CONSISTENT BAYESIAN LEARNING FOR NEURAL NETWORK MODELS: THEORY AND COMPUTATION

By

Sanket Rajendra Jantre

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Statistics – Doctor of Philosophy

2022

ABSTRACT

CONSISTENT BAYESIAN LEARNING FOR NEURAL NETWORK MODELS: THEORY AND COMPUTATION

By

Sanket Rajendra Jantre

Bayesian framework adapted for neural network learning, Bayesian neural networks, have received widespread attention and successfully applied to various applications. Bayesian inference for neural networks promises improved predictions with reliable uncertainty estimates, robustness, principled model comparison, and decision-making under uncertainty. In this dissertation, we propose novel theoretically consistent Bayesian neural network models and provide their computationally efficient posterior inference algorithms.

In Chapter 2, we introduce a Bayesian quantile regression neural network assuming an asymmetric Laplace distribution for the response variable. The normal-exponential mixture representation of the asymmetric Laplace density is utilized to derive the Gibbs sampling coupled with Metropolis-Hastings algorithm for the posterior inference. We establish the posterior consistency under a misspecified asymmetric Laplace density model. We illustrate the proposed method with simulation studies and real data examples.

Traditional Bayesian learning methods are limited by their scalability to large data and feature spaces due to the expensive inference approaches, however recent developments in variational inference techniques and sparse learning have brought renewed interest to this area. Sparse deep neural networks have proven to be efficient for predictive model building in large-scale studies. Although several works have studied theoretical and numerical properties of sparse neural architectures, they have primarily focused on the edge selection.

In Chapter 3, we propose a sparse Bayesian technique using spike-and-slab Gaussian prior to allow for automatic node selection. The spike-and-slab prior alleviates the need of an ad-hoc thresholding rule for pruning. In addition, we adopt a variational Bayes approach to circumvent the computational challenges of traditional Markov chain Monte Carlo

implementation. In the context of node selection, we establish the variational posterior consistency together with the layer-wise characterization of prior inclusion probabilities. We empirically demonstrate that our proposed approach outperforms the edge selection method in computational complexity with similar or better predictive performance.

The structured sparsity (e.g. node sparsity) in deep neural networks provides low latency inference, higher data throughput, and reduced energy consumption. Alternatively, there is a vast albeit growing literature demonstrating shrinkage efficiency and theoretical optimality in linear models of two sparse parameter estimation techniques: lasso and horseshoe. In Chapter 4, we propose structurally sparse Bayesian neural networks which systematically prune excessive nodes with (i) Spike-and-Slab Group Lasso, and (ii) Spike-and-Slab Group Horseshoe priors, and develop computationally tractable variational inference We demonstrate the competitive performance of our proposed models compared to the Bayesian baseline models in prediction accuracy, model compression, and inference latency.

Deep neural network ensembles that appeal to model diversity have been used successfully to improve predictive performance and model robustness in several applications. However, most ensembling techniques require multiple parallel and costly evaluations and have been proposed primarily with deterministic models. In Chapter 5, we propose sequential ensembling of dynamic Bayesian neural subnetworks to generate diverse ensemble in a single forward pass. The ensembling strategy consists of an exploration phase that finds high-performing regions of the parameter space and multiple exploitation phases that effectively exploit the compactness of the sparse model to quickly converge to different minima in the energy landscape corresponding to high-performing subnetworks yielding diverse ensembles. We empirically demonstrate that our proposed approach surpasses the baselines of the dense frequentist and Bayesian ensemble models in prediction accuracy, uncertainty estimation, and out-of-distribution robustness. Furthermore, we found that our approach produced the most diverse ensembles compared to the approaches with a single forward pass and even compared to the approaches with multiple forward passes in some cases.

To Dr. Rahman for introducing me to the Bayesian way

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank the people who have extended their support and assistance throughout my Ph.D. journey.

First, I would like to express my deepest gratitude to my advisors, Dr. Tapabrata Maiti and Dr. Shrijita Bhattacharya, for their support, guidance, and encouragement. Dr. Maiti's exemplary commitment to research will continue to serve as a guide in my own academic pursuits. Thank you, Dr. Bhattacharya, for consistently providing me with hands-on help in my research.

I would also like to extend my sincere appreciation to the rest of my dissertation committee members, Dr. Yuehua Cui and Dr. Andrew Finley, for providing me a careful review of my dissertation.

I would like to extend many thanks to my research collaborators, Dr. Sandeep Madireddy, Dr. Zichao Di, and Dr. Prasanna Balaprakash from Argonne National Laboratory for providing me the opportunity to work on developing practical models useful for scientific problems and allowing me to participate in exciting interdisciplinary projects at Argonne. Moreover, I want to thank Dr. Sandeep Madireddy and Dr. Prasanna Balaprakash for funding my research throughout the final year of my Ph.D.

My sincere thanks to the Michigan State University Graduate School for awarding me the Dissertation Completion Fellowship during the Summer 2022.

Lastly, I am grateful to my family. Thanks to my mother for her unequivocal support during all my studies. Thanks to my sisters, Smita and Sanchita, and brother-in-law, Sampanna, for always cheering me on throughout my dissertation.

TABLE OF CONTENTS

LIST OF TABLES ix

LIST OF FIGURES x

LIST OF ALGORITHMS xi

CHAPTER 1 INTRODUCTION 1
1.1 Neural Networks 1
1.1.1 Feedforward Neural Networks 1
1.1.2 Bayesian Neural Networks 4
1.2 Markov Chain Monte Carlo Sampling 5
1.2.1 Metropolis-Hastings Algorithm 5
1.2.2 Gibbs Sampling Algorithm 6
1.3 Variational Bayesian Inference 7
1.4 Posterior Consistency 8
1.5 Dissertation Outline 11

CHAPTER 2 BAYESIAN QUANTILE REGRESSION NEURAL NETWORKS . . 13
2.1 Introduction 13
2.2 Bayesian Quantile Regression 16
2.3 Bayesian Quantile Regression Neural Networks 18
2.3.1 Model 18
2.3.2 Algorithm 20
2.4 Theoretical Results 22
2.5 Numerical Experiments 27
2.5.1 Simulation Studies 27
2.5.2 Real Data Examples 32
2.6 Conclusion and Discussion 35
APPENDICES 36
APPENDIX A LEMMAS FOR POSTERIOR CONSISTENCY PROOF . . . 37
APPENDIX B POSTERIOR CONSISTENCY THEOREM PROOFS 43

CHAPTER 3 LAYER ADAPTIVE NODE SELECTION IN BAYESIAN NEURAL NETWORKS 51
3.1 Introduction 51
3.2 Nonparametric Modeling: Deep Learning Approach 55
3.3 Spike-and-Slab Independent Gaussian Node Selection 57
3.3.1 Model 57
3.3.2 Algorithm 60
3.4 Theoretical Results 61
3.5 Numerical Experiments 69
3.5.1 Simulation Study - I 70

3.5.2 Simulation Study - II . . . 71
3.5.3 UCI Regression Datasets . . . 72
3.5.4 Image Classification Datasets . . . 73
3.6 Conclusion and Discussion . . . 78
APPENDICES . . . 80
APPENDIX A PROOFS OF SS-IG THEORETICAL RESULTS . . . 81
APPENDIX B ADDITIONAL NUMERICAL EXPERIMENTS DETAILS . . . 102

CHAPTER 4 COMPACT BAYESIAN NEURAL NETWORKS WITH STRUCTURED SPARSITY . . . 106
4.1 Introduction . . . 106
4.1.1 Proposed Methods . . . 107
4.2 Structured Sparsity: Spike-and-Slab Hierarchical Priors . . . 108
4.2.1 Spike-and-Slab Group Lasso (SS-GL): . . . 108
4.2.2 Spike-and-Slab Group Horseshoe (SS-GHS): . . . 110
4.2.3 Algorithm and Computational Details . . . 112
4.3 Numerical Experiments . . . 113
4.3.1 MLP MNIST Classification . . . 114
4.3.2 LeNet-5-Caffe Experiments . . . 117
4.3.3 Residual Network Experiments . . . 119
4.4 Conclusion and Discussion . . . 121
APPENDIX . . . 123

CHAPTER 5 SEQUENTIAL BAYESIAN NEURAL SUBNETWORK ENSEMBLES 125
5.1 Introduction . . . 125
5.1.1 Related Work . . . 127
5.2 Sequential Bayesian Neural Subnetwork Ensembles . . . 129
5.2.1 Base Learner . . . 129
5.2.2 Sequential Ensembling and Bayesian Neural Subnetworks . . . 129
5.3 Numerical Experiments . . . 132
5.4 Sequential BNN Ensemble Analysis . . . 135
5.4.1 Function Space Analysis . . . 135
5.4.2 Dynamic Sparsity Learning . . . 137
5.4.3 Effect of Ensemble size . . . 137
5.5 Conclusion and Discussion . . . 138
APPENDICES . . . 140
APPENDIX A REPRODUCIBILITY CONSIDERATIONS . . . 141
APPENDIX B OUT-OF-DISTRIBUTION EXPERIMENT RESULTS . . . 144
APPENDIX C EFFECT OF THE ENSEMBLE SIZE . . . 145
APPENDIX D EFFECT OF THE MONTE CARLO SAMPLE SIZE . . . 146
APPENDIX E EFFECT OF THE PERTURBATION FACTOR . . . 148
APPENDIX F EFFECT OF THE CYCLIC LEARNING RATE SCHEDULE 150

CHAPTER 6 EPILOGUE . . . 153
6.1 Summary . . . 153

6.2 Broader Impacts 153
6.3 Future Research 154

BIBLIOGRAPHY 155

LIST OF TABLES

Table 2.1 Simulation study I results 30

Table 2.2 Simulation study II results 31

Table 2.3 Real data applications results 34

Table 3.1 Simulation study II results 72

Table 3.2 UCI regression datasets results 73

Table 4.1 ResNet-20/CIFAR-10 and ResNet-32/CIFAR-10 experiments results . . . 121

Table 5.1 ResNet-32/CIFAR10 experiment results 134

Table 5.2 ResNet-56/CIFAR100 experiment results 134

Table 5.3 Diversity metrics in ResNet-32/CIFAR-10 and ResNet-56/CIFAR100 experiments 135

Table B.1 OoD detection results in ResNet-32/CIFAR10 experiment 144

Table C.1 Ensemble size effect results in ResNet-32/CIFAR10 experiment 145

Table D.1 Monte Carlo sample size effect results in ResNet-32/CIFAR10 experiment 147

Table E.1 Perturbation factor effect results in ResNet-32/CIFAR10 experiment . . . 149

Table E.2 Diversity metrics for models trained with different perturbation factors in ResNet-32/CIFAR-10 experiment 149

Table F.1 Cyclic learning rate schedules results in ResNet-32/CIFAR10 experiment . 152

Table F.2 Diversity metrics for models trained with different cyclic learning rate schedules in ResNet-32/CIFAR10 experiment 152

LIST OF FIGURES

Figure 1.1 Single-layer neural network 2

Figure 1.2 ReLU and SiLU activations 3

Figure 2.1 Quantile loss function 17

Figure 3.1 Sparse neural network with node selection 54

Figure 3.2 Simulation study I results 71

Figure 3.3 MLP/MNIST and MLP/Fashion-MNIST experiments results 76

Figure 3.4 LeNet-5-Caffe/MNIST and LeNet-5-Caffe/Fashion-MNIST experiments results 77

Figure B.1 Simulation study I: additional experiment results 104

Figure B.2 MNIST experiment results for varying hidden layer widths 105

Figure 4.1 MNIST experiment results: motivation for group shrinkage priors over Gaussian prior 107

Figure 4.2 SS-GL penalty parameter choice experiment results 115

Figure 4.3 SS-GHS regularization constant choice experiment results 116

Figure 4.4 MLP/MNIST experiment results 117

Figure 4.5 LeNet-5-Caffe/MNIST and LeNet-5-Caffe/Fashion-MNIST experiments results 118

Figure 5.1 Training trajectories of base learners in ResNet32/CIFAR10 experiment . 136

Figure 5.2 Dynamic sparsity and FLOPs curves 137

Figure 5.3 Predictive performance results of the base learners and the sequential ensembles as the ensemble size varies in ResNet32/CIFAR10 experiment 138

Figure F.1 Cyclic learning rate schedules 151

LIST OF ALGORITHMS

Algorithm 3.1 Variational inference in SS-IG Bayesian neural networks 61

Algorithm 4.1 Variational inference in SS-GL and SS-GHS Bayesian neural networks 113

Algorithm 5.1 Sequential Bayesian neural subnetwork ensemble (SeBayS) algorithm 131

CHAPTER 1

INTRODUCTION

Artificial neural networks (ANN) are biologically inspired predictive models involving computations and mathematics, which simulate the human–brain processes. Many of the recent successes of the artificial intelligence such as image and voice recognition, robotics, are powered by ANNs. However, ANNs still suffer from many fundamental issues from the perspective of statistical modeling. One of the major challenges is their ability to model the uncertainty, and hence build reliable and robust models while capturing complex data dependencies and being computationally tractable. Probabilistic approaches and especially the systematic Bayesian framework provides an exciting avenue to address this challenge. In this dissertation, we propose novel Bayesian neural network models which are theoretically consistent along with their computationally efficient implementations for the model inference.

In this Chapter, we briefly introduce the main concepts which are fundamental part of this dissertation, neural networks and their Bayesian counterpart (Section 1.1), Markov chain Monte Carlo sampling methods (Section 1.2), variational Bayesian inference (Section 1.3), and posterior consistency preliminaries (Section 1.4). In addition, we discuss some existing work that have significant impact to this field. Finally, we provide a brief outline of the rest of the chapters in Section 1.5.

1.1 Neural Networks

1.1.1 Feedforward Neural Networks

Feedforward neural network can approximate any continuous function $f(.) : \mathbb{R}^p \to \mathbb{R}$ arbitrarily well. Neural network tries to simulate the human brain, so it has many layers of "neurons" just like the neurons in our brain. Originated from a multi-layer perceptron (MLP) (Rosenblatt, 1961) which has multiple hidden layers compared to single hidden layer

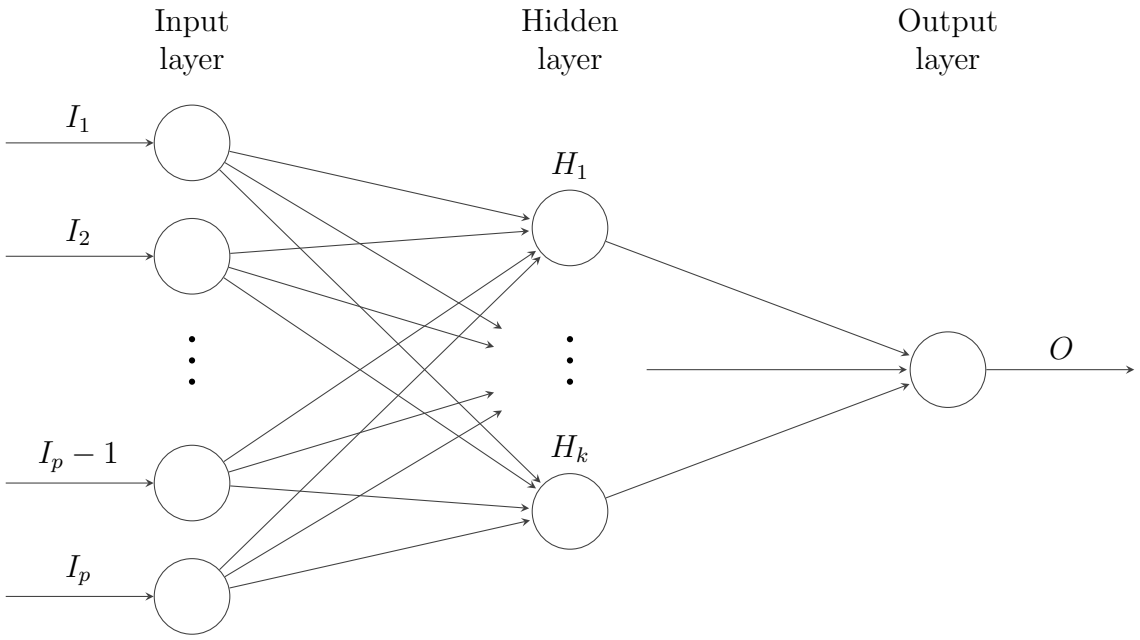


Figure 1.1 **Single-layer neural network.**

counterpart, the perceptron (Rosenblatt, 1957), neural networks are getting deeper to be more accurate in approximating a continuous function. In Figure 1.1 we illustrate a single hidden layer neural network or shallow neural network. The input to the hidden layer consists of a linear combination of the model inputs passed through a non-linear activation function.

Let us consider the input vector $\boldsymbol{x} \in \mathbb{R}^P$. Then the output of the shallow neural network in Figure 1.1 is given by

$$\eta(\boldsymbol{x}) = \beta_0 + \sum_{j=1}^{k} \beta_j \times \psi \left(\gamma_{j0} + \sum_{h=1}^{p} \gamma_{jh} x_h \right)$$

where γ_{jh} $(j = 0, \cdots, k; h = 1, \cdots, p)$ are the input layer (including intercept $h = 0$) to hidden layer weights, β_j $(j = 0, \cdots, k)$ are the hidden layer (including intercept $j = 0$) to output layer weights, and $\psi(.)$ denotes the non-linear activation function. The universal approximation theorem (Cybenko, 1989) states the approximation power of the shallow neural networks.

Theorem 1.1.1 (Universal approximation theorem)**.** *Let $\psi(.)$ be such that $\psi(t) \to 0$ as $t \to -\infty$ and $\psi(t) \to 1$ as $t \to \infty$. Then for a continuous function f on $[0,1]^p$ and an arbitrary*

$\epsilon > 0$, there exist k and parameters β_j $(j = 0, \cdots, k)$, and γ_{jh} $(j = 0, \cdots, k; h = 1, \cdots, p)$ such that,

$$|f(\boldsymbol{x}) - \eta(\boldsymbol{x})| < \epsilon, \quad \forall x \in [0, 1]^p$$

The universal approximation capacity of neural networks along with available computing power explain the widespread use of deep learning nowadays. In this dissertation, we use the following non-linear activation functions.

- Sigmoid: $\psi(x) = \frac{\exp(x)}{1+\exp(x)}$.
- Rectified Linear Unit (ReLU): $\psi(x) = x_+$.
- Sigmoid Linear Unit (SiLU or Swish): $\psi(x) = x \times \text{Sigmoid}(x)$.

ReLU is one of the most popular activation function used in many deep neural architectures. However, it sometimes suffers from the dead ReLU problem where ReLU neurons become inactive and only output 0 for any input (Lu et al., 2020). We ourselves encounter this problem in our spike-and-slab models and there instead of ReLU we use the SiLU activations (Elfwing et al., 2018; Ramachandran et al., 2017) which unlike ReLU is smooth and nonmonotonic (Figure 1.2).

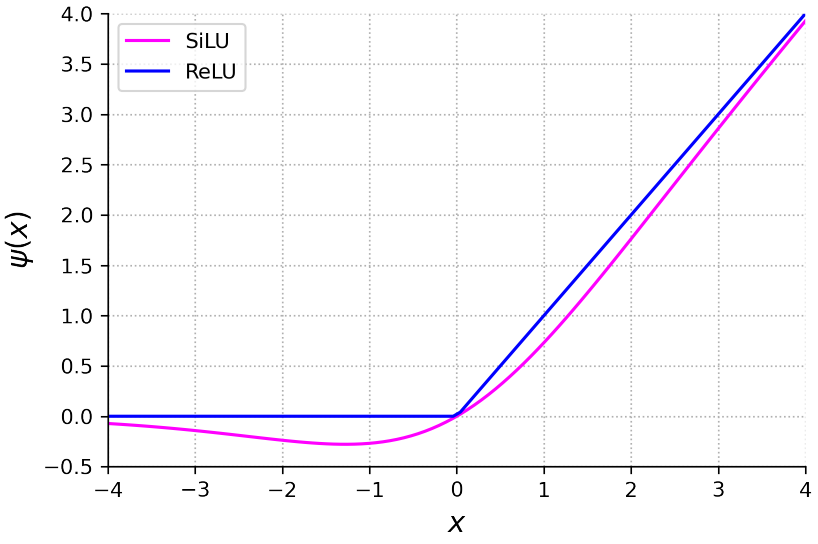


Figure 1.2 **ReLU and SiLU activations.**

1.1.2 Bayesian Neural Networks

Bayesian neural networks (BNN) differ from deterministic neural networks in that their weights are assigned a probability distribution instead of a single value or point estimate. These probability distributions describe the uncertainty in weights and can be used to estimate uncertainty in predictions.

A neural network model can be viewed as a probabilistic model: $p(y|\boldsymbol{x}, \boldsymbol{\theta})$ where $\boldsymbol{\theta}$ denotes the neural network weights. For classification, y is a set of classes and $p(y|\boldsymbol{x}, \boldsymbol{\theta})$ is a categorical distribution. For regression, y is a continuous variable and $p(y|\boldsymbol{x}, \boldsymbol{\theta})$ is a Gaussian distribution.

In Bayesian framework, instead of optimizing over a single probabilistic model, $p(y|\boldsymbol{x}, \boldsymbol{\theta})$, we discover all likely models via posterior inference over model parameters. First, we place a prior distribution $p(\boldsymbol{\theta})$ on the neural network weights $\boldsymbol{\theta}$. The Bayes' rule provides the exact posterior distribution as follows,

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}} \tag{1.1}$$

where $p(\mathcal{D}|\boldsymbol{\theta})$ denotes the likelihood of $\mathcal{D}$ given the model parameters $\boldsymbol{\theta}$.

The main goal of the neural network is predictions on the new inputs. Given the posterior in (1.1) we predict the label corresponding to a new example x_{new} by Bayesian model averaging:

$$p(y_{\text{new}}|x_{\text{new}}, \mathcal{D}) = \int p(y_{\text{new}}|x_{\text{new}}, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}$$

The key distinguishing property of Bayesian approach from deterministic one is marginalization instead of optimization, where we represent solutions given by all settings of parameters weighted by their posterior probabilities, rather than bet everything on a single setting of parameters (Wilson and Izmailov, 2020). Bayesian procedures adapted for deep learning have received widespread attention and applications of BNNs are found in several fields e.g., computer vision (Kendall and Gal, 2017), civil engineering (Bateni et al., 2007; Arangio

and Bontempi, 2015), astronomy (Perreault Levasseur et al., 2017; Cobb et al., 2019), and medicine (Kwon et al., 2020; Beker et al., 2020).

1.2 Markov Chain Monte Carlo Sampling

Markov chain Monte Carlo (MCMC) methods have been used in several Physics problems for many years and later have been widely applied to Bayesian statistical modeling (Neal, 1996). MCMC methods do not make any assumptions regarding the form of the distribution to be sampled, for instance whether a given distribution can be approximated by Gaussian. Ideally, they are supposed to cover all the modes of a target distribution during sampling. However, the high computational complexity of MCMC methods is their major disadvantage in complex Bayesian models and large scale datasets.

In what follows, we describe two well-known MCMC sampling algorithms. the combination of these two algorithm is used in Chapter 2 for posterior inference in BQRNN model.

1.2.1 Metropolis-Hastings Algorithm

One of the most common algorithms for sampling from the posterior $p(\theta|\mathcal{D})$ is the Metropolis-Hastings (MH) algorithm (Metropolis et al., 1953; Hastings, 1970). In the Markov chain defined by the MH algorithm, the new state $\theta^{(t+1)}$ is generated given the current state $\theta^{(t)}$ by first sampling a candidate state θ^* from a proposal density g_θ and subsequently accepting the proposed candidate state with probability

$$\begin{cases} \min\left\{\frac{p(\theta^*|\mathcal{D})g_{\theta^*}(\theta^{(t)})}{p(\theta^{(t)}|\mathcal{D})g_{\theta^{(t)}}(\theta^*)}, 1\right\} & \text{if } p(\theta^{(t)}|\mathcal{D})g_{\theta^{(t)}}(\theta^*) > 0, \\ 1 & \text{otherwise.} \end{cases}$$

By taking a sufficient number of trial steps all of state space is explored and the MH algorithm ensures that the points are distributed according to the required target distribution.

Typically, the proposal distribution is chosen to be symmetrical, satisfying the condition $g_{\theta'}(\theta) = g_\theta(\theta')$. Hence, the acceptance probability simplifies to $\min\{p(\theta^*|\mathcal{D})/p(\theta^{(t)}|\mathcal{D}), 1\}$ which yields the so called *random walk metropolis algorithm*. Commonly used symmetrical

proposal distribution is Gaussian with density $g_\theta = N(\theta, \Sigma)$ where Σ is a constant covariance matrix. In our BQRNN model, we use the normal proposal density for the parameters sampled using MH algorithm.

1.2.2 Gibbs Sampling Algorithm

The Gibbs algorithm was formally described by Geman and Geman (1984) and later Gelfand and Smith (1990) showed its potential in a wide variety of conventional statistical problems. In its basic version, Gibbs sampling is a special case of the MH algorithm.

The point of Gibbs sampling is that given a multivariate distribution over parameters $\boldsymbol{\theta} = \{\theta_1, \cdots, \theta_p\}$, it is simpler to sample from a conditional distribution than to marginalize by integrating over a joint distribution. This allows us to simulate a Markov chain in which $\boldsymbol{\theta}^{(t+1)}$ is generated from $\boldsymbol{\theta}^{(t)}$ as follows:

$$
\begin{aligned}
&\text{Pick} \quad \theta_1^{(t+1)} \text{ from the distribution of } \theta_1 \text{ given } \theta_2^{(t)}, \theta_3^{(t)}, \cdots, \theta_p^{(t)} \\
&\text{Pick} \quad \theta_2^{(t+1)} \text{ from the distribution of } \theta_2 \text{ given } \theta_1^{(t+1)}, \theta_3^{(t)}, \cdots, \theta_p^{(t)} \\
&\qquad\qquad\qquad\qquad \vdots \\
&\text{Pick} \quad \theta_j^{(t+1)} \text{ from the distribution of } \theta_j \text{ given } \theta_1^{(t+1)}, \cdots, \theta_{j-1}^{(t+1)}, \theta_{j+1}^{(t)} \cdots, \theta_p^{(t)} \\
&\qquad\qquad\qquad\qquad \vdots \\
&\text{Pick} \quad \theta_p^{(t+1)} \text{ from the distribution of } \theta_p \text{ given } \theta_1^{(t+1)}, \theta_2^{(t+1)}, \cdots, \theta_{p-1}^{(t+1)}
\end{aligned}
$$

The samples obtained using above procedure for all the parameters together approximate their joint distribution. The marginal distribution of any subset of variables can be approximated by simply considering the samples for that subset of variables while ignoring the rest. The expected value of any variable can be approximated by averaging over all the samples obtained from above procedure.

Although Gibbs sampling is easier to implement, it is only useful when the posterior distribution of one parameter conditional on given values of the other parameters has a known distributional form. For many conventional statistical problems, these conditional distributions are of standard forms, hence efficient Gibbs sampling procedures are implemented with

ease. On the contrary in neural networks, the conditional posterior distribution for hidden layer weights in the network given values for the rest of the weights can be extremely messy, with multiple modes. This is the case we encounter for hidden layer weights in our BQRNN model where we implement combination of random walk MH and Gibbs sampling algorithm for posterior inference.

1.3 Variational Bayesian Inference

Although, Markov chain Monte Carlo sampling is the gold standard for inference in Bayesian models, it is computationally inefficient (Izmailov et al., 2021). As an alternative, variational inference or variational approximation tends to be faster and scales well on complex Bayesian learning tasks with large datasets (Blei et al., 2017). Variational Bayesian (VB) learning recasts the sampling problem to an optimization problem minimizing Kullback-Leibler (KL) divergence which intuitively measures the dissimilarity between a surrogate distribution (called a variational distribution) $q(\boldsymbol{\theta})$ and the true posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ (Jordan et al., 1999).

Definition 1.3.1 (Kullback-Leibler (KL) divergence)**.** *For two probability measures P_1 and P_2 over a set $\mathcal{X}$ such that P_1 is absolutely continuous with respect to P_2, the KL divergence of P_1 with respect to P_2 is defined as*

$$d_{KL}(P_1, P_2) = \int_{\mathcal{X}} \log\left(\frac{dP_1}{dP_2}\right) dP_1$$

where dP_1/dP_2 is the Radon–Nikodym derivative of P_1 with respect to P_2. If P_1 is not absolutely continuous with respect to P_2 then $d_{KL}(P_1, P_2) = \infty$.

As a first step in variational learning, we define a family of surrogate distributions, also called *variational family* ($\mathcal{Q}$) which consists of distributions of simpler form than the true posterior (1.1) (ex. a family of Gaussian distributions).

$$\mathcal{Q} = \{q(\boldsymbol{\theta}, \boldsymbol{\nu}) : q \text{ is a simple candidate distribution used for approximation.}\},$$

where $\boldsymbol{\nu}$ denotes the parameters of the variational distribution, also known as variational parameters. For instance, if $\mathcal{Q}$ is a Gaussian family then $\boldsymbol{\nu}$ includes the mean ($\boldsymbol{\mu}$) and standard deviation ($\boldsymbol{\sigma}$) of a Gaussian candidate distribution.

Once we select an appropriate variational family, the VB infers the parameters of a distribution on the model parameters $q(\boldsymbol{\theta})$ that minimises the Kullback-Leibler (KL) distance from the true posterior $p(\boldsymbol{\theta}|\mathcal{D})$:

$$q^*(\boldsymbol{\theta}) = \underset{q(\boldsymbol{\theta})\in\mathcal{Q}}{\operatorname{argmin}}\ d_{\,\mathrm{KL}}(q(\boldsymbol{\theta}), p(\boldsymbol{\theta}|\mathcal{D})) \tag{1.2}$$

We simplify the $d_{\,\mathrm{KL}}(q(\boldsymbol{\theta}), p(\boldsymbol{\theta}|\mathcal{D}))$:

$$\begin{aligned} d_{\,\mathrm{KL}}(q(\boldsymbol{\theta}), p(\boldsymbol{\theta}|\mathcal{D})) &= \mathbb{E}_{q(\boldsymbol{\theta})}\left[\log q(\boldsymbol{\theta}) - \log p(\boldsymbol{\theta}|\mathcal{D})\right] \\ &= -\mathbb{E}_{q(\boldsymbol{\theta})}\left[\log p(\boldsymbol{\theta}, \mathcal{D}) - \log q(\boldsymbol{\theta})\right] + \log m(\mathcal{D}) \\ &= -\mathbb{E}_{q(\boldsymbol{\theta})}\left[\log p(\mathcal{D}|\boldsymbol{\theta})\right] + d_{\mathrm{KL}}(q(\boldsymbol{\theta}), p(\boldsymbol{\theta})) + \log m(\mathcal{D}) \end{aligned}$$

where $m(\mathcal{D})$ is the marginal distribution of the data and is free of the $\boldsymbol{\theta}$. Hence, the optimization problem in (1.2) is equivalent to minimizing the negative evidence lower bound (ELBO), which is defined as

$$\mathcal{L} = -\mathbb{E}_{q(\boldsymbol{\theta})}\left[\log p(\mathcal{D}|\boldsymbol{\theta})\right] + d_{\mathrm{KL}}(q(\boldsymbol{\theta}), p(\boldsymbol{\theta})),$$

where the first term is the data-dependent cost widely known as the negative log-likelihood (NLL), and the second term is prior-dependent and serves as regularization. Since, the direct optimization of (3.9) is computationally prohibitive, gradient descent methods are used (Kingma and Welling, 2014).

1.4 Posterior Consistency

In Bayesian analysis, one starts with a prior distribution (either informative or non-informative) on the model parameters and updates the knowledge of the model as the number of data observations grows, reflected in the posterior distribution. It is therefore important to know whether the posterior distribution concentrates on neighborhoods of the true data

generating distribution as the data is collected indefinitely. This is known as the Bayesian consistency of the posterior distribution. Although it is an asymptotic property, consistency is one of the benchmarks since the violation of consistency is clearly undesirable and one may have serious doubts against inferences based on an inconsistent posterior distribution.

Let P denote the prior distribution and $\{P_n(.|\mathcal{D}^n)\}$ denote a sequence of posterior distributions where $P_n(.|\mathcal{D}^n)$ is the posterior distribution based on the n-th data sample. Then we define the posterior consistency as follows

Definition 1.4.1 (Posterior consistency.)**.** *The sequence of posteriors is consistent at* $\boldsymbol{\theta}_0$ *if* $\{P_n(U|\mathcal{D}^n)\} \to 1$ *a.s.* $P^{\infty}_{\boldsymbol{\theta}_0}$ *for all neighborhoods* U *of* $\boldsymbol{\theta}_0$*.*

where $P^{\infty}_{\boldsymbol{\theta}_0}$ is the joint distribution of $\{D_i\}_{i=1}^{\infty}$ when $\boldsymbol{\theta}_0$ is the true value of $\boldsymbol{\theta}$.

Alternatively, let $f_0(\boldsymbol{x})$ be the underlying density of $\boldsymbol{X}$. Let $\mathbb{E}(Y|\boldsymbol{X} = \boldsymbol{x}) = \nu_0(\boldsymbol{x})$ be the true regression function of Y given $\boldsymbol{X}$, and let $\hat{\nu}_n(\boldsymbol{x})$ be the estimated regression function.

Definition 1.4.2. $\hat{\nu}_n(\boldsymbol{x})$ *is asymptotically consistent for* $\nu_0(\boldsymbol{x})$ *if*

$$\int \left(\hat{\mu}_n(\boldsymbol{x}) - \mu_0(\boldsymbol{x})\right)^2 \, f_0(\boldsymbol{x}) \, d\boldsymbol{x} \xrightarrow{p} 0$$

where p above the arrow denotes the convergence in probability. In this frequentist sense, Funahashi (1989) and Hornik et al. (1989) have established that the neural networks are asymptotically consistent by showing the existence of some neural network, $\hat{\nu}_n(\boldsymbol{x})$, whose mean squared error with the true function, $\nu_0(\boldsymbol{x})$, converges to 0 in probability.

Lee (2000) showed that the posterior distribution for feedforward (single-layer) sigmoidal neural networks is consistent. Their proof of consistency embedded the problem in a density function estimation, which uses bounds on the bracketing entropy to show that the posterior is consistent over Hellinger neighborhoods. Mathematically, let $f_0(\boldsymbol{x}, y)$ denote the true joint density function of $\boldsymbol{X}$ and Y random variables, and let $f(\boldsymbol{x}, y)$ be the corresponding density function under the neural network model. The Hellinger distance between these two density

functions is given by

$$d_{\mathrm{H}}(f, f_0) = \sqrt{\int\int \left(\sqrt{f(\boldsymbol{x}, y)} - \sqrt{f_0(\boldsymbol{x}, y)}\right)^2 \, \mathrm{d}\boldsymbol{x} \, dy}$$

Definition 1.4.3. *The posterior is asymptotically consistent for f_0 over ϵ-Hellinger neighborhoods $\forall \epsilon > 0$ if,*

$$P(\{f : d_H(f, f_0) \leq \epsilon\} | (\boldsymbol{X_1}, Y_1), \ldots, (\boldsymbol{X_n}, Y_n)) \xrightarrow{p} 1$$

We use this framework to establish the Bayesian posterior consistency in our BQRNN model in Chapter 2 where we combine the ideas provided by Lee (2000) and Sriram et al. (2013). Specifically, Sriram et al. (2013) established the posterior consistency of the Bayesian quantile regression under misspecified ALD model.

In variational Bayesian inference, Zhang and Gao (2020) studied contraction rates of the variational posterior distributions for nonparametric and high-dimensional inference. They provided the conditions on the prior, the likelihood and the variational family that characterize the contraction rates. Similar to the "prior mass and testing" conditions considered in the past literature (Schwartz, 1965), they found the contraction rate to be the sum of two terms. The first term stands for the contraction rate of the true Bayesian posterior distribution, and the second term is contributed by the variational approximation error.

Bhattacharya and Maiti (2021) used the framework provided by Zhang and Gao (2020) and established the conditions needed for the variational posterior consistency of the single-layer Bayesian neural networks. They establish that a simple Gaussian mean-field approximation is good enough to achieve the variational posterior consistency. In this direction, they show that ϵ- Hellinger neighborhood of the true density function receives close to 1 probability under the variational posterior. We use the similar framework, in our SS-IG model to establish the consistency of the variational posterior in Chapter 3.

1.5 Dissertation Outline

The main theme of this dissertation is the development of asymptotically consistent Bayesian neural network models tailored for different scenarios. Each chapter forms a separate manuscript which is either published or under review. For ease of readability, we separate appendices of each chapter providing them after the main content in each chapter.

Chapter 2 is based on our published paper titled "Quantile Regression Neural Networks: A Bayesian Approach"[1]. In this chapter, we present the Bayesian quantile regression neural network (BQRNN) and provide a Metropolis-Hastings coupled with Gibbs sampling algorithm for posterior inference. We establish the posterior consistency in our proposed model and present a set of simulation examples as well as real-data applications demonstrating the efficacy of our method. The proofs of the requisite lemmas and posterior consistency theorems are discussed in the chapter appendices.

Chapter 3 is based on our manuscript titled "Layer Adaptive Node Selection in Bayesian Neural Networks: Statistical Guarantees and Implementation Details"[2]. In this chapter, we develop spike-and-slab Gaussian node selection model and provide a variational algorithm for posterior inference. We derive the variational posterior consistency and its contraction rate for any generic shaped network structure. We measure the computational gains achieved by our approach using layer-wise node sparsities for shallow models and floating point operations in larger models. We also discuss the memory efficiency and computational speedup trade-off between edge selection and node selection approach during test time. The proofs of the lemmas required to establish the variational posterior consistency as well as additional numerical experiment details are presented in the chapter appendices.

Chapter 4 is based on our manuscript titled "Compact Bayesian Neural Networks with Structured Sparsity"[3]. In this chapter, we propose structurally sparse Bayesian neural net-

[1] Adpated by permission from **Springer Nature**: *Journal of Statistical Theory and Practice*, (Jantre et al., 2021b), License No: 5352561465807.

[2] The revision is currently under review (Jantre et al., 2021a).

[3] The manuscript is currently under preparation.

works using two distinct spike and slab prior setups, where the slab component uses hierarchical priors on the group of incoming weights on the neurons: (i) Spike-and-Slab Group Lasso (SS-GL), and (ii) Spike-and-Slab Group HorseShoe (SS-GHS). The chapter appendix discusses additional numerical experiment details.

Chapter 5 is based on our manuscript titled "Sequential Bayesian Neural Subnetwork Ensembles"[4]. In this chapter, we propose a sequential ensembling strategy for Bayesian neural networks (BNNs) which learns multiple subnetworks in a single forward-pass. We combine the strengths of the automated sparsity-inducing spike-and-slab prior that allows dynamic pruning during training, which produces structurally sparse BNNs, and the proposed sequential ensembling strategy to efficiently generate diverse and sparse Bayesian neural networks. Reproducibility considerations and additional numerical experiment details are presented in the chapter appendices.

In Chapter 6, we summarize the work we have done in this dissertation and discuss the likely future methodological and theoretical extensions of our current work. We also provide fully documented Python codes that reproduce all the results in this dissertation and can be easily modified and used by practitioners in chapter specific public repositories [5].

[4]The manuscript is currently under revision (Jantre et al., 2022). This work is a collaborative work with my Argonne mentors, Dr. Madireddy and Dr. Balaprakash.

[5]https://github.com/jsanket123

CHAPTER 2

BAYESIAN QUANTILE REGRESSION NEURAL NETWORKS

2.1 Introduction

Quantile regression (QR), proposed by Koenker and Basset (1978), models conditional quantiles of the dependent variable as a function of the covariates. The method supplements the least squares regression and provides a more comprehensive picture of the entire conditional distribution. This is particularly useful when the relationships in the lower and upper tail areas are of greater interest. Quantile regression has been extensively used in wide array of fields such as economics, finance, climatology, and medical sciences, among others (Koenker, 2005). Quantile regression estimation requires specialized algorithms and reliable estimation techniques which are available in both frequentist and Bayesian literature. Frequentist techniques include simplex algorithm (Dantzig, 1963) and the interior point algorithm (Karmarkar, 1984), whereas Bayesian technique using Markov chain Monte Carlo (MCMC) sampling was first proposed by Yu and Moyeed (2001). Their approach employed the asymmetric Laplace distribution (ALD) for the response variable, which connects to frequentist quantile estimate, since its maximum likelihood estimates are equivalent to the quantile regression using check-loss function (Koenker and Machado, 1999). Recently, Kozumi and Kobayashi (2011) proposed a Gibbs sampling algorithm, where they exploit the normal-exponential mixture representation of the asymmetric Laplace distribution which considerably simplified the computation for Bayesian quantile regression models.

Artificial neural networks are helpful in estimating possibly non-linear models without specifying an exact functional form. The neural networks which are most widely used in engineering applications are the single hidden-layer feedforward neural networks. These networks consist of a set of inputs $\boldsymbol{X}$, which are connected to each of k hidden nodes, which, in turn, are connected to an output layer ($\boldsymbol{O}$). In a typical single layer feedforward neural

network, the outputs are computed as

$$\boldsymbol{O}_i = b_0 + \sum_{j=1}^{k} b_j \psi \left(c_{j0} + \sum_{h=1}^{p} \boldsymbol{X}_{ih} c_{jh} \right)$$

where, c_{jh} is the weight from input $\boldsymbol{X}_{ih}$ to the hidden node j. Similarly, b_j is the weight associated with the hidden unit j. The c_{j0} and b_0 are the biases for the hidden nodes and the output unit. The function $\psi(.)$ is a nonlinear activation function. Some common choices of $\psi(.)$ are the sigmoid and the hyperbolic tangent functions. The interest in neural networks is motivated by the universal approximation capability of feedforward neural networks (FNNs) (Cybenko, 1989; Funahashi, 1989; Hornik et al., 1989). According to these authors, standard feedforward neural networks with as few as one hidden layer whose output functions are sigmoid functions are capable of approximating any continuous function to a desired level of accuracy, if the number of hidden layer nodes are sufficiently large. Taylor (2000) introduced a practical implementation of quantile regression neural networks (QRNN) to combine the approximation ability of neural networks with robust nature of quantile regression. Several variants of QRNN have been developed such as composite QRNN where neural networks are extended to the linear composite quantile regression (Xu et al., 2017) and later Cannon (2018) introduced monotone composite QRNN which guaranteed the non-crossing of regression quantiles.

Bayesian neural network learning models find the predictive distributions for the target values in a new test case given the inputs for that case as well as inputs and targets for the training cases. Early work of Buntine and Weigend (1991) and Mackay (1992) has inspired widespread research in Bayesian neural network models. Their work implemented Bayesian learning using Gaussian approximations. Later, Neal (1996) applied Hamiltonian Monte Carlo in Bayesian statistical applications. Further, Sequential Monte Carlo techniques applied to neural network architectures are described in De Freitas et al. (2001). A detailed review of MCMC algorithms applied to neural networks is presented by Titterington (2004). Although Bayesian neural networks have been widely developed in the context of mean regression models, there has been limited or no work available on its development in connection

to quantile regression both from a theoretical and implementation standpoint. We also note that the literature on MCMC methods applied to neural networks is somewhat limited due to several challenges including lack of parameter identifiability, high computational costs, and convergence failures (Papamarkou et al., 2022).

Our contributions: In this work, we develop the statistical framework for Bayesian quantile regression neural network and study its properties both from theoretical as well as numerical standpoint. The natural advantage of our Bayesian procedure over the frequentist models is that we have posterior variance for our conditional quantile estimates which can be used as an uncertainty quantification. The proposed Bayesian quantile regression neural network (BQRNN) uses a single hidden layer FNN with sigmoid activation function, and a linear output unit. On the numerical front, we have implemented the Bayesian procedure using Gibbs sampling combined with random walk Metropolis-Hastings algorithm. We have shown that our model outperforms Bayesian quantile regression (BQR) in the nonlinear data setup while performing comparably in linear data setup. We use mean squared error to provide empirical justification for the use of our proposed BQRNN model in any given data.

Our theoretical development includes establishment of posterior consistency, an essential property in nonparametric Bayesian statistics, which in turn provides confidence in the use of Bayesian quantile regression neural network models across all disciplines. The posterior consistency of our method makes use of techniques from the works of Lee (2000) and Sriram et al. (2013). The former has shown posterior consistency in the context of Bayesian neural network for mean models while the later has shown it in the case of Bayesian quantile regression. Following the framework of Lee (2000), we prove consistency of the posterior by using universal approximation properties of neural networks as discussed in Funahashi (1989), Hornik et al. (1989) and others. Analogous to the work of Lee (2000), our current works borrow several ideas for establishing consistency in the context of density estimation from Barron et al. (1999). Finally, to handle the case of ALD responses, we use the framework in Sriram et al. (2013)'s which provides a method for handling of ALD in BQR scenario.

The rest of this chapter is organized as follows. Section 2.2 introduces quantile regression and its Bayesian formulation by establishing relationship between quantile regression and asymmetric Laplace distribution. In Section 2.3, we propose the Bayesian quantile regression neural network (BQRNN) model and the prior used in this study. Further, we detail our hierarchical BQRNN model and provide the MCMC procedure which couples Gibbs sampling with random walk Metropolis-Hastings algorithm. Section 2.4 provides an overview of the posterior consistency results for our model. Section 2.5 presents simulation studies and real world applications. A brief discussion and conclusion is provided in Section 2.6. The proofs of the requisite lemmas and posterior consistency theorems are presented in the Appendix A and Appendix B.

2.2 Bayesian Quantile Regression

Quantile regression (Koenker and Basset, 1978) offers a practically important alternative to mean regression by allowing the inference about the conditional distribution of the response variable through modeling of its conditional quantiles. Let Y and $\boldsymbol{X}$ denote the response and the predictors respectively and $\tau \in (0,1)$ be the quantile level of the conditional distribution of Y and $F(.)$ be the cumulative distribution function of Y, then a linear conditional quantile function of Y is denoted as follows

$$Q_\tau(y_i|\boldsymbol{X_i} = \boldsymbol{x_i}) \equiv F^{-1}(\tau) = \boldsymbol{x_i}^T\beta(\tau), \qquad i = 1, \ldots, n,$$

where $\beta(\tau) \in \mathbb{R}^p$ is a vector of quantile specific regression coefficients of length p. The aim of quantile regression is to estimate the conditional quantile function $Q(.)$.

Let us consider the following linear model in order to formally define the quantile regression problem,

$$Y = \boldsymbol{X}^T\beta(\tau) + \varepsilon, \tag{2.1}$$

where ε is the error vector restricted to have its τ^{th} quantile to be zero, i.e. $\int_{-\infty}^{0} f(\varepsilon_i)d\varepsilon_i = \tau$. The probability density of this error is often left unspecified in the classical literature.

The estimation through quantile regression proceeds by minimizing the following objective function

$$\min_{\beta(\tau) \in \mathbb{R}^p} \sum_{i=1}^{n} \rho_\tau(y_i - \boldsymbol{x_i}^T \beta(\tau)) \tag{2.2}$$

where $\rho_\tau(.)$ is the check function or quantile loss function with the following form:

$$\rho_\tau(u) = u.\{\tau - I(u < 0)\}, \tag{2.3}$$

$I(.)$ is an indicator function. which is either 0 or 1 depending on whether it satisfies the given condition or not. This check function is not differentiable at zero, see Figure 2.1.

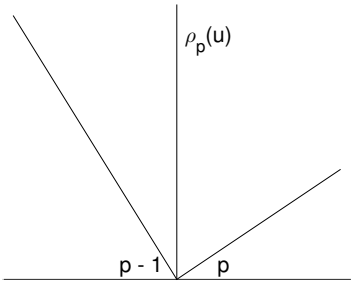


Figure 2.1 **Quantile loss function.**

Classical methods employ linear programming techniques such as the simplex algorithm, the interior point algorithm, or the smoothing algorithm to obtain quantile regression estimates for $\beta(\tau)$ (Madsen and Nielsen, 1993; Chen, 2007). The statistical programming language R makes use of `quantreg` package (Koenker, 2017) to implement quantile regression techniques whilst confidence intervals are obtained via bootstrap (Koenker, 2005).

Median regression in Bayesian setting has been considered by Walker and Mallick (1999) and Kottas and Gelfand (2001). In quantile regression, a link between maximum-likelihood theory and minimization of the sum of check functions, (2.2), is provided by asymmetric Laplace distribution (ALD) (Koenker and Machado, 1999; Yu and Moyeed, 2001). This distribution has location parameter μ, scale parameter σ and skewness parameter τ. Further details regarding the properties of this distribution are specified in Yu and Zhang (2005). If $Y \sim$ ALD(μ, σ, p), then its probability distribution function is given by

$$f(y|\mu, \sigma, \tau) = \frac{\tau(1-\tau)}{\sigma} \exp\left\{-\rho_\tau\left(\frac{y-\mu}{\sigma}\right)\right\}$$

As discussed in Yu and Moyeed (2001), using the above skewed distribution for errors provides a way to implement Bayesian quantile regression effectively. According to them, any reasonable choice of prior, even an improper prior, generates a posterior distribution for $\beta(\tau)$. Subsequently, they made use of a random walk Metropolis Hastings algorithm with a Gaussian proposal density centered at the current parameter value to generate samples from analytically intractable posterior distribution of $\beta(\tau)$.

In the aforementioned approach, the acceptance probability depends on the choice of the value of τ, hence the fine tuning of parameters like proposal step size is necessary to obtain the appropriate acceptance rates for each τ. Kozumi and Kobayashi (2011) overcame this limitation and showed that Gibbs sampling can be incorporated with AL density being represented as a mixture of normal and exponential distributions. Consider the linear model from (2.1), where $\varepsilon_i \sim \text{ALD}(0, \sigma, \tau)$, then this model can be written as

$$y_i = \boldsymbol{x_i}^T \beta(\tau) + \theta v_i + \kappa\sqrt{\sigma v_i} u_i, \qquad i = 1, \ldots, n, \tag{2.4}$$

where, u_i and v_i are mutually independent, with $u_i \sim \text{N}(0,1), v_i \sim \mathcal{E}(1/\sigma)$ and $\mathcal{E}(1/\sigma)$ is the exponential distribution with mean σ. The θ and κ constants in (2.4) are given by

$$\theta = \frac{1-2\tau}{\tau(1-\tau)} \quad and \quad \kappa = \sqrt{\frac{2}{\tau(1-\tau)}}$$

Consequently, a Gibbs sampling algorithm based on normal distribution can be implemented effectively. Currently, `Brq` (Alhamzawi, 2018) and `bayesQR` (Benoit et al., 2017) packages in R provide Gibbs sampler for Bayesian quantile regression. We are employing the same technique to derive Gibbs sampling steps for all except hidden layer node weight parameters for our Bayesian quantile regression neural network model.

2.3 Bayesian Quantile Regression Neural Networks

2.3.1 Model

In this work, we focus on feedforward neural networks with a single hidden layer of units with logistic activation functions, and a linear output unit. Consider the univariate response

variable Y_i and the covariate vector $\boldsymbol{X_i}$ $(i = 1, 2, \ldots, n)$. Further, denote the number of covariates by p and the number of hidden nodes by k which is allowed to vary as a function of n. Denote the input weights by γ_{jh} and the output weights by β_j. Let, $\tau \in (0, 1)$ be the quantile level of the conditional distribution of Y_i given $\boldsymbol{X_i}$ and keep it fixed. Then, the resulting conditional quantile function is denoted as follows

$$\begin{aligned} Q_\tau(y_i|\boldsymbol{X_i} = \boldsymbol{x_i}) &= \beta_0 + \sum_{j=1}^{k} \beta_j \frac{1}{1 + \exp(-\gamma_{j0} - \sum_{h=1}^{p} \gamma_{jh} x_{ih})} \\ &= \beta_0 + \sum_{j=1}^{k} \beta_j \psi(\boldsymbol{x_i^T \gamma_j}) = \boldsymbol{\beta}^T \boldsymbol{\eta_i(\gamma)} = \boldsymbol{L_i \beta} \end{aligned} \tag{2.5}$$

where, $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_k)^T$, $\boldsymbol{x_i} = (1, x_{i1}, \ldots, x_{ip})^T$, $\boldsymbol{\eta_i(\gamma)} = (1, \psi(\boldsymbol{x_i^T \gamma_1}), \ldots, \psi(\boldsymbol{x_i^T \gamma_k}))^T$ and $\boldsymbol{L} = (\boldsymbol{\eta_1(\gamma)}, \ldots, \boldsymbol{\eta_n(\gamma)})^T$, $i = 1, \ldots, n$. $\psi(.)$ is the logistic activation function.

The specified model for Y_i conditional on $\boldsymbol{X_i} = \boldsymbol{x_i}$ is given by $Y_i \sim ALD(\boldsymbol{L_i \beta}, \sigma, \tau)$ with a likelihood proportional to

$$\sigma^{-n} \exp\left\{ -\sum_{i=1}^{n} \frac{|\varepsilon_i| + (2\tau - 1)\varepsilon_i}{2\sigma} \right\} \tag{2.6}$$

where, $\varepsilon_i = y_i - \boldsymbol{L_i \beta}$. The above ALD based likelihood can be represented as a location-scale mixture of normals (Kozumi and Kobayashi, 2011). For any $a, b > 0$, we have the following equality (Andrews and Mallows, 1974)

$$\exp\{-|ab|\} = \int_0^\infty \frac{a}{\sqrt{2\pi v}} \exp\left\{ -\frac{1}{2}(a^2 v + b^2 v^{-1}) \right\} \mathrm{d}v$$

Letting $a = 1/\sqrt{2\sigma}$, $b = \varepsilon/\sqrt{2\sigma}$, and multiplying by $\exp\{-(2\tau - 1)\varepsilon/2\sigma\}$ the (2.6) becomes

$$\sigma^{-n} \exp\left\{ -\sum_{i=1}^{n} \frac{|\varepsilon_i| + (2\tau - 1)\varepsilon_i}{2\sigma} \right\} = \prod_{i=1}^{n} \int_0^\infty \frac{1}{\sigma\sqrt{4\pi\sigma v_i}} \exp\left\{ -\frac{(\varepsilon_i - \xi v_i)^2}{4\sigma v_i} - \zeta v_i \right\} \mathrm{d}v_i \tag{2.7}$$

where, $\xi = (1 - 2\tau)$ and $\zeta = \tau(1 - \tau)/\sigma$. (2.7) is beneficial in the sense that there is no need to worry about the prior distribution of v_i as it is extracted in the same equation. The prior of v_i in (2.7) is exponential distribution with mean ζ^{-1} and it depends on the value of τ.

Further we observe that, the output of the aforementioned neural network remains unchanged under a set of transformations, like certain weight permutations and sign flips

which renders the neural network non-identifiable. For example, in the above model 2.5, take $p, k = 2$ and $\beta_0, \gamma_{j0} = 0$. Then,

$$\sum_{j=1}^{2} \beta_j \psi(\boldsymbol{x_i^T \gamma_j}) = \beta_1 \left[1 + \exp(-\gamma_{11} x_{i1} - \gamma_{12} x_{i2})\right]^{-1} + \beta_2 \left[1 + \exp(-\gamma_{21} x_{i1} - \gamma_{22} x_{i2})\right]^{-1}$$

In the foregoing equation we can notice that when $\beta_1 = \beta_2$, two different sets of values of $(\gamma_{11}, \gamma_{12}, \gamma_{21}, \gamma_{22})$ obtained by flipping the signs, namely (1,2,-1,-2) and (-1,-2,1,2) result in the same value for $\sum_{j=1}^{2} \beta_j \psi(\boldsymbol{x_i^T \gamma_j})$. However, as a special case of lemma 1 of Ghosh et al. (2000), the joint posterior of the parameters is proper if the joint prior is proper, even in the case of posterior invariance under the parameter transformations. Note that, as long as the interest is on prediction rather parameter estimation, this property is sufficient for predictive model building. In this work, we focus only on proper priors, hence the non-identifiability of the parameters in 2.5 doesn't cause any problem.

2.3.2 Algorithm

We take mutually independent priors for $\boldsymbol{\beta}, \boldsymbol{\gamma_1}, \ldots, \boldsymbol{\gamma_k}$ with $\boldsymbol{\beta} \sim N(\boldsymbol{\beta_0}, \sigma_0^2 \boldsymbol{I}_{k+1})$ and $\boldsymbol{\gamma_j} \sim N(\boldsymbol{\gamma_{j0}}, \sigma_1^2 \boldsymbol{I}_{p+1})$, $j = 1, \ldots, k$. Further, we take inverse gamma prior for σ such that $\sigma \sim IG(a/2, b/2)$. Prior selection is problem specific and it is useful to elicit the chosen prior from the historical knowledge. However, for most practical applications, such information is not readily available. Furthermore, neural networks are commonly applied to big data for which a priori knowledge regarding the data as well as about the neural network parameters is not typically known. Hence, prior elicitation from experts in the area is not applicable to neural networks in practice. As a consequence, it seems reasonable to use near-diffuse priors for the parameters of the given model.

Now, the joint posterior for $\boldsymbol{\beta}, \boldsymbol{\gamma}, \sigma, \boldsymbol{v}$ given $\boldsymbol{y}$, is

$$f(\boldsymbol{\beta}, \boldsymbol{\gamma}, \sigma, \boldsymbol{v} | \boldsymbol{y})$$

$$\propto l(\boldsymbol{y} | \boldsymbol{\beta}, \boldsymbol{\gamma}, \sigma, \boldsymbol{v})\, \pi(\boldsymbol{\beta})\, \pi(\boldsymbol{\gamma})\, \pi(\sigma),$$

$$\propto \left(\frac{1}{\sigma}\right)^{\frac{3n}{2}} \left(\prod_{i=1}^{n} v_i\right)^{-\frac{1}{2}} \exp\left\{-\frac{1}{4\sigma}\left[(\boldsymbol{y} - \boldsymbol{L\beta} - \xi \boldsymbol{v})^T \boldsymbol{V} (\boldsymbol{y} - \boldsymbol{L\beta} - \xi \boldsymbol{v})\right] - \frac{\tau(1-\tau)}{\sigma} \sum_{i=1}^{n} v_i\right\}$$

$$\times \exp\left\{-\frac{1}{2\sigma_0^2}(\boldsymbol{\beta}-\boldsymbol{\beta_0})^T(\boldsymbol{\beta}-\boldsymbol{\beta_0})\right\}$$
$$\times \exp\left\{-\frac{1}{2\sigma_1^2}\sum_{j=1}^{k}(\boldsymbol{\gamma_j}-\boldsymbol{\gamma_{j0}})^T(\boldsymbol{\gamma_j}-\boldsymbol{\gamma_{j0}})\right\}$$
$$\times \left(\frac{1}{\sigma}\right)^{\frac{a}{2}+1}\exp\left(-\frac{b}{2\sigma}\right).$$

where, $\boldsymbol{V} = diag(v_1^{-1}, v_2^{-1}, \ldots, v_n^{-1})$. A Gibbs sampling algorithm is used to generate samples from the analytically intractable posterior distribution $f(\boldsymbol{\beta}, \boldsymbol{\gamma}|\boldsymbol{y})$. Some of the full conditionals required in this procedure are available only up to unknown normalizing constants, and we used random walk Metropolis-Hastings algorithm to sample from these full conditional distributions.

These full conditional distributions are as follows:

(a) $\pi(\boldsymbol{\beta}|\boldsymbol{\gamma}, \sigma, \boldsymbol{v}, \boldsymbol{y})$

$$\sim N\left[\left(\frac{\boldsymbol{L}^T\boldsymbol{V}\boldsymbol{L}}{2\sigma}+\frac{\boldsymbol{I}}{\sigma_0^2}\right)^{-1}\left(\frac{\boldsymbol{L}^T\boldsymbol{V}(\boldsymbol{y}-\xi\boldsymbol{v})}{2\sigma}+\frac{\boldsymbol{\beta_0}}{\sigma_0^2}\right), \left(\frac{\boldsymbol{L}^T\boldsymbol{V}\boldsymbol{L}}{2\sigma}+\frac{\boldsymbol{I}}{\sigma_0^2}\right)^{-1}\right]$$

(b) $\pi(\boldsymbol{\gamma_j}|\boldsymbol{\beta}, \sigma, \boldsymbol{v}, \boldsymbol{y})$

$$\propto \exp\left\{-\frac{1}{4\sigma}\left[(\boldsymbol{y}-\boldsymbol{L}\boldsymbol{\beta}-\xi\boldsymbol{v})^T\boldsymbol{V}(\boldsymbol{y}-\boldsymbol{L}\boldsymbol{\beta}-\xi\boldsymbol{v})\right]\right\}$$
$$\times \exp\left\{-\frac{1}{2\sigma_1^2}(\boldsymbol{\gamma_j}-\boldsymbol{\gamma_{j0}})^T(\boldsymbol{\gamma_j}-\boldsymbol{\gamma_{j0}})\right\}$$

(c) $\pi(\sigma|\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{v}, \boldsymbol{y})$

$$\sim IG\left(\frac{3n+a}{2}, \frac{1}{4}\left[(\boldsymbol{y}-\boldsymbol{L}\boldsymbol{\beta}-\xi\boldsymbol{v})^T\boldsymbol{V}(\boldsymbol{y}-\boldsymbol{L}\boldsymbol{\beta}-\xi\boldsymbol{v})\right]+\tau(1-\tau)\sum_{i=1}^{n}v_i+\frac{b}{2}\right)$$

(d) $\pi(v_i|\boldsymbol{\gamma}, \boldsymbol{\beta}, \sigma, \boldsymbol{y})$

$$\sim GIG(\nu, \rho_1, \rho_2) \text{ where, } \nu=\frac{1}{2}, \rho_1^2=\frac{1}{2\sigma}(y_i-\boldsymbol{L_i}\boldsymbol{\beta})^2, \text{ and } \rho_2^2=\frac{1}{2\sigma}.$$

The generalized inverse Gaussian distribution is defined as, if $x \sim GIG(\nu, \rho_1, \rho_2)$ *then the probability density function of* x *is given by*

$$f(x|\nu, \rho_1, \rho_2) = \frac{(\rho_2/\rho_1)^\nu}{2K_\nu(\rho_1\rho_2)}x^{\nu-1}exp\left\{-\frac{1}{2}(x^{-1}\rho_1^2+x\rho_2^2)\right\},$$

where $x > 0, -\infty < \nu < \infty, \rho_1, \rho_2 \geq 0$ *and* $K_\nu(.)$ *is a modified Bessel function of the third kind (see, Barndorff-Nielsen and Shephard (2001)).*

Unlike the parsimonious parametric models, the Bayesian nonparametric models require additional statistical justification for their theoretical validity. For that reason we are going to provide asymptotic consistency of the posterior distribution derived in our proposed neural network model.

2.4 Theoretical Results

Let $(\boldsymbol{x_1}, y_1), \ldots, (\boldsymbol{x_n}, y_n)$ be the given data, let $f_0(\boldsymbol{x})$ be the underlying density of $\boldsymbol{X}$. Let $Q_\tau(y|\boldsymbol{X} = \boldsymbol{x}) = \mu_0(\boldsymbol{x})$ be the true conditional quantile function of Y given $\boldsymbol{X}$, and let $\hat{\mu}_n(\boldsymbol{x})$ be the estimated conditional quantile function.

Definition 2.4.1. $\hat{\mu}_n(\boldsymbol{x})$ *is asymptotically consistent for* $\mu_0(\boldsymbol{x})$ *if*

$$\int |\hat{\mu}_n(\boldsymbol{x}) - \mu_0(\boldsymbol{x})| \ f_0(\boldsymbol{x}) \ d\boldsymbol{x} \xrightarrow{p} 0$$

We are essentially making use of Markov's inequality to ultimately show that $\hat{\mu}_n(\boldsymbol{X}) \xrightarrow{p} \mu_0(\boldsymbol{X})$. In similar frequentist sense, Funahashi (1989) and Hornik et al. (1989) have shown the asymptotic consistency of the neural networks for mean-regression models by showing the existence of some neural network, $\hat{\mu}_n(\boldsymbol{x})$, whose mean squared error with the true function, $\mu_0(\boldsymbol{x})$, converges to 0 in probability.

We consider the notion of posterior consistency for Bayesian non-parametric problems which is quantified by concentration around the true density function (see Wasserman (1998), Barron et al. (1999)). This boils down to the above definition of consistency on the conditional quantile functions. The main idea is that the density functions deal with the joint distribution of $\boldsymbol{X}$ and Y, while the conditional quantile function deals with the conditional distribution of Y given $\boldsymbol{X}$. This conditional distribution can then be used to construct the joint distribution by assuming certain regularity condition on the distribution of $\boldsymbol{X}$. This allows the use of some techniques developed in density estimation field. Some of the

ideas presented here can be found in Lee (2000) which developed the consistency results for non-parametric regression using single hidden-layer feed forward neural networks.

Let the posterior distribution of the parameters $P(.|(\boldsymbol{X_1}, Y_1), \ldots, (\boldsymbol{X_n}, Y_n))$. Let $f(\boldsymbol{x}, y)$ and $f_0(\boldsymbol{x}, y)$ denote the joint density function of $\boldsymbol{x}$ and y under the model and the truth respectively. Indeed, one can construct the joint density $f(\boldsymbol{x}, y)$ from the condition quantile function $f(y|\boldsymbol{x})$ by taking $f(\boldsymbol{x}, y) = f(y|\boldsymbol{x})f(\boldsymbol{x})$ where $f(\boldsymbol{x})$ denotes the underlying density of X. Since, one is only interested in $f(y|\boldsymbol{x})$ and $\boldsymbol{X}$ is ancillary to the estimation of $f(y|\boldsymbol{x})$, one can use some convenient distribution for $f(x)$. Similar to Lee (2000), we define Hellinger neighborhoods of the true density function $f_0(\boldsymbol{x}, y) = f_0(y|\boldsymbol{x})f_0(\boldsymbol{x})$ which allows us to quantify the consistency of the posterior. The Hellinger distance between f_0 and any joint density function f of x and y is defined as follows.

$$D_H(f, f_0) = \sqrt{\int \int \left(\sqrt{f(\boldsymbol{x}, y)} - \sqrt{f_0(\boldsymbol{x}, y)}\right)^2 \, \mathrm{d}\boldsymbol{x} \, dy} \tag{2.8}$$

Based on (2.8), an ϵ-sized Hellinger neighborhood of the true density function f_0 is given by

$$A_\epsilon = \{f : \, D_H(f, f_0) \leq \epsilon\} \tag{2.9}$$

Definition 2.4.2 (Posterior Consistency)**.** *Suppose* $(\boldsymbol{X_i}, Y_i) \sim f_0$*. The posterior is asymptotically consistent for* f_0 *over Hellinger neighborhoods if* $\forall \epsilon > 0$*,*

$$P(A_\epsilon|(\boldsymbol{X_1}, Y_1), \ldots, (\boldsymbol{X_n}, Y_n)) \xrightarrow{p} 1$$

i.e. the posterior probability of any Hellinger neighborhood of f_0 *converges to 1 in probability.*

Similar to Lee (2000), we prove the asymptotic consistency of the posterior for neural networks with number of hidden nodes, k, being function of sample size, n. This sequence of models indexed with increasing sample size is called *sieve*. We take sequence of priors, $\{\pi_n\}$, where each π_n is defined for a neural network with k_n hidden nodes in it. The predictive density (Bayes estimate of f) then be given by

$$\hat{f}_n(.) = \int f(.) \, \mathrm{d}P(f|(\boldsymbol{X_1}, Y_1), \ldots, (\boldsymbol{X_n}, Y_n)) \tag{2.10}$$

Let $\mu_0(\boldsymbol{x}) = Q_{\tau,f_0}(Y|\boldsymbol{X} = \boldsymbol{x})$ be the true conditional quantile function and let $\hat{\mu}_n(\boldsymbol{x}) = Q_{\tau,\hat{f}_n}(Y|\boldsymbol{X} = \boldsymbol{x})$ be the posterior predictive conditional quantile function using a neural network. For notational convenience we are going to drop $\boldsymbol{x}$ and denote these functions as μ_0 and $\hat{\mu}_n$ occasionally.

The following is the key result in this case.

Theorem 2.4.3. *Let the prior for the regression parameters, π_n, be an independent normal with mean 0 and variance σ_0^2 (fixed) for each of the parameters in the neural network. Suppose that the true conditional quantile function is either continuous or square integrable. Let k_n be the number of hidden nodes in the neural network, and let $k_n \to \infty$. If there exists a constant a such that $0 < a < 1$ and $k_n \leq n^a$, then $\int |\hat{\mu}_n(\boldsymbol{x}) - \mu_0(\boldsymbol{x})|\, d\boldsymbol{x} \xrightarrow{p} 0$ as $n \to \infty$*

In order to prove Theorem 2.4.3, we assume that $\boldsymbol{X_i} \sim U(0,1)$, i.e. density function of $\boldsymbol{x}$ is identically equal to 1. This implies joint densities $f(\boldsymbol{x}, y)$ and $f_0(\boldsymbol{x}, y)$ are equal to the conditional density functions, $f(y|\boldsymbol{x})$ and $f_0(y|\boldsymbol{x})$ respectively. Next, we define Kullback-Leibler distance to the true density $f_0(\boldsymbol{x}, y)$ as follows

$$D_K(f_0, f) = \mathbb{E}_{f_0}\left[\log \frac{f_0(\boldsymbol{X}, Y)}{f(\boldsymbol{X}, Y)}\right] \tag{2.11}$$

Based on (2.11), a $\delta-$ sized neighborhood of the true density f_0 is given by

$$K_\delta = \{f : D_K(f_0, f) \leq \delta\} \tag{2.12}$$

Further towards the proof of Theorem 2.4.3, we define the sieve $\mathcal{F}_n$ as the set of all neural networks with each parameter less than C_n in absolute value,

$$|\gamma_{jh}| \leq C_n, \;\; |\beta_j| \leq C_n, \quad j = 0, \ldots, k_n, \;\; h = 0, \ldots, p \tag{2.13}$$

where C_n grows with n such that $C_n \leq \exp(n^{b-a})$ for any constant b where $0 < a < b < 1$, and a is same as in Theorem 2.4.3.

For the above choice of sieve, we next provide a set of conditions on the prior π_n which guarantee the posterior consistency of f_0 over the Hellinger neighborhoods. At the end of

this section, we demonstrate that the following theorem and corollary serve as an important tool towards the proof of Theorem 2.4.3.

Theorem 2.4.4. *Suppose a prior π_n satisfies*

i $\exists\, r > 0$ *and* N_1 *s.t.* $\pi_n(\mathcal{F}_n^c) < exp(-nr),\ \forall n \geq N_1$

ii $\forall \delta, \nu\ > 0, \exists\, N_2$ *s.t.* $\pi_n(K_\delta) \geq exp(-n\nu),\ \forall n \geq N_2.$

Then $\forall \epsilon > 0$,

$$P(A_\epsilon|(\boldsymbol{X_1}, Y_1), \ldots, (\boldsymbol{X_n}, Y_n)) \xrightarrow{p} 1$$

where A_ϵ is the Hellinger neighborhood of f_0 as in (2.9).

Corollary 2.4.5. *Under the conditions of Theorem 2.4.4, $\hat{\mu}_n$ is asymptotically consistent for μ_0, i.e.*

$$\int |\hat{\mu}_n(x) - \mu_0(x)|\ d\boldsymbol{x} \xrightarrow{p} 0$$

We present the proofs of Theorem 2.4.4 and Corollary 2.4.5 in

The main idea behind the proof of Theorem 2.4.4 is to consider the complement of $P(A_\epsilon|(\boldsymbol{X_1}, Y_1), .., (\boldsymbol{X_n}, Y_n))$ as a ratio of integrals. Hence let

$$R_n(f) = \frac{\prod_{i=1}^{n} f(\boldsymbol{x_i}, y_i)}{\prod_{i=1}^{n} f_0(\boldsymbol{x_i}, y_i)} \tag{2.14}$$

Then

$$\begin{aligned} P(A_\epsilon^c|(\boldsymbol{X_1}, Y_1), \ldots, (\boldsymbol{X_n}, Y_n)) &= \frac{\int_{A_\epsilon^c} \prod_{i=1}^{n} f(\boldsymbol{x_i}, y_i) \mathrm{d}\pi_n(f)}{\int \prod_{i=1}^{n} f(\boldsymbol{x_i}, y_i) \mathrm{d}\pi_n(f)} = \frac{\int_{A_\epsilon^c} R_n(f) \mathrm{d}\pi_n(f)}{\int R_n(f) \mathrm{d}\pi_n(f)} \\ &= \frac{\int_{A_\epsilon^c \cap \mathcal{F}_n} R_n(f) \mathrm{d}\pi_n(f) + \int_{A_\epsilon^c \cap \mathcal{F}_n^c} R_n(f) \mathrm{d}\pi_n(f)}{\int R_n(f) \mathrm{d}\pi_n(f)} \end{aligned}$$

In the proof, we have shown that the numerator is small as compared to the denominator, thereby ensuring $P(A_\epsilon^c|(\boldsymbol{X_1}, Y_1), \ldots, (\boldsymbol{X_n}, Y_n)) \xrightarrow{p} 0$. The convergence of the second term in the numerator uses assumption i) of the Theorem 2.4.4. It systematically shows that $\int_{\mathcal{F}_n^c} R_n(f)\mathrm{d}\pi_n(f) < \exp(-nr/2)$ except on a set with probability tending to zero (see Lemma A.1.3 in A for further details). The denominator is bounded using assumption ii) of Theorem 2.4.4. First the KL distance between f_0 and $\int f d\pi_n(f)$ is bounded and subsequently used to prove that $P(R_n(f) \leq e^{-n\varsigma}) \xrightarrow{p} 0$, where ς depends on δ defined earlier. This leads to a conclusion that for all $\varsigma > 0$ and for sufficiently large n, $\int R_n(f) d\pi_n(f) > e^{-n\varsigma}$ except on a set of probability going to zero. The result in this case has been condensed in Lemma A.1.4 presented in A.

Lastly, the first term in the numerator is bounded using the Hellinger bracketing entropy defined below

Definition 2.4.6 (Bracketing Entropy)**.** *For any two functions l and u, define the bracket $[l, u]$ as the set of all functions f such that $l \leq f \leq u$. Let $\|.\|$ be a metric. Define an ϵ-bracket as a bracket with $\|u - l\| < \epsilon$. Define the bracketing number of a set of functions $\mathcal{F}^*$ as the minimum number of ϵ-brackets needed to cover the $\mathcal{F}^*$, and denote it by $N_{[]}(\epsilon, \mathcal{F}^*, \|.\|)$. Finally, the bracketing entropy, denoted by $H_{[]}()$, is the natural logarithm of the bracketing number. (Pollard, 1991)*

Wong and Shen (1995, Theorem 1, pp.348-349) gives the conditions on the rate of growth of the Hellinger bracketing entropy in order to ensure $\int_{A_\epsilon^c \cap \mathcal{F}_n} R_n(f)\mathrm{d}\pi_n(f) \xrightarrow{p} 0$. We next outline the steps to bound the bracketing entropy induced by the sieve structure in (2.13).

In this direction, we first compute the covering number and use it as an upper bound in order to find the bracketing entropy for a neural network. Let's consider k, number of hidden nodes to be fixed for now and restrict the parameter space to $\mathcal{F}_n$ then $\mathcal{F}_n \subset \mathbb{R}^d$ where $d = (p + 2)k + 1$. Further let the covering number be $N(\epsilon, \mathcal{F}_n, \|.\|)$ and use L_∞ as a metric to cover the $\mathcal{F}_n$ with balls of radius ϵ. Then, one does not require more than $((C_n + 1)/\epsilon)^d$

such balls which implies

$$N(\epsilon, \mathcal{F}_n, L_\infty) \leq \left(\frac{2C_n}{2\epsilon} + 1\right)^d = \left(\frac{C_n + \epsilon}{\epsilon}\right)^d \leq \left(\frac{C_n + 1}{\epsilon}\right)^d \tag{2.15}$$

Together with (2.15), we use results from to bound the bracketing number of $\mathcal{F}^*$ (the space of all functions on x and y with parameter vectors lying in $\mathcal{F}_n$) as follows:

$$N_{[]}(\epsilon, \mathcal{F}^*, \|.\|_2) \leq \left(\frac{dC_n^2}{\epsilon}\right)^d \tag{2.16}$$

This allows us to determine the rate of growth of Hellinger bracketing entropy which is nothing but the log of the quantity in (2.16). For further details, we refer to Lemmas A.1.2 and A.1.3 in A.

Going back to the proof of Theorem 2.4.3, we show that the π_n in Theorem 2.4.3 satisfies the conditions of Theorem 2.4.4 for $\mathcal{F}_n$ as in (2.13). Then, the result of Theorem 2.4.3 follows from the Corollary 2.4.5 which is derived from Theorem 2.4.4. Further details of the proof of Theorem 2.4.3 are presented in B. Although Theorem 2.4.3 uses a fixed prior, the results can be extended to a more general class of prior distributions as long as the assumptions of Theorem 2.4.4 hold.

2.5 Numerical Experiments

2.5.1 Simulation Studies

We investigate the performance of the proposed BQRNN method using two simulated examples and compare the estimated conditional quantiles of the response variable against frequentist quantile regression (QR), Bayesian quantile regression (BQR), and quantile regression neural network (QRNN) models. We implement QR from `quantreg` package, BQR from `bayesQR` package and QRNN from `qrnn` (Cannon, 2011) package available in R. We choose two simulation scenarios, (i) a linear additive model, (ii) a nonlinear polynomial model. In both the cases we consider a heteroscedastic behavior of y given $\boldsymbol{x}$.

Scenario I: Linear heteroscedastic; Data are generated from

$$Y = \boldsymbol{X}^T\beta_1 + \boldsymbol{X}^T\beta_2\varepsilon,$$

Scenario II: Non-linear heteroscedastic; Data are generated from

$$Y = (\boldsymbol{X}^T\beta_1)^4 + (\boldsymbol{X}^T\beta_2)^2\varepsilon,$$

where, $\boldsymbol{X} = (X_1, X_2, X_3)$ and X_i's are independent and follow $U(0, 5)$. The parameters β_1 and β_2 are set at $(2, 4, 6)$ and $(0.1, 0.3, 0.5)$ respectively.

We chose these scenarios to make our simulation studies comparable across the past literature in quantile regression neural network domain (Xu et al., 2017; Cannon, 2018). The robustness of our method is illustrated using three different types of random error component (ε): $N(0, 1)$, $U(0, 1)$, and $\mathcal{E}(1)$ where, $\mathcal{E}(\zeta)$ is the exponential distribution with mean ζ^{-1}. Fore each scenario, we generate 200 independent observations.

We work with a single layer feedforward neural network with a fixed number of nodes k. We have tried several values of k in the range of 2-8 and settled on $k = 4$ which yielded better results than other choices while bearing reasonable computational cost. We generated 100000 MCMC samples and then discarded first half of the sampled chain as burn-in period. The 50% burn-in samples in MCMC simulations is not quite unusual and has been suggested by Gelman and Rubin (1992). We also choose every 10^{th} sampled value for the estimated parameters to diminish the effect of autocorrelation between consecutive draws. Convergence of MCMC was checked using standard MCMC diagnostic tools (Gelman et al., 2013).

We have tried several different values of the hyperparameters. For brevity, we report the results for only choice of hyperparameters given by $\boldsymbol{\beta_0} = \mathbf{0}$, $\sigma_0^2 = 100$, $\boldsymbol{\gamma_j} = \mathbf{0}$, $\sigma_1^2 = 100$, $a = 3$, and $b = 0.1$. This particular choice of hyperparameters reflect our preference for near-diffuse priors since in many of the real applications of neural network we don't have information about the input and output variables relationship. Therefore, we wanted to test our model performance in the absence of specific prior elicitation. We also tried different starting values for $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ chains and found that model output is robust to different starting values of $\boldsymbol{\beta}$ but it varies noticeably for different starting values of $\boldsymbol{\gamma}$. Further, we observed that our model yields optimal results when we use QRNN estimates of $\boldsymbol{\gamma}$ as its starting value in our model. We also have to fine-tune the step size of random walk Metropolis-Hastings

(MH) updates in the $\boldsymbol{\gamma}$ generation process and settled on random walk variance of 0.01^2 for scenario I while 0.001^2 for scenario II. These step sizes lead to reasonable rejection rates for MH sampling of $\boldsymbol{\gamma}$ values. However, they indicate the slow traversal of the parameter space for $\boldsymbol{\gamma}$ values.

To compare the model performance of QR, BQR, QRNN and BQRNN, we have calculated the theoretical conditional quantiles (Cond Q) and contrasted them with the estimated conditional quantiles from the given simulated models. Additionally, we compute standard deviation (SD) and root mean squared error (RMSE) values for predicted conditional quantiles of each observation in the BQR and BQRNN model using sampled chains. For scenarios 1 and 2, Table 2.1 and Table 2.2, respectively, present these results at quantile levels, $\tau = (0.05, 0.50, 0.95)$ for 3 observations. The Table 2.1 indicates neural network models performs comparably with the linear models. This ensures the use of neural network models even if the underlying relationship is linear. In Table 2.2, we can observe that BQRNN outperforms other models in the tail area, i.e. $\tau = 0.05, 0.95$, whereas it's performance is comparable to QRNN at the median. Furthermore, We notice that our relatively complex BQRNN model has lower bias but higher variance (SD^2) as compared to BQR model. However, BQRNN outperforms BQR proved by the lower RMSE values which consists of squared bias and variance terms. We notice one exception when RMSE value in BQR is lower than BQRNN for 20^{th} observation in 95^{th} quantile of all 3 error models. The occurrence of this exception is random and not because BQR performed systematically better than our model. In summary, we observe the tradeoff between bias and variance in our model which overall performs better than a linear BQR model in a nonlinear setup. The natural advantage of our Bayesian procedure over the frequentist QRNN is that we have posterior variance for our conditional quantile estimates which can be used as an uncertainty quantification.

Table 2.1 **Simulation study I results.** Simulated Conditional Quantiles for QR, BQR, QRNN and BQRNN

Noise	Quantile	Obs	Theo	QR	BQR			QRNN	BQRNN		
ε	τ	No	Cond Q	Cond Q	Cond Q	SD	RMSE	Cond Q	Cond Q	SD	RMSE
$N(0,1)$	0.05	20	17.56	17.19	17.19	0.60	0.70	16.98	**17.15**	**0.34**	**0.53**
		50	37.38	37.69	37.67	0.74	0.80	38.80	**39.03**	**0.68**	**1.79**
		100	42.53	42.45	42.56	0.92	0.92	41.43	**41.31**	**0.69**	**1.40**
	0.50	20	20.23	20.23	20.25	0.37	0.37	20.23	**20.23**	**0.51**	**0.51**
		50	42.62	42.87	42.65	0.31	0.31	42.62	**42.68**	**0.77**	**0.77**
		100	48.78	48.81	48.67	0.39	0.41	48.78	**48.01**	**0.97**	**1.24**
	0.95	20	22.90	21.81	22.38	0.68	0.86	21.42	**21.85**	**0.46**	**1.15**
		50	47.86	47.92	47.70	0.69	0.71	47.52	**47.19**	**0.80**	**1.04**
		100	55.02	54.28	54.20	0.92	1.24	53.80	**53.47**	**1.05**	**1.88**
$U(0,1)$	0.05	20	20.31	20.39	20.25	0.40	0.41	20.55	**20.44**	**0.30**	**0.33**
		50	42.78	42.68	42.56	0.47	0.52	42.89	**42.92**	**0.61**	**0.63**
		100	48.97	48.93	48.82	0.56	0.58	48.34	**48.82**	**0.70**	**0.72**
	0.50	20	21.04	21.28	21.25	0.19	0.29	20.98	**20.97**	**0.33**	**0.34**
		50	44.21	44.30	44.22	0.22	0.22	43.95	**43.98**	**0.66**	**0.70**
		100	50.68	50.88	50.79	0.25	0.28	50.39	**50.82**	**0.76**	**0.78**
	0.95	20	21.77	21.87	22.10	0.46	0.56	21.72	**21.82**	**0.33**	**0.34**
		50	45.65	45.58	45.69	0.46	0.47	45.43	**45.38**	**0.65**	**0.70**
		100	52.39	52.36	52.45	0.58	0.58	52.49	**52.52**	**0.75**	**0.76**
$\mathcal{E}(1)$	0.05	20	20.31	20.24	19.94	0.45	0.58	19.94	**20.27**	**0.31**	**0.31**
		50	42.78	42.92	42.93	0.51	0.53	42.98	**43.17**	**0.63**	**0.74**
		100	48.97	49.06	49.05	0.61	0.62	47.24	**49.60**	**0.75**	**0.98**
	0.50	20	21.36	20.95	21.04	0.23	0.40	21.11	**20.91**	**0.39**	**0.59**
		50	44.83	45.54	45.47	0.27	0.70	46.30	**45.66**	**0.74**	**1.11**
		100	51.41	51.93	51.91	0.41	0.65	51.68	**51.94**	**0.89**	**1.04**
	0.95	20	25.09	25.08	25.17	0.80	0.81	23.73	**24.27**	**0.79**	**1.14**
		50	52.17	53.15	52.61	0.90	1.00	54.98	**54.43**	**1.24**	**2.58**
		100	60.15	61.24	60.75	1.06	1.22	59.87	**62.96**	**1.46**	**3.17**

Obs No: Observation Number; Theo: Theoretical; Cond Q: Conditional Quantile; SD: Standard Deviation; RMSE: Root Mean Squared Error.

Table 2.2 **Simulation study II results.** Simulated Conditional Quantiles for QR, BQR, QRNN and BQRNN

Noise	Quantile	Obs	Theo	QR	BQR			QRNN	BQRNN		
ε	τ	No	Cond Q	Cond Q	Cond Q	SD	RMSE	Cond Q	Cond Q	SD	RMSE
$N(0,1)$	0.05	20	167491.82	-195687.34	10207.80	33.03	157284.03	30400.87	**166112.85**	**8460.05**	**8570.87**
		50	3298781.46	2107072.28	24948.89	73.02	3273832.57	2626269.69	**3289014.06**	**48028.83**	**49007.23**
		100	5660909.58	2741102.73	25680.38	75.52	5635229.20	3312208.28	**5311964.68**	**88478.24**	**359985.24**
	0.50	20	167496.16	83421.40	92937.30	101.97	74558.93	183185.47	**169748.75**	**2615.13**	**3451.33**
		50	3298798.17	2661086.40	228321.19	282.04	3070476.99	3292073.45	**3290858.99**	**46577.95**	**47245.13**
		100	5660933.30	3550782.83	233862.83	258.14	5427070.47	5652427.31	**5662655.46**	**80356.32**	**80366.74**
	0.95	20	167500.49	1184601.83	171358.16	180.90	3861.91	193187.39	**174073.37**	**2751.37**	**7125.39**
		50	3298814.88	4731674.95	423324.96	481.51	2875489.96	3309795.50	**3298328.52**	**46685.76**	**46683.63**
		100	5660957.02	5575413.13	429129.54	458.70	5231827.50	5772437.35	**5676164.49**	**80825.89**	**82236.16**
$U(0,1)$	0.05	20	167496.29	-195833.01	10207.87	33.11	157288.42	-27470.71	**154598.77**	**7923.98**	**15136.80**
		50	3298798.68	2107278.20	24949.04	73.11	3273849.64	2500915.26	**3295376.39**	**47739.91**	**47857.66**
		100	5660934.02	2741400.69	25680.55	75.79	5635253.47	3100036.18	**5285593.84**	**98238.89**	**387980.93**
	0.50	20	167497.47	83435.89	92937.55	101.16	74559.99	172793.77	**168331.89**	**2583.07**	**2714.25**
		50	3298803.25	2661086.35	228322.57	278.93	3070480.69	3314585.80	**3296332.61**	**46650.01**	**46710.73**
		100	5660940.51	3550796.98	233863.94	256.32	5427076.57	5607084.86	**5660871.21**	**80101.17**	**80093.19**
	0.95	20	167498.66	1184587.54	171358.60	163.87	3863.42	196767.97	**174714.72**	**5874.89**	**9304.78**
		50	3298807.82	4731690.36	423325.69	464.88	2875482.17	3316486.10	**3303092.85**	**47168.25**	**47357.80**
		100	5660947.00	5575416.01	429130.38	430.23	5231816.64	5757769.91	**5661073.43**	**80290.13**	**80282.20**
$\mathcal{E}(1)$	0.05	20	167496.29	-195784.51	10207.99	32.90	157288.30	-172912.92	**161521.55**	**5217.48**	**7931.84**
		50	3298798.69	2107220.10	24949.33	72.48	3273849.36	2380587.55	**3295994.41**	**47385.22**	**47463.40**
		100	5660934.04	2741316.61	25680.81	75.25	5635253.23	2825010.54	**5194410.73**	**82821.54**	**473816.46**
	0.50	20	167497.98	83442.50	92937.24	102.27	74560.81	175113.10	**169606.69**	**2568.74**	**3323.21**
		50	3298805.21	2661099.63	228321.18	282.92	3070484.04	3309978.81	**3297248.31**	**46648.10**	**46669.41**
		100	5660943.29	3550827.77	233862.58	258.35	5427080.71	5603733.56	**5660975.15**	**80088.73**	**80080.73**
	0.95	20	167504.05	1184595.04	171358.64	163.27	3858.05	182663.10	**172254.32**	**2917.90**	**5574.72**
		50	3298828.60	4731683.78	423325.49	458.35	2875503.15	3337299.83	**3298567.13**	**46680.30**	**46676.36**
		100	5660976.50	5575416.49	429130.11	422.79	5231846.41	5838226.18	**5687421.49**	**80742.36**	**84955.07**

Obs No: Observation Number; Theo: Theoretical; Cond Q: Conditional Quantile; SD: Standard Deviation; RMSE: Root Mean Squared Error.

2.5.2 Real Data Examples

In this section, we apply our proposed method to three real world datasets which are publicly available.

The first dataset is the Boston Housing dataset which is available in R package MASS (Venables and Ripley, 2002). It contains 506 census tracts of Boston Standard Metropolitan Statistical Area in 1970. There are 13 predictor variables and one response variable, corrected median value of owner-occupied homes (in USD 1000s). Predictor variables include per capita crime rate by town, proportion of residential land zoned for lots over 25,000 sq.ft., nitrogen oxide concentration, proportion of owner-occupied units built prior to 1940, full-value property-tax rate per $10,000, and lower status of the population in percent, among others. There is high correlation among some of these predictor variables and the goal here is to determine the best fitting functional form to improve the housing value forecasts.

The second dataset is the Gilgais dataset available in R package MASS. This data was collected on a line transect survey in gilgai territory in New South Wales, Australia. Gilgais are repeated mounds and depressions formed on flat land, and many-a-times are regularly distributed. The data collection with 365 sampling locations on a linear grid of 4 meters spacing aims to check if the gilgai patterns are reflected in the soil properties as well. At each of the sampling location, samples were taken at depths 0-10 cm, 30-40 cm and 80-90 cm below the surface. The input variables included pH, electrical conductivity and chloride content and were measured on a 1:5 soil:water extract from each sample. Here, the response variable is e80 (electrical conductivity in mS/cm: 80–90 cm) and we focus on finding the true functional relationship present in the dataset.

The third dataset is concrete data which is compiled by Yeh (1998) and is available on UCI machine learning repository. It consists of 1030 records, each containing 8 input features and compressive strength of concrete as an output variable. The input features include the amounts of ingredients in high performance concrete (HPC) mixture which are cement, fly ash, blast furnace slag, water, superplasticizer, coarse aggregate, and fine aggregate. More-

over, age of the mixture in days is also included as one of the predictor variable. According to Yeh (1998), the compressive strength of concrete is a highly non-linear function of the given inputs. The central purpose of the study is to predict the compressive strength of HPC using the input variables.

In our experiments, we compare the performance of QR, BQR, QRNN, and BQRNN estimates for $f(\boldsymbol{x})$, the true functional form of the data, in both training and testing data using mean check function (or, mean tilted absolute loss function). The mean check function (MCF) is given as

$$\text{MCF} = \frac{1}{N}\sum_{i=1}^{N}\rho_\tau(y_i - \hat{f}(\boldsymbol{x}_i))$$

where, $\rho_\tau(.)$ is defined in (2.3) and $\hat{f}(\boldsymbol{x})$ is an estimate of $f(\boldsymbol{x})$. We resort to this comparison criterion since we don't have the theoretical conditional quantiles for the data at our disposal. For each dataset, we randomly choose 80% of data points for training the model and then remaining 20% is used to test the prediction ability of the fitted model. Our single hidden-layer neural network has $k = 4$ hidden layer nodes and the random walk variance is chosen to be 0.01^2. These particular choices of the number of hidden layer nodes and random walk step size are based on their optimal performance among several different choices while providing reasonable computational complexity. We perform these analyses for quantiles, $\tau = (0.05, 0.25, 0.50, 0.75, 0.95)$, and present the model comparison results for both training and testing data in Table 2.3.

It can be seen that our model performs comparably well with QRNN model while outperforming linear models (QR and BQR) in all the datasets. We can see that both QRNN and BQRNN have lower mean check function values for training data than their testing counterpart. This suggests that neural networks may be overfitting the data while trying to find the true underlying functional form. The model performance of QR and BQR models is inferior compared to neural network models, particularly when the regression relationship is non-linear. Furthermore, our BQRNN model provides uncertainty estimation as a natural byproduct which is not available in the frequentist QRNN model.

Table 2.3 **Real data applications results.** MCF values are reported

Noise	Quantile	Sample	QR	BQR	QRNN	BQRNN
Boston	$\tau = 0.05$	Train	0.3009	0.3102	0.2084	**0.1832**
		Test	0.3733	0.3428	0.3356	**0.5842**
	$\tau = 0.25$	Train	1.0403	1.0431	0.6340	**0.6521**
		Test	1.2639	1.2431	1.0205	**1.2780**
	$\tau = 0.50$	Train	1.4682	1.4711	0.8444	**0.8864**
		Test	1.8804	1.8680	1.4638	**1.5882**
	$\tau = 0.75$	Train	1.3856	1.3919	0.6814	**0.7562**
		Test	1.8426	1.8053	1.3773	**1.4452**
	$\tau = 0.95$	Train	0.5758	0.6009	0.2276	**0.2206**
		Test	0.7882	0.6174	0.8093	**0.6880**
Gilgais	$\tau = 0.05$	Train	3.6610	3.7156	3.0613	**2.7001**
		Test	3.4976	3.2105	2.9137	**3.9163**
	$\tau = 0.25$	Train	13.9794	14.5734	8.6406	**8.4565**
		Test	11.8406	11.4832	9.3298	**10.1386**
	$\tau = 0.50$	Train	18.1627	21.3587	10.4667	**10.7845**
		Test	15.8210	17.2037	13.5297	**14.3699**
	$\tau = 0.75$	Train	13.6598	18.8357	7.9679	**7.9905**
		Test	12.3926	18.2477	9.4711	**10.6414**
	$\tau = 0.95$	Train	3.8703	6.4137	2.3289	**2.2508**
		Test	4.1266	6.4300	3.0280	**2.5586**
Concrete	$\tau = 0.05$	Train	2.9130	4.4500	2.0874	**2.0514**
		Test	2.9891	4.2076	2.2021	**2.6793**
	$\tau = 0.25$	Train	10.0127	14.7174	7.0063	**7.0537**
		Test	9.6451	14.3567	6.9179	**7.4069**
	$\tau = 0.50$	Train	13.1031	19.8559	9.3638	**9.3728**
		Test	12.7387	18.2309	9.8936	**10.9172**
	$\tau = 0.75$	Train	11.5179	17.7680	7.6789	**7.3932**
		Test	10.8299	16.3257	8.5755	**9.4147**
	$\tau = 0.95$	Train	3.9493	6.8747	2.4403	**2.5262**
		Test	3.6489	6.8435	2.7768	**4.1369**

2.6 Conclusion and Discussion

This chapter introduces the Bayesian neural network model for quantile estimation in a systematic way. The practical implementation of Gibbs sampling coupled with Metropolis-Hastings updates method have been discussed in detail. The method exploits the location-scale mixture representation of the asymmetric Laplace distribution which makes its implementation easier. The model can be thought as a hierarchical Bayesian model which makes use of independent normal priors for the neural network weight parameters. A future work in this area could be sparsity induced priors to allow for node and layer selection in multi-layer neural network architecture.

Further, we have developed asymptotic consistency of the posterior distribution of the neural network parameters. The presented result can be extended to a more general class of prior distributions if they satisfy the Theorem 2.4.4 assumptions. Following the theory developed here, we bridge the gap between asymptotic justifications separately available for Bayesian quantile regression and Bayesian neural network regression. The theoretical arguments developed here justify using neural networks for quantile estimation in nonparametric regression problems using Bayesian methods.

The proposed MCMC procedure has been shown to work when the number of parameters are relatively low compared to the number of observations. We noticed that the convergence of the posterior chains take long time and there is noticeable autocorrelation left in the sampled chains even after burn-in period. We also acknowledge that our random-walk Metropolis-Hastings algorithm has small step size which might lead to slow traversal of the parameter space ultimately raising the computational cost of our algorithm. The computational complexity in machine learning methods are well-known. Further research is required in these aspects of model implementation.

APPENDICES

APPENDIX A

LEMMAS FOR POSTERIOR CONSISTENCY PROOF

For all the proofs in Appendix A and Appendix B, we assume $\boldsymbol{X}_{p\times 1}$ to be uniformly distributed on $[0,1]^p$ and keep them fixed. Thus, $f_0(\boldsymbol{x}) = f(\boldsymbol{x}) = 1$. Conditional on $\boldsymbol{X}$, the univariate response variable Y has asymmetric Laplace distribution with location parameter determined by the neural network. We are going to fix its scale parameter, σ, to be 1 for the posterior consistency derivations. Thus,

$$Y|\boldsymbol{X} = \boldsymbol{x} \sim ALD\left(\beta_0 + \sum_{j=1}^{k} \beta_j \frac{1}{1+\exp(-\gamma_{j0} - \sum_{h=1}^{p} \gamma_{jh} x_h)}, 1, \tau\right) \tag{A.1}$$

The number of input variables, p, is taken to be fixed while the number of hidden nodes, k, is allowed to grow with the sample size, n.

A.1 Requisite Lemmas

All the lemmas described below are taken from Lee (2000).

Lemma A.1.1. *Suppose* $H_{[]}(u) \leq \log[(C_n^2 d_n/u)^{d_n}], d_n = (p+2)k_n + 1, k_n \leq n^a$ *and* $C_n \leq exp(n^{b-a})$ *for* $0 < a < b < 1$. *Then for any fixed constants* $c, \epsilon > 0$, *and for all sufficiently large* n, $\int_0^\epsilon \sqrt{H_{[]}(u)} \leq c\sqrt{n}\epsilon^2$.

Proof. The proof follows from the proof of Lemma 1 from (Lee, 2000, p. 634-635). □

For Lemmas A.1.2, A.1.3 and A.1.4, we make use of the following notations. From (2.14), recall

$$R_n(f) = \prod_{i=1}^{n} \frac{f(x_i, y_i)}{f_0(x_i, y_i)}$$

is the ratio of likelihoods under neural network density f and the true density f_0. $\mathcal{F}_n$ is the sieve as defined in (2.13) and A_ϵ is the Hellinger neighborhood of the true density f_0 as in (2.9).

Lemma A.1.2. $\sup_{f \in A_\epsilon^c \cap \mathcal{F}_n} R_n(f) \leq 4exp(-c_2 n\epsilon^2)$ *a.s. for sufficiently large* n.

Proof. Using the outline of the proof of Lemma 2 from (Lee, 2000, p. 635), first we have to bound the Hellinger bracketing entropy using Van Der Vaart and Wellner (1996, Theorem 2.7.11 on p.164). Next we use Lemma A.1.1 to show that the conditions of Wong and Shen (1995, Theorem 1 on p.348-349) hold and finally we apply that theorem to get the result presented in the Lemma 2.

In our case of BQRNN, we only need to derive first step using ALD density mentioned in (A.1). And rest of the steps follow from the proof given in Lee (2000). As we are looking for the Hellinger bracketing entropy for neural networks, we use L_2 norm on the square root of the density functions, f. The L_∞ covering number was computed above in (2.15), so here $d^* = L_\infty$. The version of Van Der Vaart and Wellner (1996, Theorem 2.7.11) that we are interested in is

$$\begin{aligned} &\text{If } \left|\sqrt{f_t(x,y)} - \sqrt{f_s(x,y)}\right| \leq d^*(s,t)F(x,y) \quad \text{for some } F, \\ &\text{then, } N_{[]}(2\epsilon \left\|F\right\|_2, \mathcal{F}^*, \left\|.\right\|_2) \leq N(\epsilon, \mathcal{F}_n, d^*) \end{aligned}$$

Now let's start by defining some notations,

$$\begin{aligned} f_t(x,y) &= \tau(1-\tau)\text{exp}\left(-(y-\mu_t(x))(\tau - I_{(y\leq\mu_t(x))})\right), \\ &\text{where, } \mu_t(x) = \beta_0^t + \sum_{j=1}^{k} \frac{\beta_j^t}{1+\exp(-A_j(x))} \text{ and } A_j(x) = \gamma_{j0}^t + \sum_{h=1}^{p} \gamma_{jh}^t x_h \end{aligned} \tag{A.2}$$

$$\begin{aligned} f_s(x,y) &= \tau(1-\tau)\text{exp}\left(-(y-\mu_s(x))(\tau - I_{(y\leq\mu_s(x))})\right), \\ &\text{where, } \mu_s(x) = \beta_0^s + \sum_{j=1}^{k} \frac{\beta_j^s}{1+\exp(-B_j(x))} \text{ and } B_j(x) = \gamma_{j0}^s + \sum_{h=1}^{p} \gamma_{jh}^s x_h \end{aligned} \tag{A.3}$$

For notational convenience, we drop x and y from $f_s(x,y)$, $f_t(x,y)$, $\mu_s(x)$, $\mu_t(x)$, $B_j(x)$, and $A_j(x)$ and denote them as f_s, f_t, μ_s, μ_t, B_j, and A_j.

$$\begin{aligned} &\left|\sqrt{f_t} - \sqrt{f_s}\right| \\ &= \sqrt{\tau(1-\tau)}\left|\exp\left(-\frac{1}{2}(y-\mu_t)(\tau - I_{(y\leq\mu_t)})\right) - \exp\left(-\frac{1}{2}(y-\mu_s)(\tau - I_{(y\leq\mu_s)})\right)\right| \end{aligned}$$

As, $\tau \in (0,1)$ is fixed.

$$\leq \frac{1}{2}\left|\exp\left(-\frac{1}{2}(y-\mu_t)(\tau - I_{(y\leq\mu_t)})\right) - \exp\left(-\frac{1}{2}(y-\mu_s)(\tau - I_{(y\leq\mu_s)})\right)\right| \tag{A.4}$$

Now let's separate above term into two cases when: (a) $\mu_s \leq \mu_t$ and (b) $\mu_s > \mu_t$. Further let's consider case-a and break it into three subcases when: (i) $y \leq \mu_s \leq \mu_t$, (ii) $\mu_s < y \leq \mu_t$, and (iii) $\mu_s \leq \mu_t < y$.

Case-a (i) $y \leq \mu_s \leq \mu_t$

The (A.4) simplifies to

$$\frac{1}{2}\left|\exp\left(-\frac{1}{2}(y-\mu_t)(\tau-1)\right) - \exp\left(-\frac{1}{2}(y-\mu_s)(\tau-1)\right)\right|$$
$$= \frac{1}{2}\left|\exp\left(-\frac{1}{2}(y-\mu_s)(\tau-1)\right)\right|\left|\exp\left(-\frac{1}{2}(\mu_s-\mu_t)(\tau-1)\right) - 1\right|$$

As first term in modulus is ≤ 1

$$\leq \frac{1}{2}\left|1 - \exp\left(-\frac{1}{2}(\mu_t-\mu_s)(1-\tau)\right)\right|$$

Note: $1-\exp(-z) \leq z \ \ \forall z \in \mathbb{R} \implies |1-\exp(-z)| \leq |z| \ \ \forall z \geq 0$ (A.5)

$$\leq \frac{1}{4}|\mu_t - \mu_s|(1-\tau)$$
$$\leq \frac{1}{4}|\mu_t - \mu_s|$$
$$\leq \frac{1}{2}|\mu_t - \mu_s|$$

Case-a (ii) $\mu_s < y \leq \mu_t$

The (A.4) simplifies to

$$\frac{1}{2}\left|\exp\left(-\frac{1}{2}(y-\mu_t)(\tau-1)\right) - \exp\left(-\frac{1}{2}(y-\mu_s)\tau\right)\right|$$
$$= \frac{1}{2}\left|\exp\left(-\frac{1}{2}(y-\mu_s)(\tau-1)\right) - 1 + 1 - \exp\left(-\frac{1}{2}(y-\mu_s)\tau\right)\right|$$
$$\leq \frac{1}{2}\left|1 - \exp\left(-\frac{1}{2}(y-\mu_t)(\tau-1)\right)\right| + \frac{1}{2}\left|1 - \exp\left(-\frac{1}{2}(y-\mu_s)\tau\right)\right|$$

Let's use calculus inequality mentioned in (A.5)

$$\leq \frac{1}{4}|(y-\mu_t)(\tau-1)| + \frac{1}{4}|(y-\mu_s)\tau|$$

Both terms are positive so we combine them in one modulus

$$= \frac{1}{4} \left| (y - \mu_t)(\tau - 1) + (y - \mu_t + \mu_t - \mu_s)\tau \right|$$

$$= \frac{1}{4} \left| (y - \mu_t)(2\tau - 1) + (\mu_t - \mu_s)\tau \right|$$

$$\leq \frac{1}{4} \left[\left| (y - \mu_t) \right| \left| 2\tau - 1 \right| + \left| \mu_t - \mu_s \right| \tau \right]$$

Here, $|y - \mu_t| \leq |\mu_t - \mu_s|$ and $|2\tau - 1| \leq 1$

$$\leq \frac{1}{2} \left| \mu_t - \mu_s \right|$$

Case-a (iii) $\mu_s \leq \mu_t < y$

The (A.4) simplifies to

$$\frac{1}{2} \left| \exp\left(-\frac{1}{2}(y - \mu_t)\tau \right) - \exp\left(-\frac{1}{2}(y - \mu_s)\tau \right) \right|$$

$$= \frac{1}{2} \left| \exp\left(-\frac{1}{2}(y - \mu_t)\tau \right) \right| \left| 1 - \exp\left(-\frac{1}{2}(\mu_t - \mu_s)\tau \right) \right|$$

As first term in modulus is ≤ 1

$$\leq \frac{1}{2} \left| 1 - \exp\left(-\frac{1}{2}(\mu_t - \mu_s)\tau \right) \right|$$

Using the calculus inequality mentioned in (A.5)

$$\leq \frac{1}{4} \left| \mu_t - \mu_s \right| \tau$$

$$\leq \frac{1}{4} \left| \mu_t - \mu_s \right|$$

$$\leq \frac{1}{2} \left| \mu_t - \mu_s \right|$$

We can similarly bound the (A.4) in case-(b) where $\mu_s > \mu_t$ by $|\mu_t - \mu_s| /2$. Now,

$$\left| \sqrt{f_t} - \sqrt{f_s} \right|$$

$$\leq \frac{1}{2} \left| \mu_t - \mu_s \right|$$

Now, let's substitute μ_t and μ_s from A.2 and A.3

$$= \frac{1}{2} \left| \beta_0^t + \sum_{j=1}^{k} \frac{\beta_j^t}{1 + \exp(-A_j)} - \beta_0^s - \sum_{j=1}^{k} \frac{\beta_j^s}{1 + \exp(-B_j)} \right|$$

$$\leq \frac{1}{2}\left[\left|\beta_0^t-\beta_0^s\right|+\sum_{j=1}^{k}\left|\frac{\beta_j^t}{1+\exp(-A_j)}-\frac{\beta_j^s}{1+\exp(-B_j)}\right|\right]$$

$$= \frac{1}{2}\left[\left|\beta_0^t-\beta_0^s\right|+\sum_{j=1}^{k}\left|\frac{\beta_j^t-\beta_j^s+\beta_j^s}{1+\exp(-A_j)}-\frac{\beta_j^s}{1+\exp(-B_j)}\right|\right]$$

$$= \frac{1}{2}\left[\left|\beta_0^t-\beta_0^s\right|+\sum_{j=1}^{k}\frac{\left|\beta_j^t-\beta_j^s\right|}{1+\exp(-A_j)}+\sum_{j=1}^{k}\left|\beta_j^s\right|\left|\frac{1}{1+\exp(-A_j)}-\frac{1}{1+\exp(-B_j)}\right|\right]$$

Recall that $\left|\beta_j^s\right| \leq C_n$

$$\leq \frac{1}{2}\left[\left|\beta_0^t-\beta_0^s\right|+\sum_{j=1}^{k}\left|\beta_j^t-\beta_j^s\right|+\sum_{j=1}^{k}C_n\left|\frac{\exp(-B_j)-\exp(-A_j)}{(1+\exp(-A_j))(1+\exp(-B_j))}\right|\right] \tag{A.6}$$

Note: $|\exp(-B_j)-\exp(-A_j)| = \begin{cases}\exp(-A_j)(1-\exp(-(B_j-A_j))), & \text{when } B_j-A_j\geq 0\\ exp(-B_j)(1-\exp(-(A_j-B_j))), & \text{when } A_j-B_j\geq 0\end{cases}$

Using the calculus inequality mentioned in (A.5)

$$\leq \begin{cases}\exp(-A_j)(B_j-A_j), & \text{when } B_j-A_j\geq 0\\ \exp(-B_j)(A_j-B_j), & \text{when } A_j-B_j\geq 0\end{cases}$$

So, $\left|\frac{\exp(-B_j)-\exp(-A_j)}{(1+\exp(-A_j))(1+\exp(-B_j))}\right| \leq \begin{cases}\frac{\exp(-A_j)(B_j-A_j)}{(1+\exp(-A_j))(1+\exp(-B_j))}, & \text{when } B_j-A_j\geq 0\\ \frac{\exp(-B_j)(A_j-B_j)}{(1+\exp(-A_j))(1+\exp(-B_j))}, & \text{when } A_j-B_j\geq 0\end{cases}$

$$\leq |A_j-B_j|$$

Hence we can bound the (A.6) as follows

$$\left|\sqrt{f_t}-\sqrt{f_s}\right| \leq \frac{1}{2}\left[\left|\beta_0^t-\beta_0^s\right|+\sum_{j=1}^{k}\left|\beta_j^t-\beta_j^s\right|+\sum_{j=1}^{k}C_n\left|A_j-B_j\right|\right]$$

Now, let's substitute A_j and B_j from A.2 and A.3

$$\leq \frac{1}{2}\left[\left|\beta_0^t-\beta_0^s\right|+\sum_{j=1}^{k}\left|\beta_j^t-\beta_j^s\right|+\sum_{j=1}^{k}C_n\left|\gamma_{j0}^t+\sum_{h=1}^{p}\gamma_{jh}^t x_h-\gamma_{j0}^s-\sum_{h=1}^{p}\gamma_{jh}^s x_h\right|\right]$$

$$\leq \frac{1}{2}\left[\left|\beta_0^t-\beta_0^s\right|+\sum_{j=1}^{k}\left|\beta_j^t-\beta_j^s\right|+\sum_{j=1}^{k}C_n\left(\left|\gamma_{j0}^t-\gamma_{j0}^s\right|+\sum_{h=1}^{p}\left|x_h\right|\left|\gamma_{jh}^t-\gamma_{jh}^s\right|\right)\right]$$

$$
\begin{aligned}
&\text{Recall that } |x_h| \leq 1 \text{ and w.l.o.g assume } C_n > 1 \\
&\leq \frac{C_n}{2}\left[\left|\beta_0^t - \beta_0^s\right| + \sum_{j=1}^{k}\left|\beta_j^t - \beta_j^s\right| + \sum_{j=1}^{k}\left(\left|\gamma_{j0}^t - \gamma_{j0}^s\right| + \sum_{h=1}^{p}\left|\gamma_{jh}^t - \gamma_{jh}^s\right|\right)\right] \\
&\leq \frac{C_n d}{2}\left\|t - s\right\|_\infty
\end{aligned}
$$

Now rest of the steps follow from the proof of Lemma 2 in Lee (2000, p. 635-636). □

Lemma A.1.3. *If there exists a constant $r > 0$ and N, such that $\mathcal{F}_n$ satisfies $\pi_n(\mathcal{F}_n^c) < exp(-nr), \forall n \geq N$, then there exists a constant c_2 such that $\int_{A_\epsilon^c} R_n(f)\, d\pi_n(f) < exp(-nr/2) + exp(-nc_2\epsilon^2)$ except on a set of probability tending to zero.*

Proof. The proof is same as the proof of Lemma 3 from (Lee, 2000, p. 636). □

Lemma A.1.4. *Let K_δ be the KL-neighborhood as in (2.12. Suppose that for all $\delta, \nu > 0, \exists\, N$ s.t. $\pi_n(K_\delta) \geq exp(-n\nu)$, $\forall n \geq N$. Then for all $\varsigma > 0$ and sufficiently large n, $\int R_n(f) d\pi_n(f) > e^{-n\varsigma}$ except on a set of probability going to zero.*

Proof. The proof is same as the proof of Lemma 5 from (Lee, 2000, p. 637). □

Lemma A.1.5. *Suppose that μ is a neural network regression with parameters $(\theta_1, \dots \theta_d)$, and let $\tilde{\mu}$ be another neural network with parameters $(\tilde{\theta}_1, \dots \tilde{\theta}_{\tilde{d}_n})$. Define $\theta_i = 0$ for $i > d$ and $\tilde{\theta}_j = 0$ for $j > \tilde{d}_n$. Suppose that the number of nodes of μ is k, and that the number of nodes of $\tilde{\mu}$ is $\tilde{k}_n = O(n^a)$ for some a, $0 < a < 1$. Let*

$$
M_\varsigma = \{\tilde{\mu} \Big| \left|\theta_i - \tilde{\theta}_i\right| \leq \varsigma, i = 1, 2, \dots\} \tag{A.7}
$$

Then for any $\tilde{\mu} \in M_\varsigma$ and for sufficiently large n,

$$
\sup_{x \in \mathcal{X}} (\tilde{\mu}(x) - \mu(x))^2 \leq (5n^a)^2 \varsigma^2
$$

Proof. The proof is same as the proof of Lemma 6 from (Lee, 2000, p. 638-639). □

APPENDIX B

POSTERIOR CONSISTENCY THEOREM PROOFS

B.1 Theorem 2.4.4 Proof

For the proof of Theorem 2.4.4 and Corollary 2.4.5, we use the following notations. From (2.14), recall that

$$R_n(f) = \prod_{i=1}^{n} \frac{f(\boldsymbol{x}_i, y_i)}{f_0(\boldsymbol{x}_i, y_i)}$$

is the ratio of likelihoods under neural network density f and the true density f_0. Also, $\mathcal{F}_n$ is the sieve as defined in (2.13). Finally, A_ϵ is the Hellinger neighborhood of the true density f_0 as in (2.9).

By Lemma A.1.3, there exists a constant c_2 such that $\int_{A_\epsilon^c} R_n(f)\mathrm{d}\pi_n(f) < \exp(-nr/2) + \exp(-nc_2\epsilon^2)$ for sufficiently large n. Next, from Lemma A.1.4, $\int R_n(f)\mathrm{d}\pi_n(f) \geq \exp(-n\varsigma)$ for sufficiently large n.

$$\begin{aligned} P(A_\epsilon^c|(\boldsymbol{X_1}, Y_1), \ldots, (\boldsymbol{X_n}, Y_n)) &= \frac{\displaystyle\int_{A_\epsilon^c} R_n(f)\mathrm{d}\pi_n(f)}{\displaystyle\int R_n(f)\mathrm{d}\pi_n(f)} \\ &< \frac{\exp\left(-\frac{nr}{2}\right) + \exp(-nc_2\epsilon^2)}{\exp(-n\varsigma)} \\ &= \exp\left(-n\left[\frac{r}{2} - \varsigma\right]\right) + \exp\left(-n\epsilon^2[c_2 - \varsigma]\right) \end{aligned}$$

Now we pick ς such that for $\varphi > 0$, both $\frac{r}{2} - \varsigma > \varphi$ and $c_2 - \varsigma > \varphi$. Thus,

$$P(A_\epsilon^c|(\boldsymbol{X_1}, Y_1), \ldots, (\boldsymbol{X_n}, Y_n)) \leq \exp(-n\varphi) + \exp(-n\epsilon^2\varphi)$$

Hence, $P(A_\epsilon^c|(\boldsymbol{X_1}, Y_1), \ldots, (\boldsymbol{X_n}, Y_n)) \xrightarrow{p} 0$. □

B.2 Corollary 2.4.5 Proof

Theorem 2.4.4 implies that $D_H(f_0, f) \xrightarrow{p} 0$ where $D_H(f_0, f)$ is the Hellinger distance between f_0 and f as in (2.8) and f is a random draw from the posterior. Recall from (2.10),

the predictive density function

$$\hat{f}_n(.) = \int f(.)\, \mathrm{d}P(f|(\boldsymbol{X_1}, Y_1), \ldots, (\boldsymbol{X_n}, Y_n))$$

gives rise to the predictive conditional quantile function, $\hat{\mu}_n(\boldsymbol{x}) = Q_{\tau,\hat{f}_n}(y|\boldsymbol{X} = \boldsymbol{x})$. We next show that $D_H(f_0, \hat{f}_n) \xrightarrow{p} 0$, which in turn implies $\hat{\mu}_n(\boldsymbol{x})$ converges in L_1-norm to the true conditional quantile function,

$$\mu_0(\boldsymbol{x}) = Q_{\tau,f_0}(y|\boldsymbol{X} = \boldsymbol{x}) = \beta_0 + \sum_{j=1}^{k} \beta_j \frac{1}{1 + \exp\left(-\gamma_{j0} - \sum_{h=1}^{p} \gamma_{jh} x_{ih}\right)}$$

First we show that $D_H(f_0, \hat{f}_n) \xrightarrow{p} 0$. Let $X^n = ((\boldsymbol{X_1}, Y_1), \ldots, (\boldsymbol{X_n}, Y_n))$. For any $\epsilon > 0$:

$$\begin{aligned}
D_H(f_0, \hat{f}_n) &\leq \int D_H(f_0, f)\, \mathrm{d}\pi_n(f|X^n) \\
&\qquad \text{By Jensen's Inequality} \\
&\leq \int_{A_\epsilon} D_H(f_0, f)\, \mathrm{d}\pi_n(f|X^n) + \int_{A_\epsilon^c} D_H(f_0, f)\, \mathrm{d}\pi_n(f|X^n) \\
&\leq \int_{A_\epsilon} \epsilon\, \mathrm{d}\pi_n(f|X^n) + \int_{A_\epsilon^c} D_H(f_0, f)\, \mathrm{d}\pi_n(f|X^n) \\
&\leq \epsilon + \int_{A_\epsilon^c} D_H(f_0, f)\, \mathrm{d}\pi_n(f|X^n)
\end{aligned}$$

The second term goes to zero in probability by Theorem 2.4.4 and ϵ is arbitrary, therefore $D_H(f_0, \hat{f}_n) \xrightarrow{p} 0$.

In the remaining part of the proof, for notational simplicity, we take $\hat{\mu}_n(\boldsymbol{x})$ and $\mu_0(\boldsymbol{x})$ to be $\hat{\mu}$ and $\hat{\mu}_0$ respectively. The Hellinger distance between f_0 and $\hat{f}_n$ is

$$\begin{aligned}
&D_H(f_0, \hat{f}_n) \\
&= \left(\iint \left[\sqrt{\hat{f}_n(\boldsymbol{x}, y)} - \sqrt{f_0(\boldsymbol{x}, y)} \right]^2 \mathrm{d}y\, \mathrm{d}x \right)^{1/2} \\
&= \left(\iint \tau(1-\tau) \left[\exp\left(-\frac{1}{2}(y - \hat{\mu}_n)(\tau - I_{(y \leq \hat{\mu}_n)}) \right) \right.\right. \\
&\qquad \left.\left. -\exp\left(-\frac{1}{2}(y - \mu_0)(\tau - I_{(y \leq \mu_0)}) \right) \right]^2 \mathrm{d}y\, \mathrm{d}\boldsymbol{x} \right)^{1/2} \\
&= \left(2 - 2 \iint \tau(1-\tau) \exp\left(-\frac{1}{2}(y - \hat{\mu}_n)(\tau - I_{(y \leq \hat{\mu}_n)}) - \frac{1}{2}(y - \mu_0)(\tau - I_{(y \leq \mu_0)}) \right) \mathrm{d}y\, \mathrm{d}\boldsymbol{x} \right)^{1/2}
\end{aligned}$$

$$\text{let, } T = -\frac{1}{2}(y-\hat{\mu}_n)(\tau - I_{(y\le\hat{\mu}_n)}) - \frac{1}{2}(y-\mu_0)(\tau - I_{(y\le\mu_0)})$$
$$= \left(2 - 2\iint \tau(1-\tau)\exp(T)\ \mathrm{d}y\,\mathrm{d}\boldsymbol{x}\right)^{1/2} \tag{B.1}$$

Now let's break T into two cases: (a) $\hat{\mu}_n \le \mu_0$, and (b) $\hat{\mu}_n > \mu_0$.

Case-(a) $\hat{\mu}_n \le \mu_0$

$$T = \begin{cases} -\left(y - \frac{\hat{\mu}_n+\mu_0}{2}\right)\tau, & \hat{\mu}_n \le \mu_0 < y \\ -\left(y - \frac{\hat{\mu}_n+\mu_0}{2}\right)\tau + \frac{(y-\mu_0)}{2}, & \hat{\mu}_n \le \frac{\hat{\mu}_n+\mu_0}{2} < y \le \mu_0 \\ -\left(y - \frac{\hat{\mu}_n+\mu_0}{2}\right)(\tau-1) - \frac{(y-\hat{\mu}_n)}{2}, & \hat{\mu}_n < y \le \frac{\hat{\mu}_n+\mu_0}{2} \le \mu_0 \\ -\left(y - \frac{\hat{\mu}_n+\mu_0}{2}\right)(\tau-1), & y \le \hat{\mu}_n \le \mu_0 \end{cases}$$

Case-(b) $\hat{\mu}_n > \mu_0$

$$T = \begin{cases} -\left(y - \frac{\hat{\mu}_n+\mu_0}{2}\right)\tau, & \mu_0 \le \hat{\mu}_n < y \\ -\left(y - \frac{\hat{\mu}_n+\mu_0}{2}\right)\tau + \frac{(y-\hat{\mu}_n)}{2}, & \mu_0 \le \frac{\hat{\mu}_n+\mu_0}{2} < y \le \hat{\mu}_n \\ -\left(y - \frac{\hat{\mu}_n+\mu_0}{2}\right)(\tau-1) - \frac{(y-\mu_0)}{2}, & \mu_0 < y \le \frac{\hat{\mu}_n+\mu_0}{2} \le \hat{\mu}_n \\ -\left(y - \frac{\hat{\mu}_n+\mu_0}{2}\right)(\tau-1), & y \le \mu_0 \le \hat{\mu}_n \end{cases}$$

Hence now,

$$\begin{aligned} &\int \tau(1-\tau)\exp(T)\ \mathrm{d}y \\ &= \int \left[I_{(\hat{\mu}_n\le\mu_0)} + I_{(\hat{\mu}_n>\mu_0)}\right]\tau(1-\tau)\exp(T)\,\mathrm{d}y \\ &= I_{(\hat{\mu}_n\le\mu_0)}\tau(1-\tau) \times \left[\int_{\mu_0}^{\infty} \exp\left\{-\left(y - \frac{\hat{\mu}_n+\mu_0}{2}\right)\tau\right\}\ \mathrm{d}y \right. \\ &\qquad + \int_{\frac{\hat{\mu}_n+\mu_0}{2}}^{\mu_0} \exp\left\{-\left(y - \frac{\hat{\mu}_n+\mu_0}{2}\right)\tau + \frac{(y-\mu_0)}{2}\right\}\ \mathrm{d}y \\ &\qquad + \int_{\hat{\mu}_n}^{\frac{\hat{\mu}_n+\mu_0}{2}} \exp\left\{-\left(y - \frac{\hat{\mu}_n+\mu_0}{2}\right)(\tau-1) - \frac{(y-\hat{\mu}_n)}{2}\right\}\ \mathrm{d}y \\ &\qquad \left. + \int_{-\infty}^{\hat{\mu}_n} \exp\left\{-\left(y - \frac{\hat{\mu}_n+\mu_0}{2}\right)(\tau-1)\right\}\ \mathrm{d}y\right] \end{aligned}$$

$$
\begin{aligned}
&+ I_{(\hat{\mu}_n>\mu_0)}\tau(1-\tau)\times\Bigg[\int_{\hat{\mu}_n}^{\infty}\exp\left\{-\left(y-\frac{\hat{\mu}_n+\mu_0}{2}\right)\tau\right\}\,\mathrm{d}y \\
&\qquad+\int_{\frac{\hat{\mu}_n+\mu_0}{2}}^{\hat{\mu}_n}\exp\left\{-\left(y-\frac{\hat{\mu}_n+\mu_0}{2}\right)\tau+\frac{(y-\hat{\mu}_n)}{2}\right\}\,\mathrm{d}y \\
&\qquad+\int_{\mu_0}^{\frac{\hat{\mu}_n+\mu_0}{2}}\exp\left\{-\left(y-\frac{\hat{\mu}_n+\mu_0}{2}\right)(\tau-1)-\frac{(y-\mu_0)}{2}\right\}\,\mathrm{d}y \\
&\qquad+\int_{-\infty}^{\mu_0}\exp\left\{-\left(y-\frac{\hat{\mu}_n+\mu_0}{2}\right)(\tau-1)\right\}\,\mathrm{d}y\Bigg] \\
&=\frac{1-\tau}{1-2\tau}\exp\left(-\frac{|\hat{\mu}_n-\mu_0|}{2}\tau\right)-\frac{\tau}{1-2\tau}\exp\left(-\frac{|\hat{\mu}_n-\mu_0|}{2}(1-\tau)\right)
\end{aligned}
$$

Substituting the above expression in Equation B.1 we get $D_H(f_0,\hat{f}_n)$ equal to,

$$
\left(2-2\int\left[\frac{1-\tau}{1-2\tau}\exp\left(-\frac{|\hat{\mu}_n-\mu_0|}{2}\tau\right)-\frac{\tau}{1-2\tau}\exp\left(-\frac{|\hat{\mu}_n-\mu_0|}{2}(1-\tau)\right)\right]\mathrm{d}\boldsymbol{x}\right)^{1/2}
$$

Since $D_H(f_0,\hat{f}_n)\xrightarrow{p}0$,

$$
\int\left[\frac{1-\tau}{1-2\tau}\exp\left(-\frac{|\hat{\mu}_n-\mu_0|}{2}\tau\right)-\frac{\tau}{1-2\tau}\exp\left(-\frac{|\hat{\mu}_n-\mu_0|}{2}(1-\tau)\right)\right]\mathrm{d}\boldsymbol{x}\xrightarrow{p}1
$$

Our next step is to show that above expression implies that $|\hat{\mu}_n-\mu_0|\to 0$ a.s. on a set Ω, with probability tending to 1, and hence $\int|\hat{\mu}_n-\mu_0|\,\mathrm{d}\boldsymbol{x}\xrightarrow{p}0$.

We are going to prove this using contradiction technique. Suppose that, $|\hat{\mu}_n-\mu_0|\nrightarrow 0$ a.s. on Ω. Then, there exists an $\epsilon>0$ and a subsequence $\hat{\mu}_{n_i}$ such that $|\hat{\mu}_{n_i}-\mu_0|>\epsilon$ on a set A with $P(A)>0$. Now decompose the integral as

$$
\begin{aligned}
&\int\left[\frac{1-\tau}{1-2\tau}\exp\left(-\frac{|\hat{\mu}_n-\mu_0|}{2}\tau\right)-\frac{\tau}{1-2\tau}\exp\left(-\frac{|\hat{\mu}_n-\mu_0|}{2}(1-\tau)\right)\right]\mathrm{d}\boldsymbol{x} \\
&\quad=\int_A\left[\frac{1-\tau}{1-2\tau}\exp\left(-\frac{|\hat{\mu}_n-\mu_0|}{2}\tau\right)-\frac{\tau}{1-2\tau}\exp\left(-\frac{|\hat{\mu}_n-\mu_0|}{2}(1-\tau)\right)\right]\mathrm{d}\boldsymbol{x} \\
&\qquad+\int_{A^c}\left[\frac{1-\tau}{1-2\tau}\exp\left(-\frac{|\hat{\mu}_n-\mu_0|}{2}\tau\right)-\frac{\tau}{1-2\tau}\exp\left(-\frac{|\hat{\mu}_n-\mu_0|}{2}(1-\tau)\right)\right]\mathrm{d}\boldsymbol{x} \\
&\quad\le\underbrace{P(A)}_{>0}\underbrace{\left[\frac{(1-\tau)\exp(-\epsilon\tau/2)-\tau\exp(-\epsilon(1-\tau)/2)}{1-2\tau}\right]}_{<1\ \ (\max=1\text{ for }\epsilon=0)\text{ and strictly }\downarrow\text{ for }\epsilon\in(0,\infty)}+\underbrace{P(A^c)}_{<1}\ <1
\end{aligned}
$$

So we have a contradiction since the integral converges in probability to 1. Thus $|\hat{\mu}_n-\mu_0|\to 0$ a.s. on Ω. Once we apply Scheffe's theorem we get $\int|\hat{\mu}_n-\mu_0|\,\mathrm{d}\boldsymbol{x}\to 0$ a.s. on Ω and hence $\int|\hat{\mu}_n-\mu_0|\,\mathrm{d}\boldsymbol{x}\xrightarrow{p}0$. □

Below we prove the Theorem 2.4.3 and for that we make use of Theorem 2.4.4 and Corollary 2.4.5.

B.3 Theorem 2.4.3 Proof

We proceed by showing that with $\mathcal{F}_n$ as in (2.13), the prior π_n of Theorem 2.4.3 satisfies the condition (i) and (ii) of Theorem 2.4.4.

The proof of Theorem 2.4.4 condition-(i) presented in Lee (2000, proof of Theorem 1 on p. 639) holds in BQRNN case without any change. Next we need to show that condition-(ii) holds in BQRNN model. Let K_δ be the KL-neighborhood of the true density f_0 as in (2.12) and μ_0 the corresponding conditional quantile function. We first fix a closely approximating neural network μ^* of μ_0. We then find a neighborhood M_ς of μ^* as in (A.7) and show that this neighborhood has sufficiently large prior probability. Suppose that μ_0 is continuous. For any $\delta > 0$, choose $\epsilon = \delta/2$ in theorem from Funahashi (1989, Theorem 1 on p.184) and let μ^* be a neural network such that $\sup_{x\in\mathcal{X}} |\mu^* - \mu_0| < \epsilon$. Let $\varsigma = (\sqrt{\epsilon}/5n^a) = \sqrt{(\delta/50)}n^{-a}$ in Lemma A.1.5. Then following derivation shows us that for any $\tilde{\mu} \in M_\varsigma, D_K(f_0, \tilde{f}) \leq \delta$ i.e. $M_\varsigma \subset K_\delta$.

$$
\begin{aligned}
D_K(f_0, \tilde{f}) &= \iint f_0(x,y) \log \frac{f_0(x,y)}{\tilde{f}(x,y)} \,\mathrm{d}y \,\mathrm{d}x \\
&= \iint \left[(y-\tilde{\mu})(\tau - I_{(y\leq\tilde{\mu})}) - (y-\mu_0)(\tau - I_{(y\leq\mu_0)})\right] f_0(y|x)\, f_0(x) \,\mathrm{d}y \,\mathrm{d}x \\
&\quad \text{let, } T = (y-\tilde{\mu})(\tau - I_{(y\leq\tilde{\mu})}) - (y-\mu_0)(\tau - I_{(y\leq\mu_0)}) \\
&= \int \left[\int T f_0(y|x) \,\mathrm{d}y\right] f_0(x) \,\mathrm{d}x
\end{aligned}
$$

Now let's break T into two cases: (a) $\tilde{\mu} \geq \mu_0$, and (b) $\tilde{\mu} < \mu_0$.

Case-(a) $\tilde{\mu} \geq \mu_0$

$$
T = \begin{cases} (\mu_0 - \tilde{\mu})\tau, & \mu_0 \leq \tilde{\mu} < y \\ (\mu_0 - \tilde{\mu})\tau - (y - \tilde{\mu}), & \mu_0 < y \leq \tilde{\mu} \\ (\mu_0 - \tilde{\mu})(\tau - 1), & y \leq \mu_0 \leq \tilde{\mu} \end{cases}
$$

Case-(b) $\tilde{\mu} \leq \mu_0$

$$T = \begin{cases} (\mu_0 - \tilde{\mu})\tau, & \tilde{\mu} \leq \mu_0 < y \\ (\mu_0 - \tilde{\mu})(\tau - 1) + (y - \tilde{\mu}), & \tilde{\mu} < y \leq \mu_0 \\ (\mu_0 - \tilde{\mu})(\tau - 1), & y \leq \tilde{\mu} \leq \mu_0 \end{cases}$$

So now,

$$\begin{aligned}
&\int T f_0(y|x)\, \mathrm{d}y \\
&= \int \left[I_{(\tilde{\mu}-\mu_0 \geq 0)} \times \left\{ (\tilde{\mu} - \mu_0)(1-\tau) I_{(y \leq \mu_0)} - (y - \tilde{\mu}) I_{(\mu_0 < y \leq \tilde{\mu})} - (\tilde{\mu} - \mu_0)\tau I_{(y > \mu_0)} \right\} \right. \\
&\quad \left. + I_{(\tilde{\mu}-\mu_0 < 0)} \times \left\{ (\tilde{\mu} - \mu_0)(1-\tau) I_{(y \leq \mu_0)} + (y - \tilde{\mu}) I_{(\tilde{\mu} < y \leq \mu_0)} - (\tilde{\mu} - \mu_0)\tau I_{(y > \mu_0)} \right\} \right] f_0(y|x)\, \mathrm{d}y \\
&= \int \left[(\tilde{\mu} - \mu_0)(1-\tau) I_{(y \leq \mu_0)} - (\tilde{\mu} - \mu_0)\tau I_{(y > \mu_0)} \right. \\
&\quad \left. - (y - \mu_0 + \mu_0 - \tilde{\mu}) I_{(\mu_0 < y \leq \tilde{\mu})} + (y - \mu_0 + \mu_0 - \tilde{\mu}) I_{(\tilde{\mu} < y \leq \mu_0)} \right] f_0(y|x)\, \mathrm{d}y \\
&\quad \text{let, } z = y - \mu_0, b = \tilde{\mu} - \mu_0 \text{ and note that } P(y \leq \mu_0|x) = \tau, \text{ and } P(y > \mu_0|x) = 1 - \tau. \\
&= E\left[-(z-b) I_{(0<z<b)} + (z-b) I_{(b<z<0)} | x \right] \\
&\leq E\left[b I_{(0<z<b)} - b I_{(b<z<0)} | x \right] \\
&= |b| \times [P(0 < z < b|x) + P(b < z < 0|x)] \\
&= |b| \times P(0 < |z| < |b|\, |x) \\
&\leq |b|
\end{aligned}$$

Hence,

$$\begin{aligned}
\iint T f_0(y|x)\, \mathrm{d}y\, \mathrm{d}x &\leq \int |b|\, f_0(x)\, \mathrm{d}x \\
&= \int |\tilde{\mu} - \mu_0|\, f_0(x)\, \mathrm{d}x \\
&= \int |\tilde{\mu} - \mu^* + \mu^* - \mu_0|\, f_0(x)\, \mathrm{d}x \\
&\leq \int \left[\sup_{x \in \mathcal{X}} |\tilde{\mu} - \mu^*| + \sup_{x \in \mathcal{X}} |\mu^* - \mu_0| \right] f_0(x)\, \mathrm{d}x
\end{aligned}$$

Use Lemma A.1.5 to bound the first term and

use Funahashi (1989, Theorem 1 on p.184) to bound the second term.

$$\leq \int [\epsilon + \epsilon]\, f_0(x)\, \mathrm{d}x$$

$$= 2\epsilon = \delta$$

Finally we prove that $\forall \delta, \nu > 0, \exists N_\nu$ s.t. $\pi_n(K_\delta) \geq \exp(-n\nu)\ \forall n \geq N_\nu$,

$$\pi_n(K_\delta) \geq \pi_n(M_\varsigma)$$

$$= \prod_{i=1}^{\tilde{d}_n} \int_{\theta_i - \varsigma}^{\theta_i + \varsigma} \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(-\frac{1}{2\sigma_0^2} u^2\right) \mathrm{d}u$$

$$\geq \prod_{i=1}^{\tilde{d}_n} 2\varsigma \inf_{u \in [\theta_i - 1, \theta_i + 1]} \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(-\frac{1}{2\sigma_0^2} u^2\right)$$

$$= \prod_{i=1}^{\tilde{d}_n} \varsigma \sqrt{\frac{2}{\pi\sigma_0^2}} \exp\left(-\frac{1}{2\sigma_0^2} \vartheta_i\right)$$

$$\vartheta_i = \max((\theta_i - 1)^2, (\theta_i + 1)^2)$$

$$\geq \left(\varsigma \sqrt{\frac{2}{\pi\sigma_0^2}}\right)^{\tilde{d}_n} \exp\left(-\frac{1}{2\sigma_0^2} \vartheta \tilde{d}_n\right) \qquad \text{where, } \vartheta = \max_i(\vartheta_1, \ldots, \vartheta_{\tilde{d}_n})$$

$$= \exp\left(-\tilde{d}_n \left[a \log n - \log \sqrt{\frac{\delta}{25\pi\sigma_0^2}}\right] - \frac{1}{2\sigma_0^2} \vartheta \tilde{d}_n\right)$$

$$\varsigma = \sqrt{\frac{\delta}{50}} n^{-a}$$

$$\geq \exp\left(-\left[2a \log n + \frac{\vartheta}{2\sigma_0^2}\right] \tilde{d}_n\right) \qquad \text{for large } n$$

$$\geq \exp\left(-\left[2a \log n + \frac{\vartheta}{2\sigma_0^2}\right] (p+3) n^a\right)$$

$$\tilde{d}_n = (p+2)\tilde{k}_n + 1 \leq (p+3) n^a$$

$$\geq \exp(-n\nu) \qquad \text{for any } \nu \text{ and } \forall n \geq N_\nu \text{ for some } N_\nu$$

Hence, we have proved that both the conditions of Theorem 2.4.4 hold. The result of Theorem 2.4.3 thereby follows from the Corollary 2.4.5 which is derived from Theorem 2.4.4. □

Further, we can use similar argument to show that a neural network can approximate

any L_2 function arbitrarily closely. Note for any L_2 function h, $\|h\|_2 \geq \|h\|_1$, so

$$\left(\int (\mu - \mu_0)^2 \, \mathrm{d}\lambda(x) \right)^{\frac{1}{2}} < \epsilon \implies \int |\mu - \mu_0| \, \mathrm{d}\lambda(x) < \epsilon$$

Hence,

$$\begin{aligned}
D_K(f_0, \tilde{f}) &\leq \int |\tilde{\mu} - \mu_0| \, f_0(x) \, \mathrm{d}x \\
&\leq \int \left[\sup_{x \in \mathcal{X}} |\tilde{\mu} - \mu| + \sup_{x \in \mathcal{X}} |\mu - \mu_0| \right] f_0(x) \, \mathrm{d}x \\
&\qquad \text{Use Hornik et al. (1989, Theorem 2.4 on p.362) and Lemma A.1.5} \\
&\leq \int [\epsilon + \epsilon] \, f_0(x) \, \mathrm{d}x \\
&= 2\epsilon = \delta
\end{aligned}$$

□

CHAPTER 3

LAYER ADAPTIVE NODE SELECTION IN BAYESIAN NEURAL NETWORKS

3.1 Introduction

Deep learning profoundly impacts science and society due to its impressive empirical success driven primarily by copious amounts of datasets, ever increasing computational resources, and deep neural network's (DNN) ability to learn task-specific representations (LeCun et al., 2015). The key characteristic of deep learning is that accuracy empirically scales with the size of the model and the amount of training data. As such, large neural network models such as OpenAI GPT-3 (175 Billion) now typify the state-of-the-art across multiple domains such as natural language processing, computer vision, speech recognition etc. Nevertheless deep neural networks do have some drawbacks despite their wide ranging applications. First, this form of model scaling is exorbitantly prohibitive in terms of computational requirements, financial commitment, energy requirements etc. Second, DNNs tend to overfit leading to poor generalization in practice (Zhang et al., 2017). Finally, there are numerous scenarios where training and deploying such huge models is practically infeasible. Examples of such scenarios include federated learning, autonomous vehicles, robotics, recommendation systems where models have to be refreshed daily/hourly or in an online manner for optimal performance.

A promising direction for addressing these issues while improving the efficiency of DNNs is exploiting sparsity. From a practical perspective, it has been well-known that neural networks can be sparsified without significant loss in performance (Mozer and Smolensky, 1988; LeCun et al., 1990; Hassibi and Stork, 1993) and there is growing evidence that it is more so in the case of modern DNNs (Han et al., 2015). Recently Frankle and Carbin (2019) proposed the lottery ticket (LT) hypothesis, namely that there exist sparse, trainable sub-networks within

the larger network which can match the performance of their dense counterpart. To this end, sparsity in DNNs provides a promising way to reduce the network complexity by eliminating nonessential connections from a neural network thereby improving its calibration (Hoefler et al., 2021). A number of approaches to neural network compression via sparsity have been proposed in the literature (Cheng et al., 2018; Gale et al., 2019). Recent approaches (Guo et al., 2016; Molchanov et al., 2017; Zhu and Gupta, 2018) in magnitude-based pruning of neural network weights provide high model compression rates with minimal accuracy loss. Whereas, sparse evolutionary training learns sparse neural networks with a fixed parameter budget throughout the training based on adaptive sparse connectivity (Mocanu et al., 2018).

A key feature of sparsity in neural networks is its structure on the topology of the neural network weights. Weight pruning approaches perform high model compression leading to significant storage cost reduction at test-time (Han et al., 2015, 2016; Molchanov et al., 2017; Zhu and Gupta, 2018; Frankle and Carbin, 2019). However, they result in unstructured sparsity in deep neural architectures which leads to inefficient computational gains in practical setups (Wen et al., 2016). Instead, inducing group sparsity on collection of incoming weights into a given node (or node selection) reduces the dimensions of weight matrices per layer allowing for significant computational savings. To that effect, edge selection and node selection approaches are complementary with the former leading to storage reduction and the later leading to computational speedup during inference stage. Although one may argue node selection arises as a byproduct of edge selection, we clearly demonstrate that an approach which targets node selection directly leads to lower latency models (smaller number of nodes per layer) compared to an approach which achieves node selection through edge selection.

Node selection through group sparsity in deep neural networks has been explored under frequentist setting in Murray and Chiang (2015), Alvarez and Salzmann (2016), Ochiai et al. (2017), Liu et al. (2017), Luo et al. (2017) and Louizos et al. (2018), etc.. On the other hand, Louizos et al. (2017), Neklyudov et al. (2017), and Ghosh et al. (2019) incorporate group

sparsity via shrinkage priors in Bayesian paradigm. These group sparsity approaches specifically applied for node selection have shown significant computational speedup and lower memory footprint at inference stage. However, all of the proposed methods of neuron selection perform ad-hoc pruning requiring fine-tuned thresholding rules. Moreover, the posterior inference of network weights in Bayesian neural networks (BNN) through standard MCMC method, ex. Hamiltonian Monte Carlo (Neal, 1992), does not scale well to modern neural network architectures and large datasets used in practice. Instead computationally efficient variational inference as an alternative to MCMC (Jordan et al., 1999; Blei et al., 2017), has been explored in the context of edge selection both theoretically and numerically by Blundell et al. (2015), Chérief-Abdellatif (2020), and Bai et al. (2020). On the other hand, Louizos et al. (2017) and Ghosh et al. (2019) have explored variational inference for node selection problem. In this work, we propose a Gaussian spike-and-slab prior for automatic node selection in Bayesian neural networks thereby alleviating the need of an ad-hoc thresholding rule for pruning. Further for scalability, we develop a variational Bayes algorithm for posterior inference of BNN model parameters in our proposed model and demonstrate its numerical performance through simulation and real regression and classification datasets. Finally, we provide the theoretical guarantees to our node selection method under mild restrictions on the network topology.

Related Work. A closely related work to our proposed model is Bai et al. (2020)'s automated edge selection model using spike-and-slab prior. There the slab distribution controls the magnitude of weights and spike allows for the exact setting of weights to 0. We introduce spike-and-slab framework for node selection in BNNs and show the key resource efficiency trade-off between node and edge selection at test-time. There are two main advantages to node selection over edge selection (1) fewer parameters to train during optimization, (2) results in structurally compact network leading to computational speedup at test-time.

On the theoretical front, sparse BNNs have been studied in the works of Polson and Ročková (2018) and Sun et al. (2021). In the context of variational inference, sparse BNNs

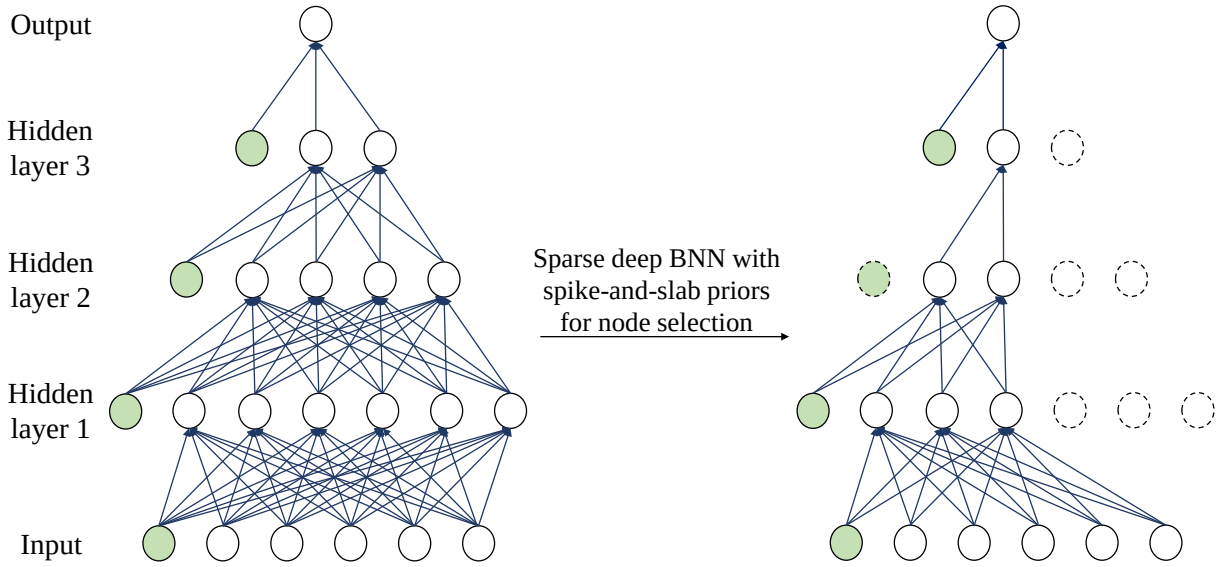


Figure 3.1 **Sparse neural network with node selection.** Sparse deep BNN using spike-and-slab priors achieves node selection in the given dense network on left leading to a sparse network on right.

have been studied in the recent works of Chérief-Abdellatif (2020) and Bai et al. (2020). All these works concentrate on the problem of edge selection facilitated through the use of Gaussian spike-and-slab priors. In the context of node selection, Ghosh et al. (2019) makes use of regularized horseshoe prior. The main limitations of their approach include (1) need for fine tuning of the thresholding rule for node selection, and (2) lack of a theoretical justification.

The only two works which have provided theoretical guarantees of their proposed sparse DNN methods under variational inference include those of Chérief-Abdellatif (2020) and Bai et al. (2020). Since they focus on the problem of edge selection, their theoretical developments are related to the results of Schmidt-Hieber (2020) (see the sieve construction in relation (4) in Schmidt-Hieber (2020)) and not directly extendable to our setup. Additionally, they assume certain restrictions on the network topology like (i) equal number of nodes in each layer, (ii) a known uniform bound B on all network weights, and (iii) a global sparsity parameter which may not lead to a structurally compact network. Although from a numerical standpoint, one may implicitly extend the problem of edge selection to node selection, the theoretical

guarantees of node selection consistency in sparse DNNs is not immediate.

Detailed Contributions.

1. We propose a Gaussian spike-and-slab node selection model and develop a variational Bayes approach for posterior inference of the model parameters. We call our approach **SS-IG** (**S**pike-and-**S**lab **I**ndependent **G**aussian) model.

2. We derive the variational posterior consistency using a functional space of neural networks which takes two layer dependent bounds, one which upper bounds the number of neurons in each layer and the other which upper bounds the L_1 norm of the weights incident onto each node of a layer. These layer dependent bounds allow the generalization of the theoretical results presented to guarantee the consistency of any generic shaped network structure. Further, it also guides the calculation of layer-wise prior inclusion probabilities which allow for optimal node recovery per layer in the computational experiments.

3. We measure the computational gains achieved by our approach using layer-wise node sparsities for shallow models and floating point operations in larger models. Our numerical results validate the proposed theoretical framework for the node selection in DNN models. These empirical experiments further justify the use of layer-wise node inclusion probabilities to facilitate the optimal node recovery.

3.2 Nonparametric Modeling: Deep Learning Approach

Non-parametric modeling assumes an arbitrary relationship between the response and the variables. The term non-parametric does not mean that the value lack inherent parameters, but rather that the parameters are flexible and can vary. In particular, we would like to find a function $\eta_0(\cdot) : \mathbb{R}^p \to \mathbb{R}$ such that $\eta_0(\boldsymbol{x})$ is a good approximation or a representation of y. A standard neural network is, technically speaking, parametric since it has a fixed number of parameters. However, most deep neural networks (DNNs) have thousands or millions of

parameters that they could be interpreted as nonparametric. In fact, it has been proven that in the limit of infinite width, a deep neural network can be seen as a Gaussian process, which is a nonparametric model (Lee et al., 2018).

Mathematically, let $Y \in \mathbb{R}$, $X \in \mathcal{X}$ be two random variables with the following conditional distribution

$$f_0(y|\boldsymbol{x}) = \exp\left[h_1(\eta_0(\boldsymbol{x}))y + h_2(\eta_0(\boldsymbol{x})) + h_3(y)\right] \tag{3.1}$$

where $\eta_0(\cdot) : \mathcal{X} \to \mathbb{R}$ is a continuous function satisfying certain regularity assumptions and $\mathcal{X}$ is usually a compact subspace of $\mathbb{R}^p$. Note, the functions h_1, h_2, h_3 are pre-determined and different choices give rise to different families of generalized linear models. For $h_1(u) = u$, $h_2(u) = -\log(1+e^u)$, $h_3(y) = 1$, we get the classification model. For $h_1(u) = u$, $h_2(u) = -u^2$, $h_3(y) = -y^2/2 - \log(2\pi)/2$, we get the regression model with $\sigma^2 = 1$. Usually $\mathcal{X} = \mathbb{R}^p$. Note, $\boldsymbol{x}$ is a feature vector from a marginal distribution P_X and y is the corresponding output from $Y|X = \boldsymbol{x}$ in (3.1). Let $P_{X,Y}$ be the joint distribution of (X, Y).

Let $g : \mathcal{X} \to \mathbb{R}$ be a measurable function, the risk of g is $R(g) = \int_{\mathcal{Y}\times\mathcal{X}} \mathcal{L}(Y, g(X))dP_{X,Y}$ for some loss function $\mathcal{L}$. The Bayes estimator minimizes this risk (Friedman et al., 2009). For regression with squared error loss and classification with 0-1 loss, the optimal Bayes estimators are $g^*(\boldsymbol{x}) = \eta_0(\boldsymbol{x})$ and $g^*(\boldsymbol{x}) = 1\{\eta_0(\boldsymbol{x}) \geq 0\}$ respectively. In practice, Bayes estimator is not useful since the function $\eta_0(\boldsymbol{x})$ is unknown. Thus, an estimator is obtained based on the training observations, $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\}$. A good estimator enjoys universal consistency properties, i.e., its risk approaches Bayes risk as $n \to \infty$ irrespective of P_X. To find this optimal class, we use Bayesian neural networks, $\eta_{\boldsymbol{\theta}}(\boldsymbol{x})$ with $\boldsymbol{\theta}$ denoting the network weights, as an approximation to $\eta_0(\boldsymbol{x})$.

Mathematical Framework

For $\boldsymbol{x} \in \mathbb{R}^p$, consider a BNN with L hidden layers with $k_1, \cdots, k_L$ the number of nodes in the hidden layers with $k_0 = p$, $k_{L+1} = 1$ (in regression). $k_{L+1} > 1$ *allows the generalization*

to $Y \in \mathbb{R}^d, d > 1$, thereby providing a handle on multi class classification problems. The total number of parameters is $K = \prod_{l=0}^{L} k_{l+1}(k_l + 1)$. With $\boldsymbol{W}_l = [\boldsymbol{w}_l^0, \boldsymbol{W}_l^1]$, let

$$\eta_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{w}_L^0 + \boldsymbol{W}_L^1 \psi(\boldsymbol{w}_{L-1}^0 + \boldsymbol{W}_{L-1}^1 \psi(\cdots \psi(\boldsymbol{w}_1^0 + \boldsymbol{W}_1^1 \psi(\boldsymbol{w}_0^0 + \boldsymbol{W}_0^1 \boldsymbol{x})))), \tag{3.2}$$

where ψ is a nonlinear activation function, $\boldsymbol{w}_l^0$ are $k_{l+1} \times 1$ vectors and $\boldsymbol{W}_l^1$ are $k_{l+1} \times k_l$ matrices. Using the BNN in (3.2) to approximate the true function $\eta_0(\boldsymbol{x})$, conditional probabilities of $Y|X = \boldsymbol{x}$ are

$$f_{\boldsymbol{\theta}}(y|\boldsymbol{x}) = \exp\left[h_1(\eta_{\boldsymbol{\theta}}(\boldsymbol{x}))y + h_2(\eta_{\boldsymbol{\theta}}(\boldsymbol{x})) + h_3(y)\right]. \tag{3.3}$$

Thus, the likelihood function for the data $\mathcal{D}$ under the model and the truth is

$$P_{\boldsymbol{\theta}}^n = \prod_{i=1}^{n} f_{\boldsymbol{\theta}}(y_i|\boldsymbol{x}_i), \quad P_0^n = \prod_{i=1}^{n} f_0(y_i|\boldsymbol{x}_i). \tag{3.4}$$

3.3 Spike-and-Slab Independent Gaussian Node Selection

3.3.1 Model

To allow for automatic node selection, we consider a spike-and-slab prior consisting of a Dirac spike (δ_0) at 0 and a slab distribution (Mitchell and Beauchamp, 1988). The spike part is represented by an indicator variable which is set to 0 if a node is not present in the network. The slab part comes from a Gaussian distributed random variable. To allow for the layer-wise node selection, we assume that the prior inclusion probability λ_l varies as a function of the layer index l. The symbol i.d. is used to denote independently distributed random variables.

Prior: We assume a spike-and-slab prior of the following form with z_{lj} as the indicator for the presence of j^{th} node in the l^{th} layer

$$\overline{\boldsymbol{w}}_{lj}|z_{lj} \overset{\text{i.d.}}{\sim} \left[(1 - z_{lj})\boldsymbol{\delta}_0 + z_{lj} N(0, \sigma_0^2 \boldsymbol{I})\right], \quad z_{lj} \overset{\text{i.d.}}{\sim} \text{Ber}(\lambda_l)$$

where $l = 0, \ldots, L$, $j = 1, \ldots, k_{l+1}$. Also, $\overline{\boldsymbol{w}}_{lj} = (\overline{w}_{lj1}, \ldots, \overline{w}_{ljk_l+1})$ is a vector of edges incident on the j^{th} node in the l^{th} layer. In the above formula, note $\boldsymbol{\delta}_0$ is a Dirac spike

vector of dimension $k_l + 1$ with all entries zero and $\boldsymbol{I}$ is the identity matrix of dimension $k_l + 1 \times k_l + 1$. Furthermore, z_{lj} with $j = (1, \ldots, k_{l+1})$ all follow Bernoulli(λ_l) to allow for common prior inclusion probability, λ_l, for each node from a given layer l. We set $\lambda_L = 1$ to ensure no node selection occurs in the output layer.

Posterior: With $\boldsymbol{z}_l = (z_{l1}, \cdots, z_{lk_{l+1}})$, let $\boldsymbol{z} = (\boldsymbol{z}_1, \cdots, \boldsymbol{z}_L)$ denote the vector of all indicator variables. The posterior distribution of $(\boldsymbol{\theta}, \boldsymbol{z})$ given $\mathcal{D}$ is given by

$$\pi(\boldsymbol{\theta}, \boldsymbol{z}|\mathcal{D}) = \frac{P_{\boldsymbol{\theta}}^n \pi(\boldsymbol{\theta}|\boldsymbol{z})\pi(\boldsymbol{z})}{\sum_{\boldsymbol{z}} \int P_{\boldsymbol{\theta}}^n \pi(\boldsymbol{\theta}|\boldsymbol{z})\pi(\boldsymbol{z}) d\boldsymbol{\theta}} = \frac{P_{\boldsymbol{\theta}}^n \pi(\boldsymbol{\theta}|\boldsymbol{z})\pi(\boldsymbol{z})}{m(\mathcal{D})} \tag{3.5}$$

where $P_{\boldsymbol{\theta}}^n = \prod_{i=1}^n f_{\boldsymbol{\theta}}(y_i|\boldsymbol{x}_i)$ is the likelihood function as in (3.4), $\pi(\boldsymbol{z})$ is the probability mass function of $\boldsymbol{z}$ with respect to the counting measure and $\pi(\boldsymbol{\theta}|\boldsymbol{z})$ is the conditional probability density function with respect to the Lebesgue measure of $\boldsymbol{\theta}$ given $\boldsymbol{z}$. Further, $m(\mathcal{D})$ is the marginal density of the data and is free of $(\boldsymbol{\theta}, \boldsymbol{z})$.

Let $\widetilde{\pi}(\boldsymbol{\theta}) = \sum_{\boldsymbol{z}} \pi(\boldsymbol{\theta}, \boldsymbol{z})$ be the marginal prior of $\boldsymbol{\theta}$. We shall use the notation

$$\widetilde{\Pi}(\mathcal{A}) = \int_{\mathcal{A}} \widetilde{\pi}(\boldsymbol{\theta}) d\boldsymbol{\theta} \tag{3.6}$$

to denote the probability distribution function corresponding to the density function $\widetilde{\pi}$. The marginal posterior of $\boldsymbol{\theta}$ expressed as a function of the marginal prior for $\boldsymbol{\theta}$ is

$$\widetilde{\pi}(\boldsymbol{\theta}|\mathcal{D}) = \sum_{\boldsymbol{z}} \pi(\boldsymbol{\theta}, \boldsymbol{z}|\mathcal{D}) = \frac{P_{\boldsymbol{\theta}}^n \widetilde{\pi}(\boldsymbol{\theta})}{\int P_{\boldsymbol{\theta}}^n \widetilde{\pi}(\boldsymbol{\theta}) d\boldsymbol{\theta}} = \frac{P_{\boldsymbol{\theta}}^n \widetilde{\pi}(\boldsymbol{\theta})}{m(\mathcal{D})}$$

Thus, the probability distribution function corresponding to the density function $\widetilde{\pi}(|\mathcal{D})$ is then given by

$$\widetilde{\Pi}(\mathcal{A}|\mathcal{D}) = \int_{\mathcal{A}} \widetilde{\pi}(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta} \tag{3.7}$$

Variational family: We posit the following mean field variational family ($\mathcal{Q}^{\mathbf{MF}}$) on network weights as

$$\mathcal{Q}^{\mathbf{MF}} = \left\{ \overline{\boldsymbol{w}}_{lj}|z_{lj} \overset{\text{i.d.}}{\sim} \left[(1 - z_{lj})\boldsymbol{\delta}_0 + z_{lj} N(\boldsymbol{\mu}_{lj}, \text{diag}(\boldsymbol{\sigma}_{lj}^2))\right], \quad z_{lj} \overset{\text{i.d.}}{\sim} \text{Ber}(\gamma_{lj}) \right\}$$

for $l = 0, \ldots, L$, $j = 1, \ldots, k_{l+1}$. This ensures that weight distributions follow spike-and-slab structure which allows for node sparsity through variational approximation. Further, the

weight distributions conditioned on the node indicator variables are all independent of each other (hence use of the term mean field family). The variational distribution of parameters obtained post optimization will then inherently prune away redundant nodes from each layer. Also, Gaussian distribution for slab component is widely popular for approximating neural network weight distributions (Blundell et al., 2015; Louizos et al., 2017; Bai et al., 2020).

Additionally, $\boldsymbol{\mu}_{lj} = (\mu_{lj1}, \ldots, \mu_{ljk_l+1})$ and $\boldsymbol{\sigma}^2_{lj} = (\sigma^2_{lj1}, \ldots, \sigma^2_{ljk_l+1})$ denote the vectors of variational mean and standard deviation parameters of the edges incident on the j^{th} node in the l^{th} layer. Similarly, γ_{lj} denotes the variational inclusion probability of the j^{th} node in the l^{th} layer. We set $\gamma_{Lj} = 1$ to ensure no node selection occurs in the output layer.

Variational posterior: Variational posterior aims to reduce the Kullback-Leibler (KL) distance between a variational family and the true posterior (Blei and Lafferty, 2007; Hinton and Van Camp, 1993) as

$$\pi^* = \underset{q \in \mathcal{Q}^{\mathbf{MF}}}{\operatorname{argmin}} \; d_{\mathrm{KL}}(q, \pi(|\mathcal{D})) \tag{3.8}$$

where $d_{\mathrm{KL}}(q, \pi(|\mathcal{D}))$ denotes the KL-distance between q and $\pi(|\mathcal{D})$.

Note, the variational member q can be written as $q(\boldsymbol{\theta}, \boldsymbol{z}) = q(\boldsymbol{\theta}|\boldsymbol{z})q(\boldsymbol{z})$ where $q(\boldsymbol{z})$ is the probability mass function of $\boldsymbol{z}$ with respect to the counting measure and $q(\boldsymbol{\theta}|\boldsymbol{z})$ is the conditional density function given with respect to the Lebesgue measure of $\boldsymbol{\theta}$ given $\boldsymbol{z}$. Further,

$$\begin{aligned}
\pi^* &= \underset{q \in \mathcal{Q}^{\mathbf{MF}}}{\operatorname{argmin}} \sum_{\boldsymbol{z}} \int [\log q(\boldsymbol{\theta}, \boldsymbol{z}) - \log \pi(\boldsymbol{\theta}, \boldsymbol{z}|\mathcal{D})] q(\boldsymbol{\theta}, \boldsymbol{z}) d\boldsymbol{\theta} \\
&= \underset{q \in \mathcal{Q}^{\mathbf{MF}}}{\operatorname{argmin}} \left(\sum_{\boldsymbol{z}} \int [\log q(\boldsymbol{\theta}, \boldsymbol{z}) - \log \pi(\boldsymbol{\theta}, \boldsymbol{z}, \mathcal{D})] q(\boldsymbol{\theta}, \boldsymbol{z}) d\boldsymbol{\theta} + \log m(\mathcal{D}) \right) \\
&= \underset{q \in \mathcal{Q}^{\mathbf{MF}}}{\operatorname{argmin}} \; [-\mathrm{ELBO}(q, \pi(|\mathcal{D}))] + \log m(\mathcal{D}) = \underset{q \in \mathcal{Q}^{\mathbf{MF}}}{\operatorname{argmax}} \; \mathrm{ELBO}(q, \pi(|\mathcal{D}))
\end{aligned} \tag{3.9}$$

Since $\log m(\mathcal{D})$ is free from q, it suffices to maximize the evidence lower bound (ELBO) above.

Let $\widetilde{\pi}^*(\boldsymbol{\theta}) = \sum_{\boldsymbol{z}} \pi^*(\boldsymbol{\theta}|\boldsymbol{z})\pi^*(\boldsymbol{z})$ then $\widetilde{\pi}^*$ denotes the marginal variational posterior for $\boldsymbol{\theta}$.

We shall use the notation

$$\widetilde{\Pi}^*(\mathcal{A}) = \int_{\mathcal{A}} \widetilde{\pi}^*(\boldsymbol{\theta}) d\boldsymbol{\theta} \tag{3.10}$$

to denote the probability distribution function corresponding to the density function $\widetilde{\pi}^*$.

3.3.2 Algorithm

Evidence Lower Bound. The ELBO presented in (3.9) is given by $\mathcal{L} = -E_q[\log P_{\boldsymbol{\theta}}^n] + d_{\mathrm{KL}}(q, \pi)$ which is further simplified as

$$\begin{aligned}
&- E_q[\log P_{\boldsymbol{\theta}}^n] + d_{\mathrm{KL}}(q, \pi) \\
&= -\mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{z})q(\boldsymbol{z})}[\log P_{\boldsymbol{\theta}}^n] + d_{\mathrm{KL}}\Big(q(\boldsymbol{\theta}|\boldsymbol{z})q(\boldsymbol{z}), \pi(\boldsymbol{\theta}|\boldsymbol{z})\pi(\boldsymbol{z})\Big) \\
&= -\mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{z})q(\boldsymbol{z})}[\log P_{\boldsymbol{\theta}}^n] + \sum_{l,j} d_{\mathrm{KL}}(q(z_{lj})||\pi(z_{lj})) \\
&\quad + \sum_{l,j} \Big[q(\boldsymbol{z}_{lj} = 1) d_{\mathrm{KL}}(q(\overline{\boldsymbol{w}}_{lj}|z_{lj} = 1)||\pi(\overline{\boldsymbol{w}}_{lj}|z_{lj} = 1)) \\
&\qquad\qquad + q(\boldsymbol{z}_{lj} = 0) d_{\mathrm{KL}}(q(\overline{\boldsymbol{w}}_{lj}|z_{lj} = 0)||\pi(\overline{\boldsymbol{w}}_{lj}|z_{lj} = 0))\Big] \\
&= -\mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{z})q(\boldsymbol{z})}[\log P_{\boldsymbol{\theta}}^n] + \sum_{l,j} d_{\mathrm{KL}}(q(z_{lj})||\pi(z_{lj})) \\
&\quad + \sum_{l,j} q(\boldsymbol{z}_{lj} = 1) d_{\mathrm{KL}}(q(\overline{\boldsymbol{w}}_{lj}|z_{lj} = 1)||\pi(\overline{\boldsymbol{w}}_{lj}|z_{lj} = 1)) \\
&= -\mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{z})q(\boldsymbol{z})}[\log P_{\boldsymbol{\theta}}^n] + \sum_{l,j} d_{\mathrm{KL}}(q(z_{lj})||\pi(z_{lj})) \\
&\quad + \sum_{l,j} q(\boldsymbol{z}_{lj} = 1) d_{\mathrm{KL}}(N(\boldsymbol{\mu}_{lj}, \mathrm{diag}(\boldsymbol{\sigma}_{lj}^2))||N(0, \sigma_0^2 \boldsymbol{I}))
\end{aligned}$$

The KL of discrete variables appearing in the above expression creates a challenge in practical implementation. Jang et al. (2017) and Maddison et al. (2017) proposed to replace discrete random variable with its continuous relaxation. Specifically, the continuous relaxation approximation is achieved through Gumbel-softmax (GS) distribution, that is $q(z_{lj}) \sim \mathrm{Ber}(\gamma_{lj})$ is approximated by $q(\tilde{z}_{lj}) \sim \mathrm{GS}(\gamma_{lj}, \tau)$, where

$$\tilde{z}_{lj} = (1 + \exp(-\eta_{lj}/\tau))^{-1}, \quad \eta_{lj} = \log(\gamma_{lj}/(1 - \gamma_{lj})) + \log(u_{lj}/(1 - u_{lj})), \quad u_{lj} \sim U(0, 1)$$

Algorithm 3.1 Variational inference in SS-IG Bayesian neural networks

Inputs: training dataset, network architecture, and optimizer tuning parameters.
Model inputs: prior parameters for $\boldsymbol{\theta}$, $\boldsymbol{z}$.
Variational inputs: number of Monte Carlo samples S.
Output: Variational parameter estimates of network weights and sparsity.
Method: Set initial values of variational parameters.
repeat
    Generate S samples from $\boldsymbol{\zeta}_{lj} \sim N(0, \boldsymbol{I})$ and $u_{lj} \sim U(0, 1)$
    Generate S samples for $(z_{lj}, \tilde{z}_{lj})$ using u_{lj}
    Use $\boldsymbol{\mu}_{lj}, \boldsymbol{\sigma}_{lj}, \boldsymbol{\zeta}_{lj}$ and z_{lj} to compute loss (ELBO) in forward pass
    Use $\boldsymbol{\mu}_{lj}, \boldsymbol{\sigma}_{lj}, \boldsymbol{\zeta}_{lj}$ and $\tilde{z}_{lj}$ to compute gradient of loss in backward pass
    Update the variational parameters with gradient of loss using stochastic gradient descent algorithm (e.g. Adam (Kingma and Ba, 2015))
until change in ELBO $< \epsilon$

where τ is the temperature. We set $\tau = 0.5$ for this work (also see section 5 in Bai et al. (2020)). $\tilde{z}_{lj}$ is used in the backward pass for easier gradient calculation, while z_{lj} is used for selecting nodes in the forward pass. We use non-centered parameterization for the Gaussian slab variational approximation where $N(\boldsymbol{\mu}_{lj}, \text{diag}(\boldsymbol{\sigma}_{lj}^2))$ is reparameterized as $\boldsymbol{\mu}_{lj} + \boldsymbol{\sigma}_{lj} \odot \boldsymbol{\zeta}_{lj}$ for $\boldsymbol{\zeta}_{lj} \sim N(0, \boldsymbol{I})$, where $\odot$ denotes the entry-wise (Hadamard) product.

3.4 Theoretical Results

In this section, we develop the theoretical consistency of the variational posterior in (3.10) in context of node selection. Previous works which establish the statistical consistency of sparse deep neural networks do so only in the context of edge selection. Thereby, the works of Polson and Ročková (2018), Chérief-Abdellatif (2020) and Bai et al. (2020) use several results from the pioneer work of Schmidt-Hieber (2020). In addition to node selection consistency, we also relax certain network restrictions considered in the previous works. These restrictions include (1) equal number of nodes in each layer which restricts one from using any previous information on the number of nodes in the deep neural architecture (2) a known bound B on all the neural network weights as they essentially rely on the sieve construction in equation 3 of Schmidt-Hieber (2020) which assumes that L_∞ norm of all $\boldsymbol{\theta}$ entries is smaller than 1 (3) a global sparsity parameter s which does not always consider structurally sparse networks.

Towards the proof, firstly our sieve construction allows the number of nodes of the neural network to vary as a function of the layer. Secondly, instead of global sparsity parameter s (see the sieve construction in relation (4) of Schmidt-Hieber (2020)) we allow for layer wise sparsity vector $\boldsymbol{s}$ to account for the number of nodes in each layer. Finally, we relax the assumption of a known bound B by considering a sieve with a layer wise constraint (denoted by the vector $\boldsymbol{B}$) on the L_1 norm of the incoming edges of a node. Thus, our work extends on current literature along three directions (1) theoretically quantifies predictive performance of Bayesian neural networks with node based pruning (2) establishes that even without a fixed bound on network weights, one can recover true solution by appropriate choice of the prior (3) provides layer wise node inclusion probabilities to allow for structurally sparse solutions. The relaxation of these network structure assumptions requires us to provide the framework for node selection including appropriate sieve construction together with the derivation of the results in Schmidt-Hieber (2020) customized to our problem.

To establish the posterior contraction rates, we show that the variational posterior in (3.8) concentrates in shrinking Hellinger neighborhoods of the true density function P_0 with overwhelming probability. Since $\boldsymbol{X} \sim U[0,1]^p$, thus $f_0(\boldsymbol{x}) = f_{\boldsymbol{\theta}}(\boldsymbol{x}) = 1$. This further implies $P_0 = f_0(y|\boldsymbol{x})f_0(\boldsymbol{x}) = f_0(y|\boldsymbol{x})$ and similarly $P_{\boldsymbol{\theta}} = f_{\boldsymbol{\theta}}(y|\boldsymbol{x})$. We next define the Hellinger neighborhood of the true density P_0 as

$$\mathcal{H}_\varepsilon = \{\boldsymbol{\theta} : d_{\mathrm{H}}(P_0, P_{\boldsymbol{\theta}}) < \varepsilon\}$$

where the Hellinger distance between the true density function P_0 and the model density $P_{\boldsymbol{\theta}}$ is

$$d^2_{\mathrm{H}}(P_0, P_{\boldsymbol{\theta}}) = \frac{1}{2}\int \left(\sqrt{f_{\boldsymbol{\theta}}(y|\boldsymbol{x})} - \sqrt{f_0(y|\boldsymbol{x})}\right)^2 dyd\boldsymbol{x}$$

We also define the KL neighborhood of the true density P_0 as

$$\mathcal{N}_\varepsilon = \{\boldsymbol{\theta} : d_{\mathrm{KL}}(P_0, P_{\boldsymbol{\theta}}) < \varepsilon\}$$

where the KL distance d_{KL} between the true density function P_0 and the model density $P_{\boldsymbol{\theta}}$

is

$$d_{\mathrm{KL}}(P_0, P_{\boldsymbol{\theta}}) = \int \log \frac{f_0(y|\boldsymbol{x})}{f_{\boldsymbol{\theta}}(y|\boldsymbol{x})} f_0(y|\boldsymbol{x}) dy d\boldsymbol{x}$$

Let $\boldsymbol{k} = (k_0, \cdots, k_{L+1})$ be the node vector, $\overline{\boldsymbol{W}}_l = (\boldsymbol{w}_{l1}^\top, \cdots, \boldsymbol{w}_{lk_{l+1}}^\top)^\top$ be the row representation of $\overline{\boldsymbol{W}}_l$ and $\widetilde{\boldsymbol{w}}_l = (||\boldsymbol{w}_{l1}||_1, \cdots, ||\boldsymbol{w}_{lk_{l+1}}||_1)$ be the vector of L_1 norms of the rows of $\overline{\boldsymbol{W}}_l$. Next we consider layer-wise sparsity, $\boldsymbol{s} = (s_1, \cdots, s_L)$ for node selection. Similarly, we consider layer-wise norm constraints, $\boldsymbol{B} = (B_1, \cdots, B_L)$ on L_1 norms of weights including bias incident onto any given node in each layer. Based on $\boldsymbol{s}$ and $\boldsymbol{B}$, we define the following sieve of neural networks (check definition A.1.1).

$$\mathcal{F}(L, \boldsymbol{k}, \boldsymbol{s}, \boldsymbol{B}) = \{\eta_{\boldsymbol{\theta}} \in (3.2) : ||\widetilde{\boldsymbol{w}}_l||_0 \leq s_l, ||\widetilde{\boldsymbol{w}}_l||_\infty \leq B_l\}. \tag{3.11}$$

The construction of a sieve is one of the most important tools towards the proof of consistency in infinite-dimensional spaces. In the works of Schmidt-Hieber (2020), Polson and Ročková (2018), Chérief-Abdellatif (2020) and Bai et al. (2020), the sieve in the context of edge selection is given by

$$\mathcal{F}(L, \boldsymbol{k}, s) = \{\eta_{\boldsymbol{\theta}} \in (3.2) : ||\boldsymbol{\theta}||_0 \leq s, ||\boldsymbol{\theta}||_\infty \leq 1\}.$$

which works with an overall sparsity parameter s. In addition, note the L_∞ norm of all the entries in $\boldsymbol{\theta}$ is assumed to be known constant equal to 1 (see relation (4) in Schmidt-Hieber (2020) and section 4 in Polson and Ročková (2018)). Section 3 in Bai et al. (2020) does not explicitly mention the dependence of their sieve on some fixed bound B on the edges in a network, however, their derivations on covering numbers (see proof of Lemma 1.2 in the supplement of Bai et al. (2020)) borrow results from Schmidt-Hieber (2020) which is based on sieve with $B = 1$.

Consider any sequence ϵ_n. For Lemmas 3.4.1 and 3.4.2, we use the sieve $\mathcal{F}(L, \boldsymbol{k}, \boldsymbol{s}, \boldsymbol{B})$ in (3.11) with $\boldsymbol{s} = \boldsymbol{s}^\circ$ and $\boldsymbol{B} = \boldsymbol{B}^\circ$ where $s_l^\circ + 1 = n\epsilon_n^2/(\sum_{j=0}^L u_j)$ and $\log B_l^\circ = (n\epsilon_n^2)/((L+1)\sum_{j=0}^L (s_j^\circ + 1))$ with $u_l = (L+1)^2(\log n + \log(L+1) + \log k_{l+1} + \log(k_l + 1))$. Note, s_l° and B_l° do not depend on l.

Lemma 3.4.1 below holds when the covering number (check definition A.1.2) of the functions which belong to the sieve $\mathcal{F}(L, \boldsymbol{k}, \boldsymbol{s}^\circ, \boldsymbol{B}^\circ)$ is well under control. Lemma 3.4.2 below states that for the same choice of the sieve, the prior gives sufficiently small probabilities on the complement space $\mathcal{F}(L, \boldsymbol{k}, \boldsymbol{s}^\circ, \boldsymbol{B}^\circ)^c$ (see the discussion under Theorem 3.4.4 for more details).

For the subsequent results, the symbol $\mathcal{A}^c$ is used to denote complement of a set $\mathcal{A}$.

Lemma 3.4.1 (Existence of Test Functions)**.** *Let $\epsilon_n \to 0$ and $n\epsilon_n^2 \to \infty$. There exists a testing function $\phi \in [0,1]$ and constants $C_1, C_2 > 0$,*

$$\mathbb{E}_{P_0}(\phi) \leq exp\{-C_1 n\epsilon_n^2\}$$

$$\sup_{\boldsymbol{\theta} \in \mathcal{H}_{\epsilon_n}^c, \eta_{\boldsymbol{\theta}} \in \mathcal{F}(L, \boldsymbol{k}, \boldsymbol{s}^\circ, \boldsymbol{B}^\circ)} \mathbb{E}_{P_{\boldsymbol{\theta}}}(1-\phi) \leq exp\{-C_2 n d_H^2(P_0, P_{\boldsymbol{\theta}})\}$$

where $\mathcal{H}_{\epsilon_n} = \{\boldsymbol{\theta} : d_H(P_0, P_{\boldsymbol{\theta}}) \leq \epsilon_n\}$ is the Hellinger neighborhood of radius ϵ_n.

Lemma 3.4.2 (Prior mass condition.)**.** *Let $\epsilon_n \to 0$, $n\epsilon_n^2 \to \infty$ and $n\epsilon_n^2 / \sum_{l=0}^L u_l \to \infty$, then for $\widetilde{\Pi}$ as in* (3.6) *and some constant $C_3 > 0$,*

$$\widetilde{\Pi}(\mathcal{F}(L, \boldsymbol{k}, \boldsymbol{s}^\circ, \boldsymbol{B}^\circ)^c) \leq exp(-C_3 n\epsilon_n^2 / \sum_{l=0}^{L} u_l)$$

Whereas Lemmas 3.4.1 and 3.4.2 work with a specific choice of the sieve, the following Lemma 3.4.3 is developed for any generic choice of sieve indexed by $\boldsymbol{s}$ and $\boldsymbol{B}$. The final piece of the theory developed next tries to addresses two main questions (1) Can we get a sparse network solution whose layer-wise sparsity levels and L_1 norms of incident edges (including the bias) of the nodes are controlled at levels $\boldsymbol{s}$ and $\boldsymbol{B}$ respectively? (2) Does this sparse network retain the same predictive performance as the original network?

In this direction, let

$$\xi = \min_{\eta_{\boldsymbol{\theta}} \in \mathcal{F}(L, \boldsymbol{k}, \boldsymbol{s}, \boldsymbol{B})} ||\eta_{\boldsymbol{\theta}} - \eta_0||_\infty^2$$

Based on the values $\boldsymbol{s}$ and $\boldsymbol{B}$, we also define

$$\vartheta_l = {B_l}^2/(k_l+1) + \sum_{m=0, m\neq l}^{L} \log B_m + L + \log k_{l+1} + \log(k_l+1) + \log n + \log(\sum_{m=0}^{L} u_m)$$

$$r_l = s_l(k_l + 1)\vartheta_l/n \tag{3.12}$$

Lemma 3.4.3 has two sub conditions. Condition 1. requires that shrinking KL neighborhood of the true density function P_0 gets sufficiently large probability. This along with Lemma 3.4.1 and 3.4.2 is an essential condition to guarantee the convergence of the true posterior in (3.5). Condition 2. is the assumption needed to control the KL distance between true posterior and variational posterior and thereby guarantees the convergence of the variational posterior in (3.8) (see the discussion under Theorem 3.4.4 for more details).

Lemma 3.4.3 (Kullback-Leibler conditions)**.** *Suppose* $\sum_{l=0}^{L} r_l + \xi \to 0$ *and* $n(\sum_{l=0}^{L} r_l + \xi) \to \infty$ *and the following two conditions hold for the prior* $\widetilde{\Pi}$ *in* (3.6) *and some* $q \in \mathcal{Q}^{\boldsymbol{MF}}$

1. $\widetilde{\Pi}\left(\mathcal{N}_{\sum_{l=0}^{L} r_l + \xi}\right) \geq exp(-C_4 n(\sum_{l=0}^{L} r_l + \xi))$
2. $d_{KL}(q, \pi) + n \sum_{\boldsymbol{z}} \int d_{KL}(P_0, P_{\boldsymbol{\theta}}) q(\boldsymbol{\theta}, \boldsymbol{z}) d\boldsymbol{\theta} \leq C_5 n(\sum_{l=0}^{L} r_l + \xi)$

where π *is the joint prior of* $(\boldsymbol{\theta}, \boldsymbol{z})$*,* q *is the joint variational distribution of* $(\boldsymbol{\theta}, \boldsymbol{z})$ *and* $\mathcal{N}_{\sum_{l=0}^{L} r_l + \xi}$ *is the KL neighborhood of radius* $\sum_{l=0}^{L} r_l + \xi$*.*

The following result shows that the variational posterior is consistent as long as Lemma 3.4.1, Lemma 3.4.2 and Lemma 3.4.3 hold. The proof of Theorem 3.4.4 demonstrates how the validity of these three lemmas imply variational posterior consistency.

Theorem 3.4.4. *Suppose Lemma 3.4.3 holds and Lemmas 3.4.1 and 3.4.2 hold for* $\epsilon_n = \sqrt{(\sum_{l=0}^{L} r_l + \xi) \sum_{l=0}^{L} u_l}$*. Then for some slowly increasing sequence* $M_n \to \infty$*,* $M_n \epsilon_n \to 0$ *and* $\widetilde{\Pi}^*$ *as in* (3.10)*,*

$$\widetilde{\Pi}^*(\mathcal{H}^c_{M_n \epsilon_n}) \to 0, \quad n \to \infty$$

in P_0^n *probability where* $\mathcal{H}^c_{M_n \epsilon_n} = \{\boldsymbol{\theta} : d_H(P_0, P_{\boldsymbol{\theta}}) \leq M_n \epsilon_n\}$ *is the Hellinger neighborhood of radius* $M_n \epsilon_n$*.*

Note, the above contraction rate depends mainly on two quantities r_l and ξ. Note r_l controls the number of nodes in the neural network. If the network is not sparse, then r_l is

$k_{l+1}(k_l+1)\vartheta_l/n$ instead of $s_l(k_l+1)\vartheta_l/n$ which can in turn make the convergence of $\epsilon_n \to 0$ difficult. On the other hand, if s_l and B_l are too small, it will cause ξ to explode since a good approximation to the true function may not exist in a very sparse space.

Remark (Rates as a function of n). *Let $L \sim O(\log n)$, $B_l^2 \sim O(k_l+1)$ and $s_l(k_l+1) = O(n^{1-2\varrho})$, for some $\varrho > 0$, then one can work with $\epsilon_n = n^{-\varrho}\log^3(n)$ as long as $\xi = O(n^{-2\varrho}\log^2(n))$. The exact expression of ϱ is determined by the degree of smoothness of the function η_0.*

Proof of Theorem 3.4.4 Discussion. To further enunciate Lemmas 3.4.1 and 3.4.2 consider the quantity $\mathcal{E}_{1n} = \int_{\mathcal{H}^c_{M_n\epsilon_n}} (P^n_{\boldsymbol{\theta}}/P^n_0)\widetilde{\pi}(\boldsymbol{\theta})d\boldsymbol{\theta}$ as used in the following proof. Here, $\mathcal{E}_{1n}$ can be split into two parts

$$\mathcal{E}_{1n} = \int_{\mathcal{H}^c_{M_n\epsilon_n}\cap\mathcal{F}(L,\boldsymbol{k},\boldsymbol{s}^\circ,\boldsymbol{B}^\circ)} (P^n_{\boldsymbol{\theta}}/P^n_0)\widetilde{\pi}(\boldsymbol{\theta})d\boldsymbol{\theta} + \int_{\mathcal{H}^c_{M_n\epsilon_n}\cap\mathcal{F}(L,\boldsymbol{k},\boldsymbol{s}^\circ,\boldsymbol{B}^\circ)^c} (P^n_{\boldsymbol{\theta}}/P^n_0)\widetilde{\pi}(\boldsymbol{\theta})d\boldsymbol{\theta}$$

Whereas Lemma 3.4.1 provides a handle on the first term by controlling the covering number of the sieve $\mathcal{F}(L,\boldsymbol{k},\boldsymbol{s}^\circ,\boldsymbol{B}^\circ)$, Lemma 3.4.2 gives a handle on the second term by controlling $\widetilde{\Pi}(\mathcal{F}(L,\boldsymbol{k},\boldsymbol{s}^\circ,\boldsymbol{B}^\circ)^c)$ (for more details we refer to Lemma A.2.6 in the Appendix A).

Next, consider the quantity $\mathcal{E}_{2n} = \log\int(P^n_{\boldsymbol{\theta}}/P^n_0)\widetilde{\pi}(\boldsymbol{\theta})d\boldsymbol{\theta}$ in the following proof. Lemma 3.4.3 part 1. provides a control on this term (see Lemma A.2.7 in the the Appendix A for more details). Finally, consider the quantity $\mathcal{E}_{3n} = d_{\mathrm{KL}}(q,\pi) + \sum_{\boldsymbol{z}}\int\log(P^n_0/P^n_{\boldsymbol{\theta}})q(\boldsymbol{\theta},\boldsymbol{z})d\boldsymbol{\theta}$ in the following proof. Indeed Lemma 3.4.3 part 2. provides a control on this term (see Lemma A.2.8 in the Appendix A for further details).

Proof. Let $\widetilde{\Pi}$ and $\widetilde{\Pi}^*$ be as in (3.7) and (3.10) respectively. Now,

$$\begin{aligned}
d_{\mathrm{KL}}(\widetilde{\pi}^*,\widetilde{\pi}(|\mathcal{D})) &= \int_{\mathcal{A}}\widetilde{\pi}^*(\boldsymbol{\theta})\log\frac{\widetilde{\pi}^*(\boldsymbol{\theta})}{\widetilde{\pi}(\boldsymbol{\theta}|\mathcal{D})}d\boldsymbol{\theta} + \int_{\mathcal{A}^c}\widetilde{\pi}^*(\boldsymbol{\theta})\log\frac{\widetilde{\pi}^*(\boldsymbol{\theta})}{\widetilde{\pi}(\boldsymbol{\theta}|\mathcal{D})}d\boldsymbol{\theta} \\
&= -\widetilde{\Pi}^*(\mathcal{A})\int_{\mathcal{A}}\frac{\widetilde{\pi}^*(\boldsymbol{\theta})}{\widetilde{\Pi}^*(\mathcal{A})}\log\frac{\widetilde{\pi}(\boldsymbol{\theta}|\mathcal{D})}{\widetilde{\pi}^*(\boldsymbol{\theta})}d\boldsymbol{\theta} - \widetilde{\Pi}^*(\mathcal{A}^c)\int_{\mathcal{A}^c}\frac{\widetilde{\pi}^*(\boldsymbol{\theta})}{\widetilde{\Pi}^*(\mathcal{A}^c)}\log\frac{\widetilde{\pi}(\boldsymbol{\theta}|\mathcal{D})}{\widetilde{\pi}^*(\boldsymbol{\theta})}d\boldsymbol{\theta} \\
&\geq \widetilde{\Pi}^*(\mathcal{A})\log\frac{\widetilde{\Pi}^*(\mathcal{A})}{\widetilde{\Pi}(\mathcal{A}|\mathcal{D})} + \widetilde{\Pi}^*(\mathcal{A}^c)\log\frac{\widetilde{\Pi}^*(\mathcal{A}^c)}{\widetilde{\Pi}(\mathcal{A}^c|\mathcal{D})}, \qquad \text{Jensen's inequality}
\end{aligned}$$

where the above lines hold for any set $\mathcal{A}$. Since $\widetilde{\Pi}(\mathcal{A}|\mathcal{D}) \leq 1$,

$$
\begin{aligned}
&\geq \widetilde{\Pi}^*(\mathcal{A}) \log \widetilde{\Pi}^*(\mathcal{A}) + \widetilde{\Pi}^*(\mathcal{A}^c) \log \widetilde{\Pi}^*(\mathcal{A}^c) - \widetilde{\Pi}^*(\mathcal{A}^c) \log \widetilde{\Pi}(\mathcal{A}^c|\mathcal{D}) \\
&\geq -\widetilde{\Pi}^*(\mathcal{A}^c) \log \widetilde{\Pi}(\mathcal{A}^c|\mathcal{D}) - \log 2, \quad (\because\ x \log x + (1-x)\log(1-x) \geq -\log 2) \\
&= -\widetilde{\Pi}^*(\mathcal{A}^c) \left(\underbrace{\log \int_{\mathcal{A}^c} (P_{\boldsymbol{\theta}}^n / P_0^n) \widetilde{\pi}(\boldsymbol{\theta}) d\boldsymbol{\theta}}_{\mathcal{E}_{1n}} - \underbrace{\log \int (P_{\boldsymbol{\theta}}^n / P_0^n) \widetilde{\pi}(\boldsymbol{\theta}) d\boldsymbol{\theta}}_{\mathcal{E}_{2n}} \right) - \log 2
\end{aligned}
$$

The above representation is similar to the proof of Theorems 3.1 and 3.2 in Bhattacharya and Maiti (2021). For any $q \in \mathcal{Q}^{\mathbf{MF}}$,

$$
\begin{aligned}
-\widetilde{\Pi}^*(\mathcal{A}^c)\mathcal{E}_{1n} &\leq d_{\mathrm{KL}}(\widetilde{\pi}^*, \widetilde{\pi}(|\mathcal{D})) - \widetilde{\Pi}^*(\mathcal{A}^c)\mathcal{E}_{2n} + \log 2 \\
&\leq d_{\mathrm{KL}}(\pi^*, \pi(|\mathcal{D})) - \widetilde{\Pi}^*(\mathcal{A}^c)\mathcal{E}_{2n} + \log 2 \qquad \text{by Lemma A.2.3} \\
&\leq d_{\mathrm{KL}}(q, \pi(|\mathcal{D})) - \widetilde{\Pi}^*(\mathcal{A}^c)\mathcal{E}_{2n} + \log 2 \qquad \pi^* \text{ is the KL minimizer} \\
&\leq \underbrace{d_{\mathrm{KL}}(q, \pi) + \sum_{\boldsymbol{z}} \int \log \frac{P_0^n}{P_{\boldsymbol{\theta}}^n} q(\boldsymbol{\theta}, \boldsymbol{z}) d\boldsymbol{\theta}}_{\mathcal{E}_{3n}} + (1 - \widetilde{\Pi}^*(\mathcal{A}^c))\mathcal{E}_{2n} + \log 2 \\
&= \mathcal{E}_{3n} + (1 - \widetilde{\Pi}^*(\mathcal{A}^c))\mathcal{E}_{2n} + \log 2 \qquad (3.13)
\end{aligned}
$$

where the fourth inequality in the above equation follows since

$$
\begin{aligned}
&d_{\mathrm{KL}}(q, \pi(|\mathcal{D})) \\
&= \sum_{\boldsymbol{z}} \int (\log q(\boldsymbol{\theta}, \boldsymbol{z}) - \log P_{\boldsymbol{\theta}}^n - \log \pi(\boldsymbol{\theta}, \boldsymbol{z}) + \log m(\mathcal{D})) q(\boldsymbol{\theta}, \boldsymbol{z}) d\boldsymbol{\theta} \\
&= \underbrace{\sum_{\boldsymbol{z}} \int (\log q(\boldsymbol{\theta}, \boldsymbol{z}) - \log \pi(\boldsymbol{\theta}, \boldsymbol{z})) q(\boldsymbol{\theta}, \boldsymbol{z}) d\boldsymbol{\theta}}_{d_{\mathrm{KL}}(q, \pi)} + \sum_{\boldsymbol{z}} \int (\log P_0^n - \log P_{\boldsymbol{\theta}}^n) q(\boldsymbol{\theta}, \boldsymbol{z}) d\boldsymbol{\theta} \\
&+ \underbrace{\log m(\mathcal{D}) - \log P_0^n}_{\mathcal{E}_{2n}}
\end{aligned}
$$

where $m(\mathcal{D})$ is the marginal distribution of data as in (3.5).

Take $\mathcal{A} = \mathcal{H}^c_{M_n \epsilon_n} = \{\boldsymbol{\theta} : d_{\mathrm{H}}(P_0, P_{\boldsymbol{\theta}}) > M_n \epsilon_n\}$

If Lemma 3.4.1 and Lemma 3.4.2 hold, then by Lemma A.2.6, $\mathcal{E}_{1n} \leq -nCM_n^2\epsilon_n^2 / \sum u_l$ for any $M_n \to \infty$ with high probability.

If Lemma 3.4.3 condition 1 holds, then by Lemma A.2.7, $\mathcal{E}_{2n} \leq nM_n(\sum_{l=0}^{L} r_l + \xi)$ for any $M_n \to \infty$ with high probability.

If Lemma 3.4.3 condition 2 holds, then by Lemma A.2.8, $\mathcal{E}_{3n} \leq nM_n(\sum_{l=0}^{L} r_l + \xi)$ for any $M_n \to \infty$ with high probability.

Therefore, by (3.13), we get

$$\begin{aligned}
\frac{nCM_n^2\epsilon_n^2}{\sum u_l}\widetilde{\Pi}^*\left(\mathcal{H}^c_{M_n\epsilon_n}\right) &\leq nM_n(\sum_{l=0}^{L} r_l + \xi) + nM_n(\sum_{l=0}^{L} r_l + \xi) + \log 2 \\
&\leq nM_n(\sum_{l=0}^{L} r_l + \xi) + nM_n(\sum_{l=0}^{L} r_l + \xi) + M_n(\sum_{l=0}^{L} r_l + \xi) \\
\implies \widetilde{\Pi}^*\left(\mathcal{H}^c_{M_n\epsilon_n}\right) &\leq \frac{3M_n(\sum_{l=0}^{L} r_l + \xi)\sum u_l}{C_1 M_n^2\epsilon_n^2}
\end{aligned}$$

Taking $\epsilon_n = \sqrt{\sum_{l=0}^{L}(r_l + \xi)\sum u_l}$ and noting $M_n \to \infty$, the proof follows. □

We next give conditions on the prior probabilities λ_l and σ_0 to guarantee that Lemmas 3.4.1, 3.4.2 and 3.4.3 hold. This in turn implies the conditions of Theorem 3.4.4 hold and variational posterior is consistent.

Corollary 3.4.5. *Let $\sigma_0^2 = 1$, $-\log\lambda_l = \log(k_{l+1}) + C_l(k_l+1)\vartheta_l$, then conditions of Theorem 3.4.4 hold and $\widetilde{\Pi}^*$ as in* (3.10) *satisfies*

$$\widetilde{\Pi}^*(\mathcal{H}^c_{M_n\epsilon_n}) \to 0, \quad n \to \infty$$

in P_0^n probability where and $\mathcal{H}_{M_n\epsilon_n} = \{\boldsymbol{\theta} : d_H(P_0, P_{\boldsymbol{\theta}}) \leq M_n\epsilon_n\}$ is the Hellinger neighborhood of radius $M_n\epsilon_n$.

The proof of the corollary has been provided in Appendix A. In this corollary, note that our expression of prior inclusion probability varies as a function of l thereby providing a handle on layer-wise sparsity. Indeed, using these expressions in numerical studies further substantiates the theoretical framework developed in this section.

Remark (Optimal Contraction). *For a fixed choice of $\boldsymbol{k}$, the optimal contraction rate is achieved at $\boldsymbol{s}^\star, \boldsymbol{B}^\star = \underset{\boldsymbol{s},\boldsymbol{B}}{argmin}(\sum r_l + \xi)$. Thus, $\boldsymbol{s}^\star$ and $\boldsymbol{B}^\star$ are the optimal values of $\boldsymbol{s}$ and $\boldsymbol{B}$*

which give the best sparse network with minimal loss in the true accuracy. The corresponding probability expressions in Corollary 3.4.5 can be accordingly modified by setting $\boldsymbol{s} = \boldsymbol{s}^\star$ *and* $\boldsymbol{B} = \boldsymbol{B}^\star$ *in the expressions of* ϑ_l *and* r_l *in* (3.12).

3.5 Numerical Experiments

In this section, we present several numerical experiments to demonstrate the performance of our spike-and-slab independent Gaussian (SS-IG) Bayesian neural networks which we implement in PyTorch (Paszke et al., 2019). Further, to evaluate the efficacy of the variational inference we benchmark our model on synthetic as well as real datasets. Our numerical investigation justifies the use of proposed choices of prior hyperparameters specifically layer-wise prior inclusion probabilities, which in turn substantiates the significance of our theoretical developments. With fully Bayesian treatment, we are also able to quantify the uncertainties for the parameter estimates and variational inference helps to scale our model to large network architectures as well as complex datasets.

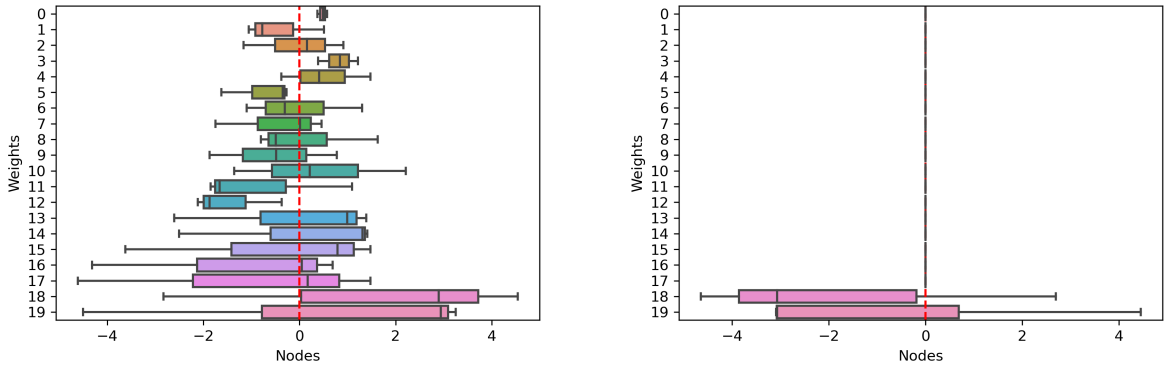
We compare our sparse model with a node selection technique: horseshoe BNN (HS-BNN) (Ghosh et al., 2019) and an edge selection technique: spike-and-slab BNN (SV-BNN) (Bai et al., 2020) in the second simulation study and UCI regression dataset examples. We use optimal choices of prior parameters and fine tuning parameters provided by the authors of HS-BNN and SV-BNN in their respective models. Further we compare our model against dense variational BNN model (VBNN) (Blundell et al., 2015) in all of the experiments. Since it has no sparse structure, it serves as a baseline allowing to check whether sparsity compromises accuracy. In all the experiments, we fix $\sigma_0^2 = 1$ and $\sigma_e^2 = 1$. For our model, the choices of layer-wise λ_l follow from Corollary 3.4.5: $\lambda_l = (1/k_{l+1})\exp(-C_l(k_l+1)\vartheta_l)$. We take C_l values in the negative order of 10 such that prior inclusion probabilities do not fall below 10^{-50} otherwise λ_l values close to 0 might prune away all the nodes from a layer (check appendix B for more discussion). The remaining tuning parameter details such as learning rate, minibatch size, and initial parameter choice are provided in the appendix B.

The prediction accuracy is calculated using variational Bayes posterior mean estimator with 30 Monte Carlo samples in testing phase.

Node sparsity estimates. In our experiments, we provide node sparsity estimates for each hidden layer separately. For all models, the node sparsity in a given hidden layer is the ratio of number of neurons with atleast one nonzero incoming edge over the original number of neurons present in that layer before training. The layer-wise node sparsity estimates give clear picture of the structural compactness of the trained model during test time. The structurally compact trained model has lower latency during inference stage.

3.5.1 Simulation Study - I

We consider a two dimensional regression problem where the true response y_0 is generated by sampling X from $U([-1,1]^2)$ and feeding it to a deep neural network with known parameters. We add a random Gaussian noise with $\sigma = 5\%\sqrt{Var(y_0)}$ to y_0 to get noisy outputs y. We create the dataset using a shallow neural network consisting of 2 inputs, one hidden layer with 2 nodes and 1 output (2-2-1 network). We train our SS-IG model and VBNN model using a single hidden layer network with 20 neurons in the hidden layer and administer sigmoid activation. Each model is trained till convergence. We found that both models give competitive predictive performance while fitting the given data. In Figure 3.2 we plot the magnitudes of the incoming weights into the hidden layer nodes using boxplots. Our model with the help of spike and slab prior is able to prune away redundant nodes not required for fitting the model. Since VBNN is densely connected, it shows all the nodes being active in its final model. From this experiment, it is clear that neural networks can be pruned leading to more compact models at inference stage without compromising the accuracy. We also performed the same experiment with a wider neural network consisting of 100 nodes in the single hidden layer and provide the results in the appendix B. There again we show that our model can easily recover the sparse solution with competitive performance.



(a) VBNN (b) SS-IG

Figure 3.2 **Simulation study I results.** Node-wise weight magnitudes recovered by VBNN and proposed SS-IG model in the synthetic regression data generated using 2-2-1 network. The boxplots show the distribution of incoming weights into a given hidden layer node.

3.5.2 Simulation Study - II

We consider a nonlinear regression example and generate the data from the following model:

$$y = \frac{7x_2}{1+x_1^2} + \sin(x_3 x_4) + 2x_5 + \varepsilon,$$

where $\varepsilon \sim N(0,1)$. Further all the covariates are i.i.d. $N(0,1)$ and independent of ε. We generated 3000 data entries to create the training data for the experiment. Additional 1000 observations were generated for testing. We modeled this data using 2-hidden layer neural network which consists of 20 neurons per hidden layer. Sigmoid activation function is administered for each model used for comparative analysis. Table 3.1 provides the RMSEs on train and test dataset as well as layer-wise node sparsity estimates for SS-IG, SV-BNN, HS-BNN, and VBNN models. Our model is extremely well at pruning redundant nodes which leads to the most compact model compared to the other sparse models: SV-BNN and HS-BNN. Moreover it exhibits lower root mean squared error (RMSE) values on test data among the sparse models while showing similar predictive performance compared to the densely connected VBNN. This experiment further underscores the major benefit of

Table 3.1 **Simulation study II results.** Performance of the proposed SS-IG, SV-BNN, HS-BNN, and VBNN models where each model was trained for 10k epochs with learning rate 5×10^{-3}. Mean and S.D. of RMSE values and median sparsity estimates were calculated from last 1000 epochs (with jump of 10 giving us sample of 100). The sparsity estimates are given as a tuple of 2 values representing layer-1 and layer-2 node sparsities.

Model	Train RMSE	Test RMSE	Sparsity Estimate
SS-IG	1.2087±0.0490	1.1947±0.0587	(0.35,0.05)
SV-BNN	1.2897±0.0323	1.2760±0.0363	(0.45,0.35)
HS-BNN	1.2580±0.0305	1.2436±0.0394	(1.00, 1.00)
VBNN	1.1661±0.0335	1.1614±0.0349	NA

our proposed approach to generate very compact models which could reduce computational times and memory usage at inference stage.

3.5.3 UCI Regression Datasets

We apply our model to traditional UCI regression datasets (Dua and Graff, 2017) and contrast our performance against SV-BNN, HS-BNN, and VBNN models. We follow the protocol proposed by (Hernandez-Lobato and Adams, 2015) and train a single layer neural network with sigmoid activations. For smaller datasets - *Concrete, Wine, Power Plant, Kin8nm*, we take 50 nodes in the hidden layer, while for larger datasets - *Protein, Year*, we take 100 nodes in the hidden layer. We spilt data randomly while maintaining 9:1 train-test ratio in each case and for smaller datasets we repeat this technique 20 times. In *Protein* data we perform 5 repetitions while in *Year* data we use a single random split (more details in the appendix B). For the comparative analysis, we benchmark against SV-BNN, HS-BNN and VBNN. Moreover, VBNN test RMSEs serve as baseline in each dataset. Table 3.2 summarises our results including the sparsity estimate representing hidden layer-1 node sparsity (since there is only one hidden layer in the networks considered).

We achieve lower RMSEs compared to SV-BNN and HS-BNN in *Power Plant, Kin8nm,* and *Year* datasets and in other cases we achieve comparable RMSE values. In all the

Table 3.2 **UCI regression datasets results.**

		Test RMSE				Sparsity Estimate	
Dataset	$n(k_0)$	SS-IG	SV-BNN	HS-BNN	VBNN	SS-IG	SV-BNN
Concrete	1030 (8)	7.92±0.68	8.22±0.70	5.34±0.53	7.34±0.62	0.42±0.06	0.98±0.02
Wine	1599 (11)	0.66±0.05	0.65±0.05	0.66±0.05	0.64±0.05	0.18±0.05	0.87±0.04
Power Plant	9568 (4)	4.28±0.20	4.32±0.19	4.34±0.18	4.27±0.17	0.18±0.03	0.24±0.03
Kin8nm	8192 (8)	0.09±0.00	0.11±0.01	0.10±0.00	0.09±0.00	0.43±0.04	0.47±0.04
Protein	45730 (9)	4.85±0.05	4.93±0.06	4.59±0.02	4.78±0.06	0.81±0.03	0.93±0.03
Year	515345 (90)	8.68±NA	8.78±NA	9.33±NA	8.67±NA	0.71±NA	0.78±NA

datasets, our predictive performance is close to the dense baseline of VBNN. We provide node sparsity estimates in our SS-IG and SV-BNN models. HS-BNN was not able to achieve sparse structure which is consistent with the results provided in the appendix of (Ghosh et al., 2019). In contrast to HS-BNN, our model sparsifies the model during training without requiring ad-hoc pruning rule. Table 3.2 demonstrates that our approach uniformly achieves better sparsity than SV-BNN. In particular, *Concrete* and *Wine* datasets show the high compressive ability of our model over SV-BNN leading to very compact models for inference.

3.5.4 Image Classification Datasets

Here, we benchmark the empirical performance of our proposed SS-IG method on network architectures and image classification datasets used in practice.

Baselines. We compare our model against VBNN model which serves as a dense baseline to gauge the trade-off between predictive performance and sparsity. Moreover, to highlight the complementary behavior in memory and computational efficiency of node selection compared to edge selection achieved via Bayesian spike-and-slab prior framework, we compare our model against the edge selection model, SV-BNN.

Network architectures. We consider 2 neural network model architectures: (i) multi-layer perceptron (MLP), and (ii) LeNet-5-Caffe. In MLP model, we take 2 hidden layers

with 400 neurons in each layer. Output layer has 10 neurons since there are 10 classes in both datasets. Next, LeNet-5-Caffe model has 2 convolutional layers with 20 and 50 feature maps respectively with filter size 5×5 for both layers. In SS-IG model, for convolution layers, we prune output channels (similar to neurons in linear layers) using our spike-and-slab prior where each output channel is assigned an Bernoulli variable to collectively prune parameters incident on that channel. We apply 2×2 max pooling layer after each convolution layer. The flattened feature layer after second convolution layer has size $4 * 4 * 50 = 800$ serving as input to the fully connected block, where there are 2 hidden layers with 800 and 500 neurons respectively. The output layer has 10 neurons.

Datasets. We apply each network architecture on 2 image classification datasets: (i) MNIST: dataset of 60,000 small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9, and (ii) Fashion-MNIST (Xiao et al., 2017): dataset of 60,000 small square 28×28 pixel grayscale images of items of 10 types of clothing. We preprocess the images in the MNIST data by dividing their pixel values by 126. In Fashion-MNIST data, we horizontally flip images at random with probability of 0.5.

Metrics. We quantify the predictive performance using the accuracy of the test data (MNIST and Fashion-MNIST). Besides the test accuracy, we evaluate our model against SV-BNN using the metrics that relate to the model compression and computational complexity. First the *compression ratio* is the ratio of number of nonzero weights in the compressed network versus the dense model and is an indicator of storage cost at test-time. Next, we present layer-wise node sparsities in MLP experiments to highlight the computational speedups at test-time. In LeNet-5-Caffe experiments, we provide the *floating point operations (FLOPs) ratio* which is the ratio of number of FLOPs required to predict y from x during test time in the compressed network versus its dense counterpart. We have detailed the FLOPs calculation in neural networks in Appendix B.

Nonlinear activation. We use swish activations (Elfwing et al., 2018; Ramachandran et al.,

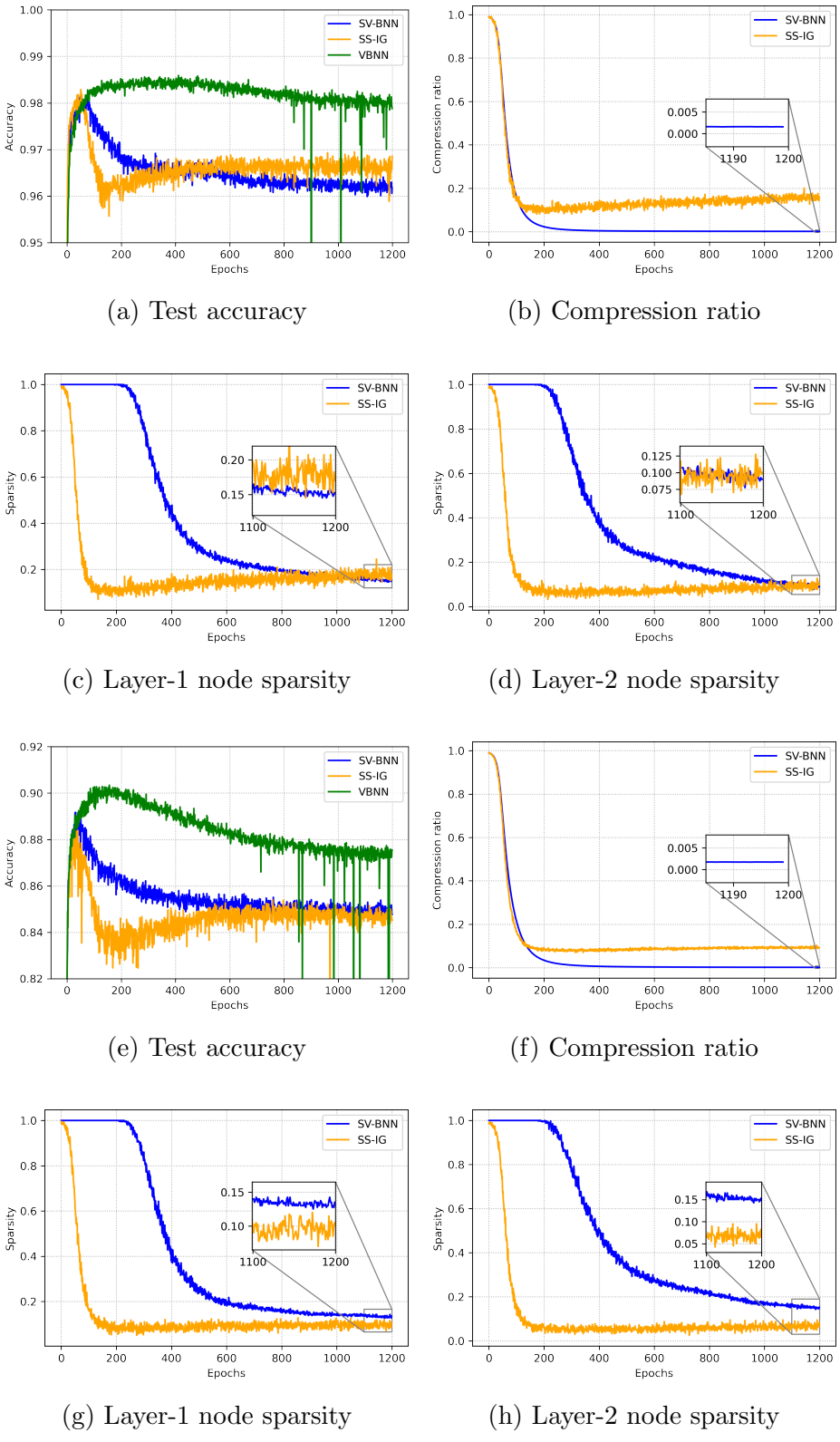
2017) instead of ReLUs in our proposed SS-IG model to avoid the dying neuron problem (Lu et al., 2020). Specifically in large scale datasets turning off a node with more than 100 incoming edges adversely impacts the training process of ReLU networks. Smoother activation functions such as sigmoid, tanh, swish etc help alleviate this problem. We choose swish since it has the best performance. For VBNN and SV-BNN, we use ReLU activations as recommended by their authors.

MLP Experiments

The results of MLP network experiments on MNIST and Fashion-MNIST are presented in Figure 3.3. We provide test data accuracy, model compression ratio, and layer-wise node sparsities in each experiment.

In MLP/MNIST experiment (Figure 3.3a - 3.3d), we observe that VBNN and SS-IG models only require ~ 400 epochs to achieve stable predictive performance (Figure 3.3a). In contrast, SV-BNN slightly degrades after 600 epochs and takes longer to achieve convergence in layer-wise node sparsities compared to our approach (Figure 3.3c and 3.3d). Moreover, for SS-IG model, we observe that as we start to learn sparse network our model shows peak test accuracy when most of the nodes are present in the model and it starts to drop as we learn sparser network and ultimately the test accuracy stabilizes when the node sparsities converge. Furthermore, SV-BNN has better model compression ratio (Figure 3.3b) in this experiment at the expense of lower predictive performance. Our method is prunes off $\sim 80\%$ of first hidden layer nodes and $\sim 90\%$ of second hidden layer nodes at the expense of $\sim 2\%$ accuracy loss due to sparsification compared to the dense VBNN.

In MLP/Fashion-MNIST experiment (Figure 3.3e - 3.3h), we observe that VBNN model takes $\sim$ 200 epochs and our model takes $\sim$ 600 epochs for convergence. SV-BNN model takes longer to achieve convergence in layer-wise node sparsities (Figure 3.3g and 3.3h). We also observe the complementary behavior of our model and SV-BNN in memory and computational efficiency where our model achieves better layer-wise node sparsities and SV-



(a) Test accuracy (b) Compression ratio

(c) Layer-1 node sparsity (d) Layer-2 node sparsity

(e) Test accuracy (f) Compression ratio

(g) Layer-1 node sparsity (h) Layer-2 node sparsity

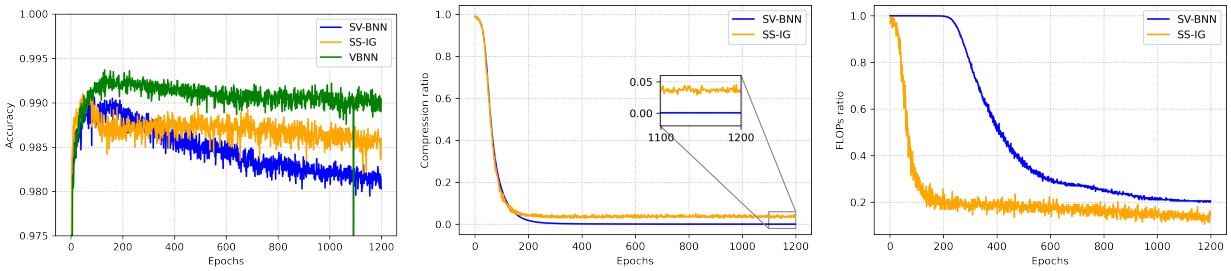
Figure 3.3 **MLP/MNIST and MLP/Fashion-MNIST experiments results.** First two rows (a)-(d) represent the MLP on MNIST experiment results. Bottom two rows (e)-(h) represent the MLP on Fashion-MNIST experiment results.

BNN has better model compression ratio (Figure 3.3f) with both models having similar predictive performance (Figure 3.3e). Furthermore, our method prunes off $\sim 90\%$ of first hidden layer nodes and $\sim 92\%$ of second hidden layer nodes at the expense of $\sim 3\%$ accuracy loss due to sparsification compared to the densely connected VBNN.

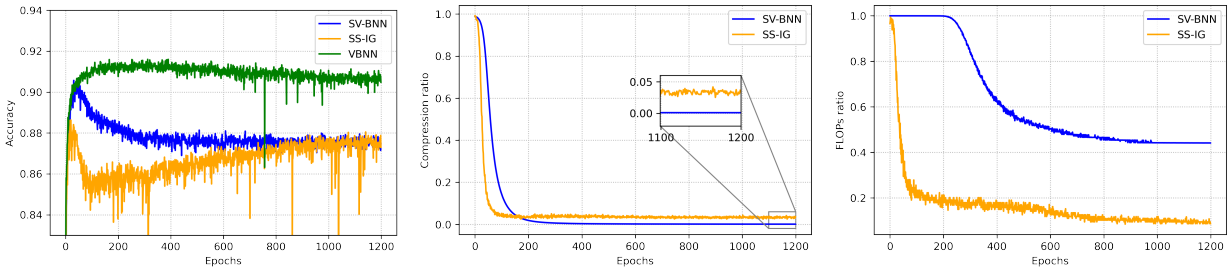
LeNet-5-Caffe Experiments

The results of more complex LeNet-5-Caffe network experiments on MNIST and Fashion-MNIST are presented in Figure 3.4. We provide test data accuracy, model compression ratio, and FLOPs ratio in each experiment over 1200 epochs. Here, FLOPs ratio serve as a collective indicator of layer-wise node sparsities since FLOPs are directly related to how many neurons or channels are remaining in linear or convolution layers respectively.

In LeNet-5-Caffe/MNIST experiment (Figure 3.4a - 3.4c), we observe that our model has better predictive accuracy than SV-BNN (Figure 3.4a). Moreover, we achieve 10% more



(a) Test accuracy (b) Compression ratio (c) FLOPs ratio



(d) Test accuracy (e) Compression ratio (f) FLOPs ratio

Figure 3.4 **LeNet-5-Caffe/MNIST and LeNet-5-Caffe/Fashion-MNIST experiments results.** Top row (a)-(c) represent the LeNet-5-Caffe/MNIST experiment results. Bottom row (d)-(f) represent the LeNet-5-Caffe/Fashion-MNIST experiment results.

reduction in Flops (Figure 3.4c)) compared to SV-BNN whereas SV-BNN achieves better model compression than our approach (Figure 3.4b). Lastly, our method is able to reduce the FLOPs of the model during inference at test-time by 90% at the expense of $\sim 0.5\%$ accuracy loss due to sparsification compared to the densely connected VBNN.

In LeNet-5-Caffe/Fashion-MNIST experiment (Figure 3.4d - 3.4f), we observe that both SS-IG and SV-BNN have similar test accuracies at convergence (Figure 3.4d). However, our model has 40% less FLOPs (Figure 3.4f) during inference stage compared to SV-BNN which again achieves better model compression (Figure 3.4e). This highlights the complementary nature of our method of node selection that leads to a structurally sparse model with significantly lower (almost 5 times) FLOPs compared to weight pruning approach, SV-BNN, which induces unstructured sparsity in the pruned network leading to significant model compression with low storage cost. Lastly, our method leads to a sparse model with only 8% of the FLOPs as compared to VBNN at the expense of $\sim 3\%$ accuracy loss underscoring the trade-off between predictive accuracy and sparsity.

3.6 Conclusion and Discussion

In this chapter, we have proposed sparse deep Bayesian neural networks using spike-and-slab priors for optimal node recovery. Our method incorporates layer-wise prior inclusion probabilities and recovers underlying structurally sparse model effectively. Our theoretical developments highlight the conditions required for the posterior consistency of the variational posterior to hold. With layer-wise characterisation of prior inclusion probabilities we show that the proposed sparse BNN approximations can achieve predictive performance comparable to dense networks. Our results relax the constraints of equal number of nodes and uniform bounds on weights thereby achieving optimal node recovery on more generic neural network structure. The closeness of a true function to the topology induced by layer-wise node distribution depends on the degree of smoothness of the true underlying function. In this work, this has not been studied in depth and forms a future direction for work.

Note, in contrast to previous works, our work assumes a spike-and-slab prior on the entire vector of incoming weights and bias onto a node. We underscore the fact that node selection has complementary behavior with edge selection approaches as established by our empirical experiments. Node selection offers significant computational speedup whereas edge selection achieves significant model compression at test-time. The demonstration of the efficacy of our node selection approach opens the avenue for exploration of sophisticated group sparsity priors for node selection. Our detailed experiments show the subnetwork selection ability of our method which underscores the notion that deep neural networks can be heavily pruned without losing predictive performance. The experiment with convolution neural network (LeNet-5-Caffe) highlights the generalizability of our approach from mere multi layer perceptron to complex deep learning models. Although our method performs model reduction while maintaining predictive power, some further improvements may be obtained by choosing the number of layers in a data-driven fashion and can be a part of future work.

APPENDICES

APPENDIX A

PROOFS OF SS-IG THEORETICAL RESULTS

A.1 Definitions

Definition A.1.1 (Sieve)**.** *Consider a sequence of function classes* $\mathcal{F}_1 \subseteq \mathcal{F}_2 \subseteq \cdots \subseteq \mathcal{F}_n \subseteq \mathcal{F}_{n+1} \subseteq \cdots \subseteq \mathcal{F}$*, where* $\forall f \in \mathcal{F}$, $\exists\, f_n \in \mathcal{F}_n$ *s.t.* $d(f, f_n) \to 0$ *as* $n \to \infty$ *where* $d(.,.)$ *is some pseudo-metric on* $\mathcal{F}$*. More precisely,* $\cup_{n=1}^{\infty}\mathcal{F}_n$ *is dense in* $\mathcal{F}$*. Then,* $\mathcal{F}_n$ *is called a sieve space of* $\mathcal{F}$ *with respect to the pseudo-metric* $d(.,.)$*, and the sequence* $\{f_n\}$ *is called a sieve (Grenander, 1981).*

Definition A.1.2 (Covering number)**.** *Let* $(V, ||.||)$ *be a normed space, and* $\mathcal{F} \subset V$*. Then,* $\{V_1, \cdots, V_N\}$ *is an* $\varepsilon-$*covering of* $\mathcal{F}$ *if* $\mathcal{F} \subset \cup_{i=1}^{N} B(V_i, \varepsilon)$*, or equivalently,* $\forall\ \varrho \in \mathcal{F}$, $\exists\ i$ *such that* $||\varrho - V_i|| < \varepsilon$*. The covering number of* $\mathcal{F}$ *denoted by* $N(\varepsilon, \mathcal{F}, ||.||) = \min\{n : \exists\, \varepsilon-$ *covering over* $\mathcal{F}$ *of size* $n\}$ *(Pollard, 1991).*

A.2 General Lemmas

Lemma A.2.1. *Let* g_1 *and* g_2 *be any two density functions. Then*

$$E_{g_1}(|\log(g_1/g_2)|) \leq d_{KL}(g_1, g_2) + 2/e$$

Proof. Refer to Lemma 4 in Lee (2000). □

Lemma A.2.2. *For any* $K > 0$*, let* $\boldsymbol{a}, \boldsymbol{a^0} \in [0,1]^K$ *such that* $\sum_{k=1}^{K} a_k = \sum_{k=1}^{K} a_k^0 = 1$*, then the KL divergence between mixture densities* $\sum_{k=1}^{K} a_k g_k$ *and* $\sum_{k=1}^{K} a_k^0 g_k^0$ *is bounded as*

$$d_{KL}\left(\sum_{k=1}^{K} a_k^0 g_k^0, \sum_{k=1}^{K} a_k g_k\right) \leq d_{KL}(\boldsymbol{a^0}, \boldsymbol{a}) + \sum_{k=1}^{K} a_k^0 d_{KL}(g_k^0, g_k)$$

Proof. Refer to Lemma 6.1 in Chérief-Abdellatif and Alquier (2018). □

Lemma A.2.3.

$$d_{KL}(\widetilde{\pi}^*, \widetilde{\pi}(|\mathcal{D})) \leq d_{KL}(\pi^*, \pi(|\mathcal{D}))$$

Proof. Using Lemma A.2.2 with $\boldsymbol{a}^0 = \pi^*(\boldsymbol{z})$, $\boldsymbol{a} = \pi(\boldsymbol{z}|\mathcal{D})$, $g^0 = \pi^*(\boldsymbol{\theta}|\boldsymbol{z})$ and $g = \pi(\boldsymbol{\theta}|\boldsymbol{z}, \mathcal{D})$, we get

$$\begin{aligned} d_{\mathrm{KL}}(\widetilde{\pi}^*, \widetilde{\pi}(|\mathcal{D})) &= d_{\mathrm{KL}}(\sum_{\boldsymbol{z}} \pi^*(\boldsymbol{\theta}|\boldsymbol{z})\pi^*(\boldsymbol{z}), \sum_{\boldsymbol{z}} \pi(\boldsymbol{\theta}|\boldsymbol{z}, \mathcal{D})\pi(\boldsymbol{z}|\mathcal{D})) \\ &\leq d_{\mathrm{KL}}(\pi^*(\boldsymbol{z}), \pi(\boldsymbol{z}|\mathcal{D})) + \sum_{\boldsymbol{z}} d_{\mathrm{KL}}(\pi^*(\boldsymbol{\theta}|\boldsymbol{z}), \pi(\boldsymbol{\theta}|\boldsymbol{z}, \mathcal{D}))\pi^*(\boldsymbol{z}) \\ &= d_{\mathrm{KL}}(\pi^*(\boldsymbol{\theta}, \boldsymbol{z}), \pi(\boldsymbol{\theta}, \boldsymbol{z}|\mathcal{D})) = d_{\mathrm{KL}}(\pi^*, \pi(|\mathcal{D})) \end{aligned}$$

□

Lemma A.2.4. *For any 1-Lipschitz continuous activation function ψ such that $\psi(x) \leq x \; \forall x \geq 0$,*

$$N(\delta, \mathcal{F}(L, \boldsymbol{k}, \boldsymbol{s}, \boldsymbol{B}), ||.||_\infty) \leq \sum_{s_L^* \leq s_L} \cdots \sum_{s_0^* \leq s_0} \left[\prod_{l=0}^{L} \left(\frac{B_l}{\delta B_l / (2(L+1)(\prod_{j=0}^{L} B_j))} k_{l+1} \right)^{s_l} \right]$$

where N denotes the covering number.

Proof. Given a neural network

$$\eta(\boldsymbol{x}) = \boldsymbol{v}_L + \boldsymbol{W}_L \psi(\boldsymbol{v}_{L-1} + \boldsymbol{W}_{L-1}\psi(\boldsymbol{v}_{L-2} + \boldsymbol{W}_{L-2}\psi(\cdots \psi(\boldsymbol{v}_1 + \boldsymbol{W}_1\psi(\boldsymbol{v}_0 + \boldsymbol{W}_0\boldsymbol{x}))))$$

for $l \in \{1, \cdots, L\}$, we define $A_l^+\eta : [0,1]^p \to \mathbb{R}^{k_l}$,

$$A_l^+\eta(\boldsymbol{x}) = \psi(\boldsymbol{v}_{l-1} + \boldsymbol{W}_{l-1}\psi(\boldsymbol{v}_{l-2} + \boldsymbol{W}_{l-2}\psi(\cdots \psi(\boldsymbol{v}_1 + \boldsymbol{W}_1\psi(\boldsymbol{v}_0 + \boldsymbol{W}_0\boldsymbol{x}))))$$

and $A_l^-\eta : \mathbb{R}^{k_{l-1}} \to \mathbb{R}^{k_{L+1}}$,

$$A_l^-\eta(\boldsymbol{y}) = \boldsymbol{v}_L + \boldsymbol{W}_L \psi(\boldsymbol{v}_{L-1} + \boldsymbol{W}_{L-1}\psi(\cdots \psi(\boldsymbol{v}_l + \boldsymbol{W}_l\psi(\boldsymbol{v}_{l-1} + \boldsymbol{W}_{l-1}\boldsymbol{y}))))$$

The above framework is also used in the proof of lemma 5 in Schmidt-Hieber (2020). Next, set $A_0^+\eta(\boldsymbol{x}) = A_{L+2}^-\eta(\boldsymbol{x}) = \boldsymbol{x}$ and further note that for $\eta \in \mathcal{F}(L, \boldsymbol{k})$, $|A_l^+\eta(\boldsymbol{x})|_\infty \leq \prod_{j=0}^{l-1} B_j$

where $\boldsymbol{k} = (p, k_1, \cdots, k_L, k_{L+1})$ and $k_{L+1} = 1$. Next, we derive upper bound on Lipschitz constant of $A_l^- \eta$.

$$|\boldsymbol{W}_L A_L^+ \eta(\boldsymbol{x}_1) - \boldsymbol{W}_L A_L^+ \eta(\boldsymbol{x}_2)|_\infty = |A_l^- \eta(A_{l-1}^+ \eta(\boldsymbol{x}_1)) - A_l^- \eta(A_{l-1}^+ \eta(\boldsymbol{x}_2))|_\infty \tag{A.1}$$

l.h.s. is bounded above by $\prod_{j=0}^{L} B_j$ and r.h.s consists of composition of Lipschitz functions $A_l^- \eta$ and $A_{l-1}^+ \eta$ with C_1 and C_2 being corresponding Lipschitz constants. So we can bound r.h.s. by,

$$|A_l^- \eta(A_{l-1}^+ \eta(\boldsymbol{x}_1)) - A_l^- \eta(A_{l-1}^+ \eta(\boldsymbol{x}_2))|_\infty \leq C_1 C_2 ||\boldsymbol{x}_1 - \boldsymbol{x}_2||_\infty \quad \forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^p$$

If we choose $\boldsymbol{x}_1 = \boldsymbol{x} \in [0,1]^p$ and $\boldsymbol{x}_2 = \boldsymbol{0}$ then,

$$|A_l^- \eta(A_{l-1}^+ \eta(\boldsymbol{x})) - A_l^- \eta(A_{l-1}^+ \eta(\boldsymbol{0}))|_\infty \leq C_1 C_2 \quad \forall \boldsymbol{x} \in [0,1]^p$$

Since C_2 is Lipschitz constant for $A_{l-1}^+ \eta$ and we know that $|A_{l-1}^+ \eta|_\infty \leq \prod_{j=0}^{l-2} B_j$. So we get $C_2 \leq 2 \prod_{j=0}^{l-2} B_j$. We use this in above expression,

$$|A_l^- \eta(A_{l-1}^+ \eta(\boldsymbol{x})) - A_l^- \eta(A_{l-1}^+ \eta(\boldsymbol{0}))|_\infty \leq 2C_1 \prod_{j=0}^{l-2} B_j \quad \forall \boldsymbol{x} \in [0,1]^p \tag{A.2}$$

Next we know that l.h.s. of (A.2) can be bounded above by $2 \prod_{j=0}^{L} B_j$ because of (A.1). So we get bound on Lipschitz constant of $A_l^- \eta$,

$$2C_1 \prod_{j=0}^{l-2} B_j \leq 2 \prod_{j=0}^{L} B_j \implies C_1 \leq \prod_{j=l-1}^{L} B_j$$

Let $\eta, \eta^* \in \mathcal{F}(L, \boldsymbol{k}, \boldsymbol{s}, \boldsymbol{B})$ be two neural networks with $\overline{\boldsymbol{W}}_l = (\boldsymbol{v}_l, \boldsymbol{W}_l)$ and $\overline{\boldsymbol{W}}_l^* = (\boldsymbol{v}_l^*, \boldsymbol{W}_l^*)$ respectively. Here, we define $\overline{\boldsymbol{\delta}}_l$ using the L_1 norms of the rows of $\overline{\boldsymbol{D}}_l = \overline{\boldsymbol{W}}_l - \overline{\boldsymbol{W}}_l^*$ as follows

$$\overline{\boldsymbol{D}}_l = (\overline{\boldsymbol{d}}_{l1}^\top, \cdots, \overline{\boldsymbol{d}}_{lk_{l+1}}^\top)^\top \quad \overline{\boldsymbol{\delta}}_l = (||\overline{\boldsymbol{d}}_{l1}||_1, \cdots, ||\overline{\boldsymbol{d}}_{lk_{l+1}}||_1)$$

We choose η, η^* such that $||\overline{\boldsymbol{\delta}}_l||_\infty \leq \zeta B_l$. This also means that all parameters in each layer of these two networks are at most ζB_l distance away from each other. Then, we can bound the absolute difference between these two neural networks by,

$$|\eta(\boldsymbol{x}) - \eta^*(\boldsymbol{x})|$$

$$
\begin{aligned}
&\leq \sum_{l=1}^{L+1} |A_{l+1}^{-}\eta(\psi(\boldsymbol{v}_{l-1} + \boldsymbol{W}_{l-1}A_{l-1}^{+}\eta^{*}(\boldsymbol{x}))) - A_{l+1}^{-}\eta(\psi(\boldsymbol{v}_{l-1}^{*} + \boldsymbol{W}_{l-1}^{*}A_{l-1}^{+}\eta^{*}(\boldsymbol{x})))| \\
&\leq \sum_{l=1}^{L+1} \left(\prod_{j=l}^{L} B_j\right) ||\psi(\boldsymbol{v}_{l-1} + \boldsymbol{W}_{l-1}A_{l-1}^{+}\eta^{*}(\boldsymbol{x})) - \psi(\boldsymbol{v}_{l-1}^{*} + \boldsymbol{W}_{l-1}^{*}A_{l-1}^{+}\eta^{*}(\boldsymbol{x}))||_{\infty} \\
&\leq \sum_{l=1}^{L+1} \left(\prod_{j=l}^{L} B_j\right) ||\boldsymbol{v}_{l-1} - \boldsymbol{v}_{l-1}^{*} + (\boldsymbol{W}_{l-1} - \boldsymbol{W}_{l-1}^{*})A_{l-1}^{+}\eta^{*}(\boldsymbol{x}))||_{\infty} \\
&\leq \sum_{l=1}^{L+1} \left(\prod_{j=l}^{L} B_j\right) ||\overline{\boldsymbol{\delta}}_{l-1}||_{\infty}||A_{l-1}^{+}\eta^{*}(\boldsymbol{x}))||_{\infty} \\
&\leq \sum_{l=1}^{L+1} \left(\prod_{j=l}^{L} B_j\right) \zeta B_{l-1} \prod_{j=0}^{l-2} B_j = \zeta(L+1)\left(\prod_{j=0}^{L} B_j\right) \qquad \text{(A.3)}
\end{aligned}
$$

Recall that we have at most k_l number of nodes in each layer and there are $\binom{k_{l+1}}{s_l} \leq k_{l+1}^{s_l}$ combinations of nodes to choose s_l active nodes in the given layer. Since supremum norm of L_1 norms of the rows of $\overline{\boldsymbol{W}_l}$ is bounded above by B_l in our family of neural networks $\mathcal{F}(L, \boldsymbol{k}, \boldsymbol{s}, \boldsymbol{B})$ so we can discretize these L_1 norms with grid size $\delta B_l/(2(L+1)(\prod_{j=0}^{L} B_j))$ and obtain upper bound on covering number as follows

$$
\begin{aligned}
N(\delta, \mathcal{F}(L, \boldsymbol{k}, \boldsymbol{s}, \boldsymbol{B}), ||.||_{\infty}) &\leq \sum_{s_L^{*} \leq s_L} \cdots \sum_{s_0^{*} \leq s_0} \left[\prod_{l=0}^{L} \left(\frac{B_l}{\delta B_l/(2(L+1)(\prod_{j=0}^{L} B_j))} k_{l+1}\right)^{s_l}\right] \\
&\leq \prod_{l=0}^{L} \left(2\delta^{-1}(L+1)\left(\prod_{j=0}^{L} B_j\right) k_{l+1}\right)^{(s_l+1)} \qquad \text{(A.4)}
\end{aligned}
$$

□

Lemma A.2.5. *Let* $\boldsymbol{\theta}^{*} = \underset{\boldsymbol{\theta} \in \mathcal{F}(L, \boldsymbol{k}, \boldsymbol{s}, \boldsymbol{B})}{\arg\min} |\eta_{\boldsymbol{\theta}} - \eta_0|_{\infty}^{2}$ *and* $\widetilde{W}_l = \sup_i ||\overline{\boldsymbol{w}}_{li} - \overline{\boldsymbol{w}}_{li}^{*}||_1$*, then for any density* $q = \prod_{j=0}^{L} q(\theta_j)$*,*

$$
\begin{aligned}
\int ||\eta_{\boldsymbol{\theta}} - \eta_{\boldsymbol{\theta}^{*}}||_2^2 q(\boldsymbol{\theta}) d\boldsymbol{\theta} &\leq \sum_{j=0}^{L} c_{j-1}^2 \int \widetilde{W}_j^2 q_j(\theta_j) d\theta_j \prod_{m=j+1}^{L} \int (\widetilde{W}_m + B_m)^2 q(\boldsymbol{\theta}) d\boldsymbol{\theta} \\
&+ 2\sum_{j=0}^{L} \sum_{j'=0}^{j-1} c_{j-1} c_{j'-1} \int \widetilde{W}_j(\widetilde{W}_j + B_j) q_j(\theta_j) d\theta_j \prod_{m=j+1}^{L} \int (\widetilde{W}_m + B_m)^2 q(\boldsymbol{\theta}) d\boldsymbol{\theta} \\
&\times \int \widetilde{W}_{j'} q_{j'}(\theta_{j'}) d\theta_{j'} \prod_{m=j'+1}^{j-1} \int (\widetilde{W}_m + B_m) q(\boldsymbol{\theta}) d\boldsymbol{\theta} \qquad \text{(A.5)}
\end{aligned}
$$

where $c_{j-1} \leq \prod_{m=0}^{j-1} B_m$*.*

Proof. Let $\eta_{\boldsymbol{\theta}}^l$ be the partial networks defined as

$$\begin{cases} \eta_{\boldsymbol{\theta}}^0(\boldsymbol{x}) := \psi(\boldsymbol{W}_0\boldsymbol{x} + \boldsymbol{v}_0), \\ \eta_{\boldsymbol{\theta}}^l(\boldsymbol{x}) := \psi(\boldsymbol{W}_l\eta_{\boldsymbol{\theta}}^{l-1}(\boldsymbol{x}) + \boldsymbol{v}_l), \\ \eta_{\boldsymbol{\theta}}^L(\boldsymbol{x}) := \boldsymbol{W}_L\eta_{\boldsymbol{\theta}}^{L-1}(\boldsymbol{x}) + \boldsymbol{v}_L. \end{cases}$$

Similar to the proof of theorem 2 in Chérief-Abdellatif (2020), define

$$\varphi_l(\boldsymbol{\theta}) = \sup_{x\in[0,1]^p} \sup_{1\le i\le k_{l+1}} |\eta_{\boldsymbol{\theta}}^l(\boldsymbol{x})_i - \eta_{\boldsymbol{\theta}^*}^l(\boldsymbol{x})_i|.$$

We next show by induction

$$\varphi_l(\boldsymbol{\theta}) \le \sum_{j=0}^{l} \widetilde{W}_j c_{j-1} R_{j+1}^l$$

where we define $c_l = \max(\sup_{x\in[0,1]^p} \sup_{1\le i\le k_{l+1}} |\eta_{\boldsymbol{\theta}^*}^l(\boldsymbol{x})_i|, 1)$, $c_0 = 1$, $R_{j+1}^l = \prod_{m=j+1}^l (\widetilde{W}_m + B_m)$.

Claim: $c_l \le B_l c_{l-1}$. Note

$$\begin{aligned} c_l &\le \sup_{x\in[0,1]^p} \sup_{1\le i\le k_{l+1}} (|\boldsymbol{w}_{li}^{*\top} \eta_{\boldsymbol{\theta}^*}^{l-1}(\boldsymbol{x})| + |v_{li}|) \\ &\le \sup_{x\in[0,1]^p} \sup_{1\le i\le k_{l+1}} (\sum_{j=1}^{k_l} |w_{lij}^*||\eta_{\boldsymbol{\theta}^*}^{l-1}(\boldsymbol{x})_j| + |v_{li}|) \\ &\le \sup_{1\le i\le k_{l+1}} (c_{l-1} \sum_{j=1}^{k_l} |w_{lij}^*| + c_{l-1}|v_{li}|) \\ &\le c_{l-1} \sup_{1\le i\le k_{l+1}} ||\overline{\boldsymbol{w}}_{li}^*||_1 = B_l c_{l-1} \end{aligned}$$

where the above result holds since $\sup_i ||\overline{\boldsymbol{w}}_{li}^*||_1 \le B_l$. Next,

$$\begin{aligned} \varphi_l(\boldsymbol{\theta}) &\le \sup_{x\in[0,1]^p} \sup_{1\le i\le k_{l+1}} (\sum_{j=1}^{k_l} |w_{lij}\eta_{\boldsymbol{\theta}}^{l-1}(\boldsymbol{x})_j - w_{lij}^*\eta_{\boldsymbol{\theta}^*}^{l-1}(\boldsymbol{x})_j| + |v_{li} - v_{li}^*|) \\ &\le \sup_{x\in[0,1]^p} \sup_{1\le i\le k_{l+1}} (\sum_{j=1}^{k_l} |w_{lij}\eta_{\boldsymbol{\theta}}^{l-1}(\boldsymbol{x})_j - w_{lij}^*\eta_{\boldsymbol{\theta}}^{l-1}(\boldsymbol{x})_j| \\ &\qquad + |w_{lij}^*\eta_{\boldsymbol{\theta}}^{l-1}(\boldsymbol{x})_j - w_{lij}^*\eta_{\boldsymbol{\theta}^*}^{l-1}(\boldsymbol{x})_j| + |v_{li} - v_{li}^*|) \\ &\le \sup_{x\in[0,1]^p} \sup_{1\le i\le k_{l+1}} (\sum_{j=1}^{k_l} |w_{lij} - w_{lij}^*||\eta_{\boldsymbol{\theta}}^{l-1}(\boldsymbol{x})_j| \end{aligned}$$

$$+\sum_{j=1}^{k_l}|w^*_{lij}||\eta^{l-1}_{\boldsymbol{\theta}}(\boldsymbol{x})_j-\eta^{l-1}_{\boldsymbol{\theta}^*}(\boldsymbol{x})_j|+|v_{li}-v^*_{li}|)$$

$$\leq \sup_{x\in[0,1]^p}\sup_{1\leq i\leq k_{l+1}}(\sum_{j=1}^{k_l}|w_{lij}-w^*_{lij}||\eta^{l-1}_{\boldsymbol{\theta}}(\boldsymbol{x})_j-\eta^{l-1}_{\boldsymbol{\theta}^*}(\boldsymbol{x})_j|$$

$$+\sum_{j=1}^{k_l}|w_{lij}-w^*_{lij}||\eta^{l-1}_{\boldsymbol{\theta}^*}(\boldsymbol{x})_j|+|v_{li}-v^*_{li}|)+\varphi_{l-1}(\boldsymbol{\theta})B_l$$

$$\leq \widetilde{W}_l(\varphi_{l-1}(\boldsymbol{\theta})+c_{l-1})+\varphi_{l-1}(\boldsymbol{\theta})B_l=\varphi_{l-1}(\boldsymbol{\theta})(\widetilde{W}_l+B_l)+c_{l-1}\widetilde{W}_l$$

Now applying recursion we get

$$\varphi_l(\boldsymbol{\theta})\leq(\varphi_{l-2}(\boldsymbol{\theta})(\widetilde{W}_{l-1}+B_{l-1})+c_{l-2}\widetilde{W}_{l-1})(\widetilde{W}_l+B_l)+c_{l-1}\widetilde{W}_l$$

$$=\varphi_{l-2}(\boldsymbol{\theta})(\widetilde{W}_l+B_l)(\widetilde{W}_{l-1}+B_{l-1})+c_{l-2}\widetilde{W}_{l-1}(\widetilde{W}_l+B_l)+c_{l-1}\widetilde{W}_l$$

Repeating this we get

$$\varphi_l(\boldsymbol{\theta})\leq\varphi_0(\boldsymbol{\theta})\prod_{j=1}^{l}(\widetilde{W}_j+B_j)+\sum_{j=1}^{l}c_{j-1}\widetilde{W}_j\prod_{u=j+1}^{l}(\widetilde{W}_j+B_j)$$

$$=\widetilde{W}_0\prod_{j=1}^{l}(\widetilde{W}_j+B_j)+\sum_{j=1}^{l}B_1\cdots B_{j-1}\widetilde{W}_j\prod_{u=j+1}^{l}(\widetilde{W}_j+B_j)$$

$$=\sum_{j=0}^{l}B_1\cdots B_{j-1}\widetilde{W}_j\prod_{u=j+1}^{l}(\widetilde{W}_j+B_j)=\sum_{j=0}^{l}\widetilde{W}_jc_{j-1}R^l_{j+1}$$

$$\int||\eta_{\boldsymbol{\theta}}-\eta_{\boldsymbol{\theta}^*}||_2^2q(\boldsymbol{\theta})d\boldsymbol{\theta}\leq\int||\eta_{\boldsymbol{\theta}}-\eta_{\boldsymbol{\theta}^*}||_\infty^2q(\boldsymbol{\theta})d\boldsymbol{\theta}=\int\varphi_L^2(\boldsymbol{\theta})q(\boldsymbol{\theta})d\boldsymbol{\theta}$$

$$=\int(\sum_{j=0}^{L}\widetilde{W}_jc_{j-1}R^L_{j+1})^2q(\boldsymbol{\theta})d\boldsymbol{\theta}$$

$$=\sum_{j=0}^{L}c_{j-1}^2\int\widetilde{W}_j^2(R^L_{j+1})^2q(\boldsymbol{\theta})d\boldsymbol{\theta}+2\sum_{j=0}^{L}\sum_{j'=0}^{j-1}c_{j-1}c_{j'-1}\int\widetilde{W}_j\widetilde{W}_{j'}R^L_{j+1}R^L_{j'+1}q(\boldsymbol{\theta})d\boldsymbol{\theta}$$

$$=\sum_{j=0}^{L}c_{j-1}^2\int\widetilde{W}_j^2\left(\prod_{m=j+1}^{L}(\widetilde{W}_m+B_m)\right)^2q(\boldsymbol{\theta})d\boldsymbol{\theta}$$

$$+2\sum_{j=0}^{L}\sum_{j'=0}^{j-1}c_{j-1}c_{j'-1}\int\widetilde{W}_j\widetilde{W}_{j'}\prod_{m=j+1}^{L}(\widetilde{W}_m+B_m)\prod_{m=j'+1}^{L}(\widetilde{W}_m+B_m)q(\boldsymbol{\theta})d\boldsymbol{\theta}$$

The proof follows by noting $q(\boldsymbol{\theta})=\prod_{j=0}^{L}q(\theta_j)$. □

Lemma A.2.6. *Suppose Lemma 3.4.1 and Lemma 3.4.2 in the Section 3.4 hold, with dominating probability*

$$\log \int_{\mathcal{H}^c_{\epsilon_n}} \frac{P^n_{\boldsymbol{\theta}}}{P^n_0}\pi(\boldsymbol{\theta})d\boldsymbol{\theta} \leq -\frac{Cn\epsilon_n^2}{\sum u_l}$$

Proof. Let $\mathcal{F}_n = \mathcal{F}(L, \boldsymbol{k}, \boldsymbol{s}^\circ, \boldsymbol{B}^\circ)$, $s_l^\circ + 1 = n\epsilon_n^2/\sum_{j=0}^L u_j$, $\log B_l^\circ = n\epsilon_n^2/((L+1)\sum_{j=0}^L (s_j^\circ+1))$ and $\mathcal{H}_{\epsilon_n} = \{\boldsymbol{\theta} : d_{\mathrm{H}}(P_0, P_{\boldsymbol{\theta}}) < \epsilon_n\}$ is the Hellinger neighborhood of size ϵ_n

$$\begin{aligned}\int_{\mathcal{H}^c_{\epsilon_n}} \frac{P^n_{\boldsymbol{\theta}}}{P^n_0}\widetilde{\pi}(\boldsymbol{\theta})d\boldsymbol{\theta} &\leq \int_{\mathcal{H}^c_{\epsilon_n}\cap\mathcal{F}_n} \frac{P^n_{\boldsymbol{\theta}}}{P^n_0}\widetilde{\pi}(\boldsymbol{\theta})d\boldsymbol{\theta} + \int_{\mathcal{F}^c_n} \frac{P^n_{\boldsymbol{\theta}}}{P^n_0}\widetilde{\pi}(\boldsymbol{\theta})d\boldsymbol{\theta} \\ &\leq \int_{\mathcal{H}^c_{\epsilon_n}\cap\mathcal{F}_n} \frac{P^n_{\boldsymbol{\theta}}}{P^n_0}\widetilde{\pi}(\boldsymbol{\theta})d\boldsymbol{\theta} + \exp\left(-\frac{(C_0/2)n\epsilon_n^2}{\sum u_l}\right)\end{aligned}$$

where the last inequality follows from Lemma 3.4.2 because by Markov's inequality

$$\begin{aligned}&\mathbb{P}_{P^n_0}\left(\int_{\mathcal{F}^c_n} \frac{P^n_{\boldsymbol{\theta}}}{P^n_0}\widetilde{\pi}(\boldsymbol{\theta})d\boldsymbol{\theta} > \exp\left(-\frac{(C_0/2)n\epsilon_n^2}{\sum u_l}\right)\right) \\ &\leq \exp\left(\frac{(C_0/2)n\epsilon_n^2}{\sum u_l}\right)\mathbb{E}_{P^n_0}\left(\int_{\mathcal{F}^c_n} \frac{P^n_{\boldsymbol{\theta}}}{P^n_0}\widetilde{\pi}(\boldsymbol{\theta})d\boldsymbol{\theta}\right) \\ &\leq \exp\left(\frac{(C_0/2)n\epsilon_n^2}{\sum u_l}\right)\widetilde{\Pi}(\mathcal{F}_n^c) = \exp\left(-\frac{(C_0/2)n\epsilon_n^2}{\sum u_l}\right) \to 0\end{aligned}$$

Further,

$$\int_{\mathcal{H}^c_{\epsilon_n}\cap\mathcal{F}_n} \frac{P^n_{\boldsymbol{\theta}}}{P^n_0}\widetilde{\pi}(\boldsymbol{\theta})d\boldsymbol{\theta} \leq \underbrace{\int_{\mathcal{H}^c_{\epsilon_n}\cap\mathcal{F}_n} \phi\frac{P^n_{\boldsymbol{\theta}}}{P^n_0}\widetilde{\pi}(\boldsymbol{\theta})d\boldsymbol{\theta}}_{T_1} + \underbrace{\int_{\mathcal{H}^c_{\epsilon_n}\cap\mathcal{F}_n} (1-\phi)\frac{P^n_{\boldsymbol{\theta}}}{P^n_0}\widetilde{\pi}(\boldsymbol{\theta})d\boldsymbol{\theta}}_{T_2}$$

Next, borrowing steps from proof of theorem 3.1 in Pati et al. (2018), we have $\mathbb{E}_{P^n_0}(\phi) \leq \exp(-C_1 n\epsilon_n^2)$, thus for any $C_1' < C_1$, $\phi \leq \exp(-C_1' n\epsilon_n^2)$ with probability at least $1 - \exp(-(C_1 - C_1')n\epsilon_n^2)$. Thus,

$$T_1 \leq \exp(-C_1' n\epsilon_n^2)T_1 + T_2$$

which implies with dominating probability $T_1 \leq T_2$. Thus, it only remains to show $T_2 \leq \exp(-C_2'(n\epsilon_n^2)/(\sum u_l))$ for some $C_2' > 0$. This is true since

$$\mathbb{P}_{P^n_0}(T_2 > e^{-\frac{C_2 n\epsilon_n^2}{\sum u_l}}) \leq e^{C_2\frac{n\epsilon_n^2}{\sum u_l}}\mathbb{E}_{P^n_0}(T_2) \leq e^{\frac{C_2 n\epsilon_n^2}{\sum u_l}}\int_{\mathcal{H}^c_{\epsilon_n}\cap\mathcal{F}_n} \mathbb{E}_{P_{\boldsymbol{\theta}}}(1-\phi)\widetilde{\pi}(\boldsymbol{\theta})d\boldsymbol{\theta}$$

$$\leq e^{\frac{C_2 n\epsilon_n^2}{\sum u_l}} \int_{\mathcal{H}_{\epsilon_n}^c \cap \mathcal{F}_n} e^{-C_2 n d_{\mathrm{H}}^2(P_0, P_{\boldsymbol{\theta}})} \widetilde{\pi}(\boldsymbol{\theta}) d\boldsymbol{\theta}$$
$$\leq e^{\frac{C_2 n\epsilon_n^2}{\sum u_l}} e^{-C_2 n\epsilon_n^2} \int_{\mathcal{H}_{\epsilon_n}^c \cap \mathcal{F}_n} \widetilde{\pi}(\boldsymbol{\theta}) d\boldsymbol{\theta} \leq \exp(-C_2' n\epsilon_n^2 / \sum u_l)$$

Therefore, for sufficiently large n and $C = \min(C_0/2, C_2')/2$

$$\int_{\mathcal{H}_{\epsilon_n}^c} \frac{P_{\boldsymbol{\theta}}^n}{P_0^n} \widetilde{\pi}(\boldsymbol{\theta}) d\boldsymbol{\theta} \leq 2\exp(-C_2' n\epsilon_n^2 / \sum u_l) + \exp(-(C_0/2) n\epsilon_n^2 / \sum u_l) \leq \exp(-C n\epsilon_n^2 / \sum u_l)$$

□

Lemma A.2.7. *Suppose Lemma 3.4.3 part 1. in the Section 3.4 holds, then for any $M_n \to \infty$, with dominating probability,*

$$\log \int \frac{P_0^n}{P_{\boldsymbol{\theta}}^n} \widetilde{\pi}(\boldsymbol{\theta}) d\boldsymbol{\theta} \leq n M_n (\sum r_l + \xi)$$

Proof. By Markov's inequality,

$$\mathbb{P}_{P_0^n} \left(\log \int \frac{P_0^n}{P_{\boldsymbol{\theta}}^n} \widetilde{\pi}(\boldsymbol{\theta}) \geq n M_n (\sum r_l + \xi) \right) \leq \frac{1}{n M_n (\sum r_l + \xi)} \mathbb{E}_{P_0^n} \left| \log \int \frac{P_{\boldsymbol{\theta}}^n}{P_0^n} \widetilde{\pi}(\boldsymbol{\theta}) d\boldsymbol{\theta} \right|$$
$$= \frac{1}{n M_n (\sum r_l + \xi)} \int \left| \log \int \frac{P_{\boldsymbol{\theta}}^n}{P_0^n} \widetilde{\pi}(\boldsymbol{\theta}) d\boldsymbol{\theta} \right| P_0^n d\mu$$
$$\leq \frac{1}{n M_n (\sum r_l + \xi)} \left(d_{\mathrm{KL}}(P_0^n, L^*) + \frac{2}{e} \right)$$

where $L^* = \int P_{\boldsymbol{\theta}}^n \widetilde{\pi}(\boldsymbol{\theta}) d\boldsymbol{\theta}$ and the last inequality follows from Lemma A.2.1.

$$d_{\mathrm{KL}}(P_0^n, L^*) = \mathbb{E}_{P_0^n} \left(\log \frac{P_0^n}{\int P_{\boldsymbol{\theta}}^n \widetilde{\pi}(\boldsymbol{\theta}) d\boldsymbol{\theta}} \right) \leq \mathbb{E}_{P_0^n} \left(\log \frac{P_0^n}{\int_{N_{\sum r_l + \xi}} P_{\boldsymbol{\theta}}^n \widetilde{\pi}(\boldsymbol{\theta}) d\boldsymbol{\theta}} \right)$$
$$\leq \int_{\mathcal{N}_{\sum r_l + \xi}} \widetilde{\pi}(\boldsymbol{\theta}) d\boldsymbol{\theta} + \int_{\mathcal{N}_{\sum r_l + \xi}} d_{\mathrm{KL}}(P_0^n, P_{\boldsymbol{\theta}}^n) \widetilde{\pi}(\boldsymbol{\theta}) d\boldsymbol{\theta} \quad \text{Jensen's inequality}$$
$$\leq -\log e^{-nC(\sum r_l + \xi)} + n(\sum r_l + \xi) = n(C+1)(\sum r_l + \xi)$$

where the last inequality follows from Lemma 3.4.3 part 1. in the Section 3.4. The proof follows by noting $C/M_n \to 0$. □

Lemma A.2.8. *Suppose Lemma 3.4.3 part 2. in the Section 3.4 holds, then for any $M_n \to \infty$, with dominating probability,*

$$d_{KL}(q, \pi) + \sum_{\boldsymbol{z}} \int \log \frac{P_0^n}{P_{\boldsymbol{\theta}}^n} q(\boldsymbol{\theta}, \boldsymbol{z}) d\boldsymbol{\theta} \leq n M_n (\sum r_l + \xi)$$

Proof. By Markov's inequality we have

$$\begin{aligned}
&\mathbb{P}_{P_0^n}\left(d_{\mathrm{KL}}(q,\pi)+\sum_{\boldsymbol{z}}\int q(\boldsymbol{\theta},\boldsymbol{z})\log\frac{P_0^n}{P_{\boldsymbol{\theta}}^n}d\boldsymbol{\theta} > nM_n(\sum r_l+\xi)\right)\\
&\leq \frac{1}{nM_n(\sum r_l+\xi)}\left(d_{\mathrm{KL}}(q,\pi)+\mathbb{E}_{P_0^n}\left|\sum_{\boldsymbol{z}}\int q(\boldsymbol{\theta},\boldsymbol{z})\log\frac{P_0^n}{P_{\boldsymbol{\theta}}^n}d\boldsymbol{\theta}\right|\right)\\
&\leq \frac{1}{nM_n(\sum r_l+\xi)}\left(d_{\mathrm{KL}}(q,\pi)+\mathbb{E}_{P_0^n}\left(\sum_{\boldsymbol{z}}\int q(\boldsymbol{\theta},\boldsymbol{z})\left|\log\frac{P_{\boldsymbol{\theta}}^n}{P_0^n}\right|d\boldsymbol{\theta}\right)\right)\\
&= \frac{1}{nM_n(\sum r_l+\xi)}\left(d_{\mathrm{KL}}(q,\pi)+\sum_{\boldsymbol{z}}\int q(\boldsymbol{\theta},\boldsymbol{z})\int\left|\log\frac{P_0^n}{P_{\boldsymbol{\theta}}^n}\right|P_0^n d\mu d\boldsymbol{\theta}\right)
\end{aligned}$$

By Lemma A.2.1, we get

$$\begin{aligned}
&\leq \frac{1}{nM_n(\sum r_l+\xi)}\left(d_{\mathrm{KL}}(q,\pi)+\sum_{\boldsymbol{z}}\int q(\boldsymbol{\theta},\boldsymbol{z})\left(d_{\mathrm{KL}}(P_0^n,P_{\boldsymbol{\theta}}^n)+\frac{2}{e}\right)d\boldsymbol{\theta}\right)\\
&= \frac{1}{nM_n(\sum r_l+\xi)}\left(d_{\mathrm{KL}}(q,\pi)+n\sum_{\boldsymbol{z}}\int q(\boldsymbol{\theta},\boldsymbol{z})d_{\mathrm{KL}}(P_0,P_{\boldsymbol{\theta}})d\boldsymbol{\theta}+\frac{2}{e}\right)\\
&= \frac{C}{nM_n(\sum r_l+\xi)}\Big(n(\sum r_l+\xi)+(2/e)\Big)\to 0
\end{aligned}$$

where the last line in the above holds due to Lemma 4.3 part 2. in the Section 3.4. □

A.3 Proof of Lemmas and Corollary in the Section 3.4

Proof of Lemma 3.4.1

Take $s_l^\circ+1=(n\epsilon_n^2)/(\sum_{j=0}^L u_j)$ and $\log B_l^\circ=(n\epsilon_n^2)/((L+1)\sum_{j=0}^L(s_j^\circ+1))$.

We know from Lemma 2 of Ghosal and Van Der Vaart (2007) that, there exists a function $\varphi\in[0,1]$, such that

$$\mathbb{E}_{P_0}(\varphi)\leq\exp\{-nd_{\mathrm{H}}^2(P_{\boldsymbol{\theta}_1},P_0)/2\}$$

$$\mathbb{E}_{P_{\boldsymbol{\theta}}}(1-\varphi)\leq\exp\{-nd_{\mathrm{H}}^2(P_{\boldsymbol{\theta}_1},P_0)/2\}$$

for all $P_{\boldsymbol{\theta}}\in\mathcal{F}(L,\boldsymbol{k},\boldsymbol{s}^\circ,\boldsymbol{B}^\circ)$ satisfying $d_{\mathrm{H}}(P_{\boldsymbol{\theta}},P_{\boldsymbol{\theta}_1})\leq d_{\mathrm{H}}(P_0,P_{\boldsymbol{\theta}_1})/18$.

Let $H=N(\epsilon_n/19,\mathcal{F}(L,\boldsymbol{k},\boldsymbol{s}^\circ,\boldsymbol{B}^\circ),d_{\mathrm{H}}(.,.))$ denote the covering number of $\mathcal{F}(L,\boldsymbol{k},\boldsymbol{s}^\circ,\boldsymbol{B}^\circ)$, i.e., there exist H Hellinger balls of radius $\epsilon_n/19$, that entirely cover $\mathcal{F}(L,\boldsymbol{k},\boldsymbol{s}^\circ,\boldsymbol{B}^\circ)$. For

any $\boldsymbol{\theta} \in \mathcal{F}(L, \boldsymbol{k}, \boldsymbol{s}^\circ, \boldsymbol{B}^\circ)$ w.l.o.g we assume $P_{\boldsymbol{\theta}}$ belongs to the Hellinger ball centered at $P_{\boldsymbol{\theta}_h}$ and if $d_{\mathrm{H}}(P_{\boldsymbol{\theta}}, P_0) > \epsilon_n$, then we must have that $d_{\mathrm{H}}(P_0, P_{\boldsymbol{\theta}_h}) > (18/19)\epsilon_n$ and there exists a testing function φ_h, such that

$$\begin{aligned}
\mathbb{E}_{P_0}(\varphi_h) &\leq \exp\{-n d_{\mathrm{H}}^2(P_{\boldsymbol{\theta}_h}, P_0)/2\} \\
&\leq \exp\{-((18^2/19^2)/2) n\epsilon_n^2\} \\
\mathbb{E}_{P_{\boldsymbol{\theta}}}(1-\varphi_h) &\leq \exp\{-n d_{\mathrm{H}}^2(P_{\boldsymbol{\theta}_h}, P_0)/2\} \\
&\leq \exp\{-n(d_{\mathrm{H}}(P_0, P_{\boldsymbol{\theta}}) - \epsilon_n/19)^2/2\} \\
&\leq \exp\{-((18^2/19^2)/2) n d_{\mathrm{H}}^2(P_0, P_{\boldsymbol{\theta}})\}.
\end{aligned}$$

Next we define $\phi = \max_{h=1,\cdots,H} \varphi_h$. Then we must have

$$\begin{aligned}
\mathbb{E}_{P_0}(\phi) &\leq \sum_h \mathbb{E}_{P_0}(\varphi_h) \leq H \exp\{-((18^2/19^2)/2) n\epsilon_n^2\} \\
&\leq \exp\{-((18^2/19^2)/2) n\epsilon_n^2 - \log H\}
\end{aligned}$$

Using Lemma A.2.4 with $\boldsymbol{s} = \boldsymbol{s}^\circ$ and $\boldsymbol{B} = \boldsymbol{B}^\circ$, we get

$$\begin{aligned}
&\log H = \log N(\epsilon_n/19, \mathcal{F}(L, \boldsymbol{k}, \boldsymbol{s}^\circ, \boldsymbol{B}^\circ), d_{\mathrm{H}}(.,.)) \\
&\leq \log N(\sqrt{8}\sigma_e^2\epsilon_n/19, \mathcal{F}(L, \boldsymbol{k}, \boldsymbol{s}^\circ, \boldsymbol{B}^\circ), ||.||_\infty) \\
&\leq \log \left[\prod_{l=0}^{L} \left(\frac{38}{\sqrt{8}\sigma_e^2\epsilon_n}(L+1) \left(\prod_{j=0}^{L} B_j^\circ \right) k_{l+1} \right)^{(s_l^\circ+1)} \right] \\
&= \sum_{l=0}^{L} (s_l^\circ + 1) \log \left(\frac{38}{\sqrt{8}\sigma_e^2\epsilon_n}(L+1) \left(\prod_{j=0}^{L} B_j^\circ \right) k_{l+1} \right) \\
&\leq C \left[\sum_{l=0}^{L} (s_l^\circ + 1) \left(\log \frac{1}{\epsilon_n} + \log(L+1) + \sum_{j=0}^{L} \log B_j^\circ + \log k_{l+1} \right) \right] \\
&\leq C \sum_{l=0}^{L} (s_l^\circ + 1)(\log n + \log(L+1) + \sum_{j=0}^{L} \log B_j^\circ + \log k_{l+1}) \\
&\leq C \sum_{l=0}^{L} (s_l^\circ + 1)(\log n + \log(L+1) + \sum_{j=0}^{L} \log B_j^\circ + \log k_{l+1} + \log(k_l+1)) \leq C n\epsilon_n^2
\end{aligned}$$

where, C in each step is different which tends to absorb the extra constants in it. First inequality holds due to the following

$$d^2_{\mathrm{H}}(P_{\boldsymbol{\theta}}, P_0) \leq 1 - \exp\left\{-\frac{1}{8\sigma_e^2}||\eta_0 - \eta_{\boldsymbol{\theta}}||^2_\infty\right\}$$

and $\epsilon_n = o(1)$, the second inequality is because of (A.4), and fourth inequality is because of $s_l^\circ \log(1/\epsilon_n) \asymp s_l^\circ \log n$. Therefore,

$$\mathbb{E}_{P_0}(\phi) \leq \sum_h \mathbb{E}_{P_0}(\varphi_h) = \exp\{-C_1 n\epsilon_n^2\}$$

for some $C_1 = (18^2/19^2)/2 - 1/4$. On the other hand, for any $\boldsymbol{\theta}$, such that $d_{\mathrm{H}}(P_{\boldsymbol{\theta}}, P_0) \geq \epsilon_n$, say $P_{\boldsymbol{\theta}}$ belongs to the hth Hellinger ball, then we have

$$\mathbb{E}_{P_{\boldsymbol{\theta}}}(1-\phi) \leq \mathbb{E}_{P_{\boldsymbol{\theta}}}(1-\varphi_h) \leq \exp\{-C_2 n d^2_{\mathrm{H}}(P_0, P_{\boldsymbol{\theta}})\}$$

where $C_2 = (18^2/19^2)/2$. This concludes the proof. □

Proof of Lemma 3.4.2

$$\textit{Assumption:} \quad s_l^\circ + 1 = (n\epsilon_n^2)/(\sum_{j=0}^{L} u_j),\ \lambda_l k_{l+1}/s_l^\circ \to 0,\ \sum u_l \log L = o(n\epsilon_n^2) \tag{A.6}$$

$$\begin{aligned}
\widetilde{\Pi}(\mathcal{F}(L, \boldsymbol{k}, \boldsymbol{s}^\circ, \boldsymbol{B}^\circ)^c) &\leq \widetilde{\Pi}\left(\bigcup_{l=0}^{L}\{||\widetilde{\boldsymbol{w}}_l||_0 > s_l^\circ\}\right) + \widetilde{\Pi}\left(\bigcup_{l=0}^{L}\{||\widetilde{\boldsymbol{w}}_l||_\infty > B_l^\circ\}\right) \\
&\leq \sum_{l=0}^{L}\widetilde{\Pi}(||\widetilde{\boldsymbol{w}}_l||_0 > s_l^\circ) + \sum_{l=0}^{L}\widetilde{\Pi}(||\widetilde{\boldsymbol{w}}_l||_\infty > B_l^\circ) \\
&= \sum_{l=0}^{L}\sum_{\boldsymbol{z}}\Pi(||\widetilde{\boldsymbol{w}}_l||_0 > s_l^\circ|\boldsymbol{z})\pi(\boldsymbol{z}) + \sum_{l=0}^{L}\sum_{\boldsymbol{z}}\Pi(||\widetilde{\boldsymbol{w}}_l||_\infty > B_l^\circ|\boldsymbol{z})\pi(\boldsymbol{z}) \\
&\leq \sum_{l=0}^{L}\mathbb{P}\left(\sum_{i=1}^{k_{l+1}} z_{li} > s_l^\circ\right) + \sum_{l=0}^{L}\mathbb{P}\left(\sup_{i=1,\cdots,k_{l+1}}||\overline{\boldsymbol{w}}_{li}||_1 > B_l^\circ \Big| \boldsymbol{z}\right)
\end{aligned}$$

where $\widetilde{\boldsymbol{w}}_l = (||\overline{\boldsymbol{w}}_{l1}||_1, \cdots, ||\overline{\boldsymbol{w}}_{lk_{l+1}}||_1)^T$ and the last inequality holds since $\Pi(||\widetilde{\boldsymbol{w}}_l||_0 > s_l^\circ|\boldsymbol{z}) \leq 1$, $\Pi(||\widetilde{\boldsymbol{w}}_l||_0 > s_l^\circ|\boldsymbol{z}) = 1$ iff $\sum z_{li} > \boldsymbol{s}_l^\circ$ and $\pi(\boldsymbol{z}) \leq 1$. We now break the proof in two parts as follows.

Part 1.

$$\sum_{l=0}^{L}\mathbb{P}\left(\sum_{i=1}^{k_{l+1}} z_{li} > s_l^\circ\right) = \sum_{l=0}^{L}\mathbb{P}\left(\sum_{i=1}^{k_{l+1}} z_{li} - k_{l+1}\lambda_l > s_l^\circ - k_{l+1}\lambda_l\right)$$

By Bernstein inequality

$$\begin{aligned}
&\leq \sum_{l=0}^{L}\exp\left(\frac{-1/2(s_l^\circ - k_{l+1}\lambda_l)^2}{k_{l+1}\lambda_l(1-\lambda_l) + 1/3(s_l^\circ - k_{l+1}\lambda_l)}\right) \leq \sum_{l=0}^{L}\exp\left(\frac{-1/2(s_l^\circ - k_{l+1}\lambda_l)^2}{k_{l+1}\lambda_l + 1/3(s_l^\circ - k_{l+1}\lambda_l)}\right)\\
&= \sum_{l=0}^{L}\exp\left(\frac{-s_l^\circ/2(1-k_{l+1}\lambda_l/s_l^\circ)^2}{1/3(1+2k_{l+1}\lambda_l/s_l^\circ)}\right) \to \sum_{l=0}^{L}\exp\left(-\frac{3s_l^\circ}{2}\right) \qquad \text{since } \frac{k_{l+1}\lambda_l}{s_l^\circ} \to 0 \text{ by (A.6)}\\
&= \sum_{l=0}^{L}\exp\left(-\frac{3n\epsilon_n^2}{4\sum u_l} + \frac{3}{2}\right) \leq 5(L+1)\exp\left(-\frac{n\epsilon_n^2}{2\sum u_l}\right) \leq \exp\left(-\frac{n\epsilon_n^2}{4\sum u_l}\right)
\end{aligned}$$

since $\sum u_l \log(5(L+1)) \sim \sum u_l \log L = o(n\epsilon_n^2)$ by (A.6).

Part 2.

$$\begin{aligned}
&\sum_{l=0}^{L}\mathbb{P}\left(\sup_{i=1,\cdots,k_{l+1}} ||\boldsymbol{w}_{li}||_1 > B_l^\circ \Big| \boldsymbol{z}\right)\\
&\leq \sum_{l=0}^{L}\sum_{i=1}^{k_{l+1}}\mathbb{P}\left(||\boldsymbol{w}_{li}||_1 > B_l^\circ \Big| \boldsymbol{z}\right)\\
&\leq \sum_{l=0}^{L}\sum_{i=1}^{k_{l+1}}\mathbb{P}\left(||\boldsymbol{w}_{li}||_\infty > \frac{B_l^\circ}{k_l+1} \Big| \boldsymbol{z}\right)\\
&\leq \sum_{l=0}^{L}\sum_{i=1}^{k_{l+1}}\sum_{j=1}^{k_l+1}\mathbb{P}\left(|w_{lij}| > \frac{B_l^\circ}{k_l+1} \Big| \boldsymbol{z}\right)\\
&\leq 2\sum_{l=0}^{L}\sum_{i=1}^{k_{l+1}}\sum_{j=1}^{k_l+1}\exp\left(-\frac{B_l^{\circ 2}}{(k_l+1)^2}\right) \qquad \text{By concentration inequality}\\
&= 2\sum_{l=0}^{L}\sum_{i=1}^{k_{l+1}}\sum_{j=1}^{k_l+1}\exp\Big(-\exp\Big(\frac{2n\epsilon_n^2}{((L+1)\sum_{j'=0}^{L}(s_{j'}^\circ+1)} - 2\log(k_l+1)\Big)\Big)\\
&\leq \sum_{l=0}^{L}\sum_{i=1}^{k_{l+1}}\sum_{j=1}^{k_l+1}\frac{1}{(L+1)k_{l+1}(k_l+1)}\exp(-n\epsilon_n^2) = \exp(-n\epsilon_n^2)
\end{aligned}$$

where the third inequality holds since $|w_{lij}|$ given $\boldsymbol{z}$ is bound above by a $|N(0,\sigma_0^2)|$ random variable. The above proof holds as long as

$$\exp\left(\frac{2n\epsilon_n^2}{(L+1)\sum_{j'=0}^{L}(s_{j'}^\circ+1)} - 2\log(k_l+1)\right) \geq n\epsilon_n^2 + \log(L+1) + \log k_{l+1} + \log(k_l+1) + \log 2$$

Taking log on both sides we get

$$\left(\frac{n\epsilon_n^2}{(L+1)\sum_{j'=0}^{L}(s_{j'}^{\circ}+1)}-\log(k_l+1)\right)\geq\frac{1}{2}\log(n\epsilon_n^2+\log(L+1)+\log k_{l+1}+\log(k_l+1)+\log 2)$$

This is true since $\sum_{j'=0}^{L}(s_{j'}^{\circ}+1)=(L+1)n\epsilon_n^2/\sum u_l$ is bounded above by

$$\frac{n\epsilon_n^2}{(L+1)(\log(k_l+1)+\frac{1}{2}\log(n\epsilon_n^2+\log(L+1)+\log k_{l+1}+\log(k_l+1)+\log 2)}$$

□

Proof of Lemma 3.4.3 part 1.

$$\textit{Assumption:}\quad -\log\lambda_l=O\{(k_l+1)\vartheta_l\},\ -\log(1-\lambda_l)=O\{(s_l/k_{l+1})(k_l+1)\vartheta_l\}\tag{A.7}$$

$$d_{\mathrm{KL}}(P_0,P_{\boldsymbol{\theta}})=\int_{\boldsymbol{x}\in[0,1]^p}\int_{y\in R}\left(\log\frac{P_0(y,\boldsymbol{x})}{P_{\boldsymbol{\theta}}(y,\boldsymbol{x})}\right)P_0(y,\boldsymbol{x})dyd\boldsymbol{x}$$

$$P_0(y,\boldsymbol{x})=\frac{1}{\sqrt{2\pi\sigma_e^2}}\exp\left(-\frac{(y-\eta_0(\boldsymbol{x}))^2}{2\sigma_e^2}\right)\qquad P_{\boldsymbol{\theta}}(y,\boldsymbol{x})=\frac{1}{\sqrt{2\pi\sigma_e^2}}\exp\left(-\frac{(y-\eta_{\boldsymbol{\theta}}(\boldsymbol{x}))^2}{2\sigma_e^2}\right)$$

So we get,

$$\begin{aligned}
d_{\mathrm{KL}}(P_0,P_{\boldsymbol{\theta}})&=\int_{\boldsymbol{x}\in[0,1]^p}\int_{y\in\mathbb{R}}\log\left(\exp\left[-\frac{(y-\eta_0(\boldsymbol{x}))^2}{2\sigma_e^2}+\frac{(y-\eta_{\boldsymbol{\theta}}(\boldsymbol{x}))^2}{2\sigma_e^2}\right]\right)P_0(y,\boldsymbol{x})dyd\boldsymbol{x}\\
&=\int_{\boldsymbol{x}\in[0,1]^p}\int_{y\in\mathbb{R}}\frac{2y(\eta_0(\boldsymbol{x})-\eta_{\boldsymbol{\theta}}(\boldsymbol{x}))-(\eta_0^2(\boldsymbol{x})-\eta_{\boldsymbol{\theta}}^2(\boldsymbol{x}))}{2\sigma_e^2}P_0(y,\boldsymbol{x})dyd\boldsymbol{x}\\
&=\int_{\boldsymbol{x}\in[0,1]^p}\frac{2\eta_0^2(\boldsymbol{x})-2\eta_0(\boldsymbol{x})\eta_{\boldsymbol{\theta}}(\boldsymbol{x})-\eta_0^2(\boldsymbol{x})+\eta_{\boldsymbol{\theta}}^2(\boldsymbol{x})}{2\sigma_e^2}d\boldsymbol{x}\\
&=\int_{\boldsymbol{x}\in[0,1]^p}\frac{(\eta_0(\boldsymbol{x})-\eta_{\boldsymbol{\theta}}(\boldsymbol{x}))^2}{2}d\boldsymbol{x}=\frac{1}{2}||\eta_0-\eta_{\boldsymbol{\theta}}||_2^2
\end{aligned}\tag{A.8}$$

where, $\sigma_e^2=1$ can be chosen w.l.o.g.

Next, let $\eta_{\boldsymbol{\theta}^*}(\boldsymbol{x})$ be $\boldsymbol{\theta}^*$ satisfying $\arg\min_{\eta_{\boldsymbol{\theta}}\in\mathcal{F}(L,\boldsymbol{k},\boldsymbol{s},\boldsymbol{B})}|\eta_{\boldsymbol{\theta}}-\eta_0|_\infty^2$. Then,

$$||\eta_{\boldsymbol{\theta}^*}-\eta_0||_1\leq||\eta_{\boldsymbol{\theta}^*}-\eta_0||_\infty=\sqrt{\xi}\tag{A.9}$$

Here, we redefine $\overline{\boldsymbol{\delta}}_l$ by considering the L_1 norms of the rows of $\overline{\boldsymbol{D}}_l=\overline{\boldsymbol{W}}_l-\overline{\boldsymbol{W}}_l^*$ as follows

$$\overline{\boldsymbol{D}}_l=(\overline{\boldsymbol{d}}_{l1}^\top,\cdots,\overline{\boldsymbol{d}}_{lk_{l+1}}^\top)^\top\qquad\overline{\boldsymbol{\delta}}_l=(||\overline{\boldsymbol{d}}_{l1}||_1,\cdots,||\overline{\boldsymbol{d}}_{lk_{l+1}}||_1)$$

Next we define a neighborhood $\mathcal{M}_{\sqrt{\sum r_l}}$ as follows:

$$\mathcal{M}_{\sqrt{\sum r_l}} = \left\{ \boldsymbol{\theta} : ||\overline{\boldsymbol{d}}_{li}||_1 \leq \frac{\sqrt{\sum r_l} B_l}{(L+1)(\prod_{j=0}^{L} B_j)}, i \in \mathcal{S}_l, ||\overline{\boldsymbol{d}}_{li}||_1 = 0, i \in \mathcal{S}_l^c, l = 0, \cdots, L \right\}$$

where $\mathcal{S}_l^c$ is the set where $||\overline{\boldsymbol{w}}_{li}^*||_1 = 0$, $l = 0, \cdots, L$. Then, for every $\boldsymbol{\theta} \in \mathcal{M}_{\sqrt{\sum r_l}}$ using (A.3), we have

$$||\eta_{\boldsymbol{\theta}} - \eta_{\boldsymbol{\theta}^*}||_1 \leq \sqrt{\sum r_l} \tag{A.10}$$

Combining (A.9) and (A.10), we get for $\boldsymbol{\theta} \in \mathcal{M}_{\sqrt{\sum r_l}}$, $||\eta_{\boldsymbol{\theta}} - \eta_0||_1 \leq \sqrt{\sum r_l} + \sqrt{\xi}$. So we get,

$$d_{\mathrm{KL}}(P_0, P_{\boldsymbol{\theta}}) \leq \frac{(\sqrt{\sum r_l} + \sqrt{\xi})^2}{2} \leq \sum r_l + \xi$$

Since $\boldsymbol{\theta} \in \mathcal{N}_{\sum r_l + \xi}$ for every $\boldsymbol{\theta} \in \mathcal{M}_{\sqrt{\sum r_l}}$; therefore,

$$\int_{\boldsymbol{\theta} \in \mathcal{N}_{\sum r_l + \xi}} \widetilde{\pi}(\boldsymbol{\theta}) d\boldsymbol{\theta} \geq \int_{\boldsymbol{\theta} \in \mathcal{M}_{\sqrt{\sum r_l}}} \widetilde{\pi}(\boldsymbol{\theta}) d\boldsymbol{\theta}$$

Let $\delta_n = (\sqrt{\sum r_l} B_l)/((L+1)(\prod_{j=0}^{L} B_j))$ and $A = \{\overline{\boldsymbol{w}}_{li} : \ ||\overline{\boldsymbol{w}}_{li} - \overline{\boldsymbol{w}}_{li}^*||_1 \leq \delta_n\}$

$$\begin{aligned}
\widetilde{\Pi}\left(\mathcal{M}_{\sqrt{\sum r_l}}\right) &= \sum_{\boldsymbol{z}} \Pi\left(\mathcal{M}_{\sqrt{\sum r_l}} \middle| \boldsymbol{z}\right) \pi(\boldsymbol{z}) \\
&\geq \sum_{\{\boldsymbol{z}: z_{li}=1, i \in \mathcal{S}_l, z_{li}=0, i \in \mathcal{S}_l^c, l=0, \cdots, L\}} \Pi\left(\mathcal{M}_{\sqrt{\sum r_l}} \middle| \boldsymbol{z}\right) \pi(\boldsymbol{z}) \\
&= \prod_{l=0}^{L} (1-\lambda_l)^{k_{l+1}-s_l} \lambda_l^{s_l} \prod_{i \in S_l} \mathbb{E}(\mathbb{1}_{\{\overline{\boldsymbol{w}}_{li} \in A\}} | z_{li} = 1) \\
&\geq \prod_{l=0}^{L} (1-\lambda_l)^{k_{l+1}-s_l} \lambda_l^{s_l} \prod_{i \in \mathcal{S}_l} \int_{\overline{\boldsymbol{w}}_{li} \in A} \left(\frac{1}{2\pi}\right)^{\frac{k_l+1}{2}} \prod_{j=1}^{k_l+1} \exp\left(-\frac{\overline{w}_{lij}^2}{2}\right) d\overline{w}_{lij} \\
&\geq \prod_{l=0}^{L} (1-\lambda_l)^{k_{l+1}-s_l} \lambda_l^{s_l} \prod_{i \in \mathcal{S}_l} \left(\frac{1}{2\pi}\right)^{\frac{k_l+1}{2}} \prod_{j=1}^{k_l+1} \int_{\overline{w}_{lij}^* - \frac{\delta_n}{k_l+1}}^{\overline{w}_{lij}^* + \frac{\delta_n}{k_l+1}} \exp\left(-\frac{\overline{w}_{lij}^2}{2}\right) d\overline{w}_{lij} \\
&= \prod_{l=0}^{L} (1-\lambda_l)^{k_{l+1}-s_l} \lambda_l^{s_l} \prod_{i \in \mathcal{S}_l} \left(\frac{1}{2\pi}\right)^{\frac{k_l+1}{2}} \prod_{j=1}^{k_l+1} \frac{2\delta_n}{k_l+1} \exp\left(-\frac{\widehat{w}_{lij}^2}{2}\right)
\end{aligned}$$

where the third equality follows since $\mathbb{E}(\mathbb{1}_{\{\overline{\boldsymbol{w}}_{li} \in A\}} | z_{li} = 0) = 1$ since $||\overline{\boldsymbol{w}}_{li}^*||_1 = 0$, for $i \in \mathcal{S}_l^c$. The last equality is by mean value theorem, $\widehat{w}_{lij} \in [\overline{w}_{lij}^* - \delta_n/(k_l+1), \overline{w}_{lij}^* + \delta_n/(k_l+1)]$, thus

$$= \prod_{l=0}^{L} (1-\lambda_l)^{k_{l+1}-s_l} \lambda_l^{s_l} \prod_{i \in \mathcal{S}_l} \exp\left(\frac{k_l+1}{2} \log \frac{1}{2\pi} + (k_l+1) \log \frac{2\delta_n}{k_l+1} - \sum_{j=1}^{k_l+1} \frac{\widehat{w}_{lij}^2}{2}\right)$$

$$= \exp\Bigg[- \sum_{l=0}^{L} \Big\{ s_l \log\left(\frac{1}{\lambda_l}\right) + (k_{l+1} - s_l) \log\left(\frac{1}{1-\lambda_l}\right) + \sum_{i \in \mathcal{S}_l} \left(- \frac{k_l+1}{2} \log \frac{1}{2\pi} - (k_l+1) \log \frac{2\delta_n}{k_l+1} + \sum_{j=1}^{k_l+1} \frac{\widehat{w}_{lij}^2}{2} \right) \Big\} \Bigg]$$

$$= \exp\Bigg[- \sum_{l=0}^{L} \Bigg\{ s_l \log\left(\frac{1}{\lambda_l}\right) + (k_{l+1} - s_l) \log\left(\frac{1}{1-\lambda_l}\right) - \frac{s_l(k_l+1)}{2} \log \frac{1}{2\pi} - s_l(k_l+1) \log \frac{2\delta_n}{k_l+1} + \sum_{i \in \mathcal{S}_l} \sum_{j=1}^{k_l+1} \frac{\widehat{w}_{lij}^2}{2} \Bigg\} \Bigg] \tag{A.11}$$

Now,

$$\begin{aligned} \sum_{l=0}^{L} \sum_{i \in \mathcal{S}_l} \sum_{j=1}^{k_l+1} \frac{\widehat{w}_{lij}^2}{2} &\leq \frac{1}{2} \sum_{l=0}^{L} \sum_{i \in \mathcal{S}_l} \sum_{j=1}^{k_l+1} \max((\overline{w}_{lij}^* - \delta_n/(k_l+1))^2, (\overline{w}_{lij}^* + \delta_n/(k_l+1))^2) \\ &\leq \sum_{l=0}^{L} \sum_{i \in \mathcal{S}_l} \sum_{j=1}^{k_l+1} (\overline{w}_{lij}^{*2} + \delta_n^2/(k_l+1)^2) \leq \sum_{l=0}^{L} \sum_{i \in \mathcal{S}_l} ||\overline{\boldsymbol{w}}_{li}^*||_1^2 + \sum_{l=0}^{L} \sum_{i \in \mathcal{S}_l} \delta_n^2/(k_l+1) \\ &\leq \sum_{l=0}^{L} s_l(B_l^2+1) \leq n \sum r_l \leq n \left(\sum r_l + \xi \right) \end{aligned} \tag{A.12}$$

where the above line uses $\delta_n \to 0$. Finally

$$\begin{aligned} &\sum_{l=0}^{L} \left(s_l \log\left(\frac{1}{\lambda_l}\right) + (k_{l+1} - s_l) \log\left(\frac{1}{1-\lambda_l}\right) - \frac{s_l(k_l+1)}{2} \log \frac{1}{2\pi} - s_l(k_l+1) \log \frac{2\delta_n}{k_l+1} \right) \\ &\leq \sum_{l=0}^{L} \left(Cnr_l + \frac{s_l(k_l+1)}{2} \left\{ 2\log(k_l+1) + 2\log(L+1) + 2 \sum_{m=0, m \neq l}^{L} \log B_m - \log \sum r_l \right\} \right) \\ &\leq Cn \sum r_l \leq Cn \left(\sum r_l + \xi \right) \end{aligned} \tag{A.13}$$

where the first inequality follows from (A.7) and expanding δ_n. The last inequality follows since $n \sum r_l \to \infty$ which implies $-\log \sum r_l = O(\log n)$. Combining (A.12) and (A.13) and replacing (A.11), the proof follows. □

Proof of Lemma 3.4.3 part 2.

$$\textit{Assumption:} \quad -\log \lambda_l = O\{(k_l+1)\vartheta_l\}, \; -\log(1-\lambda_l) = O\{(s_l/k_{l+1})(k_l+1)\vartheta_l\}$$

Suppose there exists $q \in \mathcal{Q}^{\mathbf{MF}}$ such that

$$d_{\mathrm{KL}}(q, \pi) \leq C_1 n \sum r_l,$$
$$\sum_{\boldsymbol{z}} \int_{\boldsymbol{\Theta}} |\eta_{\boldsymbol{\theta}} - \eta_{\boldsymbol{\theta}^*}|_2^2 \, q(\boldsymbol{\theta}, \boldsymbol{z}) d\boldsymbol{\theta} \leq \sum r_l. \tag{A.14}$$

Recall $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\theta}(L,p,\boldsymbol{s},\boldsymbol{B})} |\eta_{\boldsymbol{\theta}} - \eta_0|_\infty^2$. By relation (A.8),

$$\begin{aligned}
\sum_{\boldsymbol{z}} \int n d_{\mathrm{KL}}(P_0, P_{\boldsymbol{\theta}}) q(\boldsymbol{\theta}, \boldsymbol{z}) d\boldsymbol{\theta} &= \sum_{\boldsymbol{z}} \frac{n}{2} \int ||\eta_0 - \eta_{\boldsymbol{\theta}}||_2^2 q(\boldsymbol{\theta}, \boldsymbol{z}) d\boldsymbol{\theta} \\
&\leq \frac{n}{2} \sum_{\boldsymbol{z}} \int ||\eta_{\boldsymbol{\theta}^*} - \eta_{\boldsymbol{\theta}}||_2^2 q(\boldsymbol{\theta}, \boldsymbol{z}) d\boldsymbol{\theta} + \frac{n}{2} ||\eta_{\boldsymbol{\theta}^*} - \eta_0||_\infty^2 \\
&\leq Cn(\sum r_l + \xi)
\end{aligned}$$

where the above relation is due to (A.14) which completes the proof.

We next construct $q \in \mathcal{Q}^{\mathbf{MF}}$ as

$$\overline{w}_{lij} | z_{li} \sim z_{li} \mathcal{N}(\overline{w}_{lij}^*, \sigma_l^2) + (1 - z_{li}) \delta_0, \qquad z_{li} \sim \mathrm{Bern}(\gamma_{li}^*) \qquad \gamma_{li}^* = \mathbb{1}(||\boldsymbol{w}_{li}^*||_1 \neq 0)$$

where $\sigma_l^2 = \frac{s_l}{8n(L+1)} (4^{L-l}(k_l + 1) \log(k_{l+1} 2^{k_l+1}) \prod_{m=0, m \neq l}^{L} B_m^2)^{-1}$.

We next consider the relation (A.5) in Lemma A.2.5.

We upper bound the expectation of the supremum of L_1 norm of multivariate Gaussian variables:

$$\int \widetilde{W}_l q(\boldsymbol{\theta}, \boldsymbol{z}) d\boldsymbol{\theta} \leq \int \sup_i ||\overline{\boldsymbol{w}}_{li} - \overline{\boldsymbol{w}}_{li}^*||_1 q(\boldsymbol{\theta} | \boldsymbol{z}) d\boldsymbol{\theta} \leq \int \sup_i ||\overline{\boldsymbol{w}}_{li} - \overline{\boldsymbol{w}}_{li}^*||_1 q(\boldsymbol{\theta} | \boldsymbol{z} = \mathbf{1}) d\boldsymbol{\theta}$$

since $q(\boldsymbol{z}) \leq 1$. If $z_{li} = 1$, then $||\overline{\boldsymbol{w}}_{li} - \overline{\boldsymbol{w}}_{li}^*||_1 = 0$, thus the above integral is maximized at $\boldsymbol{z} = \mathbf{1}$ where $\boldsymbol{z} = \mathbf{1}$ indicates all neurons are present in the network. In this case, all w_{lij} are nothing but independent Gaussian random variables. In this direction we make use of concentration inequalities similar to the proof of theorem 2 in Chérief-Abdellatif (2020). Let, $Y = \sup_i ||\overline{\boldsymbol{w}}_{li} - \overline{\boldsymbol{w}}_{li}^*||_1$.

$$\begin{aligned}
\exp(t \mathbb{E} Y) \leq \mathbb{E}(\exp(tY)) &= \mathbb{E}[\sup_i \exp(t ||\overline{\boldsymbol{w}}_{li} - \overline{\boldsymbol{w}}_{li}^*||_1)] \\
&\leq \sum_{i=1}^{k_{l+1}} \mathbb{E}[\exp(t \sum_{j=1}^{k_l+1} |\overline{w}_{lij} - \overline{w}_{lij}^*|)] = \sum_{i=1}^{k_{l+1}} \prod_{j=1}^{k_l+1} \mathbb{E}[\exp(t |\overline{w}_{lij} - \overline{w}_{lij}^*|)]
\end{aligned}$$

$$= \sum_{i=1}^{k_{l+1}} \prod_{j=1}^{k_l+1} 2\exp\left[\frac{\sigma_l^2 t^2}{2}\right] \Phi(\sigma_l t) \leq k_{l+1} 2^{k_l+1} \exp\left[(k_l+1)\frac{\sigma_l^2 t^2}{2}\right]$$

Thus, $\mathbb{E}Y \leq (\log(k_{l+1}2^{k_l+1}) + (k_l+1)\sigma_l^2 t^2/2)/t$. Let $t = (1/\sigma_l)\sqrt{(2/(k_l+1))\log(k_{l+1}2^{k_l+1})}$,

$$\begin{aligned}\mathbb{E}Y &\leq \sigma_l \sqrt{\frac{k_l+1}{2}} \left[\sqrt{\log(k_{l+1}2^{k_l+1})} + \sqrt{\log(k_{l+1}2^{k_l+1})}\right] \\ &= \sqrt{2\sigma_l^2(k_l+1)\log(k_{l+1}2^{k_l+1})} \leq \sqrt{4\sigma_l^2(k_l+1)\log(k_{l+1}2^{k_l+1})}\end{aligned}$$

Similarly,

$$\int \widetilde{W}_l^2 q(\boldsymbol{\theta}, \boldsymbol{z}) d\boldsymbol{\theta} = \int \sup_i (||\overline{\boldsymbol{w}}_{li} - \overline{\boldsymbol{w}}_{li}^*||_1)^2 q(\boldsymbol{\theta}, \boldsymbol{z}) d\boldsymbol{\theta} \leq \int \sup_i (||\overline{\boldsymbol{w}}_{li} - \overline{\boldsymbol{w}}_{li}^*||_1)^2 q(\boldsymbol{\theta}|\boldsymbol{z} = \mathbf{1})$$

Let, $Y' = \sup_i (||\overline{\boldsymbol{w}}_{li} - \overline{\boldsymbol{w}}_{li}^*||_1)^2$.

$$\begin{aligned}\exp(t\mathbb{E}Y') &\leq \mathbb{E}(\exp(tY')) = \mathbb{E}[\sup_i \exp(t(||\overline{\boldsymbol{w}}_{li} - \overline{\boldsymbol{w}}_{li}^*||_1)^2)] \\ &\leq \sum_{i=1}^{k_{l+1}} \mathbb{E}[\exp(t(\sum_{j=1}^{k_l+1} |\overline{w}_{lij} - \overline{w}_{lij}^*|)^2)] \leq \sum_{i=1}^{k_{l+1}} \mathbb{E}[\exp(t(k_l+1)\sum_{j=1}^{k_l+1} (\overline{w}_{lij} - \overline{w}_{lij}^*)^2)] \\ &= \sum_{i=1}^{k_{l+1}} \prod_{j=1}^{k_l+1} \mathbb{E}[\exp(t(k_l+1)(\overline{w}_{lij} - \overline{w}_{lij}^*)^2)] = \sum_{i=1}^{k_{l+1}} \prod_{j=1}^{k_l+1} \left(\frac{1}{1-2t(k_l+1)\sigma_l^2}\right)^{\frac{1}{2}} \\ &\leq k_{l+1} \left(\frac{1}{1-2t(k_l+1)\sigma_l^2}\right)^{\frac{k_l+1}{2}}\end{aligned}$$

Thus, $\mathbb{E}Y' \leq (\log k_{l+1} - ((k_l+1)/2)\log(1-2t(k_l+1)\sigma_l^2))/t$. Let $t = 1/(4\sigma_l^2(k_l+1))$,

$$\begin{aligned}\mathbb{E}Y' &\leq 4\sigma_l^2(k_l+1)\left[\log k_{l+1} + \left(\frac{k_l+1}{2}\right)\log 2\right] = 4\sigma_l^2(k_l+1)\log(k_{l+1}2^{\frac{k_l+1}{2}}) \\ &\leq 4\sigma_l^2(k_l+1)\log(k_{l+1}2^{k_l+1})\end{aligned}$$

Next we also get,

$$\int (\widetilde{W}_l + B_l) q(\boldsymbol{\theta}, \boldsymbol{z}) d\boldsymbol{\theta} = \int \widetilde{W}_l q(\boldsymbol{\theta}, \boldsymbol{z}) d\boldsymbol{\theta} + B_l \leq \sqrt{4\sigma_l^2(k_l+1)\log(k_{l+1}2^{k_l+1})} + B_l \leq 2B_l$$

$$\begin{aligned}&\int (\widetilde{W}_l + B_l)^2 q(\boldsymbol{\theta}, \boldsymbol{z}) d\boldsymbol{\theta} = \int \widetilde{W}_l^2 q(\boldsymbol{\theta}, \boldsymbol{z}) d\boldsymbol{\theta} + 2B_l \int \widetilde{W}_l q(\boldsymbol{\theta}, \boldsymbol{z}) d\boldsymbol{\theta} + B_l^2 \\ &\leq 4\sigma_l^2(k_l+1)\log(k_{l+1}2^{k_l+1}) + 2B_l\sqrt{4\sigma_l^2(k_l+1)\log(k_{l+1}2^{k_l+1})} + B_l^2 \leq 4B_l^2\end{aligned}$$

$$\int \widetilde{W}_l(\widetilde{W}_l + B_l)q(\boldsymbol{\theta}, \boldsymbol{z})d\boldsymbol{\theta} = \int \widetilde{W}_l^2 q(\boldsymbol{\theta}, \boldsymbol{z})d\boldsymbol{\theta} + B_l \int \widetilde{W}_l q(\boldsymbol{\theta}, \boldsymbol{z})d\boldsymbol{\theta}$$
$$\leq 4\sigma_l^2(k_l+1)\log(k_{l+1}2^{k_l+1}) + B_l\sqrt{4\sigma_l^2(k_l+1)\log(k_{l+1}2^{k_l+1})}$$
$$\leq \sqrt{4\sigma_l^2(k_l+1)\log(k_{l+1}2^{k_l+1})}\left(\sqrt{4\sigma_l^2(k_l+1)\log(k_{l+1}2^{k_l+1})} + B_l\right)$$
$$\leq 2B_l\sqrt{4\sigma_l^2(k_l+1)\log(k_{l+1}2^{k_l+1})}$$

since $\sqrt{4\sigma_l^2(k_l+1)\log(k_{l+1}2^{k_l+1})}$ is bounded above by

$$\sqrt{\frac{4s_l}{8n(L+1)}\left(4^{L-l}(k_l+1)\log(k_{l+1}2^{k_l+1})\prod_{m=0,m\neq l}^{L} B_m^2\right)^{-1}(k_l+1)\log(k_{l+1}2^{k_l+1})}$$
$$= B_l\sqrt{\frac{s_l}{2n(L+1)}\left(4^{L-l}\prod_{m=0}^{L} B_m^2\right)^{-1}} \leq B_l, \text{ The quantity in square root} < 1 \text{ for large } n.$$

Let $b_j = (k_j+1)\log(k_{j+1}2^{k_j+1})$. From relation (A.5), we get

$$\int ||\eta_{\boldsymbol{\theta}} - \eta_{\boldsymbol{\theta}^*}||_2^2 q(\boldsymbol{\theta}, \boldsymbol{z})d\boldsymbol{\theta} \leq \sum_{j=0}^{L} c_{j-1}^2(4\sigma_j^2 b_j)\left(\prod_{m=j+1}^{L} 4B_m^2\right)$$
$$+ 2\sum_{j=0}^{L}\sum_{j'=0}^{j-1} c_{j-1}c_{j'-1}2B_j\sqrt{4\sigma_j^2 b_j}\left(\prod_{m=j+1}^{L} 4B_m^2\right)\sqrt{4\sigma_{j'}^2 b_{j'}}\left(\prod_{m=j'+1}^{j-1} 2B_m\right)$$
$$= 4\sum_{j=0}^{L} 4^{L-j}\sigma_j^2 b_j\left(\prod_{m=0}^{j-1} B_m^2\right)\left(\prod_{m=j+1}^{L} B_m^2\right)$$
$$+ 8\sum_{j=0}^{L}\sum_{j'=0}^{j-1}\left(\prod_{m=0}^{j-1} B_m\right)\left(\prod_{m=0}^{j-1} B_m\right)2B_j\left(\prod_{m=j+1}^{L} 4B_m^2\right)\left(\prod_{m=j'+1}^{j-1} 2B_m\right)\sqrt{\sigma_j^2 b_j}\sqrt{\sigma_{j'}^2 b_{j'}}$$
$$= 4\sum_{j=0}^{L} 2^{2L-2j}\sigma_j^2 b_j \prod_{m=0,m\neq j}^{L} B_m^2$$
$$+ 8\sum_{j=0}^{L}\sum_{j'=0}^{j-1} 4^{L-j}2^{j-j'}\left(\prod_{m=0}^{j-1} B_m\right)\left(\prod_{m=0}^{j-1} B_m\right)\left(\prod_{m=j+1}^{L} B_m\right)\left(\prod_{m=j'+1}^{L} B_m\right)\sqrt{\sigma_j^2 b_j}\sqrt{\sigma_{j'}^2 b_{j'}}$$
$$= 4\sum_{j=0}^{L} 2^{2L-2j}\sigma_j^2 b_j\left(\prod_{m=0,m\neq j}^{L} B_m^2\right)$$
$$+ 8\sum_{j=0}^{L}\sum_{j'=0}^{j-1} 2^{L-j}2^{L-j'}\left(\prod_{m=0,m\neq j}^{L} B_m\right)\left(\prod_{m=0,m\neq j'}^{L} B_m\right)\sqrt{\sigma_j^2 b_j}\sqrt{\sigma_{j'}^2 b_{j'}}$$
$$= 4\left(\sum_{j=0}^{L} 2^{L-j}\sqrt{\sigma_j^2 b_j}\left(\prod_{m=0,m\neq j}^{L} B_m\right)\right)^2 = 4\left(\sum_{j=0}^{L}\sqrt{\frac{s_j}{8n(L+1)}}\right)^2$$

$$= \frac{1}{2n(L+1)}\left(\sum_{j=0}^{L}\sqrt{s_j}\right)^2 \leq \frac{\sum_{j=0}^{L} s_j}{2n} \leq \sum_{j=0}^{L} r_l$$

This concludes the proof of (A.14). Next,

$$\begin{aligned}
d_{\mathrm{KL}}(q,\pi) &\leq \log\frac{1}{\pi(\boldsymbol{z})} + \mathbb{1}(\boldsymbol{z}=\boldsymbol{\gamma}^*) d_{\mathrm{KL}}\Bigg(\Big\{\prod_{l=0}^{L-1}\prod_{i=1}^{k_{l+1}}\prod_{j=1}^{k_l+1}\Big\{\gamma_{li}^*\mathcal{N}(\overline{w}_{lij}^*,\sigma_l^2)+(1-\gamma_{li}^*)\delta_0\Big\} \\
&\quad \prod_{j=1}^{k_L+1}\mathcal{N}(\overline{w}_{Lj}^*,\sigma_L^2)\Big\}, \Big\{\prod_{l=0}^{L-1}\prod_{i=1}^{k_{l+1}}\prod_{j=1}^{k_l+1}\Big\{z_{li}\mathcal{N}(0,\sigma_0^2)+(1-z_{li})\delta_0\Big\}\prod_{j=1}^{k_L+1}\mathcal{N}(0,\sigma_0^2)\Big\}\Bigg) \\
&= \log\frac{1}{\prod_{l=0}^{L-1}\lambda_l^{s_l}(1-\lambda_l)^{k_{l+1}-s_l}} + \sum_{l=0}^{L-1}\sum_{i=1}^{k_{l+1}}\sum_{j=1}^{k_l+1} d_{\mathrm{KL}}\Big(\gamma_{li}^*\mathcal{N}(\overline{w}_{lij}^*,\sigma_l^2)+(1-\gamma_{li}^*)\delta_0, \\
&\quad \gamma_{li}^*\mathcal{N}(0,\sigma_0^2)+(1-\gamma_{li}^*)\delta_0\Big) + \sum_{j=1}^{k_L+1} d_{\mathrm{KL}}\Big(\mathcal{N}(\overline{w}_{Lj}^*,\sigma_L^2),\mathcal{N}(0,\sigma_0^2)\Big) \\
&= \sum_{l=0}^{L-1}\left(s_l\log\frac{1}{\lambda_l}+(k_{l+1}-s_l)\log\frac{1}{1-\lambda_l}\right) + \sum_{l=0}^{L-1}\sum_{i=1}^{k_{l+1}}\sum_{j=1}^{k_l+1}\gamma_{li}^*\left\{\frac{1}{2}\log\frac{\sigma_0^2}{\sigma_l^2}+\frac{\sigma_l^2+{\overline{w}_{lij}^*}^2}{2\sigma_0^2}-\frac{1}{2}\right\} \\
&\quad + \sum_{j=1}^{k_L+1}\left\{\frac{1}{2}\log\frac{\sigma_0^2}{\sigma_L^2}+\frac{\sigma_L^2+{\overline{w}_{Lj}^*}^2}{2\sigma_0^2}-\frac{1}{2}\right\} \\
&\leq \sum_{l=0}^{L-1} Cnr_l + \sum_{l=0}^{L-1}\frac{s_l k_l + s_l}{2}\left[\frac{\sigma_l^2}{\sigma_0^2}+\frac{B_l^2}{\sigma_0^2(k_l+1)}-1+\log\frac{\sigma_0^2}{\sigma_l^2}\right] \\
&\qquad + \frac{k_L+1}{2}\left[\frac{\sigma_L^2}{\sigma_0^2}\frac{B_L^2}{\sigma_0^2(k_L+1)}-1+\log\frac{\sigma_0^2}{\sigma_L^2}\right]
\end{aligned}$$

where the first inequality follows from Lemma A.2.2. The inequality in the above line uses $\sum_{j=1}^{k_l+1}{\overline{w}_{lij}^*}^2 \leq B_l^2$ and similar to the proof of Lemma 4.1 in Bai et al. (2020) uses (A.7).

Let $\sigma_0^2 = 1$ and it could be easily derived that $\sigma_l^2 \leq 1$.

$$\begin{aligned}
d_{\mathrm{KL}}(q,\pi) &\leq \sum_{l=0}^{L-1} Cnr_l + \sum_{l=0}^{L-1}\frac{s_l}{2}(k_l+1)\left[\frac{B_l^2}{k_l+1}-\log\sigma_l^2\right] + \frac{(k_L+1)}{2}\left[\frac{B_L^2}{k_L+1}-\log\sigma_L^2\right] \\
&= \sum_{l=0}^{L-1} Cnr_l + \sum_{l=0}^{L-1}\frac{s_l}{2}(k_l+1)\left[\frac{B_l^2}{k_l+1}-\log\left(\frac{s_l}{8n(L+1)}\left[4^{L-l}b_l\prod_{m=0,m\neq l}^{L}B_m^2\right]^{-1}\right)\right] \\
&\quad + \frac{(k_L+1)}{2}\left[\frac{B_L^2}{k_L+1}-\log\left(\frac{1}{8n(L+1)}\left[b_L\prod_{m=0,m\neq L}^{L}B_m^2\right]^{-1}\right)\right] \\
&= \sum_{l=0}^{L-1} Cnr_l + \sum_{l=0}^{L}\frac{s_l}{2}(k_l+1)\left[\frac{B_l^2}{k_l+1}-\log\left(\frac{s_l}{8n(L+1)}\left[4^{L-l}b_l\prod_{m=0,m\neq l}^{L}B_m^2\right]^{-1}\right)\right]
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{l=0}^{L-1} Cnr_l + \sum_{l=0}^{L} \frac{s_l}{2} B_l^2 + \sum_{l=0}^{L} \frac{s_l}{2}(k_l+1)\log\left(\frac{8n(L+1)}{s_l}\right) \\
&+ \sum_{l=0}^{L} s_l(k_l+1)(L-l)\log 2 + \sum_{l=0}^{L} \frac{s_l}{2}(k_l+1)\log(k_l+1) \\
&+ \sum_{l=0}^{L} \frac{s_l}{2}(k_l+1)\log\Big(\log(k_{l+1}2^{k_l+1})\Big) + \sum_{l=0}^{L} s_l(k_l+1)\left(\sum_{m=0,m\neq l}^{L} \log B_m\right) \\
&\leq \sum_{l=0}^{L-1} Cnr_l + \sum_{l=0}^{L} \frac{s_l}{2} B_l^2 + \sum_{l=0}^{L} \frac{s_l}{2}(k_l+1)\log\left(\frac{8n(L+1)}{s_l}\right) + L\sum_{l=0}^{L} s_l(k_l+1) \\
&+ \sum_{l=0}^{L} \frac{s_l}{2}(k_l+1)(\log(k_l+1)+\log(k_{l+1}+k_l+1)) + \sum_{l=0}^{L} s_l(k_l+1)\left(\sum_{m=0,m\neq l}^{L} \log B_m\right) \\
&\leq \sum_{l=0}^{L-1} Cnr_l + \sum_{l=0}^{L} \frac{s_l}{2} B_l^2 + \sum_{l=0}^{L} \frac{s_l}{2}(k_l+1)\log\left(\frac{8n(L+1)}{s_l}\right) + L\sum_{l=0}^{L} s_l(k_l+1) \\
&+ \sum_{l=0}^{L} s_l(k_l+1)\log(k_{l+1}+k_l+1) + \sum_{l=0}^{L} s_l(k_l+1)\left(\sum_{m=0,m\neq l}^{L} \log B_m\right) \\
&\leq \sum_{l=0}^{L-1} Cnr_l + \sum_{l=0}^{L} s_l(k_l+1)\left[\frac{B_l^2}{2(k_l+1)} + \left(\sum_{m=0,m\neq l}^{L} \log B_m\right) + L + \log(k_{l+1}+k_l+1)\right. \\
&\left. + \frac{1}{2}\log\left(\frac{8n(L+1)}{s_l}\right)\right] \\
&\leq \sum_{l=0}^{L-1} (C+C')nr_l + C'nr_L \\
&+ \sum_{l=0}^{L} s_l(k_l+1)\left[\frac{B_l^2}{k_l+1} + \left(\sum_{m=0,m\neq l}^{L} \log B_m\right) + L + \log(k_{l+1}+k_l+1) + \log\left(\frac{n}{s_l}\right)\right] \\
&\leq \sum_{l=0}^{L-1} (C+C')nr_l + C'nr_L + \sum_{l=0}^{L} s_l(k_l+1)\vartheta_l \leq C_1 n \sum_{l=0}^{L} r_l
\end{aligned}
$$

This concludes the proof of (A.14). □

Proof of Corollary 3.4.5

The proof is a direct consequence of Theorem 3.4.4 in the Section 3.4 as long as assumptions of Lemma 3.4.2 and Lemma 3.4.3 parts 1 and 2 hold when $\sigma_0^2 = 1$, $-\log\lambda_l = \log(k_{l+1}) + C_l(k_l+1)\vartheta_l$ and $\epsilon_n = \sqrt{(\sum_{l=0}^{L} r_l + \xi)\sum_{l=0}^{L} u_l}$. This what we show next.

Verifying assumption (A.6) *under Proof of Lemma 3.4.2:* Note, $\sum u_l = O(\epsilon_n^2)$, thus

$$\sum u_l \log L = o(n\epsilon_n^2) \iff \log L = o(n(\sum r_l + \xi))$$

which is indeed true since $\log L = o(L^2)$ and $L^2 \leq n \sum r_l$. We show that $(k_{l+1}\lambda_l)/s_l^\circ \to 0$. With $\lambda_l = (1/k_{l+1})\exp(-C_l(k_l+1)\vartheta_l)$,

$$\begin{aligned}\frac{k_{l+1}\lambda_l}{s_l^\circ} &\leq \frac{\sum u_l \exp(-C(k_l+1)\vartheta_l)}{n\epsilon_n^2} = \frac{\exp(-C(k_l+1)\vartheta_l + \log\sum u_l)}{n\epsilon_n^2} \\ &\leq \frac{\exp(-C(k_l+1)\vartheta_l + \vartheta_l)}{n\epsilon_n^2} \to 0\end{aligned}$$

where the above relation holds since $\log \sum u_l \leq \vartheta_l$, $\vartheta_l \to \infty$, $k_l \to \infty$ and $n\epsilon_n^2 \to \infty$.

Verifying assumption (A.7) *under Proof of Lemma 3.4.3 part 1. and part 2.* Note,

$$-\log\lambda_l = \log(k_{l+1}) + C_l(k_l+1)\vartheta_l \leq \vartheta_l + C_l(k_l+1)\vartheta_l = O\{(k_l+1)\vartheta_l\}$$

And then,

$$1 - \lambda_l = 1 - \exp(-C_l\vartheta_l(k_l+1))/k_{l+1}$$

$$-\log(1-\lambda_l) \sim \exp(-C_l\vartheta_l(k_l+1))/k_{l+1} = O\{(k_l+1)s_l\vartheta_l/k_{l+1}\}$$

since $\exp(-C_l\vartheta_l(k_l+1)) \to 0$ and $(k_l+1)s_l\vartheta_l \to \infty$. □

APPENDIX B

ADDITIONAL NUMERICAL EXPERIMENTS DETAILS

B.1 FLOPs Calculation

We only count multiply operation for floating point operations (FLOPs) similar to Zhao et al. (2019). In 2D convolution layer, we assume convolution is implemented as a sliding window and that the nonlinearity function is computed for free. Then, for a 2D convolutional layer (given bias is present) we get FLOPs as:

$$\text{FLOPs} = (C_{in,pruned} K_w K_h + 1) O_w O_h C_{out,pruned}$$

where, $C_{in,pruned}, C_{out,pruned}$ are the number of input channels and output channels after pruning. Channels are pruned if all the parameters associated with that channel in convolution mapping are zero. K_w and K_h are the kernel width and height respectively. Finally, O_w, O_h are output width and height where $O_w = (I_w + 2 \times P_w - D_w \times (K_w - 1) - 1)/S_w + 1$ and $O_h = (I_h + 2 \times P_h - D_h \times (K_h - 1) - 1)/S_h + 1$. Here, I_w, I_h are input, P_w, P_h are padding, D_w, D_h are dilation, S_w, S_h are stride widths and heights respectively.

For fully connected (linear) layers (with bias) we get FLOPs as:

$$\text{FLOPs} = (I_{pruned} + 1) O_{pruned}$$

where, I_{pruned} is the number of pruned input neurons and O_{pruned} is the number of pruned output neurons.

B.2 Variational parameters initialization

We initialize the γ_{lj}'s at a value close to 1 for all of our experiments. This ensures that at epoch 0, we have a fully connected deep neural network. This also warrants that most of the weights do not get pruned off at a very early stage of training which might lead

to bad performance. The variational parameters $\mu_{ljj'}$ are initialized using $U(-0.6, 0.6)$ for simulation and UCI regression examples whereas for classification Kaiming uniform initialization (He et al., 2015) is used. Moreover, $\sigma_{ljj'}$ are reparameterized using softplus function: $\sigma_{ljj'} = \log(1 + \exp(\rho_{ljj'}))$ and $\rho_{ljj'}$ are initialized using a constant value of -6. This keeps initial values of $\sigma_{ljj'}$ close to 0 ensuring that the initial values of network weights stay close to Kaiming uniform initialization.

B.3 Hyperparameters for training

We keep MC sample size (S) to be 1 during training. We choose learning rate of 3×10^{-3}, batch size of 400, and 10000 epochs in the 20 neurons case of simulation study-I. We use learning rate of 10^{-3}, batch size of 400, and 20000 epochs in the 100 neurons case of simulation study-I. Next, we use learning rate of 5×10^{-3}, full batch, and 10000 epochs for simulation study-II. In UCI regression datasets, we choose batch size = 128 and run 500 epochs for *Concrete, Wine, Power Plant*, 800 epochs for *Kin8nm*. For *Protein* and *Year* datasets, we choose batch size of 256 and run 100 epochs. For all the UCI regression datasets we keep learning rate of 10^{-3}. The Adam algorithm is chosen for optimization of model parameters.

In image classification datasets, for SS-IG model, we use 10^{-3} learning rate and minibatch size of 1024 in all experiments except in LeNet-5-Caffe on Fashion-MNIST experiment where we use 2×10^{-3} learning rate and 1024 minibatch size. For SV-BNN model, we take 10^{-3} learning rate and 1024 minibatch size in all experiments after extensive hyperparameter search. For VBNN model, we take learning rate of 10^{-4} and minibatch size of 128 according to Blundell et al. (2015). We train each model for 1200 epochs using Adam optimizer in all the image classification experiments provided in the Section 3.5.

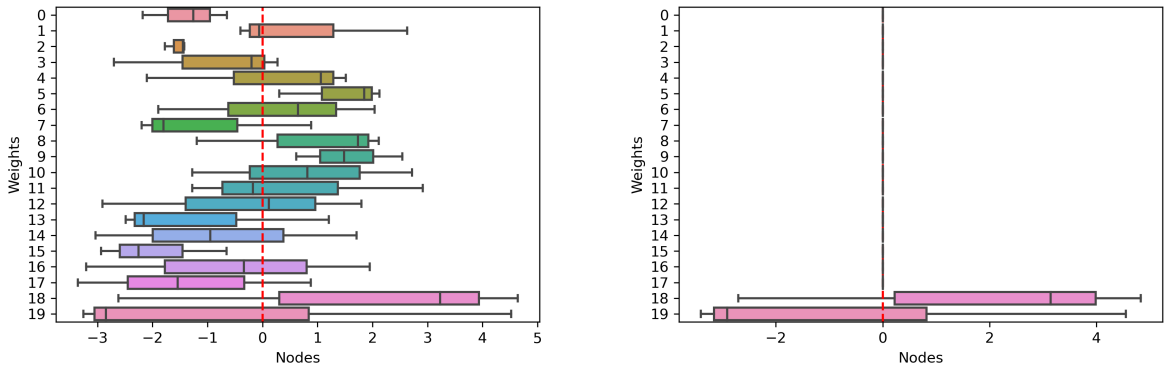
B.4 Fine-tuning of the constant in prior inclusion probability expression

Recall the layer-wise prior inclusion probabilities: $\lambda_l = (1/k_{l+1})\exp(-C_l(k_l + 1)\vartheta_l)$ from the Corollary 3.4.5. In our numerical experiments, we use this expression to choose an

optimal value of λ_l in each layer of a given network. The λ_l varies as we vary our constant C_l and we next describe how is C_l chosen. The influence of C_l is mainly due to the $k_l + 1$ term and ${B_l}^2/(k_l + 1)$ from ϑ_l term. We ensure that each incoming weight and bias onto the node from layer $l + 1$ is bounded by 1 which leads us to choose B_l to be $k_l + 1$. So the leading term from $(k_l + 1)\vartheta_l$ is $(k_l + 1)$ and C_l has to be chosen such that we avoid making exponential term from λ_l expression close to 0. In our experiments we choose C_l values in the negative order of 10 such that prior inclusion probabilities do not fall below 10^{-50}. If we instead choose a λ_l value very close to 0 then we might prune off all the nodes in each layer or might make the training unstable which is not ideal. Overall the aforementioned strategy of choosing C_l constant values ensure reasonable values for the λ_l in each layer.

B.5 Simulation study I: extra details

First we provide the network parameters used to generate the data for this simulation experiment. The edge weights in the underlying 2-2-1 network are as follows: $\boldsymbol{W}_0 = \{w_{011} = 10, w_{012} = 15, w_{021} = -15, w_{022} = 10\}$; $\boldsymbol{W}_1 = \{w_{111} = -3, w_{121} = 3\}$ and $\boldsymbol{v}_0 = \{v_{01} = -5, v_{02} = 5\}$; $\boldsymbol{v}_1 = \{v_{11} = 4\}$.



(a) VBNN (b) SS-IG

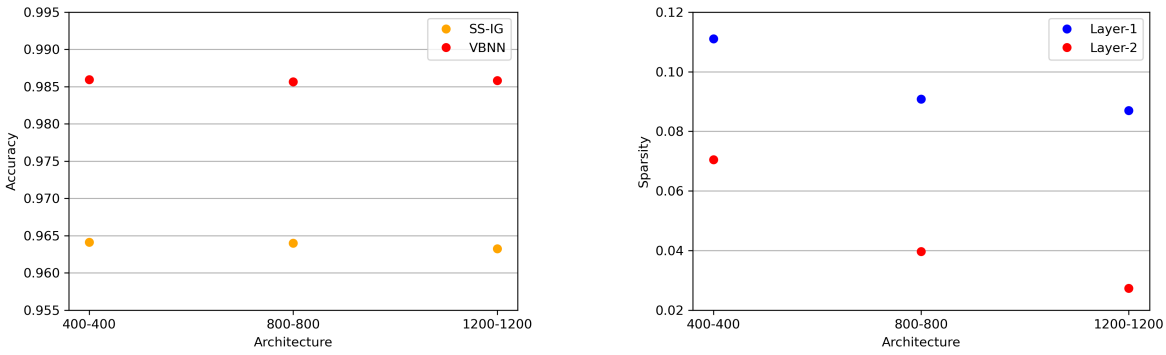
Figure B.1 **Simulation study I: additional experiment results.** Node-wise weight magnitudes recovered by VBNN and proposed SS-IG model in the synthetic regression data generated using 2-2-1 network. The boxplots show the distribution of incoming weights into a given hidden layer node. Only the 20 nodes with the largest edge weights are displayed.

In Figure B.1, we provide additional results demonstrating the model selection ability of our SS-IG approach in a wider network consisting of 100 nodes in the single hidden layer structure considered in the simulation study-I from the Section 3.5.

B.5.1 Effect of Hidden Layer Widths

Here, we explore 2-hidden layer neural networks with varying widths. For our SS-IG model we use 10^{-3} learning rate and minibatch size of 1024 while for VBNN model, we take learning rate of 10^{-4} and minibatch size of 128 according to Blundell et al. (2015). We train both the models for 400 epochs using Adam optimizer.

Figure B.2 summarizes the results. We have provided results for 3 different architectures which have 400, 800, and 1200 nodes each in their 2-hidden layers. In Figure B.2a, we find that across the architectures both SS-IG and VBNN models have similar predictive performance. Further, our method is able to prune off more than 88% of first hidden layer nodes and more than 92% of second hidden layer nodes (Figure B.2b) at the expense of 2% accuracy loss due to sparsification compared to the densely connected VBNN. We also observe that as model capacity increases the sparsity percentage per layer decreases. This suggests that, each architecture is trying to reach a sparse network of comparable size.



(a) Prediction accuracy per architecture (b) Layer-wise sparsity per architecture

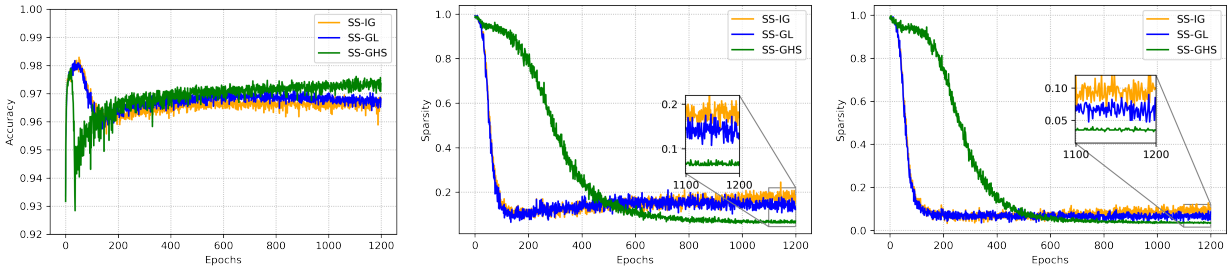
Figure B.2 **MNIST experiment results for varying hidden layer widths.**

CHAPTER 4

COMPACT BAYESIAN NEURAL NETWORKS WITH STRUCTURED SPARSITY

4.1 Introduction

In high-dimensional modeling, predictor selection and sparse signal recovery are routine statistical and machine learning practices. Sparse parameter estimation via high-dimensional regularization penalizing model dimensionality is well studied in the literature. Two of the most popular regularization techniques are lasso and horseshoe regularizers (Bhadra et al., 2019). The lasso estimator (Tibshirani, 1996) induces sparsity by constraining the L_1 norm of the parameters in the model. The horseshoe estimator (Carvalho et al., 2010) places absolutely continuous shrinkage priors on the entire parameter vector that selectively shrinks the small signals since horseshoe prior has heavy tails supporting both zero values and large values. Both Lasso and horseshoe procedures come with strong theoretical guarantees for estimation and prediction. In this work, we propose a spike-and-slab prior framework similar to SS-IG model proposed in Chapter 3 for dynamic node pruning with slab component using either group lasso or group horseshoe priors. This combination of spike-and-slab prior and group shrinkage priors, first ensures that unnecessary collection of weights incident on a node are shrunk to zero and then the spike-and-slab setup allows for automated pruning of such shrunken weights. In Figure 4.1, we provide an image classification experiment where our proposed approaches of spike-and-slab Group Lasso (SS-GL) and spike-and-slab Group Horseshoe (SS-GHS) demonstrate the improvement over simple Gaussian prior in the slab part. In order to conduct posterior approximation, VI in the sparse BNNs with spike-and-slab Gaussian prior framework for edge selection was introduced by Chérief-Abdellatif (2020) and later Jantre et al. (2021a) extended it to node pruning. In this work, we adopt variational Bayesian inference leading to tractable model training in conjunction with continuous



(a) Prediction accuracy (b) Layer-1 node sparsity (c) Layer-2 node sparsity

Figure 4.1 **MNIST experiment results: motivation for group shrinkage priors over Gaussian prior.** Here, we demonstrate the performance of our SS-GL and SS-GHS models in 2-layer perceptron network to classify MNIST, hand-written digits dataset. (a) we plot the classification accuracy on the test data for our models and include (Jantre et al., 2021a)'s SS-Gauss model. (b) and (c) we plot the proportion of active nodes (node sparsity) in the layer-1 and layer-2 of the network respectively. We observe that our SS-GHS yields the most compact network with the best classification accuracy.

relaxation of discrete Bernoulli variables associated with the spike part (Maddison et al., 2017; Jang et al., 2017) similar to SS-IG model.

4.1.1 Proposed Methods

Firstly, there does not exist any cohesive literature which establishes the numerical efficiency of shrinkage priors over Gaussian slabs in the context of training structurally sparse networks. Secondly, the numerical properties of the corresponding variational implementation remain unexplored. To address these issues, we consider a spike-and-slab framework with group shrinkage priors: (i) group lasso and (ii) group horseshoe, which first shrinks the redundant model weights through the slab component and the spike component prunes out the nodes with weights whose values are shrunk close to zero. Accordingly,

Detailed Contribution.

- We propose structurally sparse Bayesian neural networks using two distinct spike and slab prior setups, where the slab component uses hierarchical priors on the group of incoming weights (including bias) on the neurons: (i) **S**pike-and-**S**lab **G**roup **L**asso (**SS-GL**), and (ii) **S**pike-and-**S**lab **G**roup **H**orse**S**hoe (**SS-GHS**).

4.2 Structured Sparsity: Spike-and-Slab Hierarchical Priors

In order to carry out automatic node selection to induce structured sparsity in BNNs, we consider spike-and-slab priors. A zero-mean Gaussian distribution is the commonly used slab distribution in spike-and-slab priors (Jantre et al., 2021a). However, their use can lead to inflated predictive uncertainties, especially when used in conjunction with fully factorized variational inference (Ghosh et al., 2019). Instead, if we consider a slab distribution having zero-mean Gaussian distribution with its scale being a random variable then the slab part of the marginal prior distribution will have heavier tails and higher mass at zero. Such hierarchical distributions in slab part further improve the sparsity as well as circumvent the inflated predictive uncertainties. Below, we describe the two hierarchical spike-and-slab priors and corresponding fully factorized variational family that we use in each of our proposed approaches.

4.2.1 Spike-and-Slab Group Lasso (SS-GL):

To facilitate the optimal layer-wise node selection, we allow the prior inclusion probability λ_l to vary as a function of the layer index l.

Prior: We assume a spike-and-slab prior of the following form with z_{lj} being the indicator for the presence of j^{th} node in the l^{th} layer.

$$\pi(\overline{\boldsymbol{w}}_{lj}|z_{lj}) = \left[(1-z_{lj})\boldsymbol{\delta}_0 + z_{lj}N(0,\sigma_0^2\tau_{lj}^2\boldsymbol{I})\right], \quad \pi(z_{lj}) = \text{Ber}(\lambda_l), \quad \pi(\tau_{lj}^2) = G(k_l/2+1, \varsigma^2/2)$$

where $l = 0,\ldots,L$, $j = 1,\ldots,k_{l+1}$. $N(.,.)$, Ber(.), and $G(.,.)$ represent Gaussian, Bernoulli, and Gamma distributions. $\overline{\boldsymbol{w}}_{lj} = (\overline{w}_{lj1},\ldots,\overline{w}_{ljk_l+1})$ is a vector of edges incident on the j^{th} node in the l^{th} layer. $\boldsymbol{\delta}_0$ is a Dirac spike vector of dimension k_l+1 with all zero entries and $\boldsymbol{I}$ is identity matrix of dimension $k_l+1 \times k_l+1$. z_{lj} with $j = (1,\ldots,k_{l+1})$ all follow $\text{Ber}(\lambda_l)$ to allow for common prior inclusion probability, λ_l, for each node from a given layer l. We set $\lambda_L = 1$ to ensure no node selection occurs in the output layer. σ_0 and τ_{lj} are the

constant global and the variable local (per node) scale mixture components of the Gaussian slab distribution. $\varsigma^2/2$ is the constant rate hyperparameter of the Gamma distribution.

Variational family: We consider the following fully factorized variational family

$$q(\overline{\boldsymbol{w}}_{lj}|z_{lj}) = \left[(1-z_{lj})\boldsymbol{\delta}_0 + z_{lj}N(\boldsymbol{\mu}_{lj}, \text{diag}(\boldsymbol{\sigma}_{lj}^2))\right], \quad q(z_{lj}) = \text{Ber}(\gamma_{lj})$$
$$q(\tau_{lj}^2) = LN(\mu_{lj}^{\{\tau\}}, {\sigma_{lj}^{\{\tau\}}}^2)$$

for $l = 0, \ldots, L$, $j = 1, \ldots, k_{l+1}$. $LN(.,.)$ denotes Log-Normal distribution and $q(\log \tau_{lj}^2) \sim N(\mu_{lj}^{\{\tau\}}, {\sigma_{lj}^{\{\tau\}}}^2)$. The spike-and-slab structure of the variational family ensures that the variational weight distributions follow spike-and-slab structure allowing for exact node sparsity through variational approximation. Further, the weight distributions conditioned on the node indicator variables are all independent of each other. The variational distribution of parameters obtained post optimization will then inherently prune away redundant nodes from each layer. Moreover, we use Log-Normal family instead of Gamma family to approximate Gamma distributed τ_{lj}^2 since we obtain closed form expressions for $d_{\text{KL}}(q(\tau_{lj}^2), \pi(\tau_{lj}^2|\varsigma^2))$.

Additionally, $\boldsymbol{\mu}_{lj} = (\mu_{lj1}, \ldots, \mu_{ljk_l+1})$ and $\boldsymbol{\sigma}_{lj}^2 = (\sigma_{lj1}^2, \ldots, \sigma_{ljk_l+1}^2)$ denote the vectors of variational mean and standard deviation parameters of the slab component of $q(\overline{\boldsymbol{w}}_{lj}|z_{lj})$. $\text{diag}(\boldsymbol{\sigma}_{lj}^2)$ is the diagonal matrix with $\sigma_{ljj'}^2$ being the j'^{th} diagonal entry. Similarly, γ_{lj} denotes the variational inclusion probability parameter of $q(z_{lj})$. We set $\gamma_{Lj} = 1$ to ensure no node selection occurs in the output layer. $\mu_{lj}^{\{\tau\}}$ and ${\sigma_{lj}^{\{\tau\}}}^2$ denote the variational mean and standard deviation parameters of the Gaussian distribution associated with $q(\log \tau_{lj}^2)$.

ELBO: Let $\boldsymbol{\theta}$ be the network weights and $\boldsymbol{\varpi} = (\boldsymbol{z}, \boldsymbol{\tau}^2)$ be the remaining parameters. We minimize the loss function: $\mathcal{L} = -\text{ELBO}(q(\boldsymbol{\theta}, \boldsymbol{\varpi}), \pi(\boldsymbol{\theta}, \boldsymbol{\varpi}|\mathcal{D}))$,

$$\begin{aligned}
\mathcal{L} &= -\mathbb{E}_{q(\boldsymbol{\theta},\boldsymbol{\varpi})}[\log L(\boldsymbol{\theta})] + \sum_{l,j} q(\boldsymbol{z}_{lj} = 1) \int d_{\text{KL}}(q(\overline{\boldsymbol{w}}_{lj}|z_{lj} = 1), \pi(\overline{\boldsymbol{w}}_{lj}|\tau_{lj}^2, z_{lj} = 1)) q(\tau_{lj}^2) d\tau_{lj}^2 \\
&\quad + \sum_{l,j} d_{\text{KL}}(q(z_{lj}), \pi(z_{lj})) + \sum_{l,j} d_{\text{KL}}(q(\tau_{lj}^2), \pi(\tau_{lj}^2|\varsigma^2)) \\
&= -\mathbb{E}_{q(\boldsymbol{\theta},\boldsymbol{\varpi})}[\log L(\boldsymbol{\theta})] + \sum_{l,j} q(\boldsymbol{z}_{lj} = 1) \int d_{\text{KL}}(N(\boldsymbol{\mu}_{lj}, \text{diag}(\boldsymbol{\sigma}_{lj}^2)), N(0, \sigma_0^2 \tau_{lj}^2 \boldsymbol{I})) q(\tau_{lj}^2) d\tau_{lj}^2
\end{aligned}$$

$$+\sum_{l,j} d_{\mathrm{KL}}(\mathrm{Ber}(\gamma_{lj}),\mathrm{Ber}(\lambda_l))+\sum_{l,j} d_{\mathrm{KL}}\left(LN(\mu_{lj}^{\{\tau\}},\sigma_{lj}^{2\{\tau\}}),G(k_l/2+1,\varsigma^2/2)\right)$$

4.2.2 Spike-and-Slab Group Horseshoe (SS-GHS):

In this model, we consider spike-and-slab prior with group horseshoe distribution in the slab part.

Prior: We consider regularized version of group horseshoe (Piironen and Vehtari, 2017) in the slab part to circumvent the numerical stability issues associated with the unregularized group horseshoe. We define our prior similar to SS-GL earlier.

$$\pi(\overline{\boldsymbol{w}}_{lj}|z_{lj})=\left[(1-z_{lj})\boldsymbol{\delta}_0+z_{lj}N(0,\sigma_0^2\widetilde{\tau}_{lj}^2 s^2\boldsymbol{I})\right],\quad \widetilde{\tau}_{lj}^2=c^2\tau_{lj}^2/(c^2+\tau_{lj}^2 s^2)$$
$$\pi(z_{lj})=\mathrm{Ber}(\lambda_l),\quad \pi(\tau_{lj})=C^+(0,1),\quad \pi(s)=C^+(0,s_0)$$

where $l=0,\ldots,L$, $j=1,\ldots,k_{l+1}$. $C^+(.,.)$ denotes half Cauchy distribution. $\widetilde{\tau}_{lj}^2$ is varying local (per node) scale parameter, s^2 is the varying global scale parameter, and σ_0^2 is the constant global scale parameter. Note that, when weights are strongly shrinking towards 0 then $\tau_{lj}^2 s^2 \ll c^2$ and $\widetilde{\tau}_{lj}^2 \to \tau_{lj}^2 s^2$ which leads to the unregularized version of the group horseshoe. Whereas, when weights are away from 0 then corresponding $\tau_{lj}^2 s^2$ will be large, i.e, $\tau_{lj}^2 s^2 \gg c^2$ and $\widetilde{\tau}_{lj}^2 \to c^2$, where c^2 is constant. For these weights corresponding version of regularized group horseshoe in the slab follows $N(0,\sigma_0^2 c^2\boldsymbol{I})$. This helps in thinning out the heavy tails associated with the horseshoe prior. Next, the prior inclusion probabilities, λ_l are common for for all nodes from a given layer similar to SS-GL. Additionally, s_0 is the scale parameter of half Cauchy prior on s that can be tuned for specific situations.

Instead of directly working with the half-Cauchy distributions, we employ a decomposition of the half-Cauchy that relies upon gamma and inverse gamma distributions (Louizos et al., 2017) as this allows us to compute the negative KL-divergence from the scale distribution $\pi(\tau)$ to an approximate log-normal scale posterior $q(\tau)$ in closed form. More specifically, we have a half-Cauchy distribution that can be expressed in a non-centered parametrization

as:

$$\tilde{\beta} \sim IG(1/2, 1), \quad \tilde{\alpha} \sim G(1/2, k^2), \quad \tau^2 = \tilde{\beta}\tilde{\alpha}$$

where $IG(.,.), G(.,.)$ correspond to the inverse Gamma and Gamma distributions in the scale parametrization, and τ follows a half-Cauchy distribution with scale k. Therefore we re-express the whole SS-GHS prior hierarchy as:

$$\pi(\overline{\boldsymbol{w}}_{lj}|z_{lj}) = \left[(1-z_{lj})\boldsymbol{\delta}_0 + z_{lj}N(0, \sigma_0^2\widetilde{\tau}_{lj}^2 s^2 \boldsymbol{I})\right], \quad \pi(z_{lj}) = \text{Ber}(\lambda_l)$$
$$\pi(\beta_{lj}) = IG\left(1/2, 1\right), \quad \pi(\alpha_{lj}) = G\left(1/2, 1\right), \quad \pi(s_b) = IG\left(1/2, 1\right), \quad \pi(s_a) = G\left(1/2, s_0^2\right)$$

Variational family: We consider the following fully factorized variational family

$$q(\overline{\boldsymbol{w}}_{lj}|z_{lj}) = \left[(1-z_{lj})\boldsymbol{\delta}_0 + z_{lj}N(\boldsymbol{\mu}_{lj}, \text{diag}(\boldsymbol{\sigma}_{lj}^2))\right], \quad q(z_{lj}) = \text{Ber}(\gamma_{lj})$$
$$q(\beta_{lj}) = LN(\mu_{lj}^{\{\beta\}}, {\sigma_{lj}^{\{\beta\}}}^2), \quad q(\alpha_{lj}) = LN(\mu_{lj}^{\{\alpha\}}, {\sigma_{lj}^{\{\alpha\}}}^2),$$
$$q(s_b) = LN(\mu^{\{s_b\}}, {\sigma^{\{s_b\}}}^2), \quad q(s_a) = LN(\mu^{\{s_a\}}, {\sigma^{\{s_a\}}}^2)$$

for $l = 0, \ldots, L$, $j = 1, \ldots, k_{l+1}$. Similar to SS-GL variational family, we use Log-Normal family instead of Gamma and Inverse-Gamma families to approximate Gamma and Inverse-Gamma distributed variables to obtain closed form expression for the KL divergence between prior and variational distributions. Moreover, $(\mu_{lj}^{\{\beta\}}, {\sigma_{lj}^{\{\beta\}}}^2), (\mu_{lj}^{\{\alpha\}}, {\sigma_{lj}^{\{\alpha\}}}^2), (\mu^{\{s_b\}}, {\sigma^{\{s_b\}}}^2)$, and $(\mu^{\{s_a\}}, {\sigma^{\{s_a\}}}^2)$ denote the variational mean and standard deviation parameters of the Gaussian distribution associated with $q(\log \beta_{lj}), q(\log \alpha_{lj}), q(s_b)$ and $q(s_a)$.

ELBO: Let $\boldsymbol{\theta}$ be the network weights and $\boldsymbol{\varpi} = (\boldsymbol{z}, \boldsymbol{\tau}^2, s^2)$ be the remaining parameters. Similar to SS-GL, We minimize the loss function: $\mathcal{L} = -\text{ELBO}(q(\boldsymbol{\theta}, \boldsymbol{\varpi}), \pi(\boldsymbol{\theta}, \boldsymbol{\varpi}|\mathcal{D}))$,

$$\begin{aligned}
\mathcal{L} &= -\text{ELBO}(q(\boldsymbol{\theta}, \boldsymbol{\varpi}), \pi(\boldsymbol{\theta}, \boldsymbol{\varpi}|\mathcal{D})) \\
&= -\mathbb{E}_{q(\boldsymbol{\theta}, \boldsymbol{\varpi})}[\log L(\boldsymbol{\theta})] + \sum_{l,i} d_{\text{KL}}(q(z_{lj}), \pi(z_{lj})) \\
&\quad + \int\int \left[\sum_{l,j} q(\boldsymbol{z}_{lj} = 1) \int\int d_{\text{KL}}(q(\overline{\boldsymbol{w}}_{lj}|z_{lj} = 1), \pi(\overline{\boldsymbol{w}}_{lj}|\beta_{lj}, \alpha_{lj}, s_b, s_a, z_{lj} = 1)) \right. \\
&\qquad \left. \times q(\beta_{lj})\, q(\alpha_{lj})\, d\beta_{lj}\, d\alpha_{lj} \right] q(s_b)\, q(s_a)\, ds_b\, ds_a
\end{aligned}$$

$$+\sum_{l,j}\Big[d_{\mathrm{KL}}(q(\beta_{lj}),\pi(\beta_{lj}))+d_{\mathrm{KL}}(q(\alpha_{lj}),\pi(\alpha_{lj}))\Big]+d_{\mathrm{KL}}(q(s_b),\pi(s_b))+d_{\mathrm{KL}}(q(s_a),\pi(s_a))$$

$$=-\mathbb{E}_{q(\boldsymbol{\theta},\boldsymbol{\varpi})}[\log L(\boldsymbol{\theta})]+\sum_{l,i}d_{\mathrm{KL}}(\mathrm{Ber}(\gamma_{lj}),\mathrm{Ber}(\lambda_l))$$

$$+\int\int\Bigg[\sum_{l,j}q(\boldsymbol{z}_{lj}=1)\int\int d_{\mathrm{KL}}(N(\boldsymbol{\mu}_{lj},\mathrm{diag}(\boldsymbol{\sigma}_{lj}^2)),N(0,\sigma_0^2\beta_{lj}\alpha_{lj}s_bs_a\boldsymbol{I}))$$

$$\times q(\beta_{lj})\,q(\alpha_{lj})\,d\beta_{lj}\,d\alpha_{lj}\Bigg]\;q(s_b)\,q(s_a)\,ds_b\,ds_a$$

$$+\sum_{l,j}\Big[d_{\mathrm{KL}}(LN(\mu_{lj}^{\{\beta\}},{\sigma_{lj}^{\{\beta\}}}^2),IG\,(1/2,1))+d_{\mathrm{KL}}(LN(\mu_{lj}^{\{\alpha\}},{\sigma_{lj}^{\{\alpha\}}}^2),G\,(1/2,1))\Big]$$

$$+d_{\mathrm{KL}}(LN(\mu^{\{s_b\}},{\sigma^{\{s_b\}}}^2),IG\,(1/2,1))+d_{\mathrm{KL}}(LN(\mu^{\{s_a\}},{\sigma^{\{s_a\}}}^2),G\left(1/2,s_0^2\right))$$

4.2.3 Algorithm and Computational Details

We minimise the loss $\mathcal{L}$ for both SS-GL and SS-GHS models by recursively sampling their corresponding variational posterior, allowing us to propagate the information through the network. The Gaussian variational approximations, $N(\boldsymbol{\mu}_{lj},\mathrm{diag}(\boldsymbol{\sigma}_{lj}^2))$, are reparameterized as $\boldsymbol{\mu}_{lj}+\boldsymbol{\sigma}_{lj}\odot\boldsymbol{\zeta}_{lj}$ for $\boldsymbol{\zeta}_{lj}\sim N(0,\boldsymbol{I})$, where $\odot$ denotes the entry-wise (Hadamard) product.

Continuous Relaxation. The discrete spike variables ($\boldsymbol{z}$) are replaced with their continuous relaxation to circumvent the nondifferentiablility in $\mathcal{L}$ making practical implementation easier (Jang et al., 2017; Maddison et al., 2017). Specifically, the Gumbel-softmax (GS) distribution is used for continuous relaxation, that is $q(z_{lj})\sim\mathrm{Ber}(\gamma_{lj})$ is approximated by $q(\tilde{z}_{lj})\sim\mathrm{GS}(\gamma_{lj},\tau)$, where

$$\tilde{z}_{lj}=(1+\exp(-\eta_{lj}/\tau))^{-1},\quad \eta_{lj}=\log(\gamma_{lj}/(1-\gamma_{lj}))+\log(u_{lj}/(1-u_{lj})),\quad u_{lj}\sim U(0,1)$$

where τ is the temperature. We keep $\tau=0.5$ for all the experiments similar to (Jantre et al., 2021a). The use of $\tilde{z}_{lj}$ in the backward pass eases gradient calculation, while z_{lj} is used in the forward pass for exact node sparsity.

Algorithm 4.1 Variational inference in SS-GL and SS-GHS Bayesian neural networks

Inputs: training dataset, network architecture, and optimizer tuning parameters.
Model inputs: prior parameters for $\mathcal{T} = (\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{\tau}^2)$ in SS-GL and $\mathcal{T} = (\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{\tau}^2, s^2)$ in SS-GHS.
Variational inputs: number of Monte Carlo samples S.
Output: Variational parameter estimates of network weights, scales, and sparsity.
Method: Set initial values of variational parameters.
repeat
  Generate S samples of $\boldsymbol{\beta}_{lj}, z_{lj}, \tilde{z}_{lj}, \tau_{lj}^2, s^2$ (for SS-GHS)
  Use $\boldsymbol{\beta}_{lj}, \tau_{lj}^2, s^2$ and z_{lj} to compute $\mathcal{L}$ in forward pass
  Use $\boldsymbol{\beta}_{lj}, \tau_{lj}^2, s^2$ and $\tilde{z}_{lj}$ to compute gradient of $\mathcal{L}$ in backward pass
  Update the variational parameters with gradient of loss using stochastic gradient descent algorithm (e.g. SGD with momentum (Sutskever et al., 2013))
until change in ELBO $< \epsilon$

4.3 Numerical Experiments

In this section, we demonstrate the performance of our proposed **SS-GL** and **SS-GHS** approaches on network architectures and techniques used in practice. We consider multilayer perceptron (MLP), LeNet-5-Caffe, and ResNet architectures which we implement in PyTorch (Paszke et al., 2019). We perform image classification using aforementioned neural networks in widely used MNIST, Fashion-MNIST, and CIFAR-10 datasets.

In all the experiments, we fix $\sigma_0^2 = 1$ and $\sigma_e^2 = 1$. The remaining tuning parameter details such as learning rate, minibatch size, and initial parameter choice are provided in the Appendix. The prediction accuracy is calculated using variational Bayes posterior mean estimator with 10 Monte Carlo samples at test-time. We use swish (SiLU) activations (Elfwing et al., 2018; Ramachandran et al., 2017) instead of ReLU in our proposed SS-GL and SS-GHS models similar to Jantre et al. (2021a)'s spike-and-slab Gaussian node selection model (SS-IG) to avoid the dying neuron problem (Lu et al., 2020). Smoother activation functions such as sigmoid, tanh, etc also help alleviate this problem. We choose swish since it has the best performance.

We provide node sparsity estimates for each linear hidden layer separately. For all models, the node sparsity in a given linear layer is the ratio of number of neurons with atleast one

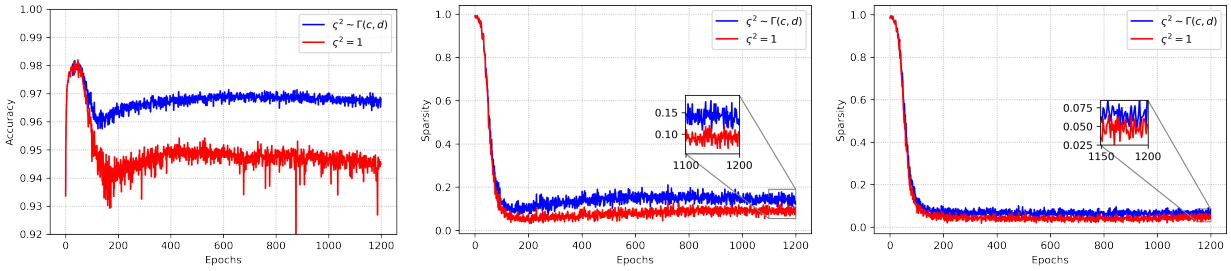
nonzero incoming edge over the original number of neurons present in that layer before training. In a convolution layer we provide channel sparsity estimate which is ratio of the number of output channels with atleast one nonzero incoming connection over the total number of output channels present in the dense counterpart. The layer-wise node or channel sparsity estimates provide granular illustration of the structural compactness of the trained model. The structural sparsity in the trained model leads to lower computational complexity at test-time which is vital for resource constrained devices.

4.3.1 MLP MNIST Classification

In this experiment, we use MLP model with 2 hidden layers having 400 nodes per layer to fit the MNIST data which consists of 60,000 small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9. We preprocess the images in the MNIST data by dividing their pixel values by 126. Output layer has 10 neurons since there are 10 classes in the MNIST data. We provide the graph of the prediction accuracy of the i.i.d. test data over training period of 1200 epochs. We provide layer-wise node sparsity plots for both layers to highlight the dynamic structural compactness of the model under training. In what follows, we discuss the choice of ς^2 in SS-GL model as well as the choice of c_{reg} values in SS-GHS model.

SS-GL penalty parameter choice. In SS-GL model, the value of ς^2 need to be carefully tuned for numerical experiments (Xu and Ghosh, 2015). A very large value of ς^2 will over-shrink the network weights and leading to biased estimates; $\varsigma^2 \to 0$ will lead to a very diffuse distribution for the slab part. Instead we place a conjugate gamma prior on the penalty parameter, $\varsigma^2 \sim \Gamma(c, d)$, and estimate it through our variational inference framework via an approximating family $q(\varsigma^2) := LN(\mu_\varsigma, \sigma_\varsigma^2)$.

Figure 4.2 summarizes the results of MLP-MNIST experiment using SS-GL model with fixed $\varsigma^2 = 1$ and variable $\varsigma^2 \sim \Gamma(c = 4, d = 2)$. The values of the shape $(c = 4)$ and rate $(d = 2)$ parameters where chosen based on hyperparameters search and past literature. We



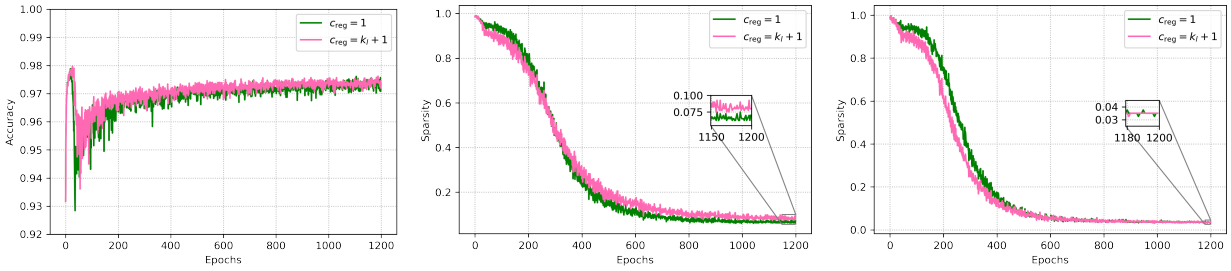
(a) Prediction accuracy (b) Layer-1 node sparsity (c) Layer-2 node sparsity

Figure 4.2 **SS-GL penalty parameter choice experiment results.** Here, we demonstrate the performance of SS-GL with fixed $\varsigma^2 = 1$ and variable $\varsigma^2 \sim \Gamma(c = 4, d = 2)$. (a) The classification accuracy on the test data. (b) and (c) the node sparsity in the layer-1 and layer-2 of the network respectively. We observe that placing a prior on ς^2 yields better classification accuracy.

observe that inferring the value of ς^2 from Bayesian estimation significantly improves the predictive accuracy compared to fixed ς^2 (Figure 4.2a). The fixed ς^2 model has better node sparsities in both the layers of the MLP model (Figure 4.2b and 4.2c). This suggests that $\varsigma^2 = 1$ might be overshrinking the weights which assists in pruning them via spike-and-slab prior, however this also hampers the predictive performance of the model. In rest of the experiments involving SS-GL, we place the gamma prior on $\varsigma^2 \sim \Gamma(c = 4, d = 2)$.

SS-GHS regularization constant choice. In what follows, we provide MLP-MNIST experiment using SS-GHS model with with regularization constant values of $c_{\text{reg}} = 1$ and $c_{\text{reg}} = k_l + 1$. In MLP, the $k_l + 1 = 400 + 1 = 401$ is a large constant and essentially acts as an unregularized model. We ran unregularized version of the model and verified this claim but do not provide the results for brevity.

Figure 4.3 summarizes the results of MLP-MNIST experiment using SS-GHS model $c_{\text{reg}} = 1$ and $c_{\text{reg}} = k_l + 1$. We observe that both values of c_{reg} lead to same predictive accuracies on test data. However in $c_{\text{reg}} = 1$ scenario, SS-GHS model has better layer-1 node sparsity (Figure 4.3b). Layer-2 node sparsity is same in both the c_{reg} values (Figure 4.3c). In rest of the experiments involving SS-GHS, we choose $c_{\text{reg}} = 1$.



(a) Prediction accuracy (b) Layer-1 node sparsity (c) Layer-2 node sparsity

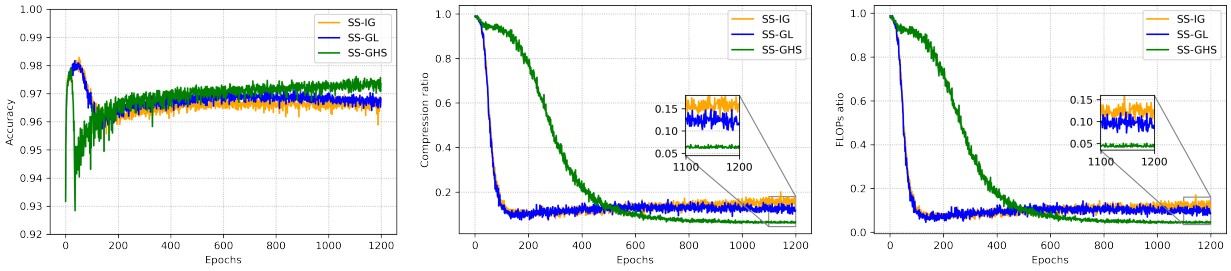
Figure 4.3 **SS-GHS regularization constant choice experiment results.** We demonstrate the performance of SS-GHS with regularization constant of $c_{\text{reg}} = 1$ and $c_{\text{reg}} = k_l + 1 = 401$. (a) The classification accuracy on the test data. (b) and (c) The node sparsity in the layer-1 and layer-2 of the network respectively. We observe that both c_{reg} choices lead to similar classification accuracies with $c_{\text{reg}} = 1$ having better layer-1 node sparsity.

MLP-MNIST comparison with SS-IG

We provide MLP-MNIST experiment where we compare our proposed models with SS-IG model. The results are presented in Figure 4.4. We provide test data accuracy, model compression ratio, flops ratio, and layer-wise node sparsities in each experiment.

Additional metrics. We provide two additional metrics that relate to the model compression and computational complexity. (i) *compression ratio:* it is the ratio of number of nonzero weights in the compressed network versus the dense model and is an indicator of storage cost at test-time. (ii) *floating point operations (FLOPs) ratio:* it is the ratio of number of FLOPs required to predict the output from the input during test-time in the compressed network versus its dense counterpart. We have detailed the FLOPs calculation in neural networks in Chapter 3 Appendix B. Layer-wise node and channel sparsities are directly related to FLOPs ratio hence we only provide FLOPs ratio in LeNet-5-Caffe and ResNet models.

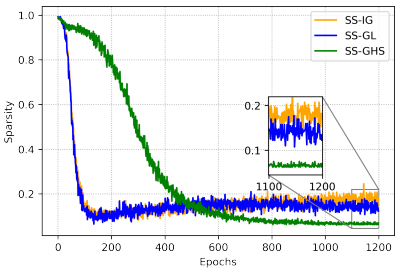
In Figure 4.4a, we observe that SS-GHS has better predictive accuracy compared to SS-GL and SS-IG models. Moreover, SS-GHS model not only has minimal storage cost among the node selection models compared (Figure 4.4b) but also the least number of FLOPs required for inference during test-time (Figure 4.4c). In Figure 4.4d and 4.4e, we observe



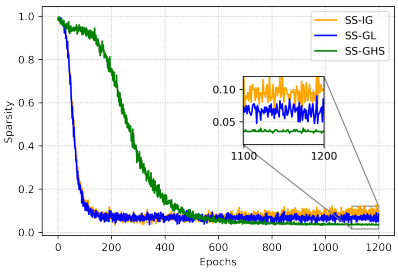
(a) Prediction accuracy

(b) Compression ratio

(c) FLOPs ratio



(d) Layer-1 node sparsity



(e) Layer-2 node sparsity

Figure 4.4 **MLP/MNIST experiment results.** Here, we demonstrate the performance of our SS-GL ($\varsigma^2 \sim \Gamma(c = 4, d = 2)$) and SS-GHS ($c_{\text{reg}} = 1$) models compared against SS-IG model. (a) we plot the classification accuracy on the test data. (b) and (c) we plot the node sparsity in the layer-1 and layer-2 of the network respectively. We observe that our SS-GHS yields the most compact network with the best classification accuracy.

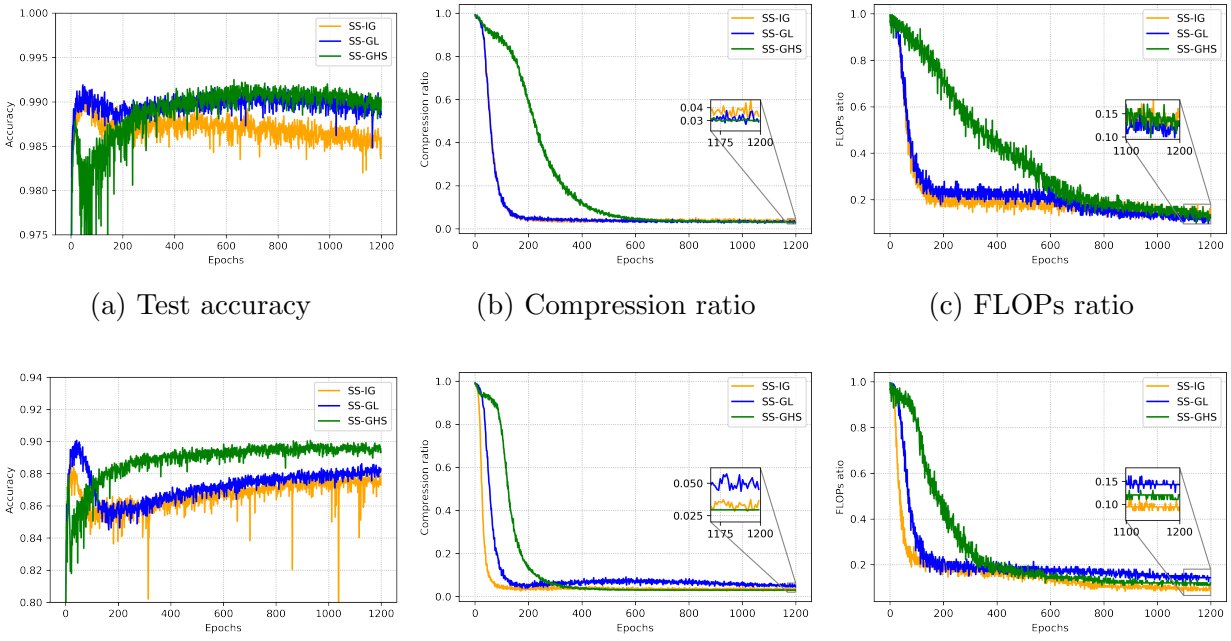
that SS-GHS has pruned away maximum number of nodes in contrast to SS-GL and SS-IG models and this also leads to the maximum reduction in FLOPs evident from (Figure 4.4c). Lastly, SS-GL and SS-IG models have similar predictive accuracies; however, SS-GL has lower layer-wise node sparsities in both layers, hence lower FLOPs ratio and it also has lower storage cost at test-time compared to SS-IG.

4.3.2 LeNet-5-Caffe Experiments

The results of more complex LeNet-5-Caffe network experiments on MNIST and Fashion-MNIST are presented in Figure 4.5. We provide test data accuracy, model compression ratio, and FLOPs ratio in each experiment over 1200 epochs. Here, FLOPs ratio serve as a collective indicator of layer-wise node sparsities since FLOPs are directly related to how many neurons or channels are remaining in linear or convolution layers respectively.

In LeNet-5-Caffe/MNIST experiment (Figure 4.5a - 4.5c), we observe that our SS-GHS and SS-GL models have better predictive accuracy than SS-IG (Figure 4.5a). We observe that both SS-GHS and SS-GL models have better model compression ratio (Figure 4.5b). Moreover, all three models compared achieve similar reduction in Flops (Figure 4.5c)). In contrast with MLP-MNIST experiment (Figure 4.4) our SS-GHS and SS-GL have same performance on all metrics in LeNet-5-Caffe-MNIST experiment.

In LeNet-5-Caffe/Fashion-MNIST experiment (Figure 4.5d - 4.5f), we observe that SS-GHS has better predictive accuracy compared to SS-GL and SS-IG models. The storage cost reduction in SS-GHS model is similar to SS-IG but better than SS-GL (Figure 4.5e). Next, SS-IG achieves best reduction in FLOPs compared to both our approaches and SS-GHS has lower FLOPs than SS-GL. Lastly, SS-GL and SS-IG models have similar predictive accuracies; however, SS-IG has lower FLOPs and storage cost at test-time.



(a) Test accuracy (b) Compression ratio (c) FLOPs ratio

(d) Test accuracy (e) Compression ratio (f) FLOPs ratio

Figure 4.5 **LeNet-5-Caffe/MNIST and LeNet-5-Caffe/Fashion-MNIST experiments results.** Top row (a)-(c) represent the LeNet-5-Caffe on MNIST experiment results. Bottom row (d)-(f) represent the LeNet-5-Caffe on Fashion-MNIST experiment results.

4.3.3 Residual Network Experiments

This section presents an example demonstrating the trade-off between the computational complexity and memory cost at test-time among our structured pruning methods and recent unstructured pruning methods in Residual Networks (ResNet) applied on CIFAR-10 dataset (He et al., 2016). The CIFAR-10 dataset (Krizhevsky, 2009) consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. We use ResNet-20 and ResNet-32 architectures and follow the experimental setting provided by Sun et al. (2021). We compare our proposed SS-GL and SS-GHS methods with SS-IG (Jantre et al., 2021a), consistent sparse deep learning (BNN_{cs}) (Sun et al., 2021), and variational BNN with mixture Gaussian prior (VBNN) (Blundell et al., 2015).

In all of our experiments, we follow the same training setup as used in Sun et al. (2021), that is, each model considered in Table 4.1 was trained using SGD with momentum for 300 epochs with mini-batch size of 128 during training, the momentum parameter was set to 0.9, and the initial learning rate was set to 0.1. The training data was preprocessed using the random erasing data augmentation strategy proposed by Zhong et al. (2020). We use step-wise constant learning rate schedule and decrease the learning rate by a factor of 10 at epochs 150 and 225. We refine the parameters associated with sparse sub-network obtained after first 300 epochs for additional 100 epochs where sparsity parameters in SS-IG and our models are not trained and only parameters associated with the slab part are learned. We use swish activations in our SS-GL and SS-GHS models as well as SS-IG. For BNN_{cs} and VBNN models, we use ReLU activations as recommended by their authors. In all the models, we set $\sigma_e^2 = 1$ and $\sigma_0^2 = 0.04$ same as Sun et al. (2021). In SS-IG, SS-GL, and SS-GHS we use common $\lambda_l = 10^{-4}$ after hyperparamter search, since our theory does not cover Bayesian CNNs. However, establishing the posterior consistency in the Bayesian CNN is an interesting future direction of work.

We quantify the predictive performance using the accuracy of the test data. Besides the test accuracy, we use *compression (%)* and *pruned FLOPs (%)* which are compression ratio

and FLOPs ratio discussed earlier converted to percentages respectively. In this experiment, we only count parameters and FLOPs over convolutional and last fully connected layer, because our proposed methods focus on channel and node pruning of convolutional and linear layers respectively.

For ResNet architecture, our proposed methods under centered parameterization similar to previous experiments in Section 4.3.1 and 4.3.2 have unstable performance. Instead we incorporate non-centered parameterization (Ghosh et al., 2019) to stabilize the training. Below we detail the non-centered parameterization strategy:

Non-Centered Parameterization

We adopt non-centered parameterization for the Gaussian slab component in both the prior setups to circumvent the pathological funnel shaped geometries associated with the coupled posterior (Ingraham and Marks, 2017; Ghosh et al., 2019). Accordingly, the coupling between weights $\overline{\boldsymbol{w}}_{lj}$ and scales $\tau_{lj}^{*2} = \tau_{lj}^2$ (for SS-GL) or $\tau_{lj}^{*2} = \widetilde{\tau}_{lj}^2 s^2$ (for SS-GHS) can be reformulated as

$$\boldsymbol{\beta}_{lj} \sim N(0, \sigma_0^2 \tau_{lj}^{*2} \boldsymbol{I}), \quad \overline{\boldsymbol{w}}_{lj} = \tau_{lj}^* \beta_{lj},$$

This formulation leads to independent sampling of weights and scales from their respective prior distributions which are now marginally uncorrelated leads to simpler posterior geometries (Betancourt and Girolami, 2015). This non-centered reparameterization leads to efficient posterior inference without change in functional form of the respective prior.

We summarize the ResNet experiment results in Table 4.1. The comparison with BNN_{cs} and VBNN models indicates that our SS-GL and SS-GHS methods have significantly better prediction accuracy in both ResNet-20 and ResNet-32 setups. Moreover, we demonstrate that even though BNN_{cs} and VBNN models have predefined high levels of pruned parameters, our models have significantly less FLOPs during inference at test-time. This highlights the trade-off between unstructured sparsity and structured sparsity methods, where the former leads to significant reduction in storage cost and the later has significantly less computa-

Table 4.1 **ResNet-20/CIFAR-10 and ResNet-32/CIFAR-10 experiments results.** The results of each method is calculated by averaging over 3 independent runs with standard deviation reported in parentheses. For BNN_{cs} and VBNN models, we show predefined percentages of pruned parameters used for magnitude pruning given in (Sun et al., 2021).

Model	Method	Test Accuracy	Compression (%)	Pruned FLOPs (%)
ResNet-20	BNN_{cs} (20%)	92.23 (0.16)	19.29 (0.12)	98.94 (0.38)
	BNN_{cs} (10%)	91.43 (0.11)	9.18 (0.13)	99.13 (0.37)
	VBNN (20%)	89.61 (0.04)	19.55 (0.01)	100.00 (0.00)
	VBNN (10%)	88.43 (0.13)	9.50 (0.00)	99.93 (0.00)
	SS-IG	92.94 (0.15)	79.52 (0.98)	88.39 (1.00)
	SS-GL (ours)	**92.99 (0.11)**	76.10 (1.55)	**85.15 (1.76)**
	SS-GHS (ours)	92.87 (0.23)	78.70 (0.42)	86.18 (1.02)
ResNet-32	BNN_{cs} (10%)	92.65 (0.03)	9.15 (0.03)	94.53 (0.86)
	BNN_{cs} (5%)	91.39 (0.08)	4.49 (0.02)	90.79 (1.35)
	VBNN (10%)	89.37 (0.04)	9.61 (0.01)	99.99 (0.02)
	VBNN (5%)	87.38 (0.22)	4.59 (0.01)	94.27 (0.54)
	SS-IG	93.08 (0.23)	55.28 (2.96)	67.59 (2.36)
	SS-GL (ours)	**93.33 (0.11)**	54.27 (1.73)	66.93 (2.98)
	SS-GHS (ours)	93.15 (0.23)	53.72 (2.11)	**66.68 (2.75)**

tional complexity at test-time. In comparison with SS-IG node selection model, we observe that our SS-GL and SS-GHS models have lower storage cost and FLOPS at test-time with comparable predictive accuracy in ResNet-20 architecture. In ResNet-32 case, SS-GL has better predictive accuracy than SS-IG, while compression (%) and pruned FLOPs (%) in our models are comparable to SS-IG within standard deviation. This comparison further highlights the advantage of using group shrinkage priors instead of Gaussian in the slab part of spike-and-slab framework to achieve better test accuracies with lower computational and memory footprint.

4.4 Conclusion and Discussion

In this chapter, we introduced the compact Bayesian neural network methods to handle the model compression in a principled manner. Our proposed spike-and-slab models combine the automated sparsity learning with the hierarchical group shrinkage priors: group lasso

and group horseshoe. We provide computationally efficient and scalable variational inference algorithms in both the models. In the large scale experiments involving ResNet architectures, we relied on the non-centered parameterization which ensured the numerical stability of our models. We demonstrate the superior performance of the group shrinkage priors over Gaussian prior in slab component in several experiments which further highlights our point that group shrinkage priors shrink the collection of weights incident on the node close to zero which helps in removal of that node through spike-and-slab framework.

An immediate future work would be to establish the variational posterior consistency and corresponding contraction rate in both the models. Moreover, the superiority of the SS-GL and SS-GHS models over the SS-IG model could be established through the faster posterior convergence rate for the former models compared to the later.

APPENDIX

APPENDIX

ADDITIONAL NUMERICAL EXPERIMENTS DETAILS

A.1 Variational parameters initialization

We initialize the γ_{lj}'s at a value close to 1 for all of our experiments. This ensures that at epoch 0, we have a fully connected deep neural network. The variational parameters $\mu_{ljj'}$ are initialized using Kaiming uniform initialization (He et al., 2015). Moreover, $\sigma_{ljj'}$ are reparameterized using softplus function: $\sigma_{ljj'} = \log(1 + \exp(\rho_{ljj'}))$ and $\rho_{ljj'}$ are initialized using a constant value of -6. This keeps initial values of $\sigma_{ljj'}$ close to 0 ensuring that the initial values of network weights stay close to Kaiming uniform initialization.

In SS-GL, $\mu_{lj}^{\{\tau\}}$ is initialized using $U(-0.6, 0.6)$ and $\sigma_{lj}^{\{\tau\}} = \log(1 + \exp(\rho_{lj}^{\{\tau\}}))$ where $\rho_{lj}^{\{\tau\}}$ are initialized to -6. Moreover, μ_{ς} is initialized to 1 and $\sigma_{\varsigma} = \log(1 + \exp(\rho^{\{\tau\}}))$ where $\rho^{\{\tau\}}$ is initialized to -6.

In SS-GHS, $\mu_{lj}^{\{\alpha\}}$ and $\mu_{lj}^{\{\beta\}}$ are initialized using $U(-0.6, 0.6)$. $\sigma_{lj}^{\{\alpha\}} = \log(1 + \exp(\rho_{lj}^{\{\alpha\}}))$ and $\sigma_{lj}^{\{\beta\}} = \log(1 + \exp(\rho_{lj}^{\{\beta\}}))$ where $\rho_{lj}^{\{\alpha\}}$ and $\rho_{lj}^{\{\beta\}}$ are initialized to -6. Next, $\mu^{\{s_a\}}$ and $\mu^{\{s_b\}}$ are initialized to 1. $\sigma^{\{s_a\}} = \log(1 + \exp(\rho^{\{s_a\}}))$ and $\sigma^{\{s_b\}} = \log(1 + \exp(\rho^{\{s_b\}}))$ where $\rho^{\{s_a\}}$ and $\rho^{\{s_b\}}$ are initialized to -6.

A.2 Hyperparameters for training

In MLP-MNIST and LeNet-5-Caffe-MNIST experiments, we use 10^{-3} learning rate and 1024 minibatch size for all three models compared. In LeNet-5-Caffe-Fashion-MNIST experiment, we train SS-GL with 10^{-3} learning rate whereas, SS-GHS and SS-IG are trained with 2×10^{-3} where minibatch size is 1024 in all three models compared. We train each model for 1200 epochs using Adam optimizer in all the MLP and LeNet-5-Caffe experiments.

CHAPTER 5

SEQUENTIAL BAYESIAN NEURAL SUBNETWORK ENSEMBLES

5.1 Introduction

Bayesian neural networks (BNNs) have pushed the envelope of probabilistic machine learning through the combination of deep neural network architecture and Bayesian inference. However, due to the enormous number of parameters, BNNs adopt approximate inference techniques such as variational inference with a fully factorized approximating family (Jordan et al., 1999). Although this approximation is crucial for computational tractability, they could lead to under-utilization of BNN's true potential (Izmailov et al., 2021).

Recently, ensemble of neural networks (Lakshminarayanan et al., 2017) has been proposed to account for the parameter/model uncertainty, which has been shown to be analogous to the Bayesian model averaging and sampling from the parameter posteriors in the Bayesian context to estimate the posterior predictive distribution (Wilson and Izmailov, 2020). In this spirit, the diversity of the ensemble has been shown to be a key to improving the predictions, uncertainty, and robustness of the model. To this end, diverse ensembles can mitigate some of the shortcomings introduced by approximate Bayesian inference techniques without compromising computational tractability. Several different diversity-inducing techniques have been explored in the literature. The approaches range from using a specific learning rate schedule (Huang et al., 2017), to introducing kernalized repulsion terms among the ensembles in the loss function at train time (D'Angelo and Fortuin, 2021), mixture of approximate posteriors to capture multiple posterior modes (Dusenberry et al., 2020), appealing to sparsity (albeit ad-hoc) as a mechanism for diversity (Havasi et al., 2021; Liu et al., 2022) and finally appealing to diversity in model architectures through neural architecture and hyperparameter searches (Egele et al., 2021; Wenzel et al., 2020).

However, most approaches prescribe parallel ensembles, with each individual model part

of an ensemble starting with a different initialization, which can be expensive in terms of computation as each of the ensembles has to train longer to reach the high-performing neighborhood of the parameter space. Although the aspect of ensemble diversity has taken center stage, the cost of training these ensembles has not received much attention. However, given that the size of models is only growing as we advance in deep learning, it is crucial to reduce the training cost of multiple individual models forming an ensemble in addition to increasing their diversity.

To this end, sequential ensembling techniques offer an elegant solution to reduce the cost of obtaining multiple ensembles, whose origin can be traced all the way back to Swann and Allinson (1998); Xie et al. (2013), wherein ensembles are created by combining epochs in the learning trajectory. Jean et al. (2015); Sennrich et al. (2016) use intermediate stages of model training to obtain the ensembles. Moghimi et al. (2016) used boosting to generate ensembles. In contrast, recent works by Huang et al. (2017); Garipov et al. (2018); Liu et al. (2022) force the model to visit multiple local minima by cyclic learning rate annealing and collect ensembles only when the model reaches a local minimum. Notably, the aforementioned sequential ensembling techniques in the literature have been proposed in the context of deterministic machine learning models. Extending the sequential ensembling technique to Bayesian neural networks is attractive because we can potentially get high-performing ensembles without the need to train from scratch, analogous to sampling with a Markov chain Monte Carlo sampler that extracts samples from the posterior distribution. Furthermore, sequential ensembling is complementary to the parallel ensembling strategy, where, if the models and computational resources permit, each parallel ensemble can generate multiple sequential ensembles, leading to an overall increase in the total number of diverse models in an ensemble.

On the other hand, Chapters 3 and 4 make a case for (i) the automatic data-driven sparsity learning in Bayesian neural networks through the use of spike-and-slab priors, (ii) the use of group sparsity priors Louizos et al. (2017); Ghosh et al. (2019); Jantre et al.

(2021a) to provide structural sparsity in Bayesian neural networks leading to significant computational gains. In this work, we leverage the automated structural sparsity learning using spike-and-slab priors similar to Jantre et al. (2021a) in our approach to sequentially generate multiple Bayesian neural subnetworks with varying sparse connectivities which when combined yields highly diverse ensemble.

To this end, we propose **Se**quential **Bay**esian Neural **S**ubnetwork Ensembles (`SeBayS`) with the following major contributions:

- We propose a sequential ensembling strategy for Bayesian neural networks (BNNs) which learns multiple subnetworks in a single forward-pass. The approach involves a single exploration phase with a large (constant) learning rate to find high-performing sparse network connectivity yielding structurally compact network. This is followed by multiple exploitation phases with sequential perturbation of variational mean parameters using corresponding variational standard deviations together with piecewise-constant cyclic learning rates.

- We combine the strengths of the automated sparsity-inducing spike-and-slab prior that allows dynamic pruning during training, which produces structurally sparse BNNs, and the proposed sequential ensembling strategy to efficiently generate diverse and sparse Bayesian neural networks, which we refer to as *Bayesian neural subnetworks*.

5.1.1 Related Work

Ensembles of neural networks: Ensembling techniques in the context of neural networks are increasingly being adopted in the literature due to their potential to improve accuracy, robustness, and quantify uncertainty. The most simple and widely used approach is Monte Carlo dropout, which is based on Bernoulli noise (Gal and Ghahramani, 2016) and deactivates certain units during training and testing. This, along with techniques such as DropConnect (Wan et al., 2013), Swapout (Singh et al., 2016) are referred to as"implicit"

ensembles as model ensembling is happening internally in a single model. Although they are efficient, the gain in accuracy and robustness is limited and they are mainly used in the context of deterministic models. Although most recent approaches have targeted parallel ensembling techniques, few approaches such as BatchEnsemble (Wen et al., 2020) appealed to parameter efficiency by decomposing ensemble members into a product of a shared matrix and a rank-one matrix, and using the latter for ensembling and MIMO (Havasi et al., 2021) which discovers subnetworks from a larger network via multi-input multi-output configuration. In the context of Bayesian neural network ensembles, Dusenberry et al. (2020) proposed a rank-1 parameterization of BNNs, where each weight matrix involves only a distribution on a rank-1 subspace and uses mixture approximate posteriors to capture multiple modes.

Sequential ensembling techniques offer an elegant solution to ensemble training but have not received much attention recently due to a wider focus of the community on diversity of ensembles and less on the computational cost. Notable sequential ensembling techniques are Huang et al. (2017); Garipov et al. (2018); Liu et al. (2022) that enable the model to visit multiple local minima through cyclic learning rate annealing and collect ensembles only when the model reaches a local minimum. The difference is that Huang et al. (2017) adopts cyclic cosine annealing, Garipov et al. (2018) uses a piecewise linear cyclic learning rate schedule that is inspired by geometric insights. Finally, Liu et al. (2022) adopts a piecewise-constant cyclic learning rate schedule. We also note that all of these approaches have been primarily in the context of deterministic neural networks.

Our proposed approach (i) introduces sequential ensembling into Bayesian neural networks, (ii) combines it with dynamic sparsity through sparsity-inducing Bayesian priors to generate Bayesian neural subnetworks, and subsequently (iii) produces diverse model ensembles efficiently. It is also complementary to other parallel ensembling as well as efficient ensembling techniques.

5.2 Sequential Bayesian Neural Subnetwork Ensembles

5.2.1 Base Learner

We call individual models which are part of an ensemble the "Base learners". Here we provide the prior and corresponding fully factorized variational family that we use in our proposed sequential ensembles.

Prior Choice. Zero-mean Gaussian distribution is a widely popular choice of prior for the model parameters ($\boldsymbol{\theta}$) (Izmailov et al., 2021; Louizos et al., 2017; Mackay, 1992; Neal, 1996; Blundell et al., 2015). In our sequential ensemble of dense BNNs, we adopt the zero-mean Gaussian prior similar to Blundell et al. (2015) in each individual BNN model part of an ensemble. The prior and corresponding fully factorized variational family is given as follows

$$p(\theta_{ljk}) = N(0, \sigma_0^2), \qquad q(\theta_{ljk}) = N(\mu_{ljk}, \sigma_{ljk}^2)$$

where θ_{ljk} is the k^{th} weight incident onto the j^{th} node (in MLP) or output channel (in CNN) in the l^{th} layer. $N(.,.)$ represents the Gaussian distribution. σ_0^2 is the constant prior Gaussian variance and is chosen through hyperparameter search. μ_{ljk} and σ_{ljk}^2 are the variational mean and standard deviation parameters of $q(\theta_{ljk})$.

Dynamic sparsity learning for our sequential ensemble of sparse BNNs is achieved via spike-and-slab prior. We adopt the sparse BNN model, SS-IG (Jantre et al., 2021a) as a base learner in to achieve the structural sparsity in Bayesian neural networks. The prior and corresponding variational family in SS-IG model is given in Section 2.3.

5.2.2 Sequential Ensembling and Bayesian Neural Subnetworks

We propose an ensembling procedure to obtain the base learners $\{\boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \cdots, \boldsymbol{\theta}^M\}$ sequentially in a single training run and construct the ensemble. The ensemble predictions are calculated using the uniform average of the predictions obtained from each base learner.

Specifically, if y^j_{new} represents the outcome of the m^{th} base learner, then the ensemble prediction of M base learners (for continuous outcomes) is $y_{\text{new}} = \frac{1}{M}\sum_{m=1}^{M} y^m_{\text{new}}$.

Sequential Perturbations. Our ensembling strategy produces diverse set of base learners from a single end-to-end training process. It consists of an exploration phase followed by M exploitation phases. The exploration phase is carried out with a large constant learning rate for t_0 time. This allows us to explore high-performing regions of the parameter space. At the conclusion of the exploration phase, the variational posterior approximation for the model parameters reaches a good region on the posterior density surface. Next, during each equally spaced exploitation phase (time = t_{ex}) of the ensemble training, we first use moderately large learning rate for $t_{\text{ex}}/2$ time followed by small learning rate for remaining $t_{\text{ex}}/2$ time. After the first model convergence step (time = $t_0 + t_{\text{ex}}$), we perturb the mean parameters of the variational posterior distributions of the model weights using their corresponding standard deviations. The initial values of these mean variational parameters at each subsequent exploitation phase become $\mu'_{ljk} = \mu_{ljk} \pm \rho * \sigma_{ljk}$, where ρ is a perturbation factor. This perturbation and subsequent model learning strategy is repeated a total of $M - 1$ times, generating M base learners (either dense or sparse BNNs) creating our sequential ensemble.

Sequential Bayesian Neural Subnetwork Ensemble (`SeBayS`). In this ensembling procedure we use a large (and constant) learning rate (e.g., 0.1) in the exploration phase to find high-performing sparse network connectivity in addition to exploring a wide range of model parameter variations. The use of large learning rate facilitates pruning of excessive nodes or output channels, leading to a compact Bayesian neural subnetwork. This structural compactness of the Bayesian neural subnetwork further helps us after each sequential perturbation step by quickly converging to different local minima potentially corresponding to the different modes of the true Bayesian posterior distribution of the model parameters.

Freeze vs No Freeze Sparsity. In our `SeBayS` ensemble, we propose to evaluate two different strategies during the exploitation phases: (1) `SeBayS-Freeze`: freezing the sparse connectivity after the exploration phase, and (2) `SeBayS-No Freeze`: letting the sparsity

Algorithm 5.1 Sequential Bayesian neural subnetwork ensemble (SeBayS) algorithm

Inputs: training data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, network architecture $\eta_{\boldsymbol{\theta}}$, ensemble size M, perturbation factor ρ, exploration phase training time t_0, training time of each exploitation phase t_{ex}.
Model inputs: prior hyperparameters for $\boldsymbol{\theta}$, $\boldsymbol{z}$ (for sparse models).
Output: Variational parameter estimates of network weights and sparsity.
Method: Set initial values of variational parameters: $\boldsymbol{\mu}_{\text{init}}, \boldsymbol{\sigma}_{\text{init}}, \boldsymbol{\gamma}_{\text{init}}$.
# Exploration Phase
for $t = 1, 2, \ldots, t_0$ **do**
  Update $\boldsymbol{\mu}_{lj}^0, \boldsymbol{\sigma}_{lj}^0$, and γ_{lj}^0 (for sparse models) $\leftarrow$ SGD($\mathcal{L}$).
end for
Fix the sparsity variational parameters, γ_{lj}, for freeze sparse models
# M Sequential Exploitation Phases
for $m = 1, 2, \ldots, M$ **do**
  for $t = 1, 2, \ldots, t_{\text{ex}}$ **do**
    Update $\boldsymbol{\mu}_{lj}^m, \boldsymbol{\sigma}_{lj}^m$, and γ_{lj}^m (for no freeze sparse models) $\leftarrow$ SGD($\mathcal{L}$).
  end for
  Save variational parameters of converged base learner $\eta_{\boldsymbol{\theta}}^m$.
  Perturb variational mean parameters using standard deviations: $\mu_{\text{init}}^{m+1} = \mu^m \pm \rho * \sigma^m$
  Set variational standard deviations to a small value: $\sigma_{\text{init}}^{m+1} = 10^{-6}$.
end for

parameters learn after the exploration phase. The first approach fixes the sparse connectivity leading to lower computational complexity during the exploitation phase training. The diversity in the `SeBayS-Freeze` ensemble is achieved via sequential perturbations of the mean parameters of the variational distribution of the active model parameters in the subnetwork. The second approach lets the sparsity learn beyond the exploration phase, leading to highly diverse subnetworks at the expense of more computational complexity compared to the `SeBayS-Freeze` approach.

We found that the use of sequential perturbations and dynamic sparsity leads to high-performant subnetworks with different sparse connectivities. Compared to parallel ensembles, we achieve higher ensemble diversity in single forward pass. The use of a spike-and-slab prior allows us to dynamically learn the sparsity during training, while the Bayesian framework provides uncertainty estimates of the model and sparsity parameters associated with the network. Our approach is the first one in the literature that performs sequential en-

sembling of dynamic sparse neural networks, and more so in the context of Bayesian neural networks.

Initialization Strategy. We initialize the variational mean parameters, $\boldsymbol{\mu}$, using Kaiming He initialization (He et al., 2015) while variational standard deviations, $\boldsymbol{\sigma}$, are initialized to a value close to 0. For dynamic sparsity learning, we initialize the variational inclusion probability parameters associated with the sparsity, γ_{lj}, to be close to 1, which ensures that the training starts from a densely connected network. Moreover, it allows our spike-and-slab framework to explore potentially different sparse connectivities before the sparsity parameters are converged after the initial exploration phase. After initialization, the variational parameters are optimized using the stochastic gradient descent with momentum algorithm (Sutskever et al., 2013).

Algorithm. We provide the pseudocode for our sequential ensembling approaches: (i) BNN sequential ensemble, (ii) `SeBayS-Freeze` ensemble, (iii) `SeBayS-No Freeze` ensemble in Algorithm 5.1.

5.3 Numerical Experiments

In this section, we demonstrate the performance of our proposed `SeBayS` approach on network architectures and techniques used in practice. We consider ResNet-32 on CIFAR10 (He et al., 2016), and ResNet-56 on CIFAR100. These networks are trained with batch normalization, stepwise (piecewise constant) decreasing learning rate schedules, and augmented training data. We provide the source code, all the details related to fairness, uniformity, and consistency in training and evaluation of these approaches and reproducibility considerations for `SeBayS` and other baseline models in the Appendix A.

Baselines. Our baselines include the frequentist model of a deterministic deep neural network (trained with SGD), BNN (Blundell et al., 2015), spike-and-slab BNN for node sparsity (Jantre et al., 2021a), single forward pass ensemble models including rank-1 BNN Gaussian ensemble (Dusenberry et al., 2020), MIMO (Havasi et al., 2021), and EDST ensem-

ble (Liu et al., 2022), multiple forward pass ensemble methods: DST ensemble (Liu et al., 2022) and Dense ensemble of deterministic neural networks. For fair comparison, we keep the training hardware, environment, data augmentation, and training schedules of all the models same. We adopted and modified the open source code provided by the Liu et al. (2022) and Nado et al. (2021) to implement the baselines and train them. Extra details about model implementation and learning parameters are provided in the Appendix A.

Metrics. We quantify predictive accuracy and robustness focusing on the accuracy and negative log-likelihood (NLL) of the i.i.d. test data (CIFAR-10 and CIFAR-100) and corrupted test data (CIFAR-10-C and CIFAR-100-C) involving 19 types of corruption (e.g., added blur, compression artifacts, frost effects) (Hendrycks and Dietterich, 2019). More details on the evaluation metrics are given in the Appendix A.

Results. The results for CIFAR10 and CIFAR100 experiments are presented in Tables 5.1 and 5.2, respectively. For all ensemble baselines, we keep the number of base models $M = 3$ similar to Liu et al. (2022). We report the results for sparse models in the upper half and dense models in the lower half of Tables 5.1 and 5.2. In our models, we choose the perturbation factor (ρ) to be 3. See the Appendix E for additional results on the effect of perturbation factor.

We observe that our BNN sequential ensemble consistently outperforms single sparse and dense models, as well as sequential ensemble models in both CIFAR10 and CIFAR100 experiments. Whereas compared to models with 3 parallel runs, our BNN sequential ensemble outperforms the DST ensemble while being comparable to the dense ensemble in simpler CIFAR10 experiments. Next, our `SeBayS-Freeze` and `SeBayS-No Freeze` ensembles outperform single-BNN, SSBNN, and MIMO while being comparable to deterministic and rank-1 BNN in CIFAR10 case. Whereas, in CIFAR10-C they outperform SSBNN, MIMO, and rank-1 BNN. Additionally, `SeBayS-No Freeze` ensemble has comparable performance to deterministic and EDST ensemble, while `SeBayS-Freeze` ensemble outperforms deterministic and EDST ensemble in CIFAR10-C. In ResNet-32/CIFAR10 case, we dynamically pruned

Table 5.1 **ResNet-32/CIFAR10 experiment results.** We mark the best results out of single-pass sparse models in bold and single-pass dense models in blue.

Methods	Acc (↑)	NLL (↓)	cAcc (↑)	cNLL (↓)	# Forward passes (↓)
SSBNN	91.2	0.320	67.5	1.479	1
MIMO (M=3)	88.9	0.333	65.9	**1.102**	1
EDST Ensemble (M = 3)	**93.1**	**0.214**	69.8	1.236	1
SeBayS-Freeze Ensemble (M = 3)	92.5	0.273	**70.4**	1.344	1
SeBayS-No Freeze Ensemble (M = 3)	92.4	0.274	69.8	1.356	1
DST Ensemble (M = 3)	93.3	0.206	71.9	1.018	3
Deterministic	92.6	0.378	69.9	2.143	1
BNN	91.9	0.353	71.3	1.422	1
Rank-1 BNN (M=3)	92.4	**0.238**	68.7	**1.271**	1
BNN Sequential Ensemble (M = 3)	**93.8**	0.265	**73.3**	1.341	1
Dense Ensemble (M=3)	93.8	0.214	72.5	1.381	3

off close to **50%** of the parameters in `SeBayS` approach.

In more complex ResNet56/CIFAR100 experiment, our `SeBayS-Freeze` ensemble outperforms SSBNN and MIMO in both CIFAR100 and CIFAR100-C, while it outperforms deterministic model in CIFAR100-C. Next our `SeBayS-No Freeze` ensemble outperforms SSBNN in both CIFAR100 and CIFAR100-C while it outperforms MIMO in CIFAR100 and

Table 5.2 **ResNet-56/CIFAR100 experiment results.** We mark the best results out of single-pass sparse models in bold and single-pass dense models in blue.

Methods	Acc (↑)	NLL (↓)	cAcc (↑)	cNLL (↓)	# Forward passes (↓)
SSBNN	67.9	1.511	38.9	4.527	1
MIMO (M=3)	65.8	1.528	42.3	**2.522**	1
EDST Ensemble (M = 3)	**71.9**	**0.997**	**44.3**	2.787	1
SeBayS-Freeze Ensemble (M = 3)	69.4	1.393	42.4	3.855	1
SeBayS-No Freeze Ensemble (M = 3)	69.4	1.403	41.7	3.906	1
DST Ensemble (M = 3)	74.0	0.914	46.7	2.529	3
Deterministic	69.8	1.786	41.6	5.856	1
BNN	70.4	1.335	43.2	3.774	1
Rank-1 BNN (M=3)	70.7	**1.075**	43.9	**2.752**	1
BNN Sequential Ensemble (M = 3)	**72.2**	1.250	**44.9**	3.537	1
Dense Ensemble (M=3)	74.2	1.236	45.4	4.093	3

deterministic model in CIFAR100-C. Given the complexity of the CIFAR100, our `SeBayS` approach was able to dynamically prune off close to **18%** of the ResNet56 model parameters.

5.4 Sequential BNN Ensemble Analysis

5.4.1 Function Space Analysis

Quantitative Metrics. We measure the diversity of the base learners in our sequential ensembles by quantifying the pairwise similarity of the base learner's predictions on the test data. The average pairwise similarity is given by

$$\mathcal{D}_d = \mathbb{E}\left[d(\mathcal{P}_1(y|x_1, \cdots, x_N), \mathcal{P}_2(y|x_1, \cdots, x_N))\right]$$

where $d(.,.)$ is a distance metric between the predictive distributions and $\{(\boldsymbol{x}_i, y_i)\}_{i=1,\cdots,N}$ are the test data. We consider two distance metrics:

(1) *Disagreement:* the fraction of the predictions on the test data on which the base learners disagree: $d_{\text{dis}}(\mathcal{P}_1, \mathcal{P}_2) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(\arg\max_{\hat{y}_i} \mathcal{P}_1(\hat{y}_i) \neq \arg\max_{\hat{y}_i} \mathcal{P}_2(\hat{y}_i))$.

(2) *Kullback-Leibler (KL) divergence*: $d_{\text{KL}}(\mathcal{P}_1, \mathcal{P}_2) = \mathbb{E}\left[\log \mathcal{P}_1(y) - \log \mathcal{P}_2(y)\right]$.

When given two models have the same predictions for all the test data, then both disagreement and KL divergence are zero.

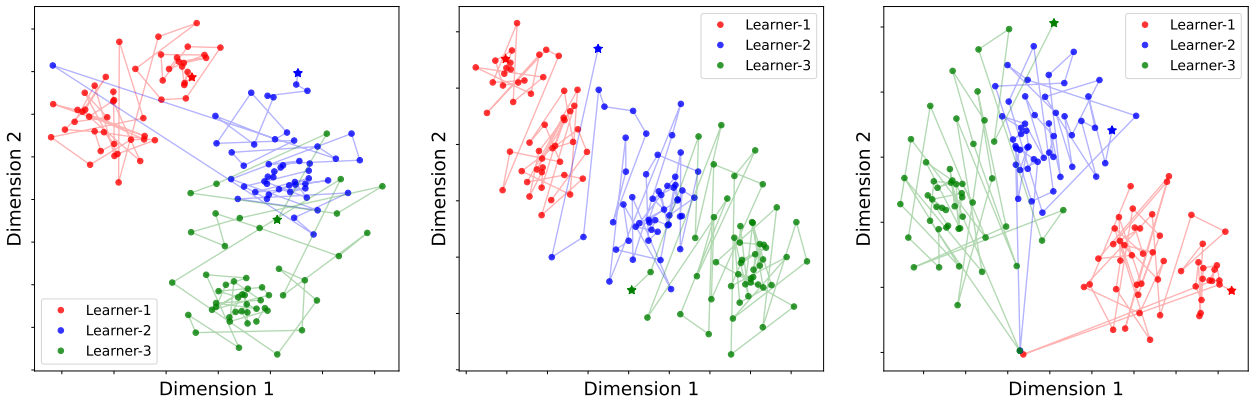
Table 5.3 **Diversity metrics in ResNet-32/CIFAR-10 and ResNet-56/CIFAR100 experiments.** We mark the best results out of single-pass models in bold.

	ResNet-32/CIFAR10			ResNet-56/CIFAR100		
Methods	d_{dis} (↑)	d_{KL} (↑)	Acc (↑)	d_{dis} (↑)	d_{KL} (↑)	Acc (↑)
EDST Ensemble	0.058	0.106	93.1	0.209	0.335	71.9
BNN Sequential Ensemble	0.061	0.201	**93.8**	0.208	0.493	**72.2**
SeBayS-Freeze Ensemble	0.060	0.138	92.5	0.212	0.452	69.4
SeBayS-No Freeze Ensemble	**0.106**	**0.346**	92.4	**0.241**	**0.597**	69.4
DST Ensemble	0.085	0.205	93.3	0.292	0.729	74.0

We report the results of the diversity analysis of the base learners that make up our sequential ensembles in Table 5.3 and compare them with the DST and EDST ensembles. We observe that for simpler CIFAR10 case, our sequential perturbation strategy helps in

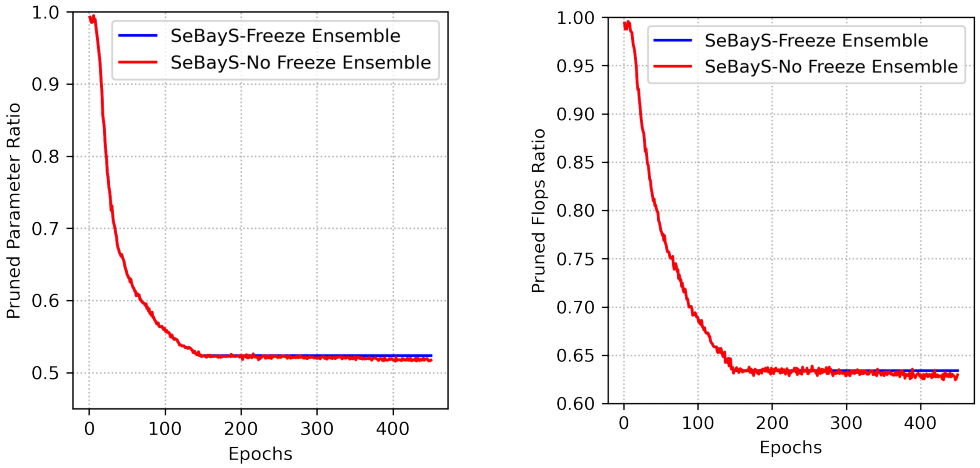
generating diverse base learners compared to EDST ensemble. Specifically, the `SeBayS-No Freeze` ensembles have significantly high prediction disagreement and KL divergence among all the methods, especially surpassing DST ensembles which involve multiple parallel runs. In a more complex setup of CIFAR100, we observe that `SeBayS-No Freeze` ensemble has the highest diversity metrics among single-pass ensemble learners. This highlights the importance of dynamic sparsity learning during each exploitation phase.

Training Trajectory. We use t-SNE (Van Der Maaten and Hinton, 2008) to visualize the training trajectories of the base learners obtained using our sequential ensembling strategy in functional space. In the ResNet32-CIFAR10 experiment, we periodically save the checkpoints during each exploitation training phase and collect the predictions on the test dataset at each checkpoint. After training, we use t-SNE plots to project the collected predictions into the 2D space. In Figure 5.1, the local optima reached by individual base learners using sequential ensembling in all three models is fairly different. The distance between the optima can be explained by the fact that the perturbed variational parameters in each exploitation phase try to reach nearby local optima.



(a) BNN Sequential Ensemble (b) `SeBayS-Freeze` (c) `SeBayS-No Freeze`

Figure 5.1 **Training trajectories of base learners in ResNet32/CIFAR10 experiment.** Training trajectories obtained by BNN sequential ensemble, `SeBayS-Freeze` Ensemble, and `SeBayS-No Freeze` Ensemble.



(a) Pruned Parameter Ratio

(b) Pruned Flops Ratio

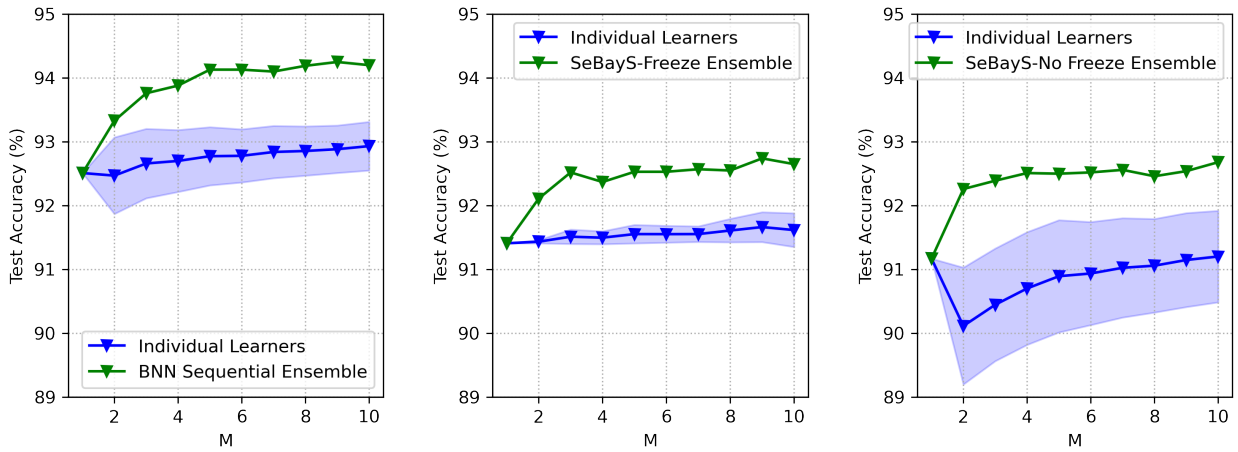
Figure 5.2 **Dynamic sparsity and FLOPs curves.** They show ratio of remaining parameters and FLOPs for our `SeBayS-Freeze` and `SeBayS-No Freeze` ensembles in ResNet32-CIFAR10 experiment.

5.4.2 Dynamic Sparsity Learning

In this section, we highlight the dynamic sparsity training in our `SeBayS` ensemble methods. We focus on ResNet-32/CIFAR10 experiment and consider $M = 3$ exploitation phases. In particular, we plot the ratios of remaining parameters and floating point operations (FLOPs) in the `SeBayS` sparse base learners. In Figure 5.2, We observe that during exploration phase, `SeBayS` prunes off 50% of the network parameters and more than 35% of the FLOPs compared to its dense counterpart.

5.4.3 Effect of Ensemble size

In this section, we explore the effect of the ensemble size M in ResNet-32/CIFAR10 experiment. According to the ensembling literature (Hansen and Salamon, 1990; Ovadia et al., 2019), increasing number of diverse base learners in the ensemble improves predictive performance, although with a diminishing impact. In our ensembles, we generate models and aggregate performance sequentially with increasing M, the number of base learners in the ensemble.



(a) BNN Sequential Ensemble (b) `SeBayS-Freeze` (c) `SeBayS-No Freeze`

Figure 5.3 **Predictive performance results of the base learners and the sequential ensembles as the ensemble size M varies in ResNet32/CIFAR10 experiment.**

In Figure 5.3, we plot the performance of the individual base learners, as well as the sequential ensemble as M varies. For individual learners, we provide the mean test accuracy with corresponding one standard deviation spread. When $M = 1$, the ensemble and individual model refer to a single base learner and hence their performance is matched. As M grows, we observe significant increase in the performance of our ensemble models with diminishing improvement for higher Ms. The high performance of our sequential ensembles compared to their individual base models further underscores the benefits of ensembling in sequential manner.

5.5 Conclusion and Discussion

In this work, we propose `SeBayS` ensemble, which is an approach to generate sequential Bayesian neural subnetwork ensembles through a combination of novel sequential ensembling approach for BNNs and dynamic sparsity with sparsity-inducing Bayesian prior that provides a simple and effective approach to improve the predictive performance and model robustness. The highly diverse Bayesian neural subnetworks converge to different optima in function space and, when combined, form an ensemble which demonstrates improving performance with increasing ensemble size. Our simple yet highly effective sequential per-

turbation strategy enables a dense BNN ensemble to outperform deterministic dense ensemble. Whereas, the Bayesian neural subnetworks obtained using spike-and-slab node pruning prior produce ensembles that are highly diverse, especially the `SeBays-No Freeze` ensembles compared to EDST in both CIFAR10/100 experiments and DST ensemble in our simpler CIFAR10 experiment. Future work will explore the combination of parallel ensembling of our sequential ensembles leading to a multilevel ensembling model. In particular, we will leverage the exploration phase to reach highly sparse network and next perturb more than once and learn each subnetwork in parallel while performing sequential exploitation phases on each subnetwork. We expect this strategy would lead to highly diverse base learners with potentially significant improvements in model performance and robustness.

APPENDICES

APPENDIX A

REPRODUCIBILITY CONSIDERATIONS

A.1 Hyperparameters

Hyperparameters for single and parallel ensemble models. For the ResNet/CIFAR models, we use minibatch size of 128 uniformly across all the methods. We train each single model (Deterministic, BNN, SSBNN) as well as each member of Dense and DST ensemble for 250 epochs with a learning rate of 0.1 which is decayed by a factor of 0.1 at 150 and 200 epochs. For frequentist methods, we use weight decay of $5e-4$ whereas for Bayesian models the weight decay is 0 (since the KL term in the loss acts as a regularizer). For the DST ensemble, we take the sparsity $S = 0.8$, the update interval $\Delta T = 1000$, and the exploration rate $p = 0.5$, same as Liu et al. (2022).

Hyperparameters for sequential ensemble models. For the ResNet/CIFAR models, the minibatch size is 128 for all the methods compared. We train each sequential model with $M = 3$ for 450 epochs. In the BNN and `SeBayS` ensembles, the exploration phase is run for $t_0 = 150$ epochs and each exploitation phase is run for $t_{\text{ex}} = 100$ epochs. We fix the perturbation factor to be 3. During the exploration phase, we take a high learning rate of 0.1. Whereas, for each exploitation phase, we use learning rate of 0.01 for first $t_{\text{ex}}/2 = 50$ epochs and 0.001 for remaining $t_{\text{ex}}/2 = 50$ epochs. For the EDST ensemble, we take an exploration time (t_{ex}) of 150 epochs, each refinement phase time (t_{re}) of 100 epochs, sparsity $S = 0.8$, and exploration rate $q = 0.8$, same as Liu et al. (2022).

A.2 Data Augmentation

For CIFAR10 and CIFAR100 training dataset, we first pad the train images using 4 pixels of value 0 on all borders and then crop the padded image at a random location generating train images of the same size as the original train images. Next, with a probability

of 0.5, we horizontally flip a given cropped image. Finally, we normalize the images using `mean` $= (0.4914, 0.4822, 0.4465)$ and `standard deviation` $= (0.2470, 0.2435, 0.2616)$ for CIFAR10. Whereas, we use `mean` $= (0.5071, 0.4865, 0.4409)$ and `standard deviation` $= (0.2673, 0.2564, 0.2762)$ for CIFAR100. Next, we split the train data of size 50000 images into a TRAIN/VALIDATION split of 45000/5000 transformed images. For CIFAR10/100 test data, we normalize the 10000 test images in each data case using the corresponding `mean` and `standard deviation` of their respective training data.

A.3 Evaluation Metrics

We quantify the predictive performance of each method using the accuracy of the test data (Acc). For a measure of robustness or predictive uncertainty, we use negative log-likelihood (NLL) calculated on the test dataset. Moreover, we adopt {cAcc, cNLL} to denote the corresponding metrics on corrupted test datasets. We also use VALIDATION data to determine the best epoch in each model which is later used for TEST data evaluation.

In the case of Deterministic and each member of Dense, MIMO, and DST ensemble, we use a single prediction for each test data element and calculate the corresponding evaluation metrics for each individual model. In case of all the Bayesian models, we use one Monte Carlo sample to generate the network parameters and correspondingly generate a single prediction for each single model, which is used to calculate the evaluation metrics in those individual models. For all the ensemble models, we generate a single prediction from each base learner present in the ensemble. Next, we evaluate the ensemble prediction using a simple average of M predictions generated from M base learners and use this averaged prediction to calculate the evaluation metrics mentioned above for the ensemble models.

A.4 Hardware and Software

The Deterministic, MIMO, Rank-1 BNN, and Dense Ensemble models are run using the Uncertainty Baselines (Nado et al., 2021) repository, but with the data, model and

hyperparameter settings described in Section 5.3. Moreover, we consistently run all the experiments on a single NVIDIA A100 GPU for all the approaches evaluated in this work.

APPENDIX B

OUT-OF-DISTRIBUTION EXPERIMENT RESULTS

In Table B.1, we present the AU-ROC results for out-of-distribution (OoD) detection for the ResNet-32/CIFAR10 models. In this case, the out-of-distribution data was taken to be CIFAR100. The results show that our `SebayS`-Freeze Ensemble perform better than the single SSBNN and MIMO model. On the other hand, `SebayS`-No Freeze Ensemble performss better than SSBNN. Next, our BNN sequential ensemble performs better than deterministic and BNN models.

Table B.1 **OoD detection results in ResNet-32/CIFAR10 experiment.** We mark the best results out of single-pass sparse models in bold and single-pass dense models in blue.

Methods	AUROC (↑)	# Forward passes (↓)
SSBNN	0.806	1
MIMO (M=3)	0.840	1
EDST Ensemble (M = 3)	**0.872**	1
SeBayS-Freeze Ensemble (M = 3)	0.864	1
SeBayS-No Freeze Ensemble (M = 3)	0.842	1
DST Ensemble (M = 3)	0.879	3
Deterministic	0.854	1
BNN	0.841	1
Rank-1 BNN (M=3)	**0.866**	1
BNN Sequential Ensemble (M = 3)	0.863	1
Dense Ensemble (M=3)	0.879	3

APPENDIX C

EFFECT OF THE ENSEMBLE SIZE

In Section 5.4.3, we have explored the effect of the ensemble size M in ResNet-32/CIFAR10 experiment through comparison of mean individual learner test accuracies compared to ensemble accuracies in uncorrupted test dataset. In Table C.1, we provide the results on both CIFAR10 and CIFAR10-C datasets for our sequential ensembles with an increasing number of base learners $M = 3, 5, 10$. We also provide BNN and SSBNN baselines to compare against BNN sequential ensemble and `SeBayS` ensembles, respectively. We observe that our BNN sequential ensemble and `SeBayS` ensembles of any size significantly outperform single BNN and SSBNN models, respectively. With an increasing number of base learners ($M = 3, 5, 10$) within each of our sequential ensembles, we observe a monotonically increasing predictive performance. The NLLs for BNN sequential ensemble decrease as M increases. The NLLs for the `SeBayS` ensembles are either similar or increasing as M increases, which suggests the influence of the KL divergence term in ELBO optimization in variational inference.

Table C.1 **Ensemble size effect results in ResNet-32/CIFAR10 experiment.** We mark the best results out of the sparse models in bold and the dense models in blue.

Methods	Acc (↑)	NLL (↓)	cAcc (↑)	cNLL (↓)	# Forward passes (↓)
SSBNN	91.2	0.320	67.5	1.479	1
SeBayS-Freeze Ensemble (M = 3)	92.5	0.273	70.4	**1.344**	1
SeBayS-Freeze Ensemble (M = 5)	92.5	0.273	70.9	1.359	1
SeBayS-Freeze Ensemble (M = 10)	**92.7**	0.275	**71.0**	1.386	1
SeBayS-No Freeze Ensemble (M = 3)	92.4	0.274	69.8	1.356	1
SeBayS-No Freeze Ensemble (M = 5)	92.5	**0.271**	70.2	1.375	1
SeBayS-No Freeze Ensemble (M = 10)	**92.7**	0.272	70.8	1.375	1
BNN	91.9	0.353	71.3	1.422	1
BNN Sequential Ensemble (M = 3)	93.8	0.265	73.3	1.341	1
BNN Sequential Ensemble (M = 5)	94.1	0.253	73.7	1.318	1
BNN Sequential Ensemble (M = 10)	**94.2**	**0.244**	**73.9**	**1.300**	1

APPENDIX D

EFFECT OF THE MONTE CARLO SAMPLE SIZE

In variational inference during evaluation phase, model prediction is calculated using the average of the predictions from ensemble of networks where the weights of each network represent one sample from the posterior distributions of the weights. The number of such networks used to build ensemble prediction is called the Monte Carlo (MC) sample. In Table D.1, we present our sequential ensemble models as well as BNN and SSBNN baselines in the ResNet-32/CIFAR10 experiment. Here, we take $MC = 1$ which is used in the Section 5.3 experiments and compare it with $MC = 5$ for each method. In single BNN and SSBNN models, we observe significant improvement in model performance when using $MC = 5$ instead of 1. However, when we compare the `SebayS` ensembles using $MC = 1$ or 5 with SSBNN using $MC = 5$, we observe that their performance is similar, indicating that $MC = 1$ is sufficient for our `SebayS` ensembles. On the other hand, sequential BNN ensembles using $MC = 1$ has better performance compared to BNN with $MC = 5$. Whereas, sequential BNN ensemble using $MC = 1$ and 5 have similar performance. This highlights the importance of sequential perturbation strategy, which leads to more diverse ensembles compared to mere Monte Carlo sampling.

Table D.1 **Monte Carlo sample size effect results in ResNet-32/CIFAR10 experiment.** We mark the best results out of the sparse models in bold and dense models in blue. MC is the Monte Carlo sample size

Methods	MC	Acc (↑)	NLL (↓)
SSBNN	1	91.2	0.320
SSBNN	5	92.3	0.270
SeBayS-Freeze Ensemble (M=3)	1	92.5	0.273
SeBayS-Freeze Ensemble (M=3)	5	92.5	0.270
SeBayS-No Freeze Ensemble (M=3)	1	92.4	0.274
SeBayS-No Freeze Ensemble (M=3)	5	**92.6**	**0.268**
BNN	1	91.9	0.353
BNN	5	93.2	0.271
BNN Sequential Ensemble (M=3)	1	93.8	0.265
BNN Sequential Ensemble (M=3)	5	**93.9**	**0.254**

APPENDIX E

EFFECT OF THE PERTURBATION FACTOR

In this Appendix, we explore the influence of the perturbation factor on our sequential ensemble models through the ResNet-32/CIFAR10 experiment. In Table E.1, we report the results for our three sequential approaches for three perturbation factors, $\rho = 2, 3, 5$. For our `SeBayS`-Freeze and No Freeze ensembles, the lower perturbations with $\rho = 2$ lead to higher test accuracies and NLLs over $\rho = 3, 5$ in both the CIFAR10 and CIFAR10-C test datasets. This means that higher perturbations $\rho = 3, 5$ might need a higher number of epochs to reach the convergence in each exploitation phase. However, in the BNN sequential ensemble $\rho = 3$ has an overall higher performance compared to $\rho = 2, 5$. This points to the fact that the lower perturbation, $\rho = 2$, may not lead to the best ensemble model. In Table E.2, we present the prediction disagreement and KL divergence metrics for the experiments described in this Appendix. In the BNN sequential ensemble, the $\rho = 5$ perturbation model has the best diversity metrics, whereas the $\rho = 3$ perturbation model has the best accuracy. In the `SeBayS` approach, the perturbation of $\rho = 3$ leads to the best diversity metrics nonetheless at the expense of slightly lower predictive performance. This highlights the fact that the $\rho = 3$ `SeBayS` approaches lead to the best ensembles given the training budget constraint. Hence, we use $\rho = 3$ for our three sequential models in all the experiments presented in the Section 5.3.

Table E.1 **Perturbation factor effect results in ResNet-32/CIFAR10 experiment.** We mark the best results out of different perturbation models under a given method in bold. Ensemble size is fixed at $M = 3$. ρ is the perturbation factor.

Methods	ρ	Acc (↑)	NLL (↓)	cAcc (↑)	cNLL (↓)
SeBayS-Freeze Ensemble	2	**92.7**	**0.264**	**70.6**	**1.303**
SeBayS-Freeze Ensemble	3	92.5	0.273	70.4	1.344
SeBayS-Freeze Ensemble	5	92.5	0.267	**70.6**	1.314
SeBayS-No Freeze Ensemble	2	**92.7**	**0.268**	**70.4**	**1.331**
SeBayS-No Freeze Ensemble	3	92.4	0.274	69.8	1.356
SeBayS-No Freeze Ensemble	5	92.4	0.272	70.1	1.353
BNN Sequential Ensemble	2	93.6	0.269	**73.3**	1.361
BNN Sequential Ensemble	3	**93.8**	0.265	**73.3**	**1.341**
BNN Sequential Ensemble	5	93.6	**0.262**	73.0	1.366

Table E.2 **Diversity metrics for models trained with different perturbation factors in ResNet-32/CIFAR-10 experiment.** We mark the best results out of different perturbation models under a given method in bold. Ensemble size is fixed at $M = 3$. ρ is the perturbation factor.

		ResNet-32/CIFAR10		
Methods	ρ	d_{dis} (↑)	d_{KL} (↑)	Acc (↑)
BNN Sequential Ensemble	2	0.062	0.205	93.6
BNN Sequential Ensemble	3	0.061	0.201	**93.8**
BNN Sequential Ensemble	5	**0.063**	**0.211**	93.6
SeBayS-Freeze Ensemble	2	0.058	0.135	**92.7**
SeBayS-Freeze Ensemble	3	**0.060**	**0.138**	92.5
SeBayS-Freeze Ensemble	5	0.059	0.137	92.5
SeBayS-No Freeze Ensemble	2	0.082	0.222	**92.7**
SeBayS-No Freeze Ensemble	3	**0.106**	**0.346**	92.4
SeBayS-No Freeze Ensemble	5	0.083	0.228	92.4

APPENDIX F

EFFECT OF THE CYCLIC LEARNING RATE SCHEDULE

In this Appendix, we provide the effect of different cyclic learning rate strategies during exploitation phases in our three sequential ensemble methods. We explore the stepwise (our approach), cosine (Huang et al., 2017), linear-fge (Garipov et al., 2018), and linear-1 cyclic learning rate schedules.

Cosine. The cyclic cosine learning rate schedule reduces the higher learning rate of 0.01 to a lower learning rate of 0.001 using the shifted cosine function (Huang et al., 2017) in each exploitation phase.

Linear-fge. In the cyclic linear-fge learning rate schedule, we first drop the high learning rate of 0.1 used in the exploration phase to 0.01 linearly in $t_{ex}/2$ epochs and then further drop the learning rate to 0.001 linearly for the remaining $t_{ex}/2$ epochs during the first exploitation phase. Afterwards, in each exploitation phase, we linearly increase the learning rate from 0.001 to 0.01 for $t_{ex}/2$ and then linearly decrease it back to 0.001 for the next $t_{ex}/2$ similar to Garipov et al. (2018).

Linear-1. In the linear-1 cyclic learning rate schedule, we linearly decrease the learning rate from 0.01 to 0.001 for t_{ex} epochs in each exploitation phase and then suddenly increase the learning to 0.01 after each sequential perturbation step.

In Figure F.1, we present the plots of the cyclic learning rate schedules considered in this Appendix.

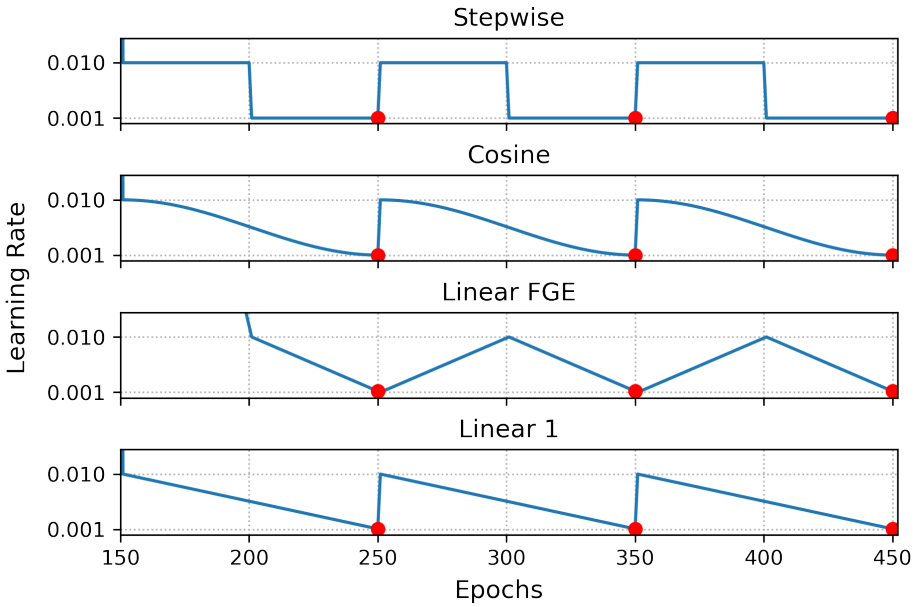


Figure F.1 **Cyclic learning rate schedules.** The red dots represent the converged models after each exploitation phase used in our final sequential ensemble.

In Table F.1, we present the results for our three sequential ensemble methods under the four cyclic learning rate schedules mentioned above. We observe that, in all three sequential ensembles, the cyclic stepwise learning rate schedule yields the best performance in almost all criteria compared to the rest of the learning rate schedules in each sequential ensemble method. In Table F.2, we present the prediction disagreement and KL divergence metrics for the experiments described in this Appendix. We observe that, in `SeBayS`-No Freeze ensemble, cyclic stepwise schedule generates highly diverse subnetworks, which also leads to high predictive performance. Whereas, in the BNN sequential and `SeBayS`-Freeze ensemble, we observe lower diversity metrics for the cyclic stepwise learning rate schedule compared to the rest of the learning rate schedules.

Table F.1 **Cyclic learning rate schedules results in ResNet-32/CIFAR10 experiment.** We mark the best results out of different learning rate (LR) schedules under a given method in bold. Ensemble size is fixed at $M = 3$.

Methods	LR Schedule	Acc (↑)	NLL (↓)	cAcc (↑)	cNLL (↓)
SeBayS-Freeze Ensemble	stepwise	**92.5**	0.273	**70.4**	**1.344**
SeBayS-Freeze Ensemble	cosine	92.3	0.301	69.8	1.462
SeBayS-Freeze Ensemble	linear-fge	**92.5**	**0.270**	70.1	1.363
SeBayS-Freeze Ensemble	linear-1	92.1	0.310	69.8	1.454
SeBayS-No Freeze Ensemble	stepwise	**92.4**	**0.274**	69.8	**1.356**
SeBayS-No Freeze Ensemble	cosine	92.2	0.294	69.9	1.403
SeBayS-No Freeze Ensemble	linear-fge	**92.4**	0.276	**70.0**	1.379
SeBayS-No Freeze Ensemble	linear-1	92.2	0.296	69.7	1.412
BNN Sequential Ensemble	stepwise	**93.8**	**0.265**	**73.3**	**1.341**
BNN Sequential Ensemble	cosine	93.7	0.279	72.7	1.440
BNN Sequential Ensemble	linear-fge	93.5	0.270	73.1	1.342
BNN Sequential Ensemble	linear-1	93.4	0.287	72.2	1.430

Table F.2 **Diversity metrics for models trained with different cyclic learning rate schedules in ResNet-32/CIFAR10 experiment.** We mark the best results out of different learning rate (LR) schedules under a given method in bold. Ensemble size is fixed at $M = 3$.

		ResNet-32/CIFAR10		
Methods	LR Schedule	d_{dis} (↑)	d_{KL} (↑)	Acc (↑)
BNN Sequential Ensemble	stepwise	0.061	0.201	**93.8**
BNN Sequential Ensemble	cosine	0.068	0.256	93.7
BNN Sequential Ensemble	linear-fge	0.070	0.249	93.5
BNN Sequential Ensemble	linear-1	**0.071**	**0.275**	93.4
SeBayS-Freeze Ensemble	stepwise	0.060	0.138	**92.5**
SeBayS-Freeze Ensemble	cosine	0.072	0.204	92.3
SeBayS-Freeze Ensemble	linear-fge	**0.076**	**0.215**	**92.5**
SeBayS-Freeze Ensemble	linear-1	0.074	0.209	92.1
SeBayS-No Freeze Ensemble	stepwise	**0.106**	**0.346**	**92.4**
SeBayS-No Freeze Ensemble	cosine	0.078	0.222	92.2
SeBayS-No Freeze Ensemble	linear-fge	0.074	0.199	**92.4**
SeBayS-No Freeze Ensemble	linear-1	0.077	0.217	92.2

CHAPTER 6

EPILOGUE

6.1 Summary

This dissertation focuses on the development of novel theoretically consistent Bayesian neural networks (BNN) models for wide range of data scenarios. We have proposed: the Bayesian quantile regression neural networks (BQRNN) in Chapter 2, the spike-and-slab Gaussian node selection technique (SS-IG) in Chapter 3, the spike-and-slab group lasso (SS-GL) and the spike-and-slab group horseshoe (SS-GHS) in Chapter 4. In each of BQRNN and SS-IG methods, we provide rigorous theoretical justification via posterior consistency results and the optimal contraction rate. We also provide numerical evidence establishing the advantage of our proposed methods compared to the recent competing techniques in the literature. In Chapter 5, we propose sequential Bayesian neural subnetwork ensembles (SeBayS) which use SS-IG models as the base models in the ensemble. We concluded that chapter with several experiments showcasing the effectiveness of our proposed approach as well as few studies where we explore the effect of changing some parameters in the model.

6.2 Broader Impacts

Our BQRNN approach is particularly useful when the relationships in the lower and upper tail areas of the response variable distribution are of greater interest such as extreme weather events, cascading failures in electric power grids, and other rare events modeling. On the other hand, as deep learning gets harnessed by big industrial corporations in recent years to improve their products, the demand for models with both high predictive and uncertainty estimation performance is rising. The vast variety of its applications range from computer vision, pattern recognition, to natural language processing. However, as deep learning models are pushed into smaller and smaller embedded devices, such as, smart cameras recognizing

visitors at your front door, the design of resource efficient neural networks is of extreme practical importance. These real-world applications demand real-time, on-device neural network inference. Our work on sparse BNNs addresses this computational bottleneck by compressing neural networks by inducing sparsity during training. The Bayesian framework estimates the posterior of model parameters allowing for uncertainty quantification around the parameter estimates which can be vital in medical diagnostics. For example, many of the brain imaging data could be processed through our model yielding a decision on certain medical condition with added benefit of quantified confidence associated with that decision.

6.3 Future Research

In the future, the Bayesian neural networks still have a huge room to investigate. There are few promising research directions which stem from our current work.

- The development of sparse deep Bayesian quantile networks which can allow for extreme quantile inference with fewer data points. Such a model can benefit in cases where the event of interest is rarely manifested in a given data.

- Bayesian convolutional neural network approximation theory which consists of the derivations of posterior consistency, variable selection consistency, and asymptotically optimal generalization bounds.

- The theoretical framework for the Bayesian ensembling including not only the posterior consistency but also the nonasymptotic generalization error upper bounds. Such a bound might depend on the data size as well as explicit number of base models in an ensemble. This theoretical development will also help in deciding the optimal number of base models in a given ensemble.

- Development of sparse Bayesian tensor-to-tensor convolution neural networks involving structured sparsity learning which would potentially benefit ill-posed as well as well-posed problems which can occur for instance in tomographic reconstruction.

BIBLIOGRAPHY

BIBLIOGRAPHY

Alhamzawi, R. (2018). Brq: R package for bayesian quantile regression. https://cran.r-project.org/web/packages/Brq/Brq.pdf. Online; accessed 15 May 2020.

Alvarez, J. M. and Salzmann, M. (2016). Learning the number of neurons in deep networks. In Proceedings of the 29th Advances in Neural Information Processing Systems (NIPS 2016).

Andrews, D. F. and Mallows, C. L. (1974). Scale mixtures of normal distributions. Journal of the Royal Statistical Society – Series B, 36(1):99–102.

Arangio, S. and Bontempi, F. (2015). Structural health monitoring of a cable-stayed bridge with bayesian neural networks. Structure and Infrastructure Engineering, 11(4):575–587.

Bai, J., Song, Q., and Cheng, G. (2020). Efficient variational inference for sparse deep learning with theoretical guarantee. In Proceedings of the 33th Advances in Neural Information Processing Systems (NeurIPS 2020).

Barndorff-Nielsen, O. and Shephard, N. (2001). Non-Gaussian OU based models and some of their uses in financial economics. Journal of the Royal Statistical Society – Series B, 63:167–241.

Barron, A., Schervish, M. J., and Wasserman, L. (1999). The consistency of posterior distributions in nonparametric problems. Annals of Statistics, 10:536–561.

Bateni, S. M., Jeng, D.-S., and Melville, B. W. (2007). Bayesian neural networks for prediction of equilibrium and time-dependent scour depth around bridge piers. Advances in Engineering Software, 38(2):102–111.

Beker, W., Wołos, A., Szymkuć, S., and Grzybowski, B. (2020). Minimal-uncertainty prediction of general drug-likeness based on bayesian neural networks. Nature Machine Intelligence, 2:457–465.

Benoit, D. F., Alhamzawi, R., Yu, K., and den Poel, D. V. (2017). R package 'bayesqr'. https://cran.r-project.org/web/packages/bayesQR/bayesQR.pdf. Online; accessed 15 May 2020.

Betancourt, M. and Girolami, M. (2015). Hamiltonian monte carlo for hierarchical models. Current trends in Bayesian methodology with applications, 79(30).

Bhadra, A., Datta, J., Polson, N., and Willard, B. (2019). Lasso meets horseshoe: A survey. Statistical Science, 34(3):405–427.

Bhattacharya, S. and Maiti, T. (2021). Statistical foundation of variational bayes neural networks. Neural Networks, 137:151–173.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. Journal of the American Statistical Association, 112(518):859–877.

Blei, D. M. and Lafferty, J. D. (2007). A correlated topic model of science. The Annals of Applied Statistics, 1(1):17–35.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural network. In Proceedings of Machine Learning Research, volume 37, pages 1613–1622. PMLR.

Buntine, W. L. and Weigend, A. S. (1991). Bayesian back-propagation. Complex Systems, 5:603–643.

Cannon, A. J. (2011). R package 'qrnn' for quantile regression neural network. https://cran.r-project.org/web/packages/qrnn/qrnn.pdf. Online; accessed 15 May 2020.

Cannon, A. J. (2018). Non-crossing nonlinear regression quantiles by monotone composite quantile regression neural network, with application to rainfall extremes. Stoch Environ Res Risk Assess, 32:3207–3225.

Carvalho, C. M., Polson, N. G., and Scott, J. G. (2010). The horseshoe estimator for sparse signals. Biometrika, 97(2):465–480.

Chen, C. (2007). A finite smoothing algorithm for quantile regression. Journal of Computational and Graphical Statistics, 16:136–164.

Cheng, Y., Wang, D., Zhou, P., and Zhang, T. (2018). Model compression and acceleration for deep neural networks: The principles, progress, and challenges. IEEE Signal Processing Magazine, 35(1):126–136.

Chérief-Abdellatif, B.-E. (2020). Convergence rates of variational inference in sparse deep learning. In Proceedings of the 37th International Conference on Machine Learning (ICML-2020).

Chérief-Abdellatif, B.-E. and Alquier, P. (2018). Consistency of variational bayes inference for estimation and model selection in mixtures. Electronic Journal of Statistics, 12(2):2995–3035.

Cobb, A. D. et al. (2019). An ensemble of bayesian neural networks for exoplanetary atmospheric retrieval. The Astronomical Journal, 158(1).

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. Mathematics

of Controls, Signals, and Systems, 2:303–314.

D'Angelo, F. and Fortuin, V. (2021). Repulsive deep ensembles are bayesian. In Proceedings of the 34th Advances in Neural Information Processing Systems (NeurIPS 2021).

Dantzig, G. B. (1963). Linear Programming and Extensions. Princeton University Press, Princeton.

De Freitas, N., Andrieu, C., Højen-Sørensen, P., Niranjan, M., and Gee, A. (2001). Sequential monte carlo methods for neural networks. In Sequential Monte Carlo Methods in Practice, pages 359—-379. Springer, New York.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Dusenberry, M., Jerfel, G., Wen, Y., Ma, Y., Snoek, J., Heller, K., Lakshminarayanan, B., and Tran, D. (2020). Efficient and scalable bayesian neural nets with rank-1 factors. In Proceedings of the 37th International Conference on Machine Learning (ICML-2020).

Egele, R., Maulik, R., Raghavan, K., Balaprakash, P., and Lusch, B. (2021). Autodeuq: Automated deep ensemble with uncertainty quantification. arXiv preprint arXiv:2110.13511.

Elfwing, S., Uchibe, E., and Doya, K. (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. Neural Networks, 107:3–11. Special issue on deep reinforcement learning.

Frankle, J. and Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In 7th International Conference on Learning Representations (ICLR-2019).

Friedman, J., Hastie, T., and Tibshirani, R. (2009). The elements of statistical learning. Springer series in statistics. Springer, New York.

Funahashi, K. (1989). On the approximate realization of continuous mappings by neural networks. Neural Networks, 2:183–192.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Proceedings of the 33rd International Conference on Machine Learning (ICML-2016.

Gale, T., Elsen, E., and Hooker, S. (2019). The state of sparsity in deep neural networks. arXiv preprint arXiv:1902.09574.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. (2018). Loss surfaces, mode connectivity, and fast ensembling of dnns. In Proceedings of the 31st Advances in Neural Information Processing Systems (NeurIPS-2018).

Gelfand, E. E. and Smith, A. F. M. (1990). Sampling-based approaches to calculating marginal densities. Journal of the American Statistical Association, 85:398–409.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). Bayesian Data Analysis. CRC Press, Third edition.

Gelman, A. and Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. Statistical Science, 7(4):457–472.

Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 6(6):721–741.

Ghosal, S. and Van Der Vaart, A. W. (2007). Convergence rates of posterior distributions for noniid observations. The Annals of Statistics, 35(1):192–223.

Ghosh, M., Ghosh, A., Chen, M. H., and Agresti, A. (2000). Noninformative priors for one-parameter item models. Journal of Statistical Planning and Inference, 88:99–115.

Ghosh, S., Yao, J., and Doshi-Velez, F. (2019). Model selection in bayesian neural networks via horseshoe priors. Journal of Machine Learning Research, 20:1–46.

Grenander, U. (1981). Abstract Inference. Wiley.

Guo, Y., Yao, A., and Chen, Y. (2016). Dynamic network surgery for efficient dnns. In Proceedings of the 29th Advances in Neural Information Processing Systems (NIPS 2016).

Han, S., Mao, H., and Dally, W. J. (2016). Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In 4th International Conference on Learning Representations (ICLR-2016).

Han, S., Pool, J., Tran, J., and Dally, W. (2015). Learning both weights and connections for efficient neural network. In Proceedings of the 28th Advances in Neural Information Processing Systems (NIPS 2015).

Hansen, L. and Salamon, P. (1990). Neural network ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(10):993–1001.

Hassibi, B. and Stork, D. G. (1993). Second order derivatives for network pruning: Optimal brain surgeon. In Advances in Neural Information Processing Systems.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. Biometrika, 57:97–109.

Havasi, M., Jenatton, R., Fort, S., Liu, J. Z., Snoek, J., Lakshminarayanan, B., Dai, A. M.,

and Tran, D. (2021). Training independent subnetworks for robust prediction. In 9th International Conference on Learning Representations (ICLR-2021).

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In IEEE International Conference on Computer Vision (ICCV-2015), pages 1026–1034.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR-2016), pages 770–778.

Hendrycks, D. and Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations. In 7th International Conference on Learning Representations (ICLR-2019).

Hernandez-Lobato, J. M. and Adams, R. (2015). Probabilistic backpropagation for scalable learning of bayesian neural networks. In Proceedings of the 32nd International Conference on Machine Learning (ICML-2015).

Hinton, G. E. and Van Camp, D. (1993). 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-1993), pages 5–13.

Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., and Peste, A. (2021). Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. Journal of Machine Learning Research, 22(241):1–124.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. Neural Networks, 2:359–366.

Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E., and Weinberger, K. Q. (2017). Snapshot ensembles: Train 1, get m for free. In 5th International Conference on Learning Representations (ICLR-2017).

Ingraham, J. B. and Marks, D. S. (2017). Variational inference for sparse and undirected models. In Proceedings of the 34th International Conference on Machine Learning (ICML-2017).

Izmailov, P., Vikram, S., Hoffman, M. D., and Wilson, A. G. G. (2021). What are bayesian neural network posteriors really like? In Proceedings of the 38th International Conference on Machine Learning (ICML-2021).

Jang, E., Gu, S., and Poole, B. (2017). Categorical reparameterization with gumbel-softmax. In 5th International Conference on Learning Representations (ICLR-2017).

Jantre, S., Bhattacharya, S., and Maiti, T. (2021a). Layer adaptive node selection in bayesian neural networks: Statistical guarantees and implementation details. arXiv preprint arXiv:2108.11000.

Jantre, S., Bhattacharya, S., and Maiti, T. (2021b). Quantile regression neural networks: a bayesian approach. Journal of Statistical Theory and Practice, 15(3):1–34.

Jantre, S., Madireddy, S., Bhattacharya, S., Maiti, T., and Balaprakash, P. (2022). Sequential bayesian neural subnetwork ensembles. arXiv preprint arXiv:2206.00794.

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2015). On using very large target vocabulary for neural machine translation. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1–10.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Sau, L. K. (1999). An introduction to variational methods for graphical models. Machine Learning, 37:183–233.

Karmarkar, N. (1984). A new polynomial time algorithm for linear programming. Combinatorica, 4(4):373–395.

Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? In Proceedings of the 31st Advances in Neural Information Processing Systems (NIPS-2017).

Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In 3rd International Conference on Learning Representations (ICLR-2015).

Kingma, D. and Welling, M. (2014). Auto-encoding variational bayes. In 2nd International Conference on Learning Representations (ICLR-2014).

Koenker, R. (2005). Quantile Regression. Cambridge University Press, Cambridge, First edition.

Koenker, R. (2017). R package 'quantreg' for quantile regression. https://cran.r-project.org/web/packages/quantreg/quantreg.pdf. Online; accessed 15 May 2020.

Koenker, R. and Basset, G. (1978). Regression quantiles. Econometrica, 46:33–50.

Koenker, R. and Machado, J. (1999). Goodness of fit and related inference processes for quantile regression. Journal of the American Statistical Association, 94:1296–1309.

Kottas, A. and Gelfand, A. E. (2001). Bayesian semiparametric median regression modeling. Journal of the American Statistical Association, 96:1458–1468.

Kozumi, H. and Kobayashi, G. (2011). Gibbs sampling methods for bayesian quantile regression. Journal of Statistical Computation and Simulation, 81:1565–1578.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. M.Sc. Thesis.

Kwon, Y., Won, J.-H., Kim, B. J., and Paik, M. C. (2020). Uncertainty quantification using bayesian neural networks in classification: Application to biomedical image segmentation. Computational Statistics and Data Analysis, 142.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In Proceedings of the 30th Advances in Neural Information Processing Systems (NIPS 2017).

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. Nature, 521:436–444.

LeCun, Y., Denker, J. S., and Solla, S. A. (1990). Optimal brain damage. In Proceedings of the 3rd Advances in Neural Information Processing Systems (NIPS-1990).

Lee, H. K. H. (2000). Consistency of posterior distributions for neural networks. Neural Networks, 13:629–642.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. (2018). Deep neural networks as gaussian processes. In 6th International Conference on Learning Representations (ICLR-2018).

Liu, S., Chen, T., Atashgahi, Z., Chen, X., Sokar, G., Mocanu, E., Pechenizkiy, M., Wang, Z., and Mocanu, D. C. (2022). Deep ensembling with no overhead for either training or testing: The all-round blessings of dynamic sparsity. In 10th International Conference on Learning Representations (ICLR-2022).

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. (2017). Learning efficient convolutional networks through network slimming. In Proceedings of the IEEE International Conference on Computer Vision (ICCV).

Louizos, C., Ullrich, K., and Welling, M. (2017). Bayesian compression for deep learning. In Proceedings of the 30th Advances in Neural Information Processing Systems (NIPS 2017).

Louizos, C., Welling, M., and Kingma, D. P. (2018). Learning sparse neural networks through l_0 regularization. In 6th International Conference on Learning Representations (ICLR-2018).

Lu, L., Shin, Y., Su, Y., and Em Karniadakis, G. (2020). Dying relu and initialization: Theory and numerical examples. Communications in Computational Physics, 28(5):1671–1706.

Luo, J.-H., Wu, J., and Lin, W. (2017). Thinet: A filter level pruning method for deep neural network compression. In Proceedings of the IEEE International Conference on Computer Vision (ICCV).

Mackay, D. J. C. (1992). A practical bayesian framework for backpropagation networks. Neural Computation, 4:448–472.

Maddison, C. J., Mnih, A., and Teh, Y. W. (2017). The concrete distribution: A continuous relaxation of discrete random variables. In 5th International Conference on Learning Representations (ICLR-2017).

Madsen, K. and Nielsen, H. B. (1993). A finite smoothing algorithm for linear l_1 estimation. SIAM Journal of Optimization, 3:223–235.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. Journal of Chemical Physics, 21:1087–1092.

Mitchell, T. J. and Beauchamp, J. J. (1988). Bayesian variable selection in linear regression. Journal of the American Statistical Association, 83(404):1023–1032.

Mocanu, D. C., Mocanu, E., Stone, P., Nguyen, P. H., Gibescu, M., and Liotta, A. (2018). Evolutionary training of sparse artificial neural networks: A network science perspective. Nature Communication, 9(2383).

Moghimi, M., Belongie, S. J., Saberian, M. J., Yang, J., Vasconcelos, N., and Li, L.-J. (2016). Boosted convolutional neural networks. In BMVC, volume 5, page 6.

Molchanov, D., Ashukha, A., and Vetrov, D. (2017). Variational dropout sparsifies deep neural networks. In Proceedings of the 34th International Conference on Machine Learning (ICML-2017).

Mozer, M. C. and Smolensky, P. (1988). Skeletonization: A technique for trimming the fat from a network via relevance assessment. In Proceedings of the 1st Advances in Neural Information Processing Systems (NIPS-1988).

Murray, K. and Chiang, D. (2015). Auto-sizing neural networks: With applications to n-gram language models. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015), pages 908–916.

Nado, Z. et al. (2021). Uncertainty Baselines: Benchmarks for uncertainty & robustness in deep learning. In Bayesian Deep Learning workshop, NeurIPS-2021.

Neal, R. (1992). Bayesian learning via stochastic dynamics. In Proceedings of the 5th Advances in Neural Information Processing Systems (NIPS-1992).

Neal, R. M. (1996). Bayesian Learning for Neural Networks. New York: Springer Verlag.

Neklyudov, K., Molchanov, D., Ashukha, A., and Vetrov, D. P. (2017). Structured bayesian pruning via log-normal multiplicative noise. In Proceedings of the 33rd Advances in Neural Information Processing Systems (NeurIPS-2020).

Ochiai, T., Matsuda, S., Watanabe, H., and Katagiri, S. (2017). Automatic node selection for deep neural networks using group lasso regularization. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Dillon, J. V., Lakshminarayanan, B., and Snoek, J. (2019). Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In Proceedings of the 32th Advances in Neural Information Processing Systems (NeurIPS-2019).

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):425–442.

Paszke, A. et al. (2019). PyTorch: An imperative style, high-performance deep learning library. In Proceedings of the 32nd Advances in Neural Information Processing Systems (NeurIPS-2019).

Pati, D., Bhattacharya, A., and Yang, Y. (2018). On statistical optimality of variational bayes. In Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS-2018).

Perreault Levasseur, L., Hezaveh, Y. D., and Wechsler, R. H. (2017). Uncertainties in parameters estimated with neural networks: Application to strong gravitational lensing. The Astrophysical Journal, 850(1).

Piironen, J. and Vehtari, A. (2017). On the hyperprior choice for the global shrinkage parameter in the horseshoe prior. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS-2017).

Pollard, D. (1991). Bracketing methods in statistics and econometrics. In Nonparametric and semiparametric methods in econometrics and statistics: Proceedings of the Fifth International Symposium in Econometric Theory and Econometrics, pages 337–355. Cambridge, UK: Cambridge University Press.

Polson, N. and Ročková, V. (2018). Posterior concentration for sparse deep learning. In Proceedings of the 31st Advances in Neural Information Processing Systems (NeurIPS-2018).

Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for activation functions. arXiv preprint arXiv:1710.05941.

Rosenblatt, F. (1957). The Perceptron - a perceiving and recognizing automaton (project para). Cornell Aeronautical Laboratory.

Rosenblatt, F. (1961). Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical Report, Cornell Aeronautical Laboratory.

Schmidt-Hieber, J. (2020). Nonparametric regression using deep neural networks with relu activation function. The Annals of Statistics, 48(4):1875–1897.

Schwartz, L. (1965). On bayes procedures. Z. Wahrsch. Verw. Gebiete, 4:10–26.

Sennrich, R., Haddow, B., and Birch, A. (2016). Edinburgh neural machine translation systems for WMT 16. In Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers.

Singh, S., Hoiem, D., and Forsyth, D. (2016). Swapout: Learning an ensemble of deep architectures. Proceedings of the 29th Advances in Neural Information Processing Systems (NIPS-2016).

Sriram, K., Ramamoorthi, R. V., and Ghosh, P. (2013). Posterior consistency of bayesian quantile regression based on the misspecified asymmetric laplace density. Bayesian Analysis, 8(2):479–504.

Sun, Y., Song, Q., and Liang, F. (2021). Consistent sparse deep learning: Theory and computation. Journal of the American Statistical Association, pages 1–15.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In Proceedings of the 30th International Conference on Machine Learning (ICML-2013).

Swann, A. and Allinson, N. (1998). Fast committee learning: Preliminary results. Electronics Letters, 34(14):1408–1410.

Taylor, J. W. (2000). A quantile regression neural network approach to estimating the conditional density of multiperiod returns. Journal of Forecasting, 19(4):299–311.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society – Series B, 58:267–288.

Titterington, D. M. (2004). Bayesian methods for neural networks and related models. Statistical Science, 19:128–139.

Van Der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. Journal of Machine Learning Research, 9(86):2579–2605.

Van Der Vaart, A. W. and Wellner, J. A. (1996). Weak convergence and empirical processes. Springer, First edition.

Venables, W. N. and Ripley, B. D. (2002). Modern Applied Statistics with S. Springer, Fourth edition.

Walker, S. G. and Mallick, B. K. (1999). A bayesian semiparametric accelerated failure time model. Biometrics, 55:477–483.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In Proceedings of the 30th International Conference on Machine Learning (ICML-2013).

Wasserman, L. (1998). Asymptotic properties of nonparametric bayesian procedures. In Practical nonparametric and semiparametric Bayesian statistics, pages 293–304. New York: Springer.

Wen, W., Wu, C., Wang, Y., Chen, Y., and Li, H. (2016). Learning structured sparsity in deep neural networks. In Proceedings of the 29th Advances in Neural Information Processing Systems (NIPS-2016).

Wen, Y., Tran, D., and Ba, J. (2020). Batchensemble: an alternative approach to efficient ensemble and lifelong learning. 8th International Conference on Learning Representations (ICLR-2020).

Wenzel, F., Snoek, J., Tran, D., and Jenatton, R. (2020). Hyperparameter ensembles for robustness and uncertainty quantification. Proceedings of the 33rd Advances in Neural Information Processing Systems (NeurIPS-2020).

Wilson, A. G. and Izmailov, P. (2020). Bayesian deep learning and a probabilistic perspective of generalization. Proceedings of the 33rd Advances in Neural Information Processing Systems (NeurIPS-2020).

Wong, W. H. and Shen, X. (1995). Probability inequalities for likelihood ratios and convergence rates of sieve mles. Annals of Statistics, 23:339–362.

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

Xie, J., Xu, B., and Chuang, Z. (2013). Horizontal and vertical ensemble with deep representation for classification. arXiv preprint arXiv:1306.2759.

Xu, Q., Deng, K., Jiang, C., Sun, F., and Huang, X. (2017). Composite quantile regression neural network with applications. Expert Systems with Applications, 76:129–139.

Xu, X. and Ghosh, M. (2015). Bayesian variable selection and estimation for group lasso. Bayesian Analysis, 10(4):909–936.

Yeh, I.-C. (1998). Modeling of strength of high performance concrete using artificial neural networks. Cement and Concrete Research, 28(12):1797–1808.

Yu, K. and Moyeed, R. A. (2001). Bayesian quantile regression. Statistics and Probability Letters, 54(4):437–447.

Yu, K. and Zhang, J. (2005). A three-parameter asymmetric laplace distribution and its extensions. Communications in Statistics- Theory and Methods, 34:1867–1879.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. In 5th International Conference on Learning Representations (ICLR-2017).

Zhang, F. and Gao, C. (2020). Convergence rates of variational posterior distributions. The Annals of Statistics, 48(4):2180–2207.

Zhao, C., Ni, B., Zhang, J., Zhao, Q., Zhang, W., and Tian, Q. (2019). Variational convolutional neural network pruning. In 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

Zhong, Z., Zheng, L., Kang, G., Li, S., and Yang, Y. (2020). Random erasing data augmentation. Proceedings of the AAAI Conference on Artificial Intelligence, 34(7):13001–13008.

Zhu, M. and Gupta, S. (2018). To prune, or not to prune: Exploring the efficacy of pruning for model compression. In 6th International Conference on Learning Representations (ICLR-2018), Workshop Track Proceedings.