Out of distribution robustness with pre-trained Bayesian neural networks

Xi Wang  
College of Information and Computer Science  
University of Massachusetts Amherst  
xwang3@cs.umass.edu

Laurence Aitchison  
Department of Computer Science  
University of Bristol  
laurence.aitchison@bristol.ac.uk

Abstract

We develop ShiftMatch, a new training-data-dependent likelihood for out of distribution (OOD) robustness in Bayesian neural networks (BNNs). ShiftMatch is inspired by the training-data-dependent “EmpCov” priors from Izmailov et al. (2021a), and efficiently matches test-time spatial correlations to those at training time. Critically, ShiftMatch is designed to leave neural network training unchanged, allowing it to use publically available samples from pretrained BNNs. Using pre-trained HMC samples, ShiftMatch gives strong performance improvements on CIFAR-10-C, outperforms EmpCov priors, and is perhaps the first Bayesian method capable of convincingly outperforming plain deep ensembles. ShiftMatch can be integrated with non-Bayesian methods like deep ensembles, where it offers smaller, but still considerable, performance improvements. Overall, Bayesian ShiftMatch gave slightly better accuracy than ensembles with ShiftMatch, though they both had very similar log-likelihoods.

1 Introduction

Neural networks are increasingly being deployed in real-world, safety-critical settings such as self-driving cars (Bojarski et al., 2016) and medical imaging (Esteva et al., 2017). Accurate uncertainty estimation in these settings is critical, and a common approach is to use Bayesian neural networks (BNNs) to reason explicitly about uncertainty in the weights (MacKay, 1992; Neal, 2012; Graves, 2011; Blundell et al., 2015; Gal & Ghahramani, 2016; Maddox et al., 2019; Aitchison, 2020a,b; Ober & Aitchison, 2021; Unlu & Aitchison, 2021; Khan & Rue, 2021; Daxberger et al., 2021). BNNs indeed are highly effective at improving uncertainty estimation in the in-distribution setting, where the train and test distributions are equal (Zhang et al., 2019; Izmailov et al., 2021b). Critically, we also need the model to continue to perform effectively (or at least degrade gracefully) under distribution shift, such as moving a self-driving car to a different region, or applying medical imaging models to images from different scanners or hospitals. Superficially, BNNs seem like a good choice for this setting: we would hope they would give more uncertainty in regions far from the training data, and thus degrade gracefully as the test domain diverges from the training domain.

However, recent work has highlighted that BNNs can fail to generalise to OOD samples, potentially performing worse than non-Bayesian ensembles (Lakshminarayanan et al., 2017; Ovadia et al., 2019; Izmailov et al., 2021a,b). Izmailov et al. (2021a) gave a key intuition as to why this failure might occur. In particular, consider directions in input/feature space with zero variance under the training data. As the weights in this direction have little or no effect on the output, any weight regularisation reduces the weights in these directions to zero. The zero weights imply that these directions continue to have no effect on the outputs, even if distribution shift subsequently increases variance in those directions. However, BNNs do not work like this. BNNs sample weights in these zero-input-variance directions from the prior. That is fine in the training data domain, as there is no variance in the input
in those directions, so the non-zero weights do not affect the outputs. However, if distribution shift subsequently increases input variance in those directions, then those new high-variance inputs will interact with the non-zero weights to corrupt the output.

Izmailov et al. (2021a) suggested an approach for fixing this issue, by modifying the prior over weights at the input layer, to reduce the variance in the prior over weights in directions where the inputs have little variance. While their approach did beat BNNs with standard Gaussian priors, it performed worse than ensembles (Izmailov et al., 2021a, their Figs. 4,11,12). This failure is surprising: part of the promise of BNNs is that they should perform reasonably OOD by giving more uncertainty away from the training data.

However, the in and out of distribution performance of any Bayesian method depends heavily on the choice of model (prior and likelihood combined). We begin by noting that the training-data-dependent EmpCov priors from Izmailov et al. (2021a) can be equivalently viewed as training-data-dependent likelihoods. We then develop a new training-data-dependent likelihood, ShiftMatch, which has two advantages over EmpCov priors (Izmailov et al., 2021a). First, EmpCov priors apply only to the input layer, so might not be effective for more complex corruptions which are best understood and corrected at later layers. In contrast, our likelihoods modify the activity at every layer in the network, so have the potential to fix complex, nonlinear corruptions. Second, EmpCov modifies the prior over weights, preventing the use of publically available samples from BNNs with standard Gaussian priors (Izmailov et al., 2021b), which is especially important as some gold-standard Bayesian sampling methods are extremely expensive (e.g. by Hamiltonian Monte Carlo, HMC in Izmailov et al. (2021b) took one hour to get a sample on a ResNet trained on CIFAR-10 using “a cluster of 512 TPUs” Izmailov et al., 2021b). In contrast, ShiftMatch keeps the prior and training-time likelihood unchanged, allowing us to directly re-use e.g. gold-standard samples drawn using standard Gaussian IID priors from Izmailov et al. (2021b). Indeed, ShiftMatch is highly efficient as it does not require further retraining or fine-tuning at test time, allowing us to e.g. use a very large batch size.

We found that ShiftMatch improved considerably over all previous baselines for BNN OOD robustness, including BNNs with standard Gaussian and EmpCov priors, and non-Bayesian methods like plain deep ensembles (Fig. 1,3). Further, ShiftMatch can be combined with non-Bayesian methods: we found significant improvement for all methods tested, SGD, ensembles, and Bayes (HMC) (Fig. 4).

2 Background

In BNNs, we use Bayes theorem to compute a posterior distribution over the weights \( w \) given the training input, \( X_{\text{train}} \), and labels, \( y_{\text{train}} \),

\[
P(w|X_{\text{train}}, y_{\text{train}}) = \frac{P(y_{\text{train}}|X_{\text{train}}, w) P(w)}{P(y_{\text{train}}|X_{\text{train}})}.
\]

At test-time, we use Bayesian model averaging (BMA) to compute a predictive distribution over test labels, \( y_{\text{test}} \), given test inputs, \( X_{\text{test}} \), by integrating over the posterior,

\[
P(y_{\text{test}} | X_{\text{test}}, X_{\text{train}}, y_{\text{train}}) = \int P(y_{\text{test}} | X_{\text{test}}, w) P(w|X_{\text{train}}, y_{\text{train}}) \, dw.
\]

In practice, directly using Bayes theorem is intractable in all but very low dimensional settings. As such, alternative approaches, ranging from (HMC) (Neal et al., 2011; Izmailov et al., 2021b), variational inference (Graves, 2011; Blundell et al., 2015), stochastic gradient Langevin dynamics (SGLD) (Welling & Teh, 2011; Zhang et al., 2019) or expectation propagation (EP) (Hernández-Lobato & Adams, 2015) are usually used. HMC is usually regarded as the “gold-standard”, as it will produce exact samples given enough time to sample, while the other methods here will not.

3 Related work

The closest prior work is Izmailov et al. (2021a), which introduces the “EmpCov” prior that uses the training inputs, \( X_{\text{train}} \in \mathbb{R}^{P_{\text{train}} \times N} \), to specify the prior over weights,

\[
P_{\text{EmpCov}}(w|X_{\text{train}}) = \mathcal{N} \left( w; 0, \frac{1}{P_{\text{train}}} X_{\text{train}}^\top X_{\text{train}} \right).
\]
Here, $P_{\text{train}}$ is total number of training examples, $N$ is the number of features, and $w \in \mathbb{R}^N$ is a weight vector mapping from the inputs to a single feature in the first hidden layer. The covariance of the Gaussian prior over weights, $\frac{1}{N_{\text{train}}}X_{\text{train}}^T X_{\text{train}}$, is proportional to the empirical covariance of the training inputs. Thus, the prior suppresses weights along directions with low input variance. Interestingly, the idea of using the inputs to specify a prior over weights arises earlier, in the Zellner G-prior (Zellner, 1986). Remarkably, Izmailov et al. (2021a) and Zellner (1986) propose priors that are in some sense “opposite”: while the EmpCov prior suppresses weights in directions with low input variance, the Zellner-G prior magnifies weights in those directions, so as to ensure that all directions in input space have the same impact on the outputs,

$$
P_{\text{Zellner-G}} (w | X_{\text{train}}) = \mathcal{N} \left( w; 0, \frac{1}{N_{\text{train}}} X_{\text{train}}^T X_{\text{train}} \right).$$

(4)

Of course, the best prior depends on the setting: we suspect that the Zellner-G prior is most appropriate when informative features have very different scales, whereas the EmpCov prior is most appropriate for e.g. images, where the low-variance input directions are likely to be uninformative.

One approach to using unlabelled data to improve OOD robustness is to keep batchnorm on at test-time, so the batch statistics are computed on corrupted test data (Li et al., 2016; Carlucci et al., 2017; Nado et al., 2020; Schneider et al., 2020; Benz et al., 2021). Of course, this work does not consider BNNs: we find the surprising result that just test-time batchnorm gives huge performance improvements in BNNs of 21.9% for CIFAR-10-C for the highest corruption level, which is more than twice the performance improvement for ensembles of 10.6%. We then go on to develop ShiftMatch which gives even larger improvements of 24.6% for BNNs and 12.4% for ensembles. While ShiftMatch bears some superficial similarities to test-time batchnorm, it is in reality quite different. In particular, on convolutional neural networks, batchnorm matches only the means and variances for each channel. In contrast, ShiftMatch matches the spatial covariances statistics for each channel separately, allowing us to correct complex, spatially extended corruptions, and to perfectly correct certain simpler corruptions such as Gaussian blur (Sec. 4.5). We find ShiftMatch gives considerable performance improvements over test-time batchnorm (Figs. 4, 5).

There is work suggesting whitening or transforming the feature correlations (Huang et al., 2018), including for OOD robustness (Roy et al., 2019; Choi et al., 2021). These approaches have at least three key differences from ShiftMatch. First, these approaches whiten at train-time, which dramatically slows down training, and prevents the use of pre-trained networks. In contrast, ShiftMatch training is exactly equivalent to faster standard (B)NN training, allowing the use of pretrained models. Second, these papers transform aggregate across space, and whiten only over channels (e.g. using $C \times C$ matrices, where $C$ is the number of channels). However, this ignores the complex spatial structure present in most corruptions: we find that by transforming spatial correlations we can perfectly remove simple corruptions such as Gaussian blur (Sec. 4.5). As such, we instead transform the spatial correlations of each channel separately (see Fig. 6 for variants). Importantly, these two choices interact: it probably is not possible to work with spatial correlations at training time, because at training time, we only have a few samples in the minibatch (e.g. $N = 128$) to estimate high-dimensional spatial covariances (e.g. for MNIST, these covariance matrices would be $HW \times HW = 784 \times 784$). ShiftMatch largely avoids this issue because it computes the spatial covariances once at the end, using all training and testing data, giving extremely accurate covariance estimates (see Sec. 4.4 for further details).

There is also a line of work (e.g. Sun et al., 2020; Wang et al., 2020; Zhang et al., 2022) aiming at improving the robustness of model by performing self-supervised training at test time. This approach is likely to be too computationally expensive in our setting, as we need to evaluate on many different HMC samples/ensemble members.

4 Methods

4.1 Reinterpreting EmpCov as a training-data-dependent likelihood

The EmpCov prior over weights (Eq. 3) induces a distribution over $f = Xw \in \mathbb{R}^P$, the value of a single output feature for all $P$ inputs,

$$
P_{\text{EmpCov}} (f | X_{\text{test}}) = \mathcal{N} \left( f; 0, \frac{1}{N_{\text{test}}} X_{\text{test}}^T X_{\text{test}} \right).$$

(5)
We can obtain an equivalent distribution over \( f \) by first transforming the inputs using \( \text{ShiftEmpCov} \),

\[
\text{ShiftEmpCov}(X_{\text{test}}, X_{\text{train}}) = X_{\text{test}} \left( \frac{1}{P_{\text{train}}} X_{\text{train}}^T X_{\text{train}} \right)^{1/2} \ .
\] (6)

where the “input”, \( X_{\text{test}} \) is to the left of the semicolon, and the quantities used purely for normalization, \( X_{\text{train}} \), are to the right of the semicolon. Computing \( f \) from these transformed inputs, \( f = \text{ShiftEmpCov}(X_{\text{test}}, X_{\text{train}})/w \), and using a standard, isotropic prior, \( P(w) = \mathcal{N}(0, \frac{1}{\lambda} I) \), recovers the EmpCov distribution over \( f \) (Eq. 5). Importantly, \( \text{ShiftEmpCov} \) moves the explicit training-data-dependence from the prior, to the likelihood, which, for training and test data, are

\[
P_{\text{ShiftEmpCov}}(y_{\text{train}}|X_{\text{train}}, w) = \text{Categorical}(y_{\text{train}}; \pi_{\text{ShiftEmpCov}}(X_{\text{train}}, w; X_{\text{train}})) \ ,
\] (7a)

\[
P_{\text{ShiftEmpCov}}(y_{\text{test}}|X_{\text{test}}, w, X_{\text{train}}) = \text{Categorical}(y_{\text{test}}; \pi_{\text{ShiftEmpCov}}(X_{\text{test}}, w; X_{\text{train}})) \ .
\] (7b)

Here, \( \pi_{\text{ShiftEmpCov}} \) is the neural network with \( \text{ShiftEmpCov} \) at the input layer; it takes input data (\( X_{\text{train}} \) or \( X_{\text{test}} \)) and weights, \( w \), as input and returns a distribution over labels. Again, the last argument, \( X_{\text{train}} \), is to the right of the semicolon because it is used purely for normalization. Importantly, we are using exactly the same function, \( \pi_{\text{ShiftEmpCov}}(X, w; X_{\text{train}}) \) for the train and test likelihoods. Even with this reinterpretation, we still cannot apply \( \text{ShiftEmpCov} \) on pre-trained BNN samples as the training likelihood (Eq. 7a) is different from the standard training likelihood.

### 4.2 Developing improved training-data-dependent likelihoods

Motivated by EmpCov’s equivalent training-data-dependent likelihoods and priors, we considered other training-data-dependent likelihoods that might resolve issues with EmpCov priors. In particular, we designed \( \text{ShiftMatch} \) as a training-data-dependent likelihood that is capable of reusing samples from pre-trained networks. The \( \text{ShiftMatch} \) likelihood is built upon the \( \text{ShiftMatch} \) transformation

\[
\text{ShiftMatch}(H_{\text{test}}; H_{\text{train}}) = \tilde{H}_{\text{test}} \left( \frac{1}{P_{\text{train}}} \tilde{H}_{\text{test}}^T \tilde{H}_{\text{test}} \right)^{-1/2} \left( \frac{1}{P_{\text{train}}} \tilde{H}_{\text{train}}^T \tilde{H}_{\text{train}} \right)^{1/2} + M_{\text{train}} \ .
\] (8)

where \( \tilde{H}_{\text{test}} \) and \( \tilde{H}_{\text{train}} \) are mean-subtracted versions of the test and training inputs/features, \( M_{\text{train}} \) is the training mean, and we use the symmetric matrix square root, (Huang et al., 2018). \( \text{ShiftMatch} \) matches the mean and covariance of the test data to that of the training data; for the covariances,

\[
\frac{1}{P_{\text{train}}} (\text{ShiftMatch}(H_{\text{test}}; H_{\text{train}}) - M_{\text{train}}) (\text{ShiftMatch}(H_{\text{test}}; H_{\text{train}}) - M_{\text{train}})^T = \frac{1}{P_{\text{train}}} \tilde{H}_{\text{train}}^T \tilde{H}_{\text{train}} \ .
\] (9)

As such, applying \( \text{ShiftMatch} \) to the training data leaves it unchanged,

\[
\text{ShiftMatch}(H_{\text{train}}; H_{\text{train}}) = \tilde{H}_{\text{train}} + M_{\text{train}} = H_{\text{train}} \ .
\] (10)

allowing us to train the network without including the \( \text{ShiftMatch} \) layers, which allows us to use e.g. pre-trained gold-standard samples from HMC-trained BNNs (Izmailov et al., 2021b). A summary of the full \( \text{ShiftMatch} \) algorithm can be found in Appendix Alg.1. Overall, \( \text{ShiftMatch} \) likelihoods have the same form as the EmpCov likelihoods (Eq. 7),

\[
P_{\text{ShiftMatch}}(y_{\text{train}}|X_{\text{train}}, w) = \text{Categorical}(y_{\text{train}}; \pi_{\text{ShiftMatch}}(X_{\text{train}}, w; X_{\text{train}})) \ .
\] (11a)

\[
P_{\text{ShiftMatch}}(y_{\text{test}}|X_{\text{test}}, w, X_{\text{train}}) = \text{Categorical}(y_{\text{test}}; \pi_{\text{ShiftMatch}}(X_{\text{test}}, w; X_{\text{train}})) \ .
\] (11b)

The key difference is that for the training data, the likelihoods are equal to those in standard BNNs,

\[
\pi_{\text{ShiftMatch}}(X_{\text{train}}, w; X_{\text{train}}) = \pi(X_{\text{train}}, w) \ 
\] (12a)

\[
P_{\text{ShiftMatch}}(y_{\text{train}}|X_{\text{train}}, w) = P(y_{\text{train}}|X_{\text{train}}, w) \ 
\] (12b)

where standard BNN likelihoods and NNs are denoted by \( P \) and \( \pi \) without any subscript. This implies that \( \text{ShiftMatch} \) posteriors are exactly equal to posteriors in a BNN with standard likelihoods, allowing us to use samples of the weights from pretrained networks,

\[
P_{\text{ShiftMatch}}(w|y_{\text{train}}, X_{\text{train}}) \propto P_{\text{ShiftMatch}}(y_{\text{train}}|X_{\text{train}}, w) P(w) \\
= P(y_{\text{train}}|X_{\text{train}}, w) P(w) \propto P(w|y_{\text{train}}, X_{\text{train}}) .
\] (13)
4.3 Structured covariance estimation in CNNs

The ShiftMatch operation shown in Eq. (8) is appropriate for fully-connected networks with a relatively small number of features. However, this approach cannot be applied directly to convolutional neural networks, because the feature maps are of size $CHW$, where $C$ is the number of channels, $H$ is the height and $W$ is the width. Thus, we end up with huge $CHW \times CHW$ covariance matrices, which presents computational difficulties (e.g. the matrix square roots require $O(C^3 H^3 W^3)$ compute), makes it difficult to accurately estimate them, especially with small (e.g. 128) minibatches.

Instead, it is necessary to choose to structure our covariance estimates in order to improve accuracy and speed up matrix computations. We can choose to match the spatial covariance for each channel separately. This would require us to estimate $C$ separate $HW \times HW$ covariance matrices. Even then, this poses difficulties because images or feature maps can have very large numbers of pixels. Thus, we need to impose further spatial structure, and we choose to treat the covariance along the height and width separately, using a Kronecker factored covariance (Martens & Grosse, 2015), so we end up with a $H \times H$ matrix and a $W \times W$ matrix for each channel (see Fig. 6 for other variants).

4.4 Comparison with batchnorm

ShiftMatch does bear some minor resemblance to batchnorm, in that it matches the feature moments to prespecified values. Indeed, inspired by ShiftMatch, we could develop a “spatial batchnorm”, a generalisation of batchnorm that matches spatial covariances of feature maps to learned values. However, even spatial batchnorm would ultimately be quite different from ShiftMatch. In particular, batchnorm is usually applied only at train-time, and turned off at test-time (though it is sometimes left on at test-time, where it can improve OOD generalisation performance Nado et al., 2020; Schneider et al., 2020). However, ShiftMatch is the opposite in that it is applied only at test-time: at train-time, ShiftMatch is identical to a standard NN without extra normalization layers (Eq. 10). Thus, ShiftMatch allows us to decouple test time from train time normalization, which turns out to be an extremely useful idea. In particular, ShiftMatch has two key advantages over a spatial batchnorm.

First, the required spatial covariance matrices are complex, high-dimensional objects, which require a lot of data to estimate accurately. With ShiftMatch, it is straightforward to get extremely accurate estimates because we estimate these matrices after training the weights using all the test and training data. As we compute the covariances after training, we can use very large batchesizes (e.g. we can fit a batch of 10,000 samples even on a single 2080ti with 11GB of memory), because we do not need to retain feature maps at all layers for use in the backward pass (for further discussion, see e.g. Sander et al., 2021). In contrast, training batches used in a spatial batchnorm would need to be much smaller (e.g. 128 examples), which can give very inaccurate or even low-rank, non-invertible estimates of the covariance matrices. Second, ShiftMatch trains much faster than a spatial batchnorm. In particular, even after carefully simplifyng the covariance matrices, computing the matrix square roots is still very slow. In ShiftMatch, that is fine because we only perform these operations once after training. In contrast, a spatial batchnorm, would need to do these operations at every training iteration, which causes a dramatic slowdown: from around 0.35s for a forward pass without spatial batchnorm to 0.77s for a forward pass with spatial batchnorm using a mini-batch of 128 samples.

4.5 ShiftMatch perfectly removes stationary linear corruptions

ShiftMatch retains all useful the theoretical properties of EmpCov priors, in suppressing pathologies that arise in BNNs when there is low-input variance along a direction in the training but not in the corrupted test data (see Sec. 1 and Izmailov et al., 2021a for more details). ShiftMatch also has considerable theoretical advantages over EmpCov priors, in that certain simpler corruptions can be removed perfectly (Fig. 2). In particular, ShiftMatch can perfectly remove linear spatially stationary corruptions such as Gaussian blur (in the limit of infinitely many samples). Without loss of generality, consider $X_{\text{train}} \in \mathbb{R}^{P_{\text{train}} \times HW}$ as zero-mean training images with a single input channel (e.g. a grayscale image). We have uncorrupted test data, $X_{\text{uncor}} \in \mathbb{R}^{P_{\text{test}} \times HW}$ which is corrupted with a stationary linear spatial operation, repre-
Figure 1: Performance of HMC, SGD, Deep Ensemble, SGLD, mean-field VI and ShiftMatch (applied on HMC samples from Izmailov et al., 2021b) on CIFAR-10-C averaged over 16 different corruption types with 5 intensity level (1 to 5). Intensity 0 stands for clean CIFAR-10 test set without corruption. ShiftMatch significantly improves the robustness against corruption compared with plain HMC and even outperforms Deep Ensemble.

Critical, applying ShiftMatch to the corrupted test data exactly recovers the uncorrupted test data,

\[ \text{ShiftMatch}(X_{\text{cor}}; X_{\text{train}}) = X_{\text{cor}} \left( \frac{1}{P_{\text{test}}} X_{\text{cor}} X_{\text{cor}} \right)^{-1/2} \left( \frac{1}{P_{\text{train}}} X_{\text{train}} X_{\text{train}} \right)^{1/2} \]

\[ = U_{\text{test}} D D_{\epsilon} V^\top \left( V \left( D D_{\epsilon} \right)^{-1} V \right) \left( V D V^\top \right) = U_{\text{test}} D V^\top = X_{\text{uncor}} \]

5 Experiments

First, we applied ShiftMatch to the HMC samples from Izmailov et al. (2021b)\(^1\) for a large-scale ResNet trained on CIFAR-10\(^2\) and tested on CIFAR-10-C\(^3\) (Hendrycks & Dietterich, 2019), and compare to the baselines in Izmailov et al. (2021b). ShiftMatch gives a considerable improvement over using the raw HMC samples and over all baselines (including e.g. ensembles). Second, we show that ShiftMatch performs better than EmpCov priors on small CNNs from Izmailov et al. (2021a) trained on MNIST\(^4\) and tested on MNIST-C\(^5\) (Mu & Gilmer, 2019). Third, we find that ShiftMatch works better than test-time batchnorm when we use HMC samples from Izmailov et al. (2021b), and when we consider non-Bayesian baselines (SGD and ensembles; Fig. 4). Fourth, the HMC samples

\(^1\)izmailovpavel.github.io/neurips_bdl_competition/resources.html no license mentioned
\(^2\)cs.toronto.edu/~kriz/cifar.html no license mentioned
\(^3\)github.com/hendrycks/robustness Apache 2.0
\(^4\)yann.lecun.com/exdb/mnist/ no license mentioned
\(^5\)https://github.com/google-research/mnist-c Apache 2.0
Figure 3: Performance of HMC, EmpCov and ShiftMatch on MNIST-C under 15 different corruption types. The left-most column gives the performance on clean MNIST while the right-most column gives the performance averaged across the 15 corruptions.

from Izmailov et al. (2021b) are slightly odd in that they do not use batchnorm or data-augmentation. We therefore show that ShiftMatch continues to be effective in standard settings with train-time batchnorm and data-augmentation (Fig. 5). Finally, we justify our particular choice of ShiftMatch method by trying out different variants of ShiftMatch (Fig. 6). Experiments were performed on a server of 4 Nvidia 2080ti GPU, and took a total of around 300 GPU hours.

CIFAR-10-C baselines from Izmailov et al. (2021b). We began by applying ShiftMatch to the gold-standard HMC samples from a BNN trained on CIFAR-10 from Izmailov et al. (2021b), and comparing to the baselines from that paper. They used a ResNet-20 with only 40,960 of the 50,000 training samples (in order to “evenly share the data across the TPU devices”), and to ensure deterministic likelihood evaluations (which is necessary for HMC), turned off data augmentation and data subsampling (i.e. full batch training), and used filter response normalization (FRN) (Singh & Krishnan, 2020) rather than batch normalization (Ioffe & Szegedy, 2015). Despite these stringent constraints, their network achieved excellent performance, with greater than 90% accuracy on the clean test set. For further details of how these baselines were generated, please see Izmailov et al. (2021b). We found that ShiftMatch considerably improved over all baselines included in that paper, including the raw HMC samples themselves, and deep ensembles (Fig. 1, Appendix Fig. 7).

MNIST-C baselines from Izmailov et al. (2021a). To compare against EmpCov priors, we were forced to use smaller-scale networks and datasets, because that is all that is provided in the EmpCov paper (Izmailov et al., 2021a). Presumably, this was because they wanted gold-standard HMC samples, but no longer had access to the large scale compute in Izmailov et al. (2021b). In particular, they used smaller models such as multilayer perceptron and a LeNet-5 (LeCun et al., 1998), which contains 2 convolutional layers followed by 3 fully-connected layers. We used the HMC samples for LeNet trained on MNIST under standard Gaussian prior and the EmpCov prior provided by Izmailov et al., 2021a to evaluate different methods on MNIST-C (Mu & Gilmer, 2019). Additionally, we used their code to get an ensemble of 50 independently initialised models trained using SGD with momentum as the combined with cosine-annealed learning rate decay and a batchsize of 80. Again, we found that ShiftMatch performed better than the other methods, including HMC with standard Gaussian and EmpCov priors, and non-Bayesian baselines such as ensembles (Fig. 3).

Test-time batchnorm with BNNs, and ShiftMatch with non-Bayesian methods. Above, we focused on combining ShiftMatch with BNNs, and we found that these models performed better than previous approaches used in the BNN literature (Izmailov et al., 2021b,a). However, ShiftMatch could also be applied to non-Bayesian networks, and test-time batchnorm could be applied to BNNs. We
therefore considered all possible combinations of ShiftMatch and batchnorm with SGD, ensembles and HMC (Fig. 4). To ensure a fair comparison, we used networks, HMC samples, training code and experimental settings from Izmailov et al. (2021b). To be more specific, for SGD, we used a network architecture identical to that of HMC: A ResNet-20 with batchnorm replaced by FRN. The ensemble model was constructed using 50 SGD models with different initialisations. As the underlying network did not have batchnorm layers, we used a variant of ShiftMatch to match the mean and the variance of each channel with the training sample statistics (see Fig. 5 for full batchnorm).

In all cases we found that test-time batchnorm improved over plain models, and ShiftMatch improved over test-time batchnorm. Confirming results from Izmailov et al. (2021b), we found that HMC without either test-time batchnorm or ShiftMatch performed very poorly: worse than ensembles for any level corruption. In contrast, with ShiftMatch, HMC had the highest accuracy but similar test log-likelihoods to deep ensembles with ShiftMatch. Indeed, ShiftMatch gave spectacular improvements in accuracy of 24.6% for HMC, and smaller improvements of 12.4% for ensembles (Appendix Tab. 1).

Non-Bayesian ShiftMatch with data augmentation and training time batchnorm. Next, remember that Izmailov et al. (2021b) used an unusual setting without data augmentation or train-time batchnorm. We therefore additionally evaluated whether ShiftMatch improved OOD robustness in the standard setting with data augmentation and train-time batchnorm. In particular, we trained a ResNet-20 using SGD with momentum, with an initial learning rate of 0.05, a cosine annealing schedule, and a weight decay coefficient of $10^{-4}$. We augmented the training data through random horizontal flip and random crop. To combine ShiftMatch with batchnorm, we added ShiftMatch after the batchnorm layer. Again, we found that ShiftMatch offered improvements over simply using batchnorm, especially at the higher corruption intensities (Fig. 5).

Variants of ShiftMatch. We performed an ablation study to choose the design of ShiftMatch (Fig. 6). EmpCov priors might suggest that applying ShiftMatch on the input alone might be sufficient. However, we found applying ShiftMatch only on the input data had performance much worse than other ShiftMatch variants that matched intermediate features, which emphasised the importance of correcting corruptions at all layers. Second, we considered modifying the spatial covariances. In particular, we considered “w/o Kronecker factored covariance”, where we directly estimate the $HW \times HW$ covariance matrices, and “channel-joint”, where we compute only a single spatial covariance matrix, aggregating information across all channels (we usually have $C$ separate $HW \times HW$ matrices — one for each channel). Both of these modifications performed worse than our approach. Presumably, “w/o Kronecker factored covariance” compromises the ability to obtain accurate covariance estimates, and “channel-joint” compromises the model’s ability to match account for different spatial statistics in different channels. Finally, we considered applying ShiftMatch on the post-activation feature, which gave very slightly worse performance than the usual pre-activation...
Figure 5: All combinations of SGD (red, left three bars), ensembles (blue, right three bars) with batchnorm and ShiftMatch for CIFAR-10-C with data augmentation and batchnorm at train-time.

Figure 6: The performance of various variants of ShiftMatch (described in the main text).

approach. Nonetheless, note that all of these sub-optimal variants outperform plain HMC across all corruption intensities.

6 Conclusion

We developed a new training-data-dependent likelihood for OOD robustness for BNNs, called ShiftMatch. We found that ShiftMatch indeed offered considerable improvements over past approaches to OOD robustness in BNNs, both in CIFAR-10-C (Fig. 1) and in MNIST-C (Fig. 3), and even can give improvements over test-time batchnorm when used in non-Bayesian settings (Figs. 4,5).
References

Aitchison, L. Bayesian filtering unifies adaptive and non-adaptive neural network optimization methods. *Advances in Neural Information Processing Systems*, 33:18173–18182, 2020a.

Aitchison, L. A statistical theory of cold posteriors in deep neural networks. *arXiv preprint arXiv:2008.05912*, 2020b.

Benz, P., Zhang, C., Karjauv, A., and Kweon, I. S. Revisiting batch normalization for improving corruption robustness. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 494–503, 2021.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural network. In *International conference on machine learning*, pp. 1613–1622. PMLR, 2015.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Carlucci, F. M., Porzi, L., Caputo, B., Ricci, E., and Bulo, S. R. Autodial: Automatic domain alignment layers. In *2017 IEEE international conference on computer vision (ICCV)*, pp. 5077–5085. IEEE, 2017.

Choi, S., Jung, S., Yun, H., Kim, J. T., Kim, S., and Choo, J. Robustnet: Improving domain generalization in urban-scene segmentation via instance selective whitening. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11580–11590, 2021.

Daxberger, E., Kristiadi, A., Immer, A., Eschenhagen, R., Bauer, M., and Hennig, P. Laplace redux-effortless bayesian deep learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., and Thrun, S. Dermatologist-level classification of skin cancer with deep neural networks. *nature*, 542(7639):115–118, 2017.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

Graves, A. Practical variational inference for neural networks. *Advances in neural information processing systems*, 24, 2011.

Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

Hernandez, W., Mendez, A., and Göksel, T. Application of principal component analysis to image compression. *Statistics-Growing Data Sets and Growing Demand for Statistics*, 2018.

Hernández-Lobato, J. M. and Adams, R. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International conference on machine learning*, pp. 1861–1869. PMLR, 2015.

Huang, L., Yang, D., Lang, B., and Deng, J. Decorrelated batch normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 791–800, 2018.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Izmailov, P., Nicholson, P., Lotfi, S., and Wilson, A. G. Dangers of bayesian model averaging under covariate shift. *Advances in Neural Information Processing Systems*, 34, 2021a.

Izmailov, P., Vikram, S., Hoffman, M. D., and Wilson, A. G. G. What are bayesian neural network posteriors really like? In *International Conference on Machine Learning*, pp. 4629–4640. PMLR, 2021b.

Khan, M. E. and Rue, H. The bayesian learning rule. *arXiv preprint arXiv:2107.04562*, 2021.
Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, Y., Wang, N., Shi, J., Liu, J., and Hou, X. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016.

MacKay, D. J. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., and Wilson, A. G. A simple baseline for bayesian uncertainty in deep learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Martens, J. and Grosse, R. Optimizing neural networks with kronecker-factorized approximate curvature. In *International conference on machine learning*, pp. 2408–2417. PMLR, 2015.

Mu, N. and Gilmer, J. MNIST-C: A robustness benchmark for computer vision. *arXiv preprint arXiv:1906.02337*, 2019.

Nado, Z., Padhy, S., Sculley, D., D’Amour, A., Lakshminarayanan, B., and Snoek, J. Evaluating prediction-time batch normalization for robustness under covariate shift. *arXiv preprint arXiv:2006.10963*, 2020.

Neal, R. M. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

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

Ober, S. W. and Aitchison, L. Global inducing point variational posteriors for bayesian neural networks and deep gaussian processes. In *International Conference on Machine Learning*, pp. 8248–8259. PMLR, 2021.

Olshausen, B. A. and Field, D. J. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., and Snoek, J. Can you trust your model’s uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.

Roy, S., Siarohin, A., Sangineto, E., Bulo, S. R., Sebe, N., and Ricci, E. Unsupervised domain adaptation using feature-whitening and consensus loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9471–9480, 2019.

Sander, M. E., Ablin, P., Blondel, M., and Peyré, G. Momentum residual neural networks. In *International Conference on Machine Learning*, pp. 9276–9287. PMLR, 2021.

Schneider, S., Rusak, E., Eck, L., Bringmann, O., Brendel, W., and Bethge, M. Improving robustness against common corruptions by covariate shift adaptation. *Advances in Neural Information Processing Systems*, 33:11539–11551, 2020.

Singh, S. and Krishnan, S. Filter response normalization layer: Eliminating batch dependence in the training of deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11237–11246, 2020.

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A., and Hardt, M. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning*, pp. 9229–9248. PMLR, 2020.

Unlu, A. and Aitchison, L. Gradient regularization as approximate variational inference. *Entropy*, 23 (12):1629, 2021.
Wang, D., Shelhamer, E., Liu, S., Olshausen, B., and Darrell, T. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2020.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688. Citeseer, 2011.

Zellner, A. On assessing prior distributions and bayesian regression analysis with g-prior distributions. *Bayesian inference and decision techniques*, 1986.

Zhang, M. M., Levine, S., and Finn, C. Test time robustification of deep models via adaptation and augmentation, 2022. URL [https://openreview.net/forum?id=J1uOGgf-bP](https://openreview.net/forum?id=J1uOGgf-bP).

Zhang, R., Li, C., Zhang, J., Chen, C., and Wilson, A. G. Cyclical stochastic gradient mcmc for bayesian deep learning. *arXiv preprint arXiv:1902.03932*, 2019.
Figure 7: Performance of HMC, SGD, Deep Ensemble, SGLD, Mean-field VI (from dark red to yellow) and ShiftMatch (blue) on CIFAR-10-C under 16 different corruption types with 5 intensity levels.

Table 1: Accuracy improvement given by using ShiftMatch over the plain method. Note that the benefits of ShiftMatch under HMC are around double the benefits of ShiftMatch for SGD or deep ensembles.

| Corruption Type | Intensity 1 | Intensity 2 | Intensity 3 | Intensity 4 | Intensity 5 |
|-----------------|-------------|-------------|-------------|-------------|-------------|
| SGD             | 2.8%        | 5.1%        | 7.7%        | 10.4%       | 13.1%       |
| Deep Ens        | 1.8%        | 3.7%        | 6.2%        | 9.1%        | 12.4%       |
| HMC             | 5.2%        | 8.9%        | 12.8%       | 18.3%       | 24.6%       |

Table 2: Accuracy improvement given by using test-time batchnorm over the plain method.

| Corruption Type | Intensity 1 | Intensity 2 | Intensity 3 | Intensity 4 | Intensity 5 |
|-----------------|-------------|-------------|-------------|-------------|-------------|
| SGD             | 1.3%        | 2.6%        | 4.2%        | 5.8%        | 8.2%        |
| Deep Ens        | 1.4%        | 2.8%        | 4.9%        | 7.2%        | 10.6%       |
| HMC             | 4.5%        | 7.6%        | 10.9%       | 15.8%       | 21.9%       |

A Further experimental results

Here, we present full performance across different corruptions for HMC and other baselines from (Izmailov et al., 2021b) (Fig. 7). Tab. 1 gives performance differences with and without ShiftMatch, and Tab. 2 gives performance differences with and without ShiftMatch, for the usual Izmailov et al. (2021b) setting. Note that HMC benefits far more than non-Bayesian methods, with improvements often being more than double those for ensembles.

B Algorithms

We summarised the end-to-end inference procedure of ShiftMatch in Alg. 1.
Algorithm 1 End-to-end procedure of ShiftMatch on a pre-trained BNN

Input: Test set $X_{\text{test}}$, Training set $X_{\text{train}}$

$M$ posterior samples from an $L$-layer BNN $\{\phi_{l,m}^m(\cdot), \ldots, \phi_{L,m}^m(\cdot)\}, m \in \{1, \ldots, M\}$.

Step 1. Acquire training statistics

for $m = 1, 2, \ldots, M$ do
  $\triangleright$ Acquire the training statistics for all $M$ posterior samples.
  $H_{\text{train}}^0 \leftarrow X_{\text{train}}$
  for $\ell = 0, 1, \ldots, L-1$ do
    $\triangleright$ Compute the training statistics $M_{\text{train}}^{\ell,m}, Q_{\text{train}}^{\ell,m}$.
    $M_{\text{train}}^{\ell,m} \leftarrow \text{Average}(H_{\text{train}}^\ell)$
    $\tilde{H}_{\text{train}}^\ell \leftarrow H_{\text{train}}^\ell - M_{\text{train}}^{\ell,m}$
    $Q_{\text{train}}^{\ell,m} = \left(\frac{1}{P_{\text{train}}} (\tilde{H}_{\text{train}}^\ell)^\top (\tilde{H}_{\text{train}}^\ell)\right)^{1/2}$
    $H_{\text{train}}^{\ell+1} \leftarrow \phi_{\ell+1}(H_{\text{train}}^\ell)$
  end for
  Save the training statistics $(M_{\text{train}}^{\ell,m}, Q_{\text{train}}^{\ell,m}), \ell \in \{0, \ldots, L-1\}$.
end for

Step 2. Perform Bayesian Model Averaging

$p \leftarrow 0$

for $m = 1, 2, \ldots, M$ do
  $\triangleright$ Bayesian Model Averaging over $M$ posterior samples.
  Load $(M_{\text{train}}^{\ell,m}, Q_{\text{train}}^{\ell,m}), \ell \in \{0, \ldots, L-1\}$ from the disk.
  $H_{\text{test}}^0 \leftarrow X_{\text{test}}$
  for $\ell = 0, 1, \ldots, L-1$ do
    $\triangleright$ Match the test-time feature for each layer.
    $M_{\text{test}}^{\ell} \leftarrow \text{Average}(H_{\text{test}}^\ell)$
    $\tilde{H}_{\text{test}}^\ell \leftarrow H_{\text{test}}^\ell - M_{\text{test}}^{\ell}$
    $Q_{\text{test}}^{\ell} = \left(\frac{1}{P_{\text{test}}} (\tilde{H}_{\text{test}}^\ell)^\top (\tilde{H}_{\text{test}}^\ell)\right)^{-1/2}$
    $H_{\text{test}}^{\ell+1} \leftarrow \phi_{\ell+1}(H_{\text{test}}^\ell)$
  end for
  $p \leftarrow p + \frac{1}{M} f(H_{\text{test}}^L)$, where $f$ can be e.g. softmax.
end for

return $p$