Introduction

Adversarial attack involves intentionally applying small manipulations to given input image leading to high confidence for the wrong classes [1]. These perturbations to images, called adversarial attacks is visually imperceptible to humans but, embarrassingly, they can fool even state-of-the-art classification models [2]. Classifier gets fooled by the adversarial examples because they lie on the lower likelihood region of the training distribution – which is observed regardless of the type of attack [3]. The paper describes the creative application of a density estimation model to clean up adversarial examples before applying an image model (for classification, in this setup). The basic idea is that the image is first moved back to the probable region of images before applying the classifier. For images, the successful PiexlCNN [4] model is used as a density estimator and is applied to clean up the image before the classification is attempted. Once the cleaned up image is obtained, it is transferred through the unmodified classifier which is now more resilient against impurities added during adversarial attack. The authors experiment on various attacking methods using CIFAR-10 and Fashion MNIST, yielding state-of-the-art results which have been also reproduced by our team, however only on CIFAR-10 dataset.

Proposed Approach

Base claim: A successful adversarial attack works by shifting a given image to the lower probability regions of the training distribution. Authors uses PixelCNN as density estimator and show that adversarial example lies in lower probable regions of the density function.
Distribution of adversarial examples: To verify that "unclean" images belong to a different distribution than training data, authors use distribution of p-values under the PixelCNN generative model. Images with adversarial impurities have non-uniform distribution of p-values whereas "clean" images have uniform distribution, which has been reproduced by our team (discussed in later sections).
Countering adversarial examples: By using the PixelCNN, authors develop a PixelDefend approach with a motive to project the adversarial examples back onto the high-density regions on the training distribution thus defending against the attacks. This approach is agnostic to attack or classification methods being deployed.
Purifying method: To defend against adversarial attacks, authors manipulate every pixel of the image to *argmax* of 256-way softmax obtained from the PixelCNN such that it lies within ϵ distance of the given pixel's value. After applying this procedure, output image is in higher probability regions under the training distribution; henceforth, lowering the risk of misclassification.

Reproducing the Research

We have verified overall correctness and reproduciblity of the paper. Paper presents a very novel idea to use a generative modeling (PixelCNN in this case) to "clean" adversarial perturbations of the image. We carried our reproduciblity effort by verifying three logical claims made within the paper as follows.

1. **Claim 1:** Set of adversarial perturbed images lie in lower likely-hood regions of density estimation of training data.

2. **Claim 2:** Prevention of adversarial attacks is done by moving the images to higher likely-hood regions through pixel-by-pixel manipulation using PixelCNN.

3. **Claim 3:** Empirically verifying efficacy of the proposed "cleaning" method by measuring change in prediction accuracy using classifier.

Claim 1

First claim presented in the paper states that PixelCNN can be used to determine whether set of given images belong to distribution of adversarially perturbed images or clean images. In order to do that, authors calculate the histogram of p-values obtained from PixelCNN using Equation 1. The deviation of the histogram from uniform represents the adversarial perturbation present in the given population of images. If all the images are "clean" then histogram will be very close to uniform distribution.

$$p = \frac{1}{N+1}\left(\sum_{i=1}^{N} I[T_i \leq T] + 1\right) = \frac{T+1}{N+1} = \frac{1}{N+1}\left(\sum_{i=1}^{N} I[p_{CNN}(X_i) \leq p_{CNN}(X')] + 1\right) \tag{1}$$

Where, I is indicator function whose value is 1 if condition inside is true, N is number of training example, $P_{CNN}(X)$ is output of the PixelCNN and X' is image from a given population.
We successful verified this claim. We took following steps to do so.

1. Obtained PixelCNN trained on CIFAR-10.
2. Used **foolbox** library to perform various adversarial attacks on test dataset of CIFAR-10.
3. Calculate the p-value distributions of clean set of images and set of images obtained from each of adversarial attacks.
4. **Extra efforts:** We performed few adversarial attacks that were not even specified in the paper such as, FGM, Gaussian Blur, Saliency Map and Salt & Paper Noise, LBFGS and Contrast attach.

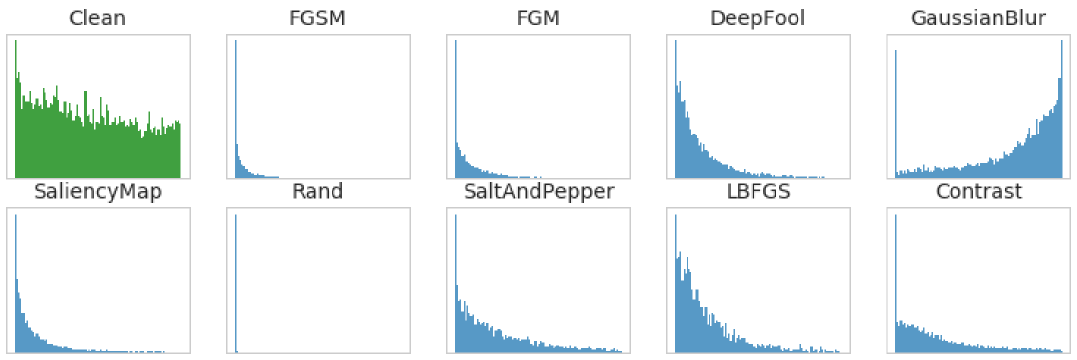

Figure 1: The distribution of p-values when using the PixelCNN model which is in agreement with Figure 3 of the original paper

Claim 2

Second claim of the paper suggests `PixelDefend` approach that moves the given input image to the higher likely-hood regions of the training distribution. In order to do so, authors uses the predictions from the PixelCNN and iteratively manipulates the given image pixel-by-pixel in Algo 1. Each pixel is changed to the most probable value predicted from PixelCNN such that it lies within the ϵ-ball distance from the actual pixel's value. In the paper, ϵ is chosen to be 16, however we chose it as 8.

```
Algorithm 1 PixelDefend
Input: Image $\mathbf{X}$, Defense parameter $\epsilon_{\text{defend}}$, Pre-trained PixelCNN model $p_{\text{CNN}}$
Output: Purified Image $\mathbf{X}^*$
 1: $\mathbf{X}^* \leftarrow \mathbf{X}$
 2: for each row $i$ do
 3:     for each column $j$ do
 4:         for each channel $k$ do
 5:             $x \leftarrow \mathbf{X}[i, j, k]$
 6:             Set feasible range $R \leftarrow [\max(x - \epsilon_{\text{defend}}, 0), \min(x + \epsilon_{\text{defend}}, 255)]$
 7:             Compute the 256-way softmax $p_{\text{CNN}}(\mathbf{X}^*)$.
 8:             Update $\mathbf{X}^*[i, j, k] \leftarrow \arg\max_{z \in R} p_{\text{CNN}}[i, j, k, z]$
 9:         end for
10:     end for
11: end for
```

Figure 2: PixelDefend Algorithm taken from the original paper

Claim 3

Third claim that we try to reproduce is the predictions accuracies of classifiers (ResNet and VGG19) after performing PixelDefend purification. Due to limitation of computation resources, we reproduced results using ResNet only and using only the subset of test datase from CIFAR-10. Even though original paper has claimed to achieve 3 second/per image speed on Titan XP, however, fastest we reached is 2 mins/per image using 8 K80 GPUs, thus making it unfeasible to run experimentation's on entire test dataset with 10*10000 or (100k) adversarial images which would result in 200,000 minutes or 130 days of user-time for computation.

Reproduction Details

In this project, we applied similar settings to those in that paper. For PixelCNN model, we used the pre-trained model on CIFAR-10 by OpenAI. Considering the extreme time consuming the training the PixelCNN model, we failed to train our model on Fashion-MNIST dataset.

In the improved version PixelCNN++, the authors use the discretized logistic mixture likelihood as the output instead of the original 256-way softmax to reduce the computational expense. However, in the project, we need the loss over 256 pixel values for further argmax calculation. Therefore, with hints and confirmations from the authors, we succeeded to bin the mixture likelihood into 256 intervals and perform the defend phase.

One serious bug and challenge we met is the argmax within defend range. The defend strategy requires a mask over the 256 dimension outputs such that argmax will not be performed in the entire pixel value range(0, 255), but the range within (x-ϵ, x+ϵ). These mask actually prevents the predicted pixel from differing dramatically compared to the input pixel value. During implementing, we add huge negative values on the mask where value is out of range, but stuck in getting feasible purified images. The parallel GPU code and channel dependency in PixelCNN generation made it hard so debugging and it turned out it is the unsigned int data type that caused overflow on uint8 range and shrinked the output to the range around 255.

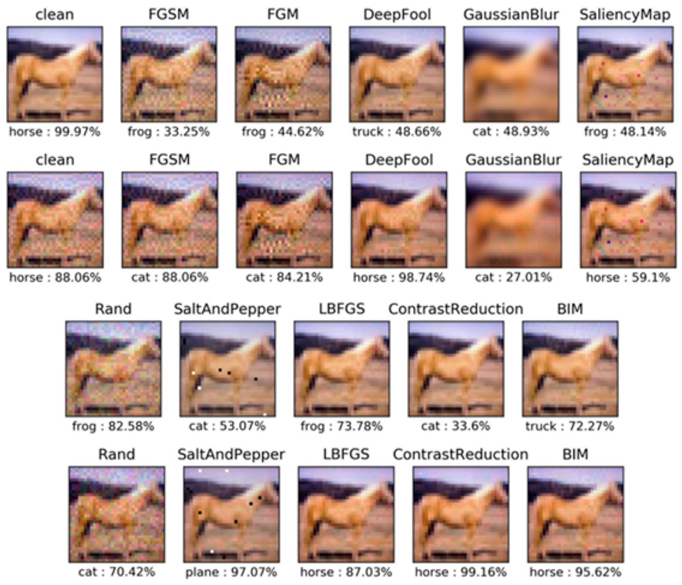

Figure 3: Image shows the prediction results obtained from ResNet after cleaning the perturbed image using PixelDefend algorithm; Each image shows the prediction probablity, some methods are successfully cleaned however other are not and still predicted wrongly.

Conclusion

In summary, the PixelDefend is able to understand and defend against adversarial examples by projecting these examples back onto higher density regions which" "purifies" the images, thereby lowering misclassification. In terms of reproducibility, this paper is rather complicated since most of the work is very recent and it is difficult to find assisting resources or source-code on the internet. Hence, we directly communicated with the first author of the paper, Yang Song, who provided a snippet of the code which assisted to verify our understanding of the algorithm. Although the paper is reproducible, we mainly encountered computation limitations. Specifically, we compromised by reproducing the method by only validating on CIFAR-10 dataset and further utilizing subset of CIFAR-10 for purification method. We used only ResNet for prediction though paper uses both ResNet and VGG19. At its entirety, we are able to (i) successfully identify population of adversarial perturbed images, (ii) use PixelDefend algorithm to purify perturbed images and (iii) verify especially performance of PixelDefend on prediction accuracy of perturbed images on ResNet classifier.

References

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

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

[3] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," *arXiv preprint arXiv:1710.10766*, 2017.

[4] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," *arXiv preprint arXiv:1601.06759*, 2016.