METADYN-JEPA: LEARNING TO PREDICT ACROSS PHYSICAL WORLDS WITH INTERACTION EVIDENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Physical interaction data often span systems with different underlying dynamics: the same action from a similar observed configuration can produce different outcomes. A short observation history may therefore be insufficient to predict the consequences of an action. We propose MetaDyn-JEPA (MetaDyn), a support-conditioned joint-embedding world model that uses additional interactions from the same physical system to inform prediction. MetaDyn encodes directed action–response transitions and aggregates them into a reusable world code that conditions multi-step latent prediction without test-time weight updates. Treating transitions as the units of evidence allows MetaDyn to aggregate support from disconnected fragments without relying on fragment-level sequence summaries. Across PokeWorld and MuJoCo, same-world support improves prediction over no-support baselines, while replacing it with wrong-world support degrades performance. MetaDyn remains stable under evidence-preserving repartitioning without fragment augmentation, and generally benefits from increasing same-world support toward the training length. In multi-dynamics Push-T, same-world support improves latent and physical prediction, and MetaDyn achieves higher reach-and-hold success than LeWM and LeWM + AdaJEPA. Overall, these results show that world-specific interaction evidence can provide a useful conditioning signal for prediction across varying dynamics.

1 INTRODUCTION

World models deployed across physical environments need to predict how actions affect systems with different underlying physical parameters. A shared model therefore needs to account for different dynamical behaviors, including cases where similar observations and identical actions lead to different outcomes. This raises a central question: how can a model predict the behavior of an unseen physical system from observed interactions?

For a given prediction task, a world model may receive only a short segment of observations and actions. This input may provide limited information about the system's dynamics. Additional interactions with the same physical system can reveal how it responds to actions. Although these interactions may start from different states and involve different actions, they share the same underlying dynamics. The challenge is to extract this shared dynamics information and use it to predict a new interaction.

We study this problem through support-conditioned prediction across physical worlds. A *world* denotes a physical system whose dynamics remain shared across its interactions. We call the additional observed interactions *support*, and the interaction to be predicted the *query*. During training, a shared model learns from support–query pairs drawn from multiple worlds. At test time, it uses support from an unseen world to condition query prediction, without updating its weights.

In practice, robot interaction datasets contain recordings of varying lengths (O'Neill et al., 2024; Fang et al., 2024). A support representation should therefore accommodate different amounts of interaction evidence. It should also avoid depending on how the same observed transitions are grouped into fragments. One approach is to encode each fragment as a sequence and then pool the resulting summaries. However, each summary can depend on where its fragment begins and ends, and pooling does not by itself remove this dependence. We instead use directed transitions as the units of evidence. Each transition pairs the observations before and after an action with the action

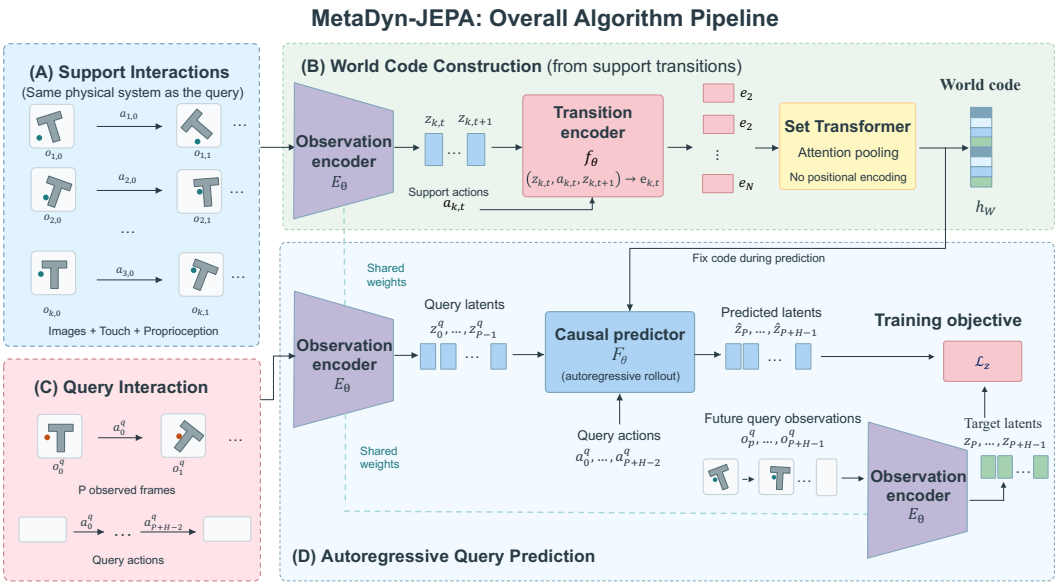

Figure 1: **MetaDyn overview.** Directed transitions from same-world support interactions are aggregated into a reusable world code that conditions autoregressive query prediction. The representation preserves local action–response relations without requiring temporal links between fragments. Fragment indices are omitted for clarity.

itself. Aggregating these transitions accommodates variable amounts of support while preserving local action–response relations, without relying on fragment-level summaries.

We propose MetaDyn-JEPA (MetaDyn), a support-conditioned world model based on a joint-embedding predictive architecture (JEPA) (Assran et al., 2025; Maes et al., 2026). It combines prediction in latent space with context inferred from interactions (Lee et al., 2020; Achterhold & Stueckler, 2021). As shown in Figure 1, a set encoder aggregates support transition representations into a fixed-dimensional *world code*, following permutation-invariant context aggregation (Garnelo et al., 2018; Lee et al., 2019). This code summarizes the support evidence and conditions the predictor at each step. Together with the query's initial observations and actions, it enables autoregressive prediction of future observations in latent space. The model learns to use support across training worlds without physical-parameter labels.
Our contributions are as follows:

- We propose MetaDyn, a support-conditioned JEPA world model that uses interactions from the same physical system to construct a reusable world code for multi-step latent prediction, without physical-parameter supervision or test-time weight updates.
- We represent support as a set of directed action–response transitions rather than fragment-level sequence summaries. This preserves local interaction evidence and makes prediction invariant to evidence-preserving repartitioning, without fragment augmentation.
- We systematically evaluate how the source, organization, and amount of support affect prediction in PokeWorld and MuJoCo, and extend the evaluation to multi-dynamics Push-T. Same-world support improves prediction, MetaDyn remains stable under repartitioning, and world-matched support also improves Push-T prediction and achieves higher reach-and-hold success.

2 Background and Related Work

2.1 Joint-embedding predictive world models

Joint-embedding predictive architectures (JEPAs) predict learned representations rather than reconstruct full observations. I-JEPA applies this principle to images, predicting target-region represen-

tations from a visible context region (Assran et al., 2023). V-JEPA extends feature prediction to video, learning representations of spatial and temporal structure (Bardes et al., 2024). Moving from video representation learning to interactive world modeling additionally requires conditioning predictions on actions. V-JEPA 2 takes this step through action-free video pretraining followed by action-conditioned training on robot interactions, yielding V-JEPA 2-AC for latent prediction and planning (Assran et al., 2025).

An alternative to staged pretraining is to learn the observation encoder and action-conditioned predictor jointly from interaction data. Such training must avoid representation collapse, since prediction matching alone admits constant embeddings. LeJEPA provides an explicit regularization principle: Sketched Isotropic Gaussian Regularization (SIGReg) constrains the embedding distribution toward an isotropic Gaussian (Balestriero & LeCun, 2025). LeWorldModel brings this principle to action-conditioned world modeling, combining next-embedding prediction with SIGReg for end-to-end learning from pixels (Maes et al., 2026).

Beyond how these representations are learned, Tan et al. (2026) investigate what physical information they retain. Their PokeWorld experiments distinguish physical parameters recoverable from observations from those decodable from learned representations. They show that input modalities affect what physical information a model can learn. This motivates a complementary question: how can a latent predictor use additional interactions from the same physical system when the query's short observations provide limited information about its dynamics?

2.2 Context-conditioned dynamics and amortized adaptation

Given additional interactions, context can be used through two distinct mechanisms: updating the model's parameters, or inferring a context that conditions a shared model. In the first case, adaptation involves optimizing model parameters using the observed evidence. In the second, a learned encoder maps new evidence to a context representation without test-time weight updates. This is a form of amortized adaptation.

A recent example of the optimization-based adaptation method applied in a latent world model is AdaJEPA (Wang et al., 2026). Within closed-loop planning, it executes an action chunk, uses the observed transition to update the model, and replans. Both routes use interaction evidence, but differ in whether that evidence changes the conditioning context or the model parameters.

On the other hand, conditional neural processes provide a foundation for the context-inference route (Garnelo et al., 2018). They encode observed input–output pairs and aggregate them as a set into a fixed-dimensional context, which conditions predictions at new inputs. PEARL infers a probabilistic task context from transitions using a permutation-invariant encoder and conditions an actor–critic on this context for off-policy meta-reinforcement learning (Rakelly et al., 2019). CaDM applies context conditioning to dynamics modeling: an encoder extracts dynamics-specific information from recent interactions, and a shared dynamics model uses it to predict the next state (Lee et al., 2020). Building on context-conditional dynamics, Achterhold & Stueckler (2021) study how to select interactions that reduce uncertainty about the system's context. Connecting context inference to latent world modeling, Nakheai et al. (2026) use multi-step latent temporal consistency to learn task representations that capture task-dependent dynamics for offline meta-reinforcement learning.

MetaDyn follows the context-inference route, building on transition-set context inference. Our focus is multi-step prediction in a multimodal JEPA world model, with explicit interventions on the source, partition, and amount of support evidence. Instead of first summarizing each support fragment as a sequence, it encodes directed transitions and aggregates them as a set. This preserves local action–response information while avoiding dependence on how the same transitions are grouped into fragments.

3 Method: MetaDyn

3.1 Problem formulation

A world model predicts how observations o_t evolve under actions a_t. In this paper, a world W means a physical system governed by persistent, unobserved physical parameters. Interactions within a

world share these parameters but may have different initial states and actions, while different worlds may have different underlying dynamics.

We call the interaction whose future is to be predicted the query, denoted by q. Given its first P observations $o^q_{0:P-1}$ and the actions $a^q_{0:P+H-2}$, the model predicts the observation embeddings for the next H steps, corresponding to $o^q_{P:P+H-1}$. These future observations provide training targets.

A short query history may provide insufficient evidence about the world's dynamics to predict the consequences of subsequent actions. We therefore also provide a support set $\mathcal{S}_W = \{\tau_k\}_{k=1}^K$ of observed interaction fragments from the same world W. Each fragment τ_k consists of a sequence of observations and intervening actions, and K denotes the number of fragments. We assume that the support and query are known to come from the same world. The support supplies additional action–response evidence about the dynamics governing the query.

MetaDyn summarizes $\mathcal{S}_W$ into a world code h_W and uses this code to condition query prediction. The code summarizes the available evidence about the shared dynamics. A shared model is trained on support–query pairs from multiple worlds and uses support from an unseen world at test time without updating its weights.

3.2 From interaction evidence to a world code

We construct the world code from directed action–response transitions in the support. Each support fragment τ_k contains T_k transitions $(o_{k,t}, a_{k,t}, o_{k,t+1})$, where k indexes fragments and t indexes time steps. Each transition records an observation, the action taken, and the resulting observation.

We then encode each observation as $z_{k,t} = E_\theta(o_{k,t})$, using an observation encoder shared by support and query. The encoder jointly represents images, tactile measurements, and proprioception, motivated by the role of input modalities in learning physical information (Tan et al., 2026). A transition encoder then combines the two observation embeddings with the intervening action:

$$e_{k,t} = f_\phi(z_{k,t}, a_{k,t}, z_{k,t+1}). \tag{1}$$

The ordered inputs preserve the direction of the local action–response relation. The same transition encoder is applied independently to every triple, without fragment identifiers or positional encodings among transitions.

Applying the transition encoder to all support transitions yields $M_W = \sum_{k=1}^K T_k$ encoded transition tokens $e_{k,t}$, which we collect into a multiset $\mathcal{E}_W$. The number of transition tokens can vary across support sets from different worlds because both the number and lengths of their fragments may differ. To obtain a fixed-dimensional world code from these variable-size token sets, we combine set encoding with permutation-invariant aggregation (Zaheer et al., 2017). Specifically, a Set Transformer encoder (Lee et al., 2019) uses self-attention without positional encodings to model interactions among transition tokens, producing $c_{1:M_W} = \mathrm{SetEnc}_\psi(\mathcal{E}_W)$. For permutation-invariant aggregation, a shared scoring network s_ζ assigns each contextualized token a scalar score. We normalize these scores with softmax and compute a weighted sum, which a learned map g_η transforms into the world code:

$$w_i = \frac{\exp(s_\zeta(c_i))}{\sum_{j=1}^{M_W} \exp(s_\zeta(c_j))}, \qquad h_W = g_\eta\Big(\sum_{i=1}^{M_W} w_i c_i\Big). \tag{2}$$

The resulting code summarizes the support evidence in a fixed-dimensional vector, regardless of the number or lengths of its fragments.

3.3 Query prediction and training

The world code h_W summarizes the support evidence and conditions prediction for a query from the same world W. For the query, we denote observations by o^q_t, where t indexes time steps. The shared observation encoder maps the first P observations to $z^q_t = E_\theta(o^q_t)$, $t = 0, \ldots, P-1$, forming the observed embedding $z^q_{0:P-1}$.

Starting from these embeddings, the causal predictor F_ω autoregressively predicts the next H steps. At each step, it uses the encoded observations, previously predicted embeddings, and supplied ac-

tions, conditioned on the world code:

$$\hat{z}_t^q = F_\omega\big(z_{0:P-1}^q, \hat{z}_{P:t-1}^q, a_{0:t-1}^q \mid h_W\big), \qquad P \le t < P + H. \tag{3}$$

At $t = P$, the predicted history $\hat{z}_{P:t-1}^q$ is empty. Each prediction is appended to the history for subsequent steps. The same world code h_W conditions the predictor throughout the rollout.

For each training world, we sample a support set and query interactions from that world. We jointly optimize the observation encoder, support encoding modules, and query predictor. The autoregressive predictions are supervised by embeddings of the future query observations, obtained using the shared encoder E_θ. Let $z^q = z_{P:P+H-1}^q$ denote this target sequence and $\hat{z}^q = \hat{z}_{P:P+H-1}^q$ its autoregressive prediction. The objective combines latent prediction error with SIGReg (Balestriero & LeCun, 2025):

$$\mathcal{L} = \mathbb{E}_W\left[\sqrt{\mathbb{E}_{\mathcal{S}_W, q|W}[\mathrm{MSE}(\hat{z}^q, z^q)]}\right] + \lambda_{\mathrm{SIGReg}}\mathcal{L}_{\mathrm{SIGReg}}. \tag{4}$$

Here, MSE averages squared residuals over forecast steps and latent dimensions. The inner expectation averages over support–query pairs $(\mathcal{S}_W, q)$ sampled from a world W. The outer expectation averages the resulting per-world root mean squared errors over training worlds.

We use $\lambda_{\mathrm{SIGReg}} = 0.1$ and apply SIGReg across the training batch to discourage representation collapse. This regularization acts on the observation embeddings rather than the world codes. At test time, support determines the conditioning code, while all model parameters remain fixed. Implementation details are provided in Appendix A.

4 Experiments

We first describe the experiment setup and baselines, then evaluate MetaDyn through two prediction questions and a downstream control study.

4.1 Experimental setup

Environments and evaluation. We evaluate prediction in PokeWorld, a planar tool–object environment (Tan et al., 2026); a spatial two-link mechanism in MuJoCo (Todorov et al., 2012); and the Push-T task (Chi et al., 2025). Figure 5 in Appendix A.1 illustrates these three environments and the physical parameters varied in each. For prediction, models receive images, proprioception, touch, and actions, and forecast future latents from two observed query frames. We measure physical average displacement error (ADE) using a separately fitted readout while keeping the trained model frozen. PokeWorld and MuJoCo experiments are evaluated on dynamics in-distribution (ID) and physics out-of-distribution (OOD) worlds. ID worlds contain unseen systems whose physical parameters are sampled within the training ranges, whereas Physics OOD worlds contain systems with one or more physical parameters extrapolated beyond those ranges. We report means and sample standard deviations over five training seeds. In Push-T, we evaluate both reaching the goal and maintaining the object near its target pose with low velocity for two seconds. Further environment and evaluation details are in Appendix A.

Baselines. Our comparisons aim to answer two questions: whether additional same-world support improves prediction, and whether MetaDyn's transition support representation provides an advantage over sequence-level support encoding. To study the first question, we compare MetaDyn with **Query-only** in PokeWorld and MuJoCo, a matched no-support ablation that retains MetaDyn's query architecture and prediction objective but removes support conditioning. We additionally include **LeWM** as a support-free latent world-model reference, adapting the LeWorldModel recipe (Maes et al., 2026) to the shared X-JEPA backbone (Tan et al., 2026). For Push-T control, we use the original **LeWM**, and additionally evaluate **LeWM + AdaJEPA**, which performs online test-time adaptation from recent interactions starting from the same trained LeWM checkpoint (Wang et al., 2026). To study the second question, we compare MetaDyn with **History**, which receives the same support interactions but encodes each fragment as a temporal sequence before pooling the resulting fragment summaries into a world code. Thus, **History** and MetaDyn differ primarily in how support evidence is represented and aggregated. Architectural and training details for all baselines are provided in Appendix A.

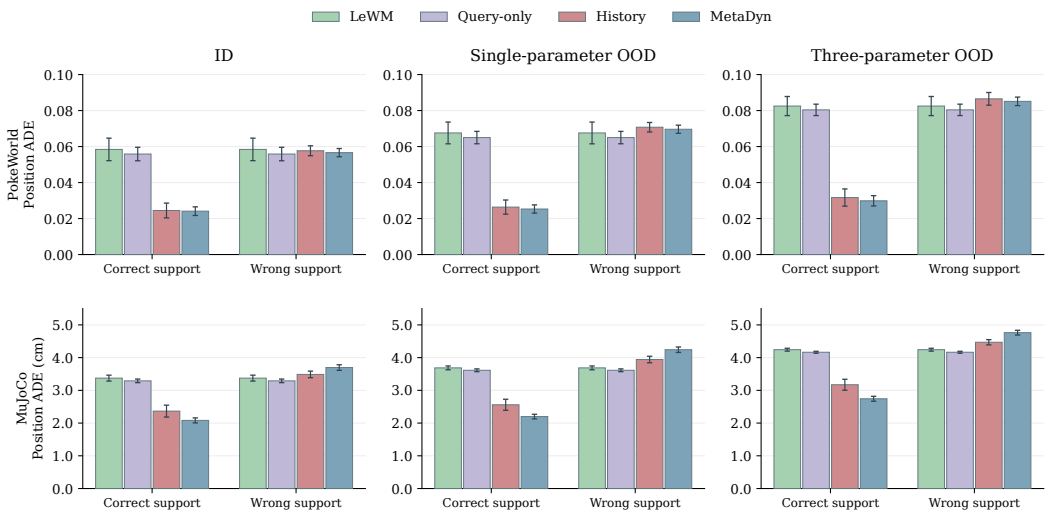

Figure 2: **Prediction depends on support's world assignment.** Rows show PokeWorld and MuJoCo; columns show ID, single-parameter OOD, and three-parameter OOD. Correct-world support yields lower error than wrong-world support for History and MetaDyn throughout. LeWM and Query-only do not use support.

Our evaluation proceeds in three stages. We first test whether prediction benefits from support drawn from the query's own physical world by comparing correct- and wrong-world support (Section 4.2). We then study how prediction changes with the organization and amount of same-world support, examining both repartitioning of fixed evidence and varying support length (Section 4.3). Finally, we extend the evaluation to multi-dynamics Push-T to assess support-conditioned prediction and its use for downstream reach-and-hold control (Section 4.4).

4.2 Does same-world support provide predictive evidence?

We first ask whether support from the same physical world as the query provides useful predictive evidence. To test this, we compare MetaDyn with correct-world support against the two no-support baselines, Query-only and LeWM. For the support-conditioned models, MetaDyn and History, we then replace correct-world support with support from other worlds and measure the resulting change in prediction error. The query observations, future actions, prediction targets, and model weights are kept fixed. For each query world, support from eight randomly sampled other worlds is evaluated separately, and the resulting prediction errors are averaged. Support contains 24 transitions in PokeWorld and 48 in MuJoCo, and we evaluate ID, single-parameter OOD, and three-parameter OOD worlds.

Figure 2 shows that correct-world support consistently improves prediction. With correct-world support, MetaDyn achieves lower mean ADE than both no-support baselines across both environments and all three evaluation distributions. Replacing correct-world support with support from other worlds increases MetaDyn's prediction error in every setting. Under three-parameter OOD, for example, its ADE increases from 0.02985 to 0.08510 in PokeWorld and from 2.743 to 4.763 cm in MuJoCo. The same assignment effect is present in MetaDyn's latent prediction space (Appendix A.3). History shows the same pattern, with lower prediction error under correct-world support than under wrong-world support. Together, these results show that support from the query's own physical world provides useful information for prediction. We next keep the support transitions fixed and examine whether their partition into fragments affects prediction.

4.3 How do support organization and amount affect prediction?

Section 4.2 shows that support from the query's own world improves prediction. We next examine how MetaDyn responds to two different changes in that support. First, we change only how a fixed

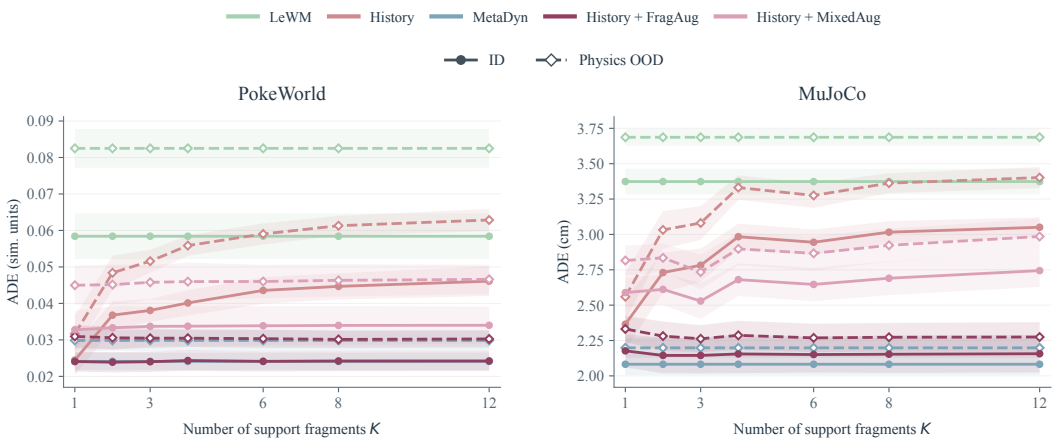

Figure 3: **Prediction under support repartitioning.** MetaDyn remains stable when the same transitions are split into more fragments. History approaches this stability through fragment or mixed augmentation, with FragAug also achieving comparable accuracy. Solid and dashed curves denote ID and Physics OOD, respectively; OOD is three-parameter extrapolation in PokeWorld and single-parameter extrapolation in MuJoCo.

set of transitions is organized into fragments, while keeping the evidence itself unchanged. Second, we change the length of support, i.e. number of support transitions available at inference. Together, these experiments examine whether prediction is stable under different partitions of the same support and how it changes as more support becomes available.

Support organization. We repartition the same 24 PokeWorld or 48 MuJoCo support transitions into $K \in \{1, 2, 3, 4, 6, 8, 12\}$ fragments, so that only the fragment boundaries change. Figure 3 compares MetaDyn with History and two augmented History variants. History + Fragment Augmentation (FragAug) varies the fragmentation of a fixed support budget during training, while History + Mixed Augmentation (MixedAug) also varies support length and trajectory source (Appendix A.4). To test whether the sequence-based baselines can recover the partition invariance built into MetaDyn, we additionally evaluate at $K = 4, 8$, which are excluded from augmentation training and model selection. Under this controlled repartitioning, the two support encoders behave differently. History is sensitive to the introduction of new fragment boundaries: in PokeWorld, its ID ADE increases from 0.02448 at $K = 1$ to 0.04610 at $K = 12$, while its Physics-OOD ADE increases from 0.03166 to 0.06287. In contrast, MetaDyn remains effectively unchanged as the same transitions are repartitioned. Fragment augmentation largely removes History's sensitivity to the partition, including at the held-out values of K, and brings its prediction accuracy close to MetaDyn. MixedAug also improves partition stability, but changes support length and trajectory source in addition to fragmentation, making it a less isolated comparison. MetaDyn is invariant to this evidence-preserving repartitioning by construction, whereas History must learn comparable robustness through augmentation.

Support amount. We then vary the number of support transitions available at inference while keeping the query observations, actions, targets, and model weights fixed. We evaluate $N \in \{4, 8, 16, 24, 32\}$ transitions in PokeWorld and $N \in \{8, 16, 32, 48, 64\}$ in MuJoCo; the corresponding training support lengths are 24 and 48 transitions. For each world, the support intervals are nested, with shorter intervals contained within longer ones. Figure 4 shows that prediction error decreases as the amount of correct-world support increases toward the training length for MetaDyn and History, whereas MetaDyn does not obtain comparable gains from wrong-world support. For MetaDyn, PokeWorld ID/Physics-OOD ADE decreases from 0.05848/0.07654 with four transitions to 0.02411/0.02985 with 24, while MuJoCo decreases from 3.578/3.866 cm with eight transitions to 2.082/2.198 cm with 48. At the training support length ($N = 24$ in PokeWorld and $N = 48$ in MuJoCo), MetaDyn has the lowest five-seed mean ADE among the compared methods in all four evaluation settings. Increasing wrong-world support does not produce a comparable improvement. With shorter support, MixedAug can perform better, consistent with its explicit training over varying support lengths. However, increasing support beyond the training length does not consis-

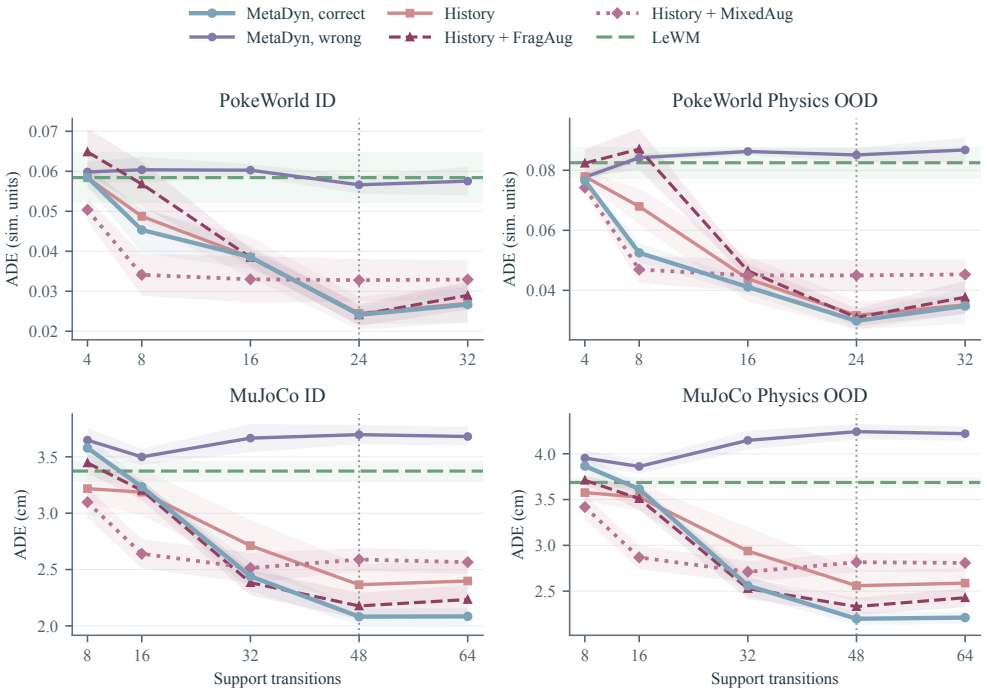

Figure 4: **Effect of inference-time support length.** Prediction error versus the number of support transitions, evaluated on ID and three-parameter OOD in PokeWorld and ID and single-parameter OOD in MuJoCo. MetaDyn is shown with same- and wrong-world support; History variants use same-world support. Dotted lines mark training support lengths, dashed gray lines show LeWM, and bands denote sample SD over five seeds.

tently yield further gains: performance slightly degrades at 32 transitions in PokeWorld and approximately plateaus beyond 48 transitions in MuJoCo. Since extending a contiguous support interval also changes which interactions are included, prediction need not improve monotonically with support length. Overall, MetaDyn remains stable to repartitioning of the same evidence and benefits from increasing matching support toward the training length.

4.4 Support-conditioned prediction and control in Push-T

We further evaluate MetaDyn in multi-dynamics Push-T, where object mass and translational/angular decay rate vary across worlds. The dataset contains 8,192 simulator-planned expert trajectories from 128 worlds, including 96 training, 16 validation, and 16 test worlds. We also evaluate on 160 Physics-OOD tasks from 40 worlds, divided into five groups: lower and higher mass, lower and higher decay rate, and joint extrapolation. Each group contains eight worlds and 32 tasks. In the Push-T experiments, all models use a multimodal extension of the native LeWorldModel backbone and are trained from scratch for 50,000 updates.

Support-conditioned prediction. We first ask whether using correct-world support still improves prediction in the Push-T domain. For each query, we keep the query observations, future actions, prediction targets, and model fixed, and replace correct-world support with support from other worlds. Table 1a reports the latent-space prediction error at the three-second forecast horizon. On ID test worlds, replacing correct support with wrong support increases the mean error from 0.06980 to 0.07750, with correct support yielding lower error in 15 of 16 worlds. Averaged equally across the five Physics-OOD groups, correct-world support reduces latent MSE from 0.11670 to 0.10739, an improvement of approximately 8.0%. This indicates that world-matched support provides useful information for prediction under physical distribution shifts.

Table 1: **Multi-dynamics Push-T.** (a) MetaDyn latent MSE under correct- and wrong-world support for ID and Physics OOD. (b) Decoded object-position ADE over 0.5–3 s (pixels; lower is better). (c) Object reach-and-hold success under the same controller for ID and Physics-OOD tasks. OOD results average equally across five groups, each containing 32 tasks.

(a) Support intervention (MSE).

Evaluation Types	Correct	Wrong
ID	0.06980	0.07750
OOD	0.10739	0.11670

(b) Physical prediction ADE (pixels).

Method	ID	OOD
LeWM	32.63	39.34
MetaDyn (correct)	**30.64**	**38.56**
MetaDyn (wrong)	31.52	38.64

(c) Control success (%).

Method	ID	OOD
LeWM	15.63	13.75
LeWM + AdaJEPA	18.75	10.63
MetaDyn	**20.31**	**18.13**

Physical prediction. We next evaluate whether the latent-space improvement translates to lower physical prediction error. For this evaluation, we keep LeWM and MetaDyn frozen and fit a separate readout for each model to decode object position over forecast horizons from 0.5 to 3 seconds. As shown in Table 1b, MetaDyn with correct support yields lower mean ADE than LeWM in both ID and OOD, with 30.64/38.56 pixels compared with 32.63/39.34. Within MetaDyn, correct support reduces ID ADE from 31.52 to 30.64 pixels, while the corresponding OOD difference is modest (38.64 vs. 38.56).

Downstream control. Prior latent world-model studies such as LeWM and AdaJEPA evaluate Push-T primarily with a goal-reaching criterion, without requiring the object to remain at the target for a sustained period (Maes et al., 2026; Wang et al., 2026). In a multi-dynamics setting, however, reaching the target momentarily does not ensure that the task has been completed stably: different mass and decay rates can produce different post-contact motion and settling behavior. We therefore use a stricter reach-and-hold criterion, requiring the object to remain near the target pose with low linear and angular velocity throughout the final two seconds. We then test whether the benefit of support-conditioned prediction translates into closed-loop control. For all downstream control evaluations, we use the same CEM controller configuration across methods. As shown in Table 1c, MetaDyn achieves 20.31% ID and 18.13% OOD reach-and-hold success, compared with 15.63% / 13.75% for LeWM and 18.75% / 10.63% for LeWM + AdaJEPA. Together, these results show that the predictive benefit of world-matched support is accompanied by higher reach-and-hold success in this multi-dynamics task.

5 Conclusion

We presented MetaDyn-JEPA (MetaDyn), a support-conditioned latent world model that uses interactions from the same physical system to inform multi-step prediction without test-time weight updates. MetaDyn represents support as a set of directed action–response transitions and aggregates them into a reusable world code that summarizes how the system responds to actions.

Across PokeWorld and MuJoCo, same-world support consistently improves prediction, while replacing it with wrong-world support increases prediction error. This contrast shows that the source of support matters for prediction and that MetaDyn makes meaningful use of the information it contains. MetaDyn also remains stable when the same support transitions are repartitioned into different fragments, whereas a sequence-based History encoder requires fragmentation augmentation to approach similar stability. Prediction generally improves as the amount of matching support increases toward the support length used during training. We further extend the evaluation to multi-dynamics Push-T. World-matched support improves latent prediction and yields lower physical prediction error, and MetaDyn achieves higher reach-and-hold success than LeWM and LeWM + AdaJEPA. These results suggest that interaction evidence from an unseen physical system can provide a useful conditioning signal for both prediction and downstream decision making.

Limitations and future work. MetaDyn assumes known support–query world correspondence. Selecting compatible support from unlabelled interaction datasets is an important extension when this association is unavailable. In addition, our evaluation is confined to simulation, which enables controlled comparisons across physical worlds. Future work should test prediction and closed-loop control on real systems, where sensor noise and contact variability may affect how much dynamics information support provides.

Reproducibility statement

Appendix A provides the environment and dataset specifications, model architectures, training settings, and evaluation protocols for all three domains, including support organization and Push-T control. Upon publication, we will release the code and configurations for data generation, model training, evaluation, and figure generation to enable reproduction of the reported results.

AI use statement

Generative AI assisted with grammar checking, figure preparation, and organizing the appendix. The authors are responsible for independently verifying the methods, code, results, and citations, and for all claims in the final manuscript.

References

Jan Achterhold and Joerg Stueckler. Explore the context: Optimal data collection for context-conditional dynamics models. In *International Conference on Artificial Intelligence and Statistics*, pp. 3529–3537. PMLR, 2021.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15619–15629. IEEE, 2023.

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.

Randall Balestriero and Yann LeCun. Lejepa: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Hao-Shu Fang, Hongjie Fang, Zhenyu Tang, Jirong Liu, Chenxi Wang, Junbo Wang, Haoyi Zhu, and Cewu Lu. Rh20t: A comprehensive robotic dataset for learning diverse skills in one-shot. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 653–660. IEEE, 2024.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International conference on machine learning*, pp. 1704–1713. PMLR, 2018.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pp. 3744–3753. PMLR, 2019.

Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 5757–5766. PMLR, 2020.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

Mohammadreza Nakheai, Aidan Scannell, Kevin Luck, and Joni Pajarinen. Contextual latent world models for offline meta reinforcement learning. *arXiv preprint arXiv:2603.02935*, 2026.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pp. 5331–5340. PMLR, 2019.

Kaizhen Tan, Xin Xu, Siru Tao, Hanzhe Hong, Yang Feng, and Heqing Du. What can latent world models know? physical parameter identifiability in multimodal predictive representations. *arXiv preprint arXiv:2607.27017*, 2026.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Ying Wang, Oumayma Bounou, Yann LeCun, and Mengye Ren. Adajepa: An adaptive latent world model. *arXiv preprint arXiv:2606.32026*, 2026.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

A Additional Experimental Details

A.1 Environments and Data

Figure 5 illustrates the three experimental environments and representative interactions. Each row identifies the interacting components and shows their evolution over time, with the physical parameters varied across worlds listed on the left.

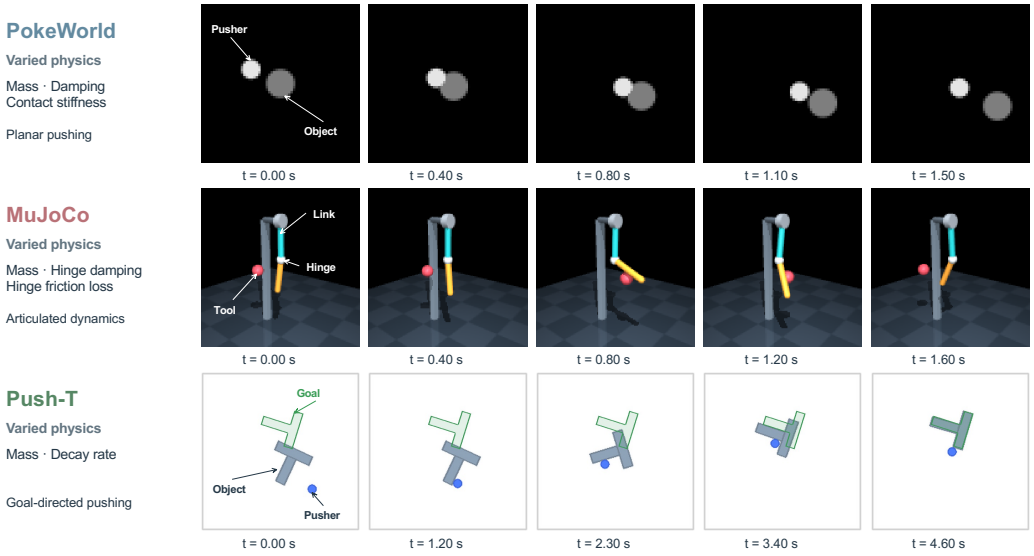

Figure 5: **Environments and representative dynamics.** PokeWorld shows forward pushing and release, MuJoCo shows an actuated tool interacting with a two-link mechanism, and Push-T shows rotation and translation toward a target pose. The initial frames identify the interacting components; subsequent frames show their evolution over time. The green outline marks the Push-T target pose.

PokeWorld environment and data. PokeWorld is a planar tool–object environment in which a pusher interacts with an object. We vary object mass, damping, and contact stiffness. The dataset contains 4096 training worlds, 128 validation worlds, and 512 probe worlds, with 512 worlds in each ID and Physics-OOD test split. Each world contains 12 trajectories of 42 frames at 64×64 image resolution. Training, validation, probe, and ID worlds are sampled independently using Latin-hypercube sampling and are disjoint at the world level.

The training ranges are approximately $[0.5981, 2.5079]$ for mass, $[0.85, 3.65]$ for damping, and $[641.04, 4679.86]$ for contact stiffness. Mass and stiffness are sampled in log space, while damping is sampled in linear space. ID worlds use unseen parameter combinations within these ranges. For three-parameter Physics OOD, each parameter independently falls below or above its training range with equal probability: mass lies in $[0.5, 0.5981] \cup [2.5079, 3.0]$, damping in $[0.5, 0.85] \cup [3.65, 4.0]$, and stiffness in $[500, 641.04] \cup [4679.86, 6000]$.

MuJoCo environment and data. We use MuJoCo 3.13.0 to simulate a two-link mechanism with two passive, nonparallel hinges and link lengths of 0.28 and 0.26 m. A spherical tool interacts with the mechanism through three force-limited translational position actuators. The simulator timestep is 0.001 s, and observations and actions are recorded every 0.04 s, yielding 81 observations and 80 actions per episode. Across worlds, we vary a shared mass multiplier for the mechanism, hinge damping, and hinge friction loss. Inertia scales proportionally with mass, while tool mass, geometry, gravity, camera, appearance, joint axes, and contact-surface parameters remain fixed.

The training ranges are $[0.5, 2.0]$ for the mass multiplier, $[0.004, 0.10]$ for hinge damping, and $[0.001, 0.012]$ for hinge friction loss. For three-parameter Physics OOD, all three parameters lie outside their respective training ranges. Each parameter takes a value from its lower or upper extrapolation range with equal probability, sampled log-uniformly within the selected range. These

ranges are $[0.35, 0.48]$ and $[2.08, 2.85]$ for the mass multiplier, $[0.002, 0.0038]$ and $[0.105, 0.20]$ for damping, and $[0.0005, 0.00095]$ and $[0.0126, 0.022]$ for friction loss.

Each parameter's training range is divided into four logarithmic bins, forming 64 parameter-combination cells: 48 training, eight combination-validation, and eight combination-test cells. Each partition covers all four bins of every individual parameter. The prediction experiments use 2048 training worlds, 128 validation worlds, and 256 probe worlds, with 256 worlds in each ID and Physics-OOD test split. Validation, probe, and ID worlds are independently sampled within the training cells and do not overlap with training worlds or one another. Each episode follows a two-push action program sampled independently of the physical parameters, with initial hinge angles in $[-0.18, 0.18]$ radians.

Single-parameter extrapolation. In addition to three-parameter Physics OOD, Figure 2 evaluates single-parameter extrapolation in both PokeWorld and MuJoCo. Exactly one physical parameter lies outside its training range, while the other two remain within their training ranges. The extrapolated parameter is balanced across mass, damping, and contact stiffness in PokeWorld, and across the mass multiplier, hinge damping, and hinge friction loss in MuJoCo. The single-parameter test splits contain 512 PokeWorld worlds and 256 MuJoCo worlds. The corresponding extrapolation ranges are the same as those specified above. Figure 2 reports ID, single-parameter OOD, and three-parameter OOD in both environments. Figures 3 and 4 use ID and three-parameter OOD for PokeWorld, and ID and single-parameter OOD for MuJoCo.

Push-T environment and data. We construct a multi-dynamics Push-T dataset containing 8,192 simulator-planned expert trajectories from 128 distinct physical worlds. Each world contributes 64 trajectories, yielding 6,144 trajectories from 96 training worlds, 1,024 from 16 validation worlds, and 1,024 from 16 ID test worlds. The splits are disjoint at the world level, with validation and ID worlds following the training parameter distribution.

Each world independently samples object mass $m \sim \mathrm{LogUniform}(0.5, 2)$ and decay rate $\gamma \sim \mathrm{Uniform}(\log(2)/0.8, \log(2)/0.05)$. Object inertia scales proportionally with mass. Translational and angular velocities decay by $\exp(-\gamma \Delta t)$ at each simulator step, giving both the same half-life $\log(2)/\gamma$; equivalently, linear drag is $m\gamma$ and angular drag is $I\gamma$, where I is the object's moment of inertia. The decay rate is sampled uniformly; the corresponding half-life is not. Geometry, appearance, and the pusher actuator remain fixed. The simulator timestep is 0.01 s, and observations and actions are recorded at 10 Hz.

We additionally evaluate on five Physics-OOD groups: lower mass, higher mass, lower decay rate, higher decay rate, and joint extrapolation. Each group contains eight worlds with four tasks per world, giving 40 worlds and 160 tasks in total. In the first four groups, only the specified parameter lies outside its training range, while the other remains within its training range. In the joint group, both parameters lie outside their training ranges, covering the four combinations of lower and upper extrapolation. OOD mass is sampled log-uniformly from $[0.4, 0.5)$ or $(2, 2.4]$, and OOD decay rate is sampled uniformly from $[0.8\gamma_{\min}, \gamma_{\min})$ or $(\gamma_{\max}, 1.2\gamma_{\max}]$. Their trajectories are separate from the 8,192-trajectory dataset.

Push-T control setting. All methods use the same latent-space CEM controller, with 200 candidates, ten iterations, 30 elites, and an initial standard deviation of one in normalized action space. Each planning round optimizes five action blocks, each containing five raw two-dimensional actions, covering 2.5 s. The model then predicts a further 2 s under physical zero action. The planning objective is the fused goal-latent MSE at the resulting 4.5 s endpoint. Only the first five raw actions are executed before replanning. The remaining optimized actions warm-start the next round, with zero-mean padding in normalized action space. Each task runs for 100 active steps followed by 20 zero-action steps, without early stopping. Ground-truth physical states are used only for evaluating executed trajectories, not for planning.

Push-T success criteria. Joint-position reach requires the combined pusher–object position error

$$\sqrt{\|p_{\mathrm{pusher}} - p^*_{\mathrm{pusher}}\|^2 + \|p_{\mathrm{object}} - p^*_{\mathrm{object}}\|^2} < 20$$

pixels and wrapped object-angle error below 20° at any active step. Object reach-and-hold requires object-position error below 20 pixels, angle error below 20°, linear speed below 5 pixels/s, and

angular speed below 0.1 rad/s throughout the final 21 observations, spanning two seconds. Joint reach-and-hold additionally requires the combined position threshold throughout this interval. Strict overlap-and-hold further requires T-polygon IoU of at least 0.90 and angle error below 5°.

A.2 Models and Training

Shared prediction backbone. PokeWorld and MuJoCo use the X-JEPA implementation of Tan et al. (2026), with a multimodal observation encoder and an action-conditioned causal predictor. Models are trained separately in each environment and predict autoregressively, appending predicted latents to the query history. PokeWorld inputs are grayscale 64×64 images, four finger-position/velocity channels, and seven tactile channels. GELU activations are used in the convolutional and MLP encoders; the fusion and latent output projections use BatchNorm. The query predictor retains temporal positional encodings.

MuJoCo inputs are $3 \times 128 \times 128$ RGB images, six tool-position/velocity channels, and seven local force/torque/contact channels; actions specify three-dimensional Cartesian target positions. The backbone uses an RGB input convolution, fixed 2×2 spatial average pooling, and adapted action/proprioceptive interfaces. The latent dimension remains 128, with 80 query positional embeddings. Backbone and MetaDyn support-branch dimensions are summarized below; action-input dimensions shown are for PokeWorld.

Component	Configuration
Image encoder	Four stride-two 3×3 convolutions, channels 32/64/128/256; projection $4096 \to 192$
Proprioception/touch	Each: hidden width 64, output width 96
Observation fusion	MLP $384 \to 192 \to 128$
Query Transformer	Four blocks; width 192; eight heads; feed-forward width 768; dropout 0.1
Support action encoder	MLP $2 \to 32 \to 32$
Transition encoder	MLP $288 \to 192 \to 192$
Support set Transformer	Two pre-normalized layers; width 192; four heads; feed-forward width 768; dropout 0.1
Attention-pooling scores	MLP $192 \to 192 \to 1$, tanh hidden activation
World-code head	GELU MLP $192 \to 192 \to 192$, followed by a linear eight-dimensional output

MetaDyn. MetaDyn encodes each support transition by concatenating the two endpoint latents, normalized with a shared LayerNorm, and the embedded action. A set Transformer without positional encodings processes the resulting directed tokens. Softmax attention pooling produces a deterministic world code, reused throughout the rollout and across queries sharing the support set. MuJoCo retains transition-token width 192 and code dimension eight, adapting the action input to three dimensions. Each predictor block uses a zero-initialized linear map from the code to six width-sized vectors that modify its action-conditioned adaptive-normalization shifts, scales, and gates.

History and its variants. History processes each support fragment in temporal order with the causal query trunk and pools its final hidden states into a world code of the same dimension as MetaDyn. It therefore aggregates sequence summaries rather than individual transitions. Neither method receives cross-fragment temporal connections, physical parameters, or future query observations. History + FragAug and History + MixedAug retain this architecture; their support sampling and organization are described in Appendix A.4.

Query-only. For PokeWorld and MuJoCo, query-only removes MetaDyn's support encoding and aggregation while retaining the query architecture, output heads, and world-modulation layers. It supplies a fixed eight-dimensional zero code, with trainable modulation biases, and uses MetaDyn's prediction objective as a loss-matched no-support comparison.

LeWM. For PokeWorld and MuJoCo, LeWM adapts the LeWorldModel recipe (Maes et al., 2026) to the shared X-JEPA backbone and autoregressive training. It uses only query observations and actions, with an MSE-plus-SIGReg objective.

Push-T models. Push-T uses the native LeWorldModel backbone extended with pusher proprioception and binary pusher–object contact: a trainable ViT-tiny encoder for 224×224 images, width 192 and 12 layers, followed by a six-layer predictor. Training queries contain four observations at offsets $0, 5, 10, 15$ raw steps. The first three latents predict shifted targets with causal teacher forcing; each action token concatenates five two-dimensional actions. Control uses up to three observed frames spaced five steps apart, followed by autoregressive prediction.

MetaDyn adds a width-192 transition encoder and a two-layer set encoder without positional embeddings, pooling a world code that modulates each predictor block.

LeWM with AdaJEPA. For Push-T control, LeWM + AdaJEPA adapts the trained LeWM online using executed query interactions (Wang et al., 2026). Every five execution steps, one Adam update fits detached next-latent targets using at most the latest 25 raw steps. It updates the final predictor Transformer layer and normalization at learning rate 5×10^{-4}, and the latent projector at 10^{-5}; other parameters and BatchNorm statistics remain fixed. Each update uses a fresh optimizer. Weights accumulate within a task and reset between tasks; history and goal latents are re-encoded after adaptation. There are 19 updates per task and none during final release.

Training objectives. In PokeWorld and MuJoCo, MetaDyn, History and its variants, and Query-only use latent RMSE plus 0.1 SIGReg. RMSE is $\sqrt{\mathrm{mean}(x^2) + 10^{-8}}$, where x contains prediction residuals over queries, forecast steps, and latent channels within each world; losses are then averaged across worlds. LeWM uses latent MSE with the same SIGReg coefficient. Auxiliary tactile/proprioceptive heads have zero loss weights and are unused in rollout or evaluation.

Encoder and predictor are trained jointly, with gradients through predicted and target latents and no stop-gradient or EMA target encoder. SIGReg uses 1024 normalized random projections and 17 integration points over $[-5, 5]$, with randomness determined by the training step. All Push-T models use latent MSE plus 0.09 SIGReg. Physical-state labels do not train the world model.

Optimization.

Setting	PokeWorld	MuJoCo	Push-T
Optimizer	AdamW	AdamW	AdamW
Training updates	20,000	20,000	50,000
Learning rate	3×10^{-4}	3×10^{-4}	5×10^{-5}
Weight decay	0.01	0.01	0.001
Batch size	32 worlds	16 worlds	128 queries
Warmup updates	500	500	200
Learning-rate schedule	Cosine	Cosine	Cosine

PokeWorld samples two queries per world. PokeWorld and MuJoCo use gradient clipping at five; MuJoCo uses deterministic float32 computation.

Checkpoint selection. In PokeWorld and MuJoCo, all prediction models, including Query-only, use validation-selected checkpoints from the same 20,000-update training budget. Selection minimizes validation prediction loss, checked every 1000 updates with each model's training objective. For checkpoint selection only, MuJoCo averages losses over two- and eight-observation query contexts; reported figures use two observed frames. Augmented History variants validate at $K = 1, 3$, and default models at $K = 1$. Push-T uses final 50,000-update checkpoints. LeWM and the RMSE-trained models share datasets and principal optimization settings, but use different sampling streams and, in MuJoCo, different validation episode subsets. No test metric selects model weights.

A.3 PokeWorld and MuJoCo Evaluation Protocols

At inference, BatchNorm uses fixed running statistics and dropout is disabled.

Query initialization and sensor noise. In PokeWorld, the first two query frames are generated using a kinematic initialization. This initial transition advances position at the existing velocity without updating velocity from contact forces or damping, and supplies no contact measurement.

Subsequent transitions use the full physical simulator. The initial two-frame context therefore provides position and motion information, but not the force response during that transition. This is a dataset-generation procedure shared by all methods, not an operation performed by the prediction model. Query centers are fixed by the recorded action programs. Independent zero-mean Gaussian noise with standard deviation 0.02 is added to the five force-derived touch channels on protocols 2–11, leaving contact fraction and the contact indicator unchanged. These dataset settings are shared by all compared methods.

PokeWorld support and query sampling. Default training samples one 24-transition support interval, choosing free motion, sustained contact (push), or impulsive contact (strike) with equal probability, then one of two recorded trajectories. Evaluation uses a fixed interval from a push trajectory. Training samples two of four query trajectories per world; validation and evaluation use all four. Each query provides two observed frames and a 16-frame prediction target.

MuJoCo support and query sampling. Support and queries are drawn from separate episodes of the same world. Each support contains 48 transitions, sampled with a random starting frame during training and a fixed starting frame during evaluation. Training samples two query episodes per world, while evaluation uses all three held-out query episodes. Queries contain two or eight observed frames and share the same 32-frame prediction target.

PokeWorld physical readout. After model selection and freezing, a $128 \to 256 \to 256 \to 4$ GELU MLP maps true-observation latents to (x, y, v_x, v_y). Probe worlds split 409/103 for fitting/validation. Normalization uses probe-training worlds only. Readouts use 1500 AdamW updates, learning rate 10^{-3}, weight decay 10^{-4}, batch 512; probe-validation normalized MSE selects the readout. Every model/seed receives its own readout with identical access rules. At test, it processes predicted latents. ADE measures Euclidean object-position error using only the decoded (x, y) coordinates, averaged over forecast frames and queries within each world; velocity outputs do not enter ADE. True-observation readout error is not a guaranteed lower bound on prediction error.

MuJoCo physical readout. A frozen $128 \to 256 \to 256 \to 6$ GELU readout predicts two distal-link 3D endpoints. Probe worlds split 204/52; normalization, optimization, and selection follow the PokeWorld readout protocol. ADE averages endpoint Euclidean error over two endpoints, 32 future frames, and three queries within a world. Reported errors are converted from metres to centimetres. Neither endpoint coordinates nor hidden physics train the representation learner.

Correct- and wrong-world support. Figure 2 uses the same validation-selected History and MetaDyn checkpoints and probe-fitted readouts as Figure 3, without retraining. Correct support contains one contiguous interval of 24 transitions in PokeWorld or 48 in MuJoCo. Support and queries come from separate episodes of the same world. Wrong-world support replaces only the support source, keeping the queries, future actions, targets, and model weights fixed.

For each query world, we uniformly sample eight distinct support-source worlds from the same evaluation split. All methods and training seeds share these assignments. Each source is evaluated separately with the same support length, and prediction errors are averaged across support sources, queries, and worlds within each training seed. World codes and predicted trajectories are not averaged. In single-parameter MuJoCo OOD, support-source worlds may differ in which parameter is extrapolated. LeWM uses no support and is evaluated once per query.

This test changes the source of support while fixing the query and target. Random donor initial states and action programs are not paired, so the result is not an isolated intervention on physical parameters. Correct-support ADE reproduces the corresponding Figure 3 $K = 1$ value within 2×10^{-5} in native units. The eight donors are repeated evaluations, not eight independent training replications.

Query-only evaluation. Query-only uses the same evaluation queries and independently fitted physical readouts, without receiving support. Its predictions are therefore unchanged across support conditions.

Physical and latent error aggregation. Physical ADE is evaluated through each model's independently probe-fitted readout in a common physical label space. The latent MSE pairs in Section 4.2 instead use each MetaDyn checkpoint's own encoder for the true future targets, averaging five seeds in each domain (PokeWorld: 10–14; MuJoCo: 30–34), with the eight wrong-donor errors averaged within each seed. These measure within-representation support sensitivity; independently trained encoders do not define a shared latent error scale. Seed counts are matched within each dataset, without interpreting task or noise repetitions as independent training replications.

Latent-space support intervention. MetaDyn's correct/wrong-support latent MSE is 0.18286/1.28331 on PokeWorld ID and 0.47007/1.51815 on three-parameter Physics OOD. MuJoCo gives 0.08503/0.16221 on ID and 0.09201/0.18872 on single-parameter Physics OOD. Each comparison uses the same trained representation and query targets while changing only support assignment. These values measure within-representation sensitivity, not error on a shared latent scale across independently trained encoders.

Statistics. The three main methods use five completed seeds in PokeWorld (10–14) and continuous-parameter MuJoCo (30–34). History and MetaDyn retain the original RMSE runs; LeWM uses the completed MSE runs. All completed seeds are included without outcome-based filtering. Means and sample SDs are computed across training seeds after averaging within-seed query and noise repetitions. Matching seed indices does not make the RMSE and MSE training streams identical, and these descriptive summaries do not constitute paired-seed significance tests. Physical ADE is comparable in a common label space; latent support-intervention numbers use each RMSE MetaDyn checkpoint's own encoder and are not compared across independently learned representations.

A.4 Support Organization and History Augmentation

Default support organization. MetaDyn and History use one contiguous support interval during default training, containing 24 transitions in PokeWorld and 48 in MuJoCo. History encodes the interval as a temporal sequence, whereas MetaDyn encodes its individual directed transitions before set aggregation. History + FragAug and History + MixedAug retain the History architecture and RMSE-plus-SIGReg objective. Both variants use the same underlying training datasets, 20,000 updates, and five training seeds. Each model has its own validation-selected checkpoint and separately fitted physical readout.

Support repartitioning. Figure 3 evaluates $K \in \{1, 2, 3, 4, 6, 8, 12\}$ fragments while retaining the same 24 PokeWorld or 48 MuJoCo support transitions. A contiguous interval is divided into equal-length fragments, preserving temporal order within each fragment. The transition endpoints are retained at fragment boundaries, so every original transition remains available with unchanged multiplicity. No transitions are constructed between separate fragments. Query observations, future actions, prediction targets, model weights, and readouts remain fixed. History encodes the resulting fragments separately and pools their summaries; MetaDyn aggregates the same transition tokens regardless of their partition.

History + FragAug. FragAug samples K uniformly from $\{1, 2, 3, 6, 12\}$ during training. It partitions one contiguous interval of 24 PokeWorld or 48 MuJoCo transitions into equal-length fragments and randomly permutes the fragment order while preserving temporal order within each fragment. The support transition budget remains fixed. Neither $K = 4$ nor $K = 8$ is used for training or checkpoint selection; both are held out for evaluating generalization to unseen fragment counts. Augmented History variants use validation partitions $K = 1, 3$, whereas default History and MetaDyn use $K = 1$, as specified in Appendix A.2.

History + MixedAug. MixedAug varies support trajectory source, length, and partition. With probability 0.25, it samples three trajectories from the same world, taking eight transitions from each of PokeWorld's three support motion protocols or 16 transitions from each of MuJoCo's three support episodes. Otherwise, it samples one contiguous interval with a transition budget chosen uniformly from $\{4, 8, 16, 24, 40\}$ in PokeWorld or $\{8, 16, 32, 48, 64\}$ in MuJoCo. When the sampled budget equals the default length of 24 or 48 transitions, respectively, the interval is partitioned

with probability 0.5, using K sampled uniformly from $\{2, 3, 6, 12\}$ and randomly permuting the fragments. The resulting branch probabilities are 25% multi-trajectory support, 67.5% contiguous support, and 7.5% partitioned support. All support trajectories come from the same physical world. Unlike FragAug, MixedAug does not keep the training transition budget constant.

Inference-time support length. Figure 4 varies only the number of support transitions supplied to the trained models, using $N \in \{4, 8, 16, 24, 32\}$ in PokeWorld and $N \in \{8, 16, 32, 48, 64\}$ in MuJoCo. Each prediction receives one contiguous interval beginning at frame 0 in PokeWorld or frame 8 in MuJoCo. Shorter intervals are nested within longer ones. Model weights, readouts, query observations, future actions, and prediction targets remain fixed; no model is retrained for a different support length. The checkpoints, readouts, and query episodes match Figures 2 and 3; query worlds use the ID and three-parameter OOD splits for PokeWorld and the ID and single-parameter OOD splits for MuJoCo. MetaDyn is evaluated with both correct-world and wrong-world support, using the same eight other worlds as in the support-assignment evaluation. History and its augmented variants use correct-world support. LeWM uses no support and is evaluated on the same queries as a length-independent reference. Errors are averaged over queries and worlds within each training seed, with wrong-world errors additionally averaged over the eight support-source worlds. Curves show means and sample standard deviations across five training seeds.

A.5 Push-T Experimental Setup

Expert acceptance, cropping, and normalization. Raw trajectories contain 320 actions and 321 observations. Simulator planning allows 300 active steps followed by 20 zero-action steps. Acceptance requires polygon IoU ≥ 0.90, object angle error $< 5^\circ$, position error < 20 pixels, speed < 5 pixels/s, and angular speed < 0.1 rad/s over the final 20 observations. Accepted trajectories are selected in attempt order, not by learned-model performance. Training trims each trajectory at the earliest eligible 21-observation stable interval (two full seconds), retaining at least 40 actions; trajectories without an eligible endpoint retain their full length. No trajectories are removed or padded. Action normalization is fitted only to the 1,966,080 raw actions of the 6,144 training trajectories. The dataset expands the previous 4,096-trajectory version within the same worlds; all current models are trained from scratch, not continued from earlier weights.

Each Physics-OOD world contributes four control tasks, giving 160 tasks across five groups. Two separate cropped episodes from the same world provide support. OOD data are excluded from world-model training and normalization statistics.

Support clips and run details. MetaDyn uses two clips from separate same-world episodes, each containing nine observations at raw-step offsets $0, 5, \ldots, 40$, spanning four seconds. Each clip supplies eight transitions, giving 16 in total; each transition includes the intervening five two-dimensional actions. Online AdaJEPA is evaluated on the same checkpoint and task set.

Additional adaptation details. The predictor uses training mode during adaptation and evaluation mode during planning. Adaptation has a separate random stream from CEM, uses no support or privileged physics, and preserves the same planning budget. The first five actions, before any update, match the frozen counterpart.

Physical prediction readouts. LeWM and MetaDyn remain frozen at their final-50K checkpoints, and each has its own MLP readout, with widths 192–256–256–9 and GELU activations. Targets are pusher position, object position, sine/cosine of object angle, object linear velocity, and angular velocity. We use the first eight accepted episodes per world, taking 16 evenly spaced frames from each cropped episode: 12,288 frames from the 96 training worlds and 2,048 frames from the 16 validation worlds. All models use identical frame indices. Input and output normalization are fitted only on training frames. Readouts train for 20,000 AdamW updates (batch 256, learning rate 10^{-3}, weight decay 10^{-3}); validation normalized state MSE is checked every 200 updates and selects the readout checkpoint. No world-model parameter is trained with these labels.

Evaluation uses four fixed queries per world: 64 queries across the 16 ID worlds and 160 across the 40 Physics-OOD worlds. OOD ADE is averaged equally across the five groups. Three observed frames spaced 0.5 s apart provide a one-second prefix. Given recorded future actions, the model rolls

out six latent predictions over the next three seconds. We decode each prediction and report mean Euclidean object-position error over these six horizons (ADE), in simulator pixel coordinates, not meters. Correct/wrong support uses the same donor protocol as the latent diagnostic; wrong-world physical errors are averaged after separate forecasts and decoding.

Support-intervention diagnostic. We evaluate four cropped query windows per ID world. Given three observed latents and recorded future actions, the model autoregressively predicts six latents over 0.5–3 seconds. Correct support comes from the query world; wrong support comes from each of the other 15 ID worlds, with prediction errors averaged across support-source worlds. Queries, actions, targets, and model weights remain fixed. Evaluation uses held-out expert trajectories.

Table 1a reports the three-second endpoint. Errors are first averaged over the four queries within each world, then equally across worlds. For OOD evaluation, each query receives correct support from its own world and wrong support from each of the other seven worlds within the same OOD group. Each wrong-world forecast is evaluated separately before averaging errors. We compute the mean over the eight worlds in each group and then average equally across the five groups. Within-model correct/wrong comparisons share the same representation; raw MSE magnitudes are not used to rank independently learned encoders. This diagnostic measures support-dependent prediction on held-out expert trajectories, separately from closed-loop control.

Control tasks. The ID control set contains 64 tasks from 16 independent ID worlds, with four tasks per world. The OOD control set contains 160 tasks from 40 worlds across the five groups, also with four tasks per world. All methods use the same tasks and controller configuration.