Neural Bayesian Filtering

Christopher Solinas[1, †], Radovan Haluška[3], David Sychrovský[3], Finbarr Timbers, Nolan Bard[5], Michael Buro[1], Martin Schmid[3,4], Nathan R. Sturtevant[1,2], Michael Bowling[1,2]
[1]University of Alberta, [2]Alberta Machine Intelligence Institute (Amii)
[3]Charles University, Prague, [4]EquiLibre Technologies, Inc.,
[5]Sony Corporate Technology Center America, Inc
{[†]solinas,mburo,nathanst,mbowling}@ualberta.ca,
{haluskr,sychrovsky,schmidm}@kam.mff.cuni.cz, nolan.bard@sony.com

Abstract

Sequential estimation under partial observability requires tracking beliefs that may be high-dimensional, multimodal, and non-Gaussian. Classical Bayesian filters generalize zero-shot to any system whose dynamics can be evaluated, but their representations scale poorly: parametric filters struggle to capture multimodality, and particle filters require exponentially many particles in the state dimension. *Generative Distribution Embeddings* (GDEs) learn compact representations of complex distributions but have no sequential update. We present *Neural Bayesian Filtering* (NBF), which represents beliefs as embeddings and approximates their Bayesian update. At each step, NBF samples particles from a learned conditional generator, propagates them through the dynamics, and embeds the propagated set, weighted by the likelihood of the new observation. Regenerating the set from the embedding at each step, rather than resampling a surviving pool, mitigates the risk of *impoverishment* that degrades particle filters. The system dynamics and likelihood enter only through propagation and weighting, so NBF retains zero-shot adaptability to new dynamics as long as the beliefs they produce fall within the family the embedding was trained on. Training the GDE to represent that family requires only the states realized at training time. We validate NBF on Lorenz-96 and a pursuit–evasion domain, showing graceful scaling with state dimension, accurate tracking of multimodal posteriors, and zero-shot generalization to dynamics and likelihoods held out from training.

1 Introduction

Tracking the evolution of probability distributions over time is a fundamental problem across science and engineering. Applications range from state estimation and sequential decision-making under partial observability [Kaelbling et al., 1998] to tracking population-level responses to perturbation [Haug, 2012, Barfoot, 2024, Särkkä and Svensson, 2023]. When the underlying distributions are high-dimensional, multimodal, or non-Gaussian, maintaining accurate estimates is challenging.

Classical filtering methods fall into two broad categories. Parametric filters such as the Kalman filter [Kalman, 1960], its nonlinear extensions [Sorenson, 1966, Julier and Uhlmann, 2004, Arasaratnam and Haykin, 2009], and the ensemble Kalman filter [Evensen, 1994] maintain a Gaussian posterior, yielding efficient updates that scale gracefully with state dimension. However, their distributional assumptions are violated when posteriors are multimodal or non-Gaussian. Sample-based filters such as particle filters [Doucet and Johansen, 2011] can, in principle, represent arbitrary distributions through weighted particle sets, but the number of particles required for accurate approximation grows rapidly with the dimensionality of the state space. Particle filters also suffer from *impoverishment*: repeated resampling duplicates high-weight particles while discarding the rest, so the particle set

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

progressively collapses to a few ancestral samples and loses distributional coverage [Sokota et al., 2022]. Neither class satisfactorily handles distributions that are simultaneously complex and high-dimensional, motivating approaches that learn and maintain compact distributional representations.

Recent work on *Generative Distribution Embeddings* (GDEs) shows that learned, distributionally invariant encoders coupled with conditional generators can represent complex distribution families [Fishman et al., 2025]. In principle, these embeddings are predictive sufficient statistics, capturing all information needed for downstream tasks. Through an associated generative model, they enable sampling (and density estimation, in some cases) on families of distributions, assuming only finite sample access to each distribution. Critically, GDEs are static: embeddings represent single distributions, with no mechanism for sequential updates between them.

We combine classical filters and GDEs in an algorithm, which we call *Neural Bayesian Filtering* (NBF), that runs Bayesian updates on particles drawn from the GDE's generator. Given an embedding, the method (i) generates particles from the conditional generator, (ii) propagates each through the system dynamics, (iii) weights them by observation likelihood, and (iv) embeds the resulting weighted set as a representation of the posterior. The dynamics and likelihood enter only at steps (ii) and (iii), so neither the embedding nor the generator depends on them, and NBF adapts to new dynamics without retraining, provided the beliefs those dynamics induce are ones the GDE was trained to represent. Step (i) also draws a fresh set from the embedding at every update, so NBF never resamples and the collapse described above cannot accumulate over the steps of an episode. The GDE is trained on the particle sets NBF itself produces, with the realized state as the only target, so training never requires samples from the beliefs themselves.

NBF is the first method to combine the representational benefits of distribution embeddings with the generalization properties of classical Bayesian filters. Experiments on Lorenz-96 and a pursuit–evasion domain verify four properties: (1) graceful scaling with state dimension, (2) faithful representation of multimodal, non-Gaussian posteriors, (3) native parametric density estimation, and (4) zero-shot generalization to dynamics and observation likelihoods held out from training. We also give a finite-horizon consistency result for NBF, showing that its estimates converge almost surely to the true expectation under standard particle-filter assumptions and an exact embedding.

2 Background

2.1 Bayesian Filtering

An environment $G \stackrel{\text{def}}{=} (X, Y, T, \mathcal{Y}, b_0)$ specifies a state space X, an observation space Y, a transition function T, an observation function $\mathcal{Y}$, and an initial state distribution b_0. A policy $\pi(a \mid x)$ draws an action a from an action space A in each state, and the transition function $T(x' \mid x, a)$ conditions on that action. The filter does not observe the actions, so it marginalizes them, and we overload T for the transition under the policy, $T(x' \mid x, \pi) \stackrel{\text{def}}{=} \int_A \pi(a \mid x)\, T(x' \mid x, a)\, da$.

A *belief state* $b(x)$ is the posterior over the Markov state $x \in X$ given the observations so far. Given b_t and a new observation $y \in Y$, the Bayesian update is

$$b_{t+1}(x') \propto \int_X b_t(x)\, K_y(x'|x)\, dx, \qquad K_y(x'|x) \stackrel{\text{def}}{=} T(x' \mid x, \pi)\, \mathcal{Y}(y \mid x, x'), \tag{1}$$

where K_y is the kernel induced by the transition under π and the observation function, and normalization is over $x' \in X$. This update is exact but requires integrating over X, which is generally intractable when the state space is large or continuous. Classical filters approximate it with a fixed representation of the belief and evaluate T and $\mathcal{Y}$ inside the update, so they apply unchanged to any system whose dynamics can be evaluated.

Kalman filters [Kalman, 1960] provide closed-form updates for linear Gaussian dynamics; nonlinear variants (Extended [Sorenson, 1966], Unscented [Julier and Uhlmann, 2004], Cubature [Arasaratnam and Haykin, 2009]) extend the update to mild nonlinearities, and ensemble variants [Evensen, 1994, Burgers et al., 1998] estimate the covariance from an ensemble of states, which scales it to the high-dimensional systems of data assimilation. Every member of the family propagates a single Gaussian. Gaussian sum filters [Alspach and Sorenson, 1972] propagate a mixture of them instead, which recovers separated modes but fits a Gaussian to each.

Particle filters [Doucet and Johansen, 2011] approximate arbitrary distributions through weighted sample sets, but scale poorly with state dimensionality [Snyder et al., 2008]. They also suffer from *impoverishment*: repeated resampling concentrates the weight onto a few ancestors, so the empirical distribution loses support that the true posterior still has [Sokota et al., 2022]. Regularization, MCMC moves, and better proposals [Musso et al., 2001, Gilks and Berzuini, 2001, Doucet and Johansen, 2011] slow the loss but keep the surviving pool, so the loss still compounds over an episode.

We consider the problem of sequentially tracking beliefs $b_t(x)$ as observations arrive. At *training time* we assume access to G and to a set of policies $\{\pi_1, \ldots, \pi_J\}$, and the encoder and generator are fit to the beliefs these induce. In a system without actions, a parameter of T that varies between episodes takes the place of the policy, such as the forcing of Lorenz-96 in Section 5. At *test time* we are given G and a single policy π, and the filter evaluates $T(\cdot \mid \cdot, \pi)$ and $\mathcal{Y}$ directly, so no part of the dynamics is learned.

2.2 Distribution Embeddings

Generative Distribution Embeddings (GDEs) provide a principled framework for representing families of distributions via learned encoders and conditional generators [Fishman et al., 2025]. Consider a family of distributions $\mathcal{P}$ over a sample space, accessed through finite sample sets $x_{1:n} \sim p$ for $p \in \mathcal{P}$. An encoder $\mathcal{E}_\phi$ is *distributionally invariant* if it depends on the sample set only through its empirical distribution, so reordering the set or duplicating every sample leaves its output unchanged. Mean-pooled architectures (DeepSets, Set Transformers without positional encoding) satisfy this property [Fishman et al., 2025]. A conditional generator $\mathcal{G}_\psi$ takes an embedding θ and produces samples approximating the embedded distribution.

Two properties that Fishman et al. [2025] establish under regularity conditions make GDEs suitable for filtering. First, distributionally invariant embeddings are asymptotically *predictive sufficient statistics*: as the sample set grows, the embedding $\mathcal{E}_\phi(x_{1:n})$ retains all information in the set relevant for predicting new samples from the underlying distribution. This justifies treating it as a faithful surrogate for the beliefs when propagating dynamics and conditioning on observations. Second, if the encoder–generator pair can represent the target distribution p exactly, i.e. $\mathcal{G}_\psi(\mathcal{E}_\phi(p)) = p$, and $\mathcal{G}_\psi \circ \mathcal{E}_\phi$ is weakly continuous at p, then the plug-in $\mathcal{G}_\psi(\mathcal{E}_\phi(p_n))$ formed from the empirical measure p_n of n i.i.d. samples converges weakly to p as $n \to \infty$. Together these properties say that a well-trained GDE is a consistent, lossless surrogate for the underlying distribution, with error vanishing as the provided sample set size increases. Distribution embeddings, however, have no built-in sequential update.

3 Embedding Belief States

GDEs pair a distributionally invariant encoder with a conditional generator (Section 2). This section instantiates one for beliefs: the encoder maps a weighted particle set representing a belief to an embedding, and the conditional generator turns that embedding into a tractable approximation of the belief. The generator must support sampling, which the filtering update of Section 4 requires at every step, and density evaluation, which scores beliefs against the realized state on held-out data.

Encoder. Let $x_{1:n} \stackrel{\text{def}}{=} (x_1, x_2, \ldots, x_n)$ be a set of particles with weights $w_{1:n}$ that represents a belief $b(x)$. The encoder $\mathcal{E}_\phi : \bigcup_{n \geq 1} X^n \times \mathbb{R}^n \to \mathbb{R}^{d_{\text{emb}}}$ is distributionally invariant and maps the weighted set to a d_{emb}-dimensional embedding $\theta \stackrel{\text{def}}{=} \mathcal{E}_\phi(x_{1:n}, w_{1:n})$. Together with the conditional generator described next, θ defines the approximation $b_\theta(x)$.

Our $\mathcal{E}_\phi$ is a multi-round Set Transformer: each round applies an Induced Set Attention Block (ISAB) [Lee et al., 2019] followed by a pool over the particle set. The pooled output is concatenated to every particle's input for the next round, so each round conditions on the summary of the round before it. The weight of a particle enters the encoder alongside its state, so the embedding is a function of the weighted set. Appendix C describes the encoders we use in Section 5 in more detail.

Conditional generator. We use Flow Matching [Lipman et al., 2022], which supports both sampling and density evaluation, and write $\mathcal{G}_\psi(\theta) \stackrel{\text{def}}{=} b_\theta(x)$ for the conditional generator viewed as a map from

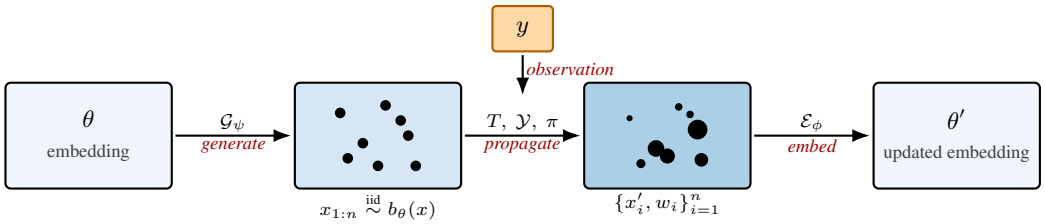

Figure 1: NBF update. The conditional generator draws n particles from θ; each is simulated through $T(\cdot \mid x, \pi)$ and reweighted by $\mathcal{Y}(y \mid x, x')$; the weighted set is embedded into the next embedding θ'.

an embedding to its induced distribution. A flow-matching model learns a vector field $v_\psi(z, \tau; \theta)$, indexed by an integration time $\tau \in [0, 1]$, that defines a continuous transport from the base $p(z)$ at $\tau = 0$ to the target at $\tau = 1$ [Lipman et al., 2022]. Sampling draws $z \sim p(z)$ and integrates the ODE $\dot{z}_\tau = v_\psi(z_\tau, \tau; \theta)$ from $\tau = 0$ to $\tau = 1$, returning $x \overset{\text{def}}{=} z_1$ as the generated sample. With the optimal-transport interpolant $z_\tau = (1 - \tau)\, z + \tau\, x$ between $z \sim p(z)$ and $x \sim b(x)$, the conditional flow-matching loss is

$$\mathcal{L}_{\text{FM}}(\phi, \psi) = \mathbb{E}_{\tau,z,x} \| v_\psi(z_\tau, \tau; \theta) - (x - z) \|^2 .$$

The density at x follows from integrating the divergence of v_ψ along the same trajectory [Lipman et al., 2022]. The encoder and generator are trained jointly on the realized states seen during training (Section 4.1).

4 Filtering with Embeddings

Section 3 embeds a single belief as $\theta = \mathcal{E}_\phi(x_{1:n}, w_{1:n})$ and decodes it by sampling from the conditional generator. Filtering needs an update that maps an embedding and a new observation to the embedding of the posterior, and it needs that update to hold for any dynamics the simulator can evaluate. The second requirement rules out a purely parametric update that maps (θ, y) to θ': the transition and observation models are not among its inputs, so its dependence on them is fixed at training time and the filter is tied to the dynamics it was trained on. Classical filters meet both requirements by evaluating T and $\mathcal{Y}$ inside the update, and NBF does the same by running the update on particles drawn from the embedding.

A single update draws n particles from $\mathcal{G}_\psi(\theta)$, propagates each through $T(\cdot \mid x, \pi)$, weights each by $\mathcal{Y}(y \mid x, x')$, and embeds the weighted set as θ'. Figure 1 sketches the loop, and Algorithm 1 gives the procedure in full. Where a particle filter resamples the weighted set, NBF embeds it, and the next set is drawn fresh from θ', so no particle survives from one step to the next. In data-assimilation terms, propagation is the forecast step, and weighting and embedding together form the analysis step, whose output is the embedding rather than an analysis ensemble. Neither the encoder nor the generator is a function of T, $\mathcal{Y}$, or π, so the trained model is unchanged when any of them changes.

Proposals. NBF admits the same choice of proposal as a particle filter. Particles can be propagated through any proposal kernel $\tilde{T}(x' \mid x, y)$ in place of the transition function, so the pairs (x, x') are drawn from the proposal $\mathcal{G}_\psi(\theta) \cdot \tilde{T}$ and carry the importance weight

$$W(x, x') \overset{\text{def}}{=} \frac{T(x' \mid x, \pi)\, \mathcal{Y}(y \mid x, x')}{\tilde{T}(x' \mid x, y)}.$$

The kernel must be positive wherever $T(\cdot \mid x, \pi)\, \mathcal{Y}(y \mid x, \cdot)$ is, so that the weighted pairs represent the Bayes update of Eq. 1. The bootstrap kernel $\tilde{T}(\cdot \mid x, y) = T(\cdot \mid x, \pi)$ gives $W = \mathcal{Y}$ and only requires sampling from $T(\cdot \mid x, \pi)$; other kernels also require evaluating the transition density.

4.1 Training

GDEs are trained under a plug-in scheme in which each training distribution contributes a set of samples, half used to compute θ and half to score the conditional generator [Fishman et al., 2025].

Algorithm 1: Neural Bayesian Filtering

input : θ (belief embedding), y (observation), $T, \mathcal{Y}, \pi$ (system model), $\tilde{T}$ (proposal kernel), n (number of particles), φ (test function)

output : θ' (updated belief embedding), $\hat{\mu}^{(n)}(\varphi)$ (estimate of the posterior expectation of φ)

$x_{1:n} \overset{\text{iid}}{\sim} \mathcal{G}_\psi(\theta)$ `// generate particles from the conditional generator`

for $i \leftarrow 1 \dots n$ **do** $x'_i \sim \tilde{T}(\cdot \mid x_i, y),\ w_i \leftarrow W(x_i, x'_i)$ `// propagate and weight`

$\hat{\mu}^{(n)}(\varphi) \leftarrow \frac{\sum_{i=1}^n w_i \varphi(x'_i)}{\sum_{i=1}^n w_i}$ `// self-normalized estimate`

return $\mathcal{E}_\phi(x'_{1:n}, w_{1:n}),\ \hat{\mu}^{(n)}(\varphi)$ `// embed the weighted set`

In filtering problems those distributions are beliefs, and sampling a belief is the filtering problem itself. NBF instead trains on the sets the filter itself embeds, which carry whatever error the model has accumulated along the episode, and scores the generator on the single realized state provided by the environment.

Training data is generated by simulating the environment and filtering with NBF under the current GDE parameters. An episode draws an initial state[1] and rolls the environment forward for $t_{\max}$ steps, which yields the states $x^{(0)}, \dots, x^{(t_{\max})}$ and the observations $y^{(1)}, \dots, y^{(t_{\max})}$. The filter is also given the initial belief, the policy, and any parameters of T and $\mathcal{Y}$ that vary between episodes. We collect these into $y^{(0)}$ and treat them as an observation, so b_t in Proposition 4.1 is the posterior of the episode process given $y^{(0:t)}$. NBF is then run along the observations. At each step it generates particles from the previous embedding, propagates them, and weights them by the new observation, and S_t denotes the resulting weighted set that Algorithm 1 embeds at step t, with S_0 drawn from the initial belief. The filter runs on the observations alone, and the realized states are used only as targets. Particle sets for filtering are generated under the current parameters, and the loss is differentiated with the sets treated as constants, so no gradient passes back through the rollout that produced them. Given the generated sets and realized states, the objective is

$$\mathcal{L}(\phi, \psi) \overset{\text{def}}{=} \mathbb{E}\left[\frac{1}{t_{\max}+1} \sum_{t=0}^{t_{\max}} \ell_\psi\big(x^{(t)};\, \mathcal{E}_\phi(S_t)\big)\right], \tag{2}$$

where the expectation is over episodes and over the particle sets NBF generates along them. $\ell_\psi(x; \theta)$ is the generator's loss at the single target x given the embedding θ.

An episode draws $x^{(t)}$ and $y^{(0:t)}$ from the same process that defines b_t ($x^{(t)} \sim b_t$, given the observations) and S_t is a function of those observations and the filter's randomness. When ℓ_ψ is the negative log-density, the objective is the expected cross-entropy of the true posterior under the generator's belief. The **set posterior** $p(\cdot \mid S_t)$ is the conditional distribution of the state $x^{(t)}$ given the set S_t, as b_t is its conditional distribution given the observations. The negative log-density is a strictly proper scoring rule [Gneiting and Raftery, 2007], so the set posterior is the unique minimizer of each term of the objective over conditional densities of the state given the set. Every target is a realized state, so the objective needs no reference posterior and the generator is never scored against its own samples. Proposition 4.1 decomposes the loss into three terms (proof in Appendix A).

Proposition 4.1 (Realized-state objective)**.** *Let S_t be a random set that is a function of $y^{(0:t)}$ and of randomness independent of the episode, and assume that $x^{(t)}$ has finite entropy, both unconditionally and given $y^{(0:t)}$. For any conditional density $\rho(\cdot \mid S)$ over states given a set,*

$$\mathbb{E}\big[-\log \rho(x^{(t)} \mid S_t)\big] = \mathrm{H}\big(x^{(t)} \mid y^{(0:t)}\big) + \mathbb{E}\big[\mathrm{KL}\big(b_t \,\|\, p(\cdot \mid S_t)\big)\big] + \mathbb{E}\big[\mathrm{KL}\big(p(\cdot \mid S_t) \,\|\, \rho(\cdot \mid S_t)\big)\big].$$

The GDE parameters that generate an episode's sets are fixed during the episode, so each set is a function of the episode's observations, and this decomposition applies throughout training. The flow-matching loss that trains the models of Section 5 has the same minimizer, so Proposition 4.1 decomposes the NLL they are evaluated on, but not their training loss. The first term is the entropy of the belief, which no filter can score below. The second is the information about the state that the set has lost relative to the observations; it depends on the sets and not on ρ, and it is zero when the set

[1] In decision processes, a policy must also be sampled from a set of training policies.

determines b_t. The third is the model error, the only term that depends on ρ, so a trained generator's excess loss is its expected KL to the set posterior.

For a generator with a tractable likelihood, $\ell_\psi(x;\theta)$ is the negative log-density of x under $\mathcal{G}_\psi(\theta)$. For flow matching it is $\mathcal{L}_{\text{FM}}$ with x as the only target, which has the same minimizer over conditional distributions [Lipman et al., 2022]. Gradients flow through the embedding of each set and through the generator, and particle sets are never stored across transitions.

4.2 Convergence of NBF

NBF is *consistent* under the following assumptions:

(A1) *Polish state and observation spaces.* X and Y are separable and completely metrizable.

(A2) *GDE regularity.* $\mathcal{E}_\phi$ is distributionally invariant [Fishman et al., 2025] in the weighted sense: it depends on a weighted set only through the set's weighted empirical measure, and it extends to probability measures on X. The induced plug-in map $\Phi(P) \stackrel{\text{def}}{=} \mathcal{G}_\psi(\mathcal{E}_\phi(P))$ is well-specified ($\Phi(b_t) = b_t$) and weakly continuous at every belief state b_t, $t \le t_{\max}$, along the observation sequence.

(A3) *Regular weight and proposal kernel.* At each step $t < t_{\max}$, let q denote the proposal distribution over pairs (x, x') obtained by drawing $x \sim b_t$ and then $x' \sim \tilde{T}(\cdot \mid x, y^{(t+1)})$, and let $W(x, x')$ be the importance weight for the observation $y^{(t+1)}$. We require:

- (i) *Bounded continuous weight:* W is continuous and takes values in $[0, W_{\max}]$ for some $W_{\max} < \infty$.
- (ii) *Non-degenerate likelihood:* $\mathbb{E}_q[W] > 0$.
- (iii) *Feller kernel [Crisan and Doucet, 2002]:* if $x_k \to x$ then $\tilde{T}(\cdot \mid x_k, y^{(t+1)})$ converges weakly to $\tilde{T}(\cdot \mid x, y^{(t+1)})$.

Assumptions (A1) and (A3) adapt the conditions of the particle filter convergence theorem of Crisan and Doucet [2002], a Feller kernel and a bounded continuous likelihood on Euclidean spaces, to Polish spaces and to a weight on pairs of states. Assumption (A2) is an idealization of the trained model. It requires the generator to return every belief the filter reaches from that belief's embedding, and a weighted set whose empirical measure is close to that belief to decode to a distribution close to it, so the error the embedding adds at each step vanishes with the sampling error of the particle set as n grows.

Theorem 4.2 (NBF Consistency)**.** *Fix a finite horizon t_{max} and an observation sequence $y^{(1:t_{max})}$ with belief states b_t, $t \le t_{max}$, for which Assumptions (A1)–(A3) hold, and let $\varphi : X \to \mathbb{R}$ be bounded and continuous. For $t \ge 1$, let $\hat{\mu}_t^{(n)}(\varphi)$ be the self-normalized estimate of $\mathbb{E}_{b_t}[\varphi]$ computed by NBF using Algorithm 1 with n particles per step, and let $\hat{\mu}_0^{(n)}(\varphi)$ be the average of φ over the n particles drawn from b_0. Then $\sup_{0\le t\le t_{max}} |\hat{\mu}_t^{(n)}(\varphi) - \mathbb{E}_{b_t}[\varphi]| \xrightarrow{a.s.} 0$ as $n \to \infty$, and b_{θ_t}, the generator's distribution at the embedding θ_t of step t, converges weakly to b_t almost surely for every $t \le t_{max}$.*

We add one further condition, on the error the embedding adds at each step, to establish the standard Monte-Carlo rate. Further details, including proofs, are included in Appendix A.

Corollary 4.3 (NBF Convergence Rate)**.** *Under the same conditions as Theorem 4.2, suppose additionally that the embedding error is of Monte-Carlo order, $\mathbb{E}_{\mathcal{G}_\psi(\theta_t)}[g] - \hat{\mu}_t^{(n)}(g) = O_p(n^{-1/2})$ for every $t \le t_{max}$ and every bounded continuous g, where θ_t is the embedding at step t and $\hat{\mu}_t^{(n)}$ is the weighted empirical measure of the set it embeds. Then $\sup_{0\le t\le t_{max}} |\hat{\mu}_t^{(n)}(\varphi) - \mathbb{E}_{b_t}[\varphi]| = O_p(n^{-1/2})$.*

5 Experiments

We evaluate NBF on Lorenz-96, a chaotic system in up to 40 dimensions whose posterior is unimodal but non-Gaussian, and on pursuit–evasion, a tracking problem whose posterior is multimodal. The experiments test four properties: representation of multimodal posteriors, graceful scaling with state dimension, native density estimation, and zero-shot generalization to dynamics and likelihoods held out from training. More details about both domains can be found in Appendix B.

Table 1: Lorenz-96 at the held-out $F = 8$. *ms* is wall-clock per filter step on one L40S GPU. SEMs are at most 0.2 nats on NLL and 0.03 on RMSE and CRPS.

	$K = 20$				$K = 40$			
Method	NLL	RMSE	CRPS	ms	NLL	RMSE	CRPS	ms
NBF(1024) (ours)	23.4	0.965	0.524	3.6	53.6	1.060	0.564	4.0
NBF(4096) (ours)	**23.1**	0.943	0.513	11.0	**52.6**	**1.029**	**0.549**	11.8
PF-Reg(131072)	–	**0.877**	**0.489**	9.4	–	4.332	3.286	20.9
EnKF(100)	31.7	1.484	0.770	0.01	63.1	1.506	0.764	0.01

Lorenz-96. The state is a ring of $K \in \{20, 40\}$ coordinates under the chaotic dynamics of Lorenz [1996] with forcing F. Every second site is observed, perturbed with Gaussian noise, once per window of 0.4 time units. This time interval is long enough that the forecast is nonlinear between observations (Appendix B.2). The forcing F is the quantity that varies between episodes (Section 2). Training draws F from $[5, 7] \cup [9, 11]$ and σ_{obs} from $[0.5, 1.1]$. Evaluation runs at $F = 6$ and $F = 10$ inside the training range and at the held-out $F = 8$, at $\sigma_{\text{obs}} = 1$ unless stated. The likelihood factors over the $K/2$ observed sites, and the product of that many factors concentrates the weight of a particle set on a few particles from the first update [Snyder et al., 2008]. On this domain the particle weights provided to NBF's encoder are the vectors of standardized residuals $(y_j - x'_j)/\sigma_{\text{obs}}$ over the observed sites, with zero at unobserved sites.

Pursuit–evasion. The agent tracks k evaders moving through an obstacle-filled arena with a single stationary range sensor. Each evader follows a stochastic heading policy drawn from a training mixture of four, and four further policies are held out for evaluation (Appendix B.1.1). The range noise σ is drawn once per episode from $[0.01, 0.05]$. Both evader movements and sensor range readings are perturbed with Gaussian noise. The k range readings output by the sensor are randomly shuffled before they reach the filter, so the likelihood is a sum over the $k!$ assignments of ranges to evaders. This likelihood does not factor over the joint state in $\mathbb{R}^{2k}$, and induces a multimodal posterior.

Methods. Every filter is given the same dynamics, observation model, and initial belief, except that GMF starts from a Gaussian mixture centred on the arena. **NBF**(n) is our method with n particles in the generate-propagate-embed loop. **PF**(n) is a bootstrap (SIR) particle filter [Gordon et al., 1993] with n particles and systematic resampling, and **PF-Reg**(n) is a regularized particle filter [Musso et al., 2001], which jitters the resampled set with the shrinkage kernel of Liu and West [2001] to counter impoverishment. **EnKF**(m) is a stochastic ensemble Kalman filter with m members [Evensen, 1994, Burgers et al., 1998], and **GMF**(m) is a Gaussian sum filter with m components [Alspach and Sorenson, 1972], run on pursuit–evasion only. Results for resample-move and locally optimal proposal particle filters appear in Appendix E, and the proposals and settings of every baseline are detailed in Appendix D.

Metrics. Every method is scored against the realized state. NLL is the negative log-density of the true state under the filter's density. Particle filters have no native density, so we also measure the continuous ranked probability score (CRPS) [Gneiting et al., 2007] of each marginal against the true coordinate. CRPS is computed from an equal-weight sample for every filter. RMSE of the posterior mean is the standard data-assimilation score and is reported on Lorenz-96. We do not report RMSE on pursuit–evasion, since the mean of a multimodal belief can lie between its modes, where the posterior has little mass. Scores are averaged over the steps of an episode after a five-step spin-up and over 500 episodes, generated from five evaluation seeds shared by every method. Every NBF entry is the mean over three independently trained models with the SEM over those models. For other filters with no training phase, SEM is computed over episodes. The tables in this section report the headline cells, and Appendix E reports every method at every particle budget with additional calibration metrics.

Scaling with state dimension. Without local analyses [Poterjoy, 2016, Schenk et al., 2022], a particle filter's effective sample size collapses as the state dimension grows and the number of independent observations grows with it [Snyder et al., 2008]. NBF regenerates its set from its embedding at every step, so its particle budget does not have to grow as rapidly with state dimension.

Table 2: Pursuit–evasion in distribution, averaged with equal weight over the four policies of the training mixture. *ms* is as in Table 1. SEMs are at most 1.0 nats on NLL for NBF and 2.6 for the EnKF, and 0.03 on CRPS for all methods.

	$k = 1$			$k = 3$			$k = 5$		
Method	NLL	CRPS	ms	NLL	CRPS	ms	NLL	CRPS	ms
NBF(512) (ours)	-0.01	1.285	0.44	11.6	1.087	1.6	22.0	0.937	2.2
NBF(2048) (ours)	**-0.07**	1.280	1.6	**10.5**	1.070	4.5	**20.3**	0.926	5.8
PF(32768)	–	**1.090**	0.071	–	1.698	0.23	–	1.685	0.49
PF-Reg(32768)	–	1.157	0.091	–	1.678	0.28	–	1.698	0.61
EnKF(100)	5.25	1.500	0.008	15.0	**1.025**	0.008	31.1	**0.897**	0.009
GMF(64)	26.8	1.697	0.12	958.6	1.320	0.18	3703.4	1.199	0.30

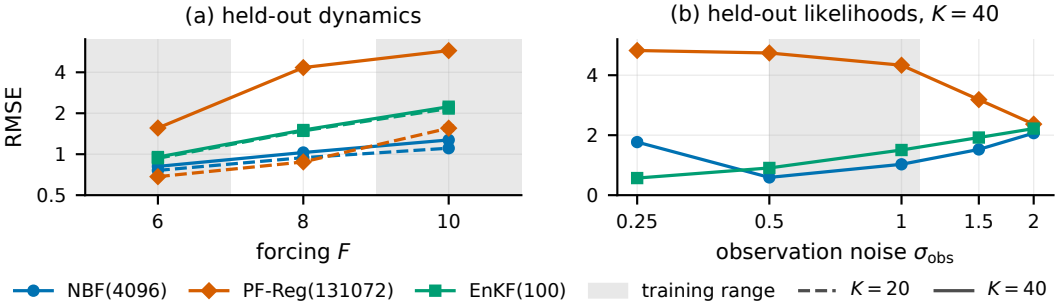

Figure 2: Zero-shot generalization on Lorenz-96. (a) RMSE against the forcing F at $K = 20$ and $K = 40$. (b) RMSE against the observation noise at $K = 40$. Shading marks the training ranges, and SEMs are below 0.1.

NBF's error should therefore move with the EnKF's as the dimension grows, while the particle filters go from the most accurate filter at lower dimensions to collapsed at the highest. Tables 1 and 2 show this on both domains. A particle filter is the most accurate at $K = 20$ and $k = 1$ and the particle filters are the least accurate at $K = 40$ and $k \geq 3$, while NBF's error follows the EnKF's and stays below it on Lorenz-96. Quadrupling NBF's particles barely changes its error, so at these budgets its error is not limited by the particle count.

Multimodal posteriors. In pursuit–evasion, a single range constrains an evader to a ring around the sensor, so the belief is annular at $k = 1$ and multimodal at every k. A single Gaussian centred on the ring places its mass inside and around it, where the true state is not, so its density at the true state should be low however accurate its mean, while NBF's generator can place the mass directly on the ring. In Table 2, NBF has the lowest NLL at every k, and GMF has the highest by a wide margin, which we attribute to its components fitting slivers of the ring. On CRPS the EnKF is worse than NBF at $k = 1$ and comparable at $k \geq 3$, so its mean is a competitive point estimate while its Gaussian misplaces the mass.

Zero-shot generalization. With the bootstrap proposal our experiments use, system dynamics enter NBF only through propagation, so the algorithm should generalize to a held-out forcing or policy unless the beliefs it induces fall outside the family the encoder was trained on. Likewise, the likelihood enters through weighting, and the encoder is given the weights but not the likelihood's parameters, so a held-out likelihood should cost nothing where the weights it produces resemble those seen in training but could fail where they do not. Figure 2 plots RMSE on Lorenz-96 against forcing and observation noise. At both K, NBF's RMSE at held-out $F = 8$ lies on the trend over the two training intervals, and its margin over the EnKF grows with F, which we attribute to the forecast ensemble moving further from Gaussian as the dynamics become more chaotic. At $K = 40$, NBF has the lowest RMSE from the bottom of the training range of σ_{obs} to well above its top, but below it NBF trails the EnKF, which we attribute to weight features there that are unlike any the encoder saw in training.

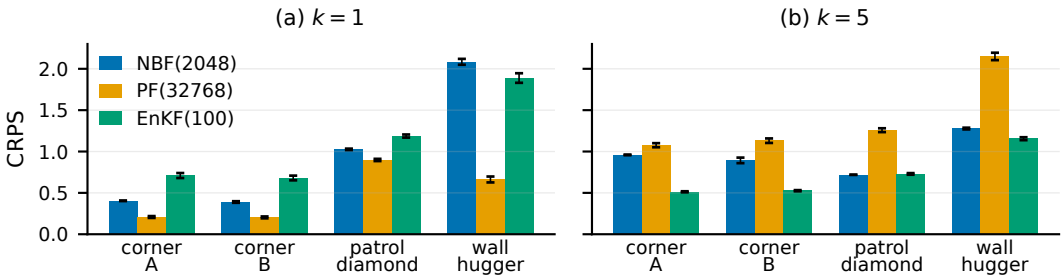

Figure 3: Zero-shot generalization on four held-out policies in pursuit–evasion. Error bars are SEMs over trained models for NBF and over episodes for the other filters.

Figure 3 plots CRPS on the four held-out pursuit–evasion policies. At $k = 1$ the state is two-dimensional and the likelihood is a single factor, so PF is close to the best attainable CRPS and has the lowest CRPS on every held-out policy. NBF trails it on all four, and by the most on *wall hugger*. We attribute the *wall hugger* gap to coverage of the training distribution. The beliefs induced by wall-following concentrate along structures that the training policies rarely visit, so the embedding has not learned to represent them. At $k = 5$, PF trails NBF on every held-out policy, including *wall hugger*, so NBF's advantage over the particle filters at high dimension can hold even on a policy outside the training mixture for which it performs relatively poorly in low dimensions.

6 Limitations

NBF generalizes zero-shot to held-out dynamics and likelihoods, but only to those whose beliefs and particle weights resemble the ones seen in training. In our experiments this limit appears below the training range of σ_{obs} on Lorenz-96 at $K = 40$, which we attribute to weight features unlike any the encoder saw in training, and under the wall-following policy on pursuit–evasion, which we attribute to beliefs that the training policies rarely visit (Figures 2 and 3). Widening the training range or the policy mixture to cover these cases requires no change to the update. NBF also costs two to three orders of magnitude more per step than an ensemble Kalman filter, whose CRPS is comparable to NBF's on pursuit–evasion at $k \geq 3$, so the expense is warranted when the belief is multimodal or the dimension is high enough that particle filters collapse.

Theorem 4.2 assumes in (A2) an embedding that represents every reachable belief exactly, so it establishes the correctness of the update in isolation and bounds none of the representation error of a trained encoder, which is of fixed size and does not necessarily improve with n. Proposition 4.1 identifies the information the set has lost, and Lemma A.1 bounds it when the set determines the current observation, but the bound does not depend on n. The experiments measure only the sum of the proposition's three terms, as the NLL of the realized state. We leave a bound on that term to future work. Both results concern the update and the objective, so a better encoder or generator improves the filter without any change to either.

NBF assumes that T and $\mathcal{Y}$ are given as input to the problem, since every update evaluates them. It therefore cannot be applied as-is to a system whose dynamics or observation model are unknown. Recurrent state-space models [Krishnan et al., 2015, Karl et al., 2016, Chung et al., 2015] and KalmanNet [Revach et al., 2022] learn the update itself, so their dynamics are fixed at training time, and differentiable particle filters [Jonschkowski et al., 2018, Karkus et al., 2018, Chen and Li, 2024] learn the transition and observation models inside a particle update. The same dynamics enter NBF only through propagation and weighting, so those learned models can replace those defined by the system without changing the update.

7 Conclusion

Neural Bayesian Filtering stores a belief as a learned embedding and updates it through a particle set. Its conditional generator samples particles, the given dynamics and observation likelihood propagate and weight them, and its encoder embeds the result. The embedding represents multimodal, high-dimensional beliefs, and regenerating particles at every step mitigates impoverishment. System

dynamics and likelihood enter only through propagation and weighting, so NBF adapts to dynamics unseen at training time as long as the beliefs they induce lie in the family the embedding was trained on. The model is trained on the particle sets the filter itself produces and scored against the realized state, so training needs episodes with recorded single ground-truth states rather than samples from the target beliefs.

NBF represents a belief as a fixed-size, differentiable summary of the posterior, so a planner can condition on its embedding directly. This could mitigate the particle depletion that depth-limited Monte Carlo planners [Silver and Veness, 2010, Šustr et al., 2021] observe in large games. The generator also gives the belief a tractable density, so a planner can reward reductions in the belief's entropy without fitting a kernel density to particles [Fischer and Tas, 2020]. Beyond planning, Lorenz-96 and pursuit–evasion are small instances of data assimilation for chaotic geophysical models [Evensen, 1994, Hunt et al., 2007] and of target tracking [Haug, 2012] with unlabeled measurements. In both, the dynamics can be simulated, which NBF's training requires, and the posterior can be too high-dimensional or multimodal for Gaussian and particle filters.

Societal impact

NBF is a general-purpose filter that we evaluate only on simulated systems, so its societal impact is that of state estimation in general. Like any tracking method, it has broader potential applications in surveillance.

LLM use statement

Large language models assisted the authors with writing, coding, and theory. They were used to help the authors refine writing, help implement and test the experiment pipeline, and refine and validate theoretical results. The authors revised and verified all LLM work carefully.

Acknowledgments

This work was supported in part by the Canada CIFAR AI Chairs program and the Alberta Machine Intelligence Institute (Amii) and the Natural Sciences and Engineering Research Council of Canada (NSERC), and was enabled in part by support provided by Prairies DRI and the Digital Research Alliance of Canada (`https://www.alliancecan.ca`). R. Haluška and M. Schmid were supported by the grant no. 25-18031S of the Czech Science Foundation (GAČR) and the Charles University Grant Agency (GAUK) project no. 326525. Additional computational resources were provided by the e-INFRA CZ project (ID:90254), supported by the Ministry of Education, Youth and Sports of the Czech Republic.

References

Daniel L. Alspach and Harold W. Sorenson. Nonlinear Bayesian estimation using Gaussian sum approximations. *IEEE Transactions on Automatic Control*, 17(4):439–448, 1972.

Ienkaran Arasaratnam and Simon Haykin. Cubature kalman filters. *IEEE Transactions on automatic control*, 54(6):1254–1269, 2009.

Timothy D Barfoot. *State Estimation for Robotics*. Cambridge University Press, 2024.

Gerrit Burgers, Peter Jan van Leeuwen, and Geir Evensen. Analysis scheme in the ensemble Kalman filter. *Monthly Weather Review*, 126(6):1719–1724, 1998.

Xiongjie Chen and Yunpeng Li. Normalizing flow-based differentiable particle filters. *IEEE Transactions on Signal Processing*, 2024.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. *Advances in neural information processing systems*, 28, 2015.

Dan Crisan and Arnaud Doucet. A survey of convergence results on particle filtering methods for practitioners. *IEEE Transactions on Signal Processing*, 50(3):736–746, 2002.

Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In Dan Crisan and Boris Rozovskii, editors, *The Oxford Handbook of Nonlinear Filtering*, pages 656–704. Oxford University Press, 2011.

R. M. Dudley. *Real Analysis and Probability*. Cambridge University Press, 2 edition, 2002.

Geir Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research: Oceans*, 99(C5): 10143–10162, 1994.

Johannes Fischer and Ömer Şahin Tas. Information particle filter tree: An online algorithm for POMDPs with belief-based rewards on continuous domains. In *International Conference on Machine Learning*, 2020.

Nic Fishman, Gokul Gowri, Peng Yin, Jonathan Gootenberg, and Omar Abudayyeh. Generative distribution embeddings. In *Advances in Neural Information Processing Systems*, 2025.

Gregory Gaspari and Stephen E Cohn. Construction of correlation functions in two and three dimensions. *Quarterly Journal of the Royal Meteorological Society*, 125(554):723–757, 1999.

Walter R Gilks and Carlo Berzuini. Following a moving target: Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society: Series B*, 63(1):127–146, 2001.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E. Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society B*, 69(2):243–268, 2007.

Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140(2):107–113, 1993.

Anton J Haug. *Bayesian Estimation and Tracking: A Practical Guide*. John Wiley & Sons, 2012.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

Brian R Hunt, Eric J Kostelich, and Istvan Szunyogh. Efficient data assimilation for spatiotemporal chaos: A local ensemble transform Kalman filter. *Physica D: Nonlinear Phenomena*, 230(1-2): 112–126, 2007.

Rico Jonschkowski, Divyam Rastogi, and Oliver Brock. Differentiable particle filters: End-to-end learning with algorithmic priors. *arXiv preprint arXiv:1805.11122*, 2018.

Simon J Julier and Jeffrey K Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

Peter Karkus, David Hsu, and Wee Sun Lee. Particle filter networks with application to visual localization. In *Conference on robot learning*, pages 169–178. PMLR, 2018.

Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick Van der Smagt. Deep variational Bayes filters: Unsupervised learning of state space models from raw data. *arXiv preprint arXiv:1605.06432*, 2016.

Rahul G Krishnan, Uri Shalit, and David Sontag. Deep Kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, 2019.

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

Jane Liu and Mike West. Combined parameter and state estimation in simulation-based filtering. In Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors, *Sequential Monte Carlo Methods in Practice*, pages 197–223. Springer, 2001.

Edward N Lorenz. Predictability: A problem partly solved. In *Proceedings of the Seminar on Predictability*, volume 1, pages 1–18. ECMWF, 1996.

Christian Musso, Nadia Oudjane, and François Le Gland. Improving regularised particle filters. In Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors, *Sequential Monte Carlo Methods in Practice*, pages 247–271. Springer, 2001.

Art B Owen. Monte Carlo theory, methods and examples, 2013.

Jonathan Poterjoy. A localized particle filter for high-dimensional nonlinear systems. *Monthly Weather Review*, 144(1):59–76, 2016.

Patrick N Raanes, Yumeng Chen, and Colin Grudzien. DAPPER: Data assimilation with Python, a package for experimental research. *Journal of Open Source Software*, 9(94):5150, 2024.

Guy Revach, Nir Shlezinger, Xiaoyong Ni, Adria Lopez Escoriza, Ruud JG Van Sloun, and Yonina C Eldar. Kalmannet: Neural network aided kalman filtering for partially known dynamics. *IEEE Transactions on Signal Processing*, 70:1532–1547, 2022.

Simo Särkkä and Lennart Svensson. *Bayesian filtering and smoothing*, volume 17. Cambridge university press, 2023.

Nora Schenk, Roland Potthast, and Anne Rojahn. On two localized particle filter methods for Lorenz 1963 and 1996 models. *Frontiers in Applied Mathematics and Statistics*, 8:920186, 2022.

David Silver and Joel Veness. Monte-Carlo planning in large POMDPs. *Advances in neural information processing systems*, 23, 2010.

Bernard W Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

Chris Snyder, Thomas Bengtsson, Peter Bickel, and Jeff Anderson. Obstacles to high-dimensional particle filtering. *Monthly Weather Review*, 136(12):4629–4640, 2008.

Samuel Sokota, Hengyuan Hu, David J Wu, J Zico Kolter, Jakob Nicolaus Foerster, and Noam Brown. A fine-tuning approach to belief state modeling. In *International Conference on Learning Representations*, 2022.

H. W. Sorenson. Kalman filtering techniques. In C. T. Leondes, editor, *Advances in Control Systems*, volume 3, chapter 5. Academic Press, New York, 1966.

Michal Šustr, Vojtech Kovarík, and Viliam Lisy. Particle value functions in imperfect information games. In *AAMAS Adaptive and Learning Agents Workshop*, volume 133, page 138, 2021.

A Proofs

Proposition 4.1 (Realized-state objective)**.** *Let S_t be a random set that is a function of $y^{(0:t)}$ and of randomness independent of the episode, and assume that $x^{(t)}$ has finite entropy, both unconditionally and given $y^{(0:t)}$. For any conditional density $\rho(\cdot \mid S)$ over states given a set,*

$$\mathbb{E}\big[-\log \rho(x^{(t)} \mid S_t)\big] = \mathrm{H}\big(x^{(t)} \mid y^{(0:t)}\big) + \mathbb{E}\big[\mathrm{KL}\big(b_t \,\|\, p(\cdot \mid S_t)\big)\big] + \mathbb{E}\big[\mathrm{KL}\big(p(\cdot \mid S_t) \,\|\, \rho(\cdot \mid S_t)\big)\big].$$

Proof. Write $S \stackrel{\text{def}}{=} S_t$, $x \stackrel{\text{def}}{=} x^{(t)}$, and $y \stackrel{\text{def}}{=} y^{(0:t)}$. Adding and subtracting $\log p(x \mid S)$ inside the expectation gives

$$\mathbb{E}\big[-\log \rho(x \mid S)\big] = \mathrm{H}(x \mid S) + \mathbb{E}\big[\mathrm{KL}\big(p(\cdot \mid S) \,\|\, \rho(\cdot \mid S)\big)\big].$$

By hypothesis the set is a function of the observations and of randomness independent of the episode, so the state and the set are conditionally independent given the observations, $p(x \mid y, S) = p(x \mid y) = b_t$, and $\mathrm{H}(x \mid y, S) = \mathrm{H}(x \mid y)$. By the definition of conditional mutual information,

$$\mathrm{H}(x \mid S) = \mathrm{H}(x \mid y, S) + \mathrm{I}(x; y \mid S) = \mathrm{H}(x \mid y) + \mathrm{I}(x; y \mid S),$$

and all three quantities are finite because $\mathrm{H}(x \mid y) \le \mathrm{H}(x \mid S) \le \mathrm{H}(x)$. Finally,

$$\mathrm{I}(x; y \mid S) = \mathbb{E}\big[\mathrm{KL}\big(p(\cdot \mid y, S) \,\|\, p(\cdot \mid S)\big)\big] = \mathbb{E}\big[\mathrm{KL}\big(b_t \,\|\, p(\cdot \mid S)\big)\big].$$

□

The second term $\mathrm{I}(x^{(t)}; y^{(0:t)} \mid S_t)$ is the information about the state that the observations carry beyond the set. When the set determines the current observation, that information is bounded by what the past observations add to the current one, so the set posterior scores at least as well as the posterior given the current observation alone. The decomposition holds at every particle count and for any quality of the sets, and the set posterior is calibrated at every n because it is a conditional distribution of the state given the set. The plug-in generator $\mathcal{G}_\psi(\mathcal{E}_\phi(S_t))$ of the GDE training scheme [Fishman et al., 2025] is itself a conditional density given the set, so its expected loss is at least that of the set posterior.

Lemma A.1 (Set that determines the observation)**.** *Under the hypotheses of Proposition 4.1, if $y^{(t)}$ is a function of S_t, then*

$$\mathrm{I}\big(x^{(t)};\, y^{(0:t)} \mid S_t\big) \le \mathrm{I}\big(x^{(t)};\, y^{(0:t-1)} \mid y^{(t)}\big),$$

so the expected loss of the set posterior is at most $\mathrm{H}(x^{(t)} \mid y^{(t)})$.

Proof. Write S, x, and y as above. Conditional independence gives $\mathrm{I}(x; y, S) = \mathrm{I}(x; y)$, so $\mathrm{I}(x; y \mid S) = \mathrm{I}(x; y) - \mathrm{I}(x; S)$. Expanding $\mathrm{I}(x; S, y^{(t)})$ in both orders gives $\mathrm{I}(x; S) + \mathrm{I}(x; y^{(t)} \mid S) = \mathrm{I}(x; y^{(t)}) + \mathrm{I}(x; S \mid y^{(t)})$. Since $y^{(t)}$ is a function of S, $\mathrm{I}(x; y^{(t)} \mid S) = 0$, so $\mathrm{I}(x; S) \ge \mathrm{I}(x; y^{(t)})$, and by the chain rule $\mathrm{I}(x; y) - \mathrm{I}(x; y^{(t)}) = \mathrm{I}(x; y^{(0:t-1)} \mid y^{(t)})$. Adding $\mathrm{H}(x \mid y)$ to both sides of the bound gives $\mathrm{H}(x \mid y) + \mathrm{I}(x; y \mid S) \le \mathrm{H}(x \mid y^{(t)})$, which by Proposition 4.1 is the expected loss of the set posterior. □

We next state the strong law for self-normalized importance sampling, which we use to handle the propagate-and-reweight step in the inductive argument. In the filter, the particles at step $t+1$ are drawn from the generator conditioned on the embedding of the step-t set, which was itself computed from n random particles. The proposal is therefore a random distribution that differs for every n, and the particles are i.i.d. only conditionally on it. The identities follow Owen [2013].

Lemma A.2 (Strong Law for Self-Normalized Importance Sampling Estimators)**.** *Let $\mathcal{U}$ be a Polish space, let $W : \mathcal{U} \to [0, W_{\max}]$ be a bounded measurable weight function, and let $\varphi : \mathcal{U} \to \mathbb{R}$ be any bounded measurable function. For each n, let q_n be a random proposal distribution on $\mathcal{U}$, draw $u_1, \dots, u_n \overset{iid}{\sim} q_n$ conditionally on q_n, and let $w_i = W(u_i)$.*

(a) $\frac{1}{n}\sum_{i=1}^n w_i \varphi(u_i) - \mathbb{E}_{q_n}[W\varphi] \xrightarrow{a.s.} 0$ *as* $n \to \infty$.

(b) Let $b^{(n)} \stackrel{\text{def}}{=} c_n \cdot W q_n$ *be the target with normalization constant c_n, and let* $\hat{\mu}^{(n)}(\varphi) = \sum_i w_i \varphi(u_i) / \sum_i w_i$. *If* $\liminf_n \mathbb{E}_{q_n}[W] > 0$ *almost surely, then* $\hat{\mu}^{(n)}(\varphi) - \mathbb{E}_{b^{(n)}}[\varphi] \xrightarrow{a.s.} 0$.

In particular, for a fixed proposal q with $\mathbb{E}_q[W] > 0$ and target $b = c \cdot Wq$, $\hat{\mu}^{(n)}(\varphi) \xrightarrow{a.s.} \mathbb{E}_b[\varphi]$.

Proof. Conditionally on q_n, the summands $w_i\varphi(u_i)$ are i.i.d. and take values in an interval of length $2W_{\max}\|\varphi\|_\infty$, so Hoeffding's inequality [Hoeffding, 1963] gives, for any $\epsilon > 0$,

$$\Pr\Big(\Big|\tfrac{1}{n}\textstyle\sum_{i=1}^n w_i\varphi(u_i) - \mathbb{E}_{q_n}[W\varphi]\Big| > \epsilon\Big) \le 2\exp\left(-\frac{n\epsilon^2}{2W_{\max}^2\|\varphi\|_\infty^2}\right).$$

The bound does not depend on q_n, so it holds unconditionally, and it is summable over n. By the Borel–Cantelli lemma, the difference exceeds ϵ for only finitely many n almost surely, and taking $\epsilon = 1/k$ for $k = 1, 2, \dots$ gives (a).

For (b), $b^{(n)}$ integrates to one, so $c_n^{-1} = \mathbb{E}_{q_n}[W]$, and multiplying and dividing by c_n gives $\mathbb{E}_{q_n}[W\varphi] = c_n^{-1}\bar{\varphi}_n$ with $\bar{\varphi}_n \stackrel{\text{def}}{=} \mathbb{E}_{b^{(n)}}[\varphi]$. Let $N_n \stackrel{\text{def}}{=} \sum_{i=1}^n w_i\varphi(u_i)$ and $B_n \stackrel{\text{def}}{=} \sum_{i=1}^n w_i$, so that $\hat{\mu}^{(n)}(\varphi) = N_n/B_n$, and let

$$\alpha_n \stackrel{\text{def}}{=} \frac{N_n}{n} - c_n^{-1}\bar{\varphi}_n, \qquad \beta_n \stackrel{\text{def}}{=} \frac{B_n}{n} - c_n^{-1}$$

be the errors of the numerator and of the denominator. Part (a) applied to φ and to the constant function one gives $\alpha_n \xrightarrow{a.s.} 0$ and $\beta_n \xrightarrow{a.s.} 0$. Substituting $N_n/n = c_n^{-1}\bar{\varphi}_n + \alpha_n$ and $c_n^{-1} = B_n/n - \beta_n$,

$$\hat{\mu}^{(n)}(\varphi) - \bar{\varphi}_n = \frac{c_n^{-1}\bar{\varphi}_n + \alpha_n}{B_n/n} - \bar{\varphi}_n = \frac{\alpha_n - \bar{\varphi}_n\beta_n}{B_n/n}.$$

By hypothesis there is almost surely a $\delta > 0$ with $c_n^{-1} \ge \delta$ for all large n, so $B_n/n = c_n^{-1} + \beta_n \ge \delta/2$ for all large n. Since $|\bar{\varphi}_n| \le \|\varphi\|_\infty$,

$$\big|\hat{\mu}^{(n)}(\varphi) - \bar{\varphi}_n\big| \le \frac{2}{\delta}\Big(|\alpha_n| + \|\varphi\|_\infty\,|\beta_n|\Big) \xrightarrow{a.s.} 0.$$

□

In the filter, $\mathcal{U} = X \times X$, a point is a pair $u = (x, x')$, the weight is $W(x, x')$, and test functions depend on x' only. Applying the lemma to one update of NBF gives the inductive step of the theorem below: if the distribution the generator samples from at step t converges weakly to b_t, then the distribution at step $t + 1$ converges weakly to b_{t+1}.

Theorem 4.2 (NBF Consistency)**.** *Fix a finite horizon t_{max} and an observation sequence $y^{(1:t_{max})}$ with belief states b_t, $t \le t_{max}$, for which Assumptions (A1)–(A3) hold, and let $\varphi : X \to \mathbb{R}$ be bounded and continuous. For $t \ge 1$, let*

$$\hat{\mu}_t^{(n)}(\varphi) = \frac{\sum_{i=1}^n w_i\varphi(x_i')}{\sum_{i=1}^n w_i}$$

be the estimate of $\mathbb{E}_{b_t}[\varphi]$ computed by NBF using Algorithm 1 with n particles per step, and let $\hat{\mu}_0^{(n)}(\varphi)$ be the average of φ over the n particles drawn from b_0. Then,

$$\sup_{0 \le t \le t_{max}} |\hat{\mu}_t^{(n)}(\varphi) - \mathbb{E}_{b_t}[\varphi]| \xrightarrow{a.s.} 0$$

as $n \to \infty$, and b_{θ_t}, the generator's distribution at the embedding θ_t of step t, converges weakly to b_t almost surely for every $t \le t_{max}$.

Proof. According to Algorithm 1, the weight for transitioning from x_i to x_i' is $w_i = W(x_i, x_i')$. Let θ_t denote the embedding at step t and $b_{\theta_t} = \mathcal{G}_\psi(\theta_t)$ the distribution the generator samples from, which is random and depends on n. We prove by induction that b_{θ_t} converges weakly to b_t almost surely for every $t \le t_{\max}$, and convergence of the estimate follows. Here weak convergence refers to distributions over X and almost surely to the filter's draws: with probability one over those draws, the sequence of distributions b_{θ_t} indexed by n converges weakly.

Base case. $t = 0$ Here $b_t = b_0$. By Lemma A.2(a) with unit weights, applied to each function in a countable convergence-determining class [Dudley, 2002, proof of Theorem 11.4.1], the empirical measure $\hat{\mu}_0^{(n)}$ of the n particles drawn from b_0 converges weakly to b_0 almost surely as $n \to \infty$. By (A2), the plug-in map is weakly continuous and well-specified at b_0, so $b_{\theta_0} \stackrel{\text{def}}{=} \mathcal{G}_\psi(\mathcal{E}_\phi(\hat{\mu}_0^{(n)})) \Rightarrow b_0$ almost surely.

Inductive step. Assume that b_{θ_t} converges weakly to b_t almost surely as $n \to \infty$. Throughout the step, $y \stackrel{\text{def}}{=} y^{(t+1)}$ is the new observation.

After the loop in Algorithm 1, the pairs (x_i, x_i') are, conditionally on b_{θ_t}, i.i.d. draws from $\hat{q}(x, x') \stackrel{\text{def}}{=} b_{\theta_t}(x)\,\tilde{T}(x' \mid x, y)$ with weight function $W(x, x')$. The exact posterior at time $t+1$ can be written as

$$b_{t+1}(x') = c \int b_t(x)\,\tilde{T}(x' \mid x, y)\,W(x, x')\,dx \tag{3}$$

for some normalization constant c, which is the Bayes update of Eq. 1 with the kernel K_y multiplied and divided by $\tilde{T}$, since $\tilde{T}\,W = T(\cdot \mid \cdot, \pi)\,\mathcal{Y}$. The integrand is the proposal $q = b_t \cdot \tilde{T}$ over pairs from (A3) times the weight, so b_{t+1} is the x'-marginal of the target $c\,W q$ of Lemma A.2 with $c^{-1} = \mathbb{E}_q[W]$, and $\mathbb{E}_{b_{t+1}}[\varphi] = \mathbb{E}_q[W\varphi]/\mathbb{E}_q[W]$.

The same identity with b_{θ_t} in place of b_t gives the Bayes update $\tilde{b}_{t+1}$ of the generator's belief, with $\mathbb{E}_{\tilde{b}_{t+1}}[\varphi] = \mathbb{E}_{\hat{q}}[W\varphi]/\mathbb{E}_{\hat{q}}[W]$. The error of the estimate splits into a sampling error and a propagated error,

$$\hat{\mu}_{t+1}^{(n)}(\varphi) - \mathbb{E}_{b_{t+1}}[\varphi] = \big(\hat{\mu}_{t+1}^{(n)}(\varphi) - \mathbb{E}_{\tilde{b}_{t+1}}[\varphi]\big) + \big(\mathbb{E}_{\tilde{b}_{t+1}}[\varphi] - \mathbb{E}_{b_{t+1}}[\varphi]\big), \tag{4}$$

where the first term comes from drawing only n particles and the second from drawing them from b_{θ_t} rather than from b_t.

Propagated error. For any bounded continuous h on $X \times X$, let $g(x) \stackrel{\text{def}}{=} \int h(x, x')\,\tilde{T}(x' \mid x, y)\,dx'$ be the average of h over the successor x' of a particle at x, so that $\mathbb{E}_{\hat{q}}[h] = \mathbb{E}_{b_{\theta_t}}[g]$ and $\mathbb{E}_q[h] = \mathbb{E}_{b_t}[g]$. The function g is bounded by $\|h\|_\infty$, and it is continuous. If $x_k \to x$, then the successor distributions $\tilde{T}(\cdot \mid x_k, y)$ converge weakly to $\tilde{T}(\cdot \mid x, y)$ by (A3-iii). The state x_k is deterministic, so the pair (x_k, x') with $x' \sim \tilde{T}(\cdot \mid x_k, y)$ converges in distribution to the pair (x, x') with $x' \sim \tilde{T}(\cdot \mid x, y)$, and $g(x_k)$, the expectation of h over the former, converges to $g(x)$. The inductive hypothesis therefore gives $\mathbb{E}_{\hat{q}}[h] \to \mathbb{E}_q[h]$ almost surely. The functions W and $W\varphi$ are bounded and continuous (A3-i), hence $\mathbb{E}_{\hat{q}}[W] \to \mathbb{E}_q[W] > 0$ (A3-ii) and $\mathbb{E}_{\hat{q}}[W\varphi] \to \mathbb{E}_q[W\varphi]$ almost surely, and taking the ratio gives $\mathbb{E}_{\tilde{b}_{t+1}}[\varphi] \xrightarrow{\text{a.s.}} \mathbb{E}_{b_{t+1}}[\varphi]$.

Sampling error. Lemma A.2(b) applies with $q_n = \hat{q}$, since its condition $\liminf_n \mathbb{E}_{\hat{q}}[W] > 0$ holds by the previous paragraph, and gives $\hat{\mu}_{t+1}^{(n)}(\varphi) - \mathbb{E}_{\tilde{b}_{t+1}}[\varphi] \xrightarrow{\text{a.s.}} 0$. Both terms of Eq. 4 vanish, so $\hat{\mu}_{t+1}^{(n)}(\varphi) \xrightarrow{\text{a.s.}} \mathbb{E}_{b_{t+1}}[\varphi]$, which is the convergence of the estimate at step $t+1$.

Embedding. The estimate $\hat{\mu}_{t+1}^{(n)}(\varphi)$ is the integral of φ against the weighted empirical measure $\hat{\mu}_{t+1}^{(n)}$ of the new set, which places mass $w_i / \sum_j w_j$ on x_i'. Applying the convergence of the estimate to each function in the same class, whose countably many exceptional events have total probability zero, shows that $\hat{\mu}_{t+1}^{(n)}$ converges weakly to b_{t+1} almost surely. By (A2), the next embedding depends on the set only through $\hat{\mu}_{t+1}^{(n)}$, so $b_{\theta_{t+1}} = \Phi(\hat{\mu}_{t+1}^{(n)})$, which converges weakly to $\Phi(b_{t+1}) = b_{t+1}$ by weak continuity and well-specification. This completes the induction.

Almost sure convergence of the sequence. Now that we have shown almost sure convergence for any $t \le t_{\max}$, we can complete the proof of the theorem. For any $\delta > 0$ and $t \le t_{\max}$, there is almost surely a finite random integer n_t such that $|\hat{\mu}_t^{(n')}(\varphi) - \mathbb{E}_{b_t}[\varphi]| \le \delta$ for all $n' \ge n_t$. Let $n^* \stackrel{\text{def}}{=} \max_{0 \le t \le t_{\max}} n_t$; then for all $n \ge n^*$ we have $|\hat{\mu}_t^{(n)}(\varphi) - \mathbb{E}_{b_t}[\varphi]| \le \delta$ for every $t \le t_{\max}$. The induction shows that b_{θ_t} converges weakly to b_t almost surely for every $t \le t_{\max}$, which is the second claim. □

Corollary 4.3 (NBF Convergence Rate)**.** *Under the same conditions as Theorem 4.2, suppose additionally that the embedding error is of Monte-Carlo order,* $\mathbb{E}_{\mathcal{G}_\psi(\theta_t)}[g] - \hat{\mu}_t^{(n)}(g) = O_p(n^{-1/2})$

for every $t \le t_{max}$ and every bounded continuous g, where θ_t is the embedding at step t and $\hat{\mu}_t^{(n)}$ is the weighted empirical measure of the set it embeds. Then

$$\sup_{0 \le t \le t_{max}} |\hat{\mu}_t^{(n)}(\varphi) - \mathbb{E}_{b_t}[\varphi]| = O_p(n^{-1/2})$$

as $n \to \infty$.

Proof. Use the notation of the proof of Theorem 4.2, and let R_t denote the claim that $\mathbb{E}_{b_{\theta_t}}[g] - \mathbb{E}_{b_t}[g] = O_p(n^{-1/2})$ for every bounded continuous g. We prove R_t by induction to obtain the rate.

At $t = 0$, $\hat{\mu}_0^{(n)}(g)$ is an average of n i.i.d. terms bounded by $\|g\|_\infty$, so $\hat{\mu}_0^{(n)}(g) - \mathbb{E}_{b_0}[g] = O_p(n^{-1/2})$ by Chebyshev's inequality, and adding the hypothesis gives R_0. Assume R_t, fix a bounded continuous φ, and bound the two terms of Eq. 4.

Propagated error. Let $g_1(x) \stackrel{\text{def}}{=} \int W(x, x')\, \varphi(x')\, \tilde{T}(x' \mid x, y)\, dx'$ and let g_2 be the same integral with φ replaced by one; both are bounded and continuous by the proof of Theorem 4.2. Then

$$\mathbb{E}_{\tilde{b}_{t+1}}[\varphi] - \mathbb{E}_{b_{t+1}}[\varphi] = \frac{\mathbb{E}_{b_{\theta_t}}[g_1]}{\mathbb{E}_{b_{\theta_t}}[g_2]} - \frac{\mathbb{E}_{b_t}[g_1]}{\mathbb{E}_{b_t}[g_2]}.$$

By R_t the two numerators differ by $O_p(n^{-1/2})$, as do the two denominators, and the denominators converge to $\mu_w \stackrel{\text{def}}{=} \mathbb{E}_q[W] > 0$ (A3-ii), so the difference of ratios is $O_p(n^{-1/2})$.

Sampling error. Let $Z_i \stackrel{\text{def}}{=} w_i\big(\varphi(x_i') - \mathbb{E}_{\tilde{b}_{t+1}}[\varphi]\big)$ and $B_n \stackrel{\text{def}}{=} \sum_{i=1}^n w_i$, so that

$$\hat{\mu}_{t+1}^{(n)}(\varphi) - \mathbb{E}_{\tilde{b}_{t+1}}[\varphi] = \frac{\sum_{i=1}^n Z_i}{B_n}.$$

Conditionally on b_{θ_t}, the Z_i are i.i.d. with mean $\mathbb{E}_{\hat{q}}[W\varphi] - \mathbb{E}_{\hat{q}}[W]\,\mathbb{E}_{\tilde{b}_{t+1}}[\varphi] = 0$ and $|Z_i| \le 2W_{\max}\|\varphi\|_\infty$, hence

$$\mathbb{E}\Big[\big(\textstyle\sum_{i=1}^n Z_i\big)^2 \,\Big|\, b_{\theta_t}\Big] \le 4nW_{\max}^2 \|\varphi\|_\infty^2 .$$

Let $\bar{\mu}_w \stackrel{\text{def}}{=} \mathbb{E}_{\hat{q}}[W] = \mathbb{E}_{b_{\theta_t}}[g_2]$, which converges to μ_w in probability by R_t. The weights lie in $[0, W_{\max}]$, so by Hoeffding's inequality

$$\Pr\Big(\tfrac{B_n}{n} \le \tfrac{\bar{\mu}_w}{2} \,\Big|\, b_{\theta_t}\Big) \le \exp\Big(-\tfrac{n\bar{\mu}_w^2}{2W_{\max}^2}\Big).$$

Let A_n be the event that $B_n/n > \mu_w/4$. Its complement requires either $\bar{\mu}_w \le \mu_w/2$ or $B_n/n \le \bar{\mu}_w/2$ with $\bar{\mu}_w > \mu_w/2$, so $\Pr(A_n^c) \le \Pr(\bar{\mu}_w \le \mu_w/2) + \exp(-n\mu_w^2/8W_{\max}^2) \to 0$. On A_n the sampling error is at most $\frac{4}{\mu_w n}\big|\sum_i Z_i\big|$ in absolute value, so Chebyshev's inequality gives

$$\Pr\Big[\sqrt{n}\,\big|\hat{\mu}_{t+1}^{(n)}(\varphi) - \mathbb{E}_{\tilde{b}_{t+1}}[\varphi]\big| \ge M\Big] \le \Pr(A_n^c) + \frac{64\, W_{\max}^2 \|\varphi\|_\infty^2}{\mu_w^2 M^2},$$

so the sampling error is $O_p(n^{-1/2})$.

Both terms of Eq. 4 are $O_p(n^{-1/2})$, so $\hat{\mu}_{t+1}^{(n)}(\varphi) - \mathbb{E}_{b_{t+1}}[\varphi] = O_p(n^{-1/2})$, which is the rate at step $t+1$. Applying it with g in place of φ and adding the hypothesis at step $t+1$,

$$\mathbb{E}_{b_{\theta_{t+1}}}[g] - \mathbb{E}_{b_{t+1}}[g] = \big(\mathbb{E}_{b_{\theta_{t+1}}}[g] - \hat{\mu}_{t+1}^{(n)}(g)\big) + \big(\hat{\mu}_{t+1}^{(n)}(g) - \mathbb{E}_{b_{t+1}}[g]\big) = O_p(n^{-1/2}) + O_p(n^{-1/2}),$$

which is R_{t+1}. A maximum of finitely many $O_p(n^{-1/2})$ terms is $O_p(n^{-1/2})$, which gives the supremum over $t \le t_{\max}$. □

B Domains

B.1 Pursuit–evasion

The arena is the square $[0, L]^2$ with $L = 10$ and a single stationary range-only sensor at $\mathbf{x}_{\text{sensor}} = (5, 5)$. Three rectangular obstacles create a rooms-and-corridors layout: a left vertical wall $[2, 3] \times$

$[0, 7]$ with a gap above it, a right vertical wall $[7, 8] \times [3, 10]$ with a gap below it, and a central block $[4, 6] \times [4, 6]$.

At each step every evader's policy draws its action, a heading a; the next position is $x' = x + v(\cos a, \sin a) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma_{\text{dyn}}^2 I)$, and the move is integrated through N_{sub} substeps with reflective wall and obstacle clipping. For $k > 1$, evaders are pushed away from each other according to the repulsion strength s_{int} if they are within each other's radius r_{int}.

Each evader produces a per-step range observation $y_j = \|x_j - \mathbf{x}_{\text{sensor}}\| + \mathcal{N}(0, \sigma^2)$ with σ drawn once per episode from $\mathcal{U}[\sigma_{\min}, \sigma_{\max}]$. The per-evader observations are randomly permuted before being passed to the filter, so the index of an observation carries no information about which evader produced it. This forces the filter to maintain a posterior over the joint state rather than k independent marginals. Only the evaders' policies, initial positions and headings, and the observation noise σ vary between episodes. All other quantities, including obstacle positions, are fixed across training and evaluation. The exact settings are summarized in Table 3.

Table 3: Pursuit–evasion environment settings.

Symbol	Value	Description
L	10.0	Grid size.
k	$\{1, 3, 5\}$	Number of evaders.
t_{max}	20	Episode length (steps).
v	0.5	Per-step speed.
σ_{dyn}	0.2	Dynamics noise std (isotropic).
N_{sub}	4	Substeps per move, each clipped against walls and obstacles.
$\mathbf{x}_{\text{sensor}}$	$(5, 5)$	Sensor position.
$\sigma_{\min}, \sigma_{\max}$	0.01, 0.05	Range-noise std bounds.
r_{int}	1.5	Pairwise repulsion radius between evaders.
s_{int}	1.0	Pairwise repulsion strength.

B.1.1 Policies

Every evader is controlled by a stochastic heading policy built from one of five primitives. Table 4 gives the heading rule of each primitive and Table 5 the training mixture and the held-out policies.

Table 4: Policy primitives; the action a is the heading and $\eta \sim \mathcal{N}(0, 1)$ a fresh draw at every step.

Primitive	Heading rule
`goal`$(\mathbf{g}, c)$	$a = \text{atan2}(\mathbf{g} - x) + \eta/c$
`random_walk`	$a \sim \mathcal{U}[0, 2\pi)$
`patrol`$(\{\mathbf{p}_i\}, c)$	$a = \text{atan2}(\mathbf{p} - x) + \eta/c$ toward the nearest waypoint $\mathbf{p}$ farther than 0.5
`evasive`	$a = \text{atan2}(x - \mathbf{x}_{\text{sensor}}) + 0.3\,\eta$
`wall_hugger`	a along the nearest wall $+ 0.2\,\eta$

Table 5: Training mixture and held-out policies.

Policy	Weight	Parameters
Training mixture		
`goal`	40%	$\mathbf{g} \sim \mathcal{U}[0, L]^2$, $c \sim \mathcal{U}(1, 5)$
`random_walk`	20%	—
`patrol`	20%	$n_{\text{wp}} \sim \mathcal{U}\{2, 3, 4\}$, $\mathbf{p}_i \sim \mathcal{U}[0.5, L-0.5]^2$, $c = 3$
`evasive`	20%	—
Held-out policies		
`opposite_corner_A`	—	`goal`$((1, 1), 5)$
`opposite_corner_B`	—	`goal`$((9, 9), 5)$
`patrol_diamond`	—	`patrol`$(\{(2, 5), (5, 8), (8, 5), (5, 2)\}, 3)$
`wall_hugger`	—	—

Training draws each evader's policy independently from the mixture. The in-distribution evaluation runs one training-mixture component per cell, drawn with the cell's seed and shared by the k evaders, so that each policy's columns in Table 14 isolate one belief family. Tables 2, 8, and 13 average the four with equal weight. The held-out policies are evaluated at $k = 1$ and $k = 5$.

B.2 Lorenz-96

The state is $x \in \mathbb{R}^K$ with dynamics $\dot{x}_j = (x_{j+1} - x_{j-2})\, x_{j-1} - x_j + F$, where the index j is taken modulo K so that the coordinates form a ring. Each window integrates the dynamics with N_{sub} RK4 steps of size Δt and adds Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2_{\text{dyn}} I)$, so the transition kernel is stochastic. The odd-indexed sites are observed, $y_j = x_j + \mathcal{N}(0, \sigma^2_{\text{obs}})$ for $j \in \{1, 3, \ldots, K-1\}$. During training σ_{obs} is drawn once per episode from $\mathcal{U}[\sigma_{\min}, \sigma_{\max}]$; at evaluation it is 1.0, and the likelihood sweep of Section 5 sets it to each value in $\{0.25, 0.5, 1.0, 1.5, 2.0\}$. An episode draws F from one of the two training bands (Table 6) and starts from $F + \mathcal{N}(0, 0.25\, I)$. The trajectory is integrated for t_{spin} time units before the first window so that it has settled onto the attractor, the region of the state space that the chaotic dynamics visit in the long run. The initial belief is $\mathcal{N}(x^{(0)} + \delta, \sigma^2_{\text{init}} I)$ with $\delta \sim \mathcal{N}(0, I)$ drawn at the start of the episode. The supplied belief is broader than the distribution of $x^{(0)}$ given its centre, which is the attractor's measure conditioned on a noisy reading and has no closed form. The classical filters keep the supplied belief, while the realized-state objective lets NBF learn that conditional from its training sets. The exact settings are found in Table 6. The window of 0.4 time units is eight times the 0.05 step of the standard benchmark [Lorenz, 1996]. At 0.05 the forecast is close to linear and a localized ensemble Kalman filter is near-optimal, while at 0.4 the chaotic dynamics run long enough between observations that the forecast ensemble is non-Gaussian.

Table 6: Lorenz-96 environment settings.

Symbol	Value	Description
K	$\{20, 40\}$	State dimension.
t_{max}	20	Episode length (windows).
Δt, N_{sub}	0.01, 40	RK4 step and steps per window (window 0.4).
σ_{dyn}	0.25	Process noise std per window.
$\sigma_{\min}, \sigma_{\max}$	0.5, 1.1	Observation-noise std bounds (training).
F (training)	$[5, 7] \cup [9, 11]$	Forcing bands.
F (evaluation)	$\{6, 8, 10\}$	In-range and held-out forcings.
t_{spin}	10	Spin-up (time units).
σ_{init}	2.0	Initial belief std.

The particle weights in this domain are the vectors of standardized residuals of Section 5. The scalar weight of Algorithm 1, for which Theorem 4.2 is stated, is a function of the residual vector up to the clipping of the input features, so by the identity in the proof of Lemma A.1 the set posterior given the particles and their residual vectors loses no more information than the set posterior given the same particles and their importance weights.

The residual is standardized by σ_{obs}, which is not given to the encoder, but the set still determines the observation. Two particles i and k that differ at an observed site j give $\sigma_{\text{obs}} = (x'_{k,j} - x'_{i,j})/(r_{i,j} - r_{k,j})$ from their residuals r, and then $y_j = x'_{i,j} + \sigma_{\text{obs}}\, r_{i,j}$, so Lemma A.1 applies at every noise level. Covariance inflation and localization [Gaspari and Cohn, 1999, Hunt et al., 2007] scale the ensemble Kalman update to this dimension, and local analyses [Poterjoy, 2016, Schenk et al., 2022] scale the particle filter, in both cases by restricting each update to nearby sites. Our implementation of EnKF is localized and inflated in the same way, and NBF's encoder is passed the whole state and residual vectors with no localization imposed.

C Architecture and Training Details

The same model is trained on both domains: a Set Transformer encoder that maps a weighted particle set to θ and a conditional flow-matching generator. One model is trained per state dimension (K on Lorenz-96, k on pursuit–evasion), with the sizes in Table 7. Hyperparameters were not tuned extensively.

Encoder. The encoder input for each particle is its state concatenated with its weight feature: on pursuit–evasion the scalar weight of Algorithm 1 as a max-shifted log-likelihood, floored at -30 nats, and on Lorenz-96 the K-vector of standardized residuals. The weight enters only as this feature, with no resampling or weighted pooling during propagation or embedding. The set is processed by three rounds of an Induced Set Attention Block [Lee et al., 2019], each of which compresses the n particles into M learned inducing points by cross-attention and broadcasts the summary back, at $O(nM)$ cost per block, followed by a pool over the particles whose output is concatenated to every particle's input for the next round. The pool is a mean over the particles on pursuit–evasion and a single-query attention pool [Lee et al., 2019] on Lorenz-96, and the final pool produces $\theta \in \mathbb{R}^{d_{\text{emb}}}$. States are standardized before they enter the encoder and the generator, on Lorenz-96 by the climatological mean and standard deviation of the state estimated from 512 simulated episodes and on pursuit–evasion by the affine map of $[0, L]^2$ onto $[-1, 1]^2$. Every reported NLL is in the unnormalized state units.

Generative model. The generator is a conditional flow-matching model whose velocity network is an MLP on the concatenation of the state, a time embedding, and θ. Sampling integrates the flow from $z \sim \mathcal{N}(0, I)$ with a fixed-step midpoint method, and the density integrates the flow and its exact divergence backward with Euler steps.

Training. Training follows Section 4.1. Every cycle simulates 64 episodes of $t_{\max} = 20$ steps, runs NBF along them with the current parameters and n particles, and takes 40 gradient steps on batches of 256 (set, realized state) pairs drawn from those rollouts. The optimizer is AdamW with a linear warmup of 5,000 steps and cosine decay from 10^{-4} to 10^{-6}, with gradients clipped at norm 1.0; gradients flow through the encoder and the generator and not through the rollout. During rollouts the generator samples with 10 midpoint steps. The same 10-step generator is the one the deployed filter uses at evaluation; 100 Euler steps are used to score NLL. Training runs for 100,000 total steps.

Lorenz-96. On Lorenz-96 a particle's weight is the vector of standardized residuals at the observed sites (Section 5), clipped to $[-20, 20]$ and zero at the unobserved sites. The rest of the model is unchanged. Each training episode draws F from the training bands and σ_{obs} from $[0.5, 1.1]$ (Table 6). Rollouts use $n = 4096$ particles at $K = 40$, with each gradient step accumulated over 8 micro-batches, and $n = 2048$ at $K = 20$, with the velocity network in bfloat16 during rollouts. Three different seeds were used to train models, each on one NVIDIA L40S GPU. One model trains in 16 hours at $K = 20$ and in 36 hours at $K = 40$, the same to within 0.1 hours across the three seeds.

Pursuit–evasion. Rollouts use $n = 2048$ particles at every k with the velocity network in bfloat16, and the deployed filter is the 10-step bfloat16 generator as on Lorenz-96. Each training episode draws k policies from the training mixture (Table 5) and σ from $[0.01, 0.05]$. Three different seeds were used to train models, each on one NVIDIA L40S GPU. One model trains in 7.7 hours at $k = 1$, 14.4 hours at $k = 3$ and 16.4 hours at $k = 5$, the same to within 0.2 hours across the three seeds.

D Baselines

Particle filters. PF is the bootstrap filter [Gordon et al., 1993]: each particle is simulated through the transition, reweighted by the observation likelihood, and the set is resampled systematically whenever the effective sample size $1/\sum_i w_i^2$ of the normalized weights falls below $n/2$. NBF runs with the bootstrap proposal in every experiment; the variants below change the proposal or add a move after resampling, and any of them could be used as a drop-in replacement in Algorithm 1. PF-Reg, PF-RM and PF-OP share the initialization and resampling of PF. PF-Reg jitters the resampled set with the shrinkage kernel of Liu and West [2001]: particles are contracted toward the weighted mean by $\sqrt{1 - h^2}$ and perturbed with covariance h^2 times the pre-resample weighted covariance, so the ensemble covariance is preserved. The bandwidth h is chosen according to Silverman's rule [Silverman, 1986] for the particle count and state dimension. PF-RM, the resample-move filter of Gilks and Berzuini [2001], follows each resampling with an MCMC move that leaves the posterior invariant. On Lorenz-96 the move is an exact draw from $p(x' \mid x, y)$ given the particle's ancestor x, which is Gaussian because the transition adds Gaussian noise to the integrated state and the observation is Gaussian on the observed sites. PF-OP draws every particle from the same conditional and weights it by the marginal likelihood $p(y \mid x)$, the locally optimal proposal of Doucet and Johansen [2011].

Table 7: NBF encoder, generator, and training hyperparameters per model.

	Lorenz-96		Pursuit–evasion		
	$K = 20$	$K = 40$	$k = 1$	$k = 3$	$k = 5$
Encoder (Set Transformer)					
Encoder input width	40	80	3	7	11
Embedding dim d_{emb}	192	256	64	128	128
Hidden size	512	512	256	256	256
Rounds	3				
Heads	8				
Inducing points M	64	64	32	64	64
Pool	attention	attention	mean	mean	mean
Generative model (flow matching)					
MLP hidden size	512	512	256	512	512
MLP layers	8	8	6	6	8
Time-embedding dim	16				
Sampling steps (rollout / deployed)	10 / 10				
Density steps	100				
Training					
Rollout particles n	2048	4096	2048	2048	2048
Episodes per cycle	64				
Gradient steps per cycle	40				
Batch size	256				
Gradient steps	100k				
Peak / final LR	10^{-4} / 10^{-6}, 5000 warmup steps, cosine decay				
Grad-clip norm	1.0				

Ensemble Kalman filters. EnKF is the stochastic ensemble Kalman filter with perturbed observations [Evensen, 1994, Burgers et al., 1998], with m equal-weight members, multiplicative inflation of 1.05 on the member deviations before the analysis, and the Gaspari–Cohn taper [Gaspari and Cohn, 1999] on the sample covariance with a localization radius of 4 sites. The filter was validated against published Lorenz results: in the fully observed $K = 40$, window 0.05 configuration of the DAPPER suite [Raanes et al., 2024] it reaches an analysis RMSE of 0.200 at 40 members against the reported 0.22, and in the half-observed, window 0.3 configuration of Schenk et al. [2022] it reaches 0.840 between their LMCPF at 0.77 and their LETKF at 0.86. Its NLL is the log-density of the true state under the Gaussian with the ensemble mean and the tapered ensemble covariance. On pursuit–evasion the observation is unlabelled, so each member's predicted ranges and the shuffled observation are sorted before the update, and no localization is applied.

Gaussian sum filter. GMF carries m weighted Gaussian components [Alspach and Sorenson, 1972], each updated as an unscented Kalman filter [Julier and Uhlmann, 2004] whose forecast propagates the sigma points with the policy's mean heading and whose analysis matches the sorted observed ranges in rank order to the ranges predicted from the component mean. The mixture weights are multiplied by each component's likelihood of the observation and renormalized. After every update, pairs of components whose symmetric KL divergence is below 0.5 are merged by moment matching and components below a weight of 0.01 are dropped, so the count stays at most m. The components start at the arena centre with jitter of scale $L/10$ and covariance $(L/2)^2 I$, and the NLL is the log-density of the true state under the mixture.

E Additional Experiments

Mechanism. Within one step, NBF's encoder receives the same propagated, weighted particles that a particle filter would resample. Table 8 scores NBF's propagated set at every step in two ways, as a *weighted draw* under its importance weights, which is what a particle filter reports, and after *embed-and-regenerate*, which is what NBF reports. At $k = 1$ the two score alike on CRPS, since the weights still carry the posterior. At $k = 5$ the effective sample size is about two particles, so the weighted

draw is close to a point mass that covers few of the true coordinates, while embed-and-regenerate of the same particles spreads its samples over the belief.

Table 8: Pursuit–evasion, NBF's propagated set with $n = 2048$ and $\sigma = 0.02$, scored as a weighted draw and after embed-and-regenerate. ESS/n is the effective sample size as a fraction of n.

Read of the set	CRPS	cov50	cov90	ESS/n
$k = 1$				
weighted draw	1.266 ± 0.020	0.47	0.87	0.129
embed-and-regenerate	1.284 ± 0.020	0.55	0.93	0.129
$k = 5$				
weighted draw	1.833 ± 0.012	0.02	0.04	0.001
embed-and-regenerate	0.924 ± 0.007	0.60	0.97	0.001

Calibration metrics. CRPS and coverage are computed from 1024 equal-weight samples of every filter at every step, drawn from the generator of the updated embedding for NBF and by weight from the particles or members of the classical filters. Coverage (cov50, cov90) is the fraction of true coordinates inside the central 50% and 90% intervals of the filter's marginals, 0.50 and 0.90 for a calibrated filter and lower for an overconfident one. On pursuit–evasion with $k > 1$, the shuffled observations make the belief symmetric under relabelling of the evaders. Before CRPS and coverage are computed, every method's samples are relabelled by the assignment of evaders to the true ones that minimizes the squared distance. The relabelling depends on the true state, so at $k > 1$ CRPS and coverage are not proper scores.

Per-step error. Figure 4 plots RMSE against the step of the episode at the held-out $F = 8$.

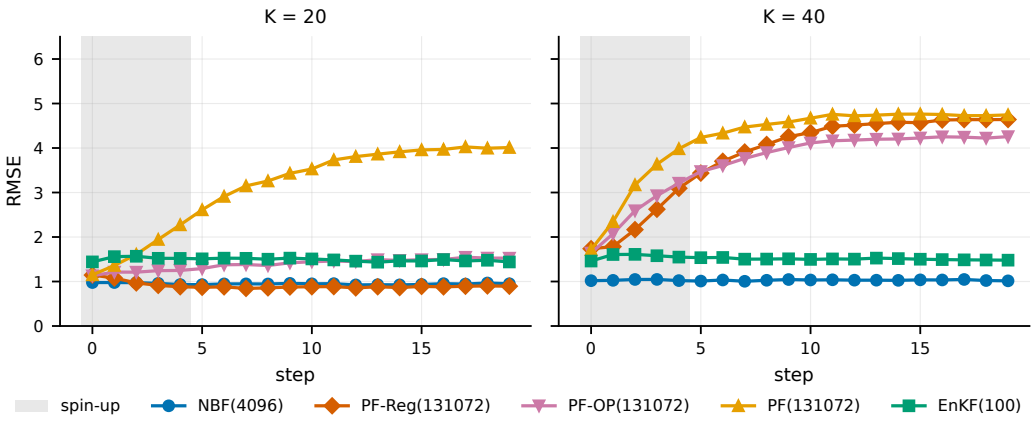

Figure 4: RMSE against the step of the episode at the held-out $F = 8$, averaged over episodes. The shaded area indicates spin-up time.

Compute-matched comparison. Figure 5 plots RMSE against wall-clock per filter step for NBF and every particle-filter family.

Held-out settings. Tables 9 and 10 report every method on Lorenz-96 across the forcing and the observation noise, and Table 11 reports the main methods on the individual held-out pursuit–evasion policies.

Expanded results. Table 12 reports every method and budget on Lorenz-96 at the held-out $F = 8$, including the resample-move and locally optimal proposal filters. Table 13 reports every method on pursuit–evasion, averaged over the four training-mixture policies as in Table 2. Table 14 reports each policy separately.

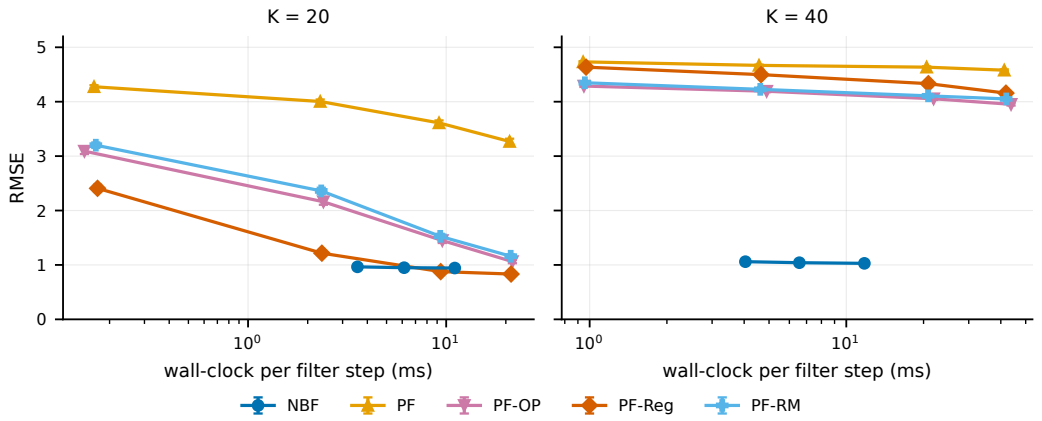

Figure 5: RMSE at the held-out $F = 8$ against wall-clock per filter step; each line joins the budgets of one family, from 1024 to 4096 particles for NBF and from 8192 to 262144 particles for the particle filters.

Table 9: Lorenz-96 RMSE across the forcing F at $\sigma_{\text{obs}} = 1$; training draws F from $[5, 7] \cup [9, 11]$. SEMs are at most 0.09.

	$K = 20$			$K = 40$		
Method	$F = 6$	$F = 8$	$F = 10$	$F = 6$	$F = 8$	$F = 10$
NBF(1024)	0.769	0.965	1.162	0.824	1.060	1.365
NBF(2048)	0.762	0.949	1.125	0.818	1.041	1.311
NBF(4096)	0.760	0.943	1.107	0.814	1.029	1.273
EnKF(50)	0.939	1.503	2.192	0.960	1.524	2.274
EnKF(100)	0.933	1.484	2.166	0.953	1.506	2.234
PF(8192)	1.971	4.275	5.640	3.045	4.732	5.899
PF(32768)	1.476	4.004	5.488	2.905	4.668	5.871
PF(131072)	1.093	3.611	5.324	2.735	4.635	5.839
PF(262144)	0.939	3.268	5.228	2.639	4.579	5.806
PF-OP(8192)	0.965	3.091	4.942	2.347	4.287	5.543
PF-OP(32768)	0.777	2.161	4.467	2.078	4.192	5.464
PF-OP(131072)	0.701	1.449	3.394	1.813	4.054	5.407
PF-OP(262144)	0.694	1.062	2.716	1.597	3.952	5.375
PF-Reg(8192)	0.799	2.408	4.769	2.638	4.635	5.865
PF-Reg(32768)	0.703	1.216	3.139	2.201	4.497	5.817
PF-Reg(131072)	0.684	0.877	1.553	1.558	4.332	5.773
PF-Reg(262144)	0.682	0.832	1.132	1.247	4.158	5.712
PF-RM(8192)	1.063	3.201	5.015	2.465	4.348	5.572
PF-RM(32768)	0.801	2.361	4.551	2.188	4.227	5.516
PF-RM(131072)	0.716	1.526	3.618	1.884	4.108	5.466
PF-RM(262144)	0.702	1.160	2.983	1.765	4.051	5.434

F Code

All source code to reproduce these experiments is publicly available at `https://github.com/polar-rl/nbf`.

Table 10: Lorenz-96 RMSE and NLL at the held-out $F = 8$ across the observation noise σ_{obs}. Training draws σ_{obs} from $[0.5, 1.1]$. SEMs are at most 0.09 on RMSE and 2.1 nats on NLL.

	$K = 20$					$K = 40$				
Method	0.25	0.5	1	1.5	2	0.25	0.5	1	1.5	2
RMSE										
NBF(1024)	0.521	0.561	0.965	1.454	2.031	2.337	0.617	1.060	1.569	2.117
NBF(2048)	0.463	0.551	0.949	1.434	2.008	2.070	0.600	1.041	1.542	2.090
NBF(4096)	0.433	0.547	0.943	1.420	1.997	1.770	0.592	1.029	1.525	2.069
EnKF(50)	0.570	0.903	1.503	1.918	2.214	0.591	0.926	1.524	1.940	2.242
EnKF(100)	0.557	0.886	1.484	1.898	2.197	0.570	0.907	1.506	1.922	2.220
PF(8192)	4.686	4.547	4.275	3.940	3.706	4.855	4.841	4.732	4.595	4.417
PF(32768)	4.634	4.453	3.993	3.583	3.304	4.832	4.797	4.667	4.498	4.320
PF(131072)	4.564	4.294	3.622	3.119	2.856	4.799	4.764	4.635	4.413	4.172
PF(262144)	4.522	4.140	3.274	2.772	2.612	4.805	4.752	4.582	4.331	4.118
PF-OP(8192)	1.495	2.716	3.091	3.086	2.972	2.636	3.776	4.287	4.260	4.143
PF-OP(32768)	1.085	1.814	2.159	2.361	2.391	2.552	3.721	4.190	4.119	3.929
PF-OP(131072)	0.721	1.280	1.456	1.688	1.960	2.409	3.648	4.057	3.930	3.744
PF-OP(262144)	0.604	0.854	1.075	1.443	1.759	2.333	3.578	3.964	3.813	3.642
PF-Reg(8192)	4.683	4.367	2.423	1.663	1.696	4.869	4.819	4.635	4.260	3.704
PF-Reg(32768)	4.597	3.815	1.234	1.239	1.550	4.844	4.761	4.494	3.850	3.095
PF-Reg(131072)	4.509	2.717	0.900	1.154	1.508	4.824	4.741	4.333	3.185	2.369
PF-Reg(262144)	4.390	1.922	0.836	1.136	1.499	4.774	4.714	4.131	2.706	2.090
PF-RM(8192)	1.877	2.957	3.201	3.102	3.016	2.714	3.833	4.348	4.300	4.136
PF-RM(32768)	1.653	2.452	2.351	2.389	2.464	2.666	3.797	4.230	4.156	3.971
PF-RM(131072)	1.298	1.639	1.506	1.729	1.988	2.576	3.731	4.108	3.950	3.753
PF-RM(262144)	1.115	1.216	1.154	1.428	1.758	2.545	3.678	4.051	3.851	3.624
NLL										
NBF(1024)	11.9	13.2	23.4	32.1	41.7	113.6	32.8	53.6	71.3	90.6
NBF(2048)	9.8	12.9	23.2	31.8	41.3	103.0	32.1	52.9	70.5	89.6
NBF(4096)	8.8	12.8	23.1	31.6	41.1	90.4	31.7	52.6	70.0	88.8
EnKF(50)	10.8	20.6	32.1	38.3	41.9	20.8	40.9	64.0	76.3	83.9
EnKF(100)	9.5	19.8	31.7	37.8	41.6	18.1	39.6	63.1	75.5	83.2

Table 11: Pursuit–evasion held-out policies at $k = 1$ and $k = 5$ for the methods of Table 2. SEMs are at most 0.06 on CRPS.

	$k = 1$		$k = 5$	
Method	NLL	CRPS	NLL	CRPS
corner A				
NBF(512)	-0.39	0.424	20.8	0.947
NBF(2048)	-0.45	0.404	19.9	0.959
PF(32768)	–	0.208	–	1.076
PF-Reg(32768)	–	0.257	–	1.133
EnKF(100)	3.15	0.710	21.3	0.513
GMF(64)	10.9	0.818	333.2	0.668
corner B				
NBF(512)	-0.46	0.403	20.3	0.891
NBF(2048)	-0.51	0.390	19.6	0.893
PF(32768)	–	0.204	–	1.131
PF-Reg(32768)	–	0.249	–	1.147
EnKF(100)	3.25	0.680	21.2	0.526
GMF(64)	9.32	0.785	351.3	0.711
patrol diamond				
NBF(512)	0.51	1.022	23.4	0.734
NBF(2048)	0.48	1.026	20.8	0.719
PF(32768)	–	0.898	–	1.258
PF-Reg(32768)	–	0.925	–	1.298
EnKF(100)	4.44	1.187	24.5	0.729
GMF(64)	4.62	1.275	357.0	0.994
wall hugger				
NBF(512)	0.51	2.119	22.1	1.280
NBF(2048)	0.39	2.085	20.9	1.278
PF(32768)	–	0.662	–	2.150
PF-Reg(32768)	–	1.070	–	2.147
EnKF(100)	6.63	1.888	40.1	1.155
GMF(64)	6.12	2.414	2161.7	1.486

Table 12: Lorenz-96 at the held-out $F = 8$, every method and budget; ms is wall-clock per filter step. SEMs are at most 0.3 nats on NLL and 0.07 on RMSE and CRPS.

	$K = 20$						$K = 40$					
Method	NLL	RMSE	CRPS	cov50	cov90	ms	NLL	RMSE	CRPS	cov50	cov90	ms
NBF(1024)	23.4	0.965	0.524	0.51	0.89	3.6	53.6	1.060	0.564	0.52	0.89	4.0
NBF(2048)	23.2	0.949	0.516	0.51	0.90	6.1	52.9	1.041	0.555	0.53	0.89	6.6
NBF(4096)	23.1	0.943	0.513	0.52	0.90	11.0	52.6	1.029	0.549	0.53	0.89	11.8
EnKF(50)	32.1	1.503	0.782	0.50	0.89	0.01	64.0	1.524	0.776	0.50	0.89	0.01
EnKF(100)	31.7	1.484	0.770	0.51	0.91	0.01	63.1	1.506	0.764	0.51	0.91	0.01
PF(8192)	–	4.275	3.285	0.04	0.10	0.17	–	4.732	3.708	0.02	0.05	0.94
PF(32768)	–	4.004	3.030	0.06	0.14	2.3	–	4.668	3.639	0.02	0.05	4.6
PF(131072)	–	3.611	2.685	0.09	0.20	9.2	–	4.635	3.597	0.02	0.06	20.6
PF(262144)	–	3.268	2.391	0.12	0.26	20.9	–	4.579	3.545	0.03	0.06	41.3
PF-OP(8192)	–	3.091	2.191	0.14	0.31	0.15	–	4.287	3.195	0.05	0.12	0.95
PF-OP(32768)	–	2.161	1.443	0.25	0.51	2.4	–	4.192	3.093	0.05	0.13	4.9
PF-OP(131072)	–	1.449	0.902	0.36	0.70	9.6	–	4.054	2.964	0.06	0.15	21.9
PF-OP(262144)	–	1.062	0.617	0.43	0.80	21.6	–	3.952	2.865	0.07	0.16	43.8
PF-Reg(8192)	–	2.408	1.701	0.24	0.47	0.17	–	4.635	3.582	0.03	0.07	0.97
PF-Reg(32768)	–	1.216	0.747	0.41	0.77	2.4	–	4.497	3.450	0.03	0.08	4.7
PF-Reg(131072)	–	0.877	0.489	0.48	0.87	9.4	–	4.332	3.286	0.04	0.10	20.9
PF-Reg(262144)	–	0.832	0.455	0.49	0.89	21.3	–	4.158	3.127	0.05	0.12	41.9
PF-RM(8192)	–	3.201	2.285	0.13	0.29	0.17	–	4.348	3.255	0.05	0.11	0.96
PF-RM(32768)	–	2.361	1.610	0.23	0.47	2.3	–	4.227	3.138	0.05	0.13	4.6
PF-RM(131072)	–	1.526	0.959	0.34	0.67	9.3	–	4.108	3.017	0.06	0.14	20.8
PF-RM(262144)	–	1.160	0.687	0.40	0.77	21.2	–	4.051	2.967	0.06	0.14	42.0

Table 13: Pursuit–evasion in distribution, every method, averaged with equal weight over the four training-mixture policies. SEMs are at most 1.0 nats on NBF's NLL and 0.03 on CRPS.

	$k = 1$				$k = 3$				$k = 5$			
Method	NLL	CRPS	cov50	cov90	NLL	CRPS	cov50	cov90	NLL	CRPS	cov50	cov90
NBF(512)	-0.01	1.285	0.54	0.92	11.6	1.087	0.59	0.97	22.0	0.937	0.62	0.96
NBF(1024)	-0.05	1.284	0.54	0.92	10.7	1.077	0.58	0.97	20.8	0.929	0.62	0.97
NBF(2048)	-0.07	1.280	0.54	0.93	10.5	1.070	0.57	0.97	20.3	0.926	0.61	0.97
PF(8192)	–	1.150	0.49	0.88	–	1.889	0.15	0.31	–	1.754	0.06	0.14
PF(32768)	–	1.090	0.50	0.90	–	1.698	0.23	0.46	–	1.685	0.09	0.20
PF-Reg(8192)	–	1.234	0.41	0.76	–	1.928	0.15	0.32	–	1.774	0.07	0.16
PF-Reg(32768)	–	1.157	0.43	0.77	–	1.678	0.24	0.47	–	1.698	0.10	0.22
EnKF(50)	22.4	1.503	0.47	0.80	91.2	1.114	0.46	0.80	1346.4	1.026	0.41	0.75
EnKF(100)	5.25	1.500	0.50	0.82	15.0	1.025	0.54	0.83	31.1	0.897	0.50	0.81
GMF(16)	35.0	1.692	0.46	0.96	2918.7	1.380	0.39	0.72	8644.9	1.229	0.37	0.70
GMF(64)	26.8	1.697	0.45	0.94	958.6	1.320	0.41	0.75	3703.4	1.199	0.38	0.72

Table 14: Pursuit–evasion in distribution on each training-mixture policy. SEMs on NLL are at most 0.4 nats for NBF and 0.6 for the EnKF.

	goal		random walk		patrol		evasive	
Method	NLL	CRPS	NLL	CRPS	NLL	CRPS	NLL	CRPS
$k = 1$								
NBF(2048)	-0.56	0.489 ± 0.005	1.14	1.585 ± 0.002	0.18	0.781 ± 0.017	-1.04	2.265 ± 0.005
PF(32768)	–	0.305 ± 0.017	–	1.583 ± 0.027	–	0.253 ± 0.018	–	2.220 ± 0.026
EnKF(100)	3.17	0.750 ± 0.029	5.14	1.768 ± 0.036	3.64	1.001 ± 0.025	9.04	2.481 ± 0.038
$k = 3$								
NBF(2048)	10.4	0.852 ± 0.012	12.5	1.128 ± 0.002	11.1	0.898 ± 0.008	7.88	1.403 ± 0.003
PF(32768)	–	0.983 ± 0.037	–	1.945 ± 0.038	–	1.197 ± 0.040	–	2.666 ± 0.061
EnKF(100)	13.5	0.639 ± 0.013	15.7	1.179 ± 0.016	12.9	0.747 ± 0.014	18.2	1.536 ± 0.021
$k = 5$								
NBF(2048)	19.9	0.852 ± 0.007	23.4	0.952 ± 0.001	21.0	0.788 ± 0.003	17.0	1.114 ± 0.002
PF(32768)	–	1.174 ± 0.027	–	1.879 ± 0.027	–	1.291 ± 0.029	–	2.397 ± 0.042
EnKF(100)	24.9	0.567 ± 0.009	29.2	1.024 ± 0.014	23.3	0.648 ± 0.011	47.0	1.347 ± 0.017

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Contributions from Sections 3, 4 through 5 are summarized in the second half of the abstract and the last paragraph of the introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The method's performance and limitations are discussed in Section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Theoretical results are presented with the full set of assumptions in Section 4.2 with formal proofs in Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Domain details with a full set of hyperparameters are provided Appendix B.1, policies and their hyperparameters in Appendix B.1.1, the pseudocode for the algorithm is in Algorithm 1, architecture and training details are in Appendix C, and finally, baselines and metrics are detailed in Appendix D.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The source code for every experiment, with instructions to reproduce the reported results, is released on GitHub as stated in Appendix F.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Architecture, training details and hyperparameters are provided in Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All results report the mean and the standard deviation/standard error of the mean over multiple episodes, policies and random seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Compute resources used by our experiments are detailed in Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The conducted research conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The Societal impact paragraph after the conclusion discusses the impact of the work, which is that of state estimation in general, including its potential application to surveillance.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: No data is being released, and the models used in the paper have little risk for misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [N/A]

    Justification: No existing assets have been used.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The released code is documented with instructions to reproduce the experiments, as stated in Appendix F.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing or research with human subjects have been conducted.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No crowdsourcing or research with human subjects have been conducted.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs assisted with writing and with implementing and testing the experiment pipeline, as stated in the LLM use statement after the conclusion; they are not a component of the method.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.