Adversarial Defense Using Deep Image Priors for Deep Image Models

Shivang Singh*
Department of Computer Science
University of Texas at Austin
shivang.singh@utexas.edu

Daniel Almeraz*
Department of Computer Science
University of Texas at Austin
dalmeraz@utexas.edu

Zhou Fang*
Department of Mathematics
University of Texas at Austin
fazhou@utexas.edu

Abstract

Deep learning architectures are susceptible to adversarial attacks, which are examples where a small input perturbation is added to the input leading the model to incorrectly classify. One interesting development in adverserial defense for images is the use of Deep Image Prior(DIP) to reconstruct the original image, a Convolutional Neural Network used to enhance the an input image with no prior training data other than the image itself. We extend this work by replicating this network as well as using this architecture on image classification models, such as Visual Transformers. Implementation of our model can be seen at https://github.com/dalmeraz/Classification-DIP-Defense.

1. Problem Statement

1.1. Motivation

Due to the increasing use of neural networks in countless real life applications, adversarial attacks could pose a large challenge in the deployment of real time deep learning systems. Therefore, the exploration of adversarial defense is vital to increase the security of deep learning systems. In this work we will focus on deep learning models centered around image classification. We will be exploring the use of the Deep Image Prior, which is a CNN which allows for image enhancement. In this case, we will be using the Deep Image Prior to reconstruct the original image after it has been altered using an adversarial attack. Additionally, we will be analysing its effects on a new wave of vision models brought to us as Vision Transformers.

*equal contribution

1.2. High Level Overview

Our model makes crucial use of the DIP trace, which is generated when executing the DIP model. As seen in Figure 1, a DIP trace includes many sub-images for which different prediction labels can be created. Therefore, the task for our defense model is using this DIP trace to recreate a clean image that our model can predict correctly. We begin by ignoring the first several noisy sub-images in the trace. Then, we filter out cross-boundary images, which are consecutive pairs of image in the DIP trace that have different predicted labels. We then use each pair of images to create a list of on-boundary images. These images are obtained by linearly interpolating the two images in the pair so it has a prediction as close as possible to equal to both classes. We then create on-manifold images by perturbing the on-boundary images towards adversarial noise, and average them together to create our final x_{rec}.

For example, after truncating our DIP trace changes label predictions 20 times, there will be 20 cross-boundary images, 20 on-boundary images, 20-manifold images, and 1 final reconstructed image.

2. Related Work

Image reconstruction plays a vital role in adversarial defense for image based deep learning architectures. Furthermore, many works utilize Deep Image Priors to reconstruct the original image after it has been altered.

2.1. Deep Image Prior

Deep Image Prior is an iterative model developed for enhancing images with no prior training data other than the image itself. [11]. It can be used to fill in missing gaps within images, increase image resolutions and remove noise. It works by starting with a set of random values

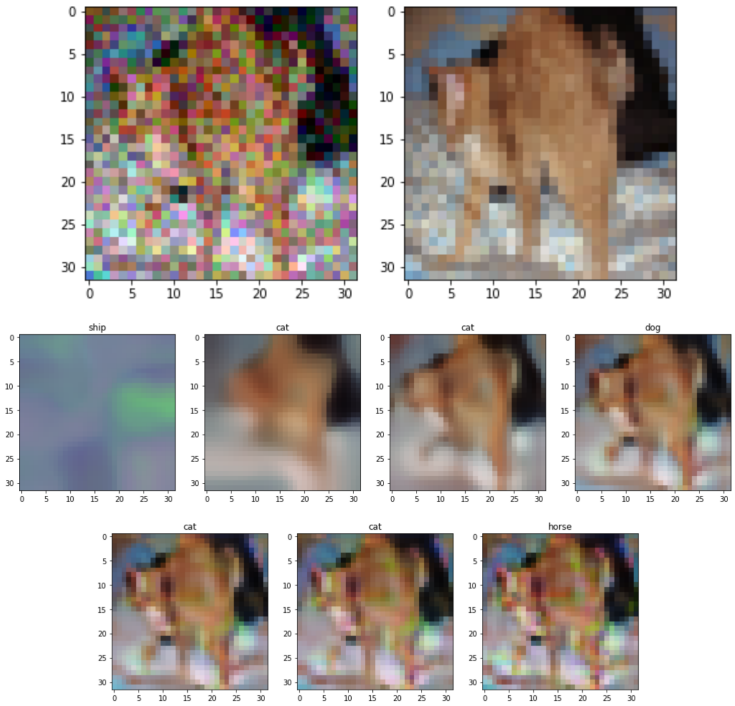

Figure 1. Preliminary results showing an adversarial image, the original image, and then a DIP trace including the predictions that were generated. The original label is 'cat' and adversarial prediction was 'horse'. The DIP reconstruction starts with a random initialization of weights. It then starts to construct images that are of the same label as the original image. However, by the end the DIP learns the small perturbations present in the adversarial attack. Thus our approach focuses on the intermediate DIP iterations.

for each pixel in the unenhanced image. It then repeatedly passes the image through a convolutional network and calculates loss based off the input image. This in result generalizes the input image.

2.2. DIP-Defense

DIP-Defense takes advantage of the denoising capabilities of DIP to recreate images that could contain noise [1]. The defense model feeds input images through a DIP network a number of times based off of a heuristic which in theory denoises adverserial images. However, this work assumes that the DIP network could not learn some of the perturbations that were made to the original image.

2.3. Delving into Deep Image Prior for Adversarial Defense

Delving into Deep Image Prior for Adversarial Defense is an iterative improvement over DIP-defense in which the heuristic used to know how many DIP recreations to create is based off of finding decision boundaries in the classification space of a model. The authors evaluate their work on Resnet-18 over a set of image classification tasks. [2]. We elaborate on this work by analyzing it for other models.

3. Technical approach

In order to generate adverserial attacks, our paper uses torchattacks [6]. This library allows us to specify what kind of attack we want to execute and given a victim model, generates adverserial attacks for it. Additionally, for our defense model, we make use of the official DIP repository and use their utils to generate out defense model [10]. Sample results of adverserial attacks can be see in Figure 2

Once we have a victim model and an adversarial attack we begin our implementation by creating a DIP trace. The DIP trace is generated by doing forward passes to the DIP model and at each iteration storing the DIP generated image and the resulting label when that image is passed to the victim model. Figure 1 gives an example of a generated DIP trace.

We assume all images from the dataset are distributed on a high dimensional manifold. The victim model divides the manifold into several areas so that all images in the same area have the same label. DIP trace is a list of images distributed on the manifold. We denote DIP trace as $\textbf{DIP} = \{I_k\}_{k=1}^{k=N}$, and $\textbf{label}(I_k)$ be the class to which the image I_k has the highest possibility to be classified.

First, we construct a sub-list of DIP trace called cross-

Figure 2. Adverserial Attack images generated by our model. The victim model used was a simple, small CNN architecture trained on CIFAR10 with adverserial images generated with projected gradient decent.

Model	No Attack	PGD	FGSM	BIM
VIT	0.93	0.09	0.52	0.49
VIT + Defense	0.50	0.51	0.53	0.41
VGG16	0.87	0.82	0.92	0.92
VGG16 + Defense	0.27	0.14	0.15	0.19

Table 1. This table contains the difference in accuracy on the CIFAR-10 dataset. Here we alternate between the victim model as well as defense and no defense. We observe that the ViT model is more prone to gradient based PGD attack versus the ResNet model, but is able to recover performance with the the defense.

boundary images in which each image in that sub-list has different label than the next image in the DIP trace. There are two more minor conditions that an image should meet to be included in cross-boundary images. In summary, an image I_k is included in the cross-boundary images if
1. $t_0 \leq k$ where t_0 is hyper-parameter
2. $SSIM(I_k, I_0) > \tau$ where I_0 is the original image, SSIM function computes the similarity between two images, and τ is hyper-parameter. We choose $\tau = 0.4$ in this paper.
3. **label**$(I_k) \neq$ **label**(I_{k+1})

Secondly, we linearly interpolating each pair of cross-boundary images to construct a list called on-boundary images. For I_k in cross-boundary images, choosing $i \in \{1, 2, ..., 100\}$ such that the difference between the probability of the constructed image $\frac{i}{100} * I_k + \frac{100-i}{100} * I_{k+1}$ to be classified as **label**(I_k) and as **label**(I_{k+1}) is the smallest among these 100 constructed images leads to the resuling cross-boundary image.

Next, by perturbing each on-boundary image a little bit, we can get manifold-images which would be classified correctly by the victim model, and very close to on-boundary images as well. In detail, for each on-boundary image I, we choose a hyper-parameter β that is very small, and construct the manifold images, $I_{manifold} = I + \beta * (I - I_0)$ where I_0 is the original image.

Finally, we average the generated manifold-images $\{I_i\}_{i=1}^m$ to get a reconstructed image by the following formula

$$I_{reconstruct} = \frac{1}{m}\Sigma_{i=1}^m I_i$$

We test the reconstructed image on our victim model to see if it is classified correctly. This is done for all the images on our dataset.

4. Optimizations

Due to the multiple models needing to keep track of intermediate DIP images one of the largest constraints encountered was memory. Initial implementations would consume >32 GB of memory and so optimizations were needed to be done. The most straight forward and impactful optimization that we did was instead of keeping track of the full DIP trace the model began to ignore DIP images were the predicted label would not transition, as these were images that were not needed for the rest of the model to function. From here, additional memory management such as using both GPU and CPU memories allowed us to achieve a model were model executions could still be done on GPU. There's still additional improvements to be done here such as generating the on-manifold images as the DIP-trace is created however this is left as future work.

5. Results

The full system has multiple configurable components and hyper-parameters that can be configured among each component. The three main components we can control are the DIP architecture used for our defense model, the attack model used to generate adversarial attacks and victim model. For the DIP model, we use a skip net. For the victim model, we use ViT architecture. As for dataset, our testing uses the CIFAR-10 dataset.

We analyzed the impact of our adversarial defense over the CIFAR-10 dataset. [7] We alternated between two victim models, a pretrained ViT (Visual Transformers) [3] and a VGG-16 model [9] . We also alternated between 3 different adversarial attack methodologies, PGD (Projected Gradient Descent) [12], Fast Gradient Signed Method (FGSM) [4], and Basic Iterative Method (BIM) [8] to see the affect of different adversarial attacks on the the different victim

models as well as with our adversarial defense. We present the results of this experiment in Table 1.

As can be seen, the most interesting results come when comparing ViT vs Vit + Defense in No attack and PGD. Here we see that although the ViT model has high accuracy it is extremely sensitive to Adverserial Attacks and is fooled by 91% of them. Meanwhile, our ViT-Defense model sees a heavy hit when not receiving adverial attacks by achieving 50% accuracy but when given adverserial images, it generates an extremely similar accuracy. This similarity can also be seen when looking across the whole row. Our interpretation of this is that our model is fairly good at removing adverserial noise however it might be introducing its own type of DIP-based noise.

When looking at our comparison results for our CNN-based model we see a fall in success of our model. The attacks are generally less successful and less likely to be successful through our model. This is correlated to each other as our model's beta hyper parameter pushes models to predict away from the original label, thus if the original label is right beta pushes our predictions slightly away from this.

In addition to the results presented, our model was also tested with an ImageNet pretrained ResNet50 [5] as a way of analysing if the success of our ViT model was due to the resizing done to fit the images or due to the pretrained nature of the models. What we found for these tests were extremely similar results to that of VGG16 and thus disprove the two concerns as being large factors in the results we achieve.

6. Conclusion

Our model's strengths show in the scenarios in which large amounts of adverserial attacks are expected to be encountered. Additionally, there are many hyper parameters that could probably still be tuned and explored, some which would directly affect the metrics that could be most crucial when concerned about precision on adverserial attacks or general accuracy.

References

[1] Tao Dai, Yan Feng, Dongxian Wu, Bin Chen, Jian Lu, Yong Jiang, and Shu-Tao Xia. Dipdefend: Deep image prior driven defense against adversarial examples. In *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20, page 1404–1412, New York, NY, USA, 2020. Association for Computing Machinery. 2

[2] Li Ding, Yongwei Wang, Xin Ding, Kaiwen Yuan, Ping Wang, Hua Huang, and Z. Jane Wang. Delving into deep image prior for adversarial defense: A novel reconstruction-based defense framework, 2021. 2

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 3

[4] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015. 3

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 4

[6] Hoki Kim. Torchattacks : A pytorch repository for adversarial attacks. *CoRR*, abs/2010.01950, 2020. 2

[7] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). 3

[8] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016. 3

[9] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 3

[10] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. *arXiv:1711.10925*, 2017. 2

[11] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. *International Journal of Computer Vision*, 128(7):1867–1888, Mar 2020. 1

[12] Tianhang Zheng, Changyou Chen, and Kui Ren. Distributionally adversarial attack. *CoRR*, abs/1808.05537, 2018. 3