Adversarial Distillation of Bayesian Neural Network Posteriors

Kuan-Chieh Wang [1 2] **Paul Vicol** [1 2] **James Lucas** [1 2] **Li Gu** [1] **Roger Grosse** [1 2] **Richard Zemel** [1 2]

Abstract

Bayesian neural networks (BNNs) allow us to reason about uncertainty in a more principled way. Stochastic Gradient Langevin Dynamics (SGLD) enables efficient BNN learning by drawing samples from the BNN posterior using mini-batches. However, SGLD and its extensions require storage of many copies of the model parameters, a potentially prohibitive cost, especially for large neural networks. We propose a framework, Adversarial Posterior Distillation, to distill the SGLD samples using a Generative Adversarial Network (GAN). At test-time, samples are generated by the GAN. We show that this distillation framework incurs no loss in performance on recent BNN applications including anomaly detection, active learning, and defense against adversarial attacks. By construction, our framework not only distills the Bayesian predictive distribution, but the posterior itself. This allows one to compute quantities such as the *approximate model variance*, which is useful in downstream tasks. To our knowledge, these are the first results applying MCMC-based BNNs to the aforementioned downstream applications.

1. Introduction

Neural Networks (NNs) are often viewed as powerful black-box systems whose behaviors are difficult to interpret and control. Despite significant progress made on supervised learning with Stochastic Gradient Descent (SGD), users are still apprehensive when applying NNs in safety-critical systems. Moreover, recent work has shown that modern neural networks can be miscalibrated and may be over-confident about their predictions (Guo et al., 2017). The overarching motivation of this research is to have reliable estimates of uncertainty when making predictions with a NN.

Uncertainty is important in many scenarios. For example, designers of autonomous cars might want passengers to take control when the system is uncertain about the scene. Uncertainty can also be used to understand a system, such as in the prediction of basketball player movements. A player's offensive skill can be gauged by the amount of uncertainty he is able to induce by his movements.

Bayesian methods provide a principled way to model uncertainty through the posterior distribution over model parameters. Most approaches for learning Bayesian Neural Networks (BNNs) (MacKay, 1992) fall into one of two categories: variational inference (VI) or Markov chain Monte-Carlo (MCMC). The disadvantage of MCMC methods is their computational cost, both in terms of time and storage, and the difficulty in evaluation. However, MCMC methods are appealing because, in the limit, they produce samples from the true posterior. In contrast, VI methods require one to choose a family of approximating distributions, and thus can only produce samples from an approximate posterior.

The main motivation of our work is to reduce the storage overhead involved in maintaining MCMC samples, and to show the usefulness of MCMC methods in modern BNN applications. We employ Generative Adversarial Networks (GANs) (Goodfellow et al., 2014a) to model the MCMC samples. This yields a parametric approximation (i.e., the generator) of the distribution of MCMC samples, that eliminates the storage overhead while providing access to the posterior at test-time. We evaluate our approach on a range of applications including classification, anomaly detection, active learning, and defense against adversarial examples, and show that the distilled samples perform as well as the original MCMC samples. We also show analysis studies that GANs are suitable for this distillation process, especially with the recent advances in stabilizing GANs.

2. Background

In this section, we provide a brief overview of BNNs and the technical background required for our study.

2.1. Uncertainty Estimates with BNNs

In the standard deterministic NN setup, we aim to optimize the network parameters θ given a loss function $\mathcal{L}$ and a dataset of input-output pairs $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ as follows:

[1]University of Toronto, Toronto, Ontario, Canada [2]Vector Institute, Toronto, Ontario, Canada. Correspondence to: Kuan-Chieh Wang <wangkua1@cs.toronto.edu>.

$$\theta = \operatorname{argmin}_{\theta'} \mathcal{L}(\theta', \mathcal{D}) \tag{1}$$

where typically the loss is the regularized negative log-likelihood:

$$\mathcal{L} = -\sum_i \log p(y_i|x_i, \theta') + \|\theta'\|_2^2 \tag{2}$$

This is equivalent to MAP estimation of a BNN with a Gaussian prior, $\theta \sim \mathcal{N}(0, I) = p(\theta)$. At test time, one uses the learned parameters to make predictions.

In contrast to a deterministic NN, where parameters are represented by a point estimate, the parameters in a BNN are represented by probability distributions. Given a prior distribution $p(\theta)$ over model parameters, the goal is to obtain the posterior $p(\theta|\mathcal{D})$. At test time, instead of using the point-estimate approximation, one needs to marginalize out the posterior. We provide a more detailed discussion in Section 3.

2.2. Stochastic Gradient Langevin Dynamics

Stochastic Gradient Langevin Dynamics (SGLD) is a pivotal work for the application of MCMC methods to BNNs (Welling & Teh, 2011). Each iteration of MCMC traditionally requires computation over the full dataset (e.g., to compute the Metropolis-Hastings acceptance ratio). Welling & Teh (2011) develop theoretical justification for learning the posterior using mini-batches of data. Given a standard classification problem with a loss function as given in Eqn. 2, the usual SGD update can be written as:

$$\Delta \theta^t = \frac{\epsilon^t}{2} \left(\nabla \log p(\theta^t) + \frac{N}{n} \sum_{i=1}^n \nabla \log p(y_i^t|x_i^t, \theta^t) \right) \tag{3}$$

where n is the mini-batch size, and superscript t denotes the update iteration. Then, the SGLD update is just:

$$\Delta \theta^t = \frac{\epsilon^t}{2} \left(\nabla \log p(\theta^t) + \frac{N}{n} \sum_{i=1}^n \nabla \log p(y_i^t|x_i^t, \theta^t) \right) + \eta^t \tag{4}$$

where $\eta^t \sim \mathcal{N}(0, \epsilon^t)$. Note that Eqn. 4 is simply Eqn. 3 with added Gaussian noise. Welling & Teh (2011) show that when the step-size ϵ^t is decayed polynomially, the process transitions from stochastic optimization to Langevin dynamics, where each update yields an unbiased sample from the posterior.

In this work, we use SGLD to obtain samples of θ. However, our framework is also compatible with other extensions to SGLD; these extensions are complementary to our method.

2.3. Generative Adversarial Networks

Generative Adversarial Networks (GANs) are an approach to generative modeling that consist of two components: a generator, G, maps random noise $z \sim \mathcal{N}(0, I)$ to approximate data samples $G(z)$; and a discriminator, D, tries to distinguish between generated data $G(z)$ and real data x. G is trained to confuse D. In our approach, the "data" are the model samples θ, hence we write θ instead of x when appropriate. The GAN objective is:

$$\min_G \max_D \mathbb{E}_{\theta \sim p(\theta|\mathcal{D})}[\log D(\theta)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{5}$$

A recent GAN formulation with empirical success uses Wasserstein distance as the loss (Arjovsky et al., 2017). In (Arjovsky et al., 2017), D is required to be Lipschitz continuous, and weight clipping is used as a crude approximation. Extensions include using a gradient penalty (Gulrajani et al., 2017), and finite differences of gradients (Wei et al., 2018) to enforce this constraint. Other formulations of the Wasserstein distance have also been studied (Salimans et al., 2018). In this work, we opt to use the WGAN with gradient penalty (WGAN-GP) (Gulrajani et al., 2017), which optimizes the following objective:

$$\mathcal{L} = \mathbb{E}_{\tilde{\theta} \sim P_g}[D(\tilde{\theta})] - \mathbb{E}_{\theta \sim P_r}[D(\theta)] + \lambda \mathbb{E}_{\hat{\theta} \sim P_{\hat{\theta}}}[(\|\nabla_{\hat{\theta}} D(\hat{\theta})\|_2 - 1)^2] \tag{6}$$

Evaluation. Despite recent advances in GANs, evaluation remains a challenge. Most research in GANs deals with image generation, for which it is difficult to quantify performance. Some metrics are based on visual quality, including the InceptionScore (Salimans et al., 2016) and unsupervised SSIM (Rosca et al., 2017). Similarly, evaluating the quality of MCMC posterior samples has long been a challenge. In this work, because our samples are network parameters, we have the opportunity to quantitatively evaluate the samples—both the original SGLD samples and the ones generated by the GAN—by applying the BNN with those parameters across a range of applications (see Section 4).

3. Method

In this section, we introduce a framework for Bayesian inference that consists of two steps: 1) obtain a set of samples from the posterior distribution over network parameters $\theta \sim p(\theta|\mathcal{D})$ using SGLD; 2) train a WGAN-GP to model the posterior samples. This process is illustrated in Figure 1. The result is a single generative model that distills the posterior distribution; this allows us to draw samples efficiently (i.e., in parallel, as opposed to traditional MCMC steps which are performed sequentially), with little storage overhead (i.e., we only need to store the parameters of a relatively small generator). We call this formulation Adversarial Posterior Distillation (APD).

The distillation process can be performed either *offline* or *online*. We outline the offline variant of APD in Algorithm 1. This requires the user to store a large number of samples prior to distillation. We further generalize this framework
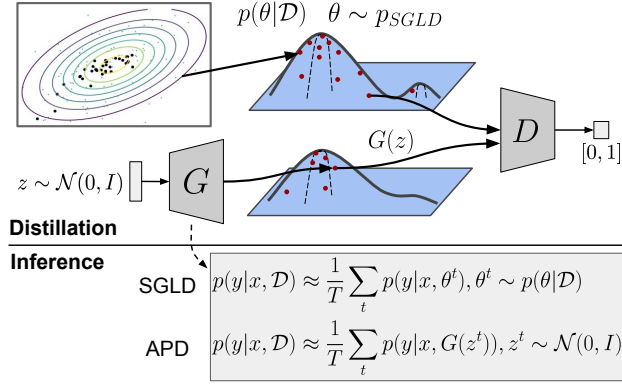
Figure 1. **APD Framework.** *Distillation:* Posterior samples are generated from the target network (top) and used to train the generator network (bottom). *Inference:* When performing inference, we sample from the generator network.

Algorithm 1 Offline APD

1: Sample $\{\theta^t\}_{t=1}^T$ using MCMC updates, where T denotes the number of updates.
2: Optimize G with WGAN-GP loss using $\{\theta^t\}_{t=1}^T$ as real data.

---

to the online setting where MCMC steps and GAN updates are interleaved (Algorithm 2). Both variants perform similarly when the buffer θ_R is reasonably large.

Details. For SGLD, instead of using the sampling scheme suggested by Welling & Teh (2011), we follow Balan et al. (2015), where the number of burn-in iterations and the sampling interval are treated as hyperparameters instead of monitoring when SGLD transitions into the sampling phase. Similarly to Balan et al. (2015), we use a fixed learning rate, as we found that this led to better performance in our experiments. It has been shown that the bias introduced by this modified version of SGLD is quantifiable (Vollmer et al., 2016).

3.1. Uncertainty Estimates

Once the generator is sufficiently trained, we can discard the SGLD posterior samples and use the generator to produce samples at test time, for use in downstream tasks. The common feature of the downstream applications we use for evaluation is that they require an estimate of uncertainty. Here we outline the uncertainty estimates used in Section 5.

For prediction using APD, we perform MC integration by drawing samples from the trained generator:

$$p(y|x, \mathcal{D}) = \mathbb{E}_{\theta|\mathcal{D}}[p(y|x, \theta)] \qquad (7)$$

$$\approx \frac{1}{T}\sum_{t=1}^T p(y|x, G(z^t)), \quad z^t \sim \mathcal{N}(0, I) \quad (8)$$

Algorithm 2 Online APD

1: Initialize $\{\theta^{0,k}\}_{k=1}^K$, the K independent BNN parameters using generated samples from G.
2: **while** not converged **do**
3:     Sample $\{\theta^{t,k}\}_{t=t}^{t+T_m}$ using MCMC updates, where T_m denotes the number of updates for all k.
4:     Add $\{\theta^{t,k}\}$ to θ_R denoting a buffer of samples to distill.
5:     Optimize G with WGAN-GP loss using θ_R as real data for T_g gradient steps.
6: **end while**

To measure uncertainty, we can compute the **entropy**, $\mathrm{H}(y|x, \mathcal{D})$ or the Bayesian Active Learning by Disagreement objective (**BALD**) (Houlsby et al., 2011):

$$\mathbb{I}(y, \theta|x, \mathcal{D}) = \mathrm{H}(y|x, \mathcal{D}) - \mathbb{E}_{\theta|\mathcal{D}}[\mathrm{H}(y|x, \theta)] \qquad (9)$$

or the **variations-ratio** (VR):

$$\mathrm{VR}(x) = 1 - \frac{1}{T}\sum_t \mathbb{1}[y^t = c^*],$$

$$c^* = \mathrm{argmax}_c \sum_t \mathbb{1}[y^t = c], \text{ where } c \text{ indexes classes}$$

$$y^t = \mathrm{argmax}_y p(y|x, G(z^t))$$

$$(10)$$

or the **approximate model variance** as defined in Feinman et al. (2017):

$$U(x) = \frac{1}{T}\sum_{t=1}^T \mathbf{p}_t^T \mathbf{p}_t - (\frac{1}{T}\sum_{t=1}^T \mathbf{p}_t)^T(\frac{1}{T}\sum_{t=1}^T \mathbf{p}_t) \quad (11)$$

where $\mathbf{p}_t$ are the stochastic vectors of class predictions.

4. Related Work

In this section, we give an overview of recent BNN learning methods and modern applications of BNNs, to motivate our experimental studies.

4.1. Learning BNNs

Variational Inference (VI). VI methods construct an approximating distribution $q(\theta) \approx p(\theta|\mathcal{D})$ and optimize to make the approximation close to the true posterior. This often involves making assumptions such as that the parameters can be fully-factorized, as $p(\theta \mid \mathcal{D}) = \prod p(w_i \mid \mathcal{D})$.

VI was first proposed for neural networks by Hinton & Van Camp (1993). Graves (2011) made VI practical by introducing a stochastic VI method with a diagonal Gaussian posterior. Graves' method uses a biased Monte Carlo estimate of the variational lower bound. Later, Blundell et al. (2015) introduced an algorithm for training BNNs called Bayes by Backprop (BBB), that uses an unbiased Monte

Carlo estimate of the lower bound, based on the reparameterization trick (Kingma & Welling, 2014; Rezende et al., 2014).

An alternative approach is Expectation Propagation (EP) (Minka, 2001), a deterministic approximation method that extends *assumed density filtering (ADF)* by iteratively refining the approximations. Probabilistic Backpropagation (PBP) (Hernández-Lobato & Adams, 2015) is a recently-introduced online extension of EP.

More recently, Louizos & Welling (2017) proposed to use the idea of a *flow* to break the simplifying mean-field assumption, but this method is still restricted by the kinds of transformations allowed by a flow (e.g., invertible). Bayesian Hypernetworks (BH) (Krueger et al., 2017) use a hypernetwork to generate shift and scale distribution over network activations, where the hypernet is restricted to be an invertible generative model.

MCMC   MCMC methods have long been used to learn BNNs (Neal, 1996). However, traditional MCMC methods require computation over the whole dataset per iteration. Since the introduction of SGLD, a suite of stochastic-gradient MCMC algorithms have been proposed (Ahn et al., 2012; 2014; Balan et al., 2015; Chen et al., 2014; Ding et al., 2014), drawing from the wealth of knowledge behind general MCMC techniques. This demonstrates the potential of MCMC methods for BNNs. However, to make use of these methods, one needs to store a sufficient number of posterior samples, which incurs significant storage overhead.

Balan et al. (2015) proposed a method to approximate the Bayesian predictive distribution using a single network. They train a student model $\mathcal{S}(y|x, w)$ to approximate the Bayesian predictive distribution $q(y|x)$ by minimizing the KL divergence $D_{\mathrm{KL}}[\mathbb{E}_{\theta|\mathcal{D}}[p(y|x, \theta)]||\mathcal{S}(y|x, w)]$. This avoids the storage cost and integration at test-time. However, the posterior $p(\theta|\mathcal{D})$ is lost at test-time, which makes this method unfit for cases where the posterior is required for other computations. In contrast, our formulation aims to distill the posterior to be sampled from for downstream use.

Li et al. (2017) studied a framework that most resembles ours. Both their work and ours use a GAN to replace MCMC samples, yet the setting and goals are different. Li et al. (2017) only provided results related to BNNs on small NNs with 50 hidden units on binary classification accuracy/loss. They instead explored additional tasks such as using their sampler to improve latent variable inference for missing-data imputation.

4.2. Recent BNN Applications

A number of interesting applications of BNNs have been studied in the context of recent VI methods. Similarly, MC dropout (Gal & Ghahramani, 2016)—which is a simple approximation that applies dropout (Srivastava et al., 2014) at test-time—has recently been used in real-world applications. Below we summarize these results by task.

Standard Classification/Regression.   Though perhaps not the most informative tasks for examining BNNs, regression loss or classification accuracy are usually reported by BNN studies. Some use 1-dimensional regression problems (Hernández-Lobato & Adams, 2015). Others use standard deep learning classification benchmarks such as MNIST (LeCun et al., 1998; Balan et al., 2015; Blundell et al., 2015; Gal & Ghahramani, 2016), or CIFAR10 (Krizhevsky & Hinton, 2009; Krueger et al., 2017; Louizos & Welling, 2017), which we adopt as well. Since good classification accuracy only requires a good point-estimate of θ, additional tasks are usually used to evaluate BNNs.

Anomaly Detection.   Anomaly detection refers to detecting *out-of-distribution* (OOD) data (such as white noise) given a BNN trained only on *in-distribution* data (e.g., MNIST). Intuitively, a good BNN should be more uncertain about OOD inputs, enabling better detection of OOD data. Hendrycks & Gimpel (2016) provide benchmarks for anomaly detection. Krueger et al. (2017) applied BH to this task, and showed that both MC dropout and BH outperform deterministic NNs. Hence, we provide detailed results using both existing MCMC-based methods and ours.

Exploration.   Blundell et al. (2015) used Thompson sampling based on BNN outputs to minimize regret in a bandit problem. Hernández-Lobato & Adams (2015) performed active learning on a 1-dimensional regression problem. Both of these settings involve fairly small datasets and models (e.g., input size on the order of 10, and NN with 1-hidden layer of width 50). More recently, Gal et al. (2017) achieved good results using MC dropout for active learning with image data (e.g., MNIST and real-world medical images).

Detecting Adversarial Attacks.   Adversarial examples (Szegedy et al., 2013) are inputs to a neural network that are designed to force misclassification. These inputs often appear normal to humans but cause the neural network to make inaccurate predictions. This raises an interesting question: can we train neural networks to detect adversarial examples? Bayesian neural networks are an obvious candidate for this task and thus they have been explored before (Feinman et al., 2017; Louizos & Welling, 2017; Rawat et al., 2017). Related to our work, Feinman et al. (2017) used MC dropout inference to detect adversarial examples and showed promising results. To the best of our knowledge, the use of SGLD for this task has not been explored.

Lastly, our work is focused on distilling BNNs and study-

ing modern downstream applications. To the best of our knowledge, our work is the first to provide results for MCMC-based BNNs for these applications.

5. Experiments

In this section, we show that APD performs as well as the true SGLD samples in terms of classification accuracy, and on the recent anomaly detection benchmarks provided by Hendrycks & Gimpel (2016). On other difficult tasks— active learning and defense against attacks—we show that SGLD does at least as well as MC dropout (Gal & Ghahramani, 2016). APD did not match SGLD, but did perform as well as MC dropout in each case. Lastly, ablation studies show that distilling the posterior is a challenging task, and that recent advances in GANs improve APD. [1]

5.1. Toy 2D Classification

We first validated our approach using a 2D toy dataset following Balan et al. (2015). The dataset consists of two clusters of 10 points each in 2D space, easily separable by a linear classifier. This toy task is used to determine the ability of a model to capture uncertainty far from the data distribution. We trained a simple NN to perform binary classification on this dataset, using both SGD and SGLD. We used a fully-connected NN (fcNN) with two hidden layers of 10 units each (2-10-10-2). The results are shown in Figure 2. The predictive uncertainty increases in regions far from the observed data. APD was also able to capture this behavior.

5.2. Predictive Performance and Uncertainty

Recently, anomaly detection has been used as a benchmark for BNNs. Here, we show that SGLD and APD are able to detect anomalies better than the SGD and MC dropout baselines.

5.2.1. EXPERIMENTAL SETUP

We used MNIST for our classification and anomaly detection experiments. We trained on 50,000 examples, and reserved 10,000 from the standard training set as a fixed validation set. When training with SGD, we tuned the learning rate and weight decay on the validation set: we found the best values to be 0.05 and 0.001, respectively. We did not use momentum, for fair comparison with vanilla SGLD, which did not use momentum. For baselines, we used **SGD** and MC dropout (**MC-Drop**), where we used the same parameters as for SGD, with an additional dropout rate set to 0.5. For SGLD, we did not use dropout, and the number of burn-in iterations and sampling interval were 500 and 20, respectively. The batch size for training was fixed at 100 for all methods. For the approximate BNNs—MC-Drop,
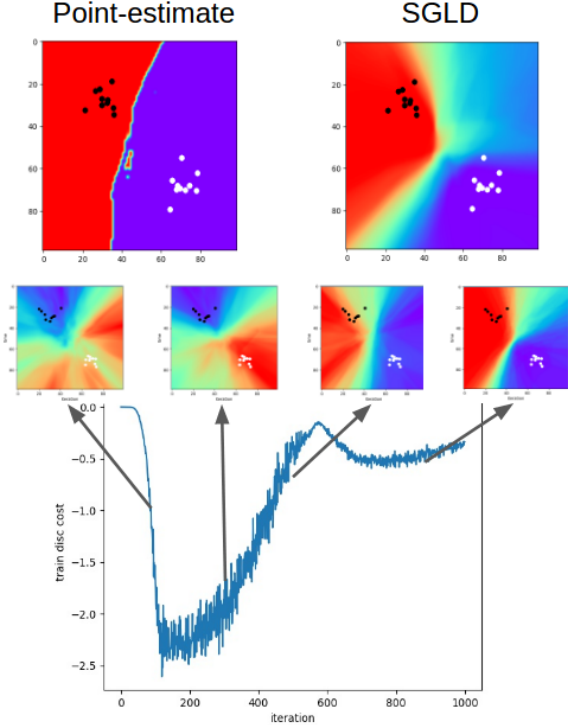
---

[1]Implementation details can be found at https://github.com/wangkua1/apd_public

Figure 2. **Toy2D Classification Results.** *Top*: The decision boundaries of models trained with SGD vs SGLD. The 10×2 white/black dots are inputs from each of the two classes. The model learned with SGD is very confident everywhere, whereas the one learned with SGLD is uncertain far from the training data. *Bottom*: The learning curve for the discriminator loss of WGAN-GP. The initial generated samples result in a random decision boundary. *Middle*: As the WGAN-GP loss improves, the decision boundary looks more similar to the one obtained with real SGLD samples.

SGLD, and APD—predictions were based on the MC estimate of 200 network samples unless otherwise specified.

We experimented with two fcNN architectures: **fcNN1**, with architecture 784-100-10 (79,510 parameters), and **fcNN2**, with architecture 784-400-400-10 (478,410 parameters). For APD, we used a 3-layer fcNN with 100 hidden units per layer for both our generator and discriminator, for all tasks.

5.2.2. CLASSIFICATION ACCURACY

We evaluated the classification accuracy of our method on MNIST, and compared to SGD, SGD with MC dropout (MC-Drop), and SGLD. The results are shown in Table 1. The architectures we explored use narrow hidden layers compared to typical dropout architectures but nonetheless contain a large number of parameters. We include these results to demonstrate that APD is able to distill the posterior distribution of large networks without sacrificing perfor-

mance on this task. Also, these networks, which performed reasonably on classification, were also used for anomaly detection in the following subsection.

Dataset	Model	SGD	MC-Drop	SGLD	APD (Ours)
MNIST	**fcNN1**	0.981	0.973	0.979	0.978
MNIST	**fcNN2**	0.981	0.983	0.980	0.981

Table 1. **MNIST Classification Results.** The samples of network parameters produced by APD achieve classification accuracy competitive with other BNN methods, including MC dropout and SGLD.

5.2.3. ANOMALY DETECTION

We measured the performance of our method on the MNIST anomaly detection task introduced by Hendrycks & Gimpel (2016) and used by Louizos & Welling (2017) and Krueger et al. (2017). Training was unmodified from the previous subsection. At test-time, the inputs consisted of both *in distribution* and OOD data. We evaluated performance using the area under receiver operating curve (AU-ROC) and the area under precision-recall curves (AUPR+/-). ROC is the curve of true-positive rate versus false-positive rate. AUPR+/- is similar, but adjusts for different base rates between the two classes. For both metrics, higher numbers indicate better detection performance. We refer readers to Hendrycks & Gimpel (2016) for details. [2]

Although Hendrycks & Gimpel (2016) and Krueger et al. (2017) showed reasonable results, the deterministic baseline NN already performed very well on this task (i.e., $> 90\%$). We found that the baseline is susceptible to scaling of the pixel intensity of the OOD data (i.e., when the intensities are multiplied by a scalar value $\neq 1$). Hence, for our experiments we scaled the OOD datasets by a factor of 5. Table 2 shows that our method outperforms SGD and MC dropout, and is competitive with SGLD.

We also investigated the impact of the sample size for test-time inference, using the notMNIST OOD dataset. Figure 3 shows that as we increased the sample size, anomaly detection performance improved. Both SGLD and APD consistently outperformed MC dropout.

5.3. Active Learning

For active learning, we followed the experimental setup of Gal et al. (2017), and evaluated on MNIST. For the acquisition function we used entropy, which performed well in Gal et al. (2017). The architecture for our prediction model was based on that of Springenberg et al. (2014). Instead of ReLU, we used LeakyReLU with a negative slope of 0.2, and we used half the number of filters of the original architecture. Our initial training set consisted of 20 labeled

---

[2] We adapted the evaluation code provided at http://github.com/hendrycks/error-detection
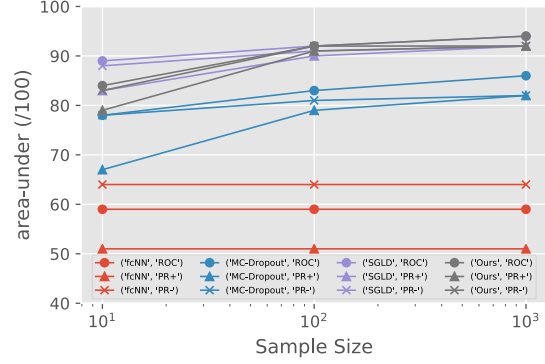

Figure 3. **Effect of Sample Size on Anomaly Detection.** We see improved performance with increasing sample size for test-time inference (5000 total samples).
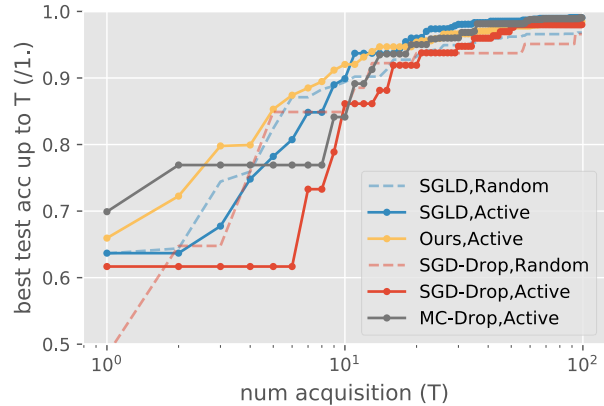

Figure 4. **Active Learning Results.** For BNNs, active learning is able to learn faster than random acquisition. SGD-Drop is SGD with dropout during training.

images, 2 from each of the 10 digit classes; the rest of the images formed a *poolset*. In each acquisition iteration, the model used an acquisition function to choose a set of 10 images from the pool to be labeled (e.g., by a human or oracle). Figure 4 shows that BNNs outperform point-estimate counterparts consistently, and that using entropy as the acquisition function is better than random. Our method performs best early (i.e., < 10 acquisitions) due to either a better regularization effect or better uncertainty for active learning.

5.4. Adversarial Example Detection

Adversarial examples are inputs to a classifier which have been maliciously designed to force misclassification. These inputs are typically produced by taking some existing data point and applying a small perturbation to cause inaccurate predictions (Szegedy et al., 2013; Goodfellow et al., 2014b). Given an input data point **x** which is correctly

Det.	Dataset area under	SGD			MC-Dropout			SGLD			APD (Ours)		
		ROC	PR+	PR-	ROC	PR+	PR-	ROC	PR+	PR-	ROC	PR+	PR-
VR	**notMNIST**	64.2	67.6	54.4	88.0	87.2	82.1	98.1	97.8	98.3	97.8	97.4	98.1
	OmniGlot	84.2	84.9	78.7	91.5	90.8	90.3	99.0	98.8	99.1	98.8	98.6	99.1
	CIFAR10bw	61.4	66.1	52.2	90.1	88.5	86.5	97.4	97.0	97.5	96.9	96.5	96.7
	Gaussian	67.3	70.2	57.4	91.3	89.8	89.0	99.6	99.6	99.7	99.6	99.5	99.6
	Uniform	85.4	80.7	85.8	93.6	91.2	94.8	99.8	99.8	99.9	99.8	99.7	99.8
BALD	**notMNIST**	-	-	-	87.0	85.0	81.0	99.7	99.8	99.6	99.6	99.7	99.5
	OmniGlot	-	-	-	91.4	90.7	90.5	99.9	100.0	99.9	99.9	99.9	99.8
	CIFAR10bw	-	-	-	89.3	86.2	86.0	99.4	99.4	99.2	99.1	99.3	98.3
	Gaussian	-	-	-	90.9	88.6	89.3	100.0	100.0	100.0	100.0	100.0	100.0
	Uniform	-	-	-	97.3	96.6	97.9	100.0	100.0	100.0	100.0	100.0	100.0

Table 2. **MNIST Anomaly Detection Results with fcNN2 (784-400-400-10).** We use variations ratio (top), and BALD (bottom). We show anomaly detection results on several OOD datasets (Hendrycks & Gimpel, 2016), with OOD data scaled by a factor of 5.

Source	Attack Type	MC-Drop	SGLD	Ours
MC-Drop	**FGSM**	89.53	**94.01**	91.70
	PGD	88.37	**93.95**	91.63
SGLD	**FGSM**	54.99	**83.76**	75.93
	PGD	56.91	**84.98**	82.80
Ours	**FGSM**	54.51	83.05	**86.02**
	PGD	54.98	88.01	**93.15**

Table 3. **MNIST Adversarial Detection Results.** Each row shows the AUROC for FGSM and PGD adversaries under each source model.

classified as y, a small perturbation, $\boldsymbol{\delta}$, is added such that $\hat{\mathbf{x}} = \mathbf{x} + \boldsymbol{\delta}$ is now classified as $\hat{y} \neq y$ by the classifier. Different choices of attack determine the form of the perturbation. In practice, we can often find $\boldsymbol{\delta}$ which is imperceptibly small.

We focused on two attacks: Fast Gradient-Sign Method (FGSM) (Goodfellow et al., 2014b), and Projected Gradient Descent (PGD) (Kurakin et al., 2016; Madry et al., 2017). FGSM utilizes a simple unit step in the direction of increasing gradient of the network's cost function. PGD utilizes repeated smaller steps of the same form, while projecting the output onto $B_\epsilon(\mathbf{x})$, the ℓ-infinity ball of size ϵ. FGSM is considered a relatively easy attack to defend against, while PGD is a strong attack with evidence supporting it as a "universal" first-order attack (Madry et al., 2017). We made use of the foolbox library (Rauber et al., 2017) for both of these attacks, using our own implementation for PGD.

In this work, the adversary has access to the network architecture and a single posterior sample. We generated 6000 adversarial examples from the validation set using this posterior sample as fixed network weights. We then used 1000 samples from the posterior distribution to detect whether an input point was an adversarial example or belonged to the test set. We refer to this scheme as a gray-box attack because the attacker does not have access to the full posterior distribution. For these experiments, we used the approximate model variance (Eqn. 11) which we found outperformed entropy and BALD on this problem.

Table 3 shows the results of using various MC techniques

to detect adversarial examples. We compared SGLD and our method to MC dropout inference detection (Feinman et al., 2017) as a baseline. We used a small single-hidden-layer network with 100 units. For each detection method, we generated attacks using samples from the source model and tested these attacks against all other approaches.

We found that all three approaches were effective when detecting adversarial examples crafted with their own networks. However, when transferring attacks between networks there was a steep drop-off for MC dropout inference which performed only slightly better than random. Both SGLD inference and our own method were able to detect transferred attacks. In this setting we were able to see a more substantial gap between SGLD and our own method—suggesting that the GAN was unable to capture some quality of the posterior distribution that is critical for adversarial example detection.

Finally, we note that previous attempts at detection have proven ill-tested. Following the guidance of Carlini & Wagner (2017), we argue that it is critical to evaluate these methods on a more challenging dataset such as CIFAR-10 (Krizhevsky & Hinton, 2009) and using attacks which take advantage of the detection scheme. We hope to explore this in future work.

5.5. Distillation with GANs

Here, we show that distilling the parameters of a network is not trivial, and that recent advances in GANs make them a promising approach. For the experiments in this section, we used the **fcNN1** architecture and measured performance on the anomaly detection task (see Section 5.2.3) with the notMNIST OOD dataset (as it is one of the most challenging ones).

Do we need a GAN? Using MCMC methods allows us to avoid making simplifying assumptions about the posterior distribution. In this section we show evidence that the multimodal posterior distribution induced by SGLD samples cannot be completely represented with simple, fully factorized approximations. We performed this analysis using a series of Mixture of Gaussians (MoG) with increasing

number of components, N_c. We fit the MoG to posterior samples using the EM algorithm (Dempster et al., 1977).
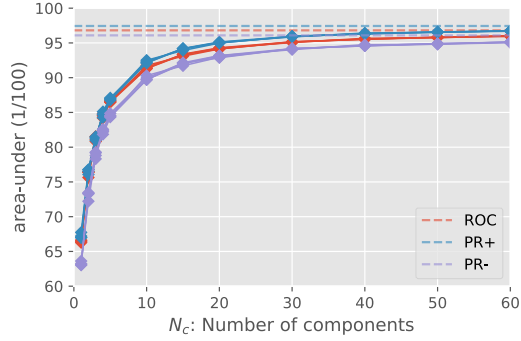

Figure 5. **Effect of Increasing Number of GMM Components on Anomaly Detection.** The dashed lines correspond to the performance when using SGLD samples used to train the GMM directly. We see improved performance as the number of components increases—approaching the performance of using the SGLD samples directly.

We compared the performance of MoG with varying N_c for anomaly detection in Figure 5. For each value of N_c the MoG was trained using 2000 posterior samples drawn using SGLD, and the uncertainty measure was entropy. The performance of using SGLD directly is plotted as a dashed line. We see that when using a single component (corresponding to a fully factorized Gaussian) the performance is poor—suggesting that a single mode is not sufficient to model the posterior. As the number of components increases, the MoG begins to approach the performance achieved by using the SGLD samples themselves.

With enough components, the MoG is able to perform well on the anomaly detection tasks. In order to do so, the diagonal covariance MoG requires at least a mean and variance parameter for each network parameter per component. This reduces the memory overhead of using SGLD directly, but is still costly compared to the GAN. Using a MoG model with 60 components (9.54M parameters) retains 99.3% of the performance on this task w.r.t. the original SGLD samples. APD (using WGAN-GP) retains 99.8% of the performance while using fewer parameters (1.67M parameters with GAN hidden size 20).

APD Storage Savings. We compared the performance of SGLD and APD using different numbers of samples at test-time. With a 3-layer GAN with 20 hidden units per layer, generating 20 samples performs worse than simply using 20 original SGLD samples. However, the storage cost of APD is not affected by drawing more samples. Figure 6 shows that as we generate more samples from the GAN, the performance improves and reaches that of 50 SGLD samples (i.e., 2.5x storage savings).

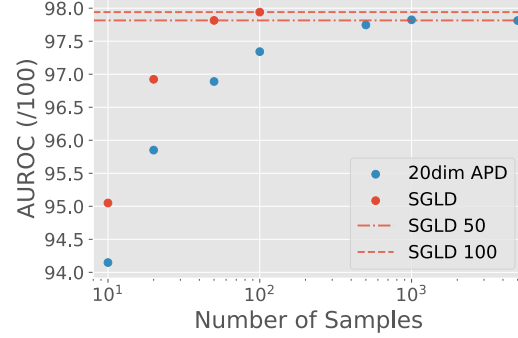**Comparing GAN Formulations.** Our framework makes use of recent advances in GANs. We compared the training

Figure 6. **Effect of SGLD and APD Sample Size.** With APD, the storage cost (i.e., generator size) is fixed regardless of the number of samples we generate at test-time. The two horizontal lines extend the y-value of SGLD at 50 and 100 samples. Here, we used BALD; VR and entropy yield similar results.
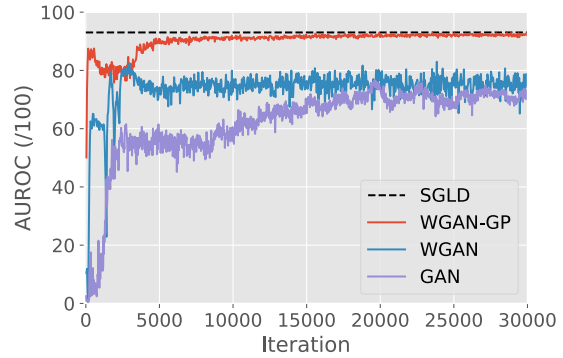

Figure 7. **Comparison of GAN Formulations.** WGAN-GP achieves better performance and converges faster than WGAN or the original GAN w.r.t. AUROC. Here, we used VR.

progress of our GAN using three popular variants: 1) the original formulation; 2) the Wasserstein GAN with weight-clipping; and 3) WGAN with gradient penalty. Figure 7 shows that WGAN-GP converges faster, and exhibits fewer oscillations from iteration to iteration.

6. Conclusion

We introduced a framework for distilling BNN posterior samples drawn using SGLD, which we call Adversarial Posterior Distillation (APD). Experimental results show that APD is able to retain the characteristics of the SGLD samples, as measured by performance on downstream applications including anomaly detection, active learning, and defense against adversarial attacks. MCMC methods have attracted relatively little attention in BNNs due to their computational cost. APD provides a way to reduce the storage cost. Our findings thus demonstrate that these MCMC methods have the potential to outperform simple alternatives such as MC dropout on important tasks that require uncertainty estimates. For future directions, we aim to explore other generative models that can further reduce the storage cost, such as autoregressive models.

References

Ahn, S., Korattikara, A., and Welling, M. Bayesian posterior sampling via stochastic gradient Fisher scoring. *arXiv preprint arXiv:1206.6380*, 2012.

Ahn, S., Shahbaba, B., and Welling, M. Distributed stochastic gradient MCMC. In *International Conference on Machine Learning*, pp. 1044–1052, 2014.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017.

Balan, A. K., Rathod, V., Murphy, K. P., and Welling, M. Bayesian dark knowledge. In *Advances in Neural Information Processing Systems*, pp. 3438–3446, 2015.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Carlini, N. and Wagner, D. Adversarial examples are not easily detected: Bypassing ten detection methods. *arXiv preprint arXiv:1705.07263*, 2017.

Chen, T., Fox, E., and Guestrin, C. Stochastic gradient Hamiltonian Monte Carlo. In *International Conference on Machine Learning*, pp. 1683–1691, 2014.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B*, pp. 1–38, 1977.

Ding, N., Fang, Y., Babbush, R., Chen, C., Skeel, R. D., and Neven, H. Bayesian sampling using stochastic gradient thermostats. In *Advances in Neural Information Processing Systems*, pp. 3203–3211, 2014.

Feinman, R., Curtin, R. R., Shintre, S., and Gardner, A. B. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

Gal, Y. and Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pp. 1050–1059, 2016.

Gal, Y., Islam, R., and Ghahramani, Z. Deep Bayesian active learning with image data. *arXiv preprint arXiv:1703.02910*, 2017.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014a.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014b.

Graves, A. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pp. 2348–2356, 2011.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. Improved training of Wasserstein GANs. *arXiv preprint arXiv:1704.00028*, 2017.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017.

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

Hernández-Lobato, J. M. and Adams, R. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *International Conference on Machine Learning*, pp. 1861–1869, 2015.

Hinton, G. E. and Van Camp, D. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pp. 5–13. ACM, 1993.

Houlsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

Kingma, D. P. and Welling, M. Stochastic gradient VB and the variational auto-encoder. In *Second International Conference on Learning Representations*, 2014.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*, 2009.

Krueger, D., Huang, C.-W., Islam, R., Turner, R., Lacoste, A., and Courville, A. Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*, 2017.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

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

Li, Y., Turner, R. E., and Liu, Q. Approximate inference with amortised MCMC. *arXiv preprint arXiv:1702.08343*, 2017.

Louizos, C. and Welling, M. Multiplicative normalizing flows for variational Bayesian neural networks. *arXiv preprint arXiv:1703.01961*, 2017.

MacKay, D. J. A practical Bayesian framework for back-propagation networks. *Neural Computation*, 4(3):448–472, 1992.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Minka, T. P. Expectation propagation for approximate Bayesian inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pp. 362–369. Morgan Kaufmann Publishers Inc., 2001.

Neal, R. M. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1996.

Rauber, J., Brendel, W., and Bethge, M. Foolbox v0.8.0: A Python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017.

Rawat, A., Wistuba, M., and Nicolae, M.-I. Adversarial phenomenon in the eyes of Bayesian deep learning. *arXiv preprint arXiv:1711.08244*, 2017.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Rosca, M., Lakshminarayanan, B., Warde-Farley, D., and Mohamed, S. Variational approaches for auto-encoding generative adversarial networks. *arXiv preprint arXiv:1706.04987*, 2017.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

Salimans, T., Zhang, H., Radford, A., and Metaxas, D. Improving GANs using optimal transport. *International Conference on Learning Representations*, 2018.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Vollmer, S. J., Zygalakis, K. C., and Teh, Y. W. Exploration of the (non-) asymptotic bias and variance of stochastic gradient langevin dynamics. *Journal of Machine Learning Research*, 17(159):1–48, 2016.

Wei, X., Liu, Z., Wang, L., and Gong, B. Improving the improved training of Wasserstein GANs. *International Conference on Learning Representations*, 2018.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 681–688, 2011.