MAKING DEEP NEURAL NETWORK FOOLING PRACTICAL

Ambar Pal *Chetan Arora*

Department of Computer Science and Engineering, IIIT Delhi

ABSTRACT

With the success of deep neural networks (DNNs), the robustness of such models under adversarial or fooling attacks has become extremely important. It has been shown that a simple perturbation of the image, invisible to a human observer, is sufficient to fool a deep network. Building on top of such work, methods have been proposed to generate adversarial samples which are robust to natural perturbations (camera noise, rotation, shift, scaling etc.). In this paper, we review multiple such fooling algorithms and show that the generated adversarial samples exhibit distributions largely different from the true distribution of the training samples, and thus are easily detectable by a simple meta classifier. We argue that for truly practical DNN fooling, not only should the adversarial samples be robust against various distortions, but must also follow the training set distribution and be undetectable from such meta classifiers. Finally we propose a new adversarial sample generation technique that outperforms commonly known methods when evaluated simultaneously on robustness and detectability.

Index Terms— Deep Network Fooling, Robustness of Adversarial Attacks

1. INTRODUCTION

Success of Deep Neural Networks (DNNs) on the ImageNet dataset [1, 2] heralded a new era in machine learning and computer vision. Easy availability of large amounts of data as well as computational resources have led DNNs to produce remarkable success in many problems across computer vision. While DNNs are highly expressive with millions of parameters, they also end up learning a lot of unintended features. Given the central role of DNNs in many safety critical applications, there is a lot of interest in looking into ways to generate "adversarial samples", i.e. inputs which can fool a network by forcing undesired false positives.

The first type of adversarial techniques are those that perform perturbations on the input image, typically looking at the backpropagated gradient, or aiming to minimize a given adversarial objective. The *fast gradient sign method* ([3]), pushes the image away from the source class by computing the gradient of the loss with respect to the input and taking a small step in an opposite direction. The *basic iterative method* is an iterative extension of FGSM([3]) with a similar update step, but performed multiple times with pixel wise clipping. [4] show that we can even change the internal learnt representations adversarially. Recently, *deepfool* [5] was proposed to improve upon the above approximate methods. The method accurately approximates the distance to the closest hyperplane in the high dimensional feature space that encloses the source image $\mathbf{I}$, and then computes a perturbation $\mathbf{r}$ such that $(\mathbf{I} + \mathbf{r})$ just goes over the decision boundary.

Fig. 1: Robustness across various distortions: The figure shows(left to right): Original Fooling Image, Gaussian Noise, Rotation, Scaling and Translation. Various existing methods are shown(top to bottom): FGSM, Iterative, DeepFool, Ours. The source class is *Joystick* and the adversarial class is *Bee*. Images with a green box are classified as *Bee*. It is seen that our adversarial image generation algorithm is able to fool the network under all perturbations.

On the other hand, research has shown that these adversaries can be easily defended against, by either modifying the loss function [6], binning the color space [7] or using various forms of defensive distillation [8, 9]. Since most of the fooling techniques make very small adversarial perturbations, they are highly fragile in the presence of natural perturbations (camera noise, rotation, shift, scaling etc.) [10], making fooling deep neural networks very difficult in practice.

The key to generating robust adversarial examples([11]) lies in inserting class specific "structure" in the images which makes them invariant to natural perturbations. It has been shown that one can produce adversarial images with rich geometric structure using *CPPN* and *Genetic Algorithms*([12]). [13] approach the robustness problem by simultaneously fooling an *ensemble of networks*, each operating at a different scale, rotation and shift. This makes the resultant image ro-

bust to all such natural transformations. However, all these methods produce unnatural distributions in the features learnt by the underlying DNN, and hence are detectable. [14] tap into the activations at different layers of a CNN, and generate a score to differentiate an adversarial sample from benign.

We argue that there is an inherent trade-off between robustness and detectability in various adversarial techniques proposed in recent works. Through our experiments, we conclude that there are still no techniques which we can generate adversarial samples that are robust to sensor noise and distortions while still remaining undetectable to meta detectors. In this work, we make incremental advances towards the objective. The specific contributions of this paper are:

1. We evaluate various state of the art adversarial image generation techniques in terms of robustness and detectability and make observations on the structure of adversarial activations and compare them to non-adversarial activations. We propose a simple adversarial detector that reliably separates adversarial inputs from non-adversarial ones.

2. We propose two new robust adversarial image generation algorithms which are difficult to detect. The first one takes as input a source image database (this can be the data set on which a network is trained on) and generates adversarial images just by stitching together various portions of images from the database. In the second algorithm, we suggest to generate adversarial images while keeping activations at all intermediate layers of the network similar to the training set, making the generated adversarial example extremely hard to detect.

2. PROPOSED METHODS

We explain below our algorithm for detecting adversarial samples generated from existed state of the art. We go on to suggest two new algorithms for adversarial generation, trading off robustness for detectability. Given the space restrictions, we have kept the description brief. More details can be found at the project page: `http://www.iiitd.edu.in/~chetan/projects/deepnetfooling`.

Detecting Adversarial Images By carefully evaluating some recently proposed DNN fooling methods ([3, 5, 15, 16, 17, 18]), we observe that the adversarial images generated by these methods can be easily detected to be malicious. Investigating the internal layer activations of the DNN when fed in adversarial images, we find that adversarial activations show distribution graphs which are much more "peakier" than their non-adversarial counterparts. This motivates us to suggest a meta-algorithm to detect adversarial samples which works by simply looking at any change in the distributions of the layer-wise activations.

We build a 2 class Support Vector Machine(SVM), using activations of the network on giving an image as the input. For each image in the train dataset (non-adversarial), we obtain the DNN activations at the intermediate layers, and then flatten and concatenate all these activations to obtain the input vectors corresponding to the first class. Similarly, for obtaining the second class input vectors, we generate adversarial samples for each of the classes and obtain the DNN activations for these samples.

As we show in the experiments section later, we find that this simple classifier performs quite well in practice in terms of detecting any malicious images generated by [3] and [5].

Strong Fooling Algorithm The strong performance of our relatively simple meta classifier for detecting adversarial images motivated us to explore a method of generating images that can fool a DNN. We observe that all existing adversarial detectors, including ours, try to detect a change in the statistics of the activations at some layer of a DNN. Hence, by making all the intermediate activations of an adversarial image close to those of a non-adversarial image, we can make it hard to detect. Accordingly, we modify the source image so that all the intermediate layer activations of a DNN become close to that of an image selected from the target class. Specifically, given a source image I_0 and an image J belonging to the target class, we solve the following optimization problem to obtain the adversarial sample:

$$\min_I \Sigma_k \|\phi_k(I) - \phi_k(J)\|_2^2,$$

where $\phi_k(I)$ denotes the activations at layer k for image I. The optimization objective is the sum of the differences in the layer activations output between generated image and an image of the target class. We start with $I = I_0$ and use the Adam Optimizer([19]) to update I. We normalize the images to $[0, 1]$ and maintain it within the range by clipping at the end of each iteration.

As we show in the experiments, the proposed algorithm generates adversarial images which are much harder to detect than the state of the art. However, we also observe that the images generated by this algorithm tend to be less robust against distortions compared to competitive methods. Trading detectability for robustness, our new method, described below, builds up on the Image Stitching algorithm of [20] and creates robust as well as hard to detect adversarial samples.

Geometric Fooling Algorithm The key to achieving the robustness objective while maintaining undetectability is to pick important parts/structures from the input set which when present in an arbitrary image can make it look like the target class to a DNN. To achieve the goal, we begin with selecting a set of candidate images, say $\mathcal{D}$, which as is may not be classified as the target class by the DNN but do still manage to get a high probability score from the DNN (for the target class). This step is not very critical for the proposed algorithm, but weeding out images with a low confidence score at an early

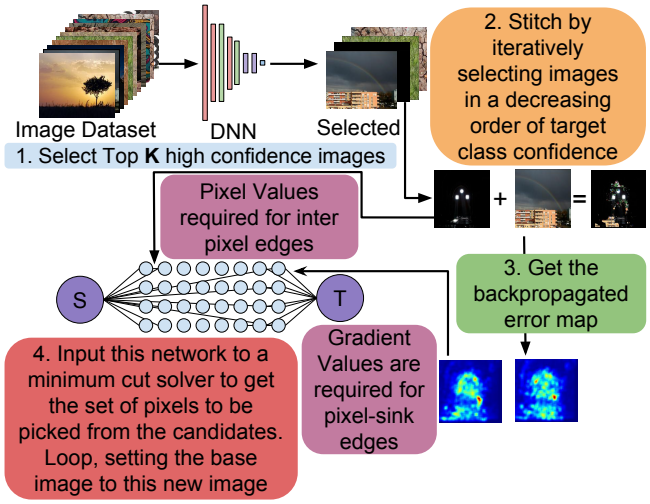

Fig. 2: Block diagram of the Geometric Fooling technique

stage ensures that the algorithm makes rapid progress in generating adversarial samples.

Once the candidates have been selected, we sort them in decreasing order of DNN confidence for the target class. Now we start the iterative procedure(Alg. 1), whereby in each step we pick the next image from the candidate set in sorted order. This image is then paired with the current base image. For each image in this pair, we take a forward pass through the DNN and then back-propagate the error for the target class. This gives us a per pixel error map for each image. To generate a noise robust and natural looking image from the two images in the pair on the basis of their error map, we would like to pick important parts/structures rather than individual pixels.

Kwatra et al. [20] have suggested a graph based technique to stitch two images by finding a minimum cut in an appropriately constructed flow graph. We use their technique and create a graph $G(V, E)$ where V is the set of nodes (referred to as Pixel nodes) and are equal to the number of pixels in the image. We also add special nodes s and t and add edges between s to pixel nodes and from pixel nodes to t. We also add undirected edges between two pixel nodes if they are neighbours spatially(Line 4 of Alg. 2). The edges are supplied weights as defined in Line 5 of Alg. 2. Regularising constants Γ and λ are used to balance between the adversarial image confidence and structure preservation in the output image.

We then find the minimum cut in the created flow graph, and compute sets S and T containing nodes s and t respectively. For all the pixel nodes which are in S set, we copy the value of pixel from the second candidate image to the output. Remaining pixels are copied from first candidate image.

The process is repeated each time by using the output image of the previous iteration (as the base image) and the next candidate image in order. We iterate until the output image is declared as the target class by the DNN. Alg. 1 gives a complete description with Alg. 2 as a subroutine. Fig. 2 gives a block diagram of our method. Though we have given a method to generate $\mathcal{D}$, the algorithm itself is independent of it, and any other $\mathcal{D}$ can be provided by the user.

Algorithm 1: Stitch Adversary Creation

Input: Image List $\mathcal{D}$, Target Class T, Steps K
Output: Adversarial sample A_k
1 Sort $\mathcal{D}$ in decreasing order of confidence for class T
2 Initialize $A_1 \leftarrow \mathcal{D}_1$
3 **for** *each image* $\mathcal{D}_i$ *in sorted order,* $i \in [2:K]$ **do**
4 $\quad A_i$ = Stitch($A_{i-1}, \mathcal{D}_i, T$) {# Algorithm 2}
5 **end**

Algorithm 2: Image Stitching Algorithm

Input: Image A, Candidate C, Target Class T
Output: New Adversarial Image A'
1 $P_A, P_C \leftarrow$ Confidence of A and C for T respectively
2 $B_A, B_C \leftarrow$ Error map backpropagated from the DNN setting target class as T for A, C respectively
3 Normalize B_A and B_C to $[0, 1]$
4 Let $G = (V, E)$ where $V = \{v_{i,j} \mid 1 \leq i \leq \text{width}(A), 1 \leq j \leq \text{height}(A)\} \cup \{s, t\}$ and $E = \{(v_{i,j}, v_{a,b}) : |i-a| + |j-b| \leq 1\}$
5 Let E_{pq} be the edge between pixels p and q. The edge weight function $w : E \rightarrow \mathbb{R}$ is defined as:
$w(v_p, v_q) = \lambda \cdot (\|A(p) - C(p)\|_1 + \|A(q) - C(q)\|_1)$
$w(s, v_p) = B_C(p) + P_A \cdot \Gamma$
$w(v_p, t) = B_A(p) + P_C \cdot \Gamma$, where Γ, λ are tunable
6 Compute max $s-t$ flow in G, let $(S, T) \leftarrow$ mincut(G)
7 Compute new adversarial image A' by copying values of pixels in set S from A and other pixels from C

3. EXPERIMENTS

We now describe our experiments aimed at evaluating existing adversarial image generation techniques and subsequently look at the proposed defense and attacks from a joint perspective of robustness and detectability. We aim to show how existing methods perform poorly on a "practical" fooling metric, and how our methods improve upon them. All the experiments are performed on a 3 Ghz CPU with 8 GB RAM and Nvidia GTX 1080 Ti GPU card.

We use the datasets **CIFAR-10** and **ImageNet-10**(a subset of ImageNet created by randomly selecting 10 synsets [1] from the set of available ImageNet synsets, following [14]). For each class, we generate 500 training and 100 testing adversarial examples. For CIFAR-10, we use the model architecture shown in Table 3. For ImageNet-10, we use the commonly used VGG-16 Architecture [21].

The robustness score is created to capture the extent to

[1] *bee*, *cardigan*, *confectionary*, *dugong*, *joystick*, *modem*, *palace*, *persian cat*, *stone wall*, *valley*

Dataset	O	FG	BI	DF	SF	G
Gaussian	0.50	0.56	0.89	0.82	0.42	0.59
Rotation	0.50	0.44	0.72	0.67	0.31	0.61
Translation	0.55	0.39	0.72	0.67	0.38	0.66
Scaling	0.61	0.51	0.82	0.76	0.43	0.74
Average	0.54	0.47	0.78	0.73	0.38	0.65

Table 1: Robustness Scores on CIFAR-10. Please refer to the text for methods compared.

Dataset	FG	BI	DF	SF	G
`conv2`	0.48 (0.99)	0.78 (1.00)	1.19 (0.54)	0.97 (0.41)	1.33 (0.32)
`dense1`	0.53 (0.93)	0.79 (0.99)	1.27 (0.46)	1.16 (0.22)	1.39 (0.26)

Table 2: Combined Scores for CIFAR-10, calculated as (Robustness Average) + (1−Detectability). The score in the bracket denotes the detectability score

which a generated adversarial sample is resistant to a particular perturbation. In other words, it is a measure of the extent to which an adversarial sample gets classified as the target class even after adding noise/distortion. For original, non-adversarial images, the robustness scores reported denote the extent upto which the image is classified as the original class on adding noise/distortion.

For **noise robustness** scores we use gaussian noise, $N(\mu, \sigma)$ with varying μ and σ. All the input images are scaled to $[0, 1]$ before feeding in to the networks. At a particular μ and σ, the robustness score for a set of adversarial images, S_{adv}, generated according by the adversary *adv* and DNN f is defined as:

$$R_f(S_{\text{adv}}, \mu, \sigma) = \frac{\Sigma_{I_a \in S_{\text{adv}}}[f(I_a) = f(I_a + r)]}{|S_{\text{adv}}|}, r \sim \mathcal{N}(\mu, \sigma)$$

The reported scores are averages over all pairs of μ and σ, such that $\mu, \sigma \in \{0.0, 0.05, 0.1, 0.15, 0.2, 0.25\}$. We rotate the adversarial images by 0° to 45°, and report **rotation robustness** scores. We translate the adversarial images by upto 30% of the image size in all the four directions and report **translation robustness** scores. We scale the adversarial images from $1.1\times$ to $2\times$, in steps of 0.1, and take an average over the **scaling robustness** scores thus obtained. Finally, we take features from one of the middle layers of the DNN and classify using an SVM for the **detectability** scores. All the scores are also shown on the Original Unperturbed Images(**O**) for comparison.

For FGSM (**FG**, [3]), we perform standard non-targeted fooling with ϵ set to 0.1 in our experiments. For the Iterative method (**BI**, [3]), 10 iterations of the Algorithm are performed, with the step size $\alpha = 0.05$. We perform 50 and 1000 iterations per image for DeepFool(**DF** [5]), and proposed Strong Fooling(**SF**) algorithms, respectively. **G** refers to the proposed Geometric Fooling algorithm(Alg. 1). The robustness scores are shown in Table 1 and Table 4. It can be seen from Table 2 that a detector looking at an intermediate `conv` layer is better than one closer to the output. This can be explained as a decrease of the fooling "effect", as we go deeper into the network.

Layer	Type	Filter Size	Output Size
data	-	-	(32, 32, 3)
conv1	Convolution	(3, 3)	(32, 32, 32)
pool1	Pooling	(2, 2)	(16, 16, 32)
drop1	Dropout(0.25)	-	(16, 16, 32)
conv2	Convolution	(3, 3)	(16, 16, 64)
conv3	Convolution	(3, 3)	(16, 16, 64)
pool2	Pooling	(2, 2)	(8, 8, 64)
drop1	Dropout(0.25)	-	(8, 8, 64)
dense1	Fully Connected	-	(512)
drop2	Dropout(0.5)	-	(512)
dense2	Fully Connected	-	(10)

Table 3: Model Used on CIFAR-10

Dataset	O	FG	BI	DF	SF
Gaussian	0.11	0.16	0.16	0.14	0.16
Rotation	0.44	0.20	0.20	0.18	0.09
Translation	0.54	0.18	0.17	0.14	0.07
Scaling	0.56	0.29	0.27	0.26	0.16
Average	0.41	0.20	0.20	0.17	0.12

Table 4: Robustness Scores on ImageNet-10. Please refer to the text for methods compared.

4. CONCLUSION

In this work, we have introduced the notion of *Practical Fooling* of Deep Neural Networks, where we look at adversarial samples such that they are both undetectable to be an adversarial sample as well as robust to natural perturbations like camera noise, scaling, rotation, shift etc. We found that existing adversarial image generation techniques either create unnatural distributions in the DNN internal activations or make imperceptible changes to the image. In the former case, the fooling image is easily detectable while in the latter case it is not robust. We propose that preserving the natural structure of an image is crucial to generating practical adversarial samples. We have suggested a new method for adversarial image generation, using information from the back-propagated gradient and the pixel intensities to stitch together structurally important regions from the image.

Acknowledgement: Chetan Arora is supported by Infosys Center for Artificial Intelligence and Visvesaraya Young Faculty Research Fellowship by MEITy, Government of India.

5. REFERENCES

[1] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, 2015.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, 2012.

[3] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy, "Explaining and harnessing adversarial examples," *International Conference on Learning Representations*, 2015.

[4] Sara Sabour, Yanshuai Cao, Fartash Faghri, and David J. Fleet, "Adversarial manipulation of deep representations," *International Conference on Learning Representations*, 2016.

[5] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," 2016.

[6] Chunchuan Lyu, Kaizhu Huang, and Hai-Ning Liang, "A unified gradient regularization family for adversarial examples," *IEEE International Conference on Data Mining*, 2015.

[7] Akash V Maharaj, "Improving the adversarial robustness of convnets by reduction of input dimensionality," 2016.

[8] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network," *Neural Information Processing Systems(Workshop)*, 2014.

[9] Nicolas Papernot, Patrick Drew McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," *IEEE Symposium on Security and Privacy*, 2015.

[10] Abigail Graese, Andras Roza, and Terrance E Boult, "Assessing threat of adversarial examples on deep neural networks," in *Machine Learning and Applications (ICMLA), 2016 15th IEEE International Conference on*, 2016.

[11] Nicholas Carlini and David Wagner, "Towards evaluating the robustness of neural networks," in *Security and Privacy (SP), 2017 IEEE Symposium on*, 2017.

[12] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, 2002.

[13] Anish Athalye and Ilya Sutskever, "Synthesizing robust adversarial examples," *arXiv preprint arXiv:1707.07397*, 2017.

[14] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff, "On detecting adversarial perturbations," *arXiv preprint arXiv:1702.04267*, 2017.

[15] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami, "The limitations of deep learning in adversarial settings," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, 2016.

[16] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Xiaolin Hu, Jianguo Li, and Jun Zhu, "Boosting adversarial attacks with momentum," *CoRR*, vol. abs/1710.06081, 2017.

[17] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus, "Intriguing properties of neural networks," *International Conference on Learning Representations*, 2014.

[18] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh, "Ead: elastic-net attacks to deep neural networks via adversarial examples," *arXiv preprint arXiv:1709.04114*, 2017.

[19] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[20] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick, "Graphcut textures: image and video synthesis using graph cuts," in *ACM Transactions on Graphics (ToG)*, 2003.

[21] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.