
Measuring Uncertainty through Bayesian Learning of Deep Neural Network Structures

Zhijie Deng, Yucen Luo, Jun Zhu

Department of Computer Science

Tsinghua University

{dzj17, luoyc15}@mails.tsinghua.edu.cn, dcszj@mail.tsinghua.edu.cn

Abstract

Bayesian neural networks (BNNs) introduce uncertainty estimation to deep networks by performing Bayesian inference on weights. However, such models face the challenge of Bayesian inference in a high-dimensional and usually over-parameterized space. This paper investigates a new line of Bayesian deep learning by performing Bayesian inference on the structure of deep networks in a compact manner. Instead of building structures from scratch inefficiently, we draw inspirations from the neural architecture search to define the network structure as masks on the redundant operations between computational nodes, and develop an efficient stochastic variational inference method to infer the distribution of the network structure. We further develop several practical improvements to facilitate the weight convergence in DBSN. Empirically, our method exhibits competitive predictive performance (slightly better than DenseNet in the same model size) across a range of classification and segmentation tasks. More importantly, our method preserves benefits of Bayesian principles, producing improved uncertainty estimation than elaborated, competing baselines, including maximum a posteriori, Monte Carlo dropout, and variational BNN algorithms.

1 Introduction

One core goal of Bayesian deep learning is to equip the expressive deep neural networks (DNNs) with appropriate uncertainty quantification. Bayesian neural networks (BNNs) measure uncertainty by performing principled Bayesian inference over network weights. The uncertainty can facilitate exploration in model-based reinforcement learning [9], active and continual learning [20, 47], and is crucial in risk-sensitive applications like healthcare [34] and autonomous driving [26].

Modeling weight uncertainty is well-evaluated [6, 14], with its limitation also exposed — as widely witnessed [42, 55], BNNs usually preserve benefits of Bayesian principles such as well-calibrated predictions at the expense of compromising performance, rendering them impractical in real-world applications. The root is the fact that the network weights are usually very high-dimensional, incurring intrinsic challenges in modeling and inference. It is challenging for us to specify a sensible weight prior, or perform inference with flexible variational posteriors, due to the high-dimensional nature of weights [51, 44, 37, 49, 60]. Recently, the efficient particle-based variational methods [36] have been developed with promise, but they still suffer from the particle collapsing and degrading issues for BNNs due to the above fact [61, 54]. A most recent work [55] attempts to understand why the Bayes posterior on weights is systematically worse than the point estimates, providing valuable insights but remaining the main question unsolved.

As the prevalence of Neural Architecture Search (NAS) [62, 35], we have realised the promise of learning distributions on network structures: (i) fitting the network structure *w.r.t.* the data benefits the predictive performance of the learned models; (ii) the structures of DNNs are commonly much more concise and compact than the weights, alleviating many of the main criticisms of modeling weight uncertainty; (iii) in the posterior predictive (also known as *Bayes ensemble*), stochasticity on

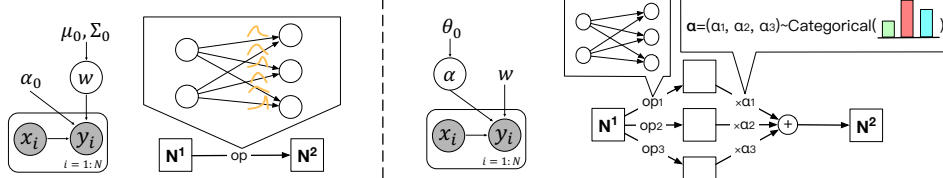


Figure 1: BNNs with uncertainty on the network weights (left) vs. DBSN with uncertainty on the network structure (right) (we only depict three operations between tensors N^1 and N^2 for simplicity). w and α represent network weights and network structure, respectively. In DBSN, w is also learnable.

structure tends to introduce higher diversity into the ensemble components, making the ensemble based prediction more accurate and robust. In fact, Bayesian learning on structures in probabilistic networks has a long history [56, 5, 1]. However, these methods were developed under a fully probabilistic framework, making it impractical to extend to DNNs. Furthermore, these methods inefficiently built structures from scratch, *e.g.*, learning the connections between units in adjacent layers, so there still exists the aforementioned inference challenge in the high-dimensional space.

Motivated by the potential supremacy of learning structures, we propose *Deep Bayesian Structure Networks* (DBSN), to measure uncertainty through Bayesian learning of the network structure. Following differentiable NAS [35, 58], we develop a general representation of network structure. Specifically, we build a super network with redundant operations (*i.e.*, there are multiple computational branches between hidden feature maps), and define the network structure as masks over them (see Fig. 1). The mask is discrete and in the same amount as the network operations, namely, with low dimension, allowing us to compactly characterize structure uncertainty.

To perform posterior inference over network structures with such representation *w.r.t.* data, we develop a unified framework integrating the learning of both the structure and weights. To bypass the aforementioned challenges of explicitly maintaining a distribution over weights, we approximately perform maximum a posteriori (MAP) estimation on the weights, which is theoretically sound and differentiates us from classic BNNs. Technically, we adopt a stochastic variational inference paradigm to achieve joint training of the whole model, suggested by the success of variational methods [6, 29].

We further develop several improvements of DBSN to facilitate the convergence of the weights such that they can provide good performance under various structure samples. Similar as the “cold posterior” trick [55], we propose to sharpen the variational posterior of the structure via two refinements. We also fix an issue of batch normalization [23] to make the model more expressive.

We empirically evaluate DBSN via predictive performance and uncertainty estimates. The specific network design for structure learning in DBSN makes it not directly comparable to most existing BNNs, so we implement a wide spectrum of baselines under the same settings of DBSN (some of them also adopt learnable structures; some explicitly model weight uncertainty), including MAP, Monte Carlo (MC) dropout [13], Bayes by Backprop (BBB) [6], noisy EK-FAC [3], and Variational Online Gauss-Newton [28]. Fair comparisons validate the effectiveness of DBSN.

2 Deep Bayesian Structure Networks (DBSN)

In this section, we describe the motivation of building a network with structure uncertainty. Then, we briefly introduce how to represent the network structure to enable compact structure learning. At last, we elaborate DBSN in terms of its theoretical framework and learning principles.

2.1 Motivation of Structure Uncertainty

To motivate our modeling of structure uncertainty, recall the challenges that are frequently posed on characterizing weight uncertainty due to the high-dimensional and over-parameterized nature of the network weights. On one side, it is hard to specify an effective prior or employ a flexible variational for the high-dimensional weights [51, 44], and hence BNNs usually exhibit unsatisfactory performance. On the other side, diverse weight samples from the weight posterior are likely to predict similarly [12], resulting in meaningless weight uncertainty, due to the over parameterization of weights. These challenges raise the necessity to develop an alternative to weight uncertainty for quantifying data uncertainty.

Given a neural network model, apart from weights, the network structure will also affect the predictive behaviour of the model, even more prominently. Besides, as shown in neural architecture search (NAS) [62, 45, 35, 8], the network structure can be defined in a compact manner, *e.g.*, a set of

scalars harnessing the information flow among hidden feature maps, while keeping its ability to introduce specific inductive bias on directing model behaviour. Therefore, opportunities arise where we can learn the posterior over the compact, low-dimensional network structure for modeling uncertainty. Moreover, with Bayesian structure learning equipped, the model perhaps gains extra benefits including boosted predictive performance due to adapting structure *w.r.t.* data, and diversified posterior ensemble thanks to the global and high-level nature of structure.

To these ends, we attempt to develop a Bayesian structure learning approach for deep models, named *Deep Bayesian Structure Networks* (DBSN), to conjoin the benefits from Bayesian principles and the prediction performance. We elaborate DBSN in the following parts.

2.2 The Structure Representation in DBSN

Before delving into the details of DBSN, we answer a core question, *i.e.*, how to define a compact but expressive representation of the network structure, by drawing inspiration from NAS.

Instead of building network structures from scratch [1] inefficiently, we advocate representing the structure in a more concise and high-level way, to control the network behaviour globally and inherit the well-designed cells in NAS [63, 45, 35]. We construct a network by stacking a sequence of computation cells which have the same internal structure and are separated by downsampling modules. Every cell contains B sequential nodes (*i.e.*, feature maps): N^1, \dots, N^B . Each node N^j is connected to all of its predecessors N^i (*i.e.*, $i < j$) by K redundant operations $o_1^{(i,j)}, \dots, o_K^{(i,j)}$. We inherit the operation set of DARTS [35], composed of convolution, skip connection, pooling, etc. We define the basic component of the structure as the discrete mask on the K available operations from N^i to N^j , *i.e.*, $\alpha^{(i,j)} \in \{0, 1\}^K$, $\sum_k \alpha_k^{(i,j)} = 1$. The basic components totally compose the whole structure $\alpha = \{\alpha^{(i,j)} | 1 \leq i < j \leq B\}$. Given such a design, we can further formulate the information flow inside the model as (w denotes the network weights):

$$N^{(i,j)} = \sum_{k=1}^K \alpha_k^{(i,j)} \cdot o_k^{(i,j)}(N^i; w), \quad N^j = \sum_{i < j} N^{(i,j)}. \quad (1)$$

This compact and widely evaluated representation of network structure provides us opportunities to effectively perform deep Bayesian structure learning.

2.3 The Theoretical Framework of DBSN

Based on the compact representation of network structure, we develop a unified learning framework of the network structure and weights.

Formally, let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ be a set of N data points, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathcal{Y}$. By viewing the network with weights w and structure α as a probabilistic model with a likelihood $p(\mathcal{D}|\alpha, w)$ and a factorized prior $p(\alpha, w) = p(\alpha)p(w)$, we aim at inferring the posterior distribution $p(\alpha, w|\mathcal{D})$. Directly deriving the posterior is intractable because it is impossible to integrate *w.r.t.* α and w for deep models. As suggested by variational BNNs [21, 16, 6], we can alternatively deploy a variational $q(\alpha, w) \in \mathcal{Q}$ to approximate $p(\alpha, w|\mathcal{D})$. The posterior inference is accomplished by minimizing the Kullback-Leibler (KL) divergence between them:

$$\min_{q \in \mathcal{Q}} D_{\text{KL}}(q(\alpha, w) \| p(\alpha, w|\mathcal{D})) = -\mathbb{E}_{q(\alpha, w)}[\log p(\mathcal{D}|\alpha, w)] + D_{\text{KL}}(q(\alpha, w) \| p(\alpha, w)) + C, \quad (2)$$

where $C = \log p(\mathcal{D})$ is a constant *w.r.t.* q and usually omitted in the minimization.

To avoid the aforementioned barrier of learning distribution over w , we untie the entanglement between α and w in the variational, then let w follow a Dirac distribution and α follow a θ -parameterized distribution¹. Namely, $q(\alpha, w) = q(\alpha|\theta)\mathbb{I}(w = w_0)$. With a common assumption that the prior of w is a standard Gaussian, the KL divergence between the variational and prior of w can be approximately estimated as $\gamma\|w_0\|_2^2$ (γ denotes a coefficient), which is also known as the L2 regularization of neural networks. Such a practical refinement indeed helps to convert the challenging problem of inferring posterior of w into a more simplified one of performing maximum a posteriori (MAP) estimation on w , proved by the empirical comparisons between these two modeling choices in Sec. 5.2 and Sec. 5.3 (the FBN baseline corresponds to DBSN with weight uncertainty).

¹Here we impose the independency between α and w in the variational for simplicity. Modeling their relationship is beneficial and practically feasible, but very difficult or computationally costly owing to the prohibitive complexity inside such relationship. We perform a trial of this in Sec 5.2.

Given these, we can re-write Eq. (2) into a practical loss for training θ and w_0 :

$$\mathcal{L}(\theta, w_0) = -\mathbb{E}_{q(\alpha|\theta)}[\log p(\mathcal{D}|\alpha, w_0)] + D_{\text{KL}}(q(\alpha|\theta)||p(\alpha)) + \gamma\|w_0\|_2^2. \quad (3)$$

In practice, γ is always regarded as a tunable hyper-parameter. Given the composable and discrete nature of the structure, we factorize $q(\alpha|\theta)$ to be a multiplication of categorical distributions, namely, $q(\alpha|\theta) = \prod_{i < j} q(\alpha^{(i,j)}|\theta^{(i,j)})$, where $\theta = \{\theta^{(i,j)} \in \mathbb{R}^K | 1 \leq i < j \leq B\}$ denote the trainable categorical logits. If no further information is provided, we assume an independent prior: $p(\alpha) = \prod_{i < j} p(\alpha^{(i,j)})$ where $p(\alpha^{(i,j)})$ are categorical distributions (or concrete ones after applying continuous relaxation) with uniform class probabilities. The graphical model of DBSN is illustrated in Fig. 1. We denote w_0 as w in the following if there is no misleading.

To minimize the objective Eq. (3) w.r.t. θ and w by gradient descent, we relax both $p(\alpha^{(i,j)})$ and $q(\alpha^{(i,j)}|\theta^{(i,j)})$ to be the concrete distributions [39], to enable the usage of low-variance reparameterization trick [29, 6]. By convention, we draw samples from $q(\alpha|\theta)$ via the following softmax-based transformation:

$$\alpha = g(\theta, \epsilon) = \{\text{softmax}((\theta^{(i,j)} + \epsilon^{(i,j)})/\tau)\}, \quad (4)$$

where $\epsilon = \{\epsilon^{(i,j)} \in \mathbb{R}^K | \epsilon_k^{(i,j)} \sim \text{Gumbel i.i.d.}\}$ are a set of Gumbel variables, $\tau \in \mathbb{R}_+$ denotes the temperature. Then, based on the reparameterization trick, we derive the following low-variance gradient estimators for updating θ and w :

$$\nabla_{\theta} \mathcal{L}(\theta, w) = \mathbb{E}_{\epsilon}[-\nabla_{\theta} \log p(\mathcal{D}|g(\theta, \epsilon), w) + \nabla_{\theta} \log q(g(\theta, \epsilon)|\theta) - \nabla_{\theta} \log p(g(\theta, \epsilon))], \quad (5)$$

$$\nabla_w \mathcal{L}(\theta, w) = \mathbb{E}_{\epsilon}[-\nabla_w \log p(\mathcal{D}|g(\theta, \epsilon), w)] + 2\gamma w. \quad (6)$$

The first term in Eq. (5) corresponds to the gradient of the negative log likelihood and we leave how to estimate the last two terms (*i.e.* log densities) in the next section. In practice, we approximate the expectation in Eq. (5) and (6) with T MC samples, and update α and w simultaneously.

After training, let θ^* and w^* denote the converged parameters, then we estimate the predictive distribution of unseen data by:

$$p(y|x_{\text{new}}, w^*) = \mathbb{E}_{q(\alpha|\theta^*)}[p(y|x_{\text{new}}, \alpha, w^*)]. \quad (7)$$

This is also known as *Bayes ensemble*. For a tractable estimation, we use MC samples to approximate the expectation. Note that DBSN assembles the predictions from networks whose structures are randomly sampled and weights are shared, which is in sharp difference from classic BNNs.

3 Practical Improvements of DBSN

As discussed, DBSN bypasses the challenges of modeling the relationship between the high-dimensional weights and the structure by explicitly factorizing them in the variational. Thus, the weights are structure-agnostic and shared among diverse structure samples during training and inference. One advantage of it is that any structure sample can be used to perform inference, based on Eq. (7). But it also raises a training challenge that the stochasticity inside network structure poses too strong regularization effects on the weights, hindering the network from achieving satisfactory convergence. In light of adopting ‘‘cold posterior’’ to practically improve the predictive performance of BNNs [55], we decide to sharpen the variational posterior of the structure, yet without underestimating the KL term in evidence lower bound [60, 3, 42]. We leverage the discrete and combinatorial nature of the structure, and propose two more controllable refinements to achieve the goal.

Scheduled variance reduction. We propose to directly harness the sample variance of the structure posterior by augmenting its sampling procedure with a tunable factor. Specifically, for concrete distribution, we propose to multiply a scalar $\beta^{(i,j)}$ with $\epsilon^{(i,j)}$ in the sampling:

$$\alpha^{(i,j)} = g(\theta^{(i,j)}, \beta^{(i,j)}, \epsilon^{(i,j)}) = \text{softmax}((\theta^{(i,j)} + \beta^{(i,j)} \epsilon^{(i,j)})/\tau). \quad (8)$$

It corresponds to a distribution with the following log probability density (detailed in Appendix A):

$$\begin{aligned} \log p(\alpha^{(i,j)}|\theta^{(i,j)}, \beta^{(i,j)}) &= \log((K-1)!) + (K-1) \log \tau - (K-1) \log \beta^{(i,j)} - \sum_{k=1}^K \log \alpha_k^{(i,j)} \\ &\quad + \sum_{k=1}^K \left[\frac{\theta_k^{(i,j)} - \tau \log \alpha_k^{(i,j)}}{\beta^{(i,j)}} \right] - K \cdot \text{L}\Sigma\text{E} \left[\frac{\theta_k^{(i,j)} - \tau \log \alpha_k^{(i,j)}}{\beta^{(i,j)}} \right], \end{aligned} \quad (9)$$

where LΣE is log-sum-exp. With this, the last two terms of Eq. (5) can be estimated exactly.

As shown in Fig. 2, sliding $\beta^{(i,j)}$ from 1 to 0 gradually decreases the diversity of the sampled structures, leading to a colder and colder posterior. When $\beta^{(i,j)} = 0$, the estimation on the structure degenerates to MAP, making us lose the benefits of Bayesian principles. Thus, we decide to sharpen the posterior with a pre-defined schedule to control the posterior dynamics more reliably. For example, in practice, we gradually decay $\beta^{(i,j)}$ from 1 to 0.5 along with the training process.

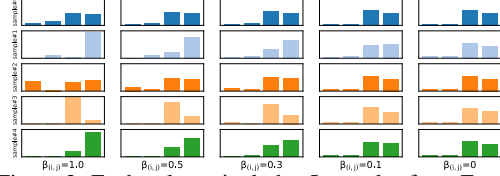


Figure 2: Each column includes 5 samples from Eq. (8) with the same $\beta^{(i,j)}$. Samples in every row share the same $\epsilon^{(i,j)}$. The base class probabilities of each sample are $\text{softmax}(\theta^{(i,j)}) = [0.05, 0.05, 0.5, 0.4]$, a common pattern in the learned structure distribution.

Support reduction. Given the combinatorial nature of the structure, the support of the structure posterior is exponentially large, perhaps leading to undesirable over-regularization on weights. Furthermore, as can be seen from Eq. (1), only the weights of the operations activated by current structure sample are updated in one training step, thus we may require more (up to $K \times$) training steps to drive the model to converge. Therefore, we propose to reduce the support of the structure posterior, by removing inessential operations from the operation set of DARTS [35], including all the 5×5 convolutions that can be replaced by stacked 3×3 convolutions and all the pooling layers which are mainly used for the downsampling module, and defining the structure as the masks on the others, *i.e.*, 3×3 separable convolutions, 3×3 dilated separable convolutions, identity and *zero*.

Activating the expressivity of BN. Apart from sharpening the posterior, we also address an issue of batch normalization (BN) [23] to enhance performance. As suggested by DARTS [35], we use ReLU-Conv-BN style operations in DBSN. However, the learnable affine transformations in BNs are required to be disabled to prevent the scaling effect of the structure mask being cancelled out. As a consequence, the expressivity of the model is discounted. To address this issue and in turn enhance the data fitting of DBSN, we propose to place a complete batch normalization in the front of the next operation. Namely, we adopt the BN-ReLU-Conv-BN style operations, where the first BN has learnable affine parameters while the second one does not.

4 Related Work

Learning flexible Bayesian models has long been the goal of the community [38, 41, 4, 53, 26]. The stochastic variational inference methods for Bayesian neural networks are particularly appealing owing to their analogy to the ordinary back-propagation [16, 6]. More expressive distributions, such as matrix-variate Gaussians [50] or multiplicative normalizing flows [37], have also been introduced to represent the posterior dependencies, but they are hard to train without heavy approximations. Recently, there is an increasing interest in developing Adam-like optimizers to perform natural-gradient variational inference for BNNs [60, 3, 28]. Despite enabling the scalability, these methods seem to demonstrate compromising performance compared to the state-of-the-art deep models. Interpreting the stochastic techniques of the deep models as Bayesian inference is also insightful [13, 30, 52, 40, 32], but these methods still have relatively restricted and inflexible posterior approximations. [10] propose a unified Bayesian framework to infer the posterior of both the network weights and the structure, which is most similar to DBSN, but the structure considered by them, *i.e.*, layer size and network depth, is essentially impractical for complicated deep models. Instead, we inherit the structure representation of NAS, and provide insightful techniques to improve the convergence, thus enabling effective Bayesian structure learning for deep networks.

Neural architecture search (NAS) has drawn tremendous attention, where reinforcement learning [62, 63, 45], evolution [46] and Bayesian optimization [25] have all been introduced to solve it. More recently, differentiable NAS [35, 58, 8, 57] is attractive because it reduces the prohibitive computational cost immensely. However, existing differentiable NAS methods search the network structure in a meta-learning way [11], and need to re-train another network with the pruned compact structure after the searching. In contrast, DBSN unifies the learning of weights and structure in one training stage, alleviating the mismatch of structures during the search and re-training, as well as inefficiency issues suffered by differentiable NAS. Concurrently, BASE [48] has been proposed to learn a universal representation for architecture search over multiple tasks, which is also formulated in a Bayesian perspective and share a similar inspiration with ours. DBSN, however, focuses on the problem of uncertainty estimation and is well motivated by the difficulties of modeling weight uncertainty, thus is significantly distinct from BASE.

Table 1: Comparison with popular DNNs in terms of the number of parameters and test error rate.

Method	Params (M)	CIFAR-10 (%)	CIFAR-100 (%)
ResNet [18]	1.7	6.61	-
ResNet (pre-activation) [19]	1.7	5.46	24.33
DenseNet [22]	1.0	5.24	24.42
DenseNet-BC [22]	0.8	4.51	22.27
DBSN	1.0	4.98 \pm 0.24	22.50 \pm 0.26

5 Experiments

We evaluate DBSN by concerning two aspects: predictive performance and uncertainty estimation. We first introduce the experimental settings of DBSN, then describe the competing and comparable baselines implemented by ourselves, and present empirical results after that.

5.1 Experimental Settings and Comparable Baselines

Network and training details. To facilitate more effective information flow, we concatenate all the intermediate nodes along with the input to get the cell’s output. We constrain downsampling modules to be the typical BN-ReLU-Conv-Pooling operations, to ease structure learning. We set the number of nodes per cell $B = 7$. The whole network comprises 12 cells and 2 downsampling modules located at the 1/3 and 2/3 depth. The redundant operations all have 16 output channels. We set $\tau = \max(3 \times \exp(-0.000015t), 1)$ where t is the global training step. We use $T = 4$ training MC samples to estimate gradients. A momentum SGD with initial learning rate 0.1 and an Adam optimizer with learning rate 3×10^{-4} are used to optimize w and α , respectively. More details are given in Appendix B. We train DBSN for 100 epochs with batch size 64, which takes one day on 4 GTX 1080-Tis. During test, we use *100 MC samples* for Bayes ensemble for all methods with inference stochasticity (refer to Appendix C for an ablation study on this).

Elaborated baselines. We have implemented a wide range of baselines, most of which have learnable structures to make fair comparisons with DBSN, including: (i) **MAP**: a variant of *DBSN* with both point-estimate weights and point-estimate structure under L2 penalty; (ii) **MAP-fixed α** , a variant of *MAP* with fixed, uniform structure masks; (iii) **MC dropout**: a variant of *MAP* with dropout after every convolution (0.2 dropout rate); (iv) **BBB**: a variant of *MAP* with weight uncertainty (and point-estimate structure), where the fully factorized Gaussians are deployed on weights and Bayes by Backprop [6] is used for inference; (v) **FBN** (fully Bayesian network): a variant of *DBSN* with weight uncertainty, where Bayes by Backprop is also adopted for inference; (vi) **NEK-FAC**: a *VGG16* network (3.7M parameters) with weight uncertainty, where the noisy EK-FAC [3] algorithm is used for inference (it is hard to implement this algorithm with learnable structure); (vii) **VOGN**: a variant of *DBSN* with weight uncertainty, where the Variational Online Gauss-Newton [28] is employed to infer weight posterior (but due to its high inefficiency, we only provide its results in small model size in Appendix D). Note that (iv)-(vii) all locate in the variational BNN family. Besides, we build a NAS baseline *DARTS* to further figure out if Bayesian principles benefit structure learning. *DARTS* is based on *MAP*, but we train structure on one half of the training set yet train weights on the other half [35]; after that, we re-train a new model based on the derived compact structure for final comparison.

5.2 Predictive Performance on CIFAR-10 and CIFAR-100

We compare DBSN to popular DNNs as well as elaborated baselines in aspect of predictive performance on CIFAR-10 and CIFAR-100 tasks. We repeat every experiment for 3 times and report the averaged error rate and standard deviation in Table 1 and 2.

As shown, DBSN is notably comparable with or even better than the popular DNN models. DBSN only presents modestly higher error rates than DenseNet-BC [22]. The comparisons highlight the practical value of DBSN.

More convincingly, as shown in Table 2, the comparisons between DBSN and elaborated baselines help to confirm the effectiveness of DBSN, and the supremacy of learning distribution of structure over weights. MAP and MAP-fixed α present worse results than DBSN, which

Table 2: Comparison with elaborated and comparable baselines.

Method	CIFAR-10 (%)	CIFAR-100 (%)
DBSN	4.98 \pm 0.24	22.50 \pm 0.26
MAP	5.79 \pm 0.34	24.19 \pm 0.17
MAP-fixed α	5.66 \pm 0.24	24.27 \pm 0.15
MC dropout	5.83 \pm 0.19	23.67 \pm 0.28
BBB	9.85 \pm 0.42	30.98 \pm 0.36
FBN	9.57 \pm 0.55	31.39 \pm 0.06
NEK-FAC	7.43	37.47
DARTS	10.52 \pm 4.84	27.36 \pm 0.92

Table 3: Comparison of model calibration in terms of the Expected Calibration Error (ECE). Smaller is better.

Method	DBSN	MAP	MAP-fixed α	MC dropout	BBB	FBN	NEK-FAC
CIFAR-10	0.0109	0.0339	0.0327	0.0150	0.0745	0.0966	0.0434
CIFAR-100	0.0599	0.1240	0.1259	0.0617	0.0700	0.1091	0.1665

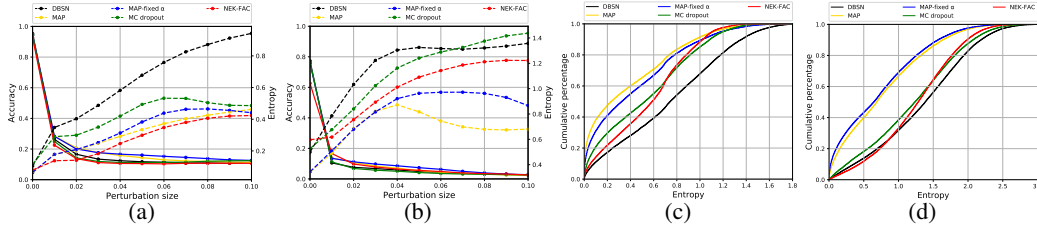


Figure 3: (a)(b): accuracy (solid) and entropy (dashed) vary *w.r.t.* the adversarial perturbation size on CIFAR-10 and CIFAR-100, respectively. (c)(d): empirical CDF for the entropy of the predictive distributions of OOD samples (curves closer to the bottom right corner are better).

may result from the over-fitting of the point-estimate parameters. MC dropout is defeated by DBSN, and even cannot obviously improve upon MAP. We speculate this is because the weight samples in MC dropout suffer from the mode collapse issue, namely, different weight samples correspond to the same network function, making the posterior predictive (*i.e.*, Bayes ensemble) ineffective [12]. DBSN alleviates this via more global and impactful structure stochasticity, proved by the results.

BBB and FBN both exhibit compromising results, strongly verifying the fundamental difficulties of modeling distributions over high dimensional weights, even with a learnable network structure. NEK-FAC also demonstrates rather undesirable performance, despite one of the most advanced variational BNNs algorithms with the powerful VGG16 architecture. All these comparisons suggest a preference to DBSN instead of the classic BNNs in the scenarios where the performance is a major concern. Also of note that DARTS is defeated by DBSN, and we found that most edges in the derived network structure are skip connections (which is widely observed by the community, see also [59]). But DBSN does not have such problem.

At last, we provide a refinement of DBSN by explicitly modeling the dependency of network weights on structure with extra training cost. We sample 10 diverse structures from the learned structure posterior of DBSN, and train 10 networks with these structures individually. We uniformly ensemble these networks to predict. The error rate on CIFAR-10 is 3.96%, slightly surpassing DBSN (4.98%). As a baseline, we randomly select one from the 10 structures and train a deep ensemble (10 networks with the this structure but various weights), yielding 4.33% CIFAR-10 error rate. This comparison helps us to further confirm that diverse structures used in Bayes ensemble benefit accuracy.

5.3 Evaluation on Uncertainty Estimates

To validate that DBSN can yield promising predictive uncertainty, we evaluate it in terms of calibration, predictive uncertainty on adversarial examples, and that on out-of-distribution (OOD) samples.

Calibration. Calibration is orthogonal to the accuracy [32] and can be well estimated by the Expected Calibration Error (ECE) [17]. Thus, we evaluate the trained models on the test set of CIFAR-10 and CIFAR-100 and calculate their ECE, as shown in Table 3. We also plot some reliability diagrams [17] in Appendix E, to provide a direct explanation of calibration. DBSN outperforms the golden baseline MC dropout slightly, revealing its practical value. BBB, FBN, and NEK-FAC present much worse ECE than DBSN, implying structure uncertainty’s superiority over weight uncertainty. As reference, we also train a DenseNet-40-12 (1M) and a DenseNet-BC-100-12 (0.8M) on CIFAR10. The two models yield 0.0338 and 0.0290 ECEs (with 5.55% and 4.89% error rates), highlighting that DBSN (0.0109 ECE and 4.98% error rate) conjoins good accuracy and calibration.

Predictive uncertainty on the adversarial examples. To test if the trained model *knows what it knows*, we assess their uncertainty estimates on adversarial examples, with the predictive entropy as a proxy, suggested by existing works [37, 43]. Concretely, we apply the frequently adopted, performant fast gradient sign method (FGSM) [15] to attack the posterior predictive distribution of the trained models on CIFAR-10 and CIFAR-100. Then we calculate the predictive entropy of the crafted adversarial examples and depict the average entropy *w.r.t.* the adversarial perturbation size in Fig. 3(a)(b). We do not include BBB and FBN into comparison owing to their compromising performance and calibration. As expected, the entropy of DBSN grows rapidly as perturbation size

Table 4: Comparison of semantic segmentation performance on CamVid dataset. * indicates results from our implementation.

Method	Pretrained	Params (M)	Mean IoU	Global accuracy
SegNet [2]	✓	29.5	46.4	62.5
Bayesian SegNet [27]	✓	29.5	63.1	86.9
FC-DenseNet67 [24]	✗	3.5	63.1*	90.4*
DBSN	✗	3.3	65.4	91.4

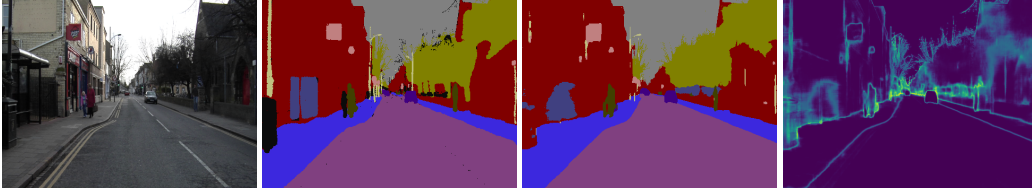


Figure 4: Visualization of the segmentation results of DBSN on CamVid. From left to right: original image, ground-truth segmentation, the estimated segmentation, and pixel-wise predictive uncertainty. The black color in ground-truth labels represents the void class.

increases, implying DBSN becomes pretty uncertain when encountering adversarial perturbations. By contrast, the change in the entropy of MC dropout and NEK-FAC is relatively moderate, showing less sensitivity to the adversarial examples. We further attack with more powerful algorithms, *e.g.*, the Basic Iterative Method (BIM) [31], and provide the results in Appendix F.

Predictive uncertainty on the out-of-distribution samples. One further step, we look into the entropy of the predictive distributions on OOD samples, to adequately evaluate the quality of uncertainty estimation. We use the trained models on CIFAR-10 and CIFAR-100 to predict test data of SVHN. We calculate their predictive entropy as uncertainty estimates and draw the empirical CDF of the entropy in Fig. 3(c)(d), following [37]. The curve close to the bottom right corner is expected as it means most OOD samples have relatively large entropy (*i.e.*, low prediction confidence). Obviously, DBSN is less susceptible to OOD samples than the baselines.

5.4 Semantic Segmentation on CamVid

At last, to show DBSN is readily application to diverse scenarios, we generalize DBSN to the challenging semantic segmentation task on CamVid [7]. Our implementation is based on the brief FC-DenseNet framework [24]. Specifically, we only replace the original dense blocks with structure-learnable cells, without introducing further advanced techniques from the semantic segmentation community, to figure out the performance gain only from the proposed structure learning approach. For the setup, we set $B = 5$ (same as the number of layers in every dense block of FC-DenseNet67) and $T = 1$, and learn two cell structures for the downsampling path and upsampling path, respectively. The other settings follow [24] and the classification experiments above. We also implement FC-DenseNet67 for fair comparison. The results are reported in Table 4 and Fig. 4.

It is evident that DBSN surpasses the competing FC-DenseNet67 by a large margin while using fewer parameters. DBSN also demonstrates significantly better performance than the classic Bayesian SegNet which adopts MC dropout for uncertainty estimation. Though a direct comparison to [26] is meaningless due to various implementations, DBSN’s performance gain over the shared baseline DenseNet even defeats [26]. In Fig. 4, it is also worth noting that the uncertainty produced by DBSN is interpretable: the edges of the objects and the regions which contain overlapping have substantially higher uncertainty than the other parts. These results validate the potential of DBSN to be applied into diverse, challenging tasks.

6 Conclusion

In this work, we have introduced a novel Bayesian structure learning approach for DNNs. The proposed DBSN draws the inspiration from NAS and models the network structure as Bayesian variables. Stochastic variational inference is employed to jointly learn the weights and the distribution of the structure. We further develop techniques to facilitate the convergence of the whole model. DBSN has revealed impressive performance on the discriminative learning tasks, surpassing the advanced deep models, and presented outperforming predictive uncertainty in various scenarios. In conclusion, DBSN provides a more practical way for Bayesian deep learning, without compromise between the predictive performance and the Bayesian uncertainty.

Broader Impact

This work proposes a Bayesian structure learning approach for deep models, conjoining good performance and benefits from Bayesian principle. Its potential positive impacts in the society are evident: its ability to enable uncertainty estimation while maintaining predictive performance is crucial in industry, *e.g.*, automatic driving, disease analysis, and financial applications. In this scenarios, the uncertainty provided by DBSN could be used to reject uncertain predictions, and raise the requirement of inviting humans into the decision process. Another potential benefit which has not been explored in this work is that DBSN is likely to be capable of achieving reliable modeling on limited data, since the Bayesian formulation helps to alleviate over-fitting. As a fundamental research in machine learning, the negative consequences are not obvious. Though in theory any techniques can be misused, it is not likely to happen at the current stage.

References

- [1] Ryan Adams, Hanna Wallach, and Zoubin Ghahramani. Learning the structure of deep sparse graphical models. In *International Conference on Artificial Intelligence and Statistics*, pages 1–8, 2010.
- [2] Vijay Badrinarayanan, Ankur Handa, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *arXiv preprint arXiv:1505.07293*, 2015.
- [3] Juhan Bae, Guodong Zhang, and Roger Grosse. Eigenvalue corrected noisy natural gradient. *arXiv preprint arXiv:1811.12565*, 2018.
- [4] Anoop Korattikara Balan, Vivek Rathod, Kevin P Murphy, and Max Welling. Bayesian dark knowledge. In *Advances in Neural Information Processing Systems*, pages 3438–3446, 2015.
- [5] Matthew J Beal, Zoubin Ghahramani, et al. Variational Bayesian learning of directed graphical models with hidden variables. *Bayesian Analysis*, 1(4):793–831, 2006.
- [6] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622, 2015.
- [7] Gabriel J Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In *European Conference on Computer Vision*, pages 44–57. Springer, 2008.
- [8] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2019.
- [9] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Learning and policy search in stochastic dynamical systems with Bayesian neural networks. *arXiv preprint arXiv:1605.07127*, 2016.
- [10] Georgi Dikov and Justin Bayer. Bayesian learning of neural network architectures. In *International Conference on Artificial Intelligence and Statistics*, pages 730–738, 2019.
- [11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135, 2017.
- [12] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.
- [13] Yarín Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.
- [14] Soumya Ghosh, Jiayu Yao, and Finale Doshi-Velez. Structured variational learning of Bayesian neural networks with horseshoe priors. In *International Conference on Machine Learning*, pages 1739–1748, 2018.
- [15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
- [16] Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pages 2348–2356, 2011.

- [17] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330, 2017.
- [18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.
- [19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645, 2016.
- [20] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.
- [21] Geoffrey Hinton and Drew Van Camp. Keeping neural networks simple by minimizing the description length of the weights. In *ACM Conference on Computational Learning Theory*, 1993.
- [22] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017.
- [23] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- [24] Simon Jégou, Michal Drozdal, David Vazquez, Adriana Romero, and Yoshua Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 11–19, 2017.
- [25] Kirthivasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Neural architecture search with Bayesian optimisation and optimal transport. In *Advances in Neural Information Processing Systems*, pages 2016–2025, 2018.
- [26] Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, pages 5574–5584, 2017.
- [27] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015.
- [28] Mohammad Emtiyaz Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and scalable Bayesian deep learning by weight-perturbation in adam. In *International Conference on Machine Learning*, pages 2616–2625, 2018.
- [29] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [30] Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583, 2015.
- [31] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.
- [32] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.
- [33] P. Langley. Crafting papers on machine learning. In Pat Langley, editor, *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pages 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.
- [34] Christian Leibig, Vaneeda Allken, Murat Seçkin Ayhan, Philipp Berens, and Siegfried Wahl. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific Reports*, 7(1):1–14, 2017.
- [35] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.
- [36] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose Bayesian inference algorithm. In *Advances in Neural Information Processing Systems*, pages 2378–2386, 2016.

- [37] Christos Louizos and Max Welling. Multiplicative normalizing flows for variational Bayesian neural networks. In *International Conference on Machine Learning*, pages 2218–2227, 2017.
- [38] David JC MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992.
- [39] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.
- [40] Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate Bayesian inference. *Journal of Machine Learning Research*, 18(1):4873–4907, 2017.
- [41] Radford M Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1995.
- [42] Kazuki Osawa, Siddharth Swaroop, Anirudh Jain, Runa Eschenhagen, Richard E Turner, Rio Yokota, and Mohammad Emtiyaz Khan. Practical deep learning with Bayesian principles. *arXiv preprint arXiv:1906.02506*, 2019.
- [43] Nick Pawłowski, Andrew Brock, Matthew CH Lee, Martin Rajchl, and Ben Glocker. Implicit weight uncertainty in neural networks. *arXiv preprint arXiv:1711.01297*, 2017.
- [44] Tim Pearce, Mohamed Zaki, Alexandra Brintup, and Andy Neely. Expressive priors in Bayesian neural networks: Kernel combinations and periodic functions. *arXiv preprint arXiv:1905.06076*, 2019.
- [45] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *International Conference on Machine Learning*, pages 4092–4101, 2018.
- [46] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4780–4789, 2019.
- [47] Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured Laplace approximations for overcoming catastrophic forgetting. In *Advances in Neural Information Processing Systems*, pages 3738–3748, 2018.
- [48] Albert Shaw, Wei Wei, Weiyang Liu, Le Song, and Bo Dai. Meta architecture search. In *Advances in Neural Information Processing Systems*, pages 11225–11235, 2019.
- [49] Jiaxin Shi, Shengyang Sun, and Jun Zhu. Kernel implicit variational inference. In *International Conference on Learning Representations*, 2018.
- [50] Shengyang Sun, Changyou Chen, and Lawrence Carin. Learning structured weight uncertainty in Bayesian neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 1283–1292, 2017.
- [51] Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional variational Bayesian neural networks. In *International Conference on Learning Representations*, 2019.
- [52] Mattias Teye, Hossein Azizpour, and Kevin Smith. Bayesian uncertainty estimation for batch normalized deep networks. In *International Conference on Machine Learning*, pages 4914–4923, 2018.
- [53] Hao Wang and Dit-Yan Yeung. Towards Bayesian deep learning: A framework and some existing methods. *IEEE Transactions on Knowledge and Data Engineering*, 28(12):3395–3408, 2016.
- [54] Ziyu Wang, Tongzheng Ren, Jun Zhu, and Bo Zhang. Function space particle optimization for Bayesian neural networks. In *International Conference on Learning Representations*, 2019.
- [55] Florian Wenzel, Kevin Roth, Bastiaan S Veeling, Jakub Świątkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the Bayes posterior in deep neural networks really? *arXiv preprint arXiv:2002.02405*, 2020.
- [56] Frank Wood, Thomas L Griffiths, and Zoubin Ghahramani. A non-parametric Bayesian method for inferring hidden causes. In *Conference on Uncertainty in Artificial Intelligence*, pages 536–543, 2006.
- [57] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019.

- [58] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: stochastic neural architecture search. In *International Conference on Learning Representations*, 2019.
- [59] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *International Conference on Learning Representations*, 2020.
- [60] Guodong Zhang, Shengyang Sun, David Duvenaud, and Roger Grosse. Noisy natural gradient as variational inference. In *International Conference on Machine Learning*, pages 5847–5856, 2018.
- [61] Jingwei Zhuo, Chang Liu, Jiaxin Shi, Jun Zhu, Ning Chen, and Bo Zhang. Message passing Stein variational gradient descent. In *International Conference on Machine Learning*, pages 6013–6022, 2018.
- [62] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.
- [63] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2018.

Supplementary Material for: Measuring Uncertainty through Bayesian Learning of Deep Neural Network Structures

Zhijie Deng, Yucen Luo, Jun Zhu
Department of Computer Science
Tsinghua University

{dzj17, luoyc15}@mails.tsinghua.edu.cn, dcszj@mail.tsinghua.edu.cn

A Derivation of the Log Probability Density of the Concrete Distribution with Reduced Variance

For clear expressions, We simply denote $\alpha^{(i,j)}$, $\theta^{(i,j)}$, $\beta^{(i,j)}$ and $\epsilon^{(i,j)}$ as α , θ , β and ϵ , respectively. Let $\mathbf{p} = \text{softmax}(\theta)$. Consider

$$\begin{aligned}\alpha_k &= \frac{\exp((\theta_k + \beta\epsilon_k)/\tau)}{\sum_{i=1}^K \exp((\theta^i + \beta\epsilon^i)/\tau)} \\ &= \frac{\exp((\log \mathbf{p}_k + \beta\epsilon_k)/\tau)}{\sum_{i=1}^K \exp((\log \mathbf{p}^i + \beta\epsilon^i)/\tau)}.\end{aligned}$$

Let $\mathbf{z}_k = \log \mathbf{p}_k + \beta\epsilon_k = \log \mathbf{p}_k - \beta \log(-\log(\mathbf{u}_k))$, where $\mathbf{u}_k \sim \mathcal{U}(0, 1)$ i.i.d.. It has density

$$\frac{1}{\beta} \mathbf{p}_k^{1/\beta} \exp(-\frac{\mathbf{z}_k}{\beta}) \exp(-\mathbf{p}_k^{1/\beta} \exp(-\frac{\mathbf{z}_k}{\beta})).$$

We denote $c = \sum_{i=1}^K \exp(\mathbf{z}_i/\tau)$, then $\alpha_k = \exp(\mathbf{z}_k/\tau)/c$. Consider this transformation:

$$F(\mathbf{z}_1, \dots, \mathbf{z}_K) = (\alpha_1, \dots, \alpha_{K-1}, c),$$

which has the following inverse transformation:

$$\begin{aligned}F^{-1}(\alpha_1, \dots, \alpha_{K-1}, c) &= (\tau(\log \alpha_1 + \log c), \dots, \\ &\quad \tau(\log \alpha_K + \log c)),\end{aligned}$$

whose Jacobian has the determinant (refer to the derivation of the concrete distribution [?]):

$$\frac{\tau^K}{c \prod_{i=1}^K \alpha_i}.$$

Multiply this with the density of \mathbf{z} , we get the density

$$\begin{aligned}&\frac{\tau^K}{c \prod_{i=1}^K \alpha_i} \prod_{i=1}^K \left[\frac{1}{\beta} \mathbf{p}_i^{1/\beta} \exp(-\frac{\tau(\log \alpha_i + \log c)}{\beta}) \right. \\ &\quad \left. \cdot \exp(-\mathbf{p}_i^{1/\beta} \exp(-\frac{\tau(\log \alpha_i + \log c)}{\beta})) \right].\end{aligned}$$

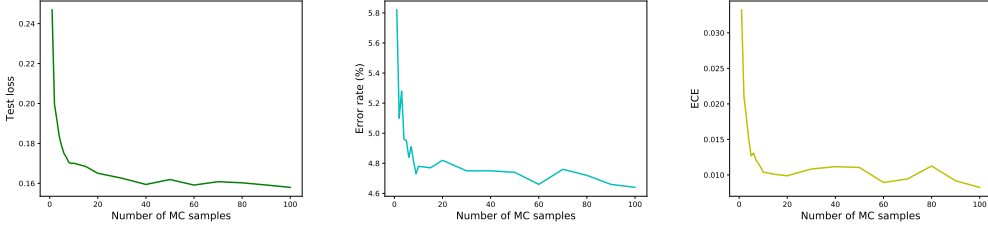


Figure 5: Test loss (left), test error rate (middle), and test ECE (right) of DBSN vary w.r.t. the number of MC samples used in estimating Eq. (7). (CIFAR-10)

Let $r = \log c$, then apply the change of variables formula, we easily obtain the density:

$$\frac{\tau^K \prod_{i=1}^K \mathbf{p}_i^{1/\beta}}{\beta^K \prod_{i=1}^K \boldsymbol{\alpha}_i^{(1+\tau/\beta)}} \exp\left(-\frac{K\tau r}{\beta}\right) \cdot \exp\left(-\sum_{i=1}^K (\mathbf{p}_i \boldsymbol{\alpha}_i^{-\tau})^{1/\beta} \exp\left(-\frac{\tau r}{\beta}\right)\right).$$

We use γ to substitute $\log \sum_{i=1}^K (\mathbf{p}_i \boldsymbol{\alpha}_i^{-\tau})^{1/\beta}$, then get:

$$\frac{\tau^K \prod_{i=1}^K \mathbf{p}_i^{1/\beta}}{\exp(\gamma) \beta^K \prod_{i=1}^K \boldsymbol{\alpha}_i^{(1+\tau/\beta)}} \exp\left(\gamma - \frac{K\tau r}{\beta}\right) \cdot \exp\left(-\exp\left(\gamma - \frac{\tau r}{\beta}\right)\right).$$

Naturally, we can integrate out r , and get:

$$\begin{aligned} & \frac{\tau^K \prod_{i=1}^K \mathbf{p}_i^{1/\beta}}{\exp(\gamma) \beta^K \prod_{i=1}^K \boldsymbol{\alpha}_i^{(1+\tau/\beta)}} \left[\frac{\beta}{\tau} \exp(\gamma - K\gamma) \Gamma(K) \right] \\ &= \frac{\tau^{K-1} \prod_{i=1}^K \mathbf{p}_i^{1/\beta}}{\beta^{K-1} \prod_{i=1}^K \boldsymbol{\alpha}_i^{(1+\tau/\beta)}} \exp(-K\gamma) \Gamma(K) \\ &= \frac{((K-1)!) \tau^{K-1}}{\beta^{K-1} \prod_{i=1}^K \boldsymbol{\alpha}_i} \times \frac{\prod_{i=1}^K (\mathbf{p}_i \boldsymbol{\alpha}_i^{-\tau})^{1/\beta}}{(\sum_{i=1}^K (\mathbf{p}_i \boldsymbol{\alpha}_i^{-\tau})^{1/\beta})^K}. \end{aligned}$$

Then, the log density is:

$$\begin{aligned} & \log((K-1)!) + (K-1) \log \frac{\tau}{\beta} - \sum_{i=1}^K \log \boldsymbol{\alpha}_i \\ &+ \sum_{i=1}^K \frac{\log \mathbf{p}_i - \tau \log \boldsymbol{\alpha}_i}{\beta} - K \cdot \text{L}\Sigma\text{E} \frac{\log \mathbf{p}_i - \tau \log \boldsymbol{\alpha}_i}{\beta} \\ &= \log((K-1)!) + (K-1) \log \frac{\tau}{\beta} - \sum_{i=1}^K \log \boldsymbol{\alpha}_i \\ &+ \sum_{i=1}^K \frac{\boldsymbol{\theta}_i - \tau \log \boldsymbol{\alpha}_i}{\beta} - K \cdot \text{L}\Sigma\text{E} \frac{\boldsymbol{\theta}_i - \tau \log \boldsymbol{\alpha}_i}{\beta}, \end{aligned}$$

which is equal to Eq. (9).

B Detailed Experiment settings

In the classification tasks, the whole network is composed of 12 cells and 2 downsampling modules which have a channel compression factor of 0.4 and are located at the 1/3 and 2/3 depth. We

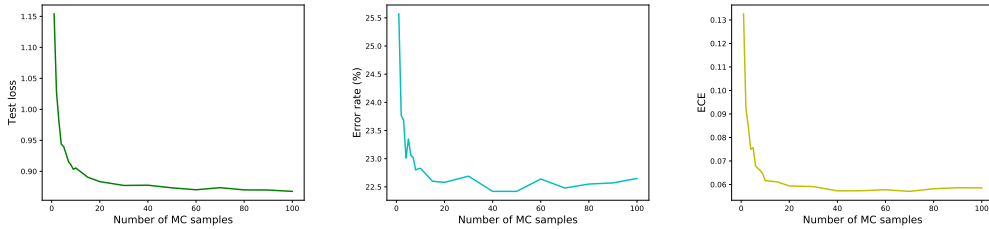


Figure 6: Test loss (left), test error rate (middle), and test ECE (right) of DBSN vary w.r.t. the number of MC samples used in estimating Eq. (7). (CIFAR-100)

Table 6: Comparisons between DBSN and competing baselines which deploy uncertainty on weights and adopt VOGN algorithm for variational inference on weights. Baseline *VOGN* performs posterior inference on weights but performs MAP estimation on structure. Baseline *VOGN-FBN* performs posterior inference on both weights and structure, constructing a fully Bayesian network. (CIFAR-10)

	Training time (hours)	Test error rate (%)	ECE
DBSN	1.2	9.90	0.0070
<i>VOGN</i>	13.0	28.4	0.5391
<i>VOGN-FBN</i>	13.0	30.5	0.5169

employ a 3×3 convolution before the first cell and put a global average pooling followed by a fully connected (FC) layer after the last cell. In the cell, we let the input of a cell (i.e., the output of the previous cell) be fixedly connected to all the internal nodes by $1 \times 1/3 \times 3$ convolutions in the classification/segmentation task, as shown in Appendix G. We set $\tau = \max(3 \times \exp(-0.000015t), 1)$ where t is the global training step. We initialize w and θ following [?] and [?], respectively. A momentum SGD with initial learning rate 0.1 (divided by 10 at 50% and 75% of the training procedure following [?]), momentum 0.9 and weight decay 10^{-4} is used to train the weights w . We clip the gradient norm $\|\nabla_w\|$ at 5. An Adam optimizer with learning rate 3×10^{-4} , momentum (0.5, 0.999) is used to learn θ . We deploy the standard data augmentation scheme (mirroring/shifting) and normalize the data with the channel statistics. The whole training set is used for optimization.

In the segmentation task, we use a momentum SGD with initial learning rate 0.01 (which decays linearly after 350 epochs), momentum 0.9 and weight decay 10^{-4} instead of the original RMSprop for better results.

C The Effects of the Number of MC Samples in Test Phase

We draw the change of test loss, test error rate and test ECE with respect to the number of MC samples used for testing DBSN in Fig. 5 (CIFAR-10) and Fig. 6 (CIFAR-100). It is clear that ensembling the predictions from models with various sampled network structures enhances the final predictive performance and calibration significantly. This is in marked contrast to the situation of classic variational BNNs, where using more MC samples does not necessarily bring improvement over using the most likely sample. As shown in the plots, we would better utilize 20+ MC samples to predict the unseen data, for adequately exploiting the learned structure distribution. Indeed, we use 100 MC samples in all the experiments, except for the adversarial attack experiments where we use 30 MC samples for attacking and evaluation.

D Comparisons between DBSN and Competing Baselines using VOGN for Inferring Weight Posterior

We realized the Bayes by Backprop [?] method used for inferring weight posterior in *BBB* and *FBN* baselines may be restrictive, resulting in such weakness. Therefore, we replaced it with Variational Online Gauss-Newton (VOGN) [?], a most-recently proposed mean-field natural-gradient variational inference method. VOGN is known to work well with advanced techniques, e.g., momentum, batch normalisation, data augmentation. As claimed by [?], VOGN demonstrates

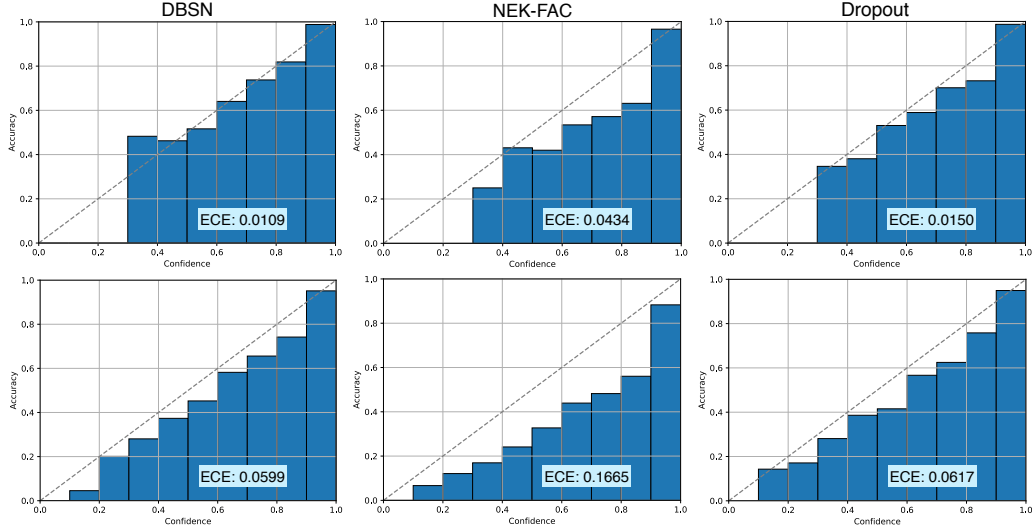


Figure 7: Reliability diagrams for DBSN, NEK-FAC, and MC dropout on CIFAR-10 (top row) and CIFAR-100 (bottom row). The bars aligning more closely to the diagonal are preferred. Smaller ECE is better.

comparable results to Adam. Our implementation is based on VOGN’s official repository (<https://github.com/team-approx-bayes/dl-with-bayes>). With the original network size ($B = 7$, 12 cells), the baselines trained with VOGN needed more than one hour for one epoch, which is very slow. Thus we adopted smaller networks ($B = 4$, 3 cells), which have almost 41K parameters, for the two baselines. We also trained a DBSN in the same setting. The experiments were conducted on CIFAR-10 and the results are provided in Table 6. The predictive performance and uncertainty gaps between DBSN and the two baselines are very huge, which possibly results from the under-fitting of the high-dim weight distributions. We think that our implementation is correct because our results are consistent with the original results in Table 1 of [?] (VOGN has 75.48% and 84.27% validation accuracy even with even larger 2.5M AlexNet and 11.1M ResNet-18 architectures). Further, DBSN is much more efficient than them. These comparisons strongly reveal the benefits of modeling structure uncertainty over modeling weight uncertainty.

E More Results for Calibration

We plot the reliability diagrams of 3 typical methods, which represent BNN with structure uncertainty, BNN with weight uncertainty, and deterministic NN with MC dropout, respectively, in Fig. 7. Obviously, DBSN has better reliability diagrams than NEK-FAC and MC dropout, proving the effectiveness of the uncertainty on network structure.

F Attack with BIM

We perform an adversarial attack using BIM algorithm. Concretely, we set the number of iteration to be 3 and set the perturbation size in every step to be 1/3 of the whole perturbation size. The experiments mainly focus on the models trained on CIFAR-10. Fig. 8 shows the results. DBSN has increasing entropy when the perturbation size changes from 0 to 0.01, but all the other approaches are attacked successfully with entropy dropping. However, we have to agree that BIM is powerful enough to break all the methods, including DBSN. So we advise adjusting DBSN accordingly (e.g., employing adversarial training, using more robust loss) if we want to use DBSN to defend the adversarial attacks.

G Visualization of the Learned Structures

We visualize the learned structure distributions on different tasks in Fig. 9, Fig. 10, Fig. 11 and Fig. 12 (we do not draw the *zero* operation). The structure distributions learned on different tasks look

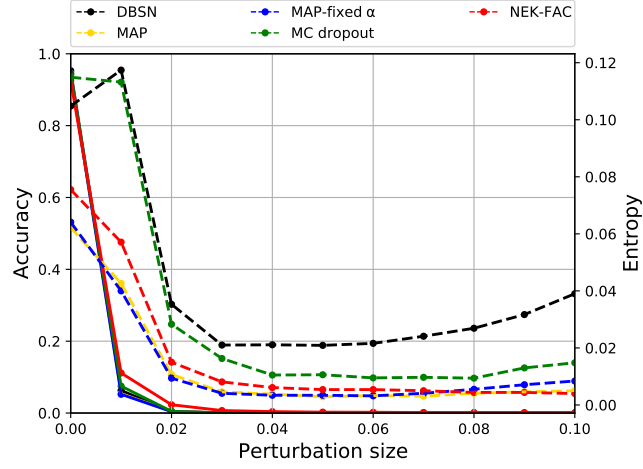


Figure 8: Accuracy (solid) vs entropy (dashed) as a function of the adversarial perturbation size on CIFAR-10. Attack with BIM.

different, validating that DBSN can adapt the network structure according to the data distribution flexibly.

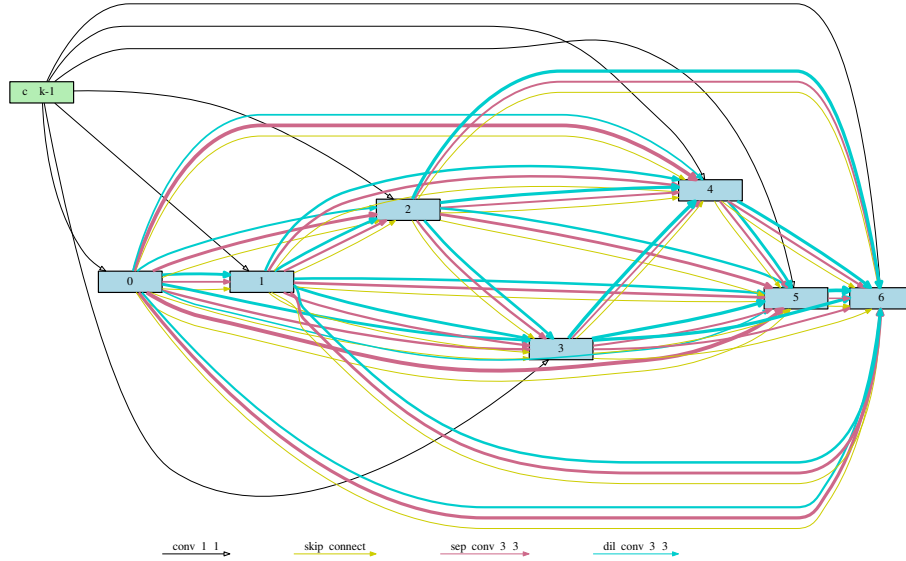


Figure 9: Structure of the cell learned on CIFAR-10. The width of an edge implies the sampling probability of its corresponding operation.

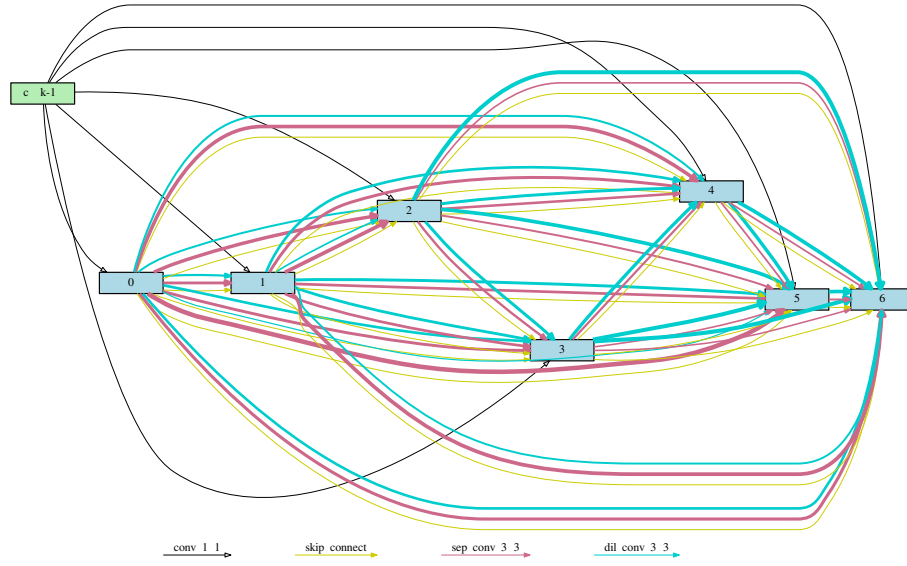


Figure 10: Structure of the cell learned on CIFAR-100. The width of an edge implies the sampling probability of its corresponding operation.

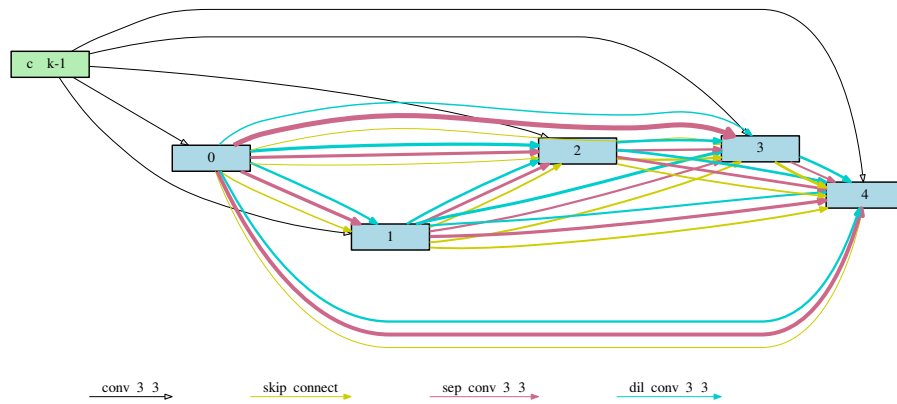


Figure 11: Structure of the cell learned on CamVid (in the downsampling path). The width of an edge implies the sampling probability of its corresponding operation.

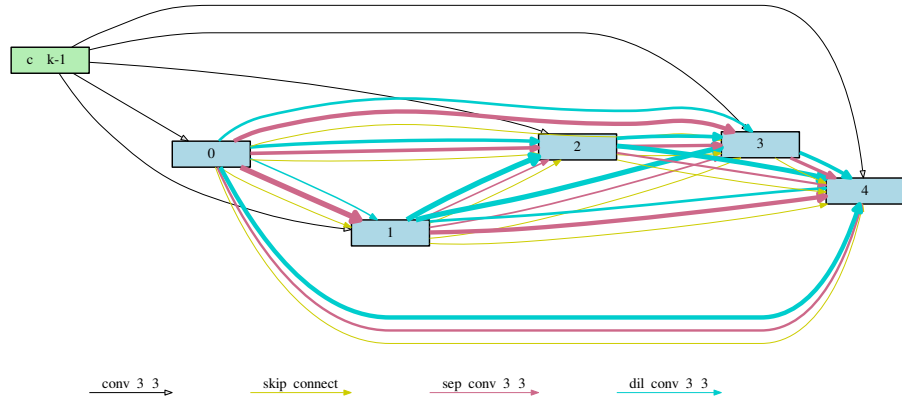


Figure 12: Structure of the cell learned on CamVid (in the upsampling path). The width of an edge implies the sampling probability of its corresponding operation.