LEON: OPERATOR-STRUCTURED TRANSITIONS FOR WORLD ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent world models for embodied agents model how system states evolve under action. In robot manipulation, World Action Models (WAMs) couple such predictions to policy learning as supervision or future conditions for action generation. Yet their latent transitions typically rely on general-purpose predictors, leaving dynamical structure to be learned implicitly from data. We derive a latent-transition formulation from controlled mechanical evolution. A common finite-dimensional projection of observable dynamics yields a shared operator family affine in physical control and an explicit projection residual. Guided by these results, we introduce the Latent Evolution Operator Network (LEON), which adapts pretrained visual features into learned observables, learns latent control coefficients from the WAM context, and combines factorized operator evolution with complementary additive updates. We instantiate LEON in prediction-supervised policy learning and future-conditioned action generation while preserving the original prediction targets and policy interfaces. Across LIBERO, LIBERO-Plus, and RoboTwin 2.0, LEON improves average success rates over the corresponding WAM baselines and matched transition controls. Ablations support the joint representation and transition design, and controlled-system diagnostics show improved state-range extrapolation. These results establish dynamics-derived latent-transition structure as an effective inductive principle for latent WAMs.

1 INTRODUCTION

A central objective of world modeling for embodied agents is to anticipate how the physical environment evolves under action, allowing future consequences to inform interaction and control (Hafner et al., 2019). For visually observed environments, latent world models formulate this prediction over encoded observations, avoiding full visual reconstruction in pixel space (Zhou et al., 2024; Assran et al., 2025; Sun et al., 2026). In robot manipulation, latent World Action Models (WAMs) integrate such future prediction with policy learning, using it either as training supervision or as a future condition for action generation (Sun et al., 2026; Chen et al., 2026; Yuan et al., 2026). In either role, the predictive core is a latent transition conditioned on action-related representations. This raises a more fundamental question for latent world modeling: How should the underlying physical evolution shape this transition's internal formulation?

The predicted changes share a common mechanical origin: within a given mechanical system and smooth interaction regime, system evolution under different states and control inputs is governed by the same equations of motion, which relate inertia, internal forces, and control to the resulting motion (Spong et al., 2020). Visual observation and encoding recast this same evolution in the variables used by the model (Watter et al., 2015; Lusch et al., 2018), while its underlying mechanical dynamics remain unchanged. Representative latent WAMs parameterize the resulting predictive relation with Transformer-based predictors, leaving the organization implied by the underlying dynamics to be learned implicitly from data (Sun et al., 2026; Chen et al., 2026). This motivates a latent formulation that carries the structural organization of controlled mechanics into the learned model, jointly informing how state is represented and how it evolves under control.

This joint view admits a natural formulation in terms of differentiable observables: their choice specifies the state description, while the governing dynamics determine their evolution. For a broad class of actuated mechanical systems, these dynamics admit a control-affine form (Spong et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2020). By the chain rule, this affine control dependence is preserved in the instantaneous evolution of any differentiable observable. Koopman theory expresses this observable evolution through a generator that decomposes affinely in control into shared linear components (Proctor et al., 2018; Peitz et al., 2020; Otto et al., 2024). Projection onto a finite observable subspace converts these generator components into finite-dimensional matrices, with a residual accounting for the projection error (Klus et al., 2020). The fidelity of this finite realization depends on the observable subspace. This yields a general structure for latent world modeling: in an appropriate observable space, controlled evolution can be represented by a shared operator family affine in control, complemented by unresolved change.

Building on this principle, we introduce the Latent Evolution Operator Network (LEON), which instantiates the finite-observable evolution derived above within a learnable latent world model. LEON learns an observable description of the visually observed state and constructs a control abstraction from the observables and existing action-related representations to modulate a shared operator family. A complementary additive update accounts for evolution beyond the restricted operator family, while a learned readout returns the evolved observables to the retained prediction space. The observable map, control abstraction, and evolution operators are learned jointly under the original predictive objective. We instantiate the same formulation in VLA-JEPA and LaWAM, representative of prediction-supervised policy learning and predicted-future-conditioned action generation, fully replacing their Transformer transitions while preserving the original prediction targets and policy interfaces (Sun et al., 2026; Chen et al., 2026).

Our contributions are threefold: (i) We derive a dynamics-grounded formulation of latent world modeling from controlled mechanical evolution, linking the observable state description with a shared operator structure for controlled evolution; (ii) We introduce LEON, which realizes this principle through jointly learned observable adaptation, latent control, and shared operator evolution with complementary updates; (iii) We evaluate the same formulation under two prediction–policy interfaces, with manipulation benchmarks, matched transition ablations, and controlled-system diagnostics examining its performance and structural choices.

2 Related Work

World models for robot manipulation. World models have become an increasingly prominent component of robot manipulation, where future prediction augments policy learning and execution with explicit modeling of action consequences. Existing methods differ in the representation of the predicted future: WorldVLA predicts future visual observations, whereas DINO-WM and VLA-JEPA predict pretrained visual features (Cen et al., 2025; Zhou et al., 2024; Sun et al., 2026), with recent work further exploring fine-grained visual rollouts and latent diffusion prediction (Zhu et al., 2025; Huang et al., 2025). Their use of prediction within the policy also varies. VLA-JEPA and Fast-WAM employ predictive objectives as training-time supervision, whereas Seer, CoT-VLA, and LaWAM condition action generation on predicted visual states or latent subgoals (Sun et al., 2026; Yuan et al., 2026; Tian et al., 2025; Zhao et al., 2025; Chen et al., 2026). These works establish different predictive representations and prediction–policy interfaces, while latent evolution is typically realized by general-purpose neural predictors. LEON instead studies how the underlying controlled evolution can organize the latent state description and its action-conditioned transition within visual WAMs.

Koopman-structured dynamics for robot learning and world models. Koopman-based structure has been used to couple learned visual representations with dynamics and control in robot learning. KOROL learns visual object features through Koopman rollout supervision for trajectory propagation, while RoboKoop learns task-conditioned visual representations for stabilizing control (Chen et al., 2024; Kumawat et al., 2024). Operator World Models use conditional mean embeddings to learn MDP transition operators for value estimation and policy optimization (Novelli et al., 2024). Koopman Dreamer introduces spectrally structured, action-conditioned latent dynamics for long-horizon imagination in continuous-control world models (Li et al., 2026a). Prior work introduces operator structure for representation learning, policy optimization, or long-horizon imagination. LEON instead starts from controlled mechanical evolution to formulate latent transitions in visual WAMs, jointly structuring the learned observable representation and its action-conditioned evolution while retaining the original role of future prediction in policy learning or action generation.

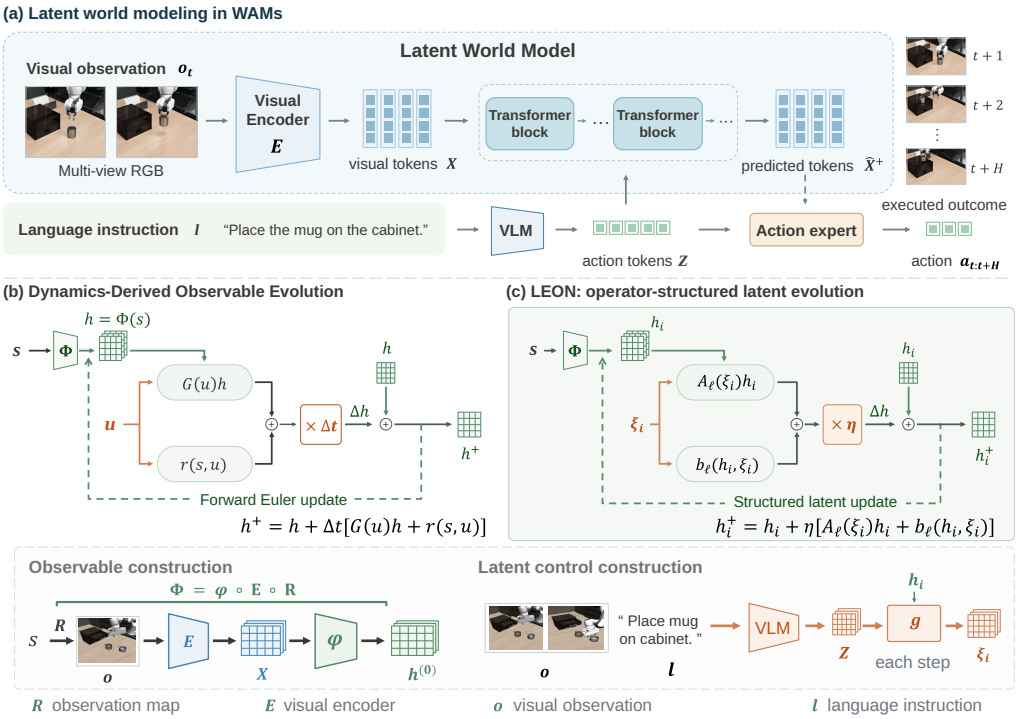

Figure 1: **From latent world prediction to operator-structured evolution.** (a) Latent WAMs predict future visual representations from the current observation and action-related context. (b) Controlled dynamics motivate observable evolution as shared operator propagation with complementary residual change. (c) LEON realizes this principle in learned visual observables using latent-control-modulated operators and additive updates. The observable and latent-control constructions are shown below.

3 Controlled Dynamics as a Transition Design Principle

Given visual observations, WAMs predict how the underlying robot–environment system evolves under action. We formulate this physical evolution as a control-affine dynamical system and use the Koopman operator perspective to expose a shared-operator structure in observable space for controlled evolution, which motivates our transition architecture.

Controlled mechanical evolution. Consider a robot–environment system within a smooth interaction regime. Its mechanical dynamics take the form $M(q)\ddot{q} + c(q, \dot{q}) = B(q)u$, where q denotes generalized coordinates, $u \in \mathbb{R}^m$ denotes physical control inputs, M is a nonsingular inertia matrix, c collects non-actuation terms, and B maps control inputs to generalized forces (Spong et al., 2020). With the physical state $s = (q, \dot{q}) \in \mathbb{R}^{d_s}$, the second-order dynamics are equivalently written as the first-order control-affine system

$$\dot{s} = f(s, u) = f_0(s) + \sum_{j=1}^{m} u_j f_j(s). \tag{1}$$

The drift f_0 and input vector fields $\{f_j\}_{j=1}^m$ are shared across states and control values of the same system, with f_0 governing the evolution when $u = 0$. These fields may depend nonlinearly on s, while each control component u_j weights the corresponding input field f_j.

Shared generators in observable space. To express the same controlled evolution through state-dependent quantities, let $\Phi = (\phi_1, \ldots, \phi_d)^\top : \mathbb{R}^{d_s} \to \mathbb{R}^d$ be a differentiable observable map and define $h = \Phi(s)$. For a fixed input u, the Koopman generator L_u maps each observable function to its instantaneous rate of change along the system dynamics (Proctor et al., 2018). It acts linearly on observable functions and componentwise on Φ, giving $\dot{h} = (L_u\Phi)(s) = D\Phi(s)f(s, u)$ by the

chain rule. Substituting Eq. (1) yields $L_u = L_0 + \sum_{j=1}^{m} u_j L_j$, where $(L_j\Phi)(s) = D\Phi(s) f_j(s)$ for $j = 0, \ldots, m$. Thus, different control values modulate the contributions of the same input generators $\{L_j\}_{j=1}^{m}$, preserving affine control dependence in observable space (Peitz et al., 2020). These generators act on a generally infinite-dimensional function space, and their linearity does not by itself imply closed linear dynamics for the finite vector h (Otto et al., 2024). Using this shared structure in a learned transition requires a finite-dimensional representation of the generator action.

Finite observable representations. Let $\mathcal{V}_\Phi = \operatorname{span}\{\phi_1, \ldots, \phi_d\}$ be the finite observable subspace, and let P_Φ be a fixed linear projection onto this subspace, applied componentwise to vector-valued observables. For each generator component, choose $G_j \in \mathbb{R}^{d \times d}$ such that $P_\Phi L_j \Phi = G_j \Phi$, expressing the projected observable derivatives as linear combinations of the components of Φ (Klus et al., 2020). Defining $r_j = (I - P_\Phi) L_j \Phi$ yields the exact decomposition

$$\dot{h} = \underbrace{G(u)h}_{\text{shared operator dynamics}} + \underbrace{r(s,u)}_{\text{projection residual}}, \qquad G(u) = G_0 + \sum_{j=1}^{m} u_j G_j, \tag{2}$$

where $r(s,u) = r_0(s) + \sum_{j=1}^{m} u_j r_j(s)$. The common linear projection preserves affine control dependence without requiring exact finite closure. The residual accounts for observable change outside the selected subspace and vanishes when $\mathcal{V}_\Phi$ is invariant under all generator components. Appendix A.2 details an orthogonal projection construction.

For these matrix representations, the control-dependent part $G(u) - G_0$ lies in $\mathcal{S}_G = \operatorname{span}\{G_1, \ldots, G_m\}$. Its dimension $r_* := \dim(\mathcal{S}_G) \leq m$ gives the number of independent directions in this matrix family. Choosing a basis $\{\widetilde{G}_k\}_{k=1}^{r_*}$ gives $G(u) = G_0 + \sum_{k=1}^{r_*} (Tu)_k \widetilde{G}_k$, where $T \in \mathbb{R}^{r_* \times m}$ maps physical control to coefficients in this shared basis. The choice of observables affects which changes the shared operators capture and which remain unresolved, motivating joint learning of the observable representation and its control-dependent evolution in LEON.

Discrete observable evolution. If $\dot{h}$ is Lipschitz continuous on $[t, t + \Delta t]$ for $\Delta t > 0$, integrating Eq. (2) gives

$$h^+ = h + \Delta t \, [G(u)h + r(s,u)] + \mathcal{O}(\Delta t^2), \tag{3}$$

where $h^+ = h(t + \Delta t)$ and the increment is evaluated at t. Neglecting the remainder yields a forward Euler update that adds operator and residual contributions to the current observable. This first-order update retains affine control dependence, which need not hold for exact finite-time evolution. Together, these derivations establish the organization used by LEON: represent the state in learned observables, modulate shared operator directions, and retain a complementary update for unresolved change. The next section specifies how this organization is preserved under learned conditioning and a computationally restricted operator family.

4 Latent Evolution Operator Network

Turning the derived structure into a learned transition. We introduce the *Latent Evolution Operator Network* (LEON), a latent transition architecture that jointly learns observable representations and their control-dependent evolution. Given a visual observation $o = R(s)$ of physical state s, the transition receives features $X = E(o)$ from the retained visual encoder and predictor input projection. To obtain observables suited to the learned operator family, a learned map φ constructs observables $h = \varphi(X)$, so that $\Phi = \varphi \circ E \circ R$. The overall policy maps o and language instruction l to physical control, $u = \pi(o, l)$. From this context, we learn a latent control variable ξ to encode control information relevant to observable evolution, without requiring recovery of u. The WAM transition problem is thus to predict the future observable h^+ from (h, ξ).

To realize this transition, we adopt the shared-operator organization derived in Section 3, where the reduced coefficients Tu weight shared operator directions. LEON uses $\xi \in (-1, 1)^{r_A}$ to weight r_A learned shared operator directions, with r_A controlling the capacity of the operator family. To predict the future observable h^+ from (h, ξ), LEON composes learned updates following the forward-Euler organization of Eq. (3):

$$\widehat{h}^+ = h + \eta \, [A(\xi)h + b(h, \xi)], \tag{4}$$

where $\eta > 0$ scales the increment, $A(\xi)$ combines the shared operator directions affinely in ξ, and $b(h, \xi)$ captures complementary change beyond the restricted operator family. Each update thus retains the current observable and adds operator-mediated evolution together with a complementary correction. A learned readout maps the updated observable back to the retained prediction space.

4.1 Operator-Structured Latent Transitions

Learned observable space. Given the retained visual feature tokens $X = [x_1, \dots, x_N]^\top = E(o) \in \mathbb{R}^{N \times d_x}$, LEON constructs observables through the token-wise mapping $h_i = \varphi(x_i) = W_\varphi x_i + g_\varphi(\mathrm{LN}(x_i)) \in \mathbb{R}^{d_h}$, where $W_\varphi \in \mathbb{R}^{d_h \times d_x}$ is a learned linear projection and g_φ is an MLP residual branch. Thus, applying φ with shared parameters across tokens gives the observable representation $h = [h_1, \dots, h_N]^\top = \Phi(s)$, where $\Phi = \varphi \circ E \circ R$. Learning φ jointly with the evolution operators allows this representation to adapt to the restricted operator family.

Observable-conditioned latent control. Let Z denote the WAM's action-related representation inferred from the current observation and language instruction. For each observable h_i, we construct a latent control representation $\xi_i = \tanh\big(f_\xi(g_{\text{fuse}}(h_i, Z))\big) \in (-1, 1)^{r_A}$, where g_{fuse} combines h_i with Z, and f_ξ maps the fused representation to $\mathbb{R}^{r_A}$. Conditioning on h_i allows the same action-related context to modulate the evolution of each observable differently. The bounded, signed components of ξ_i weight the shared operator directions, and ξ_i also conditions the complementary update $b(h_i, \xi_i)$. Although the control construction is nonlinear in (h_i, Z), the operator $A(\xi_i)$ remains affine in ξ_i. Section 4.2 specifies the fusion map for each WAM instantiation.

Shared operator family. To realize the shared affine structure in the learned observable space, LEON parameterizes the operator family as

$$A(\xi_i) = D + \sum_{k=1}^{r_A} \xi_{ik} A_k = D + U \,\mathrm{diag}(\xi_i) V^\top. \tag{5}$$

Here, $D \in \mathbb{R}^{d_h \times d_h}$ is a learned diagonal base, $U = [\mathbf{p}_1, \dots, \mathbf{p}_{r_A}]$ and $V = [\mathbf{q}_1, \dots, \mathbf{q}_{r_A}]$ are learned factor matrices in $\mathbb{R}^{d_h \times r_A}$. Each shared direction has the rank-one form $A_k = \mathbf{p}_k \mathbf{q}_k^\top$. Its action on h_i produces a change along $\mathbf{p}_k$, scaled by the observable combination $\mathbf{q}_k^\top h_i$, with its contribution further weighted by ξ_{ik}. Within each update, D, U, and V are shared across observables and input conditions, while ξ_i determines the combination of the shared directions.

Let $\mathcal{S}_A = \mathrm{span}\{A_1, \dots, A_{r_A}\}$. Then $A(\xi_i) - D \in \mathcal{S}_A$, with $\dim(\mathcal{S}_A) \le r_A$ and $\mathrm{rank}(A(\xi_i) - D) \le \min(d_h, r_A)$. Thus, r_A bounds both the number of independent operator directions and the rank of the condition-dependent operator variation. The diagonal base provides a condition-independent component-wise contribution, and the factorized directions allow cross-component interactions. The operator action is evaluated as $A(\xi_i)h_i = Dh_i + U[\xi_i \odot (V^\top h_i)]$, at $\mathcal{O}(d_h r_A)$ cost per observable token without forming a dense operator matrix.

Complementary update and composition. The unresolved contribution $r(s, u)$ in Eq. (2) motivates a learned complementary update $b(h_i, \xi_i)$ for changes beyond the restricted operator family. We implement b as a normalized two-layer MLP conditioned on the observable and latent control. The operator and complementary branches are both initialized to produce zero observable increments, while the observable map and readout remain active.

Starting from $x_i^{(0)} = x_i$, LEON composes L learned evolution updates with separate parameters and shared context Z. For $\ell = 0, \dots, L-1$, each update constructs $h_i^{(\ell)} = \varphi_\ell(x_i^{(\ell)})$ and $\xi_i^{(\ell)}$ as above, applies Eq. (4) to obtain $\widehat{h}_i^{+(\ell)}$, and reads out $x_i^{(\ell+1)} = \psi_\ell(\widehat{h}_i^{+(\ell)})$ through normalization and an affine projection. The retained output projection maps the final features to the original WAM prediction space. Here, ℓ indexes network depth, and η scales the internal increments. Appendices A.5 and C.1 detail the computation and initialization.

4.2 Instantiation across Prediction–Policy Interfaces

We instantiate LEON in two representative WAM architectures with distinct prediction–policy interfaces while retaining their prediction spaces, targets, and policy interfaces. Both instantiations

use the same operator-structured transition formulation, while the available action representation determines the fusion map g_{fuse}, and the WAM determines how prediction influences the policy.

Prediction-supervised policy learning. In VLA-JEPA (Sun et al., 2026), future prediction supervises policy representation learning without providing an explicit future condition to the action head at deployment. For each future prediction, the VLM provides a set of K latent-action tokens $Z \in \mathbb{R}^{K \times d_z}$ from the current observation and task instruction. We construct g_{fuse} from an observable-specific action summary and a global action summary. Projecting and normalizing Z to the observable dimension gives $\bar{Z}$. The observable-specific summary is $c_i^{\text{loc}} = \text{MHA}(\text{LN}(h_i), \bar{Z}, \bar{Z})$, where MHA denotes multi-head attention with the normalized observable as the query and $\bar{Z}$ as the keys and values. In parallel, a two-layer MLP forms the global summary $c^{\text{glob}} = g_{\text{glob}}(\text{Mean}(\bar{Z}))$. The two summaries are fused as $g_{\text{fuse}}(h_i, Z) = g_{\text{merge}}([c_i^{\text{loc}}; c^{\text{glob}}])$, where $[\cdot;\cdot]$ denotes concatenation along the feature dimension, and g_{merge} applies normalization followed by an affine projection to $\mathbb{R}^{d_h}$. Following Section 4.1, f_ξ and $\tanh$ map the fused representation to the latent control ξ_i, which weights shared operator directions and, with h_i, conditions the complementary update $b(h_i, \xi_i)$.

The retained output projection maps LEON's final features to $\widehat{X}^+$ in the frozen V-JEPA target space. We train on robot demonstrations with the task objective $\mathcal{L}_{\text{task}} = \mathcal{L}_{\text{act}} + \beta \mathcal{L}_{\text{WM}}$, where $\mathcal{L}_{\text{WM}}$ supervises future prediction using X^+ as the target, and $\mathcal{L}_{\text{act}}$ supervises action generation. Gradients of $\mathcal{L}_{\text{WM}}$ pass through LEON and the latent-action pathway to the shared VLM, thereby shaping the embodied-action representations that condition the action head. At inference, the action head generates action sequences conditioned on the embodied-action representations, without using predicted future features as conditioning inputs.

Future-conditioned action generation. In LaWAM (Chen et al., 2026), the predicted future representation directly conditions action generation. The current-state features and future prediction targets remain in the frozen DINOv3 feature space. The WAM supplies a single aggregated latent-action representation Z from the current observation and language instruction, so no further token-level action aggregation is needed. After the retained action projection P_Z and normalization, a two-layer MLP forms the global summary $c^{\text{glob}} = g_{\text{glob}}(\text{LN}(P_Z(Z))) \in \mathbb{R}^{d_h}$. This summary is fused with each observable as $g_{\text{fuse}}(h_i, Z) = g_{\text{merge}}([h_i; c^{\text{glob}}])$, where $g_{\text{merge}} : \mathbb{R}^{2d_h} \to \mathbb{R}^{d_h}$ is a normalized two-layer MLP. Following Section 4.1, f_ξ and $\tanh$ map the fused representation to the latent control ξ_i, which weights the shared operator directions and, together with h_i, conditions the complementary update $b(h_i, \xi_i)$.

LEON predicts the future representation $\widehat{X}^+$ from (X, Z). Training retains the task objective $\mathcal{L}_{\text{task}} = \mathcal{L}_{\text{act}} + \lambda_{\text{WM}} \mathcal{L}_{\text{WM}} + \lambda_{\text{distill}} \mathcal{L}_{\text{distill}}$, where $\mathcal{L}_{\text{WM}}$ supervises future prediction using X^+ as the target, and $\mathcal{L}_{\text{act}}$ supervises action generation. The distillation loss $\mathcal{L}_{\text{distill}}$ aligns Z with a teacher latent action inferred from the observed current–future pair. The retained flow-matching action expert is conditioned on X, $\widehat{X}^+$, and the VLM context H_{VLM} from which Z is obtained. As in LaWAM, $\mathcal{L}_{\text{act}}$ does not backpropagate through the predicted-future input, while $\mathcal{L}_{\text{WM}}$ updates LEON and the latent-action pathway. At inference, Z and $\widehat{X}^+$ are computed from the current observation and language instruction, and the action expert generates actions conditioned on $(X, \widehat{X}^+, H_{\text{VLM}})$. Appendices C.2 and C.3 detail the feature pipelines, losses, and gradient paths for both WAMs.

5 Experiments

We organize the experiments around three questions: (i) whether the proposed latent-transition formulation improves policy performance across the two WAM formulations; (ii) whether its core structural choices account for the observed performance differences, particularly under distribution shift; and (iii) whether the structured transition provides a measurable predictive advantage over generic predictors under extrapolative evaluation on controlled physical systems with known dynamics. To address (i), we instantiate LEON under both prediction–policy interfaces and evaluate policy performance in the corresponding WAM settings. For (ii), matched transition variants keep the retained WAM components and experimental protocol fixed while changing only the specified transition design. Controlled-system diagnostics address (iii) independently of policy learning. Full experimental details are provided in Appendices B–E.

Table 1: LIBERO success rates (%). Results for prior methods are taken from their original papers unless marked †. † denotes our re-evaluation of VLA-JEPA and LaWAM under their original settings using three evaluation seeds matched to the corresponding LEON variants, with 50 trials per task per seed and seed-wise statistics reported in Appendix D.1. Avg. denotes the mean over all 40 tasks. Best and second-best results are shown in bold and underlined, respectively.

Method	Spatial	Object	Goal	LIBERO-10	Avg.
OpenVLA-OFT (Kim et al., 2025)	97.6	98.4	97.9	94.5	97.1
$\pi_{0.5}$ (Black et al., 2025)	98.8	98.2	98.0	92.4	96.9
UniVLA (Bu et al., 2025)	96.5	96.8	95.6	92.0	95.2
WorldVLA (Cen et al., 2025)	87.6	96.2	83.4	60.0	81.8
Fast-WAM (Yuan et al., 2026)	98.2	**100.0**	97.0	95.2	97.6
Cosmos-Policy (Kim et al., 2026)	98.1	**100.0**	98.2	97.6	98.5
LingBot-VA (Li et al., 2026b)	98.5	99.6	97.2	**98.5**	98.5
VLA-JEPA† (Sun et al., 2026)	98.0	99.7	98.4	95.4	97.9
LaWAM† (Chen et al., 2026)	98.1	98.9	98.1	97.1	98.1
VLA-JEPA + LEON	**99.5**	99.7	<u>99.0</u>	<u>97.8</u>	**99.0**
LaWAM + LEON	<u>99.2</u>	<u>99.8</u>	**99.4**	97.3	<u>98.9</u>

Table 2: RoboTwin 2.0 success rates (%) on 50 tasks under mixed clean-and-randomized training, with 100 trials per task. Clean and Randomized denote nominal and domain-randomized evaluation, respectively. Avg. is their mean. Prior-method results follow the LaWAM evaluation (Chen et al., 2026). Best and second-best results are shown in bold and underlined, respectively.

Method	Clean	Randomized	Avg.
Fast-WAM	91.98	<u>90.52</u>	<u>91.25</u>
GigaWorld-Policy	86.36	85.04	85.70
LingBot-VA	91.50	**90.92**	91.21
$\pi_{0.5}$	82.74	76.76	79.75
Motus	88.66	87.02	87.84
LaWAM	<u>92.64</u>	89.80	91.22
LaWAM + LEON	**93.48**	90.02	**91.75**

5.1 Policy Performance Across WAMs

LIBERO (Liu et al., 2023) spans spatial, object, goal, and long-horizon manipulation, providing a shared multi-task setting for evaluating policy performance. VLA-JEPA + LEON and LaWAM + LEON are jointly trained on the same 40-task demonstration set used by their corresponding WAMs. As shown in Table 1, LEON raises the average SR from 97.9% to 99.0% in VLA-JEPA and from 98.1% to 98.9% in LaWAM. The former improves three suites and matches Object, while the latter improves all four, so the gains are not confined to a single task category. Their consistent direction under the two prediction–policy interfaces supports the applicability of the proposed latent-transition formulation across both WAM settings.

RoboTwin 2.0 (Chen et al., 2025) extends the policy-facing evaluation to a broader set of bimanual manipulation tasks with domain-randomized evaluation. As shown in Table 2, LaWAM + LEON increases SR from 92.64% to 93.48% on Clean and from 89.80% to 90.02% on Randomized, yielding the highest overall average of 91.75% among the compared methods. Fast-WAM and LingBot-VA remain higher on Randomized. The improvement therefore extends beyond the nominal condition, providing complementary evidence for the policy-facing LEON integration on diverse bimanual tasks and randomized environments.

5.2 Transition Design and Ablations

We examine transition realization, observable adaptation, and operator modulation with update composition within VLA-JEPA. All variants use the same training and evaluation setup and differ

Table 3: VLA-JEPA transition ablations (success rate, %). LIBERO reports the average across four suites. LIBERO-Plus reports four representative perturbation categories and the average across all seven. For LEON variants, the reported increment is scaled by η before residual addition. Bold and underlining indicate the best and second-best results.

Transition variant	Increment	LIBERO	LIBERO-Plus				
		Avg.	Camera	Light	Background	Layout	Avg.
Transformer	—	97.4	53.3	94.4	93.5	82.1	75.5
Residual MLP	$g(h, \rho)$	97.7	56.8	92.6	92.0	85.7	75.2
w/o observable adaptation	$A(\xi)x + b(x, \xi)$	98.1	58.1	92.4	90.6	81.1	74.0
w/o operator modulation	$\bar{A}h + b(h, \xi)$	97.6	63.6	<u>96.2</u>	95.5	<u>86.4</u>	<u>78.4</u>
Operator only	$A(\xi)h$	97.8	63.3	93.4	92.0	81.4	77.8
Additive only	$b(h, \xi)$	<u>98.2</u>	<u>63.9</u>	94.0	<u>96.6</u>	82.4	77.0
LEON	$A(\xi)h + b(h, \xi)$	**99.0**	**66.2**	**98.9**	**98.0**	**87.7**	**80.6**

only in the specified transition component. Since performance on LIBERO is near saturation, we also evaluate the same checkpoints on LIBERO-Plus (Fei et al., 2026) without adaptation to expose differences under distribution shift. Full definitions and results appear in Appendix D.2.

Transition realization. We first isolate the contribution of the structured transition parameterization from transition reinitialization and generic nonlinear capacity. The *Transformer* control uses the original 12-block VLA-JEPA predictor with reinitialized blocks, while all non-transition components share the same pretrained initialization and training protocol as LEON. The parameter-comparable *Residual MLP* retains LEON's observable adaptation, conditioning, readout, and residual connection, but replaces the complete increment with a conditioned two-layer mapping $g(h, \rho)$, where $\rho = g_{\text{fuse}}(h, Z)$. LEON exceeds Transformer and Residual MLP by 1.6 and 1.3 points on LIBERO, with the margins widening to 5.1 and 5.4 points on LIBERO-Plus. These matched controls indicate that the gains are not attributable to transition reinitialization or generic conditioned nonlinear capacity alone, supporting the structured parameterization under the matched protocol.

Observable adaptation. We next examine the role of the learned observable-space adaptation in the structured transition. *w/o observable adaptation* removes the observable map φ and readout ψ_ℓ and applies the operator and additive updates directly in the retained predictor feature space. It also changes the evolution width from 768 to 1024. The LIBERO-Plus average falls from 80.6% to 74.0%, the largest decrease among the LEON component variants, compared with a 0.9-point reduction on LIBERO. The substantially larger separation under distribution shift supports the effectiveness of the adaptation module as a whole, consistent with the dependence of finite-dimensional operator approximation on the chosen observable subspace discussed in Section 3.

Operator modulation and update composition. We examine the roles of input-dependent operator modulation and the two update paths. *w/o operator modulation* replaces the token-dependent coefficients $\xi_i^{(\ell)}$ in $A_\ell(\xi)$ with a learned layer-specific constant $\bar{\xi}_\ell = \tanh(\bar{c}_\ell)$, yielding $\bar{A}_\ell = D_\ell + U_\ell \operatorname{diag}(\bar{\xi}_\ell) V_\ell^\top$. The coefficients are shared across inputs and tokens, while the latent control and conditional additive path remain unchanged. This reduces the LIBERO-Plus average from 80.6% to 78.4%, indicating that the shared operator family remains effective without input-dependent coefficients, while their modulation provides an additional gain. *Operator only* removes $b_\ell(h, \xi)$, whereas *Additive only* removes the complete $A_\ell(\xi)h$ term, including $D_\ell h$, while retaining the observable adaptation, latent-control construction, and original additive branch. Their LIBERO-Plus averages decrease to 77.8% and 77.0%, respectively. Unlike the generic *Residual MLP*, *Additive only* therefore isolates LEON's existing correction path rather than replacing the complete structured increment. The larger degradation from either single-path variant supports complementary contributions from operator propagation and additive correction, with input-dependent modulation providing a smaller additional benefit.

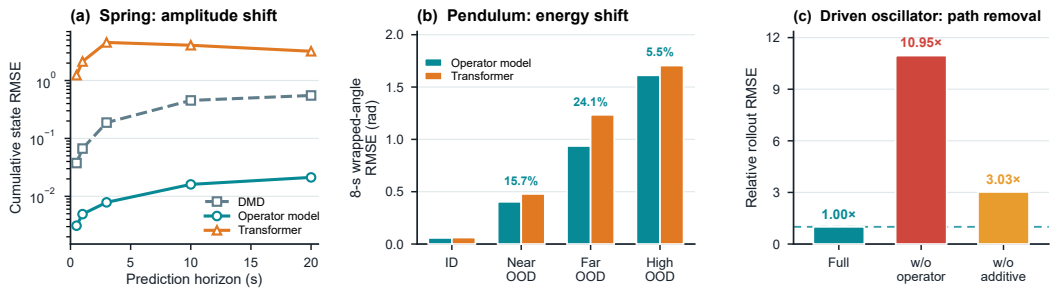

Figure 2: **Controlled diagnostics of structured prediction.** (a) Cumulative state RMSE on damped-spring trajectories beyond the training amplitude range. (b) Mean 8-s wrapped-angle RMSE across pendulum energy regimes; percentages denote relative RMSE reductions of the operator-structured predictor over the Transformer. (c) Evaluation-time path removal without retraining. Bars show the ratio of mean OOD rollout RMSE across ten seeds to the Full mean, using the same trained Full models and test trajectories for all three conditions.

5.3 Controlled Diagnostics of Structured Prediction

We use physical systems with known governing dynamics to examine LEON's design of modeling state evolution through shared operators, independently of policy learning. The Operator model adopts LEON's multiplicative operator structure, acting on physical states in the spring and learned observables in the pendulum. The spring provides exact linear closure, where the same operator governs trajectories with different initial amplitudes. With training observation noise of standard deviation 0.015, the Operator model achieves lower cumulative state RMSE than the Transformer and DMD on clean amplitude-OOD trajectories at all reported horizons (Fig. 2a). The pendulum examines this design under nonlinear dynamics: the Operator model reduces mean cumulative 8-s wrapped-angle RMSE relative to the Transformer by 15.7%, 24.1%, and 5.5% in the Near-OOD, Far-OOD, and High-OOD regimes, respectively (Fig. 2b). Shared operator, the variant with input-independent operator coefficients, also outperforms the Transformer and achieves lower mean 8-s angle RMSE than the Operator model in all three regimes (Table E.4). Together, these diagnostics support LEON's use of shared operator structure to model evolution in physical and learned observable spaces.

We use a driven oscillator to examine LEON's operator and additive updates, as its dynamics combine forces depending on position and velocity with an additive external force. In the complete predictor (Full), stiffness, which controls how strongly the oscillator is pulled toward equilibrium, conditions the operator path, while the external force conditions the additive path. Across ten independent training seeds, we remove each path without retraining, preserving the identity residual and using identical test trajectories with stiffness beyond the training range. Both removals increase error in every seed, yielding mean OOD rollout RMSE of $10.95\times$ and $3.03\times$ that of Full for w/o operator and w/o additive, respectively (Fig. 2c). These changes reflect dependence on each path together with its associated stiffness or force input, which is also disabled. Full also achieves lower rollout error than separately trained Residual MLP and Transformer baselines with comparable parameter counts and matched data, multistep objectives, and training budgets (Table E.7), supporting the joint update design. Experimental details are provided in Appendix E.1–E.4.

6 Conclusion

We derive an operator-structured transition design for latent World Action Models from control-affine observable dynamics and characterize the representational capacity of its factorized operator family. LEON jointly learns observables, latent control, and operator–complementary updates while retaining the WAM's prediction targets and policy interface. It improves average success rates in both evaluated WAM settings, with matched ablations and controlled diagnostics supporting its structural choices. The theory specifies how the operator family can represent projected dynamics, while the experiments assess the resulting learned transition and policy performance.

AI USE STATEMENT

Generative AI tools were used solely for language editing and polishing of the manuscript, including improvements to grammar, wording, clarity, and readability. They were not used to develop the theoretical framework, formulate mathematical claims, design the methodology or experiments, implement methods, analyze data, interpret results, or generate scientific conclusions. All AI-assisted edits were reviewed and verified by the authors. The authors take full responsibility for the final content of the manuscript.

REPRODUCIBILITY STATEMENT

Appendix A provides the projection construction, representation proof, and evolution equations. Appendices B–C specify evaluation protocols, initialization, architecture, losses, and optimization. Appendix D reports rollout-seed and task-level results and defines the matched ablations; Appendix E details the controlled-system diagnostics. Policy results use one training run per variant; repeated rollout evaluations characterize evaluation variability.

REFERENCES

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert 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. In Joseph Lim, Shuran Song, and Hae-Won Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 27–30 Sep 2025.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. UniVLA: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025.

Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, et al. WorldVLA: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

Hongyi Chen, Abulikemu Abuduweili, Aviral Agrawal, Yunhai Han, Harish Ravichandar, Changliu Liu, and Jeffrey Ichnowski. KOROL: Learning visualizable object feature with Koopman operator rollout for manipulation. *arXiv preprint arXiv:2407.00548*, 2024.

Jialei Chen, Kai Wang, Kang Chen, Shuaihang Chen, Feng Gao, Wenhao Tang, Zhiyuan Li, Weilin Liu, Zhuyu Yao, Boxun Li, et al. LaWAM: Latent world action models for efficient dynamics-aware robot policies. *arXiv preprint arXiv:2606.15768*, 2026.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. LIBERO-Plus: A progressive robustness benchmark for visual-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 38574–38583, 2026.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Yuhang Huang, Jiazhao Zhang, Shilong Zou, Xinwang Liu, Ruizhen Hu, and Kai Xu. LaDi-WM: A latent diffusion-based world model for predictive manipulation. *arXiv preprint arXiv:2505.11528*, 2025.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, et al. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

Stefan Klus, Feliks Nüske, Sebastian Peitz, Jan-Hendrik Niemann, Cecilia Clementi, and Christof Schütte. Data-driven approximation of the Koopman generator: Model reduction, system identification, and control. *Physica D: Nonlinear Phenomena*, 406:132416, 2020.

Hemant Kumawat, Biswadeep Chakraborty, and Saibal Mukhopadhyay. RoboKoop: Efficient control conditioned representations from visual input in robotics using Koopman operator. *arXiv preprint arXiv:2409.03107*, 2024.

Jiaqi Li, Xinglong Zhang, Haibin Xie, Yixing Lan, Wei Pan, and Xin Xu. Koopman dreamer: Spectrally constrained latent dynamics for stable world-model imagination. *arXiv preprint arXiv:2607.19719*, 2026a.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026b.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1):4950, 2018.

Pietro Novelli, Marco Pratticò, Massimiliano Pontil, and Carlo Ciliberto. Operator world models for reinforcement learning. *Advances in Neural Information Processing Systems*, 37:111432–111463, 2024.

Samuel Otto, Sebastian Peitz, and Clarence Rowley. Learning bilinear models of actuated Koopman generators from partially observed trajectories. *SIAM Journal on Applied Dynamical Systems*, 23 (1):885–923, 2024.

Sebastian Peitz, Samuel E Otto, and Clarence W Rowley. Data-driven model predictive control using interpolated Koopman generators. *SIAM Journal on Applied Dynamical Systems*, 19(3): 2162–2193, 2020.

Joshua L Proctor, Steven L Brunton, and J Nathan Kutz. Generalizing Koopman theory to allow for inputs and control. *SIAM Journal on Applied Dynamical Systems*, 17(1):909–930, 2018.

Mark W Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot modeling and control*. John Wiley & Sons, second edition, 2020.

Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. VLA-JEPA: Enhancing vision-language-action model with latent world model. In *European Conference on Computer Vision*, pp. 478–497. Springer, 2026.

Yang Tian, Sizhe Yang, Jia Zeng, Ping Wang, Dahua Lin, Hao Dong, and Jiangmiao Pang. Predictive inverse dynamics models are scalable learners for robotic manipulation. In *International Conference on Learning Representations*, volume 2025, pp. 92033–92052, 2025.

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. *Advances in Neural Information Processing Systems*, 28, 2015.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. CoT-VLA: Visual chain-of-thought reasoning for vision-language-action models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1702–1713. IEEE, 2025.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

Fangqi Zhu, Hongtao Wu, Song Guo, Yuxiao Liu, Chilam Cheang, and Tao Kong. IRASim: A fine-grained world model for robot manipulation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9834–9844. IEEE, 2025.

APPENDIX A CONTROLLED OBSERVABLE EVOLUTION AND LEON

This section gives the finite-projection construction used in the main paper and specifies the LEON evolution step.

A.1 FROM MECHANICAL DYNAMICS TO SHARED GENERATORS

For a smooth control-affine system, $\dot{s} = f_0(s) + \sum_{j=1}^{m} u_j f_j(s)$, the generator acting on a differentiable observable g satisfies

$$\mathcal{L}_u g = Dg\, f_0 + \sum_{j=1}^{m} u_j Dg\, f_j = \mathcal{L}_0 g + \sum_{j=1}^{m} u_j \mathcal{L}_j g. \tag{A.1}$$

Each generator is linear in g, and its dependence on physical control is affine (Proctor et al., 2018; Peitz et al., 2020; Otto et al., 2024). This statement applies within a smooth dynamical regime; finite observable closure is a separate condition.

A.2 FINITE PROJECTION AND OPERATOR VARIATION

Let $V_\Phi = \operatorname{span}\{\phi_1, \ldots, \phi_d\} \subset L^2(\mu)$, where μ is a probability measure on the state region of interest. Assume that the observables are linearly independent in $L^2(\mu)$ and that each $\mathcal{L}_j \phi_i$ belongs to $L^2(\mu)$. For $\Phi = (\phi_1, \ldots, \phi_d)^\top$, define

$$K_\Phi = \mathbb{E}_\mu[\Phi\Phi^\top], \qquad H_j = \mathbb{E}_\mu[(\mathcal{L}_j\Phi)\Phi^\top], \qquad G_j = H_j K_\Phi^{-1}. \tag{A.2}$$

The componentwise orthogonal projection P_Φ then satisfies $P_\Phi \mathcal{L}_j \Phi = G_j \Phi$. With $r_j = (I - P_\Phi)\mathcal{L}_j\Phi$, the same projection for all generator components gives

$$\begin{aligned} \dot{h} &= G(u)h + r(s,u), \\ G(u) &= G_0 + \sum_{j=1}^{m} u_j G_j, \qquad r(s,u) = r_0(s) + \sum_{j=1}^{m} u_j r_j(s). \end{aligned} \tag{A.3}$$

Here $\mathbb{E}_\mu[r_j \Phi^\top] = 0$; the residual vanishes if V_Φ is invariant under every generator component (Klus et al., 2020). If $r_* = \dim \operatorname{span}\{G_1, \ldots, G_m\}$, a basis $\{\widetilde{G}_k\}_{k=1}^{r_*}$ admits

$$G(u) = G_0 + \sum_{k=1}^{r_*} (Tu)_k \widetilde{G}_k, \qquad r_* \le m. \tag{A.4}$$

This identifies a shared matrix subspace without imposing a rank constraint on its basis matrices.

A.2.1 Factorized Operator Representation and Approximation

Equation (A.4) characterizes the projected dynamics by a generally unrestricted matrix family. LEON realizes this family with $D + U\,\mathrm{diag}(\xi)V^\top$, using shared rank-one factors and bounded coefficients. The following result connects these two representations and quantifies the effect of the factor budget. Its proof applies the classical truncated-SVD approximation bound to the generator components.

Proposition A.1 (Capacity of the factorized operator family)**.** *Let $G(u) = G_0 + \sum_{j=1}^m u_j G_j$, with $G_j \in \mathbb{R}^{d\times d}$, on the bounded control set*

$$\mathcal{U} = \{u \in \mathbb{R}^m : |u_j| \le M_j,\ j = 1, \dots, m\} \tag{A.5}$$

where $M_j > 0$; coordinates fixed at zero are omitted. Fix a diagonal base $D \in \mathbb{R}^{d\times d}$ allowed by the chosen parameterization, and write

$$B_0 = G_0 - D, \qquad B_j = G_j, \quad j = 1, \dots, m. \tag{A.6}$$

For integers $k_j \in \{0, \dots, d\}$ satisfying $\sum_{j=0}^m k_j \le r_A$, let $\sigma_p(B)$ denote the p-th singular value of B, with $\sigma_{d+1}(B) = 0$. Then there exist shared factors $U, V \in \mathbb{R}^{d\times r_A}$ and an affine coefficient map $\xi_\star : \mathcal{U} \to (-1, 1)^{r_A}$ such that

$$\sup_{u\in\mathcal{U}} \left\| G(u) - \left[D + U\,\mathrm{diag}(\xi_\star(u))V^\top\right] \right\|_2 \le \sigma_{k_0+1}(B_0) + \sum_{j=1}^m M_j \sigma_{k_j+1}(B_j). \tag{A.7}$$

In particular, exact representation for every $u \in \mathcal{U}$ is possible if

$$r_A \ge \mathrm{rank}(G_0 - D) + \sum_{j=1}^m \mathrm{rank}(G_j). \tag{A.8}$$

Proof. Write the rank-k_j truncated SVD as $B_j^{(k_j)} = \sum_{a=1}^{k_j} \sigma_{j,a} p_{j,a} q_{j,a}^\top$, where $p_{j,a}$ and $q_{j,a}$ are unit left and right singular vectors. Then $\|B_j - B_j^{(k_j)}\|_2 = \sigma_{k_j+1}(B_j)$, including $k_j = 0$ and $k_j = d$. Index the first $\sum_j k_j$ factor columns by pairs (j, a) and set

$$\begin{aligned} U_{0,a} &= 2\sigma_{0,a} p_{0,a}, \qquad V_{0,a} = q_{0,a}, \quad \xi_{0,a}(u) = \tfrac{1}{2}, \\ U_{j,a} &= 2M_j \sigma_{j,a} p_{j,a}, \quad V_{j,a} = q_{j,a}, \quad \xi_{j,a}(u) = \tfrac{u_j}{2M_j} \quad (j \ge 1). \end{aligned} \tag{A.9}$$

Here $U_{j,a}$ and $V_{j,a}$ denote columns, not matrix entries. Set unused columns and coefficients to zero. The factors are shared across u, the coefficient map is affine, and $\|\xi_\star(u)\|_\infty \le 1/2 < 1$. Direct multiplication gives

$$D + U\,\mathrm{diag}(\xi_\star(u))V^\top = D + B_0^{(k_0)} + \sum_{j=1}^m u_j B_j^{(k_j)}. \tag{A.10}$$

Therefore, by the triangle inequality and the bounds on u,

$$\begin{aligned} &\left\| G(u) - \left[D + U\,\mathrm{diag}(\xi_\star(u))V^\top\right] \right\|_2 \\ &\quad \le \|B_0 - B_0^{(k_0)}\|_2 + \sum_{j=1}^m |u_j|\, \|B_j - B_j^{(k_j)}\|_2 \\ &\quad \le \sigma_{k_0+1}(B_0) + \sum_{j=1}^m M_j \sigma_{k_j+1}(B_j), \end{aligned} \tag{A.11}$$

Taking the supremum proves Eq. (A.7). If Eq. (A.8) holds, choose $k_j = \mathrm{rank}(B_j)$; all singular-value tails vanish. □

Implications for LEON. The bound quantifies operator-family capacity through the singular-value tails remaining after allocating r_A factors. It is sufficient rather than necessary: sharing factors across different B_j can improve on their separate truncated decompositions. The base D must respect the implementation: it is unrestricted diagonal in VLA-JEPA, while LaWAM uses

$D = \mathrm{diag}(0.1 \tanh \widetilde{d})$, with diagonal entries in $(-0.1, 0.1)$ (Appendix C). Learning D allows adaptation within the corresponding admissible set.

The factor budget r_A differs from the intrinsic matrix-family dimension r_* in Eq. (A.4). The latter counts independent full-matrix directions; one such direction can require multiple rank-one factors. The bounded construction is compatible with the range of LEON's final $\tanh$, but does not guarantee that its learned coefficient head realizes $\xi_\star$ from the available visual and action context. Proposition A.1 establishes representational capacity for fixed projected generators. It leaves the projection residual, coefficient-learning error, optimization, and rollout accuracy to separate analysis or empirical evaluation. The complementary update is learned jointly and is not constrained to equal the projection residual.

A.3 From Physical Observable Evolution to the Learned Transition

If the observable derivative along a trajectory is Lipschitz in time, integration over Δt yields

$$h(t + \Delta t) = h(t) + \Delta t \, [G(u(t))h(t) + r(s(t), u(t))] + O(\Delta t^2). \tag{A.12}$$

LEON uses the operator–complementary organization of this first-order update in learned visual coordinates. Its latent coefficients do not identify physical control, and $b(h, \xi)$ is learned jointly with the operator rather than constrained to equal the projection residual. The update scale η and block index ℓ describe network computation, not calibrated physical integration time. The low-rank factorization below is a capacity choice rather than a consequence of Koopman theory.

A.4 Illustrative Finite-Observable Examples

For a controlled spring, $\dot{q} = v$ and $m_s \dot{v} = -k_s q - c_s v + u$. With $\Phi = (1, q, v)^\top$, the evolution closes exactly:

$$\dot{h} = (G_0 + uG_1)h, \qquad G_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -k_s/m_s & -c_s/m_s \end{bmatrix}, \quad G_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1/m_s & 0 & 0 \end{bmatrix}. \tag{A.13}$$

For a pendulum, $\dot{\theta} = \omega$ and $\dot{\omega} = -\omega_p^2 \sin\theta - \beta_p \omega + u/I_p$. On an angle chart with $\Phi = (1, \theta, \omega)^\top$, the least-squares approximation $\sin\theta \simeq a^\top \Phi$ uses $a^\top = \mathbb{E}_\mu[\sin\theta \, \Phi^\top] K_\Phi^{-1}$ and leaves residual

$$r_0(s) = (0, 0, -\omega_p^2 [\sin\theta - a^\top \Phi(s)])^\top, \qquad r_1 = 0. \tag{A.14}$$

These examples distinguish exact closure from unresolved nonlinear change. The learned diagnostic predictors are specified in Appendix E.

A.5 Common LEON Evolution Computation

For token $x_i^{(\ell)} \in \mathbb{R}^{d_x}$, each of the L independently parameterized steps computes

$$\begin{aligned} h_i^{(\ell)} &= W_{\varphi,\ell} x_i^{(\ell)} + g_{\varphi,\ell}(\mathrm{LN}(x_i^{(\ell)})), \\ \rho_i^{(\ell)} &= g_{\mathrm{fuse},\ell}(h_i^{(\ell)}, Z), \qquad \xi_i^{(\ell)} = \tanh(f_{\xi,\ell}(\rho_i^{(\ell)})). \end{aligned} \tag{A.15}$$

The complementary branch first embeds the latent control to observable width and adds it to the current observable:

$$s_{b,i}^{(\ell)} = h_i^{(\ell)} + g_{p,\ell}(\xi_i^{(\ell)}), \qquad b_\ell(h_i^{(\ell)}, \xi_i^{(\ell)}) = g_{b,\ell}(s_{b,i}^{(\ell)}). \tag{A.16}$$

The two-layer embedding MLP is

$$g_{p,\ell}(\xi) = W_{p,2,\ell} \, \mathrm{GELU}\left(W_{p,1,\ell} \, \mathrm{LN}(\xi) + c_{p,1,\ell}\right) + c_{p,2,\ell}, \tag{A.17}$$

where $W_{p,1,\ell} \in \mathbb{R}^{256 \times 96}$ and $W_{p,2,\ell} \in \mathbb{R}^{768 \times 256}$. The normalized two-layer MLP $g_{b,\ell}$ has input, hidden, and output dimensions $(768, 768, 768)$ and a zero-initialized output projection. Its hidden activation is Tanh for VLA-JEPA and SiLU for LaWAM. Thus b receives both h and ξ through the summed input in Eq. (A.16).

The operator and complete evolution step are

$$\begin{aligned} A_\ell(\xi) &= D_\ell + U_\ell \,\mathrm{diag}(\xi) V_\ell^\top, \\ \widetilde{h}_i^{(\ell)} &= h_i^{(\ell)} + \eta\big[A_\ell(\xi_i^{(\ell)}) h_i^{(\ell)} + b_\ell(h_i^{(\ell)}, \xi_i^{(\ell)})\big], \\ x_i^{(\ell+1)} &= W_{\psi,\ell}\,\mathrm{LN}(\widetilde{h}_i^{(\ell)}) + c_{\psi,\ell}. \end{aligned} \tag{A.18}$$

Here D_ℓ is diagonal and $U_\ell, V_\ell \in \mathbb{R}^{768\times 96}$. Both WAM instantiations use this composition; their conditioning maps and step settings are given in Appendix C. The retained WAM output projection converts the final tokens to the future-feature prediction.

A.6 Computation and Parameterization

The operator is applied as $d_\ell \odot h + U_\ell[\xi \odot (V_\ell^\top h)]$, using $O(d_h r_A)$ operations per token and $d_h + 2d_h r_A$ parameters per step. Its conditional variation satisfies $\mathrm{rank}(A_\ell(\xi) - D_\ell) \le \min(d_h, r_A)$. Costs of observable adaptation, conditioning, complementary updates, and readout are additional. The complete evolution-path count used for the Residual-MLP comparison appears in Appendix D.2.

Appendix B Benchmark Data and Evaluation Protocols

B.1 Benchmark Data and Evaluation Conditions

LIBERO combines the Spatial, Object, Goal, and LIBERO-10 suites, with ten tasks and 50 demonstrations per task (Liu et al., 2023). LEON is trained jointly on the 40-task pool. LIBERO-Plus evaluates these checkpoints without adaptation on seven perturbation families: Camera, Robot, Language, Light, Background, Noise, and Layout (Fei et al., 2026). RoboTwin 2.0 uses the 50-task mixture of 2,500 clean and 25,000 randomized demonstrations adopted by LaWAM (Chen et al., 2025; 2026).

Table B.1: Policy-evaluation settings. Each rollout seed evaluates the same trained checkpoint.

Benchmark	Evaluation	Aggregation
LIBERO	50 trials per task; seeds $\{0, 1, 2\}$	Mean over 40 tasks and three rollout seeds
LIBERO-Plus	One rollout per released case; 10,030 cases	Unweighted mean over seven perturbation families
RoboTwin 2.0	100 trials per task and condition	Mean over 50 tasks, then over Clean and Randomized

B.2 Evaluation Metrics and Aggregation

We report task success rate (SR) using the benchmark success criteria. LIBERO suite averages weight their ten tasks equally; the overall average weights all 40 tasks equally. The seed-wise means and sample standard deviations in Appendix D.1 describe rollout variability for fixed trained weights. LIBERO-Plus averages the seven family-level SRs rather than pooling all cases. RoboTwin reports Clean and Randomized separately and uses their arithmetic mean for the overall score. Unless stated otherwise, robot results use one training run per model or variant.

B.3 Comparison Scope and Checkpoint Selection

The matched VLA-JEPA transition study shares data, non-transition initialization, task objectives, training schedule, and evaluation across variants. The transition modules are newly initialized. For the dagger-marked benchmark rows, we evaluate the officially released VLA-JEPA and LaWAM weights using their corresponding official environments and evaluation configurations. The RoboTwin LaWAM row follows the published reference. These benchmark references are distinct from matched retraining controls.

LEON uses the final checkpoint at the update budget in Appendix C.4. The same LIBERO checkpoints are evaluated on LIBERO-Plus. The four-task RoboTwin intervention study uses a separate 30,000-update checkpoint and changes branch outputs only at evaluation.

APPENDIX C WAM-SPECIFIC TRAINING PROTOCOLS AND LEON INSTANTIATIONS

We instantiate LEON under the prediction-supervised VLA-JEPA formulation and the future-conditioned LaWAM formulation. Their data preprocessing, feature targets, action representation, and policy interfaces follow the respective WAM protocols (Sun et al., 2026; Chen et al., 2026).

C.1 SHARED LEON INITIALIZATION

Retained model components are initialized from the officially released checkpoint of the corresponding WAM; LEON modules are newly initialized. The target visual encoder remains frozen in both instantiations. For each step, $W_{\varphi,\ell} = Q^\top$ and $W_{\psi,\ell} = Q$, where $Q \in \mathbb{R}^{1024\times 768}$ has orthonormal columns. The same initial Q is obtained by reduced QR decomposition of a Gaussian matrix with seed 1729 and used across steps; the maps are then optimized independently. Factors U_ℓ, V_ℓ use Gaussian initialization with standard deviation 0.01. The diagonal increment, coefficient-head output, complementary output, and nonlinear observable residual start at zero. The initial observable increment is therefore zero; the observable map and normalized readout remain active.

C.2 PREDICTION-SUPERVISED POLICY LEARNING: VLA-JEPA

The current observation and instruction produce latent-action representations Z and policy representations. LEON uses the encoded source features and Z to predict the retained future-feature targets. Prediction gradients pass through LEON and the latent-action pathway to the shared VLM. Action generation uses the policy representations and robot state; observed or predicted future features are not inputs to the deployed action head. Feature preprocessing and target construction follow VLA-JEPA.

For each prediction pair, Z contains eight width-2048 action tokens. The retained action projection has dimensions $(2048, 1024)$; each LEON step applies its own affine projection with dimensions $(1024, 768)$ and LayerNorm to obtain $\bar{Z}_\ell$. The conditioning map combines a local attention summary with a global summary:

$$\begin{aligned} c^{\mathrm{loc}}_{i,\ell} &= \mathrm{MHA}_\ell(\mathrm{LN}(h^{(\ell)}_i), \bar{Z}_\ell, \bar{Z}_\ell), \\ c^{\mathrm{glob}}_\ell &= g_{\mathrm{glob},\ell}(\mathrm{Mean}(\bar{Z}_\ell)), \qquad \rho^{(\ell)}_i = g_{\mathrm{merge},\ell}([c^{\mathrm{loc}}_{i,\ell}; c^{\mathrm{glob}}_\ell]). \end{aligned} \tag{C.1}$$

Each observable attends only to the action tokens assigned to its prediction pair. LEON applies no direct self-attention between visual tokens; the retained encoder and VLM provide contextualized inputs. The transition uses eight evolution steps with $\eta = 0.35$. In the implementation's propagation parameterization, $D_\ell = \mathrm{diag}(a_\ell - \mathbf{1})$, so $a_\ell = \mathbf{1}$ initializes the diagonal increment at zero. All other operations follow Appendix A.5.

The task objective retains VLA-JEPA's action and feature-prediction terms,

$$\mathcal{L}_{\mathrm{task}} = \mathcal{L}_{\mathrm{act}} + \beta \mathcal{L}_{\mathrm{WM}}. \tag{C.2}$$

The target feature encoder is frozen, and target features receive no gradient. Prediction targets and loss definitions follow the official VLA-JEPA implementation. The prediction pathway is used during training only. We set $\beta = 0.1$ in Eq. (C.2), and $\mathcal{L}_{\mathrm{task}}$ constitutes the complete training loss. Optimization uses AdamW with weight decay 10^{-8}. The peak learning rates are 5×10^{-7} for the VLM, 2×10^{-5} for the action head, and 10^{-5} for the remaining trainable parameters, including the LEON transition. The learning rates increase linearly during the first $2{,}000$ optimizer updates and then follow cosine decay. The scheduler uses a base learning rate of 10^{-5} and `min_lr` set to 5×10^{-7}.

C.3 FUTURE-CONDITIONED ACTION GENERATION: LAWAM

The current observation and instruction produce an aggregated latent action Z. LEON combines Z with current-state features to predict a future representation that conditions the action expert together with current features and VLM context. During training, a teacher inferred from the observed current–future pair supervises the latent action. At deployment, the policy supplies Z and LEON supplies the predicted future; no observed future is required. Feature preprocessing and prediction targets follow LaWAM.

LaWAM supplies one width-32 latent action, and the retained projection P_Z has dimensions $(32, 1024)$. It uses a global action summary and observable-conditioned fusion:

$$c_\ell^{\mathrm{glob}} = g_{\mathrm{glob},\ell}(\mathrm{LN}(P_Z Z)), \qquad \rho_i^{(\ell)} = g_{\mathrm{merge},\ell}([h_i^{(\ell)}; c_\ell^{\mathrm{glob}}]). \tag{C.3}$$

LEON applies twelve steps with $\eta = 0.1$ and $D_\ell = \mathrm{diag}(0.1 \tanh(\widetilde{d}_\ell))$. The coefficient head, complementary construction, observable update, and readout follow Appendix A.5.

The retained policy objective is

$$\mathcal{L}_{\mathrm{task}} = \mathcal{L}_{\mathrm{act}} + 0.1\mathcal{L}_{\mathrm{WM}} + 0.1\mathcal{L}_{\mathrm{distill}}. \tag{C.4}$$

The predicted-future input to the action expert is detached for the action loss. The world-model loss updates LEON and the latent-action pathway, while distillation aligns the policy latent action with the teacher. Loss definitions and target handling follow LaWAM. Optimization uses AdamW with weight decay 10^{-8}. The peak learning rate is 3×10^{-4} for the LEON world-model transition and 10^{-4} for the VLM, action head, and remaining trainable parameters. Linear warmup lasts for $1{,}500$ optimizer updates on LIBERO and $2{,}000$ updates on RoboTwin, followed by cosine decay. For both benchmarks, the scheduler uses a base learning rate of 10^{-4} and `min_lr` set to 5×10^{-7}.

C.4 Architecture and Training Configuration

Table C.1 records LEON-specific choices. An MLP dimension tuple lists input, hidden, and output widths, normalized MLPs apply LayerNorm at their input, and all MLP affine layers include biases. The activation listed for each MLP is its hidden activation. Both coefficient heads additionally apply the final $\tanh$ in Eq. (A.15). The control embedding g_p uses GELU in both instantiations.

Table C.1: LEON architecture. The complementary MLP receives $h + g_p(\xi)$ in both settings.

Component	VLA-JEPA + LEON	LaWAM + LEON
Retained feature-input projection	$(2048, 1024)$	$(768, 1024)$
Retained feature-output projection	$(1024, 2048)$	$(1024, 768)$
Predictor / observable widths	$1024/768$	$1024/768$
Evolution steps / scale	$8/0.35$	$12/0.1$
Operator order r_A	96	96
Observable residual MLP	$(1024, 1024, 768)$, GELU	$(1024, 1024, 768)$, SiLU
Global-summary MLP	$(768, 768, 768)$, GELU	$(1024, 768, 768)$, SiLU
Context fusion	Normalized affine $(1536, 768)$	Normalized MLP $(1536, 768, 768)$, SiLU
Local action attention	Eight heads	None
Coefficient-head MLP	$(768, 768, 96)$, Tanh	$(768, 768, 96)$, SiLU
Control-embedding MLP g_p	$(96, 256, 768)$, GELU	$(96, 256, 768)$, GELU
Complementary MLP g_b	$(768, 768, 768)$, Tanh	$(768, 768, 768)$, SiLU
Readout	LayerNorm and affine $(768, 1024)$	LayerNorm and affine $(768, 1024)$

The optimizer, weight decay, warmup updates, and cosine-scheduler parameters are specified in Sections C.2 and C.3. Table C.2 summarizes the training budgets and global batch sizes used for LEON.

Table C.2: LEON policy-training settings. Update counts refer to optimizer updates.

Setting	Updates	Global batch size
VLA-JEPA + LEON, LIBERO	40,000	256
LaWAM + LEON, LIBERO	40,000	256
LaWAM + LEON, RoboTwin	100,000	1,024
Four-task RoboTwin intervention model	30,000	1,024

APPENDIX D SUPPLEMENTARY MANIPULATION RESULTS AND TRANSITION ANALYSIS

D.1 POLICY EVALUATION RESULTS

D.1.1 LIBERO ROLLOUT-SEED RESULTS

Table D.1 gives the three rollout-seed evaluations underlying Table 1. Each method uses one trained checkpoint. The dagger-marked rows evaluate released WAM checkpoints; the LEON rows use the final checkpoints specified in Appendix C.4. VLA-JEPA + LEON and LaWAM + LEON achieve mean SRs of 99.0% and 98.9%, respectively.

Table D.1: LIBERO success rates (%) over evaluation seeds 0, 1, and 2. Mean and sample standard deviation (SD) are computed across the three evaluation seeds; each seed uses 50 trials per task. † denotes re-evaluation of a released WAM checkpoint.

Suite	Seed 0	Seed 1	Seed 2	Mean	SD
VLA-JEPA†					
Spatial	98.4	97.4	98.2	98.00	0.53
Object	99.8	99.6	99.8	99.73	0.12
Goal	98.6	98.0	98.6	98.40	0.35
LIBERO-10	95.8	94.8	95.6	95.40	0.53
All 40 tasks	98.15	97.45	98.05	97.88	0.38
VLA-JEPA+LEON					
Spatial	99.6	99.6	99.4	99.53	0.12
Object	99.6	99.8	99.8	99.73	0.12
Goal	99.0	99.2	98.8	99.00	0.20
LIBERO-10	98.0	97.8	97.6	97.80	0.20
All 40 tasks	99.05	99.10	98.90	99.02	0.10
LaWAM†					
Spatial	98.2	98.0	98.2	98.13	0.12
Object	98.8	99.0	99.0	98.93	0.12
Goal	98.8	96.0	99.6	98.13	1.89
LIBERO-10	97.0	98.2	96.2	97.13	1.01
All 40 tasks	98.20	97.80	98.25	98.08	0.25
LaWAM+LEON					
Spatial	99.0	99.6	99.0	99.20	0.35
Object	99.8	100.0	99.6	99.80	0.20
Goal	99.4	99.6	99.2	99.40	0.20
LIBERO-10	97.6	97.2	97.0	97.27	0.31
All 40 tasks	98.95	99.10	98.70	98.92	0.20

D.1.2 ROBOTWIN 50-TASK RESULTS

Table D.2 expands Table 2. LaWAM entries follow the published reference (Chen et al., 2026). LEON obtains 93.48% on Clean and 90.02% on Randomized, corresponding to differences of 0.84 and 0.22 percentage points. Under Randomized evaluation, 21 tasks improve, 22 decrease, and seven are unchanged. These are benchmark outcomes for the evaluated checkpoints.

Table D.2: All 50 RoboTwin tasks. Success rates are percentages and Δ is LaWAM+LEON minus the published LaWAM result in percentage points.

Task	Clean			Randomized		
	LaWAM +LEON	LaWAM	Δ	LaWAM +LEON	LaWAM	Δ
Adjust Bottle	100	100	0	100	100	0
Beat Block Hammer	98	90	+8	92	93	-1
Blocks Ranking RGB	98	97	+1	99	100	-1
Blocks Ranking Size	86	93	-7	82	89	-7
Click Alarmclock	100	100	0	100	100	0
Click Bell	100	100	0	100	100	0
Dump Bin Bigbin	95	97	-2	95	95	0

Table D.2 (continued)

Task	Clean			Randomized		
	LaWAM +LEON	LaWAM	Δ	LaWAM +LEON	LaWAM	Δ
Grab Roller	100	100	0	100	100	0
Handover Block	98	96	+2	76	87	-11
Handover Mic	97	93	+4	100	98	+2
Hanging Mug	55	51	+4	55	43	+12
Lift Pot	100	100	0	100	99	+1
Move Can Pot	96	98	-2	95	93	+2
Move Pillbottle Pad	99	97	+2	95	90	+5
Move Playingcard Away	100	100	0	99	100	-1
Move Stapler Pad	81	94	-13	71	87	-16
Open Laptop	99	100	-1	100	100	0
Open Microwave	73	41	+32	70	43	+27
Pick Diverse Bottles	94	91	+3	84	88	-4
Pick Dual Bottles	100	100	0	91	95	-4
Place A2B Left	97	98	-1	93	91	+2
Place A2B Right	96	89	+7	92	94	-2
Place Bread Basket	93	92	+1	91	85	+6
Place Bread Skillet	94	90	+4	90	83	+7
Place Burger Fries	97	93	+4	98	96	+2
Place Can Basket	88	92	-4	71	65	+6
Place Cans Plasticbox	100	100	0	99	95	+4
Place Container Plate	100	100	0	98	100	-2
Place Dual Shoes	96	98	-2	92	94	-2
Place Empty Cup	100	99	+1	99	100	-1
Place Fan	91	92	-1	91	93	-2
Place Mouse Pad	87	91	-4	80	84	-4
Place Object Basket	89	92	-3	92	90	+2
Place Object Scale	93	95	-2	89	88	+1
Place Object Stand	99	92	+7	91	93	-2
Place Phone Stand	93	93	0	92	94	-2
Place Shoe	97	100	-3	99	100	-1
Press Stapler	95	98	-3	99	97	+2
Put Bottles Dustbin	92	94	-2	86	92	-6
Put Object Cabinet	84	90	-6	85	82	+3
Rotate QRcode	95	94	+1	90	89	+1
Scan Object	94	96	-2	92	90	+2
Shake Bottle	100	100	0	99	100	-1
Shake Bottle Horizontally	100	100	0	100	100	0
Stack Blocks Three	87	90	-3	79	75	+4
Stack Blocks Two	100	100	0	98	97	+1
Stack Bowls Three	94	90	+4	81	80	+1
Stack Bowls Two	98	100	-2	98	99	-1
Stamp Seal	89	89	0	79	88	-9
Turn Switch	67	47	+20	54	56	-2
Mean	93.48	92.64	+0.84	90.02	89.80	+0.22

D.2 Transition Architecture Ablations

D.2.1 Matched Controls and Variant Definitions

All variants use the joint four-suite VLA-JEPA protocol, common non-transition initialization, and a 40,000-update budget. One training run is reported per variant; LIBERO-Plus evaluates the corresponding final checkpoints without adaptation. LEON-based variants retain eight evolution steps and $\eta = 0.35$ unless a component is explicitly removed.

Transformer. The twelve original predictor blocks are reinitialized, while compatible non-transition components retain the common pretrained initialization.

Residual MLP. This control retains observable adaptation, context fusion, and readout, but predicts the complete increment using

$$g(h_i, \rho_i) = W_2 \,\mathrm{GELU}\left(W_1 \,\mathrm{LN}([h_i; \rho_i]) + c_1\right) + c_2. \tag{D.1}$$

Here $\rho = g_{\mathrm{fuse}}(h, Z)$ is the fused conditioning representation. Its input, hidden, and output dimensions are $(1536, 960, 768)$. The output projection is zero-initialized. Including input LayerNorm and affine biases, this MLP has $2{,}216{,}640$ parameters per step. LEON's diagonal and low-rank factors, coefficient head, control embedding, and complementary MLP have

$$P_{\mathrm{LEON}} = 148{,}224 + 665{,}952 + 222{,}400 + 1{,}182{,}720 = 2{,}219{,}296 \tag{D.2}$$

parameters per step. The counts differ by $2{,}656$, or 0.120% of LEON's evolution path. Shared observable maps, context-fusion modules, and readouts are excluded from both counts.

Component variants. *w/o observable adaptation* removes the per-step observable map and learned readout, applying the structured update at predictor width 1024. The coefficient head has dimensions $(1024, 1024, 96)$, the control embedding $(96, 256, 1024)$, and the complementary MLP $(1024, 1024, 1024)$; an output LayerNorm is retained. *w/o operator modulation* replaces the operator coefficients with a learned layer-specific constant $\bar{\xi}_\ell = \tanh(\bar{c}_\ell)$, while $b(h, \xi)$ continues to use input-dependent ξ.

Operator only removes g_p and g_b and retains $A(\xi)h$. *Additive only* removes the complete operator contribution, including D, U, V, and retains the observable map, conditioning, coefficient head, control embedding, complementary MLP, and readout:

$$\widetilde{h} = h + \eta\, g_b(h + g_p(\xi)), \qquad x^+ = \psi(\widetilde{h}). \tag{D.3}$$

It preserves LEON's original complementary branch without reallocating the removed parameters. Residual MLP instead replaces the complete increment with Eq. (D.1). All these variants are trained separately.

Table D.3: Transition variants corresponding to Table 3. Width denotes the evolution coordinates.

Variant	Depth	Width	Increment or predictor
Transformer	12	1024	Action-conditioned predictor blocks
Residual MLP	8	768	$g(h, \rho)$; hidden width 960
w/o observable adaptation	8	1024	$A(\xi)x + b(x, \xi)$
w/o operator modulation	8	768	$\bar{A}h + b(h, \xi)$
Operator only	8	768	$A(\xi)h$
Additive only	8	768	$b(h, \xi)$
LEON	8	768	$A(\xi)h + b(h, \xi)$

D.2.2 LIBERO Results

Table D.4 reports all four suite scores. LEON reaches 99.0% mean SR, compared with 97.4% for Transformer and 97.7% for Residual MLP.

Table D.4: LIBERO transition ablations (SR, %). The average weights the four ten-task suites equally.

Variant	Spatial	Object	Goal	LIBERO-10	Avg.
Transformer	97.6	98.6	98.0	95.4	97.4
Residual MLP	98.4	99.4	97.8	95.0	97.7
w/o observable adaptation	98.4	99.4	98.6	96.0	98.1
w/o operator modulation	98.8	99.0	97.0	95.6	97.6
Operator only	98.0	98.0	98.2	96.8	97.8
Additive only	98.4	99.2	98.0	97.0	98.2
LEON	**99.5**	**99.7**	**99.0**	**97.8**	**99.0**

D.2.3 LIBERO-PLUS RESULTS

Table D.5 includes all seven perturbation families. LEON obtains an average of 80.6%, compared with 75.5% and 75.2% for Transformer and Residual MLP. Removing observable adaptation produces the largest decrease among the LEON component variants. Both single-path variants are below the complete model.

Table D.5: LIBERO-Plus transition ablations (SR, %). Avg. is the unweighted mean over seven perturbation families.

Variant	Cam.	Light	Backg.	Layout	Robot	Lang.	Noise	Macro Avg.
Transformer	53.3	94.4	93.5	82.1	60.7	78.5	66.0	75.5
Residual MLP	56.8	92.6	92.0	85.7	58.9	76.2	64.2	75.2
w/o observable adaptation	58.1	92.4	90.6	81.1	57.9	74.9	63.0	74.0
w/o operator modulation	63.6	96.2	95.5	86.4	61.2	79.2	66.7	78.4
Operator only	63.3	93.4	92.0	81.4	63.4	82.0	69.1	77.8
Additive only	63.9	94.0	96.6	82.4	59.8	77.3	65.0	77.0
LEON	66.2	98.9	98.0	87.7	63.0	81.5	68.6	80.6

D.3 POLICY-LEVEL TRANSITION-PATH INTERVENTIONS ON ROBOTWIN

A separate model is trained jointly on Hanging Mug, Pick Diverse Bottles, Lift Pot, and Dump Bin Bigbin. Evaluation uses one 30,000-update checkpoint, with 100 trials per task and condition. Let $M(\xi) = U\,\mathrm{diag}(\xi)V^\top$. We retain the diagonal term and mask the other branch outputs:

$$\widetilde{h} = h + \eta[Dh + m_{\mathrm{mod}} M(\xi)h + m_{\mathrm{add}} b(h, \xi)]. \tag{D.4}$$

The settings $(m_{\mathrm{mod}}, m_{\mathrm{add}}) = (0,0), (0,1), (1,1)$ produce the three rows of Table D.6. All rows retain the same readout and learned weights; the coefficient head and control embedding remain active whenever b is retained. Restoring b increases the Clean and Randomized averages by 7.25 and 11.25 points. Restoring modulation then raises Clean by one point and lowers Randomized by one point, leaving the two-condition mean at 83.50%. This fixed-checkpoint diagnostic measures path sensitivity, with a task-dependent effect of modulation.

Table D.6: Transition-path interventions on one four-task RoboTwin checkpoint, trained for 30,000 updates. Each condition uses 100 trials per task without retraining. Mean averages the four tasks.

	Hanging Mug		Pick Diverse Bottles		Lift Pot		Dump Bin Bigbin		Mean	
Retained increment	**Clean**	**Randomized**	**Clean**	**Randomized**	**Clean**	**Randomized**	**Clean**	**Randomized**	**Clean**	**Randomized**
Dh	32	32	92	71	95	97	89	86	77.00	71.50
$Dh + b$	52	61	95	86	98	98	92	86	84.25	82.75
$Dh + M(\xi)h + b$	53	55	96	81	99	100	93	91	85.25	81.75

APPENDIX E DYNAMICAL-SYSTEM DIAGNOSTICS

The spring, pendulum, and driven oscillator examine amplitude extrapolation, nonlinear state-regime extrapolation, and operator–additive path sensitivity, respectively. These predictors specialize the transition organization to physical states or compact observations; their dimensions and conditioning differ from the robot models. The spring and pendulum are autonomous, so their adaptive coefficients depend on the state. The driven oscillator additionally provides measured force and stiffness as conditioning variables.

E.1 COMMON PROTOCOL, COMPARISONS, AND METRICS

All systems use sampling interval $\Delta t = 0.05\,\mathrm{s}$ and recursively generated rollouts. Compared models share data splits, normalization, and prediction targets within each seed. The spring, pendulum, and driven oscillator use five, three, and ten independent seeds, respectively. Results report the mean and sample standard deviation across seeds, with 128 test trajectories per regime and seed.

For J trajectories, H predicted samples, and state dimension d_s, cumulative state RMSE is

$$R_s(H) = \left[\frac{1}{JHd_s} \sum_{j=1}^{J} \sum_{n=1}^{H} \|\widehat{s}_{j,n} - s_{j,n}\|_2^2 \right]^{1/2}. \tag{E.1}$$

The spring and driven oscillator use physical position and velocity. For the pendulum, define the wrapped angular error

$$e_{\theta,j,n} = \operatorname{atan2}\Big(\sin(\widehat{\theta}_{j,n} - \theta_{j,n}), \cos(\widehat{\theta}_{j,n} - \theta_{j,n})\Big), \qquad R_\theta(H) = \left[\frac{1}{JH} \sum_{j=1}^{J} \sum_{n=1}^{H} e_{\theta,j,n}^2 \right]^{1/2}. \tag{E.2}$$

Angular-velocity RMSE uses ordinary differences in the same aggregation. Metrics at time t include the first $H = t/\Delta t$ predicted samples beyond the measured context. Each rollout metric is computed within a seed before averaging across seeds. Teacher-forced one-step evaluation instead supplies measured context at each forecast time, so prediction errors are not fed back into subsequent inputs.

All neural predictors use AdamW and select the lowest-validation-error checkpoint after the stated training budget. Spring and pendulum training average squared errors over the normalized coordinates, sampled windows, and listed horizon endpoints. All horizons are produced recursively from the measured context; the driven oscillator supervises every step in its shorter training rollout. Transformer references use pre-normalized causal blocks with GELU feed-forward activation, zero dropout, and a final LayerNorm; their final token predicts the normalized observation or state increment, which is rescaled before residual addition. Predicted observations are appended to the rolling context.

Table E.1: Diagnostic data and training settings. Trajectory counts are per seed; horizons count predicted samples.

Setting	Spring	Pendulum	Driven oscillator
Train / validation / test trajectories	16/24/128	128/48/128	32/48/128
Training / test trajectory samples	520/520	180/190	129/105
Context length	8	8	4
Longest supervised rollout	60	80	8
Evaluation rollout	400	160	80
Training updates	1,400	1,600	1,400
Batch size	96	64	96
Base learning rate	8×10^{-4}	8×10^{-4}	2×10^{-3}
Independent seeds	5	3	10

E.2 Damped Spring: Exact Linear Closure and Amplitude Extrapolation

The unit-mass spring has $s = (q, v)$ and

$$\dot{s} = F_s s, \qquad F_s = \begin{bmatrix} 0 & 1 \\ -4 & -0.12 \end{bmatrix}, \qquad K_s = \exp(\Delta t F_s). \tag{E.3}$$

Trajectories use fourth-order Runge–Kutta integration. Training, validation, and ID tests sample q_0 uniformly in $[-1, 1]$ and v_0 in $[-1.5, 1.5]$. OOD tests independently sample their magnitudes in $[2.5, 4]$ with random signs. Training observations have independent Gaussian noise of standard deviation 0.015; validation and test states are clean. Coordinates are scaled by training standard deviations, with floors $(0.25, 0.35)$ and no mean subtraction.

Predictors. DMD fits $K_{\mathrm{DMD}} = S_+ S_-^\top (S_- S_-^\top + 10^{-5} I)^{-1}$ to adjacent training snapshots. The learned fixed operator is $K_0 = I + 0.20 \tanh(R)$ with four parameters and $R_{ij} \sim \mathcal{N}(0, 0.01^2)$. The adaptive predictor uses

$$\widehat{s}_{n+1} = \big[K_0 + U \operatorname{diag}(\xi(s_n)) V^\top\big] s_n, \qquad \xi(s) = 0.15 \tanh(C(s/\sigma)). \tag{E.4}$$

Here $U, V \in \mathbb{R}^{2\times 2}$ have independent Gaussian initialization with standard deviation 0.02, and C is a SiLU MLP with dimensions $(2, 32, 2)$ and zero-initialized output. This predictor has 174 parameters.

The Transformer uses two causal blocks, width 32, four heads, feed-forward width 64, and a SiLU increment head with dimensions $(32, 64, 2)$, for 19,746 parameters. The spring is an exact-closure diagnostic rather than a parameter-matched comparison. Its exact transition is fixed and linear; the learned results compare identification and extrapolation under the stated noisy-data protocol.

Training and evaluation. Training minimizes normalized-state prediction error at horizons $\{1, 10\}$ for the first 699 updates and $\{1, 10, 30, 60\}$ thereafter. Models receive the same sampled windows within each seed. Validation evaluates 60-step rollouts every 100 updates and scores the fixed horizons $\{1, 10, 30, 60\}$. Tests predict 400 samples beyond the eight-state context. The adaptive predictor obtains 20-s OOD RMSE 0.0212, compared with 0.5518 for DMD and 3.2006 for Transformer (Table E.2).

Table E.2: Spring cumulative state RMSE in physical coordinates. Entries are means ± sample standard deviations over five seeds, with 128 test trajectories per seed and split. Lower is better.

Predictor	0.5 s	3 s	10 s	20 s
ID				
DMD	0.0080 ± 0.0014	0.0396 ± 0.0060	0.0909 ± 0.0124	0.1108 ± 0.0138
Shared operator (SGD)	0.1373 ± 0.0025	0.1680 ± 0.0037	0.1453 ± 0.0042	0.1262 ± 0.0045
Transformer	0.0103 ± 0.0054	0.0217 ± 0.0124	0.0293 ± 0.0166	0.0332 ± 0.0173
Operator-structured	0.0007 ± 0.0002	0.0017 ± 0.0005	0.0038 ± 0.0010	0.0061 ± 0.0007
Amplitude OOD				
DMD	0.0376 ± 0.0058	0.1876 ± 0.0282	0.4531 ± 0.0631	0.5518 ± 0.0700
Shared operator (SGD)	0.6826 ± 0.0118	0.8231 ± 0.0143	0.7079 ± 0.0150	0.6107 ± 0.0175
Transformer	1.2364 ± 0.1058	4.5463 ± 0.2054	4.0635 ± 0.1807	3.2006 ± 0.1865
Operator-structured	0.0031 ± 0.0012	0.0079 ± 0.0033	0.0161 ± 0.0085	0.0212 ± 0.0116

Table E.3 reports scale-equivariance and energy diagnostics. Scale error, evaluated on validation probe states, is $\|\mathcal{F}(3s) - 3\mathcal{F}(s)\|_2 / \max(\|3\mathcal{F}(s)\|_2, 10^{-6})$. For $E(q, v) = 2q^2 + v^2/2$, energy MAE averages the absolute predicted-versus-true energy difference over test trajectories and forecast times; the balance residual averages $|E(\widehat{s}_{n+1}) - E(\widehat{s}_n) + 0.12\Delta t(\widehat{v}_n^2 + \widehat{v}_{n+1}^2)/2|$.

Table E.3: Spring physical diagnostics. Scale equivariance uses validation probe states with a threefold input rescaling; energy MAE and energy-balance residual use amplitude-OOD test trajectories with $E(q, v) = 2q^2 + v^2/2$. Lower is better.

Predictor	Scale-equiv. error	20-s energy MAE	Energy-balance residual
Transformer	6.20×10^{-2}	5.3546	0.0387
DMD	4.60×10^{-8}	3.0315	0.0223
Operator-structured	8.66×10^{-5}	0.0554	0.0018

E.3 Damped Pendulum: Nonlinear State-Regime Extrapolation

The pendulum follows $\dot{\theta} = \omega$, $\dot{\omega} = -9.81 \sin\theta - 0.12\omega$, with energy $E = \omega^2/2 + 9.81(1 - \cos\theta)$. Its observation is $x = (\sin\theta, \cos\theta - 1, \omega)$, and predictions decode angle using $\mathrm{atan2}(\widehat{x}_1, \widehat{x}_2 + 1)$. Fourth-order Runge–Kutta trajectories have no observation noise. Initial states are sampled uniformly in $[-\pi, \pi) \times [-4.5, 4.5)$ and accepted by initial energy: $[0.5, 12)$ for training, validation, and ID; $[12, 16)$ for Near OOD; $[16, 19)$ for Far OOD; and $[20.5, 28)$ for High OOD. Regimes are assigned by initial energy and remain fixed as damping reduces energy along a trajectory. The conservative separatrix has energy 19.62, placing Far OOD below and High OOD above it. Inputs use training means and standard deviations, with scale floors $(0.35, 0.20, 0.55)$.

Predictors. The structured models encode $\bar{x} = (x - \mu)/\sigma$ using $r^0 = P_{\text{in}}\bar{x} + a_{\text{in}} \in \mathbb{R}^{24}$ and apply two blocks with 20-dimensional observables:

$$\begin{aligned} h^\ell &= Q_\ell r^\ell + f_\ell(\mathrm{LN}(r^\ell)), \\ \widetilde{h}^\ell &= h^\ell + \eta_\ell A_\ell(\xi(h^\ell))h^\ell, \qquad r^{\ell+1} = r^\ell + W_\ell(\widetilde{h}^\ell - h^\ell), \\ \widehat{x}_{n+1} &= x_n + \sigma \odot P_{\text{out}}(r^2 - r^0). \end{aligned} \tag{E.5}$$

The SiLU MLP f_ℓ has dimensions $(24, 64, 20)$ and a zero-initialized output. The input and bias-free output projections have dimensions $(3, 24)$ and $(24, 3)$ and paired orthogonal initialization, with $a_{\text{in}} = 0$. Each predicted observation is re-encoded before the next rollout step. Each operator has diagonal $0.20\tanh(\widetilde{d}_\ell)$ and six factorized directions. The scale $\eta_\ell = 0.70\,\mathrm{sigmoid}(\widetilde{\eta}_\ell)$ starts at 0.20; the diagonal starts at zero. Observable maps and readouts use paired orthogonal initialization; the operator factors use Gaussian initialization with standard deviation 0.02.

The fixed-coefficient model uses $\xi_{0,\ell} = 0.25\tanh(B_\ell(\tanh(G_\ell(\mathbf{1}))))$, with MLP dimensions $(20, 32, 4)$ for G_ℓ and $(4, 32, 6)$ for B_ℓ, whose output is initialized at zero. The adaptive model adds

$$\xi(h^\ell) = \xi_{0,\ell} + 0.02\tanh\left(W_{\alpha,\ell}\tanh(W_{c,\ell}h^\ell + a_{c,\ell}) + a_{\alpha,\ell}\right), \tag{E.6}$$

with dimensions $(20, 4, 6)$ and zero-initialized output. The fixed and adaptive models have 10,830 and 11,058 parameters. The Transformer has two causal blocks, width 24, four heads, feed-forward width 56, and an affine observation-increment head, for 10,939 parameters.

Training and evaluation. Normalized-observation prediction uses horizons $\{1, 5, 10\}$ for updates 1–319, additionally 20 and 40 for updates 320–879, and additionally 80 thereafter. The adaptive coefficient residual uses learning rate 10^{-4}; other parameters follow Table E.1. Validation evaluates 96 fixed windows every 100 updates. Tests recursively predict 160 samples, or 8 s, beyond the eight-observation context. The adaptive predictor reduces 8-s OOD angle RMSE relative to Transformer

Table E.4: Pendulum rollout errors across initial-energy regimes. Angle errors use wrapped differences in radians; angular-velocity errors are in radians per second. Entries are means ± sample standard deviations over three seeds. Lower is better.

Predictor	θ, 1 s	θ, 4 s	θ, 8 s	ω, 8 s
ID				
Transformer	0.0253 ± 0.0025	0.0412 ± 0.0056	0.0619 ± 0.0114	0.1846 ± 0.0334
Shared operator	0.0305 ± 0.0015	0.0445 ± 0.0075	0.0627 ± 0.0117	0.1677 ± 0.0357
Operator-structured	0.0306 ± 0.0013	0.0438 ± 0.0079	0.0583 ± 0.0129	0.1678 ± 0.0566
Near OOD				
Transformer	0.2458 ± 0.1208	0.3782 ± 0.0617	0.4786 ± 0.0126	1.3142 ± 0.0455
Shared operator	0.2068 ± 0.0416	0.3084 ± 0.1120	0.3789 ± 0.0979	1.0219 ± 0.2660
Operator-structured	0.2044 ± 0.0426	0.3120 ± 0.0788	0.4036 ± 0.0417	1.1041 ± 0.1568
Far OOD				
Transformer	0.7080 ± 0.2955	1.0943 ± 0.1610	1.2347 ± 0.1009	7.3063 ± 7.1914
Shared operator	0.7299 ± 0.1745	0.8086 ± 0.0896	0.9363 ± 0.0849	2.3581 ± 0.3329
Operator-structured	0.7141 ± 0.1643	0.7925 ± 0.0518	0.9368 ± 0.1212	2.3883 ± 0.3944
High OOD				
Transformer	1.2331 ± 0.0799	1.7496 ± 0.0670	1.7054 ± 0.0149	10.0159 ± 10.4069
Shared operator	1.1936 ± 0.0634	1.6769 ± 0.0524	1.6112 ± 0.0229	4.0232 ± 0.0779
Operator-structured	1.1915 ± 0.0614	1.6850 ± 0.0390	1.6121 ± 0.0224	3.9984 ± 0.0752

by 15.7%, 24.1%, and 5.5% in Near, Far, and High OOD. The fixed-coefficient model has slightly lower error than the adaptive model in each regime. These results support the structured family, with no uniform advantage from adaptive coefficients.

Local and trajectory-level errors. Table E.5 evaluates local one-step prediction using measured eight-observation contexts. Its trajectory-level columns first compute wrapped-angle RMSE over each 8-s rollout, then take the median or 90th percentile over trajectories within a seed. The table

averages these summaries across the three seeds. They describe the distribution of trajectory errors, whereas Table E.4 pools squared errors before taking the square root.

Table E.5: Local prediction and trajectory-level pendulum errors, averaged over three seeds. The median and 90th percentile summarize per-trajectory wrapped-angle RMSE over 8 s; they complement the pooled RMSE in Table E.4.

Regime	Predictor	1-step θ	1-step ω	Median	90th pct.
ID	Transformer	0.00442	0.02236	0.0400	0.0941
ID	Shared operator	0.00593	0.00912	0.0435	0.0896
ID	Operator-structured	0.00585	0.00909	0.0332	0.0723
Near OOD	Transformer	0.01568	0.08052	0.3896	0.6745
Near OOD	Shared operator	0.01993	0.01503	0.2323	0.6086
Near OOD	Operator-structured	0.02006	0.01514	0.2938	0.6437
Far OOD	Transformer	0.03751	0.16053	1.1130	1.6320
Far OOD	Shared operator	0.03673	0.02519	0.8392	1.3549
Far OOD	Operator-structured	0.03693	0.02531	0.8610	1.3338
High OOD	Transformer	0.08452	0.29289	1.7055	1.9383
High OOD	Shared operator	0.07317	0.04173	1.5797	1.9418
High OOD	Operator-structured	0.07340	0.04193	1.5820	1.9365

Parameter shifts. Table E.6 changes nominal pendulum length $L = 1$ and damping $c = 0.12$. The single-parameter tests use $L \in \{0.7, 1.3\}$ or $c \in \{0.06, 0.24\}$ while retaining the other nominal parameter; the joint tests use all four combinations. The reported error uses only the terminal prediction after 160 recursive steps:

$$R_{\theta,\mathrm{terminal}}(8\,\mathrm{s}) = \left[\frac{1}{J}\sum_{j=1}^{J} e_{\theta,j,160}^2\right]^{1/2}. \tag{E.7}$$

This terminal statistic tests prediction under changed system parameters and is distinct from the cumulative error over the nominal system's initial-energy regimes.

Table E.6: Pendulum parameter shifts: wrapped-angle RMSE (rad) at the terminal 8-s prediction. Single-parameter shifts retain the nominal value of the other parameter; the final four rows change length and damping jointly.

Shift	Transformer	Shared operator	Operator-structured
$c = 0.06$	0.3637	0.3792	0.3344
$c = 0.24$	0.3360	0.2834	0.3238
$L = 0.7$	0.6351	0.6066	0.5707
$L = 1.3$	1.0508	1.0356	1.0559
$(L, c) = (0.7, 0.06)$	0.9096	0.8638	0.8440
$(L, c) = (0.7, 0.24)$	0.3871	0.3763	0.3616
$(L, c) = (1.3, 0.06)$	1.1981	1.1776	1.2068
$(L, c) = (1.3, 0.24)$	0.7583	0.7446	0.7582

E.4 Driven Oscillator: Operator–Additive Path Diagnostics

The driven oscillator follows

$$\dot{q} = v, \qquad \dot{v} = -\kappa q - 0.12v + u(t). \tag{E.8}$$

Stiffness is uniform in $[1, 4]$ for training, validation, and ID, and $[4.5, 6]$ for OOD. Initial position and velocity are uniform in $[-1, 1]$ and $[-1.25, 1.25]$. Actions are uniform in $[-1.2, 1.2]$ and held for eight samples. Fourth-order Runge–Kutta trajectories have no observation noise. States use the training root-mean-square scale, and $\bar{\kappa} = (\kappa - 2.5)/1.5$.

For an action held constant over one sample, the physical system has the affine transition

$$s_{n+1} = K_\kappa s_n + G_\kappa u_n, \qquad K_\kappa = e^{\Delta t F_\kappa}, \quad G_\kappa = \int_0^{\Delta t} e^{\tau F_\kappa} B \, d\tau, \tag{E.9}$$

where $F_\kappa = \left[\begin{smallmatrix} 0 & 1 \\ -\kappa & -0.12 \end{smallmatrix}\right]$ and $B = (0,1)^\top$. This decomposition separates state propagation from action response and motivates the two-path diagnostic below. Simulated trajectories use the Runge–Kutta integration specified above.

Predictors and training. The complete predictor uses $h = s/\sigma$, where σ is the training root-mean-square scale plus 10^{-6}, and

$$\xi = C(\bar{\kappa}), \quad A(\xi) = D + U \operatorname{diag}(\xi) V^\top, \quad \widetilde{h} = h + \Delta t[A(\xi)h + b_u(u)]. \tag{E.10}$$

The coefficient and action-response networks are SiLU MLPs with dimensions $(1, 32, 2)$, and $U, V \in \mathbb{R}^{2\times 2}$. The diagonal starts at zero, and U, V use Gaussian initialization with standard deviation 0.25. The predicted physical state is $\sigma \odot \widetilde{h}$. This physical-state diagnostic routes stiffness to the operator and measured force to b_u; b_u is distinct from the robot complementary branch $b(h, \xi)$. The complete model has 270 parameters. Separately trained w/o-additive and w/o-modulation models have 140 and 142 parameters. The latter uses constant coefficients. Residual MLP receives $(h, \bar{\kappa}, u)$ and is a SiLU MLP with dimensions $(4, 40, 2)$, for 282 parameters. Transformer uses one causal block, width 4, two heads, feed-forward width 16, and four context states, for 298 parameters. It receives normalized states and actions at each context position and stiffness at the final position.

All models minimize mean squared normalized-state error over eight recursive steps. Training windows are paired across models within each seed. Validation uses the first eligible context of every validation trajectory, at update 1 and every 100 updates; tests predict 80 samples from four measured context states using the recorded action sequence. Table E.7 reports the separately trained comparisons. ID and OOD RMSE pool all 80 forecast steps and both physical-state coordinates within each seed. The OOD L1 and L2 columns report teacher-forced one-step prediction errors from measured histories; they assess local fit rather than recursively accumulated rollout error.

Table E.7: Driven-oscillator predictions over ten seeds. Rollout RMSE pools 80 prediction steps and both state coordinates before taking the square root; one-step metrics use the observed history. Values after ± are sample standard deviations.

Predictor	Parameters	ID RMSE	OOD RMSE	OOD L1	OOD L2
Full predictor	270	0.0067 ± 0.0046	0.1015 ± 0.0687	0.00200	0.00313
w/o additive	140	0.3300 ± 0.0150	0.3949 ± 0.0908	0.01828	0.03291
w/o modulation	142	0.3802 ± 0.0418	0.9153 ± 0.0621	0.02703	0.05276
Residual MLP	282	0.0337 ± 0.0094	0.2772 ± 0.0557	0.00630	0.01149
Transformer	298	0.1723 ± 0.0673	0.7741 ± 0.2337	0.03371	0.05709

Evaluation-time path removal. Figure 2(c) uses the same trained complete model and test trajectories for all conditions. Removing the operator sets $A(\xi)h = 0$; removing the additive path sets $b_u(u) = 0$. Both retain the identity state term. The OOD error ratios are 10.95 and 3.03, respectively. Each ratio divides the intervention mean RMSE by the full-model mean across the same ten seeds; it is not an average of seed-wise ratios. These interventions remove the branch computation together with its exclusive stiffness or force information; they measure trained-model sensitivity rather than independent branch contributions. Tables E.8 and E.9 give the aggregate and seed-level results.

Table E.8: Post-training path-removal sensitivity on the driven oscillator. All rows use the same ten trained full-model checkpoints and OOD trajectories; relative RMSE is normalized by the complete predictor.

Evaluation update	OOD RMSE	Relative RMSE
Full predictor	0.10152 ± 0.06874	1.00×
w/o operator (evaluation)	1.11205 ± 0.02746	10.95×
w/o additive (evaluation)	0.30720 ± 0.02747	3.03×

Table E.9: Driven-oscillator OOD rollout RMSE for each seed. The last two columns remove a path from the corresponding trained full predictor at evaluation time.

Seed	Full predictor	Transformer	Residual MLP	w/o operator	w/o additive
0	0.11638	1.14518	0.27533	1.15030	0.29581
1	0.13586	0.54645	0.29996	1.09734	0.32886
2	0.17116	0.80432	0.37401	1.11159	0.34949
3	0.06312	0.50759	0.23668	1.09699	0.27548
4	0.23315	0.92340	0.20397	1.10627	0.34889
5	0.01639	1.05316	0.29988	1.13021	0.29605
6	0.02112	0.77242	0.26142	1.13069	0.27589
7	0.04097	0.44209	0.21953	1.12378	0.28374
8	0.10956	0.87236	0.35460	1.12365	0.30838
9	0.10746	0.67360	0.24660	1.04963	0.30941

Condition swapping. OOD trajectories are paired by reversed stiffness order, keeping initial context and future actions fixed. For pair j and seed r, let $e_{\alpha\beta,j,r}$ be trajectory RMSE when conditioning on stiffness α and comparing with the future generated under stiffness β. We report

$$R_{\text{matched}} = \frac{1}{2SJ}\sum_{r,j}(e_{aa,j,r} + e_{bb,j,r}), \qquad R_{\text{cross}} = \frac{1}{2SJ}\sum_{r,j}(e_{ab,j,r} + e_{ba,j,r}), \tag{E.11}$$

with $S = 10$ and $J = 128$. The swap penalty is $R_{\text{cross}} - R_{\text{matched}}$. Retrieval measures whether a prediction is closer to its matched future than to the swapped alternative, averaging both condition assignments. These quantities average trajectory RMSE rather than pooling squared errors. Table E.10 shows sensitivity to the supplied stiffness; the constant-coefficient model is invariant to swapping.

Table E.10: Driven-oscillator condition sensitivity. Errors average per-trajectory RMSE symmetrically over the two condition assignments and then over ten seeds. Retrieval is reported as a percentage.

Predictor	Matched RMSE	Cross RMSE	Swap penalty	Retrieval
w/o modulation (trained)	0.84688	0.84688	0.00000	50.0
Transformer	0.68908	0.71303	0.02395	54.5
Residual MLP	0.22717	0.24707	0.01990	50.3
w/o additive (trained)	0.36281	0.43761	0.07480	64.4
Full predictor	0.08434	0.24944	0.16510	81.4