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

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 $q_\phi(\cdot)$, which is implicitly a function of $D$.

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 $q_\phi(z|x_i)$ where $z$ is the latent variable and the conditioning is explicitly done on $x_i$ by predicting the parameters of the distribution from $x_i$, e.g., $\mathcal{N}(\mu_\phi(x_i), \Sigma_\phi(x_i))$.

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 $D$, instead of a single data point $x_i$, to obtain the approximate Bayesian posterior directly through inference.

Prior work achieves this objective by either solving the forward KL objective $\mathbb{KL}[p(\cdot|D)||q_\phi(\cdot|D)]$ (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 $q_\phi$ 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 $p(\theta|D)$ given a dataset $D := \{x_1, x_2, \ldots, x_n\} \subseteq \mathbb{R}^{d \times n}$ where $x_i \sim p(x|\theta)$. To do this, we learn an amortized distribution $q_\phi(\theta|D)$ conditioned explicitly on the full dataset.

### 2. Background

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

$$p(\theta|D) = \frac{p(\theta)}{p(D)} \prod_{i=1}^{n} p(x_i|\theta). \tag{1}$$

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

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

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

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

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

In the more realistic setting where parameters $\theta$ 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.
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.

<table>
<thead>
<tr>
<th>Objective</th>
<th>$q_\phi$</th>
<th>Model</th>
<th>Gaussian Mean</th>
<th>GMM</th>
<th>LR</th>
<th>NLR</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>5D 25D</td>
<td>2D 2 cl</td>
<td>5D 100D</td>
<td>5D 25D</td>
<td>5D 100D</td>
</tr>
<tr>
<td>Baseline</td>
<td>-</td>
<td>Random</td>
<td>11.7 54.3</td>
<td>1.8 5.4 69.6</td>
<td>85.1 284.6</td>
<td>50.5 50.0</td>
<td>50.0 49.8</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>Optimization</td>
<td>1.6 4.6</td>
<td>0.1 0.3 8.6</td>
<td>1.0 30.1</td>
<td>95.6 69.1</td>
<td>92.1 79.3</td>
</tr>
<tr>
<td>Fwd.-KL</td>
<td>Gaussian</td>
<td>DeepSets</td>
<td>1.6 4.4</td>
<td>0.8 0.3 50.0</td>
<td>69.4 238.8</td>
<td>80.2 50.1</td>
<td>59.3 57.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transformer</td>
<td>1.6 4.4</td>
<td>0.8 0.3 18.2</td>
<td>66.7 234.2</td>
<td>80.0 63.0</td>
<td>60.2 57.7</td>
</tr>
<tr>
<td>Rev.-KL</td>
<td>Gaussian</td>
<td>DeepSets</td>
<td>1.6 4.4</td>
<td>0.1 0.3 25.2</td>
<td>3.2 44.1</td>
<td>90.9 59.7</td>
<td>63.5 59.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transformer</td>
<td>1.6 4.4</td>
<td>0.1 0.3 11.0</td>
<td>2.1 31.3</td>
<td>91.2 66.2</td>
<td>82.2 75.0</td>
</tr>
<tr>
<td>Fwd.-KL</td>
<td>Flow</td>
<td>DeepSets</td>
<td>1.6 4.5</td>
<td>0.1 0.3 52.0</td>
<td>57.6 246.7</td>
<td>91.6 50.5</td>
<td>60.9 57.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transformer</td>
<td>1.6 4.4</td>
<td>0.1 0.3 19.2</td>
<td>54.0 200.2</td>
<td>92.2 63.8</td>
<td>60.5 58.2</td>
</tr>
<tr>
<td>Rev.-KL</td>
<td>Flow</td>
<td>DeepSets</td>
<td>1.6 4.4</td>
<td>0.1 0.3 25.3</td>
<td>3.2 59.6</td>
<td>92.1 59.0</td>
<td>64.7 60.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transformer</td>
<td>1.6 4.4</td>
<td>0.1 0.3 10.6</td>
<td>1.7 31.3</td>
<td>92.4 66.8</td>
<td>83.3 74.5</td>
</tr>
</tbody>
</table>

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

$$\text{arg max } \phi \mathbb{E}_{\theta \sim q_\phi(D)} \left[ \log \frac{p(D, \theta)}{q_\phi(\theta | D)} \right]$$ (4)

While this is the case for VI on a single dataset, we are interested in generalizing to a family of datasets $\{D_i\}$. To obtain the posterior distribution for each, we consider a mean-field assumption over the variational distribution $q_\phi$ and an IID assumption over the datasets

$$\text{arg max } \phi \mathbb{E}_{\theta_i \sim q_\phi(\cdot | D_i)} \left[ \log \frac{1}{K} \prod_{i=1}^{K} \frac{p(D_i, \theta_i)}{q_\phi(\theta_i | D_i)} \right]$$ (5)

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

$$\text{arg max } \phi \mathbb{E}_{D \sim \chi} \mathbb{E}_{\theta \sim q_\phi(\cdot | D)} \left[ \log \frac{p(D, \theta)}{q_\phi(\theta | D)} \right].$$ (6)

The choice of $\chi$ could just be obtained by sampling from $p(n)p(\theta) \prod_{i=1}^{n} p(x_i | \theta)$ 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 $q_\phi$ is up to the user. For example, we can model $q_\phi$ as a Gaussian distribution. This requires learning the mean $\mu_\phi: D \rightarrow \mathbb{R}^m$ and covariance matrix $\Sigma_\phi : D \rightarrow \mathbb{R}^{m \times m}$ of the Gaussian distribution. Importantly, to handle exchangeable datasets, these functions take an arbitrarily sized dataset $D$ as input and are invariant to permutations in $D$. The exact same formulation holds for obtaining posteriors when only some observed variables are modeled, e.g., consider the model $p(y, \theta | x)$, where the estimation problem is to approximate $p(\theta | D)$ where $D = \{ (x_i, y_i) \}_{i=1}^{n}$.

4. Experiments

We consider different well-known probabilistic models $p(D, \theta)$ for our experiments and approximate the posterior distribution with the variational distribution $q_\phi(\theta | D)$. We consider two different families of distributions for $q_\phi$: (a) Gaussian Distribution with a diagonal covariance matrix and (b) Conditional Normalizing Flows. A permutation invariant architecture takes $D$ as input and outputs the parameters of $q_\phi$, 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 $p(D, \theta)$ 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...
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 $p(x, \theta)$ is given by the likelihood $p(x|\mu) = \mathcal{N}(x|\mu, \Sigma)$ and the prior $p(\mu) = \mathcal{N}(\mu|0, I)$, and $\Sigma$ is known beforehand.

**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 $p(y|x, \theta) = \mathcal{N}(y|w^T x + b, \sigma^2)$ and the prior $p(w) = \mathcal{N}(0, I)$, and the task is to estimate $p(w, b|D)$ with $\sigma^2$ known beforehand.

**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 $p(y|x, W) = \text{Categorical}(y|Wx)$ and the prior as $W = \mathcal{N}(W|0, I)$.

**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 $f_\theta$ where $\theta$ are the network parameters. Then, for regression, we specify the probabilistic model using the likelihood $p(y|x, \theta) = \mathcal{N}(y|f_\theta(x), \sigma^2)$ and the prior $p(\theta) = \mathcal{N}(\theta|0, I)$, where $\sigma^2$ is a known quantity, and the estimation problem is to approximate $p(\theta|D)$.

**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 $p(y|x, \theta) = \text{Categorical}(y|f_\theta(x))$ and the prior $p(\theta) = \mathcal{N}(\theta|0, I)$, with the same estimation task of approximating $p(\theta|D)$.

**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 $x$, 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 $p(x|\mu_1:k) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$ and the prior $p(\mu_k) = \mathcal{N}(\mu_k|0, I)$, assuming known covariance matrices $\Sigma_k$ and mixing coefficients $\pi_k$ 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 $\theta$ 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 $q_\phi$ 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 models 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


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 \( x \) conditioned on latent variables \( z \) through the joint distribution \( p_\theta(x, z) = p_\theta(x|z)p(z) \) where \( p(z) \) is generally chosen as \( \mathcal{N}(0, I) \). Training the model is done through VI where \( q_\phi(z|x) \) is obtained by explicit amortization over the data point, that is, \( q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x)) \). Training this system on a dataset \( D \) is done by similarly optimizing the Evidence Lower-Bound, which boils down to the following optimization problem

\[
\arg \max_{\theta, \phi} \mathbb{E}_{x \sim D} \mathbb{E}_{z \sim q(\cdot|x)} \left[ \log \frac{p_\theta(x, z)}{q_\phi(z|x)} \right]
\]

which can be easily optimized using gradient-based learning and reparameterization trick. While typically, a diagonal VI where \( q \) is an arbitrary latent variable often uninterpretable, and the parameters of the probabilistic model \( \theta \) do not get a Bayesian treatment. In particular, NPs are more suited to modeling datasets of the form \( D = \{x_i, y_i\}_{i=1}^n \), where all probabilities in Equation 8 are conditioned on the input \( x \)'s, and only the predictive over \( y \)'s is modeled, and \( \p_\theta \) is modeled as a Neural Network.

BayesFlow. In the case of likelihood-free inference, when the likelihood \( p(x|\theta) \) 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 \( \theta \) given observed data \( D \) for all \( \theta \) and \( D \). 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.

\[
\arg \min_\phi \KL[p(\theta|D)||q_\phi(\theta|D)] = \arg \min_\phi \mathbb{E}_{(\theta, D) \sim \p(\theta, D)} \left[ -\log p_\phi(f_\phi(\theta; h_\psi(D))) - \log |\det J_{f_\phi}| \right] \quad (9)
\]

Since their goal is to learn a global estimator for the probabilistic mapping from \( D \) to data generating \( \theta \), the information about the observed dataset is encoded in the output of a summary network \( h_\psi \). It is used as conditional input to the normalizing flow \( f_\phi \). Although the likelihood function does not need to be known in this case, the method requires access to paired observations \((x, \theta)\) 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 \( q_\phi \).
Exploring Exchangeable Dataset Amortization

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 \( q_{\phi}(\theta|D) \) conditioned on the output of the summary network \( h_{\psi} \). Specifically, let \( g_{\nu} : z \mapsto \theta \) be a diffeomorphism parameterized by a conditional invertible neural network (cINN) with network parameters \( \nu \) such that \( \theta = g_{\nu}(z; h_{\psi}(D)) \). With the change-of-variables formula it follows that \( p(\theta) = p(z) |\det \frac{\partial g_{\nu}}{\partial z}(z; h_{\psi}(D))|^{-1} = p(z) |\det J_{\nu}(z; h_{\psi}(D))|^{-1} \), where \( J_{\nu} \) is the Jacobian matrix of \( g_{\nu} \). Further, integration by substitution gives us \( d\theta = |\det J_{\nu}(z; h_{\psi}(D))|dz \) to rewrite the objective from eq. 6 as:

\[
\begin{align*}
\arg \min_{\phi} \mathbb{E}[\mathbb{D}[q_{\phi}(\theta|D)||p(\theta|D)] = \arg \min_{\phi} \mathbb{E}_{\mathbb{D}} \mathbb{E}_{\theta \sim q_{\phi}(\theta|D)} [\log q_{\phi}(\theta|D) - \log p(\theta, D)] \\
= \arg \min_{\phi} \mathbb{E}_{\mathbb{P}} \mathbb{E}_{z \sim p(z)} \left[ \frac{q_{\nu}(z|h_{\psi}(D))}{|\det J_{\nu}(z; h_{\psi}(D))|} - \log p(z; h_{\psi}(D), D) \right]
\end{align*}
\]

As shown in BayesFlow (Radev et al., 2020), the normalizing flow \( g_{\nu} \) and the summary network \( h_{\psi} \) can be trained simultaneously. The 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. 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 \( \chi \), 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 \( \chi = 128 \) 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 \( x_i \sim \mathcal{N}(0, I) \) 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 1e-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.