SPARSE DEEP LEARNING AND STOCHASTIC NEURAL NETWORK

by

Yan Sun

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

Department of Statistics

West Lafayette, Indiana

August 2022

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Faming Liang, Chair

Distinguished Professor of Statistics

Dr. Xiao Wang

Professor of Statistics

Dr. Chuanhai Liu

Professor of Statistics

Dr. Vinayak Rao

Associate Professor of Statistics

Approved by:

Dr. Jun Xie

To my beloved family.

ACKNOWLEDGMENTS

First and foremost, I want to express my sincere gratitude to my advisor, Dr. Faming Liang, for his invaluable guidance and support over the years. Dr. Liang's guidance not only contribute to our research project, but also help me grow more professionally. His passion on our research has always inspired me to work hard and achieve more. This dissertation would not be possible without his enormous time and energy spent on me. I would also like to thank Dr. Qifan Song for his support and contribution on the theoretical development of our research. And I would like to thank Dr. Chuanhai Liu, Dr. Xiao Wang, Dr. Vinayak Rao who generously served as my advisory committee members. Their insightful suggestions and comments provided different perspectives and inspired me to think deeper.

I want to express sincere appreciation to my friends at Purdue, Mao Ye, Xinlin Tao, Chuan Zuo, Wei Deng, Peiyi Zhang, Siqi Liang, Sehwan Kim, Xinyi Pei, Zhanyu Wang, and many others. Thank you for the fun time together and the discussion, collaboration, exchanging of thoughts in both study and life.

To my parents and beloved family, thank you for your unconditional love, I would not have been achieve anything without your support.

TABLE OF CONTENTS

LIST OF TABLES 8

LIST OF FIGURES 9

ABSTRACT 10

1 INTRODUCTION 11
- 1.1 Sparse Deep Learning 12
- 1.2 Kernel-Expanded Stochastic Neural Network 14

2 SPARSE DEEP LEARNING 15
- 2.1 Bayesian Sparse DNNs with mixture Gaussian Prior 15
- 2.2 Posterior Consistency 16
- 2.3 Consistency of DNN Structure Selection 21
  - 2.3.1 Marginal Posterior Inclusion Probability Approach 22
  - 2.3.2 Laplace Approximation of Marginal Posterior Inclusion Probabilities 23
- 2.4 Asymptotic Normality of Connection Weights 26
  - 2.4.1 Asymptotic Normality of Prediction 27
- 2.5 Asymptotically Optimal Generalization Bound 28
- 2.6 Computation 31
  - 2.6.1 Bayesian Evidence Approach 31
  - 2.6.2 Prior Annealing: Frequentist Computation 34
  - 2.6.3 Prior Annealing: Bayesian Computation 36
  - 2.6.4 Construct Confidence Interval 39
- 2.7 Numerical Experiments 41
  - 2.7.1 Synthetic Example 41
  - 2.7.2 Real Data Example 42
  - 2.7.3 Experimental Setups 46
    - Synthetic Example 46
    - Real Data Examples 47

2.8 Discussion 48
2.9 Technical Proofs 50
2.9.1 Proofs on Posterior Consistency 50
Basic Formulas of Bayesian Neural Networks 50
Normal Regression. 50
Logistic Regression. 50
Posterior Consistency of General Statistical Models 51
General Shrinkage Prior Settings for Deep Neural Networks 52
Proof of Theorem 2.2.1 55
2.9.2 Proofs on Structure Selection Consistency 56
Proof of Theorem 2.3.1 56
Proof of Theorem 2.3.2 56
Proof of Lemma 2.3.1 60
Verification of the Bounded Gradient Condition in Theorem 2.3.2 . . 61
Approximation of Bayesian Evidence 63
2.9.3 Proofs of Asymptotic Normality 64
Proof of Theorem 2.4.1 64
Proof of Theorem 2.4.2 70
2.9.4 Proofs on Generalization Bounds 71
Proof of Theorem 2.5.1 71
Proof of Theorem 2.5.2 72
Proof of Theorems 2.5.3 and 2.5.4 73
2.9.5 Mathematical facts of sparse DNN 73
2.9.6 Proof of Theorem 2.6.1 76

3 A KERNEL-EXPANDED STOCHASTIC NEURAL NETWORK 78
3.1 A Kernel-Expanded Stochastic Neural Network 78
3.1.1 A Kernel-Expanded Neural Network 78
3.1.2 A Kernel-Expanded StoNet as an Approximator to KNN 81
3.1.3 The Imputation-Regularized Optimization Algorithm 85

3.1.4 Hyperparameter Setting 91
3.2 Illustrative Examples 92
3.2.1 A full row rank example 92
3.2.2 A measurement error example 95
3.3 Real Data Examples 99
3.3.1 QSAR Androgen Receptor 99
3.3.2 MNIST Data 101
3.3.3 CoverType Data 103
3.3.4 More UCI Datasets 104
3.4 Prediction Uncertainty Quantification with K-StoNet 106
3.4.1 A Recursive Formula for Uncertainty Quantification 106
3.4.2 A Numerical Example 108
3.5 Parameter Settings for K-StoNet 108
3.6 Discussion 111
3.7 Technical Proofs 114
3.7.1 Proof of Theorem 3.1.2 114
3.7.2 Proof of Lemma 3.1.2 115

REFERENCES 117

LIST OF TABLES

2.1 Simulation Result: MSFE and MSPE were calculated by averaging over 10 datasets, and their standard deviations were given in the parentheses. 42

2.2 ResNet network pruning results for CIFAR-10 data, which were calculated by averaging over 3 independent runs with the standard deviation reported in the parentheses. 45

2.3 ResNet network pruning results for CIFAR-10 data, which were calculated by averaging over 3 independent runs with the standard deviation reported in the parentheses. 46

3.1 Performance of the K-StoNet model with different values of ε, where the model was evaluated at the last iteration, #SV represents the average number of support vectors selected by the SVRs at the first hidden layer, and the number in the parentheses represents the standard deviation of the average. 99

3.2 Training and prediction accuracy(%) for QSAR androgen receptor data, where "T" and "P" denote the training and prediction accuracy, respectively. 102

3.3 Training and prediction accuracy(%) for CoverType, where "T" and "P" denote the training and prediction. 104

3.4 Average test RMSE (and its standard error) by variational inference (VI, [108]), probabilistic back-propagation (PBP, [107]), dropout (Dropout, [106]), SGD via back-propagation (BP), and KNN, where N denotes the dataset size and p denotes the input dimension. For each dataset, the boldfaced values are the best result or the second best result if it is insignificantly different from the best one according to a t-test with a significance level of 0.05. 105

LIST OF FIGURES

2.1 Negative logarithm of the mixture Gaussian prior. 35

2.2 Negative log-prior and $\pi(\gamma = 1|\beta)$ for different choices of $\sigma_{0,n}^2$ and λ_n. 37

2.3 Prediction intervals of 20 testing points, where the y-axis is the response value, the x-axis is the index, and the blue point represents the true observation. . . . 43

3.1 An illustrative plot of K-StoNet 82

3.2 *Upper Panel*: paths of the mean squared error (MSE) produced by K-StoNet and an unregularized DNN for one simulated dataset; and *Lower Panel*: best MSE (by the current epoch) produced by SGD for a regularized DNN and K-StoNet over 10 runs. 94

3.3 MSE paths produced by two K-StoNets, one KNN, and two DNNs for the data generated from a KNN model: the left plot is for training and the right plot is for testing. 96

3.4 MSE paths produced by K-StoNets and DNNs: (upper) one-hidden-layer networks; (lower) three-hidden-layer networks. 98

3.5 Training and prediction accuracy paths (along with epochs) produced by K-StoNet, KNN and DNN in one fold of the cross-validation experiment for the QSAR androgen receptor data. 100

3.6 Training and prediction accuracy paths (along with computational time) produced by K-StoNets, KNN and DNNs in one fold of the cross-validation experiment for the QSAR androgen receptor data. 101

3.7 Training and test accuracy versus epochs produced by K-StoNet and DNN (LeNet-300-100) for the MNIST data, where K-StoNet achieved a prediction accuracy of 98.87%, and LeNet-300-100 achieved a prediction accuracy of 98.38%. 103

3.8 95% prediction intervals produced by K-StoNet for 20 test points, where the x-axis indexes the test points, the y-axis represents the response value, and the blue star represents the true observation. 109

ABSTRACT

Deep learning has achieved state-of-the-art performance on many machine learning tasks. But the deep neural network(DNN) model still suffers a few issues. Over-parametrized neural network generally has better optimization landscape, but it is computationally expensive, hard to interpret and the model usually can not correctly quantify the prediction uncertainty. On the other hand, small DNN model could suffer from local trap and will be hard to optimize. In this dissertation, we tackle these issues from two directions, sparse deep learning and stochastic neural network.

For sparse deep learning, we proposed Bayesian neural network(BNN) model with mixture of normal prior. Theoretically, We established the posterior consistency and structure selection consistency, which ensures the sparse DNN model can be consistently identified. We also demonstrate the asymptotic normality of the prediction, which ensures the prediction uncertainty to be correctly quantified. Computationally, we proposed a prior annealing approach to optimize the posterior of BNN. The proposed methods share similar computation complexity to the standard stochastic gradient descent method for training DNN. Experiment results show that our model performs well on high dimensional variable selection as well as neural network pruning.

For stochastic neural network, we proposed a Kernel-Expanded Stochastic Neural Network model or K-StoNet model in short. We reformulate the DNN as a latent variable model and incorporate support vector regression (SVR) as the first hidden layer. The latent variable formulation breaks the training into a series of convex optimization problems and the model can be easily trained using the imputation-regularized optimization (IRO) algorithm. We provide theoretical guarantee for convergence of the algorithm and the prediction uncertainty quantification. Experiment results show that the proposed model can achieve good prediction performance and provide correct confidence region for prediction.

1. INTRODUCTION

During the past decade, the deep neural network (DNN) has achieved great successes in solving many complex machine learning tasks such as pattern recognition and natural language processing. However, the DNN model still suffers a few issues. The DNNs used in practice may consist of hundreds of layers and millions of parameters, see e.g. [1] on image classification. Most of those DNNs are severely over-parametrized. For example, [2] showed that in some networks, only 5% of the parameters are enough to achieve acceptable models. Training and operation of DNNs of this scale entail formidable computational challenges. Over-parameterization also makes the DNN model less interpretable and miscalibrated [3], which can cause serious issues in human-machine trust and thus hinder applications of artificial intelligence (AI) in human life.

On the other hand, training a small DNN model from scratch often performs worse than the over-parametrized DNN model [4]. A line of researches have been done towards understanding the optimization landscape and training process of DNN model. For example, [5] and [6] studied the training loss surface of over-parameterized DNNs. They showed that for a fully connected DNN, almost all local minima are globally optimal, if the width of one layer of the DNN is no smaller than the training sample size and the network structure from this layer on is pyramidal. Recently, [7]–[9] and [10] explored the convergence theory of the gradient-based algorithms in training over-parameterized DNNs. They showed that the gradient-based algorithms with random initialization can converge to global minima provided that the width of the DNN is polynomial in training sample size. A small DNN model does not enjoy those good property of optimization landscape or training process. It can suffer from local trap and be hard to optimize.

In this dissertation, we tackle these issues from two directions. First, we consider sparse deep learning. Sparse deep learning start with over-parametrized DNN model, then identify a sparse model with most parameters being zero and can perform as good as a dense one. Starting with over-parametrization allows the model to enjoy good optimization property, while the sparse model can be easier to interpret and well calibrated. Using the sparse model for future prediction can also save computation cost. From another direction, we propose a so-called kernel-expanded stochastic neural network (K-StoNet) model, which incorporates support vector regression (SVR) as the first hidden layer and reformulates the neural network as a latent variable model. The former maps the input vector into an infinite dimensional feature space via a radial basis function (RBF) kernel, ensuring absence of local minima on its training loss surface. The latter breaks the high-dimensional nonconvex neural network training problem into a series of low-dimensional convex optimization problems, and enables its prediction uncertainty easily assessed. For both directions, we provide theoretical guarantee for the proposed model and demonstrate the performance on synthetic and real data sets. The remaining part of this dissertation is organized as follows. In Section 1.1, we introduce the background of sparse deep learning and our proposed approach. In Section 1.2, we review stochastic neural network model and introduce our K-StoNet model. The subsequent two chapters, Chapter 2 and 3 contains detailed formulation of the model, theoretical properties and experiment results. Discussion and technical proofs are given at the end of each chapter.

1.1 Sparse Deep Learning

The desire to identify sparse model naturally lead to two questions: (i) Is a sparsely connected DNN able to approximate the target mapping with a desired accuracy? and (ii) how to train and determine the structure of a sparse DNN? There have been some work in the literature trying to address these questions.

The approximation power of sparse DNNs has been studied in the literature from both frequentist and Bayesian perspectives. From the frequentist perspective, [11] quantifies the minimum network connectivity that guarantees uniform approximation rates for a class of affine functions; and [12] and [13] characterize the approximation error of a sparsely connected neural network for Hölder smooth functions. From the Bayesian perspective, [14] established posterior consistency for Bayesian shallow neural networks under mild conditions; and [15] established posterior consistency for Bayesian DNNs but under some restrictive conditions such as a spike-and-slab prior is used for connection weights, the activation function is ReLU, and the number of input variables keeps at an order of $O(1)$ while the sample size grows to infinity.

The existing methods for learning sparse DNNs are usually developed separately from the approximation theory. For example, [16], [17] and [18] developed some regularization methods for learning sparse DNNs; [19] showed that dropout training is approximately equivalent to an L_2-regularization; [20] introduced a deep compression pipeline, where pruning, trained quantization and Huffman coding work together to reduce the storage requirement of DNNs; [21] proposed a sparse decomposition method to sparsify convolutional neural networks (CNNs); [22] considered a lottery ticket hypothesis for selecting a sparse subnetwork; and [23] proposed to learn Bayesian sparse neural networks via node selection with a horseshoe prior under the framework of variational inference. For these methods, it is generally unclear if the resulting sparse DNN is able to provide a desired approximation accuracy to the true mapping and how close in structure the sparse DNN is to the underlying true DNN.

In this dissertation, we proposed a Bayesian Neural Network(BNN) with mixture of normal prior. Theoretically, we first establish posterior consistency for the BNN and consistency of structure selection based on the marginal posterior inclusion probabilities, which ensures the posterior will concentrate around the true model. Then we establish consistency of the sparsified DNN via Laplace approximation to the marginal posterior inclusion probabilities, which ensures the sparse structure can be consistently identified by finding maximum of the posterior distribution. To quantify the prediction uncertainty of the model, we established the Bernstein-von Mises (BvM) theorem for network prediction. Computationally, we provide prior annealing algorithm to learn the sparse neural network model. Our proposed learning algorithm shares similar computation cost as standard stochastic gradient descent(SGD) method. Our numerical results indicate that the proposed models can work very well for large-scale DNN compression and high-dimensional nonlinear variable selection. In addition to the mixture of normal prior, we will also discuss other possible choice of priors and their properties from both theoretical and computational perspective.

1.2 Kernel-Expanded Stochastic Neural Network

Introducing noise in neural network training or stochastic neural network model has also been an promising approach to improve performance of neural network. Famous examples include deep belief networks [24] and deep Boltzmann machines [25], which have ever advanced the development of machine learning. Recently, some researchers have proposed to add noise to the DNN to improve its performance. For example, [26] proposed the dropout method to prevent the DNN from over-fitting by randomly dropping some hidden and visible units during training; [27] proposed to add gradient noise to improve training; and [28]–[30] proposed to use stochastic activations through adding noise to improve generalization and adversarial robustness. However, these methods are usually not systematic and theoretical guarantees are hard to be provided.

In this dissertation, we propose a new neural network model, the so-called kernel-expanded stochastic neural network (K-StoNet). The new model incorporates support vector regression (SVR) [31], [32] as the first hidden layer and reformulates the neural network as a latent variable model. The former maps the input vector from its original space into an infinite dimensional feature space, ensuring all local minima on the loss surface are globally optimal. The latter resolves the parameter optimization and statistical inference issues associated with the neural network: it breaks the high-dimensional nonconvex neural network training problem into a series of low-dimensional convex optimization problems, and enables the prediction uncertainty easily assessed. The new model can be easily trained using the imputation-regularized optimization (IRO) algorithm [33], which converges very fast, usually within a small number of epochs. Moreover, the introduction of the SVR layer with a universal kernel [34], [35] enables K-StoNet to work with a smaller network, while ensuring the universal approximation capability.

Compared to existing stochastic neural network model, K-StoNet is developed under a rigorous statistical framework, whose convergence to the global optimum is asymptotically guaranteed and whose prediction uncertainty can be easily assessed.

2. SPARSE DEEP LEARNING

2.1 Bayesian Sparse DNNs with mixture Gaussian Prior

Let $D_n = (\boldsymbol{x}^{(\mathrm{i})}, y^{(\mathrm{i})})_{\mathrm{i}=1,\ldots,n}$ denote a training dataset of n i.i.d observations, where $\boldsymbol{x}^{(\mathrm{i})} \in R^{p_n}$, $y^{(\mathrm{i})} \in R$, and p_n denotes the dimension of input variables and is assumed to grow with the training sample size n. We first study the posterior approximation theory of Bayesian sparse DNNs under the framework of generalized linear models, for which the distribution of y given $\boldsymbol{x}$ is given by

$$f(y|\mu^*(\boldsymbol{x})) = \exp\{A(\mu^*(\boldsymbol{x}))y + B(\mu^*(\boldsymbol{x})) + C(y)\},$$

where $\mu^*(\boldsymbol{x})$ denotes a nonlinear function of $\boldsymbol{x}$, and $A(\cdot)$, $B(\cdot)$ and $C(\cdot)$ are appropriately defined functions. The theoretical results presented in this work mainly focus on logistic regression models and normal linear regression models. For logistic regression, we have $A(\mu^*) = \mu^*$, $B(\mu^*) = -\log(1 + \mathrm{e}^{\mu^*})$, and $C(y) = 1$. For normal regression, by introducing an extra dispersion parameter σ^2, we have $A(\mu^*) = \mu^*/\sigma^2$, $B(\mu^*) = -\mu^{*2}/2\sigma^2$ and $C(y) = -y^2/2\sigma^2 - \log(2\pi\sigma^2)/2$. For simplicity, $\sigma^2 = 1$ is assumed to be known. How to extend our results to the case that σ^2 is unknown will be discussed in Remark 2.2.3.

We approximate $\mu^*(\boldsymbol{x})$ using a DNN. Consider a DNN with $H_n - 1$ hidden layers and L_h hidden units at layer h, where $L_{H_n} = 1$ for the output layer and $L_0 = p_n$ for the input layer. Let $\boldsymbol{w}^h \in \mathbb{R}^{L_h \times L_{h-1}}$ and $\boldsymbol{b}^h \in \mathbb{R}^{L_h \times 1}$, $h \in \{1, 2, ..., H_n\}$ denote the weights and bias of layer h, and let $\psi^h : R^{L_h \times 1} \to \mathbb{R}^{L_h \times 1}$ denote a coordinate-wise and piecewise differentiable activation function of layer h. The DNN forms a nonlinear mapping

$$\mu(\boldsymbol{\beta}, \boldsymbol{x}) = \boldsymbol{w}^{H_n} \psi^{H_n - 1} \left[\cdots \psi^1 \left[\boldsymbol{w}^1 \boldsymbol{x} + \boldsymbol{b}^1 \right] \cdots \right] + \boldsymbol{b}^{H_n}, \tag{2.1}$$

where $\boldsymbol{\beta} = (\boldsymbol{w}, \boldsymbol{b}) = \left\{ w^h_{\mathrm{ij}}, b^h_k : h \in \{1, 2, ..., H_n\}, \mathrm{i}, k \in \{1, ..., L_h\}, \mathrm{j} \in \{1, ..., L_{h-1}\} \right\}$ denotes the collection of all weights and biases, consisting of $K_n = \sum_{h=1}^{H_n} (L_{h-1} \times L_h + L_h)$ elements in total. To facilitate representation of the sparse DNN, we introduce an indicator variable for each weight and bias of the DNN, which indicates the existence of the connection in the network. Let $\boldsymbol{\gamma}^{\boldsymbol{w}^h}$ and $\boldsymbol{\gamma}^{\boldsymbol{b}^h}$ denote the matrix and vector of the indicator variables associated with $\boldsymbol{w}^h$ and $\boldsymbol{b}^h$, respectively. Further, we let $\boldsymbol{\gamma} = \{\boldsymbol{\gamma}_{\mathrm{ij}}^{\boldsymbol{w}^h}, \boldsymbol{\gamma}_k^{\boldsymbol{b}^h} : h \in \{1, 2, ..., H_n\},\ \mathrm{i}, k \in \{1, ..., L_h\}, \mathrm{j} \in \{1, ..., L_{h-1}\}\}$ and $\boldsymbol{\beta}_{\boldsymbol{\gamma}} = \{w_{\mathrm{ij}}^h, b_k^h : \boldsymbol{\gamma}_{\mathrm{ij}}^{\boldsymbol{w}^h} = 1, \boldsymbol{\gamma}_k^{\boldsymbol{b}^h} = 1, h \in \{1, 2, ..., H_n\}, \mathrm{i}, k \in \{1, ..., L_h\}, \mathrm{j} \in \{1, ..., L_{h-1}\}\}$, which specify, respectively, the structure and associated parameters for a sparse DNN.

To conduct Bayesian analysis for the sparse DNN, we consider a mixture Gaussian prior specified as follows:

$$\begin{aligned} \boldsymbol{\gamma}_{\mathrm{ij}}^{\boldsymbol{w}^h} &\sim Bernoulli(\lambda_n), \quad \boldsymbol{\gamma}_k^{\boldsymbol{b}^h} \sim Bernoulli(\lambda_n), \\ \boldsymbol{w}_{\mathrm{ij}}^h | \boldsymbol{\gamma}_{\mathrm{ij}}^{\boldsymbol{w}^h} &\sim \boldsymbol{\gamma}_{\mathrm{ij}}^{\boldsymbol{w}^h} N(0, \sigma_{1,n}^2) + (1 - \boldsymbol{\gamma}_{\mathrm{ij}}^{\boldsymbol{w}^h}) N(0, \sigma_{0,n}^2), \\ \boldsymbol{b}_k^h | \boldsymbol{\gamma}_k^{\boldsymbol{b}^h} &\sim \boldsymbol{\gamma}_k^{\boldsymbol{b}^h} N(0, \sigma_{1,n}^2) + (1 - \boldsymbol{\gamma}_k^{\boldsymbol{b}^h}) N(0, \sigma_{0,n}^2), \end{aligned} \tag{2.2}$$

where $h \in \{1, 2, ..., H_N\}, \mathrm{i} \in \{1, ..., L_{h-1}\}, \mathrm{j}, k \in \{1, ..., L_h\}$, and $\sigma_{0,n}^2 < \sigma_{1,n}^2$ are prespecified constants. Marginally, we have

$$\boldsymbol{w}_{\mathrm{ij}}^h \sim \lambda_n N(0, \sigma_{1,n}^2) + (1 - \lambda_n) N(0, \sigma_{0,n}^2), \quad \boldsymbol{b}_k^h \sim \lambda_n N(0, \sigma_{1,n}^2) + (1 - \lambda_n) N(0, \sigma_{0,n}^2). \tag{2.3}$$

Typically, we set $\sigma_{0,n}^2$ to be a very small value while $\sigma_{1,n}^2$ to be relatively large. When $\sigma_{0,n}^2 \to 0$, the prior is reduced to the spike-and-slab prior [36]. Therefore, this prior can be viewed as a continuous relaxation of the spike-and-slab prior. Such a prior has been used by many authors in Bayesian variable selection, see e.g., [37] and [38].

2.2 Posterior Consistency

Posterior consistency plays a major role in validating Bayesian methods especially for high-dimensional models, see e.g. [39] and [40]. For DNNs, since the total number of parameters K_n is often much larger than the sample size n, posterior consistency provides a general guideline in prior setting or choosing prior hyperparameters for a class of prior distributions. Otherwise, the prior information may dominate data information, rendering a biased inference for the underlying true model. In what follows, we prove the posterior consistency of the DNN model with the mixture Gaussian prior (2.3).

With slight abuse of notation, we rewrite $\mu(\boldsymbol{\beta}, \boldsymbol{x})$ in (2.1) as $\mu(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{x})$ for a sparse network by including its network structure information. We assume $\mu^*(\boldsymbol{x})$ can be well approximated by a *sparse DNN* with relevant variables, and call this sparse DNN as the *true DNN*. More precisely, we define the *true DNN* as

$$(\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*) = \underset{(\boldsymbol{\beta},\boldsymbol{\gamma})\in\mathcal{G}_n,\, \|\mu(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{x})-\mu^*(\boldsymbol{x})\|_{L^2(\Omega)}\leq\varpi_n}{\arg\min} |\boldsymbol{\gamma}|, \tag{2.4}$$

where $\mathcal{G}_n := \mathcal{G}(C_0, C_1, \varepsilon, p_n, H_n, L_1, L_2, \ldots, L_{H_n})$ denotes the space of valid sparse networks satisfying condition A.2 (given below) for the given values of H_n, p_n, and L_h's, and ϖ_n is some sequence converging to 0 as $n \to \infty$. For any given DNN $(\boldsymbol{\beta}, \boldsymbol{\gamma})$, the error $\mu(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{x}) - \mu^*(\boldsymbol{x})$ can be generally decomposed as the network approximation error $\mu(\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*, \boldsymbol{x}) - \mu^*(\boldsymbol{x})$ and the network estimation error $\mu(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{x}) - \mu(\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*, \boldsymbol{x})$. The L_2 norm of the former one is bounded by ϖ_n, and the order of the latter will be given in Theorem 2.2.1. In what follows, we will treat ϖ_n as the network approximation error. In addition, we make the following assumptions:

A.1 The input $\boldsymbol{x}$ is bounded by 1 entry-wisely, i.e. $\boldsymbol{x} \in \Omega = [-1, 1]^{p_n}$, and the density of $\boldsymbol{x}$ is bounded in its support Ω uniformly with respect to n.

A.2 The true sparse DNN model satisfies the following conditions:

- A.2.1 The network structure satisfies: $r_n H_n \log n + r_n \log \overline{L} + s_n \log p_n \leq C_0 n^{1-\varepsilon}$, where $0 < \varepsilon < 1$ is a small constant, $r_n = |\boldsymbol{\gamma}^*|$ denotes the connectivity of $\boldsymbol{\gamma}^*$, $\overline{L} = \max_{1\leq \mathrm{j}\leq H_n-1} L_\mathrm{j}$ denotes the maximum hidden layer width, s_n denotes the input dimension of $\boldsymbol{\gamma}^*$.

- A.2.2 The network weights are polynomially bounded: $\|\boldsymbol{\beta}^*\|_\infty \leq E_n$, where $E_n = n^{C_1}$ for some constant $C_1 > 0$.

A.3 The activation function ψ is Lipschitz continuous with a Lipschitz constant of 1.

Assumption A.1 is a typical assumption for posterior consistency, see e.g., [15] and [39]. In practice, all bounded data can be normalized to satisfy this assumption, e.g. image data are bounded and usually normalized before training. Assumption A.3 is satisfied by many conventional activation functions such as sigmoid, tanh and ReLU.

Assumption A.2 specifies the class of DNN models that we are considering. They are sparse, while still being able to approximate many types of functions arbitrarily well as the training sample size becomes large, i.e., $\lim_{n\to\infty} \varpi_n = 0$. The approximation power of sparse DNNs has been studied in several existing work. For example, for the functions that can be represented by an affine system, [11] proved that if the network parameters are bounded in absolute value by some polynomial $g(r_n)$, i.e. $||\boldsymbol{\beta}^*||_\infty \leq g(r_n)$, then the approximation error $\varpi_n = O(r_n^{-\alpha^*})$ for some constant α^*. To fit this this result into our framework, we can let $r_n \asymp n^{(1-\epsilon)/2}$ for some $0 < \epsilon < 1$, $p_n = d$ for some constant d, $H_n < r_n + d$ and $\bar{L} < r_n$ (i.e. the setting given in Proposition 3.6 of [11]). Suppose that the degree of $g(\cdot)$ is c_2, i.e. $g(r_n) \prec r_n^{c_2}$, then $\|\boldsymbol{\beta}^*\|_\infty \prec n^{c_2(1-\epsilon)/2} \prec n^{C_1} = E_n$ for some constant $C_1 > c_2(1-\epsilon)/2$. Therefore, Assumption A.2 is satisfied with the approximation error $\varpi_n = O(r_n^{-\alpha^*}) = O(n^{-\alpha^*(1-\epsilon)/2}) \triangleq O(n^{-\varsigma})$ (by defining $\varsigma = \alpha^*(1-\epsilon)/2$), which goes to 0 as $n \to \infty$. In summary, the minimax rate in $\sup_{\mu^*(\boldsymbol{x})\in\mathcal{C}} \inf_{(\beta,\gamma)\in\mathcal{G}} \|\mu(\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{x})-\mu^*(\boldsymbol{x})\|_{L^2(\Omega)} \in \mathcal{O}(n^{-\varsigma})$ can be achieved by sparse DNNs under our assumptions, where $\mathcal{C}$ denotes the class of functions represented by an affine system.

Other than affine functions, our setup for the sparse DNN also matches the approximation theory for many other types of functions. For example, Corollary 3.7 of [41] showed that for a wide class of piecewise smooth functions with a fixed input dimension, a fixed depth ReLU network can achieve an ϖ_n-approximation with $\log(r_n) = O(-\log\varpi_n)$ and $\log E_n = O(-\log\varpi_n)$. This result satisfies condition A.2 by setting $\varpi_n = O(n^{-\varsigma})$ for some constant $\varsigma > 0$. As another example, Theorem 3 of [12] (see also lemma 5.1 of [15]) proved that any bounded α-Hölder smooth function $\mu^*(\boldsymbol{x})$ can be approximated by a sparse ReLU DNN with the network approximation error $\varpi_n = O(\log(n)^{\alpha/p_n} n^{-\alpha/(2\alpha+p_n)})$ for some $H_n \asymp \log n \log p_n$, $L_j \asymp p_n n^{p_n/(2\alpha+p_n)}/\log n$, $r_n = O(p_n^2 \alpha^{2p_n} n^{p_n/(2\alpha+p_n)} \log p_n)$, and $E_n = C$ for some fixed constant $C > 0$. This result also satisfies condition A.2.2 as long as $p_n^2 \ll \log n$.

It is important to note that there is a fundamental difference between the existing neural network approximation theory and ours. In the existing neural network approximation theory, no data is involved and a small network can potentially achieve an arbitrarily small approximation error by allowing the connection weights to take values in an unbounded space. In contrast, in our theory, the network approximation error, the network size, and the bound of connection weights are all linked to the training sample size. A small network approximation error is required only when the training sample size is large; otherwise, over-fitting might be a concern from the point of view of statistical modeling. In the practice of modern neural networks, the depth and width have been increased without much scruple. These increases reduce the training error, improve the generalization performance under certain regimes [42], but negatively affect model calibration [3]. We expect that our theory can tame the powerful neural networks into the framework of statistical modeling; that is, by selecting an appropriate network size according to the training sample size, the proposed method can generally improve the generalization and calibration of the DNN model while controlling the training error to a reasonable level.

Let P^* and E^* denote the respective probability measure and expectation for data D_n. Let $d(p_1,p_2) = \left(\int \left[p_1^{\frac{1}{2}}(\boldsymbol{x},y) - p_2^{\frac{1}{2}}(\boldsymbol{x},y)\right]^2 dy d\boldsymbol{x}\right)^{\frac{1}{2}}$ denote the Hellinger distance between two density functions $p_1(\boldsymbol{x},y)$ and $p_2(\boldsymbol{x},y)$. Let $\boldsymbol{\pi}(A \mid D_n)$ be the posterior probability of an event A. The following theorem establishes posterior consistency for sparse DNNs under the mixture Gaussian prior (2.3).

Theorem 2.2.1. *Suppose Assumptions A.1-A.3 hold. If the mixture Gaussian prior (2.3) satisfies the conditions:* $\lambda_n = O(1/\{K_n[n^{H_n}(\overline{L}p_n)]^\tau\})$ *for some constant* $\tau > 0$, $E_n/\{H_n \log n + \log \overline{L}\}^{1/2} \lesssim \sigma_{1,n} \lesssim n^\alpha$ *for some constant* $\alpha > 0$, *and* $\sigma_{0,n} \lesssim \min\left\{1/\{\sqrt{n}K_n(n^{3/2}\sigma_{1,0}/H_n)^{H_n}\}, 1/\{\sqrt{n}K_n(nE_n/H_n)^{H_n}\}\right\}$, *then there exists an error sequence* $\epsilon_n^2 = O(\varpi_n^2) + O(\zeta_n^2)$ *such that* $\lim_{n\to\infty}\epsilon_n = 0$ *and* $\lim_{n\to\infty} n\epsilon_n^2 = \infty$, *and the posterior distribution satisfies*

$$\begin{aligned} &P^*\left\{\boldsymbol{\pi}[d(p_{\boldsymbol{\beta}}, p_{\mu^*}) > 4\epsilon_n | D_n] \geq 2e^{-cn\epsilon_n^2}\right\} \leq 2e^{-cn\epsilon_n^2}, \\ &E^*_{D_n}\boldsymbol{\pi}[d(p_{\boldsymbol{\beta}}, p_{\mu^*}) > 4\epsilon_n | D_n] \leq 4e^{-2cn\epsilon_n^2}, \end{aligned} \tag{2.5}$$

for sufficiently large n, *where* c *denotes a constant,* $\zeta_n^2 = [r_n H_n \log n + r_n \log \overline{L} + s_n \log p_n]/n$, p_{μ^*} *denotes the underlying true data distribution, and* $p_{\boldsymbol{\beta}}$ *denotes the data distribution reconstructed by the Bayesian DNN based on its posterior samples.*

The proof of Theorem 2.2.1 can be found in Section 2.9. Regarding this theorem, we have a few remarks:

Remark 2.2.1. *Theorem 2.2.1 provides a posterior contraction rate ϵ_n for the sparse BNN. The contraction rate contains two components, ϖ_n and ζ_n, where ϖ_n, as defined previously, represents the network approximation error, and ζ_n represents the network estimation error measured in Hellinger distance. Since the estimation error ζ_n grows with the network connectivity r_n, there is a trade-off between the network approximation error and the network estimation error. A larger network has a lower approximation error and a higher estimation error, and vice versa.*

Remark 2.2.2. *Theorem 2.2.1 implies that given a training sample size n, the proposed method can learn a sparse neural network with at most $O(n/\log(n))$ connections. Compared to the fully connected DNN, the sparsity of the proposed BNN enables some theoretical guarantees for its performance. The sparse BNN has nice theoretical properties, such as posterior consistency, variable selection consistency, and asymptotically optimal generalization bounds, which are beyond the ability of general neural networks. The latter two properties will be established in Section 2.3 and Section 2.5, respectively.*

Remark 2.2.3. *Although Theorem 2.2.1 is proved by assuming σ^2 is known, it can be easily extended to the case that σ^2 is unknown by assuming an inverse gamma prior $\sigma^2 \sim IG(a_0, b_0)$ for some constants $a_0, b_0 > 0$. If a relatively uninformative prior is desired, one can choose $a_0 \in (0,1)$ such that the inverse gamma prior is very diffuse with a non-existing mean value. However, if $a_0 = b_0 = 0$, i.e., the Jeffreys prior $\pi(\sigma^2) \propto 1/\sigma^2$, the posterior consistency theory established Theorem 2.2.1 might not hold any more. In general, to achieve posterior consistency, the prior is required, at least in our framework, to satisfy two conditions [39], [43]: (i) a not too little prior probability is placed over the neighborhood of the true density, and (ii) a very little prior probability is placed outside of a region that is not too complex. Obviously, the Jeffreys prior and thus the joint prior of σ^2 and the regression coefficients do not satisfy neither of the two conditions. We note that the inverse gamma prior $\sigma^2 \sim IG(a_0, b_0)$ has long been used in Bayesian inference for many different statistical models,* *such as linear regression [44], nonparametric regression [45], and Gaussian graphical models [46].*

2.3 Consistency of DNN Structure Selection

This section establishes consistency of DNN structure selection under posterior consistency. It is known that the DNN model is generally nonidentifiable due to the symmetry of the network structure. For example, the approximation $\mu(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{x})$ can be invariant if one permutes the orders of certain hidden nodes, simultaneously changes the signs of certain weights and biases if *tanh* is used as the activation function, or re-scales certain weights and bias if ReLU is used as the activation function. However, by introducing appropriate constraints, see e.g., [47] and [14], we can define a set of neural networks such that any possible neural networks can be represented by one and only one neural network in the set via nodes permutation, sign changes, weight rescaling, etc. Let Θ denote such set of DNNs, where each element in Θ can be viewed as an equivalent class of DNN models. Let $\nu(\boldsymbol{\gamma}, \boldsymbol{\beta}) \in \Theta$ be an operator that maps any neural network to Θ via appropriate transformations such as nodes permutation, sign changes, weight rescaling, etc. To serve the purpose of structure selection in the space Θ, we consider the marginal posterior inclusion probability approach proposed in [40] for high-dimensional variable selection.

For a better description of this approach, we reparameterize $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ as

$$\boldsymbol{\beta} = (\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_{K_n}), \quad \boldsymbol{\gamma} = (\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \ldots, \boldsymbol{\gamma}_{K_n}),$$

respectively, according to their elements. Without possible confusions, we will often use the indicator vector $\boldsymbol{\gamma}$ and the active set $\{\mathrm{i} : \boldsymbol{\gamma}_\mathrm{i} = 1, \mathrm{i} = 1, 2, \ldots, K_n\}$ exchangeably; that is, $\mathrm{i} \in \boldsymbol{\gamma}$ and $\boldsymbol{\gamma}_\mathrm{i} = 1$ are equivalent. In addition, we will treat the connection weights $\boldsymbol{w}$ and the hidden unit biases $\boldsymbol{b}$ equally; that is, they will not be distinguished in $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$. For convenience, we will call each element of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ a ‘connection’ in what follows.

2.3.1 Marginal Posterior Inclusion Probability Approach

For each connection c_{i}, we define its marginal posterior inclusion probability by

$$q_{\mathrm{i}} = \int \sum_{\boldsymbol{\gamma}} e_{\mathrm{i}|\nu(\boldsymbol{\gamma},\boldsymbol{\beta})} \pi(\boldsymbol{\gamma}|\boldsymbol{\beta}, D_n)\pi(\boldsymbol{\beta}|D_n)d\boldsymbol{\beta}, \quad \mathrm{i} = 1, 2, \ldots, K_n, \tag{2.6}$$

where $e_{\mathrm{i}|\nu(\boldsymbol{\gamma},\boldsymbol{\beta})}$ is the indicator for the existence of connection c_{i} in the network $\nu(\boldsymbol{\gamma}, \boldsymbol{\beta})$. Similarly, we define $e_{\mathrm{i}|\nu(\boldsymbol{\gamma}^*,\boldsymbol{\beta}^*)}$ as the indicator for the existence of connection c_{i} in the true model $\nu(\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*)$. The proposed approach is to choose the connections whose marginal posterior inclusion probabilities are greater than a threshold value $\hat{q}$; that is, setting $\hat{\boldsymbol{\gamma}}_{\hat{q}} = \{\mathrm{i} : q_{\mathrm{i}} > \hat{q}, \mathrm{i} = 1, 2, \ldots, K_n\}$ as an estimator of $\boldsymbol{\gamma}_* = \{\mathrm{i} : e_{\mathrm{i}|\nu(\boldsymbol{\gamma}^*,\boldsymbol{\beta}^*)} = 1, \mathrm{i} = 1, \ldots, K_n\}$, where $\boldsymbol{\gamma}_*$ can be viewed as the uniquenized true model. To establish the consistency of $\hat{\boldsymbol{\gamma}}_{\hat{q}}$, an identifiability condition for the true model is needed. Let $A(\epsilon_n) = \{\boldsymbol{\beta} : d(p_{\boldsymbol{\beta}}, p_{\mu^*}) \geq \epsilon_n\}$. Define

$$\rho(\epsilon_n) = \max_{1 \leq \mathrm{i} \leq K_n} \int_{A(\epsilon_n)^c} \sum_{\boldsymbol{\gamma}} |e_{\mathrm{i}|\nu(\boldsymbol{\gamma},\boldsymbol{\beta})} - e_{\mathrm{i}|\nu(\boldsymbol{\gamma}^*,\boldsymbol{\beta}^*)}| \pi(\boldsymbol{\gamma}|\boldsymbol{\beta}, D_n)\pi(\boldsymbol{\beta}|D_n)d\boldsymbol{\beta},$$

which measures the structure difference between the true model and the sampled models on the set $A(\epsilon_n)^c$. Then the identifiability condition can be stated as follows:

B.1 $\rho(\epsilon_n) \to 0$, as $n \to \infty$ and $\epsilon_n \to 0$.

That is, when n is sufficiently large, if a DNN has approximately the same probability distribution as the true DNN, then the structure of the DNN, after mapping into the parameter space Θ, must coincide with that of the true DNN. Note that this identifiability is different from the one mentioned at the beginning of the section. The earlier one is only with respect to structure and parameter rearrangement of the DNN. Theorem 2.3.1 concerns consistency of $\hat{\boldsymbol{\gamma}}_{\hat{q}}$ and its sure screening property, whose proof is given in Section 2.9.

Theorem 2.3.1. *Assume that the conditions of Theorem 2.2.1 and the identifiability condition B.1 hold. Then*

(i) $\max_{1 \leq \mathrm{i} \leq K_n}\{|q_{\mathrm{i}} - e_{\mathrm{i}|\nu(\boldsymbol{\gamma}^*,\boldsymbol{\beta}^*)}|\} \xrightarrow{p} 0$, *where* $\xrightarrow{p}$ *denotes convergence in probability;*

(ii) (sure screening) $P(\boldsymbol{\gamma}_* \subset \hat{\boldsymbol{\gamma}}_{\hat{q}}) \xrightarrow{p} 1$ *for any pre-specified* $\hat{q} \in (0, 1)$.

(iii) (Consistency) $P(\boldsymbol{\gamma}_* = \hat{\boldsymbol{\gamma}}_{0.5}) \xrightarrow{p} 1$.

For a network $\boldsymbol{\gamma}$, it is easy to identify the relevant variables. Recall that $\boldsymbol{\gamma}^{\boldsymbol{w}^h} \in \mathbb{R}^{L_h \times L_{h-1}}$ denotes the connection indicator matrix of layer h. Let

$$\boldsymbol{\gamma}^{\boldsymbol{x}} = \boldsymbol{\gamma}^{\boldsymbol{w}^{H_n}} \boldsymbol{\gamma}^{\boldsymbol{w}^{H_n - 1}} \cdots \boldsymbol{\gamma}^{\boldsymbol{w}^1} \in \mathbb{R}^{1 \times p_n}, \tag{2.7}$$

and let $\boldsymbol{\gamma}^{\boldsymbol{x}}_{\mathrm{i}}$ denote the i-th element of $\boldsymbol{\gamma}^{\boldsymbol{x}}$. It is easy to see that if $\boldsymbol{\gamma}^{\boldsymbol{x}}_{\mathrm{i}} > 0$ then the variable $\boldsymbol{x}_{\mathrm{i}}$ is effective in the network $\boldsymbol{\gamma}$, and $\boldsymbol{\gamma}^{\boldsymbol{x}}_{\mathrm{i}} = 0$ otherwise. Let $\mathrm{e}_{\boldsymbol{x}_{\mathrm{i}}|\nu(\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*)}$ be the indicator for the effectiveness of variable $\boldsymbol{x}_{\mathrm{i}}$ in the network $\nu(\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*)$, and let $\boldsymbol{\gamma}^{\boldsymbol{x}}_* = \{\mathrm{i} : \mathrm{e}_{\boldsymbol{x}_{\mathrm{i}}|\nu(\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*)} = 1, \mathrm{i} = 1, \ldots, p_n\}$ denote the set of true variables. Similar to (2.6), we can define the marginal inclusion probability for each variable:

$$q^{\boldsymbol{x}}_{\mathrm{i}} = \int \sum_{\boldsymbol{\gamma}} \mathrm{e}_{\boldsymbol{x}_{\mathrm{i}}|\nu(\boldsymbol{\gamma}, \boldsymbol{\beta})} \boldsymbol{\pi}(\boldsymbol{\gamma}|\boldsymbol{\beta}, D_n) \boldsymbol{\pi}(\boldsymbol{\beta}|D_n) d\boldsymbol{\beta}, \quad \mathrm{i} = 1, 2, \ldots, p_n, \tag{2.8}$$

Then we can select the variables whose marginal posterior inclusion probabilities greater than a threshold $\hat{q}^{\boldsymbol{x}}$, e.g., setting $\hat{q}^{\boldsymbol{x}} = 0.5$. As implied by (2.7), the consistency of structure selection implies consistency of variable selection.

It is worth noting that the above variable selection consistency result is with respect to the relevant variables defined by the true network $\boldsymbol{\gamma}^*$. To achieve the variable selection consistency with respect to the relevant variables of $\mu^*(\boldsymbol{x})$, some extra assumptions are needed in defining $(\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*)$. How to specify these assumptions is an open problem and we would leave it to readers. However, as shown by our simulation example, the sparse model $(\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*)$ defined in (2.4) works well, which correctly identifies all the relevant variables of the underlying nonlinear system.

2.3.2 Laplace Approximation of Marginal Posterior Inclusion Probabilities

Theorem 2.3.1 establishes the consistency of DNN structure selection based on the marginal posterior inclusion probabilities. To obtain Bayesian estimates of the marginal posterior inclusion probabilities, intensive Markov Chain Monte Carlo (MCMC) simulations are usually required. Instead of performing MCMC simulations, we propose to approximate the marginal posterior inclusion probabilities using the Laplace method based on the DNN model trained by an optimization method such as SGD. Traditionally, such approximation is required to be performed at the *maximum a posteriori* (MAP) estimate of the DNN. However, finding the MAP for a large DNN is not computationally guaranteed, as there can be many local minima on its energy landscape. To tackle this issue, we proposed a Bayesian evidence method for eliciting sparse DNN models learned by an optimization method in multiple runs with different initializations. Alternatively, we provide an prior annealing approach, which incorporate the property of the energy landscape of over-parameterized DNN into model training, to find the optimal of posterior. See Section 2.6 for the detail. Since conventional optimization methods such as SGD can be used to train the DNN here, the proposed method is generally more computationally efficient than the standard Bayesian method. More importantly, as explained in Section 2.6, consistent estimates of the marginal posterior inclusion probabilities might be obtained at a local maximizer of the log-posterior instead of the MAP estimate. In what follows, we justify the validity of Laplace approximation for marginal posterior inclusion probabilities.

Based on the marginal posterior distribution $\boldsymbol{\pi}(\boldsymbol{\beta}|D_n)$, the marginal posterior inclusion probability q_{i} of connection c_{i} can be re-expressed as

$$q_{\mathrm{i}} = \int \boldsymbol{\pi}(\boldsymbol{\gamma}_{\mathrm{i}} = 1|\boldsymbol{\beta})\boldsymbol{\pi}(\boldsymbol{\beta}|D_n)d\boldsymbol{\beta}, \quad \mathrm{i} = 1, 2, \dots, K_n.$$

Under the mixture Gaussian prior, it is easy to derive that

$$\boldsymbol{\pi}(\boldsymbol{\gamma}_{\mathrm{i}} = 1|\boldsymbol{\beta}) = \tilde{b}_{\mathrm{i}}/(\tilde{a}_{\mathrm{i}} + \tilde{b}_{\mathrm{i}}), \tag{2.9}$$

where

$$\tilde{a}_{\mathrm{i}} = \frac{1-\lambda_n}{\sigma_{0,n}} \exp\{-\frac{\boldsymbol{\beta}_{\mathrm{i}}^2}{2\sigma_{0,n}^2}\}, \quad \tilde{b}_{\mathrm{i}} = \frac{\lambda_n}{\sigma_{1,n}} \exp\{-\frac{\boldsymbol{\beta}_{\mathrm{i}}^2}{2\sigma_{1,n}^2}\}.$$

Let's define

$$h_n(\boldsymbol{\beta}) = \frac{1}{n}\sum_{\mathrm{i}=1}^{n} \log(p(y_{\mathrm{i}}, \boldsymbol{x}_{\mathrm{i}}|\boldsymbol{\beta})) + \frac{1}{n}\log(\boldsymbol{\pi}(\boldsymbol{\beta})), \tag{2.10}$$

where $p(y_{\rm i}, \boldsymbol{x}_{\rm i}|\boldsymbol{\beta})$ denotes the likelihood function of the observation $(y_{\rm i}, \boldsymbol{x}_{\rm i})$ and $\boldsymbol{\pi}(\boldsymbol{\beta})$ denotes the prior as specified in (2.3). Then $\boldsymbol{\pi}(\boldsymbol{\beta}|D_n) = \frac{e^{nh_n(\boldsymbol{\beta})}}{\int e^{nh_n(\boldsymbol{\beta})}d\boldsymbol{\beta}}$ and, for a function $b(\boldsymbol{\beta})$, the posterior expectation is given by $\frac{\int b(\boldsymbol{\beta})e^{nh_n(\boldsymbol{\beta})}d\boldsymbol{\beta}}{\int e^{nh_n(\boldsymbol{\beta})}d\boldsymbol{\beta}}$. Let $\hat{\boldsymbol{\beta}}$ denote a strict local maximum of $\boldsymbol{\pi}(\boldsymbol{\beta}|D_n)$. Then $\hat{\boldsymbol{\beta}}$ is also a local maximum of $h_n(\boldsymbol{\beta})$. Let $B_\delta(\boldsymbol{\beta})$ denote an Euclidean ball of radius δ centered at $\boldsymbol{\beta}$. Let $h_{{\rm i}_1,{\rm i}_2,\ldots,{\rm i}_d}(\boldsymbol{\beta})$ denote the d-th order partial derivative $\frac{\partial^d h(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}_{{\rm i}_1}\partial\boldsymbol{\beta}_{{\rm i}_2}\cdots\partial\boldsymbol{\beta}_{{\rm i}_d}}$, let $H_n(\boldsymbol{\beta})$ denote the Hessian matrix of $h_n(\boldsymbol{\beta})$, let $h_{\rm ij}$ denote the $({\rm i},{\rm j})$-th component of the Hessian matrix, and let $h^{\rm ij}$ denote the $({\rm i},{\rm j})$-component of the inverse of the Hessian matrix. Recall that $\boldsymbol{\gamma}^*$ denotes the set of indicators for the connections of the true sparse DNN, r_n denotes the size of the true sparse DNN, and K_n denotes the size of the fully connected DNN. The following theorem justifies the Laplace approximation of the posterior mean for a bounded function $b(\boldsymbol{\beta})$.

Theorem 2.3.2. *Assume that there exist positive numbers ϵ, M, η, and n_0 such that for any $n > n_0$, the function $h_n(\boldsymbol{\beta})$ in (2.10) satisfies the following conditions:*

C.1 $|h_{{\rm i}_1,\ldots,{\rm i}_d}(\hat{\boldsymbol{\beta}})| < M$ hold for any $\boldsymbol{\beta} \in B_\epsilon(\hat{\boldsymbol{\beta}})$ and any $1 \le {\rm i}_1, \ldots, {\rm i}_d \le K_n$, where $3 \le d \le 4$.

C.2 $|h^{\rm ij}(\hat{\boldsymbol{\beta}})| < M$ if $\boldsymbol{\gamma}^*_{\rm i} = \boldsymbol{\gamma}^*_{\rm j} = 1$ and $|h^{\rm ij}(\hat{\boldsymbol{\beta}})| = O(\frac{1}{K_n^2})$ otherwise.

C.3 $\det(-\frac{n}{2\pi}H_n(\hat{\boldsymbol{\beta}}))^{\frac{1}{2}} \int_{\mathbb{R}^{K_n}\setminus B_\delta(\hat{\boldsymbol{\beta}})} e^{n(h_n(\boldsymbol{\beta})-h_n(\hat{\boldsymbol{\beta}}))}d\boldsymbol{\beta} = O(\frac{r_n^4}{n}) = o(1)$ for any $0 < \delta < \epsilon$.

For any bounded function $b(\boldsymbol{\beta})$, if $|b_{{\rm i}_1,\ldots,{\rm i}_d}(\boldsymbol{\beta})| = |\frac{\partial^d b(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}_{{\rm i}_1}\partial\boldsymbol{\beta}_{{\rm i}_2}\cdots\partial\boldsymbol{\beta}_{{\rm i}_d}}| < M$ holds for any $1 \le d \le 2$ and any $1 \le {\rm i}_1, \ldots, {\rm i}_d \le K_n$, then for the posterior mean of $b(\boldsymbol{\beta})$, we have

$$\frac{\int b(\boldsymbol{\beta})e^{nh_n(\boldsymbol{\beta})}d\boldsymbol{\beta}}{\int e^{nh_n(\boldsymbol{\beta})}d\boldsymbol{\beta}} = b(\hat{\boldsymbol{\beta}}) + O\left(\frac{r_n^4}{n}\right).$$

Conditions C.1 and C.3 are typical conditions for Laplace approximation, see e.g., [48]. Condition C.2 requires the inverse Hessian to have very small values for the elements corresponding to the false connections. To justify condition C.2, we note that for a multivariate normal distribution, the inverse Hessian is its covariance matrix. Thus, we expect that for the weights with small variance, their corresponding elements in the inverse Hessian matrix would be small as well. The following lemma quantifies the variance of the weights for the false connections.

Lemma 2.3.1. *Assume that* $\sup_n \int |\boldsymbol{\beta}_{\mathrm{i}}|^{2+\delta}\pi(\beta_{\mathrm{i}}|D_n)d\boldsymbol{\beta}_{\mathrm{i}} \le C < \infty$ *a.s. for some constants* $\delta > 0$ *and* $C > 0$ *and* $\rho(\epsilon_n) \asymp \pi(d(p_{\boldsymbol{\beta}}, p_{\mu^*}) \ge \epsilon_n|D_n)$*, where* $\rho(\epsilon_n)$ *is defined in Condition B.1. Then with an appropriate choice of prior hyperparameters and* ϵ_n*,* $P^*\{E(\boldsymbol{\beta}_{\mathrm{i}}^2|D_n) \prec \frac{1}{K_n^{2H_n-1}}\} \ge 1 - 2\mathrm{e}^{-n\epsilon_n^2/4}$ *holds for any false connection* c_{i} *in* $\boldsymbol{\gamma}^*$ *(i.e.,* $\boldsymbol{\gamma}_{\mathrm{i}}^* = 0$*).*

In addition, with an appropriate choice of prior hyperparameters, we can also show that $\pi(\boldsymbol{\gamma}_{\mathrm{i}} = 1|\boldsymbol{\beta})$ satisfies all the requirements of $b(\boldsymbol{\beta})$ in Theorem 2.3.2 with a probability tending to 1 as $n \to \infty$. Then, by Theorem 2.3.2, q_k and $\pi(\boldsymbol{\gamma}_{\mathrm{i}} = 1|\hat{\boldsymbol{\beta}})$ are approximately the same as $n \to \infty$, where $\pi(\boldsymbol{\gamma}_{\mathrm{i}} = 1|\hat{\boldsymbol{\beta}})$ is as defined in (2.9) but with $\boldsymbol{\beta}$ replaced by $\hat{\boldsymbol{\beta}}$. Combining with Theorem 2.3.1, we have that $\pi(\boldsymbol{\gamma}_{\mathrm{i}} = 1|\hat{\boldsymbol{\beta}})$ is a consistent estimator of $\mathrm{e}_{\mathrm{i}|\nu(\boldsymbol{\gamma}^*,\boldsymbol{\beta}^*)}$.

2.4 Asymptotic Normality of Connection Weights

In this section, we establish the asymptotic normality of the network parameters and predictions. Let $nl_n(\boldsymbol{\beta}) = \sum_{\mathrm{i}=1}^n \log(p_{\boldsymbol{\beta}}(\boldsymbol{x}_{\mathrm{i}}, y_{\mathrm{i}}))$ denote the log-likelihood function, and let $\pi(\boldsymbol{\beta})$ denote the density of the mixture Gaussian prior (2.3). Let $h_{\mathrm{i}_1,\mathrm{i}_2,\ldots,\mathrm{i}_d}(\boldsymbol{\beta})$ denote the d-th order partial derivatives $\frac{\partial^d l_n(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}_{\mathrm{i}_1}\partial\boldsymbol{\beta}_{\mathrm{i}_2}\cdots\partial\boldsymbol{\beta}_{\mathrm{i}_d}}$. Let $H_n(\boldsymbol{\beta})$ denote the Hessian matrix of $l_n(\boldsymbol{\beta})$. Let $h_{\mathrm{ij}}(\boldsymbol{\beta})$ and $h^{\mathrm{ij}}(\boldsymbol{\beta})$ denote the (i,j)-th component of $H_n(\boldsymbol{\beta})$ and $H_n^{-1}(\boldsymbol{\beta})$, respectively. Let $\bar{\lambda}_n(\boldsymbol{\beta})$ and $\underline{\lambda}_n(\boldsymbol{\beta})$ denotes the maximum and minimum eigenvalue of the Hessian matrix $H_n(\boldsymbol{\beta})$, respectively. Let $B_{\lambda,n} = \bar{\lambda}_n^{1/2}(\boldsymbol{\beta}^*)/\underline{\lambda}_n(\boldsymbol{\beta}^*)$ and $b_{\lambda,n} = \sqrt{r_n/n}B_{\lambda,n}$, where r_n is the connectivity of $\boldsymbol{\gamma}^*$. For a DNN parameterized by $\boldsymbol{\beta}$, we define the weight truncation at the true model $\boldsymbol{\gamma}^*$: $(\boldsymbol{\beta}_{\boldsymbol{\gamma}^*})_{\mathrm{i}} = \boldsymbol{\beta}_{\mathrm{i}}$ for $\mathrm{i} \in \boldsymbol{\gamma}^*$ and $(\boldsymbol{\beta}_{\boldsymbol{\gamma}^*})_{\mathrm{i}} = 0$ otherwise. For the mixture Gaussian prior (2.3), let $B_{\delta_n}(\boldsymbol{\beta}^*) = \{\boldsymbol{\beta} : |\boldsymbol{\beta}_{\mathrm{i}} - \boldsymbol{\beta}_{\mathrm{i}}^*| < \delta_n, \forall \mathrm{i} \in \boldsymbol{\gamma}^*, |\boldsymbol{\beta}_{\mathrm{i}} - \boldsymbol{\beta}_{\mathrm{i}}^*| < 2\sigma_{0,n}\log(\frac{\sigma_{1,n}}{\lambda_n\sigma_{0,n}}), \forall \mathrm{i} \notin \boldsymbol{\gamma}^*\}$. We follow the definition of asymptotic normality in [49] and [50]:

Definition 2.4.1. *Denote by* $d_{\boldsymbol{\beta}}$ *the bounded Lipschitz metric for weak convergence and by* ϕ_n *the mapping* $\phi_n : \boldsymbol{\beta} \to \sqrt{n}(g(\boldsymbol{\beta}) - g_*)$*. We say that the posterior distribution of the functional* $g(\boldsymbol{\beta})$ *is asymptotically normal with the center* g_* *and variance* G *if* $d_{\boldsymbol{\beta}}(\pi[\cdot \mid D_n] \circ \phi_n^{-1}, N(0, G)) \to 0$ *in* P^**-probability as* $n \to \infty$*. We will write this more compactly as* $\pi[\cdot \mid D_n] \circ \phi_n^{-1} \rightsquigarrow N(0, G)$*.*

Theorem 2.4.1 establishes the asymptotic normality of $\tilde{\nu}(\boldsymbol{\beta})$, where $\tilde{\nu}(\boldsymbol{\beta})$ denotes a transformation of $\boldsymbol{\beta}$ which is invariant with respect to $\mu(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{x})$ while minimizing $\|\tilde{\nu}(\boldsymbol{\beta}) - \boldsymbol{\beta}^*\|_\infty$.

Theorem 2.4.1. *Assume the conditions of Lemma 2.3.1 hold with $\rho(\epsilon_n) = o(\frac{1}{K_n})$ and $C_1 > \frac{2}{3}$ in Condition A.2.2. For some δ_n s.t. $\frac{r_n}{\sqrt{n}} \lesssim \delta_n \lesssim \frac{1}{\sqrt[3]{n r_n}}$, let $A(\epsilon_n, \delta_n) = \{\boldsymbol{\beta} : \max_{\mathrm{i}\in\boldsymbol{\gamma}^*} |\boldsymbol{\beta}_{\mathrm{i}} - \boldsymbol{\beta}^*_{\mathrm{i}}| > \delta_n, d(p_{\boldsymbol{\beta}}, p_{\mu^*}) \le \epsilon_n\}$, where ϵ_n is the posterior contraction rate as defined in Lemma 2.2.1. Assume there exists some constants $C > 2$ and $M > 0$ such that*

D.1 $\boldsymbol{\beta}^* = (\boldsymbol{\beta}^*_1, \boldsymbol{\beta}^*_2, \ldots, \boldsymbol{\beta}^*_{K_n})$ is generic [51], [52], $\min_{\mathrm{i}\in\boldsymbol{\gamma}^*} |\boldsymbol{\beta}^*_{\mathrm{i}}| > C\delta_n$ and $\boldsymbol{\pi}(A(\epsilon_n, \delta_n) \mid D_n) \to 0$ as $n \to \infty$.

D.2 $|h_{\mathrm{i}}(\boldsymbol{\beta}^*)| < M$, $|h_{\mathrm{j},k}(\boldsymbol{\beta}^*)| < M$, $|h^{\mathrm{j},k}(\boldsymbol{\beta}^*)| < M$, $|h_{\mathrm{i},\mathrm{j},k}(\boldsymbol{\beta})| < M$, $|h_l(\boldsymbol{\beta})| < M$ hold for any $\mathrm{i}, \mathrm{j}, k \in \boldsymbol{\gamma}^*$, $l \notin \boldsymbol{\gamma}^*$ and $\boldsymbol{\beta} \in B_{2\delta_n}(\boldsymbol{\beta}^*)$.

D.3 $\sup\left\{|E_{\boldsymbol{\beta}}(a^T U)^3| : \|\boldsymbol{\beta}_{\gamma^*} - \boldsymbol{\beta}^*\| \le 1.2 b_{\lambda,n}, \|a\| = 1\right\} \le 0.1\sqrt{n/r_n}\underline{\lambda}_n^2(\boldsymbol{\beta}^*)/\bar{\lambda}_n^{1/2}(\boldsymbol{\beta}^*)$ and $B_{\lambda,n} = O(1)$, where $U = Z - E_{\boldsymbol{\beta}_{\gamma^*}}(Z)$, Z denotes a random variable drawn from a neural network model parameterized by $\boldsymbol{\beta}_{\gamma^*}$, and $E_{\boldsymbol{\beta}_{\gamma^*}}(Z)$ denotes the mean of Z.

Then $\boldsymbol{\pi}[\sqrt{n}(\tilde{\nu}(\boldsymbol{\beta}) - \boldsymbol{\beta}^*) \mid D_n] \rightsquigarrow N(0, \boldsymbol{V})$ in P^*-probability as $n \to \infty$, where $\boldsymbol{V} = (v_{\mathrm{ij}})$, and $v_{\mathrm{i},\mathrm{j}} = E(h^{\mathrm{i},\mathrm{j}}(\boldsymbol{\beta}^*))$ if $\mathrm{i}, \mathrm{j} \in \boldsymbol{\gamma}^*$ and 0 otherwise.

Condition D.1 is essentially an identifiability condition, i.e., when n is sufficiently large, the DNN weights cannot be too far away from the true weights if the DNN produces approximately the same distribution as the true data. Condition D.2 gives typical conditions on derivatives of the DNN. Condition D.3 ensures consistency of the MLE of $\boldsymbol{\beta}^*$ for the given structure $\boldsymbol{\gamma}^*$ [53].

2.4.1 Asymptotic Normality of Prediction

Theorem 2.4.2 establishes asymptotic normality of the prediction $\mu(\boldsymbol{\beta}, \boldsymbol{x}_0)$ for a test data point $\boldsymbol{x}_0$, which implies that a faithful prediction interval can be constructed for the learnt sparse neural network. Refer to Section 2.6.4 for how to construct the prediction interval based on the theorem. Let $\mu_{\mathrm{i}_1,\mathrm{i}_2,\ldots,\mathrm{i}_d}(\boldsymbol{\beta}, \boldsymbol{x}_0)$ denote the d-th order partial derivative $\frac{\partial^d \mu(\boldsymbol{\beta}, \boldsymbol{x}_0)}{\partial \boldsymbol{\beta}_{\mathrm{i}_1} \partial \boldsymbol{\beta}_{\mathrm{i}_2} \cdots \partial \boldsymbol{\beta}_{\mathrm{i}_d}}$.

Theorem 2.4.2. *Assume the conditions of Theorem 2.4.1 and the following condition hold: $|\mu_{\mathrm{i}}(\boldsymbol{\beta}^*, \boldsymbol{x}_0)| < M$, $|\mu_{\mathrm{i},\mathrm{j}}(\boldsymbol{\beta}, \boldsymbol{x}_0)| < M$, $|\mu_k(\boldsymbol{\beta}, \boldsymbol{x}_0)| < M$ hold for any $\mathrm{i}, \mathrm{j} \in \boldsymbol{\gamma}^*, k \notin \boldsymbol{\gamma}^*$ and*

$\boldsymbol{\beta} \in B_{2\delta_n}(\boldsymbol{\beta}^*)$, *where* M *is as defined in Theorem 2.4.1. Then* $\boldsymbol{\pi}[\sqrt{n}(\mu(\boldsymbol{\beta}, \boldsymbol{x}_0) - \mu(\boldsymbol{\beta}^*, \boldsymbol{x}_0)) \mid D_n] \rightsquigarrow N(0, \Sigma)$, *where* $\Sigma = \nabla_{\gamma^*}\mu(\boldsymbol{\beta}^*, \boldsymbol{x}_0)^T H^{-1} \nabla_{\gamma^*}\mu(\boldsymbol{\beta}^*, \boldsymbol{x}_0)$ *and* $H = E(-\nabla^2_{\gamma^*} l_n(\boldsymbol{\beta}^*))$ *is the Fisher information matrix.*

The asymptotic normality for general smooth functional has been established in [49]. For linear and quadratic functional of deep ReLU network with a spike-and-slab prior, the asymptotic normality has been established in [50]. The DNN prediction $\mu(\boldsymbol{\beta}, \boldsymbol{x}_0)$ can be viewed as a point evaluation functional over the neural network function space. However, in general, this functional is not smooth with respect to the locally asymptotic normal (LAN) norm. The results of [49] and [50] are not directly applicable for the asymptotic normality of $\mu(\boldsymbol{\beta}, \boldsymbol{x}_0)$.

2.5 Asymptotically Optimal Generalization Bound

This section shows the sparse BNN has asymptotically an optimal generalization bound. First, we introduce a PAC Bayesian bound due to [54], [55], where the acronym PAC stands for Probably Approximately Correct. It states that with an arbitrarily high probability, the performance (as provided by a loss function) of a learning algorithm is upper-bounded by a term decaying to an optimal value as more data is collected (hence “approximately correct”). PAC-Bayes has proven over the past two decades to be a powerful tool to derive theoretical guarantees for many machine learning algorithms.

Lemma 2.5.1 (PAC Bayesian bound)**.** *Let* P *be any data independent distribution on the machine parameters* $\boldsymbol{\beta}$, *and* Q *be any distribution that is potentially data-dependent and absolutely continuous with respective to* P. *If the loss function* $l(\boldsymbol{\beta}, \boldsymbol{x}, y) \in [0, 1]$, *then the following inequality holds with probability* $1 - \delta$,

$$\int E_{\boldsymbol{x},y} l(\boldsymbol{\beta}, \boldsymbol{x}, y) dQ \leq \int \frac{1}{n} \sum_{\mathrm{i}=1}^{n} l(\boldsymbol{\beta}, \boldsymbol{x}^{(\mathrm{i})}, y^{(\mathrm{i})}) dQ + \sqrt{\frac{d_0(Q, P) + \log \frac{2\sqrt{n}}{\delta}}{2n}},$$

where $d_0(Q, P)$ *denotes the Kullback-Leibler divergence between* Q *and* P, *and* $(\boldsymbol{x}^{(\mathrm{i})}, y^{(\mathrm{i})})$ *denotes the* i*-th observation of the dataset.*

For the binary classification problem, the DNN model fits a predictive distribution as $\hat{p}_1(\boldsymbol{x};\boldsymbol{\beta}) := \widehat{Pr}(y=1|\boldsymbol{x}) = \text{logit}^{-1}(\mu(\boldsymbol{\beta},\boldsymbol{x}))$ and $\hat{p}_0(\boldsymbol{x};\boldsymbol{\beta}) := \widehat{Pr}(y=0|\boldsymbol{x}) = 1 - \text{logit}^{-1}(\mu(\boldsymbol{\beta},\boldsymbol{x}))$. Given an observation $(\boldsymbol{x}, y)$, we define the loss with margin $\nu > 0$ as

$$l_\nu(\boldsymbol{\beta},\boldsymbol{x},y) = 1(\hat{p}_y(\boldsymbol{x};\boldsymbol{\beta}) - \hat{p}_{1-y}(\boldsymbol{x};\boldsymbol{\beta}) < \nu).$$

Therefore, the empirical loss for the whole data set $\{\boldsymbol{x}^{(\mathrm{i})}, y^{(\mathrm{i})}\}_{\mathrm{i}=1}^n$ is defined as $L_{emp,\nu}(\boldsymbol{\beta}) = \sum l_\nu(\boldsymbol{\beta},\boldsymbol{x}^{(\mathrm{i})},y^{(\mathrm{i})})/n$, and the population loss is defined as $L_\nu(\boldsymbol{\beta}) = E_{\boldsymbol{x},\boldsymbol{y}} l_\nu(\boldsymbol{\beta},\boldsymbol{x},y)$.

Theorem 2.5.1 (Bayesian Generalization error for classification)**.** *Suppose the conditions of Theorem 2.2.1 hold. For any $\nu > 0$, when n is sufficiently large, the following inequality holds with probability greater than $1 - \exp\{c_0 n\epsilon_n^2\}$,*

$$\begin{aligned}&\int L_0(\boldsymbol{\beta})d\boldsymbol{\pi}(\boldsymbol{\beta}|D_n)\\ \le&\frac{1}{1-2\exp\{-c_1 n\epsilon_n^2\}}\int L_{emp,\nu}(\boldsymbol{\beta})d\boldsymbol{\pi}(\boldsymbol{\beta}|D_n) + O(\epsilon_n + \sqrt{\log n/n} + \exp\{-c_1 n\epsilon_n^2\}),\end{aligned}$$

for some c_0, $c_1 > 0$, where ϵ_n is as defined in Theorem 2.2.1.

Theorem 2.5.1 characterizes the relationship between Bayesian population risk $\int L_0(\boldsymbol{\beta})d\boldsymbol{\pi}(\boldsymbol{\beta}|D_n)$ and Bayesian empirical risk $\int L_{emp,\nu}(\boldsymbol{\beta})d\boldsymbol{\pi}(\boldsymbol{\beta}|D_n)$, and implies that the difference between them is $O(\epsilon_n)$. Furthermore, this generalization performance extends to any point estimator $\hat{\boldsymbol{\beta}}$, as long as $\hat{\boldsymbol{\beta}}$ belongs to the dominating posterior mode.

Theorem 2.5.2. *Suppose that the conditions of Theorem 2.2.1 hold and estimation $\hat{\boldsymbol{\beta}}$ belongs to the dominating posterior mode under Theorem 2.2.1, then for any $\nu > 0$, the following inequality holds with probability greater than $1 - \exp\{c_0 n\epsilon_n^2\}$,*

$$L_0(\hat{\boldsymbol{\beta}}) \le L_{emp,\nu}(\hat{\boldsymbol{\beta}}) + O(\epsilon_n),$$

for some $c_0 > 0$.

It is worth to clarify that the statement "$\hat{\boldsymbol{\beta}}$ belongs to the dominating posterior mode" means $\hat{\boldsymbol{\beta}} \in B_n$ where B_n is defined in the proof of Theorem 2.2.1 and its posterior is greater than $1-\exp\{-cn\epsilon_n^2\}$ for some $c>0$. Therefore, if $\hat{\boldsymbol{\beta}} \sim \boldsymbol{\pi}(\boldsymbol{\beta}|D_n)$, i.e., $\hat{\boldsymbol{\beta}}$ is one valid posterior sample, then with high probability, it belongs to the dominating posterior mode. The proof of the above two theorems can be found in Section 2.9

Now we consider the generalization error for regression models. Assume the following additional assumptions:

E.1 The activation function $\psi \in [-1,1]$.

E.2 The last layer weights and bias in $\boldsymbol{\beta}^*$ are restricted to the interval $[-F_n, F_n]$ for some $F_n \le E_n$, while $F_n \to \infty$ is still allowed as $n \to \infty$.

E.3 $\max_{\boldsymbol{x}\in\Omega} |\mu^*(\boldsymbol{x})| \le F$ for some constant F.

Correspondingly, the priors of the last layer weights and bias are truncated on $[-F_n, F_n]$, i.e., the two normal mixture prior (2.3) truncated on $[-F_n, F_n]$. By the same argument of Theorem 2.9.1 (in Section 2.9), Theorem 2.2.1 still holds.

Note that the Hellinger distance for regression problem is defined as

$$d^2(p_\beta, p_{\mu^*}) = \mathbb{E}_{\boldsymbol{x}}\left(1-\exp\left\{-\frac{[\mu(\boldsymbol{\beta},\boldsymbol{x})-\mu^*(\boldsymbol{x})]^2}{8\sigma^2}\right\}\right).$$

By our assumption, for any $\boldsymbol{\beta}$ on the prior support, $|\mu(\boldsymbol{\beta},\boldsymbol{x})-\mu^*(\boldsymbol{x})|^2 \le (F+\overline{L}F_n)^2 := \overline{F}^2$, thus,

$$d^2(p_\beta, p_{\mu^*}) \ge C_{\overline{F}} E_{\boldsymbol{x}} |\mu(\boldsymbol{\beta},\boldsymbol{x})-\mu^*(\boldsymbol{x})|^2, \tag{2.11}$$

where $C_F = [1-\exp(-4\overline{F}^2/8\sigma^2)]/4\overline{F}^2$. Furthermore, (2.5) implies that with probability at least $1-2\exp\{-cn\epsilon_n^2\}$,

$$\int d^2(p_\beta, p_{\mu^*}) d\boldsymbol{\pi}(\boldsymbol{\beta}|D_n) \le 16\epsilon_n^2 + 2\mathrm{e}^{-cn\epsilon_n^2}. \tag{2.12}$$

By Combining (2.11) and (2.12), we obtain the following Bayesian generalization error result:

Theorem 2.5.3. *(Bayesian generalization error for regression) Suppose the conditions of Theorem 2.2.1 hold. When n is sufficiently large, the following inequality holds with probability at least $1-2\exp\{-cn\epsilon_n^2\}$,*

$$\int E_{\boldsymbol{x}}|\mu(\boldsymbol{\beta},\boldsymbol{x})-\mu^*(\boldsymbol{x})|^2 d\boldsymbol{\pi}(\boldsymbol{\beta}|D_n) \leq [16\epsilon_n^2+2\mathrm{e}^{-cn\epsilon_n^2}]/C_F \asymp [\epsilon_n^2+\mathrm{e}^{-cn\epsilon_n^2}]\overline{L}^2 F_n^2. \tag{2.13}$$

Similarly, if an estimator $\hat{\boldsymbol{\beta}}$ belongs to the dominating posterior mode (refer to the discussion of Theorem 2.5.2 for more details), then $\hat{\boldsymbol{\beta}} \in \{\boldsymbol{\beta} : d(p_{\boldsymbol{\beta}}, p_{\mu^*}) \leq 4\epsilon_n\}$ and the following result hold:

Theorem 2.5.4. *Suppose the conditions of Theorem 2.2.1 hold, then*

$$E_{\boldsymbol{x}}|\mu(\hat{\boldsymbol{\beta}},\boldsymbol{x})-\mu^*(\boldsymbol{x})|^2 \leq [16\epsilon_n^2]/C_F \asymp \epsilon_n^2\overline{L}^2 F_n^2. \tag{2.14}$$

2.6 Computation

2.6.1 Bayesian Evidence Approach

The theoretical results established in previous sections show that the Bayesian sparse DNN can be learned with a mixture Gaussian prior and, more importantly, the posterior inference is not necessarily directly drawn based on posterior samples, which avoids the convergence issue of the MCMC implementation for large complex models. As shown in Theorems 2.3.2, 2.5.2 and 2.5.4, for the sparse BNN, a good local maximizer of the log-posterior distribution also guarantees consistency of the network structure selection and asymptotic optimality of the network generalization performance. This local maximizer, in the spirit of condition C.3 and the conditions of Theorems 2.5.2 and 2.5.4, is not necessarily a MAP estimate, as the factor $\det(-\frac{n}{2\pi}H_n(\hat{\boldsymbol{\beta}}))^{\frac{1}{2}}$ can play an important role. In other words, an estimate of $\boldsymbol{\beta}$ lies in a wide valley of the energy landscape is generally preferred. This is consistent with the view of many other authors, see e.g., [56] and [57], where different techniques have been developed to enhance convergence of SGD to a wide valley of the energy landscape.

Algorithm 1 Sparse DNN Elicitation with Bayesian Evidence

```
Input: $T$—the number of independent tries in training the DNN, and the prior hyperparameters $\sigma_{0,n}$, $\sigma_{1,n}$, and $\lambda_n$.
for $t = 1, 2, ..., T$ do
    (i) Initialization: Randomly initialize the weights and biases, set $\boldsymbol{\gamma}_{\mathrm{i}}$=1 for i $= 1, 2, \ldots, K_n$.
    (ii) Optimization: Run SGD to maximize $h_n(\boldsymbol{\beta})$ as defined in (2.10). Denote the estimate of $\boldsymbol{\beta}$ by $\hat{\boldsymbol{\beta}}$.
    (iii) Connection sparsification: For each i $\in \{1, 2, \ldots, K_n\}$, set $\boldsymbol{\gamma}_{\mathrm{i}} = 1$ if $|\hat{\boldsymbol{\beta}}_{\mathrm{i}}| > \frac{\sqrt{2}\sigma_{0,n}\sigma_{1,n}}{\sqrt{\sigma_{1,n}^2 - \sigma_{0,n}^2}}\sqrt{\log\left(\frac{1-\lambda_n}{\lambda_n}\frac{\sigma_{1,n}}{\sigma_{0,n}}\right)}$ and 0 otherwise. Denote the yielded sparse DNN structure by $\boldsymbol{\gamma}^t$, and set $\hat{\boldsymbol{\beta}}_{\boldsymbol{\gamma}^t} = \hat{\boldsymbol{\beta}} \circ \boldsymbol{\gamma}^t$, where $\circ$ denotes element-wise production.
    (iv) Nonzero-weights refining: Refine the nonzero weights of the sparsified DNN by maximizing
        $$h_n(\boldsymbol{\beta}_{\boldsymbol{\gamma}^t}) = \frac{1}{n}\sum_{\mathrm{i}=1}^{n}\log(p(y_{\mathrm{i}}, \boldsymbol{x}_{\mathrm{i}}|\boldsymbol{\beta}_{\boldsymbol{\gamma}^t})) + \frac{1}{n}\log(\boldsymbol{\pi}(\boldsymbol{\beta}_{\boldsymbol{\gamma}^t})), \qquad (2.15)$$
    which can be accomplished by running SGD for a few epochs with the initial value $\hat{\boldsymbol{\beta}}_{\boldsymbol{\gamma}^t}$. Denote the resulting DNN model by $\tilde{\boldsymbol{\beta}}_{\boldsymbol{\gamma}^t}$.
    (v) Model evaluation: Calculate the Bayesian Evidence: $Evidence^t = \det(-\frac{n}{2\pi}H_n(\tilde{\boldsymbol{\beta}}_{\boldsymbol{\gamma}^t}))^{-\frac{1}{2}}\mathrm{e}^{nh_n(\tilde{\boldsymbol{\beta}}_{\boldsymbol{\gamma}^t})}$, where $H_n(\boldsymbol{\beta}_{\boldsymbol{\gamma}}) = \frac{\partial^2 h_n(\boldsymbol{\beta}_{\boldsymbol{\gamma}})}{\partial\boldsymbol{\beta}_{\boldsymbol{\gamma}}\partial^T\boldsymbol{\beta}_{\boldsymbol{\gamma}}}$ is the Hessian matrix.
end for
Output $\tilde{\boldsymbol{\beta}}_{\boldsymbol{\gamma}^t}$ with the largest Bayesian evidence.
```

Condition C.3 can be re-expressed as $\int_{\mathbb{R}^{K_n}\setminus B_\delta(\hat{\boldsymbol{\beta}})} \mathrm{e}^{nh_n(\boldsymbol{\beta}))}d\boldsymbol{\beta} = o(\det(-\frac{n}{2\pi}H_n(\hat{\boldsymbol{\beta}}))^{-\frac{1}{2}}\mathrm{e}^{nh_n(\hat{\boldsymbol{\beta}})})$, which requires that $\hat{\boldsymbol{\beta}}$ is a dominating mode of the posterior. Based on this observation, we suggest to use the Bayesian evidence [58], [59] as the criterion for eliciting estimates of $\boldsymbol{\beta}$ produced by an optimization method in multiple runs with different initializations. The Bayesian evidence is calculated as $\det(-\frac{n}{2\pi}H_n(\hat{\boldsymbol{\beta}}))^{-\frac{1}{2}}\mathrm{e}^{nh_n(\hat{\boldsymbol{\beta}})}$. Since Theorem 2.3.1 ensures only consistency of structure selection but not consistency of parameter estimation, we suggest to refine its nonzero weights by a short optimization process after structure selection. The complete algorithm is summarized in Algorithm 1.

For a large-scale neural network, even if it is sparse, the number of nonzero elements can easily exceed a few thousands or millions, see e.g. the networks considered in Section 2.7.2. In this case, evaluation of the determinant of the Hessian matrix can be very time consuming. For this reason, we suggest to approximate the log(Bayesian evidence) by

$nh_n(\hat{\boldsymbol{\beta}}_{\boldsymbol{\gamma}}) - \frac{1}{2}|\boldsymbol{\gamma}|\log(n)$ with the detailed arguments given in Section 2.9 As explained there, if the prior information imposed on the sparse DNNs is further ignored, then the sparse DNNs can be elicited by BIC.

The main parameters for Algorithm 1 are the prior hyperparameters $\sigma_{0,n}$, $\sigma_{1,n}$, and λ_n. Theorem 2.2.1 provides theoretical suggestions for the choice of the prior-hyperparameters, see also the proof of Lemma 2.3.1 for a specific setting for them. Our theory allows $\sigma_{1,n}$ to grow with n from the perspective of data fitting, but in our experience, the magnitude of weights tend to adversely affect the generalization ability of the network. For this reason, we usually set $\sigma_{1,n}$ to a relatively small number such as 0.01 or 0.02, and then tune the values of $\sigma_{0,n}$ and λ_n for the network sparsity as well as the network approximation error. As a trade-off, the resulting network might be a little denser than the ideal one. If it is too dense to satisfy the sparse constraint given in Assumption A.2.2, one might increase the value of $\sigma_{0,n}$ and/or decrease the value of λ_n, and rerun the algorithm to get a sparser structure. This process can be repeated until the constraint is satisfied.

Algorithm 1 employs SGD to optimize the log-posterior of the BNN. Since SGD generally converges to a local optimal solution, the multiple initialization method is used in order to find a local optimum close to the global one. It is interesting to note that SGD has some nice properties in non-convex optimization: It works on the convolved (thus smoothed) version of the loss function [60] and tends to converge to flat local minimizers which are with very high probability also global minimizers [61]. In all of our experiments, we set the number of initializations to $T = 10$ as default unless otherwise stated. We note that Algorithm 1 is not very sensitive to the value of T, although a large value of T can generally improve its performance.

For network weight initialization, we adopted the standard method, see [62] for tanh activation and [63] for ReLU activation, which ensures that the variance of the gradient of each layer is of the same order at the beginning of the training process.

2.6.2 Prior Annealing: Frequentist Computation

In order to avoid the multiple run for the algorithm, we suggest to use a prior annealing approach which incorporate the study of optimization landscape of the over-parametrized DNN. It has been shown in [5], [6] that the loss of an over-parameterized DNN exhibits good properties:

(S^*) For a fully connected DNN with an analytic activation function and a convex loss function at the output layer, if the number of hidden units of one layer is larger than the number of training points and the network structure from this layer on is pyramidal, then almost all local minima are globally optimal.

Motivated by this result, we propose the following approach

Algorithm 2 Prior annealing: Frequentist

(i) (*Initial training*) Train a DNN satisfying condition (S*) such that a global optimal solution $\boldsymbol{\beta}_0 = \arg\max_{\boldsymbol{\beta}} l_n(\boldsymbol{\beta})$ is reached, which can be accomplished using SGD or Adam [64].

(ii) (*Prior annealing*) Initialize $\boldsymbol{\beta}$ at $\boldsymbol{\beta}_0$ and simulate from a sequence of distributions $\pi(\boldsymbol{\beta}|D_n, \tau, \eta^{(k)}, \sigma_{0,n}^{(k)}) \propto e^{nl_n(\boldsymbol{\beta})/\tau}\pi_k^{\eta^{(k)}/\tau}(\boldsymbol{\beta})$ for $k = 1, 2, \ldots, m$, where $0 < \eta^{(1)} \le \eta^{(2)} \le \cdots \le \eta^{(m)} = 1$, $\pi_k = \lambda_n N(0, \sigma_{1,n}^2) + (1-\lambda_n)N(0, (\sigma_{0,n}^{(k)})^2)$, and $\sigma_{0,n}^{init} = \sigma_{0,n}^{(1)} \ge \sigma_{0,n}^{(2)} \ge \cdots \ge \sigma_{0,n}^{(m)} = \sigma_{0,n}^{end}$. The simulation can be done in an annealing manner using a stochastic gradient MCMC algorithm [65]–[68]. After the stage m has been reached, continue to run the simulated annealing algorithm by gradually decreasing the temperature τ to a very small value. Denote the resulting DNN by $\hat{\boldsymbol{\beta}} = (\hat{\boldsymbol{\beta}}_1, \hat{\boldsymbol{\beta}}_2, \ldots, \hat{\boldsymbol{\beta}}_{K_n})$.

(iii) (*Structure sparsification*) For each connection $\mathrm{i} \in \{1, 2, \ldots, K_n\}$, set $\tilde{\boldsymbol{\gamma}}_{\mathrm{i}} = 1$ if $|\hat{\boldsymbol{\beta}}_{\mathrm{i}}| > \frac{\sqrt{2}\sigma_{0,n}\sigma_{1,n}}{\sqrt{\sigma_{1,n}^2 - \sigma_{0,n}^2}} \sqrt{\log\left(\frac{1-\lambda_n}{\lambda_n}\frac{\sigma_{1,n}}{\sigma_{0,n}}\right)}$ and 0 otherwise, where the threshold value of $|\hat{\boldsymbol{\beta}}_{\mathrm{i}}|$ is obtained by solving $\pi(\boldsymbol{\gamma}_{\mathrm{i}} = 1|\boldsymbol{\beta}_{\mathrm{i}}) > 0.5$ based on the mixture Gaussian prior as in [69]. Denote the yielded sparse DNN structure by $\tilde{\boldsymbol{\gamma}}$.

(iv) (*Nonzero-weights refining*) Refine the nonzero weights of the sparsified DNN by maximizing $l_n(\boldsymbol{\beta})$. Denote the resulting estimate by $\tilde{\boldsymbol{\beta}}_{\tilde{\boldsymbol{\gamma}}}$, which represents the MLE of $\boldsymbol{\beta}^*$.

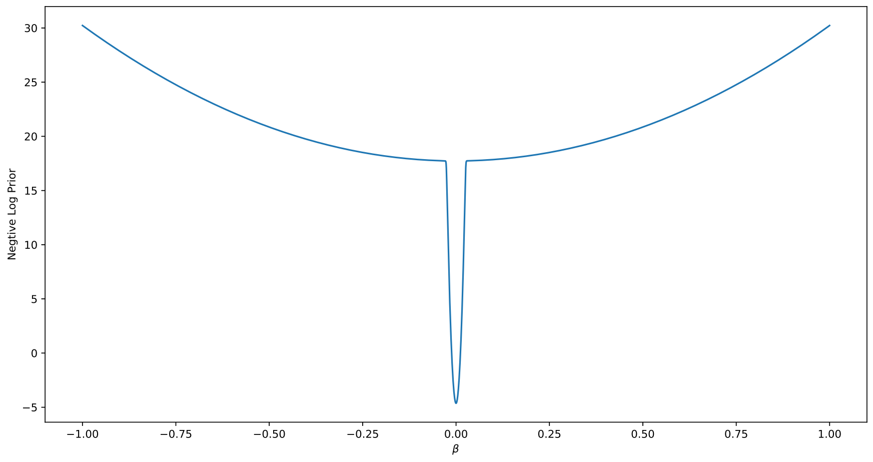


Figure 2.1. Negative logarithm of the mixture Gaussian prior.

For Algorithm 2, the consistency of $(\tilde{\boldsymbol{\gamma}}, \tilde{\boldsymbol{\beta}}_{\tilde{\boldsymbol{\gamma}}})$ as an estimator of $(\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*)$ can be proved based on Theorem 3.4 of [6] for global convergence of $\boldsymbol{\beta}_0$, the property of simulated annealing (by choosing an appropriate sequence of η_k and a cooling schedule of τ), Theorem 2.3.2 for consistency of structure selection, and Theorem 2.1 of [53] for consistency of MLE under the scenario of dimension diverging.

Intuitively, the initial training phase can reach the global optimum of the likelihood function. In the prior annealing phase, as we slowly add the effect of the prior, the landscape of the target distribution is gradually changed and the MCMC algorithm is likely to hit the region around the optimum of the target distribution. In practice, let t denote the step index, a simple implementation of the initial training and prior annealing phases of Algorithm 2 can be given as follows: (i) for $0 < t < T_1$, run initial training; (ii) for $T_1 \le t \le T_2$, fix $\sigma_{0,n}^{(t)} = \sigma_{0,n}^{init}$ and linearly increase η_t by setting $\eta^{(t)} = \frac{t-T_1}{T_2-T_1}$; (iii) for $T_2 \le t \le T_3$, fix $\eta^{(t)} = 1$ and linearly decrease $\left(\sigma_{0,n}^{(t)}\right)^2$ by setting $\left(\sigma_{0,n}^{(t)}\right)^2 = \frac{T_3-t}{T_3-T_2}\left(\sigma_{0,n}^{\text{init}}\right)^2 + \frac{t-T_2}{T_3-T_2}\left(\sigma_{0,n}^{\text{end}}\right)^2$; (iv) for $t > T_3$, fix $\eta^{(t)} = 1$ and $\sigma_{0,n}^{(t)} = \sigma_{0,n}^{end}$ and gradually decrease the temperature τ, e.g., setting $\tau_t = \frac{c}{t-T_3}$ for some constant c.

To better understand the prior annealing procedure, we provide some graphical illustrations. In practice, the negative log-prior puts penalty on parameter weights. The mixture Gaussian prior behaves like a piecewise L_2 penalty with different weights on different regions. Figure 2.1 shows the shape of a negative log-mixture Gaussian prior. In step (iii) of Algorithm 2, the condition $\pi(\boldsymbol{\gamma}_{\rm i} = 1|\boldsymbol{\beta}_{\rm i}) > 0.5$ splits the parameters into two parts. For the $\boldsymbol{\beta}_{\rm i}$'s with large magnitudes, the slab component $N(0, \sigma_{1,n}^2)$ plays the major role in the prior, imposing a small penalty on the parameter. For the $\boldsymbol{\beta}_{\rm i}$'s with smaller magnitudes, the spike component $N(0, \sigma_{0,n}^2)$ plays the major role in the prior, imposing a large penalty on the parameters to push them toward zero in training.

Figure 2.2 shows the shape of negative log-prior and $\pi(\boldsymbol{\gamma}_{\rm i} = 1|\boldsymbol{\beta}_{\rm i})$ for different choices of $\sigma_{0,n}^2$ and λ_n. As we can see from the plot, $\sigma_{0,n}^2$ plays the major role in determining the effect of the prior. Let α be the threshold in step (iii) of Algorithm 2, i.e. the positive solution to $\pi(\boldsymbol{\gamma}_{\rm i} = 1|\boldsymbol{\beta}_{\rm i}) = 0.5$. In general, a smaller $\sigma_{0,n}^2$ will result in a smaller α. If a very small $\sigma_{0,n}^2$ is used in the prior from the beginning, then most of $\boldsymbol{\beta}_{\rm i}$'s at initialization will have a magnitude larger than α and the slab component $N(0, \sigma_{1,n}^2)$ of the prior will dominate most parameters. As a result, it will be difficult to find the desired sparse structure. Following the proposed prior annealing procedure, we can start with a larger $\sigma_{0,n}^2$, i.e. a larger threshold α and a relatively smaller penalty for those $|\boldsymbol{\beta}_{\rm i}| < \alpha$. As we gradually decrease the value of $\sigma_{0,n}^2$, α decreases, and the penalty imposed on the small weights increases and drives them toward zero. The prior annealing allows us to gradually sparsify the DNN and impose more and more penalties on the parameters close to 0.

2.6.3 Prior Annealing: Bayesian Computation

For certain problems the size (or #nonzero elements) of $\boldsymbol{\gamma}^*$ is large, calculation of the Fisher information matrix is difficult. In this case, the prediction uncertainty can be quantified via posterior simulations. The simulation can be started with a DNN satisfying condition (S*) and performed using a SGMCMC algorithm [67], [68] with an annealed prior as defined in step (ii) of Algorithm 2 (For Bayesian approach, we may fix the temperature $\tau = 1$). The over-parameterized structure and annealed prior make the simulations immune to local traps.

To justify the Bayesian estimator for the prediction mean and variance, for some test function $\phi(\boldsymbol{\beta})$, we study the deviation of the path averaging estimator $\frac{1}{T}\sum_{t=1}^{T}\phi(\boldsymbol{\beta}^{(t)})$ and

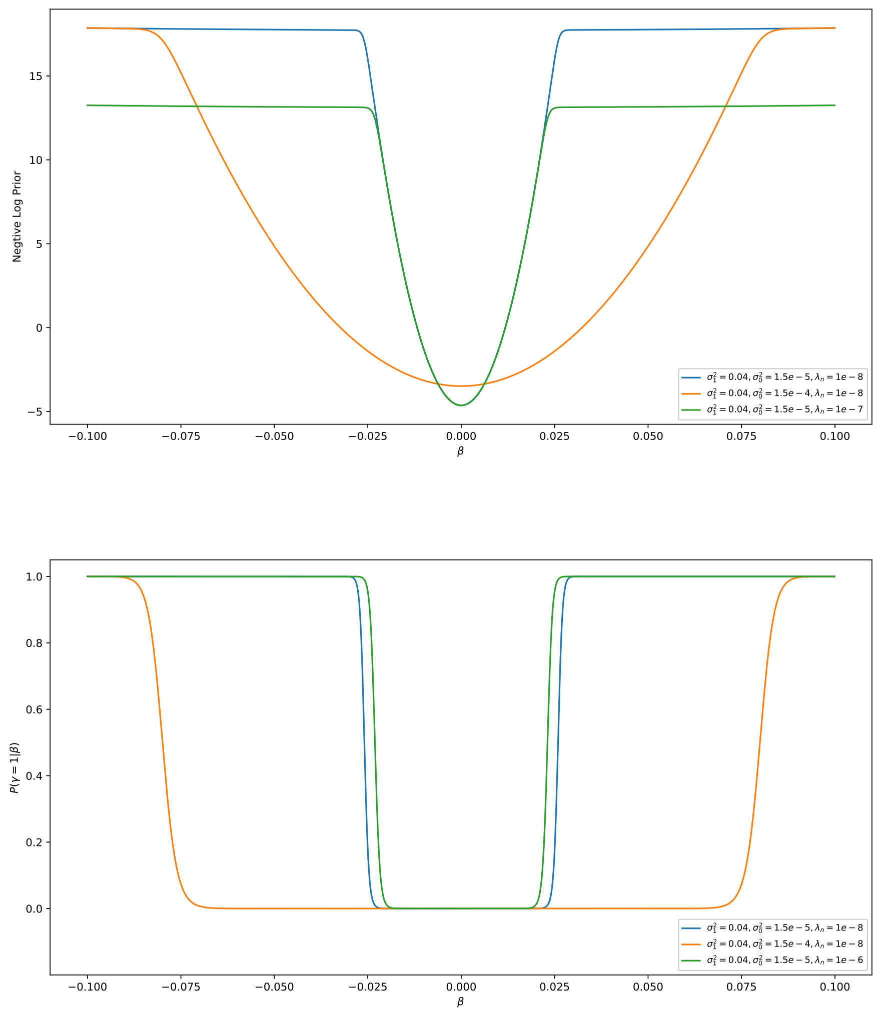


Figure 2.2. Negative log-prior and $\boldsymbol{\pi}(\boldsymbol{\gamma} = 1|\boldsymbol{\beta})$ for different choices of $\sigma_{0,n}^2$ and λ_n.

the posterior mean $\int \phi(\boldsymbol{\beta})\pi(\boldsymbol{\beta}|D_n, \eta^*, \sigma_{0,n}^*)d\boldsymbol{\beta}$. For simplicity, we will focus on SGLD with prior annealing. Our analysis can be easily generalized to other SGMCMC algorithms [70].

For a test function $\phi(\cdot)$, the difference between $\phi(\boldsymbol{\beta})$ and $\int \phi(\boldsymbol{\beta})\pi(\boldsymbol{\beta}|D_n, \eta^*, \sigma_{0,n}^*)d\boldsymbol{\beta}$ can be characterized by the Poisson equation:

$$\mathcal{L}\psi(\boldsymbol{\beta}) = \phi(\boldsymbol{\beta}) - \int \phi(\boldsymbol{\beta})\pi(\boldsymbol{\beta}|D_n, \eta^*, \sigma_{0,n}^*)d\boldsymbol{\beta},$$

where $\psi(\cdot)$ is the solution of the Poisson equation and $\mathcal{L}$ is the infinitesimal generator of the Langevin diffusion. i.e. for the following Langevin diffusion

$$\mathrm{d}\boldsymbol{\beta}^{(t)} = \nabla \log(\pi(\boldsymbol{\beta}|D_n, \eta^*, \sigma_{0,n}^*))\mathrm{d}t + \sqrt{2}I\mathrm{d}W_t,$$

where I is identity matrix and W_t is Brownian motion, we have

$$\mathcal{L}\psi(\boldsymbol{\beta}) := \langle \nabla\psi(\boldsymbol{\beta}), \nabla \log(\pi(\boldsymbol{\beta}|D_n, \eta^*, \sigma_{0,n}^*)) + \mathrm{tr}(\nabla^2\psi(\boldsymbol{\beta})).$$

Let $\mathcal{D}^k\psi$ denote the kth-order derivatives of ψ. To control the perturbation of $\phi(\boldsymbol{\beta})$, we need the following assumption about the function $\psi(\boldsymbol{\beta})$:

Assumption 2.6.1. *For $k \in \{0,1,2,3\}$, $\mathcal{D}^k\psi$ exists and there exists a function $\mathcal{V}$, s.t. $||\mathcal{D}^k\psi|| \lesssim \mathcal{V}^{p_k}$ for some constant $p_k > 0$. In addition, $\mathcal{V}$ is smooth and the expectation of $\mathcal{V}^p$ on $\boldsymbol{\beta}^{(t)}$ is bounded for some $p \leq 2\max_k\{p_k\}$, i.e. $\sup_t \mathbb{E}(\mathcal{V}^p(\boldsymbol{\beta}^{(t)})) < \infty$, $\sum_{s\in(0,1)} \mathcal{V}^p(s\boldsymbol{\beta}_1 + (1-s)\boldsymbol{\beta}_2) \lesssim \mathcal{V}^p(\boldsymbol{\beta}_1) + \mathcal{V}^p(\boldsymbol{\beta}_2)$.*

In step t of the SGLD algorithm, the drift term is replaced by $\nabla_{\boldsymbol{\beta}} \log \pi(\boldsymbol{\beta}^{(t)}|D_{m,n}^{(t)}, \eta^{(t)}, \sigma_{0,n}^{(t)})$, where $D_{m,n}^{(t)}$ is used to represent the mini-batch data used in step t. Let $\mathcal{L}_t$ be the corresponding infinitesimal generator. Let $\delta_t = \mathcal{L}_t - \mathcal{L}$. To quantify the effect of δ_t, we introduce the following assumption:

Assumption 2.6.2. *$\boldsymbol{\beta}^{(t)}$ has bounded expectation and the expectation of log-prior is Lipschitz continuous with respect to $\sigma_{0,n}$, i.e. there exists some constant M s.t. $\sup_t \mathbb{E}(|\boldsymbol{\beta}^{(t)}|) \leq M < \infty$. For all t, $|\mathbb{E}\log(\pi(\boldsymbol{\beta}^{(t)}|\lambda_n, \sigma_{0,n}^{(t_1)}, \sigma_{1,n})) - \mathbb{E}\log(\pi(\boldsymbol{\beta}^{(t)}|\lambda_n, \sigma_{0,n}^{(t_2)}, \sigma_{1,n}))| \leq M|\sigma_{0,n}^{(t_1)} - \sigma_{0,n}^{(t_2)}|$.*

Then we have the following theorem:

Theorem 2.6.1. *Suppose the model satisfy assumption 2.6.2, and a constant learning rate of ϵ is used. For a test function $\phi(\cdot)$, if the solution of the Poisson equation $\psi(\cdot)$ satisfy assumption 2.6.1, then*

$$\begin{aligned}&\mathbb{E}\left(\frac{1}{T}\sum_{t=1}^{T-1}\phi(\boldsymbol{\beta}^{(t)})-\int\phi(\boldsymbol{\beta})\pi(\boldsymbol{\beta}|D_n,\eta^*,\sigma_{0,n}^*)d\boldsymbol{\beta}\right)\\ =&O\left(\frac{1}{T\epsilon}+\frac{\sum_{t=0}^{T-1}(|\eta^{(t)}-\eta^*|+|\sigma_{0,n}^{(t)}-\sigma_{0,n}^*|)}{T}+\epsilon\right),\end{aligned} \tag{2.16}$$

where $\sigma_{0,n}^*$ is treated as a fixed constant.

Theorem 2.6.1 shows that with prior annealing, the path averaging estimator can still be used for estimating the mean and variance of the prediction and constructing the confidence interval. The detailed procedure is given in next section. For the case that a decaying learning rate is used, a similar theorem can be developed as in [70].

2.6.4 Construct Confidence Interval

Theorem 2.4.2 implies that a faithful prediction interval can be constructed for the sparse neural network learned by the proposed algorithms. In practice, for a normal regression problem with noise $N(0,\sigma^2)$, to construct the prediction interval for a test point $\boldsymbol{x}_0$, the terms σ^2 and $\Sigma=\nabla_{\boldsymbol{\gamma}^*}\mu(\boldsymbol{\beta}^*,\boldsymbol{x}_0)^T H^{-1}\nabla_{\boldsymbol{\gamma}^*}\mu(\boldsymbol{\beta}^*,\boldsymbol{x}_0)$ in Theorem 2.4.2 need to be estimated from data. Let $D_n=(\boldsymbol{x}^{(\mathrm{i})},y^{(\mathrm{i})})_{\mathrm{i}=1,\ldots,n}$ be the training set and $\mu(\boldsymbol{\beta},\cdot)$ be the predictor of the network model with parameter $\boldsymbol{\beta}$. We can follow the following procedure to construct the prediction interval for the test point $\boldsymbol{x}_0$:

- Run algorithm 1 or 2, let $\hat{\boldsymbol{\beta}}$ be an estimation of the network parameter at the end of the algorithm and $\hat{\boldsymbol{\gamma}}$ be the correspoding network structure.

- Estimate σ^2 by
$$\hat{\sigma}^2=\frac{1}{n}\sum_{\mathrm{i}=1}^{n}(\mu(\hat{\boldsymbol{\beta}},\boldsymbol{x}^{(\mathrm{i})})-y^{(\mathrm{i})})^2.$$

- Estimate Σ by
$$\hat{\Sigma}=\nabla_{\hat{\boldsymbol{\gamma}}}\mu(\hat{\boldsymbol{\beta}},\boldsymbol{x}_0)^T(-\nabla_{\hat{\boldsymbol{\gamma}}}^2 l_n(\hat{\boldsymbol{\beta}}))^{-1}\nabla_{\hat{\boldsymbol{\gamma}}}\mu(\hat{\boldsymbol{\beta}},\boldsymbol{x}_0).$$

- Construct the prediction interval as

$$\left(\mu(\hat{\boldsymbol{\beta}}, \boldsymbol{x}_0) - 1.96\sqrt{\frac{1}{n}\hat{\Sigma} + \hat{\sigma}^2}, \mu(\hat{\boldsymbol{\beta}}, \boldsymbol{x}_0) + 1.96\sqrt{\frac{1}{n}\hat{\Sigma} + \hat{\sigma}^2}\right).$$

Here, by the structure selection consistency 2.3.2 and consistency of the MLE for the learnt structure [53], we replace $\boldsymbol{\beta}^*$ and $\boldsymbol{\gamma}^*$ in Theorem 2.4.2 by $\hat{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{\gamma}}$.

If the dimension of the sparse network is still too high and the computation of $\hat{\Sigma}$ becomes prohibitive, the following Bayesian approach can be used to construct confidence intervals.

- Running SGMCMC algorithm to get a sequence of posterior samples: $\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(m)}$.

- Estimating σ^2 by $\hat{\sigma}^2 = \frac{1}{n}\sum_{\mathrm{i}=1}^{n}(y^{(\mathrm{i})} - \mu^{(\mathrm{i})})^2$, where

$$\mu^{(\mathrm{i})} = \frac{1}{m}\sum_{\mathrm{j}=1}^{m}\mu(\boldsymbol{\beta}^{(\mathrm{j})}, \boldsymbol{x}^{(\mathrm{i})}), \mathrm{i} = 1, \ldots, n,$$

- Estimate the prediction mean by

$$\hat{\mu} = \frac{1}{m}\sum_{\mathrm{i}=1}^{m}\mu(\boldsymbol{\beta}^{(\mathrm{i})}, \boldsymbol{x}_0).$$

- Estimate the prediction variance by

$$\hat{V} = \frac{1}{m}\sum_{\mathrm{i}=1}^{m}(\mu(\boldsymbol{\beta}^{(\mathrm{i})}, \boldsymbol{x}_0) - \hat{\mu})^2 + \hat{\sigma}^2.$$

- Construct the prediction interval as

$$(\mu - 1.96\sqrt{V}, \mu + 1.96\sqrt{V}).$$

2.7 Numerical Experiments

This section illustrates the performance of the proposed method on synthetic and real data examples.[1] For the synthetic example, the frequentist algorithm is employed to construct prediction intervals. The real data example involves a large network, so both the frequentist and Bayesian algorithms are employed along with comparisons with some existing network pruning methods.

2.7.1 Synthetic Example

We consider a high-dimensional nonlinear regression problem, which shows that our method can identify the sparse network structure and relevant features as well as produce prediction intervals with correct coverage rates. The explanatory variables $x_1, \ldots, x_{p_n}$ were simulated by independently generating $\mathrm{e}, z_1, \ldots, z_{p_n}$ from $N(0,1)$ and setting $x_\mathrm{i} = \frac{\mathrm{e}+z_\mathrm{i}}{\sqrt{2}}$. The response variable was generated from a nonlinear regression model:

$$y = \frac{5x_2}{1+x_1^2} + 5\sin(x_3x_4) + 2x_5 + 0x_6 + \cdots + 0x_{2000} + \epsilon,$$

where $\epsilon \sim N(0,1)$. Ten datasets were generated, each consisting of 10000 samples for training and 1000 samples for testing.

We apply both Bayesian Evidence approach 1 and prior annealing approach 2 on this data set. To demonstrate the difference of the algorithm, for Bayesian evidence approach, we use a small DNN of structure 2000-6-4-3-1. For prior annealing approach, to satisfy condition (S*), we used a DNN of structure 2000-10000-100-10-1. We use tanh as activation function. The variable selection performance were measured using the false selection rate $FSR = \frac{\sum_{\mathrm{i}=1}^{10}|\hat{S}_\mathrm{i}\backslash S|}{\sum_{\mathrm{i}=1}^{10}|\hat{S}_\mathrm{i}|}$ and negative selection rate $NSR = \frac{\sum_{\mathrm{i}=1}^{10}|S\backslash \hat{S}_\mathrm{i}|}{\sum_{\mathrm{i}=1}^{10}|S|}$, where S is the set of true variables, $\hat{S}_\mathrm{i}$ is the set of selected variables from dataset i and $|\hat{S}_\mathrm{i}|$ is the size of $\hat{S}_\mathrm{i}$. The predictive performance is measured by mean square prediction error (MSPE) and mean square fitting error (MSFE). We compare our method with the other existing variable selection methods

[1]↑The code for running these experiments can be found in https://github.com/sylydya/Sparse-Deep-Learning-A-New-Framework-Immuneto-Local-Traps-and-Miscalibration and https://github.com/sylydya/Consistent-Sparse-Deep-Learning-Theory-and-Computation

Table 2.1. Simulation Result: MSFE and MSPE were calculated by averaging over 10 datasets, and their standard deviations were given in the parentheses.

Method	$\|\hat{S}\|$	FSR	NSR	MSFE	MSPE
BNN_anneal	5(0)	0	0	2.353(0.296)	2.428(0.297)
BNN_Evidence	5(0)	0	0	2.372(0.093)	2.439(0.132)
Spinn	10.7(3.874)	0.462	0	4.157(0.219)	4.488(0.350)
DNN	-	-	-	1.1701e-5(1.1542e-6)	16.9226(0.3230)
Dropout	-	-	-	1.104(0.068)	13.183(0.716)
BART50	16.5(1.222)	0.727	0.1	11.182(0.334)	12.097(0.366)
LASSO	566.8(4.844)	0.993	0.26	8.542(0.022)	9.496(0.148)
SIS	467.2(11.776)	0.991	0.2	7.083(0.023)	10.114(0.161)

including Sparse input neural network(Spinn) [51], Bayesian adaptive regression tree (BART) [71], linear model with lasso penalty (LASSO) [72], and sure independence screening with SCAD penalty (SIS)[73]. To demonstrate the importance of selecting correct variables, we also compare our method with two dense model with the same network structure: DNN trained with dropout(Dropout) and DNN trained with no regularization(DNN). Detailed experiment setups are given in the Section 2.7.3. The results were summarized in Table 2.1. With a single run, prior annealing approach achieves similar result with the multiple-run method. The latter trained the model for 10 times and selected the best one using Bayesian evidence. While for Spinn (with LASSO penalty), even with over-parametrized structure, it performs worse than the sparse BNN model.

To quantify the uncertainty of the prediction, we conducted 100 experiments over different training sets as generated previously. We constructed 95% prediction intervals over 1000 test points. Over the 1000 test points, the average coverage rate of the prediction intervals is 94.72%(0.61%), where (0.61%) denote the standard deviation. Figure 2.3 shows the prediction intervals constructed for 20 of the testing points.

2.7.2 Real Data Example

As a different type of applications of the proposed method, we conducted unstructured network pruning experiments on CIFAR10 dataset[74]. Following the setup in [75], we train

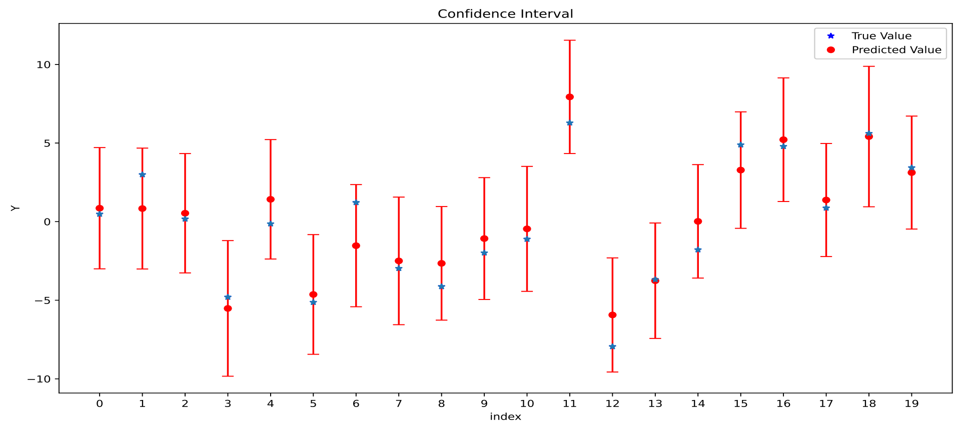


Figure 2.3. Prediction intervals of 20 testing points, where the y-axis is the response value, the x-axis is the index, and the blue point represents the true observation.

the residual network[76] with different networks size and pruned the network to different sparsity levels. The detailed experimental setup can be found in Section 2.7.3

We compared the proposed methods, BNN_anneal (Algorithm 2), BNN_average (averaged over last 75 networks simulated by the Bayesian version of the prior annealing algorithm) and BNN_BIC (multiple-run and select best model by BIC) with several state-of-the-art unstructured pruning methods, including Dynamic pruning with feedback (DPF) [75], Dynamic Sparse Reparameterization (DSR) [77] and Sparse Momentum (SM) [78]. The results of the baseline methods were taken from [75] The results of prediction accuracy for different models and target sparsity levels were summarized in Table 2.2. Due to the threshold used in step (iii) of Algorithm 2, it is hard for our method to make the pruning ratio exactly the same as the targeted one. We intentionally make the pruning ratio smaller than the target ratio, while our method still achieve better test accuracy. To further demonstrate that the proposed method result in better model calibration, we followed the setup of [79] and compared the proposed method with DPF on several metrics designed for model calibration, including negtive log likelihood (NLL), symmetrized, discretized KL distance between in and out of sample entropy distributions (JS-Distance), and expected calibration error (ECE). For JS-Distance, we used the test data of SVHN data set[2] as out-of-distribution samples. The results were summarized in Table 2.3. As discussed in [3], [79], a well calibrated model tends to have smaller NLL, larger JS-Distance and smaller ECE. The comparison shows that the proposed method outperforms DPF in most cases. In addition to the network pruning method, we also train a dense model with the standard training set up. Compared to the dense model, the sparse network has worse accuracy, but it tends to outperform the dense network in terms of ECE and JS-Distance, which indicates that sparsification is also a useful way for improving calibration of the DNN.

[2]↑The Street View House Numbers (SVHN) Dataset: http://ufldl.stanford.edu/housenumbers/

Table 2.2. ResNet network pruning results for CIFAR-10 data, which were calculated by averaging over 3 independent runs with the standard deviation reported in the parentheses.

Method	ResNet-20		ResNet-32	
	Pruning Ratio	Test Accuracy	Pruning Ratio	Test Accuracy
DNN_dense	100%	92.93(0.04)	100%	93.76(0.02)
BNN_average	19.85%(0.18%)	92.53(0.08)	9.99%(0.08%)	93.12(0.09)
BNN_anneal	19.80%(0.01%)	92.30(0.16)	9.97%(0.03%)	92.63(0.09)
BNN_BIC	19.67%(0.05%)	92.27(0.03)	9.53%(0.04%)	92.74(0.07)
SM	20%	91.54(0.16)	10%	91.54(0.18)
DSR	20%	91.78(0.28)	10%	91.41(0.23)
DPF	20%	92.17(0.21)	10%	92.42(0.18)
BNN_average	9.88%(0.02%)	91.65(0.08)	4.77%(0.08%)	91.30(0.16)
BNN_anneal	9.95%(0.03%)	91.28(0.11)	4.88%(0.02%)	91.17(0.08)
BNN_BIC	9.55%(0.03%)	91.27(0.05)	4.78%(0.01%)	91.21(0.01)
SM	10%	89.76(0.40)	5%	88.68(0.22)
DSR	10%	87.88(0.04)	5%	84.12(0.32)
DPF	10%	90.88(0.07)	5%	90.94(0.35)

Table 2.3. ResNet network pruning results for CIFAR-10 data, which were calculated by averaging over 3 independent runs with the standard deviation reported in the parentheses.

Method	Model	Pruning Ratio	NLL	JS-Distance	ECE
DNN_dense	ResNet20	100%	0.2276(0.0021)	7.9118(0.9316)	0.02627(0.0005)
BNN_average	ResNet20	9.88%(0.02%)	0.2528(0.0029)	9.9641(0.3069)	0.0113(0.0010)
BNN_anneal	ResNet20	9.95%(0.03%)	0.2618(0.0037)	10.1251(0.1797)	0.0175(0.0011)
DPF	ResNet20	10%	0.2833(0.0004)	7.5712(0.4466)	0.0294(0.0009)
BNN_average	ResNet20	19.85%(0.18%)	0.2323(0.0033)	7.7007(0.5374)	0.0173(0.0014)
BNN_anneal	ResNet20	19.80%(0.01%)	0.2441(0.0042)	6.4435(0.2029)	0.0233(0.0020)
DPF	ResNet20	20%	0.2874(0.0029)	7.7329(0.1400)	0.0391(0.0001)
DNN_dense	ResNet32	100%	0.2042(0.0017)	6.7699(0.5253)	0.02613(0.00029)
BNN_average	ResNet32	9.99%(0.08%)	0.2116(0.0012)	9.4549(0.5456)	0.0132(0.0001)
BNN_anneal	ResNet32	9.97%(0.03%)	0.2218(0.0013)	8.5447(0.1393)	0.0192(0.0009)
DPF	ResNet32	10%	0.2677(0.0041)	7.8693(0.1840)	0.0364(0.0015)
BNN_average	ResNet32	4.77%(0.08%)	0.2587(0.0022)	7.0117(0.2222)	0.0100(0.0002)
BNN_anneal	ResNet32	4.88%(0.02%)	0.2676(0.0014)	6.8440(0.4850)	0.0149(0.0006)
DPF	ResNet32	5%	0.2921(0.0067)	6.3990(0.8384)	0.0276(0.0019)

2.7.3 Experimental Setups

Synthetic Example

For prior annealing, we follow simple implementation of Algorithm given in Section 2.6.2. We run SGHMC for $T = 80000$ iterations with constant learning rate $\epsilon_t = 0.001$, momentum $1-\alpha = 0.9$ and subsample size $m = 500$. We set $\lambda_n = 1\mathrm{e}-7, \sigma_{1,n}^2 = 1\mathrm{e}-2$, $(\sigma_{0,n}^{init})^2 = 5\mathrm{e}-5$, $(\sigma_{0,n}^{end})^2 = 1\mathrm{e}-6$ and $T_1 = 5000, T_2 = 20000, T_3 = 60000$. We set temperature $\tau = 0.1$ for $t < T_3$ and for $t > T_3$, we gradually decrease temperature τ by $\tau = \frac{0.1}{t-T_3}$. After structure selection, the model is fine tuned for 40000 iterations.

For Bayesian Evidence approach 1. we ran SGD for 80,000 iterations to train the neural network with a learning rate of $\epsilon_t = 0.005$. The subsample size was set to 500. For the mixture Gaussian prior, we set $\sigma_{1,n} = 0.01$, $\sigma_{0,n} = 0.0001$, and $\lambda_n = 0.00001$. The number of independent tries was set to $T = 10$. After structure selection, the DNN was retrained using SGD for 40,000 iterations.

Spinn, Dropout and DNN are trained with the same network structure as the prior annealing method using SGD with momentum. Same as our method, we use constant learning rate 0.001, momentum 0.9, subsample size 500 and traing the model for 80000 iterations. For Spinn, we use LASSO penalty and the regularization parameter is selected from $\{0.05, 0.06, \ldots, 0.15\}$ according to the performance on validation data set. For Dropout, the dropout rate is set to be 0.2 for the first layer and 0.5 for the other layers. Other baseline methods BART50, LASSO, SIS are implemented using R-package $randomForest$, $glmnet$, $BART$ and SIS respectively with default parameters.

Real Data Examples

We follow the standard training procedure as in [75], i.e. we train the model with SGHMC for $T = 300$ epochs, with initial learning rate $\epsilon_0 = 0.1$, momentum $1 - \alpha = 0.9$, temperature $\tau = 0.001$, mini-batch size $m = 128$. The learning rate is divided by 10 at 150th and 225th epoch.

For prior annealing, we follow the implementation given in section 2.6.2 and use $T_1 = 150, T_2 = 200, T_3 = 225$, where T_is are number of epochs. We set temperature $\tau = 0.01$ for $t < T_3$ and gradually decrease τ by $\tau = \frac{0.01}{t-T_3}$ for $t > T_3$. We set $\sigma_{1,n}^2 = 0.04$ and $(\sigma_{0,n}^{init})^2 = 10 \times (\sigma_{0,n}^{end})^2$ and use different $\sigma_{0,n}^{end}, \lambda_n$ for different network size and target sparsity level. The detailed settings are given below:

- ResNet20 with target sparsity level 20%: $(\sigma_{0,n}^{end})^2 = 1.5\text{e}-5, \lambda_n = 1\text{e}-8$
- ResNet20 with target sparsity level 10%: $(\sigma_{0,n}^{end})^2 = 6\text{e}-5, \lambda_n = 1\text{e}-9$
- ResNet32 with target sparsity level 10%: $(\sigma_{0,n}^{end})^2 = 3\text{e}-5, \lambda_n = 2\text{e}-9$
- ResNet32 with target sparsity level 5%: $(\sigma_{0,n}^{end})^2 = 1\text{e}-4, \lambda_n = 2\text{e}-8$

For BIC approach, the only different setting with the prior annealing approach is the prior. We set $\sigma_{1,n}^2 = 0.02$ and tried different values for $\sigma_{0,n}$ and λ_n to achieve different sparsity levels. For ResNet-20, to achieve 10% target sparsity, we set $\sigma_{0,n}^2 = 4\text{e}-5$ and $\lambda_n = 1\text{e}-6$; to achieve 20% target sparsity, we set $\sigma_{0,n}^2 = 6\text{e}-6$ and $\lambda_n = 1\text{e}-7$. For ResNet-32, to achieve 5% target sparsity, we set $\sigma_{0,n}^2 = 6\mathrm{e}-5$ and $\lambda_n = 1\mathrm{e}-7$; to achieve 10% target sparsity, we set $\sigma_{0,n}^2 = 2\mathrm{e}-5$ and $\lambda_n = 1\mathrm{e}-5$.

2.8 Discussion

In this dissertation, we provide a complete treatment for sparse DNNs in both theory and computation. The sparse DNN can be simply viewed as a nonlinear statistical model which, like a traditional statistical model, possesses many nice properties such as posterior consistency, variable selection consistency, asymptotic normality, and asymptotically optimal generalization bound.

In computation, we proposed to use Bayesian evidence or BIC for eliciting sparse DNN models learned by an optimization method in multiple runs with different initializations and an prior annealing approach for over-parametrized DNN model. The computation complexity of the proposed method is of same order as standard SGD method for DNN training. Our numerical results show that the proposed method can perform very well in large-scale network compression and high-dimensional nonlinear variable selection. The networks learned by the proposed method tend to predict better than the existing methods and have better calibration.

In this chapter, we choose the two-mixture Gaussian prior for the weights and biases of the DNN, mainly for the sake of computational convenience. Other choices, such as two-mixture Laplace prior [80], which will lead to the same posterior contraction with an appropriate choice for the prior hyperparameters. To be more specific, Theorem 2.9.1 establishes sufficient conditions that guarantee the posterior consistency, and any prior distribution satisfying the sufficient conditions can yield consistent posterior inferences for the DNN.

Beyond the absolutely continuous prior, the hierarchical prior used in [14] and [15] can be adopted for DNNs. To be more precise, one can assume that

$$\boldsymbol{\beta}_{\boldsymbol{\gamma}} \mid \boldsymbol{\gamma} \sim N(0, \sigma_{1,n}^2 \boldsymbol{I}_{\|\boldsymbol{\gamma}\| \times \|\boldsymbol{\gamma}\|}), \quad \boldsymbol{\beta}_{\boldsymbol{\gamma}^c} = 0; \tag{2.17}$$

$$\pi(\boldsymbol{\gamma}) \propto \lambda_n^{\|\boldsymbol{\gamma}\|} (1-\lambda_n)^{K_n - \|\boldsymbol{\gamma}\|} \mathbf{1}\left\{1 \le \|\boldsymbol{\gamma}\| \le \bar{r}_n, \boldsymbol{\gamma} \in \mathcal{G}\right\}, \tag{2.18}$$

where $\boldsymbol{\beta}_{\gamma^c}$ is the complement of $\boldsymbol{\beta}_{\gamma}$, $\|\boldsymbol{\gamma}\|$ is the number of nonzero elements of $\boldsymbol{\gamma}$, $\boldsymbol{I}_{\|\gamma\|\times\|\gamma\|}$ is a $\|\boldsymbol{\gamma}\| \times \|\boldsymbol{\gamma}\|$ identity matrix, $\bar{r}_n$ is the maximally allowed size of candidate networks, $\mathcal{G}$ is the set of valid DNNs, and the hyperparameter λ_n, as in (2.3), can be read as an approximate prior probability for each connection or bias to be included in the DNN. Under this prior, the product of the weight or bias and its indicator follows a discrete spike-and-slab prior distribution, i.e.

$$\boldsymbol{w}_{\text{ij}}^h \boldsymbol{\gamma}_{\text{ij}}^{\boldsymbol{w}^h} | \boldsymbol{\gamma}_{\text{ij}}^{\boldsymbol{w}^h} \sim \boldsymbol{\gamma}_{\text{ij}}^{\boldsymbol{w}^h} N(0, \sigma_{1,n}^2) + (1 - \boldsymbol{\gamma}_{\text{ij}}^{\boldsymbol{w}^h})\delta_0, \quad \boldsymbol{b}_k^h \boldsymbol{\gamma}_k^{\boldsymbol{b}^h} | \boldsymbol{\gamma}_k^{\boldsymbol{b}^h} \sim \boldsymbol{\gamma}_k^{\boldsymbol{b}^h} N(0, \sigma_{1,n}^2) + (1 - \boldsymbol{\gamma}_k^{\boldsymbol{b}^h})\delta_0,$$

where δ_0 denotes the Dirac delta function. Under this hierarchical prior, it is not difficult to show that the posterior consistency and structure selection consistency theory developed in this chapter still hold. However, from the computational perspective, the hierarchical prior might be inferior to the mixture Gaussian prior adopted in the chapter, as the posterior $\pi(\boldsymbol{\beta}_{\gamma}, \boldsymbol{\gamma}|D_n)$ is hard to be optimized or simulated from. It is known that directly simulating from $\pi(\boldsymbol{\beta}_{\gamma}, \boldsymbol{\gamma}|D_n)$ using an acceptance-rejection based MCMC algorithm can be time consuming. A feasible way is to formulate the prior of $\boldsymbol{\beta}_{\gamma}$ as $\boldsymbol{\beta}_{\gamma} = \boldsymbol{\theta} \otimes \boldsymbol{\gamma}$, where $\boldsymbol{\theta} \sim N(0, \sigma_{1,n}^2 I_{H_n \times H_n})$ can be viewed as a latent variable and $\otimes$ denotes entry-wise product. Then one can first simulate from the marginal posterior $\pi(\boldsymbol{\theta}|D_n)$ using a stochastic gradient MCMC algorithm and then make inference of the network structure based on the conditional posterior $\pi(\boldsymbol{\gamma}|\boldsymbol{\theta}, D_n)$. We note that the gradient $\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{\theta}|D^n)$ can be approximated based on the following identity developed in [81],

$$\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{\theta}|D^n) = \sum_{\boldsymbol{\gamma}} \pi(\boldsymbol{\gamma}|\boldsymbol{\theta}, D^n) \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{\theta}|\boldsymbol{\gamma}, D^n),$$

where D_n can be replaced by a dataset duplicated with mini-batch samples if the subsampling strategy is used to accelerate the simulation. This identity greatly facilitates the simulations for the dimension jumping problems, which requires only some samples to be drawn from the conditional posterior $\pi(\boldsymbol{\gamma}|\boldsymbol{\theta}, D^n)$ for approximating the gradient $\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{\theta}|D^n)$ at each iteration. A further exploration of this discrete prior for its use in deep learning is of great interest, although there are some difficulties needing to be addressed in computation.

2.9 Technical Proofs

This section is organized as follows. Section 2.9.1 gives the proofs on posterior consistency, Section 2.9.2 gives the proofs on structure selection consistency, Section 2.9.3 gives the proofs on BvM theorem for weights and predictions. Section 2.9.4 gives the proofs on generalization bounds, and Section 2.9.5 gives some mathematical facts of the sparse DNN.

2.9.1 Proofs on Posterior Consistency

Basic Formulas of Bayesian Neural Networks

Normal Regression.

Let p_μ denote the density of $N(\mu, \sigma^2)$ where σ^2 is a known constant, and let $p_{\boldsymbol{\beta}}$ denote the density of $N(\mu(\boldsymbol{\beta}, \boldsymbol{x}), \sigma^2)$. Extension to the case σ^2 is unknown is simple by following the arguments given in [39]. In this case, an inverse gamma prior can be assumed for σ^2 as suggested by [39]. Define the Kullback-Leibler divergence as $d_0(p, p^*) = \int p^* \log(p^*/p)$ for two densities p and p^*. Define a distance $d_t(p, p^*) = t^{-1}(\int p^*(p^*/p)^t - 1)$ for any $t > 0$, which decreases to d_0 as t decreases toward 0. A straightforward calculation shows

$$d_1(p_{\mu_1}, p_{\mu_2}) = \int p_{\mu_1}(p_{\mu_1}/p_{\mu_1}) - 1 = \exp\left(\frac{1}{\sigma^2}(\mu_2 - \mu_1)^2\right) - 1 = \frac{1}{\sigma^2}(\mu_2 - \mu_1)^2 + o((\mu_2 - \mu_1)^3), \tag{2.19}$$

$$d_0(p_1, p_2) = \frac{1}{2\sigma^2}(\mu_1 - \mu_2)^2. \tag{2.20}$$

Logistic Regression.

Let p_μ denote the probability mass function with the success probability given by $1/(1 + \mathrm{e}^{-\mu})$. Similarly, we define $p_{\boldsymbol{\beta}}$ as the logistic regression density for a binary classification DNN with parameter $\boldsymbol{\beta}$. For logistic regression, we have

$$d_1(p_{\mu_1}, p_{\mu_2}) = \int p_{\mu_2}(p_{\mu_2}/p_{\mu_1}) - 1 = \frac{\mathrm{e}^{2\mu_2 - \mu_1} + \mathrm{e}^{\mu_1} - 2\mathrm{e}^{\mu_2}}{(1 + \mathrm{e}^{\mu_2})^2},$$

which, by the mean value theorem, can be written as

$$d_1(p_{\mu_1}, p_{\mu_2}) = \frac{\mathrm{e}^{\mu'} - \mathrm{e}^{2\mu_2-\mu'}}{(1+\mathrm{e}^{\mu_2})^2}(\mu_{\mu_1} - \mu_2) = \frac{\mathrm{e}^{\mu'}(1-\mathrm{e}^{2\mu_2-2\mu'})}{(1+\mathrm{e}^{\mu_2})^2}(\mu_1-\mu_2),$$

where μ' denotes an intermediate point between μ_1 and μ_2, and thus $|\mu' - \mu_2| \le |\mu_1 - \mu_2|$. Further, by Taylor expansion, we have

$$\mathrm{e}^{\mu'} = \mathrm{e}^{\mu_2}[1 + (\mu' - \mu_2) + O((\mu'-\mu_2)^2)], \quad \mathrm{e}^{2\mu_2-2\mu'} = 1 + 2(\mu_2 - \mu') + O((\mu_2-\mu')^2).$$

Therefore,

$$d_1(p_{\mu_1}, p_{\mu_2}) \le \frac{\mathrm{e}^{\mu_2}}{(1+\mathrm{e}^{\mu_2})^2}\left[2|\mu_2 - \mu'| + O((\mu_2-\mu')^2)\right]|\mu_1 - \mu_2| \le \frac{1}{2}(\mu_1-\mu_2)^2 + O((\mu_1-\mu_2)^3), \tag{2.21}$$

and

$$d_0(p_u, p_v) = \int p_v(\log p_v - \log p_u)v_y(dy) = \log(1+\mathrm{e}^{\mu_u}) - \log(1+\mathrm{e}^{\mu_v}) + \frac{\mathrm{e}^{\mu_v}}{1+\mathrm{e}^{\mu_v}}(\mu_v - \mu_u).$$

By the mean value theorem, we have

$$d_0(p_u, p_v) = \frac{\mathrm{e}^{\mu'}}{1+\mathrm{e}^{\mu'}}(\mu_u - \mu_v) + \frac{\mathrm{e}^{\mu_v}}{1+\mathrm{e}^{\mu_v}}(\mu_v - \mu_u) = [\frac{\mathrm{e}^{\mu_v}}{1+\mathrm{e}^{\mu_v}} - \frac{\mathrm{e}^{\mu'}}{1+\mathrm{e}^{\mu'}}](\mu_v - \mu_u), \tag{2.22}$$

where μ' denotes an intermediate point between μ_u and μ_v.

Posterior Consistency of General Statistical Models

We first introduce a lemma concerning posterior consistency of general statistical models. This lemma has been proved in [39]. Let $\mathbb{P}_n$ denote a sequence of sets of probability densities, let $\mathbb{P}_n^c$ denote the complement of $\mathbb{P}_n$, and let ϵ_n denote a sequence of positive numbers. Let $N(\epsilon_n, \mathbb{P}_n)$ be the minimum number of Hellinger balls of radius ϵ_n that are needed to cover $\mathbb{P}_n$, i.e., $N(\epsilon_n, \mathbb{P}_n)$ is the minimum of all k's such that there exist sets $S_{\mathrm{j}} = \{p : d(p, p_{\mathrm{j}}) \le \epsilon_n\}$, $\mathrm{j} = 1, \ldots, k$, with $\mathbb{P}_n \subset \cup_{\mathrm{j}=1}^k S_{\mathrm{j}}$ holding, where $d(p,q) = \sqrt{\int(\sqrt{p} - \sqrt{q})^2}$ denotes the Hellinger distance between the two densities p and q.

Let $D_n = (z^{(1)}, \ldots, z^{(n)})$ denote the dataset, where the observations $z^{(1)}, \ldots, z^{(n)}$ are iid with the true density p^*. The dimension of $z^{(1)}$ and p^* can depend on n. Define $\pi(\cdot)$ as the prior density, and $\pi(\cdot|D_n)$ as the posterior. Define $\hat{\pi}(\epsilon) = \pi[d(p, p^*) > \epsilon | D_n]$ for each $\epsilon > 0$. Define the KL divergence as $d_0(p, p^*) = \int p^* \log(p^*/p)$. Define $d_t(p, p^*) = t^{-1}(\int p^*(p^*/p)^t - 1)$ for any $t > 0$, which decreases to d_0 as t decreases toward 0. Let P^* and E^* denote the probability measure and expectation for the data D_n, respectively. Define the conditions:

(a) $\log N(\epsilon_n, \mathbb{P}_n) \leq n\epsilon_n^2$ for all sufficiently large n;

(b) $\pi(\mathbb{P}_n^c) \leq \mathrm{e}^{-bn\epsilon_n^2}$ for all sufficiently large n;

(c) $\pi[p : d_t(p, p^*) \leq b'\epsilon_n^2] \geq \mathrm{e}^{-b'n\epsilon_n^2}$ for all sufficiently large n and some $t > 0$,

where $2 > b > 2b' > 0$ are positive constants. The following lemma is due to the same argument of [39, Proposition 1].

Lemma 2.9.1. *Under the conditions (a), (b) and (c) (for some $t > 0$), given sufficiently large n, we have*

(i) $P^* \left[\hat{\pi}(4\epsilon_n) \geq 2\mathrm{e}^{-0.5n\epsilon_n^2 \min\{1, 2-x, b-x, t(x-2b')\}}\right] \leq 2\mathrm{e}^{-0.5n\epsilon_n^2 \min\{1, 2-x, b-x, t(x-2b')\}}$,

(ii) $E^*\hat{\pi}(4\epsilon_n) \leq 4\mathrm{e}^{-n\epsilon_n^2 \min\{1, 2-x, b-x, t(x-2b')\}}$.

for any $2b' < x < b$.

General Shrinkage Prior Settings for Deep Neural Networks

Let $\boldsymbol{\beta}$ denote the vector of parameters, including the weights of connections and the biases of the hidden and output units, of a deep neural network. Consider a general prior setting that all entries of $\boldsymbol{\beta}$ are subject to independent continuous prior π_b, i.e., $\pi(\boldsymbol{\beta}) = \prod_{\mathrm{j}=1}^{K_n} \pi_b(\beta_{\mathrm{j}})$. Theorem 2.9.1 provides a sufficient condition for posterior consistency.

Theorem 2.9.1 (Posterior consistency). *Assume the conditions A.1, A.2 and A.3 hold, if the prior* $\pi(\boldsymbol{\beta})$ *satisfies that*

$$\log(1/\underline{\pi}_b) = O(H_n \log n + \log \overline{L}), \tag{2.23}$$

$$\pi_b\{[-\eta_n, \eta_n]\} \geq 1 - \frac{1}{K_n}\exp\{-\tau[H_n \log n + \log \overline{L} + \log p_n]\} \text{ and } \pi_b\{[-\eta'_n, \eta'_n]\} \geq 1 - \frac{1}{K_n}, \tag{2.24}$$

$$-\log\left[K_n \pi_b(|\beta_{\mathrm{j}}| > M_n)\right] \succ n\epsilon_n^2, \tag{2.25}$$

for some $\tau > 0$*, where* $\eta_n < 1/\{\sqrt{n}K_n(n/H_n)^{H_n}(c_0 M_n)^{H_n}\}$*,* $\eta'_n < 1/\{\sqrt{n}K_n(r_n/H_n)^{H_n}(c_0 E_n)^{H_n}\}$ *with some* $c_0 > 1$*,* $\underline{\pi}_b$ *is the minimal density value of* π_b *within interval* $[-E_n - 1, E_n + 1]$*, and* M_n *is some sequence satisfying* $\log(M_n) = O(\log(n))$*. Then, there exists a sequence* ϵ_n*, satisfying* $n\epsilon_n^2 \asymp r_n H_n \log n + r_n \log \overline{L} + s_n \log p_n + n\varpi_n^2$ *and* $\epsilon_n \prec 1$*, such that*

$$\begin{aligned} &P^*\left\{\pi[d(p_\beta, p_{\mu^*}) > 4\epsilon_n | D_n] \geq 2e^{-nc\epsilon_n^2}\right\} \leq 2e^{-cn\epsilon_n^2}, \\ &E^*_{D_n}\pi[d(p_\beta, p_{\mu^*}) > 4\epsilon_n | D_n] \leq 4e^{-2cn\epsilon_n^2}. \end{aligned} \tag{2.26}$$

for some $c > 0$*.*

To prove Theorem 2.9.1, we first introduce a useful Lemma:

Lemma 2.9.2 (Theorem 1 of [82]). *Let* $X \sim B(n, v)$ *be a Binomial random variable. For any* $1 < k < n - 1$*,*

$$Pr(X \geq k+1) \leq 1 - \Phi(sign(k - nv)\{2nH(v, k/n)\}^{1/2}),$$

where Φ *is the cumulative distribution function (CDF) of the standard Gaussian distribution and* $H(v, k/n) = (k/n)\log(k/nv) + (1 - k/n)\log[(1 - k/n)/(1 - v)]$*.*

Proof of Theorem 2.9.1

Theorem 2.9.1 can be proved using Lemma 2.9.1, so it suffices to verify conditions (a)-(c) given in Section 2.9.1.

Checking condition (c) for $t = 1$*:*

Consider the set $A = \{\boldsymbol{\beta} : \max_{\mathrm{j}\in\gamma^*} \|\beta_{\mathrm{j}} - \beta^*_{\mathrm{j}}\|_\infty \le \omega_n, \max_{\mathrm{j}\notin\gamma^*} \|\beta_{\mathrm{j}} - \beta^*_{\mathrm{j}}\|_\infty \le \omega'_n\}$, where $\omega_n = c_1\epsilon_n/[H_n(r_n/H_n)^{H_n}(c_0E_n)^{H_n}]$ and $\omega'_n = c_1\epsilon_n/[K_n(r_n/H_n)^{H_n}(c_0E_n)^{H_n}]$ for some constant $c_1 > 0$ and $c_0 > 1$. If $\boldsymbol{\beta} \in A$, then by Lemma 2.9.5, we have $|\mu(\boldsymbol{\beta},\boldsymbol{x}) - \mu(\boldsymbol{\beta}^*,\boldsymbol{x})| \le 3c_1\epsilon_n$. By condition A.2.1, $|\mu(\boldsymbol{\beta},\boldsymbol{x}) - \mu^*(\boldsymbol{x})| \le 3c_1\epsilon_n + \varpi_n$. Combining it with (2.19)–(2.22), for both normal and logistic models, we have

$$d_1(p_{\boldsymbol{\beta}}, p_{\mu^*}) \le C(1+o(1))E_x(\mu(\boldsymbol{\beta},\boldsymbol{x}) - \mu^*(\boldsymbol{x}))^2 \le C(1+o(1))(3c_1\epsilon_n + \varpi_n)^2, \quad \text{if } \boldsymbol{\beta} \in A,$$

for some constant C. Thus for any small $b' > 0$, condition (c) holds as long as that c_1 is sufficiently small, $n\epsilon_n^2 \ge M_0 n\varpi_n^2$ for large M_0, and the prior satisfies $-\log \boldsymbol{\pi}(A) \le b'n\epsilon_n^2$.

Since $\boldsymbol{\pi}(A) \ge (2\underline{\boldsymbol{\pi}}_b\omega_n)^{r_n} \times \boldsymbol{\pi}(\{\max_{\mathrm{j}\notin\gamma^*} \|\beta_{\mathrm{j}}\| \le \omega'_n\})$, $\boldsymbol{\pi}_b([-\omega'_n, \omega'_n]) \ge 1 - 1/K_n$ (due to the fact $\omega'_n \gg \eta'_n$), and $\log(1/\omega_n) \asymp \log(1/\epsilon_n) + H_n \log E_n + H_n \log(r_n/H_n) + \text{constant} = O(H_n \log n)$ (note that $\log(1/\epsilon_n) = O(\log n)$), the above requirement holds when $n\epsilon_n^2 \ge M_0 r_n H_n \log n$ for some sufficiently large constant M_0.

Checking condition (a):

Let $\mathbb{P}_n$ denote the set of all DNN models whose weight parameter $\boldsymbol{\beta}$ satisfies that

$$\boldsymbol{\beta} \in B_n = \{|\beta_{\mathrm{j}}| \le M_n, \boldsymbol{\gamma}_{\boldsymbol{\beta}} = \{\mathrm{i} : |\beta_{\mathrm{i}}| \ge \delta'_n\} \text{ satisfies } |\boldsymbol{\gamma}_{\boldsymbol{\beta}}| \le k_n r_n \text{ and } |\boldsymbol{\gamma}_{\boldsymbol{\beta}}|_{\mathrm{in}} \le k'_n s_n\}, \quad (2.27)$$

where $|\boldsymbol{\gamma}|_{\mathrm{in}}$ denotes the input dimension of sparse network $\boldsymbol{\gamma}$, $k_n(\le n/r_n)$ and $k'_n(\le n/s_n)$ will be specified later, and $\delta_n = c_1\epsilon_n/[H_n(k_nr_n/H_n)^{H_n}(c_0M_n)^{H_n}]$ and
$\delta'_n = c_1\epsilon_n/[K_n(k_nr_n/H_n)^{H_n}(c_0M_n)^{H_n}]$ for some constant $c_1 > 0$ and $c_0 > 1$. Consider two parameter vectors $\boldsymbol{\beta}^u$ and $\boldsymbol{\beta}^v$ in set B_n, such that there exists a model $\boldsymbol{\gamma}$ with $|\boldsymbol{\gamma}| \le k_nr_n$ and $|\boldsymbol{\gamma}|_{\mathrm{in}} \le k'_ns_n$, and $|\beta^u_{\mathrm{j}} - \beta^v_{\mathrm{j}}| \le \delta_n$ for all $\mathrm{j} \in \boldsymbol{\gamma}$, $\max(|\beta^u_{\mathrm{j}}|, |\beta^v_{\mathrm{j}}|) \le \delta'_n$ for all $\mathrm{j} \notin \boldsymbol{\gamma}$. Hence, by Lemma 2.9.5, we have that $|\mu(\boldsymbol{\beta}^u,\boldsymbol{x}) - \mu(\boldsymbol{\beta}^v,\boldsymbol{x})|^2 \le 9c_1^2\epsilon_n^2$, and furthermore, due to (2.19)-(2.22), we can easily derive that

$$d(p_{\boldsymbol{\beta}^u}, p_{\boldsymbol{\beta}^v}) \le \sqrt{d_0(p_{\boldsymbol{\beta}^u}, p_{\boldsymbol{\beta}^v})} \le \sqrt{(9+o(1))c_1^2C\epsilon_n^2} \le \epsilon_n,$$

for some C, given a sufficiently small c_1. On the other hand, if some $\boldsymbol{\beta}^u \in B_n$ and its connections whose magnitudes are larger than δ'_n don't form a valid network, then by Lemma

2.9.5 and (2.19)-(2.22), we also have that $d(p_{\beta^u}, p_{\beta^o}) \leq \epsilon_n$, where $\boldsymbol{\beta}^o = 0$ denotes a empty output network.

Given the above results, one can bound the packing number $N(\mathbb{P}_n, \epsilon_n)$ by $\sum_{\mathrm{j}=1}^{k_n r_n} \mathcal{X}_{H_n}^{\mathrm{j}} \left(\frac{2M_n}{\delta_n}\right)^{\mathrm{j}}$, where $\mathcal{X}_{H_n}^{\mathrm{j}}$ denotes the number of all valid networks who has exact j connection and has no more than $k'_n s_n$ inputs. Since $\log \mathcal{X}_{H_n}^{\mathrm{j}} \leq k'_n s_n \log p_n + \mathrm{j} \log(k'_n s_n L_1 + H_n \overline{L}^2)$,

$$\begin{aligned} \log N(\mathbb{P}_n, \epsilon_n) &\leq \log k_n r_n + k_n r_n \log H_n + 2k_n r_n \log(\overline{L} + k'_n s_n) + k'_n s_n \log p_n \\ &\quad + k_n r_n \log \frac{2M_n H_n (k_n r_n / H_n)^{H_n} M_n^{H_n}}{c_1 \epsilon_n} \\ &= k_n r_n * O\{H_n \log n + \log \overline{L} + \text{constant}\} + k'_n s_n \log p_n \end{aligned}$$

where the second inequality is due to $\log M_n = O(\log n)$, $k_n r_n \leq n$ and $k'_n s_n \leq n$. We can choose k_n and k'_n such that $k_n r_n\{H_n \log n + \log \overline{L}\} \asymp k'_n s_n \asymp n\epsilon_n^2$ and $\log N(\mathbb{P}_n, \epsilon_n) \leq n\epsilon_n^2$.

Checking condition (b):

$\boldsymbol{\pi}(\mathbb{P}_n^c) \leq Pr(\text{Binomial}(K_n, v_n) > k_n r_n) + K_n \boldsymbol{\pi}_b(|\beta_{\mathrm{j}}| > M_n) + Pr(|\boldsymbol{\gamma}_{\boldsymbol{\beta}}|_{\mathrm{in}} \geq k'_n s_n)$, where $v_n = 1 - \boldsymbol{\pi}_b([-\delta'_n, \delta'_n])$. By the condition of $\boldsymbol{\pi}_b$ and the fact that $\delta'_n \gg \eta_n$, $v_n \leq \exp\{-\tau[H_n \log n + \log \overline{L} + \log p_n)] - \log K_n\}$ for some positive constant τ.

Hence, by Lemma 2.9.2, $-\log Pr(\text{Binomial}(K_n, v_n) > k_n r_n) \approx \tau k_n r_n [H_n \log n + \log \overline{L} + \log p_n] \gtrsim n\epsilon_n^2$ due to the choice of k_n, and $-\log Pr(|\boldsymbol{\gamma}_{\boldsymbol{\beta}}|_{\mathrm{in}} \geq k'_n s_n) \approx k'_n s_n [\tau(H_n \log n + \log \overline{L} + \log p_n) + \log(K_n / L_1 p_n)] \gtrsim n\epsilon_n^2$ due to the choice of k'_n. Thus, condition (b) holds as well.

Proof of Theorem 2.2.1

Proof. It suffices to verify the conditions listed in Theorem 2.9.1. Let $M_n = \max(\sqrt{2n}\sigma_{1,n}, E_n)$. Condition (2.23) is due to $E_n^2/2\sigma_{1,n}^2 + \log \sigma_{1,n}^2 = O[H_n \log n + \log \overline{L}]$; Condition (2.24) can be verified by $\lambda_n = 1/\{K_n [n^{H_n} (\overline{L} p_n)]^{\tau'}\}$ and $\sigma_{0,n} \prec 1/\{\sqrt{n} K_n (n/H_n)^{H_n} (c_0 M_n)^{H_n}\}$; Condition (2.25) can be verified by $M_n \geq 2n\sigma_{0,n}^2$ and $\tau[H_n \log n + \log \overline{L} + \log p_n] + M_n^2 / 2\sigma_{1,n}^2 \geq n$.

□

2.9.2 Proofs on Structure Selection Consistency

Proof of Theorem 2.3.1

Proof.

$$\begin{aligned}
\max|q_{\mathrm{i}} - \mathrm{e}_{\mathrm{i}|\nu(\gamma^*,\beta^*)}| &\leq \max \int \sum_{\gamma} |\mathrm{e}_{\mathrm{i}|\nu(\gamma,\beta)} - \mathrm{e}_{\mathrm{i}|\nu(\gamma^*,\beta^*)}|\boldsymbol{\pi}(\boldsymbol{\gamma}|\boldsymbol{\beta}, D_n)\boldsymbol{\pi}(\boldsymbol{\beta}|D_n)d\boldsymbol{\beta} \\
=&\max \int_{A(4\epsilon_n)} \sum_{\gamma} |\mathrm{e}_{\mathrm{i}|\nu(\gamma,\beta)} - \mathrm{e}_{\mathrm{i}|\nu(\gamma^*,\beta^*)}|\boldsymbol{\pi}(\boldsymbol{\gamma}|\boldsymbol{\beta}, D_n)\boldsymbol{\pi}(\boldsymbol{\beta}|D_n)d\boldsymbol{\beta} + \rho(4\epsilon_n) \qquad (2.28) \\
\leq&\hat{\boldsymbol{\pi}}(4\epsilon_n) + \rho(4\epsilon_n) \xrightarrow{p} 0,
\end{aligned}$$

where $\xrightarrow{p}$ denotes convergence in probability, and $\hat{\boldsymbol{\pi}}(c)$ denotes the posterior probability of the set $A(c) = \{\boldsymbol{\beta} : d(p_{\boldsymbol{\beta}}, p_{\mu^*}) \geq c\}$. The last convergence is due to the identifiability condition B.1 and the posterior consistency result. This completes the proof of part (i).

Part (ii) & (iii): They are directly implied by part (i).

□

Proof of Theorem 2.3.2

Proof. Let $\boldsymbol{u} = \sqrt{n}(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}) = (u_1, \ldots, u_{K_n})^T$, and let $g(\boldsymbol{\beta}) = nh(\boldsymbol{\beta}) - nh(\hat{\boldsymbol{\beta}}) - \frac{n}{2}\sum h_{\mathrm{i,j}}(\hat{\boldsymbol{\beta}})(\beta_{\mathrm{i}} - \hat{\beta}_{\mathrm{i}})(\beta_{\mathrm{j}} - \hat{\beta}_{\mathrm{j}})$. It is easy to see that for all $1 \leq \mathrm{i}_1, \ldots, \mathrm{i}_d \leq K_n$, $g_{\mathrm{i}_1,\ldots,\mathrm{i}_d}(\hat{\boldsymbol{\beta}}) = 0$ if $1 \leq d \leq 2$, and $g_{\mathrm{i}_1,\ldots,\mathrm{i}_d}(\boldsymbol{\beta}) = nh_{\mathrm{i}_1,\ldots,\mathrm{i}_d}(\boldsymbol{\beta})$ if $d \geq 3$.

Consider Taylor's expansions of $b(\boldsymbol{\beta})$ and $\exp(g(\boldsymbol{\beta}))$ at $\hat{\boldsymbol{\beta}}$, we have

$$\begin{aligned}
b(\boldsymbol{\beta}) &= b(\hat{\boldsymbol{\beta}}) + \sum b_{\mathrm{i}}(\hat{\boldsymbol{\beta}})(\beta_{\mathrm{i}} - \hat{\beta}_{\mathrm{i}}) + \frac{1}{2}\sum b_{\mathrm{i,j}}(\tilde{\boldsymbol{\beta}})(\beta_{\mathrm{i}} - \hat{\beta}_{\mathrm{i}})(\beta_{\mathrm{j}} - \hat{\beta}_{\mathrm{j}}) \\
&= b(\hat{\boldsymbol{\beta}}) + \frac{1}{\sqrt{n}}\sum b_{\mathrm{i}}(\hat{\boldsymbol{\beta}})u_{\mathrm{i}} + \frac{1}{2n}\sum b_{\mathrm{i,j}}(\tilde{\boldsymbol{\beta}})u_{\mathrm{i}}u_{\mathrm{j}},
\end{aligned}$$

$$\begin{aligned}
\mathrm{e}^{g(\boldsymbol{\beta})} &= 1 + \frac{n}{3!}\sum_{\mathrm{i,j},k} h_{\mathrm{i,j},k}(\hat{\boldsymbol{\beta}})(\beta_{\mathrm{i}} - \hat{\beta}_{\mathrm{i}})(\beta_{\mathrm{j}} - \hat{\beta}_{\mathrm{j}})(\beta_k - \hat{\beta}_k) \\
&+ \frac{n}{4!}\mathrm{e}^{g(\check{\beta})}\sum_{\mathrm{i,j},k,l} h_{\mathrm{i,j},k,l}(\check{\boldsymbol{\beta}})(\beta_{\mathrm{i}} - \hat{\beta}_{\mathrm{i}})(\beta_{\mathrm{j}} - \hat{\beta}_{\mathrm{j}})(\beta_k - \hat{\beta}_k)(\beta_l - \hat{\beta}_l) \\
&= 1 + \frac{1}{6\sqrt{n}}\sum h_{\mathrm{i,j},k}(\hat{\boldsymbol{\beta}})u_{\mathrm{i}}u_{\mathrm{j}}u_k + \frac{1}{24n}\mathrm{e}^{g(\check{\beta})}\sum h_{\mathrm{i,j},k,l}(\check{\boldsymbol{\beta}})u_{\mathrm{i}}u_{\mathrm{j}}u_k u_l,
\end{aligned}$$

where $\tilde{\boldsymbol{\beta}}$ and $\check{\boldsymbol{\beta}}$ are two points between $\boldsymbol{\beta}$ and $\hat{\boldsymbol{\beta}}$. In what follows, we also treat $\tilde{\boldsymbol{\beta}}$ and $\check{\boldsymbol{\beta}}$ as functions of $\boldsymbol{u}$, while treating $\hat{\boldsymbol{\beta}}$ as a constant. Let $\phi(\boldsymbol{u}) = \det(-\frac{1}{2\pi}H_n(\hat{\boldsymbol{\beta}}))\mathrm{e}^{\frac{1}{2}\sum h_{\mathrm{i},\mathrm{j}}(\hat{\boldsymbol{\beta}})u_\mathrm{i}u_\mathrm{j}}$ be the centered normal density with covariance matrix $-H_n^{-1}$. Then

$$\begin{aligned}
&\int_{B_{\delta(\boldsymbol{\beta})}} b(\boldsymbol{\beta})\mathrm{e}^{nh(\boldsymbol{\beta})}d\boldsymbol{\beta} = \mathrm{e}^{nh(\hat{\boldsymbol{\beta}})}\int_{B_{\delta(\boldsymbol{\beta})}} \mathrm{e}^{\frac{n}{2}\sum h_{\mathrm{i},\mathrm{j}}(\hat{\boldsymbol{\beta}})(\beta_\mathrm{i}-\hat{\beta}_\mathrm{i})(\beta_\mathrm{j}-\hat{\beta}_\mathrm{j})} b(\boldsymbol{\beta})\mathrm{e}^{g(\boldsymbol{\beta})}d\boldsymbol{\beta} \\
&= \mathrm{e}^{nh(\hat{\boldsymbol{\beta}})}\det(-\frac{n}{2\pi}H_n(\hat{\boldsymbol{\beta}}))^{-\frac{1}{2}}\int_{B_{\sqrt{n}\delta}(0)} \phi(\boldsymbol{u})\left(b(\hat{\boldsymbol{\beta}}) + \sum\frac{1}{\sqrt{n}}b_\mathrm{i}(\hat{\boldsymbol{\beta}})u_\mathrm{i} + \frac{1}{2}\frac{1}{n}\sum b_{\mathrm{i},\mathrm{j}}(\tilde{\boldsymbol{\beta}}(\boldsymbol{u}))u_\mathrm{i}u_\mathrm{j}\right) \\
&\times\left(1 + \frac{1}{6}\frac{1}{\sqrt{n}}\sum h_{\mathrm{i},\mathrm{j},k}(\hat{\boldsymbol{\beta}})u_\mathrm{i}u_\mathrm{j}u_k + \frac{1}{24}\frac{1}{n}\mathrm{e}^{g(\tilde{\boldsymbol{\beta}}(\boldsymbol{u}))}\sum h_{\mathrm{i},\mathrm{j},k,l}(\check{\boldsymbol{\beta}}(\boldsymbol{u}))u_\mathrm{i}u_\mathrm{j}u_ku_l\right)d\boldsymbol{u} \\
&= \mathrm{e}^{nh(\hat{\boldsymbol{\beta}})}\det(-\frac{n}{2\pi}H_n(\hat{\boldsymbol{\beta}}))^{-\frac{1}{2}}\int_{B_{\sqrt{n}\delta}(0)} \phi(\boldsymbol{u})(I_1 + I_2)d\boldsymbol{u},
\end{aligned}$$

where

$$\begin{aligned}
I_1 &= b(\hat{\boldsymbol{\beta}}) + \frac{1}{n^{\frac{1}{2}}}\left(\sum b_\mathrm{i}(\hat{\boldsymbol{\beta}})u_\mathrm{i} + \frac{b(\hat{\boldsymbol{\beta}})}{6}\sum h_{\mathrm{i},\mathrm{j},k}(\hat{\boldsymbol{\beta}})u_\mathrm{i}u_\mathrm{j}u_k\right), \\
I_2 &= \frac{1}{n}\left(b_\mathrm{i}(\hat{\boldsymbol{\beta}})u_\mathrm{i}\frac{1}{6}\sum h_{\mathrm{i},\mathrm{j},k}(\hat{\boldsymbol{\beta}})u_\mathrm{i}u_\mathrm{j}u_k\right) \\
&+ \frac{1}{2}\frac{1}{n}\sum b_{\mathrm{i},\mathrm{j}}(\tilde{\boldsymbol{\beta}}(\boldsymbol{u}))u_\mathrm{i}u_\mathrm{j}\left(1 + \frac{1}{6}\frac{1}{\sqrt{n}}\sum h_{\mathrm{i},\mathrm{j},k}(\hat{\boldsymbol{\beta}})u_\mathrm{i}u_\mathrm{j}u_k + \frac{1}{24}\frac{1}{n}\mathrm{e}^{g(\tilde{\beta}(u))}\sum h_{\mathrm{i},\mathrm{j},k,l}(\check{\boldsymbol{\beta}}(\boldsymbol{u}))u_\mathrm{i}u_\mathrm{j}u_ku_l\right) \\
&+ \frac{1}{24}\frac{1}{n}\mathrm{e}^{g(\tilde{\boldsymbol{\beta}}(\boldsymbol{u}))}\sum h_{\mathrm{i},\mathrm{j},k,l}(\check{\boldsymbol{\beta}}(\boldsymbol{u}))u_\mathrm{i}u_\mathrm{j}u_ku_l\left(b(\hat{\boldsymbol{\beta}}) + \sum\frac{1}{\sqrt{n}}b_\mathrm{i}(\hat{\boldsymbol{\beta}})u_\mathrm{i}\right),
\end{aligned}$$

and we will study the two terms $\int_{B_{\sqrt{n}\delta}(0)}\phi(\boldsymbol{u})I_1d\boldsymbol{u}$ and $\int_{B_{\sqrt{n}\delta}(0)}\phi(\boldsymbol{u})I_2d\boldsymbol{u}$ separately.

To quantify the term $\int_{B_{\sqrt{n}\delta}(0)}\phi(\boldsymbol{u})I_1d\boldsymbol{u}$, we first bound $\int_{\mathbb{R}^{K_n}\setminus B_{\sqrt{n}\delta}(0)}\phi(\boldsymbol{u})I_1d\boldsymbol{u}$. By assumption C.2 and the Markov inequality,

$$\int_{\mathbb{R}^{K_n}\setminus B_{\sqrt{n}\delta}(0)}\phi(\boldsymbol{u})d\boldsymbol{u} = P(\sum_{\mathrm{i}=1}^{K_n}U_\mathrm{i}^2 > n\delta^2) \le \frac{\sum_{\mathrm{i}=1}^{K_n}E(U_\mathrm{i}^2)}{n\delta^2} \le \frac{r_nM + \frac{C(K_n-r_n)}{K_n^2}}{n\delta^2} = O(\frac{r_n}{n}), \quad (2.29)$$

where $(U_1, \dots, U_{K_n})^T$ denotes a multivariate normal random vector following density $\phi(\boldsymbol{u})$. Now we consider the term $\int_{\mathbb{R}^{K_n}\setminus B_{\sqrt{n}\delta}(0)} \frac{1}{\sqrt{n}}\frac{b(\hat{\boldsymbol{\beta}})}{6} h_{\mathrm{i,j},k}(\hat{\boldsymbol{\beta}}) u_{\mathrm{i}} u_{\mathrm{j}} u_k \phi(\boldsymbol{u}) d\boldsymbol{u}$, by Cauchy-Schwarz inequality and assumption C.1, we have

$$
\begin{aligned}
&|\int_{\mathbb{R}^{K_n}\setminus B_{\sqrt{n}\delta}(0)} \frac{1}{\sqrt{n}}\frac{b(\hat{\boldsymbol{\beta}})}{6} h_{\mathrm{i,j},k}(\hat{\boldsymbol{\beta}}) u_{\mathrm{i}} u_{\mathrm{j}} u_k \phi(\boldsymbol{u}) d\boldsymbol{u}| \\
=&|E\left(\frac{1}{\sqrt{n}}\frac{b(\hat{\boldsymbol{\beta}})}{6} h_{\mathrm{i,j},k}(\hat{\boldsymbol{\beta}}) U_{\mathrm{i}} U_{\mathrm{j}} U_k \mathbf{1}\left(\sum_{t=1}^{K_n} U_t^2 > n\delta^2\right)\right)| \\
\leq&\sqrt{\frac{M_1}{n} E(U_{\mathrm{i}}^2 U_{\mathrm{j}}^2 U_k^2) P(\sum_{t=1}^{K_n} U_t^2 > n\delta^2)} \\
=&O(\frac{\sqrt{r_n}}{n})\sqrt{E(U_{\mathrm{i}}^2 U_{\mathrm{j}}^2 U_k^2)}
\end{aligned}
$$

where M_1 is some constant. To bound $E(U_{\mathrm{i}}^2 U_{\mathrm{j}}^2 U_k^2)$,we refer to Theorem 1 of [83], which proved that for $1 \leq \mathrm{i}_1, \dots, \mathrm{i}_6 \leq K_n$,

$$
E(|U_{\mathrm{i}_1} \dots U_{\mathrm{i}_6}|) \leq \sqrt{\sum_{\pi} \prod_{\mathrm{j}=1}^{6} h^{\mathrm{i_j}, \mathrm{i}_{\pi(\mathrm{j})}}},
$$

where the sum is taken over all permutations $\boldsymbol{\pi} = (\boldsymbol{\pi}(1), \dots, \boldsymbol{\pi}(6))$ of set $\{1, \dots, 6\}$ and $h^{\mathrm{i,j}}$ is the $(\mathrm{i,j})$-th element of the covariance matrix H^{-1}. Let $m := m(\mathrm{i}_1, \dots, \mathrm{i}_d) = |\{\mathrm{j} : \mathrm{i_j} \in \boldsymbol{\gamma}^*, \mathrm{j} \in \{1, 2, \dots, d\}\}|$ count the number of indexes belonging to the true connection set. Then, by condition C.2, we have

$$
E(|U_{\mathrm{i}_1} \dots U_{\mathrm{i}_6}|) \leq \sqrt{C_0 M^m (\frac{1}{K_n^2})^{6-m}} = O(\frac{1}{K_n^{6-m}}).
$$

The above inequality implies that $E(U_{\mathrm{i}}^2 U_{\mathrm{j}}^2 U_k^2) = O(\frac{1}{K_n^{6-2m_0}})$, where $m_0 = |\{\mathrm{i}, \mathrm{j}, k\} \cap \boldsymbol{\gamma}^*|$. Thus, we have

$$
\begin{aligned}
&|\int_{\mathbb{R}^{K_n}\setminus B_{\sqrt{n}\delta}(0)} \frac{1}{n^{\frac{1}{2}}} \sum \frac{b(\hat{\beta})}{6} h_{\mathrm{i,j},k}(\hat{\boldsymbol{\beta}}) u_{\mathrm{i}} u_{\mathrm{j}} u_k \phi(\boldsymbol{u}) d\boldsymbol{u}| \\
\leq&\sum_{m_0=0}^{3} \begin{pmatrix} 3 \\ m_0 \end{pmatrix} r_n^{m_0} (K_n - r_n)^{3-m_0} O(\frac{1}{K_n^{3-m_0}} \frac{\sqrt{r_n}}{n}) = O(\frac{r_n^{3.5}}{n}).
\end{aligned}
\tag{2.30}
$$

By similar arguments, we can get the upper bound of the term

$$|\int_{\mathbb{R}^{K_n}\setminus B_{\sqrt{n}\delta}(0)} \sum_{\mathrm{i}} (b_{\mathrm{i}}(\hat{\boldsymbol{\beta}})u_{\mathrm{i}})/\sqrt{n}\phi(\boldsymbol{u})d\boldsymbol{u}|.$$

Thus, we obtain that $|\int_{\mathbb{R}^{K_n}\setminus B_{\sqrt{n}\delta}(0)} \phi(\boldsymbol{u})I_1 d\boldsymbol{u}| \leq O(\frac{r_n^{3.5}}{n})$. Due to the fact that $\int_{\mathbb{R}^{K_n}} \phi(\boldsymbol{u})I_1 d\boldsymbol{u} = b(\hat{\boldsymbol{\beta}})$, we have $\int_{B_{\sqrt{n}\delta}(0)} \phi(\boldsymbol{u})I_1 d\boldsymbol{u} = b(\hat{\boldsymbol{\beta}}) + O(\frac{r_n^{3.5}}{n})$.

Due to assumption C.1 and the fact that $b_{\mathrm{i,j}} \leq M$, within $B_{\sqrt{n}\delta}(0)$, each term in I_2 is trivially bounded by a polynomial of $|\boldsymbol{u}|$, such as,

$$\begin{aligned}
&|\frac{1}{2}\frac{1}{n}\sum b_{\mathrm{i,j}}(\tilde{\boldsymbol{\beta}}(\boldsymbol{u}))u_{\mathrm{i}}u_{\mathrm{j}}\frac{1}{24}\frac{1}{n}\mathrm{e}^{g(\check{\beta}(\boldsymbol{u}))}\sum h_{\mathrm{i,j},k,l}(\check{\boldsymbol{\beta}}(\boldsymbol{u}))u_{\mathrm{i}}u_{\mathrm{j}}u_k u_l| \\
\leq &\frac{1}{48}M^2\mathrm{e}^M\frac{1}{n^2}\sum|u_{\mathrm{i}}u_{\mathrm{j}}|\sum|u_{\mathrm{i}}u_{\mathrm{j}}u_k u_l|.
\end{aligned}$$

Therefore, there exists a constant M_0 such that within $B_{\sqrt{n}\delta}(0)$,

$$|I_2| \leq M_0\left(\frac{1}{n}\sum|u_{\mathrm{i}}u_{\mathrm{j}}| + \frac{1}{n}\sum|u_{\mathrm{i}}u_{\mathrm{j}}u_k u_l| + \frac{1}{n^{\frac{3}{2}}}\sum|u_{\mathrm{i}}u_{\mathrm{j}}u_k u_l u_s| + \frac{1}{n^2}\sum|u_{\mathrm{i}}u_{\mathrm{j}}u_k u_l u_s u_t|\right) := I_3,$$

Then we have

$$|\int_{B_{\sqrt{n}\delta}(0)} \phi(\boldsymbol{u})I_2 d\boldsymbol{u}| \leq \int_{B_{\sqrt{n}\delta}(0)} \phi(\boldsymbol{u})I_3 d\boldsymbol{u} \leq \int_{\mathbb{R}^{K_n}} \phi(\boldsymbol{u})I_3 d\boldsymbol{u},$$

By the same arguments as used to bound $E(U_{\mathrm{i}}^2U_{\mathrm{j}}^2U_k^2)$, we can show that

$$\int_{\mathbb{R}^{K_n}} \phi(\boldsymbol{u})\frac{1}{n^2}\sum|u_{\mathrm{i}}u_{\mathrm{j}}u_k u_l u_s u_t|d\boldsymbol{u} = O(\frac{r_n^6}{n^2})$$

holds. The rest terms in $\int_{\mathbb{R}^{K_n}} \phi(\boldsymbol{u})I_3 d\boldsymbol{u}$ can be bounded by the same manner, and in the end we have $|\int_{B_{\sqrt{n}\delta}(0)} \phi(\boldsymbol{u})I_2 d\boldsymbol{u}| \leq \int_{\mathbb{R}^{K_n}} \phi(\boldsymbol{u})I_3 d\boldsymbol{u} = O(\frac{r_n^4}{n})$.

Then $\int_{B_{\delta(\boldsymbol{\beta})}} b(\boldsymbol{\beta})\mathrm{e}^{nh(\boldsymbol{\beta})}d\boldsymbol{\beta} = \mathrm{e}^{nh(\hat{\boldsymbol{\beta}})}\det(-\frac{n}{2\pi}H_n(\hat{\boldsymbol{\beta}}))^{-\frac{1}{2}}(b(\hat{\boldsymbol{\beta}}) + O(\frac{r_n^4}{n}))$ holds. Combining it with condition C.3 and the boundedness of b, we get

$$\int b(\boldsymbol{\beta})\mathrm{e}^{nh_n(\boldsymbol{\beta})}d\boldsymbol{\beta} = \mathrm{e}^{nh(\hat{\boldsymbol{\beta}})}\det(-\frac{n}{2\pi}H_n(\hat{\boldsymbol{\beta}}))^{-\frac{1}{2}}(b(\hat{\boldsymbol{\beta}}) + O(\frac{r_n^4}{n})).$$

With similar calculations, we can get $\int e^{nh_n(\beta)} d\boldsymbol{\beta} = e^{nh(\hat{\beta})} \det(-\frac{n}{2\pi} H_n(\hat{\boldsymbol{\beta}}))^{-\frac{1}{2}}(1 + O(\frac{r_n^4}{n}))$. Therefore,

$$\frac{\int b(\boldsymbol{\beta}) e^{nh_n(\beta)} d\boldsymbol{\beta}}{\int e^{nh_n(\beta)} d\boldsymbol{\beta}} = \frac{e^{nh(\hat{\beta})} \det(-\frac{n}{2\pi} H_n(\hat{\boldsymbol{\beta}}))^{-\frac{1}{2}}(b(\hat{\boldsymbol{\beta}}) + O(\frac{r_n^4}{n}))}{e^{nh(\hat{\beta})} \det(-\frac{n}{2\pi} H_n(\hat{\boldsymbol{\beta}}))^{-\frac{1}{2}}(1 + O(\frac{r_n^4}{n}))} = b(\hat{\boldsymbol{\beta}}) + O(\frac{r_n^4}{n}).$$

□

Proof of Lemma 2.3.1

Proof. Consider the following prior setting: (i) $\lambda_n = K_n^{-(1+\tau')H_n}$, (ii) $\log \frac{1}{\sigma_{0,n}} = H_n \log(K_n)$, (iii) $\sigma_{1,n} = 1$, and (iv) $\epsilon_n \geq \sqrt{\frac{(\frac{16}{\delta}+16) r_n H_n \log K_n}{n}}$. Note that this setting satisfies all conditions of previous theorems. Recall that the marginal posterior inclusion probability is given by

$$q_{\mathrm{i}} = \int \sum_{\gamma} e_{\mathrm{i}|\nu(\gamma,\beta)} \pi(\boldsymbol{\gamma}|\boldsymbol{\beta}, D_n) \pi(\boldsymbol{\beta}|D_n) d\boldsymbol{\beta} := \int \pi(\nu(\gamma_{\mathrm{i}}) = 1|\beta_{\mathrm{i}}) \pi(\boldsymbol{\beta}|D_n) d\boldsymbol{\beta}.$$

For any false connection $c_{\mathrm{i}} \notin \boldsymbol{\gamma}_*$, we have $e_{\mathrm{i}|\nu(\gamma^*,\beta^*)} = 0$ and

$$|q_{\mathrm{i}}| = \int \sum_{\gamma} |e_{\mathrm{i}|\nu(\gamma,\beta)} - e_{\mathrm{i}|\nu(\gamma^*,\beta^*)}| \pi(\boldsymbol{\gamma}|\boldsymbol{\beta}, D_n) \pi(\boldsymbol{\beta}|D_n) d\boldsymbol{\beta} \leq \hat{\pi}(4\epsilon_n) + \rho(4\epsilon_n).$$

A straightforward calculation shows

$$\begin{aligned} \pi(\nu(\gamma_{\mathrm{i}}) = 1|\beta_{\mathrm{i}}) &= \frac{1}{1 + \frac{1-\lambda_n}{\lambda_n} \frac{\sigma_{1,n}}{\sigma_{0,n}} \exp\left\{-\frac{1}{2}(\frac{1}{\sigma_{0,n}^2} - \frac{1}{\sigma_{1,n}^2})\beta_{\mathrm{i}}^2\right\}} \\ &= \frac{1}{1 + \exp\left\{-\frac{1}{2}(K_n^{2H_n} - 1)(\beta_{\mathrm{i}}^2 - \frac{(4+2\gamma')H_n \log(K_n) + 2\log(1-\lambda_n)}{K_n^{2H_n}-1})\right\}}. \end{aligned}$$

Let $M_n = \frac{(4+2\tau')H_n \log(K_n) + 2\log(1-\lambda_n)}{K_n^{2H_n}-1}$. Then, by Markov inequality,

$$P(\beta_{\mathrm{i}}^2 > M_n|D_n) = P\left(\pi(\nu(\gamma_{\mathrm{i}}) = 1|\beta_{\mathrm{i}}) > 1/2|D_n\right) \leq 2|q_{\mathrm{i}}| \leq 2(\hat{\pi}(4\epsilon_n) + \rho(4\epsilon_n)).$$

Therefore,

$$\begin{aligned}
E(\beta_{\mathrm{i}}^2|D_n) &\leq M_n + \int_{\beta_{\mathrm{i}}^2 > M_n} \beta_{\mathrm{i}}^2 \pi(\beta|D_n) d\beta \\
&\leq M_n + \int_{M_n < \beta_{\mathrm{i}}^2 < M_n^{-\frac{2}{\delta}}} \beta_{\mathrm{i}}^2 \pi(\beta|D_n) d\beta + \int_{\beta_{\mathrm{i}}^2 > M_n^{-\frac{2}{\delta}}} M_n |\beta_{\mathrm{i}}|^{2+\delta} \pi(\beta|D_n) d\beta \\
&\leq M_n + M_n^{-\frac{2}{\delta}} P(\beta_{\mathrm{i}}^2 > M_n) + C M_n.
\end{aligned}$$

Since $\frac{1}{K_n^{2H_n}} \prec M_n \prec \frac{1}{K_n^{2H_n - 1}}$, $\epsilon_n \geq \sqrt{\frac{(\frac{16}{\delta}+16) r_n H_n \log K_n}{n}}$, we have $M_n^{-\frac{2}{\delta}} \mathrm{e}^{-n\epsilon_n^2/4} \prec \frac{1}{K_n^{2H_n-1}}$. Thus

$$P^* \left\{ E(\beta_{\mathrm{i}}^2|D_n) \prec \frac{1}{K_n^{2H_n-1}} \right\} \geq P^* \left(\hat{\pi}(4\epsilon_n) < 2\mathrm{e}^{-n\epsilon_n^2/4} \right) \geq 1 - 2\mathrm{e}^{-n\epsilon_n^2/4}.$$

□

Verification of the Bounded Gradient Condition in Theorem 2.3.2

This section shows that with an appropriate choice of prior hyperparameters, the first and second order derivatives of $\pi(\nu(\gamma_{\mathrm{i}}) = 1|\beta_{\mathrm{i}})$ (i.e. the function $b(\boldsymbol{\beta})$ in Theorem 2.3.2) and the third and fourth order derivatives of $\log \pi(\boldsymbol{\beta})$ are all bounded with a high probability. Therefore, the assumption C.1 in Theorem 2.3.2 is reasonable.

Under the same setting of the prior as that used in the proof of Lemma 2.3.1, we can show that the derivative of $\pi(\nu(\gamma_{\mathrm{i}}) = 1|\beta_{\mathrm{i}})$ is bounded with a high probability. For notational simplicity, we suppress the subscript i in what follows and let

$$f(\beta) = \pi(\nu(\gamma) = 1|\beta) = \frac{1}{1 + \frac{1-\lambda_n}{\lambda_n} \frac{\sigma_{1,n}}{\sigma_{0,n}} \exp\left\{ -\frac{1}{2}(\frac{1}{\sigma_{0,n}^2} - \frac{1}{\sigma_{1,n}^2})\beta^2 \right\}} := \frac{1}{1 + C_2 \exp\{-C_1 \beta^2\}},$$

where $C_1 = \frac{1}{2}(\frac{1}{\sigma_{0,n}^2} - \frac{1}{\sigma_{1,n}^2}) = \frac{1}{2}(K_n^{2H_n} - 1)$ and $C_2 = \frac{1-\lambda_n}{\lambda_n} \frac{\sigma_{1,n}}{\sigma_{0,n}} = (1-\lambda_n) K_n^{(2+\tau')H_n}$. Then we have $C_1 \beta^2 = \log(C_2) + \log(f(\beta)) - \log(1 - f(\beta))$. With some algebra, we can show that

$$\left|\frac{df(\beta)}{d\beta}\right| = 2\sqrt{C_1} f(\beta)(1 - f(\beta)) \sqrt{\log(C_2) + \log(f(\beta)) - \log(1 - f(\beta))},$$

$$\frac{d^2 f(\beta)}{d\beta^2} = f(\beta)(1 - f(\beta))(2C_1 + 4C_1(\log(C_2) + \log(f(\beta)) - \log(1 - f(\beta)))(1 - 2f(\beta))).$$

By Markov inequality and Theorem 2.3.1, for the false connections,

$$P\left\{f(\beta)(1-f(\beta)) > \frac{1}{C_1^2}|D_n\right\} \leq P(f(\beta) > \frac{1}{C_1^2}|D_n) \leq C_1^2 E(f(\beta)|D_n) \leq C_1^2(\hat{\pi}(4\epsilon_n) + \rho(4\epsilon_n))$$

holds, and for the true connections,

$$\begin{aligned} P\left\{f(\beta)(1-f(\beta)) > \frac{1}{C_1^2}|D_n\right\} &\leq P(1-f(\beta) > \frac{1}{C_1^2}|D_n) \\ &\leq C_1^2 E(1-f(\beta)|D_n) \leq C_1^2(\hat{\pi}(4\epsilon_n) + \rho(4\epsilon_n)) \end{aligned}$$

holds. Under the setting of Lemma 2.3.1, by Theorem 2.2.1, it is easy to see that $C_1^2(\hat{\pi}(4\epsilon_n) + \rho(4\epsilon_n)) \to 0$ as $n \to \infty$, and thus $(f(\beta)(1-f(\beta)) < \frac{1}{C_1^2}$ with high probability. Note that $\frac{\log(C_2)}{C_1} \to 0$ and $|f(\beta)\log(f(\beta))| < \frac{1}{e}$. Thus, when $(f(\beta)(1-f(\beta)) < \frac{1}{C_1^2}$ holds,

$$|\frac{df(\beta)}{d\beta}| \leq \sqrt{\frac{\log(C_2)}{C_1} + (f(\beta)(1-f(\beta))\log(f(\beta)) - (f(\beta)(1-f(\beta))\log(1-f(\beta))},$$

is bounded. Similarly we can show that $\frac{d^2 f(\beta)}{d\beta^2}$ is also bounded. In conclusion, $\frac{df(\beta)}{d\beta}$ and $\frac{d^2 f(\beta)}{d\beta^2}$ is bounded with probability $P\left\{f(\beta)(1-f(\beta)) \leq \frac{1}{C_1^2}|D_n\right\}$ which tends to 1 as $n \to \infty$.

Recall that $\pi(\beta) = \frac{1-\lambda_n}{\sqrt{2\pi}\sigma_{0,n}}\exp\{-\frac{\beta^2}{2\sigma_{0,n}^2}\} + \frac{\lambda_n}{\sqrt{2\pi}\sigma_{1,n}}\exp\{-\frac{\beta^2}{2\sigma_{1,n}^2}\}$. With some algebra, we can show

$$\begin{aligned} \frac{d^3\log(\pi(\beta))}{d\beta^3} &= 2(\frac{1}{\sigma_{0,n}^2} - \frac{1}{\sigma_{1,n}^2})\frac{df(\beta)}{d\beta} + (\frac{\beta}{\sigma_{0,n}^2} - \frac{\beta}{\sigma_{1,n}^2})\frac{d^2 f(\beta)}{d\beta^2} \\ &= 4C_1\frac{df(\beta)}{d\beta} + 2\sqrt{C_1}\frac{d^2 f(\beta)}{d\beta^2}\sqrt{\log(C_2) + \log(f(\beta)) - \log(1-f(\beta))}. \end{aligned}$$

With similar arguments to that used for $\frac{d^2 f(\beta)}{d\beta^2}$ and $\frac{d^2 f(\beta)}{d\beta^2}$, we can make the term $f(\beta)(1-f(\beta))$ very small with a probability tending to 1. Therefore, $\frac{d^3\log(\pi(\beta))}{d\beta^3}$ is bounded with a probability tending to 1. Similarly we can bound $\frac{d^4\log(\pi(\beta))}{d\beta^4}$ with a high probability.

Approximation of Bayesian Evidence

In Algorithm 1, each sparse model is evaluated by its Bayesian evidence:

$$Evidence = \det(-\frac{n}{2\pi} H_n(\boldsymbol{\beta}_{\gamma}))^{-\frac{1}{2}} e^{nh_n(\boldsymbol{\beta}_{\gamma})},$$

where $H_n(\boldsymbol{\beta}_{\gamma}) = \frac{\partial^2 h_n(\boldsymbol{\beta}_{\gamma})}{\partial \boldsymbol{\beta}_{\gamma} \partial^T \boldsymbol{\beta}_{\gamma}}$ is the Hessian matrix, $\boldsymbol{\beta}_{\gamma}$ denotes the vector of connection weights selected by the model $\boldsymbol{\gamma}$, i.e. $H_n(\boldsymbol{\beta}_{\gamma})$ is a $|\boldsymbol{\gamma}| \times |\boldsymbol{\gamma}|$ matrix, and the prior $\pi(\boldsymbol{\beta}_{\gamma})$ in $h_n(\boldsymbol{\beta})$ is only for the connection weights selected by the model $\boldsymbol{\gamma}$. Therefore,

$$\log(Evidence) = nh_n(\boldsymbol{\beta}_{\gamma}) - \frac{1}{2}|\boldsymbol{\gamma}|\log(n) + \frac{1}{2}|\boldsymbol{\gamma}|\log(2\pi) - \frac{1}{2}\log(\det(-H_n(\boldsymbol{\beta}_{\gamma}))), \qquad (2.31)$$

and $-H_n(\boldsymbol{\beta}_{\gamma}) = -\frac{1}{n}\sum_{i=1}^{n} \frac{\partial^2 \log(p(y_i, \boldsymbol{x}_i|\boldsymbol{\beta}_{\gamma}))}{\partial \boldsymbol{\beta}_{\gamma} \partial^T \boldsymbol{\beta}_{\gamma}} - \frac{1}{n}\sum_{i=1}^{n} \frac{\partial^2 \log(\pi(\boldsymbol{\beta}_{\gamma}))}{\partial \boldsymbol{\beta}_{\gamma} \partial^T \boldsymbol{\beta}_{\gamma}}$. For the selected connection weights, the prior $\pi(\boldsymbol{\beta}_{\gamma})$ behaves like $N(0, \sigma_{1,n}^2)$, and then $-\frac{\partial^2 \log(\pi(\boldsymbol{\beta}_{\gamma}))}{\partial \boldsymbol{\beta}_{\gamma} \partial^T \boldsymbol{\beta}_{\gamma}}$ is a diagonal matrix with the diagonal elements approximately equal to $\frac{1}{\sigma_{1,n}^2}$ and $\frac{1}{n\sigma_{1,n}^2} \to 0$.

If $(y_i, \boldsymbol{x}_i)$'s are viewed as i.i.d samples drawn from $p(y, \boldsymbol{x}|\boldsymbol{\beta}_{\gamma}))$, then $-\frac{1}{n}\sum_{i=1}^{n} \frac{\partial^2 \log(p(y_i, \boldsymbol{x}_i|\boldsymbol{\beta}_{\gamma}))}{\partial \boldsymbol{\beta}_{\gamma} \partial^T \boldsymbol{\beta}_{\gamma}}$ will converge to the Fisher information matrix $I(\boldsymbol{\beta}_{\gamma}) = E(-\frac{\partial^2 \log(p(y, \boldsymbol{x}|\boldsymbol{\beta}_{\gamma}))}{\partial \boldsymbol{\beta}_{\gamma} \partial^T \boldsymbol{\beta}_{\gamma}})$. If we further assume that $I(\boldsymbol{\beta}_{\gamma})$ has bounded eigenvalues, i.e. $C_{min} \leq \lambda_{min}(I(\boldsymbol{\beta}_{\gamma})) \leq \lambda_{max}(I(\boldsymbol{\beta}_{\gamma})) \leq C_{max}$ for some constants C_{min} and C_{max}, then $\log(\det(-H_n(\boldsymbol{\beta}_{\gamma}))) \asymp |\boldsymbol{\gamma}|$. This further implies

$$\frac{1}{2}|\boldsymbol{\gamma}|\log(2\pi) - \frac{1}{2}\log(\det(-H_n(\boldsymbol{\beta}_{\gamma}))) \prec |\boldsymbol{\gamma}|\log(n).$$

By keeping only the dominating terms in (2.31), we have

$$\log(Evidence) \approx nh_n(\boldsymbol{\beta}_{\gamma}) - \frac{1}{2}|\boldsymbol{\gamma}|\log(n) = -\frac{1}{2}BIC + \log(\pi(\boldsymbol{\beta}_{\gamma})).$$

Since we are comparing a few low-dimensional models (with the model size $|\boldsymbol{\gamma}| \prec n$), it is intuitive to further ignore the prior term $\log(\pi(\boldsymbol{\beta}_{\gamma}))$. As a result, we can elicit the low-dimensional sparse neural networks by BIC.

2.9.3 Proofs of Asymptotic Normality

Proof of Theorem 2.4.1

Proof. We first define the equivalent class of neural network parameters. Given a parameter vector $\boldsymbol{\beta}$ and the corresponding structure parameter vector $\boldsymbol{\gamma}$, its equivalent class is given by

$$Q_E(\boldsymbol{\beta}, \boldsymbol{\gamma}) = \{(\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\gamma}}) : \nu_g(\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\gamma}}) = (\boldsymbol{\beta}, \boldsymbol{\gamma}), \mu(\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\gamma}}, \boldsymbol{x}) = \mu(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{x}), \forall \boldsymbol{x}\},$$

where $\nu_g(\cdot)$ denotes a generic mapping that contains only the transformations of node permutation and weight sign flipping. Specifically, we define

$$Q_E^* = Q_E(\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*),$$

which represents the equivalent class of *true DNN model*.

Let $B_{\delta_n}(\boldsymbol{\beta}^*) = \{\boldsymbol{\beta} : |\boldsymbol{\beta}_\mathrm{i} - \boldsymbol{\beta}^*_\mathrm{i}| < \delta_n, \forall \mathrm{i} \in \boldsymbol{\gamma}^*, |\boldsymbol{\beta}_\mathrm{i} - \boldsymbol{\beta}^*_\mathrm{i}| < 2\sigma_{0,n} \log(\frac{\sigma_{1,n}}{\lambda_n \sigma_{0,n}}), \forall \mathrm{i} \notin \boldsymbol{\gamma}^*\}$. By assumption D.1, $\boldsymbol{\beta}^*$ is generic (i.e. $Q_E(\boldsymbol{\beta}^*)$ contains only reparameterizations of weight sign-flipping or node permutations as defined in [51] and [52]) and $\min_{\mathrm{i} \in \boldsymbol{\gamma}^*} |\boldsymbol{\beta}^*_\mathrm{i}| - \delta_n > (C-1)\delta_n > \delta_n$, then for any $\boldsymbol{\beta}^{*(1)}, \boldsymbol{\beta}^{*(2)} \in Q_E^*$, $B_{\delta_n}(\boldsymbol{\beta}^{*(1)}) \cap B_{\delta_n}(\boldsymbol{\beta}^{*(2)}) = \emptyset$, and thus $\{\boldsymbol{\beta} : \tilde{\nu}(\boldsymbol{\beta}) \in B_{\delta_n}(\boldsymbol{\beta}^*)\} = \cup_{\boldsymbol{\beta} \in Q_E^*} B_{\delta_n}(\boldsymbol{\beta})$. In what follows, we will first show $\boldsymbol{\pi}(\cup_{\boldsymbol{\beta} \in Q_E^*} B_{\delta_n}(\boldsymbol{\beta}) \mid D_n) \to 1$ as $n \to \infty$, which means the most posterior mass falls in the neighbourhood of true parameter.

Remark on the notation: $\tilde{\nu}(\cdot)$ is similar to $\nu(\cdot)$ defined in Section 2.3 They both map the set $Q_E(\boldsymbol{\beta}, \boldsymbol{\gamma})$ to a unique network. The difference between them is that $\|\nu(\boldsymbol{\beta}) - \boldsymbol{\beta}^*\|_\infty$ may be arbitrary, but $\|\tilde{\nu}(\boldsymbol{\beta}) - \boldsymbol{\beta}^*\|_\infty$ is minimized. In other words, $\nu(\boldsymbol{\beta}, \boldsymbol{\gamma})$ is to find an arbitrary network in $Q_E(\boldsymbol{\beta}, \boldsymbol{\gamma})$ as the representative of the equivalent class, while $\tilde{\nu}(\boldsymbol{\beta}, \boldsymbol{\gamma})$ is to find a representative in $Q_E(\boldsymbol{\beta}, \boldsymbol{\gamma})$ such that the distance between $\boldsymbol{\beta}^*$ and the representative is minimized. In what follows, we will use $\tilde{\nu}(\boldsymbol{\beta})$ and $\tilde{\nu}(\boldsymbol{\gamma})$ to denote the connection weight and network structure of $\tilde{\nu}(\boldsymbol{\beta}, \boldsymbol{\gamma})$, respectively. With a slight abuse of notation, we will use $\tilde{\nu}(\boldsymbol{\beta})_\mathrm{i}$ to denote the ith component of $\tilde{\nu}(\boldsymbol{\beta})$, and use $\tilde{\nu}(\boldsymbol{\gamma})_\mathrm{i}$ to denote the ith component of $\tilde{\nu}(\boldsymbol{\gamma})$.

Recall that the marginal posterior inclusion probability is given by

$$q_{\mathrm{i}} = \int \sum_{\boldsymbol{\gamma}} e_{\mathrm{i}|\tilde{\nu}(\boldsymbol{\beta},\boldsymbol{\gamma})} \pi(\boldsymbol{\gamma}|\boldsymbol{\beta}, D_n)\pi(\boldsymbol{\beta}|D_n)d\boldsymbol{\beta} = \int \pi(\tilde{\nu}(\boldsymbol{\gamma})_{\mathrm{i}} = 1|\boldsymbol{\beta})\pi(\boldsymbol{\beta}|D_n)d\boldsymbol{\beta}.$$

For the mixture Gaussian prior,

$$\pi(\boldsymbol{\gamma}_{\mathrm{i}} = 1|\boldsymbol{\beta}) = \frac{1}{1 + \frac{\sigma_{1,n}(1-\lambda_n)}{\sigma_{0,n}\lambda_n} e^{-(\frac{1}{2\sigma_{0,n}^2} - \frac{1}{2\sigma_{1,n}^2})\beta_{\mathrm{i}}^2}},$$

which increases with respect to $|\boldsymbol{\beta}_{\mathrm{i}}|$. In particular, if $|\boldsymbol{\beta}_{\mathrm{i}}| > 2\sigma_{0,n}\log(\frac{\sigma_{1,n}}{\lambda_n\sigma_{0,n}})$, then $\pi(\boldsymbol{\gamma}_{\mathrm{i}} = 1|\boldsymbol{\beta}) > \frac{1}{2}$.

For the mixture Gaussian prior,

$$\begin{aligned}
&\pi(\boldsymbol{\beta} \notin \cup_{\beta\in Q_E^*} B_{\delta_n}(\boldsymbol{\beta}) \mid D_n) \\
\leq &\pi(\exists \mathrm{i} \notin \boldsymbol{\gamma}^*, |\tilde{\nu}(\boldsymbol{\beta})_{\mathrm{i}}| > 2\sigma_{0,n}\log(\frac{\sigma_{1,n}}{\lambda_n\sigma_{0,n}}) \mid D_n) + \pi(\exists \mathrm{i} \in \boldsymbol{\gamma}^*, |\tilde{\nu}(\boldsymbol{\beta})_{\mathrm{i}} - \boldsymbol{\beta}_{\mathrm{i}}^*| > \delta_n \mid D_n).
\end{aligned}$$

For the first term, note that for a given $\mathrm{i} \notin \boldsymbol{\gamma}^*$,

$$\begin{aligned}
\pi(|\tilde{\nu}(\boldsymbol{\beta})_{\mathrm{i}}| > 2\sigma_{0,n}\log(\frac{\sigma_{1,n}}{\lambda_n\sigma_{0,n}}) \mid D_n) \leq &\pi(\pi(\tilde{\nu}(\boldsymbol{\gamma})_{\mathrm{i}} = 1|\boldsymbol{\beta}) > \frac{1}{2} \mid D_n) \\
\leq &2\int \pi(\tilde{\nu}(\boldsymbol{\gamma})_{\mathrm{i}} = 1|\boldsymbol{\beta})\pi(\boldsymbol{\beta}|D_n)d\boldsymbol{\beta} \\
\leq &2\rho(\epsilon_n) + 2\pi(d(p_{\boldsymbol{\beta}}, p_{\mu^*}) \geq \epsilon_n \mid D_n) \to 0.
\end{aligned}$$

Then we have

$$\begin{aligned}
&\pi(\exists \mathrm{i} \notin \boldsymbol{\gamma}^*, |\tilde{\nu}(\boldsymbol{\beta})_{\mathrm{i}}| > 2\sigma_{0,n}\log(\frac{\sigma_{1,n}}{\lambda_n\sigma_{0,n}}) \mid D_n) \\
= &\pi(\max_{\mathrm{i}\notin\boldsymbol{\gamma}^*} |\tilde{\nu}(\boldsymbol{\beta})_{\mathrm{i}}| > 2\sigma_{0,n}\log(\frac{\sigma_{1,n}}{\lambda_n\sigma_{0,n}}) \mid D_n) \\
\leq &\pi(\max_{\mathrm{i}\notin\boldsymbol{\gamma}^*} \pi(\tilde{\nu}(\boldsymbol{\gamma})_{\mathrm{i}} = 1|\boldsymbol{\beta}) > \frac{1}{2} \mid D_n) \\
\leq &\sum_{\mathrm{i}\notin\boldsymbol{\gamma}^*} \pi(\pi(\tilde{\nu}(\boldsymbol{\gamma})_{\mathrm{i}} = 1|\boldsymbol{\beta}) > \frac{1}{2} \mid D_n) \\
\leq &2K_n\rho(\epsilon_n) + 2K_n\pi(d(p_{\boldsymbol{\beta}}, p_{\mu^*}) \geq \epsilon_n \mid D_n) \to 0.
\end{aligned}$$

For the second term, by condition D.1 and Theorem 2.3.1,

$$\begin{aligned}
&\pi(\exists \mathrm{i} \in \boldsymbol{\gamma}^*, |\tilde{\nu}(\boldsymbol{\beta})_\mathrm{i} - \boldsymbol{\beta}^*_\mathrm{i}| > \delta_n \mid D_n) = \pi(\max_{\mathrm{i}\in\boldsymbol{\gamma}^*} |\tilde{\nu}(\boldsymbol{\beta})_\mathrm{i} - \boldsymbol{\beta}^*_\mathrm{i}| > \delta_n \mid D_n) \\
=&\pi(\max_{\mathrm{i}\in\boldsymbol{\gamma}^*} |\tilde{\nu}(\boldsymbol{\beta})_\mathrm{i} - \boldsymbol{\beta}^*_\mathrm{i}| > \delta_n, d(p_{\boldsymbol{\beta}}, p_{\mu^*}) \le \epsilon_n \mid D_n) \\
&+ \pi(\max_{\mathrm{i}\in\boldsymbol{\gamma}^*} |\tilde{\nu}(\boldsymbol{\beta})_\mathrm{i} - \boldsymbol{\beta}^*_\mathrm{i}| > \delta_n, d(p_{\boldsymbol{\beta}}, p_{\mu^*}) \ge \epsilon_n \mid D_n) \\
\le&\pi(A(\epsilon_n, \delta_n) \mid D_n) + \pi(d(p_{\boldsymbol{\beta}}, p_{\mu^*}) \ge \epsilon_n \mid D_n) \to 0.
\end{aligned}$$

Summarizing the above two terms, we have $\pi(\cup_{\boldsymbol{\beta}\in Q^*_E} B_{\delta_n}(\boldsymbol{\beta}) \mid D_n) \to 1$.

Let $Q_n = |Q^*_E|$ be the number of equivalent *true DNN model*. By the generic assumption of $\boldsymbol{\beta}^*$, for any $\boldsymbol{\beta}^{*(1)}, \boldsymbol{\beta}^{*(2)} \in Q^*_E$, $B_{\delta_n}(\boldsymbol{\beta}^{*(1)}) \cap B_{\delta_n}(\boldsymbol{\beta}^{*(2)}) = \emptyset$. Then in $B_{\delta_n}(\boldsymbol{\beta}^*)$, the posterior density of $\tilde{\nu}(\boldsymbol{\beta})$ is Q_n times the posterior density of $\boldsymbol{\beta}$. Then for a function $f(\cdot)$ of $\tilde{\nu}(\boldsymbol{\beta})$, by changing variable,

$$\int_{\tilde{\nu}(\boldsymbol{\beta})\in B_{\delta_n}(\boldsymbol{\beta}^*)} f(\tilde{\nu}(\boldsymbol{\beta}))\pi(\tilde{\nu}(\boldsymbol{\beta})|D_n)d\tilde{\nu}(\boldsymbol{\beta}) = Q_n \int_{B_{\delta_n}(\boldsymbol{\beta}^*)} f(\boldsymbol{\beta})\pi(\boldsymbol{\beta}|D_n)d\boldsymbol{\beta}.$$

Thus, we only need to consider the integration on $B_{\delta_n}(\boldsymbol{\beta}^*)$. Define

$$\hat{\boldsymbol{\beta}}_\mathrm{i} = \begin{cases} \boldsymbol{\beta}^*_\mathrm{i} - \sum_{\mathrm{j}\in\boldsymbol{\gamma}^*} h^{\mathrm{i},\mathrm{j}}(\boldsymbol{\beta}^*)h_\mathrm{j}(\boldsymbol{\beta}^*), & \forall \mathrm{i} \in \boldsymbol{\gamma}^*, \\ 0, & \forall \mathrm{i} \notin \boldsymbol{\gamma}^*. \end{cases}$$

We will first prove that for any real vector $\boldsymbol{t}$,

$$\begin{aligned}
E(\mathrm{e}^{\sqrt{n}\boldsymbol{t}^T(\tilde{\nu}(\beta)-\hat{\beta})} \mid D_n, B_{\delta_n}(\boldsymbol{\beta}^*)) :=& \frac{\int_{B_{\delta_n}(\boldsymbol{\beta}^*)} \mathrm{e}^{\sqrt{n}\boldsymbol{t}^T(\tilde{\nu}(\beta)-\hat{\beta})}\pi(\tilde{\nu}(\boldsymbol{\beta})|D_n)d\tilde{\nu}(\boldsymbol{\beta})}{\int_{B_{\delta_n}(\boldsymbol{\beta}^*)} \pi(\tilde{\nu}(\boldsymbol{\beta})|D_n)d\tilde{\nu}(\boldsymbol{\beta})} \\
=& \frac{\int_{B_{\delta_n}(\boldsymbol{\beta}^*)} \mathrm{e}^{\sqrt{n}\boldsymbol{t}^T(\beta-\hat{\beta})}\mathrm{e}^{nl_n(\beta)}\pi(\boldsymbol{\beta})d\boldsymbol{\beta}}{\int_{B_{\delta_n}(\boldsymbol{\beta}^*)} \mathrm{e}^{nl_n(\beta)}\pi(\boldsymbol{\beta})d\boldsymbol{\beta}} \qquad (2.32) \\
=& \mathrm{e}^{\frac{1}{2}\boldsymbol{t}^T\boldsymbol{V}\boldsymbol{t}+o_{P^*}(1)}.
\end{aligned}$$

For any $\boldsymbol{\beta} \in B_{\delta_n}(\boldsymbol{\beta}^*)$, we have

$$|\sqrt{n}(\boldsymbol{t}^T(\boldsymbol{\beta} - \boldsymbol{\beta}_{\boldsymbol{\gamma}^*}))| \le \sqrt{n}K_n||\boldsymbol{t}||_\infty 2\sigma_{0,n}\log(\frac{\sigma_{1,n}}{\lambda_n\sigma_{0,n}}) = o(1),$$

$$|n(l_n(\boldsymbol{\beta}) - l_n(\boldsymbol{\beta}_{\boldsymbol{\gamma}^*}))| = |n\sum_{\mathrm{i}\notin\boldsymbol{\gamma}^*} \boldsymbol{\beta}_{\mathrm{i}}(h_{\mathrm{i}}(\tilde{\boldsymbol{\beta}}))| \leq nK_nM2\sigma_{0,n}\log(\frac{\sigma_{1,n}}{\lambda_n\sigma_{0,n}}) = o(1).$$

Then, we have

$$\begin{aligned}\sqrt{n}\boldsymbol{t}^T(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}) =&\sqrt{n}\boldsymbol{t}^T(\boldsymbol{\beta} - \boldsymbol{\beta}_{\boldsymbol{\gamma}^*} + \boldsymbol{\beta}_{\boldsymbol{\gamma}^*} - \boldsymbol{\beta}^*) + \sqrt{n}\sum_{\mathrm{i,j}\in\boldsymbol{\gamma}^*} h^{\mathrm{i,j}}(\boldsymbol{\beta}^*)\boldsymbol{t}_{\mathrm{j}}h_{\mathrm{i}}(\boldsymbol{\beta}^*)) \\ =&o(1) + \sqrt{n}\sum_{\mathrm{i}\in\boldsymbol{\gamma}^*}(\boldsymbol{\beta}_{\mathrm{i}} - \boldsymbol{\beta}^*_{\mathrm{i}})\boldsymbol{t}_{\mathrm{i}} + \sqrt{n}\sum_{\mathrm{i,j}\in\boldsymbol{\gamma}^*} h^{\mathrm{i,j}}(\boldsymbol{\beta}^*)\boldsymbol{t}_{\mathrm{j}}h_{\mathrm{i}}(\boldsymbol{\beta}^*),\end{aligned} \tag{2.33}$$

$$\begin{aligned}nl_n(\boldsymbol{\beta}) - nl_n(\boldsymbol{\beta}^*) =&n(l_n(\boldsymbol{\beta}) - l_n(\boldsymbol{\beta}_{\boldsymbol{\gamma}^*}) + l_n(\boldsymbol{\beta}_{\boldsymbol{\gamma}^*}) - nl_n(\boldsymbol{\beta}^*)) \\ =&o(1) + n\sum_{\mathrm{i}\in\boldsymbol{\gamma}^*}(\boldsymbol{\beta}_{\mathrm{i}} - \boldsymbol{\beta}^*_{\mathrm{i}})h_{\mathrm{i}}(\boldsymbol{\beta}^*) + \frac{n}{2}\sum_{\mathrm{i,j}\in\boldsymbol{\gamma}^*} h_{\mathrm{i,j}}(\boldsymbol{\beta}^*)(\boldsymbol{\beta}_{\mathrm{i}} - \boldsymbol{\beta}^*_{\mathrm{i}})(\boldsymbol{\beta}_{\mathrm{j}} - \boldsymbol{\beta}^*_{\mathrm{j}}) \\ &+ \frac{n}{6}\sum_{\mathrm{i,j},k\in\boldsymbol{\gamma}^*} h_{\mathrm{i,j},k}(\check{\boldsymbol{\beta}})(\boldsymbol{\beta}_{\mathrm{i}} - \boldsymbol{\beta}^*_{\mathrm{i}})(\boldsymbol{\beta}_{\mathrm{j}} - \boldsymbol{\beta}^*_{\mathrm{j}})(\boldsymbol{\beta}_k - \boldsymbol{\beta}^*_k),\end{aligned} \tag{2.34}$$

where $\check{\boldsymbol{\beta}}$ is a point between $\boldsymbol{\beta}_{\boldsymbol{\gamma}^*}$ and $\boldsymbol{\beta}^*$. Note that for $\boldsymbol{\beta} \in B_{\delta_n}(\boldsymbol{\beta}^*)$, $|\boldsymbol{\beta}_{\mathrm{i}} - \boldsymbol{\beta}^*_{\mathrm{i}}| \leq \delta_n \lesssim \frac{1}{\sqrt[3]{n}r_n}$, we have $\frac{n}{6}\sum_{\mathrm{i,j},k\in\boldsymbol{\gamma}^*} h_{\mathrm{i,j},k}(\check{\boldsymbol{\beta}})(\boldsymbol{\beta}_{\mathrm{i}} - \boldsymbol{\beta}^*_{\mathrm{i}})(\boldsymbol{\beta}_{\mathrm{j}} - \boldsymbol{\beta}^*_{\mathrm{j}})(\boldsymbol{\beta}_k - \boldsymbol{\beta}^*_k) = o(1)$.

Let $\boldsymbol{\beta}^{(t)}$ be network parameters satisfying $\boldsymbol{\beta}^{(t)}_{\mathrm{i}} = \boldsymbol{\beta}_{\mathrm{i}} + \frac{1}{\sqrt{n}}\sum_{\mathrm{j}\in\boldsymbol{\gamma}^*} h^{\mathrm{i,j}}(\boldsymbol{\beta}^*)\boldsymbol{t}_{\mathrm{j}}, \forall \mathrm{i} \in \boldsymbol{\gamma}^*$ and $\boldsymbol{\beta}^{(t)}_{\mathrm{i}} = \boldsymbol{\beta}_{\mathrm{i}}, \forall \mathrm{i} \notin \boldsymbol{\gamma}^*$. Note that $\frac{1}{\sqrt{n}}\sum_{\mathrm{j}\in\boldsymbol{\gamma}^*} h^{\mathrm{i,j}}(\boldsymbol{\beta}^*)\boldsymbol{t}_{\mathrm{j}} \leq \frac{r_n||\boldsymbol{t}||_\infty M}{\sqrt{n}} \lesssim \delta_n$, for large enough n, $|\boldsymbol{\beta}^{(t)}_{\mathrm{i}}| < 2\delta_n$ $\forall \mathrm{i} \in \boldsymbol{\gamma}^*$. Thus, we have

$$\begin{aligned}nl_n(\boldsymbol{\beta}^{(t)}) - nl_n(\boldsymbol{\beta}^*) =&n(l_n(\boldsymbol{\beta}^{(t)}) - l_n(\boldsymbol{\beta}^{(t)}_{\boldsymbol{\gamma}^*}) + l_n(\boldsymbol{\beta}^{(t)}_{\boldsymbol{\gamma}^*}) - nl_n(\boldsymbol{\beta}^*)) \\ =&o(1) + n\sum_{\mathrm{i}\in\boldsymbol{\gamma}^*}(\boldsymbol{\beta}^{(t)}_{\mathrm{i}} - \boldsymbol{\beta}^*_{\mathrm{i}})h_{\mathrm{i}}(\boldsymbol{\beta}^*) + \frac{n}{2}\sum_{\mathrm{i,j}\in\boldsymbol{\gamma}^*} h_{\mathrm{i,j}}(\boldsymbol{\beta}^*)(\boldsymbol{\beta}^{(t)}_{\mathrm{i}} - \boldsymbol{\beta}^*_{\mathrm{i}})(\boldsymbol{\beta}^{(t)}_{\mathrm{j}} - \boldsymbol{\beta}^*_{\mathrm{j}}) \\ =&o(1) + n\sum_{\mathrm{i}\in\boldsymbol{\gamma}^*}(\boldsymbol{\beta}_{\mathrm{i}} - \boldsymbol{\beta}^*_{\mathrm{i}})h_{\mathrm{i}}(\boldsymbol{\beta}^*) + \frac{n}{2}\sum_{\mathrm{i,j}\in\boldsymbol{\gamma}^*} h_{\mathrm{i,j}}(\boldsymbol{\beta}^*)(\boldsymbol{\beta}_{\mathrm{i}} - \boldsymbol{\beta}^*_{\mathrm{i}})(\boldsymbol{\beta}_{\mathrm{j}} - \boldsymbol{\beta}^*_{\mathrm{j}}) \\ &+ \sqrt{n}\sum_{\mathrm{i,j}\in\boldsymbol{\gamma}^*} h^{\mathrm{i,j}}(\boldsymbol{\beta}^*)\boldsymbol{t}_{\mathrm{j}}h_{\mathrm{i}}(\boldsymbol{\beta}^*) + \sqrt{n}\sum_{\mathrm{i}\in\boldsymbol{\gamma}^*}(\boldsymbol{\beta}_{\mathrm{i}} - \boldsymbol{\beta}^*_{\mathrm{i}})\boldsymbol{t}_{\mathrm{i}} + \frac{1}{2}\sum_{\mathrm{i,j}\in\boldsymbol{\gamma}^*} h^{\mathrm{i,j}}(\boldsymbol{\beta}^*)\boldsymbol{t}_{\mathrm{i}}\boldsymbol{t}_{\mathrm{j}} \\ =&o(1) + \sqrt{n}\boldsymbol{t}^T(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}) + nl_n(\boldsymbol{\beta}) - nl_n(\boldsymbol{\beta}^*) + \frac{1}{2}\sum_{\mathrm{i,j}\in\boldsymbol{\gamma}^*} h^{\mathrm{i,j}}(\boldsymbol{\beta}^*)\boldsymbol{t}_{\mathrm{i}}\boldsymbol{t}_{\mathrm{j}},\end{aligned} \tag{2.35}$$

where the last equality is derived by replacing appropriate terms by $\sqrt{n}\boldsymbol{t}^T(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}})$ and $nl_n(\boldsymbol{\beta}) - nl_n(\boldsymbol{\beta}^*)$ based on (2.33) and (2.34), respectively; and the third equality is derived based on the following calculation:

$$
\begin{aligned}
&\frac{n}{2}\sum_{\mathrm{i,j}\in\boldsymbol{\gamma}^*} h_{\mathrm{i,j}}(\boldsymbol{\beta}^*)(\boldsymbol{\beta}_{\mathrm{i}}^{(t)} - \boldsymbol{\beta}_{\mathrm{i}}^*)(\boldsymbol{\beta}_{\mathrm{j}}^{(t)} - \boldsymbol{\beta}_{\mathrm{j}}^*) \\
=&\frac{n}{2}\sum_{\mathrm{i,j}\in\boldsymbol{\gamma}^*} h_{\mathrm{i,j}}(\boldsymbol{\beta}^*)(\boldsymbol{\beta}_{\mathrm{i}} - \boldsymbol{\beta}_{\mathrm{i}}^* + \frac{1}{\sqrt{n}}\sum_{k\in\boldsymbol{\gamma}^*} h^{\mathrm{i},k}(\boldsymbol{\beta}^*)\boldsymbol{t}_k)(\boldsymbol{\beta}_{\mathrm{j}} - \boldsymbol{\beta}_{\mathrm{j}}^* + \frac{1}{\sqrt{n}}\sum_{k\in\boldsymbol{\gamma}^*} h^{\mathrm{j},k}(\boldsymbol{\beta}^*)\boldsymbol{t}_k) \\
=&\frac{n}{2}\sum_{\mathrm{i,j}\in\boldsymbol{\gamma}^*} h_{\mathrm{i,j}}(\boldsymbol{\beta}^*)(\boldsymbol{\beta}_{\mathrm{i}} - \boldsymbol{\beta}_{\mathrm{i}}^*)(\boldsymbol{\beta}_{\mathrm{j}} - \boldsymbol{\beta}_{\mathrm{j}}^*) + 2\times\frac{n}{2}\sum_{\mathrm{i,j}\in\boldsymbol{\gamma}^*} h_{\mathrm{i,j}}(\boldsymbol{\beta}^*)\frac{1}{\sqrt{n}}\sum_{k\in\boldsymbol{\gamma}^*} h^{\mathrm{i},k}(\boldsymbol{\beta}^*)\boldsymbol{t}_k(\boldsymbol{\beta}_{\mathrm{j}} - \boldsymbol{\beta}_{\mathrm{j}}^*) \qquad (2.36)\\
&+\frac{n}{2}\sum_{\mathrm{i,j}\in\boldsymbol{\gamma}^*} h_{\mathrm{i,j}}(\boldsymbol{\beta}^*)(\frac{1}{\sqrt{n}}\sum_{k\in\boldsymbol{\gamma}^*} h^{\mathrm{i},k}(\boldsymbol{\beta}^*)\boldsymbol{t}_k)(\frac{1}{\sqrt{n}}\sum_{k\in\boldsymbol{\gamma}^*} h^{\mathrm{j},k}(\boldsymbol{\beta}^*)\boldsymbol{t}_k) \\
=&\frac{n}{2}\sum_{\mathrm{i,j}\in\boldsymbol{\gamma}^*} h_{\mathrm{i,j}}(\boldsymbol{\beta}^*)(\boldsymbol{\beta}_{\mathrm{i}} - \boldsymbol{\beta}_{\mathrm{i}}^*)(\boldsymbol{\beta}_{\mathrm{j}} - \boldsymbol{\beta}_{\mathrm{j}}^*) + \sqrt{n}\sum_{\mathrm{i}\in\boldsymbol{\gamma}^*}(\boldsymbol{\beta}_{\mathrm{i}} - \boldsymbol{\beta}_{\mathrm{i}}^*)\boldsymbol{t}_{\mathrm{i}} + \frac{1}{2}\sum_{\mathrm{i,j}\in\boldsymbol{\gamma}^*} h^{\mathrm{i,j}}(\boldsymbol{\beta}^*)\boldsymbol{t}_{\mathrm{i}}\boldsymbol{t}_{\mathrm{j}},
\end{aligned}
$$

where the second and third terms in the last equality are derived based on the relation $\sum_{\mathrm{i}\in\boldsymbol{\gamma}^*} h_{\mathrm{i,j}}(\boldsymbol{\beta}^*)h^{\mathrm{i},k}(\boldsymbol{\beta}^*) = \delta_{\mathrm{j},k}$, where $\delta_{\mathrm{j},k} = 1$ if $\mathrm{j} = k$, $\delta_{\mathrm{j},k} = 0$ if $\mathrm{j} \neq k$.

By rearranging the terms in (2.35), we have

$$
\begin{aligned}
&\int_{B_{\delta_n}(\boldsymbol{\beta}^*)} \exp\{\sqrt{n}\boldsymbol{t}^T(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}) + nl_n(\boldsymbol{\beta})\}\boldsymbol{\pi}(\boldsymbol{\beta})d\boldsymbol{\beta} \\
&= \exp\left\{-\frac{1}{2}\sum_{\mathrm{i,j}\in\boldsymbol{\gamma}^*} h^{\mathrm{i,j}}(\boldsymbol{\beta}^*)\boldsymbol{t}_{\mathrm{i}}\boldsymbol{t}_{\mathrm{j}} + o(1)\right\}\int_{B_{\delta_n}(\boldsymbol{\beta}^*)} \mathrm{e}^{nl_n(\boldsymbol{\beta}^{(t)})}\boldsymbol{\pi}(\boldsymbol{\beta})d\boldsymbol{\beta}.
\end{aligned}
$$

For $\boldsymbol{\beta} \in B_{\delta_n}(\boldsymbol{\beta}^*), \mathrm{i} \in \boldsymbol{\gamma}^*$, by Assumption D.1, there exists a constant $C > 2$ such that

$$
\begin{aligned}
|\boldsymbol{\beta}_{\mathrm{i}}^{(t)}| \geq |\boldsymbol{\beta}_{\mathrm{i}}| - \frac{r_n||\boldsymbol{t}||_\infty M}{\sqrt{n}} &\geq |\boldsymbol{\beta}_{\mathrm{i}}^*| - 2\delta_n \geq (C-2)\delta_n \gtrsim \frac{r_n}{\sqrt{n}} \\
&\gtrsim \sqrt{\left(\frac{1}{2\sigma_{0,n}^2} - \frac{1}{2\sigma_{1,n}^2}\right)^{-1}\log\left(\frac{r_n(1-\lambda_n)\sigma_{1,n}}{\sigma_{0,n}\lambda_n}\right)}.
\end{aligned}
$$

Then we have

$$
\frac{\sigma_{1,n}(1-\lambda_n)}{\sigma_{0,n}\lambda_n}\mathrm{e}^{-(\frac{1}{2\sigma_{0,n}^2} - \frac{1}{2\sigma_{1,n}^2})(\boldsymbol{\beta}_{\mathrm{i}}^{(t)})^2} \lesssim \frac{1}{r_n}.
$$

It is easy to see that the above formula also holds if we replace $\boldsymbol{\beta}_{\mathrm{i}}^{(t)}$ by $\boldsymbol{\beta}_{\mathrm{i}}$. Note that the mixture Gaussian prior of $\boldsymbol{\beta}_{\mathrm{i}}$ can be written as

$$\pi(\boldsymbol{\beta}_{\mathrm{i}}) = \frac{\lambda_n}{\sqrt{2\pi}\sigma_{1,n}} \mathrm{e}^{-\frac{\boldsymbol{\beta}_{\mathrm{i}}^2}{2\sigma_{1,n}^2}} \left(1 + \frac{\sigma_{1,n}(1-\lambda_n)}{\sigma_{0,n}\lambda_n} \mathrm{e}^{-(\frac{1}{2\sigma_{0,n}^2} - \frac{1}{2\sigma_{1,n}^2})\boldsymbol{\beta}_{\mathrm{i}}^2}\right).$$

Since $|\boldsymbol{\beta}_{\mathrm{i}} - \boldsymbol{\beta}_{\mathrm{i}}^{(t)}| \lesssim \delta_n \lesssim \frac{1}{\sqrt[3]{n} r_n}$, $|\boldsymbol{\beta}_{\mathrm{i}} + \boldsymbol{\beta}_{\mathrm{i}}^{(t)}| < 2E_n + 3\delta_n \lesssim E_n$, and $\frac{1}{\sigma_{1,n}^2} \lesssim \frac{H_n \log(n) + \log(\bar{L})}{E_n^2}$, we have

$$\frac{r_n}{\sigma_{1,n}^2}(\boldsymbol{\beta}_{\mathrm{i}} - \boldsymbol{\beta}_{\mathrm{i}}^{(t)})(\boldsymbol{\beta}_{\mathrm{i}} + \boldsymbol{\beta}_{\mathrm{i}}^{(t)}) = \frac{H_n \log(n) + \log(\bar{L})}{n^{C_1 + 1/3}} = o(1),$$

by the condition $C_1 > 2/3$ and $H_n \log(n) + \log(\bar{L}) \prec n^{1-\epsilon}$. Thus, $\frac{\pi(\boldsymbol{\beta})}{\pi(\boldsymbol{\beta}^{(t)})} = \prod_{\mathrm{i} \in \boldsymbol{\gamma}^*} \frac{\pi(\boldsymbol{\beta}_{\mathrm{i}})}{\pi(\boldsymbol{\beta}_{\mathrm{i}}^{(t)})} = 1 + o(1)$, and

$$\begin{aligned}\int_{B_{\delta_n}(\boldsymbol{\beta}^*)} \mathrm{e}^{n l_n(\boldsymbol{\beta}^{(t)})} \pi(\boldsymbol{\beta}) d\boldsymbol{\beta} =& (1 + o(1)) \int_{\boldsymbol{\beta}^{(t)} \in B_{\delta_n}(\boldsymbol{\beta}^*)} \mathrm{e}^{n l_n(\boldsymbol{\beta}^{(t)})} \pi(\boldsymbol{\beta}^{(t)}) d\boldsymbol{\beta}^{(t)} \\ =& (1 + o(1)) C_N \pi(\boldsymbol{\beta}^{(t)} \in B_{\delta_n}(\boldsymbol{\beta}^*) \mid D_n),\end{aligned} \tag{2.37}$$

where C_N is the normalizing constant of the posterior. Note that $||\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}||_\infty \lesssim \delta_n$, we have $\pi(\boldsymbol{\beta}^{(t)} \in B_{\delta_n}(\boldsymbol{\beta}^*) \mid D_n) \to \pi(\boldsymbol{\beta} \in B_{\delta_n}(\boldsymbol{\beta}^*) \mid D_n)$. Moreover, since $-\frac{1}{2}\sum_{\mathrm{i},\mathrm{j} \in \boldsymbol{\gamma}^*} h^{\mathrm{i},\mathrm{j}}(\boldsymbol{\beta}^*) \boldsymbol{t}_{\mathrm{i}} \boldsymbol{t}_{\mathrm{j}} \to \frac{1}{2}\boldsymbol{t}^T \boldsymbol{V} \boldsymbol{t}$, we have

$$E(\mathrm{e}^{\sqrt{n}\boldsymbol{t}^T(\tilde{\nu}(\boldsymbol{\beta}) - \hat{\boldsymbol{\beta}})} \mid D_n, B_{\delta_n}(\boldsymbol{\beta}^*)) = \frac{\int_{B_{\delta_n}(\boldsymbol{\beta}^*)} \mathrm{e}^{\sqrt{n}\boldsymbol{t}^T(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}})} \mathrm{e}^{n h_n(\boldsymbol{\beta})} \pi(\boldsymbol{\beta}) d\boldsymbol{\beta}}{\int_{B_{\delta_n}(\boldsymbol{\beta}^*)} \mathrm{e}^{n h_n(\boldsymbol{\beta})} \pi(\boldsymbol{\beta}) d\boldsymbol{\beta}} = \mathrm{e}^{\frac{\boldsymbol{t}^T \boldsymbol{V} \boldsymbol{t}}{2} + o_{P^*}(1)}.$$

Combining the above result with the fact that $\pi(\tilde{\nu}(\boldsymbol{\beta}) \in B_{\delta_n}(\boldsymbol{\beta}^*) \mid D_n) \to 1$, by section 1 of [84], we have

$$\pi[\sqrt{n}(\tilde{\nu}(\boldsymbol{\beta}) - \hat{\boldsymbol{\beta}}) \mid D_n] \rightsquigarrow N(0, \boldsymbol{V}).$$

We will then show that $\hat{\boldsymbol{\beta}}$ will converge to $\boldsymbol{\beta}^*$, then essentially we can replace $\hat{\boldsymbol{\beta}}$ by $\boldsymbol{\beta}^*$ in the above result. Let $\boldsymbol{\Theta}_{\boldsymbol{\gamma}^*} = \{\boldsymbol{\beta} : \boldsymbol{\beta}_{\mathrm{i}} = 0, \forall \mathrm{i} \notin \boldsymbol{\gamma}^*\}$ be the parameter space given the model $\boldsymbol{\gamma}^*$, and let $\hat{\boldsymbol{\beta}}_{\boldsymbol{\gamma}^*}$ be the maximum likelihood estimator given the model $\boldsymbol{\gamma}^*$, i.e.

$$\hat{\boldsymbol{\beta}}_{\boldsymbol{\gamma}^*} = \arg\max_{\boldsymbol{\beta} \in \boldsymbol{\Theta}_{\boldsymbol{\gamma}^*}} l_n(\boldsymbol{\beta}).$$

Given condition D.3 and by Theorem 2.1 of [53], we have $||\hat{\boldsymbol{\beta}}_{\boldsymbol{\gamma}^*} - \boldsymbol{\beta}^*|| = O(\sqrt{\frac{r_n}{n}}) = o(1)$. Note that $h_{\mathrm{i}}(\hat{\boldsymbol{\beta}}_{\boldsymbol{\gamma}^*}) = 0$ as $\hat{\boldsymbol{\beta}}_{\boldsymbol{\gamma}^*}$ is maximum likelihood estimator. Then for any $\mathrm{i} \in \boldsymbol{\gamma}^*$, $|h_{\mathrm{i}}(\boldsymbol{\beta}^*)| = |h_{\mathrm{i}}(\hat{\boldsymbol{\beta}}_{\boldsymbol{\gamma}^*}) - h_{\mathrm{i}}(\boldsymbol{\beta}^*)| = |\sum_{\mathrm{j}\in\boldsymbol{\gamma}^*} h_{\mathrm{ij}}(\tilde{\boldsymbol{\beta}})((\hat{\boldsymbol{\beta}}_{\boldsymbol{\gamma}^*})_{\mathrm{j}} - \boldsymbol{\beta}^*_{\mathrm{j}})| \le M||\hat{\boldsymbol{\beta}}_{\boldsymbol{\gamma}^*} - \boldsymbol{\beta}^*||_1 = O(\sqrt{\frac{r_n}{n}})$. Then for any $\mathrm{i}, \mathrm{j} \in \boldsymbol{\gamma}^*$, we have $\sum_{\mathrm{j}\in\boldsymbol{\gamma}^*} h^{\mathrm{i},\mathrm{j}}(\boldsymbol{\beta}^*)h_{\mathrm{j}}(\boldsymbol{\beta}^*) = O(\sqrt{\frac{r_n^3}{n}}) = o(1)$. By the definition of $\hat{\boldsymbol{\beta}}$, we have $\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^* = o(1)$. Therefore, we have

$$\boldsymbol{\pi}[\sqrt{n}(\tilde{\nu}(\boldsymbol{\beta}) - \boldsymbol{\beta}^*) \mid D_n] \rightsquigarrow N(0, \boldsymbol{V}).$$

Proof of Theorem 2.4.2

Proof. The proof of Theorem 2.4.2 can be done using the same strategy as that used in proving Theorem 2.4.1. Here we provide a simpler proof using the result of Theorem 2.4.1. The notations we used in this proof are the same as in the proof of Theorem 2.4.1. In the proof of Theorem 2.4.1, we have shown that $\boldsymbol{\pi}(\tilde{\nu}(\boldsymbol{\beta}) \in B_{\delta_n}(\boldsymbol{\beta}^*) \mid D_n) \to 1$. Note that $\mu(\boldsymbol{\beta}, \boldsymbol{x}_0) = \mu(\tilde{\nu}(\boldsymbol{\beta}), \boldsymbol{x}_0)$. We only need to consider $\boldsymbol{\beta} \in B_{\delta_n}(\boldsymbol{\beta}^*)$. For $\boldsymbol{\beta} \in B_{\delta_n}(\boldsymbol{\beta}^*)$, we have

$$\begin{aligned}
&\sqrt{n}(\mu(\boldsymbol{\beta}, \boldsymbol{x}_0) - \mu(\boldsymbol{\beta}^*, \boldsymbol{x}_0)) \\
=&\sqrt{n}(\mu(\boldsymbol{\beta}, \boldsymbol{x}_0) - \mu(\boldsymbol{\beta}_{\boldsymbol{\gamma}^*}, \boldsymbol{x}_0) + \mu(\tilde{\nu}(\boldsymbol{\beta}_{\boldsymbol{\gamma}^*}), \boldsymbol{x}_0) - \mu(\boldsymbol{\beta}^*, \boldsymbol{x}_0)).
\end{aligned}$$

Since $\boldsymbol{\beta} \in B_{\delta_n}(\boldsymbol{\beta}^*)$, for $\mathrm{i} \notin \boldsymbol{\gamma}^*$, $|\boldsymbol{\beta}_{\mathrm{i}}| < 2\sigma_{0,n} \log(\frac{\sigma_{1,n}}{\lambda_n \sigma_{0,n}})$; and for $\mathrm{i} \in \boldsymbol{\gamma}^*$, $|\tilde{\nu}(\boldsymbol{\beta})_{\mathrm{i}} - \boldsymbol{\beta}^*_{\mathrm{i}}| < \delta \lesssim \frac{1}{\sqrt[3]{n} r_n}$. Therefore,

$$|\sqrt{n}\mu(\boldsymbol{\beta}, \boldsymbol{x}_0) - \mu(\boldsymbol{\beta}_{\boldsymbol{\gamma}^*}, \boldsymbol{x}_0))| = |\sqrt{n}\sum_{\mathrm{i}\notin\boldsymbol{\gamma}^*} \boldsymbol{\beta}_{\mathrm{i}}(\mu_{\mathrm{i}}(\tilde{\boldsymbol{\beta}}, \boldsymbol{x}_0))| \le \sqrt{n} K_n M 2\sigma_{0,n} \log(\frac{\sigma_{1,n}}{\lambda_n \sigma_{0,n}}) = o(1),$$

where $\mu_{\mathrm{i}}(\boldsymbol{\beta}, \boldsymbol{x}_0)$ denotes the first derivative of $\mu(\boldsymbol{\beta}, \boldsymbol{x}_0)$ with respect to the ith component of $\boldsymbol{\beta}$, and $\tilde{\boldsymbol{\beta}}$ denotes a point between $\boldsymbol{\beta}$ and $\boldsymbol{\beta}_{\boldsymbol{\gamma}^*}$. Further,

$$\begin{aligned}
&\mu(\tilde{\nu}(\boldsymbol{\beta}_{\boldsymbol{\gamma}^*}), \boldsymbol{x}_0) - \mu(\boldsymbol{\beta}^*, \boldsymbol{x}_0) \\
=&\sqrt{n}\sum_{\mathrm{i}\in\boldsymbol{\gamma}^*}(\tilde{\nu}(\boldsymbol{\beta})_{\mathrm{i}} - \boldsymbol{\beta}^*_{\mathrm{i}})\mu_{\mathrm{i}}(\boldsymbol{\beta}^*, \boldsymbol{x}_0) + \sqrt{n}\sum_{\mathrm{i}\in\boldsymbol{\gamma}^*}\sum_{\mathrm{j}\in\boldsymbol{\gamma}^*}(\tilde{\nu}(\boldsymbol{\beta})_{\mathrm{i}} - \boldsymbol{\beta}^*_{\mathrm{i}})\mu_{\mathrm{i},\mathrm{j}}(\check{\boldsymbol{\beta}}, \boldsymbol{x}_0)(\tilde{\nu}(\boldsymbol{\beta})_{\mathrm{j}} - \boldsymbol{\beta}^*_{\mathrm{j}}) \\
=&\sqrt{n}\sum_{\mathrm{i}\in\boldsymbol{\gamma}^*}((\tilde{\nu}(\boldsymbol{\beta})_{\mathrm{i}} - \boldsymbol{\beta}^*_{\mathrm{i}})\mu_{\mathrm{i}}(\boldsymbol{\beta}^*, \boldsymbol{x}_0) + o(1),
\end{aligned}$$

where $\mu_{\mathrm{i,j}}(\boldsymbol{\beta}, \boldsymbol{x}_0)$ denotes the second derivative of $\mu(\boldsymbol{\beta}, \boldsymbol{x}_0)$ with respect to the ith and jth components of $\boldsymbol{\beta}$ and $\check{\boldsymbol{\beta}}$ is a point between $\tilde{\nu}(\boldsymbol{\beta})$ and $\boldsymbol{\beta}^*$. Summarizing the above two equations, we have

$$\sqrt{n}\mu(\boldsymbol{\beta}, \boldsymbol{x}_0) - \mu(\boldsymbol{\beta}^*, \boldsymbol{x}_0)) = \sqrt{n}\sum_{\mathrm{i}\in\gamma^*}((\tilde{\nu}(\boldsymbol{\beta}_{\mathrm{i}}) - \boldsymbol{\beta}^*_{\mathrm{i}})\mu_{\mathrm{i}}(\boldsymbol{\beta}^*, \boldsymbol{x}_0) + o(1).$$

By Theorem 2.4.1, $\boldsymbol{\pi}[\sqrt{n}(\tilde{\nu}(\boldsymbol{\beta}) - \boldsymbol{\beta}^*) \mid D_n] \rightsquigarrow N(0, \boldsymbol{V})$, where $\boldsymbol{V} = (v_{\mathrm{ij}})$, and $v_{\mathrm{i,j}} = E(h^{\mathrm{i,j}}(\boldsymbol{\beta}^*))$ if $\mathrm{i,j} \in \boldsymbol{\gamma}^*$ and 0 otherwise. Then we have $\boldsymbol{\pi}[\sqrt{n}(\mu(\boldsymbol{\beta}, \boldsymbol{x}_0) - \mu(\boldsymbol{\beta}^*, \boldsymbol{x}_0)) \mid D_n] \rightsquigarrow N(0, \Sigma)$, where $\Sigma = \nabla_{\boldsymbol{\gamma}^*}\mu(\boldsymbol{\beta}^*, \boldsymbol{x}_0)^T H^{-1} \nabla_{\boldsymbol{\gamma}^*}\mu(\boldsymbol{\beta}^*, \boldsymbol{x}_0)$ and $H = E(-\nabla^2_{\boldsymbol{\gamma}^*} l_n(\boldsymbol{\beta}^*))$.

2.9.4 Proofs on Generalization Bounds

Proof of Theorem 2.5.1

Proof. Consider the set B_n defined in (2.27). By the argument used in the proof of Theorem 2.9.1, there exists a class of $\tilde{B}_n = \{\boldsymbol{\beta}^{(l)} : 1 \le l < L\}$ for some $L < \exp\{cn\epsilon_n^2\}$ with a constant c such that for any $\boldsymbol{\beta} \in B_n$, there exists some $\boldsymbol{\beta}^{(l)}$ satisfying $|\mu(\boldsymbol{\beta}, \boldsymbol{x}) - \mu(\boldsymbol{\beta}^{(l)}, \boldsymbol{x})| \le c'\epsilon_n$.

Let $\tilde{\boldsymbol{\pi}}$ be the truncated distribution of $\boldsymbol{\pi}(\boldsymbol{\beta}|D_n)$ on B_n, and let $\check{\boldsymbol{\pi}}$ be a discrete distribution on $\tilde{B}_n$ defined as $\check{\boldsymbol{\pi}}(\boldsymbol{\beta}^{(l)}) = \tilde{\boldsymbol{\pi}}(B_l)$, where $B_l = \{\boldsymbol{\beta} \in B_n : \|\mu(\boldsymbol{\beta}, \boldsymbol{x}) - \mu(\boldsymbol{\beta}^{(l)}, \boldsymbol{x})\|_\infty < \min_{\mathrm{j}\neq l} \|\mu(\boldsymbol{\beta}, \boldsymbol{x}) - \mu(\boldsymbol{\beta}^{(\mathrm{j})}, \boldsymbol{x})\|_\infty\}$ by defining the norm $\|f\|_\infty = \max_{\{\boldsymbol{x}\in\Omega\}} f(\boldsymbol{x})$. Note that for any $\boldsymbol{\beta} \in B_l$, $\|\mu(\boldsymbol{\beta}, \boldsymbol{x}) - \mu(\boldsymbol{\beta}^{(l)}, \boldsymbol{x})\| \le c'\epsilon_n$, thus,

$$l_0(\boldsymbol{\beta}, \boldsymbol{x}, y) \le l_{c'\epsilon_n/2}(\boldsymbol{\beta}^{(l)}, \boldsymbol{x}, y) \le l_{c'\epsilon_n}(\boldsymbol{\beta}, \boldsymbol{x}, y). \tag{2.38}$$

The above inequality implies that

$$\begin{aligned} &\int E_{\boldsymbol{x},y} l_0(\boldsymbol{\beta}, \boldsymbol{x}, y) d\tilde{\boldsymbol{\pi}} \le \int E_{\boldsymbol{x},y} l_{c'\epsilon_n/2}(\boldsymbol{\beta}^{(l)}, \boldsymbol{x}, y) d\check{\boldsymbol{\pi}}, \\ &\int \frac{1}{n}\sum_{\mathrm{i}=1}^n l_{c'\epsilon_n/2}(\boldsymbol{\beta}^{(l)}, \boldsymbol{x}^{(\mathrm{i})}, y^{(\mathrm{i})}) d\check{\boldsymbol{\pi}} \le \int \frac{1}{n}\sum_{\mathrm{i}=1}^n l_{c'\epsilon_n}(\boldsymbol{\beta}, \boldsymbol{x}^{(\mathrm{i})}, y^{(\mathrm{i})}) d\tilde{\boldsymbol{\pi}}. \end{aligned} \tag{2.39}$$

Let P be a uniform prior on $\tilde{B}_n$, by Theorem 2.3.1, with probability $1-\delta$,

$$\begin{aligned}\int E_{\boldsymbol{x},y} l_\nu(\boldsymbol{\beta}^{(l)}, \boldsymbol{x}, y) d\check{\boldsymbol{\pi}} &\leq \int \frac{1}{n}\sum_{\mathrm{i}=1}^{n} l_\nu(\boldsymbol{\beta}^{(l)}, \boldsymbol{x}^{(\mathrm{i})}, y^{(\mathrm{i})}) d\check{\boldsymbol{\pi}} + \sqrt{\frac{d_0(\check{\boldsymbol{\pi}}, P) + \log\frac{2\sqrt{n}}{\delta}}{2n}} \\ &\leq \int \frac{1}{n}\sum_{\mathrm{i}=1}^{n} l_\nu(\boldsymbol{\beta}^{(l)}, \boldsymbol{x}^{(\mathrm{i})}, y^{(\mathrm{i})}) d\check{\boldsymbol{\pi}} + \sqrt{\frac{cn\epsilon_n^2 + \log\frac{2\sqrt{n}}{\delta}}{2n}},\end{aligned} \tag{2.40}$$

for any $\nu \geq 0$ and $\delta > 0$, where the second inequality is due to the fact that $d_0(\mathcal{L}, P) \leq \log L$ for any discrete distribution $\mathcal{L}$ over $\{\boldsymbol{\beta}^{(l)}\}_{l=1}^{L}$.

Combining inequalities (2.39) and (2.40), we have that, with probability $1-\delta$,

$$\int E_{\boldsymbol{x},y} l_0(\boldsymbol{\beta}, \boldsymbol{x}, y) d\tilde{\boldsymbol{\pi}} \leq \sqrt{\frac{cn\epsilon_n^2 + \log\frac{2\sqrt{n}}{\delta}}{2n}} + \int \frac{1}{n}\sum_{\mathrm{i}=1}^{n} l_{c'\epsilon_n}(\boldsymbol{\beta}, \boldsymbol{x}^{(\mathrm{i})}, y^{(\mathrm{i})}) d\tilde{\boldsymbol{\pi}}. \tag{2.41}$$

Due to the boundedness of l_ν, we have

$$\begin{aligned}&\int E_{\boldsymbol{x},y} l_0(\boldsymbol{\beta}, \boldsymbol{x}, y) d\boldsymbol{\pi}(\boldsymbol{\beta}|D_n) \leq \int E_{\boldsymbol{x},y} l_0(\boldsymbol{\beta}, \boldsymbol{x}, y) d\tilde{\boldsymbol{\pi}} + \boldsymbol{\pi}(B_n^c|D_n), \\ &\int \frac{1}{n}\sum_{\mathrm{i}=1}^{n} l_{c'\epsilon_n}(\boldsymbol{\beta}, \boldsymbol{x}^{(\mathrm{i})}, y^{(\mathrm{i})}) d\tilde{\boldsymbol{\pi}} \leq \frac{1}{1-\boldsymbol{\pi}(B_n^c|D_n)} \int \frac{1}{n}\sum_{\mathrm{i}=1}^{n} l_{c'\epsilon_n}(\boldsymbol{\beta}, \boldsymbol{x}^{(\mathrm{i})}, y^{(\mathrm{i})}) d\boldsymbol{\pi}(\boldsymbol{\beta}|D_n).\end{aligned} \tag{2.42}$$

Note that the result of Theorem A.1 implies that, with probability at least $1-\exp\{-c''n\epsilon_n^2\}$, $\boldsymbol{\pi}(B_n^c|D_n) \leq 2\exp\{-c''n\epsilon_n^2\}$. Therefore, with probability greater than $1-\delta-\exp\{-c''n\epsilon_n^2\}$,

$$\begin{aligned}\int E_{\boldsymbol{x},y} l_0(\boldsymbol{\beta}, \boldsymbol{x}, y) d\boldsymbol{\pi}(\boldsymbol{\beta}|D_n) \leq & \frac{1}{1-2\exp\{-c''n\epsilon_n^2\}} \int \frac{1}{n}\sum_{\mathrm{i}=1}^{n} l_{c'\epsilon_n}(\boldsymbol{\beta}, \boldsymbol{x}^{(\mathrm{i})}, y^{(\mathrm{i})}) d\boldsymbol{\pi}(\boldsymbol{\beta}|D_n) \\ & + \sqrt{\frac{cn\epsilon_n^2 + \log\frac{2\sqrt{n}}{\delta}}{2n}} + 2\exp\{-c''n\epsilon_n^2\}.\end{aligned}$$

Thus, the result holds if we choose $\delta = \exp\{-c'''n\epsilon_n^2\}$ for some c'''. $\square$

Proof of Theorem 2.5.2

Proof. To prove the theorem, we first introduce a lemma on generalization error of finite classifiers, which can be easily derived based on Hoeffding's inequality:

Lemma 2.9.3 (Generalization error for finite classifier)**.** *Given a set B which contains H elements, if the estimator $\hat{\boldsymbol{\beta}}$ belongs to B and the loss function $l \in [0,1]$, then with probability $1-\delta$,*

$$E_{\boldsymbol{x},y} l(\hat{\boldsymbol{\beta}}, \boldsymbol{x}, y) \leq \frac{1}{n} \sum_{\mathrm{i}=1}^{n} l(\hat{\boldsymbol{\beta}}, \boldsymbol{x}^{(\mathrm{i})}, y^{(\mathrm{i})}) + \sqrt{\frac{\log H + \log(1/\delta)}{2n}}.$$

Next, let's consider the same sets B_n and $\tilde{B}_n$ as defined in the proof of Theorem 2.5.1. Due to the posterior contraction result, with probability at least $1 - \exp\{-c'' n \epsilon_n^2\}$, the estimator $\hat{\boldsymbol{\beta}} \in B_n$. Therefore, there must exist some $\boldsymbol{\beta}^{(l)} \in \tilde{B}_n$ such that (2.38) holds, which implies that with probability at least $1 - \exp\{-c'' n \epsilon_n^2\} - \delta$,

$$\begin{aligned} L_0(\hat{\boldsymbol{\beta}}) &\leq L_{c'\epsilon_n/2}(\boldsymbol{\beta}^{(l)}) \leq L_{\mathrm{emp},c'\epsilon_n/2}(\boldsymbol{\beta}^{(l)}) + \sqrt{\frac{\log H + \log(1/\delta)}{2n}} \\ &\leq L_{\mathrm{emp},c'\epsilon_n}(\hat{\boldsymbol{\beta}}) + \sqrt{\frac{\log H + \log(1/\delta)}{2n}}, \end{aligned}$$

where the second inequality is due to Lemma 2.9.3, and $H \leq \exp\{c n \epsilon_n^2\}$. The result then holds if we set $\delta = \exp\{-c n \epsilon_n^2\}$. □

Proof of Theorems 2.5.3 and 2.5.4

The proofs are straightforward and thus omitted.

2.9.5 Mathematical facts of sparse DNN

Consider a sparse DNN model with $H_n - 1$ hidden layer. Let $L_1, \ldots, L_{H_n-1}$ denote the number of node in each hidden layer and r_i be the number of active connections that connect *to* the ith hidden layer (including the bias for the ith hidden layer and weight connections between $\mathrm{i}-1$th and ith layer). Besides, we let $O_{\mathrm{i},\mathrm{j}}(\boldsymbol{\beta}, \boldsymbol{x})$ denote the output value of the jth node in the ith hidden layer

Lemma 2.9.4. *Under assumption A.1, if a sparse DNN has at most* r_n *connectivity (i.e.,* $\sum r_{\mathrm{i}} = r_n$*), and all the weight and bias parameters are bounded by* E_n *(i.e.,* $\|\boldsymbol{\beta}\|_\infty \le E_n$*), then the summation of the outputs of the* i*th hidden layer for* $1 \le \mathrm{i} \le H_n$ *is bounded by*

$$\sum_{\mathrm{j}=1}^{L_{\mathrm{i}}} O_{\mathrm{i,j}}(\boldsymbol{\beta}, \boldsymbol{x}) \le E_n^{\mathrm{i}} \prod_{k=1}^{\mathrm{i}} r_k,$$

where the H_n*-th hidden layer means the output layer.*

Proof. For the simplicity of representation, we rewrite $O_{\mathrm{i,j}}(\boldsymbol{\beta}, \boldsymbol{x})$ as $O_{\mathrm{i,j}}$ when causing no confusion. The lemma is the result from the facts that

$$\sum_{\mathrm{i}=1}^{L_1} |O_{\mathrm{i,j}}| \le r_n E_n, \text{ and } \sum_{\mathrm{j}=1}^{L_{\mathrm{i}}} |O_{\mathrm{i,j}}| \le \sum_{\mathrm{j}=1}^{L_{\mathrm{i}-1}} |O_{\mathrm{i}-1,\mathrm{j}}| E_n r_{\mathrm{i}}.$$

□

Consider two neural networks, $\mu(\boldsymbol{\beta}, \boldsymbol{x})$ and $\mu(\widetilde{\boldsymbol{\beta}}, \boldsymbol{x})$, where the formal one is a sparse network satisfying $\|\boldsymbol{\beta}\|_0 = r_n$ and $\|\boldsymbol{\beta}\|_\infty = E_n$, and its model vector is $\boldsymbol{\gamma}$. If $|\boldsymbol{\beta}_{\mathrm{i}} - \tilde{\boldsymbol{\beta}}_{\mathrm{i}}| < \delta_1$ for all $\mathrm{i} \in \boldsymbol{\gamma}$ and $|\boldsymbol{\beta}_{\mathrm{i}} - \tilde{\boldsymbol{\beta}}_{\mathrm{i}}| < \delta_2$ for all $\mathrm{i} \notin \boldsymbol{\gamma}$, then

Lemma 2.9.5.

$$\max_{\|\boldsymbol{x}\|_\infty \le 1} |\mu(\boldsymbol{\beta}, \boldsymbol{x}) - \mu(\widetilde{\boldsymbol{\beta}}, \boldsymbol{x})| \le \delta_1 H_n (E_n + \delta_1)^{H_n - 1} \prod_{\mathrm{i}=1}^{H_n} r_{\mathrm{i}} + \delta_2 (p_n L_1 + \sum_{\mathrm{i}=1}^{H_n} L_{\mathrm{i}}) \prod_{\mathrm{i}=1}^{H_n} [(E_n + \delta_1) r_{\mathrm{i}} + \delta_2 L_{\mathrm{i}}].$$

Proof. Define $\check{\boldsymbol{\beta}}$ such that $\check{\boldsymbol{\beta}}_{\mathrm{i}} = \tilde{\boldsymbol{\beta}}_{\mathrm{i}}$ for all $\mathrm{i} \in \boldsymbol{\gamma}$ and $\check{\boldsymbol{\beta}}_{\mathrm{i}} = 0$ for all $\mathrm{i} \notin \boldsymbol{\gamma}$. Let $\check{O}_{\mathrm{i,j}}$ denote $O_{\mathrm{i,j}}(\check{\boldsymbol{\beta}}, \boldsymbol{x})$. Then,

$$\begin{aligned}
|\check{O}_{\mathrm{i,j}} - O_{\mathrm{i,j}}| &\le \delta_1 \sum_{\mathrm{j}=1}^{L_{\mathrm{i}-1}} |O_{\mathrm{i}-1,\mathrm{j}}| + E_n \sum_{\mathrm{j}=1}^{L_{\mathrm{i}-1}} |\check{O}_{\mathrm{i}-1,\mathrm{j}} - O_{\mathrm{i}-1,\mathrm{j}}| + \delta_1 \sum_{\mathrm{j}=1}^{L_{\mathrm{i}-1}} |\check{O}_{\mathrm{i}-1,\mathrm{j}} - O_{\mathrm{i}-1,\mathrm{j}}| \\
&\le \delta_1 \sum_{\mathrm{j}=1}^{L_{\mathrm{i}-1}} |O_{\mathrm{i}-1,\mathrm{j}}| + (E_n + \delta_1) \sum_{\mathrm{j}=1}^{L_{\mathrm{i}-1}} |\check{O}_{\mathrm{i}-1,\mathrm{j}} - O_{\mathrm{i}-1,\mathrm{j}}|.
\end{aligned}$$

This implies a recursive result

$$\sum_{\mathrm{j}=1}^{L_{\mathrm{i}}} |\check{O}_{\mathrm{i},\mathrm{j}} - O_{\mathrm{i},\mathrm{j}}| \leq r_{\mathrm{i}}(E_n + \delta_1) \sum_{\mathrm{j}=1}^{L_{\mathrm{i}-1}} |\check{O}_{\mathrm{i}-1,\mathrm{j}} - O_{\mathrm{i}-1,\mathrm{j}}| + r_{\mathrm{i}}\delta_1 \sum_{\mathrm{j}=1}^{L_{\mathrm{i}-1}} |O_{\mathrm{i}-1,\mathrm{j}}|.$$

Due to Lemma 2.9.4, $\sum_{\mathrm{j}=1}^{L_{\mathrm{i}-1}} |O_{\mathrm{i}-1,\mathrm{j}}| \leq E_n^{\mathrm{i}-1} r_1 \cdots r_{\mathrm{i}-1}$. Combined with the fact that $\sum_{\mathrm{j}=1}^{L_1} |\check{O}_{1,\mathrm{j}} - O_{1,\mathrm{j}}| \leq \delta_1 r_1$, one have that

$$|\mu(\boldsymbol{\beta}, \boldsymbol{x}) - \mu(\check{\boldsymbol{\beta}}, \boldsymbol{x})| = \sum_{\mathrm{j}} |\check{O}_{H_n,\mathrm{j}} - O_{H_n,\mathrm{j}}| \leq \delta_1 H_n (E_n + \delta_1)^{H_n - 1} \prod_{\mathrm{i}=1}^{H_n} r_{\mathrm{i}}.$$

Now we compare $\widetilde{O}_{\mathrm{i},\mathrm{j}} := \mu(\widetilde{\boldsymbol{\beta}}, \boldsymbol{x})$ and $\check{O}_{\mathrm{i},\mathrm{j}}$. We have that

$$\sum_{\mathrm{j}=1}^{L_{\mathrm{i}}} |\widetilde{O}_{\mathrm{i},\mathrm{j}} - \check{O}_{\mathrm{i},\mathrm{j}}| \leq \delta_2 L_{\mathrm{i}} \sum_{\mathrm{j}=1}^{L_{\mathrm{i}-1}} |\widetilde{O}_{\mathrm{i}-1,\mathrm{j}} - \check{O}_{\mathrm{i}-1,\mathrm{j}}| + \delta_2 L_{\mathrm{i}} \sum_{\mathrm{j}=1}^{L_{\mathrm{i}-1}} |\check{O}_{\mathrm{i}-1,\mathrm{j}}| + r_{\mathrm{i}}(E_n + \delta_1) \sum_{\mathrm{j}=1}^{L_{\mathrm{i}-1}} |\widetilde{O}_{\mathrm{i}-1,\mathrm{j}} - \check{O}_{\mathrm{i}-1,\mathrm{j}}|,$$

and

$$\sum_{\mathrm{j}=1}^{L_1} |\widetilde{O}_{1,\mathrm{j}} - \check{O}_{1,\mathrm{j}}| \leq \delta_2 p_n L_1.$$

Due to Lemma 2.9.4, we also have that $\sum_{\mathrm{j}=1}^{L_{\mathrm{i}-1}} |\check{O}_{\mathrm{i}-1,\mathrm{j}}| \leq (E_n + \delta_1)^{\mathrm{i}-1} r_1 \cdots r_{\mathrm{i}-1}$. Together, we have that

$$\begin{aligned} |\mu(\widetilde{\boldsymbol{\beta}}, \boldsymbol{x}) - \mu(\check{\boldsymbol{\beta}}, \boldsymbol{x})| &= \sum_{\mathrm{j}} |\widetilde{O}_{H_n,\mathrm{j}} - \check{O}_{H_n,\mathrm{j}}| \\ \leq& \delta_2 (p_n L_1 + \sum_{\mathrm{i}=1}^{H_n} L_{\mathrm{i}}) \prod_{\mathrm{i}=1}^{H_n} [(E_n + \delta_1) r_{\mathrm{i}} + \delta_2 L_{\mathrm{i}}]. \end{aligned}$$

The proof is concluded by summation of the bound for $|\mu(\boldsymbol{\beta}, \boldsymbol{x}) - \mu(\check{\boldsymbol{\beta}}, \boldsymbol{x})|$ and $|\mu(\widetilde{\boldsymbol{\beta}}, \boldsymbol{x}) - \mu(\check{\boldsymbol{\beta}}, \boldsymbol{x})|$. □

2.9.6 Proof of Theorem 2.6.1

Our proof follows the proof of Theorem 2 in [70]. SGLD use the first order integrator (see Lemma 12 of [70] for the detail). Then we have

$$\begin{aligned}\mathbb{E}(\psi(\boldsymbol{\beta}^{(t+1)})) =& \psi(\boldsymbol{\beta}^{(t)}) + \epsilon_t \mathcal{L}_t \psi(\boldsymbol{\beta}^{(t)}) + O(\epsilon_t^2)\\ =& \psi(\boldsymbol{\beta}^{(t)}) + \epsilon_t(\mathcal{L}_t - \mathcal{L})\psi(\boldsymbol{\beta}^{(t)}) + \epsilon_t \mathcal{L}\psi(\boldsymbol{\beta}^{(t)}) + O(\epsilon_t^2).\end{aligned}$$

Note that by Poisson equation, $\mathcal{L}\psi(\boldsymbol{\beta}) = \phi(\boldsymbol{\beta}) - \int \phi(\boldsymbol{\beta})\boldsymbol{\pi}(\boldsymbol{\beta}|D_n, \eta^*, \sigma_{0,n}^*)d\boldsymbol{\beta}$. Taking expectation on both sides of the equation, summing over $t = 0, 1, \ldots, T-1$, and dividing ϵT on both sides of the equation, we have

$$\begin{aligned}&\mathbb{E}\left(\frac{1}{T}\sum_{t=1}^{T-1}\phi(\boldsymbol{\beta}^{(t)}) - \int \phi(\boldsymbol{\beta})\boldsymbol{\pi}(\boldsymbol{\beta}|D_n, \eta^*, \sigma_{0,n}^*)\right)\\ =&\frac{1}{T\epsilon}(\mathbb{E}(\psi(\boldsymbol{\beta}^{(T)})) - \psi(\boldsymbol{\beta}^{(0)})) - \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}(\delta_t\psi(\boldsymbol{\beta}^{(t)})) + O(\epsilon).\end{aligned}$$

To characterize the order of $\delta_t = \mathcal{L}_t - \mathcal{L}$, we first study the difference of the drift term

$$\begin{aligned}&\nabla\log(\boldsymbol{\pi}(\boldsymbol{\beta}^{(t)}|D_{m,n}^{(t)}, \eta^{(t)}, \sigma_{0,n}^{(t)})) - \nabla\log(\boldsymbol{\pi}(\boldsymbol{\beta}^{(t)}|D_n, \eta^*, \sigma_{0,n}^*))\\ =&\sum_{\mathrm{i}=1}^{n}\nabla\log(p_{\boldsymbol{\beta}^{(t)}}(\boldsymbol{x}_{\mathrm{i}}, y_{\mathrm{i}})) - \frac{n}{m}\sum_{\mathrm{j}=1}^{m}\nabla\log(p_{\boldsymbol{\beta}^{(t)}}(\boldsymbol{x}_{\mathrm{i_j}}, y_{\mathrm{i_j}}))\\ &+ \eta^{(t)}\nabla\log(\boldsymbol{\pi}(\boldsymbol{\beta}^{(t)}|\lambda_n, \sigma_{0,n}^{(t)}, \sigma_{1,n})) - \eta^*\nabla\log(\boldsymbol{\pi}(\boldsymbol{\beta}^{(t)}|\lambda_n, \sigma_{0,n}^*, \sigma_{1,n})).\end{aligned}$$

Use of the mini-batch data gives an unbiased estimator of the full gradient, i.e.

$$\mathbb{E}(\sum_{\mathrm{i}=1}^{n}\nabla\log(p_{\boldsymbol{\beta}^{(t)}}(\boldsymbol{x}_{\mathrm{i}}, y_{\mathrm{i}})) - \frac{n}{m}\sum_{\mathrm{j}=1}^{m}\nabla\log(p_{\boldsymbol{\beta}^{(t)}}(\boldsymbol{x}_{\mathrm{i_j}}, y_{\mathrm{i_j}}))) = 0.$$

For the prior part, let $p(\sigma)$ denote the density function of $N(0, \sigma)$. Then we have

$$\begin{aligned}&\nabla\log(\boldsymbol{\pi}(\boldsymbol{\beta}^{(t)}|\lambda_n, \sigma_{0,n}^{(t)}, \sigma_{1,n}))\\ =&-\frac{(1-\lambda_n)p(\sigma_{0,n}^{(t)})}{(1-\lambda_n)p(\sigma_{0,n}^{(t)}) + \lambda_n p(\sigma_{1,n})}\frac{\boldsymbol{\beta}^{(t)}}{\sigma_{0,n}^{(t)\,2}} - \frac{\lambda_n p(\sigma_{1,n})}{(1-\lambda_n)p(\sigma_{0,n}^{(t)}) + \lambda_n p(\sigma_{1,n})}\frac{\boldsymbol{\beta}^{(t)}}{\sigma_{1,n}^2},\end{aligned}$$

and thus $\mathbb{E}|\nabla \log(\boldsymbol{\pi}(\boldsymbol{\beta}^{(t)}|\lambda_n, \sigma_{0,n}^{(t)}, \sigma_{1,n}))| \leq \frac{2\mathbb{E}|\boldsymbol{\beta}^{(t)}|}{{\sigma_{0,n}^*}^2}$. By Assumption 5.2, we have

$$\begin{aligned}
&\mathbb{E}(|\eta^{(t)}\nabla \log(\boldsymbol{\pi}(\boldsymbol{\beta}^{(t)}|\lambda_n, \sigma_{0,n}^{(t)}, \sigma_{1,n})) - \eta^*\nabla \log(\boldsymbol{\pi}(\boldsymbol{\beta}^{(t)}|\lambda_n, \sigma_{0,n}^{*}, \sigma_{1,n}))|) \\
=&\mathbb{E}(|\eta^{(t)}\nabla \log(\boldsymbol{\pi}(\boldsymbol{\beta}^{(t)}|\lambda_n, \sigma_{0,n}^{(t)}, \sigma_{1,n})) - \eta^*\nabla \log(\boldsymbol{\pi}(\boldsymbol{\beta}^{(t)}|\lambda_n, \sigma_{0,n}^{(t)}, \sigma_{1,n}))|) \\
&+ \mathbb{E}(|\eta^*\nabla \log(\boldsymbol{\pi}(\boldsymbol{\beta}^{(t)}|\lambda_n, \sigma_{0,n}^{(t)}, \sigma_{1,n})) - \eta^*\nabla \log(\boldsymbol{\pi}(\boldsymbol{\beta}^{(t)}|\lambda_n, \sigma_{0,n}^{*}, \sigma_{1,n}))|) \\
\leq&\frac{2M}{{\sigma_{0,n}^*}^2}|\eta^{(t)} - \eta^*| + \eta^* M|\sigma_{0,n}^{(t)} - \sigma_{0,n}^*|.
\end{aligned}$$

By Assumption 5.1, $\mathbb{E}(\psi(\boldsymbol{\beta}^{(t)})) \leq \infty$. Then

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}(\delta_t\psi(\boldsymbol{\beta}^{(t)})) = O\left(\frac{1}{T}\sum_{t=0}^{T-1}(|\eta^{(t)} - \eta^*| + |\sigma_{0,n}^{(t)} - \sigma_{0,n}^*|)\right).$$

Note that by assumption 5.1, $|(\psi(\boldsymbol{\beta}^{(T)})) - \psi(\boldsymbol{\beta}^{(0)})|$ is bounded. Then

$$\begin{aligned}
&\mathbb{E}\left(\frac{1}{T}\sum_{t=1}^{T-1}\phi(X_t) - \int \phi(\boldsymbol{\beta})\boldsymbol{\pi}(\boldsymbol{\beta}|D_n, \eta^*, \sigma_{0,n}^*)\right) \\
=&O\left(\frac{1}{T\epsilon} + \frac{\sum_{t=0}^{T-1}(|\eta^{(t)} - \eta^*| + |\sigma_{0,n}^{(t)} - \sigma_{0,n}^*|)}{T} + \epsilon\right).
\end{aligned}$$

3. A KERNEL-EXPANDED STOCHASTIC NEURAL NETWORK

3.1 A Kernel-Expanded Stochastic Neural Network

3.1.1 A Kernel-Expanded Neural Network

Let's start with a brief review for the theory developed in [5] and [6]. Consider a neural network model with h hidden layers. Let $\boldsymbol{Z}_0 = \boldsymbol{X} \in \mathbb{R}^{m_0}$ denote an input vector, let $\boldsymbol{Z}_\mathrm{i} \in \mathbb{R}^{m_\mathrm{i}}$ denote the output vector at layer i for $\mathrm{i} = 1, 2, \ldots, h, h+1$, and let $\boldsymbol{Y} \in \mathbb{R}^{m_{h+1}}$ denote the target output. At each layer i, the neural network calculates its output:

$$\boldsymbol{Z}_\mathrm{i} = \Psi(\boldsymbol{w}_\mathrm{i}\boldsymbol{Z}_{\mathrm{i}-1} + \boldsymbol{b}_\mathrm{i}), \quad \mathrm{i} = 1, 2, \ldots, h, h+1, \tag{3.1}$$

where $\boldsymbol{w}_\mathrm{i} \in \mathbb{R}^{m_\mathrm{i}} \times \mathbb{R}^{m_{\mathrm{i}-1}}$ and $\boldsymbol{b}_\mathrm{i} \in \mathbb{R}^{m_\mathrm{i}}$ denote the weights and bias of the layer i respectively, $\Psi(\boldsymbol{s}) = (\psi(s_1), \ldots, \psi(s_{m_\mathrm{i}}))^T$, and $\psi(\cdot)$ is the activation function used in the network. For convenience, let $p = m_0$ denote the dimension of the input vector, let $\tilde{\boldsymbol{w}}_\mathrm{i} = [\boldsymbol{w}_\mathrm{i}, \boldsymbol{b}_\mathrm{i}]$ denote the matrix of all parameters of layer i for $\mathrm{i} = 1, 2, \ldots, h+1$, and let $\boldsymbol{\theta} = (\tilde{\boldsymbol{w}}_1, \tilde{\boldsymbol{w}}_2, \ldots, \tilde{\boldsymbol{w}}_{h+1}) \in \Theta$. Further, we assume that the network structure is pyramidal with $m_0 \geq m_1 \geq \cdots \geq m_h \geq m_{h+1}$ and, for simplicity, the same activation function $\psi(\cdot)$ is used for all hidden units. Let $U : \Theta \to \mathbb{R}$ be the loss function of the neural network, which is given by

$$U(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{\mathrm{i}=1}^{n} \log \pi(\boldsymbol{Y}^{(\mathrm{i})}|\boldsymbol{\theta}, \boldsymbol{X}^{(\mathrm{i})}) \triangleq \frac{1}{n}\sum_{\mathrm{i}=1}^{n} l(\boldsymbol{Z}_{h+1}^{(\mathrm{i})}), \tag{3.2}$$

where $\pi(\cdot)$ denotes the density/mass function of each observation under the neural network model, n denotes the training sample size, i indexes the training sample, and $\boldsymbol{Z}_{h+1}^{(\mathrm{i})}$ is the output vector of layer $h+1$, and $l : \mathbb{R}^{m_{h+1}} \to \mathbb{R}$ is assumed to be a continuously differentiable loss function, i.e., $l \in C^2(\mathbb{R}^{m_{h+1}})$. In order to study the property of the loss function, [6] made the following assumption:

Assumption 3.1.1. *(i) All training samples are distinct, i.e., $\boldsymbol{X}^{(\mathrm{i})} \neq \boldsymbol{X}^{(\mathrm{j})}$ for all $\mathrm{i} \neq \mathrm{j}$;*

(ii) $\psi(\cdot)$ *is real analytic, strictly monotonically increasing and (a)* $\psi(\cdot)$ *is bounded or (b) there are positive* ρ_1*,* ρ_2*,* ρ_3 *and* ρ_4 *such that* $|\psi(t)| \leq \rho_1 e^{\rho_2 t}$ *for* $t < 0$ *and* $|\psi(t)| \leq \rho_3 t + \rho_4$ *for* $t \geq 0$*;*

(iii) $l \in C^2(\mathbb{R}^{m_{h+1}})$ *and if* $l'(\boldsymbol{a}) = 0$ *then* $\boldsymbol{a}$ *is a global optimum.*

Here a function $\psi : \mathbb{R} \to \mathbb{R}$ is called real analytic if the corresponding Taylor series converges to $\psi(s)$ on an open subset of $\mathbb{R}$. It is easy to see that many of the activation functions, such as *tanh*, *sigmoid* and *softplus*, satisfy 3.1.1-(ii). It is known that the *softplus* function can be viewed as a differentiable approximation to ReLU. 3.1.1-(iii) can be satisfied by any twice continuously differentiable convex loss function, e.g., negative log-Gaussian and log-binomial density/mass functions. The following lemma is a restatement of Theorem 3.4 of [6]. A similar result has also been established in [5].

Lemma 3.1.1. *(Theorem 3.4 of [6]) Suppose Assumption 3.1.1 holds. If (i) the training samples are linearly independent, i.e., rank*$([\boldsymbol{X}, 1_n]) = n$*; and (ii) the weight matrices* $(\tilde{\boldsymbol{w}}_l)_{l=2}^{h+1}$ *have full row rank, i.e., rank*$(\tilde{\boldsymbol{w}}_l) = m_l$ *for* $l = 2, 3, \ldots, h+1$*, then every critical point of the loss function* $U(\boldsymbol{\theta})$ *is a global minimum.*

Among the conditions of Lemma 3.1.1, Assumption 3.1.1 is regular as discussed above, and condition (ii) can be almost surely satisfied by restricting the network structure to be pyramidal. However, condition (i) is not satisfied by many machine learning problems for which the training sample size is much larger than the dimension of the input. To have this condition satisfied, we propose a kernel-expanded neural network (or KNN in short), where each input vector $\boldsymbol{x}$ is mapped into an infinite dimensional feature space by a radial basis function (RBF) kernel $\phi(\boldsymbol{x})$. More precisely, the KNN can be expressed as

$$\begin{aligned} \tilde{\boldsymbol{Y}}_1 &= \boldsymbol{b}_1 + \boldsymbol{\beta}\phi(\boldsymbol{X}), \\ \tilde{\boldsymbol{Y}}_\mathrm{i} &= \boldsymbol{b}_\mathrm{i} + \boldsymbol{w}_\mathrm{i}\Psi(\tilde{\boldsymbol{Y}}_{\mathrm{i}-1}), \quad \mathrm{i} = 2, 3, \ldots, h, \\ \boldsymbol{Y} &= \boldsymbol{b}_{h+1} + \boldsymbol{w}_{h+1}\Psi(\tilde{\boldsymbol{Y}}_h) + \mathrm{e}_{h+1}, \end{aligned} \tag{3.3}$$

where $\mathrm{e}_{h+1} \sim N(0, \sigma^2_{h+1} I_{m_{h+1}})$ is Gaussian random error; $\tilde{\boldsymbol{Y}}_\mathrm{i}, \boldsymbol{b}_\mathrm{i} \in \mathbb{R}^{m_\mathrm{i}}$ for $\mathrm{i} = 1, 2, \ldots, h$; $\boldsymbol{Y}_{h+1}, \boldsymbol{b}_{h+1} \in \mathbb{R}^{m_{h+1}}$; $\Psi(\tilde{\boldsymbol{Y}}_{\mathrm{i}-1}) = (\psi(\tilde{Y}_{\mathrm{i}-1,1}), \psi(\tilde{Y}_{\mathrm{i}-1,2}), \ldots, \psi(\tilde{Y}_{\mathrm{i}-1,m_{\mathrm{i}-1}}))^T$ for $\mathrm{i} = 2, 3, \ldots, h+1$, and $\tilde{Y}_{\mathrm{i}-1,\mathrm{j}}$ is the jth element of $\tilde{\boldsymbol{Y}}_{\mathrm{i}-1}$; $\boldsymbol{w}_{\mathrm{i}} \in \mathbb{R}^{m_{\mathrm{i}} \times m_{\mathrm{i}-1}}$ for $\mathrm{i} = 2, 3, \ldots, h+1$; $\boldsymbol{\beta} \in \mathbb{R}^{m_1 \times d_\phi}$ and d_ϕ denotes the dimension of the feature space of the kernel $\phi(\cdot)$. For the RBF kernel, $d_\phi = \infty$. Note that different kernels can be used for different hidden units of the first hidden layer. For notational simplicity, we consider only the case that the same kernel is used for all the hidden units and $\boldsymbol{Y}$ follows a normal regression model. Replacing the third equation of (3.3) by a logit model will lead to the classification case. In general, we consider only the distribution $\boldsymbol{\pi}(\boldsymbol{Y}|\boldsymbol{\theta}, \boldsymbol{X})$ such that Assumption 3.1.1-(iii) is satisfied, where, with a slight abuse of notation, we use $\boldsymbol{\theta} = (\boldsymbol{b}_1, \boldsymbol{\beta}; \boldsymbol{b}_2, \boldsymbol{w}_2; \ldots, \boldsymbol{b}_{h+1}, \boldsymbol{w}_{h+1})$ to denote the collection of all weights of the KNN.

Compared to formula (3.1), formula (3.3) gives a new presentation form for neural networks, where the feeding operator (used for calculating $\boldsymbol{w}_{\mathrm{i}}\boldsymbol{Z}_{\mathrm{i}-1} + \boldsymbol{b}_{\mathrm{i}}$) and the activation operator $\psi(\cdot)$ are separated into two equations. As shown later, such a representation facilitates parameter estimation for the neural network when auxiliary noise are introduced into the model.

For KNN, since the input vector has been mapped into an infinite dimensional feature space, the Gram matrix $\boldsymbol{K} = (k_{\mathrm{ij}})$, where $k_{\mathrm{ij}} = \phi^T(\boldsymbol{x}_{\mathrm{i}})\phi(\boldsymbol{x}_{\mathrm{j}})$, can be of full rank, i.e., $\mathrm{rank}(\boldsymbol{K}) = n$. This means the transformed samples $\phi(\boldsymbol{X}^{(1)}), \phi(\boldsymbol{X}^{(2)}), \ldots, \phi(\boldsymbol{X}^{(n)})$ are linearly independent. In addition, we can restrict the structure of the KNN to be pyramidal, and choose the activation and loss function such that Assumption 3.1.1 is satisfied. Therefore, by Lemma 3.1.1, every critical point of the KNN model is a global minimum. In summary, we have the following theorem with the proof as argued above.

Theorem 3.1.1. *For a KNN model given in (3.3), if Assumption 3.1.1 holds, an RBF kernel is used in the input layer, and the weight matrices $(\tilde{\boldsymbol{w}}_l)_{l=2}^{h+1}$ are of full row rank, then every critical point of its loss function is a global minimum.*

Other than the RBF kernel, the polynomial kernel might also satisfy Theorem 3.1.1 for certain problems. For an input vector $\boldsymbol{x} \in \mathbb{R}^p$, the dimension of its feature space is $\binom{p+q}{q}$, where q denotes the degree freedom of the polynomial kernel. Therefore, if the resulting Gram matrix is of full rank, then the transformed samples $\phi(\boldsymbol{X}^{(1)}), \ldots, \phi(\boldsymbol{X}^{(n)})$ are also linearly independent. However, as stated in Assumption 3.1.4, the K-StoNet requires the kernel to be universal, so the polynomial kernel is not used.

3.1.2 A Kernel-Expanded StoNet as an Approximator to KNN

As shown in Theorem 3.1.1, the KNN has a nice loss surface, where every critical point is a global minimum. However, training the KNN using a gradient-based algorithm is infeasible, as the transformed features are not explicitly available. Based on the kernel representer theorem [85], [86], one might consider to replace the first equation of (3.3) by

$$\tilde{\boldsymbol{Y}}_1 = \boldsymbol{b}_1 + \sum_{\mathrm{i}=1}^{n} \boldsymbol{w}_1^{(\mathrm{i})} K(\boldsymbol{X}^{(\mathrm{i})}, \boldsymbol{X}), \tag{3.4}$$

where $\boldsymbol{w}_1^{(\mathrm{i})} \in \mathbb{R}^{m_1}$ and $K(\boldsymbol{X}^{(\mathrm{i})}, \boldsymbol{X}) = \phi^T(\boldsymbol{X}^{(\mathrm{i})})\phi(\boldsymbol{X})$ is explicitly available, and then train such an over-parameterized neural network model using a regularization method. However, the global optimality property established in Theorem 3.1.1 might not hold for the regularized KNN any more, because the proof of Theorem 3.1.1 relies on the back propagation formula of the neural network (see the proof of Theorem 3.4 in [6] for the detail), while that formula cannot be easily generalized to regularized loss functions. Moreover, for a nonlinear kernel regression $\boldsymbol{Y} = g(\tilde{\boldsymbol{Y}}_1) + \mathrm{e} = g(\boldsymbol{b}_1 + \boldsymbol{\beta}\phi(\boldsymbol{X})) + \mathrm{e}$, where $g(\cdot)$ represents a nonlinear mapping from $\tilde{\boldsymbol{Y}}_1$ to the output layer, the kernel representer theorem does not hold for $g(\cdot)$ in general and, therefore, (3.4) and the first equation of (3.3) might not be equivalent for the KNN. Recall that SVR is a special case of the kernel regression with the identity mapping $g(\tilde{\boldsymbol{Y}}_1) = \tilde{\boldsymbol{Y}}_1$.

To tackle this issue, we introduce a K-StoNet model (depicted by Figure 3.1) by adding auxiliary noise to $\tilde{\boldsymbol{Y}}_{\mathrm{i}}$'s, $\mathrm{i} = 1, 2, \ldots, h$, in (3.3). The resulting model is given by

$$\begin{aligned} \boldsymbol{Y}_1 &= \boldsymbol{b}_1 + \boldsymbol{\beta}\phi(\boldsymbol{X}) + \mathrm{e}_1, \\ \boldsymbol{Y}_{\mathrm{i}} &= \boldsymbol{b}_{\mathrm{i}} + \boldsymbol{w}_{\mathrm{i}}\Psi(\boldsymbol{Y}_{\mathrm{i}-1}) + \mathrm{e}_{\mathrm{i}}, \quad \mathrm{i} = 2, 3, \ldots, h, \\ \boldsymbol{Y} &= \boldsymbol{b}_{h+1} + \boldsymbol{w}_{h+1}\Psi(\boldsymbol{Y}_h) + \mathrm{e}_{h+1}, \end{aligned} \tag{3.5}$$

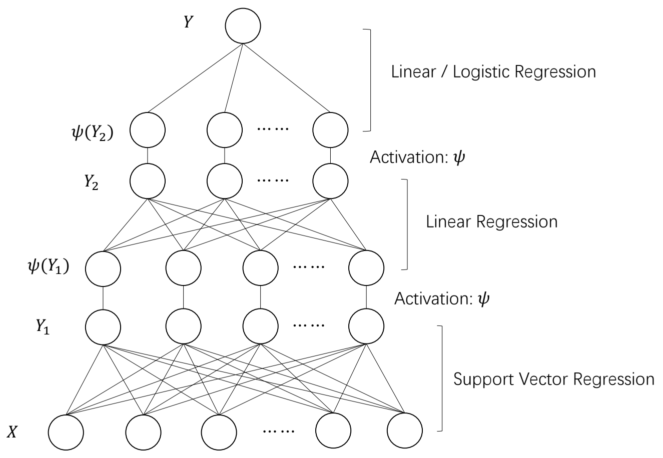


Figure 3.1. An illustrative plot of K-StoNet

where $Y_1, Y_2, \ldots, Y_h$ are latent variables. To complete the model specification, we assume that $\mathrm{e}_\mathrm{i} \sim N(0, \sigma_\mathrm{i}^2 I_{m_\mathrm{i}})$ for $\mathrm{i} = 2, 3, \ldots, h, h+1$, and each component of e_1 is independent and identically distributed with the density function given by

$$f(x) = \frac{C}{2(1+C\epsilon)} \mathrm{e}^{-C|x|_\varepsilon}, \tag{3.6}$$

where $|x|_\epsilon = \max(0, |x| - \varepsilon)$ is an ε-intensive loss function, and C is a scale parameter. It is known that this distribution has mean 0 and variance $\frac{2}{C^2} + \frac{\varepsilon^2(\varepsilon C+3)}{3(\varepsilon C+1)}$. For classification networks, the last equation of (3.5) is replaced by a generalized linear model (GLM), for which the parameter σ_{h+1}^2 plays the role of temperature for the binomial or multinomial distribution formed at the output layer. In summary, $\{C, \varepsilon, \sigma_2^2, \ldots, \sigma_h^2, \sigma_{h+1}^2\}$ work together to control the variation of the latent variables $\{\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_h\}$ as discussed in Section 3.1.4. As shown later, such specifications for the auxiliary noise enable the K-StoNet parameters to be estimated by solving a series of convex optimization problems and the prediction uncertainty to be easily assessed via a recursive formula.

To establish that K-StoNet is a valid approximator to KNN, i.e., asymptotically they have the same loss function, some assumptions need to be imposed on the model. To indicate their dependence on the training sample size n, we redenote C by C_n, ε by ε_n, and σ_i by $\sigma_{n,\mathrm{i}}$ for $\mathrm{i} = 2, 3, \ldots, h+1$. For (3.6), we assume $\varepsilon_n \leq 1/C_n$ holds as $n \to \infty$. As in KNN, we let $\boldsymbol{\theta}$ denote the parameter vector of K-StoNet, and let d_θ denote the dimension of $\boldsymbol{\theta}$. Since, for the KNN, any local minimum is also a global minimum, we can restrict Θ to a compact set which is large enough such that one local minimum is contained. This is essentially a technical condition. In practice, if a local convergence algorithm is used for training the KNN, it is then equivalent to set $\Theta = \mathbb{R}^{d_\theta}$, as the regions beyond a neighborhood of the starting point will never be visited by the algorithm.

Assumption 3.1.2. *(i) Θ is compact, which can be contained in a d_θ-ball centered at the origin and of radius r; (ii) $\mathbb{E}(\log \boldsymbol{\pi}(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta}))^2 < \infty$ for any $\boldsymbol{\theta} \in \Theta$; (iii) the activation function $\psi(\cdot)$ is c'-Lipschitz continuous for some constant c'; (iv) the network's depth h and widths m_i's are all allowed to increase with n; and (v) $\sigma_{n,h+1} = O(1)$, and $m_{h+1}(\prod_{\mathrm{i}=k+1}^{h} m_\mathrm{i}^2) m_k \sigma_{n,k}^2 \prec \frac{1}{h}$ for $k \in \{1, 2, \ldots, h\}$, where $\sigma_{n,1} = 1/C_n$.*

Assumption 3.1.2-(ii) is the regularity condition for the distribution of $\boldsymbol{Y}$. Assumption 3.1.2-(iii) can be satisfied by many activation functions such as *tanh*, *sigmoid* and *softplus*. Assumption 3.1.2-(v) constrains the size of the noise added to each hidden layer such that the K-StoNet has asymptotically the same loss function as the KNN when the training sample size becomes large, where the factor $m_{h+1}(\prod_{i=k+1}^{h} m_i^2)m_k$ is derived in the proof of Theorem 3.1.2 and its square root can be interpreted as the amplification factor of the noise e_k at the output layer.

As stated in Assumption 3.1.4, the SVR in K-StoNet is required to work with a universal kernel such as RBF. By [34], [35], [87], such a SVR possesses the universal approximation capability, so does K-StoNet. Therefore, K-StoNet is not necessarily very deep or wide, while having any continuous function approximated arbitrarily well as the training sample size $n \to \infty$. For this reason, we may restrict the depth $h = O(1)$, and restrict $m_1 = o(\sqrt{n})$ and thus $m_i = o(\sqrt{n})$ for all $i = 2, 3, \ldots, h$ due to the pyramidal structure of K-StoNet. The universal approximation property of SVR is quite different from that of the neural networks. The former depends on the training sample size, while the latter depends on the network size. The K-StoNet lies in the between of them.

Theorem 3.1.2 shows that the K-StoNet and KNN have asymptotically the same training loss function, whose proof is given in the Section 3.7.1.

Theorem 3.1.2. *Suppose Assumption 3.1.2 holds. Then the K-StoNet (3.5) and the KNN (3.3) have asymptotically the same loss function, i.e., as n → ∞,*

$$\sup_{\boldsymbol{\theta}\in\Theta} \Big|\frac{1}{n}\sum_{i=1}^{n}\log \pi(\boldsymbol{Y}^{(i)}, \boldsymbol{Y}_{\text{mis}}^{(i)}|\boldsymbol{X}^{(i)}, \boldsymbol{\theta}) - \frac{1}{n}\sum_{i=1}^{n}\log \pi(\boldsymbol{Y}^{(i)}|\boldsymbol{X}^{(i)}, \boldsymbol{\theta})\Big| \xrightarrow{p} 0, \tag{3.7}$$

where $\boldsymbol{Y}_{\text{mis}}^{(i)} = (\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_h)$ denotes the collection of latent variables in (3.5), and $\xrightarrow{p}$ denotes convergence in probability.

Let $Q^*(\boldsymbol{\theta}) = \mathbb{E}(\log \pi(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta}))$, where the expectation is taken with respect to the joint distribution $\pi(\boldsymbol{X}, \boldsymbol{Y})$. By Assumption 3.1.2-(i) & (ii), and the law of large numbers,

$$\frac{1}{n}\sum_{i=1}^{n}\log \pi(\boldsymbol{Y}^{(i)}|\boldsymbol{X}^{(i)}, \boldsymbol{\theta}) - Q^*(\boldsymbol{\theta}) \xrightarrow{p} 0, \tag{3.8}$$

holds uniformly over Θ. Further, we make the following assumptions for $Q^*(\boldsymbol{\theta})$:

Assumption 3.1.3. *(i) $Q^*(\boldsymbol{\theta})$ is continuous in $\boldsymbol{\theta}$ and uniquely maximized at $\boldsymbol{\theta}^*$; (ii) for any $\epsilon > 0$, $\sup_{\boldsymbol{\theta}\in\Theta\setminus B(\epsilon)} Q^*(\boldsymbol{\theta})$ exists, where $B(\epsilon) = \{\boldsymbol{\theta} : \|\boldsymbol{\theta} - \boldsymbol{\theta}^*\| < \epsilon\}$, and $\delta = Q^*(\boldsymbol{\theta}^*) - \sup_{\boldsymbol{\theta}\in\Theta\setminus B(\epsilon)} Q^*(\boldsymbol{\theta}) > 0$.*

Assumption 3.1.3 restricts the shape of $Q^*(\boldsymbol{\theta})$ around the global maximizer, which cannot be discontinuous or too flat. Given nonidentifiability of the neural network model (see e.g. [69]), we here have implicitly assumed that each $\boldsymbol{\theta}$ in the KNN and K-StoNet is unique up to loss-invariant transformations, such as reordering some hidden units and simultaneously changing the signs of some weights and biases.

Lemma 3.1.2. *Suppose Assumptions 3.1.2-3.1.3 hold, and $\pi(\boldsymbol{Y}, \boldsymbol{Y}_{\text{mis}}|\boldsymbol{X}, \boldsymbol{\theta})$ is continuous in $\boldsymbol{\theta}$. Let $\hat{\boldsymbol{\theta}}_n = \arg\max_{\boldsymbol{\theta}\in\Theta} \left\{\frac{1}{n}\sum_{\text{i}=1}^{n} \log \pi(\boldsymbol{Y}^{(\text{i})}, \boldsymbol{Y}_{\text{mis}}^{(\text{i})} \middle| \boldsymbol{X}^{(\text{i})}, \boldsymbol{\theta})\right\}$. Then $\|\hat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}^*\| \xrightarrow{p} 0$ as $n \to \infty$.*

The proof of Lemma 3.1.2 is given in the Section 3.7.1. It implies that the KNN can be trained by training K-StoNet as the sample size n becomes large.

3.1.3 The Imputation-Regularized Optimization Algorithm

To train the K-StoNet, we propose to use the imputation-regularized optimization (IRO) algorithm [33]. Consider a missing data problem, where $\boldsymbol{Z}_{\text{obs}}$ denotes observed data, $\boldsymbol{Z}_{\text{mis}}$ denotes missing data, and $\boldsymbol{\vartheta}$ denotes the parameter. The IRO algorithm aims to find a consistent estimate of $\boldsymbol{\vartheta}$ by maximizing $\mathbb{E}\log\pi(\boldsymbol{Z}_{\text{obs}}, \boldsymbol{Z}_{\text{mis}}|\boldsymbol{\vartheta})$, where the expectation is taken with respect to the joint distribution of $(\boldsymbol{Z}_{\text{obs}}, \boldsymbol{Z}_{\text{mis}})$. Conceptually, this is a little different from the expectation-maximization (EM) algorithm [88] and stochastic EM algorithm [89], which aim to estimate $\boldsymbol{\vartheta}$ by maximizing the marginal likelihood function $\pi(\boldsymbol{Z}_{\text{obs}}|\boldsymbol{\vartheta})$. Practically, the IRO algorithm works in similar way to stochastic EM by iterating between an imputation step and an optimization step, but for which a regularization term can be included in the loss function at each optimization step for ensuring the convergence of the estimate under the high-dimensional scenario.

For K-StoNet, the IRO algorithm is to estimate $\boldsymbol{\theta}$ by maximizing $\mathbb{E}\log\pi(\boldsymbol{Y}, \boldsymbol{Y}_{\text{mis}}|\boldsymbol{X}, \boldsymbol{\theta})$, which is equivalent to maximizing $Q^*(\boldsymbol{\theta}) = \mathbb{E}\log\pi(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta})$ as implied by (3.7) and (3.8). This coincides with the goal of KNN training if the stochastic gradient descent (SGD) algorithm is used. Let $\hat{\boldsymbol{\theta}}_n^{(t)}$ denote the estimate of $\boldsymbol{\theta}$ obtained by the IRO algorithm at iteration t. The IRO algorithm starts with an initial guess $\hat{\boldsymbol{\theta}}_n^{(0)}$ and then iterates between the following two steps:

- **I-step**: *For each sample* $(\boldsymbol{X}^{(\mathrm{i})}, \boldsymbol{Y}^{(\mathrm{i})})$, *draw* $\boldsymbol{Y}_{\mathrm{mis}}^{(\mathrm{i})}$ *from the predictive distribution*

$$g(\boldsymbol{Y}_{\mathrm{mis}}|\boldsymbol{Y}^{(\mathrm{i})}, \boldsymbol{X}^{(\mathrm{i})}, \hat{\boldsymbol{\theta}}^{(t)}).$$

- **RO-step**: *Based on the pseudo-complete data, find an updated estimate* $\hat{\boldsymbol{\theta}}_n^{(t+1)}$ *by minimizing the penalized loss function, i.e.,*

$$\hat{\boldsymbol{\theta}}_n^{(t+1)} = \arg\min \left\{ -\frac{1}{n} \sum_{\mathrm{i}=1}^{n} \log \pi(\boldsymbol{Y}^{(\mathrm{i})}, \boldsymbol{Y}_{\mathrm{mis}}^{(\mathrm{i})} \Big| \boldsymbol{X}^{(\mathrm{i})}, \boldsymbol{\theta}) + P_{\lambda_n}(\boldsymbol{\theta}) \right\}, \tag{3.9}$$

where the penalty function $P_{\lambda_n}(\boldsymbol{\theta})$ *is chosen such that* $\hat{\boldsymbol{\theta}}_n^{(t+1)}$ *forms a consistent estimate of*

$$\begin{aligned} \boldsymbol{\theta}_*^{(t)} &= \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\theta}_n^{(t)}} \log \pi(\boldsymbol{Y}, \boldsymbol{Y}_{\mathrm{mis}}|\boldsymbol{X}, \boldsymbol{\theta}) \\ &= \int \log \pi(\boldsymbol{Y}_{\mathrm{mis}}, \boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta})) g(\boldsymbol{Y}_{\mathrm{mis}}|\boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{\theta}_n^{(t)}) \pi(\boldsymbol{Y}|\boldsymbol{\theta}^*, \boldsymbol{X}) \pi(\boldsymbol{X}) d\boldsymbol{Y}_{\mathrm{mis}} d\boldsymbol{Y} d\boldsymbol{X} \end{aligned}$$

$\boldsymbol{\theta}^*$ *denotes the true parameter of the model, and* $\pi(\boldsymbol{X})$ *denotes the density function of* $\boldsymbol{X}$.

For the K-StoNet, the joint distribution $\pi(\boldsymbol{Y}, \boldsymbol{Y}_{\mathrm{mis}}|\boldsymbol{X}, \boldsymbol{\theta})$ can be factored as

$$\pi(\boldsymbol{Y}, \boldsymbol{Y}_{\mathrm{mis}}|\boldsymbol{X}, \boldsymbol{\theta}) = \pi(\boldsymbol{Y}_1|\boldsymbol{X}, \tilde{\boldsymbol{w}}_1)[\prod_{\mathrm{i}=2}^{h} \pi(\boldsymbol{Y}_{\mathrm{i}}|\boldsymbol{Y}_{\mathrm{i}-1}, \tilde{\boldsymbol{w}}_{\mathrm{i}})]\pi(\boldsymbol{Y}|\boldsymbol{Y}_h, \tilde{\boldsymbol{w}}_{h+1}). \tag{3.10}$$

Therefore, the optimization in (3.9) can be executed separately for each of the hidden and output layers with an appropriately specified penalty function. That is, *K-StoNet can be trained by solving a series of lower dimensional optimization problems.*

For the first hidden layer, the RO-step is reduced to solving a SVR for each hidden unit. As described in [90], the parameter $\boldsymbol{\beta}$ in (3.5) can be estimated by solving a regularized optimization problem:

$$\arg\min_{\boldsymbol{\beta},\boldsymbol{b}_1} \frac{1}{2}||\boldsymbol{\beta}||_2^2 + \frac{\tilde{C}_n}{n}\sum_{\mathrm{i}=1}^{n} |\boldsymbol{Y}_1^{(\mathrm{i})} - \boldsymbol{\beta}\phi(\boldsymbol{X}^{(\mathrm{i})}) - \boldsymbol{b}_1|_{\varepsilon}, \tag{3.11}$$

where the first term represents a penalty function, and $\tilde{C}_n$ represents the regularization parameter. We can set $\tilde{C}_n = C_n$ given in (3.6), but not necessarily. In general, their values should make Assumptions 3.1.2-(v) and 3.1.4-(i) hold. The consistency of the SVR estimator $\hat{\boldsymbol{\beta}}^T\phi(\boldsymbol{x}) + \hat{\boldsymbol{b}}_1$, which is the basic requirement by the IRO algorithm, has been established in [91] by assuming that a universal kernel [34], [92] such as RBF is used in (3.11). Equivalently, this is to reparameterize the SVR layer by kernel-based regression. By the kernel representer theorem [85], [86], the solution to the regularized optimization problem (3.11) leads to the representer of the first equation of (3.5) as

$$\boldsymbol{Y}_1 = \hat{\boldsymbol{b}}_1 + \sum_{\mathrm{i}=1}^{n} \hat{\boldsymbol{w}}_1^{(\mathrm{i})} K(\boldsymbol{X}^{(\mathrm{i})}, \boldsymbol{X}) + \mathrm{e}_1. \tag{3.12}$$

In what follows, we will use $\check{\boldsymbol{w}}_1 = (\hat{\boldsymbol{w}}_1^{(1)}, \ldots, \hat{\boldsymbol{w}}_1^{(n)}, \hat{\boldsymbol{b}}_1)$ to denote the estimator for the parameters of the SVR layer.

For other hidden layers, the RO-step is reduced to solving a linear regression for each hidden unit using a regularization method. To ensure convexity of the resulting objective function, a Lasso penalty [72] can be used. Alternatively, some nonconvex amenable penalties with vanishing derivatives away from the origin, such as the SCAD [93] and MCP [94], can also be used. As shown in [95], for such nonconvex amenable penalties, any stationary point in a compact region around the true regression coefficients can be used to consistently estimate the parameters and recover the support of the underlying true regression.

For the output layer, the RO-step is reduced to solving a multinomial logistic or multivariate linear regression, depending on the problem under consideration. The Lasso, SCAD and MCP penalties can again be used for them by the theory of [95]. In practice, this step can also be simplified to solving a linear or logistic regression for each output unit by ignoring the correlation between different components of $\boldsymbol{Y}$.

In summary, we have the pseudo-code given in Algorithm 3 for training K-StoNet, where $\breve{\boldsymbol{w}}_{\mathrm{i}}^{(t)}$ denotes the estimate of the parameters for the layer i at iteration t, $(\boldsymbol{Y}_0^{(s)}, \boldsymbol{Y}_{h+1}^{(s)}) = (\boldsymbol{X}^{(s)}, \boldsymbol{Y}^{(s)})$ denotes a training sample, $(\boldsymbol{Y}_1^{(s,t)}, \ldots, \boldsymbol{Y}_h^{(s,t)})$ denotes the latent variables imputed for training sample s at iteration t. For convenience, we occasionally use the notation $\boldsymbol{Y}_0^{(s,t)} = \boldsymbol{Y}_0^{(s)}$ and $\boldsymbol{Y}_{h+1}^{(s,t)} = \boldsymbol{Y}_{h+1}^{(s)}$.

For Algorithm 3, we have a few remarks:

- The Hamiltonian Monte Carlo (HMC) algorithm [96]–[98] is employed in the backward imputation step. Other MCMC algorithms such as Langevin Monte Carlo [99] and the Gibbs sampler [100] can also be employed there.

- In the parameter update step, a Lasso penalty [72] is used in (3.15) to induce the sparsity of StoNet, while ensuring convexity of the minimization problems. Therefore, *K-StoNet is trained by solving a series of convex optimization problems.* Note that the minimization in (3.14) is known as a convex quadratic programming problem [101], [102]. Although solving the convex optimization problems is more expensive than a single gradient update, the IRO algorithm converges very fast, usually within tens of iterations.

- The major computational cost of K-StoNet comes from the SVR step when the sample size is large. The computational complexity for solving an SVR is $O(n^2p+n^3)$, and that for solving a linear/logistic regression is bounded by $O(nm_1^2 + m_1^3)$, while $m_1 \prec n^{1/2}$ is usually recommended. A scalable SVR solver will accelerate the computation of K-StoNet substantially. This issue will be further discussed at the end of the chapter.

- If $m_1 \prec \sqrt{n}$ holds, then the penalty in (3.15) can be simply removed for computational simplicity, while ensuring asymptotic normality of the resulting regression coefficient estimates by [53].

Algorithm 3 The IRO Algorithm for K-StoNet Training

```
**Input**: the total iteration number $T$, the Monte Carlo step number $t_{HMC}$, and the learning rate sequences $\{\epsilon_{t,\mathrm{i}} : t = 1, 2, \ldots, T; \mathrm{i} = 1, 2, \ldots, h+1\}$.
**Initialization**: Randomly initialize the network parameters $\hat{\boldsymbol{\theta}}_n^{(0)} = (\check{\boldsymbol{w}}_1^{(0)}, \ldots, \check{\boldsymbol{w}}_{h+1}^{(0)})$.
**for** t=1,2,...,T **do**
  **STEP 1. Backward Imputation:** For each observation $s$, impute the latent variables in the order from layer $h$ to layer 1. More explicitly, impute $\boldsymbol{Y}_\mathrm{i}^{(s,t)}$ from the distribution $\pi(\boldsymbol{Y}_\mathrm{i}^{(s,t)}|\boldsymbol{Y}_{\mathrm{i}+1}^{(s,t)}, \boldsymbol{Y}_{\mathrm{i}-1}^{(s,t)}, \check{\boldsymbol{w}}_\mathrm{i}^{(t-1)}, \check{\boldsymbol{w}}_{\mathrm{i}+1}^{(t-1)}) \propto \pi(\boldsymbol{Y}_\mathrm{i}^{(s,t)}|\boldsymbol{Y}_{\mathrm{i}-1}^{(s,t)}, \check{\boldsymbol{w}}_\mathrm{i}^{(t-1)})\pi(\boldsymbol{Y}_{\mathrm{i}+1}^{(s,t)}|\boldsymbol{Y}_\mathrm{i}^{(s,t)}, \check{\boldsymbol{w}}_{\mathrm{i}+1}^{(t-1)})$ by running HMC in $t_{HMC}$ steps, where $\pi(\boldsymbol{Y}_1^{(s,t)}|\boldsymbol{X}^{(s)}, \check{\boldsymbol{w}}_1^{(t-1)})$ can be expressed based on (3.12).
  **(1.1) Initialization**: Initialize $\boldsymbol{v}_\mathrm{i}^{(s,0)} = \boldsymbol{0}$, and initialize $\boldsymbol{Y}_\mathrm{i}^{(s,t,0)}$ by KNN, i.e., calculating $\boldsymbol{Y}_\mathrm{i}^{(s,t,0)}$ for $\mathrm{i} = 1, 2, \ldots, h$ in (3.5) by setting the random errors to zero.
  **(1.2) Imputation**:
  **for** k = 1, 2, ..., $t_{HMC}$ **do**
    **for** i = h,h-1,..., 1 **do**
      $$\begin{aligned} \boldsymbol{v}_\mathrm{i}^{(s,k)} &= (1-\alpha)\boldsymbol{v}_\mathrm{i}^{(s,k-1)} + \epsilon_{t,\mathrm{i}}\nabla_{\boldsymbol{Y}_\mathrm{i}^{(s,t,k-1)}} \log \pi(\boldsymbol{Y}_\mathrm{i}^{(s,t,k-1)}|\boldsymbol{Y}_{\mathrm{i}-1}^{(s,t,k-1)}, \check{\boldsymbol{w}}_\mathrm{i}^{(t-1)}) \\ &\quad + \epsilon_{t,\mathrm{i}}\nabla_{\boldsymbol{Y}_\mathrm{i}^{(s,t,k-1)}} \log \pi(\boldsymbol{Y}_{\mathrm{i}+1}^{(s,t,k)}|\boldsymbol{Y}_\mathrm{i}^{(s,t,k-1)}, \check{\boldsymbol{w}}_{\mathrm{i}+1}^{(t-1)}) + \sqrt{2\alpha\epsilon_{t,\mathrm{i}}}\boldsymbol{z}^{(s,t,k)}, \quad (3.13) \\ \boldsymbol{Y}_\mathrm{i}^{(s,t,k)} &= \boldsymbol{Y}_\mathrm{i}^{(s,t,k-1)} + \boldsymbol{v}_\mathrm{i}^{(s,k)}, \end{aligned}$$
      where $\boldsymbol{z}^{(s,t,k)} \sim N(0, \boldsymbol{I}_{m_\mathrm{i}})$, $\epsilon_{t,\mathrm{i}}$ is the learning rate, and $1-\alpha$ is the momentum decay factor ($\alpha = 1$ corresponds to Langevin Monte Carlo).
    **end for**
  **end for**
  **(1.3) Output**: Set $\boldsymbol{Y}_\mathrm{i}^{(s,t)} = \boldsymbol{Y}_\mathrm{i}^{(s,t,t_{HMC})}$ for $\mathrm{i} = 1, 2, \ldots, h$.
  **STEP 2. Parameter Updating:** Update the estimates $(\check{\boldsymbol{w}}_1^{(t-1)}, \check{\boldsymbol{w}}_2^{(t-1)}, \ldots, \check{\boldsymbol{w}}_{h+1}^{(t-1)})$ by solving $h+1$ penalized multivariate regressions separately.
  **(2.1) SVR layer**:
  $$\check{\boldsymbol{w}}_1^{(t)} = \arg\min_{\boldsymbol{\beta}, \boldsymbol{b}_1} \left\{ \frac{\tilde{C}_{n,1}^{(t)}}{n} \sum_{s=1}^{n} \||\boldsymbol{Y}_1^{(s,t)} - \boldsymbol{\beta}^T\phi(Y_0^{(s,t)}) - \boldsymbol{b}_1|_\varepsilon\|_1 + \frac{1}{2}\|\boldsymbol{\beta}\|_2^2 \right\}, \quad (3.14)$$
  where $\tilde{C}_{n,1}^{(t)}$ is the regularization parameter used at iteration $t$.
  **(2.2) Regression layers**:
  **for** i=2,3,...,h+1 **do**
    $$\check{\boldsymbol{w}}_\mathrm{i}^{(t)} = \arg\min_{\boldsymbol{w}_\mathrm{i}, \boldsymbol{b}_\mathrm{i}} \left\{ \frac{1}{n} \sum_{s=1}^{n} \|\boldsymbol{Y}_\mathrm{i}^{(s,t)} - \boldsymbol{w}_\mathrm{i}\psi_\mathrm{i}(\boldsymbol{Y}_{\mathrm{i}-1}^{(s,t)}) - \boldsymbol{b}_\mathrm{i}\|_2^2 + P_{\lambda_{n,\mathrm{i}}^{(t)}}(\tilde{\boldsymbol{w}}_\mathrm{i}) \right\}, \quad (3.15)$$
    where $\lambda_{n,\mathrm{i}}^{(t)}$ is the regularization parameter used for layer i at iteration $t$.
  **end for**
  **(2.3) Output:** Denote the updated estimate by $\hat{\boldsymbol{\theta}}^{(t)} = (\check{\boldsymbol{w}}_1^{(t)}, \ldots, \check{\boldsymbol{w}}_{h+1}^{(t)})$.
**end for**
```

Like the stochastic EM algorithm, the IRO algorithm generates two interleaved Markov chains:

$$\hat{\boldsymbol{\theta}}_n^{(0)} \to (\boldsymbol{Y}_1^{(1)}, \ldots, \boldsymbol{Y}_h^{(1)}) \to \hat{\boldsymbol{\theta}}_n^{(1)} \to (\boldsymbol{Y}_1^{(2)}, \ldots, \boldsymbol{Y}_h^{(2)}) \to \hat{\boldsymbol{\theta}}_n^{(2)} \to \cdots ,$$

whose convergence theory has been studied in [33]. To ensure the convergence of the Markov chains in K-StoNet training, we make following assumptions for the regularization parameters used in (3.14) and (3.15):

Assumption 3.1.4. *(i) A universal kernel such as RBF is used in the SVR layer, and for each* $t \in \{1, 2, \ldots, T\}$, $1 \prec \tilde{C}_{n,1}^{(t)} \prec \sqrt{n}$ *holds; and (ii) for each* $t \in \{1, 2, \ldots, T\}$ *and each* $\mathrm{i} \in \{2, 3, \ldots, h+1\}$, $\sup_{\tilde{\boldsymbol{w}}_\mathrm{i} \in \Theta_\mathrm{i}} P_{\lambda_{n,\mathrm{i}}}^{(t)}(\tilde{\boldsymbol{w}}_\mathrm{i}) \to 0$ *holds as* $n \to \infty$*, where* Θ_i *denotes the sample space of* $\tilde{\boldsymbol{w}}_\mathrm{i}$.

Assumption 3.1.4-(i) ensures consistency of the regression function estimator in the SVR step by Theorem 12 of [91]. Assumptions 3.1.4-(ii) ensures consistency of the weight estimators in the output and other hidden layers. For the Lasso penalty, we can set $P_{\lambda_{n,\mathrm{i}}^{(t)}}(\tilde{\boldsymbol{w}}_\mathrm{i}) = \lambda_{n,\mathrm{i}}^{(t)} \|\tilde{\boldsymbol{w}}_\mathrm{i}\|_1$ and $\lambda_{n,\mathrm{i}}^{(t)} = O(\sqrt{\log(m_{\mathrm{i}-1})/n})$ for any $t \in \{1, 2, \ldots, T\}$ and $\mathrm{i} \in \{2, 3, \ldots, h+1\}$. Since Θ is bounded as assumed in 3.1.2-(i), 3.1.4-(ii) is satisfied. In summary, we have the following theorem which is essentially a restatement of Theorem 4 and Corollary 3 of [33] and therefore whose proof is omitted.

Theorem 3.1.3. *(Consistency) Suppose that Assumptions 3.1.1-3.1.4 hold and, further, the general regularity conditions on missing data (given in [33]) hold. Then for sufficiently large* n*, sufficiently large* T*, and almost every* $(\boldsymbol{X}, \boldsymbol{Y})$*-sequence,* $\|\hat{\boldsymbol{\theta}}_n^{(T)} - \boldsymbol{\theta}^*\| \xrightarrow{p} 0$ *and* $\|\frac{1}{T}\sum_{t=1}^T \hat{\boldsymbol{\theta}}_n^{(t)} - \boldsymbol{\theta}^*\| \xrightarrow{p} 0$*. In addition, for any Lipschitz continuous function* $\zeta(\cdot)$ *on* Θ, $\|\frac{1}{T}\sum_{t=1}^T \zeta(\hat{\boldsymbol{\theta}}_n^{(t)}) - \zeta(\boldsymbol{\theta}^*)\| \xrightarrow{p} 0$.

As implied by Theorems 3.1.1-3.1.3 and Lemma 3.1.2, $\hat{\boldsymbol{\theta}}_n^{(t)}$ asymptotically converges to a global optimum of the KNN. For each $\hat{\boldsymbol{\theta}}_n^{(t)}$, when making predictions, one can simply calculate the output in (3.5) by ignoring the auxiliary noise, i.e., treating $\hat{\boldsymbol{\theta}}_n^{(t)}$ as the weights of a KNN. In this way, K-StoNet can be viewed as a tool for training the KNN, although it means more than that.

3.1.4 Hyperparameter Setting

As mentioned previously, DNN is often over parameterized to avoid getting trapped into a poor local minimum. In contrast, as implied by Theorems 3.1.1-3.1.3, the local minimum trap is not an issue to K-StoNet any more. This, together with the universal approximation property of K-StoNet and the parsimony principle of statistical modeling, suggests that a small K-StoNet might work well for complex problems. As shown in Section 3.3, the K-StoNet with a single hidden layer and a small number of hidden units works well for many complex datasets.

Other than the network structure, the performance of K-StoNet also depends on the network hyperparameters as well as the hyperparameters introduced by the IRO algorithm. The former include C_n, ε_n and $\sigma_{n,k}$'s for $k = 2, \ldots, h+1$. The latter include the learning rates and iteration number used in HMC backward imputation and the regularization parameters used in solving the optimizations (3.14) and (3.15). The hyperparameters C_n, ε_n and $\sigma_{n,k}$'s control the variation of the latent variables and thus the variation of $\boldsymbol{\theta}_n^{(T)}$ by the theory developed in [33] and [103]. In general, setting the latent variables to have slightly large variations can facilitate the convergence of the training process. On the other hand, as required by Assumption 3.1.2-(v), we need to control the variations of the latent variables sufficiently small for ensuring the convergence of K-StoNet to a global minimum of the corresponding KNN by noting the stochastic optimization nature of the IRO algorithm. Assumption 3.1.2-(v) provides a clue for setting the network hyperparameters. Here we would like to note that when $1/C_n$ and $\sigma_{n,\mathrm{i}}^2$'s are set to be very small, to ensure the stability of the algorithm, we typically need to adjust the learning rate $\epsilon_{t,\mathrm{i}}$'s to be very small as well such that their effects on the drift term of (3.13) can be canceled or partially canceled. Meanwhile, to compensate the negative effect of the reduced learning rate on the mobility of the Markov chain, we need to lengthen the MCMC iterations, i.e., increasing the value of t_{HMC}, appropriately. Finally, we note that setting $\sigma_{n,\mathrm{i}}$'s in the monotonic pattern $\sigma_{n,h+1} \geq \sigma_{n,h} \geq \cdots \geq \sigma_{n,2} \geq 1/C_n$ is generally unnecessary, as long as their values have been in a reasonable range.

In our experience, the performance of K-StoNet is not very sensitive to these hyperparameters as long as they are set in an appropriate range. As shown in Section 3.5, which collects all parameter settings of K-StoNet used in this chapter, many examples share the same parameter setting.

3.2 Illustrative Examples

This section contains two examples. The first example demonstrates that K-StoNet indeed avoids local traps in training, and the second example demonstrates the performance of K-StoNet in the large-n-small-p scenario that DNN typically works in. K-StoNet is compared with DNN and KNN. For the KNN, the kernel representer given by equation (3.4) is used as the first hidden layer, and the kernel is set to be the same as that used by K-StoNet.

3.2.1 A full row rank example

The dataset was generated from a two-hidden layer neural network with structure 1000-5-5-1. The input variables $x_1, \ldots, x_{1000}$ were generated by independently simulating the variables $\mathrm{e}, z_1, \ldots, z_{1000}$ from the standard Gaussian distribution and then setting $x_\mathrm{i} = \frac{\mathrm{e}+z_\mathrm{i}}{\sqrt{2}}$. In this way, all the input variables are mutually correlated with a correlation coefficient of 0.5. The response variable was generated by setting

$$\boldsymbol{y} = \boldsymbol{w}_3 \tanh(\boldsymbol{w}_2 \tanh(\boldsymbol{w}_1 \boldsymbol{x})) + \boldsymbol{\epsilon}, \tag{3.16}$$

where $\boldsymbol{w}_1 \in \mathbb{R}^{5\times 1000}$, $\boldsymbol{w}_2 \in \mathbb{R}^{5\times 5}$ and $\boldsymbol{w}_3 \in \mathbb{R}^{1\times 5}$ represent the weights at different layers of the neural network, $\tanh(\cdot)$ is the hyperbolic tangent function, and the random error $\boldsymbol{\epsilon} \sim N(0,1)$. Each elements of $\boldsymbol{w}_\mathrm{i}$'s was randomly sampled from the set $\{-2,-1,1,2\}$. The full dataset consisted of 1000 training samples and 1000 test samples.

We first refit the model (3.16) using SGD. Since the training samples form a full row rank matrix of size $n = 1000$ by $p = 1001$ (including the bias term), SGD will not get trapped into a local minimum by Lemma 3.1.1. SGD was run for 2000 epochs with a mini-batch size of 100 and a constant learning rate of 0.005. Figure 3.2 (upper panel) indicates that SGD indeed converges to a global optimum.

For K-StoNet, we tried a model with one hidden layer and 5 hidden units. The model was trained by IRO for 40 epochs. Since all training samples were used at each iteration, an iteration is equivalent to an epoch for K-StoNet. We also tried a KNN model for this example, which has the same structure as K-StoNet. The KNN was trained using SGD with a constant learning rate of 0.005 for 2000 epochs. Figure 3.2 (upper panel) compares the training and testing MSE paths of the three models. It shows that K-StoNet converges to the global optimum in a few epochs; while DNN needs over 100 epochs, and KNN needs even more. More importantly, K-StoNet is less bothered by over-fitting, whose prediction performance is stable after convergence has been reached. However, the DNN tends to be over fitted, whose prediction becomes worse and worse as training goes on. The KNN is more stable than DNN in prediction, but worse than K-StoNet. As discussed in Section 3.1.2, the KNN with the kernel representer for the first hidden layer is not equivalent to K-StoNet in general. This experiment further demonstrates the importance of the stochastic structure introduced in K-StoNet.

To explore the loss surface of the regularized DNN, we re-trained the true DNN model (3.16) using SGD with a Lasso penalty ($\lambda = 0.1$) imposed on all the weights. SGD was run for 500 epochs with a mini-batch size of 100 and a constant learning rate of 0.005. The run was repeated for 10 times. For comparison, K-StoNet was also retained for 10 times. Their convergence paths were shown in the lower panel of Figure 3.2. The comparison shows that the regularized DNN might suffer from local traps (different runs converged to different MSE values), while K-StoNet does not although its RO step also involves penalty terms. According to the theory developed in [33], the convergence of the IRO algorithm requires a consistent estimate of $\boldsymbol{\theta}_*^{(t)}$ to be obtained at each RO step and an appropriate penalty term is allowed for obtaining the consistent estimate. For K-StoNet, to ensure a consistent estimate to be obtained at each parameter updating step, we impose a L_2-penalty on the SVR layer and a Lasso penalty on the regression layers. For both of them, the resulting loss functions are convex, and the corresponding consistent (optimal) estimates are uniquely determined.

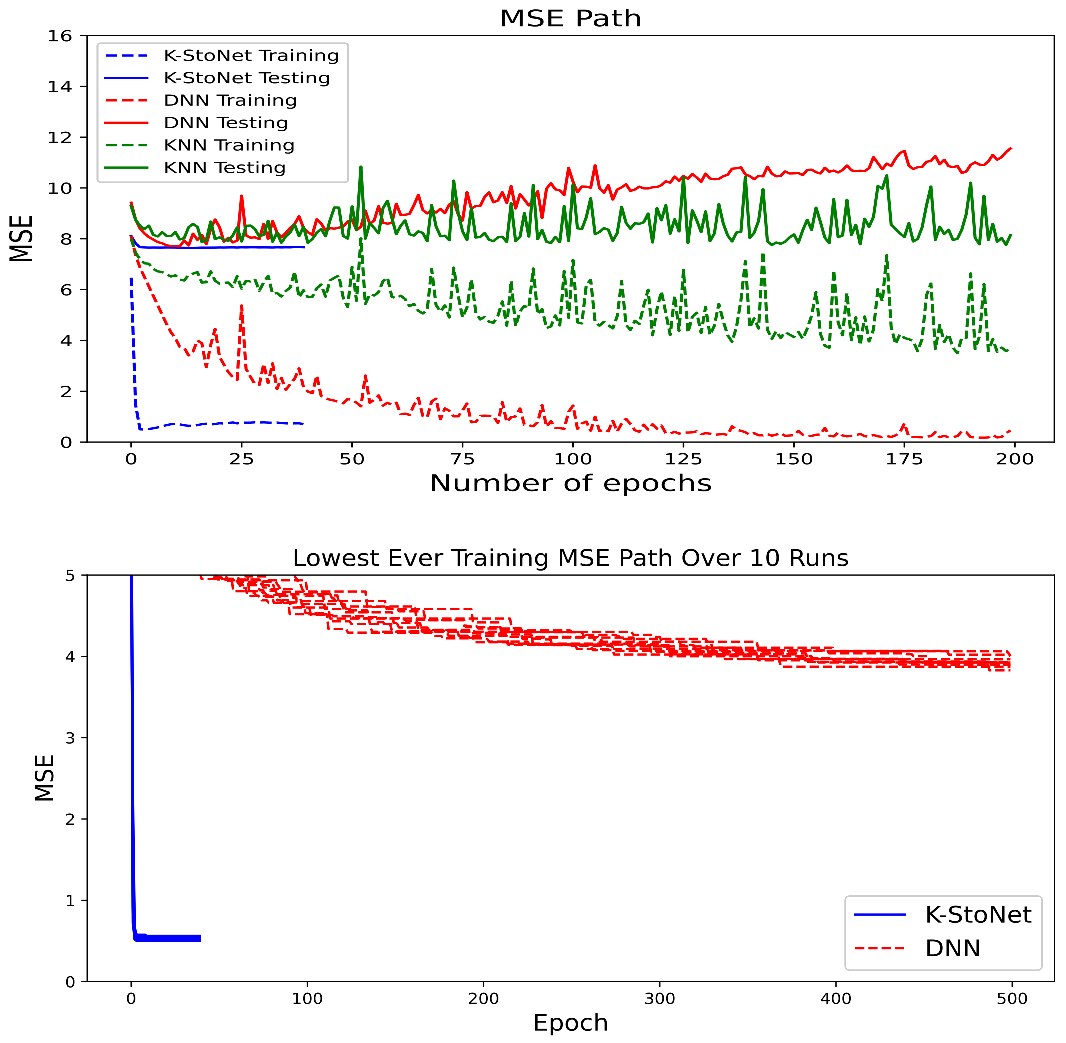


Figure 3.2. *Upper Panel*: paths of the mean squared error (MSE) produced by K-StoNet and an unregularized DNN for one simulated dataset; and *Lower Panel*: best MSE (by the current epoch) produced by SGD for a regularized DNN and K-StoNet over 10 runs.

Then, by Theorems 3.1.1–3.1.3 and Lemma 3.1.2, the convergence of K-StoNet to the global optimum is asymptotically guaranteed.

In the previous example, the data was generated from a DNN model. Even working with the true structure, DNN is still inferior to K-StoNet in training and prediction. To further demonstrate the advantage of K-StoNet, we generated a dataset from a KNN model. The dataset consisted of 5000 training samples, where the input variables $\boldsymbol{x} \in \mathbb{R}^5$ with each component being a standard Gaussian random variable and a mutual correlation coefficient of 0.5. Let $\boldsymbol{k} = (K(\boldsymbol{x}^{(1)}, \boldsymbol{x}), \ldots, K(\boldsymbol{x}^{(5000)}, \boldsymbol{x}))^T \in \mathbb{R}^{5000}$, where $K(\cdot, \cdot)$ is the RBF kernel and $\boldsymbol{x}^{(\mathrm{i})}$ denotes the ith training sample. The response variable was generated by

$$\boldsymbol{y} = \boldsymbol{w}_2 \tanh(\boldsymbol{w}_1 \boldsymbol{k} + \boldsymbol{b}_1) + \boldsymbol{b}_2 + \boldsymbol{\epsilon},$$

where $\boldsymbol{w}_2 \in \mathbb{R}^{1\times 5}$, $\boldsymbol{w}_1 \in \mathbb{R}^{5\times 5000}$, $\boldsymbol{b}_1 \in \mathbb{R}^5$, $\boldsymbol{b}_2 \in \mathbb{R}$, and $\boldsymbol{\epsilon} \sim N(0,1)$. The components of $\boldsymbol{w}_2$ and $\boldsymbol{b}_2$ were randomly generated from $N(0,1)$, and $\boldsymbol{w}_1$ and $\boldsymbol{b}_1$ were the dual parameters of five SVR models with the above training samples as input and some vectors randomly generated from $N(0, I_5)$ as response. We set $C = 5$ and $\epsilon = 0.01$ for the SVR model. We also generated another 5000 samples from the same model as test data. Then we modeled the data by K-StoNet with one hidden layer, for which we tried the cases with 5 hidden units and 10 hidden units and set $C = 5$ and $\epsilon = 0.01$ for each SVR. For comparison, we tried a KNN with 5 hidden units, a DNN with one hidden layer and 50 hidden units, and a DNN with 3 hidden layers and 50 hidden units on each layer. Figure 3.3 shows the training and testing paths of the five models. For this example, K-StoNet achieved a training MSE about 1.0 and significantly outperformed the DNN and KNN models in prediction.

3.2.2 A measurement error example

This example mimics the typical scenario under which DNN works. We generated 500 training samples and 500 test samples from a nonlinear regression: for each sample $(Y, \boldsymbol{X})$, where $Y \in \mathbb{R}$ and $\boldsymbol{X} = (X_1, \ldots, X_5) \in \mathbb{R}^5$. The explanatory variables $X_1, \ldots, X_5$ were generated such that each follows the standard Gaussian distribution, while they are mutually

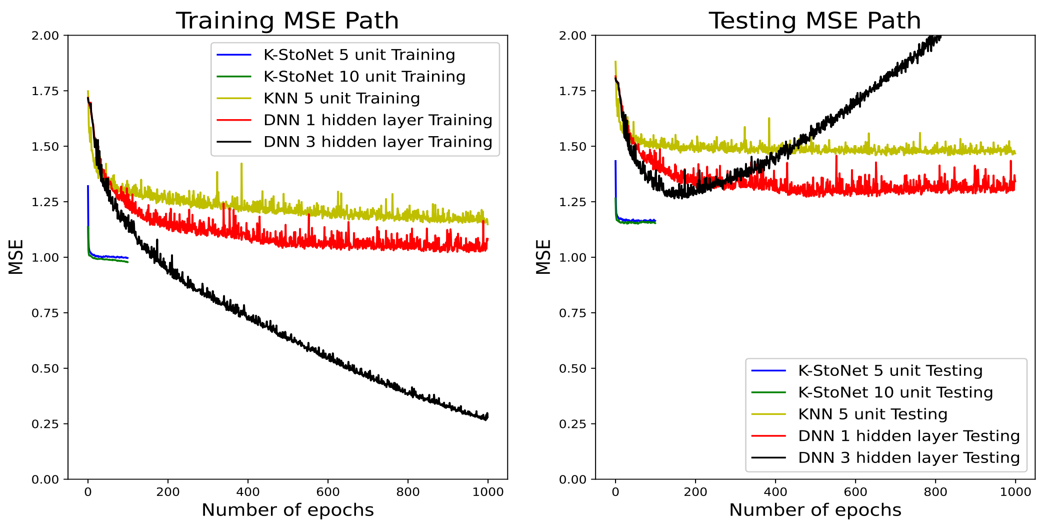


Figure 3.3. MSE paths produced by two K-StoNets, one KNN, and two DNNs for the data generated from a KNN model: the left plot is for training and the right plot is for testing.

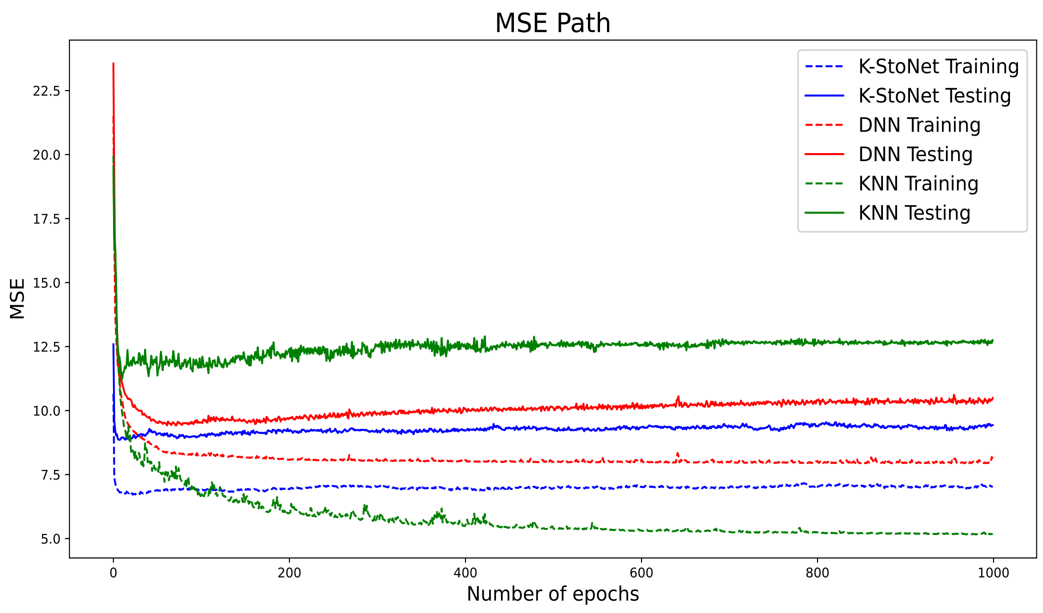
correlated with a correlation coefficient of 0.5. The response variable was generated from the nonlinear regression

$$Y = \frac{5X_2}{1 + X_1^2} + 5\sin(X_3X_4) + 2X_5 + \epsilon,$$

where $\epsilon \sim N(0, 1)$. Then each explanatory variable was perturbed by adding a random measurement error independently drawn from $N(0, 0.5)$.

We modeled the data using two different K-StoNets, one with 1-hidden layer and 5 hidden units, and the other with 3-hidden layers and 20 hidden units on each hidden layer. Both models were trained by IRO for 1000 epochs. For comparison, we also modeled the data by KNNs and DNNs with the same structures as the K-StoNets. The KNNs and DNNs were trained by SGD with momentum for 1000 epochs with a minibatch size of 100, a constant learning rate of 0.005, and a momentum decay factor of 0.9. As shown in Figure 3.4, the 1-hidden layer DNN and KNN perform stably in both training and testing, while the 3-hidden layer DNN and KNN are obviously over-fitted. Compared to the DNN and KNN, K-StoNet is resistant to over-fitting, even when an overly large model is employed.

Finally, we explored the sparsity of the SVR layer by varying the value of ε defined in (3.6). Table 3.1 shows the number of support vectors selected by the two K-StoNets, together with their training and test errors, at different values of ε. It implies that the sparsity of K-StoNet can be controlled by ε, a larger ε leading to less support vectors. However, the two K-StoNet models show different sensitivities to ε. The 3-hidden layer K-StoNet has a higher representation power and is more flexible; it can achieve relatively low training error with a large number of connections, and is more sensitive to ϵ. When ϵ increases from 0.01 to 0.09, it changes from an overfitted model to an underfitted model. Correspondingly, the training MSE increases, while the test MSE decreases in the beginning and then starts to increase. In contrast, for the 1-hidden layer K-StoNet, its representation power is limited, and it is less flexible and thus less sensitive to ϵ. It led to about the same models with different choices of ϵ (with similar training and test errors). The training and test errors varied slightly with ϵ, as different sets of support vectors were used for different choices of ϵ. In general, the set



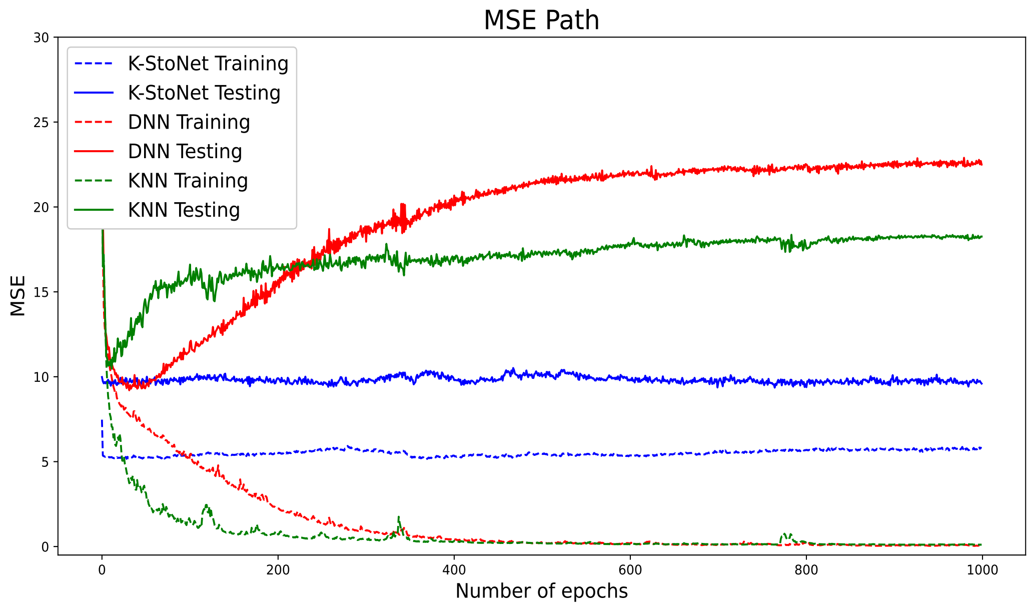


Figure 3.4. MSE paths produced by K-StoNets and DNNs: (upper) one-hidden-layer networks; (lower) three-hidden-layer networks.

of support vectors used for a large ϵ is not nested to that for a small ϵ. Therefore, a smaller ϵ does not necessarily lead to a smaller training error.

Table 3.1. Performance of the K-StoNet model with different values of ε, where the model was evaluated at the last iteration, #SV represents the average number of support vectors selected by the SVRs at the first hidden layer, and the number in the parentheses represents the standard deviation of the average.

	1-Hidden Layer			3-Hidden Layer		
ε	#SV	Train_MSE	Test_MSE	#SV	Train_MSE	Test_MSE
0.01	473.6(9.22)	6.6786	9.0852	409.0(6.419)	4.6552	10.1387
0.02	428.6(18.73)	6.7382	9.0387	320.2(13.85)	4.7257	9.9422
0.03	416.2(16.59)	6.6917	9.0531	235.8(7.94)	4.9850	9.8192
0.04	381.0(29.75)	6.7301	9.0742	165.4(8.96)	5.2578	9.3949
0.05	361.4(45.85)	6.5158	9.3358	113.2(7.83)	5.6594	9.0195
0.06	342.6(28.35)	6.5041	9.1498	69.6(4.50)	6.1869	8.7551
0.07	347.2(13.39)	6.4878	9.1893	37.8(3.60)	6.8884	9.0565
0.08	313.6(38.52)	6.5083	9.0639	17.6(3.61)	7.8616	9.5882
0.09	317.0(14.59)	6.4130	9.0330	16.2(16.45)	9.1655	9.9940
0.1	290.2(40.45)	6.4676	9.0308	8.2(13.47)	10.5946	10.7611

3.3 Real Data Examples

This section shows that a small K-StoNet can work well for a variety of problems. The example in Section 3.3.1 has a high dimension, which represents typical problems that support vector machine/regression works on. The examples in Sections 3.3.2 and 3.3.3 have large training sample sizes, which represent typical problems that the DNN works on. The examples in Section 3.3.4 represent more real world problems, with which we explore the prediction performance of K-StoNet.

3.3.1 QSAR Androgen Receptor

The QSAR androgen receptor dataset is available at the UCI machine learning repository, which consists of 1024 binary attributes (molecular fingerprints) used to classify 1687 chemicals into 2 classes (binder to androgen receptor/positive, non-binder to androgen receptor /negative), i.e. $n = 1607, p = 1024$. The experiment was done in a 5-fold cross-validation.

In each fold of the experiment, we modeled the data by a K-StoNet with one hidden layer and 5 hidden units, and trained the model by IRO for 40 epochs. The prediction was computed by averaging over the models generated in the last 10 epochs. For comparison, support vector machine (SVM), KNN and DNN were applied to this example. For SVM, we employed the RBF kernel with $C = 1$. For KNN, we used the same structure as K-StoNet. For DNN, we tried two network structures, 1024-5-1 and 1024-10-5-1, which are called DNN_one_layer and DNN_two_layer, respectively. Each of the KNN and DNN models were trained by SGD for 1000 epochs with a mini-batch size of 32 and a constant learning rate of 0.001. The weights of the DNNs were subject to the LASSO penalty with the regularization parameter $\lambda = 1e - 4$.

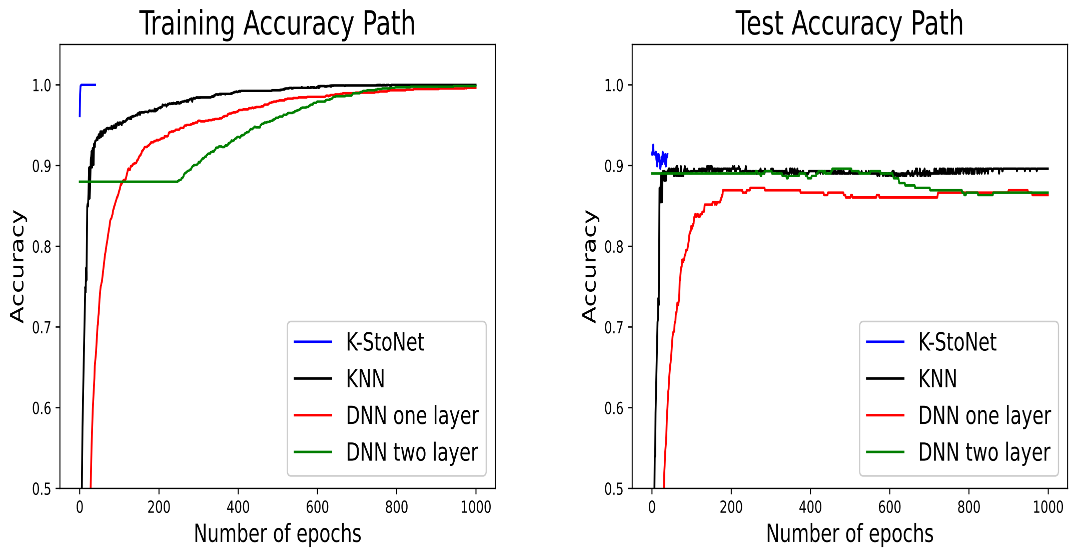


Figure 3.5. Training and prediction accuracy paths (along with epochs) produced by K-StoNet, KNN and DNN in one fold of the cross-validation experiment for the QSAR androgen receptor data.

Figure 3.5 compares the training and prediction accuracy paths produced by K-StoNet, KNN and DNN in one fold of the experiment. Table 3.2 summarizes the training and prediction accuracy produced by K-StoNet, SVM, KNN and DNN over the five folds. In summary, K-StoNet converges very fast to the global optimum with the training accuracy close to 1, and is less bothered by the over-fitting issue. In contrast, the KNN and DNN

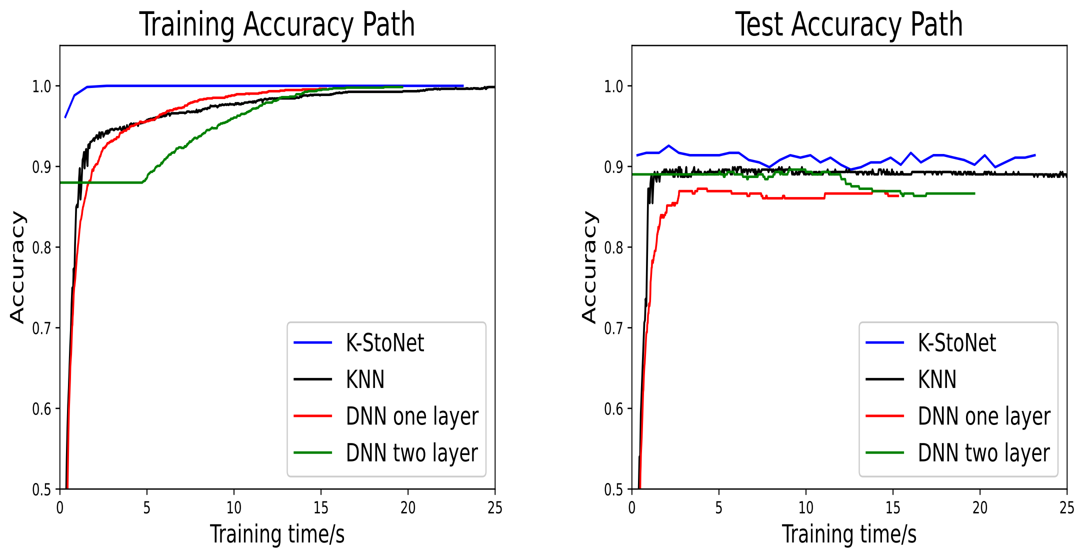


Figure 3.6. Training and prediction accuracy paths (along with computational time) produced by K-StoNets, KNN and DNNs in one fold of the cross-validation experiment for the QSAR androgen receptor data.

models took more epochs to converge and predicted less accurately than K-StoNet. SVM is inferior to K-StoNet in both training and prediction.

Since each iteration of the IRO algorithm involves imputing latent variables and solving a series of SVR/linear regressions, it is more expensive than a single gradient update step used in DNN training. To compare their computational efficiency, we include an accuracy versus time plot in Figure 3.6, which indicates that K-StoNet took less computational time than KNN and DNN to achieve the same training/prediction accuracy.

3.3.2 MNIST Data

The MNIST [104] is a benchmark dataset in machine learning. It consists of 60,000 images for training and 10,000 images for testing. We modeled the data by a K-StoNet with one hidden layer, 20 hidden units, and the softplus activation function, We trained the model by IRO for 6 epochs. For comparison, we trained a standard LeNet-300-100 model [104] by Adam [105] with default parameters for 300 epochs, a constant learning rate of 0.001, and a mini-batch size of 128. Figure 3.7 shows the training paths of two models. Both models

Table 3.2. Training and prediction accuracy(%) for QSAR androgen receptor data, where "T" and "P" denote the training and prediction accuracy, respectively.

Method		Split 1	Split 2	Split 3	Split 4	Split 5	Average
K-StoNet	T	99.93	99.85	99.85	100	100	99.926
	P	88.43	93.47	90.50	90.50	91.99	90.978
SVM	T	97.11	96.89	97.11	97.11	97.19	97.082
	P	89.32	90.80	87.83	89.02	92.28	89.850
KNN	T	99.93	100	98.44	99.93	99.93	99.646
	P	88.72	93.17	89.32	90.50	91.10	90.562
DNN_one_layer	T	99.70	99.63	99.85	99.78	99.85	99.762
	P	85.16	88.72	89.32	86.35	88.13	87.536
DNN_two_layer	T	99.93	99.93	99.93	100	100	99.958
	P	86.94	88.13	88.13	86.05	88.72	87.596

can achieve 100% training accuracy. LeNet-300-100 achieved 98.38% test accuracy, while K-StoNet achieved 98.87% test accuracy (at the 3rd iteration) without data augmentation being used in training!

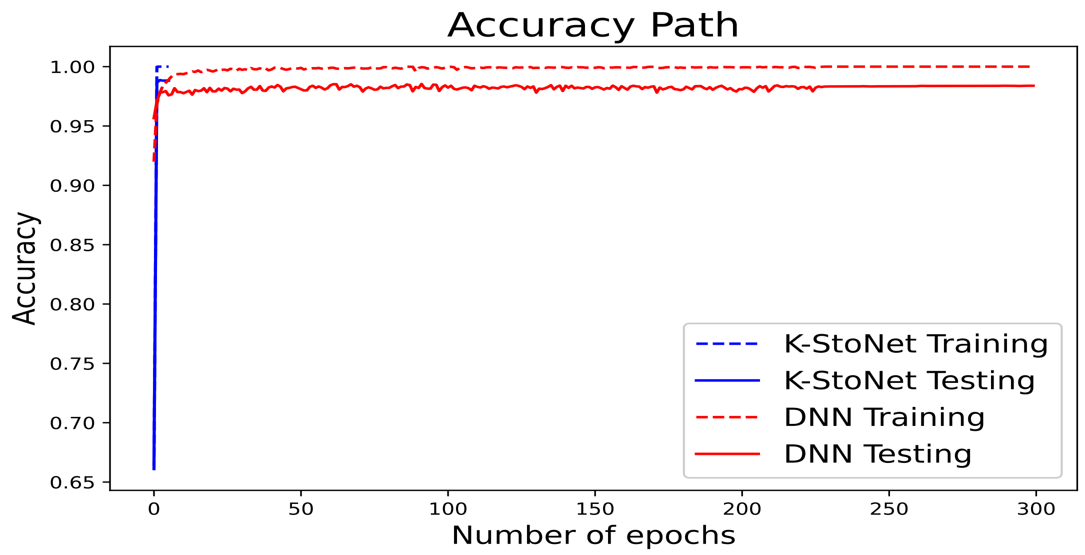


Figure 3.7. Training and test accuracy versus epochs produced by K-StoNet and DNN (LeNet-300-100) for the MNIST data, where K-StoNet achieved a prediction accuracy of 98.87%, and LeNet-300-100 achieved a prediction accuracy of 98.38%.

3.3.3 CoverType Data

The CoverType data is available at the UCI machine learning repository. It consisted of $n = 581,012$ samples with $p = 54$ attributes, which were collected for classification of forest cover types from cartographic variables. This dataset has an extremely large sample size, which represents a typical problem that the DNN works on. We used half of the samples for training and the other half for testing. The experiments were repeated for thee times.

We modeled the data by a K-StoNet with one hidden layer and 50 hidden units, and trained the model by the IRO algorithm for 2 epochs. For comparison, we also modeled the data by a 2-hidden-layer DNN with 1000 nodes on the first hidden layer and 50 nodes on the second hidden layer. We trained the DNN model by SGD with momentum, where a mini-batch size of 500, a constant learning rate of 0.01 and a momentum decay factor of

0.9 were used. The numerical results were summarized in Table 3.3, which indicates that K-StoNet outperforms DNN in both training and prediction for this example.

Table 3.3. Training and prediction accuracy(%) for CoverType, where "T" and "P" denote the training and prediction.

Method		Run 1	Run 2	Run 3	Average
K-StoNet	T	99.22	99.32	99.33	99.29
	P	94.21	94.23	94.26	94.23
DNN	T	98.20	98.11	98.07	98.13
	P	94.11	94.06	94.04	94.07

3.3.4 More UCI Datasets

As shown by the above examples, K-StoNet can converge in only a few epochs and is less bothered by overfitting, and its prediction accuracy is typically similar or better than the best one that DNN achieved. In order to achieve the best prediction accuracy, DNN often needs to be trained with tricks such as early stopping or Dropout [26], which lack the theoretical guarantee for the down-stream statistical inference. In contrast, K-StoNet possesses the theoretical guarantee to asymptotically converge to the global optimum and enables the prediction uncertainty easily assessed (see Section 3.4). This subsection compares K-StoNet with Dropout in prediction on more real world examples, 10 datasets taken at the UCI machine learning repository.

Following the setting of [106], we randomly split each dataset into training and test sets with 90% and 10% of its samples, respectively. The random split was repeated for 20 times. The average prediction accuracy and its standard deviation were reported. As in [106], for the largest two datasets *Protein Structure* and *Year Prediction MSD*, the random splitting was done five times and one time, respectively. The baseline results were taken from [106] and [107]. The neural network model used there had one hidden layer and 50 hidden units for all datasets except for the largest two. For the largest two datasets, the neural network model had one hidden layer and 100 hidden units. The K-StoNet model we used had one hidden layer with 5 hidden units and the softplus activation function for all datasets except for the largest one. For the largest dataset, K-StoNet had one hidden layer with 50 hidden units. Other parameter settings were given in Section 3.5. The results were summarized in Table 3.4, which indicates that K-StoNet generally outperforms Dropout in prediction.

For a thorough comparison, the KNN has also been implemented for the UCI datasets except for two large ones, “Protein Structure” and “Year Prediction MSD”. For these two datasets, the training sample size n is too large, making the gram matrix hard to handle. For the same reason, it is not included in the comparisons for the MNIST and CoverType data examples, either. For the KNN, the use of minibatch data is not very helpful when n is large, as there are still n kernels $(K(\boldsymbol{x}^{(1)}, \boldsymbol{x}^*), K(\boldsymbol{x}^{(2)}, \boldsymbol{x}^*), \ldots, K(\boldsymbol{x}^{(n)}, \boldsymbol{x}^*))$ we need to evaluate for each sample $\boldsymbol{x}^*$ in the minibatch. For the other datasets, the KNN was run with the same network structure as for the K-StoNet. The detailed parameter settings were given in Section 3.5. The results are summarized in Table 3.4, which indicates that the KNN is generally inferior to K-StoNet in prediction.

Table 3.4. Average test RMSE (and its standard error) by variational inference (VI, [108]), probabilistic back-propagation (PBP, [107]), dropout (Dropout, [106]), SGD via back-propagation (BP), and KNN, where N denotes the dataset size and p denotes the input dimension. For each dataset, the boldfaced values are the best result or the second best result if it is insignificantly different from the best one according to a t-test with a significance level of 0.05.

Dataset	N	p	VI	PBP	Dropout	BP	KNN	K-StoNet
Boston Housing	506	13	4.32 ±0.29	3.014 ±0.1800	**2.97 ±0.19**	3.228 ±0.1951	4.196 ±0.069	**2.987 ±0.0227**
Concrete Strength	1,030	8	7.19 ±0.12	5.667 ±0.0933	**5.23 ±0.12**	5.977 ±0.2207	6.962 ±0.062	**5.261 ±0.0265**
Energy Efficiency	768	8	2.65 ±0.08	1.804 ±0.0481	1.66 ±0.04	**1.098 ±0.0738**	1.942 ±0.030	1.301 ±0.015
Kin8nm	8,192	8	0.10 ±0.00	0.098 ±0.0007	0.10 ±0.00	0.091 ±0.0015	0.0917 ±0.0002	**0.0747 ±0.0003**
Naval Propulsion	11,934	16	0.01 ±0.00	0.006 ±0.0000	0.01 ±0.00	**0.001 ±0.0001**	0.0151 ±0.0001	**0.00098 ±0.0001**
Power Plant	9,568	4	4.33 ±0.04	4.124 ±0.0345	**4.02 ±0.04**	4.182 ±0.0402	4.033 ±0.010	**3.952 ±0.003**
Protein Structure	45,730	9	4.84 ±0.03	4.732 ±0.0130	4.36 ±0.01	4.539 ±0.0288	na	**3.856 ±0.005**
Wine Quality Red	1,599	11	0.65 ±0.01	0.635 ±0.0079	**0.62 ±0.01**	0.645 ±0.0098	0.675 ±0.004	**0.6214 ±0.0008**
Yacht Hydrodynamics	308	6	6.89 ±0.67	**1.015 ±0.0542**	1.11 ±0.09	1.182 ±0.1645	7.5334 ±0.0893	**0.8560±0.0795**
Year Prediction MSD	515,345	90	9.034 ±na	8.879 ±na	**8.849 ±na**	8.932 ±na	na	8.881 ±na

3.4 Prediction Uncertainty Quantification with K-StoNet

3.4.1 A Recursive Formula for Uncertainty Quantification

The prediction uncertainty of the K-StoNet can be easily assessed with the variance decomposition formula (as known as Eve's law) based on the asymptotic normality theory. More precisely, we can first calculate the variance for the output of the first hidden layer based on the existing theory of SVR [109], then calculate the variance for the output of the second hidden layer based on Eve's law and the theory of linear models, and continue this process till the output layer is reached. For the case that normal regression was done at layer $h+1$ and no penalties was used in solving the optimization (3.15), the calculation is detailed as follows.

Let $\mathbb{Y}_{\rm i}^{(t)} \in \mathbb{R}^{n\times m_{\rm i}}$, $\rm i = 1, 2, \ldots, h$, denote the matrices of latent variables imputed at iteration t, which leads to the updated parameter estimate $\boldsymbol{\theta}_n^{(t)}$. Let $\boldsymbol{z}$ denote a test sample. For each layer $\rm i \in \{1, 2, \ldots, h+1\}$, let $\boldsymbol{Z}_{\rm i}^{(t)} \in \mathbb{R}^{m_{\rm i}}$ denote the output of the KNN (with the parameter $\boldsymbol{\theta}_n^{(t)}$) at layer i; and let $\boldsymbol{\mu}_{\rm i}^{(t)}$ and $\Sigma_{\rm i}^{(t)}$ denote the mean and covariance matrix of $\boldsymbol{Z}_{\rm i}^{(t)}$, respectively. Assume that $\boldsymbol{Z}_{\rm i}^{(t)}$'s are all multivariate Gaussian, which will be justified below. Then, for any layer $\rm i \in \{2, \ldots, h+1\}$, by Eve's law, we have

$$\begin{aligned}
\Sigma_{\rm i}^{(t)} &= \mathbb{E}(\mathrm{Var}(\boldsymbol{Z}_{\rm i}^{(t)}|\boldsymbol{Z}_{\rm i-1}^{(t)})) + \mathrm{Var}(\mathbb{E}(\boldsymbol{Z}_{\rm i}^{(t)}|\boldsymbol{Z}_{\rm i-1}^{(t)}))\\
&= \mathbb{E}\{(\psi(\boldsymbol{Z}_{\rm i-1}^{(t)}))^T[(\psi(\mathbb{Y}_{\rm i-1}^{(t)}))^T\psi(\mathbb{Y}_{\rm i-1}^{(t)})]^{-1}\psi(\boldsymbol{Z}_{\rm i-1}^{(t)})\}\mathrm{diag}\{\sigma_{\rm i,1}^{2(t)}, \ldots, \sigma_{{\rm i},m_{\rm i}}^{2(t)}\} + \mathrm{Var}(\tilde{\boldsymbol{w}}_{\rm i-1}^{*}\psi(\boldsymbol{Z}_{\rm i-1}^{(t)}))\\
&= \{tr([(\psi(\mathbb{Y}_{\rm i-1}^{(t)}))^T\psi(\mathbb{Y}_{\rm i-1}^{(t)})]^{-1}\mathrm{Var}(\psi(\boldsymbol{Z}_{\rm i-1}^{(t)}))) + (\mathbb{E}(\psi(\boldsymbol{Z}_{\rm i-1}^{(t)})))^T[(\psi(\mathbb{Y}_{\rm i-1}^{(t)}))^T\psi(\mathbb{Y}_{\rm i-1}^{(t)})]^{-1}(\mathbb{E}(\psi(\boldsymbol{Z}_{\rm i-1}^{(t)})))\}\\
&\quad \times diag\{\sigma_{\rm i,1}^{2(t)}, \ldots, \sigma_{{\rm i},m_{\rm i}}^{2(t)}\} + \tilde{\boldsymbol{w}}_{\rm i-1}^{*}\mathrm{Var}(\psi(\boldsymbol{Z}_{\rm i-1}^{(t)}))(\tilde{\boldsymbol{w}}_{\rm i-1}^{*})^T,
\end{aligned}$$

where $\tilde{\boldsymbol{w}}_{\rm i-1}^{*} = \mathbb{E}(\tilde{\boldsymbol{w}}_{\rm i-1}^{(t)})$ and $\sigma_{\rm i,j}^{2(t)}$'s are unknown. Let $\boldsymbol{\mu}_{\rm i-1}^{(t)} = (\mu_{\rm i-1,1}^{(t)}, \ldots, \mu_{{\rm i-1},m_{\rm i-1}}^{(t)})^T$ and $D_{\psi'}(\boldsymbol{\mu}_{\rm i-1}^{(t)}) = \mathrm{diag}\{\psi'(\mu_{\rm i-1,1}^{(t)}), \ldots, \psi'(\mu_{{\rm i-1},m_{\rm i-1}}^{(t)})\}$. By the first order Taylor expansion, it is easy to derive that

$$\begin{aligned}
\mathbb{E}(\psi(\boldsymbol{Z}_{\rm i-1}^{(t)})) &\approx \psi(\boldsymbol{\mu}_{\rm i-1}^{(t)}),\\
\mathrm{Var}(\psi(\boldsymbol{Z}_{\rm i-1}^{(t)})) &\approx D_{\psi'}(\boldsymbol{\mu}_{\rm i-1}^{(t)})\Sigma_{\rm i-1}^{(t)}D_{\psi'}(\boldsymbol{\mu}_{\rm i-1}^{(t)}).
\end{aligned}$$

We suggest to estimate $\tilde{\boldsymbol{w}}_{\mathrm{i}-1}^{*}$ by $\tilde{\boldsymbol{w}}_{\mathrm{i}-1}^{(t)}$, estimate $\boldsymbol{\mu}_{\mathrm{i}-1}^{(t)}$ by $\boldsymbol{Z}_{\mathrm{i}-1}^{(t)}$, and estimate $\sigma_{\mathrm{i,j}}^{2(t)}$ by its OLS estimator from the corresponding multiple regression. This leads to the following recursive formula for covariance estimation:

$$\begin{aligned}\widehat{\Sigma}_{\mathrm{i}}^{(t)} \approx \{&tr([(\psi(\mathbb{Y}_{\mathrm{i}-1}^{(t)}))^T\psi(\mathbb{Y}_{\mathrm{i}-1}^{(t)})]^{-1}D_{\psi'}(\boldsymbol{Z}_{\mathrm{i}-1}^{(t)})\widehat{\Sigma}_{\mathrm{i}-1}^{(t)}D_{\psi'}(\boldsymbol{Z}_{\mathrm{i}-1}^{(t)}))\\ &+(\psi(\boldsymbol{Z}_{\mathrm{i}-1}^{(t)}))^T[(\psi(\mathbb{Y}_{\mathrm{i}-1}^{(t)}))^T\psi(\mathbb{Y}_{\mathrm{i}-1}^{(t)})]^{-1}\psi(\boldsymbol{Z}_{\mathrm{i}-1}^{(t)})\}\mathrm{diag}\{\hat{\sigma}_{\mathrm{i},1}^{2(t)},\ldots,\hat{\sigma}_{\mathrm{i},m_{\mathrm{i}}}^{2(t)}\}\\ &+\tilde{\boldsymbol{w}}_{\mathrm{i}-1}^{(t)}D_{\psi'}(\boldsymbol{Z}_{\mathrm{i}-1}^{(t)})\widehat{\Sigma}_{\mathrm{i}-1}^{(t)}D_{\psi'}(\boldsymbol{Z}_{\mathrm{i}-1}^{(t)})(\tilde{\boldsymbol{w}}_{\mathrm{i}-1}^{(t)})^T,\end{aligned} \tag{3.17}$$

for $\mathrm{i}=2,3,\ldots,h+1$. For the SVR layer, the asymptotic normality of $\boldsymbol{Z}_1^{(t)}$ can be justified based on [109], which gives a Bayesian interpretation to the classical SVR with an ε-intensive loss function. Let $f(\boldsymbol{x})=\boldsymbol{\beta}^T\phi(\boldsymbol{x})+\boldsymbol{b}$ denote a SVR function, and let $\{(\boldsymbol{x}^{(1)},y^{(1)}),\ldots,(\boldsymbol{x}^{(n)},y^{(n)})\}$ denote training samples.

In [109], the authors treated $(f(\boldsymbol{x}^{(1)}),f(\boldsymbol{x}^{(2)}),\ldots,f(\boldsymbol{x}^{(n)}))$ as a random vector subject to a functional Gaussian process prior, and showed that the posterior of $f(\boldsymbol{z})$ can be approximated by a Gaussian distribution with mean $f^*(\boldsymbol{z})$ and variance

$$\sigma_f^2(\boldsymbol{z})=K_{\boldsymbol{z},\boldsymbol{z}}-K_{X_M,\boldsymbol{z}}^TK_{X_M,X_M}^{-1}K_{X_M,\boldsymbol{z}}, \tag{3.18}$$

where $f^*(\cdot)$ denotes the optimal regression function fitted by SVR, $X_M=\{\boldsymbol{x}^{(s)}:|y^{(s)}-f^{(t)}(\boldsymbol{x}^{(s)})|=\varepsilon\}$ denotes the set of marginal vectors, and $K_{A,B}$ denotes a kernel matrix with elements formed by variables in A versus variables in B. By this result, conditioned on the training samples, $\boldsymbol{Z}_1^{(t)}$ is approximately Gaussian with the covariance matrix given by $\mathrm{diag}\{\sigma_{f_1}^{2(t)}(\boldsymbol{z}),\ldots,\sigma_{f_{m_1}}^{2(t)}(\boldsymbol{z})\}$.

For the output and other hidden layers, we can restrict $m_{\mathrm{i}}\prec\sqrt{n}$ for each layer $\mathrm{i}\in\{1,2,\ldots,h\}$. Then, by the theory of [53] which allows the dimension of the parameters diverging with the training sample size, $\boldsymbol{Z}_{\mathrm{i}+1}^{(t)}$ is asymptotically Gaussian.

Let $Y_{\boldsymbol{z}}$ denote the unknown true observation at the test point $\boldsymbol{z}$, and let $\hat{\xi}^{(t)}(\boldsymbol{z})=Z_{h+1}^{(t)}$ denote its K-StoNet prediction with the parameters $\boldsymbol{\theta}_n^{(t)}$. Then the variance of $Y_{\boldsymbol{z}}-\hat{\xi}(\boldsymbol{z})$ can be approximated by

$$\widehat{\mathrm{Var}}(Y_{\boldsymbol{z}}-\hat{\xi}(\boldsymbol{z}))=\frac{1}{n}\sum_{\mathrm{i}=1}^{n}(y^{(\mathrm{i})}-\hat{\xi}^{(t)}(\boldsymbol{x}^{(\mathrm{i})}))(y^{(\mathrm{i})}-\hat{\xi}^{(t)}(\boldsymbol{x}^{(\mathrm{i})}))^T+\widehat{\Sigma}_{h+1}^{(t)}, \tag{3.19}$$

based on which the prediction interval for Y_z can be constructed at a desired confidence level. Further, by Theorem 3.1.3, a more accurate confidence interval can be obtained by averaging over those obtained at different iterations.

The above procedure can be easily extended to the probit/logistic regression (via Wald/end-point transformation) and the case that an amenable penalty is used in solving (3.15). Refer to [110] for asymptotic normality of the regularized estimators.

3.4.2 A Numerical Example

To illustrate the above procedure, we generated 100 training datasets and one test dataset as in Section 3.2.2. Each dataset consisted of 500 samples. Again, we modeled the data using a K-StoNet with one hidden layer and 5 hidden units. For each training dataset, we trained the model by IRO for 50 epochs. For each test point $\boldsymbol{z}$, a 95% prediction interval was constructed based on each training dataset according to the prediction variance calculated in (3.19) at the last epoch of the run, and the coverage rate was calculated by averaging over the coverage status (0 or 1) of the 100 prediction intervals. Further, we averaged the coverage rates over 500 test points, which produced a mean coverage rate of 93.812% with standard deviation 0.781%. It is very close to the nominal level 95%! Figure 3.8 shows the prediction intervals at 20 test points, which were obtained at the last epoch of an IRO run for a training dataset.

For each test point, we have also constructed a prediction interval based on each training dataset by averaging those obtained at the last 25 epochs. As a result, the mean coverage rate over the 500 test points was improved to 94.026% with standard deviation 0.771%.

3.5 Parameter Settings for K-StoNet

In all computations of this chapter except for the CoverType experiments, the RBF kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2)$ is used, where γ is set to the default value $\frac{1}{\mathrm{pVar}(\boldsymbol{x})}$, p is the dimension of $\boldsymbol{x}$, and $\mathrm{Var}(\boldsymbol{x})$ is the variance of $\boldsymbol{x}$. We have the following default values for the parameters: one hidden layer, 5 hidden units, $C_n = \tilde{C}_{n,1}^{(t)} = 10$ for all t, $\varepsilon = 0.01$, $t_{HMC} = 25$, $\alpha = 0.1$, $\sigma_{n,2}^2 = 0.01$, and the learning rate $\epsilon_{t,\mathrm{i}} = 5\mathrm{e}-4$ for all t and i. The

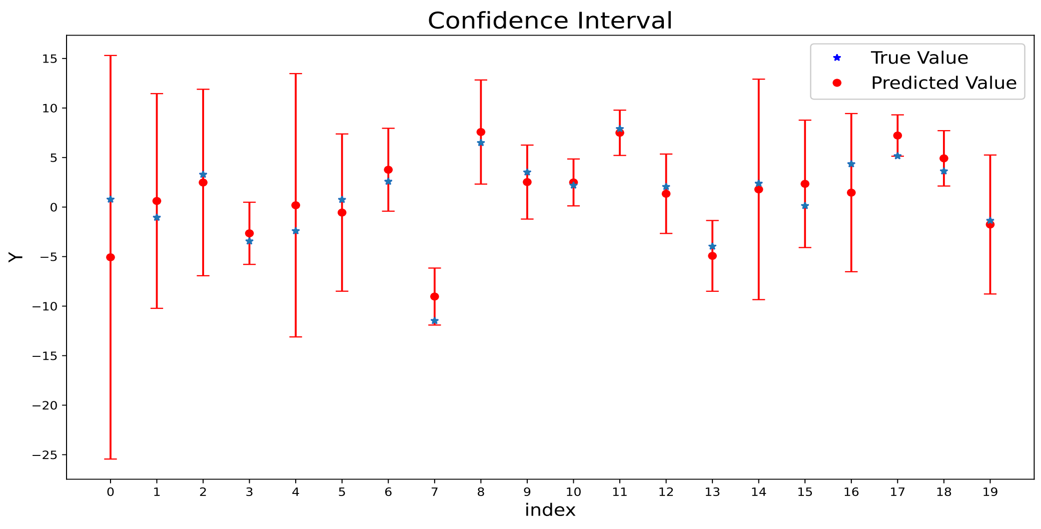


Figure 3.8. 95% prediction intervals produced by K-StoNet for 20 test points, where the x-axis indexes the test points, the y-axis represents the response value, and the blue star represents the true observation.

parameter settings may vary around the default values to achieve better performance for the K-StoNet model.

Section 3.2.1: Simulated DNN Data. Network: $C_n = 1$ for the SVR layer, $\sigma^2_{n,2} = 0.001$ for the output layer; HMC imputation: $t_{HMC} = 25$, $\alpha = 0.1$, and $\epsilon_{t,\mathrm{i}} = 5\mathrm{e}-7$ for all t and i; parameter updating: for all t and i, (i) SVR with $\tilde{C}^{(t)}_{n,1} = 1$ and $\varepsilon = 0.1$, (ii) linear regression with a Lasso penalty and the regularization parameter $\lambda^{(t)}_{n,\mathrm{i}} = 1\mathrm{e}-4$.

Section 3.2.1: Simulated KNN Data. Network: $C_n = 5$ for the SVR layer, $\sigma^2_{n,2} = 0.001$ for the output layer; HMC imputation: $t_{HMC} = 25$, $\alpha = 0.1$, and $\epsilon_{t,\mathrm{i}} = 5\mathrm{e}-4$ for all t and i; parameter updating: for all t and i, (i) SVR with $\tilde{C}^{(t)}_{n,1} = 5$ and $\varepsilon = 0.01$, (ii) linear regression with a Lasso penalty and the regularization parameter $\lambda^{(t)}_{n,\mathrm{i}} = 1\mathrm{e}-4$.

Section 3.2.2: Measurement error data. For both the one-hidden layer and three-hidden layer K-StoNets, the parameters were set as follows: Network: $C_n = 1$ for the SVR layer, $\sigma^2_{n,\mathrm{i}} = 0.001$ for layers $\mathrm{i} = 2, \ldots, h$, and $\sigma^2_{n,h+1} = 0.01$; HMC imputation: $t_{HMC} = 25$, $\alpha = 1$, and $\epsilon_{t,\mathrm{i}} = 5\mathrm{e}-5$ for all t and i; parameter updating: for all t and i, (i) SVR with $\tilde{C}^{(t)}_{n,1} = 1$ and $\varepsilon \in \{0.01, 0.02, \ldots, 0.1\}$, (ii) linear regression with a Lasso penalty and the regularization parameter $\lambda^{(t)}_{n,\mathrm{i}} = 1\mathrm{e}-4$.

Section 3.3.1: QSAR Androgen Receptor. Network: $C_n = 1$ for the SVR layer, $\sigma^2_{n,2} = 0.001$ for the output layer; HMC imputation: $t_{HMC} = 25$, $\alpha = 0.1$, and $\epsilon_{t,\mathrm{i}} = 5\mathrm{e}-5$ for all t and i; parameter updating: for all t and i, (i) SVR with $\tilde{C}^{(t)}_{n,1} = 1$ and $\varepsilon = 0.1$, (ii) logistic regression with a Lasso penalty and the regularization parameter $\lambda^{(t)}_{n,\mathrm{i}} = 1\mathrm{e}-4$.

Section 3.3.2: MNIST Data. Network: $C_n = 10$ for the SVR layer, $\sigma^2_{n,2} = 1\mathrm{e}-9$ for the output layer; HMC imputation: $t_{HMC} = 25$, $\alpha = 0.1$, and $\epsilon_{t,\mathrm{i}} = 5\mathrm{e}-13$ for all t and i; parameter updating: for all t and i, (i) SVR with $\tilde{C}^{(t)}_{n,1} = 10$ and $\varepsilon = 0.0001$, (ii) multinomial logistic regression with a Lasso penalty and the regularization parameter $\lambda^{(t)}_{n,\mathrm{i}} = 1\mathrm{e}-4$.

Section 3.3.3: CoverType Data. Network: $C_n = 10$ for the SVR layer, $\sigma^2_{n,2} = 0.005$ for the output layer; HMC imputation: $t_{HMC} = 25$, $\alpha = 0.1$, and $\epsilon_{t,\mathrm{i}} = 5\mathrm{e}-5$ for all t and i; parameter updating: for all t and i, (i) SVR with $\tilde{C}^{(t)}_{n,1} = 10$ and $\varepsilon = 0.01$. (ii) multinomial logistic regression with a Lasso penalty and the regularization parameter $\lambda^{(t)}_{n,\mathrm{i}} = 1\mathrm{e}-4$. This dataset consists of 44 binary features. When applying the RBF kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma\|\boldsymbol{x} - \boldsymbol{x}'\|^2_2)$, the default choice $\gamma = \frac{1}{p\mathrm{Var}(\boldsymbol{x})}$ does not work well. Different values of

γ were used for different SVRs in the K-StoNet model. Let γ_{i} denote the γ-value used for the SVR corresponding to the i-th hidden unit. We set $\gamma_{\mathrm{i}} = 0.5$ for $1 \leq \mathrm{i} < 30$, $\gamma_{\mathrm{i}} = 1$ for $30 \leq \mathrm{i} < 40$, $\gamma_{\mathrm{i}} = 2$ for $40 \leq \mathrm{i} < 45$, and $\gamma_{\mathrm{i}} = 5$ for $45 \leq \mathrm{i} \leq 50$.

Section 3.3.4: For all 10 datasets except for *Yacht Hydrodynamic* and *Year Prediction MSD*, we set $\sigma^2_{n,2} = 0.01$, $t_{HMC} = 25$, $\alpha = 0.1$, and $\epsilon_{t,\mathrm{i}} = 5\mathrm{e}-4$ for all t and i. For the SVRs in the first layer, we set $\epsilon = 0.01$. We used $\frac{1}{9}$ of the training data as the validation set and chose $\tilde{C}^{(t)}_{n,1} \in 1, 2, 5, 10, 20$ with the smallest MSE on the validation set. For the dataset *Yacht Hydrodynamic*, we set $\sigma^2_{n,2} = 0.0001$, $\alpha = 0.1$, $\epsilon_{t,\mathrm{i}} = 5\mathrm{e}-6$ and $\tilde{C}^{(t)}_{n,1} = 200$. For the dataset *Year Prediction MSD*, we set $\sigma^2_{n,2} = 0.02$, $\alpha = 0.1$, $\epsilon_{t,\mathrm{i}} = 1\mathrm{e}-3$ and $\tilde{C}^{(t)}_{n,1} = 1$. Similar to the CoverType dataset, when some categorical features exist in the dataset, the default choice $\gamma = \frac{1}{p\mathrm{Var}(\boldsymbol{x})}$ in the RBF kernel does not work very well. Among the 10 datasets, we set $\gamma = 3$ for *Yacht Hydrodynamic*, $\gamma = 1$ for *Protein Structure*, and employed the default setting for the others.

The KNN model was trained in a similar setting as used for the probabilistic back-propagation method in [107]: we used a one-hidden layer model with 50 hidden units, and trained the model using SGD with a constant learning rate of 0.0001 and a momentum decay factor of 0.9. As in [107], we ran SGD for 40 epochs with a mini-batch size of 1.

Section 3.4.2: Prediction Interval. Network: $C_n = 10$ for the SVR layer, $\sigma^2_{n,2} = 0.001$ for the output layer; HMC imputation: $t_{HMC} = 25$, $\alpha = 0.1$, and $\epsilon_{t,\mathrm{i}} = 5\mathrm{e}-6$ for all t and i; parameter updating: for all t and i, (i) SVR with $\tilde{C}^{(t)}_{n,1} = 10$ and $\varepsilon = 0.05$, (ii) linear regression, OLS estimation.

3.6 Discussion

We have proposed K-StoNet as a new neural network model for machine learning. The K-StoNet incorporates SVR as the first hidden layer and reformulates the neural network as a latent variable model. The former maps the input variable into an infinite dimensional feature space via the RBF kernel, ensuring absence of local minima on the loss surface of the resulting neural network. The latter breaks the high-dimensional nonconvex neural network training problem into a series of lower-dimensional convex optimization problems. In addition, the use of kernel partially addresses the over-parameterization issue suffered by the DNN; it enables a smaller network to be used, while ensuring the universal approximation capability. The K-StoNet can be easily trained using the IRO algorithm. Compared to DNN, K-StoNet avoids local traps, and enables the prediction uncertainty easily assessed. Compared to SVR, K-StoNet has better approximation capability due to the added hidden units. Our numerical results indicate its superiority over SVR and DNN in both training and prediction.

As an important ingredient of K-StoNet, StoNet is itself of interest. Under the framework of StoNet, the existing statistical theory for SVR and high-dimensional generalized linear models can be easily incorporated into the development of deep learning.

As another important ingredient of K-StoNet, kernel has long been studied in machine learning as a function approximation tool. It is known that, with "kernel trick", kernel methods enable a classifier/regression to learn a complex decision boundary with only a small number of parameters. To enhance their flexibility, some researchers proposed the so-called deep kernel learning methods, where one kernel function is repeatedly concatenated with another kernel or nonlinear function, see e.g. [111]–[117]. Under some conditions, [113] showed that the upper bound of generalization error for deep multiple kernels can be significantly lower than that for the DNNs. However, unlike shallow kernel methods such as SVM and SVR [31], [32], [118], coefficient estimation for deep kernels is not convex any more. Estimating coefficients of the inner layer kernel can be highly nonlinear and becomes more complicated for a larger number of layers [117]. By introducing latent variables, this work provides an effective way to resolve the computational challenge suffered by deep kernel learning. K-StoNet is essentially a deep kernel learning method.

The K-StoNet can be further extended in many different ways. Instead of relying on a SVR solver, K-StoNet can be implemented as a StoNet with the gram matrix being treated as input data. In this case, although a large-scale gram matrix needs to be handled when the training sample size is large, different kernels can be adopted for different tasks. For example, one might employ the convolutional kernel developed in [114] for computer vision problems. As discussed in Section 3.1.3, the regression in the output and other hidden layers can also be regularized by different amenable penalties [95], some of which might lead to better selection properties than Lasso.

The K-StoNet has an embarrassingly parallel structure; that is, solving K-StoNet can be broken into to many parallel tasks that can be solved with little or no need for communications. More precisely, the imputation step can be done in parallel for each observation; and the parameter updating step can be done in parallel for each of the regression tasks, including both SVR and Lasso regression. If both steps are implemented in parallel, the computation can be greatly accelerated. Currently, parallelization has not yet been completed: the imputation step was implemented in PyTorch, where the imputation for each observation was done in a serial manner; the parameter updating step was implemented using the package *sklearn*[119] for the normal or logistic regression and *ThunderSVM* [120] for SVR, where the regression was solved one by one. On a machine with Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz and Nvidia Tesla P100 GPU, for a dataset with $n = 10000$ and $p = 54$ and a 1-hidden layer K-StoNet with 5 hidden units, one iteration/epoch cost less than half a minute in the current serial implementation. Therefore, all the examples presented in this chapter can be done reasonably fast on the machine. We expect that the computation can be much accelerated with a parallel implementation of K-StoNet.

As mentioned in Section 3.1.3, a scalable SVR solver will accelerate the computation of K-StoNet substantially. The scalable SVR can be developed in different ways. For example, [121] developed ParitoSVR — a parallel iterated optimizer for SVR, where each machine iteratively solves a small (sub-)problem based only on a subset data and these solutions are then combined to form the solution to the global problem. ParitoSVR is provably convergent to the results obtained from the centralized algorithm, where the optimization has access to the entire data set. Alternatively, one can implement SVR in an incremental manner [122]–[125], where a SVR is first learned with a subset data and then sequentially updated based on the remaining set of samples. By the property that the decision function of SVR depends on support vectors only, [124] proposed to use only the boundary vectors in the remaining set of samples. By the same property, [125] proposed a sample selection method for SVR to maximize its validation set accuracy at the minimum number of training examples. As shown in [124], [125], both methods can accelerate the computation of SVR substantially.

3.7 Technical Proofs

3.7.1 Proof of Theorem 3.1.2

Proof. Since Θ is compact, it suffices to prove that the consistency holds for any $\boldsymbol{\theta} \in \Theta$. For notational simplicity, we rewrite $\sigma_{n,\mathrm{i}}$ by σ_{i} in the remaining part of the proof.

Let $\boldsymbol{Y}_{\mathrm{mis}} = (\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_h)$, where $\boldsymbol{Y}_{\mathrm{i}}$'s are latent variables as defined in (3.5). Let $\tilde{\boldsymbol{Y}} = (\tilde{\boldsymbol{Y}}_1, \ldots, \tilde{\boldsymbol{Y}}_h)$, where $\tilde{\boldsymbol{Y}}_{\mathrm{i}}$'s are calculated by KNN in (3.3). By Taylor expansion, we have

$$\log\pi(\boldsymbol{Y}, \boldsymbol{Y}_{\mathrm{mis}}|\boldsymbol{X}, \boldsymbol{\theta}) = \log\pi(\boldsymbol{Y}, \tilde{\boldsymbol{Y}}|\boldsymbol{X}, \boldsymbol{\theta}) + \boldsymbol{\epsilon}^T \nabla_{\boldsymbol{Y}_{\mathrm{mis}}} \log\pi(\boldsymbol{Y}, \tilde{\boldsymbol{Y}}|\boldsymbol{X}, \boldsymbol{\theta}) + O(\|\boldsymbol{\epsilon}\|^2), \tag{3.20}$$

where $\boldsymbol{\epsilon} = \boldsymbol{Y}_{\mathrm{mis}} - \tilde{\boldsymbol{Y}} = (\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2, \ldots, \boldsymbol{\epsilon}_h)$, $\nabla_{\boldsymbol{Y}_{\mathrm{mis}}} \log\pi(\boldsymbol{Y}, \tilde{\boldsymbol{Y}}|\boldsymbol{X}, \boldsymbol{\theta})$ is evaluated according to the joint distribution (3.10), and $\log\pi(\boldsymbol{Y}, \tilde{\boldsymbol{Y}}|\boldsymbol{X}, \boldsymbol{\theta}) = \log\pi(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta})$ is the log-likelihood function of the KNN.

Consider the partial derivative $\nabla_{\boldsymbol{Y}_{\mathrm{i}}} \log\pi(\boldsymbol{Y}, \boldsymbol{Y}_{\mathrm{mis}}|\boldsymbol{X}, \boldsymbol{\theta})$, for whose single component, say $Y_{\mathrm{i}}^{(k)}$, the output of neuron k at hidden layer $\mathrm{i} \in \{2, 3, \ldots, h\}$, we have

$$\begin{aligned}\nabla_{Y_{\mathrm{i}}^{(k)}} \log\pi(\boldsymbol{Y}, \boldsymbol{Y}_{\mathrm{mis}}|\boldsymbol{X}, \boldsymbol{\theta}) = {}& \frac{1}{\sigma_{\mathrm{i}+1}^2} \sum_{\mathrm{j}=1}^{m_{\mathrm{i}+1}} (Y_{\mathrm{i}+1}^{(\mathrm{j})} - b_{\mathrm{i}+1}^{(\mathrm{j})} - \boldsymbol{w}_{\mathrm{i}+1}^{(\mathrm{j})} \psi(\boldsymbol{Y}_{\mathrm{i}})) w_{\mathrm{i}+1}^{(\mathrm{j},k)} \psi'(Y_{\mathrm{i}}^{(k)}) \\ & - \frac{1}{\sigma_{\mathrm{i}}^2} (Y_{\mathrm{i}}^{(k)} - b_{\mathrm{i}}^{(k)} - \boldsymbol{w}_{\mathrm{i}}^{(k)} \psi(\boldsymbol{Y}_{\mathrm{i}-1})),\end{aligned} \tag{3.21}$$

where $\boldsymbol{w}_{\mathrm{i}+1}^{(\mathrm{j})}$ denotes the vector of the weights from neuron j at layer $\mathrm{i}+1$ to the neurons at layer i, and $w_{\mathrm{i}+1}^{(\mathrm{j},k)}$ denotes the weight from neuron j at layer $\mathrm{i}+1$ to neuron k at layer i. For layer $\mathrm{i} = 1$, the second term of (3.21) will disappear, since the ϵ-intensive loss is a constant around zero.

Since $Y_{\mathrm{i}+1}^{(\mathrm{j})} = b_{\mathrm{i}+1}^{(\mathrm{j})} + \boldsymbol{w}_{\mathrm{i}+1}^{(\mathrm{j})} \psi(\boldsymbol{Y}_{\mathrm{i}}) + \mathrm{e}_{\mathrm{i}+1}^{(\mathrm{j})}$ and $\tilde{Y}_{\mathrm{i}}^{(k)} = b_{\mathrm{i}}^{(k)} + \boldsymbol{w}_{\mathrm{i}}^{(k)} \psi(\tilde{\boldsymbol{Y}}_{\mathrm{i}-1})$, we have, for any $k \in \{1, 2, \ldots, m_{\mathrm{i}}\}$, $\nabla_{Y_{\mathrm{i}}^{(k)}} \log\pi(\boldsymbol{Y}, \tilde{\boldsymbol{Y}}|\boldsymbol{X}, \boldsymbol{\theta}) = \frac{1}{\sigma_{\mathrm{i}+1}^2} \sum_{\mathrm{j}=1}^{m_{\mathrm{i}+1}} \Big(\mathrm{e}_{\mathrm{i}+1}^{(\mathrm{j})} + \boldsymbol{w}_{\mathrm{i}+1}^{(\mathrm{j})} (\psi(\boldsymbol{Y}_{\mathrm{i}}) - \psi(\tilde{\boldsymbol{Y}}_{\mathrm{i}}))\Big) w_{\mathrm{i}+1}^{(\mathrm{j},k)} \psi'(\tilde{Y}_{\mathrm{i},k})$ if $\mathrm{i} = h$, and 0 otherwise. Then, by Assumption 3.1.2-(i)&(iii), for any $k \in \{1, 2, \ldots, m_{\mathrm{i}}\}$,

$$|\nabla_{Y_{\mathrm{i}}^{(k)}} \log\pi(\boldsymbol{Y}, \tilde{\boldsymbol{Y}}|\boldsymbol{X}, \boldsymbol{\theta})| \leq \begin{cases} \frac{1}{\sigma_{\mathrm{i}+1}^2} \Big\{ \sum_{\mathrm{j}=1}^{m_{\mathrm{i}+1}} \mathrm{e}_{\mathrm{i}+1}^{(\mathrm{j})} w_{\mathrm{i}+1}^{(\mathrm{j},k)} \psi'(\tilde{Y}_{\mathrm{i}}^{(k)}) + (c'r)^2 m_{\mathrm{i}+1} \|\boldsymbol{\epsilon}_{\mathrm{i}}\| \Big\}, & \mathrm{i} = h \\ 0 & \mathrm{i} < h, \end{cases} \tag{3.22}$$

where $\boldsymbol{\epsilon}_{\mathrm{i}} = \boldsymbol{Y}_{\mathrm{i}} - \tilde{\boldsymbol{Y}}_{\mathrm{i}}$, $\mathrm{e}_{\mathrm{i+1}}^{(\mathrm{j})}$ is the jth component of $\mathrm{e}_{\mathrm{i+1}}$, r is the upper bound of the weights, and c' is the Lipschitz constant of $\psi(\cdot)$ as well as the upper bound of $\psi'(\cdot)$.

Next, let's figure out the order of $\|\boldsymbol{\epsilon}_{\mathrm{i}}\|$. The kth component of $\boldsymbol{\epsilon}_{\mathrm{i}}$ is given by

$$Y_{\mathrm{i}}^{(k)} - \tilde{Y}_{\mathrm{i}}^{(k)} = \begin{cases} \mathrm{e}_{\mathrm{i}}^{(k)} + \boldsymbol{w}_{\mathrm{i}}^{(k)}(\psi(\boldsymbol{Y}_{\mathrm{i-1}}) - \psi(\tilde{\boldsymbol{Y}}_{\mathrm{i-1}})), & \mathrm{i} > 1, \\ \mathrm{e}_{\mathrm{i}}^{(k)}, & \mathrm{i} = 1. \end{cases}$$

Therefore, $\|\boldsymbol{\epsilon}_1\| = \|\mathrm{e}_1\|$; and for $\mathrm{i} = 2, 3, \ldots, h$, the following inequalities hold:

$$\|\boldsymbol{\epsilon}_{\mathrm{i}}\| \leq \|\mathrm{e}_{\mathrm{i}}\| + c' r m_{\mathrm{i}} \|\boldsymbol{\epsilon}_{\mathrm{i-1}}\|, \quad \|\boldsymbol{\epsilon}_{\mathrm{i}}\|^2 \leq 2\|\mathrm{e}_{\mathrm{i}}\|^2 + 2(c'r)^2 m_{\mathrm{i}}^2 \|\boldsymbol{\epsilon}_{\mathrm{i-1}}\|^2. \tag{3.23}$$

Since e_{i} and $\mathrm{e}_{\mathrm{i-1}}$ are independent, by summarizing (3.22) and (3.23), we have

$$\int \boldsymbol{\epsilon}^T \nabla_{\boldsymbol{Y}_{\mathrm{mis}}} \log \pi(\boldsymbol{Y}, \tilde{\boldsymbol{Y}} | \boldsymbol{X}, \boldsymbol{\theta}) \pi(\boldsymbol{Y}_{\mathrm{mis}} | \boldsymbol{X}, \boldsymbol{\theta}, \boldsymbol{Y}) d\boldsymbol{Y}_{\mathrm{mis}} \leq O(\sum_{k=2}^{h+1} \frac{\sigma_{k-1}^2}{\sigma_{h+1}^2} m_{h+1} (\prod_{\mathrm{i}=k}^{h} m_{\mathrm{i}}^2) m_{k-1}) = o(1), \tag{3.24}$$

where the last equality follows from 3.1.2-(v). Then, by (3.20), we have the mean value

$$\mathbb{E}\left[\log \pi(\boldsymbol{Y}, \boldsymbol{Y}_{\mathrm{mis}} | \boldsymbol{X}, \boldsymbol{\theta}) - \log \pi(\boldsymbol{Y} | \boldsymbol{X}, \boldsymbol{\theta})\right] \to 0, \quad \forall \boldsymbol{\theta} \in \Theta.$$

Further, it is easy to verify

$$\int |\boldsymbol{\epsilon}^T \nabla_{\boldsymbol{Y}_{\mathrm{mis}}} \log \pi(\boldsymbol{Y}, \tilde{\boldsymbol{Y}} | \boldsymbol{X}, \boldsymbol{\theta})|^2 \pi(\boldsymbol{Y}_{\mathrm{mis}} | \boldsymbol{X}, \boldsymbol{\theta}, \boldsymbol{Y}) d\boldsymbol{Y}_{\mathrm{mis}} < \infty,$$

which, together with (3.20) and (3.23), implies

$$\mathbb{E} |\log \pi(\boldsymbol{Y}, \boldsymbol{Y}_{\mathrm{mis}} | \boldsymbol{X}, \boldsymbol{\theta}) - \log \pi(\boldsymbol{Y} | \boldsymbol{X}, \boldsymbol{\theta})|^2 < \infty. \tag{3.25}$$

Therefore, the weak law of large numbers (WLLN) applies, and the proof can be concluded.

□

3.7.2 Proof of Lemma 3.1.2

Lemma 3.1.2 is a direct application of Lemma 3.7.1 given below.

Lemma 3.7.1. *Consider a function $Q(\boldsymbol{\theta}, \boldsymbol{X}_n)$. Suppose that the following conditions are satisfied: (B1) $Q(\boldsymbol{\theta}, \boldsymbol{X}_n)$ is continuous in $\boldsymbol{\theta}$ and there exists a function $Q^*(\boldsymbol{\theta})$, which is* *continuous in* $\boldsymbol{\theta}$ *and uniquely maximized at* $\boldsymbol{\theta}^*$*. (B2) For any* $\epsilon > 0$*,* $\sup_{\boldsymbol{\theta}\in\Theta\setminus B(\epsilon)} Q^*(\boldsymbol{\theta})$ *exists, where* $B(\epsilon) = \{\boldsymbol{\theta} : \|\boldsymbol{\theta} - \boldsymbol{\theta}^*\| < \epsilon\}$*; Let* $\delta = Q^*(\boldsymbol{\theta}^*) - \sup_{\boldsymbol{\theta}\in\Theta\setminus B(\epsilon)} Q^*(\boldsymbol{\theta})$*, then* $\delta > 0$*. (B3)* $\sup_{\boldsymbol{\theta}\in\Theta} |Q(\boldsymbol{\theta}, \boldsymbol{X}_n) - Q^*(\boldsymbol{\theta})| \xrightarrow{p} 0$ *as* $n \to \infty$*. Let* $\hat{\boldsymbol{\theta}}_n = \arg\max_{\boldsymbol{\theta}\in\Theta} Q(\boldsymbol{\theta}, \boldsymbol{X}_n)$*. Then* $\|\hat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}^*\| \xrightarrow{p} 0$*.*

Proof. Consider two events (i) $\sup_{\boldsymbol{\theta}\in\Theta\setminus B(\epsilon)} |Q(\boldsymbol{\theta}, \boldsymbol{X}_n) - Q^*(\boldsymbol{\theta})| < \delta/2$, and
(ii) $\sup_{\boldsymbol{\theta}\in B(\epsilon)} |Q(\boldsymbol{\theta}, \boldsymbol{X}_n) - Q^*(\boldsymbol{\theta})| < \delta/2$. From event (i), we can deduce that for any $\boldsymbol{\theta} \in \Theta \setminus B(\epsilon)$, $Q(\boldsymbol{\theta}, \boldsymbol{X}_n) < Q^*(\boldsymbol{\theta}) + \delta/2 \leq Q^*(\boldsymbol{\theta}^*) - \delta + \delta/2 \leq Q^*(\boldsymbol{\theta}^*) - \delta/2$.

From event (ii), we can deduce that for any $\boldsymbol{\theta} \in B(\epsilon)$, $Q(\boldsymbol{\theta}, \boldsymbol{X}_n) > Q^*(\boldsymbol{\theta}) - \delta/2$ and thus $Q(\boldsymbol{\theta}^*, \boldsymbol{X}_n) > Q^*(\boldsymbol{\theta}^*) - \delta/2$.

If both events hold simultaneously, then we must have $\hat{\boldsymbol{\theta}}_n \in B(\epsilon)$ as $n \to \infty$. By condition (B3), the probability that both events hold tends to 1. Therefore, $P(\hat{\boldsymbol{\theta}}_n \in B(\epsilon)) \to 1$, which concludes the lemma. □

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[2] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. de Freitas, "Predicting parameters in deep learning," in *NIPS*, 2013.

[3] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17, Sydney, NSW, Australia: JMLR.org, 2017, pp. 1321–1330.

[4] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=rJl-b3RcF7.

[5] M. Gori and A. Tesi, "On the problem of local minima in backpropagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 1, pp. 76–86, 1992.

[6] Q. Nguyen and M. Hein, "The loss surface of deep and wide neural networks," in *ICML*, 2017.

[7] Z. Allen-Zhu, Y. Li, and Z. Song, "A convergence theory for deep learning via over-parameterization," in *ICML*, 2019.

[8] S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai, "Gradient descent finds global minima of deep neural networks," in *ICML*, 2019.

[9] D. Zou, Y. Cao, D. Zhou, and Q. Gu, "Gradient descent optimizes over-parameterized deep relu networks," *Machine Learning*, vol. 109, pp. 467–492, 2020.

[10] D. Zou and Q. Gu, "An improved analysis of training over-parameterized deep neural networks," in *NuerIPS*, 2019.

[11] H. Bölcskei, P. Grohs, G. Kutyniok, and P. Petersen, "Optimal approximation with sparsely connected deep neural networks," *CoRR*, vol. abs/1705.01714, 2019.

[12] J. Schmidt-Hieber, "Nonparametric regression using deep neural networks with relu activation function," *arXiv:1708.06633*, 2017.

[13] B. Bauler and M. Kohler, "On deep learning as a remedy for the curse of dimensionality in nonparametric regression," *The Annals of Statistics*, vol. 47, no. 4, pp. 2261–2285, 2019.

[14] F. Liang, Q. Li, and L. Zhou, "Bayesian neural networks for selection of drug sensitive genes," *Journal of the American Statistical Association*, vol. 113, no. 523, pp. 955–972, 2018.

[15] N. G. Polson and V. Ročková, "Posterior concentration for sparse deep learning," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18, Montréal, Canada: Curran Associates Inc., 2018, pp. 938–949.

[16] J. M. Alvarez and M. Salzmann, "Learning the number of neurons in deep networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 2270–2278.

[17] S. Scardapane, D. Comminiello, A. Hussain, and A. Uncini, "Group sparse regularization for deep neural networks," *Neurocomputing*, vol. 241, pp. 81–89, 2017.

[18] R. Ma, J. Miao, L. Niu, and P. Zhang, "Transformed l_1 regularization for learning sparse deep neural networks," *ArXiv:1901.01021v1*, 2019.

[19] S. Wager, S. Wang, and P. S. Liang, "Dropout training as adaptive regularization," *Advances in neural information processing systems*, vol. 26, 2013.

[20] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[21] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky, "Sparse convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 806–814.

[22] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018.

[23] S. Ghosh and F. Doshi-Velez, "Model selection in bayesian neural networks via horseshoe priors," *arXiv preprint arXiv:1705.10388*, 2017.

[24] G. E. Hinton, "Learning multiple layers of representation," *Trends in cognitive sciences*, vol. 11, no. 10, pp. 428–434, 2007.

[25] R. Salakhutdinov and G. Hinton, "Deep boltzmann machines," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2009, pp. 448–455.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[27] A. Neelakantan, L. Vilnis, Q. V. Le, *et al.*, "Adding gradient noise improves learning for very deep networks," *arXiv preprint arXiv:1511.06807*, 2015.

[28] Z. You, J. Ye, K. Li, Z. Xu, and P. Wang, "Adversarial noise layer: Regularize neural network by adding noise," in *2019 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2019, pp. 909–913.

[29] H. Noh, T. You, J. Mun, and B. Han, "Regularizing deep neural networks by noise: Its interpretation and optimization," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[30] C. Gulcehre, M. Moczulski, M. Denil, and Y. Bengio, "Noisy activation functions," in *International conference on machine learning*, PMLR, 2016, pp. 3059–3068.

[31] V. Vapnik, "Pattern recognition using generalized portrait method," *Automation and remote control*, vol. 24, pp. 774–780, 1963.

[32] V. Vapnik, "A note one class of perceptrons," *Automation and remote control*, 1964.

[33] F. Liang, B. Jia, J. Xue, Q. Li, and Y. Luo, "An imputation–regularized optimization algorithm for high dimensional missing data problems and beyond," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 80, no. 5, pp. 899–926, 2018.

[34] C. A. Micchelli, Y. Xu, and H. Zhang, "Universal kernels.," *Journal of Machine Learning Research*, vol. 7, no. 12, 2006.

[35] B. Hammer and K. Gersmann, "A note on the universal approximation capability of support vector machines," *neural processing letters*, vol. 17, no. 1, pp. 43–53, 2003.

[36] H. Ishwaran, J. S. Rao, *et al.*, "Spike and slab variable selection: Frequentist and bayesian strategies," *The Annals of Statistics*, vol. 33, no. 2, pp. 730–773, 2005.

[37] E. I. George and R. E. McCulloch, "Variable selection via gibbs sampling," *Journal of the American Statistical Association*, vol. 88, no. 423, pp. 881–889, 1993.

[38] Q. Song and F. Liang, "Nearly optimal bayesian shrinkage for high dimensional regression," *arXiv:1712.08964*, 2017.

[39] W. Jiang, "Bayesian variable selection for high dimensional generalized linear models: Convergence rate of the fitted densities," *The Annals of Statistics*, vol. 35, pp. 1487–1511, 2007.

[40] F. Liang, Q. Song, and K. Yu, "Bayesian subset modeling for high dimensional generalized linear models," *Journal of the American Statistical Association*, vol. 108, pp. 589–606, 2013.

[41] P. Petersen and F. Voigtlaender, "Optimal approximation of piecewise smooth functions using deep relu neural networks," *Neural Networks*, vol. 108, pp. 296–330, 2018.

[42] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever, "Deep double descent: Where bigger models and more data hurt," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=B1g5sA4twr.

[43] S. Ghosal, J. K. Ghosh, and A. W. Van Der Vaart, "Convergence rates of posterior distributions," *Annals of Statistics*, vol. 28, no. 2, pp. 500–531, 2000.

[44] E. I. George and R. E. McCulloch, "Approaches for bayesian variable selection," *Statistica sinica*, vol. 7, pp. 339–373, 1997.

[45] R. Kohn, M. Smith, and D. Chan, "Nonparametric regression using linear combinations of basis functions," *Statistics and Computing*, vol. 11, no. 4, pp. 313–322, 2001.

[46] A. Dobra, C. Hans, B. Jones, J. R. Nevins, G. Yao, and M. West, "Sparse graphical models for exploring gene expression data," *Journal of Multivariate Analysis*, vol. 90, no. 1, pp. 196–212, 2004.

[47] A. A. Pourzanjani, R. M. Jiang, and L. R. Petzold, "Improving the identifiability of neural networks for bayesian inference," in *NIPS Workshop on Bayesian Deep Learning*, 2017.

[48] S. Geisser, J. Hodges, S. Press, and A. ZeUner, "The validity of posterior expansions based on laplace's method," *Bayesian and likelihood methods in statistics and econometrics*, vol. 7, p. 473, 1990.

[49] I. Castillo, J. Rousseau, *et al.*, "A bernstein–von mises theorem for smooth functionals in semiparametric models," *The Annals of Statistics*, vol. 43, no. 6, pp. 2353–2383, 2015.

[50] Y. Wang and V. Rocková, "Uncertainty quantification for sparse deep learning," in *AISTATS*, 2020.

[51] J. Feng and N. Simon, "Sparse-input neural networks for high-dimensional nonparametric regression and classification," *arXiv preprint arXiv:1711.07592*, 2017.

[52] C. Fefferman, "Reconstructing a neural net from its output," *Revista Matemática Iberoamericana*, vol. 10, no. 3, pp. 507–555, 1994.

[53] S. Portnoy, “Asymptotic behavior of likelihood methods for exponential families when the number of parameters tend to infinity,” *The Annals of Statistics*, vol. 16, no. 1, pp. 356–366, 1988.

[54] D. A. McAllester, “Pac-bayesian model averaging,” in *Proceedings of the twelfth annual conference on Computational learning theory*, 1999, pp. 164–170.

[55] D. A. McAllester, “Some pac-bayesian theorems,” *Machine Learning*, vol. 37, no. 3, pp. 355–363, 1999.

[56] P. Chaudhari, A. Choromanska, S. Soatto, *et al.*, “Entropy-sgd: Biasing gradient descent into wide valleys,” *arXivi:1611.01838*, 2016. eprint: 1611.01838 (cs.LG).

[57] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, “Averaging weights leads to wider optima and better generalization,” *arXiv:1803.05407*, 2018. eprint: 1803.05407 (cs.LG).

[58] F. Liang, “Evidence evaluation for bayesian neural networks using contour monte carlo,” *Neural Computation*, vol. 17, no. 6, pp. 1385–1410, 2005.

[59] D. J. MacKay, “The evidence framework applied to classification networks,” *Neural computation*, vol. 4, no. 5, pp. 720–736, 1992.

[60] B. Kleinberg, Y. Li, and Y. Yuan, “An alternative view: When does sgd escape local minima?” In *International Conference on Machine Learning*, PMLR, 2018, pp. 2698–2707.

[61] C. Zhang, Q. Liao, A. Rakhlin, B. Miranda, N. Golowich, and T. Poggio, “Theory of deep learning iib: Optimization properties of sgd,” *arXiv preprint*, arXiv:1801.02254, 2018.

[62] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[63] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[64] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations*, 2015.

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

[66] T. Chen, E. Fox, and C. Guestrin, "Stochastic gradient hamiltonian monte carlo," in *International conference on machine learning*, 2014, pp. 1683–1691.

[67] Y.-A. Ma, T. Chen, and E. Fox, "A complete recipe for stochastic gradient mcmc," in *Advances in Neural Information Processing Systems*, 2015, pp. 2917–2925.

[68] C. Nemeth and P. Fearnhead, "Stochastic gradient markov chain monte carlo," *arXiv preprint arXiv:1907.06986*, 2019.

[69] Y. Sun, Q. Song, and F. Liang, "Consistent sparse deep learning: Theory and computation," *Journal of the American Statistical Association*, in press, 2021.

[70] C. Chen, N. Ding, and L. Carin, "On the convergence of stochastic gradient mcmc algorithms with high-order integrators," in *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, 2015, pp. 2278–2286.

[71] J. Bleich, A. Kapelner, E. I. George, and S. T. Jensen, "Variable selection for bart: An application to gene regulation," *The Annals of Applied Statistics*, pp. 1750–1781, 2014.

[72] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[73] J. Fan and J. Lv, "Sure independence screening for ultrahigh dimensional feature space," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 70, no. 5, pp. 849–911, 2008.

[74] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[75] T. Lin, S. U. Stich, L. Barba, D. Dmitriev, and M. Jaggi, "Dynamic model pruning with feedback," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=SJem8lSFwB.

[76] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[77] H. Mostafa and X. Wang, "Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization," in *International Conference on Machine Learning*, 2019, pp. 4646–4655.

[78] T. Dettmers and L. Zettlemoyer, "Sparse networks from scratch: Faster training without losing performance," *arXiv preprint arXiv:1907.04840*, 2019.

[79] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson, “A simple baseline for bayesian uncertainty in deep learning,” in *Advances in Neural Information Processing Systems*, 2019, pp. 13 153–13 164.

[80] V. Ročková, “Bayesian estimation of sparse signals with a continuous spike-and-slab prior,” *The Annals of Statistics*, vol. 46, no. 1, pp. 401–437, 2018.

[81] Authors, “Extended stochastic gradient mcmc algorithms for large-scale bayesian computing,” *Submitted Manuscript*, 2019.

[82] A. Zubkov and A. Serov, “A complete proof of universal inequalities for the distribution function of the binomial law,” *Theory Probab. Appl.*, vol. 57, no. 3, pp. 539–544, 2013.

[83] W. V. Li and A. Wei, “A gaussian inequality for expected absolute products,” *Journal of Theoretical Probability*, vol. 25, no. 1, pp. 92–99, 2012.

[84] I. Castillo and J. Rousseau, “Supplement to “a bernstein–von mises theorem for smooth functionals in semiparametric models”,” *Annals of Statistics*, vol. 43, no. 6, pp. 2353–2383, 2015.

[85] G. Wahba, *Spline models for observational data*. SIAM, 1990.

[86] B. Schölkopf, R. Herbrich, and A. J. Smola, “A generalized representer theorem,” in *International conference on computational learning theory*, Springer, 2001, pp. 416–426.

[87] J. Park and I. W. Sandberg, “Universal approximation using radial-basis-function networks,” *Neural computation*, vol. 3, no. 2, pp. 246–257, 1991.

[88] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the em algorithm,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[89] G. Celeux, “The sem algorithm: A probabilistic teacher algorithm derived from the em algorithm for the mixture problem,” *Computational statistics quarterly*, vol. 2, pp. 73–82, 1985.

[90] A. J. Smola and B. Schölkopf, “A tutorial on support vector regression,” *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[91] A. Christmann and I. Steinwart, “Consistency and robustness of kernel-based regression in convex risk minimization,” *Bernoulli*, vol. 13, no. 3, pp. 799–819, 2007.

[92] I. Steinwart, “On the influence of the kernel on the consistency of support vector machines,” *Journal of machine learning research*, vol. 2, no. Nov, pp. 67–93, 2001.

[93] J. Fan and R. Li, "Variable selection via nonconcave penalized likelihood and its oracle properties," *Journal of the American statistical Association*, vol. 96, no. 456, pp. 1348–1360, 2001.

[94] C.-H. Zhang, "Nearly unbiased variable selection under minimax concave penalty," *The Annals of statistics*, vol. 38, no. 2, pp. 894–942, 2010.

[95] P.-L. Loh, M. J. Wainwright, *et al.*, "Support recovery without incoherence: A case for nonconvex regularization," *The Annals of Statistics*, vol. 45, no. 6, pp. 2455–2482, 2017.

[96] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid monte carlo," *Physics letters B*, vol. 195, no. 2, pp. 216–222, 1987.

[97] R. M. Neal *et al.*, "Mcmc using hamiltonian dynamics," *Handbook of markov chain monte carlo*, vol. 2, no. 11, p. 2, 2011.

[98] X. Cheng, N. S. Chatterji, P. L. Bartlett, and M. I. Jordan, "Underdamped langevin mcmc: A non-asymptotic analysis," in *Conference on learning theory*, PMLR, 2018, pp. 300–323.

[99] P. J. Rossky, J. D. Doll, and H. L. Friedman, "Brownian dynamics as smart monte carlo simulation," *The Journal of Chemical Physics*, vol. 69, no. 10, pp. 4628–4633, 1978.

[100] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 721–741, 1984.

[101] V. Vapnik, *The Nature of Statistical Learning Theory (2nd ed.)* New York: Springer, 2000.

[102] S. Balasundaram, D. D. Gupta, and K. Gupta, "Lagrangian support vector regression via unconstrained convex minimization," *Neural networks*, vol. 51C, pp. 67–79, Dec. 2013.

[103] S. F. Nielsen, "The stochastic em algorithm: Estimation and asymptotic results," *Bernoulli*, pp. 457–489, 2000.

[104] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[105] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[106] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16, New York, NY, USA: JMLR.org, 2016, pp. 1050–1059.

[107] J. M. Hernández-Lobato and R. Adams, "Probabilistic backpropagation for scalable learning of Bayesian neural networks," in *International Conference on Machine Learning*, PMLR, 2015, pp. 1861–1869.

[108] A. Graves, "Practical variational inference for neural networks," in *Advances in neural information processing systems*, Citeseer, 2011, pp. 2348–2356.

[109] J. B. Gao, S. R. Gunn, C. J. Harris, and M. Brown, "A probabilistic framework for svm regression and error bar estimation," *Machine Learning*, vol. 46, no. 1, pp. 71–89, 2002.

[110] P.-L. Loh, "Statistical consistency and asymptotic normality for high-dimensional robust M-estimators," *The Annals of Statistics*, vol. 45, no. 2, pp. 866–896, 2017.

[111] Y. Cho and L. Saul, "Kernel methods for deep learning," *Advances in neural information processing systems*, vol. 22, 2009.

[112] J. Zhuang, I. W. Tsang, and S. C. Hoi, "Two-layer multiple kernel learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 909–917.

[113] E. V. Strobl and S. Visweswaran, "Deep multiple kernel learning," in *2013 12th International Conference on Machine Learning and Applications*, IEEE, vol. 1, 2013, pp. 414–417.

[114] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid, "Convolutional kernel networks," *arXiv preprint arXiv:1406.3332*, 2014.

[115] I. Rebai, Y. BenAyed, and W. Mahdi, "Deep multilayer multiple kernel learning," *Neural Computing and Applications*, vol. 27, no. 8, pp. 2305–2314, 2016.

[116] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep kernel learning," in *Artificial intelligence and statistics*, PMLR, 2016, pp. 370–378.

[117] B. Bohn, C. Rieger, and M. Griebel, "A representer theorem for deep kernel learning," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 2302–2333, 2019.

[118] V. Vapnik, *The nature of statistical learning theory.* Springer science & business media, 1999.

[119] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, “Scikit-learn: Machine learning in python,” *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html.

[120] Z. Wen, J. Shi, Q. Li, B. He, and J. Chen, “ThunderSVM: A fast SVM library on GPUs and CPUs,” *Journal of Machine Learning Research*, vol. 19, pp. 797–801, 2018.

[121] K. Das, K. Bhaduri, B. L. Matthews, and N. C. Oza, “Large scale support vector regression for aviation safety,” in *2015 IEEE International Conference on Big Data (Big Data)*, IEEE, 2015, pp. 999–1006.

[122] J. Ma, J. Theiler, and S. Perkins, “Accurate on-line support vector regression,” *Neural computation*, vol. 15, no. 11, pp. 2683–2703, 2003.

[123] P. Laskov, C. Gehl, S. Krüger, K.-R. Müller, K. P. Bennett, and E. Parrado-Hernández, “Incremental support vector learning: Analysis, implementation and applications.,” *Journal of machine learning research*, vol. 7, no. 9, 2006.

[124] H. Xu, R. Wang, and K. Wang, “A new svr incremental algorithm based on boundary vector,” in *2010 International Conference on Computational Intelligence and Software Engineering*, IEEE, 2010, pp. 1–4.

[125] D. Ruta, L. Cen, and Q. H. Vu, “Greedy incremental support vector regression,” in *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*, IEEE, 2019, pp. 7–9.