

Generalization by Construction

What learning should leave open for inference

Yilun Du

Kempner Institute and School of Engineering and Applied Sciences
Harvard University
ydu@seas.harvard.edu

Abstract

Robots and other AI systems must operate reliably in situations their training data never covered. This essay argues that such situations are often unfamiliar only as wholes: they can be decomposed into parts that are individually familiar. Instead of training a direct mapping from observations to actions, a system can learn reusable models, such as evaluators, constraint factors, dynamics, and skills, and leave some decisions open for search at inference time. I call the division between what learning fixes and what inference resolves the *amortization boundary*, and I organize inference into five scopes: choosing a candidate solution, composing a task model, planning actions, controlling the reasoning procedure, and organizing multiple reasoners. For each scope, I discuss representative methods, from energy-based models and diffusion planners to multi-agent verification, and argue that the boundary can shift as experience accumulates.

1 Introduction

The ultimate goal of artificial intelligence is to build systems that can act effectively in situations they have never seen before. This is especially important in robotics, where a robot will face new environments, objects, constraints, and goals. The dominant approach is to collect ever-larger datasets and train neural policies end to end to map observations directly to behavior. Yet these policies often struggle in unfamiliar settings, and it is practically impossible to gather examples covering the open-ended range of situations a robot may encounter.

Symbolic AI pursued a different approach to building generalizable systems. It separated explicit knowledge from the procedures used to reason with it. Facts, relations, rules, and actions were represented as explicit objects that could be reused and recombined during inference. A planner could use this knowledge to search for a sequence of actions that achieved a new goal. McCarthy’s *Programs with Common Sense* [1] and the STRIPS [2] planning system are early examples of this approach.

A major difficulty, however, was deciding what knowledge to represent and how to search over it. For images, video, language, and physical interaction, even the right vocabulary was difficult to specify in advance. People also had to define the symbols, predicates, and operators by hand and connect them to continuous perceptual signals. Finally, search was often intractable in large and partially observed spaces.

Deep learning has made it possible to learn powerful representations directly from high-dimensional perceptual inputs. Foundation models provide a broad source of reusable knowledge that can support search and planning. However, directly training a policy model to predict behavior entangles the learned representation with the computation used to produce that behavior. The resulting mapping works well in situations that follow the regularities it has learned, but often fails when a new problem falls outside them.

Research perspective, August 29, 2026.

Web version: https://yilundu.github.io/perspective/generalization_by_construction.html

In this perspective, I argue that many globally unfamiliar problems can be solved by decomposing them into locally familiar parts. By learning models of reusable structure, we can give a robot the pieces needed to construct solutions to unfamiliar tasks. Inference then determines how to combine these pieces and searches for a solution that satisfies the current goals and constraints. The central design question is what structure learning should capture in advance and what it should leave open for inference to compose and construct.

Core claim. *Learning captures locally reusable structure. Inference uses it to construct solutions to globally unfamiliar problems.*

2 Blending amortization and inference

I call this division of work between learning and inference the **amortization boundary**. Learning can amortize recurring decisions [3] by building reusable rules that respond to new inputs without searching over alternatives each time. Inference can use learned models to compare and revise possible solutions for the current problem. For example, a robot can use a learned policy to choose a grasp directly, or use a learned model to evaluate several possible grasps before choosing one. The boundary marks which choices are resolved by a learned rule and which remain open to problem-specific search, selection, or refinement. Here, **inference** refers to this latter process.

The boundary trades computation for flexibility. Repeated search wastes computation when a learned rule can already make the decision reliably. If a new goal or constraint falls outside what a learned rule can handle, the system needs a way to reconsider its decisions. The aim is therefore not to maximize inference, but to leave open the choices whose values cannot safely be determined in advance.

The amortization boundary also changes the generalization problem itself. A direct predictor must learn how a problem maps to a complete solution. Solution inference instead uses a learned criterion for judging candidates, while search constructs a solution for the current problem. Once a candidate is proposed, judging it can depend on fewer aspects of the problem and require a simpler computation than producing the solution directly. When this judgment depends on simpler, reusable structure, learning the criterion can support better generalization than learning the full problem-to-solution mapping.

Model composition moves another choice to the inference side of the boundary. Rather than learning the criterion for each task as a whole, the system learns reusable factors for individual goals, constraints, or relationships. Because each factor depends on a narrower part of the problem, inference can recombine them to construct criteria for tasks not seen during training. Such a separation enables effective generalization to novel combinations of local relationships, dynamics, and goals.

Program specialization [4] offers a useful analogy for the amortization boundary. A programmer can compile known assumptions into an efficient executable, while retaining a higher-level program in which those assumptions can be revised. A direct neural policy resembles an aggressively specialized executable. It is fast to run, but hard to adapt when a new problem violates its compiled assumptions. An energy function or world model can preserve a higher-level description of possible solutions or futures that sampling, optimization, or planning specializes for the current problem.

The representation of a task also determines how much of the problem can be resolved through inference. A planner does not need to search over every motor command. Instead, it can search over task-relevant states, subgoals, or trajectories while learned skills handle familiar local transitions. A well-engineered representation enables effective inference and generalization.

This suggests four practical steps.

1. Identify a decision whose answer changes across tasks while the structure needed to evaluate alternatives remains reusable.

2. Choose a level of representation at which that choice can be inspected and revised.
3. Learn reusable structure that helps resolve the choice without fixing it in advance.
4. Use inference to resolve it for the current problem.

My PhD thesis [5] developed this approach for the composition of learned energy landscapes. Here I use the amortization boundary to examine how the same general principle can shape both the solutions a system constructs and the computation it uses to construct them.

3 Hierarchical inference

Consider a household robot asked to bring a mug of coffee from a crowded kitchen counter to someone at the table. It must lift the mug without disturbing the surrounding dishes, carry it without spilling, and set it within easy reach. I distinguish five scopes according to what inference can reconfigure as the robot solves the task. The first three construct the problem and its solution, while the final two construct the computation used to solve it.

In this task, solution inference searches for a stable placement of the mug. Model composition combines support, uprightness, clearance, and reachability factors into a model that judges the placement. Action planning searches for actions expected to produce that outcome. Procedure control decides when to verify a plan, repair it, or replan, while reasoner organization determines which specialists propose and evaluate plans and how they exchange information.

These scopes can overlap within a single reasoning process. Action planning can use solution inference under a composed model, and selecting a reasoner can also determine which operation runs next.

Table 1: Five scopes of inference in two families.

Inference scope	Reusable structure available in advance	Constructed for this problem resolved at test time
<i>Constructing the problem and its solution.</i> These scopes leave open the candidate solution, the model that defines the problem, or the actions that move the world toward the goal.		
Solution inference <i>Which candidate?</i>	An evaluator or distribution that guides search	A candidate under the fixed model. <i>How:</i> Sample, compare, search, and refine.
Model composition <i>Which task model?</i>	Compatible models, factors, and interfaces	The model that defines the current problem. <i>How:</i> Select, ground, and compose factors.
Action planning <i>Which actions reach the goal?</i>	Action-conditioned dynamics and reusable skills	An action sequence expected to reach the desired outcome. <i>How:</i> Predict consequences, search for actions, and replan as needed.
<i>Constructing the computation.</i> These scopes leave open which operation runs next and which reasoners contribute.		
Procedure control <i>Which operation and for how long?</i>	Solvers, verifiers, and mechanisms for estimating competence and cost	The next reasoning operation and when to stop. <i>How:</i> Choose the next operation, allocate computation, and decide when to stop.
Reasoner organization <i>Which reasoners and connections?</i>	Differentiated reasoners and reusable coordination mechanisms	The participating reasoners, their roles, and their connections. <i>How:</i> Select reasoners, assign roles, and route information.

4 Solution inference: Search a learned solution space

In solution inference, learning provides an evaluator that scores proposed solutions. Given the current problem, a sampler or optimizer uses this evaluator to compare candidates and revise them toward a better solution. The evaluator remains fixed during this search, while the candidate changes.

A major advantage of solution inference is that verification is often easier than generation [6]. In the coffee task, a direct predictor must produce a suitable placement from the kitchen scene. An evaluator can judge familiar properties of a proposed placement, such as stability and reachability, without learning how to produce the placement itself. A model that learns these simpler judgments can generalize better to unfamiliar scenes than one trained to predict a complete solution. Search uses them to construct a placement for the current scene.

An energy-based model [7] is one way to implement solution inference. A neural network learns an energy landscape over the space of possible solutions, assigning low energy to correct candidates [8]. Langevin dynamics [9] performs inference by iteratively refining a sample along the negative energy gradient while adding random noise. By varying the number of sampling steps, we can trade off sample quality against computation time. We can initialize inference either from random noise or from a generated proposal. We can also adapt inference to satisfy specific constraints and obtain desired solutions. For instance, we can implement image inpainting by fixing the values of observed pixels and letting inference generate the rest of the image.

This formulation also lets a model spend more computation on harder reasoning problems. In iterative reasoning through energy minimization [10], each optimization step revises a candidate answer under a learned energy function. A model trained to predict shortest-path distances on graphs with up to ten nodes achieves lower error on fifteen-node graphs than the feedforward and iterative baselines tested. The verifier-generator analysis shows how recognizing a solution can require far less computation than constructing one.

A language model can also be used in this way. The negative log probability it assigns to a candidate sequence acts as an energy. We can then perform solution inference through iterative sampling [11] that favors lower-energy responses. This procedure substantially improves reasoning performance over ordinary sampling from the base language model. On several reasoning tasks, it performs comparably to versions of the same models finetuned with reinforcement learning, without any additional training.

The effectiveness of this search depends on how training shapes the geometry of the energy landscape. A sequence of landscapes [12] can guide inference from broad, smooth basins toward sharper constraints, helping the solver approach a solution before resolving its finer details. Equilibrium Matching [13] instead learns a single smoothed energy landscape for optimization.

5 Model composition: Compose the model required by the problem

A new task may require a combination of goals and constraints that no single learned model represents. Model composition lets the system construct a task model from reusable factors after the task is known. Each factor evaluates a particular aspect of a candidate solution. Inference selects and combines the relevant factors to guide the search for a solution.

For the coffee task, the robot can combine factors that favor an upright mug, clearance from nearby dishes, stable support on the table, and reachability for the person. A support factor can evaluate the mug and its supporting surface, while a clearance factor evaluates the mug and nearby objects. Neither factor needs to model the entire kitchen arrangement. By composing these local judgments, the robot can evaluate an arrangement it has never seen as a whole.

Composition provides a way to program a new task using factors learned from earlier data. Diffusion-CCSP [14] learns factors for constraint types such as collision avoidance and spatial relations. A task specifies a graph connecting these factors to the object poses they constrain. Adding

their energies defines a joint criterion, allowing inference to search for poses that satisfy unfamiliar combinations of familiar constraints. Set It Up! [15] builds on this idea by using a language model to construct relation graphs from instructions, examples, and a task sketch. The graph specifies which learned factors guide the arrangement.

Visual factors can support both generating a scene and understanding one. In compositional visual generation [16], learned factors for individual concepts are combined to specify what an image should contain. For compositional scene understanding [17], inference searches for the object locations or attributes under which the combined factors best explain an observed image. The learned factors remain reusable, while the inferred locations and attributes describe the particular scene.

Composing additional instances of a learned factor can also extend inference to larger problems than those used in training. In Compositional Generative Inverse Design [18], factors trained on trajectories of two interacting bodies are composed to design initial conditions for systems with four or eight bodies. Each factor evaluates the motion of a pair within the same global trajectory. Inference jointly revises the initial conditions and trajectory under the combined dynamics factors and the design objective. The larger system does not have to be learned as a whole.

The same principle applies to reasoning. For Boolean satisfiability, clause-level factors [19] can be learned and their composition refined on small problems, then composed into larger instances at test time. With training instances of at most twenty variables, this approach solves more fifty-variable problems than the neural baselines tested. Familiar local constraints can therefore support search over a larger, unfamiliar problem.

Factors can also draw on knowledge from models pretrained for different purposes. Iterative Consensus [20] combines a pretrained generator with factors supplied by separately learned evaluators. For image generation, these factors judge whether candidate images match a requested description. Product of Experts for Visual Generation [21] combines factors derived from visual models with constraints from graphics or physics. An appearance factor can favor realistic video, while a motion factor favors agreement with a simulated trajectory. The full task is expressed by selecting and combining the relevant factors at inference time.

The vocabulary of factors does not need to be manually designed. Learning can discover factors in an unsupervised manner [22]. It can also infer a new task factor from demonstrations [23] while leaving a pretrained generator unchanged. A newly learned factor can then be combined with earlier ones to express further tasks.

The combined factors must be paired with an inference procedure suited to them. For example, summing the scores of diffusion factors and running standard reverse diffusion does not in general sample the intended product distribution. Reduce, Reuse, Recycle [24] develops sampling procedures to address this mismatch. Effective composition depends on both the reusable factors and the procedure used to reconcile them.

For decision making, factors can express both what should happen and what actions can cause. Objective factors represent goals, rewards, and constraints, while dynamics factors capture the behavior of objects, agents, and their interactions. Action planning uses their combination to search for actions whose predicted consequences satisfy the objective.

6 Action planning: Plan through possible futures

Composing a desired outcome with a model of what can happen defines a decision problem, but it does not determine how the agent should act. In the coffee task, the robot must still choose a grasp, lift the mug clear of nearby dishes, carry it without spilling, and place it at the selected location. Planning leaves this action sequence open. The planner predicts the consequences of possible actions and searches for a sequence expected to reach the desired state.

A direct policy resolves the action choice through a learned mapping from observations and goals. When a new situation falls outside the regularities captured by this mapping, the policy may fail

even if the relevant dynamics and skills are familiar. Planning keeps the action sequence available for search under the current task model. Behavior initially constructed through search can later be absorbed into a policy or skill when it recurs.

Planning does not always operate directly on individual motor commands. It can search over sub-goals, intermediate states, or complete trajectories, depending on what must be decided for the current task. Searching over intermediate states under learned dynamics [25] allowed a planner to route around obstacles not encountered during training. A generative model of complete state-action trajectories [26] provides a space of possible behaviors in which new rewards, endpoints, or constraints can guide search. In both cases, learning captures what recurs across tasks while search resolves the route required by the task at hand.

Predictive control repeats this search as the world changes. For locomotion, a learned trajectory prior can be combined with new rewards and constraints [27] to adapt behavior to the current task. A task planner can also be composed with action-conditioned dynamics [28] to improve the feasibility of trajectories that pursue the current goal.

For longer-horizon tasks, video provides an intermediate representation between a high-level goal and low-level control. It can describe how the scene should change while an embodiment-specific model determines how a particular robot makes that change happen. A learned video model can propose a task-level visual trajectory [29] from which actions are extracted. Search can compare and connect these proposals [30] into longer-horizon plans.

Different models can also divide the problem across levels of abstraction. We can compose language, video, and action models [31] so that language proposes high-level steps, video grounds them in the scene, and an action model tests their physical executability. The components provide complementary views of the same evolving future. Planning searches for a course of action supported at every level, using the structure supplied by model composition.

Action proposal, consequence prediction, and search can also be separated within a planner [32]. A vision-language model specifies a sequence of action primitives, which a low-level controller translates into robot actions. An action-conditioned world model predicts their consequences, and global correction and local search revise the proposals for the current task. The same verification-generation separation that motivates solution inference therefore reappears in sequential decision making. Judging whether an imagined consequence makes progress can be easier than generating the entire successful behavior at once.

7 Procedure control: Control the reasoning process

Generalization by construction also applies to the computation used to solve a problem. An unfamiliar task may require a new sequence of familiar reasoning operations or a different combination of specialists. By leaving these choices open to inference, a system can reuse learned capabilities to construct a reasoning process suited to the current problem.

Results from previous rounds of inference can guide what computation the system performs next. Procedure control combines this feedback with the current setting. A failed verification can identify which parts of a candidate to preserve and which to revise, while a successful check can let the system stop reasoning and begin execution. The system can also estimate task difficulty from these results and adjust the sequence and amount of computation accordingly.

For the robot, an uncertain estimate of the mug’s pose may call for another perception step before planning. If verification identifies a problem with the grasp, the next round of planning can focus on revising that part of the trajectory. If the trajectory passes verification, the controller can proceed to execution. During execution, the model’s estimated likelihood of the current plan [33] can provide further feedback for deciding whether to continue or replan from the existing trajectory.

Verification can also determine where the next round of generation begins. If a verifier identifies where a generated trajectory first goes wrong, the system can preserve a valid prefix and resume

generation from it [34]. The next round can then focus on extending a partial solution that has already passed verification. The generate–verify–restart loop stays fixed, while feedback determines which prefix to expand and when to stop. On Keys-and-Doors puzzles with solution paths longer than those in training, this procedure improves success over sampling complete rollouts at the same generation budget. Symbolic verification identifies the prefixes that can be retained, and every solved episode in this held-out setting comes from a restarted branch.

Repeated failures or slow progress can indicate that the current task is harder than expected. The controller can then devote more computation to unresolved decisions or try a different reasoning operation. The choice of model size and numerical precision [35] also affects both answer quality and latency. For an embodied system, the value of another reasoning step depends on whether it can improve the action before that action is needed.

8 Reasoner organization: Assemble the reasoning system

Procedure control chooses the next operation within an already assembled system. Reasoner organization leaves the assembly itself open. Inference may determine which specialists and tools participate, what roles they take on, what state is shared, and how information moves among them. Model composition constructs the task model that defines what counts as a solution. Reasoner organization assembles the specialists that supply or evaluate the relevant information.

One benefit of using multiple reasoners is diversity in how they approach a problem. A single reasoning process can become stuck repeatedly revising an unproductive candidate. Different reasoners can explore alternative candidates, assumptions, or strategies. By exchanging and debating their proposals [36], they can use one another’s progress to redirect their own search. This gives the system ways to move beyond an unproductive line of reasoning on a hard problem. The benefit depends on preserving useful differences among reasoners and evaluating the alternatives they propose.

A second benefit is the ability to divide a complex task into simpler subproblems. In the coffee task, separate specialists can estimate the mug’s pose, choose a stable grasp, and plan a collision-free route. Each specialist reasons about a narrower problem using knowledge relevant to that part of the task. Their decisions must then be coordinated so that the grasp and route are compatible. This decomposition can reduce the difficulty faced by each reasoner and let the system construct a solution to an unfamiliar task from familiar local capabilities. Verification can also be decomposed by assigning reasoners to check different aspects of a candidate [37] and combining their judgments.

These benefits are possible even when the participants and coordination rules are fixed before deployment. Training can produce complementary generator and critic behaviors [38] that preserve diverse reasoning chains and are reused across problems. In such systems, inference uses the interaction among reasoners to solve the current problem. Leaving the organization open adds the ability to choose which reasoners participate, how the task is divided, and how their contributions are combined for the problem at hand.

In the coffee task, suppose a single planner repeatedly fails to find a reliable grasp. The system can then assign several reasoners to explore different grasp strategies and ask another to compare their proposals. If the proposals still fail verification, it can recruit a grasp specialist and route that specialist’s feedback to the proposing reasoners. The goal and task model remain fixed while inference changes the participants, their responsibilities, and the information they exchange.

Reusable incentives and local rules can govern this adaptation by determining when reasoners participate and how their contributions are valued, allowing a task-specific organization to emerge without a central scheduler. For example, auctions can select a problem-dependent sequence of specialists [39] from a fixed population. The population stays fixed during evaluation, while economic selection reshapes it across optimization episodes. Selecting who acts also determines what computation runs next, so this mechanism spans procedure control and reasoner organization.

9 Conclusion: Generalization as construction

The ultimate goal of artificial intelligence is to build systems that generalize reliably to unfamiliar situations. Generalization as construction offers a way to solve globally unfamiliar problems by breaking them down into simpler, locally familiar parts. Learning captures recurring structure in the environment, while inference uses it to construct a solution for the current problem.

As a robot repeatedly interacts with its environment, learning can absorb the parts of problem solving that recur. A behavior first constructed through planning can become a reusable skill, and a useful sequence of reasoning operations can become a learned procedure. These learned skills and procedures give inference new pieces to combine, allowing later problems to be solved without repeating every step of the original search. The amortization boundary can then move with experience. The robot can solve familiar problems faster and use what it has learned to construct solutions to increasingly complex ones.

A Illustrative work by scope

A.1 Solution inference (Section 4)

- [9] *Implicit Generation and Generalization in Energy-Based Models*. One learned landscape supports generation, refinement, and variable inference effort.
- [10] *Learning Iterative Reasoning through Energy Minimization*. Answers are represented as minima, with computation adjustable to problem difficulty.
- [12] *Learning Iterative Reasoning through Energy Diffusion*. A sequence of learned landscapes supports continuous optimization, discrete reasoning, and planning.
- [40] *Energy-Based Transformers are Scalable Learners and Thinkers*. A shared energy scores input–candidate compatibility across modalities, and inference finds candidates by minimizing it.
- [13] *Equilibrium Matching*. A time-independent implicit landscape supports optimization-based generation.
- [11] *Reasoning with Sampling*. A frozen language model’s sequence likelihood defines a sharpened target distribution for iterative sampling.
- [41] *Generative Modeling via Drifting*. Iterative distribution refinement is moved into training when one-step generation is sufficient.

A.2 Model composition (Section 5)

- [14] *Compositional Diffusion-Based Continuous Constraint Solvers*. Learned constraint factors compose to solve new constraint graphs, including graphs with more objects.
- [15] *Set It Up!*. Task-specific relation graphs select learned factors that jointly determine object poses.
- [16] *Compositional Visual Generation and Inference with Energy-Based Models*. Concept factors are composed through logical operations for generation and inference.
- [17] *Compositional Scene Understanding through Inverse Generative Modeling*. Object locations or attributes are inferred by evaluating an observed image under combined visual factors.
- [18] *Compositional Generative Inverse Design*. Factors for two-body trajectories compose to design systems with four or eight bodies.
- [19] *Generalizable Reasoning through Compositional Energy Minimization*. Learned factors combine to define larger reasoning problems and incorporate new constraints.
- [20] *Composing Ensembles of Pre-trained Models via Iterative Consensus*. Learned evaluator factors guide candidate solutions from a pretrained generator.
- [21] *Product of Experts for Visual Generation*. Factors from visual models, evaluators, and physical constraints define a combined target distribution.
- [22] *Unsupervised Learning of Compositional Energy Concepts*. Reusable energy factors are discovered without concept labels.
- [23] *Few-Shot Task Learning through Inverse Generative Modeling*. A new task factor is inferred from demonstrations without changing pretrained model weights.

- [24] *Reduce, Reuse, Recycle*. Sampling procedures address the mismatch between standard reverse diffusion and composed target distributions.
- [42] *Learning to Compose Visual Relations*. Named relations are represented as reusable energy factors.
- [43] *Compositional Visual Generation with Composable Diffusion Models*. Diffusion factors are combined during denoising.
- [44] *PoCo*. Factors from heterogeneous robot policies and analytic costs compose at inference time.
- [45] *Compositional Generative Modeling: A Single Model Is Not All You Need*. A broader perspective on constructing new task distributions from reusable learned factors.

A.3 Action planning (Section 6)

- [25] *Model-Based Planning with Energy-Based Models*. Intermediate state trajectories are inferred through learned dynamics.
- [26] *Planning with Diffusion for Flexible Behavior Synthesis*. Complete trajectories form a generative planning representation that can be guided at inference time.
- [31] *Compositional Foundation Models for Hierarchical Planning*. Language, video, and action models are composed across abstraction levels.
- [46] *Inference-Time Enhancement of Generative Robot Policies via Predictive World Modeling*. A frozen policy proposes actions that a predictive model ranks or refines.
- [47] *DriftWorld: Fast World Modeling through Drifting*. Single-pass action-conditioned rollouts let a planner compare many policy proposals.
- [48] *Test-Time Graph Search for Goal-Conditioned Reinforcement Learning*. Graph search turns a frozen local policy into a long-horizon planning tool.
- [27] *Flexible Locomotion Learning with Diffusion Model Predictive Control*. A learned trajectory prior is optimized under new rewards and projected constraints.
- [28] *Model-Based Diffusion Sampling for Predictive Control in Offline Decision Making*. A diffusion planner and dynamics model are composed during predictive control.
- [29] *Learning Universal Policies via Text-Guided Video Generation*. Language goals become visual plans from which actions are extracted.
- [30] *Video Language Planning*. Tree search composes short predicted videos into long-horizon visual plans.
- [49] *Large Video Planner*. Large-scale video pretraining supports zero-shot plans retargeted to robots.
- [50] *World Action Models are Zero-shot Policies (DreamZero)*. Joint video–action prediction yields a real-time closed-loop policy, illustrating a more amortized alternative to explicit search.
- [32] *World Action Planner: Generalizable Decision-Making with Action-Conditioned World Models*. Action proposals are revised through search over imagined consequences.
- [51] *World-in-World*. Closed-loop evaluation separates visual realism from controllability and decision utility.
- [52] *Structured 4D Latent Predictive Model for Robot Planning*. Structured three-dimensional latents preserve geometry for prediction and control.
- [53] *Self-Improving Loops for Visual Robotic Planning*. Trajectories produced by a visual planner are consolidated into improved behavior.
- [54] *COMBO*. Per-agent action conditions compose into a video world model whose predicted consequences guide cooperative planning under partial observation.

A.4 Procedure control (Section 7)

- [33] *Adaptive Online Replanning with Diffusion Models*. Estimated plan likelihood determines when execution continues and when replanning bootstraps from the existing trajectory.
- [34] *Temporal Backtracking Search for Test-time Generative Video Reasoning*. Verifier evidence determines which valid prefix is preserved, expanded, or abandoned.
- [55] *Inference-Time Scaling of Diffusion Models through Classical Search*. Adaptive backtracking allocates computation across diffusion paths.
- [35] *Win Fast or Lose Slow*. Model size and numerical precision affect answer quality and latency in time-sensitive tasks.

A.5 Reasoner organization (Section 8)

- [36] *Improving Factuality and Reasoning in Language Models through Multiagent Debate*. Identical model instances reason through a fixed debate protocol, establishing interaction as a reasoning substrate.
- [37] *Multi-Agent Verification*. A fixed, domain-specific set of aspect verifiers combines complementary judgments at test time.
- [38] *Multiagent Finetuning: Self Improvement with Diverse Reasoning Chains*. Separate generation and critic agents are trained on different interaction data to preserve diverse reasoning chains.
- [56] *SLM-MUX: Orchestrating Small Language Models for Reasoning*. Pre-deployment search fixes a complementary model subset, placing participant choice on the amortized side of the boundary.
- [39] *Economy of Minds*. Auctions select a problem-dependent sequence of specialists, while economic selection reshapes the reusable population across optimization episodes.

References

- [1] John McCarthy. Programs with Common Sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, 1959. <https://purl.stanford.edu/tf739wh6057>
- [2] Richard E. Fikes and Nils J. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2(3–4):189–208, 1971. [doi:10.1016/0004-3702\(71\)90010-5](https://doi.org/10.1016/0004-3702(71)90010-5)
- [3] Samuel J. Gershman and Noah D. Goodman. Amortized Inference in Probabilistic Reasoning. In *Proceedings of the 36th Annual Conference of the Cognitive Science Society*, pages 517–522, 2014. <https://gershmanlab.com/pubs/GershmanGoodman14.pdf>
- [4] Yoshihiko Futamura. Partial Evaluation of Computation Process—An Approach to a Compiler-Compiler. *Systems, Computers, Controls*, 2(5):45–50, 1971. Reprinted in *Higher-Order and Symbolic Computation*, 12(4):381–391, 1999. [doi:10.1023/A:1010095604496](https://doi.org/10.1023/A:1010095604496)
- [5] Yilun Du. Learning Generalizable Systems by Learning Composable Energy Landscapes. Ph.D. thesis, Massachusetts Institute of Technology, 2024. <https://dspace.mit.edu/entities/publication/7165045b-d9de-4fa7-a9de-8507f0c2908c>
- [6] Stephen A. Cook. The Complexity of Theorem-Proving Procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing (STOC)*, pages 151–158, 1971. [doi:10.1145/800157.805047](https://doi.org/10.1145/800157.805047)
- [7] Energy-Based Models. Overview website. <https://energy-based-model.github.io/>
- [8] Yann LeCun, Sumit Chopra, Raia Hadsell, Marc’Aurelio Ranzato, and Fu Jie Huang. A Tutorial on Energy-Based Learning. In *Predicting Structured Data*. MIT Press, 2006. <http://yann.lecun.com/exdb/publis/pdf/lecun-06.pdf>
- [9] Yilun Du and Igor Mordatch. Implicit Generation and Modeling with Energy Based Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. [arXiv:1903.08689](https://arxiv.org/abs/1903.08689).
- [10] Yilun Du, Shuang Li, Joshua B. Tenenbaum, and Igor Mordatch. Learning Iterative Reasoning through Energy Minimization. In *International Conference on Machine Learning (ICML)*, 2022. [arXiv:2206.15448](https://arxiv.org/abs/2206.15448).
- [11] Aayush Karan and Yilun Du. Reasoning with Sampling: Your Base Model is Smarter Than You Think. In *International Conference on Learning Representations (ICLR)*, 2026. [arXiv:2510.14901](https://arxiv.org/abs/2510.14901).
- [12] Yilun Du, Jiayuan Mao, and Joshua B. Tenenbaum. Learning Iterative Reasoning through Energy Diffusion. In *International Conference on Machine Learning (ICML)*, 2024. [arXiv:2406.11179](https://arxiv.org/abs/2406.11179).
- [13] Runqian Wang and Yilun Du. Equilibrium Matching: Generative Modeling with Implicit Energy-Based Models. *arXiv preprint*, 2025. [arXiv:2510.02300](https://arxiv.org/abs/2510.02300).
- [14] Zhutian Yang, Jiayuan Mao, Yilun Du, Jiajun Wu, Joshua B. Tenenbaum, Tomás Lozano-Pérez, and Leslie Kaelbling. Compositional Diffusion-Based Continuous Constraint Solvers. In *Conference on Robot Learning (CoRL)*, 2023. [arXiv:2309.00966](https://arxiv.org/abs/2309.00966).
- [15] Yiqing Xu, Jiayuan Mao, Yilun Du, Tomás Lozano-Pérez, Leslie Kaelbling, and David Hsu. “Set It Up!”: Functional Object Arrangement with Compositional Generative Models. In *Robotics: Science and Systems (RSS)*, 2024. [arXiv:2405.11928](https://arxiv.org/abs/2405.11928).
- [16] Yilun Du, Shuang Li, and Igor Mordatch. Compositional Visual Generation with Energy Based Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. [arXiv:2004.06030](https://arxiv.org/abs/2004.06030).

- [17] Yanbo Wang, Justin Dauwels, and Yilun Du. Compositional Scene Understanding through Inverse Generative Modeling. In *International Conference on Machine Learning (ICML)*, 2025. [arXiv:2505.21780](#).
- [18] Tailin Wu, Takashi Maruyama, Long Wei, Tao Zhang, Yilun Du, Gianluca Iaccarino, and Jure Leskovec. Compositional Generative Inverse Design. In *International Conference on Learning Representations (ICLR)*, 2024. [arXiv:2401.13171](#).
- [19] Alexandru Oarga and Yilun Du. Generalizable Reasoning through Compositional Energy Minimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. [arXiv:2510.20607](#).
- [20] Shuang Li, Yilun Du, Joshua B. Tenenbaum, Antonio Torralba, and Igor Mordatch. Composing Ensembles of Pre-trained Models via Iterative Consensus. In *International Conference on Learning Representations (ICLR)*, 2023. [arXiv:2210.11522](#).
- [21] Yunzhi Zhang, Carson Murtuza-Lanier, Zizhang Li, Yilun Du, and Jiajun Wu. Product of Experts for Visual Generation. In *International Conference on Learning Representations (ICLR)*, 2026. [arXiv:2506.08894](#).
- [22] Yilun Du, Shuang Li, Yash Sharma, Joshua B. Tenenbaum, and Igor Mordatch. Unsupervised Learning of Compositional Energy Concepts. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. [arXiv:2111.03042](#).
- [23] Aviv Netanyahu, Yilun Du, Antonia Bronars, Jyotish Pari, Joshua B. Tenenbaum, Tianmin Shu, and Pulkit Agrawal. Few-Shot Task Learning through Inverse Generative Modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. [arXiv:2411.04987](#).
- [24] Yilun Du, Conor Durkan, Robin Strudel, Joshua B. Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Grathwohl. Reduce, Reuse, Recycle: Compositional Generation with Energy-Based Diffusion Models and MCMC. In *International Conference on Machine Learning (ICML)*, 2023. [arXiv:2302.11552](#).
- [25] Yilun Du, Toru Lin, and Igor Mordatch. Model-Based Planning with Energy-Based Models. In *Conference on Robot Learning (CoRL)*, 2019. [arXiv:1909.06878](#).
- [26] Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with Diffusion for Flexible Behavior Synthesis. In *International Conference on Machine Learning (ICML)*, 2022. [arXiv:2205.09991](#).
- [27] Runhan Huang, Haldun Balim, Heng Yang, and Yilun Du. Flexible Locomotion Learning with Diffusion Model Predictive Control. In *International Conference on Robotics and Automation (ICRA)*, 2026. [arXiv:2510.04234](#).
- [28] Haldun Balim, Na Li, and Yilun Du. Model-Based Diffusion Sampling for Predictive Control in Offline Decision Making. In *International Conference on Machine Learning (ICML)*, 2026. [arXiv:2512.08280](#).
- [29] Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning Universal Policies via Text-Guided Video Generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. [arXiv:2302.00111](#).
- [30] Yilun Du, Sherry Yang, Pete Florence, Fei Xia, Ayzaan Wahid, Brian Ichter, Pierre Sermanet, Tianhe Yu, Pieter Abbeel, Joshua B. Tenenbaum, Leslie Kaelbling, Andy Zeng, and Jonathan Tompson. Video Language Planning. In *International Conference on Learning Representations (ICLR)*, 2024. [arXiv:2310.10625](#).
- [31] Anurag Ajay, Seungwook Han, Yilun Du, Shuang Li, Abhi Gupta, Tommi Jaakkola, Joshua B. Tenenbaum, Leslie Kaelbling, Akash Srivastava, and Pulkit Agrawal. Compositional Foundation Models for Hierarchical Planning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. [arXiv:2309.08587](#).
- [32] Xiangcheng Zhang and Yilun Du. World Action Planner: Generalizable Decision-Making with Action-Conditioned World Models. *arXiv preprint*, 2026. [arXiv:2607.27599](#).
- [33] Siyuan Zhou, Yilun Du, Shun Zhang, Mengdi Xu, Yikang Shen, Wei Xiao, Dit-Yan Yeung, and Chuang Gan. Adaptive Online Replanning with Diffusion Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. [arXiv:2310.09629](#).
- [34] Sejoon Jun, Zheng Ding, Huangyuan Su, Weirui Ye, and Yilun Du. Temporal Backtracking Search for Test-time Generative Video Reasoning. In *International Conference on Machine Learning (ICML)*, 2026. [arXiv:2606.13861](#).

- [35] Hao Kang, Qingru Zhang, Han Cai, Weiyan Xu, Tushar Krishna, Yilun Du, and Tsachy Weissman. Win Fast or Lose Slow: Balancing Speed and Accuracy in Latency-Sensitive Decisions of LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. [arXiv:2505.19481](#).
- [36] Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving Factuality and Reasoning in Language Models through Multiagent Debate. In *International Conference on Machine Learning (ICML)*, 2024. [arXiv:2305.14325](#).
- [37] Shalev Lifshitz, Sheila A. McIlraith, and Yilun Du. Multi-Agent Verification: Scaling Test-Time Compute with Multiple Verifiers. In *Conference on Language Modeling (CoLM)*, 2025. [arXiv:2502.20379](#).
- [38] Vighnesh Subramaniam, Yilun Du, Joshua B. Tenenbaum, Antonio Torralba, Shuang Li, and Igor Mordatch. Multiagent Finetuning: Self Improvement with Diverse Reasoning Chains. In *International Conference on Learning Representations (ICLR)*, 2025. [arXiv:2501.05707](#).
- [39] Zhenting Qi, Huangyuan Su, Ao Qu, Chenyu Wang, Yu Yao, Han Zheng, Kushal Chattopadhyay, Guowei Xu, Zihan Wang, Weirui Ye, Vijay Janapa Reddi, Ju Li, Paul Pu Liang, Himabindu Lakkaraju, Sham Kakade, and Yilun Du. Economy of Minds: Emerging Multi-Agent Intelligence with Economic Interactions. *arXiv preprint*, 2026. [arXiv:2606.02859](#).
- [40] Alexi Gladstone, Ganesh Nanduru, Md Mofijul Islam, Peixuan Han, Hyeonjeong Ha, Aman Chadha, Yilun Du, Heng Ji, Jundong Li, and Tariq Iqbal. Energy-Based Transformers are Scalable Learners and Thinkers. In *International Conference on Learning Representations (ICLR)*, 2026. [arXiv:2507.02092](#).
- [41] Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative Modeling via Drifting. *arXiv preprint*, 2026. [arXiv:2602.04770](#).
- [42] Nan Liu, Shuang Li, Yilun Du, Joshua B. Tenenbaum, and Antonio Torralba. Learning to Compose Visual Relations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. [arXiv:2111.09297](#).
- [43] Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B. Tenenbaum. Compositional Visual Generation with Composable Diffusion Models. In *European Conference on Computer Vision (ECCV)*, 2022. [arXiv:2206.01714](#).
- [44] Lirui Wang, Alan Zhao, Yilun Du, Edward Adelson, and Russ Tedrake. PoCo: Policy Composition from and for Heterogeneous Robot Learning. In *Robotics: Science and Systems (RSS)*, 2024. [arXiv:2402.02511](#).
- [45] Yilun Du and Leslie Kaelbling. Compositional Generative Modeling: A Single Model is Not All You Need. In *International Conference on Machine Learning (ICML)*, 2024. [arXiv:2402.01103](#).
- [46] Han Qi, Haocheng Yin, Aris Zhu, Yilun Du, and Heng Yang. Inference-Time Enhancement of Generative Robot Policies via Predictive World Modeling. *IEEE Robotics and Automation Letters (RA-L)*, 2026. [arXiv:2502.00622](#).
- [47] Susie Lu, Haonan Chen, Weirui Ye, and Yilun Du. DriftWorld: Fast World Modeling through Drifting. *arXiv preprint*, 2026. [arXiv:2607.15065](#).
- [48] Evgenii Opryshko, Junwei Quan, Claas Voelcker, Yilun Du, and Igor Gilitschenski. Test-Time Graph Search for Goal-Conditioned Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2026. [arXiv:2510.07257](#).
- [49] Boyuan Chen, Tianyuan Zhang, Haoran Geng, Kiwhan Song, Caiyi Zhang, Peihao Li, William T. Freeman, Jitendra Malik, Pieter Abbeel, Russ Tedrake, Vincent Sitzmann, and Yilun Du. Large Video Planner Enables Generalizable Robot Control. *arXiv preprint*, 2025. [arXiv:2512.15840](#).
- [50] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, et al.. World Action Models are Zero-shot Policies. *arXiv preprint*, 2026. [arXiv:2602.15922](#).
- [51] Jiahao Zhang, Muqing Jiang, Nanru Dai, et al.. World-in-World: World Models in a Closed-Loop World. In *International Conference on Learning Representations (ICLR)*, 2026. [arXiv:2510.18135](#).
- [52] Zhiyi Li, Peilin Wu, Xiaoshen Han, Ruojin Cai, and Yilun Du. Structured 4D Latent Predictive Model for Robot Planning. In *International Conference on Machine Learning (ICML)*, 2026. [arXiv:2607.01166](#).
- [53] Calvin Luo, Zilai Zeng, Mingxi Jia, Yilun Du, and Chen Sun. Self-Improving Loops for Visual Robotic Planning. In *International Conference on Learning Representations (ICLR)*, 2026. [arXiv:2506.06658](#).
- [54] Hongxin Zhang, Zeyuan Wang, Qiushi Lyu, Zheyuan Zhang, Sunli Chen, Tianmin Shu, Yilun Du, and Chuang Gan. COMBO: Compositional World Models for Embodied Multi-Agent Cooperation. In *International Conference on Learning Representations (ICLR)*, 2025. [arXiv:2404.10775](#).

- [55] Xiangcheng Zhang, Haowei Lin, Haotian Ye, James Zou, Jianzhu Ma, Yitao Liang, and Yilun Du. Inference-time Scaling of Diffusion Models through Classical Search. In *International Conference on Learning Representations (ICLR)*, 2026. [arXiv:2505.23614](#).
- [56] Chenyu Wang, Zishen Wan, Hao Kang, Emma Chen, Zhiqiang Xie, Tushar Krishna, Vijay Reddi, and Yilun Du. SLM-MUX: Orchestrating Small Language Models for Reasoning. In *International Conference on Learning Representations (ICLR)*, 2026. [arXiv:2510.05077](#).