Uncertainty Quantification for Large-Scale Deep Neural Networks via Post-StoNet Modeling

University of Pennsylvania and Purdue University

Supplementary Material

S1 Theoretical Proofs

S1.1 Proof of Lemma 2

Proof. For simplicity of notations, we suppress the iteration index t. Let $\tilde{\boldsymbol{Y}}_l = \boldsymbol{b}_l + \boldsymbol{w}_l \Psi(\boldsymbol{Y}_{l-1})$ for $l = 2, \ldots, h$, and let $\tilde{\boldsymbol{Y}}_1 = \boldsymbol{b}_1 + \boldsymbol{w}_1 \boldsymbol{X}$. By the definition of the StoNet model (2.2), $\boldsymbol{Y}_l$ can be written as $\boldsymbol{Y}_l = \tilde{\boldsymbol{Y}}_l + \boldsymbol{e}_l$ for $l \in \{1, 2, \ldots, h\}$.

Since σ_l^2 has been set to a very small value, we have $\Psi(\boldsymbol{Y}_l) \approx \Psi(\tilde{\boldsymbol{Y}}_l) + \nabla_{\tilde{\boldsymbol{Y}}_l} \Psi(\tilde{\boldsymbol{Y}}_l) \circ \boldsymbol{e}_l$, where $\circ$ denotes elementwise product. Then

$$\begin{aligned}\boldsymbol{\Sigma}_l &\approx \mathrm{Var}(\mathbb{E}(\Psi(\tilde{\boldsymbol{Y}}_l) + \nabla_{\tilde{\boldsymbol{Y}}_l} \Psi(\tilde{\boldsymbol{Y}}_l) \circ \boldsymbol{e}_l | \tilde{\boldsymbol{Y}}_l)) + \mathbb{E}(\mathrm{Var}(\Psi(\tilde{\boldsymbol{Y}}_l) + \nabla_{\tilde{\boldsymbol{Y}}_l} \Psi(\tilde{\boldsymbol{Y}}_l) \circ \boldsymbol{e}_l | \tilde{\boldsymbol{Y}}_l)) \\ &= \mathrm{Var}(\Psi(\tilde{\boldsymbol{Y}}_l)) + \mathrm{diag}\left\{ \sigma_l^2 \mathbb{E}[\nabla_{\tilde{\boldsymbol{Y}}_l} \Psi(\tilde{\boldsymbol{Y}}_l) \circ \nabla_{\tilde{\boldsymbol{Y}}_l} \Psi(\tilde{\boldsymbol{Y}}_l)] \right\},\end{aligned} \tag{S1.1}$$

where $\mathrm{diag}\{\boldsymbol{v}\}$ with $\boldsymbol{v} \in \mathbb{R}^d$ denotes a $d \times d$ diagonal matrix with diagonal elements being $\boldsymbol{v}$.

By Assumption 3-(iii), the activation function is bounded. For example, *tanh* or *sigmoid* is used in the model. By Assumption 1, there exists some constant C_1 such that $\|\boldsymbol{b}_l\|_\infty < C_1, \|\boldsymbol{w}_l\|_\infty < C_1$. By Assumption 3, $\|\boldsymbol{X}\|_\infty$ is bounded. Therefore, there exists some constant C_2 such that for any $L \in \{1, 2, \ldots, h\}$, $\|\tilde{\boldsymbol{Y}}_l\|_\infty \leq C_1 + C_1 C_2$ holds by rescaling $\boldsymbol{X}$ by a factor of $\prod_{l=1}^{h} d_l$. Since both $\Psi(\tilde{\boldsymbol{Y}}_l)$ and $\nabla_{\tilde{\boldsymbol{Y}}_l} \Psi(\tilde{\boldsymbol{Y}}_l)$ are bounded, there exists a constant $\kappa_{\max,l}$ such that

$$\phi_{\max}(d_{l,n}|\boldsymbol{\Sigma}_l) \leq \kappa_{\max,l}.$$

To establish the lower bound, we note that $\|\tilde{\boldsymbol{Y}}_l\|_\infty \leq C_1 + C_1 C_2$. Therefore, for an activation function which has nonzero gradients on any closed interval, e.g., *tanh* and *sigmoid*, there exists a constant $C_3 > 0$ such that $\min_{i=1,\ldots,d_l} \nabla_{\tilde{\boldsymbol{Y}}_l} \Psi(\tilde{\boldsymbol{Y}}_l)_i > C_3$, where $\nabla_{\tilde{\boldsymbol{Y}}_l} \Psi(\tilde{\boldsymbol{Y}}_l)_i$ denotes the i-th element of $\nabla_{\tilde{\boldsymbol{Y}}_l} \Psi(\tilde{\boldsymbol{Y}}_l)$. Then we can take $\kappa_{\min,l} = \sigma_{l,n}^2 C_3^2$ such that

$$\phi_{\min}(d_{l,n}|\boldsymbol{\Sigma}_l) \geq \kappa_{\min,l},$$

which completes the proof. □

S1.2 Proof of Part (i) of Theorem 1

Proof. By Lemma 2, $\Sigma_l^{(t)}$ satisfies the requirements of Theorem 1 of Meinshausen and Yu (2009) and Theorem 1 of Huang et al. (2008). Then, by Theorem 1 of Meinshausen and Yu (2009) (for linear regression) and Theorem 1 of Huang et al. (2008) (for logistic regression), we have r_n as given in the lemma by summarizing the l_2-errors of coefficient estimation for all $\sum_{l=1}^{h+1} d_l$ regression/logistic regressions. Further, by the setting of $(\sigma_{1,n}^2, \ldots, \sigma_{h+1,n}^2)$ as specified in Assumption 3, we have $r_n \to 0$ as $n \to \infty$. This completes the proof of part (i) of Theorem 1. □

S1.3 Proof of Part (ii) of Theorem 1

Proof. Then part (ii) of Theorem 1 directly follows from Theorem 4 of Liang et al. (2018) that the estimator $\hat{\boldsymbol{\theta}}_n^{(t)}$ is consistent when both n and t are sufficiently large. □

S1.4 Proof of Corollary 1

Proof. Let $\hat{\boldsymbol{\theta}}_n^{(t)}$ denote the estimate of $\boldsymbol{\theta}_n$ at iteration t, and let $\boldsymbol{\theta}_*^{(t)}$ denote its "true" value at iteration t, and let $\boldsymbol{\theta}^*$ denote its true value in the StoNet. By the proof of Theorem 4 of Liang et al. (2018) and Theorem 1 of Meinshausen and Yu (2009), for the StoNet with the linear regression output layer, we

have

$$\mathbb{E}\|\hat{\boldsymbol{\theta}}_n^{(t)} - \boldsymbol{\theta}^*\| \leq \frac{1}{1-\rho^*}\mathbb{E}\|\hat{\boldsymbol{\theta}}_n^{(t)} - \boldsymbol{\theta}_*^{(t)}\| \prec \frac{\sqrt{r_n}}{1-\rho^*}, \quad \text{as } t \to \infty, \tag{S1.2}$$

by summarizing all $d_1 + d_2 + \cdots + d_{h+1}$ linear regressions, where ρ^* is a constant as defined in Assumption 5. For the StoNet with the logistic regression output layer, we have the same result by Theorem 1 of Huang et al. (2008). Further, by Markov inequality, there exists a constant c such that

$$P\left(\|\hat{\boldsymbol{\theta}}_n^{(t)} - \boldsymbol{\theta}^*\| > c\sqrt{r_n}\right) \to 0, \quad \text{as } n \to \infty \text{ and } t \to \infty.$$

Then, by Assumption 6,

- For any $i \in \boldsymbol{\gamma}^*$, $\|\hat{\boldsymbol{\theta}}_n^{(t)} - \boldsymbol{\theta}^*\| \leq c\sqrt{r_n}$ implies $|\hat{\boldsymbol{\theta}}_{i,n}^{(t)}| > c\sqrt{r_n}$.

- For any $i \notin \boldsymbol{\gamma}^*$, $\|\hat{\boldsymbol{\theta}}_n^{(t)} - \boldsymbol{\theta}^*\| \leq c\sqrt{r_n}$ implies $|\hat{\boldsymbol{\theta}}_{i,n}^{(t)}| < c\sqrt{r_n}$.

Therefore,

$$P(\hat{\boldsymbol{\gamma}} = \boldsymbol{\gamma}^*) \geq P((\|\hat{\boldsymbol{\theta}}_n^{(t)} - \boldsymbol{\theta}^*\| \leq c\sqrt{r_n}) \to 1, \quad \text{as } n \to \infty \text{ and } t \to \infty, \tag{S1.3}$$

which concludes the proof. □

S1.5 Proof of Corollary 2

Proof. This proof involves several notations, including $\widehat{\boldsymbol{\Sigma}}$, $\hat{\boldsymbol{\theta}}$, and $\varsigma_{h+1,j}$. As noted in the main text, their dependence on the sample size n is implicit and has been depressed for notational simplicity. As implied by (S4.9)-(S4.11), we have $\widehat{\boldsymbol{\Sigma}}_i \to 0$, $i \in \{1, 2, \ldots, h\}$, as $n \to \infty$. Additionally, as $n \to \infty$,

$$\|\mu(\boldsymbol{z}, \hat{\boldsymbol{\theta}}) - \mu(\boldsymbol{z}, \boldsymbol{\theta}^*)\| \xrightarrow{p} 0,$$

for any test point $\boldsymbol{z}$, and

$$\varsigma^2_{h+1,j} - \sigma^2_{h+1} \xrightarrow{p} 0, \quad j \in \{1, 2, \ldots, d_{h+1}\}.$$

Therefore, the nominal coverage rate $1 - \alpha$ is asymptotically guaranteed as $n \to \infty$. □

S2 The Imputation Regularized-Optimization Algorithm for StoNet Training

This algorithm is given in Algorithm S1.

Algorithm S1: IRO Algorithm for StoNet

Input: Dataset $D_n = (\mathbb{Y}, \mathbb{X})$, total iteration number T, and Monte Carlo step number t_{MC}.

Initialization: Randomly initialize the network parameters $\hat{\boldsymbol{\theta}}^{(0)} = (\hat{\boldsymbol{\theta}}_1^{(0)}, \ldots, \hat{\boldsymbol{\theta}}_{h+1}^{(0)})$.

for $t = 1, 2, \ldots, T$ **do**

• **Imputation**: For each sample $(\boldsymbol{X}^{(i)}, \boldsymbol{Y}^{(i)})$, draw $\boldsymbol{Y}_{\text{mis}}^{(i,t)}$ from $\pi(\boldsymbol{Y}_{\text{mis}}^{(i)} | \boldsymbol{Y}^{(i)}, \boldsymbol{X}^{(i)}, \hat{\boldsymbol{\theta}}_n^{(t-1)}, \boldsymbol{\sigma}_n^2)$ with a Metropolis or Langevin dynamics kernel by iterating for t_{MC} steps.

• **Regularized optimization**: Based on the pseudo-complete data $\{(\boldsymbol{Y}^{(i)}, \boldsymbol{Y}_{\text{mis}}^{(i,t)}, \boldsymbol{X}^{(i)}) : i = 1, 2, \ldots, n\}$, update $\hat{\boldsymbol{\theta}}_n^{(t-1)}$ by minimizing a penalized loss function, i.e., setting

$$\hat{\boldsymbol{\theta}}_n^{(t)} = \arg\min_{\boldsymbol{\theta}} \Big\{ -\frac{1}{n} \sum_{i=1}^{n} \log \pi(\boldsymbol{Y}^{(i)}, \boldsymbol{Y}_{\text{mis}}^{(i,t)} | \boldsymbol{X}^{(i)}, \boldsymbol{\theta}, \boldsymbol{\sigma}_n^2) + P_{\lambda_n}(\boldsymbol{\theta}) \Big\}, \quad \text{(S2.4)}$$

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

$$\begin{aligned} \boldsymbol{\theta}_*^{(t)} &= \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\theta}_n^{(t-1)}} \log \pi(\boldsymbol{Y}, \boldsymbol{Y}_{\text{mis}} | \boldsymbol{X}, \boldsymbol{\theta}, \boldsymbol{\sigma}_n^2) \\ &= \arg\max_{\boldsymbol{\theta}} \int \log \pi(\boldsymbol{Y}_{\text{mis}}, \boldsymbol{Y} | \boldsymbol{X}, \boldsymbol{\theta}, \boldsymbol{\sigma}_n^2) \pi(\boldsymbol{Y}_{\text{mis}} | \boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{\theta}_n^{(t-1)}, \boldsymbol{\sigma}_n^2) \\ &\quad \times \pi(\boldsymbol{Y} | \boldsymbol{X}, \boldsymbol{\theta}^*, \boldsymbol{\sigma}_n^2) d\boldsymbol{Y}_{\text{mis}} d\boldsymbol{Y}, \end{aligned} \quad \text{(S2.5)}$$

where $\boldsymbol{\theta}_*^{(t)}$ is called the working true parameter at iteration t.

Output: $\hat{\theta}_n^{(T)}$

S3 Adaptive Stochastic Gradient MCMC for Efficient StoNet Training

The IRO algorithm requires computation on the full dataset at each iteration and, therefore, it is less scalable with respect to big data. In practice, we can train the sparse StoNet using the adaptive stochastic gradient MCMC algorithm as proposed in (Liang et al., 2022). To make the paper self-contained, we give a review of the adaptive stochastic gradient MCMC algorithm below.

Let $\pi(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{\theta},\boldsymbol{\sigma}^2) = \int \pi(\boldsymbol{Y},\boldsymbol{Y}_{\text{mis}}|\boldsymbol{X},\boldsymbol{\theta},\boldsymbol{\sigma}^2)d\boldsymbol{Y}_{\text{mis}}$ denote the likelihood function of the observed data for the StoNet model. By Fisher's identity, we have

$$\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{\theta},\boldsymbol{\sigma}^2) = \int \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{Y},\boldsymbol{Y}_{\text{mis}}|\boldsymbol{X},\boldsymbol{\theta},\boldsymbol{\sigma}^2)\pi(\boldsymbol{Y}_{\text{mis}}|\boldsymbol{Y},\boldsymbol{X},\boldsymbol{\theta},\boldsymbol{\sigma}^2)d\boldsymbol{Y}_{\text{mis}},$$

which implies the sparse StoNet can also be trained by solving the equation

$$\int \nabla_{\boldsymbol{\theta}}[\log \pi(\boldsymbol{Y},\boldsymbol{Y}_{\text{mis}}|\boldsymbol{X},\boldsymbol{\theta},\boldsymbol{\sigma}^2) + \log P_\lambda(\boldsymbol{\theta})]\pi(\boldsymbol{Y}_{\text{mis}}|\boldsymbol{Y},\boldsymbol{X},\boldsymbol{\theta},\boldsymbol{\sigma}^2)d\boldsymbol{Y}_{\text{mis}} = 0, \tag{S3.6}$$

where $P_\lambda(\boldsymbol{\theta})$ denotes a penalty function satisfying Assumption 4. By Theorem 1 of Liang et al. (2018), solving (S3.6) will lead to the same solution

as solving the optimization problem specified below:

$$\widehat{\boldsymbol{\theta}}_n^* = \arg\max_{\boldsymbol{\theta}} \Big\{ \frac{1}{n} \sum_{i=1}^{n} \log \pi(\boldsymbol{Y}^{(i)}, \boldsymbol{Y}_{\text{mis}}^{(i)} | \boldsymbol{X}^{(i)}, \boldsymbol{\theta}, \boldsymbol{\sigma}^2) + \frac{1}{n} P_\lambda(\boldsymbol{\theta}) \Big\}.$$

By Deng et al. (2019) and Liang et al. (2022), the equation (S3.6) can be solved using an adaptive stochastic gradient MCMC algorithm, which works by iterating between the following two steps:

(a) (*Sampling*) Generate $\boldsymbol{Y}_{\text{mis}}^{(k+1)}$ from a transition kernel induced by a stochastic gradient MCMC algorithm, e.g., stochastic gradient Hamilton Monte Carlo (SGHMC) (Chen et al., 2014).

(b) (*Parameter updating*) Set $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \gamma_{k+1} g(\boldsymbol{Y}_{\text{mis}}^{(k+1)}, U_{k+1})$, where γ_{k+1} denotes the step size used in the stochastic approximation procedure.

The pseudo-code of the adaptive SGHMC algorithm is given by Algorithm S2, where we let $\boldsymbol{\theta}_i = (\boldsymbol{w}_i, \boldsymbol{b}_i)$ denote the parameters associated with layer i of the StoNet, let $(\boldsymbol{Y}_0^{(s,k)}, \boldsymbol{Y}_{h+1}^{(s,k)}) = (\boldsymbol{X}^{(s)}, \boldsymbol{Y}^{(s)})$ denote a training sample s, and let $\boldsymbol{Y}_{mis}^{(s,k)} = (\boldsymbol{Y}_1^{(s,k)}, \ldots, \boldsymbol{Y}_h^{(s,k)})$ denote the latent variables imputed for the training sample s at iteration k. Occasionally, we use the notation $\boldsymbol{Y}_0^{(s,k)} = \boldsymbol{Y}_0^{(s)} = \boldsymbol{X}^{(s)}$ and $\boldsymbol{Y}_{h+1}^{(s,k)} = \boldsymbol{Y}_{h+1}^{(s)} = \boldsymbol{Y}^{(s)}$. This algorithm is called "adaptive" as the transition kernel used in step (i) changes with

iterations through the working estimate $\boldsymbol{\theta}^{(k)}$.

S4 Covariance of Latent Variables in the StoNet

Consider the case that we have a regression StoNet trained by the IRO algorithm. In this case, the prediction uncertainty can be quantified by a recursive application of Eve's law.

Let $\boldsymbol{z}$ denote a test point at which the prediction uncertainty needs to be quantified. For simplicity of notation, we suppress the bias term by including it as a special column of the corresponding weight matrix. To indicate the iterative nature of the IRO algorithm, we include the superscript 't' in the derivation. Let $\boldsymbol{Z}_i^{(t)}$ denote the imputed latent variable, corresponding to the input $\boldsymbol{z}$, for layer i at iteration t. For convenience, we let $\boldsymbol{Z}_0^{(t)} = \boldsymbol{z}$ for all t. Let $\boldsymbol{\mu}_i^{(t)}$ and $\boldsymbol{\Sigma}_i^{(t)}$ denote, respectively, the mean and covariance matrix of $\boldsymbol{Z}_i^{(t)}$. Let $\boldsymbol{w}_{i_j}^{(t)}$ denote the j-th row of the weight matrix $\boldsymbol{w}_i^{(t)}$, which represents the weights from the neurons of layer $i-1$ to neuron j of layer i. By Eve's law, for any layer $i \in \{2, 3, \ldots, h+1\}$, we

Algorithm S2: An adaptive SGHMC algorithm for training StoNet

Input: Dataset $(\boldsymbol{X}, \boldsymbol{Y})$, total iteration number K, Monte Carlo step number t_{HMC}, the learning rate sequence $\{\epsilon_{k,i} : t = 1, 2, \ldots, T; i = 1, 2, \ldots, h+1\}$, and the step size sequence $\{\gamma_{k,i} : t = 1, 2, \ldots, T; i = 1, 2, \ldots, h+1\}$.

Initialization: Randomly initialize the network parameters

$$\hat{\boldsymbol{\theta}}^{(0)} = (\hat{\boldsymbol{\theta}}_1^{(0)}, \ldots, \hat{\boldsymbol{\theta}}_{h+1}^{(0)}).$$

for $k = 1, 2, \ldots, K$ **do**

STEP 0: Subsampling: Draw a mini-batch of data and denote it by S_k.

STEP 1: Backward Sampling: For each observation $s \in S_k$, sample $\boldsymbol{Y}_i$'s in the order from layer h to layer 1. More explicitly, we sample $\boldsymbol{Y}_i^{(s,k)}$ from the distribution

$$\pi(\boldsymbol{Y}_i^{(s,k)} | \hat{\theta}_i^{(k-1)}, \hat{\theta}_{i+1}^{(k-1)}, \boldsymbol{Y}_{i+1}^{(s,k)}, \boldsymbol{Y}_{i-1}^{(s,k)}) \propto \pi(\boldsymbol{Y}_{i+1}^{(s,k)} | \hat{\theta}_{i+1}^{(k-1)}, \boldsymbol{Y}_i^{(s,k)}) \pi(\boldsymbol{Y}_i^{(s,k)} | \hat{\theta}_i^{(k-1)}, \boldsymbol{Y}_{i-1}^{(s,k)})$$

by running SGHMC for t_{HMC} steps:

Initialize $\boldsymbol{v}_i^{(s,0)} = \boldsymbol{0}$, and initialize $\boldsymbol{Y}_i^{(s,k,0)}$ by forward pass of DNN.

for $l = 1, 2, \ldots, t_{HMC}$ **do**

for $i = h, h-1, \ldots, 1$ **do**

$$\begin{aligned} \boldsymbol{v}_i^{(s,k,l)} =& (1 - \epsilon_{k,i}\eta_i)\boldsymbol{v}_i^{(s,k,l-1)} + \epsilon_{k,i} \nabla_{\boldsymbol{Y}_i^{(s,k,l-1)}} \log \pi \left(\boldsymbol{Y}_i^{(s,k,l-1)} \mid \hat{\boldsymbol{\theta}}_i^{(k-1)}, \boldsymbol{Y}_{i-1}^{(s,k,l-1)} \right) \\ & + \epsilon_{k,i} \nabla_{\boldsymbol{Y}_i^{(s,k,l-1)}} \log \pi \left(\boldsymbol{Y}_{i+1}^{(s,k,l-1)} \mid \hat{\boldsymbol{\theta}}_{i+1}^{(k-1)}, \boldsymbol{Y}_i^{(s,k,l-1)} \right) + \sqrt{2\epsilon_{k,i}\eta} \boldsymbol{e}^{(s,k,l)}, \\ \boldsymbol{Y}_i^{(s,k,l)} =& \boldsymbol{Y}_i^{(s,k,l-1)} + \epsilon_{k,i} \boldsymbol{v}_i^{(s,k,l-1)}, \end{aligned} \tag{S3.7}$$

where $\boldsymbol{e}^{s,k,l} \sim N(0, \boldsymbol{I}_{d_i})$, $\epsilon_{k,i}$ is the learning rate, and η is the friction coefficient. The algorithm is reduced to SGLD when $\epsilon_{k,i}\eta_i \equiv 1$.

Set $\boldsymbol{Y}_i^{(s,k)} = \boldsymbol{Y}_i^{(s,k,t_{HMC})}$ for $i = 1, 2, \ldots, h$.

STEP 2: Parameter Update: Update the estimates of $\hat{\boldsymbol{\theta}}^{(k-1)} = (\hat{\boldsymbol{\theta}}_1^{(k-1)}, \hat{\boldsymbol{\theta}}_2^{(k-1)}, \ldots, \hat{\boldsymbol{\theta}}_{h+1}^{(k-1)})$ by stochastic gradient descent

$$\hat{\boldsymbol{\theta}}_i^{(k)} = \hat{\boldsymbol{\theta}}_i^{(k-1)} + \gamma_{k,i} \left(\frac{n}{|S_k|} \sum_{s \in S_k} \nabla_{\boldsymbol{\theta}_i} \log \pi(Y_i^{(s,k)} | \hat{\boldsymbol{\theta}}_i^{(k-1)}, Y_{i-1}^{(s,k)}) - n \nabla_{\boldsymbol{\theta}_i} P_\lambda(\hat{\boldsymbol{\theta}}_i) \right), \tag{S3.8}$$

for $i = 1, 2, \ldots, h+1$, where $\gamma_{k,i}$ is the step size used for updating θ_i.

Output: $\hat{\theta}_n^{(K)}$

then have

$$\begin{aligned}
\boldsymbol{\Sigma}_i^{(t)} &= \mathbb{E}(\mathrm{Var}(\boldsymbol{Z}_i^{(t)}|\boldsymbol{Z}_{i-1}^{(t)})) + \mathrm{Var}(\mathbb{E}(\boldsymbol{Z}_i^{(t)}|\boldsymbol{Z}_{i-1}^{(t)})) \\
&= \mathbb{E}\mathrm{diag}\Big\{\psi(\boldsymbol{Z}_{i-1}^{(t)}))^T \mathrm{Var}(\hat{\boldsymbol{w}}_{i_j}^{(t)})\psi(\boldsymbol{Z}_{i-1}^{(t)})) : j = 1, \dots d_i\Big\} + \mathrm{Var}\Big(\mathbb{E}(\hat{\boldsymbol{w}}_i)\psi(\boldsymbol{Z}_{i-1}^{(t)})\Big) \\
&= \mathrm{diag}\Big\{tr(\mathrm{Var}(\hat{\boldsymbol{w}}_{i_j}^{(t)}))\mathrm{Var}(\psi(\boldsymbol{Z}_{i-1}^{(t)}))) + (\mathbb{E}(\psi(\boldsymbol{Z}_{i-1}^{(t)})))^T \\
&\quad \times \mathrm{Var}(\hat{\boldsymbol{w}}_{i_j}^{(t)})(\mathbb{E}(\psi(\boldsymbol{Z}_{i-1}^{(t)}))) : j = 1, \dots d_i\Big\} + \mathbb{E}(\hat{\boldsymbol{w}}_i)\mathrm{Var}(\psi(\boldsymbol{Z}_{i-1}^{(t)}))(\mathbb{E}(\hat{\boldsymbol{w}}_i))^T,
\end{aligned}$$

where $\mathrm{Var}(\hat{\boldsymbol{w}}_{i_j}^{(t)})$ is calculated by the Lasso+OLS or Lasso+mLS procedure suggested by Liu and Yu (2013). We refer to Theorem 3 of Liu and Yu (2013) for asymptotic normality of the non-sparse components of $\hat{\boldsymbol{w}}_{i_j}^{(t)}$. For the OLS case, the non-sparse submatrix of $\mathrm{Var}(\hat{\boldsymbol{w}}_{i_j}^{(t)})$ is given by

$$\widetilde{\mathrm{Var}(\hat{\boldsymbol{w}}_{i_j}^{(t)})} = \hat{\varsigma}_{i,j}^2[(\psi(\widetilde{\mathbb{Y}}_{i-1}^{(t))})^T\psi(\widetilde{\mathbb{Y}}_{i-1}^{(t)})]^{-1},$$

where $\psi(\widetilde{\mathbb{Y}}_{i-1}^{(t)})$ is the design matrix of the linear regression

$$\mathbb{Y}_{i,j}^{(t)} = \psi(\widetilde{\mathbb{Y}}_{i-1}^{(t)})(\tilde{\boldsymbol{w}}_{i_j}^{(t)})^T + \boldsymbol{\epsilon}_{i,j}$$

selected by Lasso for neuron j of layer i at iteration t, $\boldsymbol{\epsilon}_{i,j} \sim N(0, \varsigma_i^2 I_n)$, and $\hat{\varsigma}_{i,j}^2$ denotes the OLS estimator of ς_i^2. Here $\mathbb{Y}_{i-1}^{(t)} \in \mathbb{R}^{n\times d_{i-1}}$ denotes imputed latent variables for all neurons of layer $i-1$, $\mathbb{Y}_{i,j}^{(t)} \in \mathbb{R}^n$ denotes imputed latent variables for neuron j of layer i, $\widetilde{\mathbb{Y}}_{i-1}^{(t)} \in \mathbb{R}^{n\times\tilde{q}_{i,j}}$ denotes the variables selected by Lasso, $\tilde{\boldsymbol{w}}_{i_j}^{(t)}$ denotes the corresponding regression coefficients, and $\tilde{q}_{i,j}$ denotes the number of selected variables.

Let $\boldsymbol{\mu}_{i-1}^{(t)} = (\mu_{i-1,1}^{(t)}, \ldots, \mu_{i-1,d_{i-1}}^{(t)})^T$ denote the mean of $\boldsymbol{Z}_{i-1}^{(t)}$, and let $D_{\psi'}(\boldsymbol{\mu}_{i-1}^{(t)}) = \text{diag}\{\psi'(\mu_{i-1,1}^{(t)}), \ldots, \psi'(\mu_{i-1,d_{i-1}}^{(t)})\}$, where ψ' denotes the first derivative of the activation function ψ. By the first order Taylor expansion, we have

$$\mathbb{E}(\psi(\boldsymbol{Z}_{i-1}^{(t)})) \approx \psi(\boldsymbol{\mu}_{i-1}^{(t)}), \quad \text{Var}(\psi(\boldsymbol{Z}_{i-1}^{(t)})) \approx D_{\psi'}(\boldsymbol{\mu}_{i-1}^{(t)})\Sigma_{i-1}^{(t)}D_{\psi'}(\boldsymbol{\mu}_{i-1}^{(t)}).$$

Further, if we estimate $\mathbb{E}(\hat{\boldsymbol{w}}_i)$ by $\hat{\boldsymbol{w}}_i$ and estimate $\boldsymbol{\mu}_{i-1}^{(t)}$ by $\boldsymbol{Z}_{i-1}^{(t)}$, then we have the approximation:

$$\begin{aligned}\widehat{\boldsymbol{\Sigma}}_i^{(t)} \approx \text{diag}\Big\{ & tr\big(\text{Var}(\hat{\boldsymbol{w}}_{i_j}^{(t)})D_{\psi'}(\boldsymbol{Z}_{i-1}^{(t)})\widehat{\Sigma}_{i-1}^{(t)}D_{\psi'}(\boldsymbol{Z}_{i-1}^{(t)})\big) + (\psi(\boldsymbol{Z}_{i-1}^{(t)}))^T\text{Var}(\hat{\boldsymbol{w}}_{i_j}^{(t)})\psi(\boldsymbol{Z}_{i-1}^{(t)}) : j = 1, \ldots, d_i\Big\} \\ & + \hat{\boldsymbol{w}}_i^{(t)}D_{\psi'}(\boldsymbol{Z}_{i-1}^{(t)})\widehat{\Sigma}_{i-1}^{(t)}D_{\psi'}(\boldsymbol{Z}_{i-1}^{(t)})(\hat{\boldsymbol{w}}_i^{(t)})^T.\end{aligned} \tag{S4.9}$$

For the first hidden layer, it is reduced to

$$\begin{aligned}\widehat{\boldsymbol{\Sigma}}_1^{(t)} \approx \text{diag}\Big\{ & tr\big(\text{Var}(\hat{\boldsymbol{w}}_{1_j}^{(t)})\text{Var}(\boldsymbol{z})\big) + \boldsymbol{z}^T\text{Var}(\hat{\boldsymbol{w}}_{1_j}^{(t)})\boldsymbol{z} : j = 1, \ldots, d_1\Big\} \\ & + \hat{\boldsymbol{w}}_1^{(t)}\text{Var}(\boldsymbol{z})(\hat{\boldsymbol{w}}_1^{(t)})^T.\end{aligned} \tag{S4.10}$$

Since $\text{Var}(\boldsymbol{z}) = 0$ holds for any fixed test point $\boldsymbol{z}$, $\widehat{\boldsymbol{\Sigma}}_1^{(t)}$ can be further reduced to

$$\widehat{\boldsymbol{\Sigma}}_1^{(t)} \approx \text{diag}\left\{\boldsymbol{z}^T\text{Var}(\hat{\boldsymbol{w}}_{1_j}^{(t)})\boldsymbol{z} : j = 1, 2, \ldots, d_1\right\}. \tag{S4.11}$$

S4.1 More Numerical Results

Table S1: Calibration results for CIFAR10 data, where the standard deviations of the respective measures are given in parentheses.

Network	Size	Method	ACC(%)	NLL	ECE(%)	CECE(%)
DenseNet40	176K	No Calibration	**92.80 (0.08)**	0.3101 (0.0045)	4.45 (0.15)	0.95 (0.03)
		Temp. Scaling	**92.80 (0.08)**	0.2205 (0.0017)	1.23 (0.09)	0.43 (0.02)
		Matrix Scaling	92.36 (0.15)	0.2277 (0.0034)	1.28 (0.17)	0.41 (0.02)
		Focal	92.04 (0.15)	0.2377 (0.0037)	1.36 (0.09)	0.43 (0.03)
		MMCE	92.24 (0.49)	0.2362 (0.0258)	1.37 (0.35)	0.44 (0.07)
		Post-Linear	92.34 (0.25)	0.2320 (0.0106)	1.04 (0.99)	0.44 (0.17)
		Post-StoNet	92.63 (0.13)	**0.2214 (0.0044)**	**0.54 (0.07)**	**0.31 (0.05)**
ResNet110	1.7M	No Calibration	**92.70 (0.90)**	0.3359 (0.0472)	4.84 (0.63)	1.02 (0.13)
		Temp. Scaling	**92.70 (0.90)**	0.2238 (0.0267)	1.29 (0.11)	0.45 (0.03)
		Matrix Scaling	92.15 (0.42)	0.2377 (0.0096)	1.56 (0.14)	0.46 (0.02)
		Focal	91.97 (0.29)	0.2399 (0.0107)	0.87 (0.11)	0.44 (0.03)
		MMCE	91.81 (0.38)	0.2476 (0.0216)	1.57 (0.25)	0.49 (0.07)
		Post-Linear	92.59 (0.44)	0.2230(0.0108)	1.12 (0.29)	0.39 (0.03)
		Post-StoNet	92.66 (0.84)	**0.2210 (0.0269)**	**0.47 (0.23)**	**0.32 (0.06)**
WideResNet-28-10	36M	No Calibration	**95.84 (0.18)**	0.1704 (0.0037)	2.55 (0.09)	0.56 (0.01)
		Temp. Scaling	**95.84 (0.18)**	0.1468 (0.0029)	1.16 (0.05)	0.34 (0.02)
		Matrix Scaling	93.67 (0.81)	0.1961 (0.0218)	1.47 (0.18)	0.41 (0.02)
		Focal	95.43 (0.07)	0.1943 (0.0149)	6.29 (1.33)	1.37 (0.28)
		MMCE	93.68 (0.81)	0.1993 (0.0236)	1.43 (0.09)	0.46 (0.03)
		Post-Linear	95.77 (0.13)	**0.1444 (0.0060)**	0.94 (0.40)	0.30 (0.10)
		Post-StoNet	95.64 (0.12)	0.1449 (0.0018)	**0.87 (0.11)**	**0.25 (0.02)**

S5 Hyper-parameter Settings for the Numerical Experiments

For Algorithm S2, since the learning rates $\epsilon_{k,i}$'s and the latent variable variances σ_i^2's can be largely canceled at each step of latent variable imputation, their absolute values do not mean much to the convergence of the simulation. For this reason, we often set their values to be very small in our numerical experiments, which merely controls the size of random noise

added to the corresponding latent variables.

S5.1 Settings for the Illustrative Example

One-hidden-layer StoNet: we tried three parameter settings:

(i) $\sigma_2^2 = 5e{-}5$, $\sigma_1^2 = 5e{-}6$, $\epsilon_{k,1} = 5e{-}9$, $\eta_i = \frac{1}{\epsilon_{k,i}}$, $t_{HMC} = 1$, $\frac{\gamma_{k,1}}{|S_k|} = 5e{-}4$, $\frac{\gamma_{k,2}}{|S_k|} = 5e-8$, $|S_k| = 50$;

(ii) $\sigma_2^2 = 1e{-}4$, $\sigma_1^2 = 1e{-}5$, $\epsilon_{k,1} = 1e{-}8$, $\eta_i = \frac{1}{\epsilon_{k,i}}$,$t_{HMC} = 1$, $\frac{\gamma_{k,1}}{|S_k|} = 5e{-}4$, $\frac{\gamma_{k,2}}{|S_k|} = 5e-8$, $|S_k| = 50$;

(iii) $\sigma_2^2 = 2e{-}4$, $\sigma_1^2 = 2e{-}5$, $\epsilon_{k,1} = 2e{-}8$, $\eta_i = \frac{1}{\epsilon_{k,i}}$,$t_{HMC} = 1$, $\frac{\gamma_{k,1}}{|S_k|} = 5e{-}4$, $\frac{\gamma_{k,2}}{|S_k|} = 5e-8$, $|S_k| = 50$.

Two-hidden-layer StoNet: we tried three parameter settings:

(i) $\sigma_3^2 = 5e-10, \sigma_2^2 = 5e-11, \sigma_1^2 = 5e-12$, $\epsilon_{k,2} = 5e-14, \epsilon_{k,1} = 1e-14$, $\eta_i = \frac{1}{\epsilon_{k,i}}$,$t_{HMC} = 1$, $\frac{\gamma_{k,3}}{|S_k|} = 5e-6, \frac{\gamma_{k,2}}{|S_k|} = 5e-10, \frac{\gamma_{k,1}}{|S_k|} = 5e-14$, $|S_k| = 50$;

(ii) $\sigma_3^2 = 1e-9, \sigma_2^2 = 1e-10, \sigma_1^2 = 1e-11$, $\epsilon_{k,2} = 1e-13, \epsilon_{k,1} = 1e-14$, $\eta_i = \frac{1}{\epsilon_{k,i}}$,$t_{HMC} = 1$, $\frac{\gamma_{k,3}}{|S_k|} = 5e-6, \frac{\gamma_{k,2}}{|S_k|} = 5e-10, \frac{\gamma_{k,1}}{|S_k|} = 5e-14$, $|S_k| = 50$;

(iii) $\sigma_3^2 = 2e-9, \sigma_2^2 = 2e-10, \sigma_1^2 = 2e-11$, $\epsilon_{k,2} = 1e-13, \epsilon_{k,1} = 1e-14$, $\eta_i = \frac{1}{\epsilon_{k,i}}$,$t_{HMC} = 1$, $\frac{\gamma_{k,3}}{|S_k|} = 5e{-}6$, $\frac{\gamma_{k,2}}{|S_k|} = 5e{-}10$, $\frac{\gamma_{k,1}}{|S_k|} = 5e{-}14$,$|S_k| = 50$.

For both StoNets, the major difference among the settings is at σ_i's. For convenience, we call the settings (i), (ii) and (iii), respectively.

S5.2 Settings for the Experiments in Section 5

CIFAR100 and CIFAR10: Following the setting of post-calibration methods in Guo et al. (2017), we split the training data into a training set of 45,000 images and a holdout validation set of 5,000 images. The training settings for the three models are:

- *ResNet110*: The model was trained on the training set using SGD with momentum for 200 epochs with the batch size 128, momentum 0.9, and weight decay 0.0001. The learning rate was set to 0.1 for the first 80 epochs and divided by 10 at the 80-th and 150-th epochs.

- *Densenet40*: The model was trained on the training set using SGD with momentum for 300 epochs with the batch size 128, momentum 0.9, and weight decay 0.0001. The learning rate was set to 0.1 for the first 150 epochs and divided by 10 at the 150-th and 225-th epochs.

- *WideResNet-28-10*: The model was trained on the training set using SGD with momentum for 200 epochs with momentum 0.9, and weight decay 0.0005. The learning rate was set to 0.1 for the first 60 epochs and divided by 10 at 60-th, 120-th, and 160-th epochs.

After training, we extracted the outputs of the last fully connected layer of each model on the validation set, and used them as input to a StoNet model with one hidden layer, 100 hidden units, and the activation function *tanh*. The StoNet model was trained using Algorithm (S2) with the hyper-parameters as given in Table S2. The regularization parameter λ was set to $1e-4$ for CIFAR10 and $5e-5$ for CIFAR100. As a baseline, we consider the Post-Linear model. We used the outputs of the last fully connected layer of each model as input features, and trained sparse multi-class logistic regression models with LASSO penalty on the validation set. The regularization parameter is selected via a 5-fold cross-validation using the default setting in the *scikit-learn* package.

Table S2: Post-StoNet Hyper-Parameter Setting for CIFAR10 and CIFAR100 data

Hyper-Parameter	Value
$[\sigma_1^2, \sigma_2^2]$	[1e-2, 1e-3]
$\epsilon_{k,1}$	1e-7
η_1	$\frac{1}{\epsilon_{k,1}}$
t_{HMC}	1
$[\gamma_{k,1}, \gamma_{k,2}]$	$[\frac{5e-4}{5000}, \frac{5e-6}{5000}]$
$\lvert S_k \rvert$	50
$P_\lambda(\boldsymbol{\theta})$	$\lambda\|\boldsymbol{\theta}\|_1$

Regression Examples: The data sets were from UCI machine learning repository. For all experiments, we split the data into 40% as the training set, 40% as the calibration/validation set (used to fit a StoNet model for

our approach and to compute the absolute value of the residue as the non-conformity score for Split Conformal), and 20% as the test set. The random split was repeated 10 times. We reported the mean values and standard deviations of the prediction interval length and coverage rate.

We modeled each dataset using a DNN with 2 hidden layers, with 1000 and 100 hidden units respectively, and the activation function *tanh*. The DNN was trained using Adam (Kingma and Ba, 2015) with a batch size of 50 and a constant learning rate of 0.001. The algorithm was run for 1000 epochs for the Protein data set, 200 epochs for the Year data set, and 5000 epochs for other data sets. After the DNN was trained, we refit a StoNet on the calibration set using the output of the last hidden layer of the DNN as input. The StoNet had one hidden layer, 20 hidden units, and the activation function *tanh*. Algorithm S2 was used to train the StoNet with the hyper-parameters as given in Table S3. The penalty parameter λ was selected from $\{1e{-}1, 8e{-}2, 5e{-}2, 4e{-}2, 3e{-}2, 2e{-}2, 1e{-}2, 5e{-}3, 2e{-}3, 1e{-}3, 5e{-}4\}$ by 5-fold cross-validation, where we picked a value of λ such that the average coverage rate on the calibration sets were closest to the target level 90%. Specifically, we picked $\lambda = 8e-2$ for the Liver dataset, $\lambda = 8e-2$ for QSAR dataset, $\lambda = 3e-3$ for Community dataset, $\lambda = 8e-2$ for STAR

https://archive.ics.uci.edu/dataset/60/liver+disorders
https://archive.ics.uci.edu/dataset/504/qsar+fish+toxicity
https://archive.ics.uci.edu/dataset/183/communities+and+crime

dataset, $\lambda = 5e-2$ for Abalone dataset, $\lambda = 5e-2$ for Parkinson dataset, $\lambda = 5e-3$ for Power Plant dataset, $\lambda = 2e-3$ for Bike dataset, $\lambda = 5e-3$ for Protein dataset, $\lambda = 1e-3$ for Year dataset

Table S3: StoNet Hyper-Parameter Setting for UCI data sets, where N is size of the calibration data set.

Hyper-Parameter	Value
$[\sigma_1^2, \sigma_2^2]$	[1e-4, 1e-5]
$\epsilon_{k,1}$	1e-7
η_1	$\frac{1}{\epsilon_{k,1}}$
t_{HMC}	1
$[\gamma_{k,1}, \gamma_{k,2}]$	$[\frac{1e-3}{N}, \frac{1e-5}{N}]$
$\lvert S_k \rvert$	50
$P_\lambda(\boldsymbol{\theta})$	$\lambda\|\boldsymbol{\theta}\|_1$

S6 Consistency is Essential for the Validity of the Post-StoNet Approach

The parameter estimation consistency is essential for the validity of the post-StoNet approach. To demonstrate this issue, we applied the post-StoNet modeling approach to the Community dataset without regularization (i.e. setting $\lambda = 0$), which violates the sparsity condition of Theorem

https://github.com/yromano/cqr/tree/master/datasets
https://archive.ics.uci.edu/dataset/1/abalone
https://archive.ics.uci.edu/dataset/189/parkinsons+telemonitoring
https://archive.ics.uci.edu/dataset/294/combined+cycle+power+plant
https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset
https://archive.ics.uci.edu/dataset/265/physicochemical+properties+of+protein+tertiary+structure
https://archive.ics.uci.edu/dataset/203/yearpredictionmsd

1. The resulting prediction intervals have only a coverage rate of 29.25% (with a standard deviation of 3.77%).

Bibliography

Chen, T., E. Fox, and C. Guestrin (2014). Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pp. 1683–1691.

Deng, W., X. Zhang, F. Liang, and G. Lin (2019). An adaptive empirical bayesian method for sparse deep learning. *Advances in neural information processing systems 2019*, 5563–5573.

Guo, C., G. Pleiss, Y. Sun, and K. Q. Weinberger (2017). On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR.

Huang, J., S. Ma, and C.-H. Zhang (2008). The iterated lasso for high-dimensional logistic regression. *Technical report no. 392. The University of Iowa, Department of Statistics and Actuarial Sciences*.

Kingma, D. and J. Ba (2015). Adam: a method for stochastic optimization. In *International Conference on Learning Representations*.

Liang, F., B. Jia, J. Xue, Q. Li, and Y. Luo (2018). An imputation–regularized optimization algorithm for high dimensional missing data problems and beyond. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 80*, 899–926.

Liang, S., Y. Sun, and F. Liang (2022). Nonlinear sufficient dimension reduction with a stochastic neural network. *NeurIPS 35*, 27360–27373.

Liu, H. and B. Yu (2013). Asymptotic properties of Lasso+mLS and Lasso+Ridge in sparse high-dimensional linear regression. *Electronic Journal of Statistics 7*, 3124 – 3169.

Meinshausen, N. and B. Yu (2009). Lasso-type recovery of sparse representations for high-dimensional data. *Annals of Statistics 37*, 246–270.