
Flow Posterior for Uncertainty Estimation

Neil Xu^{*1} Shalom Yiblet^{*1}

Abstract

Deep learning has produced models with high levels of accuracy on a wide variety of tasks. These models, however, are also often overconfident on predictions for noisy data or regions of the feature space that are not part of the training set. We propose a model utilizing normalizing flows for predicting the error that a deep learning model will have in regression tasks in a generalized setting. Previous models that predicted error either did so directly, or only predicted the parameters of a simple distribution which did not fully reflect the error distribution of model accurately. By fully modeling the error distribution of model for an input, our flow error model provides deeper insight into the error landscape of the real data with respect to the model. We also show that our flow model achieves greater likelihood than a baseline Gaussian model of the error distribution of a neural network regression model.

1. Introduction

Despite the predictive accuracy of deep neural networks on variety of learning tasks, a remaining challenge for DNNs is to determine how well they generalize. The reliability of DNNs on any data has become of greater importance given their effectiveness in increasingly critical tasks such as autonomous driving (Chen et al., 2015) and medical diagnosis (Liu et al., 2014). Since current DNNs have shown a tendency to overfit on the training data, it may output significantly incorrect predictions on unseen examples. Consequently, quantifying the confidence of DNNs becomes an important metric to determine when the DNN’s prediction is incorrect.

To tackle this problem, there has been a significant amount

^{*}Equal contribution ¹School of Computer Science, Carnegie Mellon University, Pittsburgh, USA. Correspondence to: Neil Xu <ziyux@andrew.cmu.edu>, Shalom Yiblet <syiblet@andrew.cmu.edu>.

of work in recent years using a Bayesian deep learning framework to encode uncertainty into the model. A common approach for Bayesian deep learning models is to parameterize a distribution over the weights, and use variational inference (Graves, 2011; Kingma & Welling, 2013) for training. These models, however, suffer from utilizing Monte Carlo methods for optimization, which large increases the training time and often decreases the performance to the the increased variance from learning against gradients of samples.

As an alternative to Bayesian approach of learning a distribution over weights, we can also create models that directly predict the uncertainty of the data. These models, however, often predict simple distributions over their outputs, and consequently may fail to include the true uncertainty relationship in the data. This is primarily a problem when the model is required to model a continuous probability distribution in its output. In classification, the softmax output of many classification models have the capacity to model the entire range of distributions over discrete classes, and does not suffer from this lack of capacity. For regression, however, current approaches often uses simple distributions, such outputting a Gaussian distribution by predicting both its mean and variance parameters (Lakshminarayanan et al., 2017). A key feature that a Gaussian fails to model is any form of multimodality.

To address this deficiency in the mapping capability of the output, we expand the range of distributions that our network is capable of learning by incorporating normalizing flows (Rezende & Mohamed, 2015) into our distribution representation. We show that by having a more flexible family of posteriors, we are able to more accurately model the uncertainty of a model and that the flow posterior is capable of representing output distributions that the standard Gaussian distribution is incapable of. Furthermore, we derive theoretical bounds for analyzing the variance of the learned distribution that can be computed in a deterministic fashion, avoiding the computational cost of Monte Carlo. Thus our paper makes the following contributions:

1. Provides a DNN architecture for utilizing normalizing flows in learning a more complex posterior that can predict model uncertainty.
2. Empirically demonstrating the ability of this normal-

izing flow model to more accurately predict the uncertainty or performance of the model on a given input, and also more closely reflect the true noise distribution in the data.

3. Provide theoretical bounds of the variance of the output distribution

2. Related Work

Bayesian deep learning (Neal, 1996) has become a primary formulation for modeling uncertainty in DNNs. In Bayesian learning a prior distribution is assumed over the weights of a DNN, and a posterior distribution over the weights is predicted given an input. The weight distribution is then used to quantitatively infer an uncertainty value. A significant challenge in Bayesian learning is performing approximate inference of the posterior distribution.

Historically, Hamiltonian Monte Carlo inference for Bayesian neural networks was first introduced in (Neal, 1996). Modern inference methods, however, generally try to compute the approximate posterior with deterministic methods to avoid the computational expense of MCMC methods. This line of research was introduced by (Graves, 2011), which proposed a variational inference using a factorized Gaussian posterior. This method, however, formulated a biased estimator of the posterior. Thus, (Blundell et al.) formulates a unbiased estimator by using Monte Carlo gradients and the reparameterization trick (Kingma et al., 2015) to still backpropagate losses defined on samples from the weight distribution to the variational parameters. (Hernandez-Lobato & Adams, 2015) builds upon this work by backpropagating approximated probabilities of the loss to the variational parameters. In addition to deriving new paradigms, many common DNN regularization techniques can be reinterpreted under the Bayesian framework. Dropout (Srivastava et al., 2014) and batch normalization (Ioffe & Szegedy, 2015) have both been shown to be forms of Bayesian DNNs by (Gal & Ghahramani, 2015) and (Teye et al., 2018) respectively. More recently, (Louizos & Welling, 2017) proposes utilizing normalizing flows to model the weight distribution in Bayesian neural networks, and is the work most similar to ours.

There has also been work done in directly modeling the uncertainty of model prediction. This typically is done by ensembling together the predictions of many trained models. In a reinforcement learning settings, (Osband et al., 2016) uses bootstrapping to train multiple model heads, from which predictions are sampled. On the other hand, (Lakshminarayanan et al., 2017) demonstrate that ensembled DNNs that each predict uncertainty by outputting the parameters of a Gaussian posterior over the target have comparable performance to Bayesian learning approaches while avoiding the difficulty of approximating the weight posterior.

We consider our work an extension of (Lakshminarayanan et al., 2017) that replaces the Gaussian posterior with a more complex flow posterior. There has also been work in using data that has label disagreements, and then directly training a model on the uncertainty in the labels (Raghu et al., 2018).

Normalizing flows (Rezende & Mohamed, 2015) have been introduced in recent years initially as a method for creating more complex posteriors, but computationally efficient, in variational inference. Inverse autoregressive flows (IAF) (Kingma et al., 2016) develop an autoregressive technique for creating distributions more flexible in higher dimensions. IAF has been used successfully to perform model distillation for audio synthesis (Oord et al., 2017), and image synthesis (Kingma & Dhariwal, 2018), and text generation (Yang et al., 2017). Other flow types that have had success include masked autoregressive flows (Papamakarios et al., 2017) and most recently, neural autoregressive flows (Huang et al., 2018), which unite existing normalizing flows under a more general framework based upon neural networks (?). Our work aims to introduce these modern flow techniques to the domain of uncertainty estimation.

3. Background

3.1. Normalizing Flows

Normalizing flows (Rezende & Mohamed, 2015) are a method through which we can learn a best fitting distribution through learning a number of invertible transformations from a simple distribution. This allows us to parameterize a distribution that is more complicated than a Gaussian and calculate the probability density and sample from it efficiently.

Let f be $\mathbb{R}^d \rightarrow \mathbb{R}^d$ and invertible, and d be the dimensionality of \mathbf{z} , a random variable that has been drawn from a known distribution p that is computationally tractable. We sample from the flow distribution q simply by calculating $\mathbf{z}' = f(\mathbf{z})$.

To compute the probability density of q , we utilize f^{-1} , the inverse of f :

$$q(\mathbf{z}') = p(\mathbf{z}) \left| \det \frac{\delta f^{-1}}{\delta \mathbf{z}'} \right| = p(\mathbf{z}) \left| \det \frac{\delta f}{\delta \mathbf{z}} \right|^{-1} \quad (1)$$

Thus, we can compose k layers of flows together to create a more complicated distribution. For invertible functions f_1, \dots, f_k , we can sample from the distribution by applying the functions sequentially to a sample from the base distribution p :

$$\mathbf{z}_k = f_k(\dots f_1(\mathbf{z})) \sim q_k(\mathbf{z}_k) \quad (2)$$

Thus, to perform inference about statistics of the learned

flow distribution, we can easily perform Monte Carlo integration by using this sampling technique.

Consequently, we can also derive a recursive formulation for computing the log-likelihood of q_k :

$$\ln q_k(\mathbf{z}_k) = \ln q_0(\mathbf{z}_0) - \sum_{k=1}^K \ln \left| \det \frac{\delta f_k}{\delta \mathbf{z}_{k-1}} \right| \quad (3)$$

In a deep learning setting, this log-likelihood can be a optimizable objective, since each f_i can also have a set of parameters. Thus maximization of the log-likelihood can simply be accomplished by backpropagating the the gradient to the parameters of each f_i . This makes training using standard neural network frameworks relatively easy since it follows the same backpropagation paradigm.

In the following two sections we will discuss the types of normalizing flows we use in order to construct the network. For regression problems, the Real Nonvolume Preserving Flow (RealNVP) (Dinh et al., 2016) and the Masked Autoregressive Flow (MAF) (Papamakarios et al., 2017) are commonly the best performing normalizing flow paradigms.

3.2. Masked Autoregressive Flows

Different types of normalizing flows are fundamentally different families of models that characterize the f function. A common paradigm in density estimation is to use some form of autoregression i.e. parameterizing the flow transformation in some order indexed by the index set $\{i_1, \dots, i_d\}$ (although typically in the indexing is just the order the dimensions are in). For output random variable \mathbf{z}' , an autoregressive flow function results in:

$$\mathbf{z}'_{i_k} = g(\mathbf{z}_{i_{k-1}}, \dots, \mathbf{z}_{i_0}) \quad (4)$$

Specifically in Masked Autoregressive Flows (MAF) (Papamakarios et al., 2017), the autoregressive function used for the flow distribution is the Masked Autoencoder for Distribution Estimation (MADE) estimator introduced by (Germain et al., 2015). For each dimension index i in \mathbf{z} being sampled, we calculate the function as:

$$\mathbf{z}_i = \mathbf{u}_i * \alpha_i + \mu_i \quad (5)$$

where \mathbf{u} is in the input random variable. We compute α_i, μ_i to be:

$$\alpha_i = f_\alpha(\mathbf{z}_{i-1}, \dots, \mathbf{z}_0) \quad (6)$$

$$\mu_i = f_\mu(\mathbf{z}_{i-1}, \dots, \mathbf{z}_0) \quad (7)$$

where f_α, f_μ are implemented as autoregressive neural network functions. For computational efficiency, the autoregressive property is implemented as a mask over a feedforward network that zeroes out the connections that don't exist

between the dimensions of the output. The autoregressive nature of the algorithm means that evaluating \mathbf{z}_i is expensive. It cannot be done in parallel since each \mathbf{z}_i is dependent on all previous \mathbf{z}_j where for all $j < i$.

Fortunately, computing inverse Jacobian does not require this expensive autoregressive loop along the event dimension. Thus at training time, we are able to learn while side-stepping this issue.

3.3. RealNVP

The RealNVP (Dinh et al., 2016) flow on the other hand does not require this expensive autoregressive computation. It is a far simpler model.

For an output vector \mathbf{z} of dimension d , and an input vector \mathbf{u} of the same dimension. The transformation can be written as:

$$\mathbf{z}_{1:k} = \mathbf{u}_{1:k} \quad (8)$$

$$\mathbf{z}_{k+1:d} = \mathbf{u}_{k+1:d} * \sigma(\mathbf{u}_{1:k}) + \mu(\mathbf{u}_{1:k}) \quad (9)$$

where u , and z are arbitrary neural network that output a $d - k$ dimensional value. Effectively, it copies the first k dimensions while scaling and shifting the other remaining dimensions as a function on the first k .

While the flow is still autoregressive in the sense that it acts on it's own values. It is a far simpler operation than the MAF, and thus also far less computationally expensive and more parallelizable.

3.4. Input Convex Neural Networks

In order to apply the bounds on the variance we must ensure that the network is convex with respect to it's input vectors. (Amos et al., 2016) have applied input convex to various different hard problems in machine learning. They empirically therefore argue that despite the fact that constraining the DNN to be input convex greatly reduces the size of function space, the DNN can still nevertheless learn to perform well. For some of the problems they solved using their input convex neural network, their methods matched the performance of the state of the art.

3.5. Chernoff Variance Inequality

For a standard normal random variable Z and a transformation $g : \mathbb{R} \rightarrow \mathbb{R}$ that is absolutely continuous with a derivative g' such that $\mathbb{E}[g'(X)]^2 \leq \infty$, (Chernoff, 1981) shows that

$$\text{Var}(Z) \leq \mathbb{E}[g'(Z)]^2 \quad (10)$$

with equality if and only if $g(X)$ is linear.

Chen (Chen, 1982) expanded this inequality to the multivariate case. For a series of standard normal variables $\{X_1, X_2, X_3 \dots X_n\}$ and $\{g, g_1, g_2, g_3 \dots g_n\}$ be real measurable functions on \mathbb{R}^n where

$$\begin{aligned} &g(x_1, x_2, \dots, x_n) \\ &= \int_0^{x_i} g_i(x_1, \dots, x_{i-1}, t, x_{i+1}, \dots, x_n) dt \\ &+ g(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n) \end{aligned}$$

almost everywhere. Then

$$\text{Var}[g(X_1, \dots, X_n)] \leq \sum_{i=1}^n \mathbb{E}[g_i(X_1, \dots, X_n)]^2$$

This set of functions g_i with this specific relationship satisfied by the partial derivatives

$$\frac{\partial}{\partial X_i} g(X_1, X_2, \dots, X_n)$$

4. Methods

4.1. Defining The Task

For a regression problem with input \mathbf{x} and true value \mathbf{y} . We write the regression model as r where $r(\mathbf{x}) = \hat{\mathbf{y}}$

Given this formulation, our task is to at best infer the variance

$$\text{Var}[\mathbf{y} - \hat{\mathbf{y}} \mid \hat{\mathbf{y}}, \mathbf{x}] \quad (11)$$

We approach this problem, not by some direct regression task on this variance, but through probabilistic inference. We attempt to estimate the distribution $p(\mathbf{y} - \hat{\mathbf{y}} \mid \mathbf{y}, \mathbf{x})$

By constructing a parametric distribution (via normalizing flows):

$$q_\theta(\mathbf{y} - \hat{\mathbf{y}} \mid \mathbf{y}, \mathbf{x}) \quad (12)$$

We can approximate p by minimizing against the conditional cross entropy:

$$\mathbb{E}_{\mathbf{y} - \hat{\mathbf{y}} \sim p}[-\log q_\theta(\mathbf{y} - \hat{\mathbf{y}} \mid \mathbf{y}, \mathbf{x}) \mid \mathbf{y}, \mathbf{x}] \quad (13)$$

Since the cross entropy is lower bounded by the entropy we know that this loss is minimized when $q_\theta(\mathbf{y} - \hat{\mathbf{y}} \mid \mathbf{y}, \mathbf{x}) = p(\mathbf{y} - \hat{\mathbf{y}} \mid \mathbf{y}, \mathbf{x})$ almost everywhere.

Since it is intractable to directly minimize against this objective, we instead approximate the objective through Monte Carlo sampling.

4.2. Model Structure

Our model for regression error estimation is made out of two components - the *regression model* itself and the *error model* for estimating the distribution of the true error of the regression model's prediction. Our regression model is a simple perceptron model trained with mean squared error.

The error model is composed of two parts - a perceptron layer h that takes in x, \hat{y} as input and converts them into μ, σ that parameterize a Gaussian:

$$\mu = h_\mu(x, \hat{y}) = \text{softplus}(W_\mu[x; \hat{y}] + b_\mu) \quad (14)$$

$$\sigma = h_\sigma(x, \hat{y}) = \text{softplus}(W_\sigma[x; \hat{y}] + b_\sigma) \quad (15)$$

and the stacked flow layers that transform the Gaussian parameters into parameters of a flow distribution.

The error model is consequently agnostic to the regression model - this error model can be separately trained added to any regression model and learn its error distribution.

4.3. Deriving The Variance

With the objective, and the model in tow, we now will develop the core contributions of this paper. Given that we have this probabilistic model q_θ we now develop three different ways of constructing the variance in the following sections:

4.3.1. MONTE CARLO INTEGRATION

Above all this is by far the most accurate and also the most expensive way to estimate the variance. Using approximation methods, we estimate the expectation:

$$\text{Var}[\mathbf{y} - \hat{\mathbf{y}} \mid \hat{\mathbf{y}}, \mathbf{x}] = \int_{\Omega} (\mathbf{y} - \mathbb{E}[\mathbf{y}])^2 q_\theta(\mathbf{y} - \hat{\mathbf{y}} \mid \mathbf{y}, \mathbf{x}) d\mathbf{y} \quad (16)$$

By sampling from q_θ and taking the sum. To our knowledge, this is the first time someone has used normalizing flows with Monte Carlo estimation to derive variance estimates. Yet nevertheless, we use this method as a kind of baseline to compare our upper and lower bounds on the variance.

4.3.2. THE JENSEN'S INEQUALITY LOWER BOUND

For an original random variable

$$N \sim \mathcal{N}(\mu, \Sigma)$$

the random variable of the flow output is $Y = f_\theta(N)$. where f_θ is the parametric flow transformation.

In order to construct this lower bound, we rely on a simplifying assumption. We assume that $\mathbb{E}[f_\theta(N)]$ which approximates $\mathbb{E}[Y - \hat{Y}]$ is equal to zero. While this assumption

seems to be prohibitive, in most cases it's a fair assumption to make. For a regressor trained against MSE for example, the optimum point is when the regressor is an unbiased estimator of Y .

With that assumption we can rewrite the variance:

$$\begin{aligned} \text{Var}[f_\theta(N) \mid \hat{y}, x] &= \mathbb{E}[f_\theta(N)^2 \mid \hat{y}, x] - \mathbb{E}[f_\theta(N) \mid \hat{y}, x]^2 \\ &= \mathbb{E}[f_\theta(N)^2 \mid \hat{y}, x] \end{aligned}$$

If $f_\theta(N)^2$ is a convex function, we can apply Jensen's inequality, and construct an analytic bound on the variance.

$$\text{Var}[f_\theta(N) \mid \hat{y}, x] = \mathbb{E}[f_\theta(N)^2 \mid \hat{y}, x] \quad (17)$$

$$\geq f_\theta(\mathbf{E}[N])^2 \quad (18)$$

$$\geq f_\theta(\mu)^2 \quad (19)$$

In the following sections we will explain exactly how to ensure that $f_\theta(N)^2$ is convex in N .

4.3.3. CHERNOFF VARIANCE INEQUALITY UPPER BOUND

We directly use Chen's bound (Chen, 1982):

$$\text{Var}[g(X_1, \dots, X_n)] \leq \sum_{i=1}^n \mathbb{E}\left[\frac{\partial}{\partial X_i} g(X_1, \dots, X_n)\right]^2$$

If the partial derivatives

$$\frac{\partial}{\partial X_i} g(X_1, \dots, X_n)$$

are concave, we can apply Jensen's inequality.

$$\text{Var}[g(X_1, \dots, X_n)] \leq \sum_{i=1}^n \frac{\partial}{\partial X_i} g(\mathbb{E}[X_1], \dots, \mathbb{E}[X_n])^2$$

4.4. Constructing a Dense Convex Neural Network

Core to the upper and lower bounds we achieve is a construction of the a convex dense neural network. For brevity, we will provide a proof sketch showing that a feedforward neural network with nonnegative kernel weights $W^{(i)}$

$$W^{(i)} \geq 0$$

and relu activations is convex and nondecreasing.

First, let's discuss the terminology: Our network is made up of l layers. where the output of of the i 'th layer:

$$a_i = \text{relu}(W^{(i)}a_{i-1} + b^{(i)}) \quad (20)$$

And the whole dense network is written as the vector valued function

$$f(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}) \dots f_n(\mathbf{x})]^T \quad (21)$$

Thus when we say that the vector valued function is convex, we mean each value $f_i(\mathbf{x})$ is convex with respect to it's input.

The proof follows inductively by layer. let $a_0 = \mathbf{x}$, note that a_0 is thus trivially convex since the identity function is convex.

For the inductive case, we need to prove that a_i is convex and nondecreasing given that a_{i-1} is convex and nondecreasing.

Note that since, $W^{(i)}a_{i-1} + b$ is convex it is affine, and since we constrain the kernel weights $w^{(i)}$ to be nonnegative the function is also nondecreasing. The outward relu operation is convex, and is also nondecreasing in it's input. Therefore, it's composition must also be convex and nondecreasing.

This shows inductively that for each i , $f_i(\mathbf{x})$ is convex and nondecreasing. We can apply the same reasoning to construct concave dense feedforward neural networks the only difference being that we replace the activation function with

$$f(x) = \min(0, -x)$$

$$W^{(i)} \leq 0$$

4.5. Constructing a Convex MAF

$$\mathbf{z}_i = \mathbf{u}_i * \alpha_i + \mu_i \quad (22)$$

In the MAF we use dense convex neural networks, for α_i and μ_i . To show that the MAF is convex, in each dimension it's sufficient to show this is convex via induction.

The base case is simple:

$$\mathbf{z}_1 = \mathbf{u}_1 * \alpha_1 + \mu_1 \quad (23)$$

μ_1 and α_1 are not functions of the input. Therefore they are affine and convex.

The inductive case then uses the fact that we know α_i and μ_i is convex. Since the addition of convex functions is still

convex we simply need to show that the product $u_i * \alpha_i$ is still convex. This is not true generally, however if we constrain u_i to be positive then the result is true, since one can write that operation as the composition of nondecreasing convex function with a convex function.

4.6. Constructing a Convex RealNVP

The same technique of limiting the input X to be nonnegative can be used to derive a convex RealNVP.

Recall the transformation (copied over here for ease):

$$\begin{aligned} \mathbf{z}_{1:k} &= \mathbf{u}_{1:k} \\ \mathbf{z}_{k+1:d} &= \mathbf{u}_{k+1:d} * \sigma(\mathbf{u}_{1:k}) + \mu(\mathbf{u}_{1:k}) \end{aligned}$$

We use dense convex networks for σ and μ , so it is again sufficient to show that $\mathbf{u}_{k+1:d} * \sigma(\mathbf{u}_{1:k})$ is convex. If $u_{k+1:d}$ is nonnegative this operation is convex. For the same reason constructing the convex MAF, we can write this product term as a composition of nondecreasing convex functions which we know constructs a convex function.

4.7. Concavity

For brevity, we'll only provide a sketch of how to construct the concave version of the MAF and RealNVP, since it borrows many of the same techniques as the concavity proof in the sections above. To make the functions concave, we use the fact that a concave nonincreasing function composed with a concave function results in a concave function. Thus instead of enforcing the variables to be nonnegative we constraint them to be nonpositive, while using the concave dense network construction for the scale and shift networks.

5. Results

5.1. Experiment setup

A consequence of using autoregressive normalizing flow techniques is that they depend on the target being multivariate. We emulate a similar setup to (Lakshminarayanan et al., 2017), which has a methodology comparable to ours for a univariate target. We use the same regression datasets as (Lakshminarayanan et al., 2017), but we create multivariate datasets by adding some of the features to the set of target dimensions. Specifically, we randomly sample $\lfloor d/2 \rfloor$ of the feature dimensions to be target labels and let the remaining $d - \lfloor d/2 \rfloor$ be the features to predict with, where d is the dimensionality of the original dataset.

For each dataset, we randomly select 3 train/test splits where each split is 90% of the original data is allocated for training, and the remaining 10% is used for test.

5.2. Model specification

We use the same architecture for the regression model throughout for all the methodologies: one hidden layer of 100 units with ReLU activations. We trained the model with a learning rate of 0.001, a batch size of 16, for 40 epochs.

For the error prediction model, our model has an initial hidden layer of 100 units, and 3 flow layers, with the flow network for each flow layer being a single hidden layer perceptron with 100 hidden units and softplus activation. We train the flow model for 100 epochs, and also with a learning rate of 0.001 and batch size of 16.

For the error prediction model, our model has an initial hidden layer of 100 units, and 3 flow layers, with the flow network for each flow layer being a single hidden layer perceptron with 100 hidden units and softplus activation. We train the flow model for 100 epochs, and also with a learning rate of 0.001 and batch size of 16. For the error prediction model, our model has an initial hidden layer of 100 units, and 3 flow layers, with the flow network for each flow layer being a single hidden layer perceptron with 100 hidden units and softplus activation. We train the flow model for 100 epochs, and also with a learning rate of 0.001 and batch size of 16.

5.3. Analysis

5.3.1. ERROR PREDICTION PERFORMANCE

Overall in 1, the flow distributions achieve a higher log-likelihood than the Gaussian baseline on the multivariate datasets. The Gaussian baseline, however, does do better on some datasets. Notably, the datasets with lower overall likelihood across all methods. This is consistent with the motivation behind a flow error distribution being better at characterizing more complex error distributions that are inherently non-Gaussian. Notably, the highest likelihood dataset naval-propulsion-plant has the highest likelihood with the Gaussian model while the dataset with the lowest likelihood across all methods, power-plant, has the highest likelihood with RealNVP.

5.3.2. PERFORMANCE COMPARISON OF ADDING CONVEXITY / CONCAVITY CONSTRAINT

As part of 3 we compare how a convex flow network compares against the a concave flow network and the unconstrained flow network. This comparison is to show that constraining the network to be concave or convex does not hinder the inference performance of the network. The results in 3 support this hypothesis, the difference in variance between the Monte Carlo integrations methods *Concave MC* vs *Convex MC*, *Normal MC* is insignificant.

5.3.3. TIGHTNESS OF UPPER BOUND AND LOWER BOUND

3 also provides data on how tight upper bound and lower bounds are compared to the baseline *Normal MC*. On average the lower bound is approximately 0.15 times the compared to the normal MC, and the upper bound is roughly 4 times greater than the baseline normal MC. With some preliminary results, we see that these bounds grow tighter with more training time. For example, the lower bound hovers around 0.2 and the upper bound at 1.4 when the model is trained against the *energy* dataset.

6. Conclusion

Overall, we demonstrate that using normalizing flows to parameterize the error distribution of a regression model as a viable alternative to using simple Gaussian models to accomplish the same task. Furthermore, we derived bounds on the variance of the flow distribution and empirically determined they can provide an analytic estimate of the error distribution’s variance without having to incorporate Monte Carlo integration. The theoretical bounds we derive are dependent upon the convexity/concavity of the flow and neural network functions, but we empirically that the even given concavity and convexity constraints, the learned flow distribution has similar statistics (e.g. variance) to flows learned without this constraint.

6.1. Future Work

For future work, we suggest further investigation into what specific distributions, or characteristics of specific distribution lead to better parameterization by normalizing flows. We found various avenues of research that we could not pursue due to time constraints.

6.1.1. GENERALIZING AWAY FROM GAUSSIANS

One of the most promising was generalizing the base layer distribution past the Gaussian. The normalizing flow is generally a transformation on a Gaussian random variable. However, throughout the course of this research we only used one property of the Gaussian to get these results. We only used the fact that the Gaussian mean was analytically derivable. Nearly, all well known distributions have analytically derivable means. It’s therefore perhaps a very interesting avenue of research to see if one can generalize away from Gaussians.

6.1.2. BOUNDING THE COVARIANCE

As part of our contributions for this paper we had planned on providing a bound not on the variance but on the covariance. However, that attempt proved to be too ambitious. Since the covariance can be seen as the product of two func-

tions in expectation, to show that the covariance could be bounded by a Jensen-like bound we needed to prove some kind of convexity result for general function products. We quickly realized that when the range of the function is not constrained, such a convexity result is impossible. We even came up with a sketch that it is impossible to show that the covariance of a transformation is convex. As a result we were unsure about how to proceed with the problem and reverted to just using variance (instead of covariance).

6.1.3. FORMALLY BOUNDING THE TIGHTNESS

There has been a lot of work in providing bounds on the Jensen’s gap which is the size

$$|\mathbb{E}[g(x)] - g(\mathbb{E}[x])| \quad (24)$$

the gap is written as function of the moments of the expectation and the rate of change of $g(x)$. In future work, it might be a good avenue of research to see if we can apply these existing bounds to the Jensen’s inequalities we use.

Flow Posterior for Uncertainty Estimation

Dataset	baseline (Gaussian only)	MAF	RealNVP
concrete	-8.033 ± 0.772	-5.695 ± 1.646	-7.056 ± 2.203
wine-quality-red	-3.402 ± 0.173	-5.415 ± 1.359	-5.214 ± 0.230
energy	-59.619 ± 23.635	-70.799 ± 32.325	-32.871 ± 5.624
kin8nm	-4.219 ± 0.032	-5.344 ± 0.720	-6.099 ± 0.714
bostonHousing	-9.416 ± 1.900	-14.570 ± 29.198	-8.307 ± 2.422
power-plant	-0.258 ± 0.055	-0.888 ± 0.416	-1.008 ± 0.414
yacht	-12.886 ± 2.671	-6.001 ± 1.865	-4.607 ± 0.872
naval-propulsion-plant	-123.423 ± 44.940	-86.042 ± 54.039	-109.908 ± 19.575

Figure 1. Mean and standard deviation of negative log-likelihood of over the train/test splits for each dataset.

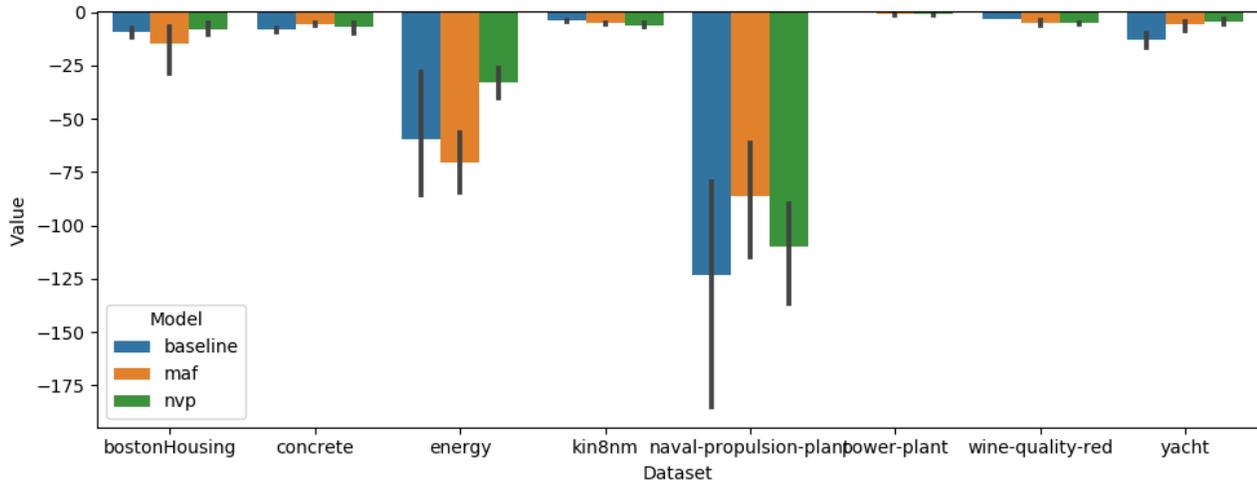


Figure 2. Bar plot of the negative log likelihood for each dataset, visualizing the above statistics.

Dataset	Concave (upper bound)	Concave MC	Convex (lower bound)	Convex MC	Normal MC
concrete	2.643	0.498	0.078	0.469	0.499
wine-quality-red	4.162	0.559	0.152	0.609	0.565
energy	1.807	0.379	0.077	0.325	0.327
kin8nm	4.592	0.946	0.149	0.961	0.969
bostonHousing	3.395	0.459	0.064	0.492	0.479
power-plant	0.972	0.459	0.010	0.299	0.310
yacht	2.013	0.518	0.063	0.554	0.546

Figure 3. Comparisons of the upper bound, lower bound and the Monte Carlo integration variance and Monte Carlo integration variance when we limit the transformation to be convex or concave

References

- Amos, B., Xu, L., and Kolter, J. Z. Input Convex Neural Networks. *arXiv:1609.07152 [cs, math]*, September 2016. URL <http://arxiv.org/abs/1609.07152>. arXiv: 1609.07152.
- Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight Uncertainty in Neural Networks. pp. 10. Distribution. *The Annals of Probability*, 9(3):533–535, June 1981. ISSN 0091-1798, 2168-894X. doi: 10.1214/aop/1176994428. URL <https://projecteuclid.org/euclid.aop/1176994428>.
- Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. *arXiv:1605.08803 [cs, stat]*, May 2016. URL <http://arxiv.org/abs/1605.08803>. arXiv: 1605.08803.
- Gal, Y. and Ghahramani, Z. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *arXiv:1506.02142 [cs, stat]*, June 2015. URL <http://arxiv.org/abs/1506.02142>. arXiv: 1506.02142.
- Germain, M., Gregor, K., Murray, I., and Larochelle, H. MADE: Masked Autoencoder for Distribution Estimation. *arXiv:1502.03509 [cs, stat]*, February 2015. URL <http://arxiv.org/abs/1502.03509>. arXiv: 1502.03509.
- Graves, A. Practical Variational Inference for Neural Networks. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 24*, pp. 2348–2356. Curran Associates, Inc., 2011. URL <http://papers.nips.cc/paper/4329-practical-variational-inference-for-neural-networks.pdf>.
- Hernandez-Lobato, J. M. and Adams, R. P. Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. *arXiv:1502.05336 [stat]*, February 2015. URL <http://arxiv.org/abs/1502.05336>. arXiv: 1502.05336.
- Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. Neural Autoregressive Flows. In *International Conference on Machine Learning*, pp. 2078–2087, July 2018. URL <http://proceedings.mlr.press/v80/huang18d.html>.
- Ioffe, S. and Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]*, February 2015. URL <http://arxiv.org/abs/1502.03167>. arXiv: 1502.03167.
- Kingma, D. P. and Dhariwal, P. Glow: Generative Flow with Invertible 1x1 Convolutions. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 10215–10224. Curran Associates, Inc., 2018. URL <http://papers.nips.cc/paper/8224-glow-generative-flow-with-invertible-1x1-convolutions.pdf>.
- Chernoff, H. A Note on an Inequality Involving the Normal

- Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, December 2013. URL <http://arxiv.org/abs/1312.6114>. arXiv: 1312.6114.
- Kingma, D. P., Salimans, T., and Welling, M. Variational Dropout and the Local Reparameterization Trick. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 2575–2583. Curran Associates, Inc., 2015. URL <http://papers.nips.cc/paper/5666-variational-dropout-and-the-local-reparameterization-trick.pdf>.
- Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improving Variational Inference with Inverse Autoregressive Flow. *arXiv:1606.04934 [cs, stat]*, June 2016. URL <http://arxiv.org/abs/1606.04934>. arXiv: 1606.04934.
- Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 6402–6413. Curran Associates, Inc., 2017. URL <http://papers.nips.cc/paper/7219-simple-and-scalable-predictive-uncertainty-estimation-using-deep-ensembles.pdf>.
- Liu, S., Liu, S., Cai, W., Pujol, S., Kikinis, R., and Feng, D. Early diagnosis of Alzheimer’s disease with deep learning. In *2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI)*, pp. 1015–1018, April 2014. doi: 10.1109/ISBI.2014.6868045.
- Louizos, C. and Welling, M. Multiplicative Normalizing Flows for Variational Bayesian Neural Networks. *arXiv:1703.01961 [cs, stat]*, March 2017. URL <http://arxiv.org/abs/1703.01961>. arXiv: 1703.01961.
- Neal, R. M. *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer New York, New York, NY, 1996. ISBN 978-0-387-94724-2 978-1-4612-0745-0. doi: 10.1007/978-1-4612-0745-0. URL <http://link.springer.com/10.1007/978-1-4612-0745-0>.
- Oord, A. v. d., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., Driessche, G. v. d., Lockhart, E., Cobo, L. C., Stimberg, F., Casagrande, N., Grewe, D., Noury, S., Dieleman, S., Elsen, E., Kalchbrenner, N., Zen, H., Graves, A., King, H., Walters, T., Belov, D., and Hasabis, D. Parallel WaveNet: Fast High-Fidelity Speech Synthesis. *arXiv:1711.10433 [cs]*, November 2017. URL <http://arxiv.org/abs/1711.10433>. arXiv: 1711.10433.
- Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep Exploration via Bootstrapped DQN. *arXiv:1602.04621 [cs, stat]*, February 2016. URL <http://arxiv.org/abs/1602.04621>. arXiv: 1602.04621.
- Papamakarios, G., Pavlakou, T., and Murray, I. Masked Autoregressive Flow for Density Estimation. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 2338–2347. Curran Associates, Inc., 2017. URL <http://papers.nips.cc/paper/6828-masked-autoregressive-flow-for-density-estimation.pdf>.
- Raghu, M., Blumer, K., Sayres, R., Obermeyer, Z., Kleinberg, R., Mullainathan, S., and Kleinberg, J. Direct Uncertainty Prediction for Medical Second Opinions. *arXiv:1807.01771 [cs, stat]*, July 2018. URL <http://arxiv.org/abs/1807.01771>. arXiv: 1807.01771.
- Rezende, D. J. and Mohamed, S. Variational Inference with Normalizing Flows. *arXiv:1505.05770 [cs, stat]*, May 2015. URL <http://arxiv.org/abs/1505.05770>. arXiv: 1505.05770.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL <http://jmlr.org/papers/v15/srivastava14a.html>.
- Teye, M., Azizpour, H., and Smith, K. Bayesian Uncertainty Estimation for Batch Normalized Deep Networks. In *International Conference on Machine Learning*, pp. 4907–4916, July 2018. URL <http://proceedings.mlr.press/v80/teyel8a.html>.
- Yang, Z., Hu, Z., Salakhutdinov, R., and Berg-Kirkpatrick, T. Improved Variational Autoencoders for Text Modeling Using Dilated Convolutions. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML’17*, pp. 3881–3890. JMLR.org, 2017. URL <http://dl.acm.org/citation.cfm?id=3305890.3306082>. event-place: Sydney, NSW, Australia.

References

- Amos, B., Xu, L., and Kolter, J. Z. Input Convex Neural Networks. *arXiv:1609.07152 [cs, math]*, September 2016. URL <http://arxiv.org/abs/1609.07152>. arXiv: 1609.07152.
- Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight Uncertainty in Neural Networks. pp. 10.
- Chen, C., Seff, A., Kornhauser, A., and Xiao, J. Deep-Driving: Learning Affordance for Direct Perception in Autonomous Driving. pp. 2722–2730, 2015. URL http://openaccess.thecvf.com/content_iccv_2015/html/Chen_DeepDriving_Learning_Affordance_ICCV_2015_paper.html.
- Chen, L. H. Y. An inequality for the multivariate normal distribution. *Journal of Multivariate Analysis*, 12(2):306–315, June 1982. ISSN 0047-259X. doi: 10.1016/0047-259X(82)90022-7. URL <http://www.sciencedirect.com/science/article/pii/0047259X82900227>.
- Chernoff, H. A Note on an Inequality Involving the Normal Distribution. *The Annals of Probability*, 9(3):533–535, June 1981. ISSN 0091-1798, 2168-894X. doi: 10.1214/aop/1176994428. URL <https://projecteuclid.org/euclid.aop/1176994428>.
- Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. *arXiv:1605.08803 [cs, stat]*, May 2016. URL <http://arxiv.org/abs/1605.08803>. arXiv: 1605.08803.
- Gal, Y. and Ghahramani, Z. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *arXiv:1506.02142 [cs, stat]*, June 2015. URL <http://arxiv.org/abs/1506.02142>. arXiv: 1506.02142.
- Germain, M., Gregor, K., Murray, I., and Larochelle, H. MADE: Masked Autoencoder for Distribution Estimation. *arXiv:1502.03509 [cs, stat]*, February 2015. URL <http://arxiv.org/abs/1502.03509>. arXiv: 1502.03509.
- Graves, A. Practical Variational Inference for Neural Networks. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 24*, pp. 2348–2356. Curran Associates, Inc., 2011. URL <http://papers.nips.cc/paper/4329-practical-variational-inference-for-neural-networks.pdf>.
- Hernandez-Lobato, J. M. and Adams, R. P. Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. *arXiv:1502.05336 [stat]*, February 2015. URL <http://arxiv.org/abs/1502.05336>. arXiv: 1502.05336.
- Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. Neural Autoregressive Flows. In *International Conference on Machine Learning*, pp. 2078–2087, July 2018. URL <http://proceedings.mlr.press/v80/huang18d.html>.
- Ioffe, S. and Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]*, February 2015. URL <http://arxiv.org/abs/1502.03167>. arXiv: 1502.03167.
- Kingma, D. P. and Dhariwal, P. Glow: Generative Flow with Invertible 1x1 Convolutions. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 10215–10224. Curran Associates, Inc., 2018. URL <http://papers.nips.cc/paper/8224-glow-generative-flow-with-invertible-1x1-conv.pdf>.
- Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, December 2013. URL <http://arxiv.org/abs/1312.6114>. arXiv: 1312.6114.
- Kingma, D. P., Salimans, T., and Welling, M. Variational Dropout and the Local Reparameterization Trick. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 2575–2583. Curran Associates, Inc., 2015. URL <http://papers.nips.cc/paper/5666-variational-dropout-and-the-local-reparameterization-trick.pdf>.
- Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improving Variational Inference with Inverse Autoregressive Flow. *arXiv:1606.04934 [cs, stat]*, June 2016. URL <http://arxiv.org/abs/1606.04934>. arXiv: 1606.04934.
- Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 6402–6413. Curran Associates, Inc., 2017. URL <http://papers.nips.cc/paper/7219-simple-and-scalable-predictive-uncertainty-estimation-using-deep-ensembles.pdf>.

- Liu, S., Liu, S., Cai, W., Pujol, S., Kikinis, R., and Feng, D. Early diagnosis of Alzheimer’s disease with deep learning. In *2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI)*, pp. 1015–1018, April 2014. doi: 10.1109/ISBI.2014.6868045.
- Louizos, C. and Welling, M. Multiplicative Normalizing Flows for Variational Bayesian Neural Networks. *arXiv:1703.01961 [cs, stat]*, March 2017. URL <http://arxiv.org/abs/1703.01961>. arXiv: 1703.01961.
- Neal, R. M. *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer New York, New York, NY, 1996. ISBN 978-0-387-94724-2 978-1-4612-0745-0. doi: 10.1007/978-1-4612-0745-0. URL <http://link.springer.com/10.1007/978-1-4612-0745-0>.
- Oord, A. v. d., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., Driessche, G. v. d., Lockhart, E., Cobo, L. C., Stimberg, F., Casagrande, N., Grewe, D., Noury, S., Dieleman, S., Elsen, E., Kalchbrenner, N., Zen, H., Graves, A., King, H., Walters, T., Belov, D., and Hasbabis, D. Parallel WaveNet: Fast High-Fidelity Speech Synthesis. *arXiv:1711.10433 [cs]*, November 2017. URL <http://arxiv.org/abs/1711.10433>. arXiv: 1711.10433.
- Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep Exploration via Bootstrapped DQN. *arXiv:1602.04621 [cs, stat]*, February 2016. URL <http://arxiv.org/abs/1602.04621>. arXiv: 1602.04621.
- Papamakarios, G., Pavlakou, T., and Murray, I. Masked Autoregressive Flow for Density Estimation. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 2338–2347. Curran Associates, Inc., 2017. URL <http://papers.nips.cc/paper/6828-masked-autoregressive-flow-for-density-estimation.pdf>.
- Raghu, M., Blumer, K., Sayres, R., Obermeyer, Z., Kleinberg, R., Mullainathan, S., and Kleinberg, J. Direct Uncertainty Prediction for Medical Second Opinions. *arXiv:1807.01771 [cs, stat]*, July 2018. URL <http://arxiv.org/abs/1807.01771>. arXiv: 1807.01771.
- Rezende, D. J. and Mohamed, S. Variational Inference with Normalizing Flows. *arXiv:1505.05770 [cs, stat]*, May 2015. URL <http://arxiv.org/abs/1505.05770>. arXiv: 1505.05770.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL <http://jmlr.org/papers/v15/srivastava14a.html>.
- Teye, M., Azizpour, H., and Smith, K. Bayesian Uncertainty Estimation for Batch Normalized Deep Networks. In *International Conference on Machine Learning*, pp. 4907–4916, July 2018. URL <http://proceedings.mlr.press/v80/teye18a.html>.
- Yang, Z., Hu, Z., Salakhutdinov, R., and Berg-Kirkpatrick, T. Improved Variational Autoencoders for Text Modeling Using Dilated Convolutions. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML’17*, pp. 3881–3890. JMLR.org, 2017. URL <http://dl.acm.org/citation.cfm?id=3305890.3306082>. event-place: Sydney, NSW, Australia.