Act-Learn: Automation That Builds Its Own World Models

The System

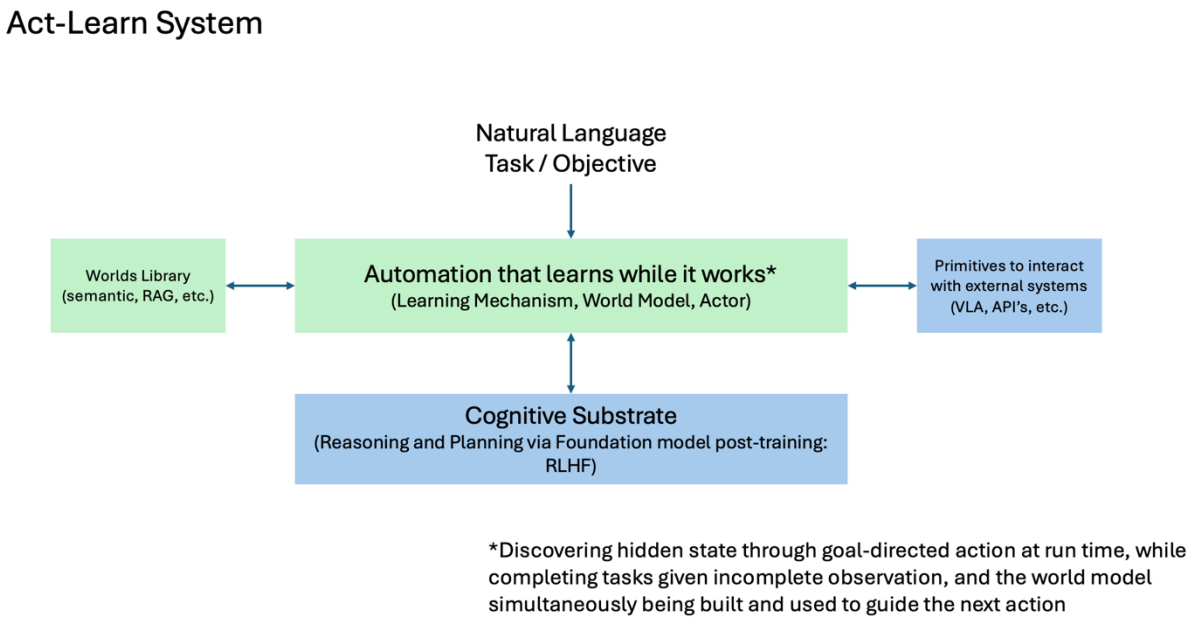
This paper describes Act-Learn, an automation system that can be tasked in natural language, build causal world models through operation without gradient training, act on those models to accomplish goals, and refine them continuously as it encounters new experiences.

The Act-Learn system can be dropped into an environment it has never seen, given a task in natural language, and begin productive work immediately – learning the environment's rules while it operates, with human guidance when needed.

Starting from the Architecture

The Act-Learn system has five components. A natural language interface for tasking. A central automation loop that learns while it acts and stores that learning into a world model. A worlds library that stores and retrieves learned world models. A set of primitives for interacting with external systems. And a cognitive substrate – a foundation model that provides reasoning and planning infrastructure.

These components interact continuously. Understanding the architecture means understanding what each component does and why it exists as a separate concern.



Natural Language Tasking

The system accepts tasks and objectives in natural language. "Sort these parts by size." "Explore this environment and report what you find." "Resume the previous task but skip positions that failed."

This is not a convenience feature, rather it is a design requirement. Natural language tasking means the system can be re-tasked in minutes by people who understand the work, without engineering intervention or gradient retraining. It also means task specifications compose naturally with accumulated world knowledge – the system interprets "sort these parts" considering everything it has learned about the current environment, including mechanisms and constraints it discovered on its own.

The planning and reasoning capabilities of the cognitive substrate makes this possible. Foundation models understand natural language natively, which means the interface between human intent and system behavior requires no translation layer, no formal specification language, and no recompilation.

The Core Loop: Learning While Acting

The center of the architecture is a system that simultaneously acts on its current understanding and updates that understanding based on what happens. This is the operational heart of Act-Learn, and it is the piece most often misunderstood.

The dominant paradigm in machine learning separates learning from operation. You collect data. You train a model. You deploy it. If performance degrades or the task changes, you collect more data and retrain. The system is inert between training cycles.

Act-Learn is always doing both: executing toward an objective and building its model of the world. Every action is both productive work and a probe of the environment. Every outcome is both a task result and evidence about how the world behaves.

This matters because most operational environments contain hidden state – facts about the world that are not given in advance, not documented, and not discoverable except through interaction. A position on a workbench is locked. A sensor drifts under certain conditions. An object can be stacked on another object, but that was not mentioned because it seemed obvious. A process behaves differently on Mondays because of a thermal cycle from the weekend shutdown.

The system discovers hidden state through goal-directed action at run time. It completes tasks given incomplete observation, and its world model is simultaneously being built and used to guide the next action. This is not learning-then-acting or acting-then-learning. It is a single integrated process.

How Discovery Works

When the system acts and the outcome matches expectations, the action succeeds and the world model is confirmed. When the outcome surprises – when what happens does not match what the world model predicted – the system has discovered something.

Surprise is the primary learning signal. The system notes what it expected, what actually happened, and the context in which the discrepancy occurred. It forms hypotheses. When it can, it tests them – not through idle exploration, but through subsequent goal-directed actions that also happen to disambiguate competing explanations.

Over time, confirmed hypotheses become generalizations: stable, human-readable statements about how the world works. "Objects can be stacked at unoccupied positions." "Position (4,4) does not accept placements." "Cylindrical parts require higher grip force than flat parts." These generalizations constitute the world model.

The process is salience-driven. Not every observation becomes a learning event. The system attends to novelty – prediction failures, unexpected patterns, surprising combinations. Routine confirmation of known facts is noted but not elevated. This selectivity is what makes the system efficient: it focuses its learning capacity on what it does not yet understand.

The World Model and Worlds Library

The world model is not a neural network. It is a structured collection of durable generalizations stored in natural language, readable and editable by human operators. This is a deliberate design choice with significant implications.

Interpretability. When the system makes a decision, you can see why. The reasoning trace references specific generalizations. "I avoided position (4,4) because of generalization G7: some positions are locked and will not accept placements." When the system makes an error, you can identify which generalization was wrong or missing.

Editability. Operators can inspect the world model at any time. They can correct a wrong generalization, add one the system hasn't discovered yet, or delete one that was true last week but no longer applies. This is direct human oversight of the system's knowledge, not a request to "provide more training data."

Portability. The world model is not embedded in the weights of a neural network. It is a portable document. It can be exported, versioned, transferred between systems, or migrated between cognitive substrate providers. Switch from one foundation model to another and the accumulated knowledge comes with you. The knowledge belongs to the operator and the business, not the platform.

Composability. World models can be combined. A base model of general physics can be composed with a site-specific model of a particular production line. Lessons learned on one shift can be shared with the next. A model developed during commissioning can be refined during production.

The Worlds Library

The worlds library stores and manages world models across environments and over time. When the system enters a new environment, it can retrieve relevant prior world models – not to apply them blindly, but to establish priors that accelerate learning. A world model from a similar production line provides starting hypotheses. The system then confirms, refines, or overrides these hypotheses through operation in the new environment.

This is analogous to how an experienced technician approaches a new site. They don't start from zero. They bring knowledge from similar environments, apply it tentatively, and update it as they learn the specifics. The worlds library makes this institutional rather than individual – knowledge accumulates across deployments, not just within one.

The library also enables retrieval by structure and context: semantic similarity, causal pattern matching, and operational relevance. This goes well beyond simple document retrieval. The system is searching for world models whose causal structure matches the current situation, not just world models that use similar words.

Primitives: The Interface to External Systems

The system interacts with the physical or digital world through primitives – discrete, well-defined actions exposed by external systems. In a robotics context, these might be motor primitives provided by a vision-language-action model: grasp, place, rotate, pour. In a software context, they might be API calls, database operations, or sensor queries.

Primitives are the system's hands. Act-Learn is the system's mind. This separation is critical.

The system does not learn how to grasp. It learns *when* to grasp, *what* to grasp, and *why* a grasp might fail in this particular environment. The kinematics of grasping are someone else's contribution – solved once by specialized methods (such as VLA) and exposed as a stable interface.

This separation means Act-Learn is substrate-independent on both sides. It works with whatever cognitive substrate provides the best reasoning (today's foundation models, tomorrow's better ones). And it works with whatever physical or digital primitives are available – different robots, different actuators, different software platforms. The world knowledge transfers because it is expressed in terms of what things do, not how the underlying system accomplishes them.

The Feedback Loop

The separation is not a wall. When primitives fail, the world knowledge layer captures the failure context in detail. As similar failures accumulate, patterns emerge: this primitive struggles with cylindrical objects over a certain diameter, or this API call times out under specific load conditions. These patterns become capability requirements – structured descriptions of what the primitive layer needs to handle but currently is unable to.

The world knowledge layer becomes, in effect, a curriculum for primitive development. It identifies gaps not through abstract analysis but through operational evidence.

The Cognitive Substrate

The bottom layer of the architecture is a foundation model – a large language model, post-trained with RLHF or similar methods. This is the reasoning and planning infrastructure that makes everything else work.

It is critical to understand what the cognitive substrate is and is not.

It is infrastructure. The foundation model provides attention mechanisms for pattern matching, reasoning capability for hypothesis formation and testing, natural language understanding for tasking and knowledge representation, and planning capability for action sequencing. These are computational services, analogous to an operating system providing memory management and process scheduling.

It is not the intelligence. The system's knowledge about a specific environment – its world model – is not stored in the foundation model's weights. The model did not learn during pre-training that position (4,4) is locked. The episodic compression mechanism, the salience detection, the generalization formation, the world model management – these are architectural contributions that use the foundation model as a tool.

This distinction matters for several reasons. The cognitive substrate can be upgraded or swapped without losing accumulated knowledge. The system's learning is not constrained by the foundation model's training data. And the intelligence of the system as a whole exceeds what the foundation model provides alone, because the architecture adds learning mechanisms the model does not possess natively.

What This Is Not

Act-Learn is not a memory system. Memory is one component. The system builds, maintains, tests, and revises causal models of environments. Memory serves this process but does not define it.

It is not RAG. Retrieval-augmented generation retrieves pre-existing documents. Act-Learn creates new knowledge structures through operational experience. The world model emerges from the interaction between the system and its environment.

It is not a chatbot with tools. The system maintains persistent, evolving state – a world model that changes with every deployment and every shift. It has continuity of knowledge across sessions and tasks. It learns.

It is not an agent framework. Agent frameworks orchestrate tool calls in response to prompts. Act-Learn constructs world models, reasons over them, discovers hidden state, and refines its understanding over time. The architecture adds epistemological and praxis machinery – a theory of how to build knowledge from experience and how to take action accordingly – that agent frameworks do not possess.

The Practical Consequence

The practical consequence of this architecture is a system that can begin useful work before it fully understands its environment, and whose understanding improves through the work itself. There is no training phase. No data collection campaign. No retraining when conditions change.

The system arrives, receives a task in natural language, begins working with whatever world knowledge it has (possibly none, possibly a relevant model from the worlds library), discovers what it doesn't know through action, builds its world model in real time, and gets better at the task as it performs – with human operators able to inspect, correct, and guide the process at every point. The longer it operates, the more valuable the world models become – and because they are human-readable and human-editable, that growing knowledge base is an institutional asset that people control, curate, and build on.

Domain-specific applications of this architecture, including industrial automation and layered robotic learning, are described in companion publications available at https://mossrake.ai.

Version 1.1