Optimizing the Targeted, Decision-Based Attack

Andy Carluccio, Jake Fullerton, Matthew McDonnell, Josh Peterson, Hank Weber
University of Virginia
Charlottesville, VA, United States

***Abstract* - This paper discusses optimizations to decision-based adversarial attacks attacks on image classifiers, expanding on the work of Brendel, Rauber, and Bethge in Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. Given only a return tag, their algorithm can create an image that appears as one object to a human but is classified as a different object by a computer, but it does so at the expense of several million queries to the image classifier in order to successfully "fool" it. This research project explores using pixel difference heuristics to reduce the number of queries sent to the image classifier in the decision-based attack. Does analyzing the pixel differences between the source and target yield any useful information for the attack, and if so, how should this data best be integrated into the current algorithm to reduce queries most effectively?**

I. Introduction

Many different systems rely on computer vision. As the algorithms have improved in accuracy and usability, they have been useful for various purposes including screening uploaded content, running self-driving cars, and searching online for images. However, there is still a large gap between these algorithms and the human mind. Consequently, they are imperfect and can misclassify images due to minor perturbations. These perturbations can be imperceptible to the human eye and yet still cause the model to classify an object as something else. Adversarial attacks are algorithms that seek to find these perturbations and fool image classifiers.

Adversarial attacks have gained interest from both the people looking to take advantage of vulnerabilities and those working to maintain security of their systems. There are various categories of adversarial attacks, including Gradient-based, Score-based, Transfer-based, and Decision-based. For this research, the focus was placed on Decision-based attacks.

The goal of this research is to optimize a targeted decision-based adversarial attack, which attempts to make the image classifier incorrectly classify an image as a specific class of object. One of the primary problems with decision-based attacks is that they require millions of queries to an image classifier in order to produce a properly modified image. Building on the work of Brendel, Rauber, and Bethge in Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models and the Foolbox code base, modifications to the attack code and test their effectiveness were created.

The following sections will outline the design and methods used to approach this problem in the context of related work and explore the findings of development efforts.

II. Related Work

As image recognition software becomes more prevalent in society, it is increasingly important to prevent attacks that can circumvent this new method of security. One of the main usages of image recognition in security is to recognize weapons and give security teams a warning. One problem with this presents is that there can be both false positives and negatives with this system. A team a MIT recently 3D printed a turtle that

was recognized as a gun by image recognition software [1]. This was accomplished by taking a 3D image and slightly warping an object's texture over its shape so that Google's image recognition model would recognize it as something completely different. This is extremely powerful because not only does the algorithm become fooled when looking directly at the object, it can be fooled from any angle of view. With the 3D printed turtle, the algorithm was fooled 82% of the time. This could lead to many dangerous situations if bad actors can create distractions that can cause whole security systems to fail.

The MIT team also spoke about another usage of 3D images that can fool image recognition algorithms: "A fairly direct application of 3D adversarial objects could be designing a T-shirt which lets people rob a store without raising any alarms because they're classified as a car by the security camera"[1].

Another case where fooling an image recognition algorithm could cause major security problems involves eBay, which uses image recognition software to check all merchandise photos for auctions to verify that illegal items are not being sold on their site under a false name. eBay does not allow the sale of live animals on their site, for example, so when pictures for an auction are uploaded, the images are checked to make sure they don't contain animals. Most of the time, this works, however, using a certain algorithm, a user can change an image to be identified as whatever they want. This algorithm works in four steps:

1. Feed in the photo for modification.

2. Check the neural network's prediction and see how far off the result is from the classification desired for this photo.

3. Tweak the photo using back-propagation to make the final prediction slightly closer to the desired answer.

4. Repeat steps 1–3 a few thousand times with the same photo until the network gives us the desired answer.

III. Attack Model

All of the attacks were performed on the ImageNet image classifier. This was achieved using foolbox to send the queries to the classifier during the attack. After the attack, the visualizations were created using the matplotlib python library. The attack was edited in the boundary attack code provided in the foolbox github by Jonas Rauber and Wieland Brendel.

IV. Design

At a glance, it seems that a significant optimization for the decision-based attack should involve a modification or replacement of the random-perturbation method of determining which pixels should be modified. The heuristic proposal is that pixel difference between the source and target images may be a suitable source of data for optimizing this step of the attack. This proposal assumes that the image classifier is concerned with quality over quantity of similar pixels between the two images, an assumption that would later be tested during implementation. When classifying an image, it seems likely that pixels or regions of greatest difference would

be important to determining how to tag an image, though this is not necessarily a guarantee (for example, edge detection could be another quality of insight for a classifier).

In brief, this heuristic is designed to prioritize modifications by the decision-based algorithm first to areas where the source and target images have the greatest degree of difference. This optimization may reduce the number of queries required because image classifiers may look to these differences first when determining which tags to assign, and by modifying these locations first, the algorithm may achieve a successful result without wasting time modifying areas of an image that are not prioritized by the classifier. This heuristic-based strategy does not involve any additional queries to the classifier and could be considered an additional "setup" step instead.

Before implementing this heuristic modification, it first seemed sensible to design a preparatory experiment to determine if pixel difference was at all an influence on the image classifier. To test this hypothesis, a simple image "blending" program was designed to modify the source image using information about the target image before passing it into the vanilla decision-based attack algorithm. By examining the distance per step metric, a determination about the influence of pixel difference would be easier to craft.

Two versions of this preparatory experiment were designed. The first was a literal pixel replacement on the source image with pixels from the target image if a candidate pixel was "sufficiently *different*" from its corresponding pixel in the target image. This "quality approach" modified regions of greatest difference by replacing pixels on the source image directly. The second experiment was a "quantity approach" where the pixels were replaced on the source image if they were "sufficiently *similar"* to pixels in the target image, the reverse approach. The resulting images were then passed into the vanilla decision-based attack algorithm, and the distances per step were examined to the same effect. To clarify, these experiments were not the heuristic modification itself, but instead they informed the design of the modification as explained below.

The first step of the heuristic modification is, again, to identify pixels of greatest difference between the source and target images. The random perturbation is now replaced with a weighted perturbation, where pixels ranked as having the highest difference receive the most weight and pixels ranked as having the least difference receive the least weight. The decision-based attack is otherwise unchanged except for the new weighted perturbation.

To examine the results, there are two methods of analysis. First, by fixing the number of steps for both the vanilla and modified attack, compare the distance and distance-per step between each algorithm. Second, select a distance upon which the human eye cannot reliably discern difference and have each algorithm halt after reaching that distance, allowing for the comparison of

the number of steps (and queries) required to reach that distance.

V. Implementation

As outlined previously, work began with the idea that the first pixels to be modified should be the pixels with the most difference. To generate the list of pixels sorted based on difference between the current adversarial image and the target image, code was written that looped through the pixels, calculated the L2-norm of each pixel, and then sorted based on the L2-norm values. The pixels with the highest associated L2-norm were the ones that needed to be modified first to get closer to the target image.

The next issue was how much to change the pixels that were identified during this process. Four different methods of changing the pixels with the greatest difference were developed:

1. Find the magnitude of the random perturbation created by foolbox. Zero out all values representing the direction of change towards the target image except for the pixel of most difference. Make the magnitude of the direction to change for the pixel of most difference equal to the magnitude of the original random perturbation.

2. Perform the same steps described in 1, except distribute the magnitude across the top 100 pixels.

3. Find the top 2% of pixels with the most difference. Use the random perturbation generated by foolbox, increase the values in the random perturbation for the top 2% by 10% and decrease all other pixels by 20%.

4. Find the L2-norm for all pixels and map the values to a scale from 0.75-1.25. Go through each pixel in the random perturbation and multiply the values by the corresponding, scaled value from the L2-norm.

Each of these methods had significantly different degrees of success as discussed in the evaluation and conclusion sections. Method 1 was developed because it seemed reasonable that by focusing on a single pixel at a time, the process would be simplified and would reach the final adversarial once each pixel had been reduced as much as possible. Method 1 did not work out as well as had been desired, so work progressed to Method 2, which was thought to work better because it distributed the magnitude of change across more pixels and wasn't as focused as the original attempt. The results still were not an improvement to the random perturbation, so progress moved to Method 3, which provided promising results. Method 4 was developed as a more methodical and efficient version of method 3 because the pixels of most difference were being changed the same amount as some of the less different pixels, and not all pixels were being changed in method three. Method 4 provided the best results, but should still be

tested for possible optimizations and improvements in the future.

VI. Evaluation

There are many factors that can be examined when evaluating the effectiveness of changes to the attack. Certain statistics such as the number of steps that successfully reduced the distance, the average distance reduced per step, and the number of steps to reach a target distance were all assessed in determining the effectiveness of our modified attack. It was decided that the most important measurement would be the total number of steps to reach the target distance, and evaluations ended up using the other metrics to examine if test methods were making improvements (whether it was making more significant distance reduction per step and/or having a higher number of steps that make a reduction in distance).

The first two methods did not work out as initially hoped for; they both had very significant distance reduction over the first 50-100 steps. This was likely because they were so focused that they made large improvements on the first pixels because they could be changed while still remaining adversarial, but once they reached pixels of less difference, the changes were too much to remain adversarial.

The third method was developed because it seemed reasonable that the random perturbation was critical to making distance reduction once the number of steps reached very large numbers and the adversarial image started to get very close to the boundary of the target class. Weighting the perturbation towards the pixels of most difference would help make more distance reductions at each step. With method 3, 83% of the steps made a successful reduction in distance while remaining adversarial and the average distance reduced per step was 0.17%. The regular foolbox code reduced the distance 90% of the time and had an average distance reduction of 0.16%. While method 3 did not make a reduction as often as the regular code, it did make a more significant reduction on average, which was desirable. This method was also able to reduce the total number of steps taken to reach the target distance by 12.67% of the regular foolbox code.

Because Method 3 was relatively arbitrary by increasing the pixels identified by the heuristic by 10% and decreasing the other pixels by 20%, Method 4 was developed. With the fourth method, even better improvements were achieved over the third method; the heuristic modification was now able to reduce the total number of steps taken to reach the target distance by 18.78%. While these results are not as drastic as some had hoped, it does show that there are significant opportunities to improve the fully random method that foolbox has implemented.

There were a few disadvantages to the methods implemented that were able to achieve these results. The heuristic had to use additional time to go through all of the pixels in the image, calculate the distance, and then sort the list of values. It also had to go through the random perturbation and increase/decrease

the weight according to this method. This increased the runtime of each step by 15x, which caused testing to progress very slowly and limited the number of tests that could be feasibly run in the available testing period.

VII. Conclusions

The main goal of this project was to improve the existing targeted decision-based attack by reducing the number of queries it requires to an image classifier. The heuristic modification was able to produce roughly a 13% decrease in this number.

The majority of development efforts were spent trying to replace the queue of random perturbations with one generated off a meaningful metric, and after a series of different approaches, satisfactory understanding of what was important when choosing pixels to modify was attained. Findings indicate that this method for altering the adversarial image reduces the distance between the original image and the adversarial faster than the random perturbations.

Throughout the process, there were a handful of challenges were encountered. After getting over the high learning curve required to understand both the material and the code base, development advances were accelerated. However, optimizing for reduced queries to the classifier resulted in greatly increased runtime despite workspace acceleration techniques. This issue will continue to be addressed in future versions of the code where modifications will be optimized. Method 4 is the most promising, but by making modifications to it, or creating a new method that goes about the process more efficiently with respect to time and number of steps, a stronger result could be achieved.

References

[1] Katyanna Quach 6 Nov 2017 at 06:14, "How we fooled Google's AI into thinking a 3D-printed turtle was a gun: MIT bods talk to El Reg." [Online]. Available: https://www.theregister.co.uk/2017/11/06/mit_fooling_ai/. [Accessed: 05-May-2018].

[2] A. Geitgey, "Machine Learning is Fun Part 8: How to Intentionally Trick Neural Networks," *Medium*, 16-Aug-2017.

[3] Brendel, Wieland, et al. "Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models." International Conference on Learning Representations, 2018, arxiv.org/pdf/1712.04248.pdf.