Conditional Density Estimation with Bayesian Normalizing Flows

Brian L. Trippe
University of Cambridge and
Massachusetts Institute of Technology
btrippe@mit.edu

Richard E. Turner
University of Cambridge
ret26@cam.ac.uk

Abstract

Modeling complex conditional distributions is useful in a variety of settings. However, conditional density estimation (CDE) is challenging, in large part due to fundamental trade-offs between modeling distributional complexity, functional complexity and avoiding overfitting. We present a Bayesian method for CDE which uses normalizing flows to model complex predictive distributions and variational Bayesian neural networks to capture their changes throughout the space without overfitting. We validate our method on several small benchmark regression datasets on some of which we obtain state of the art performance. We demonstrate the scalability of this approach on a spatial density modeling task with over 1 million datapoints using the New York City yellow taxi dataset. On this task, our model provides constant time evaluation of fine-grained conditional probability densities.

1 Introduction

Modeling complex noise distributions is useful in a variety of settings. For example, in reinforcement learning, we often want to model action value functions for risky decisions for which rewards are inherently bimodal, or state transition dynamics which may vary significantly from Gaussian [4]. In financial modeling, properly handling heavy tailed distributions may be crucial. Additionally, in spatial density modeling, we find extremely non-Gaussian distributions over locations of people and events in space.

Let $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ be a dataset of observations $(x_i \in X, y_i \in Y)$ sampled i.i.d. from some joint distribution $p(x, y)$. Conditional density estimation (CDE) refers to the problem of approximating the conditional, $p(y|x, \mathcal{D})$. In particular, parametric methods for CDE propose a class of densities to describe this conditional, $\{p, \omega \in \Omega\}$, and a class of functions, h, indexed by $\theta \in \Theta$, and use $\mathcal{D}$ to find a function, $h_\theta : X \to \Omega, x_i \mapsto \omega_i$, which is used to approximate $p(y_i|x_i)$ as $p(y_i|\omega_i = h_\theta(x_i))$.

Methods for CDE are defined entirely by choices of Ω, Θ and an inference procedure, which prescribes an objective and a learning algorithm which dictate how to choose θ. For example, linear regression models define a linear mapping from x and θ to ω, where ω defines an exponential family likelihood. Ordinary least squares additionally specifies that $p(y|\omega)$ is Gaussian, and defines a maximum likelihood objective with an analytic form to find θ. Neural network classifiers use θ to define a neural network and ω to be the softmax-transformed output defining a categorical distribution over labels, such that $p(y|\omega = h_\theta(x)) = \mathrm{Cat}(y|\omega)$.

While much work on CDE has been done, it remains a challenging problem in its general formulation, particularly when one wants to maintain the ability to model complex distributions without overfitting to small datasets. This motivates our development of a novel Bayesian method for CDE which obviates the limitations of previous methods. In this work, we propose to use a normalising flow to define conditional densities, $p(y_i|\omega_i = h_\theta(x_i))$, where the parameters, ω_i are the output of a neural network h_θ. To avoid overfitting, we perform variational inference(VI) over the parameters of the neural network, θ.

Second workshop on Bayesian Deep Learning (NIPS 2017), Long Beach, CA, USA.

The organization of this paper is as follows. In section 2 we present a method for applying normalizing flows to conditional density estimation [22]. Next, in section 3, we apply our method to several small benchmark regression tasks on which we achieve state of the art performance. We then demonstrate the scalability of our method in section 4 by applying it to a spatial density estimation task using the New York City yellow taxi dataset.

2 Bayesian Normalizing Flows for Conditional Density Estimation

A normalizing flow is a mapping between probability densities defined by a differentiable, monotonic bijection between the spaces in which they live [20]. These transformations are composable, and a series of relatively simple invertible transformations can be used define more complex transformations. [20] introduced two families of parametric transformations, and showed that a series of these transformations could warp a standard Gaussian base distribution into rich approximate posteriors. In particular, by mapping random variable z_0 through a K-stage normalizing flow, $f = (f_1, f_2, \ldots, f_K)$ we define a transformed variable, z_k and its density as:

$$z_K = f_K \circ \cdots \circ f_2 \circ f_1(z_0) \quad \text{and} \qquad \ln p(z_K) = \ln p(z_0) - \sum_{k=1}^{K} \ln \frac{df_k(z_{k-1})}{dz_{k-1}} \tag{1}$$

Where $p(z_0)$ is defined to be a standard normal base distribution and each f_k is a simple transformation which has a closed form with an easy to calculate derivative. In particular, we will consider 'radial flows' [20].

As this formulation of normalizing flows was developed for Monte Carlo variational inference, it is optimized for efficiently drawing samples and calculating likelihoods and is not particularly suitable to conditional density estimation. First, the utility of this class of transformations to conditional density estimation is limited by the inefficiency of evaluating the likelihood of externally provided data, which requires inverting each f_k [19]. We overcome this limitation by reversing the direction of the normalizing flows, defining the forward mapping from the target, y to the base distribution:

$$y_0 = f_1 \circ f_2 \ldots f_K(y) \quad \text{and} \qquad \ln p(y) = \ln p(y_0) + \sum_{k=1}^{K} \ln \frac{df_k(y_k)}{dy_k} \tag{2}$$

Where $f_K : y \mapsto y_{k-1}$ and each other $f_k : y_k \mapsto y_{k-1}$, and $p(y_0)$ is defined to be a standard normal base distribution over y_0. In the conditional setting, the parameters defining each f_k is an output of $h_\theta(x_i)$, thereby yielding a different conditional distribution for each x_i. This inversion of the direction of parameterization is necessary for the application of a large class of normalizing flows to density estimation as it allows the density of data to be evaluated trivially.

A second challenge as we move from the application of normalizing flows to amortized inference to CDE is the ease of overfitting. When fitting approximate posteriors with these methods to minimize a variational objective, an implicit entropy penalty favors broad distributions. In contrast, naive approaches to CDE such as maximum likelihood estimation are prone to overfitting given flexible models[2].

The Bayesian framework provides a compelling approach for avoiding overfitting by modeling parameter uncertainty. However, the effective use of Bayesian methods requires reasonable priors, and it is not immediately obvious how to define parameters over normalizing flows. In this vein, we developed a alternative parameterization of radial flows [20] for which we found it easier to express priors over the characteristics of distributions, such as the extent of multimodality and the even-ness of tails(Appendix A, Figure 3).

Crucially, this intuitive parameterization additionally allows us to begin reasoning about placing priors over how conditional distributions change as a function of x. One appealing way to accomplish this is by placing a Gaussian process prior over h, the function defining the parameters of normalizing flows across the space and choosing a covariance function which reflects our prior beliefs(e.g. about the extent of heteroscedasticity)(Figure 4, Appendix B).

Exact posterior inference with a Gaussian process prior is far from tractable in such a model, however. As such we turn to two approximations which allow us to take this intuition and build an expressive and tractable class of estimators. First we appeal to the equivalence established by [17] between infinitely wide Bayesian neural networks and Gaussian processes, and approximate an infinite network with a finite one. Second, because exact posterior inference in even finite networks is intractable, we resort to variational inference[9] in this Gaussian process approximation. Specifically, we use mean

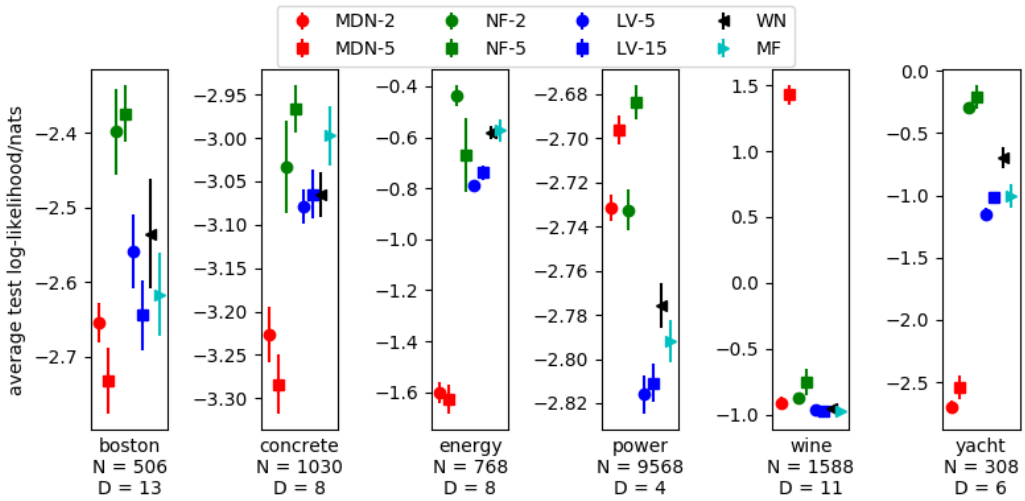

Figure 1: Comparison of performance of flexible Bayesian methods for conditional density estimation on six small UCI datasets across 20 train/test splits (Mean$\pm$1SEM). Higher is better.

field variational inference with fixed posterior variances over weights [25, 11]. Priors over weights and biases implicitly define the covariance function of the approximate Gaussian process and are kept as hyperparameters of the model, to be chosen in a problem dependent manner based upon either prior knowledge or a model selection scheme such as Bayesian optimization or cross-validation. In this way, we have turned prior beliefs about the extent of non-Gaussianity and heteroscedasticity into hyperparameters of our model. Figure 4 demonstrates this capacity by visualizing a conditional density estimator sampled with different prior parameters.

2.1 Related Work

A number of methods exist for performing CDE in its full generality which, in the limit of very large models and datasets, are able to capture arbitrary conditional distributions. These include mixtures models[8, 1], conditional variants of implicit generative models [21, 15] and autoregressive models [6, 16, 24]. Additionally, two Bayesian methods for CDE have been proposed recently; [18] performs stochastic variational inference in a mixture density network for an application to approximately Bayesian computation and [4, 5] use stochastic inputs to a Bayesian neural network for an application to reinforcement learning.

3 Results on Benchmark Datasets

We evaluated our normalizing flow model (NF) on six benchmark UCI regression datasets, comparing against two alternative Bayesian methods for CDE which can approximate arbitrarily complex conditional distributions in the limit of large models; mixture density networks (MDN) and neural networks with latent variable inputs (LV). We use Bayesian implementations of mixture density networks and neural networks with input noise following [18] and [4], respectively (details in Appendix C). For each of these three models, we tested two levels of complexity of the predictive distributions (i.e. number of warpings, mixing components and noise samples, respectively), which we judged to be roughly equivalent. We additionally compare to two variational Bayesian neural networks models with homoscedastic Gaussian likelihoods. These are two different mean-field approximations, one with learned variances (MF) and one with fixed variances (WN)[1]. Hyperparameters of all methods were optimized on held out validation sets using Bayesian optimization. Overall we see the best performance by the more expressive normalizing model, NF-5 (Figure 1). The normalizing flow based models outperform the Bayesian neural network models with Gaussian likelihoods on every dataset except for energy, on which performance is similar. Additionally, they yield state of the art

[1]We refer to this model as weight noise (WN) due to its equivalence to this model. This model has been referred to as fast dropout [25] and Gaussian dropout[11].

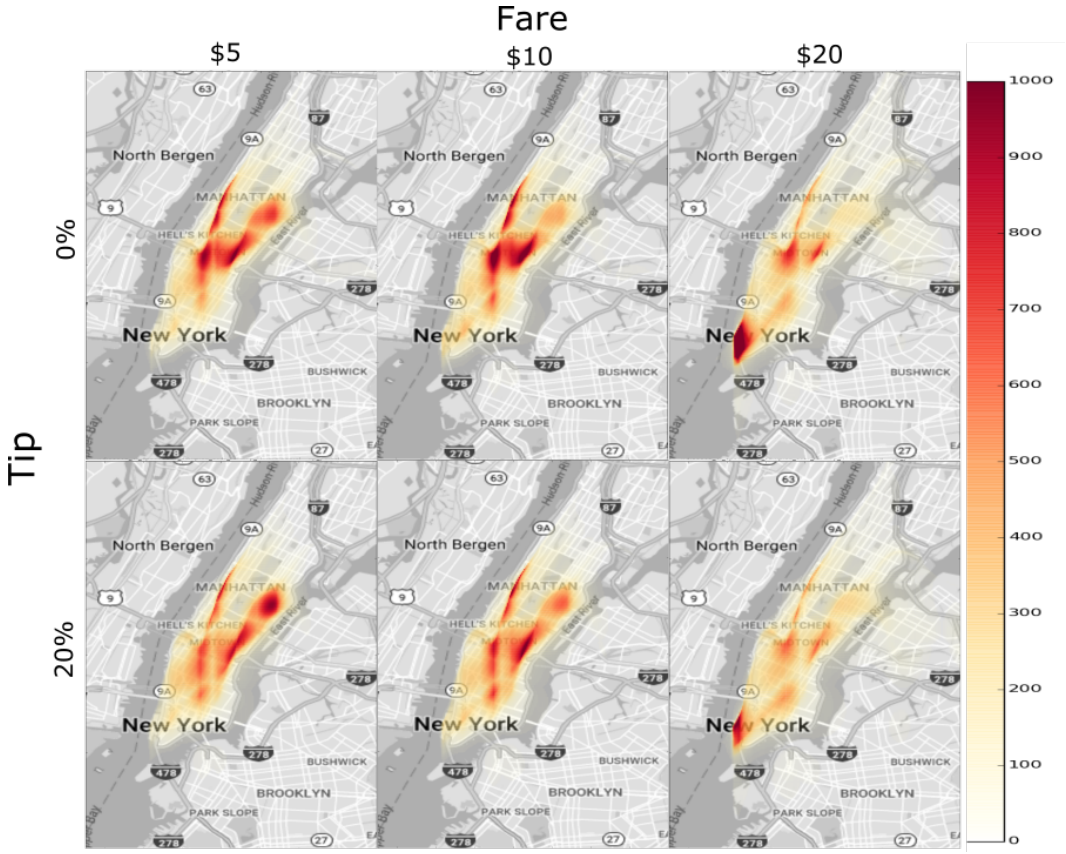

Figure 2: Heatmaps representing learned conditional densities of yellow cab pickups for different fares and tip amounts. Rides with no tip are most frequent in touristy areas and around Wall Street. Heat density is in units of $\frac{p(\text{lat},\text{long})}{d\text{lat}\ d\text{long}}$ and is capped at 1000. Best viewed in color.

performance on two of these datasets, 'yacht' and 'wine' [7, 3, 13, 14]. We note that this state of the art performance is with a neural network model consisting of a single hidden layer of 50 hidden units and expect that wider and deeper models will provide even further improvements.

MDNs perform poorly on most datasets as compared to the other methods. The exception is the Wine quality prediction dataset, on which MDN-5 had far superior performance (Figure 1). Upon closer inspection into these results, we found that the labels for this benchmark regression task are ordinal ratings on a 1 to 10 scale, and the MDN was able to fit a Gaussian with very small variance to one of these ratings. As a result, we conclude that performance on this dataset is a weak indicator of performance on continuous distributions.

4 Conditional Spatial Density Estimation on NYC Yellow Taxi Dataset

Spatial density estimation is the problem of predicting distributions over where events occur in space. To demonstrate the scalability of our method we applied it to a spatial density modeling task using a subset of the NYC yellow taxi dataset consisting of records from more than 1 million trips[2]. Specifically we performed mean field VI in model predicting a distribution over the pickup locations based on the fare, percent tip, and time of day. Applying our method to this task required two changes from the simpler instance described in the previous section. First, in order to model a 2D distribution over location, we used an autoregressive structure to model the conditional as a product of two 1D densities[12, 23]:

$$p(y|x,\theta) = p(y_1|\omega = h_{\theta_1}(x))p(y_2|\omega = h_{\theta_2}(x, y_1))$$

[2]We use data made publicly available www.nyc.gov/html/tlc. We used the data from January of 2016, as it provided an interesting proof of principle and scaling up to the entire dataset would have posed challenges outside the scope of this work.

Where h_{θ_1} and h_{θ_2} are both variational Bayesian neural networks.

Second, to accommodate the complexity of the conditional distributions and precisely capture subtle changes with time of day, fare amount and percent tip, for each of the conditionals we used a larger network, consisting of 2 layers of 200 hidden units and a $20-$stage normalizing flow. Additional details are included in Appendix D.

This approach allows us to derive insights about taxi pickups in Manhattan. For example, we observe interesting trends by looking at the conditional densities for different tip and fare amounts (Figure 2). For $\$5$ fares, we see an increased density in the upper East side, a wealthy, primarily residential area. This density is notably larger for trips with 20% tips than rides with no tip. Rides with no recorded tip with $\$5$ or $\$10$ fares, are most highly concentrated in midtown around Times Square and along 5^{th} avenue, which are particularly touristy areas of the city. For $\$20$ fares, density is higher around Wall Street, with notably higher density for trips with no tip than with a 20% tip.

5 Conclusion

We presented a Bayesian method for conditional density estimation using normalizing flows. By providing a framework for specifying priors over conditional distributions and performing approximate posterior inference using variational Bayesian neural networks, we achieved state of the art performance on two small benchmark datasets. We demonstrated the scalability of our method on the New York City yellow taxi dataset, on which our method is able to predict fine-grained predictive distributions.

References

[1] Christopher M Bishop. Mixture Density Networks. *Technical report*, 1, 1994.

[2] Christopher M Bishop. *Pattern Recognition and Machine Learning*. 2006.

[3] Thang D Bui, Daniel Hernández-Lobato, Yingzhen Li, José Miguel Hernández-Lobato, and Richard E Turner. Deep Gaussian Processes for Regression using Approximate Expectation Propagation. *ICML*, 48, 2016.

[4] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Learning and Policy Search in Stochastic Dynamical Systems with Bayesian Neural Networks. *ICLR*, 2017.

[5] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Uncertainty Decomposition in Bayesian Neural Networks with Latent Variables. *arXiv*, 2017.

[6] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear Independent Components Estimation. *Iclr*, 2015.

[7] José Miguel Hernández-Lobato and Ryan P Adams. Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. *Journal of Machine Learning Research*, 37:1–6, 2015.

[8] Robert a. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):79–87, 1991.

[9] Michael I Jordan, Tommi S Jaakkola, Lawrence K Saul, and Florham Park. An Introduction to Variational Methods for Graphical Models An Introduction to Variational Methods for Graphical Models. 233(January):183–233, 1998.

[10] Diederik P. Kingma and Jimmy Lei Ba. Adam: a Method for Stochastic Optimization. *International Conference on Learning Representations 2015*, pages 1–15, 2015.

[11] Diederik P Kingma, Tim Salimans, and Max Welling. Variational Dropout and the Local Reparameterization Trick. *arXiv*, (Mcmc):1–13, 2015.

[12] Hugo Larochelle and Iain Murray. The Neural Autoregressive Distribution Estimator. *International Conference on Machine Learning*, 15:29–37, 2011.

[13] Yingzhen Li and Yarin Gal. Dropout Inference in Bayesian Neural Networks with Alpha-divergences. *ICML*, 2017.

[14] Christos Louizos and Max Welling. Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors. *Icml*, 48, 2016.

[15] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. *arXiv*, 2014.

[16] Iain Murray and Hugo Larochelle. A Deep and Tractable Density Estimator. *ICML*, 32, 2014.

[17] Radford M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, 1995.

[18] George Papamakarios and Iain Murray. Fast epsilon-free Inference of Simulation Models with Bayesian Conditional Density Estimation. *NIPS*, pages 1–16, 2016.

[19] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked Autoregressive Flow for Density Estimation. *arXiv*, 2017.

[20] Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. *ICML*, 37:1530–1538, 2015.

[21] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning Structured Output Representation using Deep Conditional Generative Models. *NIPS*, 2015.

[22] Brian Trippe. *Complex Uncertainty in Machine Learning*. PhD thesis, University of Cambridge, 2017.

[23] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural Autoregressive Distribution Estimation. *JMLR*, 2016.

[24] Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional Image Generation with PixelCNN Decoders. *NIPS*, pages 4790–4798, 2016.

[25] Sida I Wang and Christopher D Manning. Fast dropout training. *ICML*, 28:118–126, 2013.

A An Alternative Parameterization of Radial Flows

If we are to reason about placing priors over distributions parameterized by normalizing flows, we are best served by using a parameterization which is more naturally interpretable. As such, we developed an alternative parameterization of the family of radial flows described by [20] for which we found it easier to reason about priors. In particular, we defined a new parameterization as:

$$f(z) = z + \frac{\alpha\beta(z - z_0)}{\alpha + |z - z_0|} \tag{3}$$

Where the parameters are $\{\beta, z_0 \in \mathbb{R}, \alpha \in \mathbb{R}^+\}$.

We can gain intuition into how this function shapes a probability density by examining its gradient:

$$\begin{aligned}
\frac{df(z)}{dz} &= 1 + \frac{d}{dz}\frac{\alpha\beta(z - z_0)}{\alpha + |z - z_0|} \\
&= 1 + \frac{(\alpha + |z - z_0|)\frac{d}{dz}(\alpha\beta(z - z_0)) - \alpha\beta(z - z_0)\frac{d}{dz}(\alpha + |z - z_0|)}{(\alpha + |z - z_0|)^2} \\
&= 1 + \frac{\alpha^2\beta + \alpha\beta|z - z_0| - \alpha\beta|z - z_0|}{(\alpha + |z - z_0|)^2} \\
&= 1 + \frac{\alpha^2\beta}{(\alpha + |z - z_0|)^2}
\end{aligned} \tag{4}$$

Looking closely at f and its derivative, we see that the warping varies from the identity to the greatest extent when $z = z_0$, where $f'(z_0) = 1 + \beta$. For values of z far from z_0, the gradient asymptotes to unity, which reflects no local perturbation (i.e. two inputs close to each-other but far from z_0 will be moved very little relative to one another). Additionally, we see that $\beta < 0$ leads a log-gradient which is positive, and therefore a compression of the density (we see this in equation 2). When $\beta = 0$, f is the identity, and does not warp the base density, and $\beta < 0$ leads a log-gradient which is

negative, and therefore a thinning of density. Furthermore, when $\beta = -1$, $\lim_{z \to z_0} f'(z) = 0$ and $\lim_{z \to z_0} \ln p(z_0) = -\infty$. Finally, when $\beta < -1$, f is no longer monotonic, and equation 2 no longer reflects the probability distribution.

Accordingly, to ensure strict monotonicity we restrict $\beta > -1$. We accomplish this by parameterizing β as $\beta = \exp(\hat{\beta}) - 1$. A number of possible parameterizations exist which could be used, but we prefer this for two reasons. First, in contrast to the original parameterization in [20], β is independent of second shape parameter, α. Second the magnitude of the of untransformed representation, $\hat{\beta}$, is equal to the maximum magnitude of the log derivative of the function it parameterizes. *As a result, priors we place on $\hat{\beta}$ are priors on the maximum change in log density induced by the warping:*

$$\ln \frac{df}{dz}(z_0) = \ln\big(1 + \beta\big) = \hat{\beta} \tag{5}$$

The second shape parameter, α, has an interpretable role in the warping as well. Examining equations 3 and 4, we see that α controls how quickly $\frac{df}{dz}$ decays back to 1 as z moves away from z_0. As such, large values of α lead to longer range distortions of the base distribution that shift points relative to one another to a greater extent. In this formulation, two points could be shifted by at most $2\alpha|\beta|$ relative to one another; away from one another for $\beta > 0$, and towards one another for $\beta < 0$. As α approaches 0, the perturbation defined by the warping occurs more quickly and produces a smaller distortion, with f collapsing to the identity function when $\alpha = 0$.

As with β, we have a constraint on α which we must satisfy to maintain monotonicity, $\alpha > 0$. We impose this by parameterizing α as a softplus transformed unconstrained parameter, $\alpha = \ln(\exp(\hat{\alpha}) + 1)$, where $\hat{\alpha}$ is an unconstrained real number. As such, priors on α should reflect our beliefs about the length scales at which the parameterized distribution will depart from the base distribution. These properties make it easier to reason about the relationship between the priors we place on parameters of the radial flows and the probability densities they define.

B Placing Priors on Conditional Density Estimators

We have discussed some of the challenges associated with building expressive models for conditional density estimation which do not overfit to training data and in Appendix A we introduced a parameterization of normalizing flows which can reflect assumptions about the distributions they will represent. This provides the foundation for a probabilistic framework for CDE with normalizing flows. In this section, we delve deeper into how one can place priors over distributions and conditional density estimators with normalizing flows.

B.1 Placing Priors over Probability Distributions

How much do we expect the target distribution to deviate from non-Gaussianity, and in what ways? In the context of normalizing flows, this is determined by the parametric family used, and how the parameters are set. When we use a neural network to parameterize these, our initialization and regularization scheme implicitly places a prior of these distributions, so we should be concerned with how these implicit priors manifest themselves in the resultant distributions.

We can get better sense of how the priors we place over the parameters of normalizing flows relate to the resultant probability densities by plotting sampled densities for Gaussian priors with different means and variances. For example, the densities in figure 3 represent a single set of sampled parameters of a density defined by a Gaussian base distribution passed through 10 radial flows for different prior means and variances. We can smoothly interpolate between different priors for single sample using the reparameterization trick. Looking at how the distributions change for different settings of the prior provides intuition into the kinds of distributions which have higher density under different priors.

B.2 Placing Priors over Conditional Density Estimators

Just as in the context of function approximation, where our ability to reason about the values functions take on at unseen points arises from assumptions about its the continuity and smoothness, if we

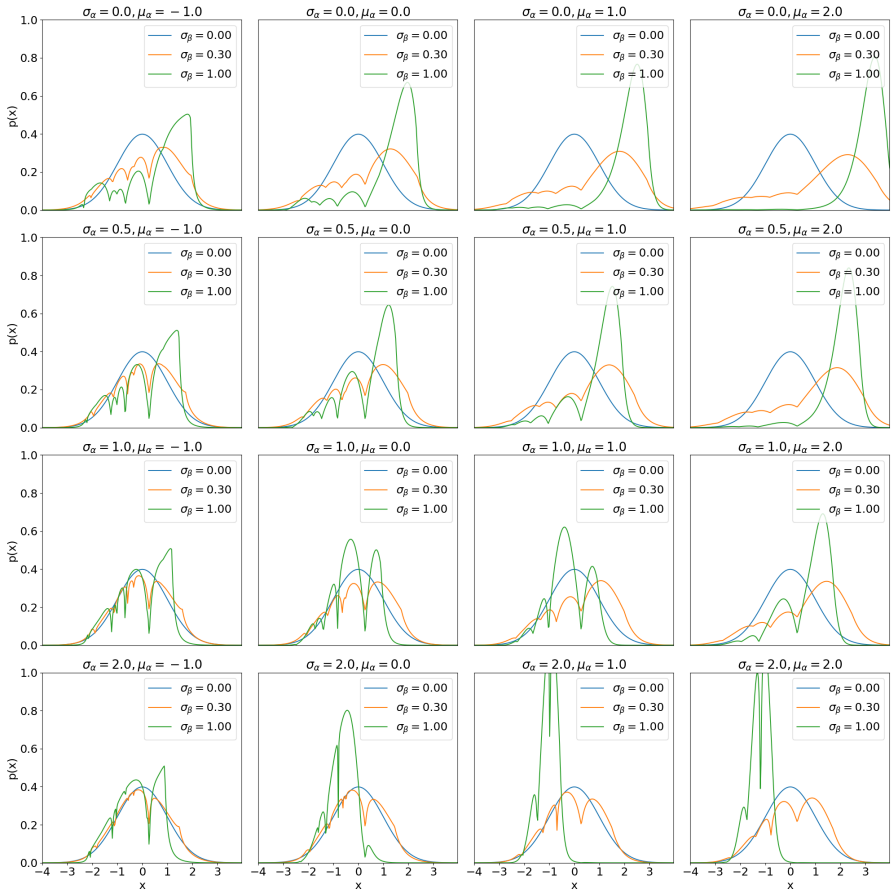

Figure 3: A manifold representing a probability densities sampled from different priors over a 10-stage normalizing flow. The same random seed is used to interpolate between different choices of priors to demonstrate the impact of different choices of priors on the resultant distribution. The variance in maximum magnitudes of the distortions are controlled by $\sigma_{\hat{\beta}}$, which varies across the densities within each subplot. The sharpness of the distortions is controlled by $\mu_{\hat{\alpha}}$, which varies from sharpest to smoothest across the columns. The variance of the sharpness of the distortions is controlled by $\sigma_{\hat{\alpha}}$, and is increased in successive rows. The remaining parameters are fixed at $\sigma_{\hat{\beta}} = 1.0$, $\mu_z = 0.0$ and $\mu_{\hat{\beta}} = 0.0$. Best viewed in color.

are to generalize in our predictions about conditional distributions we will need to make similar assumptions. For function approximation, this assumption often takes the form of assuming that inputs which are 'similar' to one another will take on 'similar' values.

$$x_i \approx x_j \implies y_i \approx y_j$$

A notion of similarity of inputs is crucial to making reasonable inferences about a function. When we choose a model class, we are defining what it means for an x_i and x_j to be close to one another, whether consciously or unconsciously, be it through the choice of a covariance function in a Gaussian process or the architecture of a deep neural network. For example, the widths, strides and dimensions of convolutions in a neural network for image classification enable us define the kinds of spatial translations under which images may be shifted and remain similar to one another.

In CDE, this is no longer the case; this property will not hold for heavy tailed or bimodal conditional distributions, for example. Instead, we build off the assumption that *for 'similar' observed variables, the corresponding conditional distributions will be 'similar' to one another*:

$$x_i \approx x_j \implies p(y|x_i) \approx p(y|x_j)$$

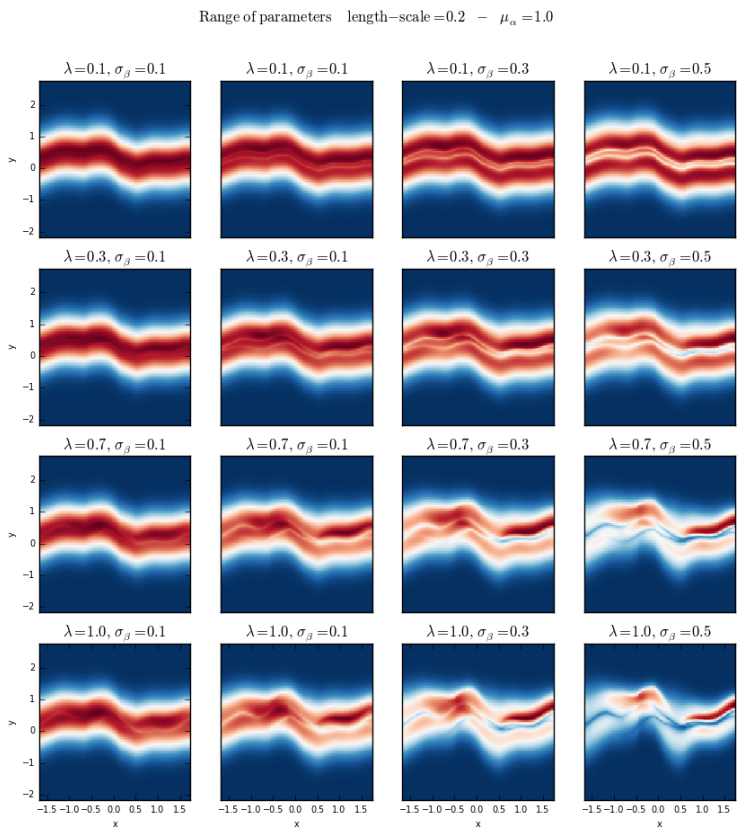

Figure 4: The characteristics of conditional density estimators sampled from different priors vary with the prior parameters. Each panel is a heatmap depicting the conditional density estimator defined by an MLP mapping to the parameters of a 5-stage normalizing flow. The same random seed for each sample, and the reparameterization trick is used to interpolate between different Gaussian priors. Moving left to right, we increase the prior standard deviation over the parameter β, which controls the magnitude of the warping. Moving top to bottom, we increase the value of the parameter λ, which controls the extent of heteroscedasticity (with larger values reflecting greater heteroscedasticity). Best viewed in color.

This assumption in CDE, clearly begs the additional question, "what does it mean for distributions to be 'similar'? ". We should ideally choice a definition of similarity which reflects our understanding of the problem we are solving.

For example, in estimating the location of taxis in NYC conditioned on time of day, we expect to see significant shifts in which areas of the city have greatest density, but we might expect that centers of density will generally exist in the same places (e.g. around major train stations and highly populated areas) and never in others (e.g. rivers). In contrast, if we are considering the probability of a worker's wage conditioned on the year, we might expect some characteristics of this distribution to be conserved across yeas (e.g. heavy upper tails or sharp peaks at minimum wages) but we also expect global translations of this distribution as a result of slowly varying trends, such as inflation, changing minimum wage or economic growth. Our priors about how $p(y|x)$ changes as x changes should inform how we define this notion of distance between distributions and infer a conditional density estimator.

Perhaps the easiest way to implement a notion of closeness of conditional distributions is in parameter space. By placing a prior on θ that reflects a belief that h_θ is smooth and slowly varying, we encode a belief that the conditional distributions change slowly throughout the space as well. This is implicitly done in previous work on CDE, however, the relationship between the closeness of parameters and

conditional distributions is perhaps less obvious for normalizing flows than it is for other distributions, such as mixture models or exponential family distributions.

This observation that neural network models with normalizing flows use a notion of similarity tied to distance in parameter space sheds additional light on the importance of choice of parameterization. For our parameterization of normalizing flows in equation 3, distance in parameter space has an intuitive interpretation when considering distance in the space of distributions. We can understand this by looking back at parameterization the radial flow and its gradient in equation 4. By placing priors how quickly the untransformed parameter $\hat{\beta}$ change throughout the input domain, we are directly placing priors on how quickly the maximum expansion or contraction of the base distribution, in terms of log probability density, induced by each stage of the normalizing flow will change. Priors how $\hat{\alpha}$ and z vary in turn act as priors on how the sharpness and center points of the distortions will vary. Tuning our of priors on the variances of these parameters relative to one another translates directly into priors over the kinds of shifts in the conditional distributions which we believe best explain data we see.

One can get a sense of the kinds of beliefs different priors express about conditional density estimators by considering samples from these priors, as illustrated in Figure 4. By placing different priors on on how each parameter of the function changes, we adjust a definition of distributional similarity. For the class of models described in this paper, the hyper-parameters define the magnitude of changes in parameters of the normalizing flows.

B.3 Homoscedasticity vs Heteroscedasticity

Perhaps the clearest motivation for defining a notion of distributional similarity through specification of priors is the trade-off between modeling complexity of conditional distributions and the complexity of their changes throughout the input space.

Especially when working with small datasets, prior beliefs about the noise distribution and how it varies across the input space heavily shape the posterior predictive distributions. Models for CDE are often split into homoscedastic and heteroscedastic models, where homoscedastic models have a stationary noise distribution and heteroscedastic models have varying noise distributions throughout the input space. However, as we discussed earlier, the tractability of any model for CDE hinges on the assumption that the noise distribution cannot change too quickly. We acknowledge that this is a fundamental trade-off and demonstrate how to smoothly interpolate between homoscedastic and very heteroscedastic models.

If we choose to parameterize h_θ as an MLP, we can smoothly interpolate between homoscedastic and heteroscedastic models by adapting our prior on the weights mapping from the final layer to the parameters of the normalizing flows, we do this with the parameter λ, which defines a multiplicative scaling of the final layer weights (but not biases). When these weights are small, the conditional distributions will vary only slightly away from their biases. The resulting functions have the same length scale for changes in noise structure, but the magnitude of the deviations are tuned by these final layer weights. Figure 4 demonstrates how larger values of λ give rise to greater heteroscedasticity.

Similarly, the length scales of the changes in the noise distribution (the weights/bias ratio in the first layer [17]) tunes the kind noise structure with high density under the prior. In the models we explored in this paper, we use the same MLP to predict the all characteristics of the conditionals, though we note that this expresses strong beliefs about the length scales of the changes in characteristics of these distributions being similar, which may or may not be a good assumption.

C Comparisons

For all models, we defined h_θ as an MLP with a single hidden layer of 50 units with tanh activations. We optimize with Adam [10] with hyper parameters $\beta_1 = 0.9$ and $\beta_2 = 0.99$ and learning rate 0.005 and ran our batch optimization for 5000 iterations. In all models, we use the local reparameterization trick [11] and calculate the $D_{\mathrm{KL}}(q||p)$ and its gradient analytically. For all models during both training and testing, used 20 MC samples of weights and biases.

C.1 Mixture density networks

We follow [18] in using stochastic VI to implement a Bayesian mixture density network (MDN). We use a diagonal Gaussian approximate posterior over weights and biases. We test two models, one with 2 mixing components (MDN-2) and a second with 5 mixing components(MDN-5).

Breaking from previous implementations of MDNs, we parameterized the model likelihood with an additional offset parameter, s, such that $\omega = \cup_{c=1}^{C}\{\mu_c, \sigma_c, \lambda_c\} \cup \{s\}$:

$$p(y|\omega) = \sum_{c=1}^{C} \lambda_c \mathcal{N}(y|\mu_c + s, \sigma_c^2)$$

where we enforce $\sum_{i=1}^{C} \lambda_c = 1$ by defining $\lambda = \mathrm{Softmax}(\hat{\lambda})$, and enforce $\sigma_c > 0$ by defining $\sigma_c = \mathrm{softplus}(\hat{\sigma}_c)$, where $\hat{\lambda}$ and $\hat{\sigma}$ are unconstrained outputs of the neural network. This parameterization is redundant in two ways, first, the global shift given by s could equivalently be encoded by shifts in the means each of the mixing components and second, λ is a C-dimensional simplex variable with only $C - 1$ degrees of freedom. We prefer this parameterization of the component means because, in the context of our variational neural network approximation, it does not penalize global shifts distributions according to the complexity of these distributions (as would be the case otherwise). As such, a complex but homoscedastic noise distribution may be more easily represented.

We placed zero-mean Gaussian priors on the weights and biases. In previous work [22], we found that the mean field approximations were often quite sensitive to the prior standard deviation, so optimized this hyperparameter using Bayesian optimization. We initialized the standard deviations of the approximate posteriors to be very small ($\sigma^{\mathrm{init}} = 10^{-5}$).

C.2 Normalizing Flows

We used a diagonal Gaussian variational approximation with fixed weight variances as this model performed well in the model assuming homoscedastic, Gaussian observations. We tested two normalizing flow based conditional density estimators with different levels of complexity, one with two radial warpings (NF-2) and one with five radial warpings (NF-5). We chose these models to have similar expressive power and the same number of parameters as the two mixture density network models, MDN-2 and MDN-5. We performed hyperparameter optimization on three of the models' hyper parameters, $\sigma_{\hat{\beta}}$, λ and the posterior variances of the weights and biases σ_w.

C.3 Neural Networks with Latent Variables

The third conditional density estimator which we tested is a neural network with latent variable inputs. We fit this model by mean field variational inference, following the implementation used by [4]. We tested two models, one in which we calculated likelihoods using 5 samples of noise(LV-5) and one calculating likelihoods with 15 samples (LV-15). We make this choice attempting to pick models with roughly the same expressive power as the corresponding normalizing flow and MDN models, however, given the marked difference between the approaches, an objective comparison of expressive power is not possible. As with the Bayesian MDN's, we initialized the standard deviations of the approximate posteriors to be 10^{-5} and selected the prior standard deviation using Bayesian optimization.

Unlike [4], we use tanh activation units. Previous work used ReLU hidden units which, when using input noise sampled from a uniform distribution, results in densities which are piece-wise linear. As such, learning such, we find learning distributions with small ReLU networks in this way to be very difficult as compared to similar networks with smooth activation functions.

This approach is more difficult to scale up in a stochastic variational Bayesian framework, where we are forced to use Monte Carlo samples over model parameters as well as the inherent stochasticity. Similarly, evaluating the density under the posterior predictive distribution requires multiple samples for both weights of the network and the input noise.

Table 1: Mean log-likelihood in nats for normalizing flows, mixture density networks and neural networks with latent inputs on six small UCI benchmark regression dataset. Higher is better.

Dataset	N	D	**MDN-2**	**MDN-5**	**LV-15**	**LV-5**	**NF-2**	**NF-5**
boston	506	13	-2.65±0.03	-2.73±0.04	-2.64±0.05	-2.56±0.05	-2.40±0.06	**-2.37±0.04**
concrete	1030	8	-3.23±0.03	-3.28±0.03	-3.06±0.03	-3.08±0.02	-3.03±0.05	**-2.97±0.03**
energy	768	8	-1.60±0.04	-1.63±0.06	-0.74±0.03	-0.79±0.02	**-0.44±0.04**	-0.67±0.15
power	9568	4	-2.73±0.01	-2.70±0.01	-2.81±0.01	-2.82±0.01	-2.73±0.01	**-2.68±0.01**
wine	1588	11	-0.91±0.04	**1.43±0.07**	-0.98±0.02	-0.96±0.01	-0.87±0.02	-0.76±0.10
yacht	308	6	-2.70±0.05	-2.54±0.10	-1.01±0.04	-1.15±0.05	-0.30±0.04	**-0.21±0.09**

$$p(y|x,\mathcal{D},\alpha,\eta) = \int_\theta p(\theta|\mathcal{D},\alpha,\eta)\int_z p(z|\eta)p(y|x,z,\theta)dzd\theta \approx \frac{1}{M}\sum_{i=1}^{M}\frac{1}{K}\sum_{j=1}^{K} p(y|x,z_j,\theta_i) \quad (6)$$

where each $\theta_i \sim p(\theta|\mathcal{D},\alpha,\eta)$ and each $z_j \sim p(z|\eta)$

We report the mean and standard error of performance on held-out datasets in table 1.

D Spatial Conditional Density Estimation Experimental Details

As our observed variables, we consider pickup time, number of passengers, fare amount, and percent tip. We presented time of day as an input to the model, parameterizing with two variables as $\left(\sin(\frac{2\pi\cdot\text{hour}}{24}), \cos(\frac{2\pi\cdot\text{hour}}{24})\right)$. This parameterization of time is preferable in that it ensures that similar times are close in input space (e.g. 11:59pm is close to 12:00am, which is not the case for a one dimensional parameterization of time).

All input features and labels are normalized to have zero-mean a unit variance. We performed variational inference using the local reparameterization trick. We used a learning rate of $2 \cdot 10^{-5}$, and ran for 1500 epochs with batch size of 2048 on a NVIDIA Tesla K80 GPU. For hyper-parameters, we set $\mu_{\hat{\alpha}} = 1$ and $\sigma_{\hat{\alpha}} = 0.1$, $\lambda = 1$,$\mu_{\hat{\beta}} = 0$ and $\sigma_{\hat{\beta}} = 1$, $\mu_z = 0$ and $\sigma_z = 1$, and the prior over weights to be unit Gaussian. We fixed the posterior uncertainties in weights and biases to 10^{-5}.