Adversarial Distillation of Bayesian Neural Networks

Satya Narayan Shukla[1], **Meet P. Vadera**[1], **Brian Jalaian**[2], **Benjamin M. Marlin**[1]
[1]University of Massachusetts Amherst, [2]US Army Research Laboratory
{snshukla, mvadera, marlin}@cs.umass.edu, brian.a.jalaian.civ@mail.mil

Abstract

In this paper, we consider the problem of assessing the adversarial robustness of deep neural network models under both Markov chain Monte Carlo (MCMC) and Bayesian Dark Knowledge (BDK) inference approximations. We characterize the robustness of each method to two types of adversarial attacks: the fast gradient sign method (FGSM) and projected gradient descent (PGD). We show that full MCMC-based inference has excellent robustness, significantly outperforming standard point estimation-based learning. On the other hand, BDK provides marginal improvements. We propose Adversarial Distillation: adversarial training method to improve the robustness of BDK. Our results on MNIST and CIFAR10 show significant robustness improvement. As an additional contribution, we present a storage-efficient approach to computing adversarial examples for large Monte Carlo ensembles using both the FGSM and PGD attacks.

1 Introduction

Deep learning models have shown promising results in areas including computer vision, natural language processing, speech recognition, and more (Krizhevsky et al., 2012; Graves et al., 2013a;b; Huang et al., 2016; Devlin et al., 2018). Despite these advances, deep neural networks are well-known to be vulnerable to adversarial examples. An adversarial example is an example that differs from a natural example through a low-norm additive perturbation while resulting in an incorrect prediction relative to the unperturbed example Goodfellow et al. (2014). The lack of robustness to adversarial examples is a crucial barrier to the safe deployment of deep learning models in many applications.

One potential source of adversarial examples in traditional point-estimated deep neural network models derives from the fact that the decision boundary geometry can be minimally constrained away from the training data during learning. This can lead to quite arbitrary decision boundaries away from the training data, which may be easily attackable Nguyen et al. (2015). By contrast, Bayesian inference methods result in predictable and well-behaved decision boundary geometry off of the training data due to the Bayesian model averaging effect when computing the posterior predictive distribution. If many decision boundary geometries are all equally likely in a given region of feature space, the posterior predictive distribution will average over all of them resulting in both posterior class probabilities and decision boundary geometry with improved smoothness, which may be harder to attack with low-norm adversarial perturbations. Indeed, Gal and Smith (2018) present theoretical evidence that under certain sufficient conditions there exists no adversarial examples for a Bayesian classification model.

Markov Chain Monte Carlo (MCMC) methods are one of the primary alternatives to variational methods for performing approximate inference in Bayesian neural networks. While we defer the background on Bayesian neural networks to the next section, it is important to note that MCMC methods give an unbiased estimate to the parameter posterior and the posterior predictive distribution, while variational methods are typically biased. However, MCMC methods require materializing or storing samples of the parameter posterior in order to make predictions, which can have high computational complexity and storage cost.

To help overcome these problems, Balan et al. (2015) introduced a model distillation method referred to as *Bayesian Dark Knowledge* (BDK). In the classification setting, Bayesian Dark Knowledge attempts to compress

the Bayesian posterior predictive distribution induced by the full parameter posterior of a "teacher" network into a single, compact "student" network. The major advantage of this approach is that the computational complexity of prediction at test time is drastically reduced. This method has been shown to successfully reproduce the full posterior predictive distribution on test data drawn from the training distribution Balan et al. (2015).

In this paper, we consider the problem of assessing the adversarial robustness of deep neural network models under both the MCMC and Bayesian Dark Knowledge approximations. We characterize the robustness of each method to two types of adversarial attacks: the fast gradient sign method (FGSM) Goodfellow et al. (2014) and projected gradient descent (PGD) Madry et al. (2017). We consider the case of a basic convolutional neural network (CNN) architecture and the MNIST and CIFAR10 data sets. Interestingly, we show that full MCMC-based inference has excellent robustness to these adversarial attacks, significantly outperforming standard point estimation-based learning, while BDK only provides marginal improvements. This indicates that the BDK distillation procedure is failing to fully capture the structure of the true posterior predictive distribution off of the training data.

To improve the robustness of BDK, we propose adversarial distillation where we distill the Bayesian posterior predictive distribution on the adversarial examples during training. This is very similar to standard adversarial training except we use the Bayesian posterior predictive distribution instead of original labels. Results on both MNIST and CIFAR10 show significant improvement in robustness towards adversarial attacks. As an additional contribution, we present a storage-efficient approach to computing adversarial examples for large Monte Carlo ensembles using both the FGSM and PGD attacks.

2 Background

2.1 Bayesian Neural Networks

Let $p(y|\mathbf{x}, \theta)$ represent the probability distribution induced by a deep neural network classifier over classes $y \in \mathcal{Y} = \{1, .., C\}$ given feature vectors $\mathbf{x} \in \mathbb{R}^D$. The most common way to fit a model of this type given a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)|1 \leq i \leq N\}$ is to use maximum conditional likelihood estimation, or equivalently, cross entropy loss minimization (or their penalized or regularized variants). However, when the volume of labeled data is low, there can be multiple advantages to considering a full Bayesian treatment of the model. Instead of attempting to find the single (locally) optimal parameter set θ_* according to a given criterion, Bayesian inference uses Bayes rule to define the posterior distribution $p(\theta|\mathcal{D}, \theta^0)$ over the unknown parameters θ given a prior distribution $P(\theta|\theta^0)$ with prior parameters θ^0 as seen in Equation 1.

$$p(\theta|\mathcal{D}, \theta^0) = \frac{p(\mathcal{D}|\theta)p(\theta|\theta^0)}{\int p(\mathcal{D}|\theta)p(\theta|\theta^0)d\theta} \tag{1}$$

$$p(y|\mathbf{x}, \mathcal{D}, \theta^0) = \int p(y|\mathbf{x}, \theta)p(\theta|\mathcal{D}, \theta^0)d\theta \tag{2}$$

For prediction problems in machine learning, the quantity of interest is typically not the parameter posterior itself, but the posterior predictive distribution $p(y|\mathbf{x}, \mathcal{D}, \theta^0)$ obtained from it as seen in Equation 2. The primary problem with applying Bayesian inference to neural network models is that the distributions $p(\theta|\mathcal{D}, \theta^0)$ and $p(y|\mathbf{x}, \mathcal{D}, \theta^0)$ are not available in closed form, so approximations are required.

Most Bayesian inference approximations studied in the machine learning literature are based on variational inference (VI) (Jordan et al., 1999) or Markov Chain Monte Carlo (MCMC) methods (Neal, 1996; Welling and Teh, 2011). In VI, an auxiliary distribution $q_\phi(\theta)$ is defined to approximate the true parameter posterior $p(\theta|\mathcal{D}, \theta^0)$. The main drawback of VI and related methods is that they typically result in biased posterior estimates for complex posterior distributions. MCMC methods provide an alternative family of sampling-based posterior approximations that are unbiased. The samples generated using MCMC methods can then be used to approximate the posterior predictive distribution using a Monte Carlo average as shown in Equation 3.

$$p(y|\mathbf{x}, \mathcal{D}, \theta^0) \approx \frac{1}{T}\sum_{t=1}^{T} p(y|\mathbf{x}, \theta_t); \quad \theta_t \sim p(\theta|\mathcal{D}, \theta^0) \tag{3}$$

Neal (1996) first addressed the problem of Bayesian inference in neural networks using Hamiltonian Monte Carlo (HMC) to provide a set of posterior samples. The stochastic gradient Langevin dynamics (SGLD) sampling method improves on HMC by enabling sampling based on minibatches of data to improve the time computational complexity of sampling (Welling and Teh, 2011); however, the problem of needing to compute over a large set of samples when making predictions at test or deployment time still remains. Bayesian Dark Knowledge (Balan et al., 2015) aims at reducing the test-time computational complexity of Monte Carlo-based approximations for neural networks by distilling the posterior predictive distribution (approximated by Equation 3) of a neural network into another neural network. We will discuss the details of both methods in Section 3.

2.2 Adversarial Attacks

Adversarial examples are carefully crafted perturbations to a classifier input that are designed to mislead a classifier while being as imperceptible as possible. Based on the information available to the adversary, these attacks are broadly categorized as *white-box* and *black-box* attacks. Most methods for adversarial attacks on deep learning models operate in the white-box setting (Goodfellow et al., 2014; Madry et al., 2017; Kurakin et al., 2016; Moosavi-Dezfooli et al., 2016; Sabour et al., 2015; Carlini and Wagner, 2016), where the model being attacked, and its gradients, are assumed to be fully known. Conversely, the black-box setting (Brendel et al., 2017; Cheng et al., 2018; Chen et al., 2017; Ilyas et al., 2018) requires an attacker to find an adversarial perturbation when its only access to the model is via labeling queries.

The most successful adversarial attacks use gradient-based optimization methods. For example, the Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014) is a one-step method that uses the sign of the gradient to create adversarial examples:

$$\mathbf{x}_{adv} = \mathbf{x} + \epsilon \text{ sign}(\nabla_{\mathbf{x}} \mathcal{L}(\theta, \mathbf{x}, y)), \tag{4}$$

where $\mathcal{L}$ is the standard cross-entropy loss computed using the model prediction given input $\mathbf{x}$ and the original label y, and ϵ governs the magnitude of the perturbation introduced and can be thought of as a step size. Kurakin et al. (2017) extended this to a multi-step variant which is more powerful than single step FGSM. Both the methods are, however, limited to generating ℓ_∞-bounded perturbations. Madry et al. (2017) introduced a more general multi-step variant, which is essentially projected gradient descent (PGD) on the negative loss function. FGSM-based attacks can be seen as specific instances of PGD under ℓ_∞-bounded perturbations. The main emphasis of the PGD attack is to apply FGSM k times (number of iterations) with step-size $\alpha \leq \epsilon/k$, where ϵ is the maximum distortion (i.e., attack strength) of the adversarial example compared to the original input. The resulting adversarial example $\mathbf{x}^{t+1}$ corresponding to input x is computed as follows:

$$\mathbf{x}^{t+1} = \Pi_{\mathbf{x}+S}(\mathbf{x}^t + \alpha \text{ sign}(\nabla_{\mathbf{x}} \mathcal{L}(\theta, \mathbf{x}, y))) \tag{5}$$

where Π_{x+S} is the projection onto the ℓ_∞ ball of radius ϵ centred at $\mathbf{x}$.

2.3 Adversarial Training

Adversarial training (Goodfellow et al., 2015; Szegedy et al., 2014) is a method for training robust deep neural networks with respect to adversarial attacks. Let f_θ be a deep neural network parameterized by θ, $\mathcal{D}_{tr}$ a training dataset, $\mathcal{L}$ the loss function, the learning is typically cast as the following optimization problem:

$$\min_{\theta} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_{tr}} \left\{ \max_{\boldsymbol{\delta}\in\Delta} \mathcal{L}(f_\theta(\mathbf{x}+\boldsymbol{\delta}), y) \right\} \tag{6}$$

where Δ is the threat model which constraints the allowed perturbation. The most commonly used threat model is ℓ_∞ where $\{\Delta = \boldsymbol{\delta} : \|\boldsymbol{\delta}\|_\infty \leq \epsilon\}$. This is the threat model used by Madry et al. (2017) and in this work. The above optimization problem is approximately solved by first generating adversarial examples using an attack method and then minimizing the loss of the adversarial example using a gradient descent like method.

Goodfellow et al. (2014) used a simple one step Fast Gradient Sign Method (FGSM) to approximate the inner maximization. It was later improved by Madry et al. (2017) by taking multiple smaller FGSM steps instead, also known as Projected Gradient Descent. To reduce the computational overhead in approximating the inner maximization, Shafahi et al. (2019) proposed interleaving of adversarial examples generation using PGD and model update. This can be achieved with little overhead to the standard training but still requires long time to train. Wong et al. (2020) showed that FGSM based adversarial training when combined with random initialization is as effective as PGD-based training with almost no overhead as compared to standard training.

2.4 Adversarial Training and Bayesian Neural Networks

Recently, several works have focused on studying the adversarial robustness of Bayesian Neural Networks. Gal and Smith (2018) presented theoretical evidence that under certain sufficient conditions there exists no adversarial examples for a Bayesian classification model. However, Liu et al. (2019) show that Bayesian neural networks implemented using variational inference lack robustness against adversarial samples, and subsequently present a method that attempts to make them more robust. Their method relies on generating adversarial samples using the PGD attack and feeding them back in the training cycle. On the other hand, Ye and Zhu (2018) aims to take a Bayesian approach to adversarial learning. Their algorithm jointly samples from the model's parameter posterior, and the distribution of adversarial samples against the current parameter posterior. At the end of the training, the ensemble is formed by collecting the samples formed by the final posterior distribution obtained.

3 Methods

3.1 Approximate Bayesian Inference Methods

For implementing approximate Bayesian inference in neural networks, we adopt the use of the stochastic gradient Langevin dynamics (SGLD) (Welling and Teh, 2011) method as a computationally efficient MCMC sampler. We compare the adversarial robustness of the SGLD approximation to the posterior predictive distribution to that provided by Bayesian dark knowledge (BDK) Balan et al. (2015), which is drastically more efficient to deploy. SGLD is also used at the core of the BDK posterior distillation algorithm. We begin by reviewing SGLD, and then describe BDK.

Let $p(\theta|\lambda)$ be the prior distribution over the network model parameters θ. The prior distribution is selected to be a spherical Gaussian distribution centered at 0 with precision denoted by λ. We define $\mathcal{S}$ to be a minibatch of size M drawn from $\mathcal{D}$. Let the total number of training samples in $\mathcal{D}$ be denoted by N. θ_t denotes the parameter set sampled from the model at sampling iteration t, while η_t denotes the learning rate at iteration t. The Langevin noise is denoted by $z_t \sim \mathcal{N}(0, \eta_t I)$. The sampling update for SGLD can be written as seen below where $p\left(y_i|x_i, \theta_t\right)$ is the likelihood:

$$\Delta\theta_{t+1} = \frac{\eta_t}{2}\left(\nabla_\theta \log p(\theta|\lambda) + \frac{N}{M}\sum_{i\in\mathcal{S}} \nabla_\theta \log p\left(y_i|x_i,\theta_t\right)\right) + z_t. \tag{7}$$

By running this update, we produce a sequence of samples that converges to the posterior distribution of the model we are sampling from. This set of samples forms a Monte Carlo ensemble and is used in Equation 3 when making predictions. In BDK terminology, the model we sample from is referred to as the "teacher" model and the set of sampled models is referred to as the "teacher ensemble." BDK aims to compress the true posterior predictive distribution as approximated by a teacher ensemble into a compact, feed-forward neural network model (referred to as the "student" model) using distillation methods to avoid the need to store samples and compute predictions using Equation 3 at deployment time.

3.2 Standard Distillation

To learn the student model, BDK generates a batch of samples $\mathcal{S}'$. $\mathcal{S}'$ is obtained by adding Gaussian noise of small magnitude to $\mathcal{S}$. The output of the teacher model on samples $\{(\mathbf{x}', y')\} \in \mathcal{S}'$ is approximated using the current sample from the teacher model: $p(y'|\mathbf{x}', \theta_{t+1})$. Similarly, the output of the student model, parameterized by ω_t, is computed as $p(y'|\mathbf{x}', \omega_t)$. The objective function for learning the student is the KL divergence between the teacher model's predictive distribution and the student model's predictive distribution:

$$\mathcal{L}(\omega_t|\mathcal{S}', \theta_{t+1}) = \sum_{(\mathbf{x}',y')\in\mathcal{S}'} \mathrm{KL}(p(y'|\mathbf{x}', \theta_{t+1})||p(y'|\mathbf{x}', \omega_t)) \tag{8}$$

and we run a single optimization iteration on it to obtain ω_{t+1}. This process of sequentially computing θ_{t+1} and ω_{t+1} is repeated until convergence.

Algorithm 1 FGSM for Attacking Teacher Ensemble

procedure FGSME($\mathbf{x}, y, \{\theta_i\}_{i=1}^K, \epsilon, \boldsymbol{\delta} = 0$)
  Initialize $\nabla = 0$
  for $k = 1$ to K **do**
    $\nabla \leftarrow \nabla + \nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x} + \boldsymbol{\delta}, y, \theta_k)$
  end for
  $\boldsymbol{\delta} \leftarrow \boldsymbol{\delta} + \epsilon$ sign(∇)
  $\boldsymbol{\delta} \leftarrow$ Clamp$(\boldsymbol{\delta}, -\epsilon, \epsilon)$
  $\mathbf{x}_{adv} \leftarrow \mathbf{x} + \boldsymbol{\delta}$
  $\mathbf{x}_{adv} \leftarrow$ Clamp$(\mathbf{x}_{adv}, 0, 1)$
  return $\mathbf{x}_{adv}$
end procedure

3.3 Attacking Approximate Bayesian Inference

To investigate the adversarial robustness of the Monte Carlo ensembles produced by SGLD and the distilled student models produced by BDK, we apply two common forms of attacks adopted in the literature: PGD and FGSM. Both of these attacks rely on computing the gradient of the cross-entropy loss function between the model output and labels w.r.t the inputs. This presents a computational challenge when the model we must attack is a large Monte Carlo ensemble consisting of hundreds or thousands of models. Indeed, even explicitly storing all of the models in the ensemble can be a challenge. To address this problem, we leverage the fact that the gradient of the loss w.r.t input of the ensemble is a sum of the gradients of the loss w.r.t the input for each model in the ensemble. For an ensemble $\{\theta_i\}_{i=1}^K$ consisting of K models sampled from the posterior, the gradient of the loss $\mathcal{L}(\cdot; \theta_{1:K})$ w.r.t. input $\mathbf{x}$ can be expressed as:

$$\nabla_{\mathbf{x}}\mathcal{L}(y, \mathbf{x}; \theta_{1:K}) = \frac{1}{K}\sum_{i=1}^{K} \nabla_{\mathbf{x}}\mathcal{L}(\cdot; \theta_i) \tag{9}$$

The previous equation follows directly from Equation 3. As can be seen clearly, we only need access to a single model from the ensemble at a time and accumulate the gradients to obtain the gradient of the ensemble. This enables us to perform FGSM and PGD attacks with constant memory while scaling up the number of models in the ensemble. Further, if we have access to the data, we

Algorithm 2 Adversarial Distillation: FGSM Adversarial Training for T epochs

procedure ADV-DISTILL($\{\mathbf{x}, y\}_{i=1}^{N}, \{\theta_k\}_{k=1}^{K}, \phi, \epsilon, \alpha$, rand_init)
  Initialize $\boldsymbol{\delta} = 0$
  for $t = 1$ to T **do**
    for $i = 1$ to N **do**
      if rand_init is True **then**
        $\boldsymbol{\delta} \leftarrow \text{Uniform}(-\epsilon, \epsilon)$ ▷ Random initialization
      end if
      $y' \leftarrow \arg\max_y \frac{1}{K} \sum_{k=1}^{K} P(Y = y | X = \mathbf{x}_i, \theta_k)$ ▷ Computing Teacher Ensemble prediction
      $\boldsymbol{\delta} \leftarrow \boldsymbol{\delta} + \alpha \ \text{sign}(\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}_i + \boldsymbol{\delta}, y', \phi_t))$ ▷ Fast Gradient Sign Method
      $\boldsymbol{\delta} \leftarrow \text{Clamp}(\boldsymbol{\delta}, -\epsilon, \epsilon)$ ▷ Projecting adversarial perturbation on ℓ_∞-ball
      $\mathbf{x}_{adv} \leftarrow \text{Clamp}(\mathbf{x}_i + \boldsymbol{\delta}, 0, 1)$ ▷ Projecting back to original image space
      $\pi_y = \frac{1}{K} \sum_{k=1}^{K} P(Y = y | X = \mathbf{x}_i, \theta_k)$ ▷ Computing Teacher Ensemble posterior for distillation
      $y'' = \arg\max_y \pi_y$
      if $y'' = y_i$ **then** ▷ Removing examples adversarial to Teacher Emsemble
        $\phi_{t+1} \leftarrow \phi_t - \nabla_{\phi_t} \mathcal{L}(\mathbf{x}_{adv}, \pi_y, \phi_t)$ ▷ Updating the parameters of student using an optimizer
      end if
    end for
  end for
end procedure

only actually need to store a starting model and a random seed. The rest of the elements of the ensemble can be sequentially materialized by re-running the SGLD sampling iteration. This allows us to also generate attacks for large ensembles while using constant storage cost. Algorithm 2 shows FGSM attack on Monte Carlo Ensemble. An iterative version of FGSM attack with small step size leads to the PGD attack.

We note that this corresponds to an incredibly strong attack against a Bayesian model as we effectively assume that we have access to every element of the approximating Monte Carlo ensemble, despite the fact that the samples could be updated at any time. Finally, we note that attacking the BDK student model is straightforward since the student is a standard feed-forward model. This attack requires no additional modifications to the original algorithms.

3.4 Adversarial Distillation

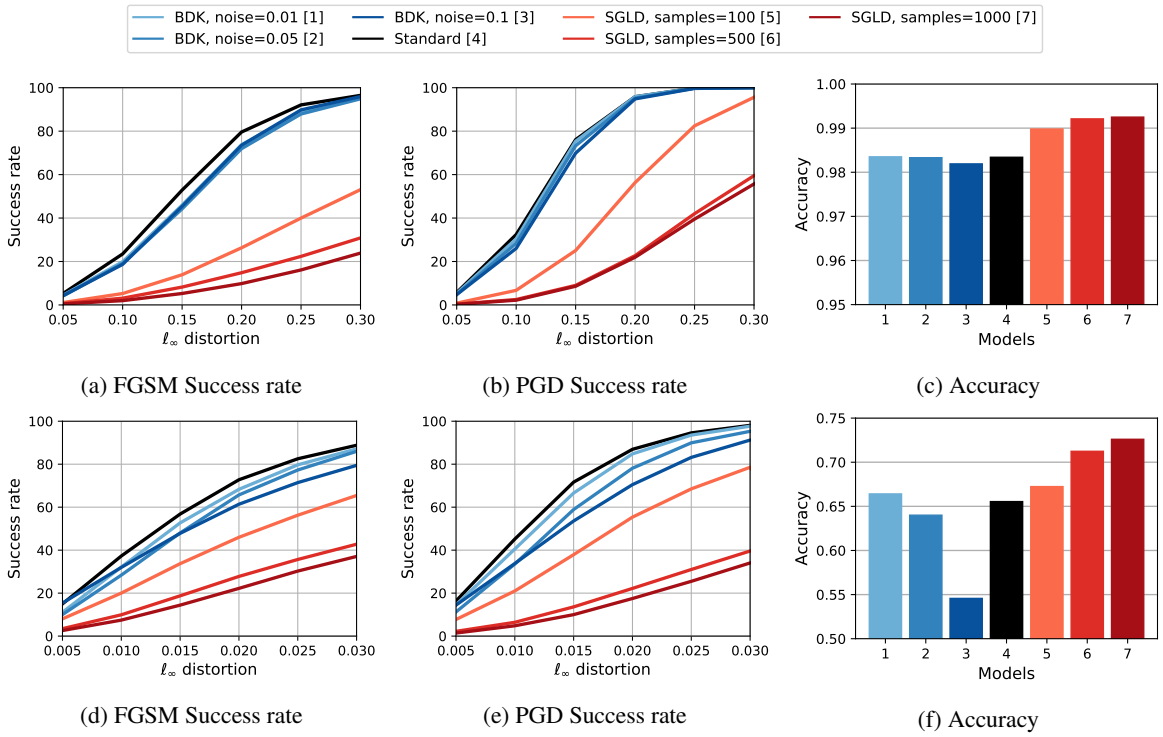
As shown in the experiments, the SGLD Monte Carlo ensembles are significantly more robust than the standard distilled student models towards adversarial attacks. Moreover, we observed that the adversarial examples to student are in most cases not adversarial to the teacher ensemble. We utilize this property of teacher ensemble in distilling adversarial robustness into student model.

In this section, we propose an adversarial distillation method to distill the robustness of teacher ensemble into a compact student network without incurring any extra cost. In our method, we combine the idea of adversarial training with Bayesian Dark Knowledge. We use FGSM based adversarial training combined with random initialization which is shown to be as effective a defense as PGD based adversarial training (Wong et al., 2020). The student model is now trained on adversarial examples, instead of the original input data, created with FGSM with teacher model's predictive distribution. Our approach for adversarial distillation is shown in Algorithm 2. We use the same objective function as described in Equation 8 on the adversarially perturbed examples. While distilling the teacher posterior on the adversarial examples, we only train with perturbed examples that are not adversarial to the teacher. We empirically show that this leads to better robustness than the case we include the perturbed examples that are adversarial to teacher in our training set.

In our adversarial distillation approach, it requires two backwards passes to compute gradients separately for the perturbation and the model weights per mini batch. As a result, the complexity of our approach is equivalent to twice of standard training but it is still much lower than standard PGD based adversarial training which solves an optimization problem inside the training loop in every iteration.

4 Experiments

Our experiments focus on the untargeted attack setting where the goal is to cause the model to misclassify inputs that are otherwise correctly classified. We consider



(a) FGSM Success rate (b) PGD Success rate (c) Accuracy

(d) FGSM Success rate (e) PGD Success rate (f) Accuracy

Figure 1: Performance comparison for ℓ_∞ white-box attacks on MNIST (top row) and CIFAR10 (bottom row).

the ℓ_∞ threat model for the experiments. We evaluate the models on MNIST (Lecun et al., 1998) and CIFAR-10 (Krizhevsky et al., 2009) datasets in terms of attack success rate and accuracy on unperturbed test data. We consider two standard white-box attack methods - FGSM (Goodfellow et al., 2014) and PGD (Madry et al., 2017), for measuring the robustness of SGLD ensembles and BDK student models. For all the experiments mentioned, we vary the ℓ_∞ distortion from 0.05 to 0.3 for MNIST and from 0.005 to 0.03 for CIFAR10 and report the attack success rates of FGSM and PGD. For PGD attacks, we run 40 iterations with step size of 0.05 for MNIST and 0.005 for CIFAR10.

We define the attack success rate as the percentage of the number of inputs successfully perturbed using the attack method to the total number of input images that we attempt to perturb. For computing the success rate, images that are already misclassified by any model are excluded from the attack set. All results are based on the test set, which consists of 10000 examples for each data set.

4.1 Empirical Protocols

4.1.1 Models

We utilize CNNs for both the teacher ensemble and the student model. Further, for a given data set, we use the same architecture for the teacher as well as the student. For MNIST, we use the following architecture: Input(1, (28,28))-Conv(num_kernels=10, kernel_size=4, stride=1) - MaxPool(kernel_size=2) - Conv(num_kernels=20, kernel_size=4, stride=1) - MaxPool(kernel_size=2) - FC (80) - FC (output). For CIFAR10, we utilize the following architecture: Input(3, (32,32)) - Conv(num_kernels=16, kernel_size=5) - MaxPool(kernel_size=2) - Conv(num_kernels=32, kernel_size=5) - MaxPool(kernel_size=2) - FC(200) - FC (50) - FC (output).

4.1.2 Training Details

We run the SGLD and standard distillation procedure using the following hyperparameters: fixed teacher learning rate $\eta_t = 4 \times 10^{-6}$ for MNIST and $\eta_t = 3 \times 10^{-6}$ for CIFAR10, teacher prior precision $\lambda = 10$, initial student learning rate $\rho_t = 10^{-3}$, burn-in iterations $B = 1000$ for MNIST and $B = 10000$ for CIFAR10, thinning interval $\tau = 100$, and total training iterations $T = 10^6$. For training the student model, we use the Adam optimizer and set a learning schedule for the student such that it halves its learning rate every 200 epochs for MNIST, and every 400 epochs for CIFAR10.

We use the same MNIST and CIFAR10 architecture for adversarial distillation. We use cyclic learning rate

(Smith, 2015) in adversarial training of student networks where learning rate linearly increases form 0 to λ in first $N/2$ epochs and decreases back to 0 in next half. As recommended by Smith and Topin (2017), we tune the $\lambda \in [0.0001, 0.5]$ to be as large as possible without causing the training loss to diverge. We treat total number of iterations as a hyperparameter and tune it within the range $N \in [10, 100]$ respectively. We use a batch size of 128 for adversarial training on both MNIST and CIFAR10. We use Adam optimizer for adversarial training on MNIST and SGD with momentum $= 0.9$ on CIFAR10.

For all the adversarial distillation experiments, we use FGSM attacks for creating adversarial examples during training. We treat the choice of initial perturbation as a hyperparameter and consider zero and uniformly random starting point between $-\epsilon$ and ϵ. We note that a FGSM step from non-zero initial state with step size $\alpha = \epsilon$ is not guaranteed to lie on the boundary of ℓ_∞-ball with radius ϵ and hence this attack can be considered weak and this would lead to a potentially weaker defense. So, in our experiments we use a larger step size $\alpha = 1.25 \times \epsilon$ which has shown to increase the robustness of adversarial training (Wong et al., 2020).

4.1.3 Model Selection

We observed that FGSM based adversarial distillation can lead to overfitting towards FGSM based attacks where the adversarial accuracy with respect to the PGD attack suddenly and drastically drops down to 0% (on the training data) leaving no ability to generalize towards PGD attacks. To avoid this failure mode, we select the best model, based on the PGD attack success rate on a training minibatch of size 1000 examples, after running the adversarial distillation for N epochs.

4.2 Robustness against White-box Attacks

We study the adversarial robustness of SGLD Monte Carlo ensembles and distilled BDK student models. We compare their performance to standard point estimation of the same model (we use Adam as the optimizer). Figure 1 compares the performance of standard, SGLD and BDK models on MNIST and CIFAR10. Note that lower attack success rate implies higher adversarial robustness. We can see that the SGLD Monte Carlo ensembles are significantly more robust to the adversarial attack than the standard models. We also observe that the adversarial robustness as well as accuracy of SGLD models increase as the number of models in the ensemble increases. We see that the BDK student models only provide a marginal increase in robustness as the amount of noise used for distillation increases. However, this comes at the price of accuracy as shown in Figures 1c and 1f where we observe that test accuracy on unperturbed test data decreases with increasing noise for BDK student models.

4.3 Robustness of Adversarial Distillation

We compare the performance of adversarially distilled student models to BDK student models and the SGLD Monte Carlo ensemble used during adversarial distillation. We use random gaussian noise of low variance to augment the training dataset during the standard BDK distillation. The BDK student (noise_std $= 0.1$) used in this experiment corresponds to the most robust BDK model. We investigate an additional baseline where we use a noise uniform sampled within ℓ_∞-ball of radius ϵ for distillation. We use SGLD Monte Carlo ensemble of 100 models[1] for adversarial distillation. We only perform adversarial distillation with the highest ℓ_∞ perturbation, i.e. 0.3 for MNIST and 0.03 for CIFAR10.

Figure 2 compares the performance of adversarial distillation to standard, SGLD and BDK student models on MNIST and CIFAR10. We observe that adversarial distillation leads to significantly robust model but it comes at the cost of accuracy (particularly on CIFAR10). Interestingly, adversarial distillation leads to more robust model than the SGLD Ensmeble itself. The adversarially distilled model chosen for comparison here corresponds to the highest PGD success rate on a small training batch. We note that model selection provides additional flexibility of trading-off adversarial robustness for more accurate models. Table 1 and 2 compare the attack success rate of all the models under FGSM and PGD attack with $\epsilon = 0.3$ for MNIST and $\epsilon = 0.03$ for CIFAR10.

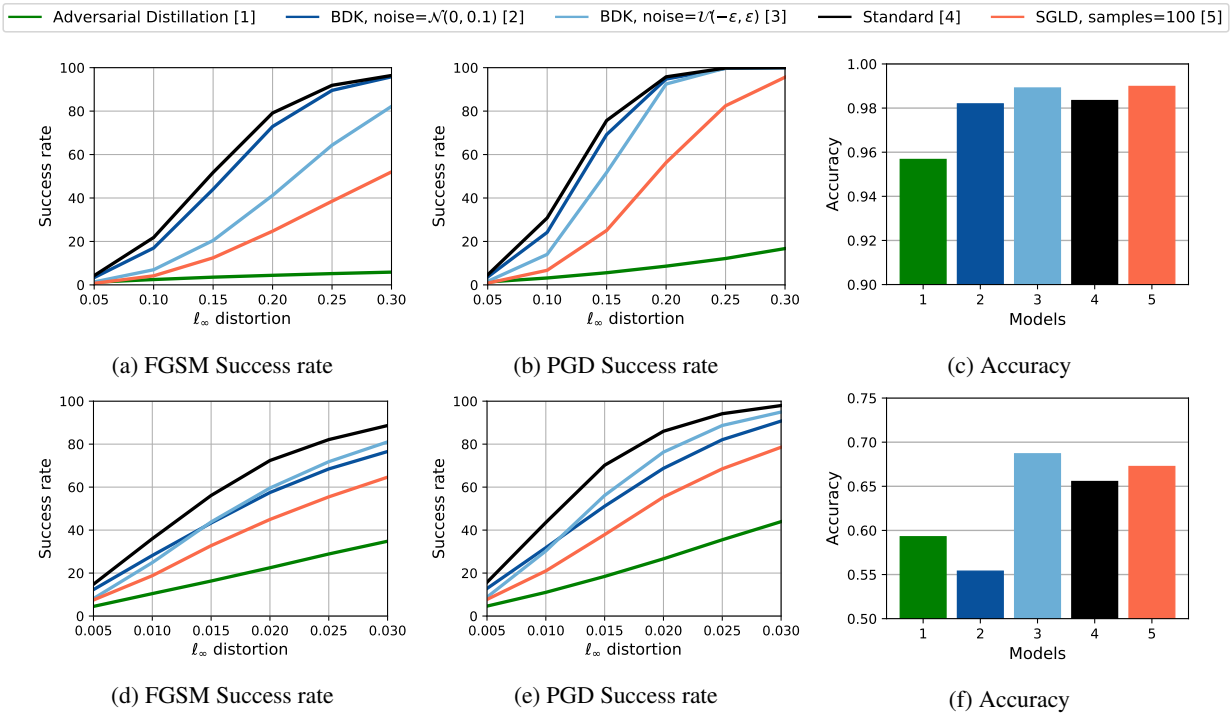
Table 1: ℓ_∞ attack success with $\epsilon = 0.3$ on MNIST. Lower scores correspond to better performance or higher robustness

Models	FGSM	PGD
Adversarial Distillation	$\mathbf{5.88\%}$	$\mathbf{16.72\%}$
BDK, $\mathcal{N}(0, 0.1)$	95.79%	100.0%
BDK, $\mathcal{U}(-\epsilon, \epsilon)$	82.17%	100.0%
Standard	96.39%	100.0%
SGLD, 100 samples	52.02%	95.59%

4.4 Robustness against Black-box Attacks

In contrast to the white-box setting where the adversary has complete knowledge of the architecture and parameters of the target network, black-box setting requires an

[1] Larger size emsembles lead to higher memory storage cost and slows the training process.



(a) FGSM Success rate (b) PGD Success rate (c) Accuracy

(d) FGSM Success rate (e) PGD Success rate (f) Accuracy

Figure 2: Comparison of Adversarial Distillation for ℓ_∞ white-box attacks on MNIST (top) and CIFAR10 (bottom).

Table 2: ℓ_∞ attack success with $\epsilon = 0.03$ on CIFAR10. Lower scores correspond to better performance or higher robustness

Models	FGSM	PGD
Adversarial Distillation	**34.82%**	**43.94%**
BDK, $\mathcal{N}(0, 0.1)$	76.57%	90.73%
BDK, $\mathcal{U}(-\epsilon, \epsilon)$	81.04%	94.94%
Standard	88.66%	97.96%
SGLD, 100 samples	64.63%	78.56%

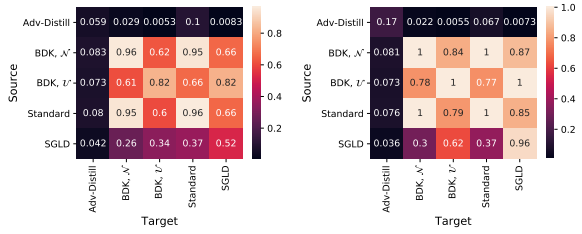
attacker to find an adversarial perturbation without such knowledge: information about the network can be obtained only through querying the target network. Specifically, we are interested in transfer-based attacks (Papernot et al., 2016; Liu et al., 2016), where an adversary trains a second, fully-observable substitute network, attack this network with white-box methods, and transfer these attacks to the original target network.

We investigate the robustness of adversarial distillation (Adv-Distill), BDK student models and SGLD ensembles against black-box attacks. For each source model, we generate adversarial examples using FGSM and PGD and examine if they are also adversarial to target models. We use ℓ_∞ perturbation bound of 0.3 and 0.03 for MNIST and CIFAR10 respectively. We report the success rate of transfer attacks in Figure 3. The diagonal values represent their respective attack success rate and the non-diagonal elements are attack success rate of the target model on the adversarial images generated using the source model.

We observe that our proposed model Adv-Distill is the hardest to attack overall. Surprisingly, SGLD ensembles seem to be easily attackable using examples generated from any model except Adv-Distill, particularly BDK student with uniform noise but the reverse is not true i.e. adversarial examples generated using SGLD ensembles are not transferable. In case of MNIST, Standard and BDK with Gaussian noise are essentially the same.

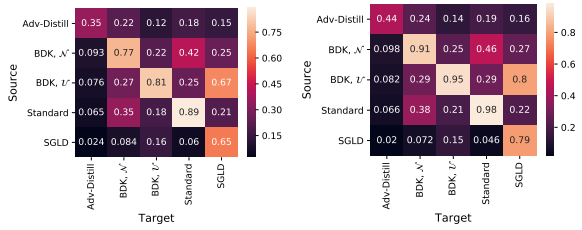
4.5 Ablation Study

Our proposed adversarial distillation excludes the perturbed examples which are adversarial to SGLD Ensemble during training. Table 3 compares the adversarial accuracy (accuracy on adversarially perturbed test set) of adversarial distillation for both the cases when adversarial examples to Ensemble are included and excluded. We see a clear drop in adversarial accuracy when the examples adversarial to SGLD Ensemble are included during training. We also study the effect of random and zero initialization during adversarial distillation in Table 4.



(a) FGSM Success rate

(b) PGD Success rate



(c) FGSM Success rate

(d) PGD Success rate

Figure 3: Transfer attacks with $\epsilon = 0.3$ for MNIST (top row) and $\epsilon = 0.03$ for CIFAR10 (bottom row).

Table 3: Comparison of standard and adversarial accuracy with $\epsilon = 0.3$ on MNIST and $\epsilon = 0.03$ on CIFAR10. Higher scores correspond to better performance.

	Adversarial to Teacher	**MNIST**	**CIFAR10**
Standard Accuracy	Excluded	95.67	59.28
	Included	94.79	55.13
FGSM	Excluded	91.66	30.80
	Included	89.80	25.45
PGD	Excluded	80.10	25.21
	Included	78.05	22.32

5 Conclusions

We have considered the problem of assessing the adversarial robustness of deep neural network models under both the Markov Chain Monte Carlo (MCMC) and Bayesian Dark Knowledge (BDK) inference approximations. Interestingly, our results show that full MCMC-based inference has excellent robustness, significantly outperforming standard point estimation-based learning, while BDK provides marginal improvement. We further present adversarial distillation: adversarial training method to improve the robustness of BDK. Experiments on MNIST and CIFAR10 successfully demonstrate the improved robustness of adversarial distillation. We also show that the adversarial distillation leads to model which are robust against black-box transfer attacks.

Table 4: Comparison of standard and adversarial accuracy with $\epsilon = 0.3$ on MNIST and $\epsilon = 0.03$ on CIFAR10 with random and zero initialization. Higher scores are better.

	Initialization	**MNIST**	**CIFAR10**
Standard Accuracy	Random	95.82	57.79
	Zero	95.67	59.28
FGSM	Random	89.82	29.71
	Zero	91.66	30.80
PGD	Random	79.19	24.82
	Zero	80.10	25.21

References

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

W. Brendel, J. Rauber, and M. Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

N. Carlini and D. A. Wagner. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016.

P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 15–26. ACM, 2017.

M. Cheng, T. Le, P.-Y. Chen, J. Yi, H. Zhang, and C.-J. Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. *arXiv preprint arXiv:1807.04457*, 2018.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2018.

Y. Gal and L. Smith. Sufficient conditions for idealised models to have no adversarial examples: a theoretical and empirical study with bayesian neural networks. *arXiv preprint arXiv:1806.00667*, 2018.

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

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2015.

A. Graves, N. Jaitly, and A.-r. Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE, 2013a.

A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013b.

G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2016.

A. Ilyas, L. Engstrom, A. Athalye, and J. Lin. Black-box adversarial attacks with limited queries and information. *arXiv preprint arXiv:1804.08598*, 2018.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

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

A. Kurakin, I. J. Goodfellow, and S. Bengio. Adversarial machine learning at scale. 2017. URL `https://arxiv.org/abs/1611.01236`.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

X. Liu, Y. Li, C. Wu, and C.-J. Hsieh. Adv-BNN: Improved adversarial defense through robust bayesian neural network. In *International Conference on Learning Representations*, 2019.

Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

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

S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

R. M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, Berlin, Heidelberg, 1996. ISBN 0387947248.

A. M. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *CVPR*, pages 427–436. IEEE Computer Society, 2015. ISBN 978-1-4673-6964-0.

N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. *CoRR*, abs/1602.02697, 2016.

S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet. Adversarial manipulation of deep representations. *arXiv preprint arXiv:1511.05122*, 2015.

A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein. Adversarial training for free! In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 3358–3369. Curran Associates, Inc., 2019.

L. N. Smith. Cyclical learning rates for training neural networks, 2015. cite arxiv:1506.01186Comment: Presented at WACV 2017; see https://github.com/bckenstler/CLR for instructions to implement CLR in Keras.

L. N. Smith and N. Topin. Super-convergence: Very fast training of residual networks using large learning rates. *CoRR*, abs/1708.07120, 2017.

C. Szegedy, W. Zaremba, I. S. J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *In ICLR*, 2014.

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

E. Wong, L. Rice, and J. Z. Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2020.

N. Ye and Z. Zhu. Bayesian adversarial learning. In *NeurIPS*, 2018.