Exploring Exchangeable Dataset Amortization for Bayesian Posterior Inference

Sarthak Mittal [*1 2] **Niels L. Bracher** [*3] **Guillaume Lajoie** [1 2] **Priyank Jaini** [4] **Marcus A. Brubaker** [3 5]

Abstract

Bayesian inference provides a natural way of incorporating uncertainties and different underlying theories when making predictions or analyzing complex systems. However, it requires computationally expensive routines for approximation, which have to be re-run when new data is observed and are thus infeasible to efficiently scale and reuse. In this work, we look at the problem from the perspective of amortized inference to obtain posterior parameter distributions for known probabilistic models. We propose a neural network-based approach that can handle exchangeable observations and amortize over datasets to convert the problem of Bayesian posterior inference into a single forward pass of a network. Our empirical analyses explore various design choices for amortized inference by comparing: (a) our proposed variational objective with forward KL minimization, (b) permutation-invariant architectures like Transformers and DeepSets, and (c) parameterizations of posterior families like diagonal Gaussian and Normalizing Flows. Through our experiments, we successfully apply amortization techniques to estimate the posterior distributions for different domains solely through inference.

1. Introduction

Bayesian analysis of data has become increasingly popular and widely used in numerous scientific disciplines. For example, in politics, predictive models based on public polling and other factors play a crucial role in the discourse around the state of a campaign. Throughout the COVID-19 pandemic, models which estimate the infectiousness of the virus, the efficacy of public health measures, and the future course of the pandemic have become critical to government planning and the public's understanding of the pandemic. In cryogenic electron microscopy (cryo-EM), the posterior over an unknown 3D atomic-resolution molecular structure is explored given the 2D image observations.

Unfortunately, these analyses are frequently burdensome, requiring substantial amounts of computation. Further, these computations often have to be re-run each time new data becomes available, for instance, when new case counts become available or previous measurements are corrected, or when applied to different geographic regions. This leads practitioners to adopt approximations (Welling & Teh, 2011; Gelfand, 2000; Brooks, 1998), simplify their models (Hoffman et al., 2013; Blei et al., 2017) or reduce the frequency with which they perform their analyses.

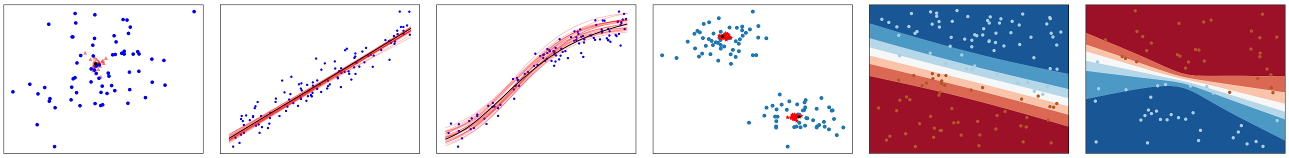
Here, we aim to address this through the use of amortized inference (Morris, 2013; Paige & Wood, 2016; Kingma & Welling, 2013; Rezende et al., 2014; Stuhlmüller et al., 2013) which will allow for efficient and principled methods for posterior analysis. A common thread among the above-mentioned examples is that the probabilistic model defining the relationship between the unknown parameters and the observed data is fixed. Poll aggregation models use hierarchical time series models, infectious diseases are studied using variations on compartment models, and cryo-EM uses a linear image formation model. This makes these models ideal candidates for amortized inference (Kingma & Welling, 2013; Rezende et al., 2014).

Our goal is to learn a function that maps an observed *dataset* to the corresponding posterior distributions without the need to perform explicit Bayesian posterior inference, e.g., with MCMC (Gelfand, 2000; Hoffman et al., 2014). This mapping aims at generalizing to families of datasets. As a result, it is trained with several datasets which are inexpensive to generate via simulation from the probabilistic model on which we want to perform Bayesian inference.

Specifically, we consider the problem of estimating the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ for known probabilistic models $p(\mathbf{x}, \boldsymbol{\theta})$, where $\mathbf{x} \in \mathbb{R}^d$ is observed through n independent and identically distributed (IID) samples $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n$ and $\boldsymbol{\theta} \in \mathbb{R}^k$ denotes the parameters of the model. Given $p(\mathbf{x}, \boldsymbol{\theta})$, it is possible to obtain paired samples $(\mathcal{D}, \boldsymbol{\theta})$ for training through simulations. However, our method is also applica-

[*]Equal contribution [1]MILA [2]Université de Montréal [3]York University [4]Google DeepMind [5]Vector Institute. Correspondence to: Sarthak Mittal <sarthmit@gmail.com>, Niels L. Bracher <nbracher@yorku.ca>.

Accepted to ICML workshop on Structured Probabilistic Inference & Generative Modeling, Honolulu, Hawaii, USA.



(a) Mean of Gaussian (b) Linear Regression (c) Nonlinear Regression (d) Gaussian Mixture (e) Linear Classification (f) Nonlinear Classification

Figure 1. Illustration of our proposed method on different underlying probabilistic models. We see that the learned amortized variational distribution appropriately captures at least a mode of the posterior.

ble in the more realistic setting where parameters $\boldsymbol{\theta}$ are not observed, making it more aligned with real-world settings where the underlying model for data streams is unknown, but we still seek to estimate the posteriors of simple models.

We evaluate the effectiveness of our approach on several domains, including estimating the posterior over (a) the mean of a Gaussian, (b) parameters of a small Bayesian Neural Network (BNN), and (c) the means of a Gaussian Mixture Model (GMM). Our analysis provides insights into the comparisons of various design choices for amortized Bayesian inference, in particular the trade-offs between forward vs. reverse KL as a training objective, the performance obtained using different permutation-invariant architectures for exchangeable data like DeepSets (Zaheer et al., 2017) vs. Transformers (Vaswani et al., 2017), and the kind of approximate posteriors considered like a diagonal Gaussian assumption vs. normalizing flows.

2. Background

For a given probabilistic model $p(\mathbf{x}, \boldsymbol{\theta})$ and observed IID samples $\mathcal{D}$, Bayesian inference refers to the problem of estimating the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$. This estimation problem boils down to an application of Bayes' rule

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\boldsymbol{\theta})}{p(\mathcal{D})} \prod_{i=1}^{n} p(\mathbf{x}_i|\boldsymbol{\theta}). \tag{1}$$

Analytically computing Equation 1 is problematic since the normalization constant requires computing $p(\mathcal{D}) = \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta}, \mathcal{D})\, d\boldsymbol{\theta}$, which is often intractable. Thus, practitioners rely on approximate approaches to estimating the posterior, namely sampling and VI.

VI methods approximate the true posterior $p(\boldsymbol{\theta}|\mathcal{D})$ with a variational distribution $q_\varphi(\boldsymbol{\theta})$ and convert the estimation problem into the following optimization problem

$$\varphi^* = \arg\min_{\varphi} \mathbb{KL}[q_\varphi(\cdot)||p(\cdot|\mathcal{D})] \tag{2}$$

which boils down to optimizing the well-known Evidence Lower-Bound (ELBO)

$$\varphi^* = \arg\max_{\varphi} \mathbb{E}_{\boldsymbol{\theta}\sim q_\varphi(\cdot)} \left[\log \frac{p(\mathcal{D}, \boldsymbol{\theta})}{q_\varphi(\boldsymbol{\theta})}\right]. \tag{3}$$

The reparameterization trick can resolve the dependence on expectation for a class of popular parametric distributions. However, a limitation of this approach is that for a given model, computing the Bayesian posterior for a new dataset requires solving a new optimization problem to learn $q_\varphi(\cdot)$, which is implicitly a function of $\mathcal{D}$.

Variational Autoencoders (VAEs) (Kingma & Welling, 2013; Rezende et al., 2014; Rezende & Mohamed, 2015) bypass this problem in latent-variable models by amortizing the variational distribution explicitly on different data points. That is, they consider $q_\varphi(\mathbf{z}|\mathbf{x}_i)$ where $\mathbf{z}$ is the latent variable and the conditioning is explicitly done on $\mathbf{x}_i$ by predicting the parameters of the distribution from $\mathbf{x}_i$, e.g., $\mathcal{N}\left(\boldsymbol{\mu}_\varphi(\mathbf{x}_i), \boldsymbol{\Sigma}_\varphi(\mathbf{x}_i)\right)$.

Taking inspiration from VAEs and their use of amortization, we turn back to the more general problem of learning Bayesian posteriors for probabilistic models through VI. However, for a given model, we rely on amortization at the dataset level $\mathcal{D}$, instead of a single data point $\mathbf{x}_i$, to obtain the approximate Bayesian posterior directly through inference.

Prior work achieves this objective by either solving the forward KL objective $\mathbb{KL}\left[p(\cdot|\mathcal{D})||q_\varphi(\cdot|\mathcal{D})\right]$ (Radev et al., 2020) or performing Bayesian inference on some latent variables in predictive systems (Garnelo et al., 2018b). The former cannot handle training with data whose underlying model is unknown and hence cannot deal with model misspecification but enjoys the benefits of not requiring a computable likelihood. The latter is predominantly designed for predictive modeling and thus cannot be used to provide information and uncertainty about model parameters. We propose a fully Bayesian approach, which like (Garnelo et al., 2018b) requires a computable and differentiable likelihood and reparameterizable q_φ but approximates the posterior through an explicit form in the parameter space and can be used in cases of model misspecification.

3. Method

Our goal is to approximate the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ given a dataset $\mathcal{D} := \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \subseteq \mathbb{R}^{d\times n}$ where $\mathbf{x}_i \sim p(\mathbf{x}|\boldsymbol{\theta})$. To do this, we learn an amortized distribution $q_\varphi(\boldsymbol{\theta}|\mathcal{D})$ conditioned explicitly on the full dataset.

Objective	q_φ	Model	L_2 *Loss* (↓)							*Accuracy* (↑)			
			Gaussian Mean		**GMM**	**LR**		**NLR**		**LC**		**NLC**	
			5D	*25D*	*2D 2 cl*	*5D*	*100D*	*5D*	*25D*	*5D*	*100D*	*5D*	*25D*
Baseline	-	Random	11.7	54.3	1.8	5.4	69.6	85.1	286.4	50.5	50.0	50.0	49.8
	-	Optimization	1.6	4.6	0.1	0.3	8.6	1.0	30.1	95.6	69.1	92.1	79.3
Fwd.-KL	Gaussian	DeepSets	1.6	4.4	0.8	0.3	50.0	69.4	238.8	80.2	50.1	59.3	57.0
		Transformer	1.6	4.4	0.8	0.3	18.2	66.7	234.2	80.0	63.0	60.2	57.7
Rev.-KL		DeepSets	1.6	4.4	0.1	0.3	25.2	3.2	44.1	90.9	59.7	63.5	59.9
		Transformer	1.6	4.4	0.1	0.3	11.0	2.1	31.3	91.2	66.2	82.2	75.0
Fwd.-KL	Flow	DeepSets	1.6	4.5	0.1	0.3	52.0	57.6	246.7	91.6	50.5	60.9	57.1
		Transformer	1.6	4.4	0.1	0.3	19.2	54.0	200.2	92.2	63.8	60.5	58.2
Rev.-KL		DeepSets	1.6	4.4	0.1	0.3	25.3	3.2	59.6	92.1	59.0	64.7	60.9
		Transformer	1.6	4.4	0.1	0.3	10.6	1.7	31.3	92.4	66.8	83.3	74.5

Table 1. Summary of experimental results. Shown are the results for the following tasks: estimating the mean of a Gaussian with full covariance (Gaussian Mean), estimating the mean of a Gaussian mixture model with full covariance (GMM), (non-)linear regression (NLR/LR) and (non-)linear classification (NLC/LC). The tasks were tackled with different experimental setups, i.e., different parameterized posteriors (diagonal Gaussian and Normalizing Flow) and different inference networks (DeepSets and Transformer). Furthermore, we computed the results for forward and reverse KL objectives and used the prior (Random) and multiple dataset-specific maximum likelihood training (Optimization) as baselines. The L_2 Loss refers to the expected posterior-predictive L_2 loss and Accuracy is the expected posterior predictive accuracy.

Similar to standard VI approaches, we can train q_φ by minimizing the $\mathbb{KL}$ divergence between the approximate and the true posterior, i.e., $\mathbb{KL}[q_\varphi(\cdot|\mathcal{D})||p(\cdot||\mathcal{D})]$ which reduces to maximizing the ELBO

$$\arg\max_{\varphi} \mathbb{E}_{\boldsymbol{\theta}\sim q_\varphi(\cdot|\mathcal{D})}\left[\log \frac{p(\mathcal{D},\boldsymbol{\theta})}{q_\varphi(\boldsymbol{\theta}|\mathcal{D})}\right] \tag{4}$$

While this is the case for VI on a single dataset, we are interested in generalizing to a family of datasets $\{\mathcal{D}_i\}_{i=1}^{K}$. To obtain the posterior distribution for each, we consider a mean-field assumption over the variational distribution q_φ and an IID assumption over the datasets

$$\arg\max_{\varphi} \mathbb{E}_{\theta_i\sim q_\varphi(\cdot|\mathcal{D}_i)}\left[\log \prod_{i=1}^{K} \frac{p(\mathcal{D}_i,\theta_i)}{q_\varphi(\theta_i|\mathcal{D})}\right] \tag{5}$$

In particular, if we have a dataset generating distribution χ, we can re-write Equation 5 as the optimization problem

$$\arg\max_{\varphi} \mathbb{E}_{\mathcal{D}\sim\chi}\mathbb{E}_{\theta\sim q_\varphi(\cdot|\mathcal{D})}\left[\log \frac{p(\mathcal{D},\boldsymbol{\theta})}{q_\varphi(\boldsymbol{\theta}|\mathcal{D})}\right]. \tag{6}$$

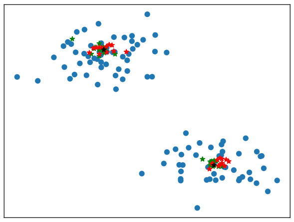
The choice of χ could just be obtained by sampling from $p(n)p(\boldsymbol{\theta})\prod_{i=1}^{n} p(\mathbf{x}_i|\boldsymbol{\theta})$ using ancestral sampling, where n is the dataset cardinality and $p(n)$ is a distribution over positive integers. Thus, given any model, obtaining a dataset-generating distribution by sampling from the model is easy.

For this setup, the choice of q_φ is up to the user. For example, we can model q_ϕ as a Gaussian distribution. This requires learning the mean $\boldsymbol{\mu}_\phi : \mathcal{D} \to \mathbb{R}^m$ and covariance matrix $\boldsymbol{\Sigma}_\phi : \mathcal{D} \to \mathbb{R}^{m\times m}$ of the Gaussian distribution. Importantly, to handle exchangeable datasets, these functions take an arbitrarily sized dataset $\mathcal{D}$ as input and are invariant to permutations in $\mathcal{D}$. The exact same formulation holds for obtaining posteriors when only some observed variables are modeled, e.g., consider the model $p(\mathbf{y},\boldsymbol{\theta}|\mathbf{x})$, where the estimation problem is to approximate $p(\boldsymbol{\theta}|\mathcal{D})$ where $\mathcal{D} = \{(\mathbf{x}_i,\mathbf{y}_i)\}_{i=1}^{n}$.

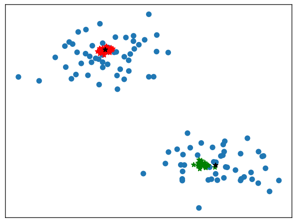
4. Experiments

We consider different well-known probabilistic models $p(\mathcal{D},\boldsymbol{\theta})$ for our experiments and approximate the posterior distribution with the variational distribution $q_\varphi(\boldsymbol{\theta}|\mathcal{D})$. We consider two different families of distributions for q_φ: (a) Gaussian Distribution with a diagonal covariance matrix and (b) Conditional Normalizing Flows. A permutation invariant architecture takes $\mathcal{D}$ as input and outputs the parameters of q_φ, e.g., the mean and diagonal variances in the case of Gaussian distribution. We consider two different permutation invariant architectures: DeepSets (Zaheer et al., 2017) and Transformers, both with approximately the same number of parameters (Appendix B) and train the models by simulating data from $p(\mathcal{D},\boldsymbol{\theta})$ as outlined in Section 3 and consider training with a forward-KL objective as a baseline.

For all our experiments, we sample 100 test datasets and validate the efficacy of different methods by averaging their performance over these datasets. Figure 1 visualizes the zero-shot performance of our proposed method on different modeling problems, and we compare numerical benefits



(a) Forward KL



(b) Reverse KL

Figure 2. Estimating the means of a Gaussian Mixture Model, where red denotes samples from the prediction of the first mean and green of the second mean. The cluster labeling switches in forward KL, implying it can model the multi-modal distribution. In contrast, in reverse KL, it only models one mode, i.e., the first mean always corresponds to the same cluster.

against forward-KL and independent optimization routines across different experimental settings in Table 1.

Mean of Gaussian: As a proof of concept, we consider the simple setup of estimating the posterior distribution over the mean of a Gaussian distribution given some observed data. In this case, the probabilistic model $p(\mathbf{x}, \boldsymbol{\theta})$ is given by the likelihood $p(\mathbf{x}|\boldsymbol{\mu}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and the prior $p(\boldsymbol{\mu}) = \mathcal{N}(\boldsymbol{\mu}|\mathbf{0}, \mathbf{I})$, and $\boldsymbol{\Sigma}$ is known before-hand.

Linear Regression: We then look at the problem of estimating the posterior over the weight vector for Bayesian Linear Regression, where the underlying model is given by the likelihood $p(y|\mathbf{x}, \mathbf{w}) = \mathcal{N}\left(y|\mathbf{w}^T\mathbf{x} + b, \sigma^2\right)$ and the prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, and the task is to estimate $p(\mathbf{w}, b|\mathcal{D})$ with σ^2 known before-hand.

Linear Classification: We now consider a setting where the true posterior cannot be obtained analytically as the likelihood and prior are not conjugate. In this case, we consider the underlying probabilistic model by defining the likelihood as $p(y|\mathbf{x}, \mathbf{W}) = \text{Categorical}\left(y|\mathbf{W}\mathbf{x}\right)$ and the prior as $\mathbf{W} = \mathcal{N}(\mathbf{W}|\mathbf{0}, \mathbf{I})$.

Nonlinear Regression: Next, we tackle the more complex problem where the posterior distribution is multi-modal and obtaining multiple modes or even a single good one is challenging. For this, we consider the model as a BNN for regression with fixed hyper-parameters like the number of layers, dimensionality of the hidden layer, etc. Let the BNN denote the function $f_{\boldsymbol{\theta}}$ where $\boldsymbol{\theta}$ are the network parameters. Then, for regression, we specify the probabilistic model using the likelihood $p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}\left(y|f_{\boldsymbol{\theta}}(\mathbf{x}), \sigma^2\right)$ and the prior $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\mathbf{0}, \mathbf{I})$, where σ^2 is a known quantity, and the estimation problem is to approximate $p(\boldsymbol{\theta}|\mathcal{D})$.

Nonlinear Classification: Similar to regression, we consider BNNs with fixed hyper-parameters for classification problems. In this formulation, we consider the probabilistic model as the likelihood $p(y|\mathbf{x}, \boldsymbol{\theta}) = \text{Categorical}\left(y|f_{\boldsymbol{\theta}}(\mathbf{x})\right)$ and the prior $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\mathbf{0}, \mathbf{I})$, with the same estimation task of approximating $p(\boldsymbol{\theta}|\mathcal{D})$.

Gaussian Mixture Model: While we have mostly looked at predictive problems, where the task is to model some predictive variable y conditioned on some input $\mathbf{x}$, we now look at a well-known probabilistic model for unsupervised learning, Gaussian Mixture Model (GMM), primarily used to cluster data. Consider a $K-$cluster GMM with the likelihood $p(\mathbf{x}|\boldsymbol{\mu}_{1:K}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ and the prior $p(\boldsymbol{\mu}_k) = \mathcal{N}(\boldsymbol{\mu}_k|\mathbf{0}, \mathbf{I})$, assuming known covariance matrices $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k for all clusters k.

5. Discussion and Conclusion

For all our experiments, we consider two permutation-invariant architectures: DeepSets and Transformers, two kinds of variational distributions: diagonal Gaussian and Normalizing Flows, as well as two optimization objectives: forward and reverse KL.

Forward vs. Reverse KL: In our experiments on GMM, which has multiple modes because of the exchangeability of cluster labels, we see that the forward KL objective does lead to learning of a multi-modal distribution. At the same time, reverse KL only captures one mode (Figure 2). However, we also see that in high-dimensional multi-modal settings like learning the parameters of a BNN, the forward KL objective does not lead to learning a reasonable distribution as it attempts to cover all modes. In contrast, the reverse KL objective does not cover multiple modes but can better model one mode, which might be good enough for solving the task (Table 1). Furthermore, unlike forward-KL, the reverse-KL paradigm can be trained without observing $\boldsymbol{\theta}$ but does require a computable and differentiable likelihood.

Transformer vs. DeepSets: We consistently see that using Transformers as the permutation-invariant architecture outperforms DeepSets. We believe that this is because a transformer model allows learning an aggregation function, as opposed to DeepSets, which relies on a fixed aggregation function (in our case, the mean).

Gaussian vs. Flows: We see that increasing the capacity of q_φ with normalizing flows does not help for reverse-KL objective but does for forward-KL. Given the mode-seeking tendency for reverse-KL, we hypothesize that even with the capacity to model different modes, finding both modes is, as expected, challenging but possible (Liu & Wang, 2016; Midgley et al., 2022).

We show that it is possible to amortize full Bayesian posterior inference for a broad class of probabilistic models and explore a variety of design decisions. We believe this approach could provide fast insights into Bayesian posteriors in practice and help accelerate further refinements of the posterior estimates. A key benefit is that it can be trained using both real and simulated data, with or without model misspecification. We believe this is an exciting direction of research that could lead to reducing the load of real-world, complex Bayesian inference problems. Scaling this approach to work on more complex probabilistic models is a significant focus of future work.

Acknowledgements

SM would like to acknowledge the computing resources provided by the Mila cluster to enable the experiments outlined in this work.

References

Ardizzone, L., Bungert, T., Draxler, F., Köthe, U., Kruse, J., Schmier, R., and Sorrenson, P. FrEIA: Framework for easily invertible architectures, 2018. URL https://github.com/vislearn/FrEIA.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Brooks, S. Markov chain monte carlo method and its application. *Journal of the royal statistical society: series D (the Statistician)*, 47(1):69–100, 1998.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real NVP. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017. URL http://arxiv.org/abs/1605.08803.

Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D., and Eslami, S. A. Conditional neural processes. In *International conference on machine learning*, pp. 1704–1713. PMLR, 2018a.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S., and Teh, Y. W. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018b.

Gelfand, A. E. Gibbs sampling. *Journal of the American statistical Association*, 95(452):1300–1304, 2000.

Gordon, J., Bruinsma, W. P., Foong, A. Y., Requeima, J., Dubois, Y., and Turner, R. E. Convolutional conditional neural processes. *arXiv preprint arXiv:1910.13556*, 2019.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. Stochastic variational inference. *Journal of Machine Learning Research*, 2013.

Hoffman, M. D., Gelman, A., et al. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., and Teh, Y. W. Attentive neural processes. *arXiv preprint arXiv:1901.05761*, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kingma, D. P., Welling, M., et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

Kobyzev, I., Prince, S. J., and Brubaker, M. A. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979, 2020.

Liu, Q. and Wang, D. Stein variational gradient descent: A general purpose bayesian inference algorithm. *Advances in neural information processing systems*, 29, 2016.

Lorch, L., Sussex, S., Rothfuss, J., Krause, A., and Schölkopf, B. Amortized inference for causal structure learning. *Advances in Neural Information Processing Systems*, 35:13104–13118, 2022.

Midgley, L. I., Stimper, V., Simm, G. N., Schölkopf, B., and Hernández-Lobato, J. M. Flow annealed importance sampling bootstrap. *arXiv preprint arXiv:2208.01893*, 2022.

Morris, Q. Recognition networks for approximate inference in bn20 networks. *arXiv preprint arXiv:1301.2295*, 2013.

Paige, B. and Wood, F. Inference networks for sequential monte carlo in graphical models. In *International Conference on Machine Learning*, pp. 3040–3049. PMLR, 2016.

Pakman, A., Wang, Y., Mitelut, C., Lee, J., and Paninski, L. Neural clustering processes. In *International Conference on Machine Learning*, pp. 7455–7465. PMLR, 2020.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *The Journal of Machine Learning Research*, 22(1):2617–2680, 2021.

Radev, S. T., Mertens, U. K., Voss, A., Ardizzone, L., and Köthe, U. Bayesflow: Learning complex stochastic models with invertible neural networks. *IEEE transactions on neural networks and learning systems*, 33(4):1452–1466, 2020.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pp. 1278–1286. PMLR, 2014.

Stuhlmüller, A., Taylor, J., and Goodman, N. Learning stochastic inverses. *Advances in neural information processing systems*, 26, 2013.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

Zaheer, M., Kottur, S., Ravanbhakhsh, S., Póczos, B., Salakhutdinov, R., and Smola, A. J. Deep sets. In *Advances in Neural Information Processing Systems*, volume 2017-December, 2017.

A. Related Work

Variational Autoencoders (VAEs). VAEs (Kingma & Welling, 2013; Rezende et al., 2014; Rezende & Mohamed, 2015; Kingma et al., 2019) are latent variable models which model observations $\mathbf{x}$ conditioned on latent variables $\mathbf{z}$ through the joint distribution $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})$ where $p(\mathbf{z})$ is generally chosen as $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Training the model is done through VI where $q_\varphi(\mathbf{z})$ is obtained by explicit amortization over the data point, that is, $q_\varphi(\mathbf{z}|\mathbf{x}) = \mathcal{N}\left(\boldsymbol{\mu}_\varphi(\mathbf{x}), \boldsymbol{\Sigma}_\varphi(\mathbf{x})\right)$. Training this system on a dataset $\mathcal{D}$ is done by similarly optimizing the Evidence Lower-Bound, which boils down to the following optimization problem

$$\arg\max_{\theta,\varphi} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\mathbb{E}_{\mathbf{z}\sim q(\cdot|\mathbf{x})}\left[\log\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\varphi(\mathbf{z}|\mathbf{x})}\right] \tag{7}$$

which can be easily optimized using gradient-based learning and reparameterization trick. While typically, a diagonal Gaussian distribution is considered for q_φ, more complex distributions utilizing normalizing flows can also be used.

Neural Processes. Neural processes (Garnelo et al., 2018a;b; Kim et al., 2019; Pakman et al., 2020; Gordon et al., 2019) (NPs) can be seen as a more flexible and powerful extension of Gaussian processes that leverage a neural network-based encoder-decoder architecture for learning to model a distribution over functions that approximate a stochastic process. However, while we are interested in approximating the posterior distribution over the parameters, neural processes are used to approximate the posterior predictive distribution to make predictions based on observed data. Similar to our setup, NPs rely on amortized VI for obtaining the predictive posterior. Still, instead of working with a known probabilistic model, they train the probabilistic model primarily for prediction-based tasks through approaches analogous to variational expectation maximization. Thus, they cannot provide explicit posterior over the parameters but are more applicable in tasks where we only care about predictive posteriors for tasks that are more akin to supervised learning. NPs, in their most basic form, accomplish this by training for the objective:

$$\arg\max_{\boldsymbol{\theta},\varphi} \mathbb{E}_{\mathcal{D}\sim\chi}\mathbb{E}_{\mathbf{z}\sim q_\varphi(\cdot|\mathcal{D})}\left[\log\frac{p_{\boldsymbol{\theta}}(\mathcal{D},\mathbf{z})}{q_\varphi(\mathbf{z}|\mathcal{D})}\right] \tag{8}$$

where $\mathbf{z} \in \mathbb{R}^p$ is an arbitrary latent variable often uninterpretable, and the parameters of the probabilistic model $\boldsymbol{\theta}$ do not get a Bayesian treatment. In particular, NPs are more suited to modeling datasets of the form $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n$, where all probabilities in Equation 8 are conditioned on the input $\mathbf{x}$'s, and only the predictive over $\mathbf{y}$'s is modeled, and $p_{\boldsymbol{\theta}}$ is modeled as a Neural Network.

BayesFlow. In the case of likelihood-free inference, when the likelihood $p(\mathbf{x}|\boldsymbol{\theta})$ is not available in closed form, BayesFlow (Radev et al., 2020) and similar methods (Lorch et al., 2022) provide a solution framework to amortize Bayesian inference of parameters in complex models. Starting from the forward KL divergence between the true and approximate posteriors, the resulting objective is to optimize for parameters of the approximate posterior distribution that maximize the posterior probability of data-generating parameters $\boldsymbol{\theta}$ given observed data $\mathcal{D}$ for all $\boldsymbol{\theta}$ and $\mathcal{D}$. Density estimation of the approximate posterior can then be done using the change-of-variables formula and a conditional invertible neural network that parameterizes the approximate posterior distribution.

$$\arg\min_{\varphi} \mathbb{KL}[p(\boldsymbol{\theta}|\mathcal{D})||q_\varphi(\boldsymbol{\theta}|\mathcal{D})] = \arg\min_{\varphi=\{\nu,\psi\}} \mathbb{E}_{(\boldsymbol{\theta},\mathcal{D})\sim p(\boldsymbol{\theta},\mathcal{D})}\left[-\log p_{\mathbf{z}}(f_\nu(\boldsymbol{\theta}; h_\psi(\mathcal{D}))) - \log|\det J_{f_\nu}|\right] \tag{9}$$

Since their goal is to learn a global estimator for the probabilistic mapping from $\mathcal{D}$ to data generating $\boldsymbol{\theta}$, the information about the observed dataset is encoded in the output of a summary network h_ψ. It is used as conditional input to the normalizing flow f_ν. Although the likelihood function does not need to be known in this case, the method requires access to paired observations $(\mathbf{x}, \boldsymbol{\theta})$ for training, which is sometimes unavailable.

B. Architecture Details

B.1. Transformer

We use a transformer model (Vaswani et al., 2017) as a permutation invariant architecture by removing positional encodings from the setup and using multiple layers of the encoder model. We append the set of observations with a [CLS] token before passing it to the model and use its output embedding to predict the parameters of the variational distribution. Since no positional encodings or causal masking is used in the whole setup, the final embedding of the [CLS] token becomes invariant to permutations in the set of observations, thereby leading to permutation invariance in the parameters of q_φ.

We use 4 encoder layers with a 256 dimensional attention block and 1024 feed-forward dimensions, with 4 heads in each attention block for our Transformer models to make the number of parameters comparative to the one of the DeepSets model.

B.2. DeepSets

Another framework that can process set-based input is Deep Sets (Zaheer et al., 2017). In our experiments, we used an embedding network that encodes the input into representation space, a mean aggregation operation, which ensures that the representation learned is invariant concerning the set ordering, and a regression network. The latter's output is either used to directly parameterize a diagonal Gaussian or as conditional input to a normalizing flow, representing a summary statistics of the set input.

For DeepSets, we use 4 layers each in the embedding network and the regression network, with a mean aggregation function, ReLU activation functions, and 627 hidden dimensions to make the number of parameters comparative to the one of the Transformer model.

B.3. Normalizing Flows

Assuming that the approximate posterior distribution is Gaussian often leads to poor results as the true posterior distribution can be far from Gaussian shape. To allow for more flexible posterior distributions, we use normalizing flows (Kingma & Dhariwal, 2018; Kobyzev et al., 2020; Papamakarios et al., 2021; Rezende & Mohamed, 2015) for approximating $q_\varphi(\boldsymbol{\theta}|\mathcal{D})$ conditioned on the output of the summary network h_ψ. Specifically, let $g_\nu : \mathbf{z} \mapsto \boldsymbol{\theta}$ be a diffeomorphism parameterized by a conditional invertible neural network (cINN) with network parameters ν such that $\boldsymbol{\theta} = g_\nu(\mathbf{z}; h_\psi(\mathcal{D}))$. With the change-of-variables formula it follows that $p(\boldsymbol{\theta}) = p(\mathbf{z}) \left|\det \frac{\partial}{\partial \mathbf{z}} g_\nu(\mathbf{z}; h_\psi(\mathcal{D}))\right|^{-1} = p(\mathbf{z})|\det J_\nu(\mathbf{z}; h_\psi(\mathcal{D}))|^{-1}$, where J_ν is the Jacobian matrix of g_ν. Further, integration by substitution gives us $d\boldsymbol{\theta} = |\det J_\nu(\mathbf{z}; h_\psi(\mathcal{D})|d\mathbf{z}$ to rewrite the objective from eq. 6 as:

$$\arg\min_{\varphi} \mathbb{KL}[q_\varphi(\boldsymbol{\theta}|\mathcal{D})||p(\boldsymbol{\theta}|\mathcal{D})] = \arg\min_{\varphi} \mathbb{E}_{\mathcal{D}\sim\chi}\mathbb{E}_{\boldsymbol{\theta}\sim q_\varphi(\boldsymbol{\theta}|\mathcal{D})}\left[\log q_\varphi(\boldsymbol{\theta}|\mathcal{D}) - \log p(\boldsymbol{\theta}, \mathcal{D})\right] \tag{10}$$

$$= \arg\min_{\varphi=\{\psi,\nu\}} \mathbb{E}_{\mathcal{D}\sim\chi}\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[\log \frac{q_\nu(\mathbf{z}|h_\psi(\mathcal{D}))}{|\det J_\nu(\mathbf{z}; h_\psi(\mathcal{D}))|} - \log p(g_\nu(\mathbf{z}; h_\psi(\mathcal{D})), \mathcal{D})\right] \tag{11}$$

As shown in BayesFlow (Radev et al., 2020), the normalizing flow g_ν and the summary network h_ψ can be trained simultaneously. The $\mathrm{AllInOneBlock}$ coupling block architecture of the FrEIA Python package (Ardizzone et al., 2018), which is very similar to the RNVP style coupling block (Dinh et al., 2017), is used as the basis for the cINN. $\mathrm{AllInOneBlock}$ combines the most common architectural components, such as ActNorm, permutation, and affine coupling operations.

For our experiments, four coupling blocks, each with a 2-layered non-linear feed-forward subnetworks with ReLU non-linearity and 256 hidden dimensions, define the normalizing flow network. We also include a starting linear transformation to allow for modeling an arbitrary diagonal Gaussian distribution in the first layer.

C. Experiment Details

For all our experiments, we do not yet consider model misspecification and obtain a stream of datasets by simply sampling from χ, where the number of observations n is sampled uniformly from in the range $[64, 128]$. For efficient mini-batching over datasets with different cardinalities, we sample datasets with maximum cardinality (128) and implement different cardinalities by masking out different numbers of observations for different datasets whenever required.

For all our experiments on supervised setups, we sample $\mathbf{x}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ for simplicity, but it is possible to explore other proposal distributions (e.g., heavy-tailed distributions) too. In our Bayesian Neural Networks experiments, we considered a single-layered neural network with Tanh activation function and 32 hidden dimensions. We considered the likelihood function as either a Gaussian or a categorical distribution using the logits, depending on regression and classification.

We do not consider explicit hyperparameter optimization for our experiments and simply use a learning rate of $1\text{e-}4$ with the Adam optimizer (Kingma & Ba, 2014). For smaller experiments like estimating the mean of a Gaussian distribution or linear regression, we trained the model for $25,000$ iterations, while for more complex systems like non-linear regression, we trained the inference model for $100,000$ iterations.