Incorporating Unlabelled Data into Bayesian Neural Networks

Anonymous authors
Paper under double-blind review

Abstract

Conventional Bayesian Neural Networks (BNNs) are unable to leverage unlabelled data to improve their predictions. To overcome this limitation, we introduce *Self-Supervised Bayesian Neural Networks*, which use unlabelled data to learn models with suitable prior predictive distributions. This is achieved by leveraging contrastive pretraining techniques and optimising a variational lower bound. We then show that the prior predictive distributions of self-supervised BNNs capture problem semantics better than conventional BNN priors. In turn, our approach offers improved predictive performance over conventional BNNs, especially in low-budget regimes.

1 Introduction

Note: this is a revised version of the original submission, where edited parts are highlighted in green.

Bayesian Neural Networks (BNNs) are powerful probabilistic models that combine the flexibility of deep neural networks with the theoretical underpinning of Bayesian methods (Mackay, 1992; Neal, 1995). Indeed, as they place priors over their parameters and perform posterior inference, BNN advocates consider them a principled approach for uncertainty estimation (Wilson & Izmailov, 2020; Abdar et al., 2021), which can be helpful for label-efficient learning (Gal et al., 2017).

Conventionally, BNN researchers have focused on improving predictive performance using human-crafted priors over network parameters or predictive functions (e.g., Louizos et al., 2017; Tran et al., 2020; Matsubara et al., 2021; Fortuin et al., 2021a). However, several concerns have been raised with BNN priors (Wenzel et al., 2020; Noci et al., 2021). It also stands to reason that the vast store of semantic information contained in unlabelled data should be incorporated into BNN priors, and that the potential benefit of doing so likely exceeds the benefit of designing better, but ultimately human-specified, priors over parameters or functions. Unfortunately, as standard BNNs are explicitly only models for supervised prediction, they cannot leverage such semantic information from unlabelled data by conditioning on it.

To overcome this shortcoming, **we introduce *Self-Supervised Bayesian Neural Networks*** (§3), which use unlabelled data to learn improved priors over functions. In other words, our approach improves the BNN prior predictive distribution (which we will just call *prior predictive* in the remainder of the paper) by incorporating unlabelled data into it. This contrasts with designing different but ultimately *human-specified* priors, which is the prevalent approach.

In practice, self-supervised BNNs generate pseudo-labelled data using unlabelled data and data augmentation, similar to contrastive learning (Oord et al., 2019; Chen et al., 2020a;b; Grill et al., 2020; Hénaff et al., 2020). We use this generated data to learn models with powerful prior predictive distributions. To do this, we perform unsupervised model learning by optimising a lower bound of a log-marginal likelihood dependent on the pseudo-labelled data. This biases the prior towards functions that assign augmented image pairs a larger likelihood of having the same label than distinct images (Fig. 1a). Following pretraining, we perform inference in the learnt model to make predictions.

We then further demonstrate that self-supervised BNN prior predictives reflect input-pair semantic similarity better than normal BNN priors (§4). To do so, we develop a methodology to better understand the prior predictive distributions of BNNs. Our approach is to measure the probability

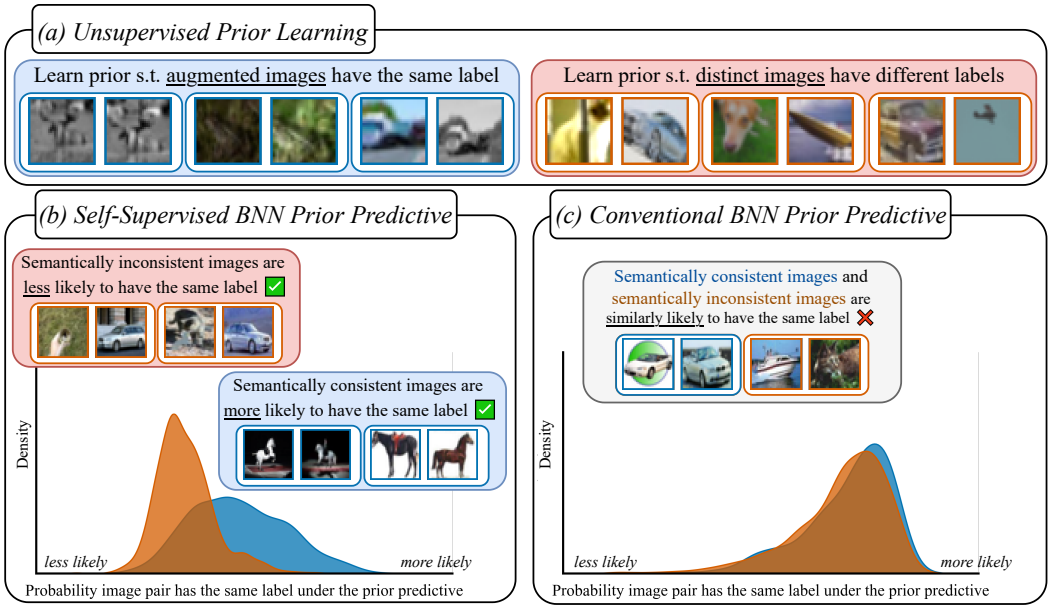

Figure 1: **Self-Supervised Bayesian Neural Networks**. (a) Pre-training in self-supervised BNNs corresponds to unsupervised prior learning. We learn a model with a prior distribution such that augmented images likely have the same label and distinct images likely have different labels under the prior predictive. (b) Self-supervised BNN priors assign higher probabilities to semantically consistent image pairs having the same label compared to semantically inconsistent image pairs. Here, semantically consistent image pairs have the same ground-truth label, and semantically inconsistent image pairs have different ground-truth labels. The plot shows a kernel density estimate of the log-probability that same-class and different-class image pairs are assigned the same label under the prior. (c) Unlike self-supervised prior predictives, conventional BNN prior predictives assign similar probabilities to semantically consistent and semantically inconsistent image pairs having the same label.

of *pairs* of data points having the same label under the prior. Intuitively, pairs of points that are more semantically similar should be more likely to have the same label under the prior predictive. Applying this methodology, we see that the functional priors learned by self-supervised BNNs distinguish same-class input pairs and different-class input pairs much better than conventional BNNs (Fig. 1b).

Finally, we empirically demonstrate that the improved prior predictives of self-supervised BNNs translate to improved predictive performance, especially in problem settings with few labelled examples (§5).

2 Background: Bayesian Neural Networks

Let $f_\theta(x)$ be a neural network with parameters θ and $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ be an dataset. We want to predict y from x. A BNN specifies a prior over parameters, $p(\theta)$, and a likelihood, $p(y|f_\theta(x))$, which in turn define the posterior $p(\theta|\mathcal{D}) \propto p(\theta) \prod_i p(y_i|f_\theta(x_i))$. To make predictions, we approximate the posterior predictive $p(y_\star|x_\star, \mathcal{D}) = \mathbb{E}_{p(\theta|\mathcal{D})}[p(y_\star|f_\theta(x_\star))]$.

Improving BNN priors has been a long-standing goal for the community, primarily through improved human-designed priors. One approach is to improve the prior over the network's parameters (Louizos et al., 2017; Nalisnick, 2018). Others place priors directly over predictive functions (Flam-Shepherd et al., 2017; Sun et al., 2019; Matsubara et al., 2021; Nalisnick et al., 2021; Raj et al., 2023). Both approaches, however, present challenges—the mapping between the network's parameters and predictive functions is complex, while directly specifying our beliefs over predictive functions is itself a highly challenging task. For these reasons, as well as convenience, isotropic Gaussian priors over network parameters remain the most common choice

(Fortuin, 2022), despite concerns (Wenzel et al., 2020). In contrast to these works, we propose to *learn* better functional priors from unlabelled data via contrastive learning.

3 Self-Supervised BNNs

Conventional BNNs are unable to use unlabelled data to improve their predictions. To overcome this limitation, we introduce *Self-Supervised BNNs*. At a high level, self-supervised BNNs allow unlabelled data to be incorporated by using it to learn a powerful prior that captures known similarities between inputs. In practice, we can utilise ideas from contrastive learning to learn models with prior predictive distributions that reflect the semantics of different input pairs. The high-level idea is thus to use prior knowledge in the form of data augmentations, for which we believe that the semantic content of the data should be invariant to them. We can then use a variational method to learn a function-space prior that assigns more weight to functions, whose outputs on unlabelled data are also invariant to these augmentations.

Problem Specification. Suppose $\mathcal{D}^u = \{x_i^u\}_{i=1}^N$ is an unlabelled dataset of examples $x_i^u \in \mathbb{R}^n$. Let $\mathcal{D}^t = \{(x_i^t, y_i^t)\}_{i=1}^T$ be a labelled dataset corresponding to a supervised "downstream" task, where y_i^t is the target associated with x_i^t. We want to use both $\mathcal{D}^u$ and $\mathcal{D}^t$ to train a deep learning model for predicting y given x with probabilistic parameters θ, where all information about the data is incorporated through the distribution on θ. That is, we predict using $p(y|x, \theta)$ for a given θ.

3.1 Incorporating Unlabelled Data into BNNs

The simplest way one might proceed, is to place a prior on θ and then condition on both $\mathcal{D}^u$ and $\mathcal{D}^t$, leading to a posterior $p(\theta|\mathcal{D}^u, \mathcal{D}^t) \propto p(\theta|\mathcal{D}^u)\, p(\mathcal{D}^t|\mathcal{D}^u, \theta)$. However, if we are working with conventional BNNs, which are explicitly models for supervised prediction, then $p(\theta|\mathcal{D}^u) = p(\theta)$. Further, as the predictions depend only on the parameters, $p(\mathcal{D}^t|\mathcal{D}^u, \theta) = p(\mathcal{D}^t|\theta)$, which then means that $p(\theta|\mathcal{D}^u, \mathcal{D}^t) = p(\theta|\mathcal{D}^t)$. Thus, we cannot incorporate $\mathcal{D}^u$ by naïvely conditioning on it.

To get around this problem, we propose to instead use $\mathcal{D}^u$ to generate hypothetical labelled data and then condition our predictive model on it. In other words, we will use $\mathcal{D}^u$ to guide a *self-supervised* training of the model, thereby incorporating the desired information from our unlabelled data. To do this, we will draw on data augmentation (Yaeger et al., 1996; Krizhevsky et al., 2012; Shorten & Khoshgoftaar, 2019) and contrastive learning (Oord et al., 2019; Chen et al., 2020b;a; Grill et al., 2020; Hénaff et al., 2020; Chen & He, 2020; Foster et al., 2020; Miao et al., 2023).

Indeed, such approaches that use data augmentation provide effective means for making use of prior information. Although it is difficult to encode our prior beliefs with hand-crafted priors over neural network parameters, we can construct augmentation schemes that we expect to preserve the semantic properties of different inputs. We thus also expect these augmentation schemes to preserve the unknown downstream labels of different inputs. The challenge is now to transfer the beliefs—implicitly defined through our data augmentation scheme—into our model.

One simple way to do this would be to just augment the data in $\mathcal{D}^t$ when training a standard BNN with standard data augmentation. However, this would ignore the rich information available in $\mathcal{D}^u$. Instead, we will use a construct from contrastive learning to generate pseudo-labelled data from $\mathcal{D}^u$, and then condition θ on both this pseudo data and $\mathcal{D}^t$.

Concretely, suppose we have a set of data augmentations $\mathcal{A} = \{a : \mathbb{R}^n \to \mathbb{R}^n\}$ that preserve semantic content. We use $\mathcal{A}$ and $\mathcal{D}^u$ to generate a *contrastive dataset* $\mathcal{D}^c$ that reflects our subjective beliefs by:

1. Drawing M examples from $\mathcal{D}^u$ at random, $\{\hat{x}_i\}_{i=1}^M$, where i indexes the subset, not $\mathcal{D}^u$;
2. For each $\hat{x}_i$, sampling $a^A, a^B \sim \mathcal{A}$ and augmenting, giving $\tilde{x}_i^A = a^A(\hat{x}_i)$ and $\tilde{x}_i^B = a^B(\hat{x}_i)$;
3. Forming $\mathcal{D}^c$ by assigning $\tilde{x}_i^A$ and $\tilde{x}_i^B$ the same class label, which is the subset index i.

We thus have $\mathcal{D}^c = \{(x_i^c, y_i^c)\}_{i=1}^{2M} = \{(\tilde{x}_i^A, i)\}_{i=1}^M \cup \{(\tilde{x}_i^B, i)\}_{i=1}^M$, where the labels are between 1 and M. The task associated with our generated data is thus to predict the subset index corresponding to each augmented

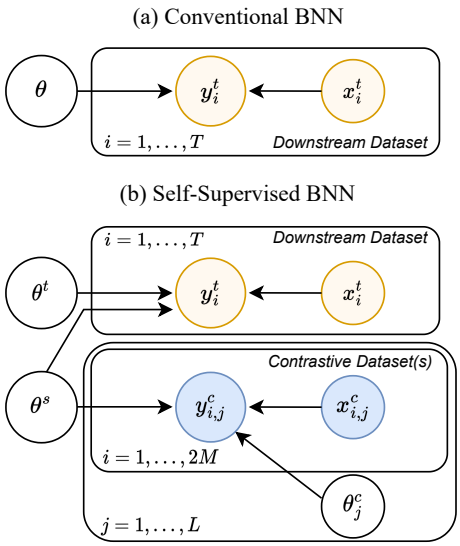

Figure 2: **BNN Probabilistic Models.** (a) Probabilistic model for conventional BNNs. (b) Probabilistic model for self-supervised BNNs. We share parameters between different tasks, which allows us to condition on generated self-supervised data. j indexes self-supervised tasks, i indexes datapoints.

example. We can repeat this process L times and create a set of contrastive task datasets, $\{\mathcal{D}_j^c\}_{j=1}^L$. Here, we consider the number of generated datasets L to be a fixed, finite hyper-parameter, but we discuss the implications of setting $L = \infty$ in Appendix A. Note that rather than using a hand-crafted prior to capture our semantic beliefs, we have instead used data augmentation in combination with unlabelled data.

Next, to link each $\mathcal{D}_j^c$ with the downstream predictions, we use parameter sharing (see Fig. 2). Specifically, we introduce parameters θ_j^c for each $\mathcal{D}_j^c$, parameters θ^t for $\mathcal{D}^t$, and shared-parameters θ^s that are used for both the downstream and the contrastive tasks. $\mathcal{D}^u$ thus informs downstream predictions through θ^s, via $\{\mathcal{D}_j^c\}_{j=1}^L$. For example, θ^t and θ_j^c could be the parameters of the last layer of a neural network, while θ^s could be the shared parameters of earlier layers.

Learning in this Framework. We now discuss different options for learning in this framework. Using the Bayesian approach, one would place priors over θ^s, θ^t, and each θ_j^c. This then defines a posterior distribution given the observed data $\{\mathcal{D}_j^c\}_{j=1}^L$ and $\mathcal{D}^t$. To make predictions on the downstream task, which depend on θ^s and θ^t only, we would then use the posterior predictive:

$$p(y_\star^t | x_\star, \{\mathcal{D}_j^c\}_{j=1}^L, \mathcal{D}^t) = \mathbb{E}_{p(\theta^s | \{\mathcal{D}_j^c\}_{j=1}^L, \mathcal{D}^t)}[\mathbb{E}_{p(\theta^t | \theta^s, \mathcal{D}^t)}[p(y_\star^t | x_\star, \theta^s, \theta^t)]], \tag{1}$$

where we have (i) noted that the downstream task parameters θ^t are independent of $\{\mathcal{D}_j^c\}_{j=1}^L$ given the shared parameters θ^s and (ii) integrated over each θ_j^c and θ^t in the definition of $p(\theta^s | \{\mathcal{D}_j^c\}_{j=1}^L, \mathcal{D}^t)$.

Alternatively, one can learn a point estimate for θ^s, e.g., with MAP estimation, and perform full posterior inference for θ^t and θ_j^c only. This would be a *partially stochastic* network, which Sharma et al. (2023) showed often outperforms fully stochastic networks while being more practical. In the case where all parameters are shared up to the last (linear) layer, this is also known as the *neural linear model* (Lázaro-Gredilla & Figueiras-Vidal, 2010), which has been shown to have many desirable properties (Ober & Rasmussen, 2019; Harrison et al., 2023). Learning in this way is also known as *model learning*, as used in deep kernels and variational autoencoders (Kingma & Welling, 2013; Rezende et al., 2014; Wilson et al., 2015; 2016), where in the case of Gaussian processes (GPs), a point estimate of kernel parameters is learned to also define a learned prior over functions. Note that our approach can also be considered kernel learning, where the representations of input data after the shared layers θ^s define the reproducing kernel Hilbert space (RKHS) and the kernel is given by their inner products. In learning a point estimate for θ^s, one is learning a suitable model to perform

inference in. This model has the following prior predictive:

$$p(y_\star^t|x_\star, \{\mathcal{D}_j^c\}_{j=1}^L) = \mathbb{E}_{p(\theta^t)}[p(y_\star^t|x_\star, \theta_\star^s, \theta^t)], \tag{2}$$

where $\theta_\star^s$ is the learnt value of θ^s.[1] We would then update our beliefs over θ^t in light of observed data. Through $\theta_\star^s$, we are using $\mathcal{D}^u$ to effectively learn a prior over the functions that can be represented by the network in combination with θ^t. Learning a point estimate for θ^s is thus our main approach.

3.2 Self-Supervised BNNs in Practice

We now use our framework to propose a practical two-step algorithm for self-supervised BNNs.

Preliminaries. We focus on image-classification problems. We use an encoder $f_{\theta^s}(\cdot)$ that maps images to representations and is shared across the contrastive tasks and the downstream task. The shared parameters θ^s thus are the base encoder's parameters. We also normalise the representations produced by this encoder. For the downstream dataset, we use a linear readout layer from the encoder representations, i.e., we have $\theta^t = \{W^t, b^t\}$ and $y_i^t \sim \text{softmax}(W^t f_{\theta^s}(x_i) + b^t)$. The rows of W_j^t are thus class template vectors, that is, a data point will achieve the highest possible softmax probability for a certain class if its representation is equal to (a scaled version of) the corresponding row of the weight matrix. For the contrastive tasks, we use a linear layer without biases, i.e., $\theta_j^c = W_j^c$, and j indexes contrastive tasks. We place Gaussian priors over θ^s, θ^t, and each θ_j^c.

Pre-training θ^s (Step I). Here, we learn a point estimate for the base encoder parameters θ^s, which induces a functional prior over the downstream task labels (see Eq. 2). To learn θ^s, we want to optimise the (potentially penalised) log-likelihood $\log p(\{\mathcal{D}_j^c\}_{j=1}^L, \mathcal{D}^t|\theta^s)$, but this would require integrating over θ^t and each θ^c. Instead, we use the evidence lower bound (ELBO):

$$\tilde{\mathcal{L}}_j^c(\theta^s) = \mathbb{E}_{q(\theta_j^c)}[\log p(\mathcal{D}_j^c|\theta^s, \theta_j^c)] - D_{\text{KL}}(q(\theta_j^c)||p(\theta_j^c)) \leq \log p(\mathcal{D}_j^c|\theta^s), \tag{3}$$

where $q(\theta_j^c)$ is a variational distribution over the contrastive task parameters.

Rather than learning a different variational distribution for each contrastive task j, we amortise the inference and exploit the structure of the contrastive task. The contrastive task is to predict the corresponding source image index for each augmented image. That is, for the first pair of augmented images in a given contrastive task dataset, we want to predict class "1", for the second pair, we want to predict class "2", and so forth. The label is the index within the contrastive dataset, not the full dataset. To predict these labels, we use a linear layer applied to an encoder that produces normalised representations. We want a suitable variational distribution for this linear layer.

To make progress, we define $\tilde{z}_i^A = f_{\theta^s}(\tilde{x}_i^A)$ and $\tilde{z}_i^B = f_{\theta^s}(\tilde{x}_i^B)$, which are the *representations* of given images. To solve the contrastive task well, we want to map z_1^A and z_1^B to class "1", z_2^A and z_2^B to class "2", and so forth. We define $\omega_i = 0.5(\tilde{z}_i^A + \tilde{z}_i^B)$, i.e., ω_i is the mean representation for each augmented pair of images. Because the rows of the linear layer weight matrix W_j^c are effectively class templates, we use $q(W_j^c; \tau, \sigma^2) = \mathcal{N}(\mu_j^c, \sigma^2 I)$ with $\mu_j^c = \begin{bmatrix}\omega_1^T & \dots & \omega_M^T\end{bmatrix}/\tau$. In words, the mean representation of each augmented image pair is the class template for each source image, which should solve the contrastive task well. Note that since this makes the last-layer weights data-dependent, it also renders them softmax outputs invariant to the arbitrary ordering of the data points in the batch, since if one permutes the x_i, and thus z_i, this also automatically permutes the ω_i and thus rows of W_j^c in the same way. Also, recall that the W_j^c are auxiliary parameters that are just needed during the contrastive learning to learn a θ^s that induces a good functional prior, but are discarded afterwards and not used for the actual supervised task of interest. The variational parameters τ and σ^2 determine the magnitude of the linear layer and the per-parameter variance, vary throughout training, are shared across contrastive tasks j, and are learnt by maximising Eq. (3) with reparameterisation gradients (Price, 1958; Kingma & Welling, 2013).

Both the contrastive tasks and downstream task provide information about the base encoder parameters θ^s. One option would be to learn the base encoder parameters θ^s using only data derived from $\mathcal{D}^u$ (Eq. 3),

[1] Alternatively, this approach can be understood as learning the prior $p(\theta^s, \theta^t) = p(\theta^t)\delta_{\theta^s = \theta_\star^s}$, where δ is the Dirac delta function.

Algorithm 1 Self-Supervised BNNs

Input: augmentations $\mathcal{A}$, unlabelled data $\mathcal{D}^u$, task data $\mathcal{D}^t$, contrastive prior $p(W^c)$
for $j = 1, \ldots, L$ **do** ▷ Unsupervised prior learning
  Draw subset $\{\hat{x}_i\}_{i=1}^M$, set $\mathcal{D}_j^c = \{\}$
  for $i = 1, \ldots, M$ **do** ▷ Create contrastive task
    Sample $a^A, a^B \sim \mathcal{A}$
    $\tilde{x}_i^A = a^A(\hat{x}_i)$, $\tilde{x}_i^B = a^B(\hat{x}_i)$
    $\tilde{z}_i^A = f_{\theta^s}(\tilde{x}_i^A)$, $\tilde{z}_i^B = f_{\theta^s}(\tilde{x}_i^B)$
    $\omega_i = 0.5(\tilde{z}_i^A + \tilde{z}_i^B)$
    Add $(\tilde{x}_i^A, i)$ and $(\tilde{x}_i^B, i)$ to $\mathcal{D}_j^c$.
  end for
  $W_j^c = \begin{bmatrix}\omega_1^T & \ldots & \omega_M^T\end{bmatrix}/\tau + \epsilon$, with $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$
  $\tilde{\mathcal{L}}(\tau, \sigma^2, \theta^s) = \log p(\theta^s) + \frac{1}{2M}\mathbb{E}_{q(W_j^c)}[p(\mathcal{D}_j^c|\theta^s, W_j^c)] - \bar{D}_{\mathrm{KL}}[q(W_j^c)||p(W^c))]$
  Update θ^s, τ, σ^2 to maximise $\tilde{\mathcal{L}}(\tau, \sigma^2, \theta^s)$
end for
Approximate $p(\theta^t|\mathcal{D}^t, \theta^s) \simeq q(\theta^t)$ ▷ Evaluation
Predict using $\mathbb{E}_{q(\theta^t)}[p(y_\star^t|x_\star, \theta^s, \theta^t)]$

which would correspond to a standard self-supervised learning setup. In this case, the learnt prior would be task-agnostic. An alternative approach is to use both $\mathcal{D}^t$ and $\mathcal{D}^u$ to learn θ^s, which corresponds to a semi-supervised setup. To do so, we can use the ELBO for the downstream data:

$$\tilde{\mathcal{L}}^t(\theta^s) = \mathbb{E}_{q(\theta^t)}[\log p(\mathcal{D}^t|\theta^t, \theta^s)] - D_{\mathrm{KL}}[q(\theta^t)||p(\theta^t)] \leq \log p(\mathcal{D}^t|\theta^s), \tag{4}$$

where $q(\theta^t) = \mathcal{N}(\theta^t; \mu^t, \Sigma^t)$ is a variational distribution over the downstream task parameters; Σ^t is diagonal. We can then maximise $\sum_j \tilde{\mathcal{L}}_j^c(\theta^s) + \alpha \cdot \tilde{\mathcal{L}}^t(\theta^s)$, where α is a hyper-parameter that controls the weighting between the downstream task and contrastive task datasets. We consider both variants of our approach, using Self-Supervised BNNs to refer to the variant that pre-trains only with $\{\mathcal{D}_j^c\}_{j=1}^L$ and Self-Supervised BNNs* to refer to the variant that uses both $\{\mathcal{D}_j^c\}_{j=1}^L$ and $\mathcal{D}^t$.

Downstream Inference (Step II). Having learnt a point estimate for θ^s, we can use any approximate inference algorithm to infer θ^t. Here, we use a post-hoc Laplace approximation (Daxberger et al., 2021).

Algorithm 1 summarises Self-Supervised BNNs, which learn θ^s with $\mathcal{D}^u$ only. We found tempering with the mean-per-parameter KL divergence, $\bar{D}_{\mathrm{KL}}$, improved performance, in line with other work (e.g., Krishnan et al., 2022). Moreover, we generate a new $\mathcal{D}_j^c$ per gradient step so L corresponds to the number of gradient steps. As shown on Algorithm 1, our full loss is $\tilde{\mathcal{L}}(\tau, \sigma^2, \theta^s) = \log p(\theta^s) + \frac{1}{2M}\mathbb{E}_{q(W_j^c)}[p(\mathcal{D}_j^c|\theta^s, W_j^c)] - \bar{D}_{\mathrm{KL}}[q(W_j^c)||p(W^c))]$. The first term of this loss is a prior over the shared parameters, in our case a Gaussian prior, which is equivalent to weight decay. The second term is where the actual contrastive learning happens, namely it is an expected Categorical log-likelihood (i.e., cross-entropy) over the softmax logits under W_j^c. Recall that the rows of this weight matrix are the mean embeddings vectors ω_i, so this likelihood encourages the inner products $\omega_i^\top \tilde{z}_j$ to be large for $i = j$, that is, drawing two augmentations of the same image towards their mean and thus each other, and to be small for $i \neq j$, that is, pushing augmentations of different images away from each other. Finally, the third KL term places a Gaussian prior on W^c, which in our case means that the ω_i that make up this matrix (and thus the embeddings $\tilde{z}_j$) cannot grow without bounds to maximize the likelihood score, but have to stay reasonably close together. Moreover, following best-practice for contrastive learning (Chen et al., 2020a), we use a non-linear *projection head* $g_\psi(\cdot)$ *only* for the contrastive tasks. For further details, see Appendix A.

Pre-training as Prior Learning. In this work, our central aim is to incorporate unlabelled data into BNNs. To achieve this, in practice, we perform model learning using contrastive datasets generated from the unlabelled data and data augmentation. This corresponds to an unsupervised prior learning step. Since

our objective function during this is a principled lower bound on the log-marginal likelihood, it is similar to type-II maximum likelihood (ML), which is often used to learn parameters for deep kernels (Wilson et al., 2015) of Gaussian processes (Williams & Rasmussen, 2006), and recently also for BNNs (Immer et al., 2021). As such, similar to type-II ML, our approach can be understood as a form of prior learning. Although we learn only a point-estimate for θ^s, this fixed value induces a prior distribution over predictive functions through the task-specific prior $p(\theta^t)$. However, while normal type-II ML learns this prior using the observed data itself, our approach maximises a marginal likelihood derived from unsupervised data.

4 How Good Are Self-Supervised BNN Prior Predictives?

We showed our approach incorporates unlabelled data into the downstream task prior predictive distribution (Eq. 2). We also argued that, as the generated contrastive data encodes our beliefs about the semantic similarity of different image pairs, incorporating the unlabelled data should improve the functional prior. We now examine whether this is indeed the case.

Unfortunately, prior predictive checks are hard to apply to BNNs because of the high dimensionality of the input space. We will therefore introduce our own novel metric to assess the suitability of the prior predictive.

The basis for our approach is to note that, intuitively, a suitable prior should reflect a belief that *the higher the semantic similarity between pairs of inputs, the more likely these inputs are to have the same label*. Therefore, rather than inspecting the prior predictive at single points in input space, we examine the *joint* prior predictive of *pairs* of inputs with known semantic relationships. Indeed, it is far easier to reason about the relationship between examples than to reason about distributions over high-dimensional functions.

Note that this is of course only a reasonable assumption in cases where we believe to have sufficiently good knowledge of semantic similarity in our data domain. That is, we need to have a set of data augmentations for the contrastive tasks, for which we can be reasonably certain that the true labels in our downstream task will be invariant to them. Recent results in contrastive learning suggest that this is indeed the case for natural images paired with the augmentations used in SimCLR (Chen et al., 2020a), which is why we use these in our experiments.

To compute our proposed metric, we consider different groups of input pairs. Each group is comprised of input pairs with known semantic similarity. For example, for image data, we could use images of the same class as a group with high semantic similarity, and image pairs from different classes as a group with lower semantic similarity. To investigate the properties of the prior, we can evaluate the probability that input pairs from different groups are assigned the same label under the prior predictive. We can qualitatively investigate the behaviour of this probability across and within different groups. For a prior to be more adapted to the task than an uninformative one, input pairs from groups with higher semantic similarity should be more likely to have the same label under the prior predictive.

Moreover, we can extend this methodology to quantitatively evaluate the prior. Suppose we have G groups of input pairs, $\mathcal{G}_g = \{(x_i^g, \hat{x}_i^g)_{i=1}^{|\mathcal{G}_g|}\}$ with $g = 1, \ldots, G$, and suppose $\mathcal{G}_1$ is the group with the highest semantic similarity, $\mathcal{G}_2$ is the group with the second highest semantic similarity, and so forth. We define $\rho(x, \hat{x})$ as the probability that inputs $x, \hat{x}$ have the same label under the prior predictive, i.e., $\rho(x, \hat{x}) = \mathbb{E}_\theta[p(y(x) = y(\hat{x})|\theta)]$ where $y(x)$ is the label corresponding to input x. We then define *the prior evaluation score*, α, as:

$$\alpha = \mathbb{E}[\mathbb{I}(\rho(x^1, \hat{x}^1) > \ldots > \rho(x^G, \hat{x}^G))], \tag{5}$$

where we compute the expectation sampling $(x^1, \hat{x}^1) \sim \mathcal{G}_1$ and so forth. This is the probability that the prior ranks randomly sampled input pairs correctly, in terms of semantically similar groups being assigned higher probabilities of their input pairs having the same label. We now use this methodology to compare conventional BNNs and self-supervised BNNs.

Experiment Details. We investigate the different priors on CIFAR10. For the BNN, we follow Izmailov et al. (2021b) and use a ResNet-20-FRN with a $\mathcal{N}(0, 1/5)$ prior over the parameters. For the self-supervised BNN, we learn a base encoder of the same architecture with $\mathcal{D}^u$ only and sample from the prior predictive using Eq. (2). θ^t are the parameters of the linear readout layer. For the image pair groups, we use: (i) an

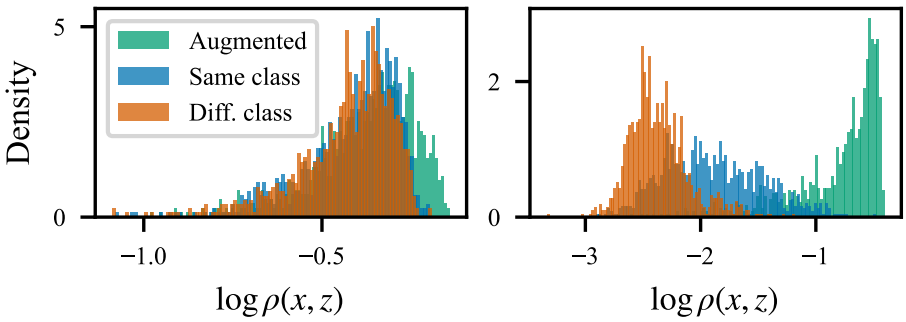

Figure 3: **BNN Prior Predictives.** We investigate prior predictives by computing the probability ρ that particular image pairs have the same label under the prior, and examining the distribution of ρ across different sets of image pairs. We consider three sets of differing semantic similarity: (i) augmented images; (ii) images of the same class; and (iii) images of different classes. Left: Conventional BNN prior. Right: Self-supervised BNN learnt prior predictive. The self-supervised learnt prior reflects the semantic similarity of the different image pairs better than the BNN prior, which is reflected in the spread between the different distributions.

Table 1: **Prior Evaluation Scores**. Mean and standard deviation across three seeds shown. Self-supervised priors are better than standard BNN priors.

Prior Predictive	Prior Evaluation Score α
BNN — Gaussian	0.261 ± 0.024
BNN — Laplace	0.269 ± 0.007
Self-Supervised BNN	**0.680** ± 0.063

image from the validation set (the "base image") and an augmented version of the same image; (ii) a base image and another image of the same class; and (iii) a base image and an image of a different class. As these image pair groups have decreasing semantic similarity, we want the first group to be the most likely to have the same label, and the last group to be the least likely. See Appendix B.3 for more details.

Graphical Evaluation. First, we visualise the BNN and self-supervised BNN prior predictive (Fig. 1 and 3). The standard BNN prior predictive reflects a belief that all three image pair groups are similarly likely to have the same label, and thus does not capture semantic information well. In contrast, the self-supervised prior reflects a belief that image pairs with higher semantic similarity are more likely to have the same label. In particular, the self-supervised prior is able to distinguish between image pairs of the same class and of different classes, *even without access to any ground-truth labels*.

Quantitative Evaluation. We now quantify how well different prior predictives reflect data semantics. In Table 1, we see that conventional BNN priors reflect semantic similarity much less than self-supervised BNN priors, matching our qualitative evaluation. Note that this measure has of course been designed by us to capture the kind of property in the prior that our contrastive training is meant to induce, and should therefore just be seen as a confirmation that our proposed approach works as expected. There are, naturally, many other properties that one could desire in a prior, which are not captured by this metric.

5 Self-Supervised BNNs Excel in Low-Label Regimes

In the previous section, we showed that self-supervised BNN prior predictives reflect semantic similarity of input pairs better than conventional BNNs (§4). One hopes that this translates to improved predictive performance, particularly when conditioning on small numbers of labels, which is where the prior has the largest effect (Gelman et al., 1995; Murphy, 2012). We now show that this is indeed the case. Self-supervised

Table 2: **BNN Predictive Performance**. We measure the performance of different BNNs for different numbers of labels. We consider in-distribution prediction, out-of-distribution (OOD) generalisation, and OOD detection. Shown is the mean across three seeds (ensembles use only one seed). The out-of-distribution generalisation results average over all corruptions with intensity level five from CIFAR10-C (Hendrycks & Dietterich, 2019). Recall that the SS BNN is performing the contrastive learning separately from and the SS BNN* jointly with the downstream task. We see that self-supervised BNNs offer improved predictive performance over conventional BNNs, especially in the low-data regime.

Dataset	# labelled points	↓ Negative Log Likelihood							
		MAP	LL Laplace	SWAG	SS BNN	SS BNN*	Deep Ensemble	SS BNN Ensemble	SS BNN* Ensemble
CIFAR10	50	7.594	2.259	2.332	1.012	**0.996**	3.689	0.939	**0.911**
	500	2.504	1.895	2.072	0.456	**0.441**	1.805	0.402	**0.379**
	5000	1.570	1.327	1.028	**0.363**	0.369	0.846	0.312	**0.297**
	50000	0.631	0.424	0.311	0.326	**0.256**	0.272	0.270	**0.197**
CIFAR100	50	11.86	4.585	6.840	4.505	**4.495**	10.38	**4.450**	4.494
	500	5.536	4.359	5.282	2.640	**2.614**	4.867	2.532	**2.500**
	5000	4.319	3.362	3.518	**1.689**	1.910	3.052	**1.522**	1.642
	50000	1.834	1.469	1.250	1.435	**1.139**	1.088	1.220	**0.934**
CIFAR10 to CIFAR10-C **(OOD Generalisation)**	50	7.140	2.275	2.353	1.723	**1.697**	3.970	1.645	**1.605**
	500	2.838	2.045	2.234	1.278	**1.260**	2.100	1.163	**1.126**
	5000	2.423	1.644	1.705	**1.217**	1.237	1.382	**1.091**	1.095
	50000	1.944	1.244	**1.215**	1.272	1.225	**0.984**	1.102	1.042
		↑ AUROC (%)							
CIFAR10 vs SVHN **(OOD Detection)**	50	54.4	48.4	52.3	92.3	**92.4**	53.6	94.4	**94.5**
	500	61.2	61.1	51.1	94.3	**94.8**	62.1	95.3	**95.8**
	5000	83.3	84.6	59.6	**96.3**	94.6	92.9	**97.3**	97.1
	50000	93.8	92.6	76.4	**96.2**	95.4	96.8	**97.3**	96.9

BNNs offer improved predictive performance over standard BNNs, with especially large gains when making predictions given small numbers of observed labels.

5.1 Semi-Supervised Learning

Training Datasets. We evaluate the performance of different BNNs on the CIFAR10 and CIFAR100 datasets, which are standard benchmarks within the BNN community. We evaluate the performance of different baselines when conditioning on 50, 500, 5000, and 50000 labels from the training set.

Algorithms. As baselines, we consider the following BNNs: MAP, SWAG (Maddox et al., 2019), a deep ensemble with 5 ensemble members (Lakshminarayanan et al., 2017), and last-layer Laplace (Daxberger et al., 2021). The conventional baselines use standard data augmentation and were chosen because they support batch normalisation (Ioffe & Szegedy, 2015). We consider two variants of self-supervised BNNs: Self-Supervised BNNs pretrain using $\mathcal{D}^u$ only, while Self-Supervised BNNs* also use $\mathcal{D}^t$. Both variants use a non-linear projection head when pretraining and the data augmentations suggested by Chen et al. (2020a). We use a post-hoc Laplace approximation for the task-specific parameters (the last-layer parameters). We further consider ensembling self-supervised BNNs.

Evaluation. To evaluate the predictive performance of these BNNs, we report the negative log-likelihood (NLL). This is a proper scoring rule that simultaneously measures the calibration and accuracy of the different networks, and is thus an appropriate measure for *overall* predictive performance (Gneiting & Raftery, 2007). We also report the accuracy and expected calibration error (ECE) in Appendix Table 5. We further assess out-of-distribution (OOD) generalisation from CIFAR10 to CIFAR10-C (Hendrycks & Dietterich, 2019). Moreover, we evaluate whether these BNNs can detect out-of-distribution inputs from SVHN (Netzer et al., 2011) when trained on CIFAR10. We report the area under the receiver operator curve (AUROC) metric using the predictive entropy. We want the OOD inputs from SVHN to have higher predictive entropies than the in-distribution inputs from CIFAR10.

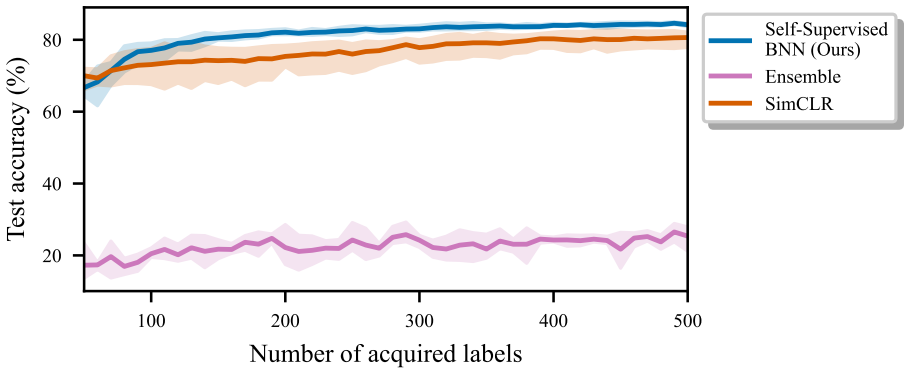

Figure 4: **Low-Budget Active Learning** on CIFAR10. We compare (i) a self-supervised BNN, (ii) SimCLR, and (iii) a deep ensemble. For the self-supervised BNN and the ensemble, we acquire points with BALD. We use predictive entropy for SimCLR, which does not provide epistemic uncertainty estimates. Mean and std. shown (3 seeds). The methods that incorporate unlabelled data perform best by far, with our method slightly outperforming SimCLR.

Results. In Table 2, we report the NLL for each BNN when making predictions with different numbers of labelled examples. We see that self-supervised BNNs offer improved predictive performance over the baselines. In fact, on the full CIFAR-10 test set, a single self-supervised BNN* outperforms a deep ensemble, whilst being 5x cheaper when making predictions. We also show that self-supervised BNNs can also be ensembled to further improve their predictive performance. Incorporating the labelled training data during pretraining (SS BNN*) usually improves predictive performance. Self-supervised BNNs also offer strong performance out-of-distribution and consistently are able to perform out-of-distribution detection. Indeed, they are the only method with an AUROC exceeding 90% at all dataset sizes. In a further analysis in Appendix Table 5, we also see that the improved NLL of self-supervised BNNs is in large part due to improvements in predictive accuracy, and that incorporating labelled data during pretraining also boosts accuracy. Overall, these results accord with our earlier findings about the improved prior predictives of self-supervised BNNs compared to standard BNNs, and highlight the substantial benefits of incorporating unlabelled data into the BNN pipeline. Moreover, we perform an ablation of our variational distribution $q(W_j^c)$ in Appendix Table 4, where we see that our data-dependent mean is indeed needed for good performance.

5.2 Active Learning

We now highlight the benefit of incorporating unlabelled data in an active learning problem.

We consider low-budget active learning, which simulates a scenario where labelling examples is extremely expensive. We use the CIFAR10 training set as the unlabelled pool set from which to label points. We assume an initial train set of 50 labelled points, randomly selected, and a validation set of the same size. We acquire 10 labels per acquisition round up to 500 labels and evaluate using the full test set. We compare self-supervised BNNs to a deep ensemble, the strongest BNN baseline. We use BALD (Houlsby et al., 2011) as the acquisition function for the deep ensemble and self-supervised BNN, which provide epistemic uncertainty estimates. We further compare to SimCLR using predictive entropy for acquisition because SimCLR does not model epistemic uncertainty.

In Fig. 4, we see that the methods that leverage unlabelled data perform the best. In particular, the self-supervised BNN with BALD acquisition achieves the highest accuracy across most numbers of labels, and substantially outperforms the deep ensemble. This confirms the benefit of incorporating unlabelled data in active learning settings, which by definition are semi-supervised and include unlabelled data. Moreover, our approach slightly outperforms SimCLR, suggesting that our Bayesian treatment of contrastive learning yields better uncertainties than conventional non-Bayesian contrastive learning. This is also confirmed in Appendix Fig. 5, where we see that our approaches yield consistently lower calibration errors than SimCLR.

6 Related Work

Improving BNN Priors. We demonstrated that BNNs have poor prior predictive distributions (§4), a concern shared by others (e.g., Wenzel et al., 2020; Noci et al., 2021; Izmailov et al., 2021a). The most common approaches to remedy this are through designing better priors, typically over network parameters (Louizos et al., 2017; Nalisnick, 2018; Atanov et al., 2019; Fortuin et al., 2021b) or predictive functions directly (Sun et al., 2019; Tran et al., 2020; Matsubara et al., 2021, see Fortuin (2022) for an overview). In contrast, our approach incorporates vast stores of unlabelled data into the prior distribution through variational model learning. Similarly, other work also *learns* priors, but typically using labelled data e.g., by using meta-learning (Garnelo et al., 2018; Rothfuss et al., 2021) or type-II maximum likelihood (Wilson et al., 2015; Immer et al., 2021), or by using transfer learning in an *ad hoc* way (Shwartz-Ziv et al., 2022). Notably, function-space variational inference methods (Sun et al., 2019; Rudner et al., 2023) often also use unlabelled data, which has been shown to potentially improve the out-of-distribution performance of these models (Lin et al., 2023). However, in this case, the unlabelled data is only used for evaluating the KL divergence in function space, a practice which has theoretically been shown to be insufficient Burt et al. (2020). Conversely, our work uses semantic information from the unlabelled data to actually *inform* the function-space prior. Another related line of work is concerned with learning invariances from data in Bayesian models using the marginal likelihood (van der Wilk et al., 2018; Immer et al., 2022). This case is essentially the opposite of our setting, as there, the labels are known but the augmentations are learned, while in our case, the augmentations constitute our prior knowledge, but we do not know the data labels.

A Perspective on Contrastive Learning. We offer a Bayesian interpretation and understanding of contrastive learning (§3). Under our framework, pretraining is understood as model learning—a technique for finding probabilistic models with prior predictive distributions that capture our semantic beliefs. There has been much other work on understanding contrastive learning (e.g., Wang & Isola, 2020; Wang & Liu, 2021). Some appeal to the InfoMax principle (Becker & Hinton, 1992). Zimmermann et al. (2022) argue that contrastive learning inverts the data-generating process, while Aitchison (2021) cast InfoNCE as the objective of a self-supervised variational auto-encoder. Ganev & Aitchison (2021) formulate several semi-supervised learning objectives as lower bounds of log-likelihoods in a probabilistic model of data curation.

Semi-Supervised Deep Generative Models. Deep generative models (DGMs) are a fundamentally different approach for label-efficient learning (Kingma & Welling, 2013; Kingma et al., 2014; Joy et al., 2020). A semi-supervised DGM models the full distribution $p(x, y)$ with generative modelling, and so incorporates unlabelled data by learning to generate it. Unlike BNNs, we can condition the parameters of a DGM on unlabelled data. In contrast, our approach does not model the data distribution—the unlabelled data is used to construct pseudo-labelled tasks that encode our prior beliefs. Self-supervised BNNs are discriminative models, which tend to be more scalable and perform better for discriminative tasks compared to full generative modelling (Ng & Jordan, 2001; Bouchard & Triggs, 2004). Finally, Sansone & Manhaeve (2022) try to unify self-supervised learning and generative modelling under one framework.

7 Conclusion

We introduced *Self-Supervised Bayesian Neural Networks*, which allow semantic information from unlabelled data to be incorporated into BNN priors. Using a novel evaluation scheme, we showed that self-supervised BNNs learn functional priors that better reflect the semantics of the data than conventional BNNs. In turn, they offer improved predictive performance over conventional BNNs, especially in low-data regimes. Going forward, we believe that effectively leveraging unlabelled data will be critical to the success of BNNs in many, if not most, potential applications. We hope our work encourages further development in this crucial area.

References

Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty

quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021.

Laurence Aitchison. InfoNCE is a variational autoencoder, July 2021. URL http://arxiv.org/abs/2107.02495. arXiv:2107.02495 [cs, stat].

Andrei Atanov, Arsenii Ashukha, Kirill Struminsky, Dmitry Vetrov, and Max Welling. The Deep Weight Prior. *arXiv:1810.06943 [cs, stat]*, February 2019. URL http://arxiv.org/abs/1810.06943. arXiv: 1810.06943.

Suzanna Becker and Geoffrey E. Hinton. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355(6356):161–163, January 1992. ISSN 1476-4687. doi: 10.1038/355161a0. URL https://www.nature.com/articles/355161a0. Number: 6356 Publisher: Nature Publishing Group.

Guillaume Bouchard and Bill Triggs. The tradeoff between generative and discriminative classifiers. In *16th IASC International Symposium on Computational Statistics (COMPSTAT'04)*, pp. 721–728, 2004.

David R Burt, Sebastian W Ober, Adrià Garriga-Alonso, and Mark van der Wilk. Understanding variational inference in function-space. In *Third Symposium on Advances in Approximate Bayesian Inference*, 2020.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. *arXiv:2002.05709 [cs, stat]*, June 2020a. URL http://arxiv.org/abs/2002.05709. arXiv: 2002.05709.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big Self-Supervised Models are Strong Semi-Supervised Learners. *arXiv:2006.10029 [cs, stat]*, October 2020b. URL http://arxiv.org/abs/2006.10029. arXiv: 2006.10029.

Xinlei Chen and Kaiming He. Exploring Simple Siamese Representation Learning, November 2020. URL http://arxiv.org/abs/2011.10566. arXiv:2011.10566 [cs].

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux-effortless bayesian deep learning. *Advances in Neural Information Processing Systems*, 34:20089–20103, 2021.

Daniel Flam-Shepherd, James Requeima, and David Duvenaud. Mapping gaussian process priors to bayesian neural networks. In *NIPS Bayesian deep learning workshop*, volume 3, 2017.

Vincent Fortuin. Priors in bayesian deep learning: A review. *International Statistical Review*, 2022.

Vincent Fortuin, Adrià Garriga-Alonso, Mark van der Wilk, and Laurence Aitchison. BNNpriors: A library for Bayesian neural network inference with different prior distributions. *Software Impacts*, 9:100079, 2021a.

Vincent Fortuin, Adrià Garriga-Alonso, Florian Wenzel, Gunnar Rätsch, Richard Turner, Mark van der Wilk, and Laurence Aitchison. Bayesian Neural Network Priors Revisited. *arXiv:2102.06571 [cs, stat]*, February 2021b. URL http://arxiv.org/abs/2102.06571. arXiv: 2102.06571.

Adam Foster, Rattana Pukdee, and Tom Rainforth. Improving transformation invariance in contrastive representation learning. In *International Conference on Learning Representations*, 2020.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pp. 1183–1192. PMLR, 2017.

Stoil Ganev and Laurence Aitchison. Semi-supervised learning objectives as log-likelihoods in a generative model of data curation, October 2021. URL http://arxiv.org/abs/2008.05913. arXiv:2008.05913 [cs, stat].

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, S. M. Ali Eslami, and Yee Whye Teh. Neural Processes, July 2018. URL http://arxiv.org/abs/1807.01622. arXiv:1807.01622 [cs, stat].

Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 1995.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised Learning, September 2020. URL http://arxiv.org/abs/2006.07733. arXiv:2006.07733 [cs, stat].

James Harrison, John Willes, and Jasper Snoek. Variational bayesian last layers. In *The Twelfth International Conference on Learning Representations*, 2023.

Dan Hendrycks and Thomas Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In *International Conference on Learning Representations*, 2018.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

Olivier J. Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, S. M. Ali Eslami, and Aaron van den Oord. Data-Efficient Image Recognition with Contrastive Predictive Coding, July 2020. URL http://arxiv.org/abs/1905.09272. arXiv:1905.09272 [cs].

Alexander Immer, Matthias Bauer, Vincent Fortuin, Gunnar Rätsch, and Khan Mohammad Emtiyaz. Scalable marginal likelihood estimation for model selection in deep learning. In *International Conference on Machine Learning*, pp. 4563–4573. PMLR, 2021.

Alexander Immer, Tycho van der Ouderaa, Gunnar Rätsch, Vincent Fortuin, and Mark van der Wilk. Invariance learning in deep neural networks with differentiable laplace approximations. *Advances in Neural Information Processing Systems*, 35:12449–12463, 2022.

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

Pavel Izmailov, Patrick Nicholson, Sanae Lotfi, and Andrew Gordon Wilson. Dangers of Bayesian Model Averaging under Covariate Shift. *arXiv:2106.11905 [cs, stat]*, June 2021a. URL http://arxiv.org/abs/2106.11905. arXiv: 2106.11905.

Pavel Izmailov, Sharad Vikram, Matthew D. Hoffman, and Andrew Gordon Wilson. What Are Bayesian Neural Network Posteriors Really Like? *arXiv:2104.14421 [cs, stat]*, April 2021b. URL http://arxiv.org/abs/2104.14421. arXiv: 2104.14421.

Tom Joy, Sebastian M Schmon, Philip HS Torr, N Siddharth, and Tom Rainforth. Capturing label characteristics in vaes. *arXiv preprint arXiv:2006.10102*, 2020.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, and Max Welling. Semi-Supervised Learning with Deep Generative Models, October 2014. URL http://arxiv.org/abs/1406.5298. arXiv:1406.5298 [cs, stat].

Ranganath Krishnan, Pi Esposito, and Mahesh Subedar. Bayesian-Torch: Bayesian neural network layers for uncertainty estimation, January 2022. URL https://github.com/IntelLabs/bayesian-torch.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Miguel Lázaro-Gredilla and Aníbal R Figueiras-Vidal. Marginalized neural network mixtures for large-scale regression. *IEEE transactions on neural networks*, 21(8):1345–1351, 2010.

Jihao Andreas Lin, Joe Watson, Pascal Klink, and Jan Peters. Function-space regularization for deep bayesian classification. In *Fifth Symposium on Advances in Approximate Bayesian Inference*, 2023.

Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. *Advances in neural information processing systems*, 30, 2017.

David J C Mackay. Bayesian Methods for Adaptive Models. Technical report, 1992.

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, 2019.

Takuo Matsubara, Chris J Oates, and François-Xavier Briol. The ridgelet prior: A covariance function approach to prior specification for bayesian neural networks. *The Journal of Machine Learning Research*, 22(1):7045–7101, 2021.

Ning Miao, Tom Rainforth, Emile Mathieu, Yann Dubois, Yee Whye Teh, Adam Foster, and Hyunjik Kim. Learning instance-specific augmentations by capturing local invariances. *International Conference on Machine Learning*, 2023.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Eric Nalisnick, Jonathan Gordon, and José Miguel Hernández-Lobato. Predictive complexity priors. In *International Conference on Artificial Intelligence and Statistics*, pp. 694–702. PMLR, 2021.

Eric Thomas Nalisnick. *On priors for Bayesian neural networks*. University of California, Irvine, 2018.

Radford M Neal. BAYESIAN LEARNING FOR NEURAL NETWORKS. Technical report, 1995.

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

Andrew Ng and Michael Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, 14, 2001.

Lorenzo Noci, Kevin Roth, Gregor Bachmann, Sebastian Nowozin, and Thomas Hofmann. Disentangling the Roles of Curation, Data-Augmentation and the Prior in the Cold Posterior Effect. *arXiv:2106.06596 [cs]*, June 2021. URL http://arxiv.org/abs/2106.06596. arXiv: 2106.06596.

Sebastian W Ober and Carl E Rasmussen. Benchmarking the neural linear model for regression. In *Second Symposium on Advances in Approximate Bayesian Inference*, 2019.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding, January 2019. URL http://arxiv.org/abs/1807.03748. arXiv:1807.03748 [cs, stat].

Robert Price. A useful theorem for nonlinear devices having gaussian inputs. *IRE Transactions on Information Theory*, 4(2):69–72, 1958.

Vishnu Raj, Tianyu Cui, Markus Heinonen, and Pekka Marttinen. Incorporating functional summary information in bayesian neural networks using a dirichlet process likelihood approach. In *International Conference on Artificial Intelligence and Statistics*, pp. 6741–6763. PMLR, 2023.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pp. 1278–1286. PMLR, 2014.

Jonas Rothfuss, Vincent Fortuin, Martin Josifoski, and Andreas Krause. Pacoh: Bayes-optimal meta-learning with pac-guarantees. In *International Conference on Machine Learning*, pp. 9116–9126. PMLR, 2021.

Tim GJ Rudner, Sanyam Kapoor, Shikai Qiu, and Andrew Gordon Wilson. Function-space regularization in neural networks: A probabilistic perspective. In *International Conference on Machine Learning*, pp. 29275–29290. PMLR, 2023.

Emanuele Sansone and Robin Manhaeve. Gedi: Generative and discriminative training for self-supervised learning. *arXiv preprint arXiv:2212.13425*, 2022.

Mrinank Sharma, Sebastian Farquhar, Eric Nalisnick, and Tom Rainforth. Do bayesian neural networks need to be fully stochastic? In *International Conference on Artificial Intelligence and Statistics*, pp. 7694–7722. PMLR, 2023.

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

Ravid Shwartz-Ziv, Micah Goldblum, Hossein Souri, Sanyam Kapoor, Chen Zhu, Yann LeCun, and Andrew Gordon Wilson. Pre-Train Your Loss: Easy Bayesian Transfer Learning with Informative Priors, May 2022. URL http://arxiv.org/abs/2205.10279. arXiv:2205.10279 [cs].

Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional variational bayesian neural networks. *arXiv preprint arXiv:1903.05779*, 2019.

Ba-Hien Tran, Simone Rossi, Dimitrios Milios, and Maurizio Filippone. All you need is a good functional prior for bayesian deep learning. *arXiv preprint arXiv:2011.12829*, 2020.

Mark van der Wilk, Matthias Bauer, ST John, and James Hensman. Learning invariances using the marginal likelihood. *Advances in Neural Information Processing Systems*, 31, 2018.

Feng Wang and Huaping Liu. Understanding the Behaviour of Contrastive Loss. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2495–2504, Nashville, TN, USA, June 2021. IEEE. ISBN 978-1-66544-509-2. doi: 10.1109/CVPR46437.2021.00252. URL https://ieeexplore.ieee.org/document/9577669/.

Tongzhou Wang and Phillip Isola. Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 9929–9939. PMLR, November 2020. URL https://proceedings.mlr.press/v119/wang20k.html. ISSN: 2640-3498.

Yeming Wen, Paul Vicol, Jimmy Ba, Dustin Tran, and Roger Grosse. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv preprint arXiv:1803.04386*, 2018.

Florian Wenzel, Kevin Roth, Bastiaan S. Veeling, Jakub Swiatkowski, 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*, February 2020. URL http://arxiv.org/abs/2002.02405. Publisher: arXiv.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Andrew G Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *Advances in neural information processing systems*, 33:4697–4708, 2020.

Andrew G Wilson, Zhiting Hu, Russ R Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. *Advances in neural information processing systems*, 29, 2016.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep Kernel Learning, November 2015. URL http://arxiv.org/abs/1511.02222. arXiv:1511.02222 [cs, stat].

Larry Yaeger, Richard Lyon, and Brandyn Webb. Effective training of a neural network character classifier for word recognition. *Advances in neural information processing systems*, 9, 1996.

Yang You, Igor Gitman, and Boris Ginsburg. Scaling sgd batch size to 32k for imagenet training. *arXiv preprint arXiv:1708.03888*, 6(12):6, 2017.

Roland S. Zimmermann, Yash Sharma, Steffen Schneider, Matthias Bethge, and Wieland Brendel. Contrastive Learning Inverts the Data Generating Process, April 2022. URL http://arxiv.org/abs/2102.08850. arXiv:2102.08850 [cs].

A Self-Supervised BNNs: Further Considerations

We introduced *Self-Supervised BNNs* (§3), which benefit from unlabelled data for improved predictive performance within the probabilistic modelling framework. To summarise, our conceptual framework uses data augmentation to create a set of contrastive datasets $\{\mathcal{D}_j^c\}_{j=1}^L$. In our probabilistic model, conditioning on this data is equivalent to incorporating unlabelled data into the task prior predictive. We now discuss further considerations and provide further details.

Theoretical Considerations. In §3.1, we treated the number of contrastive task datasets, L, as a fixed hyper-parameter. However, one could generate an potentially infinite number of datasets, in which case, the posterior $p(\theta^s|\{\mathcal{D}_j^c\}_{j=1}^L, \mathcal{D}^t) \propto p(\theta^s) \cdot p(\mathcal{D}^t|\theta^s) \cdot \prod_{j=1}^L p(\mathcal{D}_j^c|\theta^s)$ will collapse to a delta function, and will be dominated by the contrastive tasks. This justified learning a point estimate for θ^s, but if one wanted to avoid this behaviour, one could re-define the posterior:

$$\tilde{p}(\theta^s|\{\mathcal{D}_j^c\}_{j=1}^L, \mathcal{D}^t) \propto p(\theta^s) \cdot p(\mathcal{D}^t|\theta^s) \cdot \prod_{j=1}^L p(\mathcal{D}_j^c|\theta^s)^{\gamma/L}. \tag{6}$$

The log-posterior would equal, up to a constant:

$$\log \tilde{p}(\theta^s|\{\mathcal{D}_j^c\}_{j=1}^L, \mathcal{D}^t) = \log p(\theta^s) + \log p(\mathcal{D}^t|\theta^s) + \frac{\gamma}{L}\sum_{j=1}^L \log p(\mathcal{D}_j^c|\theta^s). \tag{7}$$

Here, the total evidence contributed by the contrastive datasets is independent of L, and instead controlled by the hyper-parameter γ. This is equivalent to posterior tempering. The final term of the above equation could also be re-defined as an *average* log-likelihood when sampling different $\mathcal{D}^c$, i.e., we could use:

$$\log \tilde{p}(\theta^s|\mathcal{D}^u, \mathcal{D}^t) = \log p(\theta^s) + \log p(\mathcal{D}^t|\theta^s) + \gamma \mathbb{E}_{\mathcal{D}^c}[p(\mathcal{D}^c|\theta^s)]. \tag{8}$$

In this case, we look for a distribution over θ^s where $\mathcal{D}^c$ has a high likelihood on average. Our practical algorithm samples a different $\mathcal{D}^c$ per gradient step, and instead weights the $\mathcal{D}^t$ term with α, which is similar to the above approach if we set $\alpha = 1/\gamma$ and modify the prior term as needed. Re-defining the framework in this way allows it to support potentially infinite numbers of generated contrastive datasets. We further note that this framework could naturally be extended to multi-task scenarios.

Practical Considerations. In practice, we share parameters θ^s across all tasks (i.e., across the generated contrastive tasks and the actual downstream task), and learn a θ^t separately for each task. We focus on image classification problems and let θ^s be the parameters of a *base encoder*, which produces a representation z. θ^t are the parameters of a linear readout layer that makes predictions from z. We discussed learning a point-estimate for θ^s by optimising an ELBO derived from the unlabelled data only:

$$\tilde{\mathcal{L}}_j^c(\theta^s) = \mathbb{E}_{q(\theta_j^c)}[\log p(\mathcal{D}_j^c|\theta^s, \theta_j^c)] - D_{\mathrm{KL}}(q(\theta_j^c)||p(\theta_j^c)) \leq \log p(\mathcal{D}_j^c|\theta^s). \tag{9}$$

We (optionally) further include an ELBO derived using task-specific data:

$$\tilde{\mathcal{L}}^t(\theta^s) = \mathbb{E}_{q(\theta^t)}[\log p(\mathcal{D}^t|\theta^t, \theta^s)] - D_{\mathrm{KL}}[q(\theta^t)||p(\theta^t)] \leq \log p(\mathcal{D}^t|\theta^s). \tag{10}$$

Our final objective is:

$$\mathcal{L}(\theta^s) = \log p(\theta^s) + \alpha \tilde{\mathcal{L}}^t(\theta^s) + \mathbb{E}_{\mathcal{D}^c}[\tilde{\mathcal{L}}_j^c], \tag{11}$$

where $\alpha = 0$ would learn the base-encoder parameters only using the unlabelled data. α controls the weighting between the generated contrastive task data and the downstream data. We see the above objective is closely related to a (lower bound of) Eq. (8). We further modify this objective function, following best practice, either in the contrastive learning or Bayesian deep learning communities, and improve performance by:

1. We add a non-linear projection head to the base encoder architecture *only for the contrastive task datasets*. As such, we use $z = g_\phi(f_{\theta^s}(x))/||g_\phi(f_{\theta^s}(x))||$ for $\mathcal{D}^c$. $g_\phi(\cdot)$ is the projection head, and the representation is normalised. For the downstream tasks, we use $z = f_{\theta^s}(x)$, i.e., we "throw-away" the projection head. This is best practice within the contrastive learning community (Chen et al., 2020a;b).
2. We *temper* the KL divergence term, using the mean-per-parameter KL divergence (denoted as $\bar{D}_{\mathrm{KL}}(\cdot||\cdot)$). Tempering is necessary for several Bayesian deep learning algorithms to perform well (Wenzel et al., 2020; Krishnan et al., 2022).
3. We generate a new contrastive task dataset per gradient step, update on that dataset, and then discard it. This follows standard contrastive learning algorithms (Chen et al., 2020a).
4. We rescale the likelihood terms in the ELBOs $\tilde{\mathcal{L}}^t(\theta^s)$ and $\tilde{\mathcal{L}}^c_j(\theta^s)$ to be average per-datapoint log-likelihoods, e.g., we use $\frac{1}{|\mathcal{D}^c_j|}\log p(\mathcal{D}^c_j|\theta^t_j, \theta^s)$.
5. Instead of having an explicit prior distribution over θ^s, we use standard weight-decay for training, i.e., we specify a penalty on the norm of the weights of the encoder *per gradient step*.

Together, these changes yield the objective function used in Algorithm 1. Finally, we note that different practical algorithms ensue depending on the choice of θ^t, the choice of θ^s, and the techniques used to perform approximate inference. We employ variational inference and learn a point estimate for θ^s, but there are other choices possible.

B Experiment Details

We now provide further experiment details and additional results. The vast majority of experiments were run on an internal compute cluster using Nvidia Tesla V100 or A100 GPUs. The maximum runtime for an experiment was less than 16 hours.

B.1 Semi-Supervised Learning (§5)

B.1.1 Datasets

We consider the CIFAR10 and CIFAR100 datasets (Krizhevsky et al., 2009). The entire training set is the unsupervised set, and we suppose that we have access to different numbers of labels. For the evaluation protocols, we reserve a validation set of 1000 data points from the test set and evaluate using the remaining 9000 labels.

To assess out-of-distribution generalisation, we further evaluation on the CIFAR-10-C dataset (Hendrycks & Dietterich, 2018). We compute the average performance across all corruptions with intensity level five.

B.1.2 Self-Supervised BNNs

Base Architecture. We use a ResNet-18 architecture, modified for the size of CIFAR10 images, following Chen et al. (2020a). The representations produced by this architecture have dimensionality 512. Further, for the non-linear projection head, we use a 2 layer multi-layer perceptron (MLP) with output dimensionality 128.

Contrastive Augmentations. We follow Chen et al. (2020a) and compose a random resized crop, a random horizontal flip, random colour jitter, and random grayscale for the augmentation. These augmentations make up the contrastive augmentation set $\mathcal{A}$. We finally normalise the images to have mean 0 and standard deviation 1 per channel, as is standard.

Hyperparameters. We use a $\mathcal{N}(0, \frac{1}{\tau_p^2})$ prior over the linear parameters θ^t, and tune τ_p for each dataset. As such, τ can be understood as the prior temperature. We use $\tau_p = 0.65$ for CIFAR10 and $\tau_p = 0.6$ for CIFAR100. We use weight decay $1e-6$ for the base encoder and projection head parameters.

Variational Distribution. We parameterize the temperature and noise scale using the log of their values. That is, we have $\sigma = \exp \tilde{\sigma}$.

Optimisation Details. We use the LARS optimiser (You et al., 2017), with batch size 1000 and momentum 0.9. We train for 1000 epochs, using a linear warmup cosine annealing learning rate schedule. The warmup starting learning rate for the base encoder parameters is $1e-3$ with a maximum learning rate of 0.6. For the variational parameters, the maximum learning rate is $1e-3$, which we found to be important for the stability of the algorithm.

Laplace Evaluation Protocol. We find a point estimate for θ^t found using the standard linear evaluation protocol (i.e., SGD training). We then apply a post-hoc Laplace approximation using the generalised Gauss-Newton approximation to the Hessian using the `laplace` library (Daxberger et al., 2021). For CIFAR10, we use a full covariance approximation and for CIFAR100 we use a Kroenker-factorised approximation for the Hessian of the last layer weights and biases. We tune the prior precision by maximising the likelihood of a validation set. For predictions, we use the (extended) probit approximation. These choices follow recommendations from Daxberger et al. (2021).

B.1.3 Self-Supervised BNNs*

Self-Supervised BNNs* additionally leverage labelled data when training the base encoder by including an additional ELBO $\tilde{\mathcal{L}}^t$ that depends on $\mathcal{D}^t$.

We use mean-field variational inference over θ^t with a Gaussian approximate posterior and Flipout (Wen et al., 2018). We use the implementation from Krishnan et al. (2022), and temper by setting $\beta = 1/|\theta^t|$, meaning we use the average per-parameter KL divergence. α is a hyperparameter that controls the relative weighting between the generated contrastive task datasets and the observed label data, and is tuned. For CIFAR10, We use $\alpha = 5 \cdot 10^{-5}$ when we have fewer than 100 labels, and $\alpha = 5 \cdot 10^{-3}$ otherwise. For CIFAR100, We use $\alpha = 5 \cdot 10^{-5}$ when we have fewer than 1000 labels, and $\alpha = 5 \cdot 10^{-3}$ otherwise. For $p(\theta^t)$, we use a $\mathcal{N}(0, 1)$ prior. For downstream evaluation, we use the Laplace evaluation protocol.

All other details follow Self-Supervised BNNs.

B.1.4 BNN Baselines

All baselines use the same ResNet-18 architecture, which was modified for the image size used in the CIFAR image datasets. The baselines we considered were chosen because they are all compatible with batch normalisation, which is included in the base architecture. We provide further details about the baselines below.

MAP. For the maximum-a-posterior network, we use the Adam optimiser with learning rate 10^{-3}, default weight decay, and batch size 1000. We train for a minimum of 25 epochs and a maximum of 300 epochs, terminating training early if the validation loss increases for 3 epochs in a row.

Last-Layer Laplace. For the Last-Layer Laplace baseline, we perform a post-hoc Laplace approximation to a MAP network trained using the protocol above. We use the same settings as for the self-supervised BNN's Laplace evaluation.

Deep Ensemble. For the deep ensemble baseline, we train 5 MAP networks starting from different initialisations using the above protocol, and aggregate their predictions.

SWAG. For the SWAG baseline, we first a MAP network using the above protocol. We then run SGD from this solution for 10 epochs, taking 4 snapshots per epoch, and using $K = 20$ as the rank of the covariance matrix. We choose the SWAG learning rate per run using the validation set, and consider $10^{-2}, 10^{-3}$, and 10^{-4}.

B.2 Active Learning (§5)

We simulate a low-budget active learning setting. For each method, we use their default implementation details as outlined in this Appendix. With regards to the active learning setup, we assume that we have access to a small validation set of 50 labelled examples and are provided 50 labelled training examples. We acquire 10 examples per acquisition round up to a maximum of 500 labelled examples, which corresponds to 1% of the labels in the training set. We evaluate using the full test set. The deep ensemble and self-supervised BNNs provide epistemic uncertainty estimates, so we perform active learning by selecting the points with the highest BALD metric (Houlsby et al., 2011). For SimCLR, we acquire points using the highest predictive entropy, a commonly used baseline (Gal et al., 2017). For this experiment, we use only the CIFAR10 dataset. SimCLR and the self-supervised BNNs here are pretrained on 500 epochs, not 1000 epochs as default.

B.3 Prior Predictive Checks (§4)

BNN Prior Predictive. We use the ResNet-20-FRN architecture, which is the architecture used by Izmailov et al. (2021b). Note that this architecture does not include batch normalisation, which means the prior over parameters straightforwardly corresponds to a prior predictive distribution. We use a $\mathcal{N}(0, \frac{1}{5})$ prior over all network weights, again following Izmailov et al. (2021b), and sample from the prior predictive using 8192 Monte Carlo samples.

Self-Supervised Prior Predictives. We primarily follow the details outlined earlier, except we use the same ResNet-20-FRN architecture as used for the BNNs and batch size of 500 rather than 1000. To sample from the prior predictive, we use Eq. (2) and have $y \sim \text{softmax}(W f_{\theta^s}(x))$, where we normalise the representations produced by the base encoder to have zero mean and unit variance, and we have $W \sim \mathcal{N}(0, 20)$, with the prior precision chosen by hand. We neglect the biases because they introduce additional variance. The prior evaluation scores are not sensitive to the prior variance choice, and are evaluated by sampling images from the validation set, which was not seen during training. We used 4096 Monte Carlo samples from the prior.

C Ablation Studies

Effect of Batch Size. We study the effect of the pre-training batch size on the performance of our self-supervised BNNs. We run one seed for 100 epochs across three different batch sizes on CIFAR10. We see in Table 3, the performance on CIFAR10 is robust to reducing the batch size. We hypothesise this is due to the noise injected during pre-training.

Table 3: Effect of pretraining batch size on self-supervised BNN.

Batch Size	**CIFAR10 Accuracy** (%)
100	0.81
500	0.81
1000	0.80

Effect of Variational Distribution. We run an ablation study changing the variational distribution mean on CIFAR10. We evaluated using one seed, training for 100 epochs only. We consider setting the mean of the variational distribution for image i, ω_i, to be: $0.5(\tilde{z}_i^A + \tilde{z}_i^B)$, $\tilde{z}_i^A$, and $\mathbf{0}$. We see that a suitable mean is required for good performance.

Table 4: Effect of the pretraining variational distribution on self-supervised BNN performance on CIFAR10. We see that some variant of our data-dependent mean is needed for good performance.

Variational Dist. Mean	**CIFAR10 Accuracy** (%)
$\mathbf{0}$	0.19
$\tilde{z}_i^A$	0.79
$0.5(\tilde{z}_i^A + \tilde{z}_i^B)$	0.80

Effect of Pretraining and Inference. To better understand the effect of the pretraining objective and the approximate inference scheme used, we performed an ablation study on CIFAR10. We considered both variants of our variational pretraining, and additionally deterministic pretraining that uses the NT-XENT loss. We also consider either using Laplace approximate inference or MAP estimation for the task parameters. Using the NT-XENT loss and MAP inference corresponds to SimCLR (Chen et al., 2020a), as widely used in the self-supervised learning community. For NT-XENT, we use $\tau = 0.45$ for CIFAR10 and $\tau = 0.3$ for CIFAR100. For these experiments, we only train for 500 epochs.

In Fig. 5, we see that incorporating the labelled data during pretraining boosts accuracy, but surprisingly decreases calibration. Relative to SimCLR, all of the self-supervised BNNs offer improved calibration at all dataset sizes. All approaches have high accuracy at low data regimes, highlighting the benefit of leveraging unlabelled data. Both deterministic pretraining and variational pretraining behave similarly, but performing approximate inference over task parameters substantially improves calibration.

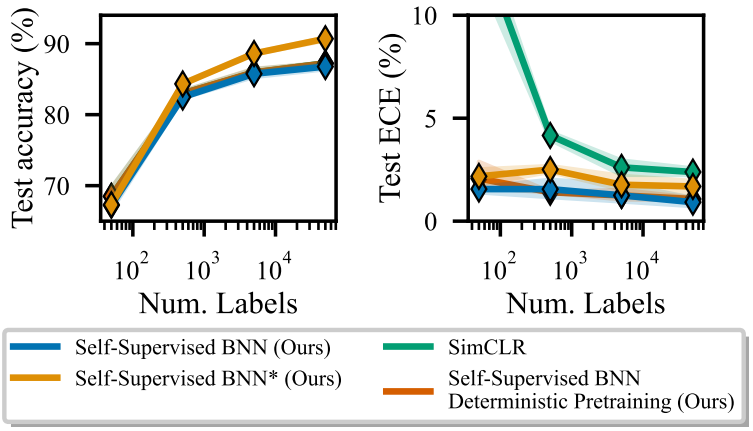

Figure 5: **Effect of Pretraining and Inference** on CIFAR10. On the left plot, red, green, and blue lines overlap. On the right plot, blue and red lines overlap. Recall that the SS BNN is performing the contrastive learning separately from and the SS BNN* jointly with the downstream task. We see that our SS BNN* slightly outperforms the other approaches in terms of accuracy and that all of our approaches yield better-calibrated uncertainties than SimCLR.

D Additional Results

We additionally report the in-distribution accuracy and expected calibration error (ECE) of different BNNs when observing different numbers of labels. These metrics, unlike the log-likelihood, are interpretable. But note that for a useful classifier, we need to have *both* accurate *and* well-calibrated predictions.[2]

In Table 5, we see that self-supervised BNNs substantially outperform conventional BNNs in terms of in-distribution accuracy. The gains are particularly large at smaller dataset sizes, precisely where improved priors are expected to make the biggest difference. Moreover, in terms of calibration, they consistently offer well-calibrated uncertainty estimates. Even though LL Laplace offers well-calibrated uncertainty estimates at a low numbers of labels, the predictions are much less accurate than self-supervised BNNs. We also see that incorporating labelled data during pretraining or ensembling self-supervised BNNs boosts accuracy, but surprisingly can harm calibration. Curiously, we find that the calibration of ensemble methods also sometimes worsens as we condition on more data.

[2]There are perfectly calibrated but useless classifiers, e.g., if 70% of examples are class A and 30% are class B, predicting $p(\text{class A}) = 0.7$ on every input achieves perfect ECE but does not discriminate between examples at all.

Table 5: **Bayesian Neural Network Predictive Performance**. Here, relative to the main results table, we report the accuracy and expected calibration error of different methods separately. Recall that the SS BNN is performing the contrastive learning separately from and the SS BNN* jointly with the downstream task.

Dataset	# labelled points	↑ **Accuracy (%)**							
		MAP	LL Laplace	SWAG	SS BNN	SS BNN*	Deep Ensemble	SS BNN Ensemble	SS BNN* Ensemble
CIFAR10	50	14.6	14.9	14.8	67.5	**68.3**	19.0	70.6	**70.8**
	500	31.2	32.4	31.5	84.9	**86.2**	39.2	86.9	**87.9**
	5000	56.5	53.4	66.4	87.6	**88.9**	72.1	89.4	**90.9**
	50000	83.4	85.7	90.5	88.8	**91.7**	91.8	91.0	**93.5**
CIFAR100	50	3.56	3.62	3.11	**14.6**	14.5	3.76	**15.6**	**15.6**
	500	6.97	6.05	7.69	38.4	**39.1**	9.17	40.4	**41.5**
	5000	22.7	21.2	25.6	54.5	**56.1**	28.2	58.8	**62.6**
	50000	60.5	59.6	67.6	60.0	**69.1**	70.7	66.0	**74.3**
CIFAR10 to CIFAR10-C **(OOD Generalisation)**	50	13.6	13.8	14.0	45.0	**45.5**	16.9	47.9	**48.3**
	500	26.7	27.2	26.2	58.0	**59.0**	32.0	60.8	**62.0**
	5000	42.1	43.2	50.5	59.8	**60.3**	55.1	63.1	**64.1**
	50000	59.2	62.1	**69.0**	60.5	62.9	**70.1**	64.3	67.0
		↓ **Expected Calibration Error (ECE; %)**							
CIFAR10	50	65.9	2.69	6.16	**1.65**	1.88	35.3	1.59	**1.33**
	500	30.0	3.19	13.4	**1.76**	2.62	10.6	**2.15**	2.41
	5000	21.0	4.48	9.95	**1.23**	2.14	3.78	**2.40**	3.00
	50000	8.43	1.50	2.78	**0.91**	1.18	3.71	2.57	**1.75**
CIFAR100	50	61.6	-	17.5	11.6	11.5	38.5	12.7	**12.6**
	500	25.3	**1.82**	22.0	2.15	2.86	14.9	**1.67**	2.57
	5000	32.4	**1.4**	16.35	1.51	7.87	4.35	**3.10**	12.5
	50000	17.8	**1.55**	8.89	1.70	2.58	**4.65**	5.68	6.05
CIFAR10 to CIFAR10-C **(OOD Generalisation)**	50	67.6	**3.95**	6.91	9.28	9.15	37.3	7.70	**7.07**
	500	33.7	**7.36**	18.0	8.44	8.86	15.5	7.30	**6.64**
	5000	30.4	9.23	19.1	**7.21**	7.76	6.85	**5.46**	6.97
	50000	24.1	11.1	13.6	**9.37**	12.1	**5.97**	6.98	6.74