Nonparametric posterior normalizing flows

Evan Ott [1] **Sinead A. Williamson** [1 2]

Abstract

Normalizing flows allow us to describe complex probability distributions, and can be used to perform flexible maximum likelihood density estimation (Dinh et al., 2014). Such maximum likelihood density estimation is likely to overfit, particularly if the number of observations is small. Traditional Bayesian approaches offer the prospect of capturing posterior uncertainty, but come at high computational cost and do not provide an intuitive way of incorporating prior information. A nonparametric learning approach (Lyddon et al., 2018) allows us to combine observed data with priors on the space of observations. We present a scalable approximate inference algorithm for nonparametric posterior normalizing flows, and show that the resulting distributions can yield improved generalization and uncertainty quantification.

1. Introduction

Normalizing flows (Tabak and Vanden-Eijnden, 2010; Dinh et al., 2014; Rezende and Mohamed, 2015) allow us to construct flexible families of probability distributions on $\mathbb{R}^d$ via a change-of-variables approach, where the change-of-variables transform is modeled using an invertible, differentiable neural network $g_\phi(z)$. When used for density estimation, normalized flows are typically trained to maximize the likelihood of the observations (possibly incorporating some regularization terms). This means we do not capture epistemic uncertainty, and are at risk of overfitting, particularly when working with small datasets. One solution might be to adopt ideas from Bayesian deep learning to approximate the Bayesian posterior, by assuming a prior distribution over the weights of the neural network (MacKay, 1992; Graves, 2011; Neal, 2012; Blundell et al., 2015). This has two drawbacks. Firstly, posterior Bayesian inference in neural networks is typically computationally challenging. Secondly, we lose one of the key advantages of a Bayesian approach: the ability to incorporate meaningful prior knowledge. While, for example, a multivariate Gaussian prior on the weights of a neural network admits a valid posterior distribution, it is not clear how the parameters of such a prior can reasonably incorporate our prior beliefs about the data generating mechanism.

Nonparametric posterior learning (NPL, Lyddon et al., 2018) has been proposed as an alternative to classical Bayesian inference. Rather than place a prior on the parameter space, NPL specifies a distribution on the space of data generating mechanisms. Inference proceeds by generating samples from this distribution over distributions, and then deterministically optimizing the expection of some function with respect to the sampled distribution. While conceptually straightforward and trivially parallelizable, this can be costly if each optimization is computationally demanding. Instead, we propose an approximate NPL algorithm that partitions the normalizing flow architecture into a shared neural network (trained once, and used for unlimited posterior samples), and sample-specific latent distributions (which have an analytically tractable form, and so can be easily calculated for a new posterior sample). This allows us to generate high-quality approximate nonparametric posterior samples from normalizing flows.

2. Background

2.1. Nonparametric posterior learning

Consider a set of observations $x_{1:n} \overset{iid}{\sim} F_0$, where F_0 is some unknown data generating distribution. We want to learn about some parameter of interest $\theta \in \Theta$—for example, the mean or median of the distribution, or the parameters of a neural network trained on unlimited data. If θ is a sufficient statistic for a parametric model, a standard Bayesian approach is to place a prior on θ and use the likelihood under that parametric model to obtain a posterior distribution. If the parametric family contains the true data generating mechanism, this posterior will converge to the true value θ_0.

Nonparametric posterior learning provides an alternative notion of a posterior, that does not rely on knowing the "correct" distributional family, and allows us to describe the relationship between parameter and data in terms of an arbi-

*Equal contribution [1]University of Texas at Austin [2]Currently at Apple; work done while at UT Austin. Correspondence to: Evan Ott <evan.ott@utexas.edu>.

Accepted to *ICML workshop on Structured Probabilistic Inference & Generative Modeling*, Honolulu, Hawaii, USA. 2023.

trary loss (not necessarily a likelihood). The key intuition is that, were we to have infinitely many samples from F_0, we would have no uncertainty in θ, and in many cases could obtain θ_0 as the minimizer of some loss $\mathcal{L}_\theta(x) = \prod_i \ell_\theta(x_i)$, i.e.,

$$\theta_0 := \theta(F_0) = \underset{\theta \in \Theta}{\operatorname{argmin}} \int_{\mathcal{X}} \ell_\theta(x) dF_0(x). \tag{1}$$

Here, $\ell_\theta(x)$ could be the log likelihood (analogous to standard Bayesian inference) or a general loss function (analogous to generalized Bayesian inference (Bissiri et al., 2016)).

Rather than place a prior on θ, NPL specifies a distribution $\pi_{\mathcal{F}}$ directly on F_0 (alternatively, on $x_{n+1:\infty}$, see Fong et al. (2021)). Under this specification, the appropriate posterior distribution—which we refer to as the nonparametric posterior—over θ is given by

$$\widetilde{\pi}(\theta | x_{1:n}) = \int_{\mathcal{M}} \delta_{\theta(F)} \pi_{\mathcal{F}}(dF | x_{1:n}), \tag{2}$$

where $\mathcal{M}$ is the space of distributions over probability measures on $\mathbb{R}^d$. Species sampling priors provide a natural choice for $\pi_{\mathcal{F}}$, since they have unbounded support and the conditional distribution has positive mass at $x_{1:n}$. In this paper, we use a Dirichlet process with base measure Ω and concentration parameter α, following Fong et al. (2019). Ω can be seen as a prior guess at the empirical distribution, and α as the relative importance assigned to this distribution. While the nonparametric posterior in Eq. 2 is not analytically tractable, we can sample from it by repeatedly sampling $F^{(b)} \sim \pi_{\mathcal{F}}(dF | x_{1:n})$ and calculating $\theta(F^{(b)})$ following Eq. 1. In practice, sampling an infinite-dimensional probability distribution is not feasible; instead we use a finite approximation (see App. A.1).

2.2. Normalizing flows

If Z is a continuous random variable on $\mathbb{R}^d$ with known probability density function (pdf) $h(z)$, and $g : \mathbb{R}^d \to \mathbb{R}^d$ is an invertible function, then we can use the change of variables formula to obtain the pdf of $X = g(Z)$, as

$$p(x) = h\left(g^{-1}(x)\right) \left| \det \left(\frac{\partial g^{-1}(x)}{\partial x} \right) \right|.$$

Normalizing flows (NFs, Dinh et al., 2014; Rezende and Mohamed, 2015) seek to *learn* an appropriate invertible mapping g_ϕ between $h(z)$ and $p(x)$. Typically, g_ϕ is parametrized by an invertible, differentiable neural network designed to have a tractable Jacobian. If we find $\operatorname{argmax}_\phi p(x; \phi, h)$ we can then sample from, and evaluate probabilities under, the associated maximum likelihood data generating process (Dinh et al., 2014). NFs are also frequently used to learn the latent distribution in a variational autoencoder, allowing for more flexible latent representations than the typical Gaussian distribution (Rezende and Mohamed, 2015).

3. Nonparametric posterior normalizing flows

As discussed in Sec. 2.2, NFs can be used to estimate the probability distribution underlying a given dataset. However, this distribution is learned in a maximum likelihood manner, with no room for uncertainty in identifying the distribution and no mechanism for incorporating prior information.

A standard Bayesian approach would be to place a prior on the flow parameters ϕ, and update our uncertainty about these parameters based on the data. In practice, this can be computationally challenging. Moreover, it is not clear how we should specify these priors, since there is not an interpretable relationship between the distribution of the weights in the neural network and the behavior of the overall NF. In particular, it is hard to imagine an *informative* prior, that incorporates prior knowledge about the data. An NPL approach allows us to distill our prior knowledge into a distribution over datasets, rather than over parameters. For example, we might describe our prior beliefs about the data generating mechanism in terms of a user-specified distribution, an empirical distribution from a related dataset, or a pre-trained generative model.

3.1. A naïve posterior normalizing flow

Let $\mathcal{G}_\Phi = \{g_\phi, \phi \in \Phi\}$ be a family of differentiable, invertible functions $g_\phi : \mathbb{R}^d \to \mathbb{R}^d$, and let $h = N(0, I)$ be a standard multivariate normal distribution on $\mathbb{R}^d$. Combined, these specify a family $\mathcal{F}_\Phi = \{f_\phi(x), \phi \in \Phi\}$ of distributions of the form

$$f_\phi(x) = h\left(g_\phi^{-1}(x)\right) \left| \det \left(\frac{\partial g_\phi^{-1}(x)}{\partial x} \right) \right|.$$

Let $\ell_\phi(x)$ be the negative log likelihood of x under f_ϕ (or some related loss), and assume $x_1, \ldots, x_n$ are iid samples from some true distribution F_0, so our "parameter of interest" is $\phi_0(F_0) = \operatorname{argmin}_\phi \int \ell_\phi(x) dF_0(x)$. We can directly apply the approach of Sec. 2.1 to obtain a distribution over ϕ, by training B independent NFs on B weighted datasets sampled from the posterior given $x_1, \ldots, x_n$ of an (approximate) Dirichlet process, with concentration parameter α and base measure Ω; see App. 2 for more details.

This approach is appealing for several reasons. It allows us to specify a base measure Ω directly on the observation space, which is more intuitive than a prior on parameter space in the case of "black box" models. Our distribution $\pi_{\mathcal{F}}$ captures the intuition that F_0 will be somewhere in between this base measure Ω and our observed empirical

distribution. Following Fong et al. (2019), the resulting nonparametric posterior is consistent at θ_0 and (if $\alpha = 0$) its posterior predictive will asymptotically dominate the standard Bayesian posterior predictive. However, this naïve approach is computationally demanding, since we must train an independent flow for each bootstrapped sample. While these can be trained in parallel, this is still a significant investment of resources; as a result we only explore this approach on toy data with limited sample size.

3.2. Jointly learning multiple flows

To reduce computational cost, we elect to train a single NF architecture to learn all B posterior samples. To do so, we augment the flow model described above by replacing the fixed latent distribution h with a parametrized distribution $h_\mu = N(\mu, I)$, such that we have a family $\mathcal{F}_{\Theta,\Phi} = \{f_{\mu,\phi}(x); \mu \in M, \phi \in \Phi\}$, with

$$f_{\mu,\phi}(x) = h_\mu\left(g_\phi^{-1}(x)\right)\left|\det\left(\frac{\partial g_\phi^{-1}(x)}{\partial x}\right)\right|.$$

We then learn a nonparametric posterior of the form

$$\begin{aligned}\widetilde{\pi}_{\hat{\phi}}(\mu|x_{1:n}) &= \textstyle\int_{\mathcal{M}} \delta_{\mu_{\hat{\phi}}(F)} \pi_{\mathcal{F}}(dF|x_{1:n}) \\ \mu_\phi(F) &= \operatorname{argmin}_{\mu\in M} \textstyle\int_{\mathbb{R}^d} \ell_{\mu,\phi}(x) dF(x) \\ \hat{\phi} &= \operatorname{argmax}_\phi \widetilde{\pi}_\phi(\mu|x_{1:n}).\end{aligned}$$

In other words, we globally optimize the parameters of the neural network, while locally optimizing the mean μ of the latent distribution. This type of approach—optimizing some parameters; inferring others—is often used in Bayesian inference, where we infer a posterior distribution over local parameters and optimize with respect to global parameters.

Alg. 3 in App. A.1 describes how we might achieve this. First, we jointly optimize our flow parameters ϕ by repeatedly sampling $F \sim \pi_{\mathcal{F}}(F|x_{1:n})$; setting the latent mean μ to the maximum likelihood values given F and the current ϕ; and performing gradient descent on ϕ holding F constant. This allows us to construct a single flow that can, in theory, map any point in the support of $\pi_{\mathcal{F}}(F|x)$ to the latent space, implying a distribution over the latent space for any distribution in observation space. Since, in training, the distributions in latent space are encouraged to be Gaussian, and since we train on a wide variety of samples from $\pi_{\mathcal{F}}(F|x_{1:n})$, this encourages learning a flow that maps arbitrary distributions in the support of $\pi_{\mathcal{F}}(F|x_{1:n})$ to Gaussians in the latent space. We then fix the flow parameters $\phi = \hat{\phi}$ and generate samples from the nonparametric posterior by repeatedly sampling $F^{(b)} \sim \pi_{\mathcal{F}}(F|x_{1:n})$; mapping the sample back to the latent space; and obtaining the maximum likelihood parameters $\mu^{(b)}$ of the latent distribution. We note that given $F^{(b)} = \sum_i w_i \delta_{\psi_i}$, conditioned on the flow parameters $\hat{\phi}$, $\mu^{(b)}$ is simply the weighted mean $\sum_i w_i g_{\hat{\phi}}^{-1}(\psi_i)$.

4. Related work

Several works have applied a classical Bayesian approach to learning neural networks, placing priors on the weights and updating them via Laplace approximations (MacKay, 1992; Daxberger et al., 2021), variational inference (Graves, 2011; Blundell et al., 2015), or assumed density filtering (Hernández-Lobato and Adams, 2015). Of particular relevance to this paper, Trippe and Turner (2018) uses a variational inference approach to learn conditional normalizing flows. However, these Bayesian methods tend to be slower than non-Bayesian approaches, and do not allow us to easily incorporate domain knowledge. Gal and Ghahramani (2016) shows that Monte Carlo dropout can be interpreted as approximate Bayesian inference in a neural network, providing a lightweight alternative to the aforementioned Bayesian approaches; however, again, we cannot incorporate domain-specific knowledge and have minimal ability to adjust the priors on the weights. Less attention has been given to incorporating meaningful prior information while training neural networks. Osband et al. (2018) obtains samples from a prior data generating mechanism, and used a bootstrapping-like approach to model the difference between the prior sample and the data. While the mechanism for incorporating a prior is different from that proposed in this paper, it shares the fact that the "prior" is specified in the function space, rather than the parameter space.

5. Experimental evaluation

We begin with some qualitative experiments to show how our nonparametric posterior normalizing flows (NPL-NFs) behave in a low-dimensional, easily-visualized setting, before applying our method to real-world datasets. Throughout, we use $M = 10$ pseudo-samples in our NPL bootstrapped samples $F^{(b)}$ (see Alg. 2). For the NPL methods, we use 100 bootstrapped samples $F^{(b)}_{\text{eval}}$ (that were not part of the samples used during training) to estimate the posterior distribution and generate posterior samples.

5.1. Qualitative evaluation

We begin by considering sawtooth-shaped synthetic data, shown in the first row of Fig. 1. We train a NF on this data, plus jointly-learned NPL-NFs with varying concentration parameters; see App. A.2.1 for further details. Fig. 1 shows the mean posterior pdf for each flow (for standard NFs, we obtain a single estimate, which we show in place of the mean), using a standard normal base measure Ω. Fig. 2 shows the standard deviation of the posterior. Samples from this distribution, plus corresponding results using a

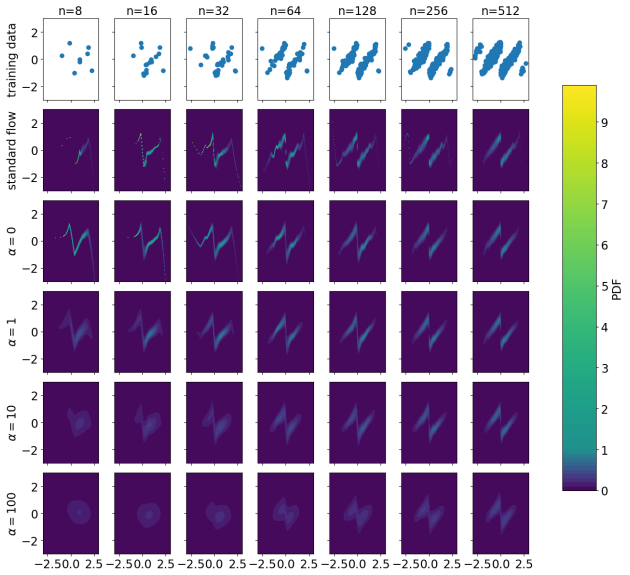

Figure 1: Mean pdfs for our model. Top row: Training samples of varying sample size. Subsequent rows: corresponding mean pdf for standard NFs, and NPL-NFs with various values of concentration parameter α and Gaussian base measure Ω. Values are truncated to 10; this only affects standard NFs with $n = 8, 16, 32$, and $\alpha = 0$ NPL-NFs with $n = 8, 16$.

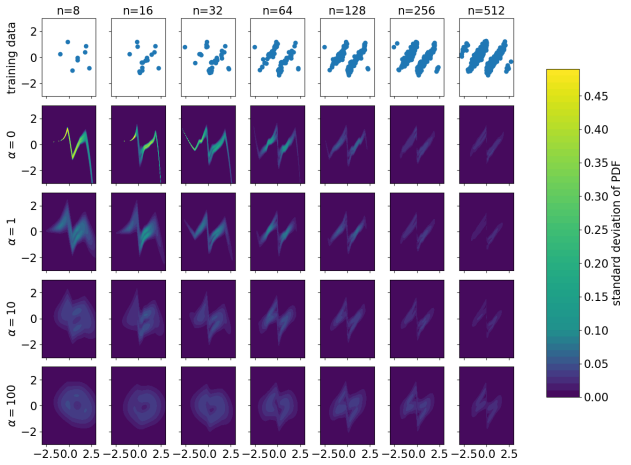

Figure 2: Standard deviation of pdfs for our model. Top row: Training samples of varying sample size. Subsequent rows: standard deviation of the posterior pdf of NPL-NFs with various values of α and Gaussian base measure Ω. Values are truncated to 0.5; this truncation only affects $\alpha = 0$ NPL-NFs with $n = 8, 16$.

Gaussian process base measure, are shown in App. A.2.2. We see that that the NPL-NFs capture uncertainty in the estimated pdf; as n increases, the uncertainty decreases. As expected, increasing α pushes our posterior closer to the base measure Ω. For larger training sizes, results are more similar across the models, since the posterior uncertainty is reduced. However, for smaller training sizes, we see that the standard NF overfits to the data.

In Fig. 3, we repeat this analysis, using the "naïve" approach where a separate NF is trained for each bootstrap sample. The computational cost here is dramatically higher; for this reason, only ten samples were used in each case, resulting in a less smooth posterior estimate. The general performance trend is similar to that seen in Fig. 1, suggesting our approximate NPL-NF method is a reasonable, cost-efficient alternative to exact NPL in normalizing flows.

5.2. Evaluation on tabular data

Next, we explore how NPL-NFs perform on real-world tabular datasets, as the size of the dataset increases. We consider five datasets from the UCI repository (Dua and Graff, 2017). We construct training sets of different sizes by selecting the first n training examples from each dataset, for $n \in \{500, 1000, 5000\}$. For each dataset, we train four models (repeating each with five random seeds): a standard

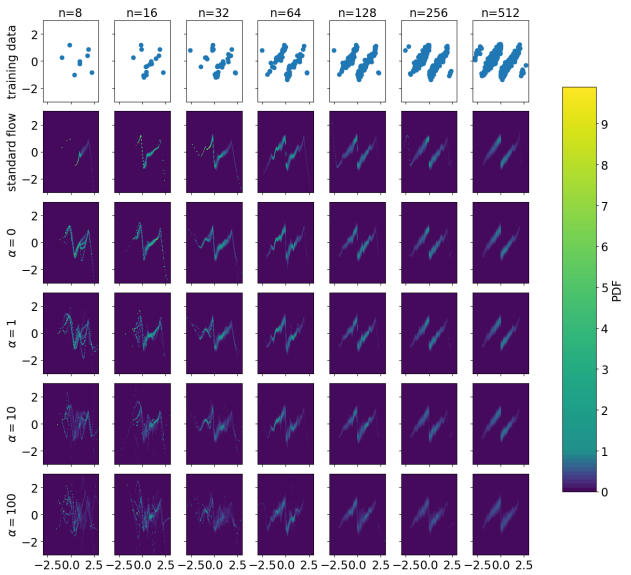

Figure 3: Mean pdfs for naïve NPL-NFs. Top row: Training samples of varying sample size. Subsequent rows: corresponding mean pdf for standard NFs, and independently trained NPL-NFs (i.e., one flow trained per bootstrap sample) with various values of α and Gaussian base measure Ω. Values are truncated to 10; this truncation only affects standard NFs with $n = 8, 16, 32$, and NPL-NFs with $n = 8, 16$ (for all values of α).

NF, and approximate NPL-NF with concentration parameter $\alpha \in \{0, 10, 100\}$. For further details, see App. A.3.1.

Fig. 4 shows how the average test set log likelihood varies as

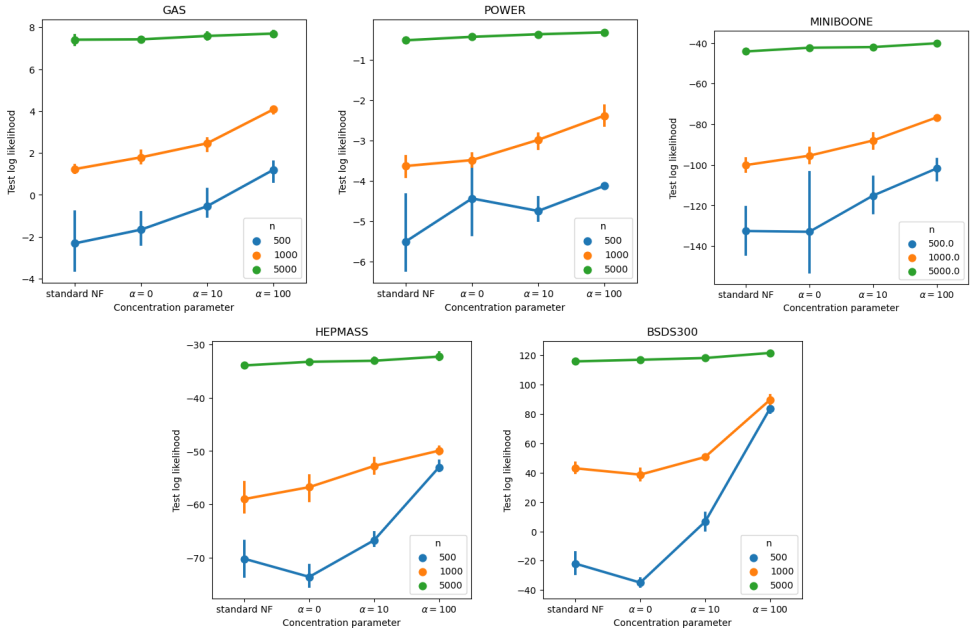

Figure 4: Test set log likelihood on five tabular datasets, using various sized training datasets.

we move from a standard NF, to an NPL flow with increasing concentration parameter α. For small datasets ($n = 500$ and $n = 1000$), we see that increasing α leads to improved test set performance, by reducing the tendency of the NF to overfit. When we consider a larger dataset ($n = 5000$), we see less of a benefit: we already have a significant amount of information from the data, and the risk of overfitting is reduced. As in Section 5.1, we see limited evidence for the NPL model with $\alpha = 0$ leading to improved performance over the standard model: while it tends to perform better when $n = 1000$, performance is mixed at $n = 500$. We hypothesise that this is because the NF is still able to overfit to the data under reweighting.

5.3. Evaluation on image data

Next, we explore the use of NPL-NFs on images of handwritten digits from the MNIST dataset[1]. We train our models on the first 1000 training examples, to simulate a scenario with high model uncertainty, and evaluate on the full test set. We train a standard NF, plus approximate NPL-NF models. For the NPL-NF models, we consider two base measures Ω: a) a mixture of 10 Gaussians with class-specific parameters, and b) an empirical distribution based on the USPS dataset, showing how we might use empirical "prior information". See App. A.4 for further details.

In Fig. 5, we see how test set bits per dimension (BPD) varies across the models. We see for low and moderate values of α, both NPL-NF methods outperform the standard NF. For the mixture of Gaussian variant, performance begins to degrade for high α. This is not surprising, as our base measure is a poor approximation for the true distribution. Conversely, the empirical base measure – presumably a more realistic model – continues to perform well at higher values of α.

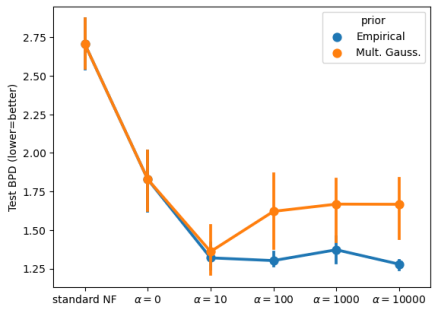

Figure 5: Test set BDP on MNIST.

6. Discussion

Nonparametric posterior learning is a new and interesting alternative to classical Bayesian inference; however it can scale poorly when applied to complex, highly parametrized models such as normalizing flows. We present an approximate inference algorithm that allows us to construct nonparametric posterior normalizing flows, and show that the resulting posterior distributions offer improved generalization, particularly when working with small datasets.

[1] http://yann.lecun.com/exdb/mnist/

References

Baldi, P. and Sadowski, P. J. (2013). Understanding dropout. In *Advances in Neural Information Processing Systems*.

Bissiri, P. G., Holmes, C. C., and Walker, S. G. (2016). A general framework for updating belief distributions. *Journal of the Royal Statistical Society, Series b, Statistical Methodology*, 78(5):1103.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. In *International Conference on Machine Learning*.

Daxberger, E., Kristiadi, A., Immer, A., Eschenhagen, R., Bauer, M., and Hennig, P. (2021). Laplace redux-effortless Bayesian deep learning. *Advances in Neural Information Processing Systems*.

Dinh, L., Krueger, D., and Bengio, Y. (2014). NICE: Nonlinear independent components estimation. *arXiv preprint arXiv:1410.8516*.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using real NVP. In *International Conference on Learning Representations*.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. (2019). Neural spline flows. In *Advances in Neural Information Processing Systems*.

Fong, E., Holmes, C., and Walker, S. G. (2021). Martingale posterior distributions. *arXiv preprint arXiv:2103.15671*.

Fong, E., Lyddon, S., and Holmes, C. (2019). Scalable nonparametric sampling from multimodal posteriors with the posterior bootstrap. In *International Conference on Machine Learning*.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*.

Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. (2018). Gpytorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems*.

Graves, A. (2011). Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*.

Hernández-Lobato, J. M. and Adams, R. (2015). Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *International Conference on Machine Learning*.

Hull, J. J. (1994). A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554.

Lyddon, S., Walker, S., and Holmes, C. C. (2018). Nonparametric learning from Bayesian models with randomized objective functions. In *Advances in Neural Information Processing Systems*.

MacKay, D. J. (1992). Bayesian interpolation. *Neural Computation*, 4(3):415–447.

Neal, R. M. (2012). *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

Osband, I., Aslanides, J., and Cassirer, A. (2018). Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems*.

Papamakarios, G., Pavlakou, T., and Murray, I. (2017). Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*.

Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International Conference on Machine Learning*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Tabak, E. G. and Vanden-Eijnden, E. (2010). Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233.

Trippe, B. L. and Turner, R. E. (2018). Conditional density estimation with Bayesian normalising flows. *arXiv preprint arXiv:1802.04908*.

A. Appendix

A.1. Summary of algorithms used in the main paper

We begin by sharing algorithms for NPL in a general context. Alg. 1 shows how one might, in theory, perform nonparametric posterior inference an arbitrary distribution $\pi_{\mathcal{F}}$ with countable support.

Algorithm 1 Exact sampling from a nonparametric posterior (Lyddon et al., 2018; Fong et al., 2019)

Require: Observations $x_{1:n}$, distribution $\pi_{\mathcal{F}}$, desired number of samples B.
  for $b = 1, \ldots, B$ **do**
    Sample $F^{(b)} = \sum_{i=1}^{\infty} w_i \delta_{\psi_i} \sim \pi_{\mathcal{F}}(F|x_{1:n})$.
    $\theta^{(b)} = \arg\min_\theta \sum_{i=1}^{\infty} w_i \ell_\theta(\phi_i)$.
  end for
  return $\{\theta^{(b)}\}_{b=1}^{B}$.

In practice, if $\pi_{\mathcal{F}}$ has countably infinite support we cannot sample from the full conditional distribution. Instead we must resort to approximations. Alg. 2 shows how we can obtain approximate samples from the NPL posterior, where $\pi_{\mathcal{F}}$ is a Dirichlet process, approximated using a weak limit approximation.

Algorithm 2 Approximate NPL, using a Dirichlet process data generating mechanism (Fong et al., 2019)

Require: Observations $x_{1:n}$, concentration parameter α, base measure Ω, desired number of samples B, approximation level $K \geq n$.
  $m = K - n$.
  for $b = 1, \ldots, B$ **do**
    Sample m pseudo-observations $\widetilde{x}_j \sim \Omega$
    Sample weights
      $(w_{1:n}, \widetilde{w}_{1:m}) \sim \text{Dir}\left(1, \ldots, 1, \frac{\alpha}{m}, \ldots, \frac{\alpha}{m}\right)$.
    $F^{(b)} = \sum_{i=1}^{n} w_i \delta_{x_i} + \sum_{j=1}^{m} \widetilde{w}_j \delta_{\widetilde{x}_j}$
    $\theta^{(b)} = \arg\min_\theta \sum_{i=1}^{n} w_i \ell_\theta(x_i) + \sum_{j=1}^{m} \widetilde{w}_j \ell_\theta(\widetilde{x}_j)$.
  end for
  return $\{\theta^{(b)}\}_{b=1}^{B}$.

The naïve algorithm described in Sec. 3.1 is a direct application of Alg. 2, where θ are the parameters of the normalizing flow.

Finally, Alg. 3 describes our method for jointly learning multiple flows, as described in Sec. 3.2.

Algorithm 3 Sampling from an approximate NPL normalizing flow

Require: Observations $x_{1:n}$, prior distribution $\pi_{\mathcal{F}}$, required number of samples B, approximation level $K \geq n$, initial flow parameters ϕ, learning rate τ.
  $m = K - n$
  while Not converged **do**
    Sample $F^{(b)} = \sum_i w_i \delta_{\psi_i} \sim \pi_{\mathcal{F}}(F|x_{1:n})$ (or an approximation thereof, see Alg. 2).
    $\mu = \sum_i w_i g_\phi^{-1}(\psi_i)$.
    $\phi \leftarrow \phi + \tau \nabla_\phi \sum_i w_i \ell_{\mu,\phi}(\psi_i)$.
  end while
  $\hat{\phi} \leftarrow \phi$.
  for $b = 1, \ldots, B$ **do**
    Sample $F^{(b)} = \sum_i w_i \delta_{\psi_i} \sim \pi_{\mathcal{F}}(F|x_{1:n})$ (or an approximation thereof).
    $\mu^{(b)} = \sum_i w_i g_{\hat{\phi}}^{-1}(\psi_i)$.
  end for
  return $\{\mu^{(b)}\}_{b=1}^{B}, \hat{\phi}$.

Given a set of latent means $\{\mu^{(b)}\}_{b=1}^{B}$ and the shared flow parameters ϕ (obtained via Alg. 3) we can sample from the posterior predictive distribution by sampling

$$\begin{aligned} b \sim & \text{Uniform}\{1, \dots, B\} \\ z^* | b \sim & N(\mu^{(b)}, \mathbb{I}) \\ x^* = & g_\phi(z^*). \end{aligned}$$

A.2. Experimental details and additional experimental results for the qualitative evaluation

A.2.1. Synthetic data details

We generated sawtooth-shaped synthetic data generated according to

$$\begin{aligned} x_i \sim & N(0, 1) \\ \epsilon_i \sim & N(0, 0.2^2) \\ y_i = & \frac{3}{\pi}\left(x_i \mod \frac{2\pi}{3}\right) + \epsilon_i - 1. \end{aligned}$$

We model this using a Real NVP flow (Dinh et al., 2016). We use four coupling layers, with the first dimension masked at each layer (i.e., the flow only changes the distribution of y, not x). For each layer, the scale and location were modeled using a two-layer FFNN, with 16-dimensional hidden layers. We trained each flow for 5000 epochs, with a learning rate of 10^{-3}.

For our NPL version of Real NVP, we explored two choices of base measure Ω. First, we used a multivariate Gaussian, with mean and covariance given by the mean and covariance of the training data. Second, we used a two-stage Gaussian process model, where x is modeled using a moment-matched Gaussian, and $y|x$ is modeled using a Gaussian process with squared exponential kernel, trained on the training data. The Gaussian processes were trained and sampled using the `gpytorch` package (Gardner et al., 2018).For the NPL flows, we explored $\alpha \in \{0, 1, 10, 100\}$. Note that NPL with $\alpha = 0$ does not make use of the base measure Ω.

A.2.2. Results

In Figures 1, 2 and 6, we show the mean and standard deviation of the posterior pdfs obtained using a Gaussian base measure, plus samples from the posterior predictive. In Figures 7, 8 and 9, we show analogous results using a Gaussian process base measure. As we see from these figures, behavior is consistent with our expectations. We see that the posterior flows capture uncertainty in the estimated pdf; as n increases, the uncertainty decreases. For larger training sizes, results are similar across the models, since the posterior uncertainty is reduced. However, for smaller training sizes, we see that the standard normalizing flow overfits to the data. The "Bayesian bootstrap" version of our algorithm, with $\alpha = 0$, captures some uncertainty, but still overfits. We note that other works on NPL have found this setting to work well (Fong et al., 2019; 2021); however, these papers focus on simpler models. We hypothesise that in our case, the $\alpha = 0$ version does not perform well due to the increased model capacity: the flows have the capacity to almost perfectly replicate the empirical pdf, and therefore simply reweighting the data does little to change the distribution far from the observations.

Conversely, as we increase α—i.e., increase the weight assigned to the Gaussian base measure Ω—we see our inferred pdfs and samples become closer to this base measure. The figures match reasonable intuitions of how a posterior should extrapolate between "prior" and data.

We see that the posteriors obtained using a Gaussian process base measure (Figures 7, 8 and 9), with a more informative base measure, we have more plausible estimates with lower values of n, compared with the standard Gaussian base measure (Figures 1, 2 and 6). This is an example of how we can make use of existing domain information, such as the output of a weaker model trained on the data.

Fig. 10 shows samples from the "naive" NPL normalizing flow, corresponding to the mean pdf shown in Fig. 3.

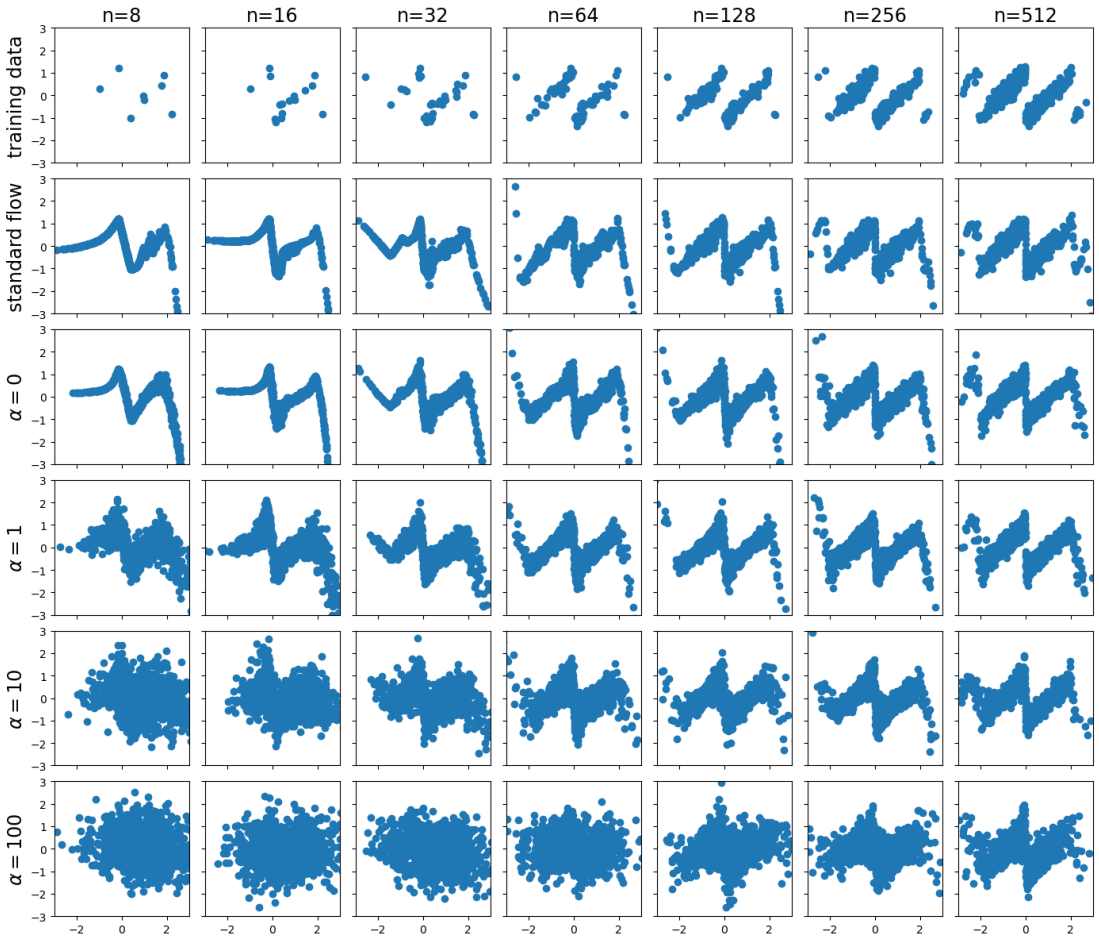

Figure 6: Samples from our model with Gaussian base measure. Top row: Training samples from sawtooth data, for varying sample size. Subsequent rows: samples from standard normalizing flows, and NPL normalizing flows with various values of α and Gaussian base measure Ω.

A.3. Additional experimental details for the experiments on tabular data

A.3.1. Implementation details for tabular datasets

For all experiments, we used a neural spline flow architecture (Durkan et al., 2019) with 10 autoregressive rational-quadratic splines, each with one hidden layer with 64 units. We used 8 bins. We chose these parameters to be between the parameter values used in Durkan et al. (2019) on the MINIBOONE and HEPMASS datasets, since these were the two smallest datasets considered in that paper. For all experiments, we used dropout with dropout probability 0.5, and trained for 10,000 epochs with a learning rate of 5×10^{-4}. For the NPL models with $\alpha > 0$, we used a spherical Gaussian as our base measure Ω, noting that the datasets have been pre-processed to have zero mean and unit standard deviation along each dimension. All experiments were repeated using 5 different random seeds.

Table 1 shows the number of features, and the test set sizes, of the datasets used in Sec. 5.2.

	GAS	POWER	HEPMASS	MINIBOONE	BSDS300
# features	6	8	21	43	63
Test set size	105,206	204,928	174,987	3,648	250,000

Table 1: Properties of datasets used in Sec. 5.2.

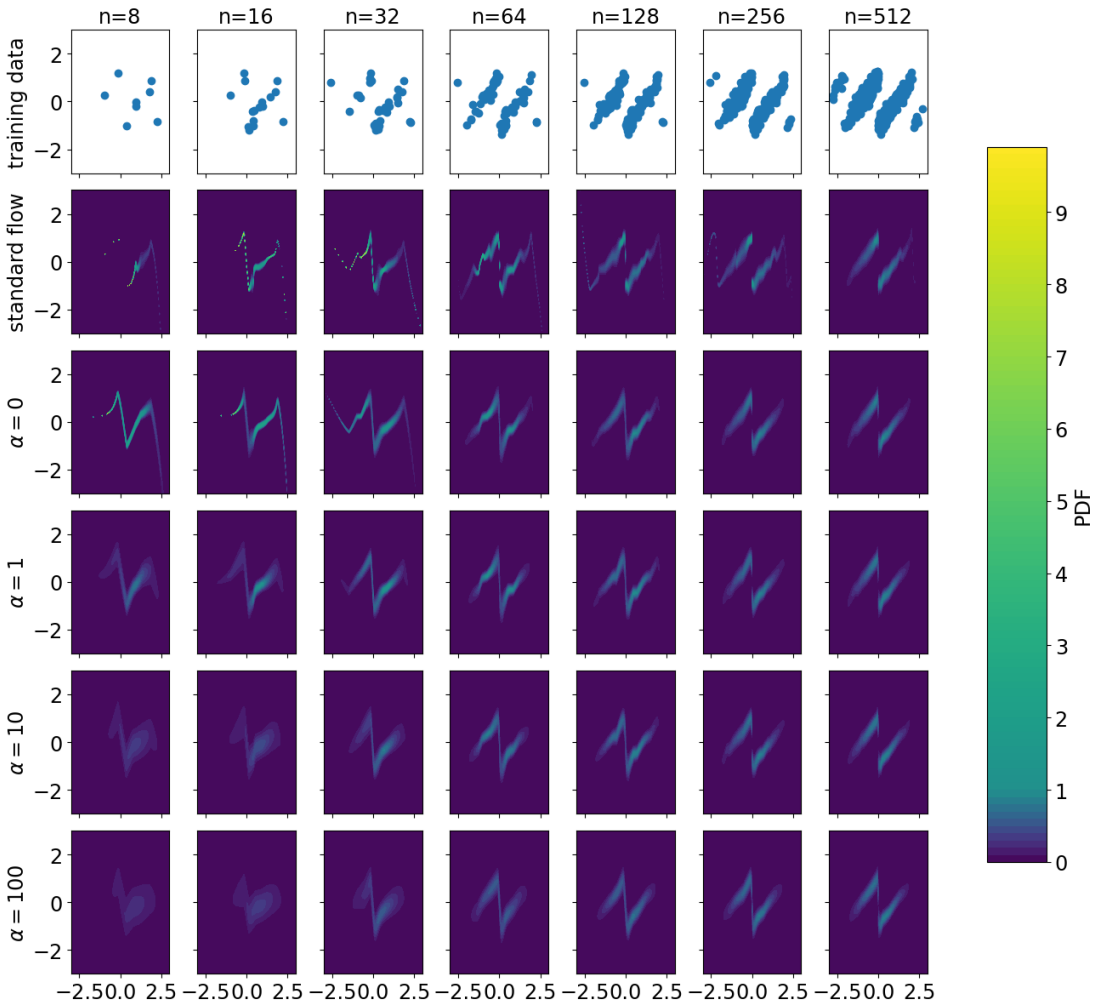

Figure 7: Mean pdf for our model with Gaussian process base measure. Top row: Training samples from sawtooth data, for varying sample size. Subsequent rows: corresponding mean pdf for standard normalizing flows, and NPL normalizing flows with various values of α and Gaussian process base measure Ω. Values of the mean pdf are truncated to 10; this truncation only affects standard NFs with $n = 8, 16, 32$, and $\alpha = 0$ NPL-NFs with $n = 8, 16$.

These datasets were all used by Papamakarios et al. (2017) and Durkan et al. (2019), and we used the splits and preprocessing from those papers[2].

A.3.2. Comparison with Monte Carlo dropout

In Sec. 3.2, we optimized the neural network parameters ϕ while performing nonparametric posterior inference on latent distribution parameter μ. While this is much more computationally efficient than a naïve NPL approach where B models are learned independently, it will underestimate uncertainty due to optimizing ϕ.

Fong et al. (2019) uses multiple, random restarts to avoid getting stuck in local optima. We could similarly learn multiple estimates of ϕ following random reinitializations of our flow, although this would increase the computational cost. An alternative that we explore is to use dropout (Srivastava et al., 2014). Under dropout, nodes in the neural network are randomly excluded from the overall function during training. This can be loosely interpreted as constructing an ensemble of neural networks (Baldi and Sadowski, 2013). If we use dropout during evaluation—a practice known as Monte Carlo dropout (Gal and Ghahramani, 2016)—we can think of this as selecting a single element from the ensemble.

[2] https://zenodo.org/record/1161203#.Y0hy5ezML6Y

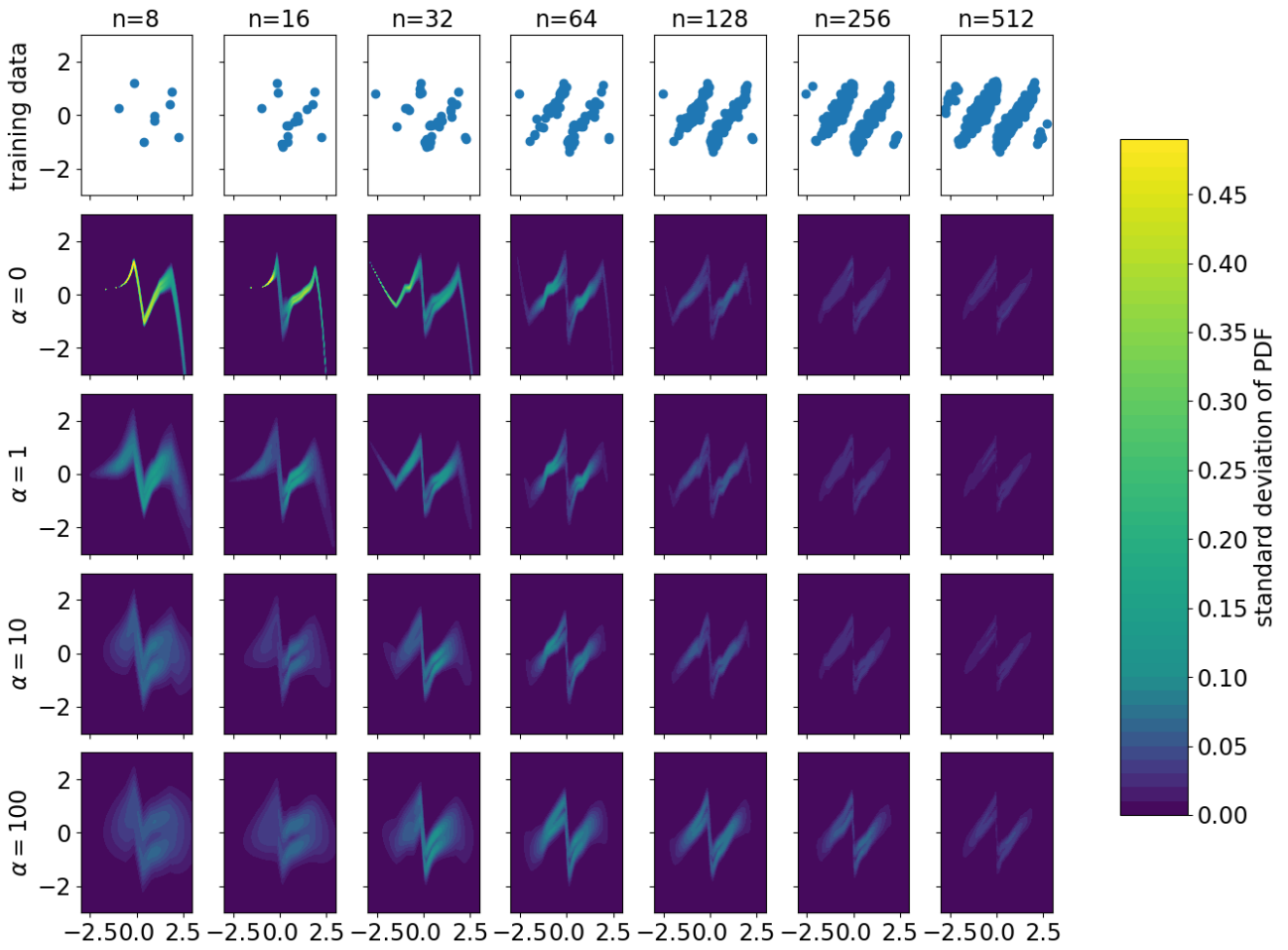

Figure 8: Standard deviation of pdfs for our model with Gaussian process base measure. Top row: Training samples from sawtooth data, for varying sample size. Subsequent rows: standard deviation of the posterior pdf of NPL normalizing flows with various values of α and Gaussian process base measure Ω. Values of the standard deviation are truncated to 0.5; this truncation only affects $\alpha = 0$ NPL-NFs with $n = 8, 16$.

We note that this does *not* directly correspond to exact nonparametric learning. The "dropped out" $\phi^{(b)}$ almost certainly has not appeared during training, and so will not correspond to the maximum likelihood parameters for ϕ conditioned on the selected pattern of zeros and the bootstrap sample $F^{(b)}$.

In Table 2, we show the numeric results from Fig. 4 alongside the results obtained using Monte Carlo dropout. For the standard normalizing flow (NF + MC dropout), we generated 100 MC dropout samples. For the NPL normalizing flows (NPL-NF + MC dropout), we used separate MC dropout samples for each of the 100 bootstrapped samples.

We see that, in each case, the results with Monte Carlo dropout appear similar to those without Monte Carlo dropout. This indicates two things. First, that the ability of the nonparametric posterior to incorporate meaningful prior knowledge leads to better generalization performance than approximate Bayesian inference via Monte Carlo dropout (NF + MC dropout). Second, there does not seem to be an advantage to using Monte Carlo dropout in conjunction with our nonparametric learning approach, versus learning a single global $\hat{\phi}$.

A.4. Additional experimental details for the experiments on image data

For our NF architecture, we used a modified RealNVP flow (Dinh et al., 2016) with variational dequantization on the input. Our architecture follows the example given in `https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/tutorial11/NF_image_modeling.html`.

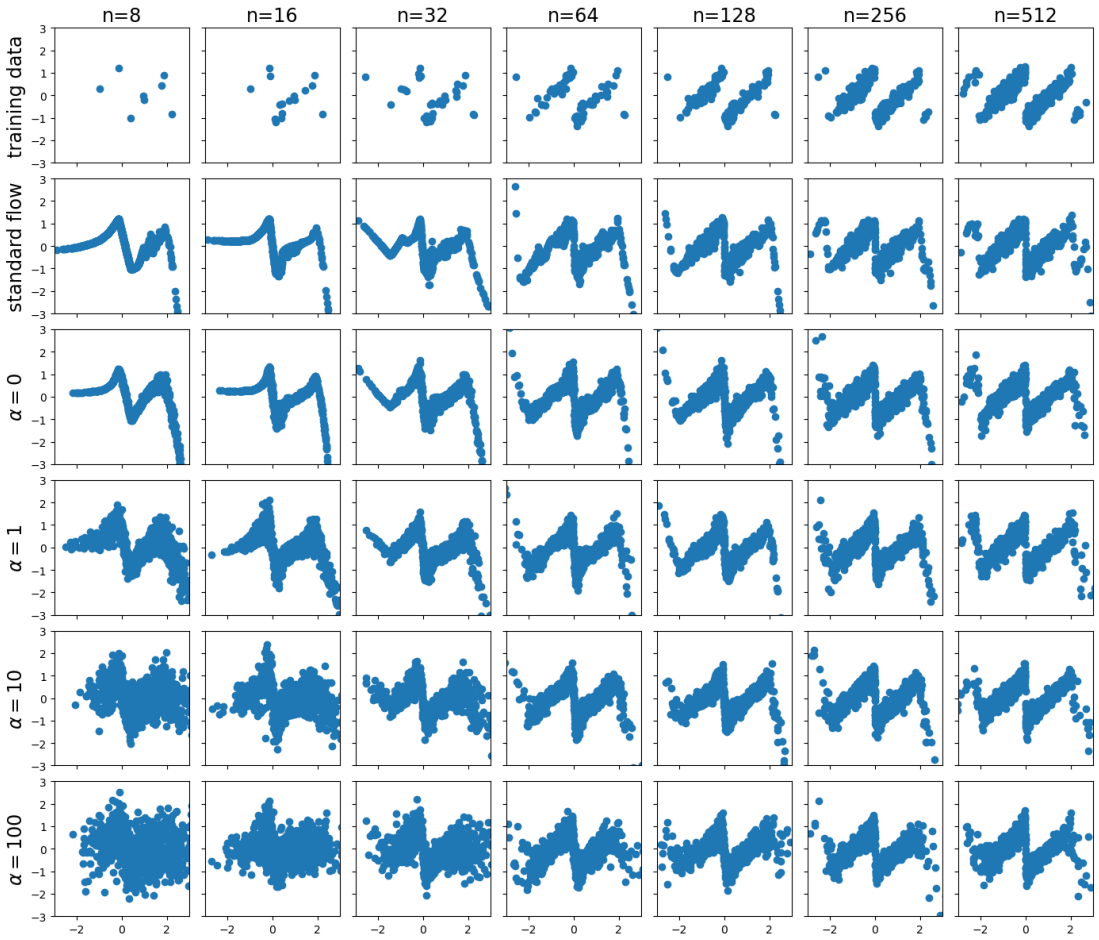

Figure 9: Samples from our model with Gaussian process base measure. Top row: Training samples from sawtooth data, for varying sample size. Subsequent rows: samples from standard normalizing flows, and NPL normalizing flows with various values of α and Gaussian process base measure Ω.

For our base measures Ω, we considered two choices. The first is a mixture of ten Gaussians, with uniform probabilities and with mean and covariance obtained empirically from the corresponding digit in the training data. The second is an empirical dataset, generated from the USPS image dataset (Hull, 1994), scaled to be the same dimension as the MNIST images.

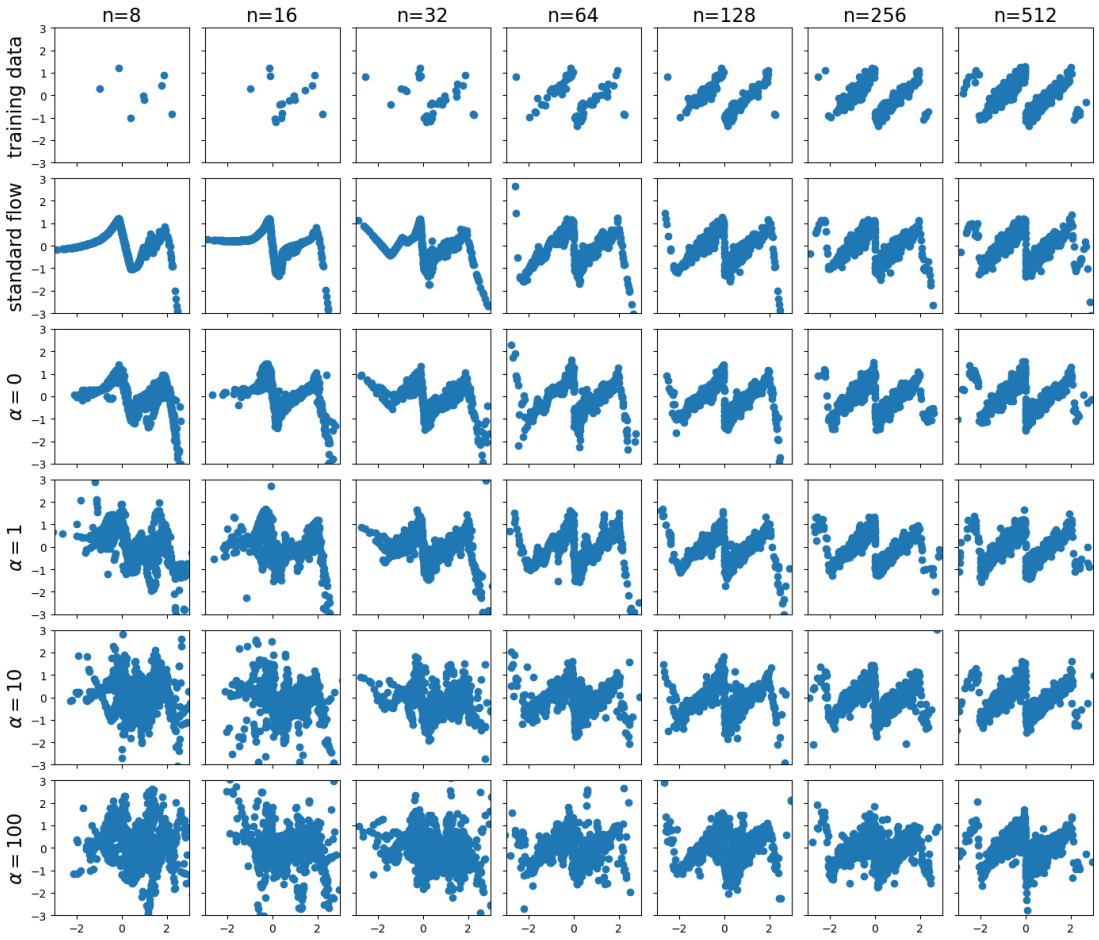

Figure 10: Samples from naïve NPL-NFs with Gaussian base measure. Top row: Training samples from sawtooth data, for varying sample size. Subsequent rows: samples from standard normalizing flows, and "naïve" NPL normalizing flows with various values of α and Gaussian base measure Ω.

	GAS	POWER	MINIBOONE	HEPMASS	BSDS300
NF	-2.31(0.88)	-5.50(0.59)	-132.58(7.26)	-70.25(2.09)	-22.00(4.62)
NF + MC dropout	-2.58(0.91)	-5.66(0.62)	-132.93(7.29)	-70.72(2.08)	-22.21(4.62)
NPL-NF ($\alpha = 0$)	-1.66(0.42)	-4.44(0.55)	-132.96(14.88)	-73.65(1.23)	-34.94(1.71)
NPL-NF ($\alpha = 0$) + MC dropout	-1.90(0.42)	**-4.10(0.47)**	-133.38(14.97)	-74.18(1.26)	-35.23(1.73)
NPL-NF ($\alpha = 0$)	-0.54(0.38)	-4.75(0.17)	-115.15(5.60)	-66.76(0.78)	6.57(3.63)
NPL-NF ($\alpha = 10$) + MC dropout	-0.78(0.39)	-5.05(0.05)	-115.51(5.63)	-67.27(0.79)	6.30(3.62)
NPL-NF ($\alpha = 100$)	**1.20(0.29)**	-4.12(0.02)	**-101.77(3.25)**	**-53.06(0.61)**	**83.71(1.50)**
NPL-NF ($\alpha = 100$) + MC dropout	1.01(0.31)	-4.24(0.02)	-102.51(3.27)	-53.33(0.62)	83.36(1.50)

(a) $n = 500$

	GAS	POWER	MINIBOONE	HEPMASS	BSDS300
NF	1.23(0.10)	-3.63(0.16)	-100.12(2.02)	-59.03(1.75)	42.97(2.19)
NF + MC dropout	0.94(0.11)	-3.81(0.16)	-100.65(2.04)	-59.62(1.78)	42.71(2.20)
NPL-NF ($\alpha = 0$)	1.79(0.17)	-3.49(0.10)	-95.48(2.20)	-56.79(1.43)	38.67(2.35)
NPL-NF ($\alpha = 0$) + MC dropout	1.51(0.18)	-3.65(0.10)	-95.97(2.22)	-57.22(1.44)	38.29(2.35)
NPL-NF ($\alpha = 0$)	2.46(0.17)	-2.99(0.12)	-88.02(2.24)	-52.81(0.85)	50.68(1.11)
NPL-NF ($\alpha = 10$) + MC dropout	2.20(0.17)	-3.12(0.12)	-88.45(2.26)	-53.20(0.87)	50.35(1.12)
NPL-NF ($\alpha = 100$)	**4.08(0.08)**	**-2.39(0.15)**	**-76.66(0.49)**	**-49.93(0.39)**	**89.61(1.75)**
NPL-NF ($\alpha = 100$) + MC dropout	3.84(0.09)	-2.50(0.16)	-77.02(0.49)	-50.19(0.40)	89.34(1.75)

(b) $n = 1000$

	GAS	POWER	MINIBOONE	HEPMASS	BSDS300
NF	7.41(0.13)	-0.52(0.03)	-44.17(0.31)	-33.92(0.24)	115.87(0.50)
NF + MC dropout	7.14(0.12)	-0.61(0.03)	-44.40(0.31)	-34.16(0.24)	115.71(0.50)
NPL-NF ($\alpha = 0$)	7.42(0.07)	-0.43(0.02)	-42.33(0.51)	-33.23(0.12)	116.97(0.83)
NPL-NF ($\alpha = 0$) + MC dropout	7.18(0.06)	-0.51(0.02)	-42.54(0.48)	-33.44(0.12)	116.74(0.83)
NPL-NF ($\alpha = 0$)	7.59(0.10)	-0.37(0.01)	-41.99(0.15)	-33.04(0.31)	118.18(0.71)
NPL-NF ($\alpha = 10$) + MC dropout	7.32(0.11)	-0.44(0.02)	-42.18(0.16)	-33.24(0.32)	117.95(0.72)
NPL-NF ($\alpha = 100$)	**7.70(0.06)**	**-0.32(0.01)**	**-40.14(0.31)**	**-32.26(0.41)**	**121.62(0.15)**
NPL-NF ($\alpha = 100$) + MC dropout	7.44(0.06)	-0.39(0.01)	-40.29(0.31)	-32.45(0.42)	121.43(0.15)

(c) $n = 5000$

Table 2: Test set log likelihoods, based on size-n training sets from five UCI datasets. Standard errors shown in parentheses.