Hybrid Models with Deep and Invertible Features

Eric Nalisnick[*][†][‡], **Akihiro Matsukawa**[*], **Yee Whye Teh, Dilan Gorur, Balaji Lakshminarayanan**[†]
DeepMind

1 Introduction

Deep neural networks commonly model conditional distributions of the form $p(y|\boldsymbol{x})$, where y denotes a label and $\boldsymbol{x}$ features or covariates. However, in many applications, modeling just the conditional distribution is insufficient. 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 representation of the data than a conditional one. Models of this form are known as *hybrid models* (Ng & Jordan, 2002; Raina et al., 2004; Lasserre et al., 2006) as they are defined by combining discriminative (i.e. $p(y|\boldsymbol{x})$) and generative (i.e. $p(\boldsymbol{x})$) components. Other benefits of hybrid models include novelty detection (Bishop, 1994), semi-supervised learning (Kingma et al., 2014), information regularization (Szummer & Jaakkola, 2003), and decomposition of uncertainty (Depeweg et al., 2018).

Unfortunately, training a high-fidelity generative model alone is difficult, especially in high dimensions. Hybrid models exacerbate this challenge as a predictive model must be fit in addition to a generative one. It is not surprising then that many previous approaches have had to rely on approximate inference (Alemi et al., 2018; Kingma et al., 2014) for efficient model fitting. In this paper, we propose a neural hybrid model with *exact inference and evaluation* for both $p(y|\boldsymbol{x})$ and $p(\boldsymbol{x})$. To do this, we leverage recent advances in deep invertible generative models (Dinh et al., 2017; Kingma & Dhariwal, 2018). We condition on the features computed by the invertible transform to define a high-capacity neural predictive model (Gomez et al., 2017; Jacobsen et al., 2018). Yet, as the function is invertible, we can employ the change-of-variables formula to compute exact densities of the inputs as well—thereby obtaining a generative model as a byproduct for little additional cost.

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 (COV) formula for *exact* density evaluation:

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

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 parameters $\boldsymbol{\phi}$ are estimated via maximizing the exact log-likelihood $\log p(\boldsymbol{X}; \boldsymbol{\phi})$. See Appendix A for more details on invertible transformations. 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 traditional 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.

[*]Equal contribution.
[†]Corresponding authors: e.nalisnick@eng.cam.ac.uk and balajiln@google.com.
[‡]Work done during an internship at DeepMind.

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

$$\mathbb{E}[y_n|\mathbf{z}_n] = g^{-1}\left(\boldsymbol{\beta}^T\mathbf{z}_n + \beta_0\right) \tag{2}$$

where $\mathbb{E}[y|\mathbf{z}]$ denotes the expected value of y_n, $\boldsymbol{\beta}$ a $\mathbb{R}^d$ vector of parameters, β_0 a scalar bias, $\boldsymbol{z}_n$ the covariates, and $g^{-1}(\cdot)$ a *link function* such that $g^{-1}: \mathbb{R} \mapsto \mu_{y|\boldsymbol{z}}$. A Bayesian GLM could be defined by specifying a prior $p(\boldsymbol{\beta})$ and computing the posterior $p(\boldsymbol{\beta}|\boldsymbol{y}, \boldsymbol{Z})$.

2 Combining Deep Invertible Transforms and Generalized Linear Models

We propose a 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*. Let $\boldsymbol{\theta} = \{\boldsymbol{\phi}, \boldsymbol{\beta}, \beta_0\}$ denote the set of generative and discriminative parameters. The model defines the following joint distribution over a label-feature pair $(\boldsymbol{x}_n, y_n)$:

$$\begin{aligned} p(y_n, \boldsymbol{x}_n; \boldsymbol{\theta}) &= p(\mathbf{y}_n|\boldsymbol{x}_n; \boldsymbol{\phi}, \boldsymbol{\beta}, \beta_0)\ \ p(\boldsymbol{x}_n; \boldsymbol{\phi}) \\ &= p(\mathbf{y}_n|f(\boldsymbol{x}_n; \boldsymbol{\phi}); \boldsymbol{\beta}, \beta_0)\ \ p_z(f(\boldsymbol{x}_n; \boldsymbol{\phi}))\ \left|\frac{\partial \boldsymbol{f}_{\boldsymbol{\phi}}}{\partial \boldsymbol{x}_n}\right| \end{aligned} \tag{3}$$

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,[4] and $p(\mathbf{y}_n|f(\boldsymbol{x}_n; \boldsymbol{\phi}); \boldsymbol{\beta}, \beta_0)$ is a GLM with the latent variables as its input features.

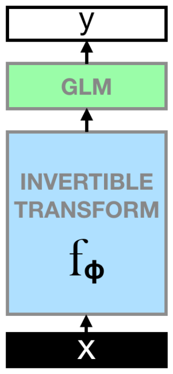


Figure 1: Model Architecture

Figure 1 shows a diagram of the model: we see that the computation pipeline is essentially that of a traditional neural network. The additional costs of obtaining a distribution on $\boldsymbol{x}$ comes entirely from evaluating $p_z(f(\boldsymbol{x}_n; \boldsymbol{\phi}))$, which is $\mathcal{O}(D)$ for factorized distributions, and $|\partial \boldsymbol{f}_{\boldsymbol{\phi}}/\partial \boldsymbol{x}_n|$, which is $\mathcal{O}(KD)$ for RNVP architectures, where K is the number of affine coupling layers and D is the input dimensionality. We term this 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.

$$\mathcal{J}(\boldsymbol{\theta}) = \log p(\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(y_n, \mathbf{x}_n; \boldsymbol{\theta}), \tag{4}$$

which is equivalent to minimizing $D_{\mathrm{KL}}\left(p^*(\boldsymbol{x}, y)\|p_{\boldsymbol{\theta}}(\boldsymbol{x}, y)\right)$. It is possible to give a Bayesian treatment to the parameters,[5] but for simplicity we use point estimates for all the parameters.

In practice we found better performance is obtained by scaling the contribution of $p(\boldsymbol{x})$ to account for the drastic difference in dimensionality between y and $\boldsymbol{x}$. We denote this modified objective as:

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

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). But in our particular case, we can also interpret the down-weighting as follows. Given the likelihood of invertible generative models in equation 1, down-weighting the contribution of $\log p(\mathbf{x}_n; \boldsymbol{\phi})$ 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 $\partial \boldsymbol{f}_\phi/\partial \boldsymbol{x}_n$ derivatives. This optimization objective stands

[4]For simplicitly, we assume a factorial $p(\boldsymbol{z}) = \prod_d p(z_d)$, however it is straightforward to define a richer $p_{\boldsymbol{\eta}}(\boldsymbol{z})$ by defining additional invertible transformations with parameters $\boldsymbol{\eta}$ that transform $\boldsymbol{z}$ into a factorial distribution.

[5] Riquelme et al. (2018) showed that Bayesian linear regression with deep features is competitive on contextual bandit problems, which suggests that it might be more important to capture the weight uncertainty in prediction/decision-making parameters $\boldsymbol{\beta}$ than the weight uncertainty in representation parameters $\boldsymbol{\phi}$. Bayesian inference for GLMs is well-studied and it is possible to use these computational tricks to efficiently capture the uncertainty over $\boldsymbol{\beta}$.

in direct contrast to a long history of gradient-based regularization penalties (Girosi et al., 1995; Bishop, 1995; Rifai et al., 2011). These 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 $\lambda = (1 - \tilde{\lambda})$. If the latent distribution term is, say, a factorized Gaussian, the variance can be scaled by a factor of $1/\tilde{\lambda}$ to introduce regularization only to the Jacobian term.

3 Experiments

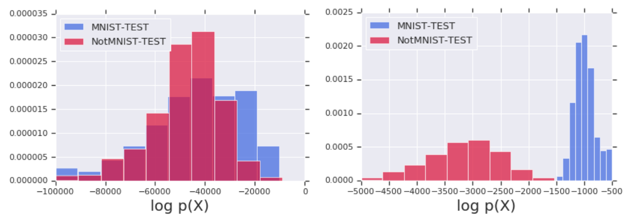
3.1 Classification Experiments on MNIST

We train a DIGLM on the MNIST dataset with GLOW (Kingma & Dhariwal, 2018) as the invertible architecture. We compare the hybrid model to the discriminative model, which is obtained by setting the generative weight $\lambda = 0$. We compare test classification error, negative log-likelihood (NLL) and entropy of the predictive distribution $p(y|\boldsymbol{x})$. Following Lakshminarayanan et al. (2017), we evaluate on both the MNIST test set and the out-of-distribution (OOD) NotMNIST test set. The OOD test is a proxy to test if the system exhibits higher uncertainty on inputs not seen during training data.

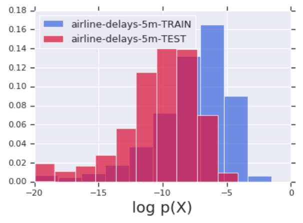
The results are shown in Table 1. The discriminative model achieves slightly lower test error, however the hybrid model achieves better NLL and entropy. Next, we compare generative density $p(\boldsymbol{x})$ for hybrid model as well as discrimative model. Unlike a pure discriminative model which assigns similar density to OOD inputs, the hybrid model outputs lower density $p(\boldsymbol{x})$ for OOD inputs in Figure 2a, hence a decision maker can abstain from trusting the model's predictions when the density is lower.

Model	MNIST-error ↓	MNIST-NLL ↓	NotMNIST-NLL ↓	NotMNIST-Entropy ↑
Discriminative ($\lambda = 0$)	**0.67%**	0.081	29.27	0.1302
Hybrid ($\lambda = 0.01/D$)	0.73%	**0.033**	**10.10**	**0.3695**

Table 1: Results comparing hybrid model to discriminative model.



(a) Histogram of $\log p(\boldsymbol{x})$ for discriminative (left) and hybrid (right).



(b) Regression experiments

Figure 2: Results: histogram of $\log p(\boldsymbol{x})$ on classification and regression experiments.

3.2 Regression Experiments on Flight Delay dataset

Next, we illustrate performance on a regression task using the *flight delay* dataset processed by Hensman et al. (2013), where the goal is to predict how long flights are delayed from eight attributes. We use RNVP flows as the bijector, and evaluate performance by measuring the root mean squared error (RMSE) and NLL. Following Deisenroth & Ng (2015), we train using the first $5M$ (million) data points and use the following 100,000 as test data points. We picked this split of the dataset 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 due to non-stationarity.

To the best of our knowledge, the state-of-the-art performance is RMSE of 38.38 and NLL of 6.91 (Lakshminarayanan et al., 2016). Our hybrid model, which assumes $p(y|\boldsymbol{x})$ to be heteroscedastic Gaussian (Lakshminarayanan et al., 2017), achieves slightly worse RMSE 40.46, but achieves better NLL of **5.07**. The lower NLL shows the usefulness of the hybrid model on non-stationary problems; Figure 2b confirms that the test data points indeed have lower density than the training points.

Discussion We have shown that combining deep invertible features and GLMs can lead to hybrid models which are competitive with discriminative models in terms of predictive performance, but potentially more robust to out-of-distribution inputs[6] and non-stationary problems. There are several extensions possible, one could add a random effects component to model heteroscedasticity, and perform Bayesian inference over prediction parameters, see Appendix B for extensions. 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.

References

Alexander A Alemi, Ian Fischer, and Joshua V Dillon. Uncertainty in the variational information bottleneck. *arXiv preprint arXiv:1807.00906*, 2018.

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

Christopher M Bishop. Training with Noise is Equivalent to Tikhonov Regularization. *Neural Computation*, 7(1):108–116, 1995.

Marc Peter Deisenroth and Jun Wei Ng. Distributed Gaussian processes. In *Proc. Int. Conf. Mach. Learn. (ICML)*, 2015.

S. Depeweg, J. M. Hernández-Lobato, F. Doshi-Velez, and S. Udluft. Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-sensitive Learning. In *ICML*, 2018.

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

Federico Girosi, Michael Jones, and Tomaso Poggio. Regularization Theory and Neural Networks Architectures. *Neural Computation*, 7(2):219–269, 1995.

Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The Reversible Residual Retwork: Backpropagation without Storing Activations. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2214–2224, 2017.

James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. In *Conf. Uncertainty Artificial Intelligence (UAI)*, 2013.

Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. i-RevNet: Deep Invertible Networks. *arXiv preprint arXiv:1802.07088*, 2018.

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

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pp. 3581–3589, 2014.

Balaji Lakshminarayanan, Daniel M Roy, and Yee Whye Teh. Mondrian forests for large scale regression when uncertainty matters. In *Int. Conf. Artificial Intelligence Stat. (AISTATS)*, 2016.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Julia A Lasserre, Christopher M Bishop, and Thomas P Minka. Principled hybrids of generative and discriminative models. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pp. 87–94. IEEE, 2006.

Andrew McCallum, Chris Pal, Gregory Druck, and Xuerui Wang. Multi-conditional learning: Generative/discriminative training for clustering and classification. In *AAAI*, pp. 433–439, 2006.

[6]In (Nalisnick et al., 2018), we show that deep generative models can sometimes assign higher density to out-of-distribution inputs, so relying on $p(\boldsymbol{x})$ from hybrid models to detect inputs similar to training data is not always guaranteed to work.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do Deep Generative Models Know What They Don't Know? *ArXiv e-Print arXiv:1810.09136*, 2018.

John Ashworth Nelder and R Jacob Baker. *Generalized Linear Models*. Wiley Online Library, 1972.

Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *Advances in neural information processing systems*, pp. 841–848, 2002.

Rajat Raina, Yirong Shen, Andrew Mccallum, and Andrew Y Ng. Classification with hybrid generative/discriminative models. In *NIPS*, pp. 545–552, 2004.

Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, 2011.

Carlos Riquelme, George Tucker, and Jasper Snoek. Deep Bayesian bandits showdown: An empirical comparison of Bayesian deep networks for Thompson sampling. *arXiv preprint arXiv:1802.09127*, 2018.

Martin Szummer and Tommi S Jaakkola. Information Regularization with Partially Labeled Data. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

Supplementary Material

A Review of R-NVP and GLOW

Dinh et al. (2017) defines R-NVP transformations through repeated composition of the following operations:

1. **Splitting**: For a given input $\mathbf{x}$, the first step is to split it at dimension d into two separate vectors $\mathbf{x}_{:d}$ and $\mathbf{x}_{d:}$ (assuming Python list syntax).
2. **Copying and Affine Transformations**: Given the split $\{\mathbf{x}_{:d}, \mathbf{x}_{d:}\}$, we compute two new vectors
$$\mathbf{h}_1 = \mathbf{x}_{:d} \quad \text{(copy)}, \qquad \mathbf{h}_2 = t(\mathbf{x}_{:d}; \boldsymbol{\phi}_t) + \mathbf{x}_{d:} \odot \exp\{s(\mathbf{x}_{:d}; \boldsymbol{\phi}_s)\} \quad \text{(affine)} \tag{6}$$
where $t(\mathbf{x}_{:d}; \boldsymbol{\phi}_t)$ and $s(\mathbf{x}_{:d}; \boldsymbol{\phi}_s)$ are translation and scaling operations with no restrictions on their functional form. We can compute them with neural networks that take as input $\mathbf{x}_{:d}$, the other half of the original vector, and since $\mathbf{x}_{:d}$ has been preserved (copied) by the first operation, no information is lost that would jeopardize invertibility.
3. **Shuffling**: Lastly, the new representations $\{\mathbf{h}_1, \mathbf{h}_2\}$ are ready to be either (#1) treated as output or (#2) fed into another R-NVP transform. If the latter, then the elements should be shuffled such that most of the elements in $\mathbf{h}_1$ are not again copied but rather subject to the affine transformation.

These three steps are composed a user-specified number of times. Crucially, the Jacobian of these operations is efficient to compute via:
$$\log \left| \frac{\partial f}{\partial \mathbf{x}} \right| = \log \exp \left\{ \sum_{l=1}^{L} s_l(\mathbf{x}_{:d}; \boldsymbol{\phi}_s) \right\} = \sum_{l=1}^{L} s_l(\mathbf{x}_{:d}; \boldsymbol{\phi}_s). \tag{7}$$
The Jacobian simplifies to the sum of all the scale transformations.

B Extensions

B.1 Mixed Effects Model

To model heteroscedastic noise, we can also add "random effects" to the model at the latent level. The model is then:
$$\mathbb{E}[y_n | \mathbf{x}_n] = g^{-1}\left(\boldsymbol{\beta}^T f(\mathbf{x}_n; \boldsymbol{\phi}) + \mathbf{u}^T \mathbf{a}_n \right), \quad \mathbf{u} \sim p(\mathbf{u}), \quad \mathbf{a}_n \sim p(\mathbf{a}) \tag{8}$$
where $\mathbf{a}_n$ is a vector of random effects associated with the fixed effects $\mathbf{x}_n$, and $\mathbf{u}$ are the corresponding parameters in the GLM. Note that Depeweg et al. (2018) also use random effects model to handle heteroscedastic noise, but add them at the input level.

B.2 Bayesian inference

As discussed earlier, it might be more important to capture weight uncertainty on prediction/decision-making parameters $\boldsymbol{\beta}$ than the weight uncertainty in representation parameters $\boldsymbol{\phi}$. For regression problems, it is possible to integrate out $\boldsymbol{\beta}$ exactly, as was also pointed by Riquelme et al. (2018). For classification problems, it is not possible to integrate out $\boldsymbol{\beta}$ exactly, but we can use efficient approximation techniques (Laplace approximation, variational approximations, etc) as the posterior over $\boldsymbol{\beta}$ is log-concave when the link function corresponds to a likelihood $p(\mathbf{y}_n | \boldsymbol{x}_n; \boldsymbol{\phi}, \boldsymbol{\beta}, \beta_0)$ within the exponential family, and the prior is set to the conjugate prior for this exponential family.