Adversarial Robustness

Mihir Sutariya 20110208
sutariya.mihirkumar@iitgn.ac.in
IIT Gandhinagar

R Yeeshu Dhurandhar 20110152
r.yeeshu@iitgn.ac.in
IIT Gandhinagar

Sukruta Prakash Midigeshi 20110206
sukruta.midigeshi@iitgn.ac.in
IIT Gandhinagar

ABSTRACT

Deep learning models have found widespread application across various domains, yet they remain susceptible to adversarial attacks exploiting their sensitivity to subtle input perturbations. Specifically focusing on classification models, numerous attack algorithms such as Fast Gradient Sign Method, Projected Gradient Descent, and DeepFool have been developed, alongside corresponding adversarial training strategies aimed at bolstering robustness. In this study, we introduce novel attack algorithms tailored to enhance adversarial robustness and conduct a comprehensive comparative analysis against conventional methods. Adversarial inputs, characterized by imperceptible variations from original inputs, pose significant challenges as they evade human detection. Our research contributes to advancing the understanding of adversarial vulnerabilities in deep learning models, offering insights into effective defense mechanisms.

KEYWORDS

Adversarial attacks, FGSM (Fast Gradient Sign Method), Classification models, Neural networks

R Yeeshu Dhurandhar 20110152 Sukruta Prakash Midigeshi 20110206 R Yeeshu Dhurandhar 20110152 Sukruta Prakash Midigeshi 20110206

ACM Reference Format:
Mihir Sutariya 20110208. 2018. Adversarial Robustness. In . ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Classification models, integral to many machine learning applications, are notably vulnerable to adversarial attacks. These attacks exploit the model's susceptibility to subtle perturbations in input data, leading to misclassifications or erroneous predictions. At the core of adversarial attacks lies the concept of crafting inputs that closely resemble the original inputs while evading human perception. The primary objective of adversarial attacks is to maximize the loss incurred by the model for a particular input. To achieve this objective, various optimization methods have been devised. One prominent method is the Fast Gradient Sign Method (FGSM)[2], which calculates the gradient of the loss function with respect to the input and takes a single step in the direction of the sign of the gradient. Below are equations that define the FGSM algorithm.

$$\delta = \underset{\|\delta\| \leq \epsilon}{\operatorname{argmax}} \ell\left(h_\theta(x+\delta), y\right) \tag{1}$$

Fast Gradient Sign Method (FGSM)

$$g = \nabla_\delta \ell\left(h_\theta(x+\delta), y\right) \tag{2}$$

$$\begin{aligned} \delta &= \epsilon \cdot \operatorname{sign}(g) \\ x_{adv} &= x + \delta \end{aligned} \tag{3}$$

$where,$

$x = originalimage$

$x_{adv} = perturbedimage$

$\delta = perturbation$

$l = lossfunction$

$y = trueclassofimage$

$h_\theta = modelwithparameters\theta$

Here, we have a constraint on δ. It should lie on a defined surface around the origin. This constraint could be changed to another surface. What adversarial training does is first calculate adversarial perturbation and train loss with the input as the original input + perturbation. So, here is one problem with one type of algorithm. This surface we defined could not include all the adversarial input that looks similar to the original. So, we want to show that there are ways to vary the original input so that adversarial input lies outside this surface and crosses the classifying boundary.[1]

2 EXPERIMENTAL SETUP

For all the modified methods, we utilized the ResNet50 model to compare new attack methods with FGSM. This model is trained on the ImageNet dataset and is specifically trained to infer the breed of a dog in a given image.[3]

3 MODIFIED METHODS

We developed 3 methods that use the vulnerability in the training algorithm and form new attacks. Below is the list

- Modified FGSM attack
- Color-based Functional attack
- Modified DeepFool attack

3.1 Modefied FGSM attack

3.1.1 ***Theory.*** We will make use of the vulnerability of FGSM. FGSM assumes that all the pixels of an image can change to the extreme boundary ϵ, then the image will be similar to the original image. This method changes too many pixels, and images don't look similar if ϵ is high.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

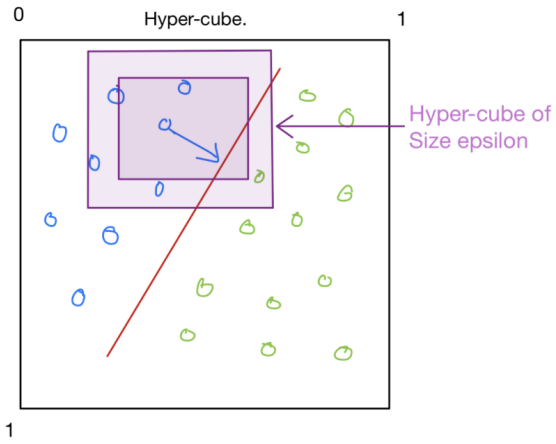

Figure 1: FGSM perturbation boundary

To form an adversarial image similar to the original image, there are 2 objectives:

- Vary fewer number of pixels
- Vary values of pixels by a small amount

To fulfill both requirements, we need to find pixels that contribute more to loss. If we are able to find this, then we can vary these pixels by a lesser amount, and the change in loss will be higher without varying other pixels.

$$g = \nabla_\delta \ell\left(h_\theta(x+\delta), y\right) \tag{4}$$

$$g_{sorted} = sort\_reverse\left(|\nabla_\delta \ell\left(h_\theta(x+\delta), y\right)|\right) \tag{5}$$

$$\delta = \epsilon_1 \cdot \text{sign}\left(g_A\right) + \epsilon_2 \cdot \text{sign}\left(g_B\right) \tag{6}$$

$$x_{adv} = x + \delta \tag{7}$$

$$\text{k pixels} \quad : \epsilon_1$$
$$(n-k) \text{ pixels} \quad : \epsilon_2$$

Formally, we will find the gradient with respect to the image. Now, we will sort the gradient value based on its absolute value (figure **??**). The higher the value, the higher the chance of a change in loss when we change the pixel value. So, now we can select the first k pixels and form set A, and the rest belong to set B. If we sum set A gradients and set B gradients, then these 2 different directions will be perpendicular to each other. Set A direction will be towards the boundary, and set B direction will be perpendicular to set A direction (Figure 3).

3.1.2 ***Experiments and Results.*** First, we see how varying ϵ changes an original image. When we pick $\epsilon = 1$ we can see where model is giving more weights to pixels (Figure 4). Observe that, with varying epsilon up to 0.2, the adversarial image is similar to the original image.

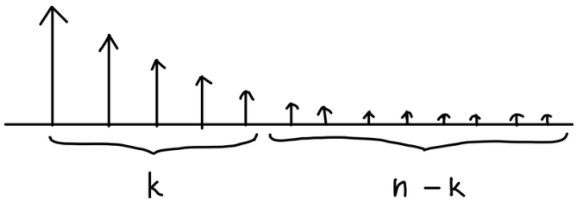

Figure 2: Sorted gradient values based on absolute values of gradients

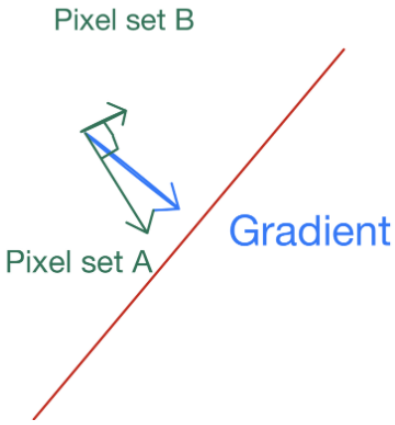

Figure 3: Partition of pixel's gradients in set A and set B

Now, comparing with FGSM transformed image. In Figure 5, we can see that the same ϵ, FGSM transformed image looks more noisy than the modified FGSM transformed image. Further confidence in predicting the wrong class has been increased in modified FGSM attacks. Figure 6 shows the variation of the probability of the true class v/s epsilon value graph. You can see that the graph is decreasing as we increase the value of epsilon.

3.2 Color-based Functional attack

3.2.1 ***Theory.*** The objective of perturbation is to change the image so that the classifier classifies it wrong, but the changes should not be visible to human eyes. In an image, the nearby pixel values are similar for most places. Even the channel-wise pixel values are similar. Now, let's say, for a channel, if the gradient of adjacent pixels comes out to be in the opposite direction, then the final color values of those adjacent pixels will be different after the update. Such changes make it easy for the human eye to see the difference in the image compared to the original image. Therefore, we propose an approach in which we change all pixels with the same channel-wise pixel values in the same direction. For example, for the red channel, if three-pixel values (Figure 7) are the same, say 10, we calculate the gradient for them as usual and sum them up (or take

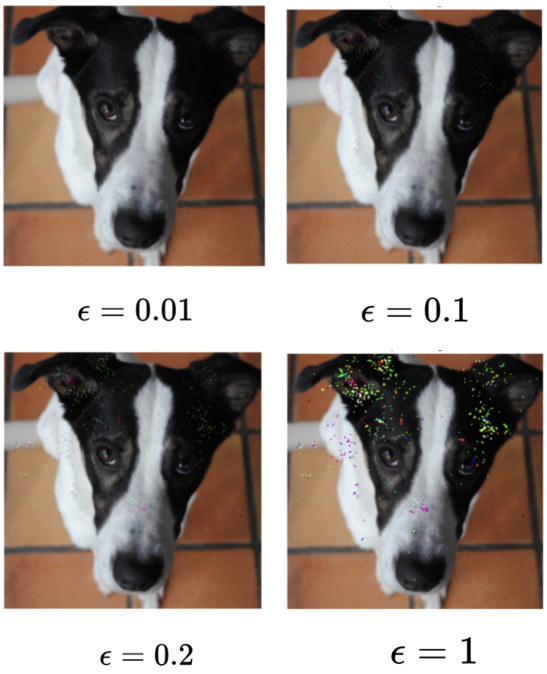

Figure 4: Effect of varying ϵ on modified FGSM images

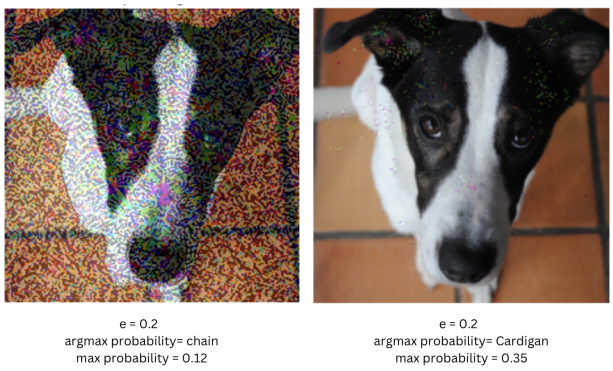

Figure 5: Comparison of Modified FGSM with FGSM. Left: FGSM, Right: Modified FGSM.

the mean; it does not matter since we need direction only) of those gradients. Then, we change these three pixels in the direction of the sum of the original gradients. This results in varying the values in the same direction, making it hard for the human eye to see the variations in the image.

If we group the pixels based upon the same values of all three channels, we will get 255x255x255=16,581,375 groups, which is overkill for the given task and will cause high computational cost. Therefore, for simplicity, we consider the three channels individually and make groups based on the channel values of each pixel. In this way, the number of groups reduces to only 255+255+255 = 765 groups.

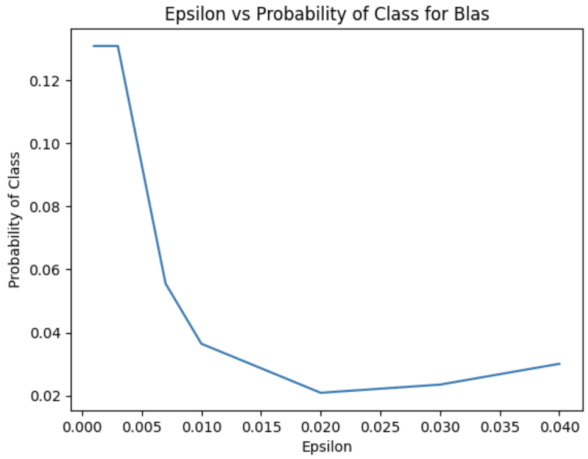

Figure 6: Probability of true class v/s epsilon plot for Modified FGSM

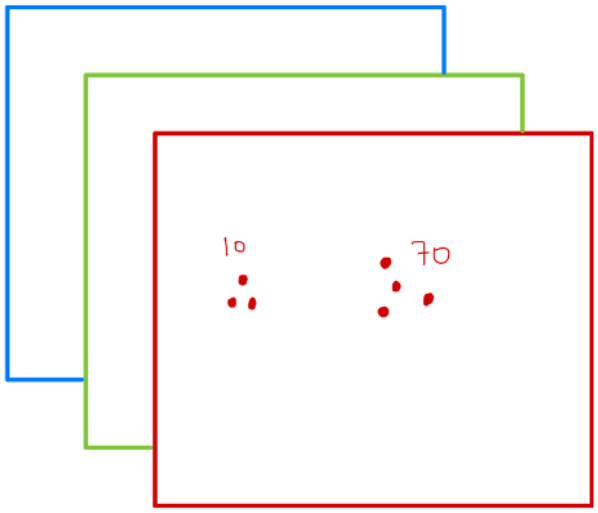

Figure 7: RGB Channel of an Image. Pixels with the same value channel are grouped together.

The functional attack can be represented in the mathematical form as follows:

$$F(x) = x_{adv} = (f(x_1), f(x_2), ..., f(x_i), ..., f(x_n)) \quad (8)$$

$$f(x_i) = (f_1(x_{ir}), f_2(x_{ig}), f_3(x_{ib})) \quad (9)$$

where,
x = *original image*
$F(x)$ = *perbation on image* x
$f(x)$ = *perturbation for each channel*
x_{adv} = *perturbed image*
x_i = *a pixel* i
x_{ir} = *red channel of pixel* i
x_{ig} = *green channel of pixel* i
x_{ib} = *blue channel of pixel* i

Note that the function $F(x) = f(x_i)$ is the same for all the pixels, meaning the same function will be applied to all the pixels. These are the characteristics of the functional attack.

Now, the group-wise gradient can be calculated as:

$$g_{cj} = \nabla_{cj}\ell = \sum_i \nabla_\delta \ell\left(h_\theta(x_{ic} + \delta), y\right) \tag{10}$$

where,
c = *color* $\in \{r, g, b\}$
i = *pixel number* $\in \{1, 2, ..., n\}$
j = *pixel values* $\in \{0, 1, ..., 255\}$
n = *number of pixels*
ℓ = *loss functions*
δ = *perturbation*
h = *model*
θ = *model parameters*

These g_{cj}'s combine to give rise to δ. Then, the update equation will follow the same steps as the FGSM attack.

$$\delta = \epsilon \cdot \text{sign}(g) \tag{11}$$

$$x_{adv} = x + \delta \tag{12}$$

3.2.2 ***Experiments and Results.*** First, we see how varying ϵ changes an original image (Figure 8). We can observe the pattern arising in the image. Similar color pixels undergo similar changes. For $\epsilon = 0.1$, the image looks similar, however, after this, the changes are visible to human eye. For $\epsilon = 1$, we can observe the color patterns clearly.

Now, compare the results of the functional attack with the FGSM-transformed image. In Figure 9, we can see that the same ϵ FGSM transformed image looks overall noisy, but the functional attacked image looks less noisy. Further confidence in predicting the wrong class has increased significantly in the functional attacks, indicating our model's good performance.

Figure 10 shows the variation of probability of true class v/s epsilon value graph. The overall trend is decreasing. However, the spike in the middle is due to non-linear boundaries in hyperspace.

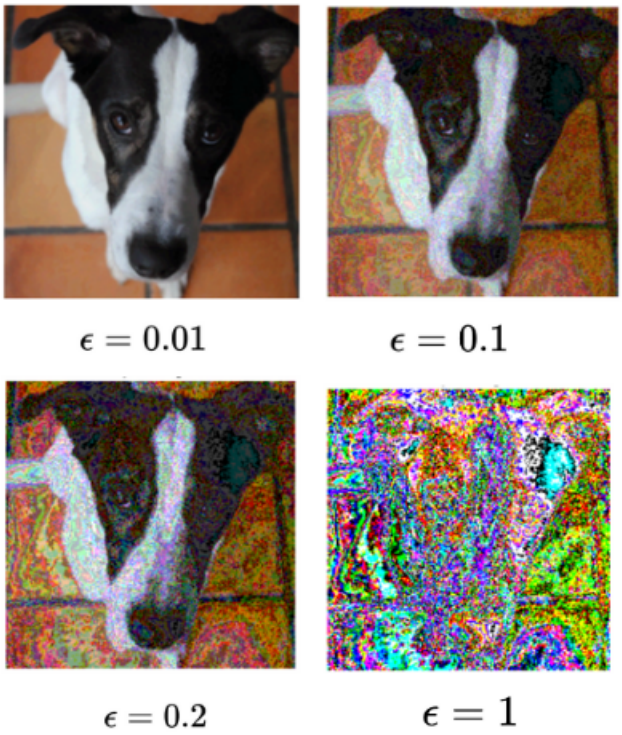

Figure 8: Effect of varying epsilon on color-based functional attacked images

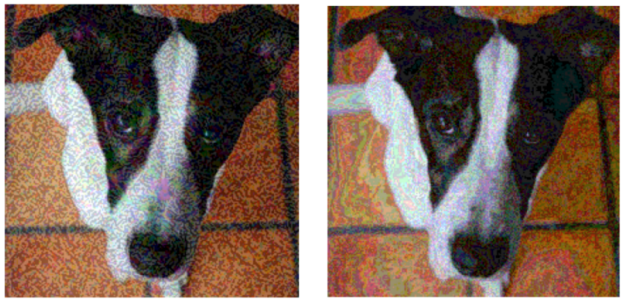

Figure 9: Comparison of functional attack vs FGSM. Left: FGSM, Right: Functional Attack

3.3 Modified DeepFool

3.3.1 ***Basic Idea:*** In the paper for DeepFool, an adversarial perturbation is defined as the minimal perturbation r that is sufficient to change the estimated label $\hat{k}(x)$:

$$\Delta(x, \hat{k}) := \min_r \|r\|_2 \quad \text{subject to} \quad \hat{k}(x + r) \neq \hat{k}(x),$$

where x represents an image and $\hat{k}(x)$ is the estimated label. The term $\Delta(x, \hat{k})$ denotes the robustness of the classifier $\hat{k}$ at point x.

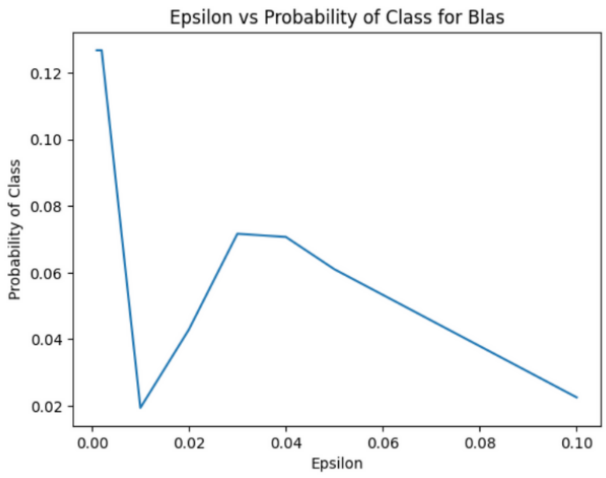

Figure 10: Probability of true class v/s epsilon plot for color-based functional attack

The primary assumption of the method is that the given classifier has linear boundaries around the point taken at each step. Following this assumption we can find the distance between the point as the closest border. For a binary classification task, with linear function $f(x) = w^T x + b$ and boundary $w^T x + b = 0$. The optimization problem will be -

$$\arg\min_{r_i} \quad \|r_i\|_2 \quad \text{subject to} \quad f(x_i) + \nabla f(x_i)^T r_i = 0$$

Thus, the update in each iteration can be calculated as $r_i \leftarrow -\frac{f(x_i)}{\|\nabla f(x_i)\|_2^2} \nabla f(x_i)$

This can also be extended to multi-class classification problems. Let us assume that there are k classes, with linear functions $f_k(x) = w_k^T x + b_k$. Let $\hat{k}$ be the current class of x and $w_k = \nabla f_k(x_i)$. Now the optimization function would be -

$$\arg\min_{r} \|r\|_2^2 \quad \text{s.t.} \quad \exists k : w_k^T(x_0 + r) + b_k \geq w_{\hat{k}}^T(x_0 + r) + b_{\hat{k}}$$

Thus, the closed class $\hat{l}$ will be as follows :

$$\hat{l}(x_0) = \arg\min_{k \neq \hat{k}(x_0)} \frac{|f_k(x_0) - f_{\hat{k}}(x_0)|}{\|w_k - w_{\hat{k}}(x_0)\|_2^2}$$

3.3.2 ***Implementation:*** We created an architecture to implement the DeepFool method on the imagenet dataset. We also compared the performance of these attacks with that of FGSM for the same level of confidence. Following were some observations :

(1) The algorithm was iterating through all the 1000 classes of imagenet to find the optimal class for each iteration.
(2) While the method performs much better than FGSM, it is unable to reach higher levels of confidence even with a large number of iterations.

3.3.3 ***DeepFool 2.0.*** Based on our observations, we made the following changes to the algorithm:

(1) Since there are a large number of classes, and each iteration takes $O(K)$ time, the runtime of the algorithm is too high. This in the initial iteration we select only the classes that return the highest gradient values and run the search only amongst these classes.
(2) In order to make sure that the confidence levels increased at a faster pace, at each iteration we only update those perturbation values that are in the direction of the gradient with respect to the original class. This will make sure that the updates do not ever occur in the favour of the original class.

Thus the final formula would be as follows:

$$x_o[i] = \begin{cases} x_o[i] + (1+\alpha) \cdot r_*(x_o)[i] & \text{sign}\nabla f_{\hat{k}}(x_0)[i] = \text{sign} r_*(x_o)[i] \\ x_o[i] + (1-\alpha) \cdot r_*(x_o)[i] & \text{otherwise} \end{cases}$$

3.3.4 ***Experimental results:***

(1) The time complexity of each iteration decreased overall, however there was increase in the time taken due to checking the sign of each dimension of the perturbation for the updates.
(2) There was a surprising increase in the efficiency of the algorithm with it reaching much higher levels of confidence with lesser number of iterations. It reached a level of confidence of 0.3 within 1 7iteration, which the original algorithm failed to do even with 100 iteration.

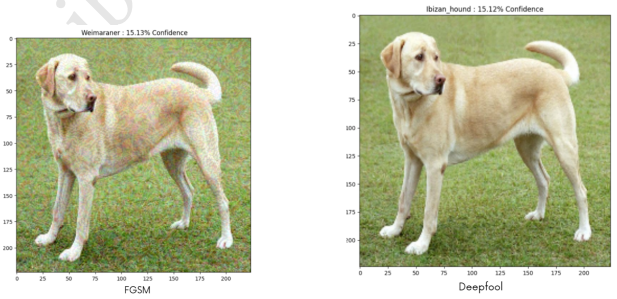

Figure 11: Comparison between FGSM and DeepFool

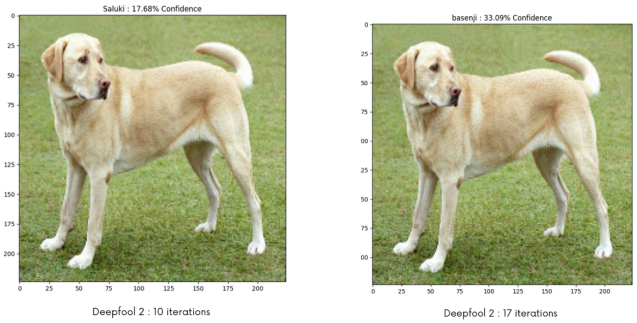

Figure 12: Improved results obtained using the new algorithm

4 CONCLUSION

In conclusion, we have demonstrated a method to optimize the inner maximization problem to formulate a more successful attack than FGSM. Comparing it with FGSM, we have presented metrics illustrating that the modified attacks are significantly superior to traditional approaches. Currently, there are no algorithms that train models robust to these attacks. Moreover, we have highlighted the challenge in formalizing the similarity between two images with respect to the human eye. Nevertheless, we have shown that by optimizing the inner maximization, it is possible to create adversarial images that appear similar to the original ones while successfully executing the attack. Particularly, achieving adversarial robustness poses a significant challenge when dealing with high-dimensional inputs such as images.

5 FUTURE WORK

Our research questions the robustness of adversarial training to all types of attacks and emphasizes the difficulty in formalizing the notion of similarity between images from the perspective of human perception. Future work could focus on designing an adversarial training algorithm that is resilient to all types of attacks and does not adhere to constraints imposed by specific boundary shapes, as FGSM does in adversarial training. Another intriguing approach could involve designing a network capable of distinguishing between adversarial and original images, with the added requirement that this network itself remains robust to adversarial attacks.

REFERENCES

[1] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and Harnessing Adversarial Examples. *arXiv* (2014). https://doi.org/10.48550/arxiv.1412.6572

[2] Zico Kolter and Aleksander Madry. 2018. Adversarial Robustness: Theory and Practice. Website. https://adversarial-ml-tutorial.org/ NeurIPS 2018 tutorial.

[3] marcusGH. 2022. Adversarial Attacks on ImageNet Models. Website. https://github.com/marcusGH/adversarial-attacks-on-imagenet-models