Evaluating the Spectrum of Classifier Robustness

Anonymous Author(s)
Affiliation
Address
email

Abstract

Several recent works in machine learning have focused on evaluating the test-time robustness of a classifier: how well the classifier performs not just on the target domain it was trained upon, but upon perturbed examples. In these settings, the focus has largely been on two extremes of robustness: the robustness to perturbations drawn *at random* from within some distribution (i.e., robustness to random perturbations), and the robustness to the *worst case* perturbation in some set (i.e., adversarial robustness). In this paper, we argue that a sliding scale between these two extremes provides a valuable additional metric by which to gauge robustness. Specifically, we illustrate that each of these two extremes are naturally characterized by a (functional) p-norm over perturbation space, with $p = 1$ corresponding to robustness to random perturbations and $p = \infty$ corresponding to adversarial perturbations. We then present the main technical contribution of our paper: a method for efficiently estimating the value of these norms by interpreting them as the partition function of a particular distribution, then using MCMC methods to estimate this partition function (either traditional Metropolis-Hastings for non-differentiable perturbations, or Hamiltonian Monte Carlo for differentiable perturbations). We show that our approach provides substantially better estimates than simple random sampling of the actual "intermediate-p" robustness of standard, data-augmented, and adversarially-trained classifiers, illustrating a clear tradeoff between classifiers that optimize different metrics.

1 Introduction

There has been an increasing focus in recent years in evaluating the *robustness* of machine learning classifiers, broadly interpreted as evaluating their performance not just on a test set, but also evaluating the performance relative to some additional (possibly domain-specific) uncertainty or bounds on the problems. Although there are many formal definitions of robustness, most work in this area has focused on two particular settings. In the "classical" sense of robustness, we can consider evaluating the classifier in terms of its *worst-case* loss under some perturbation set applied to the inputs, i.e., we could evaluate (via finite sample approximation)

$$\mathbf{E}_{x,y\sim\mathcal{D}}\left[\max_{\delta\in\Delta(x)} \ell(h(x+\delta), y)\right] \tag{1}$$

where $\mathcal{D}$ denotes a distribution over x, y pairs, h denotes the hypothesis function, ℓ denotes a loss, and $\Delta(x)$ denotes some (input-dependent) *uncertainty region*. This formulation, for instance, underlies adversarial examples and also motivates the classical adversarial training approaches. However, substantial work has also been done in evaluating the setting of robustness to *random* perturbations, i.e., evaluating a classifier via the loss

$$\mathbf{E}_{x,y\sim\mathcal{D}}\left[\mathbf{E}_{\delta\sim\mathcal{P}(x)}[\ell(h(x+\delta), y)]\right] \tag{2}$$

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

where now $\mathcal{P}(x)$ denotes some (again, input-dependent) *distribution* over possible perturbations. This formulation underlies common data augmentation strategies in deep learning, as well as most formulations of "natural" robustness [Hendrycks and Dietterich, 2018] (even if not always written in this formal manner).

In this paper, we advocate for a more fine-grained spectrum of robustness definitions, which naturally interpolates between both these two extremes. In particular, we argue that robustness to random perturbations and worst-case robustness are naturally interpreted as (functional) ℓ_p norms[1] of the loss function evaluated over the perturbation distribution (which can be a uniform distribution in the case of traditional adversarial loss). In particular, the random setting corresponds to the choice of $p = 1$ and the adversarial setting corresponds to the choice of $p = \infty$. We believe that it is also valuable and informative to consider the performance of classifiers in a wide range in between these two extremes, i.e., the performance of "intermediate-p" robustness. However, evaluating this intermediate-p robustness is non-trivial, owing to the fact that it requires computing a high dimensional integral over the perturbation space. Thus, our main technical contribution of this paper is the proposal of a simple approach to evaluating the relevant robustness norms, using a combination of path sampling and Markov chain Monte Carlo (MCMC) approaches (specifically the Hamiltonian Monte Carlo method for cases where loss is a differentiable function of the perturbation space). Despite their seeming complexity, in this particular case the eventual estimators take the very simple form of a geometric mean computed over samples from an annealed distribution over the perturbation region.

We evaluate our approach on networks trained via standard training, data augmentation, and adversarial training. In all the cases our proposed approach shows a clear trade-off between different levels of robustness that would missed by solely considering just the random or adversarial perturbation setting. Furthermore, we show that our path sampling and HMC-based estimator for this robustness is vastly superior to naive estimates produced e.g. by Monte Carlo sampling. We also briefly highlight the possibility of actually *training* networks using these estimators to create classifiers more robust to these intermediate notions of robustness.

2 Background and related work

Robustness in deep learning Adversarial robustness is the "classicial" view of robustness that traces its roots back to robust optimization, a field in optimization theory that allows for the study of worst-case uncertainty [Ben-Tal et al., 2009]. Szegedy et al. [2013] noted the marked presence of adversarial examples in deep classifiers, in which the addition of imperceptible amounts of noise leads to gross misclassification of input images by deep networks. Goodfellow et al. [2014] then introduced the Fast Gradient Sign Method (FGSM), which uses a single gradient step to generate adversarial examples and was used to perturb training examples as the original form of adversarial training. Subsequently, the Basic Iterative Method was introduced by Kurakin et al. [2016], and rendered FGSM adversarial training ineffective through attacks using multiple, smaller FGSM steps (although the robustness of FGSM adversarial training was later improved [Wong et al., 2020, Andriushchenko and Flammarion, 2020]). The work of Madry et al. [2017] further improved upon this iterative attack, and incorporated the resulting Projected Gradient Descent (PGD) adversary into adversarial training. Today, PGD remains the most empirically effective adversary for adversarial training and attacks. While traditional adversarial training focuses on perturbations within some ℓ_p-norm ball, it has also been expanded to more realistic threat models, from spatial transformations [Engstrom et al., 2019, Xiao et al., 2018] to weather corruptions and lighting changes [Wong and Kolter, 2020].

However, when considering robustness to more "realistic" (yet synthetic) perturbations, the focus is often shifted from robustness to the worst-case perturbation within some set to robustness to random perturbations [Geirhos et al., 2018, Hendrycks and Dietterich, 2018, Hendrycks et al., 2019b,a, Yin et al., 2019]. It was been further studied whether such synthetic corruptions can universally improve robustness to real-world distribution shifts, such as geographic location, or camera hardware [Hendrycks et al., 2020]. Other naturally occurring distribution shifts of interest include consistency shifts [Shankar et al., 2019] and dataset shifts [Storkey, 2009, Taori et al., 2020]. For example, the impact of dataset shift has been emphasized on the ImageNet dataset, where the performance of

[1] We should emphasize that this use of ℓ_p norms is entirely orthogonal to the use of ℓ_p balls as perturbation regions, commonly done in adversarial robustness. The ℓ_p norms here can be applied to any perturbation region, as we will highlight.

models trained on ImageNet was shown to be drastically worse on a reproduction of the validation set [Recht et al., 2019]. Such work illustrates the divide between studies on adversarial robustness and robustness to random data perturbations.

MCMC methods for partition function estimation Because our definition of intermediate-p robustness involves computation of a high dimensional integral over a perturbation distribution, our work can also be related to methods for estimating an intractable partition function of a probability distribution. The problem of computing normalizing constants ties to the problem of computing differences in free energy in physics. Several Monte Carlo sampling-based approaches exist for approximately estimating normalizing constants. Importance sampling is one approach at estimating ratios of normalizing constants that relies on samples from one of the distributions. Bridge sampling [Bennett, 1976, Meng and Wong, 1996], on the other hand, uses a single "bridge" distribution to interpolate between the two distributions and can reduce the Monte Carlo errors associated with importance sampling. Annealed importance sampling [Jarzynski, 1997, Neal, 2001] bridges the gap between the two distributions by chaining together intermediate distributions, and linked importance sampling Neal [2005] then combined bridge sampling with annealed importance sampling to bridge these intermediate distributions. Path sampling [Gelman and Meng, 1998], which originated under the name of thermodynamic integration in physics [Ogata, 1989], aims at reducing Monte Carlo errors by introducing flexible paths.

Our proposed approach also leverages Hamiltonian Monte Carlo methods to improve the sample efficiency of our eventual estimator [Duane et al., 1987, Betancourt, 2017, Neal et al., 2011]. Hamiltonian Monte Carlo (HMC) is useful due to its avoidance of random walks, and is compatible with sampling in a constrained space through use of reflection [Afshar et al., 2015].

3 A unified characterization of the spectrum of robustness

To begin, we first make the simple observation that there is a natural spectrum between the notions of adversarial robustness and robustness to random perturbations. Specifically, we note that both these notions can be expressed as function ℓ_p norms over the perturbation space. Specifically, we define the following functional norm

Definition 1. *For a function $f : \mathbb{R}^n \to \mathbb{R}$ and density $\mu : \mathbb{R}^n \to \mathbb{R}_+$, $\int d\mu(x) = 1$, let $\|f\|_{\mu,p}$ be the p-norm of the function under this density*

$$\|f\|_{\mu,p} = \mathbf{E}_{x\sim\mu}\left[|f(x)|^p\right]^{1/p} = \left(\int |f(x)|^p d\mu(x)\right)^{1/p}. \tag{3}$$

Then as the proposition shows, random perturbation loss and adversarial perturbation loss simply correspond to two extremes of this functional norm over the perturbation δ, as formalized by the following proposition.

Proposition 1. *Let δ be a random variable with density μ and consider the expectation*

$$\mathbf{E}_{x,y\sim D}\Big[\|\ell(h(x+\delta), y)\|_{\mu,p}\Big]. \tag{4}$$

Then this corresponds to the expected loss on random samples from μ when $p = 1$, and to the expected adversarial loss over the domain of μ when $p = \infty$.

This proposition follows immediately from the fact that losses are non-negative, and the fact that the ℓ_∞ norm is given by the pointwise maximum of the function. Note that "traditional" adversarial loss actually arises more specifically when $p = \infty$ and μ is a uniform distribution over some norm ball (unrelated to the norm p). When we have $1 < p < \infty$, we enable a full spectrum of robustness measurements, which we refer to as *intermediate-p robustness*, that evaluates the performance of classifiers in a wide range in between these two extreme cases. Furthermore, we argue that there are naturally appealing properties of these intermediate-p robustness measures: whereas adversarial robustness may overestimate the risk of what effectively amount to "measure zero" regions of the perturbation space, random robustness may likewise fail to take into account smaller but non-negligible regions that *do* contain areas of high loss.

4 Estimating intermediate-p robustness via MCMC methods

Of course, simply writing the loss in this manner is not particularly useful on its own. In most cases, the integral in Equation 3 cannot be computed exactly, and so we must resort to numerical approximation methods. Specifically, in order to estimate the p-norm robustness for an arbitrary nonlinear model h, loss ℓ, and (known) density μ, we consider the problem of computing the integral

$$Z := \left(\int \ell(h(x+\delta), y)^p d\mu(\delta) \right)^{(1/p)}. \tag{5}$$

One could naively estimate this integral by using Monte Carlo sampling, and sample $\delta^{(1)}, \ldots, \delta^{(m)}$ randomly from density μ, and approximate the objective as the following,

$$\hat{Z}_{\text{random}} = \Big(\frac{1}{m} \sum_{i=1}^{m} \ell(h(x+\delta^{(i)}), y)^p \mu(\delta^{(i)}) \Big)^{(1/p)}. \tag{6}$$

However, because the integral in 5 will be dominated by values with large $\ell(h(x+\delta), y)$, simply using Monte Carlo sampling will be insufficient to approximate this integral well for larger values of p, as random sampling will place too much weight on regions of low loss.

Instead, in this work we argue that it is beneficial to interpret the task at hand as one of evaluating the *partition function* or a particular unnormalized probability density. Specifically, we define an (unnormalized) density over the perturbation δ

$$\tilde{p}(\delta) = \ell(h(x+\delta), y)^p \mu(\delta). \tag{7}$$

Then clearly, just from construction, we see that the task of evaluating the partition function (the normalizing constant) of this distribution is exactly the same as that of computing the integral of interest.

The advantage of this perspective on the integral of interest, however, is that we can use a wealth of techniques developed for partition function estimation in order to better estimate this particular integral. Specifically, we argue to use the use path sampling [Gelman and Meng, 1998] approach, a Markov chain Monte Carlo based method, to approximate the desired partition function. In fact, we show that for the precise form of the integral in question, the eventual estimator produced by this method takes on a very simple form: it consists of a geometric mean over samples generated from a certain annealed distribution. This makes the estimator particularly simple to implement and even in some cases to train networks based upon, via standard automatic differentiation toolkits.

4.1 Evaluating the particular function of the loss-based distribution

We state our main result on the path sampling formulation of the integral of interest via the following theorem.

Theorem 1. *Consider the task of approximately computing the following integral:*

$$\left(\int \ell(h(x+\delta), y)^p d\mu(\delta) \right)^{(1/p)}.$$

Let $\{\theta^{(1)}, \theta^{(2)}, \cdots, \theta^{(m)}\}$ be m scalars corresponding to linearly interpolated values from 0 to p. For $i = 1, \ldots, m$, sample $\delta^{(i)}$ from the following unnormalized density

$$\delta^{(i)} \sim p(\delta|\theta^{(i)}), \text{ where, } p(\delta|\theta) \propto \ell(h(x+\delta), y)^{\theta} \mu(\delta).$$

Then the following estimator, given by the geometric mean of the resulting samples

$$\hat{Z}_{path} := \Big(\prod_{i=1}^{m} \ell(h(x+\delta^{(i)}), y) \Big)^{1/m}. \tag{8}$$

is a consistent estimator of our integral Z. Practically, the m samples can be drawn using MCMC.

Proof. For notational completeness, we define the density

$$p(\delta|\theta) = \frac{1}{z(\theta)}\tilde{p}(\delta|\theta) \tag{9}$$

where

$$\tilde{p}(\delta|\theta) = \ell(h(x+\delta), y)^{\theta}\mu(\delta). \tag{10}$$

and where $z(\theta)$ denotes the partition function of this distribution

$$z(\theta) = \int \tilde{p}(\delta|\theta). \tag{11}$$

Taking the log of both sides of and differentiating with respect to θ, we get:

$$\begin{aligned}
\frac{d}{d\theta}\log(z(\theta)) &= \frac{d}{d\theta}\log\int \tilde{p}(\delta|\theta)d\delta \\
&= \frac{\int \frac{d}{d\theta}\tilde{p}(\delta|\theta)d\delta}{\int \tilde{p}(\delta|\theta)d\delta} \\
&= \int \frac{\tilde{p}(\delta|\theta)}{z(\theta)}\frac{\frac{d}{d\theta}\tilde{p}(\delta|\theta)}{\tilde{p}(\delta|\theta)}d\delta \\
&= \mathbf{E}_{\delta\sim p(\delta|\theta)}\Big[\frac{d}{d\theta}\log\tilde{p}(\delta|\theta)\Big] \\
&= \mathbf{E}_{\delta\sim p(\delta|\theta)}\Big[\log\ell((h(x+\delta), y))\Big]
\end{aligned} \tag{12}$$

where we use the fact that

$$\log\tilde{p}(\delta|\theta) = \theta\log\ell((h(x+\delta), y)). \tag{13}$$

Then we can integrate (12) from 0 to p to get

$$\log\Big[\frac{z(p)}{z(0)}\Big] = \int_0^p \mathbf{E}_{\delta\sim p(\delta|\theta)}\Big[\log(\ell(h(x+\delta), y))\Big] \tag{14}$$

Given in our case $z(0) = 1$ and incorporating the exponent of $(1/p)$, we have

$$\log\Big[z(p)^{(1/p)}\Big] = \frac{1}{p}\int_0^p \mathbf{E}_{\delta\sim p(\delta|\theta)}\Big[\log(\ell(h(x+\delta), y))\Big] \tag{15}$$

Then the right hand side of Equation 15 can be interpreted as the expectation of $\log(\ell(h(x+\delta), y))$ over the joint distribution of (δ, θ), where θ is a random variable with a uniform distribution in $[0, p]$.

$$\log\Big[z(p)^{(1/p)}\Big] = \mathbf{E}_{\theta\sim U[0,p]}\Big[\mathbf{E}_{\delta\sim p(\delta|\theta)}\Big[\log(\ell(h(x+\delta), y))\Big]\Big] \tag{16}$$

Finally, sampling $(\delta^{(i)}, \theta^{(i)})$ for $i = 1, \ldots, m$ from this joint distribution $p(\delta, \theta)$ (which we can do by linearly interpolating $\theta^{(i)}$ between 0 and p and then sampling $\delta^{(i)}$ from $p(\delta|\theta^{(i)})$), we have the fact that

$$\hat{Z}_{\text{path}} := \exp\Big(\frac{1}{m}\sum_{i=1}^{m}\log(\ell(h(x+\delta^{(i)}), y)\Big) = \Big(\prod_{i=1}^{m}\ell(h(x+\delta^{(i)}), y)\Big)^{1/m}, \tag{17}$$

is a consistent estimator of the desired integral. □

The key point of this result is that it allows us to approximate the desired integral just through the ability to *sample* from the distribution $p(\delta|\theta)$. While this is still a challenging task, sampling from unnormalized probability distributions is a well-studied problem, and we can apply MCMC sampling methods to this task. Further, while the sampling of $(\delta^{(i)}, \theta^{(i)})$ can be done in different ways, we choose to linearly anneal $\theta^{(i)}$ from 0 to p, and then draw $\delta^{(i)} \sim \tilde{p}(\delta|\theta)$ using some MCMC sampler. This has the nice feature that it starts with sampling from an "easy" distribution (when $\theta = 0$, the distribution over δ is simply given by μ), and gradually anneals to a more peaked distribution as θ increases. The resulting algorithm for evaluating a network using this geometric mean estimator is shown in Algorithm 1.

Algorithm 1 Path sampling evaluation of a neural network function h with m MCMC samples with $x, y \sim D$ for some norm p.

Initialize $\delta^{(0)}$ randomly
for $i = 1 \dots m$ **do**
  Let $\theta^{(i)} := p \cdot \frac{i-1}{m-1}$
  Sample $\delta^{(i)} \sim p(\delta|\theta^{(i)})$ using MCMC from initial state $\delta^{(i-1)}$
end for
return $\left(\prod_{i=1}^{m} \ell(h(x + \delta^{(i)}), y) \right)^{1/m}$

4.2 Sampling from the (unnormalized) loss distribution via HMC

In order to generate the samples for the path sampling estimation (17) from the desired distribution $p(\delta, \theta)$, we use Markov chain Monte Carlo (MCMC) methods to sample δ from the unnormalized distribution $\tilde{p}(\delta|\theta)$. When the loss is a differentiable function of the perturbation distribution, we can take advantage of gradient-based methods to reduce random walk behavior in MCMC sampling and achieve more efficient sampling. Hamiltonian Monte Carlo (HMC) is one such gradient-based MCMC method that simulates Hamiltonian dynamics to improve sample efficiency in high-dimensional spaces. HMC is based on the Hamiltonian function $H(q, p) = U(q) + K(p)$, where q is a d-dimensional position vector, p is a d-dimensional momentum vector, $U(q)$ is the potential energy, and $K(p)$ is the kinetic energy. To translate this to our setting, $U(\delta) = -\log(\ell(x + \delta), y)^{\theta} \mu(\delta))$ is just the negative log probability density of the distribution we want to sample from, and $K(p) = ||p||^2/(2\sigma^2)$ is the negative log probability density of the zero-mean Gaussian distribution with variance σ^2. Then the Hamiltonian function H is equal to the following:

$$H(\delta, p) = -\theta \log(\ell(h(x + \delta), y)) + \log \mu(\delta) + \frac{||p||^2}{\sigma^2} \tag{18}$$

In order to make use of Hamiltonian dynamics in practice, Hamiltonian's equations must be discretized. The Leapfrog method is one such discretization of Hamiltonian dynamics, using a small stepsize α to discretize time and numerically integrating the system of differential equations as follows:

$$\begin{aligned} p &= p + \alpha\theta\nabla_\delta \log(\ell(h(x + \delta), y))/2 \\ \delta &= \delta + \alpha p/\sigma^2 \\ p &= p + \alpha\theta\nabla_\delta \log(\ell(h(x + \delta), y))/2 \end{aligned} \tag{19}$$

The leapfrog method is reversible by simply negating p (although this need not be done in practice, since $K(p) = K(-p)$). The HMC algorithm begins by first sampling a new momentum vector $p \sim \mathcal{N}(0, \sigma^2)$, independent of the current state of δ. Then, the Leapfrog method in (19) is repeated for L steps to propose a new Markov state (δ', p'). This proposed state is then accepted with probability $\min[1, \exp(-\Delta H)]$, where $\Delta H = H(\delta', p') - H(\delta, p))$. If the state (δ', p') is not accepted, the next state is then set to (δ, p).

Given that in general, our perturbation distribution will likely be constrained in some manner, we need to modify the HMC algorithm so that our proposals remain within the boundaries of the perturbation distribution. Consider the case where our perturbation distribution is the ℓ_∞ norm ball with radius ϵ, so that each element of δ is constrained to be within $-\epsilon$ and ϵ. In order to enforce these constraints while preserving the dynamics, we incorporate what is known as *reflection* in HMC. In this case, after setting $\delta = \delta + \epsilon p/\sigma^2$, we check if any $\delta'_i > \epsilon$ or $\delta'_i < -\epsilon$ for $i = 1, \dots, n$. If so, we negate the corresponding momentum term, p_i, and if $\delta'_i > \epsilon$, we set $\delta_i = 2\epsilon - \delta'_i$, whereas if $\delta'_i < -\epsilon$, we set $\delta'_i = -2\epsilon - \delta'_i$. We repeat this reflection step until δ_i satisfies our constraints. One can think of this behavior as effectively simulating reflecting off a physical boundary.

We note that we can still easily use the path sampling estimator for non-differentiable perturbations by replacing the Hamilton Monte Carlo sampler with any other non-gradient based MCMC sampler, such as a random walk Metropolis. This is roughly similar to random sampling, but with a more subtle weighting on terms that involve higher loss.

4.3 Estimating the partition function during training

We further consider the possibility of *training* networks using the estimators we have discussed to achieve more intermediate robustness. However, this becomes less computationally feasible due to the number of samples required to get a good estimate of the objective, due to the $m \times L$ iterations of the path sampling estimator with Hamiltonian Monte Carlo, where L is the number of Leapfrog steps. Additionally, the step size α and variance σ^2 in HMC require careful tuning. However, for larger values of p, the path sampling estimator is essential to getting good estimates of the training objective in this case, as random sampling will be much less likely to come across regions of the perturbation distribution with high loss. Path sampling, on the other hand, draws samples from the unnormalized loss distribution using MCMC, and so with increased p, there will be higher weighting on samples that induce higher loss. The Hamiltonian Monte Carlo method has the additional benefit of following the gradient (along with some noise), and so for larger p, even with a small number of iterations, path sampling can have advantages over random sampling during training.

5 Experimental evaluation

In this section, we evaluate the intermediate-p robustness of standard training, data augmentation, and adversarial training, and compare estimations using the naive, random sampling estimator (6) and the path sampling estimator (8) with Hamiltonian Monte Carlo (in the differentiable case), and random walk Metropolis (in the non-differentiable case). We additionally show preliminary results of training using these estimators. All of our experiments are either run on the MNIST dataset [LeCun et al., 1998] or the CIFAR-10 dataset [Krizhevsky et al., 2009]. We consider robustness over two perturbation distributions: the ℓ_∞ norm ball and spatial transformations. Results reported in this section are for a single random seed. The mean and standard deviation over repeated runs can be found in Appendix A.1, along with training details in Appendix A.3.

5.1 Robustness over the ℓ_∞-norm ball

In order to easily translate the notion of intermediate-p robustness to that of adversarial robustness, we consider perturbations uniformly distributed within an ℓ_∞-norm ball with radius ϵ. Our evaluations of intermediate-p robustness on MNIST (Table 1) and CIFAR-10 (Table 2) both show that the estimations of the functional ℓ_p norm of the loss function evaluated over the perturbation distribution naturally interpolate between loss over random samples (corresponding to RS $p = 1$) and adversarial (Projected Gradient Descent) loss (corresponding to PGD-50). For larger values of p, specifically $p = 100$ and $p = 1000$, the HMC-based path sampling estimator consistently produces much better estimates when compared to the random sampling (RS) estimator. We show that this result holds for all of the models we evaluated, each of which were trained using a different training objective, and across both datasets. The advantage of the path sampling estimator over the random sampling estimator is further illustrated in Figure 1, where we show the convergence of the their corresponding estimates on a single test batch for a standard trained model given increasing number of samples. While for $p = 1$ the estimates converge to the same value given the same number of iterations, for $p = 100$, the estimates quickly diverge, with path sampling producing a much higher estimate.

In addition to evaluating intermediate-p robustness using both estimators, we additionally show the potential benefits of training using these estimators. We train using the random sampling estimator on both MNIST (with $m = 50$) and CIFAR-10 (with $m = 10$). We note that training using random sampling with $p = 1$ corresponds to just applying m random perturbations per example and taking the average loss, similar to data augmentation methods. We show in Tables 1 and 2 that even training using this naive estimator can produce better intermediate-p robustness than existing training methods. We also train using the HMC-based path sampling estimator on MNIST (with $m = 25$ with $L = 2$ leapfrog steps in order to compare an equivalent number of iterations), and find that most models trained according to this objective outperform those trained using the random sampling objective for the same value of p in terms of intermediate-p robustness. This suggests that even with a reasonably small number of samples, path sampling can result in a better estimate of the objective. While these training results are rather preliminary, as the computational complexity given the number of samples required to get a reasonable estimate makes training more challenging for larger datasets, it shows promise for training deep networks to robustness levels somewhere between robustness over random perturbations and robustness over worst-case perturbations.

Table 1: Robustness estimates for different evaluation methods on MNIST with perturbations from the ℓ_∞ ball with radius $\epsilon = 0.3$.

Method	Standard	RS $m = 2000$				HMC $m = 100, l = 20$				PGD-100
		$p = 1$	$p = 10$	$p = 10^2$	$p = 10^3$	$p = 1$	$p = 10$	$p = 10^2$	$p = 10^3$	
Standard	0.028	0.037	0.108	0.193	0.204	0.037	0.121	1.136	3.757	10.386
RS $p = 1$	0.033	0.032	0.083	0.141	0.151	0.032	0.086	0.664	2.075	6.951
RS $p = 10$	0.029	0.028	0.060	0.100	0.107	0.028	0.060	0.401	1.291	3.554
RS $p = 10^2$	0.027	0.026	0.056	0.094	0.100	0.026	0.056	0.372	1.189	3.254
RS $p = 10^3$	0.028	0.027	0.057	0.095	0.101	0.027	0.057	0.379	1.223	3.371
HMC $p = 1$	0.036	0.034	0.091	0.156	0.166	0.034	0.095	0.743	2.252	8.078
HMC $p = 10$	0.033	0.031	0.074	0.124	0.131	0.031	0.074	0.447	1.267	4.638
HMC $p = 10^2$	0.034	0.031	0.069	0.114	0.122	0.031	0.067	0.313	0.786	2.867
HMC $p = 10^3$	0.025	0.025	0.048	0.078	0.084	0.025	0.046	0.232	0.648	1.563
PGD-50	0.030	0.036	0.047	0.070	0.075	0.036	0.044	0.091	0.160	0.245

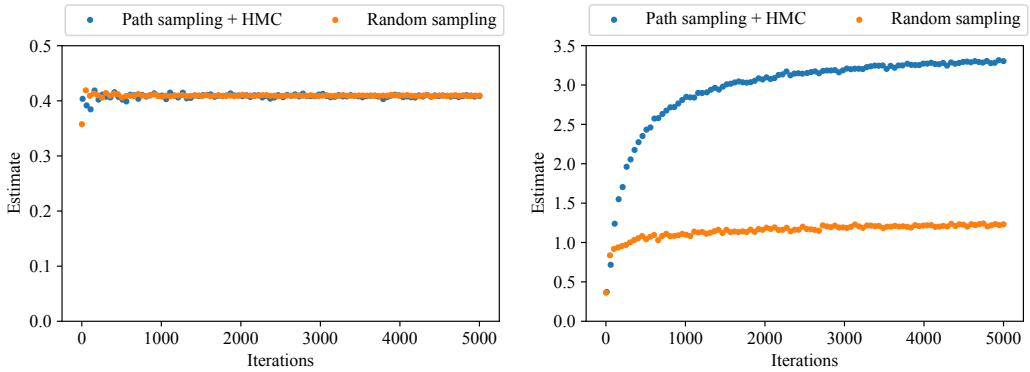
Table 2: Robustness estimates for different evaluation methods on CIFAR-10 with perturbations from the ℓ_∞ ball with radius $\epsilon = 0.03$.

Method	Standard	RS $m = 500$				HMC $m = 50, l = 10$				PGD-50
		$p = 1$	$p = 10$	$p = 10^2$	$p = 10^3$	$p = 1$	$p = 10$	$p = 10^2$	$p = 10^3$	
Standard	0.380	0.449	0.7621	1.113	1.175	0.449	0.811	2.618	4.939	18.148
RS $p = 1$	0.405	0.410	0.539	0.727	0.768	0.410	0.553	1.507	3.158	14.240
RS $p = 10$	0.396	0.401	0.473	0.605	0.637	0.401	0.476	1.054	2.389	12.014
RS $p = 10^2$	0.398	0.401	0.464	0.586	0.617	0.401	0.466	0.974	2.243	11.907
RS $p = 10^3$	0.395	0.401	0.466	0.592	0.624	0.401	0.469	0.997	2.303	12.102
PGD-10	0.724	0.726	0.727	0.737	0.755	0.726	0.727	0.737	0.791	1.415

5.2 Non-differentiable spatial transformations

In order to show that the path sampling estimator can naturally be extended to non-differentiable perturbations, we consider a perturbation set consisting of parameterizations of spatial transformations on CIFAR-10. The parameters of the spatial transformations include horizontal flips, rotations between -10 to 10 degrees, scaling factors between 0.9 to 1.1, and cropping between 0 to 4 pixels horizontally and vertically. We compare the estimates generated by the random sampling estimator and by the path sampling estimator using random-walk Metropolis sampling. The results in Table 3 show that as p is increased, the Metropolis-based path sampling estimator produces better estimates than random sampling. Additionally, for large p, the Metropolis estimator approaches the adversarial loss over the transformation space, suggesting that our estimator can effectively capture intermediate-p notions of robustness. This pattern holds for both standard trained models, as well as models trained using the random sampling estimator. Due to the computational complexity, we only evaluate on the first 1000 test examples on the CIFAR-10 dataset.

We now describe each estimator in more detail. Both the random sampling estimator and the Metropolis-based path sampling estimator take $m = 500$ samples of transformations. For the Metropolis-based path sampling estimator, we initially randomly sample a transformation, and then propose updates to the previous transformation, where the updates are accepted or rejected based on the ratio of the unnormalized loss-based densities. We use the following proposal distributions: 1) perform a horizontal flip with probability 0.5; 2) scale the image by resizing by a factor of $r \sim \mathcal{N}(0, 0.5)$; 3) rotate d degrees with $d \sim \mathcal{N}(0, 5)$; 4) crop x or y in the horizontal or vertical direction, each drawn i.i.d. $\sim \mathcal{N}(0, 2)$ and then rounded to the nearest integer value. Additionally, at each value of the annealed θ, we perform 20 burn-in steps and only keep the last value for estimation purposes. The adversarial estimate was obtained by finding the maximum loss value encountered for each example during the Metropolis sampling process, and then combining these max losses via an average over the test set.



(a) (b)

Figure 1: Comparison of the convergence of HMC-based path sampling and random sampling estimates for (a) $p = 1$, and (b) $p = 100$ with increasing iterations on a standard trained model on CIFAR-10. Iterations corresponds m for the random sampling estimator, and to $m \times L$ for the path sampling/HMC estimator using $L = 10$ leapfrog steps. For a smaller $p = 1$ (a), random sampling works reasonably well, whereas for $p = 100$ (b), using path sampling with HMC is essential to get good estimates.

Table 3: Robustness estimates for different evaluation methods on CIFAR-10 with non-differentiable parameterizations of flips, rotation, translation, and scaling.

Method	Standard	Random Sampling				Metropolis				Adversarial
		$p = 1$	$p = 10$	$p = 10^2$	$p = 10^3$	$p = 1$	$p = 10$	$p = 10^2$	$p = 10^3$	
Standard	0.196	0.434	2.207	3.623	3.749	0.431	2.401	4.540	4.786	5.506
RS-1	0.140	0.189	0.769	1.214	1.267	0.184	0.854	1.560	1.686	1.974
RS-10	0.953	1.034	1.088	1.325	1.397	1.030	1.082	1.380	1.397	1.553
RS-100	2.106	2.129	2.130	2.135	2.163	2.128	2.129	2.134	2.170	2.182

6 Conclusion

In this paper, we proposed a definition of intermediate-p robustness that smooths the gap between robustness to random perturbations and adversarial robustness by generalizing these notions of robustness as functional ℓ_p norms of the loss function over the perturbation distribution. In order to evaluate intermediate-p robustness in practice, we introduced an approach for approximating the high dimensional integral over the perturbation distribution that uses path sampling, an effective estimator based on MCMC sampling. We showed that across different datasets, models trained on different training objectives, and different perturbation distributions (both differentiable and non-differentiable cases), our path sampling approach produces much better estimates of the integral than simple Monte Carlo sampling. Additionally, we illustrated the benefit of using the gradient-based Hamiltonian Monte Carlo method as the MCMC sampler when the loss is differentiable with respect to the perturbation distribution. Lastly, we highlighted the possibility of training using these estimators, and showed that models trained on MNIST using these estimators *do* exhibit improved intermediate-p robustness, with path sampling again outperforming random sampling. However, the best way of estimating the integral during training of deep networks remains an open question, due to the computational complexity of accumulating enough MCMC samples at each epoch to get a reasonable estimation.

References

Hadi Mohasel Afshar, Justin Domke, et al. Reflection, refraction, and hamiltonian monte carlo. In *NIPS*, pages 3007–3015, 2015.

Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. *arXiv preprint arXiv:2007.02617*, 2020.

Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*. Princeton University Press, 2009.

Charles H Bennett. Efficient estimation of free energy differences from monte carlo data. *Journal of Computational Physics*, 22(2):245–268, 1976.

Michael Betancourt. A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv:1701.02434*, 2017.

Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the landscape of spatial robustness. In *International Conference on Machine Learning*, pages 1802–1811. PMLR, 2019.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

Andrew Gelman and Xiao-Li Meng. Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical science*, pages 163–185, 1998.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2018.

Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. *arXiv preprint arXiv:1906.12340*, 2019a.

Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019b.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. *arXiv preprint arXiv:2006.16241*, 2020.

Christopher Jarzynski. Nonequilibrium equality for free energy differences. *Physical Review Letters*, 78(14):2690, 1997.

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

Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. Adversarial examples in the physical world, 2016.

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

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Xiao-Li Meng and Wing Hung Wong. Simulating ratios of normalizing constants via a simple identity: a theoretical exploration. *Statistica Sinica*, pages 831–860, 1996.

Radford M Neal. Annealed importance sampling. *Statistics and computing*, 11(2):125–139, 2001.

Radford M Neal. Estimating ratios of normalizing constants using linked importance sampling. *arXiv preprint math/0511216*, 2005.

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

Yosihiko Ogata. A monte carlo method for high dimensional integration. *Numerische Mathematik*, 55(2):137–157, 1989.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pages 5389–5400. PMLR, 2019.

Vaishaal Shankar, Achal Dave, Rebecca Roelofs, Deva Ramanan, Benjamin Recht, and Ludwig Schmidt. Do image classifiers generalize across time? *arXiv preprint arXiv:1906.02168*, 2019.

Amos J Storkey. When training and test sets are different: characterising learning transfer. In *In Dataset Shift in Machine Learning*, pages 3–28. MIT Press, 2009.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. *Advances in Neural Information Processing Systems*, 33, 2020.

Eric Wong and J Zico Kolter. Learning perturbation sets for robust machine learning. *arXiv preprint arXiv:2007.08450*, 2020.

Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. *arXiv preprint arXiv:1801.02612*, 2018.

Dong Yin, Raphael Gontijo Lopes, Jonathon Shlens, Ekin D Cubuk, and Justin Gilmer. A fourier perspective on model robustness in computer vision. *arXiv preprint arXiv:1906.08988*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 4.3 for the limitations for training.
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 4.1.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Section 4.1.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5 and Appendix A.3.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Appendix A.1.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix A.3.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5.

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]