# Supplementary Material:
# Variational Bayesian Dropout with a Hierarchical Prior

Yuhang Liu[1,2], Wenyong Dong[1], Lei Zhang[3,2], Dong Gong[2], and Qinfeng Shi[2]
[1]School of Computer Science, Wuhan University, Hubei, China
[2]The University of Adelaide, Australia [3]Inception Institute of Artificial Intelligence, UAE
https://sites.google.com/view/yuhangliu/pro

## 1. The derivation of Variational Bayesian Dropout

With the proposed hierarchical prior $p(\mathbf{W},\gamma)=p(\mathbf{W}|\gamma)p(\gamma)$, variational posterior distribution $q(\mathbf{W},\gamma)=q(\mathbf{W})q(\gamma)$, and the original objective:

$$D_{KL}(q(\mathbf{W})||p(\mathbf{W},\gamma|\mathcal{D})), \tag{1}$$

we have:

$$\begin{aligned}
&\min_{\alpha,\theta,\gamma} D_{KL}(q(\mathbf{W},\gamma)||p(\mathbf{W},\gamma|\mathcal{D}))\\
=&\min_{\alpha,\theta,\gamma}\int q(\mathbf{W},\gamma)\log\frac{q(\mathbf{W},\gamma)}{p(\mathbf{W},\gamma|\mathcal{D})}\\
=&\min_{\alpha,\theta,\gamma}\int q(\mathbf{W},\gamma)\log q(\mathbf{W},\gamma)-\int q(\mathbf{W},\gamma)\log p(\mathbf{W},\gamma|\mathcal{D})\\
=&\max_{\alpha,\theta,\gamma}\int q(\mathbf{W},\gamma)\log p(\mathbf{W},\gamma|\mathcal{D})-\int q(\mathbf{W},\gamma)\log q(\mathbf{W},\gamma)\\
=&\max_{\alpha,\theta,\gamma}\int q(\mathbf{W},\gamma)\log p(\mathbf{W},\gamma)p(\mathcal{D}|\mathbf{W})-\int q(\mathbf{W},\gamma)\log p(\mathcal{D})-\int q(\mathbf{W},\gamma)\log q(\mathbf{W},\gamma)\\
\equiv&\max_{\alpha,\theta,\gamma}\int q(\mathbf{W},\gamma)\log p(\mathcal{D}|\mathbf{W})+\int q(\mathbf{W},\gamma)\log p(\mathbf{W},\gamma)-\int q(\mathbf{W},\gamma)\log q(\mathbf{W},\gamma)\\
=&\max_{\alpha,\theta,\gamma}\int q(\mathbf{W},\gamma)\log p(\mathcal{D}|\mathbf{W})-\int q(\mathbf{W},\gamma)\log\frac{q(\mathbf{W},\gamma)}{p(\mathbf{W},\gamma)}
\end{aligned} \tag{2}$$

As mentioned in the manuscript, we assume that $q(\mathbf{W},\gamma)=q(\mathbf{W})q(\gamma)$, hence we have:

$$\begin{aligned}
&\max_{\alpha,\theta,\gamma}\int q(\mathbf{W},\gamma)\log p(\mathcal{D}|\mathbf{W})-\int q(\mathbf{W},\gamma)\log\frac{q(\mathbf{W},\gamma)}{p(\mathbf{W},\gamma)}\\
=&\max_{\alpha,\theta,\gamma}\int q(\mathbf{W})q(\gamma)\log p(\mathcal{D}|\mathbf{W})-\int q(\mathbf{W})q(\gamma)\log\frac{q(\mathbf{W})q(\gamma)}{p(\mathbf{W}|\gamma)p(\gamma)}\\
=&\max_{\alpha,\theta,\gamma}\int q(\mathbf{W})\log p(\mathcal{D}|\mathbf{W})-\int q(\mathbf{W})q(\gamma)\log\frac{q(\mathbf{W})}{p(\mathbf{W}|\gamma)}-\int q(\mathbf{W})q(\gamma)\log\frac{q(\gamma)}{p(\gamma)}\\
=&\max_{\alpha,\theta,\gamma}L_{\mathcal{D}}(\alpha,\theta)-D_{KL}(q(\mathbf{W})||p(\mathbf{W}|\gamma))-D_{KL}(q(\gamma)||p(\gamma))
\end{aligned} \tag{3}$$

## 2. The proof of Proposition 2

To proof **Proposition 2**, we only need to proof that $D_{KL}(q(\gamma)||p(\gamma))$ becomes a constant and can be neglected during optimization. To this end, we employ mean-field variational approximation viz., $q(\gamma) = \prod_{k=1}^{K}\prod_{d=1}^{D} q(\gamma_{k,d})$, to gather with $p(\gamma) = \prod\limits_{k=1}^{K}\prod\limits_{d=1}^{D} p(\gamma_{k,d}) = \prod\limits_{k=1}^{K}\prod\limits_{d=1}^{D} \mathcal{U}(\gamma_{k,d}|a,b)$, we have:

$$D_{KL}(q(\gamma)||p(\gamma)) = \prod_{k=1}^{K}\prod_{d=1}^{D} D_{KL}(q(\gamma_{k,d})||p(\gamma_{k,d})), \tag{4}$$

Now, we can directly investigate $D_{KL}(q(\gamma_{k,d})||p(\gamma_{k,d}))$,

$$\begin{aligned} & D_{KL}(q(\gamma_{k,d})||p(\gamma_{k,d})) \\ =& \int q(\gamma_{k,d}) \log \frac{q(\gamma_{k,d})}{p(\gamma_{k,d})} \\ =& \int q(\gamma_{k,d}) \log q(\gamma_{k,d}) - \int q(\gamma_{k,d}) \log p(\gamma_{k,d}) \\ =& -\mathcal{H}(\gamma_{k,d}) - \int q(\gamma_{k,d}) \log p(\gamma_{k,d}), \end{aligned} \tag{5}$$

where $\mathcal{H}(\gamma_{k,d})$ denotes the entropy of a random variable $\gamma_{k,d}$. As defined in main paper, $q(\gamma_{k,d})$ obeys a delta distribution, hence $\mathcal{H}(\gamma_{k,d}) = 0$ because delta distribution do not provide any uncertainty [4]. As a result, we obtain:

$$D_{KL}(q(\gamma_{k,d})||p(\gamma_{k,d})) = -\int q(\gamma_{k,d}) \log p(\gamma_{k,d}), \tag{6}$$

To simplify the problem, assuming the dimension of $\gamma_{k,d}$ is 1, e.g., $p(\gamma_{k,d}) = 1/(b-a)$, thus we have:

$$\begin{aligned} & D_{KL}(q(\gamma_{k,d})||p(\gamma_{k,d})) \\ =& -\int_{-\infty}^{a} q(\gamma_{k,d}) \log p(\gamma_{k,d}) - \int_{a}^{b} q(\gamma_{k,d}) \log p(\gamma_{k,d}) - \int_{b}^{+\infty} q(\gamma_{k,d}) \log p(\gamma_{k,d}) \\ =& -\log\beta \int_{-\infty}^{a} q(\gamma_{k,d}) - \log(1/(b-a)) \int_{a}^{b} q(\gamma_{k,d}) - \log\beta \int_{b}^{+\infty} q(\gamma_{k,d}), \end{aligned} \tag{7}$$

where $\beta \to 0$ (Strictly speaking, $\beta$ should be equal to 0, since $p(\gamma_{k,d}) = 0$ in the intervals $(-\infty, a]$ and $[b, +\infty)$). Thus, if $q(\gamma_{k,d})$ as a delta function lies in [a, b], e.g., $\delta(\gamma_{k,d} - \gamma'_{k,d})$ and $\gamma'_{k,d}$ lies in [a,b], then $\int_{-\infty}^{a} q(\gamma_{k,d})$ and $\int_{b}^{+\infty} q(\gamma_{k,d})$ is equal to 0, so $D_{KL}(q(\gamma_{k,d})||p(\gamma_{k,d})) = \log(b-a)$, else, $\int_{a}^{b} q(\gamma_{k,d})$ is equal to 0, and $D_{KL}(q(\gamma_{k,d})||p(\gamma_{k,d})) = +\infty$. To avoid $D_{KL}(q(\gamma_{k,d})||p(\gamma_{k,d})) = +\infty$, $[a, b]$ is generally regarded as a large enough interval, so the KL is a constant $\log(b-a)$ and can be neglected.

## 3. The experimental setting on MNIST Dataset

Following the suggestion in [6], we adopt a fully connected neural network consisting of 3 hidden layers and rectified linear units as the basic deep neural network. For the dropout with Bernoulli noise or Gaussian noise, we fix the dropout rate as $p = 0.5$ for the hidden layers and $p = 0.2$ for the input layer as recommended in the references. All networks are trained for 50 epochs with batch size 1000 and Adam optimizer [1] with initial learning rate $0.001$. To reduce the influence of random initialization, we train the network for each methods within 5 random runs and report the average results of those 5 trained networks during testing.

## 4. The experimental setting on Cifar-10 Dataset

Following the setting in [2], the basic neural network for all methods consists of two convolutional layers followed by two fully connected layers. In these two convolutional layer, there are $32 \times scale$ and $64 \times scale$ feature maps, respectively. The

kernels size and stride parameter is $3 \times 3$ and 2. Each convolutional layer is further fed into a softplus unit for improving non-linearity. In addition, each fully connected layer contains $128 \times scale$ hidden units. In experiments, we fix the dropout rate as $0.5$ for the dropout with Bernoulli noise or Gaussian noise. For each method, the network is trained for 100 epochs with batch size 1000 and Adam optimizer with initial learning rate $0.001$. Similarly, we also report the average test result for each method through training the network in 5 random runs.

## 5. Structured compressing for VGG on Cifar-10 Dataset

To prove that VBD scales to deep architectures, we apply it to a VGG-like network [1] that was adapted for the CIFAR dataset. The network consists of 13 convolutional and two fully-connected layers, trained with pre-activation batch normalization and Binary Dropout. At the start of the training procedure, we use pre-trained weights for initialization. We compare the proposed method with existing variational dropout based methods, Variational Dropout (VD) [3] and SBP [5]. Table 1 shows the advantage of the proposed VBD in structured compressing for the deep VGG-like architecture, compared with the others.

Table 1. Compressing VGG on Cifar-10 Dataset.

| Methods | Error % | Neurons per Layer |
|---|---|---|
| Original | 7.2 | 64 - 64 - 128 - 128 - 256 - 256 - 256 - 512 - 512 - 512 - 512 - 512 - 512 - 512 |
| VD | 7.2 | 64 - 62 - 128 - 126 - 234 - 155 - 31 - 81 - 76 - 9 - 138 - 101 - 413 - 373 |
| SBP(a) | 7.5 | 64 - 62 - 128 - 126 - 234 - 155 - 31 - 79 - 73 - 9 - 59 - 73 - 56 - 27 |
| SBP(b) | 9.0 | 44 - 54 - 92 - 115 - 234 - 155 - 31 - 76 - 55 - 9 - 34 - 35 - 21 - 280 |
| Ours | 8.8 | 38 - 62 - 127 - 128 - 229 - 193 - 93 - 32 - 16 - 11 - 5 - 8 - 12 - 12 |

## References


[1] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[2] D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583, 2015. 2

[3] D. Molchanov, A. Ashukha, and D. Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2498–2507, 2017. 3

[4] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. 2

[5] K. Neklyudov, D. Molchanov, A. Ashukha, and D. P. Vetrov. Structured bayesian pruning via log-normal multiplicative noise. In *Advances in Neural Information Processing Systems*, pages 6775–6784, 2017. 3

[6] 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*, 15(1):1929–1958, 2014. 2


[1] http://torch.ch/blog/2015/07/30/cifar.html