Action Chunking Transformers for SO-101 Snack Pick-and-Place

Practical Reproduction Report on Demonstrations, Training, Deployment, and Evaluation

Yanshi Robotics

Experiment: 12–13 October 2026

Report: 15 October 2026

Abstract

This project examines how an Action Chunking Transformer (ACT) can learn snack-bag pick-and-place on a low-cost SO-101 arm from human demonstrations. A leader–follower setup recorded 90 episodes containing 31,968 time steps, six-dimensional joint states and position targets, and two RGB camera streams at a nominal 30 Hz. An ACT policy with an ImageNet-initialized ResNet-18 backbone was trained for 100,000 optimizer updates with batch size 8 on an NVIDIA GeForce RTX 5070 Ti. The recorded process duration was 2 h 41 min 27 s. Five retained logging intervals show normalized action L1 decreasing from 0.346 at step 1,400 to 0.051 at step 100,000; the intervening loss history is unavailable, and no independent validation split was used. In two operator-confirmed autonomous trials without manual takeover, the bag was ultimately placed and released in the basket in both trials. The first trial completed after one grasp cycle; the second required three visible cycles. Thus final placement was observed in 2/2 trials and first-cycle completion in 1/2 trials. These observations establish task execution in the recorded conditions, while leaving generalization and reliability unmeasured. The preserved evidence connects demonstration quality, the implemented network and objective, optimization progress, and autonomous behavior; the observed placement results remain distinct from independent validation or a reliability estimate.

Contents

1 Introduction **1**

2 Task and Experimental Setup **1**

2.1 Physical setup and task . . . 1
2.2 What the policy controls . . . 4
2.3 Trial units and observable outcomes . . . 4

3 Background and Foundations **5**

3.1 Methodological context . . . 5
3.2 Observations, targets, and training samples . . . 5
3.3 Imitation learning and local action sequences . . . 7
3.4 Visual representations and attention . . . 8
3.5 Conditional latent model and training objective . . . 10
3.6 Action chunking and feedback . . . 15
3.7 Evaluation quantities and diagnostic models . . . 18

4 ACT Network Architecture **21**

4.1 Visual backbone and input projections . . . 21
4.2 Tensor shapes and positions . . . 21
4.3 Encoder, decoder, and regression head . . . 22
4.4 Stored model size . . . 22
4.5 Original method, implementation, and actual settings . . . 22

5 Demonstrations and Dataset **28**

5.1 Dataset size and version identity . . . 28
5.2 File batches and spatial coverage . . . 28
5.3 Inspection scope . . . 30
5.4 Quality observations retained in training . . . 30
5.5 Data separation and leakage . . . 31
5.6 Recorded example and teacher action block . . . 31

6 Training and Deployment Protocol **33**

6.1 Software and initialization . . . 33
6.2 Training configuration . . . 33
6.3 Run management and retained artifacts . . . 34
6.4 Deployment settings and operator actions . . . 35
6.5 Implementation and reproducibility materials . . . 35

7 Results and Analysis **35**

7.1 Completion evidence and checkpoints . . . 35
7.2 Retained loss measurements . . . 36
7.3 Loss composition and interpretation . . . 38
7.4 GPU utilization and memory . . . 39
7.5 Offline-check status and limitations . . . 40
7.6 Autonomous trials . . . 40

8 Discussion and Future Work **46**
8.1 What the evidence supports . . . 46
8.2 Why low average error can coexist with grasp misses . . . 46
8.3 Candidate explanations and discriminating tests . . . 46
8.4 A staged next evaluation . . . 46
8.5 Controlled ablations . . . 47
8.6 Limitations . . . 47

9 Conclusion **48**

A Notation and Units **49**

B Saved Configuration and Evidence Provenance **50**
B.1 Configuration records . . . 50
B.2 Files that support the measurements . . . 51
B.3 Model, video, and source identities . . . 52
B.4 Key implementation locations . . . 53

C Extended Video Frame Sequences **54**

D Common Questions **56**
D.1 Task and data . . . 56
D.2 Network and objective . . . 56
D.3 Execution and feedback . . . 57
D.4 Training and evaluation . . . 57

References **59**

1 Introduction

A camera can show a snack bag and a basket, but an SO-101 controller ultimately needs a sequence of joint position targets. The experiment asks whether human demonstrations can supply the relation between those observations and targets without programming a separate grasp trajectory for every initial bag position. ACT learns this relation by predicting a short sequence of future actions from the current images and joint state (Zhao et al., 2023).

The task includes several distinct demands: approaching a flexible package, closing the gripper at a useful contact, lifting without losing the package, moving above a basket, and releasing. Errors near contact can decide the outcome even when the remainder of the trajectory is close to a demonstration. This makes the task useful for studying the difference between a supervised learning objective and a physical task criterion.

The available experiment contains one training run and two recorded robot trials. Within Lets Build Robots S1, the project documents a practical ACT reproduction from demonstration collection through training to deployment. Data, implementation, optimization, and visible behavior are examined as separate sources of evidence. Figure 1 gives the complete chain; the action queue is included because predicting a block and executing that block are separate operations.

The completed work comprises a frozen demonstration dataset, a locally trained ACT policy, preserved training and configuration records, and two autonomous execution recordings. No competing policy, independent validation set, or multi-seed study was evaluated. The distinction between task completion and generalization is maintained throughout the analysis.

2 Task and Experimental Setup

2.1 Physical setup and task

The apparatus consists of an SO-101 leader arm, an SO-101 follower arm, a two-finger gripper, a flexible snack package, a basket, and two RGB cameras. During demonstration collection, the operator moves the leader and the follower tracks the corresponding targets. During the recorded policy trials, the follower receives ACT-generated targets. Figure 2 shows the physical scene from the external recording viewpoint.

The fixed camera, named `top`, observes the bag, arm, and basket. The camera named `wrist` moves with the gripper and provides a closer view of contact. These roles are complementary: the fixed view supplies task geometry, while the wrist view supplies local appearance around the fingertips. Both policy inputs are 640×480 RGB; the external trial recordings are 1280×720. The larger recording dimensions are not the policy input resolution.

The task string stored with the demonstrations is:

> Pick the snack bag, place it in the snack box, then return empty to the rest pose.

This sentence identifies the dataset task. The installed ACT model does not encode the sentence with a language model. It learns a single-task relation between visual and joint observations and action targets. Consequently, this experiment does not test language-conditioned instruction following.

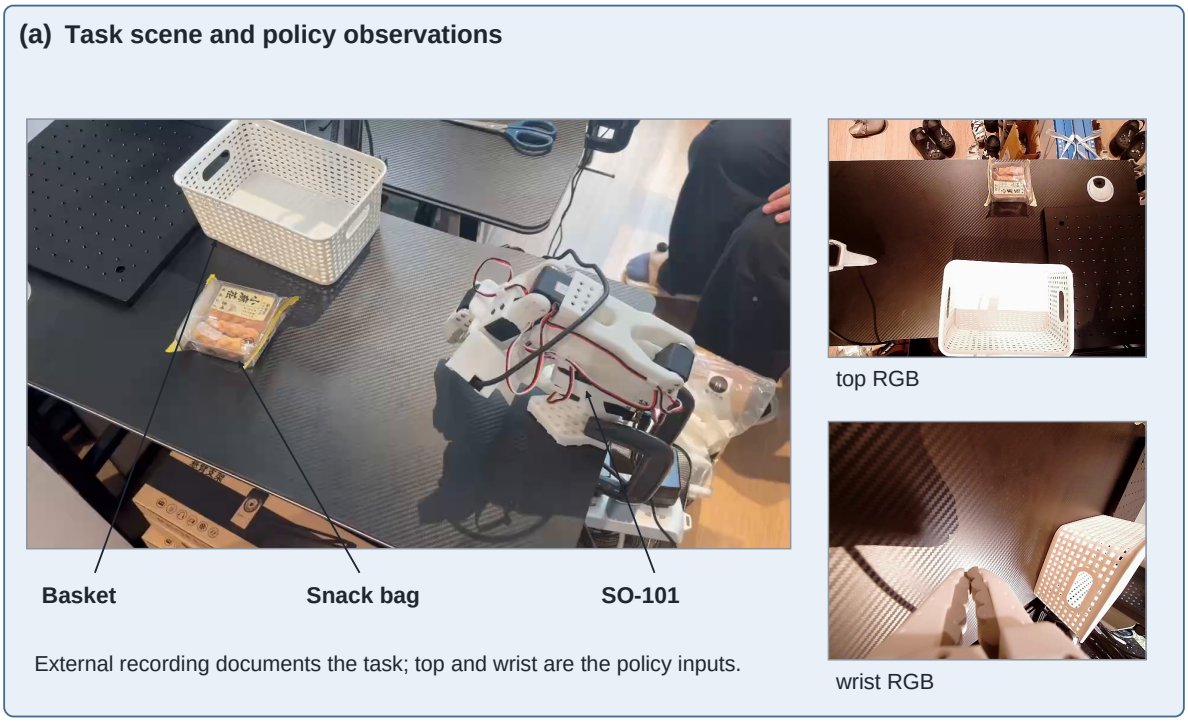
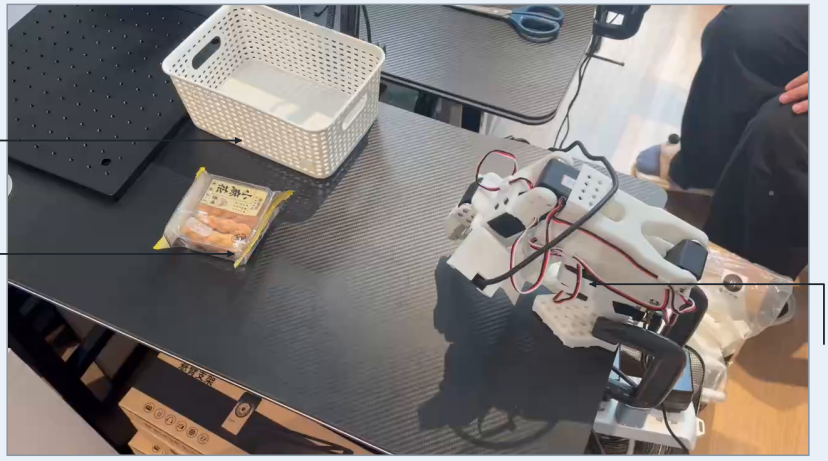


Figure 1: Project overview. The external scene photograph comes from video 2, while the two policy-camera examples come from teleoperated episode 8. The lower sequence shows video 1. Photographs document actual observations; arrows summarize the configured demonstration, optimization, and deployment paths and do not reconstruct control timing.

(a) SO-101 snack pick-and-place apparatus



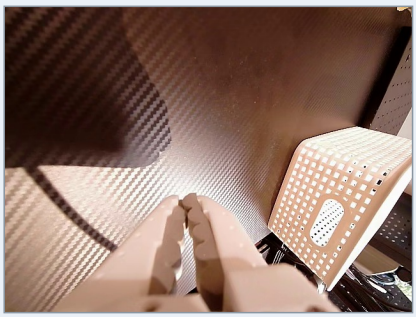
External documentation view · 1280 × 720

(b) Two complementary policy-camera inputs



top RGB · 640 × 480

Fixed scene geometry



wrist RGB · 640 × 480

Local gripper appearance

Figure 2: The external recording at the start of video 2, with the apparatus labeled, and the two policy-camera examples from demonstration episode 8. The external camera documents the scene; ACT receives only top and wrist RGB. Object dimensions and exact initial coordinates were not measured.

2.2 What the policy controls

Let $q_t \in \mathbb{R}^6$ be the follower state read at time t, and $a_t \in \mathbb{R}^6$ the commanded target. Table 1 defines the ordering. The first five components use degrees in this configuration; the gripper uses a 0–100 opening scale. The gripper value is not an angle in degrees, a measured force, or a distance in millimeters.

Table 1: State and action components, in dataset order.

Index	Feature name	Physical meaning	Unit
1	`shoulder_pan.pos`	Base/shoulder horizontal rotation	degrees
2	`shoulder_lift.pos`	Shoulder lifting position	degrees
3	`elbow_flex.pos`	Elbow flexion	degrees
4	`wrist_flex.pos`	Wrist flexion	degrees
5	`wrist_roll.pos`	Wrist rotation	degrees
6	`gripper.pos`	Gripper opening target or reading	0–100 scale

The learned policy specifies joint positions. The motor electronics still perform the lower-level tracking that converts targets into motor actuation. ACT does not directly produce Cartesian end-effector coordinates, inverse-kinematics solutions, or motor torques. A change in target can require time to realize, so generally $a_t \neq q_t$. This distinction matters both for training and for the offline evaluator: replacing the observed state with the teacher target changes the model input.

The images do not determine the contact mechanics completely. A flexible package can deform, reflect light, and slip. A two-finger gripper can therefore follow a visually plausible path while failing to maintain contact. The videos establish successful transport and release for the two shown cases, but no bag dimensions, basket-rim height, contact force, or workspace-clearance measurements were taken. Those quantities are not reconstructed from an uncalibrated photograph.

2.3 Trial units and observable outcomes

A *trial* is one operator-designated attempt at the complete task. A *grasp cycle* is a visible approach, closing/contact, and lifting movement within that trial. Passing near the bag does not automatically count as another cycle. The definitions in Table 2 were applied retrospectively to the supplied videos; they were not a preregistered acceptance protocol.

Table 2: Observable criteria used for the two video cases.

Outcome	Criterion
Final placement and release	Bag enters the basket, separates from the gripper, and remains in the basket at the end of the recording.
First-cycle completion	The first grasp cycle sustains the bag through transport and release; a brief lift followed by a return to the table does not satisfy it.
Empty retraction	The gripper visibly withdraws without the bag. Its controller source is recorded separately.
Manual intervention	Operator contact or takeover is documented when known; absence of visible contact alone cannot establish all control inputs.

The operator confirmed that the two supplied trials were ACT-controlled and had no manual

takeover. The visible frames do not show a person touching the bag, basket, or arm during the task. The camera does not cover every control device, so the autonomy statement is attributed to that confirmation. The return motion requires a separate qualification: the deployment program also interpolates toward the initial pose during shutdown. Visible retraction therefore does not establish that the entire return was generated by the learned policy.

3 Background and Foundations

The learning problem maps two camera views and the measured follower configuration to future position targets. Supervised sequence regression, attention, conditional latent learning, and queue-based execution determine which information is available when the gripper approaches, closes, lifts, and releases the package. Constructed calculations below are separate from experiment measurements.

3.1 Methodological context

Behavior cloning learns an action predictor from expert observations and actions. ACT extends this idea to a block of actions and uses a Transformer with a conditional variational training objective (Zhao et al., 2023). The image backbone comes from residual networks (He et al., 2016); attention follows Vaswani et al. (2017); the parallel query decoder is related to DETR (Carion et al., 2020). Variational and conditional variational learning provide the latent-variable background (Kingma and Welling, 2014; Sohn et al., 2015).

The original ACT study used a different platform: a bimanual ALOHA system, four camera views, and a 50 Hz setting. Here the policy controls one SO-101 follower with two camera views and a nominal 30 Hz loop. Architectural ideas transfer; published task success rates do not become expected performance for this hardware and dataset.

Table 3 compares mechanisms, not experimental scores. No single-step or diffusion baseline was trained in this study. The contrast with Diffusion Policy is useful because both predict action sequences, while the sequence-generation procedure differs (Chi et al., 2023).

Table 3: Conceptual comparison of policy mechanisms. No comparative performance experiment was conducted.

Method	Output at a query	Generation	Execution choice
Single-step behavior cloning	One action	Direct network prediction	Request an action each policy step
ACT	A fixed-length action block	Parallel Transformer decoder; latent fixed at deployment	Queue a prefix or combine overlapping predictions
Diffusion Policy	An action sequence	Iterative denoising	Execute a prefix, then replan

3.2 Observations, targets, and training samples

An *episode* is a recorded demonstration trajectory. A saved *frame* or time step contains the joint state, action target, image references, task identity, and indexing fields. A training *sample* chooses an anchor time t, obtains its current observation, and requests the following action block. A *batch* groups samples; an optimizer *step* updates parameters using that batch. An *epoch* is a

pass-equivalent through the sampling anchors, not a robot trial.

For two cameras, define the observation as

$$o_t = (I_t^{\text{top}}, I_t^{\text{wrist}}, q_t), \qquad A_t = (a_t, a_{t+1}, \ldots, a_{t+k-1}) \in \mathbb{R}^{k\times 6}. \tag{1}$$

There are exactly k targets in A_t, including the anchor target. The installed ACT action offsets are $0, 1, \ldots, k-1$. A sample does not require k new images: this configuration uses one current observation and predicts the future target sequence from it.

At an episode boundary, requested future indices cannot be taken from the next episode. The data pipeline supplies boundary values together with a padding mask. The mask distinguishes genuine targets from entries inserted to complete the fixed shape. The training loss and latent encoder must respect that mask. Otherwise the final pose can receive artificial importance simply because many repeated targets were inserted.

3.2.1 Normalization and physical units

ACT receives images and a joint vector whose numerical scales differ. Pixel values, degree-valued joints, and the gripper opening cannot be treated as a single unprocessed array. The saved processors normalize state and action components by their means and standard deviations. For component d, the conceptual transformation is

$$\tilde{q}_{t,d} = \frac{q_{t,d} - \mu_{q,d}}{s_{q,d} + \varepsilon_s}, \qquad \tilde{a}_{t,d} = \frac{a_{t,d} - \mu_{a,d}}{s_{a,d} + \varepsilon_s}, \tag{2}$$

where ε_s denotes numerical stabilization, distinct from latent sampling noise; the saved normalizer uses 10^{-8}. The checkpoint processors define the actual transformation and its constants. Deployment must reverse the action transformation before sending physical targets. A normalized value near zero means near the recorded mean; it does not mean zero degrees or a closed gripper.

Image normalization uses the saved ImageNet statistics in this run. For an RGB channel scaled to $[0, 1]$, the corresponding mean is subtracted and the channel standard deviation divides the result. ResNet-18 then maps the image to a spatial feature grid. Its residual blocks add a learned transformation to a skip path, $y = F(x) + x$, helping optimize deeper visual networks (He et al., 2016).

Equation (2) is reversed by multiplying by the stored standard deviation and adding the mean, subject to the processor's stabilization convention. As a constructed example, a joint with mean 20 degrees and standard deviation 10 degrees maps a normalized target of 0.2 to about 22 degrees. A gripper with mean 40 and standard deviation 20 maps a normalized target of -0.5 to about 30 on its 0–100 scale. These values illustrate units; they are not the recorded dataset statistics.

Normalization permits a shared optimization scale across heterogeneous components, but it does not eliminate the need to inspect physical-unit errors. A normalized six-joint mean can conceal one joint with a larger physical deviation. Conversely, an unnormalized aggregate over degrees and gripper points has no single physical unit. Per-joint reporting is needed for either interpretation.

3.3 Imitation learning and local action sequences

3.3.1 Learning targets from examples

A policy is a function that maps an observation to an action. In ordinary behavior cloning, a parameterized policy π_θ is fitted to demonstrated pairs (o_t, a_t). A generic empirical objective is

$$\hat{R}(\theta) = \frac{1}{N} \sum_{t=1}^{N} \ell(\pi_\theta(o_t), a_t), \tag{3}$$

where ℓ measures disagreement with the teacher. Gradient-based optimization changes θ so that this disagreement decreases on sampled data. No task reward or reinforcement-learning exploration was used in the recorded training run.

The teacher action is a command label, not a guarantee of achieved pose. A state reading describes what the follower reports; a target describes where it was asked to go. If the follower lags during motion, the two can differ systematically. Feeding the teacher action as if it were the measured state can provide a more favorable or simply different input distribution than the deployed policy will encounter.

3.3.2 Why an action sequence can help

Fine manipulation contains correlated decisions. Closing, lifting, and transporting depend on timing and on what happened just before them. Predicting a block allows the network to represent that local sequence jointly. It also reduces the number of policy queries when several targets are executed from each prediction. These are different effects: joint prediction concerns the model output, while reduced querying concerns the execution wrapper.

For ACT, replace the one-step output with $\hat{A}_t = f_\theta(o_t, z)$. Its k rows share the encoded observation, latent variable, and decoder computation. In the saved one-layer decoder, observation-dependent output representations are formed by cross-attention; they are not subsequently mixed by another decoder self-attention layer. The shared representation does not enforce physical feasibility or guarantee a secure grasp, and the predicted block is not a motion-planning certificate.

Multiple demonstrations can respond to similar observations with different timing or paths. With an L2 pointwise objective, a deterministic regressor is associated with a conditional mean; with L1, a componentwise conditional median. Either can be unsuitable when different valid trajectories form distinct alternatives. The conditional latent variable is intended to help represent variation during training. It does not prove that the learned latent coordinates have interpretable styles or that deployment samples several plans.

3.3.3 Distribution shift during execution

Training observations are generated by the operator's demonstration policy. A deployed learner generates its own future observations through its actions. If a grasp is slightly displaced, the next wrist image and joint configuration can differ from those commonly seen in training. A subsequent action is then predicted in a changed context. This dependence is why small supervised error does not automatically translate into reliable sequential execution (Ross et al., 2011).

The second video contains repeated cycles and eventual completion. That observation is compatible with several explanations, including visual feedback at later queries and execution of a

learned repeated pattern. It does not by itself identify distribution shift as the cause of the earlier failures or prove a learned failure detector. A discriminating experiment would deliberately vary the post-failure state while recording query times and the resulting actions.

Corrective demonstrations are a possible next data source. The term DAgger should be reserved for a process that iteratively gathers states visited by the learner and supplies expert labels for them (Ross et al., 2011). Merely collecting more successful episodes is not sufficient to establish that protocol.

3.4 Visual representations and attention

A fixed-camera image contains scene layout, while the wrist image emphasizes the package near the fingers. A convolutional backbone transforms both into local feature vectors. These vectors are representations learned from image statistics and demonstration losses, not explicit bag coordinates or measured contact forces. Attention can combine those representations with the current joint configuration when predicting the next segment of motion.

3.4.1 Attention, positions, and information retrieval

Attention lets one representation retrieve relevant information from a set of other representations. A query asks what information is useful; keys describe which information can be matched; values contain the information that is combined. Learned linear projections construct these vectors. For one head,

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V. \tag{4}$$

Each query row is normalized across the key positions. The resulting nonnegative weights sum to one for that query. They weight value vectors; they are not probabilities that the robot will grasp successfully. Scaling by $\sqrt{d_k}$ limits the growth of dot-product magnitude as the feature dimension increases (Vaswani et al., 2017).

Self-attention takes queries, keys, and values from the same sequence. In the policy encoder it allows image and nonvisual tokens to exchange information. *Cross-attention* takes queries from one sequence and keys/values from another. The ACT decoder uses output queries to read the encoded observation. Multiple heads learn different projections; with hidden size 512 and eight heads, each head has key dimension 64. The symbol H denotes head count here so that h remains reserved for the executed action prefix.

Figure 3 shows a constructed calculation. Scores $[1, 0, -1]$ give weights approximately $[0.665, 0.245, 0.090]$. With values $(2, 0)$, $(0, 2)$, and $(-2, 0)$, the output is approximately $(1.150, 0.489)$. These numbers explain the operation; they are not attention measurements from the trained policy. The calculation is expanded below.

Position information is needed because attention itself does not assign physical or temporal meaning to a slot. Reordering input tokens reorders a self-attention computation in a corresponding way when position information is absent. Spatial and temporal encodings tell the model which location a representation belongs to; they do not supply a hard-coded robot trajectory.

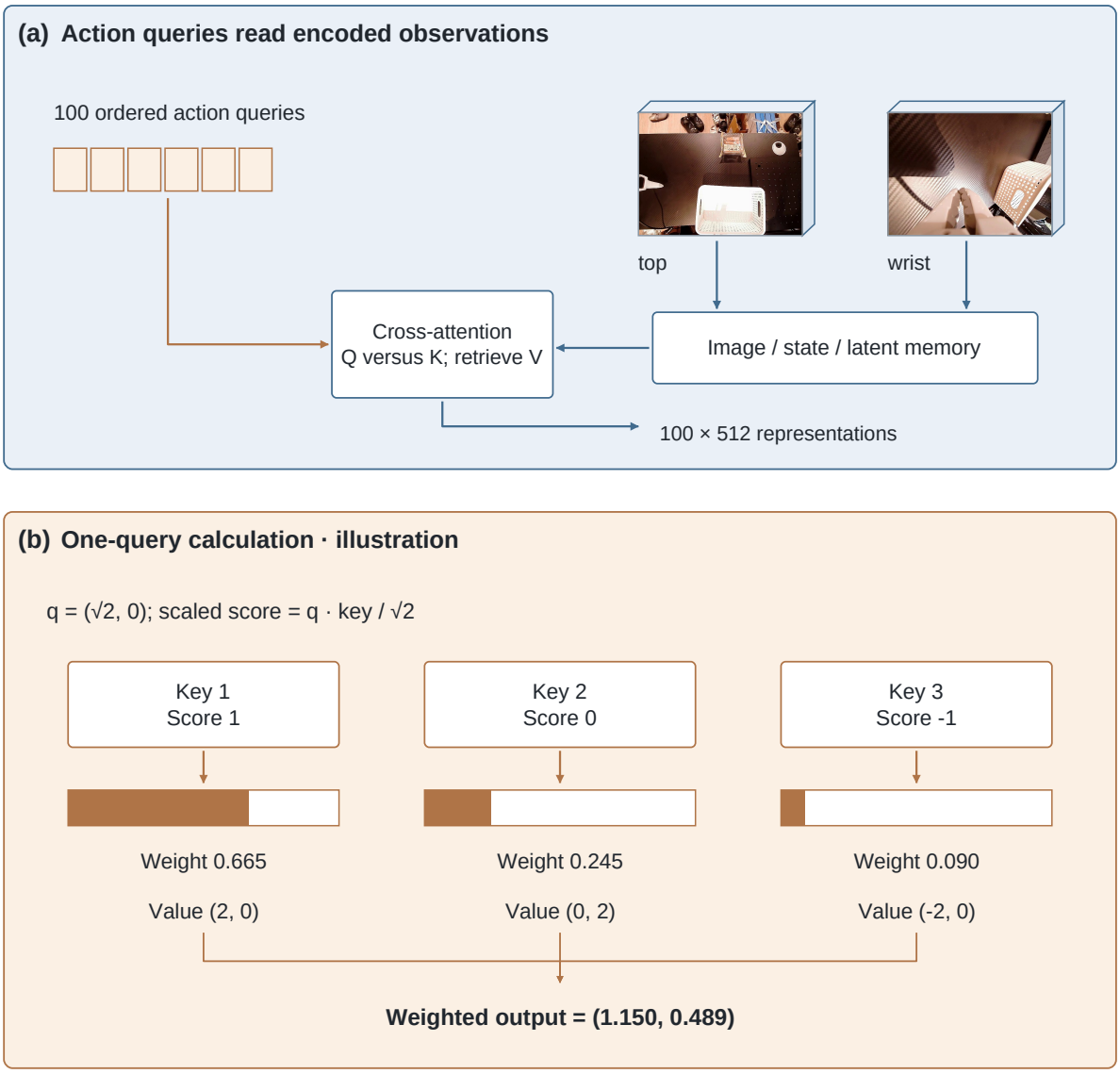


Figure 3: A numerical attention illustration. The scores and value vectors are constructed to illustrate attention, independently of the robot data. Each query produces a weighted value vector, not a grasp-success score.

3.4.2 Worked attention calculation

Let $X_q \in \mathbb{R}^{n_q \times d_{\text{model}}}$ contain query-side representations and $X_k \in \mathbb{R}^{n_k \times d_{\text{model}}}$ contain source representations. A head forms

$$Q = X_q W_Q, \quad K = X_k W_K, \quad V = X_k W_V, \tag{5}$$

with $Q \in \mathbb{R}^{n_q \times d_k}$, $K \in \mathbb{R}^{n_k \times d_k}$, and $V \in \mathbb{R}^{n_k \times d_v}$. The score matrix $QK^\top$ has one row per query and one column per key. Row-wise softmax produces $P \in \mathbb{R}^{n_q \times n_k}$, and $PV \in \mathbb{R}^{n_q \times d_v}$ is the retrieved representation.

As an initialization intuition, if the components of a query and a key are independent, centered, and have unit variance, the variance of their dot product grows approximately as d_k. Dividing by $\sqrt{d_k}$ brings that variance to order one. This motivates the scaling; it is not an assertion that every trained representation remains independent and unit-variance (Vaswani et al., 2017).

For the two-dimensional worked example, take

$$q = (\sqrt{2}, 0), \quad K = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \end{bmatrix}, \quad V = \begin{bmatrix} 2 & 0 \\ 0 & 2 \\ -2 & 0 \end{bmatrix}. \tag{6}$$

The scaled scores are $(1, 0, -1)$. With $Z = e^1 + e^0 + e^{-1}$,

$$P = \left(\frac{e}{Z}, \frac{1}{Z}, \frac{e^{-1}}{Z} \right) \approx (0.66524, 0.24473, 0.09003), \tag{7}$$

and

$$PV = (2(0.66524) - 2(0.09003),\ 2(0.24473)) \approx (1.15042, 0.48946). \tag{8}$$

The output dimension follows the value dimension, not the number of keys. In multi-head attention, separately projected head outputs are concatenated and projected by W_O. For the experiment, eight 64-dimensional heads produce a concatenated 512-dimensional representation.

In this run, the decoder cross-attention has $n_q = 100$ action positions and $n_k = 602$ observation-side tokens in the architecture-derived shape ledger. Its attention weights therefore concern which encoded information each future action position reads. They do not constitute 602 object detections or a distribution over 100 task outcomes.

3.5 Conditional latent model and training objective

A conditional variational autoencoder introduces a latent variable z to model variation in outputs associated with a condition. In ACT, the condition for the action predictor contains the current images and joint state, and the output is a demonstrated action block. The training posterior encoder estimates

$$q_\phi(z \mid q_t, A_t) = \mathcal{N}\Big(\mu_\phi(q_t, A_t), \operatorname{diag}\big(\sigma^2_\phi(q_t, A_t)\big)\Big). \tag{9}$$

In the inspected implementation, that encoder sees joint state and future teacher actions, not the camera images. The policy's observation encoder still sees images, state, and the resulting latent sample. This is a particular conditional design, not a requirement that every CVAE use the same recognition inputs (Kingma and Welling, 2014; Sohn et al., 2015; Zhao et al., 2023).

The latent dimension is 32. The encoder outputs 32 means and 32 log variances, not 64 independently commanded robot values. A learned summary token aggregates the posterior input sequence. The other tokens consist of the projected current state and the 100 projected teacher targets, with padded targets masked.

Figure 4 separates training and deployment. Future teacher actions are legitimate training targets; they are unavailable during real execution. The latent encoder is therefore a training component. Removing it at inference does not remove the observation encoder inside the policy.

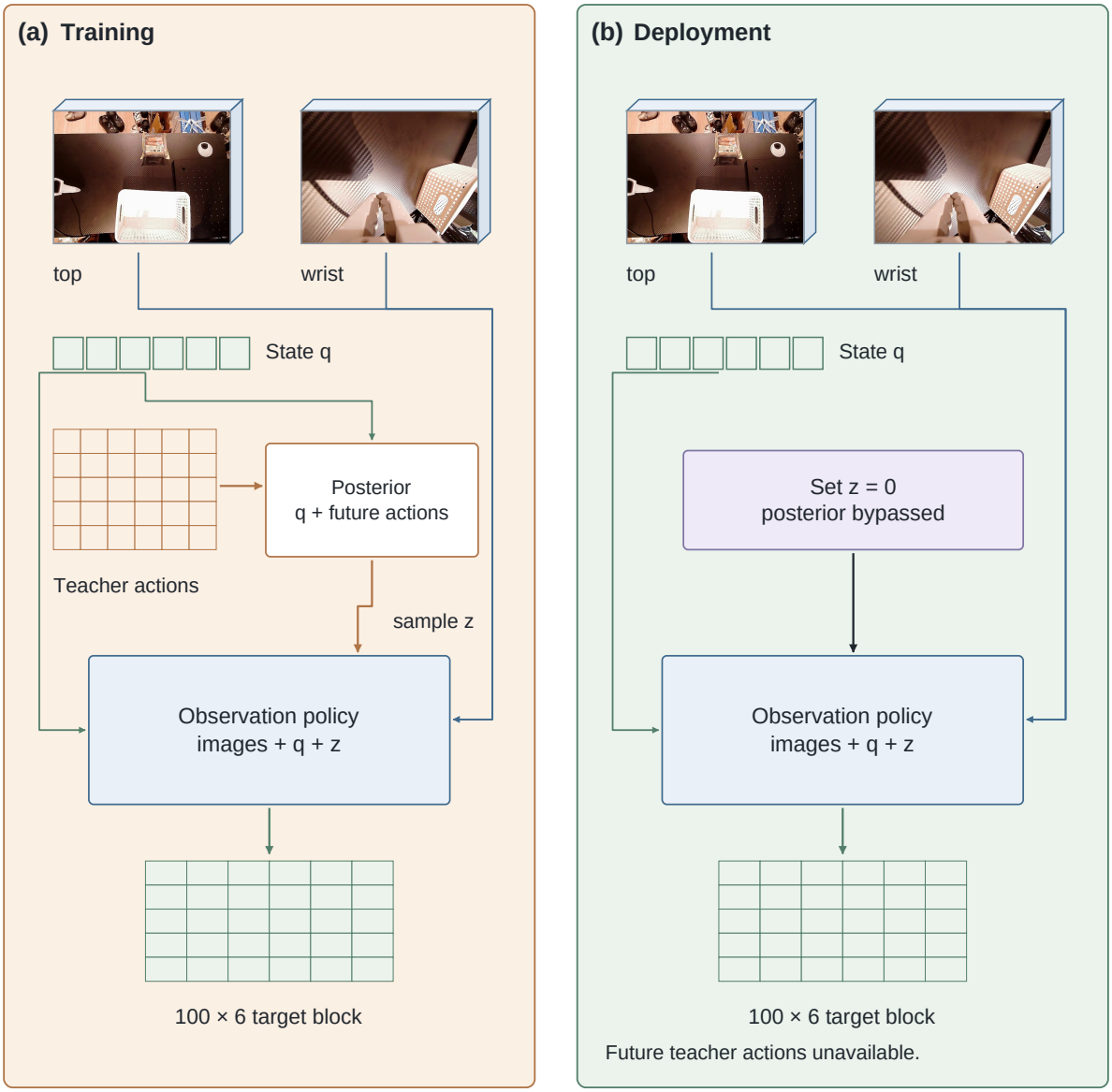


Figure 4: Training and deployment of the conditional latent model. The training posterior consumes the state and future teacher actions. Deployment fixes z to zero and retains the visual/state policy network; no future ground-truth actions enter the deployed predictor.

3.5.1 Conditional evidence lower bound

A latent-variable model writes the conditional output density as

$$p_\theta(A \mid o) = \int p_\theta(A \mid o, z) p(z)\, dz. \tag{10}$$

For a variational distribution $q_\phi(z \mid q, A)$ with suitable support, insert the same density in numerator and denominator:

$$\log p_\theta(A \mid o) = \log \mathbb{E}_{q_\phi}\left[\frac{p_\theta(A \mid o, z)p(z)}{q_\phi(z \mid q, A)}\right] \tag{11}$$

$$\geq \mathbb{E}_{q_\phi}[\log p_\theta(A \mid o, z)] - D_{\mathrm{KL}}(q_\phi(z \mid q, A)\|p(z)). \tag{12}$$

The inequality follows from Jensen's inequality. The right-hand side is the evidence lower bound, or ELBO. Maximizing it encourages reconstruction likelihood while regularizing the inferred latent distribution (Kingma and Welling, 2014; Sohn et al., 2015). A variational posterior may ignore some conditioning information, such as the images, and still define a bound; that choice affects the approximation family.

For independent Laplace output components with common scale b, a negative log likelihood contains

$$\sum_{j,d} \frac{|A_{j,d} - \hat{A}_{j,d}|}{b} + N \log(2b), \tag{13}$$

where N is the number of scored scalar elements. This explains a connection between L1 reconstruction and a likelihood assumption. The implemented mean over valid elements, the handling of padding, and the chosen KL weight alter the relative scale. The numeric training loss should therefore be interpreted through Equations (17)–(20), rather than assigned an unmeasured probabilistic calibration.

For comparison, an L2 reconstruction term corresponds to a different noise assumption. Choosing L1 versus L2 changes the objective and its response to large errors. A paper's high-level "reconstruction" label alone is not sufficient to reproduce the actual loss reduction.

3.5.2 Reparameterization and its gradients

Direct sampling from a distribution described by neural-network parameters makes naive differentiation awkward. Reparameterization writes sampling as a deterministic transformation of independent noise (Kingma and Welling, 2014). If $\ell = \log \sigma^2$, the implementation uses

$$\epsilon \sim \mathcal{N}(0, I), \qquad z = \mu + \exp(\ell/2) \odot \epsilon. \tag{14}$$

The random draw is ϵ; gradients can pass through the transformation to μ and ℓ. Dividing the log variance by two is essential: $\exp(\ell)$ is the variance, whereas $\exp(\ell/2)$ is the standard deviation.

The prior is the fixed standard normal $p(z) = \mathcal{N}(0, I)$. General CVAEs may use a learned conditional prior, but the ACT code inspected here does not require an additional prior network. The KL term discourages the posterior from becoming arbitrarily different from this deployment reference.

For $z_r = \mu_r + e^{\ell_r/2}\epsilon_r$, holding the sampled noise fixed during differentiation gives

$$\frac{\partial z_r}{\partial \mu_r} = 1, \qquad \frac{\partial z_r}{\partial \ell_r} = \frac{1}{2}e^{\ell_r/2}\epsilon_r. \tag{15}$$

The reconstruction gradient can therefore affect both distribution parameters through the decoder. The KL additionally has direct derivatives

$$\frac{\partial D_{\mathrm{KL}}}{\partial \mu_r} = \mu_r, \qquad \frac{\partial D_{\mathrm{KL}}}{\partial \ell_r} = \frac{1}{2}(e^{\ell_r} - 1). \tag{16}$$

For the KL alone, its minimum is at $\mu = 0$ and $\ell = 0$, corresponding to unit variance. Training balances that preference against reconstruction. The balance does not make a latent coordinate a labeled contact mode or failure flag.

3.5.3 Masked reconstruction at episode boundaries

Let $\hat{a}_{b,j,d}$ and $\tilde{a}_{b,j,d}$ be predicted and teacher actions in normalized coordinates. Here b indexes a batch, j a future time offset, and d an action component. Define $m_{b,j} = 1$ for a genuine target and zero for padding. With $D = 6$, the implemented reconstruction loss is

$$L_{\mathrm{L1}} = \frac{\sum_{b=1}^{B} \sum_{j=0}^{k-1} m_{b,j} \sum_{d=1}^{D} |\hat{a}_{b,j,d} - \tilde{a}_{b,j,d}|}{D \sum_{b=1}^{B} \sum_{j=0}^{k-1} m_{b,j}}. \tag{17}$$

The code clamps the denominator to at least one for numerical protection. With ordinary valid samples, the denominator is the number of valid scalar action elements. It is not the number of all padded-and-unpadded elements.

For a constructed example with one sample, four output rows, and only two valid rows, there are $2 \times 6 = 12$ valid scalar elements. If their total absolute error is 2.4, the L1 loss is 0.2. Dividing by all $4 \times 6 = 24$ elements would incorrectly give 0.1. The same real errors would appear better merely because more padding had been added. The component-level example below makes this denominator explicit.

Normalization changes how physical deviations contribute. An error of one degree in a joint with a small demonstrated standard deviation is numerically different from one degree in a widely varying joint. Gripper error is also normalized on its own scale. A value such as 0.051 is therefore a dimensionless normalized fitting statistic, not 0.051 degrees and not a 5.1% task failure rate.

Consider a constructed sample with $k = 4$ and $D = 6$, where only the first two future rows are valid. Suppose their componentwise absolute errors are

$$E = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.1 & 0.2 & 0.3 \\ 0.3 & 0.2 & 0.1 & 0.3 & 0.2 & 0.1 \end{bmatrix}. \tag{18}$$

Their sum is 2.4, the valid element count is 12, and the loss is 0.2. The errors in the two padded rows contribute zero after masking. Dividing by 24 would halve the loss despite no change in genuine prediction quality.

With several samples, Equation (17) divides the summed valid errors across the entire batch by the summed valid scalar count. A sample with fewer valid rows contributes fewer scalar terms. This is distinct from averaging an equal-weight per-episode or per-sample MAE. The valid-element reduction determines the loss scale used in this comparison.

The posterior encoder also has a padding mask, but its summary and state tokens remain valid. Masking reconstruction alone would not prevent a latent encoder from attending to artificial tail entries. Both uses should be checked when reproducing the implementation.

3.5.4 KL regularization and the combined objective

For a diagonal Gaussian posterior, the divergence from a standard normal is

$$L_{\mathrm{KL}} = \frac{1}{B} \sum_{b=1}^{B} \frac{1}{2} \sum_{r=1}^{d_z} \left(\mu_{b,r}^2 + \exp(\ell_{b,r}) - 1 - \ell_{b,r} \right), \tag{19}$$

with $d_z = 32$. The installed implementation sums latent dimensions and averages batch samples. The run uses

$$L = L_{\mathrm{L1}} + \beta L_{\mathrm{KL}}, \qquad \beta = 10. \tag{20}$$

The two terms measure different things. L1 measures reconstruction disagreement. KL measures posterior/prior difference. Their weighted sum can decrease because either term decreases, so total loss alone does not isolate action fitting.

The variational interpretation connects a reconstruction likelihood and a KL penalty through an evidence lower bound. The practical mean-L1 loss with a chosen KL weight is a surrogate training objective; its reported numeric value is not automatically a calibrated negative log likelihood. The evidence-bound derivation above distinguishes the variational interpretation from the implemented reduction and weighting. The ACT preprint also contains a difference between the loss named in its algorithm box and its implementation discussion; the equations above follow the local source snapshot rather than assuming pseudocode fixes the reduction (Zhao et al., 2023).

For one dimension, $q(z) = \mathcal{N}(\mu, \sigma^2)$ and $p(z) = \mathcal{N}(0, 1)$. By definition,

$$D_{\mathrm{KL}}(q\|p) = \mathbb{E}_q[\log q(z) - \log p(z)]. \tag{21}$$

Substituting Gaussian log densities gives

$$\log q(z) - \log p(z) = -\log \sigma - \frac{(z-\mu)^2}{2\sigma^2} + \frac{z^2}{2}. \tag{22}$$

Using $\mathbb{E}[(z-\mu)^2] = \sigma^2$ and $\mathbb{E}[z^2] = \mu^2 + \sigma^2$ yields

$$D_{\mathrm{KL}}(q\|p) = \frac{1}{2}(\mu^2 + \sigma^2 - 1 - \log \sigma^2). \tag{23}$$

Independent dimensions add, and the installed loss averages examples after summing dimensions. With $\ell = \log \sigma^2$, replace σ^2 by e^ℓ to obtain Equation (19). The sign here is positive because the objective is minimized; some ELBO derivations write the negative KL because the bound is maximized.

A constructed two-dimensional posterior illustrates the scale. For $(\mu_1, \sigma_1) = (0.5, 1)$, the contribution is 0.125. For $(\mu_2, \sigma_2) = (0, 0.5)$, it is approximately 0.31815. The summed KL is approximately 0.44315. If a constructed L1 were 0.2 and $\beta = 10$, the combined loss would be approximately 4.6315. These are constructed example values, not posterior measurements from the experiment.

3.5.5 Deployment at the prior center

At inference, the model fixes $z = 0$. This is deterministic decoding at the mean of the standard-normal prior. It is not generally the mean of the output trajectories, because for a nonlinear decoder

$$f_\theta(o, \mathbb{E}[z]) \neq \mathbb{E}_z[f_\theta(o, z)]. \tag{24}$$

It is also not evidence that the resulting sequence is the most likely complete trajectory. The experiment did not sample latent values or evaluate output diversity.

The last retained KL values print as 0.000. Their three-decimal representation does not establish an exact zero. Nor does it by itself diagnose posterior collapse, meaning that the output has become insensitive to the latent. That diagnosis would require additional posterior and latent-sensitivity measurements. No such inference experiment is claimed here. For this snack task, the posterior has access to the demonstrated close–lift–transport sequence during training; deployment has no future teacher actions. Fixing the 32-dimensional latent to zero selects the prior center before the learned latent projection. Its affine bias means that the resulting 512-dimensional latent token need not be zero. No latent dimension was labeled as a grasp mode or a recovery state.

3.6 Action chunking and feedback

3.6.1 Prediction horizon and executed prefix

Let k denote the number of predicted targets and h the number taken from each block before another query. A query at time t returns $\hat{A}_t$, but the execution wrapper may use only rows $0, \ldots, h-1$. The distinction is also useful for other action-sequence policies (Chi et al., 2023).

In the constructed example of Figure 5, $k = 4$ and $h = 2$. At $t = 0$, the block covers target steps 0, 1, 2, and 3. Targets 0 and 1 are executed. At $t = 2$, a new observation produces targets 2, 3, 4, and 5. The old predictions for 2 and 3 are discarded. Every control tick still receives just one six-dimensional vector.

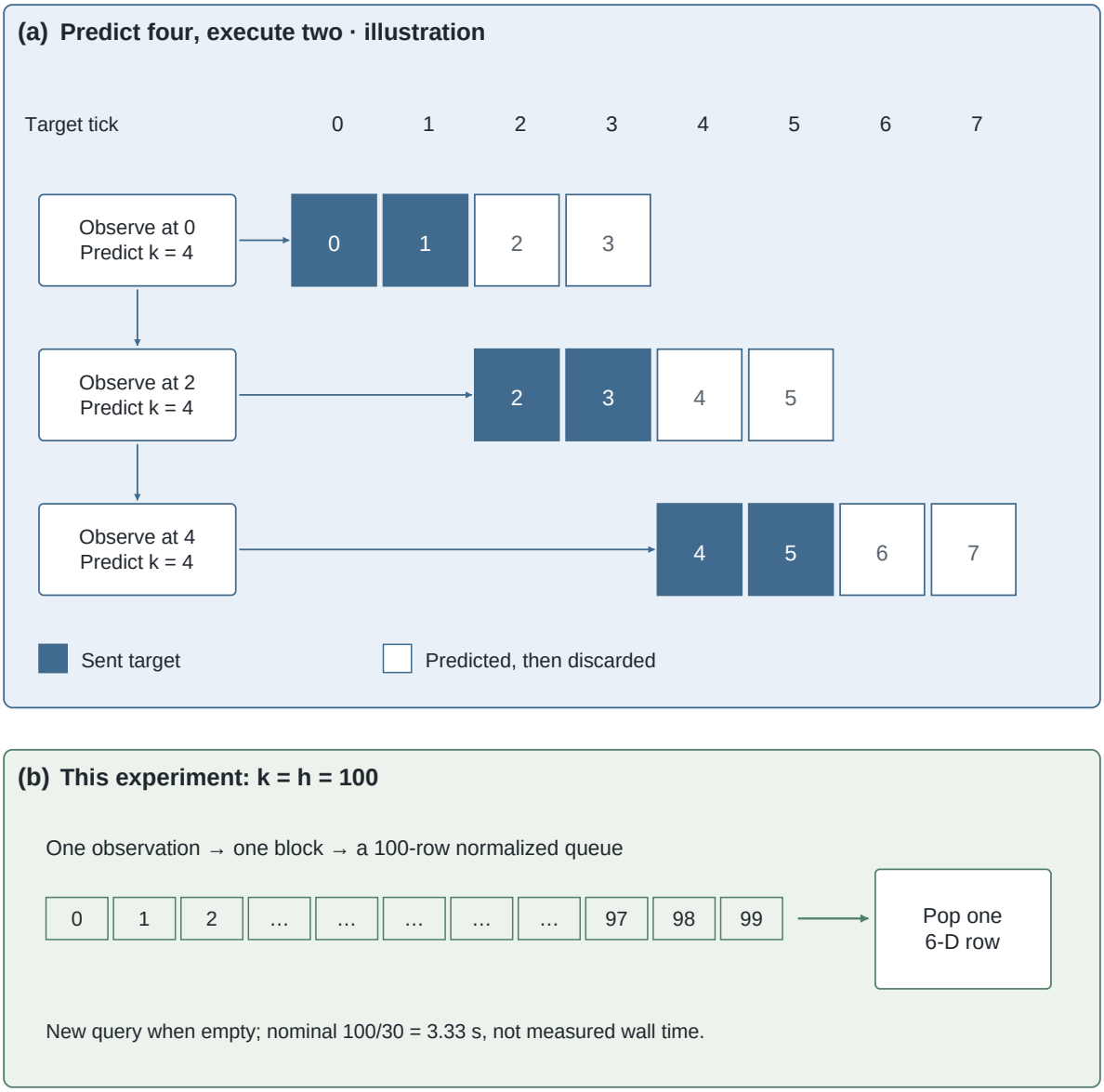


Figure 5: Constructed $k = 4, h = 2$ example. Numbers are target control steps, not measured joint values. Dark cells are executed; pale cells are predicted but discarded when the next observation is used.

This example is not temporal ensembling. It uses a queue of a prefix from each block. A smaller h requests new predictions more often; it does not shorten the network's output unless k is also changed. Changing k changes output queries and the training target structure, normally

requiring a new trained configuration. Changing a compatible execution prefix is a deployment change and should be evaluated separately.

3.6.2 The actual 100-step queue

The saved configuration has `chunk_size=100`, `n_action_steps=100`, and `temporal_ensembl e_coeff=None`. The policy's `select_action` method fills an empty queue with a block and returns one queued target per call. It requests another block when the queue is empty. Current images may still be obtained by the surrounding loop while the queue is nonempty; those images do not cause a fresh ACT prediction in this mode.

At a target control frequency $f = 30$ Hz, the nominal time represented by an executed queue is

$$T_{\text{queue}} = h/f = 100/30 \approx 3.33 \text{ s}. \tag{25}$$

The first and last target sample times differ by $(h-1)/f = 3.30$ s. Both quantities are useful when interpreting a plotted block. Actual wall-clock query spacing can differ because of observation, inference, communication, and scheduling delays.

The actual rollout selected synchronous inference. A query and its resulting actions are part of the same synchronous loop; no asynchronous controller was evaluated. A short initial iteration can run slowly because of initialization or other work, but the available warning does not isolate that cause. The report therefore distinguishes configured frequency from measured sustained timing.

3.6.3 Temporal ensembling

Temporal ensembling is a separate strategy: obtain overlapping predictions and combine those aimed at the same control time. Let p be a query time and $\hat{a}_{p\to t}$ its prediction for target time t. If $\mathcal{P}_t$ is the set of available queries covering t, then

$$\bar{a}_t = \frac{\sum_{p\in\mathcal{P}_t} w_{p,t}\hat{a}_{p\to t}}{\sum_{p\in\mathcal{P}_t} w_{p,t}}. \tag{26}$$

The result remains one action vector. Averaging actions for different target times would blend different phases of motion rather than combine competing estimates of one target (Zhao et al., 2023).

The inspected exponential convention enumerates participating predictions from oldest to newest and assigns $w_i = \exp(-mi)$ with $i = 0$ for the oldest. Under that indexing, positive m gives the oldest prediction the largest weight. An "age" index counted in the opposite direction would require a different interpretation. Figure 6 makes this ordering explicit with constructed values.

In this LeRobot implementation, enabling temporal ensembling requires `n_action_steps=1`. It cannot be described as a switch that preserves the same 100-step queue behavior. More frequent inference can improve access to recent observations, but it also changes timing demands. This experiment does not measure the resulting grasp outcome or latency, so no performance improvement is assigned to the untested alternative.

3.6.4 Three forms of feedback

The motor controller tracks joint targets using its own readings. The policy updates action predictions at query boundaries. The operator supervises the experimental procedure and may

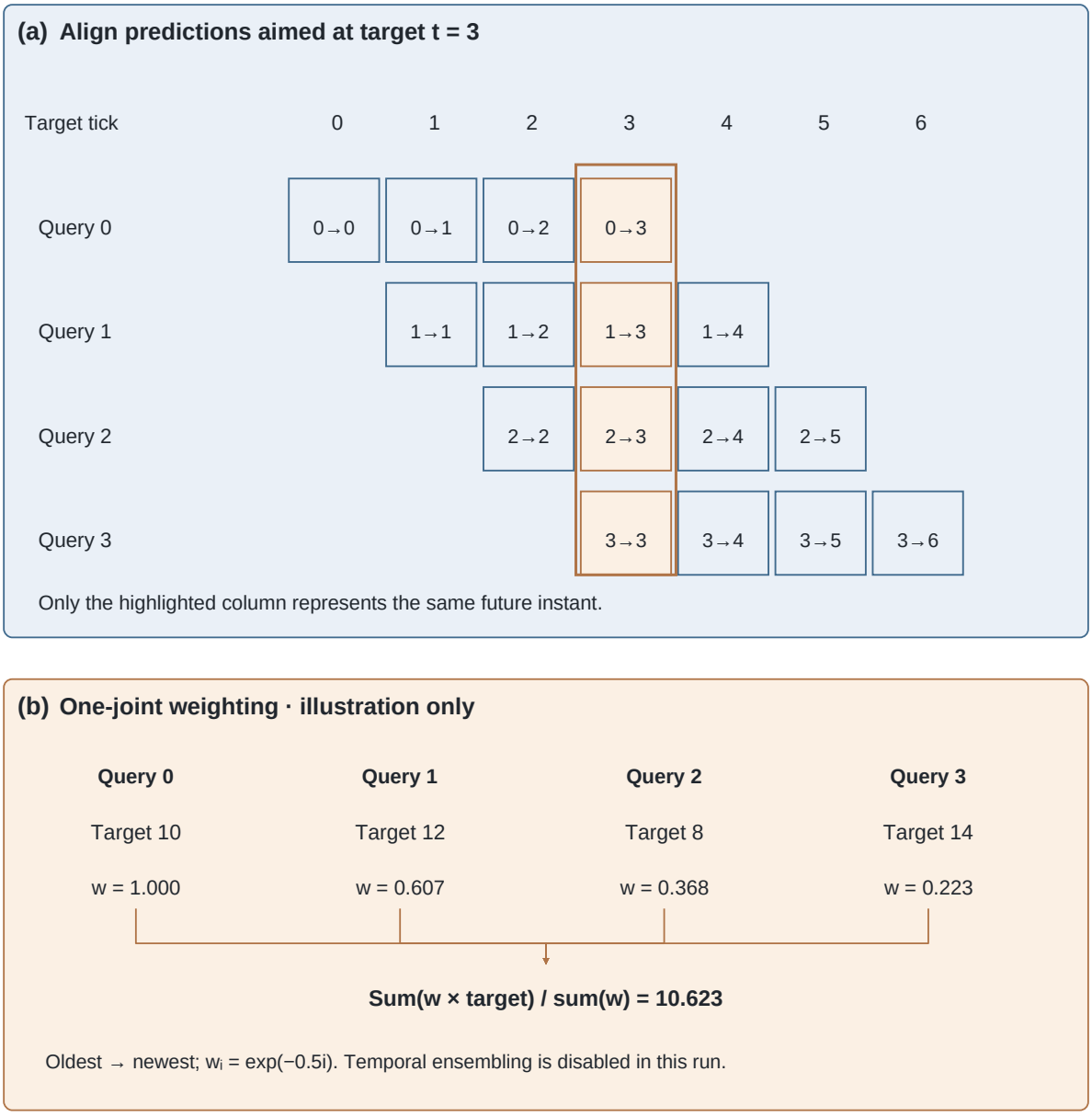


Figure 6: Constructed temporal-ensemble calculation for one joint and one target time. The coefficient 0.5 and targets are constructed inputs, not settings or outputs from this run. The actual run has temporal ensembling disabled.

stop a process. These are different feedback levels. A system can have continuous low-level position feedback while its visual policy holds a predicted block for many ticks.

This distinction prevents two common interpretations: camera frames arriving at 30 Hz do not prove 30 policy replans per second, and successful motor tracking does not prove that the selected grasp is correct. A future timing study should record observation capture, policy-query start/end, target publication, and joint readings on a common clock.

3.6.5 Same-target ensemble calculation

For a constructed $k = 4$ policy queried at every control step, target time $t = 3$ may have predictions from queries $p = 0, 1, 2, 3$. They are rows 3, 2, 1, and 0 of their respective blocks. Only those rows refer to the same target time and belong in Equation (26).

If one joint's four predictions are $(10, 12, 8, 14)$ in oldest-to-newest order, equal weighting gives 11. With coefficient $m = 0.5$, the unnormalized weights are $(1, e^{-0.5}, e^{-1}, e^{-1.5})$. Figure 6 computes the resulting weighted target from these values. The oldest prediction receives the largest weight under that indexing. At startup, fewer overlapping predictions are available and normalization uses the available set.

In the queue-prefix example with $h = 2$, queries occur at $p = 0, 2, 4, \ldots$ instead. It should not be drawn as if predictions were generated at every tick. Query scheduling determines which predictions exist before any weighting is applied.

3.7 Evaluation quantities and diagnostic models

The task outcome is whether the package is transported into the basket and released. An action-error statistic measures agreement with a teacher trajectory. A gripper target can be close on average while missing the package during the short contact interval. Repeated close-and-lift cycles within one execution also do not create independent robot trials. These distinctions determine how training summaries and the two recordings are interpreted.

3.7.1 Offline prediction and simplified state feedback

The offline evaluator feeds recorded observations to a saved policy and compares predictions with demonstrated targets. It does not connect to the robot, retrain the model, or detect whether a real bag was grasped. Table 4 separates its three functions. No offline result JSON was produced for this report, so the explanations below describe the tool rather than completed measurements.

Table 4: Offline evaluator functions and interpretation.

Check	Computation	Useful interpretation
Open-loop prediction	Predict a block from recorded images and measured state; compare each future target with the demonstration.	Fitted target error by joint and prediction horizon.
Simplified state feedback	Move a virtual joint state toward predicted targets; reuse the recorded images.	Sensitivity under this artificial feedback model.
Range/extrapolation	Compare outputs with supplied bounds and observed demonstration ranges.	Numeric range flags, not collision or task success.

Figure 7 separates the recorded-input prediction check, the evaluator's virtual-state feedback, and the physical feedback path in which robot motion changes the camera images.

The field `open_loop.per_joint_mean_abs` is the most interpretable primary error summary. Each of the first five entries is in degrees after postprocessing; the gripper entry is in 0–100 scale points. A mean over all six joints combines unlike units. It must not be reported as one angle-valued error or compared numerically with the normalized training L1.

The `open_loop.by_horizon_curve` describes error at future offsets within predicted blocks. Near episode ends, fewer genuine targets are available. Comparing checkpoints requires consistent episodes, horizons, preprocessing, and execution settings; changing the scoring windows can change the result independently of model quality.

The JSON group named `closed_loop` implements a virtual update

$$q_{t+1}^{\text{sim}} = q_t^{\text{sim}} + 0.35\left(\hat{a}_t - q_t^{\text{sim}}\right). \tag{27}$$

The factor 0.35 is an evaluator assumption, not an identified SO-101 dynamic parameter. The images remain those of the operator's recorded trajectory, even when the virtual joints depart from it. The resulting image/state pair may be physically inconsistent. This is not a closed-loop visual simulator.

The script's divergence flag compares the last and first quarters of action-target error using a factor-of-three heuristic. It does not observe a falling bag, collision, or successful release. Episodes no longer than one executed chunk are separately flagged because the check contains little or no new policy feedback after the first query. These flags are diagnostics, not acceptance thresholds validated for this robot.

3.7.2 Numerical bounds and units

In `degrees` mode without `--model-dir`, the five body-joint bounds are unbounded in this script and only the gripper's 0–100 range is checked. Zero out-of-bounds predictions would therefore not establish legal body-joint motion. Supplying an appropriate model directory allows the script to read `so101_new_calib.urdf` and convert its angular limits. The model must match the intended joint names and conventions; its numeric limits still do not check basket collisions or contact safety.

The alternate `percent` mode assumes body joints on a -100 to 100 scale and gripper on a 0–100 scale. It is not appropriate merely because one output component is called a percentage. The current dataset uses degrees for the five body joints.

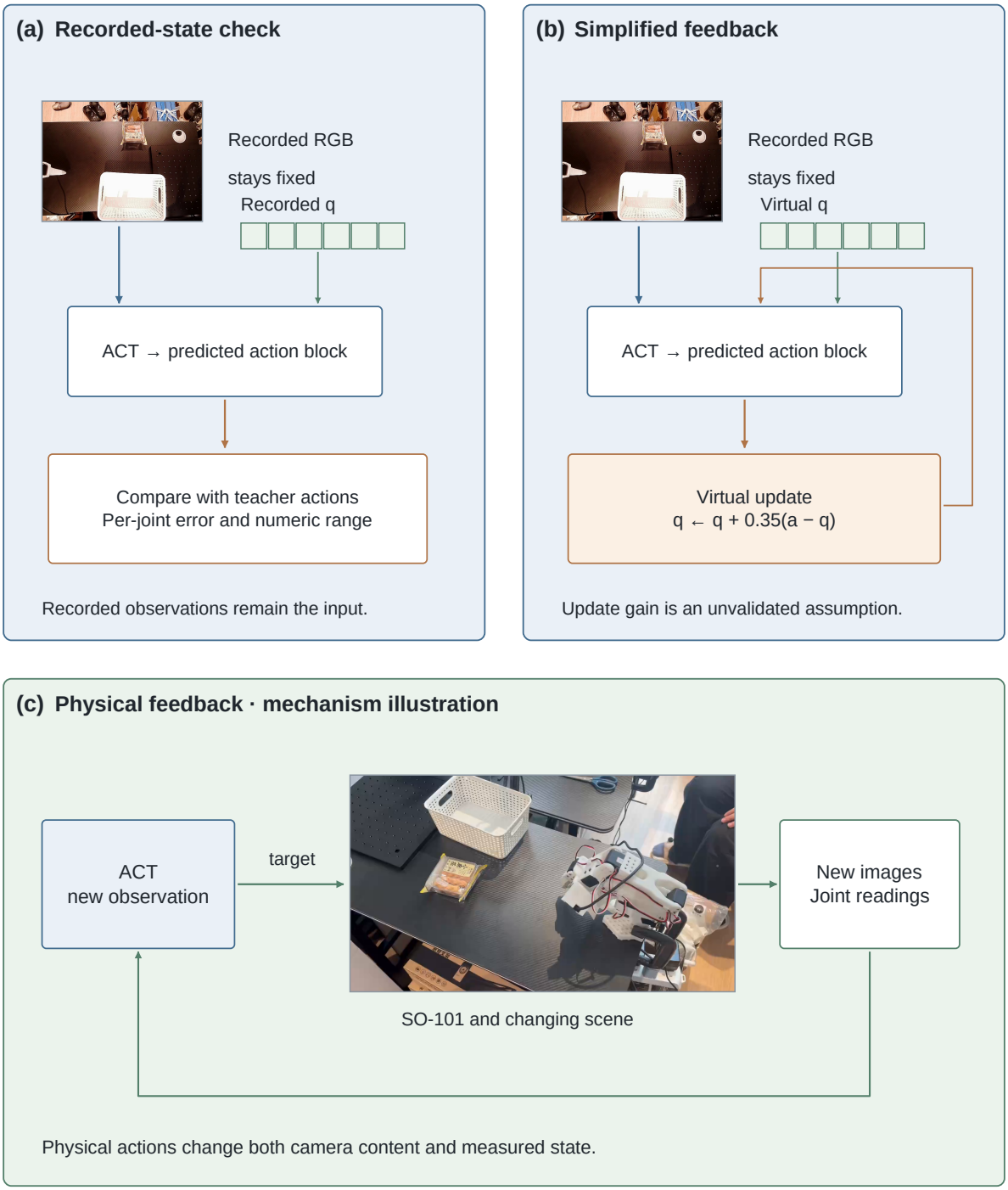


Figure 7: Offline checking and physical feedback. The recorded-image stream stays fixed when the evaluator updates virtual state using gain 0.35. A physical robot instead changes both joint readings and images. The lower panel is a mechanism illustration using a trial photograph; it does not report an additional test.

4 ACT Network Architecture

The configured network has two encoders with different roles: a training-only posterior encoder and a visual/state policy encoder. The decoder generates all 100 action positions in parallel. Figure 8 connects those components to preprocessing, the loss, and deployment. Figures 9 and 10 expand visual tokenization and the posterior objective. Figure 11 resolves the attention and residual structure. The four architecture plates describe complementary parts of the same saved network.

4.1 Visual backbone and input projections

The backbone was initialized from `ResNet18_Weights.IMAGENET1K_V1`, rather than from a pretrained complete robot policy. The same backbone parameters process both cameras. Convolutional parameters participate in optimization, while the installed backbone uses frozen batch-normalization affine values and statistics. Frozen normalization statistics do not mean that the complete backbone is frozen. The distinction matters when reproducing fine-tuning (PyTorch Contributors, n.d.).

With no final-stride dilation and 480×640 inputs, the architecture's stride 32 implies a 15×20 grid of 512-channel features for each view. A 1×1 projection maps features to the Transformer hidden size, also 512. Each spatial location becomes one *token*, a feature vector handled by attention. Here "token" does not mean a word or a discrete language identifier.

4.2 Tensor shapes and positions

Figure 8 and Table 5 make the shape changes explicit. The two views contribute $2(15)(20) = 600$ visual tokens. A projected state token and a projected latent token bring the policy encoder input to 602 tokens per sample. These spatial sizes are derived from the saved architecture, not measured by running a new model forward pass for this report.

Table 5: Shapes in batch-first notation, except where the installed attention code transposes sequence and batch. $B = 8$ in training and normally $B = 1$ in deployment.

Quantity	Shape	Meaning
Each camera tensor	$B \times 3 \times 480 \times 640$	Current RGB observation
State	$B \times 6$	Measured follower positions
Teacher block	$B \times 100 \times 6$	Normalized future targets
Padding mask	$B \times 100$	True for padded target time steps
Each image feature map	$B \times 512 \times 15 \times 20$	Architecture-derived grid
Policy encoder sequence	$B \times 602 \times 512$	600 visual tokens, state, latent
Posterior encoder sequence	$B \times 102 \times 512$	Summary token, state, 100 targets
Posterior mean / log variance	$B \times 32$ each	Diagonal Gaussian parameters
Decoder representation	$B \times 100 \times 512$	Ordered future positions
Policy action prediction	$B \times 100 \times 6$	One six-vector for each future step

The two-dimensional sinusoidal positions locate image features in their own image grids. Separate learned positions distinguish nonvisual state and latent tokens. The saved implementation appends the camera feature sequences in their configured order; this description does not introduce an additional camera-ID embedding that is absent from the inspected code. The posterior uses a learned CLS content vector and fixed one-dimensional sinusoidal positions for its 102-token input. Its first two tokens are always valid; the remaining mask entries correspond to

action padding. This is a key padding mask, not a causal triangular mask. The CLS output passes through a $512 \to 64$ affine head, whose halves are the mean and log variance.

The visual backbone and 1×1 projection are shared across top and wrist. Both image grids reuse the same two-dimensional coordinate encoding, without a separate camera-ID embedding. The actual token order is latent, state, top features, then wrist features. Frozen batch normalization retains its affine values and running statistics as buffers; convolutional parameters remain in the optimized backbone parameter group.

4.3 Encoder, decoder, and regression head

The saved policy uses four observation-encoder layers and one decoder layer, with eight attention heads, hidden size 512, feed-forward size 3200, ReLU activation, and dropout 0.1. Feed-forward layers transform each token after attention, while residual paths and normalization support optimization. The training-only posterior encoder uses four layers of its own. Thus "encoder" must be qualified: the posterior encoder and the observation-fusion encoder have different inputs and roles.

The decoder starts with zero content vectors and 100 learned positional embeddings. These embeddings identify future output slots. Its self-attention precedes cross-attention. With the saved single decoder layer, observation-dependent contents are introduced by cross-attention and do not pass through a later decoder self-attention layer. The decoder produces all 100 representations in one forward pass, and a linear head maps each 512-dimensional representation to six action components. It does not first generate one action and then recursively feed that action back to generate the next. ACT therefore differs from the autoregressive translation decoder in the original Transformer (Vaswani et al., 2017; Zhao et al., 2023).

DETR supplies a useful architectural comparison because it also uses parallel learned queries (Carion et al., 2020). Its queries represent an unordered object set and use a matching objective. ACT's slots represent an ordered action sequence; Hungarian matching, detection boxes, and "no object" labels are not part of the inspected ACT loss.

The posterior and policy encoder stacks do not add a final LayerNorm when `pre_norm=false`; their individual layers already contain normalization after each residual addition. The decoder adds a final LayerNorm after its last layer. In its self-attention, Q and K receive content plus action-position embeddings, while V receives content. Cross-attention uses the current decoder content plus action positions for Q, encoder memory plus encoder positions for K, and memory alone for V. The multi-head modules then apply their learned projections. Initial zero content does not imply a guaranteed zero attention output, because trained affine biases can contribute.

4.4 Stored model size

The checkpoint inspection counted 234 stored tensors and 51,668,614 state-dictionary elements, with a model file size of 206,699,736 bytes. State dictionaries can contain buffers as well as learned parameters, so this count is reported as stored elements rather than an independently measured trainable-parameter count. Weight-file size also excludes optimizer state, activations, gradients, and runtime allocations. It cannot be used by itself to predict training GPU memory.

4.5 Original method, implementation, and actual settings

Table 6 separates the method described by Zhao et al. (2023), the inspected LeRobot implementation, and the settings actually used. An available capability is not evidence that it was

enabled. The source snapshots and saved configuration, rather than current software defaults, determine the last two columns.

Table 6: ACT mechanisms across the original study, the inspected implementation, and this experiment. The comparison concerns design and configuration, not relative task performance.

Aspect	Original ACT study	Inspected LeRobot code	Recorded SO-101 run
Platform	Bimanual ALOHA	Input/output dimensions supplied by configuration	One follower; six position targets
Images	Four RGB camera views	Camera feature sequences concatenated by configured order	Two RGB views: top and wrist
Vision	ImageNet-initialized ResNet-18 features	Configurable backbone with frozen batch-normalization affine values and statistics	Shared ResNet-18; convolution weights trainable
Action sequence	Predict a future action chunk	Separate prediction length and executed-prefix settings	$k = 100$, $h = 100$
Latent training	Conditional latent encoder; reconstruction and KL	Posterior from state and teacher actions; masked L1 plus KL	32 latent dimensions; KL weight 10
Deployment latent	Use a fixed zero latent	Bypass the posterior when teacher actions are absent	$z = 0$
Temporal ensemble	Combine overlapping same-time predictions	Optional ensemble; requires $h = 1$ in this version	Disabled; use a 100-target queue
Task conditioning	Visual and proprioceptive observations	No language encoder in the inspected ACT class	Task sentence retained as metadata

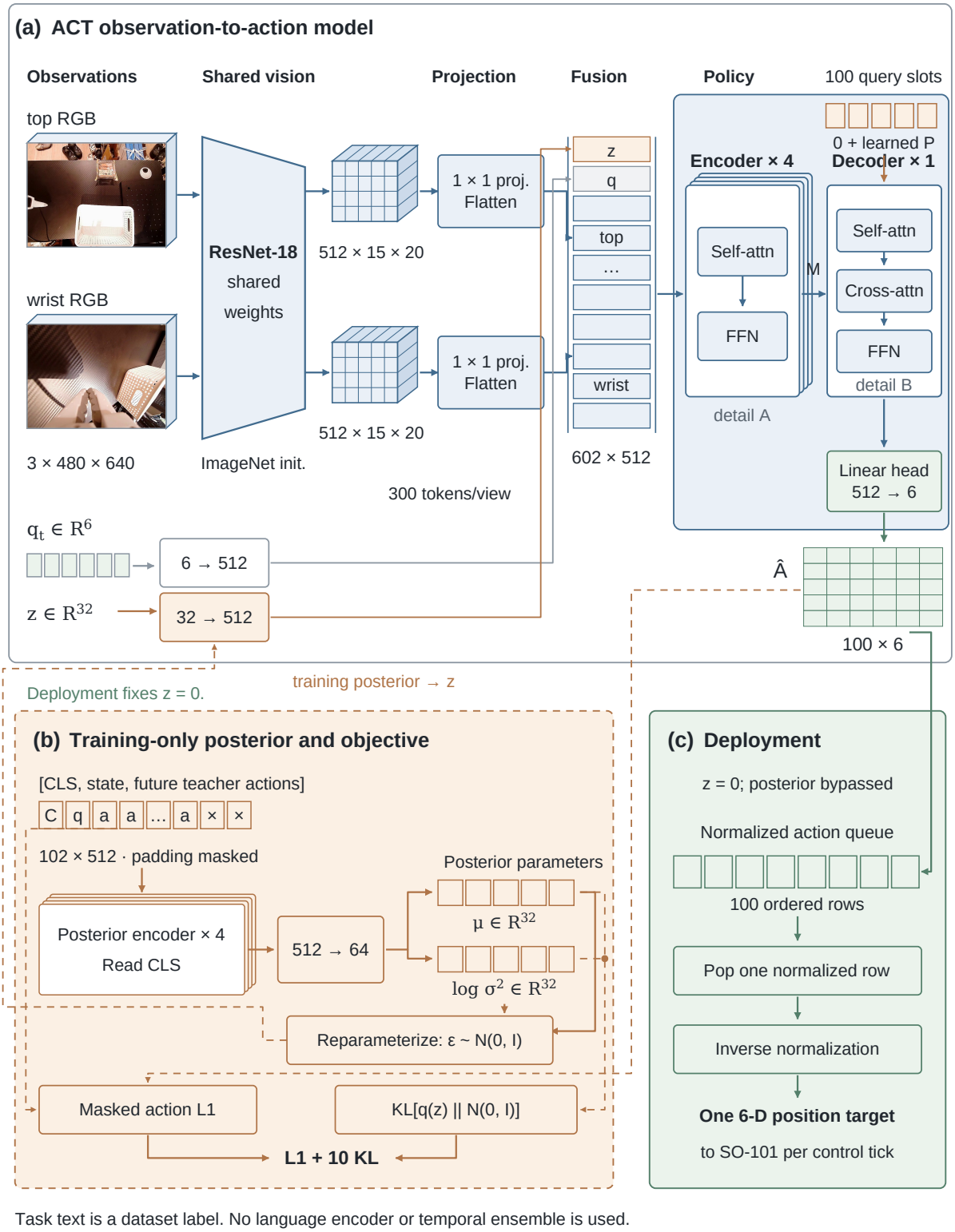


Figure 8: Complete ACT data flow in the recorded configuration. Real top and wrist examples feed a shared backbone; latent, measured state, and visual tokens enter the policy encoder. The decoder predicts an ordered block, which supports training reconstruction and queued deployment. The posterior and its KL branch are training-only. Deployment fixes $z = 0$ and pops normalized rows before inverse normalization. Position routes are abbreviated; the detailed Transformer plate explains their Q/K roles. Feature glyphs are symbolic, not measured activations.

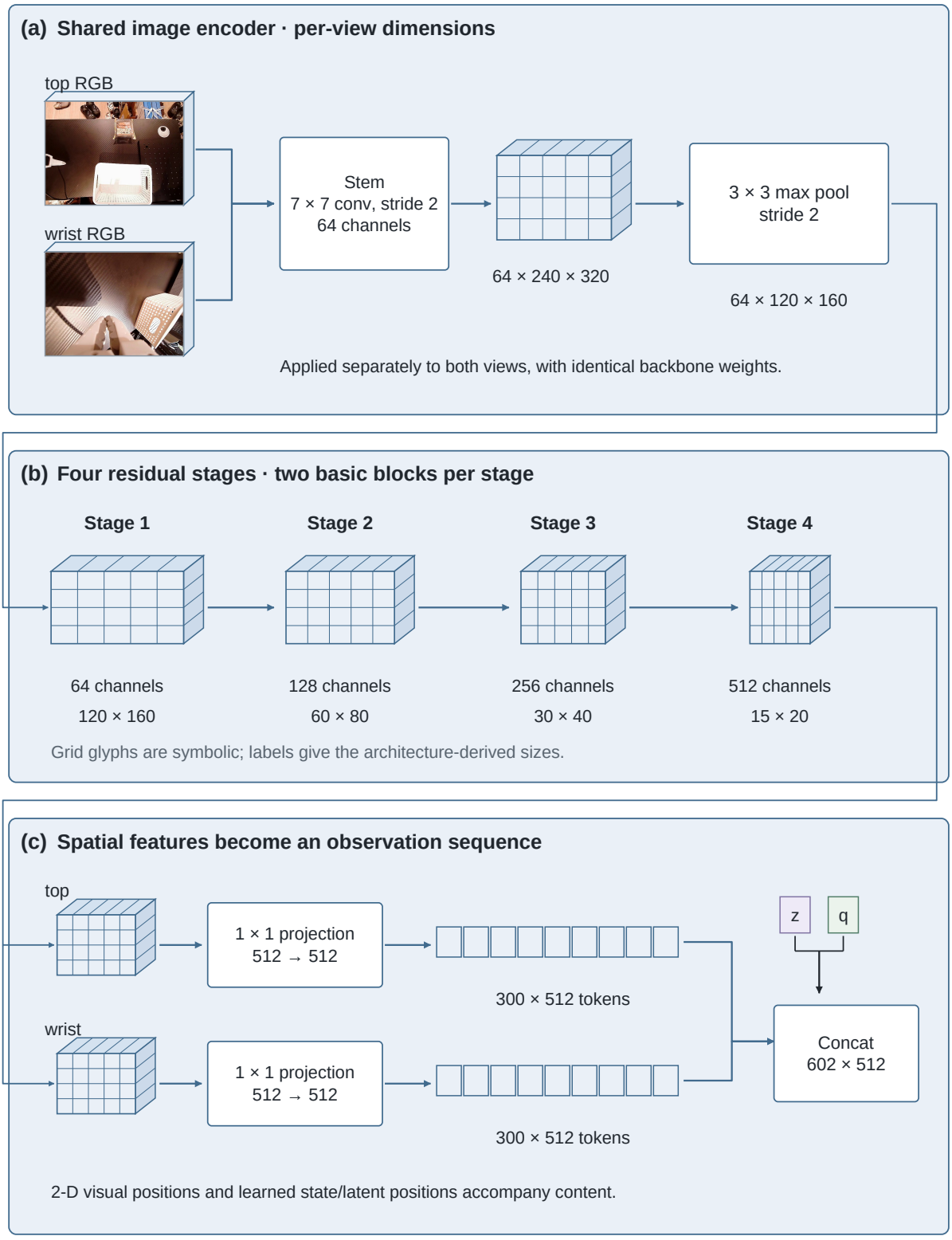


Figure 9: Visual encoding and tokenization. Source-derived spatial sizes follow the undilated ResNet-18 stem and four residual stages. Each camera contributes 300 projected spatial tokens; latent and state add two nonvisual tokens. The feature stacks illustrate tensor structure, not observed feature values. No new model forward pass was used to obtain these shapes.

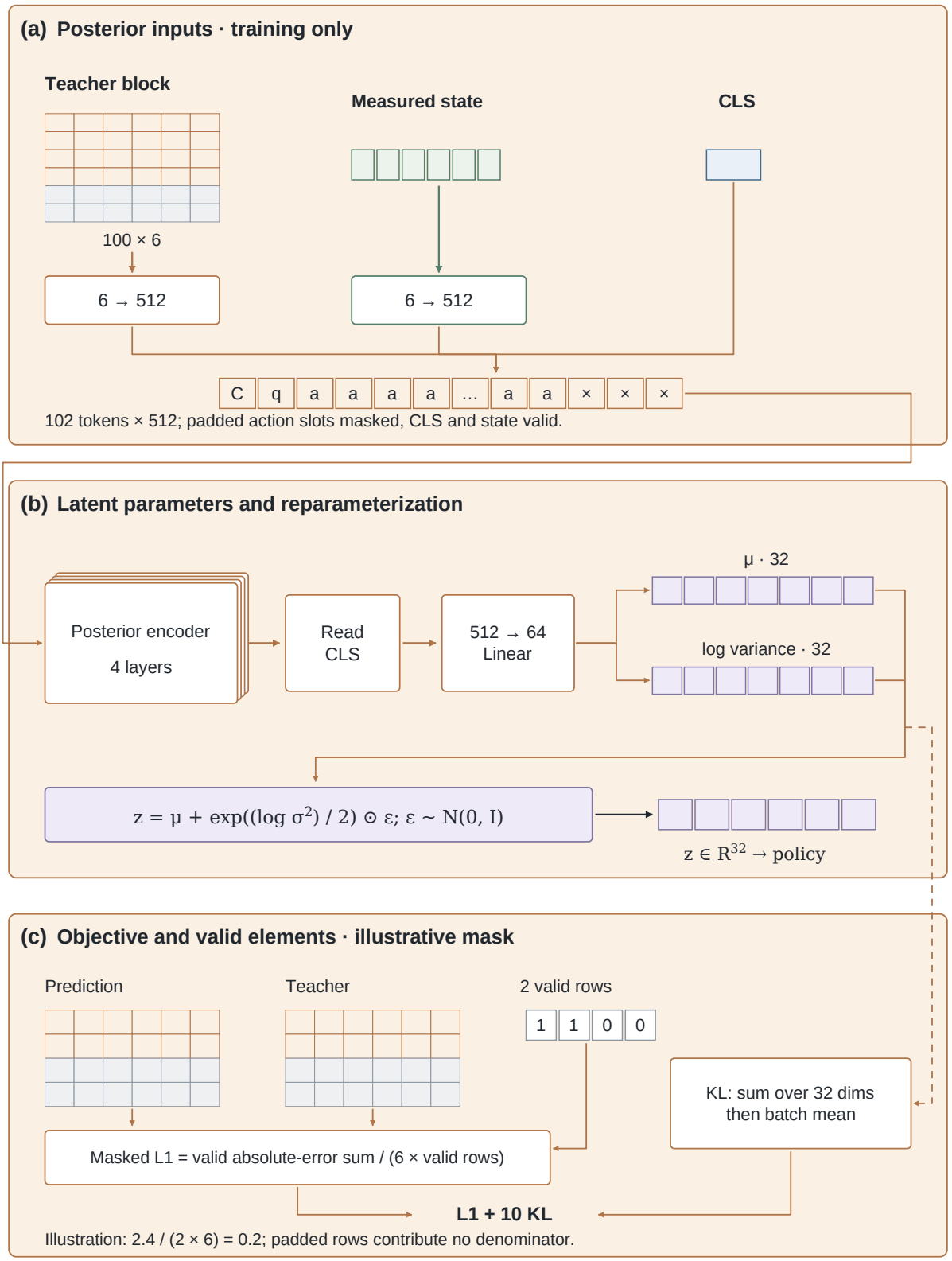


Figure 10: Training posterior and objective. The 102-token posterior sequence contains CLS, current state, and 100 teacher actions. Mean and log variance define a 32-dimensional latent. The lower four-row example is constructed: two valid six-dimensional rows and error sum 2.4 give masked L1 of 0.2. KL instead sums latent dimensions and averages the batch.

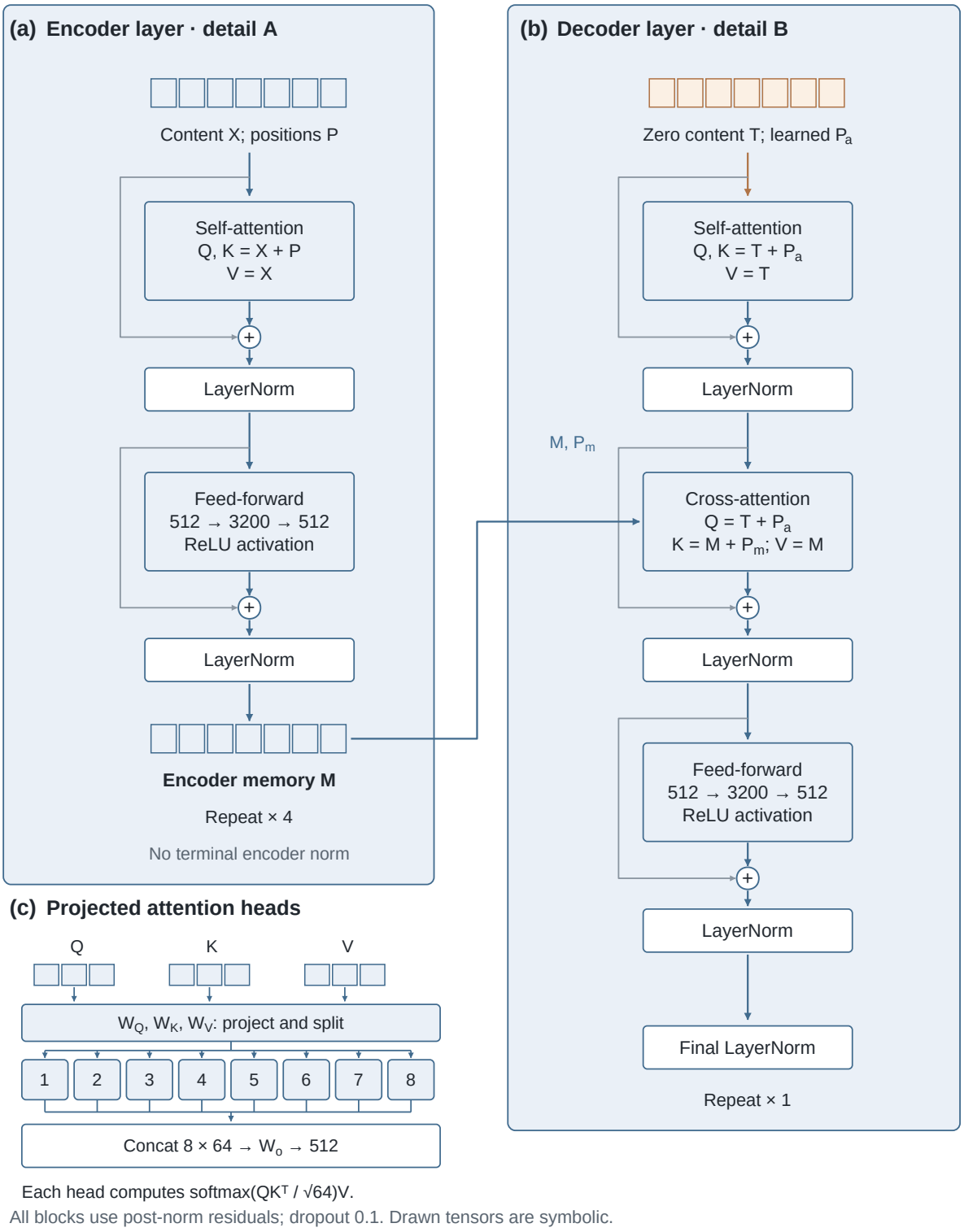


Figure 11: Post-norm Transformer structure. Encoder and decoder residual branches, Q/K/V sources, feed-forward dimensions, and the decoder's terminal LayerNorm are explicit. Encoder memory M enters decoder cross-attention. The lower schematic represents eight projected attention heads; it does not display learned attention scores.

5 Demonstrations and Dataset

5.1 Dataset size and version identity

Table 7 reports the frozen training copy. The original directory is `datasets/exp1_snack_into_box`; the training copy is `datasets/exp1_snack_into_box_v1`. A prior file-by-file SHA-256 comparison found all 27 core files under `data`, `meta`, and `videos` identical between them. The suffix `v1` therefore identifies a complete copy, not a curated subset.

Table 7: Dataset measurements from the archived structure inspection. Duration is computed from saved frames and nominal frame rate.

Quantity	Value
Episodes / tasks	90 (indices 0–89) / 1
Saved time steps	31,968
Nominal sampling rate	30 Hz
Saved duration	1,065.60 s (17 min 45.60 s)
Episode duration: mean / median	11.84 s / 11.37 s
Episode duration: minimum / maximum	9.33 s / 25.80 s
RGB inputs	Two streams, each $640 \times 480 \times 3$
Joint state / action target	Six float32 components each
Storage	LeRobotDataset v3.0; Parquet, MP4, and metadata
Dataset video codec	AV1, yuv420p, no audio; recorded settings CRF 30, GOP 2
Core storage size	1,700,479,885 bytes; 1,621.7 MiB (about 1.58 GiB)

The data-inspection utility printed "1621.7 MB," but its conversion used 1024^2 bytes. MiB is used here to state that conversion precisely. Saved duration excludes placement, reset, and recording overhead; it is not the operator's total collection time. Likewise, two camera streams do not double the number of aligned training time steps.

5.2 File batches and spatial coverage

The dataset is organized into six blocks of 15 episodes. Table 8 and Figure 12 separate this storage organization from spatial coverage. The first 15 episodes were explicitly identified by the operator as the arm's upper-left grid cell, A1. The camera's left/right direction is not used to redefine that label.

Table 8: Storage blocks and known location labels. Unknown labels are not inferred from block numbering.

Block	Episodes	Count	Frames	Saved seconds	Grid label
1	0–14	15	5518	183.93	A1
2	15–29	15	5765	192.17	Unknown
3	30–44	15	5553	185.10	Unknown
4	45–59	15	5307	176.90	Unknown
5	60–74	15	4823	160.77	Unknown
6	75–89	15	5002	166.73	Unknown

The course collection plan proposed nine cells with 15 qualified demonstrations per cell, or 135 demonstrations. The operator chose to run an initial training experiment after 90. The

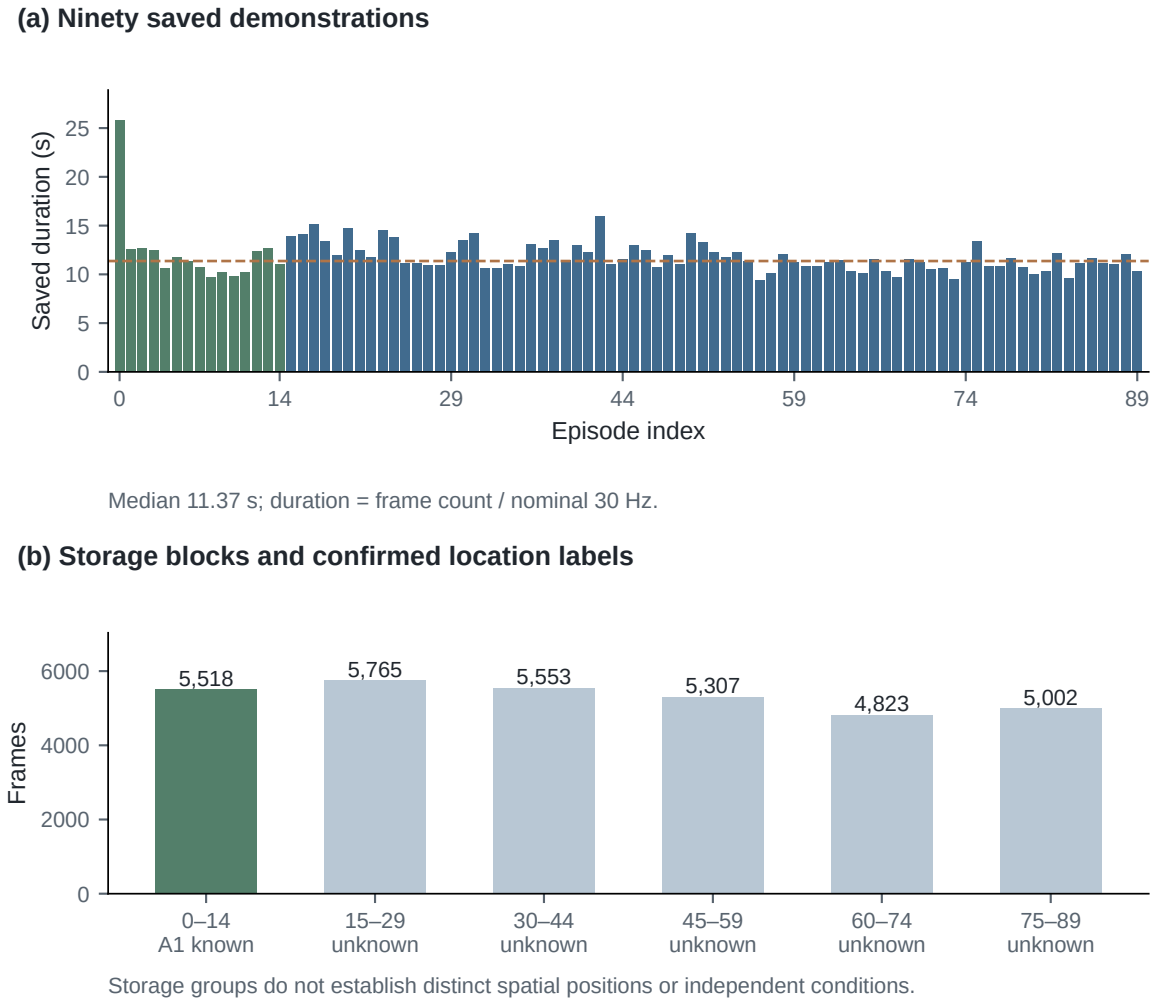


Figure 12: Saved episode durations and frame counts by storage block. The known A1 label covers episodes 0–14 only. Neither the six blocks nor their frame counts establish six different spatial conditions.

completed training and two executions show that this smaller dataset can support the recorded cases; they do not establish that 90 is a sufficient general threshold or that nine-cell coverage was achieved. Demonstration quality and condition coverage matter in addition to trajectory count (Mandlekar et al., 2022).

5.3 Inspection scope

All 31,968 data rows and 90 episode records were checked against the metadata. Global indices were continuous; local frame indices started at zero and matched episode lengths and boundaries. The state and action arrays both had shape $[31968, 6]$ and contained no NaN or infinite values. These checks establish structural consistency, not physical demonstration success.

Container metadata was inspected for all 12 camera videos. Both camera streams for the first 15 episodes were completely decoded, yielding 5,518 frames per stream without a decoding failure. Time-sequence frame sheets of those episodes were reviewed, usually at 0.5–1 s spacing with endpoints. The sampled views showed placement, release, and empty withdrawal. This sampling could miss events shorter than its interval. The remaining 75 episodes did not receive full-frame visual acceptance review.

The saved timestamp matched local frame index divided by 30, with a maximum numerical difference of approximately 8.9×10^{-7} s. Because these timestamps derive from the saved indexing scheme, this agreement does not measure camera exposure timing, camera-to-camera hardware synchronization, or leader-to-follower sample alignment.

5.4 Quality observations retained in training

Table 9 records observations that were not removed from the training copy. These observations suggest later controlled checks; they are not proof of the causes of deployment behavior.

Table 9: Recorded data-quality observations and their interpretation. All affected data remain in the training copy.

Observation	Record	Implication
Long stationary tail	Episode 0 lasts 25.80 s, with about 12.43 s stationary after return.	Stationary targets occupy more sampling anchors; later duration normalization would change the training distribution.
Orientation changes	Episodes 12 and 13 differ in bag orientation from many nearby demonstrations.	The collected conditions are not strictly a position-only variation.
Tracking offset	Some joint trajectories best matched at a 4–5-frame lag.	About 0.13–0.17 s at nominal rate; not evidence of an image synchronization error.
Adjacent repeated images	Isolated one-interval repeats in the inspected block.	No sustained multi-frame freeze was observed there; every saved frame is not necessarily a new exposure.
Appearance variation	Local motion blur and package reflections.	Relevant visual conditions, without measured causal contribution to grasp failures.

5.5 Data separation and leakage

A random split of frames can place nearly identical neighboring images in training and validation. Overlapping action windows add another leakage path: two anchors may contain many of the same future targets. A future experiment should first divide complete episodes, or groups of related episodes/conditions, and then construct windows separately inside each partition. Normalization statistics should be fitted on the training partition only.

All 90 episodes were used for the current checkpoint. Selecting some of them now does not make them independent held-out examples for that checkpoint. They remain useful for checking fitted predictions, tensor shapes, and preprocessing. An independent validation study requires new unseen data or retraining under a split fixed beforehand. The distinction between training fit and task evaluation is especially important for human demonstration datasets (Mandlekar et al., 2022).

5.6 Recorded example and teacher action block

Figure 13 illustrates one real demonstration. The local time within an episode differs from the presentation timestamp within a larger video file, because a file may store several episodes. The archived frame manifest records both. Figure 14 then shows an actual 100-action target block from the same episode. It is a teacher sequence, not a policy prediction or an offline comparison.

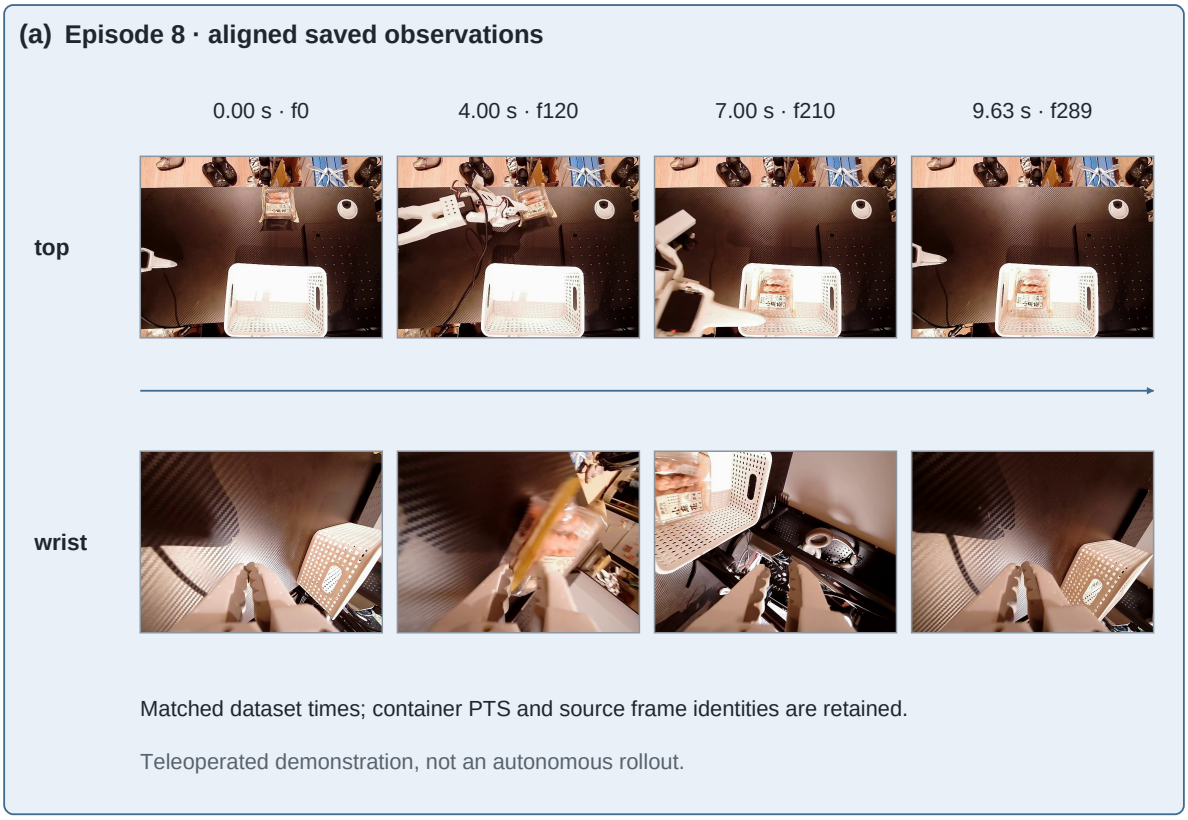


Figure 13: Episode 8 at four local times, shown from the fixed and wrist cameras. These are teleoperated demonstration frames. The archived source manifest retains episode-local frame numbers and container presentation timestamps.

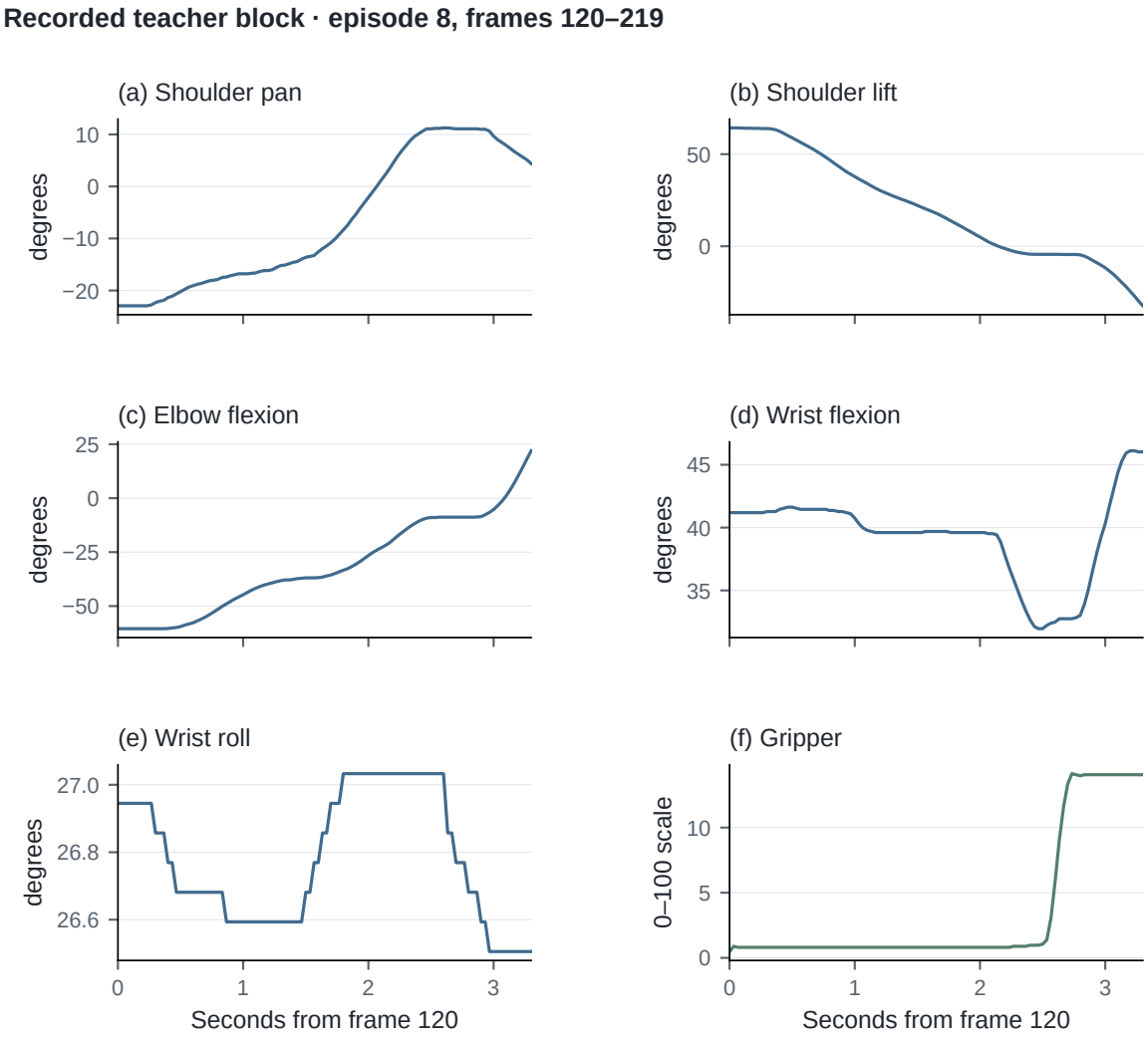


Figure 14: Recorded teacher targets from episode 8, local frames 120–219. The first five components are degrees and the gripper is on a 0–100 scale. One hundred nominal control steps span 3.33 s, while the first-to-last sample-time difference is $99/30 = 3.30$ s.

6 Training and Deployment Protocol

6.1 Software and initialization

LeRobot supplies the dataset, ACT, and training infrastructure (Hugging Face, n.d.). The training process used the project's existing Python 3.12.3 virtual environment and LeRobot 0.6.1. Table 10 lists the principal packages from the captured environment. The complete package list is retained with the original evidence. A package-version list records what was installed; it is not a wheel-hash lock file or a guarantee that another machine will produce identical floating-point results.

Table 10: Principal software versions captured during training. Recording and training use different video components.

Component	Version	Role
Python	3.12.3	Project interpreter
LeRobot	0.6.1	Dataset, ACT, training and rollout code
PyTorch / Torchvision	2.11.0 / 0.26.0	Neural computation and ResNet
PyTorch CUDA build	13.0	CUDA build used by this PyTorch package
Accelerate	1.15.0	Training execution support
Datasets / Hugging Face Hub	4.8.5 / 1.30.0	Dataset and artifact support
Safetensors	0.8.0	Model/processor tensor storage
TorchCodec / PyAV	0.11.1 / 15.1.0	Training decoding / recording video support
NumPy / Pandas / PyArrow	2.2.6 / 2.3.3 / 25.0.1	Numerical arrays and tabular data
OpenCV headless	4.13.0.92	Camera processing
W&B	0.27.2	Installed, but disabled for this run

Running `lerobot-train` does not run `pipinstall`. It imports already installed packages. A missing dependency normally causes an import or startup failure. Loading ImageNet weights is a different operation: Torchvision checks its weight cache and downloads a weight file when needed (PyTorch Contributors, n.d.). That download does not install a new Python package.

The cached ResNet-18 file existed before training: 46,830,571 bytes, with a recorded modification date of 6 August 2026. Its SHA-256 is retained in Appendix B. This supports the availability of the required initialization file, not a complete audit of the process's network traffic. The complete robot policy was trained on the collected demonstrations; it was not downloaded as a pretrained task policy.

The environment inspection found no broken declared package requirements using `pipcheck`. This checks dependency declarations, not camera timing, calibration validity, or manipulation performance. The five relevant implementation files captured during training still matched their recorded hashes when the report was prepared. Local source identities are important because a version string alone does not disclose local changes (Henderson et al., 2018).

6.2 Training configuration

The final checkpoint's saved policy and training configurations confirm Table 11. Values not overridden at the command line came from the installed defaults; they are reported from saved files rather than assumed from current documentation.

The process made $100000 \times 8 = 800000$ sampled anchor presentations, about $800000/31968 = 25.03$ dataset-size equivalents. Each anchor includes a future block, often overlapping blocks

Table 11: Optimization and data-loading settings from the completed run.

Setting	Value	Interpretation
Optimizer updates	100000	Batch updates, not robot trials
Batch size	8	Eight observation anchors per update
Optimizer	AdamW	ACT optimization preset
Learning rate / backbone rate	10^{-5} / 10^{-5}	Both parameter groups optimized
Weight decay	10^{-4}	AdamW regularization
Scheduler	None	No configured learning-rate schedule
Seed	1000	One training seed
Workers / prefetch factor	4 / 4	Data-loader settings
Video backend	TorchCodec	Training-time decoding
Image augmentation	Disabled	No enabled transform augmentation
AMP	Disabled	Not an FP16/BF16 AMP run
TF32	Allowed by code	Distinct from AMP
Validation fraction / interval	0 / 0	No independent validation loss
Automatic environment evaluation	None	No simulator or robot success test in training
Save interval / log interval	20000 / 200	Checkpoint and logging cadence
W&B / automatic Hub push	Disabled	No remote run history or model upload

from nearby anchors. These equivalents do not mean 25 independently collected datasets or 800,000 different robot trials.

The actual command retained the quoted literal `<HF_USER>/exp1_snack_into_box_v1`. Local loading succeeded through the explicit dataset root, so the placeholder did not prevent the recorded optimization. It is nevertheless not a valid demonstrated Hub identity. The original invocation is retained in the companion historical-operations file. A subsequent run must use its actual dataset identity and a fresh output directory.

6.3 Run management and retained artifacts

The recorded run loaded the frozen dataset copy from its explicit local root. Camera feature names, state/action shapes, device selection, batch size, and model settings were available in startup and saved configuration records. The completed run was launched by the operator; report preparation did not launch an additional training process or recreate optimization.

The output retained checkpoints at 20,000-update intervals, each with policy configuration, weights, processors, and training-state artifacts. The final stored step was inspected independently of the abbreviated terminal display. The complete terminal stream was not retained, leaving only the five metric windows discussed in Section 7. W&B and automatic Hub upload were disabled.

No independent validation split, automatic task evaluation, or checkpoint-selection comparison was part of this training protocol. Subsequent validation must use a declared episode/condition split and normalization fitted on the training partition; the existing 90 episodes cannot retrospectively become an independent set for this checkpoint.

6.4 Deployment settings and operator actions

The deployment command selected the base rollout strategy, an SO-101 follower, the saved calibration directory, two configured cameras, synchronous inference, a target 30 Hz, and a 45 s duration. The log stated `Policyloaded:type=act,device=cuda`. This confirms CUDA selection at loading, while no deployment GPU-utilization series was collected.

The base strategy executes the policy; it does not automatically annotate grasp success or create an evaluation dataset. The 45 s window is a configuration limit, not a measured task duration or a success detector. The camera snapshot records the fixed view at 0 degrees rotation and the wrist view at 180 degrees. Matching these conventions is part of reproducing the observation distribution.

The operator is responsible for powering, connecting, calibrating, and positioning the real system. A failed read of a motor register is a communication failure at that step, not a signal to begin recording or executing a policy. The earlier startup attempt failed to read `Torque_Enable` on motor 4; power reconnection and a successful course-command restart were operator actions. The available record does not isolate whether the original cause was a command choice, cabling, timing, or device state. It also does not establish a universal claim about the mechanical or electrical consequences of abrupt power removal.

The rollout shutdown code records a return to the initial position and performs about a 3 s interpolation. This programmed action must be considered when judging the learned return behavior. Neither calibration nor software bounds alone establish collision-free motion around the basket.

6.5 Implementation and reproducibility materials

The project repository is Yanshi-Robotics/Lets-Build-Robots-S1-SO101; the ACT infrastructure is provided by Hugging Face LeRobot. The project supplies recording, rollout, and offline-check utilities. LeRobot supplies the dataset interface, processors, ResNet/Transformer policy, optimizer preset, and checkpoint support. Exact local source identities and configuration records accompany the report because a repository link alone does not identify local changes.

The offline evaluator is `ACT-1-Pick/offline_eval.py`. The inspected local version takes `observation.state` as its state input; the pinned public revision instead used teacher action targets for the open-loop input and virtual-state initialization. These are different evaluation conditions. The local snapshot and version-comparison record in Appendix B identify the checked implementation; the public repository was not modified by report preparation (Yanshi Robotics, 2026).

Historical invocations and implementation examples are retained in `report_assets/final_report/data/former_code_appendices.txt`. These records accompany the configuration tables, algorithm equations, and source identities used to identify the run.

7 Results and Analysis

7.1 Completion evidence and checkpoints

The recorded process began at 00:24:13 and ended at 03:05:40 on 13 October 2026, Beijing time. The difference is 9,687 s, or 2 h 41 min 27 s. The progress bar's training-loop time was 9,684 s; the two measures include slightly different execution scopes. The overall rate is about

100000/9687 = 10.32 optimizer updates per second.

The terminal retained the final step, save message, `Endoftraining`, and a returned shell prompt. The numerical shell exit status was not saved, so it is not reported as an observed exit code of zero. The final training-state file records step 100000, one process, and batch size 8.

All five checkpoint files were present. Stored tensors and processor statistics were inspected on CPU and had no detected nonfinite values. Figure 15 shows the save progression. The local `last` link resolved to the 100000-step directory; its model hash is recorded in Appendix B.

Table 12: Checkpoint model-file times on 13 October 2026, Beijing time. Each model file contains 206,699,736 bytes; the time is a filesystem observation, not a GPU timing measurement.

Step	Model-file modification time	Tensor inspection
20000	00:57:08.783	No nonfinite values detected
40000	01:29:16.710	No nonfinite values detected
60000	02:01:24.626	No nonfinite values detected
80000	02:33:32.734	No nonfinite values detected
100000	03:05:40.835	No nonfinite values detected

The final checkpoint was used as the first deployment candidate because training had completed and the artifact checks passed. No same-condition comparison with earlier checkpoints was performed. “Final” and “best” therefore remain different descriptions. Selecting the maximum observed success among repeatedly tested checkpoints would also need an explicit selection protocol and a separate final test (Mandlekar et al., 2022).

7.2 Retained loss measurements

Table 13 and Figure 16 contain all five retained metric intervals. They are logging summaries, not a fresh evaluation over every training sample. The displayed step abbreviations in the original terminal were resolved using adjacent progress output when needed.

Table 13: Retained training metrics. KL is unweighted; the total uses weight 10. Rounding prevents exact reconstruction of every total from displayed components.

Step	Total	L1	KL	Pre-clip norm	Update time (s)
1400	1.657	0.346	0.131	63.917	0.099
3000	0.786	0.251	0.054	42.641	0.101
5000	0.389	0.204	0.019	28.852	0.101
99800	0.053	0.052	0.000	6.260	0.096
100000	0.052	0.051	0.000	6.162	0.095

The retained L1 decreases from 0.346 to 0.051, approximately an 85.3% reduction relative to the earliest retained interval. This supports improved normalized fitting on the sampled training data. It does not locate the onset of a plateau, prove smooth convergence through the missing interval, or demonstrate absence of overfitting. The two final summaries are close, which describes that short retained period only.

With a log interval of 200, approximately 500 metric summaries could have been produced during 100000 updates. W&B was disabled, full terminal output was not redirected to a persistent file, and the installed logging initialization did not supply a log-file path. The retained snapshots contain five summaries. Checkpoint weights do not encode the missing historical loss values.

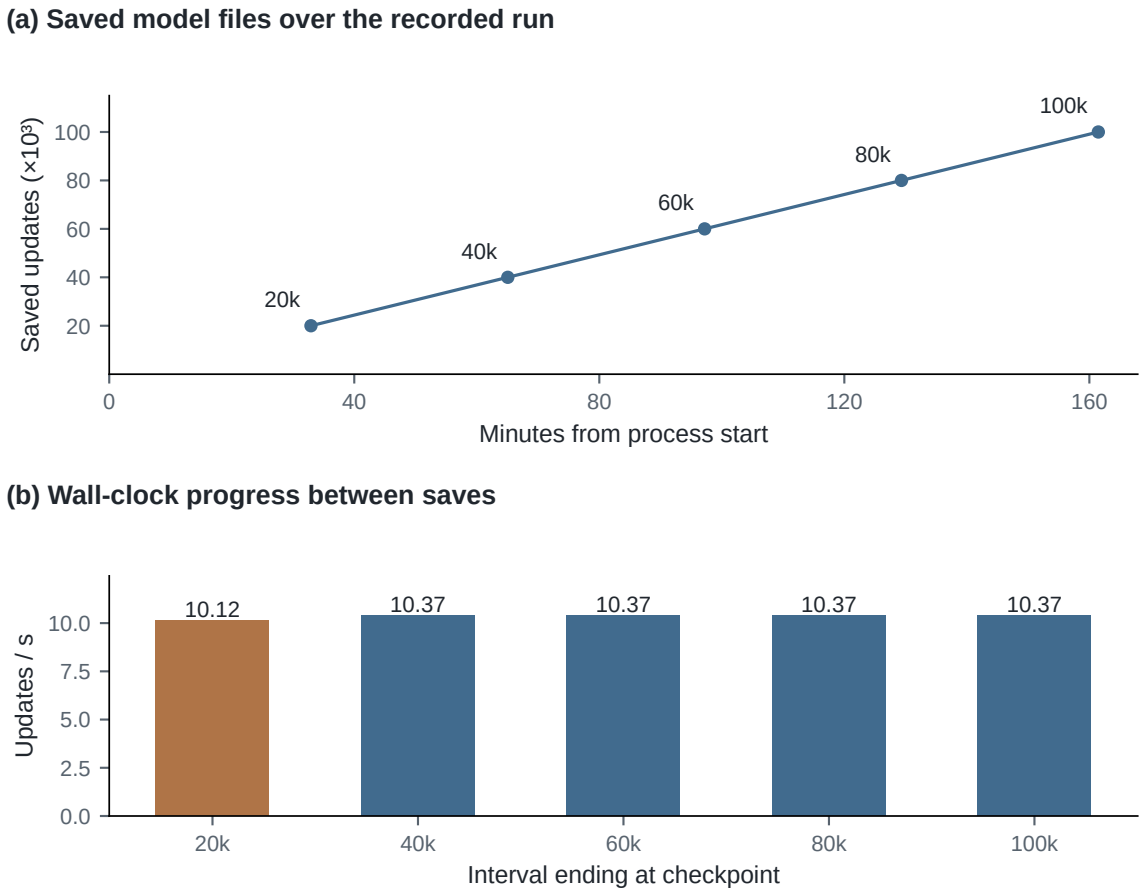


Figure 15: Checkpoint step versus model-file time and throughput between saves. The first interval includes initialization, and every interval may include data loading and saving. These measurements describe progress and wall-clock throughput; they do not replace missing loss records.

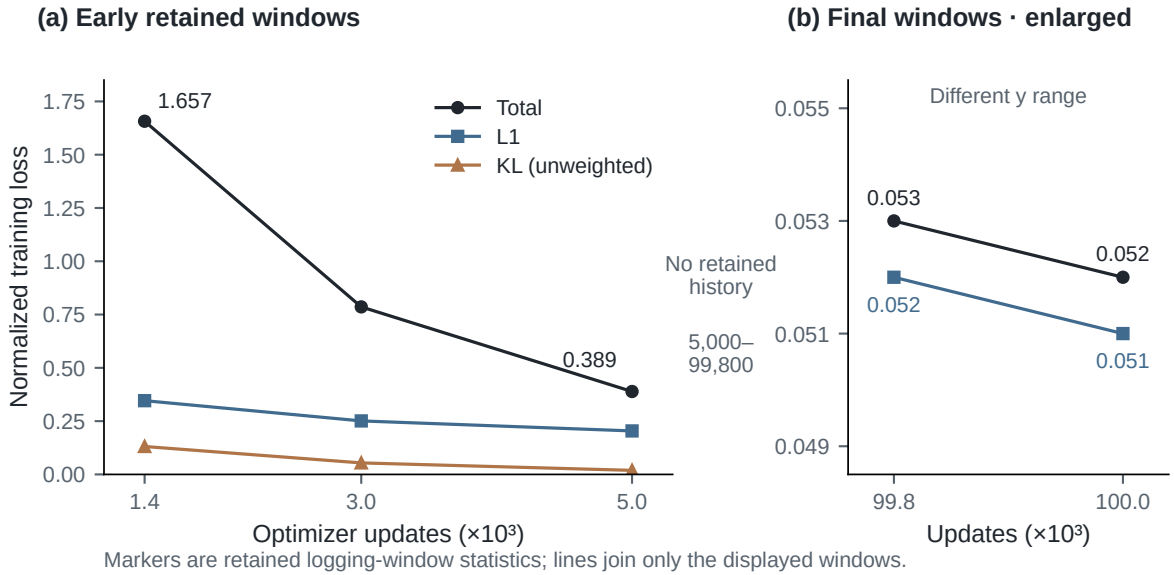


Figure 16: All retained loss observations, separated into early and final intervals. There is no retained loss history between steps 5000 and 99800. Each panel has its own scale, so their visual slopes should not be compared. No interpolation, smoothing, or validation series has been added.

Recomputing errors now would be a new evaluation; retraining would be a new run, not a recovery of this history.

The data-loading time in retained logs was about 0.001 s, with update times near 0.10 s. Gradient norms were finite in these intervals. Those records help identify a functioning optimization pipeline, but they do not provide a complete audit of every update or detect all possible training anomalies.

7.3 Loss composition and interpretation

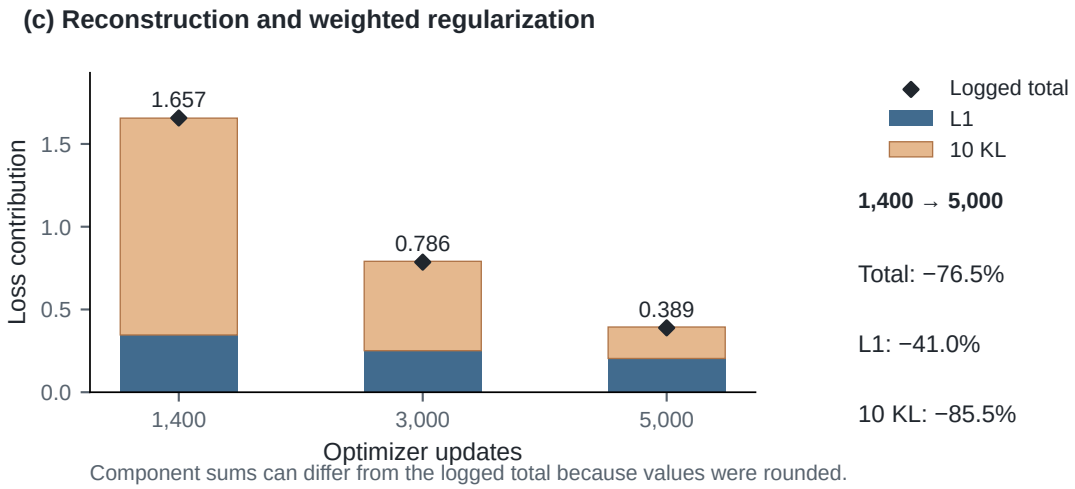


Figure 17: Early loss components and their relative decreases, calculated from the retained rounded log values. The regularization contribution is $10L_{\mathrm{KL}}$, while the raw metric table reports unweighted KL. Small inconsistencies between component sums and total loss reflect logging precision.

Between updates 1,400 and 5,000, total loss decreases by approximately 76.5%, whereas normalized L1 decreases by 41.0% and the weighted KL contribution by 85.5%. The earliest displayed KL contributes about 1.31 to a total of 1.657; by step 5,000 its displayed contribution is about 0.19. Thus a large part of the early total-loss reduction is consistent with the posterior approaching its standard-normal reference. It cannot all be credited to better action reconstruction. These component comparisons describe the objective balance, not measured grasp quality or latent interpretability.

Across the earliest and final retained summaries, normalized L1 falls by approximately 85.3%. That supports improved fitting on sampled demonstration windows. The final two L1 values, 0.052 and 0.051, differ by 0.001 over 200 updates. Their proximity does not establish when a plateau began, what happened during the missing 94,800-update interval, or how an independent validation loss would behave.

The displayed final total remains slightly above L1 even though KL prints as 0.000. Independently rounded components need not reproduce the displayed total. It would be incorrect to substitute an exact zero into a claim about posterior collapse, to infer a calibrated likelihood, or to calculate a precise final regularization fraction from these rounded entries.

The logged gradient norm is the logging-window mean of the total L2 norm returned by the gradient-clipping function before clipping. The saved threshold is 10.0. Consequently, retained values such as 63.917 can exceed the threshold without implying that gradients were applied at that norm. The observed window means decline to 6.162, but the five windows do not reveal individual-update extremes, clipping frequency, or every training anomaly. AdamW uses $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$; its adaptive first/second-moment updates and decoupled

weight decay were applied with the saved fixed learning rates.

Update times of 0.095–0.101 s and logged data time near 0.001 s suggest that the retained intervals were dominated by model updates rather than waiting for the next data batch. This is a local observation: prefetching and logging aggregation limit what those counters reveal about overall I/O. Checkpoint-derived throughput additionally includes initialization, saving, and other wall-clock overhead. Neither timing measure is a substitute for robot control frequency.

7.4 GPU utilization and memory

The training process was observed in GPU memory, and six subsequent `nvidia-smi` samples were captured at roughly 3 s spacing from 00:32:31 to 00:32:46. The device was an NVIDIA GeForce RTX 5070 Ti with 16,303 MiB total memory and driver 580.159.03. Figure 18 separates utilization from memory scope.

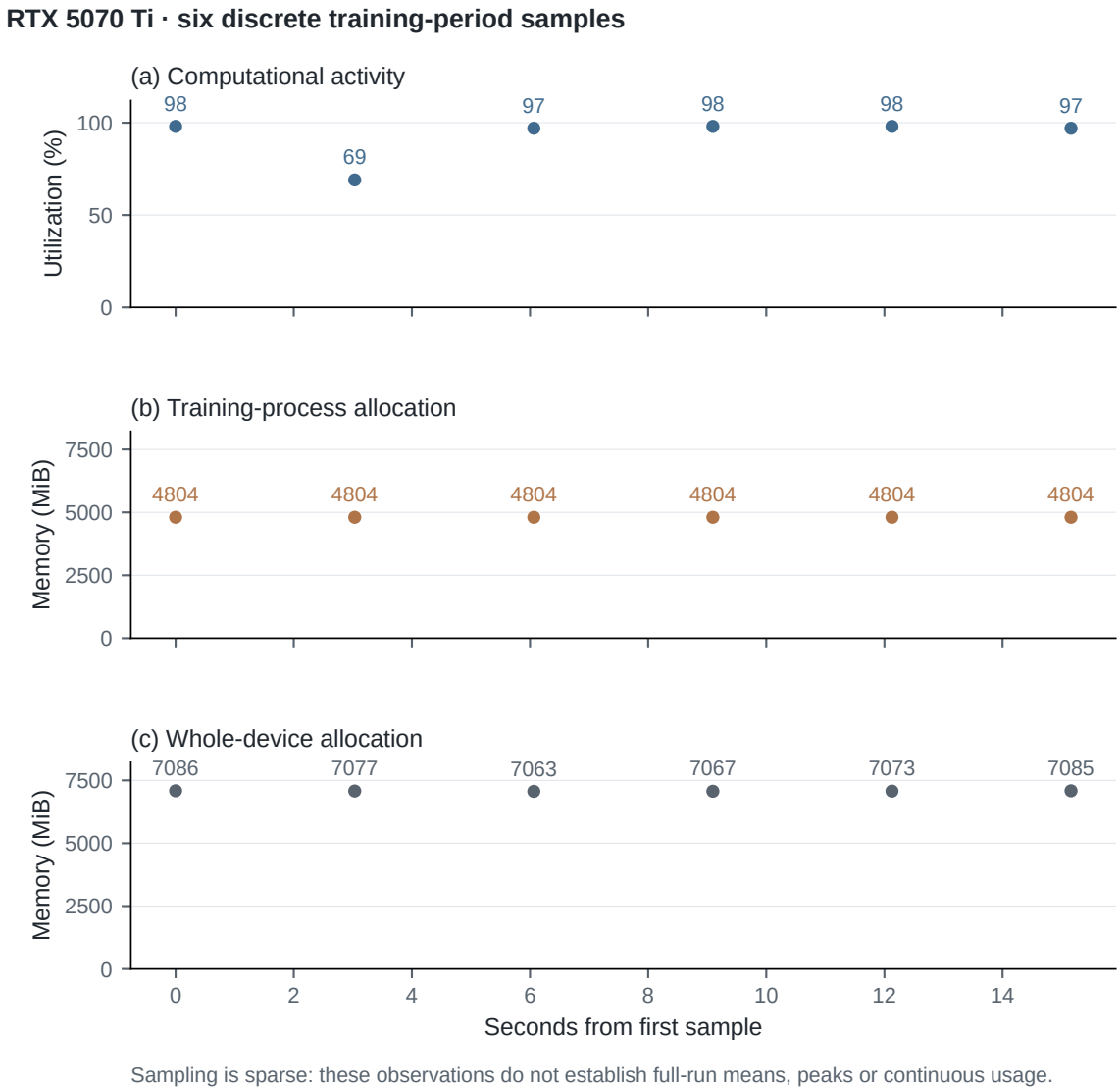


Figure 18: Six training-period GPU samples. Utilization ranges from 69% to 98%; training-process memory is 4804 MiB in each sample. Whole-device memory includes other allocations and is not interchangeable with process memory. Only six discrete observations are plotted.

Five utilization samples were 97–98% and one was 69%. The training process used 4,804 MiB,

about 4.69 GiB. Whole-device usage ranged from 7,063 to 7,086 MiB. Sampled power was 217.08–229.69 W and temperature 61–62 degrees Celsius. The training log's `mem_gb=3.73` uses a PyTorch memory metric with a different accounting scope; its difference from process memory is not evidence of an error.

The process allocation, high sampled utilization, CUDA configuration, and ongoing update logs together support that the recorded training used the GPU. Available memory can coexist with high compute utilization because capacity and computational activity are different quantities. AMP was disabled, while the code permitted TF32 matrix operations; this is not an FP16/BF16 mixed-precision run.

Six samples over about 15 s are insufficient to estimate the run's average utilization, peak memory, or energy consumption. Multiplying sampled power by the full training duration would assume a power history that was not measured. A later read-only check found the training process absent, with desktop allocations still present on the GPU; process completion does not imply zero whole-device memory usage.

7.5 Offline-check status and limitations

No offline-evaluation JSON is retained for this run. The evaluator's prediction, simplified-feedback, and range checks are described in Section 3, but no corresponding error score is reported. Episode 8, like the other 89 episodes, participated in training; selecting it afterward would be a training-set self-check, not held-out validation. A successful evaluator exit would establish completion of the computation, not a successful grasp.

Any later checkpoint comparison must keep the recorded observations, future offsets, preprocessing, bounds source, and scoring windows fixed. Body-joint MAE and gripper-scale MAE require separate interpretation. A fixed-image virtual feedback model cannot establish the physical response to a missed package.

7.6 Autonomous trials

7.6.1 Recordings and observation method

The operator supplied two H.264 recordings with AAC audio, each 1280×720 at nominal 30 fps. Video 1 contains 275 frames and lasts 9.167 s; video 2 contains 740 frames and lasts 24.667 s. Both were decoded completely without a reported decoding failure in the recorded inspection. Their hashes are retained, and report production preserved the original files.

The initial review sampled an overview every 0.5 s and added 0.2 s samples near selected contact/release events. The printable sequences use exact decoded frame indices and presentation timestamps, while event boundaries remain approximate human observations. Main-text stage labels remain outside the full frames, while presentation-time and frame-index tags occupy the lower-left corners. Outlined contact regions correspond to separately labeled crops, with source frame identities retained; the compact appendix uses lower-left time overlays. Original frames remain unannotated. The videos were not calibrated against a control clock, and playback speed was not independently verified. Their durations are recording durations, not separately measured complete-task latencies.

Three autonomous-loop startup records occur in the terminal buffer: 11:19:46, 11:20:15, and 11:21:35. One process reached its configured duration, while the other two ended with Ctrl+C. The operator identified two task trials represented by the supplied recordings. There is no synchronized mapping between those videos and the three launches. Process launches are therefore

not used as the trial denominator.

7.6.2 Trial 1: sustained grasp and placement

Figure 19 shows the first recording from initial placement through empty retraction. Contact and lifting occur around 3–4.5 s, sustained transport is visible from approximately 4.5 s, and release occurs around 6.5–7 s. The bag remains inside the basket afterward. One visible grasp cycle is counted, and it satisfies the first-cycle completion criterion.

This observation supports a complete visible transport-and-release sequence. It does not quantify contact force, fingertip position error, or the probability of repeating the result. The final withdrawal is visible but is qualified by the programmed shutdown return discussed in Section 6.

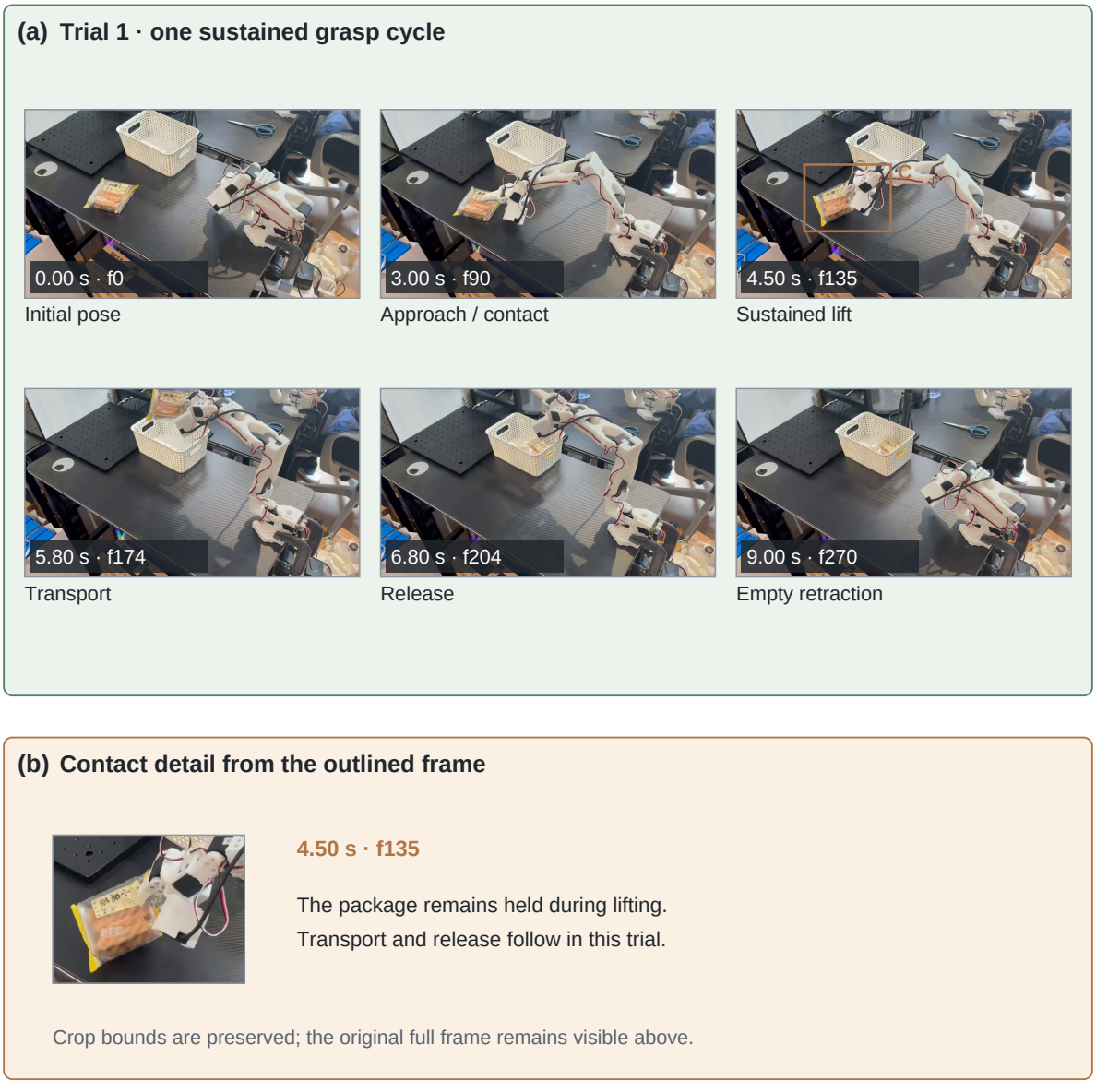


Figure 19: Video 1: one visible grasp cycle followed by sustained transport and release. Labels give decoded presentation times and frame indices; Panel (b) enlarges the outlined contact region in frame 135. Empty retraction is visible; its division between policy output and shutdown interpolation is unresolved.

7.6.3 Trial 2: multiple grasp cycles

The second recording differs at the contact stage. Figures 20 and 21 preserve the unsuccessful holding phases as well as the eventual placement. In the first cycle, the gripper approaches and closes, but by about 5 s it moves without the bag while the bag remains on the table. The second cycle briefly lifts or tilts the package near 11.5 s; by about 12 s the package is again on the table.

The third cycle begins around 17.5 s and produces sustained lifting and transport around 20–21.5 s. Release and settling occur around 21.5–22.5 s. The bag remains in the basket and the empty gripper withdraws. The operator reported no manual takeover or supplementary human grasp. Three visible cycles are counted within this one trial.

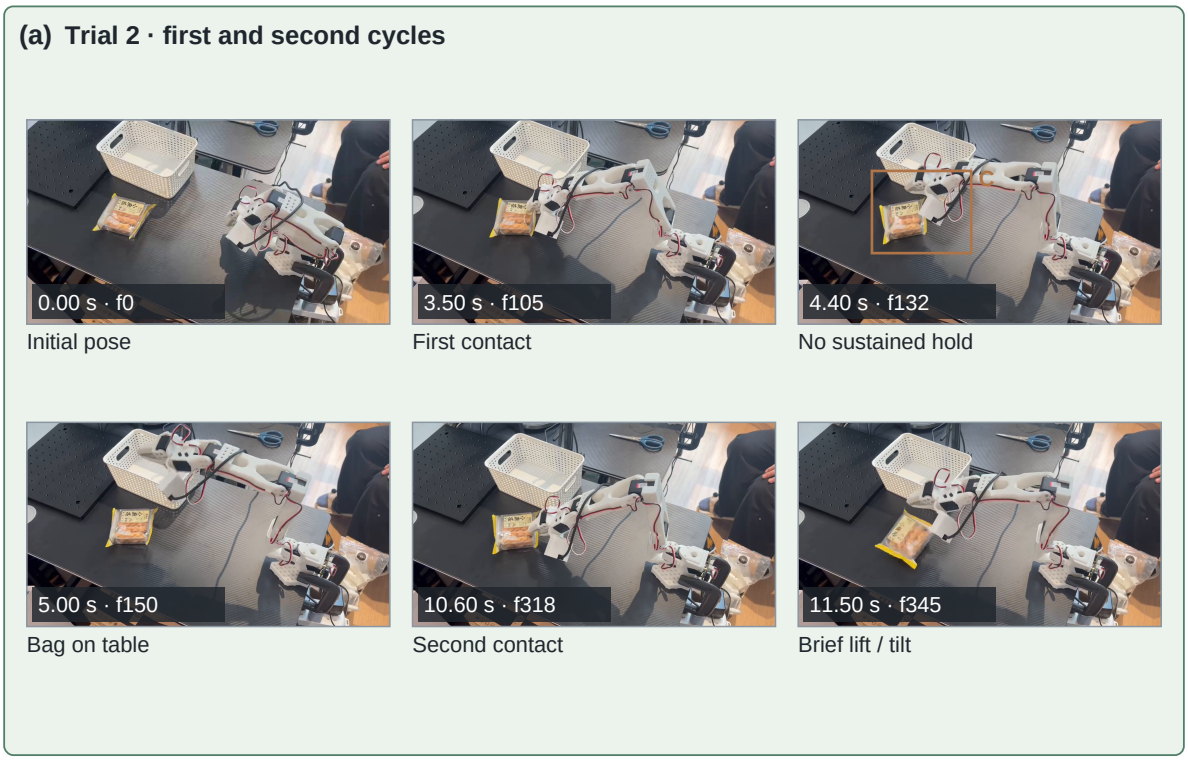
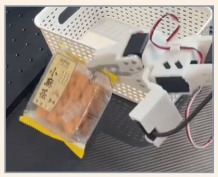


Figure 20: Video 2, first and second cycles. The first cycle does not sustain the package; the second produces a brief lift or tilt. Panel (b) enlarges the contact region from frame 132. The full sequence retains earlier contact failures alongside the second cycle.

(a) Trial 2 · transition, third cycle and placement



(b) Contact detail from the outlined frame



20.50 s · f615

A sustained lift is visible in the third cycle.

The bag is subsequently released in the basket.

Crop bounds are preserved; the original full frame remains visible above.

Figure 21: Video 2, return to the table and third cycle. The package is subsequently carried to the basket and released. Panel (b) enlarges frame 615 during lifting and transport. The 12.00 s frame belongs to the same trial as the final placement; the three cycles are not independent trials.

7.6.4 Counts and event timing

Describing this sequence as eventual completion after multiple cycles is supported by the images. Describing it as a verified failure-detection algorithm would require evidence about the policy's responses to controlled alternative observations. The current model has no separately measured success classifier, and the video alone does not reveal the internal reason for each later motion.

Table 14 separates endpoint completion from first-cycle completion. Figure 22 summarizes approximate event ranges. It does not reconstruct joint trajectories or policy-query instants. The extended sequences in Appendix C provide additional temporal context.

Table 14: Results for the two operator-designated trials. Counts refer to the recorded conditions and visible cycles.

Trial	Grasp cycles	Final placement	First-cycle completion	Empty retraction
Video 1	1	Yes	Yes	Visible
Video 2	3	Yes	No	Visible
Aggregate	4 cycles	2/2 trials	1/2 trials	2/2 visible

The endpoint count answers whether the package ultimately reached the basket and was released. The first-cycle count answers whether the task completed without another visible grasp cycle. Combining them avoids two misleading descriptions: the second case was not a first-attempt grasp, and its early misses did not prevent eventual placement.

With only two trials from one trained policy and unregistered positions, there is no basis for a reliable population success estimate, location-generalization claim, or comparison with another method. Training seeds, task trials, and within-trial grasp cycles are three different statistical units. Statistical practices designed for larger benchmark studies should not be copied mechanically to this sample (Henderson et al., 2018; Agarwal et al., 2021).

7.6.5 Deployment warnings and attribution

Startup-adjacent warnings reported approximately 5.7, 5.7, and 5.8 Hz, below the 30 Hz target. The inspected code derives that warning from an individual loop duration. It does not measure the complete recording's average frequency. Without per-step observation, inference, and publication timing, neither the warning nor the configured target identifies the actual sustained rate.

The terminal also reports a Rerun proxy buffer exceeding 1 GiB, dropping old visualization messages, and gRPC interruptions. These observations limit visualization-history completeness. They do not directly demonstrate that the robot control failed, that dataset files were damaged, or that a particular contact miss was caused by visualization overhead. The report retains the warnings without converting them into an unsupported causal explanation.



Figure 22: Approximate video event ranges from manual review. Colored segments identify contact/lift, transport, release, and other visible motion. Gaps are not interpolated. The recording end is dotted; time is not synchronized with control logs.

8 Discussion and Future Work

8.1 What the evidence supports

The strongest result is a physical one: the trained model participated in two operator-confirmed autonomous trials in which the bag was visibly transported into the basket and released. This is stronger evidence of task execution than a successful import, finite checkpoint tensors, or decreasing training loss. The first trial also shows that a single visible grasp cycle can suffice under one of the recorded conditions.

The training evidence supports successful completion of the prescribed update count and improved fitting in the retained intervals. The hardware evidence supports GPU use during the sampled training period. These results form a coherent end-to-end demonstration, but each remains tied to its measurement scope. There was no independent validation set, complete learning curve, multi-seed training comparison, or controlled checkpoint comparison.

8.2 Why low average error can coexist with grasp misses

The supervised objective aggregates errors over joints, times, and sampled windows. A short contact error can be numerically small in that aggregate while changing whether the bag stays between the fingertips. A flexible package can also change shape or orientation, so similar joint targets need not produce identical contact outcomes. The second trial makes the distinction visible without identifying a single cause.

The latent regularizer provides another reason to inspect loss components separately. A large early decrease in total loss partly reflects KL reduction. Even a lower reconstruction term measures target imitation rather than a task-specific contact objective. No loss in this run directly labels "bag securely held" or "bag inside basket."

8.3 Candidate explanations and discriminating tests

Table 15 organizes follow-up reasoning. Each proposed mechanism is plausible enough to test, but none is claimed as the established explanation. A useful test changes or measures the quantity that distinguishes competing causes.

8.4 A staged next evaluation

The first next experiment can use the current checkpoint without retraining: fix a documented set of initial placements, repeat them, and preserve complete video/control timing. A proposed small protocol is ten predefined placements with three trials each, for 30 trials. Record arm-relative position, bag orientation, lighting, basket location, reset method, manual intervention, first-cycle completion, final placement, cycle count, and stopping reason. Randomize or counterbalance condition order where practical.

These new trials would be independent executions. Because 75 current training episodes lack location labels, they would not automatically establish unseen-position generalization. To make that claim, a subsequent dataset must record training positions and reserve test positions before training. A test may use familiar locations to measure repeatability, or distinct locations to measure a specified spatial change; the label must match the protocol.

Checkpoint selection should use a declared validation procedure. The final test cases should not also be used repeatedly to choose the checkpoint or tune deployment settings. A model selected

Table 15: Observed issues, candidate explanations, and tests that could distinguish them. Proposed tests were not performed.

Observation	Candidate explanation	Discriminating measurement or comparison
Early cycles fail to maintain the bag	Contact point, bag orientation, gripper timing, or deformation differs from successful cases.	Synchronized close-up and joint/target logging; fixed position with controlled orientation changes.
Repeated cycles before completion	New observations influence later queries, or the learned sequence repeats a familiar pattern.	Record query instants; compare controlled post-miss bag placements under the same checkpoint.
Potentially delayed visual correction	A 100-step queue postpones the next visual replan.	Compare compatible prefixes with the same checkpoint and log end-to-end timing.
Long stationary training tail	Sampling places extra weight on stationary targets.	Retrain on a separately versioned, explicitly segmented dataset with other conditions controlled.
Startup frequency warning	Observation, inference, communication, or scheduling cost exceeds one tick.	Timestamp each component on a common clock, rather than infer from one warning.
Low retained loss, unknown generalization	Training fit may be good only near demonstrated conditions.	New independent trials with recorded conditions; a pretraining group split for a new model.

after observing many trials can appear stronger on those same trials than on a fresh evaluation. Demonstration-learning studies emphasize that selection protocol and data quality materially affect interpretation (Mandlekar et al., 2022).

8.5 Controlled ablations

Table 16 proposes a small set of useful comparisons. The entries are plans, with no reported performance. All comparisons require identical condition definitions and consistent scoring. Training seed repeats address training variability; repeated robot trials address execution variability. Neither substitutes for the other (Henderson et al., 2018; Agarwal et al., 2021).

Before a method-superiority claim, a later study should include matched baselines, multiple training seeds, and a sufficient number of consistently defined robot trials. The current experiment has no evidence that ACT is better than a single-step policy or Diffusion Policy on this task. Additional algorithms are a later research question, after the basic data and evaluation records are reliable.

8.6 Limitations

The study has one task, one object type, one training seed, one deployed checkpoint, and two recorded trials. Initial test coordinates were not registered. Only the first 15 demonstrations received the stated visual review; the full training curve and independent validation are absent. No offline evaluator result, synchronized action trace, contact-force measurement, or complete deployment latency series is available.

Table 16: Unperformed comparisons for a later authorized experiment.

Question	Controlled comparison	Required distinction
Does more frequent feedback help?	Same checkpoint, $k = 100$, compatible $h = 100, 20, 1$ settings.	Deployment comparison; measure latency and outcomes.
Does temporal ensembling help?	Compare $h = 1$ without and with a declared ensemble coefficient.	Keep query frequency comparable; do not confound ensembling with a 100-to-1 prefix change.
Does data cleanup help?	New version with explicit stationary-tail policy; retain the original.	Requires retraining and a split fixed beforehand.
Which view is useful?	Matched training/evaluation with top, wrist, or both.	Input changes require corresponding training; do not treat an untrained missing camera as a fair ablation.
Does chunk length matter?	Matched models with different k.	Output architecture and target windows change; a new training run is needed.

The operator's autonomy confirmation and the visible video evidence answer different aspects of the record. The supplied recording durations have not been verified as one-to-one real-time task durations. The three process launches cannot be mapped to the two videos. The visible return motion cannot be completely assigned to the policy because shutdown interpolation also returns the arm. These limits restrict claims about timing, recovery, learned return, and generalization while leaving the observed placement/release results intact.

9 Conclusion

Ninety SO-101 demonstrations supported a completed 100000-step ACT training run and two recorded autonomous pick-and-place cases. The model used current fixed/wrist images and six joint readings to predict 100 six-dimensional position targets. Training combined valid-element L1 reconstruction with a weighted KL regularizer; deployment fixed the latent to zero and consumed a 100-step action queue before requesting another block.

The retained training records show improved fitting and GPU execution. The videos show final placement and release in both trials, with first-cycle completion in one. The second trial's early holding failures show why endpoint completion, first-cycle performance, and average target error should all be reported. They also identify contact-stage behavior as a useful subject for the next controlled investigation.

A stronger next result would come from complete logs, synchronized query/action/video records, documented initial conditions, and a fixed evaluation protocol. Those measurements would allow timing, feedback, and contact hypotheses to be distinguished. A new training experiment with episode/condition separation would then support independent validation and more defensible checkpoint selection.

A Notation and Units

Table 17: Notation used throughout the report.

Symbol	Meaning	Convention
t	Anchor or control-step index	Discrete index; wall-clock time requires separate measurement.
o_t	Current observation	Two RGB images and measured follower state.
q_t	Measured joint-state vector	Six components; five angles and one gripper scale.
a_t	Target vector	Same six-component ordering as the state.
$\tilde{a}_t$	Normalized target	Dimensionless after per-component normalization.
A_t	Future target block	k rows, indexed 0 through $k-1$.
$\hat{A}_t$	Predicted target block	Ordered outputs, not an unordered object set.
k	Prediction length	100 in the run; 4 in the worked example.
h	Executed prefix length	100 in the run; 2 in the worked example.
f	Target control frequency	30 Hz; not a measured full-run frequency.
B	Batch size	8 for training, usually 1 for live inference.
D	Action dimension	6.
d_{model}	Transformer hidden dimension	512.
H	Attention-head count	8; avoids reusing h.
d_k	Key dimension per head	64 when 512 is divided among 8 heads.
z	Latent vector	32 components; fixed to zero at deployment.
μ, ℓ	Posterior parameters	$\ell = \log \sigma^2$, not $\log \sigma$.
ϵ	Sampling noise	Standard normal in the reparameterization equation.
ε_s	Normalization stabilization	Distinct from the sampled latent noise.
$m_{b,j}$	Valid-target mask	1 for genuine data, 0 for padding.
β	KL weight	10 for this run.
m	Ensemble coefficient	Distinct from mask $m_{b,j}$; no ensemble was enabled.
$\hat{a}_{p \to t}$	Prediction made at p for target t	Ensemble alignment notation.

B Saved Configuration and Evidence Provenance

B.1 Configuration records

Tables 18 and 19 are generated from the archived completed-checkpoint configuration. The full JSON files accompany the report under `report_assets/data/saved_policy_config.json` and `saved_train_config.json`. They preserve fields beyond the settings discussed in the main text. Feature shapes are explained in Table 5; software versions appear in Table 10 and the original installed-package list.

Table 18: Algorithm fields from the saved policy configuration. The companion JSON retains all fields, including feature shapes.

Field	Saved value
type	act
n_obs_steps	1
device	cuda
use_amp	false
chunk_size	100
n_action_steps	100
vision_backbone	resnet18
pretrained_backbone_weights	ResNet18_Weights.IMAGENET1K_V1
replace_final_stride_with_dilation	false
pre_norm	false
dim_model	512
n_heads	8
dim_feedforward	3200
feedforward_activation	relu
n_encoder_layers	4
n_decoder_layers	1
use_vae	true
latent_dim	32
n_vae_encoder_layers	4
temporal_ensemble_coeff	None
dropout	0.1
kl_weight	10.0
optimizer_lr	1e-05
optimizer_lr_backbone	1e-05
optimizer_weight_decay	0.0001

Table 19: Additional saved training and optimizer fields.

Field	Saved value
seed	1000
cudnn_deterministic	false

Field	Saved value
num_workers	4
batch_size	8
prefetch_factor	4
persistent_workers	true
dataloader_multiprocessing_context	spawn
steps	100000
env_eval_freq	20000
log_freq	200
eval_steps	0
max_eval_samples	0
tolerance_s	0.0001
save_checkpoint	true
save_freq	20000
use_policy_training_preset	true
scheduler	None
optimizer.type	adamw
optimizer.lr	1e-05
optimizer.weight_decay	0.0001
optimizer.grad_clip_norm	10.0
optimizer.betas	[0.9,0.999]
optimizer.eps	1e-08

B.2 Files that support the measurements

Table 20 identifies the original evidence by its function. Paths are relative to `experiment_assets/evidence`. The report's figures are derived from these records; no training loss or evaluation score was synthesized to fill a missing interval.

Table 20: Original evidence files and the claims they support.

File	Content and scope
dataset_snapshot.json	Dataset counts, structure checks, durations, known labels, and review limitations.
core_manifest.json	SHA-256 and byte sizes for the 27 original/training-copy core files.
episodes.csv	Episode durations, indices, metadata matches, and timestamp differences.
frames/frames.json	Demonstration frame indices and episode/container times.
action_chunk_sample.json	Recorded teacher targets for episode 8, frames 120–219.
environment.json	Training environment, defaults, core source identities, cached backbone identity.

File	Content and scope
`installed_packages.txt`	Installed package versions; not a complete hashed installation lock.
`gpu_process_snapshot.json`	Process identity, command, start time, and six GPU samples.
`training_metrics_completed.csv`	The five retained loss/gradient/timing summaries.
`terminal-snapshot-completed.txt`	Final summaries, save message, and end-of-training text.
`training_completion.json`	Checkpoint steps, hashes, model times, tensor and processor inspections.
`gpu_after_training.json`	Later absence of the training process, with remaining desktop allocations.
`video_metadata.json`	Recording codecs, dimensions, frame counts, and durations.
`video_review.json`	Operator statement, video hashes, manual event ranges, and outcome counts.
`deployment_snapshot.json`	Loaded policy settings, final model identity, launches, and mapping limitations.
`terminal-rollout-20261013.txt`	CUDA selection, frequency/visualization warnings, and shutdown messages.
`deployment_cameras.json`	Camera names and direction settings captured at review.

The generated `report_assets/data/frame_manifest.json` gives the exact decoded frame selected for each requested video time. The `report_assets/paper_final/data/generation-all.json` manifest identifies each revised figure's sources and purpose, including constructed illustrations. The accompanying crop record retains source frame identities and pixel bounds. These records identify the evidence behind each frame or curve; the plotting script does not supply experimental outcomes.

B.3 Model, video, and source identities

The final model's SHA-256 is:

```
df59354de551f19a3c406b85ce8f143701abd3fce648781f5f8ec1b7ff387999
```

It refers to `outputs/train/act_snack_in_box/checkpoints/100000/pretrained_model/model.safetensors`. The policy package includes configuration and processors; training checkpoints additionally retain optimizer/training state files. Model weights are not duplicated into this report package.

The original recordings retain these identities:

Video 1

`fc9bc9aec2d7a952a35461bb7b2a447e2264575ffcb54dc93597f03045beedde`

Video 2

`5adf0ed938f5d616b659924d9e79c08c4ac2ad0b1bd28bdde118c5c54a140e36`

The cached ImageNet initialization identity is:

```
f37072fd47e89c5e827621c5baffa7500819f7896bbacec160b1a16c560e07ec
```

Table 21 records the five implementation files captured during training. Exact copies are included under `report_assets/source_snapshots`; their original headers are retained. The complete installed package is still an environment dependency, so these source files are provenance references rather than a stand-alone replacement for LeRobot.

Table 21: Training-period source SHA-256 values. The names identify companion snapshots.

Snapshot	SHA-256
`act_config.py`	`35a60c0dd3c811b137b5eb035bdbd40d40e2c24ba0cda61756052e6e1c829282`
`act_model.py`	`a6c1a49c5daafa513b9abee9b986205d327e25c70ed13623ab2e6c0ab6b1f577`
`train_script.py`	`059bda054bbb188d4a40d0a32941aa4d9ac59252a37d67a258e685a0a6d063ec`
`train_config.py`	`117d9829af0db49a681876861ac6fb014bf1927b03655c3428d0ec1db968af79`
`dataset_config.py`	`e8307a48aa97c1b3fd0722e9a849deb1bc1bb48b63e3109b63683eb7ffe08105`

The locally corrected offline evaluator has SHA-256

```
c4858cfcece9040baf04db1d1838dc31f2f2c15a4a39f9d79fbc2e3a97b3cd46
```

The checked public file revision is `d3c410123a4c3c7c6146ed7c83a116aa7f3d5142`, dated 29 September 2026. Its content SHA-256 is

```
16cb539f0e4f5b06f0e2cbbb7bfe465469301d0b1f4500f32a8e6ba9178c323c
```

The public revision's input uses teacher actions where the local version uses recorded state. This is a methodological difference, not simply an alternative filename. The comparison metadata is included under `report_assets/data/offline_source_comparison.json`.

B.4 Key implementation locations

The ACT policy configuration defines action offsets and architectural settings. The model snapshot contains `ACTPolicy.forward`, `ACT.forward`, `ACTPolicy.select_action`, and `ACTTemporalEnsembler`. These functions respectively define loss reduction, the posterior/observation/decoder paths, queue consumption, and ensemble weighting. The local evaluator's `evaluate_one` and `_obs` functions define its scoring and state-input construction.

Function names and hashes are more useful than an unqualified line number: line numbers can move even when a behavior is unchanged. Reproduction should record the installed implementation and saved configuration together. Exact numeric repeatability additionally depends on libraries, hardware, seeds, and execution details that are not all controlled by a version label (Henderson et al., 2018).

C Extended Video Frame Sequences



Figure 23: Video 1: twelve preserved overview frames from 0 to 9.0 s, in two six-frame groups. Read left to right, then top to bottom. The overlay reports actual selected PTS and zero-based frame index; all original full-frame images remain available without overlays.



Figure 24: Video 2: eighteen preserved overview frames from 0 to 24.5 s, in three six-frame groups. The sequence retains the early unsuccessful contact periods and the final placement. Overlays identify selected video frames, not synchronized control events.

D Common Questions

D.1 Task and data

Q1. What are the six policy outputs, and why do their units matter?

The outputs are shoulder pan, shoulder lift, elbow flexion, wrist flexion, wrist roll, and gripper opening. The first five are degrees; the last is a 0–100 scale. A combined physical-unit average mixes degrees and scale points, so per-joint errors are needed.

Q2. Why is an action target different from an observed joint state?

The target is a command; the reading is the realized/observed position. Tracking dynamics and delay can make them differ. Substitution changes the input distribution and can invalidate a comparison with deployment.

Q3. Which targets and observations form a four-action training sample anchored at $t = 10$?

Targets are $a_{10}, a_{11}, a_{12}, a_{13}$. This run uses one current pair of images and one current state, not four image pairs.

Q4. How do optimizer steps, epoch-equivalents, trials, and grasp cycles differ?

A step is one optimizer update. An epoch-equivalent counts sampled anchors relative to dataset size. A trial is one designated task execution; several grasp cycles can occur within it.

Q5. Can the 90 training episodes later become a held-out set?

The checkpoint already fitted all 90. A post hoc selection remains training data for that model. New independent data or a new model trained with a prior split is required.

D.2 Network and objective

Q6. How are the visual-token and encoder-token counts obtained?

Each view has $15 \times 20 = 300$ spatial tokens. Two views give 600; state and latent add two, for 602 policy encoder tokens. The posterior encoder has a different 102-token sequence.

Q7. What does an action query represent, and why is DETR matching absent?

A query identifies a future time slot in an ordered action block. DETR predicts an unordered object set and uses matching; those labels and losses are not the ACT objective.

Q8. What enters the posterior encoder, and what remains at deployment?

The posterior consumes current joint state and future demonstrated actions, with padding masked. The observation-fusion encoder remains during deployment; only the training posterior branch is bypassed.

Q9. How is a latent sampled from a predicted log variance?

Use $z = \mu + e^{\ell/2} \odot \epsilon$ with standard-normal noise. Since $\ell = \log \sigma^2$, e^{ℓ} is variance and its square root is $e^{\ell/2}$.

Q10. Does fixing $z = 0$ average all possible action sequences?

The decoder is nonlinear. Its value at the mean of the input latent need not equal the mean of its outputs over latent draws; it also need not be the marginal output mode.

Q11. For four six-dimensional rows with only two valid rows and absolute-error sum 2.4, what masked L1 is obtained? What would division by all 24 elements produce?

The valid-element mean is $2.4/12 = 0.2$. Dividing by all 24 elements incorrectly gives 0.1, making extra padding appear to improve fitting.

Q12. How is KL reduced, and what does its weight 10 mean?

KL is summed over 32 latent coordinates and averaged over the batch. The value 10 scales its contribution relative to the implemented reconstruction reduction; it is not a probability or a physical-unit conversion.

Q13. What does a logged KL value of 0.000 establish?

It is a rounded displayed value. It neither proves exact equality to zero nor establishes posterior collapse without additional latent/posterior measurements.

D.3 Execution and feedback

Q14. Which query supplies each action when $k = 4$ and $h = 2$?

Query 0 supplies times 0 and 1; query 2 supplies times 2 and 3. Its old unexecuted rows are discarded in the prefix strategy.

Q15. How long is a 100-target queue at a nominal 30 Hz?

Executing 100 nominal ticks takes $100/30 \approx 3.33$ s. The first-to-last target sample-time span is $99/30 = 3.30$ s. Actual wall-clock timing requires measurements.

Q16. Does a 30 Hz camera imply 30 ACT queries per second?

The execution wrapper may consume queued actions while acquiring newer observations. In this run, a new block is predicted only when the 100-action queue empties.

Q17. How should temporal ensembling be compared fairly?

Compare $h = 1$ without ensembling with $h = 1$ and a declared coefficient, holding checkpoint, input processing, conditions, and scoring fixed. Log timing. Comparing $h = 100$ against an ensemble confounds query rate and weighting.

Q18. What evidence would attribute the final rest motion to ACT?

Synchronize policy targets, process-stop/return events, and video; inspect whether withdrawal precedes the programmed shutdown interpolation. Visible retraction alone cannot separate the sources.

D.4 Training and evaluation

Q19. Can five loss summaries identify convergence or the best checkpoint?

Missing intermediate records allow many unobserved training paths. Final and early summaries show fitting improvement but not when a plateau occurred, whether overfitting occurred, or which checkpoint performs best on independent data.

Q20. How do process memory, device memory, and GPU utilization differ?

Process memory belongs to the recorded process's accounting scope; whole-device usage includes other allocations. Utilization describes computational activity during a sampling interval, not the fraction of memory occupied.

Q21. Why did an unquoted episode placeholder cause a shell error?

Bash interprets the angle bracket as input redirection and tries to open a named file. A numeric episode argument removes that parsing error; quoted repository placeholders still need real identities.

Q22. What is checked in degrees mode without a model directory?

Only the gripper's 0–100 bounds are finite in that mode. The five body-joint ranges are not checked. A matching URDF adds numeric joint limits, not collision detection.

Q23. Why is the simplified feedback check physically inconsistent?

The virtual state follows predicted targets, but images remain those of the recorded demonstration. The pair can describe incompatible physical scenes, and the gain 0.35 is not measured robot dynamics.

Q24. What are the recorded placement and first-cycle completion counts?

Final placement and release occurred in 2/2 trials; first-cycle completion occurred in 1/2. There are four visible grasp cycles, one in video 1 and three within video 2. Four is not the trial denominator.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://papers.nips.cc/paper/2021/file/f514cec81cb148559cf475e7426eed5e-Paper.pdf.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision – ECCV 2020*, volume 12346 of *Lecture Notes in Computer Science*, pages 213–229, 2020. doi: 10.1007/978-3-030-58452-8_13. URL https://arxiv.org/abs/2005.12872.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://www.roboticsproceedings.org/rss19/p026.html.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. URL https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018. doi: 10.1609/aaai.v32i1.11694. URL https://ojs.aaai.org/index.php/AAAI/article/view/11694.

Hugging Face. LeRobot: Action chunking with transformers documentation, n.d. URL https://huggingface.co/docs/lerobot/act. Accessed 14 October 2026. Installed implementation in this experiment: version 0.6.1; local source identities are recorded in the report.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014. URL https://arxiv.org/abs/1312.6114. Preprint first submitted in 2013.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1678–1690, 2022. URL https://proceedings.mlr.press/v164/mandlekar22a.html. Conference held in 2021.

PyTorch Contributors. Torchvision: resnet18 model documentation, n.d. URL https://docs.pytorch.org/vision/stable/models/generated/torchvision.models.resnet18.html. Accessed 14 October 2026.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

Kihyuk Sohn, Xinchen Yan, and Honglak Lee. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, volume 28, 2015. URL https://papers.nips.cc/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://arxiv.org/abs/1706.03762.

Yanshi Robotics. So-101 offline evaluation script, 2026. URL https://github.com/Yanshi-Robotics/Lets-Build-Robots-S1-SO101/blob/d3c410123a4c3c7c6146ed7c83a116aa7f3d5142/ACT-1-Pick/offline_eval.py. Public file revision dated 29 September 2026, checked 14 October 2026; differs from the local evaluated source snapshot.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.016. URL https://www.roboticsproceedings.org/rss19/p016.pdf.