**Physical /Patch**

Physical adversarial attacks are directly deployed on the object in the real word by adding patches or stickers on or near the targeted object that will cause the model to “misclassify” the object. In contrast to adding imperceptible perturbation to the image , patches are salient yet often innocuous looking materials . They are created by maximising the loss function of the network. Since they are often kept in different locations and orientations in the proximity of the targeted object they essentially becomes the prominent feature of the image resulting the targeted object to be treated like a background [AP]

The misclassification was proved to be not a result of overfitting by [IPN] using **cross-training-set generalisation**: A model trained on a completely different training set than the one on which the model used to generate the adversarial samples was trained, misclassified the images ; **cross model generalisation:** A model trained from scratch with different hyperparameters than the model used to generate the adversarial samples, misclassified the images

There are several ways patches can be adversarial according to [SIB]:

* Objectness Based Attacks
* Classification Based attacks

Objectness Based Attacks:

This is achieved by Feature Interference Reinforcement (FIR) and Enhanced Realistic Constraint Generation (ERCS).

FIR is a white box technique which works by causing the adversarial example to perturb the earlier hidden layers of the model . In order to achieve this , the model is trained with a ‘normal’ and an ‘adversarial’ versions of the same image. At each layer, a feature vector of the targeted object in both the images are created. One feature value is extracted from one feature map using mean pooling. The loss function is defined as a difference between the feature vectors and the goal is to maximise this loss function.

ERCS works by creating adversarial examples subject to realistic constraints . These constraints ensures that the relationship of the object to its background and the object semantic integrity is maintained in the creation of adversarial examples.

Classification Based Attacks:

Nested AE is based on the hypothesis that the part of an object detection model like YOLOV3 that detects objects at small scale (at long distance) is easier to be ‘fooled’ as they only consider a small number of pixels. In this method we target the central area of the object in short distance range , add perturbation only to that area as once the object is in a long distance (hence occupying a small change in the image) , the whole object would occupy only the aforementioned central area.

[AP] : Adversarial Patch : <https://arxiv.org/abs/1712.09665>

[IPN]: Intriguing properties of neural networks : <https://arxiv.org/pdf/1312.6199>

[SIB]: Seeing isn’t Believing: Towards More Robust Adversarial Attack Against Real World Object Detectors : <https://sci-hub.se/https:/doi.org/10.1145/3319535.3354259>

**Mathematical Intuition:**

In order to understand why small perturbations can cause an image to be misclassified , lets us consider an image and let represent the adversarial example created by adding a small perturbation to

The smallest change we can make in pixel value that can be identified by a model is 1 unit. So both pixel values 123.75 and 124 will be classified as 124. This would mean that the model shouldn’t classify and differently, for , where is small enough to be neglected by the model.

This is the point where the effect of dimensionality comes in . To understand it, consider the dot product of the adversarial example with the weight vector  **:**

We can see that the result of the perturbation is the additional component of . If subject to the max norm constraint , and the weight vector has dimensions and the average magnitude of an element is , then the value of this additional component is .

Thus with an increase in the dimension , the effect of the increases linearly . This can cause even small perturbations to have a dramatic effect for inputs with large dimensions.

**Algorithms**

(Fast Gradient Sign Method) FGSM:

Consider an input image , the true label of the image is . Let be the parameters of the network. Then loss function is and the gradient of the loss function is . This tells us the rate of change of the loss function with respect to the input image. In other words, it tells us how does changing the value of the of image pixels affect the loss function.

In gradient descent algorithm, we change the weights in the direction opposite to the direction of the loss function (with respect to the weights) to decrease the loss function. Therefore, while building an adversarial example, we do the opposite , that is we change the pixel value of the image in the direction of the loss function which subsequently causes the loss function to increase.

Therefore, the update rule for the image pixels is :