Supplemental Material: Learning Weight Uncertainty with Stochastic Gradient MCMC for Shape Classification

Chunyuan Li, Andrew Stevens, Changyou Chen, Yunchen Pu, Zhe Gan, Lawrence Carin
Duke University
{cl319, ajs104, cc448, yp42, zg27, lcarin}@duke.edu

1. Hyper-parameter setting

We first discuss the hyper-parameter settings for SG-MCMC, and then provide their values for our experiments. For detailed concepts in MCMC, we suggest [1]. For a review of recent advances in SG-MCMC algorithms, we recommend [2].

1.1. Discussion of Hyper-parameters

Step Size The step size ϵ for SGLD and pSGLD corresponds to the optimization counterparts. Similar to RMSprop, the default step size value for pSGLD is 10^{-3}. The step size for SGLD should be tuned case by case, similar to SGD, but we have found that 10^{-1} is a good starting point. A block decay strategy is used on several datasets, it decreases by the stepsize by half every L epochs.

Mini-batch Size The gradient at step t is evaluated on a batch of data $\mathcal{S}_t$. For small datasets, the batch size can be set to the training sample size $|\mathcal{S}_t| = N$, giving the true gradient for each step. This recovers the MCMC with Langevin dynamics [3]. For large datasets, a stochstic gradient evaluted from a mini-batch of size $|\mathcal{S}_t| < N$ is used to approximate true gradient.

Burn-in In theory, MCMC can reach the equilibrium distribution asymptotically. In practice, we only have a limited amount of time. MCMC usually starts from a "poor" starting point, which may over-sample regions that are actually very low probability under the equilibrium distribution before it settles into the equilibrium distribution. To get parameter samples from regions of higher probability, we need to throw away samples at the beginning of an MCMC run, prior to collection, this is called "burn-in".

Thinning In the main text, for sample-based uncertainty estimation (9) of the posterior predictive distribution, making predictions using all collected samples from the posterior of DNNs can be computationally expensive. Due to the fact of high autocorrelation time between samples in SG-MCMC methods, we suggest to thin the Markov chain which leaves fewer, less correlated samples to evaluate (9). As with conventional MCMC, these thinned samples have a lower autocorrelation time and can help maintain a higher effective sample size while reducing the computational burden.

Variance of Gaussian Prior The prior distributions on the weights of DNNs are Gaussian, with mean 0 and variance σ^2. The variance of this Gaussian distribution determines the prior belief of how strongly these weights should concentrate on 0. This setting depends on user perception of the amount variability existing in the data. A larger variance in the prior leads to a wider range of weight choices, thus higher uncertainty. The weight decay value of ℓ_2 regularization in stochastic optimization is related to the prior variance in SG-MCMC.

Table 1. FNN: Hyper-parameter settings of pSGLD for different datasets.

Datasets	MNIST	Animal	Body	Geometry	Textured	20Newsgroup
Batch Size	100	200	60	96	96	100
Step Size	10^{-3}	5×10^{-4}	10^{-3}	10^{-3}	10^{-3}	10^{-4}
# Total Epoch	100	300	1000	500	100	100
Burn-in (#Epoch)	1	10	1	1	1	50
Thinning Interval (#Epoch)	1/6	2	5	5	5	1/10
Block Decay (#Epoch)	20	300	1000	100	20	100
Variance in prior	1	10	1	1	10	10

Table 2. CNN: Hyperparameter settings of pSGLD for different datasets.

Datasets	MNIST		Caltech		ShapeNets		Cifar10	
Networks	LeNet	4-CNN	5-CNN	5-CNN-BN	Vol-CNN	View-CNN	VGG-B	VGG-B-BN
Batch Size	100	100	128	128	128	60	128	128
Step Size	10^{-3}	10^{-3}	5×10^{-4}	5×10^{-4}	10^{-5}	10^{-4}	10^{-3}	10^{-3}
# Total Epoch	100	100	50	50	60	30	300	300
Burn-in (#Epoch)	1	10	10	10	20	10	20	20
Thinning Interval (#Epoch)	1/6	1/6	1	1	1	1	1	1
Block Decay (#Epoch)	20	20	20	20	25	10	25	25
Variance in prior	1	1	100	100	1	1	1	1

1.2. Settings in Our Experiments

The hyper-parameter settings of pSGLD on each dataset is specified in Table 1 for FNN and Table 2 for CNN.

We found the settings for RMSprop and then used them for pSGLD. The settings are specified to be the same for a fair comparison: step size, batch size, total epoch and block decay are the same. The weight decay value of ℓ_2 regularization in RMSProp is set to the reciprocal of the prior variance, a proper correspondence of the Bayesian setup.

In the CNN experiments, when Dropout is applied, we follow [4] to set up probability p in Dropout: lower probabilities (0.2) are set to drop out hideen units on the layers closer to the data, and higher probabilities (0.5) on units further away from the data.

2. Network Configuration

The network architectures of the CNNs employed in the main text are detailed in Table 3.

3. Further Results on ShapeNets

The confusion matrices for *View-CNN* and *Vol-CNN* on `ShapeNets` are shown in Fig. 1. Both approches have excellent performance on most of the classes. As expected, both of them show poor performance distinguishing flower-pots and plants. This is understandable since there are plants in the flower-pot objects. Interestingly, the view-based method and the volume-based method have a complementary deep representation ability on several classes. For example, *Vol-CNN* can better recognize box, radio, cup, curtain, bench *etc*., while *View-CNN* can better classify table, sofa, bookshelf, chair *etc*. An interesting direction for future work is joint learning of a deep representation that combines information from both methods.

References

[1] S. Brooks, A. Gelman, G. Jones, and X.-L. Meng. *Handbook of Markov Chain Monte Carlo*. CRC press, 2011. 1

[2] Y. A. Ma, T. Chen, and E. B. Fox. A complete recipe for stochastic gradient MCMC. *NIPS*, 2015. 1

[3] R. M. Neal. MCMC using Hamiltonian dynamics. *Handbook of MCMC*, 2011. 1

[4] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 2014. 2

Table 3. Network configuration of CNN on `MNIST`, `Caltech`, `ShapeNets` and `Cifar10`. The convolutional layer parameters are denoted as "Conv ⟨number of filters⟩ - ⟨filter size⟩ - ⟨striding size⟩ - ⟨padding size⟩". Every convolutional layer is followed by the element-wise ReLU activation function, which is not shown for brevity. "BN" denotes Batch-Normalization, with "✓" means applied, "✗" otherwise.

Dataset	`MNIST`		`Caltech`		`ShapeNets`		`Cifar10`	
Networks	LeNet	4-CNN	5-CNN	5-CNN-BN	Vol-CNN	View-CNN	VGG-B	VGG-B-BN
Conv	32-5-1-4	32-5-1-4	64-3-1-2	64-3-1-2	48-6-2-2	96-7-2-2	64-3-1-1	64-3-1-1
BN	✗	✗	✗	✓	✗	✓	✗	✓
Max Pooling	2	2				2		
Conv	64-5-1-4	64-5-1-4	64-3-1-2	64-3-1-2	160-5-2-2	256-5-2-1	64-3-1-1	64-3-1-1
BN	✗	✗	✗	✓	✗	✓	✗	✓
Max Pooling	2	2				2		2
Conv		64-3-1-1	64-3-1-2	64-3-1-2	512-4-1-1	512-3-1-1	128-3-1-1	128-3-1-1
BN		✗	✗	✓	✗		✗	✓
Max Pooling			2	2				
Conv		64-3-1-1	128-3-1-2	128-3-1-2		512-3-1-1	128-3-1-1	128-3-1-1
BN		✗	✗	✓			✗	✓
Max Pooling		2						2
Conv			128-3-1-2	128-3-1-2		512-3-1-1	256-3-1-1	256-3-1-1
BN			✗	✓		✓	✗	✓
Max Pooling			2	2		2		
Conv							256-3-1-1	256-3-1-1
BN							✗	✓
Conv							256-3-1-1	256-3-1-1
BN							✗	✓
Conv							512-3-1-1	512-3-1-1
BN							✗	✓
Max Pooling							2	2
Conv							512-3-1-1	512-3-1-1
BN							✗	✓
Conv							512-3-1-1	512-3-1-1
BN							✗	✓
Max Pooling							2	2
Conv							512-3-1-1	512-3-1-1
BN							✗	✓
Conv							512-3-1-1	512-3-1-1 s
BN							✗	✓
Conv							512-3-1-1	512-3-1-1
BN							✗	✓
Max Pooling							2	2
ReLU	200	200	512	512	1200	4096	512	512
BN	✗	✗	✗	✓	✓	✓	✗	✓
ReLU	200	200	512	512	4000	1000		
BN	✗	✗	✗	✓	✓	✓		

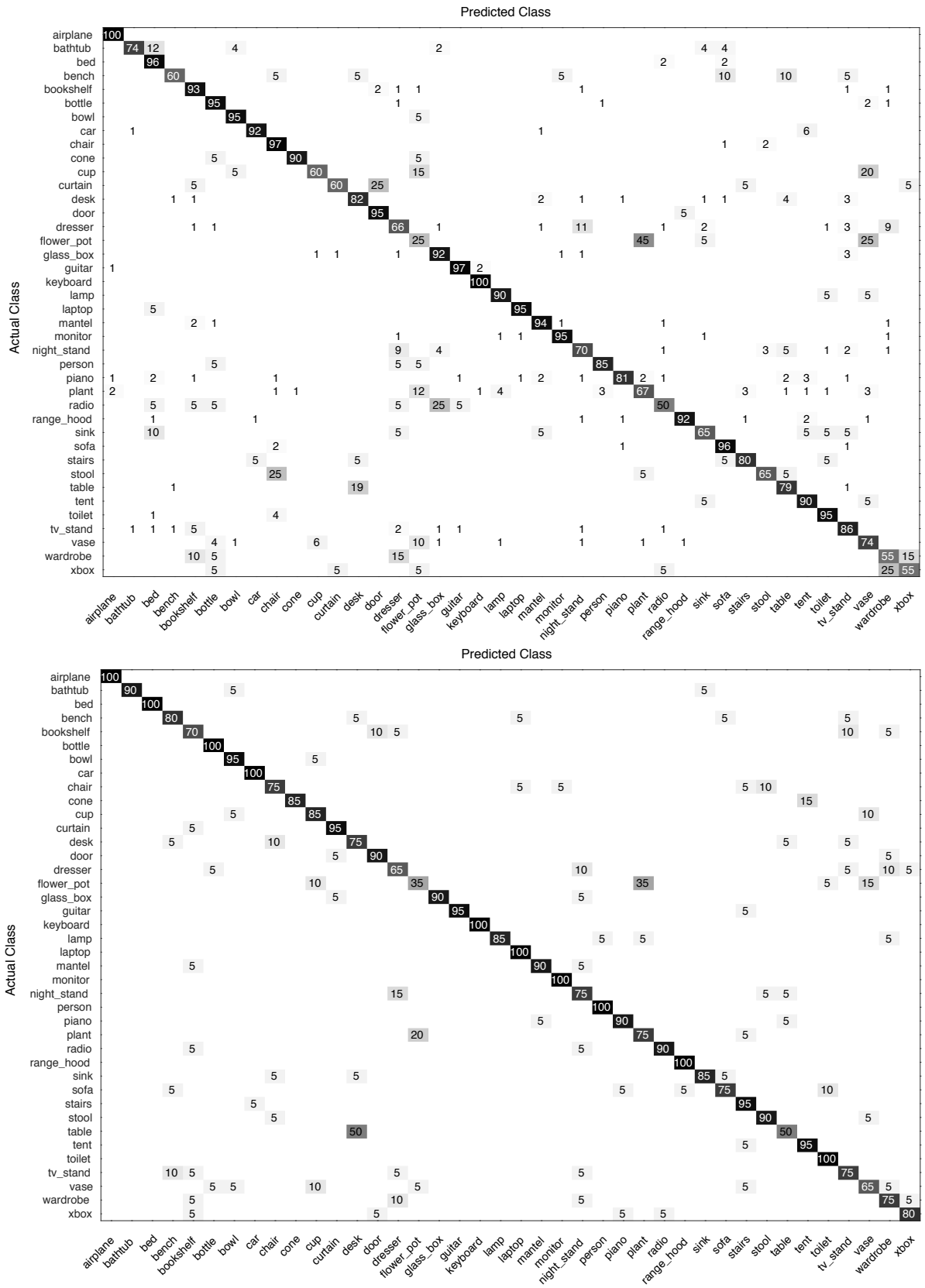

Figure 1. Consufion matrices for *Vol-CNN* (top) and *View-CNN* (bottom) on `ShapeNets`