Neural Bayesian Filtering

Anonymous Author(s)
Affiliation
Address
email

Abstract

We present Neural Bayesian Filtering (NBF), an algorithm for maintaining posteriors, called beliefs, over hidden states in partially observable systems. NBF is trained to find a good latent representation of the beliefs induced by a task. It maps beliefs to fixed-length embedding vectors, which can condition generative models for sampling. During filtering, particle-style updates compute posteriors in this embedding space using incoming observations and environment dynamics. NBF combines the computational efficiency of classical filters with the expressiveness of deep generative models—tracking rapidly shifting, multimodal beliefs while mitigating the risk of *particle impoverishment*. We validate NBF in state estimation tasks in partially observable variants of Gridworld and the card game Goofspiel.

1 Introduction

Belief state modeling, or computing a posterior distribution over hidden states in partially observable systems, has numerous applications in sequential estimation and decision-making problems, including tracking autonomous robots and learning to play card games [Haug, 2012; Sokota *et al.*, 2022; Barfoot, 2024]. As an example, consider the problem of tracking an autonomous robot with an unknown starting position in a $d \times d$ grid (Figure 1). Suppose the agent's policy is known and an observer sees that the agent moved a step without colliding into a wall. This information indicates how the observer should update their *beliefs* about the agent's position. Tracking these belief states can be challenging when they are either continuous or too large to enumerate [Solinas *et al.*, 2023]—even when the agent's policy is known and the dynamics of the environment are easy to simulate accurately.

A common approach frames belief state modeling as a *Bayesian filtering* problem in which a posterior is maintained and updated with each new observation. Classical Bayesian filters, such as the Kalman Filter [Kalman, 1960] and its nonlinear variants (e.g., Extended and Unscented Kalman Filters [Sorenson, 1985; Julier and Uhlmann, 2004]), assume that the underlying distributions are unimodal and approximately Gaussian. While computationally efficient, this limits their applicability in settings that do not satisfy these assumptions. Particle filters alternatively approximate arbitrary target distributions through sets of weighted particles. However, in high-dimensional state spaces, they can require maintaining exponentially large sets of particles or risking *particle impoverishment*—a phenomenon where the set contains very few particles with significant weight [Doucet *et al.*, 2009].

Advances in generative modeling have provided new methods for filtering in problems with complex, multimodal belief states. However, they approximate the full system dynamics (including agent policies) and update an internal representation of the belief state with each observation. Addressing this limitation is crucial for applications where the policy or environment is known but changes, which happens naturally in some learning algorithms [Moravčík *et al.*, 2017; Schmid *et al.*, 2023].

In this paper, we propose **Neural Bayesian Filtering** (NBF), which models complex, multimodal belief states and updates posteriors efficiently for input policies and environments. Central to our approach is the idea that belief states in a given task form a parameterized set. Much like how mean

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

and variance parameterize the family of Gaussians, a learned embedding vector specifies a particular belief state instance. This embedding can be computed exclusively using samples from the target belief state—making it specific to a given policy, environment, and observation sequence. Given a new observation, NBF updates the embedding to approximate the new posterior by generating, simulating, and then re-embedding particles. Effectively combining particle filtering and deep generative modeling, the algorithm maintains expressive approximations of complex, multimodal belief states. We validate NBF empirically in variants of *Gridworld* and the card game *Goofspiel*.

1.1 Main Contributions

Belief State Embeddings We propose learning an embedding network that compresses sample sets from belief states into a set-invariant vector. Conditioning a generative model on this vector allows for efficient sampling and density estimation on a family of complex posterior distributions.

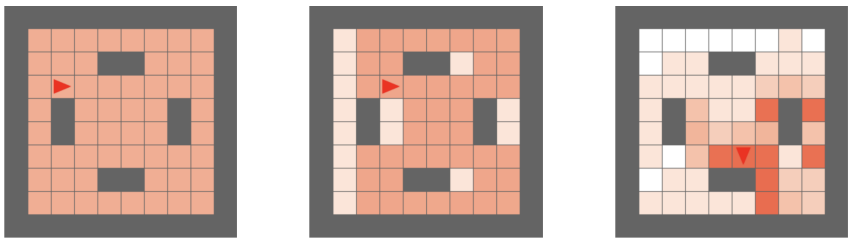
A Flexible Parametric Framework For Filtering We introduce Neural Bayesian Filtering (NBF), a novel parametric filtering framework in the embedding space that combines classical filtering with deep generative modeling. The resulting framework can approximate multimodal, non-Gaussian, and discrete state distributions without prohibitively large particle sets or fixed parametric assumptions.

2 Background

Belief state modeling has been studied in numerous contexts, including Hidden Markov Models (HMMs) [Rabiner, 1989], Partially Observable Markov Decision Processes (POMDPs) [Kaelbling *et al.*, 1998], and Factored Observation Stochastic Games (FOSGs) [Kovařík *et al.*, 2022], and is critical to many decision-time search algorithms. Sokota *et al.* [2022] provide a unified notation for belief state modeling. This work extends their formulation to sets of environments and explicitly models non-stationarity in the environment and control variables. These non-stationarities arise naturally when the agent learns or the environment changes (e.g. different obstacles in the grid in Figure 1).

2.1 Notation

Let $x \in X$ be a **Markov state** and $\pi \in \Pi$ be an external **control** variable (such as a policy in a POMDP or a joint policy in an FOSG). Let $T : X \times \Pi \to \Delta X$ be the **transition function** that determines the underlying dynamics of the process. Emission function $H : X \times X \to \Delta Y$ outputs a probability distribution over the the **observations** (emissions) $y \in Y$ upon transition from x to x'. An **environment** $G \stackrel{\text{def}}{=} (X, Y, T, H, p_0)$ contains state and observation spaces, a transition function, an observation function, and an initial state distribution p_0. In this paper, we extend this formulation to include a set of state spaces $\mathcal{X}$, transition functions $\mathcal{T}$, and observation spaces $\mathcal{Y}$ to model the scenario where an environment instance is known (to the agent or an external observer) but non-stationary. $\mathcal{G} \stackrel{\text{def}}{=} \{(X, Y, T, H, p_0) : X \in \mathcal{X}, Y \in \mathcal{Y}, T \in \mathcal{T}, H \in \mathcal{H}, p_0 \in \Delta X\}$ is the set of environments.



(a) 8x8 grid with initial beliefs.

(b) Beliefs after moving right.

(c) After more steps.

Figure 1: Tracking an agent with an unknown starting position from observations about which direction the agent moved (with some probability of error) and whether or not it hit a wall. Colored cells indicate probabilities of possible agent positions.

Given an instance (G, π), **belief state modeling** is expressed as modeling the distribution over the Markov states at a particular time t, conditional on the control and emission variables: $p(x|\pi, y^{(1)}, \ldots, y^{(t)})$. In discrete Markov systems with small state spaces, belief states can be computed analytically using posterior updates for each observation in the sequence.

In this work, we consider the problem where both G and π are known by the agent or external observer and computationally efficient to evaluate. Together, $\mathcal{G}$ and Π parameterize a set of belief states $\mathcal{P}_G^\Pi \stackrel{\text{def}}{=} \{p(x|\pi, y^{(1)}, \ldots, y^{(t)}) : \pi \in \Pi, G \in \mathcal{G}, y^{(i)} \in Y, t \in \mathbb{N}\}$. For brevity, we will drop the conditional and refer to a member of this set as $p(x)$. Unlike much of the prior work in neural belief state modeling, our approach models this set directly by conditioning on both specific G and π.

2.2 Classical Filtering Algorithms

Bayesian filtering [Särkkä and Svensson, 2023] has been studied extensively as a method for belief state modeling. Environments with linear Gaussian dynamics are the simplest case. In these settings, **Kalman filters** [Kalman, 1960] provide efficient closed-form solutions for the posterior mean and covariance. However, many real-world applications involve nonlinear, non-Gaussian processes. Improvements such as Extended [Sorenson, 1985], Unscented [Julier and Uhlmann, 2004], and Cubature [Arasaratnam and Haykin, 2009] Kalman filters are suitable for non-linear systems but still propagate unimodal beliefs.

Particle filters [Doucet *et al.*, 2009] maintain a representative set of weighted samples for the belief state. This set gets updated according to the environment dynamics upon each new observation. Particle filters can, in principle, approximate a wide range of distributions, but they come with other challenges. Particle impoverishment happens when many particles in the set have little or no weight given the observation sequence and can be catastrophic because replacement particles can only be sampled by duplicating others in the set [Sokota *et al.*, 2022]. Computational efficiency can also become a concern in high-dimensional state spaces because accurate filtering generally requires maintaining an exponential number of particles [Thrun, 2002].

In the next two sections, we describe NBF: a novel approach that models the set of belief states parameterized by Π and $\mathcal{G}$ as a latent space of *belief state embeddings*. These embeddings condition a generative model for sampling and density estimation. Approximate posteriors are updated by sampling particles from the embedding, simulating these particles using the input $G \in \mathcal{G}$ and $\pi \in \Pi$, and then re-computing a weighted embedding using the resulting particles. Learned neural embeddings and fast posterior computation in the embedding space let NBF combine the computational efficiency of parametric approaches like Kalman filters with the flexibility of particle or model-free methods.

3 Embedding Belief States

In this section, we formalize our approach to modeling complex, multimodal belief states using learned neural embeddings. Our goal is to represent and efficiently sample from a given belief state induced by known but potentially changing control variables and environment dynamics.

Given a known control variable $\pi \in \Pi$ and environment $G \in \mathcal{G}$, the induced belief state after t observations is the posterior distribution $p(x)$. One challenge is efficiently modeling the set of these posterior distributions, which may be diverse as π or G vary. We propose embedding these distributions from sets of ground truth samples of example belief states. Our approach aims to construct an embedding vector $\theta \in \mathbb{R}^m$ that uniquely represents the posterior distribution defined by $y^{(1:t)}$, π, and G, and conditions a model for sampling and density estimation.

3.1 Model Definition

We model the target set of belief states using an embedding function and a Normalizing Flow [Papamakarios *et al.*, 2021] conditioned on its output.

Embedding function. Let $x_{1:n} \stackrel{\text{def}}{=} (x_1, x_2, \ldots, x_n)$ denote i.i.d. samples from belief state $p(x)$. We define a permutation and cardinality invariant function $\mathcal{E}_\phi : \mathcal{X}^n \times \mathbb{R}^n \to \mathbb{R}^m$ that maps (weighted) samples $x_{1:n}$ to an m-dimensional embedding vector. A **belief embedding** $\theta \stackrel{\text{def}}{=} \mathcal{E}_\phi(x_{1:n}, w_{1:n})$

approximates the salient features (e.g. shape, location, spread) of a target distribution $p(x)$ as a vector in latent space. Together with the flow described next, it defines the distribution $p_\theta(x)$.

Permutation and cardinality invariance ensures that neither n nor sample order affects θ. In this paper, we take the (weighted) mean over individual sample embeddings. If ϕ is expressive enough, the mean-pooled embedding θ serves as a sufficient moment-based approximation of $p(x)$, but other higher-order architectures such as DeepSets [Zaheer *et al.*, 2017] may also be viable.

Conditional Normalizing Flow. Normalizing flows are a class of generative models that transform a simple base distribution (e.g., Gaussian) into a complex target distribution through a series of invertible and differentiable mappings. They allow for exact likelihood computation via change-of-variables. Flows can also be constructed in continuous time by defining an ordinary differential equation that describes the dynamics of the transformation over time [Lipman *et al.*, 2022].

Given an embedding θ, we can define tractable sampling and density estimation operations on $p_\theta(x)$ by conditioning a normalizing flow on θ. Let $f_\psi(\cdot;\theta) : \mathbb{R}^d \to \mathbb{R}^d$ be an invertible, differentiable transformation conditioned on θ and parameterized by ψ. Given a simple base distribution $p(z)$ (e.g. standard normal), the following two-step sampling procedure:

$$z \sim p(z);\ x = f_\psi(z;\theta),$$

gives the desired density $p_\theta(x)$ (by change-of-variables) [Papamakarios *et al.*, 2021]:

$$p_\theta(x) \stackrel{\text{def}}{=} p\left(f_\psi^{-1}(x;\theta)\right)\left|\det \frac{\partial f_\psi^{-1}(x;\theta)}{\partial x}\right|.$$

If $f_\psi(z;\theta)$ has a tractable inverse, then evaluating this density is also tractable. ψ and ϕ can be optimized jointly by maximizing the log-likelihood over all samples $1,\ldots,N$ and distributions $1,\ldots,K$ in the training set:

$$\mathcal{L}(\phi,\psi) \stackrel{\text{def}}{=} \sum_{k=1}^{K}\sum_{i=1}^{N} \log p_\theta^{(k)}(x_i)$$

Discrete Belief States and Variational Dequantization. Normalizing flows are defined for continuous inputs, but belief states in many relevant domains are discrete. *Variational Dequantization* [Ho *et al.*, 2019] is an approach for applying flows to discrete data. Each discrete sample is perturbed by learned noise, resulting in a continuous space. The noise distribution is trained jointly with the flow by maximizing a variational lower bound on the true discrete log-likelihood—preserving exact likelihood evaluation and stabilizing training for discrete belief states.

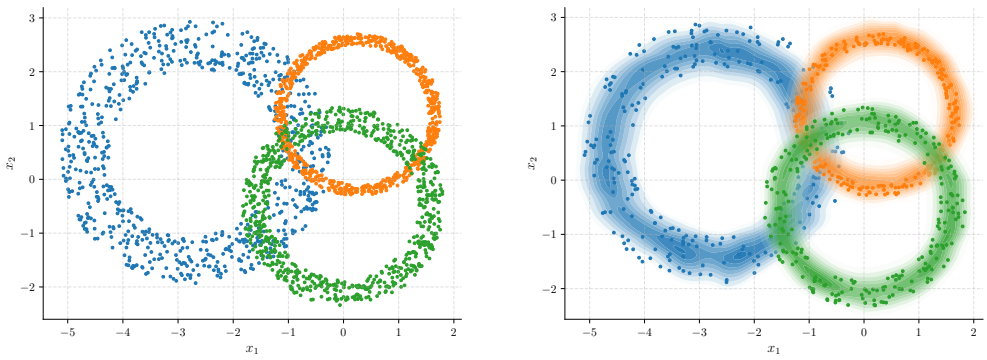
3.2 Illustrative Example: Donuts

Consider a toy domain, called donuts, consisting of a simple set of continuous distributions in $\mathbb{R}^2$. Donuts (Figure 2(a)) are parameterized by a mean, a radius, and a width. Setting these parameters specify a particular donut D, and are sufficient for closed-form sampling from D. Suppose these parameters are not known, and instead $\mathcal{E}_\phi$ receives a set of sample points $x_{1:n} \stackrel{\text{iid}}{\sim} D$ and outputs θ as a parameterization of D. θ conditions the generative model $f_\psi(z;\theta)$, providing an approximation of D and enabling i.i.d sampling from it.

We train our model on randomly generated example donuts using $n = 128$ samples per donut—with $n/2$ used for generating the embedding and the rest used for minimizing the negative log-likelihood objective. At test time, we create embeddings using 64 samples from unseen target donuts. Figure 2(b) shows 3 randomly selected test donuts. Points are samples from the target distribution used to generate θ, while the contours are generated by evaluating the log densities of grid points according to $f_\psi^{-1}(x;\theta)$. Hyperparameters and other training details can be found in the Appendix.

4 Filtering in the Embedding Space

If $p(x)$ is a posterior induced by a sequence of observations, obtaining the samples needed to compute θ may carry significant computational overhead. More general-purpose belief state approximation



(a) Sample donut distributions. Each distribution has three parameters: mean, radius, and width.

(b) Learned densities after conditioning the model on 128 samples from the target distribution.

Figure 2: Embedding the set of donut distributions in $\mathbb{R}^2$

using our model requires tracking the belief state over the observation sequence. In this section, we describe an algorithm that tracks belief states in the embedding space.

Upon receiving an observation, classical parametric methods, such as variants of Kalman filters, compute the posterior in closed form. If θ represents the parameters for a given belief state, such methods define an update function $g : \mathbb{R}^m \times Y \to \mathbb{R}^m$ such that $\theta' = g(\theta, y)$. The modeling assumptions that enable closed-form updates are often violated, which motivates approximate methods. Approximating g for a fixed G and π is a viable choice, but often lacks efficient methods for parameterizing g with G and π in settings where they are variable inputs.

Particle filters are non-parametric: they represent arbitrary target distributions as sets of weighted sampled points called particles. Posterior updates to these empirical distributions are performed by simulating transitions using G and π for each particle and updating weights according to the induced transition probabilities. Below we provide a typical posterior update for a particle filter given observation y, $\pi \in \Pi$, and $G \in \mathcal{G}$:

For each particle x_i and particle weight w_i, $i \in 1, \ldots, n$:

1. Simulate transition $x_i' \sim T_G(x_i, \pi)$
2. Update weight $w_i' \leftarrow w_i \cdot T_G(x_i, \pi)[x_i'] \cdot H_G(x_i, x_i')[y]$

New weights are typically used to *resample* particles by duplicating particles which are more likely to match the updated observation sequence and discarding others. Impoverishment occurs when all particle weights become small and new particles cannot be resampled from outside $x_{1:n}$.

Neural Bayesian Filtering approximates belief states by performing a similar update in the embedding space of a pre-trained belief embedding model. It incorporates $\pi \in \Pi$ and $G \in \mathcal{G}$ into the posterior update like a particle filter, but avoids impoverishment by resampling from the model at every step. Given an embedding, NBF generates particles according to $p_\theta(x)$, simulates them forward while computing their weights exactly like a particle filter, and then computes a new weighted embedding from the result. Figure 3 shows an overview of NBF's posterior update. Full details, including pseudocode, are shown in the Appendix.

4.1 Convergence of NBF with a Perfect Model

NBF is *consistent* and converges at the standard Monte-Carlo rate for finite X and Y under the following assumptions: (i) the embedding model is expressive enough to represent every belief state exactly, and (ii) there exists some global $\epsilon \in (0, 1]$ such that given an observation y, the probability transitioning from any state x to one of its successors x' and observing y is at least ϵ. We call (ii) *ϵ-global observation positivity* of (G, π).

Theorem 4.1 (NBF Consistency)**.** *Assume ϵ-global observation positivity of (G, π) and a finite X and Y. For any finite horizon t_{max}, belief state $p_t(x), t \leq t_{max}$, and any bounded function $\varphi : X \to \mathbb{R}$,*

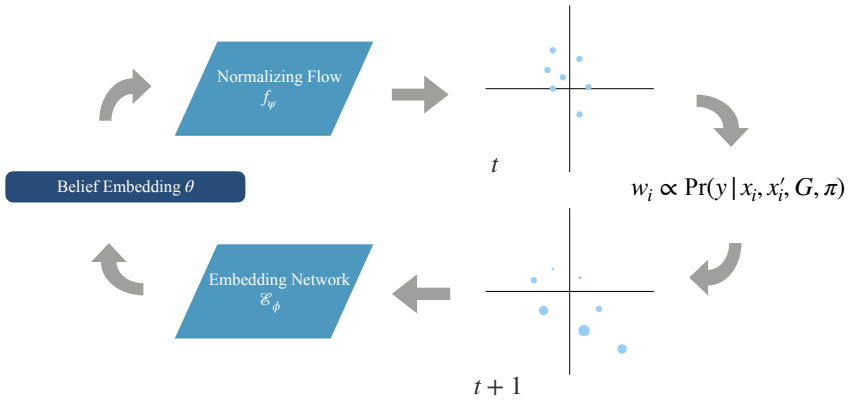


Figure 3: Neural Bayesian Filtering generates particles from a belief embedding, simulates them according to the environment dynamics, and re-embeds them with a likelihood weight proportional to the probability of the input observation y.

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 $\mathbb{E}_{p_t}[\varphi]$ *computed by NBF with a perfect embedding model and* n *particles. Then,*

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

as $n \to \infty$.

Theorem 4.1 states that for any bounded function f on the state space and belief state $p(x)$, NBF's estimate of $\mathbb{E}_p[\varphi]$ almost surely converges to the true value as its number of particles approaches ∞. The following Corollary states its convergence rate under the same conditions.

Corollary 4.2 (NBF Convergence Rate)**.** *Under the same conditions as Theorem 4.1, as* $n \to \infty$,

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

Further details, including proofs, are included in the Appendix.

5 Experiments

We validated Neural Bayesian Filtering in partially observable variants of Gridworld and the card game Goofspiel. Belief states in both domains are discrete, so we used variational dequantization for the belief model described in Section 3. More details, including source code and hyperparameter settings, for all experiments are available in the supplementary material.

We compared four filtering algorithms:

- **Approx Beliefs:** The embedding model described in Section 3 with access to $p(x)$ to generate samples for an embedding size of 32. Each training instance consists of 64 samples from some $p(x) \in \mathcal{P}_G^\Pi$. Model hyperparameters were not tuned extensively.
- **PF** (n)**:** A Sequential Importance Resampling Particle Filter with n weighted particles representing the belief state. An effective sample size less than $n/2$ triggers a systematic resample [Doucet *et al.*, 2009] of the particles.
- **NBF** (n)**:** A Neural Bayesian Filter with the same belief embedding model as "Approx Beliefs" and n particles for posterior computation.
- **Recurrent:** A two-layer LSTM trained to predict $p(x) \in \mathcal{P}_G^\Pi$ from its observation trajectory.

Performance was measured in terms of Jensen-Shannon (JS) divergence between the model's predicted belief state and the ground-truth posterior, with lower values indicating better performance.

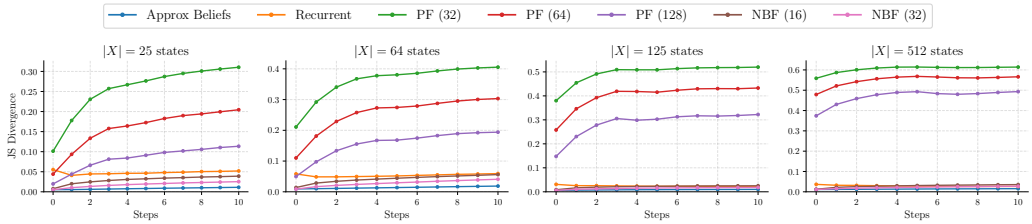


Figure 4: Jensen-Shannon divergence on **fixed** grids and policies (left to right: 5-2D, 8-2D, 5-3D, 8-3D). Training is repeated for 100 random seeds, with each model evaluated over 500 episodes. Shaded areas indicate $\pm$ 1 standard error on the average model performance.

5.1 Partially-Observable Gridworld

We conducted experiments on a partially observable variant of Gridworld with grids of size 5 and 8, and dimensionality 2 and 3. Each grid contains a fixed number of square (or cube) obstacles, and every agent step results in an observation indicating whether the agent hit a wall. In each episode, the observer has access to the agent's policy and a simulator for the grid. Policies are generated by biasing the agent's movement toward a randomly selected goal—softmax temperature controls policy entropy to create noise in the agent's path.

For each dimensionality and size, we evaluated performance in two conditions: a *fixed* grid and policy and a *randomized* grid and policy, yielding eight total experimental configurations (5-2D-fixed, 5-2D-random, 8-2D-fixed, 8-2D-random, 5-3D-fixed, 5-3D-random, 8-3D-fixed, 8-3D-random). In each configuration, the number and size of obstacles are constant, but their location is either fixed or randomized. Each experiment was repeated for 500 episodes to compute a model's average JS divergence at a given step, and model training was repeated for 100 random seeds. Shaded areas in the figures represent one standard error of a model's performance at each step.

Fixed Grids and Policies. Figure 4 summarizes results for fixed grids and policies. The belief model computes its embedding using samples from the target distribution, so it provides an expected performance ceiling for NBF. This shows that the embedding is expressive enough to model the set of belief distributions in this partially observable fixed grid. The recurrent approach is capable of modeling posterior updates on a fixed grid, and further tuning could potentially allow it to perform better than the belief model in the fixed setting. NBF maintains a low JS divergence, comparable to the belief model over many steps, while using a relatively low number of particles. This suggests that NBF's update is effective for approximating the posterior computation in the embedding space. Even with orders of magnitude more particles than NBF, the PFs struggle to achieve comparable performance and demonstrate scalability issues with grid size and dimensionality.

Randomized Grids and Policies. Performance in randomized grids and policies is summarized in Figure 5. Belief embeddings effectively model this much larger set of belief distributions (given that the policy and obstacle placement are now randomized and changing at every episode). With no ability to incorporate policy and grid information into the model, the performance of the Recurrent filter degrades significantly compared to the fixed grid setting. This highlights its limited adaptability to novel or changing environments, regardless of its capacity to express complex belief states in fixed settings. Particle-based methods are more robust to dynamic grids and policies, with NBF performing the best overall despite using relatively few particles. NBF's performance gain over particle filtering likely arises because the belief-embedding model captures relevant information about $\mathcal{P}_G^{\Pi}$.

5.2 Partially-Observable Goofspiel

Our second set of experiments uses a modified version of the card game Goofspiel [Lanctot *et al.*, 2013] with $k \in \{4, 5, 6, 7\}$ cards. In the standard k-card version, both players and the prize deck start with the same set of cards, labeled 0 through $k-1$. A round starts when a *prize card* is revealed, indicating the value of winning the round. Players act by simultaneously *bidding* a card and then observe only the outcome of who played the highest card (win, draw, or loss). In our variant, the card symmetry is broken: each player and the prize deck receives a random subset of size $k-1$, while

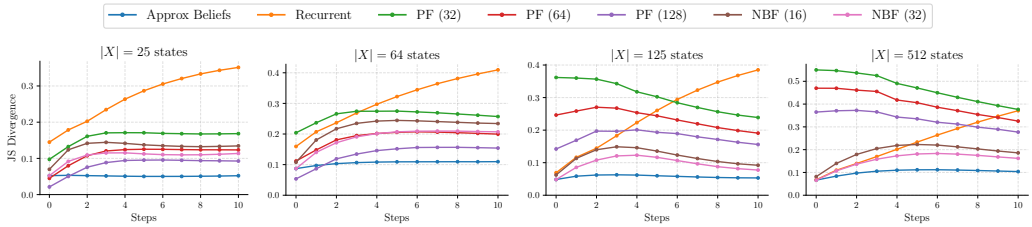


Figure 5: Jensen-Shannon divergence on **randomized** grids and policies (left to right: 5-2D, 8-2D, 5-3D, 8-3D). Training is repeated for 100 random seeds, with each model evaluated over 500 episodes. Shaded areas indicates $\pm$ 1 standard error on the average model performance.

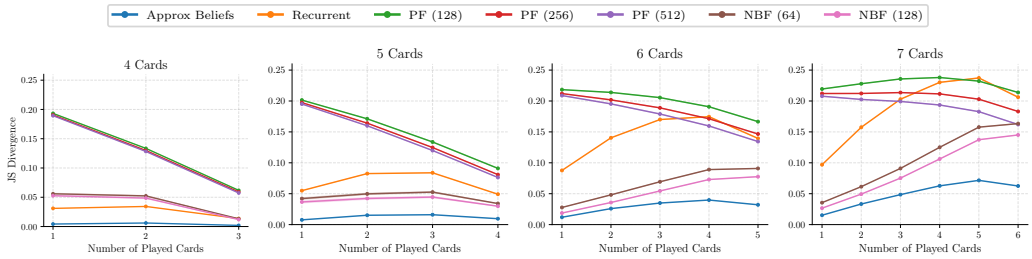


Figure 6: Jensen-Shannon divergence on partially-observable Goofspiel with four, five, six, and seven cards. Training is repeated for 5 random seeds, with each model evaluated over 500 episodes for 10 random seeds. Shaded areas indicate $\pm$ 1 standard error on the average model performance.

all other rules remain unchanged. Small k means exact posterior computation is tractable, enabling efficient training and evaluation of our models and baselines.

During training, samples are obtained by following policies of both players to a randomly selected depth and sampling opponent action histories from the true posterior given the generated observations. The policies are sampled randomly from a pool generated by independent self-play using PPO [Schulman *et al.*, 2017]. These policies were randomly split into a training and test set used only for evaluation. We trained each model on five different random seeds and each filter's reported performance is averaged over 10 different runs, each consisting of 500 episodes.

k-card Goofspiel performance is summarized in Figure 6. Modeling late-game belief states in Goofspiel seems more challenging than in Gridworld. We see this in the growing error of the "Approx Beliefs" filter as the size of the game increases. It is possible that this is due to strong constraints on *legal* states (e.g. hand sizes when t cards have been played). Unsurprisingly, significant inaccuracies in the embedding model appear to have negative downstream effects on NBF's performance. We observe this in the larger variants, where the gap between "Approx Beliefs" and NBF steadily increases. On the other hand, in all four sizes, the particle filter's performance improves at later timesteps as belief state entropy drops. Despite these difficulties, NBF still outperforms the particle filters with an order of magnitude fewer particles (64 vs. 512) in all four sizes.

6 Discussion

Our empirical evaluation demonstrates that NBF remains robust in environments where belief states are small enough to be computed analytically. This is key to evaluating our approach, since metrics like JS divergence require knowledge of the target distribution, but does not demonstrate the scalability of our method compared to the baselines. For deep recurrent approaches to modeling $g(\theta, y)$, scalability and model expressiveness are insignificant when they cannot incorporate critical environment information (G, π). Constraining the particle budget to grow sub-linearly with $|X|$ immediately exposes the scaling pathology of classical particle filters, even in our smallest testbeds. NBF achieves good performance with orders-of-magnitude fewer particles, whereas PFs remain inaccurate despite far larger particle sets. In such cases, the additional cost of embedding and generating a much smaller set of particles with our model is insignificant compared to particle

simulation costs. Confirming this in larger domains for downstream tasks such as learning and sequential decision-making is a promising avenue for future work.

In light of our promising results, NBF has limitations related to its belief embedding model and particle-based updates. For instance, experiments on Goofspiel highlighted the importance of an accurate belief embedding model. In some cases, filtering performance could be highly dependent on choosing appropriate task-specific architectures and training methods.

Training data for the domains tested in this work is both easy to generate and reflective of the set of belief states encountered during filtering. Learning an embedding model from the data encountered while filtering would make NBF applicable to settings where representative training data is difficult or impossible to obtain before filtering. That said, many state-of-the-art online search algorithms [Silver *et al.*, 2017; Moravčík *et al.*, 2017; Schrittwieser *et al.*, 2020; Schmid *et al.*, 2023] require significant computation for offline training but keep online search at decision-time less computationally intensive. These settings suit NBF perfectly and match the experiments conducted in this paper. While NBF's posterior updates reduce the chance of particle impoverishment, they do not eliminate it, especially under extreme conditions where impoverishment can occur in a single step. Such updates can also potentially incur computational overhead during inference compared to pure model-based approaches or the analytical updates of some classical filtering methods. In this sense, NBF trades inference speed for increased representational capacity and adaptability to environmental and control dynamics.

6.1 Related Work

There is a notable connection between variational hidden states in deep recurrent models and belief state modeling [Chung *et al.*, 2015]. Recurrent neural filtering algorithms [Krishnan *et al.*, 2015; Karl *et al.*, 2016; Lim *et al.*, 2020; Revach *et al.*, 2022] can incorporate external observations and learn the overall transition dynamics defined with a fixed π and G. However, this implies that, regardless of their expressivity, these models are fixed at test time and cannot easily be adapted to new transition dynamics.

Alternative methods [Fickinger *et al.*, 2021; Sokota *et al.*, 2022] *fine-tune* a large pre-trained deep generative model via gradient updates at decision time to adapt to changes in the environment dynamics. The model is initially trained on a large sample set aggregated from many belief states and then refined to fit a test-time belief state. Like NBF, such methods can resample from the full support of the distribution, which mitigates impoverishment risks. However, this comes at a cost as fine-tuning may require many costly gradient updates for each target belief state. This makes it less suitable as a component of fast online search algorithms.

Belief state modeling has often been implicitly studied in downstream tasks such as search and learning in partially observable environments. The aforementioned fine-tuning approaches [Fickinger *et al.*, 2021; Sokota *et al.*, 2022] have been applied to search and learning in Hanabi. POMCP [Silver and Veness, 2010] performs Monte Carlo Tree Search from particle-based approximations of belief states. Neural Filtering and Belief Embedding can potentially act as a drop-in replacement for particle filtering and offer richer belief state approximations for search. Likewise, Šustr *et al.* [2021] uses particle-based approximations of value functions for depth-limited search. Approximating value functions in the embedding space is also a promising avenue for future work.

7 Conclusion

We introduced Neural Bayesian Filtering, a method for modeling belief states in partially observable Markov systems. It models the set of distributions induced by a Markov system as a latent space and performs particle-based posterior updates in this latent space upon new observations. Its underlying models for embedding beliefs are trained strictly from sample sets of example belief states, and its posterior update directly integrates non-stationary dynamics and control variables. We show empirically in two partially observable domains that it retains the robustness of traditional particle filtering while approximating rich, multimodal belief states with far fewer particles. Neural Bayesian Filtering has potential applicability well beyond the tasks demonstrated in this paper, extending naturally to various domains involving sequential decision-making, planning, and estimation under uncertainty.

References

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.

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.

Arnaud Doucet, Adam M Johansen, et al. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009.

Arnaud Fickinger, Hengyuan Hu, Brandon Amos, Stuart Russell, and Noam Brown. Scalable online planning via reinforcement learning fine-tuning. *Advances in Neural Information Processing Systems*, 34:16951–16963, 2021.

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

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International conference on machine learning*, pages 2722–2730. PMLR, 2019.

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.

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.

Vojtěch Kovařík, Martin Schmid, Neil Burch, Michael Bowling, and Viliam Lisý. Rethinking formal models of partially observable multiagent decision making. *Artificial Intelligence*, 303:103645, 2022.

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

Marc Lanctot, Viliam Lisỳ, and Mark HM Winands. Monte carlo tree search in simultaneous move games with applications to goofspiel. In *Workshop on Computer Games*, pages 28–43. Springer, 2013.

Bryan Lim, Stefan Zohren, and Stephen Roberts. Recurrent neural filters: Learning independent Bayesian filtering steps for time series prediction. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

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

Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

Lawrence R Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

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.

Martin Schmid, Matej Moravčík, Neil Burch, Rudolf Kadlec, Josh Davidson, Kevin Waugh, Nolan Bard, Finbarr Timbers, Marc Lanctot, G Zacharias Holland, et al. Student of games: A unified learning algorithm for both perfect and imperfect information games. *Science Advances*, 9(46):eadg3256, 2023.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

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.

Christopher Solinas, Doug Rebstock, Nathan Sturtevant, and Michael Buro. History filtering in imperfect information games: algorithms and complexity. *Advances in Neural Information Processing Systems*, 36:43634–43645, 2023.

Harold Wayne Sorenson. Kalman filtering: theory and application. *(No Title)*, 1985.

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.

Sebastian Thrun. Particle filters in robotics. In *UAI*, volume 2, pages 511–518. Citeseer, 2002.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

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: We claim that we can approximately represent belief distributions as vectors in a latent space and filter in the latent space. We demonstrate this empirically in two domains.

   Guidelines:

   - The answer NA 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 NA 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: We discuss potential limitations of Neural Bayesian Filtering in Section 6.

   Guidelines:

   - The answer NA 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: The main body of the paper states assumptions in plain language, but the supplementary material contains formal assumptions and full proofs for the theoretical results presented.

Guidelines:

- The answer NA 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: The main paper summarizes empirical results and provides crucial details about our experimental process, while the supplementary material contains further details and includes all experiment source code.

Guidelines:

- The answer NA 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:

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.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://nips.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, etc.) necessary to understand the results?

   Answer: [Yes] .

   Justification: Training details, hyperparameters, etc. are found in the supplementary material.

   Guidelines:

   - The answer NA 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:

   Guidelines:

   - The answer NA 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: The compute resources used are described in the supplementary material.

   Guidelines:

   - The answer NA 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:

   Guidelines:

   - The answer NA 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: [No]

    Justification: As this work falls under the category of foundational research, we feel both the potential direct positive and negative societal impacts of this are limited.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

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

    Guidelines:

    - The answer NA 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: [NA]

    Justification:

    Guidelines:

    - The answer NA 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:

    Guidelines:

    - The answer NA 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: [NA]

    Justification:

    Guidelines:

    - The answer NA 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: [NA]

    Justification:

    Guidelines:

    - The answer NA 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 rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA 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 (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.