

# Structured Dropout Variational Inference for Bayesian Neural Networks

Son Nguyen<sup>◊</sup> Duong Nguyen<sup>‡</sup> Khai Nguyen<sup>◊</sup> Nhat Ho<sup>†</sup> Khoat Than<sup>‡</sup> Hung Bui<sup>◊</sup>

VinAI Research, Vietnam<sup>◊</sup>; University of Texas, Austin<sup>†</sup>;  
Hanoi University of Science and Technology<sup>‡</sup>  
June 21, 2021

## Abstract

Approximate inference in deep Bayesian networks exhibits a dilemma of how to yield high fidelity posterior approximations while maintaining computational efficiency and scalability. We tackle this challenge by introducing a novel variational structured approximation inspired by the Bayesian interpretation of Dropout regularization. Concretely, we focus on the inflexibility of the factorized structure in Dropout posterior and then propose an improved method called *Variational Structured Dropout* (VSD). VSD employs an orthogonal transformation to learn a structured representation on the variational noise and consequently induces statistical dependencies in the approximate posterior. Theoretically, VSD successfully addresses the pathologies of previous Variational Dropout methods and thus offers a standard Bayesian justification. We further show that VSD induces an adaptive regularization term with several desirable properties which contribute to better generalization. Finally, we conduct extensive experiments on standard benchmarks to demonstrate the effectiveness of VSD over state-of-the-art variational methods on predictive accuracy, uncertainty estimation, and out-of-distribution detection.

## 1 Introduction

Bayesian Neural Networks (BNNs) [45, 59] offer a probabilistic interpretation for deep learning models by imposing a prior distribution on the weight parameters and aiming to infer a posterior distribution instead of only point estimates. By marginalizing over this posterior for prediction, BNNs perform a procedure of ensemble learning. These principles improve the model generalization, robustness and allow for uncertainty quantification. However, computing exactly the posterior of non-linear BNNs is infeasible and approximate inference has been devised. The core challenge is how to construct an expressive approximation for the true posterior while maintaining computational efficiency and scalability, especially for modern deep learning architectures.

Variational inference is a popular deterministic approximation approach to deal with this challenge. The first practical methods were proposed in [20, 6, 36], in which the approximate posterior is assumed to be a fully factorized distribution, also called mean-field variational inference. In general, the mean-field approximation family encourages several advantages in inference including computational tractability and effective optimization with the stochastic gradient-based methods. However, it ignores the strong statistical dependencies among random weights of neural nets, leading to the inability to capture the complicated structure of the true posterior and to estimate the true model uncertainty.

To overcome this limitation, many extensive studies proposed to approximate the true posterior with richer expressiveness. For instance, [43] treats the weight matrix as a whole via a matrix variate Gaussian [22] and approximates the posterior based on this parametrization. Several later

works have exploited this distribution to investigate different structured representations for the variational Gaussian posterior, such as Kronecker-factored [83, 67, 68], k-tied distribution [73], non-centered or rank-1 parameterization [19, 13]. Another original idea to represent the true covariance matrix of Gaussian posterior is by employing the low-rank approximation [63, 32, 76]. For robust approximation with multimodality, [44] adopted hierarchical variational model framework [65] for inferring an implicit marginal distribution in high dimensional Bayesian setting. Despite significant improvements in both predictive accuracy and uncertainty estimation, some of these methods incur a large computational complexity and are difficult to integrate into deep convolutional networks.

**Motivations.** In this paper, we approach the structured posterior approximation in Bayesian neural nets from a different perspective which has been inspired by the Bayesian interpretation of Dropout training [70, 47]. More specifically, the methods proposed in [36, 18] reinterpret Dropout regularization as approximate inference in deep Bayesian models and base on this connection to learn a variational Dropout posterior over the weight parameters. From the literature, inference approaches based on Bayesian Dropout have shown competitive performances in terms of predictive accuracy on various tasks, even compared to the structured Bayesian methods aforementioned, but with much cheaper computational complexity. Moreover, with the solid and intriguing theories on effective regularization [77, 24, 79], generalization bound [48, 55], convergence rate or robust optimization [51, 50], Dropout principle offers several potentials to further improve approximate inference in deep Bayesian nets. However, as these Bayesian Dropout methods also employed simple structures of the mean-field family, their approximations often fails to obtain satisfactory uncertainty estimates [15]. In addition, the Variational Dropout methods based on multiplicative Gaussian noise also suffer from theoretical pathologies, including improper prior leading to ill-posed true posterior, and singularity of the approximate posterior making the variational objective undefined [29].

**Contributions.** With the above insights, we propose a novel structured variational inference framework based on Dropout principle. Our method adopts an orthogonal approximation called Householder transformation to learn a structured representation for multiplicative Gaussian noise in Variational Dropout method [36, 53]. As a consequence of the Bayesian interpretation, we go beyond the mean-field family and obtain a variational Dropout posterior with structured covariance. Furthermore, to make our approximation more expressive, we deploy a hierarchical Dropout procedure, which is equivalent to inferring a joint posterior in a hierarchical Bayesian nets. We name the proposed method as *Variational Structured Dropout* (VSD) and summarize its advantages as follows:

1. Our structured approximation is implemented on low dimensional space of variational noise with considerable computational efficiency. VSD can be employed for deep CNNs in a direct way while maintaining the backpropagation in parallel and optimizing efficiently with gradient-based methods.
2. VSD has a standard Bayesian justification, in which our method can overcome the critiques from the non-Bayesian perspective of previous Variational Dropout methods. Our inference framework uses a proper prior, non-singular approximate posterior and derives a tractable variational lower bound without further simplified approximation.
3. We investigate the inductive bias induced by the adaptive regularization of structured Dropout noise. We also provide an interpretation that VSD implicitly facilitates the networks to converge to a local minima with smaller spectral norms and stable rank. This properties suggests better generalization and we present empirical results to support this implication.
4. Finally, we carry out extensive experiments with standard datasets and network architectures to validate the effectiveness of our method on many criteria, including scalability, predictive accuracy, calibration, and out-of-distribution detection, in comparison to popular variational inference methods.

**Notation.** For a matrix  $\mathbf{A}$ ,  $\|\mathbf{A}\|_F$  and  $\mathbf{A}^\top$  denotes the Frobenius norm and the transpose matrix,  $\mathbf{A}_{i:}$  and  $\mathbf{A}_{:j}$  denote the  $i$ -th row and the  $j$ -th column. For an interger  $i$ ,  $\mathbf{e}_i$  is the  $i$ -th standard basis,  $\mathbf{1}_i \in \mathbb{R}^i$  is the vector of all ones. The diagonal matrix with diagonal entries as the elements of a vector  $\mathbf{x}$  is denoted by  $\text{diag}(\mathbf{x})$ . The inner product between two matrices  $\mathbf{A}$  and  $\mathbf{B}$  is denoted by  $\langle \mathbf{A}, \mathbf{B} \rangle$ .

## 2 Background

**Variational inference for Bayesian neural networks:** Given a dataset  $\mathcal{D}$  consisting of input-output pairs  $(\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ . In BNNs, we impose a prior distribution over random weights  $p(\mathbf{W})$  whose form is in a tractable parametric family and aim to infer the intractable true posterior  $p(\mathbf{W}|\mathcal{D})$ . Variational inference (VI) [28, 31] can do this by specifying a variational distribution  $q_\phi(\mathbf{W})$  with free parameter  $\phi$  and then minimizing the Kullback-Leibler (KL) divergence  $\mathbb{D}_{KL}(q_\phi(\mathbf{W})\|p(\mathbf{W}|\mathcal{D}))$ . This optimization is equivalent to maximizing the Evidence Lower Bound (ELBO) with respect to variational parameters  $\phi$  as follows:

$$\mathcal{L}(\phi) = \mathbb{E}_{q_\phi(\mathbf{W})} \log p(\mathcal{D}|\mathbf{W}) - \mathbb{D}_{KL}(q_\phi(\mathbf{W})\|p(\mathbf{W})). \quad (1)$$

By leveraging the reparameterization trick [37] combined with the Monte Carlo integration, we can derive an unbiased differentiable estimation for the variational objective above. Then, this estimation can be effectively optimized using stochastic gradient methods with the variance reduction technique such as the *local* reparameterization trick [36].

**Variational Bayesian inference with Dropout regularization:** Given a deterministic neural net with the weight parameter  $\Theta$  of size  $K \times Q$ , training this model with stochastic regularization techniques such as Dropout [27, 70] can be interpreted as approximate inference in Bayesian probabilistic models. This is because injecting a stochastic noise into the input layer is equivalent to multiplying the rows of the subsequent deterministic weight by the same random variable, namely with each datapoint  $(\mathbf{x}_n, \mathbf{y}_n)$  and a noise vector  $\xi$ , we have:  $\mathbf{y}_n = (\mathbf{x}_n \odot \xi)\Theta = \mathbf{x}_n \text{diag}(\xi)\Theta$ . This induces a BNN with random weight matrix defined by  $\mathbf{W} = \text{diag}(\xi)\Theta$ . Applying VI to this Bayesian model, with some specific choices for prior and approximate posterior, the variational lower bound (1) can resemble the form of the Dropout objective in the original deterministic network. This principle is referred to as the KL condition [16]. Gal et al. [18], Kingma et al. [36] used this principle to propose Bayesian Dropout inference methods such as MC Dropout (MCD) and Variational Dropout (VD).

Dropout inference is practical approximate framework especially in high dimensional setting. However, the scope of Bayesian inference in these methods is restricted in terms of flexibility of the prior and approximate posterior. Concretely, the Dropout posteriors  $q_\phi(\mathbf{W})$  in MCD and VD both have simple structures of mean-field approximation which often underestimate the variance of true posterior, possibly leading to a poor uncertainty representation [15]. Moreover, in theory, VD employed an improper log-uniform prior which can result in an ill-posed true posterior and generally push the parameters towards the penalized maximum likelihood solution [29]. Finally, VD also suffers from the singularity issue of approximate posterior that makes the KL divergence term undefined. Our work gains an efficient remedy to these pathologies.

### 3 Variational Structured Dropout

We focus on Bayesian Dropout methods using multiplicative Gaussian noise with correlated parameterization [36], namely  $\mathbf{W} = \text{diag}(\xi)\Theta$  with the Dropout noise  $\xi$  is a multivariate Gaussian with diagonal covariance  $q_\alpha(\xi) = \mathcal{N}(\mathbf{1}_K, \text{diag}(\alpha))$  and  $\alpha$  is the dropout vector. Then, the Dropout posterior  $q_\phi(\mathbf{W})$  exhibits a factorized structure on each column with the form of  $q(\mathbf{W}_{:j}) = \mathcal{N}(\Theta_{:j}, \text{diag}(\alpha \odot \Theta_{:j}^2))$ , whilst allowing a correlation on each row because each scalar noise  $\xi_i$  is shared across the row  $\mathbf{W}_{i:}$  respectively. The parameters  $\phi = (\alpha, \Theta)$  are then optimized via maximizing a variational lower bound as follows:

$$\begin{aligned}\mathcal{L}(\phi) &:= \mathbb{E}_{q_\phi(\mathbf{W})} \log p(\mathcal{D}|\mathbf{W}) - \mathbb{D}_{KL}(q_\phi(\mathbf{W})||p(\mathbf{W})) \\ &= \mathbb{E}_{q_\alpha(\xi)} \log p(\mathcal{D}|\xi, \Theta) - \mathbb{D}_{KL}(q_\phi(\mathbf{W})||p(\mathbf{W})),\end{aligned}\tag{2}$$

where the later equation is derived from the change of variables formula.

#### 3.1 The orthogonal approximation for variational structured noise

Intuitively, a richer representation for the noise distribution can enrich the expressiveness of Dropout posterior via the Bayesian interpretation. We implement this intuition with an assumption that the Dropout noise is sampled from a Gaussian distribution with a full covariance matrix instead of a diagonal structure, namely,  $q_\Sigma(\xi) = \mathcal{N}(\mathbf{1}_K, \Sigma)$  with  $\Sigma$  is a positive definite matrix of size  $K \times K$ . To make this covariance matrix learnable, we first represent  $\Sigma$  in the form of the spectral decomposition:  $\Sigma = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^T$ , where  $\mathbf{P}$  is an orthogonal matrix with its eigenvectors in columns,  $\mathbf{\Lambda}$  is a diagonal matrix where diagonal elements are the eigenvalues. By the basis-kernel representation theorem [5, 72], we parameterize the orthogonal matrix  $\mathbf{P}$  as a product of Householder matrices in the following form:

$$\mathbf{P} = \mathbf{H}_T \mathbf{H}_{T-1} \dots \mathbf{H}_1,\tag{3}$$

where  $\mathbf{H}_t = \mathbf{I} - 2\mathbf{v}_t\mathbf{v}_t^T/\|\mathbf{v}_t\|_2^2$ ,  $\mathbf{v}_t$  is the Householder vector of size  $K$ , and  $T$  is the degree of  $\mathbf{P}$ . This parameterization relaxes the orthogonal constraint of matrix  $\mathbf{P}$ , and we can then directly optimize the covariance matrix  $\Sigma$  via gradient-based methods.

Notably, this transformation can be interpreted as a sequence of invertible mappings. More explicitly, we extract a zero-mean Gaussian noise  $\eta^{(0)}$  from the original noise  $\xi^{(0)} \sim \mathcal{N}(\mathbf{1}_K, \text{diag}(\alpha))$  in the form of  $\xi^{(0)} = \mathbf{1} + \eta^{(0)}$ , and by successively transforming  $\eta^{(0)}$  through a chain of  $T$  Householder reflections, we obtain the induced noise and the corresponding density at each step  $t$  as follows:

$$\xi^{(t)} := \mathbf{1} + \mathbf{H}_t \mathbf{H}_{t-1} \dots \mathbf{H}_1 \eta^{(0)} = \mathbf{1} + \mathbf{U} \eta^{(0)}, \quad \text{and} \quad q_t(\xi) := \mathcal{N}(\mathbf{1}_K, \mathbf{U} \text{diag}(\alpha) \mathbf{U}^T).$$

By injecting the structured noise  $\xi^{(t)}$  into the deterministic weight  $\Theta$ , we obtain a random weight  $\mathbf{W}^{(t)} = \text{diag}(\xi^{(t)})\Theta$  and a structured Dropout posterior  $q_t(\cdot)$  with *fully correlated* representation, in which the marginal column distribution is given by:  $q_t(\mathbf{W}_{:j}) = \mathcal{N}(\Theta_{:j}, \text{diag}(\Theta_{:j})\mathbf{U} \text{diag}(\alpha) \mathbf{U}^T \text{diag}(\Theta_{:j}))$ . Detailed discussion about the expressiveness of this correlated structure is presented in Appendix A.1. With the above derivations, we use variational inference with approximate posterior  $q_t(\mathbf{W})$  and optimize the variational lower bound as follows:

$$\begin{aligned}\mathcal{L}(\phi) &:= \mathbb{E}_{q_t(\mathbf{W})} \log p(\mathcal{D}|\mathbf{W}) - \mathbb{D}_{KL}(q_t(\mathbf{W})||p(\mathbf{W})) \\ &= \mathbb{E}_{q_\alpha(\xi)} \log p(\mathcal{D}|\Theta, \xi^{(t)}) - \mathbb{D}_{KL}(q_t(\mathbf{W})||p(\mathbf{W})).\end{aligned}\tag{4}$$

**Overcoming the singularity issue of approximate posterior.** In Variational Dropout with correlated parameterization, there is a mismatch in support between the approximate posterior and the prior, thus making the KL term  $\mathbb{D}_{KL}(q(\mathbf{W})||p(\mathbf{W}))$  undefined [29]. Specifically, the form  $\mathbf{W} = \text{diag}(\xi)\Theta$  is equivalent to multiplying each row  $\mathbf{W}_{i:}$  by the same scalar noise  $\xi_i$ , namely  $\mathbf{W}_{i:} = \xi_i\Theta_{i:}$  with  $q(\xi_i) = \mathcal{N}(1, \alpha_i)$ . This means that the approximate distribution always assigns all its mass on subspaces defined by the directions aligned with the rows of  $\Theta$ . These subspaces have Lebesgue measure zero causing the singularities in approximate posterior and the KL term will be undefined whenever the prior  $p(\mathbf{W})$  puts zero mass to these subspaces. However, in VSD, the scalar noises are not treated separately due to the structured correlation. In other words, VSD injects each  $\xi_i$  into the whole matrix  $\Theta$  instead of some individual directions. Indeed, we have:

$$\mathbf{W}^{(VD)} = \text{diag}(\xi^{(0)})\Theta = \Theta + \text{diag}(\eta^{(0)})\Theta = \Theta + \sum_{i=1}^K \eta_i^{(0)}(\text{diag}(\mathbf{e}_i)\Theta) = \Theta + \sum_{i=1}^K \eta_i^{(0)}\Theta_{(i)} \quad (5)$$

$$\mathbf{W}^{(VSD)} = \text{diag}(\xi^{(t)})\Theta = \Theta + \text{diag}(\mathbf{U}\eta^{(0)})\Theta = \Theta + \sum_{i=1}^K \eta_i^{(0)}(\text{diag}(\mathbf{U}_{i:})\Theta) \quad (6)$$

where  $\Theta_{(i)}$  is the matrix  $\Theta$  with only the  $i$ -th row retained. While VD causes *singular* components represented by  $\{\Theta_{(i)}\}_{i=1}^K$ , VSD maintains a trainable orthogonal matrix  $\mathbf{U}$  which prevents the approximate posterior from having degenerate supports with measure zero, thereby avoiding the singularity issue. In the next section, we will make the KL term well-defined with an appropriate prior.

### 3.2 Derivation of tractable variational lower bound

We consider employing an isotropic Gaussian as the prior distribution, namely  $p(\mathbf{W}) = \prod_{j=1}^Q p(\mathbf{W}_{:j})$  with  $p(\mathbf{W}_{:j}) = \mathcal{N}(0, \text{diag}(\beta_{:,j}^{-1}))$  and  $\beta$  is a hyper-parameter matrix of the same size with  $\mathbf{W}$ . With the analysis in previous section, our approximate posterior  $q_t(\mathbf{W})$  is absolutely continuous w.r.t the prior  $p(\mathbf{W})$ , and thus the KL term  $\mathbb{D}_{KL}(q_t(\mathbf{W})||p(\mathbf{W}))$  is defined. Furthermore, the Gaussian prior helps our proposal avoid the pathologies of improper true posterior and ill-posed inference in VD.

Note that, the prior  $p(\mathbf{W})$  is a fully factorized Gaussian which usually facilitates simple analysis and efficient computation. This factorized structure is chosen also because we have no reason for the correlation between non-identical weights at first. Moreover, several arguments indicate that a simple prior over parameter  $p(\mathbf{W})$ , when interacts with neural nets architecture  $f(\mathcal{D}; \mathbf{W})$ , induces a sophisticated prior over function  $p(f(\mathcal{D}; \mathbf{W}))$ , with desirable properties and useful inductive biases [81]. However, when we are interested in structured approximations in parameter space, the factorized prior may raise some contradictions. By relative entropy decomposition, we have:

$$\mathbb{D}_{KL}(q_t(\mathbf{W})||p(\mathbf{W})) = \sum_{j=1}^Q \mathbb{D}_{KL}(q_t(\mathbf{W}_{:j})||p(\mathbf{W}_{:j})) + \mathbf{I}(\mathbf{W}_{:1}, \mathbf{W}_{:2}, \dots, \mathbf{W}_{:Q}), \quad (7)$$

where  $\mathbf{I}(\cdot)$  is the mutual information which is completely defined in VSD. Maximizing the variational lower bound tends to encourage smaller KL term, and hence constrains the components in RHS of equation (7). Intuitively, a relatively small mutual information can break the strong correlations between the columns of  $\mathbf{W}$ . Several studies have focused on this limitation and suggested using richer priors such as matrix variate Gaussian [71, 84], doubly semi-implicit distribution [54]. Our

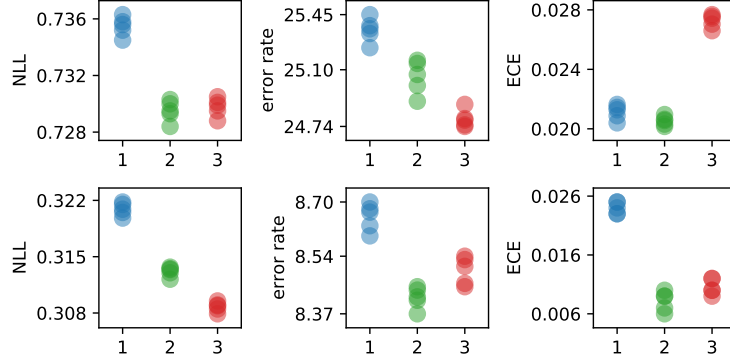


Figure 1: The performance of VSD when using the number of transformations  $T \in \{1, 2, 3\}$ . Evaluation over 5 runs on CIFAR10 (above) and SVHN (below) with LeNet architecture.

solution for this scenario is derived naturally from equation (7), in which we leverage the mutual information as an additional regularization term. Concretely, we maximize an alternative variational objective as follows:

$$\begin{aligned} \mathcal{L}_{MI}(\phi) &:= \mathbb{E}_{q_{\alpha}(\xi)} \log p(\mathcal{D}|\Theta, \xi^{(t)}) - \mathbb{D}_{KL}(q_t(\mathbf{W})||p(\mathbf{W})) + \mathbf{I}(\mathbf{W}_{:1}, \mathbf{W}_{:2}, \dots, \mathbf{W}_{:Q}) \\ &= \mathbb{E}_{q_{\alpha}(\xi)} \log p(\mathcal{D}|\Theta, \xi^{(t)}) - \mathbb{D}_{KL}(q_t^*(\mathbf{W})||p(\mathbf{W})) \end{aligned} \quad (8)$$

where  $q_t^*(\mathbf{W}) = \prod_{j=1}^Q q_t(\mathbf{W}_{:j})$  is the product of marginal column distributions. From the information-theoretic perspective, augmenting the mutual information is a standard principle for structure learning in Bayesian networks [38]. Interestingly, this technique is utilized reasonably in our method. This is because our dependence structure allows to specify explicitly the marginal distribution on each column of  $\mathbf{W}$ , leading to a tractable objective in equation (8) with the KL divergence between two multivariate Gaussian that can be calculated analytically in closed-form. We note that a similar application to other structured approximations, such as low-rank, Kronecker-factored or matrix variate Gaussian, can be non-trivial.

**The KL condition in VSD.** With the new variational objective in equation (8), to benefit VSD from the Bayesian statistics, we need to ensure  $\mathbb{D}_{KL}(q_t^*(\mathbf{W})||p(\mathbf{W}))$  satisfies the KL condition. We solve this prerequisite by determining the precision parameter  $\beta$  of the prior  $p(\mathbf{W})$  via the Empirical Bayes approach. Then, the KL term has the form independent of the weight parameter  $\Theta$  as follows:

$$\mathbb{D}_{KL}(q_t^*(\mathbf{W})||p(\mathbf{W})) = \frac{Q}{2} \sum_{i=1}^Q \log \left( \left( 1 + \sum_{j=1}^K \alpha_j U_{ij}^2 \right) / \alpha_i \right). \quad (9)$$

A formal proof of equation 9 is in Appendix B. We further improve the efficiency of Householder transformation in VSD by utilizing the idea from [75, 4] where this technique was deployed to approximate a full covariance Gaussian posterior on latent space of VAEs model [37]. In particular, they made the approximation more flexible by using fully connected layers between the Householder vectors. The low dimension of variational noise of VSD leads to a good adaptation, but with a little trade-off in computational complexity when increasing the number of transformation steps  $T$ . We show the performance of VSD when using this neural approximate technique with  $T \in \{1, 2, 3\}$  in Figure 1, where a larger  $T$  could potentially improve the results on predictive measures. However, to maintain computational efficiency, we recommend using  $T = 1$  or 2 in large-scale experiments.



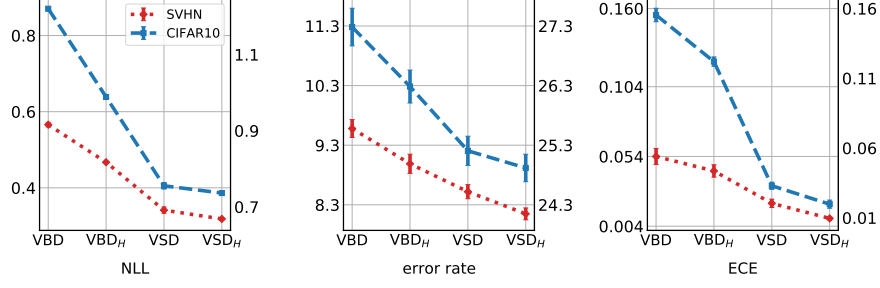


Figure 2: The performance of VBD and VSD when using the prior hierarchy, with the labels are VBD<sub>H</sub> and VSD<sub>H</sub> respectively.

### 3.3 Joint inference with hierarchical prior

We further promote our proposal by introducing a prior hierarchy in VSD framework and then obtain a joint approximation for the Dropout posterior. This will facilitate the scalability of Bayesian inference in our method in terms of the expressiveness of both prior distribution and approximate posterior. We design a two-level hierarchical prior given by:  $p(\mathbf{W}, \mathbf{z}) = p(\mathbf{W}|\mathbf{z}, \beta)p(\mathbf{z})$ , where:

$$p(\mathbf{W}|\mathbf{z}, \beta) = \prod_{j=1}^Q p(\mathbf{W}_{:j}|\mathbf{z}, \beta_{:j}) \quad \text{and} \quad p(\mathbf{W}_{:j}|\mathbf{z}, \beta_{:j}) = \mathcal{N}(0, \text{diag}(\mathbf{z} \odot \beta_{:j}^{-1})); \quad (10)$$

the hyperprior  $p(\mathbf{z})$  is a distribution with positive support such as Gamma or half-Cauchy distribution; the latent  $\mathbf{z}$  has the size of the number of rows and is shared across columns of the weight matrix  $\mathbf{W}$ ; the hyper-parameter matrix  $\beta$  is treated as a scaling factor. This prior family has a center parameterization and induces several compelling properties such as facilitating feature sparsity [42, 10], model selection [19] or improving robustness, uncertainty calibration [13].

We implement variational inference with a joint approximate posterior, also referred to as the joint Dropout posterior, which is parameterized as follows:

$$q_\phi(\mathbf{W}, \mathbf{z}) = q_\psi(\mathbf{z})q_\phi(\mathbf{W}|\mathbf{z}) \quad \text{with} \quad q_\phi(\mathbf{W}_{:j}|\mathbf{z}) = \mathcal{N}(\mathbf{z} \odot \Theta_{:j}, \text{diag}(\mathbf{z}^2 \odot \alpha \odot \Theta_{:j}^2)), \quad (11)$$

where  $q_\phi(\mathbf{W}|\mathbf{z})$  is the conditional Dropout posterior,  $q_\psi(\mathbf{z})$  is chosen depending on the family of prior  $p(\mathbf{z})$  so that the reparametrization trick can be utilized. Sampling the random weight  $\mathbf{W}$  from the joint variational posterior  $q_\phi(\mathbf{W}, \mathbf{z})$  includes two steps:  $\mathbf{z}^* \sim q_\psi(\mathbf{z})$  and  $\mathbf{W}^* \sim q_\phi(\mathbf{W}|\mathbf{z}^*)$ , in which the second one can be reparameterized as:

$$\mathbf{W}^* = \text{diag}(\mathbf{z}^*)\text{diag}(\xi)\Theta = \text{diag}(\mathbf{z}^* \odot \xi)\Theta, \quad (12)$$

with the noise  $\xi \sim \mathcal{N}(\mathbf{1}_K, \text{diag}(\alpha))$ . This new representation adapts to the vanilla Dropout procedure but allows our method to regularize each unit layer with different levels of stochasticity. We derive some insights about the role of hierarchical prior in our framework in Appendix A.2. We apply the Householder transformation to the variational noise  $\xi$  and obtain a new joint approximate posterior:

$$q_t(\mathbf{W}, \mathbf{z}) = q_\psi(\mathbf{z})q_t(\mathbf{W}|\mathbf{z}) \quad \text{with} \quad q_t(\mathbf{W}_{:j}|\mathbf{z}) = \mathcal{N}(\mathbf{z} \odot \Theta_{:j}, \mathbf{V}_j \mathbf{U} \text{diag}(\alpha) (\mathbf{V}_j \mathbf{U})^T),$$

where  $\mathbf{V}_j = \text{diag}(\mathbf{z} \odot \Theta_{:j})$ . Similarly, we define  $q_t^*(\mathbf{W}|\mathbf{z}) = \prod_{j=1}^Q q_t(\mathbf{W}_{:j}|\mathbf{z})$  and then optimize an alternative variational objective given by:

$$\begin{aligned} \mathcal{L}(\phi, \psi) := & \mathbb{E}_{q_\psi(\mathbf{z})q_t(\mathbf{W}|\mathbf{z})} \log p(\mathcal{D}|\mathbf{W}) - \\ & \mathbb{E}_{q_\psi(\mathbf{z})} (\mathbb{D}_{KL}(q_t^*(\mathbf{W}|\mathbf{z}, \phi) || p(\mathbf{W}|\mathbf{z}, \beta)) - \mathbb{D}_{KL}(q_\psi(\mathbf{z}) || p(\mathbf{z}))). \end{aligned}$$

Table 1: Computational complexity per layer of MAP and different variational methods.

| Method              | Time                                      | Memory                                |
|---------------------|---|---------------------------------------|
| MAP                 | $\mathcal{O}(KL \mathcal{B} )$            | $\mathcal{O}(L \mathcal{B} )$         |
| BBB                 | $\mathcal{O}(sKL \mathcal{B} )$           | $\mathcal{O}(sKL + L \mathcal{B} )$   |
| BBB-LTR             | $\mathcal{O}(2KL \mathcal{B} )$           | $\mathcal{O}(2L \mathcal{B} )$        |
| VMG                 | $\mathcal{O}(m^3 + 2KL \mathcal{B} )$     | $\mathcal{O}(KL \mathcal{B} )$        |
| SLANG               | $\mathcal{O}(r^2KL + rsKL \mathcal{B} )$  | $\mathcal{O}(rKL + sKL \mathcal{B} )$ |
| ELRG                | $\mathcal{O}(r^3 + (r+2)KL \mathcal{B} )$ | $\mathcal{O}((r+2)L \mathcal{B} )$    |
| <b>VSD</b>          | $\mathcal{O}(K^2 + KL \mathcal{B} )$      | $\mathcal{O}(K^2 + K \mathcal{B} )$   |
| <b>VSD-low rank</b> | $\mathcal{O}(rK + KL \mathcal{B} )$       | $\mathcal{O}(K^2 + K \mathcal{B} )$   |

Table 2: Computation time of variational methods compared to standard MAP (1x).

| Methods     | Time/epochs (s) |         |          |
|-------------|-----------------|---------|----------|
|             | LeNet5          | AlexNet | ResNet18 |
| BBB-LTR     | 1.53x           | 1.75x   | 3.28x    |
| MNF         | 2.86x           | 3.40x   | 4.88x    |
| VD          | 1.18x           | 1.15x   | 1.32x    |
| VSD $T = 1$ | 1.25x           | 1.32x   | 1.86x    |
| VSD $T = 2$ | 1.35x           | 1.49x   | 2.90x    |

A full derivation of  $\mathcal{L}(\phi, \psi)$  is given in Appendix C. About the KL condition, under the particular parametrization of variational posterior at equation (11), and a specific value of the scaling factor  $\beta$ , the KL-divergence between the structured conditional posterior  $q_t^*(\mathbf{W}_{:j}|\mathbf{z}, \phi)$  and the conditional prior  $p(\mathbf{W}_{:j}|\mathbf{z}, \beta)$  is independent of the model parameter  $\Theta$  as follows:

$$\mathbb{D}_{KL}(q_t^*(\mathbf{W}_{:j}|\mathbf{z}, \phi) || p(\mathbf{W}_{:j}|\mathbf{z}, \beta_{:j})) = \frac{1}{2} \sum_{i=1}^K \left[ z_i - \log z_i - 1 - \log \frac{1 + \sum_{j=1}^K \alpha_j U_{ij}^2}{\alpha_i} \right]. \quad (13)$$

The proof can be found in Appendix C. The parameterization of the prior hierarchy in our method is flexible without any simplifying assumptions about hyperprior  $p(\mathbf{z})$ . We can directly apply it for Variational Bayesian Dropout framework (VBD) [33] (a derivation is in Appendix F). As the experimental results are shown in Figure 2, the hierarchical prior significantly improves the performance of both VBD and VSD on predictive metrics. Therefore, we aim to introduce VSD with hierarchical prior as a unified framework of our proposal in this paper.

### 3.4 Scalability of Variational Structured Dropout

Approximating a structured posterior directly on the random weights of deep convolution models is challenging. Besides expensive computation, it is difficult to employ the local reparameterization trick [36], leading to the high variance issue in training. We apply VSD to convolutional layer by learning a structured noise with the size of the number of kernels and imposing it to convolutional weights:  $\xi \sim \mathcal{N}(\mathbf{1}_K, \mathbf{U} \text{diag}(\alpha) \mathbf{U}^T)$  and  $\mathbf{W}_{ijk} = \xi_k \Theta_{ijk}$ , with  $i, j, k$  are the indexes representing height, width, and kernel respectively. This simple solution greatly reduces computational complexity while being able to captures the dependencies among kernels of the convolutional layer.

We present the complexity of MAP and VI methods in terms of computational cost and memory storage in Table 1 with detailed analysis in Appendix D. These variational methods use Monte Carlo sampling combined with the reparameterization trick to approximate the intractable log-likelihood. VSD adopts the advantage of Dropout training and maintains an efficiency on both criteria, in which our method just samples the low dimensional noise instead of whole random weights like BBB, SLANG. This is similar to the effect of the local reparameterization trick and maintains our memory cost at a low level of  $\mathcal{O}(K^2 + K|\mathcal{B}|)$ . Without the local reparameterization trick, the methods such as vanilla BBB, SLANG also need many Monte Carlo samples for the random weights to reduce the variance in gradient estimator. It leads to loss of parallelism when backpropagating, and incurs extra costs on both time and memory. Moreover, we investigate low-rank structure in VSD by using a low



dimensional hidden unit with a ReLU activation in the neural approximation of Householder vectors. We then get a computational time of  $\mathcal{O}(rK + KL|\mathcal{B}|)$  without sacrificing much the performance (see Table 8 in Appendix D). We also give more results in Table 2 about the empirical computation time of VSD and some other methods. Based on these tables, VSD shows more effective running time than the mean-field BBNs (BBB-LTR). Although there is a trade-off when using a larger number of  $T$ , VSD does not incur much extra computation time compared to VD.

### 3.5 On explicit regularization of Variational Structured Dropout

There are compelling theories to explain the tremendous success of Dropout, of which regularization-based is one of the most active approaches [77, 24, 51, 49, 79, 8]. To characterize the regularization of VSD, we consider a deep linear neural net with  $L$  layers parameterized by  $\{\Theta^{(i)}\}_{i=1}^L$ , and define some notations as:  $\mathbf{x}$  is an input data,  $\mathcal{B}$  is the data batch;  $\mathbf{h}_i$  is the  $i$ -th hidden layer;  $\mathbf{J}_i(\mathbf{x})$  denotes the Jacobian of network output w.r.t  $\mathbf{h}_i(\mathbf{x})$ ;  $\mathbf{H}_i(\mathbf{x})$  and  $\mathbf{H}_{\text{out}}(\mathbf{x})$  denotes the Hessian of the loss w.r.t  $\mathbf{h}_i(\mathbf{x})$  and the network output, respectively. Then we have  $\mathbf{J}_i = (\prod_{l=i}^L \Theta^{(l)})^T \triangleq \Theta^{[i:L]}$  the transposition of linear multiplication of weight matrices from  $i$ -th layer to the last one. From a detailed derivation presented in Appendix E, VSD induces an explicit regularization given by:

$$R_{VSD} = \mathbb{E}_{(\mathbf{x} \sim \mathcal{B})} \sum_{i=1}^L \langle \mathbf{H}_i, \text{diag}(\mathbf{h}_i) \mathbf{U} \text{diag}(\alpha) \mathbf{U}^T \text{diag}(\mathbf{h}_i) \rangle.$$

Note that,  $\mathbf{H}_i$  can be approximated by  $\mathbf{J}_i^T \mathbf{H}_{\text{out}} \mathbf{J}_i$  after ignoring the non-PSD term which is less important empirically [69]. This regularizer offers some intriguing but popular interpretations related to the flatness of local optima or the curvature of loss landscape [79, 8] (see a detailed explanation in Appendix E). We now show novel properties of VSD induced by the orthogonal matrix  $\mathbf{U}$ .

*VSD imposes a Tikhonov-like regularization and reshapes the gradient of network weights:* We rewrite our regularization corresponding to layer  $i$ -th by:  $R_{VSD}^{(i)} = \mathbb{E}_{(\mathbf{x} \sim \mathcal{B})} \|\mathbf{H}_i^{1/2} \text{diag}(\mathbf{h}_i) \mathbf{U} \text{diag}(\alpha^{1/2})\|_F^2$ . This form can be interpreted as the Tikhonov-like regularization imposed on the square root of Hessian matrix  $\mathbf{H}_i$ , in which the Tikhonov matrix  $\Gamma := \text{diag}(\mathbf{h}_i) \mathbf{U} \text{diag}(\alpha^{1/2})$  is automatically learned during training. This principle can improve the conditioning of the estimation problem. Furthermore, when considering the case of regression problem, we have:

$$R_{VSD}^{(i)} = \mathbb{E}_{(\mathbf{x} \sim \mathcal{B})} \left[ \Theta^{[i:L]} \text{diag}(\mathbf{h}_i) \mathbf{U} \text{diag}(\alpha) \mathbf{U}^T \text{diag}(\mathbf{h}_i) \Theta^{[i:L]:T} \right]. \quad (14)$$

This is a data-dependent regularization with adaptive structure determined by the matrix  $\Gamma \Gamma^T = \text{diag}(\mathbf{h}_i) \mathbf{U} \text{diag}(\alpha) \mathbf{U}^T \text{diag}(\mathbf{h}_i)$ . From the algorithmic perspective, this regularizer allows VSD to reshape the gradient of network weights according to the geometry of the data based on both scale and direction information [12, 23]. Meanwhile  $\Gamma \Gamma^T = \text{diag}(\mathbf{h}_i) \text{diag}(\alpha) \text{diag}(\mathbf{h}_i)$  only plays as a scaling factor in VD.

*VSD penalizes implicitly the spectral norm of weight matrices:* Let  $\Omega_i := \text{diag}(\mathbf{h}_i) \mathbf{J}_i^T \mathbf{H}_{\text{out}} \mathbf{J}_i \text{diag}(\mathbf{h}_i)$ , then our regularizer can be rewritten as:  $R_{VSD}^{(i)} = \mathbb{E}_{(\mathbf{x} \sim \mathcal{B})} \sum_{k=1}^K \alpha_k^2 \mathbf{U}_{:k}^T \Omega_i \mathbf{U}_{:k}$ . Since the trainable matrix  $\mathbf{U}$  satisfies  $\mathbf{U}_{:k}^T \mathbf{U}_{:k} = 1$  for any  $k$ , a penalty on  $\mathbf{U}_{:k}^T \Omega_i \mathbf{U}_{:k}$  implies that VSD likely prefers a solution with smaller spectral norms of the matrix  $\mathbf{H}_{\text{out}}^{1/2} \mathbf{J}_i \text{diag}(\mathbf{h}_i)$  and thus of the network weights. This implication points us to the well-studied theories about generalization bound based on the spectral norm [3, 61]. Concretely, Neyshabur et al. [61] suggests that smaller spectral norm and

Table 3: Results for VSD and baselines on vectorized MNIST, CIFAR10 and SVHN. Results are averaged over 5 random seeds. For all metrics, lower is better.

| Method | MNIST    |           |       |          |           |       | CIFAR10       |           |       | SVHN  |           |       |
|--------|----------|-----------|-------|----------|-----------|-------|---------------|-----------|-------|-------|-----------|-------|
|        | FC 400x2 |           |       | FC 750x3 |           |       | CNN 32x64x128 |           |       |       |           |       |
|        | NLL      | err. rate | ECE   | NLL      | err. rate | ECE   | NLL           | err. rate | ECE   | NLL   | err. rate | ECE   |
| MAP    | 0.098    | 1.32      | 0.011 | 0.109    | 1.27      | 0.011 | 2.847         | 34.04     | 0.272 | 0.855 | 12.26     | 0.086 |
| BBB    | 0.109    | 1.59      | 0.011 | 0.140    | 1.50      | 0.013 | 1.202         | 30.11     | 0.098 | 0.545 | 10.57     | 0.017 |
| MCD    | 0.049    | 1.26      | 0.007 | 0.057    | 1.22      | 0.007 | 0.794         | 26.91     | 0.024 | 0.365 | 9.23      | 0.013 |
| VD     | 0.051    | 1.21      | 0.007 | 0.061    | 1.17      | 0.008 | 1.176         | 27.45     | 0.156 | 0.534 | 9.47      | 0.055 |
| ELRG   | 0.053    | 1.54      | -     | -        | -         | -     | 0.871         | 29.43     | -     | -     | -         | -     |
| VSD    | 0.042    | 1.08      | 0.006 | 0.048    | 1.09      | 0.006 | 0.730         | 24.92     | 0.020 | 0.299 | 8.39      | 0.008 |
| D.E    | 0.057    | 1.29      | 0.009 | 0.063    | 1.21      | 0.009 | 1.815         | 26.44     | 0.042 | 0.783 | 9.31      | 0.070 |
| SWAG   | 0.044    | 1.27      | 0.008 | 0.043    | 1.25      | 0.007 | 0.799         | 26.94     | 0.012 | 0.312 | 8.42      | 0.021 |

stable rank can lead to better generalization. This expectation can be observed empirically in VSD through Table 9 in Appendix E. A more solid investigation about the generalization of VSD is of interest.

## 4 Experiments

In this section, we provide experimental evaluations to show the effectiveness of our proposed methods compared with the existing methods in terms of both predictability and scalability. We focus mainly on variational inference methods of the following two directions: the first one is direct approximations of the posterior on the random weights of Bayesian nets, including Bayes by Backprop (BBB) [6], Variational Matrix Gaussian (VMG) [43], low-rank approximations (SLANG, ELRG) [52, 76]; and the other one is the Bayesian Dropout methods with MC Dropout (MCD) [17, 18], Variational Dropout (VD) [36], and our method-Variational Structured Dropout (VSD). In addition, we evaluate the performance of point estimate framework MAP and two standard non-variational baselines Deep Ensemble (D.E) [40] and SWAG [46]. Details about data descriptions, network architectures, hyper-parameter tuning are presented in Appendix I.

### 4.1 Image classification

We now compare the predictive performance of the aforementioned methods for classification tasks on three standard image datasets: MNIST [41], CIFAR10 [39], and SVHN [60]. We evaluate the predictive probabilities using negative log-likelihood (NLL), error rate, and expected calibration error (ECE) [57, 21]. Details on experimental settings are available in Appendix I.4.

The synthesis results of this experiment are in Table 3. On MNIST, VMG achieves err. rates of 1.17% and 1.27% with FC 400x2 and FC 750x3 respectively, while SLANG reports 1.72% err. rate with FC 400x2. In general, VSD outperforms consistently other variational methods in most settings. For D.E and SWAG, VSD exhibits competitive results on all three metrics. Especially, the figures on NLL and ECE indicate well-calibrated probabilities in our model. This is also a common but noteworthy behavior in structured approximations. On the other hand, for the remaining methods such as MAP and BBB, the error rates are worse by a large margin compared to VSD (respectively about 9% and 5% on CIFAR10, 4% and 2% on SVHN). On CIFAR10 and SVHN, these

Table 4: Image classification using AlexNet architecture. Results are averaged over 5 random seeds.

| AlexNet    | CIFAR10      |              |              | CIFAR100     |              |              | SVHN         |              |              | STL10        |              |              |
|------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
|            | NLL          | ACC          | ECE          | NLL          | ACC          | ECE          | NLL          | ACC          | ECE          | NLL          | ACC          | ECE          |
| MAP        | 1.038        | 69.58        | 0.121        | 4.705        | 40.23        | 0.393        | 0.418        | 87.56        | 0.033        | 2.532        | 65.70        | 0.267        |
| BBB        | 0.994        | 65.38        | 0.062        | 2.659        | 32.41        | <b>0.049</b> | 0.476        | 87.30        | 0.094        | 1.707        | 65.46        | 0.222        |
| MCD        | 0.717        | 75.22        | 0.023        | 2.503        | 42.91        | 0.151        | 0.401        | 88.03        | 0.023        | 1.059        | 63.65        | <b>0.052</b> |
| VD         | 0.702        | 77.28        | <b>0.028</b> | 2.582        | 43.10        | 0.106        | 0.327        | 90.76        | 0.010        | 2.130        | 65.48        | 0.195        |
| ELRG       | 0.723        | 76.87        | 0.065        | 2.368        | 42.90        | 0.099        | 0.312        | 90.66        | <b>0.006</b> | 1.088        | 59.99        | <b>0.018</b> |
| <b>VSD</b> | <b>0.656</b> | <b>78.21</b> | <b>0.046</b> | <b>2.241</b> | <b>46.85</b> | 0.112        | <b>0.290</b> | <b>91.62</b> | <b>0.008</b> | <b>1.019</b> | <b>67.98</b> | 0.079        |
| D.E        | 0.872        | 77.56        | 0.115        | 3.402        | 46.42        | 0.314        | 0.319        | 90.30        | <b>0.008</b> | 2.229        | <b>68.51</b> | 0.241        |
| SWAG       | <b>0.651</b> | <b>78.14</b> | 0.059        | <b>1.958</b> | <b>49.81</b> | <b>0.028</b> | 0.331        | 90.04        | 0.031        | 1.522        | <b>68.41</b> | 0.161        |

Table 5: Image classification using ResNet18 architecture. Results are averaged over 5 random seeds.

| ResNet18   | CIFAR10      |              |              | CIFAR100     |              |              | SVHN         |              |              | STL10        |              |              |
|------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
|            | NLL          | ACC          | ECE          | NLL          | ACC          | ECE          | NLL          | ACC          | ECE          | NLL          | ACC          | ECE          |
| MAP        | 0.644        | 86.34        | 0.093        | 2.410        | 55.38        | 0.243        | 0.232        | 95.32        | 0.028        | 1.401        | 71.26        | 0.199        |
| BBB        | 0.697        | 76.63        | 0.071        | 2.239        | 41.07        | 0.100        | 0.218        | 94.53        | 0.047        | 1.290        | 71.55        | 0.179        |
| MCD        | 0.534        | 87.47        | 0.084        | 2.121        | 59.28        | 0.227        | 0.207        | 95.78        | 0.026        | 1.333        | 72.28        | 0.188        |
| VD         | 0.451        | 87.68        | <b>0.024</b> | 2.888        | 56.80        | 0.284        | 0.164        | 96.11        | 0.017        | 1.084        | 73.29        | 0.084        |
| ELRG       | <b>0.382</b> | 87.24        | <b>0.018</b> | 1.634        | 58.14        | <b>0.096</b> | 0.145        | 96.03        | <b>0.003</b> | 0.811        | 73.66        | <b>0.080</b> |
| <b>VSD</b> | 0.464        | 87.44        | 0.061        | <b>1.504</b> | <b>60.15</b> | 0.116        | <b>0.140</b> | <b>96.41</b> | <b>0.003</b> | <b>0.769</b> | <b>74.50</b> | <b>0.083</b> |
| D.E        | 0.488        | <b>88.91</b> | 0.069        | 1.913        | <b>60.16</b> | 0.203        | 0.171        | <b>96.36</b> | 0.020        | 1.197        | 73.16        | 0.177        |
| SWAG       | <b>0.330</b> | <b>88.77</b> | 0.026        | <b>1.417</b> | <b>62.45</b> | <b>0.028</b> | <b>0.130</b> | <b>96.72</b> | 0.016        | 0.843        | 73.15        | <b>0.069</b> |

two methods and VD all show poor results on both NLL and ECE measures, implying that it will be difficult for them to reason properly about the model uncertainty especially in the out-of-distribution context. For MC Dropout, we observe a pretty good performance with the second-best result in variational methods that is similar to those reported of other works [52, 64, 76]. These results of the Bayesian Dropout methods are competitive with structured methods such as VMG, SLANG, ELRG. This further reinforces our motivation about the potential of Dropout methods for improving the predictive performance.

## 4.2 Scaling up Bayesian deep convolutional networks

We conduct additional experiments to integrate VSD into large-scale convolutional networks. We reproduce the experiments proposed in [76] (ELRG), in which we trained AlexNet and ResNet18 on 4 datasets CIFAR10, SVHN, CIFAR100 [39], and STL10 [9] to evaluate the predictive performance of our proposal compared to other methods.

The final results are given in Table 4 and Table 5, in which the top two results will be highlighted in bold. For AlexNet, the performance of VSD is more consistent and higher than other variational methods. Modern deeper architectures facilitate deterministic estimates like MAP to better learn discriminative information extracted from training data but also makes its predictions more confident when picking excessively on unique optima. Therefore, although MAP has comparable predictive accuracy on some settings, it comes at a trade-off with the worst results on both NLL and ECE. Meanwhile, ELRG with a low-rank structure on the variational posterior gains desirable properties on uncertainty metrics. Its performance on NLL and ECE are competitive to that of VSD, however,

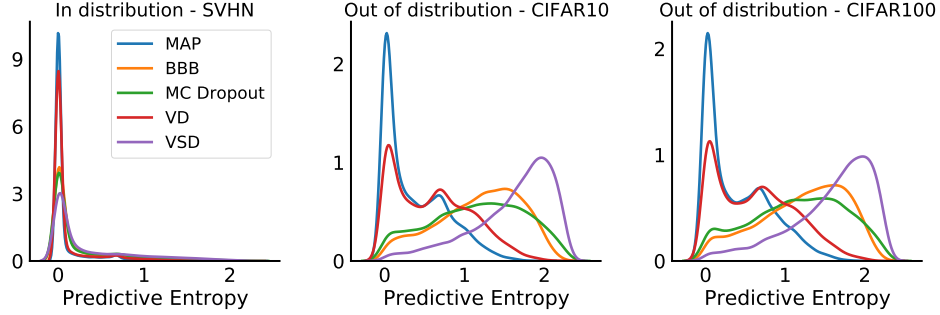


Figure 3: Histograms of predictive entropy for LeNet architecture trained on SVHN dataset.

our method obtains significant improvements on accuracy metric. For the remaining methods, BBB performs poorly on CIFAR10 and CIFAR100 in predictive accuracy, but it still has better performance than MAP in terms of NLL and ECE.

For ResNet18 architecture, while MAP and BBB still exhibit the same behavior as mentioned above, VSD continues to achieve the convincing results, namely, it has the best performance on CIFAR100 and SVHN over all three metrics when compared to variational-based methods. On CIFAR10 and STL10, VSD also gets competitive statistics on NLL and accuracy. Overall, compared with MCD, VD, and ELRG, our method maintains good performance with better stability.

### 4.3 Predictive entropy performance

We now evaluate the predictive uncertainty of each model on out-of-distribution settings that have been implemented in previous works [40, 44, 64, 76]. We evaluate the entropy of predictive distribution  $p(y^*|x^*, \mathcal{D})$  and use the density of this entropy to assess the quality of uncertainty estimates.

For LeNet, we train the model on SVHN dataset and then consider out-of-distribution data from CIFAR10 and CIFAR100. The results are shown in Figure 3. All methods work well on in-distribution data SVHN with the entropy value being distributed mostly around zero. However, the entropy densities of MAP and VD are concentrated excessively. This indicates that these methods tend to make overconfident predictions on out-of-distribution data. This claim is further supported by the quantitative results on CIFAR10 and CIFAR100 datasets. In contrast, MCD, BBB, and VSD are well-calibrated with a moderate level of confidence for in-distribution data. On CIFAR10 and CIFAR100 datasets, VSD gains better results with entropy values being distributed over a larger support, meaning that the predictions of VSD are closer to uniform on unseen classes.

We run a similar experiment, in which we train AlexNet on CIFAR10 and use SVHN, CIFAR100 as out-of-distribution data. The results are shown in the top row of Figure 4 in Appendix G.1. While MAP and VD still exhibit the same overconfident phenomenon as on LeNet, we observe the underconfident predictions of BBB and MC Dropout even on in-distribution data, which possibly leads to a high uncertainty on out-of-distribution data. We hypothesize that this is because the models trained with these methods are likely underfit with a low accuracy on the in-distribution training data. In contrast, VSD estimates reasonably the predictive entropy in both settings. The remaining scenario with ResNet18 trained on CIFAR100 is shown in the bottom row of Figure 4 with the same behaviors.

## 5 Conclusions

We proposed a novel approximate inference framework for deep Bayesian nets, named Variational Structured Dropout (VSD). In VSD, instead of performing complicated and expensive techniques directly on the random weights of deep Bayesian models, we learn a structured representation on low-dimensional space of variational noise in deterministic neural nets. VSD is successful in acquiring a flexible inference while maintaining computational efficiency and scalability, especially for deep convolutional models. Moreover, VSD can overcome theoretical critiques of the previous Variational Gaussian Dropout framework and so provide a standard Bayesian justification. The extensive experiments have evidenced the advantages of VSD such as well-calibrated prediction, better generalization, good out-of-distribution detection. Given a consistent performance of VSD as presented throughout the paper, an extension of that method to other problems, such as Bayesian active learning or reinforcement learning, is of interest.

## References

- [1] O. Arenz, M. Zhong, and G. Neumann. Trust-region variational inference with Gaussian mixture models. *Journal of Machine Learning Research*, 21(163):1–60, 2020. (Cited on page 25.)
- [2] A. Asuncion and D. Newman. UCI machine learning repository, 2007. (Cited on page 31.)
- [3] P. Bartlett, D. J. Foster, and M. Telgarsky. Spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1706.08498*, 2017. (Cited on pages 9 and 29.)
- [4] R. v. d. Berg, L. Hasenclever, J. M. Tomczak, and M. Welling. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*, 2018. (Cited on pages 6 and 25.)
- [5] C. H. Bischof and X. Sun. On orthogonal block elimination. *Preprint MCS-P450-0794, Mathematics and Computer Science Division, Argonne National Laboratory*, 1994. (Cited on page 4.)
- [6] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015. (Cited on pages 1, 10, 21, and 24.)
- [7] M. Burger and A. Neubauer. Analysis of tikhonov regularization for function approximation by neural networks. *Neural Networks*, 16(1):79–90, 2003. (Cited on page 28.)
- [8] A. Camuto, M. Willetts, U. Şimşekli, S. Roberts, and C. Holmes. Explicit regularisation in Gaussian noise injections. *arXiv preprint arXiv:2007.07368*, 2020. (Cited on page 9.)
- [9] A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011. (Cited on page 11.)
- [10] T. Cui, A. Havulinna, P. Marttinen, and S. Kaski. Informative Gaussian scale mixture priors for Bayesian neural networks. *arXiv preprint arXiv:2002.10243*, 2020. (Cited on pages 7, 21, and 23.)
- [11] E. Daxberger, E. Nalisnick, J. U. Allingham, J. Antorán, and J. M. Hernández-Lobato. Expressive yet tractable bayesian deep learning via subnetwork inference. *arXiv preprint arXiv:2010.14689*, 2020. (Cited on page 29.)
- [12] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011. (Cited on pages 9 and 28.)
- [13] M. W. Dusenberry, G. Jerfel, Y. Wen, Y.-a. Ma, J. Snoek, K. Heller, B. Lakshminarayanan, and D. Tran. Efficient and scalable Bayesian neural nets with rank-1 factors. *arXiv preprint arXiv:2005.07186*, 2020. (Cited on pages 2, 7, 21, and 23.)
- [14] B. Efron. *Large-scale inference: empirical Bayes methods for estimation, testing, and prediction*, volume 1. Cambridge University Press, 2012. (Cited on page 20.)
- [15] A. Y. Foong, D. R. Burt, Y. Li, and R. E. Turner. On the expressiveness of approximate inference in Bayesian neural networks. *arXiv preprint arXiv:1909.00719*, 2019. (Cited on pages 2, 3, and 33.)



- [16] Y. Gal. Uncertainty in deep learning. *University of Cambridge*, 1(3), 2016. (Cited on page 3.)
- [17] Y. Gal and Z. Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2015. (Cited on pages 10 and 24.)
- [18] Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016. (Cited on pages 2, 3, 10, 24, 31, and 38.)
- [19] S. Ghosh, J. Yao, and F. Doshi-Velez. Structured variational learning of Bayesian neural networks with horseshoe priors. *arXiv preprint arXiv:1806.05975*, 2018. (Cited on pages 2, 7, and 21.)
- [20] A. Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011. (Cited on page 1.)
- [21] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017. (Cited on pages 10 and 35.)
- [22] A. K. Gupta and D. K. Nagar. *Matrix variate distributions*, volume 104. CRC Press, 2018. (Cited on page 1.)
- [23] V. Gupta, T. Koren, and Y. Singer. A unified approach to adaptive regularization in online and stochastic optimization. *arXiv preprint arXiv:1706.06569*, 2017. (Cited on pages 9 and 28.)
- [24] D. P. Helmbold and P. M. Long. On the inductive bias of dropout. *The Journal of Machine Learning Research*, 16(1):3403–3454, 2015. (Cited on pages 2 and 9.)
- [25] D. P. Helmbold and P. M. Long. Surprising properties of dropout in deep networks. *The Journal of Machine Learning Research*, 18(1):7284–7311, 2017. (Cited on page 28.)
- [26] J. M. Hernández-Lobato and R. Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015. (Cited on pages 31 and 32.)
- [27] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012. (Cited on page 3.)
- [28] G. E. Hinton and D. Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993. (Cited on page 3.)
- [29] J. Hron, A. G. d. G. Matthews, and Z. Ghahramani. Variational Bayesian dropout: pitfalls and fixes. *arXiv preprint arXiv:1807.01969*, 2018. (Cited on pages 2, 3, 5, 30, 32, and 37.)
- [30] P. Izmailov, W. J. Maddox, P. Kirichenko, T. Garipov, D. Vetrov, and A. G. Wilson. Subspace inference for bayesian deep learning. In *Uncertainty in Artificial Intelligence*, pages 1169–1179. PMLR, 2020. (Cited on pages 29 and 30.)

- [31] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999. (Cited on page 3.)
- [32] M. E. Khan, D. Nielsen, V. Tangkaratt, W. Lin, Y. Gal, and A. Srivastava. Fast and scalable Bayesian deep learning by weight-perturbation in adam. *arXiv preprint arXiv:1806.04854*, 2018. (Cited on page 2.)
- [33] V. Kharitonov, D. Molchanov, and D. Vetrov. Variational dropout via empirical bayes. *arXiv preprint arXiv:1811.00596*, 2018. (Cited on pages 8, 29, and 31.)
- [34] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. (Cited on page 38.)
- [35] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016. (Cited on page 25.)
- [36] 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. (Cited on pages 1, 2, 3, 4, 8, 10, 22, 24, 29, 30, 32, and 37.)
- [37] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. (Cited on pages 3 and 6.)
- [38] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009. (Cited on page 6.)
- [39] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009. (Cited on pages 10 and 11.)
- [40] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016. (Cited on pages 10 and 12.)
- [41] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. (Cited on page 10.)
- [42] C. Louizos, K. Ullrich, and M. Welling. Bayesian compression for deep learning. *arXiv preprint arXiv:1705.08665*, 2017. (Cited on pages 7 and 21.)
- [43] C. Louizos and M. Welling. Structured and efficient variational deep learning with matrix Gaussian posteriors. In *International Conference on Machine Learning*, pages 1708–1716, 2016. (Cited on pages 1, 10, and 24.)
- [44] C. Louizos and M. Welling. Multiplicative normalizing flows for variational Bayesian neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2218–2227. JMLR. org, 2017. (Cited on pages 2, 12, 24, 25, and 35.)
- [45] D. J. MacKay. A practical Bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992. (Cited on page 1.)

- [46] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson. A simple baseline for Bayesian uncertainty in deep learning. *Advances in Neural Information Processing Systems*, 32:13153–13164, 2019. (Cited on page 10.)
- [47] S.-i. Maeda. A Bayesian encourages dropout. *arXiv preprint arXiv:1412.7003*, 2014. (Cited on page 2.)
- [48] D. McAllester. A pac-Bayesian tutorial with a dropout bound. *arXiv preprint arXiv:1307.2118*, 2013. (Cited on page 2.)
- [49] P. Mianjy and R. Arora. On dropout and nuclear norm regularization. In *International Conference on Machine Learning*, pages 4575–4584. PMLR, 2019. (Cited on page 9.)
- [50] P. Mianjy and R. Arora. On convergence and generalization of dropout training. *Advances in Neural Information Processing Systems*, 33, 2020. (Cited on page 2.)
- [51] P. Mianjy, R. Arora, and R. Vidal. On the implicit bias of dropout. In *International Conference on Machine Learning*, pages 3537–3545, 2018. (Cited on pages 2 and 9.)
- [52] A. Mishkin, F. Kunstner, D. Nielsen, M. Schmidt, and M. E. Khan. Slang: Fast structured covariance approximations for Bayesian deep learning with natural gradient. In *Advances in Neural Information Processing Systems*, pages 6245–6255, 2018. (Cited on pages 10, 11, and 24.)
- [53] D. Molchanov, A. Ashukha, and D. Vetrov. Variational dropout sparsifies deep neural networks. *arXiv preprint arXiv:1701.05369*, 2017. (Cited on pages 2, 32, and 37.)
- [54] D. Molchanov, V. Kharitonov, A. Sobolev, and D. Vetrov. Doubly semi-implicit variational inference. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2593–2602, 2019. (Cited on page 5.)
- [55] W. Mou, Y. Zhou, J. Gao, and L. Wang. Dropout training, data-dependent regularization, and generalization bounds. In *International Conference on Machine Learning*, pages 3645–3653, 2018. (Cited on pages 2 and 28.)
- [56] J. Mukhoti, P. Stenetorp, and Y. Gal. On the importance of strong baselines in Bayesian deep learning. *arXiv preprint arXiv:1811.09385*, 2018. (Cited on page 38.)
- [57] M. P. Naeini, G. Cooper, and M. Hauskrecht. Obtaining well calibrated probabilities using Bayesian binning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015. (Cited on page 10.)
- [58] E. Nalisnick, J. M. Hernández-Lobato, and P. Smyth. Dropout as a structured shrinkage prior. In *International Conference on Machine Learning*, pages 4712–4722. PMLR, 2019. (Cited on page 21.)
- [59] R. M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1995. (Cited on page 1.)
- [60] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011. (Cited on page 10.)

- [61] B. Neyshabur, S. Bhojanapalli, and N. Srebro. A pac-Bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*, 2017. (Cited on pages 9 and 29.)
- [62] C. Oh, K. Adamczewski, and M. Park. Radial and directional posteriors for Bayesian neural networks. *arXiv preprint arXiv:1902.02603*, 2019. (Cited on page 21.)
- [63] V. M.-H. Ong, D. J. Nott, and M. S. Smith. Gaussian variational approximation with a factor covariance structure. *Journal of Computational and Graphical Statistics*, 27(3):465–478, 2018. (Cited on page 2.)
- [64] K. Osawa, S. Swaroop, A. Jain, R. Eschenhagen, R. E. Turner, R. Yokota, and M. E. Khan. Practical deep learning with Bayesian principles. *arXiv preprint arXiv:1906.02506*, 2019. (Cited on pages 11 and 12.)
- [65] R. Ranganath, D. Tran, and D. Blei. Hierarchical variational models. In *International Conference on Machine Learning*, pages 324–333, 2016. (Cited on pages 2 and 25.)
- [66] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015. (Cited on page 25.)
- [67] H. Ritter, A. Botev, and D. Barber. A scalable laplace approximation for neural networks. In *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*, volume 6. International Conference on Representation Learning, 2018. (Cited on page 2.)
- [68] S. Rossi, S. Marmin, and M. Filippone. Walsh-hadamard variational inference for Bayesian deep learning. *arXiv preprint arXiv:1905.11248*, 2019. (Cited on page 2.)
- [69] L. Sagun, U. Evci, V. U. Guney, Y. Dauphin, and L. Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017. (Cited on pages 9 and 27.)
- [70] 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. (Cited on pages 2 and 3.)
- [71] S. Sun, C. Chen, and L. Carin. Learning structured weight uncertainty in Bayesian neural networks. In *Artificial Intelligence and Statistics*, pages 1283–1292, 2017. (Cited on page 5.)
- [72] X. Sun and C. Bischof. A basis-kernel representation of orthogonal matrices. *SIAM journal on matrix analysis and applications*, 16(4):1184–1196, 1995. (Cited on page 4.)
- [73] J. Swiatkowski, K. Roth, B. S. Veeling, L. Tran, J. V. Dillon, S. Mandt, J. Snoek, T. Salimans, R. Jenatton, and S. Nowozin. The k-tied normal distribution: A compact parameterization of Gaussian mean field posteriors in Bayesian neural networks. *arXiv preprint arXiv:2002.02655*, 2020. (Cited on page 2.)
- [74] M. Teye, H. Azizpour, and K. Smith. Bayesian uncertainty estimation for batch normalized deep networks. In *International Conference on Machine Learning*, pages 4907–4916. PMLR, 2018. (Cited on page 38.)

- [75] J. M. Tomczak and M. Welling. Improving variational auto-encoders using convex combination linear inverse autoregressive flow. *arXiv preprint arXiv:1706.02326*, 2017. (Cited on page 6.)
- [76] M. Tomczak, S. Swaroop, and R. Turner. Efficient low rank Gaussian variational inference for neural networks. *Advances in Neural Information Processing Systems*, 33, 2020. (Cited on pages 2, 10, 11, 12, 24, and 39.)
- [77] S. Wager, S. Wang, and P. S. Liang. Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems*, pages 351–359, 2013. (Cited on pages 2, 9, and 28.)
- [78] S. Wang and C. Manning. Fast dropout training. In *international conference on machine learning*, pages 118–126. PMLR, 2013. (Cited on page 32.)
- [79] C. Wei, S. Kakade, and T. Ma. The implicit and explicit regularization effects of dropout. In *International Conference on Machine Learning*, pages 10181–10192. PMLR, 2020. (Cited on pages 2 and 9.)
- [80] F. Wenzel, K. Roth, B. S. Veeling, J. Świątkowski, L. Tran, S. Mandt, J. Snoek, T. Salimans, R. Jenatton, and S. Nowozin. How good is the Bayes posterior in deep neural networks really? *arXiv preprint arXiv:2002.02405*, 2020. (Cited on page 23.)
- [81] A. G. Wilson and P. Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *arXiv preprint arXiv:2002.08791*, 2020. (Cited on page 5.)
- [82] M. Yin and M. Zhou. Semi-implicit variational inference. *arXiv preprint arXiv:1805.11183*, 2018. (Cited on page 25.)
- [83] G. Zhang, S. Sun, D. Duvenaud, and R. Grosse. Noisy natural gradient as variational inference. In *International Conference on Machine Learning*, pages 5852–5861. PMLR, 2018. (Cited on page 2.)
- [84] H. Zhao, Y.-H. H. Tsai, R. Salakhutdinov, and G. J. Gordon. Learning neural networks with adaptive regularization. *arXiv preprint arXiv:1907.06288*, 2019. (Cited on page 5.)
- [85] O. Zobay et al. Variational Bayesian inference with Gaussian-mixture approximations. *Electronic Journal of Statistics*, 8(1):355–389, 2014. (Cited on page 25.)

# Supplement to “Structured Dropout Variational Inference for Bayesian Neural Networks”

In this supplementary material, we collect proofs and remaining materials that were deferred from the main paper. In Appendix A, we analyze the expressiveness of VSD through the approximate posterior structure and the parameterization of prior hierarchy. In Appendix B, we provide proof for the KL condition in VSD. In Appendix C, we derive in details the variational objective of VSD with hierarchical prior. Details of computational complexity of different Bayesian methods are in Appendix D. In Appendix E, we present the explicit regularization of VSD and then provide several theoretical insights induced by our regularizer. In Appendix F, we provide a new and complementary Bayesian justification for Variational Dropout methods. The remaining Appendices are for the additional experiments with Variational Structured Dropout (VSD).

## A The expressiveness of Variational Structured Dropout

### A.1 The fully correlated structure

A essential question is how expressive the Dropout posterior in VSD is. In VSD framework, we inject a structured noise  $\xi^{(t)}$  into the deterministic weight  $\Theta$ , then obtain a random weight  $\mathbf{W}^{(t)} = \text{diag}(\xi^{(t)})\Theta$  and an induced posterior  $q_t(\mathbf{W})$ . Because each scalar noise  $\xi_i^{(t)}$  is shared across the row  $\mathbf{W}_{i:}$ , respectively, it results in a correlation on each row of  $\mathbf{W}$ . Meanwhile, the marginal distribution on each column  $\mathbf{W}_{:,j}$  is a Gaussian distribution with full covariance matrix:  $q_t(\mathbf{W}_{:,j}) = \mathcal{N}(\Theta_{:,j}, \text{diag}(\Theta_{:,j})\mathbf{U}\text{diag}(\alpha)\mathbf{U}^T\text{diag}(\Theta_{:,j}))$ . Therefore, the Dropout posterior in VSD has a fully correlated structure.

The correlation structure of VSD has a natural interpretation which bridges our approach to other structured approximations. Indeed, a full correlation over the whole random matrix can be parameterized separately into the correlations among the rows and columns of that matrix, which implicitly affects the correlations among the input and output hidden units. Interestingly, this connection can be exhibited explicitly in our method. When performing the forward pass, Dropout procedure will generally introduce the correlations between elements at pre-activation layer, namely output hidden units. In addition, the structured noise in our method can be considered as an auxiliary variable, which has the ability to captures the correlations among neurons of input hidden units, or otherwise, it will encourage neurons to borrow statistical strength from one another through variational learning. On the other hand, we know that due to the statistical noise, the correlations among hidden units appear naturally, and this therefore rationalizes the original proposal in our paper.

### A.2 The role of hierarchical prior

We derive here some detailed discussions about the role of hierarchical prior in our method.

**Gaussian scale mixture prior and mixture approximate posterior.** The well-known property of expanding a model hierarchically is that it induces new dependencies between the data, either through shrinkage or an explicitly correlated prior [14]. The hierarchical representation in our method is a *center parameterization*, and by integrating out the latent variable  $\mathbf{z}$ , we obtain a



marginal prior distribution as follows:

$$p(\mathbf{W}_{:j}|\beta_{:j}, \tau) = \int \mathcal{N}(\mathbf{W}_{:j}|0, \text{diag}(\mathbf{z} \odot \beta_{:j}^{-1}))p(\mathbf{z}|\tau)d\mathbf{z}, \quad (15)$$

where  $p(\mathbf{z}|\tau)$  is treated as the mixing distribution. The equation (15) can also be written in an equivalent *expanded* or *non-centered parameterization* as:  $\mathbf{W}_{:j} = \gamma \odot \mathbf{z}$  with  $\gamma \sim \mathcal{N}(0, \text{diag}(\beta_{:j}^{-1}))$  and  $\mathbf{z} \sim p(\mathbf{z}|\tau)$ . This prior family have been widely used in BNN literature [42, 19, 62, 13]. By approximating the above integral with Monte Carlo sampling  $\mathbf{z}^{(i)} \sim p(\mathbf{z}|\tau)$ , we can resemble an informative prior known as Gaussian scale mixtures (GSM) [6, 10] with the following term:

$$p(\mathbf{W}_{:j}|\beta_{:j}, \tau) \approx \frac{1}{M} \sum_{i=1}^M \mathcal{N}(\mathbf{W}_{:j}|0, \text{diag}(\mathbf{z}^{(i)} \odot \beta_{:j}^{-1})). \quad (16)$$

Therefore, our hierarchical framework can provide an appealing connection between multiplicative Gaussian noise with GSM priors. This interpretation is close to the recent work in [58], in which the authors show that multiplicative noise is equivalent to the structured shrinkage prior and interestingly, MC Dropout objective is a lower bound on the scale mixture model’s marginal MAP objective.

Instead of investigating extensively the expressiveness of this prior through different parameterizations, in the scope of this work, we focus on the advantages of the joint inference that can make a mixture approximation in the variational objective function:

$$\mathbb{E}_{q_{\psi}(\mathbf{z})q_t(\mathbf{W}|\mathbf{z})} \log p(\mathcal{D}|\mathbf{W}) \approx \frac{1}{M} \sum_{i=1}^M \mathbb{E}_{q_t(\mathbf{W}|\mathbf{z}^{(i)})} \log p(\mathcal{D}|\mathbf{W}).$$

This approximation is equivalent to a mixture of structured covariance Gaussian, which has a reasonable potential to recover the multimodality of the true Bayesian posterior. Moreover, this mixture is also practical in the high dimensional setting of deep Bayesian models, because it does not require the additional computational cost from multiplying the components. We also suggest that hierarchical parameterization with joint inference is a promising approach to improve the predictive performance of deep Bayesian models, especially compared with non-Bayesian methods such as Deep Ensemble, which is evidenced by a recent work in [13].

**Hierarchical prior imposes a global stochastic noise.** At each iteration of the training process, while we draw  $\mathbf{W}$  from the conditional posterior  $q_t(\mathbf{W}|\mathbf{z})$  separately for each data as the Dropout procedure, the latent  $\mathbf{z} \sim q_{\psi}(\mathbf{z})$  is shared across the entire data batch. This means for each data input  $\mathbf{x}_n$ , we introduce a joint-structured noise  $\hat{\xi}_n^{(t)}$  with the following form:

$$\begin{aligned} \mathbf{z} \sim q_{\psi}(\mathbf{z}), \quad \xi_n^{(t)} &\sim \mathcal{N}(\mathbf{1}_K, \mathbf{U} \text{diag}(\alpha) \mathbf{U}^T), \\ \hat{\xi}_n^{(t)} &= \mathbf{z} \odot \xi_n^{(t)}, \quad \mathbf{W} = \text{diag}(\hat{\xi}_n^{(t)})\Theta. \end{aligned} \quad (17)$$

The above reinterpretation demonstrates that by the Monte Carlo estimation, the joint variational inference with hierarchical prior in our method adapts to the vanilla Dropout procedure. The new representation of Dropout noise allows our method to regularize each unit layer with different levels of stochasticity. The latent  $\mathbf{z}$  under this representation can be considered as a global variational noise and by learning its variational distribution  $q_{\psi}(\mathbf{z})$ , we can capture correlation characteristics of input samples in each data batch.

## B The KL condition in Variational Structured Dropout

The alternative variational lower bound of Variational Structured Dropout (VSD) is given as follows:

$$\begin{aligned}\mathcal{L}(\alpha, \Theta, \cdot) &= \mathbb{E}_{q_\phi(\mathbf{W})} \log p(\mathcal{D}|\mathbf{W}^{(t)}) - \mathbb{D}_{KL}(q_t^*(\mathbf{W})||p(\mathbf{W})) \\ &= \mathbb{E}_{q_\alpha(\xi)} \log p(\mathcal{D}|\Theta, \xi^{(t)}) - \sum_{j=1}^Q \mathbb{D}_{KL}(q_t(\mathbf{W}_{:j})||p(\mathbf{W}_{:j})).\end{aligned}\quad (18)$$

By applying Monte Carlo sampling combined with the reparameterization trick to approximate the expected log-likelihood, we perform a procedure being equivalent to injecting a structured noise  $\xi^{(t)}$  into the model parameters  $\Theta$ . However, to make this Monte Carlo estimation follows the Dropout training procedure, we separately draw one realization  $\xi_n^{(t)}$  for each data point  $(\mathbf{x}_n, \mathbf{y}_n)$ . This can be interpreted as arising from the local reparameterization trick [36], in which the global parameter uncertainty is translated backward into local unit uncertainty at the pre-linear layer instead of the post-linear one. Finally, to ensure the KL condition, we will specify the KL term in the form independent of  $\Theta$ . Then, the above lower bound recovers the Dropout objective function.

We have the Dropout posterior and the prior determined on the  $j$ -th column of  $\mathbf{W}$ , respectively:

$$q_t(\mathbf{W}_{:j}) = \mathcal{N}(\Theta_{:j}, \text{diag}(\Theta_{:j})\mathbf{U}\text{diag}(\alpha)\mathbf{U}^T\text{diag}(\Theta_{:j})) \quad \text{and} \quad p(\mathbf{W}_{:j}|\beta) = \mathcal{N}(0, \text{diag}(\beta_{:j}^{-1})).$$

Let  $\boldsymbol{\mu}_1 = \Theta_{:j}$ ,  $\boldsymbol{\Sigma}_1 = \text{diag}(\Theta_{:j})\mathbf{U}\text{diag}(\alpha)\mathbf{U}^T\text{diag}(\Theta_{:j})$  and  $\boldsymbol{\mu}_2 = 0$ ,  $\boldsymbol{\Sigma}_2 = \text{diag}(\beta_{:j}^{-1})$ . The KL divergence can then be calculated as follows:

$$\mathbb{D}_{KL}(q_t(\mathbf{W}_{:j})||p(\mathbf{W}_{:j})) = \frac{1}{2} \left[ \log \frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|} - K + \text{Trace}(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1) + (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) \right].$$

Since  $\mathbf{U}$  is a orthogonal matrix, we have:

$$\begin{aligned}\log \frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|} &= - \sum_{i=1}^K \log \beta_i - \sum_{i=1}^K \log \alpha_i \Theta_{ij}^2, \\ \text{Trace}(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1) &= \text{Trace}(\text{diag}(\beta_{:j} \odot \Theta_{:j}^2)\mathbf{U}\text{diag}(\alpha)\mathbf{U}^T) = \sum_{i=1}^K \beta_i \Theta_{ij}^2 \left( \sum_{j=1}^K \alpha_j U_{ij}^2 \right), \\ (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) &= \Theta_{:j}^T \text{diag}(\beta_{:j}) \Theta_{:j} = \sum_{i=1}^K \beta_i \Theta_{ij}^2.\end{aligned}$$

Given the above equations, the KL term on the  $j$ -th column of  $\mathbf{W}$  can be rewritten by:

$$\mathbb{D}_{KL}(q_t(\mathbf{W}_{:j})||p(\mathbf{W}_{:j})) = \frac{1}{2} \sum_{i=1}^K \left[ -\log \beta_i - \log \alpha_i \Theta_{ij}^2 - 1 + \beta_i \Theta_{ij}^2 \left( 1 + \sum_{j=1}^K \alpha_j U_{ij}^2 \right) \right]. \quad (19)$$

We can find that the orthogonality of Householder transformations facilitates the tractable calculation for the KL term without complicated analyses. Next, we choose the prior hyper-parameter  $\beta$  via the

Empirical Bayes approach which is achieved by optimizing  $\beta$  based upon the data. More specifically, taking the partial derivative of the RHS in equation (19) with respect to  $\beta$ , we get:

$$\frac{\partial \mathbb{D}_{KL}}{\partial \beta_i} = \frac{1}{2} \left[ -\frac{1}{\beta_i} + \Theta_{ij}^2 \left( 1 + \sum_{j=1}^K \alpha_j U_{ij}^2 \right) \right]. \quad (20)$$

Letting this derivative to zero, we obtain the optimal value for  $\beta$  in the analytical form:  $\beta_i = 1 / \left( \Theta_{ij}^2 (1 + \sum_{j=1}^K \alpha_j U_{ij}^2) \right)$ . Substitute this value in the expression of the KL term, we get the form independent of the weight parameter  $\Theta$  as follows:

$$\mathbb{D}_{KL}(q_t(\mathbf{W}_{:j}) || p(\mathbf{W}_{:j})) = \frac{1}{2} \sum_{i=1}^K \log \frac{1 + \sum_{j=1}^K \alpha_j U_{ij}^2}{\alpha_i}. \quad (21)$$

As a consequence, we obtain the conclusion in the main text about the KL condition in VSD.

## C The derivation of the variational objective with hierarchical prior

In Bayesian inference, the prior distribution plays an important role in representing the capacity of Bayesian neural networks. By utilizing intentionally the informative priors, we can significantly improve the predictive performance [80, 13]. This distribution also allows incorporating external domain knowledge or specific properties such as feature sparsity, into the deep Bayesian models [10].

However, in Dropout approximate inference, the KL condition restricts the scope of prior distribution family. Our works has overcome this limitation by proposing a unified framework using variational structured Dropout combined with hierarchical prior, in which we guarantees the KL condition without any simplifying assumptions about the prior family. Concretely, with our proposed prior hierarchy, we maximize a variational lower bound from joint variational inference as follows:

$$\begin{aligned} \mathcal{L}(\alpha, \Theta, \psi, \cdot) &= \mathbb{E}_{q_\psi(\mathbf{z})q_t(\mathbf{W}|\mathbf{z})} \log p(\mathcal{D}|\mathbf{W}) \\ &\quad - \mathbb{E}_{q_\psi(\mathbf{z})} (\mathbb{D}_{KL}(q_t^*(\mathbf{W}|\mathbf{z}, \phi) || p(\mathbf{W}|\mathbf{z}, \beta)) - \mathbb{D}_{KL}(q_\psi(\mathbf{z}) || p(\mathbf{z})). \end{aligned} \quad (22)$$

For the latent variable  $\mathbf{z}$ , we choose the prior  $p(\mathbf{z}|\eta)$  and the variational distribtuion  $q(\mathbf{z})$  as (inverse) Gamma( $a, b$ ) and log-Normal( $\gamma, \delta$ ) distribution respectively. These distributions have positive support and can be reparametrized. The KL-divergence between them also has a closed-form expression, which is given by:

$$\mathbb{D}_{KL}(q_\psi(\mathbf{z}) || p(\mathbf{z})) = a \log b - \log \Gamma(a) - a\gamma - \beta \exp(-\gamma + 0.5\delta) + 0.5(\log \delta + 1 + \log(2\pi)).$$

For the KL-divergence between the conditional posterior  $q_t^*(\mathbf{W}|\mathbf{z}, \phi)$  and the conditional prior  $p(\mathbf{W}|\mathbf{z}, \beta)$ , similarly, we need a form that does not depend on the deterministic weight  $\Theta$ . Since:

$$q_t(\mathbf{W}_{:j}|\mathbf{z}) = \mathcal{N}(\mathbf{z} \odot \Theta_{:j}, \mathbf{V}_j \mathbf{U} \text{diag}(\alpha)(\mathbf{V}_j \mathbf{U})^T) \quad \text{and} \quad p(\mathbf{W}_{:j}|\mathbf{z}, \beta_{:j}) = \mathcal{N}(0, \text{diag}(\mathbf{z} \odot \beta_{:j}^{-1}))$$

where  $\mathbf{V}_j = \text{diag}(\mathbf{z} \odot \Theta_{:j})$ , same as the analysis in the previous section, we have:

$$\begin{aligned} \mathbb{D}_{KL}(q_t^*(\mathbf{W}_{:j}|\mathbf{z}, \phi) || p(\mathbf{W}_{:j}|\mathbf{z}, \beta_{:j})) &= \\ \frac{1}{2} \sum_{i=1}^K \left[ -\log z_i - 1 - \log \beta_i - \log \alpha_i \Theta_{ij}^2 + \beta_i z_i \Theta_{ij}^2 \left( 1 + \sum_{j=1}^K \alpha_j U_{ij}^2 \right) \right]. \end{aligned} \quad (23)$$

Table 6: Computational complexity per layer of MAP and different variational methods.

| Method              | Time                                      | Memory                                |
|---------------------|---|---------------------------------------|
| MAP                 | $\mathcal{O}(KL \mathcal{B} )$            | $\mathcal{O}(L \mathcal{B} )$         |
| BBB                 | $\mathcal{O}(sKL \mathcal{B} )$           | $\mathcal{O}(sKL + L \mathcal{B} )$   |
| BBB-LTR             | $\mathcal{O}(2KL \mathcal{B} )$           | $\mathcal{O}(2L \mathcal{B} )$        |
| VMG                 | $\mathcal{O}(m^3 + 2KL \mathcal{B} )$     | $\mathcal{O}(KL \mathcal{B} )$        |
| SLANG               | $\mathcal{O}(r^2KL + rsKL \mathcal{B} )$  | $\mathcal{O}(rKL + sKL \mathcal{B} )$ |
| ELRG                | $\mathcal{O}(r^3 + (r+2)KL \mathcal{B} )$ | $\mathcal{O}((r+2)L \mathcal{B} )$    |
| <b>VSD</b>          | $\mathcal{O}(K^2 + KL \mathcal{B} )$      | $\mathcal{O}(K^2 + K \mathcal{B} )$   |
| <b>VSD-low rank</b> | $\mathcal{O}(rK + KL \mathcal{B} )$       | $\mathcal{O}(K^2 + K \mathcal{B} )$   |

Table 7: Computation time of variational methods compared to standard MAP (1x).

| Methods     | Time/epochs (s) |         |          |
|-------------|-----------------|---------|----------|
|             | LeNet5          | AlexNet | ResNet18 |
| BBB-LTR     | 1.53x           | 1.75x   | 3.28x    |
| MNF         | 2.86x           | 3.40x   | 4.88x    |
| VD          | 1.18x           | 1.15x   | 1.32x    |
| VSD $T = 1$ | 1.25x           | 1.32x   | 1.86x    |
| VSD $T = 2$ | 1.35x           | 1.49x   | 2.90x    |

Because  $\beta$  is referred to as the scaling factor, we can choose it by:  $\beta_i = 1/\left(\Theta_{ij}^2(1 + \sum_{j=1}^K \alpha_j U_{ij}^2)\right)$ . The above KL then can be rewritten in the following form:

$$\mathbb{D}_{KL}(q_t^*(\mathbf{W}_{:,j}|\mathbf{z}, \phi) || p(\mathbf{W}_{:,j}|\mathbf{z}, \beta_{:,j})) = \frac{1}{2} \sum_{i=1}^K \left[ z_i - \log z_i - 1 - \log \frac{1 + \sum_{j=1}^K \alpha_j U_{ij}^2}{\alpha_i} \right]. \quad (24)$$

As a result, this form satisfies the KL condition.

## D Computational complexity and low-rank approximation

We describe here in detail the computational complexity of the different algorithms, in which the computation is considered when performing a forward pass through a single layer of the network. We also discuss the memory usage while constructing the dynamic computation graph. To ease the presentation, we briefly recall the abbreviations of methods from the main text, in particular: Bayes by Backprop (BBB) [6], Variational Matrix Gaussian (VMG) [43], Multiplicative Normalizing Flow (MNF) [44], low-rank approximations (SLANG, ELRG) [52, 76]; and the Bayesian Dropout methods including MC Dropout (MCD) [17, 18], Variational Dropout (VD) [36].

**Computational complexity.** Assume the weight matrix of the layer is of size  $K \times L$ . First, MAP estimation performs a matrix multiplication with time cost  $K \times L$  to forward each input  $\mathbf{x}_i$  of size  $K$  in data batch  $\mathcal{B}$ . MAP estimation needs to store the output of these calculations which gives a memory cost  $L|\mathcal{B}|$ .

Next, BBB with naive reparameterization trick, in practice, needs to use  $s \geq 2$  sampled weights of dimension  $K \times L$  to reduce the variance of gradient estimator. This makes the computation hard to be performed in parallel, thus incurs multiple costs of both time and memory with  $\mathcal{O}(sKL|\mathcal{B}|)$  and  $\mathcal{O}(sKL + L|\mathcal{B}|)$  respectively. On the other hand, with the local reparameterization trick that translates uncertainty about global random weights into local noise in pre-activation unit, BBB can gain an alternative unbiased estimator with low variance while maintaining low complexity via sampling only a local noise. However, it requires two forward passes to obtain means and variances of the pre-activation.

For VMF, SLANG, and ELRG, the detailed analysis can be found on the original papers, and note that SLANG is a method that fails to employ the local reparameterization trick, thereby leading to very high complexity on both time and memory.

VSD adopts the advantage of Dropout training (VD) via just sampling the low dimension noise instead of whole random weights. An additional benefit is that VSD only requires one forward pass in parallel compared with two steps of the local reparameterization trick. When using a fully connected layer (FC) size of  $K \times K$  to parameterize the Householder vector, namely:

$$\mathbf{v}_t = \mathbf{FC}(\mathbf{v}_{t-1}), \quad \mathbf{S}_t = \left( \mathbf{I} - 2 \frac{\mathbf{v}_t \mathbf{v}_t^T}{\|\mathbf{v}_t\|_2^2} \right) \mathbf{S}_{t-1} = \mathbf{H}_t \mathbf{S}_{t-1},$$

for  $t = 1, \dots, T$ , it will induce a complexity of  $\mathcal{O}(K^2)$  to our method in terms of both time and memory cost. However, we also reduce the number of parameters of this FC by adding a low dimensional hidden layer. This simple solution results in lower computational time of  $\mathcal{O}(rK + KL|\mathcal{B}|)$  without sacrificing much the performance (see Table 8 in Appendix D). In general, VSD has shown better computational efficiency than other structured approximation methods, even comparable with the mean-field BBNs.

Note that, taking the advantage of low dimensional space to enrich the quality of approximation is a popular idea. The recent advances in variational inference have offered many modern techniques to exploit this idea, such as normalizing flow [66, 35, 4], auxiliary random variable, implicit distribution [65, 82], or mixture approximation [85, 1]. Nevertheless, the novelty depends on the sophistication when applied to specific models with certain constraints. More specifically, in the context of the problem we aim for, applying these above techniques to Bayesian Dropout frameworks requires dealing with some challenges including the difficulty of parallel backpropagation, high computational complexity, and more importantly, how to ensure the KL condition. The Householder parameterization helps our approach address these challenges in both theory and applicability. Moreover, we can extend our method to other parameterizations, for example Sylvester-based flows [4], as long as the orthogonality of matrix  $\mathbf{U}$  is preserved.

**Practical runtime.** We show in Table 7 the empirical computation time of VSD and some other methods, in which BBB-LTR and MNF are *direct* approximations of BNNs, while VD and VSD represent Dropout inference frameworks. BBB-LTR and VD maintain a mean-field structure for the approximation, while MNF and VSD share the same intuition of enriching the variational approximate distribution via low dimensional space. However, there are some key considerations that distinguish VSD from MNF including: (1) MNF facilitates flexible approximation via normalizing flows (NFs) which is much more expensive compared with the orthogonal parameterizations of VSD, MNF even used two sequences of NFs to tighten the variational lower bound; (2) VSD exploits Bayesian hierarchical modeling for the Dropout inference framework and then learns a joint posterior, while MNF adopts a implicit-marginal distribution for approximate weight posterior. The settings we use to implement MNF are given in the original paper [44].

**Low-rank approximation for VSD.** We investigate a *low-rank* structure in the fully connected layer used to parameterize the Householder vectors in our method. Instead of using one layer with full size  $K \times K$ , we add a low dimensional hidden layer with ReLU activation. The size of this hidden layer is  $r \in \{2, 5, 10\}$ . This idea is quite natural because it reduces significantly the number of parameters in our method while ensuring flexible parametrization for the Householder vectors thanks to the nonlinearity in hidden activation.

We show the performance of VSD with low-rank approximation in Table 8, where we repeat the experiment on image classification in Section 4.1 of our main paper. We can see that although the

Table 8: The performance of VSD when using low-rank approximation, where  $r$  is the dimension of hidden unit,  $T = 2$  is the number of Householder transformations. For all metrics, lower is better.

|           | MNIST    |             |       | CIFAR10       |              |              | SVHN          |             |       |
|-----------|----------|-------------|-------|---------------|--------------|--------------|---------------|-------------|-------|
| $T = 2$   | FC 750x3 |             |       | CNN 32x64x128 |              |              |               |             |       |
|           | NLL      | err. rate   | ECE   | NLL           | err. rate    | ECE          | NLL           | err. rate   | ECE   |
| $r = 2$   | 0.049    | <b>1.12</b> | 0.007 | 0.7298        | 25.45        | <b>0.022</b> | <b>0.3007</b> | <b>8.36</b> | 0.008 |
| $r = 5$   | 0.046    | 1.15        | 0.006 | <b>0.7199</b> | <b>24.91</b> | 0.023        | 0.3024        | <b>8.36</b> | 0.009 |
| $r = 10$  | 0.049    | 1.15        | 0.007 | 0.7365        | 25.24        | 0.024        | 0.3048        | 8.47        | 0.009 |
| full rank | 0.045    | 1.13        | 0.006 | 0.7297        | 25.18        | 0.023        | 0.3021        | 8.41        | 0.008 |

rank  $r$  is very small, the decrease in performance is negligible (still outperforms the baselines). This natural idea even improves the results on some settings such as the ECE metric in SVHN dataset.

## E The derivation of the induced regularization of VSD

In this appendix, we present the explicit regularization induced by the structured noise in our framework. Let  $\mathcal{X} \subseteq \mathbb{R}^{d_{\text{in}}}$  and  $\mathcal{Y} \subseteq \mathbb{R}^{d_{\text{out}}}$  denote the input and output spaces, respectively. We consider a deep linear neural net  $f : \mathcal{X} \rightarrow \mathbb{R}^{d_{\text{out}}}$  with  $L$  layers parameterized by  $\{\Theta^{(i)}\}_{i=1}^L$ , and then define a corresponding surrogate loss  $\ell : \mathbb{R}^{d_{\text{out}}} \times \mathcal{Y} \rightarrow \mathbb{R}$ . The goal of learning is to find a hypothesis  $f$  that minimizes the population risk:  $L_0 := \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim \mathcal{X} \times \mathcal{Y}} \ell(f(\mathbf{x}), \mathbf{y})$ . The expected term can be written when we instead optimize the empirical risk using data batch  $\mathcal{B}$  as follows:  $\hat{L} := \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim \mathcal{B}} \ell(f(\mathbf{x}), \mathbf{y})$ . For convenience, we will ignore output  $\mathbf{y}$  in the subsequent analyses.

Note that, while the overall function is linear, the representation in factored form makes the optimization landscape non-convex and hence, challenging to analyze. For simplicity, we start by considering Dropout applied to a single layer  $i$  of the network and define some detailed notations as:  $\mathbf{h}_i$  is the  $i$ -th hidden layer;  $f_i(\cdot)$  denotes the composition of the layers after  $\mathbf{h}_i$ , that means  $f_i(\mathbf{h}_i(\mathbf{x})) = f(\mathbf{x})$ ;  $\mathbf{J}_i(\mathbf{x})$  denotes the Jacobian of network output w.r.t  $\mathbf{h}_i(\mathbf{x})$ ;  $\mathbf{H}_i(\mathbf{x})$  and  $\mathbf{H}_{\text{out}}(\mathbf{x})$  denotes the Hessian of the loss w.r.t  $\mathbf{h}_i(\mathbf{x})$  and the network output, respectively. Then we have  $\mathbf{J}_i = (\prod_{l=i}^L \Theta^{(l)})^T \triangleq \Theta^{[i:L]}$  the transposition of linear multiplication of weight matrices from  $i$ -th layer to the last one. Then, we define the loss function with multiplicative structured Dropout by:

$$\hat{L}_{\text{drop}} := \mathbb{E}_{\mathbf{x} \sim \mathcal{B}} \mathbb{E}_{\xi^{(t)}} \ell(f(\mathbf{x} \odot \xi^{(t)})),$$

where  $\xi^{(t)} = \{\xi^{(t,i)}\}_{i=1}^L$  with the noise variable  $\xi^{(t,i)}$  is specified to the  $i$ -th layer respectively (the definition of  $\xi^{(t)}$  here is a bit different from that in the main text, but it is clear from the context). Then we define an explicit regularizer induced by Dropout as follows:

$$R_{VSD} := \hat{L}_{\text{drop}} - \hat{L} = \frac{1}{L} \sum_{i=1}^L \mathbb{E}_{\mathbf{x} \sim \mathcal{B}} \left[ \mathbb{E}_{\xi^{(t,i)}} \ell(f(\mathbf{x} \odot \xi^{(t,i)})) - \ell(f(\mathbf{x})) \right].$$

Let  $\xi^{(t,i)} = 1 + \eta^{(t,i)}$ , we then rewrite the loss after applying dropout on  $\mathbf{h}_i$  by:  $\ell(f(\mathbf{x} \odot \xi^{(t,i)})) = \ell(f_i(\mathbf{h}_i(\mathbf{x}) + \delta^{(t,i)}))$  with  $\delta^{(t,i)} = \mathbf{h}_i(\mathbf{x}) \odot \eta^{(t,i)}$ . To analyze the effect of this perturbation, we apply the Taylor expansion around  $\delta^{(t,i)} = \mathbf{0}$  as follows:

$$\ell(f(\mathbf{x} \odot \xi^{(t,i)})) - \ell(f(\mathbf{x})) \approx D_{\mathbf{h}_i}(\ell \circ f_i)[\mathbf{h}_i(\mathbf{x})] \delta^{(t,i)} + \delta^{(t,i)T} D_{\mathbf{h}_i}^2(\ell \circ f_i)[\mathbf{h}_i(\mathbf{x})] \delta^{(t,i)},$$



where  $\mathcal{D}_{(\cdot)}$  denotes the derivative operator. Take the expectations on both sides of the above equation with the note that  $\delta^{(t,i)}$  is a zero-mean random variable, we obtain an approximation of the Dropout explicit regularizer in VSD corresponding to  $i$ -th layer:

$$\begin{aligned} R_{VSD}^{(i)} &:= \mathbb{E}_{\mathbf{x} \sim \mathcal{B}} \mathbb{E}_{\delta^{(t,i)}} \left[ \delta^{(t,i)T} D_{\mathbf{h}_i}^2(\ell \circ f_i)[\mathbf{h}_i(\mathbf{x})] \delta^{(t,i)} \right] \\ &= \mathbb{E}_{\mathbf{x} \sim \mathcal{B}} \left\langle D_{\mathbf{h}_i}^2(\ell \circ f_i)[\mathbf{h}_i(\mathbf{x})], \mathbb{E}_{\delta^{(t,i)}} [\delta^{(t,i)} \delta^{(t,i)T}] \right\rangle \\ &= \mathbb{E}_{\mathbf{x} \sim \mathcal{B}} \left\langle \mathbf{H}_i(\mathbf{x}), \mathbb{E}_{\delta^{(t,i)}} [\delta^{(t,i)} \delta^{(t,i)T}] \right\rangle. \end{aligned}$$

Because  $\delta^{(t,i)} = \mathbf{h}_i(\mathbf{x}) \odot \eta^{(t,i)}$  and  $\eta^{(t,i)} \sim \mathcal{N}(0, \mathbf{U} \text{diag}(\alpha) \mathbf{U}^T)$ , we have:

$$\mathbb{E}_{\delta^{(t,i)}} [\delta^{(t,i)} \delta^{(t,i)T}] = \mathbb{V}_{\delta^{(t,i)}} [\delta^{(t,i)}] = \text{diag}(\mathbf{h}_i(\mathbf{x})) \mathbf{U} \text{diag}(\alpha) \mathbf{U}^T \text{diag}(\mathbf{h}_i(\mathbf{x})).$$

Meanwhile, for the term  $\mathbf{H}_i(\mathbf{x}) = D_{\mathbf{h}_i}^2(\ell \circ f_i)[\mathbf{h}_i(\mathbf{x})]$ , we can decompose into two components as:

$$D_{\mathbf{h}_i}^2(\ell \circ f_i)[\mathbf{h}_i(\mathbf{x})] = \mathbf{J}_i^T(\mathbf{x}) \mathbf{H}_{\text{out}}(\mathbf{x}) \mathbf{J}_i(\mathbf{x}) + \sum_j (D_f \ell[f(\mathbf{x})])_j D_{\mathbf{h}_i}^2(f_i)_j [\mathbf{h}_i(\mathbf{x})], \quad (25)$$

where  $j$  indicates the  $j$ -th coordinate in the output. The first term in the RHS of equation (25) is positive-semidefinite (when  $\ell$  is the mean-square error or the cross-entropy loss), but the second term is likely to be non positive-semidefinite since it involves the Hessian of a non-convex model. However, [69] suggests ignoring this quantity because it is less important empirically. Then, our regularizer can be approximated by:

$$R_{VSD}^{(i)} = \mathbb{E}_{\mathbf{x} \sim \mathcal{B}} \left\langle \mathbf{H}_i(\mathbf{x}), \text{diag}(\mathbf{h}_i(\mathbf{x})) \mathbf{U} \text{diag}(\alpha) \mathbf{U}^T \text{diag}(\mathbf{h}_i(\mathbf{x})) \right\rangle \quad (26)$$

$$\approx \mathbb{E}_{\mathbf{x} \sim \mathcal{B}} \left\langle \mathbf{J}_i^T(\mathbf{x}) \mathbf{H}_{\text{out}}(\mathbf{x}) \mathbf{J}_i(\mathbf{x}), \text{diag}(\mathbf{h}_i(\mathbf{x})) \mathbf{U} \text{diag}(\alpha) \mathbf{U}^T \text{diag}(\mathbf{h}_i(\mathbf{x})) \right\rangle, \quad (27)$$

which fulfills the criteria for a valid regulariser.

We next will derive some variants of our regularizer and from which provide several interpretations of the algorithmic aspects. To simplify the presentation, we denote the matrix  $\Gamma := \text{diag}(\mathbf{h}_i) \mathbf{U} \text{diag}(\alpha^{1/2})$  and the matrix  $\mathbf{Q} := \Gamma \Gamma^T = \text{diag}(\mathbf{h}_i(\mathbf{x})) \mathbf{U} \text{diag}(\alpha) \mathbf{U}^T \text{diag}(\mathbf{h}_i(\mathbf{x}))$ .

**Interpretation 1:** *Dropout regularizer encourages the flatness of local minima.* In deep linear network, the second derivatives of the loss w.r.t the hidden unit and the weight parameter are tightly correlated to each other. Indeed, assume  $\mathbf{Z}$  is a weight matrix following hidden layer  $\mathbf{h}_i$  in the network architecture, namely  $f(\mathbf{x}) = f_i(\mathbf{h}_i(\mathbf{x})) = f_{i+1}(\mathbf{Z} \mathbf{h}_i(\mathbf{x}))$ , then we have:

$$D_{\mathbf{Z}}(\ell(f(\mathbf{x})))[\mathbf{Z}] = \mathbf{h}_i(\mathbf{x}) D_{\mathbf{h}_{i+1}}(\ell \circ f_i)[\mathbf{h}_{i+1}(\mathbf{x})].$$

Thus, the loss derivatives w.r.t weight parameters can be expressed in terms of those w.r.t the hidden layers. For ReLU-like activations such as ELU, Softplus, our functions are at most linear, so this above equation still holds. Therefore, when the regularization of a deep linear network is measured by Hessian matrix  $\mathbf{H}_i$ , it will tend to penalize the magnitudes of the eigenvalues of the second derivative of the loss w.r.t the weight parameters, intuitively leading to small curvature of the corresponding local loss landscape. This helps the network converges to the flat minima which contributes to better generalization. We can also analyze this property directly from equation (27), specifically we penalize the Jacobian based on the norm of  $\mathbf{H}_{\text{out}}$ . This enables the learning algorithm

to maintain a low empirical Lipschitz constant, and so facilitates the optimizer to exploit flat regions of the loss function.

**Interpretation 2:** *The structured Dropout regularizer adapts a Tikhonov-like regularization and reshapes the gradient of network weights.* From equation (26) and the relation between trace operator and inner product, we have:

$$\begin{aligned} R_{VSD}^{(i)} &\approx \mathbb{E}_{\mathbf{x} \sim \mathcal{B}} \text{Trace} \left( \text{diag}(\mathbf{h}_i(\mathbf{x})) \mathbf{U} \text{diag}(\alpha) \mathbf{U}^T \text{diag}(\mathbf{h}_i(\mathbf{x})) \mathbf{H}_i(\mathbf{x}) \right) \\ &= \mathbb{E}_{\mathbf{x} \sim \mathcal{B}} \|\mathbf{H}_i^{1/2}(\mathbf{x}) \text{diag}(\mathbf{h}_i(\mathbf{x})) \mathbf{U} \text{diag}(\alpha^{1/2})\|_F^2. \end{aligned} \quad (28)$$

This form can be interpreted as Tikhonov-like regularization imposed on the square root of Hessian matrix  $\mathbf{H}_i$ , in which the Tikhonov matrix is  $\Gamma = \text{diag}(\mathbf{h}_i) \mathbf{U} \text{diag}(\alpha^{1/2})$ . Thanks to variational learning in our method, the matrix  $\Gamma$  is automatically learned instead of being manually designed. This can bring beneficial properties for training objective by encoding a notion of the smoothness of the loss function [7]. In addition, when considering the case of regression problem, from equation (27) we have  $\mathbf{H}_{\text{out}} = 1$  and:

$$\begin{aligned} R_{VSD}^{(i)} &\approx \mathbb{E}_{\mathbf{x} \sim \mathcal{B}} \text{Trace} \left( \mathbf{Q} \mathbf{J}_i^T(\mathbf{x}) \mathbf{J}_i(\mathbf{x}) \right) \\ &= \mathbb{E}_{\mathbf{x} \sim \mathcal{B}} \text{Trace} \left( \mathbf{J}_i(\mathbf{x}) \mathbf{Q} \mathbf{J}_i^T(\mathbf{x}) \right) \\ &= \mathbb{E}_{\mathbf{x} \sim \mathcal{B}} \left( \mathbf{J}_i(\mathbf{x}) \mathbf{Q} \mathbf{J}_i^T(\mathbf{x}) \right) \\ &= \mathbb{E}_{\mathbf{x} \sim \mathcal{B}} \left( \Theta^{[i:L]} \mathbf{Q} \Theta^{[i:L].T} \right), \end{aligned} \quad (29)$$

in which the penultimate equation is because the quantity in the trace operator is scalar. This form is a data-dependent regularization with adaptive structure determined by the matrix  $\mathbf{Q}$ . With vanilla Dropout, the matrix  $\mathbf{Q} = \text{diag}(\alpha \odot \mathbf{h}_i(\mathbf{x})^2)$  plays a role as a scaling factor which allows Dropout regularizer to capture highly discriminative characteristics in each data feature. This interpretation generalizes some prior works studying Dropout for simple linear models [77, 25, 55]. Moreover, in VSD, the trainable orthogonal matrix  $\mathbf{U}$  offers a more distinctive regularization effect. Specifically, the regularizer  $R_{VSD}$  in our method makes the training algorithm capable of adapting to non-isotropic the geometric shape of data distribution. In other words, it reshapes the gradient of network weights according to the geometry of the data based on both scale and direction information. This property also connects our method with subgradient methods for online convex optimization in [12, 23], from which our regularization is closely related to adaptive proximal functions in these online frameworks.

**Interpretation 3:** *VSD penalizes implicitly the spectral norm of weight matrices, which has connection to generalization.* Let  $\Omega_i := \text{diag}(\mathbf{h}_i(\mathbf{x})) \mathbf{J}_i^T(\mathbf{x}) \mathbf{H}_{\text{out}}(\mathbf{x}) \mathbf{J}_i(\mathbf{x}) \text{diag}(\mathbf{h}_i(\mathbf{x}))$ , then also from equation (27), we have:

$$\begin{aligned} R_{VSD}^{(i)} &\approx \mathbb{E}_{\mathbf{x} \sim \mathcal{B}} \text{Trace} \left( \text{diag}(\mathbf{h}_i(\mathbf{x})) \mathbf{U} \text{diag}(\alpha) \mathbf{U}^T \text{diag}(\mathbf{h}_i(\mathbf{x})) \mathbf{J}_i^T(\mathbf{x}) \mathbf{H}_{\text{out}}(\mathbf{x}) \mathbf{J}_i(\mathbf{x}) \right) \\ &= \mathbb{E}_{\mathbf{x} \sim \mathcal{B}} \text{Trace} \left( \mathbf{H}_{\text{out}}(\mathbf{x})^{1/2} \mathbf{J}_i(\mathbf{x}) \text{diag}(\mathbf{h}_i(\mathbf{x})) \mathbf{U} \text{diag}(\alpha) \mathbf{U}^T \text{diag}(\mathbf{h}_i(\mathbf{x})) \mathbf{J}_i^T(\mathbf{x}) \mathbf{H}_{\text{out}}(\mathbf{x})^{1/2} \right) \\ &= \mathbb{E}_{\mathbf{x} \sim \mathcal{B}} \|\mathbf{H}_{\text{out}}(\mathbf{x})^{1/2} \mathbf{J}_i(\mathbf{x}) \text{diag}(\mathbf{h}_i(\mathbf{x})) \mathbf{U} \text{diag}(\alpha^{1/2})\|_F^2 \\ &= \mathbb{E}_{\mathbf{x} \sim \mathcal{B}} \sum_{k=1}^K \alpha_k \|\mathbf{H}_{\text{out}}(\mathbf{x})^{1/2} \mathbf{J}_i(\mathbf{x}) \text{diag}(\mathbf{h}_i(\mathbf{x})) \mathbf{U}_{:k}\|_2^2 = \mathbb{E}_{\mathbf{x} \sim \mathcal{B}} \sum_{k=1}^K \alpha_k \mathbf{U}_{:k}^T \Omega_i \mathbf{U}_{:k}. \end{aligned} \quad (30)$$

Table 9: Comparisons of Spectral Norms (SN) and Stable Ranks (SR) from different methods. Lower is better.

| Methods | MLP         |             | LeNet       |             |             |             | AlexNet     |             |
|---------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
|         | MNIST       |             | SVHN        |             | CIFAR10     |             | SVHN        |             |
|         | SN          | SR          | SN          | SR          | SN          | SR          | SN          | SR          |
| MAP     | <b>1.47</b> | 6.33        | 3.95        | 6.64        | 3.15        | 5.45        | 1.62        | 6.78        |
| MCD     | 1.93        | 4.36        | 3.30        | 3.99        | 3.35        | 5.19        | 1.83        | 5.42        |
| VD      | 1.88        | 5.28        | 2.94        | 6.04        | 2.71        | 5.39        | 1.83        | 5.30        |
| VSD     | 2.05        | <b>4.19</b> | <b>2.28</b> | <b>2.54</b> | <b>1.94</b> | <b>4.94</b> | <b>1.36</b> | <b>5.23</b> |

Since the trainable matrix  $\mathbf{U}$  satisfies  $\mathbf{U}_{:,k}^T \mathbf{U}_{:,k} = 1$ , the above form suggests that our regularization encourages the model to converge to solution with smaller spectral norms of the matrix  $\mathbf{H}_{\text{out}}(\mathbf{x})^{1/2} \mathbf{J}_i(\mathbf{x}) \text{diag}(\mathbf{h}_i(\mathbf{x}))$  and thus of the network weights.

Our analysis here is well-motivated by the theoretical study about generalization bound of neural nets based on the spectral norm. Concretely, Neyshabur et al. [61] and Bartlett et al. [3] showed that the generalization error is upper bounded by  $\mathcal{O}\left(\sqrt{\prod_{i=1}^L \|\Theta^{(i)}\|_2^2 \sum_{i=1}^L \text{srank}(\Theta^{(i)})}\right)$  that depends on two parameter-dependent quantities: a) the scale-dependent Lipschitz constant upper-bound  $\prod_{i=1}^L \|\Theta^{(i)}\|_2^2$  (product of spectral norms) and b) the sum of scale-independent stable ranks  $\sum_{i=1}^L \text{srank}(\Theta^{(i)})$ . Essentially, this upper bound implies that smaller spectral norm and stable rank can lead to better generalization. We empirically investigate this implication and show the results in Table 9, in which we measure both spectral norm and stable rank of the weight matrix in the last layer of MLP, LeNet, and AlexNet architecture trained on MNIST, CIFAR10 and SVHN respectively. It can be seen that our method leads to a consistent reduction in terms of both the spectral norm and stable rank when compared with weight decay (MCD) and vanilla Dropout methods (MCD, VD). This also is evidenced in [3] that weight decay does not significantly impact margins or generalization.

## F A complementary Bayesian justification for Variational Dropout

In this section, we will bring a new Bayesian perspective for Variational Dropout methods [36, 33] and then derive a variational objective to implement them in our experiments. Note that, the analysis below does not directly address the issues of original VD, thus have not yet provided any standard Bayesian justification for this method. Our new interpretation, however, is consistent with some recent research in the community.

**Variational Gaussian Dropout as Subspace inference.** Reusing the analysis in Section 3.1 in the main text, we have:

$$\mathbf{W}^{(VD)} = \text{diag}(\xi)\Theta = \Theta + \text{diag}(\eta)\Theta = \Theta + \sum_{i=1}^K \eta_i (\text{diag}(\mathbf{e}_i)\Theta) = \Theta + \sum_{i=1}^K \eta_i \Theta_{(i)}, \quad (31)$$

where  $\xi \sim \mathcal{N}(1, \text{diag}(\alpha))$ ,  $\xi = 1 + \eta$  and  $\Theta_{(i)}$  is the matrix  $\Theta$  with only the  $i$ -th row retained. It turns out this representation can be well-interpreted under the subspace inference frameworks proposed in [30, 11]. Specifically, at each iteration of the inference phase, the weight parameter can be treated as the shift matrix,  $\{\Theta_{(i)}\}_{i=1}^K$  are basic vectors of the subspace  $\mathcal{S}$  and the noise  $\eta = \{\eta_i\}_{i=1}^K$  is the

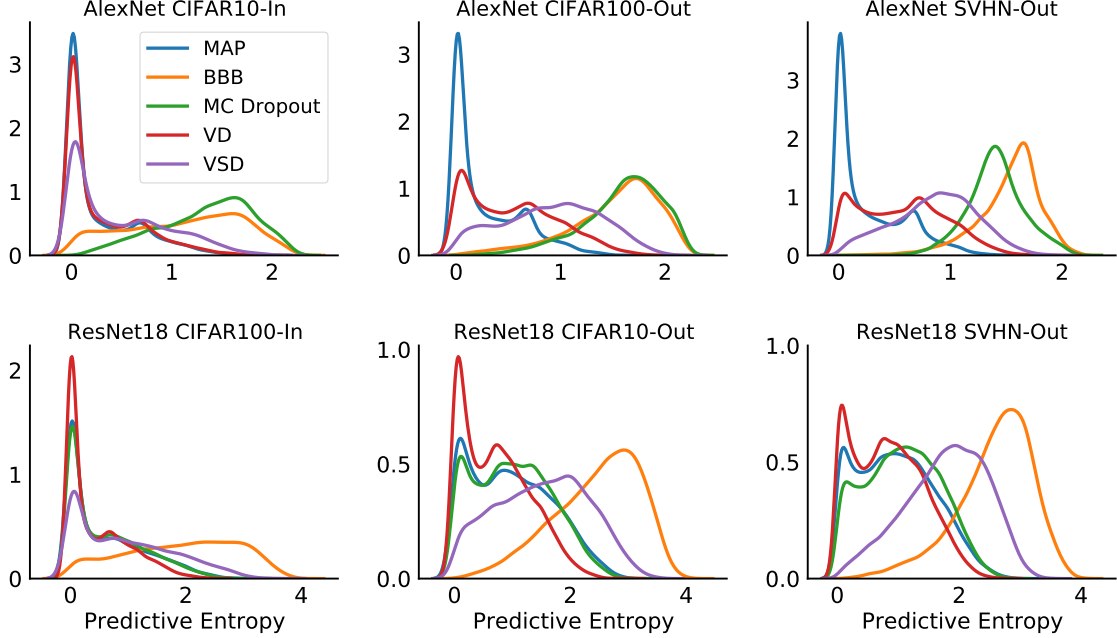


Figure 4: Histograms of predictive entropy for AlexNet (top) and ResNet18 (bottom) trained on CIFAR10 and CIFAR100 respectively.

low dimensional subspace parameter. Because  $\{\Theta_{(i)}\}_{i=1}^K$  is linearly independent, so we can consider  $\mathcal{S}$  as a projected space of the full parameter one.

We then perform variational inference over the low dimensional parameter  $\eta$ , also over  $\xi$ , in which the true posterior and the approximate posterior are defined by  $p(\xi|\mathcal{D})$  and  $q_\alpha(\xi)$  respectively. The variational objective function is given by:

$$\mathbb{E}_{q_\alpha(\xi)} \log p(\mathcal{D}|\xi, \Theta) - \mathbb{D}_{KL}(q_\alpha(\xi) \| p(\xi)). \quad (32)$$

This form suggests us a similar procedure using approximate inference on the variational noise  $\xi$ . However, our interpretation can show a distinctness in terms of model specification. Concretely, to employ variational inference on the noise *in a principle way*, we need to define a probabilistic model where the noise should play a role as a latent variable. While plausible, we are mildly cautious when introducing a new Bayesian model with an implicit treatment.

**Adjust the KL divergence term with temperature scaling.** Note that, the equation (32) clarifies the derivation of type-A version in Variational Dropout paper [36], in which the authors used  $\mathbb{D}_{KL}(q_\alpha(\xi) \| p(\xi))$  instead of the undefined term  $\mathbb{D}_{KL}(q_\phi(\mathbf{W}) \| p(\mathbf{W}))$  for implementation, but it seems just a heuristic way. On the other hand, [29] claimed that optimizing the above objective function might lead to significant overfitting due to the lack of regularization of  $\Theta$ . This issue has actually been mentioned in the subspace inference framework [30] with a similar way, from which we can propose to leverage the temperature technique to prevent the posterior from concentrating around the maximum likelihood estimate. In particular, we use the tempered posterior:

$$p_T(\xi|\mathcal{D}) \propto p(\mathcal{D}|\xi)^{1/T} p(\xi), \quad (33)$$

with the temperature hyperparameter  $T \gg 1$  chosen through cross-validation. It is also well-known

that this technique is equivalent to the KL annealing, in which we optimize a fixed objective as:

$$\mathbb{E}_{q_\alpha(\xi)} \log p(\mathcal{D}|\xi, \Theta) - T * \mathbb{D}_{KL}(q_\alpha(\xi)||p(\xi)) \quad (34)$$

We use this form for our implementation, the hyperparameter  $T$  is tuned in the wide range instead of just picking values greater than 1. In particular for ARD-Variational Dropout method proposed in [33], the prior  $p(\xi)$  is assigned by an isotropic Gaussian with the hyperparameter chosen via the Empirical Bayes, and then we obtain  $\mathbb{D}_{KL}(q_\alpha(\xi)||p(\xi)) = 0.5 \sum_{i=1}^K \log(1 + \alpha_i^{-1})$ . It can be found that this term is a degenerate case of VSD when Householder transformation is deactivated, namely the orthogonal matrix  $\mathbf{U}$  is just an identity matrix.

## G Additional empirical results

### G.1 Predictive entropy performance on ResNet18 architecture

In the bottom row of Figure 4, we show the predictive entropy of methods when training ResNet18 architecture on CIFAR100 and testing out-of-distribution on CIFAR10 and SVHN. While most methods underestimate uncertainty in out-of-distribution data, our method - VSD, calibrates the prediction with moderate confidence on in-distribution data and provides proper uncertainty on out-of-distribution settings. We also observe that BBB fails to estimate the predictive uncertainty even on in-distribution data (same behavior as on AlexNet), and with a very low accuracy, this baseline has exhibited very poor results of the mean-field BNNs compared with the Bayesian Dropout methods in terms of predictive performance.

### G.2 Regression with UCI datasets

We implement a standard experiment for Bayesian regression task on UCI dataset [2] proposed in [26]. We follow the original setup used in [18]. Detailed descriptions of the data and experimental setting can be found in Appendix I.3. We present the performance of methods based on standard metrics including root mean square error (RMSE) in Table 10 and predictive log-likelihood (LL) in Table 11. As shown in the tables, VSD performs better than baselines on most datasets in terms of both criteria (5/8 tasks on RMSE and 7/8 tasks on predictive LL). Especially, in comparison with other structured approximations on BNNs such as VMG, MNF and SLANG, our method presents much more convincing results, while MNF even shows no noticeable improvement compared to mean-field approximation (BBB). VSD also achieves better results with a significant margin compared with VD and Deep Ensemble (D.E), specifically on *Boston*, *Concrete*, *Energy*, *Yacht* datasets. This demonstrates the effectiveness of learning a structured representation for multiplicative Gaussian noise instead of using a diagonal distribution as in VD.

For predictive log-likelihood measure, although VSD is comparable to MCD and VMG on some datasets such as *Concrete*, *Energy*, *Power Plant*, our method however overall shows better results consistently above almost all settings. For instance, MCD gets poor results on *Energy*, *Kin8nm* and *Yacht*, while VMG is worse than VSD in *Boston*, *Naval* and *Yacht* by large margins. In comparison to VD and VBD, our method improves considerably the performance on *Concrete*, *Energy* and *Yacht*.

Table 10: Average test performance for UCI regression task. Results are reported with RMSE and Std. Errors.

| Dataset     | BBB             | VMG             | MNF                               | SLANG           | MCD             | VD              | D.E             | VSD                               |
|-------------|-----------------|-----------------|-----------------------------------|-----------------|-----------------|-----------------|-----------------|-----------------------------------|
| Boston      | 3.43 $\pm$ 0.20 | 2.70 $\pm$ 0.13 | 2.98 $\pm$ 0.06                   | 3.21 $\pm$ 0.19 | 2.83 $\pm$ 0.17 | 2.98 $\pm$ 0.18 | 3.28 $\pm$ 0.22 | <b>2.64 <math>\pm</math> 0.17</b> |
| Concrete    | 6.16 $\pm$ 0.13 | 4.89 $\pm$ 0.12 | 6.57 $\pm$ 0.04                   | 5.58 $\pm$ 0.19 | 4.93 $\pm$ 0.14 | 5.16 $\pm$ 0.13 | 6.03 $\pm$ 0.13 | <b>4.72 <math>\pm</math> 0.11</b> |
| Energy      | 0.97 $\pm$ 0.09 | 0.54 $\pm$ 0.02 | 2.38 $\pm$ 0.07                   | 0.64 $\pm$ 0.03 | 1.08 $\pm$ 0.03 | 0.64 $\pm$ 0.02 | 2.09 $\pm$ 0.06 | <b>0.47 <math>\pm</math> 0.01</b> |
| Kin8nm      | 0.08 $\pm$ 0.00 | 0.08 $\pm$ 0.00 | 0.09 $\pm$ 0.00                   | 0.08 $\pm$ 0.00 | 0.09 $\pm$ 0.00 | 0.08 $\pm$ 0.00 | 0.09 $\pm$ 0.00 | <b>0.08 <math>\pm</math> 0.00</b> |
| Naval       | 0.00 $\pm$ 0.00 | 0.00 $\pm$ 0.00 | 0.00 $\pm$ 0.00                   | 0.00 $\pm$ 0.00 | 0.00 $\pm$ 0.00 | 0.00 $\pm$ 0.00 | 0.00 $\pm$ 0.00 | <b>0.00 <math>\pm</math> 0.00</b> |
| Power Plant | 4.21 $\pm$ 0.03 | 4.04 $\pm$ 0.04 | 4.19 $\pm$ 0.01                   | 4.16 $\pm$ 0.04 | 4.00 $\pm$ 0.04 | 3.99 $\pm$ 0.03 | 4.11 $\pm$ 0.04 | <b>3.92 <math>\pm</math> 0.04</b> |
| Wine        | 0.64 $\pm$ 0.01 | 0.63 $\pm$ 0.01 | <b>0.61 <math>\pm</math> 0.00</b> | 0.65 $\pm$ 0.01 | 0.61 $\pm$ 0.01 | 0.62 $\pm$ 0.01 | 0.64 $\pm$ 0.00 | 0.63 $\pm$ 0.01                   |
| Yacht       | 1.13 $\pm$ 0.06 | 0.71 $\pm$ 0.05 | 2.13 $\pm$ 0.05                   | 1.08 $\pm$ 0.06 | 0.72 $\pm$ 0.05 | 1.09 $\pm$ 0.09 | 1.58 $\pm$ 0.11 | <b>0.69 <math>\pm</math> 0.06</b> |

Table 11: Average test performance for UCI regression task. Results are reported with test LL and Std. Errors.

| Dataset     | BBB              | VMG              | MNF              | SLANG            | MCD                                | VD               | D.E              | VSD                                |
|-------------|------------------|------------------|------------------|------------------|------------------------------------|------------------|------------------|------------------------------------|
| Boston      | -2.66 $\pm$ 0.06 | -2.46 $\pm$ 0.09 | -2.51 $\pm$ 0.06 | -2.58 $\pm$ 0.05 | -2.40 $\pm$ 0.04                   | -2.39 $\pm$ 0.04 | -2.41 $\pm$ 0.06 | <b>-2.35 <math>\pm</math> 0.05</b> |
| Concrete    | -3.25 $\pm$ 0.02 | -3.01 $\pm$ 0.03 | -3.35 $\pm$ 0.04 | -3.13 $\pm$ 0.03 | <b>-2.97 <math>\pm</math> 0.02</b> | -3.07 $\pm$ 0.03 | -3.06 $\pm$ 0.04 | <b>-2.97 <math>\pm</math> 0.02</b> |
| Energy      | -1.45 $\pm$ 0.10 | -1.06 $\pm$ 0.03 | -3.18 $\pm$ 0.07 | -1.12 $\pm$ 0.01 | -1.72 $\pm$ 0.01                   | -1.30 $\pm$ 0.01 | -1.38 $\pm$ 0.05 | <b>-1.06 <math>\pm</math> 0.01</b> |
| Kin8nm      | 1.07 $\pm$ 0.00  | 1.10 $\pm$ 0.01  | 1.04 $\pm$ 0.00  | 1.06 $\pm$ 0.00  | 0.97 $\pm$ 0.00                    | 1.14 $\pm$ 0.01  | 1.20 $\pm$ 0.00  | <b>1.17 <math>\pm</math> 0.01</b>  |
| Naval       | 4.61 $\pm$ 0.01  | 2.46 $\pm$ 0.00  | 3.96 $\pm$ 0.01  | 4.76 $\pm$ 0.00  | 4.76 $\pm$ 0.01                    | 4.81 $\pm$ 0.00  | 5.63 $\pm$ 0.00  | <b>4.83 <math>\pm</math> 0.01</b>  |
| Power Plant | -2.86 $\pm$ 0.01 | -2.82 $\pm$ 0.01 | -2.86 $\pm$ 0.01 | -2.84 $\pm$ 0.01 | <b>-2.79 <math>\pm</math> 0.01</b> | -2.82 $\pm$ 0.01 | -2.79 $\pm$ 0.01 | <b>-2.79 <math>\pm</math> 0.01</b> |
| Wine        | -0.97 $\pm$ 0.01 | -0.95 $\pm$ 0.01 | -0.93 $\pm$ 0.00 | -0.97 $\pm$ 0.01 | <b>-0.92 <math>\pm</math> 0.01</b> | -0.94 $\pm$ 0.01 | -0.94 $\pm$ 0.03 | -0.95 $\pm$ 0.01                   |
| Yacht       | -1.56 $\pm$ 0.02 | -1.30 $\pm$ 0.02 | -1.96 $\pm$ 0.05 | -1.88 $\pm$ 0.01 | -1.38 $\pm$ 0.01                   | -1.42 $\pm$ 0.02 | -1.18 $\pm$ 0.05 | <b>-1.14 <math>\pm</math> 0.02</b> |

### G.3 Uncertainty with toy regression

We provide an additional experiment to assess qualitatively the predictive uncertainty of methods using a synthetic regression dataset introduced in [26]. We generated 20 training inputs from  $\mathcal{U}[-4, 4]$  and assigned the corresponding target as  $y_n = x_n^3 + \epsilon_n$  where  $\epsilon_n \sim \mathcal{N}(0, 9)$ . We then fitted a neural network with a single hidden layer of 100 units. We also fixed the variance of likelihood regression to the true variance of noise  $\epsilon$ . We compare the performance of methods including BBB, MCD, VD and VSD. For the Dropout-based methods, we do not apply the dropout noise for the input layer since it is 1-dimensional. At the test time of all methods, we use 1000 MC samples to approximate the predictive distribution. The results are shown in Figure 5.

We would expect that in the area of observed data, the models should obtain the predictive means closer to the ground truth with high confidence, and at the same time, increase the predictive variance when moving away from the data. Thus we can see that our method, VSD, provides a more realistic predictive distribution than the remaining ones.

For MCD, we fixed the dropout rate  $p$  at default 0.5, because tuning manually this value does not increase the variance of noise distribution Bernoulli( $1 - p$ ), and this will lead to less variance in subsequent pre-activation unit by the Central Limit Theorem [78]. Therefore, with the shallow architecture of one hidden unit in this experiment, any tuning of the dropout rate  $p$  can result into a decrease in the predictive variance of model.

Whereas with VD, the dropout rate  $\alpha$  needs to be restricted in range  $(0, 1)$  during the training to avoid the poor local optima and to prevent the objective function from being degenerate [36, 53, 29]. As a consequence, this can reduce the ensemble diversity in the predictions of this method that leads to underestimate the uncertainty of predictive distribution.

On the contrary, the parameters in our method can be optimized without any limited assumptions. Moreover, with a structured representation for Gaussian perturbation, our method can capture rich statistical dependencies in the true posterior that facilitates fidelity posterior approximation and



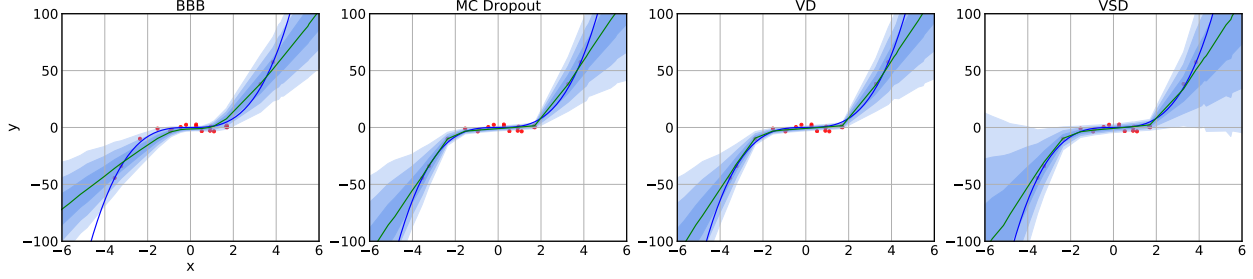


Figure 5: Predictive distributions for the toy dataset. The observations are shown as red dots. The blue line represents the true data generating function and the mean predictions are shown as the green line. Blue areas correspond to  $\pm 3$  standard deviation around the mean.

then provides proper estimates of the true model uncertainty.

In addition, we also conduct one more experiment to investigate the ability of VSD to estimate in-between uncertainty. This experiment is motivated by a recent work of Foong et al. [15], in which the authors proved that for shallow Bayesian neural nets, neither mean-field Gaussian nor Dropout posterior are capable of expressing meaningful in-between uncertainty in many situations. Due to the acquired distinctiveness in the structure of Dropout posterior distribution, we suggest that our method-VSD can theoretically overcome this limitation of MC Dropout and Variational Dropout. We validate empirically this statement by considering a regression problem with the dataset consisting of two well-separated clusters of covariates. We apply VSD to train a ReLU network with one hidden layer of size 50 and then present the predictive distributions in Figure 6. Contrary to the behavior of MC Dropout and VD reported in [15], VSD can represent a reasonable uncertainty between two data clusters. A more comprehensive analysis would be promising work for future research.

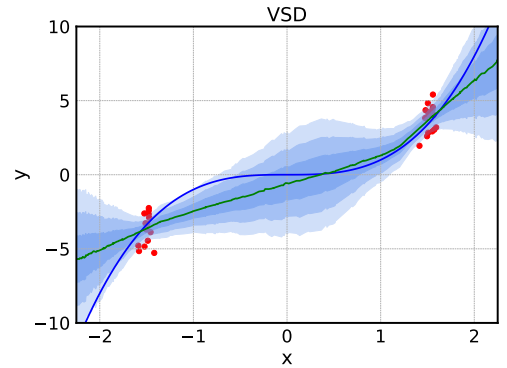


Figure 6: In-between uncertainty.

## H Further discussion of uncertainty measures

We present here several approaches used to measure or assess the quality of the predictive uncertainty of models. We also present some insights into what those metrics mean. For simplicity, we consider the multi-class classification problems on the supervised dataset  $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$ . For Bayesian methods, we can compute the predictive probabilities for each sample  $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$  that belongs to class  $c$  as:

$$\begin{aligned} \hat{p}_{ic} &:= \int p(\mathbf{y}_i = c | \mathbf{x}_i, \mathbf{W}) p(\mathbf{W} | \mathcal{D}) d\mathbf{W} \\ &\approx \int p(\mathbf{y}_i = c | \mathbf{x}_i, \mathbf{W}) q(\mathbf{W}) d\mathbf{W} \approx \frac{1}{S} \sum_{s=1}^S p(\mathbf{y}_i = c | \mathbf{x}_i, \mathbf{W}^{(s)}) \end{aligned} \quad (35)$$

with  $\{\mathbf{W}^{(s)}\}_{s=1}^S$  are variational Monte Carlo samples. The following metrics are all based on this predictive probability.

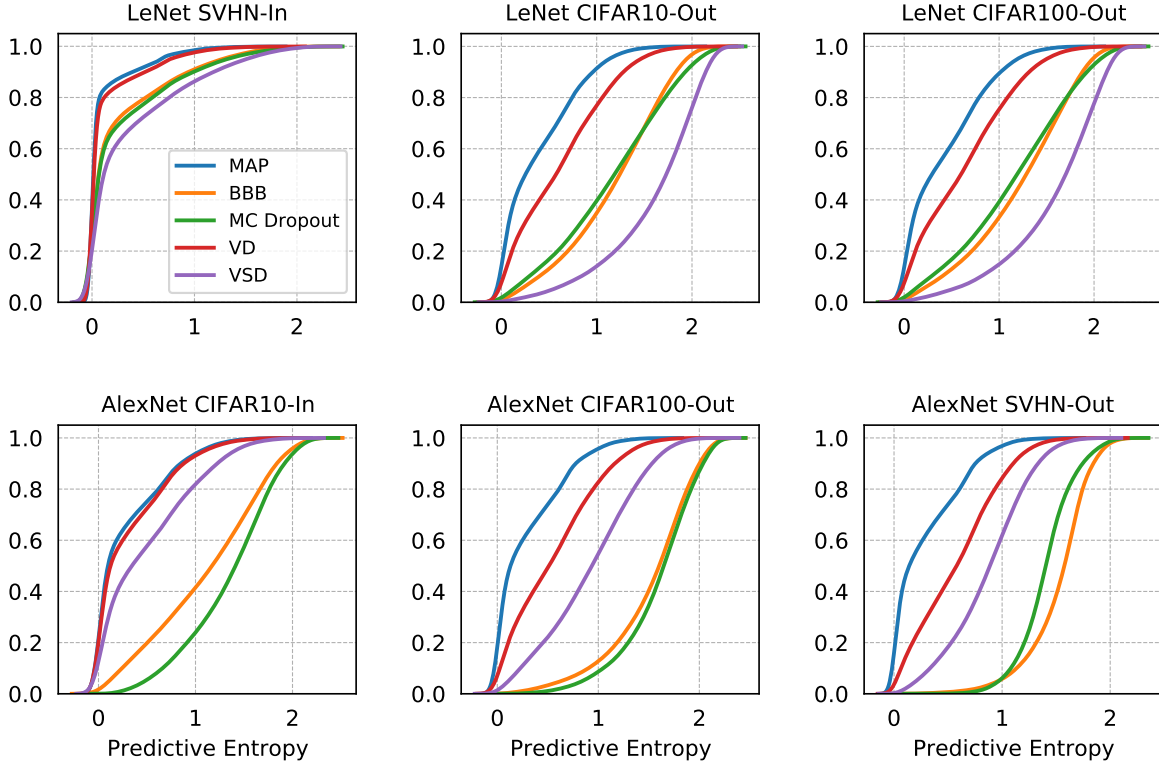


Figure 7: Empirical CDF of the predictive entropy for LeNet (top) and AlexNet (bottom) trained on SVHN and CIFAR10 respectively.

## H.1 Negative log-likelihood

The negative log-likelihood (NLL) is a standard measure of a probabilistic model’s quality and also a common uncertainty metric which is defined by:  $\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^K -y_{ic} \log \hat{p}_{ic}$ . If the predicted probabilities are overconfident, NLL will severely penalize incorrect predictions, causing this quantity to become large even if the test error rate is low. In contrast, an underconfident prediction contributes a substantial amount to the NLL regardless of whether the prediction is correct or not. Therefore, a model that achieves a good test NLL tends to make predictions with sufficiently high confidence on easy samples and hesitant predictions on hard, easy-to-fail samples.

These arguments can be evidenced by the results of experiments on modern convolutional networks (Table 4 and Table 5). Although MAP has the predictive accuracy being competitive with our method, it comes at a trade-off with the worst results on NLL. Thus the predictions of MAP are more likely to be overconfident. Corresponding experiments on out-of-distribution settings (Figure 3 bottom and Figure 4) confirmed this.

## H.2 Predictive entropy

Predictive entropy determined on a input sample  $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}$  is given by  $\frac{1}{K} \sum_{c=1}^K -\hat{p}_{ic} \log \hat{p}_{ic}$ . Underconfident models give noisy predictive predictions which result in high entropy on even in-distribution data. In contrast, overconfident models with spike predictive distributions tend to produce near zero

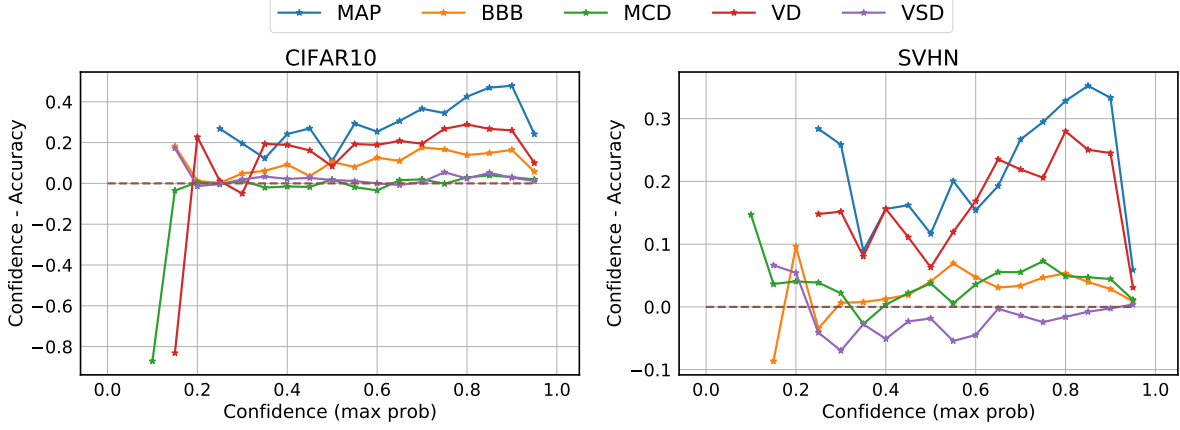


Figure 8: Reliability diagrams of LeNet-5 on CIFAR10 and SVHN dataset.

predictive entropies.

In Figures 3 and Figure 4 we plot the histogram of predictive entropies to quantify uncertainty estimation of the methods on out-of-distribution settings. In some cases, when the histograms are difficult to distinguish from each other, we instead use the empirical CDF which may be more informative [44]. We redraw the Figure 3 in the main text by empirical CDF in Figure 7. The distance between lines gives more visual views on the performance differences between the methods.

### H.3 Expected Calibration Error

Expected Calibration Error (ECE) [21] captures the discrepancy between model’s predicted probability estimates and the actual accuracy. This quantity is computed by first binning the predicted probabilities into  $M$  distinct bins and calculate the accuracy of each bin. Let  $B_m$  be the set of indices of samples whose prediction confidence falls into the interval  $I_m = (\frac{m-1}{M}, \frac{m}{M}]$ . The accuracy of  $B_m$  and the average confidence within bin  $B_m$  are defined as follows:

$$\text{acc}(B_m) := \frac{1}{|B_m|} \sum_{i \in B_m} \mathbf{1}[\hat{y}_i = y_i], \quad \text{conf}(B_m) := \frac{1}{|B_m|} \sum_{i \in B_m} q(\mathbf{x}_i),$$

where  $q(\mathbf{x}_i)$  is the confidence for sample  $i$ . In this work, we define the confidence score  $q$  as the maximum predictive probability on each data of the classifier. ECE is then computed by taking a weighted average of the bins’ accuracy/confidence difference:

$$\text{ECE} := \sum_{m=1}^M \frac{|B_m|}{n} |\text{acc}(B_m) - \text{conf}(B_m)|. \quad (36)$$

**Reliability Diagrams** [21] in Figure 8 are visual representations of model calibration. These diagrams plot accuracy as a function of confidence. Any deviation from a perfect horizontal zero-line represents miscalibration. We can observe that MAP and VD exhibit overconfident prediction (low accuracy on many bins of high confidence). Meanwhile, VSD calibrates well the predictive probabilities resulting in the lowest ECE in both settings.

Table 12: Quality of out-of-distribution detection on image classification tasks. (Top) LeNet-5 train on SVHN, evaluate on CIFAR10, CIFAR100. (Middle) AlexNet train on CIFAR10, evaluate on CIFAR100, SVHN. (Bottom) ResNet-18 train on CIFAR, evaluate on CIFAR10, CIFAR100.  $\uparrow$  (AUROC, AUPR IN, AUPR OUT) indicates larger value is better, and  $\downarrow$  (FPR, Detection error) indicates lower value is better. VSD performs the best on almost all metrics and datasets.

| LeNet-5<br>(SVHN) | CIFAR10     |             |             |             |             | CIFAR100    |             |             |             |             |
|-------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
|                   | FPR         | Det. err.   | AUROC       | AUPR IN     | AUPR OUT    | FPR         | Det. err.   | AUROC       | AUPR IN     | AUPR OUT    |
| MAP               | 0.78        | 0.23        | 0.83        | 0.93        | 0.58        | 0.76        | 0.22        | 0.84        | 0.93        | 0.60        |
| BBB               | 0.56        | 0.17        | 0.90        | 0.96        | 0.73        | 0.54        | 0.17        | 0.90        | 0.96        | 0.75        |
| MCD               | 0.50        | 0.15        | 0.92        | <b>0.97</b> | 0.78        | 0.49        | 0.15        | <b>0.92</b> | <b>0.97</b> | 0.78        |
| VD                | 0.62        | 0.17        | 0.89        | 0.96        | 0.71        | 0.64        | 0.17        | 0.89        | 0.96        | 0.71        |
| VSD               | <b>0.45</b> | <b>0.14</b> | <b>0.93</b> | <b>0.97</b> | <b>0.81</b> | <b>0.47</b> | <b>0.14</b> | <b>0.92</b> | <b>0.97</b> | <b>0.79</b> |

| AlexNet<br>(CIFAR10) | CIFAR100    |             |             |             |             | SVHN        |             |             |             |             |
|----------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
|                      | FPR         | Det. err.   | AUROC       | AUPR IN     | AUPR OUT    | FPR         | Det. err.   | AUROC       | AUPR IN     | AUPR OUT    |
| MAP                  | 0.88        | 0.35        | 0.70        | 0.73        | 0.65        | <b>0.89</b> | 0.33        | 0.71        | 0.59        | <b>0.83</b> |
| BBB                  | 0.93        | 0.46        | 0.55        | 0.54        | 0.54        | 0.99        | 0.45        | 0.53        | 0.33        | 0.70        |
| MCD                  | 0.91        | 0.41        | 0.63        | 0.63        | 0.60        | 0.97        | 0.39        | 0.59        | 0.47        | 0.74        |
| VD                   | 0.87        | 0.35        | 0.69        | 0.72        | 0.64        | 0.89        | 0.32        | 0.72        | 0.60        | <b>0.83</b> |
| VSD                  | <b>0.85</b> | <b>0.33</b> | <b>0.72</b> | <b>0.76</b> | <b>0.68</b> | 0.91        | <b>0.30</b> | <b>0.73</b> | <b>0.65</b> | <b>0.83</b> |

| ResNet-18<br>(CIFAR100) | CIFAR10     |             |             |             |             | SVHN        |             |             |             |             |
|-------------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
|                         | FPR         | Det. err.   | AUROC       | AUPR IN     | AUPR OUT    | FPR         | Det. err.   | AUROC       | AUPR IN     | AUPR OUT    |
| MAP                     | 0.89        | <b>0.37</b> | 0.67        | 0.70        | 0.63        | 0.91        | 0.36        | 0.68        | 0.56        | 0.81        |
| BBB                     | 0.93        | 0.41        | 0.62        | 0.66        | 0.58        | 0.89        | 0.37        | 0.68        | 0.51        | 0.82        |
| MCD                     | 0.89        | <b>0.37</b> | 0.68        | 0.71        | 0.63        | 0.89        | 0.34        | 0.71        | 0.58        | 0.83        |
| VD                      | 0.90        | 0.38        | 0.66        | 0.70        | 0.62        | 0.87        | 0.34        | 0.70        | 0.58        | 0.83        |
| VSD                     | <b>0.87</b> | <b>0.37</b> | <b>0.69</b> | <b>0.72</b> | <b>0.65</b> | <b>0.83</b> | <b>0.31</b> | <b>0.76</b> | <b>0.65</b> | <b>0.86</b> |

## H.4 Out-of-distribution detection metrics

We measure some metrics to evaluate the model’s ability of distinguishing in distribution and out-of-distribution images.

An ideal classifier should have a low probability of false alarm, corresponding to a low False Positive Rate (FPR), while maintaining a high sensitivity, corresponding to a high True Positive Rate (TPR). **FPR at 95% TPR** is a suitable metric to measure the reality of this desire.

Receiver Operating Characteristic (ROC) curve plots the TPR against the FPR at all possible decision thresholds. The larger the Area Under the ROC curve (**AUROC**), the better the classifier’s ability to separate the negative and positive samples. More intuitively, AUROC indicates the chance that the classifier produces a higher score on a random positive sample than a random negative one.

Besides the ROC curve, which represents the trade-off between the sensitivity and the probability of false alarm, Precision-Recall (PR) curves are often plotted to represent a trade-off between making accurate positive predictions and covering a majority of all positive results. We report the Area Under the Precision-Recall curve (AUPR) in two scenarios: (i) in-distribution samples are used as the positive samples (**AUPR-In**), (ii) out-of-distribution samples are used as the positive samples (**AUPR-Out**).

Finally, we report the **detection error**, a measure of the minimum expected probability that

the model incorrectly detects whether data samples come from in or out of training data distribution. This quantity is defined as  $\min_{\delta}\{0.5P_{in}(q(\mathbf{x}) \leq \delta) + 0.5P_{out}(q(\mathbf{x}) > \delta)\}$  where  $\delta$  is the decision threshold and  $q(\mathbf{x})$  is the maximum value of softmax probability.

## I Details for experimental settings

### I.1 Training techniques

**Initialization and Learning rate scheduling.** It is well known that good initialization and proper learning rate can improve both the speed and quality of convergence in the training process. Concretely, we use `init.xavier_uniform` in PyTorch to initialize the weight parameters of all methods. In each experiment, we use 5 random seeds for different initializations and then report average results.

For positive-valued parameters such as dropout rate  $\alpha$  or Gaussian variance  $\sigma^2$ , we optimize on the logarithmic form to avoid numerical issues and bad local optima. We use Adam optimizer with initial learning rate  $lr \in \{0.001, 0.002\}$  on all of our experiments, and then we also apply `MultiStepLR` scheduler with multiplicative factor `gamma=0.3` to adjust the learning rate after every 10 epochs.

**KL annealing.** This technique re-weights the expected log-likelihood and regularized term by a scaling factor  $\lambda$  as follows:

$$\mathbb{E}_{q_{\phi}} \mathbf{W} \log p(\mathcal{D}|\mathbf{W}) - \lambda \mathbb{D}_{KL}(q_{\phi}(\mathbf{W})||p(\mathbf{W})). \quad (37)$$

The KL annealing in many contexts is remarkably effective to the problems using variational Bayesian inference. It seems to prevent underfitting, over-parameterization, or to mitigate KL vanishing. We employ this technique for all methods in this paper and then tuning the weighting hyper-parameter  $\lambda$  depending on the data setting (of course with  $\lambda = 1$ , we have no modification).

### I.2 Hyperparameter tuning for all methods

We present here in detail the hyper-parameter setups of each method used in our experiments. These methods include MAP, Bayes by Backprop (BBB), MC Dropout (MCD), Variational Dropout (VD, VBD), and our method-Variational Structured Dropout (VSD); for the remaining methods such as Variational Matrix Gaussian (VMG), low-rank approximations (SLANG, and ELRG), we inherited the results reported in the original paper with the same experimental settings.

For **BBB** without the local reparameterization trick, we use two Monte Carlo samples for all of our experiments. This method needs to tune the hyper-parameters in the scale mixture Gaussian prior  $p(\mathbf{W}) = \pi\mathcal{N}(0, \sigma_1^2) + (1 - \pi)\mathcal{N}(0, \sigma_2^2)$  with the search as follows:  $-\log \sigma_1 \in \{0, 1, 2\}$ ,  $-\log \sigma_2 \in \{6, 7, 8\}$  and mixture ratio  $\pi \in \{0.25, 0.5, 0.75\}$ .

For **MC Dropout** in image classification task, we tune the droprate  $p \in \{0.2, 0.3, 0.4, 0.5, 0.6\}$  in Bernoulli distribution and the length-scale  $l^2 \in \{0.0001, 0.001, 0.01, 0.1, 1, 10, 100\}$  in the isotropic Gaussian prior  $p(\mathbf{W}) = \mathcal{N}(0, l^{-2}\mathbf{I}_K)$ .

For **VD** and **VSD**, we find a good initialization for the Gaussian dropout rate  $\alpha = p/(1 - p)$ . However, this droprate  $\alpha$  in VD methods needs to be restricted in range  $(0, 1)$  during the training to prevent the objective function from being degenerate which can lead to the poor local optima as mentioned in some related papers [36, 53, 29].

We also employ the KL annealing mentioned in the previous section [I.1](#) for BBB, VD and VSD methods. This technique has actually been exploited in the original papers of BBB and VD to improve the predictive performance.

For **SWAG**, we train the models using SGD with momentum  $\gamma = 0.9$ . For the learning rate we adopt the same decaying schedule as in the paper: a constant learning rate of  $5 \times 10^{-2}$ , up to 50% of the total number of epochs, and then a linear decay down to a learning rate of  $5 \times 10^{-4}$  until 90% of the total number of epochs, where once again the learning rate is kept constant until the end of training. We use rank  $K = 20$  for estimation of Gaussian covariance matrix. At test time we use 30 weight samples for Bayesian model averaging. We also tune the weight decay in the range of  $\{1e-3, 5e-3, 1e-4, 5e-4\}$ .

### I.3 UCI regression settings

Following the original settings, we used a Bayesian neural network with one hidden layer of 50 units and ReLU activation functions. We also used the 20 splits of the data provided by [\[18\]](#)<sup>1</sup> for training and testing. The models are trained to convergence using Adam optimizer [\[34\]](#) with the learning rate  $lr = 0.001$ , the batchsize  $M = 128$  and 2000 epochs for all datasets. To make an ensemble prediction at the test time, we used 10000 Monte Carlo samples for the Bayesian Dropout methods as the suggestion in [\[18\]](#).

For VMG, SLANG and Deep Ensemble, we inherited the results reported in the original paper. The results of MNF is report in [\[74\]](#). For a more fair comparison, we used the results of MC Dropout reported in [\[56\]](#) with the version using 4000 epochs for convergence training and tuning hyper-parameters by Bayesian Optimization.

In this experiment, we used the number of Householder transformations  $T = 2$ , and need to tune the precision  $\tau$  of Gaussian likelihood  $p(\mathbf{y}|\mathbf{W}, \mathbf{x}, \tau) = \mathcal{N}(\mathbf{y}|f(\mathbf{W}, \mathbf{x}), \tau)$ . Similar to MC Dropout and SLANG, we used 40 iterations of Bayesian Optimization (BO) to tune this precision. For each iteration of BO, 5-fold cross-validation is used to evaluate the considered hyperparameter setting. This is repeated for each of the 20 train-test splits for each dataset. The final values of each dataset are reported with the mean and standard error from these 20 runs.

### I.4 Image classification settings

With the MNIST dataset, 60,000 training points were split into a training set of 50,000 and a validation set of 10,000. We then vectorized the images and trained using two fully connected Bayesian neural networks with the size of hidden layers of 400x2 and 750x3 respectively.

For the remaining two datasets CIFAR10 and SVHN, we used the same simple convolutional neural network consisting of two convolutional layers with 32 and 64 kernels, followed by a fully connected network with one hidden layer of size 128.

We trained the models with the default Adam optimizer using learning rate 0.001, batchsize 100, and the number of epochs 100. At the test time, we used 10 Monte Carlo samples for all methods. We also used the number of Householder transformations  $T \in \{1, 2, 3\}$  for our method on all three datasets. With Bayes by Backprop (BBB), we did not employ the local reparameterization trick, instead we used two MC samples during the training follow the code published by the authors. We utilized the available results of the baselines in the same setting, including NLL and error rate of ELRG on MNIST and CIFAR10 dataset, error rate of VMG and SLANG on MNIST dataset.

---

<sup>1</sup>The splits are publicly available from <https://github.com/yaringal/DropoutUncertaintyExps>

## I.5 Scaling up modern CNNs settings

We follow the experimental setup for Bayesian deep convolutional networks proposed in [76]. We trained all algorithms for 300 epochs using a batch size of 200 and the ADAM optimizer with learning rate 0.001. We normalize datasets using empirical mean and standard deviation and then employ data augmentation for these experiments: random padding followed by flipping left/right (except SVHN). In the testing phase, we used 100 variational Monte Carlo samples for both AlexNet and ResNet18 architecture.

We report average results over 5 random initializations. We also refer to the code for MC Dropout and Bayes by Backprop on AlexNet and ResNet18 that is available at <https://github.com/team-approx-bayes/dl-with-bayes>.