RILS LAB Seminar 1

RILS LAB

Robot Intelligence Learning & System

임찬우

2026. 07. 16

01

Novelty Detection in Reinforcement Learning with World Models

Geigh Zollicoffer, Kenneth Eaton, Jonathan Balloch, Julia Kim, Wei Zhou, Robert Wright, Mark Riedl

ICML / 2025 (spotlight)

Contents

- **Preliminaries**
- **Problem**
- **Method**

Preliminaries

- **What is Novelty?**
  - A sudden, unanticipated change to the environment's dynamics or observations, occurring at inference time (not seen during training)
  - Permanent — distinguishing it from a one-off anomaly (e.g., sensor glitch)
  - e.g., cars suddenly become invisible · lava appears · physics changes

- **Novelty ≠ Anomaly**
  - **Anomaly**: one-time, localized error (passes by) — *a single noisy frame*
  - **Novelty**: permanent shift in how the world works (here to stay) — *the rules changed*

- **What is world model?**
  - A model that predicts the next state given the current state and action — learned from the agent's own experience
  - Once learned, the agent can imagine rollouts internally, without interaction with the real environment

- **Why it matters?**
  - When novelty occurs, the agent's trained policy becomes unreliable → risk of catastrophic mistakes (robotics, finance, high-stakes systems)

Problem

- **Traditional ML Methods** — *Break the i.i.d. assumption*
  - RL data is sequential & context-dependent; treating it as independent misses crucial dependencies.
  - **Solution**: The world model's memory (h_t) encodes sequential context — detection is conditioned on the current context, not on i.i.d. samples.

- **Generalization** (augmentation, procedural gen.) — *Costly & fundamentally limited*
  - High sample complexity, and modeling *every* unseen situation is impossible — contradicts "novelty is unanticipatable."
  - **Solution**: No augmented data, no pre-modeling of novelties — reuse the already-trained world model's own distributions and measure the mismatch in real time.

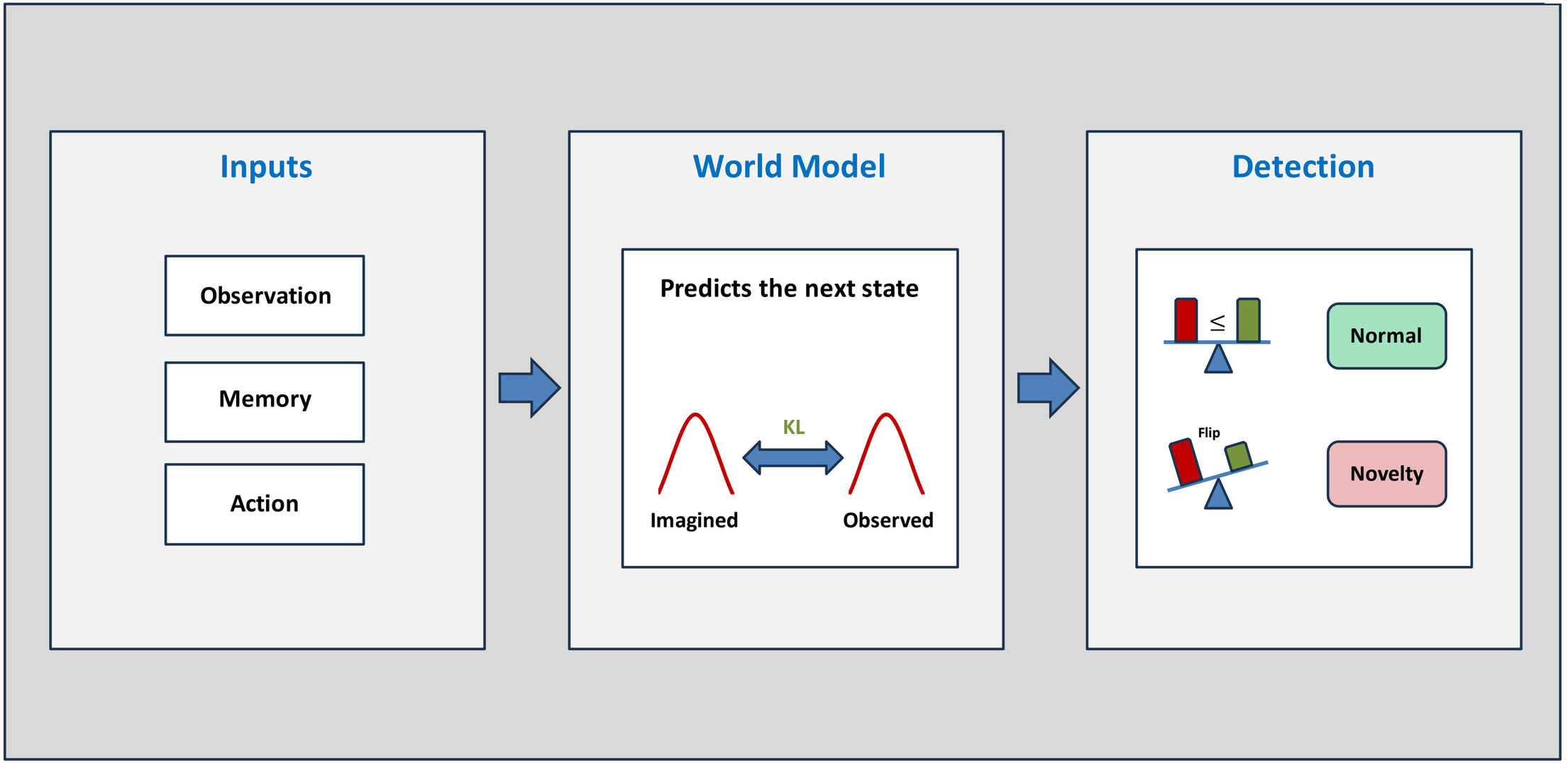
- **Reward-Deterioration** — *Too late & misleading*
  - By the time reward drops, recovery may be impossible; the cause may not even be the novelty.
  - **Solution**: Score every transition by the latent divergence between imagined and observed states — flags at the observation itself, with no waiting for reward signals.

Method

- **The common flaw** — *All prior methods assume novelty is known in advance*
  - They require prior knowledge of the anomaly — but novelty is unanticipatable → premise violated
- **Our insight** — *Use the World Model's own learned inequality*
  - Normal → inequality holds · Novelty → inequality flips → no manual threshold needed
- **Handling time-dependency** — *Memory captures context*
  - Memory h_t reflects sequential context → unlike i.i.d.-based traditional ML
- **No pre-modeling needed** — *Measures real-time mismatch*
  - Detects the live gap between imagined & observed, not pre-modeled futures
- **Fast & early detection** — *Instant distribution gap*
  - Flags via immediate distribution difference, not delayed reward signals; ~10^2× (Atari) / ~10^3× (DMC) faster than RIQN

$$KL[p_\phi(z_t|h_t,x_t)||p_\phi(z_t|h_t)] \leq KL[p_\phi(z_t|h_t,x_t)||p_\phi(z_t|h_0)] - KL[p_\phi(z_t|h_t,x_t)||p_\phi(z_t|h_0,x_t)]$$

Method


Inputs
Observation
Memory
Action
World Model
Predicts the next state
KL
Imagined
Observed
Detection
≤
Normal
Flip
Novelty

02

GameFactory: Creating New Games with Generative Interactive Videos

Jiwen Yu, Yiran Qin, Xintao Wang, Pengfei Wan, Di Zhang, Xihui Liu

ICCV / 2025 (highlight)

Preliminaries

- **Video Diffusion Model & Latent Space** — *Generate by denoising, in a compressed space*
  - Restores a clean video from noise via step-by-step denoising
  - Operates not on raw pixels but on a VAE-compressed latent → temporal compression $r = 4$ (4 frames → 1 latent)

- **LoRA (Low-Rank Adaptation)** — *A detachable "filter" for the model*
  - Freezes original weights, trains only a small added matrix → efficient fine-tuning
  - Can be plugged in / out freely → removing it restores the original model

- **Open-Domain vs. Scene Generalization** — *Escaping the narrow domain*
  - Both mean working beyond the trained domain (Minecraft) → into the wide world
  - *Open-domain* = the capability / distribution view · *Scene* = the visual content view

- **Autoregressive Generation** — *Extend infinitely by conditioning on the past*
  - Generates next frames conditioned on previously generated outputs
  - Enables unlimited-length interactive video (vs. fixed-length generation)

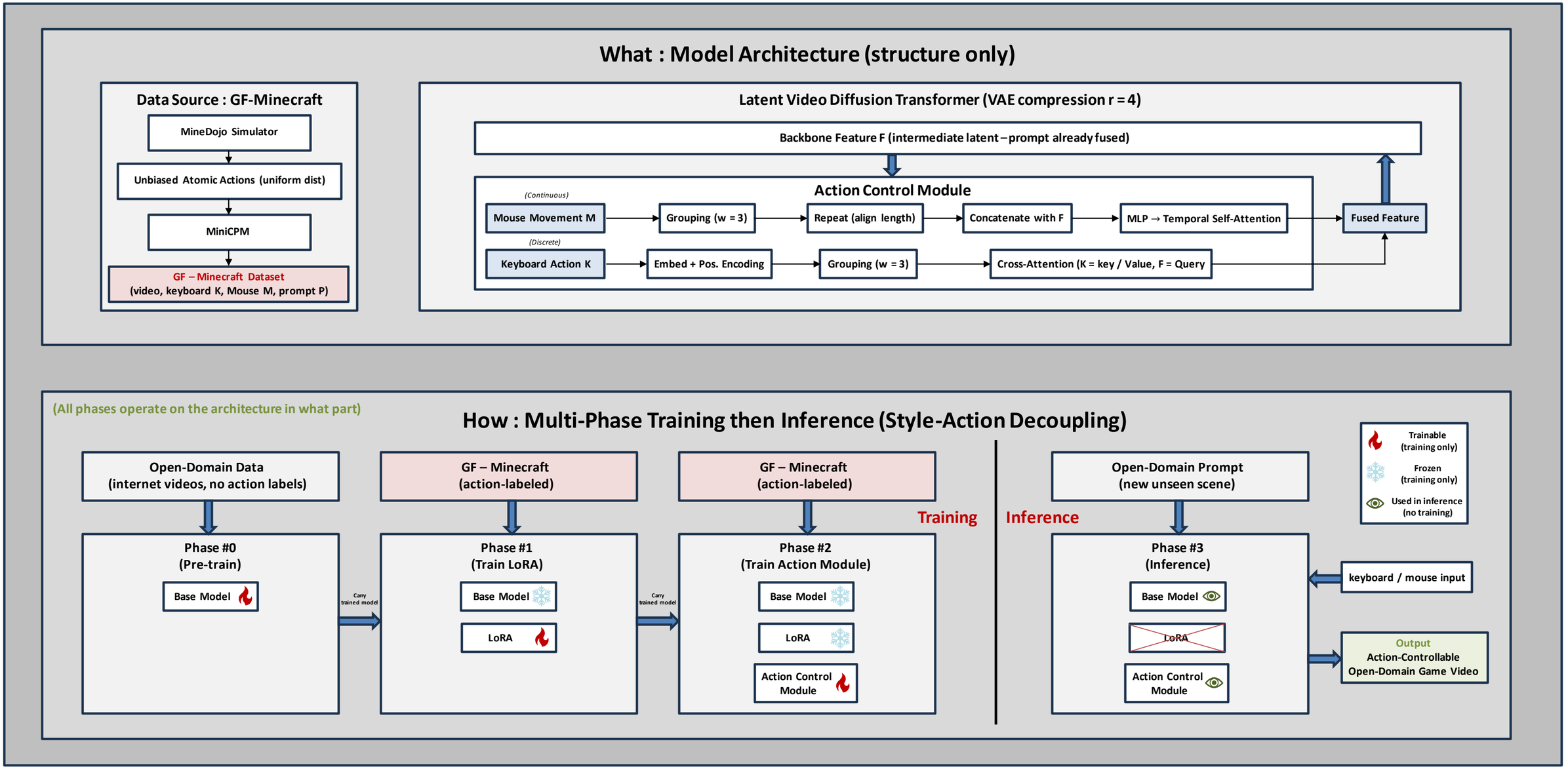
Problem

- **Game-Specific Approach** — *Trapped in a single game (Overfitting)*
  - Prior works (DOOM, Atari, CS:GO, Minecraft) overfit to individual games → cannot create content beyond existing games → *lacks scene generalization*
  - **Solution**: Reuse the open-domain generative priors already inside pre-trained video diffusion models — instead of learning scenes from scratch.

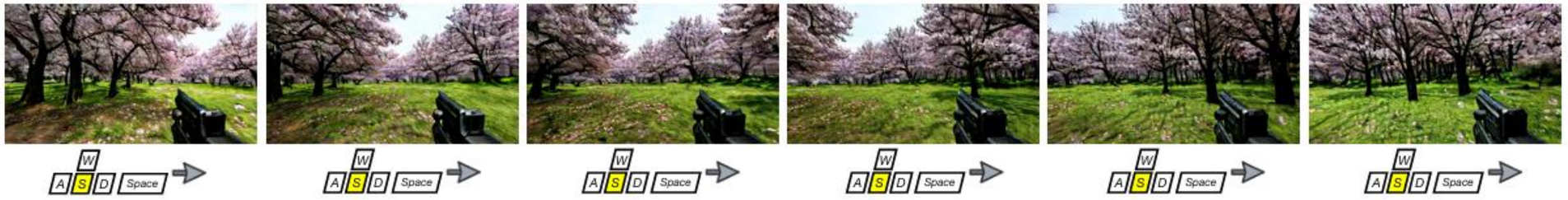
- **Direct Solution (Large-Scale Labeling)** — *Fundamentally impractical*
  - Arbitrary scenes require large-scale action-labeled data → but labeling "which key was pressed" on open-domain videos is prohibitively expensive & impractical
  - **Solution:** GF-Minecraft — learn action control from a small-scale, cheap, unbiased action-annotated dataset only; never label open-domain data.

- **Naive Fine-Tuning** — *Style Collapse*
  - Fine-tuning a pre-trained model (with open-domain prior) directly on small game data → model collapses into Minecraft style → loses open-domain generalization
  - **Solution**: Domain adapter (LoRA) + multi-phase decoupled training — style is absorbed by LoRA, then removed at inference, keeping only action control.

Method

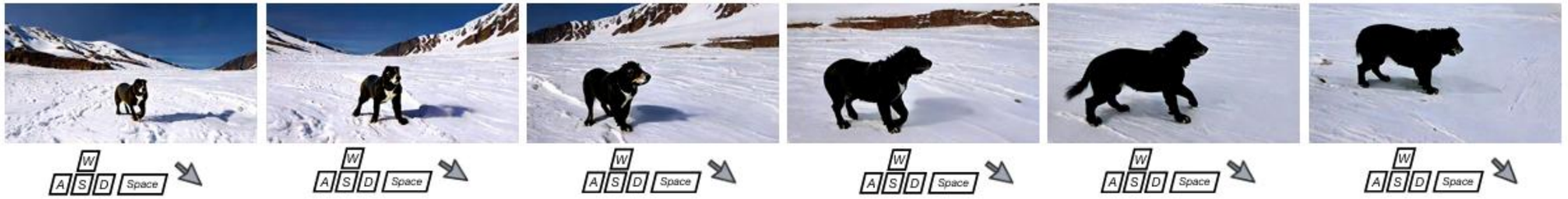

What : Model Architecture (structure only)
Data Source : GF-Minecraft
MineDojo Simulator
Unbiased Atomic Actions (uniform dist)
MiniCPM
GF – Minecraft Dataset
(video, keyboard K, Mouse M, prompt P)
Latent Video Diffusion Transformer (VAE compression r = 4)
Backbone Feature F (intermediate latent – prompt already fused)
Action Control Module
(Continuous)
Mouse Movement M
Grouping (w = 3)
Repeat (align length)
Concatenate with F
MLP → Temporal Self-Attention
Fused Feature
(Discrete)
Keyboard Action K
Embed + Pos. Encoding
Grouping (w = 3)
Cross-Attention (K = key / Value, F = Query
(All phases operate on the architecture in what part)
How : Multi-Phase Training then Inference (Style-Action Decoupling)
Open-Domain Data
(internet videos, no action labels)
GF – Minecraft
(action-labeled)
GF – Minecraft
(action-labeled)
Training
Phase #0
(Pre-train)
Base Model
Carry trained model
Phase #1
(Train LoRA)
Base Model
LoRA
Carry trained model
Phase #2
(Train Action Module)
Base Model
LoRA
Action Control Module
Inference
Open-Domain Prompt
(new unseen scene)
Phase #3
(Inference)
Base Model
LoRA
Action Control Module
Trainable (training only)
Frozen (training only)
Used in inference (no training)
keyboard / mouse input
Output
Action-Controllable
Open-Domain Game Video

Method



(1) **Prompt:** Standing in the cherry blossom forest with a gun in first person perspective.

(2) **Prompt:** An emerald green velvet accent chair sits prominently in the center of a minimalist room.



(3) **Prompt:** A large Saint Bernard walks steadily across the vast, snow-covered expanse of a mountain range.

(4) **Prompt:** In a Renaissance palace, from a first person perspective, one can see a lion in front.

03

DiWA: Diffusion Policy Adaptation with World Models

Akshay L Chandra, Iman Nematollahi, Chenguang Huang, Tim Welschehold, Wolfram Burgard, Abhinav Valada

CoRL / 2025 (spotlight)

Preliminaries

- **Imitation Learning** — *Learning a policy by copying expert demonstrations*
  - Trains a policy $\pi_\theta(a \mid s)$ on expert state-action pairs via supervised learning
  - **Behavior Cloning (BC)** is the simplest form: directly regress the expert's action

- **Distribution Shift** — *An imitation-trained policy drifting outside its training distribution*
  - Small errors accumulate → the robot reaches positions never shown in the expert demos
  - Actions for such states were never trained → the policy does not know how to respond

- **Play Data vs. Expert Demos** — *Two datasets, two purposes*
  - **Play data**: unlabeled, task-agnostic, large scale (~500K transitions) → learns environment dynamics
  - **Expert demos**: success-annotated, small scale (50 per skill) → trains the policy & reward classifier

- **Diffusion Model** — *Add noise to train, remove noise to generate*
  - **Forward**: add noise to clean data / **Reverse**: remove noise from pure noise over K steps
  - Represents **multi-modal** distributions → does not average distinct valid actions into one

- **Zero-Shot Deployment** — *Running a trained policy in a new setting without further training*
  - The policy is transferred **unchanged** — no additional interaction or adaptation

Problem

- **Imitation Learning Alone** — *Inherits IL's core limitations*
  - Diffusion policies trained purely by imitation on offline demos struggle with **distribution shifts** and fail in unseen scenarios, due to imperfect or narrowly scoped expert trajectories.
  - **Solution**: Use RL to fine-tune the pre-trained diffusion policy — improving beyond the demonstrations through trial and error.

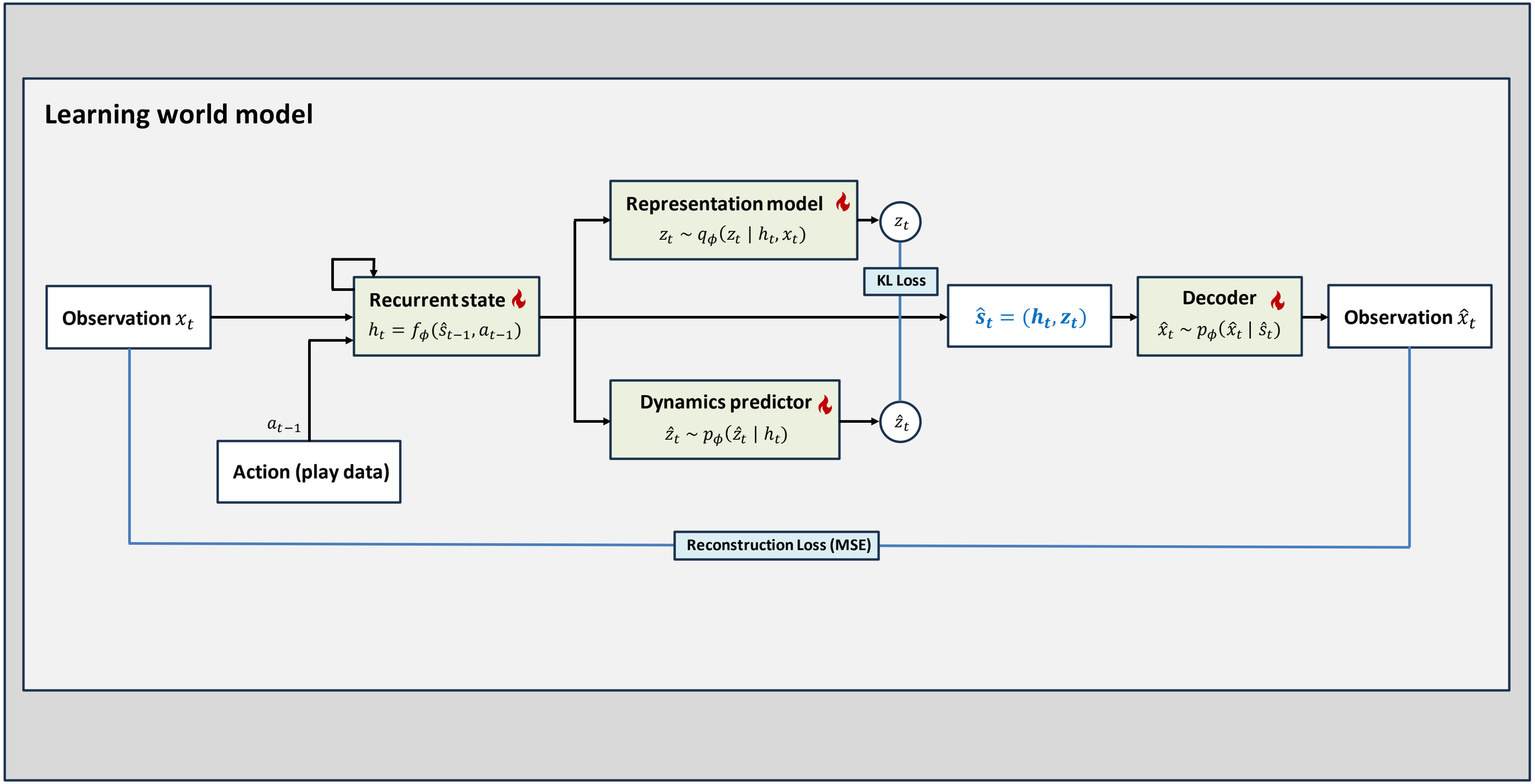
- **RL in the Real World** — *Sample-inefficient & unsafe*
  - Unlike language/vision, robotics fine-tuning demands physical interaction; deploying RL on real robots is sample-inefficient (millions of interactions) and raises **safety concerns**.
  - **Solution**: Replace the real environment with a learned world model — train entirely on offline play data and improve via imagined rollouts (0 physical interactions).

- **DPPO (prior SOTA)** — *Needs online interaction & ground-truth state*
  - The state-of-the-art DPPO fine-tunes diffusion policies via PPO, but requires millions of online interactions and relies on ground-truth simulator state, **blocking real-world use** (sim-to-real gap).
  - **Solution**: DiWA performs fine-tuning fully offline in latent space, using world-model latents instead of privileged state — enabling zero-shot real-world deployment.

Method

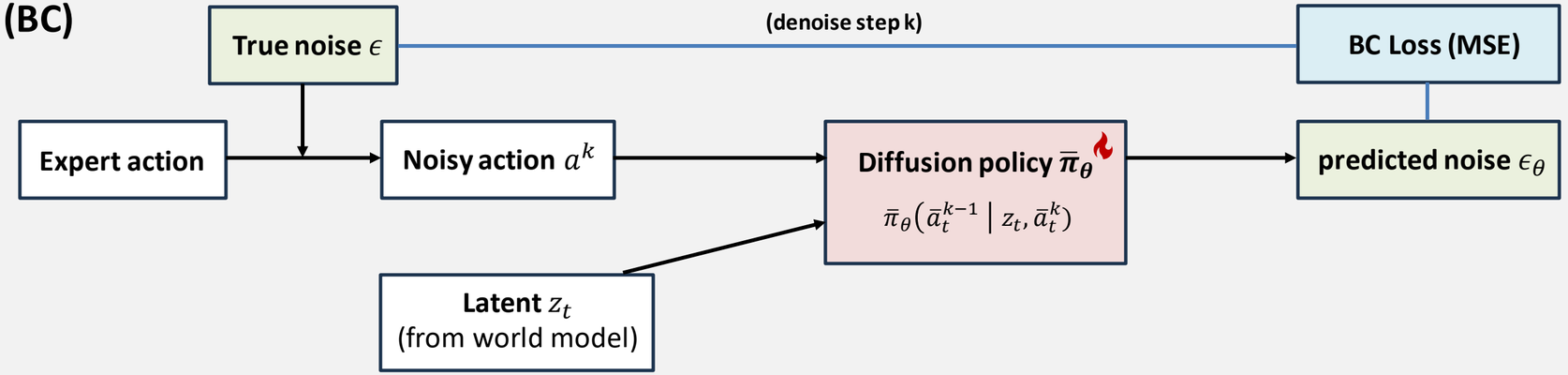
- **DiWA consists of four phases in total**
  - Train world model
  - Train diffusion policy
  - Train reward classifier
  - Dream diffusion MDP

Method (world model)

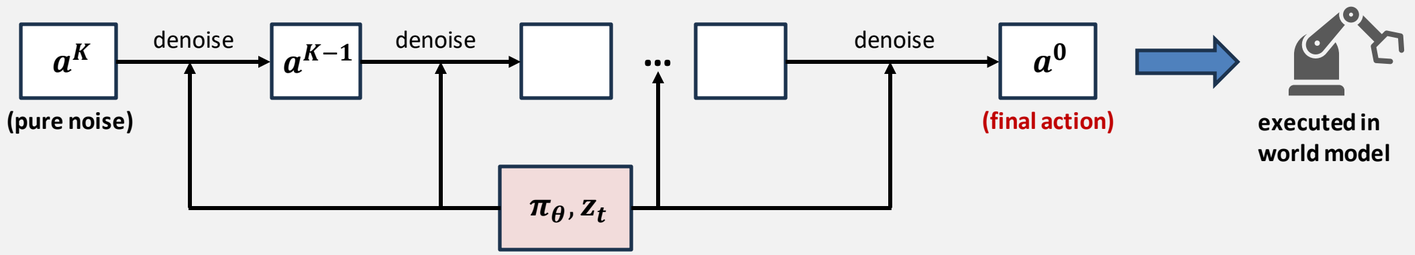


Method (Diffusion policy)

Pre-training (BC)

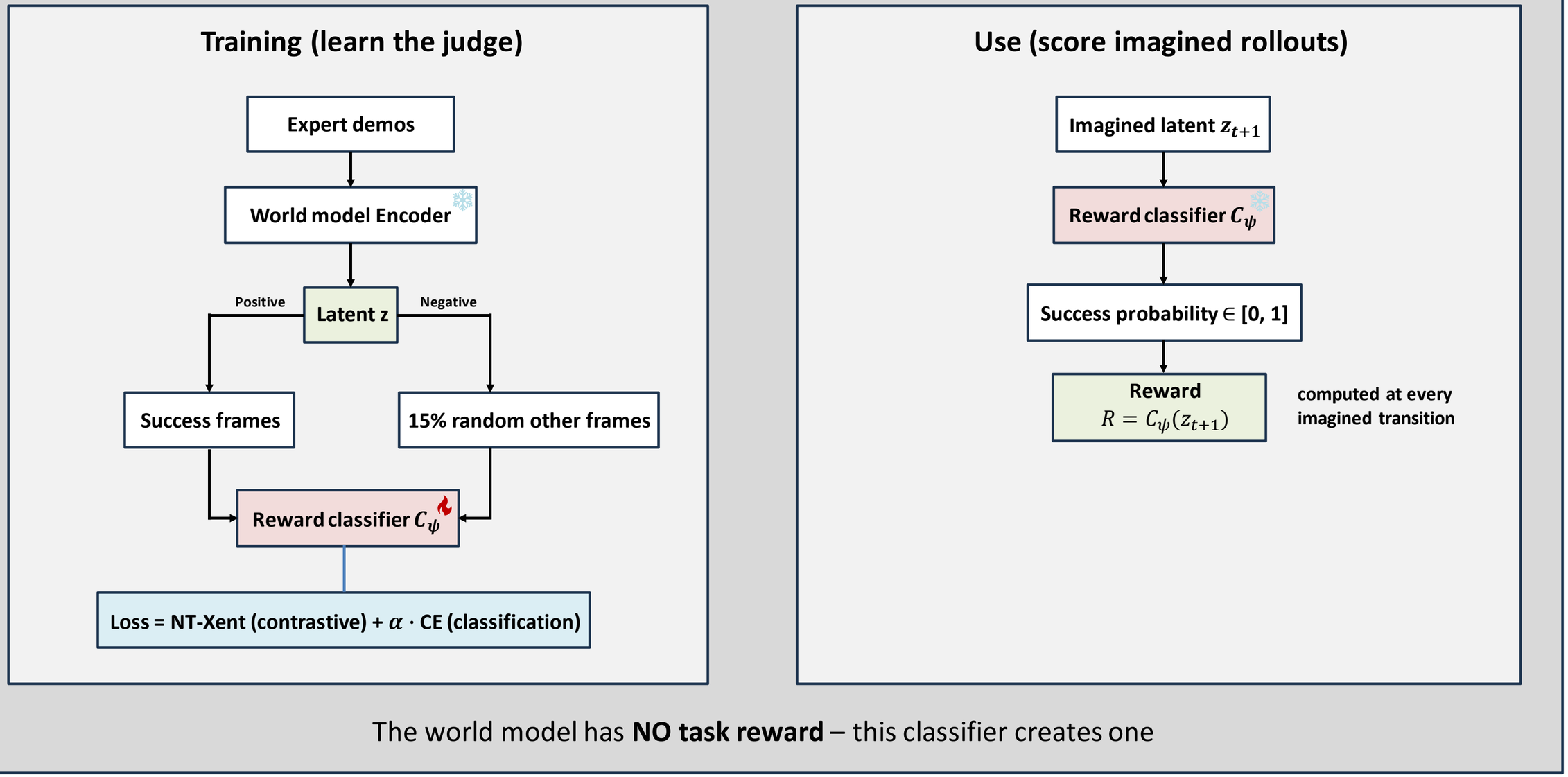


Inference (Generate action in imagination)



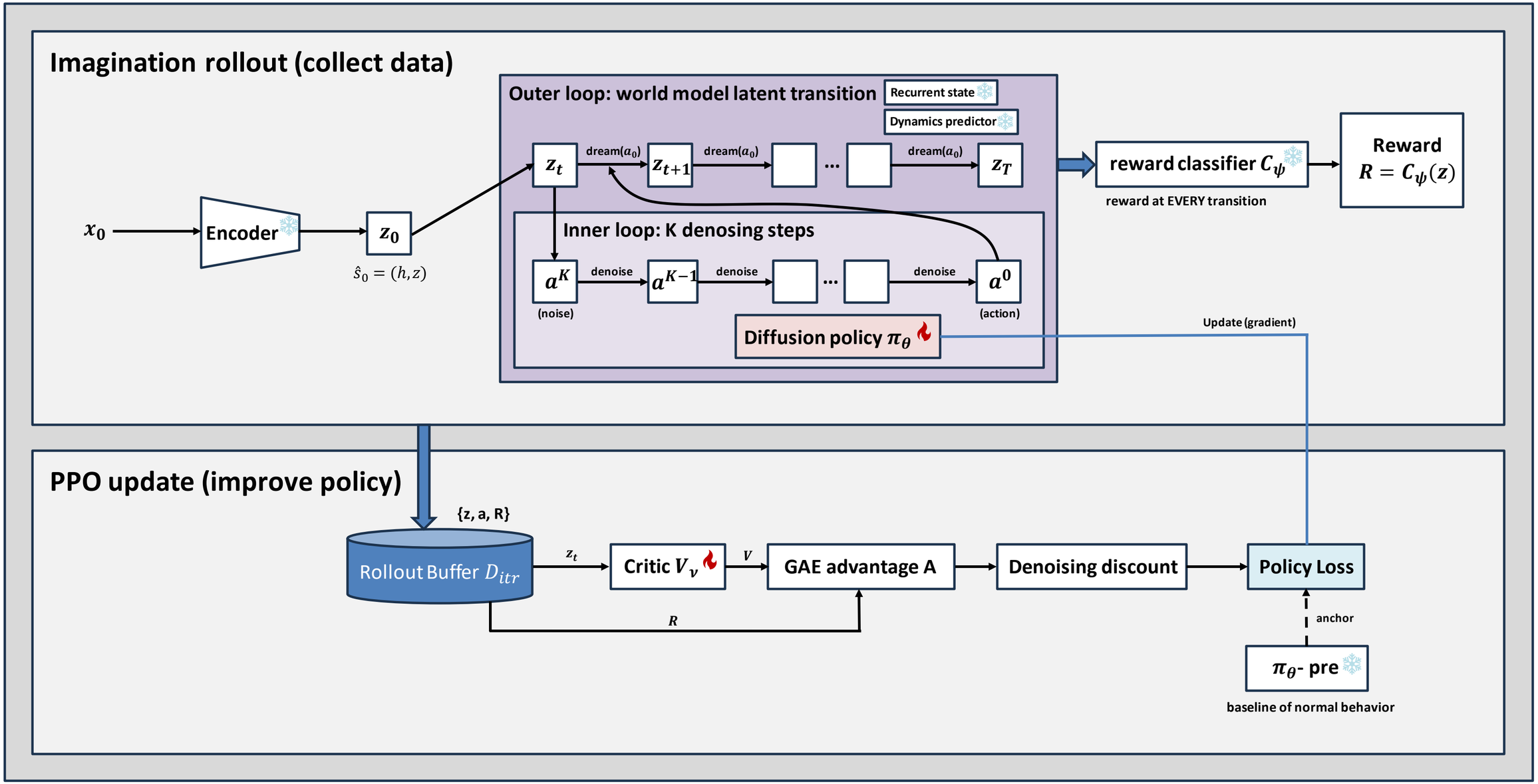
Same trainable network π_θ, used both directions: noise is **added** during **training**, **removed** during **inference**

Method (reward classifier)



The world model has **NO task reward** – this classifier creates one

Method (Dream Diffusion MDP)



Thank you

Question & Disscusion

RILS LAB