CoboLT: An LfO Pipeline

Whitepaper
InversAI-2023-09-CLT-1

InversAI

Contents

1: Advancing Toward Automation

Automation has taken practically every industry by storm. It began with factories, where rote tasks are often performed in the same manner, but when artificial intelligence entered the picture, the global economy crossed an event horizon. AI has allowed automation to step into jobs previously assumed to require a human touch, such as software engineering. Now every business is asking the question of where this innovation can propel them forward. Yet, there are many obstacles that stand in the way of those looking to take the next step.

Cost: Many forms of automation do not come cheap. If a manager wants to install robots on their factory floor, the equipment alone can cost hundreds of thousands of dollars per unit. If they also want to add intelligence for tasks such as processing items on a conveyor belt, that costs extra—to say nothing of the expense involved in hiring an expert to program the robot. Many businesses operate on thin margins and cannot shoulder the short-term burden, even if it equates to more revenue down the road.

Time: To complicate matters further, for physical implementations of AI, installations can take months or even years. This includes systems integrators interviewing employees and disrupting productivity, adding to the financial sting of the process.

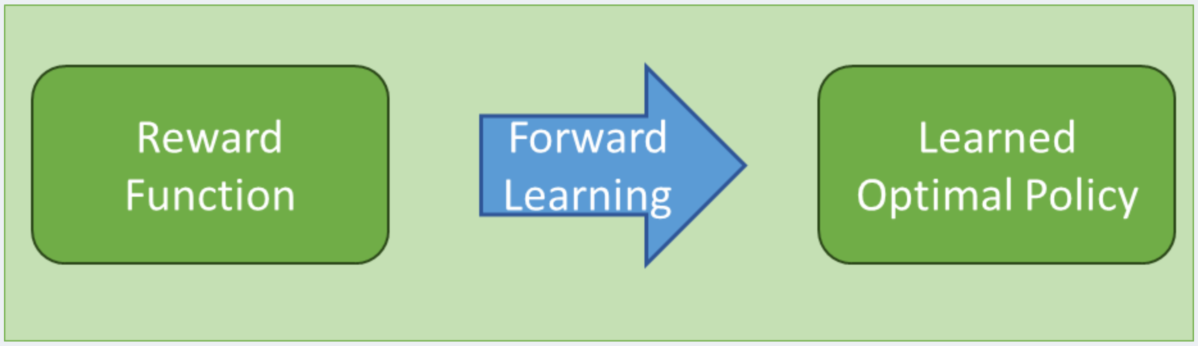
Opacity: Many business owners vaguely understand that automation can help boost productivity and focus their employees on more interesting tasks, but they often do not understand how it works. This is because robotics and AI are seen as “black boxes” by many outside those working directly with them, and the lack of knowledge can lead to frustration.

But these obstacles can be overcome, and it starts with being informed—both about some of the underlying theory, inverse reinforcement learning, and the technology that uses it.

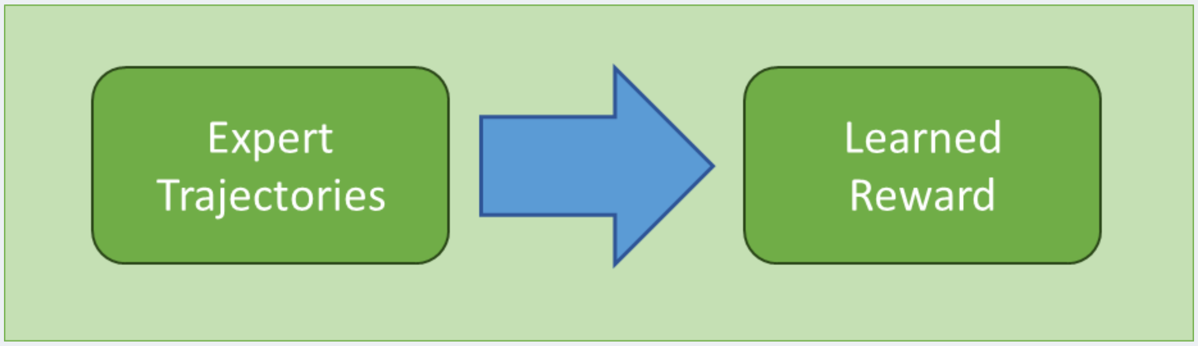
2: Understanding IRL

Before discussing inverse reinforcement learning, it can be useful to contextualize with reinforcement learning first, as many have heard of the latter. In both cases, the task, often formally known as a “domain,” must have some parameters defined. The first is *S*, the set of all states. A state is a configuration of the domain that comprises of the relevant features. The second is *A*, the set of all actions. The actions available to an agent can be used to potentially change the state of the domain.

The aim of reinforcement learning (RL) is to determine an optimal policy, π^*, which is a mapping of every state to a best action or distribution of actions. What marks a policy as “optimal” stems from the reward function. The reward function is typically expressed as a function of current state and action, $R(s, a)$, and is a matrix of numerical feedback; i.e., how “good” or “bad” it is to take action *a* in state *s*. The optimal policy is calculated such that the actions mapped to each state are expected to obtain the greatest current and future reward.



***Inverse* reinforcement learning** (IRL), as the name suggests, turns this idea on its head. Given a record of an agent’s behavior in a domain, referred to as the set of trajectories, χ, the aim is to determine that agent’s reward function. The assumption made is that the policy the agent is implicitly using is optimal, which makes IRL a problem of finding the reward function such that the trajectories given are optimal.



There are two main uses for performing IRL on a problem. First, trajectories typically only demonstrate how an expert performs on a subset of the domain, especially if the state space is particularly large. However, IRL calculates the reward function for the *entire* state and action space. By obtaining a reward function, the problem can potentially be reverted to a straightforward RL one instead. Determining a reward function manually for use by an AI can be difficult and produce unreliable results, but a human has an intuition for what is desirable and what is not, meaning the reward function derived from their behavior will likewise be intuitive. Deriving a policy from a human's reward function is especially useful in the field of robotics: by watching a person perform a task, the robot can then replicate that task using IRL to drive its understanding. With proper application of IRL, the robot will perform the task the exact same way.

A robot could be taught how to sort onions at a conveyor belt by watching a human worker do it and learning that they keep good onions and throw bad ones away.

Instead of a robot sorting the onions, an AI could watch human workers perform the task it learned and monitor them to ensure proper and quick execution.

The same learned policy does not necessarily have to be *executed* by the AI; sometimes it is useful simply to *understand* what that policy is. Some tasks require a combination of decision-making and precise movement, and with current technology, humans still tend to outperform robots in this area. But an AI in a supervisory role that understands the task can watch human workers to ensure that they are completing their tasks in a correct and quick manner.

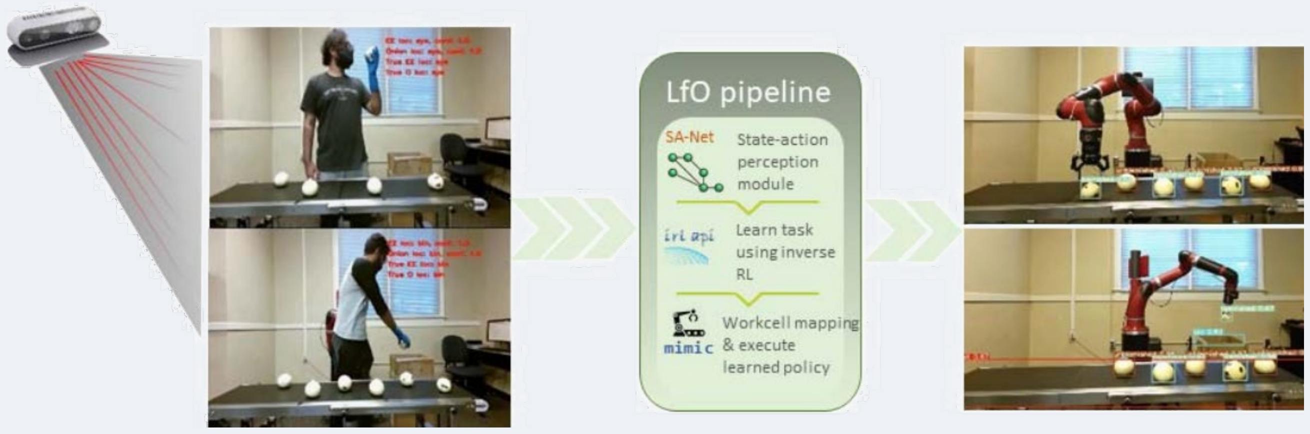
The second use relates to using the reward function itself to learn more about the agent observed. The reward function is a numerical representation of the preferences in the domain. In the context of IRL, it denotes what the agent prioritized and what they would rather avoid. By learning more about how that agent thinks, the observer can react to that agent accordingly, whether the environment in question be cooperative or competitive.

Having learned a hacker's preferences through IRL, a cybersecurity firm can anticipate future attacks and take appropriate steps to combat them.

IRL is a versatile tool that can be applied to nearly any problem that can be observed and becomes more powerful with the proper technology to utilize it.

3: Leveraging IRL With CoboLT

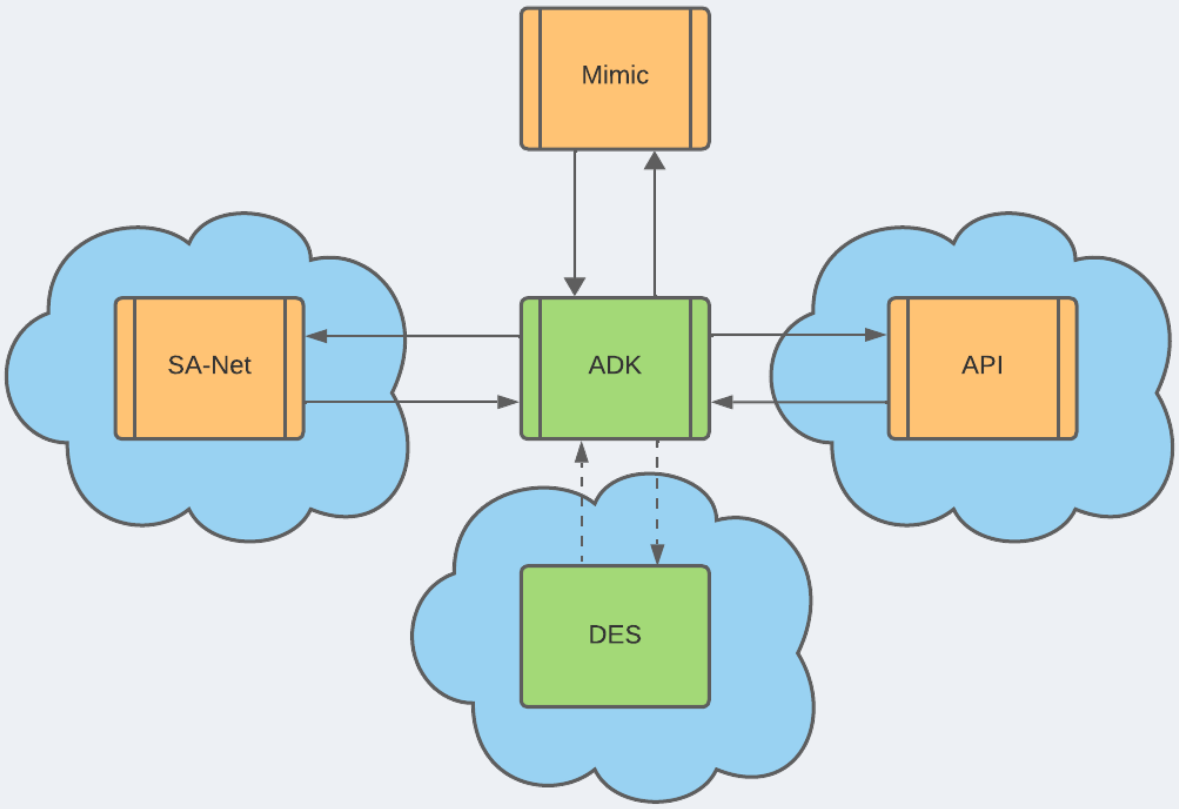
CoboLT is a software pipeline with IRL as its linchpin. It is designed with the tenets of Industry 5.0 in mind, with the belief that the next step forward is having humans and AI collaborate to utilize their respective strengths. It is composed of three parts, and these parts are controlled with a central component.


LfO pipeline
SA-Net
State-action perception module
irl api
Learn task using inverse RL
mimic
Workcell mapping & execute learned policy

That central component is the CoboLT ADK. Each element of the pipeline requires an intimate understanding of IRL to work with, but the ADK allows anyone with programming abilities and general AI and ML knowledge to interact with CoboLT through abstract calls that invoke the desired behavior.

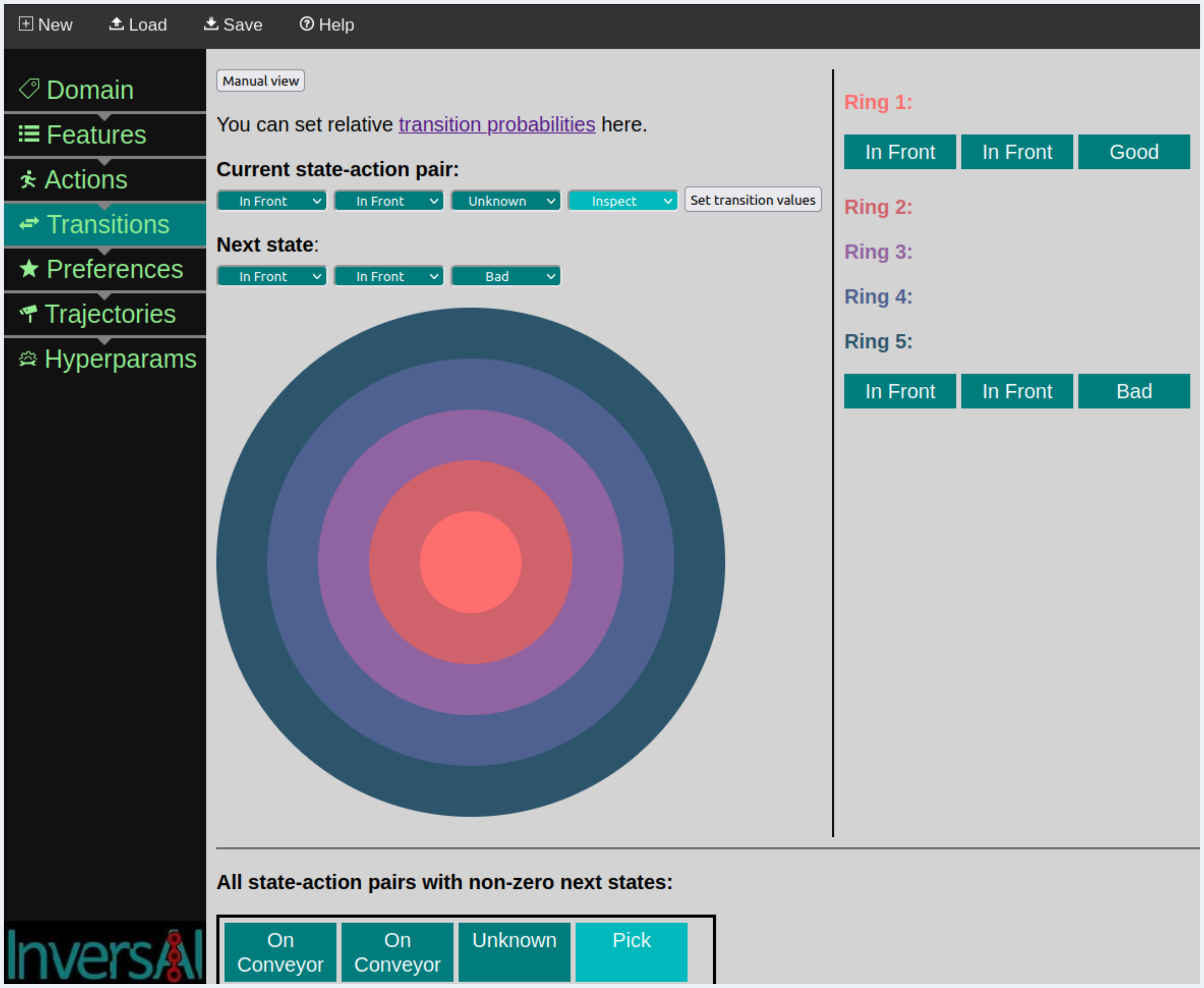
The first component of the pipeline is **SA-Net**. SA-Net is a deep-learning architecture that takes data streamed to cameras and classifies what is shown into states and actions. In many physical tasks, SA-Net can be used to gather the trajectories of a human agent performing in the domain, which is passed to the next part of the pipeline.

The second component is the **irl-api**. Hosted on the Cloud via AWS's EC2 service is the API. The API is a group of libraries that contain the vital functionality for executing IRL, from tensor logic to the IRL algorithms themselves. Given proper data from the outside, it can apply the desired IRL algorithm to it and send the resultant reward function and policy back. The aforementioned "outside" is the ADK. It is responsible for passing the data to the API, telling it what algorithm to use, and retrieving that data.



Additionally, the domain needs to be formally defined and formatted in a specific manner so that the ADK and API are able to interpret its content. This is the role of our Domain Elicitation Software (DES). DES is hosted on AWS's Amplify app and guides the user through an intuitive process of describing the domain, from its states (shown here as features, a combination of notable aspects that comprise a state together) to its transition function (a model of the uncertainty of the domain).

DES is also responsible for ensuring that the trajectories gathered from SA-Net or other sources are also in a readable format for the ADK and API. By stepping through this process in a GUI, the user is spared the frustration of the error-prone procedure of specifying the inputs entirely by hand.



You can set relative transition probabilities here.

Current state-action pair: In Front | In Front | Unknown | Inspect

Next state: In Front | In Front | Bad

Ring 1: In Front | In Front | Good

Ring 2:

Ring 3:

Ring 4:

Ring 5: In Front | In Front | Bad

All state-action pairs with non-zero next states:

On Conveyor | On Conveyor | Unknown | Pick

The final part of the CoboLT pipeline is the mimic module. This module takes the policy that is learned from the irl-api and plugs it into a robot so that it may reference that policy and execute on it.

CoboLT is modular; not every problem will require all parts of its architecture, but they are not required to work in tandem. This versatility enables CoboLT to function in essentially any domain that also works with IRL and assist all sorts of businesses in flourishing through automation.

4: Benefitting Businesses

In the examples given for IRL before, they can use some or all of CoboLT's pipeline. For onion sorting, SA-net can watch a human worker perform the task, and these recordings can be passed along to the *irl-api* to learn that worker's preferences. This can either be plugged into the *mimic* module so that a robot can do the same or programmed into an AI "supervisor" to ensure that employees are doing their jobs correctly.

As for cybersecurity, system logs would likely take the place of SA-net, and one would have little interest in replicating a hacker through *mimic*, but obtaining their preferences through the *irl-api* would be of great interest to ensuring protection against future attacks. Generally, CoboLT works to solve every problem mentioned at the beginning. It is:

Inexpensive: CoboLT cuts out the middleman integrator because the teaching is done by the expert, and the learning is done by CoboLT. For tasks being replicated, the programming of robots is handled by *mimic*, leaving the only expenses being CoboLT itself and whatever robotics equipment is used.

Fast: Using IRL reduces the process of learning to a matter of weeks. The task is taught by intuition through the expert that is being observed, meaning that there is no need for on-site visits and interviews. If the task in question uses SA-net, it is also minimally disruptive, since only cameras need to be installed to watch the task being performed.

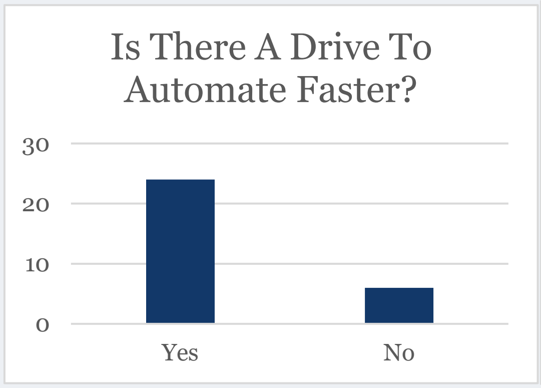
Transparent: CoboLT is designed with simplicity in mind. Anyone possessing a general familiarity of AI and machine learning can use the components of CoboLT to suit their needs. If a business does not have such an employ, the developers of CoboLT are also able to assist in any capacity and are receptive to feedback for making the technology even more intuitive.

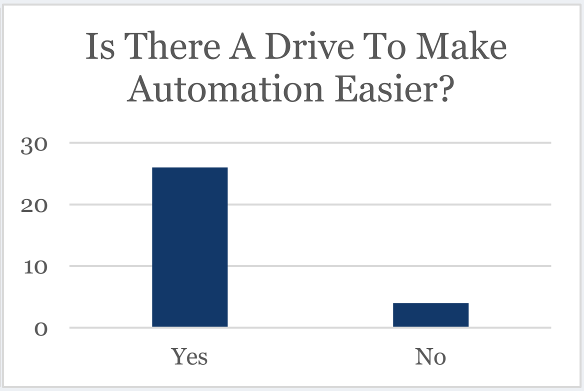
5: Staying Informed

There is a surfeit of options when it comes to how one chooses to add automation or AI to their business. It is to the benefit of anyone considering their choices to imagine not just how the final automation system will look, but what it will take to achieve their goals.

CoboLT presents an option that is intuitive. It mirrors the process of learning from a teacher, which makes the idea comprehensible to anyone. Yet it is even less disruptive in the learning process, as that behavior is already recorded by pre-existing systems, or can be recorded with small cameras via SA-net.

Our team interviewed thirty different companies in the field of or interested in automation, and most of them expressed a need to implement their solutions more quickly, and many also agreed that there was a need for their solutions to be more easily implemented. CoboLT was designed with both benefits in mind.





The pipeline is adaptable to suit the needs of many. Whether one wishes to replicate the way a person performs a task or to simply learn how that person thinks, CoboLT has the means for both. Any task that can be formally described and observed can be learned. Whether it be medicine, agriculture, finance, or any other industry that comes to mind, CoboLT has a use case in it.