CONFAL-WM: CONFIDENCE-GUIDED ACTIVE LEARNING FOR ACTION-CONDITIONED WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Action-conditioned world models have become an important foundation for embodied prediction, planning, and synthetic data generation, but their errors under new task and scene distributions are often concentrated in localized spatiotemporal regions such as robot arms, manipulated objects, contact areas, and occluded objects. This paper presents **ConfAL-WM**, a confidence-guided active learning framework for post-training embodied world models. Building upon EnerVerse-AC (EVAC), we attach a lightweight confidence probe to UNet decoder features and predict dense confidence maps in the latent space. These maps are aggregated into task-, frame-, and patch-level scores, enabling data-budget allocation and localized training enhancement. Our pipeline trains the probe and warms up EVAC on a small target-domain subset. EVAC-v1 then supplies task-level acquisition and optional frame/patch weighting signals; all selected-data models are initialized from the original pretrained EVAC checkpoint for retraining. Experiments on RoboTwin2.0 at the default 40% data budget show that confidence-guided selection improves post-training quality, while dense frame and patch weighting offers complementary reconstruction and semantic gains compared with scalar reward, progress, and judge-based scoring baselines. A quick visual overview of this work is available at `https://ConfAL-WM.github.io`.

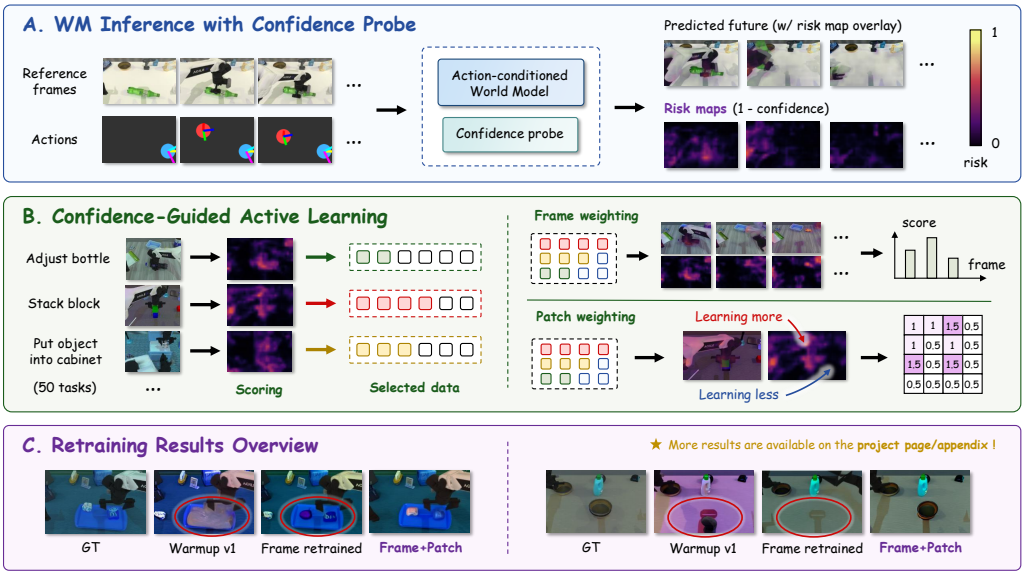


Figure 1: **An introduction of our work.** (A) Given reference frames and action conditions, we use a UNet-based latent diffusion world model as the backbone and train a confidence probe on decoder features to predict dense future risk maps. (B) Confidence guides active learning in two ways: selecting more informative post-training data and enhancing supervision on unreliable frames or patches. (C) Two representative episodes compare the same video frame from the ground truth, warmup model, frame-weighted retraining, and frame-and-patch-weighted retraining.

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.

1 INTRODUCTION

Action-conditioned embodied world models aim to predict future visual observations under given robot actions, enabling offline policy evaluation, synthetic data generation, and planning without direct environment interaction. EnerVerse-AC (EVAC) is a representative action-conditioned world model that generates future multi-view observations conditioned on robot actions (Jiang et al., 2025). However, whether under zero-shot transfer to new datasets or during post-training on existing domains, prediction errors are rarely uniformly distributed over the whole video. They often concentrate around moving robot arms, manipulated objects, contact regions, occlusions, and long-horizon interaction events. Recent studies on confidence-aware video generation and physics-reinforced world simulation also suggest that unreliable or physically inconsistent regions are usually spatially and temporally localized (Mei et al., 2026; Zhang et al., 2026). This motivates a post-training pipeline that allocates its available data budget and strengthens local supervision according to the tasks, frames, and regions where the world model is likely to fail.

In this work, we study confidence-guided active learning for action-conditioned world models, and refer to the framework as **ConfAL-WM**. We build upon EVAC as the backbone and attach a lightweight confidence probe to its UNet decoder features. Following C^3 (Mei et al., 2026), the probe learns dense confidence from local latent errors with random threshold conditioning. Decoder features provide spatially resolved representations that combine local detail with contextual information from the latent diffusion backbone. The probe predicts dense confidence maps in the latent space, which are aggregated into patch-, frame-, and task-level scores for selection and retraining. During probe training, an exponential moving average (EMA) adjusts the threshold sampling interval to track the distribution of local prediction errors. Ablations show that decoder features improve confidence prediction, while EMA matches the accuracy of a fixed calibration range.

Our active learning pipeline uses confidence in a staged and budget-aware manner. EVAC is originally pretrained on AgiBot World, a large-scale real-world manipulation dataset (Bu et al., 2025), and we use RoboTwin2.0 as the post-training dataset to evaluate the effectiveness of active learning under a new task and scene distribution (Chen et al., 2025). First, a small subset of data is used to retrain the confidence probe and warm up EVAC, producing a domain-adapted EVAC-v1. Second, we perform fast task-level prescreening with the confidence probe: harder tasks are assigned larger sampling budgets, while easier tasks receive fewer selected scenes. Third, after within-task random sampling, all selected-data models are initialized from the original pretrained EVAC checkpoint. EVAC-v1 supplies the acquisition and weighting signals. EVAC-v2 uses either the selected data directly or frame- and patch-level scores from additional EVAC-v1 inference and probe scoring. This extra scoring stage enables confidence-guided data enhancement during retraining.

This design gives confidence two roles. At the selection level, it allocates sampling budgets across tasks before random within-task selection. At the training level, it localizes which frames and patches should receive stronger supervision. This differs from scalar reward, progress, or judge-based scoring models, which can rank trajectories or frames but usually cannot provide patch-level prediction risk for world-model retraining (Lee et al., 2026; Liang et al., 2026; Ma et al., 2025; Ji et al., 2026). In our comparison, these scoring models are used as alternative selection signals, and some of them are further equipped with frame-weighted retraining. Confidence is used as our main signal, and we additionally study frame-and-patch-weighted retraining as a complementary dense enhancement strategy. Following the EWMBench-style evaluation protocol (Hu et al., 2025), our results show that confidence-guided selection and confidence-guided frame/patch weighting can improve prediction quality and embodied trajectory consistency over scalar scoring baselines.

Our contributions are summarized as follows:

1. We propose ConfAL-WM, a staged confidence-guided active learning pipeline for data-budget allocation when post-training embodied action-conditioned world models.
2. We adapt dense confidence estimation to a UNet-based action-conditioned world model using a decoder-feature confidence probe with EMA-based adaptive threshold supervision.
3. We introduce confidence-guided data enhancement, especially frame-and-patch-weighted retraining, to focus supervision on unreliable spatiotemporal regions.
4. We compare confidence with scalar scoring and global top-error selection, demonstrating gains in data selection, low-budget post-training, and localized weighted retraining.

2 Related Work

Action-conditioned world models and evaluation. Action-conditioned world models predict future observations under robot actions. EnerVerse-AC (EVAC), pretrained on AgiBot World, is the UNet-based backbone of our framework (Jiang et al., 2025; Bu et al., 2025); we post-train it on RoboTwin2.0 under new tasks, scenes, and robot embodiments (Chen et al., 2025). Recent alternatives improve action controllability or latent prediction, including IRASim and DINO-WM (Zhu et al., 2025; Zhou et al., 2025). Most closely related, Mei et al. (2026) introduce C^3 for dense confidence estimation in DiT-based video models.

EWMBench evaluates embodied world models through reconstruction, scene, motion, and semantic metrics, and provides our main evaluation protocol (Hu et al., 2025). Other benchmarks connect generated futures with policy performance or assess instruction following and physical plausibility (Jang et al., 2025; Li et al., 2025; NVIDIA, 2025).

Dense confidence, reward, progress, and judge signals. Dense uncertainty is the closest signal to our method. S^3 analyzes uncertainty in generative video models, while C^3 introduces dense calibrated confidence maps for controllable video generation (Mei et al., 2025; 2026). PRM-as-a-Judge provides process-level robotic auditing with macro- and micro-level signals (Ji et al., 2026). Our confidence additionally serves as an acquisition score and a local training weight.

We compare against several scalar scoring methods. GVL estimates temporal progress through in-context value learning (Ma et al., 2025); RoboReward learns general-purpose vision-language rewards for robotics (Lee et al., 2026); Robometer combines frame-level progress with trajectory preferences (Liang et al., 2026); and LRMs generate process and completion rewards online (Wu et al., 2026). These methods score trajectories or frames, whereas we also localize patch errors.

Localized supervision has recently been explored from complementary perspectives. CD-LAM reduces action-irrelevant latent bias through embodiment-focused reconstruction and action-aware objectives (Wei et al., 2026). PhysisForcing strengthens physical consistency by emphasizing physics-informative interaction regions (Zhang et al., 2026). Our method identifies such regions through world-model confidence and uses them for active selection and patch-weighted retraining.

Active learning, uncertainty-aware robot learning, and auxiliary world-model objectives. Classical active learning selects samples using uncertainty, disagreement, or coverage (Settles, 2009; Seung et al., 1992; Sener & Savarese, 2018). Robotics-specific methods extend these principles to view selection, rollout control, and demonstration curation (Eren et al., 2026; Dasgupta et al., 2024; Li et al., 2026; Dass et al., 2026). Most closely related, Römer et al. (2026) derive velocity-field disagreement (VFD) to quantify epistemic uncertainty in flow-based VLAs, and further introduce SAVE, which uses this uncertainty to prioritize tasks and initial scenes for active multitask finetuning. This shares our goal of using model uncertainty to reduce adaptation cost and focus supervision on informative data, but operates on action-policy uncertainty, whereas our method estimates dense video-prediction confidence and uses it for both data selection and localized world-model retraining. Future-aware auxiliary objectives can also improve embodied representations (Yuan et al., 2026; Lv et al., 2026; Luo et al., 2026).

3 Method

In this section, we first introduce a dense confidence probe for a UNet-based action-conditioned world model, which estimates patch-level prediction reliability from intermediate decoder features. We then describe how the resulting dense risk maps are aggregated into task-level acquisition scores and further converted into weights for confidence-guided EVAC retraining.

3.1 Dense Confidence Probe for UNet World Models

Confidence definition. Given an action-conditioned world model, we define confidence as the predicted probability that a local prediction error is below a threshold. In EVAC, the diffusion model predicts a denoising direction $\hat{v}^{(\tau)}$ for the ground-truth target $v^{(\tau)}$. For a predicted future

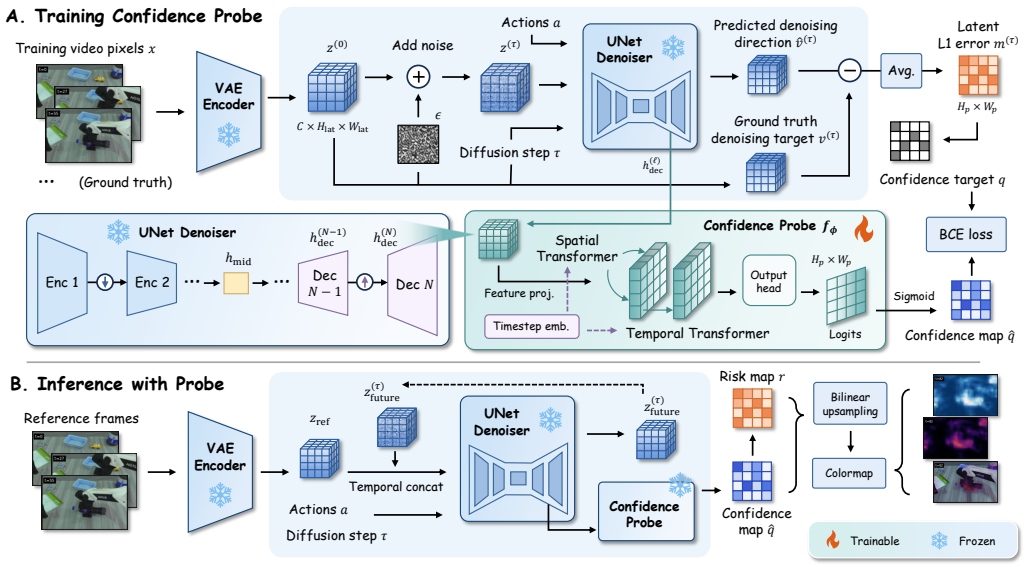


Figure 2: **Training and inference of the confidence probe in the UNet latent diffusion world model.** Video pixels are encoded into latent variables, diffused into $z^{(\tau)}$, processed by the UNet denoiser, and supervised by the denoising target $v^{(\tau)}$. The confidence probe is attached to selectable decoder features $h^{(\ell)}_{\text{dec}}$ and predicts dense confidence maps.

frame t and latent patch (i, j), we compute a **local mean absolute error** as

$$m^{(\tau)}_{t,(i,j)} = \frac{\sqrt{1-\bar{\alpha}_\tau}}{|\mathcal{P}_{i,j}|} \sum_{p \in \mathcal{P}_{i,j}} \left| \hat{v}^{(\tau)}_{t,p} - v^{(\tau)}_{t,p} \right| = \frac{1}{|\mathcal{P}_{i,j}|} \sum_{p \in \mathcal{P}_{i,j}} \left| \hat{z}^{(\tau)}_{t,p} - z^{(0)}_{t,p} \right|. \tag{1}$$

Here, $p = (c, x, y)$ indexes one channel-spatial entry in the latent tensor: c is the latent feature channel and (x, y) is a spatial location in the latent grid. The set $\mathcal{P}_{i,j}$ denotes the latent region corresponding to probe patch (i, j). $z^{(0)}_{t,p}$ is the clean video latent encoded from the ground-truth video, and $\hat{z}^{(\tau)}_{t,p}$ is its closed-form prediction recovered from $\hat{v}^{(\tau)}_{t,p}$ at diffusion timestep τ. This equivalent form avoids executing a complete reverse diffusion trajectory when constructing the confidence target. $\bar{\alpha}_\tau$ is the cumulative noise-schedule coefficient at timestep τ.

The **binary confidence target** at frame t is defined as

$$q_t = \{q_{t,(i,j)}\}_{i,j} \in \{0,1\}^{H_p \times W_p}, \qquad q_{t,(i,j)} = \mathbb{I}\{m^{(\tau)}_{t,(i,j)} < \theta_t\}, \tag{2}$$

where θ_t is a stochastic threshold sampled from an adaptive interval. Further architectural details of the latent-space error construction are provided in Appendix A.1.

Decoder-feature confidence probe. Let $h^{(\ell)}_{\text{dec}}$ denote the feature map extracted from the ℓ-th UNet decoder block. For each predicted future frame, the **confidence probe prediction** is

$$\hat{q}_t = \sigma\Big(f_\phi\Big(h^{(\ell)}_{\text{dec}}, e_\tau, e_\theta\Big)\Big) \in [0,1]^{H_p \times W_p}, \tag{3}$$

where e_τ denotes the diffusion-timestep embedding, e_θ denotes the sampled error threshold embedding, f_ϕ is the trainable confidence probe parameterized by ϕ, and $\sigma(\cdot)$ is the element-wise sigmoid function. Internally, the probe applies channel projection, spatial Transformer layers, temporal Transformer layers, and a patch-wise output head. The decoder index ℓ trades coarse semantic context for finer spatial resolution, as shown in Figure 3; we use $\ell = 5$, yielding a 20×32 spatial grid for dense confidence prediction.

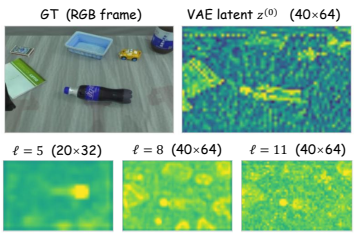


Figure 3: **Direct comparison for a single frame.** All operations are performed in the VAE latent space. The lower panels visualize the channel-averaged $\mathbb{E}_c\left[\left|h^{(\ell)}_{\text{dec}}\right|\right]$.

Adaptive threshold sampling. At training step s, we compute the lower and upper percentiles of the batch-wise local error distribution, and we update the threshold bounds as

$$p_l^{(s)} = \mathrm{Quantile}_{0.1}\left(m^{(s)}\right), \qquad p_h^{(s)} = \mathrm{Quantile}_{0.9}\left(m^{(s)}\right), \tag{4}$$

$$l^{(s)} = (1-\gamma_s)l^{(s-1)} + \gamma_s p_l^{(s)}, \qquad h^{(s)} = (1-\gamma_s)h^{(s-1)} + \gamma_s p_h^{(s)}. \tag{5}$$

The update rate γ_s is larger during warmup to follow the initial change in error scale, then decreases to smooth the batch-wise quantile estimates.

A separate threshold is sampled for each predicted future frame t as $\theta_t \sim \mathcal{U}\left(l^{(s)}, h^{(s)}\right)$. The sampled threshold θ_t is shared by all spatial patches (i, j) within the same frame. Further implementation details and the conditional interpretation of θ_t are provided in Appendix A.2.

Probe training objective. The probe is trained with binary cross-entropy over all supervised patches in the predicted future frames:

$$\mathcal{L}_{\mathrm{conf}} = -\mathbb{E}_{t,(i,j)}\Big[q_{t,(i,j)} \log \hat{q}_{t,(i,j)} + (1 - q_{t,(i,j)}) \log(1 - \hat{q}_{t,(i,j)})\Big]. \tag{6}$$

During probe training, the EVAC backbone remains frozen and gradients are propagated only through f_ϕ. At inference time, the frame-wise dense risk map is defined as $r_{t,(i,j)} = 1 - \hat{q}_{t,(i,j)}$.

3.2 Confidence-Guided Active Learning

Full pipeline. Given a candidate data pool, our active learning pipeline uses confidence for both sample acquisition and localized retraining. First, we train the confidence probe and warm up EVAC on a small subset of the data pool. The resulting EVAC-v1 is used for task-level prescreening, where representative episodes estimate task difficulty and determine the per-task candidate quota, according to which episodes are randomly sampled within each task.

All selected-data models are initialized from the original pretrained EVAC checkpoint; EVAC-v1 supplies the acquisition and weighting signals. EVAC-v2 is trained on the selected subset, optionally using confidence maps from additional EVAC-v1 inference to weight its retraining objective.

Risk-based selection criteria. We consider three confidence-based selection criteria: **1. mean risk**, **2. tail risk**, and **3. persistent risk**. Mean risk measures the overall prediction difficulty by averaging the dense risk map over all spatial locations and predicted future frames, whereas tail risk emphasizes severe localized failures. For tail risk, we flatten all patch risks $\{r_{t,(i,j)}\}$ into a vector of length $K = TH_pW_p$ and sort its entries in ascending order as $r_{[1]} \le r_{[2]} \le \cdots \le r_{[K]}$. Let η denote the selected high-risk tail ratio. The two scores are defined as

$$S_{\mathrm{mean}} = \mathbb{E}_{t,(i,j)}\left[r_{t,(i,j)}\right], \quad S_{\mathrm{tail}} = \frac{1}{\lceil \eta K \rceil} \sum_{n=K-\lceil \eta K \rceil+1}^{K} r_{[n]}. \tag{7}$$

Persistent risk captures high-risk local failures that recur across multiple frames. Let $K_p = H_pW_p$. For frame t, we sort its patch risks as $r_{t,[1]} \le \cdots \le r_{t,[K_p]}$ and average the largest fraction of them to obtain a frame-level score u_t. We then sort these frame scores as $u_{[1]} \le \cdots \le u_{[T]}$ and average the highest-scoring frames:

$$u_t = \frac{1}{\lceil \eta_p K_p \rceil} \sum_{n=K_p-\lceil \eta_p K_p \rceil+1}^{K_p} r_{t,[n]}, \quad S_{\mathrm{persistent}} = \frac{1}{\lceil \eta_t T \rceil} \sum_{n=T-\lceil \eta_t T \rceil+1}^{T} u_{[n]}. \tag{8}$$

Here, η_p denotes the fraction of high-risk patches retained within each frame, and η_t denotes the fraction of high-risk frames retained across the predicted video.

Confidence-guided frame and patch weighting. For weighted EVAC-v2 retraining, we perform an additional inference and confidence-scoring stage on the selected episodes. Frame weighting assigns the spatially averaged risk uniformly to all patches within frame t, whereas patch weighting retains the original dense risk $r_{t,(i,j)}$. We unify the two forms as

$$r_{t,(i,j)}^{(\alpha)} = \mathbb{E}_{(i',j')}\left[r_{t,(i',j')}\right] + \alpha\left(r_{t,(i,j)} - \mathbb{E}_{(i',j')}\left[r_{t,(i',j')}\right]\right), \qquad \alpha \in [0,1]. \tag{9}$$

When $\alpha = 0$, all patches within a frame receive the same frame-level risk; when $\alpha = 1$, the original patch-level risk map is recovered. Intermediate values preserve the frame-level baseline while introducing local residual modulation.

After quantile normalization and clipping, we denote the resulting risk by $\widetilde{r}^{(\alpha)}_{t,(i,j)} \in [0,1]$. It is converted into a local loss multiplier and applied to the world-model objective as

$$w_{t,(i,j)} = 1 + \lambda_{\text{eff}}(s)\widetilde{r}^{(\alpha)}_{t,(i,j)}, \qquad \mathcal{L}_{\text{WM}} = \frac{\mathbb{E}_{t,(i,j)}\left[w_{t,(i,j)}\ell_{t,(i,j)}\right]}{\mathbb{E}_{t,(i,j)}\left[w_{t,(i,j)}\right]}. \tag{10}$$

Here, $\ell_{t,(i,j)}$ is the local world-model training loss, and $\lambda_{\text{eff}}(s) = \lambda_{\text{conf}} \min(1, s/s_{\text{warm}})$ linearly increases the weighting strength during the first s_{warm} EVAC-v2 retraining steps. Thus, higher-risk frames and patches receive larger optimization weights. The confidence maps are detached during EVAC retraining, so gradients are propagated only through the world model.

4 Experiments

Our experiments address two questions. **1. Why Confidence?** Does predicted risk track world-model errors and support selection relative to direct error? **2. Why Active Learning?** How does confidence-guided allocation affect post-training quality across fixed data and optimization budgets?

4.1 Experimental Setup

Data setting. In our setting, EVAC is pretrained on AgiBot World (Bu et al., 2025), while active learning and post-training are conducted on a subset of RoboTwin2.0 (Chen et al., 2025). This creates a transfer setting involving new tasks, scenes, and robot embodiments for evaluating adaptation to the target manipulation domain. We use data from the Aloha-AgileX dual-arm robot, covering 50 manipulation tasks. Each task contains approximately 500 randomized scenes, resulting in 24,992 videos in total. The video length ranges from 98 to 578 frames.

Implementation details. During confidence-probe training, the entire EVAC backbone is frozen and only the confidence probe f_ϕ is optimized. During EVAC-v1 warmup and EVAC-v2 retraining, the UNet denoiser and two projection modules are trainable, while the VAE, CLIP embedder, and action-conditioning resampler remain frozen. The full EVAC model has about 2.33B parameters, and the confidence probe has about 8.2M parameters.

We split the data into 6,248 episodes for probe training and EVAC-v1 warmup, 18,244 candidates, and 250 each for validation and testing. The nominal selection budget is 7,298 episodes. EVAC-v2 uses AdamW, learning rate 5×10^{-5}, fp16, and 4,000 updates on two A800 GPUs. Scoring averages three probe timesteps $\tau \in [50, 200]$. Main tables retain the three reported runs and their averages. Detailed per-seed results and bootstrap statistics are provided in Appendix B.2. Appendix B.4 reports the direct-error comparison, and Appendix B.5 evaluates shorter, fixed-update budget sweeps. Appendix B.6 specifies quotas, weighting, and the seeds.

Baselines. Probe ablations evaluate bottleneck and decoder features on EVAC using C^3-style fixed-range supervision (Mei et al., 2026) and EMA supervision (Appendix B.3). For data selection, we compare the following methods (superscripts match Figure 6): RoboReward[1], a vision-language reward model for robotics (Lee et al., 2026); GVL[2], which uses VLMs as in-context value learners (Ma et al., 2025); Robometer-Prog and Robometer-Pref[3], trajectory progress and preference models (Liang et al., 2026); PRM-as-Judge[4], a dense process-level robotic auditing method (Ji et al., 2026); and LRMs[5], large reward models for online robot reward generation (Wu et al., 2026).

World-model evaluation. We follow the EWMBench-style evaluation pipeline (Hu et al., 2025). PSNR and SSIM measure low-level reconstruction quality. Scene consistency measures layout and object preservation. Logics evaluates higher-level physical and interaction plausibility. Sem.-CLIP and Sem.-BLEU measure visual-semantic and textual-semantic agreement. Traj-HSD, Traj-Dyn, and Traj-nDTW evaluate robot trajectories; higher values indicate better trajectory consistency.

Beyond quantitative evaluation, Appendix B.7 and Appendix B.8 provide qualitative examples of retraining evolution and confidence-map visualization, respectively.

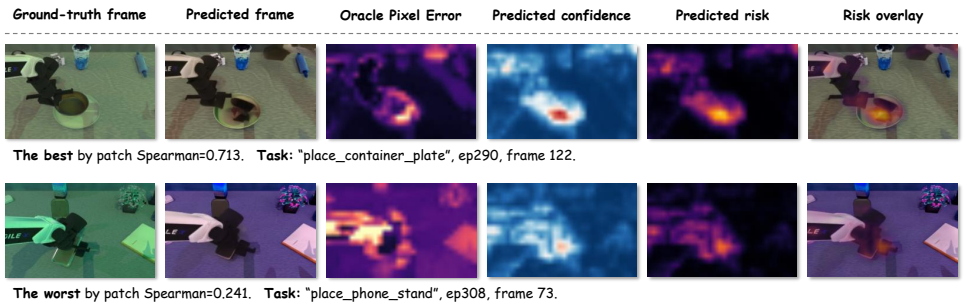


Figure 4: **Qualitative confidence visualization.** High- and low-agreement episodes from different manipulation tasks are shown. High-risk regions generally coincide with errors around robot arms, manipulated objects, contacts, and occlusions. More examples: see Appendix B.8.

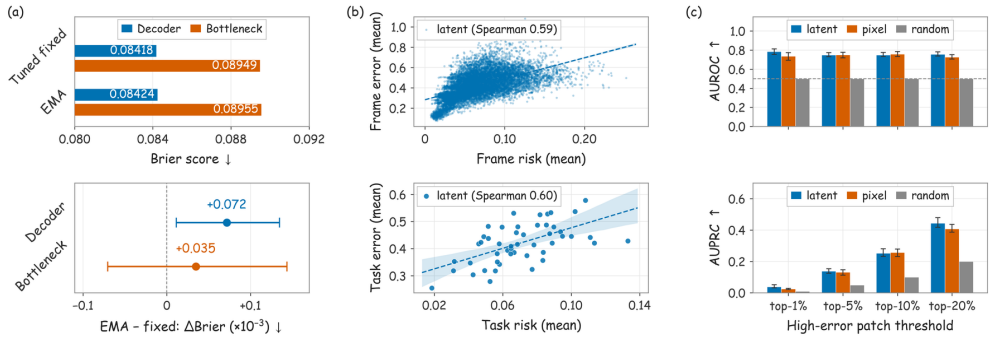


Figure 5: **Confidence-probe ablations and multi-scale risk validity.** (a) Feature and threshold-supervision ablations on EVAC-v1, showing episode-mean Brier (top) and paired task-macro EMA-minus-tuned-fixed Brier (bottom). Tuned fixed uses the calibrated $[Q_{.05}, Q_{.95}]$ range; error bars denote 95% task-bootstrap intervals, with positive differences favoring fixed supervision. (b) Mean risk versus latent RMS rollout error at frame (top) and task (bottom) levels. (c) AUROC (top) and AUPRC (bottom) for detecting the top-$Q\%$ highest-error patches in latent and pixel spaces.

4.2 Why Confidence?

We evaluate EVAC-v1 confidence as an estimate of world-model prediction error under two protocols. Probe ablations use 250 held-out episodes spanning 43 tasks. Multi-scale risk diagnostics use one episode from each of the 50 prescreened tasks.

Qualitative localization. Figure 4 shows that predicted risk maps identify spatially localized failures across the illustrated manipulation episodes and their future video frames. High-risk regions commonly appear around moving manipulators, object interactions, contacts, and temporarily occluded objects. Their agreement with pixel prediction-error maps indicates that the probe captures local failures during the execution of diverse manipulation tasks.

Confidence as a multi-scale ranking signal. We quantitatively compare risk with oracle error across patch, frame, and task levels. Figure 5(a) summarizes the controlled probe ablations: decoder features reduce Brier by 5.93% under both supervision rules, while EMA and tuned fixed supervision yield closely matched scores. Figure 5(b–c) evaluates the resulting risk signal across scales. In latent space, risk achieves Spearman correlations of 0.540, 0.590, and 0.595 at the patch, frame, and task levels, respectively. For detecting the top-5% highest-error patches, it obtains an AUROC of 0.761 and an AUPRC of 0.146, compared with random baselines of 0.5 and 0.05. These results support aggregating dense risk into frame- and task-level scores for active data selection.

Spatial and temporal behavior. Table 1 further shows that risk maps are temporally stable: adjacent frames obtain a top-region IoU of 0.740, while the flicker score is only 0.005. Risk and latent error are also temporally synchronized, with a peak correlation of 0.602 occurring near zero lag. Nevertheless, the top-5% spatial IoU is 0.130, suggesting that confidence partially identifies error-prone regions but does not precisely reproduce their boundaries.

Overall, the probe provides an *ordinal* risk signal for ranking patches, frames, and tasks. On frozen EVAC-v1, decoder features reduce Brier from 0.08949 to 0.08418 under fixed-range supervision, while EMA achieves a similar Brier of 0.08424. Appendix B.3 presents feature and threshold ablations; further confidence diagnostics appear in Appendix B.1.

Table 1: **Summary of latent-space confidence validity across multiple scales.** Frame- and task-level statistics use mean aggregation.

Property	Metric	Result
Multi-scale ranking	Patch / Frame / Episode Spearman (↑)	0.540/0.590/0.595
High-error detection	AUROC / AUPRC@top-5% (↑)	0.761/0.146
Spatial agreement	Top-5% IoU (↑)	0.130
Temporal stability	Adjacent-frame IoU (↑) / Flicker (↓)	0.740/0.005
Temporal alignment	Peak correlation (↑) / |lag| (↓)	0.602/ ≈ 0

4.3 Why Active Learning?

We evaluate whether confidence-guided selection and retraining improve the action-conditioned world model under a fixed data budget. We report selection-only retraining and selection with additional weighting, since the latter requires an extra inference and scoring stage after selection.

For compact comparison, all component metrics are normalized to $[0, 1]$ and aggregated into four dimensions: Reconstruction $=$ (PSNR $+$ SSIM)$/2$, Scene $=$ Scene Consistency, Semantics $=$ (Logics $+$ Sem.-CLIP $+$ Sem.-BLEU)$/3$, and Motion $=$ Traj-HSD $+$ Traj-Dyn $+$ Traj-nDTW.

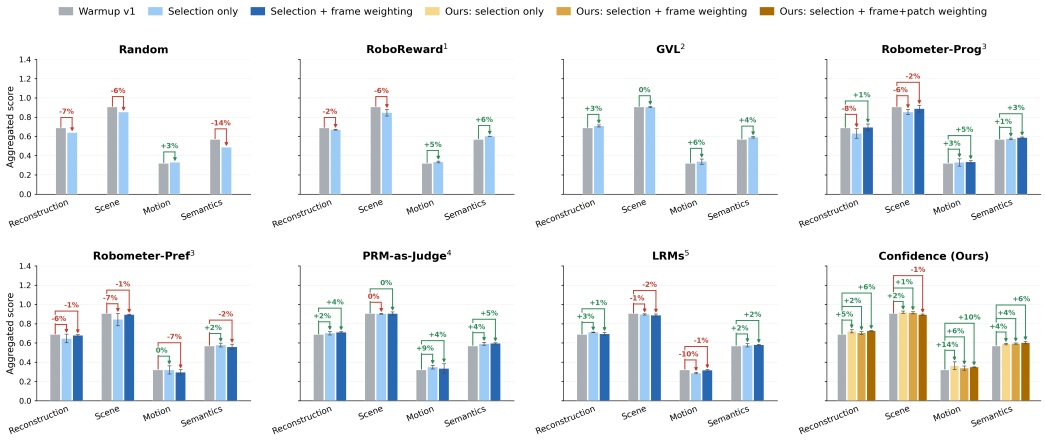


Figure 6: **Main comparison of active-learning scoring and retraining strategies.** Each panel compares EVAC-v1 with selection-only retraining and additional frame weighting. Confidence enables frame+patch weighting. Learned-method bars report means over three seeds, and arrows report relative changes from EVAC-v1. Error bars indicate paired-bootstrap 95% confidence intervals.

Main comparison. Figure 6 and Table 2 show that confidence-guided mean-risk selection leads seven of nine metrics among selection-only methods at the 40% budget. Confidence-guided task allocation exceeds global top-error selection on eight of nine metrics in the individual-run comparison, improving reconstruction, scene consistency, and motion (Appendix B.4).

Additional weighting provides complementary gains. Among weighted methods, confidence with frame weighting leads Scene Consistency; frame-and-patch weighting leads PSNR, SSIM, Logics (tied), Sem.-CLIP, Sem.-BLEU, and all three trajectory metrics. These results motivate two uses of confidence: **1. Budget-aware selection:** mean risk allocates available training data across tasks; **2. Localized enhancement:** dense maps emphasize unreliable frames and patches during retraining.

These gains involve trade-offs: compared with selection-only confidence retraining, frame-and-patch weighting improves Reconstruction and Semantics but reduces Scene Consistency and Motion. It improves Motion relative to frame-only weighting. This pattern is compatible with metric trade-offs discussed in prior work (Liao et al., 2024; Dou et al., 2026; Ye et al., 2026), but does not establish their cause. It also highlights the limitations of these proxy metrics.

We also ablate active learning across different candidate-pool budget fractions. With 750 updates and a separate 64-episode evaluation subset, confidence improves all four aggregates over random at 10% budget; larger budgets show metric-dependent trade-offs (Appendix B.5).

Table 2: **Detailed normalized results under the same active-learning setting.** Learned-method results are averaged over three seeds (42, 3407, and 123), and oracle uses optimizer seed 42. Methods are separated into selection-only retraining and selection with additional weighting. Green, yellow, and red indicate the best, second-best, and third-best results within each block, respectively.

Scoring	Weighting	Reconstruction (↑)		Scene (↑)	Semantics (↑)			Motion (↑)		
		PSNR (↑)	SSIM (↑)	Scene Cons. (↑)	Logics (↑)	Sem.-CLIP (↑)	Sem.-BLEU (↑)	Traj-HSD (↑)	Traj-Dyn (↑)	Traj-nDTW (↑)
Base EVAC		0.5532	0.5778	0.8757	0.4298	0.8523	0.1799	0.0007	0.0001	0.0006
Base EVAC (Warmup v1)		0.6446	0.7309	0.9047	0.5537	0.8824	0.2708	0.1045	0.0721	0.1439
Selection-only retraining: no additional weighting after selection										
Random	None	0.5968	0.6849	0.8524	0.3554	0.8689	0.2372	0.1067	0.0725	0.1518
Oracle error	None	0.6458	0.7542	0.8988	0.5826	0.8875	**0.2914**	0.0985	0.0587	0.1396
GVL[2]	None	0.6630	0.7546	0.9057	0.5992	0.8898	0.2849	0.1114	0.0688	0.1592
RoboReward[1]	None	0.6266	0.7158	0.8470	**0.6391**	0.8861	0.2815	0.1113	0.0693	0.1554
Robometer-Prog[3]	None	0.5809	0.6825	0.8532	0.5840	0.8810	0.2627	0.1090	0.0690	0.1526
Robometer-Pref[3]	None	0.5919	0.7029	0.8443	0.5813	0.8859	0.2695	0.1046	0.0638	0.1524
PRM-as-Judge[4]	None	0.6467	0.7580	0.9033	0.6033	0.8880	0.2852	0.1154	0.0746	0.1606
LRMs[5]	None	0.6617	0.7603	0.8966	0.5744	0.8827	0.2756	0.0946	0.0587	0.1352
Confidence (Ours)	**None**	**0.6746**	**0.7692**	**0.9196**	0.5923	**0.8903**	0.2865	**0.1181**	**0.0812**	**0.1650**
Selection + additional weighting: extra scoring after selection										
Robometer-Prog[3]	Frame	0.6439	0.7449	0.8882	0.5882	0.8893	0.2878	0.1097	0.0697	0.1563
Robometer-Pref[3]	Frame	0.6317	0.7289	0.8945	0.5289	0.8849	0.2665	0.0974	0.0654	0.1342
PRM-as-Judge[4]	Frame	0.6584	0.7661	0.9053	**0.6226**	0.8868	0.2795	0.1082	0.0728	0.1536
LRMs[5]	Frame	0.6463	0.7448	0.8882	0.5826	0.8869	0.2750	0.1053	0.0635	0.1483
Confidence (Ours)	**Frame**	0.6595	0.7500	**0.9143**	0.6171	0.8874	0.2789	0.1106	0.0684	0.1608
Confidence (Ours)	**Fr.+Patch**	**0.6772**	**0.7758**	0.8942	**0.6226**	**0.8925**	**0.2952**	**0.1118**	**0.0759**	**0.1633**

Table 3: **Ablation of selection criteria.** All experiments use the default seed 42 and no additional weighting. Green, yellow, and red indicate the best, second-best, and third-best results.

Variant	Reconstruction (↑)		Scene (↑)	Semantics (↑)			Motion (↑)		
	PSNR (↑)	SSIM (↑)	Scene Cons. (↑)	Logics (↑)	Sem.-CLIP (↑)	Sem.-BLEU (↑)	Traj-HSD (↑)	Traj-Dyn (↑)	Traj-nDTW (↑)
Random	0.5968	0.6849	0.8524	0.3554	0.8689	0.2372	0.1067	0.0725	0.1518
Mean Risk	**0.6838**	**0.7797**	**0.9322**	**0.6198**	0.8894	0.2754	**0.1313**	**0.0921**	**0.1780**
Tail Risk	0.6317	0.7197	0.9218	0.5579	0.8890	**0.3047**	0.1207	0.0711	0.1630
Persistent Risk	0.6529	0.7470	0.9230	0.5496	**0.8928**	0.2858	0.0926	0.0608	0.1342

Ablation studies. Table 3 isolates the effect of the confidence-risk aggregation used for data selection, without additional frame or patch weighting. Mean risk gives the strongest overall result, achieving the best performance on seven of the nine component metrics, including both reconstruction metrics, Scene Consistency, Logics, and all three trajectory metrics. Tail risk obtains the highest Sem.-BLEU, while persistent risk obtains the highest Sem.-CLIP. We therefore use mean risk as the default acquisition criterion in our main experiments.

5 Conclusion

We presented ConfAL-WM, a confidence-guided active learning framework for post-training action-conditioned world models. By attaching a lightweight confidence probe to UNet decoder features, our method supports task-level data selection as well as frame- and patch-level weighted retraining. Experiments on RoboTwin2.0 show that confidence provides an effective risk signal and improves reconstruction, scene consistency, and semantic quality over scalar scoring baselines.

Our current approach still has several limitations. First, the confidence probe is designed around the internal decoder features of a UNet diffusion backbone and is trained only with EVAC on RoboTwin2.0, making it difficult to transfer directly across world-model architectures and domains. Future work could train a general-purpose world confidence model using multiple backbone world models and diverse embodied datasets. Such a model would also require a more universal input interface. Second, the probe-design ablations use cached single-step features. Evaluating confidence along free-running rollouts and jointly varying probe design, acquisition, and loss weighting would clarify how local error estimates affect post-training performance. Third, current world-model evaluation remains incomplete, particularly because visual fidelity and motion accuracy may conflict. Future protocols could separately evaluate robot-arm motion, manipulated-object dynamics, interaction regions, and static backgrounds, providing a more precise account of both local reconstruction quality and action-conditioned physical evolution.

AI USE STATEMENT

Generative AI tools were used throughout this work as research assistants in three main aspects. First, they were used to aid and polish manuscript writing, including language editing, reorganization, and improvements to presentation and readability. Second, they assisted with retrieval and discovery of relevant literature and baseline methods; all cited papers, bibliographic information, and related-work claims were subsequently checked and verified by the authors. Third, they assisted with research ideation and execution, including feedback on experimental design, preparation and debugging of experiment and evaluation scripts and commands, organization and interpretation of experimental results, and preparation of the released code, datasets, checkpoints, and associated documentation.

All AI-assisted experimental procedures were reviewed and executed by the authors. Reported numerical results were obtained from the actual experimental runs and were checked against the corresponding logs and evaluation artifacts. AI-assisted code and release materials were manually inspected and verified before use or publication. The authors take full responsibility for the final content of this work, including its text, scientific claims, experimental results, code, and released artifacts.

REPRODUCIBILITY STATEMENT

We provide the implementation, model checkpoints, data and evaluation artifacts necessary to reproduce the main results of this work. The anonymous project page[1] provides an overview of the method, experimental results, confidence visualizations, and links to all released resources. The source code and experimental pipeline are available in the anonymous code repository[2]. We additionally release the EVAC-v1 and EVAC-v2 checkpoints, confidence probes, and trajectory detector through the anonymous model repository[3], together with the corresponding data and evaluation artifacts[4].

The released artifacts include task-level prescreening outputs, EVAC-v1 inference results, dense confidence maps and metadata used for confidence-guided retraining, baseline scoring artifacts, trajectory annotations, and the evaluation tables and bootstrap statistics reported in the paper. Detailed model architecture, confidence-target construction, active-learning procedure, training objectives, implementation settings, random seeds, parameter sensitivity analyses, and seed-wise numerical results are further documented in the main paper and appendices.

ETHICS STATEMENT

This work aims to improve the data efficiency and reliability of embodied world-model post-training by focusing computation and supervision on informative data and unreliable spatiotemporal regions. The experiments do not involve human subjects or personally identifiable information. More accurate action-conditioned world models may benefit robotic simulation, planning, policy evaluation, and synthetic-data generation. However, predictions produced by such models are not guaranteed to correspond to physically valid outcomes, particularly under distribution shifts or in previously unseen environments. They should therefore not be treated as substitutes for real-world validation in safety-critical robotic applications. The released code, checkpoints, and derived artifacts are intended for research use and should be used in accordance with the licenses and usage conditions of the corresponding upstream models and datasets.

[1] https://confal-wm.github.io/

[2] https://github.com/ConfAL-WM/ConfAL-WM

[3] https://huggingface.co/anonymous89793/ConfAL-WM

[4] https://huggingface.co/datasets/anonymous89793/ConfAL-WM-Dataset

REFERENCES

Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xu Huang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3549–3556. IEEE, 2025. doi: 10.1109/IROS60139.2025.11247088.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan-ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation, 2025. URL https://arxiv.org/abs/2506.18088.

Saptarshi Dasgupta, Akshat Gupta, Shreshth Tuli, and Rohan Paul. Uncertainty-aware active learning of nerf-based object models for robot manipulators using visual and re-orientation actions, 2024. URL https://arxiv.org/abs/2404.01812.

Shivin Dass, Alaa Khaddaj, Logan Engstrom, Aleksander Madry, Andrew Ilyas, and Roberto Martín-Martín. Datamil: Selecting data for robot imitation learning with datamodels. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 1311–1336, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/033d9e8dbbbc0ad90e59222cf1db0fc2-Paper-Conference.pdf.

Weijia Dou, Wenzhao Zheng, Weiliang Chen, Yu Zheng, Jie Zhou, and Jiwen Lu. Measuring 3d spatial geometric consistency in dynamic video generation, 2026. URL https://arxiv.org/abs/2603.19048.

Mehmet Arda Eren, Jan Babič, and Erhan Oztop. Sample-efficient robot learning for supervised effect prediction tasks. In *2026 IEEE/SICE International Symposium on System Integration (SII)*, pp. 1198–1203, 2026. doi: 10.1109/SII64115.2026.11404575.

Yue Hu, Siyuan Huang, Yue Liao, Shengcong Chen, Pengfei Zhou, Liliang Chen, Guanghui Ren, and Maoqing Yao. Ewmbench: Evaluating scene, motion, and semantic quality in embodied world models. In *36th British Machine Vision Conference 2025, BMVC 2025, Sheffield, UK, November 24–27, 2025*. BMVA, 2025. URL https://bmva-archive.org.uk/bmvc/2025/assets/papers/Paper_736/paper.pdf.

Joel Jang, Seonghyeon Ye, Zongyu Lin, Jiannan Xiang, Johan Bjorck, Yu Fang, Fengyuan Hu, Spencer Huang, Kaushil Kundalia, Yen-Chen Lin, Loic Magne, Ajay Mandlekar, Avnish Narayan, You Liang Tan, Guanzhi Wang, Jing Wang, Qi Wang, Yinzhen Xu, Xiaohui Zeng, Kaiyuan Zheng, Ruijie Zheng, Ming-Yu Liu, Luke Zettlemoyer, Dieter Fox, Jan Kautz, Scott Reed, Yuke Zhu, and Linxi Fan. Dreamgen: Unlocking generalization in robot learning through video world models. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 5170–5194. PMLR, 2025. URL https://proceedings.mlr.press/v305/jang25a.html.

Yuheng Ji, Yuyang Liu, Huajie Tan, Xuchuan Huang, Fanding Huang, Yijie Xu, Cheng Chi, Yuting Zhao, Huaihai Lyu, Peterson Co, Mingyu Cao, Qiongyu Zhang, Zhe Li, Enshen Zhou, Pengwei Wang, Zhongyuan Wang, Shanghang Zhang, and Xiaolong Zheng. Prm-as-a-judge: A dense evaluation paradigm for fine-grained robotic auditing, 2026. URL https://arxiv.org/abs/2603.21669.

Yuxin Jiang, Shengcong Chen, Siyuan Huang, Liliang Chen, Pengfei Zhou, Yue Liao, Xindong He, Chiming Liu, Hongsheng Li, Maoqing Yao, and Guanghui Ren. Enerverse-ac: Envisioning embodied environments with action condition, 2025. URL https://arxiv.org/abs/2505.09723.

Tony Lee, Andrew Wagenmaker, Karl Pertsch, Percy Liang, Sergey Levine, and Chelsea Finn. Roboreward: General-purpose vision-language reward models for robotics, 2026. URL https://arxiv.org/abs/2601.00675.

Chenhao Li, Andreas Krause, and Marco Hutter. Uncertainty-aware robotic world model makes offline model-based reinforcement learning work on real robots, 2026. URL https://arxiv.org/abs/2504.16680.

Dacheng Li, Yunhao Fang, Yukang Chen, Shuo Yang, Shiyi Cao, Justin Wong, Michael Luo, Xiaolong Wang, Hongxu Yin, Joseph Gonzalez, Ion Stoica, Song Han, and Yao Lu. Worldmodelbench: Judging video generation models as world models. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025. doi: 10.52202/085713-1834. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/4ec03ed08a3fcb59e1c815b5598beff1-Paper-Datasets_and_Benchmarks_Track.pdf.

Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S. Huang, Luke Zettlemoyer, Dieter Fox, Yu Xiang, Anqi Li, Andreea Bobu, Abhishek Gupta, Stephen Tu, Erdem Biyik, and Jesse Zhang. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons, 2026. URL https://arxiv.org/abs/2603.02115.

Mingxiang Liao, Hannan Lu, Xinyu Zhang, Fang Wan, Tianyu Wang, Yuzhong Zhao, Wangmeng Zuo, Qixiang Ye, and Jingdong Wang. Evaluation of text-to-video generation models: A dynamics perspective. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 109790–109816. Curran Associates, Inc., 2024. doi: 10.52202/079017-3483. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/c6483c8a68083af3383f91ee0dc6db95-Paper-Conference.pdf.

Hao Luo, Wanpeng Zhang, Yicheng Feng, Sipeng Zheng, Haiweng Xu, Chaoyi Xu, Ziheng Xi, Yuhui Fu, and Zongqing Lu. Being-h0.7: A latent world-action model from egocentric videos, 2026. URL https://arxiv.org/abs/2605.00078.

Jindi Lv, Hao Li, Jie Li, Fankun Kong, Yang Wang, Pengfei Yi, Yifei Nie, Xiaofeng Wang, Zheng Zhu, Chaojun Ni, Qiuping Deng, Hengtao Li, Jiancheng Lv, and Guan Huang. Viva: A video-generative value model for robot reinforcement learning, 2026. URL https://arxiv.org/abs/2604.08168.

Yecheng Jason Ma, Joey Hejna, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, Osbert Bastani, Dinesh Jayaraman, Wenhao Yu, Tingnan Zhang, Dorsa Sadigh, and Fei Xia. Vision language models are in-context value learners. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 33984–34009, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/54854cf15a24fff9f5134a8641136fe4-Paper-Conference.pdf.

Zhiting Mei, Ola Shorinwa, and Anirudha Majumdar. How confident are video models? empowering video models to express their uncertainty, 2025. URL https://arxiv.org/abs/2510.02571.

Zhiting Mei, Tenny Yin, Micah Baker, Ola Shorinwa, and Anirudha Majumdar. World models that know when they don't know: Controllable video generation with calibrated uncertainty, 2026. URL https://arxiv.org/abs/2512.05927.

NVIDIA. Pbench: A physical ai benchmark for world models, 2025. URL https://huggingface.co/datasets/nvidia/PBench.

Ralf Römer, Maximilian Seeliger, Saida Liu, Ben Sturgis, Marco Bagatella, Daniel Marta, Andreas Krause, and Angela P. Schoellig. Uncertainty quantification for flow-based vision-language-action models, 2026. URL https://arxiv.org/abs/2606.18043.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1aIuk-RW.

Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009. URL https://burrsettles.com/pub/settles.activelearning.pdf.

H. Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pp. 287–294. Association for Computing Machinery, 1992. doi: 10.1145/130385.130417. URL https://doi.org/10.1145/130385.130417.

Yufan Wei, Kun Zhou, Lingjun Mao, Zijun Zhang, Ziming Xu, Ziqiao Xi, Shuang Liang, Ruobing Han, Yuchen Yan, Xinyue Wang, Fan Feng, and Biwei Huang. Causally debiased latent action model for embodied action conditioned world models, 2026. URL https://arxiv.org/abs/2607.09185.

Yanru Wu, Weiduo Yuan, Ang Qi, Vitor Guizilini, Jiageng Mao, and Yue Wang. Large reward models: Generalizable online robot reward generation with vision-language models, 2026. URL https://arxiv.org/abs/2603.16065.

Xi Ye, Wenjia Yang, Yangyang Xu, Xiaoyang Liu, Duo Su, Mengfei Xia, and Jun Zhu. Shift: Motion alignment in video diffusion models with adversarial hybrid fine-tuning. In *European Conference on Computer Vision (ECCV)*, 2026.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination?, 2026. URL https://arxiv.org/abs/2603.16666.

Peiwen Zhang, Yufan Deng, Shangkun Sun, Juncheng Ma, Duomin Wang, Jonas Du, Zilin Pan, Ye Huang, Hao Liang, Songyan Huang, Ruihua Zhang, Enze Xie, Ming-Yu Liu, and Daquan Zhou. Physisforcing: Physics reinforced world simulator for robotic manipulation, 2026. URL https://arxiv.org/abs/2606.28128.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79115–79135. PMLR, 2025.

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

A More Details on Methods

A.1 Patch-level Error Construction

Figure 7 illustrates the overall architecture used to construct the patch-level confidence target.

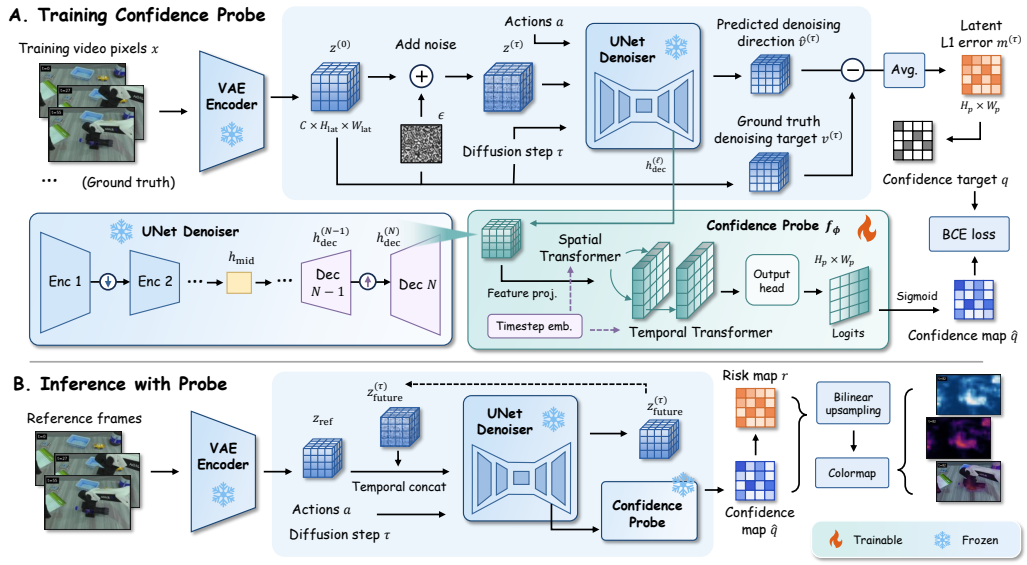


Figure 7: Training and inference of the confidence probe in the UNet latent diffusion world model. The figure follows the same training and inference pipeline as Figure 2.

Latent diffusion and velocity target. Given a ground-truth training video x, the frozen VAE encoder first maps it into a clean latent representation

$$z^{(0)} = \text{Enc}_{\text{VAE}}(x), \qquad z^{(0)} \in \mathbb{R}^{C \times H_{\text{lat}} \times W_{\text{lat}}}, \tag{11}$$

where C denotes the latent channel dimension and $(H_{\text{lat}}, W_{\text{lat}})$ denotes the spatial resolution of the latent grid. For simplicity, the batch and video-time dimensions are omitted in this subsection.

At diffusion timestep τ, Gaussian noise $\varepsilon \sim \mathcal{N}(0, I)$ is sampled and added according to the forward diffusion schedule:

$$z^{(\tau)} = \sqrt{\bar{\alpha}_\tau}\, z^{(0)} + \sqrt{1 - \bar{\alpha}_\tau}\, \varepsilon, \tag{12}$$

where $\bar{\alpha}_\tau$ is the cumulative noise-schedule coefficient at timestep τ. The action-conditioned UNet predicts the corresponding velocity parameterization as

$$\hat{v}^{(\tau)} = \text{UNet}_\Theta\left(z^{(\tau)}, a, e_\tau\right), \qquad \hat{v}^{(\tau)} \in \mathbb{R}^{C \times H_{\text{lat}} \times W_{\text{lat}}}, \tag{13}$$

where a is the robot-action condition and e_τ is the diffusion-timestep embedding. Under the velocity-prediction formulation, the ground-truth denoising target is analytically constructed as

$$v^{(\tau)} = \sqrt{\bar{\alpha}_\tau}\, \varepsilon - \sqrt{1 - \bar{\alpha}_\tau}\, z^{(0)}, \qquad v^{(\tau)} \in \mathbb{R}^{C \times H_{\text{lat}} \times W_{\text{lat}}}. \tag{14}$$

Therefore, $\hat{v}^{(\tau)}$ and $v^{(\tau)}$ have the same tensor shape and lie in the same velocity-prediction latent space. The action a and timestep embedding e_τ condition the prediction $\hat{v}^{(\tau)}$, whereas the target $v^{(\tau)}$ is directly determined by $z^{(0)}$, ε, and the diffusion schedule.

Equivalent clean-latent reconstruction. Rather than executing the complete reverse diffusion trajectory, we can directly recover the predicted clean latent from the current noisy latent and the velocity prediction:

$$\hat{z}^{(\tau)} = \sqrt{\bar{\alpha}_\tau}\, z^{(\tau)} - \sqrt{1 - \bar{\alpha}_\tau}\, \hat{v}^{(\tau)}. \tag{15}$$

This provides an equivalent way to measure the local prediction error in the clean-latent space. Specifically, substituting equation 12 and equation 15 into $\left|\hat{z}^{(\tau)} - z^{(0)}\right|$ gives

$$\begin{aligned}\left|\hat{z}^{(\tau)} - z^{(0)}\right| &= \left|\sqrt{\bar{\alpha}_\tau}\, z^{(\tau)} - \sqrt{1-\bar{\alpha}_\tau}\, \hat{v}^{(\tau)} - z^{(0)}\right| \\ &= \left|(\bar{\alpha}_\tau - 1)\, z^{(0)} + \sqrt{\bar{\alpha}_\tau\,(1-\bar{\alpha}_\tau)}\,\varepsilon - \sqrt{1-\bar{\alpha}_\tau}\,\hat{v}^{(\tau)}\right| \\ &= \sqrt{1-\bar{\alpha}_\tau}\left|\sqrt{1-\bar{\alpha}_\tau}\, z^{(0)} - \sqrt{\bar{\alpha}_\tau}\,\varepsilon + \hat{v}^{(\tau)}\right|.\end{aligned} \tag{16}$$

Using the velocity target in equation 14, we therefore obtain

$$\left|\hat{z}^{(\tau)} - z^{(0)}\right| = \sqrt{1-\bar{\alpha}_\tau}\left|\hat{v}^{(\tau)} - v^{(\tau)}\right|. \tag{17}$$

Hence, at a fixed diffusion timestep τ, the clean-latent reconstruction error and the velocity-prediction error differ only by the scalar factor $\sqrt{1-\bar{\alpha}_\tau}$. This allows confidence supervision to use velocity-prediction error directly, without running the full reverse diffusion process.

Patch-level latent prediction error. We now restore the frame index t. For each latent spatial location (x, y), we first average the absolute clean-latent reconstruction error over the C latent channels:

$$\begin{aligned}E^{(\tau)}_{t,(x,y)} &= \frac{\sqrt{1-\bar{\alpha}_\tau}}{C}\sum_{c=1}^{C}\left|\hat{v}^{(\tau)}_{t,p} - v^{(\tau)}_{t,p}\right| \\ &= \frac{1}{C}\sum_{c=1}^{C}\left|\hat{z}^{(\tau)}_{t,p} - z^{(0)}_{t,p}\right|.\end{aligned} \tag{18}$$

Here $p = (c, x, y)$ denotes a latent entry; c indexes the latent channel and (x, y) denotes the spatial location on the latent grid. Thus, $E^{(\tau)}_t \in \mathbb{R}^{H_{\text{lat}} \times W_{\text{lat}}}$ forms a spatial error map for frame t.

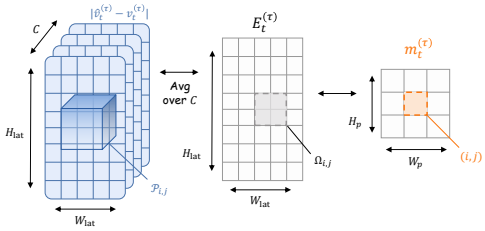


Figure 8: **Patch-level latent error construction.** Channel-wise latent prediction errors are first averaged into a spatial error map and then pooled over the region aligned with each confidence-probe patch.

As illustrated in Figure 8, let $\Omega_{i,j}$ denote the latent spatial region aligned with confidence-probe output location (i, j). The corresponding patch-level error is obtained by spatially averaging $E^{(\tau)}_t$ over this region. Equivalently, defining

$$\mathcal{P}_{i,j} = \{1, \ldots, C\} \times \Omega_{i,j}, \tag{19}$$

the same quantity can be written directly over all latent entries within the patch:

$$\begin{aligned}m^{(\tau)}_{t,(i,j)} &= \frac{1}{|\Omega_{i,j}|}\sum_{(x,y)\in\Omega_{i,j}} E^{(\tau)}_{t,(x,y)} \\ &= \frac{1}{C\cdot|\Omega_{i,j}|}\sum_{(x,y)\in\Omega_{i,j}}\sum_{c=1}^{C}\left|\hat{z}^{(\tau)}_{t,p} - z^{(0)}_{t,p}\right| \\ &= \frac{1}{|\mathcal{P}_{i,j}|}\sum_{p\in\mathcal{P}_{i,j}}\left|\hat{z}^{(\tau)}_{t,p} - z^{(0)}_{t,p}\right|.\end{aligned} \tag{20}$$

Hence, $m^{(\tau)}_t = \{m^{(\tau)}_{t,(i,j)}\}_{i,j} \in \mathbb{R}^{H_p \times W_p}$ has the same spatial indexing as the confidence output.

Binary confidence target. The binary supervision target is obtained by comparing each local prediction error with the adaptive threshold θ_t:

$$q_{t,(i,j)} = \mathbb{I}\left\{m^{(\tau)}_{t,(i,j)} < \theta_t\right\}, \qquad q_t = \left\{q_{t,(i,j)}\right\}_{i,j} \in \{0,1\}^{H_p \times W_p}. \tag{21}$$

A value $q_{t,(i,j)} = 1$ indicates that the corresponding latent patch is treated as reliable, while $q_{t,(i,j)} = 0$ denotes a high-error prediction. The adaptive construction of θ_t is described in the following subsection.

UNet feature tapping and confidence prediction. The denoising UNet contains multiple encoder blocks, a bottleneck, and multiple decoder blocks:

$$z^{(\tau)} \to h_{\text{enc}}^{(1)} \to \cdots \to h_{\text{enc}}^{(K)} \to h_{\text{mid}} \to h_{\text{dec}}^{(1)} \to \cdots \to h_{\text{dec}}^{(L)} \to \hat{v}^{(\tau)}. \tag{22}$$

Our confidence probe taps a selectable decoder feature $h_{\text{dec}}^{(\ell)}$, where the layer index ℓ controls the balance between global context and spatial locality. Earlier decoder layers provide coarser, more global representations, whereas later layers retain finer spatial information.

For future frame t, the dense confidence prediction is

$$\hat{q}_t = \sigma\left(f_\phi\left(h_{\text{dec}}^{(\ell)}, e_\tau, e_\theta\right)\right), \qquad \hat{q}_t \in [0,1]^{H_p \times W_p}, \tag{23}$$

where e_τ denotes the diffusion-timestep embedding, e_θ denotes the sampled error threshold embedding, f_ϕ is the trainable confidence probe and $\sigma(\cdot)$ denotes the element-wise sigmoid function. Each output location (i, j) is spatially aligned with $\Omega_{i,j}$, and therefore predicts the confidence associated with the local error $m_{t,(i,j)}^{(\tau)}$.

A.2 EMA-Calibrated Random Threshold Supervision

The numerical scale of the local prediction error can vary across datasets and training regimes, making a fixed confidence threshold difficult to transfer. We therefore estimate the threshold interval online from the batch-wise error distribution and stabilize it using exponential moving averages (EMA). For notational simplicity, the derivation below considers a fixed training sample and omits the sample index; the percentile statistics $p_l^{(s)}$ and $p_h^{(s)}$ are still computed from all local errors in the current training batch.

Batch-wise percentile estimation. At training step s, let $m^{(s)}$ denote the collection of patch-level latent errors in the current batch, where each element follows the definition of $m_{t,(i,j)}^{(\tau)}$ in Equation 20. We estimate the lower and upper error statistics as

$$p_l^{(s)} = \text{Quantile}_{0.1}\left(m^{(s)}\right), \qquad p_h^{(s)} = \text{Quantile}_{0.9}\left(m^{(s)}\right). \tag{24}$$

These percentiles adapt the supervision range to the current error scale without manually specifying dataset-dependent MAE thresholds.

Two-stage EMA update. The lower and upper threshold bounds are updated as

$$l^{(s)} = (1-\gamma_s) l^{(s-1)} + \gamma_s p_l^{(s)}, \qquad h^{(s)} = (1-\gamma_s) h^{(s-1)} + \gamma_s p_h^{(s)}. \tag{25}$$

We use a two-stage update schedule,

$$\gamma_s = \begin{cases} 0.20, & s < S_{\text{warm}}, \\ 0.002, & s \geq S_{\text{warm}}, \end{cases} \qquad \beta_s = 1 - \gamma_s = \begin{cases} 0.80, & s < S_{\text{warm}}, \\ 0.998, & s \geq S_{\text{warm}}, \end{cases} \tag{26}$$

where γ_s is the injection rate of the current batch statistics and β_s is the equivalent EMA momentum. Thus, the conventional EMA form is $\mu^{(s)} = \beta_s \mu^{(s-1)} + (1-\beta_s) p^{(s)}$. The larger update rate during warmup rapidly adapts the threshold range to the target-domain error scale, while the smaller rate afterwards tracks slower distributional changes with stronger smoothing.

Random threshold sampling and interpretation. After updating the bounds, a separate threshold is sampled for each predicted future frame t:

$$\theta_t^{(s)} \sim \mathcal{U}\left(l^{(s)}, h^{(s)}\right), \qquad q_{t,(i,j)}^{(s)} = \mathbb{I}\left\{m_{t,(i,j)}^{(\tau)} < \theta_t^{(s)}\right\}. \tag{27}$$

The same $\theta_t^{(s)}$ is shared by all spatial patches (i, j) within frame t, while different future frames receive independently sampled thresholds.

For a fixed local error m, the binary target becomes random only through $\theta_t^{(s)}$. Since $\theta_t^{(s)} \sim \mathcal{U}(l^{(s)}, h^{(s)})$, its density is $1/(h^{(s)} - l^{(s)})$ over the current threshold interval. Marginalizing over the sampled threshold gives

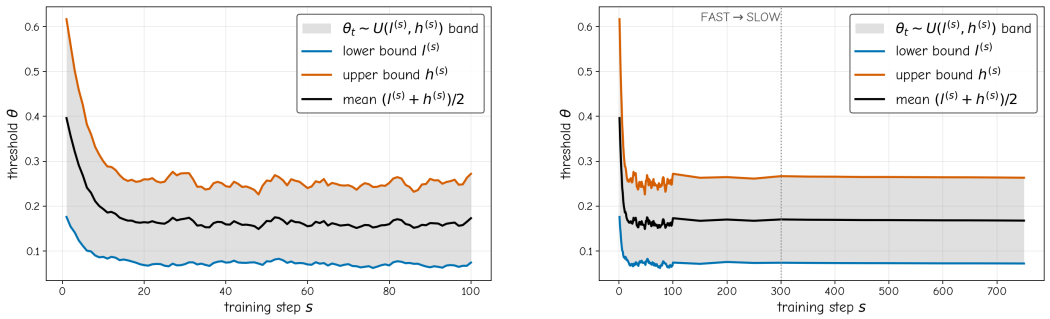
$$\begin{aligned}\mathbb{E}_\theta[q \mid m] = \mathbb{P}_\theta(m < \theta) &= \int_{l^{(s)}}^{h^{(s)}} \mathbb{I}\{m < \theta\} \frac{1}{h^{(s)} - l^{(s)}} \, \mathrm{d}\theta \\ &= \frac{\left[h^{(s)} - \max\{m, l^{(s)}\}\right]_+}{h^{(s)} - l^{(s)}} = \mathrm{clamp}\left(\frac{h^{(s)} - m}{h^{(s)} - l^{(s)}}, 0, 1\right) \\ &= \begin{cases} 1, & m \le l^{(s)}, \\ \dfrac{h^{(s)} - m}{h^{(s)} - l^{(s)}}, & l^{(s)} < m < h^{(s)}, \\ 0, & m \ge h^{(s)}. \end{cases}\end{aligned} \tag{28}$$

Hence, although each training target is binary, averaging over random thresholds induces a continuous target that decreases monotonically with prediction error.

In the threshold-marginalized case, the optimal prediction under binary cross-entropy is this conditional probability (using $0 \log 0 = 0$):

$$\begin{aligned}\hat{q}^*(m) &= \arg\min_{\hat{q} \in [0,1]} \mathbb{E}_{q|m}\left[-q \log \hat{q} - (1 - q) \log(1 - \hat{q})\right] \\ &= \mathbb{E}[q \mid m] = \mathrm{clamp}\left(\frac{h^{(s)} - m}{h^{(s)} - l^{(s)}}, 0, 1\right).\end{aligned} \tag{29}$$

In our implementation, $\theta_t^{(s)}$ is additionally embedded into the confidence probe as described in the main method, so the probe learns threshold-conditioned reliability rather than only this marginalized form. Equation 28 describes the threshold-marginalized target and does not determine the threshold-conditioned probe output.



(a) Zoom-in of the first 100 training steps

(b) Extended training-step evolution

Figure 9: **Dynamics of the EMA-based adaptive threshold band.** The probe uses a stochastic threshold $\theta \sim \mathcal{U}(l^{(s)}, h^{(s)})$, where $l^{(s)}$ and $h^{(s)}$ are the EMA-tracked lower and upper bounds at training step s, and the black curve shows their midpoint $(l^{(s)} + h^{(s)})/2$. (a) During the first 100 steps, both bounds decrease rapidly and quickly approach a stable range, reflecting fast adaptation to the current error scale. (b) Over a longer set of sampled training steps, the threshold band shows a clear transition from a fast-changing warmup regime to a slower and more stable regime.

Implementation and EMA dynamics. For RoboTwin2.0, we initialize $l^{(0)} = 0.20$ and $h^{(0)} = 0.70$, use $S_{\text{warm}} = 300$, and continue updating the EMA bounds throughout the 6000-step probe training. The bounds eventually stabilize around $l \approx 0.0705$ and $h \approx 0.2610$, showing how adaptive supervision automatically tracks the local error scale as training proceeds on this dataset and backbone.

Figure 9 visualizes the adaptation process. During the initial stage, the threshold band rapidly contracts toward the observed target-domain error scale. Over a longer range of sampled training steps, the evolution becomes substantially slower and more stable, showing the intended transition from fast initialization to long-term tracking.

B More Experimental Results

B.1 Additional Confidence Evaluation Details

This subsection supplements the confidence evaluation in Section 4.2. We report additional results for alternative risk aggregations, spatial localization, operational calibration, and parameter sensitivity. Latent-space errors remain our primary oracle, while pixel-space results are included as an external visual-domain reference.

Alternative task-level risk aggregations. Figure 10 compares mean, tail, and persistent risk aggregation in latent and pixel spaces. Mean risk achieves the strongest latent-space Spearman correlation because averaging suppresses local noise and measures the overall prediction difficulty of an episode. Tail risk instead emphasizes sparse severe errors, while persistent risk targets failures that recur across multiple frames; consequently, they need not correlate as strongly with the episode-wide mean error. These correlations therefore verify that each score contains a meaningful error signal, but do not by themselves determine which acquisition rule yields the best post-training model. Their downstream differences are evaluated separately in Table 3.

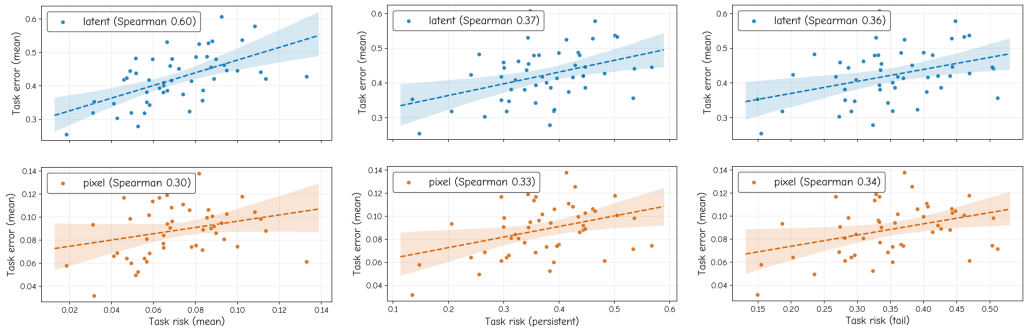
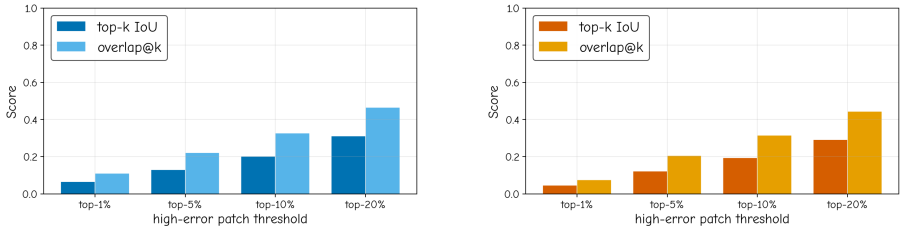


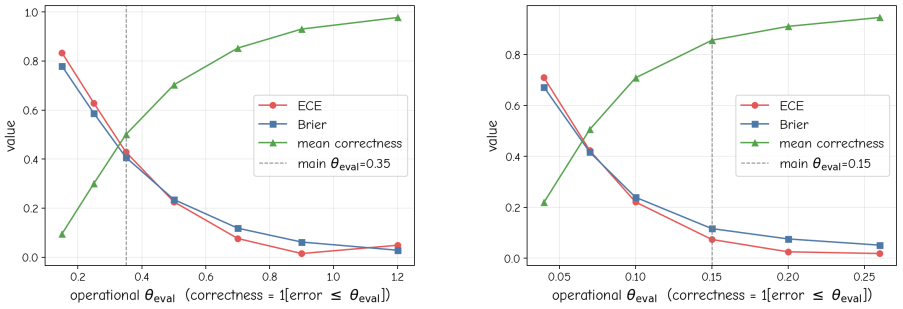
Figure 10: **Task-level risk aggregation in latent and pixel spaces.** Mean, tail, and persistent risk scores are compared with the mean oracle-error statistics over the 50 prescreened tasks. Dashed lines show linear fits, shaded regions denote confidence intervals, and each panel reports the Spearman correlation. Mean aggregation gives the strongest latent-space correlation, whereas tail and persistent aggregation emphasize different failure patterns.

Spatial localization agreement. We further compare the spatial locations selected by risk and oracle error. For each ratio k, top-k IoU measures the intersection-over-union between the highest-risk and highest-error patch sets, while overlap@k measures the fraction of high-error patches recovered by the high-risk set. Figure 11 shows that both metrics increase as the selected region becomes larger. At top-5%, latent-space localization reaches approximately 0.13 IoU and 0.22 overlap, with slightly lower values in pixel space. The limited spatial overlap and moderate IoU indicate that risk identifies some relevant error regions without exactly reproducing their boundaries. This supports soft patch weighting rather than hard spatial selection.



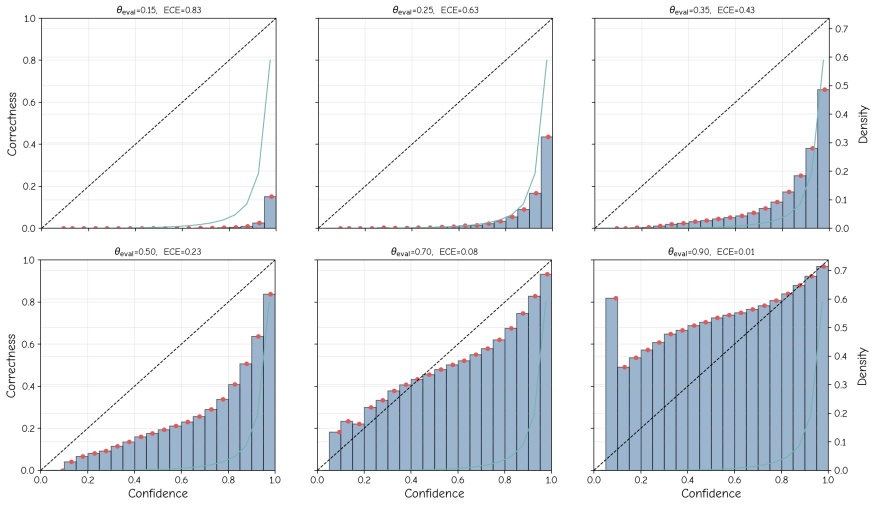
(a) Latent-space localization (b) Pixel-space localization

Figure 11: **Spatial agreement between high-risk and high-error patches.** Top-k IoU and overlap@k are reported for the top-1%, 5%, 10%, and 20% patches in (a) latent and (b) pixel spaces. The increasing scores at larger ratios reflect coarser spatial coverage and should not be interpreted as improved localization resolution.

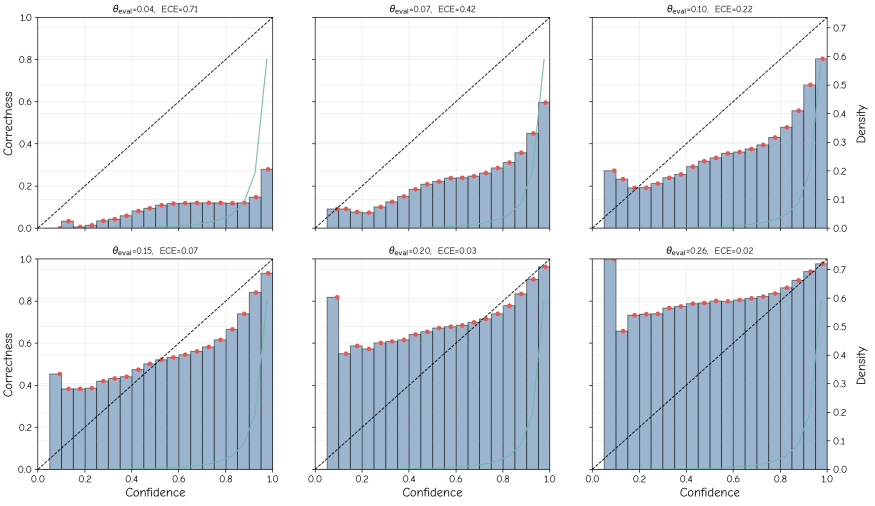


(a) Latent-space calibration sweep (b) Pixel-space calibration sweep

Figure 12: **ECE and Brier scores under different operational thresholds.** Calibration is evaluated by sweeping the correctness threshold θ_{eval} in (a) latent space and (b) pixel space. The confidence maps are fixed throughout; only the binary correctness definition changes.



(a) Latent-space reliability diagrams



(b) Pixel-space reliability diagrams

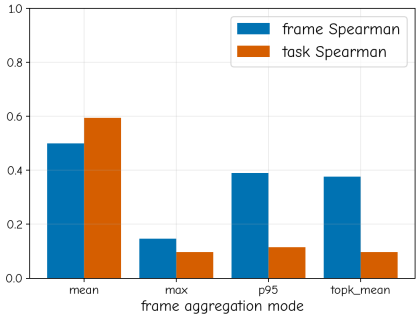
Figure 13: **Reliability diagrams across operational thresholds.** Each subfigure contains reliability diagrams under different choices of θ_{eval}: (a) latent space and (b) pixel space. As the threshold becomes looser, the observed correctness generally moves closer to the predicted confidence, although the latent-space probe remains more overconfident near the main operating point.

Operational calibration. Operational calibration evaluates fixed confidence maps against generated-rollout errors. We denote the operational threshold by θ_{eval} and define correctness as $\mathbb{I}\{e^{\text{roll}}_{t,(i,j)} \leq \theta_{\text{eval}}\}$. Here, $e^{\text{roll}}_{t,(i,j)}$ denotes channel-averaged absolute rollout error, spatially pooled to the probe grid, in latent or pixel space as indicated. Varying θ_{eval} changes the post-hoc correctness labels while keeping the predicted confidence maps fixed.

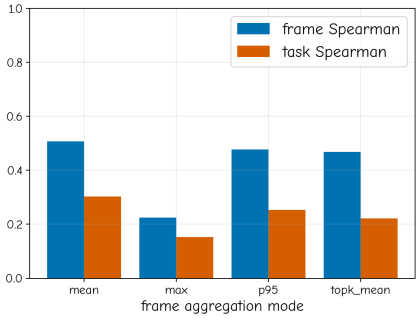
Figure 12 summarizes calibration quality under different choices of θ_{eval}. In latent space, ECE and Brier improve as the threshold becomes looser, but the probe remains noticeably overconfident around the main operating point. Pixel-space calibration appears better under its own selected threshold, although the latent and pixel settings are not directly comparable because they rely on different error scales. Figure 13 further visualizes the corresponding reliability diagrams. In both spaces, the curves become closer to the diagonal as θ_{eval} increases, confirming that the apparent calibration quality depends strongly on the operational correctness definition. These 50-episode operational sweeps use rollout MAE; Appendix B.3 evaluates matched thresholds on 250 episodes using single-step MAE.

Parameter sensitivity. We examine two implementation choices that are not reported in the main text. First, Figures 14(a–b) compare mean, maximum, percentile, and top-k frame aggregation. Mean aggregation provides the strongest frame- and task-level Spearman correlations in both latent and pixel spaces, whereas extreme-value aggregation amplifies isolated noisy patches. This motivates the mean aggregation adopted in our main evaluation.

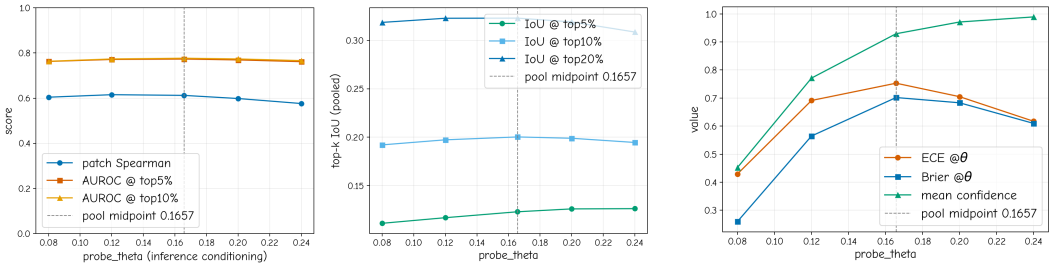
Second, Figures 14(c–d) vary the probe conditioning threshold θ_{cond} and regenerate the confidence maps. Low-to-moderate thresholds provide the strongest spatial discrimination, while large thresholds make most patches highly confident and reduce localization contrast. The default threshold lies near the best-performing range, and the overall variation remains moderate, indicating that the confidence-ranking conclusion is not sensitive to a narrowly tuned threshold.



(a) Frame aggregation in latent space



(b) Frame aggregation in pixel space



(c) Localization sensitivity

(d) Calibration sensitivity

Figure 14: **Sensitivity to confidence aggregation and probe conditioning.** (a–b) Frame- and task-level Spearman correlations under mean, maximum, percentile, and top-k aggregation in latent and pixel spaces. (c) High-error detection and spatial localization under different probe conditioning thresholds θ_{cond}. (d) ECE, Brier score, and mean confidence under the same thresholds. Unlike the operational threshold θ_{eval}, changing θ_{cond} regenerates the confidence maps.

B.2 Additional Numerical Results for Active Learning

This subsection provides the complete numerical results underlying the active-learning comparison in Section 4.3. We first report the four aggregated dimensions used in the main comparison, followed by paired-bootstrap improvements relative to EVAC-v1. We then provide the complete seed-wise component metrics for seeds 42, 3407, and 123.

Aggregated results and paired-bootstrap evidence. Table 4 reports the aggregated active-learning results. Reconstruction, Scene, Motion, and Semantics follow the same definitions as in Section 4.3. For learned scoring methods, the reported values are averaged over seeds 42, 3407, and 123. Table 5 further reports paired-bootstrap improvements over EVAC-v1. For each learned scoring method, we concatenate the episode-level paired differences from all three seeds and perform 10,000 percentile-bootstrap resamples at 95% confidence. The pairing with EVAC-v1 is preserved within each seed before pooling. Pooled intervals quantify episode-level variation; the seed-wise tables report the results of each individual run.

Table 4: Aggregated active-learning results. Each entry reports the aggregated mean followed by its relative change from EVAC-v1. Results for learned scoring methods are averaged over three seeds.

Scoring	Weighting	Reconstruction (↑)	Scene (↑)	Motion (↑)	Semantics (↑)
Base EVAC		0.5655 (-17.8%)	0.8757 (-3.2%)	0.0014 (-99.6%)	0.4873 (-14.3%)
Base EVAC (Warmup v1)		0.6878 (0.0%)	0.9047 (0.0%)	0.3205 (0.0%)	0.5690 (0.0%)
Selection-only retraining: no additional weighting after selection					
Random	None	0.6409 (-6.8%)	0.8524 (-5.8%)	0.3310 (+3.3%)	0.4872 (-14.4%)
RoboReward[1]	None	0.6712 (-2.4%)	0.8470 (-6.4%)	0.3360 (+4.8%)	**0.6022 (+5.8%)**
GVL[2]	None	0.7088 (+3.1%)	0.9057 (+0.1%)	0.3393 (+5.9%)	0.5913 (+3.9%)
Robometer-Prog[3]	None	0.6317 (-8.2%)	0.8532 (-5.7%)	0.3307 (+3.2%)	0.5759 (+1.2%)
Robometer-Pref[3]	None	0.6474 (-5.9%)	0.8443 (-6.7%)	0.3208 (+0.1%)	0.5789 (+1.7%)
PRM-as-Judge[4]	None	0.7024 (+2.1%)	0.9033 (-0.2%)	0.3505 (+9.4%)	0.5922 (+4.1%)
LRMs[5]	None	0.7110 (+3.4%)	0.8966 (-0.9%)	0.2885 (-10.0%)	0.5776 (+1.5%)
Confidence (Ours)	**None**	**0.7219 (+5.0%)**	**0.9196 (+1.6%)**	**0.3643 (+13.7%)**	0.5897 (+3.6%)
Selection + additional weighting: extra scoring after selection					
Robometer-Prog[3]	Frame	0.6944 (+1.0%)	0.8882 (-1.8%)	0.3357 (+4.7%)	0.5884 (+3.4%)
Robometer-Pref[3]	Frame	0.6803 (-1.1%)	0.8945 (-1.1%)	0.2969 (-7.4%)	0.5601 (-1.6%)
PRM-as-Judge[4]	Frame	0.7123 (+3.6%)	0.9053 (+0.1%)	0.3347 (+4.4%)	0.5963 (+4.8%)
LRMs[5]	Frame	0.6955 (+1.1%)	0.8882 (-1.8%)	0.3171 (-1.1%)	0.5815 (+2.2%)
Confidence (Ours)	**Frame**	0.7047 (+2.5%)	**0.9143 (+1.1%)**	0.3398 (+6.0%)	0.5944 (+4.5%)
Confidence (Ours)	**Fr.+Patch**	**0.7265 (+5.6%)**	0.8942 (-1.2%)	**0.3510 (+9.5%)**	**0.6034 (+6.1%)**

Table 5: Paired-bootstrap improvements over EVAC-v1. For each learned scoring method, episode-level paired differences from seeds 42, 3407, and 123 are pooled before bootstrap resampling. Each entry gives the 95% confidence interval followed by the paired mean in parentheses.

Scoring	Weighting	Δ Reconstruction (↑)	Δ Scene (↑)	Δ Motion (↑)	Δ Semantics (↑)
Base EVAC		[-0.133, -0.112] (-0.122)	[-0.038, -0.020] (-0.029)	–	[-0.113, -0.049] (-0.081)
Base EVAC (Warmup v1)		–	–	–	–
Selection-only retraining: no additional weighting after selection					
Random	None	[-0.055, -0.039] (-0.047)	[-0.060, -0.045] (-0.052)	[-0.035, +0.056] (+0.011)	[-0.112, -0.046] (-0.079)
RoboReward[1]	None	[-0.020, -0.013] (-0.017)	[-0.063, -0.052] (-0.058)	[-0.006, +0.038] (+0.015)	**[+0.016, +0.053] (+0.035)**
GVL[2]	None	[+0.018, +0.024] (+0.021)	[-0.001, +0.003] (+0.001)	[-0.008, +0.046] (+0.019)	[+0.003, +0.042] (+0.023)
Robometer-Prog[3]	None	[-0.061, -0.051] (-0.056)	[-0.056, -0.047] (-0.051)	[-0.017, +0.038] (+0.010)	[-0.014, +0.024] (+0.006)
Robometer-Pref[3]	None	[-0.046, -0.035] (-0.040)	[-0.067, -0.054] (-0.060)	[-0.024, +0.025] (+0.000)	[-0.012, +0.028] (+0.008)
PRM-as-Judge[4]	None	[+0.011, +0.018] (+0.015)	[-0.004, +0.001] (-0.001)	[+0.005, +0.056] (+0.030)	[+0.001, +0.040] (+0.021)
LRMs[5]	None	[+0.020, +0.027] (+0.023)	[-0.011, -0.006] (-0.008)	[-0.057, -0.008] (-0.032)	[-0.010, +0.029] (+0.010)
Confidence (Ours)	**None**	**[+0.031, +0.037] (+0.034)**	**[+0.012, +0.018] (+0.015)**	**[+0.018, +0.070] (+0.044)**	[-0.002, +0.038] (+0.018)
Selection + additional weighting: extra scoring after selection					
Robometer-Prog[3]	Frame	[+0.003, +0.011] (+0.007)	[-0.021, -0.012] (-0.017)	[-0.010, +0.041] (+0.015)	[-0.000, +0.038] (+0.019)
Robometer-Pref[3]	Frame	[-0.011, -0.004] (-0.007)	[-0.013, -0.007] (-0.010)	[-0.049, +0.002] (-0.024)	[-0.031, +0.008] (-0.011)
PRM-as-Judge[4]	Frame	[+0.021, +0.028] (+0.024)	[-0.003, +0.004] (+0.001)	[-0.010, +0.039] (+0.014)	[+0.008, +0.046] (+0.027)
LRMs[5]	Frame	[+0.004, +0.011] (+0.008)	[-0.020, -0.013] (-0.016)	[-0.027, +0.020] (-0.003)	[-0.005, +0.033] (+0.014)
Confidence (Ours)	**Frame**	[+0.014, +0.020] (+0.017)	**[+0.007, +0.012] (+0.010)**	[-0.007, +0.045] (+0.019)	[+0.007, +0.044] (+0.026)
Confidence (Ours)	**Fr.+Patch**	**[+0.036, +0.042] (+0.039)**	[-0.013, -0.008] (-0.010)	**[+0.008, +0.054] (+0.031)**	**[+0.011, +0.049] (+0.030)**

Seed-wise detailed results. Tables 6–8 report the complete normalized component metrics for seeds 42, 3407, and 123, respectively. Seed 42 is the default seed. Base EVAC, EVAC-v1, and Random are shared reference results and are repeated in each table for ease of comparison.

Table 6: Detailed normalized active-learning results for seed 42.

Scoring	Weighting	Reconstruction (↑)		Scene (↑)	Semantics (↑)			Motion (↑)		
		PSNR (↑)	SSIM (↑)	Scene Cons. (↑)	Logics (↑)	Sem.-CLIP (↑)	Sem.-BLEU (↑)	Traj-HSD (↑)	Traj-Dyn (↑)	Traj-nDTW (↑)
Base EVAC		0.5532	0.5778	0.8757	0.4298	0.8523	0.1799	0.0007	0.0001	0.0006
Base EVAC (Warmup v1)		0.6446	0.7309	0.9047	0.5537	0.8824	0.2708	0.1045	0.0721	0.1439
Selection-only retraining: no additional weighting after selection										
Random	None	0.5968	0.6849	0.8524	0.3554	0.8689	0.2372	0.1067	0.0725	0.1518
RoboReward[1]	None	0.6276	0.7067	0.8712	0.6322	0.8895	0.2832	0.1076	0.0662	0.1501
GVL[2]	None	0.6550	0.7449	0.9136	0.5826	0.8885	0.2733	0.1033	0.0607	0.1381
Robometer-Prog[3]	None	0.5547	0.6711	0.8208	0.5496	0.8800	0.2680	0.1166	0.0774	**0.1805**
Robometer-Pref[3]	None	0.6047	0.7005	0.9005	0.6157	0.8866	0.2729	0.0912	0.0492	0.1242
PRM-as-Judge[4]	None	0.6523	0.7641	0.9102	**0.6405**	**0.8926**	**0.3023**	0.1095	0.0696	0.1495
LRMs[5]	None	0.6577	0.7528	0.8869	0.5785	0.8803	0.2550	0.0960	0.0595	0.1422
Confidence (Ours)	**None**	**0.6838**	**0.7797**	**0.9322**	0.6198	0.8894	0.2754	**0.1313**	**0.0921**	0.1780
Selection + additional weighting: extra scoring after selection										
Robometer-Prog[3]	Frame	0.5932	0.6981	0.8399	0.6074	0.8883	**0.2976**	0.1151	0.0703	0.1655
Robometer-Pref[3]	Frame	0.6412	0.7324	0.8948	0.6033	0.8868	0.2805	0.0981	0.0678	0.1291
PRM-as-Judge[4]	Frame	0.6695	0.7733	0.9193	0.6281	0.8880	0.2937	0.1000	0.0592	0.1443
LRMs[5]	Frame	0.6716	0.7616	0.8958	0.5785	0.8896	0.2733	0.0985	0.0623	0.1497
Confidence (Ours)	**Frame**	0.6737	0.7650	**0.9295**	**0.6405**	0.8867	0.2843	**0.1179**	0.0728	**0.1706**
Confidence (Ours)	**Fr.+Patch**	**0.6795**	**0.7746**	0.9014	0.6240	**0.8945**	0.2859	0.1066	**0.0761**	0.1665

Table 7: Detailed normalized active-learning results for seed 3407.

Scoring	Weighting	Reconstruction (↑)		Scene (↑)	Semantics (↑)			Motion (↑)		
		PSNR (↑)	SSIM (↑)	Scene Cons. (↑)	Logics (↑)	Sem.-CLIP (↑)	Sem.-BLEU (↑)	Traj-HSD (↑)	Traj-Dyn (↑)	Traj-nDTW (↑)
Base EVAC		0.5532	0.5778	0.8757	0.4298	0.8523	0.1799	0.0007	0.0001	0.0006
Base EVAC (Warmup v1)		0.6446	0.7309	0.9047	0.5537	0.8824	0.2708	0.1045	0.0721	0.1439
Selection-only retraining: no additional weighting after selection										
Random	None	0.5968	0.6849	0.8524	0.3554	0.8689	0.2372	0.1067	0.0725	0.1518
RoboReward[1]	None	0.6262	0.7290	0.7985	**0.6488**	0.8863	0.2775	0.1089	0.0732	0.1554
GVL[2]	None	0.6546	0.7503	0.8959	0.6157	0.8892	0.2868	**0.1200**	0.0759	0.1681
Robometer-Prog[3]	None	0.6492	0.7490	0.8857	0.6033	0.8814	0.2580	0.1148	0.0715	0.1491
Robometer-Pref[3]	None	0.6442	0.7535	0.8779	0.6033	0.8895	0.2877	0.1066	0.0671	0.1565
PRM-as-Judge[4]	None	**0.6714**	**0.7728**	0.9020	0.5950	**0.8896**	0.2873	0.1197	**0.0784**	**0.1757**
LRMs[5]	None	0.6619	0.7658	0.8991	0.5331	0.8817	0.2672	0.0925	0.0564	0.1299
Confidence (Ours)	**None**	0.6521	0.7443	**0.9029**	0.5537	0.8891	**0.2953**	0.0998	0.0618	0.1469
Selection + additional weighting: extra scoring after selection										
Robometer-Prog[3]	Frame	0.6785	0.7739	**0.9148**	0.5661	0.8896	0.2925	0.1042	0.0683	0.1454
Robometer-Pref[3]	Frame	0.6329	0.7343	0.9000	0.5372	0.8875	0.2796	0.0867	0.0542	0.1241
PRM-as-Judge[4]	Frame	0.6544	0.7606	0.8816	0.6364	0.8861	0.2681	**0.1276**	**0.0949**	**0.1827**
LRMs[5]	Frame	0.6391	0.7323	0.8984	0.5868	0.8872	0.2884	0.1047	0.0582	0.1508
Confidence (Ours)	**Frame**	0.6644	0.7518	0.9127	0.6074	0.8887	0.2734	0.1121	0.0704	0.1663
Confidence (Ours)	**Fr.+Patch**	**0.6798**	**0.7822**	0.8948	**0.6612**	**0.8926**	**0.2979**	0.1161	0.0679	0.1665

Table 8: Detailed normalized active-learning results for seed 123.

Scoring	Weighting	Reconstruction (↑)		Scene (↑)	Semantics (↑)			Motion (↑)		
		PSNR (↑)	SSIM (↑)	Scene Cons. (↑)	Logics (↑)	Sem.-CLIP (↑)	Sem.-BLEU (↑)	Traj-HSD (↑)	Traj-Dyn (↑)	Traj-nDTW (↑)
Base EVAC		0.5532	0.5778	0.8757	0.4298	0.8523	0.1799	0.0007	0.0001	0.0006
Base EVAC (Warmup v1)		0.6446	0.7309	0.9047	0.5537	0.8824	0.2708	0.1045	0.0721	0.1439
Selection-only retraining: no additional weighting after selection										
Random	None	0.5968	0.6849	0.8524	0.3554	0.8689	0.2372	0.1067	0.0725	0.1518
RoboReward[1]	None	0.6260	0.7117	0.8712	**0.6364**	0.8825	0.2838	0.1174	0.0685	0.1607
GVL[2]	None	0.6795	0.7686	0.9075	0.5992	0.8918	0.2946	0.1109	0.0698	0.1713
Robometer-Prog[3]	None	0.5388	0.6272	0.8532	0.5992	0.8815	0.2620	0.0955	0.0582	0.1282
Robometer-Pref[3]	None	0.5267	0.6546	0.7544	0.5248	0.8816	0.2479	0.1160	0.0752	**0.1764**
PRM-as-Judge[4]	None	0.6164	0.7372	0.8977	0.5744	0.8817	0.2659	0.1168	0.0758	0.1565
LRMs[5]	None	0.6655	0.7624	0.9037	0.6116	0.8862	**0.3046**	0.0952	0.0603	0.1335
Confidence (Ours)	**None**	**0.6878**	**0.7836**	**0.9237**	0.6033	**0.8924**	0.2889	**0.1232**	**0.0896**	0.1702
Selection + additional weighting: extra scoring after selection										
Robometer-Prog[3]	Frame	0.6600	0.7626	0.9099	0.5909	0.8899	0.2734	0.1097	0.0706	**0.1582**
Robometer-Pref[3]	Frame	0.6210	0.7199	0.8887	0.4463	0.8804	0.2395	0.1073	0.0741	0.1493
PRM-as-Judge[4]	Frame	0.6514	0.7645	**0.9150**	**0.6033**	0.8862	0.2766	0.0971	0.0643	0.1339
LRMs[5]	Frame	0.6282	0.7404	0.8706	0.5826	0.8839	0.2634	**0.1128**	0.0700	0.1443
Confidence (Ours)	**Frame**	0.6403	0.7332	0.9008	**0.6033**	0.8867	0.2790	0.1019	0.0620	0.1455
Confidence (Ours)	**Fr.+Patch**	**0.6723**	**0.7705**	0.8864	0.5826	**0.8903**	**0.3017**	0.1126	**0.0837**	0.1570

B.3 Controlled Ablations of Features and Threshold Supervision

Experimental design. We evaluate two design choices for dense confidence estimation in EVAC: the feature layer used by the probe and the supervision threshold range. Following the C^3 formulation (Mei et al., 2026), the probe predicts dense confidence from binary targets constructed by thresholding local latent errors. We implement this formulation on a frozen EVAC backbone and compare bottleneck and decoder features under fixed-range and EMA supervision. Throughout this subsection, q denotes the binary confidence target and $\hat{q}$ denotes the predicted confidence, following Section 3.1.

Backbone, data, and probe training. All variants use the same EVAC-v1 checkpoint after 2,000 warmup updates. We partition the original 6,248-episode probe/warmup subset into 5,992 probe-training episodes and 256 calibration episodes using deterministic task-stratified selection. The calibration episodes participated in the earlier EVAC-v1 warmup and are held out during probe optimization. Evaluation uses a separate set of 250 held-out episodes spanning 43 task labels represented in the dataset. Threshold-range selection uses the calibration set, and all variants are evaluated at the final 6,000-step checkpoint.

For each episode, we cache one 20-frame clip consisting of four conditioning frames and $T = 16$ future frames, together with one noise realization and the frozen EVAC features. The integer diffusion timestep is sampled from $\tau \in \{150, \dots, 599\}$ for training and matched-noise evaluation. All variants share the cached examples. We use the local latent error $m^{(\tau)}_{t,(i,j)}$ defined in Section 3.1 and Appendix A.1: the channel-averaged absolute error between the single-step recovered clean latent $\hat{z}^{(\tau)}$ and the ground-truth clean latent $z^{(0)}$, spatially pooled to probe patch (i, j). Thresholding this error gives the binary confidence target $q_{t,(i,j)}$.

We compare bottleneck features h_{mid} at spatial resolution 5×8 with decoder features $h^{(\ell)}_{\text{dec}}$ at $\ell = 5$ and resolution 20×32; both have 1,280 channels. Bottleneck features are bilinearly resized to 20×32 before entering the shared probe architecture f_ϕ. Both variants therefore use the same output grid, with $H_p = 20$ and $W_p = 32$, the same confidence targets, and 8,157,185 trainable parameters. The probe consists of a 256-dimensional projection, four attention heads, three spatial and three temporal Transformer layers, and diffusion-timestep and threshold conditioning through e_τ and e_θ. Each variant is trained with seed 42, batch size four, and 6,000 AdamW updates, using a learning rate of 10^{-3}, weight decay of 0.01, 300 warmup updates, cosine decay, gradient clipping at 1.0, and bf16 computation. Initialization and training random streams are matched across variants.

Threshold supervision. Let Q_ρ denote the ρ-quantile of pooled local latent errors in the training cache. We consider four fixed supervision ranges: $[Q_{.10}, Q_{.90}] = [0.064462, 0.240203]$, $[Q_{.05}, Q_{.95}] = [0.053167, 0.294761]$, $[Q_{.20}, Q_{.80}] = [0.080663, 0.186931]$, and the manually specified range $[0.20, 0.70]$. Within each fixed range, a threshold θ_t is sampled uniformly and independently for each sample and future frame, and is shared by all spatial patches within that frame.

EMA supervision follows the update rule in Section 3.1 and Appendix A.2. The bounds are initialized at $l^{(0)} = 0.20$ and $h^{(0)} = 0.70$ and updated using the batch-wise 10th and 90th error percentiles. The update rate γ_s is 0.20 during the first $S_{\text{warm}} = 300$ updates and 0.002 thereafter. At each training step s, thresholds are sampled as $\theta^{(s)}_t \sim \mathcal{U}(l^{(s)}, h^{(s)})$. An additional variant, EMA-Qinit, initializes the bounds at $[Q_{.10}, Q_{.90}]$ to compare fixed and adaptive supervision from the same starting range. Combining two feature choices with six supervision settings yields 12 probe fits.

We select the fixed range by minimizing calibration-set Brier averaged across the two feature choices. This procedure selects $[Q_{.05}, Q_{.95}]$, which is also selected when each feature choice is calibrated separately.

Common-threshold evaluation. All probes are evaluated at the same nine absolute thresholds:

$$\theta_k = Q_{.05} + \frac{k-1}{8}(Q_{.95} - Q_{.05}), \qquad k = 1, \dots, 9. \tag{30}$$

Here, k indexes evaluation thresholds, while t continues to index future frames. We add an episode index e to the notation of Section 3.1. At each θ_k, the probe produces the predicted confidence

Table 9: **Controlled feature and supervision comparison on EVAC-v1.** All rows use seed 42 and identical probe capacity. Tuned fixed denotes the calibration-selected $[Q_{.05}, Q_{.95}]$ range. Scores are computed on 250 test episodes with $\tau \in \{150, \ldots, 599\}$. Lower Brier, BCE, and ECE and higher AUROC indicate better performance. Boldface marks the best values among the four configurations.

Feature	Supervision	Brier ↓	BCE ↓	ECE ↓	AUROC ↑
Bottleneck	Tuned fixed	0.089487	0.288859	0.00567	0.78379
Bottleneck	EMA	0.089551	0.289220	0.00639	0.78487
Decoder	Tuned fixed	**0.084181**	**0.271915**	**0.00402**	**0.82237**
Decoder	EMA	0.084243	0.272314	0.00643	0.82184

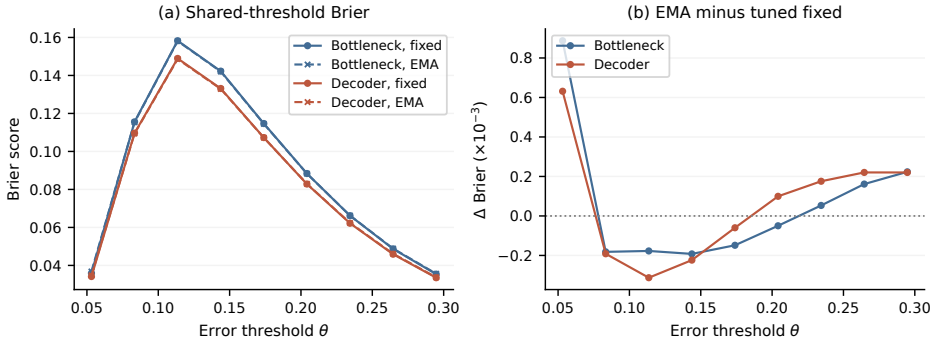


Figure 15: **Probe performance at shared thresholds.** (a) Brier scores for the four core configurations. (b) EMA-minus-tuned-fixed Brier at each threshold; positive values favor fixed supervision. Fixed and EMA supervision yield closely aligned curves, with a larger separation between decoder and bottleneck features. All configurations use the same binary targets $q_{e,t,(i,j)}(\theta_k)$, constructed from held-out local latent errors.

$\hat{q}_{e,t,(i,j)}(\theta_k)$ using the corresponding threshold embedding e_{θ_k}. Its binary evaluation target is

$$q_{e,t,(i,j)}(\theta_k) = \mathbb{I}\{m^{(\tau)}_{e,t,(i,j)} < \theta_k\}.$$

The diffusion timestep τ is the one sampled for the evaluated example; its dependence is implicit in q and $\hat{q}$, as in the main text. Using the terminology of Appendix B.1, this evaluation sets $\theta_{\text{cond}} = \theta_{\text{eval}} = \theta_k$ at each query, so the predicted confidence and binary target refer to the same error threshold. The primary metric is the episode-averaged Brier score:

$$\text{Brier} = \frac{1}{N_{\text{ep}}} \sum_{e=1}^{N_{\text{ep}}} \frac{1}{9THW} \sum_{k=1}^{9} \sum_{t=1}^{T} \sum_{i=1}^{H_p} \sum_{j=1}^{W_p} \big(\hat{q}_{e,t,(i,j)}(\theta_k) - q_{e,t,(i,j)}(\theta_k)\big)^2, \tag{31}$$

where N_{ep} is the number of evaluated episodes and $THW = 16 \cdot 20 \cdot 32$.

We also report BCE, ECE, and AUROC. ECE uses 15 equal-width probability bins, pools frame/patch positions across episodes at each threshold, and averages the resulting values over thresholds. AUROC is computed for each episode/threshold pair and averaged over pairs containing both classes. The shared grid provides common conditioning values and evaluation labels for all variants. It spans the training-error interval $[Q_{.05}, Q_{.95}]$, including queries outside the supervision range of variants trained on narrower or shifted intervals.

Effect of feature choice. Decoder features improve probe quality under both supervision rules (Table 9). With tuned fixed supervision, replacing bottleneck features with decoder features reduces Brier from 0.089487 to 0.084181, a relative reduction of 5.93%, and increases AUROC from 0.78379 to 0.82237. With EMA supervision, Brier decreases from 0.089551 to 0.084243, also a 5.93% relative reduction. These improvements are measured directly on probe predictions before acquisition and weighted retraining.

Effect of threshold supervision. EMA and tuned fixed supervision yield similar Brier scores across the shared threshold grid (Figure 15). For decoder features, Brier is 0.084243 with EMA and

Table 10: **Complete single-seed range ablation.** All 12 fits use seed 42. Quantiles Q_ρ are computed from training errors. The dagger marks the fixed range selected by calibration Brier, both when averaging across feature choices and when selecting separately for each feature. EMA-Qinit starts from $[Q_{.10}, Q_{.90}]$. Test columns use $\tau \in \{150, \ldots, 599\}$; the final column evaluates the same episodes with $\tau \in \{50, \ldots, 199\}$.

Feature	Supervision	Cal. Brier	Test Brier	Test ECE	Test AUROC	Low-noise Brier
Bottleneck	Fixed $[0.20, 0.70]$	0.154102	0.153483	0.12854	0.77596	0.090326
Bottleneck	Fixed $[Q_{.10}, Q_{.90}]$	0.092173	0.089468	0.00670	0.78338	0.057385
Bottleneck	Fixed $[Q_{.05}, Q_{.95}]^\dagger$	0.092149	0.089487	0.00567	0.78379	0.057210
Bottleneck	Fixed $[Q_{.20}, Q_{.80}]$	0.092776	0.090206	0.01182	0.78185	0.058627
Bottleneck	EMA	0.092239	0.089551	0.00639	0.78487	0.057544
Bottleneck	EMA-Qinit	0.092101	0.089477	0.00694	0.78517	0.057503
Decoder	Fixed $[0.20, 0.70]$	0.133136	0.133257	0.10630	0.81088	0.085256
Decoder	Fixed $[Q_{.10}, Q_{.90}]$	0.086885	0.084235	0.00617	0.82313	0.054462
Decoder	Fixed $[Q_{.05}, Q_{.95}]^\dagger$	0.086872	0.084181	0.00402	0.82237	0.054344
Decoder	Fixed $[Q_{.20}, Q_{.80}]$	0.087667	0.085248	0.01142	0.82040	0.056637
Decoder	EMA	0.086867	0.084243	0.00643	0.82184	0.054400
Decoder	EMA-Qinit	0.086840	0.084266	0.00548	0.82235	0.054522

Table 11: **Paired task-macro Brier contrasts.** All values, including interval endpoints, are multiplied by 10^3. Intervals are percentile 95% task-bootstrap intervals at seed 42. Negative differences favor the first named configuration. Let $B_{a,b}$ denote task-macro Brier for feature choice a and supervision rule b, with dec and mid denoting decoder and bottleneck features. The interaction is $(B_{\mathrm{dec,EMA}} - B_{\mathrm{mid,EMA}}) - (B_{\mathrm{dec,fixed}} - B_{\mathrm{mid,fixed}})$.

	Training-noise range		Low-noise range	
Contrast	Difference	95% CI	Difference	95% CI
Decoder − bottleneck (fixed)	-5.331	$[-5.657, -4.987]$	-2.767	$[-3.156, -2.243]$
Decoder − bottleneck (EMA)	-5.294	$[-5.637, -4.934]$	-3.141	$[-3.475, -2.762]$
EMA − fixed (decoder)	$+0.072$	$[+0.011, +0.134]$	-0.016	$[-0.151, +0.091]$
EMA − fixed (bottleneck)	$+0.035$	$[-0.071, +0.143]$	$+0.358$	$[+0.270, +0.449]$
Feature × supervision	$+0.037$	$[-0.090, +0.170]$	-0.374	$[-0.562, -0.229]$

0.084181 with the calibration-selected fixed range, an episode-mean difference of $+6.22 \times 10^{-5}$. The fixed range also gives lower ECE, at 0.00402 compared with 0.00643 for EMA.

The full range ablation shows a larger effect of supervision-range choice (Table 10). The manual range $[0.20, 0.70]$ yields a decoder Brier of 0.133257, compared with 0.084181–0.085248 for the three training-quantile ranges. Initializing EMA at $[Q_{.10}, Q_{.90}]$ gives a Brier of 0.084266, close to the 0.084235 obtained by keeping those bounds fixed. Thus, training-error quantiles provide effective supervision ranges in this setting, and EMA tracks the error scale online with performance close to the calibrated fixed alternatives.

Paired task-level comparisons. We estimate uncertainty across tasks by averaging episode Brier within each of the 43 test tasks, computing paired method differences, and resampling tasks with replacement 3,000 times. Table 11 reports task-macro differences and percentile 95% confidence intervals. Task-macro estimates weight each task equally, while the episode means in Table 9 weight tasks according to their episode counts. The intervals are conditional on seed 42 and use unadjusted percentile bounds for the exploratory comparisons.

The decoder-minus-bottleneck intervals lie below zero under both supervision rules. Within the training-noise range, the decoder EMA-minus-fixed difference is small and positive, with an interval above zero, indicating a slight advantage for fixed supervision. The corresponding bottleneck interval includes zero. Figure 16 shows that the feature effect is substantially larger than the difference between EMA and tuned fixed supervision.

Evaluation at lower noise levels. We further evaluate the same test episodes at $\tau \in \{50, \ldots, 199\}$ with fixed probes and threshold grids. This range partially overlaps the training range and is closer

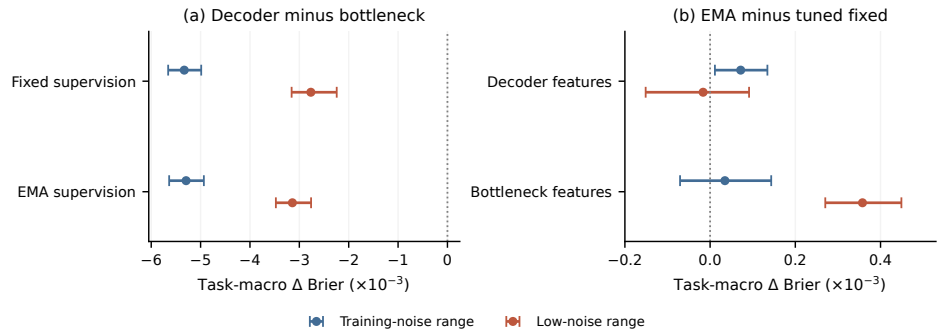


Figure 16: **Feature and supervision effects across tasks.** (a) Decoder-minus-bottleneck Brier contrasts under fixed and EMA supervision. (b) EMA-minus-tuned-fixed contrasts for each feature choice. The panels use different horizontal scales to display the respective effect sizes. Error bars indicate percentile 95% task-bootstrap intervals at seed 42.

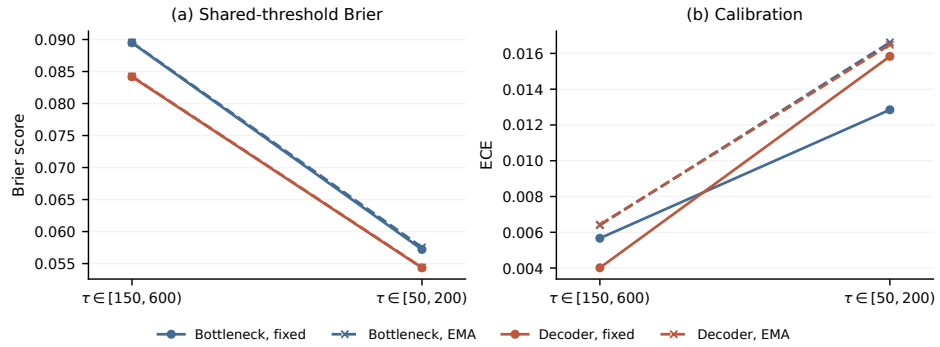


Figure 17: **Noise-range transfer on held-out episodes.** Brier (left) and ECE (right) are evaluated on the same episodes with fixed probe parameters and threshold ranges. Decoder features achieve lower Brier in both noise ranges. Low-noise evaluation shows lower Brier but higher ECE due to changes in local-error distributions and target balance.

to acquisition-time scoring. Decoder features achieve lower Brier under both supervision rules: 0.054344 versus 0.057210 with fixed supervision, and 0.054400 versus 0.057544 with EMA. The decoder EMA-minus-fixed task-bootstrap interval includes zero (Table 11).

At lower noise levels, Brier decreases while ECE increases (Figure 17). For decoder features, ECE increases from 0.00402 to 0.01584 with fixed supervision and from 0.00643 to 0.01648 with EMA. The diffusion timestep changes the distribution of $m^{(\tau)}_{e,t,(i,j)}$ and the positive-target ratio $q_{e,t,(i,j)}(\theta_k) = 1$, so cross-range comparisons depend on both prediction quality and target statistics.

Alignment of confidence-derived weights with local error. We examine how probe predictions translate into weighting signals by applying the same transformation to every variant. Following the risk definition in Section 3.1, the risk at threshold θ_k is $r_{e,t,(i,j)}(\theta_k) = 1 - \hat{q}_{e,t,(i,j)}(\theta_k)$. We average this risk over the nine evaluation thresholds:

$$\bar{r}_{e,t,(i,j)} = \frac{1}{9}\sum_{k=1}^{9} r_{e,t,(i,j)}(\theta_k) = 1 - \frac{1}{9}\sum_{k=1}^{9} \hat{q}_{e,t,(i,j)}(\theta_k).$$

Here, the bar denotes averaging over thresholds. We normalize $\bar{r}$ using each episode's 5th and 95th percentiles and clip the result to $[0, 1]$, denoting the normalized risk by $\widetilde{\bar{r}}_{e,t,(i,j)}$. The diagnostic weights are formed as $1 + \widetilde{\bar{r}}_{e,t,(i,j)}$, normalized to unit mean within each episode, and clipped to $[0.5, 3]$. This transformation uses the patch-level weighting setting $\alpha = 1$ and unit weighting strength, with the nine-threshold average supplying the risk input. We evaluate the resulting weights directly on the common probe grid before task-budget allocation and EVAC retraining.

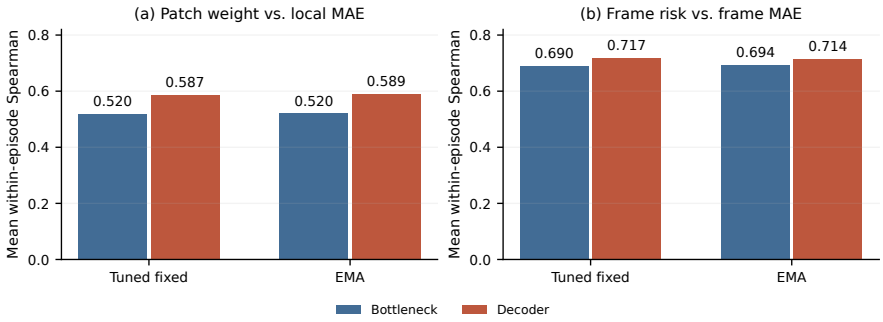


Figure 18: **Alignment of confidence-derived signals with prediction error.** (a) Mean within-episode Spearman correlation between normalized patch weights and local latent MAE. (b) Mean within-episode Spearman correlation between spatially averaged frame risk and frame MAE. Risk is averaged over the nine evaluation thresholds before constructing these diagnostics. All variants use the same weighting transformation, and values are computed before EVAC retraining at seed 42.

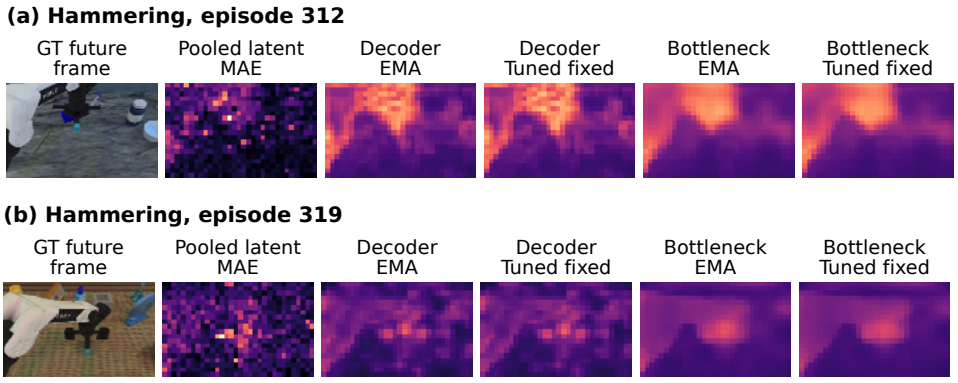


Figure 19: **Qualitative comparison of feature choices and threshold supervision.** Two held-out episodes from the hammering task. Each row shows the first ground-truth future frame, pooled latent MAE, and probe responses for decoder features with EMA and tuned fixed supervision, followed by bottleneck features with EMA and tuned fixed supervision. Decoder responses contain finer spatial structure, while EMA and tuned fixed supervision yield similar patterns within each feature choice.

With fixed supervision, the mean within-episode Spearman correlation between patch weights and local latent error $m^{(\tau)}_{e,t,(i,j)}$ increases from 0.51954 for bottleneck features to 0.58734 for decoder features. With EMA, it increases from 0.52043 to 0.58859. Correlations between spatially averaged frame risk and frame MAE show the same feature trend (Figure 18). The stronger local-error alignment of decoder predictions is therefore retained after the weighting transformation.

Qualitative comparison. Figure 19 shows two episodes from the hammering task, together with pooled latent-error maps and the responses of the four probe configurations. Decoder-based responses exhibit finer spatial variation, while bottleneck-based responses are smoother and cover broader regions. The central response in the second example illustrates this difference. Within each feature choice, EMA and tuned fixed supervision produce similar spatial patterns, consistent with their closely matched quantitative results.

Discussion. Decoder features consistently improve confidence prediction and error alignment at matched probe capacity and resolution. This reflects both finer spatial information and decoder feature fusion. Training-quantile fixed ranges and EMA achieve similar Brier scores, with the calibrated fixed range slightly outperforming EMA for decoder features under training noise. The experiments characterize probe behavior on frozen EVAC-v1 with one seed and one cached clip/noise realization per episode. Evaluations with additional seeds, free-running rollouts, and other architectures would further examine generalization. Crossed retraining experiments could quantify interactions among probe design, acquisition, and weighting in EVAC-v2.

B.4 Direct-Error Selection from the Complete Candidate Pool

Direct-error acquisition. The candidate pool contains complete, pre-collected demonstrations, making future ground-truth frames available for offline selection. We rank all 18,244 episodes by the mean squared difference between EVAC-v1's generated latents and their ground-truth counterparts, averaged over channels, spatial locations, and the first 16 predicted future frames. The four conditioning frames are excluded. This rollout reconstruction score complements the local denoising error used to supervise the confidence probe in Section 3.1. The 7,298 highest-error episodes are selected for unweighted retraining under the 4,000-update protocol in Section 4.1. Confidence uses the task-level allocation described in Appendix B.6.

Table 12: **Direct-error and confidence-based selection.** Both models use optimizer seed 42, 7,298 selected episodes, 4,000 updates, and no loss weighting. Confidence (Selection) uses the seed-42 result in Table 6, while Table 2 reports the average over three confidence runs. Bold values indicate the better result in each column. All metrics are higher-is-better.

Selection	Reconstruction (↑)		Scene (↑)	Semantics (↑)			Motion (↑)		
	PSNR	SSIM	Scene Cons.	Logics	Sem.-CLIP	Sem.-BLEU	Traj-HSD	Traj-Dyn	Traj-nDTW
Oracle error	0.6458	0.7542	0.8988	0.5826	0.8875	**0.2914**	0.0985	0.0587	0.1396
Confidence	**0.6838**	**0.7797**	**0.9322**	**0.6198**	**0.8894**	0.2754	**0.1313**	**0.0921**	**0.1780**

Results. Confidence-guided task allocation exceeds global top-error selection on eight of nine component metrics in Table 12; Oracle attains the higher Sem.-BLEU. The aggregate gains are 0.0318 in Reconstruction, 0.0334 in Scene, and 0.1045 in Motion. Figure 20 presents paired differences and their uncertainty at both aggregate and component levels. This comparison evaluates two acquisition policies: global ranking by current rollout MSE and confidence-guided task quotas with within-task random sampling. Their differences jointly reflect the scoring signal and the distribution of selected episodes across tasks.

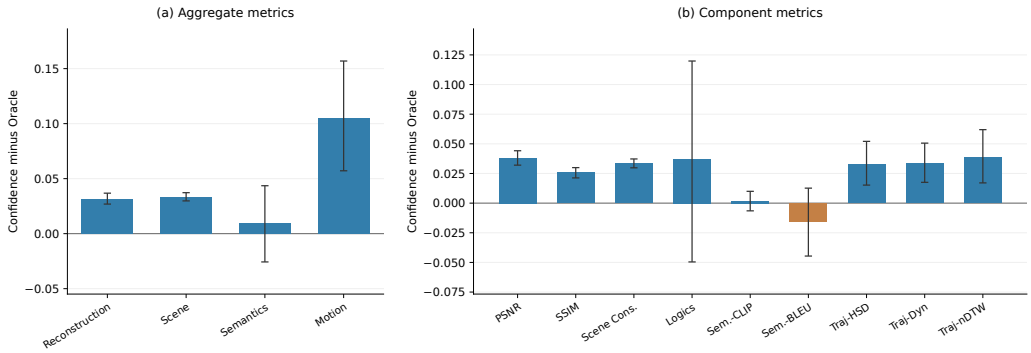


Figure 20: **Confidence versus direct-error selection.** Bars show Confidence minus Oracle for four aggregate metrics (left) and nine component metrics (right), using the unweighted models in Table 12. Positive values favor Confidence. Error bars denote paired 95% episode-bootstrap intervals. Most comparisons use 242 common episodes; Sem.-CLIP, Sem.-BLEU, and aggregate Semantics use 190 common episodes.

B.5 Data-Budget Sensitivity under Fixed Short Optimization

Protocol. We compare unweighted confidence and uniform-task random selection at 10%, 20%, and 40% of the candidate pool, together with a 100% full-pool reference. All models receive 750 updates with effective batch size 16 and optimizer seed 42. Selection also uses seed 42. The two selection methods share the quota allocator and within-task sampling procedure; random selection assigns equal weights to tasks. Evaluation uses a fixed 64-episode subset spanning the 32 tasks represented in the original validation split. This sweep measures data-budget sensitivity under the same short optimization schedule.

Table 13: **Budget sensitivity under 750-update post-training.** All models use optimizer seed 42 and no loss weighting. Bold values mark the higher score within each confidence/random pair. Aggregates follow the main-text definitions and use each component's available mean. Reconstruction, Scene, and Motion cover 62 episodes; CLIP/BLEU cover 56 episodes.

Budget	Selection	Episodes	Reconstruction ↑	Scene ↑	Semantics ↑	Motion ↑	Train GPU-h
10%	Random	1,824	0.6341	0.8772	0.5007	0.1701	4.47
10%	Confidence	1,824	**0.6995**	**0.9088**	**0.5900**	**0.2502**	4.45
20%	Random	3,649	**0.6679**	**0.9003**	0.5391	**0.2360**	4.45
20%	Confidence	3,649	0.6642	0.8812	**0.6167**	0.1970	4.44
40%	Random	7,298	**0.6874**	0.8786	0.5576	0.2004	4.44
40%	Confidence	7,298	0.6611	**0.8981**	**0.5987**	**0.2305**	4.47
100%	Full pool	18,244	0.6902	0.9030	0.5263	0.2023	4.46

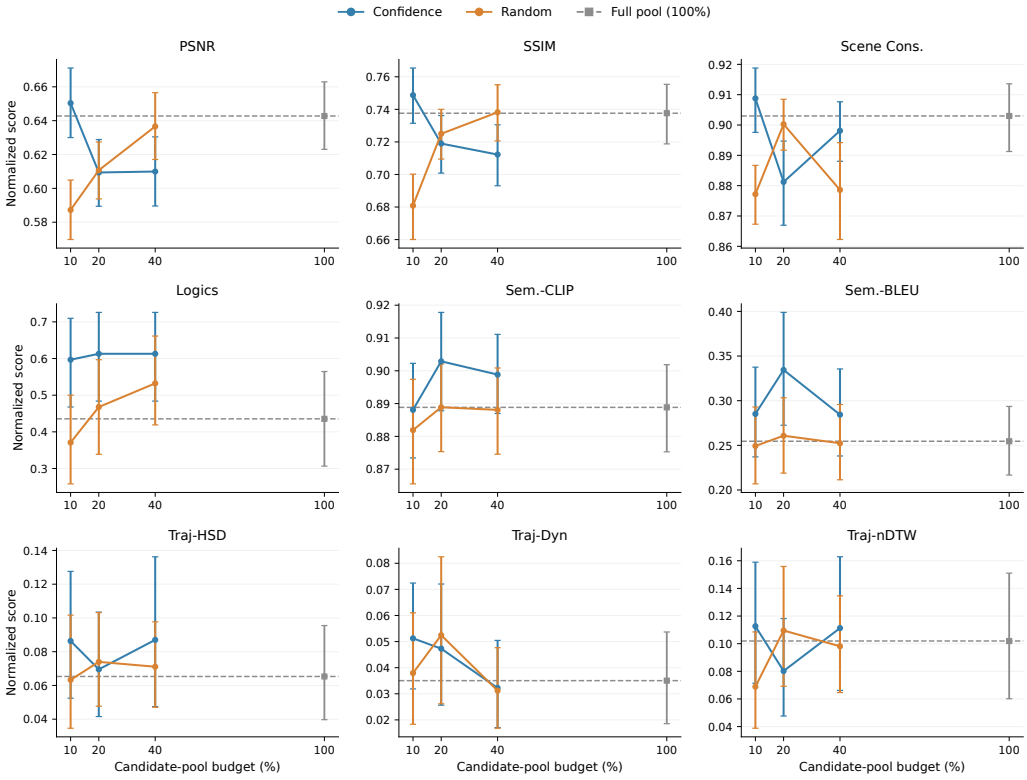


Figure 21: **Budget–performance curves under fixed optimization.** All metrics are higher-is-better. Confidence and random receive the same 750 updates. In each panel, the gray square marks the 100% full-pool result, and the gray horizontal dashed line extends its mean across budgets for reference. Error bars show 95% episode-bootstrap intervals, using 62 common episodes for most metrics and 56 for CLIP/BLEU. Lines between measured budget points serve as visual guides.

Results and effect of data budget. Under the fixed 750-update schedule, performance depends on both the acquisition strategy and the available data budget. At 10%, confidence improves all four aggregate scores over random, with positive paired intervals in Table 14, and its point estimates also exceed the full-pool reference. At larger budgets, the gains become metric-dependent: confidence improves Semantics at 20%, and Scene, Semantics, and Motion at 40%, while random achieves higher Reconstruction at both budgets. Each run processes 12,000 sample presentations, corresponding to approximately 6.58 passes over the 10% subset but only 0.66 passes over the full pool of 18,244 eligible episodes. Thus, increasing the pool broadens data coverage while reducing

Table 14: **Paired confidence-minus-random differences.** Each entry gives the paired mean and its 95% interval from 10,000 episode-bootstrap resamples. Larger differences favor confidence. Reconstruction, Scene, and Motion use 62 common episodes; Semantics uses the 56-episode intersection of its three components. This common-episode calculation can differ slightly from subtracting the component-wise aggregates in Table 13.

Budget	ΔReconstruction ↑	ΔScene ↑	ΔSemantics ↑	ΔMotion ↑
10%	$0.0655\ [0.0534, 0.0772]$	$0.0316\ [0.0229, 0.0396]$	$0.0914\ [0.0247, 0.1563]$	$0.0801\ [0.0293, 0.1391]$
20%	$-0.0037\ [-0.0186, 0.0112]$	$-0.0190\ [-0.0296, -0.0089]$	$0.0768\ [0.0119, 0.1425]$	$-0.0389\ [-0.1023, 0.0185]$
40%	$-0.0263\ [-0.0411, -0.0120]$	$0.0195\ [0.0031, 0.0362]$	$0.0440\ [-0.0164, 0.1021]$	$0.0301\ [-0.0570, 0.1390]$

repeated exposure under the same optimization budget, providing one explanation for the observed non-monotonic trends. Overall, the clearest advantage of confidence-guided selection appears at the smallest tested data budget, and the full-pool result serves as a reference under the same fixed-update setting.

Cost accounting. All short-budget models require approximately 2.22–2.23 wall-clock hours on two A800 GPUs (4.44–4.47 GPU-hours). Selecting fewer eligible episodes therefore does not reduce training compute under this fixed-update protocol. The 4,000-update Oracle run takes 11.67 hours on two A800 GPUs (23.35 GPU-hours). Given generated validation videos, EWMBench scoring takes 2.09 hours on one A800 for the 250-episode manifest, and 0.51–0.57 hours for the 64-episode subset. Generating these videos takes a further 4.66 and 1.16–1.21 hours, respectively, on two A800 GPUs. These timings separate post-training, rollout generation, and metric evaluation; acquisition and probe-training overhead require additional accounting in an end-to-end comparison.

B.6 Selection, Weighting, and Evaluation Protocol

Task scores and sampling quotas. Each task is represented by one randomly chosen episode and scored using S_{mean} from Section 3.2. Let K denote the number of distinct task scores. We rank these distinct scores in ascending order as $k = 0, \ldots, K-1$, with tasks sharing the same score assigned the same k. The allocation multiplier for a task with rank k is

$$w_{\text{task}}(k) = 0.25 + 1.75 \frac{k}{\max(K-1, 1)}. \tag{32}$$

Thus, the multipliers range linearly from 0.25 to 2.0 when $K > 1$, while $w_{\text{task}} = 0.25$ for all tasks when $K = 1$. Each nonempty task first receives one episode. The remaining budget is allocated proportionally to w_{task} using floored shares, with residual units assigned by largest fractional remainder. Tasks are capped by their available episode counts, and any unused quota is redistributed among the remaining tasks. A single Python random-number generator with seed 42 shuffles each task's episode list in lexicographic task order. Setting all task scores equal recovers the uniform-task random baseline.

Features and confidence weighting. We use the decoder features $h_{\text{dec}}^{(\ell)}$ with $\ell = 5$, corresponding to zero-based `output_blocks[5]` and a $1280 \times 20 \times 32$ feature tensor. The frame-and-patch setting uses $\alpha = 0.5$ in $r_{t,(i,j)}^{(\alpha)}$. After interpolation to the training-loss grid, the 5th and 95th percentiles are computed separately for each video's complete time–space risk tensor. Subtracting the lower percentile, dividing by the percentile range plus 10^{-6}, and clipping to $[0, 1]$ gives $\widetilde{r}_{t,(i,j)}^{(\alpha)}$. We use the main-text schedule $\lambda_{\text{eff}}(s) = \lambda_{\text{conf}} \min(1, s/s_{\text{warm}})$ with $\lambda_{\text{conf}} = 1$ and $s_{\text{warm}} = 1000$. The resulting $w_{t,(i,j)}$ is normalized to unit mean per video and clipped to $[0.5, 3]$. Confidence maps remain detached during world-model training.

Reward and progress baselines. Each baseline scores generated and ground-truth videos. RoboReward parses progress $s_{\text{raw}} \in \{1, \ldots, 5\}$; we read its normalized `score`, $s = (s_{\text{raw}} - 1)/4$, with $s = 0$ for parse failures. GVL reads `task_completion_percentage` on $[0, 100]$. All 1,400 cached values are zero or above 1, so the saved scores equal $s_{\text{raw}}/100$; replaying this percentage conversion leaves scores and selection risks unchanged. Both Robometer variants read `progress_pred` in $[0, 1]$. Prog and Pref denote the `progress` and `preference`

Table 15: **Selection and retraining settings.** The weighting parameters apply to the weighted experiments in the main text. The new Oracle and budget-sweep models use unweighted losses.

Setting	Value
Task representatives / acquisition score	1 per task / S_{mean}
Rank-based allocation multiplier	0.25–2.0
Decoder index ℓ	5 (zero-based)
Frame-and-patch mixing α	0.5
Confidence strength λ_{conf}	1.0
Weighting warmup s_{warm}	1,000 updates
Risk normalization percentiles	5th and 95th, per video
Weight clipping after mean normalization	$[0.5, 3.0]$
Probe / EVAC-v1 training	6,000 / 2,000 updates
Selected-data initialization	Original pretrained EVAC
Effective training batch size	16 (8 per GPU, two GPUs)

adapter modes, respectively; both reported variants use the progress output head and the common GT-relative risk calculation below. LRMs read per-frame `score` from completion progress in $[0, 1]$. PRM-as-Judge parses signed percentages $p \in [-100, 100]$ and uses backward progress $s = 1 + p/100$. We clip these converted scores to $[0, 1]$, linearly resample both sequences to the longer length, and broadcast scalars. The aligned scores $s^{(b)}_{\text{GT},t}$ and $s^{(b)}_{\text{pred},t}$ define risk

$$r^{(b)}_t = 1 - \frac{s^{(b)}_{\text{pred},t} - s^{(b)}_{\text{GT},t} + 1}{2} = \frac{1 + s^{(b)}_{\text{GT},t} - s^{(b)}_{\text{pred},t}}{2}. \tag{33}$$

Larger ground-truth-relative deficits therefore receive higher risk. We average these frame risks and apply the same task-quota allocator used for confidence.

Evaluation and statistical summaries. PSNR is divided by 30 and CLIPScore by 100; normalized components are clipped to $[0, 1]$. A shared GT-trajectory filter retains 242/250 full-set episodes and 62/64 subset episodes for all methods. The reported semantic results use 190 and 56 CLIP/BLEU records, respectively, because shortened episode IDs in these evaluation outputs collide. Each metric is averaged over its available records; paired comparisons use records shared by the relevant methods and components. For the new direct-error and budget-sweep comparisons, 10,000 episode-bootstrap resamples with seed 42 quantify evaluation-set variability at fixed checkpoints. Main comparisons pool paired episode differences across three runs (Appendix B.2); probe ablations use 3,000 task-bootstrap resamples (Appendix B.3).

Full task/episode/trial IDs were used to verify every retained CLIP/BLEU record against its corresponding GT caption. All 190/56 retained prediction–GT pairs are correctly matched, and all methods use the same fixed subsets. Overwritten records are excluded consistently, with no unresolved ID ambiguity among the retained records.

Random seeds. All single-seed experiments use seed 42, including the Confidence reference in Table 12, the Oracle experiment, and the seven short-budget models. The main learned-method comparison additionally reports averages over three runs with seeds 42, 3407, and 123, following the setting in Section 4.1.

B.7 Qualitative Evolution from Base EVAC to EVAC-v2

We further visualize how world-model predictions evolve throughout the post-training pipeline. We select six representative RoboTwin2.0 episodes covering object placement, cabinet interaction, block stacking, bowl stacking, and switch manipulation. These examples expose a consistent progression from severe zero-shot cross-embodiment failure to target-domain adaptation and, finally, confidence-guided refinement.

Per-episode progression. Table 16 summarizes the four aggregated evaluation dimensions for the six selected episodes. All relative changes are computed with respect to EVAC-v1, which serves as the common post-warmup baseline. Across the selected examples, Base EVAC performs substantially worse under zero-shot cross-embodiment transfer, while EVAC-v1 already recovers much of the target-domain prediction quality. Confidence-guided EVAC-v2 further improves most dimensions, with frame-and-patch weighting achieving the strongest Reconstruction in all six episodes.

For this per-episode analysis, Semantics is aggregated from normalized Sem.-CLIP and Sem.-BLEU only; the binary Logics score is omitted because a single $0/1$ decision would disproportionately affect an individual episode.

Table 16: **Aggregated evolution on six representative RoboTwin2.0 episodes.** Each entry reports the normalized aggregate followed by its relative change from EVAC-v1. Green and red percentages indicate improvement and degradation, respectively. For Motion, relative changes are omitted when the EVAC-v1 score is zero. Green, yellow, and red cells indicate the best, second-best, and third-best values within each episode.

Episode	Variant	Reconstruction (↑)	Scene (↑)	Semantics (↑)	Motion (↑)
place burger fries (ep130)	Base EVAC	0.4592 (-27.3%)	0.7890 (-10.0%)	0.3591 (-28.9%)	0.0000 (-100.0%)
	EVAC-v1	0.6318 (0.0%)	0.8770 (0.0%)	0.5050 (0.0%)	0.4570 (0.0%)
	EVAC-v2 Frame	0.6618 (+4.7%)	**0.9430** (+7.5%)	0.5221 (+3.4%)	1.0830 (+137.0%)
	EVAC-v2 Fr.+Patch	**0.7020** (+11.1%)	0.9370 (+6.8%)	**0.5486** (+8.6%)	**1.3080** (+186.2%)
place empty cup (ep294)	Base EVAC	0.4605 (-38.6%)	0.7120 (-27.0%)	0.5454 (+2.0%)	0.0000 (-100.0%)
	EVAC-v1	0.7503 (0.0%)	**0.9750** (0.0%)	0.5349 (0.0%)	0.5600 (0.0%)
	EVAC-v2 Frame	0.7353 (-2.0%)	0.9700 (-0.5%)	0.2604 (-51.3%)	0.4150 (-25.9%)
	EVAC-v2 Fr.+Patch	**0.7558** (+0.7%)	0.9610 (-1.4%)	**0.6015** (+12.5%)	**0.5900** (+5.4%)
put object cabinet (ep282)	Base EVAC	0.4372 (-23.2%)	0.7770 (-10.4%)	0.4856 (-0.5%)	0.0000 (–)
	EVAC-v1	0.5693 (0.0%)	0.8670 (0.0%)	0.4879 (0.0%)	0.0000 (0.0%)
	EVAC-v2 Frame	0.6178 (+8.5%)	**0.9260** (+6.8%)	0.5264 (+7.9%)	0.0000 (–)
	EVAC-v2 Fr.+Patch	**0.6207** (+9.0%)	0.8950 (+3.2%)	**0.5651** (+15.8%)	0.0000 (–)
stack blocks two (ep325)	Base EVAC	0.3793 (-38.4%)	0.7970 (-10.4%)	0.4391 (-4.6%)	0.0000 (-100.0%)
	EVAC-v1	0.6160 (0.0%)	0.8900 (0.0%)	0.4603 (0.0%)	1.1470 (0.0%)
	EVAC-v2 Frame	0.6937 (+12.6%)	0.9220 (+3.6%)	0.4606 (+0.1%)	1.1770 (+2.6%)
	EVAC-v2 Fr.+Patch	**0.7053** (+14.5%)	**0.9240** (+3.8%)	**0.4622** (+0.4%)	**1.2470** (+8.7%)
stack bowls three (ep087)	Base EVAC	0.5532 (-15.5%)	0.8000 (+12.5%)	0.4340 (+4.8%)	0.0000 (-100.0%)
	EVAC-v1	0.6545 (0.0%)	0.7110 (0.0%)	0.4143 (0.0%)	0.3390 (0.0%)
	EVAC-v2 Frame	0.7407 (+13.2%)	**0.9060** (+27.4%)	0.4403 (+6.3%)	0.0000 (-100.0%)
	EVAC-v2 Fr.+Patch	**0.7555** (+15.4%)	0.8880 (+24.9%)	**0.4687** (+13.1%)	**0.4550** (+34.2%)
turn switch (ep310)	Base EVAC	0.5578 (-19.4%)	0.8260 (-11.2%)	0.4431 (-31.5%)	0.0000 (–)
	EVAC-v1	0.6923 (0.0%)	0.9300 (0.0%)	0.6464 (0.0%)	0.0000 (0.0%)
	EVAC-v2 Frame	0.7028 (+1.5%)	**0.9560** (+2.8%)	0.6869 (+6.3%)	0.0000 (–)
	EVAC-v2 Fr.+Patch	**0.7350** (+6.2%)	0.9410 (+1.2%)	**0.8269** (+27.9%)	0.0000 (–)

Representative visual evolution. Figures 22–27 provide matched temporal comparisons for the six representative episodes. Base EVAC is strongly out-of-distribution under the RoboTwin2.0 embodiment and often fails to preserve a recognizable robot arm or coherent interaction trajectory. After warmup, EVAC-v1 recovers the robot embodiment and basic scene structure, but noticeable color drift, object corruption, temporal flickering, and interaction errors remain. EVAC-v2 with frame weighting substantially improves the overall rollout, although local object geometry and gripper–object interactions can still fail. Frame-and-patch weighting further concentrates learning on these difficult local regions and most clearly improves manipulated objects, gripper geometry, and fine interaction details in the selected cases.



Figure 22: **Training evolution on `place_burger_fries` (ep130).** EVAC-v1 develops a strong blue color shift and the tray contents collapse from the middle of the rollout. EVAC-v2 Frame removes most tray corruption, but the left arm hallucinates an object instead of grasping the burger and the object held by the right arm is visibly deformed. EVAC-v2 Fr.+Patch correctly grasps and places the burger while preserving the fries most faithfully.

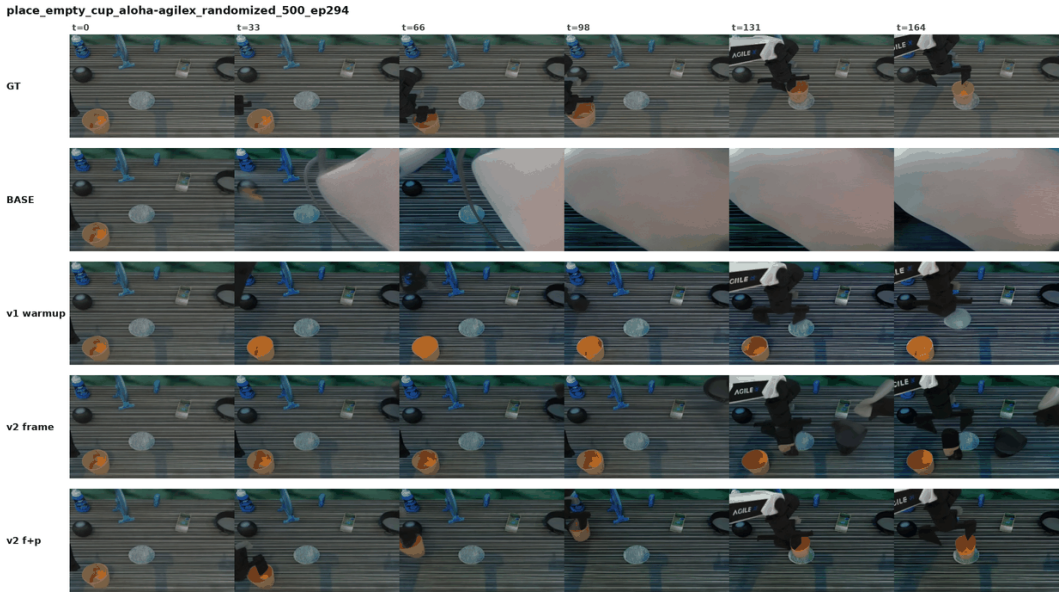


Figure 23: **Training evolution on `place_empty_cup` (ep294).** In both EVAC-v1 and EVAC-v2 Frame, the arm only partially appears near the image boundary and fails to grasp the cup at the beginning; EVAC-v2 Frame additionally deteriorates toward the end of the rollout. EVAC-v2 Fr.+Patch correctly grasps the cup from the boundary, places it on the plate, and maintains a coherent scene throughout the prediction.



Figure 24: **Training evolution on `put_object_cabinet` (ep282).** EVAC-v1 shows moderate color bias and eventually produces a floating arm together with a corrupted cabinet. EVAC-v2 Frame does not resolve these failures: the blue box in the upper-left region disappears and the arm is generated at an incorrect location. EVAC-v2 Fr.+Patch preserves the objects, cabinet structure, and robot geometry most consistently.

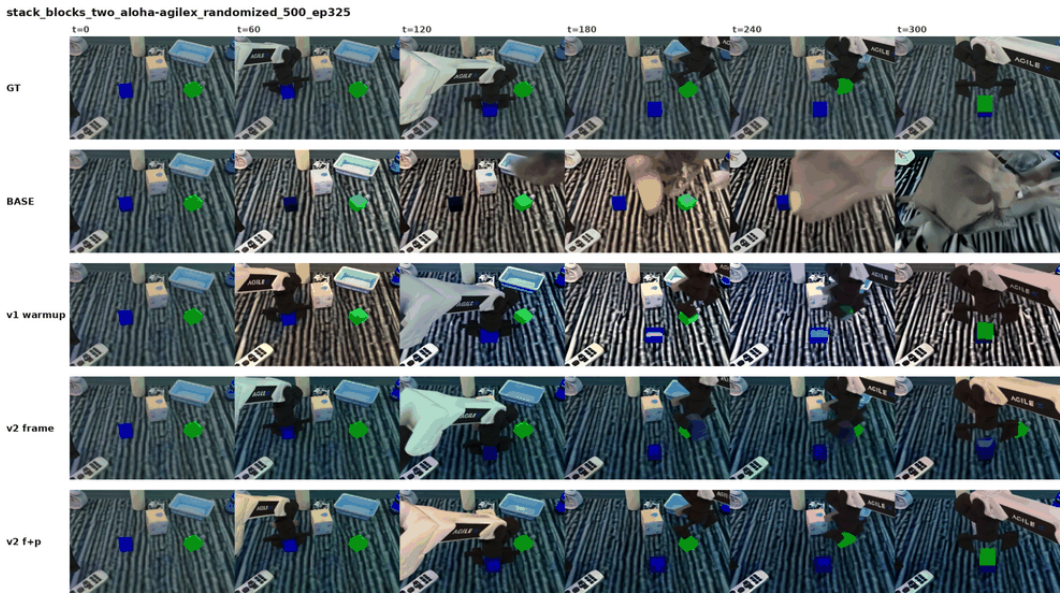


Figure 25: **Training evolution on `stack_blocks_two` (ep325).** EVAC-v1 exhibits strong color flickering, and the lower blue block disappears when the green block is lifted and stacked. EVAC-v2 Frame largely removes the color shift but fails to keep the green block attached to the gripper and consequently misses the stacking interaction. EVAC-v2 Fr.+Patch resolves both the object-persistence and grasping failures.

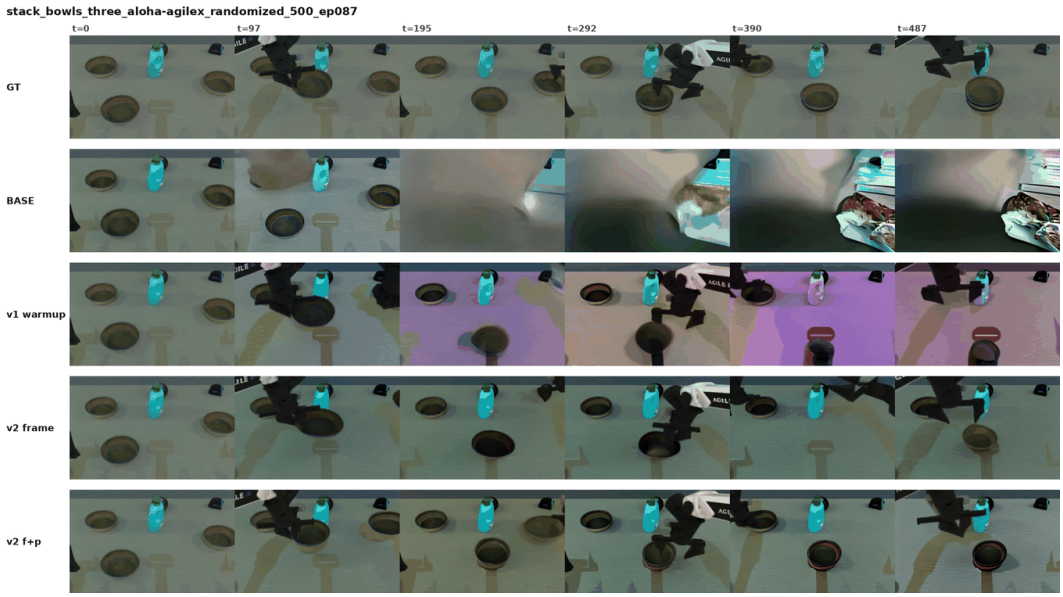


Figure 26: **Training evolution on `stack_bowls_three` (ep087).** EVAC-v1 suffers from pronounced color drift and repeated red flickering, followed by severe object corruption late in the rollout. EVAC-v2 Frame removes the color shift but still produces substantial bowl deformation, disappearance, and reappearance. EVAC-v2 Fr.+Patch better preserves the bowl geometry and maintains object persistence through the later frames.

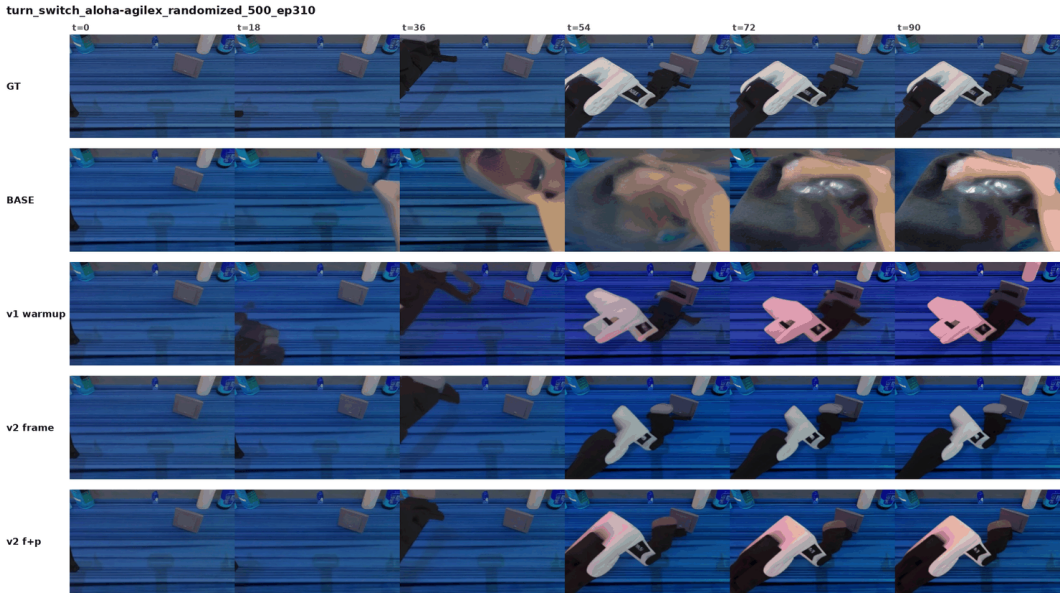


Figure 27: **Training evolution on `turn_switch` (ep310).** EVAC-v1 produces a reddish robot arm, a bluish background, and implausible floating-arm geometry. EVAC-v2 Frame substantially reduces the color bias, but the arm shape remains distorted. EVAC-v2 Fr.+Patch further improves the local arm geometry and produces an embodiment substantially closer to the ground truth.

Detailed EWMBench measurements. Table 17 reports the normalized component metrics underlying these representative examples. Overall, the transition from Base EVAC to EVAC-v1 substantially improves target-domain prediction quality, while confidence-guided EVAC-v2 provides further gains across reconstruction, scene, semantic, and trajectory dimensions. Among the selected episodes, frame-and-patch weighting gives the strongest overall progression, particularly for semantic quality and Motion when valid trajectory signals are available. These quantitative trends are consistent with the qualitative improvements shown above.

Table 17: **Detailed normalized EWMBench results for the qualitative evolution study.** All metrics are higher-is-better. Green, yellow, and red indicate the best, second-best, and third-best distinct values within each episode, respectively.

Episode	Variant	Reconstruction (↑)		Scene (↑)	Semantics (↑)		Motion (↑)		
		PSNR (↑)	SSIM (↑)	Scene Cons. (↑)	Sem.-CLIP (↑)	Sem.-BLEU (↑)	Traj-HSD (↑)	Traj-Dyn (↑)	Traj-nDTW (↑)
place burger fries (ep130)	Base EVAC	0.4413	0.477	0.789	0.7183	0.000	0.000	0.000	0.000
	EVAC-v1	0.5957	0.668	0.877	0.7879	0.222	0.095	0.001	0.361
	EVAC-v2 Frame	0.6777	0.646	**0.943**	0.8121	0.232	0.545	**0.024**	0.514
	EVAC-v2 Fr.+Patch	**0.6980**	**0.706**	0.937	**0.8182**	**0.279**	**0.761**	0.002	**0.545**
place empty cup (ep294)	Base EVAC	0.5100	0.411	0.712	0.8448	0.246	0.000	0.000	0.000
	EVAC-v1	**0.7217**	0.779	**0.975**	0.8038	0.266	0.108	0.365	0.087
	EVAC-v2 Frame	0.7117	0.759	0.970	0.3568	0.164	0.097	0.238	0.080
	EVAC-v2 Fr.+Patch	0.7197	**0.792**	0.961	**0.8760**	**0.327**	**0.117**	**0.379**	**0.094**
put object cabinet (ep282)	Base EVAC	0.4593	0.415	0.777	0.8081	0.163	0.000	0.000	0.000
	EVAC-v1	0.5377	0.601	0.867	0.7927	0.183	0.000	0.000	0.000
	EVAC-v2 Frame	0.5727	**0.663**	**0.926**	0.8708	0.182	0.000	0.000	0.000
	EVAC-v2 Fr.+Patch	**0.5903**	0.651	0.895	**0.9262**	**0.204**	0.000	0.000	0.000
stack blocks two (ep325)	Base EVAC	0.4537	0.305	0.797	0.7823	0.096	0.000	0.000	0.000
	EVAC-v1	0.5650	0.667	0.890	**0.8246**	0.096	0.303	0.205	0.639
	EVAC-v2 Frame	0.6313	0.756	0.922	0.8052	**0.116**	0.304	0.218	0.655
	EVAC-v2 Fr.+Patch	**0.6447**	**0.766**	**0.924**	0.8244	0.100	**0.319**	**0.259**	**0.669**
stack bowls three (ep087)	Base EVAC	0.4953	0.611	0.800	0.7860	0.082	0.000	0.000	0.000
	EVAC-v1	0.5630	0.746	0.711	**0.8287**	0.000	**0.063**	0.161	0.115
	EVAC-v2 Frame	0.6803	**0.801**	**0.906**	0.8116	0.069	0.000	0.000	0.000
	EVAC-v2 Fr.+Patch	**0.7160**	0.795	0.888	0.8204	**0.117**	0.062	**0.225**	**0.168**
turn switch (ep310)	Base EVAC	0.5507	0.565	0.826	0.7632	0.123	0.000	0.000	0.000
	EVAC-v1	0.6587	0.726	0.930	0.9108	0.382	0.000	0.000	0.000
	EVAC-v2 Frame	0.6807	0.725	**0.956**	0.9308	0.443	0.000	0.000	0.000
	EVAC-v2 Fr.+Patch	**0.7170**	**0.753**	0.941	**0.9318**	**0.722**	0.000	0.000	0.000

B.8 More Results of Qualitative Confidence Visualization

We provide additional qualitative examples to complement the confidence analysis in the main text. Figures 28 and 29 show the two episodes with the highest patch-level Spearman correlations, reaching 0.713 and 0.695, respectively. Across different future frames, the predicted risk maps consistently concentrate around manipulators, operated objects, and interaction regions, and largely agree with both pixel- and latent-space oracle errors. The corresponding risk overlays further illustrate that confidence provides spatially localized signals rather than merely reflecting global video quality.

Figures 30 and 31 show two challenging cases with lower patch-level Spearman correlations of 0.241 and 0.265. In these examples, oracle errors extend beyond the regions highlighted by risk, and predicted maps can remain weak despite substantial local errors, revealing missed failures rather than only imprecise spatial boundaries. Together, the best and worst cases illustrate both the localization capability and the remaining limitations of the confidence probe, consistent with our quantitative finding that confidence provides useful rankings but does not reliably recover all high-error regions, particularly when the predicted risk map remains weak despite substantial oracle prediction errors.

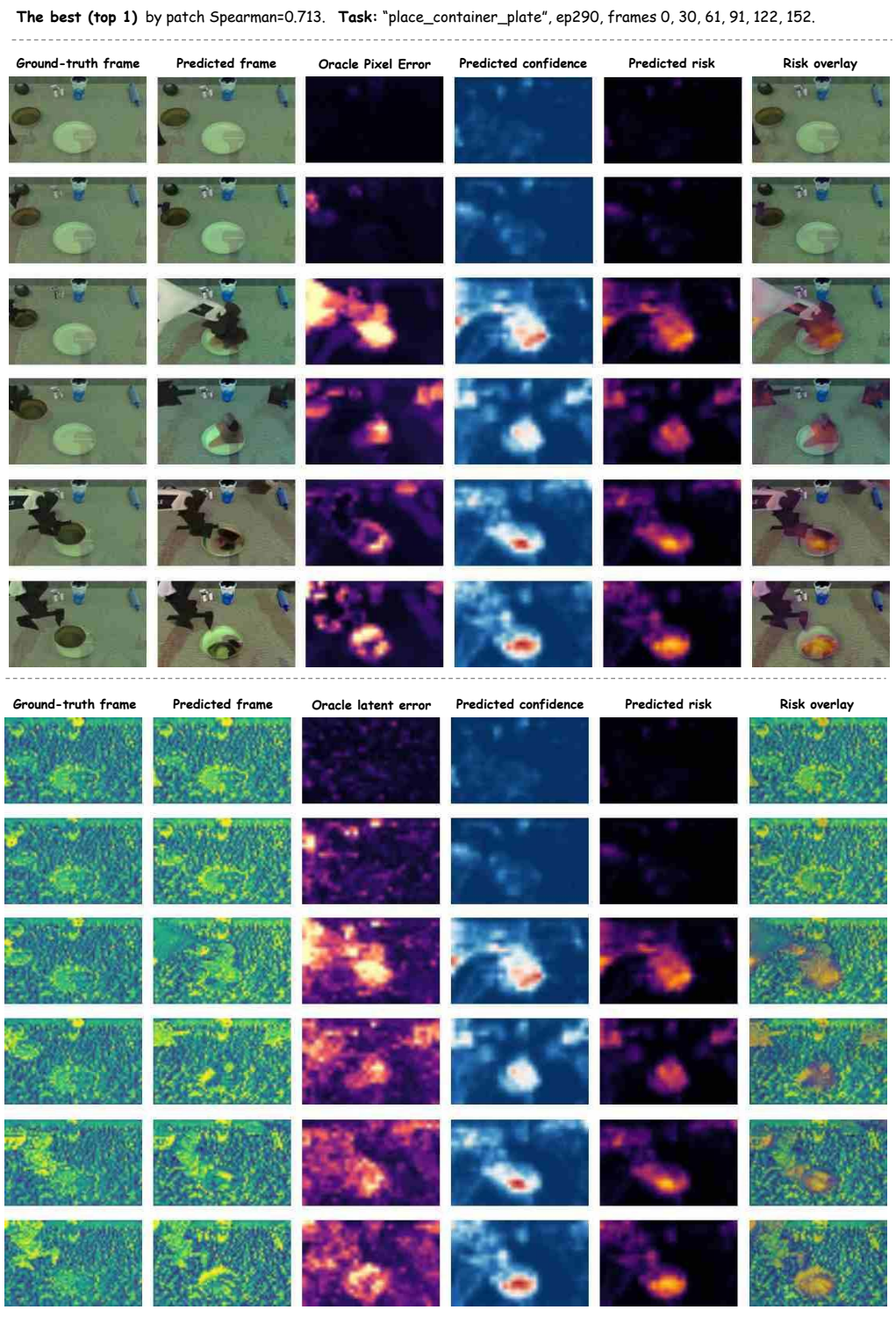


Figure 28: **More results of qualitative confidence visualization.** Best-ranked example (top 1) by patch-level Spearman correlation.

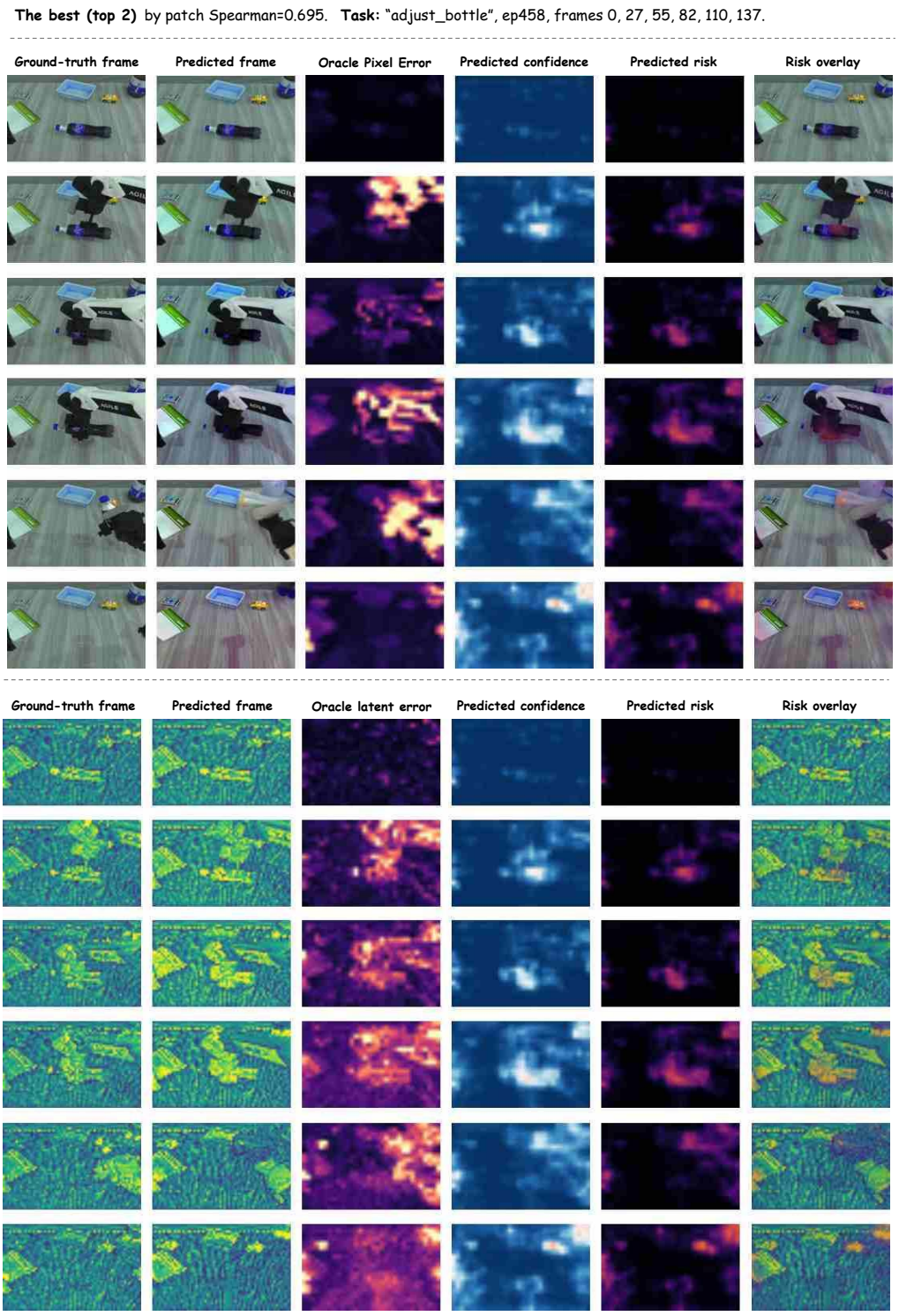


Figure 29: **More results of qualitative confidence visualization.** Second-highest-ranked example by patch-level Spearman correlation.

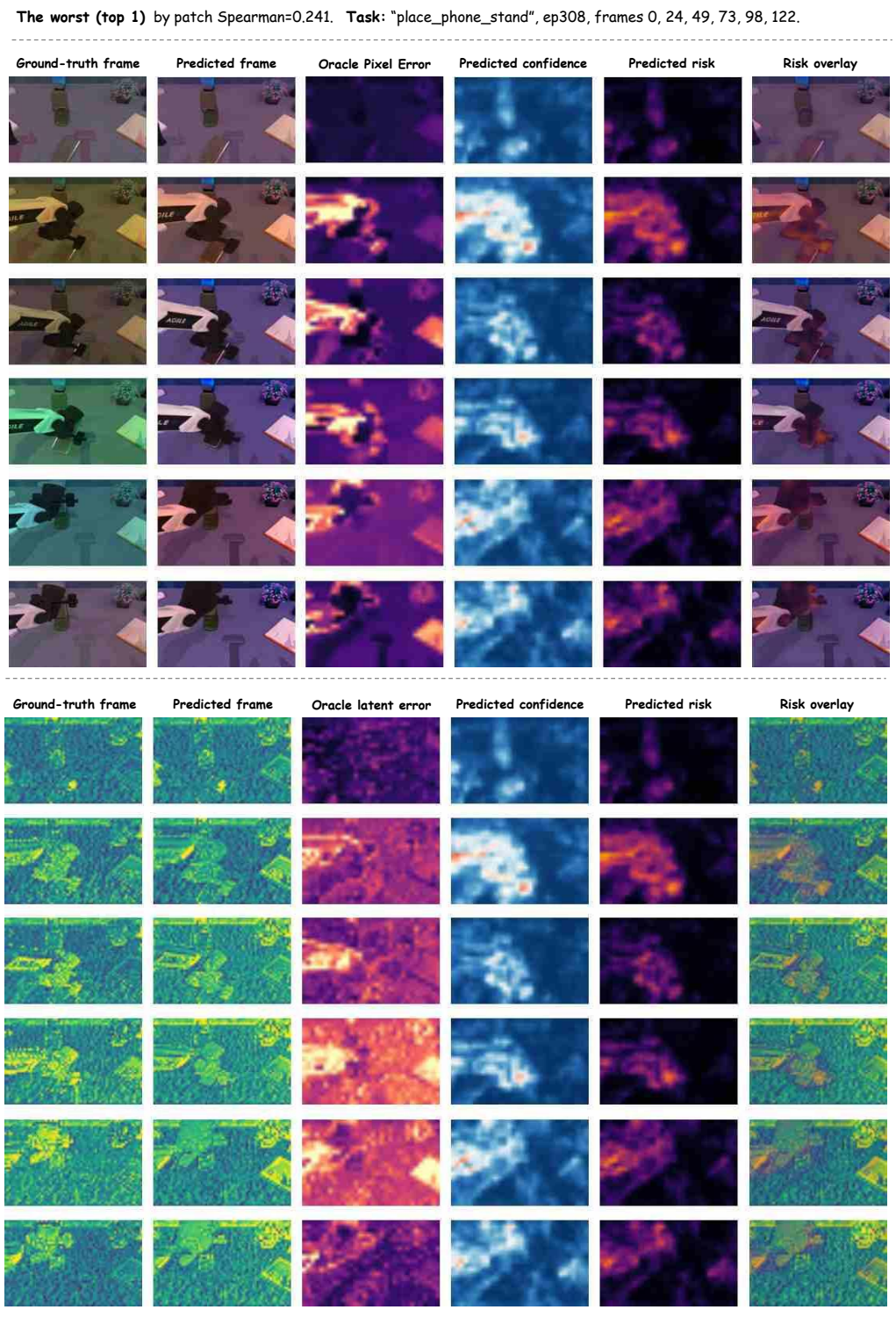


Figure 30: **More results of qualitative confidence visualization.** Lowest-ranked example (top 1) by patch-level Spearman correlation.

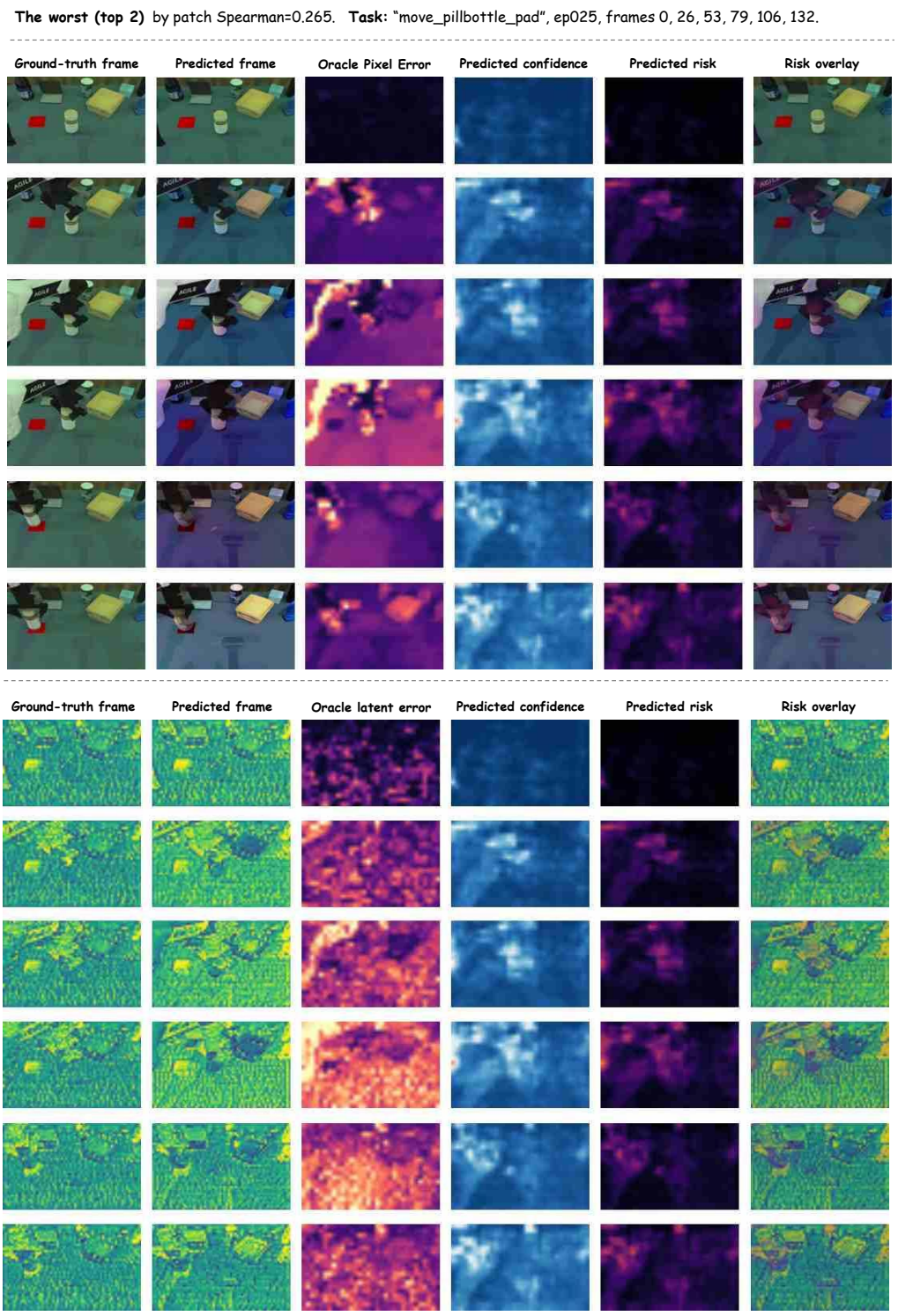


Figure 31: **More results of qualitative confidence visualization.** Second-lowest-ranked example by patch-level Spearman correlation.