

World Models and World Action Models for Embodied AI

From DreamZero and Dream-1 to VLA/WAM Compute: Training, Inference, and Embodiment Transfer

2026-10-22

Contents

1. Why predict the world before or while acting?	2
2. Four architectural paradigms for WAMs	2
3. Data: aligning video, action, and embodiment	3
4. Generic WAM training and inference interfaces	3
5. DreamZero: a fully specified research case study	4
6. Dream-1: cross-embodiment zero-shot control as a system	6
7. DreamZero versus Dream-1: a precise comparison	9
8. VLA versus WAM: a systems comparison of capability, compute, and deployment	9
9. Efficient inference: joining the model clock and robot clock	12
10. Contact-rich manipulation and recovery	13
11. Evaluating the contribution of a WAM	13
12. Boundaries within the course and reading order	14
13. References and official resources	14

[Course home](#) · [简体中文](#) · [DreamZero paper](#) · [Dream-1 announcement](#) · [PDF](#)

Item	Details
Topic	DreamZero, Dream-1 and cross-embodiment transfer
Revision	2026-10-22
License	CC BY-NC-SA 4.0 · Xiong Zeng and contributors

1. Why predict the world before or while acting?

Scope. This chapter follows the course material on imitation learning, flow matching, and GR00T/SONIC. It explains how predicted futures become executable robot actions. The original treatment of training data, inference options, efficiency, and deployment constraints is retained and augmented with two substantive case studies: **DreamZero, an open research method**, and **Dream-1, a cross-embodiment deployment system**, followed by a **systems-level VLA/WAM comparison of memory, compute, latency, and deployment costs**. We do not re-derive the flow-matching continuity equation, PPO, or general Transformer fundamentals; there are no programming exercises.

Key question added in this revision. Why can Dream-1 sometimes operate on unseen robots and tasks without fine-tuning? The explanation combines a **shared EEF action interface, joint video-action world modeling, and transferable visual representations**; EEF control and inverse kinematics are not exclusive to WAMs. Sections 6, 8, and 11 discuss mechanisms, limits, and controlled comparisons.

The result of a robotic command depends on object pose, contact mode, geometry, friction, occlusion, and disturbances. A conventional vision-language-action policy directly predicts a future action chunk:

$$\pi_{\theta}(A_t \mid h_t, \ell), \quad h_t = (o_{t-k:t}, q_{t-k:t}), \quad A_t = (a_t, \dots, a_{t+H-1}).$$

Here o_t denotes sensory observations, q_t proprioceptive state, and ℓ a language goal. **A VLA can certainly contain implicit dynamics knowledge.** The added value of an explicit prediction objective is temporal supervision, counterfactual reasoning, and a possible interface for feasibility checking.

World models (WMs) commonly predict action-conditioned observations or compact latent states:

$$p_{\phi}(o_{t+1:t+H} \mid h_t, A_t), \quad p_{\phi}(z_{t+1} \mid z_t, a_t).$$

World Action Models (WAMs) couple predicted futures to **executable action**. This does not require decoding videos at test time, nor does it imply explicit model-predictive control (MPC). A visually convincing predicted trajectory need not be physically feasible, particularly when contact forces are not observable.

Model class	Typical output	Principal supervision	Deployment role
VLA / action policy	Action chunks	Recorded actions, feedback	Select control commands
World model	Future observations or states	Temporal transitions	Predict, score, plan
World Action Model	Executable future-linked actions	Coupled prediction and action learning	Learn and/or execute policies

2. Four architectural paradigms for WAMs

2.1 Imagine then execute. Generate an anticipated future and condition an action generator on it:

$$\hat{O}^+ \sim p_{\phi}(O^+ \mid h_t, \ell), \quad A_t \sim \pi_{\theta}(A \mid h_t, \ell, \hat{O}^+).$$

This supports inspection of imagined outcomes, at the cost of video sampling and the risk of hallucinating unreachable states. Some systems additionally condition future prediction on candidate actions; the above is an illustrative factorization, not a universal definition.

2.2 Video-feature-conditioned actions. Consume the internal representation of a predictive backbone without fully decoding future video:

$$A_t \sim \pi_\theta(A \mid f_\phi(h_t, \ell), q_t).$$

Whether a predictive feature is truly controllable must be established through interaction, not visual reconstruction quality alone.

2.3 Joint video-action modeling. Learn a conditional distribution over future world observations and motor actions:

$$p_{\theta, \phi}(O^+, A \mid h_t, \ell).$$

The model may use one shared denoising backbone, separate modality decoders, and/or chunk-wise causal generation. **DreamZero** provides a fully specified example. Joint modeling does not entail identical autoregressive tokenization of all modalities.

2.4 Auxiliary predictive supervision. The world-prediction objective is used in training, while deployment retains only the action computation:

$$\min_{\theta, \phi} \mathcal{L}_{\text{action}} + \lambda \mathcal{L}_{\text{future}}, \quad A_t \leftarrow \pi_\theta(h_t, \ell).$$

Fast-WAM demonstrates that, in some settings, prediction improves the policy representation even when explicit test-time imagination is omitted. This is an empirical trade-off, not a theorem covering every WAM.

3. Data: aligning video, action, and embodiment

A minimal recorded interaction may be represented as

$$\mathcal{D}_R = \{(o_t, q_t, a_t, o_{t+1}, r_t, d_t, \ell, \Delta t)\}_{t=0}^{T-1}.$$

Record timestamps, action units, frames, controller mode, proprioception, and the distinction between requested commands and what the robot actually executed. Rewards r_t are optional for supervised pretraining but useful for planning or RL. Actuator saturation, filters, and network latency make these distinctions essential.

Data source	Transferable information	Fundamental limitation
Human / internet videos	Visual motion, interaction priors	No ground-truth robot commands
Robot teleoperation	Time-aligned observations and actions	Limited task/embodiment coverage
Simulation rollouts	States, contact, commands, reward	Rendering and contact sim-to-real gap
Online robot interactions	Failures, recoveries, distribution shifts	Safety, expense, data governance

A video-only sample must receive a video or latent-prediction loss, **not fabricated action ground truth**. Loss masks should reflect the modalities actually observed. Retargeted or inverse-dynamics-inferred pseudo-actions must be labeled as such. Joint space, end-effector $SE(3)$ targets, finger positions, and latent actions are different control semantics and require separate metadata.

4. Generic WAM training and inference interfaces

4.1 Learning predictive representations and motor commands

The world objective may be defined using video diffusion, flow matching, JEPA-style latent prediction, regression, or discrete tokens. These objectives are not interchangeable. A schematic conditional action flow-matching objective is

$$A_\tau = (1 - \tau)\varepsilon + \tau A^*, \quad \mathcal{L}_{\text{act}} = \mathbb{E} \left[\|v_\theta(A_\tau, \tau, c_t) - (A^* - \varepsilon)\|_2^2 \right].$$

Here the generative time τ is **not** physical robot time t . A conceptual family of joint training objectives can be written as

$$\mathcal{L} = \lambda_a \mathcal{L}_{\text{action}} + \lambda_v \mathcal{L}_{\text{video}} + \lambda_z \mathcal{L}_{\text{latent}} + \lambda_c \mathcal{L}_{\text{consistency}}.$$

This is a pedagogical bookkeeping expression, **not the verbatim objective of a named paper**. In particular, DreamZero jointly regresses a video-action velocity field, rather than necessarily training two unrelated modules and summing their losses.

4.2 Three deployment-time inference choices

Explicit imagination: sample anticipated video/latent futures and generate corresponding actions. The visual plan is inspectable, but the stages may incur sequential latency.

Joint generation: sample future visual content and executable motor chunks through one coupled generative process. DreamZero uses this approach, while periodically grounding the process in real observations.

Action-only inference: use prediction during training but avoid future-video decoding at deployment. Latency can improve, provided the predictive benefit survives this architectural choice.

5. DreamZero: a fully specified research case study

Sources: Ye et al., *World Action Models are Zero-shot Policies* (2026); [project](#); [released implementation](#). This section describes disclosed DreamZero methods, **not inferred implementation details of Dream-1**.

5.1 Central idea: a video prior aligned with inverse dynamics

DreamZero starts from Wan2.1-I2V-14B-480P, a pretrained image-to-video diffusion backbone. Conditioned on instructions c , visual history $o_{0:l}$, and proprioception q_l , it jointly predicts future video O^+ and actions A . By the chain rule,

$$\underbrace{p_\theta(O^+, A \mid h_l, c, q_l)}_{\text{joint WAM}} = \underbrace{p_\theta(O^+ \mid h_l, c, q_l)}_{\text{future video}} \underbrace{p_\theta(A \mid O^+, h_l, c, q_l)}_{\text{inverse-dynamics-related term}}.$$

The factorization is conceptual, not a recipe involving two independently trained models. DreamZero uses a shared causal Diffusion Transformer (DiT) and joint flow matching to align future visual dynamics with motor commands. Pre-training on temporal visual data supplies an inductive bias that is different from initializing an action head with a static vision-language model. This does not imply that the generated future is guaranteed to obey physics.

5.2 Architecture and modality organization

Observed frames are compressed through a VAE into video latents z ; text and proprioception are encoded separately. Action encoders and decoders interface motor commands with the causal DiT. Multi-camera images are concatenated into composite video frames rather than modifying the fundamental video backbone for every camera.

Three time indices must not be confused: **physical time** (camera and control cycles), **flow time** (τ from noise to data), and **chunk index** (k in autoregressive prediction). In the reported setup, each chunk contains $K = 2$ video latent frames; a training sample can contain up to $M = 4$ chunks. The AgiBot setup uses approximately 5 FPS video, 30 Hz motor actions, and $H = 48$ actions per output chunk. K , M , and H describe different quantities.

5.3 Joint video-action flow matching and teacher forcing

For chunk k , let z_1^k, a_1^k denote clean training targets and z_0^k, a_0^k independently sampled Gaussian noise. The shared normal-time schedule of ordinary DreamZero uses $\tau_k \sim \mathcal{U}[0, 1]$:

$$z_{\tau_k}^k = (1 - \tau_k)z_0^k + \tau_k z_1^k, \quad a_{\tau_k}^k = (1 - \tau_k)a_0^k + \tau_k a_1^k.$$

With $x_{\tau_k}^k = [z_{\tau_k}^k, a_{\tau_k}^k]$ and joint target velocity $v^k = [z_1^k, a_1^k] - [z_0^k, a_0^k]$, the training objective can be presented as

$$\mathcal{L}_{\text{DZ}} = \mathbb{E} \left[\frac{1}{M} \sum_{k=1}^M w(\tau_k) \|u_{\theta}(x_{\tau_k}^k; \mathcal{C}_k, c, q_k, \tau_k) - v^k\|_2^2 \right], \quad \mathcal{C}_k = \{(z_1^j, a_1^j) : j < k\}.$$

Here M denotes the number of trajectory chunks. We use distinct M and K for clarity because the paper uses a simplified/reused index notation in its loss expression. **Teacher forcing** supplies clean earlier chunks during training. Causal attention masks keep the current noisy chunk from attending to inaccessible future information, while the shared backbone can model video-action interactions within a chunk.

5.4 Training data and what “30 minutes” really means

The paper reports roughly **500 hours, 7,200 episodes, and 22 environments** of diverse AgiBot G1 demonstrations, with a separate DROID/Franka study. The AgiBot and DROID runs each report 100K optimizer steps and global batch size 128. Training updates the DiT and state/action encoders and decoders while freezing the text encoder, image encoder, and VAE. The authors report that LoRA did not outperform their chosen update scheme.

The famous **30-minute** figure refers to *adapting an already-pretrained model to a new YAM robot*, not learning a 14B WAM from scratch. An ablation using approximately 500 hours of data per condition reports 50% progress on diverse trajectories compared with 33% on repetitive trajectories for one PnP-Easy setup. The implication is that diverse state/action correspondences can matter more than repeating one demonstration many times; this is not a task-independent scaling law.

5.5 Closed-loop inference and correct interpretation of KV cache

At inference, the model jointly samples a predicted video chunk and an action chunk. After a portion of the actions is executed, the next inference uses **real camera observations and real proprioception**, refreshing the conditional context and its causal attention cache:

$$(\hat{O}_{t:t+H}, \hat{A}_{t:t+H}) \sim p_{\theta}(\cdot \mid h_t, c, q_t), \quad h_{t+1} \leftarrow \text{Update}(h_t, o_{t+1}^{\text{real}}, q_{t+1}^{\text{real}}).$$

This limits the rollout drift of a video generator that would otherwise condition indefinitely on its own hallucinated frames. **It does not eliminate sensing errors, partial observability, contact uncertainty, or all compounding control errors.** Historical-context **KV caching** is distinct from reuse of intermediate DiT calculations across denoising steps.

5.6 From 5.7 seconds to 150 milliseconds: DreamZero-Flash

Naive 14B inference with 16 denoising steps took approximately 5.7 seconds per chunk in the paper’s baseline. The reported acceleration stacks asynchronous model inference and motor execution, multi-GPU classifier-free-guidance parallelism, DiT computation reuse, `torch.compile`/CUDA Graphs, hardware-dependent quantization, and optimized kernels and scheduling.

DreamZero-Flash additionally **decouples video and action noise schedules during training**. It deliberately exposes the policy to heavily noisy video latents while learning to produce clean actions. In an illustrative configuration, $\eta \sim \text{Beta}(7, 1)$ and $\tau_{\text{video}} = 1 - \eta$, so the average video interpolation time is around 0.125, while action time remains uniform. This addresses a specific few-step inference mismatch: actions must be useful even before video latents have fully denoised.

Paper condition	Denoising steps	Table-bussing task progress	Latency per chunk
DreamZero	4	83%	350 ms
DreamZero, naive reduction	1	52%	150 ms
DreamZero-Flash	1	74%	150 ms

The paper reports an aggregate **38-fold speedup**, **150 ms/chunk**, and **approximately 7 Hz model-level closed-loop control on two GB200 GPUs**. These are not guaranteed results on single consumer GPUs. The robot still executes lower-level commands at approximately 30 Hz. A chunk of $H = 48$ commands therefore **covers 1.6 seconds of motion**; that is the action horizon, **not** the observed model or network latency. Asynchronous execution still needs safe chunk handover, time alignment, and fallback control.

5.7 What the evaluation shows—and what it does not

For AgiBot in unseen environments, reported mean task progress was 62.2% on the seen-task category versus 27.4% for the strongest pretrained VLA baseline evaluated there; on unseen tasks it was approximately 39.5% versus 16.3%. **These are task-progress metrics, not unconditional task-completion rates**. Video-only transfer from 12 minutes of human data or 20 minutes of YAM data increased progress on one unseen-task evaluation from 38.3% to 54.3% or 55.4%, respectively. Separately, 55 trajectories comprising about 30 minutes of YAM play data were used to **fine-tune for the new embodiment**, followed by tests of transfer to novel objects and tasks.

Important limitations remain: the reported temporal context is around six seconds, not verified long-horizon memory; video plausibility need not entail contact correctness or reachable poses; and 7 Hz model-level control is not a substitute for high-frequency feedback at first contact. Cross-embodiment transfer from video-only data, adaptation with 30 minutes of robot data, and Dream-1’s **no-robot-data zero-shot transfer** are different experimental claims.

6. Dream-1: cross-embodiment zero-shot control as a system

Source: [Dream Labs, Zero-Shot to New Robots: Introducing Dream-1 \(October 2026\)](#). **This is an official technical/product announcement, not a full peer-reviewed release of the training recipe**. The following distinguishes disclosed interfaces and evidence from open questions.

6.1 From few-shot adaptation to zero-shot embodiment generalization

DreamZero’s YAM experiment fine-tuned on **30 minutes of data from the target robot** and subsequently tested generalization. Dream-1 instead demonstrates deployment on **bimanual parallel-gripper robots that were unseen during training**, without collecting target-robot demonstrations or updating the policy for that particular robot first.

It is useful to maintain three distinct definitions:

- **Unseen task/environment**: the goal or scene was absent from training; the robot body may be familiar.
- **Few-shot embodiment adaptation**: a new robot provides some training data and the policy is updated before evaluation.
- **Zero-shot embodiment generalization**: a robot that provides no prior embodiment-specific training data executes commands via a compatible shared interface.

The claimed advance is that changing arm kinematics or gripper appearance need not always require an independently trained policy head. The currently documented transfer domain remains narrower than “arbitrary robot with arbitrary actuators.”

6.2 A shared end-effector action space plus robot-specific IK

Dream-1 can produce native joint actions for embodiments it supports, but its cross-robot mode also predicts **end-effector (EEF) poses for two arms plus parallel-gripper open/close signals**. The receiving robot’s inverse kinematics (IK) solver converts these into joint commands. A schematic abstraction is

$$a_t^{\text{shared}} = \{(T_{EE,i}^{\text{target}}, g_i)\}_{i \in \{L,R\}}, \quad q_{t+1}^{(r)} = \text{IK}_r(a_t^{\text{shared}}, q_t^{(r)}, \mathcal{C}_r).$$

Here $T_{EE,i} \in SE(3)$ denotes an EEF target, g_i a gripper signal, and IK_r the particular robot’s solver and constraints. **This equation conveys the public interface at a conceptual level; it is not a disclosed exact action-tensor specification.** Sharing Cartesian EEF semantics removes some dependence on each arm’s joint count and link layout. **However, EEF + IK is not specific to WAMs: a VLA with a shared Cartesian action interface can use the same mechanism.** It answers *how to issue commands to a new body*, not by itself *which physical interaction the robot should perform*.

However, IK is primarily a geometric mapping. It does not alone guarantee dynamic feasibility, collision-free paths, safe contact, or comparable reachable workspaces. Dream Labs explicitly notes that performance depends on IK quality; a commanded target pose does not fully specify the path taken by the arm and can become problematic near obstacles or joint limits.

6.3 Why visual world modeling can help transfer—and what is unknown

The official interface uses **three RGB views: one head camera and two wrist cameras**. The head view offers scene context, while each wrist view provides a local end-effector perspective. Compared with UMI’s cross-platform transfer under matched gripper hardware and wrist-camera views, the WAM formulation can use head-camera pixels to model the changing appearance and motion of different robot embodiments.

A robot’s joint coordinates and link geometry are embodiment-specific, whereas the visual consequences of reaching, contact, and object motion can exhibit transferable structure. Predicting those consequences supplies temporal supervision; it does not by itself identify contact forces or guarantee physical feasibility.

One useful conceptual chain is **shared visual dynamics representation $\rightarrow$ intended physical evolution $\rightarrow$ actions expressed in a shared EEF space $\rightarrow$ robot-specific IK**. It is a plausible account of the public system design, not a fully identified causal mechanism. The announcement does not demonstrate through released ablations which private internal loss, attention design, or training mixture is necessary for embodiment generalization.

6.4 Three complementary mechanisms behind zero-shot transfer

Dream-1’s public design suggests **three complementary mechanisms**. Some primarily make commands portable to a new robot; others encourage generalization across appearance and tasks. Their individual causal contributions have **not** been established by a controlled, released ablation.

Mechanism	Generalization obstacle it addresses	What it cannot guarantee by itself
Shared action interface (EEF + IK)	Different joint counts, link geometries, and joint-level action encodings	Collision-free trajectories or dynamics
Joint video–action world modeling	Limited temporal supervision when a policy only imitates training actions	Accurate forces or success on arbitrary unseen tasks
Visual cross-embodiment representation (three cameras)	Unfamiliar robot appearance and gripper geometry	Compatibility with arbitrary camera layouts and actuator interfaces

An *explanatory abstraction* separates transferable prediction from embodiment-specific execution:

$$(o_{t-k:t}^{(r)}, q_t^{(r)}, \ell) \xrightarrow{\text{shared WAM}} (\hat{O}_t^+, A_t^{\text{EEF}}) \xrightarrow{\text{IK}_r} a_t^{(r)}.$$

This is **not** a disclosed two-stage Dream-1 computation graph: video and action can be jointly generated, and the diagram does not imply that Dream-1 must first render all frames. **Cross-embodiment transfer** benefits from portable command semantics and temporal visual priors; **cross-task transfer** additionally requires diverse interaction data, language conditioning, and reusable structure in contact and object motion. Experience with reaching, grasping, transporting, and pushing may help with new instructions, but reliability remains empirical.

Compared with a direct-action VLA, joint future prediction provides an extra temporal learning signal; it does **not** imply that VLAs contain no dynamics knowledge or cannot achieve zero-shot transfer. [LAP-3B](#), which expresses robot actions in language, reports **over 50% mean zero-shot success** on its evaluated previously unseen robots and tasks. To isolate the contribution of video prediction, VLA and WAM variants should be compared with matched cameras, shared action spaces, demonstrations, and control protocols.

6.5 Zero-shot coverage is not reliable task completion

Two questions must remain separate: **how much data is needed before using a new body or task**, and **how reliably does the deployed policy complete it**? Predictive video supervision can help with observable reaching and object motion, but insertion, tool use, fine alignment, and slip recovery may still fail because of uncertain contact physics, occlusion, or feasibility constraints. Dream-1 itself reports substantial variation across task categories. Hence “can attempt unseen tasks without adaptation” should never become “reliably solves all unseen tasks without fine-tuning.” Additional demonstrations, FaaS, local feedback control, or RL post-training remain relevant when zero-shot performance is inadequate.

6.6 API inference and the clock: action coverage is not latency

Dream Labs offers a cloud API that accepts observations and returns actions from a selected Dream-1 checkpoint. Published pricing is approximately **\$3 per hour for Dream-1** and **\$2 per hour for Dream-1-flash**. The announcement states that each call returns approximately **1.6 seconds of actions**; continuously requesting coverage corresponds to roughly 2,250 calls per hour. This is a pricing/action-coverage calculation, not a measurement of GPU execution time or network round-trip latency.

A robot-integrated loop must consider

$$T_{\text{e2e}} = T_{\text{sense}} + T_{\text{encode}} + T_{\text{upload}} + T_{\text{model}} + T_{\text{download}} + T_{\text{dispatch}}$$

The launch article does not disclose Dream-1’s exact parameter count, denoising steps, p95 cloud latency, internal caching design, or motor-control frequency. DreamZero’s two-GB200 150 ms or 7 Hz numbers therefore cannot be copied onto Dream-1. Cloud operation introduces additional requirements for network loss, stale chunks, timestamp alignment, and safe shutdown.

6.7 Evidence for unseen-robot generalization

Dream Labs presents rollouts from 27 **unseen simulated robots** and multiple previously unseen real bimanual parallel-gripper robots, alongside examples from 12 **partner/beta sites**. The announcement describes a simulation protocol of **300 rollouts per embodiment**: 150 rollouts over 50 language-following tasks and another 150 rollouts over 150 unseen tasks. This protocol is more informative than a video montage, but one must still distinguish selected demonstration clips, individual rollouts, complete success-rate distributions, and confidence intervals.

The official unseen-simulation scores are **normalized to the trained YAM reference (100%)**, rather than absolute task success: for example, two-arm PiPER **91.2%**, two-arm Franka FR3 **58.3%**, and YAM-7 **38.0%**. Their spread demonstrates both real transfer capability and persistent robot-dependent performance gaps.

Dream Labs also cites an external RoboCurve RoboDojo-RC Tier 1 evaluation. That benchmark has a different task and cost definition from DreamZero’s task-progress studies. **The two publications’ headline numbers are not directly comparable without a shared evaluation protocol**. We should neither turn cherry-picked clips into an aggregate success percentage nor treat a marketing comparison as a universal ranking of robot-control algorithms.

6.8 Fine-tuning-as-a-Service and its data-efficiency limits

When zero-shot performance is insufficient, the company offers **Fine-tuning-as-a-Service (FaaS)**. The post shows two adaptation examples using around **five minutes of target-robot data each**. A modified YAM end-effector setup required approximately **98 minutes**. In one simulation comparison across twelve beta-partner robots, fine-tuning improved language-following scores on ten; for example Trossen went from **71% to 88%** and Maker Arm from **56% to 75%** in the reported normalized language-following metric. These are **not absolute success rates**.

These are **data durations used in specific case studies**, not a universal guarantee that five minutes suffices for an arbitrary robot or task. The fine-tuning loss, parameter-update scope, compute budget, uncertainty, and comprehensive safety validation are not fully provided. The phrase “one-shot embodiment learning” in the announcement must not be read as a rigorous claim of adaptation from exactly one demonstration trajectory.

6.9 Open architecture questions and present scope

The Dream-1 announcement does **not** reveal a reproducible specification of its complete backbone, parameter count, prediction loss, action decoder, denoising scheduler, relative amounts of video versus robot data, detailed embodiment description protocol, or which parameters FaaS updates. It should not be asserted to be a simple repackaging of DreamZero’s Wan2.1-I2V-14B or DreamZero-Flash.

The product is currently positioned principally for **bimanual parallel-gripper manipulation**.

7. DreamZero versus Dream-1: a precise comparison

Aspect	DreamZero (research paper)	Dream-1 (official announcement)
Central question	Can a pretrained video prior become a reactive robot policy?	Can one policy control unseen bimanual robots without their own training data?
Disclosed backbone	Wan2.1-I2V 14B, causal DiT, joint FM	WAM framing and EEf/IK interface; full backbone undisclosed
Action representation	Primarily relative joint positions in the paper	Shared EEf poses and gripper states; native joint mode also possible
Training evidence	Data, objective, optimizer schedule, ablations documented	Much of the full training recipe undisclosed
Embodiment claim	YAM adaptation after 30 minutes of its own data	Zero-shot on compatible unseen robot embodiments
Inference numbers	150 ms/chunk on two GB200s under reported setup	API price and 1.6 s action coverage; comparable latency undisclosed
Main limitations	Compute cost, imperfect world prediction and contacts	IK/workspace/path constraints and current gripper scope
Reproducibility	Public code and checkpoints	API available; full model training not released

Interpretation. DreamZero is evidence that joint video-action generative training can produce a viable closed-loop robotic policy under specific hardware/data conditions. Dream-1 adds promising evidence that a shared EEf control interface and temporal visual priors can enable transfer to unseen bimanual robots. Neither establishes a universal theorem that WAMs dominate VLAs or that plausible imagined videos guarantee safe, physically correct manipulation.

8. VLA versus WAM: a systems comparison of capability, compute, and deployment

This section places DreamZero and Dream-1 in the larger robot-foundation-model design space. “**VLA = one cheap forward pass, WAM = iterative video synthesis**” is **not a definition**. OpenVLA predicts discrete action tokens, while π_0 , $\pi_{0.5}$ and SmolVLA can use multistep flow matching to produce actions. Conversely, Fast-WAM learns from future video during

training but need not synthesize future video at deployment. Resource requirements depend on **model and activation size, image/video token counts, solver evaluations, and whether future prediction remains on the execution path.**

8.1 Compare learning objectives before comparing model labels

Dimension	Typical direct-action VLA	Typical WAM (joint or efficient)
Training signal	Demonstrated actions; BC, action flow matching, or action tokens	Additional future-video/latent supervision, possibly coupled to action prediction
Pretraining prior	Vision-language semantics plus robot behavior	Video motion/dynamics prior plus robot-action alignment
Actuator-facing output	Discrete commands or continuous action chunks	Ultimately also executable actions; video may be intermediate or auxiliary
Deployment computation	Visual encoding plus action decoding/sampling	Full imagination, joint sampling, or action-only inference
Hypothesized generalization	Semantic targets and behavior transfer within learned action distributions	Unfamiliar motion, temporal outcomes, and changes caused by actions
Failure modes	Distribution shift in motions, action-data sparsity, perceptual errors	Additionally, hallucinated futures, runtime overhead, and inaccurate contact predictions

These are **design trends, not universal rankings**. Zero-shot transfer to a new embodiment additionally depends on the action interface, training embodiment coverage, and downstream controller. Dream-1's shared end-effector/inverse-kinematics interface is neither automatic in WAMs nor forbidden in VLAs.

8.2 Why a video branch can be costly: an explicit memory and token budget

Peak GPU memory is not equal to the number of weights. In decimal gigabytes, theoretical parameter storage and the full memory budget can be approximated by

$$M_{\text{weights}}(\text{GB}) \simeq \frac{Pb}{8 \times 10^9}, \quad M_{\text{peak}} = M_{\text{weights}} + M_{\text{activations}} + M_{\text{KV}} + M_{\text{video}} + M_{\text{workspace}}.$$

Here P is the number of resident parameters and b the stored bits per parameter. BF16 costs **about 2 GB per billion parameters for weights alone**; this is a lower bound, **not** a GPU recommendation. Quantized inference also needs scale/zero-point data, supported kernels, and workspace. Naive attention computation scales quadratically in the token count, while video tokens increase with frame count and latent spatial resolution:

$$C_{\text{attention}} \propto LN^2d, \quad N_{\text{video}} \propto F \frac{H_{\text{img}}W_{\text{img}}}{s_H s_W}.$$

L is Transformer depth, N the attention-token count, F the number of future frames, and s_H, s_W stand for spatial compression or patch factors. These are **order-of-magnitude teaching relations**, not an exact DiT FLOP counter or a substitute for GPU profiling.

Public reference	Parameter scale	BF16 weights only	Correct interpretation
SmolVLA	About 0.45B	About 0.9 GB	Compact VLA; vision activations and cache remain
π_0	About 3.3B	About 6.6 GB	About 3B VLM plus 0.3B action expert

Public reference	Parameter scale	BF16 weights only	Correct interpretation
OpenVLA	About 7B	About 14 GB	Discrete-action VLA; training costs far exceed the weight floor
Fast-WAM	About 5B video backbone + 1B action expert	About 12 GB for these components	Other encoders, cache, and workspace excluded
DreamZero	About 14B video backbone	About 28 GB	Core BF16 weights exceed a 24 GB GPU
Dream-1	Not disclosed	Cannot estimate credibly	Cloud API pricing is not GPU cost

Model specifications are drawn from [SmolVLA](#), [π₀](#), [OpenVLA](#), [Fast-WAM](#), [DreamZero](#), and [Dream-1](#). **A weight-only estimate is neither inference VRAM nor latency.** The rows also differ in input resolution, cameras, action horizon, and runtime stack.

8.3 Training budgets: pretraining, full fine-tuning, and LoRA are different regimes

For full-parameter BF16 training with AdamW, one simplified parameter/gradient/optimizer accounting is $2 + 2 + 8 = 12$ bytes per parameter; retaining a separate FP32 master copy can raise this to about 16 bytes. **Activations, backward intermediates, and video latents are not included.** At 14B parameters, that implies a rough **168–224 GB aggregate lower-bound accounting** for these tensors; it does not imply that a GPU pool totaling that much memory will successfully train the model. Checkpointing, ZeRO/FSDP, offloading, precision, sequence length, and communication all change the actual requirement.

OpenVLA reports pretraining on about 970k robot episodes using **64 A100 GPUs for 15 days**. The official [openpi repository](#) gives single-GPU guideline thresholds of **> 8 GB for inference**, **> 22.5 GB for LoRA fine-tuning**, and **> 70 GB for full fine-tuning**; these are not universal sufficiency guarantees. DreamZero reports roughly 500 hours of AgiBot data and 100k training steps in its robotics alignment stage, in addition to its video pretrained initialization. Since the methods differ in total pretraining data, frozen modules, resolutions, batch shapes, throughput, and GPU utilization, **there is no defensible single multiplier for “WAM training cost versus VLA training cost.”**

Future-video supervision usually adds **data I/O, activation, and backpropagation** costs, but can exploit human or web videos without robot-action labels. Direct-action VLA fine-tuning is often easier to iterate on consumer GPUs. LoRA cuts trainable weight and optimizer-state footprints; it **does not eliminate forward passes through frozen backbones, necessary intermediate activations, or visual preprocessing.**

8.4 Inference: measure closed-loop decision time, not action-chunk coverage

Let T_e denote encoding time, N_a the number of action evaluations, N_j the number of joint video-action evaluations, and T_a, T_j their per-evaluation time. Ignoring overlaps, three **schematic cost models** are

$$\begin{aligned}
 T_{\text{VLA}} &\approx T_e + N_a T_a + T_{\text{IO}}, \\
 T_{\text{WAM, joint}} &\approx T_e + N_j T_j + T_{\text{IO}}, \\
 T_{\text{WAM, action only}} &\approx T_e + T_{\text{world features}} + N_a T_a + T_{\text{IO}}.
 \end{aligned}$$

For an autoregressive-token VLA, $N_a T_a$ represents cumulative token decoding rather than flow-integration steps. KV caching, joint computations, and asynchronous pipelines change realized latency. Always benchmark the **sensing–encoding–network–inference–command** path with $p50/p95/p99$ statistics.

Reported measurement	Value and hardware	What it does—and does not—show
DreamZero-Flash	About 150 ms/chunk on two GB200 GPUs , around 7 Hz high-level feedback	Highly optimized multi-GPU result, not an RTX 4090 benchmark
Fast-WAM (paper)	About 190 ms on one RTX 5090D V2 32GB	Removing full future-video generation can reduce latency
Fast-WAM (newer code optimization)	About 110 ms on an RTX 4090 for a specified LIBERO inference configuration	Repository measurement, not a guaranteed cross-task end-to-end budget
Dream-1 API	One response covers about 1.6 s of motor commands ; API p95 not disclosed	Action coverage is not request latency or local GPU cost

Sources: [DreamZero project](#), [Fast-WAM paper](#), [Fast-WAM implementation](#), and [Dream-1 announcement](#). A robot executing an action chunk at 30–50 Hz does **not** imply that the large model observes and replans every 20–33 ms. **Low-level command rate, decision rate, and chunk duration must be reported separately.**

8.5 The accuracy–generalization–compute trade-off is not monotonic

Fast-WAM finds that on its evaluations **video co-training** matters more than full test-time video synthesis, supporting action-only deployment. Yet **Faster-WAM**, under its controlled training recipe and LIBERO-Plus distribution shifts, reports 49.14% for Fast-WAM and 73.57% for its sparsely future-conditioned alternative, with the latter also **2.21 times faster than its Joint-WAM baseline**. The implication is conditional: **preserving inexpensive, compressed future information may buy valuable out-of-distribution robustness**, even if rendering a video does not.

Hence deployment choices form a continuum rather than a binary choice: **direct VLA → action policy trained with world supervision → one-pass or sparsely cached future-conditioned policy → full joint video-action generation**. Which point wins depends on contact tolerance, expected distribution shift, and GPU budget.

8.6 Engineering choices for 24 GB GPUs and contact-rich manipulation

On a single **RTX 3090/4090 with 24 GB**, reasonable starting points include SmolVLA, $\pi_0/\pi_{0.5}$ inference, appropriately configured OpenVLA, or LoRA/partially frozen VLA adaptation. However, openpi’s > 22.5 GB LoRA guideline already sits close to the 24 GB ceiling; measure the effect of batch size, camera count, and peak activation memory. **DreamZero’s 14B core weights alone require about 28 GB in BF16**, so its unmodified BF16 model cannot reside wholly on one 24 GB card. Quantization and sharding may enable experiments, but do not inherit its two-GB200 latency figure. Four separate 24 GB cards do not constitute one 96 GB address space: tensor/model parallelism and interconnect costs must be planned.

For a **robot arm and head/wrist RGB-D** performing precision manipulation, use the foundation model as a lower-frequency skill/goal/action-chunk generator, while keeping **joint servoing, IK, collision checks, tactile/current-based contact detection, and emergency stops** in a local high-rate loop. If predictive priors appear valuable, first test a **video-supervised WAM with action-only inference or one-pass future latent conditioning**. Pay for multistep video generation only when controlled ablations demonstrate gains in failure recovery, novel contacts, or disturbance robustness.

A fair systems study should hold camera views/resolution, embodiment and action semantics, robot-demonstration budget, task distribution, initial states, action-chunk duration, precision, and replanning protocol fixed. Report **end-to-end p50/p95/p99, peak VRAM, GPU-seconds per action chunk, physical task success, out-of-distribution perturbation/regrasp recovery, and safety stops**. Video quality or mean model-only latency is insufficient.

9. Efficient inference: joining the model clock and robot clock

Real deployments must coordinate camera sampling frequency f_s , model update frequency f_m , and low-level control frequency f_c . An action chunk spans $H\Delta t_c$ physical time while model computation takes T_m plus network and dispatch delays. When $f_m \ll f_c$, smooth execution is still possible with continuous chunk handover, but high-frequency contact correction must be handled separately.

Mechanism	Reduces	Caveat / required validation
Causal KV cache	Recomputing historical attention context	Ground-truth observation update and invalidation
DiT caching between flow steps	Repeated denoising backbone computation	Reuse error can affect motor quality
FlashAttention	Attention memory traffic and temporaries	End-to-end gain may be smaller than kernel-level gain
Step reduction/distillation	Number of denoising/ODE evaluations	Tail failures and contact accuracy
Asynchronous action chunks	Motion blocked during model inference	Stale chunks, timestamps, handover, timeout fallback
Quantization/multi-GPU parallelism	GPU compute and memory constraints	Precision, communication, long-tail latency

Two different caches: the causal KV cache stores attention representations of past context; DiT cache methods reuse computations across denoising steps. Their validity depends on distinct invalidation rules. General derivations of FlashAttention and KV caching belong in the companion lecture on efficient training and real-time inference.

10. Contact-rich manipulation and recovery

Visual alignment between a predicted and observed video does not reveal normal force, friction margin, or incipient slip. A practical manipulation stack should combine world predictions with joint/motor feedback, optional current and tactile sensing, and uncertainty estimates. One abstract state is

$$z_t = (T_{\text{obj}}, T_{\text{hand}}, q_t, \hat{c}_t, \Sigma_t),$$

where $\hat{c}_t$ denotes estimated contact mode and Σ_t uncertainty. A hallucinated contact force must not be treated as a measured sensor value.

Consider picking a screwdriver from an arbitrary pose and reorienting it at a fixed workstation. The world-prediction branch may anticipate motion and occlusion; the action branch chooses arm targets; the low-level controller maintains grip and joint safety. Measured current residuals, pose deviations, or possible slip must be able to trigger re-observation, hold, and recovery rather than blindly continuing an old action chunk.

11. Evaluating the contribution of a WAM

Evaluation axis	Relevant measures	Common mistake
Predicted dynamics	Contact/kinematic consistency, calibrated uncertainty	Equating beautiful video with physical correctness
Policy generalization	Completion rate, task progress, novel tasks/poses	Conflating progress and success
Cross embodiment	Target-robot data amount, distribution of scores across robots	Reporting only a selected highlight reel
Feedback robustness	Occlusions, dropped objects, perturbation recovery	Treating open-loop replay as closed-loop robustness
Runtime	Full-loop p50/p95/p99, network jitter	Calling 1.6 s of action coverage “inference latency”
Safety	Contact/velocity limits, collision and timeout fallbacks	Assuming imagined trajectories enforce safety

Causal claims require separating the contributions of video pretraining, joint multimodal training, shared action representations, genuine closed-loop re-grounding, and embodiment-specific fine-tuning. Headline scores from incompatible datasets, robots, and success definitions should not be merged into a single ranking.

12. Boundaries within the course and reading order

This chapter covers WAMs from prediction supervision through executable actions, from DreamZero's detailed joint generation to Dream-1's shared robot interface. Refer to the Flow Matching lecture for general probability-path mathematics; imitation learning for BC, DAgger, ACT, and RTC; RL post-training for policy updates; GR00T/SONIC for System 1–System 0 execution; and Robot Agent for outer-loop planning, tool orchestration, success checks, and replanning. **DreamZero and Dream-1 are primarily learned action systems, not automatically complete tool-using Robot Agents.**

Suggested progression: taxonomy → aligned data → DreamZero joint FM/KV/Flash → Dream-1 EEf/IK and zero-shot embodiment generalization → latency, safety, and contact boundaries. A central research question is whether visual dynamics priors can bridge not only arm kinematics, but also end-effector morphology and contact models, without losing motor precision.

13. References and official resources

1. Zhang, Zeng, Zhang. *From World Models to World Action Models: A Concise Tutorial for Robotics*, 2026. Overview of concepts and architectural families.
2. Ye et al. *World Action Models are Zero-shot Policies (DreamZero)*, 2026. Joint 14B model, training objectives, online execution, transfer evidence.
3. NVIDIA GEAR. *DreamZero project page and public code/models*. Implementation and released media.
4. Dream Labs. *Zero-Shot to New Robots: Introducing Dream-1*, October 2026. EEf/IK transfer, API, unseen robots, FaaS. **Product article, not a full technical paper.**
5. Yuan et al. *Fast-WAM: Do World Action Models Need Test-time Future Imagination?*, 2026. Training-time future prediction vs. test-time imagination.
6. Chi et al. *Universal Manipulation Interface*, 2024. Cross-robot transfer with matched grippers and wrist-camera interfaces.
7. NVIDIA. *Cosmos; Physical Intelligence*. openpi. Engineering entry points for world and motor models.
8. Lipman et al. *Flow Matching for Generative Modeling*, 2023. Mathematical foundation of generative velocity fields.
9. Kim et al. *OpenVLA: An Open-Source Vision-Language-Action Model*, 2024; [project website](#). A 7B VLA and its 64-A100 pretraining reference.
10. Black et al. *π_0 : A Vision-Language-Action Flow Model for General Robot Control*; openpi code and memory guidance.
11. Hugging Face. *SmolVLA*, 2025. A compact 450M VLA and consumer-GPU workflow.
12. Yuan et al. *Fast-WAM; official implementation*. Video co-training, action-only inference, and reported optimized runtimes.
13. Zhao et al. *Faster-WAM*, 2026. Sparse future conditioning and LIBERO-Plus OOD evaluation.
14. Zha et al. *LAP: Language-Action Pre-Training Enables Zero-shot Cross-Embodiment Transfer*, RSS 2026. An important VLA counterexample reporting zero-shot transfer without target-embodiment fine-tuning.

Editorial note. Equations marked schematic are explanatory, not claims about proprietary implementations. No model training or real-robot experiments were conducted for this teaching note. Sources last checked October 22, 2026; neither the Dream-1 launch materials nor these notes constitute a safety certification or performance guarantee.