Layered Learning in Robotics

Motor Primitives and World Knowledge as Distinct Domains

Abstract

Vision-language-action models and episodic compression may be framed as competing approaches to robot learning. This framing obscures a more useful insight: they address different problems at different layers of the system. In our proposed approach, VLA learns the mapping from commands to fine kinematics, producing robust motor primitives. Episodic compression learns world knowledge while relying on those primitives as a stable interface. A layered architecture that uses each where appropriate may prove more practical than either approach alone. Further, the layers can work symbiotically: the world knowledge layer identifies gaps in primitive capability, generating targeted requirements for primitive development.

Introduction

The robotics research community has converged on end-to-end learning as the dominant paradigm: Train a model on demonstrations; Map perception to motor commands; Let task knowledge emerge from sufficient data.

This approach has produced impressive results in controlled settings. It has also proven difficult to deploy in production environments where tasks change frequently, failures must be explained, and retraining cycles cannot keep pace with operational demands.

We propose an alternative framing: the end-to-end pipeline conflates two distinct learning problems that have different characteristics and different optimal solutions. Separating these problems clarifies where each learning approach belongs and suggests a layered architecture that uses both.

Two Learning Problems

Consider what a robot must know to perform useful work. There are at least two distinct categories of knowledge, and they have different properties.

Motor Primitives are skills that translate commands into physical action: grasping, placing, rotating, inserting, pouring. These skills share several characteristics. They require mapping from commands to fine kinematic sequences that accomplish the commanded action. They benefit from large datasets and gradient-based learning. Once learned, they generalize across many tasks. They change slowly over time. And they are shared across deployments: a grasping primitive trained for one factory works in another.

World Knowledge is different. Positional constraints, environmental quirks, operational patterns, equipment-specific behaviors. These facts are specific to each deployment environment, they change as the environment changes, they can often be learned from single observations, they must be inspectable and editable by operators, and critically, learning them does not require updating model weights.

VLA as typically applied treats both problems as one. It learns the complete chain from perception to motor commands for each task, which means every task implicitly re-learns the command-to-kinematics mapping inside the task-specific model. This explains two persistent difficulties with VLA deployment. First, the approach requires many demonstrations per task because motor skills are being learned anew each time rather than reused. Second, knowledge transfer between tasks is implicit and difficult to control. Fine-tuning can carry capability forward, but learned motor skills remain entangled with learned world knowledge in the model weights. You cannot extract a grasping primitive from one model and compose it with rotation from another.

The alternative: factor the problems apart. Learn motor primitives once, amortized across tasks, and expose them as an interface. Learn world knowledge continuously during operation, building on that interface.

Biological Precedent

Human motor and cognitive development does not seem to proceed end-to-end.

Infants arrive with reflexes: grasping, rooting, stepping. These are not learned. They are innate motor programs that provide a foundation for later development.

Toddlers develop motor primitives through months of undirected play. Reaching for objects. Grasping and releasing. Modulating grip force (learning that squeezing too hard crushes soft objects). Rotating the wrist. These primitives develop through repetition until they become automatic. A three-year-old does not consciously plan how to grasp. Grasping is a solved problem, available as a building block for higher-level activity.

By the time a child attempts a shape-sorting puzzle, grasping is not the cognitive challenge. The work is understanding which shape fits which hole. That is world knowledge: learning the properties of this particular toy, in this particular situation. The motor primitives are already in place, operating as an interface that cognition calls upon.

The developmental sequence can be summarized as follows. First, reflexes and early motor control, which are innate plus early neurological development. Second, motor primitives, learned through play and eventually automatic. Third, world physics, learned through experience with the environment. Fourth, task reasoning, which is cognitive composition over primitives and accumulated world knowledge.

VLA attempts to collapse this sequence into a single learning phase for each new task. A robot learning to stack blocks must implicitly learn grasping, placing, and the physics of stacking all at

once, from demonstrations of the complete behavior. This contradicts what we observe in biological systems, where motor competence develops separately from and prior to task-specific world knowledge.

A Layered Architecture

We propose separating robot learning into distinct layers, each addressed by appropriate mechanisms.

Layer 1: Kinematics and Control. This layer handles the physics of the robot body: joint control, trajectory planning, force limits, safety constraints. It is engineering, not machine learning in the modern sense. Classical control theory, carefully tuned parameters, and rigorous testing. This layer changes only when the hardware changes.

Layer 2: Motor Primitives. This layer provides reusable skills that translate commands into action: grasp, place, rotate, insert, pour. These primitives are exposed as an interface to higher layers. The primitive accepts a command ("grasp object at position X") and produces the fine kinematics required to execute it.

VLA is well-suited to building this layer. Programming fine kinematics by hand is genuinely difficult. One must account for variation in object shape, size, weight, and surface properties. Positioning is uncertain. Gripper dynamics and force control interact in ways that are hard to model. Classical approaches require extensive tuning and still fail on edge cases you did not anticipate. The resulting code is brittle and full of special cases.

VLA sidesteps this: Show it enough demonstrations and gradient descent discovers kinematic solutions that handle variation one would never enumerate by hand. Robustness emerges from data rather than engineering effort. The investment amortizes: a grasping primitive trained once serves every task that requires grasping.

The key insight is that primitives should be task-general, not task-specific. Training a grasping primitive is valuable. Training a "grasp-this-particular-bracket-for-this-particular-assembly" model is a inefficient use of the technology. The primitive should work across brackets, bins, tools, and parts. Task-specific knowledge belongs in a different layer.

Layer 3: World Knowledge. This is where episodic compression belongs. Learn environmental constraints, operational patterns, and context-specific physics at runtime, while treating the primitives from Layer 2 as a stable interface.

Salience-driven consolidation is appropriate here because observations are sparse and environment-specific. A single failed placement at position (4,4) is enough to learn that position (4,4) is locked. You do not need thousands of examples.

This layer must support inspection and editing. When the system learns something incorrect, operators need to fix it directly rather than wait for a retraining cycle. Human-readable

generalizations stored in an editable knowledge base meet this requirement. Neural network weights do not.

Layer 4: Task Reasoning. Given primitives and world knowledge, compose action sequences to achieve goals. This is inference, not learning. A language model or similar reasoning engine operates over the primitive interface and accumulated world model. The task "stack parts into bins by size" becomes a planning problem: which primitives to invoke, in what sequence, given what is known about the environment.

The Feedback Loop

The layers need not be independent. The world knowledge layer can identify gaps in primitive capability, creating a feedback loop that guides primitive development.

Consider an example sequence. A system operates using available primitives. During a task, a grasp-and-rotate operation fails repeatedly on cylindrical objects over 5cm in diameter. This is a salient event: the system expected success and observed failure. The failure is recorded as an episode with full context (object properties, gripper configuration, failure mode).

As similar episodes accumulate, a pattern emerges. The system does not merely learn "large cylinders are hard." It accumulates structured information about exactly when and how the primitive fails. This information constitutes a capability requirement: grasp-rotate needs to handle cylinders in the 5-10cm range, smooth surfaces, 200-500g weight.

This requirement can drive targeted VLA training. Rather than collecting millions of demonstrations hoping to cover the space of possible objects, primitive development responds to identified operational needs. The new or improved primitive becomes available. The system continues with expanded capability.

This mirrors biological development more closely than either approach alone. Humans do not randomly develop motor skills. Motor learning responds to task demands. A child learning to use scissors develops a specific grip pattern because the task requires it. The cognitive layer (wanting to cut paper) identifies what is needed. Motor learning responds.

The episodic compression layer becomes, in effect, a curriculum designer for primitive development.

Implications

For VLA research. Our view is not that VLA is the wrong technology – we believe current practice asks it to solve problems that are best served by separate treatment. Bundling primitive learning with task planning in end-to-end models conflates what should be distinct. Learning the mapping from action commands to fine kinematics is exactly what is needed for building robust, general-purpose primitives. It solves a problem that is genuinely hard to solve by other means. Hand-coded kinematics are brittle. VLA-trained primitives could handle variation that would be impractical to enumerate manually.

The research investment in VLA pays off when primitives amortize across the industry. A robust grasping primitive, trained once and deployed everywhere, justifies the many demonstrations and weeks of training time. A task-specific model that must be retrained for each new assembly does not.

This suggests a shift in research focus: from task-specific performance benchmarks to primitive robustness and generalization. The question is not "can this model fold laundry" but "can this grasping primitive handle the range of objects and configurations it will encounter across thousands of deployments."

For deployment. The layered architecture changes deployment economics significantly.

Primitives become platform capabilities, developed once by robotics vendors or research institutions and shared across the industry. This is similar to how operating systems provide standard interfaces that application developers build on.

World knowledge accumulates at each deployment site during normal operation. The system learns that bin B sticks, that position (4,4) is locked, that parts from supplier X have tighter tolerances. This knowledge is specific to the site and cannot be trained in advance.

Task changes require natural language specification, not retraining. An operator says "now sort by color instead of size." The task reasoning layer plans a new sequence using existing primitives and existing world knowledge.

When primitive gaps are identified, they are addressed systematically rather than through ad-hoc retraining. The gap is documented, requirements are specified, and primitive development proceeds with clear targets.

For the research agenda. The question shifts from "which learning approach is better" to "which layer does each approach address." This suggests different research priorities for each community.

For VLA research: robustness and generalization of primitives across objects, environments, and edge cases. Transfer learning between primitive types. Efficient fine-tuning when gaps are identified.

For episodic compression research: scaling world knowledge to hundreds or thousands of generalizations. Compositionality and consistency as knowledge accumulates. Integration of human curation without creating bottlenecks.

For integration research: interface design between layers. How primitives should expose failure modes with enough detail for the world knowledge layer to learn from them. Feedback mechanisms that translate operational experience into primitive development requirements.

Open Questions

Many questions remain unresolved, including:

Primitive interface design. Where exactly should the primitive interface live? Options include firmware on the robot, middleware that abstracts across robot platforms, or learned adapter layers that translate high-level commands to robot-specific control. Each has different implications for portability and capability.

Failure mode exposure. How do primitives communicate failure to the world knowledge layer? A binary success/failure signal may lose important information. "Grasp failed" is less useful than "grasp failed due to slip at 3N force on smooth surface." But richer failure information requires more sophisticated primitive design.

Scope boundaries. Are there task classes where the layer separation does not hold? Dexterous manipulation, contact-rich assembly, or tasks requiring continuous force feedback might require tighter coupling between sensing and action than discrete primitives allow. The architecture may be domain-specific rather than universal.

Primitive granularity. What is the right level of abstraction for primitives? Too coarse (e.g., "assemble part") and you lose the benefit of composition. Too fine (e.g., "move joint 3 by 0.1 radians") and you lose the benefit of factoring out motor skills. The optimal granularity likely varies by domain.

Primitive discovery. Can the system identify when it needs a primitive that does not exist, as opposed to when an existing primitive is failing? "I need grasp-rotate to work better on cylinders" is different from "I need a primitive for insertion with compliance that I do not currently have."

Conclusion

The debate between end-to-end learning and alternative architectures has been framed as a competition. We suggest a more productive framing: these are solutions to different problems at different layers.

VLA learns motor primitives by mapping commands to the fine kinematics required to execute them. Episodic compression learns world knowledge while treating those primitives as an interface. A layered architecture that uses each where appropriate offers a practical path to deployable robot learning that neither approach achieves alone.

The approaches are not merely compatible. They are symbiotic, and the symbiosis is genuine: each solves a problem that is hard to solve otherwise. VLA learns robust kinematics from data, avoiding the brittleness of hand-coded solutions. Episodic compression learns world knowledge at runtime, avoiding the retraining cycles that make VLA impractical for deployment-specific facts. The world knowledge layer identifies primitive gaps through operational experience.

Primitive development responds to identified needs. The system expands its capabilities over time through the interaction of both learning mechanisms.

Robotics requires learning at multiple layers, with different mechanisms appropriate to each. Recognizing this may help the research community move past methodological debates toward architectures that actually deploy.

This article describes research from Mossrake Group LLC on layered learning architectures. Related work on episodic compression is available at https://github.com/mossrake/learning-system and at our website.