Fooling automated surveillance cameras:
adversarial patches to attack person detection

Simen Thys∗
simen.thys@student.kuleuven.be

Wiebe Van Ranst∗
wiebe.vanranst@kuleuven.be

Toon Goedemé
toon.goedeme@kuleuven.be

KU Leuven
EAVISE, Technology Campus De Nayer, KU Leuven, Belgium.
∗ Authors contributed equally to this paper.

Abstract

Adversarial attacks on machine learning models have seen increasing interest in the past years. By making only subtle changes to the input of a convolutional neural network, the output of the network can be swayed to output a completely different result. The first attacks did this by changing pixel values of an input image slightly to fool a classifier to output the wrong class. Other approaches have tried to learn “patches” that can be applied to an object to fool detectors and classifiers. Some of these approaches have also shown that these attacks are feasible in the real-world, i.e. by modifying an object and filming it with a video camera. However, all of these approaches target classes that contain almost no intra-class variety (e.g. stop signs). The known structure of the object is then used to generate an adversarial patch on top of it.

In this paper, we present an approach to generate adversarial patches to targets with lots of intra-class variety, namely persons. The goal is to generate a patch that is able successfully hide a person from a person detector. An attack that could for instance be used maliciously to circumvent surveillance systems, intruders can sneak around undetected by holding a small cardboard plate in front of their body aimed towards the surveillance camera.

From our results we can see that our system is able significantly lower the accuracy of a person detector. Our approach also functions well in real-life scenarios where the patch is filmed by a camera. To the best of our knowledge we are the first to attempt this kind of attack on targets with a high level of intra-class variety like persons.

1. Introduction

The rise of Convolutional Neural Networks (CNNs) has seen huge successes in the field of computer vision. The data-driven end-to-end pipeline in which CNNs learn on images has proven to get the best results in a wide range of computer vision tasks. Due to the depth of these architectures, neural networks are able to learn very basic filters at the bottom of the network (where the data comes in) to very abstract high level features at the top. To do this, a typical CNN contains millions of learned parameters. While this approach results in very accurate models, the interpretability decreases dramatically. Understanding exactly why a network classifies an image of a person as a
person is very hard. The network has learned what a person looks likes by looking at many pictures of other persons. By evaluating the model we can determine how well the model work for person detection by comparing it to human annotated images. Evaluating the model in such a way however only tells us how well a detector performs on a certain test set. This test set does not typically contain examples that are designed to steer the model in the wrong way, nor does it contains examples that are especially targeted to fool the model. This is fine for applications where attacks are unlikely such as for instance fall detection for elderly people, but can pose a real issue in for instance security systems. A vulnerability in the person detection model of a security system might be used to circumvent a surveillance camera that is used for break in prevention in a building.

In this paper we highlight the risks of such an attack on person detection systems. We create a small (around 40cm × 40cm) “adversarial patch” that is used as a cloaking device to hide people from object detectors. A demonstration of this is shown in Figure 1.

The rest of this paper is structured as follows: Section 2 goes over the related work on adversarial attacks. Section 3 discusses how we generate these patches. In Section 4 we evaluate our patch both quantitatively on the Inria dataset, and qualitatively on real-life video footage taken while holding a patch. We reach a conclusion in Section 5.

Source code is available at: https://gitlab.com/EAVISE/adversarial-yolo

2. Related work

With the rise in popularity of CNNs, adversarial attacks on CNNs have seen an increase in popularity in the past years. In this section we go over the history of these kind of attacks. We first talk about digital attacks on classifiers, then talk about real-world attacks both for face recognition and object detection. Then we briefly discuss the object detector, YOLOv2 that in this work is the target of our attacks.

Adversarial attacks on classification tasks Back in 2014 Bigio et at. [2] showed the existence of adversarial attacks. After that, Szegedy et al. [19] succeeded in generating adversarial attacks for classification models. They use a method that is able to fool the network to miss-classify an image, while only changing the pixel values of the image slightly so that the change is not visible to the human eye. Following that, Goodfellow et al. [9] create a faster gradient sign method that made it more practical (faster) to generate adversarial attacks on images. Instead of finding the most optimal image as in [19], they find a single image in a larger set of images that is able to do an attack on the network. In [14], Moosavi-Dezfooli et al. present an algorithm that is able generate an attack by changing the image less and is also faster than the previous. They use hyperplanes to model the border between different output classes to the input image. Carlini et al. [4] present another adversarial attack, again, using optimisation methods, they improve in both accuracy and difference in images (using different norms) compared to the already mentioned attacks. In [3] Brown et at. create a method that, instead of changing pixel values, generates patches that can be digitally placed on the image to fool a classifier. Instead of using one image, they use a variety of images to build in intra-class robustness. In [8] Evtimov et al. present a real-world attack for classification. They target the task of stop sign classification which proves to be challenging due to the different poses in which stop signs can occur. They generate a sticker than can be applied to a stop sign to make it unrecognizable. Athalye et al. [1] present an approach in which the texture of a 3D model is optimized. Images of different poses are shown to the optimizer to build in robustness to different poses and lighting changes. The resulting object was then printed using a 3D printer. The work of Moosavi-Dezfooli [13] presents an approach to generate a single universal image that can be used as an adversarial perturbation on different images. The universal adversarial image is also shown to be robust to different detectors.

Real-world adversarial attack for face recognition An example of real-world adversarial attack is presented in [17]. Sharif et al. demonstrate the use of printed eyeglasses that can be used to fool facial recognition systems. To guarantee robustness the glasses need to work on a wide variety of different poses. To do this, they optimize the print on the glasses in such a way that they work on a large set of images instead of just a single image. They also include a Non Printability Score (NPS) which makes sure that the colors used in the image can be represented by a printer.

Real-world adversarial attacks for object detection Chen et al. [5] present a real-world attack for object detection. They target the detection of stop signs in the Faster R-CNN detector [16]. Like [1], they use the concept of Expectation over Transformation (EOT) (doing various transformation on the image) to build in robustness against different poses. The most recent work we found to fool object detectors in the real-world is the work of Eykholt et al [18]. In it, they again target stop signs and use the YOLOv2 [15] detector to do a white box attack, where they fill in a pattern in the entire red area of the stop sign. They also evaluate on Faster-RCNN where they found that their attack also transfers to other detectors.

Compared to this work all attacks against object detectors focus on objects with fixed visual patterns like traffic signs and do not take into account intra-class variety. To the best of our knowledge no previous work has proposed
Figure 2: Overview of the YOLOv2 architecture. The
detector outputs an objectness score (how likely it is that
this detection contains an object), shown in the middle
top figure, and a class score (which class is in the
bounding box), shown in the middle bottom figure. Image source:

3. Generating adversarial patches against person
detectors

The goal of this work is to create a system that is able to
generate printable adversarial patches that can be used to
tool person detectors. As discussed earlier, Chen et al. [5]
and Eykholt et al. [18] already showed that adversarial at-
tacks on object detectors in the real-world are possible. In
their work they target stop signs, in this work we focus on
persons which, unlike the uniform appearance of stop signs
can vary a lot more. Using an optimisation process (on the
image pixels) we try to find a patch that, on a large dataset,
effectively lowers the accuracy of person detection. In this
section, we explain our process of generating these adver-
sarial patches in depth.

Our optimisation goal consists of three parts:

- \( L_{\text{nps}} \) The non-printability score [17], a factor that re-
  presents how well the colours in our patch can be repre-
  sented by a common printer. Given by:

\[
L_{\text{nps}} = \sum_{p_{\text{patch}} \in P} \min_{c_{\text{print}} \in C} |p_{\text{patch}} - c_{\text{print}}|
\]

Where \( p_{\text{patch}} \) is a pixel in of our patch \( P \) and \( c_{\text{print}} \) is a
colour in a set of printable colours \( C \). This loss favours
colors in our image that lie closely to colours in our set of
printable colours.

- \( L_{\text{tv}} \) The total variation in the image as described in [17]. This loss makes sure that our optimiser favours
  an image with smooth colour transitions and prevents noisy images. We can calculate \( L_{\text{tv}} \) from a patch \( P \) as follows:

\[
L_{\text{tv}} = \sum_{i,j} \sqrt{\left( (p_{i,j} - p_{i+1,j})^2 + (p_{i,j} - p_{i,j+1})^2 \right)}
\]

The score is low if neighbouring pixels are similar, and high if neighbouring pixel are different.

- \( L_{\text{obj}} \) The maximum objectness score in the image. The
goal of our patch is to hide persons in the image. To do
this, the goal of our training is to minimize the object
or class score outputted by the detector. This score will
be discussed in depth later in this section.

Out of these three parts follows our total loss function:

\[
L = \alpha L_{\text{nps}} + \beta L_{\text{tv}} + L_{\text{obj}}
\]

We take the sum of the three losses scaled by factors \( \alpha \) and \( \beta \) which are determined empirically, and optimise using the

The goal of our optimizer is to minimise the total loss
\( L \). During the optimisation process we freeze all weights in
the network, and change only the values in the patch. The
patch is initialised on random values at the beginning of the
process.

Figure 3 gives an overview of how the object loss is cal-
culated. The same procedure is followed to calculate the
class probability. In the remaining parts of this section we
will explain how this is done in depth.
3.1. Minimizing probability in the output of the detector

As was explained in Section 2, the YOLOv2 object detector outputs a grid of cells each containing a series of anchor points (five by default). Each anchor point contains the position of the bounding box, an object probability and a class score. To get the detector to ignore persons we experimented with three different approaches: We can either minimize the classification probability of class person (example patch in Figure 4d), minimize the objectness score (Figure 4c), or a combination of both (Figures 4b and 4a). We tried out all approaches. Minimizing the class score has a tendency to switch the class person over to a different class. In our experiments with the YOLO detector trained on the MS COCO dataset [11], we found that the generated patch is detected as another class in the COCO dataset. Figure 4a and 4b is an example of taking the product of class and object probability, in the case of Figure 4a, the learned patch ended up resembling a teddy bear, which it visually also resembles. The class “teddy bear” seemed to overpower the class “person”. Because the patch starts to resemble another class however, the patch is less transferable to other models trained on datasets which do not contain the class.

The other approach we propose of minimizing the objectness score does not have this issue. Although we only put it on top of people during the optimisation process, the resulting patch is less specific for a certain class than the other approach. Figure 4c shows an example of such a patch.

3.2. Preparing training data

Compared to previous work done on stop signs [5, 18], creating adversarial patches for the class persons is much more challenging:

- The appearance of people varies much more: clothes, skin color, sizes, poses... Compared to stop signs which always have the same octagonal shape, and are usually red.
- People can appear in many different contexts. Stop signs mostly appear in the same context at the side of a street.
- The appearance of a person will be different depend-
ing on whether a person is facing away or towards the camera.

- There is no consistent spot on a person where we can put our patch. On a stop sign it’s easy to calculate the exact position of a patch.

In this section we will explain how we deal with these challenges. Firstly, instead of artificially modifying a single image of the target object and doing different transformations as was done in [5, 18], we use real images of different people. Our workflow is as follows: We first run the target person detector over our dataset of images. This yields bounding boxes that show where people occur in the image according to the detector. On a fixed position relative to these bounding boxes, we then apply the current version of our patch to the image under different transformations (which are explained in Section 3.3). The resulting image is then fed (in a batch together with other images) into the detector. We measure the score of the persons that are still detected, which we use to calculate a loss function. Using back propagation over the entire network, the optimiser then changes the pixels in the patch further in order to fool the detector even more.

An interesting side effect of this approach is that we are not limited to annotated datasets. Any video or image collection can be fed into the target detector to generate bounding boxes. This allows our system to also do more targeted attacks. When we have data available from the environment we are targeting we can simply use that footage to generate a patch specific to that scene. Which will presumably perform better than a generic dataset.

In our tests we use the images of the Inria [6] dataset. These images are targeted more towards full body pedestrians which are better suited for our surveillance camera application. We acknowledge that more challenging datasets like MS COCO [11] and Pascal VOC [7] are available, but they contain too much variety in which people occur (a hand is for instance annotated as person), making it hard to put our patch in a consistent position.

### 3.3. Making patches more robust

In this paper we target patches that have to be used in the real-world. This means that they are first printed out, and then filmed by a video camera. A lot of factors influence the appearance of the patch when you do this: The lighting can change, the patch may be rotated slightly, the size of the patch with respect to the person can change, the camera may add noise or blur the patch slightly, viewing angles might be different... To take this into account as much as possible, we do some transformations on the patch before applying it to the image. We do the following random transformations:

- The patch is rotated up to 20 degrees each way.
- There is no consistent spot on a person where we can put our patch. On a stop sign it’s easy to calculate the exact position of a patch.

In this section we evaluate the effectiveness of our patches. We evaluate our patches by applying them to the Inria test set using the same process we used during training, including random transformations. In our experiments we tried to minimise a few different parameters that have the potential to hide persons. As a control, we also compare our results to a patch containing random noise that was evaluated in the exact same way as our random patches. Figure 5 shows the result of our different patches. The objective in OBJ-CLS was to minimise the product of the object score and the class score, in OBJ only the object score, and in CLS only the class score. NOISE is our control patch of random noise, and CLEAN is the baseline with no patch applied. (Because the bounding boxes where generated by running the same detector over the dataset we get a perfect result.) From this PR-curve we can clearly see the impact a generated patch (OBJ-CLS, OBJ and CLS) has compared to a random patch which acts as a control. We can also see that minimising the object score (OBJ) has the biggest impact (lowest Average Precision (AP)) compared to using the class score.

A typical way to determine a good working point on a PR-curve to use for detection is to draw a diagonal line on

![Figure 5: PR-curve of our different approaches (OBJ-CLS, OBJ and CLS), compared to a random patch (NOISE) and the original images (CLEAN).](image-url)
Table 1: Comparison of different approaches in recall. How well do different approaches circumvent alarms?

<table>
<thead>
<tr>
<th>Approach</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLEAN</td>
<td>100</td>
</tr>
<tr>
<td>NOISE</td>
<td>87.14</td>
</tr>
<tr>
<td>OBJ-CLS</td>
<td>39.31</td>
</tr>
<tr>
<td>OBJ</td>
<td>26.46</td>
</tr>
<tr>
<td>CLS</td>
<td>77.58</td>
</tr>
</tbody>
</table>

5. Conclusion

In this paper, we presented a system to generate adversarial patches for person detectors that can be printed out and used in the real-world. We did this by optimising an image to minimise different probabilities related to the appearance of a person in the output of the detector. In our experiments we compared different approaches and found that minimising object loss created the most effective patches.

From our real-world test with printed out patches we can also see that our patches work quite well in hiding persons from object detectors, suggesting that security systems using similar detectors might be vulnerable to this kind of attack.

We believe that, if we combine this technique with a sophisticated clothing simulation, we can design a T-shirt print that can make a person virtually invisible for automatic surveillance cameras (using the YOLO detector).

6. Future work

In the future we would like to extend this work by making it more robust. One way to do this is by doing more (affine) transformation on the input data or using simulated data (i.e. apply the patch as a texture on a 3D-model of a person). Another area where more work can be done is transferability. Our current patches do not transfer well to
completely different architectures like Faster R-CNN [16], optimising for different architectures at the same time might improve upon this.

References