Abstract

We propose Bayesian hypernetworks: a framework for approximate Bayesian inference in neural networks. A Bayesian hypernetwork, $h$, is a neural network which learns to transform a simple noise distribution, $p(\epsilon) = \mathcal{N}(0, I)$, to a distribution $q(\theta) = q(h(\epsilon))$ over the parameters $\theta$ of another neural network (the "primary network"). We train $q$ with variational inference, using an invertible $h$ to enable efficient estimation of the variational lower bound on the posterior $p(\theta|D)$ via sampling. In contrast to most methods for Bayesian deep learning, Bayesian hypernets can represent a complex multimodal approximate posterior with correlations between parameters, while enabling cheap i.i.d. sampling of $q(\theta)$. We demonstrate these qualitative advantages of Bayesian hypernets, which also achieve competitive performance on a suite of tasks that demonstrate the advantage of estimating model uncertainty, including active learning and anomaly detection.

1 Introduction

Simple and powerful techniques for Bayesian inference of deep neural networks’ (DNNs) parameters have the potential to dramatically increase the scope of applications for deep learning techniques. In real-world applications, unanticipated mistakes may be costly and dangerous, whereas anticipating mistakes allows an agent to seek human guidance (as in active learning), or engage some safe default behavior (such as shutting down).

DNNs are typically trained to find the single most likely value of the parameters, but this approach neglects uncertainty about which parameters are in fact the best ("parameter uncertainty"), which may translate into higher predictive uncertainty when likely parameter values yield highly confident but contradictory predictions. Conversely, Bayesian DNNs model the full posterior distribution of a models parameters given the data, and can thus provide better calibrated confidence estimates, with corresponding safety benefits [Gal and Ghahramani, 2016][Amodei et al., 2016].

Techniques for Bayesian DNNs are an active research topic. Most recent work focuses on variational inference (Blundell et al., 2015; Gal, 2016), and restricts the variational posterior to a simple family of distributions, for instance a factorial Gaussian (Blundell et al., 2015). Unfortunately, from a safety perspective, variational approximations tend to underestimate uncertainty, because the direction of the KL-divergence $KL(q||p)$ more strongly penalizes $q$ for putting mass where $p$ has low mass than for

1 While Bayesian deep learning may capture parameter uncertainty, most approaches, including ours, emphatically do not capture uncertainty about which model is correct (e.g. neural net vs. decision tree, etc.). Parameter uncertainty is often called "model uncertainty" in the literature, but we prefer our terminology because it emphasizes the existence of further uncertainty about model specification.
not putting mass where $p$ has high mass. This problem can be exacerbated by using a restricted family of posterior distribution; for instance a unimodal approximate posterior will generally only capture a single mode of the true posterior. With this in mind, we propose learning an extremely flexible and powerful posterior, parametrized by a DNN, $h$, which we refer to as a Bayesian hypernetwork, in reference to Ha et al. (2017).

A Bayesian hypernetwork (BHN) takes random noise $\epsilon \sim \mathcal{N}(0, 1)$ as input and outputs a sample from the approximate posterior $q(\theta)$ for another DNN of interest (the “primary network”). The key insight for building such a model is the use of an invertible hypernet, which enables Monte Carlo estimation of the entropy term $\log q(\theta)$ in the variational inference training objective.

We begin the paper by reviewing previous work on Bayesian DNNs, and explaining the necessary components of our approach (Section 2). Then we explain how to compose these techniques to yield Bayesian Hypernets, the design choices we make in this work, and techniques for stabilizing training (Section 3). Finally, we present experiments which validate the expressivity of BHNs, and demonstrate their competitive performance across several tasks (Section 4).

2 Related Work

We begin with an overview of prior work on Bayesian neural networks in Section 2.1 before discussing the specific components of our technique in Sections 2.2 and 2.3.

2.1 Bayesian DNNs

Bayesian DNNs have been studied since the 1990s (Neal, 1996; MacKay, 1994). For a thorough review, see Gal (2016). Broadly speaking, existing methods either 1) use Markov Chain Monte Carlo (Welling and Teh, 2011; Neal, 1996) or 2) directly learn an approximate posterior distribution using (stochastic) variational inference (Graves, 2011; Gal and Ghahramani, 2016; Salimans et al., 2015; Blundell et al., 2015), expectation propagation (Hernandez-Lobato and Adams, 2015; Soudry et al., 2014), or $\alpha$-divergences (Li and Gal, 2017). We focus here on the most popular approach: variational inference.

Notable recent work in this area includes Gal and Ghahramani (2016) and Kingma et al. (2015), who interpret the popular dropout (Srivastava et al., 2014) algorithm as a variational inference method (“MC dropout”). This has the advantages of being simple to implement and allowing cheap samples from $q(\theta)$, but yields a unimodal approximate posterior, and does not allow arbitrary dependencies between the parameters.

The other important points of reference for our work are Bayes by Backprop (BbB) (Blundell et al., 2015), and multiplicative normalizing flows (Louizos and Welling, 2017). Bayes by Backprop can be can be viewed as a trivial instance of a Bayesian hypernet, where the hypernetwork only performs an element-wise scale and shift of the noise (yielding a factorial Gaussian distribution).

More similar is the work of Louizos and Welling (2017), who propose and dismiss BHNs due to the issues of scaling BHNs to large primary networks, which we address. Instead, in their work, they use a hypernet to generate scaling factors, $z$ on the means $\mu$ of a factorial Gaussian distribution. Because $z$ can follow a complicated distribution, this forms a highly flexible approximate posterior: $q(\theta) = \int q(\theta|z)q(z)dz$. However, this approach also requires them to introduce an auxiliary inference network to approximate $p(z|\theta)$ in order to estimate the entropy term of the variational lower bound, resulting in lower bound on the variational lower bound.

Finally, the variational autoencoder (VAE) (Jimenez Rezende et al., 2014; Kingma and Welling, 2013) family of generative models is likely the best known application of variational inference in DNNs, but note that the VAE is not a Bayesian DNN in our sense. VAEs approximate the posterior over latent variables, given a datapoint; Bayesian DNNs approximate the posterior over model parameters, given a dataset.
2.2 Hypernetworks and Conditional Batch Normalization

A hypernetwork (Ha et al., 2017; Brabandere et al., 2016; Bertinetto et al., 2016) is a neural net that outputs parameters of another neural net (the "primary network"). The hypernet and primary net together form a single model which can be trained by backpropagation. By fixing the parameters of the hypernet, and varying its inputs, one can generate different primary net parameters. In previous works, inputs to the hypernetwork were either functions of the input to the primary net (Bertinetto et al., 2016) or its activations (Brabandere et al., 2016; Ha et al., 2017), or were themselves learned parameters (Ha et al., 2017). Our work and Louizos and Welling (2017) are the first we know of to use random noise as input to a hypernet.

The number of parameters of a DNN scales approximately quadratically in the number of units per layer, so naively parametrizing a large primary net would require an impractically large hypernet. Efficient parametrization of hypernets, however, can actually compress the total size of a network (Ha et al., 2017). For a simple illustration, factoring a 100x100 weight matrix, $W$, into as $W = E_{100 \times 7} H_{7 \times 100}$ can be viewed as using a simple hypernet ($H$) to compress the rows of $W$ into 7-dimensional encodings ($E$).

Conditional Batch Norm (CBN) (Dumoulin et al., 2016), and the closely related Conditional Instance Normalization (CIN) (Huang and Belongie, 2017; Ulyanov et al., 2016), and Feature-wise Linear Modulation (FiLM) (Perez et al., 2017; Kirkpatrick et al., 2016) methods can be viewed as specific forms of a hypernet. In these works, the weights of the primary net are parametrized directly, and the hypernet only outputs scale ($\gamma$) and shift ($\beta$) parameters for every neurons; this can be viewed as selecting which features are significant (scaling) or present (shifting). These works condition the primary net’s behaviour on information about the task, in the form of task IDs (Dumoulin et al., 2016; Kirkpatrick et al., 2016), natural language (de Vries et al., 2017; Perez et al., 2017), or task-exemplars (Huang and Belongie, 2017). In our work, we employ the related technique of weight normalization (Salimans and Kingma, 2016), which normalizes the input weights for every neuron and introduces a separate parameter, $g$, for their scale.

2.3 Invertible Generative Models

Our proposed Bayesian hypernetworks employ a differentiable directed generator network (DDGN) (Goodfellow et al., 2016) as a generative model of the primary net parameters. DDGNs use a neural net to transform simple noise (most commonly isotropic Gaussian) into samples from a complex distribution, and are a common component of modern deep generative models such as variational autoencoders (VAEs) (Kingma and Welling, 2013; Jimenez Rezende et al., 2014) and generative adversarial networks (GANs) (Goodfellow et al., 2014; Goodfellow, 2017).

We take advantage of techniques for invertible DDGNs developed in several recent works on generative modeling (Dinh et al., 2014, 2016) and variational inference of latent variables (Rezende and Mohamed, 2015; Kingma et al., 2016). Training these models uses the change of variables formula, which involves computing the log-determinant of the inverse Jacobian of the generator network. This computation involves a potentially costly matrix inversion, and these works propose innovative architectures which reduce the cost but can still express complicated deformations. In particular, RealNVP (Dinh et al., 2014, 2016), the method we use, employs a series of "coupling layers" to introduce dependencies between dimensions of the input.

3 Methods

We now describe how variational inference is applied to Bayesian deep nets (Section 3.1), and how we compose the methods described in Sections 2.2 and 2.3 to produce Bayesian hypernets (Section 3.2).

---

2 The name “hypernetwork” comes from Ha et al. (2017), who describe the general hypernet framework, but applications of this idea in convolutional networks were previously explored by Brabandere et al. (2016) and Bertinetto et al. (2016).

3 Scaling is exactly quadratic when all layers have the same number of units, as in invertible generative models.
3.1 Variational Inference

In variational inference, the goal is to maximize a lower bound on the marginal likelihood of the data, \( \log p(D) \). This can involve both estimating parameters of a statistical model, and approximating the posterior distribution over unobserved random variables (which may themselves also be parameters, e.g. as in the case of Bayesian DNNs). Let \( \theta \) be parameters given the Bayesian treatment as random variables, \( D \) a training set of observed data, and \( q(\theta) \) a learned approximation to the true posterior \( p(\theta|D) \). Since the KL divergence is always non-negative, we have, for any \( q(\theta) \):

\[
\log p(D) = KL(q(\theta)||p(\theta|D)) + \mathbb{E}_q \log p(D|\theta) + \log p(\theta) - \log q(\theta) \tag{1}
\]

\[
\log p(D) \geq \mathbb{E}_q \log p(D|\theta) + \log p(\theta) - \log q(\theta) \tag{2}
\]

The right hand side of Equation \(2\) is the evidence lower bound, or "ELBO".

The above derivation applies to any statistical model and any dataset. In our experiments, we focus on modeling conditional likelihoods \( p(D) = p(Y|X) \). Using the conditional independence assumption, we can further decompose \( \log p(D|\theta) \approx \log p(Y|X, \theta) \) as \( \sum_{i=1}^n \log p(y_i|x_i, \theta) \), and apply stochastic gradient methods for optimization.

3.1.1 Variational Inference for Deep Networks

Computing this expectation (Equation \(2\)) is generally intractable for deep nets, but it can be estimated by Monte Carlo sampling. For a given value of \( \theta \), \( \log p(D|\theta) \) and \( \log(\theta) \) can be computed and differentiated exactly as in a non-Bayesian DNN, allowing training by backpropagation. The entropy term \( \mathbb{E}_q \log q(\theta) \) is also straightforward to evaluate for simple families of approximate posteriors such as Gaussians. Similarly, we can use Monte Carlo to estimate the likelihood of a test data-point under the predictive posterior \( q(\theta) \):

\[
p(Y = y|X = x, D) = \int p(Y = y|X = x, \theta)p(\theta|D)d\theta \approx \frac{1}{N} \sum_{i=1}^N p(Y = y|X = x, \theta_i) ; \; \theta_i \sim q(\theta) \tag{3}
\]

3.2 Bayesian Hypernets

Bayesian hypernets (BHNs) express a flexible \( q(\theta) \) by using a DDGN \( h \) to transform random noise \( \epsilon \sim \mathcal{N}(0, 1) \) into independent samples from \( q(\theta) \). This makes it cheap to compute Monte Carlo estimations of expectations with respect to \( q \); these include the ELBO, and its derivatives, which can be backpropagated to train \( h \).

This means BHNs can be trained (Equation \(2\)) and evaluated (Equation \(4\)) via samples of \( q(\theta) \), which makes expressing \( q(\theta) \) as a generative model a natural strategy. However, while DDGNs are convenient to sample from, computing the entropy term (\( \mathbb{E}_q \log q(\theta) \)) of the ELBO additionally requires evaluating the likelihood of generated samples, and most popular DDGNs (such as VAEs and GANs) do not provide a convenient way of doing so \(^4\). In general, these models can be many-to-one mappings, and computing the likelihood of a given parameter value requires integrating over the latent noise variables \( \epsilon \):

\[
q(\theta) = \int q(\theta; h(\epsilon))q(\epsilon)d\epsilon \tag{5}
\]

To avoid this issue, we use an invertible \( h \), allowing us to compute \( q(\theta) \) simply using the change of variables formula:

\[
q(\theta) = q_\epsilon(h^{-1}(\theta)) \left| \det \frac{\partial h^{-1}(\theta)}{\partial \epsilon} \right| \tag{6}
\]

where \( q_\epsilon \) is the distribution of \( \epsilon \).

---

\(^4\) Here we approximate the posterior distribution \( p(\theta|D) \) using the approximate posterior \( q(\theta) \). We further use \( N \) Monte Carlo samples to approximate the integral.

\(^5\) Note that the entropy term is the only thing encouraging diversity in \( q \); the other two terms of Equation \(2\) encourage the hypernet to ignore the noise inputs and deterministically output the MAP-estimate for \( \theta \).
As discussed in Section 2.3, a number of techniques have been developed for efficiently training such invertible DDGNs. While we use RealNVP (Dinh et al., 2016), Inverse Autoregressive Flows (IAF) (Kingma et al., 2016) could also be efficiently applied; this is because we only require the ability to evaluate likelihood of generated samples (not arbitrary points in the range of $h$, as in generative modeling applications, e.g. [Dinh et al. (2016)]. And this also means that we can use a lower-dimensional $\epsilon$ to generate samples along a submanifold of the entire parameter space, as detailed below.

### 3.3 Efficient Parametrization and Training of Bayesian Hypernets

In order to scale BHNs to large primary networks, we use the weight normalization reparametrization:

$$\theta = gu; \quad u \doteq \frac{v}{||v||}$$  (7)

We only output the scaling factors $g$ from the hypernet, and learn a maximum likelihood estimate of $v$. This allows us to overcome the computational limitations of naively-parametrized BHNs noted by Louizos and Welling (2017), since computation now scales linearly, instead of quadratically, in the number of primary net units. Using this parametrization restricts the family of approximate posteriors, but still allows for a high degree of multimodality and dependence between the parameters.

Since $q(\theta)$ is now a degenerate distribution (i.e. it is entirely concentrated in a lower-dimensional manifold), we cannot compute $KL(q(\theta)||p(\theta|D))$. Instead, we treat $g$ as a random variable, whose distribution induces a distribution (parametrized by $u$) over $\theta$, and simply compute $KL(q(g)||p(g|D))$.

Since the scale of $u$ is fixed, the scale of $g$ is intuitively meaningful, and we can easily "translate" commonly-used spherical prior distributions over $\theta$ into the priors over $g$.

We also employ weight normalization within the hypernet, and found this stabilizes training dramatically. Initialization plays an important role as well; we recommend initializing the hypernet weights to small values to limit the impact of noise at the beginning of training. We also improve numerical stability by clipping the outputs of the softmax to be within $(.001, .999)$.

### 4 Experiments

We perform experiments on MNIST, CIFAR10, and a 1D regression task. We first present some qualitative proof of concepts and visualizations (Section 4.1) demonstrating that Bayesian Hypernets (BHNs) can in fact learn multimodal, dependent distributions. Next, we measure the performance of BHNs quantitatively. There is no single metric for how well a model captures uncertainty; to evaluate how well BHNs capture parameter uncertainty, we perform experiments on regularization (Section 4.2), active learning (Section 4.3), and anomaly detection (Section 4.4).

Active learning and anomaly detection problems make natural use of uncertainty estimates: In anomaly detection, higher uncertainty indicates a likely anomaly. In active learning, higher uncertainty indicates a greater opportunity for learning.

Parameter uncertainty also has regularization benefits: integrating over the posterior creates an implicit ensemble. Intuitively, when the most likely hypothesis predicts "A", but the posterior places more total mass on hypotheses predicting "B", we prefer predicting "B". By improving our estimate of the posterior, we can more accurately weigh the evidence for different hypotheses.

For the hypernet architecture, we use RealNVP (Dinh et al., 2016) with 1-layer ReLU-MLP coupling functions with 200 hidden units (each). We use an isotropic standard normal prior on the weights of the network. Our baselines for comparison are Bayes by Backprop (BbB) (Blundell et al., 2015). MC dropout (MCdrop) (Gal and Ghahramani, 2016), and non-Bayesian DNN baselines (with and without dropout). We use Adam with default hyper-parameter settings (Kingma and Ba, 2014) and gradient clipping in all of our experiments. Our mini-batch size is 128, and to reduce computation, we use the same noise-sample (and thus the same primary net parameters) for all examples in a mini-batch. We experimented with independent noise, but computation is slower, and we did not notice any benefit.

---

6 This parametrization strongly resembles the "correlated" version of variational Gaussian dropout (Section 3.2 of [Kingma et al., 2015]). The only difference is that we restrict the $u$ to have norm 1.
Figure 1: Illustration of a BHN (Left) and a non-Bayesian DNN (Right) on the toy problem from Blundell et al. (2015). The solid black line shows the mean predictions while the dashed lines show 95% error bars. The red crosses are 100 examples from the training dataset. The colored lines (Left) are the predictions of primary network sampled from the BHN approximate posterior.

Figure 2: Histogram of Pearson correlation coefficient p-values (left) and random scalar parameters with line of best fit (right) for samples from a hypernet approximate posterior. We see that the hypernet posterior includes correlations between different parameters. In particular, many of the p-values of the Pearson correlation test are below .05.

4.1 Qualitative Results and Visualization

We demonstrate the behavior of the network on the toy 1D-regression problem from Blundell et al. (2015) in Figure 1. As expected, the uncertainty of the network increases away from the observed data.

Next, we demonstrate the distinctive ability of Bayesian Hypernets to learn multi-modal, dependent distributions. Figure 2 shows that BHNs do in fact learn approximate posteriors with dependence between different parameters, as measured by the Pearson correlation coefficient. Meanwhile, Figure 3 shows that BHNs can learn multimodal posteriors. For this experiment, we trained an over-parametrized linear network: \( y = a \cdot b \cdot x \) on a dataset generated as \( y = x + \epsilon \). The red curves \( a \cdot b = 1 \) represent an optimal solution to this regression problem (given sufficient data to overcome the prior), and the BHN posterior learns to capture both modes (see Figure 3).

4.2 Regularization

We now show that BHNs act as a regularizer, outperforming dropout and traditional mean field (BbB). Integrating out uncertainty in the parameters to form the predictive distribution is expected to yield superior performance and can also be viewed as a form of ensembling. Results are presented in Table 1.

For MNIST, we train an MLP with 2 hidden layers of 800 hidden units each. For CIFAR10, we train a convolutional neural net (CNN) with 4 hidden layers with \([64, 64, 128, 128] \) channels, 2x2 max
Figure 3: Learning the identity function with an overparametrized network: $y = a \cdot b \cdot x$. This parametrization results in symmetries shown by the red lines. The blue dots are samples drawn from the learned posterior $q(\theta)$. The network assigns significant mass to both $a = b = 1$ and $a = b = -1$.

| MNIST 50000 | CIFAR10 50000 |
|-------------|----------------|
| No. of Coupling Layers | Test Accuracy | No. of Coupling Layers | Test Accuracy |
| 0 | 98.28% (98.01%*) | 0 | 67.83% |
| 2 | 98.39% | 4 | 74.77% |
| 4 | 98.47% | 8 | 74.90% |
| 6 | 98.59% | dropout | 74.08% |
| 8 | 98.63% | MLE | 72.75% |
| dropout | 98.73% |

Table 1: Generalization results on MNIST and CIFAR10 for BHNs with different numbers of coupling layers, and comparison methods (dropout / maximum likelihood (MLE)). * Bayes-by-backprop (Blundell et al., 2015) models each parameter as an independent Gaussian, which is equivalent to using a hypernet with 0 coupling layers. We achieved a better result outputting a distribution over scaling factors (only).

pooling applied after the second and the fourth layers, and filter size of 3. The CNN also includes a fully connected layer of 512 before the 10-class softmax layer.

In our experiments, we find that BHNs can perform on par with dropout on full datasets of MNIST and CIFAR10; furthermore, increasing the complexity of the posterior by adding more coupling layers improves performance, especially compared with models with 0 coupling layers, which cannot model dependencies between the parameters.

We also evaluate on a subset of 5000 MNIST examples; results are presented in Table 2. We parameterize the MLP with 800 and 1200 hidden nodes, and our method outperforms dropout in both cases.

4.3 Active Learning

We now turn to active learning, where we compare to the MNIST experiments of Gal et al. (2017), replicating their architecture and training procedure. Briefly, they use an initial dataset of 20 examples (2 from each class), and acquire 10 new examples at a time, training for 50 epochs between each acquisition. While Gal et al. (2017) re-initialize the network after every acquisition, we found that "warm-starting" from the current learned parameters was essential for good performance with BHNs, although it’s likely that longer training or better initialization schemes could perform the same role. Overall, warm-started BHNs suffered at the beginning of training, but outperformed all other methods for moderate to large numbers of acquisitions.
Table 2: Generalization results on subset (5000 training data) of MNIST. (A) MLP with 800 hidden nodes. (B) MLP with 1200 hidden nodes.

| No. of Coupling Layers | Test Accuracy | No. of Coupling Layers | Test Accuracy |
|------------------------|---------------|------------------------|---------------|
| 0                      | 92.06%        | 0                      | 90.91%        |
| 8                      | 94.25%        | 8                      | 96.27%        |
| 12                     | 96.16%        | 12                     | 96.51%        |
| dropout                | 95.58%        | dropout                | 95.52%        |

Figure 4: Active learning: Bayesian Hypernets outperform other approaches after sufficient acquisitions when warm-starting (left), for both random acquisition function (top) and BALD acquisition function (bottom). Warm-starting improves stability for all methods, but can hurt performance for other approaches, compared with randomly re-initializing parameters as in Gal et al. (2017) (right). We also note that the baseline model (no dropout) is competitive with MCdropout, and outperforms the Dropout baseline used by Gal et al. (2017).

4.4 Anomaly Detection

For anomaly detection, we take Hendrycks and Gimpel (2016) as a starting point, and perform the same suite of MNIST experiments, evaluating the ability of networks to determine whether an input came from their training distribution (“Out of distribution detection”).

Hendrycks and Gimpel (2016) found that the confidence expressed in the softmax probabilities of a (non-Bayesian) DNN trained on a single dataset can actually provide a good signal for both of these detection problems. We demonstrate that Bayesian DNNs can outperform their non-Bayesian counterparts.

Footnote: For the deterministic baseline, the value of the BALD acquisition function is always zero, and so acquisitions should be random, but due to numerical instability this is not the case in our implementation; surprisingly, we found the BALD values our implementation computes provide a better-than-random acquisition function (compare the blue line in the top and bottom plots).
Table 3: Anomaly detection on MNIST. Since we use the same datasets as [Hendrycks and Gimpel (2016)], we have the same base error rates, and refer the reader to that work.

| Dataset   | MLP ROC | MLP P+ | MLP P- | MC dropout ROC | MC dropout P+ | MC dropout P- | BHN ROC | BHN P+ | BHN P- |
|-----------|---------|--------|--------|----------------|---------------|---------------|---------|--------|--------|
| Uniform   | 96.99   | 97.99  | 94.71  | 98.63          | 98.97         | 99.27         | 98.52   |        |        |
| OmniGlot  | 94.92   | 95.63  | 93.85  | 96.44          | 96.89         | 95.56         | 93.64   |        |        |
| CIFARbw   | 95.55   | 96.47  | 93.72  | 98.39          | 98.97         | 98.25         | 97.14   |        |        |
| Gaussian  | 87.7    | 87.66  | 88.05  | 98.7           | 98.11         | 98.49         | 96.86   |        |        |
| notMNIST  | 81.12   | 97.56  | 39.7   | 98.53          | 90.07         | 98.51         | 56.59   |        |        |

Just as in active learning, in anomaly detection, we use MC to estimate the predictive posterior, and use this to score datapoints. For active learning, we would generally like to acquire points where there is higher uncertainty. In a well-calibrated model, these points are also likely to be challenging or anomalous examples, and thus acquisition functions from the active learning literature are good candidates for scoring anomalies.

We consider all of the acquisition functions listed in [Gal et al. (2017)] as possible scores for the AOPR and AOROC metrics, but found that the maximum confidence of the softmax probabilities (i.e. ”variation ratio”) acquisition function used by [Hendrycks and Gimpel (2016)] gave the best performance. Both BHN and MCdropout achieve significant performance gains over the non-Bayesian baseline, and MCdropout performs significantly better than BHN in this task. Results are presented in Table 3.

Second, we follow the same experimental setup, using all the acquisition functions, and exclude one class in the training set of MNIST at a time. We take the excluded class of the training data as out-of-distribution samples. The result is presented in Table 4. This experiment shows the benefit of using scores that reflect dispersion in the posterior samples (such as mean standard deviation and BALD value) in Bayesian DNNs.

5 Conclusion

We introduce Bayesian hypernets (BHNs), a new method for variational Bayesian deep learning which uses an invertible hypernetwork as a generative model of parameters. BHNs feature efficient training and sampling, and can express complicated multimodal distributions, thereby addressing issues of overconfidence present in simpler variational approximations. We validate these properties of BHNs in toy experiments, and present a method of parametrizing BHNs which allows them to scale successfully to real tasks which demonstrate the benefits of effectively modeling uncertainty, and achieve competitive performance.

Going forward, we will explore other methods of parametrizing BHNs, such as other approaches to generating lower-dimensional posteriors (e.g. via random projections). Using another hypernet to reparametrize and compress the primary net would allow BHNs to provide a full posterior over the parameters of larger primary nets.

Another idea is to use the same hypernet to output different subsets of the primary net parameters. For instance, following [Ha et al. (2017)], we could use the same $h$ to generate different filters in a convolutional architecture (e.g. by letting the biases in the layers of $h$ be different for each filter). This could provide $h$ with a stronger training signal and might increase the speed of learning, since its parameters would get error gradients from multiple sources in a single forward pass, although this would come at the expense of decreasing the expressivity of the approximate posterior. Using different sampled $\epsilon$ for the different filters would yield a full posterior over all the weights, at the cost of losing dependencies between filters’ parameter distributions.
Table 4: Anomaly detection on MNIST with unseen classes. The first column indicates the missing class label in the training set. Top-most block: ROC score; middle: positive precision-recall; bottom: negative precision-recall.

| Variation Ratio | Mean std | BALD |
|-----------------|---------|------|
| MLP dropout     | BHN 4   | BHN 8 | dropout | BHN 4 | BHN 8 |
| 0               | 95.52   | 97.44 | 96.62 | 96.46 | 98.34 | 97.90 |
| 1               | 96.70   | 96.60 | 96.46 | 94.01 | 94.86 | 96.62 |
| 2               | 92.83   | 93.77 | 92.99 | 93.47 | 96.02 | 93.03 |
| 3               | 93.03   | 93.11 | 95.03 | 95.34 | 93.65 | 96.62 |
| 4               | 88.58   | 87.75 | 81.19 | 89.45 | 89.73 | 81.31 |
| 5               | 88.53   | 94.66 | 93.20 | 89.75 | 95.95 | 93.08 |
| 6               | 95.40   | 96.33 | 93.67 | 94.69 | 96.02 | 93.80 |
| 7               | 92.46   | 96.61 | 95.08 | 97.08 | 94.68 | 92.82 |
| 8               | 96.35   | 96.85 | 97.67 | 97.96 | 97.37 | 95.48 |
| 9               | 96.70   | 94.60 | 96.62 | 96.46 | 96.42 | 96.10 |
|                | 95.45   | 97.89 | 96.59 | 96.55 | 95.84 | 96.37 |

References

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. (2016). Concrete problems in AI safety. *CoRR*, abs/1606.06565.

Bertinetto, L., Henriques, J. F., Valmadre, J., Torr, P. H. S., and Vedaldi, A. (2016). Learning feed-forward one-shot learners. *CoRR*, abs/1606.05233.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1613–1622.

Brabandere, B. D., Jia, X., Tuytelaars, T., and Gool, L. V. (2016). Dynamic filter networks. *CoRR*, abs/1605.09673.

de Vries, H., Strub, F., Mary, J., Larochelle, H., Pietquin, O., and Courville, A. C. (2017). Modulating early visual processing by language. *CoRR*, abs/1707.00683.

Dinh, L., Krueger, D., and Bengio, Y. (2014). NICE: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using real NVP. *arXiv preprint arXiv:1605.08803*.

Dumoulin, V., Shlens, J., and Kudlur, M. (2016). A learned representation for artistic style. *CoRR*, abs/1610.07629.

Gal, Y. (2016). Uncertainty in deep learning.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059.
Gal, Y., Islam, R., and Ghahramani, Z. (2017). Deep Bayesian Active Learning with Image Data.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning. MIT Press. [http://www.deeplearningbook.org]

Goodfellow, I. J. (2017). NIPS 2016 tutorial: Generative adversarial networks. CoRR, abs/1701.00160.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. (2014). Generative adversarial nets. In Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada, pages 2672–2680.

Graves, A. (2011). Practical variational inference for neural networks. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q., editors, Advances in Neural Information Processing Systems 24, pages 2348–2356. Curran Associates, Inc.

Ha, D., Dai, A., and Le, Q. V. (2017). Hypernetworks.

Hendrycks, D. and Gimpel, K. (2016). A baseline for detecting misclassified and out-of-distribution examples in neural networks. arXiv preprint arXiv:1610.02136.

Hernandez-Lobato, J. M. and Adams, R. (2015). Probabilistic backpropagation for scalable learning of Bayesian neural networks. In Proceedings of The 32nd International Conference on Machine Learning, pages 1861–1869.

Huang, X. and Belongie, S. J. (2017). Arbitrary style transfer in real-time with adaptive instance normalization. CoRR, abs/1703.06868.

Jimenez Rezende, D., Mohamed, S., and Wierstra, D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. ArXiv e-prints.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. CoRR, abs/1412.6980.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improved variational inference with inverse autoregressive flow. In Advances in Neural Information Processing Systems, pages 4743–4751.

Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational Dropout and the Local Reparameterization Trick. ArXiv e-prints.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational Bayes. arXiv preprint arXiv:1312.6114.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N. C., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2016). Overcoming catastrophic forgetting in neural networks. CoRR, abs/1612.00796.

Li, Y. and Gal, Y. (2017). Dropout inference in bayesian neural networks with alpha-divergences. arXiv preprint arXiv:1703.02914.

Louizos, C. and Welling, M. (2017). Multiplicative Normalizing Flows for Variational Bayesian Neural Networks. ArXiv e-prints.

MacKay, D. J. (1994). Bayesian neural networks and density networks. In Nuclear Instruments and Methods in Physics Research, A, pages 73–80.

Neal, R. M. (1996). Bayesian Learning for Neural Networks. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Perez, E., de Vries, H., Strub, F., Dumoulin, V., and Courville, A. C. (2017). Learning visual reasoning without strong priors. CoRR, abs/1707.03017.

Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In Proceedings of The 32nd International Conference on Machine Learning, pages 1530–1538.

Salimans, T., Kingma, D., and Welling, M. (2015). Markov chain Monte Carlo and variational inference: Bridging the gap. In Proceedings of the 32nd International Conference on Machine Learning (ICML-15), pages 1218–1226.

Salimans, T. and Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. CoRR, abs/1602.07868.
Soudry, D., Hubara, I., and Meir, R. (2014). Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In Advances in Neural Information Processing Systems, pages 963–971.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res., 15(1):1929–1958.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. S. (2016). Instance normalization: The missing ingredient for fast stylization. CoRR, abs/1607.08022.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics.