Favour: FAst Variance Operator for Uncertainty Rating

Thomas D. Ahle
Meta
ahle@fb.com

Sahar Karimi
Meta
sahark@fb.com

Peter Tak Peter Tang
Meta
ptpt@fb.com

Abstract

Bayesian Neural Networks (BNN) have emerged as a crucial approach for interpreting ML predictions. By sampling from the posterior distribution, data scientists may estimate the uncertainty of an inference. Unfortunately many inference samples are often needed, the overhead of which greatly hinder BNN's wide adoption. To mitigate this, previous work proposed propagating the first and second moments of the posterior directly through the network. However, on its own this method is even slower than sampling, so the propagated variance needs to be approximated such as assuming independence between neural nodes. The resulting trade-off between quality and inference time did not match even plain Monte Carlo sampling.

Our contribution is a more principled variance propagation framework based on "spiked covariance matrices", which smoothly interpolates between quality and inference time. This is made possible by a new fast algorithm for updating a diagonal-plus-low-rank matrix approximation under various operations. We tested our algorithm against sampling based MC Dropout and Variational Inference on a number of downstream uncertainty themed tasks, such as calibration and out-of-distribution testing. We find that Favour is as fast as performing 2-3 inference samples, while matching the performance of 10-100 samples.

In summary, this work enables the use of BNN in the realm of performance critical tasks where they have previously been out of reach.

1 Introduction

Machine learning and Neural Network (NN) in particular have greatly exceeded expectations in their effectiveness at a diverse set of workloads including classifications, recommendations, anomaly detection, human-machine interaction, autonomous navigation; and the list goes on. Traditional NN is learned (trained) by optimizing a loss function with a set of network parameters (weights) and input data as the function's arguments. The weights are real numbers and the learned NN produces point estimates of the answer to the task at hand. For example, a binary classification NN may render a judgement that the medical image examined shows that there exist a 70% chance of cancer. But the traditional NN cannot convey the uncertainty of the opinion of 70%. Nevertheless, answers with some quantification of uncertainty are crucial as many critical and previously exclusively human controlled tasks are increasingly shared with artificial intelligence. For this example, it makes a big difference

whether the chance of cancer is $70 \pm 2\%$ or $70 \pm 25\%$. The use for uncertainty quantification in ML is evident in many auto decision processes.

In contrast to traditional NN, Bayesian Neural Network (BNN) treats the network's weight parameters not as real numbers but real-valued random variables whose distributions are to be learned via training. Similarly, results of an inference made with BNN is not a real number but the distribution of a real-valued random variable. See Figure 1 for the use of prediction distribution to reveal aleatoric and epistemic uncertainty.

Designing and training BNNs is an involved subject as the Bayesian framework often involves intractable integration of probability densities. Innovative approximations that make training BNN practical constitute the bulk of active researches. In comparison, attention to performing BNN inference is lacking. One reason is that the conceptually easy method of sampling is applicable to many (though not all) BNNs. After all, the weights are distributions. One can make multiple inferences (take multiple samples) each on the same input data but with different random drawing of the weights according to their distributions. Uncertainty quantification can be made with the collection of the resulting point estimate results. While sampling is easy to understand and implement, a moderate number of samples in the order of tens, if not more, need to be collected to ensure an acceptable uncertainty quantification. This consumes roughly ten fold of compute resources. In many industrial scale problem where deterministic inferences fully utilize compute capacity, this ten fold increase cannot be hidden at all.

Sample free methods for BNN inference have been proposed previously. In the "Expectation propagation" framework we approximate the true posterior distribution with a simpler approximation. Choosing a Multivariate Gaussian, minimizing the KL-Divergence to the true posterior is possible if we can compute the first two moments. This means we follow the evolution of not just the input multi-dimensional random variable when it passes through a BNN, but also its mean and covariance. Since the distribution of the involved random variables are very often well characterised by these two statistics, variance propagation can provide information comparable to that obtained by large sampling. The main drawback is the high cost of this technique. Roughly speaking, a linear layer with a n-by-n weight matrix has an inference cost of $O(n^2)$ while propagating the covariance costs $O(n^3)$.

By approximating covariance matrices in a diagonal-plus-low-rank (DPLR) form, also called spiked covariance, we derived the evolution of these approximate covariance and devised efficient algorithms to follow their evolution. Accuracy can improve with larger rank; although our experiments show that a small rank of 2 produces uncertainty rating of the quality comparable to that obtained by large samplings. The cost is of the order of performing inference with a deterministic NN, and in the case of MLP, the cost can be made sub-inference through further approximations.

Our contributions are several fast algorithms that propagate spiked variance through a general Bayesian Neural Network layer thus covering many network models including regression, computer vision and recommendation systems. Propagation of the spiked covariance leads to sample free substitute for MC Dropout and Variational Inference. These fast algorithms offer a Pareto curve of tradeoff between speed and accuracy, resulting in better uncertainty quantification then large samplings at a lower cost.

In what follows, Section 2 reviews related works. Section 3 presents the basic mathematics of mean and variance propagation. Section 4 explains the computational algorithm we developed to propagate spiked variance at inference complexity. We also present other fast algorithms such as an approximation to the Jensen Shannon Divergence. Section 5 presents results on using variance propagation on a number of uncertainty rating tasks. Section 6 summarises our results and present directions for future work.

2   Background

Different designs for probabilistic neural networks have been made and [14, 6, 16] give excellent survey and tutorial on the subject. Some architectures most relevant to our work include the following. A deep ensemble of models is suggested in [15] for attaining the
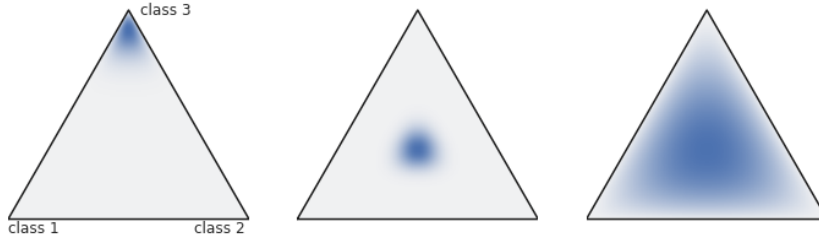
Figure 1: **Uncertainty in a classification problem with 3 classes.** Typical softmax inference gives a discrete probability distribution (p_1, p_2, p_3), but in Bayesian NNs we predict a probability distribution over such probability distributions. In the picture: (a) A confident prediction of a class. (b) A confident prediction of data/aleatoric uncertainty. (c) High epistemic uncertainty, typical of out-of-distribution data.

predictive distribution. Despite its simplicity, deep ensemble requires training multiple copies of the model which can be prohibitively expensive for realistic and large scale models. Batch ensemble [24] and Monte Carlo dropout [5] are scalable alternatives to formulating a model's weight parameters as general distributions. Another class of scalable approaches is *variational inference* (VI) [13]. Gaussian mean-field VI are proposed in [8, 2]. In [3], Bayesian neural nets with rank-1 variational factor are suggested where weight matrices are Hadamard products of a deterministic matrix and a rank-1 variational one. Going beyond mean-field VI is investigated in [19, 4]. We refer to these different architectures of probabilistic networks by the general term Bayesian Neural Network, or BNN. Training BNNs are more involved both conceptually as well as computationally than their traditional counter parts. Nevertheless, in situations where the input data distribution changes very slowly, such as natural language processing or medical diagnosis, the cost of training can be amortized over the lifetime of the network's deployment. But the additional cost in making inference with BNN compared to traditional NN cannot be amortized in general.

In Bayesian inference, the predictive distribution of an unknown target $\mathbf{y}$ is $p(\tilde{\mathbf{y}}|\tilde{\mathbf{x}}, D) = \int_{\mathbf{w}} p(\tilde{\mathbf{y}}|\tilde{\mathbf{x}}, \mathbf{w})p(\mathbf{w}|D)$; in practice, however, most algorithms rely on sampling from the posterior of w to attain the predictive posterior. MC Dropout [5] is an alternative to sampling the posterior of $\mathbf{w}$. Here the weights are deterministic, that is, real values; but probabilistic dropout layers are used even during inference time. A number of inference result samples are collected on one fixed input data. The samples of inference (point estimate) results represent the posterior distribution of the BNN inference. Sampling is thus a convenient approach to realize BNN inference for several classes of BNN designs. Unfortunately, it is unsure how many samples one need to get a reliable representation of a posterior distribution. Empirically, ten or twenty samples are used, and sometimes five samples are used in the face of compute performance constraint. This constraint is real in many different situations: decision in autonomous navigation, recommendation systems that need to remain highly responsive, real-time blockage of harmful contents, etc.

Sample free methods have been proposed before such as [23] where a deterministic model that produces uncertainty quantification is trained after the BNN in question is trained. Alternatively, our approach closely resembles that of [22]. It addresses BNN expressed by Monte Carlo Dropout [5]. The main idea is the mean and variance of a multidimensional random variable $\mathbf{x} \in \mathbb{R}^n$ whose mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\Sigma}$ are known (perhaps approximately) will evolve deterministically. In the MC Dropout setting, a linear layer $W\mathbf{x} + \mathbf{b}$ is deterministic and thus the mean and variance after this layer becomes $W\boldsymbol{\mu} + \mathbf{b}$ and $W\boldsymbol{\Sigma}W^T$, respectively. Unfortunately this propagation has a higher order complexity compare to simple inference: If $W \in \mathbb{R}^{n \times n}$, then inference costs $O(n^2)$ but computing $W\boldsymbol{\Sigma}W^T$ costs $O(n^3)$. The straightforward complexity reduction by assuming $\boldsymbol{\Sigma}$ to be diagonal, hence assuming

the components of $\mathbf{x}$ to be mutually independent, is too restrictive to offer any gains over straightforward sampling.

We reduce the complexity of sample free variance propagation method back to that of deterministic inference without assuming $\boldsymbol{\Sigma}$ to be diagonal, but of the general form of a non-negative diagonal plus a low-rank semidefinite matrix. One can view our as an extension of [1] unto inference. This method is applicable to any BNN expressible in terms of weight parameters or layers that are adequately determined by means and variances. In particular, MC Dropout, or "learned" dropout with per-neuron dropout distribution, mean-field and generalized mean-field Variational Inference.

3 Mean and Variance Propagation

Our focus is on inference with a probabilistic network so as to produce result with some uncertainty rating. Sampling is a straightforward technique where inference with one sample of the probabilistic network and input produces one sample inference result. Gathering sufficient result samples allows various uncertainty quantification be made. Empirically, the number of samples S needed to make reliable quantification is in the order of tens, which consumes S times as much compute resources.

A sample free alternative is this. Given a random variable $\mathbf{x} \in \mathbb{R}^n$ with mean and (co)variance $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, we wish to determine the mean and variance $(\boldsymbol{\mu}', \boldsymbol{\Sigma}')$ of the output $\mathbf{x} \in \mathbb{R}^m$ of a generic probabilistic neural network layer of the form $\mathbf{y} = A(W(D\mathbf{x}) + \mathbf{b})$. $A(\cdot)$ is a deterministic non-linear activation function such as ReLU, sigmoid or tanh. The affine transformation $W\mathbf{x} + \mathbf{b}$ can be deterministic, that is, the entries in the W matrix and $\mathbf{b}$ vector are simple real numbers, or probabilistic when those entries are distributions. D is a elementwise (diagonal matrix) generalized dropout operator where each element is a scaled Bernoulli random variable of possibly different dropout probabilities p. That is, the value is 0 with probability p and $1/q$ with probability $q = 1 - p$. The parameters $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ characterize important distributions such as Gaussian that are considered good representation of the underlying distributions of $\mathbf{y}$. This generic layer encapsulates a wide range of network architectures and represents multiple Bayesian realizations including MC Dropout and Variational Inference.

Consider the simple case of $\mathbf{y} = W\mathbf{x} + \mathbf{b}$ where $W, \mathbf{b}$ are deterministic but $\mathbf{x} \sim (\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Straightforward statistical derivations show $\boldsymbol{\mu}' = W\boldsymbol{\mu} + \mathbf{b}$ and $\boldsymbol{\Sigma}' = W \boldsymbol{\Sigma} W^T$. In this section we state $(\boldsymbol{\mu}', \boldsymbol{\Sigma}')$ for our generic BNN layer and full derivations are given in the appendix for completeness. We note that our treatment of affine transformation and non-linear functions are more general than [22] where the weights are deterministic, [11] where the neurons are assumed independent or [17] where one propagates a bound on the variance. We devote the next section to the details where we reduce the apparent $O(n^3)$ complexity of $\boldsymbol{\Sigma}' = W \boldsymbol{\Sigma} W^T$ to $O(n^2)$ or less.

3.1 Propagation through Dropout Layers

Let $\mathbf{p} \in \mathbb{R}_+^n$ be the dropout probabilities for each of $\mathbf{x}$'s components. D is a scaled Bernoulli random variable such that $E[D] = I$, the identity matrix yielding $\boldsymbol{\mu}' = \boldsymbol{\mu}$. Then $V[D\mathbf{x}] = \boldsymbol{\Sigma} + \mathrm{diag}((\boldsymbol{\mu}^2 + \mathrm{diag}(\boldsymbol{\Sigma})) \odot \mathbf{p}/\mathbf{q})$, where $\odot$ is the Hadamard (pointwise) product. The computational complexity is a low $O(n)$ and note that $\mathbf{p}/\mathbf{q}$ simplifies trivially to the scalar p/q if dropout probabilities are identical for all of $\mathbf{x}$'s components.

3.2 Propagation through Linear Layers

Consider now a linear layer $\mathbf{y} = W\mathbf{x} + \mathbf{b}$ where $W \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$. Both W and $\mathbf{b}$ can be distributions instead of deterministic, with $\overline{W}$ and $\bar{\mathbf{b}}$ be their respective means. Furthermore, the entries within W or $\mathbf{b}$ need not be mutually independent while we always assume W and $\mathbf{b}$ are independent of each other. We propagate the mean $\boldsymbol{\mu}'$ of $\mathbf{y}$ by $\boldsymbol{\mu}' = \overline{W}\boldsymbol{\mu} + \bar{\mathbf{b}}$. For the variance, we have

$$(\mathrm{Var}[W\mathbf{x} + \mathbf{b}])_{ij} = (\overline{W}\boldsymbol{\Sigma}\overline{W}^T + \Sigma_{\mathbf{b}})_{ij} + \mathrm{SUM}(\Sigma_{\mathbf{w},ij} \odot (\boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T)) \tag{1}$$

Equation 3 simplifies in some common special cases.

1. If the linear layer is deterministic, then $\mathbf{\Sigma_w}, \mathbf{\Sigma_b}$ are zero and the variance of $\mathbf{y}$ is simply $W\mathbf{\Sigma}W^T$.

2. If the entries of W and $\mathbf{b}$ are all mutually independent, such as in the case of mean-field VI, then $\mathbf{\Sigma_b}, \mathbf{\Sigma_w}$ are diagonal matrices. A natural way to represent $\mathbf{\Sigma_w}$ is by $W_{\sigma^2} \in \mathbb{R}^{m \times n}$ whose entries are the variances of w_{ij} and for $\mathbf{\Sigma_b}$ the vector $\mathbf{b}_{\sigma^2} \in \mathbb{R}^m$. Thus $\mathbf{\Sigma}' = \overline{W}\mathbf{\Sigma}\overline{W}^T + \mathrm{diag}\left(\mathbf{b}_{\sigma^2} + W_{\sigma^2}\left(\mathrm{diag}(\mathbf{\Sigma}) + \boldsymbol{\mu}^2\right)\right).$

3. Consider the generalized mean-field VI where each different rows of the W matrix are independent, but not so withing each row. In this case, $\mathrm{SUM}(\Sigma_{\mathbf{w},ij} \odot (\mathbf{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T))$ is non-zero only when $i = j$ and thus this expression reduces to $\mathrm{SUM}(\Sigma_{\mathbf{w},ii} \odot (\mathbf{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T))$.

3.3   Propagation through Univariate Activations

An activation function $A(\cdot)$ such as ReLU or sigmoid are univariate nonlinear functions applied to each component of an in $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$. Similar to existing work, we approximate the i-th component of $\boldsymbol{\mu}'$ by taken the expectation of $A(x)$ using $x \sim N(\mu_i, \sigma_i^2)$. This computation of $\boldsymbol{\mu}'$ is of $O(n)$ complexity.

There are two choices for variance propagation. The first approach uses the first order Taylor expansion at the input mean $\boldsymbol{\mu}$: $\mathbf{y} \approx A(\boldsymbol{\mu}) + D \cdot (\mathbf{x} - \boldsymbol{\mu})$, $D \stackrel{\mathrm{def}}{=} \mathrm{diag}([A'(\boldsymbol{\mu}_i); i = 1, 2, \ldots, n])$. This leads to the approximation $\mathbf{\Sigma}' \approx D\mathbf{\Sigma}D^T$, which is merely scaling $\mathbf{\Sigma}_{ij}$ by $A'(\boldsymbol{\mu}_i)A'(\boldsymbol{\mu}_j)$. In particular, for ReLU activation, this is zeroing out the i-th row and column whenever $\boldsymbol{\mu}_i \leq 0$. This sparsification may have have some desirable regularization effect. Note however that the covariance values in $\mathbf{\Sigma}$ remain unchanged when not made zero. This does not match the case if we assume $\mathbf{x}$ is Gaussian. This motives our second approach. We compute $\sigma_i' = \sqrt{\mathrm{Var}[A(x)]}$ for $x \sim N(\mu_i, \sigma_i^2)$ and $D \stackrel{\mathrm{def}}{=} \mathrm{diag}([\sigma_i'/\sigma_i \text{ if } \sigma_i > 0 \text{ else } 0])$. We scale $\mathbf{\Sigma}$ by $D\mathbf{\Sigma}D^T$. This guarantees that the diagonal value of $\mathbf{\Sigma}'$ matches the Gaussian assumption.

4   Diagonal Plus Low Rank Approximation

As described in the introduction, a Diagonal Plus Low Rank (DPLR) approximation to a covariance matrix is a natural way to reduce inference time while keeping a control on the quality trade-off incurred. In the appendix (appendix A.1) we describe now many useful, established properties of this decomposition. Here we present the most novel and interesting ideas for our application, which is propagating the approximation through the layers of a (Bayesian) neural network.

4.1   Fast DPLR Linear Propagation

Consider the computation of $M = \overline{W}\mathbf{\Sigma}\overline{W}^T \in \mathbb{R}^{m \times m}$ where $\overline{W} \in \mathbb{R}^{m \times n}$. Straightforward computation requires $O(mn^2)$ operations. We can however obtain a DPLR approximation to M with complexity commensurate with model inference. We exploit the DPLR structure of $\mathbf{\Sigma} = \Lambda + UU^T$ and use subspace iteration to obtain the DPLR approximation to M which only requires matrix-vector product with M without needing M explicitly.

We formulate our problem as

$$\mathrm{argmin}_{\Lambda,V} \rho(\Lambda, V), \ \Lambda = \mathrm{diag}(\boldsymbol{\lambda}), \boldsymbol{\lambda} \in \mathbb{R}_+^m, \ V \in \mathbb{R}^{m \times r} \text{ and } \rho(\Lambda, V) \stackrel{\mathrm{def}}{=} \|M - (\Lambda + VV^T)\|_F^2,$$

and give a heuristic algorithm that runs in $O(mnr + mr^2)$ time. This is based on the alternating projections algorithm for Robust PCA algorithm [21], but we suspect the particular version we need is NP Hard to solve.

The basic idea is to update V and Λ alternately while keeping the other fixed. Given the current iterate Λ_{now}, the symmetric matrix $M - \Lambda_{\mathrm{now}}$ has an eigendecomposition

$$M - \Lambda_{\mathrm{now}} = \psi_1 \mathbf{z}_1 \mathbf{z}_1^T + \psi_2 \mathbf{z}_2 \mathbf{z}_2^T + \cdots + \psi_m \mathbf{z}_m \mathbf{z}_m^T$$

$\psi_1 \geq \psi_2 \geq \cdots \geq \psi_m$. The best Frobenius norm (as well as 2-norm) rank-r semi-definite approximation is $\sum_{\ell=1}^k \psi_\ell^+ \mathbf{z}_\ell \mathbf{z}_\ell^T$ where $\psi_\ell^+ \stackrel{\mathrm{def}}{=} \max(\psi_\ell, 0)$ [12]. Provided $\psi_r > |\psi_j|$ for all $j > r$,

the rank-r truncated SVD of $M - \Lambda_{\text{now}}$ is $Z_{(r)} \operatorname{diag}([\psi_1, \psi_2, \ldots, \psi_r]) Z_{(r)}^T$. In this case, we obtain $V_{\text{next}} = Z_{(r)} \operatorname{diag}([\sqrt{\psi_1}, \sqrt{\psi_2}, \ldots, \sqrt{\psi_r}])$ Because this V_{next} is the best approximation, we have $\rho(\Lambda_{\text{now}}, V_{\text{next}}) \leq \rho(\Lambda_{\text{now}}, V_{\text{now}})$.

On the other hand, fixing V_{now}, it is easy to see that $\Lambda_{\text{next}} \leftarrow \max(\operatorname{diag}(M - V_{\text{now}} V_{\text{now}}^T), \mathbf{0})$ is the optimal and will lead to a descent of ρ.

Classical subspace iteration algorithms [18] for obtaining r-truncated SVD to a matrix M is well established and has order of complexity that of M times r vectors. Recent developments of randomized linear algebra [9] improve the performance further. We developed the following greedy algorithm by interleaving the alternating direction search with a subspace iteration prompted by the effectiveness of randomized initialization.

Algorithm 1 Fast DPLR Decomposition

Require: $\Lambda, U, \Lambda \in \mathbb{R}_+^{n \times n}$ diagonal, $U \in \mathbb{R}^{n \times r}$, W; $M = W(\Lambda + UU^T)W^T$.
1:   $V \sim N(0,1)^{m \times r}$ $\triangleright$ Or best guess of final low-rank matrix, $O(mr)$ time
2:   **for** $i = 1, 2, \ldots, K$ **do** $\triangleright$ Typically $K = 2, 3$ or 4
3:      $S \leftarrow \operatorname{diag}([\|v_1\|_2, \|v_2\|_2, \ldots, \|v_r\|_2])$ $\triangleright$ $O(mr)$ time
4:      $\Lambda \leftarrow \operatorname{diag}(M - V S^{-1} V^T)$ $\triangleright$ $O(mr)$ time
5:      $\Lambda \leftarrow \max(\Lambda, 0)$ $\triangleright$ (Component-wise max), $O(mr)$ time
6:      $V \leftarrow V S^{-1}$ $\triangleright$ (Make columns unit-norm), $O(mr)$ time
7:      $V \leftarrow (M - \Lambda) V$ $\triangleright$ $O(mnr)$ time
8:      $V \leftarrow \text{Unnormalized-Gram-Schmidt}(V)$, $\triangleright$ $O(mr^2)$ time
9:   **end for**
10:   $S \leftarrow \operatorname{diag}([\|v_1\|_2, \|v_2\|_2, \ldots, \|v_r\|_2])$ $\triangleright$ $O(mr)$ time
11:   $\Lambda \leftarrow \max(\operatorname{diag}(M - V S^{-1} V^T), 0)$ $\triangleright$ $O(mr)$ time
12:   **return** $(\Lambda, V S^{-1/2})$

The matrix-vectors product on Line 7 does not compute M explicitly but exploits the structure $M = W(\Lambda + UU^T)W^T$. See Section A.1 for more details. The normalization of U on Line 6 is not strictly necessary, but if the number of loop iterations (K) is large we might otherwise get numerical overflows. The dominant complexity is $O(mnr)$ which is of the order of making r inferences. In practice, r is typically set between 2 and 4. Moreover, in the case of linear layers where W is dense and compute intensive, we have replaced it with a low-rank approximation of rank $q \leq n/10$ in Algorithm 1 and obtained good uncertainty rating result. In this case, the complexity is $O(mqr)$ which can be lower than model inference.

4.2 Fast DPLR Propagation for Convolution

Convolution is a linear operator but with special structure: Expressed as a matrix it is sparse and Toeplitz. Nevertheless, we use the convolution and transposed convolution operators to realize the computation in the previous section by exploiting the DPLR structure. The diagonal and each rank-1 component of the DPLR representation and each "vector" used in subspace iteration is naturally in 2D shape. The W and W^T operations translates to application of `Conv2d` and `ConvTranspose2d` (in PyTorch for example).

4.3 Fast Jensen-Shannon Divergence Approximation of DPLR Variance

For classification problems, the most common last layer of the network is a softmax operator. Typically, one collect samples of the softmax results $\mathbf{p}^{(1)}, \mathbf{p}^{(2)}, \ldots, \mathbf{p}^{(n)}$ and computes the Jensen-Shannon Divergence $\mathcal{J}(\mathbf{p}^{(1)}, \mathbf{p}^{(2)}, \ldots, \mathbf{p}^{(n)}) = \mathcal{H}(\overline{\mathbf{p}}) - \frac{1}{n} \sum_\ell \mathcal{H}(\mathbf{p}^{(\ell)})$ where $\mathcal{H}(\mathbf{p}) \overset{\text{def}}{=} \langle -\mathbf{p}, \log(\mathbf{p}) \rangle$ is the entropy. We proved that the following sample free estimate accurately approximates the JSD: $\left(1 - \frac{1}{n}\right) \frac{1}{2} \left(\langle \mathbf{p}^*, \operatorname{diag} \mathbf{\Sigma} \rangle - \langle \mathbf{p}^*, \mathbf{\Sigma} \mathbf{p} \rangle \right)$ where $\mathbf{p}^* = S(\boldsymbol{\mu})$ is the softmax of the mean $\boldsymbol{\mu}$. This computation is of $O(n)$ when $\mathbf{\Sigma}$ is of DPLR form. The full statement and proof are given in the Appendix. Figure 2 illustrates the estimate's effectiveness.

We note that variance in DPLR form facilitates efficient execution of common operations such as sampling, inverse computation and linear transformation that we detail in the appendix.
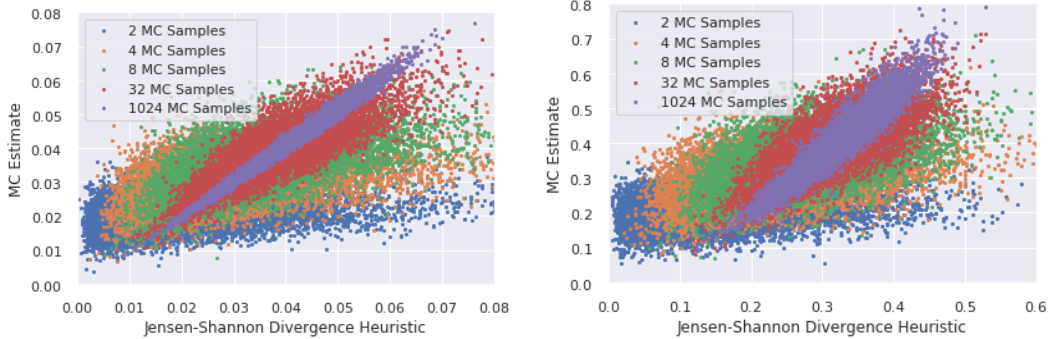
(a) JSD on Softmax on 10 dimensional random Gaussian data with small covariance.

(b) JSD on Softmax on 10 dimensional random Gaussian data with larger covariance.

Figure 2: **Concordance between JSD estimates from samples and ??.** The more MC Dropout samples are used, the better the JSD formula agrees with the estimate. This suggests the approximation gets close to the true JSD much faster than the sampling method, specially when the JSD scores are not too large.

Datasets	Model	Inference	Area Under ROC Curve			
			JSD	MaxP.	Ent.	Maha.
		Freq.	—	64.5 ± 0.0	64.2 ± 0.0	67.9 ± 0.0
	Bayes	Favour	79.6 ± 0.0	63.6 ± 0.0	63.3 ± 0.0	77.2 ± 0.0
		MC	78.5 ± 0.1	65.3 ± 0.1	65.0 ± 0.1	72.2 ± 0.1
		Freq.	—	77.2 ± 0.0	77.0 ± 0.0	93.3 ± 0.0
	DO 10%	Favour	82.8 ± 0.0	77.3 ± 0.0	77.1 ± 0.0	93.9 ± 0.0
Train dist:		MC	85.1 ± 0.1	83.1 ± 0.1	82.1 ± 0.1	93.7 ± 0.1
MNIST		Freq.	—	72.3 ± 0.0	72.0 ± 0.0	75.5 ± 0.0
Out of dist:	DO 25%	Favour	88.9 ± 0.0	72.0 ± 0.0	71.7 ± 0.0	78.2 ± 0.0
Fashion		MC	90.9 ± 0.2	73.0 ± 0.1	72.0 ± 0.2	79.5 ± 0.1
MNIST		Freq.	—	66.6 ± 0.0	65.7 ± 0.0	93.4 ± 0.0
	DO 50%	Favour	76.8 ± 0.0	67.7 ± 0.0	66.7 ± 0.0	94.5 ± 0.0
		MC	83.2 ± 0.3	78.0 ± 0.3	73.7 ± 0.2	89.2 ± 0.1

Table 1: **OOD performance at similar computational costs.** A Le-net model was trained with dropout on the two top linear layers (alternatively Bayesian linear layers) for 400 epochs MNIST and or 250 epochs on Cifar-10. Each inference mode was performed on the test set of the in-distribution as well as a different distribution's test set. Different measures of uncertainty were computed and we report the area under the ROC curve for a decision process based on those scores. The inference and auc loop was performed 10 times to measure variation due to random sampling or otherwise.

5 Experiments

To measure the time/quality trade-off of the epistemic uncertainty estimates produced by Favour, we evaluate it on the downstream tasks of *confidence calibration* and *out-of-distribution (OOD) detection*.

In *confidence calibration* we desire the model to give an accurate estimate of its possible errors. For instance, in a regression task, we expect the target value to be within the 80% confidence interval 80% of the time. In a classification task, we aim for a lower calibration error [10] as well as a model's ability to distinguish confident and uncertain predictions [20].
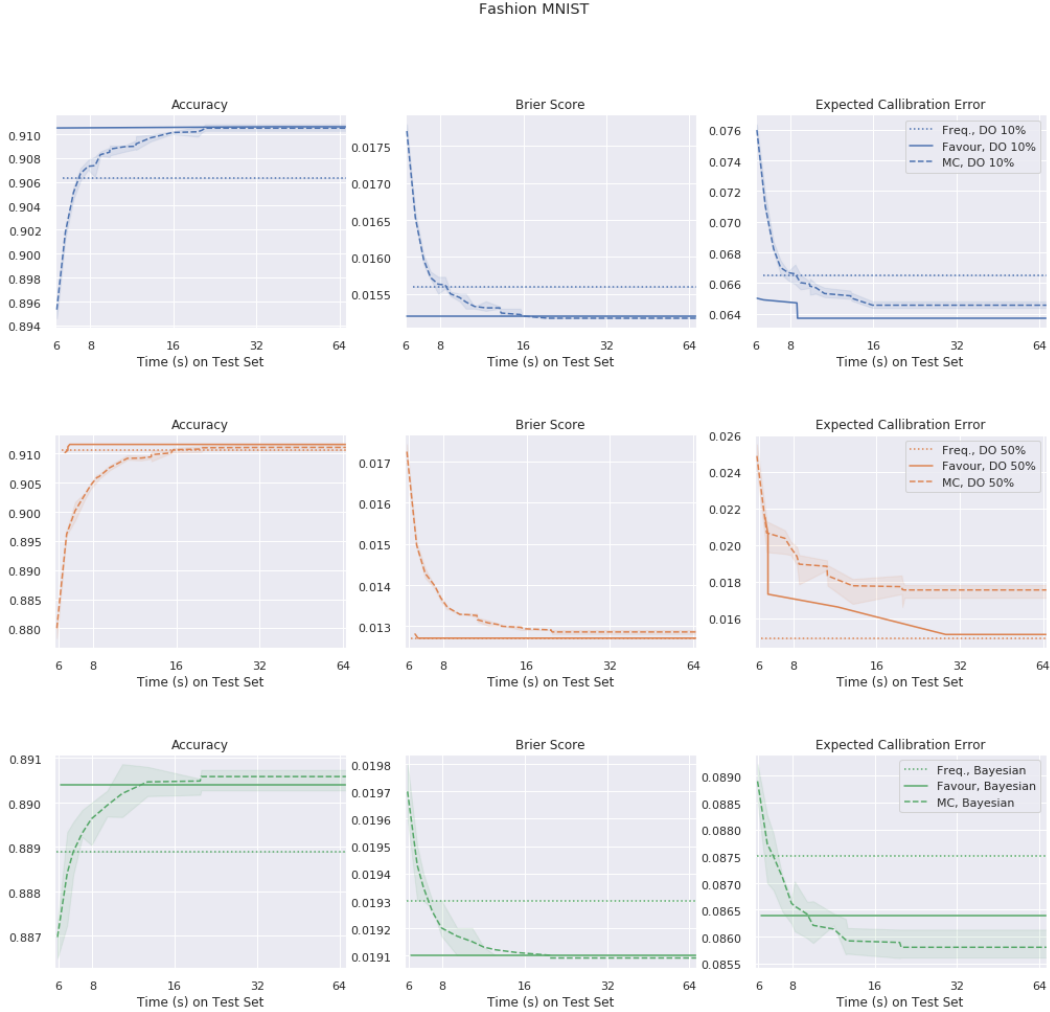
7

Figure 3: **Classification performance as computational budget increases.** Favour achieves better quality of inference than MCDO when given a small computational budget. We trained a CNN with dropout on the top two linear layers, and computed the inference Accuracy, Brier Score and ECE for different parameter choices (e.g. number of MC samples and rank of covariance matrix in Favour.)

In *OOD detection* it is assumed that a model is more uncertain about its prediction on data points that are unlike what it has been trained on. By transforming the uncertainty predicted by the model into a single number, we can fit a binary classifier to distinguish between uncertainty scores of in-distribution and out-of-distribution data. If the assumption holds, and the uncertainty estimates are good, this classifier should have a large AUC-ROC.

5.1 Confidence Calibration

For classification tasks, we compare expected calibration error (ECE) between Favour and other sampling-based Bayesian approximations on a CNN model trained on Fashion MNIST (fig. 3) and CIFAR10 respectively.

5.2 Out of Distribution Detection

For OOD detection, we use the predictive distribution to estimate the uncertainty of the model; hence the uncertainty score can be interpreted as a discriminator score to separate in and out of distribution samples. With this binary classification setting in mind, we use the

Area Under the receiver operating characteristic curve (ROC) and precision-recall (PR) as the performance metrics.

We compute OOD based on four different uncertainty scores: 1) Jensen Shannon Divergence (JSD), which is a measure of epistemic uncertain, and 2-4) Maximum Probability, Entropy and Mahalanobis Distance, which are measures of aleoric unertainty. (Recall the difference from fig. 1.)

Table 1 reports our results in the case where the network is trained on MNIST and the OOD distribution is Fashion MNIST. We see that Favour performs similarly to MC-Dropout in this case, when given a similar computational budget. Ideally we would like it to perform better than MCDO, at least at very small inferences times. This will be further explored in the suplemental materials.

In the suplemental materials we consider two settings: near OOD and far OOD; where near OOD are instances where the OOD samples are from semantically similar distribution. For near OOD, we used two settings: training on seven digits of MNIST and testing on the filtered classes; and testing a trained model on CIFAR10 on CIFAR 100. For far OOD, we tested a trained classifier on MNIST on Fashion-MNIST, and a trained classifier on CIFAR10 on SVHN.

6   Conclusion

We have shown that variance propagation can be a viable alternative to sampling based techniques on a wide range of tasks. By our subspace iterative algorithm, variance in DPLR form can be propagated at inference or even sub-inference complexity. While theoretically one may need a high rank value in the "low rank" part of DPLR, our experiments show that a very small rank value suffices to capture the variance for many uncertainty rating tasks. In essence, our approach enables the actual deployments of many BNNs. As our next steps, we plan explore how to apply this fast variance propagation idea into BNN learning, and also explore sub-inference complexity variance propagation of convolutional neural networks.

References

[1] David Barber and Christopher M. Bishop. Ensemble learning in bayesian neural networks. In *Neural Networks and Machine Learning*, pages 215–237. Springer, 1998.

[2] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. ICML'15, page 1613–1622. JMLR.org, 2015.

[3] Michael W. Dusenberry, Ghassen Jerfel, Yeming Wen, Yi-An Ma, Jasper Snoek, Katherine A. Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and scalable Bayesian neural nets with rank-1 factors. *CoRR*, abs/2005.07186, 2020.

[4] Audrey Flower, Beliz Gokkaya, Sahar Karimi, and Ilknur Kaynar Kabul. RouBL: A computationally efficient way to go beyond mean-field variational inference. ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning, 2021.

[5] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR.

[6] Ethan Goan and Clinton Fookes. Bayesian neural networks: An introduction and survey. In *Case Studies in Applied Bayesian Data Science*, pages 45–87. Springer International Publishing, 2020.

[7] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.

[8] Alex Graves. Practical variational inference for neural networks. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[9] M. Gu. Subspace iteration randomization and singular value problems. *SIAM Journal on Scientific Computing*, 37(3):A1139–A1173, January 2015.

[10] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330, 2017.

[11] José Miguel Hernández-Lobato and Ryan P. Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *ICML*, 2015.

[12] Nicholas J. Higham. Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra and its Applications*, 103:103–118, May 1988.

[13] Geoffrey E. Hinton and Drew van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *COLT '93*, 1993.

[14] Laurent Valentin Jospin, Wray L. Buntine, Farid Boussaïd, Hamid Laga, and Mohammed Bennamoun. Hands-on Bayesian neural networks - a tutorial for deep learning users. *CoRR*, abs/2007.06823, 2020.

[15] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[16] David J C Mackay. Probable networks and plausible predictions — a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6(3):469–505, 1995.

[17] Yuki Mae, Wataru Kumagai, and Takafumi Kanamori. Uncertainty propagation for dropout-based bayesian neural networks. *Neural Networks*, 144:394–406, December 2021.

[18] H. A. Mang. Numerical methods in finite element analysis, K. J. Bathe and E. L. Wilson, Prentice-Hall, Englewood Cliffs, N. J. *International Journal for Numerical Methods in Engineering*, 11(9):1485–1485, 1977.

[19] Aaron Mishkin, Frederik Kunstner, Didrik Nielsen, Mark Schmidt, and Mohammad Emtiyaz Khan. SLANG: fast structured covariance approximations for bayesian deep learning with natural gradient. *CoRR*, abs/1811.04504, 2018.

[20] Jishnu Mukhoti and Yarin Gal. Evaluating Bayesian deep learning methods for semantic segmentation. *ArXiv*, abs/1811.12709, 2018.

[21] Praneeth Netrapalli, Niranjan UN, Sujay Sanghavi, Animashree Anandkumar, and Prateek Jain. Non-convex robust pca. *Advances in Neural Information Processing Systems*, 27, 2014.

[22] Janis Postels, Francesco Ferroni, Huseyin Coskun, Nassir Navab, and Federico Tombari. Sampling-free epistemic uncertainty estimation using approximated variance propagation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2931–2940, 2019.

[23] Yichen Shen, Zhilu Zhang, Mert R. Sabuncu, and Lin Sun. Real-time uncertainty estimation in computer vision via uncertainty-aware distribution distillation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 707–716, January 2021.

[24] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: An alternative approach to efficient ensemble and lifelong learning. *ArXiv*, abs/2002.06715, 2020.

A Appendix

A.1 Fast Common Operations using DPLR

The DPLR (Diagonal + Low Rank) approximation allows a number of useful operations, which we use throughout Favour:

- *Sampling* We can combine variance propagation with sampling. To generate $\mathbf{z} \sim N(\boldsymbol{\mu}, \Lambda + UU^T)$, we compute $\mathbf{z} = \Lambda^{1/2}\mathbf{x} + U\mathbf{y}$ where $\mathbf{x} \sim N(0, I)^n$ and $\mathbf{y} \sim N(0, I)^r$.

- *Inversion* To compute Gaussian NNL and confidence regions we need to find $\boldsymbol{\Sigma}^{-1}$ (Or $\Sigma^{1/2}$). Assuming $\Lambda > 0$, inversion using the Woodbury formula [7] costs $O(r^3)$.

- *Multiplication* Algorithm **1** only requires $M\mathbf{v}$, $M = W(\Lambda + UU^T)W^T$. We compute in $O(n^2 r)$ complexity $M\mathbf{v}$ by $W(D(W^T\mathbf{v})) + Y(Y^T\mathbf{v})$ where $Y = WU$. Furthermore, using a rank r approximation to W leads to a sub-inference complexity of $O(nr^2)$.

A.2 Details on Mean and Variance Propagation

Given a random variable $\mathbf{x} \in \mathbb{R}^n$ with mean and (co)variance $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, we wish to determine the mean and variance $(\boldsymbol{\mu}', \boldsymbol{\Sigma}')$ of the output $\mathbf{x} \in \mathbb{R}^m$ of a probabilistic neural network layer

$$\mathbf{y} = A(W(D\mathbf{x}) + \mathbf{b}). \tag{2}$$

$A(\cdot)$ is a deterministic non-linear activation function such as ReLU, sigmoid or tanh. The affine transformation $W\mathbf{x} + \mathbf{b}$ can be deterministic, that is, the entries in the W matrix and $\mathbf{b}$ vector are simple real numbers, or probabilistic when those entries are distributions. D is a elementwise (diagonal matrix) generalized dropout operator where each element is a scaled Bernoulli random variable of possibly different dropout probabilities p. That is, the value is 0 with probability p and $1/q$ with probability $q = 1 - p$. The parameters $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ characterize important distributions such as Gaussian that are considered good representation of the underlying distributions of $\mathbf{y}$.

A.3 Propagation through Dropout Layers

Let $\mathbf{p} \in \mathbb{R}^n_+$ be the dropout probabilities for each of $\mathbf{x}$'s components. D is a scaled Bernoulli random variable such that $E[D] = I$, the identity matrix yielding $\boldsymbol{\mu}' = \boldsymbol{\mu}$. Next, the Law of Total Variance implies

$$\begin{aligned} \mathrm{Var}[D\mathbf{x}] &= \mathrm{Var}[\mathrm{E}[D\mathbf{x}|\mathbf{x}]] + \mathrm{E}[\mathrm{Var}[D\mathbf{x}|\mathbf{x}]], \\ &= \boldsymbol{\Sigma} + \mathrm{diag}(\mathrm{E}[\mathbf{x}^2] \odot \mathbf{p}/\mathbf{q}), \\ &= \boldsymbol{\Sigma} + \mathrm{diag}((\boldsymbol{\mu}^2 + \mathrm{diag}(\boldsymbol{\Sigma})) \odot \mathbf{p}/\mathbf{q}). \end{aligned}$$

The computational complexity is a low $O(n)$ as it only changes the diagonal portion of $\boldsymbol{\Sigma}$. Note that $\mathbf{p}/\mathbf{q}$ simplifies trivially to the scalar p/q if dropout probabilities are identical for all of $\mathbf{x}$'s components.

A.4 Propagation through Linear Layers

Consider now a linear layer $\mathbf{y} = W\mathbf{x} + \mathbf{b}$ where $W \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$. Both W and $\mathbf{b}$ can be distributions instead of deterministic, with $\overline{W}$ and $\overline{\mathbf{b}}$ be their respective means. Furthermore, the entries within W or $\mathbf{b}$ need not be mutually independent while we always assume W and $\mathbf{b}$ are independent of each other. We propagate the mean $\boldsymbol{\mu}'$ of $\mathbf{y}$ by $\boldsymbol{\mu}' = \overline{W}\boldsymbol{\mu} + \overline{\mathbf{b}}$. Let us focus on $\mathrm{Var}[\mathbf{y}] = \mathrm{Var}[W\mathbf{x} + \mathbf{b}] = \mathrm{Var}[W\mathbf{x}] + \Sigma_{\mathbf{b}} \in \mathbb{R}^{m \times m}$. The non-trivial term is $\mathrm{Var}[\mathbf{z}]$, $\mathbf{z} = W\mathbf{x}$.

Let $\mathbf{w}_i$ be the i-th row of W and consider the flattened W given by $\mathbf{w} \stackrel{\text{def}}{=} [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_m]^T \in \mathbb{R}^{mn}$. Consider the Kronecker product $I_m \otimes \mathbf{x}^T \in \mathbb{R}^{m \times mn}$. Think of it as the identity matrix I_m with each 0 replaced by a row of n zeros and each 1 replaced by $\mathbf{x}^T$. With this, $\mathbf{y} = W\mathbf{x} = (I_m \otimes \mathbf{x}^T)\mathbf{w}$ is expressed as a transform on the mn random variables of W_{ij}. We apply the Law of Total Variance as follows.

$$\mathrm{Var}[\mathbf{z}] = \mathrm{Var}[\mathrm{E}[\mathbf{z}|\mathbf{x}]] + \mathrm{E}[\mathrm{Var}[\mathbf{z}|\mathbf{x}]] = \overline{W}\boldsymbol{\Sigma}\overline{W}^T + \mathrm{E}[\mathrm{Var}[\mathbf{z}|\mathbf{x}]].$$

For the second term, consider
$$A = \text{Var}[\mathbf{z}|\mathbf{x}] = \text{Var}[(I_m \otimes \mathbf{x}^T)\mathbf{w}|\mathbf{x}] = (I_m \otimes \mathbf{x}^T)\Sigma_{\mathbf{w}}(I_m \otimes \mathbf{x})$$
because $(A \otimes B)^T = A^T \otimes B^T$. Note that $\Sigma_{\mathbf{w}} \in \mathbb{R}^{mn \times mn}$ and $A \in \mathbb{R}^{m \times m}$. The ij-th entry A_{ij} of A is given by
$$A_{ij} = \mathbf{x}^T \Sigma_{\mathbf{w},ij} \mathbf{x} = \sum_{r,s} (\Sigma_{\mathbf{w},ij})_{rs} x_r x_s.$$

where $\Sigma_{\mathbf{w},ij} \in \mathbb{R}^{n \times n}$ is the covariance between $\mathbf{w}_i$ and $\mathbf{w}_j$, the i-th and j-th row of W. Because $\text{E}[x_r x_s] = cov(x_r, x_s) + \mu_r \mu_s$ we have
$$\text{E}[A_{ij}] = \text{sum all elements of } (\Sigma_{\mathbf{w},ij} \odot (\boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T)).$$
Combining the above, we have
$$(\text{Var}[W\mathbf{x} + \mathbf{b}])_{ij} = (\overline{W}\boldsymbol{\Sigma}\overline{W}^T + \Sigma_{\mathbf{b}})_{ij} + \text{SUM}(\Sigma_{\mathbf{w},ij} \odot (\boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T)) \qquad (3)$$

Equation 3 simplifies in some common special cases.

1. If the linear layer is deterministic, then $\boldsymbol{\Sigma}_{\mathbf{w}}, \boldsymbol{\Sigma}_{\mathbf{b}}$ are zero and the variance of $\mathbf{y}$ is simply $W\boldsymbol{\Sigma}W^T$.

2. If the entries of W and $\mathbf{b}$ are all mutually independent, such as in the case of mean-field VI, then $\boldsymbol{\Sigma}_{\mathbf{b}}, \boldsymbol{\Sigma}_{\mathbf{w}}$ are diagonal matrices. A natural way to represent $\boldsymbol{\Sigma}_{\mathbf{w}}$ is by $W_{\sigma^2} \in \mathbb{R}^{m \times n}$ whose entries are the variances of w_{ij} and for $\boldsymbol{\Sigma}_{\mathbf{b}}$ the vector $\mathbf{b}_{\sigma^2} \in \mathbb{R}^m$. The propagated variance becomes
$$\overline{W}\boldsymbol{\Sigma}\overline{W}^T + \text{diag}\left(\mathbf{b}_{\sigma^2} + W_{\sigma^2}\left(\text{diag}(\boldsymbol{\Sigma}) + \boldsymbol{\mu}^2\right)\right)$$

3. Since each row of W transforms $\mathbf{x}$ to a new feature, we consider also the generalized mean-field VI where each different rows of the W matrix are independent, but not so withing each row. In this case, $\text{SUM}(\Sigma_{\mathbf{w},ij} \odot (\boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T))$ is non-zero only when $i = j$ and thus only updates the diagonal of $\boldsymbol{\Sigma}$. And the computation of that expression is
$$\text{SUM}(\Sigma_{\mathbf{w},ii} \odot (\boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T)). \qquad (4)$$

A.5 Propagation through Univariate Activations

An activation function $A(\cdot)$ such as ReLU or sigmoid are univariate nonlinear functions applied to each component of an in $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$. If we assume Gaussian distribution of the random variable $\mathbf{x}$ described by $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then the mean $\boldsymbol{\mu}'$ is given by
$$\mu_i' = \frac{1}{\sqrt{2\pi}\sigma_i} \int_{-\infty}^{\infty} A(x) e^{-(x-\mu_i)^2/2\sigma_i^2} \, dx.$$

For the common case of the ReLU activation, the integral has a closed form formula. In general, numerical quadrature can be used. Regardless, $\boldsymbol{\mu}'$ can be computed in $O(n)$ complexity.

There are two choices for variance propagation. The first approach uses the first order Taylor expansion at the input mean $\boldsymbol{\mu}$: For $\mathbf{y} = A(\mathbf{x})$,
$$\mathbf{y}_i = A(\boldsymbol{\mu}_i + (\mathbf{x}_i - \boldsymbol{\mu}_i)) \approx A(\boldsymbol{\mu}_i) + A'(\boldsymbol{\mu}_i) \cdot (\mathbf{x}_i - \boldsymbol{\mu}_i).$$
Thus,
$$\mathbf{y} \approx A(\boldsymbol{\mu}) + D \cdot (\mathbf{x} - \boldsymbol{\mu}), \quad D \overset{\text{def}}{=} \text{diag}([A'(\boldsymbol{\mu}_i); i = 1, 2, \ldots, n]).$$
This leads to the approximation $\boldsymbol{\Sigma}' \approx D\boldsymbol{\Sigma}D^T$, which is merely scaling $\boldsymbol{\Sigma}_{ij}$ by $A'(\boldsymbol{\mu}_i)A'(\boldsymbol{\mu}_j)$. In particular, for ReLU activation, this is zeroing out the i-th row and column whenever $\boldsymbol{\mu}_i \leq 0$. This sparsification may have have some desirable regularization effect.

Note however that the covariance values in $\boldsymbol{\Sigma}$ remain unchanged when not made zero. This does not match the case if we assume $\mathbf{x}$ is Gaussian because the variance of $\mathbf{y}_i$ is given by
$$(\sigma_i')^2 = \frac{1}{\sqrt{2\pi}\sigma_i} \int_{-\infty}^{\infty} (A(x) - \boldsymbol{\mu}_i)^2 e^{-(x-\mu_i)^2/2\sigma_i^2} \, dx$$

This motives our second approach. We define $D \overset{\text{def}}{=} \text{diag}([\sigma_i'/\sigma_i \text{ if } \sigma_i > 0 \text{ else } 0]$, and scale $\boldsymbol{\Sigma}$ by $D\boldsymbol{\Sigma}D^T$. This guarantees that the diagonal value of $\boldsymbol{\Sigma}'$ matches the Gaussian assumption.

A.6 Full Statement and Proof of JSD Approximation

Lemma 1. *Let* $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)} \in \mathbb{R}^d$ *be independent random variables with a common mean* $\boldsymbol{\mu} = \mathrm{E}[\mathbf{x}^{(\ell)}]$ *for all* ℓ *and denote by* $\boldsymbol{\Sigma}$ *the variance* $\mathrm{E}[XX^T] - \boldsymbol{\mu}\boldsymbol{\mu}^T$ *where* $X = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}]$. *Let* $\mathbf{p} \overset{def}{=} \mathrm{S}(\boldsymbol{\mu})$, $\mathbf{p}^{(\ell)} \overset{def}{=} \mathrm{S}(\mathbf{x}^{(\ell)})$ *and* J *be the Jacobian of* S *at* $\boldsymbol{\mu}$. *Then suppose the error* $\boldsymbol{\Delta}^{(\ell)} = \mathrm{S}(\mathbf{x}^{(\ell)}) - (\mathbf{p} + J(\mathbf{x}^{(\ell)}))$ *is bounded in expectation* $\|\mathrm{E}[\boldsymbol{\Delta}^{(\ell)}]\| < \varepsilon^2$ *for some* $\varepsilon > 0$ *where then we can estimate the Jensen Shannon Divergence by*

$$\mathcal{J}(\mathbf{p}^{(1)}, \mathbf{p}^{(2)}, \ldots, \mathbf{p}^{(n)}) = \left(1 - \frac{1}{n}\right) \frac{1}{2} \left(\langle \mathbf{p}, \mathrm{diag}\,\boldsymbol{\Sigma}\rangle - \langle \mathbf{p}, \boldsymbol{\Sigma}\mathbf{p}\rangle\right) + O(\varepsilon^3).$$

In particular, if $\boldsymbol{\Sigma} = \Lambda + UU^T$, *this estimate is* $\left(1 - \frac{1}{n}\right) \frac{1}{2} \left(\langle \mathbf{p}, \Lambda\rangle - \langle \mathbf{p}, (\Lambda + UU^T)\mathbf{p}\rangle\right)$.

Proof. Define $\boldsymbol{\delta}^{(\ell)}$ such that $\mathrm{S}(\mathbf{x}^{(\ell)}) = \mathbf{p} + \boldsymbol{\delta}^{(\ell)}$. The quantity of interest is

$$Q = \mathcal{J}(\mathbf{p} + \boldsymbol{\delta}^{(1)}, \mathbf{p} + \boldsymbol{\delta}^{(2)}, \ldots, \mathbf{p} + \boldsymbol{\delta}^{(n)}) = \mathcal{H}(\mathbf{p} + \bar{\boldsymbol{\delta}}) - \frac{1}{n}\sum_\ell \mathcal{H}(\mathbf{p} + \boldsymbol{\delta}^{(\ell)})$$

where $\mathcal{H}(\mathbf{p}) = -\langle \mathbf{p}, \log(\mathbf{p})\rangle$ is the entropy. Let $\mathbf{g}$ and H be respectively the gradient and Hessian of $\mathcal{H}$ at $\mathbf{p}$. Then by Taylor's theorem

$$Q = \left(\mathcal{H}(\mathbf{p}) + \langle \mathbf{g}, \bar{\boldsymbol{\delta}}\rangle + \frac{1}{2}\langle \bar{\boldsymbol{\delta}}, H\bar{\boldsymbol{\delta}}\rangle\right) - \left(\mathcal{H}(\mathbf{p}) + \frac{1}{n}\sum_\ell \langle \mathbf{g}, \boldsymbol{\delta}^{(\ell)}\rangle + \frac{1}{2n}\sum_\ell \langle \boldsymbol{\delta}^{(\ell)}, H\boldsymbol{\delta}^{(\ell)}\rangle\right) + O(\delta^3),$$

where $\delta \overset{def}{=} \max_\ell \|\boldsymbol{\delta}^{(\ell)}\|$. Thus canceling the linear terms

$$Q = \left(1 - \frac{1}{n}\right) \frac{1}{2n}\sum_\ell \langle \boldsymbol{\delta}^{(\ell)}, -H\boldsymbol{\delta}^{(\ell)}\rangle + \frac{1}{2n^2}\sum_{\ell \neq m} \langle \boldsymbol{\delta}^{(\ell)}, H\boldsymbol{\delta}^{(m)}\rangle + O(\delta^3).$$

By definition $\boldsymbol{\delta}^{(\ell)} = J(\mathbf{x}^{(\ell)}) + \boldsymbol{\Delta}^{(\ell)}$

$$\mathrm{E}[\sum_\ell \langle \boldsymbol{\delta}^{(\ell)}, H\boldsymbol{\delta}^{(\ell)}\rangle] = \mathrm{E}[\sum_\ell \langle J(\mathbf{x}^{(\ell)} - \boldsymbol{\mu}), HJ(\mathbf{x}^{(\ell)} - \boldsymbol{\mu})\rangle] + O(\varepsilon^3)$$

and

$$\mathrm{E}[\sum_{\ell \neq m} \langle \boldsymbol{\delta}^{(\ell)}, H\boldsymbol{\delta}^{(m)}\rangle] = \mathrm{E}[\sum_{\ell \neq m} \langle \boldsymbol{\Delta}^{(\ell)}, H\boldsymbol{\Delta}^{(m)}\rangle] = n^2 O(\varepsilon^4)$$

due to independence and $\mathrm{E}[\mathbf{x}^{(\ell)} - \boldsymbol{\mu}] = 0$. Finally,

$$\frac{1}{n}\sum_\ell \mathrm{E}[\langle J(\mathbf{x}^{(\ell)} - \boldsymbol{\mu}), HJ(\mathbf{x}^{(\ell)} - \boldsymbol{\mu})\rangle] = \mathrm{Tr}(J^T HJ\boldsymbol{\Sigma}).$$

Note that $J = \mathrm{diag}(\mathbf{p}) - \mathbf{p}\mathbf{p}^T$ and $H = \mathrm{diag}(1/\mathbf{p})$ by simple differentiation. Consequently $J^T(-H)J = \mathrm{diag}(\mathbf{p}) - \mathbf{p}\mathbf{p}^T$. Thus $\mathrm{Tr}((\mathrm{diag}(\mathbf{p}) - \mathbf{p}\mathbf{p}^T)\boldsymbol{\Sigma}) = \langle \mathbf{p}, \mathrm{diag}\,\boldsymbol{\Sigma}\rangle - \langle \mathbf{p}, \boldsymbol{\Sigma}\mathbf{p}\rangle$. Therefore

$$\mathrm{E}[Q] = (1 - \frac{1}{n})\frac{1}{2} \left(\langle \mathbf{p}, \mathrm{diag}\,\boldsymbol{\Sigma}\rangle - \langle \mathbf{p}, \boldsymbol{\Sigma}\mathbf{p}\rangle\right) + O(\varepsilon^3)$$

as desired. In general the quality of the estimate increases as n grows, so we let $n \to \infty$ and get rid of the factor $1 - 1/n$. This is similar to the classical Bessel's correction for covariance. $\square$

B Additional Experiments

B.1 Calibration Experiments

Methodology We trained LeNet with 10% dropout on the top two linear layers, on respectively MNIST, Fashion-MNIST and CIFAR 10. The models were trained for 1000 epochs each with Adam, learning rate $2e - 4$.

We ran a simple Frequentist inference, as well as using Monte Carlo samples and Favour, and compared the Accuracy as well as calibration scores. Each method was run with a number of different parameter settings. For sampling we used 1-1000 samples, and for Favour we used different values of variance rank, weight rank, type of ReLU propagation, and the number of subspace iterations to perform. Inference at each parameter setting was repeated 10 times to estimate the variance in caused by random sampling.

The pareto-optimal points on the quality / time curve were identified and plot in the figures below.

The ECE and Coverage @ 95% numbers for the sample based method is based on fitting a two dimensional Gaussian to the output point cloud.
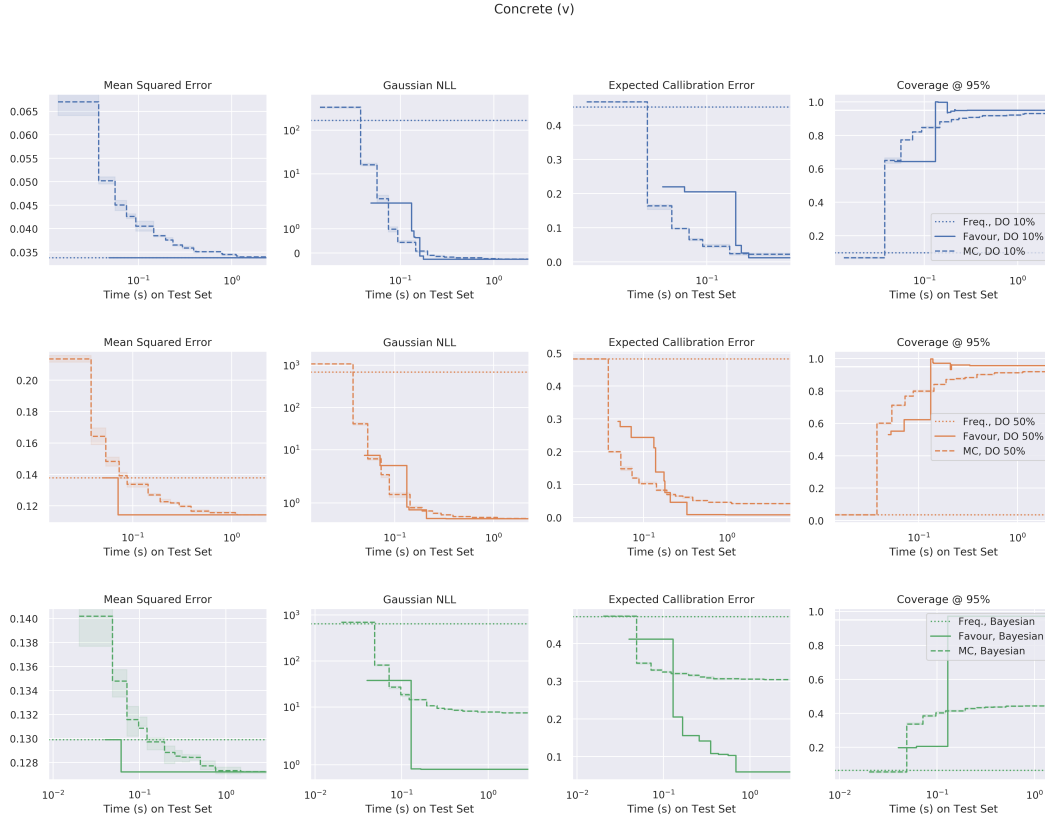
Figure 4: **Regression performance on UCI/Concrete.** We predict the strength of concrete based on 8 input variables. Favour matches roughly the quality of MCDO estimates, when the later is given 1000 samples on the dropout models. For the VI model however, it seems that many more samples are needed to obtain good calibration error.
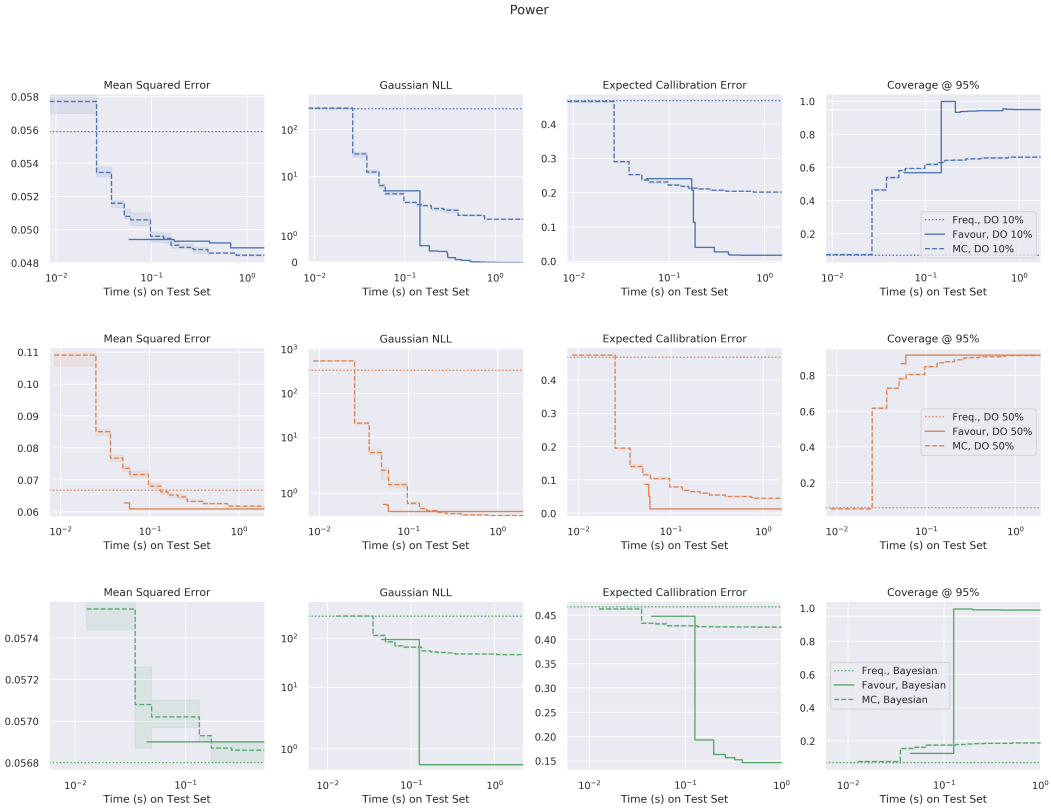
Figure 5: **Regression performance on UCI/Power.** The Combined Cycle Power Plant Data Set consist of hourly average ambient variables Temperature, Ambient Pressure, Relative Humidity and Exhaust Vacuum to predict the net hourly electrical energy output of a plant. In the case of the 50% dropout model, using a large amount of samples is able to outperform Favour on Gaussian NLL, but Favour uniformely achieves better ECE. Like with Concrete, Favour is able to much better utilize the VI model than sampling.

Energy (v)
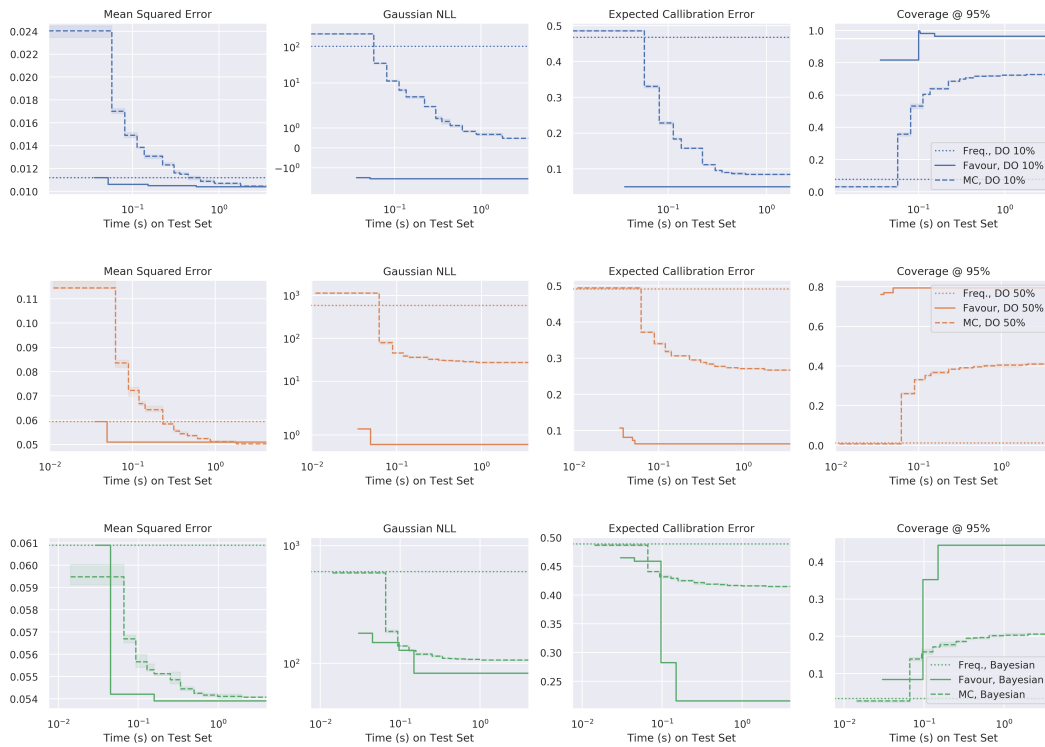
Figure 6: **Regression performance on UCI/Energy.** We perform energy analysis using 12 different building shapes simulated in Ecotect. The test-set is not very large, which make the calibration scores unreliable. Hence we report scores on a validation set instead. The findings are roughly the same as in the previous two experiments.
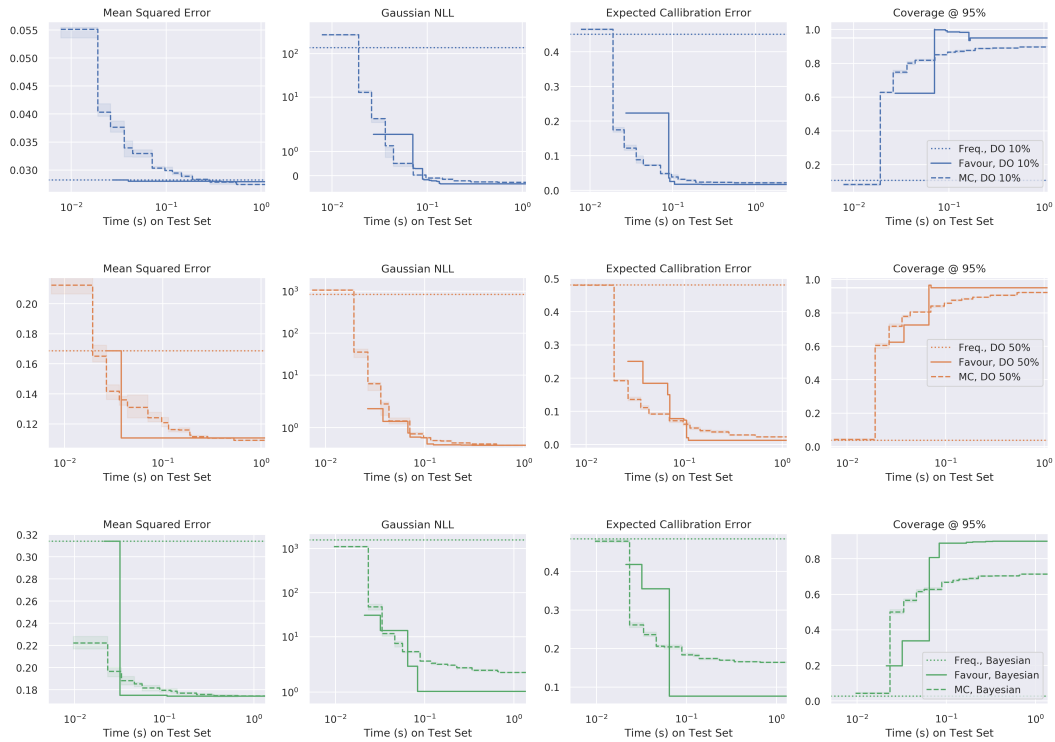
Figure 7: **Regression performance on UCI/Boston** Favour beats sample based methods at various calibration tasks, when given a medium computational budget. For the very short time-frames sampling performs better, but then we don't win much over simple frequentist inference. We again used a validation set which is somewhat larger than the small standard test-set, so we can get more precise calibration numbers.

Kin8nm
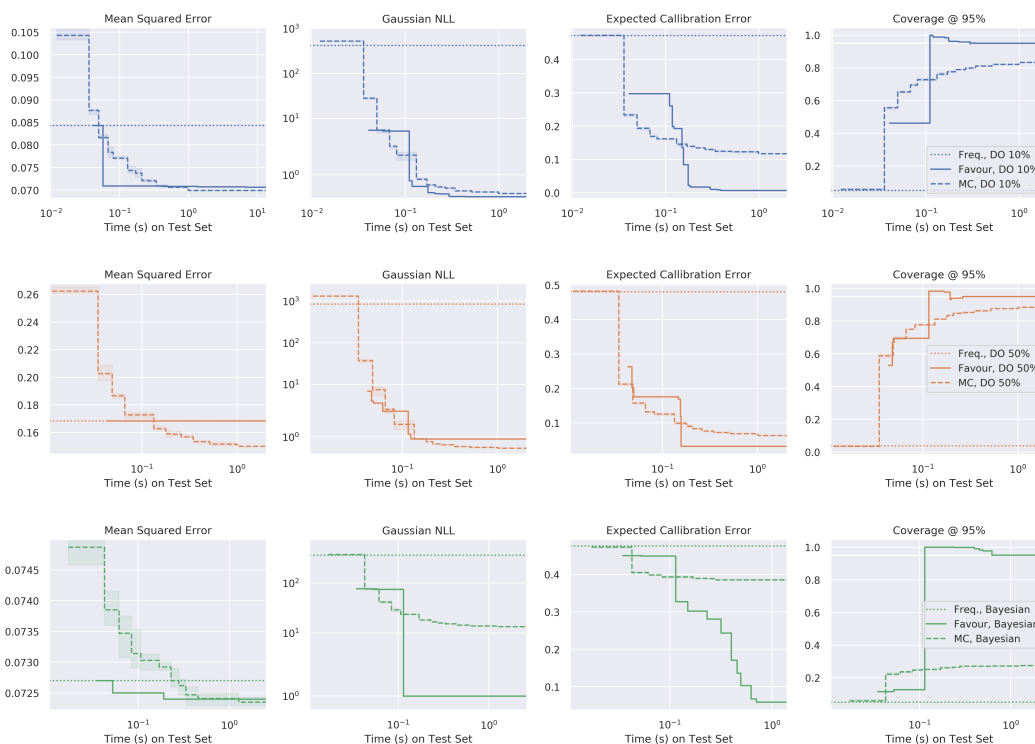
Figure 8: **Regression performance on UCI/Kin8nm.** This is data set is concerned with the forward kinematics of an 8 link robot arm. Among the existing variants of this data set we have used the variant 8nm, which is known to be highly non-linear and medium noisy. Favour does well on the Dropout 10% and Bayesian models, but not so well on the Dropout 50% model, which didn't learn the problem very well.
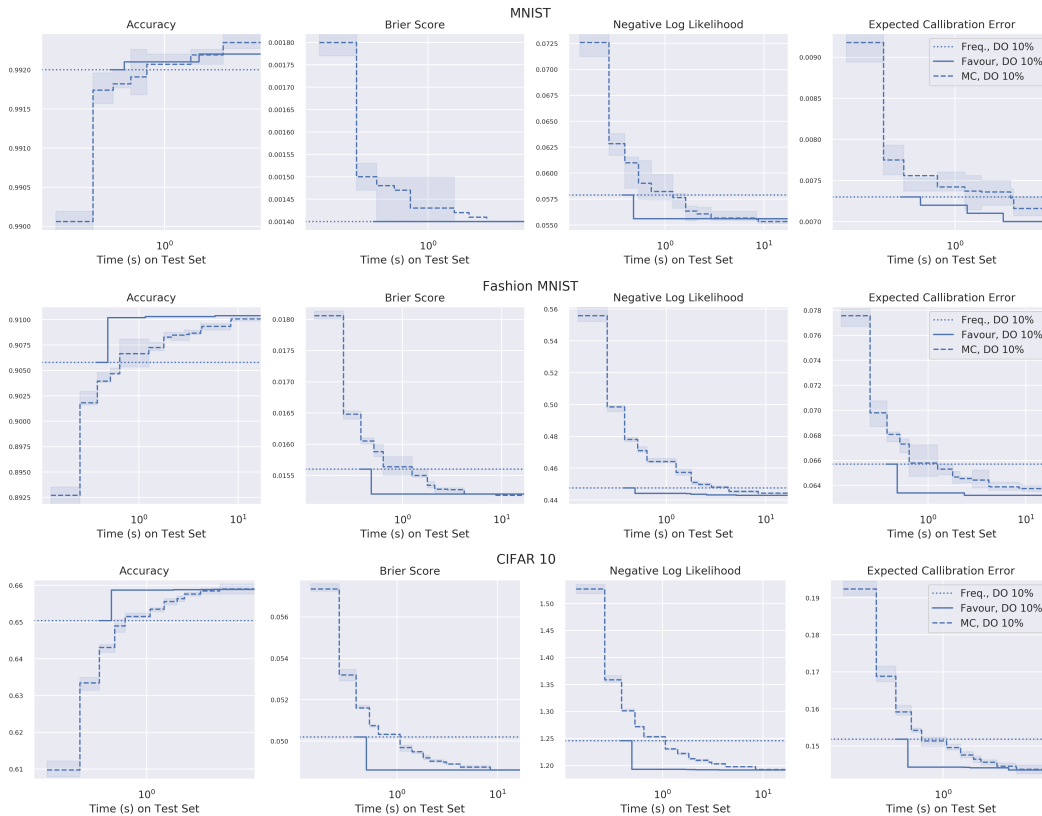
Figure 9: **Classification performance per time.** We trained LeNet with 10% dropout on the top two linear layers, on respectively MNIST, Fashion-MNIST and CIFAR 10. In every case using enough samples eventually gave better results than Favour, but Favour was able to get equivalent performance results using a time budget equivalent to just 2-3 MCDO samples.

B.2 OOD Experiments

Methodology   We trained LeNet with different levels of dropout (10%, 25% and 50%) as well as VI on the top two layers on repectively MNIST and CIFAR-10. The MNIST model was trained for 400 epochs and the CIFAR-10 for 250 epochs; each with Adam, learning rate $2e-4$.

We ran a simple Frequentist inference, as well as using Monte Carlo samples and Favour on the test sets for those datasets, as well as a number of OOD datasets (Fashion MNIST and SVHN).

We computed three deterministic scores (Maximum Probability, Entropy and Mahalanobis distance) as well as two scores that are only defined in the Bayesian setting (Jensen-Shannon Divergence and KL-Diveregence.) The scores were used in a simple in/out of distribution classifier and the performance was measured using Area under the curve, for ROC as well as Precision Recall

Datasets	Model	Inference	Area Under Receiver-Operating Characteristic (ROC) Curve				
			JSD	MaxP.	Ent.	Maha.	KL
Train dist: MNIST 0-7 Out of dist: MNIST 8-9	Bayes	Freq.	—	97.0 ± 0.0	97.1 ± 0.0	95.4 ± 0.0	—
		Favour	96.8 ± 0.0	97.1 ± 0.0	97.2 ± 0.0	96.0 ± 0.0	96.9 ± 0.0
		MC	95.8 ± 0.1	96.8 ± 0.0	97.0 ± 0.0	93.7 ± 0.1	88.1 ± 3.2
	DO 10%	Freq.	—	89.1 ± 0.0	97.1 ± 0.0	79.6 ± 0.0	—
		Favour	89.1 ± 0.0	89.0 ± 0.0	97.2 ± 0.0	80.5 ± 0.0	62.1 ± 0.0
		MC	95.8 ± 0.2	92.5 ± 0.3	96.7 ± 0.1	75.8 ± 0.3	73.7 ± 0.4
	DO 25%	Freq.	—	96.7 ± 0.0	96.9 ± 0.0	95.8 ± 0.0	—
		Favour	96.9 ± 0.0	96.8 ± 0.0	97.0 ± 0.0	95.9 ± 0.0	57.7 ± 0.0
		MC	95.7 ± 0.1	95.9 ± 0.1	96.0 ± 0.1	86.9 ± 0.2	79.1 ± 1.3
	DO 50%	Freq.	—	95.9 ± 0.0	96.0 ± 0.0	90.8 ± 0.0	—
		Favour	95.8 ± 0.0	96.0 ± 0.0	96.1 ± 0.0	90.3 ± 0.0	42.7 ± 0.0
		MC	91.5 ± 0.3	93.7 ± 0.2	93.8 ± 0.2	57.3 ± 0.4	81.6 ± 5.4
Train dist: MNIST Out of dist: Fashion MNIST	Bayes	Freq.	—	64.5 ± 0.0	64.2 ± 0.0	67.9 ± 0.0	—
		Favour	79.6 ± 0.0	63.6 ± 0.0	63.3 ± 0.0	77.2 ± 0.0	81.2 ± 0.0
		MC	78.5 ± 0.1	65.3 ± 0.1	65.0 ± 0.1	72.2 ± 0.1	49.3 ± 0.7
	DO 10%	Freq.	—	77.2 ± 0.0	77.0 ± 0.0	93.3 ± 0.0	—
		Favour	82.8 ± 0.0	77.3 ± 0.0	77.1 ± 0.0	93.9 ± 0.0	82.5 ± 0.0
		MC	85.1 ± 0.1	83.1 ± 0.1	82.1 ± 0.1	93.7 ± 0.1	88.2 ± 0.5
	DO 25%	Freq.	—	72.3 ± 0.0	72.0 ± 0.0	75.5 ± 0.0	—
		Favour	88.9 ± 0.0	72.0 ± 0.0	71.7 ± 0.0	78.2 ± 0.0	55.5 ± 0.0
		MC	90.9 ± 0.2	73.0 ± 0.1	72.0 ± 0.2	79.5 ± 0.1	75.3 ± 0.7
	DO 50%	Freq.	—	66.6 ± 0.0	65.7 ± 0.0	93.4 ± 0.0	—
		Favour	76.8 ± 0.0	67.7 ± 0.0	66.7 ± 0.0	94.5 ± 0.0	86.6 ± 0.0
		MC	83.2 ± 0.3	78.0 ± 0.3	73.7 ± 0.2	89.2 ± 0.1	79.4 ± 0.7
Train dist: CIFAR-10 Out of dist: SVHN	DO 10%	Freq.	—	79.9 ± 0.0	85.8 ± 0.0	35.4 ± 0.0	—
		Favour	65.2 ± 5.9	80.2 ± 0.1	86.0 ± 0.1	36.3 ± 0.1	82.6 ± 11.5
		MC	50.8 ± 0.2	77.8 ± 0.1	83.4 ± 0.1	37.5 ± 0.1	91.3 ± 0.1
	DO 25%	Freq.	—	88.1 ± 0.0	85.2 ± 0.0	82.5 ± 0.0	—
		Favour	84.6 ± 3.9	87.4 ± 0.1	84.6 ± 0.1	85.0 ± 0.3	64.5 ± 12.5
		MC	71.5 ± 0.2	85.3 ± 0.1	80.8 ± 0.1	83.0 ± 0.1	83.6 ± 0.1
	DO 50%	Freq.	—	80.4 ± 0.0	73.0 ± 0.0	91.5 ± 0.0	—
		Favour	78.9 ± 4.1	78.0 ± 0.4	71.4 ± 0.4	91.4 ± 0.2	53.6 ± 9.2
		MC	69.6 ± 0.2	78.9 ± 0.1	71.3 ± 0.1	87.4 ± 0.1	79.8 ± 0.2
	Bayes	Freq.	—	83.0 ± 0.0	84.0 ± 0.0	32.7 ± 0.0	—
		Favour	56.5 ± 0.0	82.9 ± 0.0	84.0 ± 0.0	32.9 ± 0.0	79.3 ± 0.0
		MC	58.4 ± 0.2	82.2 ± 0.0	83.4 ± 0.0	32.9 ± 0.0	75.1 ± 0.1

Table 2: **Performance of OOD with AUC ROC**

| | | | Area Under Precision-Recall Curve | | | | |
Datasets	Model	Inference	JSD	MaxP.	Ent.	Maha.	KL
Train dist: MNIST 0-7 Out of dist: MNIST-8-9	DO 10%	Freq.	—	41.5 ± 0.0	48.0 ± 0.0	99.6 ± 0.0	—
		Favour	28.5 ± 1.2	41.3 ± 0.1	47.9 ± 0.1	99.6 ± 0.0	97.1 ± 0.8
		MC	43.8 ± 0.8	41.8 ± 0.8	44.9 ± 0.9	99.4 ± 0.0	99.4 ± 0.0
	DO 25%	Freq.	—	52.9 ± 0.0	56.1 ± 0.0	99.7 ± 0.0	—
		Favour	46.0 ± 1.9	51.8 ± 0.7	55.7 ± 0.2	99.7 ± 0.0	94.9 ± 2.1
		MC	48.4 ± 0.9	48.9 ± 0.9	49.4 ± 0.9	99.3 ± 0.1	99.2 ± 0.1
	DO 50%	Freq.	—	64.5 ± 0.0	65.3 ± 0.0	99.7 ± 0.0	—
		Favour	61.5 ± 4.0	64.2 ± 1.1	65.3 ± 0.6	99.7 ± 0.0	71.8 ± 11.9
		MC	54.5 ± 1.1	56.1 ± 1.2	56.6 ± 1.0	97.1 ± 0.1	97.9 ± 0.3
	Bayes	Freq.	—	40.4 ± 0.0	40.6 ± 0.0	99.0 ± 0.0	—
		Favour	45.1 ± 0.0	42.6 ± 0.0	42.8 ± 0.0	98.9 ± 0.0	99.2 ± 0.0
		MC	43.7 ± 0.7	43.7 ± 0.5	43.4 ± 0.8	98.6 ± 0.1	91.6 ± 1.1
Train dist: MNIST Out of dist: Fashion MNIST	Bayes	Freq.	—	60.7 ± 0.0	61.1 ± 0.0	77.5 ± 0.0	—
		Favour	85.1 ± 0.0	56.7 ± 0.0	56.5 ± 0.0	82.7 ± 0.0	80.0 ± 0.0
		MC	83.0 ± 0.2	57.7 ± 0.2	57.8 ± 0.4	79.0 ± 0.1	64.9 ± 0.2
	DO 10%	Freq.	—	75.9 ± 0.0	75.7 ± 0.0	95.2 ± 0.0	—
		Favour	84.7 ± 0.0	75.9 ± 0.0	75.7 ± 0.0	95.5 ± 0.0	88.1 ± 0.0
		MC	87.1 ± 0.2	82.8 ± 0.3	80.3 ± 0.3	95.4 ± 0.1	91.7 ± 0.3
	DO 25%	Freq.	—	60.5 ± 0.0	59.9 ± 0.0	79.2 ± 0.0	—
		Favour	90.4 ± 0.0	59.9 ± 0.0	59.4 ± 0.0	81.6 ± 0.0	60.7 ± 0.0
		MC	92.4 ± 0.2	59.8 ± 0.1	58.9 ± 0.1	83.0 ± 0.1	79.6 ± 0.5
	DO 50%	Freq.	—	65.5 ± 0.0	62.9 ± 0.0	95.0 ± 0.0	—
		Favour	80.7 ± 0.0	65.9 ± 0.0	63.2 ± 0.0	95.8 ± 0.0	83.9 ± 0.0
		MC	86.6 ± 0.3	75.6 ± 0.4	66.9 ± 0.2	91.6 ± 0.1	84.7 ± 0.4
Train dist: CIFAR-10 Out of dist: SVHN	DO 10%	Freq.	—	89.1 ± 0.0	92.3 ± 0.0	60.5 ± 0.0	—
		Favour	74.5 ± 3.8	89.2 ± 0.0	92.0 ± 0.3	60.8 ± 0.0	85.4 ± 7.9
		MC	67.0 ± 0.1	87.3 ± 0.3	89.4 ± 0.1	61.2 ± 0.1	93.4 ± 0.1
	DO 25%	Freq.	—	91.2 ± 0.0	87.2 ± 0.0	87.0 ± 0.0	—
		Favour	88.6 ± 2.4	90.4 ± 0.1	86.3 ± 0.1	88.3 ± 0.2	76.8 ± 7.6
		MC	79.6 ± 0.2	88.8 ± 0.1	83.1 ± 0.1	87.0 ± 0.1	89.1 ± 0.2
	DO 50%	Freq.	—	85.5 ± 0.0	78.4 ± 0.0	93.8 ± 0.0	—
		Favour	83.6 ± 3.2	82.6 ± 0.4	76.9 ± 0.3	93.9 ± 0.1	70.0 ± 5.8
		MC	79.4 ± 0.1	84.2 ± 0.1	76.9 ± 0.1	91.4 ± 0.1	87.5 ± 0.2
	Bayes	Freq.	—	89.4 ± 0.0	89.1 ± 0.0	59.4 ± 0.0	—
		Favour	68.3 ± 0.0	89.4 ± 0.0	89.1 ± 0.0	59.5 ± 0.0	83.1 ± 0.0
		MC	69.7 ± 0.1	89.0 ± 0.2	88.7 ± 0.0	59.5 ± 0.0	81.8 ± 0.1

Table 3: **OOD performance using Precision Recall**