learn, then repeat

A straightforward way to bootstrap policy learning is via *imitation* of a (possibly imperfect) expert, then refining that expert through repeated interactions. In Expert Iteration, each cycle alternates between collecting new trajectory data from the current policy and using an *expert-improvement function* to produce more optimal actions. The policy is then retrained to imitate these improved actions. We can do this at the largest scale imaginable by setting the expert to be the entire human society, and then just supervising the AGI on the top 1% of data from society. Top here is defined as whatever we think of as most novel, genius, beautiful, etc.

[1] For reference, a single H100 delivers 200 TFLOPs at fp16. The new GB200 NVL2 delivers 10 PFLOPs, which is 50x, but also glues two GPUs together. A single GPU IC is actually 25x an H100. There's also the B200, which is just a single IC and runs at 4.5 PFLOPs, which is 22.5x the last generation.

Algorithm 1 Cross-Entropy Method (CEM)

Require: Environment E, initial distribution parameters θ_0, population size M, elite fraction ρ, number of iterations N

Ensure: Final distribution parameters θ_N (which define a policy distribution)

$\theta \leftarrow \theta_0$

for $i = 1, \ldots, N$ **do**

  Sample a set of M candidate state-action pairs $\{s_j, a_j\}_{j=1}^{M}$ from $\pi(\cdot \mid s, \theta)$ and / or π_{human}. Make sure to sample multiple a for each s_j.

  Evaluate each (s_j, a_j) in E to collect rewards $\{r_j\}_{j=1}^{M}$

  Identify Elite Set $\mathcal{E}$, consisting of the top ρM actions by reward

  Update $\theta \leftarrow \arg\max_{\theta'} \sum_{j \in \mathcal{E}} \log \pi(a \mid s, \theta')$ ▷ maximum-likelihood fit to the elite set, minimizing cross entropy with p (sampling dist) = society + policy and q (learning dist) = policy

  Update human $\leftarrow \arg\max_{a} \sum_{j \in \mathcal{E}} \log \pi(a \mid s)$.

end for

return θ_N

Here, the elite set identification is done either automatically or by human society, and the set of M candidate actions is also done by human society in conjunction with the model. By repeatedly collecting new and better diverse trajectories, the policy π successively learns to imitate higher-quality behavior. Since humans are naturally diversity-seeking, this method is not likely to suffer from entropy collapse and is almost guaranteed to work. The only issue is that getting new more optimal data than from society takes an extremely long time - a conservative estimate maybe a single generation, or 25 years. You can imagine that repeating this process 3 4x will get us to AGI, which puts an upper bound on the time to AGI at $\tilde{1}00$ years, provided that society doesn't collapse. Over multiple iterations, this procedure can converge to a level of performance near or exceeding that of the original expert.

5.3 Self improving AI scientist

This idea is basically the same as subsection 5.1, except we inject a bunch of inductive biases based on the scientific method. As a result, this one seems much more computationally tractable, and others have begun to explore it [6].

$$\text{Observation} \rightarrow \text{Question} \rightarrow \text{Hypothesis} \rightarrow \text{Prediction} \rightarrow \text{Experiment} \rightarrow \text{Analysis} \rightarrow \text{Conclusion} \xrightarrow{\text{repeat}} \text{Hypothesis} \rightarrow \ldots$$

See Appendix A for an AI-generated detailing of this idea.

6 Related Work

The best thing I've seen by a pretty long shot here is Marcus Hutter's work on AIXI [3]. He lays out a formal description of AGI, that has some very novel ideas. This is one of the best works I've seen in the entire field. For some accessible introduction to his work, see this interview `https://www.youtube.com/watch?v=E1AxVXt2Gv4` or this one `https://www.youtube.com/watch?v=7TgOwMW_rnk`. See [5] for informal definitions of intelligence.

Compression is intricately connected to intelligence. A compact way to say this is that *generalization is compression*: when a policy or model learns, it builds an internal code that maps many superficially different inputs to a smaller set of reusable latent causes. New data "looks old" once projected into this latent space. Formally, learning can be viewed as searching for a representation M that minimizes total description length $L(M) + L(D \mid M)$, where $L(\cdot)$ is code length and D is experience. A system generalizes when $L(D_{\text{new}} \mid M)$ stays small—i.e., the same code compresses fresh data. Overfitting, on this view, is failed compression: you memorize idiosyncrasies (long codes per example) rather than discovering structure (short codes reused across examples).

This perspective aligns with Kolmogorov/Occam ideas that also underlie AIXI and related formulations [3, 5]. In RL terms, model-based predictions compress transition statistics; policies compress high-entropy state–action choices into structured behavior; meta-learning compresses across tasks by amortizing shared subroutines. Improvements in "intelligence," as defined here, amount to finding shorter programs (representations, algorithms, skills) that explain more of the task distribution and convert prediction compression into control performance.

What about *novelty*? The compression view seems to miss it, since novelty is by definition hard to encode with the current codebook. One stance is that true novelty may be rare or ill-posed; many seemingly new things are recombinations that *become* compressible after an update to the latent space. Another is that novelty simply sits orthogonal to intelligence-as-compression: it measures *codebook growth*, not *code reuse*. So creativity drives ΔM, proposing new primitives that later reduce $L(D \mid M)$. One could design a curiosity-driven MDL agent that chooses experiments/actions to maximize the expected decrease in $L(M) + L(D \mid M)$ after permitting codebook mutations ΔM, thereby using intelligence-as-compression to actively seek genuinely novel structure.

References

[1] David Abel et al. "A definition of continual reinforcement learning". In: *Advances in Neural Information Processing Systems* 36 (2024).

[2] Michael Bowling et al. "Settling the reward hypothesis". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 3003–3020.

[3] Marcus Hutter. "A theory of universal artificial intelligence based on algorithmic complexity". In: *arXiv preprint cs/0004001* (2000).

[4] Jakub Grudzien Kuba, Christian Schroeder de Witt, and Jakob Foerster. "Mirror learning: A unifying framework of policy optimisation". In: *arXiv preprint arXiv:2201.02373* (2022).

[5] Shane Legg, Marcus Hutter, et al. "A collection of definitions of intelligence". In: (2007).

[6] Chris Lu et al. "The ai scientist: Towards fully automated open-ended scientific discovery". In: *arXiv preprint arXiv:2408.06292* (2024).

[7] John Schulman, Xi Chen, and Pieter Abbeel. "Equivalence between policy gradients and soft q-learning". In: *arXiv preprint arXiv:1704.06440* (2017).

A AI Scientist by O1-Pro (12/31/2024)

Another viewpoint on self-improvement is to explicitly inject the *scientific method* into the agent's inductive biases and learning loops. One can structure the agent's behavior around iterative hypothesis generation and testing:

1. **Observation.** The agent (scientist) collects data about itself, the environment, or any external sources.

2. **Question.** It forms a question or problem statement: "Which next experiment or theory would maximally reduce uncertainty or improve performance?"

3. **Hypothesis.** Based on the question and prior knowledge, the agent proposes a *candidate explanation* or a *candidate algorithmic improvement*.

4. **Prediction.** The agent deduces predictions or outcomes that should be observed if the hypothesis is correct.

5. **Experiment.** The agent or environment runs an experiment, collecting new data or trying a new approach to gather evidence.

6. **Analysis.** The agent measures the discrepancy between predictions and observed data.

7. **Conclusion.** The agent decides whether the hypothesis is supported or falsified. If the hypothesis is promising, it may refine it further; if not, it may discard or revise it.

8. **(Loop back) Hypothesis.** Incorporate new insights into subsequent hypotheses or algorithms, thus repeating the cycle.

Unlike the purely meta-RL approach, this method tries to explicitly enforce the cycle of *theory-formation and falsification* in a structured manner. The hope is that by employing well-tested patterns from human science (which has proven extremely effective in discovering new knowledge), an AI can systematically push the boundaries of its own model and algorithmic repertoire.

In practice, implementing the "Self-Improving AI Scientist" might involve:

- A structured **memory** or **knowledge base** for storing hypotheses.
- Methods for **active learning** or **optimal experiment design**, so the agent chooses the most informative next experiment.
- Mechanisms for **theory revision**—akin to well-known approaches in inductive logic programming or Bayesian nonparametrics, but scaled up.
- A built-in **robustness** or **safety** layer that ensures experiments that could be catastrophic are recognized and mitigated before being launched.

This approach overlaps with aspects of subsection 5.1, but it is more structured in how it searches for new algorithms or parametric improvements (framing it as a hypothesis-testing loop).

B AI Scientist by O1 (12/31/2024)

In this section, we refine the general idea from subsection 5.1 by injecting strong inductive biases inspired by the scientific method. The overarching goal is to design an AI system that acts as an *autonomous scientist*: it generates hypotheses, devises and runs experiments, interprets results, and recursively improves both (i) its understanding of the world and (ii) its own learning algorithms.

1. Observation and Abstraction.

- **Observation:** The system interacts with the environment (or a suite of environments) and collects raw data. This data might include (i) low-level sensor streams or textual corpora, (ii) existing scientific literature or historical experiment logs, and (iii) explicit demonstrations or expert policies.
- **Abstraction:** From these observations, the AI constructs or refines an internal *model* of the environment. This model can be probabilistic (e.g. Bayesian), deterministic, or hybrid. Crucially, the system attempts to *compress* or summarize large volumes of data into a small number of latent variables or symbolic representations. This is reminiscent of how real scientists identify relevant variables (temperature, pressure, mass, etc.) in a physical system.

2. **Question and Hypothesis Generation.**

   - **Question Formulation:** Based on the system's current understanding, it identifies *unresolved questions* or *inconsistencies* in its model. In practice, this may appear as high posterior uncertainty in a Bayesian model, large reconstruction errors in a generative model, or states where the system's policies fail in unanticipated ways.

   - **Hypothesis Generation:** The AI then proposes a refined or augmented model to resolve these uncertainties. For instance, it might hypothesize that there is an unobserved variable (analogue of "dark matter" in physics) or an unmodeled dynamical constraint.

   - **Hypothesis Ranking:** Because there can be infinitely many ways to explain an observation, the system applies simplicity or Occam's razor-like criteria to prioritize hypotheses with minimal description length or minimal structural complexity.

3. **Experiment Design and Prediction.**

   - **Experimental Actions:** Given a hypothesis, the AI constructs an *experiment* (a policy that probes specific states in the environment) to discriminate between competing hypotheses. Designing such experiments can be viewed as an optimization problem: the AI seeks a trajectory or set of trajectories that maximally reduces epistemic uncertainty or that cleanly *falsifies* competing models.

   - **Predictive Checks:** For each hypothesis, the AI generates predicted outcomes for the proposed experiments. The difference between predicted outcomes and actual outcomes ultimately drives the update of the AI's internal model (akin to a Bayesian posterior update or gradient-based learning step).

4. **Experiment Execution and Analysis.**

   - **Policy Deployment:** The AI executes the experimental policy in the environment (or in a simulated environment, if that suffices). It collects new data that should, in principle, reveal which hypothesis is closer to reality.

   - **Model Update:** The AI updates its model in light of any discrepancies between observed outcomes and predicted outcomes. This could involve standard procedures from machine learning (e.g. gradient descent, Bayesian updates) or more symbolic methods (e.g. rewriting logical clauses).

   - **Hypothesis Refinement:** If the experiment refutes the hypothesis, the AI either adjusts the hypothesis or discards it in favor of alternatives. If the experiment supports the hypothesis, the AI tries to see whether the model generalizes to additional tasks or domains.

5. Conclusion and Recursion.

- **Concluding a Cycle:** The AI synthesizes its findings, consolidating the newly gained knowledge into its learned representations and deciding which modifications to its *own learning algorithms* might yield further improvements.
- **Self-Improvement Recursion:**
    1. The AI not only updates its *environment-model* but also its *algorithmic-model* for how it learns and experiments. The latter step is, in essence, an *RL on RL* approach: the AI re-tunes hyperparameters, modifies its architecture, or even rewrites parts of its own training code.
    2. The newly updated learning process is then applied in future experimental cycles, continuously improving the agent's scientific inquiry process.
    3. Over many cycles, this *self-improving AI scientist* becomes increasingly proficient at asking the *right* scientific questions, designing more targeted experiments, and refining the meta-algorithms that govern how it learns and updates itself.

Implementation Considerations.

1. **Representation Learning and Symbolic Abstractions:** A key challenge is translating raw, high-dimensional data (e.g. image pixels, large language corpora) into meaningful symbolic or low-dimensional representations that facilitate *scientific* reasoning and experimentation. Techniques from variational autoencoders, diffusion models, or graph neural networks could offer a tractable interface between raw data and symbolic hypothesizing.
2. **Experiment Prioritization:** In many real-world scientific domains, *any* experiment is costly in time, resources, or risk. Thus, the system must solve a *multi-objective* optimization that balances information-gain with cost (and safety).
3. **Safety and Falsification Boundaries:** When the AI proposes self-modifications, there must be safeguards to prevent catastrophic interventions in the environment, as well as checks to ensure the AI does not degrade (or "lobotomize") itself. Formal verification or rule-based constraints may be needed.
4. **Long-Horizon Credit Assignment:** Some scientific advances pay off years (or thousands of samples) later. By formulating the "scientific process" in an RL framework with suitable discounting or memory, the AI can *credit* successful discoveries to particular lines of research or algorithmic changes—thus reinforcing good "scientific habits."

Analogy to Human Scientific Practice.

> *"The essence of mathematics (and science) is not to make simple things complicated, but to make complicated things simple."*

Just as a human scientist (or mathematician) refines their own thinking processes over a career of problem-solving, our AI repeatedly re-architects its learning procedures based on performance feedback. It mirrors how mathematicians generate proofs, identify counterexamples, and reorganize entire theories (e.g. Euclidean vs. non-Euclidean geometry) when contradictory observations emerge. Likewise, the best scientists conceive "meta-experiments" that improve their approach to future experiments, eventually honing powerful, generalizable frameworks (e.g. the shift from Newtonian mechanics to General Relativity).

Scope and Potential.

- In principle, a *self-improving AI scientist* could compress decades' worth of domain knowledge and harness a global network of scientific instruments, HPC clusters, and large-scale data sets. By systematically generating, testing, and refining theories, it could yield discoveries at a cadence far beyond human capability.
- Crucially, it could also *rewrite its own "brains"* (learning algorithms, neural architectures, memory modules) to become more adept at each iteration, a virtuous cycle that leads to exponential leaps in internal efficiency, conceptual unification, and external performance.

Taken together, the self-improving AI scientist offers a unified framework for automatically formulating and falsifying hypotheses, all while reinventing its own learning protocols. While this vision remains computationally demanding and fraught with safety concerns, it is arguably the closest approximation to how theoretical physicists, experimental chemists, mathematicians, and computer scientists practice their craft.

C Further Definitions

These are not essential, but are also extremely useful.

Definition C.1 (Model)**.** A *model* is any structured representation that supports at least (a) **predictive power**, (b) **explanation** of observed phenomena, and (c) **fertility**—the capacity to generate or guide new hypotheses. In AI, models can be explicit (e.g., symbolic formulas) or implicit (e.g., neural network weights).

Definition C.2 (Decision)**.** A *decision* is a choice of action in a given state, typically informed by a policy, a model, or both. In modern RL parlance, a decision is a sample $a \sim \pi(\cdot \mid s)$.

Definition C.3 (Uncertainty)**.** We distinguish between **aleatoric** uncertainty, which arises from intrinsic stochasticity in the environment (sometimes called "risk"), and **epistemic** uncertainty, which arises from incomplete knowledge (i.e., from the agent's ignorance). Addressing epistemic uncertainty often requires active exploration or experimentation.

Definition C.4 (Preference)**.** A *preference* is an ordering over outcomes (states, trajectories, or distributions thereof). Preferences can be made explicit via a reward function or utility function.

Definition C.5 (Utility)**.** A *utility* function is an explicit numerical mapping from outcomes to real values, capturing how desirable the agent (or user) finds each outcome. Equivalently, utility functions can be viewed as generalization of reward functions in MDPs.

Definition C.6 (Computability)**.** *Computability* refers to whether a function or mapping can be computed by a Turing machine with finite resources. In practical AI, we also consider time and memory complexity, which can be major bottlenecks in real-world policy learning.

Definition C.7 (Task Difficulty)**.** todo: distill Several research papers have examined the difficulty of reinforcement learning (RL) environments, proposing various metrics and analyses to assess and understand task complexity. Notable contributions include:

• Policy Information Capacity (PIC): Furuta et al. (2021) introduced PIC, an information-theoretic measure that quantifies task difficulty by evaluating the mutual information between policy parameters and episodic returns. Their study demonstrated that PIC correlates with task solvability across various environments, offering a quantitative metric for assessing RL task complexity.

• Ecological Reinforcement Learning: Co-Reyes et al. (2020) explored how environmental properties, such as "dynamism" (the degree to which the environment changes independently of the agent's actions), impact learning difficulty. Their findings suggest that certain environment characteristics can alleviate challenges associated with non-episodic RL in sparse reward settings.

• Random Weight Guessing (RWG): Oller et al. (2020) proposed a method to analyze RL benchmark complexity by evaluating the performance distribution of randomly initialized policy networks. This approach isolates environment complexity and provides a statistical foundation for assessing task difficulty, highlighting that some benchmarks may be more trivial than previously assumed.

• Hardness in Markov Decision Processes (MDPs): Conserva and Rauber (2022) conducted a comprehensive survey on the theoretical aspects of environment hardness in MDPs. They introduced "Colosseum," a benchmarking suite designed for empirical hardness analysis, enabling systematic evaluation of RL agents across environments with varying difficulty levels.

and don't forget effective horizon laidlaw, sample complexity kakade, and eluder dimension.

These studies contribute to a deeper understanding of RL environment difficulty, offering metrics and frameworks that aid in the design and evaluation of RL tasks and algorithms.