Supplemental Materials to “Differentially Private Bayesian Neural Networks on Accuracy, Privacy and Reliability”

A Background of Differential Privacy

At the core of DP is the Gaussian mechanism which must work on functions with bounded ℓ_2 sensitivity.

Lemma 1 (Definition 3.8 & Theorem 3.22 [12]). *The ℓ_2 **sensitivity** of any function g is*

$$\Delta g = \sup_{S,S'} \|g(S) - g(S')\|_2$$

*where the supreme is over all pairs of neighboring datasets (S, S'). Consequently, the **Gaussian mechanism** which outputs*

$$\hat{g}(S) = g(S) + \sigma \Delta g \cdot \mathcal{N}(0, \mathbf{I})$$

is (ϵ, δ)-DP for some ϵ depending on (σ, n, p, δ), where the dependence is determined by the specific privacy accountant.

Notably, in the deep learning regime, the gradients may have unbounded sensitivity. Therefore, the per-sample clipping with a clipping norm C defined *ab initio* is applied to guarantee that the sensitivity of the sum of per-sample gradients is C.

As for the privacy accountant, we focus on two of the most popular privacy accountants, which may give different ϵ's for the same DP algorithm. In practice, we choose the smallest ϵ given by multiple privacy accountants as all are valid bounds of the true privacy loss.

In this work, we apply the moments accountant (MA) [1, Theorem 1 & 2] and GDP accountant [5], both implemented efficiently in the `Tensorflow Privacy` library[4]. We remark that empirically GDP always give tighter ϵ than MA. Furthermore, GDP's ϵ is explicit in terms of training parameters, yet MA and Fourier accountant [18] require numerical integral to compute ϵ and thus the characterization is implicit.

B Details of BBP and MC Dropout

B.1 BBP

To learn the hyperparameters, we minimize a KL divergence between the posterior distribution and the variational distribution, known as the ‘variational free energy’

[4] See MA in `https://github.com/tensorflow/privacy/blob/master/tensorflow_privacy/privacy/analysis/rdp_accountant.py`; GDP in `https://github.com/tensorflow/privacy/blob/master/tensorflow_privacy/privacy/analysis/gdp_accountant.py`

and its negative is the ELBO (see `http://krasserm.github.io/2019/03/14/bayesian-neural-networks/#appendix` for proof):

$$\min_\theta KL\left(q(w|\theta)\middle\|p(w|D)\right). \tag{3}$$

We can rewrite this through the following optimization problem

$$\begin{aligned}&\min_\theta KL\left(q(w|\theta)\middle\|p(w|D)\right)\\ =&KL\left(q(w|\theta)\middle\|p(w)\right)-\mathbb{E}_{q(w|\theta)}\log p(D|w)\\ =&\mathbb{E}_{q(w|\theta)}\log q(w|\theta)-\mathbb{E}_{q(w|\theta)}\log p(w)\\ &-\mathbb{E}_{q(w|\theta)}\log p(D|w)\end{aligned} \tag{4}$$

Here $KL\left(q(w|\theta)\middle\|p(w)\right)$ is called the 'complexity cost', and the term $\mathbb{E}_{q(w|\theta)}\log p(D|w)$ is called the 'likelihood cost'. This objective function can hardly be calculated, though it can be approximated by drawing $w^{(j)}$ from $q(w|\theta)$. We thus define the optimization objective as

$$\frac{1}{N}\sum_{j=1}^{N}[\log q(w^{(j)}|\theta)-\log p(w^{(j)})-\log p(D|w^{(j)})]$$

where N is the number of sampling. We further denote the summand, with ℓ being the loss, as

$$\mathcal{L}_{\text{BBP}}(D;w^{(j)},\theta):=\log q(w^{(j)}|\theta)-\log p(w^{(j)})+\ell(D;\boldsymbol{w}^{(j)})$$

To learn θ from this objective, we borrow a transformation $\sigma=\log(1+\exp(\rho))$ from [4, Section 3.2] so that we can optimize on the unbounded ρ without constraint, instead of the non-negative σ. Hence we sample from $q(w|\theta)$ by $w=\mu+\log(1+\exp(\rho))\circ\epsilon$, where $\circ$ is Hadamard prodcut and $\epsilon\sim\mathcal{N}(0,I)$, and we optimize over $\theta=(\mu,\rho)$. The gradient of the objective with respect to the mean is

$$\begin{aligned}&\frac{d\mathcal{L}_{\text{BBP}}(D;\theta)}{d\mu}=\frac{d\mathcal{L}_{\text{BBP}}(D;\mu,\rho)}{d\mu}\\ =&\frac{1}{N}\sum_{j=1}^{N}\left(\frac{\partial\mathcal{L}_{\text{BBP}}(D;\boldsymbol{w}^{(j)},\mu,\rho)}{\partial\boldsymbol{w}^{(j)}}+\frac{\partial\mathcal{L}_{\text{BBP}}(D;\boldsymbol{w}^{(j)},\mu,\rho)}{\partial\mu}\right)\end{aligned} \tag{5}$$

The gradient with respect to the standard deviation term is

$$\begin{aligned}&\frac{d\mathcal{L}_{\text{BBP}}(D;\theta)}{d\rho}=\frac{d\mathcal{L}_{\text{BBP}}(D;\mu,\rho)}{d\rho}\\ =&\frac{1}{N}\sum_{j=1}^{N}\left(\frac{\partial\mathcal{L}_{\text{BBP}}(D;\boldsymbol{w}^{(j)},\mu,\rho)}{\partial\boldsymbol{w}^{(j)}}\frac{\epsilon}{1+\exp(-\rho)}+\frac{\partial\mathcal{L}_{\text{BBP}}(D;\boldsymbol{w}^{(j)},\mu,\rho)}{\partial\rho}\right)\end{aligned} \tag{6}$$

This objective leads to the SGD updating rule as

$$\mu_t = \mu_{t-1} - \frac{\eta_t}{|B_t|} \sum_{i \in B_t} \frac{d\mathcal{L}_{\text{BBP}}(\boldsymbol{x}_i, y_i)}{d\mu},$$
$$\rho_t = \rho_{t-1} - \frac{\eta_t}{|B_t|} \sum_{i \in B_t} \frac{d\mathcal{L}_{\text{BBP}}(\boldsymbol{x}_i, y_i)}{d\rho} \tag{7}$$

B.2 MC Dropout

We review the MC Dropout [13] for the case of a single hidden layer. Mathematically, any NN with dropout is approximated to a probabilistic Gaussian process model, which has a close relationship with BNN. We show this connection in the context of the regression problem.

Suppose the input $\mathbf{x_i}$ is an $1 \times Q$ vector, the output $\mathbf{y_i}$ is an $1 \times D$ vector and the hidden layer include K units. We denote the two weight matrices by $\mathbf{W_1}$ and $\mathbf{W_2}$ which connect the first layer to the hidden layer and the hidden layer to the output layer respectively. $\sigma(\cdot)$ is some element-wise non-linear function such as RelU (rectified linear). $\mathbf{b}$ refers to the biases controlling the input location of each layer. Therefore, the output is $\hat{\mathbf{y}} = \sigma(\mathbf{x}\mathbf{W_1} + \mathbf{b})\mathbf{W_2}$. When applying dropout, we first sample two binary vectors $z_1 = (z_{1,1}, \dots, z_{1,Q})$ and $z_2 = (z_{2,1}, \dots, z_{2,K})$ with $z_{1,q} \sim \text{Bernoulli}(p_1)$ for $q = 1, \dots, Q$, $z_{2,k} \sim \text{Bernoulli}(p_2)$ for $k = 1, \dots, K$. Now, the output with dropout is given by $\hat{\mathbf{y}} = (\sigma((\mathbf{x} \circ z_1)\mathbf{W_1} + \mathbf{b}) \circ z_2)\mathbf{W_2}$. It is mathematiclaly equivalent to $\hat{\mathbf{y}} = \sigma(\mathbf{x}(z_1\mathbf{W_1}) + \mathbf{b})(z_2\mathbf{W_2})$, which multiplies the weight matrices with the binary vector by row.

For the regression problem, the NN model is often to optimize the following objective:

$$\mathcal{L}_{\text{Dropout}} = \frac{1}{2N} \sum_{i=1}^{N} \|y_i - \hat{y}_i\|_2^2 + \lambda_1 \|\mathbf{W_1}\|_2^2 + \lambda_2 \|\mathbf{W_2}\|_2^2 + \lambda_3 \|\mathbf{b}\|_2^2, \tag{8}$$

with regularization parameters λ_i for $i = 1, 2, 3$. Now we would apply the Gaussian Process (GP) to the NN model described above to see why NN with dropout is equivalent to BNN. First, define the covariance function:

$$K(\mathbf{x}, \mathbf{y}) = \int p(w)p(b)\sigma(w^T x + b)\sigma(w^T y + b)dwdb$$

with $1 \times Q$ standard multivariate normal distribution $p(w)$ and some distribution $p(b)$. The Monte Carlo approximation to this covariance function is given by:

$$\hat{K}(\mathbf{x}, \mathbf{y}) = \frac{1}{K} \sum_{k=1}^{K} \sigma(w_k^T x + b_k)\sigma(w_k^T y + b_k)$$

with $w_k \sim p(w)$ and $b_k \sim p(b)$. Therefore, our NN model with Gaussian process is equivalent to the following generative model:

$$\begin{aligned} \mathbf{W_1} = [w_k]_{k=1}^K, \mathbf{b} = [b_k]_{k=1}^K \\ w_k \sim p(w), b_k \sim p(b) \\ \mathbf{F}(\mathbf{X})|\mathbf{X}, \mathbf{W_1}, \mathbf{b} \sim N(0, \hat{K}(\mathbf{X}, \mathbf{X})) \\ \mathbf{Y}|\mathbf{F}(\mathbf{X}) \sim N(\mathbf{F}(\mathbf{X}), \tau^{-1}\mathbf{I}_n), \end{aligned}$$

from which the predictive distribution is given by:

$$\begin{aligned} p(\mathbf{Y}|\mathbf{X}) = & \int p(\mathbf{Y}|\mathbf{F}(\mathbf{X}))p(\mathbf{F}(\mathbf{X})|\mathbf{W_1}, \mathbf{b}, \mathbf{X})p(\mathbf{W_1})p(\mathbf{b})d\mathbf{W_1}d\mathbf{b}d\mathbf{F}(\mathbf{X}) \\ = & \int \mathcal{N}(\mathbf{Y}; 0, \varPhi\varPhi^T + \tau^{-1}\mathbf{I}_n)p(\mathbf{W_1})p(\mathbf{b})d\mathbf{W_1}d\mathbf{b}, \end{aligned}$$

where $\hat{K}(\mathbf{X}, \mathbf{X}) = \varPhi\varPhi^T$. The normal distribution of $\mathbf{Y}$ could be viewed as a joint normal distribution over the column of the $N \times D$ matrix $\mathbf{Y}$. In particular, we introduce a $K \times 1$ standard multivariate normal variable w_d and each term in the joint distribution is:

$$\mathcal{N}(y_d; 0, \varPhi\varPhi^T + \tau^{-1}\mathbf{I}_n) = \int \mathcal{N}(y_d; \varPhi w_d, \tau^{-1}\mathbf{I}_n)\mathcal{N}(w_d; 0, \mathbf{I}_K)dw_d.$$

Therefore, the predictive distribution could be written in the following way:

$$p(\mathbf{Y}|\mathbf{X}) = \int p(\mathbf{Y}|\mathbf{X}, \mathbf{W_1}, \mathbf{W_2}, \mathbf{b})p(\mathbf{W_1})p(\mathbf{W_2})p(\mathbf{b})d\mathbf{W_1}d\mathbf{W_2}d\mathbf{b}, \qquad (9)$$

where $\mathbf{W_2} = [w_d]_{d=1}^D$. Naturally, the expression (9) could be viewed as BNN with multivariate standard normal distributions on the weights. To connect with the idea of dropout, we proceed with the variational inference. Suppose we use the variational distribution $q(\mathbf{W_1}, \mathbf{W_2}, \mathbf{b}) = q(\mathbf{W_1})q(\mathbf{W_2})q(\mathbf{b})$ to approximate the posterior distribution $p(\mathbf{W_1}, \mathbf{W_2}, \mathbf{b}|X, Y)$. In particular, the variational distribution on the weight matrix is factorized over the rows and each term is a mixture of normal distributions, one centered at 0 and the other centered away from 0:

$$\begin{aligned} q(\mathbf{W_1}) = & \prod_{q=1}^{Q} q(\mathbf{w}_q), \\ q(\mathbf{w}_q) = & \; p_1\mathcal{N}(\mathbf{m}_{1,q}, \sigma^2\mathbf{I}_k) + (1-p_1)\mathcal{N}(0, \sigma^2\mathbf{I}_k), \end{aligned}$$

where p_1 refers to dropout rate of the first layer, $\sigma > 0$ and $\mathbf{m}_q \in \mathbb{R}^K$. A similar distribution is assigned to $\mathbf{W}_2$. And the variational distribution $q(\mathbf{b})$ of the bias $\mathcal{N}(\mathbf{m}, \sigma^2\mathbf{I}_k)$. Given the definition above, $\mathbf{W_1}$ corresponds to a location matrix $\mathbf{M_i} = [\mathbf{m}_{1,1}, \mathbf{m}_{1,2}, \ldots, \mathbf{m}_{1,Q}]^T$. Similar for $\mathbf{W_2}$. The variational Bayes is aimed to minimize the Kullback–Leibler (KL) divergence between the variational distribution and the posterior distriburion, which results in the following objective function:

$$\mathcal{L}_{bnn}(x_i; \hat{\boldsymbol{w}}_{t-1}^{(n)}) = \frac{1}{N}\sum_{n=1}^{N} \frac{-\log p(y_n|x_n, \hat{\boldsymbol{w}}_n)}{\tau} + \sum_{i=1}^{2} \frac{p_i l^2}{2\tau N} \|\mathbf{M_i}\|_2^2 + \frac{1}{2\tau N} \|\mathbf{m}\|_2^2 , \tag{10}$$

where $\hat{\boldsymbol{w}}_n$ is sampled from $q(\mathbf{W}_1)$. Compared to Equation (9), setting parameters appropriately will lead two optimization problems equivalent. More detailed explanation could be found in [13].

C Experiments

C.1 Classifiction

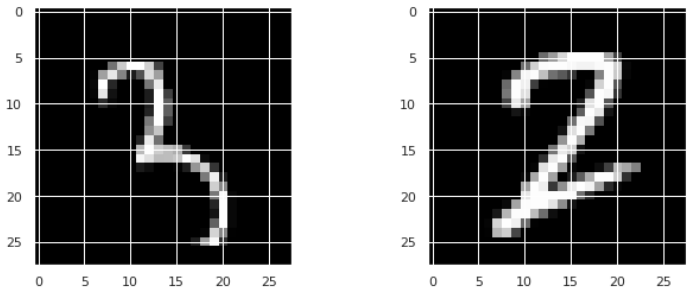

Fig. 5: Two pictures from MNIST: the input image 3 is difficult to predict (low probability in the true class across all three methods); the input image 2 is easy to predict.

In Section 5.1, we trained our algorithms on the MNIST digits dataset, in which each image is labelled with some number in between zero to nine. Methods we considered are with almost identical privacy budgets. Specifically, DP-SGLD has $\epsilon_{\text{GDP}} = 0.861$ or $\epsilon_{\text{MA}} = 0.989$; other DP models have $\epsilon_{\text{GDP}} = 0.834$ or $\epsilon_{\text{MA}} = 0.955$. We consider two NNs with different architectures: multi-layer perceptron (MLP) and convolutional neural network (CNN). The softmax output layers have ten units, corresponding to possible labels. For MLP, there are two hidden layers, both containing 1200 units and activated by ReLU. For CNN, we use the benchmark architecture in `Opacus` and `Tensorflow Privacy` libraries. For the evaluation of calibration, by splitting the predictions into M equally-spaced bins $\{B_m\}$, we have

$$\text{ECE} = \sum_{m\in[M]} \frac{|B_m|}{n}\Big|\text{acc}(B_m) - \text{conf}(B_m)\Big|, \text{MCE} = \max_{m\in[M]}\Big|\text{acc}(B_m) - \text{conf}(B_m)\Big|,$$

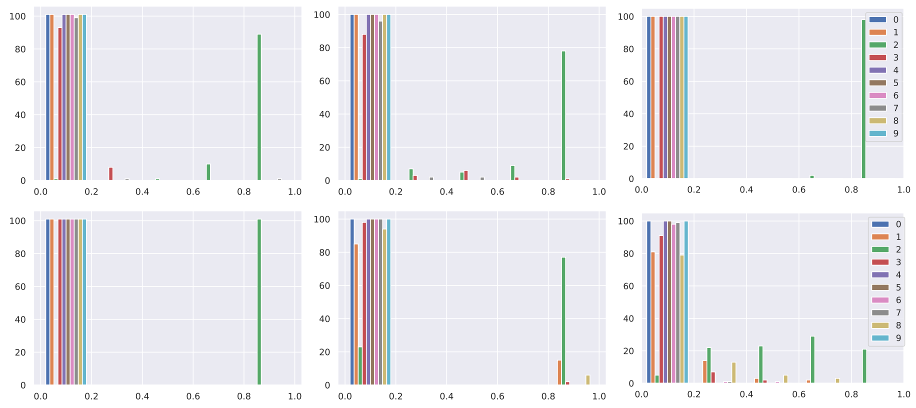

Fig. 6: y-axis refers to the frequency and x-axis refers to the prediction probability. Prediction distribution on MNIST over 100 repeated samplings (input image is 2, see Figure 5). Left to right: SGLD, BBP, MC Dropout. Upper: non-DP BNNs. Lower: DP-BNNs.

where acc is the average accuracy and conf is the average confidence.

Given the specific structure of NN, DP-BBP and DP-SGLD assign either the Gaussian prior $\mathcal{N}(0, 0.1^2)$ or the Laplacian prior $L(0, 0.1)$ to each weight. For DP-MC Dropout, the dropout rate is 0.5. After careful hyperparameter tuning, we select the following hyperparameters pairs (learning rate, batch size) for each method: $(2 \times 10^{-4}, 256)$ for DP-MC Dropout, $(0.25, 256)$ for DP-BBP and $(5 \times 10^{-6}, 256)$ for DP-SGLD. We set the privacy $\delta = 10^{-5}$ and the clipping norm as 1.5. For DP-BBP and DP-MC Dropout, the noise scale is 1.3. Experiments are conducted over 15 epochs.

To show the prediction uncertainty in Figure 7 and Figure 6, we select two digits (an easy-to-predict digit 2 and a hard-to-predict digit 3) from the test dataset, shown in Figure 5.

Fundamentally, the randomness of BNN comes from the weight uncertainty. For BBP, the weight uncertainty is from the posterior and we sample weights from their posterior distribution. For MC Dropout, the weight uncertainty is from the dropout and each time we randomly drop out the units of the trained NNs with the probability 0.5. For SGLD, the weight uncertainty is from the weight updating rule and we record the last 100 weight updates from the last epoch. Finally, for all methods, we collect the posterior probabilities of each class over 100 independent predictions.

In Figure 6, we plot the empirical distribution of each class (denoted by different colors) over 100 predictions. The x axis refers to the predicted probability and the y axis refers to the frequency. Similar as Figure 7, DP influences the posterior probability in different ways. However, even if three methods predict with different uncertainty, their predictions are correct no matter for DP or non-DP.

C.2 Regression

For the heteroscedasticity regression problem in Section 5.2, we used a network with two hidden layers of 200 rectified linear units (ReLU), same as in the MNIST experiment. For both BBP and SGLD, we only consider the weights with the Gaussian prior. We generate 400 data points, 250 for training and 150 for testing. Each input x is sampled from the distribution Uniform$(-3, 3)$, while the output $\boldsymbol{y} = (y_1, y_2, \ldots, y_{400})$ follows the multivariate normal distribution, whose covariance matrix is a function of $\boldsymbol{x}$. In the simulation study, this covariance matrix is the summation of the radial basis function kernel (RBF) with variance 1, i.e. Kernel(x, x') and a diagonal matrix whose diagonal element is $(0.3x + 0.6)^2$. See the public notebook `https://github.com/JavierAntoran/Bayesian-Neural-Networks/blob/master/notebooks/regression/gp_homo_hetero.ipynb` for the code implementation. Notice that the output of the neural network has two elements: the prediction $\hat{y}_i$ and the noise estimation $\hat{\sigma}_i^2$.[5]

In Figure 4, we introduce two types of uncertainty. The data uncertainty is calculated by $\frac{1}{n}\sum_{i=1}^{n}\hat{\sigma}_i^2$. The posterior uncertainty is calculated by $\dfrac{1}{n-1}\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2$, where $\bar{y} = \dfrac{1}{n}\sum_{i=1}^{n}\hat{y}_i$.

- MC Dropout: noise multiplier $\sigma = 10$, clipping norm $C = 2000$, learning rate 0.00005, dropout rate 0.5;
- SGLD: clipping norm $C = 100$, learning rate 0.00025;
- BBP: noise multiplier $\sigma = 10$, clipping norm $C = 100$, learning rate 0.01;

C.3 Effects of batch size and learning rate

In Figure 2, we empirically study the effects of batch size and learning rate on DP-SGLD and general DP-SGD. We use the standard DP CNN in `Opacus` library[6] and train with DP-SGD, which includes the DP-SGLD by Theorem 1. To be specific, for DP-SGLD, we set the number of epochs as 15, the clipping norm $C = 1.5$, and the noise scale as $\sigma = \frac{|B|}{\sqrt{60000 \times \eta} \times 1.5}$ in the DP-SGD; for general DP-SGD, we set the same number of epochs and clipping norm, but use a noise scale $\sigma = 1.3$, which is the benchmark in `Opacus` and `Tensorflow Privacy` libraries, achieving around 95.0% test accuracy with batch size 256.

When the batch size varies, we fix the learning rate at 0.25 for DP-SGD and 0.25/60000 for DP-SGLD; when the learning rate varies, we fix the batch size as 256.

[5] See `https://github.com/JavierAntoran/Bayesian-Neural-Networks/blob/master/notebooks/regression/bbp_hetero.ipynb` for the network architecture.

[6] See `https://github.com/pytorch/opacus/blob/master/examples/mnist.py`

D Additional Proofs

D.1 Proof of Theorem 1

We start with stating the updating rules for both DP-SGD in Algorithm 1 and DP-SGLD in Algorithm 2.

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} - \eta_t \left(\frac{1}{|B|} \sum_{i \in B} \widetilde{g}_i + \frac{\sigma \cdot C_t}{|B|} \cdot \mathcal{N}(0, I_d) + \nabla_{\boldsymbol{w}} r(\boldsymbol{w}_{t-1}) \right), \text{(DP-SGD)}$$
$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} - \eta_t \left(\frac{n}{|B|} \sum_{i \in B} \widetilde{g}_i + \sqrt{\eta_t}\mathcal{N}(0, I_d) + \nabla_{\boldsymbol{w}} r(\boldsymbol{w}_{t-1}) \right), \quad \text{(DP-SGLD)}$$

where $|B|, C_t, \sigma, \widetilde{g}_i, \eta_t$ are defined in Section 2. By matching the coefficients of these updating rules, it is easy to see

$$\eta_{\text{SGD}} = \eta_{\text{SGLD}} \cdot n, \qquad \frac{\sigma_{\text{SGD}} C_{\text{SGD}}}{|B|} = \sqrt{\eta_{\text{SGLD}}}.$$

Additionally, the clipping is performed with the same gradient norm, hence $C_{\text{SGLD}} = C_{\text{SGD}}$. Therefore, we obtain

$$\begin{aligned} &\text{DP-SGLD}\,(\eta_{\text{SGLD}} = \eta, C_{\text{SGLD}} = C) \\ \equiv &\text{DP-SGD}\Big(\eta_{\text{SGD}} = \eta n, \sigma_{\text{SGD}} = \frac{|B|}{n\sqrt{\eta}C}, C_{\text{SGD}} = C\Big), \\ &\text{DP-SGD}\,(\eta_{\text{SGD}} = \eta, \sigma_{\text{SGD}} = \sigma, C_{\text{SGD}} = C) \\ \equiv &\text{DP-SGLD}\Big(\eta_{\text{SGLD}} = \frac{\eta}{n}, C_{\text{SGLD}} = C = \frac{|B|}{\sqrt{n\eta}\sigma}\Big). \end{aligned}$$

D.2 Proof of Theorem 3

Viewing DP-SGLD as DP-SGD, and pluging the coefficients $\sigma = \frac{|B|}{n\sqrt{\eta}C}$ from Theorem 1 in Theorem 2, we then get the desired result.

D.3 Larger batch size is more private in DP-SGLD

From Theorem 3, the privacy loss in GDP is $\sqrt{T(e^{n^2\eta C^2/|B|^2} - 1)}|B|/n$. Denoting $|B|/n$ as $0 < x \le 1$, we aim to show $\sqrt{T(e^{\eta C^2/x^2} - 1)}x$ is decreasing in x:

$$\frac{d}{dx}\sqrt{T(e^{\eta C^2/x^2} - 1)}x = \sqrt{T} \cdot \frac{\Big(e^{\eta C^2/x^2}(1 - \frac{\eta C^2}{x^2}) - 1\Big)}{\sqrt{(e^{\eta C^2/x^2} - 1)}} \le 0$$

because $e^x(1-x)-1 \le 0$ for any $x \in \mathbb{R}$. This fact can be checked by the derivative of $e^x(1-x)-1$ which indicates the only stationary point is $x = 0$, and that $e^0(1-0)-1 = 0$. Hence $\sqrt{T(e^{n^2\eta C^2/|B|^2} - 1)}|B|/n$ is decreasing in $|B|$.

E Additional plots and tables

Methods	DP-ECE	DP-MCE	Non-DP ECE	Non-DP MCE
SGLD (w/ prior)	0.003	0.775	0.001	0.011
SGLD (w/o prior)	0.043	0.371	0.006	0.219
MC Dropout (w/ prior)	0.001	0.275	0.030	0.325
MC Dropout (w/o prior)	0.033	0.230	0.003	0.225
SGD (w/ prior)	0.005	0.391	0.002	0.059
SGD (w/o prior)	0.037	0.365	0.014	0.325

Table 4: Calibration errors of SGLD, MC Dropout, SGD, and their DP counterparts on MNIST with four-layer CNN, with or without Gaussian prior.

	DP-SGLD	DP-BBP	DP-MC Dropout
Variational inference	No	Yes	Yes
Sampling approach	Yes	No	No
Optimizing KL divergence	No	Yes	No
General network structure	Yes	No	Yes
General optimizers	N/A	Yes	Yes
General weight prior	Yes	Yes	No
Memory consumption	High	Low	Low
Computation complexity	Low	High	Low
Analytic posterior	No	Yes	No

Table 5: Algorithmic comparison of DP-BNNs.

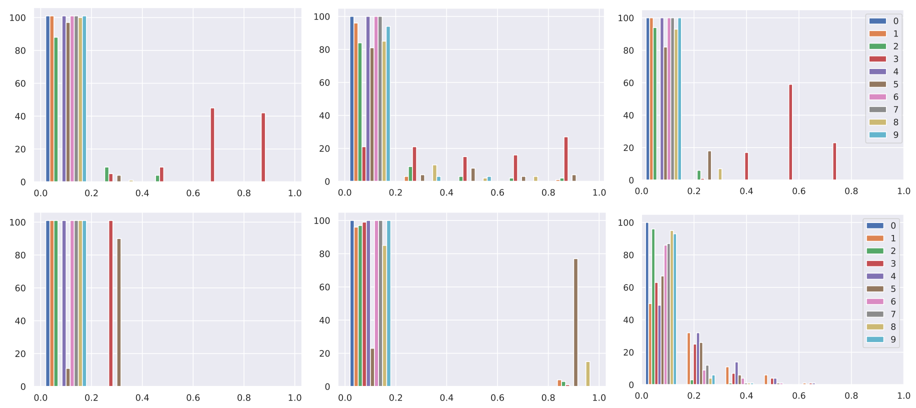

Fig. 7: Prediction distribution on MNIST with two-layer MLP over 100 repeated samplings (input image is 3, see Figure 5). Y-axis refers to the frequency and x-axis refers to the prediction probability. Left to right: SGLD, BBP, MC Dropout. Upper: non-DP BNNs. Lower: DP-BNNs.

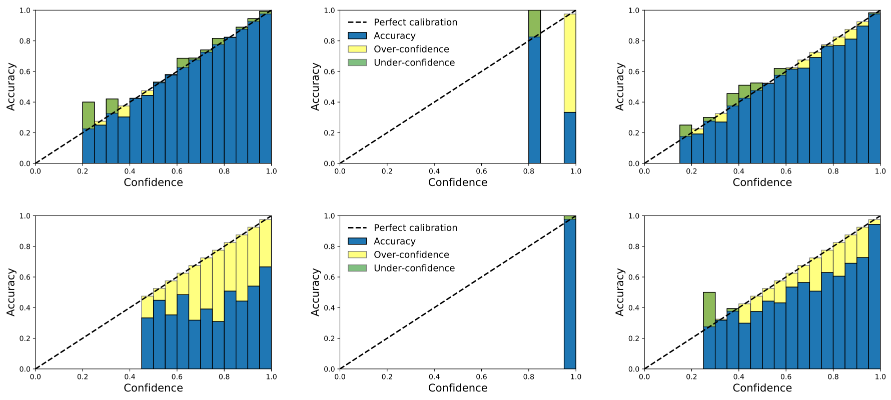

Fig. 8: Reliability diagram on MNIST with two-layer MLP under DP regime. Left to right: SGLD, BBP, MC Dropout. Upper: with Gaussian prior. Lower: without Gaussian prior.

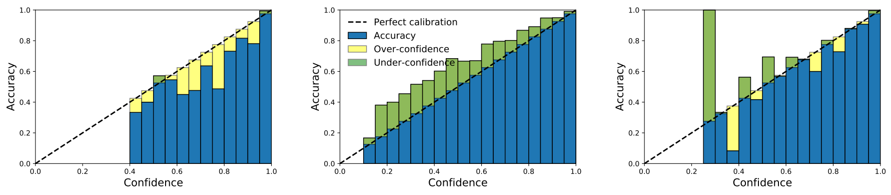

Fig. 9: Reliability diagram on MNIST with two-layer MLP under non-DP regime without prior. Left to right: SGLD, BBP, MC Dropout.

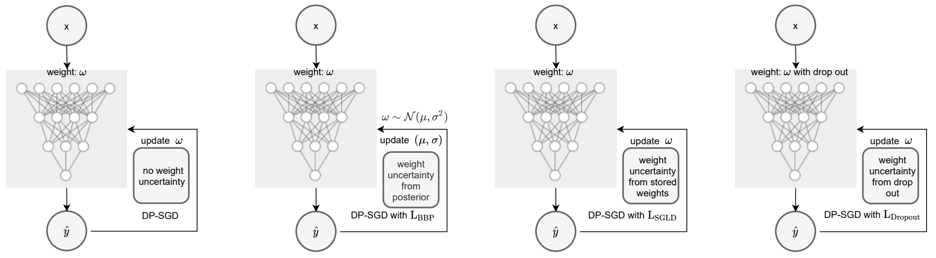
x
weight: ω
update ω
no weight uncertainty
DP-SGD
$\hat{y}$
x
weight: ω
$\omega \sim \mathcal{N}(\mu, \sigma^2)$
update (μ, σ)
weight uncertainty from posterior
DP-SGD with $\mathrm{L}_{\mathrm{BBP}}$
$\hat{y}$
x
weight: ω
update ω
weight uncertainty from stored weights
DP-SGD with $\mathrm{L}_{\mathrm{SGLD}}$
$\hat{y}$
x
weight: ω with drop out
update ω
weight uncertainty from drop out
DP-SGD with $\mathrm{L}_{\mathrm{Dropout}}$
$\hat{y}$

Fig. 10: Training procedure of private SGD, BBP, SGLD, and MC Dropout (left to right). Applying non-DP optimizers instead results in the regular training.

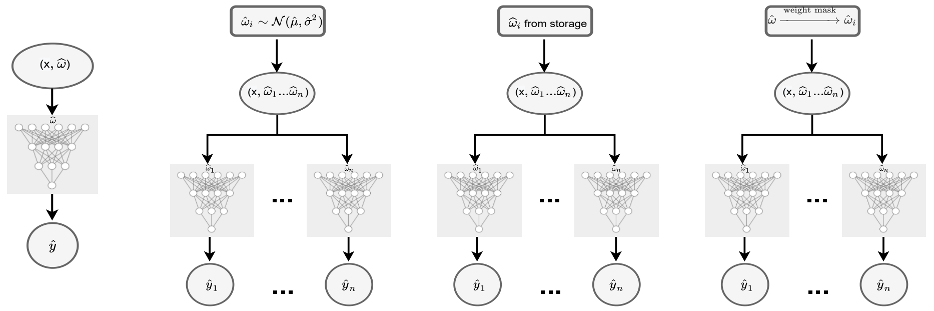
(x, $\widehat{\omega}$)
$\widehat{\omega}$
$\hat{y}$
$\hat{\omega}_i \sim \mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$
(x, $\widehat{\omega}_1 \ldots \widehat{\omega}_n$)
$\widehat{\omega}_1$
$\widehat{\omega}_n$
...
$\hat{y}_1$
$\hat{y}_n$
$\widehat{\omega}_i$ from storage
(x, $\widehat{\omega}_1 \ldots \widehat{\omega}_n$)
$\hat{y}_1$
$\hat{y}_n$
$\widehat{\omega} \xrightarrow{\text{weight mask}} \widehat{\omega}_i$
(x, $\widehat{\omega}_1 \ldots \widehat{\omega}_n$)
$\hat{y}_1$
$\hat{y}_n$

Fig. 11: Inference (or prediction) procedure of non-Bayesian NN and BNNs: BBP, SGLD, and MC Dropout (left to right). Note that DP-BNNs have the same procedure as regular BNNs, as DP is enforced during the training procedure.