Likelihood-Free Inference in State-Space Models with Unknown Dynamics

Alexander Aushev
Helsinki Institute for Information Technology
Department of Computer Science
Aalto University, Finland
alexander.aushev@aalto.fi

Thong Tran
Helsinki Institute for Information Technology
Department of Computer Science
Aalto University, Finland
thong.tran@aalto.fi

Henri Pesonen
Department of Biostatistics
University of Oslo, Norway
henri.pesonen@medisin.uio.no

Andrew Howes
School of Computer Science
University of Birmingham, UK
andrew.howes@aalto.fi

Samuel Kaski
Helsinki Institute for Information Technology
Department of Computer Science
Aalto University, Finland
Department of Computer Science
University of Manchester, UK
samuel.kaski@aalto.fi

Abstract

We introduce a method for inferring and predicting latent states in the important and difficult case of state-space models (SSM) where observations can only be simulated, and transition dynamics are unknown. In this setting, the likelihood of observations is not available and only synthetic observations can be generated from a black-box simulator. We propose a way of doing likelihood-free inference (LFI) of states and state prediction with a limited number of simulations. Our approach uses a multi-output Gaussian process for state inference, and a Bayesian Neural Network as a model of the transition dynamics for state prediction. We improve upon existing LFI methods for the inference task, while also accurately learning transition dynamics. The proposed method is necessary for modelling inverse problems in dynamical systems with computationally expensive simulations, as demonstrated in experiments with non-stationary user models.

1 Introduction

Conventional likelihood-free inference (LFI) methods [1, 2] estimate parameters $\boldsymbol{\theta}$ of a static statistical model, given the dataset $\mathbf{x}^*$ and a black-box simulator $g_{\boldsymbol{\theta}}$. When the likelihood is intractable, these methods use synthetic datasets $\mathbf{x}^{(i)} \sim g_{\boldsymbol{\theta}}(\mathbf{x}|\boldsymbol{\theta})$ produced by the simulator to assist the inference. LFI has been successfully applied to identify parameters of complex real-world systems, such as financial markets, [3, 4, 5], species populations [6, 7, 8] and cosmology models [9, 10, 11]. Many real-world systems, however, are time-dependent, and can be described as a state-space model (SSM) [12, 13], where observed measurements $\mathbf{x}_t \in \mathbb{R}^n$ are emitted given a series of latent variables, the states, $\boldsymbol{\theta}_t \in \mathbb{R}^m$. When working with dynamical systems, one should account not only for an observation model, a simulator $g_{\boldsymbol{\theta}}$, but for Markovian state transition dynamics $h_{\boldsymbol{\theta}}$, as well, defined as

NeurIPS 2021 Workshop on Deep Learning and Inverse Problems, virtual.

$\boldsymbol{\theta}_{t+1} \sim h_{\boldsymbol{\theta}}(\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_t)$. Therefore, traditional LFI methods, designed for static models, are ill-suited for dynamical systems, such as system simulation experiments in meteorology [14, 15], simulation-based forecasting in cosmology [16, 17] or adaptive modelling of human behaviour [18, 19].

Current LFI methods for dynamical systems [20, 21] proceed by assuming some simplified form of transition dynamics in an SSM. Unless the SSM is both linear and Gaussian, state values are difficult to estimate. This issue has been addressed in a large and growing set of methods, including extended Kalman filters (EKF) [22, 23, 24], GP-SSMs [25, 26], Bayes filtering (BF) [27, 28] and sequential Monte Carlo (SMC) methods [29, 30, 31]. Unfortunately, when the latent states have the additional requirement of being valid simulator parameters, these methods need a tractable likelihood that is not available when the observation model is black-box. At the same time, LFI has been applied to SSMs, but instead of focusing on the problem we address here, the work has focused on more efficient sampling-based methods [32], generation of a better matching statistics [33], on establishing theoretical convergence guarantees [34, 21, 35]. Despite these advances, inference with current LFI methods in SSMs is limited in cases when h_t is non-linear and cannot be sampled or simplified, and when the simulators are computationally expensive.

In this paper, we introduce a method capable of obtaining likelihood-free state approximations and state predictions in discrete-time SSMs. We assume that the state transition model is unknown, can be non-linear and the number of samples from the observation model is limited, when simulators are computationally expensive. To solve the LFI problem in SSMs, we first need to estimate simulator parameters for observed measurements with as few samples as possible, and then use these estimates to learn the transition dynamics. We propose a method, which uses non-parametric multi-objective models for LFI of states, and non-parametric model of transition dynamics for learning state transition dynamics. The contributions of the paper are:

- Focusing on problems where LFI has to be sample-efficient, we improve upon the current LFI methods for the state inference task, while using samples from the state posteriors to accurately learn the state transition dynamics, as shown by comparisons with state-of-the-art SSMs inference techniques. We demonstrate that the proposed method is especially suitable for applications with non-stationary user models.
- We propose sampling from transition dynamics model as a sample-efficient strategy to determine where to run simulations next. We show empirical comparisons with traditional techniques in settings where simulations are computationally expensive.

2 Likelihood-Free State Inference

For LFI, we follow a Bayesian optimization for LFI (BOLFI) approach [36], in which a Gaussian process (GP) surrogate is used for a discrepancy measure δ (e.g. Euclidean distance) between observed $\mathbf{x}^*$ and synthetic datasets $\mathbf{x}^{(i)}$. This approach assumes that synthetic observations with the small discrepancy (smaller than a user-defined threshold ϵ) were produced by the simulator parameters that could plausibly replicate the observed datasets. In this way, the likelihood can be approximated as $p(\mathbf{x}^*|\boldsymbol{\theta}) \propto F\left(\frac{\epsilon-\mu(\boldsymbol{\theta})}{\sqrt{\nu(\boldsymbol{\theta})+\sigma^2}}\right)$, where $F(\cdot)$ is a Gaussian CDF with the mean 0 and the variance 1, $\mu(\boldsymbol{\theta})$ and $\nu(\boldsymbol{\theta})$ are the mean and the standard deviation of a GP, and σ is a likelihood noise of the GP. Because BOLFI actively chooses where to run simulations, its posterior approximation requires much fewer synthetic observations than other LFI methods. However, BOLFI was not designed for SSMs, and hence, does not make use of any temporal information that is typical for SSMs to improve quality of inference.

As an extension to BOLFI, we employ a multi-objective surrogate model for discrepancies $\boldsymbol{\delta}_t$, thus considering multiple discrepancy objectives at the same time and leveraging information between consecutive states. This approach allows using a discrepancy model of the previous state to infer the current state value, instead of simply discarding it. There is an additional advantage of having a multi-objective surrogate in an SSM setting, namely we can have a much more flexible surrogate for LFI than a traditional GP. This does not need any additional data to fit the surrogates, because all synthetic observations for discrepancy objectives can be shared across all state values. When we have a new observation, we simply need to recalculate the discrepancy values for all synthetic datasets. Once we have a trained surrogate for discrepancy objectives, we infer state posteriors $p(\boldsymbol{\theta}_t|\mathbf{x}_t)$, similarly as in BOLFI.

There is an additional challenge for adapting multi-objective surrogates in SSMs, namely a high computational cost associated with considering too many objectives. SSMs can potentially have hundreds of observations, and expanding the number of considered objectives may be detrimental for the performance of the surrogate. We avoid this problem by limiting the number of objectives the surrogate can have. Instead of considering all available time-steps as objectives, we propose to consider only L recent objectives by gradually including new objectives and discarding old ones that have little impact on current states. The size of this moving window depends on how rapidly the transition dynamics change. As the size L of the window grows, the model becomes less sensitive to the noise from the dynamics, at the cost of increased computations and decreased adaptability to the most recent state transitions. The moving window reduces the number of objectives L considered at a time, making multi-objective modelling for SSMs feasible. In the experiments, we used a linear model of coregionalization (LMC) [37] as a surrogate, with a moving window size of 2.

3 Learning State Transition Dynamics

Once we have approximated the posteriors $p(\boldsymbol{\theta}_t|\mathbf{x}_t^*)$ with LFI, we can use their empirical samples to learn a state transition model. This model should be able to train from noisy samples of LFI posterior approximations, and be flexible enough to fit arbitrary function families the dynamics may follow. Thus, the transition model should be Bayesian and non-parametric. Such a model would take the uncertainty associated with posterior approximations into account and be flexible enough to follow possibly non-linear transition dynamics.

We propose to train this model in an autoregressive fashion by forming a training set from paired posterior samples. For each SSM time interval, we group consecutive state posterior samples in a training set, and expand it when new state posteriors become available. This way, the transition model does not need to be retrained when new observations present themselves and can be actively used throughout state inference for determining where to run simulations next. This can be done, by sampling the predictive posterior $p(\boldsymbol{\theta}_{T+1}|\mathbf{x}_T^*)$ from the trained model $\hat{h}_{\boldsymbol{\theta}}$ through $p(\boldsymbol{\theta}_{T+1}|\mathbf{x}_T^*) = \int \hat{h}_{\boldsymbol{\theta}}(\boldsymbol{\theta}_{T+1}|\boldsymbol{\theta}_T)p(\boldsymbol{\theta}_T|\mathbf{x}_T^*)d\boldsymbol{\theta}_T$. Thus, the state transition model $\hat{h}_{\boldsymbol{\theta}}$ does not inform state posteriors directly, but provides simulation candidates for the LFI surrogate. In the experiments, we used a Bayesian Neural Network (BNN) [38, 39] as a transition model for the state dynamics.

4 Experiments

We assess the quality of our method, LMC-BNN, for the inference and prediction tasks in a series of SSM experiments, where a simulator serves as the observation model $g_{\boldsymbol{\theta}}$. For the inference task, we compare the quality of our estimates against other LFI methods: BOLFI [36], SNPE [40], SNLE [41], SNRE [42]. We use a fixed simulation budget for all these methods, with 20 simulations to initialize the models, and then with 2 additional simulations for each new time-step. However, for the neural density estimation (NDE) approaches (SNPE , SNLE, SNRE), we provided all simulations at once, since they do not need additional simulations when observations change. For the prediction task, we sampled trajectories from the transition model and evaluated them against trajectories from the true dynamics. We performed these experiments in SSMs with simulators that have tractable likelihoods, providing them to the state-of-the-art SSM inference methods, GP-SSM [26, 43] and PR-SSM [44], while our method was still doing LFI. For all methods, we provided 50 observations, and then sampled trajectories that had the same length of 50 time-steps.

We also supplied two variations of our method that differ only in the way the next simulations are sampled: LMC-BLR, where samples were taken from a Bayesian Linear Regression (BLR) model that was used as a local linearized model of the transition dynamics; and LMC-qEHVI, where a popular acquisition function for multitask BO, q-Expected Hypervolume Improvement (qEHVI) [45], was used to provide samples.

4.1 The State-Space Models

In this section, we present two case studies with non-stationary user models and three SSMs with tractable likelihoods. In user modelling experiments, we simulated behavioural data from humans that completed two different tasks: uniform manifold approximation and projection (UMAP) [46]

Table 1: Comparison of LFI methods (rows) in different SSMs (columns) for the state inference task. The performance was measured with 95% confidence interval (CI) of the RMSE between estimated vs ground truth state values for 50 time-steps. The best results in each column are highlighted in bold.

Methods	LG	NN	SV	UMAP	Gaze
LMC-BNN	1.7 ± 0.1	6.75 ± 1.15	16.85 ± 2.75	$\mathbf{59.15 \pm 3.65}$	$\mathbf{59.4 \pm 5.9}$
LMC-BLR	$\mathbf{1.3 \pm 0.1}$	$\mathbf{5.55 \pm 0.75}$	$\mathbf{13.15 \pm 3.25}$	61.95 ± 3.05	60.6 ± 5.8
LMC-qEHVI	1.5 ± 0.1	7.1 ± 0.6	24.4 ± 2.5	65.35 ± 3.85	56.9 ± 4.5
BOLFI	1.55 ± 0.15	6.05 ± 0.45	27.35 ± 3.45	67.85 ± 3.35	73.45 ± 5.75
SNPE	7.15 ± 0.65	50.85 ± 1.35	77.4 ± 3.1	71.4 ± 3.5	68.1 ± 7.8
SNLE	6 ± 0.5	24.2 ± 1.6	54.25 ± 2.45	69.95 ± 3.35	77.25 ± 4.05
SNRE	10.4 ± 1.7	57.1 ± 1.7	57.15 ± 2.35	73.65 ± 1.25	80.75 ± 1.35

parametrization and eye movement control for gaze-based selection. Our goal in these experiments was to infer parameters of user models that generated the behavioural data, with three state dimensions in each user model. In the experimental results, these case studies are referred to as UMAP and Gaze, respectively.

In addition to non-stationary user models, we also experimented with three models with tractable likelihoods common in SSM literature: linear Gaussian (LG), non-linear non-Gaussian (NN) and stochastic volatility (SV) models. In the LG model, the state transition dynamics and observation model are both linear, with high observational noise. The NN model is a modified version (leaving only one unique solution for each observation) of a popular non-linear SSM [47]. Lastly, SV models are widely used for predicting volatility of asset prices [48, 49]. We use the specific model by [50]. The dimensionality of states in these models ranges from one to three, starting from LG and moving to SV. More detailed descriptions of all SSMs can be found in the Appendix.

4.2 Results and Discussion

The results for the inference and prediction tasks are presented in Tables 1 and 2, respectively. These tables report Root Mean Squared Error (RMSE), as a performance measure. In the inference task, LMC-based methods clearly outperformed BOLFI and NDE approaches. This indicates that considering multiple objectives at the same time was beneficial for the state inference, and that the model actually leverages information from consecutive states without hindering the performance. Additionally, it can be seen that LMC-based methods performed differently, which can be only attributed to how the next simulations were chosen, as the surrogate was exactly the same in all three methods. As results show, having BNN as a model for state transition was beneficial for experiments with non-stationary user models, while having BLR was more preferable for the simpler models. This suggests that BLR is expressive enough to replicate simple transitions, but struggles with more complex ones, for which BNN was more suitable.

The comparisons with GP-SSMs and PR-SSMs for learning transition dynamics show that our method learns the dynamics more accurately. The SSMs methods that were supposed to provide ground truth performance showed worse results than BLR and BNN approaches. This can be only explained by the lack of observations for learning state transitions by the SSM methods, which also explains the high variance in the sampled trajectories from these methods. As for comparisons between BLR and BNN, BLR performs better only in LG and SV models, while BNN performs better in more complex case studies. Moreover, it should be noted that trajectory sampling from BLR is possible only by retaining all local linearization, which is a far more limiting approach than having BNN. Therefore, BNN is more preferable as a model for transition dynamics.

In all experiments, we attribute the success of the proposed LMC-BNN method to a more flexible multi-output surrogate, and a more efficient way of choosing simulation candidates. The LMC allows multi-fidelity modelling and leveraging information from multiple consecutive time-steps, unlike standard GPs. At the same time, samples from the transition model provide better candidates for simulations than the alternatives. The flexible surrogate along with adaptive acquisition make our method particularly suitable for online settings, where only a handful of samples are allowed per time-step.

Table 2: Comparison of transition dynamics models (rows) in different SSMs (columns). The performance was measured with 95% confidence interval (CI) of the RMSE between sampled vs ground truth trajectories of length 50. The best results in each column are highlighted in bold.

Methods	LG	NN	SV	UMAP	Gaze
LMC-BNN	205 ± 9	$\mathbf{165 \pm 15}$	135 ± 22	$\mathbf{1383 \pm 30}$	$\mathbf{1353 \pm 7}$
LMC-BLR	$\mathbf{64 \pm 7}$	258 ± 37	$\mathbf{100 \pm 37}$	1409 ± 49	1372 ± 3
GP-SSM	284 ± 71	2204 ± 1111	3206 ± 1175	-	-
PR-SSM	253 ± 68	610 ± 510	1378 ± 740	-	-

Acknowledgments and Disclosure of Funding

This work was supported by the Academy of Finland (Flagship programme: Finnish Center for Artificial Intelligence FCAI; grants 328400, 319264, 292334) and UKRI Turing AI World-Leading Researcher Fellowship EP/W002973/1. HP was also supported by European Research Council grant 742158 (SCARABEE, Scalable inference algorithms for Bayesian evolutionary epidemiology). Computational resources were provided by the Aalto Science-IT Project.

References

[1] Scott A Sisson, Yanan Fan, and Mark Beaumont. *Handbook of approximate Bayesian computation*. CRC Press, 2018.

[2] Mikael Sunnåker, Alberto Giovanni Busetto, Elina Numminen, Jukka Corander, Matthieu Foll, and Christophe Dessimoz. Approximate Bayesian computation. *PLoS Comput Biol*, 9(1):e1002803, 2013.

[3] Gareth W Peters, Scott A Sisson, and Yanan Fan. Likelihood-free Bayesian inference for α-stable models. *Computational Statistics & Data Analysis*, 56(11):3743–3756, 2012.

[4] Simon Barthelmé and Nicolas Chopin. Expectation propagation for likelihood-free inference. *Journal of the American Statistical Association*, 109(505):315–333, 2014.

[5] Victor M-H Ong, David J Nott, Minh-Ngoc Tran, Scott A Sisson, and Christopher C Drovandi. Likelihood-free inference in high dimensions with synthetic likelihood. *Computational Statistics & Data Analysis*, 128:271–291, 2018.

[6] Mark A Beaumont. Approximate Bayesian computation in evolution and ecology. *Annual review of ecology, evolution, and systematics*, 41:379–406, 2010.

[7] Giorgio Bertorelle, Andrea Benazzo, and S Mona. ABC as a flexible framework to estimate demography over space and time: some cons, many pros. *Molecular ecology*, 19(13):2609–2625, 2010.

[8] Mark A Beaumont, Wenyang Zhang, and David J Balding. Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.

[9] Chad M Schafer and Peter E Freeman. Likelihood-free inference in cosmology: Potential for the estimation of luminosity functions. In *Statistical Challenges in Modern Astronomy V*, pages 3–19. Springer, 2012.

[10] Justin Alsing, Benjamin Wandelt, and Stephen Feeney. Massive optimal data compression and density estimation for scalable, likelihood-free inference in cosmology. *Monthly Notices of the Royal Astronomical Society*, 477(3):2874–2885, 2018.

[11] Niall Jeffrey, Justin Alsing, and François Lanusse. Likelihood-free inference with neural compression of DES SV weak lensing map statistics. *Monthly Notices of the Royal Astronomical Society*, 501(1):954–969, 2021.

[12] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[13] Rudolf Emil Kalman et al. Contributions to the theory of optimal control. *Boletín de la Sociedad Matemática*, 5(2):102–119, 1960.

[14] Xubin Zeng, Robert Atlas, Ronald J Birk, Frederick H Carr, Matthew J Carrier, Lidia Cucurull, William H Hooke, Eugenia Kalnay, Raghu Murtugudde, Derek J Posselt, et al. Use of observing system simulation experiments in the United States. *Bulletin of the American Meteorological Society*, 101(8):E1427–E1438, 2020.

[15] Ronald M Errico, Runhua Yang, Nikki C Privé, King-Sheng Tai, Ricardo Todling, Meta E Sienkiewicz, and Jing Guo. Development and validation of observing-system simulation experiments at NASA's global modeling and assimilation office. *Quarterly Journal of the Royal Meteorological Society*, 139(674):1162–1178, 2013.

[16] Johannes U Lange, Frank C van den Bosch, Andrew R Zentner, Kuan Wang, Andrew P Hearin, and Hong Guo. Cosmological evidence modelling: a new simulation-based approach to constrain cosmology on non-linear scales. *Monthly Notices of the Royal Astronomical Society*, 490(2):1870–1878, 2019.

[17] Siyu He, Yin Li, Yu Feng, Shirley Ho, Siamak Ravanbakhsh, Wei Chen, and Barnabás Póczos. Learning to predict the cosmological structure formation. *Proceedings of the National Academy of Sciences*, 116(28):13825–13832, 2019.

[18] Theodosis Georgiou and Yiannis Demiris. Adaptive user modelling in car racing games using behavioural and physiological data. *User Modeling and User-Adapted Interaction*, 27(2):267–311, 2017.

[19] Olivier Gimenez, Vivien Rossi, Rémi Choquet, Camille Dehais, Blaise Doris, Hubert Varella, Jean-Pierre Vila, and Roger Pradel. State-space modelling of data on marked individuals. *Ecological modelling*, 206(3-4):431–438, 2007.

[20] Tina Toni, David Welch, Natalja Strelkowa, Andreas Ipsen, and Michael PH Stumpf. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of the Royal Society Interface*, 6(31):187–202, 2009.

[21] Thomas A Dean, Sumeetpal S Singh, Ajay Jasra, and Gareth W Peters. Parameter estimation for hidden Markov models with intractable likelihoods. *Scandinavian Journal of Statistics*, 41(4):970–987, 2014.

[22] Emrah Zerdali and Murat Barut. The comparisons of optimized extended Kalman filters for speed-sensorless control of induction motors. *IEEE Transactions on industrial electronics*, 64(6):4340–4351, 2017.

[23] Brian DO Anderson and John B Moore. *Optimal filtering*. Courier Corporation, 2012.

[24] Masaru Hoshiya and Etsuro Saito. Structural identification by extended Kalman filter. *Journal of engineering mechanics*, 110(12):1757–1770, 1984.

[25] Silvan Melchior, Sebastian Curi, Felix Berkenkamp, and Andreas Krause. Structured variational inference in unstable Gaussian process state space models. *arXiv preprint arXiv:1907.07035*, 2019.

[26] Roger Frigola, Yutian Chen, and Carl Edward Rasmussen. Variational Gaussian process state-space models. *Advances in neural information processing systems*, 27:3680–3688, 2014.

[27] 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.

[28] Vaclav Smidl and Anthony Quinn. Variational Bayesian filtering. *IEEE Transactions on Signal Processing*, 56(10):5020–5030, 2008.

[29] Sinan Yıldırım, Sumeetpal S Singh, Thomas Dean, and Ajay Jasra. Parameter estimation in hidden Markov models with intractable likelihoods using sequential Monte Carlo. *Journal of Computational and Graphical Statistics*, 24(3):846–865, 2015.

[30] Adrian Smith. *Sequential Monte Carlo methods in practice*. Springer Science & Business Media, 2013.

[31] François Septier, Gareth W Peters, and Ido Nevat. Bayesian filtering with intractable likelihood using sequential MCMC. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6313–6317. IEEE, 2013.

[32] Ajay Jasra, Sumeetpal S Singh, James S Martin, and Emma McCoy. Filtering via approximate Bayesian computation. *Statistics and Computing*, 22(6):1223–1237, 2012.

[33] Gael M Martin, Brendan PM McCabe, David T Frazier, Worapree Maneesoonthorn, and Christian P Robert. Auxiliary likelihood-based approximate Bayesian computation in state space models. *Journal of Computational and Graphical Statistics*, 28(3):508–522, 2019.

[34] Laurent E Calvet and Veronika Czellar. Accurate methods for approximate Bayesian computation filtering. *Journal of Financial Econometrics*, 13(4):798–838, 2015.

[35] James S Martin, Ajay Jasra, Sumeetpal S Singh, Nick Whiteley, Pierre Del Moral, and Emma McCoy. Approximate Bayesian computation for smoothing. *Stochastic Analysis and Applications*, 32(3):397–420, 2014.

[36] Michael U Gutmann and Jukka Corander. Bayesian optimization for likelihood-free inference of simulator-based statistical models. *The Journal of Machine Learning Research*, 17(1):4256–4302, 2016.

[37] Thomas R Fanshawe and Peter J Diggle. Bivariate geostatistical modelling: a review and an application to spatial variation in radon concentrations. *Environmental and ecological statistics*, 19(2):139–160, 2012.

[38] Igor Kononenko. Bayesian neural networks. *Biological Cybernetics*, 61(5):361–370, 1989.

[39] Piero Esposito. BLiTZ - Bayesian layers in Torch zoo (a Bayesian deep learning library for torch). `https://github.com/piEsposito/blitz-bayesian-deep-learning/`, 2020.

[40] George Papamakarios and Iain Murray. Fast ε-free inference of simulation models with Bayesian conditional density estimation. In *Advances in neural information processing systems*, pages 1028–1036, 2016.

[41] George Papamakarios, David Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 837–848. PMLR, 2019.

[42] Conor Durkan, Iain Murray, and George Papamakarios. On contrastive learning for likelihood-free inference. In *International Conference on Machine Learning*, pages 2771–2781. PMLR, 2020.

[43] Alessandro Davide Ialongo, Mark Van Der Wilk, James Hensman, and Carl Edward Rasmussen. Overcoming mean-field approximations in recurrent Gaussian process models. *arXiv preprint arXiv:1906.05828*, 2019.

[44] Andreas Doerr, Christian Daniel, Martin Schiegg, Duy Nguyen-Tuong, Stefan Schaal, Marc Toussaint, and Sebastian Trimpe. Probabilistic recurrent state-space models. *arXiv preprint arXiv:1801.10395*, 2018.

[45] Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Differentiable expected hypervolume improvement for parallel multi-objective Bayesian optimization. *arXiv preprint arXiv:2006.05078*, 2020.

[46] Leland McInnes, John Healy, and James Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[47] Genshiro Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of computational and graphical statistics*, 5(1):1–25, 1996.

[48] Stephen J Taylor. Modeling stochastic volatility: A review and comparative study. *Mathematical finance*, 4(2):183–204, 1994.

[49] Neil Shephard. Statistical aspects of ARCH and stochastic volatility. *Monographs on Statistics and Applied Probability*, 65:1–68, 1996.

[50] Ole E Barndorff-Nielsen and Neil Shephard. Econometric analysis of realized volatility and its use in estimating stochastic volatility models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(2):253–280, 2002.

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

[52] Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3. `https://github.com/DLR-RM/stable-baselines3`, 2019.

[53] Xiuli Chen, Aditya Acharya, and Antti Oulasvirta. An adaptive model of gaze-based selection. In *CHI Conference on Human Factors in Computing Systems (CHI'21)*. Association for Computing Machinery, 2021.

[54] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI gym, 2016.

[55] David W Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[56] Davoud Moulavi, Pablo A Jaskowiak, Ricardo JGB Campello, Arthur Zimek, and Jörg Sander. Density-based clustering validation. In *Proceedings of the 2014 SIAM international conference on data mining*, pages 839–847. SIAM, 2014.

[57] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11), mar 2017.

[58] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

[59] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[60] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[61] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[62] Ethem Alpaydin and Cenk Kaynak. Cascading classifiers. *Kybernetika*, 34(4):369–374, 1998.

[63] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. UMAP: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.

[64] Jarno Lintusaari, Henri Vuollekoski, Antti Kangasrääsiö, Kusti Skytén, Marko Järvenpää, Pekka Marttinen, Michael U. Gutmann, Aki Vehtari, Jukka Corander, and Samuel Kaski. ELFI: Engine for likelihood-free inference. *Journal of Machine Learning Research*, 19(16):1–7, 2018.

[65] GPy. GPy: A gaussian process framework in python. `http://github.com/SheffieldML/GPy`, since 2012.

[66] Neculai Andrei. Scaled conjugate gradient algorithms for unconstrained optimization. *Computational Optimization and Applications*, 38(3):401–416, 2007.

[67] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[68] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[69] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5):1190–1208, 1995.

[70] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: a framework for efficient Monte-Carlo Bayesian optimization. In *Advances in Neural Information Processing Systems 33*, 2020.

[71] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: an imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

[72] Russel E Caflisch et al. Monte Carlo and quasi-Monte Carlo methods. *Acta numerica*, 1998:1–49, 1998.

[73] Art B Owen. Scrambling Sobol'and Niederreiter-Xing points. *Journal of complexity*, 14(4):466–489, 1998.

[74] James T Wilson, Frank Hutter, and Marc Peter Deisenroth. Maximizing acquisition functions for Bayesian optimization. *arXiv preprint arXiv:1805.10196*, 2018.

[75] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

[76] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.

[77] In Jae Myung. Tutorial on maximum likelihood estimation. *Journal of mathematical Psychology*, 47(1):90–100, 2003.

[78] John Salvatier, Thomas V Wiecki, and Christopher Fonnesbeck. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, 2:e55, 2016.

[79] Matthew D Hoffman and Andrew Gelman. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

[80] Alvaro Tejero-Cantero, Jan Boelts, Michael Deistler, Jan-Matthis Lueckmann, Conor Durkan, Pedro J. Gonçalves, David S. Greenberg, and Jakob H. Macke. sbi: A toolkit for simulation-based inference. *Journal of Open Source Software*, 5(52):2505, 2020.

[81] David Greenberg, Marcel Nonnenmacher, and Jakob Macke. Automatic posterior transformation for likelihood-free inference. In *International Conference on Machine Learning*, pages 2404–2414. PMLR, 2019.

[82] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

A The State-Space Models with Tractable Likelihoods

In this section, we present details on the three SSMs with tractable likelihoods that were used to assess the quality of state transition models in the experiments. For all three SSMs, we define transition dynamics and an observation model along with priors for LFI. As observations, we used datasets of 10 points, and their mean and standard deviation as summary statistics. The objective in Bayesian Optimization (BO) was the logarithm of Euclidean distance between the observed and simulated summary statistics.

Linear Gaussian (LG). In the LG model, state transition dynamics (Figure 1a) and an observation model are both linear

$$\theta_t = 0.95 \times \theta_{t-1} + v_t, \quad x_t = \theta_t + w_t, \tag{1}$$

with added Gaussian noise $v_t \sim \mathcal{N}(0, 1^2)$ and $w_t \sim \mathcal{N}(0, 10^2)$. The initial state value for the transition dynamics is $\theta_0 = 100$. The prior for the state values is $\theta \sim \text{Unif}(0, 15) \in \mathbb{R}$.

Non-linear non-Gaussian (NN). The NN model is a popular non-linear SSM [47], where the transition dynamics (Figure 1b) and observation model are

$$\theta_{h,t} = \frac{\theta_{h,t-1}}{2} + 25\frac{\theta_{h,t-1}}{\theta_{h,t-1}^2 + 1} + 8\cos(1.2t) + v_t, \quad x_t = \frac{\theta_{h,t}^2}{20} + w_t, \tag{2}$$

with added Gaussian noise $v_t \sim \mathcal{N}(0, 10)$ and $w_t \sim \mathcal{N}(0, 10)$. The initial state value for the noise standard deviation $\theta_{h,0}$ is 0 with the prior $\theta_h \sim \text{Unif}(-30, 30) \in \mathbb{R}$.

Stochastic volatility (SV) models are widely used for predicting volatility of asset prices [48, 49]. Here, we use the model by [50] that specifies transition dynamics (Figure 1c) of volatility θ_v as

$$\theta_{v,t+1} = (z_t - z_{t+1} + \sum_{j=1}^{k} e_j), \quad z_{t+1} = \exp(-\lambda) z_t + \sum_{j=1}^{k} \exp\{-\lambda(t + 1 - c_j)\} e_j, \tag{3}$$

with $c_{1:k} \sim \text{Unif}(t, t+1)$, $e_{1:k} \sim \text{Expon}(0.5/0.25^2)$ and $\lambda = 0.01$. The random increases of volatility are regulated by the Poisson distributed variable $k \sim \text{Poisson}(0.5\lambda^2/0.25^2)$. The observation model for the log-return of an asset $x_t \in \mathbb{R}$ follows

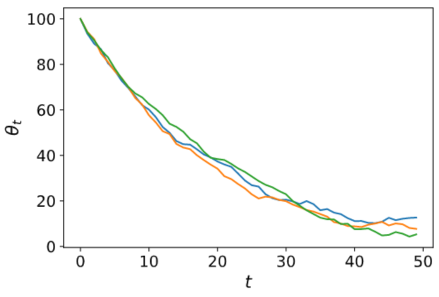
$$x_t = \theta_\mu + \theta_\beta \theta_{v,t} + (\theta_{v,t}^{0.5} + 10^{-5})\varepsilon_t, \tag{4}$$

where $\varepsilon_t \sim \mathcal{N}(0, 1)$; $\theta_\mu = 0$ and $\theta_\beta = 0$ remain the same, while the volatility θ_v follows the transition dynamics, starting with the initial value for the volatility $\theta_{v,0}$ of 1. We set the following priors for LFI of state values: $\theta_\mu \sim \text{Unif}(-2, 2) \in \mathbb{R}, \quad \theta_\beta \sim \text{Unif}(-5, 5) \in \mathbb{R}, \quad \theta_v \sim \text{Unif}(0, 3) \in \mathbb{R}$.

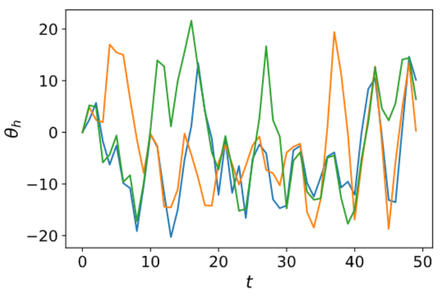
B Technical Details

B.1 Eye Movement Control for Gaze-Based Selection

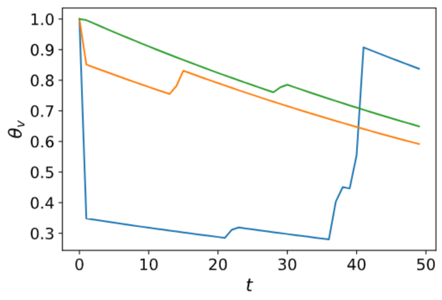
An observation model in the gaze selection experiment is a simulated environment, where a human-subject is modelled through a reinforcement learning agent. In this environment, the agent searches for a target on a 2D display, where the target location, actions, observations, and beliefs of the agent are represented by two coordinates $\{c_1, c_2\}$, $c_1, c_2 \in [-1, 1]$ on the display. At each episode of the environment e, the agent receives noisy observations of the target $\mathbf{o}_e = \mathcal{N}(\mathbf{q}, \theta_s \times E^(e))$ and moves



(a) Linear Gaussian (LG)



(b) Non-Linear Non-Gaussian (NN)



(c) Stochastic Volatility (SV)

Figure 1: Transition trajectories (different colours) of states (y-axis) sampled from three SSM transition dynamics across 50 time-steps (x-axis) with different random seeds. The complexity of the dynamics gradually increases in SSMs, starting with the smooth LG (a) dynamics, where the difference between consecutive states is very small, followed by NN (b) with more erratic behaviour, and finishing with the SV model (c), whose dynamics has occasional drastic changes of state values.

the gaze to a new location $\mathbf{a}_e = \mathcal{N}(PPO(\mathbf{o}_e, \theta_{om} \times A^(e))$. The beliefs $\mathbf{b}_e$ about the target location $\mathbf{q}$ are updated according to

$$\mathbf{b}_{e+1} = \frac{\sigma^2_{(o,e+1)}\mathbf{o}_{e+1} + \sigma^2_{(b,e)}\mathbf{b}_e}{\sigma^2_{(o,e+1)} + \sigma^2_{(b,e)}}, \quad \sigma_{(b,e+1)} = \sqrt{\frac{\sigma^2_{(o,e+1)}\sigma^2_{(b,e)}}{\sigma^2_{(o,e+1)} + \sigma^2_{(b,e)}}}. \tag{5}$$

where $\sigma_{(o,e)}$ and $\sigma_{(b,e)}$ are observation and belief uncertainties respectively, $A^{(e)}$ is the amplitude, and $E^{(e)}$ is the eccentricity of the saccade at the episode e. The user model was trained on 10 000 episodes using the Proximal Policy Optimization (PPO) algorithm [51], provided by the Stable Baselines3 library [52]. We used default parameters, a multilayer perceptron policy and a clipping parameter set to 0.15. The environment was implemented by Chen et al. [53] in Python with the Open AI gym framework [54].

B.2 UMAP Parameterization

In the UMAP parameterization model, the ground truth for state values is not available, because the human-subject cannot specify the ideal embeddings. Therefore, we approximate the ground truth by using ABC with rejection sampling. Usually, this requires running millions of simulations for each time-step, but we make use of the weighted form of the preference score that allows us to use the same simulations across all time-steps. We simulate $1,500,000$ embeddings with state values sampled from the prior, and then calculate their corresponding $\mathcal{U}^{(i)}$ and $\mathcal{P}^{(i)}$. For each time-step t, we calculate the preference scores $\boldsymbol{\delta}$ and retain only 0.06% of those states that showed the lowest $\boldsymbol{\delta}$ values. Finally, we use a Gaussian kernel density estimator (KDE) on the retained state values, and maximize the corresponding PDFs to find the estimations of the ground truth. The bandwidth b of the kernel was calculated according to a Scott's rule [55] $b = n^{-\frac{1}{d+4}}$, where n is the number of data points and d is the number of dimensions.

The preference score was computed as a weighted sum between the relative validity $\mathcal{U}^{(i)}$ and the classification accuracy $\mathcal{P}^{(i)}$ on the validation set. The relative validity $\mathcal{U}^{(i)}$ is an approximation of the Density Based Cluster Validity (DBCV) score [56], which is often used as a quality measure of clustering. Intuitively, it shows how separable all the clusters are. We use the HDBSCAN* package [57] and the HDBSCAN Boruvka KDTree [58] algorithm to cluster the points of the embeddings. We set the smallest size grouping to 60, the density parameter of clusters to 10 and leave the rest parameters to their default values. The classification accuracy $\mathcal{P}^{(i)}$ was calculated for the C-Support Vector Classifier (SVC) [59, 60] with the Scikit-learn package [61] and default parameters.

The embeddings for the UMAP parameterization model were computed for the handwritten digit dataset [62]. The dataset was randomly split on the training and validation sets, resulting in 1198 and 599 8x8 pixel images of digits in 10 digit classes. The UMAP algorithm was implemented with the UMAP-learn package [63].

B.3 Implementation Details of Methods

LFI methods from Sections B.3.1-B.3.3 were integrated in the Engine for Likelihood-Free Inference (ELFI) [64] for application and further development.

B.3.1 BOLFI

GP surrogate. The implementation for the GP surrogate was provided by the GPy package [65]. It was initiated with zero mean function, and with the following priors for the RBF kernel lengthscale $\boldsymbol{l}$, its variance σ_k^2, and the variance of the bias term σ_b^2:

$$l \sim \text{Gamma}\left(\frac{\boldsymbol{\theta}_{\max} - \boldsymbol{\theta}_{\min}}{3}, \mathbf{1}\right), \quad \sigma_k^2 \sim \text{Gamma}\left(\frac{\max(\delta^{(i)})^2}{9}, 1\right), \quad \sigma_b^2 \sim \text{Gamma}\left(\frac{\max(\delta^{(i)})^2}{36}, 1\right). \tag{6}$$

where $\boldsymbol{\theta}_{min}, \boldsymbol{\theta}_{max}$ are the lower and upper bounds for $\boldsymbol{\theta}^{(i)}$, $(\boldsymbol{\theta}^{(i)}, \delta^{(i)})$ are initial training points, and $\mathbf{1}$ is an all-ones vector. The GP was minimized by using the SCG optimizer [66] on the GP negative log-likelihood with a maximum number of iterations of 50. All inputs $\boldsymbol{\theta}^{(i)}$ for the GP were centred and normalized.

LCBSC acquisition. The BOLFI implementation uses LCBSC [67, 68] as an acquisition function, which chooses points that minimize the lower confidence bound (LCB)

$$\text{LCB}(x) = \mu(x) - \beta_t^{1/2}\sigma(x), \quad \beta_t = 2\text{log}(t^{2d+2}\pi^2/3\delta), \tag{7}$$

where β_t is the confidence parameter, $\delta = 0.1$ is the inverse exploration rate and d is an input dimension.

Posterior sampling. The BOLFI posterior was obtained by weighting the prior samples and using corresponding unnormalized likelihoods as weights

$$p(\mathbf{x}^*|\boldsymbol{\theta}) \propto F\left(\frac{\epsilon - \mu(\boldsymbol{\theta})}{\sqrt{\nu(\boldsymbol{\theta}) + \sigma^2}}\right), \tag{8}$$

with $F(\cdot)$ being a Gaussian CDF with the mean 0 and the variance 1, and where ϵ is the minimum of the GP surrogate mean function $\mu(\cdot)$ obtained by using the L-BFGS-B optimizer [69] with a maximum number of iterations of 1000.

B.3.2 Muti-Objective LFI

LMC surrogate. The LMC model was implemented in BoTorch [70]. Its latent GPs were initialized with linear mean functions $\mu(x) = kx + b$, and RBF kernels. The lengthscales of the kernels were parameterized in log scale, and initialized with 0 along with the constant b of the mean function. For the RBF kernel, ARD was also enabled to assign separate lengthscales for each input dimension. GP latents were also initialized with 50 inducing points uniformly sampled inside the input bounds for each latent GP. The LMC training used Adam optimizer from the tensor computation package PyTorch [71] with a learning rate of 0.1 to minimize the variational evidence lower bound (ELBO). The optimized parameters included LMC coefficients, inducing points locations, and hyperparameters for the mean functions and kernels. We used the default training step size and 1000 epochs for training. All inputs for the LMC were centred and normalized.

qEHVI acquisition. The qEHVI acquisition function used a Quasi-MC sampler [72] with scrambled Sobol sequences [73] and a sample size of 128. The reference point that was used for calculating the hypervolume for each objective was set to -5. The acquisition points were acquired sequentially using successive conditioning [74] with a maximum number of restarts of 20, a batch size limit of 10, and a maximum number of iterations for the optimizer of 200.

Bayesian neural network (BNN) [75] was built from stacked Bayesian layers implemented in torch zoo (BLiTZ) [39]. We used an architecture with 2 hidden layers, where each layer had 256 nodes. During the training process, stochastic gradient descent [76] was used with a learning rate of 0.001 for minimizing the ELBO loss with squared L2 norm. The loss was calculated based on 10 samples from the model, for each of 100 batches of training data in a single epoch. The training data was randomly selected from previously stored approximated posterior samples from all states (1000 samples per each state) with replacement, resulting in a total of 1 000 000 points, where one point was a pair of consecutive state values. The training data was updated each time the moving window moved.

Bayesian linear regression (BLR) is defined as $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t^T \boldsymbol{\beta} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim N(0, \sigma^2 I)$. The hyperparameters $\sigma \in \mathbb{R}$ and $\boldsymbol{\beta} \in \mathbb{R}^{m \times m}$ were inferred with maximum likelihood estimation (MLE) [77] of

$$p(\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_t) \propto \frac{1}{\sigma} \exp - \frac{(\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t \boldsymbol{\beta})^T (\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t \boldsymbol{\beta})}{2\sigma_t^2}. \tag{9}$$

The BLR model was implemented with the probabilistic programming package, PyMC3 [78] that used 100 random samples from three latest inference steps. The pairs (θ_{t-1}, θ_t) and $(\theta_{t-2}, \theta_{t-1})$ pairs were used as inputs for training, and the normal distribution was chosen as a prior family of the BLR parameters. The model was fitted by using the No-U-Turn Sampler (NUTS) [79] with two chains of 2000 samples, 2000 tuning iterations, and a target acceptance rate of 0.85.

Posterior sampling. Similar to BOLFI, the posterior was sampled by using an importance-weighted resampling. Because the main model was implemented in PyTorch, we used the Adam optimizer to learn the threshold ϵ. It used a learning rate of 10^{-4} and 100 optimizing iterations. The optimization started at the parameter point that produced a synthetic observation with the smallest discrepancy.

B.3.3 Sequential Neural Estimators

All three SNEs (SNPE, SNLE and SNRE) and their corresponding surrogate models were implemented in the simulation-based inference [80] and PyTorch [71] packages with default parameters. In all three methods, Adam optimizer with the learning rate of 5×10^{-4} and the training batch size of 50 in 20 epochs were used for training the surrogate. The total gradient norm was clipped in order to prevent exploding gradients at a value of 5.0, and z-score passing was used for surrogate model inputs and outputs.

For **SNPE** [81] and **SNLE** [41], the masked autoregressive flow (MAF) surrogate was used for approximating the posterior $p(\boldsymbol{\theta}|\mathbf{x})$ and likelihood, $p(\mathbf{x}|\boldsymbol{\theta})$ respectively. The MAF consisted of 5 transformations with 50 hidden features in each of 2 blocks. Each autoregressive transformation had tanh activation along with batch normalization. In contrast, **SNRE** [42] approximates the ratio $\frac{p(\boldsymbol{\theta},\mathbf{x})}{p(\boldsymbol{\theta})p(\mathbf{x})}$, where a residual network [82] is used as a classifier trained to approximate likelihood ratios. The goal of the classifier is to predict which of the $(\boldsymbol{\theta}, \mathbf{x})$ pairs was sampled from $p(\boldsymbol{\theta}, \mathbf{x})$ and which from $p(\boldsymbol{\theta})p(\mathbf{x})$. The residual network had 50 hidden features in 2 residual blocks with 10 $(\boldsymbol{\theta}, \mathbf{x})$ pairs to use for classification.

B.3.4 Recurrent State-Space Models with GP Transitions

In the experiments, we used two variants of recurrent state-space models with GP transitions: **GP-SSM** with a variationally coupled dynamics and trajectories, in which the variational inference scheme for GP transition dynamics is used [43], and probabilistic recurrent state-space model **PR-SSM** [44], which uses doubly stochastic variational inference for efficient incorporation of latent state temporal correlations. Both methods were implemented in the GPt package [43]. The GPs were using Matern32 kernels with linear mean functions, along with 50 randomly sampled inducing points. The number of latent dimensions was set equal to the number of simulator parameters of the observation model. The ELBO loss was calculated from 10 posterior samples and optimized with Adam using a learning rate of 0.001 in 3000 iterations.