SELF-SUPERVISED CONTINUOUS DYNAMICS LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Temporal modeling is essential for general intelligence to understand change and anticipate how the world evolves. Consecutive frames in a video often share similar semantics while carrying different state information. The dynamic state-transition information between these frames is important for describing their evolution along a continuous time axis. However, conventional self-supervised methods primarily learn associations between frames, lacking explicit awareness of continuous state evolution. To address this issue, we introduce **TemDy**, a self-supervised continuous dynamics learning framework comprising Temporal Variation Encoding and Continuous Dynamics Modeling. Temporal Variation Encoding uses 3D rotary position encoding to incorporate spatial and temporal information. A DELTA register learns to encode state transitions through supervision from deterministic state reconstruction and uncertain future prediction. Continuous Dynamics Modeling represents normalized visual states through history-conditioned rotations, separating the direction of change from its temporal progression. Integrating continuous angular velocity from an observation-derived anchor then models temporal dynamics. Evaluations of temporal localization, frame retrieval, and state reconstruction demonstrate that TemDy has improved temporal dynamics modeling capabilities compared with diverse self-supervised baselines.

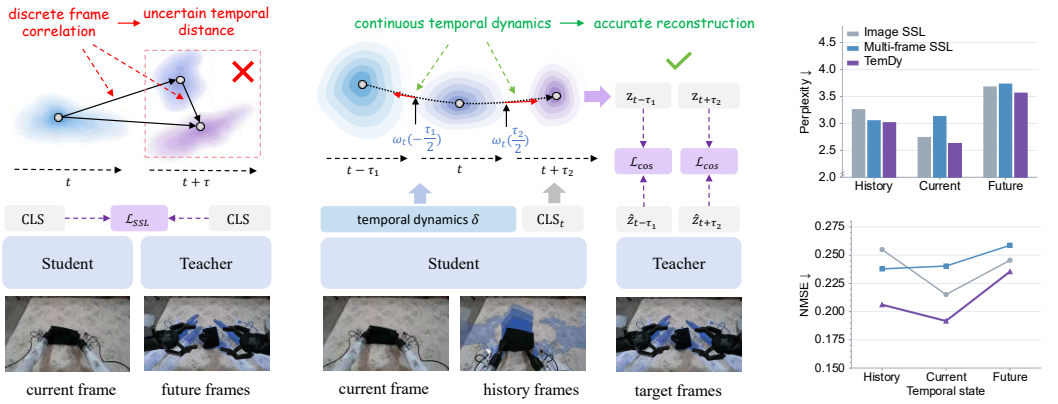
1 INTRODUCTION

As general intelligence advances, understanding temporal state transitions becomes increasingly important for visual models. In practical applications, robot manipulation connects perception with planning (Ji et al., 2025, Team et al., 2025), while first-person video understanding must capture ongoing activities (Damen et al., 2022). When observed frames are provided at varying intervals, models must account for the time gaps between frames, not just their order. Combining observed state transitions with the time intervals between frames provides a foundation for understanding continuous state transitions along the temporal axis.

Conventional image and video self-supervision establish useful associations between observations through self-distillation, masked recovery, and latent prediction (Caron et al., 2021, He et al., 2022, Bardes et al., 2024). Within a continuous video sequence, consecutive frames often share similar semantics but carry different state information. However, learning their associations alone does not explicitly capture how these states evolve along a continuous time axis (Figure 1(a)). This raises a central question for visual encoders: *how to model continuous state evolution to develop dynamic temporal understanding?*

To formalize continuous state evolution, we describe visual state transitions through a shared, history-conditioned trajectory indexed by elapsed time. This trajectory unifies deterministic reconstruction of observed states and uncertain future prediction. Modeling transitions between semantically similar frames requires describing both their direction and temporal progression. Rotations naturally separate these two factors through a plane and an angle. Inspired by this structure, we model visual states through rotations driven by continuous angular velocity. Specifically, this formulation offers two advantages. **(1) Explicit representation of state transitions.** The rotation plane specifies the direction of evolution, while angular displacement describes progression from an observation-derived anchor. This makes subtle directional differences between semantically similar frames explicit modeling targets. **(2) Continuous temporal progression.** Integrating a

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.



(a) Existing Methods. **(b) TemDy models continuous evolution.** **(c) Dynamic Prediction Quality.**

Figure 1: **Motivation of TemDy.** **(a)** Discrete frame associations alone cannot ensure accurate temporal modeling. **(b)** Continuous temporal dynamics improve TemDy's understanding of state transitions. **(c)** TemDy improves perplexity and NMSE across history, current, and future prediction on AgiBot.

shared angular-velocity function connects states at different query times, including times between observed frames. Under fixed history and plane, exact integration also makes direct transformations consistent with compositions over their subintervals (Section 3.2).

Building on this geometric formulation, we introduce **TemDy**, a self-supervised continuous dynamics learning framework comprising Temporal Variation Encoding and Continuous Dynamics Modeling. Temporal Variation Encoding incorporates spatial coordinates and actual timestamps through 3D rotary position encoding. A DELTA register aggregates information from historical frames and the visible deterministic future observation. Deterministic state reconstruction and uncertain future prediction supervise this register to encode the dynamic information needed to model state transitions. Continuous Dynamics Modeling uses CLS and DELTA to construct an observation-derived anchor, a rotation plane, and a continuous angular-velocity function. Integrating angular velocity rotates the anchor to produce states along the shared trajectory, as illustrated in Figure 1(b). We evaluate temporal dynamics modeling through temporal localization, frame retrieval, and state reconstruction against diverse self-supervised baselines. Reconstruction results show improved recovery of historical, visible-endpoint, and future states, as illustrated in Figure 1(c).

Our contributions are:

- We introduce TemDy, a self-supervised framework for learning continuous temporal dynamics from historical observations. It learns state-transition information through deterministic reconstruction and uncertain future prediction along a shared trajectory.
- We formulate a rotational model that separates the direction of state evolution from its temporal progression. We analyze how this structure explicitly represents fine-grained inter-frame transitions and connects transitions across time intervals.
- We evaluate temporal localization, frame retrieval, and state reconstruction across video sources and transfer tasks. Controlled ablations examine the roles of rotational parameterization, temporal supervision, and DELTA conditioning.

2 Related Work

2.1 Self-Supervised Visual Learning

Self-supervised visual learning constructs targets from images themselves to learn transferable representations. DINO (Caron et al., 2021, Oquab et al., 2023, Siméoni et al., 2025) methods use teacher-student self-distillation, iBOT (Zhou et al., 2021) extends supervision to masked local features, and MAE (He et al., 2022) reconstructs masked image content. I-JEPA (Assran et al., 2023) predicts target-region embeddings, providing another route to learning visual structure without

reconstructing pixels. Overall, image self-supervision provides a basis for object recognition, spatial understanding, and dense prediction. A complementary line shapes the embedding geometry directly. SimCLR (Chen et al., 2020) aligns augmented views through a contrastive objective. VICReg (Bardes et al., 2021) combines view invariance with variance and covariance regularization to discourage collapse and redundant features. These approaches learn visual content and relationships between views. However, they do not explicitly define how visual states evolve along a continuous time axis. TemDy complements visual representation learning by supervising historical state transitions and organizing them through a continuous rotational trajectory.

2.2 Temporal Dynamics Learning

Video representation learning exploits multiple frames through temporal association, masked recovery, and latent prediction. VideoMAE (Tong et al., 2022) reconstructs heavily masked video content, MaskFeat (Wei et al., 2022) predicts features of masked regions, and V-JEPA (Bardes et al., 2024, Assran et al., 2025, Mur-Labadia et al., 2026) methods predict target-encoder representations without pixel reconstruction. These objectives provide routes to learning motion and state transitions, but do not by themselves specify a continuous-time evolution rule. For continuous video representation, NeRV (Chen et al., 2021) maps an encoded frame timestamp to an RGB image by fitting a video-specific neural function. Its temporal coordinate parameterizes a recorded video, rather than a dynamics model conditioned on historical observations. Neural ODEs (Chen et al., 2018) parameterize continuous hidden-state evolution, while Latent ODEs (Rubanova et al., 2019) extend this framework to irregularly sampled sequences. These formulations do not specifically address how visual self-supervision can preserve state-transition information between semantically similar frames. To address this problem, TemDy combines historical reconstruction and future prediction with a history-conditioned rotational trajectory, separating transition direction from temporal progression.

3 Methodology

3.1 Problem Formulation

Our goal is to model continuous visual state transitions from a sequence of timestamped observations. Let $\mathcal{H}_t = \{(I_{t_i}, t_i)\}_{i=1}^{N}$ contain N historical images, where I_{t_i} is the image recorded at timestamp t_i and $t_1 < \cdots < t_N = t$. The last timestamp t defines the temporal reference. The terminal image $I_{t_N} = I_t$ is the visible deterministic future observation relative to the preceding history. It remains part of the observed input sequence, rather than an unseen prediction target. The history-conditioned function $\mathbf{z}_t(\tau) \in \mathbb{R}^d$ predicts the representation at a signed offset of τ seconds from t, where d is the feature dimension. The same function describes state evolution along a continuous time axis, rather than restricting predictions to fixed frame positions. Negative, zero, and positive τ identify earlier historical states, the visible deterministic future observation, and uncertain future states, respectively. For each available target image $I_{t+\tau}$, $\hat{\mathbf{z}}_{t+\tau}$ denotes its global feature from a same-scale exponential-moving-average (EMA) image teacher. We use unaccented $\mathbf{z}$ for predictions and $\hat{\mathbf{z}}$ for teacher targets. The teacher encodes each target independently, without history or query-time information.

To define a common prediction space for rotational state transitions, we normalize nonzero target features onto the unit sphere $\mathbb{S}^{d-1}$ in $\mathbb{R}^d$. Matched direction and norm readouts support this choice in Appendix E. Both teacher targets $\hat{\mathbf{z}}_{t+\tau}$ and predicted states $\mathbf{z}_t(\tau)$ lie on this sphere. Within this geometry, we represent evolution in the plane spanned by a unit reference-state anchor $\mathbf{c}_t$ and a unit tangent direction $\mathbf{u}_t \perp \mathbf{c}_t$. A continuous signed angular velocity $\omega_t(\tau)$ controls progression at relative time τ. Its integral $\theta_t(\tau)$ is the accumulated rotation angle in radians. The plane and velocity are inferred from the observed history, with one velocity function shared by all query times:

$$\theta_t(\tau) = \int_0^{\tau} \omega_t(\tau')\, d\tau', \qquad \mathbf{z}_t(\tau) = \cos\theta_t(\tau)\mathbf{c}_t + \sin\theta_t(\tau)\mathbf{u}_t, \tag{1}$$

Here τ' is the integration variable over the interval from zero to τ.

3.2 Theory Motivation

Rotational angular-velocity modeling offers two advantages: **explicit state-transition representation**, separating evolution direction from progression, and **continuous temporal progression**,

connecting queried states through a shared angular-velocity integral. We elaborate on these advantages through the following propositions.

Proposition 1 (Local transition representation)**.** *For a twice continuously differentiable unit-sphere trajectory with nonzero initial velocity, a planar rotation can match its initial state and velocity with local approximation error $O(\tau^2)$, whereas static copying has error of order $|\tau|$ as $\tau \to 0$.*

Proof. Let the trajectory be $\boldsymbol{\gamma}(\tau)$, with initial state $\mathbf{c} = \boldsymbol{\gamma}(0)$ and speed $v = \|\boldsymbol{\gamma}'(0)\|_2 > 0$. Differentiating the unit-norm constraint gives $\mathbf{c}^\top \boldsymbol{\gamma}'(0) = 0$. Thus, with unit tangent $\mathbf{u} = \boldsymbol{\gamma}'(0)/v$, construct $\mathbf{r}(\tau) = \cos(v\tau)\mathbf{c} + \sin(v\tau)\mathbf{u}$. This rotation satisfies $\mathbf{r}(0) = \boldsymbol{\gamma}(0)$ and $\mathbf{r}'(0) = \boldsymbol{\gamma}'(0)$. Taylor expansion gives the same first-order term for both trajectories, so

$$\|\boldsymbol{\gamma}(\tau) - \mathbf{r}(\tau)\|_2 = O(\tau^2), \qquad \|\boldsymbol{\gamma}(\tau) - \mathbf{c}\|_2 = v|\tau| + O(\tau^2). \tag{2}$$

Differentiating the unit-norm constraint again yields $\mathbf{c}^\top \boldsymbol{\gamma}''(0) = -v^2$, hence

$$1 - \mathbf{c}^\top \boldsymbol{\gamma}(\tau) = \tfrac{1}{2} v^2 \tau^2 + o(\tau^2), \qquad \mathbf{u}^\top \boldsymbol{\gamma}(\tau) = v\tau + O(\tau^2). \tag{3}$$

Even when neighboring states have cosine similarity close to one, the tangent component retains first-order signed dynamic information. Full error bounds and boundary cases are provided in Appendix C.1.

Rotation explicitly separates transition direction from angular progression, retaining local dynamic information even when neighboring states have high cosine similarity. Angular velocity specifies the rate of this evolution.

Proposition 2 (Interval composition)**.** *Under the same fixed history and plane, define the rotation generator $\mathbf{A}_t = \mathbf{u}_t \mathbf{c}_t^\top - \mathbf{c}_t \mathbf{u}_t^\top$ and the transform from relative time a to b as $R_t(b, a) = \exp\Big(\mathbf{A}_t \int_a^b \omega_t(\tau')\, d\tau'\Big)$, where* exp *denotes the matrix exponential. Then*

$$R_t(c, b) R_t(b, a) = R_t(c, a) \tag{4}$$

for any endpoints a, b, c within the same modeled time interval.

Proof. Write $\Theta_{ab} = \int_a^b \omega_t(\tau')\, d\tau'$. The orthonormal plane basis gives

$$\mathbf{A}_t \mathbf{c}_t = \mathbf{u}_t, \qquad \mathbf{A}_t \mathbf{u}_t = -\mathbf{c}_t, \qquad [\mathbf{A}_t \Theta_{ab}, \mathbf{A}_t \Theta_{bc}] = \mathbf{0}, \tag{5}$$

where $[X, Y] = XY - YX$ is the matrix commutator. The first two identities show that $R_t(b, a)$ rotates the plane through Θ_{ab}, giving $R_t(b, a)\mathbf{z}_t(a) = \mathbf{z}_t(b)$. Integral additivity gives $\Theta_{ac} = \Theta_{ab} + \Theta_{bc}$. Since all intervals share the same rotation generator $\mathbf{A}_t$,

$$R_t(c, b) R_t(b, a) = e^{\mathbf{A}_t(\Theta_{bc} + \Theta_{ab})} = R_t(c, a). \tag{6}$$

Thus, direct and composed transitions agree. Full proofs and numerical error bounds are provided in Appendix C.

3.3 Self-Supervised Continuous Dynamics Learning

Learning continuous dynamics requires preserving visual semantics while encoding the dynamic information that distinguishes successive states. As shown in Figure 2, TemDy addresses these needs by adding temporal supervision for deterministic history reconstruction and uncertain future prediction to conventional self-supervised learning.

3.3.1 Temporal Variation Encoding

A patch embedding maps each image into spatial tokens P_{t_i}. The tokens are processed in temporal order with a shared CLS token and a learnable DELTA register. Omitting auxiliary registers, the sequence is $[\mathrm{CLS}, \mathrm{DELTA}, P_{t_1}, \ldots, P_t]$. CLS summarizes global visual semantics, while DELTA learns dynamic information for modeling state transitions through deterministic reconstruction and uncertain future prediction objectives.

To distinguish spatial location from temporal separation, we extend rotary position encoding (Su et al., 2024) to three axes. A patch at relative time $\tau_i = t_i - t$ and location (y, x) receives separate

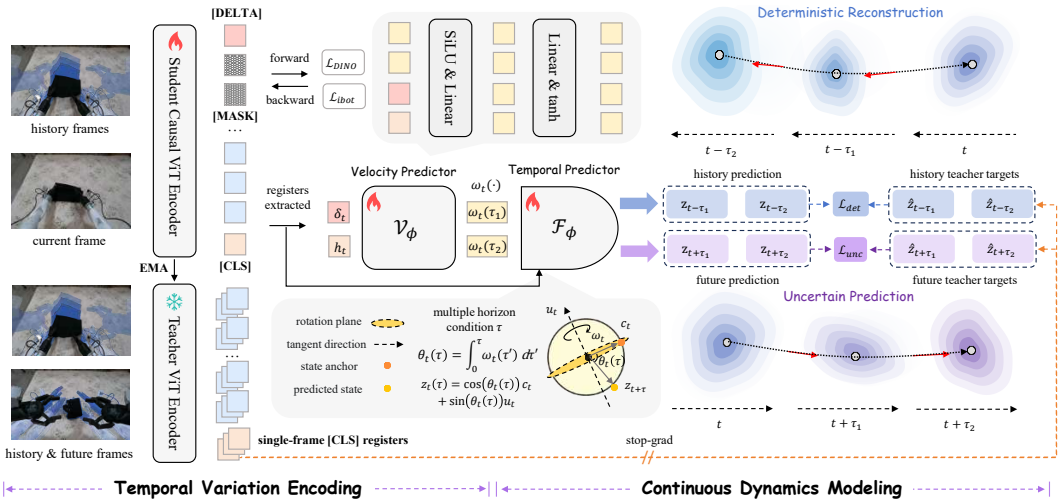


Figure 2: **TemDy overview.** DINO and iBOT preserve single-image features, while causal encoding summarizes observations in CLS and DELTA. The temporal predictor $\mathcal{F}_\phi$ constructs the anchor and rotation direction, integrates angular velocity, and rotates the anchor to predict temporal states.

rotations in the temporal and spatial channels of its query and key vectors. Temporal phases use elapsed seconds rather than frame indices, preserving the timing of irregular observations. CLS and DELTA are not position-rotated, but attend to patch keys that contain this spatiotemporal information. Therefore, their attention can distinguish where and when features appear.

Frame-causal attention preserves the order of information flow between images. Patch tokens attend to their own and earlier frames, but not to CLS or DELTA. Both global tokens can read the entire observed sequence, collecting information without passing it back into the patch stream, as shown in Figure 8. The encoder E_ϑ, with trainable parameters ϑ, produces

$$(\mathbf{h}_t, \boldsymbol{\delta}_t) = E_\vartheta(\mathcal{H}_t)_{\text{CLS,DELTA}} \tag{7}$$

where $\mathbf{h}_t$ and $\boldsymbol{\delta}_t$ are extracted from the output CLS and DELTA registers, respectively. Their concatenation conditions the prediction module. To retain image-level semantic and dense spatial information, we keep $\mathcal{L}_{\text{DINO}}$ for global student–teacher distribution alignment and $\mathcal{L}_{\text{iBOT}}$ for masked patch-level teacher matching. These image self-supervised objectives complement temporal reconstruction (Appendix B.6).

3.3.2 Continuous Dynamics Modeling

Image self-supervision preserves CLS semantics, while temporal reconstruction guides DELTA to capture state transitions. Within the temporal predictor $\mathcal{F}_\phi$, we first construct the visible deterministic future anchor through a time-independent residual map:

$$\mathbf{c}_t = \text{norm}_2(\text{sg}(\mathbf{h}_t) + W_c\,\text{LN}([\text{sg}(\mathbf{h}_t), \boldsymbol{\delta}_t]))\,. \tag{8}$$

Here $[\cdot,\cdot]$ denotes concatenation, LN is layer normalization, and $W_c \in \mathbb{R}^{d\times 2d}$ is a learned residual projection. $\text{norm}_2(\mathbf{x}) = \mathbf{x}/\|\mathbf{x}\|_2$ and sg denote unit normalization and stopped gradients. Both $\mathbf{h}_t$ and $\boldsymbol{\delta}_t$ have dimension d. At $\tau = 0$, the accumulated angle is zero, so $\mathbf{z}_t(0) = \mathbf{c}_t$ reconstructs the teacher feature of the visible deterministic future image I_t. For subsequent evolution, we define the condition $\boldsymbol{\xi}_t = [\text{sg}(\text{norm}_2(\mathbf{h}_t)), \boldsymbol{\delta}_t] \in \mathbb{R}^{2d}$. The angular-velocity head evaluates $\omega_t(\tau) = v_\phi(\boldsymbol{\xi}_t, \mathbf{e}(\tau))$, where $\mathbf{e}(\tau)$ combines elapsed seconds with Fourier features. Its signed scalar output allows acceleration, deceleration, and reversal within one plane. Here ϕ collects the learned parameters of the prediction module.

The temporal predictor $\mathcal{F}_\phi$ combines anchor construction, direction mapping, angular integration, and rotation to produce $\mathbf{z}_t(\tau) = \mathcal{F}_\phi(\mathbf{h}_t, \boldsymbol{\delta}_t, \omega_t, \tau)$. Its internal learned mapping predicts $\mathbf{a}_t = g_u(\boldsymbol{\xi}_t)$ and projects it onto the anchor's tangent space:

$$\mathbf{u}_t = \text{norm}_2\big(\mathbf{a}_t - (\mathbf{a}_t^\top \mathbf{c}_t)\mathbf{c}_t\big)\,. \tag{9}$$

For a nonzero projected direction, $\mathbf{u}_t$ and $\mathbf{c}_t$ form an orthonormal basis shared by all queries. We approximate $\theta_t(\tau)$ with K-node Gauss–Legendre quadrature, obtaining $\widetilde{\theta}_t(\tau)$. Different queries change the integration interval while sharing the same angular-velocity function. Node mappings and weights are detailed in Appendix B.

Finally, $\mathcal{F}_\phi$ converts the accumulated angle into the predicted state by rotating the anchor:

$$\mathbf{z}_t(\tau) = \cos\big(\widetilde{\theta}_t(\tau)\big)\,\mathbf{c}_t + \sin\big(\widetilde{\theta}_t(\tau)\big)\,\mathbf{u}_t. \tag{10}$$

This rotation preserves unit norm even with the approximate angle. The prediction is supervised by $\hat{\mathbf{z}}_{t+\tau}$, with gradient routing detailed in Appendix B.6.

Let $\mathcal{Q}_{\mathrm{det}}$ contain valid recorded-state queries at $\tau \le 0$, including the anchor supervision in Eq. 8, and let $\mathcal{Q}_{\mathrm{unc}}$ contain valid queries at $\tau > 0$ with available target images across a training batch. The subscripts det and unc denote deterministic and uncertain targets, respectively. Each set indexes a window and its relative query time. $|\mathcal{Q}|$ denotes the number of valid queries in that set. For nonempty query groups, with $\ell(\mathbf{z}, \hat{\mathbf{z}}) = 1 - \mathbf{z}^\top \hat{\mathbf{z}}$, we use

$$\mathcal{L}_{\mathrm{det}} = \frac{1}{|\mathcal{Q}_{\mathrm{det}}|} \sum_{\mathcal{Q}_{\mathrm{det}}} \ell(\mathbf{z}_t(\tau), \hat{\mathbf{z}}_{t+\tau}), \qquad \mathcal{L}_{\mathrm{unc}} = \frac{1}{|\mathcal{Q}_{\mathrm{unc}}|} \sum_{\mathcal{Q}_{\mathrm{unc}}} \ell(\mathbf{z}_t(\tau), \hat{\mathbf{z}}_{t+\tau}). \tag{11}$$

The two losses average over the available targets in their respective groups. We give deterministic reconstruction and uncertain future prediction equal weight in $\mathcal{L}_{\mathrm{rec}} = \frac{1}{2}(\mathcal{L}_{\mathrm{det}} + \mathcal{L}_{\mathrm{unc}})$. Combining reconstruction with the structural objective gives

$$\mathcal{L} = \mathcal{L}_{\mathrm{struct}} + \lambda_{\mathrm{rec}} \mathcal{L}_{\mathrm{rec}}. \tag{12}$$

Here λ_{rec} weights the reconstruction objective, and $\mathcal{L}_{\mathrm{struct}}$ combines image self-supervision, DELTA consistency, and representation regularization (Appendix B.6).

4 Experiments

4.1 Experimental Setting

Setup. We use ViT-B with 16-pixel patches, training on a mixture of robot demonstrations, first-person videos, other videos, and still images (Table 1). The pool contains 10 million sampled windows and images from 20 sources. TemDy follows the DINOv3 register configuration, retaining four registers and assigning one to DELTA. Training uses AdamW, a global batch of 512, and eight H100 GPUs. The optimization schedule covers approximately 20,000 updates. From-scratch training uses a peak learning rate of 2.83×10^{-3} and weight decay increasing from 0.04 to 0.4. For fine-tuning, TemDy uses peak rates of 10^{-5} for the backbone and 2.8×10^{-5} for rotation modules. The estimated full-schedule cost of the from-scratch recipe is 1,159 GPU-hours. Exact schedules and remaining settings are given in Appendix B. Hyperparameter sensitivity and pretraining loss curves appear in Appendix B.4 and Figure 10, respectively.

Table 1: **Training data.**

Data type	Main datasets	Clips	Percent
Robot	AgiBot, DROID	3,348,144	33.48%
First-person	Ego4D, EgoDex	3,949,063	39.49%
Other video	Kinetics, Something-Something V2	2,284,536	22.85%
Still images	Open Images, Places365, COCO	418,257	4.18%
Total	**20 sources**	**10,000,000**	**100%**

Evaluation protocol. We evaluate temporal understanding on three datasets. AgiBot (Bu et al., 2025) provides 21,803 training windows and 1,781 evaluation windows from disjoint episodes across ten tasks. EPIC-KITCHENS-100 (Damen et al., 2022) contributes 17,335 fitting and 2,000 evaluation windows, while Something-Something V2 (Goyal et al., 2017) provides 20,000 fitting and 2,000 evaluation windows. Following our formulation, deterministic reconstruction covers historical states and the visible endpoint, while uncertain prediction targets unseen future states. The three tasks recover earlier input frames, the terminal input, and unseen frames after t_N, respectively. We use normalized mean squared reconstruction error (NMSE) to assess target-feature recovery. We further evaluate physical reasoning on IntPhys 2 and Physion, which test physical plausibility and contact anticipation, respectively. Sampling rules and metric definitions appear in Appendix A, with benchmark-specific settings in Appendix B.

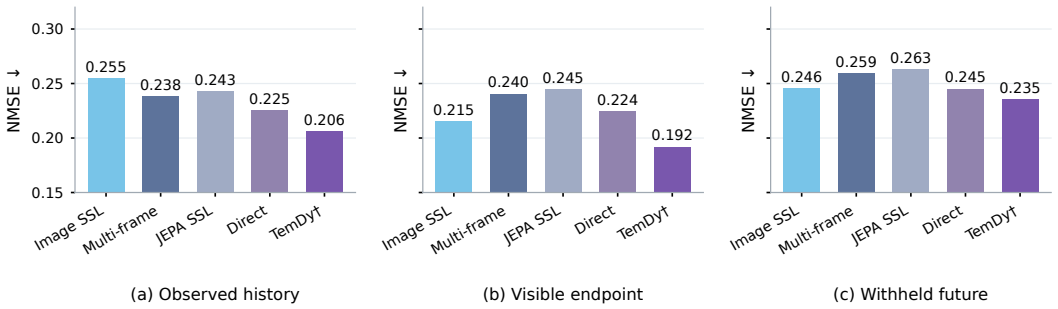


Figure 3: **State reconstruction capabilities.**

Table 2: **Temporal reconstruction capabilities.** Shared-target NMSE ↓. Current is the visible endpoint. †: scratch. ∗: DINOv3 initialization. Group-wise best in bold, second best underlined.

Method	AgiBot			EPIC-KITCHENS-100			Something-Something V2		
	History	Current	Future	History	Current	Future	History	Current	Future
ViT-B training checkpoints									
Image SSL	0.2550	0.2152	0.2455	**0.5158**	0.5166	0.6118	0.4137	0.2671	0.3908
Multi-frame SSL	0.2379	0.2404	0.2588	0.5639	0.5206	0.6266	0.4079	0.2741	0.3880
JEPA SSL	0.2430	0.2449	0.2630	0.5882	0.5075	0.6204	0.4129	0.3459	0.4213
Direct Prediction	0.2254	0.2242	0.2450	0.5918	0.4193	0.5999	0.4103	0.2736	0.3908
TemDy†	**0.2061**	**0.1917**	**0.2355**	0.5501	**0.3579**	**0.5781**	**0.3390**	**0.1646**	**0.3359**
Pretrained ViT-B models									
DINOv3	0.2961	0.2815	0.2915	0.8099	0.7258	0.7996	0.6916	0.6245	0.6725
V-JEPA2.1	0.2657	0.2643	0.2747	0.7128	0.6651	0.7238	0.6003	0.5514	0.5821
PE Core	0.2614	0.2618	0.2706	0.7723	0.7073	0.7741	0.6701	0.6074	0.6602
PE Spatial	0.2628	0.2616	0.2726	0.7364	0.6340	0.7275	**0.5361**	**0.4559**	**0.5183**
TemDy∗	**0.2434**	**0.2411**	**0.2531**	**0.7033**	**0.6025**	**0.6975**	0.5630	0.4561	0.5319

Baselines. We compare TemDy with self-supervised training baselines and pretrained visual encoders. Image SSL and Multi-frame SSL both use DINO (Caron et al., 2021)-style self-supervised training, with single-frame inputs and joint encoding of multiple observations, respectively. JEPA SSL adopts the V-JEPA2.1 (Mur-Labadia et al., 2026) masked latent prediction objective. Direct Prediction augments DINO-style self-supervision with a time-conditioned prediction head that maps observation features and query time directly to the target state. DINOv3 (Siméoni et al., 2025), V-JEPA2.1 (Mur-Labadia et al., 2026), and the Core and Spatial variants of Perception Encoder (PE) (Bolya et al., 2025) are pretrained references.

4.2 Main Results

4.2.1 Temporal Modeling Capabilities

Table 2 compares historical, visible-endpoint, and future reconstruction across three datasets (Figure 3). On AgiBot, both TemDy variants attain the lowest NMSE in their respective groups across all three tasks. Future NMSE decreases from 0.2588 (Multi-frame SSL) to 0.2355 (TemDy†), and from 0.2915 (DINOv3) to 0.2531 (TemDy∗). TemDy† likewise improves all three reconstruction metrics over Multi-frame SSL on EPIC-KITCHENS-100 and Something-Something V2. These results support the effectiveness of explicitly learning state-transition information. Temporal supervision encourages DELTA to retain frame-specific dynamics, while the rotational formulation organizes state evolution through a shared angular-velocity function. Together, these designs provide temporal structure beyond joint frame encoding, consistent with improved state recovery across the three tasks. Additional retrieval and localization results appear in Table 7, with protocols in Appendix A.

4.2.2 Physical Reasoning Capabilities

Table 3 evaluates physical plausibility on IntPhys 2 (Bordes et al., 2025) and contact prediction on Physion (Bear et al., 2021). On IntPhys 2 Continuity, TemDy† and TemDy∗ reach 60.00% and 55.83%, versus 51.67% for Direct Prediction and 50.00% for PE Spatial in their respective

Table 3: **Physical reasoning results.** IntPhys 2: unit-feature mean-window pair accuracy. Physion: scenario AUC (%, ↑). Avg.: seven-column mean. †: scratch. ∗: fine-tuned. Group-wise best in bold, second best underlined (ties shared).

Method	IntPhys 2			Physion				Avg.
	Continuity	Solidity	Immutability	Support	Drop	Link	Collide	%
ViT-B trained from scratch								
Image SSL	50.00	**50.68**	**52.50**	67.87	58.21	59.42	70.54	58.46
Multi-frame SSL	46.67	46.58	48.33	66.56	64.09	63.47	74.85	58.65
JEPA SSL	45.00	48.63	48.33	**83.43**	64.79	**67.95**	**78.12**	62.32
Direct Prediction	51.67	47.95	45.83	69.87	64.50	61.35	67.82	58.43
TemDy†	**60.00**	**50.68**	**52.50**	76.06	**66.57**	64.38	77.07	**63.89**
Pretrained ViT-B models								
DINOv3	41.67	44.52	45.83	86.63	75.70	74.07	71.59	62.86
V-JEPA2.1	46.67	43.84	39.17	**89.76**	76.72	73.68	**83.09**	64.70
PE Spatial	50.00	49.32	50.83	78.56	74.45	70.45	78.41	64.57
TemDy∗	**55.83**	**52.74**	**53.33**	87.20	**78.60**	**74.24**	77.72	**68.52**

Table 4: **Temporal parameterization.** Seen and unseen denote supervised and held-out query times. Avg.: mean of the two All columns. Bold: best learned predictor.

Method	History Recovery			Uncertain Future			Avg.
	All ↓	Seen ↓	Unseen ↓	All ↓	Seen ↓	Unseen ↓	NMSE ↓
Direct	0.4516	0.4185	0.5176	0.5072	0.4953	0.5191	0.4794
Flow Matching	0.5886	0.5805	0.6046	0.5483	0.5381	0.5586	0.5685
Constant	0.4618	0.4371	0.5113	0.5012	0.4945	0.5078	0.4815
TemDy	**0.4416**	**0.4170**	**0.4907**	**0.4340**	**0.4266**	**0.4414**	**0.4378**

Table 5: **Deterministic and uncertain objectives.** Temporal losses vary from the same DINOv3 initialization. Image SSL remains fixed. Training details appear in Appendix B.2.

$\mathcal{L}_{\text{det}}$	$\mathcal{L}_{\text{unc}}$	Deterministic Reconstruction				Uncertain Prediction			
		NMSE ↓	Top-1 ↑	MAE ↓	PPL ↓	NMSE ↓	Top-1 ↑	MAE ↓	PPL ↓
✓	–	0.2873	40.22	2.741	3.634	0.3064	**44.48**	3.031	4.269
–	✓	**0.2856**	40.22	2.766	**3.579**	0.3058	42.64	3.067	4.409
✓	✓	0.2859	**40.84**	**2.700**	3.604	**0.3027**	**44.48**	**2.940**	**4.161**

comparison groups. On Immutability, the two variants achieve 52.50% and 53.33%, respectively, tying for first and ranking first within their groups. The Continuity gains are consistent with learning state transitions along a shared trajectory, which provides a temporal reference for detecting implausible motion. On Physion, TemDy† and TemDy∗ achieve average AUCs of 71.02% and 79.44% across the four reported scenarios. These results suggest that the learned representations retain physical interaction information useful for contact prediction.

4.3 Ablation and Mechanism

Temporal parameterization. With the same frozen DINOv3 encoder and matched budgets, we compare direct state prediction, flow matching, constant angular velocity, and TemDy's time-varying angular velocity. TemDy achieves the lowest NMSE among learned predictors on both tasks (Table 4), reducing average error by 8.7% relative to Direct and 9.1% relative to Constant. The shared rotational trajectory organizes state transitions, while the time-varying rate accommodates nonuniform evolution, consistent with the structural advantages discussed in Section 3.2. Because Constant also uses a rotational trajectory, this comparison supports adapting the evolution rate to time rather than using rotation alone.

Deterministic and uncertain objectives. Table 5 compares deterministic and uncertain objectives under matched budgets. Relative to deterministic-only training, the reported joint-objective results improve historical Top-1 accuracy from 40.22% to 40.84%, reduce future NMSE from 0.3064 to 0.3027, and reduce future MAE from 3.031 to 2.940 seconds. Deterministic reconstruction supervises realized

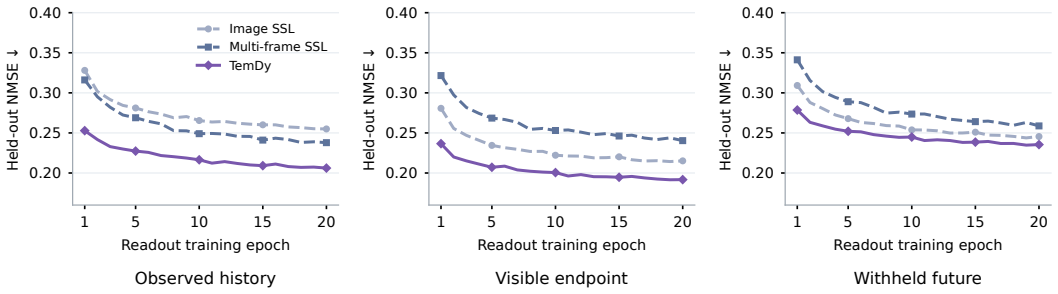


Figure 4: **Learning to recover temporal states.**

transitions, while uncertain prediction constrains their continuation beyond the observed frames. Future-only training has slightly lower historical NMSE and perplexity, which may reflect a balance between state discrimination and feature recovery in the shared parameters under joint supervision.

DELTA conditioning. Figure 5 examines DELTA's contribution by comparing correct, zero, and shuffled inputs. Correct DELTA lowers future NMSE from 1.2522 (zero) and 1.0955 (shuffled) to 0.7578. Against shuffled DELTA, historical NMSE falls from 1.0304 to 0.6514 and endpoint NMSE from 0.8757 to 0.4517. Shuffling preserves model-produced features but breaks their correspondence with the observed sequence. Unlike zeroing, it supplies a DELTA representation extracted from another observation sequence, making the comparison less dependent on an artificial zero input. The resulting degradation shows that the predictor depends on sequence-specific information in DELTA, rather than merely the presence of an additional token. Correct DELTA supports recovery across history, endpoint, and future states, consistent with its role in conditioning the state anchor and rotational trajectory. Additional DELTA interventions appear in Figure 9 and Table 13. Fixed-anchor DELTA and observation-interval interventions after head adaptation are reported in Tables 16–17.

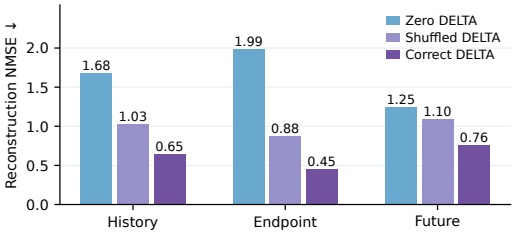


Figure 5: **DELTA dependence.**

Reconstruction learning dynamics. As shown in Figure 4, TemDy† maintains lower NMSE than Image SSL and Multi-frame SSL throughout fitting in all three tasks. Most error reduction occurs in the early epochs, but TemDy's advantage persists as the curves flatten, suggesting improved state readability beyond faster initial optimization. The separation is clearer for history and endpoint recovery, indicating better retention of preceding states and the observed endpoint. Future curves lie closer together, yet TemDy retains the lowest error, suggesting that its temporal information also supports recovery beyond the input observations. This pattern is consistent with complementary temporal supervision: deterministic reconstruction encourages retention of realized transitions, while uncertain prediction encourages information useful for their continuation. Discrimination results and efficiency measurements appear in Appendices A and B.5, respectively.

5 Conclusion

We introduced TemDy, a self-supervised framework for continuous visual state transitions. Deterministic historical and uncertain future temporal supervision guides DELTA to encode dynamics, while angular-velocity integration connects states along a shared rotational trajectory. Experiments support TemDy's effectiveness in continuous temporal modeling.

Limitations and future work. A fixed plane limits trajectory complexity, and rotations may revisit states, motivating further study of complex changes and temporal ambiguity. Computational constraints also limit the training data and model scales explored here. Scaling both remains an important direction for testing the reach of temporal dynamics learning beyond the present experimental setting.

AI USE STATEMENT

We used OpenAI Codex to assist with manuscript refinement. Codex also assisted with theoretical formalization, plotting code, data-processing and experiment-scheduling code, and LaTeX formatting. No other generative AI tools were used, and no synthetic datasets were generated using AI. The authors checked the proofs, verified result sources and references, and tested the code. The authors take responsibility for the final manuscript and associated artifacts.

ETHICS STATEMENT

This work studies self-supervised visual representations for temporal state modeling. Video and robot observations may contain identifiable people or sensitive environments, requiring care in data access and sharing. The reported offline evaluations do not establish the safety of deploying TemDy in physical systems. Such deployment would require application-specific validation and appropriate human oversight.

REPRODUCIBILITY STATEMENT

Section 3 describes the model formulation and training objectives. Appendices A and B document the temporal benchmark protocol, evaluation metrics, and implementation details, including numerical integration. Appendix C provides theoretical derivations, and Appendix D describes the flow-matching comparison. Appendix E specifies the supplementary diagnostic settings. Appendix F reports training dynamics and the associated hyperparameters.

REFERENCES

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)*, pages 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.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

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.

Daniel M Bear, Elias Wang, Damian Mrowca, Felix J Binder, Hsiao-Yu Fish Tung, RT Pramod, Cameron Holdaway, Sirui Tao, Kevin Smith, Fan-Yun Sun, et al. Physion: Evaluating physical prediction from vision in humans and machines. *arXiv preprint arXiv:2106.08261*, 2021.

Daniel Bolya, Po-Yao Huang, Peize Sun, Jang Hyun Cho, Andrea Madotto, Chen Wei, Tengyu Ma, Jiale Zhi, Jathushan Rajasegaran, Hanoona Bangalath, Junke Wang, Marco Monteiro, Hu Xu, Shiyu Dong, Nikhila Ravi, Shang-Wen Li, Piotr Dollar, and Christoph Feichtenhofer. Perception encoder: The best visual embeddings are not at the output of the network. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 60884–60937. Curran Associates, Inc., 2025. doi: 10.52202/085713-2036. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/57bc0a850255e2041341bf74c7e2b9fa-Paper-Conference.pdf.

Florian Bordes, Quentin Garrido, Justine T Kao, Adina Williams, Michael Rabbat, and Emmanuel Dupoux. Intphys 2: Benchmarking intuitive physics understanding in complex synthetic environments. *arXiv preprint arXiv:2506.09849*, 2025.

Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *2021 IEEE/CVF international conference on computer vision (ICCV)*, pages 9630–9640. IEEE, 2021.

Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser Nam Lim, and Abhinav Shrivastava. Nerv: Neural representations for videos. *Advances in Neural Information Processing Systems*, 34:21557–21568, 2021.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR, 2020.

Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. *International Journal of Computer Vision*, 130(1):33–55, 2022.

Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The "something something" video database for learning and evaluating visual common sense. In *2017 IEEE international conference on computer vision (ICCV)*, pages 5843–5851. IEEE, 2017.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 15979–15988. IEEE, 2022.

Yuheng Ji, Huajie Tan, Jiayu Shi, Xiaoshuai Hao, Yuan Zhang, Hengyuan Zhang, Pengwei Wang, Mengdi Zhao, Yao Mu, Pengju An, et al. Robobrain: A unified brain model for robotic manipulation from abstract to concrete. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1724–1734. IEEE, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mido Assran, Koustuv Sinha, Mike Rabbat, Yann LeCun, Nicolas Ballas, and Adrien Bardes. V-jepa 2.1: Unlocking dense features in video self-supervised learning. *arXiv preprint arXiv:2603.14482*, 2026.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

BAAI RoboBrain Team, Mingyu Cao, Huajie Tan, Yuheng Ji, Xiansheng Chen, Minglan Lin, Zhiyu Li, Zhou Cao, Pengwei Wang, Enshen Zhou, et al. Robobrain 2.0 technical report. *arXiv preprint arXiv:2507.02029*, 2025.

Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35:10078–10093, 2022.

Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan Yuille, and Christoph Feichtenhofer. Masked feature prediction for self-supervised visual pre-training. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 14648–14658. IEEE, 2022.

Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. *arXiv preprint arXiv:2111.07832*, 2021.

Appendix guide. Appendix A presents evaluation examples and temporal protocols. Appendix B provides training settings, physical-reasoning protocols, efficiency measurements, and hyperparameter analyses. Appendix C gives theoretical derivations, Appendix D discusses conditional flow matching, and Appendix E reports DELTA and timing interventions, native-head adaptation, and direction and norm readouts. Compact training curves appear in Appendix F.

A Temporal Evaluation Protocols and Additional Results

A.1 Datasets and Temporal Queries

We evaluate temporal understanding on AgiBot, EPIC-KITCHENS-100, and Something-Something V2. Table 6 summarizes the fitting and evaluation windows.

Table 6: **Data splits for temporal evaluation.**

Dataset	Fitting windows	Evaluation windows
AgiBot	21,803	1,781
EPIC-KITCHENS-100	17,335	2,000
Something-Something V2	20,000	2,000

AgiBot uses episode-disjoint fitting and evaluation splits. Something-Something V2 uses distinct source videos, with one window sampled per video.

Each evaluation distinguishes observed history, the visible endpoint, and unobserved future states. For AgiBot, the observed offsets are $(-12, -9, -6, -4)$ before rebasing. The last observed frame defines time zero, giving observation times $(-8, -5, -2, 0)$. The four target-only frames are queried at $(1, 2, 3, 4)$ seconds relative to this endpoint. Historical recovery evaluates the first three observations, while visible-endpoint recovery evaluates the fourth. Future target frames are excluded from the encoder input.

EPIC-KITCHENS-100 also evaluates four future queries at $(1, 2, 3, 4)$ seconds after the last observed frame, yielding 8,000 evaluation queries. Figure 6 illustrates the data examples and evaluation procedures.

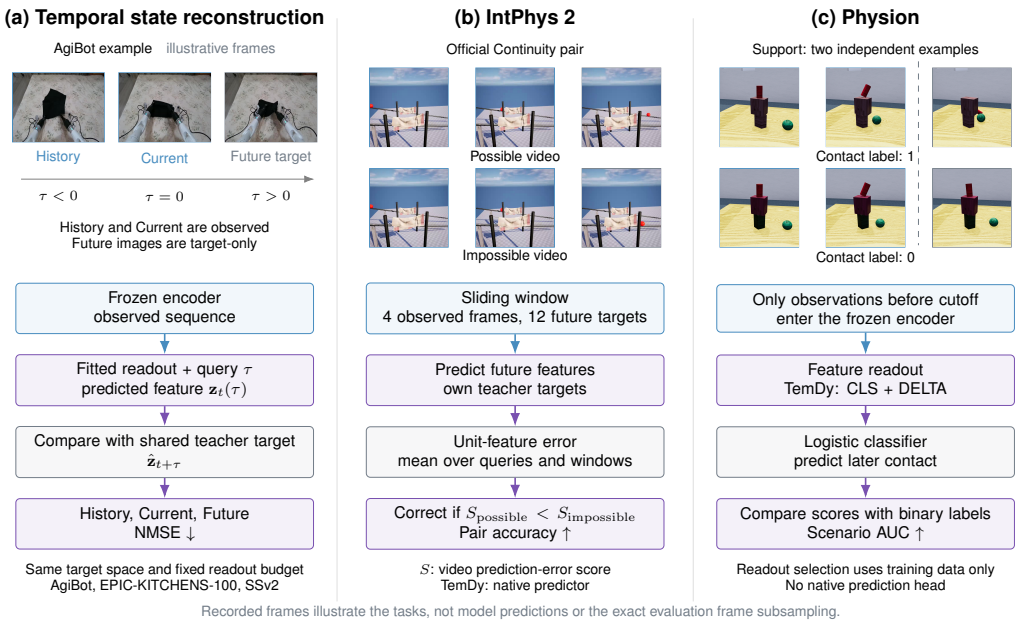


Figure 6: **Data examples and evaluation protocols for Tables 2 and 3.** (a) Temporal state reconstruction. (b) Physical plausibility assessment on IntPhys 2. (c) Contact prediction on Physion.

A.2 Readout Fitting and Evaluation Metrics

Readout fitting. Encoders remain frozen during evaluation. Regression heads and candidate scorers are fitted separately. Fresh heads use one hidden layer of width 256 and train for 20 epochs with AdamW, learning rate 10^{-3}, batch size 64, and seed 17. Evaluation uses the final epoch without selecting a checkpoint on the evaluation split.

For EPIC-KITCHENS-100 and Something-Something V2, History and Current reuse the final future-trained regression head and its training-set normalization without additional fitting. TemDy uses CLS and DELTA features for the shared-target readouts. EPIC-KITCHENS-100 uses native state representations for the Image SSL, Multi-frame SSL, Direct Prediction, and TemDy entries. The corresponding Something-Something V2 entries use framewise four-frame features. JEPA models use joint-video global features.

State recovery. We measure recovery of target features using normalized mean squared error:

$$\mathrm{NMSE} = \frac{1}{nd} \sum_{i=1}^{n} \sum_{k=1}^{d} \left(\frac{y_{ik} - \hat{y}_{ik}}{\sigma_k} \right)^2 . \tag{13}$$

Here, y_{ik} and $\hat{y}_{ik}$ denote predicted and teacher-target coordinates, respectively. The feature dimension is d, and n is the number of evaluated queries. The scale σ_k is the population standard deviation of target coordinate k, computed only from fitting queries. Coordinates with standard deviation at most 10^{-8} use a denominator of one. Fitting statistics remain fixed during evaluation.

State retrieval and temporal localization. A separately trained candidate scorer selects the corresponding target from four candidates. This evaluation uses learned scores rather than direct cosine ranking. Historical and visible-state candidates use absolute teacher features. Future candidates use feature differences relative to the visible-reference teacher feature, consistently across methods. Candidate timestamps are not provided as classifier inputs.

Top-1 accuracy measures correct candidate selection. Temporal MAE measures the absolute timestamp difference between the selected candidate and the target, in seconds. Both metrics use the same argmax decision. These metrics assess whether the learned representation distinguishes temporally corresponding states, complementing feature recovery measured by NMSE.

A.3 State Recovery and Temporal Discrimination

Table 7: **Future-state recovery and temporal discrimination.** Shared-target readouts are evaluated by NMSE, four-way retrieval accuracy, and temporal MAE in seconds. † denotes training from scratch, and $*$ denotes DINOv3 initialization. The best result within each group is highlighted.

Method	AgiBot			EPIC-KITCHENS-100			Something-Something V2		
	NMSE ↓	Top-1 ↑	MAE ↓	NMSE ↓	Top-1 ↑	MAE ↓	NMSE ↓	Top-1 ↑	MAE ↓
ViT-B training checkpoints									
Image SSL	0.2455	**41.98**	0.795	0.6118	25.74	1.229	0.3908	29.74	0.384
Multi-frame SSL	0.2588	40.34	0.819	0.6266	27.33	1.187	0.3880	30.11	0.384
JEPA SSL	0.2630	41.93	**0.793**	0.6204	28.21	1.172	0.4213	28.30	0.402
Direct Prediction	0.2450	37.58	0.889	0.5999	29.49	1.138	0.3908	28.76	0.396
TemDy†	**0.2355**	38.24	0.870	**0.5781**	**29.88**	**1.112**	**0.3359**	**33.76**	**0.349**
Pretrained ViT-B models									
DINOv3	0.2915	41.97	0.794	0.7996	26.93	1.207	0.6725	26.45	0.420
V-JEPA2.1	0.2747	**42.77**	**0.780**	0.7238	26.55	1.215	0.5821	**28.01**	0.408
PE Core	0.2706	41.35	0.795	0.7741	26.49	1.224	0.6602	27.45	0.413
PE Spatial	0.2726	42.06	0.793	0.7275	26.09	1.226	**0.5183**	27.39	0.413
TemDy*	**0.2531**	42.00	0.791	**0.6975**	**27.40**	**1.184**	0.5319	27.80	**0.406**

Table 7 shows joint improvements in state recovery and temporal discrimination on multiple datasets. On EPIC-KITCHENS-100, TemDy† achieves an NMSE of 0.5781, retrieval accuracy of 29.88%, and temporal MAE of 1.112 seconds. On Something-Something V2, the corresponding results are 0.3359, 33.76%, and 0.349 seconds. All six results are the best within the from-scratch group. TemDy* also leads the pretrained group on all three metrics on EPIC-KITCHENS-100.

These improvements are consistent with the complementary temporal objectives of TemDy. Historical reconstruction encourages DELTA to retain information about observed state transitions. Future prediction encourages it to capture information useful for subsequent evolution. Together, these objectives support both recovering target features and distinguishing temporally corresponding states.

Effect of the prediction head. Table 2 uses a standardized readout architecture and fitting protocol to reduce differences introduced by prediction heads when evaluating frozen representations. We additionally compare this readout with TemDy's native integral head on AgiBot.

Table 8: **Readout-head comparison with a frozen TemDy† encoder on AgiBot.** All variants use the same encoder checkpoint, input features, teacher targets, data splits, and fitting protocol. Avg. averages History and Future NMSE. Lower is better.

Prediction head	Initialization	History ↓	Current ↓	Future ↓	Avg. ↓
Standardized MLP readout	Random	0.2061	0.1917	0.2355	0.2208
Native integral head	Pretrained	**0.1724**	**0.1662**	**0.2068**	**0.1896**
Native integral head	Random	0.1864	0.1993	0.2092	0.1978

The pretrained integral head reduces average NMSE by 14.13% relative to the standardized MLP readout. With random initialization, the integral head also improves historical and future recovery, reducing average NMSE by 10.42%. These results support the effectiveness of the integral-head design beyond its pretrained initialization. Retaining the pretrained weights further improves all three tasks. Thus, the standardized readout in Table 2 controls the evaluation interface, while the native integral head achieves stronger state recovery in this comparison.

A.4 Contribution of DELTA to State Discrimination

We compare CLS alone with CLS combined with DELTA for TemDy* on AgiBot. Both conditions use the same candidate construction, fitting budget, and final-epoch evaluation.

Table 9: **Effect of DELTA on AgiBot state discrimination.** Entries are four-way accuracy in percent.

Representation	History ↑	Visible endpoint ↑	Future ↑
TemDy* CLS	47.97	53.34	41.80
TemDy* CLS + DELTA	**51.75**	**55.87**	**42.00**

Adding DELTA improves historical and visible-endpoint accuracy by 3.78 and 2.53 percentage points, respectively. Future accuracy increases by 0.20 percentage points. The larger observed-state gains suggest that DELTA provides complementary information for distinguishing states within the observed sequence.

B Training and Evaluation Details

This appendix describes the training configuration, evaluation protocols, and temporal predictor implementation, followed by sensitivity analyses and computational measurements.

B.1 Training Configuration

Model initialization. The pretrained-initialized variant, TemDy*, uses DINOv3 ViT-B weights with 16-pixel patches and 256×256 inputs. All four pretrained registers are retained, with the first register assigned to DELTA. Peak learning rates are 10^{-5} for the backbone, 10^{-3} for new modules, and 2.8×10^{-5} for the rotation modules, before backbone layer-wise and patch-embedding decay. These rates differ from the from-scratch recipe described below. Both recipes retain the full optimization schedule when evaluated at intermediate training steps.

The from-scratch development recipe starts from random ViT-B weights with 12 transformer blocks, width 768, 12 attention heads, and patch size 16. The complete optimization schedule spans

19,532 updates, with an effective batch of 512 and seed 17. Intermediate stopping points do not shorten this schedule. The learning-rate warmup lasts 1,953 updates, teacher-temperature warmup lasts 5,860 updates, and the self-distillation last layer is frozen for the first 195 updates. Student parameters, optimizer moments, and EMA parameters use FP32 storage, while supported encoder operations use BF16 autocast.

Training data and sampling. The realized training pool contains 10,000,000 records across 20 sources. After requiring valid observed frames, robot demonstrations contribute 3,348,144 records, first-person videos 3,949,063, other videos 2,284,536, and static-image preservation 418,257. The latter includes individual CO3D images, not multi-view sequences. Video records index temporal anchors, so different records can share an episode and overlapping frames. The sampler selects records without replacement within each source, then shuffles the combined pool. One complete pass consists of 19,531 batches of 512 and a final batch of 128 records.

Training uses up to four historical observations, including the visible deterministic future endpoint, and up to four future targets. Historical sampling prioritizes four-second bins and falls back to two-second or one-second bins when required by the recording. Future sampling selects one available target in each four-second interval up to 16 seconds. Actual timestamps enter the model. Image-only examples retain image SSL objectives but do not create temporal supervision from independently augmented static crops. If a batch contains no valid targets in one temporal query group, that group contributes zero to reconstruction. The other group retains its coefficient of $1/2$. This implementation convention extends the nonempty query averages in Equation 11 without changing their weights.

Optimization. The from-scratch AdamW peak learning rate is 2.83×10^{-3}, with $(\beta_1, \beta_2) = (0.9, 0.999)$ and weight decay increasing from 0.04 to 0.4. AdamW uses a minimum learning rate of 10^{-6}, gradient clipping at 3, layer-wise decay of 0.9, and a patch-embedding learning-rate multiplier of 0.2. The rotation submodules receive a further multiplier of 0.01. EMA momentum increases from 0.992 to 1, while the teacher temperature increases from 0.04 to 0.07. On eight H100 GPUs, the observed training-loop time averages 26.70 seconds per update, including data waiting and communication. Extrapolating to 19,532 updates gives approximately 1,159 GPU-hours for the complete schedule. This is an estimate rather than a completed full-training measurement and excludes checkpoint writes, initialization, evaluation, and paused intervals.

B.2 Evaluation and Ablation Protocols

Main-comparison protocol. Equal-step comparisons evaluate complete training recipes, while parameterization and objective ablations additionally match the relevant inputs and training settings. DINOv3 (Siméoni et al., 2025), V-JEPA2.1 (Mur-Labadia et al., 2026), and PE Core and PE Spatial (Bolya et al., 2025) are pretrained ViT-B references. Table 2 separates TemDy† (from scratch) and TemDy* (pretrained initialization). The filled Table 2 entries are equal-step training comparisons. NA denotes an unavailable metric or native interface. The Something-Something V2 protocol selects 20,000 fitting and 2,000 evaluation windows from distinct source videos, with one window per video. All ten model rows use this protocol in Table 2. The Something-Something V2 entries use framewise four-frame features, whereas the corresponding EPIC-KITCHENS-100 Image SSL, Multi-frame SSL, Direct Prediction, and TemDy entries use their native state representations. The JEPA entries use joint-video global features. These input representations are kept fixed across the three queries within each dataset.

For the temporal benchmark, PE Core-B and PE Spatial-B use their official loaders at native input resolutions of 224 and 512 pixels, respectively. Four observed-frame features are averaged before the common readout. Both use the same 21,803 training windows, 1,781 evaluation windows, shared teacher targets (DINOv3 ViT-L), future-change candidates, and fixed 20-epoch readout budget as the other references.

Objective-ablation protocol. The objective ablation in Table 5 trains three variants for 500 updates from the same DINOv3 initialization. All variants then receive identical 20-epoch readout fitting on 21,803 training windows and evaluation on 1,781 windows. Image objectives remain fixed, and removing one temporal loss does not rescale the other.

Each window contains four recorded-state queries: three historical times sampled from -12 to -1 seconds and the visible reference at zero. Four future queries are sampled from 1 to 16 seconds. The deterministic columns report the three historical queries, excluding the reference. Historical queries choose among the four recorded-state candidates, while future queries choose among the four future candidates. Candidate order is shuffled per example, and candidate timestamps are not classifier inputs. Future candidates are represented as differences from the common teacher's visible-reference feature. This target-side reference is not included in the prediction context, which consists of the evaluated encoder's CLS and DELTA features.

Each variant fits a width-256, one-hidden-layer candidate scorer with 727,297 trainable parameters. Fitting uses seed 17, batch size 64, AdamW with learning rate 10^{-3} and weight decay 10^{-4}, and gradient clipping at 3. The fixed twentieth epoch is evaluated without checkpoint selection on the evaluation split. Top-1 and time MAE use the same argmax decision, averaging neutrally over exactly tied maxima. For query i with true candidate index k_i, true time q_i, candidate times s_{ij}, and predicted probabilities p_{ij}, let $\mathcal{A}_i = \arg\max_j p_{ij}$. Then

$$\mathrm{TimeMAE} = \frac{1}{n}\sum_i \frac{1}{|\mathcal{A}_i|}\sum_{j\in\mathcal{A}_i} |s_{ij} - q_i|, \qquad \mathrm{PPL} = \exp\left(-\frac{1}{n}\sum_i \log p_{ik_i}\right). \tag{14}$$

Here n is the number of scored queries. MAE is measured in seconds. Uniform four-way predictions have Top-1 of 25% and perplexity 4.

Parameterization-ablation protocol. The parameterization comparison in Table 4 uses an identical visible-endpoint anchor architecture for all predictors. At zero time, the output is the anchor itself, and nonzero-time losses do not backpropagate through the anchor. Therefore, visible-state reconstruction is retained as an implementation diagnostic, not as a test of temporal parameterization.

This development experiment freezes the same DINOv3 ViT-B encoder and uses its independently encoded, unit-normalized CLS as every target. The 1,024 TRAIN and 256 EVAL windows have observed times $(-8, -5, -2, 0)$ seconds. Fitting supervises queries at $(-8, -2, 0, 1, 4)$. The intermediate queries $(-5, 2, 3)$ are held out from the fitting loss. All four predictors are trained for 20 epochs with batch size four, AdamW learning rate 10^{-3}, weight decay 10^{-4}, and seed 17. Table 4 averages historical errors over $(-8, -5, -2)$ and uncertain future errors over $(1, 2, 3, 4)$, including both supervised and withheld query times. Avg. is the equally weighted mean of History All and Future All, computed before rounding. Seen and Unseen describe their constituent query subsets and do not enter Avg. separately. Its NMSE uses one coordinate-wise population standard deviation fitted to the 5,120 supervised TRAIN targets. The visible endpoint at zero is available to every predictor. Copying that feature gives zero error by construction. Flow Matching uses velocity MSE and 16 Euler steps, whereas the other predictors fit cosine reconstruction.

B.3 Temporal Predictor Implementation

Temporal and spatial encoding. Three-axis rotary encoding allocates each 64-dimensional attention head to 16 temporal channels and 24 channels for each spatial axis. The global prefix tokens remain unrotated. For a batch of B windows and Q query times, velocity evaluation batches the BQK quadrature nodes while sharing the historical condition within each window.

Anchor and gradient routing. The visible-endpoint anchor uses Equation 8 without a time input. The shared CLS is detached in both the anchor residual map and the trajectory condition. DELTA remains differentiable in those paths. The rotation branch detaches the anchor, and a value-zero gradient-routing term restores its gradient only for queries at zero time. Thus, historical and future nonzero queries train the trajectory and DELTA without updating the anchor through that branch. The zero-time target is included in the observed reconstruction group. Therefore, it also supplies the learned starting state used by nonzero queries. Prediction errors propagate through the direction and angular-velocity networks and the encoder attention to update DELTA. Supervision at multiple query times encourages this pathway to encode variation across frames.

Angular velocity and numerical integration. The angular-velocity network combines the historical condition with the actual relative time and eight sine-cosine frequency pairs whose periods

range from 2 to 64 seconds. These are time features, not an assumed physical band limit. For a query τ, Gauss–Legendre nodes $x_k \in [-1, 1]$ with weights w_k are mapped to $\tau'_k = \tau(x_k + 1)/2$. The approximate accumulated angle is

$$\widetilde{\theta}_t(\tau) = \frac{\tau}{2} \sum_{k=1}^{K} w_k \, \omega_t(\tau'_k). \tag{15}$$

Equation 15 uses the same 64-node Gauss–Legendre rule for positive and negative intervals. For a general interval $[a, b]$, the node mapping is $s_k = (a + b)/2 + (b - a)x_k/2$ and the weighted sum is multiplied by $(b - a)/2$. Nodes and weights are stored as full-precision constants rather than low-precision model buffers. Rate evaluation, quadrature, and rotation use FP32 arithmetic. The tangent direction is reprojected before normalization to reduce numerical residuals. A degenerate direction is marked invalid and falls back to the visible-endpoint anchor, with no angular transformation applied to that query.

B.4 Hyperparameter Sensitivity

We examine sensitivity to loss weights, numerical integration resolution, and observation count. These experiments assess supervision balance, numerical accuracy and cost, and the benefit of additional observations, respectively.

Loss weights: setup. Let α denote the deterministic mixture weight in $\alpha\mathcal{L}_{\text{det}} + (1 - \alpha)\mathcal{L}_{\text{unc}}$, and let λ_Δ denote the DELTA consistency weight. The default is $(\lambda_{\text{rec}}, \alpha, \lambda_\Delta) = (1, 0.5, 1)$. Eight continuations start from the same pretrained-initialized TemDy checkpoint and run for 500 additional updates. We vary reconstruction weight, deterministic-target mixture weight, and DELTA consistency weight one at a time. All final checkpoints use the same 21,803 fitting windows, 1,781 development windows, fixed scratch-TemDy teacher targets, and fresh 20-epoch readouts with seed 17. These are shared-target readout results, not native-head errors.

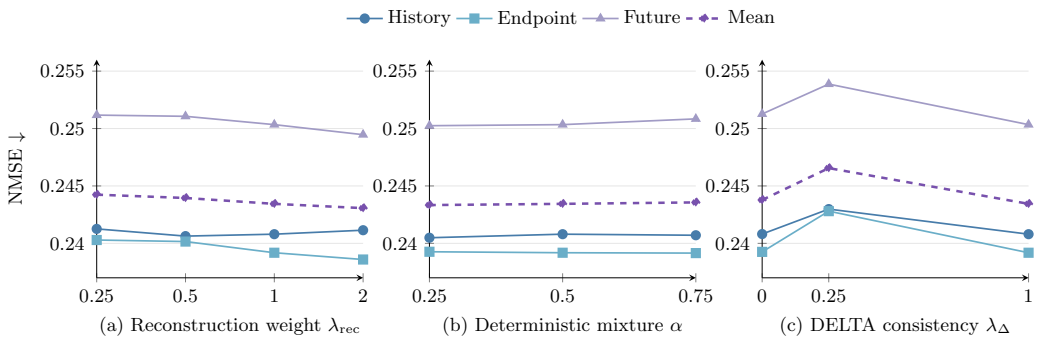


Figure 7: **Loss-weight sensitivity.** One-at-a-time sweeps of reconstruction weight, deterministic-target mixture, and DELTA consistency weight. Curves show History, Endpoint, Future, and their mean NMSE. Other parameters remain at their default values. The reconstruction-weight axis is logarithmic. Lines connect the evaluated settings.

Loss weights: results. Figure 7 reports NMSE across the evaluated settings. The mean ranges from 0.243067 to 0.246553, compared with 0.243440 for the default. The narrow range suggests limited sensitivity to these weight choices within the tested configurations. This single-seed sweep does not establish a robust optimum.

Quadrature nodes. At $K = 64$, maximum angle differences remain below 2.4×10^{-7}, while prediction-head latency is approximately 2.5 times lower than at $K = 256$. These results support using 64 nodes for the tested models, balancing numerical accuracy and inference cost.

Table 10: **Numerical accuracy and prediction-head latency at different quadrature resolutions.** Angle differences use $K = 256$ as the reference. Latency uses batch size 32 and excludes encoder inference.

Measurement	K	From scratch	Pretrained initialization
Maximum angle difference	32	1.04×10^{-4}	2.83×10^{-5}
Maximum angle difference	64	2.38×10^{-7}	8.94×10^{-8}
Head latency (ms)	64	4.54	4.56
Head latency (ms)	256	11.21	11.25

Table 11: **Computational efficiency.**

Method	**Params (M)**	**Encode (ms)**	**Inf. peak (GiB)**	**F+B (GFLOPs)**
DINOv3	85.67	10.01	0.408	429.45
V-JEPA2.1	86.83	23.37	0.408	221.44
PE Core	93.67	9.81	0.434	426.33
TemDy*	85.67	8.34	0.412	472.78

Observation count. On the same 1,781 development windows, increasing observations from one to four reduces historical NMSE from 1.305651 to 0.651432. This reduction shows the benefit of additional observed context for historical state recovery.

B.5 Computational Efficiency

Table 11 measures four 224×224 frames on the same idle H100. TemDy* encodes this input in 8.34 ms, compared with 10.01 ms for DINOv3, with nearly identical parameter counts and peak memory. Nevertheless, its joint temporal attention raises the counted forward–backward work from 429.45 to 472.78 GFLOPs. Conversely, V-JEPA2.1 has a lower operation count but higher latency, illustrating that arithmetic cost alone does not predict execution time. These measurements exclude temporal prediction heads and full training objectives.

Measurement protocol. Table 11 uses the same idle H100 80 GB for every model, FP32 parameter storage, BF16 autocast, and inference mode. The first TRAIN clip supplies four distinct images at $[-9, -5, -3, 0]$ seconds, resized and center-cropped to 224×224 before timing. Image models encode a batch of four independent frames, while video models retain their native joint encoding. PE Core represents the PE family in this efficiency comparison. V-JEPA2.1 uses official positional interpolation from its pretrained resolution of 384 to the common input resolution.

Each measurement uses five warmup forwards and thirty individually synchronized CUDA-event samples. Data loading, preprocessing, and host transfer are excluded. Parameters and peak allocated memory describe one visual encoder, including its native pooling or projection but excluding teachers, training heads, and the temporal decoder. TemDy* uses its EMA encoder with four retained registers. Its 8.89M-parameter temporal decoder is excluded from every entry in this encoder-cost table.

The reported medians retain all measured samples, including long tails. Peak allocated memory includes encoder weights, GPU inputs, and forward temporaries, but not other processes or the CUDA context. FLOPs count matrix multiplication, convolution, and attention, with one multiply–add counted as two operations. Normalization, activation, rotary position, and elementwise operations are excluded. These subtotals do not represent the complete training objective. These measurements describe only the specified input size and execution environment, not end-to-end prediction throughput or the cost of reproducing each model's original training procedure.

The F+B column specifies one encoder forward and backward pass on the same four-frame input, with parameter gradients enabled and no optimizer update. Teachers, temporal predictors, and auxiliary readouts are excluded for every model. This measures encoder differentiation cost rather than the cost of reproducing each method's pretraining recipe. Per-step FLOPs include one actual forward and backward trace, not a fixed multiple of inference FLOPs. A fixed non-trainable random

projection of the native output supplies a squared scalar loss for backpropagation. All active gradients are finite, and encoder parameters remain unchanged afterward.

Operation counts request math attention on CUDA, separately from native kernels used for timing. V-JEPA2.1 overrides this request with CUDA flash attention, whose forward and backward operators are covered by the counter. Therefore, all four entries include attention matrix products, while the elementwise operations listed above remain excluded consistently.

Separately, the full-schedule estimate in Setup uses the original TemDy log for updates 1–3,000, without discarding slow batches. The sum of per-update durations is 80,098.404 s, giving a mean of 26.6995 s. At global batch 512, throughput is $512/26.6995 = 19.18$ training windows per second. These durations include data waiting and the training loop, but exclude checkpointing, evaluation, and pauses between runs. Extrapolating the same rate to 19,532 updates on eight GPUs gives 1,158.88 GPU-hours. The estimate is rounded to 1,159 and is not the measured cost of a completed epoch. No training-speed ranking against the pretrained references is implied, since their original pretraining traces are unavailable in this measurement.

B.6 Structural Objective

The structural objective groups the existing visual and register losses without changing their optimization weights:

$$\begin{aligned}\mathcal{L}_{\text{struct}} = \mathcal{L}_{\text{DINO}} + \mathcal{L}_{\text{iBOT}} + 0.1\mathcal{L}_{\text{KoLeo}} + \lambda_{\Delta}\mathcal{L}_{\Delta} \\ + \lambda_{\text{var}}\mathcal{L}_{\text{var}} + \lambda_{\text{cov}}\mathcal{L}_{\text{cov}}.\end{aligned} \tag{16}$$

DINO (Caron et al., 2021) aligns student and teacher global distributions, iBOT (Zhou et al., 2021) aligns masked patch predictions with teacher patch targets, and KoLeo (Oquab et al., 2023) regularizes global-feature dispersion. The remaining terms align DELTA registers and regularize their batch statistics, following the variance and covariance principles of VICReg (Bardes et al., 2021). These structural terms accompany the temporally indexed frame targets in Equation 12 rather than replacing temporal reconstruction. The reconstruction and DELTA-consistency weights λ_{rec} and λ_{Δ} are both one. λ_{var} and λ_{cov} weight the variance and covariance penalties, respectively. The DELTA teacher comes from an unmasked EMA forward of the same augmented observed window, separate from the clean independent frame targets. For a batch of B windows with student and teacher registers $\boldsymbol{\delta}_i$ and $\bar{\boldsymbol{\delta}}_i$, indexed by i, the consistency term is

$$\mathcal{L}_{\Delta} = \frac{1}{B}\sum_{i=1}^{B}\left(1 - \frac{\boldsymbol{\delta}_i^{\top}\,\text{sg}(\bar{\boldsymbol{\delta}}_i)}{\|\boldsymbol{\delta}_i\|_2\|\bar{\boldsymbol{\delta}}_i\|_2}\right). \tag{17}$$

For a local batch of $B \geq 2$ registers, write $v_{ik} = \delta_{ik} - B^{-1}\sum_j \delta_{jk}$ and define

$$s_k = \sqrt{B^{-1}\sum_i v_{ik}^2 + 10^{-4}}, \qquad \mathcal{L}_{\text{var}} = \frac{1}{d}\sum_k \max(0, 1 - s_k), \tag{18}$$

$$C_{k\ell} = \frac{1}{B}\sum_i \frac{v_{ik}}{\max(s_k, 10^{-2})}\frac{v_{i\ell}}{\max(s_\ell, 10^{-2})}, \qquad \mathcal{L}_{\text{cov}} = \frac{\sum_{k\neq\ell} C_{k\ell}^2}{d(d-1)}. \tag{19}$$

Both regularizers contribute zero for $B < 2$, and their weights are 0.1. Here k and ℓ index the d register coordinates, v_{ik} is a centered coordinate, s_k its stabilized standard deviation, and $C_{k\ell}$ the covariance of the standardized coordinates. The covariance penalty operates on standardized centered registers. The time-discrimination cross-entropy is logged for diagnosis but has zero training weight in this recipe. Deterministic reconstruction $\mathcal{L}_{\text{det}}$ is the historical-and-reference loss in the implementation. It and the uncertain-future loss $\mathcal{L}_{\text{unc}}$ are separately averaged over globally valid queries before equal group weighting, as in Equation 11. The notation change does not alter the checkpoint's training objective or the visibility of any target frame.

B.7 Attention Mask

Attention is causal between frames, with global access reserved for CLS and DELTA. A patch query in P_{t_i} can attend to patches in P_{t_j} only when $t_j \leq t_i$, including all patches within its

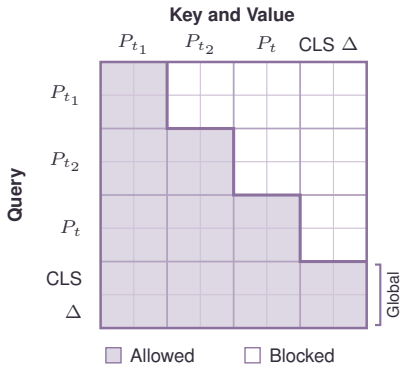


Figure 8: **Attention mask.** Rows are queries and columns are keys. Filled cells permit attention. Patches attend within their own and earlier frames but cannot read CLS or DELTA. Both global tokens can attend to every observed patch and each other through global attention.

own frame. Patch queries cannot read either shared prefix. In contrast, CLS and DELTA can attend to every observed patch and both prefixes. Blocking patch access to the prefixes prevents later-frame information from returning to earlier patches through these global summaries across transformer layers. Future frames are absent from the student input. Masked-patch training replaces token contents without changing this visibility pattern.

Figure 8 shows two representative patches per frame with $t_1 < t_2 < t$. Rows are queries and columns are keys. For display, the global tokens appear after the patch groups on both axes, without changing their prefix positions in the model. Filled blocks allow attention and white blocks are masked. The stepped boundary follows frame groups, while the bottom two rows span all keys. In the additive mask, allowed entries are zero and blocked entries are $-\infty$. These entries specify connectivity, not learned attention weights.

B.8 Comparison and Transfer Settings

Physion. We use the official 5,608 training and 1,035 test observations. Table 3 reports AUC for Support, Drop, Link, and Collide. Standardization and five-fold logistic-regression selection use training data only. Image SSL takes the cutoff frame, Multi-frame SSL jointly encodes four frames, and pretrained image encoders average four framewise global features. V-JEPA2.1 averages its final joint spatiotemporal tokens without a learned pooling head or predictor. JEPA SSL uses four 112-pixel observations and averages the frozen target encoder's final tokens. Direct Prediction and both TemDy variants use four ordered 224-pixel observations within the official cutoff, with timestamps computed under 100 FPS. Their frozen EMA features are read out by the same logistic protocol: 768D CLS for Direct Prediction and 1536D CLS+DELTA for TemDy, without using the prediction head.

IntPhys 2. We evaluate all 1,012 Main videos, forming 506 possible–impossible pairs across 253 scenes. Each sliding window contains four observed frames and twelve future targets, sampled every ten source frames using decoded timestamps. The complete EMA model predicts its own teacher's future CLS features. Table 3 uses unit-normalized features, averages absolute prediction error over query times and feature dimensions, and averages the window scores to obtain each video's surprise. A pair is correct when the possible video has lower surprise, with ties counted as incorrect. All 25,812 windows are evaluated without skipped videos. For encoders without a compatible native predictor, we freeze the encoder and fit a prediction head outside IntPhys 2 using the Something-Something V2 fitting split. An initial engineering check processed one Main scene without selecting a scoring rule, followed by a Debug throughput check before the complete Main evaluation. No predictor or classifier is fitted on Main.

C Theoretical Derivations

C.1 Representational capacity of a rotation plane

Proposition 3 (Pairwise representation and local approximation)**.** *For $d \geq 2$ and any unit vectors $\mathbf{c}, \mathbf{z} \in \mathbb{R}^d$, there exist a unit $\mathbf{u} \perp \mathbf{c}$ and $\alpha \in [0, \pi]$ such that $\mathbf{z} = \cos\alpha\,\mathbf{c} + \sin\alpha\,\mathbf{u}$. Further, let $\boldsymbol{\gamma} : (-\eta, \eta) \to \mathbb{S}^{d-1}$ be twice continuously differentiable, with $\boldsymbol{\gamma}(0) = \mathbf{c}$, $v = \|\boldsymbol{\gamma}'(0)\|_2 \geq 0$, and $\|\boldsymbol{\gamma}''(\tau')\|_2 \leq M$ on this interval. For $\mathbf{u} = \boldsymbol{\gamma}'(0)/v$ when $v > 0$ (any unit $\mathbf{u} \perp \mathbf{c}$ when $v = 0$) and $\mathbf{r}(\tau) = \cos(v\tau)\mathbf{c} + \sin(v\tau)\mathbf{u}$,*

$$\|\boldsymbol{\gamma}(\tau) - \mathbf{r}(\tau)\|_2 \leq \tfrac{1}{2}(M + v^2)\tau^2, \qquad |\tau| < \eta. \tag{20}$$

Proof. For noncollinear endpoints, set

$$\alpha = \arccos(\mathbf{c}^\top \mathbf{z}), \qquad \mathbf{u} = \frac{\mathbf{z} - \cos\alpha\,\mathbf{c}}{\sin\alpha}. \tag{21}$$

Direct substitution gives $\mathbf{c}^\top \mathbf{u} = 0$, $\|\mathbf{u}\|_2 = 1$, and the asserted representation. If $\mathbf{z} = \mathbf{c}$ or $\mathbf{z} = -\mathbf{c}$, choose any unit $\mathbf{u} \perp \mathbf{c}$ and use $\alpha = 0$ or π, respectively. Such a direction exists because $d \geq 2$.

For the local statement with $v > 0$, differentiating $\|\boldsymbol{\gamma}(\tau')\|_2^2 = 1$ at zero gives $\mathbf{c}^\top \boldsymbol{\gamma}'(0) = 0$. Thus, the chosen $\mathbf{u}$ is unit and orthogonal to $\mathbf{c}$, with $\mathbf{r}(0) = \boldsymbol{\gamma}(0)$ and $\mathbf{r}'(0) = \boldsymbol{\gamma}'(0)$. Also $\|\mathbf{r}''(\tau')\|_2 = v^2$. Taylor's integral remainder applied to each curve gives errors at most $M\tau^2/2$ and $v^2\tau^2/2$ relative to their shared linear approximation. The triangle inequality yields Equation 20 for either sign of τ. When $v = 0$, choose any unit $\mathbf{u} \perp \mathbf{c}$ and set $\mathbf{r}(\tau) = \mathbf{c}$. Taylor's remainder then gives $\|\boldsymbol{\gamma}(\tau) - \mathbf{c}\|_2 \leq M\tau^2/2$, which is the same claimed bound. □

First-order dynamic information and a static-anchor comparison. Under the same assumptions, consider $v > 0$ and $\mathbf{u} = \boldsymbol{\gamma}'(0)/v$. Differentiating the unit-norm constraint twice at zero yields

$$\mathbf{c}^\top \boldsymbol{\gamma}'(0) = 0, \qquad \mathbf{c}^\top \boldsymbol{\gamma}''(0) = -v^2. \tag{22}$$

The second-order Taylor expansion of the C^2 trajectory therefore gives

$$1 - \mathbf{c}^\top \boldsymbol{\gamma}(\tau) = \tfrac{1}{2}v^2\tau^2 + o(\tau^2), \qquad \mathbf{u}^\top \boldsymbol{\gamma}(\tau) = v\tau + O(\tau^2). \tag{23}$$

Thus, cosine similarity alone has no first-order term, whereas the tangent component encodes the signed local progression. In the constructed rotation, this component is $\sin(v\tau) = v\tau + O(\tau^3)$, so rotation retains the first-order transition while staying on the unit sphere. Writing $\boldsymbol{\gamma}(\tau) - \mathbf{c} = \tau\boldsymbol{\gamma}'(0) + \boldsymbol{\rho}(\tau)$ with $\|\boldsymbol{\rho}(\tau)\|_2 \leq M\tau^2/2$, the reverse triangle inequality gives

$$\big|\|\boldsymbol{\gamma}(\tau) - \mathbf{c}\|_2 - v|\tau|\big| \leq \tfrac{1}{2}M\tau^2. \tag{24}$$

Together with Eq. 20, this implies

$$\lim_{\tau \to 0,\, \tau \neq 0} \frac{\|\boldsymbol{\gamma}(\tau) - \mathbf{r}(\tau)\|_2}{\|\boldsymbol{\gamma}(\tau) - \mathbf{c}\|_2} = 0. \tag{25}$$

The denominator is positive for sufficiently small nonzero τ because $v > 0$. Hence, a rotation matched to the local velocity has asymptotically smaller state error than copying the anchor. If $v = 0$, both errors can be second order and no strict comparison is asserted. This is a local approximation result with a matched tangent, not a guarantee that a learned predictor outperforms every unrestricted state predictor. These local approximation arguments establish Proposition 1.

The first result shows that a two-dimensional plane is sufficient to connect any given pair of normalized states. The second shows that a plane spanned by state and tangent can approximate local evolution with quadratic error. This constant-rate construction is included in the time-varying family of Equation 1. Pairwise constructions may choose different planes, whereas TemDy learns one plane shared across queries. Exact representation of an entire trajectory in that plane requires the trajectory to remain in its intersection with the unit sphere. Neither result guarantees that a learned plane captures long-horizon changes.

Domain of normalization. Unit normalization $\mathrm{norm}_2(\mathbf{x}) = \mathbf{x}/\|\mathbf{x}\|_2$ is defined for finite, nonzero vectors. Therefore, the normalized CLS condition, teacher targets, and residual anchor require nonzero inputs to this operation. In particular, the residual vector inside Equation 8 must be nonzero before defining $\mathbf{c}_t$. In numerical evaluation, this requirement is checked against a positive tolerance. Nonfinite or near-zero anchor inputs are rejected rather than interpreted as unit states. Given a valid unit anchor, a nonzero tangent projection defines the rotation branch. A degenerate tangent is handled separately below.

Geometry preservation and rate consistency. Fix a historical condition and omit the subscript t. Assume $\|\mathbf{c}\|_2 = \|\mathbf{u}\|_2 = 1$, $\mathbf{c}^\top \mathbf{u} = 0$, and continuous ω on an interval containing zero. Then $\theta(\tau) = \int_0^\tau \omega(\tau')\, d\tau'$ is differentiable, with $\theta'(\tau) = \omega(\tau)$. Substituting $\mathbf{z}(\tau) = \cos\theta(\tau)\mathbf{c} + \sin\theta(\tau)\mathbf{u}$ gives

$$\|\mathbf{z}(\tau)\|_2^2 = \cos^2\theta(\tau) + \sin^2\theta(\tau) = 1, \tag{26}$$

$$\mathbf{z}'(\tau) = \omega(\tau)[-\sin\theta(\tau)\mathbf{c} + \cos\theta(\tau)\mathbf{u}]. \tag{27}$$

The bracketed vector has unit norm by the same orthogonality calculation, so $\|\mathbf{z}'(\tau)\|_2 = |\omega(\tau)|$, including times where $\omega(\tau) = 0$. This establishes norm preservation and rate consistency.

For any a, b in the interval, expanding the inner product yields

$$\mathbf{z}(a)^\top \mathbf{z}(b) = \cos\theta(a)\cos\theta(b) + \sin\theta(a)\sin\theta(b) \tag{28}$$

$$= \cos(\theta(b) - \theta(a)) = \cos\left(\int_a^b \omega(\tau')\, d\tau'\right). \tag{29}$$

The signed angle describes net evolution, while path length is $\int_a^b |\omega(\tau')|\, d\tau'$ for $a \le b$. These differ when the angular velocity reverses sign. The parameterization does not require positive velocity or guarantee distinct states at all distinct query times.

Degenerate tangent directions. The rotation construction requires a nonzero tangent projection $\mathbf{a} - (\mathbf{a}^\top \mathbf{c})\mathbf{c}$ so that its normalization defines $\mathbf{u}$. If this direction is degenerate, the implementation returns the unit anchor, $\mathbf{z}(\tau) = \mathbf{c}$, without applying the predicted angle. This constant branch preserves the unit norm and has $\|d\mathbf{z}/d\tau\|_2 = 0$. The rate identity derived above applies to the nondegenerate rotation branch. For the fallback it holds with an effective angular velocity of zero, not necessarily with the raw output of the velocity head.

C.2 Interval composition

Let $\mathbf{A} = \mathbf{u}\mathbf{c}^\top - \mathbf{c}\mathbf{u}^\top$. Then $\mathbf{A}^\top = -\mathbf{A}$, $\mathbf{A}\mathbf{c} = \mathbf{u}$, and $\mathbf{A}\mathbf{u} = -\mathbf{c}$. Its exponential rotates the span of $\mathbf{c}, \mathbf{u}$ and leaves the orthogonal complement unchanged. Therefore, writing $\Theta(a, b) = \int_a^b \omega(\tau')\, d\tau'$ gives $R(b, a) = \exp(\mathbf{A}\Theta(a, b))$ and $R(b, a)\mathbf{z}(a) = \mathbf{z}(b)$.

For arbitrary endpoints a, b, c, signed integral additivity gives $\Theta(a, c) = \Theta(a, b) + \Theta(b, c)$. Both matrix exponents are scalar multiples of the same $\mathbf{A}$, so they commute and

$$R(c, b)R(b, a) = \exp(\mathbf{A}\Theta(b, c))\exp(\mathbf{A}\Theta(a, b)) \tag{30}$$

$$= \exp(\mathbf{A}[\Theta(a, b) + \Theta(b, c)]) = R(c, a). \tag{31}$$

This proves Proposition 2. Repeating the identity gives composition over any finite partition, including unequal intervals. Every factor must retain the same historical condition and generator. Re-encoding a different history changes those objects and is not covered by this identity.

C.3 Numerical integration and time perturbations

If $|\omega(\tau')| \le M$ on the interval, the fundamental theorem of calculus and the rate identity in Appendix C.1 give

$$\|\mathbf{z}(b) - \mathbf{z}(a)\|_2 \le \int_{\min(a,b)}^{\max(a,b)} |\omega(\tau')|\, d\tau' \le M|b - a|. \tag{32}$$

If quadrature introduces angle error $|\widetilde{\theta}(\tau) - \theta(\tau)| \leq \epsilon$, the corresponding state error for the quadrature-based prediction $\mathbf{z}^{\mathrm{quad}}$ satisfies

$$\|\mathbf{z}^{\mathrm{quad}}(\tau) - \mathbf{z}(\tau)\|_2 = 2\left|\sin\frac{\widetilde{\theta}(\tau) - \theta(\tau)}{2}\right| \leq \epsilon. \tag{33}$$

With the same plane, the numerical discrepancy between direct integration over $[a, c]$ and composition over $[a, b]$ and $[b, c]$ is bounded by the sum of their absolute angle-integration errors. These bounds describe how time and angle errors affect the representation. They do not bound prediction error against an unknown physical trajectory.

D Comparison of Continuous Dynamics Parameterizations

D.1 Relation to Neural ODEs and Flow Matching

Neural ODEs (Chen et al., 2018) parameterize continuous state evolution, while latent ODEs (Rubanova et al., 2019) model irregularly sampled sequences. Conditional flow matching (Lipman et al., 2022) learns vector fields by regressing conditional path velocities.

TemDy instead restricts evolution to a history-conditioned rotation plane and learns a scalar angular-velocity function over physical time:

$$\frac{d\mathbf{z}_t(\tau)}{d\tau} = \omega_t(\tau)\mathbf{A}_t\mathbf{z}_t(\tau). \tag{34}$$

With a fixed historical condition, the generator $\mathbf{A}_t$ remains constant. Prediction therefore requires scalar quadrature rather than sequential updates of a general high-dimensional state. This construction preserves unit norm and gives angular velocity an explicit interpretation as latent evolution speed.

In the flow-matching comparator of Table 4, integration proceeds over a dimensionless flow coordinate conditioned on the requested physical time. This coordinate differs from the elapsed time integrated by TemDy. The comparator uses conditional velocity regression and 16 Euler steps, whereas TemDy uses cosine reconstruction supervision.

These distinctions describe the compared formulations, not universal limitations of flow matching. General ODE flows also compose, and flow matching can use structured fields. TemDy trades the expressiveness of a general state-dependent field for a structured rotational trajectory.

D.2 Angular-Velocity Integration versus Direct Angle Prediction

The same rotational trajectory can be parameterized by directly predicting its angle:

$$\theta_t^{\mathrm{direct}}(\tau) = g_\phi(\xi_t, \tau) - g_\phi(\xi_t, 0), \tag{35}$$

where ξ_t denotes the history condition. Both formulations recover the anchor at zero time, preserve unit norm, and support continuous queries. Direct angle differences also define compositionally consistent interval transformations. These properties therefore do not distinguish integration from direct angle prediction.

The distinction lies in what the network learns. Direct angle prediction learns accumulated displacement from the anchor, whereas integration learns local evolution rates and accumulates them over time. Multiple supervised queries constrain integrals of the same rate function over overlapping intervals. This provides a rate-based inductive bias that may help organize evolution between sparsely supervised timestamps.

Explicit angular velocity also makes acceleration, slowing, and reversal interpretable within the learned plane.

We compare the two formulations using the same anchor, plane construction, scalar-network architecture, initialization, supervision, and training budget under the Table 4 protocol.

Table 12: **Direct angle prediction versus angular-velocity integration at held-out query times.** Lower NMSE is better.

Method	History: unseen times ↓	Future: unseen times ↓
Direct angle	0.491315	0.445475
Angular-velocity integration	**0.460708**	**0.421449**

E Additional Results

The following diagnostics examine native prediction, DELTA conditioning, observation timing, and feature normalization. They distinguish state recovery from precise future-time correspondence and keep evaluation protocols separate.

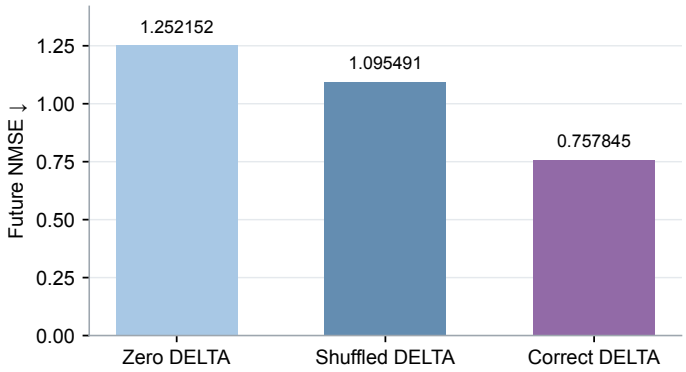


Figure 9: **Native DELTA intervention.** Frozen from-scratch TemDy evaluated on 1,781 AgiBot windows using its original integral head and its own EMA targets.

Native DELTA intervention. Figure 9 holds CLS, query times, examples, predictor weights, and the NMSE definition fixed without refitting a head. The donor DELTA comes from a different episode in the same task and split under a fixed seed. Correct DELTA reduces future NMSE by 0.494307 relative to zero DELTA and by 0.337646 relative to shuffled DELTA. Replacement precedes anchor, plane, and angular-velocity prediction, so this tests their joint dependence on DELTA. These interventions differ from the independently trained objective variants in Table 5.

Table 13: **Fixed-anchor DELTA intervention.** TemDy evaluated on 1,024 training-pool windows. The future subset contains 2,707 valid queries. Entries are $1 - \cos$ errors, with lower values preferred.

Trajectory-conditioning DELTA	History	Future
Correct	0.3757	0.5107
Donor	0.6467	0.6233

Fixed-anchor DELTA dependence. Table 13 fixes the visible-endpoint anchor, CLS, query times, model weights, and EMA targets while replacing only the trajectory-conditioning DELTA with a fixed donor permutation. The lower historical and future errors show that DELTA remains useful when anchor changes are excluded. The intervention changes both plane and rate conditioning, rather than isolating angular velocity alone.

Table 14: **Feature direction and norm.** Matched frozen-encoder readouts on 21,803 fitting and 1,781 evaluation windows from AgiBot, using TemDy's own EMA targets. Entries are NMSE, with lower values preferred.

Readout input	History	Current	Future
Norm only	0.918310	0.905325	0.919167
Direction only	0.219563	0.201971	0.247543
Direction and norm	0.219263	0.202120	0.247562

Feature direction and norm. Table 14 uses frozen CLS and DELTA, matched padded input dimensions, width-256 MLPs, query times, targets, and 20-epoch fitting with seed 17. Norm statistics use training data only. Direction substantially outperforms norms alone. Against the joint input, removing norms lowers Current and Future NMSE by 0.000149 and 0.000018, while increasing History NMSE by 0.000301. These small differences support comparable recovery with direction-only inputs, not a consistent improvement. This supports our normalization choice for unit-normalized target recovery, without implying that magnitudes lack information for other tasks.

Table 15: **Prediction beyond static copying.** Frozen encoders on 1,781 AgiBot evaluation windows, each using its own EMA targets. Static copies the current target feature. Reduction is the relative NMSE decrease from that model's static reference, not a cross-space ranking.

System	History			Future		
	Static	Prediction	Reduction (%)	Static	Prediction	Reduction (%)
Image SSL	0.456716	0.392579	14.04	0.369199	0.331770	10.14
JEPA SSL	0.241492	0.142993	40.79	0.182384	0.198180	−8.66
TemDy	0.325628	0.214306	34.19	0.281447	0.240777	14.45

System prediction and static persistence. Table 15 compares three frozen-encoder systems. Image SSL uses the current image, while JEPA SSL and TemDy use four observed frames. Image SSL and JEPA SSL fit fresh width-256 MLP heads, while TemDy adapts its existing integral head without changing its architecture or initialization. All heads train for 20 epochs on the same 21,803 AgiBot windows, using seed 17, batch size 16, and AdamW with learning rate and weight decay both 10^{-4}. Each final-epoch prediction is compared with static copying in its own EMA target space.

All three systems improve History over their static reference. For Future, Image SSL and TemDy reduce NMSE by 0.037429 and 0.040670, respectively, whereas JEPA SSL increases it by 0.015797. TemDy's relative reductions are 34.19% for History and 14.45% for Future, supporting prediction beyond static state persistence. Different teacher spaces and input capabilities prevent interpreting the raw errors or reduction ratios as a controlled cross-model ranking. Current is omitted because static copying directly returns its target and has zero error.

The shared-target comparison in Table 2 evaluates recovery against common targets. Table 15 adds evidence beyond static copying, while Tables 16–17 test dependence on dynamic conditions. Together, these comparisons separate relative recovery performance, non-static prediction, and temporal conditioning.

Table 16: **Fixed-anchor DELTA after head adaptation.** The same adapted TemDy head and 1,781 evaluation windows as Table 15. Entries are NMSE.

DELTA condition	History	Current	Future
Correct	0.214306	0.214023	0.240777
Zero, fixed anchor	1.495871	0.214023	0.458076
Shuffled, fixed anchor	0.288970	0.214023	0.293726

DELTA conditions the trajectory beyond the anchor. Table 16 keeps the anchor, CLS, query times, targets, and fitted weights fixed, without refitting under interventions. Correct DELTA reduces History and Future NMSE relative to shuffled DELTA by 0.074664 (25.84%) and 0.052948 (18.03%), respectively. Current NMSE is unchanged, consistent with retaining the same anchor. The remaining differences support DELTA's role in conditioning the evolution trajectory beyond endpoint recovery.

Table 17: **Observation-time intervals after head adaptation.** Matched full-model evaluation on 1,781 AgiBot windows with the adapted native head. Timestamps are in seconds. Entries are NMSE.

Observation timestamps	History	Current	Future
Actual $(-8, -5, -2, 0)$	0.214316	0.214014	0.240754
Uniform $(-8, -\frac{16}{3}, -\frac{8}{3}, 0)$	0.227961	0.235487	0.251486
Reversed gaps $(-8, -6, -3, 0)$	0.230662	0.246711	0.257571

Dependence on observation intervals. Table 17 changes only observation timestamps, retaining the images, chronological frame order, target queries, targets, and model weights. Both replacement schedules preserve the temporal endpoints. Reversed gaps reorder the inter-frame durations, not the frames. Actual timestamps reduce Future NMSE relative to uniform and reversed gaps by 0.010732 (4.27%) and 0.016816 (6.53%), with lower errors in the other two tasks as well. These results support the use of observation intervals beyond frame order in the adapted system.

F Training Dynamics

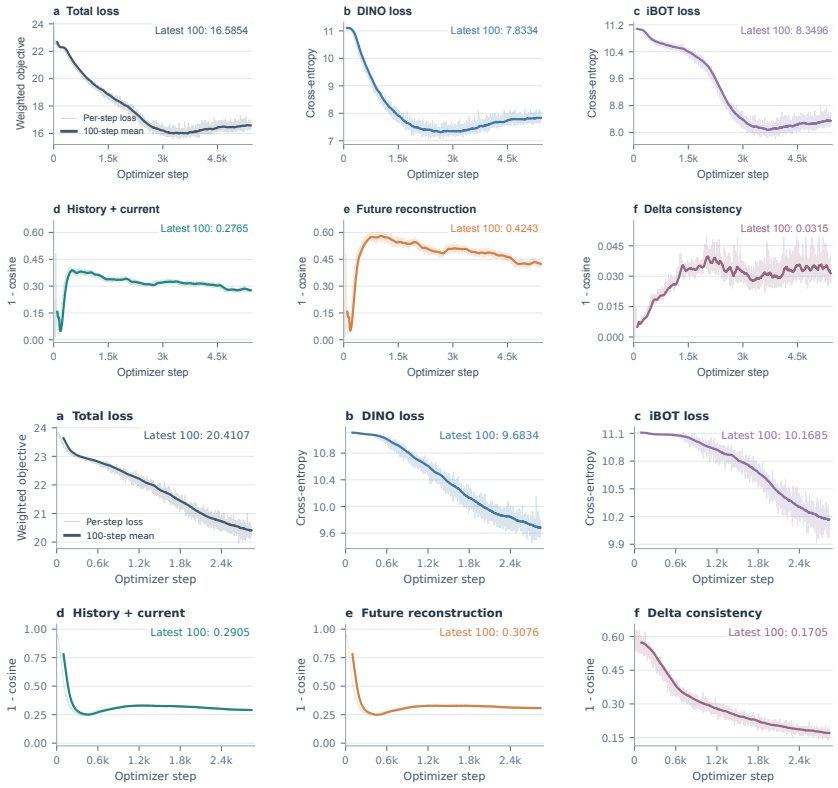


Figure 10: **Training dynamics.** From-scratch TemDy (top, 5,412 updates) and pretrained-initialized TemDy (bottom, 2,837 updates). Faint traces show per-update losses, and solid curves show trailing 100-step means.

Table 18: **Training hyperparameters for Figure 10.** Peak learning rates precede layer-wise and patch-embedding scaling. Further optimization settings appear in Appendix B.

Hyperparameter	From scratch	Pretrained initialization
Initialization	Random	DINOv3
Encoder	ViT-B, patch size 16	ViT-B, patch size 16
Optimizer	AdamW	AdamW
Global batch size	512	512
Peak backbone learning rate	2.83×10^{-3}	10^{-5}
Peak rotation-module learning rate	2.83×10^{-5}	2.8×10^{-5}
Reconstruction weight λ_{rec}	1	1
Deterministic mixture α	0.5	0.5
DELTA consistency weight λ_{Δ}	1	1
DINO and iBOT weights	1 each	1 each
KoLeo weight	0.1	0.1
Quadrature nodes K	64	64

Table 18 summarizes the training settings. The curves separate image and temporal objectives. Reconstruction losses compare predictions with evolving EMA targets, and historical supervision includes the visible endpoint. These within-run traces describe optimization, not cross-model capability gains.