# STABILIZING PRIORS FOR ROBUST BAYESIAN DEEP LEARNING - REPRODUCIBILITY CHALLENGE REPORT

**Srinjoy Ganguly**
University of Southampton
Department of Electronics and Computer Science
Student ID: 31341195
sg8n19@soton.ac.uk

**Arya Kaustava Mishra**
University of Southampton
Department of Electronics and Computer Science
Student ID: 24361054
akm1u19@soton.ac.uk

**Sandeep Kavarthapu**
University of Southampton
Department of Electronics and Computer Science
Student ID: 31403859
sck1n19@soton.ac.uk

## ABSTRACT

Bayseian Neural Networks (BNNs) are harder to train whenever there is a presence of many hidden layers and large weight variances. In this paper, we reproduce research results which proposes a signal propagation solution to the problem called self stabilizing priors which leads to more robust and simple to train BNNs and also helps in scalability of BNNs as well.

## 1 INTRODUCTION

The main research question which we are trying to address is that how can we make scalable Bayesian deep learning models, of which the most prominent one includes Bayesian Neural Network (BNNs) Graves (2011); Blundell et al. (2015). We focus on the issue of scalability of BNNs and to address that issue we need to develop efficient training strategies for the network. For all Bayesian models, the choice of the prior determines a lot about the model and its behaviour. In the case of BNNs, the priors of the BNN do not have any significant impact on the signal propagation in the BNN and the weights are only updated after the forward pass in the network. In this way, no prior information is properly utilized for training of the network. This paper is a reimplementation of McGregor et al. (2019a)

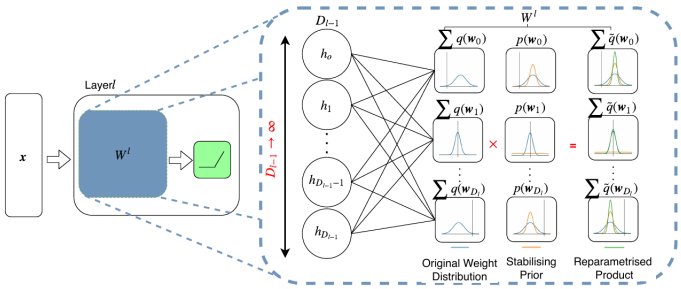



Figure 1: BNN Stabilizing Prior recreated from Demo notebook

In this paper, self stabilizing priors are proposed which actually make an impact on the priors during the forward pass training of the BNN and makes them more robust during their training, as shown in the above Figure 1. This is achieved by making modifications to the Evidence Lower Bound (ELBO) equation of the BNN which includes both the prior and the posterior. This modification ensures that

the priors are also adjusted according to the posterior for each of the forward pass during the training of the BNN which was not happening earlier. The stabilizing prior helps keep the variance of the weights to not become large and their distribution to be as close to the means as possible. This proposal has been clearly defined in McGregor et al. (2019a).

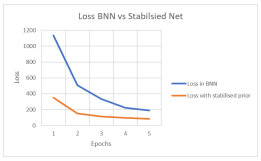
## 2 Implementation Details

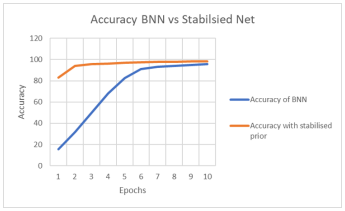
The code which has been utilized for the reproducibility of this paper is given on McGregor et al. (2019b). We have utilized a demo version of the code provided by the original author of the paper and have performed significant analysis on the MNIST and CIFAR10 dataset. We have tried to reproduce the findings in the paper, performed hyperparameter tuning and wrote some more code for visualizing results and analysis in addition to that given in the paper already. Along with this we were also able to identfy few anomalies in the code base as well.

The whole code base has been implemented using PyTorch library. Two main classes are being created, both being Bayesian Neural Networks with fully connected layers with some specific hidden layers, but one of them is normal BNN with local reparameteization trick (LRT) Kingma et al. (2015) and the other one is BNN which utilizes stabilizing priors with LRT as well. The training and test accuracy are stored as CSV files and a separate plotting code is utilized for plotting the results.

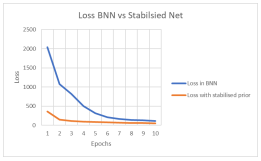
## 3 Experimentation and Analysis

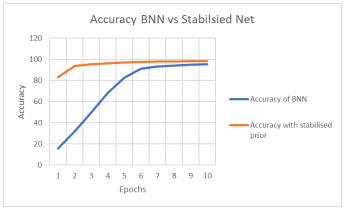
Our implementation code can be found here COMP6248-Reproducability-Challenge. The author's code had the width size (number of neurons in each hidden layer) set to be 256, but for our experiments we changed it to 512.



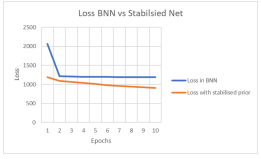



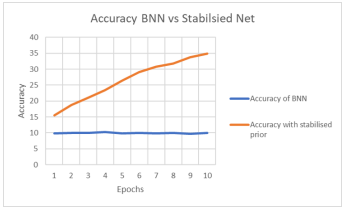

Above figures indicate the training loss and accuracy for MNIST with the chosen hyperparameters as - epochs = 5, learning rate = 0.01, prior variance = 0.02 with 3 hidden layers. For the MNIST, within 5 epochs we were able to get some nice accuracy results.



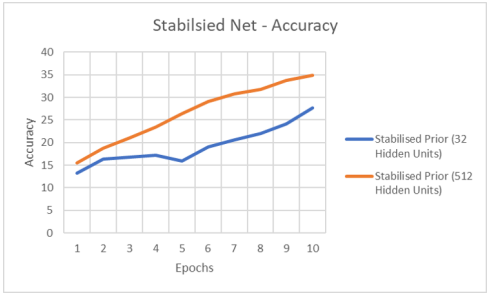




The above results compare the training accuracy and loss for MNIST with the chosen parameters as - epochs = 10, learning rate = 0.001, prior variance = 0.02 with 3 hidden layers. It can be clearly observed that stabilizing prior is performing better than the normal BNN.

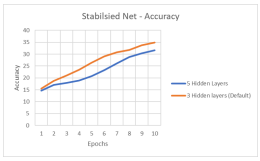


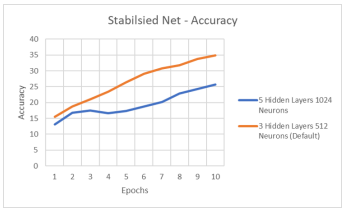



The above results compare the training accuracy and loss for CIFAR10 with the chosen parameters as - epochs = 10, learning rate = 0.001, prior variance = 0.02 with 3 hidden layers. The accuracy for CIFAR10 is poor but, stabilizing priors performs relatively better than normal BNN.




The plot above shows the improvement of accuracy of stabilized BNN with more hidden units. Since, width size of 32 for CIFAR10 was giving poor accuracy so we increased the width size to 512 and got better accuracy compared to 32. Also, in MNIST since we have binary images (less information content) hence, within a few training epochs the BNN was able to provide accurate results, whereas, in CIFAR10, the images are colored and contains more information content from RGB channels, so this deeper architecture of BNNs will be better to gain performance.




Since in CIFAR10 we were getting less accuracy, so we increased the number of hidden layers from 3 to 5, keeping width size as 512, and the left plot shows the results. The performance decreased by a small margin. Because of this we increased the width size to 1024 for 5 layers, and interestingly the performance decreased further.

The author's original code was first run in our local machine, but it gave errors related to device selection (CPU or GPU). At this point we modified the code to make the device selection automatic. With this change, we executed the code in CPU and it took around 20 minutes to train the BNNs. However, with Google Colab GPU, the training time reduced to 5 minutes. All the extra coding for the visualization of the results were completed by us. For the generation of these results we tried changing the initial variance and prior variance with different values such as 0.01, 0.02, etc, but we did not find any significant changes in our performance plots. The experimental results also show that with self stabilizing priors, the BNN is converging much faster than the normal BNN. The author's results make use of heat maps for comparing the performance of the BNN models, but we use more simple plots to demonstrate the same comparison.

## 4 Conclusion

As per our experimentation results and analysis for Stabilizing priors, we mostly agree with the original paper results which shows the plots related to accuracy. We were able to replicate the main original paper experiment results by significant experimentation and analysis. However, interestingly, when we increased the number of hidden layers and neurons for CIFAR10 case, the accuracy decreased and this should not have happened as the original paper promises that stabilizing priors works well with deep BNNs as well. This issue can open new research directions for the development of more robust training algorithms for Bayesian Deep Learning methods. In terms of resources for this research, GPUs are best suitable for the purpose of Bayesian Deep Learning methods as these involve analytically intractable normalization constants which are computationally expensive. GPUs reduce the computation time for training as we found out through experimentation.

## ACKNOWLEDGMENTS

We would like to thank Dr. Jonathan Hare as well as University of Southampton for helping us in this research. We also gratefully acknowledge Google Colab for the free online GPU resource which was utilized for this research.

## INDIVIDUAL CONTRIBUTION

Srinjoy, Sandeep and Arya together came up with the idea of selection of this research paper. Srinjoy changed the architecture of the Stabilizing priors network by addition of hidden layers and making different configurations. Sandeep compared the performance for MNIST and CIFAR10 datasets for basic BNNs. Arya contributed to the hyperparameter tuning of both the models and validated the results.

For the writing of this report, Sandeep contributed towards the Introduction and implementation details for this report. Srinjoy and Arya contributed to the experimentation, analysis and conclusion parts of this report equally.

## REFERENCES

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks, 2015.

Alex Graves. Practical variational inference for neural networks. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 24*, pp. 2348–2356. Curran Associates, Inc., 2011. URL http://papers.nips.cc/paper/4329-practical-variational-inference-for-neural-networks.pdf.

Diederik P. Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick, 2015.

Felix McGregor, Arnu Pretorius, Johan du Preez, and Steve Kroon. Stabilising priors for robust bayesian deep learning, 2019a.

Felix McGregor, Arnu Pretorius, Johan du Preez, and Steve Kroon. *Stabilising priors for robust Bayesian deep learning*, 2019b. URL https://github.com/felixmcgregor/Bayesian-Neural-Networks-with-self-stabilising-priors/blob/master/BasicDemo.ipynb.