COM3023 - Machine Learning and AI

710019712

1 Introduction

1.1 Problem Definition

A Bayesian Neural Network (BNN) extends traditional neural networks by representing weights and biases as probability distributions, allowing uncertainty quantification and incorporating prior knowledge. However, the most challenging task in following the Bayesian paradigm is the computation of the posterior, which is often infeasible, particularly for large datasets. To address this, estimation approaches have been developed and shown successful in estimating the posterior distributions. This research explores two such techniques, Variational Inference (VI) and Markov Chain Monte Carlo (MCMC), benchmarking on the popular breast cancer dataset to compare the effectiveness of the two techniques.

1.2 A Background in Neural Networks and Bayesian Neural Networks

Artificial Neural Networks (ANNs) are a fundamental paradigm in Deep Learning (DL), a subset of Machine Learning (ML). Neural Networks are inspired by the human brain and are made up of interconnected nodes (neurons) in a layered structure that process information using mathematical functions. These models have shown exceptional performance at recognising patterns in data and making decisions, which has seen their application in tasks such as computer vision and natural language processing, and in industries such as healthcare and finance [1]. Neural Networks were first proposed in 1944 by Warren McCullogh and Walter Pitts [2]. In 1957 Frank Rosenblatt of Cornell University developed the perceptron, the first trainable neural network. A perceptron is a fundamental component of ANNs; it takes multiple inputs, each assigned a specific weight and bias, computes their weighted sum, and then applies an activation function to produce a single output [3].

Although the concept of NNs has been well researched for many decades, only very recently have these networks seen such a rise in popularity for various applications. This can be largely attributed to three factors: a lack of sufficient algorithms for training, the large amount of data required to train complex networks and the large amount of computing resources required during training [4].

Efforts were made to increase the efficiency of training these networks, such as Rumelhart et al. in 1986, with the introduction of the backpropagation algorithm. Backpropagation is a fundamental algorithm in training modern NNs, as it allows the model to learn from errors by adjusting the weights and biases after each forward pass. This is done by calculating the gradient of the chosen loss function with respect to each weight by applying the chain rule, which allows the model to iteratively minimise prediction errors, thus enhancing its performance [5].

Despite these innovations that improved the efficiency of such networks, these models were still held back by the hardware at the time. It was only until the introduction of GPUs that saw the training time of neural networks accelerate significantly. GPUs are designed to handle parallel computations efficiently, making them well-suited for the matrix and vector operations prevalent in the training of NNs [6]. With the advancement of digital hardware, the ability to capture and store real world data exponentially increased, and the combination of such advancements allowed the training of complex NNs to become truly feasible.

Recent research has investigated how Bayesian approximation can be applied to NNs. With a traditional NN, the model weights are not treated as random variables, but rather a true, fixed value that is unknown, and consider the observed data as random variables. However, from a Bayesian view, these parameters are treated as random variables, with distributions of their uncertainty around their true values [7]. This ties into Bayes theorem, where the distribution of these weights is conditional on the data that has been seen.

Bayesian Neural Networks (BNNs) have shown improvements over traditional NNs as they allow you to quantify uncertainty in estimates, which is particularly valuable when applied in high-risk domains, such as medicine, finance and fraud detection. [8] They have proven successful in tasks where data is scarce, and the inclusion of posterior inference is used to control overfitting. The fundamental issue of BNNs is the calculation of the exact posterior distributions, which is often computationally intractable for complex networks due to the high dimensionality of weight spaces. [9] This requires the need for approximation methods, such as Variational Inference and Markov Chain Monte Carlo.

2 Background in Chosen Techniques

2.1 Variational Inference

Variational Inference (VI) is a popular Bayesian inference technique that approximates the posterior distribution by approaching the problem as an optimisation problem. Instead of calculating the exact posterior distribution of the network's weights and biases, a simpler, parameterised distribution (e.g., Gaussian) is used to approximate the true posterior [10]. The goal is to adjust the parameters of the approximate distribution to make it as close as possible to the true posterior. One method of achieving this is to minimise the Kullback-Leibler (KL) divergence between the approximate and true distributions. This can be inverted into maximising the Evidence Lower Bound (ELBO), which serves as a lower bound on the log-likelihood of the observed data.

VI provides many benefits over alternative approximation approaches. Firstly, VI is more scalable for larger networks over MCMC [11]. As the approximate distribution VI produces is often simpler than the true posterior, sampling from it is more computationally efficient, where its benefits are particularly noticeable for high-dimensional data. However, the quality of the variational approximation heavily depends on the chosen family of distributions. For instance, simple approximations may fail to capture the complexity of the true posterior distributions, leading to less reliable estimates compared to MCMC methods, which provide more precise samples by directly sampling from the true posterior. Overall, VI is an efficient and effective method for estimating the posteriors in large-scale and high-dimensional settings, and offers better scalability over MCMC.

2.2 Markov Chain Monte Carlo

On the other hand, MCMC is a class of algorithms designed to sample from complex probability distributions, which enables Bayesian inference in BNNs. In application, MCMC generates samples from the posterior distribution of model parameters given the observed data, to capture the uncertainty in the model. It works by constructing a Markov Chain made up of a sequence of weight configurations, where after sufficient iterations, the distribution of the states approximates the true posterior. The average of the generated samples provides estimates of expectations with respect to the posterior distribution.

Typically, MCMC approaches lead to more robust inference compared to VI. By exploring the entire posterior distribution, MCMC can identify complex dependencies, leading to more reliable inferences. Additionally, MCMC is applicable to a wide range of models and priors, making it a versatile tool for Bayesian inference. However, as its reliability relies on taking many samples from the posterior, this can be computationally demanding, particularly for high-dimensional weight spaces. [12] In addition, challenges can arise with convergence, as the Markov chain can get stuck in local modes or fail to explore the space thoroughly. Overall, MCMC provides a robust framework for estimating posterior distributions, and can provide accurate inference in BNNs.

3 Experimental Design

3.1 Data

The Breast Cancer Wisconsin dataset, sourced from the University of Wisconsin Hospital, is a widely used dataset for binary classification. It is made up of 569 samples from different patients, each characterised by 30 features. The features are computed from digitised images of breast mass cells and describe various characteristics of the cell nuclei, including, for example, mean texture and mean concavity [13]. The target variable is binary, with '1' indicating benign breast cancer, and '0' for malignant cases. This dataset's clinical application demonstrates the importance of understanding uncertainty in predictions, as its application can enhance trust among clinicians and aid in informing decisions in high-risk environments.

In this experiment, the dataset was loaded through scikit-learn. Given its relatively small size, a train/validation/test split of 70%/15%/15% was performed to ensure sufficient data for training and evaluation. The features were scaled using the native scikit-learn scale function to standardise them, as machine learning algorithms are typically sensitive to features with varying ranges.

3.2 Model Architecture

The Bayesian Neural Network for this project was developed using PyTorch and Pyro. Pyro is probabilistic programming library that builds upon PyTorch by enabling flexible and expressive deep probabilistic modelling, including Bayesian modelling techniques such as MCMC and VI [14]. The BNN architecture starts with an input layer of 30 nodes, each corresponding to one of the features in the dataset. This is followed up by two hidden layers, each consisting of 15 nodes. This configuration follows the heuristic for determining the number of hidden nodes to half the size of the input layer, which maintains the networks ability to capture relationships between the input features while avoiding the risk of overfitting [15]. The output layer consists of a single node for the purpose of binary classification.

The prior distributions for the weights and biases in the model are initialised as Normal distributions with a mean of 0 and a standard deviation of 1. This standard normal prior provides an unbiased initial estimator of the parameter values, where all weight values are equally likely within a certain range. Research indicates that standard normal priors often have effective performance in modelling neural network weights [16]. After each layer, a ReLU activation function is applied to introduce non-linearity into the model, allowing it to learn complex patterns in the data. The output of each layer is fed into a BatchNorm, which normalises the output to allow for higher learning rates and faster convergence without compromising performance [17].

This model follows a traditional feedforward structure, and the use of hidden layers allows the model to capture complex non-linear relationships. Although the model is small and relatively simple, it is well-suited for the chosen dataset and it helps to reduce the computational resources needed.

3.3 Implementation of Chosen Methods

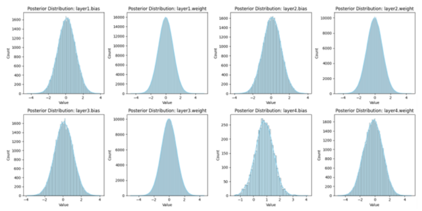
The implementation of MCMC in this experiment uses the No-U-Turns-Sampler (NUTS), an advanced algorithm that extends Hamiltonian Monte Carlo (HMC) methods. It was chosen due to its ability to adaptively determine the optimal step size during sampling, improving sampling efficiency and accuracy, unlike HMC, which requires a pre-determined step size that can be challenging to define. This makes it particularly well-suited for high-dimensional parameter spaces, where traditional MCMC methods may struggle with convergence and efficiency [18]. For this experiment, NUTS was configured with a target probability of 0.7, which strikes a balance between efficient exploration of the parameter space while maintaining a reasonable acceptance rate. MCMC was ran with 2000 samples and 200 warm-up steps, with 2 parallel chains to promote convergence.

For VI, the Stochastic Variational Inference (SVI) method was implemented using Pyro. SVI introduces a parameterised distribution known as the guide, which serves as an approximation to the posterior [14]. During training, the Evidence Lower Bound (ELBO) is used as a loss function, with the goal of optimising the loss between the approximate and true posterior distributions. VI was implemented with 20000 iterations, with an early stopper mechanism added to avoid the model overfitting. The early stopper's patience parameter was set to 3000, meaning that if the loss had not improved after 3000 epochs, training would be halted to reduce the risk of overfitting and unnecessary computation. The SVI was optimised using Pyro's built in AdamW optimiser, with a learning rate set to 0.005. AdamW is a variant of the popular Adam optimiser, however AdamW decouples weight decay from the gradient updates, which leads to better optimisation and more efficient weight regularisation [19]. Once the model was trained, 4000 samples were drawn from the distribution for use in evaluation, which matches the number of samples in MCMC.

Each method was evaluated on their ability to estimate the posterior, and the accuracy and uncertainty of the predictions the model makes. To assess performance, an ensemble approach was used to leverage the full distribution of the posterior samples. This provides a more robust and reliable prediction compared to evaluating individual samples from the posterior distributions.

4 Results

4.1 Performance Evaluation


(a) MCMC Parameter Distributions

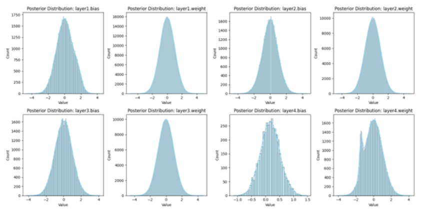
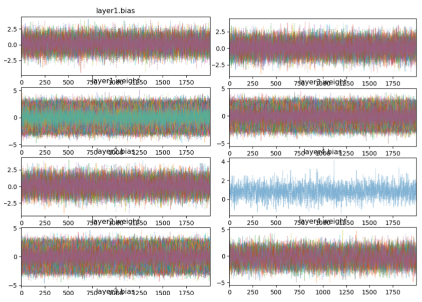
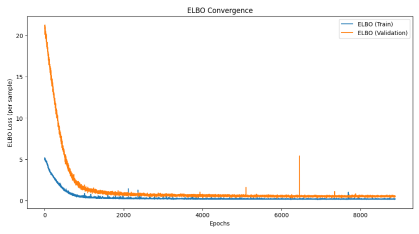

(b) VI Parameter Distributions

Figure 1: Comparison of the Distribution of Weights and Biases for each Layer


(a) MCMC Trace



(b) VI Loss per Epoch

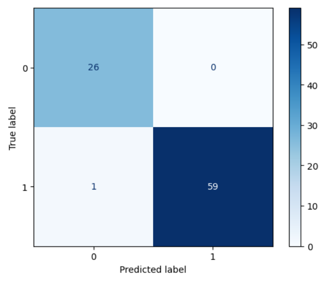
Figure 2: Comparison of the Convergence of Each Method

The posterior MCMC distributions appear to converge well. For most model parameters (weights and biases), the density plots exhibit an approximate standardised Gaussian posterior that can be observed in **Fig. 1**, as specified in the priors. The symmetry and smoothness of the distributions indicate that they are well-defined and stable, suggesting that the sampler explored the parameter space sufficiently, which can be supported in the well-mixed traces seen in **Fig. 2**. However, one exception to this is the bias in the output layer, which appears to be skewed towards 1, while maintaining a similar spread to the other parameters. This skewness could indicate potential overfitting in the final layer of the model, allowing the bias to drift further from its prior.

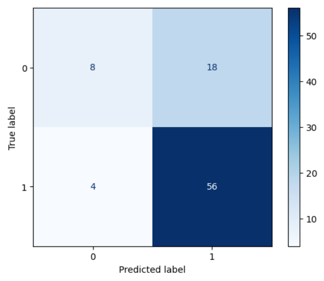
Similarly, the posterior distributions generated by VI also closely match the prior distributions. Both the weights and biases exhibit Gaussian distributions centred around the mean of zero, as seen in **Fig. 1**. However, the bias in layer 4 also stands out here, but for a different reason to MCMC. While it is centred around zero, its range is significantly narrower, which can further support the indication of overfitting in this layer. An interesting observation is the apparent bimodal distribution in layer 4 weights, which may suggest the presence of multiple stable estimates. Overall, VI appears to converge well, which can be supported in **Fig. 2**, where both the training and validation losses start high, but rapidly decrease and stabilise around a loss of 0.1 and 0.4 respectively.

MCMC achieved an ensemble accuracy of 0.99, with an F1-score of 0.99, indicating a very strong overall classification performance. These high scores suggest that the MCMC approximation produced well-calibrated posterior distributions. Additionally, the consistency between the accuracy and F1-score indicates strong generalisation, as seen in the confusion matrix in **Fig 3**, where only one malignant case (label 0) was misclassified. As shown in **Fig. 4**, MCMC's mean predictions align well with the ground truth for most labels, and the uncertainty bands overlap well with the ground truth, highlighting the model's reliable and well-calibrated uncertainty estimates. Additionally, the uncertainty bands are relatively narrow, indicating high confidence in the model's predictions. In a few cases, the uncertainty bands are wider, suggesting greater uncertainty with more difficult samples, however, the model still maintains extremely high performance.

In contrast, the VI model demonstrates a much different performance when applied to unseen data. While its accuracy and F1-score are satisfactory, at 0.74 and 0.84 respectively, this is significantly lower than MCMC. However, when evaluating individual samples from the posterior distribution, the performance drops further, with an accuracy of 0.52 and an F1-score of 0.60. This is clear when looking at **Fig. 4**, where VI predictions exhibit far greater uncertainty than MCMC. For all samples, the one-standard-deviation band spans almost the entire probability range, and the mean predictions show a lack of confidence, hovering around the 0.5 threshold. Nevertheless, when aggregating predictions across posterior samples, VI still achieves decent classification performance, which demonstrates the key advantage of Bayesian methods: while individual posterior samples may yield subpar performance, their ensemble predictions benefit from averaging out individual errors.



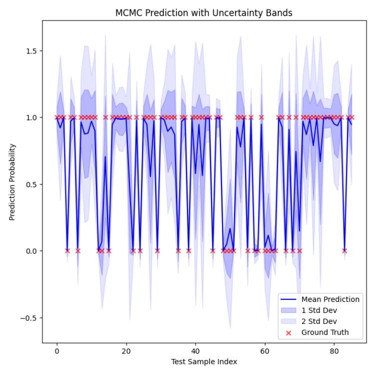
(a) MCMC Confusion Matrix



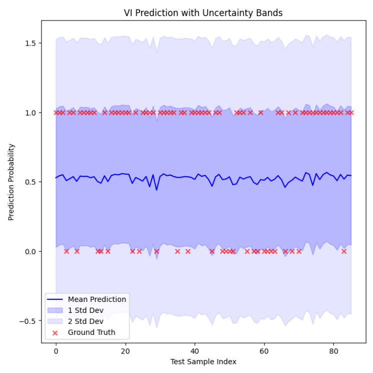
(b) VI Confusion Matrix

Figure 3: Comparison of Model Performances in Confusion Matrices

Fig.5 clearly shows the uncertainty difference between the two methods: MCMC has a tall, narrow distribution centred around 0.6, whereas VI shows a much wider, flatter distribution, centred around 0.5, reflecting far greater uncertainty. In terms of the computational cost of both methods, the training process of MCMC was significantly slower than VI. Training both methods on a CPU in Google Cloud, MCMC took around 1 hour 30 mins to run, whereas VI took just 4 minutes with an early stopper.



(a) MCMC Uncertainties



(b) VI Uncertainties

Figure 4: Comparison of Model Uncertainties for Each Method

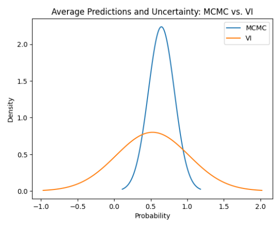


Figure 5: Comparison of Average Predictions for MCMC and VI

5 Conclusion

In conclusion, both MCMC and VI are robust methods for estimating the posteriors in BNNs. MCMC provides a more precise representation of the true posteriors at a higher computational cost, whereas VI makes computational compromises at the expense of greater uncertainties in its predictions. Despite the lengthened training time, MCMC proved to be the more effective method for this application.

6 Appendix

[1] Lyzr Team. (2024, November 23). Neural Networks: Understanding Complex Patterns and Architectures. Lyzr AI Glossary.

[2] Hardesty, L. (2017, April 14). Explained: Neural networks. MIT News.

[3] Rosenblatt, F., 1958. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65(6), p.386.

[4] Goan, E. and Fookes, C., 2020. Bayesian neural networks: An introduction and survey. Case Studies in Applied Bayesian Data Science: CIRM Jean-Morlet Chair, Fall 2018, pp.45-87.

[5] Zhou, X., Zhang, W., Chen, Z., Diao, S. and Zhang, T., 2021. Efficient neural network training via forward and backward propagation sparsification. Advances in neural information processing systems, 34, pp.15216-15229.

[6] Oh, K.S. and Jung, K., 2004. GPU implementation of neural networks. Pattern Recognition, 37(6), pp.1311-1314.

[7] Yuan, X., Li, J. and Kuruoglu, E.E., 2025. Uncertainty Quantification With Noise Injection in Neural Networks: A Bayesian Perspective. arXiv preprint arXiv:2501.12314.

[8] Magris, M. and Iosifidis, A., 2023. Bayesian learning for neural networks: an algorithmic survey. Artificial Intelligence Review, 56(10), pp.11773-11823.

[9] Charnock, T., Perreault-Levasseur, L. and Lanusse, F., 2022. Bayesian neural networks. In Artificial Intelligence for High Energy Physics (pp. 663-713).

[10] Knoblauch, J., Jewson, J. and Damoulas, T., 2022. An optimization-centric view on Bayes' rule: Reviewing and generalizing variational inference. Journal of Machine Learning Research, 23(132), pp.1-109.

[11] Costa, G. (2022, December 21). A first insight into Bayesian Neural Networks (BNNs). Medium.

[12] Papamarkou, T., Hinkle, J., Young, M.T. and Womble, D., 2022. Challenges in Markov chain Monte Carlo for Bayesian neural networks. Statistical Science, 37(3), pp.425-442.

[13] Wolberg, W., Mangasarian, O., Street, N., & Street, W. (1993). Breast Cancer Wisconsin (Diagnostic) Data Set. UCI Machine Learning Repository.

[14] Bingham, E., Chen, J.P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P. and Goodman, N.D., 2018. Pyro: Deep Universal Probabilistic Programming. Journal of Machine Learning Research, 19(1), pp.973–978.

[15] Uzair, M. and Jamil, N., 2020, November. Effects of hidden layers on the efficiency of neural networks. In 2020 IEEE 23rd international multitopic conference (INMIC) (pp. 1-6). IEEE.

[16] Fortuin, V., Garriga-Alonso, A., Ober, S.W., Wenzel, F., Rätsch, G., Turner, R.E., van der Wilk, M. and Aitchison, L., 2021. Bayesian neural network priors revisited. arXiv preprint arXiv:2102.06571.

[17] Brownlee, J., A Gentle Introduction to Batch Normalization for Deep Neural Networks. 2019. URL https://machinelearningmastery. com/batch-normalization-for-training-of-deep-neural-networks.

[18] Hoffman, M.D. and Gelman, A., 2014. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. J. Mach. Learn. Res., 15(1), pp.1593-1623.

[19] Loshchilov, I. and Hutter, F., 2017. Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.