Amortizing Bayesian Posterior Inference in Tractable Likelihood Models

Anonymous authors
Paper under double-blind review

Abstract

Bayesian inference provides a natural way of incorporating prior beliefs and assigning a probability measure to the space of hypotheses. However, it is often infeasible in practice as it requires expensive iterative routines like MCMC to approximate the posterior distribution. Not only are these methods computationally expensive, but they must also be re-run whenever new observations are available, making them impractical or of limited use. To alleviate such difficulties, we amortize the posterior parameter inference for probabilistic models through permutation invariant architectures. While this paradigm is briefly explored in Simulation Based Inference (SBI), Neural Processes (NPs) and Gaussian Process (GP) kernel estimation, a more general treatment of amortized Bayesian inference in known likelihood models has been largely unexplored. We additionally utilize a simple but strong approach to further amortize on the dimensionality of observations, allowing a single system to infer variable dimensional parameters. In particular, we rely on the reverse-KL based amortized Variational Inference (VI) approach to train inference systems and compare them with forward-KL based SBI approaches across different architectural setups. We conduct thorough experiments to demonstrate the effectiveness of our proposed approach, especially in real-world and model misspecification settings.

1 Introduction

Bayesian analysis of data has become increasingly popular and is widely used in numerous scientific disciplines. 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 that estimate the infectiousness of the virus, the efficacy of public health measures, and the future course of the pandemic became critical to government planning and the public's understanding of the pandemic (Cooper et al., 2020). In cryogenic electron microscopy (cryo-EM), the posterior over an unknown 3D atomic-resolution molecular structure is explored given the 2D image observations (Glaeser et al., 2021).

While recent years have made such methods more accessible (Bingham et al., 2019; Carpenter et al., 2017; Štrumbelj et al., 2023), they still remain computationally burdensome. Further, in practical contexts where new observations are continuously available, the analysis must be re-run every time new data becomes available, e.g., when new case counts become available, previous measurements are corrected, or when applied to different geographic regions. As a result practitioners 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.

A common thread is that the probabilistic model defining the relationship between its parameters and the observations is fixed. Poll aggregation models use hierarchical time series models (Athanasopoulos et al., 2023; Chen et al., 2023), infectious diseases are studied using variants on compartment models (Tang et al., 2020), and cryo-EM uses a linear image formation model (Glaeser et al., 2021). This makes these applications ideal candidates for amortized inference (Morris, 2013; Paige & Wood, 2016; Kingma & Welling, 2013; Rezende et al., 2014; Stuhlmüller et al., 2013).

We propose using neural networks to learn a function that maps an observed *dataset* directly to the corresponding posterior distribution without the need for iterative procedures, e.g., Markov chain Monte

Carlo (MCMC) sampling (Gelfand, 2000; Hoffman et al., 2014). To efficiently handle permutation invariance stemming from the *iid* nature of observations, we rely on efficient set-based architectures like Transformers and DeepSets (Zaheer et al., 2017; Vaswani et al., 2017; Lee et al., 2019). If learned properly, this mapping allows generalization to different datasets for the same underlying model. In addition, we also leverage a simple padding-based procedure to amortize posterior estimation for datasets with a variable number of features.

Our primary motivation is posterior inference as the parametric values themselves are often of interest in applied statistical practice, e.g., for assessing the success of a pandemic intervention or the impact of a factor on public opinion polling. Additionally, we demonstrate the utility of our proposed approach in the closely related problem of posterior prediction, where the goal is to model future predictions given some observations.

Generally, real-world datasets do not exactly follow standard models, e.g., while practitioners often rely on linear models, data rarely follows them exactly. As a result, simulation-based inference (SBI) approaches (Radev et al., 2020; Geffner et al., 2023; Cranmer et al., 2020) which rely on training with datasets and their corresponding known parameters may struggle to generalize to data of practical use. Instead, we propose a new training objective that can operate solely on datasets without knowing the underlying parameters, thereby allowing for a wider diversity of data to be incorporated during training for better generalization to real world settings. We conduct detailed experiments and benchmarking to establish the superiority of our proposed approach as well as their qualitative differences. Our contributions include

- Proposing a general framework for amortizing Bayesian posterior estimation in probabilistic models and highlighting its superior performance to SBI.
- Extending amortization in both SBI and the proposed framework over a variable number of features in each observation, and not just the number of observations.
- Quantifying benefits of our proposed approach when the true underlying model is unknown (model misspecification), as well as providing its advantages in generalization to real-world tabular datasets.
- Benchmarking various design choices like architectural backbones through extensive ablations.

2 Background

To understand our proposed method of amortizing posterior inference without the need for iterative refinement strategies, we first cover some of the important preliminaries as well as approaches already existing in the literature. We also analyze concrete differences between our proposed work and existing prior work on SBI, NPs, and amortized GP kernel estimation. While they all look at related problems of amortized posterior estimation, the underlying methodology and goals in each are different from ours, as outlined below.

Bayesian Inference. Let $\boldsymbol{x} \in \mathbb{R}^d$ denote an experiment observed through a set of independent and identically (*iid*) distributed samples $\mathcal{D} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$. Given these observations, we are often interested in either quantifying the certainty of or generating potential future observations $\boldsymbol{x}_*$. Bayesian Inference provides a natural methodology of quantifying $p(\boldsymbol{x}_*|\mathcal{D})$ by prescribing a space of hypotheses $\boldsymbol{\theta} \in \mathbb{R}^k$ and a *prior* belief $p(\boldsymbol{\theta})$ and *posterior* $p(\boldsymbol{\theta}|\mathcal{D})$ over it. These hypotheses $\boldsymbol{\theta}$ define the *likelihood* of observing a particular outcome, i.e., $p(\boldsymbol{x}|\boldsymbol{\theta})$. The quantity of interest can then be easily expressed through the laws of probability as

$$p(\boldsymbol{x}_*|\mathcal{D}) = \int_{\boldsymbol{\theta}} p(\boldsymbol{x}_*|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta} \tag{1}$$

However, the above expression poses two challenges: (a) the *posterior* $p(\boldsymbol{\theta}|\mathcal{D})$, which is often a quantity of interest in itself, is not known, and (b) even if known, the integration might be intractable. The intractability of the integration is often resolved through Monte Carlo estimation

$$\begin{aligned} p(\boldsymbol{x}_*|\mathcal{D}) &= \mathbb{E}_{\boldsymbol{\theta}|\mathcal{D}} \left[p(\boldsymbol{x}_*|\boldsymbol{\theta}) \right] & (2) \\ &\approx \frac{1}{M} \sum_{m=1}^{M} p(\boldsymbol{x}_*|\boldsymbol{\theta}^{(m)}) & (3) \end{aligned}$$

where $\boldsymbol{\theta}^{(m)} \sim p(\boldsymbol{\theta}|\mathcal{D})$. The quantity $p(\boldsymbol{\theta}|\mathcal{D})$ can be obtained through an application of Bayes rule

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})\, p(\boldsymbol{\theta})}{p(\mathcal{D})} \tag{4}$$

$$= \frac{p(\boldsymbol{\theta})}{p(\mathcal{D})} \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\theta}) \tag{5}$$

Given the form of the *likelihood* and the *prior*, the above distribution is often difficult to sample from, especially with the added complexity of the marginal $p(\mathcal{D}) = \int_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta})\, p(\boldsymbol{\theta})$ being intractable. Additionally, the posterior itself is often of interest on its own, especially in cases where $\boldsymbol{\theta}$ is interpretable, for example, if we model the bias of a coin based on multiple tosses. We refer the readers to Bishop & Nasrabadi (2006) for applications of Bayesian Inference to supervised learning, etc.

Variational Inference. To bypass the intractability of the posterior distribution, or at least the difficulty to sample from it, 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

$$\varphi^* = \arg\min_{\varphi} \mathbb{KL}[q_\varphi(\cdot)||p(\cdot|\mathcal{D})] \tag{6}$$

which is equivalent to optimizing the Evidence Lower-Bound (ELBO) (Gelman et al., 2013)

$$\varphi^* = \arg\max_{\varphi} \mathbb{E}_{\boldsymbol{\theta}\sim q_\varphi(\cdot)} \left[\log \frac{p(\mathcal{D}, \boldsymbol{\theta})}{q_\varphi(\boldsymbol{\theta})}\right] \tag{7}$$

The above optimization procedure finds a member of the considered family of variational distributions that is closest to the true posterior under the *reverse-KL* divergence. Once the optimal parameters φ^* are obtained, the *posterior predictive* distribution $p(\boldsymbol{x}_*|\mathcal{D})$ can be approximated as

$$p(\boldsymbol{x}_*|\mathcal{D}) \approx \mathbb{E}_{q_{\varphi^*}(\boldsymbol{\theta})}\left[p(\boldsymbol{x}_*|\boldsymbol{\theta})\right] \tag{8}$$

The family of distributions q_φ is chosen such that it is easy to sample from. Typical choices include independent multivariate Gaussian distribution (mean-field approximation) or normalizing flows (Rezende & Mohamed, 2015; Papamakarios et al., 2021; Ardizzone et al., 2018-2022).

Amortization. One of the most powerful capabilities of neural networks is their ability to learn and generalize to a wide variety of domains and settings provided sufficient variability during training. For example, Variational Autoencoders (VAEs) define a latent-variable model $p(\boldsymbol{x}, \boldsymbol{z})$ where $\boldsymbol{x}$ represents the observation and $\boldsymbol{z}$ the latent variable. VI typically relies on solving a separate optimization problem $q_{\varphi_i^*}(\boldsymbol{z}_i)$ for each posterior $p(\boldsymbol{z}_i|\boldsymbol{x}_i)$. The cost of learning separate variational approximations can be amortized through training of a joint network $q_\varphi(\boldsymbol{z}|\boldsymbol{x})$, where φ now represents the parameters of a neural network which takes $\boldsymbol{x}$ explicitly as input. The VI procedure then reduces to optimizing φ, which is shared across all observations, as opposed to optimizing for separate $\varphi_i's$, in the hope that $q_\varphi(\boldsymbol{z}_i|\boldsymbol{x}_i) \approx q_{\varphi_i}(\boldsymbol{z}_i)$ for any $\boldsymbol{x}_i$. When modeling using Gaussian distributions, this distinction can be seen as $q_\varphi(\boldsymbol{z}_i|\boldsymbol{x}_i) := \mathcal{N}(\cdot\ ; \boldsymbol{\mu}_\varphi(\boldsymbol{x}_i), \boldsymbol{\Sigma}_\varphi(\boldsymbol{x}_i))$ while $q_{\varphi_i}(\boldsymbol{z}_i) := \mathcal{N}(\cdot\ ; \boldsymbol{\mu}_{\varphi_i}, \boldsymbol{\Sigma}_{\varphi_i})$ (note the functional dependencies). In a similar fashion, Garnelo et al. (2018b) amortize on datasets as explicit inputs, while score-based generative models (Song et al., 2020) amortize on timesteps. Such models are largely successful owing to the generalization capabilities of neural networks to new unseen observations as long as the encoder $q_\varphi(\boldsymbol{z}|\boldsymbol{x}_i)$ is trained on diverse enough observations $\boldsymbol{x}_i'$s.

Simulation-Based Inference. SBI also considers the problem of amortized posterior inference by explicitly conditioning an approximate distribution $q_\varphi(\boldsymbol{\theta}|\mathcal{D})$ on the dataset, where the gap between the approximate and true posterior is bridged through a *Forward*-KL based optimization

$$\arg\min_{\varphi} \mathbb{E}_{\mathcal{D}} \mathbb{KL}\left[p(\boldsymbol{\theta}|\mathcal{D})||q_\varphi(\boldsymbol{\theta}|\mathcal{D})\right] \tag{9}$$

which often leads to mode averaging behavior that can be problematic in high dimensions. While the above objective enjoys applications to tasks without tractable likelihood functions, it can only be used for training when the dataset $\mathcal{D}$ is sampled according to the probabilistic model, and thus cannot utilize off-policy non-simulated data, hindering generalization to real-world scenarios. Precisely because of this, SBI has been leveraged in controlled scenarios like modeling inverse problems with low-dimensional non-differentiable simulators where the likelihood of an observation is not tractable; but it has limited applicability in more general estimation problems like the distribution over weights of a Bayesian Neural Networks.

Neural Processes. NPs also leverage amortized VI in training a latent-variable system for modeling predictive problems. However, unlike our setup, they define an approximate posterior distribution only over an arbitrary latent space and learn how this latent variable impacts the likelihood through point estimation of likelihood parameters. In particular, NPs rely on the Variational-EM setup, where they perform point estimation for the parameters of the likelihood and VI for the latent variable. In contrast, we focus on a similar setting where we instead do a full VI treatment of the parameters of the likelihood function; which in some sense means that our *latent variables* are now parameters of the likelihood model. Thus, our approach can be seen as a fully Bayesian Inference procedure for likelihood models, whereas NPs can be seen as latent-variable models which only provide point estimates for them.

Gaussian Process Kernel Estimation. A specific application of our framework is the estimation of the kernel function for Gaussian Process likelihood models (Liu et al., 2020; Simpson et al., 2021; Bitzer et al., 2023), which leverages amortized inference for tractable likelihoods defined by GP regression setups. In contrast, we provide a more general framework for conducting amortized inference, which we test across a wide variety of domains ranging from supervised to unsupervised learning and from regression to classification tasks. Thus, our proposed approach provides a framework for parameter estimation through amortized variational inference and GP kernel estimation along these lines is a specific application of this approach.

We refer the readers to Appendix A for a detailed discussion about prior work, as well as its connections and differences with our proposed mechanism.

3 Method

At a high level, given any modeling assumption, we are interested in estimating the posterior distribution over its parameters conditioned explicitly over observations to allow for a fast and scalable approximation. That is, given a probabilistic model with parameters $\boldsymbol{\theta}$, we are interested in amortizing and approximating the full Bayesian inference solution over $\boldsymbol{\theta}$. Our goal is to train a system that approximates the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ given a dataset $\mathcal{D} := \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n\} \subseteq \mathbb{R}^d$ where $\boldsymbol{x}_i \sim p(\boldsymbol{x}|\boldsymbol{\theta})$. We achieve this by learning an amortized variational distribution $q_\varphi(\boldsymbol{\theta}|\mathcal{D})$ conditioned explicitly on the dataset $\mathcal{D}$. Similar to standard VI approaches, we can train q_φ through the following optimization problem

$$\arg\min_{\varphi} \mathbb{KL}\left[q_\varphi(\cdot|\mathcal{D})||p(\cdot|\mathcal{D})\right] \tag{10}$$

While this is the case for VI on a single dataset, we are interested in generalizing to a family of datasets χ. Similar to how a VAE encoder efficiently generalizes to approximate posteriors for new images when trained across multiple image observations, we amortize over datasets by training over multiple $\mathcal{D} \sim \chi$ in the hope of generalizing to posterior approximations in new datasets. Thus, given a probability measure over the space of datasets χ, we posit learning of shared parameters φ of the variational distribution across different datasets as

$$\arg\min_{\varphi} \mathbb{E}_{\mathcal{D}\sim\chi}\mathbb{KL}\left[q_\varphi(\boldsymbol{\theta}|\mathcal{D})||p(\boldsymbol{\theta}|\mathcal{D})\right] \tag{11}$$

which equivalently reduces to maximizing the ELBO:

$$\arg\max_{\varphi} \mathbb{E}_{\mathcal{D}\sim\chi}\mathbb{E}_{\boldsymbol{\theta}\sim q_\varphi(\cdot|\mathcal{D})}\left[\log \frac{p(\mathcal{D},\boldsymbol{\theta})}{q_\varphi(\boldsymbol{\theta}|\mathcal{D})}\right] \tag{12}$$

To recap, given a known likelihood model $p(\boldsymbol{x}|\boldsymbol{\theta})$, with $\boldsymbol{\theta}$ denoting its parameters; we are interested in finding the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ for any dataset $\mathcal{D}$. While VI strategies approach this problem by learning a separate q_φ for each $\mathcal{D}$, we instead train a shared q_φ that explicitly takes $\mathcal{D}$ as input and, in doing so, possesses the ability to generalize to new datasets without additional training.

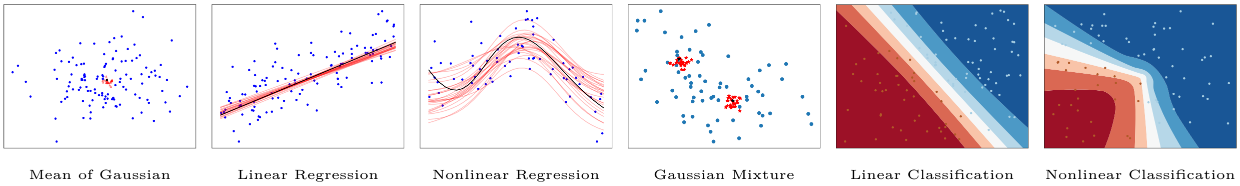


Figure 1: **Amortized Bayesian Posterior Estimation:** Illustration of predictions from proposed approach when trained on a fixed dimensional observation space. Model predictions, true predictions and sample points are shown in red, black and blue. Additionally for classification, we label sample points with their ground-truth class, and draw the decision boundary according to the model.

This training paradigm naturally introduces a dependency on the dataset generating distribution χ. Since we are working with a known probabilistic model, an obvious choice of χ is to treat this probabilistic model as a black-box simulator, akin to $p(n)p(\boldsymbol{\theta})\prod_{i=1}^{n}p(\boldsymbol{x}_i|\boldsymbol{\theta})$, samples from which can be obtained using ancestral sampling. Here, n is the dataset cardinality and $p(n)$ is a distribution over positive integers. Thus, obtaining at least one dataset-generating distribution is easy given any probabilistic model. However, χ can also be obtained from other sources, for example, a stream of real-world data, through interventions in the data-generating process, or through bagging on a large real-world dataset.

Another design choice in the above setup is the parameterization for q_φ: we explore a Gaussian distribution and a normalizing flow parameterization with either a Transformer or DeepSets architectural backbone to process $\mathcal{D}$. It is important to note that not all deep learning architectures are amenable in this setting since, given the *iid* nature of the samples, the posterior is permutation invariant

$$p(\boldsymbol{\theta}|\mathcal{D}) = p(\boldsymbol{\theta}|\Pi\mathcal{D}) \tag{13}$$

for some permutation matrix Π. This constraint should thus be reflected when modeling the approximation. To satisfy it, for example for a Gaussian distribution, we can define q_φ as

$$q_\varphi(\cdot) = \mathcal{N}\left(\cdot|\boldsymbol{\mu}_\varphi(\mathcal{D}), \boldsymbol{\Sigma}_\varphi(\mathcal{D})\right) \tag{14}$$

where $\boldsymbol{\mu}_\varphi$ and $\boldsymbol{\Sigma}_\varphi$ are modeled using permutation invariant architectures, thus satisfying the desired constraint

$$q_\varphi(\cdot|\mathcal{D}) = q_\varphi(\cdot|\Pi\mathcal{D}) \tag{15}$$

In contrast to SBI, which relies on a *Forward KL* optimization, our proposed amortized VI approach relies on the *Reverse KL* objective.[1] We defer details regarding the permutation invariance of architectures to Appendix B.

3.1 Amortizing Variable Feature Dimensions

So far, we have only considered amortization over different datasets for the same underlying likelihood model. For example, for a 2-dimensional Bayesian linear regression model, the amortized posterior $q_\varphi(\boldsymbol{\theta}|\mathcal{D})$ approximates the true posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ for arbitrary sets of 2-dimensional observations. It is important to note that a deep learning-based approach leaves hopes of generalizing to new datasets because the underlying functional form of the solution remains constant across different datasets and is given by the solution obtained from Equation 4.

We note that the functional form of the ground-truth solution remains the same even with varying dimensionality. That is, the form of the solution remains the same irrespective of whether the underlying problem is a 2- or 100-dimensional Bayesian linear regression problem. Additionally, we can see that a low-dimensional problem can just be embedded into a high-dimensional space, with the extra features and parameters set to 0. This simple but strong insight allows us to amortize q_φ over datasets with varying dimensionalities by embedding them in a higher dimensional space.

3.2 Handling Model Misspecification

Knowledge about the true likelihood model underlying the data-generating scheme is often unknown. Practitioners thus rely on making a modeling assumption by prescribing a particular likelihood model for

---

[1] Given the equivalences, we will use amortized VI and Reverse KL interchangeably.

Objective	q_φ	Model	*L_2 Loss* (↓)							*Accuracy* (↑)			
			Gaussian Mean		**GMM**	**LR**		**NLR**		**LC**		**NLC**	
			2D	*100D*	*5D 2 cl*	*1D*	*100D*	*1D*	*25D*	*2D*	*100D*	*2D*	*25D*
Baseline	-	Random	5.834	301.23	5.06	4.056	200	64.5	793	49.8	50.0	50.1	49.8
	-	Optimization	1.989	101.24	0.42	0.258	22.1	0.31	95.0	96.5	71.2	97.5	80.1
	-	MCMC	2.085	104.55	0.65	0.285	26.5	N/A	106	94.4	64.6	96.4	73.2
Fwd-KL	Gaussian	DeepSets	2.014	103.34	2.42	0.264	127	49.9	682	81.0	50.0	59.5	59.6
		Transformer	2.015	102.80	2.44	0.264	46.1	50.7	678	81.0	63.3	59.9	59.8
Rev-KL		DeepSets	2.012	102.64	0.48	0.263	64.6	0.42	124	94.1	62.4	91.6	62.6
		Transformer	2.013	102.59	0.52	0.264	28.1	0.41	99.2	94.0	68.2	91.7	76.5
Fwd-KL	Flow	DeepSets	2.013	103.34	0.64	0.264	127	15.3	549	96.0	50.1	62.0	61.0
		Transformer	2.018	102.73	0.57	0.265	44.8	16.3	530	96.2	64.7	75.7	61.1
Rev-KL		DeepSets	2.010	102.67	0.52	0.263	75.7	0.39	125	95.1	58.6	93.1	63.9
		Transformer	2.014	102.52	0.47	0.264	30.4	0.38	97.8	95.1	68.9	92.9	75.8

Table 1: **Amortized Bayesian Posterior Estimation**. Results for estimating the mean of a Gaussian (Gaussian Mean), means of a Gaussian mixture model (GMM), (non-)linear regression (NLR/LR) and (non-)linear binary classification (NLC/LC). We ablate over permutation invariant architectures (DeepSets vs Transformers) and the density parameterizations (Diagonal Gaussian vs Normalizing Flows). Our baselines include the prior (Random), dataset-specific maximum likelihood training (Optimization), Langevin-based MCMC and Forward-KL based SBI. We use L_2 loss and expected accuracy according to the posterior predictive as metrics.

the data and then fitting its parameters to best explain the data. For example, the underlying model of whether an email is spam or not may be unknown, but one can still assume a linear model to solve this classification problem. This defines the basis of model misspecification, where there is a mismatch between the underlying true model and the assumption used.

In contrast to SBI approaches, our method can efficiently handle model misspecification. This can be seen from the contrast between Equations 9 and 12, where the former can only be trained when χ is defined according to the modeling choice while the latter doesn't put any constraints on χ. Thus, in the presence of a stream of real-world data, SBI cannot leverage this data during training because the corresponding parameters in the assumed likelihood model are not known, while our method can leverage it during training and can thus lead to more robust predictions on data that is of actual practical significance.

A simple example of this setup is when the true data might be coming from a Gaussian Process, but we might not know this. To still be able to efficiently make predictions, we can model it with a linear relation. However, SBI approaches can only train the amortized posterior for the linear model using linear data, while our method can actually leverage the GP data that we observe. This leads to better generalization to linear modeling of GP data and can be further leveraged in other model misspecification setups.

4 Experiments

Our approach relies on the user specifying a probabilistic model that describes their belief about the data-generating process as well as the parameters of interest. To showcase the wide applicability of our approach, we perform experimentation on various well-known probabilistic models encompassing supervised and unsupervised scenarios. In particular, we look at the problem of estimating the Bayesian posterior over the (a) mean of a Gaussian distribution (GM), (b) means of a Gaussian mixture model (GMM), (c) parameters of a (non-)linear regression model (NLR/LR), and (d) parameters of a (non-)linear classification model (NLC/LC). We refer the readers to Appendix C for particulars about the probabilistic models, including their likelihoods and the priors considered. Throughout our experiments, we observe superior performance of our proposed approach in terms of predictive performance metrics (Appendix D), especially in problems with high-dimensional and multi-modal posteriors.

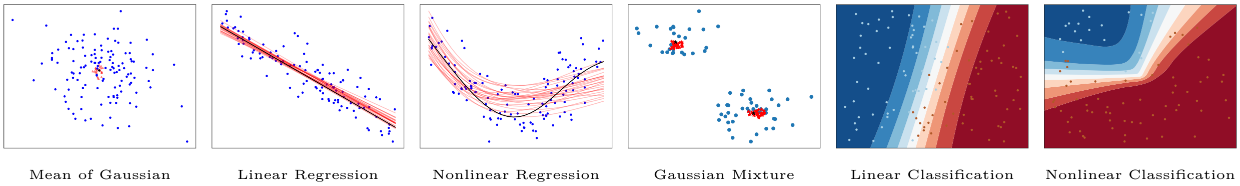


Figure 2: **Variable-Dimension Visualization:** Illustration of proposed approach when trained on a variable observational space but visualized on low-dimensional tasks, with model predictions, ground truth prediction and data points in red, black and blue. For classification, data points are colored by their ground-truth labels, and decision boundary corresponds to the model prediction.

In all our experiments, we generally consider two types of baselines: dataset-specific and amortized. For dataset-specific baselines, we use the prior (Random), perform maximum likelihood estimation using gradient-based optimization (Optimization) as well as an approximate Bayesian inference procedure through Langevin-based MCMC sampling, which also uses the gradient information (MCMC). Such baselines rely on iterative procedures and must be run independently for different datasets. For amortized baselines, we consider the SBI-based forward-KL baseline as outlined in Equation 9. We refer the readers to Appendices F, D, and H for details about the experiments, metrics, and additional results, respectively.

Zero-Shot Posterior Approximation.

Given a known probabilistic model $p(\mathcal{D}, \boldsymbol{\theta})$, we train an amortized inference system $q_\varphi(\cdot|\mathcal{D})$ to approximate the often intractable posterior over the parameter $\boldsymbol{\theta}$ for different datasets with varying cardinality. Figure 1 visualizes the performance of the amortized model on different probabilistic models spanning supervised and unsupervised learning for low dimensional problems. We see that the proposed approach provides reliable samples of the parameters for previously unseen datasets, showing that it can indeed generalize.

Next, we empirically evaluate the performance of our proposed approach in more complex, high-dimensional setups for the same set of probabilistic models. We set the data-generating distribution χ to be according to the probabilistic model in order to allow comparisons with SBI, which can only be trained in this particular setting. Table 1 highlights the performance of our proposed approach in high-dimensional frameworks, showcases its superior performance to similar SBI benchmarks, and compares it to dataset-specific non-amortized baselines. For multiple high-dimensional problems, we see that the proposed amortized posterior performs similarly to Optimization and MCMC without being iterative at inference.

Generalizing to Variable Feature Dimensions

Next, we turn our attention to performing amortization additionally over different datasets *with a variable number of feature dimensions*, as highlighted in Section 3.1. To do this, we embed all low-dimensional problems in a 100-dimensional space and then train an amortized VI system. A key benefit of this approach is that instead of having to train different models for different dimensional problems, we can rely on a single model that can generalize across all. We visualize samples from the approximate posterior for low-dimensional problems in Figure 2 and further provide quantitative metrics for high-dimensional problems in Table 2. Our results highlight that the proposed approach can generalize quite well to variable dimensional setups without requiring an iterative procedure at inference.

Handling Model Misspecification

As discussed in Section 3 and prior work (Müller et al., 2021; Hollmann et al., 2022), the performance and use-case of such amortized systems relies heavily on the choice of the data-generating distribution χ. So far, we only look at cases where χ defines simulations according to the probabilistic model to allow for a fair comparison between SBI and our proposed method. This setup, however, implicitly assumes that we know the ground-truth data-generating process, which is often not the case. A more common scenario is where we do not know the data-generating process but have access to samples from it defined through χ_{real}. To make predictions for new query points, we, as statisticians, assume a probabilistic model $p(\boldsymbol{x}|\boldsymbol{\theta})$ to explain the data, which may be quite far from the true underlying model.

	q_φ	Model	*L_2 Loss (↓)*							*Accuracy (↑)*			
			GM		**GMM**	**LR**		**NLR**		**LC**		**NLC**	
			2D	*100D*	*5D 2 cl*	*1D*	*100D*	*1D*	*50D*	*2D*	*100D*	*2D*	*50D*
Baseline	-	Random	6.300	298.7	4.68	4.380	202	59	1609	50.0	49.9	50.1	49.9
	-	Optimization	2.020	100.8	0.42	0.257	20.9	0.3	289.4	97.1	71.9	96.4	74.9
	-	MCMC	2.213	108.4	0.76	0.418	32.9	N/A	300.3	94.0	63.7	96.2	67.1
Fwd-KL	Gaussian	DeepSets	2.218	130.2	2.36	0.269	152	45	1333	81.0	50.6	49.8	50.0
		Transformer	2.369	108.8	2.39	0.276	65.1	45	1310	80.1	62.3	58.6	58.8
Rev-KL		DeepSets	2.050	104.9	0.48	0.278	66.9	0.7	449.6	94.0	61.5	89.6	61.4
		Transformer	2.059	104.7	0.47	0.271	33.3	0.6	282.7	94.2	67.8	89.1	73.0
Fwd-KL	Flow	DeepSets	2.055	130.9	0.56	0.271	157	36	1103	94.9	50.3	50.1	49.7
		Transformer	2.072	108.5	0.53	0.274	62.3	33	1066	92.2	63.6	60.2	60.0
Rev-KL		DeepSets	2.058	105.2	0.50	0.275	78.3	0.7	480.3	94.9	59.1	88.9	62.6
		Transformer	2.051	104.8	0.46	0.268	33.2	0.8	257.2	95.0	68.2	89.8	71.9

Table 2: **Variable-Dimension Posterior Prediction**. Results for estimating the mean of a Gaussian (Gaussian Mean), means of a Gaussian mixture model (GMM), (non-)linear regression (NLR/LR) and (non-)linear binary classification (NLC/LC). In this setting, for each task only a single model was trained to solve for different dimensional problems.

We first note that χ_{real} cannot be used to train a forward KL (SBI) model and is thus unusable in this case. To resolve this issue, let χ_{sim} denote the data generating distribution obtained from simulations via the assumed model, which can then be used to train the forward KL model. We compare this SBI setup with our proposed approach, either trained on χ_{sim} or χ_{real} but always evaluated on χ_{real} since it is the family of distributions of interest to us.

χ_{real} (→)	q_φ	*Data*	Linear	MLP Nonlinear	GP Nonlinear
χ_{sim} (→)		*Model*	NLR	LR	NLR
Fwd-KL	Gaussian		15.254	5.089	14.867
Rev-KL			0.380	2.431	0.153
+ switched data			0.369	1.477	0.072
Fwd-KL	Flow		7.575	1.964	8.355
Rev-KL			0.345	1.945	0.113
+ switched data			0.346	1.480	0.060

Table 3: **Model Misspecification:** We see benefits of our proposed approach when the true underlying data generating process is not known. χ_{real} denotes the real data-generating distribution, χ_{sim} the simulated one according to the (wrongly) assumed model. We train different models on χ_{sim}, with switched data denoting training on χ_{real}. Evaluation is done solely on χ_{real}.

We consider χ_{real} and χ_{sim} to be data-generating processes, each simulated according to the probabilistic models corresponding to linear, MLP, and Gaussian Process regression such that χ_{real} and χ_{sim} are never the same. Table 3 showcases the experimental results of model misspecification, where we see that the forward KL models suffer a lot due to mismatch between the training and evaluation data, compounded by the potential overestimation of variance (mode-averaging) while our proposed method significantly outperforms it. Even further, we see that the proposed method can be directly trained on χ_{real}, leading to even better performance. For example, when χ_{real} denotes data generated from a Gaussian Process and the probabilistic model assumed is MLP-based, our proposed approach outperforms forward KL and does even better when trained on the GP-based data as opposed to simulations from the MLP. We refer the readers to Appendix F.3 and H.3 for experimental details and additional results on model misspecification.

Applications to Tabular Domains

While we see clear benefits in simulated settings, we next turn our attention to real-world scenarios. We consider a suite of tasks from the OpenML platform for both regression and binary classification and filter out tasks from the *OpenML-CTR23 - A curated tabular regression benchmarking suite* (Fischer et al., 2023) for regression and *OpenML-CC18 Curated Classification benchmark* (Bischl et al., 2019) for classification problems (details in Appendix G). We end up with 9 regression and 13 classification datasets with varying

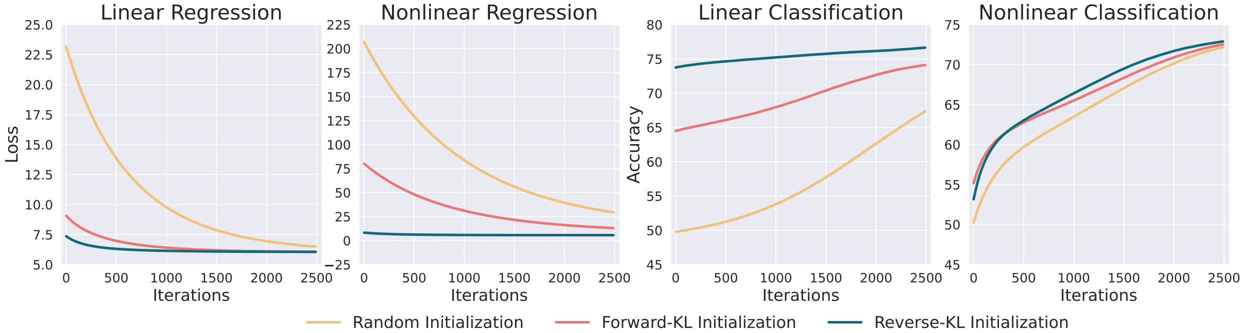


Figure 3: **Tabular Experiments:** Initializing parameters from the amortized model, especially Reverse KL, leads to good zero-shot performance across (non-)linear regression and classification. These benefits persist over the course of optimization.

number of features and use the amortized inference systems trained on simulated data with variable feature dimensions to predict the parameters of interest.

After initializing from the inference model, we further train the parameters of the respective probabilistic models with maximum a-posteriori objective and compare the performance with a corresponding model initialized from the prior. Even in this extreme case of domain shift, we see in Table 4 that the amortized model provides good initializations for different real-world tasks zero-shot after training. Figure 3 also shows that the amortized model surpasses the SBI baseline in terms of performance and convergence speed for both linear and nonlinear setups. While Figure 3 only provides a normalized aggregated performance over all datasets considered for a q_φ parameterized as a diagonal Gaussian, we refer the readers to Appendix H.4 for results on individual datasets with different q_φ, as well as Appendix F.4 for implementation details.

			L_2 Loss (↓)		*Accuracy* (↑)	
	q_φ	**Model**	LR	NLR	LC	NLC
Baseline	-	Random	23.16	206.86	49.78	50.27
Fwd-KL	Gaussian	DeepSets	10.86	93.77	67.29	50.31
		Transformer	9.90	104.15	63.06	55.05
Rev-KL		DeepSets	9.21	8.88	77.51	54.41
		Transformer	8.63	12.08	74.07	54.88
Fwd-KL	Flow	DeepSets	13.21	70.57	66.83	49.58
		Transformer	9.07	80.18	64.49	55.18
Rev-KL		DeepSets	11.18	9.28	78.19	52.31
		Transformer	7.36	8.27	73.75	53.16

Table 4: **Tabular Experiments**: Zero-shot performance of the amortized variable-dimensional models across (non-)linear regression and classification real-world tabular tasks.

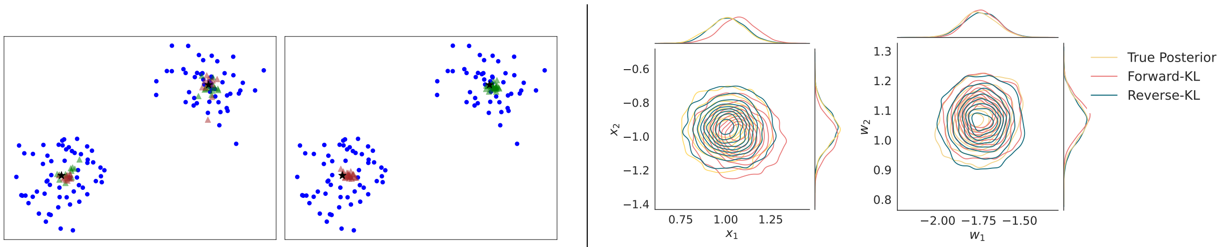
5 Discussion

Having presented a method to amortize posterior estimation for different probabilistic models and the corresponding empirical evidence, we now highlight some of our key findings.

Posterior Inference. While comparing with the true posterior is hard due to its intractability, it is still available for the problem of estimating the mean of a Gaussian distribution as well as for Bayesian Linear Regression. We plot the kernel density estimate of the samples obtained from the true posterior, the amortized forward, and the reverse KL model in Figure 4 (**Right**), which shows that both the amortization setups efficiently capture the true posterior distribution. Further, we quantify the quality of the approximate posteriors through a symmetric KL Divergence metric in Table 5, showing that the amortized model obtains good approximations but does worsen with increasing dimensionality. Additionally, our posterior predictive results show that forward KL often performs comparable to

			Symmetric KL Divergence (↓)			
	q_φ	**Model**	**Gaussian Mean**		**LR**	
			2D	*100D*	*1D*	*100D*
Baseline	-	Random	44.32	46.78	182.4	184.8
Fwd-KL	Gaussian	DeepSets	0.032	0.253	0.044	83.8
		Transformer	0.031	0.068	0.041	20.7
Rev-KL		DeepSets	0.037	0.215	0.039	92.2
		Transformer	0.027	0.062	0.040	33.5

Table 5: **Normalized Symmetric KL Divergence**. Amortized models approximate the true posterior well in tasks with tractable posteriors, when compared to the prior (Random).



(a) Forward-KL (b) Reverse-KL (c) Mean of Gaussian (d) Linear Regression

Figure 4: **Left**: Estimation of the means of a GMM, where red and green samples denote the first and second mean vectors. Unlike in reverse KL, the cluster labels switch in forward KL, highlighting its ability to capture underlying multi-modality. **Right**: Kernel density estimation of the true posterior, overlaid with estimates from forward and reverse KL systems, for different probabilistic models.

random chance (prior) in high-dimensional, multi-modal setups (Appendix H.1). In contrast, reverse KL provides reasonable estimates, alluding to its ability to better capture at least a mode of the posterior.

Design Choices. We consistently see that Transformers outperform DeepSets as the permutation-invariant backbone, potentially because they do not rely on a fixed aggregation scheme (sum or mean pooling) but instead learn it in a context-aware manner. However, we do see that in some rare cases of OoD generalization, DeepSets can outperform Transformers (Appendix H.3).

Next, we see that increasing the capacity of q_φ with normalizing flows only helps marginally for the reverse KL objective but substantially for the forward KL setup. We hypothesize that given the mode-seeking tendency of reverse KL, even with the capacity to model different modes, the algorithm seeks and latches to only a single mode and capturing multiple modes in this setup is challenging, whereas in forward KL setup without additional capacity the model overestimates the variance a lot.

Forward vs Reverse KL. Our experiments on GMM show that forward KL objective leads to the learning of a multi-modal distribution while reverse KL only captures one mode (Figure 4, **Left**). However, in high-dimensional multi-modal settings like BNNs, the former does not lead to learning of a reasonable distribution as it attempts to cover all the modes while the latter does not cover multiple modes but better models an individual mode (Tables 1 and 7; Appendix H.1 and H.2). Furthermore, unlike forward KL, the reverse KL paradigm can be trained without observing $\boldsymbol{\theta}$ but does require a known differentiable likelihood. We show an application of this setting where we don't have access to the ground-truth model but only to samples from it.

6 Conclusion

We show that Bayesian posterior inference can be amortized for a broad class of probabilistic models and explore a variety of design decisions. In particular, we show that reverse KL is effective for learning the amortization network and has significant benefits in the presence of model misspecification and generalization to out-of-domain real-world setups. It provides an exciting direction of research which could reduce the load of real-world, complex, and iterative approximations through quick and cheap inference over a trained amortized network. We believe that scaling our approach to more complex probabilistic models, and combining with diffusion-based variational systems (Zhang & Chen, 2021; Berner et al., 2022; Vargas et al., 2023) would be important directions for future work.

Impact Statement

This work studies amortizing variational inference for Bayesian posterior estimation which is a widespread strategy for performing inference in statistics. It provides a natural way of quantifying uncertainty and potentially leading to more robust predictions. While we do not foresee any negative impacts of progress in this area, we encourage caution when applying the methodologies in practice.

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`.

Ardizzone, L., Bungert, T., Draxler, F., Köthe, U., Kruse, J., Schmier, R., and Sorrenson, P. Framework for Easily Invertible Architectures (FrEIA), 2018-2022. URL `https://github.com/vislearn/FrEIA`.

Arenz, O., Dahlinger, P., Ye, Z., Volpp, M., and Neumann, G. A unified perspective on natural gradient variational inference with gaussian mixture models. *arXiv preprint arXiv:2209.11533*, 2022.

Athanasopoulos, G., Hyndman, R. J., Kourentzes, N., and Panagiotelis, A. Forecast reconciliation: A review. *International Journal of Forecasting*, 2023. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2023.10.010. URL `https://www.sciencedirect.com/science/article/pii/S0169207023001097`.

Berner, J., Richter, L., and Ullrich, K. An optimal control perspective on diffusion-based generative modeling. *arXiv preprint arXiv:2211.01364*, 2022.

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P., and Goodman, N. D. Pyro: Deep universal probabilistic programming. *The Journal of Machine Learning Research*, 20(1):973–978, 2019.

Bischl, B., Casalicchio, G., Feurer, M., Hutter, F., Lang, M., Mantovani, R. G., van Rijn, J. N., and Vanschoren, J. Openml benchmarking suites. *arXiv:1708.03731v2 [stat.ML]*, 2019.

Bishop, C. M. and Nasrabadi, N. M. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Bitzer, M., Meister, M., and Zimmer, C. Amortized inference for gaussian process hyperparameters of structured kernels. *arXiv preprint arXiv:2306.09819*, 2023.

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.

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M. A., Guo, J., Li, P., and Riddell, A. Stan: A probabilistic programming language. *Journal of statistical software*, 76, 2017.

Chauhan, V. K., Zhou, J., Lu, P., Molaei, S., and Clifton, D. A. A brief review of hypernetworks in deep learning. *arXiv preprint arxiv:2306.06955*, 2023.

Chen, Y., Garnett, R., and Montgomery, J. M. Polls, context, and time: A dynamic hierarchical bayesian forecasting model for us senate elections. *Political Analysis*, 31(1):113–133, 2023. doi: 10.1017/pan.2021.42.

Cooper, I., Mondal, A., and Antonopoulos, C. G. A sir model assumption for the spread of covid-19 in different communities. *Chaos, Solitons & Fractals*, 139:110057, 2020. ISSN 0960-0779. doi: https://doi.org/10.1016/j.chaos.2020.110057. URL `https://www.sciencedirect.com/science/article/pii/S0960077920304549`.

Cranmer, K., Brehmer, J., and Louppe, G. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, May 2020. ISSN 1091-6490. doi: 10.1073/pnas.1912789117. URL `http://dx.doi.org/10.1073/pnas.1912789117`.

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`.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Fischer, S. F., Feurer, M., and Bischl, B. OpenML-CTR23 – a curated tabular regression benchmarking suite. In *AutoML Conference 2023 (Workshop)*, 2023. URL `https://openreview.net/forum?id=HebAOoMm94`.

Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *Advances in neural information processing systems*, 31, 2018.

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.

Geffner, T., Papamakarios, G., and Mnih, A. Compositional score modeling for simulation-based inference. 2023.

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

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. *Bayesian Data Analysis, Third Edition*. CRC Press, November 2013. ISBN 9781439840955. URL `https://play.google.com/store/books/details?id=ZXL6AQAAQBAJ`.

Glaeser, R. M., Nogales, E., and Chiu, W. *Single-particle Cryo-EM of Biological Macromolecules*. 2053-2563. IOP Publishing, 2021. ISBN 978-0-7503-3039-8. doi: 10.1088/978-0-7503-3039-8. URL `https://dx.doi.org/10.1088/978-0-7503-3039-8`.

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.

Grant, E., Finn, C., Levine, S., Darrell, T., and Griffiths, T. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930*, 2018.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=Sy2fzU9gl`.

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.

Hollmann, N., Müller, S., Eggensperger, K., and Hutter, F. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*, 2022.

Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 44(09):5149–5169, sep 2022. ISSN 1939-3539. doi: 10.1109/TPAMI.2021.3079209.

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.

Koch, G., Zemel, R., Salakhutdinov, R., et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

Krueger, D., Huang, C.-W., Islam, R., Turner, R., Lacoste, A., and Courville, A. Bayesian hypernetworks. *arXiv preprint arxiv:1710.04759*, 2017.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pp. 3744–3753. PMLR, 2019.

Lin, W., Schmidt, M., and Khan, M. E. Handling the positive-definite constraint in the bayesian learning rule. In *International conference on machine learning*, pp. 6116–6126. PMLR, 2020.

Liu, S., Sun, X., Ramadge, P. J., and Adams, R. P. Task-agnostic amortized inference of gaussian process hyperparameters. *Advances in Neural Information Processing Systems*, 33:21440–21452, 2020.

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.

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

Müller, S., Hollmann, N., Arango, S. P., Grabocka, J., and Hutter, F. Transformers can do bayesian inference. *arXiv preprint arXiv:2112.10510*, 2021.

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.

Simpson, F., Davies, I., Lalchand, V., Vullo, A., Durrande, N., and Rasmussen, C. E. Kernel identification through transformers. *Advances in Neural Information Processing Systems*, 34:10483–10495, 2021.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

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

Sun, Z., Ozay, M., and Okatani, T. Hypernetworks with statistical filtering for defending adversarial examples. *arXiv preprint arxiv:1711.01791*, 2017.

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., and Hospedales, T. M. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1199–1208, 2018.

Tang, L., Zhou, Y., Wang, L., Purkayastha, S., Zhang, L., He, J., Wang, F., and Song, P. X.-K. A review of multi-compartment infectious disease models. *International Statistical Review*, 88(2):462–513, 2020. doi: https://doi.org/10.1111/insr.12402. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/insr.12402.

Vargas, F., Grathwohl, W., and Doucet, A. Denoising diffusion samplers. *arXiv preprint arXiv:2302.13834*, 2023.

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.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

Von Oswald, J., Niklasson, E., Randazzo, E., Sacramento, J., Mordvintsev, A., Zhmoginov, A., and Vladymyrov, M. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pp. 35151–35174. PMLR, 2023.

von Oswald, J., Niklasson, E., Schlegel, M., Kobayashi, S., Zucchet, N., Scherrer, N., Miller, N., Sandler, M., Vladymyrov, M., Pascanu, R., et al. Uncovering mesa-optimization algorithms in transformers. *arXiv preprint arXiv:2309.05858*, 2023.

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.

Zhang, Q. and Chen, Y. Path integral sampler: a stochastic control approach for sampling. *arXiv preprint arXiv:2111.15141*, 2021.

Štrumbelj, E., Bouchard-Côté, A., Corander, J., Gelman, A., Rue, H., Murray, L., Pesonen, H., Plummer, M., and Vehtari, A. Past, present, and future of software for bayesian inference, 2023. URL http://hdl.handle.net/10754/694575.

Appendix

A Related Work

In this section, we draw parallels of our work to various approaches that have been proposed to tackle the problem of either providing a good initialization for different tasks, performing implicit optimization to model predictive distributions for new tasks, or estimating the posterior through a different objective.

A.1 Variational Autoencoders

VAEs (Kingma & Welling, 2013; Rezende et al., 2014; Rezende & Mohamed, 2015; Kingma et al., 2019) are latent variable models which model observations $\boldsymbol{x}$ conditioned on latent variables $\boldsymbol{z}$ through the joint distribution $p_\theta(\boldsymbol{x}, \boldsymbol{z}) = p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$ where $p(\boldsymbol{z})$ is generally chosen as $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Training the model is done through VI where $q_\varphi(\boldsymbol{z})$ is obtained by explicit amortization over the data point, that is, $q_\varphi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}_\varphi(\boldsymbol{x}), \boldsymbol{\Sigma}_\varphi(\boldsymbol{x}))$. 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

$$\arg\max_{\theta,\varphi} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\mathbb{E}_{\boldsymbol{z}\sim q(\cdot|\boldsymbol{x})}\left[\log\frac{p_\theta(\boldsymbol{x}, \boldsymbol{z})}{q_\varphi(\boldsymbol{z}|\boldsymbol{x})}\right] \tag{16}$$

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

A.2 Hypernetworks

Hypernetworks are neural networks that generate weights for another neural network, used in tasks such as uncertainty quantification, zero-shot learning, etc. We refer for a comprehensive overview to Chauhan et al. (2023). Based on experiments on predicting the weights of a compact MLP (section 4), our work shows similarities with studies in this area but also has significant differences. Regarding uncertainty quantification, hypernetworks are instrumental in creating an ensemble of models by generating multiple weight vectors for the primary network. Each model within this ensemble possesses distinct parameter configurations, enabling robust estimation of uncertainty in model predictions. This feature is precious in safety-critical domains like healthcare, where confidence in predictions is essential. Multiple weight sets can be generated through techniques like dropout within hypernetworks or sampling from a noise distribution. The latter (Krueger et al., 2017) is based on a Bayesian framework where weights can be sampled using invertible network architecture, such as normalizing flows. However, while we amortize posterior inference, the weights sampled from the hypernetwork are not conditioned on information from the currently observed input data during inference time but indirectly solely on the dataset available during training, and retraining would need to be done given a new dataset. Departing from the Bayesian framework, Sun et al. (2017) have shown data-specific discriminative weight prediction, which aligns well with their specific objective of defending a convolutional neural network against adversarial attacks. Combining the ability to sample a new set of weights dataset-specifically but also handling dataset exchangeability, even in the more realistic case of missing information, our work has a distinctly different focus but also can be seen as an extension to hypernetwork research.

A.3 In-Context Learning

Amortized inference has close links to in-context learning (ICL), which has been gaining popularity, especially in natural language modeling. Various works show how in-context learning can be seen as performing implicit optimization based on the context examples, with some constructions showing exact equivalence with gradient descent in linear regression (Von Oswald et al., 2023; von Oswald et al., 2023). Other works have shown how such systems can be seen as implicitly modeling the Bayesian posterior predictive distribution (Müller et al., 2021). In a similar vein, there have been additional works aimed at directly modeling the posterior predictive distribution by providing the training data as “context" to a Transformer model and training it based on the maximum log-likelihood principle (Hollmann et al., 2022). While such approaches have been seeing tremendous success, they cannot be directly applied to cases where we care about and want to analyze the solution space as the solution space is only modeled implicitly, and thus, recovering it is not possible. For example, if our goal is to learn a linear regression model, an ICL model could end up learning a nonlinear model and would provide no information about the actual parameters used for prediction. As opposed to this, we obtain parameters explicitly. We thus can answer questions like the relevance of a particular feature (which corresponds to its weight in the output, and we know the weight vector explicitly). Even further, many systems grounded in physics and economics only admit a constrained solution space; for example, the movement of a human arm lies on a particular manifold, or the configuration of molecules and proteins cannot be arbitrary. Thus, performing predictions through an implicit solution space, which may violate several constraints, is not ideal. Furthermore, explicitly modeling the solution space and encoding the constraints present can be done through the prior and the parametric distribution used for modeling.

A.4 Meta Learning

Meta-learning (Hospedales et al., 2022) aims to equip models with the ability to quickly learn from different tasks or data sets to generalize to new tasks in resource-constrained domains. This attribute is precious in practical scenarios where obtaining large amounts of task-specific data is impractical or costly. A simple way of obtaining this is through nonparametric or similarity-based models like k-Nearest Neighbours, where no training is involved. Thus, new tasks can be solved quickly based on a few examples by computing a similarity metric with these examples (Koch et al., 2015; Vinyals et al., 2016; Sung et al., 2018). Another way of achieving this is through optimization-based setups, which use a nested optimization procedure. An inner step learns individual tasks from a shared initialization, whereas the outer loop computes the gradient of the whole inner process and moves the initialization in a way that allows for better generalization. Here, by relying on only a few iterations in the inner loop, the outer loop has the incentive to move the initialization to a point from which solutions to multiple tasks are reachable (Finn et al., 2017). Given the similarities between meta-learning and hierarchical Bayesian inference (Grant et al., 2018), our approach can be considered as a kind of meta-learning framework; however, the line between meta-learning and Bayesian posterior inference is quite blurry as any amortized approach for the latter can be seen as a case of the former.

A.5 Neural Processes

A notable approach in meta-learning related to our research is neural processes (NP), which excel in learning scenarios with few examples. NPs (Garnelo et al., 2018a;b; Kim et al., 2019; Pakman et al., 2020; Gordon et al., 2019) 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, NPs 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 an explicit posterior over the parameters, but they are suitable for tasks where only predictive posteriors are essential, such as those in supervised learning. NPs, in their most basic form, accomplish this by training for the objective:

$$\arg\max_{\boldsymbol{\theta},\varphi} \mathbb{E}_{\mathcal{D}\sim\chi}\mathbb{E}_{\boldsymbol{z}\sim q_\varphi(\cdot|\mathcal{D})}\left[\log \frac{p_{\boldsymbol{\theta}}(\mathcal{D},\boldsymbol{z})}{q_\varphi(\boldsymbol{z}|\mathcal{D})}\right] \tag{17}$$

where $\boldsymbol{z} \in \mathbb{R}^p$ 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 $\mathcal{D} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^n$, where all probabilities in Equation 17 are conditioned on the input $\boldsymbol{x}$'s, and only the predictive over $\boldsymbol{y}$'s is modeled, and $p_{\boldsymbol{\theta}}$ is modeled as a Neural Network.

These approaches can be seen as quite related to ICL, where the exchangeable architecture backbone is switched from DeepSets to Transformers. Similar to ICL, they do not provide control over the solution space as they aim to model either the posterior predictive or an arbitrary latent space. While this leads to good predictive performance on various tasks, they cannot be freely applied to problems that pose certain constraints on the underlying probabilistic model. In such cases, estimating the actual parameters is important to enforce constraints in the parameter space as well as for interpretability, which we already discussed in the ICL section.

A.6 Simulation-Based Inference

In the case of simulation-based inference (Cranmer et al., 2020), when the likelihood $p(\boldsymbol{x}|\boldsymbol{\theta})$ is intractable, 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.

$$\arg\min_{\varphi} \mathbb{KL}[p(\boldsymbol{\theta}|\mathcal{D})||q_{\varphi}(\boldsymbol{\theta}|\mathcal{D})] = \arg\min_{\varphi=\{\nu,\psi\}} \mathbb{E}_{(\boldsymbol{\theta},\mathcal{D})\sim p(\boldsymbol{\theta},\mathcal{D})}\left[-\log p_{\boldsymbol{z}}(f_{\nu}(\boldsymbol{\theta};h_{\psi}(\mathcal{D}))) - \log|\det J_{f_{\nu}}|\right] \tag{18}$$

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, the method requires access to paired observations $(\boldsymbol{x}, \boldsymbol{\theta})$ for training, which is sometimes unavailable. This approach is equivalent to the *Forward KL* setup in our experiments when trained with DeepSets and Normalizing Flows. Current research has also leveraged score-based generative models for SBI which can condition on a dataset by learning a score model conditional only on single observations (Geffner et al., 2023).

A.7 Amortization in Gaussian Processes

Gaussian Processes (GPs) define a class of probabilistic models that do enjoy tractable likelihood. However, inference in such systems is slow and sensitive to the choice of kernel function that defines the covariance matrix. Similar to meta learning and neural processes, current research also focuses on estimating the kernel function in GPs by leveraging permutation invariant architectures like transformers (Liu et al., 2020; Simpson et al., 2021; Bitzer et al., 2023). Additionally, often these approaches amortize based on point estimates and are leveraged when considering GPs for regression problems, and it is not straightforward to extend them to classification or unsupervised learning. In contrast, our approach is more general and can work for all problems that define a differentiable likelihood function. Additionally, our approach also approximates the Bayesian posterior distribution over the parameters of interest, as opposed to point estimates.

A.8 Mode Collapse in Variational Inference

Reverse KL based methods have been widely known to suffer from mode collapse due to the nature of the optimization objective (Bishop & Nasrabadi, 2006), which implies that even if the approximate distribution possesses the ability to represent multiple modes, optimization is often sub-optimal and the distribution ends up covering only a small handful of them. Improving normalizing flow based methods with repulsive terms or through the lens of natural gradient optimization procedure for a mixture approximate distribution (Arenz et al., 2022; Lin et al., 2020) is an important topic of research, and we believe it would be quite an important future work to experimentally validate if they help in learning multi-modality in amortized posterior inference problems that are studied in this work.

B Architectures respecting Exchangeability

In this section, we highlight how DeepSets and Transformer models satisfy the dataset exchangeability criteria, which is essential in modeling the posterior distribution over the parameters of any probabilistic model relying on *iid* data.

B.1 DeepSets

DeepSets (Zaheer et al., 2017) operate on arbitrary sets $\mathcal{X} = \{x_1, ...x_N\} \subset \mathbb{R}^d$ of fixed dimensionality d by first mapping each individual element $\boldsymbol{x}_i \in \mathcal{X}$ to some high-dimensional space using a nonlinear transform, which is parameterized as a multi-layered neural network with parameters φ_1

$$\boldsymbol{z}_i = f_{\varphi_1}(\boldsymbol{x}_i) \tag{19}$$

After having obtained this high-dimensional embedding of each element of the set, it applies an aggregation function $a(\cdot)$, which is a permutation invariant function that maps a set of elements $\mathcal{Z} = \{\boldsymbol{z}_1, ..., \boldsymbol{z}_N\} \in \mathbb{R}^z$ to an element $\boldsymbol{h} \in \mathbb{R}^z$,

$$\boldsymbol{h} = a(\mathcal{Z}) \tag{20}$$

Thus, the outcome does not change under permutations of $\mathcal{Z}$. Finally, another nonlinear transform, parameterized by a multi-layered neural network with parameters φ_2, is applied to the outcome $\boldsymbol{h}$ to provide the final output.

$$\boldsymbol{o} = g_{\varphi_2}(\boldsymbol{h}) \tag{21}$$

For our experiments, we then use the vector $\boldsymbol{o}$ to predict the parameters of a parametric family of distributions (e.g., Gaussian or Flows) using an additional nonlinear neural network. As an example, for the Gaussian case, we consider the distribution $\mathcal{N}(\cdot|\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where

$$\boldsymbol{\mu} := \boldsymbol{\mu}_{\varphi_3}(\boldsymbol{o}) \quad \text{and} \quad \boldsymbol{\Sigma} := \boldsymbol{\Sigma}_{\varphi_4}(\boldsymbol{o}) \tag{22}$$

which makes $\boldsymbol{\mu}$ implicitly a function of the original input set $\mathcal{X}$. To understand why the posterior distribution modeled in this fashion does not change when the inputs are permuted, let us assume that Π is a permutation over the elements of $\mathcal{X}$. If we look at one of the parameters of the posterior distribution, e.g., $\boldsymbol{\mu}$, we can see that

$$\begin{aligned}
\boldsymbol{\mu}(\Pi\mathcal{X}) &= \boldsymbol{\mu}_{\varphi_3}\left(g_{\varphi_2}\left(a\left(\{f_{\varphi_1}(\boldsymbol{x}_{\Pi(i)})\}_{i=1}^N\right)\right)\right) && (23)\\
&= \boldsymbol{\mu}_{\varphi_3}\left(g_{\varphi_2}\left(a\left(\{f_{\varphi_1}(\boldsymbol{x}_i)\}_{i=1}^N\right)\right)\right) && (24)\\
&= \boldsymbol{\mu}(\mathcal{X}) && (25)
\end{aligned}$$

which simply follows from the fact that $a(\cdot)$ is a permutation invariant operation, e.g., sum or mean. We can also provide similar reasoning for the other parameters (e.g., $\boldsymbol{\Sigma}$). This shows that DeepSets can be used to model the posterior distribution over parameters of interest as it respects the exchangeability criteria (*iid* observations) assumptions in the data through its permutation invariant structure.

B.2 Transformers

Similarly, we can look at Transformers (Vaswani et al., 2017) as candidates for respecting the exchangeability conditions in the data. In particular, we consider transformer systems without positional encodings and consider an additional [CLS] token, denoted by $\boldsymbol{c} \in \mathbb{R}^d$, to drive the prediction. If we look at the application of a layer of transformer model, it can be broken down into two components.

Multi-Head Attention. Given a query vector obtained from $\boldsymbol{c}$ and keys and values coming from our input set $\mathcal{X} \subset \mathbb{R}^d$, we can model the update of the context $\boldsymbol{c}$ as

$$\hat{\boldsymbol{c}}(\mathcal{X}) = \text{Softmax}\left(\boldsymbol{c}^T \boldsymbol{W}_Q \boldsymbol{W}_K^T \boldsymbol{X}^T\right) \boldsymbol{X} \boldsymbol{W}_V \tag{26}$$

where $\boldsymbol{W}_Q \in \mathbb{R}^{d\times k}, \boldsymbol{W}_K \in \mathbb{R}^{d\times k}, \boldsymbol{W}_V \in \mathbb{R}^{d\times k}$ and $\boldsymbol{X} \in \mathbb{R}^{N\times d}$ denotes a certain ordering of the elements in $\mathcal{X}$. Further, $\hat{\boldsymbol{c}}$ is the updated vector after attention, and Softmax is over the rows of $\boldsymbol{X}$. Here, we see that if we were to apply a permutation to the elements in $\boldsymbol{X}$, the outcome would remain the same. In particular

$$\begin{aligned}
\hat{\boldsymbol{c}}(\Pi\boldsymbol{X}) &= \text{Softmax}\left(\boldsymbol{c}^T \boldsymbol{W}_Q \boldsymbol{W}_K^T \boldsymbol{X}^T \Pi^T\right) \Pi \boldsymbol{X} \boldsymbol{W}_V && (27)\\
&= \text{Softmax}\left(\boldsymbol{c}^T \boldsymbol{W}_Q \boldsymbol{W}_K^T \boldsymbol{X}^T\right) \Pi^T \Pi \boldsymbol{X} \boldsymbol{W}_V && (28)\\
&= \text{Softmax}\left(\boldsymbol{c}^T \boldsymbol{W}_Q \boldsymbol{W}_K^T \boldsymbol{X}^T\right) \boldsymbol{X} \boldsymbol{W}_V && (29)\\
&= \hat{\boldsymbol{c}}(\boldsymbol{X}) && (30)
\end{aligned}$$

which follows because Softmax is an equivariant function, i.e., applying Softmax on a permutation of columns is equivalent to applying Softmax first and then permuting the columns correspondingly. Thus, we see that the update to the [CLS] token $\boldsymbol{c}$ is permutation invariant. This output is then used independently as input to a multi-layered neural network with residual connections, and the entire process is repeated multiple times without weight sharing to simulate multiple layers. Since all the individual parts are permutation invariant w.r.t permutations on $\mathcal{X}$, the entire setup ends up being permutation invariant. Obtaining the parameters of a parametric family of distribution for posterior estimation then follows the same recipe as DeepSets, with $\boldsymbol{o}$ replaced by $\boldsymbol{c}$.

C Probabilistic Models

This section details the various candidate probabilistic models used in our experiments for amortized computation of Bayesian posteriors over the parameters. Here, we explain the parameters associated with the probabilistic model over which we want to estimate the posterior and the likelihood and prior that we use for experimentation.

Mean of Gaussian (GM): As a proof of concept, we consider the simple setup of estimating the posterior distribution over the mean of a Gaussian distribution $p(\boldsymbol{\mu}|\mathcal{D})$ given some observed data. In this case, prior and likelihood defining the probabilistic model $p(\boldsymbol{x}, \boldsymbol{\theta})$ (with $\boldsymbol{\theta}$ being the mean $\boldsymbol{\mu}$) are given by:

$$p(\boldsymbol{\mu}) = \mathcal{N}\left(\boldsymbol{\mu}|\mathbf{0}, \mathbf{I}\right) \tag{31}$$

$$p(\boldsymbol{x}|\boldsymbol{\mu}) = \mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}\right) \tag{32}$$

and $\boldsymbol{\Sigma}$ is known beforehand and defined as a unit variance matrix.

Linear Regression (LR): We then look at the problem of estimating the posterior over the weight vector for Bayesian linear regression given a dataset $p(\boldsymbol{w}, b|\mathcal{D})$, where the underlying model $p(\mathcal{D}, \boldsymbol{\theta})$ is given by:

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\mathbf{0}, \mathbf{I}) \tag{33}$$

$$p(b) = \mathcal{N}(b|0, 1) \tag{34}$$

$$p(y|\boldsymbol{x}, \boldsymbol{w}, b) = \mathcal{N}\left(y|\boldsymbol{w}^T\boldsymbol{x} + b, \sigma^2\right) \;, \tag{35}$$

and with $\sigma^2 = 0.25$ known beforehand. Inputs $\boldsymbol{x}$ are generated from $p(\boldsymbol{x}) = \mathcal{N}(\mathbf{0}, I)$.

Linear Classification (LC): 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:

$$p(\boldsymbol{W}) = \mathcal{N}\left(\boldsymbol{W}|\mathbf{0}, \mathbf{I}\right) \tag{36}$$

$$p(y|\boldsymbol{x}, \boldsymbol{W}) = \text{Categorical}\left(y \,\middle|\, \frac{1}{\tau}\,\boldsymbol{W}\boldsymbol{x}\right) \;, \tag{37}$$

where τ is the known temperature term which is kept as 0.1 to ensure peaky distributions, and $\boldsymbol{x}$ is being generated from $p(\boldsymbol{x}) = \mathcal{N}(\mathbf{0}, I)$.

Nonlinear Regression (NLR): 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 Bayesian Neural Network (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 such that the estimation problem is to approximate $p(\boldsymbol{\theta}|\mathcal{D})$. Then, for regression, we specify the probabilistic model using:

$$p(\boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{\theta}|\mathbf{0}, \mathbf{I}\right) \tag{38}$$

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}\left(y|f_{\boldsymbol{\theta}}(\boldsymbol{x}), \sigma^2\right) \;, \tag{39}$$

where $\sigma^2 = 0.25$ is a known quantity and $\boldsymbol{x}$ being generated from $p(\boldsymbol{x}) = \mathcal{N}(\mathbf{0}, I)$.

Nonlinear Classification (NLC): Like in Nonlinear Regression, we consider BNNs with fixed hyperparameters for classification problems with the same estimation task of approximating $p(\boldsymbol{\theta}|\mathcal{D})$. In this formulation, we consider the probabilistic model as:

$$p(\boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{\theta}|\mathbf{0}, \mathbf{I}\right) \tag{40}$$

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \text{Categorical}\left(y \,\middle|\, \frac{1}{\tau}\, f_{\boldsymbol{\theta}}(\boldsymbol{x})\right) \tag{41}$$

where τ is the known temperature term which is kept as 0.1 to ensure peaky distributions, and $\boldsymbol{x}$ is being generated from $p(\boldsymbol{x}) = \mathcal{N}(\mathbf{0}, I)$.

Gaussian Mixture Model (GMM): While we have mostly looked at predictive problems, where the task is to model some predictive variable y conditioned on some input $\boldsymbol{x}$, 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:

$$p(\boldsymbol{\mu}_k) = \mathcal{N}\left(\boldsymbol{\mu}_k|\mathbf{0}, \mathbf{I}\right) \tag{42}$$

$$p(\boldsymbol{x}|\boldsymbol{\mu}_{1:K}) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)\,. \tag{43}$$

We assume $\boldsymbol{\Sigma}_k$ and π_k to be known and set $\boldsymbol{\Sigma}_k$ to be an identity matrix and the mixing coefficients to be equal, $\pi_k = 1/K$, for all clusters k in our experiments.

D Metrics

In this section, we provide details about the metrics considered for the different tasks. We generally look at two main metrics for benchmarking performance: L_2 loss and Accuracy. For estimating the mean of a Gaussian distribution, the L_2 loss is defined as

$$GM_{L_2} = \mathbb{E}_{\mathcal{D}\sim\chi}\mathbb{E}_{\boldsymbol{\mu}\sim q_\varphi(\cdot|\mathcal{D})}\left[\sum_{i=1}^{N_\mathcal{D}} (\boldsymbol{x}_i - \boldsymbol{\mu})^2\right] \tag{44}$$

where $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^{N_\mathcal{D}}$. Intuitively, this captures the quality of the estimation of the mean parameter by measuring how far the observations are from it. Lower value implies better estimation of the mean parameter. Similarly, for estimating the means of a Gaussian Mixture Model, we rely on a similar metric but we also find the cluster closest to the observation, which can be defined as

$$GMM_{L_2} = \mathbb{E}_{\mathcal{D}\sim\chi}\mathbb{E}_{\boldsymbol{\mu}_k\sim q_\varphi(\cdot|\mathcal{D})}\left[\sum_{i=1}^{N_\mathcal{D}} (\boldsymbol{x}_i - \boldsymbol{\mu}_{\text{Match}(\boldsymbol{x}_i, \{\boldsymbol{\mu}_1, \ldots \boldsymbol{\mu}_K\})})^2\right] \tag{45}$$

$$\text{Match}(\boldsymbol{x}, \{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_K\} = \arg\min_k (\boldsymbol{x} - \boldsymbol{\mu}_k)^2 \tag{46}$$

which intuitively captures the distance of observations from the cluster closest to them. Next, we define the metric for evaluating (non-)linear regression models as

$$(N-)LR_{L_2} = \mathbb{E}_{\mathcal{D}\sim\chi}\mathbb{E}_{\boldsymbol{\theta}\sim q_\varphi(\cdot|\mathcal{D})}\left[\sum_{i=1}^{N_\mathcal{D}} (y_i - \text{Mode}\left[p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta})\right])^2\right] \tag{47}$$

Finally, for the (non-)linear classification setups, we define the accuracy metric as

$$(N-)LC_{Accuracy} = \mathbb{E}_{\mathcal{D}\sim\chi}\mathbb{E}_{\boldsymbol{\theta}\sim q_\varphi(\cdot|\mathcal{D})}\left[\frac{100}{N_\mathcal{D}} \times \sum_{i=1}^{N_\mathcal{D}} \delta(y_i, \text{Mode}\left[p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta})\right])\right] \tag{48}$$

where $\delta(a, b) = 1$ if and only if $a = b$. Thus this metric captures the accuracy of the posterior predictive distribution. Another metric that we use to test the quality of the posterior is the symmetric KL divergence, defined as

$$\text{Symmetric } \mathbb{KL}(p(\boldsymbol{\theta}||\mathcal{D}), q_\varphi(\boldsymbol{\theta}|\mathcal{D})) = \frac{1}{2}\mathbb{KL}(p(\boldsymbol{\theta}||\mathcal{D})||q_\varphi(\boldsymbol{\theta}|\mathcal{D})) + \frac{1}{2}\mathbb{KL}(q_\varphi(\boldsymbol{\theta}|\mathcal{D})||p(\boldsymbol{\theta}||\mathcal{D})) \tag{49}$$

Additionally, another metric in the predictive space that we use is the expected negative conditional log likelihood (CNLL), which is defined as

$$CNLL = -\mathbb{E}_{q_\varphi(\cdot|\mathcal{D})}\left[\log p(\mathcal{D}|\boldsymbol{\theta})\right] \tag{50}$$

E Architecture Details

In this section, we outline the two candidate architectures that we consider for the backbone of our amortized variational inference model. We discuss the specifics of the architectures and the hyperparameters used for our experiments.

E.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.

E.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 comparable to those in the Transformer model.

E.3 Normalizing Flows

Assuming a Gaussian posterior distribution as the approximate often leads to poor results as the true posterior distribution can be far from the 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 $g_\nu : \boldsymbol{z} \mapsto \boldsymbol{\theta}$ be a diffeomorphism parameterized by a conditional invertible neural network (cINN) with network parameters ν such that $\boldsymbol{\theta} = g_\nu(\boldsymbol{z}; h_\psi(\mathcal{D}))$. With the change-of-variables formula it follows that $p(\boldsymbol{\theta}) = p(\boldsymbol{z}) \left|\det \frac{\partial}{\partial \boldsymbol{z}} g_\nu(\boldsymbol{z}; h_\psi(\mathcal{D}))\right|^{-1} = p(\boldsymbol{z})|\det J_\nu(\boldsymbol{z}; h_\psi(\mathcal{D}))|^{-1}$, where J_ν is the Jacobian matrix of g_ν. Further, integration by substitution gives us $d\boldsymbol{\theta} = |\det J_\nu(\boldsymbol{z}; h_\psi(\mathcal{D})|d\boldsymbol{z}$ to rewrite the objective from eq. 12 as:

$$\arg\min_{\varphi} \mathbb{KL}[q_\varphi(\boldsymbol{\theta}|\mathcal{D})||p(\boldsymbol{\theta}|\mathcal{D})] \tag{51}$$

$$= \arg\min_{\varphi} \mathbb{E}_{\mathcal{D}\sim\chi}\mathbb{E}_{\boldsymbol{\theta}\sim q_\varphi(\boldsymbol{\theta}|\mathcal{D})}\left[\log q_\varphi(\boldsymbol{\theta}|\mathcal{D}) - \log p(\boldsymbol{\theta},\mathcal{D})\right] \tag{52}$$

$$= \arg\min_{\varphi=\{\psi,\nu\}} \mathbb{E}_{\mathcal{D}\sim\chi}\mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\left[\log \frac{q_\nu(\boldsymbol{z}|h_\psi(\mathcal{D}))}{|\det J_\nu(\boldsymbol{z};h_\psi(\mathcal{D}))|} - \log p(g_\nu(\boldsymbol{z};h_\psi(\mathcal{D})),\mathcal{D})\right] \tag{53}$$

As shown in BayesFlow (Radev et al., 2020), the normalizing flow g_ν and the summary network h_ψ 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, 6 coupling blocks define the normalizing flow network, each with a 1 hidden-layered non-linear feed-forward subnetwork with ReLU non-linearity and 128 hidden dimensions.

F Experimental Details

Unless specified, we obtain a stream of datasets for all our experiments by simply sampling from the assumed probabilistic model, where the number of observations n is sampled uniformly 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.

To evaluate both our proposed approach and the baselines, we compute an average of the predictive performances across 25 different posterior samples for each of the 100 fixed test datasets for all our experiments. That means for our proposed approach, we sample 25 different parameter vectors from the approximate posterior that we obtain. For MCMC, we rely on 25 MCMC samples, and for optimization, we train 25 different parameter vectors where the randomness comes from initialization. For the optimization baseline, we perform a quick hyperparameter search over the space $\{0.01, 003, 0.001, 0.0003, 0.0001, 0.00003\}$ to pick the best learning rate that works for all of the test datasets and then use it to train for 1000 iterations using the Adam optimizer (Kingma & Ba, 2014). For the MCMC baseline, we use the open-sourced implementation of Langevin-based MCMC sampling[2] where we leave a chunk of the starting samples as burn-in and then start accepting samples after a regular interval (to not make them correlated). The details about the burn-in time and the regular interval for acceptance are provided in the corresponding experiments' sections below.

For our proposed approach of amortized inference, we do not consider explicit hyperparameter optimization and simply use a learning rate of 1e-4 with the Adam optimizer. For all experiments, we used linear scaling of the KL term in the training objectives as described in (Higgins et al., 2017), which we refer to as warmup. Furthermore, training details for each experiment can be found below.

F.1 Fixed-Dim

In this section, we provide the experimental details relevant to reproducing the results of Section 4. All the models are trained with streaming data from the underlying probabilistic model, such that every iteration of training sees a new set of datasets. Training is done with a batch size of 128, representing the number of datasets seen during one optimization step. Evaluations are done with 25 samples and we ensure that the test datasets used for each probabilistic model are the same across all the compared methods, i.e., baselines, forward KL, and reverse KL. We train the amortized inference model and the forward KL baselines for the following different probabilistic models:

Mean of Gaussian (GM): We train the amortization models over 20,000 iterations for both the 2-dimensional as well as the 100-dimensional setup. We use a linear warmup with 5000 iterations over which

---

[2] https://github.com/alisiahkoohi/Langevin-dynamics

the weight of the KL term in our proposed approach scales linearly from 0 to 1. We use an identity covariance matrix for the data-generating process, but it can be easily extended to the case of correlated or diagonal covariance-based Gaussian distributions.

Gaussian Mixture Model (GMM): We train the mixture model setup for $200,000$ iterations with $50,000$ iterations of warmup. We mainly experiment with 2-dimensional and 5-dimensional mixture models, with 2 and 5 mixture components for each setup. While we do use an identity covariance matrix for the data-generating process, again, it can be easily extended to other cases.

Linear Regression (LR): The amortization models for this setup are trained for $50,000$ iterations with $12,500$ iterations of warmup. The feature dimensions considered for this task are 1 and 100 dimensions, and the predictive variance σ^2 is assumed to be known and set as 0.25.

Nonlinear Regression (NLR): We train the setup for $100,000$ iterations with $25,000$ iterations consisting of warmup. The feature dimensionalities considered are 1-dimensional and 25-dimensional, and training is done with a known predictive variance similar to the LR setup. For the probabilistic model, we consider both a 1-layered and a 2-layered multi-layer perceptron (MLP) network with 32 hidden units in each, and either a RELU or TANH activation function.

Linear Classification (LC): We experiment with 2-dimensional and 100-dimensional setups with training done for $50,000$ iterations, out of which $12,500$ are used for warmup. Further, we train for both binary classification as well as a 5-class classification setup.

Nonlinear Classification (NLC): We experiment with 2-dimensional and 25-dimensional setups with training done for $100,000$ iterations, out of which $2,5000$ are used for warmup. Further, we train for both binary classification as well as a 5-class classification setup. For the probabilistic model, we consider both a 1-layered and a 2-layered multi-layer perceptron (MLP) network with 32 hidden units in each, and either a RELU or TANH activation function.

F.2 Variable-Dim

In this section, we provide the experimental details relevant to reproducing the results of Section 4. All the models are trained with streaming data from the underlying probabilistic model, such that every iteration of training sees a new set of datasets. Training is done with a batch size of 128, representing the number of datasets seen during one optimization step. Further, we ensure that the datasets sampled resemble a uniform distribution over the feature dimensions, ranging from 1-dimensional to the maximal dimensional setup. Evaluations are done with 25 samples and we ensure that the test datasets used for each probabilistic model are the same across all the compared methods, i.e., baselines, forward KL, and reverse KL. We train the amortized inference model and the forward KL baselines for the following different probabilistic models:

Mean of Gaussian (GM): We train the amortization models over $50,000$ iterations using a linear warmup with $12,5000$ iterations over which the weight of the KL term in our proposed approach scales linearly from 0 to 1. We use an identity covariance matrix for the data-generating process, but it can be easily extended to the case of correlated or diagonal covariance-based Gaussian distributions. In this setup, we consider a maximum of 100 feature dimensions.

Gaussian Mixture Model (GMM): We train the mixture model setup for $500,000$ iterations with $125,000$ iterations of warmup. We set the maximal feature dimensions as 5 and experiment with 2 and 5 mixture components. While we do use an identity covariance matrix for the data-generating process, again, it can be easily extended to other cases.

Linear Regression (LR): The amortization models for this setup are trained for $100,000$ iterations with $25,000$ iterations of warmup. The maximal feature dimension considered for this task is 100-dimensional, and the predictive variance σ^2 is assumed to be known and set as 0.25.

Nonlinear Regression (NLR): We train the setup for $250,000$ iterations with $62,500$ iterations consisting of warmup. The maximal feature dimension considered is 100-dimensional, and training is done with a known predictive variance similar to the LR setup. For the probabilistic model, we consider both a 1-layered and a

2-layered multi-layer perceptron (MLP) network with 32 hidden units in each, and either a RELU or TANH activation function.

Linear Classification (LC): We experiment with a maximal 100-dimensional setup with training done for $100,000$ iterations, out of which $25,000$ are used for warmup. Further, we train for both binary classification as well as a 5-class classification setup.

Nonlinear Classification (NLC): We experiment with a maximal 100-dimensional setup with training done for $250,000$ iterations, out of which $62,500$ are used for warmup. Further, we train for both binary classification as well as a 5-class classification setup. For the probabilistic model, we consider both a 1-layered and a 2-layered multi-layer perceptron (MLP) network with 32 hidden units in each, and either a RELU or TANH activation function.

F.3 Model Misspecification

In this section, we provide the experimental details relevant to reproducing the results of Section 4. All models during this experiment are trained with streaming data from the currently used dataset-generating function χ, such that every iteration of training sees a new batch of datasets. Training is done with a batch size of 128, representing the number of datasets seen during one optimization step. Evaluation for all models is done with 10 samples from each dataset-generator used in the respective experimental subsection and we ensure that the test datasets are the same across all compared methods, i.e., baselines, forward KL, and reverse KL.

Linear Regression Model: The linear regression amortization models are trained following the training setting for linear regression fixed dimensionality, that is, $50,000$ training iterations with $12,500$ iterations of warmup. The feature dimension considered for this task is 1-dimension. The model is trained separately on datasets from three different generators χ: linear regression, nonlinear regression, and Gaussian processes, and evaluated after training on test datasets from all of them. For training with datasets from the linear regression probabilistic model, the predictive variance σ^2 is assumed to be known and set as 0.25. The same variance is used for generating datasets from the nonlinear regression dataset generator with 1 layer, 32 hidden units, and TANH activation function. Lastly, datasets from the Gaussian process-based generator are sampled similarly, using the GPytorch library Gardner et al. (2018), where datasets are sampled of varying cardinality, ranging from 64 to 128. We use a zero-mean Gaussian Process (GP) with a unit lengthscale radial-basis function (RBF) kernel serving as the covariance matrix. Further, we use a very small noise of $\sigma^2 = 1\mathrm{e}^{-6}$ in the likelihood term of the GP. Forward KL training in this experiment can only be done when the amortization model and the dataset-generating function are the same: when we train on datasets from the linear regression-based χ. Table 14 provides a detailed overview of the results.

Nonlinear Regression Models: The nonlinear regression amortization models are trained following the training setting for nonlinear regression fixed dimensionality, that is, $100,000$ training iterations with $25,000$ iterations of warmup. Here, we consider two single-layer perceptions with 32 hidden units and either a RELU or TANH activation function. The feature dimensionality considered is 1 dimension. We consider the same three dataset-generating functions as in the misspecification experiment for a linear regression model above. However, the activation function used in the nonlinear regression dataset generator matches the activation function of the currently trained amortization model. In this case, forward KL training is possible in the two instances when trained on datasets from the corresponding nonlinear regression probabilistic model. A more detailed overview of the results can be found in Table 15 for the TANH and in Table 16 for the RELU activation function-based probabilistic models respectively.

F.4 Tabular Experiments

For the tabular experiments, we train the amortized inference models for (non-)linear regression (NLR/LR) as well as (non-)linear classification (NLC/LC) with $\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ as opposed to $\boldsymbol{x} \sim \mathcal{U}(-\mathbf{1}, \mathbf{1})$ in the dataset generating process χ, with the rest of the settings the same as MAXIMUM-DIM experiments. For the nonlinear setups, we only consider the RELU case as it has seen predominant success in deep learning. Further, we only consider a 1-hidden layer neural network with 32 hidden dimensions in the probabilistic model.

After having trained the amortized inference models, both for forward and reverse KL setups, we evaluate them on real-world tabular datasets. We first collect a subset of tabular datasets from the OpenML platform as outlined in Appendix G. Then, for each dataset, we perform a 5-fold cross-validation evaluation where the dataset is chunked into 5 bins, of which, at any time, 4 are used for training and one for evaluation. This procedure is repeated five times so that every chunk is used for evaluation once.

For each dataset, we normalize the observations and the targets so that they have zero mean and unit standard deviation. For the classification setups, we only normalize the inputs as the targets are categorical. For both forward KL and reverse KL amortization models, we initialize the probabilistic model from samples from the amortized model and then further finetune it via dataset-specific maximum a posteriori optimization. We repeat this setup over 25 different samples from the inference model. In contrast, for the optimization baseline, we initialize the probabilistic models' parameters from $\mathcal{N}(0, I)$, which is the prior that we consider, and then train 25 such models with maximum a posteriori objective using Adam optimizer.

While we see that the amortization models, particularly the reverse KL model, lead to much better initialization and convergence, it is important to note that the benefits vanish if we initialize using the Xavier-init initialization scheme. However, we believe that this is not a fair comparison as it means that we are considering a different prior now, while the amortized models were trained with $\mathcal{N}(0, I)$ prior. We defer the readers to the section below for additional discussion and experimental results.

G OpenML Datasets

For the tabular regression problems, we consider the suite of tasks outlined in *OpenML-CTR23 - A curated tabular regression benchmarking suite* (Fischer et al., 2023), from which we further filter out datasets that have more than 2000 examples and 100 features. We also remove datasets with missing information and NaNs. Similarly, we consider the *OpenML-CC18 Curated Classification benchmark* (Bischl et al., 2019) suite of tasks for classification and perform a similar filtering algorithm. We remove datasets with missing information and NaNs, as well as datasets with more than 2000 examples and 100 features. In addition, we also exclude datasets that are not made for binary classification. At the end of this filtering mechanism, we end up with 9 regression and 13 classification problems, and our dataset filtration pipeline is heavily inspired by Hollmann et al. (2022). We provide the datasets considered for both regression and classification below:

Regression: AIRFOIL_SELF_NOISE, CONCRETE_COMPRESSIVE_STRENGTH, ENERGY_EFFICIENCY, SOLAR_FLARE, STUDENT_PERFORMANCE_POR, QSAR_FISH_TOXICITY, RED_WINE, SOCMOB and CARS.

Classification: CREDIT-G, DIABETES, TIC-TAC-TOE, PC4, PC3, KC2, PC1, BANKNOTE-AUTHENTICATION, BLOOD-TRANSFUSION-SERVICE-CENTER, ILPD, QSAR-BIODEG, WDBC and CLIMATE-MODEL-SIMULATION-CRASHES.

H Additional Experiments

In this section, we outline the additional experiments we conducted in obtaining Bayesian posteriors for the different probabilistic models for different hyperparameters and their downstream uses. We provide a comprehensive account of the results in the relevant sections below.

H.1 Fixed-Dim

While we highlighted the results with the Gaussian mixture model and classification settings with only 2 clusters/classes, we also conducted experiments with an increased number of clusters and classes, making the problem even more challenging. Table 8 shows that both forward and reverse KL methods perform reasonably, with forward KL struggling more in challenging scenarios.

Next, we also consider harder tasks based on the Bayesian Neural Network (BNN) paradigm, where we consider nonlinear regression and classification setups with different activation functions: TANH and RELU for a 1-layered and 2-layered BNN. We provide the results of our experiments in Tables 9 and 10 respectively. The results indicate that forward KL approaches struggle a lot in such scenarios, often achieving performance comparable to random chance. On the contrary, we see that reverse KL-based amortization leads to performances often similar to dataset-specific optimization, thereby showing the superiority of our proposed method.

H.2 Variable-Dim

Our experiments on variable dimensional datasets can be evaluated for arbitrary feature cardinality, of which we show a few examples in Section 4. In this section, we provide results for additional dimensionality setups. In particular, we refer the readers to Table 11, which contains experimental results w.r.t different dimensionalities (e.g. 50D setup), as well as different number of clusters and classes, respectively, for the GMM and LC setup. Throughout, we see that amortization leads to reasonable performance, and in particular, we see forward KL-based amortization starting to struggle in high-dimensional setups.

Again, to make the setup more challenging, we consider the Bayesian Neural Network (BNN) setup where we consider nonlinear regression and classification with different activation functions: TANH and RELU for a 1-layered and 2-layered BNN, but which can now be tested for an arbitrary number of input features. Our experiments are highlighted in Tables 12 and 13, for 1- and 2-layered BNN, respectively. In such complex multi-modal and complicated setups, forward KL often performs comparable to random chance and thus does not lead to any good approximation of the true posterior distribution. On the other hand, our proposed method indeed leads to good predictive performance, often comparable to dataset-specific optimization routines.

H.3 Model Misspecification

As a representative of the results on model misspecification (Section 4), we highlighted training and evaluation of the amortization models with Transformer backbone on a subset of in-distribution and OoD data-generating functions (Table 3) to show superiority in generalization of reverse KL trained system vs. forward KL based ones on OoD data but also to highlight that training a misspecified amortization model on OoD datasets directly with our approach results in even better posterior predictive performance.

In addition to those experiments, we also conducted a broader range of experiments utilizing DeepSets as the backbone, various OoD data-generating functions for training and evaluation of the reverse KL system, and an additional nonlinear regression model with RELU activation function. For a comprehensive description of these experiments and the complete setup, please refer to Section F.3. We considered three probabilistic models, including a linear regression model and two nonlinear regression models utilizing the TANH or RELU activation function. The detailed results for each model can be found in Tables 14, 15, and 16, respectively.

In all experiments, reverse KL outperforms forward KL trained amortization models in in-distribution performance and excels in posterior prediction on OoD datasets. Although the significant difference in posterior prediction performance of forward vs. reverse KL in cases where the underlying model is nonlinear was already mentioned in previous experiments, here, reverse KL-trained models also excel in evaluations of posterior prediction for the linear regression model. Although only by a margin, in the case of approximating the posterior of the simpler linear regression model, a diagonal Gaussian-shaped posterior shows the best posterior prediction results when evaluated on OoD datasets from the nonlinear regression dataset generating function. In almost all other experiments, the posterior prediction performance could be enhanced when we used the normalizing flow based posterior. A definitive conclusion cannot be drawn regarding the superiority of one backbone over the other, i.e. between DeepSets or Transformer. However, amortization models with DeepSets as the backbone tend towards better generalization regarding OoD datasets.

H.4 Tabular Experiments

As a case of extreme OoD generalization, we test our amortized models trained to handle variable feature dimensions on the suite of regression and classification problems that we filtered out from the OpenML platform, as outlined in Appendix G. We consider both linear and nonlinear probabilistic models to tackle the regression and binary classification setups, which lead to predicting the parameters of a linear regression/classification model and a small nonlinear neural network based on RELU activation function. Further, we also perform the analysis with a diagonal Gaussian assumption and a normalizing flow-based amortization model trained with both a forward and reverse KL objective. We provide the results on the regression problems in (a) linear model with diagonal Gaussian assumption (Figure 8), (b) linear model with normalizing flow (Figure 9), (c) nonlinear model with diagonal Gaussian assumption (Figure 10), and (d) nonlinear model with normalizing flow (Figure 11). The results of the classification problems are shown in (a) linear model with diagonal Gaussian assumption (Figure 12), (b) linear model with normalizing flow (Figure 13), (c) nonlinear model with diagonal Gaussian assumption (Figure 14), and (d) nonlinear model with normalizing flow (Figure 15). Our experiments indicate that initializing with amortized models leads to better performance and training than models trained via maximum a-posteriori approach and initialized with the prior, i.e., $\mathcal{N}(0, I)$.

We do provide an additional baseline of initializing with XAVIER-INIT initialization, which often leads to faster convergence; however, as we consider the prior to be a unit normal, this is an unfair baseline as we assume the weights to be initialized from a different prior. We leave the work of computing Bayesian posteriors with different priors and testing an amortized Bayesian model with XAVIER-INIT prior for the future.

	q_φ	**Model**	*CNLL* (↓)										
			GM		**GMM**	**LR**		**NLR**		**LC**		**NLC**	
			2D	*100D*	*5D 2 cl*	*1D*	*100D*	*1D*	*25D*	*2D*	*100D*	*2D*	*25D*
Baseline	-	Random	438.5	22581.0	3580.9	787.1	37580.6	11728.3	146909.0	110.1	532.4	263.8	1026.0
	-	Optimization	264.5	13295.9	193.6	69.4	3559.6	80.4	17296.9	9.0	135.6	8.4	75.5
	-	MCMC	268.4	13449.6	303.4	74.4	4418.8	N/A	19584.5	14.4	521.4	154.4	1086.0
Fwd-KL	Gaussian	DeepSets	265.6	13395.0	1582.5	70.5	23509.0	9387.1	126926.9	38.8	532.0	212.1	833.3
		Transformer	265.7	13367.1	1598.6	70.5	7992.2	9522.5	126154.0	38.8	345.7	209.9	828.7
Rev-KL		DeepSets	265.5	13361.7	228.8	70.4	11952.9	100.9	22969.9	14.1	265.2	24.8	98.0
		Transformer	265.5	13357.2	247.5	70.5	4953.0	99.5	18269.8	14.1	262.7	24.0	181.7
Fwd-KL	Flow	DeepSets	265.6	13394.4	1035.0	70.4	23866.4	2936.3	102857.3	24.0	532.0	179.3	690.5
		Transformer	265.8	13363.9	593.4	70.6	7847.3	3111.4	99517.2	23.6	292.8	96.1	684.9
Rev-KL		DeepSets	265.5	13362.7	252.4	70.3	14057.6	95.0	23235.3	13.2	171.4	18.2	76.0
		Transformer	265.7	13354.3	223.4	70.5	5401.1	92.8	18029.3	13.2	206.2	18.4	107.0

Table 6: **Fixed-Dimension Posterior Prediction:** Experimental results for posterior inference on fixed dimensional datasets evaluated on estimating the (a) mean of a Gaussian (GM), (b) means of Gaussian mixture model (GMM), (c) parameters for (non-)linear regression (NLR/LR), and (d) parameters for (non-)linear classification (NLC/LC). We consider different backbone architectures and parametric distributions q_φ, and use dataset-specific Bayesian and point estimates as baselines. CNLL refers to the negative of the expected conditional log likelihood.

			CNLL (↓)									
	q_φ	**Model**	**GM**	**GMM**	**LR**		**NLR**		**LC**		**NLC**	
			100D	*5D 2 cl*	*1D*	*100D*	*1D*	*50D*	*2D*	*100D*	*2D*	*50D*
	-	Random	23096.4	3442.7	838.2	38841.6	11534.6	306025.4	110.8	544.4	271.3	1476.6
Baseline	-	Optimization	13630.0	200.4	69.2	3449.5	84.5	53915.7	8.4	135.3	9.9	96.6
	-	MCMC	13984.2	362.5	98.7	5981.6	N/A	56126.0	18.5	376.0	35.0	1951.4
Fwd-KL	Gaussian	DeepSets	15025.4	1536.5	71.4	29346.1	8341.6	257704.2	39.3	538.0	272.1	1479.8
		Transformer	14013.0	1560.1	72.6	12039.8	8294.7	252765.5	41.2	369.1	218.7	1235.1
Rev-KL		DeepSets	13825.3	236.8	72.9	12697.4	160.2	85724.3	14.5	275.2	31.1	101.7
		Transformer	13809.8	227.4	71.7	6130.9	145.7	53075.4	14.0	276.8	32.7	291.4
Fwd-KL	Flow	DeepSets	15064.4	723.8	71.6	30237.0	7063.0	212380.1	24.3	537.4	270.6	1497.8
		Transformer	13986.4	265.7	72.2	11416.0	6455.1	204521.5	25.4	318.5	189.9	1027.9
Rev-KL		DeepSets	13836.0	244.9	72.4	14961.8	165.1	91362.3	13.5	187.5	31.4	79.7
		Transformer	13815.5	221.6	71.0	6100.9	172.7	47972.8	13.0	222.3	28.4	143.6

Table 7: **Variable-Dimension Posterior Prediction:** Experimental results for posterior inference on variable dimensional datasets evaluated on estimating the (a) mean of a Gaussian (GM), (b) means of Gaussian mixture model (GMM), (c) parameters for (non-)linear regression (NLR/LR), and (d) parameters for (non-)linear classification (NLC/LC). We consider different backbone architectures and parametric distributions q_φ, and use dataset-specific Bayesian and point estimates as baselines. CNLL refers to the negative of the expected conditional log likelihood.

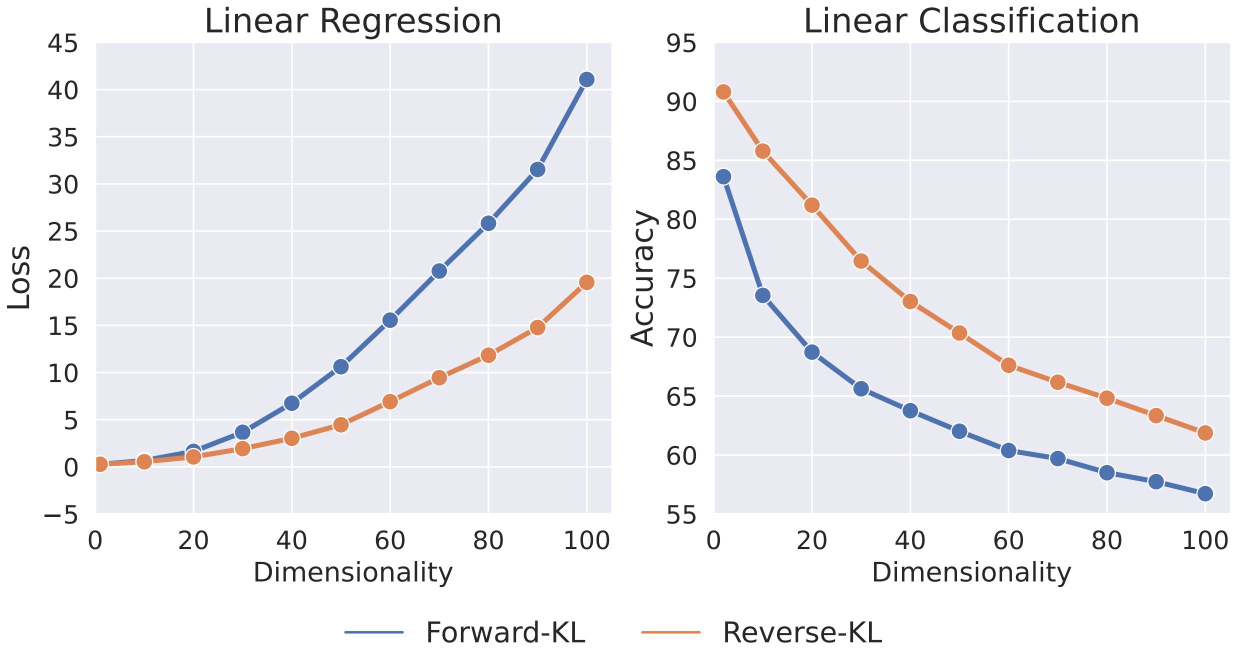


Figure 5: **Trends of Performance over different Dimensions in Variable Dimensionality Setup:** We see that our proposed reverse KL methodology outperforms the forward KL one.

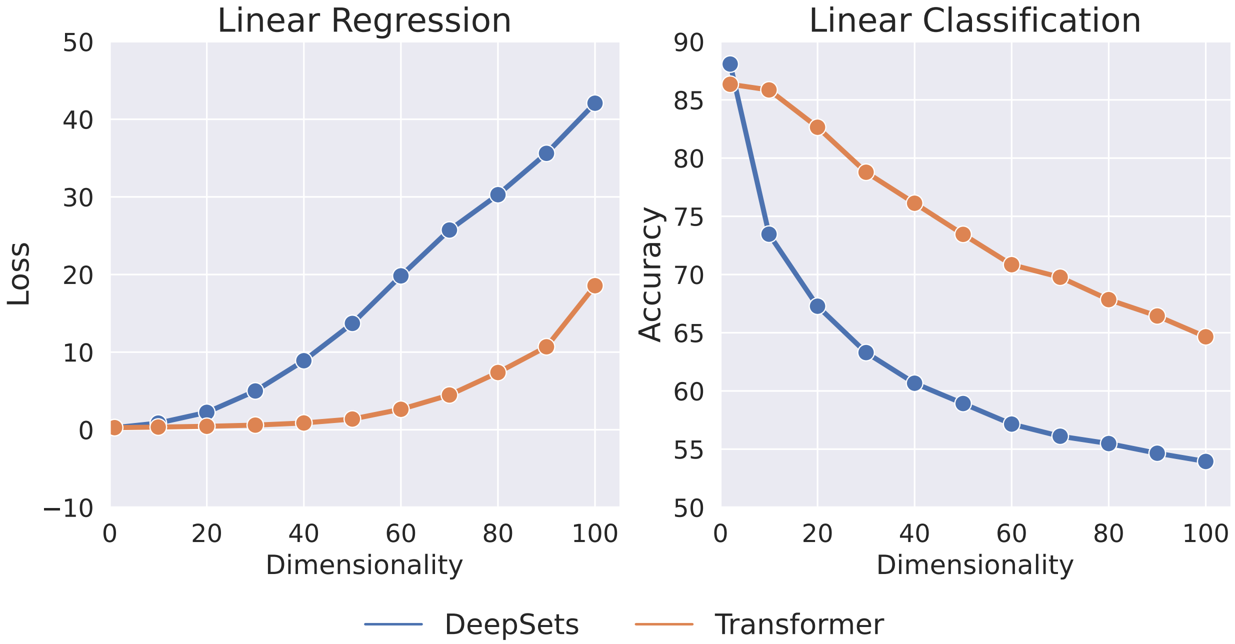


Figure 6: **Trends of Performance over different Dimensions in Variable Dimensionality Setup:** We see that transformer models generalize better to different dimensional inputs than DeepSets.

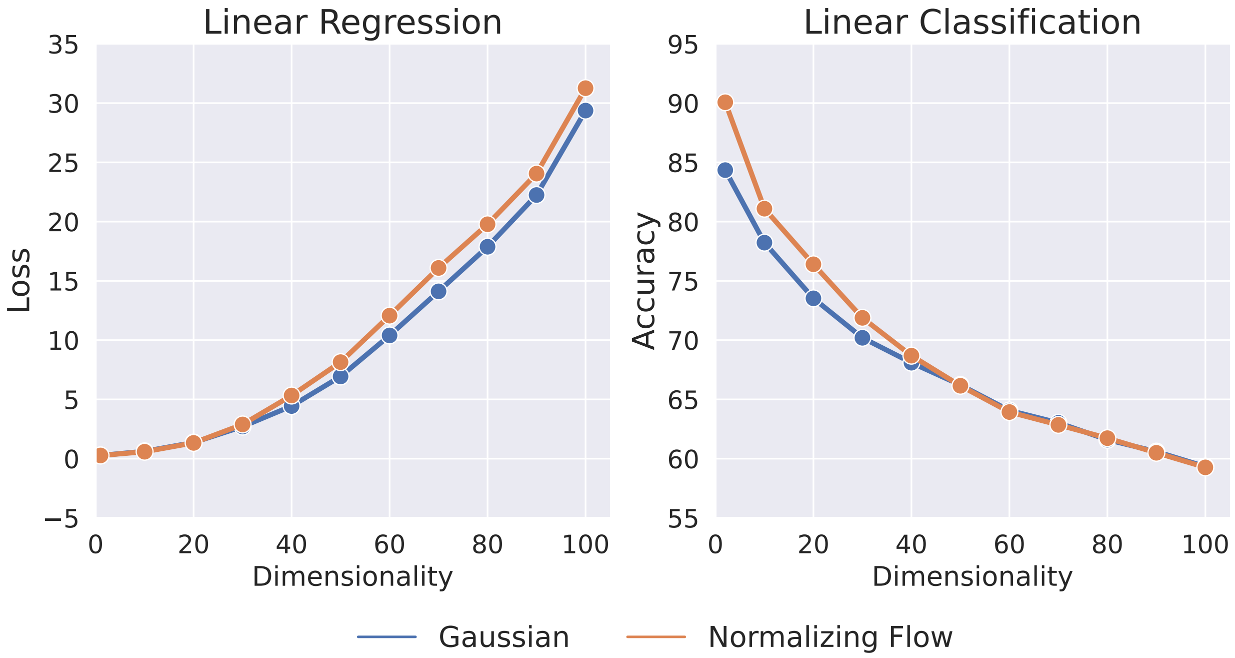


Figure 7: **Trends of Performance over different Dimensions in Variable Dimensionality Setup:** We see that normalizing flows leads to similar performances than Gaussian based variational approximation.

	q_φ	**Model**	*L_2 Loss* (↓)			*Accuracy* (↑)	
			GMM			**LC**	
			2D-2cl	*2D-5cl*	*5D-5cl*	*2D-5cl*	*100D-5cl*
Baseline	-	Random	1.88	0.72	5.06	20.28	20.03
	-	Optimization	0.17	0.12	0.43	92.04	42.31
	-	MCMC	0.20	0.13	0.65	85.18	31.73
Fwd-KL	Gaussian	DeepSets	0.91	0.53	2.42	70.96	19.94
		Transformer	0.92	0.53	2.44	71.00	26.82
Rev-KL		DeepSets	0.19	0.13	0.49	86.94	21.60
		Transformer	0.20	0.12	0.52	87.05	32.62
Fwd-KL	Flow	DeepSets	0.23	0.22	0.65	88.12	20.02
		Transformer	0.23	0.25	0.57	88.95	27.05
Rev-KL		DeepSets	0.19	0.13	0.52	88.09	21.05
		Transformer	0.19	0.12	0.48	88.21	33.20

Table 8: **Fixed-Dim Posterior Prediction:** Experimental results for posterior inference on fixed dimensional datasets evaluated on estimating the (a) means of Gaussian mixture model (GMM), and (b) parameters for linear classification (LC) for additional probabilistic model setups (eg. multi-class). We consider different backbone architectures and parametric distributions q_φ, and use dataset-specific Bayesian and point estimates as baselines. L_2 Loss and Accuracy refer to the expected posterior-predictive L_2 loss and accuracy respectively. Here, cl refers to the number of clusters for GMM and number of classes for LC.

Setup		q_φ	**Model**	L_2 *Loss* (↓)		*Accuracy* (↑)			
				NLR		**NLC**			
				1D	*25D*	*2D-2cl*	*2D-5cl*	*25D-2cl*	*25D-5cl*
TANH	Baseline	-	Random	34.48	53.62	50.09	20.03	49.96	20.09
		-	Optimization	0.28	12.02	97.21	93.41	75.67	49.41
		-	MCMC	0.30	16.25	95.80	89.48	65.08	34.77
	Fwd-KL	Gaussian	DeepSets	34.74	53.38	50.14	19.86	50.12	19.99
			Transformer	34.75	53.48	50.14	20.06	50.12	20.13
	Rev-KL		DeepSets	0.41	25.99	90.05	19.84	50.11	20.00
			Transformer	0.41	10.89	89.43	78.03	50.13	20.01
	Fwd-KL	Flow	DeepSets	33.30	53.35	49.88	20.23	50.03	20.04
			Transformer	10.60	53.58	49.87	20.41	50.02	20.17
	Rev-KL		DeepSets	0.38	26.53	89.97	49.00	49.99	19.98
			Transformer	0.38	10.89	90.56	81.79	49.95	20.02
RELU	Baseline	-	Random	64.52	793.13	50.08	20.00	49.82	19.94
		-	Optimization	0.32	95.08	97.49	95.66	80.06	59.39
		-	MCMC	N/A	106.90	96.37	93.07	73.23	45.98
	Fwd-KL	Gaussian	DeepSets	50.00	682.23	59.52	31.48	59.57	29.50
			Transformer	50.73	678.11	59.91	32.16	59.76	29.78
	Rev-KL		DeepSets	0.43	124.46	91.59	85.61	62.57	32.99
			Transformer	0.42	99.28	91.68	85.36	76.46	47.72
	Fwd-KL	Flow	DeepSets	15.31	549.13	61.98	33.55	60.98	30.78
			Transformer	16.37	530.95	75.68	38.42	61.12	30.94
	Rev-KL		DeepSets	0.39	125.90	93.13	86.93	63.87	33.31
			Transformer	0.39	97.83	92.92	86.50	75.75	49.38

Table 9: **Fixed-Dim Posterior Prediction:** Experimental results for posterior inference on fixed dimensional datasets evaluated on estimating the parameters of nonlinear regression (NLR) and classification (NLC) setups, with 1 layered MLP with different activation functions in the probabilistic model. We also consider a multi-class classification setup. We consider different backbone architectures and parametric distributions q_φ, and use dataset-specific Bayesian and point estimates as baselines. L_2 Loss and Accuracy refer to the expected posterior-predictive L_2 loss and accuracy respectively. Here, cl refers to the number classes.

Setup		q_φ	**Model**	*L_2 Loss* (↓) **NLR** *1D*	*25D*	*Accuracy* (↑) **NLC** *2D-2cl*	*2D-5cl*	*25D-2cl*	*25D-5cl*
TANH	Baseline	-	Random	52.72	55.16	50.16	19.83	50.02	20.00
		-	Optimization	0.56	30.22	96.76	89.99	69.51	40.43
		-	MCMC	0.39	32.41	94.23	81.42	53.16	24.49
	Fwd-KL	Gaussian	DeepSets	53.83	56.02	50.07	19.95	49.91	19.93
			Transformer	53.89	56.82	50.07	20.18	49.91	20.09
	Rev-KL		DeepSets	0.89	27.98	50.04	19.99	49.91	19.97
			Transformer	0.87	25.17	61.64	19.99	49.92	19.95
	Fwd-KL	Flow	DeepSets	52.46	55.85	50.47	19.79	49.93	20.06
			Transformer	52.44	55.84	50.45	20.04	49.94	20.19
	Rev-KL		DeepSets	0.68	27.97	50.67	19.72	49.99	20.06
			Transformer	0.70	27.97	86.37	19.78	50.00	20.02
RELU	Baseline	-	Random	1082.06	13301.74	49.68	20.18	49.74	20.08
		-	Optimization	2.01	1858.40	98.01	96.81	80.30	61.30
		-	MCMC	N/A	N/A	88.39	52.78	66.16	35.49
	Fwd-KL	Gaussian	DeepSets	821.57	10877.36	60.77	31.66	58.35	19.88
			Transformer	786.67	10845.96	61.21	32.12	58.28	30.17
	Rev-KL		DeepSets	1.38	2048.08	74.11	49.53	66.41	46.12
			Transformer	2.36	1976.32	87.77	76.33	66.35	30.01
	Fwd-KL	Flow	DeepSets	676.46	8236.22	62.90	33.21	60.28	20.12
			Transformer	646.60	8075.57	63.71	34.11	61.45	32.70
	Rev-KL		DeepSets	1.32	2040.10	74.98	61.65	68.06	47.05
			Transformer	2.92	1987.58	92.31	76.03	68.41	45.96

Table 10: **Fixed-Dim Posterior Prediction:** Experimental results for posterior inference on fixed dimensional datasets evaluated on estimating the parameters of nonlinear regression (NLR) and classification (NLC) setups, with 2 layered MLP with different activation functions in the probabilistic model. We also consider a multi-class classification setup. We consider different backbone architectures and parametric distributions q_φ, and use dataset-specific Bayesian and point estimates as baselines. L_2 Loss and Accuracy refer to the expected posterior-predictive L_2 loss and accuracy respectively. Here, cl refers to the number classes.

			L_2 Loss (↓)					*Accuracy* (↑)			
	q_φ	**Model**	**GM**	**GMM**			**LR**	**LC**			
			50D	*2D-2cl*	*2D-5cl*	*5D-5cl*	*50D*	*2D-5cl*	*50D-2cl*	*50D-5cl*	*100D-5cl*
Baseline	-	Random	153.45	2.24	0.66	1.63	102.04	19.97	50.02	20.02	19.96
	-	Optimization	50.51	0.18	0.12	0.33	0.76	92.22	79.74	52.17	42.58
	-	MCMC	55.13	0.34	0.18	0.49	8.10	82.39	71.30	38.01	31.15
Fwd-KL	Gaussian	DeepSets	52.04	1.30	0.49	1.21	39.63	20.00	51.77	19.95	19.99
		Transformer	51.49	1.31	0.49	1.21	2.71	63.39	69.86	40.38	26.90
Rev-KL		DeepSets	51.25	0.21	0.14	0.39	20.64	85.50	69.81	36.22	27.16
		Transformer	51.15	0.20	0.13	0.32	2.99	84.39	75.66	45.66	32.75
Fwd-KL	Flow	DeepSets	52.25	0.33	0.20	0.50	40.15	20.00	51.18	19.92	20.00
		Transformer	51.45	0.40	0.25	0.55	2.30	75.73	74.19	41.56	27.09
Rev-KL		DeepSets	51.25	0.21	0.14	0.38	20.46	86.43	69.68	25.75	22.37
		Transformer	51.17	0.20	0.13	0.33	2.44	86.37	76.71	46.06	32.98

Table 11: **Variable-Dim Posterior Prediction:** Experimental results for posterior inference on variable dimensional datasets evaluated on estimating the (a) mean of a Gaussian distribution, (b) means of Gaussian mixture model (GMM), (c) parameters for linear regression (LR), and (d) parameters for linear classification (LC) for additional probabilistic model setups (eg. multi-class). We consider different backbone architectures and parametric distributions q_φ, and use dataset-specific bayesian and point estimates as baselines. L_2 Loss and Accuracy refer to the expected posterior-predictive L_2 loss and accuracy respectively. Here, cl refers to the number of clusters for GMM and number of classes for LC.

Setup		q_φ	**Model**	*L_2 Loss* (↓)			*Accuracy* (↑)					
				NLR			**NLC**					
				1D	*50D*	*100D*	*2D-2cl*	*2D-5cl*	*50D-2cl*	*50D-5cl*	*100D-2cl*	*100D-5cl*
TANH	Baseline	-	Random	30.54	56.37	59.04	49.72	19.88	49.98	20.03	50.02	20.01
		-	Optimization	0.28	22.47	36.97	97.73	93.60	69.55	40.32	63.73	34.64
		-	MCMC	0.30	21.49	30.29	96.66	88.47	57.55	26.16	53.98	23.54
	Fwd-KL	Gaussian	DeepSets	30.98	56.31	58.84	49.71	20.06	49.93	19.93	50.02	20.04
			Transformer	30.90	55.50	58.94	49.71	20.27	49.93	20.03	50.02	20.13
	Rev-KL		DeepSets	0.57	22.28	29.06	90.99	28.41	49.93	19.93	50.02	20.04
			Transformer	0.71	16.16	49.23	91.00	74.59	64.53	19.92	54.80	20.05
	Fwd-KL	Flow	DeepSets	30.32	56.24	58.66	49.89	19.71	49.91	20.02	49.95	20.02
			Transformer	29.87	55.41	58.70	49.90	19.88	49.91	20.12	49.94	20.10
	Rev-KL		DeepSets	0.61	24.06	31.29	91.68	19.83	50.10	20.05	50.05	20.04
			Transformer	0.57	16.52	32.67	92.22	19.74	63.33	20.11	50.02	20.04
RELU	Baseline	-	Random	59.56	1609.93	3235.73	50.13	19.86	49.90	20.04	50.01	19.97
		-	Optimization	0.35	289.49	861.93	96.36	95.88	74.89	52.71	70.05	46.43
		-	MCMC	N/A	300.32	811.02	96.24	92.73	67.09	38.72	63.37	34.06
	Fwd-KL	Gaussian	DeepSets	45.24	1333.61	2744.30	49.81	20.27	50.02	19.97	50.13	20.07
			Transformer	45.12	1310.54	2739.36	58.59	32.76	58.78	29.66	57.87	28.81
	Rev-KL		DeepSets	0.76	449.68	1116.67	89.56	49.64	61.36	33.05	60.31	31.88
			Transformer	0.67	282.74	818.49	89.11	53.73	73.02	34.60	67.95	32.82
	Fwd-KL	Flow	DeepSets	36.60	1103.04	2246.73	50.08	20.09	49.74	20.13	49.87	19.95
			Transformer	33.60	1066.24	2205.13	60.15	32.71	59.95	31.18	59.05	29.89
	Rev-KL		DeepSets	0.78	480.36	1224.96	88.86	50.07	62.56	33.54	61.32	32.29
			Transformer	0.82	257.30	874.53	89.79	71.56	71.93	38.71	65.65	33.28

Table 12: **Variable-Dim Posterior Prediction:** Experimental results for posterior inference on variable dimensional datasets evaluated on estimating the parameters of nonlinear regression (NLR) and classification (NLC) setups, with 1 layered MLP with different activation functions in the probabilistic model. We also consider a multi-class classification setup. We consider different backbone architectures and parametric distributions q_φ, and use dataset-specific Bayesian and point estimates as baselines. L_2 Loss and Accuracy refer to the expected posterior-predictive L_2 loss and accuracy respectively. Here, cl refers to the number of classes.

				L_2 *Loss* (↓)			*Accuracy* (↑)					
Setup		q_φ	**Model**	**NLR**			**NLC**					
				1D	*50D*	*100D*	*2D-2cl*	*2D-5cl*	*50D-2cl*	*50D-5cl*	*100D-2cl*	*100D-5cl*
TANH	Baseline	-	Random	48.54	53.94	54.73	50.16	20.19	49.95	20.00	49.92	19.86
		-	Optimization	0.62	36.23	47.52	96.72	90.13	63.03	34.73	60.55	29.76
		-	MCMC	0.41	39.04	44.14	92.85	69.53	50.37	21.15	50.31	20.94
	Fwd-KL	Gaussian	DeepSets	48.98	53.43	54.53	49.72	20.06	50.05	19.98	50.00	19.96
			Transformer	48.78	53.16	54.53	49.72	20.28	50.05	20.12	50.00	20.04
	Rev-KL		DeepSets	23.28	26.70	27.36	49.68	20.06	50.04	19.98	50.01	19.97
			Transformer	2.34	23.08	42.81	49.69	20.07	50.04	19.97	49.99	19.96
	Fwd-KL	Flow	DeepSets	48.75	53.77	54.66	49.75	19.99	50.03	19.92	49.88	20.09
			Transformer	48.06	53.38	54.39	49.75	20.21	50.04	20.08	49.88	20.16
	Rev-KL		DeepSets	23.27	26.71	27.36	49.84	20.07	50.02	19.96	49.94	20.00
			Transformer	3.07	22.54	46.77	49.56	20.00	50.02	19.95	49.97	20.02
RELU	Baseline	-	Random	1021.41	26069.28	51949.46	50.03	20.29	50.01	20.01	50.10	19.92
		-	Optimization	1.81	4762.79	12508.07	98.20	97.31	79.06	58.43	77.30	56.16
		-	MCMC	N/A	N/A	N/A	67.62	27.30	62.35	29.78	64.10	32.49
	Fwd-KL	Gaussian	DeepSets	782.52	20746.84	39559.89	50.20	20.26	50.21	20.25	49.82	19.84
			Transformer	806.08	20461.45	39517.84	59.96	33.22	60.09	31.45	59.91	31.73
	Rev-KL		DeepSets	7.93	6024.49	13894.65	79.88	60.92	68.38	46.87	68.24	51.03
			Transformer	8.41	4788.45	12953.82	82.76	59.47	68.37	28.80	68.21	26.79
	Fwd-KL	Flow	DeepSets	741.81	20098.74	39025.35	49.72	20.28	50.28	20.00	49.89	20.09
			Transformer	634.92	17089.49	33259.73	61.76	34.75	61.93	32.48	61.87	33.44
	Rev-KL		DeepSets	5.13	7227.68	15336.46	74.27	62.05	70.05	46.90	69.77	50.85
			Transformer	7.62	4818.14	13531.90	81.32	69.07	70.19	47.82	69.90	51.16

Table 13: **Variable-Dim Posterior Prediction:** Experimental results for posterior inference on variable dimensional datasets evaluated on estimating the parameters of nonlinear regression (NLR) and classification (NLC) setups, with 2 layered MLP with different activation functions in the probabilistic model. We also consider a multi-class classification setup. We consider different backbone architectures and parametric distributions q_φ, and use dataset-specific Bayesian and point estimates as baselines. L_2 Loss and Accuracy refer to the expected posterior-predictive L_2 loss and accuracy respectively. Here, cl refers to the number of classes.

			LR			NLR			GP			← χ_{sim}
	q_φ	**Model**	*LR*	*NLR*	*GP*	*LR*	*NLR*	*GP*	*LR*	*NLR*	*GP*	← χ_{real}
Baseline	-	Random	4.094	20.394	2.659	4.094	20.394	2.659	4.094	20.394	2.659	
	-	Optimization	0.258	1.511	0.262	0.258	1.511	0.262	0.258	1.511	0.262	
	-	MCMC	0.285	4.864	0.271	0.285	4.864	0.271	0.285	4.864	0.271	
Fwd-KL	Gaussian	DeepSets	0.265	1.853	0.267	-	-	-	-	-	-	
		Transformer	0.265	5.089	0.268	-	-	-	-	-	-	
Rev-KL		DeepSets	0.264	1.643	0.267	0.264	1.472	0.267	0.262	2.246	0.264	
		Transformer	0.264	2.431	0.267	0.264	1.477	0.266	0.264	8.572	0.265	
Fwd-KL	Flow	DeepSets	0.264	1.760	0.267	-	-	-	-	-	-	
		Transformer	0.265	1.964	0.268	-	-	-	-	-	-	
Rev-KL		DeepSets	0.264	1.600	0.267	0.264	1.472	0.267	0.262	2.973	0.264	
		Transformer	0.265	1.945	0.267	0.265	1.480	0.268	0.264	7.561	0.264	

Table 14: **Model Misspecification LR Model**: Posterior predictive performance with L2 loss metric for the linear regression model. The top row highlights the data used to train the model (LR: Linear Regression, NLR: Nonlinear Regression (TANH), GP: Gaussian Process Regression), and the second row highlights the data used for evaluation. We note that a forward KL method can only be trained on data simulated from the assumed probabilistic model and thus cannot be trained on nonlinear data if the assumed probabilistic model is linear.

			LR			NLR			GP			$\leftarrow \chi_{sim}$
	q_φ	**Model**	*LR*	*NLR*	*GP*	*LR*	*NLR*	*GP*	*LR*	*NLR*	*GP*	$\leftarrow \chi_{real}$
Baseline	-	Random	17.805	34.321	16.339	17.805	34.321	16.339	17.805	34.321	16.339	
	-	Optimization	0.270	0.273	0.012	0.270	0.273	0.012	0.270	0.273	0.012	
	-	MCMC	0.290	0.295	0.043	0.290	0.295	0.043	0.290	0.295	0.043	
Fwd-KL	Gaussian	DeepSets	-	-	-	15.739	33.859	14.677	-	-	-	
		Transformer	-	-	-	15.254	33.791	14.867	-	-	-	
Rev-KL		DeepSets	0.364	2.900	0.334	0.375	0.394	0.148	0.371	4.286	0.068	
		Transformer	0.369	1.752	0.313	0.380	0.394	0.153	0.490	3.332	0.072	
Fwd-KL	Flow	DeepSets	-	-	-	15.228	30.254	14.418	-	-	-	
		Transformer	-	-	-	7.575	10.065	8.355	-	-	-	
Rev-KL		DeepSets	0.342	1.571	0.285	0.348	0.371	0.119	0.370	3.290	0.067	
		Transformer	0.346	1.682	0.287	0.345	0.369	0.113	0.468	5.749	0.060	

Table 15: **Model Misspecification NLR (tanh) Model**: Posterior predictive performance with L2 loss metric for the nonlinear regression model with tanh activation function. The top row highlights the data used to train the model (LR: Linear Regression, NLR: Nonlinear Regression (TANH), GP: Gaussian Process Regression), and the second row highlights the data used for evaluation. We note that a forward KL method can only be trained on data simulated from the assumed probabilistic model and thus cannot be trained on linear or GP data if the assumed probabilistic model is a single-layered nonlinear MLP.

			LR			NLR			GP			← χ_{sim}
	q_φ	**Model**	*LR*	*NLR*	*GP*	*LR*	*NLR*	*GP*	*LR*	*NLR*	*GP*	← χ_{real}
Baseline	-	Random	34.178	68.280	32.611	34.178	68.280	32.611	34.178	68.280	32.611	
	-	Optimization	0.284	0.321	0.027	0.284	0.321	0.027	0.284	0.321	0.027	
	-	MCMC	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	
Fwd-KL	Gaussian	DeepSets	-	-	-	26.267	50.692	26.322	-	-	-	
		Transformer	-	-	-	25.236	51.781	25.150	-	-	-	
Rev-KL		DeepSets	0.329	2190.208	0.303	0.339	0.430	0.164	0.353	N/A	0.074	
		Transformer	0.336	6.607	0.287	0.336	0.421	0.152	0.474	17.762	0.069	
Fwd-KL	Flow	DeepSets	-	-	-	9.788	14.704	10.257	-	-	-	
		Transformer	-	-	-	9.823	15.840	10.420	-	-	-	
Rev-KL		DeepSets	0.320	7.142	0.271	0.326	0.397	0.124	0.341	27107.141	0.056	
		Transformer	0.323	4.877	0.264	0.330	0.383	0.122	0.461	16.243	0.053	

Table 16: **Model Misspecification NLR (relu) Model**: Posterior predictive performance with L2 loss metric for the nonlinear regression model with ReLU activation function. The top row highlights the data used to train the model (LR: Linear Regression, NLR: Nonlinear Regression (relu), GP: Gaussian Process Regression), and the second row highlights the data used for evaluation. We note that a forward KL method can only be trained on data simulated from the assumed probabilistic model and thus cannot be trained on linear or GP data if the assumed probabilistic model is a single-layered nonlinear MLP.

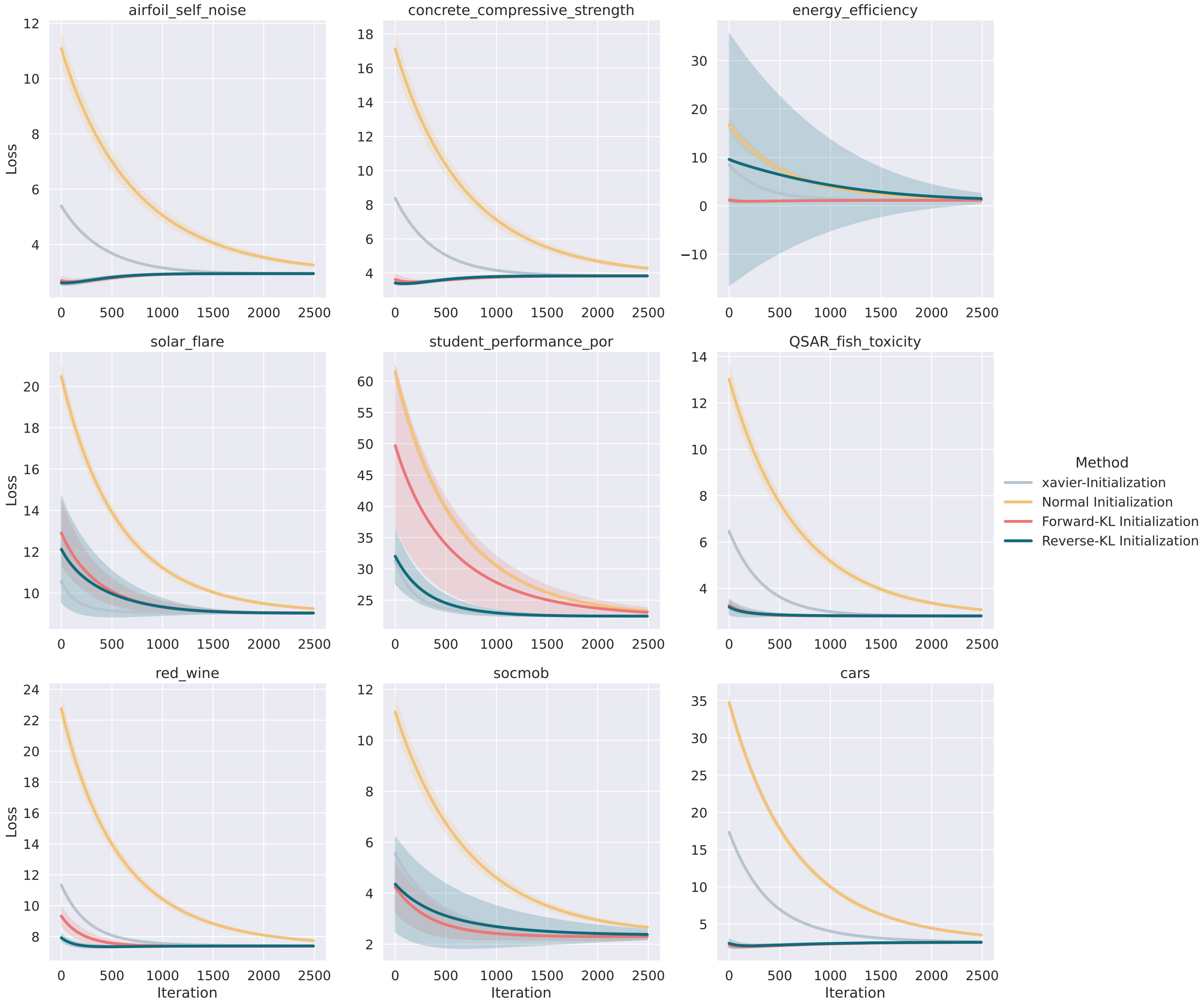


Figure 8: **Tabular Experiments | Linear Regression with Diagonal Gaussian**: For every regression dataset from the OpenML platform considered, we initialize the parameters of a linear regression-based probabilistic model with the amortized inference models which were trained with a diagonal Gaussian assumption. The parameters are then further trained with maximum-a-posteriori (MAP) estimate with gradient descent. Reverse and Forward KL denote initialization with the correspondingly trained amortized model. Optimization refers to a MAP-based optimization baseline initialized from the prior $\mathcal{N}(0, I)$, whereas Xavier-Optimization refers to initialization from the Xavier initialization scheme.

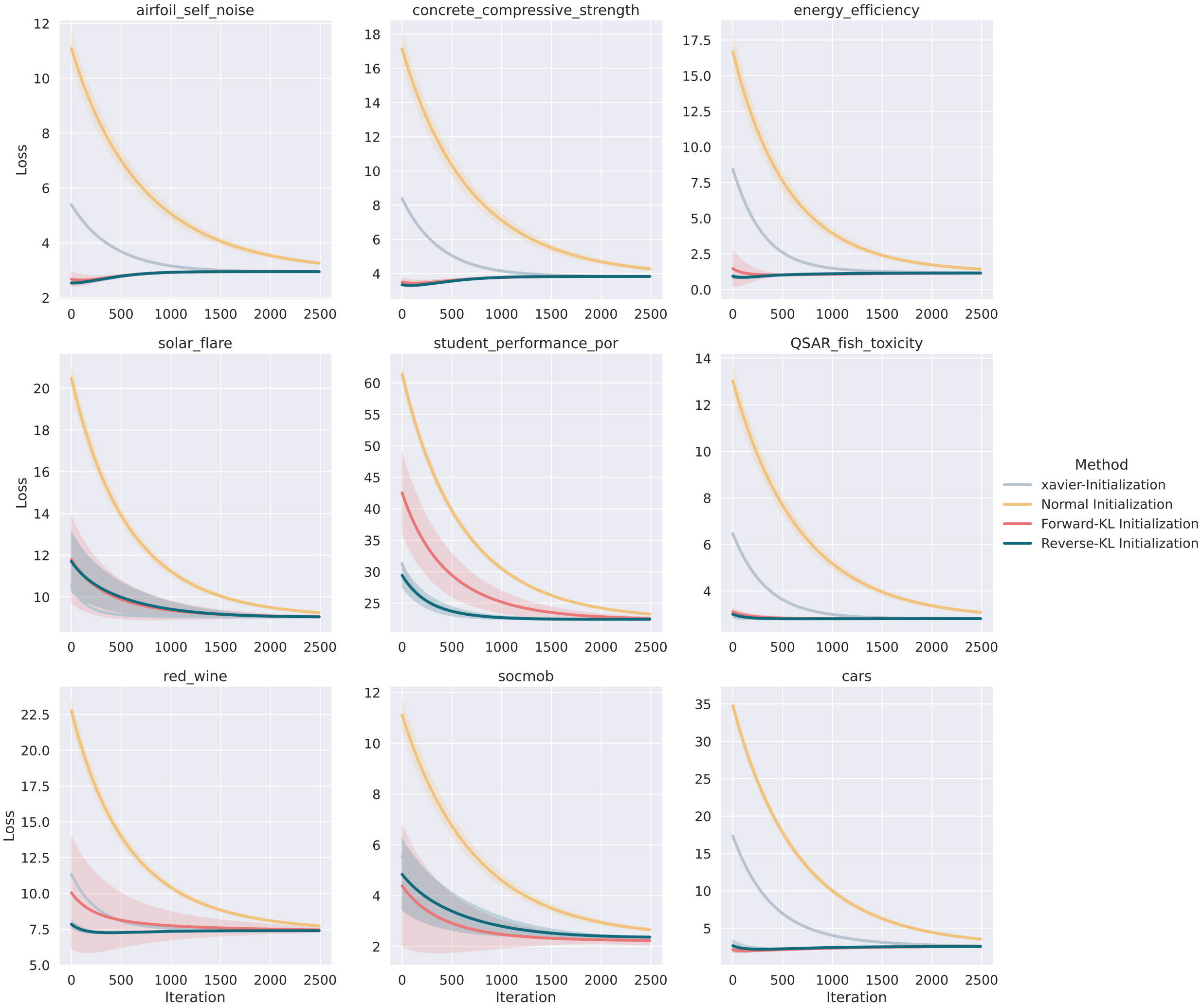


Figure 9: **Tabular Experiments | Linear Regression with Normalizing Flow**: For every regression dataset from the OpenML platform considered, we initialize the parameters of a linear regression-based probabilistic model with the amortized inference models which were trained with a normalizing flow-based model. The parameters are then further trained with maximum-a-posteriori (MAP) estimate with gradient descent. Reverse and Forward KL denote initialization with the correspondingly trained amortized model. Optimization refers to a MAP-based optimization baseline initialized from the prior $\mathcal{N}(0, I)$, whereas Xavier-Optimization refers to initialization from the Xavier initialization scheme.

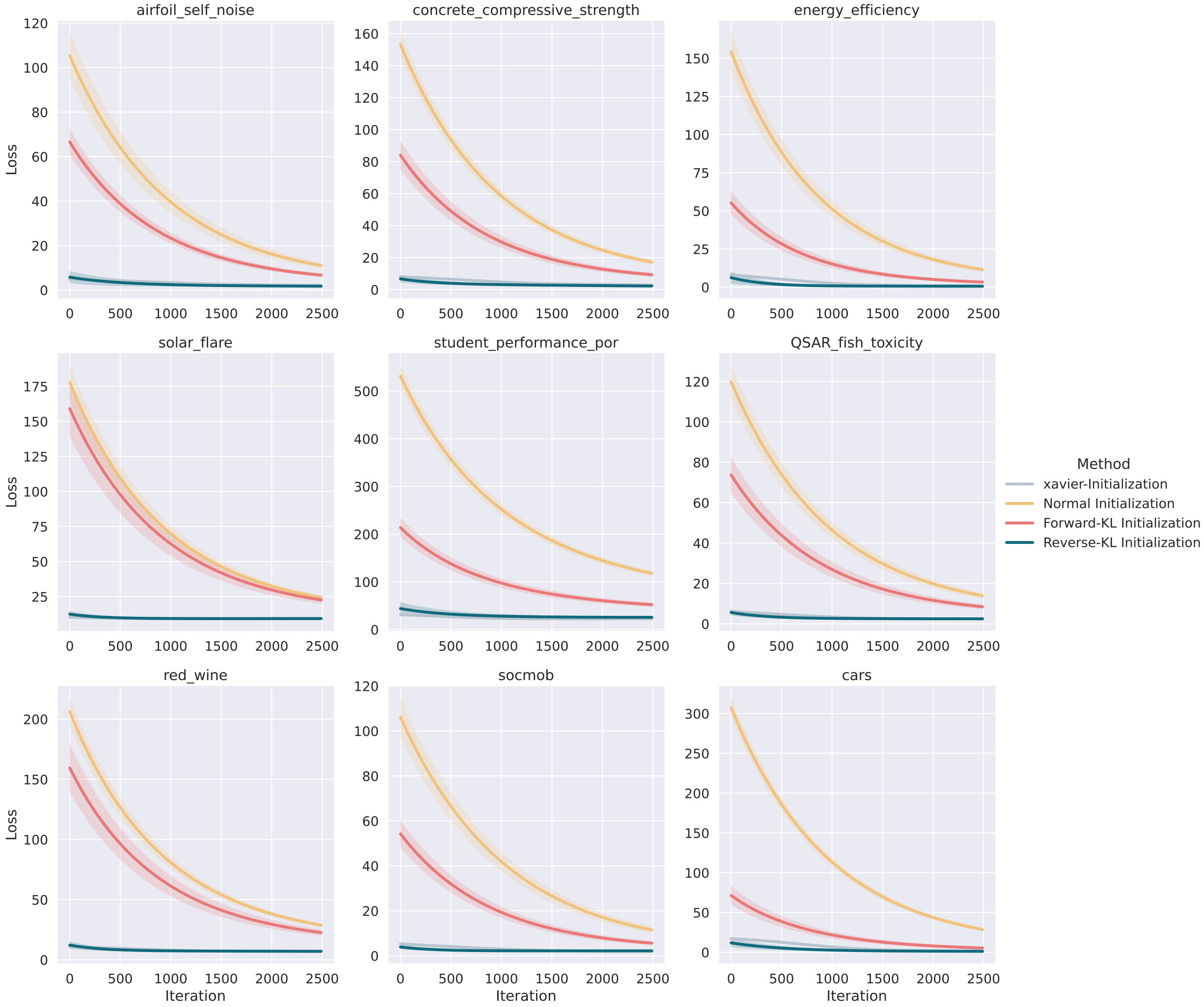


Figure 10: **Tabular Experiments | Nonlinear Regression with Diagonal Gaussian**: For every regression dataset from the OpenML platform considered, we initialize the parameters of a nonlinear regression-based probabilistic model with the amortized inference models which were trained with a diagonal Gaussian assumption. The parameters are then further trained with maximum-a-posteriori (MAP) estimate with gradient descent. Reverse and Forward KL denote initialization with the correspondingly trained amortized model. Optimization refers to a MAP-based optimization baseline initialized from the prior $\mathcal{N}(0, I)$, whereas Xavier-Optimization refers to initialization from the Xavier initialization scheme.

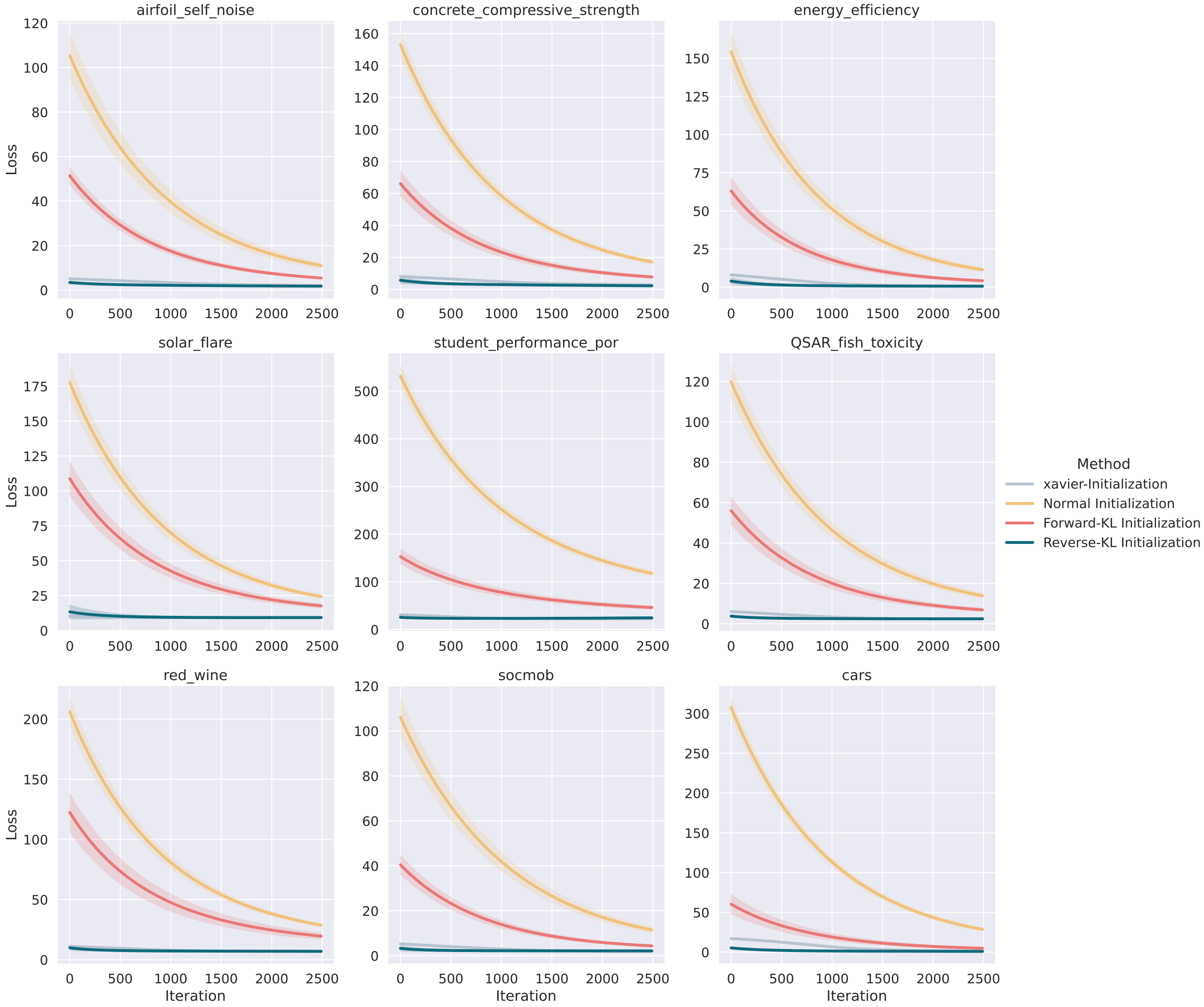


Figure 11: **Tabular Experiments | Nonlinear Regression with Normalizing Flow**: For every regression dataset from the OpenML platform considered, we initialize the parameters of a nonlinear regression-based probabilistic model with the amortized inference models which were trained with a normalizing flow-based model. The parameters are then further trained with maximum-a-posteriori (MAP) estimate with gradient descent. Reverse and Forward KL denote initialization with the correspondingly trained amortized model. Optimization refers to a MAP-based optimization baseline initialized from the prior $\mathcal{N}(0, I)$, whereas Xavier-Optimization refers to initialization from the Xavier initialization scheme.

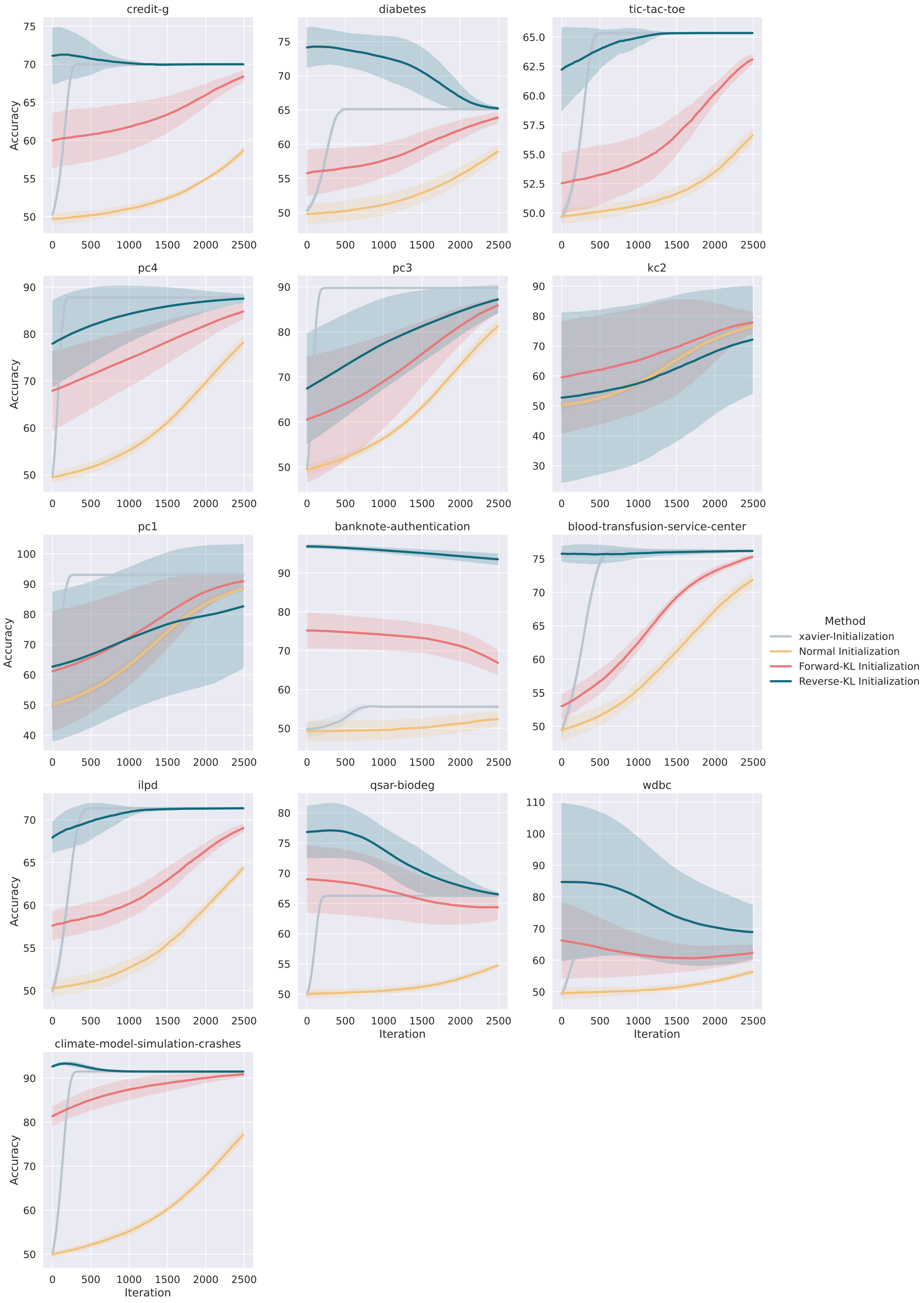


Figure 12: **Tabular Experiments | Linear Classification with Diagonal Gaussian**: For every classification dataset from the OpenML platform considered, we initialize the parameters of a linear classification-based probabilistic model with the amortized inference models which were trained with a diagonal Gaussian assumption. The parameters are then further trained with maximum-a-posteriori (MAP) estimate with gradient descent. Reverse and Forward KL denote initialization with the correspondingly trained amortized model. Optimization refers to a MAP-based optimization baseline initialized from the prior $\mathcal{N}(0, I)$, whereas Xavier-Optimization refers to initialization from the Xavier initialization scheme.

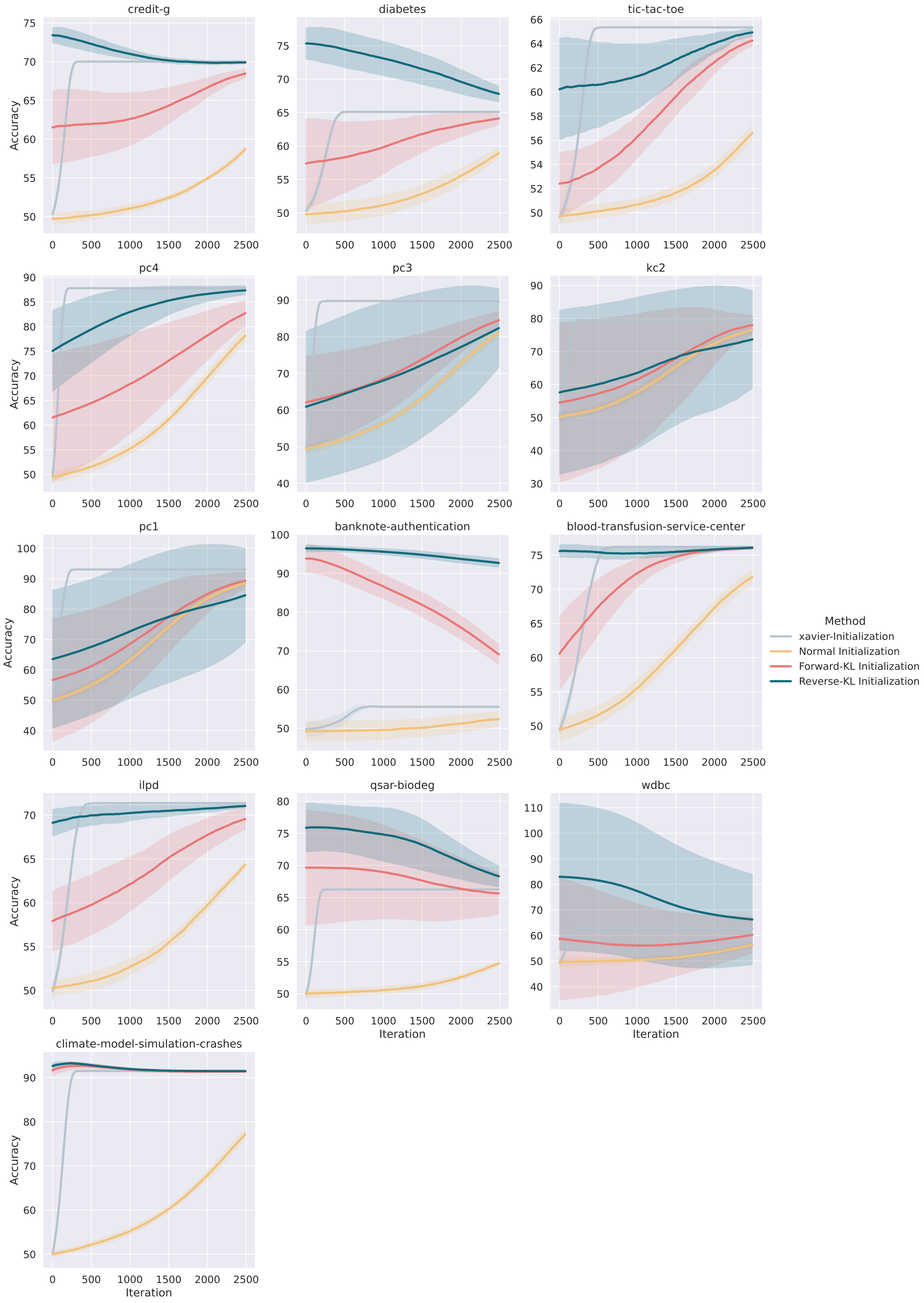


Figure 13: **Tabular Experiments | Linear Classification with Normalizing Flow**: For every classification dataset from the OpenML platform considered, we initialize the parameters of a linear classification-based probabilistic model with the amortized inference models which were trained with a normalizing flow-based model. The parameters are then further trained with maximum-a-posteriori (MAP) estimate with gradient descent. Reverse and Forward KL denote initialization with the correspondingly trained amortized model. Optimization refers to a MAP-based optimization baseline initialized from the prior $\mathcal{N}(0, I)$, whereas Xavier-Optimization refers to initialization from the Xavier initialization scheme.

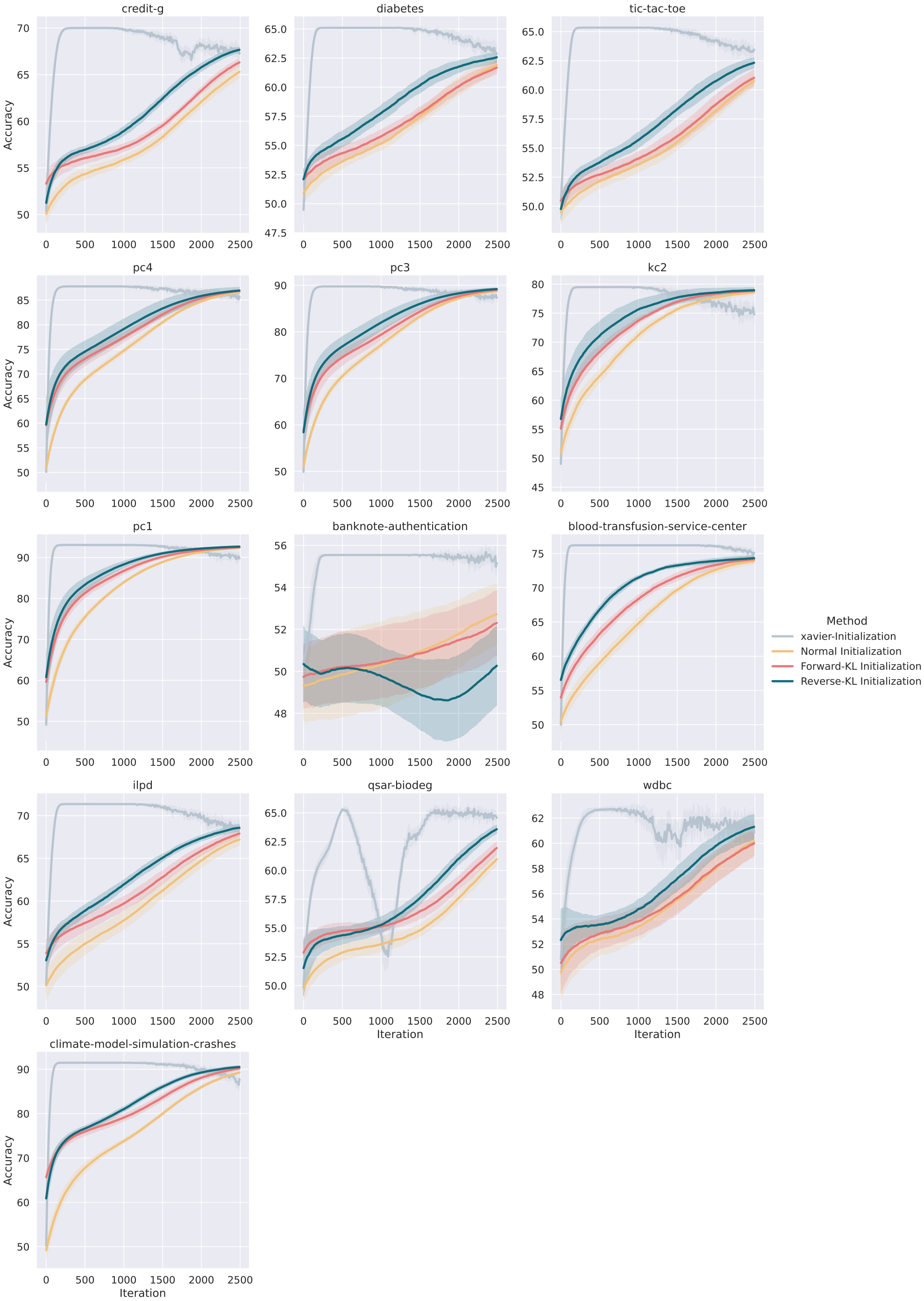


Figure 14: **Tabular Experiments | Nonlinear Classification with Diagonal Gaussian**: For every classification dataset from the OpenML platform considered, we initialize the parameters of a nonlinear classification-based probabilistic model with the amortized inference models which were trained with a diagonal Gaussian assumption. The parameters are then further trained with maximum-a-posteriori (MAP) estimate with gradient descent. Reverse and Forward KL denote initialization with the correspondingly trained amortized model. Optimization refers to a MAP-based optimization baseline initialized from the prior $\mathcal{N}(0, I)$, whereas Xavier-Optimization refers to initialization from the Xavier initialization scheme.

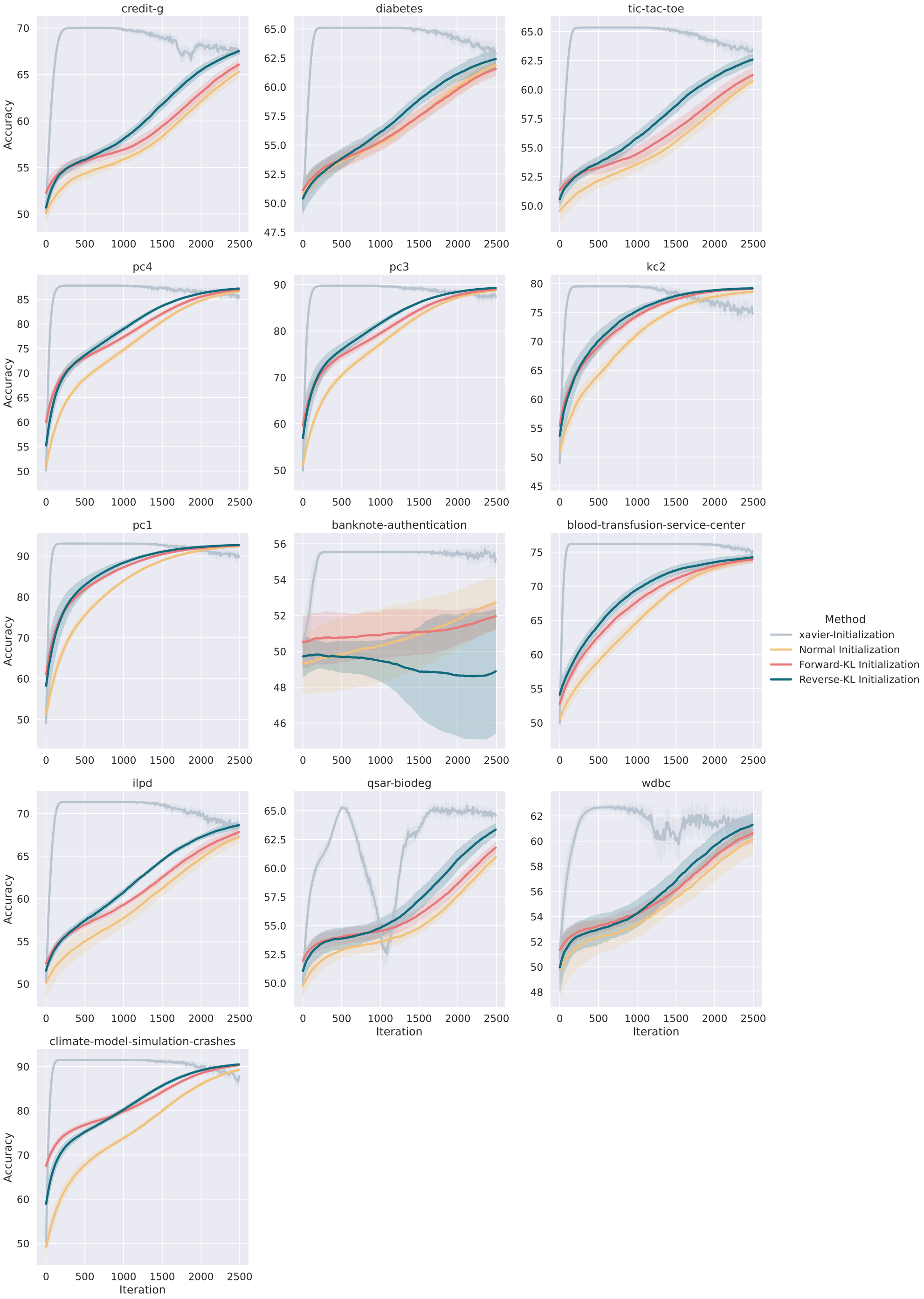


Figure 15: **Tabular Experiments | Linear Classification with Normalizing Flow**: For every classification dataset from the OpenML platform considered, we initialize the parameters of a linear classification-based probabilistic model with the amortized inference models which were trained with a normalizing flow-based model. The parameters are then further trained with maximum-a-posteriori (MAP) estimate with gradient descent. Reverse and Forward KL denote initialization with the correspondingly trained amortized model. Optimization refers to a MAP-based optimization baseline initialized from the prior $\mathcal{N}(0, I)$, whereas Xavier-Optimization refers to initialization from the Xavier initialization scheme.