Bayesian Neural Networks: An Introduction and Survey

Ethan Goan, Clinton Fookes

Abstract Neural Networks (NNs) have provided state-of-the-art results for many challenging machine learning tasks such as detection, regression and classification across the domains of computer vision, speech recognition and natural language processing. Despite their success, they are often implemented in a frequentist scheme, meaning they are unable to reason about uncertainty in their predictions. This article introduces Bayesian Neural Networks (BNNs) and the seminal research regarding their implementation. Different approximate inference methods are compared, and used to highlight where future research can improve on current methods.

1 Introduction

Biomimicry has long served as a basis for technological developments. Scientists and engineers have repeatedly used knowledge of the physical world to emulate nature's elegant solutions to complex problems which have evolved over billions of years. An important example of biomimicry in statistics and machine learning has been the development of the perceptron [1], which proposes a mathematical model based on the physiology of a neuron. The ma-

Ethan Goan
Queensland University of Technology e-mail: ej.goan@qut.edu.au

Case Studies in Applied Data Science, (Eds K Mengersen, P Pudlo and CP Robert), Lecture Notes in Mathematics, Springer 2020

chine learning community has used this concept[1] to develop statistical models of highly interconnected arrays of neurons to create Neural Networks (NNs).

Though the concept of NNs has been known for many decades, it is only recently that applications of these network have seen such prominence. The lull in research and development for NNs was largely due to three key factors: lack of sufficient algorithms to train these networks, the large amount of data required to train complex networks and the large amount of computing resources required during the training process. In 1986, [3] introduced the backpropagation algorithm to address the problem of efficient training for these networks. Though an efficient means of training was available, considerable compute resources was still required for the ever increasing size of new networks. This problem was addressed in [4, 5, 6] where it was shown that general purpose GPUs could be used to efficiently perform many of the operations required for training. As digital hardware continued to advance, the number of sensors able to capture and store real world data increased. With efficient training methods, improved computational resources and large data sets, training of complex NNs became truly feasible.

In the vast majority of cases, NNs are used within a frequentist perspective; using available data, a user defines a network architecture and cost function, which is then optimised to allow us to gain point estimate predictions. Problems arise from this interpretation of NNs. Increasing the number of parameters (often called weights in machine learning literature), or the depth of the model increases the capacity of the network, allowing it to represent functions with greater non-linearities. This increase in capacity allows for more complex tasks to be addressed with NNs, though when frequentist methodologies are applied, leaves them highly prone to overfitting to the training data. The use of large data sets and regularisation methods such as finding a MAP estimate can limit the complexity of functions learnt by the networks and aid in avoiding overfitting.

Neural Networks have provided state-of-the-art results for numerous machine learning and Artificial intelligence (AI) applications, such as image classification [6, 7, 8], object detection [9, 10, 11] and speech recognition [12, 13, 14, 15]. Other networks such as the AlphaGo model developed by DeepMind [16] have emphasised the potential of NNs for developing AI systems, garnering a wide audience interested in the development of these networks. As the performance of NNs has continued to increase, the interest in their development and adoption by certain industries becomes more prominent. NNs are currently used in manufacturing [17], asset management [18] and human interaction technologies [19, 20].

Since the deployment of NNs in industry, there have been a number of incidents where failings in these systems has led to models acting unethically and unsafely. This includes models demonstrating considerable gender and

[1] While also relaxing many of the constraints imposed by a physical model of a natural neuron [2]. It should be emphasised that there is very little evidence to suggest that that arrangement of neurons seen in NNs are an accurate model of any physiological brain.

racial bias against marginalised groups[21, 22, 23] or to more extreme cases resulting in loss of life[24, 25]. NNs are a statistical black-box models, meaning that the decision process is not based on a well-defined and intuitive protocol. Instead decisions are made in an uninterpretable manner, with hopes that the reasonable decisions will be made based on previous evidence provided in training data[2]. As such, the implementation of these systems in social and safety critical environments raises considerable ethical concerns. The European Union released a new regulation[3] which effectively states that users have a "right to an explanation" regarding decisions made by AI systems [26, 27]. Without clear understanding of their operation or principled methods for their design, experts from other domains remain apprehensive about the adoption of current technology [28, 29, 30]. These limitations have motivated research efforts into the field of Explainable AI [31].

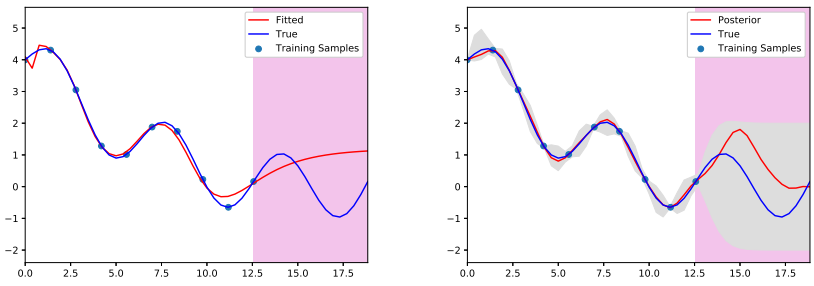
Adequate engineering of NNs requires a sound understanding of their capabilities and limitations; to identify their shortcomings prior to deployment as apposed to the current practice of investigating these limitations in the wake of these tragedies. With NNs being a statistical black-box, interpretation and explanation of the decision making process eludes current theory. This lack of interpretation and over-confident estimates provided by the frequentist perspective of common NNs makes them unsuitable for high risk domains such as medical diagnostics and autonomous vehicles. Bayesian statistics offers natural way to reason about uncertainty in predictions, and can provide insight into how these decisions are made.

Figure 1 compares Bayesian methods for performing regression with that of a simple neural network, and illustrates the importance of measuring uncertainty. While both methods perform well within the bounds of the training data, where extrapolation is required, the probabilistic method provides a full distribution of the function output as opposed to the point estimates provided by the NN. The distribution over outputs provided by probabilistic methods allows for the development of trustworthy models, in that they can identify uncertainty in a prediction. Given that NNs are the most promising model for generating AI systems, it is important that we can similarly trust their predictions.

A Bayesian perspective allows us to address many of the challenges currently faced within NNs. To do this, a distribution is placed over the network parameters, and the resulting network is then termed a Bayesian Neural Network (BNN). The goal of a BNN is to have a model of high capacity that exhibits the important theoretical benefits of Bayesian analysis. Recent research has investigated how Bayesian approximations can be applied to NNs

[2] Due to this black-box nature, the performance of these models is justified entirely through empirical means.

[3] This regulation came into effect on the 25th of May, 2018 across the EU [26].



(a) (b)

Fig. 1: Comparison of neural network to traditional probabilistic methods for a regression task, with no training data in the purple region. (a) Regression output using a neural network with 2 hidden layers; (b) Regression using a Gaussian Process framework, with grey bar representing ± 2 std. from expected value.

in practice. The challenge with these methods is deploying models that provide accurate predictions within reasonable computation constraints[4].

This document aims to provide an accessible introduction to BNNs, accompanied by a survey of seminal works in the field and experiments to motivate discussion into the capabilities and limits of current methods. A survey of all research items across the Bayesian and machine learning literature related to BNNs could fill multiple text books. As a result, items included in this survey only intend to inform the reader on the overarching narrative that has motivated their research. Similarly, derivations of many of they key results have been omitted, with the final result being listed accompanied by reference to the original source. Readers inspired by this exciting research area are encouraged to consult prior surveys: [32] which surveys the early developments in BNNs, [33] which discusses the specifics of a full Bayesian treatment for NNs, and [34] which surveys applications of approximate Bayesian inference to modern network architectures.

This document should be suitable for all in the statistics field, though the primary audience of interest are those more familiar with machine learning concepts. Despite seminal references for new machine learning scholars almost equivalently being Bayesian texts [2, 35], in practice there has been a divergence between much of the modern machine learning and Bayesian statistics research. It is hoped that this survey will help highlight similarities between

[4] The term "reasonable" largely depends on the context. Many neural networks are currently trained using some of the largest computing facilities available, containing thousands of GPU devices.

some modern research in BNNs and statistics, to emphasis the importance of a probabilistic perspective within machine learning and to encourage future collaboration/unison between the machine learning and statistics fields.

2 Literature Survey

2.1 Neural Networks

Before discussing a Bayesian perspective of NNs, it is important to briefly survey the fundamentals of neural computation and to define the notation to be used throughout the chapter. This survey will focus on the primary network structure of interest, the Multi-Layer Perceptron (MLP) network. The MLP serves as the basis for NNs, with modern architectures such as convolutional networks having an equivalent MLP representation. Figure 2 illustrates a simple MLP with a single hidden layer suitable for regression or classification. For this network with an input $\mathbf{x}$ of dimension N_1, the output of the $\mathbf{f}$ network can be modelled as,

$$\phi_j = \sum_{i=1}^{N_1} a(x_i w_{ij}^1), \tag{1}$$

$$f_k = \sum_{j=1}^{N_2} g(\phi_j w_{jk}^2). \tag{2}$$

The parameters w represent the weighted connection between neurons from subsequent layers, and the superscripts denoting the layer number. Equation 1 represents the output of the hidden layer, which will be of dimension N_2. The k^{th} output of the network is then a summation over the N_2 outputs from the prior hidden layer. This modelling scheme can be expanded to include many hidden layers, with the input of each layer being the output of the layer immediately prior. A bias value is often added during each layer, though is omitted throughout this chapter in favour of simplicity.

Equation 1 refers to the state of each neuron (or node) in the hidden layer. This is expressed as an affine transform followed by a non-linear element wise transform $\phi(\cdot)$, which is often called an activation. For the original perceptron, activation function used was the sign$(\cdot)$ function, though the use of this function has ceased due to it's derivative being equal to zero[5]. More favourable activation functions such as the Sigmoid, Hyperbolic Tangent (TanH), Rectified Linear Unit (ReLU) and Leaky-ReLU have since replaced this the sign function [36, 37]. Figure 3 illustrates these functions along with their cor-

[5] When the derivative is defined, as is a piece-wise non-differentiable function at the origin.

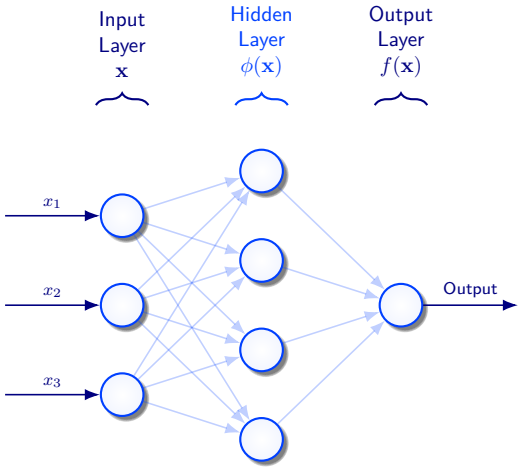


Fig. 2: Example of a NN architecture with a single hidden layer for either binary classification or 1-D regression. Each node represents a neuron or a state where the summation and activation of input states is performed. Arrows are the parameters (weights) indicating the strength of connection between neurons.

responding derivatives. When using the Sigmoid function, expression 1 is equivalent to logistic regression, meaning that the output of the network becomes the sum of multiple logistic regression models.

For a regression model, the function applied to the output $g(\cdot)$ will be the identity function[6], and for binary classification will be a Sigmoid.

Equations 1 and 2 can be efficiently implemented using matrix representations, and is often represented as such in machine learning literature. This is achieved by stacking the input vector in our data set as a column in $\mathbf{X}$. Forward propagation can then be performed as,

$$\mathbf{\Phi} = a(\mathbf{X}^T\mathbf{W}^1), \tag{3}$$

$$\mathbf{F} = g(\mathbf{\Phi}\mathbf{W}^2). \tag{4}$$

Whilst this matrix notation is more concise, the choice to use the summation notation to describe the network here is deliberate. It is hoped that with the summation notation, relations to kernel and statistical theory discussed later in this chapter becomes clearer.

In the frequentist setting of NN learning, a MLE or MAP estimate is found through the minimisation of a non-convex cost function $J(x, y)$ w.r.t.

[6] Meaning no activation is used on the output layer, $g(x) = x$.

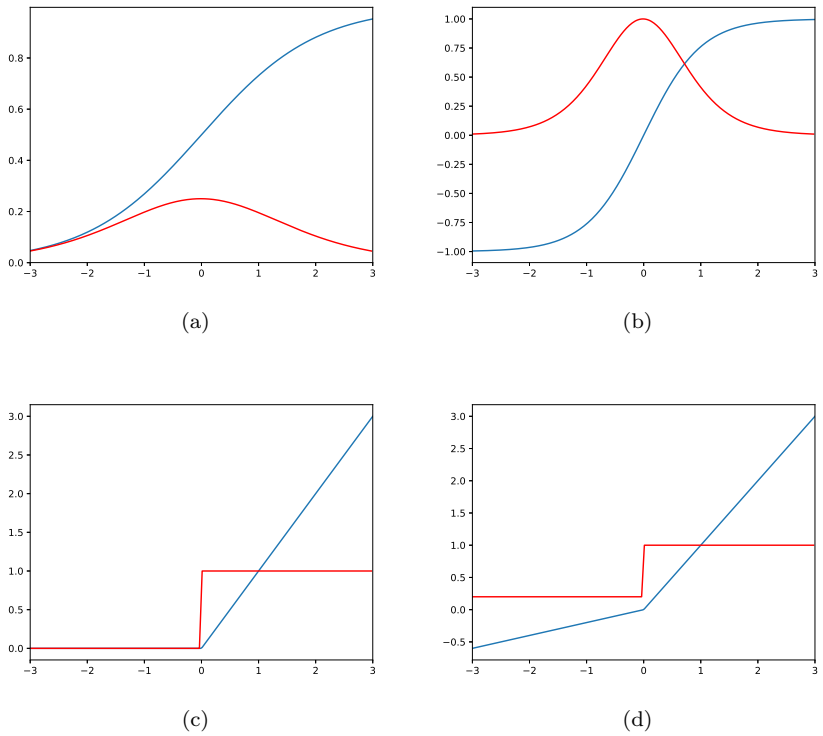


Fig. 3: Examples of commonly used activation functions in NNs. The output for each activation is shown in blue and the numerical derivative of each function is shown in red. These functions are (a) Sigmoid; (b) TanH; (c) ReLU; (d) Leaky-ReLU. Note the change in scale for the y-axis.

network weights. Minimisation of this cost-function is performed through backpropagation, where the output of the model is computed for the current parameter settings, partial derivatives w.r.t parameters are found and then used to update each parameter,

$$w_{t+i} = w_t - \alpha \frac{\partial J(x, y)}{\partial w_t}. \tag{5}$$

Equation 5 illustrates how backpropagation updates model parameters, with α representing the learning rate and the subscripts indicate the iteration in the training procedure. Partial derivatives for individual parameters at different layers in the network is found through application of the chain rule. This leads to the preference of discontinuous non-linearities such as the ReLU for

deep NNs, as the larger gradient of the ReLU assists in preventing vanishing gradients of early layers during training.

2.2 Bayesian Neural Networks

In the frequentist setting presented above, the model weights are not treated as random variables; weights are assumed to have a true value that is just unknown and the data we have seen is treated as a random variable. This may seem conterintuitive for what we want to achieve. We would like to learn what our unknown model weights are based of the information we have at hand. For statistical modelling the information available to us comes in the form of our acquired data. Since we do not know the value for our weights, it seems natural to treat them as a random variable. The Bayesian view of statistics uses this approach; unknown (or latent) parameters are treated as random variables and we want to learn a distribution of these parameters conditional on the what we can observe in the training data.

During the "learning" process of BNNs, unknown model weights are inferred based on what we do know or what we can observe. This is the problem of inverse probability, and is solved through the use of Bayes Theorem. The weights in our model $\boldsymbol{\omega}$ are hidden or latent variables; we cannot immediately observe their true distribution. Bayes Theorem allows us to represent a distribution over these weights in terms of probabilities we can observe, resulting in the distribution of model parameters conditional on the data we have seen $p(\boldsymbol{\omega}|\mathcal{D})$[7], which we call the posterior distribution.

Before training, we can observe the joint distribution between our weights and our data $p(\boldsymbol{\omega}, \mathcal{D})$. This joint distribution is defined by our prior beliefs over our latent variables $p(\boldsymbol{\omega})$ and our choice of model/likelihood $p(\mathcal{D}|\boldsymbol{\omega})$,

$$p(\boldsymbol{\omega}, \mathcal{D}) = p(\boldsymbol{\omega})p(\mathcal{D}|\boldsymbol{\omega}). \tag{6}$$

Our choice of network architecture and loss function is used to define the likelihood term in Equation 6. For example, for a 1-D homoscedastic regression problem with a mean squared error loss and a known noise variance, the likelihood is a Gaussian distribution with the mean value specified by the output of the network,

$$p(\mathcal{D}|\boldsymbol{\omega}) = \mathcal{N}\big(\mathbf{f}^{\omega}(\mathcal{D}), \sigma^2\big).$$

Under this modelling scheme, it is typically assumed that all samples from $\mathcal{D}$ are i.i.d., meaning that the likelihood can then be written as a product of the contribution from the N individual terms in the data set,

[7] $\mathcal{D}$ is used here to denote the set of training data $(\mathbf{x}, \mathbf{y})$.

$$p(\mathcal{D}|\boldsymbol{\omega}) = \prod_{i=1}^{N} \mathcal{N}\big(\mathbf{f}^{\omega}(\mathbf{x}_i), \sigma^2\big). \tag{7}$$

Our prior distribution should be specified to incorporate our belief as to how the weights should be distributed, prior to seeing any data. Due to the black-box nature of NNs, specifying a meaningful prior is challenging. In many practical NNs trained under the frequentist scheme, the weights of the trained network have a low magnitude, and are roughly centred around zero. Following this empirical observation, we may use a zero mean Gaussian with a small variance for our prior, or a spike-slab prior centred at zero to encourage sparsity in our model.

With the prior and likelihood specified, Bayes theorem is then applied to yield the posterior distribution over the model weights,

$$\pi(\boldsymbol{\omega}|\mathcal{D}) = \frac{p(\boldsymbol{\omega})p(\mathcal{D}|\boldsymbol{\omega})}{\int p(\boldsymbol{\omega})p(\mathcal{D}|\boldsymbol{\omega})d\boldsymbol{\omega}} = \frac{p(\boldsymbol{\omega})p(\mathcal{D}|\boldsymbol{\omega})}{p(\mathcal{D})}. \tag{8}$$

The denominator in the posterior distribution is called the marginal likelihood, or the evidence. This quantity is a constant with respect to the unknown model weights, and normalises the posterior to ensure it is a valid distribution.

From this posterior distribution, we can perform predictions of any quantity of interest. Predictions are in the form of an expectation with respect to the posterior distribution,

$$\mathbb{E}_{\pi}[f] = \int f(\boldsymbol{\omega})\pi(\boldsymbol{\omega}|\mathcal{D})d\boldsymbol{\omega}. \tag{9}$$

All predictive quantities of interest will be an expectation of this form. Whether it be a predictive mean, variance or interval, the predictive quantity will be an expectation over the posterior. The only change will be in the function $f(\boldsymbol{\omega})$ with which the expectation is applied to. Prediction can then be viewed as an average of the function f weighted by the posterior $\pi(\boldsymbol{\omega})$.

We see that the Bayesian inference process revolves around marginalisation (integration) over our unknown model weights. By using this marginalisation approach, we are able to learn about the generative process of a model, as opposed to an optimisation scheme used in the frequentist setting. With access to this generative model, our predictions are represented in the form of valid conditional probabilities.

In this description, it was assumed that many parameters such as the noise variance σ or any prior parameters were known. This is rarely the case, and as such we need to perform inference for these unknown variables. The Bayesian framework allows us to perform inference over these variables similarly to how we perform inference over our weights; we treat these additional variables as latent variables, assign a prior distribution (or sometimes called a hyper-

prior) and then marginalise over them to find our posterior. For more of a description of how this can be performed for BNNs, please refer to [33, 38].

For many models of interest, computation of the posterior (Equation 8) remains intractable. This is largely due to the computation of the marginal likelihood. For non-conjugate models or those that are non-linear in the latent variables (such as NNs), this quantity can be analytically intractable. For high dimensional models, a quadrature approximation of this integral can become computationally intractable. As a result, approximations for the posterior must be made. The following sections detail how approximate Bayesian inference can be achieved in BNNs.

2.3 Origin of Bayesian Neural Networks

From this survey and those conducted prior [39], the first instance of what could be considered a BNN was developed in [40]. This paper emphasises key statistical properties of NNs by developing a statistical interpretation of loss functions used. It was shown that minimisation of a squared error term is equivalent to finding the Maximum Likelihood Estimate (MLE) of a Gaussian. More importantly, it was shown that by specifying a prior over the network weights, Bayes Theorem can be used to obtain an appropriate posterior. Whilst this work provides key insights into the Bayesian perspective of NNs, no means for finding the marginal likelihood (evidence) is supplied, meaning that no practical means for inference is suggested. Denker and LeCun [41] extend on this work, offering a practical means for performing approximate inference using the Laplace approximation, though minimal experimental results are provided.

A NN is a generic function approximator. It is well known that as the limit of the number of parameters approaches infinity in a single hidden layer network, any arbitrary function can be represented [42, 43, 44]. This means that for the practical case, our finite training data set can be well approximated by a single layer NN as long as there are sufficient trainable parameters in the model. Similar to high-degree polynomial regression, although we can represent any function and even exactly match the training data in certain cases, as the number of parameters in a NN increases or the degree of the polynomial used increases, the model complexity increases leading to issues of overfitting. This leads to a fundamental challenge found in NN design; how complex should I make my model?

Building on the work of Gull and Skilling [45], MacKay demonstrates how a Bayesian framework naturally lends itself to handle the task of model design and comparison of generic statistical models [46]. In this work, two levels of inference are described: inference for fitting a model and inference for assessing the suitability of a model. The first level of inference is the typical application of Bayes rule for updating model parameters,

$$P(\boldsymbol{\omega}|\mathcal{D},\mathcal{H}_i) = \frac{P(\mathcal{D}|\boldsymbol{\omega},\mathcal{H}_i)P(\boldsymbol{\omega}|\mathcal{H}_i)}{P(\mathcal{D}|\mathcal{H}_i)}, \tag{10}$$

where $\boldsymbol{\omega}$ is the set of parameters in the generic statistical model, $\mathcal{D}$ is our data and $\mathcal{H}_i$ represents the i'th model used for this level of inference[8]. This is then described as,

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}}.$$

It is important to note that the normalising constant in Equation 10 is referred to as the evidence for the specific model of interest $\mathcal{H}_i$. Evaluation of the posterior remains intractable for most models of interest, so approximations must be made. In this work, the Laplace approximation is used.

Though computation of the posterior over parameters is required, the key aim of this work is to demonstrate methods of assessing the posterior over the model hypothesis $\mathcal{H}_i$. The posterior over model design is represented as,

$$P(\mathcal{H}_i|\mathcal{D}) \propto P(\mathcal{D}|\mathcal{H}_i)P(\mathcal{H}_i), \tag{11}$$

which translates to,

$$\text{Model Posterior} \propto \text{Evidence} \times \text{Model Prior}.$$

The data dependent term in Equation 11 is the evidence for the model. Despite the promising interpretation of the posterior normalisation constant, as described earlier, evaluation of this distribution is intractable for most BNNs. Assuming a Gaussian distribution, the Laplace approximation of the evidence can be found as,

$$\begin{aligned} P(\mathcal{D}|\mathcal{H}_i) &= \int P(\mathcal{D}|\boldsymbol{\omega},\mathcal{H}_i)P(\boldsymbol{\omega}|\mathcal{H}_i)d\boldsymbol{\omega} && (12)\\ &\approx P(\mathcal{D}|\boldsymbol{\omega}_{\text{MAP}},\mathcal{H}_i)\Big[P(\boldsymbol{\omega}_{\text{MAP}}|\mathcal{H}_i)\Delta\boldsymbol{\omega}\Big] && (13)\\ &= P(\mathcal{D}|\boldsymbol{\omega}_{\text{MAP}},\mathcal{H}_i)\Big[P(\boldsymbol{\omega}_{\text{MAP}}|\mathcal{H}_i)(2\pi)^{\frac{k}{2}}\det{}^{-\frac{1}{2}}\mathbf{A}\Big] && (14)\\ &= \text{Best Likelihood Fit} \times \text{Occam Factor}. \end{aligned}$$

This can be interpreted as a single Riemann approximation to the model evidence with the best likelihood fit representing the peak of the evidence, and the Occam factor is the width that is characterised by the curvature around the peak of the Gaussian. The Occam factor can be interpreted as the ratio of the width of the posterior $\Delta\boldsymbol{\omega}$ and the range of the prior $\Delta\boldsymbol{\omega}_0$ for the given model $\mathcal{H}_i$,

[8] $\mathcal{H}$ is used to refer to the model "hypothesis".

$$\text{Occam Factor} = \frac{\Delta\boldsymbol{\omega}}{\Delta\boldsymbol{\omega}_0}, \tag{15}$$

meaning that the Occam factor is the ratio of change in plausible parameter space from the prior to the posterior. Figure 4 demonstrates this concept graphically. With this representation, a complex model able to represent a large range of data will have a wider evidence, thus having a larger Occam factor. A simple model will have a lower capacity to capture a complex generative process, but a smaller range of data will be able to be modelled with greater certainty, resulting in a lower Occam Factor. This results in a natural regularisation for the complexity of a model. An unnecessarily complex model will typically result in a wide posterior, resulting in a large Occam factor and low evidence for the given model. Similarly, a wide or less informative prior will result in a reduced Occam factor, providing further intuition into the Bayesian setting of regularisation.

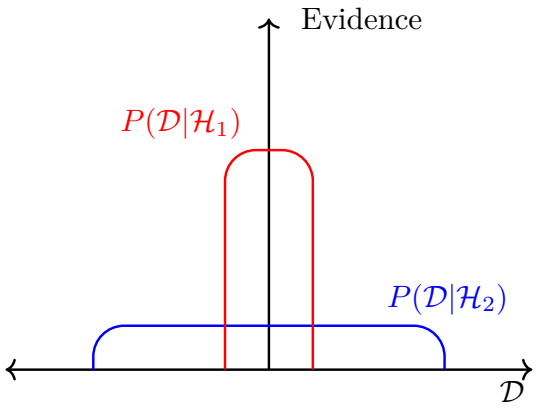


Fig. 4: Graphical illustration of how the evidence plays a role in investigating different model hypotheses. The simple model $\mathcal{H}_1$ is able to predict a small range of data with greater strength, while the more complex model $\mathcal{H}_2$ is able to represent a larger range of data, though with lower probability. Adapted from [46, 47].

Using this evidence framework requires computation of the marginal likelihood, which is an expensive (and the key challenge) within Bayesian modelling. Given the large investment required to approximate the marginal likelihood, it may be infeasible to compare many different architectures. Despite this, the use of the evidence framework can used to assess solutions for BNNs. For most NN architectures of interest, the objective function is non-convex with many local minima. Each local minima can be regarded as a possible solution for the inference problem. MacKay uses this as motivation to compare the solutions from each local minimum using the corresponding evidence

function [48]. This allows for assessment of model complexity at each solution without prohibitive computational requirements.

2.3.1 Early Variational Inference for BNNs

The machine learning community has continuously excelled at optimisation based problems. While many ML models, such as Support Vector Machines and Linear Gaussian Models result in a convex objective function, NNs have a highly non-convex objective function with many local minima. A difficult to locate global minimum motivates the use of gradient based optimisation schemes such as backpropagation [3]. This type of optimisation can be viewed in a Bayesian context through the lens of Variational Inference (VI).

VI is an approximate inference method that frames marginalisation required during Bayesian inference as an optimisation problem [49, 50, 51]. This is achieved by assuming the form of the posterior distribution and performing optimisation to find the assumed density that closest to the true posterior. This assumption simplifies computation and provides some level of tractability.

The assumed posterior distribution $q_{\boldsymbol{\theta}}(\boldsymbol{\omega})$ is a suitable density over the set of parameters ω, that is restricted to a certain family of distributions parameterised by θ. The parameters for this variational distribution are then adjusted to reduce the dissimilarity between the variational distribution and the true posterior $p(\boldsymbol{\omega}|\mathcal{D})$[9]. The means to measure similarity for VI is often the forward KL-Divergence between the variational and true distribution,

$$KL\Big(q_{\boldsymbol{\theta}}(\boldsymbol{\omega})||p(\boldsymbol{\omega}|\mathcal{D})\Big) = \int q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \log \frac{q_{\boldsymbol{\theta}}(\boldsymbol{\omega})}{p(\boldsymbol{\omega}|\mathcal{D})} d\boldsymbol{\omega}. \tag{16}$$

For VI, Equation 16 serves as the objective function we wish to minimise w.r.t variational parameters θ. This can be expanded out as,

$$\begin{aligned} \mathrm{KL}\Big(q_{\boldsymbol{\theta}}(\boldsymbol{\omega})||p(\boldsymbol{\omega}|\mathcal{D})\Big) &= \mathbb{E}_q\big[\log \frac{q_{\boldsymbol{\theta}}(\boldsymbol{\omega})}{p(\boldsymbol{\omega})} - \log p(\mathcal{D}|\boldsymbol{\omega})\big] + \log p(\mathcal{D}) && (17)\\ &= \mathrm{KL}\Big(q_{\boldsymbol{\theta}}(\boldsymbol{\omega})||p(\boldsymbol{\omega})\Big) - \mathbb{E}_q[\log p(\mathcal{D}|\boldsymbol{\omega})] + \log p(\mathcal{D}) && (18)\\ &= -\mathcal{F}[q_\theta] + \log p(\mathcal{D}), && (19) \end{aligned}$$

where $\mathcal{F}[q_\theta] = -\mathrm{KL}\Big(q_{\boldsymbol{\theta}}(\boldsymbol{\omega})||p(\boldsymbol{\omega})\Big) + \mathbb{E}_q[\log p(\mathcal{D}|\boldsymbol{\omega})]$. The combination of terms into $\mathcal{F}[q]$ is to separate the tractable terms from the intractable log marginal likelihood. We can now optimise this function using backpropagation, and since the log marginal likelihood does not depend on variational

[9] The model hypothesis $\mathcal{H}_i$ used previously will be omitted for further expressions, as little of the remaining key research items deal with model comparison and simply assume a single architecture and solution.

parameters θ, it's derivative evaluates to zero. This leaves only term of containing variational parameters, which is $\mathcal{F}[q_\theta]$.

This notation used in Equation 19, particularly the choice to include the negative of $\mathcal{F}[q_\theta]$ is deliberate to highlight a different but equivalent derivation to the identical result, and to remain consistent with existing literature. This result can be obtained by instead of minimising the KL-Divergence between the true and approximate distribution, but by approximating the intractable log marginal likelihood. Through application of Jensen's inequality, we can then find that $\mathcal{F}[q_\theta]$ forms a lower bound on the logarithm of the marginal likelihood [49, 52]. This can be seen by re-arranging Equation 19 and noting that the KL divergence is strictly ≥ 0 and only equals zero when the two distributions are equal. The logarithm of the marginal likelihood is equal to the sum of the KL divergence between the approximate and true posterior and $\mathcal{F}[q_\theta]$. By minimising the KL divergence between the approximate and true posterior, the closer $\mathcal{F}[q_\theta]$ will be to the logarithm of the marginal likelihood. For this reason, $\mathcal{F}[q_\theta]$ is commonly referred to as the Evidence Lower Bound (ELBO). Figure 5 illustrates this graphically.

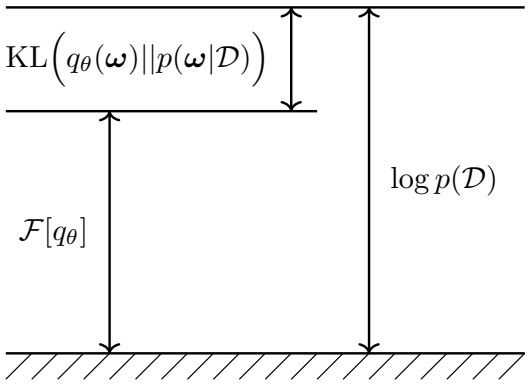


Fig. 5: Graphical illustration of how the minimisation of the KL divergence between the approximate and true posterior maximises the lower bound on the evidence. As the KL Divergence between our approximate and true posterior is minimised, the ELBO $\mathcal{F}[q_\theta]$ tightens to the log-evidence. Therefore maximising the ELBO is equivalent to minimising the KL divergence between the approximate and true posterior. Adapted from [53].

The first application of VI to BNNs was by Hinton and Van Camp [54], where they tried to address the problem of overfitting in NNs. They argued that by using a probabilistic perspective of model weights, the amount of information they could contain would be reduced and would simplify the network. Formulation of this problem was through an information theoretic basis, particularly the Minimum Descriptive Length (MDL) principle, though

its application results in a framework equivalent to VI. As is common in VI, the mean-field approach was used. Mean-Field Variational Bayes (MFVB) assumes a posterior distribution that factorises over parameters of interest. For the work in [54], the posterior distribution over model weights was assumed to be a factorisation of independent Gaussians,

$$q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) = \prod_{i=1}^{P} \mathcal{N}(w_i|\mu_i, \sigma_i^2), \tag{20}$$

where P is the number of weights in the network. For a regression network with a single hidden layer, an analytic solution for this posterior is available. The ability to achieve an analytic solution to the approximation is an desirable property, as analytic solutions significantly reduce the time to perform inference.

There are a few issues with this work, though one of the most prominent issues is the assumption of a posterior that factorises over individual network weights. It is well known that strong correlation between parameters in a NN is present. A factorised distribution simplifies computation by sacrificing the rich correlation information between parameters. MacKay highlighted this limitation in an early survey of BNNs [32] and offers insight into how a preprocessing stage of inputs to hidden layers could allow for more comprehensive approximate posterior distributions.

Barber and Bishop [53] again highlight this limitation, and offer a VI based approach that extends on the work in [54] to allow for full correlation between the parameters to be captured by using a full rank Gaussian for the approximating posterior. For a single hidden layer regression network utilising a Sigmoid activation, analytic expressions for evaluating the ELBO is provided[10]. This is achieved by replacing the Sigmoid with the appropriately scaled error function.

An issue with this modelling scheme is the increased number of parameters. For a full covariance model, the number of parameters scales quadratically with the number of weights in the network. To rectify this, Barber and Bishop propose a restricted form for the covariance often used in factor analysis, such that,

$$\mathbf{C} = \text{diag}(d_1^2, ..., d_n^2) + \sum_{i=1}^{s} \mathbf{s}_i \mathbf{s}_i^T, \tag{21}$$

where the diag operator creates a diagonal matrix from the vector $\mathbf{d}$ of size n, where n is the number of weights in the model. This form then scales linearly with the number of hidden units in the network.

These bodies of work provide important insight into how the prominent backpropagation method can be applied to challenging Bayesian problems. This allows for properties of the two areas of research to be merged and offer

[10] Numerical methods are required to evaluate certain terms in the analytic expression for the ELBO.

the benefits nominally seen in isolation. Complex regression tasks for large bodies of data sets could now be handled in a probabilistic sense using NNs.

Despite the insight offered by these methods, there are limitations to these methods. Both the work of Hinton and Van Camp and Barber and Bishop focus on development of a closed form representation of the networks[11]. This analytic tractability imposes many restrictions on the networks. As discussed previously, [54] assume a factorised posterior over individual weights which is unable to capture any correlation in parameters. Covariance structure is captured in [53], though the authors limit their analysis to the use of a Sigmoid activation function (which is well approximated by the error function), which is seldom used in modern networks due to the low magnitude in the gradient[12]. A key limitation common to both of these approaches is the restriction of a single hidden layer network.

As stated previously, a NN can approximate any function arbitrarily well by adding additional hidden units. For modern networks, empirical results have shown that similarly complex functions can be represented with fewer hidden units by increasing the number of hidden layers in the network. This has lead to the term "deep learning", where depth refers to the number of hidden layers. The reduction in number of weight variables is especially important for when trying to approximate the full covariance structure between layers. For example, correlation between hidden units within a single layer may be captured, while assuming that parameters between the different layers are independent. An assumption such as this can significantly reduce the number of correlation parameters. With modern networks having hundreds of millions of weights across many layers (with these networks only being able to offer point estimates), the need to develop practical probabilistic interpretations beyond a single layer is essential.

2.3.2 Hybrid Monte Carlo for BNNs

It is worthwhile at this point to reflect on the actual quantities of interest. So far the emphasis has been placed on finding good approximations for the posterior, though the accurate representation of the posterior is usually not the end design requirement. The main quantities of interest are predictive moments and intervals. We want to make good predictions accompanied by confidence information. The reason we emphasise computation of the posterior is that predictive moments and intervals are all computed as expectations of the posterior $\pi(\boldsymbol{\omega}|\mathcal{D})$[13]. This expectation is listed in Equation 9, and is repeated here for convenience,

[11] Although there are a large number of benefits to such an approach, as illustrated earlier.

[12] Analytic results may be achievable using other activation functions, such as TanH, which suffer less from such an issue.

[13] Note that π is used to represent the true posterior distribution here, as appose to q used previously to denote an approximation of the posterior.

$$\mathbb{E}_\pi[f] = \int f(\boldsymbol{\omega})\pi(\boldsymbol{\omega}|\mathcal{D})d\boldsymbol{\omega}.$$

This is why computation of the posterior is emphasised; accurate predictions rely on accurate approximations of the intractable posterior.

The previous methods employed optimisation based schemes such as VI or Laplace approximations of the posterior. In doing so, strong assumptions and restrictions on the form of posterior are enforced. The restrictions placed are often credited with inaccuracies induced in predictions, though this is not the only limitation.

As highlighted by [55, 56], the expectation computed for predictive quantities not just a probability mass, it the product of the probability mass and a volume. The probability mass is our posterior distribution $\pi(\boldsymbol{\omega}|\mathcal{D})$, and the volume $d\boldsymbol{\omega}$ over which we are integrating. It is likely that for all models of interest, the contribution of the expectation from this product of the density and volume will not be at the maximum for the mass. Therefore optimisation based schemes which consider only the mass can deliver inaccurate predictive quantities. To make accurate predictions with finite computational resources, we need to evaluate this expectation not just when the mass is greatest, but when the product of the mass and volume is largest. The most promising way to achieve this is with Markov Chain Monte Carlo (MCMC).

MCMC algorithms remains at the forefront of Bayesian research and applied statistics[14]. MCMC is a general approach for sampling from arbitrary and intractable distributions. The ability to sample from a distribution enables the use of Monte Carlo integration for prediction,

$$\mathbb{E}_\pi[f] = \int f(\boldsymbol{\omega})\pi(\boldsymbol{\omega}|\mathcal{D})d\boldsymbol{\omega} \approx \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{\omega}_i), \tag{22}$$

where $\boldsymbol{\omega}_i$ represents an independent sample from the posterior distribution. MCMC enables sampling from our posterior distribution, with the samples converging to when the product of the probability density and volume are greatest [55].

Assumptions previously made in VI methods, such as a factorised posterior are not required in the MCMC context. MCMC provides convergence to the true posterior as the number of samples approaches infinity. By avoiding such restrictions, with enough time and computing resources we can yield a solution that is closer to the true predictive quantities. This is an important challenge for BNNs, as the posterior distributions is typically quite complex.

Traditional MCMC methods demonstrate a random-walk behaviour, in that new proposals in the sequence are generated randomly. Due to the complexity and high dimension of the posterior in BNNs, this random-walk behaviour makes these methods unsuitable for performing inference in any reasonable time. To avoid the random-walk behaviour, Hybrid/Hamiltonian

[14] MCMC is regarded as one of the most influential algorithms of the 21st century [57].

Monte Carlo (HMC) can be employed to incorporate gradient information into the iterative behaviour. While HMC was initially proposed for statistical physics [58], Neal highlighted the potential for HMC to address Bayesian inference and specifically researched the applications to BNNs and the wider statistics community as a whole [38].

Given that HMC was initially proposed for physical dynamics, it is appropriate to build intuition for applied statistics through a physical analogy. Treat our parameters of interest $\boldsymbol{\omega}$ as a position variable. An auxiliary variable is then introduced to model the momentum $\mathbf{v}$ of our current position. This auxiliary variable is not of statistical interest, and is only introduced to aid in development of the system dynamics. With a position and momentum variable, we can represent the potential energy $U(\boldsymbol{\omega})$ and the kinetic energy $K(\mathbf{v})$ of our system. The total energy of a system is then represented as,

$$H(\boldsymbol{\omega}, \mathbf{v}) = U(\boldsymbol{\omega}) + K(\mathbf{v}). \tag{23}$$

We now consider the case of a lossless system, in that the total energy $H(\boldsymbol{\omega}, \mathbf{v})$ is constant[15]. This is described as a Hamiltonian system, and is represented as the following system of differential equations [59],

$$\frac{dw_i}{dt} = \frac{\partial H}{\partial v_i}, \tag{24}$$

$$\frac{dv_i}{dt} = -\frac{\partial H}{\partial w_i}, \tag{25}$$

where t represents time and the i denotes the individual elements in $\boldsymbol{\omega}$ and $\mathbf{v}$.

With the dynamics of the system defined, we wish to relate the physical interpretation to a probabilistic interpretation. This can be achieved through the canonical distribution[16],

$$P(\boldsymbol{\omega}, \mathbf{v}) = \frac{1}{Z} \exp\big(-H(\boldsymbol{\omega}, \mathbf{v})\big) = \frac{1}{Z} \exp\big(-U(\boldsymbol{\omega})\big) \exp\big(-K(\mathbf{v})\big), \tag{26}$$

where Z is a normalising constant and $H(\boldsymbol{\omega}, \mathbf{v})$ is our total energy as defined in Equation 23. From this joint distribution, we see that our position and momentum variable are independent.

Our end goal is to find predictive moments and intervals. For a Bayesian this makes the key quantity of interest the posterior distribution. Therefore, we can set the potential energy which we wish to sample from to,

[15] The values for $\boldsymbol{\omega}$ and $\mathbf{v}$ will change, though the total energy of the system will remain constant

[16] As is commonly done, we assume the temperature variable included in physical representations of the canonical distribution is set to one. For more information, see [59, p. 11], [60, p. 123].

$$U(\boldsymbol{\omega}) = -\log\Big(p(\boldsymbol{\omega})p(\mathcal{D}|\boldsymbol{\omega})\Big). \tag{27}$$

Within HMC, the kinetic energy can be freely selected from a wide range of suitable functions, though is typically chosen such that it's marginal distribution of $\mathbf{v}$ is a diagonal Gaussian centred at the origin.

$$K(\mathbf{v}) = \mathbf{v}^T M^{-1}\mathbf{v}, \tag{28}$$

where M is a diagonal matrix referring to the "mass" of our variables in this physical interpretation. Although this is the most common kinetic energy function used, it may not be the most suitable. [55] surveys the selection the design of other Gaussian kinetic energies with an emphasis on the geometric interpretations. It is also highlighted that selection of appropriate kinetic energy functions remains an open research topic, particularly in the case of non-Gaussian functions.

Since Hamiltonian dynamics leaves the total energy invariant, when implemented with infinite precision, the dynamics proposed are reversible. Reversibility is a sufficient property to satisfy the condition of detailed balance, which is required to ensure that the target distribution (the posterior we are trying to sample from) remains invariant. For practical implementations, numerical errors arise due to discretisation of variables. The discretisation method most commonly employed is the leapfrog method. The leapfrog method specifies a step size ϵ and a number of steps L to be used before possibly accepting the new update. The leapfrog method first performs a half update of the momentum variable v, followed by a full update of the position w and then the remaining half update of the momentum [59],

$$v_i(t+\frac{\epsilon}{2}) = v_i(t) + \frac{\epsilon}{2}\frac{dv_i}{dt}(v(t)), \tag{29}$$

$$w_i(t+\epsilon) = w_i(t) + \epsilon\frac{dw_i}{dt}(w(t)), \tag{30}$$

$$v_i(t+\epsilon) = v_i(t+\frac{\epsilon}{2}) + \frac{\epsilon}{2}\frac{dv_i}{dt}(v(t+\frac{\epsilon}{2})). \tag{31}$$

If the value of ϵ is chosen such that this dynamical system remains stable, it can be shown that this leapfrog method preserves the volume (total energy) of the Hamiltonian.

For expectations to be approximated using 22, we require each sample $\boldsymbol{\omega}_i$ to be independent from subsequent samples. We can achieve practical independence[17] by using multiple leapfrog steps L. In this way, after L leapfrog steps of size ϵ, the new position is proposed. This reduces correlation between samples and can allow for faster exploration of the posterior space. A Metropolis step is then applied to determine whether this new proposal is accepted as the newest state in the Markov Chain [59].

[17] Where for all practical purposes each sample can be viewed as independent.

For the BNN proposed by [38], a hyper-prior $p(\boldsymbol{\gamma})$ is induced to model the variance over prior parameter precision and likelihood precision. A Gaussian prior is used for the prior over-parameters and the likelihood is set to be Gaussian. Therefore, the prior over the $\boldsymbol{\gamma}$ was Gamma distributed, such that it was conditionally conjugate. This allows for Gibbs sampling to be used for performing inference over hyperparameters. HMC is then used to update the posterior parameters. Sampling from the joint posterior $P(\boldsymbol{\omega}, \boldsymbol{\gamma}|\mathcal{D})$ then involves alternating between the Gibbs sampling step for the hyperparameters and Hamiltonian dynamics for the model parameters. Superior performance of HMC for simple BNN models was then demonstrated and compared with random walk MCMC and Langevin methods [38].

2.4 Modern BNNs

Considerably less research was conducted into BNNs following early work of Neal, MacKay and Bishop proposed in the 90s. This relative stagnation was seen throughout the majority of NN research, and was largely due to the high computational demand for training NNs. NNs are parametric models that are able to capture any function with arbitrary accuracy, but to capture complex functions accurately requires large networks with many parameters. Training of such large networks became infeasible even for the traditional frequentist perspective, and the computational demand significantly increases to investigate the more informative Bayesian counterpart.

Once it was shown that general purpose GPUs could accelerate and allow training of large models, interest and research into NNs saw a resurgence. GPUs enabled large scale parallelism of the linear algebra performed during back propagation. This accelerated computation has allowed for training of deeper networks, where successive concatenation of hidden layers is used. With the proficiency of GPUs for optimising complex networks and the great empirical success seen by such models, interest into BNNs resumed.

Modern research into BNNs has largely focused on the VI approach, given that these problems can be optimised using a similar backpropagation approach used for point estimate networks. Given that the networks offering the most promising results use multiple layers, the original VI approaches shown in [54, 53], which focus on analytical approximations for regression networks utilising a single hidden layer became unsuitable. Modern NNs now exhibit considerably different architectures with varying dimensions, hidden layers, activations and applications. More general approaches for viewing networks in a probabilistic sense was required.

Given the large scale of modern networks, large data sets are typically required for robust inference[18]. For these large data sets, evaluation of the complete log-likelihood becomes infeasible for training purposes. To combat this, a Stochastic Gradient Descent (SGD) approach is used, where mini-batches of the data are used to approximate the likelihood term, such that our variational objective becomes,

$$\mathcal{L}(\boldsymbol{\omega}, \boldsymbol{\theta}) = -\frac{N}{M}\sum_{i=1}^{N}\mathbb{E}_q[\log\big(p(\mathcal{D}_i|\boldsymbol{\omega})\big)] + \mathrm{KL}\Big(q_{\boldsymbol{\theta}}(\boldsymbol{\omega})||p(\boldsymbol{\omega})\Big), \tag{32}$$

where $\mathcal{D}_i \subset \mathcal{D}$, and each subset is of size M. This provides an efficient way to utilise large data sets during training. After passing a single subset $\mathcal{D}_i$, back-propagation is applied to update the model parameters. This sub-sampling of the likelihood induces noise into our inference process, hence the name SGD. This noise that is induced is expected to average out over evaluation of each individual subset [61]. SGD is the most common method for training NNs and BNNs utilising a VI approach.

A key paper in the resurgence of BNN research was published by Graves [62]. This work proposes a MFVB treatment using a factorised Gaussian approximate posterior. The key contribution of this work is the computation of the derivatives. The VI objective (ELBO) can be viewed as a sum of two expectations,

$$\mathcal{F}[q_\theta] = \mathbb{E}_q[\log\big(p(\mathcal{D}|\boldsymbol{\omega})\big)] - \mathbb{E}_q[\log q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) - \log p(\boldsymbol{\omega})] \tag{33}$$

It is these two expectations that we need to optimise w.r.t model parameters, meaning that we require the gradient of expectations. This work shows how using the gradient properties of a Gaussian proposed in [63] can be used to perform parameter updates,

$$\nabla_{\boldsymbol{\mu}}\ \mathbb{E}_{p(\boldsymbol{\omega})}[f(\boldsymbol{\omega})] = \mathbb{E}_{p(\boldsymbol{\omega})}[\nabla_{\boldsymbol{\omega}} f(\boldsymbol{\omega})], \tag{34}$$

$$\nabla_{\Sigma}\ \mathbb{E}_{p(\boldsymbol{\omega})}[f(\boldsymbol{\omega})] = \frac{1}{2}\mathbb{E}_{p(\boldsymbol{\omega})}[\nabla_{\boldsymbol{\omega}}\nabla_{\boldsymbol{\omega}} f(\boldsymbol{\omega})]. \tag{35}$$

MC integration could be applied to Equations 34 and 35 to approximate the gradient of the mean and variance parameters. This framework allows for optimisation of the ELBO to generalise to any log-loss parametric model.

Whilst addressing the problem of applying VI to complex BNNs with more hidden layers, practical implementations have shown inadequate performance which is attributed to large variance in the MC approximations of the gradient computations [64]. Developing gradient estimates with reduced variance has become an integral research topic in VI [65]. Two of the most common

[18] Neal [38] argues that this not true for Bayesian modelling; claims that if suitable prior information is available, complexity of a model should only be limited by computational resources.

methods for deriving gradient approximations rely on the use of score functions and path-wise derivative estimators.

Score function estimators rely on the use of the log-derivative property, such that,

$$\frac{\partial}{\partial\theta}p(x|\theta) = p(x|\theta)\frac{\partial}{\partial\theta}\log p(x|\theta). \tag{36}$$

Using this property, we can form Monte Carlo estimates of the derivatives of an expectation, which is often required in VI,

$$\begin{aligned}\nabla_\theta \mathbb{E}_q[f(\omega)] &= \int f(\omega)\nabla_\theta q_\theta(\omega)\partial\omega \\ &= \int f(\omega)q_\theta(\omega)\nabla_\theta \log\Big(q_\theta(\omega)\Big)\partial\omega \\ &\approx \frac{1}{L}\sum_{i=1}^{L} f(\omega_i)\nabla_\theta \log\Big(q_\theta(\omega_i)\Big).\end{aligned} \tag{37}$$

A common problem with score function gradient estimators is that they exhibit considerable variance [65]. One of the most common methods to reduce the variance in Monte Carlo estimates is the introduction of control variates [66].

The second type of gradient estimator commonly used in the VI literature is the pathwise derivative estimator. This work builds on the "reparameterisation trick" [67, 68, 69], where a random variable is represented as a deterministic and differentiable expression. For example, for a Gaussian with $\boldsymbol{\theta} = \{\mu, \boldsymbol{\sigma}\}$,

$$\begin{aligned}\boldsymbol{\omega} &\sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2) \\ \boldsymbol{\omega} &= g(\boldsymbol{\theta}, \boldsymbol{\epsilon}) = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}\end{aligned} \tag{38}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\odot$ represents the Hadamard product. Using this method allows for efficient sampling for Monte Carlo estimates of expectations. This is shown in [68], that with $\boldsymbol{\omega} = g(\boldsymbol{\theta}, \boldsymbol{\epsilon})$, we know that $q(\boldsymbol{\omega}|\boldsymbol{\theta})d\boldsymbol{\omega} = p(\boldsymbol{\epsilon})d\boldsymbol{\epsilon}$. Therefore, we can show that,

$$\begin{aligned}\int q_{\boldsymbol{\theta}}(\boldsymbol{\omega})f(\boldsymbol{\omega})d\boldsymbol{\omega} &= \int p(\boldsymbol{\epsilon})f(\boldsymbol{\omega})d\boldsymbol{\epsilon} \\ &= \int p(\boldsymbol{\epsilon})f(g(\boldsymbol{\theta}, \boldsymbol{\epsilon}))d\boldsymbol{\epsilon} \\ \approx \frac{1}{M}\sum_{i=1}^{M} f(g(\boldsymbol{\theta}, \boldsymbol{\epsilon}_i)) &= \frac{1}{M}\sum_{i=1}^{M} f(\boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}_i)\end{aligned} \tag{39}$$

Since Equation 39 is differentiable w.r.t $\boldsymbol{\theta}$, gradient descent methods can be used to optimise this expectation approximation. This is an important property in VI, since the VI objective contains expectations of the log-likelihood

that are often intractable. The reparameterisation trick serves as the basis for pathwise-gradient estimators. Pathwise estimators are favourable for their reduced variance over score function estimators [68, 65].

A key benefit of having a Bayesian treatment of NNs is the ability to extract uncertainty in our models and their predictions. This has been a recent research topic of high interest in the context of NNs. Promising developments regarding uncertainty estimation in NNs has been found by relating existing regularisation techniques such as Dropout [70] to approximate inference. Dropout is a Stochastic Regularisation Technique (SRT) that was proposed to address overfitting commonly seen in point-estimate networks. During training, Dropout introduces an independent random variable that is Bernoulli distributed, and multiplies each individual weight element-wise by a sample from this distribution. For example, a simple MLP implementing Dropout is of the form,

$$\begin{aligned} \rho_u &\sim \text{Bernoulli}(p), \\ \phi_j &= \theta\Big(\sum_{i=1}^{N_1} (x_i \rho_u) w_{ij}\Big). \end{aligned} \tag{40}$$

Looking at Equation 40, it can be seen that the application of Dropout introduces stochasticity into the network parameters in a similar manner as to that of the reparameterisation trick shown in Equation 38. A key difference is that in the case of Dropout, stochasticity is introduced into the input space, as appose to the parameter space required for Bayesian inference. Yarin Gal [39] identified this similarity, and demonstrated how noise introduced through the application of Dropout can be transferred to the networks weights efficiently as,

$$\mathbf{W}_\rho^1 = \text{diag}(\boldsymbol{\rho})\mathbf{W}^1 \tag{41}$$

$$\boldsymbol{\Phi}_\rho = a\big(\mathbf{X}^T \mathbf{W}_\rho^1\big). \tag{42}$$

Where $\boldsymbol{\rho}$ is a vector sampled from the Bernoulli distribution, and the $\text{diag}(\cdot)$ operator creates a square diagonal matrix from a vector. In doing this it can be seen that a single dropout variable is shared amongst each row of the weight matrix, allowing some correlation within rows to be maintained. By viewing the stochastic component in terms of network weights, the formulation becomes suitable for approximate inference using the VI framework. In this work, the approximate posterior is of the form of a Bernoulli distribution multiplied by the network weights.

The reparameterisation trick is then applied to allow for partial derivatives w.r.t. network parameters to be found. The ELBO is then formed and backpropagation is performed to maximise the lower bound. MC integration is used to approximate the analytically intractable expected log-likelihood. The KL divergence between the approximate posterior and the prior distri-

bution in the ELBO is then found by approximating the scaled Bernoulli approximate posterior as a mixture of two Gaussians with very small variance.

In parallel to this work, Kingma *et al.* [71] identified this same similarity between Dropout and it's potential for use within a VI framework. As appose to the typical Bernoulli distributed r.v. introduced in Dropout, [71] focuses attention to the case when the introduced r.v. is Gaussian [72]. It is also shown how with selection of an appropriate prior that is independent of parameters, current applications of NNs using dropout can be viewed as approximate inference.

Kingma *et al.* also aims to reduce the variance in the stochastic gradients using a refined, local reparameterisation. This is done by instead of sampling from the weight distribution before applying the affine transformation, the sampling is performed afterwards. For example, consider a MFVB case where each weight is assumed to be an independent Gaussian $w_{ij} \sim \mathcal{N}(\mu_{ij}, \sigma_{ij}^2)$. After the affine transformation $\phi_j = \sum_{i=1}^{N_1}(x_i\rho_i)w_{ij}$, the posterior distribution of ϕ_j conditional on the inputs will also be a factorised Gaussian,

$$q(\phi_j|\mathbf{x}) = \mathcal{N}(\gamma_j, \delta_j^2), \tag{43}$$

$$\gamma_j = \sum_{i=1}^{N} x_i\mu_{i,j}, \tag{44}$$

$$\delta_j^2 = \sum_{i=1}^{N} x_i^2\sigma_{i,j}^2. \tag{45}$$

It is advantageous to sample from this distribution for ϕ as appose to the distribution of the weights w themselves, as this results in a gradient estimator whose variance scales linearly with the number of mini-batches used during training.[19]

These few bodies of work are important in addressing the serious lack of rigour seen in ML research. For example, the initial Dropout paper [70] lacks any significant theoretical foundation. Instead, the method cites a theory for sexual reproduction [73] as motivation for the method, and relies heavily on the empirical results given. These empirical results have been further demonstrated throughout many high impact[20] research items which utilise this technique merely as a regularisation method. The work in [39] and [71] show that there is theoretical justification for such an approach. In attempts to reduce the effect of overfitting in a network, the frequentist methodology relied on the application of a weakly justified technique that shows empirical success, while Bayesian analysis provides a rich body of theory that naturally leads to a meaningful understanding of this powerful approximation.

[19] This method also has computational advantages, as the dimension of ϕ is typically much lower than that of $\boldsymbol{\omega}$.

[20] At the time of writing, [70] has over ten thousand citations.

Whilst addressing the problem of applying VI to complex BNNs with more hidden layers, practical implementations have shown inadequate performance which is attributed to large variance in the MC approximations of the gradient computations. Hernandez *et al.*[64] acknowledge this limitation and propose a new method for practical inference of BNNs titled Probabilistic Back Propagation (PBP). PBP deviates from the typical VI approach, and instead employs an Assumed Density Filtering (ADF) method [74]. In this format, the posterior is updated in an iterative fashion through application of Bayes rule,

$$p(\boldsymbol{\omega}_{t+1}|\mathcal{D}_{t+1}) = \frac{p(\boldsymbol{\omega}_t|\mathcal{D}_t)p(\mathcal{D}_{t+1}|\boldsymbol{\omega}_t)}{p(\mathcal{D}_{t+1})}. \tag{46}$$

As opposed to traditional network training where the predicted error is the objective function, PBP uses a forward pass to compute the log-marginal probability of a target and updates the posterior distribution of network parameters. The moment matching method defined in [75] updates the posterior using a variant of backpropagation, whilst maintaining equivalent mean and variance between the approximate and variational distribution,

$$\mu_{t+1} = \mu_t + \sigma_t \frac{\partial \log p(\mathcal{D}_{t+1})}{\partial \mu} \tag{47}$$

$$\sigma_{t+1} = \sigma_t + \sigma_t^2 \Big[\big(\frac{\partial p(\mathcal{D}_{t+1})}{\partial \mu_t} \big)^2 - 2 \frac{\partial p(\mathcal{D}_{t+1})}{\partial \sigma} \Big]. \tag{48}$$

Experimental results on multiple small data-sets illustrate reasonable performance in terms of predicted accuracy and uncertainty estimation when compared with HMC methods for simple regression problems [64]. A key limitation of this method is the computational bottleneck introduced by the online training method. This approach may be suitable for some applications, or for updating existing BNNs with additional data as it becomes available, though for performing inference on large data sets the method is computationally prohibitive.

A promising method for approximate inference in BNNs was proposed by Blundell *et al.*, titled "Bayes by Backprop" [76]. The method utilises the reparameterisation trick to show how unbiased estimates of the derivative of an expectation can be found. For a random variable $\boldsymbol{\omega} \sim q_{\boldsymbol{\theta}}(\boldsymbol{\omega})$ that can be reparameterised as deterministic and differentiable function $\boldsymbol{\omega} = g(\boldsymbol{\epsilon}, \boldsymbol{\theta})$, the derivative of the expectation of an arbitrary function $f(\boldsymbol{\omega}, \boldsymbol{\theta})$ can be expressed as,

$$\frac{\partial}{\partial \boldsymbol{\theta}} \mathbb{E}_q[f(\boldsymbol{\omega}, \boldsymbol{\theta})] = \frac{\partial}{\partial \boldsymbol{\theta}} \int q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) f(\boldsymbol{\omega}, \boldsymbol{\theta}) d\boldsymbol{\omega} \tag{49}$$

$$= \frac{\partial}{\partial \boldsymbol{\theta}} \int p(\boldsymbol{\epsilon}) f(\boldsymbol{\omega}, \boldsymbol{\theta}) d\boldsymbol{\epsilon} \tag{50}$$

$$= \mathbb{E}_{q(\epsilon)} \Big[\frac{\partial f(\boldsymbol{\omega}, \boldsymbol{\theta})}{\partial \boldsymbol{\omega}} \frac{\partial \boldsymbol{\omega}}{\partial \boldsymbol{\theta}} + \frac{\partial f(\boldsymbol{\omega}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \Big]. \tag{51}$$

In the Bayes by Backprop algorithm, the function $f(\boldsymbol{\omega}, \boldsymbol{\theta})$ is set as,

$$f(\boldsymbol{\omega}, \boldsymbol{\theta}) = \log \frac{q_{\boldsymbol{\theta}}(\boldsymbol{\omega})}{p(\boldsymbol{\omega})} - \log p(\mathbf{X}|\boldsymbol{\omega}). \tag{52}$$

This $f(\boldsymbol{\omega}, \boldsymbol{\theta})$ can be seen as the argument for the expectation performed in Equation 17, which is part of the lower bound.

Combining Equations 51 and52,

$$\mathcal{L}(\boldsymbol{\omega}, \boldsymbol{\theta}) = \mathbb{E}_q[f(\boldsymbol{\omega}, \boldsymbol{\theta})] = \mathrm{e}_q \left[\log \frac{q_{\boldsymbol{\theta}}(\boldsymbol{\omega})}{p(\boldsymbol{\omega})} - \log p(\mathcal{D}|\boldsymbol{\omega}) \right] = -\mathcal{F}[q_{\boldsymbol{\theta}}] \tag{53}$$

which is shown to be the negative of the ELBO, meaning that Bayes by Backprop aims to minimise the KL divergence between the approximate and true posterior. Monte Carlo integration is used[21] to approximate the cost in Equation 53,

$$\mathcal{F}[q_{\boldsymbol{\theta}}] \approx \sum_{i=1}^{N} \log \frac{q_{\boldsymbol{\theta}}(\boldsymbol{\omega}_i)}{p(\boldsymbol{\omega}_i)} - \log p(\mathbf{X}|\boldsymbol{\omega}_i) \tag{54}$$

where $\boldsymbol{\omega}_i$ is the i^{th} sample from $q_{\boldsymbol{\theta}}(\boldsymbol{\omega})$. With the approximation in Equation 54, the unbiased gradients can be found using the result shown in Equation 51.

For the Bayes by Backprop algorithm, a fully factorised Gaussian posterior is assumed such that $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\rho}\}$, where $\boldsymbol{\sigma} = \text{softplus}(\boldsymbol{\rho})$ is used to ensure the standard deviation parameter is positive. With this, the distribution of weights $\boldsymbol{\omega} \sim \mathcal{N}(\boldsymbol{\mu}, \text{softplus}(\boldsymbol{\rho})^2)$ in the network are reparameterised as,

$$\boldsymbol{\omega} = g(\boldsymbol{\theta}, \boldsymbol{\epsilon}) = \mu + \text{softplus}(\boldsymbol{\rho}) \odot \boldsymbol{\epsilon}. \tag{55}$$

In this BNN, the trainable parameters are $\boldsymbol{\mu}$ and $\boldsymbol{\rho}$. Since a fully factorised distribution is used, following from Equation 20, the logarithm of the approximate posterior can be represented as,

$$\log q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) = \sum_{l,j,k} \log \Big(\mathcal{N}(w_{ljk}; \mu_{ljk}, \sigma_{ljk}^2) \Big). \tag{56}$$

The complete Bayes by Backprop algorithm is described in Algorithm 1.

[21] Some terms may be tractable in this integrand, depending on the form of the prior and posterior approximation. MC integration allows for arbitrary distributions to be approximated.

Algorithm 1 Bayes by Backprop (BbB) algorithm [76]

procedure BBB($\boldsymbol{\theta}, \mathbf{X}, \alpha$)
  repeat
    $\mathcal{F}[q_{\boldsymbol{\theta}}] \leftarrow 0$ ▷ Initialise cost
    for i in $[1, ..., N]$ **do** ▷ Number of samples for MC estimate
      Sample $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$
      $\boldsymbol{\omega} \leftarrow \boldsymbol{\mu} + \text{softplus}(\boldsymbol{\rho}) \cdot \boldsymbol{\epsilon}_i$
      $\mathcal{L} \leftarrow \log q(\boldsymbol{\omega}|\boldsymbol{\theta}) - \log p(\boldsymbol{\omega}) - \log p(\mathbf{X}|\boldsymbol{\omega})$
      $\mathcal{F}[q_{\boldsymbol{\theta}}] += \text{sum}(\mathcal{L})/N$ ▷ Sum across all log of weights in set $\boldsymbol{\omega}$
    end for
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_\theta \mathcal{F}[q_{\boldsymbol{\theta}}]$ ▷ Update parameters
  until convergence
end procedure

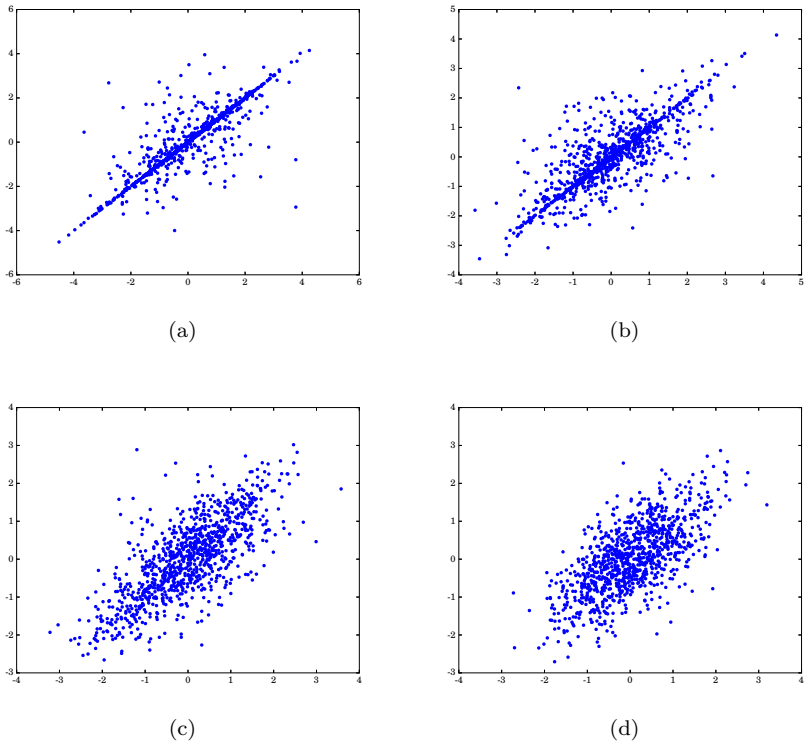
2.5 Gaussian Process Properties of BNNs

Neal [38] also provided derivation and experimentation results to illustrate that for a network with a single hidden layer, a Gaussian Process (GP) prior over the network output arises when the number of hidden units approaches infinity, and a Gaussian prior is placed over parameters[22]. Figure 6 illustrates this result.

This important link between NNs and GPs can be seen from Equations 1 and 2. From these expressions, it can be seen that a NN with a single hidden layer is a sum of N parametric basis functions applied to the input data. If the parameters for each basis function in Equation 1 are r.v.'s, Equation 2 becomes the sum of r.v.'s. Under the central limit theorem, as the number of hidden layers $N \rightarrow \infty$, the output becomes Gaussian. Since the output is then described as an infinite sum of basis functions, the output can be seen to become a GP. Following from a full derivation of this result and the illustrations show in Figure 6, [38] shows how an approximate Gaussian nature is achieved for finite computing resources and how the magnitude of this sum can be maintained. Williams then demonstrated how the form of the covariance function could be analysed for different activation functions [77]. The relation between GPs and infinitely wide networks with a single hidden layer work has recently been extended to the case of deep networks [78].

Identification of this link has motivated many research works in BNNs. GPs provide many of the properties we wish to obtain, such as reliable uncertainty estimates, interpretability and robustness. GPs deliver these benefits at the cost of predictive performance and exponentially large computational resources required as the size of data sets increase. This link between GPs and BNNs has motivated the merging of the two modelling schemes; maintaining the predictive performance and flexibility seen in NNs while incorporating

[22] For a regression model with no non-linear activation function placed on the output units.



(a) (b)

(c) (d)

Fig. 6: Illustration of GP prior induced on output when placing a Gaussian prior over parameters as the network size increases. Experimentation replicated from [38, p. 33]. Each dot corresponds to the output of a network with parameters sampled from the prior, with the x-axis as $f(0.2)$ and the y-axis as $f(-0.4)$. For each network, the number of hidden units are (a) 1, (b) 3, (c) 10, (d) 100.

the robustness and probabilistic properties enabled by GPs. This has led to the development of the Deep Gaussian Process.

Deep GPs are a cascade of individual GPs, where much like a NN, the output of the previous GP serves as the input to a new GP [79, 80]. This stacking of GPs allows for learning of non-Gaussian densities from a combination of GPs[23]. A key challenge with GPs is fitting to large data sets, as the dimensions of the Gram matrix for a single GP is quadratic with the number of data points. This issue is amplified with a Deep GP, as each individual GP in the cascade induces an independent Gram matrix. Further-

[23] A complete introduction to Deep GPs, along with code and lectures has been offered by Neil Lawrence [81].

more, the marginal likelihood for Deep GPs are analytically intractable due to non-linearities in the functions produced. Building on the work in [82], Damianou and Lawrence [79] use a VI approach to create an approximation that is tractable and reduces computational complexity to that typically seen in sparse GPs [83].

Deep GPs have shown how the GPs can benefit from methodology seen in NNs. Gal and Ghahramani [84, 85, 86] built of this work to show how a Deep GP can be approximated with a BNN[24]. This is an expected result; given that Neal [38] identified an infinitely wide network with a single hidden layer converges to a Gaussian process, by concatenating multiple infinitely wide layers we converge to a deep Gaussian process.

Alongside this analysis of deep Gaussian processes, [84, 85, 86] build on the work in [77] to analyse the relationship between the modern non-linear activation used within BNNs and the covariance function for a GP. This is promising work that could allow for more principled selection of activation functions in NNs, similar to that of GPs. Which activation functions will yield a stationary process? What is the expected length scale for our process? These questions may be able to be addressed using the rich theory existing for GPs.

The GP properties are not restricted to MLP BNNs. Recent research has identified certain relationships and conditions that induce GP properties in convolutional BNNs [87, 88]. This result is expected since CNNs can be implemented as MLPs with structure enforced in the weights. What this work identifies is how the GP is constructed when this structure is enforced. Van der Wilk *et al.*[89] proposed the Convolutional Gaussian Process, which implements a patch based operation similar to that seen in CNNs to define the GP prior over functions. Practical implementation of this method requires the use of approximation methods, due to the prohibitive cost of evaluating large data sets, and even evaluation at each patch. Inducing points are formed with a VI framework to reduce the number of data points to evaluate and the number of patches evaluated.

2.6 Limitations in Current BNNs

Whilst great effort has been put into developing Bayesian methods for performing inference in NNs, there are significant limitations to these methods and many gaps remaining in the literature. A key limitation is the heavy reliance on VI methods. Within the VI framework, the most common approach is the Mean Field approach. MFVB provides a convenient way to represent an approximate posterior distribution by enforcing strong assumptions of independence between parameters. This assumption allows for factorised distributions to be used to approximate the posterior. This assumption of inde-

[24] Approximation becomes a Deep GP as the number of hidden units in each layer approaches ∞.

pendence significantly reduces the computational complexity of approximate inference at the cost of probabilistic accuracy.

A common finding with VI approaches is that resulting models are over-confident, in that predictive means can be accurate while variance is considerably under estimated [90, 91, 92, 51, 93]. This phenomenon is described in Section 10.1.2 of [2] and Section 21.2.2 of [35], both of which are accompanied by examples and intuitive figures to illustrate this property. This property of under-estimated variance is present within much of the current research in BNNs [39]. Recent work has aimed to address these issues through the use of noise contrastive priors [94] and through use of calibration data sets [95]. The authors in [96] employ the use of the concrete distribution [97] to approximate the Bernoulli parameter in the MC Dropout method [85], allowing for it to be optimised, resulting in posterior variances that are better calibrated. Despite these efforts, the task of formulating reliable and calibrated uncertainty estimates within a VI framework for BNNs remains unsolved.

It is reasonable to consider that perhaps the limitations of the current VI approaches are influenced by the choice of approximate distribution used, particularly the usual MFVB approach of independent Gaussians. If more comprehensive approximate distributions are used, will our predictions be more consistent with the data we have and haven't seen? Mixture based approximations have been proposed for the general VI approach [98, 49], though introduction of N mixtures increases the number of variational parameters by N. Matrix-Normal approximate posteriors have been introduced to the case of BNNs [99], which reduces the number of variational parameters in the model when compared with a full rank Gaussian, though this work still factorises over individual weights, meaning no covariance structure is modelled[25]. MCDropout is able to maintain some correlation information within the rows of weight matrix, at the compromise of a low entropy approximate posterior.

A recent approach for VI has been proposed to capture more complex posterior distributions through the use of normalising flows [100, 101]. Within a normalising flow, the initial distribution "flows" through a sequence of invertible functions to produce a more complex distribution. This can be applied within the VI framework using amortized inference [102]. Amortized inference introduces an inference network which maps input data to the variational parameters of generative model. These parameters are then used to sample from the posterior of the generative process. The use of normalising flows has been extended to the case of BNNs [103]. Issues arise with this approach relating to the computational complexity, along with limitations of amortized inference. Normalising flows requires the calculation of the determinant of the Jacobian for applying the change of variables used for each invertible function, which can be computationally expensive for certain models. Computational complexity can be reduced by restricting the normalising flow to contain in-

[25] Though this work highlights that even with a fully factorised distribution over weights, the outputs of each layer will be correlated.

vertible operations that are numerically stable [102, 104]. These restrictions have been shown to severely limit the flexibility of the inference process, and the complexity of the resulting posterior approximation [105].

As stated previously, in the VI framework, an approximate distribution is selected and the ELBO is then maximised. This ELBO arises from the applying the KL divergence between the true and approximate posterior, but this begs the question, why use the KL? The KL divergence is a well known measure to assess the similarity of between two distributions, and satisfies all the key properties of a divergence (ie. is positive and only zero when the two distributions are equal). A divergence allows us to know whether our approximation is approaching the true distribution, but not how close we are to it. Why not use of a well defined distance as appose to a divergence?

The KL divergence is used as it allows us to separate the intractable quantity (the marginal likelihood) out of our objective function (the ELBO) which we can optimise. Our goal with our Bayesian inference is to identify the parameters that best fit our model under prior knowledge and the distribution of the observed data. The VI framework poses inference as an optimisation problem, where we optimise our parameters to minimise the KL divergence between our approximate and true distribution (which maximises our ELBO). Since we are optimising our parameters, by separating the marginal likelihood from our objective function, we are able to compute derivatives with respect to the tractable quantities. Since the marginal likelihood is independent of the parameters, this component vanishes when the derivative is taken. This is the key reason why the KL divergence is used, as it allows us to separate the intractable quantity out of our objective function, which will then be evaluated as zero when using gradient information to perform optimisation.

The KL divergence has been shown to be part of a generic family of divergences known as α-divergences [106, 107]. The α-divergence is represented as,

$$D_\alpha[p(\omega)||q(\omega)] = \frac{1}{\alpha(1-\alpha)}\Big(1 - \int p(\omega)^\alpha q(\omega)^{1-\alpha} d\omega\Big). \tag{57}$$

The forward KL divergence used in VI is found from Equation 57 in the limit that $\alpha \to -1$, and the reverse KL divergence $\mathrm{KL}(p||q)$ occurs in the limit of $\alpha \to 1$, which is used during expectation propagation. While the use of the forward KL divergence used in VI typically results in an under-estimated variance, the use of the reverse KL will often over-estimate variance [2]. Similarly, the Hellinger distance arises from 57 when $\alpha = 0$,

$$D_H(p(\omega)||q(\omega))^2 = \int \Big(p(\omega)^{\frac{1}{2}} - q(\omega)^{\frac{1}{2}}\Big)^2 d\omega. \tag{58}$$

This is a valid distance, in that it satisfies the triangle inequality and is symmetric. Minimisation of the Hellinger distance has shown to provide reasonable compromise in variance estimate when compared with the two KL divergences [107]. Though these measures may provide desirable qualities,

they are not suitable for direct use within VI, as the intractable marginal likelihood cannot be separated from the other terms of interest[26]. While these measures cannot be immediately used, it illustrates how a change in the objective measure can result in different approximations. It is possible that more accurate posterior expectations can be found by utilising a different measure for the objective function.

The vast majority of modern works have revolved around the notion of VI. This is largely due to its amenability to SGD. Sophisticated tools now exist to simplify and accelerate the implementation of automatic differentiation and backpropagation [108, 109, 110, 111, 112, 113, 114]. Another benefit of VI is it's acceptance of sub-sampling in the likelihood. Sub-sampling reduces the computational expense for performing inference required to train over large data sets currently available. It is this key reason that more traditional MCMC based methods have received significantly less attention in the BNN community.

MCMC serves as the gold standard for performing Bayesian inference due to it's rich theoretical development, asymptotic guarantees and practical convergence diagnostics. Traditional MCMC based methods require sampling from the full joint likelihood to perform updates, requiring all training data to be seen before any new proposal can be made. Sub-sampling MCMC, or Stochastic Gradient MCMC (SG-MCMC) approaches have been proposed in [61, 115, 116], which have since been applied to BNNs [117]. It has since been shown that the naive sub-sampling within MCMC will bias the trajectory of the stochastic updates away from the posterior [118]. This bias removes the theoretical advantages gained from a traditional MCMC approach, making them less desirable than a VI approach which is often less computationally expensive. For sampling methods to become feasible, sub-sampling methods need to be developed that assure convergence to the posterior distribution.

3 Comparison of Modern BNNs

From the literature survey presented within, two prominent methods for approximate inference in BNNs was Bayes by Backprop [76] and MC Dropout [85]. These methods have found to be the most promising and highest impact methods for approximate inference in BNNs. These are both VI methods that are flexible enough to permit the use of SGD, making deployment to large and practical data sets feasible. Given their prominence, it is worthwhile to compare the methods to see how well they perorm.

To compare these methods, a series of simple homoskedastic regression tasks were conducted. For these regression models, the likelihood is repre-

[26] This may be easier to see for the Hellinger distance, but perhaps less so for the reverse KL divergence. Enthusiastic readers are encouraged to not take my word for it, and to put pen and paper to prove this for themselves!

sented as Gaussian. With this we can write that the un-normalised posterior is,

$$p(\boldsymbol{\omega}|\mathcal{D}) \propto p(\boldsymbol{\omega})\mathcal{N}(\mathbf{f}^{\boldsymbol{\omega}}(\mathcal{D}), \sigma^2\mathbf{I}), \tag{59}$$

where $\mathbf{f}^{\boldsymbol{\omega}}(\mathcal{D})$ is the function represented by the BNNs. A mixture of Gaussians was used to model a spike-slab prior for both models. The approximate posterior $q_{\boldsymbol{\theta}}(\boldsymbol{\omega})$ was then found for each model using the respective methods proposed. For Bayes by Backprop, the approximate posterior is a fully factorised Gaussian, and for MC Dropout is a scaled Bernoulli distribution. With the approximate posterior for each model, predictive quantities can be found using MC Integration. The first two moments can be approximated as [39],

$$\mathbb{E}_q[\mathbf{y}^*] \approx \frac{1}{N}\sum_{i=1}^{N} \mathbf{f}^{\boldsymbol{\omega}_i}(\mathbf{x}^*) \tag{60}$$

$$\mathbb{E}_q[\mathbf{y}^{*T}\mathbf{y}^*] \approx \sigma^2\mathbf{I} + \frac{1}{N}\sum_{i=1}^{N} \mathbf{f}^{\boldsymbol{\omega}_i}(\mathbf{x}^*)^T\mathbf{f}^{\boldsymbol{\omega}_i}(\mathbf{x}^*) \tag{61}$$

where the star superscript denotes the new input and output sample $\mathbf{x}^*, \mathbf{y}^*$ from the test set.

The data sets used to evaluate these models were simple toy data sets from high impact papers, where similar experimentation was provided as empirical evidence [76, 119]. Both BNN methods were then compared with a GP model. Figure 7 illustrates these results.

Analysis of the regression results shown in Figure 7 shows contrasting performance in terms of bias and variance in predictions. Models trained with Bayes by Backprop and a factorised Gaussian approximate posterior show reasonable predictive results withing the distribution of training data, though variance outside the region of training data is significantly under estimated when compared with the GP. MC Dropout with a scaled Bernoulli approximate posterior typically exhibits greater variance for out of distribution data, though maintains unnecessarily high variance within the distribution of training data. Little tuning of hyperparameters was done to these models. Better results may be achieved, particularly for MC Dropout, with better selection of hyperparameters. Alternatively, a more complete Bayesian approach can be used, where hyperparameters are treated as latent variables and marginalisation is performed over these variables.

It is worthwile noting the computational and practical difficulties encountered with these methods. The MC Dropout method is incredibly versitle, in that it was less sensitive to the choice of prior distribution. It also managed to fit to more complex distributions with fewer samples and training iterations. On top all this is the significant savings in computational resources. Given that training a model using MC Dropout is often identical to how many existing deep networks are trained, inference is performed in the same

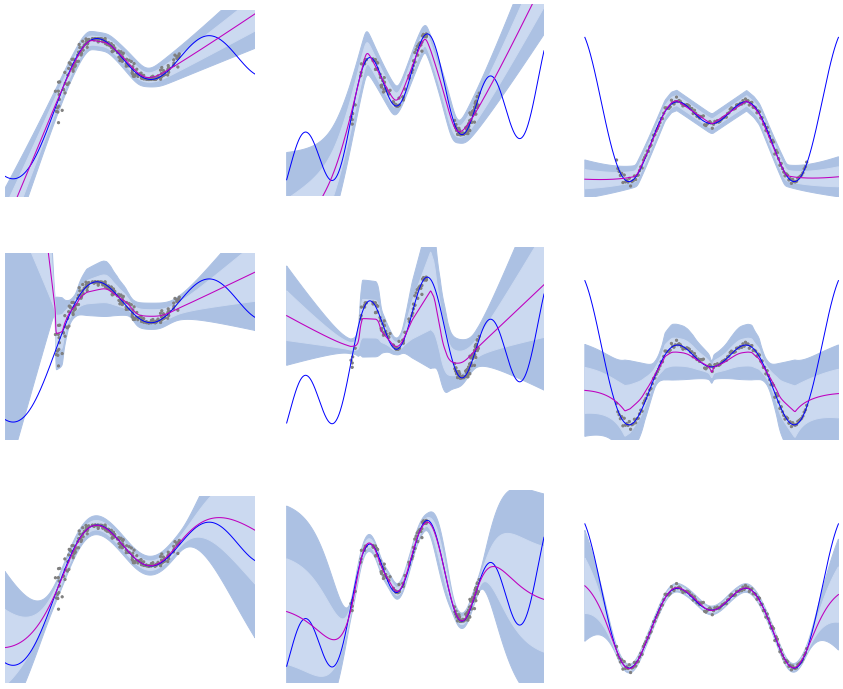


Fig. 7: Comparison of BNNs with GP for a regression task over three toy data sets. The top row is a BNN trained with Bayes By Backprop [76], the centre row is trained with MC dropout [39], and the bottom a GP with a Mattern52 kernel fitted with the GPflow package [120]. The two BNNs consisted of two hidden layers utilising ReLU activation. Training data is shown with the dark grey scatter, the mean is shown in purple, the true test function is shown in blue, and the shaded regions representing $\pm$ one and two std. from the mean. Best viewed on a computer screen.

time as traditional vanilla networks. It also offers no increase in the number of parameters to a network, where Bayes by Backprop requires twice as many. These factors should be taken into account for practical scenarios. If the data being modelled is smooth, is in sufficient quantity and additional time for inference is permitted, Bayes by Backprop may be preferable. For large networks with complex functions, sparse data and more stringent time requirements, MC Dropout may be more suitable.

3.1 Convolutional BNNs

Whilst the MLP serves as the basis for NNs, the most prominent NN architecture is the Convolutional Neural Network (CNN) [121]. These networks have excelled at challenging image classification tasks, with predictive performance far exceeding prior kernel based or feature engineered methods. A CNN differs from a typical MLP through it's application a convolution-like operator as oppose to inner products[27]. The output of a single convolutional layer can be expressed as,

$$\boldsymbol{\Phi} = u(\mathbf{X}^T * \mathbf{W}) \tag{62}$$

where $u(\cdot)$ is a non-linear activation and $*$ represents the convolution-like operation. Here the input $\mathbf{X}$ and the weight matrix $\mathbf{W}$ are no longer restricted to either vectors or matrices, and can instead be multi-dimensional arrays. It can be shown that CNNs can be written to have an equivalent MLP model, allowing for optimised linear algebra packages to be used for training with back-propagation [122].

Extending on the current research methods, a new type of Bayesian Convolutional Neural Network (BCNN) can be developed. This is achieved here by extending on the Bayes by Backprop method [76] to the case of models suitable for image classification. Each weight in the convolutional layers is assumed to be independent, allowing for factorisation over each individual parameter.

Experimentation was conducted to investigate the predictive performance of BCNNs, and the quality of their uncertainty estimates. These networks were configured for classification of the MNIST hand digit dataset [123].

Since this task is a classification task, the likelihood for the BCNN was set to a Softmax function,

$$\text{softmax}(\mathbf{f}_i^{\boldsymbol{\omega}}) = \frac{\mathbf{f}_i^{\boldsymbol{\omega}}(\mathcal{D})}{\sum_j \exp\left(\mathbf{f}_j^{\boldsymbol{\omega}}(\mathcal{D})\right)}. \tag{63}$$

The un-normalised posterior can then be represented as,

$$p(\boldsymbol{\omega}|\mathcal{D}) \propto p(\boldsymbol{\omega}) \times \text{softmax}(\mathbf{f}^{\boldsymbol{\omega}}(\mathcal{D})). \tag{64}$$

The approximate posterior is then found using Bayes by Backprop. Predictive mean for test samples can be found using Equation 60, and MC integration is used to approximate credible intervals [35].

Comparison with a vanilla CNN was made to evaluate the predictive performance of the BCNN. For both the vanilla and BCNN, the popular LeNet architecture [123] was used. Classification was conducted using the mean

[27] Emphasis is placed on "convolution like", as it is not equivalent to the mathematical operation of linear or circular convolution.

output of the BCNN, with credible intervals being used to assess the models uncertainty. Overall predictive performance for both networks on the 10,000 test images in the MNIST dataset showed comparative performance. The BCNN showed a test prediction accuracy of 98.99%, while the vanilla network showed a slight improvement with a prediction accuracy of 99.92%. Whilst the competitive predictive performance is essential, the main benefit of the BCNN is that we yield valuable information about the uncertainty of our predictions. Examples of difficult to classify digits are shown in the Appendix, accompanied by plots of the mean prediction and 95% credible intervals for each class. From these examples, we can see the large amount of predictive uncertainty for these challenging images, which could be used to make more informed decisions in practical scenarios.

This uncertainty information is invaluable for many scenarios of interest. As statistical models are increasingly employed for complex tasks containing human interaction, it is crucial that many of these systems make responsible decisions based on their perceived model of the world. For example, NNs are largely used within the development of autonomous vehicles. Development of autonomous vehicles is an incredibly challenging feat, due to the high degree of variability in scenarios and the complexity relating to human interaction. Current technologies are insufficient for safely enabling this task, and as discussed earlier, the use of these technologies have been involved in multiple deaths [24, 25]. It is not possible to model all variables within such a highly complex system. This accompanied by imperfect models and reliance on approximate inference, it is important that our models can communicate any uncertainty relating to decisions made. It is crucial that we acknowledge that in essence, our models are wrong. This is why probabilistic models are favoured for such scenarios; there is an underlying theory to help us deal with heterogeneity in our data and to account for uncertainty induced by variables not included in the model. It is vital that models used for such complex scenarios can communicate their uncertainty when used in such complex and high risk scenarios.

4 Conclusion

Throughout this report, the problems that arise with overconfident predictions from typical NNs and ad hoc model design have been illustrated. Bayesian analysis has been shown to provide a rich body of theory to address these challenges, though exact computation remains analytically and computationally intractable for any BNN of interest. In practice, approximate inference must be relied upon to yield accurate approximations to the posterior.

Many of the approximate methods for inference within BNNs have revolved around the MFVB approach. This provides a tractable lower bound to opti-

mise w.r.t variational parameters. These methods are attractive due to their relative ease of use, accuracy of predictive mean values and acceptable number of induced parameters. Despite this, it was shown through the literature survey and experimentation results that the assumptions made within a fully factorised MFVB approach result in over-confident predictions. It was shown that these MFVB approaches can be extended upon to more complex models such as CNNs. Experimental results indicate comparable predictive performance to point estimate CNNs for image classification tasks. The Bayesian CNN was able to provide credible intervals on the predictions, which were found to be highly informative and intuitive measure of uncertainty for difficult to classify data points.

This review and these experiments highlight the capabilities of Bayesian analysis to address common challenges seen in the machine learning community. These results also highlight how current approximate inference methods for BNNs are insufficient and can provide inaccurate variance information. Additional research is required to not only determine how these networks operate, but how accurate inference can be achieved with modern large networks. Methods to scale exact inference methods such as MCMC to large data sets would allow for a more principled method of performing inference. MCMC offers diagnostic methods to assess convergence and quality of inference. Similar diagnostics for VI would allow researchers and practitioners to evaluate the quality of their assumed posterior, and inform them with ways to improve on this assumption. Achieving these goals will allow us to obtain accurate posterior approximations. From this we will be able to sufficiently determine what our models know, but also what they don't know.

References

1. F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological Review*, vol. 65, no. 6, pp. 386 – 408, 1958.
2. C. Bishop, *Pattern recognition and machine learning*. New York: Springer, 2006.
3. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, p. 533, 1986.
4. K.-S. Oh and K. Jung, "Gpu implementation of neural networks," *Pattern Recognition*, vol. 37, no. 6, pp. 1311–1314, 2004.
5. D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep big simple neural nets excel on handwritten digit recognition," *CoRR*, 2010.
6. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
7. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, 2014.
8. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich *et al.*, "Going deeper with convolutions," in *CVPR*, 2015.
9. R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

10. S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.
11. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
12. A. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.
13. G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on audio, speech, and language processing*, vol. 20, no. 1, pp. 30–42, 2012.
14. G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.
15. D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, J. Chen, J. Chen, Z. Chen, M. Chrzanowski, A. Coates, G. Diamos, K. Ding, N. Du, E. Elsen, J. Engel, W. Fang, L. Fan, C. Fougner, L. Gao, C. Gong, A. Hannun, T. Han, L. Johannes, B. Jiang, C. Ju, B. Jun, P. LeGresley, L. Lin, J. Liu, Y. Liu, W. Li, X. Li, D. Ma, S. Narang, A. Ng, S. Ozair, Y. Peng, R. Prenger, S. Qian, Z. Quan, J. Raiman, V. Rao, S. Satheesh, D. Seetapun, S. Sengupta, K. Srinet, A. Sriram, H. Tang, L. Tang, C. Wang, J. Wang, K. Wang, Y. Wang, Z. Wang, Z. Wang, S. Wu, L. Wei, B. Xiao, W. Xie, Y. Xie, D. Yogatama, B. Yuan, J. Zhan, and Z. Zhu, "Deep speech 2 : End-to-end speech recognition in english and mandarin," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 173–182.
16. D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.
17. "Smartening up with artificial intelligence (ai) - what's in it for germany and its industrial sector?" McKinsey & Company, Inc, Tech. Rep., 4 2017. [Online]. Available: https://www.mckinsey.de/files/170419_mckinsey_ki_final_m.pdf
18. E. V. T. V. Serooskerken, "Artificial intelligence in wealth and asset management," Pictet on Robot Advisors, Tech. Rep., 1 2017. [Online]. Available: https://perspectives.pictet.com/wp-content/uploads/2016/12/Edgar-van-Tuyll-van-Serooskerken-Pictet-Report-winter-2016-2.pdf
19. A. van den Oord, T. Walters, and T. Strohman, "Wavenet launches in the google assistant." [Online]. Available: https://deepmind.com/blog/wavenet-launches-google-assistant/
20. Siri Team, "Deep learning for siri's voice: On-device deep mixture density networks for hybrid unit selection synthesis," 8 2017. [Online]. Available: https://machinelearning.apple.com/2017/08/06/siri-voices.html
21. J. Wakefield, "Microsoft chatbot is taught to swear on twitter." [Online]. Available: www.bbc.com/news/technology-35890188
22. J. Guynn, "Google photos labeled black people 'gorillas'." [Online]. Available: https://www.usatoday.com/story/tech/2015/07/01/google-apologizes-after-photos-identify-black-people-as-gorillas/29567465/
23. J. Buolamwini and T. Gebru, "Gender shades: Intersectional accuracy disparities in commercial gender classification," in *Conference on fairness, accountability and transparency*, 2018, pp. 77–91.
24. Tesla Team, "A tragic loss." [Online]. Available: https://www.tesla.com/en_GB/blog/tragic-loss

25. ABC News, "Uber suspends self-driving car tests after vehicle hits and kills woman crossing the street in arizona," 2018. [Online]. Available: http://www.abc.net.au/news/2018-03-20/uber-suspends-self-driving-car-tests-after-fatal-crash/9565586
26. Council of European Union, "Regulation (eu) 2016/679 of the european parliment and of the council," 2016.
27. B. Goodman and S. Flaxman, "European union regulations on algorithmic decision-making and a right to explanation," *AI magazine*, vol. 38, no. 3, pp. 50–57, 2017.
28. M. Vu, T. Adali, D. Ba, G. Buzsaki, D. Carlson, K. Heller, C. Liston, C. Rudin, V. Sohal, A. Widge, H. Mayberg, G. Sapiro, and K. Dzirasa, "A shared vision for machine learning in neuroscience," *JOURNAL OF NEUROSCIENCE*, vol. 38, no. 7, pp. 1601–1607, 2018.
29. A. Holzinger, C. Biemann, C. S. Pattichis, and D. B. Kell, "What do we need to build explainable ai systems for the medical domain?" *arXiv preprint arXiv:1712.09923*, 2017.
30. R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad, "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1721–1730.
31. D. Gunning, "Explainable artificial intelligence (xai)," *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2017.
32. D. J. MacKay, "Probable networks and plausible predictionsa review of practical bayesian methods for supervised neural networks," *Network: computation in neural systems*, vol. 6, no. 3, pp. 469–505, 1995.
33. J. Lampinen and A. Vehtari, "Bayesian approach for neural networksreview and case studies," *Neural networks*, vol. 14, no. 3, pp. 257–274, 2001.
34. H. Wang and D.-Y. Yeung, "Towards bayesian deep learning: A survey," *arXiv preprint arXiv:1604.01662*, 2016.
35. K. Murphey, *Machine learning, a probabilistic perspective*. Cambridge, MA: MIT Press, 2012.
36. X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *AISTATS*, 2011, pp. 315–323.
37. A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *ICML*, vol. 30, 2013, p. 3.
38. R. M. Neal, *Bayesian learning for neural networks*. Springer Science & Business Media, 1996, vol. 118.
39. Y. Gal, "Uncertainty in deep learning," *University of Cambridge*, 2016.
40. N. Tishby, E. Levin, and S. A. Solla, "Consistent inference of probabilities in layered networks: predictions and generalizations," in *International 1989 Joint Conference on Neural Networks*, 1989, pp. 403–409 vol.2.
41. J. S. Denker and Y. Lecun, "Transforming neural-net output levels to probability distributions," in *NeurIPS*, 1991, pp. 853–859.
42. G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.
43. K.-I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural networks*, vol. 2, no. 3, pp. 183–192, 1989.
44. K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.
45. S. F. Gull and J. Skilling, "Quantified maximum entropy memsys5 users manual," *Maximum Entropy Data Consultants Ltd*, vol. 33, 1991.
46. D. J. MacKay, "Bayesian interpolation," *Neural computation*, vol. 4, no. 3, pp. 415–447, 1992.
47. ——, "Bayesian methods for adaptive models," Ph.D. dissertation, California Institute of Technology, 1992.
48. ——, "A practical bayesian framework for backpropagation networks," *Neural computation*, vol. 4, no. 3, pp. 448–472, 1992.

49. M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Machine learning*, vol. 37, no. 2, pp. 183–233, 1999.
50. M. J. Wainwright, M. I. Jordan *et al.*, "Graphical models, exponential families, and variational inference," *Foundations and Trends® in Machine Learning*, vol. 1, no. 1–2, pp. 1–305, 2008.
51. D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.
52. M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 1303–1347, 2013.
53. D. Barber and C. M. Bishop, "Ensemble learning in bayesian neural networks," *NATO ASI SERIES F COMPUTER AND SYSTEMS SCIENCES*, vol. 168, pp. 215–238, 1998.
54. G. E. Hinton and D. 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*. ACM, 1993, pp. 5–13.
55. M. Betancourt, "A conceptual introduction to hamiltonian monte carlo," *arXiv preprint arXiv:1701.02434*, 2017.
56. M. Betancourt, S. Byrne, S. Livingstone, M. Girolami *et al.*, "The geometric foundations of hamiltonian monte carlo," *Bernoulli*, vol. 23, no. 4A, pp. 2257–2298, 2017.
57. G. Madey, X. Xiang, S. E. Cabaniss, and Y. Huang, "Agent-based scientific simulation," *Computing in Science & Engineering*, vol. 2, no. 01, pp. 22–29, jan 2005.
58. S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid monte carlo," *Physics letters B*, vol. 195, no. 2, pp. 216–222, 1987.
59. R. M. Neal *et al.*, "Mcmc using hamiltonian dynamics," *Handbook of markov chain monte carlo*, vol. 2, no. 11, p. 2, 2011.
60. S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, *Handbook of markov chain monte carlo*. CRC press, 2011.
61. M. Welling and Y. Teh, "Bayesian learning via stochastic gradient langevin dynamics," *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pp. 681–688, 2011.
62. A. Graves, "Practical variational inference for neural networks," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2011, pp. 2348–2356.
63. M. Opper and C. Archambeau, "The variational gaussian approximation revisited," *Neural computation*, vol. 21, no. 3, pp. 786–792, 2009.
64. J. M. Hernández-Lobato and R. Adams, "Probabilistic backpropagation for scalable learning of bayesian neural networks," in *International Conference on Machine Learning*, 2015, pp. 1861–1869.
65. J. Paisley, D. Blei, and M. Jordan, "Variational bayesian inference with stochastic search," *arXiv preprint arXiv:1206.6430*, 2012.
66. J. R. Wilson, "Variance reduction techniques for digital simulation," *American Journal of Mathematical and Management Sciences*, vol. 4, no. 3-4, pp. 277–312, 1984.
67. M. Opper and C. Archambeau, "The variational gaussian approximation revisited," *Neural computation*, vol. 21 3, pp. 786–92, 2009.
68. D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.
69. D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *Proceedings of the 31st International Conference on Machine Learning (ICML)*, 2014, pp. 1278–1286.
70. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

71. D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Advances in Neural Information Processing Systems*, 2015, pp. 2575–2583.
72. S. Wang and C. Manning, "Fast dropout training," in *international conference on machine learning*, 2013, pp. 118–126.
73. A. Livnat, C. Papadimitriou, N. Pippenger, and M. W. Feldman, "Sex, mixability, and modularity," *Proceedings of the National Academy of Sciences*, vol. 107, no. 4, pp. 1452–1457, 2010.
74. M. Opper and O. Winther, "A bayesian approach to on-line learning," *On-line learning in neural networks*, pp. 363–378, 1998.
75. T. P. Minka, "A family of algorithms for approximate bayesian inference," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.
76. C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," *arXiv preprint arXiv:1505.05424*, 2015.
77. C. K. Williams, "Computing with infinite networks," in *Advances in neural information processing systems*, 1997, pp. 295–301.
78. J. Lee, J. Sohl-dickstein, J. Pennington, R. Novak, S. Schoenholz, and Y. Bahri, "Deep neural networks as gaussian processes," in *International Conference on Learning Representations*, 2018.
79. A. Damianou and N. Lawrence, "Deep gaussian processes," in *AISTATS*, 2013, pp. 207–215.
80. A. Damianou, "Deep gaussian processes and variational propagation of uncertainty," Ph.D. dissertation, University of Sheffield, 2015.
81. N. Lawrence, "Deep gaussian processes," 2019. [Online]. Available: http://inverseprobability.com/talks/notes/deep-gaussian-processes.html
82. A. Damianou, M. K. Titsias, and N. D. Lawrence, "Variational gaussian process dynamical systems," in *NeurIPS*, 2011, pp. 2510–2518.
83. M. Titsias, "Variational learning of inducing variables in sparse gaussian processes," in *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, D. van Dyk and M. Welling, Eds., vol. 5. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 16–18 Apr 2009, pp. 567–574.
84. Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Insights and applications," in *Deep Learning Workshop, ICML*, vol. 1, 2015, p. 2.
85. ——, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *ICML*, 2016, pp. 1050–1059.
86. ——, "Dropout as a bayesian approximation: Appendix," *arXiv preprint arXiv:1506.02157*, 2015.
87. A. Garriga-Alonso, L. Aitchison, and C. E. Rasmussen, "Deep convolutional networks as shallow gaussian processes," *arXiv preprint arXiv:1808.05587*, 2018.
88. R. Novak, L. Xiao, Y. Bahri, J. Lee, G. Yang, D. A. Abolafia, J. Pennington, and J. Sohl-dickstein, "Bayesian deep convolutional networks with many channels are gaussian processes," in *International Conference on Learning Representations*, 2019.
89. M. Van der Wilk, C. E. Rasmussen, and J. Hensman, "Convolutional gaussian processes," in *Advances in Neural Information Processing Systems*, 2017, pp. 2849–2858.
90. D. J. MacKay and D. J. Mac Kay, *Information theory, inference and learning algorithms*. Cambridge university press, 2003.
91. B. Wang and D. Titterington, "Inadequacy of interval estimates corresponding to variational bayesian approximations." in *AISTATS*. Barbados, 2005.
92. R. E. Turner and M. Sahani, "Two problems with variational expectation maximisation for time-series models," in *Bayesian Time Series Models*, D. Barber, A. T. Cemgil, and S. Chiappa, Eds. Cambridge University Press, 2011.
93. R. Giordano, T. Broderick, and M. I. Jordan, "Covariances, robustness, and variational bayes," *Journal of Machine Learning Research*, vol. 19, no. 51, pp. 1–49, 2018.

94. D. Hafner, D. Tran, A. Irpan, T. Lillicrap, and J. Davidson, "Reliable uncertainty estimates in deep neural networks using noise contrastive priors," *arXiv preprint arXiv:1807.09289*, 2018.
95. V. Kuleshov, N. Fenner, and S. Ermon, "Accurate uncertainties for deep learning using calibrated regression," *arXiv preprint arXiv:1807.00263*, 2018.
96. Y. Gal, J. Hron, and A. Kendall, "Concrete dropout," in *Advances in Neural Information Processing Systems*, 2017, pp. 3581–3590.
97. C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," *arXiv preprint arXiv:1611.00712*, 2016.
98. T. S. Jaakkola and M. I. Jordan, "Improving the mean field approximation via the use of mixture distributions," in *Learning in graphical models*. Springer, 1998, pp. 163–173.
99. C. Louizos and M. Welling, "Structured and efficient variational deep learning with matrix gaussian posteriors," in *International Conference on Machine Learning*, 2016, pp. 1708–1716.
100. E. G. Tabak and E. Vanden-Eijnden, "Density estimation by dual ascent of the log-likelihood," *Commun. Math. Sci.*, vol. 8, no. 1, pp. 217–233, 03 2010.
101. E. G. Tabak and C. V. Turner, "A family of nonparametric density estimation algorithms," *Communications on Pure and Applied Mathematics*, vol. 66, no. 2, pp. 145–164, 2013.
102. D. J. Rezende and S. Mohamed, "Variational inference with normalizing flows," *arXiv preprint arXiv:1505.05770*, 2015.
103. C. Louizos and M. Welling, "Multiplicative normalizing flows for variational bayesian neural networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, pp. 2218–2227.
104. L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real NVP," *CoRR*, vol. abs/1605.08803, 2016.
105. C. Cremer, X. Li, and D. K. Duvenaud, "Inference suboptimality in variational autoencoders," *CoRR*, vol. abs/1801.03558, 2018.
106. S.-i. Amari, *Differential-geometrical methods in statistics*. Springer Science & Business Media, 2012, vol. 28.
107. T. Minka *et al.*, "Divergence measures and message passing," Technical report, Microsoft Research, Tech. Rep., 2005.
108. Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.
109. F. Chollet, "keras," https://github.com/fchollet/keras, 2015.
110. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *CoRR*, vol. abs/1603.04467, 2016.
111. J. V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. D. Hoffman, and R. A. Saurous, "Tensorflow distributions," *CoRR*, vol. abs/1711.10604, 2017.
112. P. Adam, G. Sam, C. Soumith, C. Gregory, Y. Edward, D. Zachary, L. Zeming, D. Alban, A. Luca, and L. Adam, "Automatic differentiation in pytorch," in *Proceedings of Neural Information Processing Systems*, 2017.
113. T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," *CoRR*, vol. abs/1512.01274, 2015.

114. A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei, "Automatic Differentiation Variational Inference," *arXiv e-prints*, p. arXiv:1603.00788, Mar 2016.
115. S. Patterson and Y. W. Teh, "Stochastic gradient riemannian langevin dynamics on the probability simplex," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3102–3110.
116. T. Chen, E. Fox, and C. Guestrin, "Stochastic gradient hamiltonian monte carlo," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. P. Xing and T. Jebara, Eds., vol. 32. PMLR, 22–24 Jun 2014, pp. 1683–1691.
117. C. Li, C. Chen, D. Carlson, and L. Carin, "Preconditioned Stochastic Gradient Langevin Dynamics for Deep Neural Networks," *arXiv e-prints*, Dec. 2015.
118. M. Betancourt, "The fundamental incompatibility of scalable hamiltonian monte carlo and naive data subsampling," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, pp. 533–540.
119. I. Osband, C. Blundell, A. Pritzel, and B. V. Roy, "Deep exploration via bootstrapped DQN," *CoRR*, vol. abs/1602.04621, 2016.
120. A. G. d. G. Matthews, M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman, "GPflow: A Gaussian process library using TensorFlow," *Journal of Machine Learning Research*, vol. 18, no. 40, pp. 1–6, 4 2017.
121. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.
122. I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.
123. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

Appendix

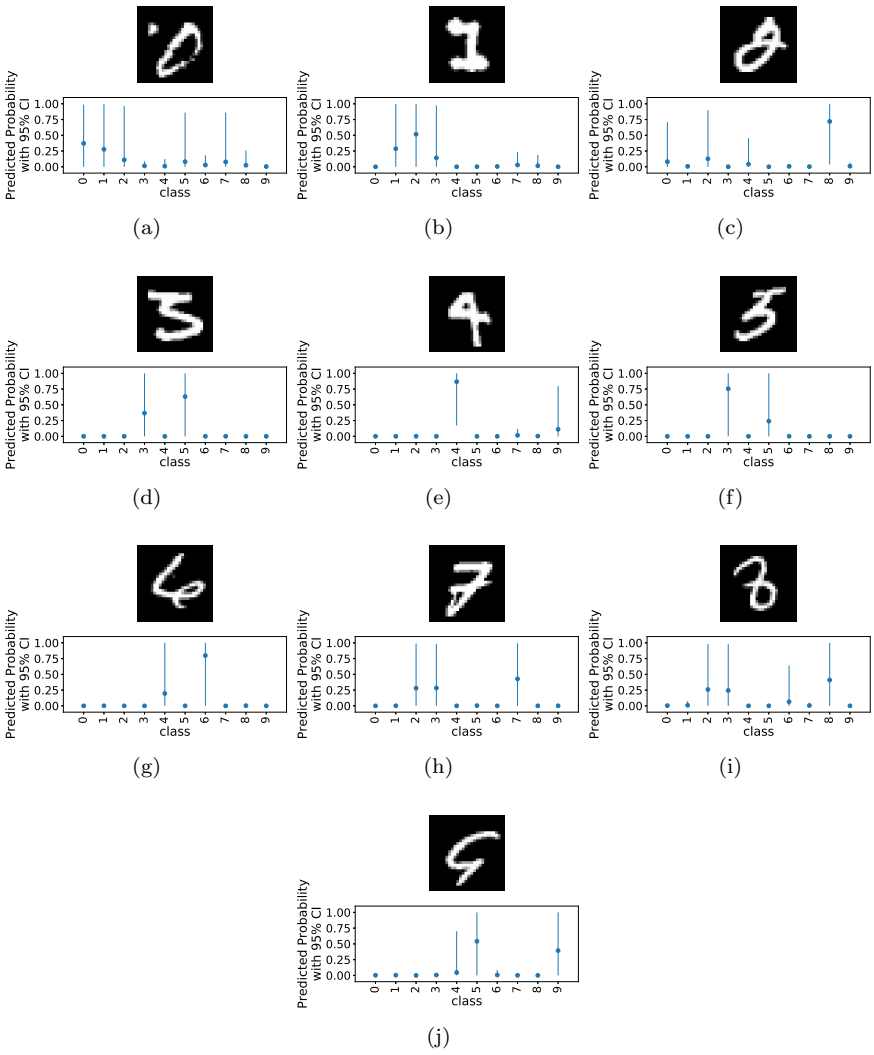


Fig. 8: Examples of difficult to classify images from each class in MNIST. True class for each image is 0-9 arranged in alphabetical order. The bottom plot illustrates the 95% credible intervals for these predictions. Best viewed on a computer screen.