ON THE ROBUSTNESS OF FEATURES LEARNED BY DEEP NEURAL NETWORKS

Jinqiao Hu
Department of Computer Science
Peking University
2000013141@stu.pku.edu.cn

Qingyue Wu
Department of Computer Science
Peking University
2000013057@stu.pku.edu.cn

Guo Tang
Department of Computer Science
Peking University
2000012962@stu.pku.edu.cn

Sirui Xie
Department of Mathematics
Peking University
2000010860@stu.pku.edu.cn

June 20, 2022

ABSTRACT

While traditional research on adversarial robustness of deep neural networks mainly focus on methods for training and attacking, we take a different angle of view: we study the robustness of features, and their contribution to the robust accuracy of classifiers. We propose a new standard, "sensitivity of features", to evaluate how robust a feature is. We give some theoretical evidence to show the significance of this standard. We then did several experiments on different models, using our standard to evaluate the robustness of features. The results align well with the robust accuracy of models.

1 Introduction

1.1 Background

The adversarial robustness of deep neural networks has attracted significant attention nowadays. Many reports has focused on training methods to improve the robustness. Madry et al. [2017] showed that PGD attack is the "strongest" among all first-order adversaries, thus the robust models trained with PGD adversary can be highly resistent to all first-order attacks.

There have also been studies on the theoretical understanding of adversarial examples. While many blame the curse of dimension for the adversary, Ilyas et al. [2019] proposed the idea of robust features, and concluded that adversarial examples can be attributed to non-robust features. In their work, the accuracy of classifier training on dataset with robust features can be up to about 50% under ℓ_2 attack, compared to less than 5% on original dataset, without the use of adversarial training. They argue that the natural pictures contains many "non-robust features" that are sensitive to deep neural networks, but not to human eyes. We found this view intriguing, and decided to study the robustness of features.

1.2 Target

In this work, we look deeper into the relation between the sensitivity of features and the adversarial robustness of deep neural networks.

We first give a standard to evaluate how robust a feature is. And we reproduce some of the experiments by Ilyas et al. [2019] to see what will happen.

[Ilyas et al., 2019] only did their experiments under ℓ_2 perturbations. Therefore to highlight the differences between ℓ_2 and ℓ_∞, we use ℓ_∞ perturbation instead of ℓ_2 for our experiments, and we also try to picture the distribution of the robustness of features.

2 Review of Robust Features

In their paper, Ilyas et al. [2019] divide features into robust features and non-robust features. Formally, a feature $f : \mathcal{X} \to \mathbb{R}$ is a ρ-useful feature, if

$$\mathop{\mathbb{E}}_{(x,y)\sim Q}[y \cdot f(x)] \geqslant \rho$$

And f is a γ-robustly useful feature, if

$$\mathop{\mathbb{E}}_{(x,y)\sim Q}\left[\inf_{\delta\in\Delta(x)} y \cdot f(x+\delta)\right] \geqslant \gamma$$

And we call a feature "non-robust", if it is useful, but not robustly useful.

While this definition captures the idea of "robustness", Ilyas et al. [2019] did not directly assess the "robustness" of the models they trained. We wonder how the robustness of features is distributed in non-robust models and robust models. For example, for a non-robust classifier, can it learn any robust features? Or all it learns are non-robust features?

To study this problem, we would first need to quantify the "robustness" of features. Inspired by Lipschitz networks[Zhang et al., 2021], we can define the robustness of a feature as the maximum change of value under certain perturbation. If this change can be bounded, then we might be able to certify the robust generalization ability of the classifier.

3 Sensitivity

We need to measure how "robust" a feature is. That is, how much its value changes when we perform perturbations on the input data.

3.1 Definitions

Let $x \in \mathcal{X}$ be the input graph where $\mathcal{X} = \mathbb{R}^d$. Let $\delta = (x, y)$ be a training data where $x \in \mathcal{X}$ is the input and $y \in \{+1, -1\}$ is the correct output. $S = \{\delta_1, \delta_2, \ldots, \delta_n\} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ is the training data set which is randomly selected from $\mathcal{X}$ with a probability distribution Q.

Let $f_1, f_2, \ldots, f_m$ be the features calculated by the neural network. Let $\boldsymbol{f}(x) := (f_1(x), f_2(x), \ldots, f_m(x))$. The classifier can be written as $F(x) = \text{sgn}(\boldsymbol{\alpha}\boldsymbol{f}(x) + b)$. Without loss of generality, we may assume that $\boldsymbol{\alpha}$ is a unit vector in the ℓ_∞ measure, since we can control the "scale" of each feature by multiplying it by some constant.

Usually, when we say a classifier F is robust, we mean that F doesn't change too much when we perform small perturbations on the input data.

Definition 1. *We say a classifier $F : \mathcal{N} \to \{\pm 1\}$ preserves δ-ε robustness, if*

$$\Pr_{(x,y)\sim Q}\left[\sup_{\|\Delta x\|_\infty \leqslant \delta} I[F(x) \neq F(x + \Delta x)]\right] \leqslant \varepsilon$$

Similarly, we can define the robustness of the features.

Definition 2. *For a feature $f : \mathcal{N} \to \mathbb{R}$, we say it preserves δ-ε robustness, if we have*

$$\mathop{\mathbb{E}}_{(x,y)\sim Q}\left[\sup_{\|\Delta x\|_\infty \leqslant \delta} |f(x) - f(x + \Delta x)|\right] \leqslant \varepsilon$$

And we define the sensitivity of f under δ perturbation as the minimum ε satisfying such constraints.

We will show that, under some good constraints, the sensitivity of features can somehow guarantee the robust accuracy of classifers.

3.2 The Significance of Our Definition

Theorem 3. *Suppose the classifier F has $1 - \gamma$ test accuracy, and it is also a large margin classifier, which means*

$$\Pr_{(x,y)\sim Q}[y(\boldsymbol{\alpha}\boldsymbol{f}(x) + b) \geqslant 1] \geqslant 1 - \gamma$$

If each feature f preserves δ-ε robustness, then $F(x) = sgn(\boldsymbol{\alpha}\boldsymbol{f}(x) + b)$ preserves δ - $(\gamma + m\varepsilon)$ robustness.

Proof. From the definition of feature robustness, we have

$$\Pr_{(x,y)\sim Q}\left[\sup_{\|\Delta x\|_\infty \leqslant \delta} |f(x) - f(x+\Delta(x))| \geqslant \frac{1}{m}\right] \leqslant m\varepsilon$$

So with probability at least $1 - m\varepsilon$, we have

$$\begin{aligned}
&|\boldsymbol{\alpha}\boldsymbol{f}(x) + b - \boldsymbol{\alpha}\boldsymbol{f}(x+\Delta x) - b| \\
=&|\boldsymbol{\alpha}(\boldsymbol{f}(x) - \boldsymbol{f}(x+\Delta x))| \\
\leqslant&\frac{1}{m}\|\boldsymbol{\alpha}\|_1 \\
\leqslant&1
\end{aligned}$$

So

$$\begin{aligned}
&\Pr_{(x,y)\sim Q} \sup_{\|\Delta_x\|_\infty \leqslant \delta} [\text{sgn}(\boldsymbol{\alpha}\boldsymbol{f}(x) + b) \neq \text{sgn}(\boldsymbol{\alpha}\boldsymbol{f}(x+\Delta x) + b)] \\
\leqslant&1 - \Pr_{(x,y)\sim Q}[|\boldsymbol{\alpha}\boldsymbol{f}(x) + b| < 1] - \Pr_{(x,y)\sim Q}[|\boldsymbol{\alpha}\boldsymbol{f}(x+\Delta x) + b| - \boldsymbol{\alpha}\boldsymbol{f}(x) - b| \geqslant 1] \\
\leqslant&1 - \gamma - m\varepsilon
\end{aligned}$$

□

We can see that for ε far less than $\frac{1}{m}$, we can attain good adversarial accuracy. Thus our definition of sensitivity captures the robustness of model to some extent.

4 Experiments

4.1 Datasets

We used the CIFAR-10 dataset for the experiments. For the "Robust Dataset", we use the one provided by Ilyas et al. [2019]. The images are all standardized so that for each of the RGB channels, the mean value is 0 while the variance is 1. That is, we scale the [0,255] pixel values to about [-2.5,2.5] linearly. During all training procedures, we perform data augmentation by random horizontal flips.

4.2 Model

For the model, we choose ResNet18[He et al., 2016], since it's easy to train, and it attains good test accuracy. Since the original ResNet18 model takes 224×224 sized images as inputs while CIFAR-10 images are of size 32×32, we substitute the first two layers with a 3×3 convolution layer with stride 1. The rest of the network is the same as ResNet18, with 4 groups of residual layers and 2 residual blocks each. We use momentum SGD as the optimizer, and set learning rate to 0.01 with batch size 256. We partly refer to `https://github.com/kuangliu/pytorch-cifar` for the implementation of ResNet18.

4.3 Adversarial Training

To obtain robust classifiers, we employ the method proposed by Madry et al. [2017]. More specifically, we train against projected gradient descend(PGD), starting at a uniformly random point in the ℓ_∞ ball around the training data. Taking into consideration the time that PGD takes and the scale of our model, we use 7 steps of PGD with a step size of $5\varepsilon/14$, where $\epsilon = 0.08$(i.e. approximately 4/255 pixel value for the original image).

4.4 Evaluating Sensitivity of Features

For a deep neural network such as ResNet18, we may view the outputs of the penultimate layer as the features for the input image, and the full connection layer at the end as a linear classifier. In the case of ResNet18, there are a total of 512 features, so we need to calculate the sensitivity for each of them.

Since the loss value of the local minima/maxima is well concentrated, and the loss achieved by the adversary increases in a fairly consistent way[Madry et al., 2017], we may assume these properties also hold for features. Again we apply

PGD over the ℓ_∞ ball around the input image to minimize/maximize the value for each activation. Each run starts at a uniformly random point in the ℓ_∞ ball around the data, and then takes 20 steps of PGD to find the approximate maxima and minima for each feature. We then normalize the sensitivity of features with the weight of the linear layer. Since evaluating all the 512 features is extremely time consuming, we only randomly sample 2 batches(512 images) from the test set to do the evaluation. We conduct four sets of experiments with $\varepsilon = 0.04, 0.06, 0.08, 0.2$ respectively.

5 Results and Analysis

We use five different ε for the three models, where $\varepsilon = 0$ corresponds to the standard test without adversary. "normal" corresponds to the model that is normally trained on normal dataset, "adversarial" is the model adversarially trained on normal dataset, while the "robust dataset" is normally trained on robust dataset[Madry et al., 2017]. We assess the adversarial accuracy as well as sensitivity of features over the same test dataset for all the models.

5.1 Accuracy

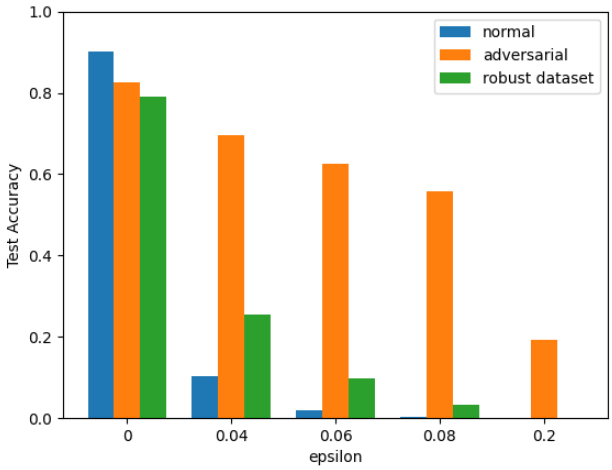

Figure 1: The accuracy map for different models with different ϵ.(20 PGD steps)

Here we do the perturbation on the standardized image, where the $[0, 255]$ pixel value is linearly scaled to approximately $[-2.5, 2.5]$ range, thus $\varepsilon = 0.04$ corresponds to 2/255 pixel value change for the original image, etc. We apply ℓ_∞ PGD with 20 steps, which is a common practice for the evaluation of robustness of models.

When we set ϵ to 0, all three models have a relatively high accuracy, while "normal" performs the best. If we gradually increase the ϵ, the accuracy of all three models will decrease as expected. In fact, the accuracy of "normal" and "robust dataset" will decrease so sharply that when ϵ is increased to 0.2, their accuracy goes down to exactly zero. This means those normally trained models are easily misled by adversarial perturbations. Relatively, the adversarial model still remains a high accuracy that even the ϵ is as high as 0.08, and still gives a much better answer than random guess when ϵ is 0.2.

5.2 Sensitivity

We draw the histogram of the sensitivity of features for different models, with different values of ε.

We can see that for each model, the sensitivity of features approximately follows normal distribution under same level of perturbation. The sensitivity of all three models gradually increases as ε grows. Besides, for each ε, the feature in the "adversarial" model has a much smaller sensitivity compared to "normal" and the "robust", while the sensitivity of "normal" is also notably higher than the "robust" when ϵ is 0.04, 0.06, 0.08, and becomes the opposite for $\varepsilon = 0.2$.

5.3 Analysis

Combining above results, we may conclude that the accuracy of classification is highly related to the distribution of sensitivity of features. The smaller the sensitivity that features have, the higher accuracy rate the model gets. While adversarial training results in much higher adversarial robustness, it also obtains features that are far less sensitive for all

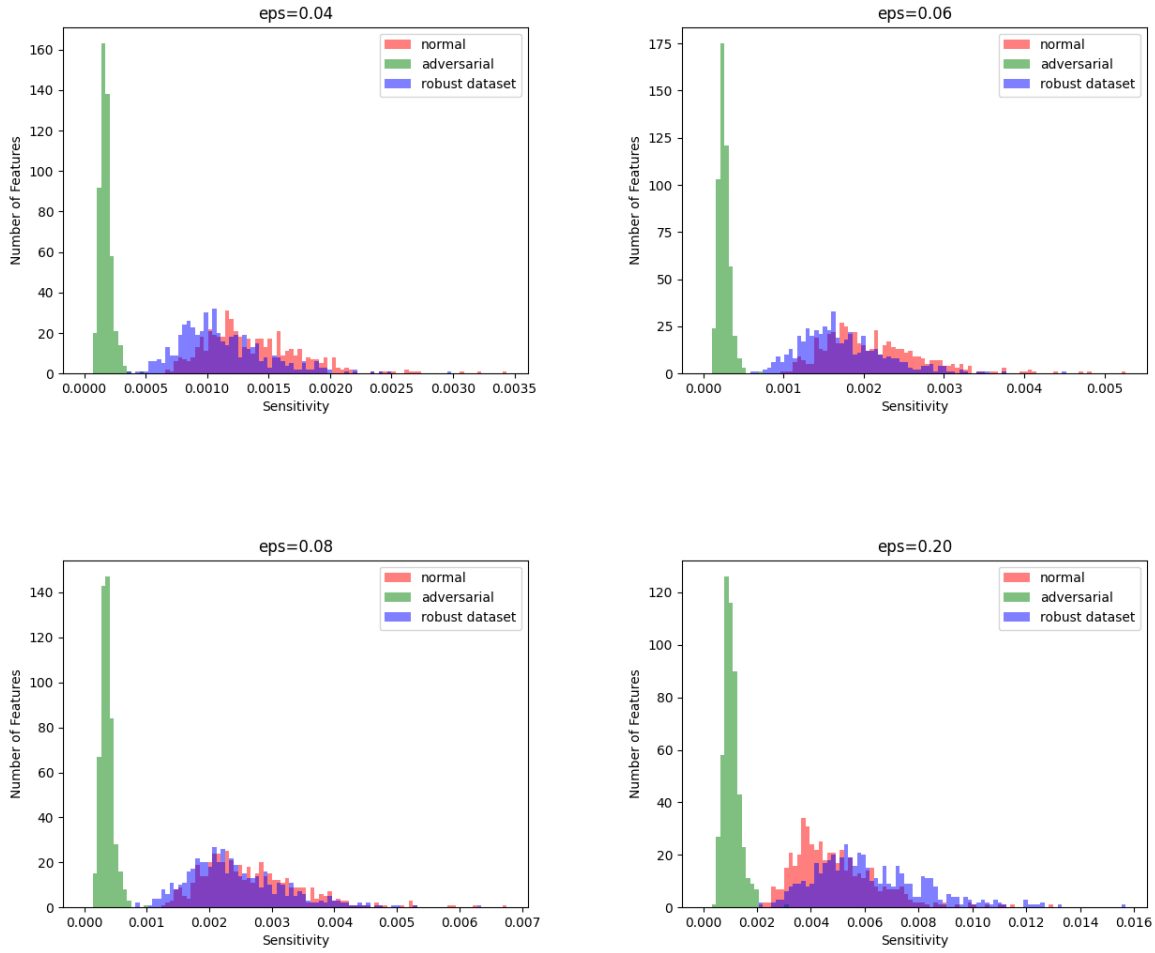

Figure 2: The sensitivity distribution of the features of different models under ℓ_∞ perturbation, with $\epsilon = 0.04, 0.06, 0.08, 0.20$ respectively

values of ε, which can also be viewed as "Robust Features"[Ilyas et al., 2019]. Also, the models normally trained on the robust dataset does learn features that are some what less sensitive than the normally trained model, which corroborates the results of Ilyas et al. [2019].

We also notice that the "normal" model trained on normal datasets does not learn the same number of robust and non robust features. In fact, almost all the learned features have fairly high sensitivity, thus we cannot improve the adversarial robustness of the model by simply rearranging the linear classifier at the last layer of the network.

There are also some results beyond expectation. First, Ilyas et al. [2019] shows that the model trained on "robust dataset" can achieve good adversarial accuracy comparable to the "adversarial" model, even when the adversarial accuracy of the "normal" model approaches 0. But our results show that its robustness is far less than the adversarial model, only a little better than the "normal" model. We speculate that this is partly due to the inherent difference between ℓ_2 adversaries used by Ilyas et al. [2019] and ℓ_∞ adversaries used by us, and it substantiate the claim that the adversarial robustness under ℓ_∞ is "harder" than ℓ_2.

Second, we notice that for $\varepsilon = 0.2$(10/255 perturbation for each pixel), the sensitivity of features of "robust dataset" is higher than the "normal" model in average , which is a counter-intuitive result. Our hypothesis is that, the model used to generate the "Robust Dataset" was not robust enough, and its features would only preserve robustness in a small radius. Once we go beyond that boundary, the value of the features can change sharply, even faster than the features learned by the "normal" model. This view is supported by the fact that, under $\varepsilon = 0.2$ adversaries, both the "normal" and "robust dataset" classify all inputs wrongly.

6 Conclusion

To evaluate the robustness of features learned by the neural network, we define the "sensitivity" of features, and perform an experiment to determine the actual distribution of sensitivity of features. We found that the sensitivity is actually distributed in a Gaussian style, and the adversarially trained model is much less sensitive than the other two models.

We also found that, unlike ℓ_2, in the ℓ_∞ scenario the adversarial accuracy of the model trained on the "robust dataset" is similar to the normally trained one, while the sensitivity of their features also have similar distribution. This means that to some extent, adversarial examples are still "bugs" rather than "features".

References

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.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *Advances in neural information processing systems*, 32, 2019.

Bohang Zhang, Tianle Cai, Zhou Lu, Di He, and Liwei Wang. Towards certifying l-infinity robustness using neural networks with l-inf-dist neurons. In *International Conference on Machine Learning*, pages 12368–12379. PMLR, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.