Praxis

A Controlled Laboratory for Vision-Language-Action Policy Research

Chaoqi Liu[1,2]

[1]Harvard University [2]Stanford University

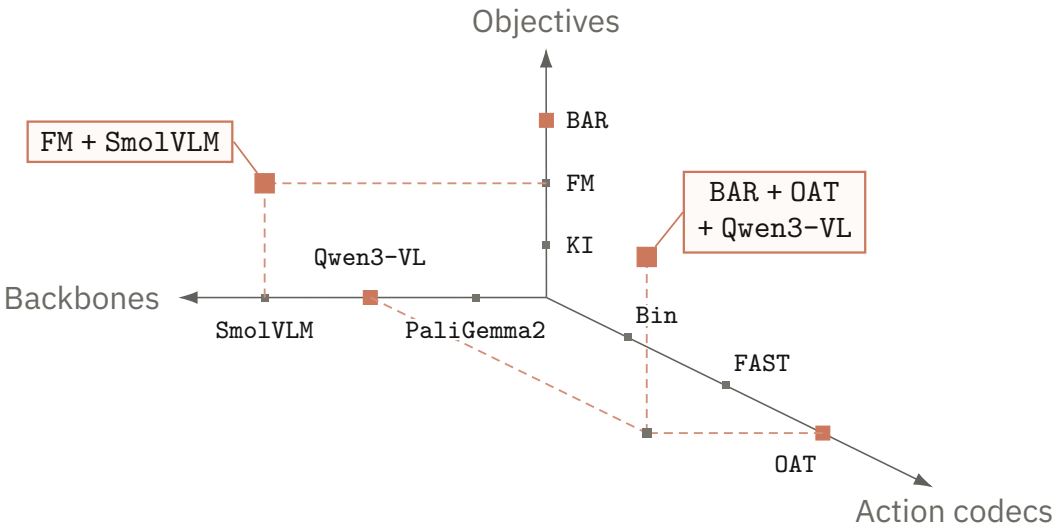

Figure 1: Praxis turns VLA ablation into a controlled coordinate system. Each policy is located by objective, VLM backbone, and action representation. The highlighted `BAR` + `OAT` + `Qwen3-VL` cell is a tokenized policy; the `FM` + `SmolVLM` point is a tokenless continuous policy. `praxis-vla` instantiates the coordinates, and `praxis-eval` supplies the evaluation boundary for asking what changed and what was held fixed.

Website: https://chaoqi-liu.com/praxis-vla

Code: Chaoqi-LIU/praxis-vla, Chaoqi-LIU/praxis-eval

Date: June 10, 2026

Abstract

Vision-language-action (VLA) policy research now spans pretrained multimodal backbones, discrete action tokenizers, continuous control objectives, and broad simulated benchmarks. Many comparisons, however, remain benchmark-centric: objective, backbone, tokenizer, decoding rule, normalization state, serving path, and benchmark adapter can change together before a rollout score is reported. Such scores are useful for tracking capability, but weak evidence for what caused a difference. We present Praxis, a research substrate for controlled VLA ablation. `praxis-vla` represents each policy as a matrix cell over objective, action representation, and vision-language model (VLM) backbone, so a study can name the coordinate it changes. `praxis-eval` moves benchmark semantics, execution mode, dependency isolation, and evaluation records outside policy families. This artifact paper reports no new architecture or leaderboard result; it defines notation, contracts, and executable boundaries for attributing differences in VLA experiments.

1 Introduction

Vision-language-action (VLA) policy research has moved from demonstrating language-conditioned robot control to asking which design choices make such policies reliable, scalable, and interpretable. Recent systems explore several routes from perception and language to action: autoregressive (AR) action-token generation (Brohan et al., 2022, 2023; Kim et al., 2024), generalist policy stacks (Octo Model Team et al., 2024), continuous flow-based control (Black et al., 2024; Intelligence et al., 2025), and action-tokenizer studies (Pertsch et al., 2025; Dong et al., 2026; Liu et al., 2026). At the same time, simulated benchmark suites have become the main reporting surface for progress across tasks and environments (Mandlekar et al., 2021; Yu et al., 2020; Liu et al., 2023; Nasiriany et al., 2024; Li et al., 2024; Shukla et al., 2025). As a result, VLA comparisons increasingly span objectives, backbones, action representations, and evaluators rather than variants of one architecture family.

Benchmark-centric progress is valuable for tracking capability, but limited as an explanation of why one VLA policy behaves differently from another. A rollout score summarizes a complete training, serving, and evaluation stack. That stack may change the policy objective, action tokenizer, vision-language model (VLM) backbone, image preprocessing, prompt format, action normalization, inference state, simulator dependency, and benchmark adapter at the same time. When these factors move together, a claim such as "this tokenizer is better" or "this backbone is stronger" is hard to interpret: the comparison may be between two private pipelines rather than two declared variables.

VLA ablation needs a laboratory that changes one declared variable while exposing the assumptions around it. A tokenizer study should vary the action representation while preserving the objective, backbone, normalization state, serving path, and evaluator semantics. A backbone study needs the same discipline around preprocessing, prompt format, cache behavior, and the action-head interface. Without that structure, readers cannot tell whether a reported difference comes from the variable named in the claim or from unreported movement elsewhere in the stack.

Praxis is designed for this role. The substrate has two parts: `praxis-vla` instantiates the policy design matrix, and `praxis-eval` supplies the evaluation boundary. The central abstraction is a *policy cell*. A tokenized policy is indexed as $\pi_{m,c,b}$, where m is the policy objective, c is the action representation, and b is the VLM backbone. Continuous cells use $c = \varnothing$ because they occupy the objective–backbone plane without a discrete tokenizer coordinate. The notation gives each implementation a declared coordinate before rollout performance is interpreted.

Fig. 1 previews this separation: tokenized and tokenless policies share the same coordinate language, while evaluation remains an explicit boundary rather than policy-specific glue.

Fig. 2 shows the indexing convention used by `praxis-vla`. The current release makes the abstraction concrete with block-wise autoregression (`BAR`), flow matching (`FM`), and knowledge insulation (`KI`) objectives; scalar, spectral, learned-latent, and ordered action representations; and `SmolVLM`, `Qwen3-VL`, and `PaliGemma2` backbones. These initial coordinates are deliberately concrete. They support one-axis studies: action representations can vary with objective and backbone fixed; backbones can vary with objective and action representation fixed; and objective comparisons can state explicitly whether the tokenizer coordinate is present.

`praxis-eval` completes the laboratory by controlling a second source of confounding. Benchmark glue, runtime dependencies, action contracts, and metric code can become hidden policy variables if each policy family owns its own evaluator. `praxis-eval` keeps benchmark semantics in benchmark drivers and exposes a narrow observation/action boundary to policies. The same boundary supports local or remote execution, allowing policy and simulator code to run in separate Python environments when dependencies conflict.

This first report defines the substrate rather than ranking its cells. It makes future experiments state the changed coordinate, the shared contracts, and the evidence needed to attribute a difference to a VLA design choice.

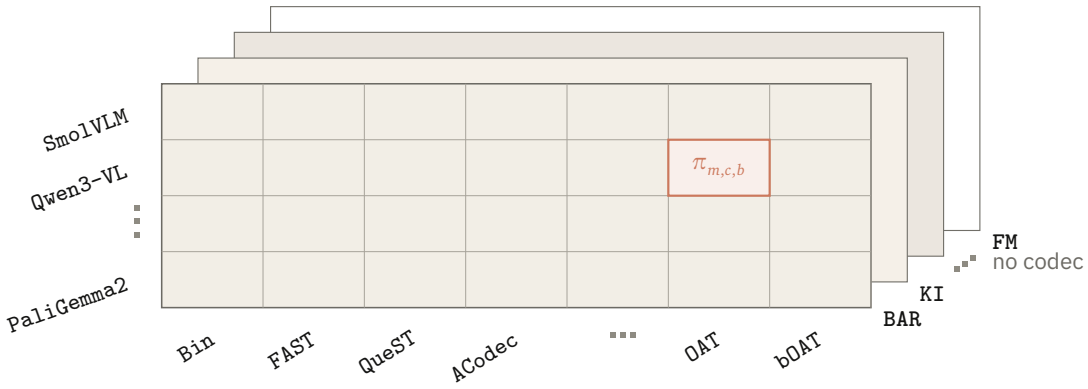

Figure 2: Formal policy-cell geometry for `praxis-vla`. Action codecs and VLM backbones define the tokenized-policy comparison plane, while policy objectives form stacked layers. A supported tokenized policy is a cell $\pi_{m,c,b}$; `FM` occupies a tokenless layer with no codec coordinate. The figure defines the indexing convention used throughout the report: moving along one grid direction changes one declared variable, and axis ellipses mark where additional backbones, action codecs, or objectives can enter.

The report contributes a controlled-design-space framing for VLA policy research; policy cells as declared units for one-axis ablations; an executable `praxis-vla` matrix over objectives, action representations, and VLM backbones; and a `praxis-eval` boundary separating benchmark semantics, runtime isolation, and evaluation records from policy families.

2 Principles for Controlled VLA Research

Controlled VLA research requires the artifact to separate the variable under study from the surrounding execution stack. A method comparison is rarely a comparison between losses, tokenizers, or model families alone. It also depends on temporal horizon, normalization statistics, prefix construction, action decode validity, simulator conventions, rollout state, and artifact loading. Praxis makes these assumptions part of the artifact before measurement. New objectives, backbones, codecs, and benchmarks enter through explicit coordinates and contracts, so a study compares declared variables rather than private pipelines.

We use four constraints to control these risks: open coordinates, shared contracts, axis-local ownership, and an evaluator boundary. Table 1 summarizes the threat each constraint addresses.

Open coordinates. The policy cell is the comparison unit in Praxis. A cell specifies the objective family, action representation, and VLM backbone, while the surrounding lab defines shared assumptions about observations, actions, artifacts, serving, and evaluation. This abstraction matters because the scientific claim usually concerns a difference between cells, not a single trained checkpoint. When a new coordinate is added, the intended change should be local to the axis being extended; the surrounding training, serving, artifact, and evaluation contracts are either reused or explicitly declared as changed. When the comparison is later reported, the reader can trace it to the named VLA design choice rather than to a new stack.

Shared contracts. Contracts protect interpretation by making non-model assumptions portable. Policy-side contracts record the observations, actions, temporal horizon, and language field expected by a policy. Artifact contracts carry the corresponding I/O description and normalization state with checkpoints and codecs. Evaluation-side contracts record the observation keys and action specification expected by a benchmark. These contracts do not make a future experiment fair by themselves; fairness still depends on the study protocol. They do, however, prevent central assumptions from

Principle	Threat to interpretation	Design response
Open coordinates	Two named policies may differ in objective, tokenizer, backbone, preprocessing, and evaluator at once	A policy cell names the intended coordinates, and additions enter the shared matrix rather than a new private stack
Shared contracts	Observation keys, action scale, horizon, dtype, and normalization can be hidden in scripts or checkpoints	Policy-facing I/O and normalization state travel with artifacts; evaluation action contracts remain benchmark-owned
Axis-local ownership	Objective-, codec-, and backbone-specific details can leak into unrelated parts of the stack	Objectives, action representations, and backbone adapters keep local semantics behind stable boundaries
Evaluation as a boundary	Simulator setup, task selection, success metrics, and runtime dependencies can silently alter the policy comparison	`praxis-eval` keeps benchmark semantics benchmark-owned while policies expose a narrow reset/act interface

Table 1: Controlled comparison as a design requirement. A VLA comparison is interpretable only when it states both the changed coordinate and the shared assumptions. Praxis encodes that requirement with named coordinates, portable contracts, axis-local modules, and an evaluator separated from policy code.

being implicit in a training script or hidden inside a checkpoint. For action-token studies, this point is especially important because a token sequence is only meaningful together with the action scale, horizon, decode rule, and validity assumptions under which it was produced.

Axis-local ownership. The axes are separated so local scientific semantics do not leak into unrelated parts of the stack. Objectives cover losses, supervision, and inference procedures; action representations cover encoding, decoding, horizons, prefixes, and schedules; backbone adapters cover multimodal preprocessing, prompt conventions, cache behavior, and attention geometry. Shared policy code handles rollout queues and episode state. This separation does not eliminate interactions between axes; it makes them inspectable by recording whether a study changed the objective, codec, backbone, evaluator boundary, or some combination of them.

Evaluation boundary. The same principle applies at rollout time: benchmark semantics are experimental assumptions. The evaluator owns task selection, simulator setup, rollout, metrics, artifacts, and runtime mode; the policy exposes a narrow reset/act interface. Sec. 4 develops this boundary in detail. New benchmarks attach to the evaluator, just as new objectives, backbones, and codecs attach to explicit policy coordinates. Future additions remain comparable when they reuse the surrounding contracts instead of arriving as isolated stacks.

3 The `praxis-vla` Policy Matrix

`praxis-vla` realizes the policy side of Sec. 2 by making coordinate tuples loadable and inspectable. It is not a proposal for one VLA architecture; it is a way to turn policy design choices into executable experimental objects. Tokenized policies occupy three axes: objective, action representation, and VLM backbone. Continuous `FM` policies occupy the objective–backbone plane with a null tokenizer coordinate because no discrete action representation is decoded at inference time. We use *action representation* for the scientific axis and *codec/tokenizer* for a concrete implementation that maps action chunks to policy targets.

Family	Supervision signal	Inference procedure	Question isolated by the cell
BAR	cross-entropy on scheduled action-token blocks	generate token blocks, then detokenize to actions	how a discrete representation behaves under VLM token prediction
KI	token cross-entropy plus stop-gradient-insulated continuous flow loss	condition a continuous expert on insulated VLM context	whether action tokens are useful as representation supervision
FM	flow-velocity regression in continuous action space	sample continuous action chunks directly	what remains when the discrete-token axis is removed

Table 2: Objective families in the policy matrix. praxis-vla separates objective families by what supervises the policy and what runs at inference time. Tokenized objectives can be paired with action representations; FM occupies the objective–backbone plane without a tokenizer coordinate.

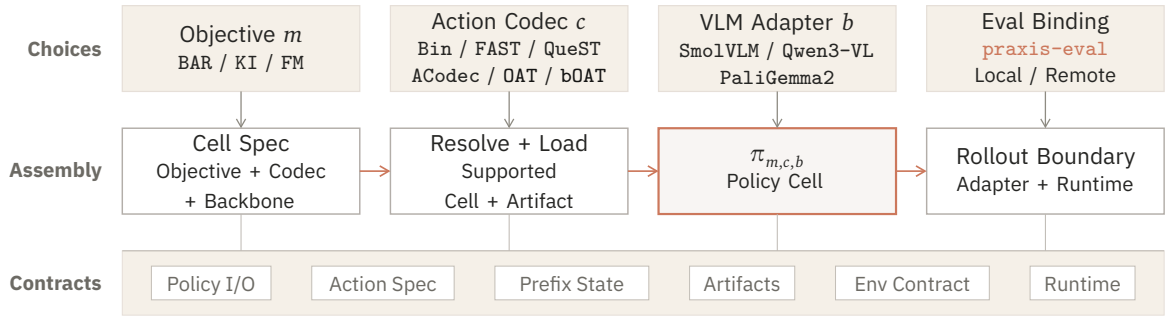

Figure 3: Local policy choices become an auditable cell. Objectives, action codecs, VLM adapters, and evaluation bindings are assembled into a supported policy cell. Shared contracts record policy I/O, action specification, prefix state, artifacts, environment contract, and runtime, so an intended axis change can be inspected without hiding construction assumptions.

Policy cell

A policy cell is the declared comparison unit in praxis-vla. Tokenized cells are indexed by objective m, action codec c, and backbone b, written $\pi_{m,c,b}$. Continuous FM cells are written $\pi_{\mathrm{FM},\varnothing,b}$. The notation states the intended comparison before measurement: change one declared variable while preserving the observation/action interface, normalization state, construction protocol, and evaluation boundary.

Objectives determine what supervision is applied, what runs at inference time, and how gradients reach the VLM. We start with objective families before turning to action representations and backbones.

Table 2 gives the compact view; the following subsections explain how these families become policy artifacts and what comparisons the remaining axes make possible.

3.1 From a Cell to a Study

A cell is useful only if the declared comparison remains attached to the trained policy through construction, loading, and rollout. In praxis-vla, artifact metadata records policy I/O, normalization state, and external codec checkpoints, so the changed variable can be read together with the assumptions held fixed.

Fig. 3 shows where objective, action representation, backbone, and evaluator binding are assem-

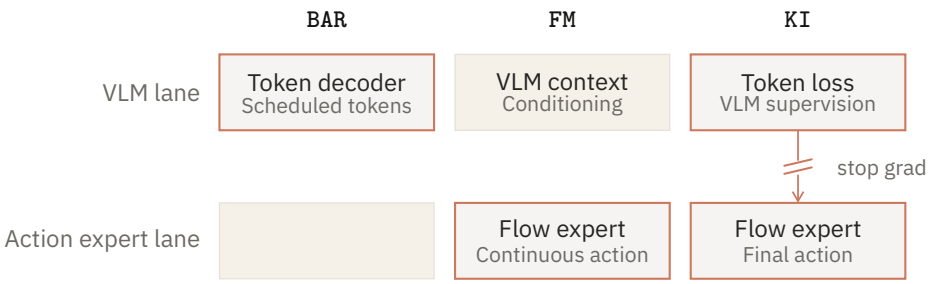

Figure 4: Objective families differ by which lane owns action generation. `BAR` uses the VLM as the scheduled action-token decoder. `FM` keeps final control in a continuous action expert conditioned on VLM context. `KI` uses token cross-entropy to shape the VLM lane, then routes final control through a stop-gradient-insulated flow expert. The figure emphasizes which lane receives supervision, which lane generates actions, and where gradients are allowed to flow.

bled into one policy cell, and where shared contracts record loading and rollout assumptions.

3.2 Objective Axis

The objective axis asks where action generation should live. One family makes the VLM generate action tokens. A second family keeps final control in a continuous action expert. A third family uses action tokens to shape the VLM representation while insulating the continuous expert that executes the action. These are different hypotheses about the interface between multimodal context and control, not interchangeable loss names.

The current `praxis-vla` release instantiates these hypotheses as `BAR`, `FM`, and `KI`. `BAR` treats action generation as scheduled token prediction. `FM` treats control as continuous flow matching over action chunks. `KI` keeps token supervision but routes deployed control through a stop-gradient-insulated continuous expert. The objective choice fixes three properties of a cell: the supervision signal, the inference procedure, and the gradient path through the VLM.

Fig. 4 makes the distinction concrete by showing which lane receives supervision, which lane generates actions, and where gradients are allowed to flow.

`BAR`. `BAR` asks how much serial dependence a tokenized VLA policy should keep. It generalizes autoregression from individual tokens to scheduled blocks. Given an action token sequence $z_{1:L}$, a schedule $\mathbf{b} = (b_1, \ldots, b_S)$ with $0 = b_0 < b_1 < \cdots < b_S = L$ defines blocks $G_s = z_{b_{s-1}+1:b_s}$ and block sizes $g_s = b_s - b_{s-1}$. The `BAR` schedule family used here requires nondecreasing block sizes, $g_1 \leq \cdots \leq g_S$. At stage s, the policy conditions on the observation and realized prefix $z_{1:b_{s-1}}$, then predicts the whole block G_s in parallel. The schedule specifies the unit of autoregressive growth: one token, a fixed-size block, a variable-size nondecreasing block sequence, or the entire action sequence.

This schedule view unifies the endpoints. Token-wise AR is the special case $S = L$ and $|G_s| = 1$. Parallel decoding (PD) is the opposite endpoint $S = 1$. Fixed-size and variable-size `BAR` occupy the space between these two limits, connecting VLA action-token prediction to blockwise, masked, and partially parallel sequence-generation methods (Stern et al., 2018; Ghazvininejad et al., 2019; Lee et al., 2018).

The cell boundary turns serial dependence into a controllable variable. Training uses block-shifted teacher forcing and a schedule-aware attention mask, so the current block is supervised without being copied into the input. For codecs not tied to a schedule, a study can compare token-wise AR, blockwise AR, and PD-style endpoints while keeping the codec, backbone, and evaluator unchanged. For a block-aware ordered codec, by contrast, the schedule is part of the checkpoint and belongs to the action representation being compared.

FM. FM is the continuous-action reference point on the objective axis. Following flow-based VLA policies (Black et al., 2024), it trains an action expert to map noisy action chunks toward data actions, and no discrete action sequence is produced at inference time. For a normalized action chunk a, Gaussian noise ϵ, and time t, the training point is

$$x_t = (1-t)a + t\epsilon, \qquad u^\star = \epsilon - a.$$

The action expert predicts the velocity target

$$\mathcal{L}_{\text{FM}} = \left\| v_\phi(x_t, t, h_\theta(o)) - u^\star \right\|_2^2,$$

where $h_\theta(o)$ is the VLM context and v_ϕ is the continuous action expert. Under this convention, $t = 1$ corresponds to noise and $t = 0$ corresponds to data. At inference time, the policy starts from noise and integrates the learned velocity field in reverse time toward the data endpoint, producing a continuous action chunk without detokenization.

Because no token sequence has to be chosen, decoded, or declared valid before action execution, FM removes tokenization error, invalid token sequences, and discrete decoding latency from the inference path. The tradeoff is that the VLM no longer receives an action-token language as structured supervision. In the policy matrix, FM is the reference point for asking what changes when final control is continuous rather than symbolic.

KI. KI tests a hybrid hypothesis: action tokens may be useful as supervision even when they are not the deployed control representation. The motivation is the concern that gradients from low-level continuous control may interfere with pretrained multimodal representations that are valuable for perception and language conditioning (Driess et al., 2025; Intelligence et al., 2026). KI separates these roles by using token cross-entropy as an action-aware representation-learning signal, while a flow expert handles high-bandwidth continuous action prediction.

When both branches are active, the objective combines a token loss and an insulated flow loss,

$$\mathcal{L}_{\text{KI}} = \lambda_{\text{CE}}\mathcal{L}_{\text{CE}}(z_{1:L}) + \lambda_{\text{flow}} \left\| v_\phi(x_t, t, \text{sg}(c_\theta(o))) - (\epsilon - a) \right\|_2^2,$$

where $c_\theta(o)$ denotes the VLM context exposed to the expert and $\text{sg}(\cdot)$ is a stop-gradient boundary. In this KI interface, the context is the image, language, and state prefill context used by the expert, not a requirement to decode action-token logits at inference. The token branch asks whether the tokenizer gives the VLM a useful action language. The flow branch asks whether an insulated expert can turn the resulting context into continuous control. Branch weights and warm-start schedules are protocol choices inside the same family; the defining design is the two-branch decomposition with an explicit gradient boundary.

At inference time, KI uses the VLM primarily as a context provider and the expert as the action generator. The policy does not need to complete a BAR-style autoregressive token decode before acting. This makes the tokenizer a representation-learning variable rather than necessarily a deployment-time decoder.

3.3 Action-Representation Axis

The action-representation axis defines the policy's action language. For a normalized action chunk $a_{1:H}$, a codec defines

$$z_{1:L} = E_c(a_{1:H}), \qquad \hat{a}_{1:H} = D_c(z_{1:L}).$$

The codec is more than a compression layer. It defines the policy output space, the length and structure of the generated suffix, the behavior of arbitrary policy samples, and the way prediction errors become executable actions. A tokenizer that reconstructs demonstrations well can still be a

Tokenizer	Compact	Total decodability	Predictive order	BAR schedule
Bin	×	✓	×	✓
FAST	✓	×	✓	×
QueST	✓	✓	×	×
ACodec	✓	✓	–	✓
OAT	✓	✓	✓	×
bOAT	✓	✓	✓	✓

Table 3: Action tokenizer criteria exposed by the policy matrix. Praxis compares action representations along four policy-facing properties: whether the representation is substantially more compact than scalar Bin under its intended use, whether fixed-length raw policy outputs decode directly to valid actions without repair, whether the emitted sequence has an intended prediction order, and whether the representation supplies stable prediction groups for a BAR schedule. Predictive order does not by itself imply block-schedule support; bOAT is the block-aware OAT variant. Check marks denote that the tokenizer satisfies the criterion; crosses denote that it does not; dashes denote intentionally order-agnostic designs.

Name	Structure	Policy-facing role
Bin	scalar quantization; total and inspectable, but long with weak prefix semantics	transparent baseline for asking whether compression and ordering help
FAST	frequency-domain coefficients with subword compression	tests compact spectral coding and the cost of variable-length decode assumptions
QueST	learned latent tokens optimized for compact action reconstruction	tests whether compact learned codes can be predicted without explicit prefix training
ACodec	learned latent groups intended for parallel action-token prediction	tests the one-shot or low-serial-depth endpoint of tokenized control
OAT	ordered, totally decodable prefixes trained as coarse-to-fine budgets	tests whether token order and prefix semantics improve BAR and KI cells
bOAT	ordered prefixes with block-aware latent dependencies	tests whether the representation itself should be matched to the BAR schedule

Table 4: Action representations as policy variables. Each row changes more than reconstruction error: it changes the policy output space, decode semantics, prefix behavior, and compatibility with objective families.

poor policy language if valid action chunks are hard to predict, prefixes are meaningless, or decoding assumptions fail under model samples.

Table 3 makes the policy-facing criteria explicit. The useful lens is rate–distortion–predictability under the policy objective: sequence length, control error after decoding, and whether a VLM-conditioned policy can generate valid sequences under its inference procedure. praxis-vla deliberately spans several points in this space: scalar quantization, frequency-domain coding, learned latent actions, one-shot latent groups, ordered prefixes, and block-aware ordered prefixes. Each choice introduces a different hypothesis about the policy's action language. Here, Ordered Action Tokenization (OAT) denotes the ordered-prefix codec, and block-wise Ordered Action Tokenization (bOAT) denotes its block-aware counterpart.

Table 4 summarizes how these representations change the policy-facing role of the action suffix before the individual families are discussed.

Bin anchors the axis with transparent scalar discretization, a common baseline in action-token policies (Brohan et al., 2022, 2023; Kim et al., 2024). Every generated symbol maps to a scalar

action value, which makes the representation easy to inspect and totally decodable. The cost is rate and prefix semantics: a horizon-H action chunk with D dimensions can require HD tokens, and early prefixes may describe coordinates rather than coherent coarse motion.

`FAST` represents the spectral-compression view: action chunks may become a better policy language after a frequency transform (Ahmed et al., 1974; Gage, 1994; Sennrich et al., 2016; Pertsch et al., 2025). Low-frequency coefficients describe broad motion while higher-frequency coefficients refine detail, giving the serialized coefficient stream an intended broad-to-fine prediction order. Its policy-facing risk is structural decoding: byte-pair encoding (BPE) tokens expand to variable-length coefficient streams, so invalid lengths or shifted coefficient positions can make arbitrary samples difficult to decode cleanly.

`QueST` and `ACodec` represent learned latent-token approaches (Lee et al., 2024; Mete et al., 2024; Dong et al., 2026). They learn compact bottlenecks for action chunks and decode quantized latents back to continuous control. Their central question is whether a compact rate–distortion representation is also predictable under the policy objective. A learned latent space may reconstruct well while offering little prefix semantics; conversely, a one-shot latent group may be appropriate when joint predictability matters more than incremental decoding.

`OAT` and `bOAT` represent the ordered-token view. `OAT` uses register latents, finite scalar quantization, causal register attention, and nested-dropout reconstruction budgets to train prefixes as progressively refined action representations (Darcet et al., 2024; Mentzer et al., 2024; Rippel et al., 2014; Kusupati et al., 2022; Cai et al., 2025; Bachmann et al., 2025; Liu et al., 2026). This gives tokenized objectives a language whose early symbols are intended to carry coarse control information, but it does not by itself train `OAT` tokens as joint `BAR` blocks. `bOAT` keeps the ordered-prefix view while changing the latent dependency pattern so tokens in the same scheduled block are trained as parallel slots conditioned on earlier blocks. This makes ordering a representation-level hypothesis rather than only an inference-time schedule.

3.4 Backbone Axis

The backbone axis specifies how images, optional robot state, and language become policy context. Backbone choice changes the preprocessing and representation contract: image preprocessing, language formatting, attention conventions, hidden-state geometry, cache layout, and fine-tuning behavior. Without a backbone boundary, a comparison between VLMs can become a comparison between unrelated data and inference pipelines.

The current matrix uses `SmolVLM`, `Qwen3-VL`, and `PaliGemma2` to span a compact VLM, a large general-purpose VLM family, and a VLM designed around image-language pretraining (Marafioti et al., 2025; Bai et al., 2025; Steiner et al., 2024). Backbone adapters make these choices reportable by localizing the family-specific processor, prompt construction, image handling, prefix representation, and hidden-state extraction needed by the policy objective.

The action head conditions on representation geometry, not on a VLM name. Image resolution, processor choices, prefix attention, hidden-state layout, and cache behavior all affect rollout while leaving the objective, action language, and evaluator nominally unchanged. Keeping these details in the backbone axis lets a study report them explicitly.

3.5 Using and Extending the Matrix

In a matrix study, the changed coordinate is explicit: an action-representation study compares $\pi_{m,c_1,b}$ and $\pi_{m,c_2,b}$, a backbone study compares π_{m,c,b_1} and π_{m,c,b_2}, and an objective study compares cells such as $\pi_{\mathrm{BAR},c,b}$, $\pi_{\mathrm{KI},c,b}$, and $\pi_{\mathrm{FM},\varnothing,b}$. The notation does not prove causality by itself. Dataset selection, optimization budget, action horizon, normalization state, rollout budget, and metrics remain part of the empirical protocol. The matrix makes the claimed variable visible enough for that proto-

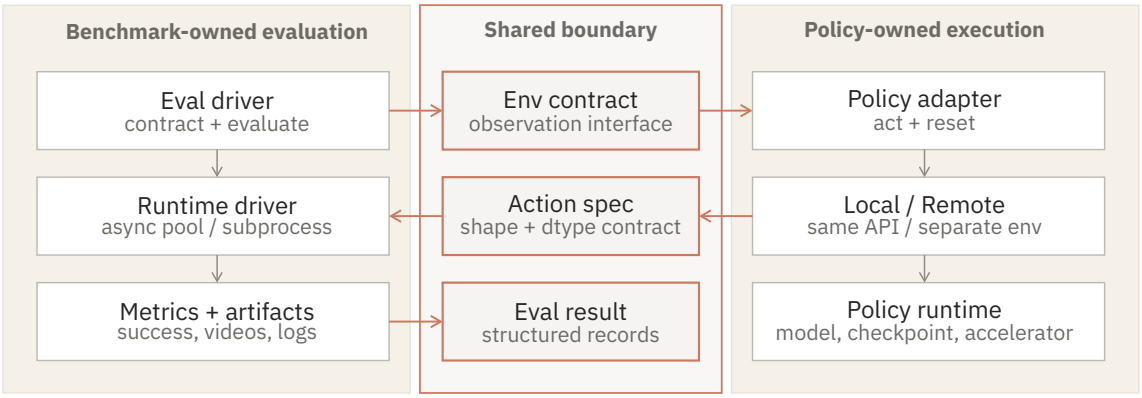

Figure 5: `praxis-eval` turns benchmark-specific runtime into a stable policy boundary. Benchmark drivers own task selection, runtime orchestration, metrics, and artifacts. Policies receive declared observations and return contract-valid actions through the same local or remote boundary. Validation and structured records keep runtime side effects and benchmark diagnostics visible rather than hidden inside policy code.

col to be inspected.

New objectives, action representations, backbones, and benchmarks remain comparable when Praxis records the hypothesis each component changes, the policy-facing semantics needed to test it, and the surrounding contracts held fixed.

4 `praxis-eval`: Evaluation as a Boundary

Sec. 3 treats policies as cells whose intended variables are explicit. `praxis-eval` (Liu, 2026) supplies the matching evaluation boundary. Evaluation fixes the observations a policy receives, the action convention it must satisfy, the tasks selected for rollout, the simulator runtime that executes those actions, and the metric code that summarizes the episode. If these assumptions live inside each policy family, benchmark integration becomes an unreported part of the policy comparison.

Evaluation boundary

Benchmark drivers own simulator semantics, rollout, metrics, and artifacts. Policies expose only reset and act through a local or remote adapter. This boundary lets one policy cell move across benchmarks without turning benchmark glue into a policy variable.

Fig. 5 shows the boundary used throughout the section: benchmark drivers own runtime and metrics, while policies see only declared observations and contract-valid actions.

Benchmark ownership. Benchmark semantics stay benchmark-owned. `LIBERO` (Liu et al., 2023), `RoboMimic` (Mandlekar et al., 2021), `MetaWorld` (Yu et al., 2020), `RoboCasa` (Nasiriany et al., 2024, 2026), `SimplerEnv` (Li et al., 2024), and `MS-HAB` (Shukla et al., 2025) rollouts differ in assets, rendering backends, action conventions, task selectors, termination rules, and success metrics. Rather than flattening those differences into one generic environment, `praxis-eval` treats each benchmark as a driver with its own contract and rollout implementation, while exposing the same kind of policy-facing boundary.

Contracts. The contract is intentionally smaller than the simulator state. An environment contract lists the observation keys available to the policy and the action shape, dtype, optional bounds, and benchmark convention it must satisfy. Boundary validation rejects malformed actions before the sim-

ulator can silently reinterpret them. Benchmark drivers still define valid interaction, while policies translate documented observations into valid actions.

Local and remote execution. Dependency isolation matters scientifically, not only operationally. Without a stable remote boundary, dependency conflicts are often resolved by changing the policy package, changing the simulator package, or forking the evaluator. Those changes can become hidden experimental variables. `praxis-eval` supports local and remote policy execution through the same narrow policy protocol: reset episode state, receive a batch of observations, and return a batched action array. Local execution is the minimal path when policy code and simulator code can share one Python environment; remote execution is the isolation path when they cannot.

Episode state. VLA policies often have temporal state. A chunked policy may cache an action chunk, consume it over several simulator steps, and then request a new chunk; a remote policy may need episode identifiers to reset the correct internal stream. `praxis-eval` treats this state as part of the policy-evaluation boundary. The evaluator resets policies at episode or wave boundaries, passes episode identifiers when needed, and can batch observations across vectorized rollouts without hiding temporal semantics inside benchmark-specific glue code. This keeps state-management errors separate from policy failures.

Extensibility. A new benchmark contributes benchmark-local driver logic, task and metric definitions, an observation/action contract, and artifact handling. If dependencies conflict, the driver can launch the simulator in a subprocess and communicate with the policy through the same remote boundary. Benchmark-specific complexity stays inside the driver, and the policy-facing contract stays narrow.

Evaluation records. Each evaluation record stores aggregate, task-level, and optional group metrics together with artifact paths and metadata. It also keeps benchmark-specific diagnostics such as videos, traces, logs, and external outputs, linking policy-cell comparisons to failure evidence.

5 Related Work

5.1 VLA Policy Architectures

Vision-language-action policies have rapidly expanded from large-scale autoregressive action-token models to generalist policy stacks and continuous action decoders. RT-1 and RT-2 established token-prediction interfaces for language-conditioned robot control at scale (Brohan et al., 2022, 2023). Octo and OpenVLA made open generalist policy stacks available for broader study (Octo Model Team et al., 2024; Kim et al., 2024), while recent flow-based VLA systems emphasize continuous action generation from multimodal context (Black et al., 2024; Intelligence et al., 2025). These papers define policy architectures. Praxis addresses a different layer: how to compare such architectures when objectives, action representations, backbones, and evaluator contracts can move independently.

5.2 Action Representations and Tokenizers

Action tokenization work treats the action interface itself as a modeling choice. Autoregressive robot policies often begin with scalar discretization (Brohan et al., 2022, 2023; Kim et al., 2024); FAST-style tokenizers compress action chunks through spectral and subword structure (Pertsch et al., 2025); learned latent-action methods introduce bottlenecks or skill abstractions (Lee et al., 2024; Mete et al., 2024; Dong et al., 2026); and OAT adds prefix semantics to the action language (Liu et al., 2026). `praxis-vla` puts these choices on one action-representation axis and compares them by decode validity, sequence structure, prefix behavior, and predictability under a fixed policy objec-

tive.

5.3 Continuous, Diffusion, and Hybrid Objectives

Continuous-control objectives change what has to be represented by the policy interface. Instead of requiring a VLA to emit a long sequence of action tokens, diffusion and flow-based policies generate action chunks through continuous generative dynamics, preserving multimodal control structure without assigning every control value to a discrete symbol (Chi et al., 2025; Black et al., 2024). Recent hybrid systems make the same issue visible from different angles: the objective can determine not only the loss, but also the decomposition, routing, symbolic structure, and token supervision exposed by the policy (Liu et al., 2026; Høeg et al., 2026; Chen et al., 2025; Driess et al., 2025; Intelligence et al., 2026). For Praxis, continuous objectives matter because they change the supervision path, runtime decoder, and evaluation-facing interface. Labeling policies only as "discrete" or "continuous" hides choices that affect training, inference, and integration.

5.4 Benchmarks and Evaluation Infrastructure

Robot-policy progress is often reported through benchmark suites and simulators. RoboMimic, Meta-World, LIBERO, RoboCasa, SimplerEnv, and ManiSkill-HAB already define manipulation tasks, datasets, simulator interfaces, and evaluation protocols (Mandlekar et al., 2021; Yu et al., 2020; Liu et al., 2023; Nasiriany et al., 2024; Li et al., 2024; Shukla et al., 2025). These resources are measurement surfaces for Praxis, not components to replace. `praxis-eval` keeps benchmark semantics benchmark-owned while exposing a narrow policy protocol, so benchmark adapters do not become hidden policy variables.

6 Discussion and Conclusion

Praxis frames VLA policy research as a controlled design-space problem. Its central claim is that evidence becomes easier to interpret when objectives, tokenizers, backbones, and evaluators are named as experimental variables before rollout scores are reported. A policy-cell notation such as $\pi_{m,c,b}$ is not bookkeeping; it names the hypothesis being tested and the surrounding assumptions that should remain fixed.

Controlled comparison has to be encoded in the artifact itself. `praxis-vla` records the policy hypothesis: where action generation lives, how actions are represented, and which VLM-family assumptions enter the policy. `praxis-eval` records the evaluation boundary by keeping simulator semantics, rollout orchestration, metrics, and artifacts benchmark-owned behind a narrow observation/action protocol.

Praxis still leaves empirical design to each study: datasets, optimization budgets, hyperparameter sweeps, random seeds, task splits, and success metrics must be chosen and reported by the researcher. It also cannot remove axis interactions such as objective-codec co-adaptation, backbone preprocessing effects, or evaluator settings that shift rollout distributions; it only makes those choices explicit instead of burying them in training and evaluation scripts.

Praxis is meant to make extensions comparable. New objectives, action codecs, VLM backbones, and benchmarks should add to their axis while preserving the surrounding contracts, so later studies can attribute behavior changes to the design choice under test rather than to accidental differences between software stacks.

References

[1] N. Ahmed, T. Natarajan, and K.R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, 1974. doi: 10.1109/T-C.1974.223784. 9

[2] Roman Bachmann, Jesse Allardice, David Mizrahi, Enrico Fini, Oğuzhan Fatih Kar, Elmira Amirloo, Alaaeldin El-Nouby, Amir Zamir, and Afshin Dehghan. Flextok: Resampling images into 1d token sequences of flexible length. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=DgdOkUUBzf. 9

[3] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL https://arxiv.org/abs/2511.21631. 9

[4] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164. 2, 7, 11, 12

[5] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale. In *arXiv preprint arXiv:2212.06817*, 2022. 2, 8, 11

[6] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alex Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *arXiv preprint arXiv:2307.15818*, 2023. 2, 8, 11

[7] Mu Cai, Jianwei Yang, Jianfeng Gao, and Yong Jae Lee. Matryoshka multimodal models. In *International Conference on Representation Learning*, 2025. 9

[8] Haonan Chen, Jiaming Xu, Hongyu Chen, Kaiwen Hong, Binghao Huang, Chaoqi Liu, Jiayuan Mao, Yunzhu Li, Yilun Du, and Katherine Driggs-Campbell. Multi-modal manipulation via multi-modal policy consensus. *arXiv preprint arXiv:2509.23468*, 2025. 12

[9] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025. doi: 10.1177/02783649241273668. URL https://journals.sagepub.com/doi/10.1177/02783649241273668. 12

[10] Timothee Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *International Conference on Representation Learning*, 2024. 9

[11] Zibin Dong, Yicheng Liu, Shiduo Zhang, Baijun Ye, Yifu Yuan, Fei Ni, Jingjing Gong, Xipeng Qiu, Hang Zhao, Yinchuan Li, and Jianye Hao. Actioncodec: What makes for good action tokenizers, 2026. URL https://arxiv.org/abs/2602.15397. 2, 9, 11

[12] Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, et al. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *arXiv preprint arXiv:2505.23705*, 2025. 7, 12

[13] Philip Gage. A new algorithm for data compression. *C Users J.*, 12(2):23–38, February 1994. ISSN 0898-9788. 9

[14] Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6112–6121, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1633. URL https://aclanthology.org/D19-1633/. 6

[15] Sigmund H. Høeg, Aksel Vaaler, Chaoqi Liu, Olav Egeland, and Yilun Du. Hybrid diffusion for simultaneous symbolic and continuous planning. *IEEE Robotics and Automation Letters*, 11(4):4489–4496, 2026. doi: 10.1109/LRA.2026.3664616. 12

[16] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054. 2, 11

[17] Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, Vedant Choudhary, Foster Collins, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Maitrayee Dhaka, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine

Glossop, Thomas Godden, Ivan Goryachev, Lachlan Groom, Haroun Habeeb, Hunter Hancock, Karol Hausman, Gashon Hussein, Victor Hwang, Brian Ichter, Connor Jacobsen, Szymon Jakubczak, Rowan Jen, Tim Jones, Gregg Kammerer, Ben Katz, Liyiming Ke, Mairbek Khadikov, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Brendon LeCount, Sergey Levine, Xinyu Li, Adrian Li-Bell, Vladislav Lialin, Zhonglin Liang, Wallace Lim, Yao Lu, Enyu Luo, Vishnu Mano, Nandan Marwaha, Aikys Mongush, Liam Murphy, Suraj Nair, Tyler Patterson, Karl Pertsch, Allen Z. Ren, Gavin Schelske, Charvi Sharma, Baifeng Shi, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Jiaming Tang, Jimmy Tanner, Shalom Tekeste, Marcel Torne, Kyle Vedder, Quan Vuong, Anna Walling, Haohuan Wang, Jason Wang, XuDong Wang, Chris Whalen, Samuel Whitmore, Blake Williams, Charles Xu, Sukwon Yoo, Lili Yu, Wuming Zhang, Zhuoyang Zhang, and Ury Zhilinsky. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities, 2026. URL https://arxiv.org/abs/2604.15483. 7, 12

[18] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. 2, 8, 11

[19] Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. Matryoshka representation learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 30233–30249. Curran Associates, Inc., 2022. 9

[20] Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement, 2018. URL https://arxiv.org/abs/1802.06901. 6

[21] Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 26991–27008. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/lee24y.html. 9, 11

[22] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024. 2, 10, 12

[23] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning, 2023. URL https://arxiv.org/abs/2306.03310. 2, 10, 12

[24] Chaoqi Liu. praxis-eval, 2026. URL https://github.com/Chaoqi-LIU/praxis-eval. 10

[25] Chaoqi Liu, Haonan Chen, Sigmund H. Høeg, Shaoxiong Yao, Yunzhu Li, Kris Hauser, and Yilun Du. Flexible multitask learning with factorized diffusion policy. *IEEE Robotics and Automation Letters*, 11(4):4697–4704, 2026. doi: 10.1109/LRA.2026.3664611. 12

[26] Chaoqi Liu, Xiaoshen Han, Jiawei Gao, Yue Zhao, Haonan Chen, and Yilun Du. Oat: Ordered action tokenization, 2026. URL https://arxiv.org/abs/2602.04215. 2, 9, 11

[27] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021. 2, 10, 12

[28] Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Vaibhav Srivastav, Joshua Lochner, Hugo Larcher, Mathieu Morlon, Lewis Tunstall, Leandro von Werra, and Thomas Wolf. Smolvlm: Redefining small and efficient multimodal models, 2025. URL https://arxiv.org/abs/2504.05299. 9

[29] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple. In *International Conference on Representation Learning*, volume 2024, pages 51772–51783, 2024. 9

[30] Atharva Mete, Haotian Xue, Albert Wilcox, Yongxin Chen, and Animesh Garg. Quest: Self-supervised skill abstractions for learning continuous control. In *Advances in Neural Information Processing Systems*, volume 37, pages 4062–4089. Curran Associates, Inc., 2024. doi: 10.52202/079017-0133. 9, 11

[31] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems*, 2024. 2, 10, 12

[32] Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations (ICLR)*, 2026. 10

[33] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024. 2, 11

[34] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. In *Robotics: Science and Systems*, 2025. 2, 9, 11

[35] Oren Rippel, Michael Gelbart, and Ryan Adams. Learning ordered representations with nested dropout. In *Proceedings of the 31st International Conference on Machine Learning*, Proceedings of Machine Learning Research, Bejing, China, 22–24 Jun 2014. PMLR. URL https://proceedings.mlr.press/v32/rippel14.html. 9

[36] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725, 2016. 9

[37] Arth Shukla, Stone Tao, and Hao Su. Maniskill-hab: A benchmark for low-level manipulation in home rearrangement tasks, 2025. URL https://arxiv.org/abs/2412.13211. 2, 10, 12

[38] Andreas Steiner, André Susano Pinto, Michael Tschannen, Daniel Keysers, Xiao Wang, Yonatan Bitton, Alexey Gritsenko, Matthias Minderer, Anthony Sherbondy, Shangbang Long, Siyang

Qin, Reeve Ingle, Emanuele Bugliarello, Sahar Kazemzadeh, Thomas Mesnard, Ibrahim Alabdulmohsin, Lucas Beyer, and Xiaohua Zhai. Paligemma 2: A family of versatile vlms for transfer, 2024. URL https://arxiv.org/abs/2412.03555. 9

[39] Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models, 2018. URL https://arxiv.org/abs/1811.03115. 6

[40] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 1094–1100. PMLR, 30 Oct–01 Nov 2020. URL https://proceedings.mlr.press/v100/yu20a.html. 2, 10, 12