Hybrid Models with Deep and Invertible Features

Eric Nalisnick [*] [1] **Akihiro Matsukawa** [*] [1] **Yee Whye Teh** [1] **Dilan Gorur** [1] **Balaji Lakshminarayanan** [1]

Abstract

We propose a neural hybrid model consisting of a linear model defined on a set of features computed by a deep, invertible transformation (i.e. a normalizing flow). An attractive property of our model is that both $p(\texttt{features})$, the density of the features, and $p(\texttt{targets}|\texttt{features})$, the predictive distribution, can be computed exactly in a single feed-forward pass. We show that our hybrid model, despite the invertibility constraints, achieves similar accuracy to purely predictive models. Yet the generative component remains a good model of the input features despite the hybrid optimization objective. This offers additional capabilities such as detection of out-of-distribution inputs and enabling semi-supervised learning. The availability of the exact joint density $p(\texttt{targets}, \texttt{features})$ also allows us to compute many quantities readily, making our hybrid model a useful building block for downstream applications of probabilistic deep learning.

1. Introduction

In the majority of applications, deep neural networks model conditional distributions of the form $p(y|\boldsymbol{x})$, where y denotes a label and $\boldsymbol{x}$ features or covariates. However, modeling just the conditional distribution is insufficient in many cases. For instance, if we believe that the model may be subjected to inputs unlike those of the training data, a model for $p(\boldsymbol{x})$ can possibly detect an outlier before it is passed to the conditional model for prediction. Thus modeling the joint distribution $p(y, \boldsymbol{x})$ provides a richer and more useful representation of the data. Models defined by combining a predictive model $p(y|\boldsymbol{x})$ with a generative one $p(\boldsymbol{x})$ are known as *hybrid models* (Jaakkola & Haussler, 1999; Raina et al., 2004; Lasserre et al., 2006; Kingma et al., 2014). Hybrid models have been shown to be useful for novelty detection (Bishop, 1994), semi-supervised learning (Druck et al., 2007), and information regularization (Szummer & Jaakkola, 2003).

Crafting a hybrid model usually requires training two models, one for $p(y|\boldsymbol{x})$ and one for $p(\boldsymbol{x})$, that share a subset (Raina et al., 2004) or possibly all (McCallum et al., 2006) of their parameters. Unfortunately, training a high-fidelity $p(\boldsymbol{x})$ model alone is difficult, especially in high dimensions, and good performance requires using a large neural network (Brock et al., 2019). Yet principled probabilistic inference is hard to implement with neural networks since they do not admit closed-form solutions and running Markov chain Monte Carlo takes prohibitively long. Variational inference then remains as the final alternative, and this now introduces a *third* model, which usually serves as the posterior approximation and/or inference network (Kingma & Welling, 2014; Kingma et al., 2014). To make matters worse, the $p(y|\boldsymbol{x})$ model may require a separate approximate inference scheme, leading to additional computation and parameters.

In this paper, we propose a neural hybrid model that overcomes many of the aforementioned computational challenges. Most crucially, our model supports *exact* inference and evaluation of $p(\boldsymbol{x})$. Furthermore, in the case of regression, Bayesian inference for $p(y|\boldsymbol{x})$ is exact and available in closed-form as well. Our model is made possible by leveraging recent advances in deep invertible generative models (Rezende & Mohamed, 2015; Dinh et al., 2017; Kingma & Dhariwal, 2018). These models are defined by composing invertible functions, and therefore the change-of-variables formula can be used to compute exact densities. Moreover, these invertible models have been shown to be expressive enough to perform well on prediction tasks (Gomez et al., 2017; Jacobsen et al., 2018). We use the invertible function as a natural feature extractor and define a linear model at the level of the latent representation, which is memory-efficient as the bulk of the parameters are shared between $p(\boldsymbol{x})$ and $p(y|\boldsymbol{x})$. Furthermore, with just *one feed-forward pass* we can obtain both $p(\boldsymbol{x})$ and $p(y|\boldsymbol{x})$, with the only additional cost being the log-determinant-Jacobian term required by the change of variables. While this term could be expensive to compute for general functions, much recent work has been done on defining expressive invertible neural networks with easy-to-evaluate volume elements (Dinh et al., 2015; 2017; Kingma & Dhariwal, 2018; Grathwohl et al., 2019).

[*]Equal contribution [1]DeepMind. Correspondence to: Balaji Lakshminarayanan <balajiln@google.com>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

In summary, our contributions are:

- Defining a neural hybrid model with exact inference and evaluation of $p(y, \boldsymbol{x})$, which can be computed in one feed-forward pass and without any Monte Carlo approximations.
- Evaluating the model's predictive accuracy and uncertainty on both classification and regression problems.
- Using the model's natural 'reject' rule based on the generative component $p(\boldsymbol{x})$ to filter out-of-distribution (OOD) inputs.
- Showing that our hybrid model performs well at semi-supervised classification.

2. Background

We begin by establishing notation and reviewing the necessary background material. We denote matrices with upper-case and bold letters (e.g. $\boldsymbol{X}$), vectors with lower-case and bold (e.g. $\boldsymbol{x}$), and scalars with lower-case and no bolding (e.g. x). Let the collection of all observations be denoted $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{y}\} = \{(\boldsymbol{x}_n, y_n)_{n=1}^{N}\}$ with $\boldsymbol{x}$ representing a vector containing features and y a scalar representing the corresponding label. We define a predictive model's density function to be $p(y|\boldsymbol{x}; \boldsymbol{\theta})$ and a generative density to be $p(\boldsymbol{x}; \boldsymbol{\theta})$, where $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ are the shared model parameters. Let the joint likelihood be denoted $p(\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{\theta}) = \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n; \boldsymbol{\theta}) p(\boldsymbol{x}_n; \boldsymbol{\theta})$.

2.1. Invertible Generative Models

Deep invertible transformations are the first key building block in our approach. These are simply high-capacity, bijective transformations with a tractable Jacobian matrix and inverse. The best known models of this class are the *real non-volume preserving* (RNVP) transform (Dinh et al., 2017) and its recent extension, the *Glow* transform (Kingma & Dhariwal, 2018). The bijective nature of these transforms is crucial as it allows us to employ the change-of-variables formula for *exact* density evaluation:

$$\log p_x(\boldsymbol{x}) = \log p_z(f(\boldsymbol{x}; \boldsymbol{\phi})) + \log \left| \frac{\partial \boldsymbol{f}_\phi}{\partial \boldsymbol{x}} \right| \qquad (1)$$

where $f(\cdot; \boldsymbol{\phi})$ denotes the transform with parameters $\boldsymbol{\phi}$, $|\partial \boldsymbol{f} / \partial \boldsymbol{x}|$ the determinant of the Jacobian of the transform, and $p_z(\boldsymbol{z} = f(\cdot; \boldsymbol{\phi}))$ a distribution on the latent variables computed from the transform. The modeler is free to choose p_z, and therefore it is often set as a factorized standard Gaussian for computational simplicity. The *affine coupling layer* (ACL) (Dinh et al., 2017) is the key building block used by RNVP and Glow to define $f(\cdot; \boldsymbol{\phi})$. It consists of transforming half of the representation with translation and scaling operations and copying the other half forward to the output. See Appendix A in the supplementary material for a detailed description of the ACL. Glow (Kingma & Dhariwal, 2018) introduces 1×1 convolutions between ACLs. The parameters $\boldsymbol{\phi}$ are estimated via maximizing the exact log-likelihood $\log p(\boldsymbol{X}; \boldsymbol{\phi})$.

While the invertibility requirements imposed on f may seem too restrictive to define an expressive model, recent work using invertible transformations for classification Jacobsen et al. (2018) reports metrics comparable to non-invertible residual networks, even on challenging benchmarks such as ImageNet, and recent work by Kingma & Dhariwal (2018) has shown that invertible generative models can produce sharp samples. Sampling from a flow is done by first sampling from the latent distribution and then passing that sample through the inverse transform: $\hat{\boldsymbol{z}} \sim p_z$, $\hat{\boldsymbol{x}} = f^{-1}(\hat{\boldsymbol{z}})$.

2.2. Generalized Linear Models

Generalized linear models (GLMs) (Nelder & Baker, 1972) are the second key building block that we employ. They model the expected response y as follows:

$$\mathbb{E}[y_n|\boldsymbol{z}_n] = g^{-1}\left(\boldsymbol{\beta}^T \boldsymbol{z}_n\right) \qquad (2)$$

where $\mathbb{E}[y|\boldsymbol{z}]$ denotes the expected value of y_n, $\boldsymbol{\beta}$ a $\mathbb{R}^d$ vector of parameters, $\boldsymbol{z}_n$ the covariates, and $g^{-1}(\cdot)$ a *link function* such that $g^{-1} : \mathbb{R} \mapsto \mu_{y|\boldsymbol{z}}$. For notational convenience, we assume a scalar bias β_0 has been subsumed into $\boldsymbol{\beta}$. A Bayesian GLM could be defined by specifying a prior $p(\boldsymbol{\beta})$ and computing the posterior $p(\boldsymbol{\beta}|\boldsymbol{y}, \boldsymbol{Z})$. When the link function is the identity (i.e. simple linear regression) and $\boldsymbol{\beta} \sim \text{N}(\boldsymbol{0}, \boldsymbol{\Lambda}^{-1})$, then the posterior distribution is available in closed-form:

$$p(\boldsymbol{\beta}|\boldsymbol{y}, \boldsymbol{X}) = \text{N}\left(\frac{\boldsymbol{X}^T\boldsymbol{y}}{\boldsymbol{X}^T\boldsymbol{X} + \sigma_0^2\boldsymbol{\Lambda}}, \frac{\sigma_0^2}{\boldsymbol{X}^T\boldsymbol{X} + \sigma_0^2\boldsymbol{\Lambda}}\right) \qquad (3)$$

where σ_0 is the response noise. In the case of logistic regression, the posterior is no longer conjugate but can be closely approximated (Jaakkola & Jordan, 1997).

3. Combining Deep Invertible Transforms and Generalized Linear Models

We propose a neural hybrid model consisting of a deep invertible transform coupled with a GLM. Together the two define a *deep* predictive model with both the ability to compute $p(\boldsymbol{x})$ and $p(y|\boldsymbol{x})$ *exactly*, in a *single feed-forward pass*. The model defines the following joint distribution over a label-feature pair $(y_n, \boldsymbol{x}_n)$:

$$\begin{aligned} p(y_n, \boldsymbol{x}_n; \boldsymbol{\theta}) &= p(y_n|\boldsymbol{x}_n; \boldsymbol{\beta}, \boldsymbol{\phi}) \;\; p(\boldsymbol{x}_n; \boldsymbol{\phi}) \\ &= p(y_n|f(\boldsymbol{x}_n; \boldsymbol{\phi}); \boldsymbol{\beta}) \;\; p_z(f(\boldsymbol{x}_n; \boldsymbol{\phi})) \left| \frac{\partial \boldsymbol{f}_\phi}{\partial \boldsymbol{x}_n} \right| \end{aligned} \qquad (4)$$

where $\boldsymbol{z}_n = f(\boldsymbol{x}_n, \boldsymbol{\phi})$ is the output of the invertible transformation, $p_z(\boldsymbol{z})$ is the latent distribution (also referred to

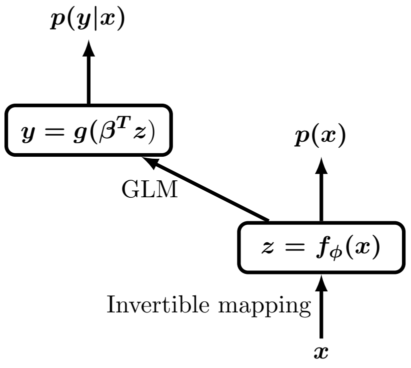


Figure 1. Model Architecture. The diagram above shows the DIGLM's computational pipeline, which is comprised of a GLM stacked on top of an invertible generative model. The model parameters are $\boldsymbol{\theta} = \{\boldsymbol{\phi}, \boldsymbol{\beta}\}$ of which $\boldsymbol{\phi}$ is shared between the generative and predictive model, and $\boldsymbol{\beta}$ denotes parametrizes the GLM in the predictive model.

as the prior or base distribution), and $p(\mathbf{y}_n|f(\boldsymbol{x}_n;\boldsymbol{\phi});\boldsymbol{\beta})$ is a GLM with the latent variables serving as its input features. For simplicity, we assume a factorized latent distribution $p(\boldsymbol{z}) = \prod_d p(z_d)$, following previous work (Dinh et al., 2017; Kingma & Dhariwal, 2018). Note that $\boldsymbol{\phi} = \{\boldsymbol{\phi}_{t,l}, \boldsymbol{\phi}_{s,l}\}_{l=1}^{L}$ are the parameters of the generative model and that $\boldsymbol{\theta} = \{\boldsymbol{\phi}, \boldsymbol{\beta}\}$ are the parameters of the joint model. Sharing $\boldsymbol{\phi}$ between both components allows the conditional distribution to influence the generative distribution and vice versa. We term the proposed neural hybrid model the *deep invertible generalized linear model* (DIGLM). Given labeled training data $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$ sampled from the true distribution of interest $p^*(\boldsymbol{x}, y)$, the DIGLM can be trained by maximizing the exact joint log-likelihood, i.e.

$$\mathcal{J}(\boldsymbol{\theta}) = \log p(\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(y_n, \boldsymbol{x}_n; \boldsymbol{\theta}),$$

via gradient ascent. As per the theory of maximum likelihood, maximizing this log probability is equivalent to minimizing the Kullback-Leibler (KL) divergence between the true joint distribution and the model: $D_{\mathrm{KL}}\big(p^*(\boldsymbol{x}, y) \| p_{\boldsymbol{\theta}}(\boldsymbol{x}, y)\big)$.

Figure 1 shows a diagram of the DIGLM. We see that the computation pipeline is essentially that of a traditional neural network but one defined by stacking ACLs. The input $\boldsymbol{x}$ first passes through $f_{\boldsymbol{\phi}}$, and the latent representation and the stored Jacobian terms are enough to compute $p(\boldsymbol{x})$. In particular, evaluating $p_z(f(\boldsymbol{x}_n;\boldsymbol{\phi}))$ has an $\mathcal{O}(D)$ run-time cost for factorized distributions, and $|\partial \boldsymbol{f}_{\boldsymbol{\phi}} / \partial \boldsymbol{x}_n|$ has a $\mathcal{O}(LD)$ run-time for RNVP architectures, where L is the number of affine coupling layers and D is the input dimensionality. Evaluating the predictive model adds another $\mathcal{O}(D)$ cost in computation, but this cost will be dominated by the prerequisite evaluation of $f_{\boldsymbol{\phi}}$.

Weighted Objective In practice we found the DIGLM's performance improved by introducing a scaling factor on the contribution of $p(\boldsymbol{x})$. The factor helps control for the effect of the drastically different dimensionalities of y and $\boldsymbol{x}$. We denote this modified objective as:

$$\mathcal{J}_\lambda(\boldsymbol{\theta}) = \sum_{n=1}^{N} \Big(\log p(y_n|\boldsymbol{x}_n; \boldsymbol{\beta}, \boldsymbol{\phi}) + \lambda \log p(\boldsymbol{x}_n; \boldsymbol{\phi}) \Big) \quad (5)$$

where λ is the scaling constant. Weighted losses are commonly used in hybrid models (Lasserre et al., 2006; McCallum et al., 2006; Kingma et al., 2014; Tulyakov et al., 2017). Yet in our particular case, we can interpret the down-weighting as encouraging robustness to input variations. In other words, down-weighting the contribution of $\log p(\boldsymbol{x}_n;\boldsymbol{\phi})$ can be considered a Jacobian-based regularization penalty. To see this, notice that the joint likelihood rewards maximization of $|\partial \boldsymbol{f}_{\boldsymbol{\phi}} / \partial \boldsymbol{x}_n|$, thereby encouraging the model to increase the $\partial f_d / \partial x_d$ derivatives (i.e. the diagonal terms). This optimization objective stands in direct contrast to a long history of gradient-based regularization penalties (Girosi et al., 1995; Bishop, 1995; Rifai et al., 2011), which add the Frobenius norm of the Jacobian as a *penalty* to a loss function (or negative log-likelihood). Thus, we can interpret the de-weighting of $|\partial \boldsymbol{f}_{\boldsymbol{\phi}} / \partial \boldsymbol{x}_n|$ as adding a Jacobian regularizer with weight $\tilde{\lambda} = (1 - \lambda)$. If the latent distribution term is, say, a factorized Gaussian, the variance can be scaled by a factor of $1/\lambda$ to introduce regularization only to the Jacobian term.

3.1. Semi-supervised learning

As mentioned in the introduction, having a representation of the joint density enables the model to be trained on data sets that do not have a label for every feature vector—i.e. semi-supervised data sets. When a label is not present, the principled approach to the situation is to integrate out the variable:

$$\int_y p(y, \boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\phi})\, dy = p(\boldsymbol{x}; \boldsymbol{\phi}) \int_y p(y|\boldsymbol{x}; \boldsymbol{\theta})\, dy = p(\boldsymbol{x}; \boldsymbol{\phi}). \quad (6)$$

Thus, we should use the unpaired $\boldsymbol{x}$ observations to train just the generative component.

3.2. Selective Classification

Equation 6 above also suggests a strategy for evaluating the model in real-world situations. One can imagine the DIGLM being deployed as part of a user-facing system and that we wish to have the model 'reject' inputs that are unlike the training data. In other words, the inputs are anomalous with respect to the training distribution, and we cannot expect the $p(y|\boldsymbol{x})$ component to make accurate predictions when $\boldsymbol{x}$ is not drawn from the training distribution. In this setting we have access only to the user-provided features $\boldsymbol{x}^*$, and thus should evaluate by way of Equation 6 again, computing $p(\boldsymbol{x}^*; \boldsymbol{\phi})$. This observation then leads to the natural rejection rule:

$$\text{if } p(\boldsymbol{x}^*; \boldsymbol{\phi}) < \tau, \text{ then } \texttt{reject } \boldsymbol{x}^* \tag{7}$$

where τ is some threshold, which we propose setting as $\tau = \min_{\boldsymbol{x} \in \mathcal{D}} p(\boldsymbol{x}; \boldsymbol{\phi}) - c$ where the minimum is taken over the training set and c is a free parameter providing slack in the margin. When rejecting a sample, we output the unconditional $p(y)$, e.g. uniform probabilities for classification problems, hence the prediction for $\boldsymbol{x}^*$ is given by

$$p(y)\, \mathbb{1}[p(\boldsymbol{x}^*; \boldsymbol{\phi}) < \tau] + p(y|\boldsymbol{x}^*)\, \mathbb{1}[p(\boldsymbol{x}^*; \boldsymbol{\phi}) \geq \tau] \tag{8}$$

where $\mathbb{1}[\cdot]$ denotes an indicator function. Similar generative-model-based rejection rules have been proposed previously (Bishop, 1994). This idea is also known as *selective classification* or *classification with a reject option* (Hellman, 1970; Cordella et al., 1995; Fumera & Roli, 2002; Herbei & Wegkamp, 2006; Geifman & El-Yaniv, 2017).

4. Bayesian Treatment

We next describe a Bayesian treatment of the DIGLM, deriving some closed-form quantities of interest and discussing connections to Gaussian processes. The Bayesian DIGLM (B-DIGLM) is defined as follows:

$$f(\boldsymbol{x}; \boldsymbol{\phi}) \sim p(\boldsymbol{z}), \;\; \boldsymbol{\beta} \sim p(\boldsymbol{\beta}), \;\; y_n \sim p(y_n | f(\boldsymbol{x}_n; \boldsymbol{\phi}), \boldsymbol{\beta}).$$

The material difference from the earlier formulation is that a prior $p(\boldsymbol{\beta})$ is now placed on the regression parameters. The B-DIGLM defines the joint distribution of three variables—$p(y_n, \boldsymbol{x}_n, \boldsymbol{\beta}; \boldsymbol{\phi})$—and to perform proper Bayesian inference, we should marginalize over $p(\boldsymbol{\beta})$ when training, resulting in the modified objective:

$$\begin{aligned} p(y_n, \boldsymbol{x}_n; \boldsymbol{\phi}) &= \int_{\boldsymbol{\beta}} p(y_n, \boldsymbol{x}_n, \boldsymbol{\beta}; \boldsymbol{\phi})\, d\boldsymbol{\beta} \\ &= \int_{\boldsymbol{\beta}} p(y_n | \boldsymbol{x}_n; \boldsymbol{\phi}, \boldsymbol{\beta}) p(\boldsymbol{\beta})\, d\boldsymbol{\beta}\;\; p(\boldsymbol{x}_n; \boldsymbol{\phi}) \\ &= p(y_n | f(\boldsymbol{x}_n; \boldsymbol{\phi}))\;\; p_z(f(\boldsymbol{x}_n; \boldsymbol{\phi}))\;\; \left| \frac{\partial \boldsymbol{f}_{\boldsymbol{\phi}}}{\partial \boldsymbol{x}_n} \right| \end{aligned} \tag{9}$$

where $p(y_n | f(\boldsymbol{x}_n; \boldsymbol{\phi}))$ is the marginal likelihood of the regression model.

While $p(y_n | f(\boldsymbol{x}_n; \boldsymbol{\phi}))$ is not always available in closed-form, it is in some cases. For instance, if we assume that the likelihood model is Gaussian as in linear regression, and that $\boldsymbol{\beta}$ is given a zero-mean Gaussian prior, i.e.

$$p(y_n | \boldsymbol{z}_n, \boldsymbol{\beta}) = \text{N}(y_n; \boldsymbol{\beta}^T \boldsymbol{z}_n, \sigma_0^2), \quad \boldsymbol{\beta} \sim \text{N}(\boldsymbol{0}, \lambda^{-1}\mathbb{I})$$

then the marginal likelihood can be written as:

$$\begin{aligned} &\log p(y_n | f(\boldsymbol{x}_n; \boldsymbol{\phi})) \\ &= \log \text{N}\left(\boldsymbol{y};\, \boldsymbol{0},\, \sigma_0^2 \mathbb{I} + \lambda^{-1} \boldsymbol{Z}_{\boldsymbol{\phi}} \boldsymbol{Z}_{\boldsymbol{\phi}}^T\right) \\ &\propto -\boldsymbol{y}^T (\sigma_0^2 \mathbb{I} + \lambda^{-1} \boldsymbol{Z}_{\boldsymbol{\phi}} \boldsymbol{Z}_{\boldsymbol{\phi}}^T)^{-1} \boldsymbol{y} - \log \left| \sigma_0^2 \mathbb{I} + \lambda^{-1} \boldsymbol{Z}_{\boldsymbol{\phi}} \boldsymbol{Z}_{\boldsymbol{\phi}}^T \right| \end{aligned} \tag{10}$$

where $\boldsymbol{Z}_{\boldsymbol{\phi}}$ is the matrix of all latent representations, which we subscript with $\boldsymbol{\phi}$ to emphasize that it depends on the invertible transform's parameters.

Connection to Gaussian Processes From Equation 10 we see that B-DIGLMs are related to *Gaussian processes* (GPs) (Rasmussen & Williams, 2006). GPs are defined through their kernel function $k(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\psi})$, which in turn characterizes the class of functions represented. The marginal likelihood under a GP is defined as

$$\log p(y_n | \boldsymbol{x}_n; \boldsymbol{\psi}) \propto -\boldsymbol{y}^T (\sigma_0^2 \mathbb{I} + \boldsymbol{K}_{\boldsymbol{\psi}})^{-1} \boldsymbol{y} - \log \left| \sigma_0^2 \mathbb{I} + \boldsymbol{K}_{\boldsymbol{\psi}} \right|$$

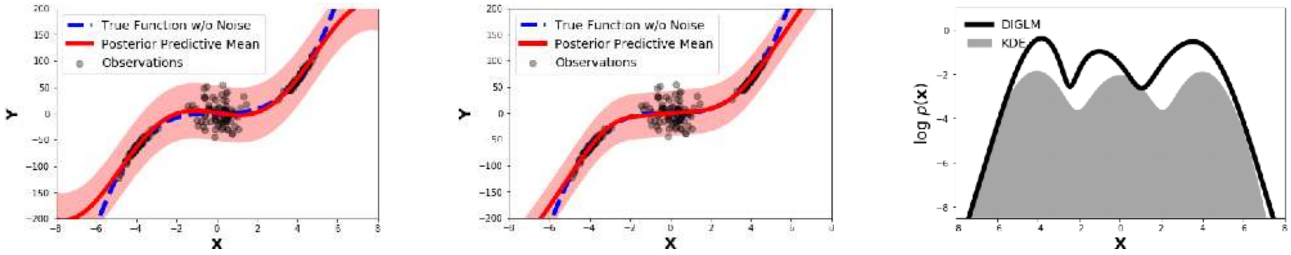
with $\boldsymbol{\psi}$ denoting the kernel parameters. Comparing this equation to the B-DIGLM's marginal likelihood in Equation 10, we see that they become equal by setting $\boldsymbol{K}_{\boldsymbol{\psi}} = \lambda^{-1} \boldsymbol{Z}_{\boldsymbol{\phi}} \boldsymbol{Z}_{\boldsymbol{\phi}}^T$, and thus we have the implied kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \lambda^{-1} f(\boldsymbol{x}_i; \boldsymbol{\phi})^T f(\boldsymbol{x}_j; \boldsymbol{\phi})$. Perhaps there are even deeper connections to be made via *Fisher kernels* (Jaakkola & Haussler, 1999) or *probability product kernels* (Jebara et al., 2004)—kernel functions derived from generative models—but we leave this investigation to future work.

Approximate Inference If the marginal likelihood is not available in closed form, then we must resort to approximate inference. In this case, understandably, our model loses the ability to compute exact marginal likelihoods. We can use one of the many lower bounds developed for variational inference to bypass the intractability. Using the usual variational Bayes evidence lower bound (ELBO) (Jordan et al., 1999), we have

$$\begin{aligned} \log p(y_n | f(\boldsymbol{x}_n; \boldsymbol{\phi})) \geq \mathbb{E}_{q(\boldsymbol{\beta})} &\left[p(y_n | \boldsymbol{x}_n; \boldsymbol{\phi}, \boldsymbol{\beta}) \right] \\ &- \text{KLD}\left[q(\boldsymbol{\beta}) || p(\boldsymbol{\beta}) \right] \end{aligned} \tag{11}$$

where $q(\boldsymbol{\beta})$ is a variational approximation to the true posterior. We leave thorough investigation of approximate inference to future work, and in the experiments we use either conjugate Bayesian inference or point estimates for $\boldsymbol{\beta}$.

One may ask: why stop the Bayesian treatment at the predictive component? Why not include a prior on the flow's parameters as well? This could be done, but Riquelme et al. (2018) showed that Bayesian linear regression with deep features (i.e. computed by a deterministic neural network) is highly effective for contextual bandit problems, which suggests that capturing the uncertainty in prediction parameters $\boldsymbol{\beta}$ is more important than the uncertainty in the representation parameters $\boldsymbol{\phi}$.



(a) Gaussian Process (b) B-DIGLM $p(y|\boldsymbol{x})$ (c) B-DIGLM $p(\boldsymbol{x})$

Figure 2. 1-dimensional Regression Task. We construct a toy regression task by sampling x-observations from a Gaussian mixture model and then assigning responses $y = x^3 + \epsilon$ with ϵ being heteroscedastic noise. Subfigure (a) shows the function learned by a Gaussian process and (b) shows the function learned by the Bayesian DIGLM. Subfigure (c) shows the $p(x)$ density learned by the same DIGLM (black line) and compares it to a KDE (gray shading).

5. Related Work

We are unaware of any work that uses normalizing flows as the generative component of a hybrid model. The most related work is the class conditional variant of Glow (Kingma & Dhariwal, 2018, Appendix D). For this model, Kingma & Dhariwal (2018) use class-conditional latent distributions and introduce a (down weighted) classification loss to the penultimate layer of the flow. However, they do not evaluate the model for its predictive capabilities and instead (qualitatively) evaluate its class-conditional generative abilities.

While several works have studied the trade-offs between generative and predictive models (Efron, 1975; Ng & Jordan, 2002), Jaakkola & Haussler (1999) were perhaps the first to meaningfully combine the two, using a generative model to define a kernel function that could then be employed by classifiers such as SVMs. Raina et al. (2004) took the idea a step further, training a subset of a naive Bayes model's parameters with an additional predictive objective. McCallum et al. (2006) extended this framework to train all parameters with both generative and predictive objectives. Lasserre et al. (2006) showed that a simple convex combination of the generative and predictive objectives does not necessarily represent a unified model and proposed an alternative prior that better couples the parameters. Druck et al. (2007) empirically compared Lasserre et al. (2006)'s and McCallum et al. (2006)'s hybrid objectives specifically for semi-supervised learning. Recent advances in deep generative models and stochastic variational inference have allowed the aforementioned frameworks to include neural networks as the predictive and/or generative components. Deep neural hybrid models haven been defined by (at least) Kingma et al. (2014), Maaløe et al. (2016), Kuleshov & Ermon (2017), Tulyakov et al. (2017), and Gordon & Hernández-Lobato (2017). However, these models, unlike ours, require approximate inference to obtain the $p(\boldsymbol{x})$ component.

As mentioned in the introduction, invertible residual networks have been shown to perform as well as non-invertible architectures on popular image benchmarks (Gomez et al., 2017; Jacobsen et al., 2018). While the change-of-variables formula could be calculated for these models, it is computationally difficult to do so, which prevents their application to generative modeling. The concurrent work of Behrmann et al. (2019) shows how to preserve invertibility in general residual architectures and describes a stochastic approximation of the volume element to allow for high-dimensional generative modeling. Hence their work could be used to define a hybrid model similar to ours, which they mention as area for future work.

6. Experiments

We now report experimental findings for a range of regression and classification tasks. Unless otherwise stated, we used the Glow architecture (Kingma & Dhariwal, 2018) to define the DIGLM's invertible transform and factorized standard Gaussian distributions as the latent prior $p(\boldsymbol{z})$.

6.1. Regression on Simulated Data

We first report a one-dimensional regression task to provide an intuitive demonstration of the DIGLM. We draw x-observations from a Gaussian mixture with parameters $\mu = \{-4, 0, +4\}$, $\sigma = \{.4, .6, .4\}$, and equal component weights. We simulate responses with the function $y = x^3 + \epsilon(k)$ where $\epsilon(k)$ denotes observation noise as a function of the mixture component k. Specifically we chose $\epsilon(k) \sim \mathbb{1}[k \in \{1,3\}]N(0,3) + \mathbb{1}[k = 2]N(0,20)$. We train a B-DIGLM on 250 observations sampled in this way, use standard Normal priors for $p(\boldsymbol{z})$ and $p(\boldsymbol{\beta})$, and three planar flows (Rezende & Mohamed, 2015) to define $f(\boldsymbol{x})$. We compare this model to a Gaussian process (GP) and a kernel density estimate (KDE), which both use squared exponential kernels.

Figure 2(a) shows the predictive distribution learned by the GP, and Figure 2(b) shows the DIGLM's predictive distribution. We see that the models produce similar results, with the only conspicuous difference being the GP has a stronger tendency to revert to its mean at the plot's edges. Figure 2(c) shows the $p(x)$ density learned by the DIGLM's flow component (black line), and we plot it against the KDE (gray shading) for comparison. The single B-DIGLM is able to achieve comparable results to the separate GP and KDE models.

Thinking back to the rejection rule defined in Equation 7, this result, albeit on a toy example, suggests that density thresholding would work well in this case. All data observations fall within $x \in [-5, 6]$, and we see from Figure 2(c) that the DIGLM's generative model smoothly decays to the left and right of this range, meaning that there does not exist an x^* that lies outside the training support but has $p(\boldsymbol{x}^*) \geq \min_{\boldsymbol{x} \in \mathcal{D}} p(\boldsymbol{x})$.

6.2. Regression on Flight Delay Data Set

Next we evaluate the model on a large-scale regression task using the *flight delay* data set (Hensman et al., 2013). The goal is to predict how long flights are delayed based on eight attributes. Following Deisenroth & Ng (2015), we train using the first 5 million data points and use the following $100,000$ as test data. We picked this split not only to illustrate the scalability of our method, but also due to the fact that the test distribution is known to be slightly different from training, which poses challenges of non-stationarity. We evaluate the performance by measuring the root mean squared error (RMSE) and the negative log-likelihood (NLL).

One could model heteroscedasticity in GLMs using random effects (see Appendix C for a discussion), however as a simpler alternative, we follow the solution proposed by Lakshminarayanan et al. (2017) for heteroscedastic regression and set $p(y|\boldsymbol{z})$ to be a two-headed model that predicts both the mean and variance. We use a RNVP transform as the invertible function where the RNVP blocks use 1-layer network with 100 hidden units, and train using Adam optimizer for 10 epochs with learning rate 10^{-3} and batch size 100. To the best of our knowledge, the state-of-the-art (SOTA) performance on this data set is a test RMSE of 38.38 and a test NLL of 6.91 (Lakshminarayanan et al., 2016). Our hybrid model achieves a slightly worse test RMSE of 40.46 but achieves a markedly **better test NLL of 5.07**. We believe that this superior NLL stems from the hybrid model's ability to detect the non-stationarity of the data. Figure 3 shows a histogram of the $\log p(\boldsymbol{x})$ evaluations for the training data (blue bars) and test data (red bars). The leftward shift in the red bars confirms that the test data points indeed have lower density under the flow than the training points.

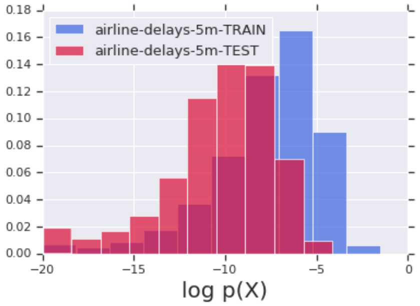


Figure 3. Histogram of $\log p(\boldsymbol{x})$ on the flight delay data set. The leftward shift in the test set (red) shows that our DIGLM model is able to detect covariate shift.

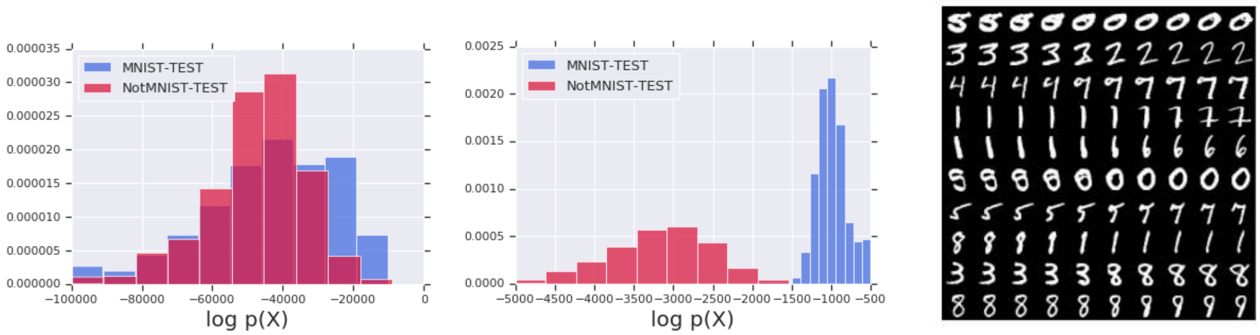
6.3. MNIST Classification

Moving on to classification, we train a DIGLM on MNIST using 16 Glow blocks (1×1 convolution followed by a stack of ACLs) to define the invertible function. Inside of each ACL, we use a 3-layer Highway network (Srivastava et al., 2015) with 200 hidden units to define the translation $t(\cdot; \boldsymbol{\phi}_s)$ and scaling $s(\cdot; \boldsymbol{\phi}_s)$ operations. We use batch normalization in the networks for simplicity in distributed coordination rather than actnorm as was used by Kingma & Dhariwal (2018). We use dropout (Srivastava et al., 2014) before passing $\boldsymbol{z}$ to the GLM, and tune dropout rate on the validation set. Optimization was done via Adam (Kingma & Ba, 2014) with a 10^{-4} initial learning rate for 100k steps, then decayed by half at iterations 800k and 900k.

We compare the DIGLM to its discriminative component, which is obtained by setting the generative weight to zero (i.e. $\lambda = 0$). We report test classification error, NLL, and entropy of the predictive distribution. Following Lakshminarayanan et al. (2017), we evaluate on both the MNIST test set and the NotMNIST test set, using the latter as an out-of-distribution (OOD) set. The OOD test is a proxy for testing if the model would be robust to anomalous inputs when deployed in a user-facing system. The results are shown in Table 1. Looking at the MNIST results, the discriminative model achieves slightly lower test error, but the hybrid model achieves better NLL and entropy. As expected, λ controls the generative-discriminative trade-off with lower values favoring discriminative performance and higher values favoring generative performance.

Model	**MNIST**			**NotMNIST**		
	BPD ↓	error ↓	NLL ↓	BPD ↑	NLL ↓	Entropy ↑
Discriminative ($\lambda = 0$)	81.80*	**0.67%**	0.082	87.74*	29.27	0.130
Hybrid ($\lambda = 0.01/D$)	1.83	0.73%	**0.035**	5.84	2.36	**2.300**
Hybrid ($\lambda = 1.0/D$)	1.26	2.22%	0.081	6.13	**2.30**	**2.300**
Hybrid ($\lambda = 10.0/D$)	**1.25**	4.01%	0.145	**6.17**	**2.30**	**2.300**

Table 1. Results on MNIST comparing hybrid model to discriminative model. Arrows indicate which direction is better.



(a) Discriminative Model ($\lambda = 0$)

(b) Hybrid Model

(c) Latent Space Interpolations

Figure 4. Histogram of $\log p(\boldsymbol{x})$ on classification experiments on MNIST. The hybrid model is able to successfully distinguish between in-distribution (MNIST) and OOD (NotMNIST) test inputs. Subfigure (c) shows latent space interpolations.

Next, we compare the generative density $p(\boldsymbol{x})$ of the hybrid model[1] to that of the pure discriminative model ($\lambda = 0$), quantifying the results in bits-per-dimension (BPD). Since the discriminative variant was not optimized to learn $p(\boldsymbol{x})$, we expect it to have a high BPD for both in- and out-of distribution sets. This experiment is then a sanity check that a discriminative objective alone is insufficient for OOD detection and a hybrid objective is necessary. First examining the discriminative models' BPD in Table 1, we see that it assigns similar values to MNIST and NotMNIST: 81.8 vs 87.74 respectively. While at first glance this difference suggests OOD detection is possible, a closer inspection of the per instance $\log p(\boldsymbol{x})$ histogram—which we provide in Subfigure 4(a)—shows that the distribution of train and test set densities are heavily overlapped. Subfigure 4(b) shows the same histograms for the DIGLM trained with a hybrid objective. We now see conspicuous separation between the NotMNIST (red) and MNIST (blue) sets, which suggests the threshold rejection rule would work well in this case.

Using the selective classification setup described earlier in equation 8, we use $p(y|\boldsymbol{x})$ head when $p(\boldsymbol{x}) > \tau$ where the threshold $\tau = \min_{\boldsymbol{x} \in X_{train}} p(\boldsymbol{x})$ and $p(y)$ estimated using the label counts. The results are shown in Table 1. As expected, the hybrid model exhibits higher uncertainty and achieves better NLL and entropy on NotMNIST. To demonstrate that the hybrid model learns meaningful representations, we compute convex combinations of the latent variables $\boldsymbol{z} = \alpha \boldsymbol{z}_1 + (1 - \alpha)\boldsymbol{z}_2$. Figure 4(c) shows these interpolations in the MNIST latent space.

6.4. SVHN Classification

We move on to natural images, performing a similar evaluation on SVHN. For these experiments we use a larger network of 24 Glow blocks and employ multi-scale factoring (Dinh et al., 2017) every 8 blocks. We use a larger Highway network containing 300 hidden units. In order to preserve the visual structure of the image, we apply only a 3 pixel random translation as data augmentation during training. The rest of the training details are the same as those used for MNIST. We use CIFAR-10 for the OOD set.

Table 2 summarizes the classification results, reporting the same metrics as for MNIST. The trends are qualitatively similar to what we observe for MNIST: the $\lambda = 0$ model has the best classification performance, but the hybrid model is competitive. Figure 5(a) reports the $\log p(\boldsymbol{x})$ evaluations for SVHN vs CIFAR-10. We see from the clear separation between the SVHN (blue) and CIFAR-10 (red) histograms that the hybrid model can detect the OOD CIFAR-10 samples. Figure 5(b) visualizes interpolations in latent space, again showing that the model learns coherent representations. Figure 5(c) shows *confidence versus accuracy* plots (Lakshminarayanan et al., 2017), using the selective classification rule described in Section 3.2, when tested on in-distribution and OOD, which shows that the hybrid model is able to successfully reject OOD inputs.

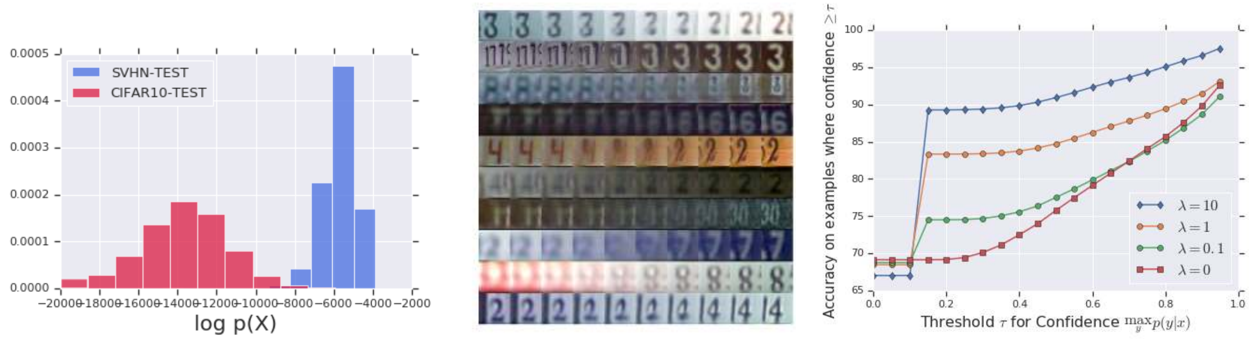
Model	SVHN			CIFAR-10		
	BPD ↓	error ↓	NLL ↓	BPD ↑	NLL ↓	Entropy ↑
Discriminative ($\lambda = 0$)	15.40*	**4.26%**	**0.225**	15.20*	4.60	0.998
Hybrid ($\lambda = 0.1/D$)	3.35	4.86%	0.260	7.06	5.06	1.153
Hybrid ($\lambda = 1.0/D$)	2.40	5.23%	0.253	6.16	4.23	1.677
Hybrid ($\lambda = 10.0/D$)	**2.23**	7.27%	0.268	**7.03**	**2.69**	**2.143**

Table 2. Results on SVHN comparing hybrid model to discriminative model. Arrows indicate which direction is better.

6.5. Semi-Supervised Learning

As discussed in Section 3.1, one advantage of the hybrid model is the ability to leverage unlabeled data. We first performed a sanity check on simulated data, using inter-

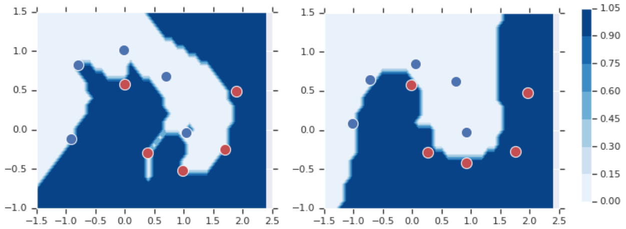
[1] We report results for $\lambda = 0.01/D$; higher values are qualitatively similar.



(a) Histogram of SVHN vs CIFAR-10 densities (b) Latent Space Interpolations (c) Confidence vs. Accuracy

Figure 5. Subfigure (a) shows the histogram of $\log p(\boldsymbol{x})$ on SVHN experiments. The hybrid model is able to successfully distinguish between in-distribution (SVHN) and OOD (CIFAR-10) test inputs. Subfigure (b) shows latent space interpolations. Subfigure (c) shows *confidence versus accuracy* plots and shows that the hybrid model is able to successfully reject OOD inputs.

leaved half moons. Figure 6 shows the decision boundaries when the model is trained without unlabeled data (left) and with unlabeled data (right). The rightmost figure shows a noticeably smoother boundary that better respects the half moon shape.



(a) Fully Supervised (b) With Unlabeled Data

Figure 6. Half Moons Simulation. The decision boundary is shown for the DIGLM trained with just labeled data (left) and with unsupervised data (right). The red and blue points are the instances that have been labeled for each class.

Next we present results on MNIST when training with only 1000 labeled points (2% of the data set) and using the rest as unlabeled data. For the unlabeled points, we maximize $\log p(\boldsymbol{x})$ in the usual way and minimize the entropy for the $p(y|\boldsymbol{x})$ head, corresponding to *entropy minimization* (Grandvalet & Bengio, 2005). We also use virtual adversarial training (VAT) (Miyato et al., 2018), which we found to boost performance. We chose weights on the generative model and on the VAT objective by performing grid sweeps on a validation set, see Appendix B for details. Table 3 shows the results. We see that incorporating the unlabeled data results in an improvement from 6.61% error to 0.99% error, which is competitive with other SOTA approaches such as ladder networks (Rasmus et al., 2015) (0.84%) and GANs (Springenberg, 2015) (1.73%).

Model	MNIST-error ↓	MNIST-NLL ↓
1000 labels only	6.61%	0.276
1000 labels + unlabeled	0.99%	0.069
All labeled	**0.73%**	**0.035**

Table 3. Results of hybrid model for semi-supervised learning on MNIST. Arrows indicate which direction is better.

7. Discussion

We have presented a neural hybrid model created by combining deep invertible features and GLMs. We have shown that this model is competitive with discriminative models in terms of predictive performance but more robust to out-of-distribution inputs and non-stationary problems. The availability of exact $p(\boldsymbol{x}, y)$ allows us to simulate additional data, as well as compute many quantities readily, which could be useful for downstream applications of generative models, including but not limited to semi-supervised learning, active learning, and domain adaptation.

There are several interesting avenues for future work. Firstly, recent work has shown that deep generative models can assign higher likelihood to OOD inputs (Nalisnick et al., 2019; Choi & Jang, 2018), meaning that our rejection rule is not guaranteed to work in all settings. This is a challenge not just for our method but for all deep hybrid models. The DIGLM's abilities may also be improved by considering flows constructed in other ways than stacking ACLs. Recently proposed continuous-time flows (Grathwohl et al., 2019) and invertible residual networks (Behrmann et al., 2019) may prove to be more powerful that the Glow transform that we use, thereby improving our results. Lastly, we have only considered KL-divergence-based training in this paper. Alternative training criteria such as Wasserstein distance could potentially further improve performance.

ACKNOWLEDGEMENTS

We thank Danilo Rezende for helpful feedback.

References

Behrmann, J., Grathwohl, W., Chen, R. T. Q., Duvenaud, D., and Jacobsen, J.-H. Invertible Residual Networks. In *International Conference on Machine Learning (ICML)*, 2019.

Bishop, C. M. Novelty Detection and Neural Network Validation. *IEE Proceedings-Vision, Image and Signal processing*, 141(4):217–222, 1994.

Bishop, C. M. Training with Noise is Equivalent to Tikhonov Regularization. *Neural Computation*, 1995.

Brock, A., Donahue, J., and Simonyan, K. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *International Conference on Learning Representations (ICLR)*, 2019.

Choi, H. and Jang, E. Generative Ensembles for Robust Anomaly Detection. *ArXiv e-Prints*, 2018.

Cordella, L. P., De Stefano, C., Tortorella, F., and Vento, M. A Method for Improving Classification Reliability of Multilayer Perceptrons. *IEEE Transactions on Neural Networks*, 1995.

Deisenroth, M. P. and Ng, J. W. Distributed Gaussian Processes. In *International Conference on Machine Learning (ICML)*, 2015.

Dinh, L., Krueger, D., and Bengio, Y. NICE: Non-Linear Independent Components Estimation. *ICLR Workshop Track*, 2015.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density Estimation Using Real NVP. In *International Conference on Learning Representations (ICLR)*, 2017.

Druck, G., Pal, C., McCallum, A., and Zhu, X. Semi-Supervised Classification with Hybrid Generative/Discriminative Methods. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007.

Efron, B. The Efficiency of Logistic Regression Compared to Normal Discriminant Analysis. *Journal of the American Statistical Association*, 70(352):892–898, 1975.

Fumera, G. and Roli, F. Support Vector Machines with Embedded Reject Option. In *Pattern Recognition with Support Vector Machines*, pp. 68–82. Springer, 2002.

Geifman, Y. and El-Yaniv, R. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Girosi, F., Jones, M., and Poggio, T. Regularization Theory and Neural Networks Architectures. *Neural Computation*, 1995.

Gomez, A. N., Ren, M., Urtasun, R., and Grosse, R. B. The Reversible Residual Retwork: Backpropagation without Storing Activations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Gordon, J. and Hernández-Lobato, J. M. Bayesian Semisupervised Learning with Deep Generative Models. *ArXiv e-Prints*, 2017.

Grandvalet, Y. and Bengio, Y. Semi-Supervised Learning by Entropy Minimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2005.

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., and Duvenaud, D. Scalable Reversible Generative Models with Free-form Continuous Dynamics. In *International Conference on Learning Representations (ICLR)*, 2019.

Hellman, M. E. The Nearest Neighbor Classification Rule with a Reject Option. *IEEE Transactions on Systems Science and Cybernetics*, 1970.

Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian Processes for Big Data. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2013.

Herbei, R. and Wegkamp, M. H. Classification with Reject Option. *Canadian Journal of Statistics*, 2006.

Jaakkola, T. and Haussler, D. Exploiting Generative Models in Discriminative Classifiers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1999.

Jaakkola, T. and Jordan, M. A Variational Approach to Bayesian Logistic Regression Models and their Extensions. In *Sixth International Workshop on Artificial Intelligence and Statistics*, volume 82, pp. 4, 1997.

Jacobsen, J.-H., Smeulders, A. W., and Oyallon, E. i-RevNet: Deep Invertible Networks. In *International Conference on Learning Representations*, 2018.

Jebara, T., Kondor, R., and Howard, A. Probability Product Kernels. *Journal of Machine Learning Research*, 5(Jul): 819–844, 2004.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37(2):183–233, 1999.

Kingma, D. and Ba, J. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*, 2014.

Kingma, D. P. and Dhariwal, P. Glow: Generative Flow with Invertible 1x1 Convolutions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. *International Conference on Learning Representations (ICLR)*, 2014.

Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. Semi-Supervised Learning with Deep Generative Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

Kuleshov, V. and Ermon, S. Deep Hybrid Models: Bridging Discriminative and Generative Approaches. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2017.

Lakshminarayanan, B., Roy, D. M., and Teh, Y. W. Mondrian Forests for Large Scale Regression when Uncertainty Matters. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Lasserre, J. A., Bishop, C. M., and Minka, T. P. Principled Hybrids of Generative and Discriminative Models. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

Maaløe, L., Sønderby, C. K., Sønderby, S. K., and Winther, O. Auxiliary Deep Generative Models. In *International Conference on Machine Learning (ICML)*, 2016.

McCallum, A., Pal, C., Druck, G., and Wang, X. Multi-Conditional Learning: Generative / Discriminative Training for Clustering and Classification. In *AAAI*, 2006.

Miyato, T., Maeda, S.-i., Ishii, S., and Koyama, M. Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

Nalisnick, E., Matsukawa, A., Whye Teh, Y., Gorur, D., and Lakshminarayanan, B. Do Deep Generative Models Know What They Don't Know? In *International Conference on Learning Representations (ICLR)*, 2019.

Nelder, J. A. and Baker, R. J. *Generalized Linear Models*. Wiley Online Library, 1972.

Ng, A. Y. and Jordan, M. I. On Discriminative vs Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2002.

Raina, R., Shen, Y., Mccallum, A., and Ng, A. Y. Classification with Hybrid Generative / Discriminative Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2004.

Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. Semi-Supervised Learning with Ladder Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 3546–3554, 2015.

Rasmussen, C. E. and Williams, C. K. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Rezende, D. and Mohamed, S. Variational Inference with Normalizing Flows. In *International Conference on Machine Learning (ICML)*, 2015.

Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In *International Conference on Machine Learning (ICML)*, 2011.

Riquelme, C., Tucker, G., and Snoek, J. Deep Bayesian Bandits Showdown: An Empirical Comparison of Bayesian Deep Networks for Thompson Sampling. In *International Conference on Learning Representations (ICLR)*, 2018.

Springenberg, J. T. Unsupervised and Semi-Supervised Learning with Categorical Generative Adversarial Networks. *International Conference on Learning Representations (ICLR)*, 2015.

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

Srivastava, R. K., Greff, K., and Schmidhuber, J. Training Very Deep Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Szummer, M. and Jaakkola, T. S. Information Regularization with Partially Labeled Data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2003.

Tulyakov, S., Fitzgibbon, A., and Nowozin, S. Hybrid VAE: Improving Deep Generative Models Using Partial Observations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.