Comparing the Complexity of Robotic Tasks

Michelle Ho*, Alec Farid*, and Anirudha Majumdar

Dept. of Mechanical and Aerospace Engineering, Princeton University
{mtho, afarid, ani.majumdar}@princeton.edu

Abstract. We are motivated by the problem of comparing the complexity of one robotic task relative to another. To this end, we define a notion of reduction that formalizes the following intuition: Task 1 reduces to Task 2 if we can efficiently transform any policy that solves Task 2 into a policy that solves Task 1. We further define a quantitative measure of the relative complexity between any two tasks for a given robot. We prove useful properties of our notion of reduction (e.g., reflexivity, transitivity, and antisymmetry) and relative complexity measure (e.g., nonnegativity and monotonicity). In addition, we propose practical algorithms for estimating the relative complexity measure. We illustrate our framework for comparing robotic tasks using (i) examples where one can analytically establish reductions, and (ii) reinforcement learning examples where the proposed algorithm can estimate the relative complexity between tasks.

Keywords: reduction, complexity, reinforcement learning

1 Introduction

Consider the following pairs of robotic tasks: (i) an autonomous truck driving on one side of the road vs. the other side, (ii) an autonomous car driving in one city vs. another, and (iii) a cartpole balancing the pole upright vs. downright. Intuitively, the two tasks in (i) are “as hard as each other”, while one task in (iii) (i.e., balancing upright) is more challenging than the other (i.e., balancing downright). The tasks in (ii) may be challenging in different ways and thus may not admit a straightforward ordering. How can we formally compare the complexity of each pair of tasks?

Despite decades of algorithmic advancements in robotics, we currently lack precise mathematical foundations for answering such questions. In contrast, computational complexity theory [4] provides a unifying framework and set of tools for establishing and comparing the difficulty of all computational problems. It also guides practitioners by setting expectations for the kinds of algorithms that will solve a given problem. For example, a problem that is as hard as 3-SAT will not admit a polynomial-time solution (unless P = NP); thus, a practitioner is motivated to find approximation algorithms for such a problem. There are currently no clear candidates for such a unifying theory in robotics.

Statement of Contributions. Motivated by this challenge, we take a step towards developing a precise framework for establishing the relative complexity of robotic tasks. To this end, we make five specific contributions.

* Equal contribution.
Reductions between tasks. Our key insight is to define a notion of reduction \cite[Ch. 2]{4} between robotic tasks (Definition 4). This definition formalizes the following intuition: Task 1 reduces to Task 2 if we can transform any policy that solves Task 2 into a policy that solves Task 1. The reduction is “easy” if the transformation is computationally efficient. Crucially, this notion of reduction captures the relative complexity of tasks in terms of the complexity of online task execution (i.e., how challenging the two tasks are from the perspective of the robot rather than the robot’s software designer).

Quantifying relative complexity. We propose a quantitative measure (that outputs values in the range \([0, 1]\)) for comparing the complexity of one robotic task relative to another (Definition 7). One can think of this measure as a “smoothed” version of our notion of reduction, where this notion of relative complexity can be defined for arbitrary tasks (in contrast to the definition of reduction, which captures a strict and binary notion of relative complexity).

Properties of reduction and relative complexity. We prove basic properties of our notion of reduction between tasks (e.g., reflexivity, transitivity, and antisymmetry in Proposition 1) and the relative complexity measure (e.g., non-negativity and monotonicity in Proposition 5).

Algorithm. We propose a practical algorithm based on adversarial training for estimating the relative complexity measure for robotic tasks in reinforcement learning contexts (Algorithm 1).

Examples. We demonstrate our framework using (i) illustrative examples where one can analytically establish reductions (Sec. 5), and (ii) numerical examples based on reinforcement learning problems where we apply our proposed algorithm for estimating relative complexity (Sec. 8).

2 Related Work

Complexity and robotics. The study of complexity theoretic questions in robotics has a long history. Early work \cite{28, 9} established the PSPACE-completeness of the general motion planning problem. Other results include PSPACE-hardness \cite{20, 19, 21, 11, 30} and NP-hardness \cite{26, 6, 7, 18, 17} of various planning problems. Complexity results for systems with nontrivial (e.g., nonlinear or uncertain) dynamics have also been explored in control theory \cite{1, 3}. The problem we consider in this paper is fundamentally different from the ones above. Specifically, the model of computation in the problems mentioned above assumes that the Turing machine is provided with a complete encoding of the problem on its tape at the very outset (e.g., rational numbers encoding the linear inequalities that define polytopic obstacles in the environment); the algorithm’s task is then to perform a given computation on this fixed encoding. In contrast, we are interested in the computational resources required by a robot as it is performing a given task. In this model, the robot’s sensors provide information incrementally and interactively based on its control actions and the environment.

Reductions for comparing robots and sensors. A formalism for measuring the intrinsic complexity of robotic tasks in terms of information invariants
was proposed in [12]. This framework allows one to formally compare the power of different robots via a notion of reduction. The work presented in [27] has a similar goal, but employs the notion of information spaces [22, Chapter 11] in order to handle tasks where sensors only provide partial state information. Notions of reductions for comparing the power of different sensors have also been developed [23, 13]. While our work also uses the idea of reductions, our focus is distinct from the frameworks above. In particular, our goal is to compare different robotic tasks (instead of different robots). In addition, we propose a measure of relative complexity between two tasks that goes beyond the strict notion of reduction and allows one to quantify how complex one task is with respect to another in terms of online computational resources required by the robot.

**Reductions and task complexity in learning.** Our definition of reduction between tasks formally captures the intuition that Task 2 is at least as complex as Task 1 if any solution (in the form of a control policy) for Task 2 can be transformed into a solution for Task 1. The idea of transforming solutions for one task into solutions for another has also been exploited for tackling problems in machine learning. In particular, [10, 25] propose approaches for composing previously-learned “modules” in order to speed up the process of learning on a new problem. In contrast to our work, these approaches do not seek to compare the difficulty of tasks (their goal is to obtain practical benefits in terms of sample efficiency and speed of learning). There has also been work on defining (asymmetric) notions of distance between supervised learning tasks [1, 2, 32] (e.g., using techniques from Kolmogorov complexity theory [24]). These approaches are motivated by problems in transfer learning and attempt to capture how quickly solutions from one learning problem can be fine-tuned for a new learning problem. In contrast to these measures, we seek to capture the relative complexity between tasks in terms of the complexity of computations that must be executed online (instead of the complexity of offline fine-tuning). In addition, we are motivated by robotic tasks in contrast to supervised learning problems.

### 3 Problem Formulation

**Robots, Environments, and Rewards.** Let \( p(s_t|s_{t-1}, a_{t-1}) \) describe a robot’s dynamics, where \( s_t \in S \) represents the combined state of the robot and its environment at time-step \( t \), and \( a_t \in A \) represents the action. Let \( p_0 \) denote the initial state distribution. We denote the robot’s sensor mapping as \( \sigma(a_t|s_t) \). We consider robotic tasks that are prescribed using reward functions; let \( \sum_t r(s_t, a_t) \in \mathbb{R} \) denote the cumulative reward over a given (finite or infinite) time horizon.

**Tasks.** A *task* is formally defined as a tuple \( \tau := (S, A, O, p, \sigma, r, p_0) \) which describes the state space, action space, observation space, dynamics, sensor, reward function, and the distribution over initial states. For example, a task \( \tau \) could correspond to a cartpole swinging up and balancing (with random initial conditions) or a drone navigating through random obstacle environments (where the randomness over obstacle placements is defined using \( p_0 \); recall that the state \( s_t \) encapsulates the combined state of the robot and its environment). We will
index tasks using $\xi$ and let $\mathcal{T} = \{\tau_\xi\}_\xi$ denote a set of tasks. A particular task in $\mathcal{T}$ is thus denoted as $\tau_\xi := (S_\xi, A_\xi, O_\xi, p_\xi, \sigma_\xi, r_\xi, p_{0,\xi})$.

**Policies.** Let $\pi_\xi : O_\xi \rightarrow A_\xi$ denote a policy for task $\tau_\xi$. We will denote the set of all policies (i.e., all mappings from $O_\xi$ to $A_\xi$) by $\Pi_\xi$. One can extend the formulation to policies with memory by augmenting the observation space to keep track of memory states. We will define the reward achieved by a policy on a particular task $\tau_\xi$ as:

$$R_\xi(\pi_\xi) := \min\left(\mathbb{E}_{p_{0,\xi}, \pi_\xi, \sigma} \sum_t r_\xi(s_t, \pi_\xi(o_t)), R^*_\xi\right),$$

where $R^*_\xi$ is a chosen success threshold for task $\tau_\xi$ (which forms an upper bound on $R_\xi(\pi_\xi)$). We will say that a policy $\pi_\xi$ is admissible on task $\tau_\xi$ if $R_\xi(\pi_\xi) = R^*_\xi$. We use shorthand $\pi^*_\xi$ to denote an admissible policy on task $\tau_\xi$. Let $\Pi^*_\xi \subseteq \Pi_\xi$ be the set of all admissible policies on task $\tau_\xi$.

**Task reduction and relative complexity.** Our goal is to introduce notions which meaningfully and quantitatively compare the complexity of robotic tasks. First, we aim to develop a *binary relation* (denoted by “$\preceq$”) between two tasks which will provide a notion of reduction, i.e., if task $\tau_1$ reduces to task $\tau_2$ ($\tau_1 \preceq \tau_2$), then task $\tau_2$ is at least as complex as task $\tau_1$. Second, we aim to formulate a measure $\mathcal{T} \times \mathcal{T} \rightarrow [0, 1]$ that compares the relative complexity of one task with respect to another. The goal is to not only establish if one task is more complex than another, but the *degree* to which it is more complex.

4 Task Reduction

We propose a definition of *reduction* for robotic tasks in order to formalize what it means for one task to be as hard as another. The idea of reductions comes from the theory of computational complexity; we review basic definitions in Sec. 4.1 before describing reductions between robotic tasks in Sec. 4.2.

4.1 Background

We begin with the definition of a *Turing* reduction between two computational problems. Let $A$ and $B$ be *decision problems*, i.e., problems where each instance has a yes/no answer (e.g., 3-SAT). Let $O_B$ be an oracle for decision problem $B$. This oracle is a blackbox which outputs the solution to any instance $b \in B$.

**Definition 1 (Turing Reduction [29]).** Decision problem $A$ reduces to $B$ (written $A \preceq_T B$) if we can compute the solution to all instances of $A$ using an oracle $O_B$ for $B$.

Intuitively, $A$ reduces to $B$ if one can use an oracle for $B$ as a sub-routine for solving instances of $A$, and thus $B$ is at least as complex as $A$. Note that this definition does not include reference to the complexity of the resulting function for solving $A$. We are specifically interested in *efficient* reductions. A particular notion of efficiency is formalized by *polynomial-time* reductions, as defined below.
Comparing the Complexity of Robotic Tasks

Fig. 1. The transformation of a policy for one task to another using encoders and decoders. An observation $o \in O_1$ from task $\tau_1$ is encoded to an observation $\tilde{o} \in O_2$ for $\tau_2$. The policy for $\tau_2$ outputs an action $\tilde{a} \in A_2$ which is decoded to an action $a \in A_1$ for task $\tau_1$. Together, the encoder, policy for $\tau_2$, and decoder are a policy for $\tau_1$.

Definition 2 (Cook Reduction [29]). Decision problem $A$ is polynomial-time reducible to $B$ (written $A \preceq^P_T B$) if $A \preceq_T B$ and the solution to any instance of $A$ makes only a polynomial number of calls to $O_B$.

Importantly, if $O_B$ is efficient (i.e., runs in polynomial time) and we have $A \preceq^P_T B$, then the resulting function for solving $A$ is also efficient. We aim to define a notion of reduction for robotic tasks that is inspired by the notions above for computational problems. In particular, we will leverage the core concept of using an oracle for one decision problem in order to solve another.

4.2 Task Reduction: Definition and Properties

In this section, we propose a notion of reduction between two robotic tasks and demonstrate that this defines a partial ordering on a given space $T$ of tasks. Our notion of reduction captures the following intuition: task $\tau_1$ reduces to task $\tau_2$ if we can utilize a policy (“oracle”) for $\tau_2$ in order to solve $\tau_1$. In order to formalize this, we first introduce encoders and decoders.

Definition 3 (Encoder and Decoder). Let $T$ be a space of tasks and let $\tau_1, \tau_2 \in T$ be two tasks (as defined in Sec. 3). Let $O_1, O_2$ and $A_1, A_2$ be the corresponding observation and action spaces. Let $H_{1,2}$ denote a space of functions from $O_1$ to $O_2$, and let $G_{2,1}$ denote a space of functions from $A_2$ to $A_1$. We will refer to a function $h \in H_{1,2}$ as an encoder and a function $g \in G_{2,1}$ as a decoder.

Intuitively, a task $\tau_1$ reduces to $\tau_2$ if we can utilize any admissible policy for $\tau_2$ and transform it via an encoder and decoder into a policy that solves $\tau_1$; see Fig. 1 for a visual representation of this transformation. We formalize this notion of task reduction below.

Definition 4 (Task Reduction). Task $\tau_1$ reduces to task $\tau_2$ (written $\tau_1 \preceq \tau_2$) if there exists an encoder $h \in H_{1,2}$ and a decoder $g \in G_{2,1}$ such that:

$$g \circ \pi^*_2 \circ h \in \Pi^*_1$$ (2)

for all admissible policies $\pi^*_2 \in \Pi^*_2$ for $\tau_2$.

This definition is illustrated in Fig. 1. It is important to note that the definition of reduction calls for the ability to transform any admissible policy $\pi^*_2 \in \Pi^*_2$
into an admissible policy for $\tau_1$. We also note that task reductions are conditioned on the selection of $H_{1,2}$ and $G_{2,1}$. Intuitively, this corresponds to the “complexity” of the reduction. For example, if $H_{1,2}$ and $G_{2,1}$ only include functions which can be evaluated in polynomial time, then the reduction $\tau_1 \preceq \tau_2$ is efficient (analogous to Cook reductions). Additionally, note that a robot executing the policy $g \circ \pi^*_2 \circ h \in \Pi^*_1$ for task $\tau_1$ over a time horizon of $T$ requires $T$ evaluations of $\pi^*_2$. Thus, if $H_{1,2}$ and $G_{2,1}$ only include efficiently-computable functions and $\pi^*_2$ is efficiently computable, then the online execution of $\tau_1$ is efficient. One can specify different classes of functions for $H_{1,2}$ and $G_{2,1}$ in order to capture different notions of efficiency (e.g., neural networks with bounded size).

Next, we define equivalence between two tasks.

**Definition 5 (Task Equivalence).** Tasks $\tau_1$ and $\tau_2$ are equivalent ($\tau_1 \equiv \tau_2$) if $\tau_1 \preceq \tau_2$ and $\tau_2 \preceq \tau_1$.

Note that task reduction and equivalence are both binary relations (over $\mathcal{T} \times \mathcal{T}$). As we show below, these relations satisfy properties that are intuitively desirable for any definition of reduction (or equivalence) between robotic tasks. In order to state these properties, we first introduce a notion of function closure.

**Definition 6 (Closure under Composition on $\mathcal{T}$).** Let $g_1 \in G_{2,1}$, $g_2 \in G_{3,2}$, $h_1 \in H_{1,2}$, and $h_2 \in H_{2,3}$. We say that $G_{3,1}$ and $H_{1,3}$ are closed under composition if for any $\tau_1, \tau_2, \tau_3 \in \mathcal{T}$, the following are true:

$$g_1 \circ g_2 \in G_{3,1}, \ \forall \ g_1 \in G_{2,1} \ \forall \ g_2 \in G_{3,2},$$

$$h_2 \circ h_1 \in H_{1,3}, \ \forall \ h_1 \in H_{1,2} \ \forall \ h_2 \in H_{2,3}. \ (3)$$

When $G_{\xi,\zeta}$ and $H_{\xi,\zeta}$ are the same for all tuples of tasks $(\tau_\xi, \tau_\zeta) \in \mathcal{T}^2$, this definition simplifies to closure under function composition for $G_{\xi,\zeta}$ and $H_{\xi,\zeta}$.

**Proposition 1 (Task Reduction is a Non-Strict Partial Ordering Relation).** Suppose that $\forall \ (\tau_\xi, \tau_\zeta) \in \mathcal{T}^2$, $H_{\xi,\zeta}$ and $G_{\xi,\zeta}$ include the identity and are closed under composition on $\mathcal{T}$. Then, task reductions satisfy the following properties and thus define a non-strict partial ordering relation.

**Property 1.a.** Reflexivity: $\tau_1 \preceq \tau_1$.

**Property 1.b.** Antisymmetry: $\tau_1 \prec \tau_2 \implies \neg(\tau_2 \preceq \tau_1)$, where $\tau_1 \prec \tau_2$ is defined as $(\tau_1 \preceq \tau_2) \land \neg(\tau_1 \equiv \tau_2)$.

**Property 1.c.** Transitivity: $(\tau_1 \preceq \tau_2) \land (\tau_2 \preceq \tau_3) \implies \tau_1 \preceq \tau_3$.

**Proof.** Property 1.a: $\tau_1 \preceq \tau_1 \implies \exists \ g \in G_{1,1}, \ h \in H_{1,1}$ such that

$$g \circ \pi^*_1 \circ h \in \Pi^*_1. \ (5)$$

If $g$ and $h$ are the identity function, then $g \circ \pi^*_1 \circ h = \pi^*_1 \ \forall \ \pi^*_1 \in \Pi^*_1$. Thus, $\tau_1 \preceq \tau_1$ when $G_{1,1}$ and $H_{1,1}$ include their respective identity functions.
Comparing the Complexity of Robotic Tasks

Property 1a. Suppose \( \tau_1 < \tau_2 \) and thus \((\tau_1 \leq \tau_2) \land \neg(\tau_1 \equiv \tau_2) \). Note that \( \neg(\tau_1 \equiv \tau_2) \implies \neg((\tau_1 \leq \tau_2) \land (\tau_2 \leq \tau_1)) \implies \neg(\tau_1 \leq \tau_2) \lor \neg(\tau_2 \leq \tau_1) \). We assumed \((\tau_1 \leq \tau_2)\), so we must have that \(\neg(\tau_2 \leq \tau_1)\).

Property 1b. Suppose \( \tau_1 \leq \tau_2 \) and \( \tau_2 \leq \tau_3 \). By Definition 4, \( \exists g_1 \in G_{2,1}, g_2 \in G_{3,2}, h_1 \in H_{1,2}, \) and \( h_2 \in H_{2,3} \) such that \( g_1 \circ \pi^*_3 \circ h_1 \in \Pi^*_1 \lor \pi^*_2 \in \Pi^*_2 \) and \( g_2 \circ \pi^*_3 \circ h_2 \in \Pi^*_2 \lor \pi^*_3 \in \Pi^*_3 \). Consider\
\[
\begin{array}{c}
g_1 \circ g_2 \circ \pi^*_3 \circ h_2 \circ h_1 \\
\end{array}
\]
for all \( \pi^*_3 \in \Pi^*_3 \). Let \( g_3 := g_1 \circ g_2 \) and \( h_3 := h_2 \circ h_1 \) so that \( g_3 \circ \pi^*_3 \circ h_3 \in \Pi^*_1 \) for all \( \pi^*_3 \in \Pi^*_3 \). If \( G_{3,1} \) and \( H_{1,3} \) are closed under composition on \( T \), then \( g_3 \in G_{3,1} \) and \( h_3 \in H_{1,3} \) and \( \tau_1 \leq \tau_3 \). Thus, task reductions are transitive if \( G_{3,1} \) and \( H_{1,3} \) are closed under composition on \( T \).

Additionally, note that strict task reduction \( \tau_1 < \tau_2 \) is a strict partial ordering relation. We provide a complete proof of this in Appendix A. These properties formalize a series of intuitions about task reductions. Task reduction consists of transforming an admissible policy from one task to another. In the case of reflexivity, intuition suggest that if the tasks are the same, no transformation will be required for the policy to work. Thus, \( G \) and \( H \) must include the identity to allow for this. Antisymmetry establishes a clear sense of “dominance” between tasks in terms of complexity — if a task \( \tau_1 \) strictly reduces to another task \( \tau_2 \), then \( \tau_2 \) should not reduce to \( \tau_1 \). Building on antisymmetry, we also require transitivity to make a clear chain of increasingly complex tasks. A series of tasks which reduce to each other cyclically (i.e., \( \tau_1 \leq \tau_2 \land \tau_2 \leq \tau_3 \land \tau_3 \leq \tau_1 \)) should only be possible if all the tasks are equivalent. The following proposition proves useful properties of task equivalence.

Proposition 2 (Task Equivalence is an Equivalence Relation). Suppose that \( \forall (\tau_x, \tau_z) \in T^2, H_{\xi, \xi} \) and \( G_{\xi, \xi} \) include the identity and are closed under composition on \( T \). Then, task equivalence satisfies the following properties and thus defines an equivalence relation.

Property 2a. Reflexivity: \( \tau_1 \equiv \tau_1 \).

Property 2b. Symmetry: \( \tau_1 \equiv \tau_2 \implies \tau_2 \equiv \tau_1 \).

Property 2c. Transitivity: \( \tau_1 \equiv \tau_2 \land \tau_2 \equiv \tau_3 \implies \tau_1 \equiv \tau_3 \).

Proof. A complete proof of these properties is provided in Appendix A.

5 Examples of Reductions

In this section, we provide two concrete examples of task reductions (Definition 4) and equivalence (Definition 5). We present an efficient reduction between navigation tasks with differing goal locations in an environment which has 90 degree rotational symmetry. We then show reduction between a set of tasks (i.e., an equivalence class of tasks) for grasping with varying camera viewpoints.
5.1 Navigation to a Goal with Map Rotations

Consider a robot which must navigate to a goal location and avoid obstacles. Suppose that the robot lives on a $[-n, n] \times [-n, n] \in \mathbb{R}^2$ world and can move a distance $d$ in a cardinal direction $\{N, E, S, W\}$ by taking action $(d, \{0, 1, 2, 3\})$, where $0, 1, 2, 3$ correspond to N, E, S, W. Additionally, for all tasks in this setting, assume that $m$ circular obstacles are randomly placed in the environment (while always allowing for a path to the goal for the robot). Suppose that the robot’s observation at time $t$ corresponds to a complete map of the grid-world which consists of a list of locations including obstacle locations, the goal location, and the robot location. Let $o_i^E = (x_i, y_i)$ correspond to the location of the $i$th observation in $o^E$. The robot is initialized in a random position and its goal is to navigate to the goal location. The robot receives a reward of 1 if it successfully navigates to the goal and a reward of 0 otherwise.

Let task $\tau_N$ ("north") have goal location at $(0, n)$ and $\tau_E$ ("east") have goal location $(n, 0)$. An admissible policy $\pi^*_N$ on $\tau_N$ will always successfully navigate to the goal location $(0, n)$ (regardless of the initial state of the robot and the locations of the obstacles).

**Proposition 3** ($\tau_E \preceq \tau_N$). Let $G_{E,N}$ contain functions that can perform addition modulo 4 and $H_{E,N}$ contain functions which can be evaluated in linear time (in the size of the observation). Then $\tau_E \preceq \tau_N$.

**Proof.** For any observation $o^E$ from task $\tau_E$, let $h^E_{EN}(o^E_{i}) := (y^i, -x^i) \land i$. For any action $a^N = (d, i)$ from $\pi^*_N$, let $g_{NE}(d, i) := (d, i + 1 \mod 4)$. Then we have that $g_{NE} \circ \pi^*_N \circ h^E_{EN} \in \Pi^*_N$. $\square$

Intuitively, $h^E_{EN}$ rotates observations from $\tau_E$ so that they look exactly like a corresponding observation from $\tau_N$. Then, any admissible policy $\pi^*_N$ must find an admissible action (i.e., one which ultimately results in solving the task) based on the rotated observation. The decoder $g_{NE}$ then transforms the output action to the corresponding one required for task $\tau_E$. With analogous constructions, we can show that $\tau_N \equiv \tau_E \equiv \tau_S \equiv \tau_E$.

5.2 Grasping Objects with Differing Camera Viewpoints

Consider a robotic arm which must grasp one of a set of known objects using an RGB-D image. A randomly-selected object is placed with a random pose on a table (which is at height $z = 0$). A task $\tau \in \mathcal{T}$ will correspond to this grasping challenge when the RGB-D camera is placed at a particular viewpoint. Assume that the camera in each task is always pointed at the center of the table (i.e., $(0, 0, 0)$). Additionally, we will require that for each of the possible objects, the identity of the particular object and its pose are uniquely determinable from the camera. Further assume that at the beginning of the task, the robot arm does not occlude the camera’s view of the object. We treat this as a single time-step task; the robot selects a grasp pose based on the camera observation. A successful grasp results in a reward of 1 and 0 otherwise. Let $\tau_{(\theta, \phi)}$ correspond to the grasping task with the camera located at spherical coordinates $(1, \theta, \phi)$. 
Comparing the Complexity of Robotic Tasks

Proposition 4 ($\tau_{(\theta_1, \phi_1)} \preceq \tau_{(\theta_2, \phi_2)}$). For any $(\theta_1, \phi_1)$ and $(\theta_2, \phi_2)$, task $\tau_{(\theta_1, \phi_1)}$ reduces to task $\tau_{(\theta_2, \phi_2)}$ if $(\theta_1, \phi_1)$ and $(\theta_2, \phi_2)$ are selected such that the cameras can view the entire table from either viewpoint.

Proof. Let $g$ be the identity. Assume that the camera placed at $(\theta_1, \phi_1)$ can view the entire table. Let $h$ take as input a RGB-D image from viewpoint $(1, \theta_1, \phi_1)$ and output the same environmental setup from the perspective of a camera placed at $(1, \theta_2, \phi_2)$. Then $g \circ \pi^*_2 \circ h \in \Pi^*_2(\theta_1, \phi_1)$.

Note that the function $h$ requires a model of the known objects and potentially a simulation of the environment in order to generate the image from a differing perspective. As such, the reduction with this encoder may not be efficient. A direct consequence of this proposition is that $\tau_{(\theta_1, \phi_1)} \equiv \tau_{(\theta_2, \phi_2)}$ if $(\theta_1, \phi_1)$ and $(\theta_2, \phi_2)$ are selected such that the cameras can view the entire table. Let $\Theta$ be the set of all $(\theta, \phi)$ values which reduce to each other using the reduction provided in the proposition. The corresponding set of tasks $T_{\Theta} \subseteq T$ defines a class of tasks which are all equivalently complex.

6 Relative Complexity

In general, it may be difficult to establish a reduction between two completely distinct tasks such as grasping an object and avoiding obstacles; indeed an arbitrary pair of tasks is unlikely to satisfy the notion of reduction introduced in Sec. 4.2. Additionally, our definition of reduction is a binary one and does not capture how complex a particular task is relative to another. Motivated by these observations, we propose a definition of relative complexity that captures the degree to which one task is more complex than another. This quantity can be thought of as a “continuous” or “smoothed” version of task reductions (in a precise sense, which we elucidate below). Note that we will omit the subscripts on $G$ and $H$ when it is clear which tasks they transform between.

Definition 7 (Relative Complexity). The relative complexity of task $\tau_1$ with respect to task $\tau_2$ is

$$C_{\tau_1/\tau_2} := \sup_{\pi^*_2 \in \Pi^*_2} \inf_{h \in H, g \in G} \left[ 1 - \frac{R_1(g \circ \pi^*_2 \circ h)}{R^*_1} \right],$$

where we assume that rewards are nonnegative. We use the notation $C_{\tau_1/\tau_2}(H, G)$ when we want to highlight dependence on $H$ and $G$.

Intuitively, if the relative complexity $C_{\tau_1/\tau_2} = 0$, this means that for any $\pi^*_2 \in \Pi^*_2$ we can find an encoder $h$ and a decoder $g$ such that $R_1(g \circ \pi^*_2 \circ h) = R^*_1$. Hence, $\tau_1$ is no more complex than $\tau_2$ in this case. If $C_{\tau_1/\tau_2} > 0$, then an admissible policy $\pi^*_2 \in \Pi^*_2$ may not be transformed into an admissible policy for $\tau_1$ using encoders and decoders in $H$ and $G$. A key advantage of this definition is that we can compare any two tasks and quantify the relative complexity of one with respect to another. As with our previous definitions, we prove a set of useful properties below. Importantly, Properties 5.c and 5.d will establish a link between the relative complexity $C_{\tau_1/\tau_2}$ and task reduction ($\tau_1 \preceq \tau_2$).
Proposition 5 (Properties of the Relative Complexity). Relative Complexity satisfies the following properties:

Property 5.a. Nonnegativity and boundedness: \( C_{\tau_1/\tau_2} \in [0, 1] \).

Property 5.b. Monotonicity with respect to \( H \) and \( G \): If \( H \subseteq H' \) and \( G \subseteq G' \), then \( C_{\tau_1/\tau_2}(H', G') \leq C_{\tau_1/\tau_2}(H, G) \).

Property 5.c. Equivalence between reduction and 0 relative complexity: \( C_{\tau_1/\tau_2} = 0 \iff \tau_1 \preceq \tau_2 \).

Property 5.d. Equivalence between no reduction and positive relative complexity: \( C_{\tau_1/\tau_2} > 0 \iff \neg(\tau_1 \preceq \tau_2) \).

Proof. A complete proof of these properties is provided in Appendix A.

7 Algorithmic Approach

We can frame the optimization problem in (7) as a game: an adversary chooses an admissible policy \( \pi_2^* \in \Pi_2^* \) which maximizes the relative complexity, and then the player chooses \( g \in G \) and \( h \in H \) such that the relative complexity is minimized. It may be difficult in general to find the solution to this game. However, we can still compute a meaningful estimate via approximate methods.

There are multiple ways to approximate the solutions to zero-sum games such as the one presented in (7); see e.g., [14]. One strategy is to use best-response dynamics, where the players update their strategies in rounds based on the best response to the opponent’s choice. We parameterize the policies, encoders, and decoders with neural networks. Thus, we aim to develop a method which uses gradient steps to approximate the solution to (7). A technique which exploits best-response dynamics is the update rule for training generative adversarial networks (GANs) [15], where gradient steps update the players’ strategies based on batches of data. The algorithm we present has a similar structure.

The resulting approach has two steps and is presented in Algorithm 1. The first step is to update the policy \( \pi_2 \) to maximize the relative complexity. However, \( \pi_2 \) must also eventually succeed on task \( \tau_2 \). Thus, the update for \( \pi_2 \) has two terms: one to train \( \pi_2 \) to succeed on \( \tau_2 \) and one to train \( g \circ \pi_2 \circ h \) to fail on \( \tau_1 \). We scale the latter with the “adversarial tuning parameter” \( \alpha \) which allows weighting of the terms relative to each other. The second step is to update the encoder and decoder to minimize the relative complexity. We present Algorithm 1 with general loss functions \( L_1, L_2 \) on tasks \( \tau_1, \tau_2 \) to allow for specialization to reinforcement learning techniques (e.g., Q-learning). An example for \( L_1 \) and \( L_2 \) based on best-response dynamics is \( L_2(\pi_2) := -R_2(\pi_2) \) and \( L_1(g \circ \pi_2 \circ h) := 1 - R_1(g \circ \pi_2 \circ h)/R_1^* \) since these directly capture the objectives of each player.

In practice, when \( \alpha \) is too low, the policy \( \pi_2 \) is not adversarial enough and may not prevent \( g \circ \pi_2 \circ h \) from succeeding on task \( \tau_1 \). When \( \alpha \) is too large, the policy \( \pi_2 \) may not ever succeed on task \( \tau_2 \). As such we increase \( \alpha \) as much as possible while ensuring \( \pi_2 \) is admissible at the end of training in order to provide an estimate of (7). After training is complete, we have \( \pi_2^*, g, \) and \( h \) which we use to compute the approximate relative complexity \( \tilde{C} = 1 - R_1(g \circ \pi_2^* \circ h)/R_1^* \).
Algorithm 1 Approximating Relative Complexity

1: Input: Threshold for success $R_1^\star, R_2^\star$ on $\tau_1, \tau_2$ respectively
2: Input: Learning rates $\lambda_1, \lambda_2$, adversarial tuning parameter $\alpha$
3: Input: Function spaces $H, G$
4: Input: Loss functions $L_1, L_2$ for $\tau_1, \tau_2$ respectively
5: Output: Approximate relative complexity $\tilde{C}_{\tau_1/\tau_2} \approx C_{\tau_1/\tau_2}$
6: while $\neg (\text{converged} \land R_2(\pi_2) = R_2^\star)$ do
7:   Step 1: $\pi_2$ update
8:   $c_1 \leftarrow L_2(\pi_2) - \alpha L_1(g \circ \pi_2 \circ h)$
9:   $\pi_2 \leftarrow \pi_2 - \lambda_1 \nabla_{\pi_2} c_1$
10: Step 2: encoder/decoder update
11:   $c_2 \leftarrow L_1(g \circ \pi_2 \circ h)$
12:   $[h, g] \leftarrow [h, g] - \lambda_2 \nabla_{[h, g]} c_2$
13: end while
14: $\tilde{C}_{\tau_1/\tau_2} \leftarrow \left[ 1 - \frac{R_1(\pi_2 \circ h)}{R_1^\star} \right]$

8 Examples

We implement Algorithm 1 on two reinforcement learning examples using Q-learning and Soft Actor-Critic (SAC). On the OpenAI Gym Cartpole, we compare the complexity of balancing the pole upright relative to balancing it downwards. Using the Mujoco 2D walker, we compare the complexity of walking at various speeds. We demonstrate that our estimates of relative complexity correspond to intuitive notions of complexity for these tasks.

8.1 Cartpole Balancing Task

Overview. We use the OpenAI Gym Cartpole environment to define two tasks: balancing the cart’s pole against gravity $\tau_\uparrow$ (at the unstable equilibrium) and balancing the cart’s pole with gravity $\tau_\downarrow$ (at the stable equilibrium). Equivalently, one can think of $\tau_\uparrow$ and $\tau_\downarrow$ being specified by the direction in which gravity acts ($-y$ and $+y$). The initial state of the system is randomized close to the equilibrium for each task. The policy receives the system’s state vector as input and can apply forces on the cart using three actions: {no force, push left, push right}. A task runs for 200 time steps and the policy receives a reward of 1 for each time step that the pole stays balanced (within $24^\circ$ of the equilibrium).

Note that since we only switch the direction of gravity, the states which achieve a reward of 1 are consistent between tasks. A reward of 0 is given if the pole falls beyond $24^\circ$ of the equilibrium or the cart strays too far from the start position; the trial is then stopped. A policy for either task is admissible if it successfully balances the pole for the entire 200 time-step trial, i.e., $R_1^\star = R_2^\star = 200$.

Policy, Encoder, Decoder, and Training. The policy, encoder, and decoder all consist of few-layer neural networks. We apply Algorithm 1 using Q-learning to approximate the relative complexities $C_{\tau_1/\tau_\downarrow}$ and $C_{\tau_\downarrow/\tau_1}$. Note that we can directly apply Q-learning to Algorithm 1 by letting the loss functions $L_1$ and $L_2$ correspond to a Q-learning loss (see Appendix B for further details).
The relative complexity of task $\tau_\uparrow$ with respect to $\tau_\downarrow$ is consistently close to 1 when $\alpha$ is large enough ($\geq 10$). In contrast, the relative complexity of $\tau_\downarrow$ with respect to $\tau_\uparrow$ is consistently close to 0 regardless of the choice of $\alpha$. The plots suggest that $\tau_\uparrow$ is more complex than $\tau_\downarrow$. We plot the mean and standard deviation (shaded region) across 5 seeds for each $\alpha$. In our experiments, we vary the adversarial tuning parameter to examine its effect and also vary the number of layers in encoder $h$ and decoder $g$ to approximate the relative complexity given varying complexity of $H$ and $G$. The result of training is an admissible policy $\pi^*$, encoder $g$, and decoder $h$; these allow us to compute the estimated relative complexity $\tilde{C}$.

**Results: Sensitivity to Adversarial Tuning Parameter.** We compute the approximate relative complexity for a wide range of the adversarial tuning parameter $\alpha$: $[10^{-5}, 10^5]$, and plot the results in Fig. 2. We choose $g$ to be a neural network with a single hidden layer and $h$ to have two hidden layers. When approximating the relative complexity of task $\tau_\uparrow$ with respect to $\tau_\downarrow$ we see that when $\alpha$ is large enough, the value settles at approximately 0.8. This suggests that $-(\tau_\uparrow \preceq \tau_\downarrow)$ by Property 5.d. Additionally, note that a decrease in $\alpha$ corresponds to decreased weight on minimizing $R_\pi(g \circ \pi_\downarrow \circ h)$. As such, when $\alpha$ is too small, $\pi_\downarrow$ cannot reliably ensure that $g \circ \pi_\downarrow \circ h$ fails on $\tau_\uparrow$. In the case of approximating the relative complexity of task $\tau_\downarrow$ with respect to $\tau_\uparrow$, we see that for all values of $\alpha$, a low relative complexity is achieved. The plot suggest that $\tau_\downarrow \preceq \tau_\uparrow$ by Property 5.c. This suggests that the task of balancing at a stable equilibrium $\tau_\downarrow$ is less complex than the task of balancing at an unstable equilibrium $\tau_\uparrow$, which is consistent with our intuition.

**Results: Sensitivity to $H$ and $G$ Model Complexity.** We also approximate the relative complexity for different $H$ and $G$ with varying model complexity (achieved by varying the number of hidden layers). In all experiments, the neural network architecture for the (inner) policy $\pi$ is kept consistent. When varying the complexity of $H$, we choose $g$ to be a neural network with a single hidden layer; when varying $G$, we choose $h$ to be a neural network with two hidden layers. The results are plotted in Fig. 3. We see that there is no significant change in the approximate relative complexity when $H$ increases in complexi-
Comparing the Complexity of Robotic Tasks

Fig. 3. The approximate relative complexity for varied $H$ and $G$ complexity. (Top) There is little or no change in the relative complexity for varied $H$. (Bottom) We see significant decrease in $C_{\tau/\tau}$, for increasing $G$ complexity. This suggest that reduction $\tau_1 \leq \tau$ is not possible when $G$ includes only linear neural networks. We plot the mean of 5 trials for each $\alpha$ value and the standard deviation is shaded.

ity. However, when the complexity of $G$ is increased, we see a clear decrease in $\tilde{C}_{\tau/\tau}$, indicating that (i) $\neg (\tau_1 \leq \tau)$ when $G$ only contains the identity or a single linear layer, and (ii) $\tau_1 \leq \tau$ for more complex $G$. We see a slight decrease in $\tilde{C}_{\tau/\tau}$ for increasing complexity of $G$. By Property 5.b, the relative complexity is monotonic with respect to $H$ and $G$; specifically, increasing the complexity of $H$ and $G$ should result in a monotonic decrease in the relative complexity. This is consistent with the results in Fig. 3.

8.2 Mujoco 2D Walker at varied speeds

Overview. Using the Mujoco [31] Walker2D-v2 environment in OpenAI Gym [8], we create a set of tasks $\tau_v$ with the goal of walking at a particular speed $v$ (from 0.6 m/s to 1.4 m/s). Thus, the maximum reward on $\tau_v$ is achieved when the robot travels at $v$. The policy receives a 17-dimensional observation vector (corresponding to joint angles and velocities) and outputs a 6-dimensional action vector (corresponding to joint torques) to control the 2D walking robot. The task runs for at most 1000 time steps. The reward at time $t$ is $1 - |v_t - v|$, where $v_t$ is the speed at the current time step. Note that to help with training, we also provide a reward for staying upright and a reward which penalizes large policy outputs to help with training. When evaluating $\tilde{C}$, we lower-bound the reward by 0 to ensure that the reward is non-negative. If the robot falls over, a reward of 0 is given and the trial stops. Since the challenge of maintaining a particular speed may differ between tasks, the threshold for success may vary. Thus, to find $R^*_v$, we first train an individual policy to travel at speed $v$. We then let $R^*_v = 0.95 \times$ (individual policy reward on $\tau_v$).
14 M. Ho, A. Farid, and A. Majumdar

Fig. 4. Reduction from the task of walking at speed $v_1$ ($\tau_{v_1}$) to walking at $1.0$ m/s ($\tau_{1.0}$). The estimated relative complexity $\tilde{C}_{\tau_{v_1}/\tau_{1.0}}$ is larger when $v_1 < 1.0$ m/s as compared with $v_1 \geq 1.0$ m/s. Thus, a policy for walking at $1.0$ m/s cannot directly be transformed to a policy for walking at $0.6$ or $0.8$ m/s. The plot suggests that $\tau_{v_1} \leq \tau_{1.0}$ for $v_1 \in \{1.0, 1.2, 1.4\}$ m/s but $\neg(\tau_{v_1} \leq \tau_{1.0})$ for $v_1 \in \{0.6, 0.8\}$. We plot the mean over 15 seeds for each $v_1$; the standard deviation is shaded.

Policy, Encoder, Decoder and Training. As in Sec. 8.1, the policy, encoder, and decoder all consist of neural networks with 3 hidden layers. However, since the action space is continuous and 6-dimensional, we use soft actor-critic (SAC) [16]. This requires a modification to Algorithm 1 for training critics for $\tau_{v_1}$ and $\tau_{v_2}$ (see Appendix B for the algorithm and additional experimental details). The result of training is an admissible policy $\pi^*_2$ and the encoder/decoder $h/g$; these are used to calculate the approximate relative complexity $\tilde{C}$.

Results: Relative Complexity for Increasing Speed. We compute the approximate relative complexity of the task of walking at speeds $v_1 \in [0.6, 1.4]$ m/s with respect to the task of walking at $1.0$ m/s ($\tau_{1.0}$). We sweep through a wide range of the adversarial tuning parameter $\alpha = [10^{-6}, 10]$, and for each $\tilde{C}$ computation, choose the largest $\alpha$ which results in an admissible policy on task $\tau_{1.0}$. The results are presented in Fig. 4. We see that $\tilde{C}_{\tau_{v_1}/\tau_{1.0}}$ is larger when $v_1 < 1.0$ m/s as compared with $v_1 \geq 1.0$ m/s. This may be due to a change in walking gait between speeds $0.8$ and $1.0$ m/s. Intuitively, this suggests that there exists an admissible policy for walking at $1.0$ m/s which cannot be directly transformed to walk at $0.6$ or $0.8$ m/s but no such policy exists which prevents a transformation for walking faster. Additionally these results indicate that $\tau_{v_1} \leq \tau_{1.0}$ for $v_1 \in \{1.0, 1.2, 1.4\}$ m/s but $\neg(\tau_{v_1} \leq \tau_{1.0})$ for $v_1 \in \{0.6, 0.8\}$.

9 Conclusion

We have presented a framework for comparing the complexity of robotic tasks. In order to achieve this, we defined a notion of reduction between two tasks that captures the ability of a robot to solve one task given a policy for another. We also presented a measure of relative complexity that quantifies how complex one task is relative to another. Our theoretical results establish basic properties satisfied by these notions and also establish the relationship between reductions and relative complexity. We presented a practical algorithm for estimating the
relative complexity between tasks and demonstrated this using reinforcement learning tasks. Our results demonstrate consistency with intuitive notions of hardness for these tasks and empirical correspondence to theoretical properties.

**Future work.** The work presented here opens up a number of exciting directions for future research. On the theoretical front, it would be interesting to devise more general notions of reductions than the one presented here, e.g., allowing the policy from one task to be used as an oracle to solve another task in a more general manner (instead of the specific encoder-policy-decoder architecture we utilize). Another research direction is to use our notion of reduction to establish equivalences between broad classes of tasks (similar to complexity classes P and NP in computational complexity). On the algorithmic front, it would be of practical interest to develop different techniques for estimating the relative complexity. One could potentially use other algorithms (beyond best-response dynamics) for approximately solving games [14]. Finally, a particularly exciting direction would be to draw inspiration from *cryptography* and turn statements about the complexity of certain tasks into statements about robustness for a robot. This could potentially be achieved by establishing the hardness of an adversary’s task of foiling our robot.

**Acknowledgements**

Funding: NSF CAREER Award [#2044149] and ONR [N00014-21-1-2803].

**Bibliography**

[1] Achille, A., Mbeng, G., Soatto, S.: Dynamics and reachability of learning tasks. arXiv: 1810.02440 (2019)
[2] Achille, A., Paolini, G., Mbeng, G., Soatto, S.: The information complexity of learning tasks, their structure and their distance. arXiv: 1904.03292 (2020)
[3] Ahmadi, A.A., Majumdar, A., Tedrake, R.: Complexity of ten decision problems in continuous time dynamical systems. In: Proceedings of the American Control Conference (ACC). pp. 6376–6381 (2013)
[4] Arora, S., Barak, B.: Computational Complexity: A Modern Approach. Cambridge University Press (2009)
[5] Blondel, V., Tsitsiklis, J.: A survey of computational complexity results in systems and control. Automatica 36(9), 1249–1274 (2000)
[6] Borie, R., Tovey, C., Koenig, S.: Algorithms and complexity results for pursuit-evasion problems. In: International Joint Conference on Artificial Intelligence (IJCAI). vol. 9, pp. 59–66 (2009)
[7] Borie, R., Tovey, C., Koenig, S.: Algorithms and complexity results for graph-based pursuit evasion. Autonomous Robots 31(4), 317–332 (2011)
[8] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym. arXiv preprint arXiv:1606.01540 (2016)
[9] Canny, J.: The Complexity of Robot Motion Planning. MIT press (1988)
[10] Chang, M.B., Gupta, A., Levine, S., Griffiths, T.L.: Automatically composing representation transformations as a means for generalization. arXiv:1807.04640 (2019)
[11] Culberson, J.: Sokoban is PSPACE-complete (1997)
[12] Donald, B.R.: On information invariants in robotics. Artificial Intelligence 72(1), 217–304 (Jan 1995)
[13] Erdmann, M.: Understanding action and sensing by designing action-based sensors. International Journal of Robotics Research 14(5), 483–509 (1995)
[14] Fudenberg, D., Drew, F., Levine, D.K., Levine, D.K.: The theory of learning in games, vol. 2. MIT press (1998)
[15] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. arXiv: 1406.2661 (2014)
[16] Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Proceedings of the International Conference on Machine Learning. pp. 1861–1870 (2018)
[17] Han, S., Stiffler, N., Krontiris, A., Bekris, K., Yu, J.: High-quality tabletop rearrangement with overhand grasps: Hardness results and fast methods. arXiv preprint arXiv:1705.09180 (2017)
[18] Hauser, K.: The minimum constraint removal problem with three robotics applications. The International Journal of Robotics Research (IJRR) 33(1), 5–17 (2014)
[19] Hopcroft, J., Joseph, D., Whitesides, S.: Movement problems for 2-dimensional linkages. SIAM Journal on Computing 13(3), 610–629 (1984)
[20] Hopcroft, J., Schwartz, J., Sharir, M.: On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the Warehouseman’s Problem. The International Journal of Robotics Research (IJRR) 3(4), 76–88 (1984)
[21] Joseph, D., Plantings, W.H.: On the complexity of reachability and motion planning questions. In: Proceedings of the First Annual Symposium on Computational Geometry. pp. 62–66. ACM (1985)
[22] LaValle, S.: Planning Algorithms. Cambridge University Press (2006)
[23] LaValle, S.M.: Sensing and Filtering: A Fresh Perspective Based on Preimages and Information Spaces. Citeseer (2012)
[24] Li, M., Vitányi, P., et al.: An introduction to Kolmogorov complexity and its applications, vol. 3. Springer (2008)
[25] Li, Y., Wu, Y., Xu, H., Wang, X., Wu, Y.: Solving compositional reinforcement learning problems via task reduction. arXiv:2103.07607 (2021)
[26] Murrieta-Cid, R., Monroy, R., Hutchinson, S., Laumond, J.P.: A complexity result for the pursuit-evasion game of maintaining visibility of a moving evader. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). pp. 2657–2664 (2008)
[27] O’Kane, J.M., LaValle, S.M.: Comparing the Power of Robots. The International Journal of Robotics Research 27(1), 5–23 (Jan 2008)
[28] Reif, J.: Complexity of the mover’s problem and generalizations. In: 20th Annual Symposium on Foundations of Computer Science. pp. 421–427. IEEE (1979)
[29] Sipser, M.: Introduction to the theory of computation. ACM (1996)
[30] Solovey, K., Halperin, D.: On the hardness of unlabeled multi-robot motion planning. The International Journal of Robotics Research (IJRR) 35(14), 1750–1759 (2016)
[31] Todorov, E., Erez, T., Tassa, Y.: Mujoco: A physics engine for model-based control. In: International Conference on Intelligent Robots and Systems. pp. 5026–5033 (2012)
[32] Tran, A.T., Nguyen, C.V., Hassner, T.: Transferability and hardness of supervised classification tasks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (October 2019)
Appendix

A Proof of Properties

Proposition 6 (Strict Task Reduction is a Strict Partial Ordering Relation). Suppose that \( \forall (\tau_1, \tau_2) \in \mathcal{T}^2 \), \( H_{\xi, \xi} \) and \( G_{\zeta, \zeta} \) include the identity and are closed under composition on \( \mathcal{T} \). Then, strict task reductions satisfy the following properties and thus define a strict partial ordering relation.

Property 6.a. Irreflexivity: \( \neg (\tau_1 \prec \tau_1) \).
Property 6.b. Asymmetry: \( \tau_1 \prec \tau_2 \implies \neg (\tau_2 \prec \tau_1) \).
Property 6.c. Transitivity: \( \tau_1 \prec \tau_2 \land \tau_2 \prec \tau_3 \implies \tau_1 \prec \tau_3 \).

Proof. Property 6.a: Suppose \( \tau_1 \prec \tau_1 \implies \tau_1 \not\equiv \tau_1 \land \neg (\tau_1 \equiv \tau_1) \). \( \tau_1 \equiv \tau_1 \) by Property 2.b when \( H_{1,1} \) and \( G_{1,1} \) include their respective identity functions.

Property 6.b: \( \tau_1 \prec \tau_2 \implies \neg (\tau_2 \not\equiv \tau_1) \) by Property 1.a since \( \tau_1 \prec \tau_2 \implies \neg (\tau_2 \not\equiv \tau_1) \). \( \neg (\tau_2 \equiv \tau_1) \iff \neg (\tau_2 \equiv \tau_1) \lor (\tau_2 \equiv \tau_1) \iff \neg (\tau_2 \equiv \tau_1) \implies \neg (\tau_2 \prec \tau_1) \).

Property 6.c: \( \tau_1 \prec \tau_2 \land \tau_2 \prec \tau_3 \implies \tau_1 \not\equiv \tau_2 \land \tau_2 \equiv \tau_3 \land \neg (\tau_1 \equiv \tau_2) \land \neg (\tau_2 \equiv \tau_3) \) \( \tau_1 \equiv \tau_2 \) by Properties 1.a and 2.b \( \Rightarrow \tau_1 \prec \tau_3 \) when \( H_{1,3} \) and \( G_{3,1} \) are closed under composition on \( \mathcal{T} \).

Proposition 2 (Task Equivalence is an Equivalence Relation). Suppose that \( \forall (\tau_1, \tau_2) \in \mathcal{T} \), \( H_{\xi, \xi} \) and \( G_{\zeta, \zeta} \) include the identity and are closed under composition on \( \mathcal{T} \). Then, task equivalence satisfies the following properties and thus defines an equivalence relation.

Property 2.a. Reflexivity: \( \tau_1 \equiv \tau_1 \).
Property 2.b. Symmetry: \( \tau_1 \equiv \tau_2 \implies \tau_2 \equiv \tau_1 \).
Property 2.c. Transitivity: \( \tau_1 \equiv \tau_2 \land \tau_2 \equiv \tau_3 \implies \tau_1 \equiv \tau_3 \).

Proof. Property 2.a: \( \tau_1 \equiv \tau_1 \) by Property 1.a when \( G_{1,1} \) and \( H_{1,1} \) include the identity. Thus, task equivalence is reflexive if \( G_{1,1} \) and \( H_{1,1} \) include the identity.

Property 2.b: \( \tau_1 \equiv \tau_2 \implies (\tau_1 \not\equiv \tau_2) \land (\tau_2 \not\equiv \tau_1) \) by Definition 5 \( \Rightarrow (\tau_2 \not\equiv \tau_1) \land (\tau_1 \not\equiv \tau_2) \Rightarrow \tau_2 \equiv \tau_1 \).

Property 2.c: \( \tau_1 \equiv \tau_2 \land (\tau_2 \equiv \tau_3) \Rightarrow (\tau_1 \not\equiv \tau_2) \land (\tau_2 \not\equiv \tau_3) \land (\tau_2 \equiv \tau_3) \) \( \tau_1 \equiv \tau_2 \) \( \Rightarrow (\tau_3 \not\equiv \tau_1) \) by Property 1.a when \( G_{3,1} \) and \( H_{1,3} \) are closed under composition on \( \mathcal{T} \). Similarly, \( (\tau_2 \not\equiv \tau_2) \land (\tau_2 \not\equiv \tau_3) \Rightarrow (\tau_2 \equiv \tau_3) \). Thus \( (\tau_1 \equiv \tau_2) \land (\tau_3 \equiv \tau_1) \Rightarrow \tau_1 \equiv \tau_3 \). Thus task equivalence is transitive if \( G_{3,1} \) and \( H_{1,3} \) are closed under composition on \( \mathcal{T} \).

Proposition 5 (Properties of the Relative Complexity). Relative Complexity satisfies the following properties:

Property 5.a. Nonnegativity and boundedness: \( C_{\tau_1/\tau_2} \in [0, 1] \).
Property 5.b. Monotonicity with respect to \( H \) and \( G \): If \( H \subseteq H' \) and \( G \subseteq G' \), then \( C_{\tau_1/\tau_2}(H', G') \leq C_{\tau_1/\tau_2}(H, G) \).
Property 5.c. Equivalence between reduction and 0 relative complexity: \( C_{\tau_1/\tau_2} = 0 \Leftrightarrow \tau_1 \not\equiv \tau_2 \).
Property 5.d. Equivalence between no reduction and positive relative complexity: 
\( C_{r_1/r_2} > 0 \iff \neg(\tau_1 \leq \tau_2) \).

Proof. Property 5.a: \( R_1(\pi \circ \pi_2 \circ \sigma) \in [0, R_1^*] \). Therefore, \( R_1(\pi \circ \pi_2 \circ \sigma) / R_1^* \in [0, 1] \implies C_{r_1/r_2} \in [0, 1] \) for any \( H, G \).

Property 5.b: Consider \( H, H' \) such that \( H \subseteq H' \) and \( G, G' \) such that \( G \subseteq G' \). For any function \( f \), the following is true:
\[
\inf_{h \in H^*, g \in G^*} f(h, g, \pi_2^*) \leq \inf_{h \in H^*, g \in G^*} f(h, g, \pi_2^*).
\]

This implies the following:
\[
\sup_{\pi_2 \in H^*} \inf_{h \in H^*, g \in G^*} \left[ 1 - \frac{R_1(\pi \circ \pi_2 \circ \sigma)}{R_1^*} \right] \leq \sup_{\pi_2 \in H^*} \inf_{h \in H^*, g \in G^*} \left[ 1 - \frac{R_1(\pi \circ \pi_2 \circ \sigma)}{R_1^*} \right].
\]

Property 5.c: Assume \( C_{r_1/r_2} = 0 \) for some \( H \) and \( G \) \( \iff \) for any \( \pi_2^* \in H^*_2 \exists g \in G \) and \( h \in H \) such that \( R_1(\pi \circ \pi_2 \circ \sigma) = R_1^* \). \( R_1(\pi_1) = R_1^* \iff \pi_1 \in H_1^* \). Thus, for all \( \pi_2^* \in H_2 \), \( g \in G \) and \( h \in H \) such that \( g \circ \pi_2^* \circ h \in H_1^* \), \( \tau_1 \leq \tau_2 \).

Property 5.d: The contrapositive of Property 5.c is \( \neg(\tau_1 \leq \tau_2) \iff C_{r_1/r_2} \neq 0 \).

By Property 5.a, the complexity measure is \( C_{r_1/r_2} \geq 0 \), thus \( C_{r_1/r_2} > 0 \iff \neg(\tau_1 \leq \tau_2) \).

B Additional Experimental Details

Approximating Relative Complexity using Q-learning. We apply Q-learning to Algorithm 1 by letting the loss functions \( L_1 \) and \( L_2 \) correspond to a Q-learning loss: \( L_4(\pi) = -\frac{1}{B} \sum_{b=1}^{B}[Q_2^*(s_b, a_b) \log p(a_b)] \), where \( p(a_b) \) corresponds to the probability of an action for policy \( \pi_2 \) (which may be a transformation of another policy such as \( \pi_2 = g \circ \pi \circ h \)), \( Q_2^*(s_b, a_b) \) are the Q-values, and \( B \) is the batch size. We run Algorithm 1 for 1000 iterations and use a batch size \( B \) of 1000 transitions.

Algorithm 2 Approximating Relative Complexity using SAC

1. **Input:** Threshold for success \( R_1^*, R_2^* \) on \( \tau_1, \tau_2 \) respectively
2. **Input:** Learning rates \( \lambda_1, \lambda_2 \), adversarial tuning parameter \( \alpha \)
3. **Input:** Function spaces \( H, G \)
4. **Input:** Q-functions \( Q, Q \) loss functions for \( \tau_1, \tau_2 \) respectively
5. **Output:** Approximate relative complexity \( C_{r_1/r_2} \approx C_{r_1/r_2} \)
6. **while** \( \neg(\text{converged} \land R_2(\pi_2) = R_2^*) \) **do**
7. **Step 0:** critic update
8. **Step 1:** \( \pi_2 \) update
9. **Step 2:** encoder/decoder update
10. **end while**
11. **end while**

12. **Algorithm 2 Approximating Relative Complexity using SAC**
Approximating Relative Complexity using SAC. We modify Algorithm 1 to use SAC for approximating the relative complexity. Let $Q^2_{\pi}$ be a critic of $\pi_2$ on task $\tau_2$ and $Q^1_{g \circ \pi_2 \circ h}$ be a critic of $g \circ \pi_2 \circ h$ on task $\tau_1$. We add an additional step to the algorithm for updating the critics on task $\tau_1$ and $\tau_2$. The critics are then used in the updates for the policy $\pi_2$ and the encoder/decoder. The resulting method is presented in Algorithm 2. We run Algorithm 2 for 50,000 iterations and use a batch size of 200 transitions.