Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples

Supplementary Material

1 Local Intrinsic Dimensionality

Defense Details. The Local Intrinsic Dimensionality (Amsaleg et al., 2015) "assesses the space-filling capability of the region surrounding a reference example, based on the distance distribution of the example to its neighbors" (Ma et al., 2018). The authors present evidence that the LID is significantly larger for adversarial examples generated by existing attacks than for normal images, and they construct a classifier that can distinguish these adversarial images from normal images. Again, the authors indicate that LID *is not intended as a defense* and only should be used to explore properties of adversarial examples. However, it would be natural to wonder whether it would be effective as a defense, so we study its robustness; our results confirm that it is not adequate as a defense. The method used to compute the LID relies on finding the k nearest neighbors, a non-differentiable operation, rendering gradient descent based methods ineffective.

Let $\mathcal{S}$ be a mini-batch of N clean examples. Let $r_i(x)$ denote the distance (under metric $d(x,y)$) between sample x and its i-th nearest neighbor in $\mathcal{S}$ (under metric d). Then LID can be approximated by

$$\text{LID}_d(x) = -\left(\frac{1}{k}\sum_{i=1}^{k} \log \frac{r_i(x)}{r_k(x)}\right)^{-1}$$

where k is a defense hyperparameter the controls the number of nearest neighbors to consider. The authors use the distance function

$$d_j(x,y) = \left\| f^{1..j}(x) - f^{1..j}(y) \right\|_2$$

to measure the distance between the jth activation layers. The authors compute a vector of LID values for each sample:

$$\overrightarrow{\text{LID}}(x) = \{\text{LID}_{d_j}(x)\}_{j=1}^{n}.$$

Finally, they compute the $\overrightarrow{\text{LID}}(x)$ over the training data and adversarial examples generated on the training data, and train a logistic regression classifier to detect adversarial examples. We are grateful to the authors for releasing their complete source code.

Discussion. While LID is not a defense itself, the authors assess the ability of LID to detect different types of attacks.

Through solving the formulation

$$\text{min. } |x - x'|_2^2 + \alpha\left(\ell(x') + \text{LID-loss}(x')\right)$$

the authors attempt to determine if the LID metric is a good metric for detecting adversarial examples. Here, LID-loss$(\cdot)$ is a function that can be minimized to reduce the LID score. However, the authors report that this modified attack still achieves 0% success. Because Carlini and Wagner's ℓ_2 attack is unbounded, any time the attack does not reach 100% success indicates that the attack became stuck in a local minima. When this happens, it is often possible to slightly modify the loss function and return to 100% attack success (Carlini & Wagner, 2017b).

In this case, we observe the reason that performing this type of adaptive attack fails is that gradient descent does not succeed in optimizing the LID loss, even though the LID computation is differentiable. Computing the LID term involves computing the k-nearest neighbors when computing $r_i(x)$. Minimizing the gradient of the distance to the current k-nearest neighbors is not representative of the true direction to travel in for the optimal set of k-nearest neighbors. As a consequence, we find that adversarial examples generated with gradient methods when penalizing for a high LID either (a) are not adversarial; or (b) are detected as adversarial, despite penalizing for the LID loss.

Evaluation. We now evaluate what would happen if a defense would directly apply LID to detect adversarial examples. Instead of performing gradient descent over a term that is difficult to differentiate through, we have found that generating high confidence adversarial examples (Carlini & Wagner, 2017a) (completely oblivious to to the detector) is sufficient to fool this detector. We obtain from the authors their detector trained on both the Carlini and Wagner's ℓ_2 attack and train our own on the Fast Gradient Sign attack, both of which were found to be effective at detecting adversarial examples generated by other methods. By generating high-confidence adversarial examples minimizing ℓ_∞ distortion, we are able to reduce model accuracy to 2% success within $\epsilon = 0.015$. LID reports these adversarial examples are benign at a 97% rate (unmodified test data is flagged as benign with a 98% rate).

This evaluation demonstrates that the LID metric can be circumvented, and future work should carefully evaluate if building a detector relying on LID is robust to adversarial examples explicitly targeting such a detector. This work also raises questions whether a large LID is a fundamental characteristic of all adversarial examples, or whether it is a by-product of certain attacks.

2 Defense-GAN

Defense Details. The defender first trains a Generative Adversarial Network with a generator $G(z)$ that maps samples from a latent space (typically $z \sim \mathcal{N}(0,1)$) to images that look like training data. Defense-GAN takes a trained classifier $f(\cdot)$, and to classify an input x, instead of returning $f(x)$, returns $f(\arg\min_z |G(z) - x|)$. To perform this projection to the manifold, the authors take many steps of gradient descent starting from different random initializations.

Defense-GAN was not shown to be effective on CIFAR-10. We therefore evaluate it on MNIST (where it was argued to be secure).

Discussion. In Samangouei et al. (2018), the authors construct a white-box attack by unrolling the gradient descent used during classification. Despite an unbounded ℓ_2 perturbation size, Carlini and Wagner's attack only reaches 30% misclassification rate on the most vulnerable model and under 5% on the strongest. This leads us to believe that unrolling gradient descent breaks gradients.

Evaluation. We find that adversarial examples *do* exist on the data manifold as described by the generator $G(\cdot)$. However, Defense-GAN *does not* completely project to the projection of the generator, and therefore often does not identify these adversarial examples actually on the manifold.

We therefore present two evaluations. In the first, we assume that Defense-GAN were to able to perfectly project to the data manifold, and give a construction for generating adversarial examples. In the second, we take the actual implementation of Defense-GAN as it is, and perform BPDA to generate adversarial examples with 50% success under reasonable ℓ_2 bounds.

Evaluation A. Performing the manifold projection is nontrivial as an inner optimization step when generating adversarial examples. To sidestep this difficulty, we show that adversarial examples exist *directly on* the projection of the generator. That is, we construct an adversarial example $x' = G(z^*)$ so that $|x - x'|$ is small and $c(x) \neq c(x')$.

To do this, we solve the re-parameterized formulation

$$\text{min.} \quad \|G(z) - x\|_2^2 + c \cdot \ell(G(z)).$$

We initialize $z = \arg\min_z |G(z) - x|$ (also found via gradient descent). We train a WGAN using the code the authors provide (Gulrajani et al., 2017), and a MNIST CNN to 99.3% accuracy.

We run for 50k iterations of gradient descent for generating each adversarial example; this takes under one minute per instance. The unsecured classifier requires a mean ℓ_2 distortion of 0.0019 (per-pixel normalized, 1.45 un-normalized) to fool. When we mount our attack, we require a mean distortion of 0.0027, an increase in distortion of 1.46×; see Figure 1 for examples of adversarial examples. The reason our attacks succeed with 100% success without suffering from vanishing or exploding gradients is that our gradient computation only needs to differentiate through the generator $G(\cdot)$ once.

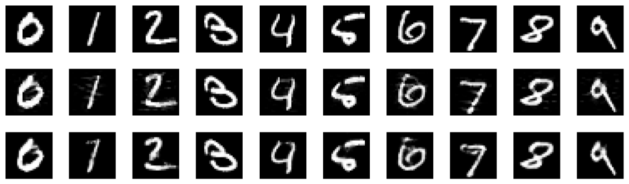
Figure 1: Images on the MNIST test set. Row 1: Clean images. Row 2: Adversarial examples on an unsecured classifier. Row 3: Adversarial examples on Defense-GAN.

Concurrent to our work, Ilyas et al. (2017) also develop a nearly identical approach to Defense-GAN; they also find it is vulnerable to the attack we outline above, but increase the robustness further with adversarial training. We do not evaluate this extended approach.

Evaluation B. The above attack *does not* succeed on Defense-GAN. While the adversarial examples *are* directly on the projection of the Generator, the projection process will actually move it *off* the projection.

To mount an attack on the approximate projection process, we use the BPDA attack regularized for ℓ_2 distortion. Our attack approach is identical to that of PixelDefend, except we replace the manifold projection with a PixelCNN with the manifold projection by gradient descent on the GAN. Under these settings, we succeed at reducing model accuracy to 55% with a maximum normalized distortion of .0051 for successful attacks.