Unreliable Monte Carlo Dropout Uncertainty Estimation

Anonymous Full Paper
Submission 33

Abstract

Reliable uncertainty estimation is crucial for machine learning models, especially in safety-critical domains. While exact Bayesian inference offers a principled approach, it is often computationally infeasible for deep neural networks. Monte Carlo dropout (MCD) was proposed as an efficient approximation to Bayesian inference in deep learning by applying dropout at inference time [1]. Hence, the method generates multiple sub-models yielding a distribution of predictions to estimate uncertainty. We investigate its ability to capture true uncertainty and compare to Gaussian Processes (GP) and Bayesian Neural Networks (BNN). We find that MCD struggles to accurately reflect the underlying true uncertainty, particularly failing to capture increased uncertainty in extrapolation and interpolation regions observed in Bayesian models. The findings suggest that uncertainty estimates from MCD, as implemented and evaluated in these experiments, may not be as reliable as those from traditional Bayesian approaches for capturing epistemic and aleatoric uncertainty.

1 Introduction

In numerous practical applications, particularly those where decisions have significant consequences, machine learning models need to provide not only accurate point predictions, but also reliable estimates of their uncertainty [2]. Understanding when a model is uncertain about its prediction is vital for safe and robust deployment.

Bayesian machine learning offers a principled approach to quantifying uncertainty by modelling probability distributions over model parameters or functions [3]. This framework allows for the capture of both epistemic uncertainty (due to lack of knowledge, potentially reducible with more data) and aleatoric uncertainty (inherent noise in the data). However, performing exact Bayesian inference in complex models like deep neural networks is computationally intensive and often impractical [4] even with modern improvements such as full batch Hamiltonian Monte Carlo [5].

To address the computational challenges of Bayesian deep learning, Monte Carlo dropout (MCD) has been proposed as a more efficient approximation [1, 6]. The method involves applying dropout not just during training, but also at inference (prediction) time. By performing multiple forward passes in parallel with different dropout masks, one effectively samples from an ensemble of thinned networks, and the variance of these predictions is used as an estimate of model uncertainty. It is claimed that this method is mathematically equivalent to an approximation to a probabilistic deep Gaussian process. The main goal of this work is to understand the MCD method, by applying it to a simple toy problem.

Here we summarize findings from a series of controlled experiments applying MCD to a simple regression task and comparing to two Bayesian benchmark models: Gaussian Processes (GP) and Bayesian Neural Networks (BNN) which converge to identical posteriors in the large-layer regime [7]. The focus is on the ability to capture uncertainty during interpolation (small gaps in data within the training range) and extrapolation (outside the training data range).

2 Related Work

Quantifying uncertainty in machine learning is a significant area of research, particularly for deep learning models which are often used in complex tasks without providing confidence measures. Traditional methods often provide only point estimates, making it difficult to assess their reliability.

Bayesian methods offer a framework for modelling uncertainty [3]. Instead of learning fixed parameters, Bayesian models infer probability distributions over parameters. Gaussian Processes (GP) are non-parametric Bayesian models that define a distribution directly over functions [8, 9]. For any finite set of input points, the corresponding function values have a joint Gaussian distribution, defined by a mean function and a covariance function (kernel). GPs naturally provide predictive distributions, capturing uncertainty based on the training data and kernel choice. The process involves computing a posterior distribution given training data and optimizing kernel hyperparameters, often by maximizing the marginal likelihood.

Bayesian Neural Networks (BNN) extend standard neural networks by placing probability distributions over their weights and biases, rather than learning single point values [10, 11]. Computing the exact posterior distribution over weights in BNNs

is generally intractable, necessitating approximation methods. Common approaches include **sampling methods** like Markov Chain Monte Carlo (MCMC) or Hamiltonian Monte Carlo (HMC), and **variational inference (VI)**. MCMC methods approximate the posterior by generating samples that converge to the target distribution.

Monte Carlo Dropout (MCD) was suggested to as a computationally efficient alternative for obtaining uncertainty estimates in deep learning [1]. The core idea is to keep dropout layers active during inference and perform multiple stochastic forward passes. Each pass uses a different dropout mask, effectively sampling from a distribution of sub-networks. The mean of these samples provides the prediction, and their variance quantifies the model's uncertainty. The original paper suggests that this is equivalent to a Bayesian approximation for deep Gaussian processes [9].

3 Methodology

The evaluation of MCD was conducted using a controlled regression task with synthetic data described in Section 3.1. The data was generated with a known noise distribution allowing for direct comparison against the model predictions and the true underlying function and known noise characteristics.

We performed the following experiments: **Comparing methods**: Compares MCD with GP and BNN on datasets with varying amounts of training data (15, 50, 150 points) and noise levels ($N(0, 0.05)$ and $N(0, 0.2)$). A gap in training data $[-0.5, 0.5]$ was included to test interpolation and extrapolation. For each method, the mean and 95% confidence interval of the predictions are compared to the true underlying function.

Reconstructing uncertainty: Providing the model with (practically) infinite data and information of the uncertainty during the fitting. It was expected that this would lead to reproduction of the underlying uncertainty. To provide practically infinite data, new data samples were drawn each epoch. The noise was modelled as var $= N(0, \sigma)\sin(x)/x$.

To inform the model of the properties of the noise, the noise variance was added to the loss function such that:

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_{\text{mean}} + \mathcal{L}_{\text{noise}} + \lambda_2 \mathcal{L}_{\text{reg}} \qquad (1)\\ &= \sum_i |y_{\text{mean}}(x_i) - y_i|^2 + \sum_i |\sigma(x_i) - \sigma_i|^2 + \mathcal{L}_{\text{reg}}.\end{aligned}$$

Where y_i and $y_{\text{mean}}(x_i)$ are the true value and the mean of the predictions at x_i, respectively, and similarly for $\sigma(x_i)$ and σ_i being the variance. $\lambda_2 \mathcal{L}_{\text{reg}}$ represents any regularization with coefficient λ_2. After each training step, Monte Carlo dropout inference is performed to estimate the predictive mean and standard deviation of the model. The objective is to minimize the Mean Squared Error (MSE) of both the prediction error and the difference in uncertainty between the predictions of the model and the true data-generating process. Hence, to estimate the posterior distribution, we need to run inference after each epoch.

Since the variance of the predictions depends on the dropout rate, we performed hyper parameter optimization to find the optimal combination of L_2 regularization coefficient and inference dropout rate by minimizing Equation 1.

3.1 Toy problem data

Synthetic data was generated from a known one-dimensional function:

$$f(x) = \sin(4x) + r_0 + r_1 x + (r_2 + 1/2)x^2, \quad (2)$$

where r_0, r_1, r_2 are sampled from $\mathcal{N}(0, 1)$ and the input data was sampled from the range -1.3 and 1.3. Experiments included both noise-free and noisy observations to study the model's ability to capture aleatoric uncertainty. The noise was either sampled from a normal distribution $\mathcal{N}(0, \sigma_{obs}^2)$, with constant σ_{obs} across the range or scaled with $\sin(x)/x$. Some experiments introduced a gap in the training data in the interval $[-0.5, 0.5]$ to specifically evaluate interpolation performance. Lastly, the data set was standardized.

3.2 Implementations

The Monte Carlo Dropout (MCD) was implemented in PyTorch using a custom MCDLayer class. The network architecture consisted of an input layer, three hidden layers with ReLU activation, and an output layer. The dropout was applied during the forward pass at inference time to obtain multiple predictions. The mean and variance of these predictions provided the estimate of the predictive distribution. In our experiments, we used a dropout inference rate of 0.15, 32 nodes in the hidden layers, 10^{-4} in L_2 regularization and 10^{-3} in learning rate.

In both experiments, dropout was applied only to the hidden layers. With small number of input features (three), the dropout removal of units risks disrupting the input structure, and adding noise too early may cause instability. Dropout in the output layer introduces randomness into the final prediction, making uncertainty estimates harder to interpret. In contrast, dropout in hidden layers effectively explores different sub-models of the decoded input, before encoding it back to the predicted output.

The Gaussian Process (GP) was implemented with GPJax [12] using a zero-mean prior and Radial Basis Function (RBF) kernel, with a Gaussian likelihood assuming independent noise. Hyperparameters

were optimized by minimizing the negative marginal log-likelihood using the Adam optimizer.

The Bayesian Neural Network (BNN) was implemented in NumPyro [13, 14] with a two-hidden-layer architecture with tanh activation. Weights were assigned normal priors, and observation noise had a Gamma prior. The No-U-Turn Sampler (NUTS) was used to sample weights [15] during inference. Predictions were made by averaging outputs from posterior weight samples, typically without adding observation noise during inference to focus on model uncertainty.

Hyper parameter optimization was performed using Optuna.

4 Results

4.1 Comparing methods

Figure 1 compares the means and 95% confidence intervals of the predictions provided by GP, BNN, and MCD on data with shot noise sampled from $N(0, 0.05)$. The training data included a gap in the range $[-0.5, 0.5]$ to evaluate interpolation. The results are shown for increasing numbers of training data points (15, 50, 150). For all three frameworks, the increase in data points leads to an improved approximation (lower MSE) of the true function, particularly in the range of the training data, but also in the interpolation range.

The GP and BNN models show a characteristic increase in the confidence interval 95% in regions without training data, i.e. in the interpolation gap $[-0.5, 0.5]$ and in extrapolation regions outside the training data range $[-1, -0.5]$ and $[0.5, 1]$. This reflects increasing uncertainty when the model is asked to predict values far from observed data. In contrast, the MCD dropout model exhibits a more constant variance across the entire input range without pronounced increase in interpolation or extrapolation intervals.

When increasing the shot noise to $N(0, 0.2)$ (Figure 2), the variance increase significantly in the data region for both GP and MCD, while the BNN remains variance remains similar to the $N(0, 0.05)$ scenario. In the interpolation region, the variance increases for all frameworks, but for the MCD, the variance is still similar across all regions, without increasing in the data gap.

These results suggest that, in this comparative setting, MCD's inherent mechanism does not automatically yield an uncertainty estimate that captures the varying levels of confidence (epistemic uncertainty) in regions further from training data as effectively as the benchmark Bayesian models. This motivates the second experiment with varying noise level across the training data (Section **??**).

It is known that random initialisations of neural networks may lead to different outcomes, especially with weak regularization [16]. Figure 1 and 2 only show one such realisation. Performing each experiment four times with different seeds showed that while initialisation lead to different mean solutions, the behaviour of the variance is consistent across individual initialisations (Figure A.1). In addition, the model was trained using 100 different seeds resulting in a mean MSE of 0.384 and with a MSE standard deviation of 0.156.

Table 1, we quantify the difference in posterior across different seeds.

Seed	MSE	Mean STD Error
5	0.1436	0.0113
6	0.3230	0.0172
7	0.3325	0.0106
8	0.1564	0.0157

Table 1. MSE and Mean STD errors for different seeds

Figure 3 shows that combining the posterior predictions from different random seeds for MCD can result in an uncertainty profile that more closely resembles that of the GP and BNN models, with increased uncertainty in extrapolation and interpolation areas. This is because the variations across different seeds are more pronounced in data-scarce regions, but this requires training an ensemble of MCD models thereby significantly increasing the computational load.

5 Discussion

Our experimental results, provide insight into the behavior of MCD as an uncertainty estimation method and allow us to address the research question regarding its ability to approximate true uncertainty.

Comparing MCD with the benchmark Bayesian models (GP and BNN) reveals a fundamental difference in how uncertainty is captured. GP and BNN models naturally show increasing uncertainty as predictions move away from training data points (extrapolation and interpolation regions), reflecting a decrease in epistemic certainty due to lack of information. This behavior is a desirable property for reliable uncertainty estimation.

In contrast, the MCD tends to produce a more uniform uncertainty across the input space. This suggests that MCD's uncertainty estimate, primarily derived from the variance of predictions across different dropout masks, may not be effectively capturing epistemic uncertainty in the same way that GP or BNN models do. Predicting with high confidence in areas far from training data or regions with large noise, where the model has limited evidence, is a significant drawback for safety-critical applications.

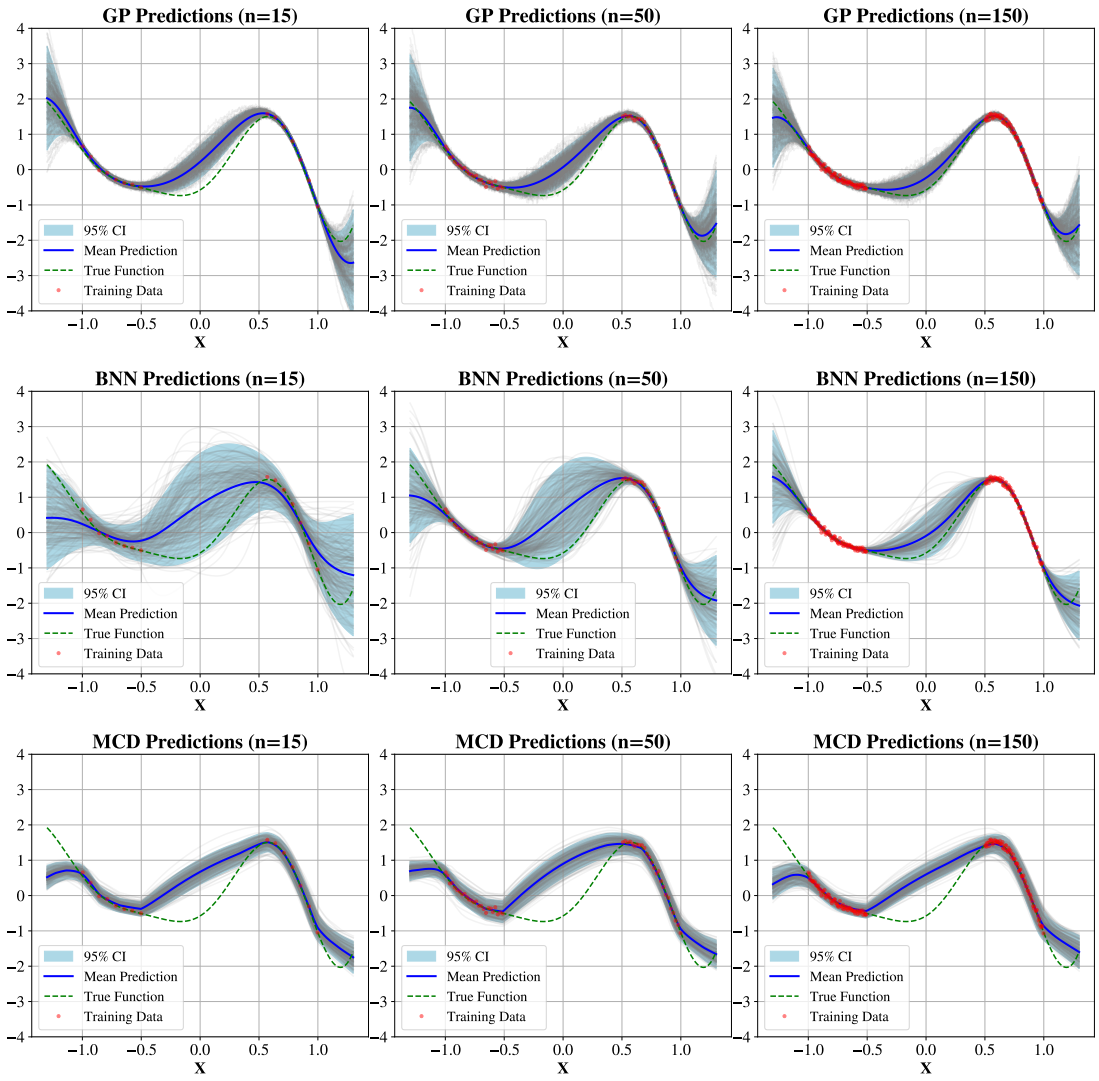

Figure 1. Comparing the Gaussian Process (GP, top row), Bayesian Neural Network (BNN, middle row) and Monte Carlo Dropout (MDC, bottom row) when fitted to 15, 50 and 150 data points (left, middle, right, respectively) with a shot noise of $\sigma = 0.05$. Notice how the GP and BNN has narrow variance in the regions where the data are sampled and higher variance in the interpolation and extrapolation regions, while the MCD has constant variance across the range.

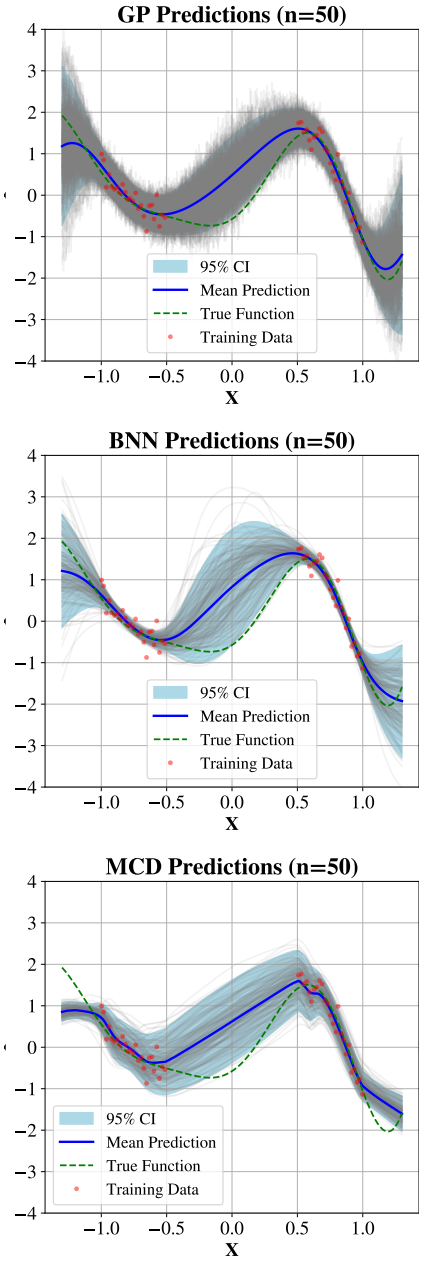

Figure 2. Comparing the Gaussian Process (GP, top row), Bayesian Neural Network (BNN, second row) and Monte Carlo Dropout (MDC, third row), when fitted to 50 data points with a shot noise of $\sigma = 0.2$.

The dependency of MCD's posterior on the random seed further questions its reliability as an objective uncertainty estimator. An ideal uncertainty model should not have its uncertainty profile heavily influenced by the specific initialization or randomness during training or inference. While combining predictions from different seeds can create a more Bayesian-like uncertainty profile, this suggests that a single MCD model might not be sufficient, potentially requiring ensemble modelling, which adds computational cost.

Based on the claim that MCD is mathematically

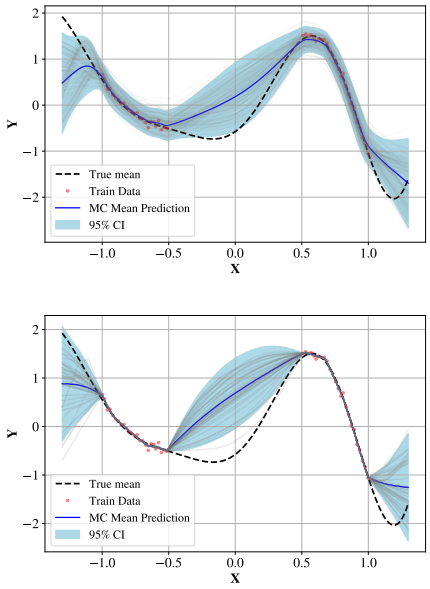

Figure 3. Random initialisations of the network also creates variation. In the example, a MCD model is fitted to 50 data points with a shot noise of $\sigma = 0.05$. Combining predictions trained with different initializations, leads to a profile that more closely resembles that of the GP and BNN models (upper panel, compared to second column of Figure 1). In the top plot, the dropout is applied in training, while in the bottom plot dropout is not applied, and gives a more smilar shape as the bayesian methods.

equivalent to an approximation to a deep Gaussian process [1] we expected similar uncertainty characteristics, particularly increased uncertainty away from data. Our results, showing a generally constant uncertainty profile for MCD contrasting with the spatially varying uncertainty of GP and BNN do not support this equivalence in practice for the problems studied.

Regularization techniques (L2, L1, dropout during training) will impact MCD's predictions and uncertainty. However, even when treating these as hyper parameters and informing the loss with the shot noise, the MDC provided variance did not resemble the shot noise.

Although our experiments showed that the MCD method struggles to reproduce the uncertainty, further experiments are needed before the method can be fully rejected. Nevertheless, caution is strongly advised when interpreting its predicted uncertainty.

6 Conclusion

We have investigated the Monte Carlo dropout method's ability to approximate the true uncertainty in a regression task, comparing it against Gaussian Processes and Bayesian Neural Networks. While the original paper reported strong performance [1], we

find that MDC failed to accurately represent the uncertainty in its posterior predictions. It struggled to capture uncertainty during extrapolation and interpolation, sometimes predicting lower uncertainty than in regions with training data, which was opposite behavior of the BNN and GP benchmarks.

Acknowledgments

This paper has taken advantage of NotebookLLM to condense sections of a master thesis into the appropriate NLDL format. All text and cited references have been manually checked by the authors.

Author contributions

References

[1] Y. Gal and Z. Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *International Conference on Machine Learning (ICML)*. 2016, pp. 1050–1059.

[2] "The foundations of statistics. By Leonard J. Savage, John Wiley & Sons, Inc., 1954, 294 pp". In: *Naval Research Logistics Quarterly* 1.3 (1954), pp. 236–236. DOI: https://doi.org/10.1002/nav.3800010316.

[3] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 1995.

[4] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. *Weight Uncertainty in Neural Networks*. 2015. arXiv: 1505.05424 [stat.ML]. URL: https://arxiv.org/abs/1505.05424.

[5] P. Izmailov, S. Vikram, M. D. Hoffman, and A. G. G. Wilson. "What Are Bayesian Neural Network Posteriors Really Like?" In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 4629–4640. URL: https://proceedings.mlr.press/v139/izmailov21a.html.

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[7] J. Hron, Y. Bahri, R. Novak, J. Pennington, and J. Sohl-Dickstein. *Exact posterior distributions of wide Bayesian neural networks*. 2020. arXiv: 2006.10541 [stat.ML]. URL: https://arxiv.org/abs/2006.10541.

[8] R. M. Neal. *Bayesian Learning for Neural Networks, Vol. 118 of Lecture Notes in Statistics*. Springer-Verlag, 1996.

[9] C. K. Williams and C. E. Rasmussen. *Gaussian Processes for Machine Learning*. Vol. 2. MIT press Cambridge, MA, 2006.

[10] D. J. C. MacKay. "A Practical Bayesian Framework for Backpropagation Networks". In: *Neural Computation* 4.3 (May 1992), pp. 448–472. ISSN: 0899-7667. DOI: 10.1162/neco.1992.4.3.448.

[11] G. E. Hinton and D. van Camp. "Keeping the neural networks simple by minimizing the description length of the weights". In: *Proceedings of the Sixth Annual Conference on Computational Learning Theory*. COLT '93. Santa Cruz, California, USA: Association for Computing Machinery, 1993, pp. 5–13. ISBN: 0897916115. DOI: 10.1145/168304.168306.

[12] T. Pinder and D. Dodd. "GPJax: A Gaussian Process Framework in JAX". In: *Journal of Open Source Software* 7.75 (2022), p. 4455. DOI: 10.21105/joss.04455. URL: https://doi.org/10.21105/joss.04455.

[13] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. A. Szerlip, P. Horsfall, and N. D. Goodman. "Pyro: Deep Universal Probabilistic Programming". In: *Journal of Machine Learning Research* 20 (2019), 28:1–28:6. URL: http://jmlr.org/papers/v20/18-403.html.

[14] D. Phan, N. Pradhan, and M. Jankowiak. "Composable Effects for Flexible and Accelerated Probabilistic Programming in NumPyro". In: *arXiv preprint arXiv:1912.11554* (2019).

[15] M. D. Hoffman and A. Gelman. "The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo". In: *arXiv preprint arXiv:1111.4246* (2011). URL: https://arxiv.org/abs/1111.4246.

[16] M. Narkhede, P. Bartakke, and M. Sutaone. "A review on weight initialization strategies for neural networks". In: *Artificial Intelligence Review* 55 (Jan. 2022), pp. 1–32. DOI: 10.1007/s10462-021-10033-z.

A Appendix

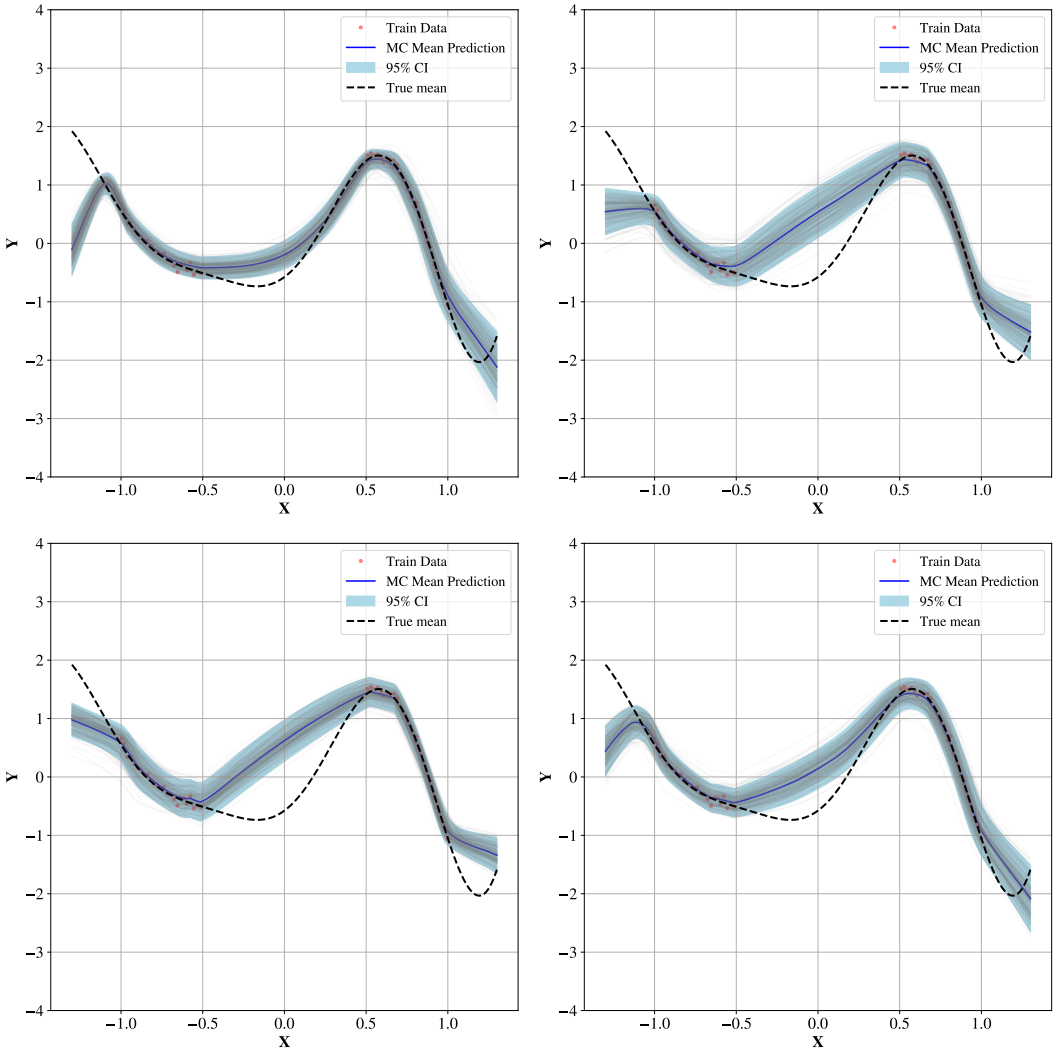

Figure A.1. MCD results for different random initialisations of the neural network.