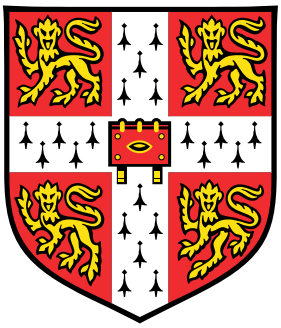
Investigating Inference in Bayesian Neural Networks via Active Learning



Riccardo Barbano

Supervisors:
Jonathan Gordon, Robert Pinsler,
Dr José Miguel Hernández-Lobato

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy in Machine Learning and Machine Intelligence

St Edmund's College August 2019

I would like to dedicate this work to Anna and Francesco, and their everlasting jazzy attitude that will keep brightening up my walks ...

Acknowledgement

First and foremost, I would like to express my deepest gratitude to my supervisors Jonathan Gordon and Robert Pinsler for their peerless guidance.
I would also like to thank the MLMI community, theatre of invaluable friendships.
Ultimately, I would like to thank my family for their unconditional support.

Declaration

I, Riccardo Barbano of St. Edmund's College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for comparable purpose.
This dissertation contains 13578 words excluding bibliography, photographs and diagrams but including tables, captions, footnotes, appendices, and abstract.

Existing Software: The project is implemented in Python and PyTorch. This project builds upon the code provided by my supervisors.

Riccardo Barbano
August 2019

Abstract

Bayesian deep learning is an active area of study where approximate Bayesian inference is applied to neural architectures. In the past decade, an abundance of inferential methods have been proposed; none of which have been tested on rigorous nor tailored benchmarks of statistical modelling.

To address the lack of strong baselines, we investigate the suitability of active learning to disambiguate the comparison among inferential methods on the grounds that information-based sequential decision-making scenarios make better use of that for which Bayesian neural networks are known.

In this work, by the means of an active learning framework we compare several approximate inference methods, such as variational inference (e.g. mean field variational inference and variational matrix Gaussian), Markov Chain Monte Carlo methods, and MC dropout, alongside architectures with different degrees of stochasticity.

We exemplify that the effects of the posterior approximation are better indicated by their usage in a sequential decision-making scenario.

Contents

List of Figures

List of Tables

Nomenclature

AI Artificial Intelligence

AL Active Learning

BALD Bayesian Active Learning with Disagreement

BBB Bayes By Backpropagation

BNN Bayesian Neural Network

CNN Convolutional Neural Network

ECE Expected Calibration Error

ELBO Evidence Lower Bound

fVI functional Variational Inference

HMC Hamiltonian Monte Carlo

KL Kullback-Leibler Divergence

MAP Maximum a Posteriori

MC Monte Carlo

MCMC Markov Chain Monte Carlo

MFVI Mean Field Variational Inference

MLE Maximum Likelihood Estimation

pSGLD preconditioned Stochastic Langevin Dynamics

SG-MCMC Stochastic Gradient Markov Chain Monte Carlo

SGD Stochastic Gradient Descent/Ascent

SGLD Stochastic Gradient Langevin Dynamics

SGVB Stochastic Gradient Variational Bayes

SRT Stochastic Regularisation Technique

VFE Variational Free Energy

VI Variational Inference

VMG Variational Matrix Gaussian

1 Introduction: On Active Learning as a Strong Baseline

Motivation

Bayesian deep learning is a fairly active field of study where approximate Bayesian inference is applied to neural architectures. Building upon David MacKay's and Neal Radford's pioneering works, the machine learning community took up the idea of riding the wave of the unrivalled deep learning revolution with great enthusiasm. Combining neural networks – a ubiquitous presence in every research sub-field that branches off machine learning – with a principled way of managing uncertainty seemed an attractive idea.

In the past decade, Bayesian deep learning became a "hyperactive" area in machine learning research. Year after year, new approximate inference methods, as a result of either substantial or minor variations of well-established ones, are discussed across conferences and symposia. On the other hand, new proposals are validated empirically, relying either on machine learners' good faith or on the good work of reviewers to note unintentional discrepancies in the experimental settings. However, debugging machine learning code is a tedious task and flaws can easily slip by the author's attention. Recently, the inadequately empirical approach, which is at the core of any evaluation in machine learning, has raised some concerns among the community. The importance of using identical experimental setups within research groups to assess and contrast the quality of the Bayesian approximate methods has been reiterated.

The scarcity of rigorous experimental setups impedes progress in this area, and not only this area. Other machine learning sub-fields have suffered from setbacks due to the lack of appropriate testing. Among the language modelling community, years of presumed results did not actually take place as the standard stack LSTM models were not being tuned properly, and it turned out the performance of the new versions showed no improvement upon the old (Melis et al., 2017).

With the intention of addressing this, some recent works have started to adopt as a common testing ground a popular regression experiment on nine UCI datasets, as well as more meaningful evaluation metrics such as the predictive log-likelihood, along with standard ones (Hernández-Lobato & Adams, 2015; Gal & Ghahramani, 2016). However, there continue to be differences in the experimental settings, leading to ambiguity and confusion. Nowadays, the praxis used for evaluating and comparing approximate inference methods is via supervised learning, where the system passively receives the data to be trained on. By doing so, we do not leverage what Bayesian neural networks are really known for: modelling uncertainty! Bayesian neural networks' most powerful characteristic is not exploited at all, and their ability to interact and influence the world by knowing what they do not know, is not even considered.

Aims & Scope

This work aims to address the scarcity of robust and rigorous baselines, and to introduce testing grounds where the performance of the model is a proxy of the quality of its uncertainty estimates. We propose the active learning framework as a more rigorous testing ground for the evaluation of the posterior approximation. We believe that a closed-loop sequential decision-making environment, in which the model by generating queries selects the most unfamiliar data-points to be added to its training dataset, is worth greater attention. Active learning application to uncertainty evaluation lies on the grounds that the informativeness of a data-point goes hand-in-hand with how the model deals with uncertainty from an information theoretic standpoint.

This work is inspired by a remarkable attempt by Riquelme et al. (2018). They compare several Bayesian inference methods in a Thompson Sampling framework over a series of contextual bandit problems. They show that many approaches that are successful in a supervised setting (e.g. supervised learning on MNIST) underperform in decision-making scenarios, hence resulting in sub-optimal exploration. Their analysis paved the way for using sequential decision-making scenarios as a meaningful ground for comparing Bayesian inference approximations. Sharing similar intentions and views, we expand on a different decision-making framework.

1.1 Thesis Contribution

The main contribution of this work is to show active learning's adequacy when it comes to comparing different approximate inference methods. We propose the use of active learning as a "Bayesian toolbox" to gain more meaningful insights into the effect of the posterior approximation, and ultimately on the quality of uncertainty estimation. We compare different approximate inference methods: we consider several variational inference algorithms as well as Monte Carlo dropout, a practical Bayesian approximate inference method, and Stochastic Gradient Langevin Dynamics, one of the most famous Stochastic Gradient Markov Chain Monte Carlo implementation. Our investigation walks on a double track: we compare several ways of approximating Bayesian inference in neural networks and apply these approximations to different neural architectures (e.g. fully stochastic fully connected models, hybrid models where stochastic and deterministic components coexist, etc.). The common thread is active learning. Through active learning, we define an experimental protocol that is identical for all our inference methods; we draw comparisons among methods on a level playing field. We do not limit our analysis to one dataset; instead, we investigate multiple ones with different degrees of complexity, and we base our findings on a combined analysis. We introduce a new testing bench for Bayesian neural networks, and present it as an alternative to passive learning on MNIST[1].
In sum, this work shows that posterior approximations are better indicated by their usage in sequential decision-making (e.g. querying for active learning). Via active learning, we ease the comparison among different approximate inference methods.

Thesis Outline

This work is structured as it follows. In §2, we review the basics of Bayesian inference, and Bayesian neural networks. From a theoretical standpoint, we discuss the Bayesian approximate methods we implement, and ultimately contrast. In §3, we assess our implementations by testing them on MNIST. In §4, we present active learning, and discuss querying strategies, along with their interpretation of uncertainty and their application to Bayesian neural networks. We elaborate on the fact that active learning and Bayesian deep learning fit well together. In §5, we move to a more in-depth investigation of approximate inference methods by the means of active learning. In §6, we draw our conclusions, and comment on future research directions.

[1] In §3, we define what we mean by passive learning.

2 Bayesian Neural Networks

A Brief Historical Excursus

Initial attempts to integrate flexibility, scalability, and predictive performance – all built-in characteristics of neural networks – with principled Bayesian modelling of uncertainty resulted in approaches strictly limited to small architectures (MacKay, 1992; Hinton & Van Camp, 1993). Further attempts to scale up to deeper Bayesian neural networks (BNNs) took advantage of restrictive approximations of the posterior distribution. Assumptions of independence between weights are often made (Graves, 2011; Blundell et al.,2015). Variational inference (VI) – a popular technique that recasts intractable Bayesian integration as an optimisation problem – is first applied to neural networks by Hinton & Van Camp (1993). Almost two decades later, Graves (2011) proposed a practical approach with fully factorised Gaussian posteriors, which implemented a simple but biased gradient estimator. Building upon that, Blundell et al. (2015) introduced an unbiased gradient estimator, leveraging on the generalised reparametrization trick presented by Kingma & Welling (2013). To reduce the variance during training, the local reparametrization trick is introduced (Kingma et al., 2015). More recent works focus on modelling correlations between weights to capture the posterior dependencies (Louizos & Welling, 2016; Sun et al., 2017; Bae et al., 2018). By turning to a different route, Neal (2012) came up with an alternative approximate inference method based on Hamiltonian dynamics – it will be named Hamiltonian Monte Carlo (HMC). His work is considerably picked-up and extended-on (Chen et al., 2014), and simplifications of the initial scheme (quite computationally costly) are suggested (Welling & Teh, 2011). In 2015, Gal & Ghahramani proposed Monte Carlo dropout as a Bayesian approximate inference method. More experimental research examines new frontiers such as noisy natural gradient, scalable Laplace approximation, or functional variational inference (fVI) (Zhang et al., 2017; Ritter et al., 2018; Sun et al., 2019).

Chapter Outline

This chapter aims to provide the reader with an overview on the theoretical framework of approximate Bayesian inference. Throughout this work, well-established approximate inference schemes are discussed thoroughly. In §2.1, we briefly cover the basics behind learning with uncertainty estimates. In §2.2, we review the probabilistic interpretation of neural networks, along with the theoretical motivation for approximate inference. In §2.3, we discuss approximate inference methods by identifying their strengths and weaknesses, and by outlining the trades-offs between the expressiveness of the posterior approximation and the scalability of the method. In §2.4, we present variational inference methods. We discuss a mean field Gaussian approximation to the posterior distribution (also known as *Bayes By Backpropagation*), along with learning more structured matrix variate Gaussian posteriors. In §2.5, we review Monte Carlo dropout. In §2.6, we discuss stochastic gradient methods, such as Stochastic Gradient Langevin Dynamics (SGLD).

2.1 Bayesian Modelling

Back to the "Past"

Back in the 18$^{\text{th}}$ century, Reverend Thomas Bayes (1702-1761) showed us how to do inference about hypothesis (i.e. uncertain quantities) from data (i.e. measured quantities).

Bayes' rule (in its most straightforward vest) is formulated as:

$$P(\text{hypothesis}|\text{data}) = \frac{P(\text{data}|\text{hypothesis})P(\text{hypothesis})}{P(\text{data})}$$

This process is called *inference*. But, what does it mean to be Bayesian within a machine learning framework? Firstly, being Bayesian in machine learning means dealing with parameters uncertainty. This allows us to obtain distributions of "answers" to a given question rather than just point estimates.
Given the training inputs $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$[1] and the corresponding outputs $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$[2], we define a likelihood distribution $p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega})$ as the probabilistic model that describes the outputs given the inputs under some parameter settings $\boldsymbol{\omega} \in \Omega$, and we place a prior distribution over the parameter space $p(\boldsymbol{\omega})$. By invoking Bayes' rule, we compute the posterior distribution over the parameter space,

$$p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}) = \frac{p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega})p(\boldsymbol{\omega})}{p(\mathbf{Y}|\mathbf{X})}$$

The posterior distribution indicates the most probable model parameters given the data. The normalisation constant is computed as:

$$p(\mathbf{Y}|\mathbf{X}) = \int_{\Omega} p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega})p(\boldsymbol{\omega})d\boldsymbol{\omega}$$

This is the (conditional) marginal likelihood, also referred to as *model evidence*. To make predictions for a new text input $\mathbf{x}^*$, we obtain the predictive posterior probabilities by integration (i.e. by averaging over all the possible parameters' configurations). This is expressed as:

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int_{\Omega} p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega})p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})d\boldsymbol{\omega}$$

The predictive posterior gives the predictive probability of a test input conditioned on the observables $\mathbf{X}, \mathbf{Y}$.

The Elephant in the Room ...

As a final remark, Bayesian methods heavily rely on integration. We integrate to compute the model evidence; we integrate over the parameter space to evaluate the output distribution. However, the elephant in the room is that the model evidence is analytically intractable for neural architectures. Hence, posterior and posterior predictive distributions are intractable too.
Later in the chapter, we review methods for practical approximate inference in BNNs.

[1] Throughout this work, we try to be consistent with the following notation: vectors are in bold lower-case letters ($\mathbf{x}$), matrices in bold upper-case letters ($\mathbf{X}$), and scalars in standard letters (x). Finally, we use $\boldsymbol{\omega}$ to denote a set of variables (e.g. $\boldsymbol{\omega} = \{\mathbf{W}_1\}_{i=1}^{L}$).

[2] We review the Bayesian analysis of a dataset $\mathbf{X}, \mathbf{Y}$ containing N i.i.d. instances.

2.2 Knowing What We Do Not Know

"Scale Well, Work Well"

In past decades, neural models have been extensively applied to a wide range of AI fields such as computer vision (Rowley, 1999), reinforcement learning (Mnih et al., 2013; Mnih et al., 2015), speech and text recognition (Bengio et al., 2003), speech synthesis (Oord et al., 2016), chemical and molecular modelling (Wei et al., 2016), and many others. Undoubtedly, neural networks popularity comes from the fact that neural architectures *scale well* and *work well* on several grounds. Neural networks are hierarchical and modular learning machines, whose fundamental units are called neurons (commonly referred to as nodes). Neurons are arranged in layers, and layers are the building blocks of the network. By increasing the depth (i.e. stacking layer after layer) neural networks progressively improve their representational power; this allows us to model more complex and abstract data features.

A Single Hidden Layer Neural Network

We review a single hidden layer fully connected network, as it is a prerequisite for future discussions. Our input $\mathbf{x}$ to the network is a vector with Q elements, and we transform it with a linear map to a K elements vector. The weight matrix $\mathbf{W}_1$ (i.e. a linear map) and the bias vector $\mathbf{b}_1$ (i.e. a translation) operate the affine linear transformation. A non-linear differentiable activation function $\sigma(\cdot)$ (such as ReLU[3] (Nair & Hinton, 2010) or TanH) is then applied to $\mathbf{x}\mathbf{W}_1 + \mathbf{b}$. Accordingly, the network output $\hat{\mathbf{y}}$ is obtained by means of a second linear transformation $\mathbf{W}_2$ that connects the hidden layer to the model output,

$$\hat{\mathbf{y}} = \sigma(\mathbf{x}\mathbf{W}_1 + \mathbf{b})\mathbf{W}_2$$

where $\hat{\mathbf{y}}$ is a vector of C elements. Therefore, $\mathbf{W}_1$ is a $Q \times K$ matrix, $\mathbf{W}_2$ is a $K \times C$ matrix, and $\mathbf{b}$ is a K dimensional vector. $\mathbf{W}_1$, $\mathbf{W}_2$, and $\mathbf{b}$ are the learnable parameters in our network. We can easily generalise to L layer network by treating each layer's output $\mathbf{f}_i^{\mathbf{W}_i}(\cdot)$ as a non-linear mapping. The output of the network is:

$$\hat{\mathbf{y}} = \mathbf{f}_L^{\mathbf{W}_L}(\dots \mathbf{f}_1^{\mathbf{W}_1}(\mathbf{x}))$$

where each network's weight matrix $\mathbf{W}_i$ has dimensions of $K_{i-1} \times K_i$ and the bias $\mathbf{b}_i$ has dimension of K_i for each layer $i = 1, \dots, L$. In classification, the network learns a categorical distribution over the classes. The model output is:

$$\hat{\mathbf{y}} \sim \text{Cat}(\mathbf{y}|\mathbf{f}^{\boldsymbol{\omega}}(\mathbf{x}))$$

where $\hat{\mathbf{y}}$ is a categorical distribution over C classes.

MLE

The model parameters are fitted using a *maximum likelihood criterion*,

$$\begin{aligned}\mathbf{W}_{\text{MLE}} &= \arg\max_{\mathbf{W}} \; p(\mathbf{Y}|\mathbf{X}, \mathbf{W}) \\ &= \arg\max_{\mathbf{W}} \; \log p(\mathbf{Y}|\mathbf{X}, \mathbf{W})\end{aligned}$$

Training our network consists of finding an optimal set of weights. The optimisation is carried out via stochastic gradient descent (SGD), and we assume that $p(\mathbf{Y}|\mathbf{X}, \mathbf{W})$ is differentiable w.r.t. the network's parameters. In classification tasks, the maximum

[3] In our work, we use rectified linear units $\text{relu}(x) = \max(x, 0)$.

likelihood criterion is the same as minimising the cross entropy[4]. The loss is computed as the cross entropy between two categoricals $\mathbf{y}$ and $\hat{\mathbf{y}}$,

$$E(\mathbf{X}, \mathbf{Y}) = -\frac{1}{N} \sum_{n=1}^{N} \log(\hat{p}_{n,c_n})$$

where $\hat{p}_c = \exp(\hat{y}_c)/\sum_{c'} \exp(\hat{y}_{c'})$ and c_n is the observed class for the input n.

MAP

To avoid overfitting, a prior distribution is introduced, and the resulting training criterion is *maximum a posteriori*,

$$\begin{aligned} \mathbf{W}_{\text{MAP}} &= \arg\max_{\mathbf{W}} \log p(\mathbf{W}|\mathbf{X}, \mathbf{Y}) \\ &= \arg\max_{\mathbf{W}} \log p(\mathbf{Y}|\mathbf{X}, \mathbf{W}) + \log p(\mathbf{W}) \end{aligned}$$

For instance, placing Gaussian priors results in L2 regularisation, and the objective function of the single hidden layer network becomes:

$$\mathcal{L}(\mathbf{W_1}, \mathbf{W_2}, \mathbf{b}) = E^{\mathbf{W_1}, \mathbf{W_2}, \mathbf{b}}(\mathbf{X}, \mathbf{Y}) + \lambda_1 ||\mathbf{W_1}||^2 + \lambda_2 ||\mathbf{W_2}||^2 + \lambda_3 ||\mathbf{b}||^2$$

where λ_i is a scaling parameter. As we train neural networks in the MAP setting, we still ignore uncertainty about the parameters. In turn, the network might not generalise well; this results in overconfident predictions in data regimes that are not well covered. MLE approaches are referred to as *frequentist* approaches.

A Probabilistic Perspective: Vanilla Vs. Bayesian

In this work, we implement neural models under a probabilistic perspective, where we deal explicitly with uncertainty by inferring distributions over the model's parameters. Compared to a frequentist approach, being Bayesian implies quantifying uncertainty in a principled way. In vanilla neural networks (i.e. deterministic models), the weights are represented by single and fixed values (i.e. point estimates). Conversely, BNNs place prior distributions over the weights, and obtain a posterior via Bayesian learning. We replace the network's weights with distributions over the parameters (Bishop, 2006). By treating the parameters as unknown quantities, the Bayesian framework provides a principled way to avoid overfitting and overconfidence. In BNNs, the model prediction is given by the weighted average of infinitely many predictions based on infinitely many models drawn from the posterior distribution. Therefore, the network average prediction tends to generalise well. However, both the posterior over the model parameters and the predictive posterior in neural architectures are usually intractable as the marginal likelihood is difficult to compute. In the following section, we discuss Bayesian approximate methods to the posterior distribution.

Notation remark. It is worth bringing to the reader's attention a minor change in the notation that applies to the next sections. This is to avoid creating confusion between deterministic weights matrices and stochastic random weights matrices. From now on in this chapter, we use $\mathbf{W}$ to denote random matrices variables.

[4]The negative log-likelihood is the sum of the cross entropy over all the instances. Maximising the likelihood implies minimising the negative log-likelihood, that in turn implies minimising the cross entropy.

2.3 The Approximate Inference's Landscape

MCMC Vs. VI

As mentioned, the posterior distribution and the predictive distribution for neural networks are intractable. To circumvent this problem, modern Bayesian inference relies on approximate inference methods. In this work, we focus on two powerful classes of algorithms that have been extensively used: Markov Chain Monte Carlo (MCMC) and VI. These methods are based on different ways of approaching Bayesian integration. The general idea behind MCMC is that we draw a finite number of samples from the posterior distribution. In VI, instead, we approximate the posterior distribution via a variational distribution (for instance a Gaussian distribution parametrised by variational parameters θ), and we obtain an approximate posterior distribution that is as close as possible to the true posterior via the optimisation of a variational objective.

We leave behind the Laplace approximation, as it stands to reason that it leads to severe under-fitting (Lawrence, 2001), and it does not scale well to deep architecture. For this reason, it has not been picked-up and extended-on by the community, at least not at a recent time[5]. A notorious alternative that is left behind as well, but is worth mentioning, and investigating further is Hernández-Lobato & Adams (2015)'s probabilistic back propagation (PBP). This method makes use of expectation propagation, and improves upon VI methods. It leads to fast predictions and accurate uncertainty estimates.

Strengths & Weaknesses

As indicated, in this work we focus on MCMC and VI methods. Both inferential methods have weaknesses and strengths, and it is an arduous task to identify which one is the best on a task-by-task basis. However, a good way to ease the comparison is to reason along two coordinates: the expressiveness of the predictive uncertainty, and the computational complexity. In terms of expressiveness, MCMC algorithms are proven to converge to the true posterior. On the other hand, VI methods cannot faithfully match the intractable true posterior distribution, which is is approximated by a class of distribution and within this class we identify (via optimisation) the best approximation. Hence, the approximating variational distribution is restricted to a given class that might not include the true posterior. Usually, we learn with fully factorised posterior (we refer to it as mean-field VI). This is a restrictive approximation that leads to under-fitting uncertainty, and it impedes the learning (Barber et al., 2011). A recent investigation on sequential decision-making scenarios shows that mean field VI performs poorly (Riquelme et al., 2018). To address this pitfall, more structured posteriors are proposed. The more flexible and richer the variational distribution is, the better the performance of the network. However, more powerful alternatives tend to limit the scalability. In this work, to incorporate correlations we place a variate matrix Gaussian over the weights' matrices, as in (Louizos & Welling, 2016).

To accommodate large-scale machine learning, both methods present a stochastic gradient minibatch-based variation (see §2.6.1 and §2.4.1): Stochastic Gradient Variational Bayes (SGVB) (Kingma & Welling, 2013), and Stochastic Gradient Langevin Dynamics (SGLD) (Welling & Teh, 2011), a scalable MCMC method. Both algorithms paved the way to minibatch-based VI and MCMC, respectively. From a practical standpoint, SGVB is an appealing method due to the fact that the variational distribution is effi-

[5] More recent works propose a revival of the Laplace inference, and overcome the overfitting via linearisation of the model output around the MAP estimate leading to a Linear Gaussian model (Foong et al., 2019). To alleviate the lack in scalability, other approaches applied Kronecker-factored Approximate Curvature (K-FAC) (Ritter et al., 2018).

ciently optimised. On the other hand, SGLD is computationally expensive as we need to store a large number of samples. Recently, Monte Carlo dropout has been proposed as a practical approximate inference method that easily scales up to deep neural architectures (Gal & Ghahramani, 2016).
In this work, we focus on VI, MC dropout, and SGLD. We review these methods in §2.4, §2.5, and §2.6, respectively.

2.4 Variational Inference

Preliminaries in Variational Inference

As discussed in §2.1, Bayesian modelling relies on integration. A practice that we would call marginalisation (in Bayesian parlance). VI replaces marginalisation with the optimisation of a variational objective. In other words, we recast inference as an optimisation problem by defining an approximating variational distribution $q_\theta(\boldsymbol{\omega})$ parametrised by θ, such that $q_\theta(\boldsymbol{\omega})$ is close to the true posterior $p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{X})$, which we cannot evaluate analytically. This is achieved by minimising the Kullback-Leibler (KL) divergence (Kullback & Leibler, 1951) between $q_\theta(\boldsymbol{\omega})$ and $p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})$ w.r.t the variational parameters θ,

$$\mathrm{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})) = \int_\Omega q_\theta(\boldsymbol{\omega}) \log \frac{q_\theta(\omega)}{p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})} d\boldsymbol{\omega} \geq 0$$

We consider the KL divergence as a measure of closeness between two distributions; by minimising the discrepancy we reduce the dissimilarity between the two. At prediction, we replace the posterior distribution over the parameter space $p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{X})$ with $q^*_\theta(\boldsymbol{\omega})$, which is the optimum of the optimisation objective. The predictive posterior $p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y})$ is approximated as a variational predictive distribution $q^*_\theta(\mathbf{y}^*|\mathbf{x}^*)$,

$$\begin{aligned} p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) &= \int_\Omega p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}) d\boldsymbol{\omega} \\ &\approx \int_\Omega p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega}) q^*_\theta(\boldsymbol{\omega}) d\boldsymbol{\omega} \triangleq q^*_\theta(\mathbf{y}^*|\mathbf{x}^*) \end{aligned}$$

In practice, we approximate the variational predictive distribution with MC integration,

$$\begin{aligned} \hat{q}_\theta(\mathbf{y}^*|\mathbf{x}^*) = \frac{1}{T} \sum_{t=1}^{T} p(\mathbf{y}^*|\mathbf{x}, \hat{\boldsymbol{\omega}}_t) \xrightarrow[T \to \infty]{} & \int_\Omega p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega}) q_\theta(\boldsymbol{\omega}) d\boldsymbol{\omega} \\ & \approx p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) \end{aligned}$$

where T is the number of MC samples of $\hat{\boldsymbol{\omega}}_t \sim q_\theta(\boldsymbol{\omega})$.

ELBO

In VI methodology, we minimise the divergence of $q_\theta(\boldsymbol{\omega})$ to $p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})$ by maximising a variational lower bound $\mathcal{L}_{\mathrm{VI}}(\theta)$ (a.k.a. *evidence lower bound* (ELBO) (Neal & Hinton, 1998; Jordan et al., 1999)). As shown by Kingma & Welling (2013), a variational lower bound that can be maximised with respect to the variational parameters θ is derived on the (conditional) marginal log-likelihood (i.e. log model evidence). This is formalised as:

$$\text{log evidence} = \log p(\mathbf{Y}|\mathbf{X}) = \mathrm{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})) + \mathcal{L}_{\mathrm{VI}}(\theta)$$

The bound is equal to the (conditional) marginal log-likelihood as the approximate pos-

terior and the true posterior match[6]. Leveraging the fact that the KL divergence is non-negative, it follows that:

$$\begin{aligned}
\log p(\mathbf{Y}|\mathbf{X}) &\geq \log p(\mathbf{Y}|\mathbf{X}) - \mathrm{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y})) \\
&= \int_\Omega q_\theta(\boldsymbol{\omega}) \log p(\mathbf{Y}|\mathbf{X}) d\boldsymbol{\omega} - \int_\Omega q_\theta(\boldsymbol{\omega}) \log \frac{q_\theta(\omega)}{p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y})} d\boldsymbol{\omega} \\
&= \int_\Omega q_\theta(\boldsymbol{\omega}) \log \frac{p(\mathbf{Y},\boldsymbol{\omega}|\mathbf{X})}{q_\theta(\boldsymbol{\omega})} d\boldsymbol{\omega} \\
&= \int_\Omega q_\theta(\boldsymbol{\omega}) \log p(\mathbf{Y}|\mathbf{X},\boldsymbol{\omega}) d\boldsymbol{\omega} - \int_\Omega q_\theta(\boldsymbol{\omega}) \log \frac{q_\theta(\boldsymbol{\omega})}{p(\boldsymbol{\omega})} \\
&= \mathbf{E}_{q_\theta(\boldsymbol{\omega})}[\log p(\mathbf{Y}|\mathbf{X},\boldsymbol{\omega})] - \mathrm{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega})) \triangleq \mathcal{L}_{VI}(\theta) \\
&= \mathbf{E}_{q_\theta(\boldsymbol{\omega})}[\log p(\mathbf{Y}|\mathbf{X},\boldsymbol{\omega})] + \mathbf{H}[q_\theta(\boldsymbol{\omega})] - \mathbf{H}[q_\theta(\boldsymbol{\omega}), p(\boldsymbol{\omega})]^{7}
\end{aligned}$$

We make the modelling assumption $p(\boldsymbol{\omega}|\mathbf{X}) = p(\boldsymbol{\omega})$ as $\boldsymbol{\omega}$ is a sufficient statistic in our model. The lower bound can be separated into two sub-functions. The first term in the RHS is the expected log-likelihood, which cannot be computed analytically. The second term is the KL divergence that can be computed in closed-form. In fact, for certain choices of $q_\theta(\boldsymbol{\omega})$ and $p(\boldsymbol{\omega})$ the KL is tractable. As we reason on BNN approximations, the choice of variational family is important, and the success of VI relies on it. This is a matter for further discussion in §2.4.2.
Alternatively, we can derive the variational lower bound by starting from the (conditional) marginal log-likelihood,

$$\begin{aligned}
\log p(\mathbf{Y}|\mathbf{X}) &= \log \int_\Omega p(\mathbf{Y},\boldsymbol{\omega}|\mathbf{X}) \frac{q_\theta(\boldsymbol{\omega})}{q_\theta(\boldsymbol{\omega})} d\boldsymbol{\omega} \\
&\geq \int_\Omega q_\theta(\boldsymbol{\omega}) [\log p(\mathbf{Y},\boldsymbol{\omega}|\mathbf{X}) - \log q_\theta(\boldsymbol{\omega})] d\boldsymbol{\omega} \\
&= \mathbf{E}_{q_\theta(\boldsymbol{\omega})}[\log p(\mathbf{Y},\boldsymbol{\omega}|\mathbf{X}) - \log q_\theta(\boldsymbol{\omega})] \triangleq \mathcal{L}_{VI}(\theta) \\
&= \mathbf{E}_{q_\theta}[\log p(\mathbf{Y},\boldsymbol{\omega}|\mathbf{X})] + \mathbf{H}[q_\theta(\boldsymbol{\omega})]
\end{aligned}$$

We use the Jensen's inequality[8] to carry out our derivation. This can be also rearranged to the canonical form:

$$\log p(\mathbf{Y}|\mathbf{X}) \geq \mathbf{E}_{q_\theta(\boldsymbol{\omega})}[\log p(\mathbf{Y}|\mathbf{X},\boldsymbol{\omega})] - \mathrm{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega}))$$

If $\log p(\mathbf{Y}|\mathbf{X})$ is constant w.r.t θ, maximising the $\mathcal{L}_{\mathrm{VI}}(\theta)$ w.r.t θ is the same as minimising $\mathrm{KL}(q_\theta(\boldsymbol{\omega}), p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y}))$. The minimisation of the KL divergence w.r.t. the variational parameters θ results in an approximating distribution that is the closest to the true

[6] $\mathrm{KL}(q||p) = 0$ if the two distributions q and p are identical.
[7] We decomposed the KL into entropy and cross-entropy given that $\mathrm{KL}(q||p) = \mathbf{H}[q,p] - \mathbf{H}[q]$.
[8] If α_i are positive numbers which sum to 1, and f is a concave function, the Jensen's inequality states that $f(\sum_{i=1}^n \alpha_i x_i) \geq (\sum_{i=1}^n \alpha_i f(x_i))$

posterior. In practice[9], our minimisation objective is:

$$\begin{aligned}\mathrm{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y})) &\approx -\int_\Omega q_\theta(\boldsymbol{\omega})\log\, p(\mathbf{Y}|\mathbf{X},\boldsymbol{\omega})d\boldsymbol{\omega} + \mathrm{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega})) \\ &= -\sum_{n=1}^{N}\int_\Omega q_\theta(\boldsymbol{\omega})\log\, p(\mathbf{y}_n|\mathbf{x}_n,\boldsymbol{\omega})d\boldsymbol{\omega} + \mathrm{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega}))\end{aligned}$$

Our attempt at fitting an approximate posterior by maximising the variational lower bound results in a distribution that both explains well the data through the expected log-likelihood term and does not overfit as the KL term acts as a regulariser[10].
A naive approach to optimising the variational lower bound is via Monte Carlo sampling on both the expected log-likelihood and the KL[11]. Thus we arrive at a tractable variational objective,

$$\mathcal{L}_{\mathrm{VI}}(\theta) \approx \frac{1}{T}\sum_{t=1}^{T}\log\, p(\mathbf{Y}|\mathbf{X},\hat{\boldsymbol{\omega}}_t) - [\log\, q_\theta(\hat{\boldsymbol{\omega}}_t) - \log\, p(\hat{\boldsymbol{\omega}}_t)]$$

with $\hat{\boldsymbol{\omega}}_t \sim q_\theta(\boldsymbol{\omega})$.

A Naive Approach

Taking the gradients of the variational objective $\mathcal{L}_{VI}(\theta)$ is not as easy as it seems. As proposed by Paisley et al. (2012), we can make a stochastic approximation of the gradients through a (naive) unbiased MC estimator,

$$\begin{aligned}\nabla_\theta \mathbf{E}_{q_\theta(\boldsymbol{\omega})}[f(\boldsymbol{\omega})] &= \nabla_\theta \int_\Omega f(\boldsymbol{\omega})q_\theta(\boldsymbol{\omega})d\boldsymbol{\omega} \\ &= \int_\Omega f(\boldsymbol{\omega})\nabla_\theta q_\theta(\boldsymbol{\omega})d\boldsymbol{\omega} \\ &= \int_\Omega f(\boldsymbol{\omega})q_\theta(\boldsymbol{\omega})\nabla_\theta \log\, q_\theta(\boldsymbol{\omega})d\boldsymbol{\omega} \\ &= \mathbf{E}_{q_\theta(\boldsymbol{\omega})}[f(\boldsymbol{\omega})\nabla_\theta \log\, q_\theta(\boldsymbol{\omega})] \\ &\approx \frac{1}{T}\sum_{t=1}^{T} f(\hat{\boldsymbol{\omega}}_t)\nabla_\theta \log\, q_\theta(\hat{\boldsymbol{\omega}}_t)\end{aligned}$$

with $\hat{\boldsymbol{\omega}}_t \sim q_\theta(\boldsymbol{\omega})$. We use the identity $\nabla_\theta q_\theta(\boldsymbol{\omega}) = q_\theta(\boldsymbol{\omega})\nabla_\theta \log\, q_\theta(\boldsymbol{\omega})$. It follows that $\nabla_\theta \mathbf{E}_{q_\theta(\boldsymbol{\omega})}[f(\boldsymbol{\omega})] = \mathbf{E}_{q_\theta(\boldsymbol{\omega})} f(\boldsymbol{\omega})\nabla_\theta \log\, q_\theta(\boldsymbol{\omega})$. This estimator exhibits high variance, and Paisley et al. (2012) have proved it to be impractical for our purposes.

2.4.1 Practical Variational Inference

Computing the Gradients

As discussed in §2.4, we need to build a MC estimator of the variational lower bound. Computing its exact gradients w.r.t. θ is unfeasible. In a not-ideal circumstance, optimisation methods have to deal with stochastic gradient estimates. We know that the

[9] Maximising the variational lower bound is carried out as minimising the negative evidence lower bound.

[10] The KL term is also interpreted as the complexity loss. We maintain the simplicity of the prior.

[11] In the previous derivations, we assume the KL is tractable and we can compute it in closed-form. For generality, instead, we assume that the KL divergence is also intractable.

convergence of SGD heavily depends on the variance of the gradient estimator[12]. If the gradient w.r.t the variational parameters is too large, in practice the optimisation will fail to converge, or it will take an impractical amount of time. The bias and variance of the estimator play a crucial role in the stochastic optimisation. The path-wise gradient estimator (Kingma & Welling, 2013; Rezende et al., 2014; Titsias & Lázaro-Gredilla, 2014), known as the *generalised parametrization trick*, is the one that has been used the most within the BNN community, as well as in this work. The reparametrization trick can be applied to continuous random variables drawn from probability densities – in our scenario $q_\theta(\boldsymbol{\omega})$ – that can be reparametrised as:

$$\mathbf{E}_{q_\theta(\boldsymbol{\omega})}[f(\boldsymbol{\omega})] \longrightarrow \mathbf{E}_{q(\boldsymbol{\epsilon})}[f(\mathcal{T}(\theta, \boldsymbol{\epsilon}))]$$

where $\mathcal{T}(\theta, \boldsymbol{\epsilon})$ is a differentiable θ-dependent transformation (Jankowiak & Obermeyer, 2018).

Stochastic Minibatch Variational Inference

Later in this chapter, we discuss how VI scales to large datasets. That implies how to adapt VI to stochastic minibatch-based backpropagation. In minibatch stochastic optimisation, for each epoch the training dataset is randomly split into M equally-sized batches, and the gradient update is averaged over the elements in each batch. In this regard, Kingma & Welling (2013) and Rezende et al. (2014) propose an unbiased differentiable minibatch-based MC estimator. More insights will be given below.

The Reparametrization Trick and Stochastic Optimisation Kingma et al. (2015) introduce a practical estimator w.r.t θ by reparametrising the variational posterior $q_\theta(\boldsymbol{\omega})$ by means of a differentiable transformation $g(\theta, \boldsymbol{\epsilon})$ with $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$. Under mild conditions, they show that it is possible to express all model random variables $\boldsymbol{\omega}$ as $\boldsymbol{\omega} = g(\theta, \boldsymbol{\epsilon})$[13], where $\boldsymbol{\epsilon}$ is an auxiliary variable drawn from an independent marginal distribution $p(\boldsymbol{\epsilon})$ (Kingma & Welling, 2013). The reparametrization trick allows us to use backpropagation. We are able to rewrite the MC estimator of an expectation w.r.t a distribution that is parametrised by θ, to be differentiable w.r.t θ. Indeed, we build a MC estimator of an expectation w.r.t. a probability distribution parametrised by θ, such that:

$$\mathbf{E}_{q_\theta(\boldsymbol{\omega})}[f(\boldsymbol{\omega})] = \mathbf{E}_{p(\boldsymbol{\epsilon})}[f(g(\theta, \boldsymbol{\epsilon}))] \approx \frac{1}{T}\sum_{t=1}^{T} f(g(\theta, \hat{\boldsymbol{\epsilon}}_t))$$

with $\hat{\boldsymbol{\epsilon}}_t \sim p(\boldsymbol{\epsilon})$. We apply this approach to the variational lower bound,

$$\hat{\mathcal{L}}(\theta)_{VI} = \sum_{t=1}^{T} \log p(\mathbf{y}_n | \mathbf{x}_n, \boldsymbol{\omega}_t = g(\theta, \boldsymbol{\epsilon}_t)) - \mathrm{KL}(q_\theta(\boldsymbol{\omega}), p(\boldsymbol{\omega}))$$

with $\hat{\boldsymbol{\epsilon}}_t \sim p(\boldsymbol{\epsilon})$. To accommodate the need for an estimator of the variational lower bound based on minibatches, Kingma & Welling (2013) propose an unbiased differen-

[12] Robbins & Monro (1951) prove that stochastic gradient ascent converges to a local optimum given an appropriately decreasing step size. However, in practice, the speed of convergence depends on the gradients' variance. Empirically, it is shown that a too large variance results in impractically slow convergence.

[13] $\mathbf{w}_{i,j} = g(\theta_{i,j}, \boldsymbol{\epsilon}_{i,j})$

tiable minibatch-based MC estimator of the expected log-likelihood,[14]

$$\mathcal{L}_{\text{VI}}(\theta) \approx \hat{\mathcal{L}}_{\text{VI}}(\theta) = \frac{N}{M} \sum_{m=i}^{M} \log p(\mathbf{Y}_m | \mathbf{X}_m, \boldsymbol{\omega} = g(\theta, \boldsymbol{\epsilon})) - \text{KL}(q_\theta(\boldsymbol{\omega}), p(\boldsymbol{\omega}))$$

where $\{\mathbf{X}_m, \mathbf{Y}_m\}_{m=1}^M$ are M samples randomly drawn from the dataset $\mathbf{X}, \mathbf{Y}$ containing N samples, and $\boldsymbol{\epsilon}$ is the random noise vector. The optimisation procedure is reported in Algorithm 1.

Algorithm 1 Stochastic Gradient Variational Bayes (SGVB)

Given our observables $\mathbf{X}, \mathbf{Y}$
Initialise parameters θ
repeat
$\mathbf{X}_m, \mathbf{Y}_m \longleftarrow$ Randomly sampled minibatch of M data-points (drawn from full dataset)
$\boldsymbol{\epsilon} \longleftarrow$ Randomly sampled variables from noise distribution
$\mathbf{g} \longleftarrow \nabla_\theta \hat{\mathcal{L}}_{\text{VI}}(\theta)$ (Gradients of the minibatch estimator w.r.t. θ)
Update variational parameters θ (e.g. SGD)
until convergence of parameters (θ)
return θ

As shown by Kingma & Welling (2013)

$$\text{Var}[\log p(\mathbf{Y}_m | \mathbf{X}_m, \boldsymbol{\omega})] = N^2 \Big(\frac{1}{M} \text{Var}[L_m] + \frac{M-1}{M} \text{Cov}[L_m, L_l] \Big)$$

where L_i is short notation for $\log p(\mathbf{y}_m | \mathbf{x}_m, \boldsymbol{\omega} = g(\theta, \boldsymbol{\epsilon}_m))$ which is the contribution to the log-likelihood of the m-th data-point.
It is inferred that the contribution to $\text{Var}[\log p(\mathbf{Y}_m | \mathbf{X}_m, \boldsymbol{\omega})]$ by the covariance does not scale with M. The $\text{Var}[L_m]$ contribution to the estimator variance is inversely proportional to the minibatch size. In other words, the estimator variance is dominated by the covariance term. This is preventing us from decreasing the estimator variance by increasing the minibatch size. As suggested by Kingma et al. (2015), a way to force the covariance term to be zero is to sample separate $\mathbf{W}_i$ for each data-point in the minibatch. This is impractical.

The Local Reparametrization Trick Kingma et al. (2015) propose a new reparame trization trick that leads to a statistically efficient gradient estimation. This is obtained by enforcing the covariance term to be zero by directly sampling random pre-activation functions, instead of weight matrices $\mathbf{W}_i$. This trick is known to translate global uncertainty into local uncertainty (Kingma et al., 2015).

A Simple Example

Let's consider the single hidden layer fully connected network – its simplicity turns out to be useful again. However, this time we learn with a fully factorised Gaussian posterior,

$$q_\theta(w_{j,k}) = \mathcal{N}(w_{j,k}; \mu_{j,k}, \sigma^2_{j,k}) \; \forall w_{j,k} \in \mathbf{W}_{\{1,2\}}$$

[14] For simplicity, again, we assume that the remaining term (i.e. KL divergence of the variational posterior from the prior) is computed analytically/deterministically; otherwise a similar procedure is carried out.

Given a $M \times Q$ input matrix $\mathbf{X}$, a $Q \times K$ weight matrix $\mathbf{W}_1$, which maps the input space to the hidden space, and a $K \times C$ matrix $\mathbf{W}_2$, which maps the hidden space to the output space, we denote by $\mathbf{A} = \mathbf{X}\mathbf{W}_1$ the input feature matrix to the next layer from the input below[15], and by $\mathbf{B} = \mathbf{A}\mathbf{W}_2$ the pre-activation matrix[16]. The conditional posterior for the pre-activation is a Gaussian as well,

$$q_\theta(b_{m,k}|\mathbf{A}) = \mathcal{N}(b_{m,k}; \mu^{\text{act}}_{m,k}, (\sigma^{\text{act}}_{m,k})^2) \; \forall b_{m,k} \in \mathbf{B}_{\{1,2\}}$$

where the mean and the variance are defined as:

$$\mu^{\text{act}}_{m,k} = \sum_j a_{m,j}\mu_{j,k}$$
$$(\sigma^{\text{act}}_{m,k})^2 = \sum_j a^2_{m,j}\sigma^2_{j,k}$$

This implies sampling the pre-activations as:

$$b_{m,k} = \mu^{\text{act}}_{m,k} + \sigma^{\text{act}}_{m,k} \cdot \eta_{m,k}$$

where $\eta_{m,k} \sim \mathcal{N}(0, 1)$.

2.4.2 Variational Inference Algorithms

Several VI approaches branched off Graves (2011). We discuss the technicalities behind the two that are under scrutiny in this work. We start with the work of Blundell et al. (2015) that led to the "ubiquitous method" known as Bayes By Backpropagation. Then, we review the work of Louizos & Welling (2016) that opened up towards learning with more structured posteriors.

Bayes By Backpropagation (BBB) A common VI method is Bayes By Backpropagation (Blundell et al., 2015). The name comes from the fact that we learn probability distributions over the weights via a backpropagation-compatible approach.

Mean Field Approx.

Blundell et al. (2015) approximate the true posterior of a Bayesian model with a fully factorised distribution,

$$q_\theta(\boldsymbol{\omega}) = \prod_{i=1}^{L} q_\theta(\mathbf{W}_i) = \prod_{i=1}^{L}\prod_{j=1}^{K_i}\prod_{k=1}^{K_{i+1}} q_{\mu_{i,j,k},\sigma_{i,j,k}}(w_{i,j,k}) = \prod_{i,j,k} \mathcal{N}(w_{i,j,k}; \mu_{i,j,k}, \sigma^2_{i,j,k})$$

Here, we imply full independence between weights. They reparametrise the weights of each fully connected layer,

$$\begin{aligned} w_{j,k} &= \mu_{j,k} + \sigma_{j,k} \cdot \epsilon_{j,k} \\ &= \mu_{j,k} + \log(1 + \exp(\rho_{j,k})) \cdot \epsilon_{j,k} \end{aligned}$$

[15] In the case of the first layer (i.e. input layer), $\mathbf{A}$ would equal to $\mathbf{X}$ and $\mathbf{B} = \mathbf{X}\mathbf{W}_1$

[16] To simplify our notation we are omitting the non-linearity function $\sigma(\cdot)$ and the bias.

where $\mu_{j,k}$ and $\rho_{j,k}$ are the variational parameters θ, and $\epsilon_{j,k}$ is drawn from a standard normal distribution $\mathcal{N}(0,1)$. We reparametrise $\sigma_{j,k}$ to enforce its positivity through a Softplus function[17]. This allows an unconstrained optimisation of the variance. The proposed reparametrization inevitably results in doubling the number of the parameters. The reparametrised variational objective is:

$$\log p(\mathbf{Y}|\mathbf{X}) \geq \mathcal{L}_{\text{VI}}(\theta) = \mathbf{E}_{p(\boldsymbol{\epsilon})}\Big[\log p(\mathbf{Y}|\mathbf{X},\boldsymbol{\omega}) - \log\frac{q_\theta(\boldsymbol{\omega})}{p(\boldsymbol{\omega})}\Big]$$

VFE which is usually formulated as variational free energy $\mathcal{F}_{\text{VFE}}(\theta)$ (Yedidia et al., 2001; Friston et al., 2007). Analogously, the latter is computed using MC estimates:

$$\hat{\mathcal{F}}_{\text{VFE}}(\theta) \approx \frac{1}{T}\sum_{t=1}^{T} \log q_\theta(\hat{\boldsymbol{\omega}}_t) - \log p(\hat{\boldsymbol{\omega}}_t) - \log p(\mathbf{Y}|\mathbf{X},\hat{\boldsymbol{\omega}}_t) = -\hat{\mathcal{L}}_{VI}(\theta)$$

with $\hat{\boldsymbol{\omega}}_t \sim q_\theta(\boldsymbol{\omega})$.

Variational Matrix Gaussian (VMG) As discussed in §2.3, learning with fully factorised distribution omits correlations between weights. This is a restrictive approximation. On the other hand, variational matrix Gaussian (VMG) allows the learning of input and output covariances for each network's layer[18]. The matrix variate Gaussian (Gupta & Nagar, 2018) is a distribution over random matrices, that it is fully parametrised by a $R \times C$ mean matrix $\mathbf{M}$, a $R \times R$ covariance (among rows) matrix $\mathbf{U}$, and a $C \times C$ covariance (among columns) matrix $\mathbf{V}$,

$$\begin{aligned} p(\mathbf{W}) &= \mathcal{MN}(\mathbf{M},\mathbf{U},\mathbf{V}) \\ &= \frac{\exp(-\frac{1}{2}\text{tr}[\mathbf{V}^{-1}(\mathbf{W}-\mathbf{M})^T\mathbf{U}^{-1}(\mathbf{W}-\mathbf{M})])}{(2\pi)^{np/2}|\mathbf{V}|^{n/2}|\mathbf{U}|^{n/2}} \end{aligned}$$

We can recast the matrix variate distribution as a multivariate Gaussian distribution (Gupta & Nagar, 2018):

$$p(\text{vec}(\mathbf{W})) = \mathcal{N}(\text{vec}(\mathbf{W}); \text{vec}(\mathbf{M}), \mathbf{V}\otimes\mathbf{U})$$

where $\text{vec}(\cdot)$ is an operator that vectorises the matrix by stacking the columns into a single vector. The resulting covariance matrix is given by Kronecker product ($\otimes$) between $\mathbf{V}$ and $\mathbf{U}$. Similarly, we derive a lower bound on the marginal likelihood,

$$\mathcal{L}_{VI}(\theta) = \mathbf{E}_{q_\theta(\boldsymbol{\omega})}[\log p(\mathbf{Y}|\mathbf{X},\boldsymbol{\omega})] - \text{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega}))$$[19]

[17] $\text{Softplus}(x) = \log(1+\exp(x))$

[18] In §2.4.2 we use a fully factorised posterior, that implies we estimate a variance for each single weight, instead, by learning with a variate matrix Gaussian we estimate a variance over each row and columns of the weight matrix.

We reparametrise the expected log-likelihood as:

$$\mathbf{E}_{q_\theta(\boldsymbol{\omega})}[\log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega})] \approx \frac{1}{T}\sum_{t=1}^{T} \log p(\mathbf{Y}|\mathbf{X}, \hat{\boldsymbol{\omega}}_t)$$

where $\boldsymbol{\omega}_t \sim q_\theta(\boldsymbol{\omega})$. Each network weight matrix is sampled as $\hat{\mathbf{W}}_{i,t} = \mathbf{M} + \mathbf{U}^{\frac{1}{2}}\hat{\mathbf{E}}_t\mathbf{V}^{\frac{1}{2}}$ and $\hat{\mathbf{E}}_t \sim \mathcal{MN}(\mathbf{0}, \mathbf{I}, \mathbf{I})$. As discussed in §2.4.1, the local reparametrization trick leads to a more efficient estimator. The local reparametrization trick can still be applied in this context due to the fact that the inner product between a matrix $\mathbf{A}_i$ and a matrix variate Gaussian $\mathbf{W}_i$ is still a matrix variate Gaussian (Gupta & Nagar, 2018). As in §2.4.1, $\mathbf{B}_i = \mathbf{A}_i\mathbf{W}_i$ is distributed as:

$$p(\mathbf{B}_i|\mathbf{A}_i) = \mathcal{MN}(\mathbf{A}_i\mathbf{M}, \mathbf{A}_i\mathbf{U}\mathbf{A}_i^T, \mathbf{V})$$

Sampling from this distribution is inefficient due to the fact that we need to compute the square root of the row covariance $\mathbf{A}_i\mathbf{U}\mathbf{A}_i^T$. Louizos & Welling (2016) propose an efficient method by sampling from the marginal distribution (i.e. diagonal) over the pre-activations,

$$p(\mathbf{b}_i|\mathbf{a}_i) = \mathcal{N}(\mathbf{a}_i\mathbf{M}, (\mathbf{a}_i\mathbf{U}\mathbf{a}_i)^T \odot \mathbf{V})$$

Again, $\mathbf{b}_i$ conditioned on $\mathbf{a}_i$ follows a multivariate Gaussian distribution. Louizos & Welling (2016) adopt the concept of pseudo-data from Snelson & Ghahramani (2006) and condition each pre-activation variable on pseudo-inputs $\tilde{\mathbf{A}}$ and pseudo-outputs $\tilde{\mathbf{B}}$ such that:

$$p(\mathbf{b}_i|\mathbf{a}_i, \tilde{\mathbf{A}}, \tilde{\mathbf{B}}) = \mathcal{N}(\mathbf{a}_i\mathbf{M} + \boldsymbol{\sigma}_{12}^T\boldsymbol{\Sigma}_{11}^{-1}(\tilde{\mathbf{B}} - \tilde{\mathbf{A}}\mathbf{M}), (\sigma_{22} - \boldsymbol{\sigma}_{12}^T\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\sigma}_{12}) \odot \boldsymbol{V})$$

Each covariance term is estimated as:

$$\boldsymbol{\Sigma}_{11} = \tilde{\mathbf{A}}\mathbf{U}\mathbf{A}^T \quad \boldsymbol{\sigma}_{12} = \tilde{\mathbf{A}}\mathbf{U}\mathbf{a}_i^T \quad \sigma_{22} = \mathbf{a}_i\mathbf{U}\mathbf{a}_i^T$$

This allows a more efficient sampling of the network pre-activations.

[19] Refer to the Appendix A for a derivation of the KL-divergence between two matrix variate Gaussians.

2.5 Monte Carlo Dropout

Gal & Ghahramani (2015) re-interpret dropout (Srivastava et al., 2014) as an approximate inference method where the approximating distribution is a mixture of two Gaussians with small variances, and one of the two Gaussians has mean equal to zero.

Dropout as SRT

Dropout is a stochastic regularisation technique[20] (SRT) where a proportion of the network weights are zeroed during the training. Briefly, we review dropout on a single hidden layer neural network. Firstly, we sample from a Bernoulli distribution with some parameters $p_i \in [0, 1]$ two vectors $\mathbf{z}_1$ and $\mathbf{z}_2$ of dimensions Q and K, respectively. Hence, the output of the first layer is obtained by $\sigma((\mathbf{x} \odot \mathbf{z}_1)\mathbf{M}_1 + \mathbf{b})$, which is equivalent to $\sigma(\mathbf{x}(\text{diag}(\mathbf{z}_1)\mathbf{M}_1))$. In this operation we "drop-out" (i.e. set to zero) elements in $\mathbf{x}$ with $1 - p_1$ probability. Analogously, this is repeated for the output layer. Accordingly, the output of the single hidden network is

$$\hat{\mathbf{y}} = \sigma(\mathbf{x}(\text{diag}(\mathbf{z}_1)\mathbf{M}_1) + \mathbf{b})(\text{diag}(\mathbf{z})\mathbf{M}_2)$$

We rearrange the latter as $\hat{\mathbf{y}} = \sigma(\mathbf{x}\mathbf{W}_1 + \mathbf{b})\mathbf{W}_2$ where $\mathbf{W}_1$ and $\mathbf{W}_2$ are random variable realisations of the network weights. At test time dropout does not take place[21].

2.5.1 Monte Carlo Dropout as Bayesian Inference

MC Dropout

Dropout can be interpreted as a practical Bayesian inference method. As in §2.4, a variational distribution $q_\theta(\boldsymbol{\omega})$ is defined, and the KL divergence is minimised,

$$\text{KL}(q(\boldsymbol{\omega})||p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})) \propto -\int_\Omega q(\boldsymbol{\omega}) \log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega}) d\boldsymbol{\omega} + \text{KL}(q(\boldsymbol{\omega})||p(\boldsymbol{\omega}))$$

where $q_\theta(\boldsymbol{\omega})$ is a Bernoulli variational approximation for each layer i in the network

$$\hat{\mathbf{W}}_i = \mathbf{M}_i \cdot \text{diag}([\hat{\mathbf{b}}_{i,j}]_{j=1}^{K_i})$$

with $\hat{\mathbf{b}}_{i,j} \sim \text{Bernoulli}(p_i)$[22] and $\mathbf{M}_i$ are the variational parameters θ. Dropout is applied before each layer. The dropout minimisation objective is given by (Gal & Ghahramani, 2016)[23]:

$$\mathcal{L}(\theta, p) = -\frac{1}{N}\sum_{n=1}^{N} \log p(\mathbf{y}_n|\mathbf{x}_n, \hat{\boldsymbol{\omega}}_t) + \frac{1-p}{2N}||\theta||^2$$

with $\hat{\boldsymbol{\omega}}_t = \{\hat{\mathbf{W}}_{i,t}\}_{i=1}^L \; \forall \hat{\mathbf{W}}_i \sim q_\theta^*(\boldsymbol{\omega})$, and $\theta = \{\mathbf{M}_i\}_{i=1}^L$. At test time, we also perform dropout to sample from the variational distribution, and we approximate the predictive

[20] Dropout is a popular regularisation technique introduced to prevent overfitting and co-adaptation between weights.

[21] This is true for regular dropout, in MC dropout we do perform dropout at testing time to sample from the variational distribution.

[22] If we sample $\mathbf{z}_{i,j}$ from a Gaussian distribution $\mathcal{N}(1, \alpha)$, similarly we recover Multiplicative Gaussian Noise (Kingma et al., 2015).

[23] The KL term is approximated as L2 regularisation over the weights (Gal & Ghahramani, 2016, Appendix, section A).

posterior using MC integration:

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) \approx \int_\Omega p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega}) q(\boldsymbol{\omega}) d\boldsymbol{\omega}$$
$$\approx \frac{1}{T} \sum_{t=1}^{T} p(\mathbf{y}^*|\mathbf{x}^*, \hat{\boldsymbol{\omega}}_t)$$

with $\hat{\boldsymbol{\omega}}_t \sim q_\theta^*(\boldsymbol{\omega})$, and T is the total number of stochastic forward passes on the same input (e.g. each forward pass corresponds to sampling masked model weights).

2.6 Sampling Methods

An alternative to VI is to use Stochastic Gradient Markov Chain Monte Carlo (SG-MCMC). Hamiltonian Monte Carlo (Neal, 2012) is the father of these methods. It is still impractical as it employs the Metropolis accept-reject step and computes the log-likelihood over the entire dataset (Riquelme et al., 2018). Different methods have been developed to adapt Hamiltonian Monte Carlo to stochastic gradient minibatch updates (Chen et al., 2014). In this work, we review Stochastic Gradient Langevin Dynamics (SGLD), one of most common SG-MCMC methods. Langevin methods are a simplification of Hamiltonian dynamics[24]. Originally, Langevin methods are derived as the discretisation of a stochastic differential equation whose equilibrium happens to conform to the posterior distribution (Welling & Teh, 2011),

$$d\boldsymbol{\omega}_t = \frac{1}{2} \nabla_{\boldsymbol{\omega}} \log p(\boldsymbol{\omega}_t) + d\eta_t$$

where η_t is the standard Brownian motion. More complex Hamiltonian approaches (e.g. enhancing the dynamics by momentum variables) do not scale well with large datasets. Welling & Teh (2011) suggest a simple modification of SGD where they inject a controlled amount of noise to the gradients' updates[25] such that the model's parameters converge to the posterior distribution. Again, we use a MC approximation of the posterior predictive by generating a set of samples $\{\hat{\boldsymbol{\omega}}_t\}$ "collected" from the trajectory. It has been proved that MCMC methods (Robert & Casella, 2013) converge to the true posterior for a decreasing step-size ϵ_t[26] (Teh et al., 2016). This is true if the Robbins and Monro conditions are satisfied (Robbins & Monro, 1951) despite the fact that we omit the accept-reject tests[27],

$$\sum_{t=1}^{\infty} = \infty \quad , \sum_{t=1}^{\infty} \epsilon_t^2 < \infty$$

[24] Langevin methods only require a single leapfrog step. This simplifies the inference considerably, and allows for larger dataset.

[25] SGLD only requires the gradient updates on minibatches.

[26] Empirically, it has been shown that a fixed step-size converges faster, and leads to better performances (Nagapetyan et al., 2017). In this work, we decrease the step-size as in (Li et al., 2016)

[27] That implies all the states visited by the SGLD's update rule are accepted.

2.6.1 Stochastic Gradient Langevin Dynamics (SGLD)

An Enhanced SGD

In practice, Stochastic Gradient Langevin Dynamics is similar to a popular class of stochastic optimisation methods (Robbins & Monro, 1951), where the gradient update on subsets of the dataset is used to approximate the true gradient over the entire dataset. For instance, SGD update rule is:

$$\Delta\boldsymbol{\omega}_t = \epsilon_t\Big(\nabla \log\, p(\boldsymbol{\omega}_t) + \frac{N}{M}\sum_{m=1}^{M} \nabla \log\, p(\mathbf{y}_m|\mathbf{x}_m, \boldsymbol{\omega}_t)\Big)$$

Unlike SGD, we inject noise in the gradient update to encourage exploration. Thanks to the noise, the parameters do not collapse to the MAP mode; instead, the trajectory of the parameters converges to the full posterior (Welling & Teh, 2011).

SGLD

The SGLD update rule is:

$$\Delta\boldsymbol{\omega}_t = \frac{\epsilon t}{2}\Big(\nabla \log\, p(\boldsymbol{\omega}_t) + \frac{N}{M}\sum_{m=1}^{M} \nabla \log\, p(\mathbf{y}_m|\mathbf{x}_m, \boldsymbol{\omega}_t)\Big) + \eta_t$$

with $\eta_t \sim \mathcal{N}(0, \epsilon_t)$. As aforementioned, the full posterior, as well as the predictive posterior at test time, is approximated through ensembling[28]. A burn-in period is considered before we start collecting samples. MCMC needs few iterations (i.e. burn-in) to approach for instance a mode of the target distribution. By injecting uniform noise in all parameters' directions, we hinder the optimisation process (Marceau-Caron & Ollivier, 2017). This results in slow mixing.

p-SGLD

Li et al. (2016) refine the method introducing a diagonal preconditioning matrix $\mathbf{G}(\boldsymbol{\omega})$ (p-SGLD). Li et al. (2016) integrate the adaptive preconditioner from the RMSProp optimisation scheme with SGLD. At time t, the update rule is:

$$\Delta\boldsymbol{\omega}_t = \frac{\epsilon t}{2}\Big[\mathbf{G}(\boldsymbol{\omega}_t)\Big(\nabla \log\, p(\boldsymbol{\omega}_t) + \frac{N}{M}\sum_{m=1}^{M} \nabla \log\, p(\mathbf{y}_m|\mathbf{x}_m, \boldsymbol{\omega}_t)\Big)\Big] + \mathbf{G}^{\frac{1}{2}}(\boldsymbol{\omega}_t)\eta_t$$

with $\eta_t \sim \mathcal{N}(0, \epsilon_t)$. We update the preconditioner sequentially, and use only the current gradient information,

$$\mathbf{G}(\boldsymbol{\omega}_{t+1}) = \mathrm{diag}\Big(\mathbf{1} \oslash (\lambda\mathbf{1} + \sqrt{V(\boldsymbol{\omega}_{t+1})})\Big)$$
$$\mathbf{V}(\boldsymbol{\omega}_{t+1}) = \alpha\mathbf{V}(\boldsymbol{\omega}_t) + (1-\alpha)\bar{\mathbf{g}}(\boldsymbol{\omega}_t; \boldsymbol{X}_m, \boldsymbol{Y}_m) \odot \bar{\mathbf{g}}(\boldsymbol{\omega}_t; \boldsymbol{X}_m, \boldsymbol{Y}_m)$$[29]

where $\bar{\mathbf{g}}(\boldsymbol{\omega}_t; \boldsymbol{X}_m, \boldsymbol{Y}_m) = \frac{1}{M}\sum_{m=1}^{M} \nabla \log\, p(\mathbf{y}_m|\mathbf{x}_m, \boldsymbol{\omega}_t)$ is the empirical mean of the gradient of a minibatch $\mathbf{X}_m, \mathbf{Y}_m$, and $\alpha \in [0, 1]$. The parameter λ controls the extreme of the curvatures, and α balances the contribution of past and current weights. By scaling both the injected noise and the gradients' updates, we fine-tune the step-size along each parameter's direction.

[28] This is an enormous drawback for this approach that scales so well on large datasets. Ensembling creates additional memory cost and computational complexity.

[29] $\oslash$ and $\odot$ represent element-wise matrix product and division, respectively.

3 Preliminary Investigations

In this chapter, we investigate several approximate inference methods and network architectures. We use *passive* learning[1] on the MNIST dataset: we train on the whole dataset, and do not query any data-point, as the model is not actively involved in the selection of the dataset it is trained on.
The chapter is structured as it follows. In §3.1, we report our preliminary investigations of several approximate inference methods, alongside insights into their implementation. In §3.2, we discuss 2 evaluation metrics that we consider in our analysis, alongside accuracy. In §3.3, we comment our results and on the need for a more principled banchmarking practice, such as active learning.
The theoretical framework needed to train, predict, and sample from the networks is thoroughly described in the previous chapters (see §2). However, in this section we also take the opportunity to discuss additional technical insights along with our results. All the implementations are in Pytorch, and are compatible with GPU processing.

3.1 Investigating Approximate Inference Methods via Passive Learning

Passive Learning on MNIST

The purpose of this section is to report the model performances for the chosen approximate inference methods, as well as providing benchmarks for further analysis and substantiating our claim for a more expressive benchmark. Ultimately, as mentioned before, this section gives detailed insights of our implementations: we describe several algorithmic design principles that we considered throughout our investigations.

Our Methodology

Overall we investigate 3 inference methods (VI, MC dropout and SGLD), and 2 architectures: fully stochastic fully connected networks and Neural Linear models (see Table 2). A description of our experimental setups follows. We split the dataset into 3 subsets: a training set, a validation set, and a test set. Each set contains 40k, 10k and 10k digits, respectively. The chosen resolution of each digit is set to 32×32. The training set and validation set are pre-processed by cropping the digits at a random location. This step is included as regularisation strategy. Empirically, we noticed that the learning is hindered if no pre-processing is done. The model tends to over-fit the training set, and this leads to a loss in classification accuracy on the validation set, and ultimately on the test set. The models are trained for 100 epochs, unless stated otherwise. For all models presented, the optimisation is done with SGD (Kiefer et al., 1952), and a batch size of 256 is used.

Our Findings

We report our findings in Table 2. To accommodate for the stochasticity in the training, the results shown in Table 2 are obtained by averaging over 5 distinct runs. As a benchmark, we report test accuracy of the fully connected network with 2 hidden layers trained in (Simard et al., 2003). We also report the performances of a fully connected network that applies dropout as a regularisation strategy, as in (Srivastava, 2013). An analysis on preliminary findings is drawn up in §3.3.
An excursus on the implementations reported in Table 2, along with their technicalities follows below.

[1] We use *passive* learning as the opposite of active learning.

3.1.1 Variational Inference Models

Variational Inference & Sharing Information among Weights

As discussed in §2.4, Variational inference (VI) approaches aim to approximate the posterior by finding a distribution within a tractable family that minimises the KL divergence to the posterior (Kingma et al., 2015). These approaches solve an optimisation problem. The main criticism of variational approaches is that they underestimate uncertainty (Bishop, 2006), which could lead to a sub-optimal way of querying data-points (in active learning parlance). We implement Bayes By Backpropagation and variational matrix Gaussian (VMG) algorithms, which correspond to the VI method of (Blundell et al., 2015), and of (Louizos & Welling, 2016), respectively. As mentioned in §2.4, it is common to approximate the posterior by a mean field approximation. In this scenario, each neural network's weight is modelled via an independent Gaussian distribution. The reason we implement VI methods with matrix variate Gaussian posteriors is to consider the correlations among weights.

Mean Field Approximation All the methods named Bayes By Backpropagation (see Table 2) implement the mean field approximation, and it is a stochastic gradient VI algorithm based on the work of (Blundell et al., 2015).

Generalised Reparametrization Trick

Optimisation is efficiently performed by using the generalised reparametrization trick to obtain an unbiased estimate of the ELBO's gradient with respect to the variational parameters. The reparametrization trick is used, and the forward pass is performed by instantiating the weights' matrices through sampling from the respective distributions. In Bayes By Backpropagation (Local Reparametrization), the local reparametrization trick presented in (Kingma et al., 2015), is implemented. In this case, we end up sampling pre-activation functions. The local reparametrization implementation computes the KL divergence term in closed-form, as we use an isotropic Gaussian centred at 0 with unit variance.

Variance Initialisation

A common issue encountered in training when introducing model uncertainty is that the optimisation process would diverge. Here, we define the posterior approximation on the weights to be a fully factorised Gaussian, i.e. $\mathrm{q}_\theta(\mathrm{w}_{j,k}) = \mathcal{N}(\mu_{j,k}, \sigma^2_{j,k})$. In order to ensure convergence in training, variance initialisation has to be carefully considered. The variational approximation for each layer is initialised as it follows: the weight means $\mu_{j,k} \sim \mathcal{N}(0, 0.05)$, along with the weight standard deviations that are parametrised as $\sigma = \log(1 + \exp(\rho))$[2], and $\rho_{j,k} \sim \mathcal{N}(-4, 0.05)$, $\forall(\mu_{i,j}, \rho_{j,k}) \in \mathbf{W}$. There are few reasons behind the success of the proposed initialisation. We initialise to such a small variance, due to the fact that the variance of the weights is directly proportional to the variance of the gradient estimator. A larger variance in the gradient estimator would cause the algorithm to diverge. By reducing the variance in the initialisation, we reduce the variance of the gradient estimator, and we favour convergence. Another important aspect that needs to be addressed is that as long as the variances are small, we are performing MAP estimation. Or at least, we are learning similarly to MAP estimation. At first, the network experiences MAP behaviour, and as the optimisation proceeds, the variance is encouraged to grow in order to minimise the KL divergence. In fact, as the weight variances are so small, the variational approximation approaches a delta function around

[2]Refer to §2.4 for the reason behind the variance reparametrization through the parameter ρ and for the use of a Softplus function.

the means. This leads to the data being well-fitted. It is worth wondering whether that leads to very limited coverage of the posterior, resulting in poor uncertainty estimates. In Table 1, we show that a less conservative variance initialisation leads to worse results in terms of accuracy, and also to a less calibrated model.

	Log ρ Initialisation	
Bayes By Backpropagation	$\mathcal{U}$(-2, -3)	$\mathcal{N}$(-4, 0.05)
Negative Predictive LL	0.0848 ± 0.0047	**0.0317 ± 0.0040**
Error (%)	2.5680 ± 0.0873	**0.9140 ± 0.0531**
Expected Calibration Error	0.0113 ± 0.0010	**0.0061 ± 0.0004**

Table 1: Average and std. test predictive log-likelihood (LL), test error, and test expected calibration error (ECE)(with M = 10 bins) for different variance initialisation in Bayes By Backpropagation using an isotropic Gaussian likelihood. A fully connected network of 2 hidden layers with 400 units each is used. Results are obtained averaging over 5 distinct runs.

Priors on Weights

In the case of Bayes By Backpropagation[3] both the components of the ELBO are estimated using the MC approximation, as both can be rewritten as an expectation over the variational posterior. We consider 2 priors, an isotropic Gaussian centred at zero and with unit variance, along with a Gaussian Mixture Model (GMM), where each density has zero mean, and the standard deviations of the 2 mixture components, σ_1 and σ_2, are 0.135 and 0.001, respectively. The scaling factor, π, is 0.5. Our prior resembles a spike-and-slab (Mitchell & Beauchamp, 1988). Here, we do not optimise the parameters of the prior. Empirically, it has been shown that it yields worse results (Blundell et al., 2015). Bayes By Backpropagation, when using priors, and estimating the ELBO with the MC approximation is trained longer, 200 epochs.

Deep Matrix Variate Bayesian Networks This VI algorithm needs extra attention on the parameters' initialisation, and on the assumptions we operate under. We initialise the mean $\mathbf{M}$, the covariance matrices $\mathbf{U}$ (among-row) and $\mathbf{V}$ (among-column) to the scheme proposed in (He et al., 2015). In order to have a computationally tractable model, we approximate the covariance with a diagonal matrix, hence we assume independent rows and columns – we are not fitting a full matrix variate Gaussian posterior as in (Zhang et al., 2017). Overall, this results in a per-layer parametrization that requires less learnable parameters compared to a simple fully factorised Gaussian posterior (see Table 3). Moreover, even if there are less parameters to learn, we do allow "information sharing" between the weights due to the correlations (see §2.4.2). We parametrised the prior of each weight matrix $p(\mathbf{W})$ to a zero mean and unit variance matrix variate Gaussian $\mathcal{MN}(\mathbf{0}, \mathbf{I}, \mathbf{I})$. As in (Hernández-Lobato & Adams, 2015), we divide the input to each layer by the square root of its dimensionality to keep the scale of the pre-activations independent of the incoming connections. We incorporate pseudo-inputs to allow a more efficient sampling of the posterior. We used 150 pseudo-data pairs $(\tilde{\mathbf{A}}, \tilde{\mathbf{B}})$ for each layer, unless stated otherwise. We initialise the pairs $(\tilde{\mathbf{A}}, \tilde{\mathbf{B}})$ sampling from $\mathcal{U}[-0.01, 0.01]$. To

[3] We use the generalised reparametrization trick.

account for the biases, we assume that each input to the layers is augmented with an extra dimension containing all 1s. Thus, code-wise, we do not have an explicit variable for the biases but rather, we concatenate a vector of 1s to the input at each layer, and then we use an augmented weight matrix. This is equivalent to optimising the biases explicitly. It is worth mentioning that the reproducibility of any model that includes a VMG layer is compromised due to the computation of a matrix inversion to sample the network pre-activations (see §2.4.2). We lose numerical precision due to rounding errors, and that results in some randomness in learning. However, this leads to mild differences in learning curves.

3.1.2 MC Dropout Models

MC Dropout as Approximate Inference

As discussed in §2.5, dropout is a SRT where the output of each neuron is zeroed out with probability p_{drop} at each forward pass (Srivastava, 2013). We implement dropout uncertainty (a.k.a MC dropout) presented in (Gal & Ghahramani, 2016), where approximate inference is done as a product of Bernoulli's implemented as dropout before each layer. The model uncertainty is estimated by collecting the results of stochastic forward passes through the network (see §2.5). The number of stochastic forward passes (T) used to estimate uncertainty during training is 5. On the other hand, we use 10 forward iterations to estimate uncertainty at test time. We do not do the forward passes concurrently; in other words, we do not loop over the forward iterations, instead we use a minibatch augmentation trick. Each minibatch is repeated as many times as the number of forward iterations. We compute 10 forward passes at once by augmenting the minibatch at the beginning of the first layer. For example, the first minibatch is 256×1024. Hence, the input to the first layer has dimensionality of 2560×1024 ($\text{T} = 10$). This is to optimise the run-time.

3.1.3 Stochastic Gradient Langevin Dynamics

Adding Stochasticity in Gradients Update

In BNNs, MCMC methods are one of the simplest but most reliable techniques. Sampling methods estimate the posterior distribution through drawing samples. We review these methods in §2.6. Here, we implement SGLD (Welling & Teh, 2011; Marceau-Caron & Ollivier, 1988) which modifies SGD to provide random values to the parameters' updates, which happen to be distributed according to a Bayesian posterior. Bayesian inference is as simple as running "noisy SGD".

Including Preconditioning

A different strategy that augments the gradients and noise according to a preconditioning matrix has been implemented as well (see §2.6.1). We use a preconditioner based on the RMSprop algorithms, as proposed in (Li et al., 2016). pSGLD uses RMSprop preconditioning. As suggested in (Li et al., 2016), to aid convergence, we scale the Langevin noise by the number of data-points in the dataset. The initial learning rate is 0.001 for both SGLD and preconditioned SGLD. The prior is a Gaussian $\mathcal{N}(0, 1)$. The network is optimised for 400 epochs. The Bayesian posterior ensemble is built by storing the last 100 samples taken every 2 epochs (thinning process), or 390 weight updates. We use an ensemble of networks sampled from the trajectory (computationally quite costly). At least 15 epochs of burn-in period are used. The learning rate is scaled exponentially

based on the number of optimisation steps[4], as in (Li et al., 2016).

3.1.4 Neural Linear Models

A Compromise between Deterministic & Stochastic

We consider architectures where approximate Bayesian inference is only used for small pieces of the overall network. The Neural Linear models consist of a deterministic CNN feature extractor followed by one stochastic fully connected layer (e.g. Mean Field Approx., Full Covariance, VMG). Our CNN is a simple deterministic feature extractor consisting of 3 convolutional layers with 8, 16, and 32 filters, respectively, a max-pool non linearity function, and a dense layer of 512 units. A 5×5 kernel is used in the first filter, and a 3×3 kernel for the next two. Batch normalisation (Ioffe & Szegedy, 2015) is used after each convolutional layer, followed by a ReLu activation function. This choice was made to be simple and maximally general. Hence, we consider inference only over the last layer of the neural network. This class of algorithms is popular among the ML community, and the popularity derives from the fact that the Neural Linear decouples representation learning and uncertainty estimation (Riquelme et al., 2018). In Mean Field Approximation, the last layer is a Bayes By Backpropagation layer (VI family), where local reparametrization is used, and the KL is computed in closed-form. Analogously, Full Covariance implements the local reparametrization trick, and a low-rank matrix approximation (rank = 2) of the variance matrix is used. This approach is dropped in further analysis due to the excessively long run-time. It is important to mention that the architecture is optimised jointly. This is to ease the inference in the "head" by means of the feature representation learnt by the "body".

3.2 Evaluation Metrics

In this work, as evaluation metrics we report the predictive log-likelihood and the expected calibration error (ECE), along with accuracy. In the next sections, we provide more insights into predictive log-likelihood and ECE, which have been extensively used to contrast BNNs performances, respectively in the work of Gal (2016) and Zhang et al. (2017), and many others.

3.2.1 Predictive Log-Likelihood

The predictive log likelihood captures how well the model fits the data. We approximate the log score using MC integration:

$$\log\left(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}\right) \approx \log\Big(\frac{1}{T}\sum_{t=1}^{T} p(\mathbf{y}^*|\mathbf{x}^*, \hat{\boldsymbol{\omega}}_t)\Big)$$

with $\hat{\boldsymbol{\omega}}_t \sim q_\theta^*(\boldsymbol{\omega})$. Larger values indicate a better fit.

[4] A practical rule is used, the step-size is scaled by $\epsilon_t = at^{-\gamma}$ and $\gamma \in (0.5, 1]$.

3.2.2 Expected Calibrated Error (ECE)

Model Calibration Matters!

We report the expected calibration error along with the predictive log-likelihood and test error as a scalar statistic of model calibration, and as a proxy for the quality of model uncertainty estimation. The underlying assumption is that well-calibrated BNNs provide good uncertainty estimation, due to the fact that calibration can be interpreted as "in-distribution" uncertainty.

Briefly, we review the work of Guo et al. (2017). Let's consider a supervised multi-class classification problem, and a neural network that is trained with input $\mathbf{X} \in \mathcal{X}$, and label $\mathbf{Y} \in \mathcal{Y} = \{1, \dots, C\}$. We define a perfect calibration as:

$$\mathcal{P}(\hat{Y} = Y \mid \hat{P} = p) = p, \quad \forall p \in [0, 1]$$

where $\hat{Y}$ is a class prediction, and $\hat{P}$ is the associated probability of correctness. To compute a scalar statistic of calibration, we first estimate the expected accuracy, by grouping predictions in M interval bins, and computing the accuracy of each bin,

$$\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbf{1}(\hat{y}_i = y_i)$$

where B_m is the set of indices of samples that fall within the m-th bin. The average confidence within the bin is defined as:

$$\text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i$$

where $\hat{p}_i$ is the i-th sample confidence obtained by applying the softmax function to the network output[5]. Accordingly, perfect calibration implies $\text{acc}(B_m) = \text{conf}(B_m)$ for all m. Thus, the expected calibration error is defined as:

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |\text{acc}(B_m) - \text{conf}(B_m)|$$

where M is a fixed number of bins, and n the number of samples.

A More Intuitive Explanation

To gain an intuitive understanding, let's think of $\text{acc}(B_m)$ as the true fraction of correct instances within the m-th bin, and $\text{conf}(B_m)$ is the mean of post calibrated probabilities for the instances in the m-th bin (i.e. the average confidence within the m-th bin). Ideally, we would like a confidence estimate $\hat{P}$ to represent a true probability. In other words, if our network makes 100 predictions, and it is confident 0.8 for each prediction, then we should expect 80 out of 100 predictions to be correct. The lower the value of ECE, the better the calibration of the model.

[5] In BNNs $\hat{p}_i = \frac{1}{T} \sum_{t=1}^{T} \text{Softmax}(\mathbf{f}^{\hat{\boldsymbol{\omega}}_t}(\mathbf{x}))$ with $\hat{\boldsymbol{\omega}}_t \sim q_\theta(\boldsymbol{\omega})$.

3.3 " One-Off " Metrics Do Not Work Too Well

Are " One-Off " Metrics Informative Enough?

At this point, it seems reasonable to ask whether standard metrics (such as predictive log-likelihood, expected calibration error, accuracy, etc.) evaluated in a passive learning scenario are meaningful when it comes to contrasting different approximate inference schemes. From here on out, we refer to these metrics as "one-off" metrics. As we infer from Table 2 – we report the performances of different inference methods and different architectures on MNIST – it is difficult to gauge the quality of the approximations based on "one-off" metrics. It is not clear at least on MNIST, whether these metrics are able to discriminate which model (or inference method) outperforms the others. We provide some evidence in favour of the idea that "one-off" metrics do not deliver on the quality of uncertainty estimation. For instance, it is not easy to contrast inferential methods in fully connected networks based on our experimental findings in Table 2. The comparison is not as thorough as we would like it to be. As discussed in §2.3, we would expect VMG to outperform models based on mean field approximation. This is a restrictive assumption. In fact, it is likely that the true posterior has correlations between weights. VMG and stochastic gradient models are the only models under consideration that take into account covariances. As a result, they allow a more truthful estimation of the posterior, and overall they should perform better. In §5, we draw similar conclusion on Fashion MNIST (see Table 5). It is not clear either whether pSGLD outperforms SGLD, as we would expect. These observations are valid for both the 400 unit fully connected networks, as well as the 800 unit fully connected ones. Regarding the Neural Linear models, it is difficult, almost impossible, to show any significant trends as well: the models perform similarly. In §5, we repeat the same analysis on Fashion MNIST and SVHN (see Table 6). Again, it is not possible to highlight any trends.

A little later in this work, we propose AL as a suitable Bayesian diagnostic toolbox, and show that more meaningful comparisons can be drawn between approximate inference schemes.

Inference Method	#Layer/#Units	Negative Predictive LL	Error (%)	Expected Calibration Error
Neural Linear Models				
Mean Field Approx.	Deterministic CNN Extractor:	0.0211 ± 0.0024	0.6320 ± 0.0605	0.0046 ± 0.0004
Full Covariance (rnk = 2)	3 Convolutional Layers	0.0217 ± 0.0032	0.6300 ± 0.0210	0.0048 ± 0.0003
Variational Matrix Gaussian	Max Pooling (2,2)	0.0256 ± 0.0030	0.6700 ± 0.0379	0.0051 ± 0.0005
MC Dropout ($p_{drop} = 0.2$)	Dense(512)	0.0209 ± 0.0040	0.6040 ± 0.0896	0.0048 ± 0.0005
Benchmark Deterministic Fully Connected Models				
Maximum Likelihood (Simard et al., 2003)	2/800	-	0.7	-
Dropout (Srivastava, 2013)	2/800	-	1.25	-
Our Benchmark	2/400	0.0348	1.0300	0.0061
Variational Inference Fully Connected Models				
Bayes By Backpropagation* (Gaussian Prior)				
	2/400	0.0317 ± 0.0040	0.9140 ± 0.0531	0.0061 ± 0.0004
	2/800	0.0313 ± 0.0023	0.9060 ± 0.0441	0.0062 ± 0.0004
Bayes By Backpropagation* (GMM Prior)				
	2/400	0.0427 ± 0.0007	1.3020 ± 0.0564	0.0119 ± 0.0005
	2/800	0.0436 ± 0.0015	1.3100 ± 0.0540	0.0118 ± 0.0007
Bayes By Backpropagation (Local Reparametrization)				
	2/400	0.0347 ± 0.0017	1.0600 ± 0.0363	0.0062 ± 0.0002
	2/800	0.0341 ± 0.0015	1.0380 ± 0.0471	0.0062 ± 0.0004
Variational Matrix Gaussian				
	2/400	0.0279 ± 0.0027	0.7980 ± 0.0491	0.0057 ± 0.0003
	2/800	0.0262 ± 0.0027	**0.7180 ± 0.0741**	0.0053 ± 0.0007
Dropout Uncertainty Fully Connected Models				
MC Dropout ($p_{drop} = 0.2$)				
	2/400	0.0353 ± 0.0014	1.1480 ± 0.1130	0.0062 ± 0.0004
	2/800	0.0306 ± 0.0007	0.9640 ± 0.0441	0.0061 ± 0.0005
Sampling Methods - Fully Connected Models				
Stochastic Gradient Langevin Dynamics				
	2/400	0.0374 ± 0.0041	0.8211 ± 0.0335	0.0062 ± 0.0002
	2/800	0.0359 ± 0.0034	0.8193 ± 0.0193	0.0066 ± 0.0002
Preconditioned Stochastic Gradient Langevin Dynamics				
	2/400	0.0400 ± 0.0014	0.8283 ± 0.0335	0.0068 ± 0.0003
	2/800	0.0444 ± 0.0070	0.8233 ± 0.0170	0.0072 ± 0.0002

Table 2: Average and std. test predictive log-likelihood (LL), test error, and test expected calibration error (ECE) (with M = 10 bins) for different approximate inference schemes and network architectures (Neural Linear models, 400-400 or 800-800 fully connected models). The results are obtained on MNIST dataset consisting of 32×32 images from 10 different classes with 50,000 training, 10,000 validation, and 10,000 testing samples. We average over 5 different runs. $*$ 200 epochs are used to guarantee convergence.

Inference Method	**#learnable parameters**
Neural Linear Models	
Mean Field Approx.	208k
Full Covariance (rnk = 2)	210k
MC Dropout	206k
VMG	247k
Fully Connected Models*	
Bayes By Backpropagation	1.148m
VMG	972k
MC Dropout	574k

Table 3: Number of learnable parameters for different network architectures. * The number of learnable parameters are reported for fully connected architectures with 2 hidden layers with 400 units each. It is worth mentioning that the number of learnable parameters for the SGLD architectures is the same as the one for MC dropout.

4 Active Learning: Theory and Applications

What is Active Learning?

Active learning (AL) (a.k.a "query learning") is a well-established statistical research field that the machine learning community has taken over, expanded-on, and eventually "re-branded" (Cohn et al., 1996). In statistics literature, AL is referred to as *optimal experimental design* (Atkinson, 1992; Chaloner & Verdinelli, 1995).

Usually, AL is presented as a powerful tool to overcome the so-called *labelling bottleneck*. Labelling large datasets through human annotators is a costly and slow process (Zhu, 2005), and finding a way to narrow down a small "salient" subset to be labelled is of course appealing. However, our work does not focus at all on attaining data efficiency, instead, the data acquisition process is the relevant aspect.

In any AL framework, the system is initially trained on a small amount of data, and then it is asked to seek the most informative data-points to be trained on. We gain experience by gathering data interactively and progressively through queries. In classification, for each query the model consults an *oracle* (e.g a human labeler) to observe the true class of the queried data, in regression the output.

Pool Sampling

Our learner is responsible for expanding its own training dataset. It queries an input $\mathbf{x}^*$ from a dataset of unlabelled data $\mathcal{D}_{pool}$, and once it observes the output y^*, it includes the new experience in the next training ($\mathbf{x}^*$ moves from $\mathcal{D}_{pool}$ to $\mathcal{D}_{train}$[1] and the model parameters $\boldsymbol{\omega}$ are updated, accordingly). This active learning scheme is often called *pool-sampling* AL.

Why Active Learning?

At this point, the reader might foresee the reason we believe AL is a suitable testing ground to disambiguate the comparison between such an abundance of approximate inference methods. Indeed, AL is a more principled tool to reason on the trade-offs discussed in §2.3. Through AL, we elaborate on the expressiveness of the posterior, as the acquisition process leverages on the model's representation of the uncertainty. We believe that a model that "knows what it knows" and as well "knows what it doesn't know" queries in a more informed way. In any AL framework, "accurate" estimates of uncertainty are needed to obtain a good performance. Building upon Hernández-Lobato & Adams (2015), Gal et al. (2017), and Zhang et al. (2017), we investigate approximate inference methods via AL, and present empirical evidence of its adequacy.

Chapter Outline

AL literature is vast, and covers several fields. In this chapter, we next turn our focus to information theoretic approaches in AL. In §4.1, we discuss how to "optimally" query data-points. In particular, we review the work of Houlsby et al. (2011) and of Gal (2016). We discuss two acquisition functions that leverage different ways of dealing with uncertainty. We also discuss how to make use of the variational distribution $q_\theta(\boldsymbol{\omega})$ in BNNs to build up computationally tractable approximations of these criteria.

4.1 Acquisition Functions

The system, once trained on a small proportion of the dataset – at this point our system is just knowledgeable enough about the environment to start actively collecting data – takes action through acquisition functions. In other words, acquisition functions dictate the way our model interacts with the environment. Different heuristics have been

[1] To be consistent with the notation in §2, we define $\mathcal{D}_{train} = \mathbf{X}, \mathbf{Y}$.

proposed on the way to query. In fact, querying strategies differ on how they assess the expected informativeness of our candidates (i.e. data-points in $\mathcal{D}_{pool}$). From a statistical standpoint, we select "optimally" the data-points to query by maximising an acquisition function:

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathcal{D}_{pool}}{\arg\max}\, \alpha(\mathbf{x}, p(\boldsymbol{\omega}|\mathcal{D}_{train}))^{2}$$

In this work, as querying strategies we use predictive entropy (often referred to as max entropy), and BALD, a more principled criterion introduced by Houlsby et al. (2011).

Predictive Entropy Predictive entropy is defined as:

$$\mathbf{H}[y|\mathbf{x}, \mathbf{X}, \mathbf{Y}] = -\sum_c p(y = c|\mathbf{x}, \mathbf{X}, \mathbf{Y}) \log p(y = c|\mathbf{x}, \mathbf{X}, \mathbf{Y})$$

The predictive entropy has its maximum value as the probability of an input $\mathbf{x}$ to take any class $c = 1, \cdots, C$ is uniform and equal across all classes (i.e. the prediction is highly uncertain). By computing $\mathbf{H}[y|\mathbf{x}, \mathbf{X}, \mathbf{Y}]\ \forall \mathbf{x} \in \mathcal{D}_{pool}$, we evaluate each data-point in $\mathcal{D}_{pool}$ according to "how much" information is contained in the predictive distribution under a given model. Hence, we can select the data-point in $\mathcal{D}_{pool}$ for which the model is most uncertain.

Predictive Entropy Approx.

Predictive entropy is intractable to compute for BNNs. An estimator approximating it is built as it follows,

$$\begin{aligned}
\mathbf{H}[y|\mathbf{x}, \mathbf{X}, \mathbf{Y}] &= -\sum_c p(y = c|\mathbf{x}, \mathbf{X}, \mathbf{Y}) \log p(y = c|\mathbf{x}, \mathbf{X}, \mathbf{Y}) \\
&= -\sum_c \Big(\int_\Omega p(y = c|\mathbf{x}, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}) d\boldsymbol{\omega} \Big) \log \Big(\int_\Omega p(y = c|\mathbf{x}, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}) d\boldsymbol{\omega} \Big) \\
&\approx -\sum_c \Big(\int_\Omega p(y = c|\mathbf{x}, \boldsymbol{\omega}) q^*_\theta(\boldsymbol{\omega}) d\boldsymbol{\omega} \Big) \log \Big(\int_\Omega p(y = c|\mathbf{x}, \boldsymbol{\omega}) q^*_\theta(\boldsymbol{\omega}) d\boldsymbol{\omega} \Big) \\
&\approx -\sum_c \Big(\frac{1}{T} \sum_{t=1}^{T} p(y = c|\mathbf{x}, \hat{\boldsymbol{\omega}}_t) \Big) \log \Big(\frac{1}{T} \sum_{t=1}^{T} p(y = c|\mathbf{x}, \hat{\boldsymbol{\omega}}_t) \Big) \\
&\triangleq \hat{\mathbf{H}}[y|\mathbf{x}, \mathbf{X}, \mathbf{Y}]
\end{aligned}$$

with $\hat{\boldsymbol{\omega}}_t \sim q^*_\theta(\boldsymbol{\omega})$ and $t = 1, \cdots, T$ indexing the t-th stochastic forward pass.
For clarity purposes, we re-propose how we approximate the posterior predictive distri-

[2] We abuse the standard notation. In batch-mode active learning, the arg max returns not a single data-point, but multiple ones at once. This is further discussed in §5.1.

bution in the context of BNNs:

$$
\begin{aligned}
p(y = c|\mathbf{x}, \mathbf{X}, \mathbf{Y}) &= \int_{\Omega} p(y = c|\mathbf{x}, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}) d\boldsymbol{\omega} \\
&\approx \int_{\Omega} p(y = c|\mathbf{x}, \boldsymbol{\omega}) q_{\theta}^{*}(\boldsymbol{\omega}) d\boldsymbol{\omega} \\
&\approx \frac{1}{T} \sum_{t=1}^{T} p(y = c|\mathbf{x}, \hat{\boldsymbol{\omega}}_t) \\
&= \frac{1}{T} \sum_{t=1}^{T} \text{Softmax}(\mathbf{f}^{\hat{\boldsymbol{\omega}}_t}(\mathbf{x}))
\end{aligned}
$$

with $\hat{\boldsymbol{\omega}}_t \sim q_{\theta}^{*}(\boldsymbol{\omega})$. As T approaches infinity, we recover the predictive entropy for a given data-point,

$$
\hat{\mathbf{H}}[y|\mathbf{x}, \mathbf{X}, \mathbf{Y}] \xrightarrow[T \to \infty]{} \mathbf{H}[y|\mathbf{x}, \mathbf{X}, \mathbf{Y}]
$$

BALD As an alternative to predictive entropy, we review a more principled acquisition function that is based on mutual information between model predictions and the posterior distribution,

$$
\begin{aligned}
\mathbf{I}[y, \boldsymbol{\omega}|\mathbf{x}, \mathbf{X}, \mathbf{Y}] &= \mathbf{H}[\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}] - \mathbf{E}_{p(y|\mathbf{x}, \mathbf{X}, \mathbf{Y})} \Big[\mathbf{H}[\boldsymbol{\omega}|y, \mathbf{x}, \mathbf{X}, \mathbf{Y}] \Big] \\
&= \mathbf{H}[y|\mathbf{x}, \mathbf{X}, \mathbf{Y}] - \mathbf{E}_{p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})} \Big[\mathbf{H}[y|\mathbf{x}, \boldsymbol{\omega}_t] \Big]
\end{aligned}
$$

where $\mathbf{H}[y|\mathbf{x}, \mathbf{X}, \mathbf{Y}]$ is the predictive entropy, $\mathbf{E}_{p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})} \Big[\mathbf{H}[y|\mathbf{x}, \boldsymbol{\omega}_t] \Big]$ is the expected predictive entropy w.r.t the posterior distribution over the model parameters $\boldsymbol{\omega}$. In AL, we maximise the decrease in expected posterior entropy. This results in selecting the data-point $\mathbf{x}$ from $\mathcal{D}_{pool}$, for which the predictive entropy is high – on average the model is highly uncertain – and the expected predictive entropy is low. There still exist parameters' configurations that result in overconfident disagreeing predictions (Gal, 2016). We can view this as seeking informativeness through disagreement between predicted outcomes under the model parameters. That is why we refer to this criterion as Bayesian Active Learning with Disagreement (BALD)[3]. The work of Depeweg et al. (2017) opens up a new interpretation of BALD as an estimation of epistemic uncertainty. By epistemic uncertainty we mean uncertainty due to the randomness in $\boldsymbol{\omega}$. On the other side, aleatoric uncertainty originates from randomness inherent to the data. The first term can be interpreted as the total uncertainty within our prediction under the model, whilst the second term is a measure of the aleatoric uncertainty. The difference between the two results in the epistemic uncertainty.

BALD Approx. As in (Gal, 2016), we approximate the BALD acquisition function via a computationally

[3]Finally, we reveal what BALD stands for, it needed a sort of an introduction. We hope the reader will excuse us.

tractable estimator,

$$
\begin{aligned}
\mathbf{I}[y, \boldsymbol{\omega} | \mathbf{x}, \mathbf{X}, \mathbf{Y}] &= \mathbf{H}[y | \mathbf{x}, \mathbf{X}, \mathbf{Y}] - \mathbf{E}_{p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y})}\Big[\mathbf{H}[y | \mathbf{x}, \boldsymbol{\omega}_t]\Big] \\
&= -\sum_c p(y = c | \mathbf{x}, \mathbf{X}, \mathbf{Y}) \log\, p(y = c | \mathbf{x}, \mathbf{X}, \mathbf{Y}) \\
&+ \mathbf{E}_{p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y})}\Big[\sum_c p(y = c | \mathbf{x}, \boldsymbol{\omega}) \log\, p(y = c | \mathbf{x}, \boldsymbol{\omega})\Big] \\
&\approx -\sum_c \int_\Omega p(y = c | \mathbf{x}, \boldsymbol{\omega}) q^*_\theta(\boldsymbol{\omega}) d\boldsymbol{\omega} \cdot \\
&\cdot \log\Big(\int_\Omega p(y = c | \mathbf{x}, \boldsymbol{\omega}) q^*_\theta(\boldsymbol{\omega}) d\boldsymbol{\omega}\Big) + \\
&+ \mathbf{E}_{q^*_\theta(\boldsymbol{\omega})}\Big[\sum_c p(y = c | \mathbf{x}, \boldsymbol{\omega}) \log\, p(y = c | \mathbf{x}, \boldsymbol{\omega})\Big] \\
&\approx -\sum_c \Big(\frac{1}{T}\sum_{t=1}^T p(y = c | \mathbf{x}, \hat{\boldsymbol{\omega}}_t)\Big) \log\Big(\frac{1}{T}\sum_{t=1}^T p(y = c | \mathbf{x}, \hat{\boldsymbol{\omega}}_t)\Big) + \\
&+ \frac{1}{T}\sum_{c,t} p(y = c | \mathbf{x}, \boldsymbol{\omega}_t) \log\, p(y = c | \mathbf{x}, \hat{\boldsymbol{\omega}}_t) \\
&\triangleq \hat{\mathbf{I}}[y, \hat{\boldsymbol{\omega}} | \mathbf{x}, \mathbf{X}, \mathbf{Y}]
\end{aligned}
$$

with $\hat{\boldsymbol{\omega}}_t \sim q^*_\theta(\boldsymbol{\omega})$.
Analogously,

$$
\hat{\mathbf{I}}[y, \hat{\boldsymbol{\omega}}_t | \mathbf{x}, \mathbf{X}, \mathbf{Y}] \xrightarrow[T \to \infty]{} \mathbf{I}[y, \boldsymbol{\omega} | \mathbf{x}, \mathbf{X}, \mathbf{Y}]
$$

5 An Empirical Evaluation via Active Learning

A New Benchmark for Bayesian Neural Network

In this section, we report and discuss the results obtained during the experimental investigations, along with the underlying rationale. We present an empirical comparison of several inference schemes for deep BNNs through a new bench-marking practice via AL. We investigate whether a sequential decision-making scenario is a suitable scheme to assess the impact of the posterior approximation. Nowadays, approximate inference methods are contrasted on the basis of metrics that are not tailored for statistical learning. Arguably, this section aims to show that an AL setting (more specifically a pool-sampling AL scheme) is a suitable testing ground for a wide range of approximate inferences.

Four Critical Points

Throughout the chapter, we attempt to answer the following questions with the aid of an AL framework:

* Are we learning sensible uncertainty estimates? Does uncertainty estimation help? In other words, do we need to be stochastic? Or is deterministic just good enough?
* Which inference method is best suited for AL? Are we seeking the most informative data-points? Are the results consistent across different datasets? If not, why?

In terms of the degree of stochasticity, and the architecture:

* To what extent does the network need to be stochastic? Do we need fully stochastic networks? Would a single or a few stochastic layers be enough?
* How does the model architecture and its capacity affect the results?

There is a twofold benefit: we attempt to provide a plausible answer to these questions by means of an AL framework; thus, we will assess its suitability when it comes to contrasting a variety of well-established inferential methods.

Chapter Outline

The chapter is structured as it follows. In §5.1, we contrast different inferential methods and architectures within an AL framework on MNIST (LeCun et al., 1998). In §5.2, we compare being Bayesian vs being deterministic in AL. In §5.3, we scale up to more complex datasets, such as Fashion MNIST (Xiao et al., 2017) and SVHN (Goodfellow et al., 2013). In Appendix B, we include preliminary investigations on a simple 2D dataset via AL.

5.1 Investigating Approximate Inference Methods via Active Learning

As aforementioned, a better way of contrasting approximate inference methods is through a benchmark that leverages uncertainty estimation on different grounds, rather blindly relying on "one-off" metrics. That is where AL comes into play. AL is an important pillar of probabilistic learning, and it is widely used to attain data-efficiency. In this chapter, we exploit its promising application to uncertainty evaluation. In AL, the model is asked to explore regions of the space which it is most unfamiliar with. From an empirical standpoint, we investigate how different posterior approximations affect the network performance within an AL framework.

5.1.1 Practical Active Learning

All models are initially trained with a random minibatch (i.e. on a small amount of data)[1]. Iteratively, the model queries its own dataset out of a pool of unlabelled data-points $\mathcal{D}_{\text{pool}}$. The aim is to seek the most informative data-points to be trained on. The querying strategy is dictated by acquisition functions. In §4.1, we review 2 acquisition functions extensively used throughout our investigations, max entropy (Shannon, 1948) and BALD (Houlsby et al., 2011). After each query, the model is updated, (i.e. re-trained from scratch as in §3), on an augmented dataset. We move the queried data from the pool of unlabelled data $\mathcal{D}_{\text{pool}}$ to the dataset the model is trained $\mathcal{D}_{\text{train}}$. In practice, AL consists of 4 steps:

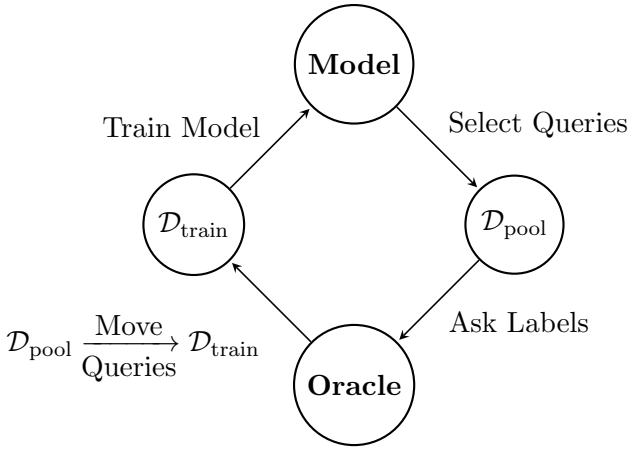


Figure 1: Active learning pool-sampling cartoon. Instead of *a priori* labelling a large dataset, we train the model only on a minibatch. Iteratively, it queries new samples from a pool of unlabelled data. It observes the labels, and updates its parameters on the augmented dataset. The process continues until we reach the budget, or a suitable degree of accuracy.

Terminology

In the majority of our investigation, we are not querying a single data-point at each acquisition step, but we query in batch. Before we dig into batch-mode AL, it is worth introducing a few AL technical words. In the following sections, we would refer to the batch size as the number of data-points queried in each learning cycle, and we would refer to the budget as the total number of data-points that the model is allowed to query. For instance if the batch size is 500, the budget is 5000 (10 steps acquisition process).

Myopic & Naive Batch-Mode Active Learning

The querying strategy in batch-mode AL is the following: we evaluate the acquisition function on an unlabelled pool set using 100 MC samples[2], then we observe the pool data-points with highest acquisition value. In fact, the "informativeness" of new data-points is established by the acquisition function. As reported in Algorithm 2, while acquiring multiple data-points, we take the N-best that report the highest acquisition function. For example, if our batch size is 50, and the pool of acquired data-points contains 50k instances, we would select 50 data-points out of 50k that report the highest

[1] It is important to mention that from now on we refer to minibatch with a batch-mode AL perspective.

[2] As discussed in §4.1, max entropy and BALD are intractable in BNNs and need to be approximated via MC integration.

Algorithm 2 Greedy Batch-Mode Active Learning (Pool-based Sampling)

Input: # acquisition steps b, batch-size N, labelled dataset $\mathcal{D}_{\text{train}}$, unlabelled pool $\mathcal{D}_{\text{pool}}$

$\mathrm{A}_0 \leftarrow \emptyset$
Train model parameters $\mathrm{p}(\boldsymbol{\omega}|\mathcal{D}_{train})$
for t $\leftarrow$ 1 to b **do**
  for $\mathbf{x} \in \mathcal{D}_{\text{pool}} \setminus \mathrm{A}_{t-1}$ **do**
    $\mathrm{s}_{\mathbf{x}} \leftarrow \alpha_{\text{acq}}(\mathbf{x}, \mathrm{p}(\boldsymbol{\omega}|\mathcal{D}_{\text{train}} \cup \mathrm{A}_{t-1}))$
  $\{\mathbf{x}_n^*\}_{n=1}^N \leftarrow \arg\max\{s_{\mathbf{x}}\}$ `argmax selects more than one element`
  $\mathrm{A}_t \leftarrow \mathrm{A}_{t-1} \cup \{\mathbf{x}_n^*\}_{n=1}^N$
  Update model parameters $\mathrm{p}(\boldsymbol{\omega}|\mathcal{D}_{\text{train}} \cup \mathrm{A}_t)$

Output: updated model parameters, and predictive posterior at each acquisition step

acquisition function value (e.g. highest BALD score). As we discuss in Appendix B, querying through the N-best criterion does not always work well, as we omit to consider the overlap in information. This results in querying correlated (or equivalently nearby) data-points (Kirsch et al., 2019; Robert Pinsler, 2019).

Getting Started As a preliminary protocol, we train our model on 500 labelled data-points that are selected randomly. The batch size is 500 and the budget 5000, accordingly. Figure 2 reports how accuracy and ECE vary throughout learning. We investigate fully connected models using as inference scheme VI (Mean Field Approximation and Variational Matrix Gaussian) and MC dropout. It is possible to infer that VI with matrix variate Gaussian outperforms the other methods in 2 distinct ways. The learning curve is significantly steeper and converges to a better final value. Predictive log-likelihood, accuracy, and ECE improve over random. This protocol is shown as our first evidence that a sequential decision-making framework is a more informative testing ground for contrasting Bayesian networks.

Paving the Way for Further Analysis

Although this protocol *per se* is not as informative as we would like it to be, it is indicative for understanding how to calibrate AL. That means, how many data-points should I start with, and how many samples at each acquisition step. As a good rule of thumb, we want our network to be at least 60% accurate, as we start acquiring new data. If the model is not properly trained, AL could go terribly wrong, and the acquired data-points might hinder the learning. Therefore, it is desirable to start with a good model. The architectures that we use for this experimental setting are fully connected models with 2 hidden layers of 400 units each. We use 2 activation functions, max entropy and BALD. As a benchmark, we also report AL as we acquire data-points randomly.

5.2 "Am I Better Than Deterministic?"

In assessing BNNs, a critical issue is to establish whether uncertainty is well-estimated, and whether it makes a positive contribution towards solving a given task. We compare Bayesian networks with a deterministic benchmark within an AL scenario. In practice, by deterministic benchmark we mean a network that has the same architecture as the Bayesian counterpart, but implements deterministic layers. Both models capture aleatoric uncertainty (i.e. data uncertainty); we make the assumption that the degree of improvement upon deterministic is related to the estimation of epistemic uncertainty

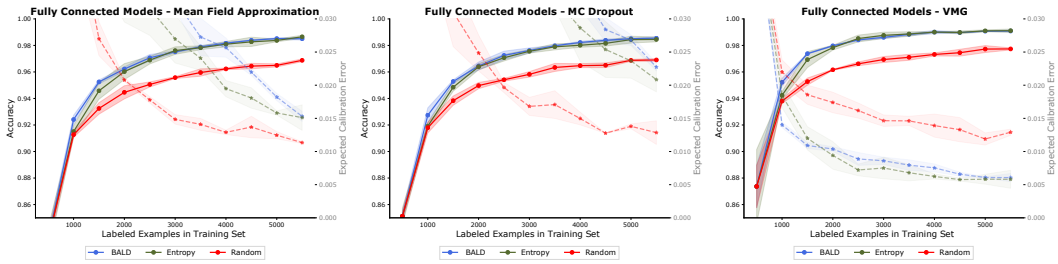


Figure 2: *Active Learning - A Starter.* We investigate 3 inference schemes in an AL framework. We use 2 hidden fully connected networks. We show accuracy and ECE (dotted line) as a function of the # acquired images. We use the MNIST dataset. We show AL for 2 acquisition functions: Max Entropy, and BALD. Random is reported as benchmark. Initially, we train on 500 labelled data-points, and progress in batch of 500 with a budget of 5000. At each acquisition step, the networks are re-trained for 200 epochs.

(i.e. model uncertainty). Hence, by "beating deterministic" we imply a good modelling of epistemic uncertainty.

A New Proxy for Uncertainty Quality

To compare inferential methods, we propose a metric that concisely summarises the learning. We report the area between the learning curves of the models under comparison. We compute numerical integration, and normalise with respect to the x axis. Ideally, the metric should be protocol-independent. For this investigation, we compute the area between the learning curves (e.g. predictive log-likelihood as a function of# acquired images) obtained by the Bayesian network and the deterministic benchmark, respectively; and use it as a proxy for epistemic uncertainty estimation.

For this analysis, we update our protocol, and start with a smaller number of labelled data-points, only 200[3], and we progress in steps of 50 data-points per query. A take home message from the previous investigation (see Figure 2) is to start with less labelled data, and to better focus on how the model behaves prior to convergence. For the deterministic benchmark, we use max entropy as the acquisition function. The protocol is repeated for fully connected models, and Neural Linear models. Moreover, we do believe that predictive log-likelihood is more informative than accuracy as it comes to assess uncertainty estimation. As discussed in §3.2.1, it is not a yes/no metric, but it considers whether the model is over-confident. Therefore, from now on we report accuracy only in the Appendix D.

It is worth mentioning that from a computational point of view, the acquisition phase is faster with Neural Linear models than with fully connected networks; during the data selection phase only the network head (in our scenario 1 layer) is used. Next, we discuss our main findings for each approximate inference method and architecture.

Fully Connected Architectures As shown in Figure 3, VI with mean field approximation performs poorly, as well as MC dropout. In terms of learning curves, the latter

[3]The size of the stochastic minibatch is adjusted dynamically. We use the following heuristic: the power of 2 closest to half of the size of the dataset.

architectures perform similarly. Both inferences do not report any improvement over the deterministic network (see Table 4). We investigate as well whether this is due to underestimating uncertainty, or to stochastic optimisation. We find that the performance of VI with mean field slowly improves as the number of training epochs increases, but still under-performs compared to the deterministic network. It seems that this is the evidence of the detrimental effect of underestimating the variance of the posterior as we are restricted to a limited approximating variational distribution (see §2.3). SGLD and pSGLD do not perform better than a deterministic network, as well. The only inference method that outperforms deterministic is VI with matrix variate Gaussian posteriors. It is our front-runner. This is coherent with our analysis in §2.3. Indeed, we believe that this is due to the fact that correlations between weights are taken into consideration. As a side remark, VI with matrix variate Gaussian posteriors using BALD as the acquisition function performs better than max entropy. This is expected since BALD greedily maximises the expected information with respect to the model parameter. As aforementioned in §4.1, BALD reasons in terms of epistemic uncertainty. Hence, we do expect BALD to be a more principled acquisition strategy than max entropy.

Neural Linear Models As shown in Figure 4, Neural Linear architectures behave as a deterministic network. We think that this results from over-parameterising the problem, and by that we mean the representational power of the CNN feature extractor is so good for MNIST it cancels out any contribution given by uncertainty. Either we simplify the feature extractor, or we test these architectures on a more complex dataset. On this matter, we should also consider that "easy datasets" report little aleatoric uncertainty, thus we would expect that acquiring via BALD to be similar to acquiring via maximising entropy. Moreover, as mentioned in (Riquelme et al., 2018), we should also hold accountable the fact that Neural Linear optimisation is tricky. The joint optimisation of the deterministic and stochastic components could lead to a too conservative optimisation of the stochastic layer (e.g. the stochastic layer is performing MAP estimation).

Inference method	**Predictive LL***	**Accuracy***	**ECE***
Mean Field Approx.	-0.0197 ± 0.0030	-0.0002 ± 0.0011	-0.0407 ± 0.0056
MC Dropout	-0.0127 ± 0.0058	0.0011 ± 0.0014	-0.0409 ± 0.0091
VMG	**0.0624 ± 0.0059**	**0.0254 ± 0.0023**	-0.0027 ± 0.0020
SGLD	0.0102 ± 0.0163	0.0019 ± 0.0046	**0.0015 ± 0.0013**
pSGLD	-0.0153 ± 0.0138	0.0004 ± 0.0045	-0.0051 ± 0.0031

Table 4: *A More Thoughtful Investigation.* We report the area between learning curves between BALD and Det. Entropy. A positive area means that the network is performing better than deterministic. Conversely, a negative implies performing worse than deterministic. Higher is better. $*$ denotes that we are reporting an area.

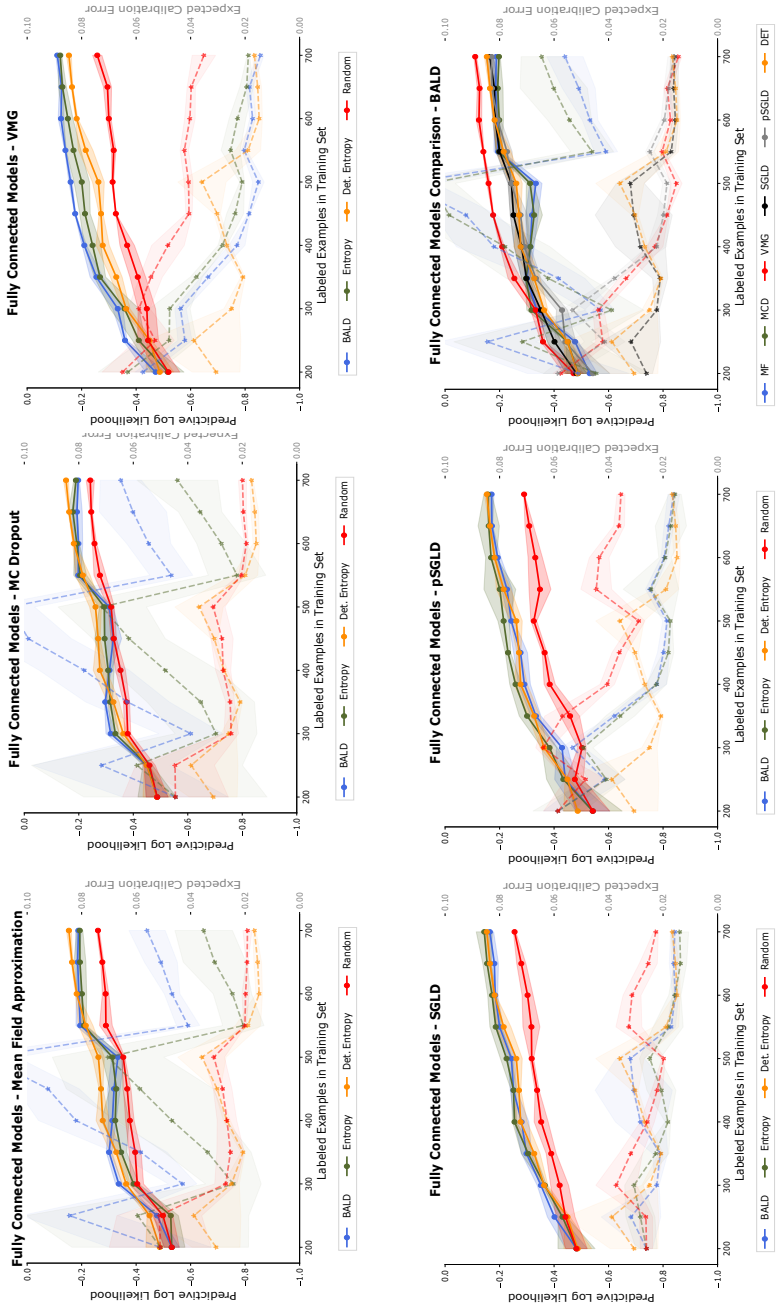


Figure 3: *A More Thoughtful Investigation.* We investigate 5 different inference schemes using a 2 hidden layer fully connected network in an AL framework. We use the MNIST dataset. We show predictive log-likelihood and ECE (dotted line) as a function of the # acquired images. We show AL for 2 acquisition functions: Max Entropy, and BALD. We report Random and Det. Entropy (short for deterministic Entropy) as benchmarks. Initially, we train on 200 labelled data-points, and progress in batches of 50 with a budget of 500. At each acquisition step, the networks are re-trained for 400 epochs.

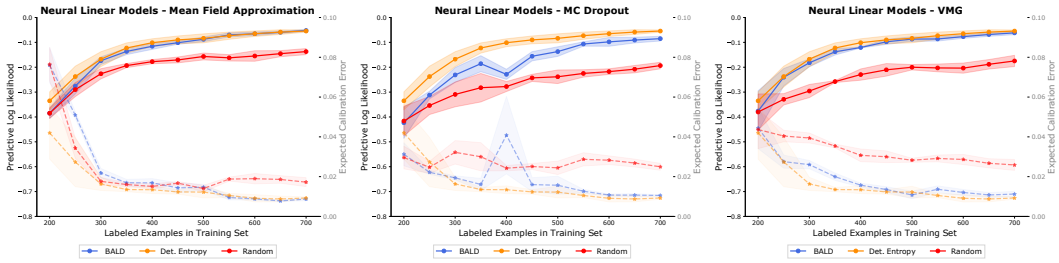


Figure 4: *"Am I Better Than Deterministic? Part I"*. We investigate the Neural Linear architecture, and use 3 different inference schemes. We use the MNIST dataset. We show predictive log-likelihood and ECE (dotted line) as a function of the # acquired images. We show AL for 2 acquisition functions: Max Entropy, and BALD. We report Random and Det. Entropy (short for deterministic Entropy) as benchmarks. Initially, we train on 200 labelled data-points, and progress in batches of 50 with a budget of 500. At each acquisition step, the networks are re-trained for 200 epochs.

5.3 Scaling Up to More Complex Datasets

Task Complexity, Network Capacity, Inference Choice

The previous analysis reports that the majority of Bayesian models behave similarly to deterministic networks when tested within an AL framework. Arguably, MNIST may not be complex enough to assess to what extent we would benefit from BNNs in a sequential decision-making scenario. Therefore, we test our models on a more rugged ground. The reason behind this choice is that modelling uncertainty might play a major role as we challenge the network with more difficult tasks (and keeping the architecture constant). We repeat the previous analysis on more complex datasets such as Fashion MNIST and SVHN. The rationale behind this section is to disclose, by the means of AL, whether we can draw trade-offs between dataset complexity (e.g. irreducible variability inherent to the data, class abstraction, etc.), network capacity, and nature of approximate inference method. Next, we discuss our main findings for each approximate inference method and architecture.

Fully Connected Architectures On a more complex dataset (see Figure 5), all the tested inferential methods perform better than deterministic in terms of predictive log-likelihood, and ECE. These results confirm that uncertainty estimation plays a major role on complex tasks within an AL framework. It is also shown that VI with matrix variate Gaussian posteriors outperforms other inferential methods in terms of predictive log-likelihood, and accuracy at the end of the acquisition process. However, it starts badly, and this is reflected in Table 7 where we cannot identify a winning inferential approach. Mean field VI and MC dropout present faster convergence in terms of predictive log-likelihood and ECE. The latter systems perform likewise, and this confirms the similar nature of these approximations. Mean field VI, VMG and MC dropout considerably outperform SGLD and pSGLD in terms of predictive log-likelihood. Surprisingly, SGLD and mean field VI result in better-calibrate models (looking at ECE). As we discuss in §2.3, this is not what we would have expected from MCMC methods. This might be due to the low mixing rate as we are limited to using a small step-size to collect accurate samples. As we would expect, pSGLD outperforms SGLD.

More on Acquisition Functions

It is also worth mentioning – the reader may keep in mind that the project does not focus on acquisition strategies, although it indirectly comes across interesting findings – that overall BALD remarkably outperformed max entropy. As already mentioned, this is found in Figure 3, as well. In Figure 5 the difference is far more accentuated: often max entropy ends up performing worse than random (see Figure 5). This shows across approximate inference methods that a more principled way of querying data-points results in better learning. In addition, as clearly shown in Table 7, looking only at accuracy is deceptive: from an accuracy standpoint most approximate inference methods behaves as deterministic. This is not the case if we consider predictive log-likelihood and ECE. In the same spirit as Table 2, Table 5 reports that "one-off" metrics do not allow such comparison.

Adding an Extra Layer

As we add a third hidden layer, as shown in Figure 6 (see Table 8), most of the inference network behave as deterministic, similarly to what we observed with MNIST (see Table 4). The only exceptions are matrix variate Gaussian VI and MCMC methods. We observe that VI with matrix variate Gaussian posteriors considerably outperforms the deterministic benchmark no matter the complexity of the dataset we use, or the fact that

Inference Method	#Layers/#Units	Negative Predictive LL	Error (%)	ECE
Mean Field Approx.	2/400	0.3014 ± 0.0108	10.8724 ± 0.3873	0.0226 ± 0.0022
MC Dropout	2/400	0.3112 ± 0.0097	11.2883 ± 0.2360	0.0229 ± 0.0008
VMG	2/400	0.2782 ± 0.0036	10.1092 ± 0.1489	0.0233 ± 0.0020
SGLD	2/400	0.2872 ± 0.0060	10.2069 ± 0.1108	0.0248 ± 0.0024
pSGLD	2/400	0.2988 ± 0.0068	10.3443 ± 0.1721	0.0269 ± 0.0019

Table 5: Average and std. test predictive log-likelihood (LL), test error, and test expected calibration error (ECE) (with M = 10 bins) for different approximate inference schemes. The network architecture is 2 hidden layer fully connected with 400 units each. The results are obtained on Fashion MNIST dataset consisting of 32×32 images from 10 different classes with 50,000 training, 10,000 validation, and 10,000 testing samples. We randomly crop and flip the images as a regularisation strategy. We average over 5 different runs. 400 epochs are used to guarantee convergence.

we increase the network capacity. Overall, a trade-off emerges between task complexity and network capacity. At constant capacity, increasing the dataset complexity leads to Bayesian networks outperforming the deterministic benchmarks (see Table 7). As we increase capacity, at constant complexity, being deterministic is just good enough (see Table 8).

Neural Linear Models On rougher ground, Neural Linear models sit near a sweet spot. Contrary to what we observed in Figure 4, Neural Linear models outperform in terms of predictive log-likelihood, and ECE the deterministic benchmark, as shown in Figure 7. Neural Linear models implementing VMG and MC dropout outperform mean field VI, in terms of predictive log-likelihood and ECE on both datasets. VMG outperforms MC dropout on Fashion MNIST in terms of final predictive log-likelihood and ECE (see Figure 7), in turn on SVHN MC dropout outperforms VMG in terms of predictive log-likelihood, but not ECE. However, it is worth mentioning that in terms of simplicity (i.e. hyper-parameters tuning, regularisation strategies, computational cost) VMG would not rank well, since the stochastic layer has an extra hyperparameter to be tuned (i.e. pseudo-inputs), a matrix inversion, and the optimisation is tricky. There is definitely room for improvements, such as investigating whether a regularisation technique of the variational parameters would ease the training, and make the architecture more stable[4].

[4]This was suggested by Christos Louizos from the Max Welling's group.

Dataset	Inference Method	Negative Predictive LL	Error (%)	ECE
Fashion MNIST				
	Mean Field Approx.	0.2099 ± 0.0030	7.6244 ± 0.1143	0.0220 ± 0.0021
	MC Dropout	0.2151 ± 0.0041	7.7944 ± 0.3299	0.0184 ± 0.0006
	VMG	0.2184 ± 0.0047	7.6317 ± 0.1083	0.0209 ± 0.0013
SVHN				
	Mean Field Approx.	0.2940 ± 0.0037	8.3804 ± 0.1586	0.0199 ± 0.0006
	MC Dropout	0.2987 ± 0.0048	8.6444 ± 0.0947	0.0188 ± 0.0019
	VMG	0.3057 ± 0.0009	8.9663 ± 0.0488	0.0220 ± 0.0018

Table 6: Average and std. test predictive log-likelihood (LL), test error, and test expected calibration error (ECE) (with M = 10 bins). We test Neural Linear architectures on Fashion MNIST and SVHN datasets. We average over 5 different runs. 200 epochs are used to guarantee convergence.

Inference method	Predictive LL*	Accuracy*	ECE*
Mean Field Approx.	0.0530 ± 0.0037	0.0069 ± 0.0010	0.0190 ± 0.0018
MC Dropout	0.0531 ± 0.0004	0.0072 ± 0.0007	0.0155 ± 0.0016
VMG	0.0477 ± 0.0063	**0.0321 ± 0.0015**	-0.0078 ± 0.0003
SGLD	0.0213 ± 0.0027	0.0003 ± 0.0010	0.0207 ± 0.0020
pSGLD	0.0266 ± 0.0172	0.0171 ± 0.0019	0.0031 ± 0.0005

Table 7: *A More Complex Dataset.* We report the area between the learning curves BALD and Det. Entropy.

Inference method	Predictive LL*	Accuracy*	ECE*
Mean Field Approx.	0.0001 ± 0.0071	-0.0138 ± 0.0014	0.0065 ± 0.0008
MC Dropout	0.0045 ± 0.0081	-0.0095 ± 0.0025	0.0003 ± 0.0036
VMG	0.0217 ± 0.0134	0.0145 ± 0.0031	-0.0076 ± 0.0011
SGLD	0.0212 ± 0.0100	0.0034 ± 0.0013	**0.0142 ± 0.0006**
pSGLD	0.0099 ± 0.0012	0.0134 ± 0.0011	-0.0072 ± 0.0030

Table 8: *A More Complex Dataset & Higher Network Capacity.* We report the area between the learning curves BALD and Det. Entropy.

Inference method	Predictive LL*	Accuracy*	ECE*
Mean Field Approx.	0.0591 ± 0.0113	0.0153 ± 0.0047	0.0114 ± 0.0011
MC Dropout	0.0711 ± 0.0067	0.0103 ± 0.0047	0.0165 ± 0.0005
VMG	0.0599 ± 0.0045	-0.0092 ± 0.0020	**0.0350 ± 0.0004**

Table 9: *Am I Better Than Deterministic? Part II - Fashion MNIST.* We report the area between the learning curves BALD and Det. Entropy.

Inference method	Predictive LL*	Accuracy*	ECE*
Mean Field Approx.	0.1259 ± 0.0259	0.0200 ± 0.0018	0.0192 ± 0.0050
MC Dropout	**0.2244 ± 0.0162**	0.0265 ± 0.0091	0.0455 ± 0.0005
VMG	0.1647 ± 0.0114	-0.0059 ± 0.0019	**0.0606 ± 0.0010**

Table 10: *Am I Better Than Deterministic? Part II - SVHN.* We report the area between the learning curves BALD and Det. Entropy.

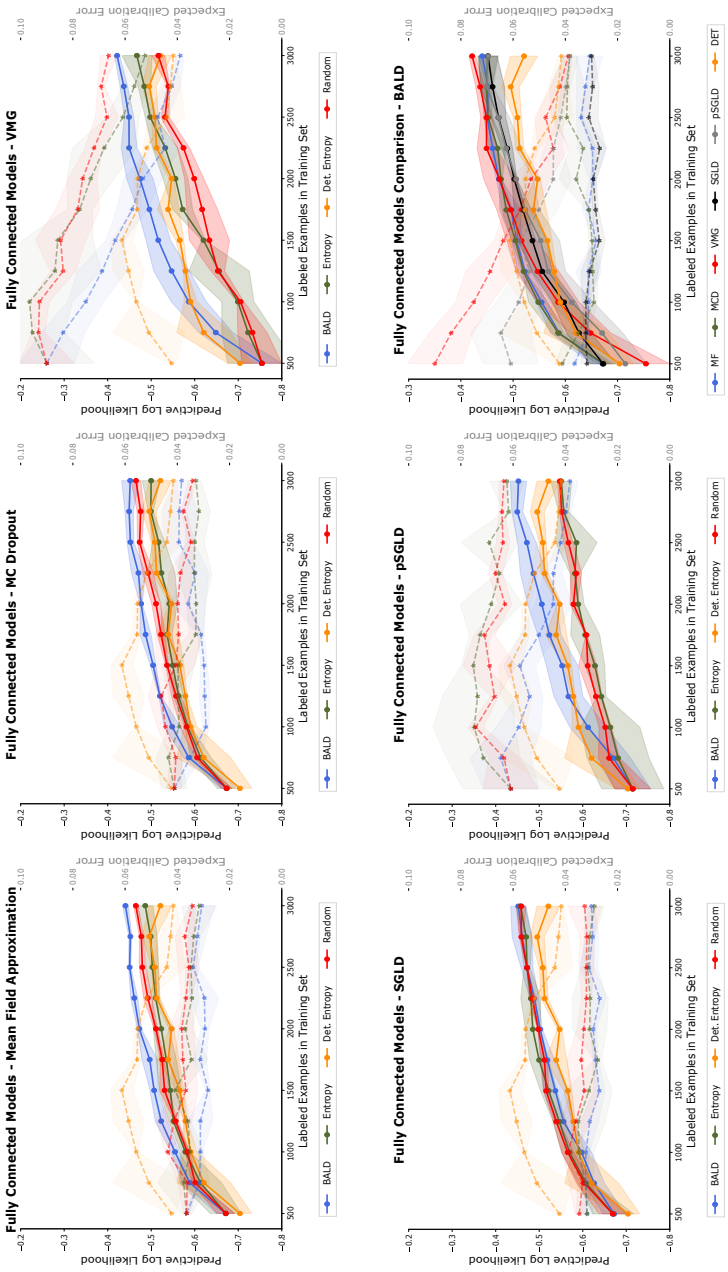


Figure 5: *A More Complex Dataset.* We investigate 5 different inference schemes using a 2 hidden layer fully connected network in an AL framework. We use the Fashion-MNIST dataset. We show predictive log-likelihood and ECE (dotted line) as a function of the # acquired images. We show AL for 2 acquisition functions: Max Entropy, and BALD. We report Random and Det. Entropy (short for deterministic Entropy) as benchmarks. Initially, we train on 500 labelled data-points, and progress in batches of 250 with a budget of 2500. At each acquisition step, the networks are re-trained for 400 epochs.

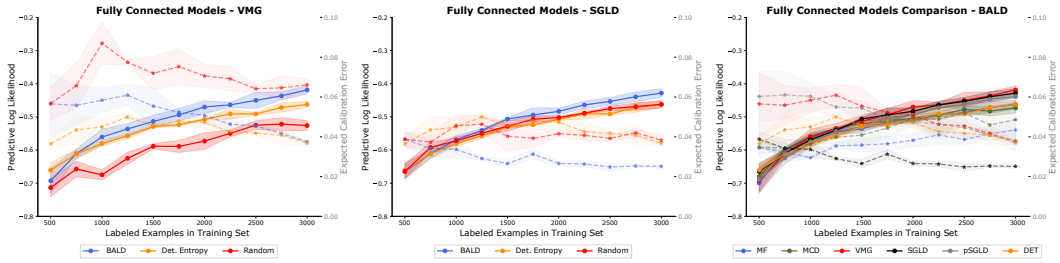


Figure 6: *"A More Complex Dataset & Higher Network Capacity"*. We investigate 5 different inference schemes using a 3 hidden layer fully connected network in an AL framework. We use the Fashion MNIST dataset. We only show predictive log-likelihood and ECE (dotted line) as a function of the # acquired images for VMG and SGLD. We show an inference methods comparison figure as well. Please refer to Appendix D for the complete figure. We show AL for one acquisition function: BALD. We report Random and Det. Entropy (short for deterministic Entropy) as benchmarks. Initially, we train on 500 labelled data-points, and progress in batches of 250 with a budget of 2500. At each acquisition step, the networks are re-trained for 400 epochs.

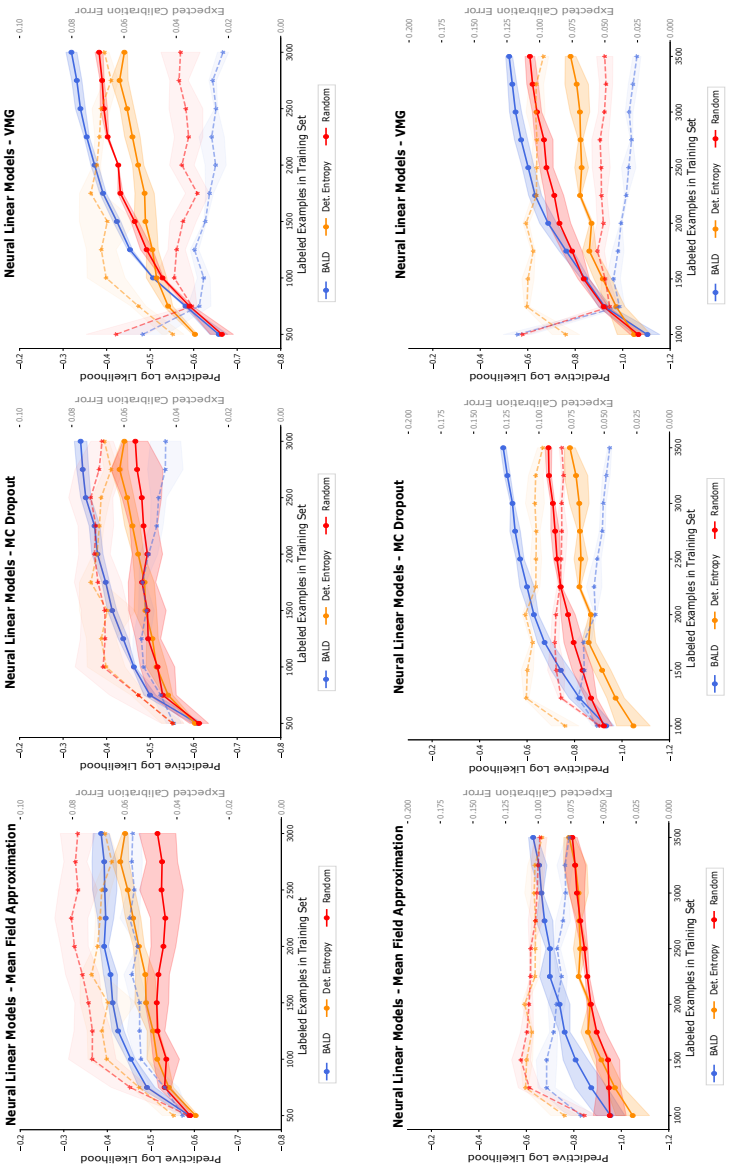


Figure 7: *Am I Better Than Deterministic? Part II.* We investigate the Neural Linear architecture, and use 3 different inference schemes. We show predictive log-likelihood and ECE (dotted line) as a function of the # acquired images. We show AL for one acquisition function: BALD. We report Random and Det. Entropy (short for deterministic Entropy) as benchmarks. *Top:* We use the Fashion-MNIST dataset. Initially, we train on 500 labelled data-points, and progress in batches of 250 with a budget of 2500. *Bottom:* We use the SVHN dataset. Initially, we train on 1000 labelled data-points, and progress in batches of 250 with a budget of 2500. At each acquisition step, the networks are re-trained for 400 epochs. For the VMG model, we reduce the number of pseudo-inputs to 50, and train initially for 400 epochs on a minibatch, and then at each acquisition step, the network is re-trained for 100 epochs (early-stopping).

Hybrid Models: an Additional Stochastic Layer By means of AL, we investigate whether we can improve the Neural Linear architecture (and whether being more Bayesian would lead to better performances in AL) by replacing the deterministic fully connected layer in the feature extractor with an additional stochastic component. We call the architecture Hybrid. We investigate VI with mean field approximation and with matrix variate Gaussian posteriors, and ultimately MC dropout. Figure 8 shows that Hybrid MC dropout outperforms the Neural Linear MC dropout on Fashion MNIST. Indeed, being more stochastic with VI does not lead to any improvement. This trend is not confirmed on SVHN, where being more stochastic does lead to improvements with respect to the the Neural Linear architecture (as benchmark). As reported in Table 11 and Table 12, MC dropout considerably outperforms the corresponding Neural Linear architecture on both datasets. With confidence, our analysis shows that MC dropout in Neural Linear architectures and Hybrid ones is the right inference to choose on account of low complexity, easy fine-tuning, and quality of uncertainty estimates. Again, AL proves to be the right ground to contrast inference methods and architectures.

Inference method	**Predictive LL***	**Accuracy***	**ECE***
Mean Field Approx.	-0.0001 $\pm$ 0.0061	-0.0085 $\pm$ 0.0017	0.0030 $\pm$ 0.0027
MC Dropout	**0.0267 $\pm$ 0.0039**	-0.0002 $\pm$ 0.0045	**0.0116 $\pm$ 0.0014**
VMG	0.0103 $\pm$ 0.0085	**0.0099 $\pm$ 0.0034**	-0.0165 $\pm$ 0.0035

Table 11: *Neural Linear Vs. Hybrid on Fashion MNIST.* We report the area between the Neural Linear and Hybrid models' learning curves. BALD is used as the acquisition function.

Inference method	**Predictive LL***	**Accuracy***	**ECE***
Mean Field Approx.	0.1109 $\pm$ 0.0275	0.0391 $\pm$ 0.0043	0.0100 $\pm$ 0.0042
MC Dropout	0.1272 $\pm$ 0.0206	0.0405 $\pm$ 0.0056	0.0110 $\pm$ 0.0013
VMG	0.0712 $\pm$ 0.0191	**0.0645 $\pm$ 0.0028**	-0.0317 $\pm$ 0.0021

Table 12: *Neural Linear Vs. Hybrid on SVHN.* We report the area between the Neural Linear and Hybrid models' learning curves. BALD is used as the acquisition function.

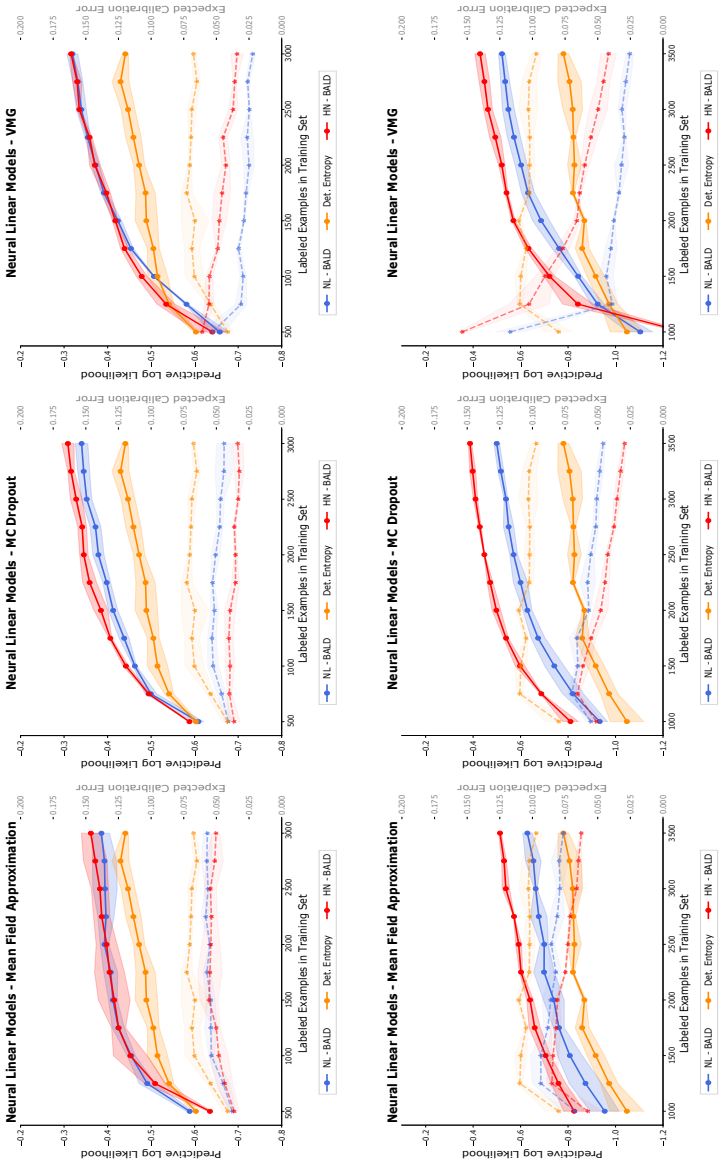


Figure 8: *Neural Linear Vs. Hybrid.* We compare the Neural Linear Vs. Hybrid, and use 3 different inferential schemes. We show predictive log-likelihood and ECE (dotted line) as a function of the # acquired images. Refer to Figure 7 for the experimental setup.

6 Conclusion

Our work demonstrated the adequacy of the AL framework for contrasting several approximate inference methods, alongside its promising application as an empirical measure of the quality of uncertainty estimation.
A significant difference between passive learning and the AL framework is that the latter plays on an information-based ground and thus makes direct use of uncertainty estimations. In §3, we demonstrated that passive learning did not allow a meaningful comparison between approximate inference methods, and their respective architectures. It did not gauge whether we were provided with sensible uncertainty estimates. Furthermore, our empirical evaluation (see Table 2) could not identify the analytical traits of the inferential methods discussed in §2.3.
On the other hand, in §5, we investigated fully stochastic fully connected architectures and hybrid models (Neural Linear and Hybrid Networks) via AL on 3 datasets: MNIST, Fashion MNIST, and SVHN. We were able to disambiguate the comparison among both inferences and architectures. As VMG outperforms the others, we have a clear frontrunner across fully connected models if we base the comparison on the evaluation metrics at the end of the acquisition process. Across Neural Linear architectures, MC dropout and VMG inference methods both outperformed mean-field VI, with the latter leading to overall better-calibrated models on both Fashion MNIST and SVHN (see Figure 7).
Being Bayesian comes at a price. The AL framework helped us to understand the effect of considering uncertainty. Overall, on the basis of decision-making processes, neural networks benefitted from being Bayesian, but not on all the environments we tested. Being Bayesian did not necessarily bring advantages over being deterministic, as shown in Figure 4. Through the AL framework, we identified a trend between task complexity, network capacity, and inference choice. We observed that there is no advantage using probabilistic models when the task is too simple (or the network capacity is too high). It found that as we scale up to more difficult environments, uncertainty estimation plays a more decisive role.
In light of these observations, arguably this work showed AL's effectiveness when it comes to providing recommendations of which architecture and inference method to use.

6.1 Future Directions

In this section, we discuss some exciting future directions. As a general direction, we believe that an analogous investigation should be repeated on a broader variety of approximate inference methods.
It follows a more in-depth analysis of 2 topics that we believe are good avenues for promising extensions. In §6.1.1, we suggest investigating whether there is a more principled metric to summarise the model performances in AL. In §6.1.2, we present some preliminary thoughts on how to decouple uncertainty in the epistemic and the aleatoric components.

6.1.1 Towards More Expressive Metrics in Active Learning

In this work, to ease the comparison among inferential methods, we propose a metric that concisely summarises the learning. We estimate the area between learning curves

throughout the acquisition process. However, this metric has a few drawbacks that need to be considered. This metric weights the history of the learning more than the performance as we reach the budget. Networks that at the beginning of the acquisition process start well, are favoured. Our metric often is inconclusive to disambiguate the comparison (see Table 7 and Table 8), and just looking at the learning curves gives more meaningful insights. There is definitely room for improvement for a more principled way to summarise the learning.

6.1.2 Expanding on Uncertainty Evaluation

Building up on §5.2, we would like to leave the reader with this query. Do we not perform better than deterministic methods despite well-calibrated uncertainty estimates? Or, do we not outperform deterministic methods as a consequence of badly-calibrated uncertainty estimates? Answering these questions is the natural progression of this work. In Appendix C we report our initial attempt.

References

Atkinson, A. C., D. A. N. (1992), 'Optimum experimental designs', *Oxford Statistical Science Series, Oxford University Press, Oxford* **8**.

Bae, J., Zhang, G. & Grosse, R. (2018), 'Eigenvalue corrected noisy natural gradient', *arXiv preprint arXiv:1811.12565* .

Barber, D., Cemgil, A. T. & Chiappa, S. (2011), *Bayesian time series models*, Cambridge University Press.

Bengio, Y., Ducharme, R., Vincent, P. & Jauvin, C. (2003), 'A neural probabilistic language model', *Journal of machine learning research* **3**(Feb), 1137–1155.

Bishop, C. M. (2006), *Pattern recognition and machine learning*, springer.

Blundell, C., Cornebise, J., Kavukcuoglu, K. & Wierstra, D. (2015), 'Weight uncertainty in neural networks', *arXiv preprint arXiv:1505.05424* .

Chaloner, K. & Verdinelli, I. (1995), 'Bayesian experimental design: A review', *Statistical Science* pp. 273–304.

Chen, T., Fox, E. & Guestrin, C. (2014), Stochastic gradient hamiltonian monte carlo, *in* 'International conference on machine learning', pp. 1683–1691.

Cohn, D. A., Ghahramani, Z. & Jordan, M. I. (1996), 'Active learning with statistical models', *Journal of artificial intelligence research* **4**, 129–145.

Depeweg, S., Hernández-Lobato, J. M., Doshi-Velez, F. & Udluft, S. (2017), 'Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning', *arXiv preprint arXiv:1710.07283* .

Foong, A. Y., Li, Y., Hernández-Lobato, J. M. & Turner, R. E. (2019), ''in-between'uncertainty in bayesian neural networks', *arXiv preprint arXiv:1906.11537* .

Friston, K., Mattout, J., Trujillo-Barreto, N., Ashburner, J. & Penny, W. (2007), 'Variational free energy and the laplace approximation', *Neuroimage* **34**(1), 220–234.

Gal, Y. (2016), Uncertainty in deep learning, PhD thesis.

Gal, Y. & Ghahramani, Z. (2015), Dropout as a bayesian approximation: Insights and applications.

Gal, Y. & Ghahramani, Z. (2016), Dropout as a bayesian approximation: Representing model uncertainty in deep learning, *in* 'international conference on machine learning', pp. 1050–1059.

Gal, Y., Islam, R. & Ghahramani, Z. (2017), Deep bayesian active learning with image data, *in* 'Proceedings of the 34th International Conference on Machine Learning-Volume 70', JMLR. org, pp. 1183–1192.

Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S. & Shet, V. (2013), 'Multi-digit number recognition from street view imagery using deep convolutional neural networks', *arXiv preprint arXiv:1312.6082* .

Graves, A. (2011), Practical variational inference for neural networks, *in* 'Advances in neural information processing systems', pp. 2348–2356.

Guo, C., Pleiss, G., Sun, Y. & Weinberger, K. Q. (2017), On calibration of modern neural networks, *in* 'Proceedings of the 34th International Conference on Machine Learning-Volume 70', JMLR. org, pp. 1321–1330.

Gupta, A. K. & Nagar, D. K. (2018), *Matrix variate distributions*, Chapman and Hall/CRC.

He, K., Zhang, X., Ren, S. & Sun, J. (2015), Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, *in* 'Proceedings of the IEEE international conference on computer vision', pp. 1026–1034.

Hernández-Lobato, J. M. & Adams, R. (2015), Probabilistic backpropagation for scalable learning of bayesian neural networks, *in* 'International Conference on Machine Learning', pp. 1861–1869.

Hinton, G. & Van Camp, D. (1993), Keeping neural networks simple by minimizing the description length of the weights, *in* 'in Proc. of the 6th Ann. ACM Conf. on Computational Learning Theory', Citeseer.

Houlsby, N., Huszár, F., Ghahramani, Z. & Lengyel, M. (2011), 'Bayesian active learning for classification and preference learning', *arXiv preprint arXiv:1112.5745* .

Ioffe, S. & Szegedy, C. (2015), 'Batch normalization: Accelerating deep network training by reducing internal covariate shift', *arXiv preprint arXiv:1502.03167* .

Jankowiak, M. & Obermeyer, F. (2018), 'Pathwise derivatives beyond the reparameterization trick', *arXiv preprint arXiv:1806.01851* .

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S. & Saul, L. K. (1999), 'An introduction to variational methods for graphical models', *Machine learning* **37**(2), 183–233.

Kendall, A. & Gal, Y. (2017), What uncertainties do we need in bayesian deep learning for computer vision?, *in* 'Advances in neural information processing systems', pp. 5574–5584.

Kiefer, J., Wolfowitz, J. et al. (1952), 'Stochastic estimation of the maximum of a regression function', *The Annals of Mathematical Statistics* **23**(3), 462–466.

Kingma, D. P., Salimans, T. & Welling, M. (2015), Variational dropout and the local reparameterization trick, *in* 'Advances in Neural Information Processing Systems', pp. 2575–2583.

Kingma, D. P. & Welling, M. (2013), 'Auto-encoding variational bayes', *arXiv preprint arXiv:1312.6114* .

Kirsch, A., van Amersfoort, J. & Gal, Y. (2019), 'Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning', *CoRR* **abs/1906.08158**.
URL: *http://arxiv.org/abs/1906.08158*

Kullback, S. & Leibler, R. A. (1951), 'On information and sufficiency', *The annals of mathematical statistics* **22**(1), 79–86.

Kwon, Y., Won, J.-H., Kim, B. J. & Paik, M. C. (2018), 'Uncertainty quantification using bayesian neural networks in classification: Application to ischemic stroke lesion segmentation'.

Lawrence, N. D. (2001), Variational inference in probabilistic models, PhD thesis.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. et al. (1998), 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE* **86**(11), 2278–2324.

Li, C., Chen, C., Carlson, D. & Carin, L. (2016), Preconditioned stochastic gradient langevin dynamics for deep neural networks, *in* 'Thirtieth AAAI Conference on Artificial Intelligence'.

Louizos, C. & Welling, M. (2016), Structured and efficient variational deep learning with matrix gaussian posteriors, *in* 'International Conference on Machine Learning', pp. 1708–1716.

MacKay, D. J. (1992), 'A practical bayesian framework for backpropagation networks', *Neural computation* **4**(3), 448–472.

Marceau-Caron, G. & Ollivier, Y. (2017), Natural langevin dynamics for neural networks, *in* 'International Conference on Geometric Science of Information', Springer, pp. 451–459.

Melis, G., Dyer, C. & Blunsom, P. (2017), 'On the state of the art of evaluation in neural language models', *arXiv preprint arXiv:1707.05589* .

Mitchell, T. J. & Beauchamp, J. J. (1988), 'Bayesian variable selection in linear regression', *Journal of the American Statistical Association* **83**(404), 1023–1032.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. & Riedmiller, M. (2013), 'Playing atari with deep reinforcement learning', *arXiv preprint arXiv:1312.5602* .

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G. et al. (2015), 'Human-level control through deep reinforcement learning', *Nature* **518**(7540), 529.

Nagapetyan, T., Duncan, A. B., Hasenclever, L., Vollmer, S. J., Szpruch, L. & Zygalakis, K. (2017), 'The true cost of stochastic gradient langevin dynamics', *arXiv preprint arXiv:1706.02692* .

Nair, V. & Hinton, G. E. (2010), Rectified linear units improve restricted boltzmann machines, *in* 'Proceedings of the 27th international conference on machine learning (ICML-10)', pp. 807–814.

Neal, R. M. (2012), *Bayesian learning for neural networks*, Vol. 118, Springer Science & Business Media.

Neal, R. M. & Hinton, G. E. (1998), A view of the em algorithm that justifies incremental, sparse, and other variants, *in* 'Learning in graphical models', Springer, pp. 355–368.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. & Kavukcuoglu, K. (2016), 'Wavenet: A generative model for raw audio', *arXiv preprint arXiv:1609.03499* .

Paisley, J., Blei, D. & Jordan, M. (2012), 'Variational bayesian inference with stochastic search', *arXiv preprint arXiv:1206.6430* .

Rezende, D. J., Mohamed, S. & Wierstra, D. (2014), 'Stochastic backpropagation and approximate inference in deep generative models', *arXiv preprint arXiv:1401.4082* .

Riquelme, C., Tucker, G. & Snoek, J. (2018), 'Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling', *arXiv preprint arXiv:1802.09127* .

Ritter, H., Botev, A. & Barber, D. (2018), 'A scalable laplace approximation for neural networks'.

Robbins, H. & Monro, S. (1951), 'A stochastic approximation method', *The annals of mathematical statistics* pp. 400–407.

Robert, C. & Casella, G. (2013), *Monte Carlo statistical methods*, Springer Science & Business Media.

Robert Pinsler, Jonathan Gordon, E. N. H.-L. J. M. (2019), 'Bayesian batch active learning as sparse subset approximation', *arXiv preprint arXiv:1908.02144* .

Rowley, H. A. (1999), Neural network-based face detection, Technical report, Carnegie-Mellon Univ. Pittsburgh PA Dept. of Computer Science.

Shannon, C. E. (1948), 'A mathematical theory of communication', *Bell system technical journal* **27**(3), 379–423.

Shridhar, K., Laumann, F. & Liwicki, M. (2018), 'Uncertainty estimations by softplus normalization in bayesian convolutional neural networks with variational inference', *arXiv preprint arXiv:1806.05978* .

Simard, P. Y., Steinkraus, D., Platt, J. C. et al. (2003), Best practices for convolutional neural networks applied to visual document analysis.

Snelson, E. & Ghahramani, Z. (2006), Sparse gaussian processes using pseudo-inputs, *in* 'Advances in neural information processing systems', pp. 1257–1264.

Srivastava, N. (2013), 'Improving neural networks with dropout', *University of Toronto* **182**, 566.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014), 'Dropout: a simple way to prevent neural networks from overfitting', *The journal of machine learning research* **15**(1), 1929–1958.

Sun, S., Chen, C. & Carin, L. (2017), Learning structured weight uncertainty in bayesian neural networks, *in* 'Artificial Intelligence and Statistics', pp. 1283–1292.

Sun, S., Zhang, G., Shi, J. & Grosse, R. (2019), 'Functional variational bayesian neural networks', *arXiv preprint arXiv:1903.05779* .

Teh, Y. W., Thiery, A. H. & Vollmer, S. J. (2016), 'Consistency and fluctuations for stochastic gradient langevin dynamics', *The Journal of Machine Learning Research* **17**(1), 193–225.

Titsias, M. & Lázaro-Gredilla, M. (2014), Doubly stochastic variational bayes for non-conjugate inference, *in* 'International conference on machine learning', pp. 1971–1979.

Wei, J. N., Duvenaud, D. & Aspuru-Guzik, A. (2016), 'Neural networks for the prediction of organic chemistry reactions', *ACS central science* **2**(10), 725–732.

Welling, M. & Teh, Y. W. (2011), Bayesian learning via stochastic gradient langevin dynamics, *in* 'Proceedings of the 28th international conference on machine learning (ICML-11)', pp. 681–688.

Xiao, H., Rasul, K. & Vollgraf, R. (2017), 'Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms', *arXiv preprint arXiv:1708.07747* .

Yedidia, J. S., Freeman, W. T. & Weiss, Y. (2001), Generalized belief propagation, *in* 'Advances in neural information processing systems', pp. 689–695.

Zhang, G., Sun, S., Duvenaud, D. & Grosse, R. (2017), 'Noisy natural gradient as variational inference', *arXiv preprint arXiv:1712.02390* .

Zhu, X. (2005), Semi-supervised learning with graphs, PhD thesis.

A Additional Theoretical Background

KL Divergence between Matrix Variate Gaussians

For clarity, we re-propose the derivation by Louizos & Welling (2016). Let's recast $\mathcal{MN}_0(\boldsymbol{M}_0, \boldsymbol{U}_0, \boldsymbol{V}_0)$, and $\mathcal{MN}_1(\boldsymbol{M}_1, \boldsymbol{U}_1, \boldsymbol{V}_1)$ as $\mathcal{N}_0(\text{vec}(\boldsymbol{M}_0), \boldsymbol{U}_0 \otimes \boldsymbol{V}_0)$ and $\mathcal{N}_1(\text{vec}(\boldsymbol{M}_1), \boldsymbol{U}_1 \otimes \boldsymbol{V}_1)$, respectively. The KL between the two multivariate Gaussians is defined as:

$$
\begin{aligned}
KL(\mathcal{N}_0 \| \mathcal{N}_1) &= \frac{1}{2}(\text{tr}(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) - K + \log \frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_0|}) \\
&= \frac{1}{2}(\text{tr}((\mathbf{V}_1 \otimes \mathbf{U}_1)^{-1}(\mathbf{V}_0 \otimes \mathbf{U}_0)) + (\text{vec}(\mathbf{M}_1) - \text{vec}(\mathbf{M}_1) - \text{vec}(\mathbf{M}_0)) - \\
&- (\mathbf{V}_1 \otimes \mathbf{U}_1)^{-1} (\text{vec}(\mathbf{M}_1) - \text{vec}(\mathbf{M}_0)) - np + \log \frac{|\mathbf{V}_1 \otimes \mathbf{U}_1|}{|\mathbf{V}_0 \otimes \mathbf{U}_0|})
\end{aligned}
$$

Splitting it into three sub-terms t_a, t_b, t_c, and applying properties of the Kronecker products,

$$
\begin{aligned}
t_a &= \text{tr}\left((\mathbf{V}_1 \otimes \mathbf{U}_1)^{-1} (\mathbf{V}_0 \otimes \mathbf{U}_0)\right) \\
&= \text{tr}\left((\mathbf{V}_1^{-1} \otimes \mathbf{U}_1^{-1}) (\mathbf{V}_0 \otimes \mathbf{U}_0)\right) \\
&= \text{tr}\left((\mathbf{V}_1^{-1}\mathbf{V}_0) \otimes (\mathbf{U}_1^{-1}\mathbf{U}_0)\right) \\
&= \text{tr}\left(\mathbf{U}_1^{-1}\mathbf{U}_0\right) \text{tr}\left(\mathbf{V}_1^{-1}\mathbf{V}_0\right) \\
t_b &= (\text{vec}(\mathbf{M}_1) - \text{vec}(\mathbf{M}_0))^T (\mathbf{V}_1 \otimes \mathbf{U}_1)^{-1} \\
&\quad (\text{vec}(\mathbf{M}_1) - \text{vec}(\mathbf{M}_0)) \\
&= \text{vec}(\mathbf{M}_1 - \mathbf{M}_0)^T \left(\mathbf{V}_1^{-1} \otimes \mathbf{U}_1^{-1}\right) \text{vec}(\mathbf{M}_1 - \mathbf{M}_0) \\
&= \text{vec}(\mathbf{M}_1 - \mathbf{M}_0)^T \text{vec}\left(\mathbf{U}_1^{-1} (\mathbf{M}_1 - \mathbf{M}_0) \mathbf{V}_1^{-1}\right) \\
&= \text{tr}\left((\mathbf{M}_1 - \mathbf{M}_0)^T \mathbf{U}_1^{-1} (\mathbf{M}_1 - \mathbf{M}_0) \mathbf{V}_1^{-1}\right) \\
t_c &= \log \frac{|\mathbf{V}_1 \otimes \mathbf{U}_1|}{|\mathbf{V}_0 \otimes \mathbf{U}_0|} \\
&= \log \frac{|\mathbf{U}_1|^p |\mathbf{V}_1|^n}{|\mathbf{U}_0|^p |\mathbf{V}_0|^n} \\
&= p \log |\mathbf{U}_1| + n \log |\mathbf{V}_1| - \\
&- p \log |\mathbf{U}_0| - n \log |\mathbf{V}_0|
\end{aligned}
$$

Combining them all together,

$$
\begin{aligned}
KL(\mathcal{MN}_0, \mathcal{MN}_1) = \frac{1}{2} &\left(\text{tr}\left(\mathbf{U}_1^{-1}\mathbf{U}_0\right) \text{tr}\left(\mathbf{V}_1^{-1}\mathbf{V}_0\right) + \right. \\
&+ \text{tr}\left((\mathbf{M}_1 - \mathbf{M}_0)^T \mathbf{U}_1^{-1} (\mathbf{M}_1 - \mathbf{M}_0) \mathbf{V}_1^{-1}\right) - \\
&- np + p \log |\mathbf{U}_1| + n \log |\mathbf{V}_1| - \\
&\left. - p \log |\mathbf{U}_0| - n \log |\mathbf{V}_0|\right)
\end{aligned}
$$

B Preliminary Investigations on a 2D Dataset

A Simple 2D Dataset

In this section, we contrast different approximate inference methods using a 2D toy dataset. A 2D space is straightforwardly displayable, and we can track all the steps in the AL process. We use the moons dataset, a simple binary classification task. Each class is generated from a deterministic function with additive Gaussian noise ($\sigma = 0.2$). We use single hidden fully connected architecture with 25 units. The aim of this experimental session is to visualise the AL process, as well to gain a clear understanding of how the different inference methods behave. Therefore, we evaluate the class probability predictions on a grid over the whole input space at each acquisition step. We report the history of the probability surface throughout the AL procedure. Initially, we train on 30 data-points (15 for each class), and we progress either in steps of 1 or 5, with a budget of 10, or 50, accordingly. Figure 9, and Figure 10 show the results of our investigation. As a general remark, it is possible to observe that at the beginning of the acquisition process, Bayesian networks are prone to enlarge the area which they are most uncertain about. Overall, Bayesian networks show well-calibrated uncertainty estimation: mean field VI and MC dropout report wider uncertainty compared to VMG and SGLD. It is inferred that VI with matrix variate Gaussian posteriors, and the sampling method Stochastic Gradient Langevin Dynamics outperform the others.

A More Principled Way of Batching

If we have a closer look at Figure 10, it is clear that there is significant room for improvement in batch-mode AL. We greedily select the most informative data-points (see Algorithm 2). However, our greedy selection might not be the most informative jointly. Not surprisingly, the queried data-points happen to be close to each other (and on a 2D space this is easily visible!). Individually, the data-points are highly informative, but arguably more spaced out draws would have resulted in better uncertainty estimation (i.e. better coverage of the gap between the 2 moons). In fact, we are not taking into account (our naive way of batching) the correlation between data-points in any acquisition batch. This is discussed in (Kirsch et al., 2019).

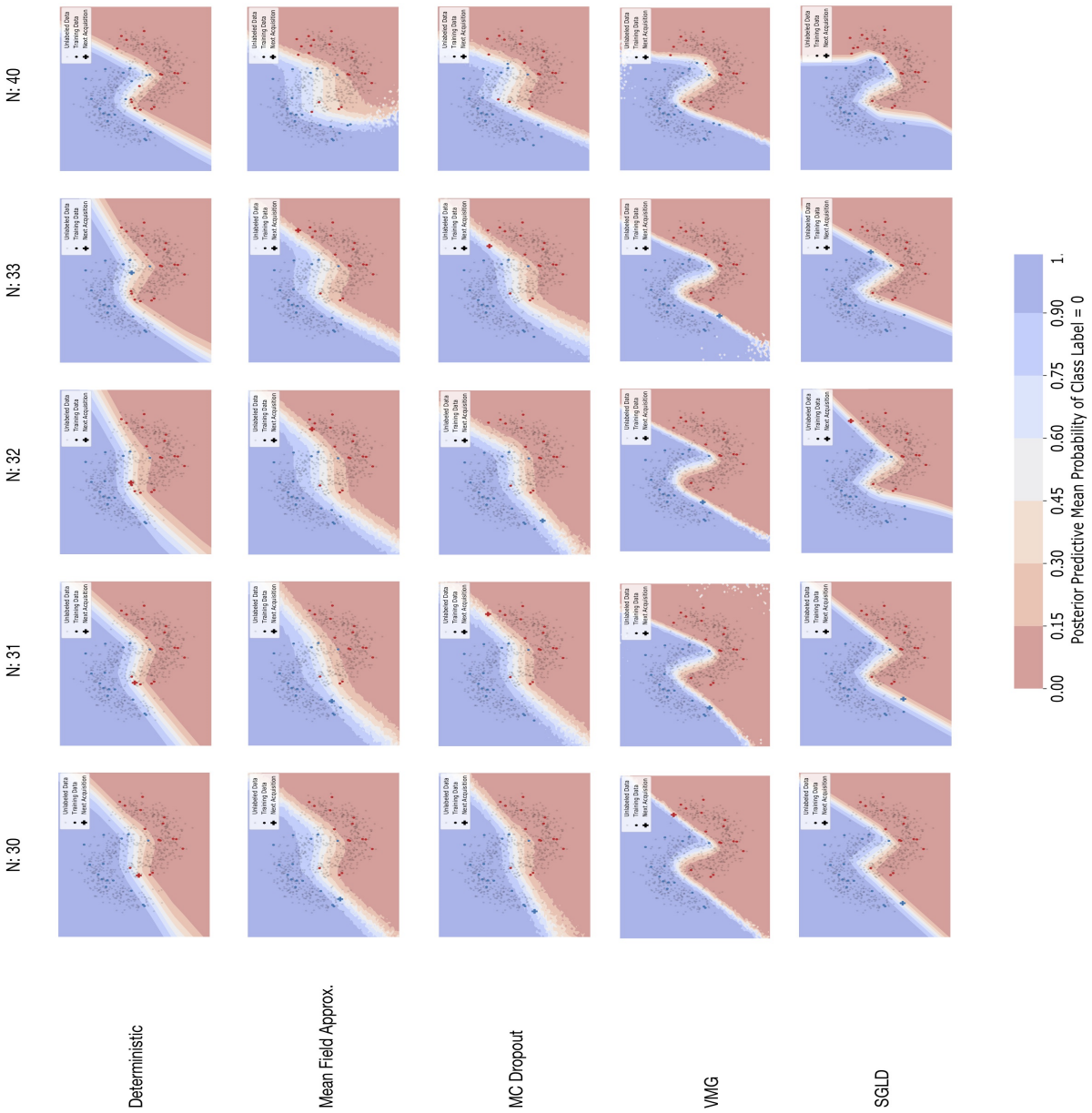


Figure 9: *The Moons Dataset & Probability Surface: Single Query.* We train a single hidden fully connected network with 25 hidden units initially on 30 data-points. We query 1 data-point at a time. Overall, we make 10 queries. We investigate 4 inference methods, along with a deterministic network. We report the mean of the posterior predictive. In the deterministic case, it is identical to maximum likelihood predicted values. We report the first 4 draws, and the final one. Queried data-points are displayed as crosses.

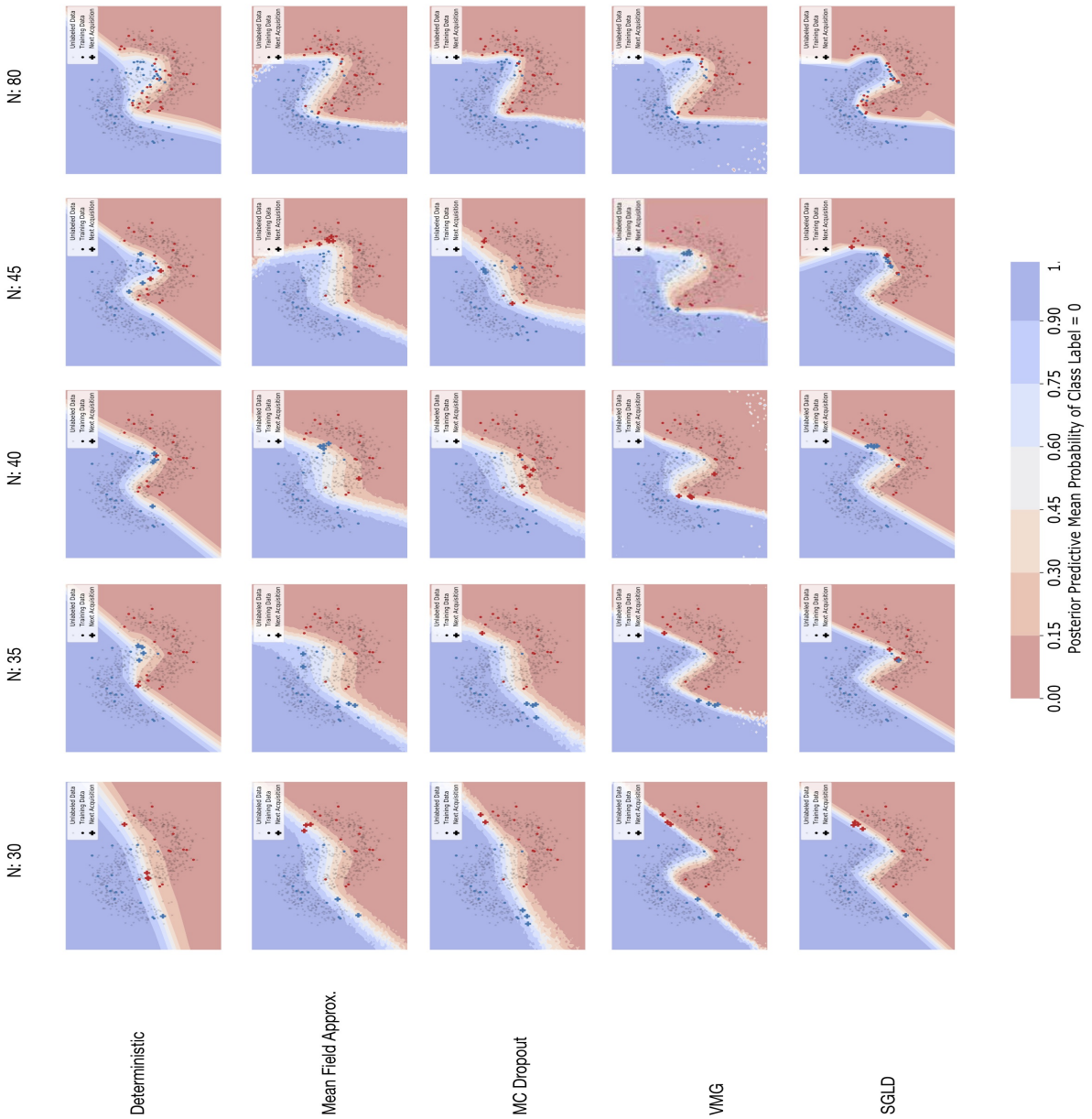


Figure 10: *The Moons Dataset & Probability Surface: 5 Queries.* We train a single hidden fully connected network with 25 hidden units initially on 30 data-points. We query 5 data-point at a time. Overall, we make 10 queries. We investigate 4 inference methods, along with a deterministic network. We report the mean of the posterior predictive. We report the first 4 draws, and the final one. Queried data-points are displayed as crosses.

C Uncertainty Decomposition in Active Learning

Decoupling Uncertainty: a Primer

In this section, we report some thoughts and preliminary analysis on epistemic uncertainty. We focused on fully connected architectures, and we contrasted VI with mean field approximation to VI with matrix variate Gaussian posteriors on both MNIST and Fashion MNIST. This experiment aimed to provide insights into the fact that VI with mean field failed to outperform the deterministic benchmark on MNIST (see Table 4). Indeed, on a more complex ground, it "beats deterministic" (see Table 7). This was also to empirically correlate active learning's performance to the modelling of epistemic uncertainty. As reported in (Kwon et al., 2018), we decomposed the variance of the variational predictive distribution $\hat{q}_{\theta(y^*|x^*)}$ into aleatoric and epistemic uncertainty:

$$\begin{aligned}\mathrm{Var}_{\hat{q}_\theta(y^*|x^*)}\left[p\left(y^*|x^*\right)\right] &= \mathbf{E}_{\hat{q}_\theta(y^*|x^*)}\left[y^{*\otimes 2}\right] - \mathbf{E}_{\hat{q}_\theta(y^*|x^*)}\left[y^*\right]^{\otimes 2} \\ &= \underbrace{\int_\Omega \left[\mathrm{diag}\left(\mathbf{E}_{p(y^*|x^*,\boldsymbol{\omega})}\left[y^*\right]\right) - \mathbf{E}_{p(y^*|x^*,\boldsymbol{\omega})}\left[y^*\right]^{\otimes 2}\right] q_\theta(\boldsymbol{\omega}) d\boldsymbol{\omega}}_{\text{aleatoric}} + \\ &+ \underbrace{\int_\Omega \left(\mathbf{E}_{p(y^*|x^*,\boldsymbol{\omega})}\left[y^*\right] - \mathbf{E}_{\hat{q}_\theta(y^*|x^*)}\left[y^*\right]\right)^{\otimes 2} q_\theta(\boldsymbol{\omega}) d\boldsymbol{\omega}}_{\text{epistemic}}\end{aligned}$$

where $v^{\otimes 2} = vv^T$ and $\mathrm{diag}(v)$ is a diagonal matrix with the elements of v. The first line is the definition of variance, the second is from a variant of the law of total variance. A detailed derivation is reported in (Kwon et al., 2018). The first term estimates the inherent variability of the output y^*, while the second weights the variability over $\boldsymbol{\omega}$. The latter can be explained away with more data (Kendall & Gal, 2017). From a Bayesian standpoint, the predictive variance is estimated as

$$\underbrace{\frac{1}{T}\sum_{t=1}^{T}\mathrm{diag}\left(\hat{p}_t\right) - \hat{p}_t^{\otimes 2}}_{\text{aleatoric}} + \underbrace{\frac{1}{T}\sum_{t=1}^{T}\left(\hat{p}_t - \bar{p}\right)^{\otimes 2}}_{\text{epistemic}}$$

where $\bar{p} = \frac{1}{T}\sum_{t=1}^{T}\hat{p}_t$ and $\hat{p}_t = \mathrm{Softmax}(\mathbf{f}^{\boldsymbol{\omega}_t}(x^*))$, where $\boldsymbol{\omega}_t$ is the t-th realised network parameter set $\{\hat{\boldsymbol{\omega}}_t\}_{t=1}^T$. As suggested in (Shridhar et al., 2018), we compute homoscedastic epistemic uncertainty by averaging over all classes (i.e. over heteroscedastic uncertainties).

More Thoughts & Observations

Our preliminary investigation, which were difficult to interpret, reported that VI with mean field on MNIST increased its epistemic uncertainty as it acquired more data. Conversely, epistemic uncertainty stayed bounded in Fashion MNIST. High epistemic uncertainty might explain the poor performance on MNIST (see Figure 3). However, these are provisional remarks and a more principled investigation is needed.

D Additional Experimental Figures

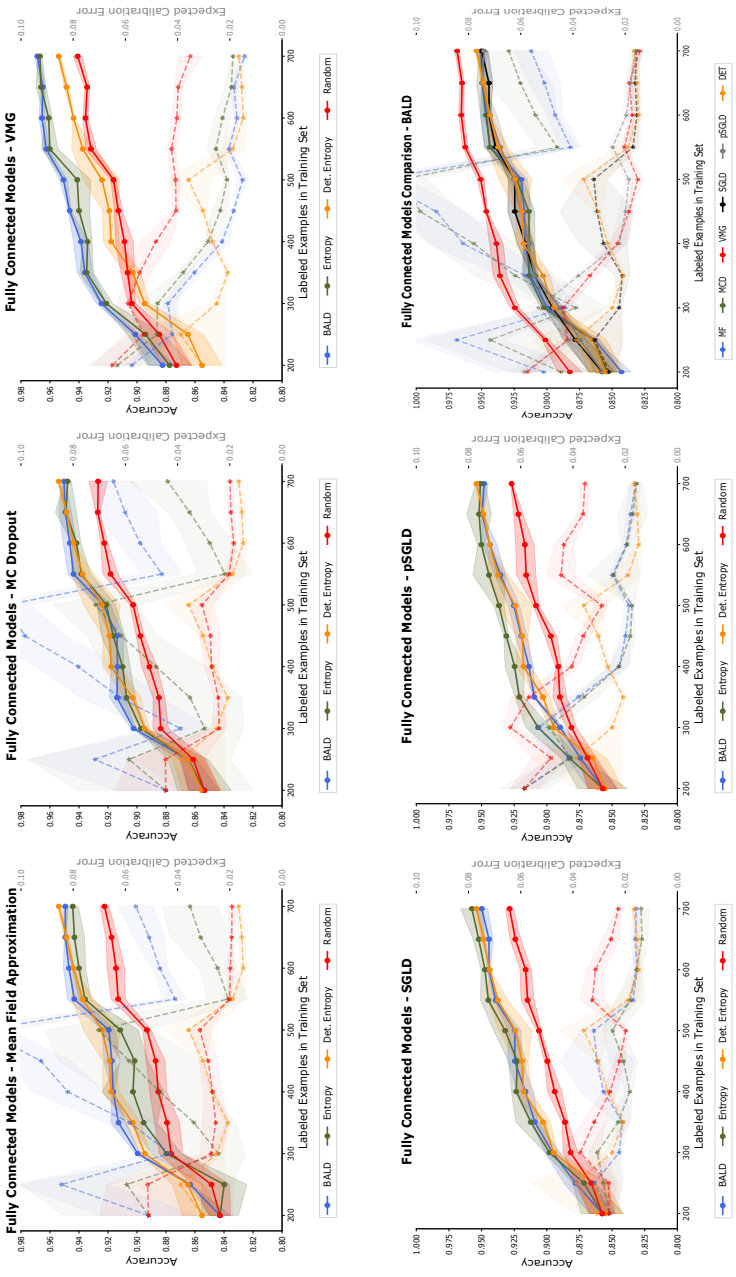


Figure 11: *A More Thoughtful Investigation.* We investigate 5 different inference schemes using a 2 hidden layer fully connected network in an active learning framework. We use MNIST dataset. We show accuracy and ECE (dotted line) as a function of the # acquired images. Initially, we train on 200 labelled data-points, and progress in batches of 50 with a budget of 200.

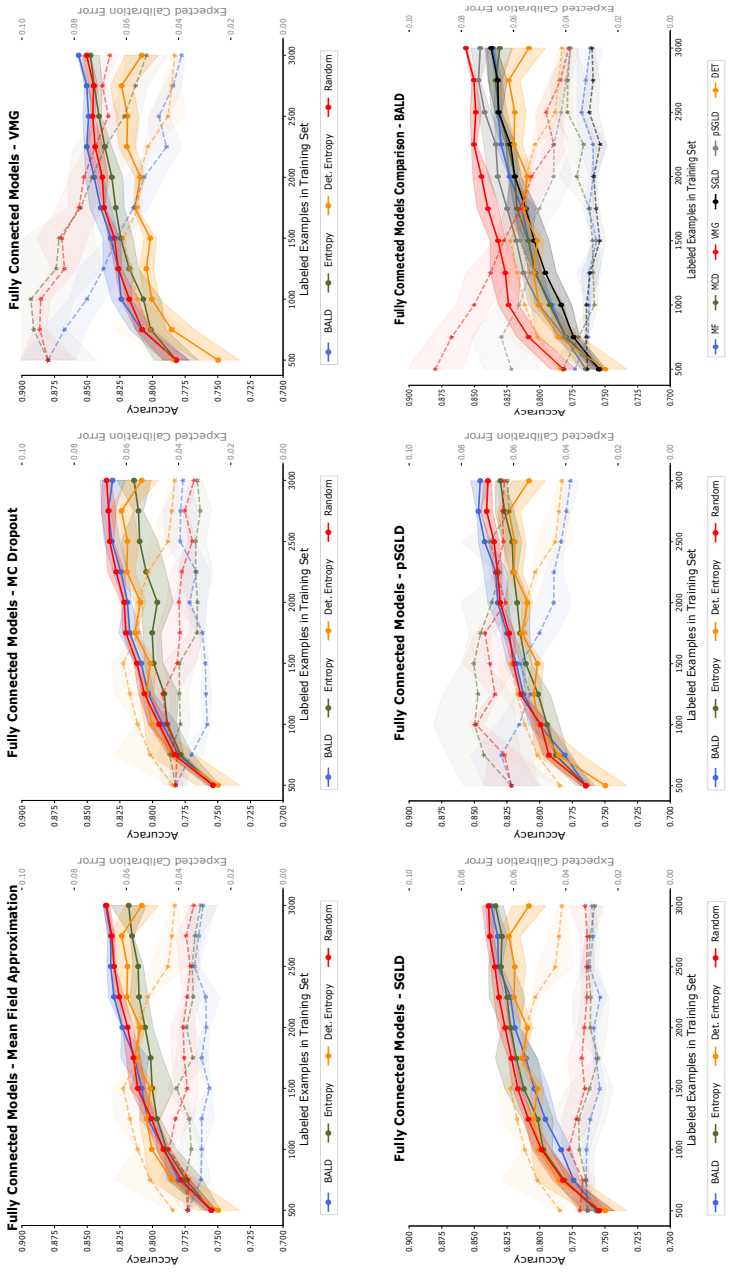


Figure 12: *A More Complex Dataset.* We investigate 5 different inference schemes using a 2 hidden layer fully connected network in an active learning framework. We use Fashion-MNIST dataset. We show accuracy and ECE (dotted line) as a function of the # acquired images. Initially, we train on 200 labelled data-points, and progress in batches of 250 with a budget of 2500.

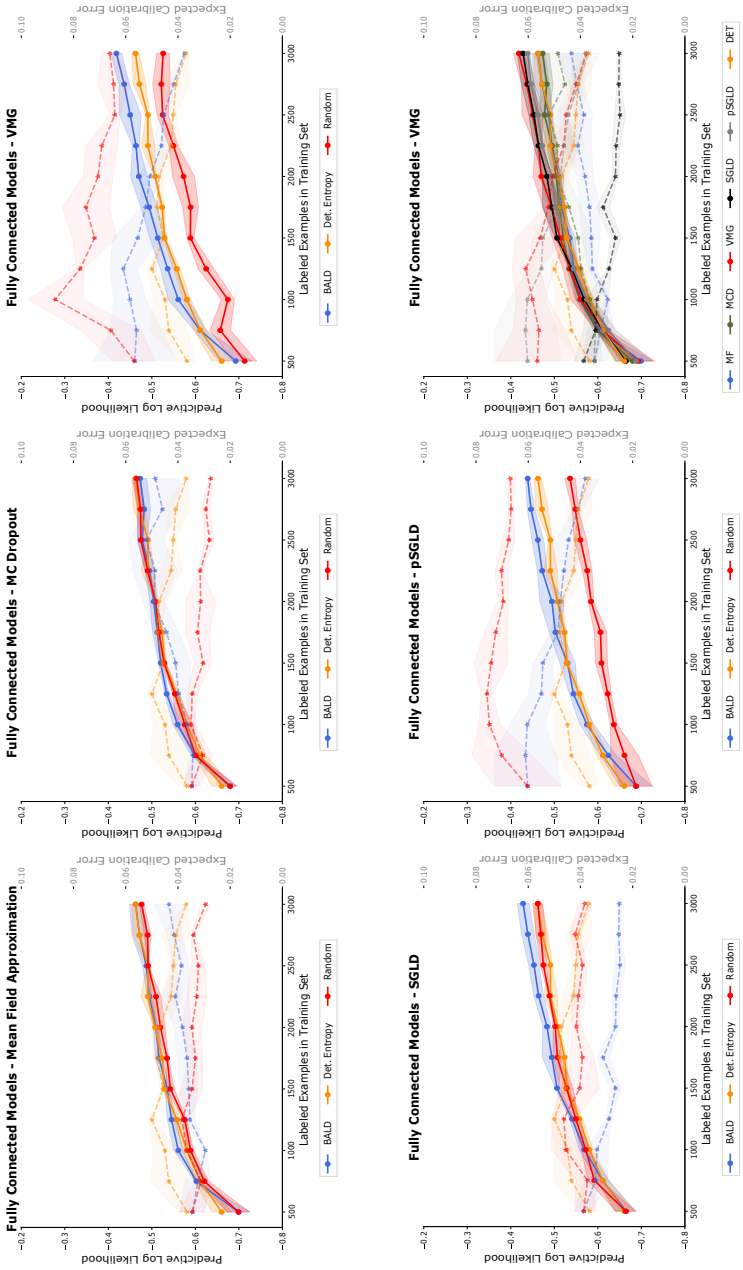


Figure 13: *"A More Complex Dataset & Higher Network Capacity"*. We investigate 5 different inference schemes using a 3 hidden layer fully connected network in an active learning framework. We use Fashion MNIST dataset. We show predictive log-likelihood and ECE (dotted line) as a function of the # acquired images. Initially, we train on 500 labelled data-points, and progress in batches of 250 with a budget of 2500.

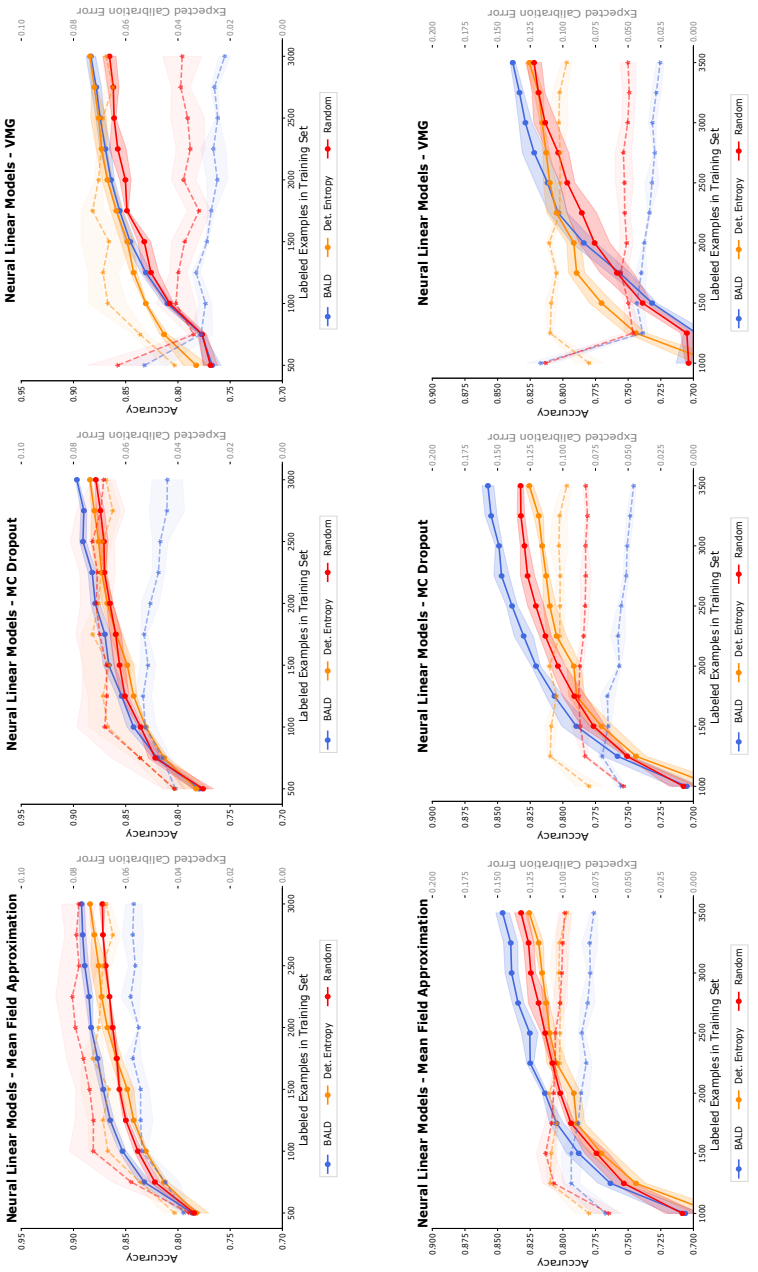


Figure 14: *Am I Better Than Deterministic? Part II.* We investigate the Neural Linear architecture, and use 3 different inference schemes. We show accuracy and ECE (dotted line) as a function of the # acquired images. *Top:* We use the Fashion-MNIST dataset. Initially, we train on 500 labelled data-points, and progress in batches of 250 with a budget of 2500. *Bottom:* We use the SVHN dataset. Initially, we train on 1000 labelled data-points, and progress in batches of 250 with a budget of 2500.

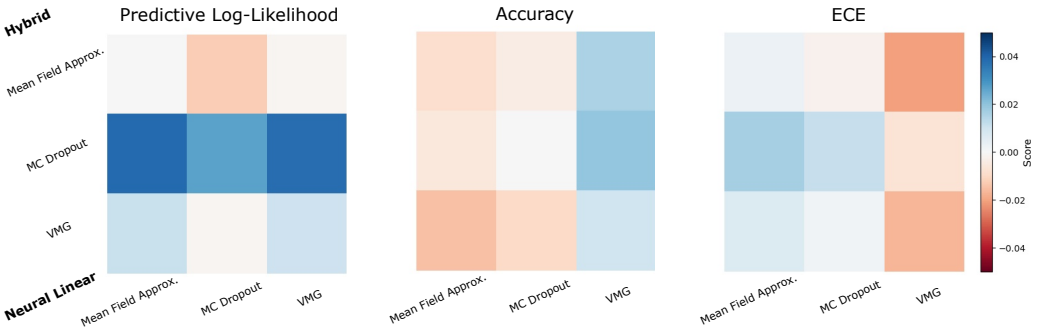


Figure 15: *"Contrasting on a Visual Scale - Fashion MNIST"*. Heat-Maps contrasting Hybrids to Neural Linear models. We report the area between Hybrid and Neural Linear models. BALD is used as the acquisition function.

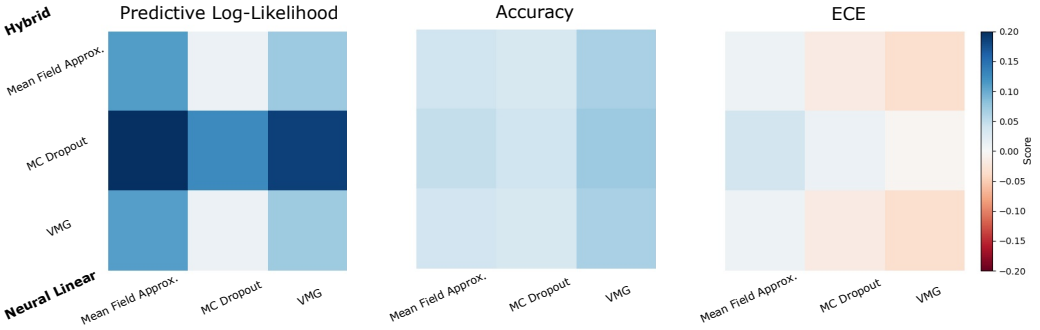


Figure 16: *"Contrasting on a Visual Scale - SVHN"*. Heat-Maps contrasting Hybrids to Neural Linear models. We report the area between Hybrid and Neural Linear models. BALD is used as the acquisition function.