Adversarial Attacks on Image Classifiers

Brayden Goldstein-Gelb
Algorithmic Aspects of Machine Learning
Brown University
Providence, RI 02912
`brayden_goldstein-gelb@brown.edu`

Abstract

Image classifiers have become increasingly important. From self-driving cars to search features on smartphones, machine learning systems are used nearly everywhere in modern life. However, it has also been shown that image classifiers and machine learning systems in general can be fooled with minimal changes to inputs. In this project, I create a printable sticker that will change the classification of an object to a desired label. The sticker can be placed anywhere in the scene and fools multiple classifiers despite only being trained on ResNet50.

1 Introduction

1.1 Previous Work

This work builds off of various papers in the field. Goodfellow, Shlens and Szegedy [2] published a major work developing the methodology behind creating and explaining adversarial examples in machine learning. They describe how to create small perturbations in an image using the Fast Gradient Sign Method (FGSM) that will cause an image classifer to mislabel the image.

Brown et. al [1] expand on this idea by removing the restriction that the perturbation must be imperceptible and develop the idea of an adversarial patch that can be placed in a scene to change the classification of an object to a target label. They produce a single sticker and show how it can be used to trick VGG16 into classifying a banana as a toaster. The goal of this project is to further build upon this idea by producing several adversarial stickers and testing their efficacy across various real-world objects, backgrounds, and lighting conditions.

1.2 Problem Description and Challenges

Research into adversarial attacks on image classifiers have largely focused on perturbations to a specific image that will change the classification of the image. However, this requires the image to be specified in advance and the optimization process to be run on this specific image. Instead, this method attempts to create a sticker that can be placed in a wide variety of scenes and will change the classification of an object in the scene to that of the desired label, which we call the *target class*. The sticker is easily added to a scene simply by printing it out and placing it into the scene. This allows the stickers to alter the classification of an object without being noticed.

In addition to the task of creating adversarial examples, we must also account for the following physical challenges:

- Position, rotation and scale of the sticker
- Effects of scene lighting on the sticker
- Effects of camera quality, dynamic range, and noise on the sticker

- Printer quality

Where a simple perturbation of an image can change pixels of the image in the way that most effectively changes the classification, the sticker is much more limited. Because it must be a physical object that is printed and placed in the scene, it must be robust to variation in the scene as well as be reproduced by a printer.

2 Approach

The algorithm for generating an adversarial sticker begins by initializing the 50×50 pixel patch with random RGB values. Then the following two main steps are repeated for a set number of iterations.

1. Apply the sticker to an image

2. Run a step of the optimizer

Step 1 uses the `overlay_sticker` function to overlay the sticker over `batch_size` random backgroundimages from the imagenet dataset [3]. Before being overlayed, the position, scale and rotation of the sticker are set to random values within appropriate ranges. While the position can be set to anywhere in the image, the rotation and scale are only changed slightly to avoid too much variation while training.

Then, we account for the physical constraints of both printing and photographing the sticker. To account for the dynamic range of the camera and printer, we interpolate between the darkest and lightest values in the background image. For a given image I with RGB pixel values p, we consider the intensity, i, of a given pixel to be $\frac{1}{3}(p_r + p_g + p_b)$, where p_r, p_g, p_b are the red, green and blue channels of the pixel. So let $i_{min} = \min_{p \in I} \frac{1}{3}(p_r + p_g + p_b)$ and $i_{max} = \min_{p \in I} \frac{1}{3}(p_r + p_g + p_b)$. Then, for a given pixel in the sticker, s, we set $s' = \frac{1}{255} s(i_{max} - i_{min}) + i_{min}$. This adjusts the sticker to more closely match the dynamic range of the camera. In practice, we may also multiply the values i_min and i_max by constants to account for noise and other anomalies in the image, as well as the dynamic range of the printer.

Finally, a small amount of gaussian noise is added to the entire scene after the sticker has been applied to help account for noise present in images captured of the sticker by a physical camera.

Step 2 uses ResNet50 to predict the label of the image once the sticker has been overlayed. Then the gradients are calculated by calculating Sparse Categorical Cross Entropy between the predicted label and the target label. This is used by the optimizer to minimize the loss. For this task, we use the Adam optimizer for `1000` iterations.

3 Experiment and Results

3.1 The Experiment

To determine the efficacy of this method, four stickers were generated and printed in various sizes. Two of the stickers were trained such that the outputted class is `brown_bear` and the other two that the outputted class is `basketball`. In addition, we use two "control" stickers that are simply a picture of the target class. These stickers as well as the control stickers were printed in sizes of 2in, 3in, 4in, 6in and 8in.

I then photographed various objects both with and without the stickers next to them and reported the class outputted by ResNet50. In addition to ResNet50, which the stickers are trained on, I also tested the images on VGG16, to determine how well this technique generalizes to other classifiers.

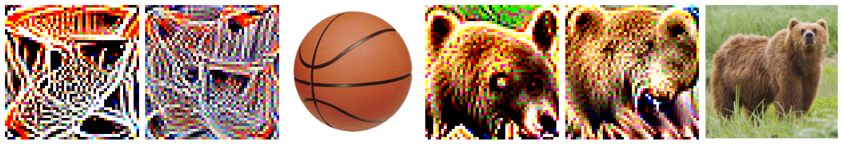

Figure 1: The final stickers used in the experiment from left to right: Basketball A, Basketball B, Basketball (Control), Brown Bear A, Brown Bear B, Brown Bear (Control)

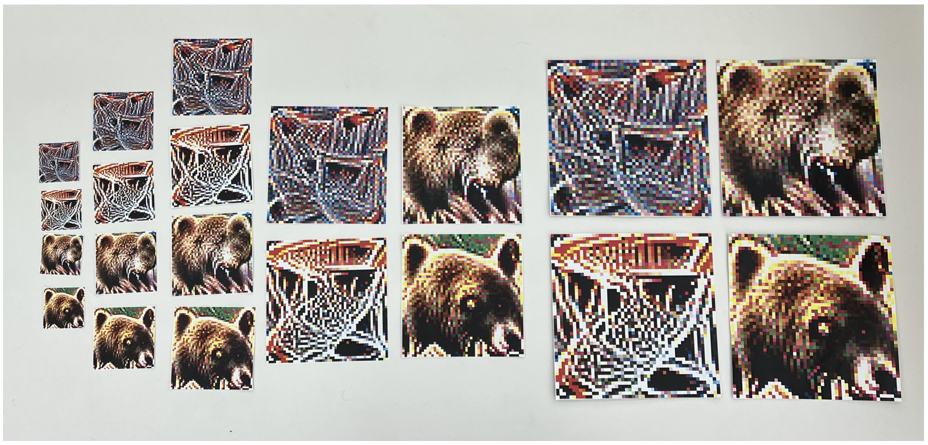
Figure 2: The stickers printed in various sizes

3.2 Results

First, we see how the size of the sticker affects classification. In the following results, a yellow highlight indicates that the object was incorrectly classified as the target label and an orange highlight indicates that the object was incorrectly classified as something else.

Table 1: Performance of stickers of various sizes on ResNet50 (original object: banana)

Sticker Size (in)	2		3		4	
	Label	Confidence	Label	Confidence	Label	Confidence
Basketball A	banana	99.97%	basketball	99.97%	basketball	82.87%
Basketball B	banana	99.62%	banana	91.39%	basketball	90.10%
Basketball (Control)	banana	98.55%	banana	97.22%	banana	83.03%
Brown Bear A	banana	99.47%	brown_bear	89.44%	brown_bear	97.78%
Brown Bear B	banana	97.97%	banana	81.91%	brown_bear	88.77%
Brown Bear (Control)	banana	98.81%	banana	99.99%	banana	99.96%

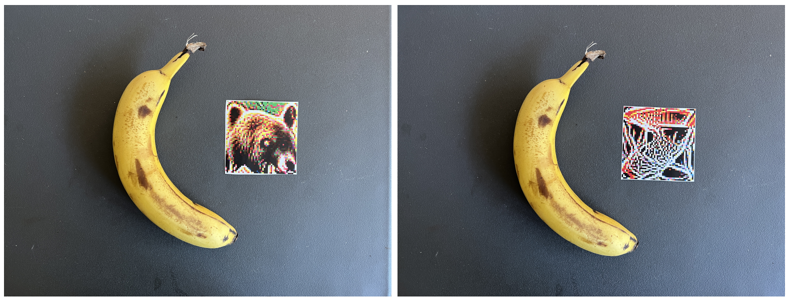
Figure 3: A banana is classified as a brown bear (left) and a basketball (right) when the 3in sticker is placed next to it

In Table 1 1, we can see that larger stickers are more likely to fool the image classifiers, as one would expect. Depending on the sticker, it only takes a 3 inch patch to fool the classifer. The control stickers do not seem to have any effect on the classification for the selected sizes shown.

Next, we test how the placement of the sticker affects the classification. In this experiment, we place the sticker in two different locations, either in the "Side" configuration to the left of the keyboard or in the "Over" configuration on top of the keyboard.

Table 2: Effect of placement of sticker in scene on ResNet50 (original object: keyboard)

Sticker Size (in)	4		6	
Sticker Placement	Side	Over	Side	Over
Basketball A	computer_keyboard, 49.75%	basketball, 78.81%	basketball, 94.54%	basketball, 76.94%
Basketball B	computer_keyboard, 92.08%	computer_keyboard, 77.81%	computer_keyboard, 49.12%	computer_keyboard, 84.20%
Basketball (Control)	computer_keyboard, 75.58%	computer_keyboard, 68.90%	basketball, 40.47%	basketball, 74.71%
Brown Bear A	computer_keyboard, 52.18%	computer_keyboard, 68.61%	computer_keyboard, 85.06%	computer_keyboard, 95.31%
Brown Bear B	computer_keyboard, 96.83%	computer_keyboard, 97.93%	computer_keyboard, 84.95%	computer_keyboard, 96.38%
Brown Bear (Control)	computer_keyboard, 87.92%	computer_keyboard, 94.25%	computer_keyboard, 79.99%	computer_keyboard, 81.81%

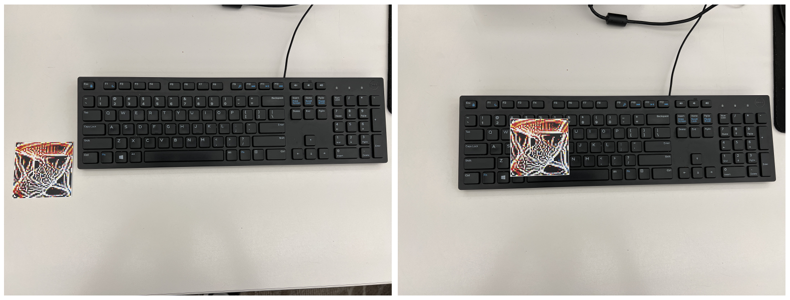
Figure 4: Placement of sticker is either "Side" (left) or "Over" (right) configurations

In this case, the brown bear sticker did not perform well at all. However, the basketball sticker was able to fool the classifer. Based on the data, we can see that placing the sticker on top of the object in the scene has a greater effect than placing the sticker next to the object. In this case, a smaller basketball sticker was able to fool the classifier when in the "Over" configuration as opposed to the "Side" configuration, which required a larger sticker.

In addition to ResNet50, the model on which the stickers were trained to fool, we also used VGG16 to classify the images from the experiment.

Table 3: Performance of stickers of various sizes on VGG16 (original object: banana)

Sticker Size (in)	2		3		4	
	Label	Confidence	Label	Confidence	Label	Confidence
Basketball A	banana	81.30%	ocarina	19.81%	banana	72.88%
Basketball B	banana	73.42%	banana	14.15%	shopping_cart	55.19%
Basketball (Control)	banana	72.63%	banana	53.49%	banana	55.64%
Brown Bear A	banana	70.70%	Norwich_terrier	47.24%	brown_bear	66.08%
Brown Bear B	banana	45.46%	banana	77.97%	eel	70.27%
Brown Bear (Control)	banana	82.83%	banana	94.08%	banana	67.27%

Table 4: Effect of placement of sticker in scene on VGG16 (original object: keyboard)

Sticker Size (in)	4		6	
Sticker Placement	Side	Over	Side	Over
Basketball A	computer_keyboard, 66.38%	buckle, 31.83%	buckle, 42.51%	buckle, 58.26%
Basketball B	computer_keyboard, 78.13%	computer_keyboard, 45%	computer_keyboard, 44.33%	computer_keyboard, 47.09%
Basketball (Control)	computer_keyboard, 76.41%	computer_keyboard, 72.64%	computer_keyboard, 57.19%	computer_keyboard, 26.83%
Brown Bear A	computer_keyboard, 58.43%	Norwich_terrier, 62.78%	brown_bear, 78.27%	brown_bear, 48.51%
Brown Bear B	computer_keyboard, 49.41%	remote_control, 20.92%	computer_keyboard, 67.8%	brown_bear, 85.59%
Brown Bear (Control)	computer_keyboard, 56.46%	computer_keyboard, 76.12%	computer_keyboard, 62.65%	computer_keyboard, 31.76%

The data in these tables indicate that the stickers are able to be generalized to other classifiers such as VGG16. While the images were less frequently classified as the target label, they were still very frequently classified as a class other than that of the original image. This demonstrates the power of the stickers still to have a major effect on the predictions of the model despite not being trained on it.

Finally, we put the stickers to the test on a more challenging scenario, one with the most potential for harm. We placed the stickers on a stop sign (and quickly removed them before any cars came) to see how they perform in natural light.

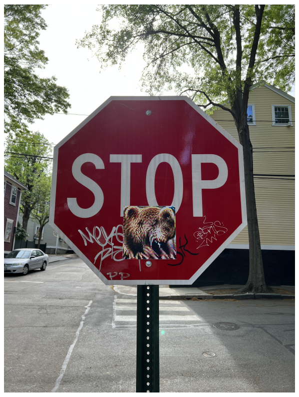

Figure 5: Due to challenging outdoor lighting conditions, this image was classified as a street sign by both ResNet50 and VGG16.

Table 5: Performance of stickers in natural light (Original object: stop sign)

Model	ResNet50		VGG16	
Sticker Size (in)	6	8	6	8
Basketball A	street_sign, 92.56%	street_sign, 92.26%	birdhouse, 85.52%	birdhouse, 60.92%
Basketball B	street_sign, 96.61%	street_sign, 90.91%	birdhouse, 76.76%	street_sign, 71.84%
Basketball (Control)	street_sign, 97.29%	street_sign, 84.56%	street_sign, 51.63%	street_sign, 65.66%
Brown Bear A	street_sign, 81.06%	street_sign, 93.63%	birdhouse, 92.00%	street_sign, 62.94%
Brown Bear B	street_sign, 85.25%	street_sign, 98.04%	birdhouse, 51.69%	streetsign, 50.58%
Brown Bear (Control)	street_sign, 97.45%	street_sign, 96.80%	birdhouse, 83.94%	birdhouse, 64.14%

In this example, the stickers were unable to trick either classifier. While VGG16 classifies the stop sign as a birdhouse with the stickers, it also does so with 56% confidence when the sticker is not present. In Figure 5, we can see that the lighting conditions change the appearance of the sticker. Clearly, this is enough of an effect to prevent the sticker from changing the classification of the image.

4 Broader Impact

This technology has the potential for real harm if they are used in malicious ways. While the sticker in this project was unable to change the label assigned to a stop sign, a more advanced sticker could theoretically trick the classifiers used in a self-driving car into misclassifying traffic signs and other important objects. If a stop sign or, worse, a pedestrian is classified incorrectly, lives would be at risk. This calls for more work to be done on ensuring the robustness of machine learning models in the field of image classification.

5 Future Work

There are various ways in which this approach could be improved.

- The patch used in this example was always square and limited to a size of 50×50 pixels. Future work could experiment with different shapes and resolutions of stickers to determine what is the most effective.
- To account for lighting effects, future work could include simple simulations of various lighting conditions during training. Perhaps it could even have the lighting match the background image that the sticker is being trained on at each step in training.
- The stickers produced by this approach tend to have a strong resemblance to the target label, which makes them easy to identify by humans. Future work could find ways to camouflage the stickers with the background or potentially add a second classifier to reduce the similarity of the sticker and the target label.

- To increase the capacity of the stickers to generalize across classifiers, multiple classifiers, potentially with very different architectures could be used during training. The Adversarial Patch paper proposes and ensemble method, which could definitly be expanded upon.
- Recent works have also produced models that are more resistant to adversarial examples. However, adversarial stickers could also be trained on these robust models. Further work could look into how adversarial stickers interact with these more robust models.

6 Data and Code

The code and raw data can be accessed on GitHub. In addition to the code used to generate the sticker, it has a spreadsheet containing the full results of the experiments and the images used to produce those results.

References

[1] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *CoRR*, abs/1712.09665, 2017.

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

[3] 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 (IJCV)*, 115(3):211–252, 2015.