University College London
Department of Computer Science
MSc Machine Learning

Bayesian Inference over Normalising Flows Parameters

Candidate Number: PFWP2
Supervisors: Brooks Paige & Matthew Willetts

This report is submitted as part requirement for the MSc degree in Machine Learning at University College London. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.

September 2021

Abstract

Normalizing flows are flexible models for defining highly-parameterised probability distributions. The flow parameters are usually assumed to be fixed, and are directly optimized under a maximum likelihood objective. In this project, we investigate the possibility of performing Bayesian inference on the flow parameters, leading to learning a distribution over flows. Through a series of experiments on simple synthetic datasets using the methods of deep ensemble, Monte Carlo dropout and mean-field variational inference, we demonstrate that distribution over flows varies significantly across methods and data size domains. We found that deep ensemble is the best method both to approximate distributions of observed data, and to perform uncertainty estimations over flows. We further interpret and explain our results through the lens of posterior distribution landscape of the flow parameters.

Contents

Chapter 1

Introduction

1.1 Motivation

One of the major goals in machine learning is to learn a probability distribution given samples that were drawn from that distribution. This task is generally referred to as probabilistic modelling and it is a one of the focuses in the area of unsupervised learning.

There are many ways to perform probabilistic modelling. Perhaps the simplest of all is the direct analytic approach, where the distribution of observed data is approximated directly and analytically. Variational approach and expectation maximisation are also popular choices for probabilistic modelling and they are comparatively more powerful than the direct analytic approach. Recent approaches such as the generative adversarial network [12] and variational autoencoder [17] has also been proposed for probabilistic modelling. In this report we will focus on one particular type of probability modelling framework, which is the normalising flow.

Normalising flows have been growing in popularity and they offer several benefits over other probabilistic modelling approaches. Namely, the ability to evaluate the densities of their distributions exactly and efficiently, and to sample from their distributions easily. Essentially, normalising flows are just probabilistic models that contain large number of parameters, with the added benefits that they are very flexible and are able to approximate complicated distributions with ease.

For a probabilistic model that is parameterised by some unknown parameters, Bayesian inference seeks to infer a distribution on the parameters based on some observed data, and by assuming a prior distribution on the parameters before observing the data. The result is that a posterior distribution of the parameters can be established, with

the distribution representing the uncertainty of the true locations of the parameters within the model.

Current methods for training normalising flows ultimately only return a single set of parameters after training, meaning that the parameters are in the form of point estimates and that there are no probability distributions placed on them, with no information regarding how uncertain the parameters are after training. In the case where a normalising flow is trained on a dataset that has insufficient data, there will be a high chance that the obtained parameters will be of inferior quality, potentially leading to inaccurate approximation to the distribution of observed data. To fix this, one would instead place a probability distribution on the parameters, and through the method of Bayesian inference, the uncertainty of the parameters can be recovered.

There have been many types of different parametric probabilistic models in the past that people have performed Bayesian inference on, however to the best of our knowledge, there has been no attempt to combine both the paradigms of Bayesian inference and normalising flows together. In light of this, we will investigate exactly that in this report, to perform Bayesian inference on normalising flows parameters. We will drop the traditional notion of training a normalising flow in a point estimate sense, and we will put probability distributions on the parameters of normalising flows, then capture their uncertainties by performing Bayesian inference on them. We will explore how the uncertainties of normalising flow parameters react under the different settings of large and finite data size domains, and to compare the results obtained from them, in hope of gaining a novel understanding into the implications of performing Bayesian inference on normalising flows parameters.

1.2 Objectives

In this report, we will place probability distributions onto the parameters of normalising flows, with the interpretation that the distributions represent uncertainties in parameters within the flows. We will investigate how uncertainties in parameters change when the flow models are trained with data in different size domains. We will also investigate how the uncertainties in parameters affect the overall the approximation ability of the flows to the distributions of observed data, also in different size domains. When explicitly estimating the uncertainties in parameters, we will use multiple methods to perform the estimations. We will compare the uncertainty estimations obtained by the different methods and investigate if they are well calibrated. Finally, with the results obtained, we will attempt to interpret and explain them.

1.3 Structure of the Report

The remaining of the report is organised as follows. In chapter 2, we will give an overview of the concepts required for our investigation. They are the main two paradigms of Bayesian inference and normalising flows. In chapter 3, we will combine Bayesian inference and normalising flows, and attempt to derive experiments that we can perform to obtain results. In chapter 4 we will describe the technicalities of experiment implementations, and how results can be obtain from them. In chapter 5, we will present all the results. Then we will attempt to compare, interpret and explain them. Finally in the last chapter, we will draw conclusions from our investigation, and briefly describe what possible research we can do in the future.

1.4 Public Repository

The code used in this project can be found in the following public repository:
`https://github.com/anthonychiuhy/Dissertation2021/`

Chapter 2

Background

In this chapter, we will discuss the background concepts behind the two main components which make up the majority of the substance in our report. They are Bayesian inference and normalising flow. On top of the fundamentals, we will also discuss other methods in the literature that are related to our investigation and we will be using in later chapters. Getting familiar with the concepts involved will be essential to understanding the material in the remaining of the report.

2.1 Bayesian Inference

Bayesian inference is a method of statistical inference for statistical models, in which the parameters of the models are assumed to be probabilistic in nature. That is in a statistical model, the parameters of the model are collectively distributed according to a probability distribution. As opposed to a frequentist approach where the parameters of the models are assumed to have fixed values without uncertainty, the Bayesian approach does not assume certainty in the parameters of the models. Instead it assumes there exist intrinsic uncertainties in the parameters and the true values of the parameters can never be known exactly. Their uncertainties can be reduced as more data or evidence are collected and taken into consideration.

The central idea of Bayesian inference evolves around the Bayes' theorem:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \tag{2.1}$$

Given a statistical model $\mathcal{M}$ parameterised by parameters $\boldsymbol{\theta}$, $p(\boldsymbol{\theta}|\mathcal{D})$ is the posterior distribution for the model parameters $\boldsymbol{\theta}$, given dataset $\mathcal{D}$. $p(\mathcal{D}|\boldsymbol{\theta})$ is the likelihood term given $\boldsymbol{\theta}$. It describes the probability of obtaining the dataset under $\boldsymbol{\theta}$ parameterisation

of the model. $p(\boldsymbol{\theta})$ is the prior distribution for parameters $\boldsymbol{\theta}$, it is the prior belief on the distribution of the model parameters $\boldsymbol{\theta}$, before seeing any data. $p(\mathcal{D})$ is the marginal likelihood term, also known as the evidence. This term describes the probability of obtaining the dataset $\mathcal{D}$ under the assumed statistical model $\mathcal{M}$. Note the name of marginal likelihood came from the fact that it is obtained by marginalising the likelihood over $\boldsymbol{\theta}$:

$$p(\mathcal{D}) = \int p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta} \tag{2.2}$$

The main quantity of interest in this report is the posterior term $p(\boldsymbol{\theta}|\mathcal{D})$ in the Bayes' theorem. Given a simple enough statistical model and an appropriate prior distribution, it is possible to compute the posterior distribution analytically according to the Bayes' theorem. However in practice, the posterior distribution is usually computationally intractable. The reason being that in many cases one cannot compute the marginal likelihood term in the denominator of the Bayes' theorem analytically. Computing the marginal likelihood as shown in (2.2) requires computing an integral, and in most interesting statistical models, the integral is usually intractable.

Fortunately, there exists many approximate methods in Bayesian inference where we can approximately compute the posterior distribution. Popular methods includes the Laplace approximation, Markov chain Monte Carlo methods and the variational Bayesian methods.

The Laplace approximation [24] approximates a posterior distribution locally using a Gaussian distribution, where the location of the Gaussian mode is to be computed such that it coincides with a mode of the posterior distribution. Markov chain Monte Carlo methods [27] allow the direct drawing of samples from the posterior distribution without actually knowing the exact analytic form for the posterior. In Variational Bayesian methods [2], the posterior distribution is to be approximated by another simpler distributions, which is called the variational distribution. Through an optimisation of a lower bound on the marginal likelihood by an iterative scheme, the variational distribution will gradually move closer to the posterior distribution, and their distance will decrease in the Kullback–Leibler divergence sense. In effect the variational distribution will approximate the posterior distribution.

In this report we will be exploring the possibility of performing Bayesian inference on the parameters of normalising flows. Due to the structural complexity of the normalising flows and the complex interdependence of the parameters, it would be almost impossible for the posterior distribution of the parameters in the normalising flows to have an analytic form given a prior distribution and the likelihood. Therefore we would need an approximation method to approxiate the the posterior distribution.

In this report we will consider a variational Bayesian method, the mean field variational inference, as a method for posterior approximation. Furthermore, we will also explore approximate Bayesian methods to estimate uncertainty in normalising flows parameters, where comparisons can be made between them.

2.1.1 Maximum Likelihood and Maximum a Posteriori Estimations

Before introducing proper Bayesian inference in the deep learning framework, it is useful to introduce two useful methods that are popular and very much related to Bayesian inference. These two methods are the maximum likelihood (ML) estimation and the maximum a posteriori (MAP) estimation. Similar to the goal of Bayesian inference where information about the model parameters $\boldsymbol{\theta}$ are to be deduced after seeing data, in ML or MAP instead of deducing the complete distribution of $\boldsymbol{\theta}$ over its support, only point estimates for $\boldsymbol{\theta}$ can be made for both methods. Thus, these two methods are not strictly Bayesian due to their point estimate natures.

As the name suggested, ML estimation maximises the likelihood of data with respect to the model parameters $\boldsymbol{\theta}$, then the optimal parameters $\boldsymbol{\theta}^*$ for ML estimation is given by:

$$\boldsymbol{\theta}^*_{\text{ML}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\mathcal{D}|\boldsymbol{\theta}) \tag{2.3}$$

Also as the name suggested, MAP estimation seeks to find the maximum point in the posterior distribution of $\boldsymbol{\theta}$ given data. This corresponds to finding the optimal $\boldsymbol{\theta}$ such that its location maximises the posterior distribution. Then the optimal parameters $\boldsymbol{\theta}^*$ for MAP estimation is given by:

$$\begin{aligned} \boldsymbol{\theta}^*_{\text{MAP}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\boldsymbol{\theta}|\mathcal{D}) \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \end{aligned} \tag{2.4}$$

Where the Bayes' theorem in (2.1) and the fact that $p(\mathcal{D})$ is constant have been used to arrived at the second line. Note that ML and MAP estimations are equivalent when the prior in MAP estimation is uniformly distributed such that $p(\boldsymbol{\theta}) = \text{constant}$.

2.1.2 Bayesian Deep Learning

Bayesian deep learning is where classical deep learning is viewed in a Bayesian angle. To understand Bayesian deep learning we will have to first understand what deep learning is. In classical deep learning, we consider neural networks as machine

learning models for learning. Given a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ with N data points, feature vectors $\boldsymbol{x}_i$ and labels y_i, classical deep learning aims to learn a function $f(\cdot\,; \boldsymbol{\theta})$ parameterised by a very large number of parameters in $\boldsymbol{\theta}$, such that

$$f(\boldsymbol{x}_i; \boldsymbol{\theta}) = \hat{y}_i \approx y_i \tag{2.5}$$

for prediction $\hat{y}_i$, which approximates the true label y_i as close as possible, for the whole dataset $\mathcal{D}$ in some metric for closeness. Popular metric includes the mean squared loss, cross-entropy loss or the negative log-likelihood. The function f here is a neural network. During learning, the parameters in $\boldsymbol{\theta}$ will get updated by a learning algorithm so that a user defined loss between $\{y_i\}_{i=1}^N$ and $\{\hat{y}_i\}_{i=1}^N$ will be minimised. At prediction time the learnt $\boldsymbol{\theta}$ would be fixed and the hope is that the trained neural network can generalise to new unseen data. In the deep learning community, the parameters are usually called weights and biases, however in here we are instead calling them as parameters and we are collectively arranging them in a giant vector and denoting them as $\boldsymbol{\theta}$.

For the actual structure of a neural network, it consists of multiple layers of affine transformations, with each layer of affine transformation followed by a non-linear transformation. In each layer, the affine transformation is parameterised by weights and biases. Mathematically a neural network with L layers is characterised by the following equations:

$$\begin{aligned}
\boldsymbol{a}_0 &= \boldsymbol{x} \\
\boldsymbol{a}_1 &= \sigma(\boldsymbol{W}_1 \boldsymbol{a}_0 + \boldsymbol{b}_1) \\
\boldsymbol{a}_2 &= \sigma(\boldsymbol{W}_2 \boldsymbol{a}_1 + \boldsymbol{b}_2) \\
&\vdots \\
\boldsymbol{a}_L &= \sigma(\boldsymbol{W}_L \boldsymbol{a}_{L-1} + \boldsymbol{b}_L) \\
\hat{\boldsymbol{y}} &= \boldsymbol{a}_L
\end{aligned} \tag{2.6}$$

with $\boldsymbol{x}$ as the input feature vector and $\hat{\boldsymbol{y}}$ as the output or the prediction of the neural network. The weights and biases in the affine transformation for the l^{th} layer are contained in the weight matrix $\boldsymbol{W}_l$ and bias vector $\boldsymbol{b}_l$ respectively. The non-linear transformation is the function σ, it is also known as the activation function. After passing through the activation function, we have reached the end of a layer with values contained in $\boldsymbol{a}_l$, which are called the neurons. For a neural network with L layers, the process of affine transformation followed by non-linear transformation will be repeated for L times for an initial input of a feature vector, and the output of the network would be the result of a combined complicated transformation, as depicted in

the equations. Overall the equations in (2.6) is effectively an elaborate representation for equation (2.5), with all the weights and bias packed into a single parameter vector

$$\boldsymbol{\theta} = [\boldsymbol{W}_1, \ldots, \boldsymbol{W}_L, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_L] \tag{2.7}$$

During the training phase of the neural network when loss is minimised, only a single fixed set of parameters in $\boldsymbol{\theta}$ would be produced at the end of training. There will be no uncertainty in the trained parameters to inform us that whether the parameters are close to the optimal parameters or not. It is possible that although we have arrived at a set of fixed parameters, due to the dataset for training our neural network model being small, the trained model would likely to have bad generalisation when the model is tested on new data. In this case it means that the parameters we obtained are very likely to be far from optimal. In a sense, we can say that we are uncertain about whether the parameters we obtained are optimal or not after training with small dataset. On the other hand if we train a neural network model with a sufficiently large dataset, we will likely to be arriving at a set of parameters that generalises well to unseen data. This means that we can be relatively certain that the parameters we obtained are close to optimal. This corresponds to the parameters having low uncertainty. As we can see in classical deep learning, the way neural networks are trained does not provide any measure of uncertainty in the parameters, and we can see that not having a measure of uncertainty might sometimes mislead us to be overconfident on the predictions of our trained models. Having the ability to measure the uncertainty of parameters allows the user to make more informed decisions based on how certain the predictions are. Is there a way where we make use of neural network and be able to get uncertainty of its parameters at the same time? This is where Bayesian deep learning comes to rescue.

Bayesian deep learning aims to solve the problem of measuring uncertainty in neural networks. On a high level Bayesian deep learning is an area where Bayesian inference is applied to classical deep learning. This means that the concepts introduced previously for Bayesian inference can be applied to the parameters $\boldsymbol{\theta}$ in neural networks, in that we will be making use of the Bayes' theorem (2.1) to compute posterior distributions for $\boldsymbol{\theta}$. However, it is usually the case that the posterior distribution on neural network parameters are computationally intractable to compute exactly, due to the complex dependence of the parameters and their transformations within neural networks. Instead we could estimate the posterior using the approximation methods we introduced previously, such as the Laplace's approximation, the Markov chain Monte Carlo methods and the variational Bayesian methods.

Related Work

Historically, Bayesian deep learning can be traced back to [7] where the weights of neural networks was approximated by Gaussian distributions, with the goal of having the neural networks to generate probabilistic outputs. Equivalently, this is the same as applying the Laplace's method on weights of neural networks. [23] performed a comprehensive study on neural networks under the Bayesian framework, where it was shown that the Bayesian evidence can be treated as a proxy for determining the generalisation ability of neural network models. For variational inference, [14] derived essentially the mean-field variational inference approach via the minimum description length principle from information theory, in that weights of the Bayesian neural networks was approximated using Gaussians with covariance matrices being diagonal. Instead of modelling the weights of the Bayesian neural network using diagonal Gaussians, [1] extended the variational inference approach by also modelling the correlation between weights, which is essentially a full covariance variational approximation. [13] extended [14] work on variational inference, allowing for the posterior to be optimised numerically using data subsampling. Previously variational inference on large neural network architectures were infeasible, however the technique introduced in [11] opened up the possibility of variational inference on large architectures. Further work of [3] introduced a simpler algorithm, Bayes by Backprop, in which the reparameterisation trick enabled the use of backpropagation in variational inference, further lowering the difficulty of performing variational inference on large neural network architectures. [11] introduced a computationally inexpensive method of measuring neural network models uncertainties, the Monte Carlo dropout, in that dropout [31] is used at test time to perform approximate Bayesian inference on the posterior predictive distribution. [22] proposed a surprisingly simple way of estimating uncertainties in neural networks. Their method works by simply combining neural networks and treating them all as one deep ensemble. They showed that this deep ensemble method can produce high quality uncertainty such that the uncertainty obtained provides a strong baseline for predictive uncertainty quantification.

2.1.3 Deep Ensemble

It is perhaps surprising to the Bayesian deep learning community that a non-Bayesian approach to predictive uncertainty can perform as well as a pure Bayesian approach. The method of uncertainty estimation using deep ensemble [22] is a method that is non-Bayesian in nature for neural networks. It is relatively easy to implement in that it requires little modification to the existing training routine where uncertainty estimation is not considered. Furthermore it is readily parallelisable and requires little

hyperparameter tuning.

Experiments in [22] shows that the deep ensemble method provides a strong baseline for predictive uncertainty estimation for neural networks, and that it can be treated as an useful benchmark for comparing other tasks of uncertainty estimation, be it Bayesian, non-Bayesian or hybrid methods. [10] explains why the deep ensemble method has such high performance when performing uncertainty estimations, through the angle of loss landscape of neural networks. It compared the deep ensemble method with the other Bayesian or non-Bayesian methods for uncertainty estimation.

In the method of deep ensemble, on a high level all we have to do is to train our neural network models for many times, then combine them together to form an ensemble. In specific, for each member in the ensemble, one aims to train a neural network model such that given an input feature vector $\boldsymbol{x}$, it produces a probabilistic predictive distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}, \boldsymbol{\theta})$ after training on dataset $\mathcal{D}$, with the final trained neural network having parameters $\boldsymbol{\theta}$. For M is the number of members in our desired deep ensemble, the exact steps for training a deep ensemble are as follows:

1. Initialise the parameters of a new neural network model randomly.
2. Perform MAP estimation on the model given training data.
3. Save trained model and go back to step 2 until M number of models have been trained.
4. All M models collectively can now be viewed as a deep ensemble.

The final combined ensemble of models can now be used for predictive uncertainty estimation by taking simple averages from all the trained models:

$$p(\boldsymbol{x}|\mathcal{D}) = \frac{1}{M} \sum_{m=1}^{M} p_{\boldsymbol{\theta}_m}(\boldsymbol{x}|\mathcal{D}, \boldsymbol{\theta}_m) \tag{2.8}$$

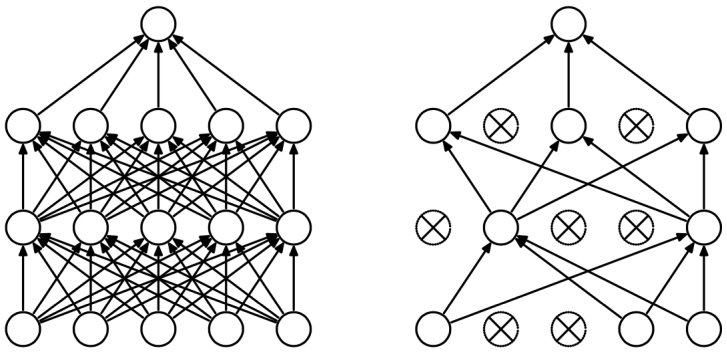
for $p_{\boldsymbol{\theta}_m}(\boldsymbol{x}|\mathcal{D}, \boldsymbol{\theta}_m)$ is the probabilistic predictive distribution for the m^{th} trained neural network over M neural networks.

2.1.4 Monte Carlo Dropout

The Monte Carlo (MC) dropout method for Bayesian approximation was introduced in [11], where this method was inspired by another method for a completely different purpose, as introduced in [32] by the name of dropout. Dropout was originally designed as a technique for neural network regularisation during training. Although MC dropout and dropout was originally designed for different purposes, they do work very similarly

in implementation.

We shall first understand how dropout works before describing MC dropout in detail. Dropout [32] is a technique that can be applied to neural networks. During a forward propagation phase in a neural network, what dropout does is that it turns some of the neurons in the neural network on or off randomly, according to a user defined dropout probability. Hence the name for dropout as neurons are being dropped out. Dropout probability refers to the probability of a neuron being turned off in an forward propagation instance. When a neuron is turned off, it means the value of the neuron is set to zero temporarily at its current forward propagation phase. In effect the neurons that being turned off will not have any contribution to the neurons that are in the downstream direction, as well as the final output. For a dropout probability of zero, it means each neuron has a zero probability of being turned off and that there will be no neuron being turned off or set to zero in the whole of neural network; whereas a dropout probability of one means each neuron will always be turned off, and it also means that all neurons in the whole neural network are being turned off all the time. For another phase for forward propagation, the whole process of choosing which neurons to be turned off will start all over again. Overall after many forward propagation phases, many neurons would have been turned on or off for many times, depending the dropout probability. A graphical representation of dropout in a neural network is shown in figure 2.1. As can be seen from the figure, on the left shows a complete neural network without dropout, where all neurons in a layer affect all neurons in the next layer immediately after; whereas on the right, we can see that some of the neurons in a layer have been turned off and they cannot affect the neurons in the next layer, or any other neurons in further layers.



(a) Standard Neural Net (b) After applying dropout.

Figure 2.1: Dropout neural network model figure taken from [32]. (a) Two layers standard neural network without dropout. (b) The same two layers neural network but with dropout turned on, where the crossed units have been dropped out.

In a typical neural network, from its creation to the actual use of it for predictions, it would have been experienced two different stages in its lifetime. They are the training stage and the testing stage. In the training stage, the neural network will adjust its parameters based on the training dataset that it is trained on. Whereas in the testing stage, the neural network will instead freeze its parameters and use the frozen parameters for predictions on a separate test dataset. The main difference in the use of dropout in [32] for standard dropout and in [11] for MC dropout is that dropout is only used in the training stage for standard dropout with dropout turned off at the testing stage, utilising the complete neural network architecture; whereas dropout is used in both training and testing stages for MC dropout.

[11] showed that applying dropout to all the layers in a neural network is mathematically equivalent to an approximation to a Bayesian probabilistic model, the probabilistic deep Gaussian process [5].

For an input vector $\boldsymbol{x}$ and a training dataset $\mathcal{D}$, the steps for performing MC dropout is described as follows:

1. Initialise a neural network model with a dropout probability, where the dropout probability is a hyperparameter that is defined by the user.
2. Perform MAP estimation on the model given dataset $\mathcal{D}$, with dropout turned on.
3. With dropout turned on, make as many predictions $\hat{p}(\boldsymbol{x}|\mathcal{D})$ as desired on the same input $\boldsymbol{x}$. This corresponds to sampling from the approximate predictive distribution.

With all the obtained predictions, we can Monte Carlo estimate the predictive uncertainty by taking simple averages using the predictions:

$$p(\boldsymbol{x}|\mathcal{D}) = \frac{1}{M}\sum_{m=1}^{M}\hat{p}_m(\boldsymbol{x}|\mathcal{D}) \tag{2.9}$$

for $\hat{p}_m(\boldsymbol{x}|\mathcal{D})$ is the predictive probability produced by the m^{th} neural network prediction with dropout over M neural networks.

2.1.5 Mean-Field Variational Inference

In previous two sections we discussed the deep ensembles and the MC dropout methods where those two methods can produce uncertainty estimates for neural network models. However, those two methods took an indirect approach to Bayesian inference,

in that the posterior distribution of neural network parameters were not directly approximated. In this section we approximate the posterior distribution directly, based on a well-known method in approximate Bayesian inference, which is called variational inference (VI). In Bayesian inference, the goal is to infer the posterior distribution of the parameters of a model, however as mentioned before, the posterior distribution usually does not have an analytical expression and it often does not belong to any well-known family of distribution. In VI, one aims to approximate the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ with a distribution $q_\lambda(\boldsymbol{\theta})$, which usually belongs to a simpler family of distributions:

$$q_\lambda(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta}|\mathcal{D}) \tag{2.10}$$

with $q_\lambda(\boldsymbol{\theta})$ parameterised by λ. Notice that $q_\lambda(\boldsymbol{\theta})$ is not a fixed distribution and can be varied by adjusting λ, thus $q_\lambda(\boldsymbol{\theta})$ is also called a variational distribution. By adjusting λ one can adjust $q_\lambda(\boldsymbol{\theta})$ to be close to the true posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ subject to the approximation ability of the family of distribution in which $q_\lambda(\boldsymbol{\theta})$ belongs to. For mathematical convenience, one would usually pick $q_\lambda(\boldsymbol{\theta})$ as a Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ such that:

$$q_\lambda(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{2.11}$$

then now the adjustable parameters are $\lambda = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$. In the remaining sections, we would always let $q_\lambda(\boldsymbol{\theta})$ to be in the Gaussian distribution family.

Mean-Field Approximation

On top of making the assumption that $q_\lambda(\boldsymbol{\theta})$ is Gaussian for mathematical convenience, we can further simplify by assuming that $q_\lambda(\boldsymbol{\theta})$ can be factorised over all parameters in $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)$, so that

$$q(\boldsymbol{\theta}) = q(\theta_1, \ldots, \theta_n) = \prod_{i=1}^{n} q_i(\theta_i) \tag{2.12}$$

with

$$q_i(\theta_i) = \mathcal{N}(\theta_i|\mu_i, \sigma_i^2) \tag{2.13}$$

The decomposition of the full distribution into factors is called the mean-field approximation. In effect we have made the full Gaussian in $q_\lambda(\boldsymbol{\theta})$ to have diagonal covariance matrix by making this mean-field approximation.

Kullback-Leibler Divergence

Kullback-Leibler (KL) Divergence is a quantitative measure of distance or similarity between two distributions. A small KL divergence between two distributions means that two distribution are similar to each other in the KL sense. In the optimal case of the KL divergence between two distributions being equal to zero, this will mean that the two distributions are exactly the same. The KL divergence D_{KL} between two distributions $p(\boldsymbol{\theta})$ and $q(\boldsymbol{\theta})$ is defined as:

$$D_{KL}(q(\boldsymbol{\theta}) \parallel p(\boldsymbol{\theta})) := \int q(\boldsymbol{\theta}) \log \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta})} d\boldsymbol{\theta} \tag{2.14}$$

For our VI setting, we want our variational distribution to be as close to the posterior distribution as possible, then now we can make use of KL divergence to measure the distance between them and attempt to minimise it. The distance between them is:

$$D_{KL}(q_\lambda(\boldsymbol{\theta}) \parallel p(\boldsymbol{\theta}|\mathcal{D})) = \int q_\lambda(\boldsymbol{\theta}) \log \frac{q_\lambda(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|\mathcal{D})} d\boldsymbol{\theta} \tag{2.15}$$

We have framed our original approximate Bayesian inference problem to an optimisation problem. By minimising the KL divergence with respect to λ, we are basically making $q_\lambda(\boldsymbol{\theta})$ and $p(\boldsymbol{\theta}|\mathcal{D})$ as close to each other as they can. The whole process of minimising the KL divergence between the variational distribution and the posterior is the central idea of VI.

Evidence Lower Bound

The next natural question to ask is how we can actually minimise the KL divergence. We can manipulate the KL divergence mathematically to make it more optimisation friendly, by making use of the rules of probability, log and expectation, we have:

$$\begin{aligned} D_{KL}(q_\lambda(\boldsymbol{\theta}) \parallel p(\boldsymbol{\theta}|\mathcal{D})) &= \mathbb{E}_{q_\lambda(\boldsymbol{\theta})}\left[\log \frac{q_\lambda(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|\mathcal{D})}\right] \\ &= \mathbb{E}_{q_\lambda(\boldsymbol{\theta})}\left[\log \frac{q_\lambda(\boldsymbol{\theta}) p(\mathcal{D})}{p(\boldsymbol{\theta}, \mathcal{D})}\right] \\ &= \mathbb{E}_{q_\lambda(\boldsymbol{\theta})}[\log p(\mathcal{D})] + \mathbb{E}_{q_\lambda(\boldsymbol{\theta})}\left[\log \frac{q_\lambda(\boldsymbol{\theta})}{p(\boldsymbol{\theta}, \mathcal{D})}\right] \\ &= \log p(\mathcal{D}) - \mathbb{E}_{q_\lambda(\boldsymbol{\theta})}\left[\log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q_\lambda(\boldsymbol{\theta})}\right] \end{aligned} \tag{2.16}$$

where we have decomposed the KL divergence term into a constant term $p(\mathcal{D})$ and another variable term $\mathbb{E}_{q_\lambda(\boldsymbol{\theta})}\left[\log \frac{p(\boldsymbol{\theta},\mathcal{D})}{q_\lambda(\boldsymbol{\theta})}\right]$. The constant term is the log evidence and the variable term is referred to as the evidence lower bound or ELBO, it is also known

as the variational lower bound. After a simple rearrangement of the above, we obtain:

$$\underbrace{\log p(\mathcal{D})}_{\text{const.}} = \underbrace{D_{KL}(q_\lambda(\boldsymbol{\theta}) \parallel p(\boldsymbol{\theta}|\mathcal{D}))}_{\geq 0} + \underbrace{\mathbb{E}_{q_\lambda(\boldsymbol{\theta})}\left[\log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q_\lambda(\boldsymbol{\theta})}\right]}_{ELBO} \tag{2.17}$$

making use of the fact that KL divergence is always greater than or equal to zero, we can deduce that the log evidence is at least as large as the ELBO term, this implies that log evidence is lower bounded by the ELBO term, which is why this term is called the evidence lower bound. Since the log evidence term is constant, the sum of the KL divergence and the ELBO terms must also be a constant. Taking advantage of this we can now minimise the KL divergence by maximising the ELBO term. By defining ELBO as the lower bound $\mathcal{L}(\lambda; \mathcal{D})$ parameterised by λ:

$$\mathcal{L}(\lambda; \mathcal{D}) := \mathbb{E}_{q_\lambda(\boldsymbol{\theta})}\left[\log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q_\lambda(\boldsymbol{\theta})}\right] \tag{2.18}$$

we can write out our optimisation problem mathematically:

$$\underset{\lambda}{\text{argmin}} \;\; D_{KL}(q_\lambda(\boldsymbol{\theta}) \parallel p(\boldsymbol{\theta}|\mathcal{D})) = \underset{\lambda}{\text{argmax}} \;\; \mathcal{L}(\lambda; \mathcal{D}) \tag{2.19}$$

We have arrived at the conclusion that minimising the distance between the variational distribution and the posterior distribution is equivalent to maximising the ELBO.

The Reparameterisation Trick

Before describing how to maximise for ELBO, there is a useful technique that will be important to our discussion of ELBO maximisation. The technique is called the reparameterisation trick. The reparameterisation trick can be thought of as a technique for sampling from a non-standard Gaussian distribution, in which instead of sampling directly from the Gaussian distribution, we sample from a standard Gaussian distribution instead then perform a transformation, such that the final sample obtained is distributed according to the original Gaussian distribution. Mathematically, for normally distributed parameters $\boldsymbol{\theta}$ with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$:

$$\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{2.20}$$

At sampling stage, we could reparameterise it using the property of Gaussian distribution such that:

$$\begin{aligned} \boldsymbol{\theta} &= \boldsymbol{\mu} + \boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\epsilon} \\ \boldsymbol{\epsilon} &\sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \boldsymbol{I}) \end{aligned} \tag{2.21}$$

Then when we draw samples of $\boldsymbol{\epsilon}$ from standard Gaussian and reparameterise according to the formula above, we will obtain samples that are distributed exactly to the original Gaussian with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Here, $\boldsymbol{\Sigma}^{\frac{1}{2}}$ could be a Cholesky decomposition of $\boldsymbol{\Sigma}$. If we make the mean-field assumption that the original Gaussian can be factorised, then it means that the covariance matrix $\boldsymbol{\Sigma}$ is diagonal with entries consist of variances $\{\sigma_i^2\}$ of the factors in the original Gaussian, with $\boldsymbol{\Sigma} = \text{diag}(\boldsymbol{\sigma}^2)$, and this implies $\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\epsilon} = \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$. Then with the mean-field approximation, we can write the reparameterisation as:

$$\boldsymbol{\theta} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} \tag{2.22}$$

Bayes by Backprop

Coming back to our optimisation problem of maximising the ELBO. We have transformed our KL divergence minimisation problem into an ELBO maximisation problem (2.19). However optimising the ELBO term is still not straight forward. We might want to optimise ELBO using any standard optimisation techniques, such as using the gradient descent method. However, most standard optimisation techniques require the computation of gradients. In the case of ELBO, computing its gradient is not straight forward. [29] attempts to estimate the ELBO gradient using a more algebraic approach, however it ultimately leads to high variance estimate of the gradient. Following the method of Bayes by Backprop in [3], the gradient can be estimated more easily with less variance. This approach involves employing the reparameterisation trick. With the trick, the ELBO term could be differentiated easily. For the parameters $\boldsymbol{\theta}$ that are distributed according to a Gaussian distribution in $q_\lambda(\boldsymbol{\theta})$, it can be reparameterised by $\lambda = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$, we then have:

$$\boldsymbol{\theta} = \boldsymbol{\theta}(\lambda, \boldsymbol{\epsilon}) \equiv \boldsymbol{\theta}(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\epsilon}) = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} \tag{2.23}$$

Making use of this reparameterisation, one can switch the variational distribution $q_\lambda(\boldsymbol{\theta})$ in the expectation of ELBO with a standard normal distribution $p(\boldsymbol{\epsilon})$, and we can differentiate through the ELBO with respect to λ by exchanging the order of expectation and the gradient:

$$\begin{aligned}
\nabla_\lambda \mathcal{L}(\lambda) &= \nabla_\lambda \mathbb{E}_{q_\lambda(\boldsymbol{\theta})}\left[\log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q_\lambda(\boldsymbol{\theta})}\right] \\
&= \nabla_\lambda \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\log \frac{p(\boldsymbol{\theta}(\lambda, \boldsymbol{\epsilon}), \mathcal{D})}{q_\lambda(\boldsymbol{\theta}(\lambda, \boldsymbol{\epsilon}))}\right] \\
&= \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\nabla_\lambda \log \frac{p(\boldsymbol{\theta}(\lambda, \boldsymbol{\epsilon}), \mathcal{D})}{q_\lambda(\boldsymbol{\theta}(\lambda, \boldsymbol{\epsilon}))}\right]
\end{aligned} \tag{2.24}$$

with $\boldsymbol{\epsilon}$ distributed normally according to a standard normal distribution $p(\boldsymbol{\epsilon})$:

$$\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}) = \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{I}) \tag{2.25}$$

Now, with the new expression for the gradient of ELBO, one can simply estimate this by performing a Monte Carlo estimation of the expectation according to the following:

1. Draw samples of $\boldsymbol{\epsilon}$ from $p(\boldsymbol{\epsilon})$
2. Use the samples of $\boldsymbol{\epsilon}$ along with $\lambda = \{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$ to compute $\boldsymbol{\theta}$ using reparameterisation in equation (2.23)
3. Compute the gradient term in the expectation for all samples of $\boldsymbol{\epsilon}$ for

$$\nabla_\lambda \log \frac{p(\boldsymbol{\theta}(\lambda, \boldsymbol{\epsilon}), \mathcal{D})}{q_\lambda(\boldsymbol{\theta}(\lambda, \boldsymbol{\epsilon}))} \tag{2.26}$$

   using any standard auto-differentiation tool.
4. Monte Carlo estimate the gradient of ELBO by taking a simple average of all the the gradient terms calculated in step 3.

With the estimate of the gradient of ELBO obtained, one can finally make use of the gradient estimate to maximise the ELBO using any standard optimisation techniques, such as the gradient descent.

Now that we can maximise the ELBO, this implies that the KL divergence of the variational distribution and the posterior can be minimised as in the relation we derived earlier in (2.19), and it also means that at the end we would obtain a variational distribution which is approximately close to the true posterior, satisfying our original aim of posterior approximation.

2.2 Normalising Flows

In machine learning and statistics, one of the main goals is to model a probability distribution given only samples that were drawn from that distribution. For example, one might want to model the distribution of income for the population in a city, given samples of income figures in the city. Or one might want to model the distribution of the speed of particles in a gas, given samples of measured speed of the particles in the gas. In more realistic situations, the distributions involved could be much more complicated. For example a popular task in machine learning is human face generation, in which the task of the machine learning algorithm is first to learn and extract pattern

from many samples of images of human faces, then it attempts to generate completely new human faces using the knowledge that the machine learning algorithm learnt. One approach of doing this is probabilistic modelling. In this approach one aims to model the probability distribution of faces given a dateset of images of faces, then from the learnt distribution, one can draw new samples from it. Given that the learnt probability distribution is a good quality approximation to the true distribution, the samples drawn would resemble human faces that are of high quality and have not existed in the training dataset.

The task of modelling a probability distribution is sometimes called generative modelling, since if the distribution is modelled successfully, one can basically generate as many new samples as they like that closely match the true distribution. Many methods have been proposed for generative modelling. For example, the variational Bayesian methods that we have discussed in previous section on Bayesian inference aims to model a distribution. Expectation maximisation is also another method which introduces the notion of latent variables to facilitate the learning of a distribution from data. More recent approaches include generative adversarial networks (GAN) [12] and variational auto-encoders (VAE) [17] are also approaches to generative modelling. However, neither GAN nor VAE explicitly learns the distribution from data. In this report we will consider a specific framework for generative modelling, which is the normalising flow.

Normalising flow (NF) is a family of generative models that have useful properties. The main advantage of NF is that the distributions it models can be expressed exactly in an algebraic sense. It means that a probabilistic model from NF can be written down as an exact mathematical expression and it is possible to calculate the learnt distribution from NF exactly, even better the distributions are designed to be always tractable. The way NF is designed also makes sampling from its distribution relatively efficient.

On a high level, NF works by first putting a simple distribution, such as a standard Gaussian, at one end of a chain, then through a flow of successive invertible and differentiable chain of learnable transformations, the simple distribution would gradually be transformed to a distribution with totally different shape to the original distribution. At the end of the chain where many transformations have been performed, a complicated distribution could be emerged, where the complicated distribution would be a target distribution to be learnt at the first place. Continuing with the example described above, the target distribution here could be a complicated distribution for human face images, where the distribution could have hundreds of dimensions. Within NF, there would be many parameters to be adjusted during the learning process, hence

NF is a flexible framework to learn arbitrarily complex distributions. Apart from generative modelling, normalising flow applications also include density estimation, outlier detection, prior construction and dataset summarisation [20].

Related Work

The earliest use of NF concepts can be traced back to [15] for whitening transformation. It is a technique for data pre-processing in which it removes correlation within data dimensions and turn the data into having a similar structure to white noise. [4] proposed Gaussianization, which makes use of whitening transformation as a technique for density estimation instead of feature prepossessing. The work in [34] and [33] established and introduced the modern conception of NFs, formalising the them as a chain of simple transformations composed together. NFs were later popularised by [8] for density estimation, and [30] for variational inference.

Application of NFs are wide range. For example, the image generation method Glow as proposed by [19], where in their experiments they generated synthesize realistic images of human faces. Inspired by the work, [21] proposed VideoFlow, where they constructed flow-based video prediction models that could be used for videos generations. [16] used a flow-based generative model for raw audio synthesis. Flow-based models were also used in reinforcement learning. [25] and [35] extend the soft actor-critic framework of reinforcement learning to a richer class of distribution via NFs, allowing the better learning of policies. NFs also have use in scientific research, such as in [28] for condensed matter physics and [37] for free energy estimation.

In the remaining of this section, we would discuss in detail what NF actually is and how to train them. Also, we would discuss the parts of NF that we will be using in later chapters for experiments.

2.2.1 Basic Definitions

The NF can be thought of as a chain of a flow of successive invertible and differentiable transformations that take points in one space to another, with those spaces related by the transformations. At one end of the NF is the latent space Z with random variable $\boldsymbol{z} \in Z$, and at the other end is the data space X with random variable $\boldsymbol{x} \in X$. In both spaces there both lived probability distributions $p_0(\boldsymbol{z})$ and $p(\boldsymbol{x})$ for Z and X spaces respectively, with the respective random variables $\boldsymbol{z}$ and $\boldsymbol{x}$ distributed according to:

$$\begin{aligned} \boldsymbol{z} &\sim p_0(\boldsymbol{z}) \\ \boldsymbol{x} &\sim p(\boldsymbol{x}) \end{aligned} \tag{2.27}$$

$p_0(\boldsymbol{z})$ is the base distribution and it is usually a simple distribution, such as a standard normal distribution, while $p(\boldsymbol{x})$ is usually a much more complicated distribution that approximates the true distribution. $p(\boldsymbol{x})$ is generated after passing $p_0(\boldsymbol{z})$ through the NF. NF ultimately transforms points in the latent space to points in the data space, hence NF can be represented by a transformation f which accepts points in latent space as argument:

$$\boldsymbol{x} = f(\boldsymbol{z}) \tag{2.28}$$

The above constitutes a compact representation of the NF. f consists of a chain of invertible or bijective transformations that are combined together, connecting latent variable $\boldsymbol{z}$ to $\boldsymbol{x}$ by pushing $\boldsymbol{z}$ in the forward direction. Given k transformations connected together and with each individual transformation be considered as a layer, with the i^{th} layer of transformation denoted as f_i, then f can be expressed as a composition of k functions:

$$f(\boldsymbol{z}) = f_k \circ f_{k-1} \circ \ldots \circ f_1(\boldsymbol{z}) \tag{2.29}$$

Equivalently, one can express the transformation of NF in the backward direction, by performing an inverse:

$$\boldsymbol{z} = f^{-1}(\boldsymbol{x}) \tag{2.30}$$

and the k composition of functions in the backward direction become

$$f^{-1}(\boldsymbol{x}) = f_1^{-1} \circ f_2^{-1} \circ \ldots \circ f_k^{-1}(\boldsymbol{x}) \tag{2.31}$$

k here is a hyperparameter that is chosen by the user. Intuitively one would expect the larger the value of k is, f would have increased ability to express complicated transformation. In the literature, each layer of transformation is called a coupling layer and therefore a complete NF would consist of many coupling layers coupled together.

For notational convenience, we define the inverse of f be a new function g such that

$$g(\boldsymbol{x}) := f^{-1}(\boldsymbol{x}) \tag{2.32}$$

then we also have the backward version of NF analogous to (2.28)

$$\boldsymbol{z} = g(\boldsymbol{x}) \tag{2.33}$$

and the composition of k functions in the backward direction analogous to (2.29)

$$g(\boldsymbol{x}) = g_1 \circ g_2 \circ \ldots \circ g_k(\boldsymbol{x}) \tag{2.34}$$

2.2.2 Change of Variables

Now that we have established the relationship between points in latent space and the corresponding points in data space, we now turn to establish the relationship between latent and data space distributions $p_0(\boldsymbol{z})$ and $p(\boldsymbol{x})$. With the points in the Z space being transformed to the points in X space, intuitively if a small patch of points in Z space are being spread out in X after transformation, then one would expect the probability density in the corresponding area in X would decrease after transformation. Vice versa, if the patch of points in Z are being shrunk in X, the density would increase after transformation. This intuition is captured by the change of variables theorem in probability.

For random variables $\boldsymbol{x} \in \mathbb{R}^d$ and $\boldsymbol{z} \in \mathbb{R}^d$ that are related by a transformation:

$$\boldsymbol{z} = g(\boldsymbol{x}) \tag{2.35}$$

and distributed according to

$$\begin{aligned} \boldsymbol{z} &\sim p_0(\boldsymbol{z}) \\ \boldsymbol{x} &\sim p(\boldsymbol{x}) \end{aligned} \tag{2.36}$$

The change of variables formula in probability states that the density in X space is the density in Z space multiplied by a factor of the absolute value of the Jacobian determinant:

$$p(\boldsymbol{x}) = p_0(\boldsymbol{z}) \left| \det \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}^T} \right| \tag{2.37}$$

where $\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}^T}$ is the Jacobian matrix, with

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}^T} = \begin{pmatrix} \frac{\partial z_1}{\partial x_1} & \cdots & \frac{\partial z_1}{\partial x_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial z_d}{\partial x_1} & \cdots & \frac{\partial z_d}{\partial x_d} \end{pmatrix} \tag{2.38}$$

The change of variable formula can then be expressed solely in terms of $\boldsymbol{x}$:

$$p(\boldsymbol{x}) = p_0(g(\boldsymbol{x})) \left| \det \frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}^T} \right| \tag{2.39}$$

Since we already have defined the base distribution in the latent space $p_0(\boldsymbol{z})$, with the NF transformation g also known, we have complete information to compute the density of $\boldsymbol{x}$ in data space.

2.2.3 Coupling Layers

We have discussed the general structure of NF where the overall structure of the NF consists of multiple coupling layers stacked together. Here we will define the general structure of a coupling layer. For a coupling layer, it should satisfy two mathematical properties:

1. It is easily invertible.
2. Its Jacobian determinant is easy to compute.

These two properties will be useful in computing the quantities we introduced previously. Property 1. will be useful when we want to push a data point in the data space backward through the NF and find its corresponding point in the latent space. Property 2. will be useful when we want to find the likelihood of data, which makes use of the formula in equation (2.39) that has a Jacobian determinant term.

[8] introduced a general form of coupling layer such that the coupling layer will always satisfy the two mathematical properties above. Given a coupling layer with transformation $f : \boldsymbol{x} \mapsto \boldsymbol{y}$, with $\boldsymbol{x} \in \mathbb{R}^D$ as input and $\boldsymbol{y} \in \mathbb{R}^D$ as output:

$$\begin{aligned} \boldsymbol{y}_{1:d} &= \boldsymbol{x}_{1:d} \\ \boldsymbol{y}_{d+1:D} &= m_1(\boldsymbol{x}_{d+1:D}; m_2(\boldsymbol{x}_{1:d})) \end{aligned} \tag{2.40}$$

where m_1 and m_2 are functions and $m_1 : \mathbb{R}^{D-d} \times m_2(\mathbb{R}^d) \mapsto \mathbb{R}^{D-d}$. m_1 will be invertible by construction with respect to its first argument given the second. In words, the first d dimensions of the input of the coupling layer stays the same, where the final $D - d$ dimensions are changed. The way the coupling layer is defined satisfies the mathematical properties we required above, as we can demonstrate that

1. By direct inversion we get

$$\begin{aligned} \boldsymbol{x}_{1:d} &= \boldsymbol{y}_{1:d} \\ \boldsymbol{x}_{d+1:D} &= m_1^{-1}(\boldsymbol{y}_{d+1:D}; m_2(\boldsymbol{y}_{1:d})) \end{aligned} \tag{2.41}$$

   hence the coupling layer is easily invertible.

2. The Jacobian matrix is

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}^T} = \begin{pmatrix} \mathbf{I}_d & \mathbf{0} \\ \frac{\partial \boldsymbol{y}_{d+1:D}}{\partial \boldsymbol{x}_{1:d}^T} & \frac{\partial \boldsymbol{y}_{d+1:D}}{\partial \boldsymbol{x}_{d+1:D}^T} \end{pmatrix} \tag{2.42}$$

   which is a triangular matrix. By the property of matrix determinant, the determinant of a triangular matrix is the product of the elements in its diagonal.

Hence the Jacobian determinant of the coupling is easy to compute.

While we have outlined a general structure of a coupling layer that satisfies our required mathematical properties, we have not defined what the functions m_1 and m_2 are. These functions will further define the structure of the NF.

For example, [8] introduced NICE, making use of additive coupling layers:

$$\begin{aligned} \boldsymbol{y}_{1:d} &= \boldsymbol{x}_{1:d} \\ \boldsymbol{y}_{d+1:D} &= \boldsymbol{x}_{d+1:D} + m(\boldsymbol{x}_{1:d}) \end{aligned} \tag{2.43}$$

where m is a function parameterised by a neural network. These are easily invertible and with the Jacobian matrix identically equal to 1. [9] introduced RealNVP with affine coupling layers:

$$\begin{aligned} \boldsymbol{y}_{1:d} &= \boldsymbol{x}_{1:d} \\ \boldsymbol{y}_{d+1:D} &= \boldsymbol{x}_{d+1:D} \odot \exp(s(\boldsymbol{x}_{1:d})) + t(\boldsymbol{x}_{1:d}) \end{aligned} \tag{2.44}$$

where s and t are functions that are parameterised by neural networks. This coupling layer is easily invertible and its Jacobian determinant is $\exp\left(\sum_j s(\boldsymbol{x}_{1:d})_j\right)$.

2.2.4 Combining Coupling Layers

The NF works by combining all coupling layers together. Due to the special mathematical properties of the coupling layers that we have imposed, it makes the calculation of the data likelihood much easier to compute, which makes use of the change of variable formula in (2.39).

For convenience we can re-express the transformation of NF and make it more explicit. Remembering that we want to push a random variable $\boldsymbol{z}$ through one end of the NF to obtain $\boldsymbol{x}$ at the other end, with forward and backward coupling layer transformations f and g respectively, then for k coupling layers the NF in the forward direction (2.29) and NF in the backward direction (2.34) can be re-written as:

$$\begin{aligned} \boldsymbol{z}_0 &= \boldsymbol{z} \\ \boldsymbol{z}_k &= \boldsymbol{x} \\ \boldsymbol{z}_i &= f_i(\boldsymbol{z}_{i-1}) \\ \boldsymbol{z}_{i-1} &= g_i(\boldsymbol{z}_i) \end{aligned} \tag{2.45}$$

for $i \in \{1, \ldots, k\}$.

Making use of the chain rule for the Jacobian matrix

$$\frac{\partial g_i \circ g_{i+1}(\boldsymbol{z}_{i+1})}{\partial \boldsymbol{z}_{i+1}^T} = \frac{\partial g_i(\boldsymbol{z}_i)}{\partial \boldsymbol{z}_i^T} \cdot \frac{\partial g_{i+1}(\boldsymbol{z}_{i+1})}{\partial \boldsymbol{z}_{i+1}^T} \tag{2.46}$$

the multiplicative rule for determinant

$$\det(AB) = \det(A) \cdot \det(B) \tag{2.47}$$

and the inverse rule for Jacobian determinant

$$\det\left(\frac{\partial g_i(\boldsymbol{z}_i)}{\partial \boldsymbol{z}_i^T}\right) = \det\left(\frac{\partial f_i(\boldsymbol{z}_{i-1})}{\partial \boldsymbol{z}_{i-1}^T}\right)^{-1} \tag{2.48}$$

then the change of variable formula (2.39) can be written as

$$\begin{aligned}
p(\boldsymbol{x}) &= p_0(g(\boldsymbol{x}))\left|\det\frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}}\right| \\
&= p_0(g(\boldsymbol{x}))\left|\det\left(\frac{\partial g_1 \circ \ldots \circ g_k(\boldsymbol{z}_k)}{\partial \boldsymbol{z}_k^T}\right)\right| \\
&= p_0(g(\boldsymbol{x}))\left|\det\left(\frac{\partial g_1(\boldsymbol{z}_1)}{\partial \boldsymbol{z}_1^T} \cdot \ldots \cdot \frac{\partial g_k(\boldsymbol{z}_k)}{\partial \boldsymbol{z}_k^T}\right)\right| \\
&= p_0(g(\boldsymbol{x})) \cdot \prod_{i=1}^{k}\left|\det\frac{\partial g_i(\boldsymbol{z}_i)}{\partial \boldsymbol{z}_i^T}\right| \\
&= p_0(g(\boldsymbol{x})) \cdot \prod_{i=1}^{k}\left|\det\frac{\partial f_i(\boldsymbol{z}_{i-1})}{\partial \boldsymbol{z}_{i-1}^T}\right|^{-1}
\end{aligned} \tag{2.49}$$

which makes repeated use of the chain rule on line 3, the multiplicative rule of determinant on the second last line and the inverse Jacobian rule on the last line. Or one can take log on both sides to get a more convenient form for computation

$$\log p(\boldsymbol{x}) = \log p_0(g(\boldsymbol{x})) - \sum_{i=1}^{k}\log\left|\det\frac{\partial f_i(\boldsymbol{z}_{i-1})}{\partial \boldsymbol{z}_{i-1}^T}\right| \tag{2.50}$$

The result is that the log density of a data point $\boldsymbol{x}$ is the sum of all the log Jacobian determinants of all coupling layers, and the log base density in the latent space for the corresponding point. By the property that the Jacobian determinant and the inverse of a coupling layer are easy to compute, one can compute the log data density very easily. Making use of this relation, not only one can have the data distribution approximated by the NF analytically, one can also compute useful quantities and perform downstream tasks with ease, such as perform maximum likelihood, maximum a posteriori estimation or Bayesian inference.

2.2.5 Learning

We now have all the pieces to describe the learning process of a NF. For a NF, we have previously define implicitly without writing out explicitly that the transformation f is adjustable and it has underlying parameters that dictate what the transformation would be. By adjusting the parameters of f the resulting transformation would change. In the learning process, f would converge to a point such that the final learnt transformation would transform the base distribution to the target distribution as closely as possible. If we let $\boldsymbol{\theta}$ be the parameters of f, then we can write out explicitly the dependence of $\boldsymbol{\theta}$ for the parameterised forward and backward transformations:

$$\begin{aligned} \boldsymbol{x} &= f(\boldsymbol{z}; \boldsymbol{\theta}) \\ \boldsymbol{z} &= g(\boldsymbol{x}; \boldsymbol{\theta}) \end{aligned} \tag{2.51}$$

with f corresponds to (2.28) and g corresponds to (2.33). We can also elaborately write out the dependence for each coupling layer, then for the i^{th} coupling layer its dependence on the i^{th} set of parameters $\boldsymbol{\theta}_i$ is:

$$\begin{aligned} \boldsymbol{z}_i &= f_i(\boldsymbol{z}_{i-1}; \boldsymbol{\theta}_i) \\ \boldsymbol{z}_{i-1} &= g_i(\boldsymbol{z}_i; \boldsymbol{\theta}_i) \end{aligned} \tag{2.52}$$

for $i \in \{1, \ldots, k\}$ with $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k]$.

With the NF parameterised, the resulting distribution at the data space would also be parameterised by $\boldsymbol{\theta}$, which we can write out its dependency explicitly as $p(\boldsymbol{x}|\boldsymbol{\theta})$. Then from (2.50), the log density for a data point $\boldsymbol{x}$ by the parameterised distribution is:

$$\log p(\boldsymbol{x}|\boldsymbol{\theta}) = \log p_0(g(\boldsymbol{x}; \boldsymbol{\theta})) - \sum_{i=1}^{k} \log \left| \det \frac{\partial f_i(\boldsymbol{z}_{i-1}; \boldsymbol{\theta})}{\partial \boldsymbol{z}_{i-1}^T} \right| \tag{2.53}$$

Given a training dataset $\mathcal{D}$, we can learn a NF of the data distribution by maximising the data likelihood with respect to $\boldsymbol{\theta}$, which corresponds to using the maximum likelihood estimation method. Assuming each data point in $\mathcal{D}$ is identically and independently distributed, the data log likelihood is the summation of individual log density:

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{\boldsymbol{x} \in \mathcal{D}} \log p(\boldsymbol{x}|\boldsymbol{\theta}) \tag{2.54}$$

If we define a loss function $\mathcal{L}(\boldsymbol{\theta}; \mathcal{D})$ as the negative log likelihood of data:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) &= -\frac{1}{|D|} \log p(\mathcal{D}|\boldsymbol{\theta}) \\ &= -\frac{1}{|D|} \sum_{\boldsymbol{x} \in D} \log p(\boldsymbol{x}|\boldsymbol{\theta}) \end{aligned} \tag{2.55}$$

then by minimising $\mathcal{L}(\boldsymbol{\theta}; \mathcal{D})$ with respect to $\boldsymbol{\theta}$ given training dataset, we are equivalently maximising the data log likelihood. $\mathcal{L}(\boldsymbol{\theta}; \mathcal{D})$ can be minimised by employing any standard optimisation techniques such as using the method of gradient descent.

With the data log likelihood maximised, this means that the resulting distribution $p(\boldsymbol{x}|\boldsymbol{\theta})$ of the NF would approximate the true data distribution which generates the training dataset $\mathcal{D}$. With $p(\boldsymbol{x}|\boldsymbol{\theta})$, one can draw samples from it and the samples would approximately distributed as the true distribution that the NF learnt from.

2.2.6 Sampling

In a NF, after we have learnt a distribution $p(\boldsymbol{x}|\boldsymbol{\theta})$ that approximates the true data distribution, one would want to draw samples from it. We can achieve this by first drawing samples $\boldsymbol{z}$ in the latent space from the base distribution

$$\boldsymbol{z} \sim p_0(\boldsymbol{z}) \tag{2.56}$$

then make use of the learnt transformation f in the NF to transform the samples in the latent space to the data space

$$\boldsymbol{x} = f(\boldsymbol{z}; \boldsymbol{\theta}) \tag{2.57}$$

then we have the corresponding samples $\boldsymbol{x}$ in the data space, which will distribute according to the learnt distribution

$$\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{\theta}) \tag{2.58}$$

Note that by construction of the NF we have put the base distribution $p_0(\boldsymbol{z})$ to be a simple distribution, thus sampling from it should also be easy.

Chapter 3

Methodology

In this chapter, we combine the frameworks of normalising flow and Bayesian inference introduced in the previous chapter, and derive relevant experiments that allow us to perform uncertainty estimations on normalising flow parameters. We will describe what it means to perform Bayesian inference on the flow parameters. Through the methods of MAP estimation, deep ensembles, MC dropout and mean-field variational inference we will be able to perform uncertainty estimation on the parameters. After that we will describe and quantify what it means to have well calibrated uncertainty estimations.

3.1 Normalising Flows Notations

In the rest of this chapter we will use the notations of a normalising flow (NF) defined below, same as the notations we used in the chapter on background on NF. For a NF with k coupling layers, with forward and backward deterministic invertible transformations denoted as f and g respectively, and with $\boldsymbol{z}$ being the latent variable in the latent space and the corresponding variable $\boldsymbol{x}$ in the data space, the equations that define the NF are:

$$\begin{aligned} \boldsymbol{x} &= f(\boldsymbol{z};\boldsymbol{\theta}) = f_k \circ \dots f_1(\boldsymbol{z};\boldsymbol{\theta}) \\ \boldsymbol{z} &= g(\boldsymbol{x};\boldsymbol{\theta}) = g_1 \circ \dots g_k(\boldsymbol{x};\boldsymbol{\theta}) \end{aligned} \tag{3.1}$$

with the inverse transformation $g(\boldsymbol{x};\boldsymbol{\theta}) = f^{-1}(\boldsymbol{x};\boldsymbol{\theta})$.

The equations that define individual coupling layer are:

$$\begin{aligned} \boldsymbol{z}_i &= f_i(\boldsymbol{z}_{i-1}; \boldsymbol{\theta}_i) \\ \boldsymbol{z}_{i-1} &= g_i(\boldsymbol{z}_i; \boldsymbol{\theta}_i) \\ \boldsymbol{z}_0 &= \boldsymbol{z} \\ \boldsymbol{z}_k &= \boldsymbol{x} \end{aligned} \tag{3.2}$$

for all layers $i = \{1, \ldots, k\}$. This also defines the inverse of each layer $g_i(\boldsymbol{x}_i; \boldsymbol{\theta}_i) = f_i^{-1}(\boldsymbol{x}_i; \boldsymbol{\theta}_i)$.

The NF represent a probability distribution on $\boldsymbol{x}$ with parameters $\boldsymbol{\theta}$, with a latent base distribution in the latent space on $\boldsymbol{z}$. The relations are:

$$\begin{aligned} \boldsymbol{z} &\sim p_0(\boldsymbol{z}) \\ \boldsymbol{x} &\sim p(\boldsymbol{x}|\boldsymbol{\theta}) \end{aligned} \tag{3.3}$$

where these distributions are connecting by the transformation of the NF.

3.2 Posterior distribution of Normalising Flows Parameters

Given a model $\mathcal{M}$ which is parameterised by normalising flow (NF). This model consists of all the possible probability distributions that the NF can represent with parameters $\boldsymbol{\theta}$. $\mathcal{M}$ can be thought of as an implicit generative model that latent data is first generated from an underlying distribution. With a sample $\boldsymbol{z}$ in the latent space drawn, one pushes this point through f and a sample $\boldsymbol{x}$ will be produced in the data space, with $\boldsymbol{x}$ distributed according to the distribution that the NF represents. The process is described as:

$$\begin{aligned} \boldsymbol{z} &\sim p_0(\boldsymbol{z}) \\ \boldsymbol{x} &= f(\boldsymbol{z}; \boldsymbol{\theta}) \\ \boldsymbol{x} &\sim p(\boldsymbol{x}|\boldsymbol{\theta}) \end{aligned} \tag{3.4}$$

The actual distribution that generates observable data is $p(\boldsymbol{x}|\boldsymbol{\theta})$. In this report, our main objective is to infer the distribution of the NF parameters $\boldsymbol{\theta}$ given data, and we can do this by Bayesian inference.

Before observing any data, we assume a prior on the distribution of $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} \sim p(\boldsymbol{\theta}) \tag{3.5}$$

Then given that we observe a dataset $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^N$ with each data point independent and identically distributed (i.i.d.). We make use of this knowledge to update our belief on the distribution of $\boldsymbol{\theta}$. Our new updated belief is captured by the posterior distribution which we can compute using the Bayes' theorem:

$$\begin{aligned} p(\boldsymbol{\theta}|\mathcal{D}) &= \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \\ &= \frac{\prod_{i=1}^N p(\boldsymbol{x}_i|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int \prod_{i=1}^N p(\boldsymbol{x}_i|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}} \end{aligned} \tag{3.6}$$

Due to the complicated interdependence of the NF parameters in $\boldsymbol{\theta}$, it is often infeasible to compute their posterior distribution analytically, this is because the marginal likelihood in the denominator requires integrating out $\boldsymbol{\theta}$ which is intractable, implying that the resulting posterior is intractable. Therefore we would instead compute the posterior approximately, by using approximate inference methods.

3.3 Maximum a Posterior Estimation on Normalising Flows Parameters

Instead of computing the full distribution of the posterior distribution of NF parameters $\boldsymbol{\theta}$, we can perform a point estimate on them using the maximum a posteriori (MAP) estimation. It is expected that the true posterior distribution will be highly multimodal with a highly non-linear surface. When we perform MAP estimate, our aim is to find the global maximum point in the actual posterior distribution. However due to the random nature of MAP estimation at initialisation, it will not usually be the case that the maximum point $\boldsymbol{\theta}^*$ found at the end of the MAP estimation would be the one that actually corresponds to the global optimum of the true posterior distribution. Instead it will be a point corresponds to one of the local modes in the true posterior distribution. Recall that in MAP estimation we want to compute the following:

$$\begin{aligned} \boldsymbol{\theta}^*_{\text{MAP}} &= \underset{\boldsymbol{\theta}}{\text{argmax}} \;\; p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \\ &= \underset{\boldsymbol{\theta}}{\text{argmax}} \;\; \log p(\mathcal{D}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) \\ &= \underset{\boldsymbol{\theta}}{\text{argmax}} \;\; \sum_{i=1}^N \log p(\boldsymbol{x}_i|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) \end{aligned} \tag{3.7}$$

We can substitute the data likelihood term based on our NF model:

$$\begin{aligned}\boldsymbol{\theta}^*_{\text{MAP}} &= \underset{\boldsymbol{\theta}}{\text{argmax}} \sum_{i=1}^{N} \log p(\boldsymbol{x}_i|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) \\ &= \underset{\boldsymbol{\theta}}{\text{argmax}} \sum_{i=1}^{N} \left(\log p_0(g(\boldsymbol{x}_i;\boldsymbol{\theta})) - \sum_{j=1}^{k} \log \left| \det \frac{\partial f_j(\boldsymbol{z}_{j-1}^{(i)};\boldsymbol{\theta})}{\partial \boldsymbol{z}_{j-1}^T} \right| \right) + \log p(\boldsymbol{\theta})\end{aligned} \tag{3.8}$$

where we have substituted (2.53) in the last line. We also assume the prior of $\boldsymbol{\theta}$ is a standard Gaussian:

$$p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\mathbf{0}, \mathbf{I}) \tag{3.9}$$

If we define a loss as a function of the parameters:

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = -\frac{1}{N} \left[\sum_{i=1}^{N} \left(\log p_0(g(\boldsymbol{x}_i;\boldsymbol{\theta})) - \sum_{j=1}^{k} \log \left| \det \frac{\partial f_j(\boldsymbol{z}_{j-1}^{(i)};\boldsymbol{\theta})}{\partial \boldsymbol{z}_{j-1}^T} \right| \right) + \log p(\boldsymbol{\theta}) \right] \tag{3.10}$$

Then we can minimise the loss and find the optimum $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}^*_{\text{MAP}} = \underset{\boldsymbol{\theta}}{\text{argmin}}\, \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) \tag{3.11}$$

We can use any standard optimisation technique such as gradient descent to minimise the loss. If the optimisation is successful then the final $\boldsymbol{\theta}^*_{\text{MAP}}$ obtained would be a point in the true posterior distribution of the NF parameters that corresponds to a local mode.

3.4 Predictive Distributions and their Uncertainty Estimations

Given a dataset, a predictive distribution is an estimate to an unknown distribution that generates the dataset. In this section we describe how we can produce predictive distributions under the NF framework using the methods of deep ensemble, Monte Carlo dropout and mean-field variational inference. For a model that produces a predictive distribution, we can treat the predictive distribution as a prediction of the model. With this prediction, we can also quantify the uncertainty of this model in making of the prediction, that is, the uncertainty estimation of the prediction.

In the following, we will describe the methods that we will use to obtain predictive distributions in the NF framework, with uncertainty estimations associated with the distributions.

3.4.1 Deep Ensemble

In the deep ensemble method, a predictive distribution is obtained by taking a simple average of all the members in the ensemble, as we have described in (2.8). Applying the deep ensemble method to the NF framework, we want to produce a predictive distribution using an ensemble of probability distributions parameterised by NF models. Then the deep ensemble in the NF framework would consist of multiple MAP estimations of the parameters of the NF models, with each set of parameters corresponding to a NF distribution. Then the final predictive distribution is the average of the NF distributions in the deep ensemble.

With the predictive distribution obtained, we also want to get the uncertainty estimation on this predictive distribution. The uncertainty estimation can be quantified by the standard deviation metric. To perform this, standard deviation is computed for the members in the deep ensemble, that is, standard deviation is calculated for the NF distributions contained within the ensemble.

3.4.2 Monte Carlo Dropout

In the Monte-Carlo (MC) dropout method, a predictive distribution is obtained similarly to the deep ensemble method. The predictive distribution is calculated by taking a simple average of a set of probability distributions that were obtained with dropout turned on at test time, as described in (2.9). In the NF framework, this means that we will first perform MAP estimation on a NF model with dropout turned on, then also with dropout turned on at test time, multiple different distributions that are parameterised by the NF model can be drawn from the dropout mechanism. Then the final predictive distribution is the average of the drawn NF distributions.

The process of performing uncertainty estimation on the predictive distribution for this MC dropout method is conceptually equivalent the process in the deep ensemble method. The uncertainty estimation is obtained by computing the standard deviation on all the NF distributions that were drawn for predictive distribution computation.

3.4.3 Mean-Field Variational Inference

In mean-field variational inference (MFVI) we approximate the true posterior $p(\boldsymbol{\theta}|\mathcal{D})$ distribution with a variational distribution $q_\lambda(\boldsymbol{\theta})$, with λ being the variable parameters of the distribution. This is done by minimising the KL divergence $D_{KL}(q_\lambda(\boldsymbol{\theta}) \parallel p(\boldsymbol{\theta}|\mathcal{D}))$ between the two distributions. By (2.19), to achieve this minimisation we maximise

the evidence lower bound (ELBO) given by:

$$\mathbb{E}_{q_\lambda(\boldsymbol{\theta})}\left[\log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q_\lambda(\boldsymbol{\theta})}\right] \tag{3.12}$$

Since we are also making the mean-field approximation, $q_\lambda(\boldsymbol{\theta})$ will be a Gaussian with independent 1-d Gaussian components, one at each dimension. Equivalently, $q_\lambda(\boldsymbol{\theta})$ will have a diagonal covariance matrix, and we can write:

$$q_\lambda(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2)) \tag{3.13}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ are vectors of means and variances that made up from all individual 1-d Gaussians, and they will be the parameters of $q_\lambda(\boldsymbol{\theta})$ with $\lambda = \{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$.

Given an i.i.d. dataset $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^N$, the ELBO for our NF model with parameters $\boldsymbol{\theta}$ can be expressed as:

$$\begin{aligned}
&\mathbb{E}_{q_\lambda(\boldsymbol{\theta})}\left[\log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q_\lambda(\boldsymbol{\theta})}\right] \\
&= \mathbb{E}_{q_\lambda(\boldsymbol{\theta})}\left[\log p(\boldsymbol{\theta}, \mathcal{D}) - \log q_\lambda(\boldsymbol{\theta})\right] \\
&= \mathbb{E}_{q_\lambda(\boldsymbol{\theta})}\left[\log p(\mathcal{D}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log q_\lambda(\boldsymbol{\theta})\right] \\
&= \mathbb{E}_{q_\lambda(\boldsymbol{\theta})}\left[\sum_{i=1}^N \log p(\boldsymbol{x}_i|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log q_\lambda(\boldsymbol{\theta})\right]
\end{aligned} \tag{3.14}$$

with the terms in last line of this equation:

$$\begin{aligned}
\log p(\boldsymbol{x}_i|\boldsymbol{\theta}) &= \log p_0(g(\boldsymbol{x}_i; \boldsymbol{\theta})) - \sum_{j=1}^k \log \left|\det \frac{\partial f_j(\boldsymbol{z}_{j-1}^{(i)}; \boldsymbol{\theta})}{\partial \boldsymbol{z}_{j-1}^T}\right| \\
p(\boldsymbol{\theta}) &= \mathcal{N}(\boldsymbol{\theta}|\mathbf{0}, \mathbf{I}) \\
q_\lambda(\boldsymbol{\theta}) &= \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2)
\end{aligned} \tag{3.15}$$

where the log likelihood term is the consequence of the NF model as in (2.53). Also, we have also assumed that the prior of $\boldsymbol{\theta}$ is a standard Gaussian.

Making use of the reparameterisation trick

$$\begin{aligned}
\boldsymbol{\theta} &= \boldsymbol{\theta}(\lambda, \boldsymbol{\epsilon}) = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} \\
\boldsymbol{\epsilon} &\sim p(\boldsymbol{\epsilon}) = \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{I})
\end{aligned} \tag{3.16}$$

the ELBO becomes:

$$\mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\sum_{i=1}^{N}\log p(\boldsymbol{x}_i|\boldsymbol{\theta}(\lambda,\boldsymbol{\epsilon}))+\log p(\boldsymbol{\theta}(\lambda,\boldsymbol{\epsilon}))-\log q_\lambda(\boldsymbol{\theta}(\lambda,\boldsymbol{\epsilon}))\right] \tag{3.17}$$

To maximise the ELBO, we can define a loss function as the average negative ELBO per data, as a function of variational distribution parameters $\lambda = \{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$:

$$\mathcal{L}(\lambda;\mathcal{D},\boldsymbol{\epsilon}) = -\frac{1}{N}\left[\sum_{i=1}^{N}\log p(\boldsymbol{x}_i|\boldsymbol{\theta}(\lambda,\boldsymbol{\epsilon}))+\log p(\boldsymbol{\theta}(\lambda,\boldsymbol{\epsilon}))-\log q_\lambda(\boldsymbol{\theta}(\lambda,\boldsymbol{\epsilon}))\right] \tag{3.18}$$

This loss can be minimised using any standard optimisation techniques, with the modification that at every gradient step, sample a new $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and use this newly sampled $\boldsymbol{\epsilon}$ to evaluate the gradient. Notice that the loss $\mathcal{L}(\lambda; \mathcal{D}, \boldsymbol{\epsilon})$ will be changing every time we sample a new $\boldsymbol{\epsilon}$. However, the λ obtained after optimisation for this loss will be an unbiased estimate of the λ that maximises the ELBO.

With the loss minimised this corresponds to the ELBO being maximised. This in turn means that the KL divergence between the variational distribution and the true posterior distribution for the NF parameters $\boldsymbol{\theta}$ is minimised. Hence we will get a resulting variational distribution that approximates the true posterior.

Now that we have optimised for an approximation to the true posterior, we can now compute the predictive distribution. Using the law of probability, a predictive distribution can be obtained using the formula:

$$p(\boldsymbol{x}|\mathcal{D}) = \int p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta} \tag{3.19}$$

In the MFVI method, since we have approximated the posterior with the variational distribution, we can approximate the predictive distribution by replacing the posterior term with the variational distribution:

$$\begin{aligned} p(\boldsymbol{x}|\mathcal{D}) &\approx \int p(\boldsymbol{x}|\boldsymbol{\theta})q(\boldsymbol{\theta})d\boldsymbol{\theta} \\ &= \mathbb{E}_{q(\boldsymbol{\theta})}\left[p(\boldsymbol{x}|\boldsymbol{\theta})\right] \end{aligned} \tag{3.20}$$

We can Monte Carlo estimate this term by sampling the NF parameters $\boldsymbol{\theta}$ from the variational distribution $q(\boldsymbol{\theta})$, with the help of the reparameterisation trick. Then given

that we sample M sets of NF parameters, the predictive distribution is:

$$\begin{aligned} p(\boldsymbol{x}|\mathcal{D}) &\approx \frac{1}{M}\sum_{i=1}^{M} p(\boldsymbol{x}|\boldsymbol{\theta}_i) \\ &= \frac{1}{M}\sum_{i=1}^{M} p(\boldsymbol{x}|\boldsymbol{\theta}(\lambda, \boldsymbol{\epsilon}_i)) \end{aligned} \tag{3.21}$$

The last line means that sampling $\boldsymbol{\theta}_i$ is equivalent to first sampling $\boldsymbol{\epsilon}_i \sim p(\boldsymbol{\epsilon})$, then use the reparameterisation trick to compute the corresponding $\boldsymbol{\theta}_i$.

In words, the equation tells is that the predictive distribution for this MFVI method is calculated by taking simple average of a set of sampled NF distributions, where each NF distribution in the set has its parameters sampled from the variational distribution. This process of taking the average of a set of distributions is similar to the processes as found in the deep ensemble and MC dropout method.

Similarly, with the predictive distribution obtained, we also want to get its uncertainty estimation. This can be done by computing the standard deviation on all the sampled NF distributions.

3.5 Uncertainty Quantification and Calibration

We have been looking at methods to estimate the uncertainty of NF parameters, however we did not specify how well calibrated the uncertainty estimations are. In the following we will follow the notion of calibration the same way as in [6], in which calibration is a frequentist notion of uncertainty which measures the difference between subjective predictions and long-run frequencies.

For a model that has learnt an uncertainty estimation, we could think of the uncertainty estimation as a distribution. Depending on the degree of confidence of the model, the distribution will be more spread out when the model is less confident, or more narrow when the model has high confidence. When the model has a high confidence, the model is certain about the location of the true value and it will represent its uncertainty by a narrow distribution, which translates to a narrow confidence interval. Vice versa, when the model has low confidence, it is very uncertain on the location of the true value and its uncertainty will be represented by a spread out distribution, which translates to a wide confidence interval.

In practical situations, a model could be uncalibrated in its uncertainty estimation, in which case it could either be under or over confident in its estimations. When the model is over-confident, it means that it has a confidence intervals that is more narrow

than it should be, meaning that for confidence intervals of 50%, the number of times true values lie within the confidence intervals will be well below 50%. Vice versa, when the model is under-confident, it means that it has a wider confidence interval than it should be, meaning that for 50% confidence intervals, the number of times the true values lie within the intervals will be well above 50%. A well calibrated uncertainty estimate will have the true values fall within the confidence intervals the right amount of times, with about 50% of true values fall within the 50% confidence intervals.

We can quantify how well calibrated the uncertainty estimation of a model is. The quantitative measure would tell us if the model is over or under confident or it is appropriately calibrated. Intuitively, a well calibrated uncertainty estimate should have the proportion of true values fall into its confidence intervals approximately equal to the probability that the confidence interval is enclosing. Consider a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ with feature vectors $\boldsymbol{x}_i$ and true values y_i. For a model that has learnt an uncertainty estimation, with its learnt confidence interval denoted as $c_p(\boldsymbol{x}_i)$ as a function of feature vector $\boldsymbol{x}_i$, with probability p contained inside the interval, a well calibrated model for uncertainty estimation should satisfy

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}[\boldsymbol{y}_i \in c_p(\boldsymbol{x}_i)] \approx p \tag{3.22}$$

where $\mathbb{I}[\cdot]$ is the indicator function.

Then an under-confident model on uncertainty estimate would have the proportion of true values lie within the confidence interval larger than p

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}[\boldsymbol{y}_i \in c_p(\boldsymbol{x}_i)] > p \tag{3.23}$$

and for an over-confident model smaller than p

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}[\boldsymbol{y}_i \in c_p(\boldsymbol{x}_i)] < p \tag{3.24}$$

Chapter 4

Experiments

In this chapter we will describe in detail the procedures to perform experiments for uncertainty estimation on normalising flow parameters. We will use the methods that we have developed in the previous chapter. We will describe how the experiments are setup and how results will be obtained. We will also describe the dataset that we will be using for training. The results from experiments will be presented in the next chapter.

4.1 Synthetic Datasets

In this report, we will carry out our experiments based on data that we can synthesise on our own. The synthetic datasets are based on [36] and the code for data generation is adapted from `https://github.com/kevin-w-li/deep-kexpfam/`.

The synthetic datasets consists of 7 idealised two-dimensional probability distributions that we have complete knowledge of their analytic forms. With that, we are able to compute their densities exactly at any points, and to sample from them and generate completely new datasets for training. These distributions cover a range of geometric complexities and multimodality that allow us to observe different behaviours when experiments are performed on them.

The distributions are Funnel, Banana, Ring, Square, Cosine, Mixture of Gaussians (MoG), Mixture of Rings (MoR). The densities of the distributions are visualised in figure 4.1.

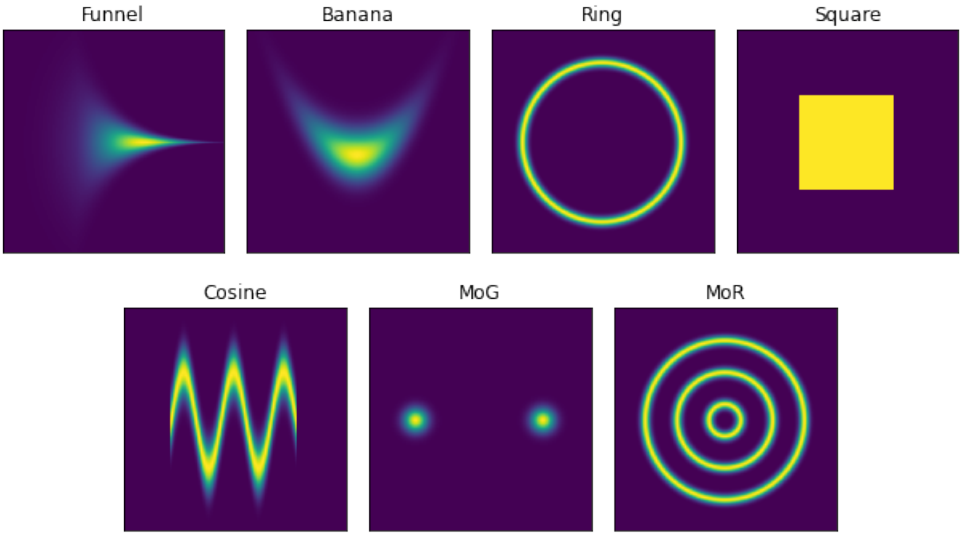


Figure 4.1: Densities for synthetic datasets.

In the remaining of this chapter, all the results obtained from the experiments will be based on models that trained on the datasets that sampled from these distributions.

4.2 Normalising Flows Setup

We will use a single NF architecture for all the experiments. The NF will have 20 coupling layers $k = 20$, with all coupling layers having the same general structure without sharing parameters. The coupling layer is defined by $f : \boldsymbol{x} \mapsto \boldsymbol{y}$, $\boldsymbol{x} \in \mathbb{R}^D$, $\boldsymbol{y} \in \mathbb{R}^D$:

$$\begin{aligned} \boldsymbol{y}_{1:d} &= \boldsymbol{x}_{1:d} \\ \boldsymbol{y}_{d+1:D} &= \boldsymbol{x}_{d+1:D} \odot \text{softplus}(s(\boldsymbol{x}_{1:d})) + t(\boldsymbol{x}_{1:d}) \end{aligned} \tag{4.1}$$

it is easily invertible with $f^{-1} : \boldsymbol{y} \mapsto \boldsymbol{x}$:

$$\begin{aligned} \boldsymbol{x}_{1:d} &= \boldsymbol{y}_{1:d} \\ \boldsymbol{x}_{d+1:D} &= \frac{(\boldsymbol{y}_{d+1:D} - t(\boldsymbol{y}_{1:d}))}{\text{softplus}(s(\boldsymbol{y}_{1:d}))} \end{aligned} \tag{4.2}$$

where the division is an elementwise operation. The softplus function is defined as:

$$\text{softplus}(\boldsymbol{x}) = \log(1 + e^{\boldsymbol{x}}) \tag{4.3}$$

s and t are neural networks with $s : \mathbb{R}^d \mapsto \mathbb{R}^{D-d}$ and $t : \mathbb{R}^d \mapsto \mathbb{R}^{D-d}$. s and t are both neural networks with dimensions $(d, 10, 10, D-d)$ and with ReLU [26] as activation functions for the hidden layers. The purpose of s and t are to scale and translate the input respectively. For the i^{th} coupling layer, the parameters in the corresponding s and t neural networks make up the parameters in this coupling layer $\boldsymbol{\theta}_i$.

Notice that the coupling layer defined above is similar to affine coupling layer as found in [9], however for numerical stability reason we replace the exponential term in the affine coupling layer to a softplus function.

This coupling layer has Jacobian matrix

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}^T} = \begin{pmatrix} \mathbf{I}_d & \mathbf{0} \\ \frac{\partial \boldsymbol{y}_{d+1:D}}{\partial \boldsymbol{x}_{1:d}^T} & \text{diag}(\text{softplus}(s(\boldsymbol{y}_{1:d}))) \end{pmatrix} \tag{4.4}$$

with easily computable Jacobian determinant

$$\det\left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}^T}\right) = \prod_{i=1}^{D-d} \text{softplus}(s(\boldsymbol{y}_{1:d}))_i \tag{4.5}$$

The base distribution for the NF is set to be a standard Gaussian

$$p_0(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\mathbf{0}, \mathbf{I}) \tag{4.6}$$

the complete NF is parameterised by $\boldsymbol{\theta} = \{\boldsymbol{\theta}_i\}_{i=1}^k$.

With the synthetic datasets that we are using for the experiments, the data dimension is $D = 2$. We are also setting for the NF that $d = 1$.

4.3 Training

With a NF parameterised by $\boldsymbol{\theta}$ and given dataset $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^N$, training is done by minimising the loss function given data $\mathcal{L}(\boldsymbol{\theta}; \mathcal{D})$:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmin}}\, \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) \tag{4.7}$$

the obtained $\boldsymbol{\theta}^*$ can be used for predictions at test time. All loss function minimisations are done by using the Adam [18] optimiser with recommended hyperparameters settings, with a learning rate of 0.001.

In the below, we describe how we perform training for each type of experiment, they

are MAP estimation, deep ensemble, MC dropout and MFVI. In each of them, learning will be repeated on different synthetic datasets from the 7 idealised distributions we described earlier. A dataset will be newly sampled every time we perform a training session.

4.3.1 Maximum a Posteriori Estimation

In MAP estimations, we simply minimise the loss function in (3.10). The training procedure is:

1. Initialise a NF with parameters initialised randomly.
2. Minimise loss with respect to NF parameters $\boldsymbol{\theta}$ as derived in (3.10)

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = -\frac{1}{N}\left[\sum_{i=1}^{N}\left(\log p_0(g(\boldsymbol{x}_i; \boldsymbol{\theta})) - \sum_{j=1}^{k}\log\left|\det\frac{\partial f_j(\boldsymbol{z}_{j-1}^{(i)}; \boldsymbol{\theta})}{\partial \boldsymbol{z}_{j-1}^{T}}\right|\right) + \log p(\boldsymbol{\theta})\right] \tag{4.8}$$

   by an Adam optimiser for a pre-specified number of iterations.

With the final trained parameters obtained, this set of parameters will represent a distribution for the NF. We will plot the densities of the learnt distributions for all synthetic datasets as results.

4.3.2 Deep Ensemble

For deep ensembles, we perform MAP estimations on a distribution the same way as before. However, we will repeat the MAP estimation multiple times to obtain multiple sets of NF parameters, each set of parameters will represent a distribution learnt by the NF. For this experiment we will repeat MAP estimations for 50 times. Collectively, the 50 MAP estimations will form an ensemble.

For the ensemble, a mean and a standard deviation of the learnt NF distributions within the ensemble will be plotted. The mean corresponds to estimating the original probability distributions, and the standard deviation corresponds to uncertainty of the NF distribution estimation.

4.3.3 Monte Carlo Dropout

For MC dropout, we perform MAP estimation on a distribution using the same training procedure for the MAP estimation experiment, with the exception that dropout is turned on for the neural networks within the NF. The dropout rate is set to be 0.1.

For each type of distribution that the MC dropout trained on, we will draw 100 NF distributions by allowing dropout to be turned on at test time, and the means and standard variations of the distributions will be plotted. The mean corresponds to estimating the original probability distribution, and the standard deviation corresponds to uncertainty of the NF distribution estimation.

4.3.4 Mean-Field Variational Inference

For MFVI we train a NF model with its parameters reparameterised using the reparameterisation trick. If we train the model from scratch using randomly initialised parameters, the NF model will not converge easily. Instead we will initialise a NF model with parameters from previous MAP estimations. The training procedure is:

1. Initialise NF parameters with parameters obtained from a MAP estimation.
2. Resample parameters from the reparameterisation trick and calculate the loss function derived in (3.18)

$$\mathcal{L}(\lambda; \mathcal{D}, \boldsymbol{\epsilon}) = -\frac{1}{N}\left[\sum_{i=1}^{N} \log p(\boldsymbol{x}_i|\boldsymbol{\theta}(\lambda, \boldsymbol{\epsilon})) + \log p(\boldsymbol{\theta}(\lambda, \boldsymbol{\epsilon})) - \log q_\lambda(\boldsymbol{\theta}(\lambda, \boldsymbol{\epsilon}))\right] \quad (4.9)$$

3. Take a gradient step on the loss with respect to λ using the Adam optimiser.
4. Repeat from step 2 until a pre-specified number of iterations.

For every trained NF using MFVI, due to the stochastic nature of the MFVI parameters, we can sample a set of NF parameters, with one set of parameters corresponds to one NF distribution. We sample 100 sets of them, which corresponds to sampling 100 NF distributions. With the sampled distributions, we compute the means and standard deviations from them and plot the results. The mean corresponds to an estimation to the original distribution and the standard deviation corresponds to the uncertainty of the estimation.

4.4 Uncertainty Quantification and Calibration

Uncertainty estimations are obtained from the methods of deep ensemble, MC dropout and MFVI in terms of standard deviations, these correspond to how precise the uncertainties are. However the precision of the uncertainties do not tell us whether the uncertainties are accurate or not. Consider the confidence intervals that have fixed probability enclosed. Uncertainties could be precise but inaccurate, in which case confidence intervals of the uncertainty estimate is narrow but do not coincide with true

values the right number of times. Uncertainties could also be imprecise but accurate, in which case the confidence interval is wide and it coincides well with true values. When uncertainties are both imprecise and inaccurate it means that despite wide confidence intervals, the intervals still miss the true values more frequently than it should. Ideally we want the uncertainty estimations to be both precise and accurate, such that a narrow enough confidence intervals coincide the true values the right number of times, this will mean the estimates are well calibrated. Here we will put forward the procedure of measuring the degree of calibration of different methods.

The calibration of uncertainty estimations from each method are obtained slightly differently, but the idea is the same. As a reminder we have performed uncertainty estimations using different methods and we have obtained:

- **Deep Ensemble**: 50 MAP estimations on NF parameters. This corresponds to sampling 50 NF distributions.
- **MC Dropout**: 100 NF distributions, sampled by having dropout turned on after modified MAP estimation.
- **MFVI**: 100 sets of NF parameters sampled from a trained variational distribution. This corresponds to sampling 100 NF distributions.

In each method we have sampled many NF distributions. For one sampled NF distribution, we evaluate its density on a 2D grid of points, one value at each point. For each method, there will be many sampled NF distributions, and we will evaluate the density values for all of them, across all of the grid points. Overall, each coordinate of the 2D grid will have many density values, each value corresponds to a NF distribution. At each coordinate point, the range of density values collectively represents an uncertainty estimate and we can define a confidence interval centered at the median by computing the quantiles of values at that point. Across the 2D grid, we would obtain many confidence intervals. Since we already have the true density values from the idealised distributions, we could count the proportion of true values lie within the confidence intervals. The result would reveal how calibrated a particular method is on uncertainty estimations.

Given N grid points with locations as vectors $\{\boldsymbol{x}_i\}_{i=1}^N$, with the corresponding true densities $\{y_i\}_{i=1}^N$, and confidence intervals denoted by $c_p(\boldsymbol{x}_i)$ as a function of the locations, with probability p enclosed inside the interval, the following describes the procedure for calibration calculations for each method.

1. Sample a pre-specified number of NF distributions.
2. Evaluate all the densities of the NF distributions across the grid of points.

3. Find quantiles per grid point to form confidence intervals.

4. Find proportion of true density points that lie within confidence intervals, according to:

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}[\boldsymbol{y}_i \in c_p(\boldsymbol{x}_i)] \tag{4.10}$$

   This quantity indicates the calibration of the model.

We will evaluate the calibrations on 10%, 50% and 90% confidence intervals. The above procedure will be repeated for all types of synthetic dataset distributions. The results of the calibrations will be summarised in tables for all methods.

Chapter 5

Results and Discussions

This chapter presents the main results obtained from experiments. The experiments were performed according to the model setup and procedures as described in the last chapter. The experiments were performed on the synthetic datasets using MAP estimation, deep ensemble, Monte-Carlo dropout and mean-field variational inference, with results obtained both in the large sample size limit and in the finite sample size regime. Interpretation, explanation and comparison of the results will be discussed.

5.1 Large Sample Size Limit

In the large sample size limit, results were obtained using datasets with sample size of 1 million. With this large sample size, we are simulating the effect of having virtually infinite data.

5.1.1 Maximum a Posteriori Estimation

MAP estimations were performed on NF parameters on synthetic datasets in the large sample size limit, and the results are shown in figure 5.1, with the top row showing the truth distributions that generated the synthetic datasets, and the bottom row showing the distributions that the NF learnt from MAP estimations on synthetic datasets.

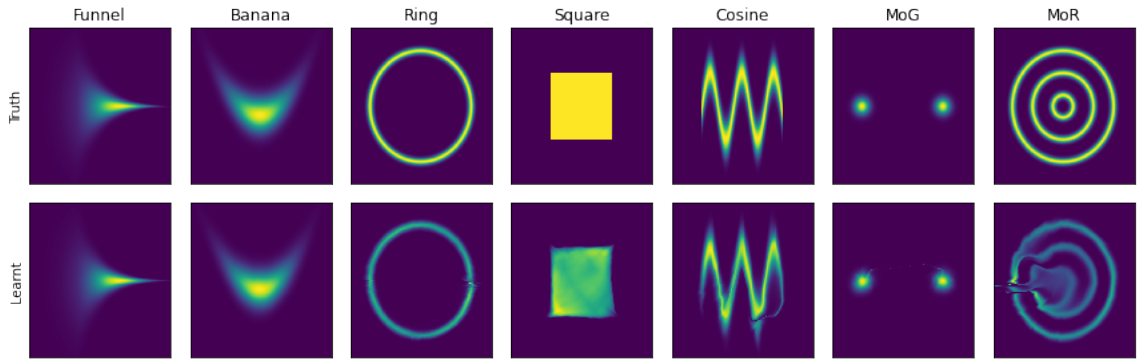


Figure 5.1: MAP estimations on synthetic datasets in the large sample size limit.

As seen from figure 5.1, qualitatively the NF could reproduce the original distributions reasonably well after being given virtually infinite data. It showed that the NF basically has the ability and flexibility to represent the original distributions. For the distribution of MoR, the NF appears to struggle a little with the rings being mixed together. Repeated experiments on the MoR distribution using NF with increased or reduced number of coupling layers do not solve the problem of mixing rings. This suggests that the way we construct and define a coupling layer could be causing this limitation to producing well separated rings.

From the posterior distributions of NF parameters perspective, we expect that the posterior would have a highly non-linear landscape with many peaks, or modes. A MAP estimate for a set of parameters would correspond to the parameters landing at a location of a single mode in the posterior. Since the posterior consist of many modes, the result obtained here would be just an instance of the MAP estimate. If we were to repeat the MAP estimation again we would get a slight different result, which correspond to a different MAP estimation for a set of NF parameters landing at a different mode. In the large sample size limit, we would expect that the landscape of the posterior distribution should be very concentrated, in which the bulk of the probability would be concentrated in a small region of the parameter space. This is due to the nature of posterior distribution of Bayesian inference in that a strong belief can be made after observing a lot of data. In the bulk there would consist of many narrow peaks each correspond to a mode. However each peak would be very close to each other. This explains that even with re-training for the MAP estimation, the final result obtained would be very similar to old results, with the new set of MAP parameters being very close to the old ones.

5.1.2 Deep Ensemble

The method of deep ensembles were performed for NF on synthetic datasets in the large sample size limit, with the ensemble size being 50, and the results are shown in figure 5.2. The top row in the figure shows the truth distributions. The middle row shows the means of ensembles, with one member in an ensemble represents an NF distribution obtained from a MAP estimation. The bottom row shows the standard deviation of ensembles.

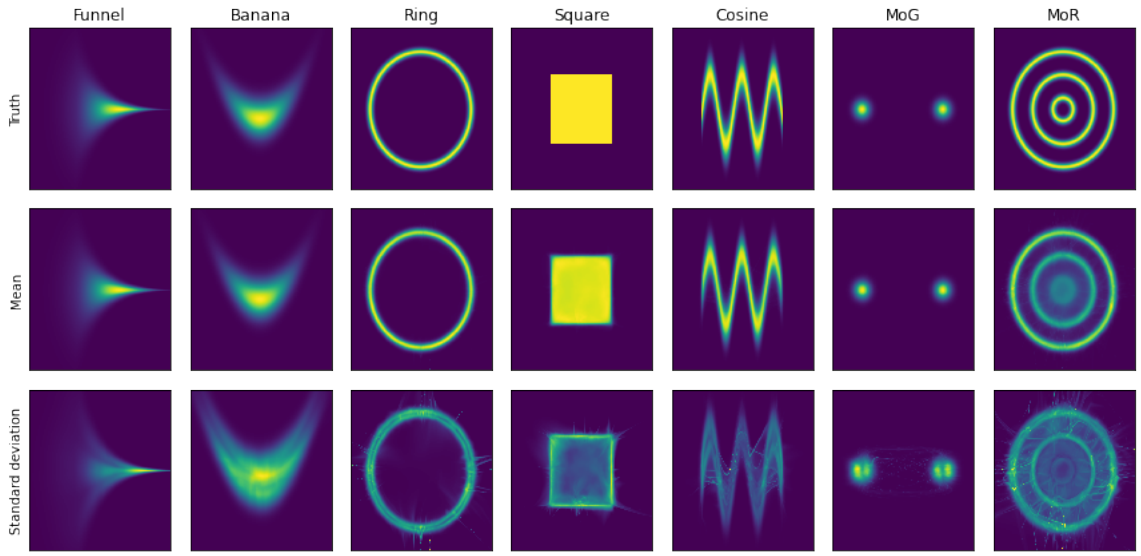


Figure 5.2: Deep ensemble predictive distributions and uncertainty estimations on synthetic datasets in the large sample size limit.

	10%	50%	90%
Funnel	0.46	0.87	0.93
Banana	0.27	0.67	0.95
Ring	0.13	0.58	0.71
Square	0.83	0.94	0.97
Cosine	0.50	0.92	0.97
MoG	0.13	0.52	0.99
MoR	0.14	0.55	0.87
Overall	0.35	0.72	0.91

Table 5.1: Deep ensemble calibration with confidence intervals trained on synthetic datasets in the large sample size limit.

As seen from figure 5.2, qualitatively the mean of the deep ensemble which make up the predictive distributions, could approximate the true distributions relatively well. For

the MoR distribution, despite one MAP estimation would get the rings being mixed up, in this deep ensemble method this mixed up effect is averaged out and the final approximation of the MoR distribution resembles the true distribution qualitatively well.

For the standard deviation of the deep ensemble, this shows pixel-wise that the uncertainty estimations for the predictive distribution. The brighter regions signify that the uncertainty is high in that region and that the confidence intervals are wide.

Notice that for the square, the model produced by deep ensemble is comparatively uncertain on the edges of the square, with less uncertainty in the middle region of the square. Intuitively this makes sense since the edge of the square is the only region where the distribution abruptly changes, which creates uncertainty when trying to learn the distribution. In the middle region the square is uniformly distributed with no abrupt changes, which we would expect the density over there is relatively easy to predict with very little uncertainty.

The same reasoning can be applied to the ring distribution, where we can see that our model predicts that the middle part of the ring has less uncertainty, and that the edges of the ring have more uncertainty. This makes sense since the density of the edges of the ring is changing relatively quickly compare to the inside of the ring. The same phenomenon can be observed in MoG as well.

For MoR, the model is in general relatively uncertain about the predictive distribution. This makes intuitive sense since as described previously that our NF structure has limitation on modelling the MoR distribution, this in turn causes uncertainty in the predictive distribution of MoR.

In the posterior distribution landscape point of view, each member in the ensemble corresponds to a location of a mode of the posterior in the parameter space. Since as mentioned before the bulk of the posterior should be concentrated around a location with high number of peaks, this will mean that the members in the ensemble will all have similar NF parameters. This explains why the uncertainty around the edge of the square is relatively narrow.

Table 5.1 shows the calibration of uncertainty estimations of the deep ensemble method trained on the synthetic datasets in the large sample size limit. The calibration is obtained from measuring the proportion of true density values that lie within the confidence intervals. The table shows the calibration for 10%, 50% and 90% confidence intervals. A perfectly calibrated uncertainty estimation should have the proportion of true values lying in the intervals equal to the probability of the confidence interval. Looking at the table the results show that the uncertainty estimates for the ensemble

method on the large sample size limit is under-confident overall, with more true density values lying within the confidence intervals than it should be. Out of all the distributions, MoR has the best calibration with their calibration values relatively close to the desired confidence intervals value. The least calibrated is for the square where the uncertainty estimations are obviously under-confident.

5.1.3 Monte Carlo Dropout

The method of MC dropout were performed for NF on synthetic datasets in the large sample size limit, with 100 sets of NF distributions drawn from dropout being turned on at test time. The results are shown in figure 5.3. The top row in the figure shows the truth distributions. The middle row shows the means of all the drawn NF distributions. The bottom row shows the standard deviation of all drawn NF distributions.

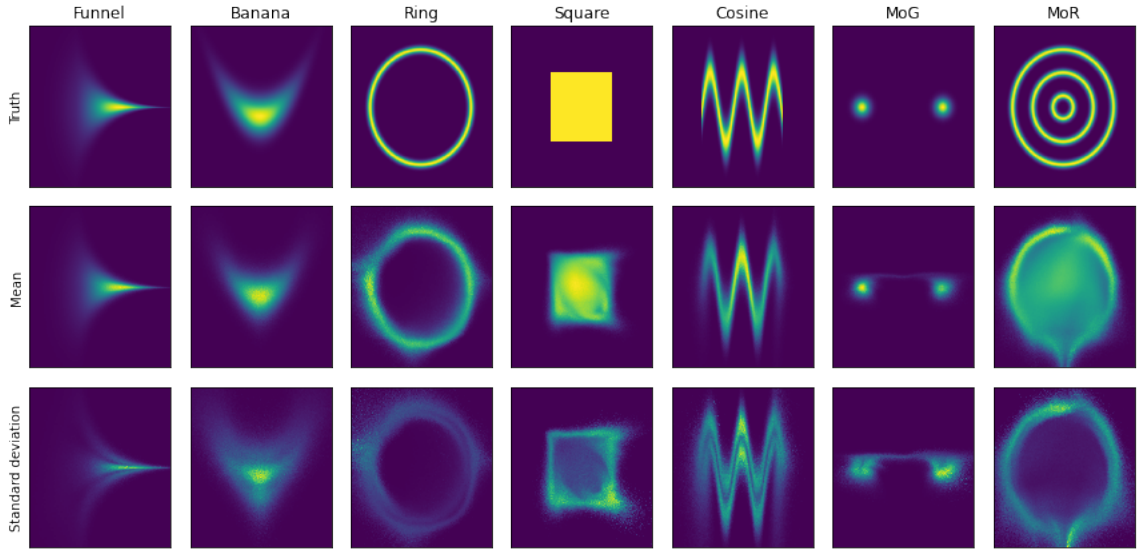


Figure 5.3: MC dropout predictive distributions and uncertainty estimations on synthetic datasets in the large sample size limit.

	10%	50%	90%
Funnel	0.40	0.89	0.98
Banana	0.07	0.53	0.90
Ring	0.04	0.23	0.45
Square	0.55	0.65	0.78
Cosine	0.12	0.32	0.58
MoG	0.06	0.34	0.83
MoR	0.04	0.23	0.48
Overall	0.18	0.46	0.71

Table 5.2: MC dropout calibration with confidence intervals trained on synthetic datasets in the large sample size limit.

As seen from figure 5.3, qualitatively the mean of the sampled NF distributions which make up the predictive distributions, approximates the truth distributions less well than the predictive distributions obtained from the deep ensemble method. However, visually the approximations are still good except for MoR. For MoR the predictive distribution failed to reproduce the inner rings of the true distribution.

In general the predictive distributions qualitatively have less resolution than the ones obtained from the deep ensemble method. One explanation is that due to dropout, every time we sample a NF distribution we are also sampling a subset of parameters from all available parameters in the NF. At training time dropout means that we are performing multiple MAP estimations on multiple subsets of parameters. Given many possible sets of NF parameters, each set can be viewed as corresponding to a unique posterior distribution. At training time we will be performing many MAP estimations on all possible posterior distributions. After training we would have obtained many possible MAP estimations on all possible unique posteriors, one MAP estimation for each of them. Since the subsets of NF parameters would be sharing a lot of parameters, the posterior distributions from the subsets would be correlated. With these correlated but different posteriors, we would obtain more diverse MAP estimations of the different posteriors than what we would have obtained from the deep ensemble method. With that there would be more diversity of sampled NF distributions in this MC dropout method, hence the overall predictive distribution for MC dropout would appear to be more vague with less resolution.

For MoR, it can be seen that the predictive distribution does not approximate the true distribution well, with the predictive distribution not able to approximate the inner rings. The reason could be that in dropout all the parameters collectively have to perform MAP estimations on many posterior distributions, which reduces the ability

for an individual subset of parameters to approximate the true distribution in its own MAP estimation. Since all the underlying posteriors are correlated from the sharing of parameters, each MAP estimation has to compromise for other MAP estimations to approximate the true distribution. In effect each MAP estimation in all MAP estimations would compromise for each other and the more complicated feature in a distribution would not be approximated, as in the case for MoR and as the NF already have difficulty to approximate the inner rings in MoR to begin with.

As for the uncertainty estimations represented by the standard deviations of all sampled NF distributions, qualitatively they retain most features of the uncertainty estimates from the deep ensemble method. Most notably the uncertainty around the edges of the square. As for MoR, the uncertainty is mostly at around the outer ring with less uncertainty around the inner rings. This can also be explained by the inability of the NF parameters to capture the inner rings complexity due to the multiple MAP estimations of subsets of NF parameters being compromised.

Table 5.2 shows the calibration of uncertainty estimations of the MC dropout method trained on the synthetic datasets in the large sample size limit. Overall, the model is under-confident at 10% , roughly calibrated at 50% and over-confident at 90% confidence intervals. This indicates that the actual distribution for the uncertainty is more spread than predicted overall.

5.1.4 Mean-Field Variational Inference

Training of MFVI in the large sample size limit for synthetic datasets was not successful and therefore there were no results of predictive distributions and uncertainty estimations regarding MFVI. However, conceptually MFVI attempts to approximate a local mode of the true posterior distribution using a Gaussian. In the large sample size limit, we expected that the posterior would be a multimodal distribution with its bulk concentrated around a small location, with the bulk consisted of many spiky peaks or modes. What MFVI does is that it attempts to approximate one of the very spiky mode with a Gaussian. This forces the variance of the Gaussian to be very small and contracts to a delta-function-liked distribution. Numerically this will cause the MFVI not to train properly, this might be the reason why the training was not successful.

5.2 Finite Sample Size

In this section we presents the results obtained using datasets with finite sample size. The sample size used is 500. With the much smaller sample size, it is expected that the predictive distribution will degrade in quality compared to the results in the large sample size limit, and the uncertainty estimates would be significantly larger.

5.2.1 Maximum a Posteriori Estimation

MAP estimations were performed on NF parameters on synthetic datasets with finite sample size, and the results are shown in figure 5.4, with the top row showing the truth distributions that generated the synthetic datasets, and the bottom row showing the distributions that the NF learnt from MAP estimations on synthetic datasets.

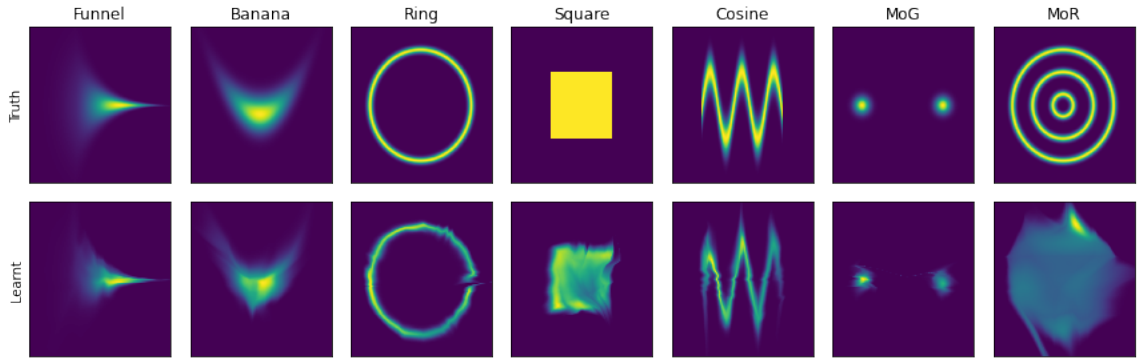


Figure 5.4: MAP estimations on synthetic datasets with finite sample size.

As seen from the results, it is expected that the learnt NF distributions of the MAP estimations in the finite data regime are qualitatively worse than what we have obtained in the large data limit regime in figure 5.1. Visually the learnt NF distributions still resemble the true distributions but with degraded quality. The learnt distribution for MoR is qualitatively the worst among other distributions, with the inner and other rings features not reproduced correctly.

In the true posterior distribution landscape point of view, since we are now in the finite data regime, the nature of Bayesian inference tells us that the true posterior would spread out more than the true posterior does in the large data size regime. The the main features of the posterior in the large data regime would remain in the current regime, with the surface of the posterior being highly non-linear and consists of many peaks or mode. With the finite data regime, the bulk of the posterior will now not be concentrated in the vicinity of a point in the parameter space and the locations of the modes would be more separated. Over many repeated MAP estimations, there

would be different results obtained, these results would correspond to the parameters converged to different modes in the true posterior.

5.2.2 Deep Ensemble

The method of deep ensembles were performed for NF on synthetic datasets in the finite data regime, with the ensemble size being 50, and the results are shown in figure 5.5. The top row in the figure shows the truth distributions. The middle row shows the means of ensembles, with one member in an ensemble represents an NF distribution obtained from a MAP estimation. The bottom row shows the standard deviation of ensembles.

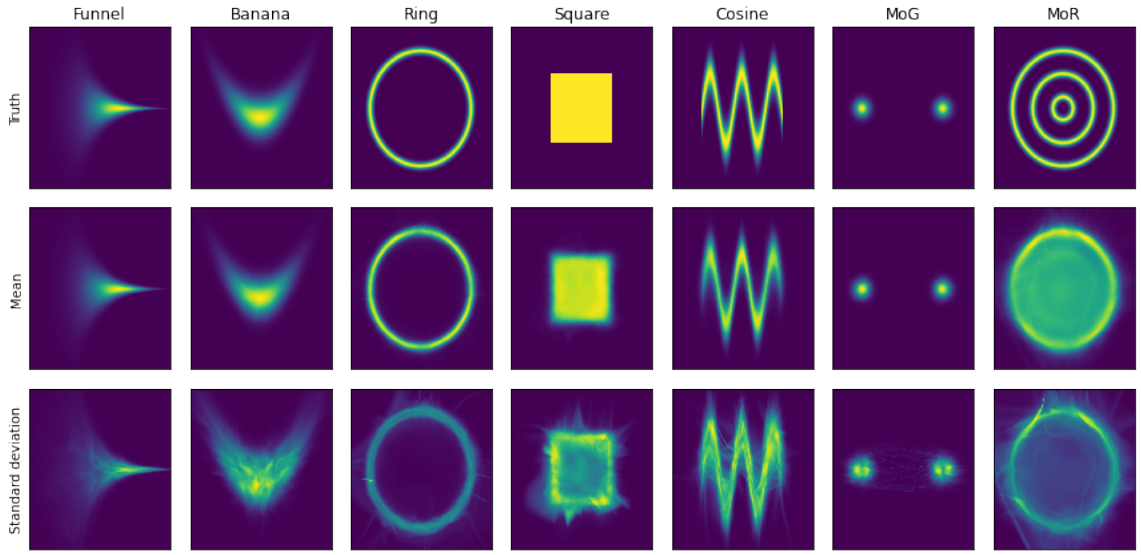


Figure 5.5: Deep ensemble predictive distributions and uncertainty estimations on synthetic datasets with finite sample size.

	10%	50%	90%
Funnel	0.25	0.65	0.93
Banana	0.13	0.54	0.91
Ring	0.10	0.43	0.72
Square	0.72	0.86	0.93
Cosine	0.43	0.84	0.96
MoG	0.08	0.34	0.86
MoR	0.05	0.32	0.63
Overall	0.25	0.57	0.85

Table 5.3: Deep ensemble calibration with confidence intervals trained on synthetic datasets with finite sample size.

As seen in the results in figure 5.5, the predictive distributions in the middle row qualitatively approximate the true distributions well, with degraded quality compared to the approximation obtained in the large sample size regime as shown in figure 5.2, as expected. MoR has noticeably worse predictive distribution with less defined rings compared to the MoR in the large sample size regime, and the square has less sharp edges.

Looking at the uncertainty estimations of the last row in the results, qualitatively the uncertainties across distributions look similar to the ones obtained in the large sample size regime. However, as we will compare and discuss in later section in table 5.7, on average the uncertainty estimates are higher compared to the results obtained in the large sample size regime, also as expected. Comparing to the large sample size regime, the uncertainty of the square has a thicker region of uncertainty around the edge, this makes sense since with less data we would expect the predictive distribution would be more uncertain in the region of rapid changing density. For MoR, the uncertainty estimate qualitatively look different than the one obtained in the large sample size regime, with most uncertainty situated at around the outer ring, and with less uncertainty around the inner rings. As we can see this uncertainty estimate is inaccurate since the predictive distribution of MoR clearly misses the true distribution.

In the perspective of the landscape of the true posterior distribution for this case of smaller sample size, it is expected that the posterior would be more spread across the NF parameter space than the case in the large sample size limit. With the members in the deep ensemble obtained by multiple MAP estimations on the true posterior, the overall ensemble would consist of resulting parameters that situated at multiple locations in the posterior, with each location being a mode of the posterior. The locations of the modes will be more separated than what would have been obtained in the large data size limit. As a result, the ensemble would consist of NF distribution members that are relatively different from each other. This explains why the predictive distributions are of inferior quality and the uncertainty is larger on the predictive distribution.

The result of calibration is shown in table 5.3. Overall the calibration of this method is relatively good for the 50% and 90% confidence intervals. This method is under-confident when looking at the 10% confidence interval. As seen from the table, the Banana distribution is comparably more well-calibrated than others.

5.2.3 Monte Carlo Dropout

The method of MC dropout were performed for NF on synthetic datasets in the finite data regime, with 100 sets of NF distributions drawn from dropout being turned on at test time. The results are shown in figure 5.6. The top row in the figure shows the truth distributions. The middle row shows the means of all the drawn NF distributions. The bottom row shows the standard deviation of all drawn NF distributions.

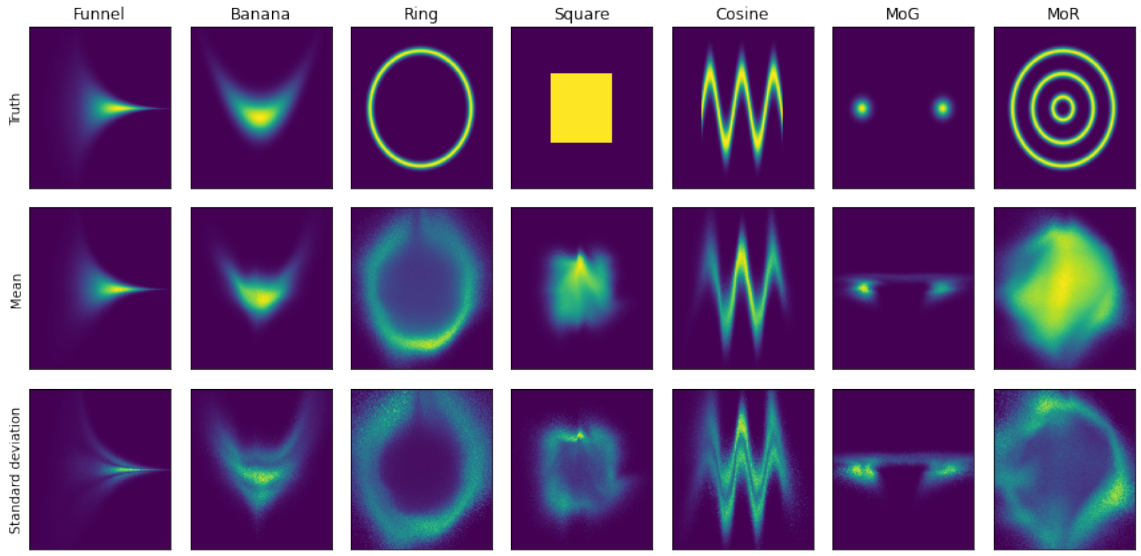


Figure 5.6: MC dropout predictive distributions and uncertainty estimations on synthetic datasets with finite sample size.

	10%	50%	90%
Funnel	0.17	0.35	0.67
Banana	0.07	0.37	0.75
Ring	0.04	0.22	0.49
Square	0.21	0.33	0.51
Cosine	0.11	0.29	0.53
MoG	0.03	0.15	0.42
MoR	0.03	0.17	0.36
Overall	0.09	0.27	0.53

Table 5.4: MC dropout calibration with confidence intervals trained on synthetic datasets with finite sample size.

As can be seen from the figure, although the predictive distributions as shown in the second row still resemble the true distributions, the quality of them has degraded significantly compared to the predictive distributions from all the previous methods. The

ring retains the ring shape but it is clear that it does not approximate the original ring distribution well. Square and MoG are starting to diverge away from approximating the true distributions, however they still managed to retain the shape qualitatively. MoR approximation does not resemble the original distribution, with the features of outer and inner rings being disappeared. Compared to the results obtained in the large sample size regime as in figure 5.3, we can see that the predictive distributions here are less accurate qualitatively, as expected.

For the uncertainty estimations on the third row of the figure. Qualitatively the estimates look similar to the ones obtained in the large sample size regime in figure 5.3, with all the main features remained the same but with larger uncertainty regions. In terms of the magnitude of the pixel-wise average uncertainty estimates, the values in the large data regime are surprisingly similar what we have here. We would have expected that the magnitude of uncertainty in our current data size regime to be larger since we have less data available. The comparisons are shown in table 5.7.

Previously we have described a theory in which during the MAP estimation phase in MC dropout, every subset of NF parameters can be treated as having its own unique posterior distribution, and we can treat the training phase of MC dropout as multiple MAP estimations on multiple posteriors at the same time. Since the subsets of the parameters would share parameters with each other, the resulting MAP estimations on the possible subsets would be correlated, with each subset of parameters correspond to a NF distribution. Now, since there is a great degree of parameters sharing among all the NF distributions, it means that the resulting NF distributions would be correlated and would not differ from each other by much. Since the degree of parameters sharing is independent of the size of training data, the degree of variation of the sampled NF distributions using dropout would approximately not be affected by the training data size. Hence the variance of many sampled distributions would not be very different when comparing the variances obtained in the finite and large data regimes, explaining the observations in the similarity of the magnitude of uncertainties.

The results of calibration is shown in table 5.4. The results showed that MC dropout in the finite data regime is relatively quite over-confident, especially in the distributions of ring, MoG and MoR. As already discussed, the method of dropout would restrict the variance of a group of sampled NF distributions due to the sharing of NF parameters between subset parameters, meaning that the pixel-wise confidence intervals will be limited in width, therefore the model will likely to be over-confident.

5.2.4 Mean-Field Variational Inference

The method of MFVI were performed for NF on synthetic datasets in the finite data regime, with 100 sets of NF distributions drawn from a variational distribution that approximates the posterior distribution. The results are shown in figure 5.7. The top row in the figure shows the truth distributions. The middle row shows the means of all the drawn NF distributions. The bottom row shows the standard deviation of all drawn NF distributions.

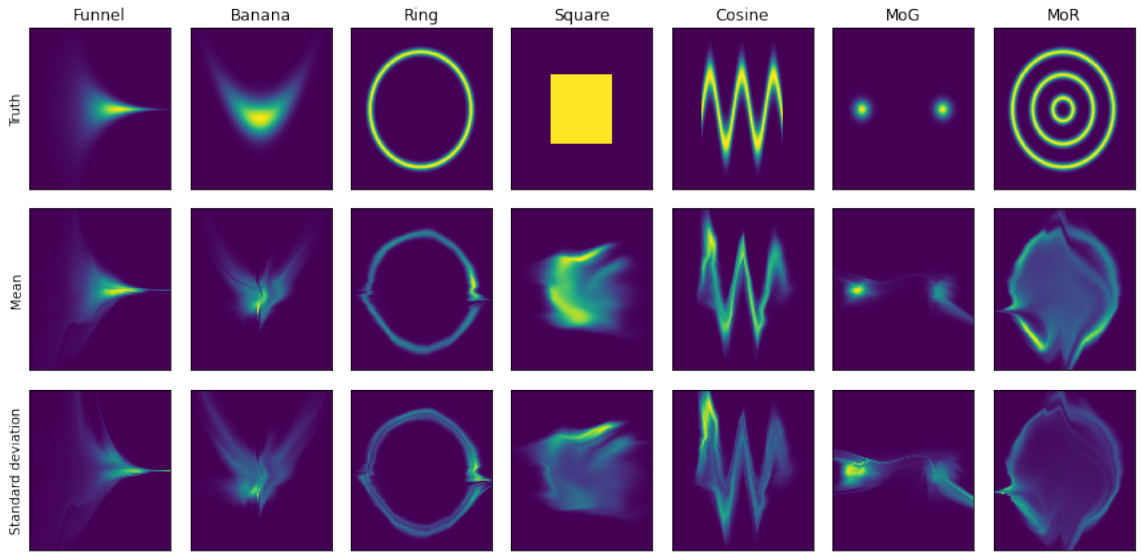


Figure 5.7: MFVI predictive distributions and uncertainty estimations on synthetic datasets with finite sample size.

	10%	50%	90%
Funnel	0.16	0.24	0.46
Banana	0.01	0.07	0.23
Ring	0.02	0.09	0.25
Square	0.47	0.50	0.56
Cosine	0.22	0.32	0.49
MoG	0.02	0.11	0.33
MoR	0.02	0.08	0.21
Overall	0.13	0.20	0.36

Table 5.5: MFVI calibration with confidence intervals trained on synthetic datasets with finite sample size.

As can be seen from the results in figure 5.7, although the predictive distributions shown in the second row do overall qualitatively resemble the true distributions, the

quality of estimations are the worst when compared to all other previous methods. The quality of predictive distributions for the square and MoR are clearly the worst, with the edges of the square not matching with the original distribution, and with the inner rings of MoR not properly approximated.

The uncertainty estimations are shown in the third row of the figure as standard deviations. As with the predictive distributions, the uncertainty estimations are relatively erratic overall. The magnitude of average pixel-wise uncertainty is shown in table 5.7.

In the true posterior distribution landscape perspective, what MFVI does is to approximate one of the modes of the posterior with a Gaussian. Since only one mode would be approximated, the resulting Gaussian approximation does not represent an accurate picture of the complete true posterior. As a result, the final Gaussian approximation will get stuck in a mode of the true posterior and any NF parameters that sampled from the Gaussian would be located at around the location of the mode. With those sampled parameters, the associated NF distributions will also be relatively similar to each other with little diversity. With this concept in mind, we can explain why the predictive distributions qualitatively do not approximate the true distributions well, this is because all sampled NF parameters are situated at around a single mode of the true posterior, and that the particular mode is likely to be far from an optimal one, making the predictive distributions approximate the true distributions inaccurately.

Table 5.5 shows the results of calibration. Overall it can be seen that the MFVI method is badly calibrated compared to the calibrations obtained from other methods. This method is heavily over-confident on all confidence intervals of 10%, 50% and 90%. In the posterior landscape point of view, this result makes sense, as the NF distributions were only sampled by sampling the associated NF parameters in the vicinity of a local mode of the true posterior, which leads to very little diversity in the NF distributions, making the confidence intervals being narrower than it should be. This leads to the uncertainty estimations obtained being overly confident.

5.3 Comparisons

This section compares the results obtained between the methods of deep ensemble, MC dropout and MFVI, both in the large and finite sample size regimes. Table 5.6 summarises the overall calibration of the uncertainty estimates for various methods. Table 5.7 shows the pixel-wise average of the standard deviations of sampled NF distributions, with the mean of the sampled NF distributions corresponds to a predictive

distribution. This measure quantifies overall how uncertain the method on its predictive distribution is.

Calibration		10%	50%	90%
Large	DE	0.35	0.72	0.91
	MCD	0.18	0.46	0.71
Finite	DE	0.25	0.47	0.85
	MCD	0.09	0.27	0.53
	MFVI	0.13	0.20	0.36

Table 5.6: Calibrations for different methods in different data size regimes, measured in different confidence intervals.

Std. Dev.	Large		Finite		
$\times 10^3$	DE	MCD	DE	MCD	MFVI
Funnel	0.4	1.0	1.5	0.7	0.9
Banana	0.4	1.9	1.5	1.2	0.7
Ring	1.7	3.9	3.7	3.0	1.3
Square	0.8	1.1	1.9	1.2	1.0
Cosine	1.1	2.0	2.7	2.4	1.3
MoG	0.9	3.1	2.3	3.0	0.8
MoR	3.0	1.5	2.6	1.2	1.4
Overall	1.2	2.1	2.3	1.8	1.1

Table 5.7: Uncertainty estimations for different methods in different data size regimes, measured in pixel-wise average standard deviations.

In the predictive distributions point of view, it is clear from the results in the figures presented earlier that the method of deep ensemble produces the best predictive distributions, in both data size regimes. The results also showed that in general, predictive distributions are better in the large data regime than in the small data regime, regardless of methods. This makes intuitive sense since large sample size means a concentrated true posterior in NF parameter space, leading to a higher change of sampling a set of NF parameters that are closer to the optimal NF parameters in the NF parameter space and hence a better predictive distribution.

In the uncertainty estimations point of view, table 5.7 summarises the estimates across different methods. The estimations are quantified using standard deviations. There are three observations.

First, the size of uncertainty estimates of deep ensemble are almost always smaller in the large sample regime. This can be seen in the two DE columns across different distribution shapes. This makes intuitive sense since thinking about the true posterior landscapes, the large data regime has a more narrow posterior distribution as oppose to a wider posterior for the smaller data regime. This means that the locations of the modes in the posterior would be closer together in the large data regime. As deep ensemble samples NF parameters at the modes of the posterior, the sampled parameters in the large data regime would be closer together than what we would have obtained in the smaller data regime. This leads to less variance in the sampled NF distributions and hence smaller uncertainty estimations.

Second, the size of uncertainty estimates of MC dropout do not change significantly for different data size regimes. This can be seen in the two columns of MCD across different distribution shapes. As discussed before this phenomenon could be attributed to the the fact that the degree of sharing of NF parameters in dropout between sampled NF distributions is independent of data size, therefore it makes sense that the uncertainty estimates for MC dropout remain similar in different data size regimes.

Third, the sizes of uncertainty estimates of MFVI are mostly the smallest across all distributions shapes when comparing with other methods. This makes sense if we think about the landscape of the true posterior again. As discussed previously, what MFVI does is to approximate a single mode of the true posterior with a Gaussian, however approximating a single mode does not give much diversity when sampling NF parameters from the Gaussian since the locations of the sampled NF parameters would only be at around the location of the single mode. With deep ensemble and MC dropout, there are able to sample NF parameters with more diversity. In effect the sampled NF distributions from the Gaussian of MFVI would not differ by much and hence giving a relatively low uncertainty estimations when comparing with the other methods.

In the calibration point of view, table 5.6 summarises the results for different methods for comparisons. There are two observations.

First, MFVI is the most over-confident method overall, especially for the 50% and 90% confidence intervals. This is again can be explain by the hypothesis that the Gaussian approximation to a mode in the true posterior produces too little variance in the sampled NF distributions. This leads to the confidence intervals being too restrictive compared to the intervals found by other methods. Therefore MFVI will very easily be over-confident on uncertainty estimations.

Second, MC dropout moves towards being over-confident from large to finite data

regimes. As discussed before, this could be explained by the hypothesis that the uncertainty estimates of MC dropout does not depend on data size, due to the degree of sharing of NF parameters between sampled NF distributions being unchanged with different data size. Since less data leads to higher uncertainty, with the MC dropout uncertainty estimates in terms of the confidence intervals being unchanged, it makes sense that the proportion of true density values lying in the confidence intervals will decrease from the large to finite data regime, therefore MC dropout would become more over-confident as data size decreases.

Chapter 6

Conclusion and Future Work

In this report, we have performed Bayesian inference on normalising flow parameters. On a high level, we have used the methods of deep ensemble, Monte Carlo dropout and mean-field variational inference to investigate uncertainties within the parameters of normalising flow models in different dataset size domains. We demonstrated that the behaviours of the uncertainty estimations varies across methods and dataset size domains. Furthermore, through the notion of Bayesian posterior predictive distributions on the normalising flow models, we also used the methods to perform density estimations. We demonstrated that the quality of the estimations also vary across methods and dataset size domains.

Results from experiments showed that the deep ensemble method is the best among other methods in uncertainty estimations in terms of calibration, and qualitatively it is also the best method to perform density estimation. We explained and interpreted the results from the point of view of the landscape of posterior distributions, and concluded that the performance of the deep ensemble method relies on its ability to explore multiple modes, gaining a broader picture within the landscape of the posterior distributions, which leads to more accurate uncertainty and density estimations. We also concluded that the inferior performances of other methods can be attributed to their limited ability in exploring diverse regions within posterior distributions, leading to inferior uncertainty and density estimations. Results from experiments also showed that in general, uncertainties in parameters are smaller and density estimations are more accurate in the larger dataset size domain. We explained the observations also in the point of view of the landscape of posterior distributions, where the contraction of the width of the posterior distributions caused by an increased dataset size leads to better uncertainty and density estimations.

In this report, we have only concerned the uncertainties within the normalising flow

parameters, however we have not considered what would happen if instead of a deterministic transformation of the normalising flow with deterministic parameters, we put the parameters of the normalising flow to be probabilistic, with the probabilistic parameters being distributed according to a posterior distribution, and allow the new normalising flow to be a probabilistic transformation. Future research could investigate in this direction. In specific, one direction could investigate whether the deterministic function obtained by taking the expectation of a normalising flow with its parameters distributed according to a posterior, has an inverse relation with another deterministic function, where the other deterministic function is obtained by taking an expectation of the inverse of the same normalising flow, with its parameters also distributed according to the same posterior. The other direction could investigate the question of how the posterior predictive distribution of a point in the latent space of a normalising flow looks like, given a fixed data point in the visible space of the flow, with the flow parameters being distributed according to a posterior.

Bibliography

[1] D. Barber and Christopher Bishop. "Ensemble learning in Bayesian neural networks". In: *Generalization in Neural Networks and Machine Learning*. Springer Verlag, Jan. 1998, pp. 215–237. URL: https://www.microsoft.com/en-us/research/publication/ensemble-learning-in-bayesian-neural-networks/.

[2] Matthew James Beal. *Variational algorithms for approximate Bayesian inference*. University of London, University College London (United Kingdom), 2003.

[3] Charles Blundell et al. "Weight uncertainty in neural network". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1613–1622.

[4] Scott Chen and Ramesh Gopinath. "Gaussianization". In: *Advances in Neural Information Processing Systems*. Ed. by T. Leen, T. Dietterich, and V. Tresp. Vol. 13. MIT Press, 2001. URL: https://proceedings.neurips.cc/paper/2000/file/3c947bc2f7ff007b86a9428b74654de5-Paper.pdf.

[5] Andreas Damianou and Neil D Lawrence. "Deep gaussian processes". In: *Artificial intelligence and statistics*. PMLR. 2013, pp. 207–215.

[6] A. Dawid. "The Well-Calibrated Bayesian". In: *Journal of the American Statistical Association* 77 (1982), pp. 605–610.

[7] John Denker and Yann Lecun. "Transforming Neural-Net Output Levels to Probability Distributions". In: *Advances in Neural Information Processing Systems 3*. Morgan Kaufmann, 1991, pp. 853–859.

[8] Laurent Dinh, David Krueger, and Yoshua Bengio. "NICE: Non-linear Independent Components Estimation". In: *CoRR* abs/1410.8516 (2015).

[9] Laurent Dinh, J. Sohl-Dickstein, and Samy Bengio. "Density estimation using Real NVP". In: *ArXiv* abs/1605.08803 (2017).

[10] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. *Deep Ensembles: A Loss Landscape Perspective*. 2020. arXiv: 1912.02757 [stat.ML].

[11] Yarin Gal and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning". In: *international conference on machine learning*. PMLR. 2016, pp. 1050–1059.

[12] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[13] Alex Graves. "Practical Variational Inference for Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor et al. Vol. 24. Curran Associates, Inc., 2011. URL: https://proceedings.neurips.cc/paper/2011/file/7eb3c8be3d411e8ebfab08eba5f49632-Paper.pdf.

[14] Geoffrey E. Hinton and Drew van Camp. "Keeping the Neural Networks Simple by Minimizing the Description Length of the Weights". In: *Proceedings of the Sixth Annual Conference on Computational Learning Theory*. COLT '93. Santa Cruz, California, USA: Association for Computing Machinery, 1993, pp. 5–13. ISBN: 0897916115. DOI: 10.1145/168304.168306. URL: https://doi.org/10.1145/168304.168306.

[15] R. M. Johnson. "The minimal transformation to orthonormality". In: *Psychometrika* 31 (1966), pp. 61–66.

[16] Sungwon Kim et al. *FloWaveNet : A Generative Flow for Raw Audio*. 2019. arXiv: 1811.02155 [cs.SD].

[17] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2014. arXiv: 1312.6114 [stat.ML].

[18] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[19] Diederik P. Kingma and Prafulla Dhariwal. *Glow: Generative Flow with Invertible 1x1 Convolutions*. 2018. arXiv: 1807.03039 [stat.ML].

[20] Ivan Kobyzev, Simon Prince, and Marcus Brubaker. "Normalizing Flows: An Introduction and Review of Current Methods". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), pp. 1–1. ISSN: 1939-3539. DOI: 10.1109/tpami.2020.2992934. URL: http://dx.doi.org/10.1109/TPAMI.2020.2992934.

[21] Manoj Kumar et al. *VideoFlow: A Conditional Flow-Based Model for Stochastic Video Generation*. 2020. arXiv: 1903.01434 [cs.CV].

[22] Balaji Lakshminarayanan, A. Pritzel, and C. Blundell. "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles". In: *NIPS*. 2017.

[23] David J. C. MacKay. "A Practical Bayesian Framework for Backpropagation Networks". In: *Neural Comput.* 4.3 (May 1992), pp. 448–472. ISSN: 0899-7667. DOI: 10.1162/neco.1992.4.3.448. URL: https://doi.org/10.1162/neco.1992.4.3.448.

[24] David JC MacKay. "Probable networks and plausible predictions-a review of practical Bayesian methods for supervised neural networks". In: *Network: computation in neural systems* 6.3 (1995), p. 469.

[25] Bogdan Mazoure et al. *Leveraging exploration in off-policy algorithms via normalizing flows*. 2019. arXiv: 1905.06893 [cs.LG].

[26] Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *ICML*. 2010.

[27] Radford M Neal. *Probabilistic inference using Markov chain Monte Carlo methods*. 1993.

[28] Frank Noé et al. *Boltzmann Generators – Sampling Equilibrium States of Many-Body Systems with Deep Learning*. 2019. arXiv: 1812.01729 [stat.ML].

[29] Rajesh Ranganath, Sean Gerrish, and David Blei. "Black Box Variational Inference". In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*. Ed. by Samuel Kaski and Jukka Corander. Vol. 33. Proceedings of Machine Learning Research. Reykjavik, Iceland: PMLR, 22–25 Apr 2014, pp. 814–822. URL: https://proceedings.mlr.press/v33/ranganath14.html.

[30] Danilo Rezende and Shakir Mohamed. "Variational Inference with Normalizing Flows". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 1530–1538.

[31] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[32] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[33] E. Tabak and Turner Cristina. "A Family of Nonparametric Density Estimation Algorithms". In: *Communications on Pure and Applied Mathematics* 66 (Feb. 2013). DOI: 10.1002/cpa.21423.

[34] Esteban Tabak and Eric Vanden-Eijnden. "Density estimation by dual ascent of the log-likelihood". In: *Communications in Mathematical Sciences - COMMUN MATH SCI* 8 (Mar. 2010). DOI: 10.4310/CMS.2010.v8.n1.a11.

[35] Patrick Nadeem Ward, Ariella Smofsky, and Avishek Joey Bose. *Improving Exploration in Soft-Actor-Critic with Normalizing Flows Policies*. 2019. arXiv: 1906.02771 [cs.LG].

[36] Li Wenliang et al. "Learning deep kernels for exponential family densities". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6737–6746.

[37] Peter Wirnsberger et al. "Targeted free energy estimation via learned mappings". In: *The Journal of Chemical Physics* 153.14 (Oct. 2020), p. 144112. ISSN: 1089-

7690. DOI: 10.1063/5.0018903. URL: http://dx.doi.org/10.1063/5.0018903.