Robustness to probabilistic perturbations

Mathematical Institute
University of Oxford

Abstract

In the past few years there has been a growing interest in robustness of neural networks to probabilistic perturbations. Several theoretical frameworks, for both evaluation and training purposes, have arisen to deal with this type of perturbations. In this paper, we provide a critical evaluation of the statistically robust risk training scheme, proposing theoretical reasons for its good generalization performance under the assumption of coordinate-wise independence of the perturbations, and giving novel insights, through computational experiments, on its generalisation power when such assumption does not hold.

1 Introduction

The early papers on adversarial examples ((Szegedy et al., 2014)), ((Goodfellow et al., 2015)) and the training methods which derive from these (for ex. fast gradient sign method), focus on **worst-case robustness** (that is, they try to identify the worst adversarial example in a neighborhood – generally an L^p ball – of each observation).

Adversarial Risk is defined as:

$$r^{adv}(f_\theta) = \mathbb{E}_{(X,Y)\sim p}\Big[max_{\delta\in\Delta}\, l(f_\theta(X+\delta), Y)\Big] \qquad (1)$$

where f_θ represents the network architecture, δ is some perturbation in the set Δ, p is the joint distribution of (X, Y), and l is a loss function.

An immediate consequence, is that such metric ignores the global performance of the classifier in the neighborhood of x. Often, we are concerned about **average robustness** with respect to perturbations that arise probabilistically (for ex. due to noise in measurements) rather than **worst-case robustness** to 'hand-picked' adversarial examples.

For example, consider a medical-imaging classification task where random noise is introduced due to different machines used in the various scans. Adversarial training aimed at minimizing (1) might give 'full-adversarial robustness' for 80% of the images, but give wrong predictions on the remaining 20%. On the other hand, a different network which is robust to 95% of the random noise, but for which most points have a small subset of adversarial examples, might be preferable as it gives higher accuracy overall, even if (1) is much higher in this case ((Wang et al., 2021)).

Moreover, Yin et al. ((2020)) proved a lower bound - using the theory of Rademacher complexities – for the generalization gap of adversarial training of order $O(\sqrt{d})$, where d is the dimension of the input. This means that performances of worse-adversarial metrics at test time tend to be poor on high-dimensional data ((Schmidt et al., 2018)).

Such considerations encouraged the development of alternative risk metrics which assessed the performance of a neural net over a probability distribution of the noise, and corresponding training methods which had better generalization properties than adversarial training.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

2 Background

Webb et al. ((2019)) proposed a notion of **pointwise robustness**. Given a point x, and a pertubation distribution $p(\cdot|x)$ around x, their metric measures the probability that a random point X' drawn from $p(\cdot|x)$ is classified differently than x by the neural network classifier f_θ.

$$\mathcal{I}_p(x) = \mathbb{E}_{X' \sim p(\cdot|x)} \Big[\mathbb{1}_{f_\theta(X') \neq f_\theta(x)} \Big] \tag{2}$$

Clear limitations of this approach are that it is localized, and that it does not take into account the true class y of observation x (for ex. a naïve classifier that maps every point in the input space to *class 1* would be pointwise robust at every point according to this metric, even though it is clearly a terrible classifier).

Motivated by these considerations Wang et al. ((2021)) introduced the **total statistical robustness metric**:

$$\mathcal{I}_p^{total} = \mathbb{E}_{(X,Y)} \Big[\mathbb{E}_{X' \sim p(\cdot|X)} \Big[\mathbb{1}_{f_\theta(X') \neq Y} \Big] \Big] \tag{3}$$

where X' , as above, is a perturbed example and Y is the true label of X.

For training purposes, the indicator function inside the inner expectation can be substituted by some differentiable loss function l (we will use categorical cross-entropy in our experiments), and the intractable outer expectation can be substituted by a Montecarlo estimator over the dataset $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ to get the **empirical statistical robust risk** (ESRR):

$$R_N^{stat} = \tfrac{1}{N} \textstyle\sum_{n=1}^{N} \mathbb{E}_{p(X'|x_n)} \Big[l(f_\theta(X'), y_n) \Big] \tag{4}$$

The inner expectation, although tractable (the theoretical work is done under the assumption than the distribution of the perturbations $p(\cdot|x)$ is known), is a computationally expensive integral which can be approximated by a Montecarlo estimator giving:

$$R_{N,C}^{statMC} = \tfrac{1}{N} \textstyle\sum_{n=1}^{N} \tfrac{1}{C} \sum_{m=1}^{C} l(f_\theta(x'_{n,m}), y_n) \tag{5}$$

where $x'_{n,m} \sim p(\cdot|x_n) \quad \forall\, m \in \{1, ..., C\}$.

This expression is a sum over the N observations of a convex differentiable function and can hence be used as an objective for SGD-based training methods for neural networks (**ESRR training**).

Moreover, building on the theoretical work of Mohri and Rostamizadeh ((2009)), it was shown that, for a neural network f_θ, under the assumption of boundedness of the loss function $l(x, y)$ on the input space and Lipschitz continuity of l with respect to the first entry, the generalization bound of statistical robust risk has an upper bound of order $O(log(w_{max}))$, where w_{max} is the maximum width of the network architecture.

Such result is independent of the dimensionality of the input and hence hints at better generalization properties of ESRR compared to worst-case adversarial metric (whose generalization gap is at least of order $O(\sqrt{d})$).

3 Experiments

Through two experiments we assess the performance of ESRR training when the test set is perturbed probabilistically.

Assume that observations in the test set d_{test} are perturbed according to an unknown distribution p_{test}:

$$\widetilde{x}_n = x_n + \epsilon_n \quad \text{where } \epsilon_n \overset{iid}{\sim} p_{test} \quad \forall n \leq |d_{test}|$$

It can be seen in (4) and (5) that the loss function for ESRR training requires us to specify a probability distribution p_{train} (which we would ideally like to be close to p_{test}); the perturbed training observations (used in the Montecarlo estimation of the statistical robust risk in (5)) will then be distributed according to:

$$x'_{n,m} = x_n + \epsilon_{n,m} \quad \text{where} \quad \epsilon_{n,m} \overset{iid}{\sim} p_{train} \quad \forall n \leq N, \, \forall m \leq C$$

Clearly, we expect the choice of p_{train} to have considerable effects on the performance of our classifier (for ex. if test images are only slightly perturbed, and we choose p_{train} with an extremely high variance, a lot of information in the training data will be lost, resulting in poor accuracy).

In the first experiment, images from the MNIST dataset are perturbed 'traditionally' using independent noise for each pixel; we show that ESRR generalizes well in these cases, provided a sensible choice of variance for p_{train}.

In the second experiment, we investigate a scenario in which noises of neighboring pixels are correlated, showing that classical ESRR performes quite poorly as training images have each pixel perturbed independently.

3.1 Experiment 1

The train set X_{train} has $n = 60,000$ observations of dimension $d = 784$ - each 28×28 image has been flattened – and the test set X_{test} has $n = 10,000$.

We perturb X_{test} according to the following algorithm:

Algorithm 1 Unifrom Independent Perturbation (UIP)

1: **Inputs**: Data matrix $X \in \mathbb{R}^{n \times d}$; $a \in [0, 5]$ ▷ we perturb the obs. using $Unif[-a, a]$
2: **for** each column $X_{\bullet j}$ $j \leq d$ **do**
3: Calculate the mean μ_j and standard deviation σ_j of $X_{\bullet j}$
4: Normalize the column: $\widetilde{X}_{\bullet j} \leftarrow (X_{\bullet j} - \mu_j)/\sigma_j$ ▷ good practice ((LeCun et al., 2012))
5: **for** each observation $i \leq n$ **do**
6: **for** each column $j \leq d$ **do**
7: $\widetilde{x}_{ij} \leftarrow \widetilde{x}_{ij} + \epsilon_{ij}$ where $\epsilon_{ij} \overset{iid}{\sim} Unif[-a, a]$
8: **Return** $\widetilde{X}$

If we the apply the inverse transformation of line 4, we get our perturbed dataset, where each pixel (x, y) is perturbed independently by an additive noise $\sigma_{x,y} \cdot Unif[-a, a]$ - where $\sigma_{x,y}$ is the variance over the test set of entries (i.e. pixel colours) in position (x, y) - (see Figure 1).

We now investigate how different choices of p_{train} perform when $p_{test} \sim Unif[-a, a]$ and a ranges from 0 to 5.

A first immediate observation from Figure 3 is that distributions p_{train} that are closer to the true perturbation p_{test} generally perform better (for ex. when $p_{test} \sim Unif[-2, 2]$, the best training scheme is $p_{train} \sim Unif[-2, 2]$ etc.). Moreover, we can see that even when the training takes place with extremely perturbed observations, the accuracy on test-sets with lower perturbations is significantly better. This indicates that the neural network classifier is not overfitting the noise of the train set, but it is learning the underlying true representations of the different classes.

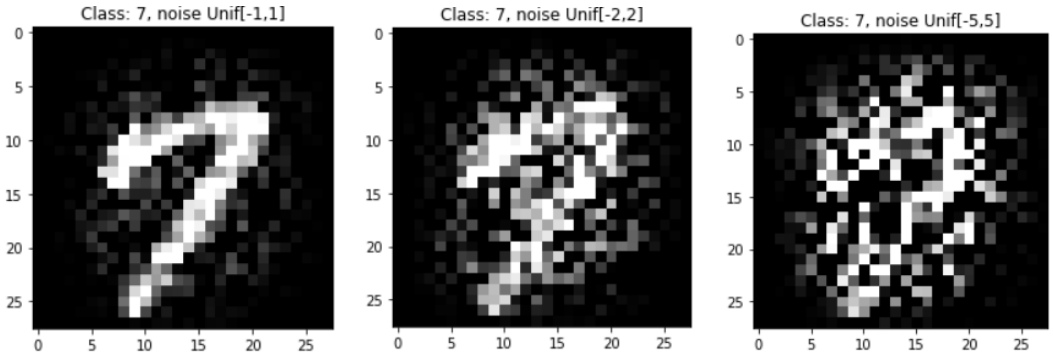

Figure 1: Uniform independent perturbations on a sample image, for increasing values of a

A second deeper consideration is that distributions p_{train} with greater variance tend to do better overall than lower variance distributions (for ex. the area under the accuracy curve for $p_{train} \sim Unif[-5, 5]$ is the highest, even though such choice is at the extreme of the true range of p_{test} distributions). A possible reason behind this is that train images with high-variance perturbations will still have a good proportion of pixels that are only modestly perturbed (for ex. if $p_{train} \sim Unif[-5, 5]$, then approximately one-fifth of the pixels will be within distance of one standard deviation from the true value of the pixel), therefore the model learns to deal with different degrees of noise. On the other hand, a model trained with $p_{train} \sim Unif[-1, 1]$ will not see any pixel perturbed more than one standard deviation, leaving it extremely susceptible to high variance perturbations.

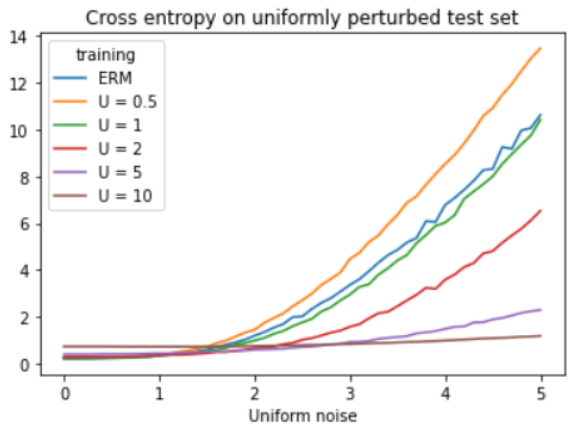

Figure 2: Cross entropy growth as UIP noise increases for different training choices

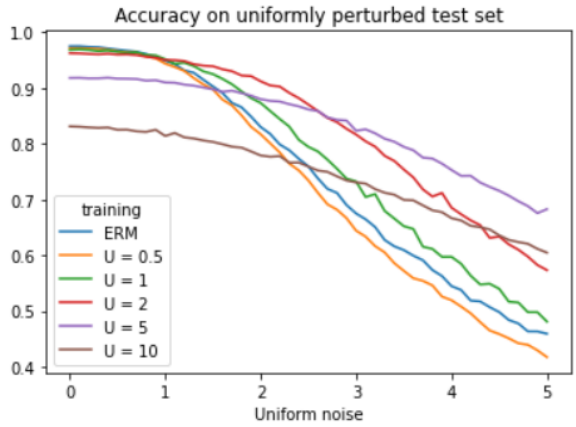

Figure 3: Accuracy for UIP noise in [0,5] for different training choices

Moreover, one might ask whether the type of distribution used in training plays a significant role (after all, we do not know that p_{test} is a uniform random variable, and picking p_{train} of the same type might have given us an advantage). To this purpose, we compared the performance of $p_{train} \sim N(0, 8)$ with that of $p_{train} \sim Unif[-5, 5]$ (which has variance 8.3), the accuracy curves for these two training methods were observed to be extremely similar. This provided evidence that, under the assumption of coordinate-wise independence of the noise, ESRR training is robust to unknown probabilistic perturbations independently of the type of distribution chosen at training time, provided that p_{train} is chosen with sufficiently high variance to 'cover' the real perturbations occurring on the test set.

3.2 Experiment 2

In this second experiment, we introduce correlation of noises for neighboring pixels. More precisely, we divide each 28×28 image in sixteen 7×7 squares, and for each square, pixels have noises normally distributed with correlation ρ^2:

Algorithm 2 Correlated Patches Perturbation (CPP)

Inputs: Dataset of n $d \times d$ images $X \in \mathbb{R}^{n \times d \times d}$; $\rho \in [0, 1]$; $\sigma > 0$
for each entry $X_{\bullet jk}$ $j \leq d$, $k \leq d$ **do**
    Calculate the mean μ_{jk} and standard deviation σ_{jk} of $X_{\bullet jk}$
    Normalize: $\widetilde{X}_{\bullet jk} \leftarrow (X_{\bullet jk} - \mu_{jk})/\sigma_{jk}$ ▷ good practice ((LeCun et al., 2012))
for each observation $i \leq n$ **do**
    for each 7×7 square patch S **do**
        Sample $\epsilon_S \sim N(0, \sigma)$
        for each pixel (j, k) in square S **do**
            $\widetilde{x}_{ijk} \leftarrow \widetilde{x}_{ij} + \rho \cdot \epsilon_S + \sqrt{1 - \rho^2} \cdot \epsilon_{ijk}$ where $\epsilon_{ijk} \overset{iid}{\sim} N(0, \sigma)$
Return $\widetilde{X}$

Figure 4: Correlated patches perturbation. In the first case $\rho = 0.5$, $\sigma = 0.5$, noticeable correlation between neighboring pixels. In the second case $\rho = 0.9$, $\sigma = 1$, extreme correlation, 7x7 patches clearly identifiable.

It is important to notice that $SD(Unif[-1, 1]) \approx 0.58 \approx SD(N(\mu = 0, \sigma = 0.5))$, and that $SD(Unif[-2, 2]) \approx 1.15 \approx SD(N(\mu = 0, \sigma = 1))$. Therefore, from Experiment 1 (Figure 3), we expect that if the noises were not correlated, we would have an accuracy above 95% in the first case and around 90% in the second. . In Table 1 we can see that, even though UIP and CPP have approximately the same pixel-wise distributions (i.e. marginals), because correlations are not present in ESRR training, the latter perturbation causes an extremely significant drop in accuracy for such classifiers.

training / test	ERM	Unif[-1,1]	Unif[-2,2]	Unif[-5,5]
$\rho = 0.5, \sigma = 0.5$	3.46/ 0.55 (0.97)	3.01/ 0.62 (0.97)	1.99/ 0.69 (0.96)	1.07/ 0.72 (0.92)
$\rho = 0.9, \sigma = 1$	9.21/ 0.29 (0.83)	8.47/ 0.34 (0.89)	6.93/ 0.38 (0.93)	3.37/ 0.42 (0.89)

Table 1: cross-entropy loss / accuracy / (accuracy when perturbations have same marginals but are independent)

Experiments were coded from scratch using TensorFlow. Full code is available at the following link.

4 Conclusions

Driven by an interest in robustness to probabilistic perturbations, we revised some of the main frameworks in the field. We showed that statistically robust training generalises well for perturbations that are coordinate-wise independent; it requires however a choice of variance sufficiently big to 'cover' the perturbations at test time. Moreover, we showed that such training scheme is not robust when correlation between perturbations is introduced, exposing a weakness of the framework which, to our knowledge, has not been covered by the deep learning literature.

References

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples, 2015.

Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In G. Montavon, G. B. Orr, and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade (2nd ed.)*, volume 7700 of *Lecture Notes in Computer Science*, pages 9–48. Springer, 2012. ISBN 978-3-642-35288-1.

M. Mohri and A. Rostamizadeh. Rademacher complexity bounds for non-i.i.d. processes. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2009.

L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Mądry. Adversarially robust generalization requires more data, 2018.

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

B. Wang, S. Webb, and T. Rainforth. Statistically robust neural network classification, 2021.

S. Webb, T. Rainforth, Y. W. Teh, and M. P. Kumar. A statistical approach to assessing neural network robustness, 2019.

D. Yin, K. Ramchandran, and P. Bartlett. Rademacher complexity for adversarially robust generalization, 2020.