

Choose the Noise, Keep the Flow:

Noise-Space Steering of Flow and Diffusion Policies, Read as a VAE

Yuhan Chi

Fudan University

<https://yuhanchi.github.io/writing/choose-the-noise/>

October 18, 2026

Abstract

A family of methods steers a pretrained flow-matching or diffusion policy without changing its weights: a small proposer chooses the input noise for each state, and the frozen generator turns that noise into an action. This note collects the family, which is not grouped together in a recent survey of flow matching in reinforcement learning, and reads it through a VAE-style lens: the proposer is an encoder, the frozen generator is a decoder, the Gaussian is the prior, and value replaces reconstruction. For a KL-regularized objective the connection is an exact variational identity. It then asks where reinforcement learning takes effect, and organizes the methods by how they carry an action-space value back into noise space: a noise critic, flow inversion, backpropagation through the sampler, a one-step decoder, or a likelihood-ratio gradient. A two-dimensional experiment trains the same proposer with all five routes. This PDF is a version of a blog post; it is not peer reviewed.

A recent survey by Nahim, Khan, and Ju, [A Survey of Flow Matching in Reinforcement Learning](#) (TMLR, 2026), organizes the field along two axes: *what* the flow models (actions, critics, dynamics) and *where* the RL signal enters (energy-weighted regression, policy gradients through the ODE, GRPO-style objectives). It is a useful map. This note examines one family that the survey does not treat as a group.

At the center are methods that leave the velocity field untouched. They keep a pretrained flow or diffusion policy frozen and train a small network that decides **which noise to feed it**. Given a state s , a proposer picks a point w in noise space, and the frozen generator turns w into an action. Consider a robot policy that sometimes reaches around an obstacle on the left and sometimes on the right. Steering the noise changes how often it chooses each route, while reusing the generator that produces the movements.

The survey already covers an important relative. Its offline section discusses ReFORM ([Zhang et al., 2026](#)), which manipulates noise inside a bounded support so that the composed policy stays within the learned behavior model’s support. Its discussion of “freezing the flow and learning a correction” covers policies that *edit the action after it is generated*. DSRL ([Wagenmaker et al., 2025](#)) and related methods that *choose the input* instead of editing the output are absent. This note follows that thread, including a few neighbors that also train the decoder or learn from corrections rather than rewards.

The note makes two points and then checks them on a small example.

1. A **VAE-style inference view** connects the methods. The noise proposer plays the role of an encoder and the frozen generator that of a decoder. For a KL-regularized objective, this becomes an exact variational identity, with value taking the place of reconstruction quality. Other methods share the architecture without optimizing that same objective.
2. The most useful question to ask of each method is **where RL takes effect**. Action-space data can train a critic directly, but the proposer lives in noise space. Each method is defined largely by how it carries value from one space to the other.
3. A **two-dimensional experiment** (Section 4) trains the same noise proposer with five different ways of carrying value, so one can watch the noise move and the actions follow.

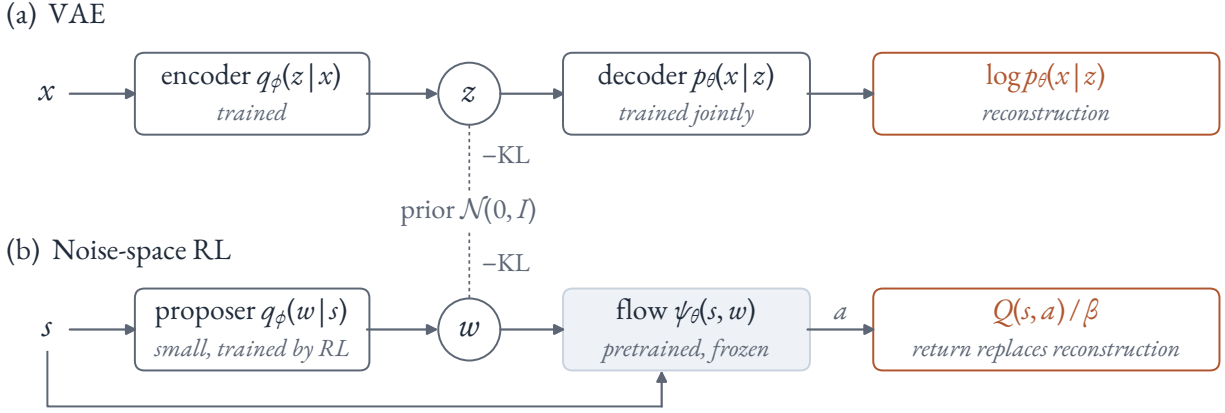


Figure 1: The shared encoder–decoder pattern. (a) A VAE trains an encoder and a decoder jointly to reconstruct its input. (b) In noise-space RL, a proposer q_ϕ chooses the noise w for a state s , and a pretrained, frozen flow ψ_θ decodes it into an action. Both latents are pulled toward the same prior $\mathcal{N}(0, I)$ by a KL term. The decoder in (b) produces an action rather than a reconstruction, and value Q/β takes the place of the reconstruction term. The correspondence is between objectives (Section 2), not between generative models.

1 Setup and notation

A flow-matching policy learns a velocity field $v_\theta(x, t, s)$ that transports Gaussian noise at time $t = 0$ to actions at $t = 1$. To act, one draws $w \sim \mathcal{N}(0, I)$ and integrates $\dot{x}_t = v_\theta(x_t, t, s)$ from $x_0 = w$, usually with a few Euler steps,

$$x_{k+1} = x_k + \Delta t v_\theta(x_k, t_k, s), \quad t_k = k \Delta t, \quad \Delta t = 1/K, \quad k = 0, \dots, K-1, \quad (1)$$

and outputs $a = x_K$. Run with any deterministic sampler (this Euler scheme, the probability-flow ODE of a diffusion policy, DDIM with zero variance, or a one-step distilled model), the policy becomes a fixed map

$$a = \psi_\theta(s, w), \quad w \sim \mathcal{N}(0, I), \quad (2)$$

and all of its randomness enters through w . Table 1 collects the notation used in the rest of the note; the decoder parameters θ are pretrained and stay frozen, and only the proposer parameters ϕ are learned.

Table 1: Notation. The decoder parameters θ are pretrained and frozen; only the proposer parameters ϕ are learned.

Symbol	Meaning
s, a	state and action (for robot policies, often a chunk of future actions)
$w \in \mathbb{R}^d$	noise fed to the generator; for flow policies, d equals the action dimension
v_θ, K	velocity field of the pretrained flow, and the number of sampler steps
$\psi_\theta(s, w)$	the decoder : the deterministic sampler that maps noise w to action a
$f_{\#}\mu$	pushforward: the distribution of $f(x)$ when $x \sim \mu$
$\pi_{\text{base}}(\cdot s)$	the pretrained policy, $\psi_\theta(s, \cdot)_{\#} \mathcal{N}(0, I)$
$q_\phi(w s)$	the noise proposer (encoder), usually a state-conditioned Gaussian
$\pi_\phi(\cdot s)$	the steered policy, $\psi_\theta(s, \cdot)_{\#} q_\phi(\cdot s)$
$Q(s, a)$	an action-value estimate (critic)
$\beta > 0$	temperature: weight of the KL penalty that keeps q_ϕ near $\mathcal{N}(0, I)$
q^*, π^*	the optimal tilted noise distribution and policy, defined in Section 2

A noise-space method keeps θ fixed and replaces only the distribution of the input, from $\mathcal{N}(0, I)$ to $q_\phi(\cdot | s)$. The proposer can be a Gaussian with a state-dependent mean and variance, a deterministic map, a residual added to fresh noise, or even a single constant vector. It is usually smaller than the generator, though not necessarily tiny: Noise PPO uses a roughly 187M-parameter proposer. The decoder can be very large (DSRL steers π_0 , a 3.3B-parameter generalist policy) and is never updated.

DSRL states the construction cleanly. Define a new MDP whose actions are noises:

$$P^W(\cdot | s, w) = P(\cdot | s, \psi_\theta(s, w)), \quad r^W(s, w) = r(s, \psi_\theta(s, w)). \quad (3)$$

The diffusion policy becomes part of the environment. Any RL algorithm can now run over w , using only forward passes of the generator, without its weights or its intermediate denoising steps.

The idea is older than diffusion policies. PLAS (Zhou et al., 2020) learned a policy in the latent space of a conditional VAE for offline RL. PARROT (Singh et al., 2021) trained a normalizing-flow behavioral prior and ran RL on its noise. It compared directly against a conditional-VAE latent space and argued that invertibility lets the agent keep full control over the final action. DSRL singles out PARROT as the precedent in control and makes the same point from the other side: diffusion models are not invertible by construction, so it is “much less clear” that steering their noise is expressive enough. Empirically, DSRL finds that it is.

2 Reading it as a VAE

Recall the evidence lower bound of an ordinary VAE (Kingma & Welling, 2014); a conditional VAE additionally conditions its distributions on context:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x|z)] - \text{KL}(q_\phi(z|x) \| p(z)). \quad (4)$$

To make the connection precise, consider the following KL-regularized policy-improvement objective:

$$J(\phi) = \mathbb{E}_s \left[\mathbb{E}_{w \sim q_\phi(\cdot|s)} Q(s, \psi_\theta(s, w)) - \beta \text{KL}(q_\phi(\cdot|s) \| \mathcal{N}(0, I)) \right]. \quad (5)$$

Here Q is held fixed during the proposer update, and states are sampled from a fixed training distribution. This is a local improvement objective, not by itself a complete solution to sequential RL. Table 2 lines up the two expressions term by term.

Table 2: Correspondence between the terms of a VAE and KL-regularized noise-space RL.

VAE	Noise-space RL
observed input x	desired optimality, with state s as context
encoder $q_\phi(z x)$	noise proposer $q_\phi(w s)$
prior $p(z) = \mathcal{N}(0, I)$	the Gaussian the flow was trained from
decoder $p_\theta(x z)$	frozen flow $\psi_\theta(s, w)$
reconstruction $\log p_\theta(x z)$	value $Q(s, \psi_\theta(s, w)) / \beta$
KL to the prior	KL to $\mathcal{N}(0, I)$: penalize departure from the training prior
decoder trained jointly	decoder pretrained by imitation, usually frozen

The exact connection is a variational identity. Define $Z(s) = \mathbb{E}_{w \sim \mathcal{N}(0, I)} [\exp(Q(s, \psi_\theta(s, w)) / \beta)]$ and assume it is finite. Then, for each state,

$$\log Z(s) = \underbrace{\mathbb{E}_{q_\phi} [Q(s, \psi_\theta(s, w)) / \beta] - \text{KL}(q_\phi \| \mathcal{N}(0, I))}_{J(\phi; s) / \beta} + \text{KL}(q_\phi \| q^*). \quad (6)$$

The last term is nonnegative, so the first two terms form a lower bound. In the control-as-inference view (Levine, 2018), this can be read as an ELBO for a desired “optimality” observation. When Q is bounded above, a state-dependent scaling makes the exponential a valid binary likelihood; the variational identity itself needs only a finite $Z(s)$. The unrestricted best proposer is

$$q^*(w|s) = \frac{\mathcal{N}(w; 0, I) \exp(Q(s, \psi_\theta(s, w)) / \beta)}{Z(s)}. \quad (7)$$

Writing $J(\phi; s)$ for the term inside the outer expectation of J , the identity also gives a ruler: for any proposer, $\log Z(s) - J(\phi; s) / \beta = \text{KL}(q_\phi \| q^*)$ measures how far it is from the best one. Section 4 uses it.

The difference from an ordinary VAE is what counts as the observation. The encoder is not asked to explain an action that was seen. It is asked to infer which noise *would have produced a good action*. The evidence is desired success, represented here by the value estimate. This is why “VAE-style” is more accurate than calling every method a conditional VAE. A deterministic proposer, for example, has infinite KL to a continuous Gaussian; box constraints, spherical latents, and supervised inversion require their own objectives.

Two facts make this reading useful. [Eyring et al. \(2025\)](#) state both for distilled text-to-image generators, and both carry over to policies without change.

Proposition 1 (Tilting the noise tilts the policy). *Pushing the noise posterior through the frozen decoder gives exactly the reward-tilted base policy:*

$$\psi_\theta(s, \cdot)_\# q^\star(\cdot | s) = \pi^\star(\cdot | s) \propto \pi_{base}(\cdot | s) \exp(Q(s, \cdot)/\beta). \quad (8)$$

The reason is that the tilt depends on w only through $a = \psi_\theta(s, w)$.

Proof of Proposition 1. For a bounded test function f , suppressing the fixed state s ,

$$\begin{aligned} \mathbb{E}_{q^\star}[f(\psi_\theta(w))] &= Z^{-1} \mathbb{E}_{w \sim \mathcal{N}(0, I)}[f(\psi_\theta(w)) e^{Q(\psi_\theta(w))/\beta}] \\ &= Z^{-1} \mathbb{E}_{a \sim \pi_{base}}[f(a) e^{Q(a)/\beta}]. \end{aligned}$$

□

No invertibility is needed. This is the same target that KL-regularized fine-tuning of the whole flow aims for, but a small proposer can only approximate it as well as its distribution family and optimization allow. A single Gaussian, for example, need not represent a multimodal noise posterior. Figure 2 shows the target on a 2D example.

Proposition 2 (The noise-space KL is a conservative proxy). *By the data processing inequality,*

$$\text{KL}(\pi_\phi(\cdot | s) \parallel \pi_{base}(\cdot | s)) \leq \text{KL}(q_\phi(\cdot | s) \parallel \mathcal{N}(0, I)), \quad (9)$$

with equality when $\psi_\theta(s, \cdot)$ is invertible. The left side can be expensive to evaluate: a continuous normalizing flow can compute densities through divergence integration, while a generic distilled decoder need not offer a tractable density. For a Gaussian proposer $\mathcal{N}(\mu, \text{diag } \sigma^2)$, the right side is the familiar VAE term $\frac{1}{2} \sum_j (\mu_j^2 + \sigma_j^2 - 1 - \log \sigma_j^2)$. Noise PPO ([Miao et al., 2025](#)) uses exactly this KL penalty, together with an entropy bonus. Noise Hypernetworks and LP-DS ([Simsir & Oguz, 2026](#)) write the steered noise as a residual $w = \epsilon + \Delta$ and approximate the KL by $\frac{1}{2} \mathbb{E} \|\Delta\|^2$, a surrogate justified under regularity and small-Jacobian assumptions. Bounding the residual’s magnitude alone does not establish those assumptions.

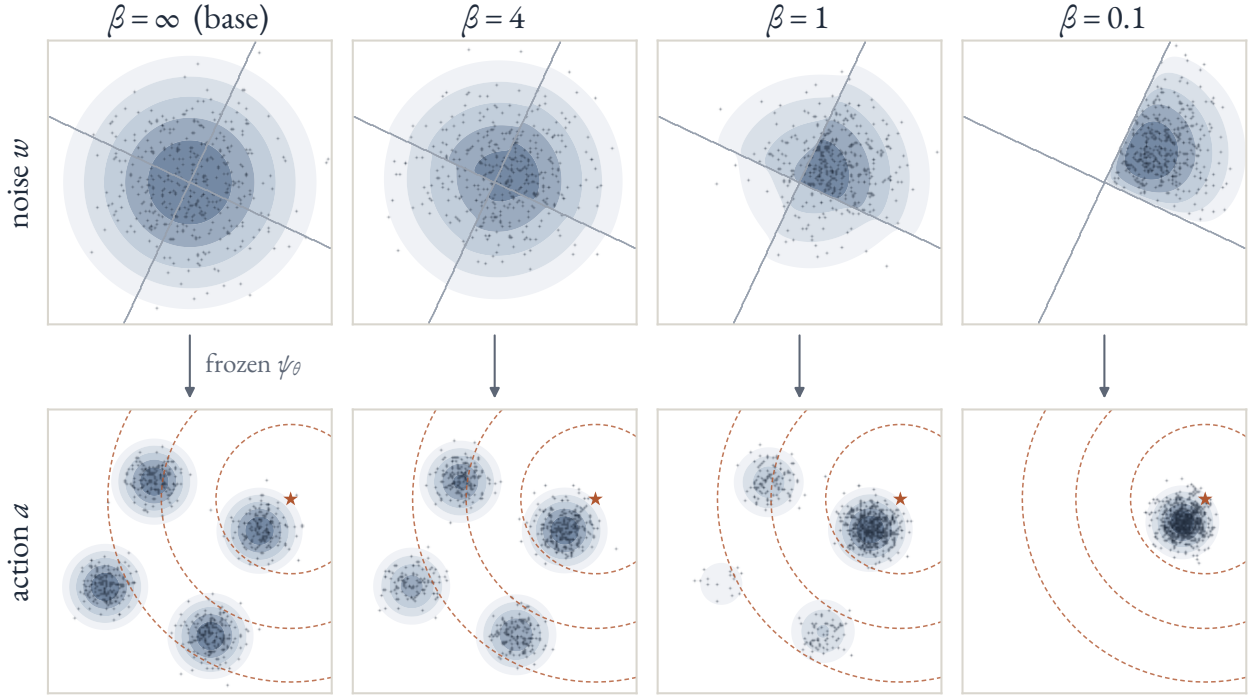


Figure 2: Tilting the noise tilts the policy (exact, no learning). The base policy is a four-cluster mixture in a 2D action space, and ψ is the exact flow-matching ODE map onto it. Each column is one value of β . Top: the tilted noise distribution $q^*(w) \propto \mathcal{N}(w; 0, I) e^{Q(\psi(w))/\beta}$ (shading) and samples from it (dots); grey lines split the noise plane into the four regions that decode to the four clusters. Bottom: the same samples after the frozen flow, over the tilted policy $\pi_{\text{base}} e^{Q/\beta}$ (shading); dashed circles are contours of Q , and the star marks its peak. At $\beta = \infty$ the base policy is recovered. As β shrinks, probability moves to the cluster nearest the peak, but no new cluster appears.

3 Where RL takes effect: carrying value into noise space

The decision variable for RL is now the noise. Much of the available supervision, however, lives in the other space. An action-space critic $Q(s, a)$ is ordinary temporal-difference learning on (s, a, r, s') transitions, and offline datasets, demonstrations, and human corrections all come labeled with actions, not noises. The proposer needs a value of w , or a gradient with respect to w , of the composition

$$w \mapsto Q(s, \psi_\theta(s, w)). \quad (10)$$

This simplifies the policy interface but not necessarily the optimization: nearby noises may have very different values. So the central design question of the family is how to **carry an action-space value back into noise space**. The papers suggest five routes, summarized in Figure 3. They are not disjoint categories: Route 4 is a special case of Route 3, and a method can combine inversion with a noise critic. What separates them is what each needs from the decoder: forward calls only, an inverse, or derivatives.

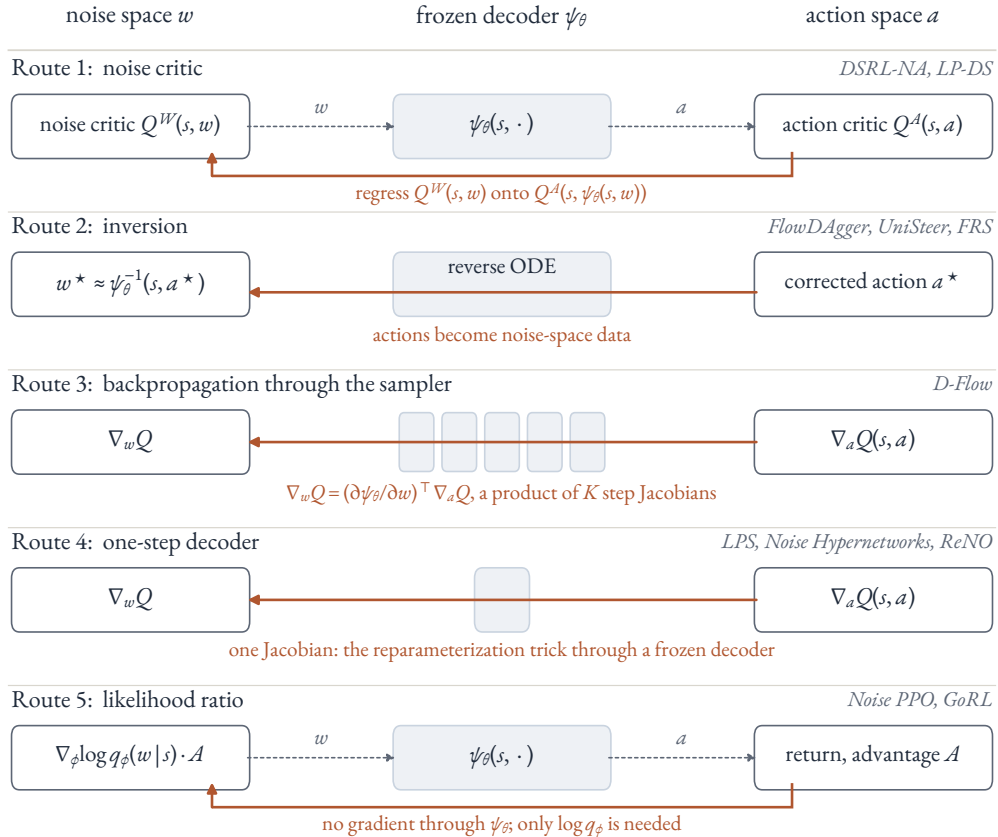


Figure 3: Five routes for training the noise proposer. In each row, noise space is on the left, the frozen decoder in the middle, and action space on the right. Grey dashed arrows are forward decoding; orange arrows show what crosses back into noise space. Route 1 is drawn in its two-critic form; Route 2 uses action supervision rather than a critic; Route 5 uses returns directly. Representative methods are listed at the right of each row.

Route 1: learn a critic in noise space. One option is to learn a noise-space critic directly from rollouts, as in DSRL-SAC. DSRL-NA (“noise-aliased”) instead keeps two critics. It learns $Q^A(s, a)$ by temporal-difference learning in action space, then distills it into a noise-space critic by regressing $Q^W(s, w)$ onto $Q^A(s, \psi_\theta(s, w))$ with $w \sim \mathcal{N}(0, I)$. The decoder is only ever run forward. Because many noises decode to similar actions, a noise that was never played can still receive an accurate value. On Robomimic Square, plain SAC in noise space (DSRL-SAC) needs about twice as many samples. On real robots, DSRL reports cases where success rose from 20% to 90% within fewer than 50 online episodes. LP-DS keeps the same two-critic design. A related forward-only idea appears in GoldenStart (Zhang et al., 2026), which picks the highest-value noise for each state by decoding candidates and scoring them with the critic, then fits a conditional VAE to these noises as a learned prior. GoldenStart is not a pure frozen-decoder method, though: its flow teacher keeps training, and the deployed policy is a distilled one-step student. *Cost*: a second approximation. LPS argues that a distilled critic can match values well and still point its gradient in the wrong direction.

Route 2: invert the flow and move the data. Sometimes the signal is an action rather than a reward: a human correction, or a coarse suggestion from a vision-language model. A flow can be integrated backwards to recover the noise that would have produced that action, $w^* \approx \psi_\theta^{-1}(s, a^*)$, and the corrected actions become noise-space training targets. A generic VAE decoder has no inverse, so it relies on a learned encoder for inference. An exact ODE flow is invertible under suitable regularity assumptions; a coarse numerical sampler or a distilled decoder need not be. FlowDagger (Murray et al., 2026) regresses the proposer onto inverted corrections. UniSteer (Lu et al., 2026) stores inverted transitions $(s, \hat{w}, r, s')$ in both a demonstration buffer and an RL buffer, and trains a noise-space critic on them by ordinary TD learning. Flow Reversal Steering (Tang et al., 2026) maps coarse reference actions to nearby modes of a generalist policy, then distills them or uses them to bootstrap DSRL. Inversion supplies a noise that reconstructs the correction, not a sample from the reward-tilted posterior above. *Cost*: inversion error. With the short ODE schedules of robot action heads, a naive reverse Euler step is not accurate enough, which is why FlowDagger and UniSteer both use per-step fixed-point refinement for their flow action heads. Convergence requires conditions such as a step size small relative to the velocity field’s Lipschitz constant; it is not automatic.

Route 3: backpropagate through the sampling iterations. The chain rule gives the noise gradient exactly:

$$\nabla_w Q(s, \psi_\theta(s, w)) = \left(\frac{\partial \psi_\theta}{\partial w} \right)^\top \nabla_a Q(s, a), \quad (11)$$

$$\frac{\partial \psi_\theta}{\partial w} = \prod_{k=K-1}^0 \left(I + \Delta t \frac{\partial v_\theta}{\partial x}(x_k, t_k, s) \right) \quad (12)$$

for the K -step Euler sampler of Section 1, with later-step Jacobians multiplying on the left. “Exact” here means the derivative of the implemented sampler and learned critic, not of the true environment return. With a reparameterized proposer $w = \mu_\phi(s) + \sigma_\phi(s) \odot \epsilon$, this is the VAE reparameterization trick with a frozen decoder, and no second critic is needed. D-Flow (Ben-Hamu et al., 2024) takes this route for per-sample noise optimization in image, audio, and molecule

generation. *Cost*: the gradient has to pass through every step of the solver. DSRL calls this “computationally intensive and numerically unstable”, and LPS calls it “expensive and unstable”.

Route 4: make the decoder one step. If ψ_θ is a single network evaluation, the product above has a single factor, and the critic gradient reaches the noise in one backward pass. LPS (Im et al., 2026) uses a one-step MeanFlow base policy for exactly this reason, and backpropagates $\nabla_a Q$ straight into the latent actor without a noise-space critic. It is a useful architectural neighbor rather than a strictly frozen-decoder method: its Algorithm 1 also updates the base policy by imitation during training. ReNO does the same with reward models on one-step image generators, and Noise Hypernetworks on distilled one- or few-step generators (SD-Turbo, SANA-Sprint, FLUX-Schnell). With a few steps the chain in Route 3 is short rather than trivial, which is still cheap. *Cost*: the base policy must be one-step, which usually means it was trained or distilled for that purpose.

Route 5: skip the transfer. The likelihood-ratio gradient of the return term, $\mathbb{E}[\nabla_\phi \log q_\phi(w | s) A(s, w)]$ with an advantage estimate A , needs only $\log q_\phi$, which is exact for a Gaussian proposer: no ODE divergence, no SDE conversion, no surrogate likelihood. Much of the survey’s online-RL section is about recovering a usable likelihood for flow policies; in noise space the problem does not arise. Noise PPO treats text-to-image generation as a one-step MDP whose action is the initial noise, and runs PPO with sampled rewards; KL and entropy regularizers contribute additional gradient terms. *Cost*: the usual variance of score-function estimates.

4 A two-dimensional experiment

To see the five routes at work, we train the same noise proposer with each of them on a toy problem small enough to plot and to evaluate exactly. Code for this section and all figures is available [on GitHub](#).

Setup. Actions are points $a \in \mathbb{R}^2$. The base policy imitates the four-cluster mixture of Figure 2: a small velocity network (an MLP) is trained once by conditional flow matching and then frozen, and the decoder ψ_θ is its $K = 10$ -step Euler sampler. The pushforward of $\mathcal{N}(0, I)$ through it puts between 24.9% and 25.1% of its mass on each cluster. A state is a goal angle $s \in [0, 2\pi)$, and the critic is known exactly: $Q(s, a) = -\frac{1}{2}\|a - g(s)\|^2$, where the goal $g(s)$ lies on the circle through the cluster centers. With $\beta = 1$, a goal between two clusters makes q^* bimodal. The proposer is a Gaussian $\mathcal{N}(\mu_\phi(s), \text{diag } \sigma_\phi(s)^2)$ whose mean and log standard deviation come from a small MLP, initialized at the prior $\mathcal{N}(0, I)$.

Because Q is given, the experiment isolates the transfer step: there is no TD learning and no exploration. The ruler from Section 2 measures the result: $\text{KL}(q_\phi \| q^*) = \log Z(s) - J(\phi; s) / \beta$, estimated by Monte Carlo and averaged over 64 evenly spaced states.

How the noise moves. Figure 4 follows one state during training with Route 3. The proposer starts as the prior, so the actions are the base policy. Within a few dozen updates, the noise cloud

shrinks and slides into the region of noise space that decodes to the cluster nearest the goal; the actions follow it without any change to the decoder.

In this state the goal lies between two clusters, so the target policy is bimodal. One Gaussian is nonetheless enough to reach both: the two clusters decode from adjacent regions of noise space, and the proposer settles across their shared boundary. **A distribution that is bimodal in action space can be unimodal in noise space.** What the Gaussian cannot match is the shape of q^* , which is cut off sharply along the grey lines; a few of its samples also still decode to the two far clusters. The remaining $\text{KL}(q_\phi \| q^*)$ at this state is about 0.17 nats.

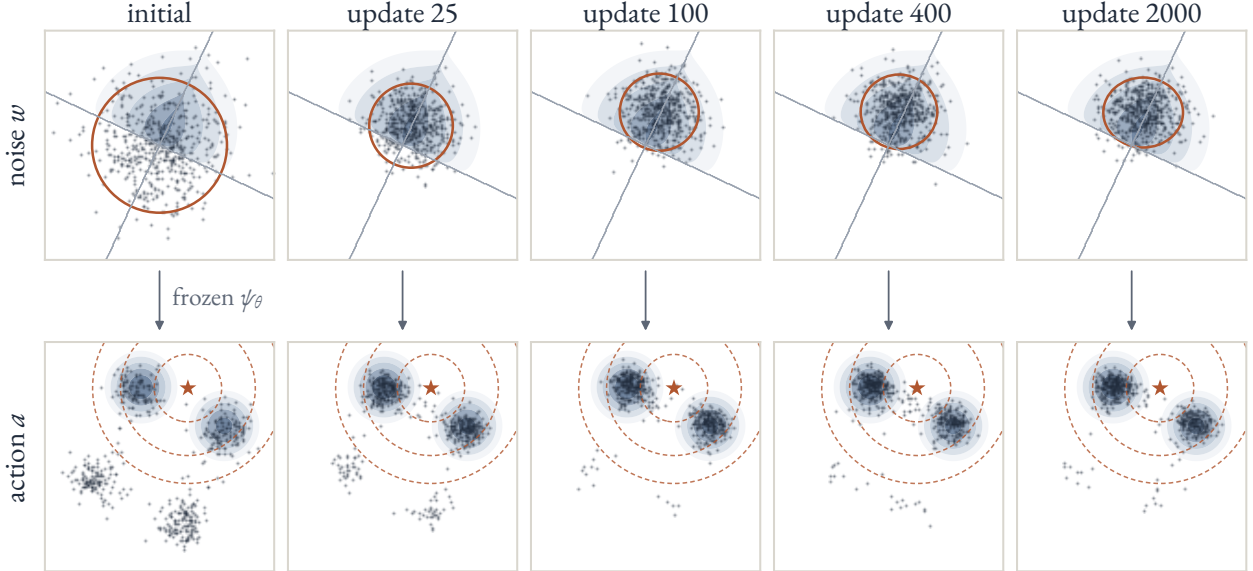


Figure 4: Learning which noise to choose, for one state (goal angle 67.5°) with Route 3. Columns are snapshots during training. Top: samples from the proposer $q_\phi(w | s)$ (dots) and its two-standard-deviation ellipse (orange), over the exact tilted posterior q^* for the learned decoder (shading); grey lines separate the noise regions that decode to different clusters. Bottom: the same samples after the frozen 10-step decoder, over the tilted target policy $\pi_{\text{base}} e^{Q/\beta}$ (shading, computed from the mixture the flow imitates); the star is the goal and dashed circles are contours of Q . The decoder never changes; only the noise does.

Comparing the routes. Each route trains the same proposer for 2,000 updates, each on 1,024 decoded noises (256 states with 4 noises each), and is run with three seeds. Routes 1, 3, 4, and 5 maximize the objective J of Section 2, using a distilled noise critic, gradients through the 10 Euler steps, gradients through a one-step decoder distilled from the 10-step sampler, and a likelihood-ratio gradient with a leave-one-out baseline. Route 2 never sees Q . It receives “corrected” actions sampled from the tilted target policy, a stand-in for an expert, inverts them through the decoder by fixed-point iteration, and fits the proposer to the recovered noises by maximum likelihood. Table 3 and Figure 5 summarize the outcome.

Three observations follow.

- **The four routes that maximize J end in the same place.** Routes 1, 3, 4, and 5 reach 0.206–0.211 nats, against 0.198 for the best single Gaussian fitted to each state separately. In this toy the remaining error comes from the Gaussian family, not from the transfer route, and

Table 3: Five routes on the 2D toy. $\text{KL}(q_\phi \| q^*)$ is averaged over 64 states (mean $\pm$ standard deviation over three seeds; lower is better). Time is the median wall-clock time per update on a laptop GPU (RTX 5060), including Python overhead. Route 4 is measured against the posterior of its own one-step decoder.

Route	Needs from the decoder	$\text{KL}(q_\phi \ q^*)$	ms / update
1. noise critic	forward calls	0.207 ± 0.004	8.6
2. inversion	an inverse (20 fixed-point iterations per step)	0.312 ± 0.007	60.2
3. backprop through 10 steps	derivatives	0.211 ± 0.002	7.2
4. one-step decoder	derivatives of a distilled one-step student	0.206 ± 0.002	2.9
5. likelihood ratio	forward calls	0.211 ± 0.003	5.6
reference: best Gaussian for each state separately	derivatives	0.198	—

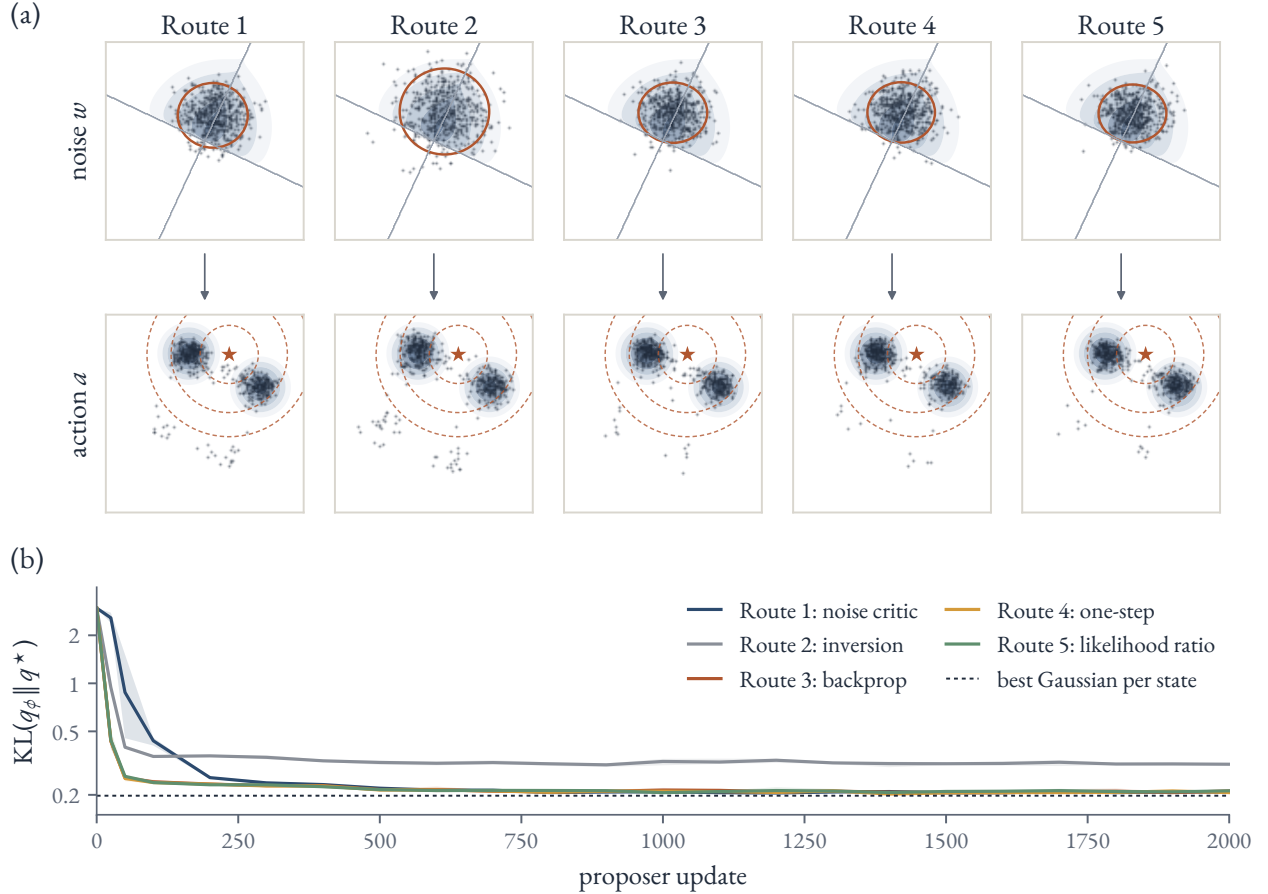


Figure 5: Five routes, one proposer. (a) The final proposer of each route for the same state as Figure 4, drawn as in Figure 4 (seed 0). (b) $\text{KL}(q_\phi \| q^*)$ averaged over 64 states during training; lines are means and bands span three seeds. The dashed line is the best single Gaussian fitted to each state separately.

amortizing over states costs little.

- **They differ in how they get there.** Route 1 starts slowly, because its noise critic has to be learned before its gradient means anything (Figure 5b). The cost per update spans a factor of twenty: 2.9 ms through the one-step decoder, 7.2 ms through ten Euler steps, and 60 ms for inversion.
- **Route 2 settles higher, and not because inversion fails.** Fixed-point inversion reconstructs the corrected actions to a median error of about 10^{-7} ; a single naive reverse Euler pass per step leaves a median error of 0.05, about an eighth of a cluster’s standard deviation (0.38). The difference lies in the objective. Maximum likelihood on samples from the target minimizes $\text{KL}(q^* \| q_\phi)$, which favors a wider Gaussian that covers the posterior (Figure 5a), while the ruler measures $\text{KL}(q_\phi \| q^*)$. Corrections teach the proposer to imitate the target, not to optimize the objective.

These numbers come from a deliberately easy setting: two dimensions, an exact critic, and a smooth decoder. They say nothing about critic error, about the variance of Route 5 in high dimensions, or about the instability that DSRL and LPS report for Route 3 with large decoders. What the toy does show is the shape of the problem: the decoder fixes which actions are reachable, the proposer only decides how much noise to send to each region, and a single Gaussian must choose between regions when the target has several.

5 A map of the family

Table 4 groups the methods by their main learning route. It includes related designs with changing decoders, labeled explicitly. “Prior control” distinguishes actual constraints from initialization or supervision alone.

ReFORM spans Routes 3 and 4: its original objective differentiates through the BC flow, and its practical design introduces one-step BC distillation to shorten this path. Its Algorithm 1 also continues fitting the BC model. D-Flow is a per-sample example of Route 3.

Three neighboring lines of work are relevant. PDP (Zhang et al., 2026) argues that Gaussian noise is a poorly conditioned control interface. Instead of steering it, PDP trains a literal VAE over whole demonstrations, with a geometry-alignment term, and modulates the denoiser with the resulting low-dimensional code. Condition-dependent source flow matching (Kim et al., 2026) and LAFM (Machado et al., 2026) learn the prior itself together with the flow at training time, which in VAE terms is a learned prior. D-Flow (Ben-Hamu et al., 2024) and ReNO (Eyring et al., 2024) optimize the noise separately for each sample instead of training a proposer.

Table 4: Noise-space methods and close relatives, grouped by the route through which value reaches the noise. “Prior control” distinguishes actual constraints from initialization or supervision alone.

Route	Method	Setting	Proposer $q_\phi(w s)$	Prior control
1. noise critic	PARROT (2021; preprint 2020)	robot manipulation	RL policy over the noise of an invertible normalizing flow	prior initialization; no lasting proximity guarantee
1. noise critic	DSRL (2025)	online and offline control; π_0	state-conditioned noise actor; SAC, or an action critic distilled into a noise critic	noise box $[-b, b]^d$
1. noise critic	LP-DS (2026)	online control; π_0	residual $\epsilon + \Delta_\phi(s, \epsilon)$ with noise-aliased critics	Lagrangian trust region on $\mathbb{E}\ \Delta\ ^2$
1. forward scoring	GoldenStart (2026)	distilled one-step flow policies; teacher not frozen	conditional VAE fit to the highest-value noise per state	CVAE fitting; entropy control is on the action student
2. inversion	FlowDagger , UniSteer (2026)	human-in-the-loop VLA adaptation	noise policy regressed onto inverted corrections; UniSteer adds a noise critic	inverted targets; inversion alone does not ensure typical noise
2. inversion	Flow Reversal Steering (2026)	generalist policies	auxiliary noise policy trained on inverted coarse actions	coarse references guide reversal; no generic KL bound
3 / 4. back-propagation	ReFORM (2026)	offline RL; BC model and distilled decoder also trained	reflected flow generating bounded input noise	projected steps keep noise inside the source ball
4. one-step	Noise Hypernetworks (2025)	distilled text-to-image	residual on fresh noise (LoRA), reward gradient through the generator	$\frac{1}{2}\ \Delta\ ^2$ as a KL surrogate
4. one-step	LPS (2026)	offline RL; real robots; base also trained by imitation	latent actor on a sphere, $\nabla_a Q$ through a one-step MeanFlow	sphere of radius $\sqrt{d}$
5. likelihood ratio	Noise PPO (2025)	text-to-image	prompt-conditioned Gaussian trained by PPO	KL to $\mathcal{N}(0, I)$, entropy bonus
5. likelihood ratio	GoRL (2025)	online control	Gaussian encoder; decoder refit in alternating phases	KL to $\mathcal{N}(0, I)$
search	Golden Ticket (2026)	generative robot policies	one constant noise vector found by search	—

6 What the VAE view explains

The decoder fixes the map, not a safety guarantee. With a frozen decoder, every steered action lies in the image of $\psi_\theta(s, \cdot)$. For finite Q and positive β , the ideal tilted policy has the same support as the base policy. But support is not the same as likely or reliable behavior: a Gaussian-driven invertible flow can have full support, including actions far from the demonstrations. ReFORM makes a stronger construction by bounding the source support; relating its learned model’s support to the dataset still depends on how well that model fits. A truly unreachable action cannot be recovered by steering, while a very unlikely action might be. GoRL (Zhang et al., 2025) changes the map itself by training the decoder again. Its decoder starts as an identity-like map. Training then alternates between two phases: optimizing a Gaussian encoder with a KL penalty toward $\mathcal{N}(0, I)$ while the decoder is frozen, and refitting the decoder by flow matching on recent rollouts while the encoder is frozen.

β is a dial between the base policy and high-value actions. For a fixed, well-behaved Q , as $\beta \rightarrow \infty$ the ideal proposer approaches the prior and the system returns to the base policy. This resembles posterior collapse in a VAE, though here returning to the base can be intentional. As $\beta \rightarrow 0$, mass concentrates near maximizers, when they exist. Multiple equally good actions—or multiple noises for one action—can remain; a single point mass is not inevitable. LP-DS reports that prior-replacing steering can lose modes, and uses a residual trust region whose multiplier plays a role similar to β , for its surrogate penalty. Golden Ticket (Patil et al., 2026) imposes a different restriction: one noise vector shared across states. It is not the general low-temperature limit of a state-conditioned proposer.

Holes in the latent space. A VAE can decode poorly in regions its training rarely visits. A Gaussian-driven flow can have the same problem: most training draws come from typical regions of the Gaussian. The family uses several ways to discourage atypical queries, with different strengths. DSRL bounds each noise coordinate in $[-b, b]$ and reports that b between 1 and 3 works well online. A box bounds coordinates; it does not by itself enforce the Gaussian’s typical radius. LPS trains both its base policy and its latent actor on the sphere of radius $\sqrt{d}$, where high-dimensional Gaussian mass concentrates, to prevent what it calls “norm explosion”; strictly, its prior is uniform on that sphere rather than $\mathcal{N}(0, I)$. LP-DS bounds the squared norm of its residual, its stand-in for the KL.

Aliasing makes some noise changes redundant. Many noises can decode to similar actions. This is related to latent ambiguity, but distinct from the statistical non-identifiability of a VAE’s latent coordinates. DSRL-NA turns this weakness into data sharing across noises. The same fact explains why the noise KL is only an upper bound: when the decoder is genuinely many-to-one, part of the KL budget can be spent on redistributing noise among identical outputs. For an invertible flow the KLs remain equal, even if different noises produce very similar actions.

Amortization has a gap. Optimizing noise separately for each sample (D-Flow, ReNO) and training a proposer that predicts it (Noise Hypernetworks, whose subtitle is “Amortizing Test-Time

Compute”) are the policy-side versions of per-datapoint variational inference and an amortized encoder. The cost of amortization has a name in the VAE literature, the amortization gap (Cremer et al., 2018). Noise Hypernetworks report recovering a substantial portion of the gains of explicit test-time optimization at a fraction of its cost, which illustrates the quality–compute tradeoff. That comparison is not, by itself, a measurement of the formal variational amortization gap.

7 Open questions

- **When is a policy steerable?** DSRL notes that the base policy must be “steerable” for the method to work and studies when this holds empirically, but does not give a formal definition. Nothing in the construction guarantees useful steering in advance. It helps to separate three questions: can the decoder produce a good action, does the allowed proposer put enough probability on its preimages, and can the learning signal find them? Invertibility addresses only the first.
- **How does the number of sampling steps matter?** Noise PPO finds the largest gains at few inference steps; the benefit shrinks, but does not disappear, as the number of steps grows. Several robot papers rely on one-step or few-step decoders for different reasons: cheap Jacobians for LPS, low latency for robot control. Few steps are not free, however: FlowDagger finds that coarse steps make naive inversion inaccurate.
- **When should the decoder be trained?** Freezing the decoder preserves the learned map and can improve sample efficiency; training it, as in GoRL, can change which useful behaviors are easy to represent. Where a pretrained policy should sit between the two is not settled.

A practical reading rule. If the decoder is available only through forward calls, start by considering a noise critic or a likelihood-ratio update. If corrections are available, inversion may turn them into direct supervision. If gradients are available and the sampler is short, direct backpropagation avoids critic distillation. Then check the paper’s actual prior constraint and whether the decoder stays frozen; the shared diagram does not settle either question.

The survey’s second axis asks where the RL signal enters a flow-matching method. This family adds one more answer to that question: **the RL signal can enter through the input-noise distribution.**

References

- Ben-Hamu, H., Puny, O., et al. (2024). [D-Flow: Differentiating through Flows for Controlled Generation](#). *ICML*.
- Cremer, C., Li, X., & Duvenaud, D. (2018). [Inference Suboptimality in Variational Autoencoders](#). *ICML*.
- Eyring, L., Karthik, S., Roth, K., Dosovitskiy, A., & Akata, Z. (2024). [ReNO: Enhancing One-step Text-to-Image Models through Reward-based Noise Optimization](#). *NeurIPS*.
- Eyring, L., Karthik, S., Dosovitskiy, A., Ruiz, N., & Akata, Z. (2025). [Noise Hypernetworks: Amortizing Test-Time Compute in Diffusion Models](#).

- Im, H., Kolobov, A., Fu, J., & Lee, Y. (2026). Latent Policy Steering through One-Step Flow Policies. *RSS. (LPS)*
- Kim, J., Park, J., Jeon, S., & Kim, S. (2026). Better Source, Better Flow: Learning Condition-Dependent Source Distribution for Flow Matching.
- Kingma, D. P., & Welling, M. (2014). Auto-Encoding Variational Bayes. *ICLR*.
- Levine, S. (2018). Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review.
- Lu, J., Qin, X., et al. (2026). UniSteer: Unified Noise Steering for Efficient Human-Guided VLA Adaptation.
- Machado, B., Chapin, A., Dellandréa, E., & Chen, L. (2026). Flowing With Purpose: Latent Action Guided Flow Matching Policies for Robotic Manipulation. (LAFM)
- Miao, Y., Loh, W., Poupart, P., & Kothawade, S. (2025). A Minimalist Method for Fine-tuning Text-to-Image Diffusion Models. (Noise PPO)
- Murray, M., Chen, D., Bagaria, S., et al. (2026). FlowDagger: Human-in-the-Loop Adaptation of Generative Robot Policies in Latent Space.
- Nahim, N. Z., Khan, F. N., & Ju, P. (2026). A Survey of Flow Matching in Reinforcement Learning. *TMLR*.
- Patil, O., Biza, O., Weng, T., et al. (2026). You’ve Got a Golden Ticket: Improving Generative Robot Policies With A Single Noise Vector.
- Simsir, H., & Oguz, O. S. (2026). Lagrangian Perturbation Diffusion Steering: Latent Reinforcement Learning for Generative Policies. *ICML. (LP-DS)*
- Singh, A., Liu, H., Zhou, G., Yu, A., Rhinehart, N., & Levine, S. (2021). PARROT: Data-Driven Behavioral Priors for Reinforcement Learning. *ICLR*.
- Tang, A., Chen, W., Wagenmaker, A., Finn, C., & Levine, S. (2026). Improving Robotic Generalist Policies via Flow Reversal Steering.
- Wagenmaker, A., Nakamoto, M., Zhang, Y., et al. (2025). Steering Your Diffusion Policy with Latent Space Reinforcement Learning. (DSRL)
- Zhang, C., Wan, Z., Chen, F., Yu, X., Tsang, I., & An, B. (2025). GoRL: An Algorithm-Agnostic Framework for Online Reinforcement Learning with Generative Policies. The current arXiv version is titled *Training Diffusion Policies via Prior-Mapping Co-Evolution*; this post follows v1.
- Zhang, H., Sun, Y., & Xiong, H. (2026). GoldenStart: Q-Guided Priors and Entropy Control for Distilling Flow Policies. *ICLR*.
- Zhang, R., Fu, H., Jia, M., Konidaris, G., Du, Y., & da Silva, B. C. (2026). From Noise to Control: Parameterized Diffusion Policies. *ICML. (PDP)*
- Zhang, S., So, O., Ahmad, H. M. S., Yu, E. Y., Cleaveland, M., Black, M., & Fan, C. (2026). ReFORM: Reflected Flows for On-Support Offline RL via Noise Manipulation. *ICLR*.
- Zhou, W., Bajracharya, S., & Held, D. (2020). PLAS: Latent Action Space for Offline Reinforcement Learning. *CoRL*.

How to cite

If you find this note useful in your work, you can cite it as:

Chi, Yuhan. “Choose the Noise, Keep the Flow.” Blog post, October 18, 2026. <https://yuhanchi.github.io/writing/choose-the-noise/>

or with the following BibTeX entry:

```
@misc{chi2026choosethenoise,  
  author = {Chi, Yuhan},  
  title = {{Choose the Noise, Keep the Flow}},  
  year = {2026},  
  month = oct,  
  day = {18},  
  howpublished = {Blog post},  
  url = {https://yuhanchi.github.io/writing/choose-the-noise/}  
}
```

This citation refers to the explanation and synthesis above; please cite the original papers for their methods and results.