Hybrid Models with Deep and Invertible Features

Eric Nalisnick*, Akihiro Matsukawa*, Yee Whye Teh, Dilan Gorur, Balaji Lakshminarayanan

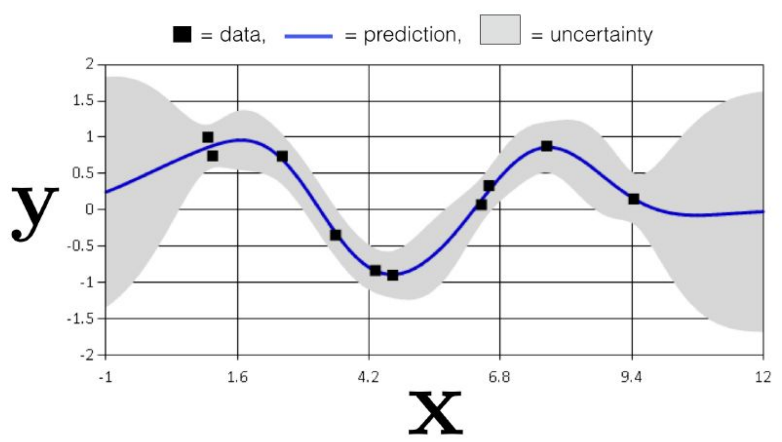
DeepMind

*equal contribution

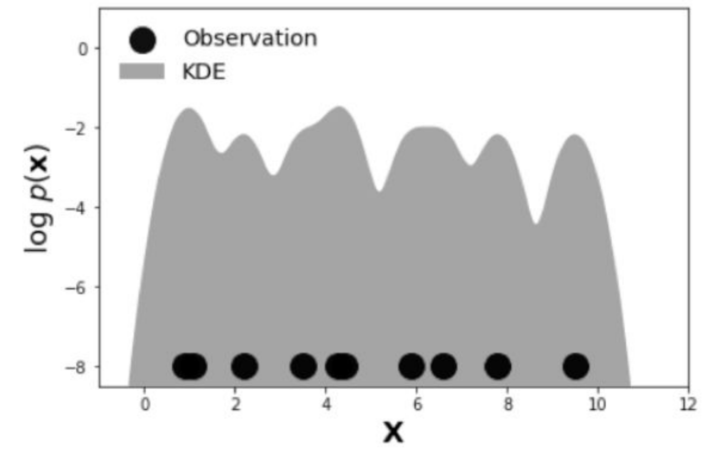
1. Introduction

- Neural networks usually model the conditional distribution $p(\mathbf{y}|\mathbf{x})$, where $\mathbf{y}$ denotes a label and $\mathbf{x}$ features.
- Generative models, on the other hand, represent the distribution over features $p(\mathbf{x})$.
- **Can we efficiently combine the two in a hybrid model of the joint distribution $p(y, x)$?**

Conditional Model



Generative Model



2. Background

Invertible Generative Models (Normalizing Flows)

Invertible generative models (a.k.a. normalizing flows) are a broad class of models defined via the change-of-variables formula. An initial density **p(x)** 'flows' through a series of transformations **f(x)** and morphs into some (usually simpler) prior distribution **p(z)**.

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

Generalized Linear Models (GLMs)

Generalized linear models (GLMs) model the expected response (or label) **y** as a transformation of the linear model $\boldsymbol{\beta}^T\mathbf{z}$ where $\boldsymbol{\beta}$ are parameters and **z** are features (covariates).

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

- **Regression:** $\mathbb{E}[y|\mathbf{z}] = \texttt{identity}(\boldsymbol{\beta}^T\mathbf{z})$
- **Binary Classification**: $\mathbb{E}[y|\mathbf{z}] = \texttt{logistic}(\boldsymbol{\beta}^T\mathbf{z})$

3. Combining Deep Generative Models and Linear Models

We define a model of the joint distribution $p(y, x)$ by instantiating a GLM on the output of a normalizing flow:

$$p(y_n, \boldsymbol{x}_n; \boldsymbol{\theta}) = p(y_n|\boldsymbol{x}_n; \boldsymbol{\beta}, \boldsymbol{\phi})\ \ p(\boldsymbol{x}_n; \boldsymbol{\phi})$$

$$= p(y_n|f(\boldsymbol{x}_n;\boldsymbol{\phi});\boldsymbol{\beta})\ \ p_z(f(\boldsymbol{x}_n;\boldsymbol{\phi})) \left| \frac{\partial \boldsymbol{f}_\phi}{\partial \boldsymbol{x}_n} \right|$$

In practice, we add a weight to the flow terms to tradeoff between predictive and generative behavior:

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

$\log|\partial f/\partial \mathbf{x}|$ Examples

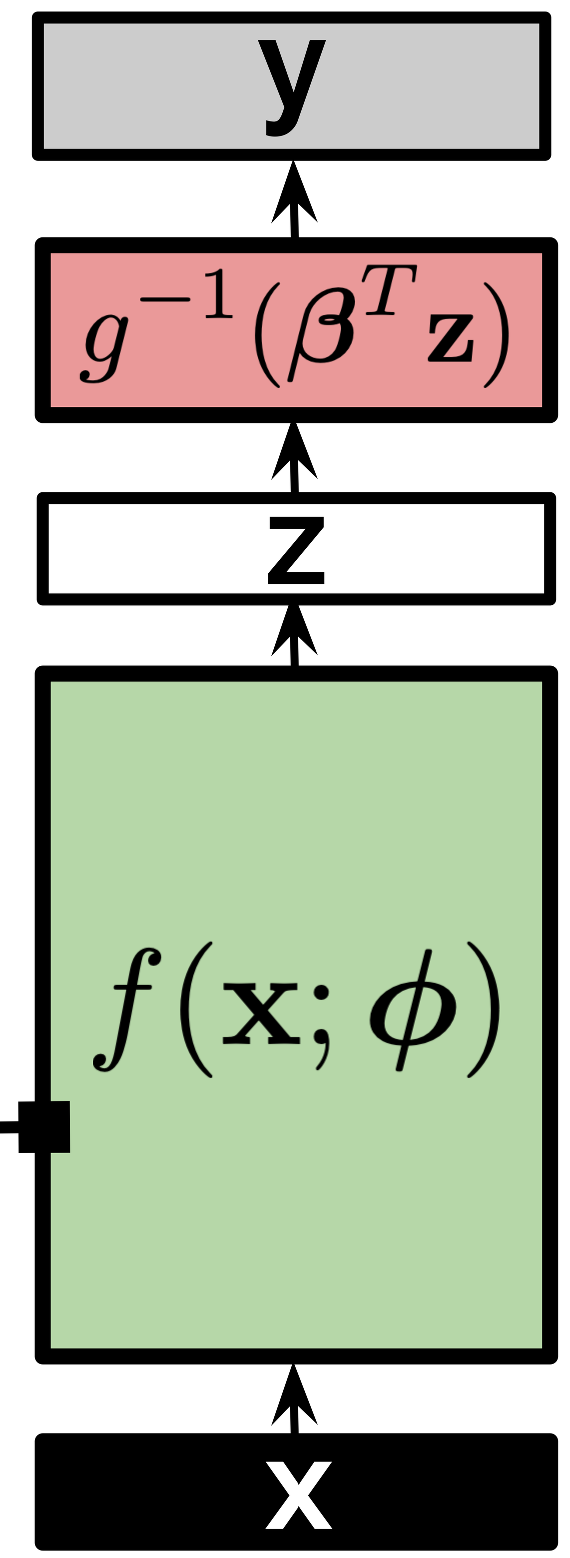
- **Planar:** $= |1 + \mathbf{u}^T f'(\mathbf{w}^T\mathbf{x}+b)\mathbf{w}|$ where $\boldsymbol{w}$, $\boldsymbol{u}$ are parameters.
- **RNVP:** $= \sum_l \sum_d s_{l,d}(\mathbf{x};\boldsymbol{\phi})$ where $\boldsymbol{s}(\,)$ are scaling operations.
- **Glow:** $= \sum_l \sum_d s_{l,d}(\mathbf{x};\boldsymbol{\phi}) + h_l w_l \log|\det \mathbf{W}_l|$, $\boldsymbol{W}$ 1x1 params.

Bayesian treatment: we can place a prior on the parameters of the GLM in order to quantify model *and* data uncertainty.

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

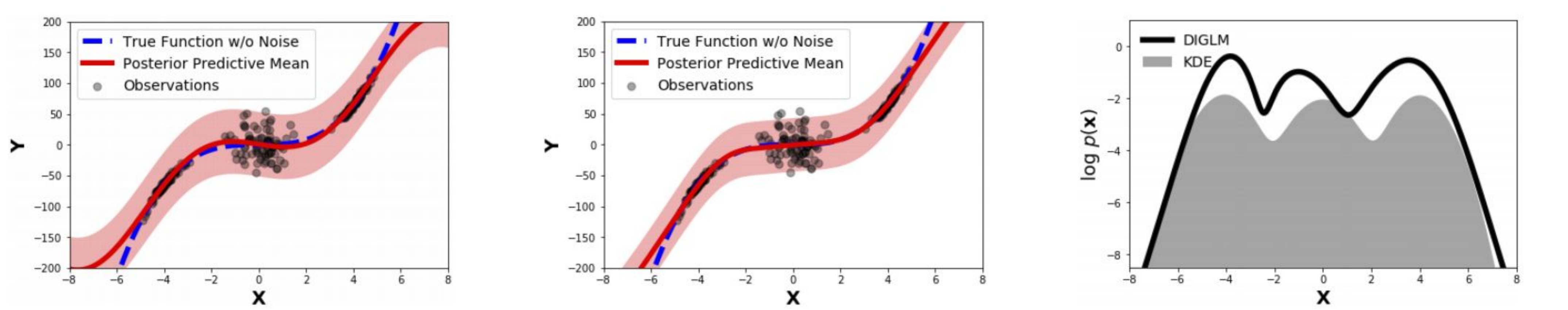
For a Gaussian prior on the GLM, the predictive model can be trained via the closed-form marginal likelihood:

$$\log p(y_n|f(\boldsymbol{x}_n;\boldsymbol{\phi})) = \log \mathrm{N}\left(\boldsymbol{y};\ \boldsymbol{0},\ \sigma_0^2\mathbb{I} + \lambda^{-1}\boldsymbol{Z}_\phi \boldsymbol{Z}_\phi^T\right)$$



Deep Invertible Generalized Linear Model (DIGLM)

4. Simulation

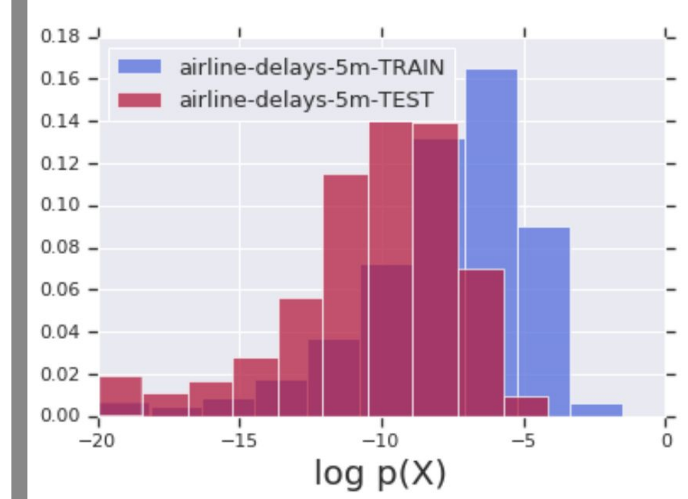


(a) Gaussian Process (b) B-DIGLM $p(y|\boldsymbol{x})$ (c) B-DIGLM $p(\boldsymbol{x})$

1D regression task with heteroscedastic noise. **Subfigure (a)** shows a Gaussian process and **Subfigure (b)** shows our Bayesian DIGLM. **Subfigure (c)** shows ***p(x)*** learned by the same DIGLM (black line) and compares it to a KDE (gray shading).

5. Experiments

Regression on Flight Delay Data Set (N=5 million, D=8)



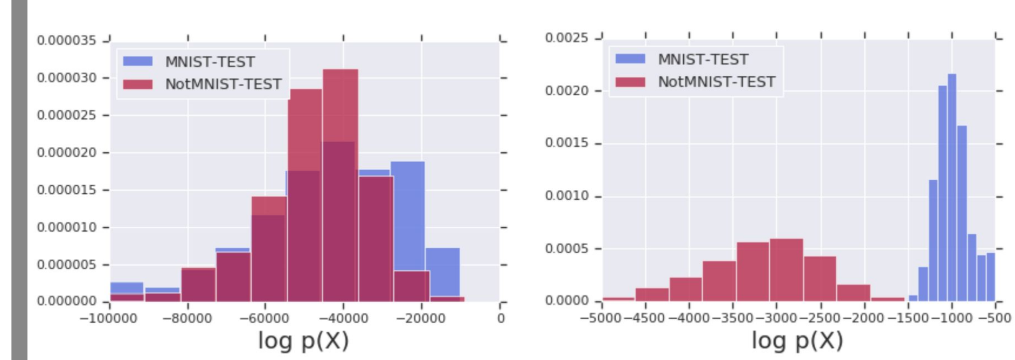
Model	RMSE ↓	NLL ↓
Mondrian Forests (SOTA)	**38.38**	6.91
DIGLM	40.46	**5.07**

- **This data set exhibits covariate shift** between the train and test splits.
- The DIGLM's ***p(x)*** component is able to detect this shift (see left).

Classification on MNIST and SVHN

Model	MNIST BPD ↓	MNIST error ↓	MNIST NLL ↓	NotMNIST BPD ↑	NotMNIST NLL ↓	NotMNIST 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**

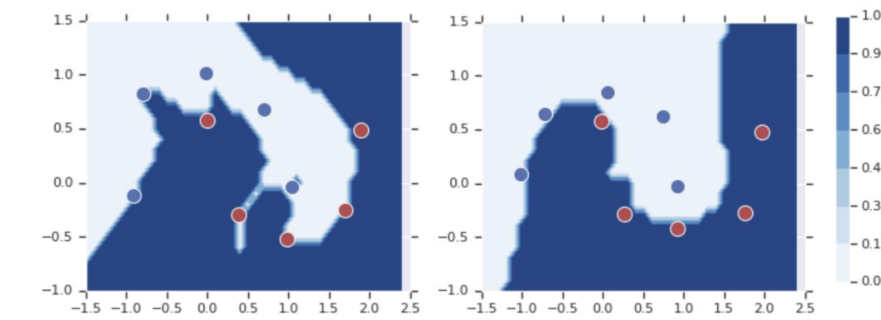
Model	SVHN BPD ↓	SVHN error ↓	SVHN NLL ↓	CIFAR-10 BPD ↑	CIFAR-10 NLL ↓	CIFAR-10 Entropy ↑
Discriminative ($\lambda = 0$)	15.40*	**4.26%**	**0.225**	15.20*	4.60	0.998
Hybrid ($\lambda = 0.1/D$)	3.35	4.86%	0.260	7.06	5.06	1.153
Hybrid ($\lambda = 1.0/D$)	2.40	5.23%	0.253	6.16	4.23	1.677
Hybrid ($\lambda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



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

- λ controls the trade-off between p(y|x) and p(x).
- **Hybrid model is better able to detect the OOD inputs via p(x).**

Semi-Supervised Learning: MNIST and Half Moons



(a) Fully Supervised (b) With Unlabeled Data

Half-moons simulation: the DIGLM leverages unlabeled data to learn a smooth decision boundary (N=10 labeled points).

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**

SSL (VAT) with only 1000 labels (2% of labeled data) achieves <1% error on MNIST

6. Summary

We defined a neural hybrid model that can efficiently compute both predictive p(y|x) and generative p(x) distributions, in a single feed-forward pass, making it a useful building block for downstream applications of probabilistic deep learning.

Paper: https://arxiv.org/abs/1902.02767