Dangers of Bayesian Model Averaging under Covariate Shift

Anonymous Author(s)
Affiliation
Address
email

Abstract

Approximate Bayesian inference for neural networks is considered a robust alternative to standard training, often providing good performance on out-of-distribution data. However, Bayesian neural networks (BNNs) with high-fidelity approximate inference via full-batch Hamiltonian Monte Carlo achieve poor generalization under covariate shift, even underperforming classical estimation. We explain this surprising result, showing how a Bayesian model average can in fact be problematic under covariate shift, particularly in cases where linear dependencies in the input features cause a lack of posterior contraction. We additionally show why the same issue does not affect many approximate inference procedures, or classical maximum a-posteriori (MAP) training. Finally, we propose novel priors that improve the robustness of BNNs to many sources of covariate shift.

1 Introduction

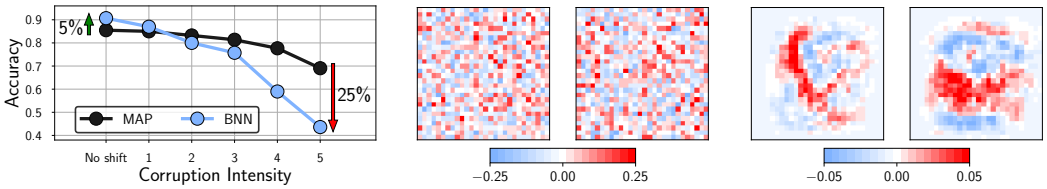
The predictive distributions of deep neural networks are often deployed in critical applications such as medical diagnosis [15, 11, 12], and autonomous driving [5, 1, 30]. These applications typically involve *covariate shift*, where the target data distribution is different from the distribution used for training [16, 2]. Accurately reflecting uncertainty is crucial for robustness to these shifts [36, 39]. Since Bayesian methods provide a principled approach to representing model (epistemic) uncertainty, they are commonly benchmarked on out-of-distribution (OOD) generalization tasks [e.g., 21, 36, 8, 10, 48].

However, Izmailov et al. [18] recently showed that Bayesian neural networks (BNNs) with high fidelity inference through Hamiltonian Monte Carlo (HMC) provide shockingly poor OOD generalization performance, despite the popularity and success of approximate Bayesian inference in this setting [14, 25, 36, 29, 48, 10, 3].

In this paper, we seek to understand, further demonstrate, and help remedy this concerning behaviour. We show that Bayesian neural networks perform poorly for different types of covariate shift, namely test data corruption, domain shift, and spurious correlations. In Figure 1 (a) we see that a ResNet-20 BNN approximated with HMC underperforms a maximum a-posteriori (MAP) solution by 25% on the *pixelate*-corrupted CIFAR-10 test set. This result is particularly surprising given that on the in-distribution test data, the BNN outperforms the MAP solution by over 5%.

Intuitively, we find that Bayesian model averaging (BMA) can be problematic under covariate shift as follows. Due to dependencies in the features of the train data distribution, some model parameters do not affect the predictions on the train data. For these parameters, the posterior coincides with the prior. The MAP solution sets the values of these parameters to zero, due to regularization from the prior, while the BMA samples these weights from the prior. At test time, the model is applied to a different data distribution, where the features do not have the same dependence, and the parameters that did not affect the predictions on train can negatively affect predictions on test.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.



(a) ResNet-20, CIFAR-10-C (b) BNN weights (c) MAP weights

Figure 1: **Bayesian neural networks under covariate shift**. **(a)**: Performance of a ResNet-20 on the *pixelate* corruption in CIFAR-10-C. For the highest degree of corruption, a Bayesian model average underperforms a MAP solution by 25% (44% against 69%) accuracy. See Izmailov et al. [18] for details. **(b)**: Visualization of the weights in the first layer of a Bayesian fully-connected network on MNIST sampled via HMC. **(c)**: The corresponding MAP weights.

In Figure 1 (b, c) we visualize the weights in the first layer of a fully-connected network for a sample from the BNN posterior and the MAP solution on the MNIST dataset. The MAP solution weights are highly structured, while the BNN sample appears extremely noisy, similar to a draw from the Gaussian prior. In particular the weights corresponding to *dead pixels* (i.e. pixel positions that are black for all the MNIST images) near the boundary of the input image are set near zero (shown in white) by the MAP solution, but sampled randomly by the BNN. If at test time the data is corrupted, e.g. by Gaussian noise, and the pixels near the boundary of the image are activated, the MAP solution will ignore these pixels, while the predictions of the BNN will be significantly affected.

We further demonstrate and analyze these results, showing that the surprising lack of robustness under covariate shift for Bayesian neural networks is fundamentally caused by linear dependencies in parameters. Based on our understanding, we introduce a novel prior that assigns a low variance to the weights in the first layer corresponding to directions orthogonal to the data manifold, leading to improved generalization under covariate shift. We additionally study the effect of non-zero mean corruptions and accordingly propose a second prior that constrains the sum of the weights, resulting in further improvements in OOD generalization for Bayesian neural networks.

2 Background

Bayesian neural networks. A Bayesian neural network model is specified by the prior distribution $p(w)$ over the weights w of the model, and the likelihood function $p(y|x, w)$, where x represents the input features and y represents the target value. Following Bayes' rule, the *posterior* distribution over the parameters w after observing the dataset $\mathcal{D} = \{(x_i, y_i)|i = 1, \ldots, n\}$ is given by

$$p(w|\mathcal{D}) = \frac{p(D|w) \cdot p(w)}{\int_{w'} p(D|w') \cdot p(w')dw'} = \frac{\prod_{i=1}^{n} p(y_i|x_i, w) \cdot p(w)}{\int_{w'} \prod_{i=1}^{n} p(y_i|x_i, w') \cdot p(w')dw'}, \tag{1}$$

where we assume that the likelihood is independent over the data points. The posterior in Equation 1 is then used to make predictions for a new input x according to the *Bayesian model average*:

$$p(y|x) = \int p(y|x, w) \cdot p(w|\mathcal{D})dw. \tag{2}$$

Unfortunately, computing the BMA in Equation 2 is intractable for Bayesian neural networks. Hence, a number of approximate inference methods have been developed. In this work, we use full-batch HMC [33, 18], as it provides a high-accuracy posterior approximation. For a more detailed discussion of Bayesian deep learning see Wilson and Izmailov [48].

Maximum a-posteriori (MAP) estimation. In contrast with Bayesian model averaging, a MAP estimator uses the single setting of weights (hypothesis) that maximizes the posterior density $w_{\text{MAP}} = \underset{w}{\operatorname{argmax}}\, p(w|\mathcal{D}) = \underset{w}{\operatorname{argmax}}(\log p(\mathcal{D}|w) + \log p(w))$, where the log prior can be viewed as a regularizer. MAP is the standard approach to training neural networks and our baseline for *classical training* throughout the paper. We perform MAP estimation with SGD unless otherwise specified.

Covariate shift. In this paper, we focus on the covariate shift setting. We assume the training dataset $\mathcal{D}_{\text{train}}$ consists of i.i.d. samples from the distribution $p_{\text{train}}(x, y) = p_{\text{train}}(x) \cdot p(y|x)$. However,

the test data may come from a different distribution $p_{\text{test}}(x, y) = p_{\text{test}}(x) \cdot p(y|x)$. We assume the conditional distribution $p(y|x)$ remains unchanged, but the marginal distribution of the input features $p_{\text{test}}(x)$ differs from $p_{\text{train}}(x)$. Arjovsky [2] provides a detailed discussion of covariate shift.

3 Related work

Several works have studied the lack of robustness of standard training procedures to covariate shift and proposed solutions to their failure. Shimodaira [40] propose to re-weight the train data points according to their density in the test data distribution. Sugiyama et al. [43, 44] adapt the importance-weighting approach to the cross-validation setting. Storkey and Sugiyama [42] and Storkey [41] propose probabilistic formulations for linear regression and Gaussian processes under covariate shift, assuming the data comes from multiple sources. More recently, Chan et al. [7] introduce an approximate inference scheme for BNNs that uses test data covariates to regularize the posterior. In general, covariate shift is a large area of study, discussed in detail in Arjovsky [2].

Moreover, Neyshabur et al. [35] show that models relying on features that encode semantic structure in the data are more robust to covariate shift. Nagarajan et al. [32] also provide insights into how classically trained max-margin classifiers can fail under covariate shift due to a reliance on spurious correlations between class labels and input features.

BNN robustness to adversarial attacks is a related area of study, but generally involves much smaller perturbations to the test covariates than covariate shift. Carbone et al. [6] prove that BNNs are robust to gradient-based adversarial attacks in the large data, overparameterized limit, while Wicker et al. [47] present a framework for training BNNs with guaranteed robustness to adversarial examples.

To the best of our knowledge, no previous work has attempted to understand or remedy the surprising failure of Bayesian neural networks under covariate shift recently discovered by Izmailov et al. [18].

4 Bayesian neural networks are not robust to covariate shift

In this section, we evaluate Bayesian neural networks under different types of covariate shift. Specifically, we focus on two types of covariate shift: test data corruption and domain shift. In Appendix C, we additionally evaluate BNNs in the presence of spurious correlations in the data.

Methods. We evaluate BNNs against two deterministic baselines: a MAP solution approximated with stochastic gradient descent (SGD) with momentum [38, 37] and a deep ensemble of 10 independently trained MAP solutions [25]. For BNNs, we provide the results using a Gaussian prior and a more heavy-tailed Laplace prior following Fortuin et al. [13]. Izmailov et al. [18] conjectured that *cold posteriors* [46] can improve the robustness of BNNs under covariate shift; to test this hypothesis, we provide results for BNNs with a Gaussian prior and cold posteriors at temperature 10^{-2}. For all BNN models, we run a single chain of HMC for 100 iterations discarding the first 10 iterations as burn-in, following Izmailov et al. [18]. We provide additional experimental details in Appendix A.

Datasets and data augmentation. We run the methods on the MNIST [27] and CIFAR-10 [24] datasets. Following Izmailov et al. [18] we do not use data augmentation with any of the methods, as it is not trivially compatible with the Bayesian neural network framework [see 18, 46, for a detailed discussion of this point].

Neural network architectures. On both the CIFAR-10 and MNIST datasets we use a small convolutional network (CNN) inspired by LeNet-5 [26], with 2 convolutional layers followed by 3 fully-connected layers. On MNIST we additionally consider a fully-connected neural network (MLP) with 2 hidden layers of 256 neurons each. We note that high-fidelity posterior sampling with HMC is extremely computationally intensive. Even on the small architectures that we consider, the experiments take multiple hours on 8 NVIDIA Tesla V-100 GPUs or 8-core TPU-V3 devices [20]. See Izmailov et al. [18] for details on the computational requirements of full-batch HMC for BNNs.

4.1 Test data corruption

We start by considering the scenario where the test data is corrupted by some type of noise. In this case, there is no semantic distribution shift: the test data is collected in the same way as the train data, but then corrupted by a generic transformation.

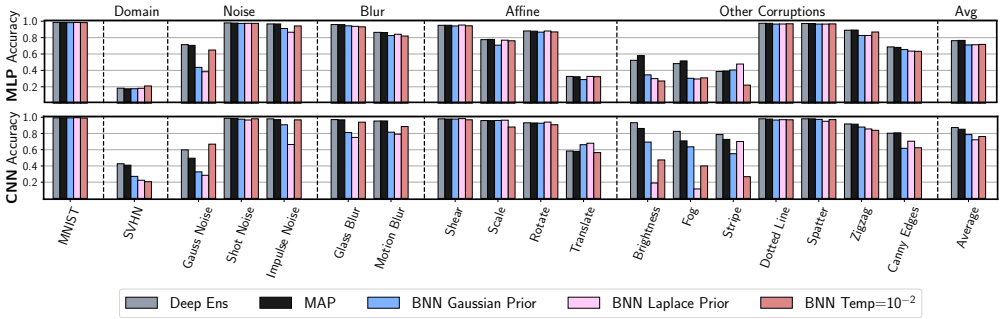


Figure 2: **Robustness on MNIST.** Accuracy for deep ensembles, MAP and Bayesian neural networks trained on MNIST under covariate shift. **Top**: Fully-connected network; **bottom**: Convolutional neural network. While on the original MNIST test set BNNs provide competitive performance, they underperform deep ensembles on most of the corruptions. With the CNN architecture, all BNN variants lose to MAP when evaluated on SVHN by almost 20%.

In Figure 2, we report the performance of the methods trained on MNIST and evaluated on the MNIST-corrupted (MNIST-C) test sets [31] under various corruptions[1]. We report the results for a fully-connected network and a convolutional network.

With the CNN architecture, deep ensembles consistently outperform BNNs. Moreover, even a single MAP solution significantly outperforms BNNs on many of the corruptions. The results are especially striking on *brightness* and *fog* corruptions, where the BNN with Laplace prior shows accuracy at the level of random guessing, while the deep ensemble retains accuracy above 90%. *Gaussian noise* and *impulse noise* corruptions also present significant challenges for BNNs. The BNNs show the most competitive performance in-distribution and on the corruptions representing affine transformations: *shear*, *scale*, *rotate* and *translate*. The results are generally analogous for the MLP: across the board the BNNs underperform deep ensembles, and often even a single MAP solution.

While the cold posteriors provide an improvement on some of the *noise* corruptions, they also hurt performance on *Brightness* and *Stripe*, and do not improve the performance significantly on average across all corruptions compared to a standard BNN with a Gaussian prior. We provide additional results on cold posterior performance in Appendix D.

Next, we consider the CIFAR-10-corrupted dataset (CIFAR-10-C) [16]. CIFAR-10-C consists of 18 transformations that are available at different levels of intensity ($1-5$). We report the results using corruption intensity 4 (results for other intensities are in Appendix B) for each of the transformations in Figure 3. Similarly to MNIST-C, BNNs outperform deep ensembles and MAP on in-distribution data, but underperform each over multiple corruptions. On CIFAR-10-C, BNNs are especially vulnerable to different types of *noise* (*Gaussian noise*, *shot noise*, *impulse noise*, *speckle noise*). For each of the *noise* corruptions, BNNs underperform even the classical MAP solution. The cold posteriors improve the performance on the *noise* corruptions, but only provide a marginal improvement across the board.

We note that Izmailov et al. [18] evaluated a ResNet-20 model on the same set of CIFAR-10-C corruptions. While they use a much larger architecture, the qualitative results for both architectures are similar: BNNs are the most vulnerable to *noise* and *blur* corruptions. We thus expect that our paper's analysis is not specific to smaller architectures and will equally apply to deeper models.

4.2 Domain shift

Next, we consider a different type of covariate shift where the test data and train data come from different, but semantically related datasets.

First, we apply our CNN and MLP MNIST models to the SVHN test set [34]. The MNIST-to-SVHN domain shift task is a common benchmark for unsupervised domain adaptation: both datasets contain images of digits, although MNIST contains hand-written digits while SVHN represents house

[1] In addition to the corruptions from MNIST-C, we consider Gaussian noise with standard deviation 3, as this corruption was considered by Izmailov et al. [18].

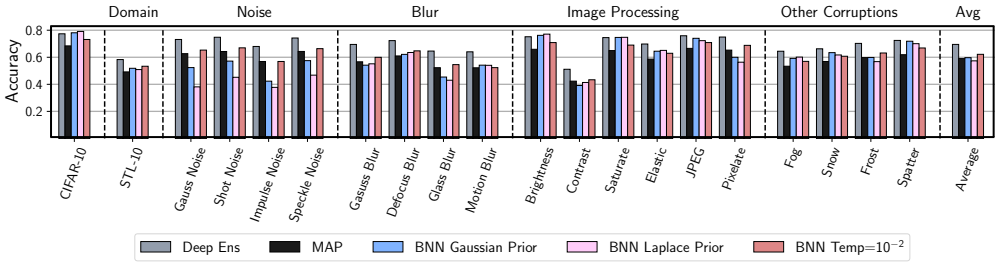


Figure 3: **Robustness on CIFAR-10.** Accuracy for deep ensembles, MAP and Bayesian neural networks using a CNN architecture trained on CIFAR-10 under covariate shift. For the corruptions from CIFAR-10-C, we report results for corruption intensity 4. While the BNNs with both Laplace and Gaussian priors outperform deep ensembles on the in-distribution accuracy, they underperform even a single MAP solution on most corruptions.

numbers. In order to apply our MNIST models to SVHN, we crop the SVHN images and convert them to grayscale. We report the results in Figure 2. While for MLPs all methods perform similarly, with the CNN architecture BNNs underperform deep ensembles and MAP by nearly 20%.

Next, we apply our CIFAR-10 CNN model to the STL-10 dataset [9]. Both datasets contain natural images with 9 shared classes between the two datasets[2]. We report the accuracy of the CIFAR-10 models on these 9 shared classes in STL-10 in Figure 3. While BNNs outperform the MAP solution, they still significantly underperform deep ensembles.

The results presented in this section highlight the generality and practical importance of the lack of robustness in BNNs: despite showing strong performance in-distribution, BNNs underperform even a single MAP solution (classical training) over an extensive variety of covariate shifts.

5 Understanding Bayesian neural networks under covariate shift

Now that we have established that Bayesian neural networks are highly susceptible to many types of covariate shift, we seek to understand why this is the case. In this section, we identify the linear dependencies in the input features as one of the key issues undermining the robustness of BNNs. We emphasize that *linear* dependencies in particular are *not* simply chosen for the simplicity of analysis, and their key role follows from the structure of the fully-connected and convolutional layers.

5.1 Motivating example: dead pixels and fully-connected layers

To provide an intuition for the results presented in this section, we start with a simple but practically relevant motivating example. Suppose we use a fully-connected Bayesian neural network on a dataset $D = \{(x_i, y_i)\}_{i=1}^n$, with m input features, where $x_i \in \mathbb{R}^m$. We use the upper indices to denote the features $x^1, \ldots, x^m$ and the lower indices to denote the datapoints x_i. Then, for each neuron j in the first hidden layer of the network, the activation can be written as $z_j = \phi(\sum_{i=1}^n x^i w_{ij}^1 + b_j^1)$, where w_{ij}^1 is the weight of the first layer of the network corresponding to the input feature i and hidden neuron j, and b_j^1 is the corresponding bias. We can prove the following lemma:

Lemma 1 *Using the notation introduced above, suppose that the input feature x_k^i is equal to zero for all the examples x_k in the training dataset D. Suppose the prior distribution over the parameters $p(W)$ factorizes as $p(W) = p(w_{ij}^1) \cdot p(W \setminus w_{ij}^1)$ for some neuron j in the first layer, where $W \setminus w_{ij}^1$ represents all the parameters W of the network except w_{ij}^1. Then, the posterior distribution $p(W|D)$ will also factorize and the marginal posterior over the parameter w_{ij}^1 will coincide with the prior:*

$$p(W|D) = p(W \setminus w_{ij}^1|D) \cdot p(w_{ij}^1). \tag{3}$$

Consequently, the MAP solution will set the weight w_{ij}^1 to the value with maximum prior density.

[2]CIFAR-10 has a class *frog* and STL-10 has *monkey*. The other nine classes coincide.

Intuitively, Lemma 1 says that if the prior for some parameter in the network is independent of the other parameters, and the value of the parameter does not affect the predictions of the model on any of the training data, then the posterior of this parameter will coincide with its prior. In particular, if one of the input features is always zero, then the corresponding weights will always be multiplied by zero, and will not affect the predictions of the network. To prove Lemma 1, we simply note that the posterior is proportional to the product of prior and likelihood, and both terms factorize with respect to w^1_{ij}. We present a formal proof in Appendix E.

So, for any sample from the true posterior, the weight w^1_{ij} will be a random draw from the prior distribution. Now suppose at test time we evaluate the model on data where the feature x^i is no longer zero. Then for these new inputs, the model will be effectively multiplying the input feature x^i by a random weight w^1_{ij}, leading to instability in predictions. Formally, we prove the following proposition (proof in Appendix E):

Proposition 1 *Suppose the assumptions of Lemma 1 hold. Assume also that the prior distribution $p(w^1_{ij})$ has maximum density at 0 and that the network uses RELU activations. Then for any test input $\bar{x}$, the expected prediction under Bayesian model averaging (Equation 2) will depend on the value of the feature $\bar{x}^i$, while the MAP solution will ignore this feature.*

For example, in the MNIST dataset there is a large number of *dead pixels*: pixels near the boundaries of the image that have intensity zero for all the inputs. In practice, we often use independent zero-mean priors (e.g. Gaussian) for each parameter of the network. So, according to Lemma 1, the posterior over all the weights in the first layer of the network corresponding to the dead pixels will coincide with the prior. If at test time the data is corrupted by e.g., Gaussian noise, the dead pixels will receive non-zero intensities, leading to a significant degradation in the performance of the Bayesian model average compared to the MAP solution.

While the situation where input features are constant and equal to zero may be rare and easily addressed, the results presented in this section can be generalized to *any* linear dependence in the data. We will now present our results in the most general form.

5.2 General linear dependencies and fully-connected layers

We now present our general results for fully-connected Bayesian neural networks when the features are linearly dependent. Intuitively, if there exists a direction in the input space such that all of the training data points have a constant projection on this direction (i.e. the data lies in a hyper-plane), then there will also exists a direction in the parameter space where the posterior coincides with the prior. Hence, the BMA predictions are highly susceptible to perturbations that move the test inputs in a direction orthogonal to the hyper-plane. The MAP solution on the other hand is completely robust to such perturbations. In Appendix E we prove the following proposition.

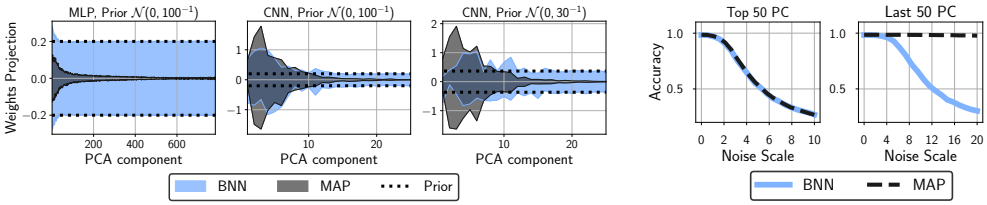
Proposition 2 *Suppose that the prior over the weights w^1_{ij} and biases b^1_j in the first layer is an i.i.d. Gaussian distribution $\mathcal{N}(0, \alpha^2)$, independent of the other parameters in the model. Suppose all the inputs $x_1 \dots x_n$ in the training dataset D lie in an affine subspace of the input space: $\sum_{j=1}^m x^j_i c_j = c_0$ for all $i = 1, \dots, n$ and some constants c such that $\sum_{i=0}^m c_i^2 = 1$. Then,*

(1) For any neuron j in the first hidden layer, the posterior distribution of random variable $w^c_j = \sum_{i=1}^m c_i w^1_{ij} - c_0 b^1_j$ (the projection of parameter vector $(w^1_{1j}, \dots, w^1_{mj}, b^1_j)$ on direction $(c_1, \dots, c_m, -c_0)$) will coincide with the prior $\mathcal{N}(0, \alpha^2)$.

(2) The MAP solution will set w^c_j to zero.

(3) Assuming the network uses ReLU activations, at test time, the BMA prediction will be susceptible to the inputs $\bar{x}$ that lie outside of the subspace, i.e. the predictive mean will depend on $\sum_{j=1}^m \bar{x}^j c_j - c_0$. The MAP prediction will not depend on this difference.

Empirical support. To test Propositions 2, we examine the performance of a fully-connected BNN on MNIST. The MNIST training dataset is not full rank, meaning that it has linearly dependent features. For a fully-connected BNN, Proposition 2 predicts that the posterior distribution of the first layer weights projected onto directions corresponding to these linearly dependent features will



(a) Weight projections

(b) Robustness along PCA

Figure 4: **Emprirical support for Propositions 2, 3. (a)**: The distribution (mean $\pm$ 2 std) of projections of the weights of the first layer on the directions corresponding to the PCA components of the data for BNN samples and MAP solution using MLP and CNN architectures with different prior scales. In each case, MAP sets the weights along low-variance components to zero, while BNN samples them from the prior. **(b)**: Accuracy of BNN and MAP solutions on the MNIST test set with Gaussian noise applied along the 50 highest and 50 lowest variance PCA components of the train data (left and right respectively). MAP is very robust to noise along low-variance PCA directions, while BMA is not; the two methods are similarly robust along the highest-variance PCA components.

coincide with the prior. In Figure 4 (a) we test this hypothesis by projecting first layer weights onto the principal components of the data. As expected, the distribution of the projections on low-variance PCA components (directions that are constant or nearly constant in the data) almost exactly coincides with the prior. The MAP solution, on the other hand, sets the weights along these PCA components close to zero, confirming conclusion (2) of the proposition. Finally, in Figure 4 (b) we visualize the performance of the BMA and MAP solution as we apply noise along high-variance and low-variance directions in the data. As predicted by conclusion (3) of Proposition 2, the MAP solution is very robust to noise along the low-variance directions, while BMA is not.

5.3 Linear dependencies and convolutional layers

Finally, we can extend Proposition 2 to convolutional layers. Unlike fully-connected layers, convolutional layers include weight sharing such that no individual weight corresponds to a specific pixel in the input images. For example, dead pixels will not necessarily present an issue for convolutional layers, unlike what is described in subsection 5.1. However, convolutional layers are still susceptible to a special type of linear dependence. Intuitively, we can think of the outputs of the first convolutional layer on all the input images as the outputs of a fully-connected layer applied to all the $K \times K$ patches of the input image. Therefore the reasoning in Proposition 2 applies to the convolutional layers, with the difference that the linear dependencies in the $K \times K$ patches cause the instability rather than dependencies in the full feature space. In Appendix E we prove the following proposition.

Proposition 3 *Suppose that the prior over the parameters of the convolutional filters and biases in the first layer is an i.i.d. Gaussian distribution $\mathcal{N}(0, \alpha^2)$, independent of the other parameters in the model. Suppose that the convolutional filters in the first layer are of size $K \times K \times C$, where C is the number of input channels. Then, consider the set $\hat{D}$ of size N of all the patches of size $K \times K \times C$ extracted from the training images in D after applying the same padding as in the first convolutional layer. Suppose all the patches $z_1 \dots z_N$ in the dataset $\hat{D}$ lie in an affine subspace of the space $\mathbb{R}^{K \times K \times C}$: $\sum_{c=1}^{C} \sum_{a=1}^{K} \sum_{b=1}^{K} z_i^{a,b} \gamma_{c,a,b} = \gamma_0$ for all $i = 1, \dots, N$ and some constants c_i such that $\sum_{c=1}^{C} \sum_{a=1}^{K} \sum_{b=1}^{K} \gamma_{c,a,b}^2 + \gamma_0^2 = 1$. Then, we can prove results analogous to (1)-(3) in Proposition 2 (see the Appendix E for the details).*

Empirical support. In Figure 4 (a), we visualize the projections of the weights in the first layer of the CNN architecture on the PCA components of the $K \times K$ patches extracted from MNIST. Analogously to fully-connected networks, the projections of the MAP weights are close to zero for low-variance components, while the projections of the BNN samples follow the prior.

What corruptions will hurt performance? Based on Propositions 2, 3, we expect that the corruptions that are the most likely to break linear dependence structure in the data will hurt the performance the most. In Appendix F we argue that *noise* corruptions are likely to break linear dependence, while the *affine* corruptions are more likely to preserve it, agreeing with our observations in section 4.

5.4 Why do some approximate Bayesian inference methods work well under covariate shift?

Unlike BNNs with HMC inference, some approximate inference methods such as SWAG [29], MC dropout [14], and deep ensembles [25] provide strong performance under covariate shift [36, 18]. For deep ensembles, we can easily understand why: a deep ensemble represents an average of approximate MAP solutions, and we have seen in conclusion (3) of Proposition 2 that MAP is robust to covariate shift in the scenario introduced in subsection 5.2. Similarly, other methods are closely connected to MAP via characterizing the posterior using MAP optimization iterates (SWAG), or modifying the training procedure for the MAP solution (MC Dropout). We provide further details in Appendix G.

6 Towards more robust Bayesian model averaging under covariate shift

In this section, we propose a new simple prior inspired by our theoretical analysis. In section 5 we showed that linear dependencies in the input features cause the posterior to coincide with the prior along the corresponding directions in the parameter space. In order to address this issue, we explicitly design the prior for the first layer of the network so that the prior variance is low along these directions.

6.1 Data empirical covariance prior

Let us consider the empirical covariance matrix of the inputs x_i. Assuming the input features are all preprocessed to be zero-mean $\sum_{i=1}^{n} x_i = 0$, we have $\Sigma = \frac{1}{n-1} \sum_{i=1}^{n} x_i x_i^T$. For fully-connected networks, we propose to use the *EmpCov* prior $p(w^1) = \mathcal{N}(0, \alpha\Sigma + \epsilon I)$ on the weights w^1 of the first layer of the network, where ϵ is a small positive constant that is used to ensure that the covariance matrix is positive definite. The parameter $\alpha > 0$ determines the scale of the prior.

Suppose there is a linear dependence in the input features of the data: $x_i^T p = c$ for some direction p and constant c. Then p will be an eigenvector of the empirical covariance matrix with the corresponding eigenvalue equal to 0 : $\Sigma p = \frac{1}{n-1} \sum_{i=1}^{n} x_i x_i^T p = \frac{c}{n-1} \sum_{i=1}^{n} x_i = 0$. Hence the prior over w^1 will have a variance of ϵ along the direction p.

More generally, the *EmpCov* prior is aligned with the principal components of the data, which are the eigenvectors of the matrix Σ. The prior variance along each principal component p_i is equal to $\alpha\sigma_i^2 + \epsilon$ where σ_i^2 is its corresponding explained variance. In Appendix H, we discuss a more general family of priors aligned with the principal components of the data.

Generalization to convolutions. We can generalize the *EmpCov* prior to convolutions by replacing the empirical covariance of the data with the empirical covariance of the patches that interact with the convolutional filter, denoted by $\hat{D}$ in Proposition 3.

Is EmpCov a valid prior? *EmpCov* falls under the umbrella of *data-dependent priors*: in order to construct the prior we need to evaluate the empirical covariance matrix of the inputs. However, we do not use the train data labels y_i to define the prior, unlike the approach known as the *Empirical Bayes* [see e.g. 4, section 3.5].

6.2 Experiments

In Figure 5 we report the performance of the BNNs using the *EmpCov* prior. In each case, we apply the *EmpCov* prior to the first layer, and a Gaussian prior to all other layers. For more details, please see Appendix A.

On both MLP on MNIST and CNN on CIFAR-10, the *EmpCov* prior significantly improves performance across the board. In both cases, the BNN with *EmpCov* prior shows competitive performance with deep ensembles, especially in the domain shift experiments. *EmpCov* is also particularly useful on the *noise* corruptions.

In Appendix I, we provide a detailed analysis of the performance of the CNN architecture on MNIST. Surprisingly, we found that using the *EmpCov* prior by itself does not provide a large improvement in this case. In Appendix I, we identify an issue specific to this particular setting, and propose another targeted prior that substantially improves performance.

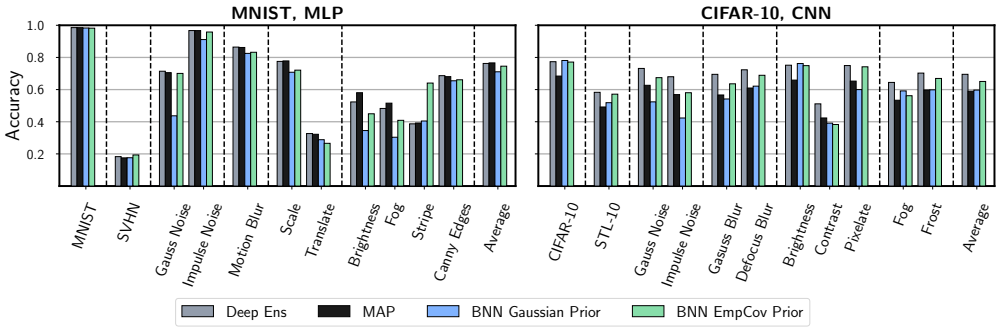


Figure 5: **EmpCov prior improves robustness.** Test accuracy under covariate shift for deep ensembles, MAP solution approximated by SGD, and BNN with Gaussian and *EmpCov* priors. **Left**: MLP architecture trained on MNIST. **Right**: CNN architecture trained on CIFAR-10. The *EmpCov* prior provides consistent improvement over the standard Gaussian prior. The improvement is particularly noticeable on the *noise* corruptions and in domail shift experiments (SVHN, STL-10).

7 Discussion

We consider the generality of our results, additional perspectives, and future directions.

Why focus on linear dependencies? This focus is dictated by the structure of both fully-connected and convolutional neural networks, which apply activation functions to linear combinations of features. Our results can be directly extended to other models, where different types of dependencies between input features would lead to the same lack of robustness to covariate shift. For example, in Appendix J we derive analogous results to subsection 5.2 for multiplicative neural network architectures [45], where a different form of dependence between features causes poor robustness.

Intermediate layers. While our analysis in section 5 is focused on the linear dependencies in the input features, similar conclusions can be made about intermediate layers of the network. For example, weights connected to *dead neurons*, which output zero, do not affect predictions of the model. We provide more details in Appendix K.

Linear Bayesian models. In models that are linear in parameters, such as Bayesian linear regression and Gaussian processes, we are typically saved from the perils of BMA under covariate shift discussed in section 5, because the MAP and the predictive mean under BMA coincide. We provide further details in Appendix L.

An optimization perspective on the covariate shift problem. In Section 5.1, we argued that SGD is more robust to covariate shift than HMC because the regularizer pushes the weights that correspond to dead pixels towards zero. This effect is mainly obtained through explicit regularization, without which these weights will remain at their initial values. We study the effect of initialization and explicit regularization on the performance of SGD under covariate shift in Appendix M.1. We also show in Appendix M.2 that other stochastic optimizers such as Adam [23] and Adadelta [49] behave similarly to SGD under covariate shift. Finally, we discuss the effect of test data corruptions on the loss landscape in Appendix M.3 and argue that the relative sharpness of low density posterior samples makes these solutions, which are included in a BMA but not MAP, more vulnerable to covariate shift.

Limitations. Due to the intense computational requirements of HMC, our experiments are limited to smaller models and datasets. Our analysis is focused on issues arising in models that are non-linear in their parameters. Moreover, while our proposed priors help improve robustness, they do not entirely resolve the issue. For example, in the *Contrast* dataset of Figure 5 (right panel), the BMA is still underperforming MAP.

Conclusion. Our work has demonstrated, both empirically and theoretically, how linear dependencies in the training data cause Bayesian neural networks to generalize poorly under covariate shift — explaining the important and unexpected findings in Izmailov et al. [18]. While the two priors we introduce achieve some improvement in BNN performance under covariate shift, we are only beginning to explore possible remedies. Our work is intended as a step towards understanding the true properties of Bayesian neural networks, and improving the robustness of Bayesian model averaging under covariate shift.

References

[1] Maruan Al-Shedivat, Andrew Gordon Wilson, Yunus Saatchi, Zhiting Hu, and Eric P Xing. Learning scalable deep kernels with recurrent structure. *The Journal of Machine Learning Research*, 18(1):2850–2886, 2017.

[2] Martin Arjovsky. Out of distribution generalization in machine learning. *arXiv preprint arXiv:2103.02667*, 2021.

[3] Gregory W Benton, Wesley J Maddox, Sanae Lotfi, and Andrew Gordon Wilson. Loss surface simplexes for mode connecting volumes and fast ensembling. *arXiv preprint arXiv:2102.13042*, 2021.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[5] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[6] Ginevra Carbone, Matthew Wicker, Luca Laurenti, Andrea Patane, Luca Bortolussi, and Guido Sanguinetti. Robustness of bayesian neural networks to gradient-based attacks. *arXiv preprint arXiv:2002.04359*, 2020.

[7] Alex Chan, Ahmed Alaa, Zhaozhi Qian, and Mihaela Van Der Schaar. Unlabelled data improves bayesian uncertainty calibration under covariate shift. In *International Conference on Machine Learning*, pages 1392–1402. PMLR, 2020.

[8] Oscar Chang, Yuling Yao, David Williams-King, and Hod Lipson. Ensemble model patching: A parameter-efficient variational bayesian neural network. *arXiv preprint arXiv:1905.09453*, 2019.

[9] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

[10] Michael Dusenberry, Ghassen Jerfel, Yeming Wen, Yian Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and scalable bayesian neural nets with rank-1 factors. In *International conference on machine learning*, pages 2782–2792, 2020.

[11] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *nature*, 542(7639):115–118, 2017.

[12] Angelos Filos, Sebastian Farquhar, Aidan N Gomez, Tim GJ Rudner, Zachary Kenton, Lewis Smith, Milad Alizadeh, Arnoud de Kroon, and Yarin Gal. A systematic comparison of bayesian deep learning robustness in diabetic retinopathy tasks. *arXiv preprint arXiv:1912.10481*, 2019.

[13] Vincent Fortuin, Adrià Garriga-Alonso, Florian Wenzel, Gunnar Rätsch, Richard Turner, Mark van der Wilk, and Laurence Aitchison. Bayesian neural network priors revisited. *arXiv preprint arXiv:2102.06571*, 2021.

[14] Yarin 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.

[15] Varun Gulshan, Lily Peng, Marc Coram, Martin C Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Jama*, 316(22):2402–2410, 2016.

[16] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

[18] Pavel Izmailov, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Wilson. What are bayesian neural network posteriors really like? *arXiv preprint arXiv:2104.14421*, 2021.

[19] Jörn-Henrik Jacobsen, Jens Behrmann, Richard Zemel, and Matthias Bethge. Excessive invariance causes adversarial vulnerability. *arXiv preprint arXiv:1811.00401*, 2018.

[20] Norman P Jouppi, Doe Hyun Yoon, George Kurian, Sheng Li, Nishant Patil, James Laudon, Cliff Young, and David Patterson. A domain-specific supercomputer for training deep neural networks. *Communications of the ACM*, 63(7):67–78, 2020.

[21] 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.

[22] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The CIFAR-10 dataset. 2014. `http://www.cs.toronto.edu/kriz/cifar.html`.

[25] 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.

[26] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[27] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[28] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.

[29] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. *Advances in Neural Information Processing Systems*, 32:13153–13164, 2019.

[30] Rhiannon Michelmore, Matthew Wicker, Luca Laurenti, Luca Cardelli, Yarin Gal, and Marta Kwiatkowska. Uncertainty quantification with statistical guarantees in end-to-end autonomous driving control. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7344–7350. IEEE, 2020.

[31] Norman Mu and Justin Gilmer. Mnist-c: A robustness benchmark for computer vision. *arXiv preprint arXiv:1906.02337*, 2019.

[32] Vaishnavh Nagarajan, Anders Andreassen, and Behnam Neyshabur. Understanding the failure modes of out-of-distribution generalization. *arXiv preprint arXiv:2010.15775*, 2020.

[33] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[34] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[35] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *arXiv preprint arXiv:2008.11687*, 2020.

[36] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D Sculley, Sebastian Nowozin, Joshua V Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *arXiv preprint arXiv:1906.02530*, 2019.

[37] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[38] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[39] Abhijit Guha Roy, Jie Ren, Shekoofeh Azizi, Aaron Loh, Vivek Natarajan, Basil Mustafa, Nick Pawlowski, Jan Freyberg, Yuan Liu, Zach Beaver, et al. Does your dermatology classifier know what it doesn't know? detecting the long-tail of unseen conditions. *arXiv preprint arXiv:2104.03829*, 2021.

[40] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[41] Amos Storkey. When training and test sets are different: characterizing learning transfer. *Dataset shift in machine learning*, 30:3–28, 2009.

[42] Amos J Storkey and Masashi Sugiyama. Mixture regression for covariate shift. *Advances in neural information processing systems*, 19:1337, 2007.

[43] Masashi Sugiyama, Benjamin Blankertz, Matthias Krauledat, Guido Dornhege, and Klaus-Robert Müller. Importance-weighted cross-validation for covariate shift. In *Joint Pattern Recognition Symposium*, pages 354–363. Springer, 2006.

[44] Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(5), 2007.

[45] Andrew Trask, Felix Hill, Scott Reed, Jack Rae, Chris Dyer, and Phil Blunsom. Neural arithmetic logic units. *arXiv preprint arXiv:1808.00508*, 2018.

[46] Florian Wenzel, Kevin Roth, Bastiaan S Veeling, Jakub Światkowski, 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.

[47] Matthew Wicker, Luca Laurenti, Andrea Patane, Zhuotong Chen, Zheng Zhang, and Marta Kwiatkowska. Bayesian inference with certifiable adversarial robustness. In *International Conference on Artificial Intelligence and Statistics*, pages 2431–2439. PMLR, 2021.

[48] Andrew Gordon Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *arXiv preprint arXiv:2002.08791*, 2020.

[49] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Sections 4, 5, and 6 provide ample evidence for the claims made in the abstract and the introduction.
   (b) Did you describe the limitations of your work? [Yes] See section 7, particularly the conclusion paragraph at the end.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix Appendix N.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See the lemmas and propositions in section 5.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix E.

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Our code is submitted with the supplemental materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix A.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We report results with error bars in the Appendix whenever that is computationally reasonable. In general, our experimental results have been consistent across different runs and on different machines.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix A.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Datasets used are cited in section 4.
   (b) Did you mention the license of the assets? [Yes] The assets used are well-known public datasets. See Appendix N for details.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Our code is submitted in the supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]